32 位 MCU
ES32F36xx

参 考 手 册

☐ 产品简介

☐ 数据手册

☑ 参考手册

上海东软载波微电子有限公司

2021-05-19

目 录

内容目录

图目录

表目录

第1章 文档约定

1.1 寄存器读写权限的设定

缩写词	说明	描述
R/W	读/写（__IO）	软件可以读写这些位
R	只读（__I）	软件只能读取这些位
W	只写（__O）	软件只能写入该位，读取该位时将返回复位值
W1	只写（写 1）	软件只能写入该位，写 1 有效，写 0 无作用。
R/C_W1	读取/清零（写 1）	软件可以读取该位，也可以通过写入 1 将该位清零。写入“0”对该位的值无影响
R/C_W0	读取/清零（写 0）	软件可以读取该位，也可以通过写入 0 将该位清零。写入“1”对该位的值无影响
R/C_R	读取/清零（读取）	软件可以读取该位。读取该位时，将自动清零。写入“0”对该位的值无影响
C_W1	清零（写 1）	通过写入 1 将该位清零。写入“0”对该位的值无影响
S_W1	置位（写 1）	通过写入 1 将该位置位。写入“0”对该位的值无影响
C_W0	清零（写 0）	通过写入 0 将该位清零。写入“1”对该位的值无影响
T_W1	触发（写 1）	通过写入 1 将触发硬件动作。写入“0”对该位的值无影响
Reserved	保留	保留位，必须保持复位值。

第2章 系统概述

2.1 概述

该章节从系统层描述 ES32F36xx 系列微控制器所涵盖的功能。

2.2 系统框图

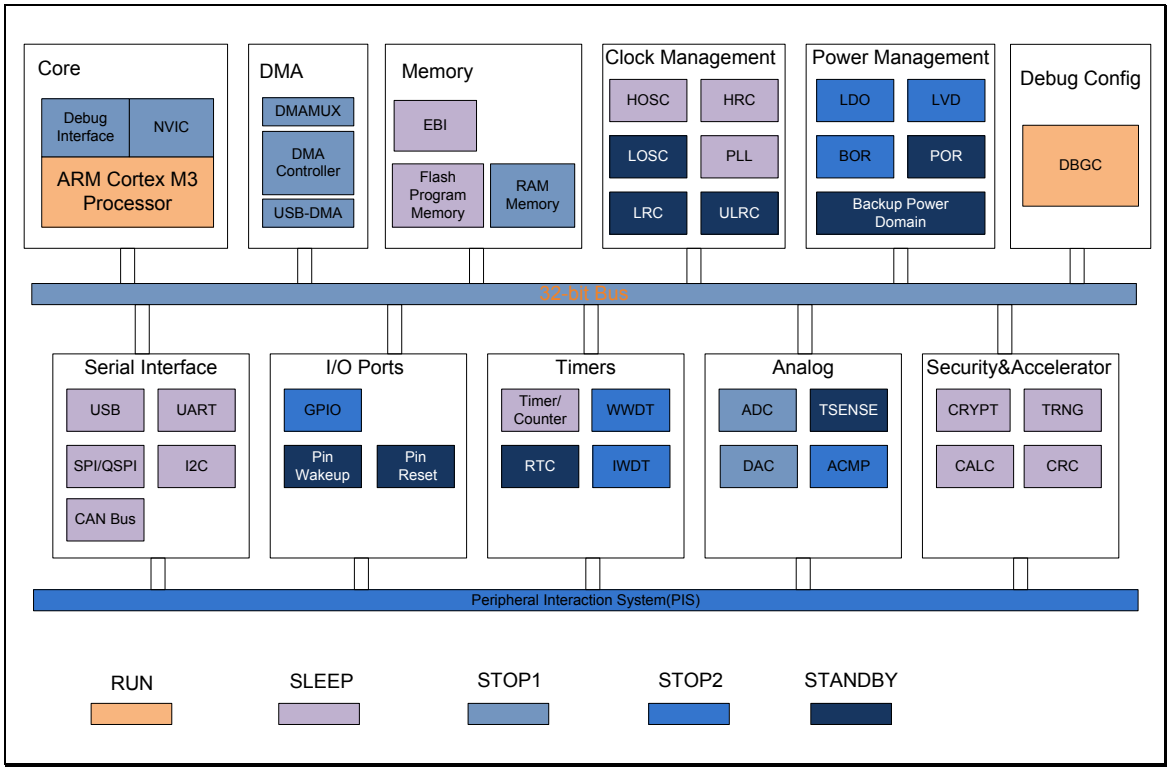


图 2-1 系统框图

2.3 模块功能类别

类别	描述
ARM 32 位 Cortex-M3 CPU	◆ ARM Cortex M 系列 32 位 MCU 内核，最高频率可达 96MHz ◆ MPU ◆ 调试 ◆ 中断和事件
存储	◆ 内部存储： ◇ Flash 存储器 ◇ SRAM 存储器 ◆ 支持外部扩展： ◇ EBI 可灵活扩展各种外部存储 ◇ QSPI 可支持扩展 SPI Flash 存储 ◆ 系统总线和存储器 ◆ 存储器系统控制
系统管理	◆ 系统配置控制器 ◆ 电源管理及低功耗模式 ◇ 可支持多种低功耗模式： - SLEEP 模式 - STOP1/STOP2 模式 - STANDBY 模式 ◆ 复位控制 ◆ DMA ◇ 支持多个 DMA 请求通道，DMAMUX 为每个 DMA 通道提供片上 DMA 请求源选择 ◆ 外设互联（PIS） ◆ 看门狗定时器 ◇ 独立看门狗定时器 ◇ 窗口看门狗定时器 ◆ 调试控制（DBGC）
时钟管理	◆ 提供外部和内部多种时钟源选择 ◇ HOSC ◇ LOSC ◇ HRC ◇ LRC ◇ ULRC ◇ PLL ◆ 外部时钟停振检测 ◆ PLL 倍频满足高速应用 ◆ 可灵活选择内核时钟，系统及外设时钟 ◆ 可灵活设置外设时钟门控及时钟分频，以满足低功耗应用需求

类别	描述
外部接口	◆ 通用端口（GPIO）
安全管理及运算加速	◆ 循环冗余校验模块（CRC） ◆ 加密处理（CRYPT） ◆ 真随机数发生器（TRNG） ◆ 运算加速器（CALC）
定时器	◆ 高级定时器（AD16C4T） ◆ 通用定时器（GP32C4T、GP16C4T） ◆ 基本定时器（BS16T） ◆ 实时时钟（RTC）
通信	◆ 内部集成电路总线（I2C） ◆ 串行外设接口（SPI） ◆ 通用异步收发器（UART） ◆ 控制区域网络（CAN） ◆ 通用串行总线接口（USB）
模拟	◆ 模数转换（ADC） ◆ 数模转换（DAC） ◆ 模拟比较器（ACMP） ◆ 温度传感器（TSENSE） ◆ 内部参考源（AVREF）

表 2-1 系统功能模块

2. 3. 1 ARM 32 位Cortex-M3 内核模块

ES32F36xx 系列微控制器内核模块包含以下功能：

模块	描述
ARM 32 位 Cortex-M3 内核	◆ 支持 Thumb 指令集； ◆ 带 ICode，DCode 和 System 总线接口； ◆ 支持快速中断响应； ◆ 支持硬件除法指令 SDIV 和 UDIV； ◆ 支持位带操作
NVIC	◆ 中断使能控制； ◆ 中断优先级设置； ◆ 支持末尾连锁和迟来； ◆ 支持 32 个外部中断向量
WIC	◆ 中断唤醒控制模块
MPU	◆ 存储保护单元，检查访问权限和内存属性
PMU	◆ 内核功耗管理模块，请结合 ARM 技术参考手册和本文档的章节“电源管理及低功耗模式”和“备份域电源控制”阅读。
调试接口	◆ SWD 协议调试接口
SYSTICK	◆ 系统节拍定时器

表 2-2 ARM 32 位 Cortex-M3 内核模块

2.3.2 存储器及存储器接口

ES32F36xx 系列微控制器包含以下存储器及存储器接口模块：

模块	描述
系统总线和存储器	◆ 系统总线架构，存储器地址空间映射
存储器系统控制	◆ Flash 存储器的访问控制
Flash 存储	◆ Flash 存储
SRAM	◆ SRAM 存储
EBI	◆ 扩展总线接口，可扩展外部存储
QSPI	◆ QUAD SPI 接口可扩展 SPI Flash 器件

表 2-3 存储器及存储接口

2.3.3 系统模块

ES32F36xx 系列微控制器包含以下系统模块：

模块	描述
系统配置控制器（SYSCFG）	◆ 系统的相关配置
电源管理及低功耗模式（PMU）	◆ 系统电源的管理及低功耗模式控制
复位控制（RMU）	◆ 系统所有复位的管理
DMA 多路复用（DMAMUX）	◆ DMA 请求源的多路复用选择器
DMA 控制器（DMA）	◆ DMA 控制器可减少 CPU 负荷，提高系统效率； ◆ 在低功耗场合也可代替 CPU 的部分工作而不必唤醒整个系统，节省功耗。
外设互联（PIS）	◆ 外设互联系统为外设提供互联接口，可减少软件负担，提高了系统响应速度，同时为扩展应用场景提供了便利和灵活性。
看门狗定时器（IWDT，WWDT）	◆ 包含了独立看门狗和窗口看门狗。
调试控制（DBGC）	◆ 系统调试相关的配置控制 ◆ 产生断点时，定时器/WDT 是否继续计数可以配置； ◆ 对 PWM 输出口在调试断点时是否输出“三态”可进行配置； ◆ 可以分别配置在 SLEEP、STOP1/2 模式下调试时是否为 FCLK 和 HCLK 提供时钟

表 2-4 系统模块

2.3.4 时钟管理

ES32F36xx 系列微控制器包含以下时钟管理模块：

模块	描述
时钟管理（CMU）	◆ 时钟源配置 ◆ 系统和外设时钟的选择及切换 ◆ 外部时钟停振检测（时钟安全机制） ◆ 外设时钟门控 ◆ 系统和外设时钟分频 ◆ PLL 倍频

表 2-5 时钟管理

2.3.5 外部接口

ES32F36xx 微控制器包含以下外部接口模块：

模块	描述
通用端口及端口控制	◆ 通用端口的输入输出功能 ◆ 对端口的控制还包括：上、下拉选择，开漏选择，驱动能力选择，端口 CMOS/TTL 输入功能选择，端口模拟滤波器使能控制等 ◆ 通用端口支持 16 个中断源和 DMA 请求

表 2-6 外部接口

2.3.6 安全管理及运算加速

ES32F36xx 微控制器包含以下安全管理模块：

模块	描述
循环冗余校验模块（CRC）	◆ 支持四个常用的多项式： ◇ CRC-CCITT ◇ CRC-8 ◇ CRC-16 ◇ CRC-32
加密处理（CRYPT）	◆ 支持的标准有 AES、DES/TDES
真随机数发生器（TRNG）	◆ 可生产 1 位串行真随机数或 8/16/32 位并行真随机数
运算加速器（CALC）	◆ 用于执行平方根的运算加速

表 2-7 安全管理及运算加速

2.3.7 定时器

ES32F36xx 微控制器包含以下定时器模块：

模块	描述
高级定时器（AD16C4T）	◆ 16 位递增、递减、递增/递减自动重载计数器 ◆ 16 位可编程预分频器，用于对计数器时钟频率进行分

模块	描述
	频（即运行时修改），分频系数介于 1 到 65536 之间 ◆ 多达 4 个独立通道，可用于： ◇ 输入捕获 ◇ 输出比较 ◇ PWM 生成（边沿和中心对齐模式） ◇ 单脉冲模式输出 ◆ 带可编程死区的互补输出 ◆ 使用外部信号控制定时器且可实现多个定时器互联的同步电路 ◆ 重复计数器，用于仅在给定数目的计数器周期后更新定时器寄存器 ◆ 用于将定时器的输出信号置于复位状态或已知状态的刹车输入 ◆ 发生如下事件时生成中断/DMA 请求： ◇ 更新：计数器上溢/下溢、计数器初始化（通过软件或内部/外部触发） ◇ 触发事件（计数器启动、停止、初始化或通过内部/外部触发计数） ◇ 输入捕获 ◇ 输出比较 ◇ 刹车输入 ◆ 支持定位用增量（正交）编码及霍尔传感器电路
通用定时器（GP32C4T）	◆ 32 位递增、递减、递增/递减自动重载计数器 ◆ 16 位可编程预分频器，用于对计数器时钟频率进行分频（即运行时修改），分频系数介于 1 到 65536 之间 ◆ 多达 4 个独立通道，可用于： ◇ 输入捕获 ◇ 输出比较 ◇ PWM 生成（边沿和中心对齐模式） ◇ 单脉冲模式输出 ◆ 使用外部信号控制定时器且可实现多个定时器互联的同步电路 ◆ 发生如下事件时生成中断/DMA 请求： ◇ 更新：计数器上溢、计数器初始化（通过软件或内部/外部触发） ◇ 触发事件（计数器启动、停止、初始化或通过

模块	描述
	内部/外部触发计数） ◇ 输入捕获 ◇ 输出比较 ◆ 支持定位用增量（正交）编码器和霍尔传感器电路 ◆ 外部时钟触发输入
通用定时器（GP16C4T）	◆ 16 位递增、递减、递增/递减自动重载计数器 ◆ 16 位可编程预分频器，用于对计数器时钟频率进行分频（即运行时修改），分频系数介于 1 到 65536 之间 ◆ 多达 4 个独立通道，可用于： ◇ 输入捕获 ◇ 输出比较 ◇ PWM 生成（边沿和中心对齐模式） ◇ 单脉冲模式输出 ◆ 使用外部信号控制定时器且可实现多个定时器互联的同步电路 ◆ 发生如下事件时生成中断/DMA 请求： ◇ 更新：计数器上溢/下溢、计数器初始化（通过软件或内部/外部触发） ◇ 触发事件（计数器启动、停止、初始化或通过内部/外部触发计数） ◇ 输入捕获 ◇ 输出比较 ◆ 支持定位用增量（正交）编码器和霍尔传感器电路 ◆ 外部时钟触发输入
基本定时器（BS16T）	◆ 16 位递增自动重载计数器。 ◆ 16 位可编程预分频器，用于对计数器时钟频率进行分频（即运行时修改），分频系数介于 1 到 65536 之间 ◆ 发生如下事件时生成中断/DMA 请求： ◇ 更新：计数器上溢
实时时钟（RTC）	◆ 时间计数（实现小时、分钟、秒和亚秒）和日历计数（实现年、月、日和星期），采用 BCD 格式 ◆ 闹钟可输出 ◆ 闹钟支持掩码功能 ◆ 支持时间戳功能 ◆ 发生入侵检测事件时，将复位备份寄存器 ◆ 周期性唤醒，由 16 位可编程自动重载递减计数器生成

表 2-8 定时器

2.3.8 通信模块

ES32F36xx 微控制器包含以下通信模块：

模块	描述
内部集成电路总线（I2C）	◆ 支持多主模式和总线仲裁 ◆ 可编程 I2C 地址检测 ◆ 最高通信速率为 1MHz ◆ 可配置时钟延长 ◆ 16 级深度 FIFO ◆ 兼容 SMBus2.0 协议 ◆ 兼容 PMBus
串行外设接口（SPI）	◆ 支持半双工/全双工的同步串行通信 ◆ 主模式或从模式操作 ◆ 8 位或 16 位传输帧格式选择 ◆ 16 级深度 FIFO ◆ I2S 主机模式通信
通用异步收发器（UART）	◆ 支持与外部设备进行全双工数据通信和单线半双工通信 ◆ 支持波特率自动测量功能 ◆ 提供 16 级深度接收和发送 FIFO ◆ 支持多点通信（RS-485） ◆ LIN（局域互连网络） ◆ 红外通信协议 ◆ 自动硬件流控制（CTS/RTS）
控制区域网络（CAN）	◆ 支持 2.0A 和 B Active 版本的 CAN 协议 ◆ 比特率高达 1 Mb/s ◆ 支持时间触发通信方案
通用串行总线（USB）	◆ 支持 USB2.0 协议 ◇ 支持 USB2.0 全速从设备 ◇ 支持 USB2.0 全速 OTG/主机/从设备 ◇ 支持 USB2.0 高速 OTG/主机/从设备 ◆ 支持 SRP/HNP ◆ 支持 4 种数据传输类型：控制传输/同步传输/中断传输/块传输 ◆ 支持挂起状态和恢复功能 ◆ 支持软件连接和断开功能 ◆ 支持 1 个双向控制端点 0 ◆ 支持 5 个发送端点：EP1OUT~EP5OUT（TX Endpoints） ◆ 支持 5 个接收端点：EP1IN~EP5IN（RX Endpoints） ◆ 支持 4KB SRAM 作为 FIFO ◆ 支持内部 DMA 访问 FIFO

表 2-9 通信模块

2.3.9 模拟模块

ES32F36xx 微控制器包含以下模拟模块：

模块	描述
模数转换（ADC）	12 位逐次逼近型模数转换器 ◆ 可配置的转换分辨率（6/8/10/12 位） ◆ 在规则转换、注入转换结束后以及发生模拟看门狗或溢出事件时产生中断 ◆ 支持单次或连续转换模式 ◆ 用于自动将通道 0 转换为通道“n”的扫描模式 ◆ 可配置的数据对齐方式 ◆ 可独立设置各通道采样时间 ◆ 可配置外部触发器选项，可为规则转换和注入转换配置极性 ◆ 支持不连续采样模式 ◆ 可配置的参考源选择 ◆ 可配置的转换时钟分频 ◆ 规则通道转换期间可产生 DMA 请求
数模转换（DAC）	◆ 12-bit 分辨率 ◆ 采样率 500ksps ◆ 可配置的参考源选择 ◆ 两个输出通道，也可配置为差分输出 ◆ 可配置的转换时钟分频，分频比可选范围为 1~128 ◆ 支持正弦波产生模式 ◆ 支持 PIS 触发 DA 转换功能 ◆ 转换完成后支持产生 DMA 请求
模拟比较器（ACMP）	◆ 可配置的迟滞选择，可选 8 个等级 ◆ 可配置多种低功耗工作模式 ◆ 在芯片 STOPx 模式下可工作
温度传感器（TSENSE）	◆ 可生成与温度成线性关系的电压，并转换成数字量输出
参考电压源（AVREF）	◆ 可生成高精度参考电压作为 ADC、DAC 的参考源，同时也可输出供外部使用

表 2-10 模拟模块

第3章 芯片配置指引

3.1 概述

本章节主要说明以下内容：

- 芯片顶层相关模块的连接及信号路径
- 阅读各模块时可参考的相关信息链接
- 芯片顶层连接资源的相关配置
- 模块之间特殊的交互，该部分内容在单独的模块说明章节中不再赘述

3.2 ARM Cortex-M3 内核配置

ARM Cortex-M3 提供了低功耗，低成本和快速中断响应的内核平台来满足 MCU 要求；能很好满足对系统响应时间有较高要求的应用。

3.2.1 ARM Cortex-M3 内核

关于 ARM Cortex-M3 内核技术细节可参考 ARM 官网 http://www.arm.com。

3.2.2 存储保护单元（MPU）

存储保护单元（MPU）用于检查访问权限和内存属性，能有效保护内存和外设寄存器，保障软件的健壮性和可靠性。

3.2.3 总线

支持 32 位 AMBA3 AHB 协议总线，Cortex-M3 内核提供 3 条 AHB 总线：指令、数据和系统总线；可支持对存储、外设的高效访问，以及对中断快速响应。

Cortex-M3 内核总线矩阵可支持对 SRAM 和外设的位带操作。

3.2.4 系统节拍定时器（Systick）

Systick 为递减计数器，计数时钟为内核时钟。具体配置可参考系统节拍控制和状态寄存器（SysTick Control and Status Register）相关说明。

3.2.5 调试器件

支持标准 SWD 协议的调试接口。

3.3 嵌套向量中断控制器

ES32F36xx 系列 MCU 的嵌套向量中断控制器可支持 16 个优先级设定。并具备以下特性：

- ◇ NVIC 与内核紧密配合支持快速中断响应时间
- ◇ 中断向量表直接传递至内核
- ◇ 支持中断嵌套，咬尾中断和迟来中断

3.3.1 中断优先级

中断优先级寄存器（Interrupt Priority Register）每个 byte 的高 4 位为有效位，支持 16 个中断优先级设置。

3.3.2 中断向量分配

中断向量分配如下表所示：

编号	优先级	名称	说明	地址
0	—	—	保留	0x0000_0000
1	-3	Reset	复位	0x0000_0004
2	-2	NMI	时钟安全事件	0x0000_0008
3	-1	HardFault	所有类型的错误	0x0000_000C
4	可设置	Memory Management	MPU 发生访问违例或不匹配	0x0000_0010
5	可设置	Bus Fault	预取错误、存储访问错误或其他总线错误	0x0000_0014
6	可设置	Usage Fault	未定义指令或非法操作	0x0000_0018
7~10	—	—	保留	0x0000_001C~ 0x0000_002B
11	可设置	SVCall	通过 SWI 指令调用的系统服务	0x0000_002C
12	可设置	Debug Monitor	内核未停止时调试监测	0x0000_0030
13	—	—	保留	0x0000_0034
14	可设置	PendSV	可挂起的系统服务	0x0000_0038
15	可设置	Systick	系统定时器	0x0000_003C
16	可设置	WWDT	WWDT 全局	0x0000_0040
17	可设置	IWDT	IWDT 全局	0x0000_0044
18	可设置	LVD	LVD 全局	0x0000_0048
19	可设置	RTC	RTC 全局	0x0000_004C
20~21	—	—	保留	0x0000_0050~0x0000_0054

编号	优先级	名称	说明	地址
22	可设置	CMU	CMU 全局	0x0000_0058
23	可设置	ADC0	ADC0 全局	0x0000_005C
24	可设置	CAN0_TX	CAN0 发送	0x0000_0060
25	可设置	CAN0_RX0	CAN0 接收 FIFO0	0x0000_0064
26	可设置	CAN0_RX1	CAN0 接收 FIFO1	0x0000_0068
27	可设置	CAN0_exception	CAN0 错误或状态变化	0x0000_006C
28	可设置	AD16C4T0_BRK	AD16C4T0 刹车中断	0x0000_0070
29	可设置	AD16C4T0_UP	AD16C4T0 更新中断	0x0000_0074
30	可设置	AD16C4T0_TRIG_COM	AD16C4T0 触发、通信中断	0x0000_0078
31	可设置	AD16C4T0_CC	AD16C4T0 捕获/比较中断	0x0000_007C
32	可设置	AD16C4T1_BRK	AD16C4T1 刹车中断	0x0000_0080
33	可设置	AD16C4T1_UP	AD16C4T1 更新中断	0x0000_0084
34	可设置	AD16C4T1_TRIG_COM	AD16C4T1 触发、通信中断	0x0000_0088
35	可设置	AD16C4T1_CC	AD16C4T1 捕获/比较中断	0x0000_008C
36	可设置	GP32C4T0	GP32C4T0 全局	0x0000_0090
37	可设置	GP32C4T1	GP32C4T1 全局	0x0000_0094
38	可设置	BS16T0	BS16T0 全局	0x0000_0098
39	可设置	BS16T1	BS16T1 全局	0x0000_009C
40	可设置	GP16C4T0	GP16C4T0 全局	0x0000_00A0
41	可设置	GP16C4T1	GP16C4T1 全局	0x0000_00A4
42	—	—	保留	0x0000_00A8
43	可设置	DAC0 通道 0	DAC0 通道 0	0x0000_00AC
44	可设置	I2C0_EV	I2C0 事件	0x0000_00B0
45	可设置	I2C0_ERR	I2C0 错误	0x0000_00B4
46	可设置	I2C1_EV	I2C1 事件	0x0000_00B8
47	可设置	I2C1_ERR	I2C1 错误	0x0000_00BC
48	可设置	SPI0	SPI0 全局	0x0000_00C0

编号	优先级	名称	说明	地址
49	可设置	SPI1	SPI1 全局	0x0000_00C4
50	可设置	UART0	UART0 全局	0x0000_00C8
51	可设置	UART1	UART1 全局	0x0000_00CC
52	可设置	UART2	UART2 全局	0x0000_00D0
53	可设置	UART3	UART3 全局	0x0000_00D4
54	可设置	UART4	UART4 全局	0x0000_00D8
55	可设置	UART5	UART5 全局	0x0000_00DC
56~57	—	—	保留	0x0000_00E0~0x0000_00E4
58	可设置	CRYPT	CRYPT 全局	0x0000_00E8
59	可设置	ACMP0	ACMP0 全局	0x0000_00EC
60	可设置	ACMP1	ACMP1 全局	0x0000_00F0
61	可设置	SPI2	SPI2 全局	0x0000_00F4
62	—	—	保留	0x0000_00F8
63	可设置	EBI	EBI 全局	0x0000_00FC
64	可设置	TRNG	TRNG 全局	0x0000_0100
65	可设置	TSENSE	TSENSE 全局	0x0000_0104
66	可设置	EXTI0	PA0~PH0 中断	0x0000_0108
67	可设置	EXTI1	PA1~PH1 中断	0x0000_010C
68	可设置	EXTI2	PA2~PH2 中断	0x0000_0110
69	可设置	EXTI3	PA3~PH3 中断	0x0000_0114
70	可设置	EXTI4	PA4~PH4 中断	0x0000_0118
71	可设置	EXTI5	PA5~PH5 中断	0x0000_011C
72	可设置	EXTI6	PA6~PH6 中断	0x0000_0120
73	可设置	EXTI7	PA7~PH7 中断	0x0000_0124
74	可设置	EXTI8	PA8~PH8 中断	0x0000_0128
75	可设置	EXTI9	PA9~PH9 中断	0x0000_012C
76	可设置	EXTI10	PA10~PH10 中断	0x0000_0130
77	可设置	EXTI11	PA11~PH11 中断	0x0000_0134
78	可设置	EXTI12	PA12~PH12 中断	0x0000_0138
79	可设置	EXTI13	PA13~PH13 中断	0x0000_013C
80	可设置	EXTI14	PA14~PH14 中断	0x0000_0140
81	可设置	EXTI15	PA15~PH15 中断	0x0000_0144
82	可设置	DMA	DMA 全局	0x0000_0148
83	可设置	ADC1	ADC1 全局	0x0000_014C
84	可设置	DAC0 通道 1	DAC0 通道 1	0x0000_0150

编号	优先级	名称	说明	地址
85	可设置	QSPI	QSPI 全局	0x0000_0154
86	可设置	USB_INT	USB 全局中断	0x0000_0158
87	可设置	USB_DMA	USB 内部 DMA 控制器中断	0x0000_015C
88	可设置	ACMP2	ACMP2 全局	0x0000_0160
89	—	—	保留	0x0000_0164

表 3-1 中断向量分配

3.4 异步唤醒中断和事件

3.4.1 异步中断唤醒源

在芯片 STOP1 低功耗模式下的中断唤醒源如下表：

唤醒源	描述
DMA 中断	DMA 中断可唤醒芯片
外部端口中断	外部端口（EXTI0~EXTI15）输入上升沿或下降沿中断
ACMP 比较器中断	模拟电压比较器可在 STOP1 模式下工作，并产生比较器中断唤醒芯片
LVD 中断	LVD 有效边沿或电平产生的中断可唤醒芯片
独立看门狗中断	使用 LRC 计数时，中断可唤醒芯片
RTC 中断	RTC 各中断源可唤醒芯片
复位	系统复位（不包含软件复位）

表 3-2 STOP1 低功耗模式的中断唤醒源

在芯片 STOP2 低功耗模式下的中断唤醒源如下表：

唤醒源	描述
外部端口中断	外部端口输入上升沿或下降沿中断
ACMP 比较器中断	模拟电压比较器可在 STOP2 模式下工作，并产生比较器中断唤醒芯片
LVD 中断	LVD 有效边沿或电平产生的中断可唤醒芯片
独立看门狗中断	使用 LRC 计数时，中断可唤醒芯片
RTC 中断	RTC 各中断源可唤醒芯片
复位	系统复位（不包含软件复位）

表 3-3 STOP2 低功耗模式的中断唤醒源

在芯片 STANDBY 模式下的中断唤醒源如下表：

唤醒源	描述
RTC 中断	RTC 各中断源可唤醒 STANDBY 模式
WAKEUP 端口中断	WAKEUP 端口（PA0~PA7）中断，可唤醒芯片。
MRSTN 复位	端口复位
POR	上电复位

表 3-4 STANDBY 低功耗模式的中断唤醒源

3.4.2 事件唤醒

ES32F36xx 微控制器支持事件唤醒机制：通过配置外设的中断控制寄存器使能一个中断，但在 NVIC 中不使能该中断(可通过设置 PRIMASK 和 BASEPRI 来禁止)，并将 Cortex-M3 内核的系统控制寄存器中的 SEVONPEND 位使能以允许中断事件唤醒 WFE。当外设中断产生后，芯片从 WFE 唤醒。芯片唤醒后，软件需要清除相应外设的中断标志位和外设在 NVIC 中断通道上的挂起位。芯片 STOP1,2 模式下的事件唤醒源如下表所示：

事件唤醒源	描述
EXTI0	选择 PA0,PB0,...PH0 之一作为唤醒源
EXTI1	选择 PA1,PB1,...PH1 之一作为唤醒源
EXTI2	选择 PA2,PB2,...PH2 之一作为唤醒源
EXTI3	选择 PA3,PB3,...PH3 之一作为唤醒源
EXTI4	选择 PA4,PB4,...PH4 之一作为唤醒源
EXTI5	选择 PA5,PB5,...PH5 之一作为唤醒源
EXTI6	选择 PA6,PB6,...PH6 之一作为唤醒源
EXTI7	选择 PA7,PB7,...PH7 之一作为唤醒源
EXTI8	选择 PA8,PB8,...PH8 之一作为唤醒源
EXTI9	选择 PA9,PB9,...PH9 之一作为唤醒源
EXTI10	选择 PA10,PB10,...PH10 之一作为唤醒源
EXTI11	选择 PA11,PB11,...PH11 之一作为唤醒源
EXTI12	选择 PA12,PB12,...PH12 之一作为唤醒源
EXTI13	选择 PA13,PB13,...PH13 之一作为唤醒源
EXTI14	选择 PA14,PB14,...PH14 之一作为唤醒源
EXTI15	选择 PA15,PB15,...PH15 之一作为唤醒源
IWDT	独立看门狗中断事件
WWDT	窗口看门狗中断事件
LVD	LVD 中断事件
RTC	RTC 中断事件
ACMP0/1/2	模拟比较器中断事件
DMA	DMA 完成中断事件

表 3-5 事件唤醒源

3.5 存储器及存储器接口

3.5.1 外设存储映射

ES32F36xx 系列产品外设存储映射如下表所示：

总线	边界地址	外设
APB1	0x4000 0000 - 0x4000 03FF	AD16C4T0
	0x4000 0400 - 0x4000 07FF	AD16C4T1
	0x4000 0800 - 0x4000 0BFF	GP32C4T0
	0x4000 0C00 - 0x4000 0FFF	GP32C4T1
	0x4000 1000 - 0x4000 13FF	BS16T0
	0x4000 1400 - 0x4000 17FF	BS16T1
	0x4000 1800 - 0x4000 1BFF	GP16C4T0
	0x4000 1C00 - 0x4000 1FFF	GP16C4T1
	0x4000 2000 - 0x4000 3FFF	—
	0x4000 4000 - 0x4000 43FF	UART0
	0x4000 4400 - 0x4000 47FF	UART1
	0x4000 4800 - 0x4000 4BFF	UART2
	0x4000 4C00 - 0x4000 4FFF	UART3
	0x4000 5000 - 0x4000 53FF	UART4
	0x4000 5400 - 0x4000 57FF	UART5
	0x4000 5800 - 0x4000 5FFF	—
	0x4000 6000 - 0x4000 63FF	SPI0
	0x4000 6400 - 0x4000 67FF	SPI1
	0x4000 6800 - 0x4000 6BFF	SPI2
	0x4000 6C00 - 0x4000 7FFF	—
	0x4000 8000 - 0x4000 83FF	I2C0
	0x4000 8400 - 0x4000 87FF	I2C1
	0x4000 8800 - 0x4000 9FFF	—
	0x4000 A000 - 0x4000 A3FF	—
	0x4000 A400 - 0x4000 AFFF	—
	0x4000 B000 - 0x4000 B3FF	CAN0
	0x4000 B400 - 0x4000 BFFF	—
	0x4000 C000 - 0x4000 D3FF	DMA
	0x4000 D400 - 0x4000 D7FF	QSPI
	0x4000 D800 - 0x4003 FFFF	—
APB2	0x4004 0000 - 0x4004 03FF	—
	0x4004 0400 - 0x4004 0FFF	—
	0x4004 1000 - 0x4004 13FF	—
	0x4004 1400 - 0x4004 1FFF	—
	0x4004 2000 - 0x4004 23FF	ADC0
	0x4004 2400 - 0x4004 27FF	ADC1

总线	边界地址	外设
	0x4004 2800 - 0x4004 2FFF	—
	0x4004 3000 - 0x4004 33FF	ACMP0
	0x4004 3400 - 0x4004 37FF	ACMP1
	0x4004 3800 - 0x4004 3BFF	ACMP2
	0x4004 3C00 - 0x4004 3FFF	—
	0x4004 4000 - 0x4004 43FF	—
	0x4004 4400 - 0x4004 47FF	—
	0x4004 4800 - 0x4004 4FFF	—
	0x4004 5000 - 0x4004 53FF	DAC0
	0x4004 5400 - 0x4004 5FFF	—
	0x4004 6000 - 0x4004 63FF	WWDT
	0x4004 6400 - 0x4004 67FF	IWDT
	0x4004 6800 - 0x4004 6FFF	—
	0x4004 7000 - 0x4004 73FF	—
	0x4004 7400 - 0x4004 7FFF	—
	0x4004 8000 - 0x4004 83FF	BKPC
	0x4004 8400 - 0x4004 87FF	RTC
	0x4004 8800 - 0x4004 8BFF	TSENSE
	0x4004 8C00 - 0x4004 8FFF	—
	0x4004 9000 - 0x4004 93FF	—
	0x4004 9400 - 0x4004 97FF	—
	0x4004 9800 - 0x4004 9FFF	—
	0x4004 A000 - 0x4004 A3FF	DBGC
	0x4004 A400 - 0x4007 FFFF	—
AHB1	0x4008 0000 - 0x4008 03FF	SYSCFG
	0x4008 0400 - 0x4008 07FF	CMU
	0x4008 0800 - 0x4008 0BFF	RMU
	0x4008 0C00 - 0x4008 0FFF	PMU
	0x4008 1000 - 0x4008 13FF	MSC
	0x4008 1400 - 0x4008 3BFF	—
	0x4008 3C00 - 0x4008 3FFF	—
	0x4008 4000 - 0x4008 4FFF	GPIO
	0x4008 5000 - 0x4008 53FF	CRC
	0x4008 5400 - 0x4008 57FF	CALC
	0x4008 5800 - 0x4008 5BFF	CRYPT
	0x4008 5C00 - 0x4008 5FFF	TRNG
	0x4008 6000 - 0x4008 63FF	PIS
	0x4008 6400 - 0x4008 67FF	USB
	0x4008 6800 - 0x4008 FFFF	—

表 3-6 外设存储映射

3. 5. 2 QSPI接口配置

3. 5. 2. 1 QSPI时钟

QSPI 的 APB 总线时钟为 PCLK1。QSPI 的 AHB 总线时钟为 HCLK2。

另外为 QSPI 提供了独立于系统时钟的通信时钟源，从而使 QSPI 通信时钟频率可高于系统时钟。QSPI 的通信时钟源可灵活配置，如下图所示。

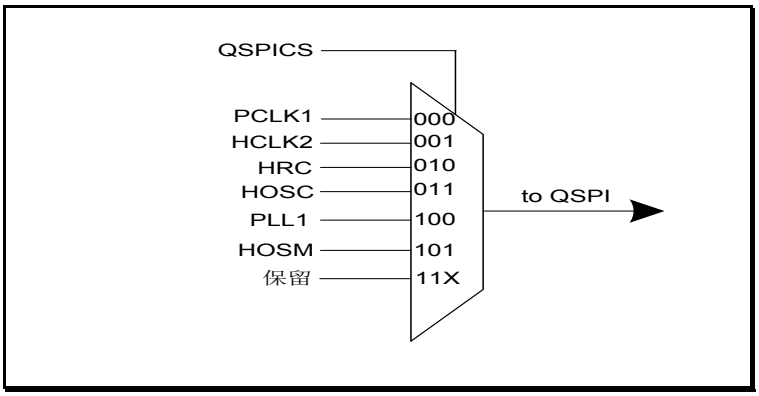


图 3-1 QSPI 通信时钟源

3. 5. 2. 2 QSPI访问空间

QSPI Flash 控制器可用来访问外部串口 Flash。可访问的外部存储空间地址范围为 0x90000000~0x9FFFFFFF。

3. 5. 3 EBI接口配置

3. 5. 3. 1 EBI时钟

EBI 总线时钟和模块时钟为 HCLK2。

3. 5. 3. 2 EBI访问空间

EBI 可用于扩展外部 SRAM,PSRAM,NOR 和 NAND Flash 等。可访问的外部存储空间地址范围为：0x60000000~0x8FFFFFFF。EBI 控制寄存器地址范围为 0xA0000000~0xA0000FFF。

3.6 系统模块配置

3.6.1 DMA控制器配置

DMA 控制器包含 12 个通道，每个 DMA 通道对应一个 DMA 多路复用器，每个多路复用器包含了微控制器所有的 DMA 申请源，由 DMA_CHx_SELCON（x=0,1…11）配置选择。多路复用器和 DMA 之间连接图如下：

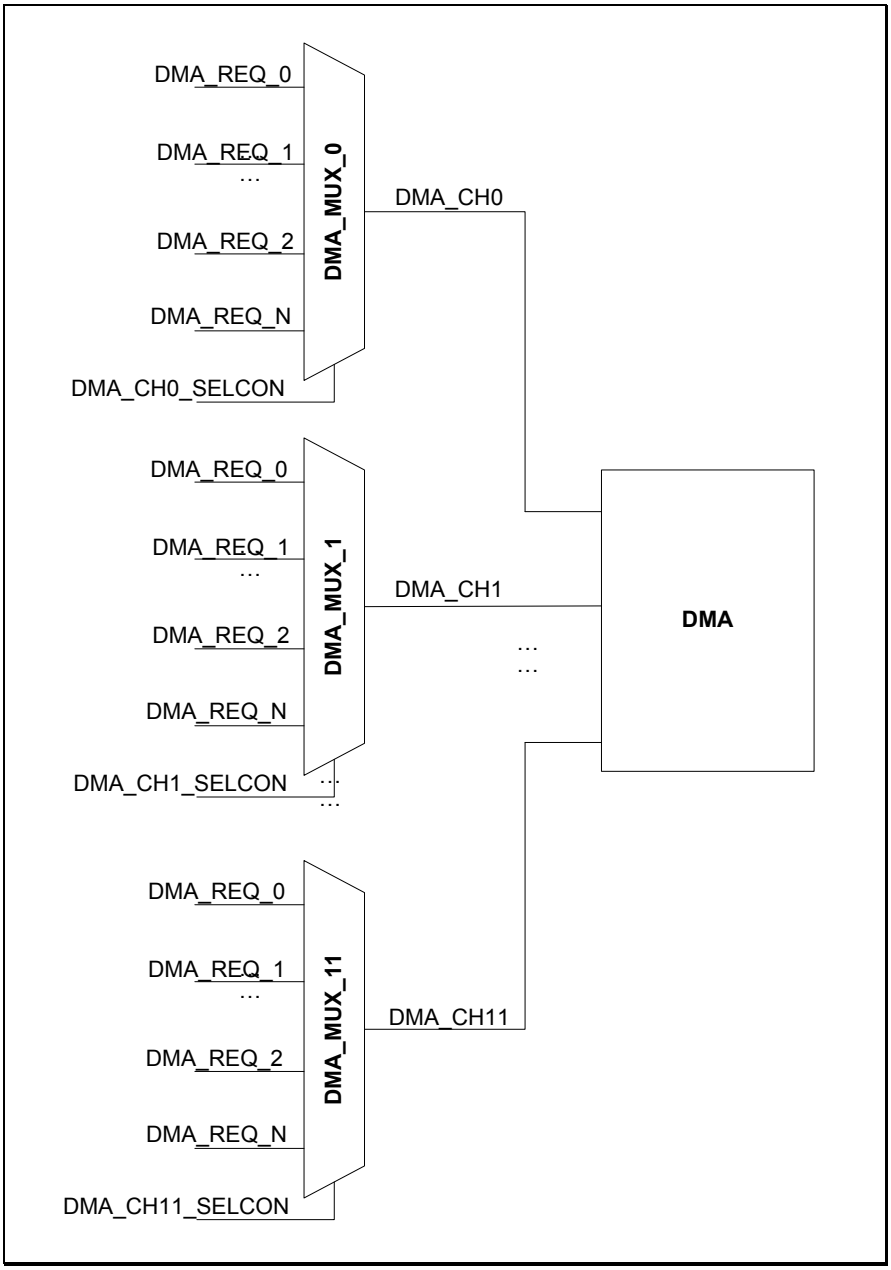


图 3-2 DMA 多路复用器与 DMA 连接图

每个 DMA 多路复用器可选择的 DMA 请求如下表所示：

模块名	DMA 请求源
GPIO	EXTI0~EXTI15

模块名	DMA 请求源
ADC0	ADC0 转换结束
ADC1	ADC1 转换结束
AD16C4T0	AD16C4T0_CH1
	AD16C4T0_CH2
	AD16C4T0_CH3
	AD16C4T0_CH4
	AD16C4T0_TRIG
	AD16C4T0_COM
	AD16C4T0_UP
AD16C4T1	AD16C4T1_CH1
	AD16C4T1_CH2
	AD16C4T1_CH3
	AD16C4T1_CH4
	AD16C4T1_TRIG
	AD16C4T1_COM
	AD16C4T1_UP
GP32C4T0	GP32C4T0_CH1
	GP32C4T0_CH2
	GP32C4T0_CH3
	GP32C4T0_CH4
	GP32C4T0_TRIG
	GP32C4T0_UP
GP32C4T1	GP32C4T1_CH1
	GP32C4T1_CH2
	GP32C4T1_CH3
	GP32C4T1_CH4
	GP32C4T1_TRIG
	GP32C4T1_UP
BS16T0	BS16T0_UP
BS16T1	BS16T1_UP
GP16C4T0	GP16C4T0_CH1
	GP16C4T0_CH2
	GP16C4T0_CH3
	GP16C4T0_CH4
	GP16C4T0_TRIG
	GP16C4T0_UP
GP16C4T1	GP16C4T1_CH1
	GP16C4T1_CH2
	GP16C4T1_CH3
	GP16C4T1_CH4
	GP16C4T1_TRIG

模块名	DMA 请求源
	GP16C4T1_UP
UART0	UART0_RX
	UART0_TX
UART1	UART1_RX
	UART1_TX
UART2	UART2_RX
	UART2_TX
UART3	UART3_RX
	UART3_TX
UART4	UART4_TX
	UART4_RX
UART5	UART5_TX
	UART5_RX
SPI0	SPI0_TX
	SPI0_RX
SPI1	SPI1_TX
	SPI1_RX
SPI2	SPI2_TX
	SPI2_RX
I2C0	I2C0_TX
	I2C0_RX
I2C1	I2C1_TX
	I2C1_RX
CRC	CRC DMA 写请求
CRYPT	CRYPT DMA 写请求
	CRYPT DMA 读请求
PIS	PIS 通道 0~7
TRNG	TRNG 数据有效
QSPI	QSPI 间接访问外部 Flash 写请求
	QSPI 间接访问外部 Flash 读请求
USB	USB 发送端点 1
	USB 发送端点 2
	USB 发送端点 3
	USB 发送端点 4
	USB 发送端点 5
	USB 接收端点 1
	USB 接收端点 2
	USB 接收端点 3
	USB 接收端点 4
	USB 接收端点 5

表 3-7 DMA 请求列表

3. 6. 2 独立看门狗定时器配置

3. 6. 2. 1 独立看门狗定时器的时钟

下图为独立看门狗定时器的计数时钟源选择：

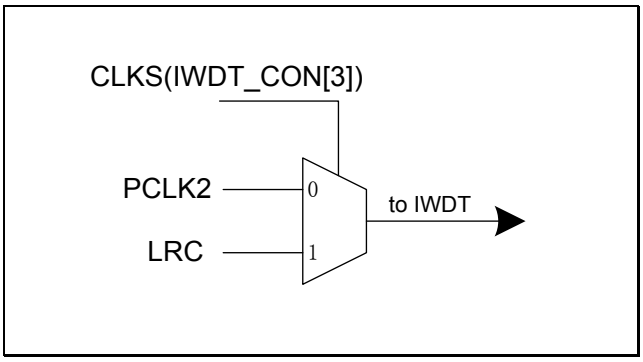


图 3-3 独立看门狗计数时钟

3. 6. 2. 2 独立看门狗定时器的低功耗动作模式

低功耗模式	模块工作模式
STOP1	可工作
STOP2	可工作
STANDBY	掉电

表 3-8 独立看门狗定时器的低功耗动作模式

3. 6. 3 窗口看门狗定时器配置

3. 6. 3. 1 窗口看门狗定时器的时钟

下图为窗口看门狗定时器的计数时钟源选择：

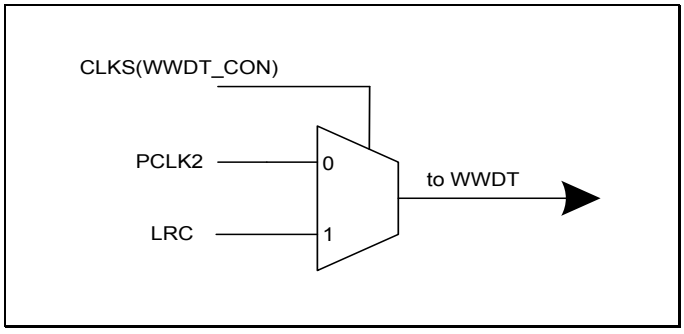


图 3-4 窗口看门狗计数时钟

3. 6. 3. 2 窗口看门狗定时器的低功耗动作模式

低功耗模式	模块工作模式
STOP1	可工作
STOP2	可工作
STANDBY	掉电

表 3-9 窗口看门狗定时器的低功耗动作模式

3. 6. 4 时钟管理配置

3. 6. 4. 1 HOSC的低功耗动作模式

低功耗模式	模块工作模式
STOP1	可配置
STOP2	关闭
STANDBY	关闭

表 3-10 HOSC 的低功耗动作模式

3. 6. 4. 2 HRC的低功耗动作模式

低功耗模式	模块工作模式
RUN,SLEEP	可通过配置 CMU 配置寄存器（CMU_CFGR）的 HRCFSW 位选择输出 24MHz 或 2MHz 时钟。
STOP1	可配置
STOP2	关闭
STANDBY	关闭

表 3-11 HRC 的低功耗动作模式

3. 6. 4. 3 LOSC的低功耗动作模式

低功耗模式	模块工作模式
STOP1	可工作
STOP2	可工作
STANDBY	可工作

表 3-12 LOSC 的低功耗动作模式

3. 6. 4. 4 LRC的低功耗动作模式

低功耗模式	模块工作模式
STOP1	可工作
STOP2	可工作
STANDBY	可工作

表 3-13 LRC 的低功耗动作模式

3.6.4.5 ULRC的低功耗动作模式

低功耗模式	模块工作模式
STOP1	可工作
STOP2	可工作
STANDBY	可工作

表 3-14 ULRC 的低功耗动作模式

3.7 外部接口配置

3.7.1 通用IO及端口控制配置

3.7.1.1 端口特殊配置说明

SWDIO 和 SWCLK 默认均为上拉。

SWDIO 和 SWCLK 默认使用 TTL 电平输入，以支持 3.3V 输入系统。

3.8 定时器配置

3.8.1 高级定时器（AD16C4T）配置

3.8.1.1 高级定时器例化说明

ES32F36xx 系列 MCU 中，高级定时器（AD16C4T）为 AD16C4T0，AD16C4T1。

3.8.1.2 高级定时器的时钟

高级定时器的总线时钟及模块时钟源为 PCLK1。

3.8.1.3 高级定时器的低功耗动作模式

低功耗模式	模块工作模式
STOP1	不工作
STOP2	不工作
STANDBY	掉电

表 3-15 高级定时器的低功耗动作模式

3.8.2 通用定时器配置

3.8.2.1 通用定时器例化说明

ES32F36xx 系列 MCU 中，GP32C4T0、GP32C4T1 为 4 通道 32 位通用定时器（GP32C4T）。GP16C4T0、GP16C4T1 为 4 通道 16 位通用定时器（GP16C4T）。

3.8.2.2 通用定时器的时钟

通用定时器的总线时钟和模块时钟源为 PCLK1。

3.8.2.3 通用定时器的低功耗动作模式

低功耗模式	模块工作模式
STOP1	不工作
STOP2	不工作
STANDBY	掉电

表 3-16 通用定时器的低功耗动作模式

3.8.3 基本定时器（BS16T）配置

3.8.3.1 基本定时器例化说明

ES32F36xx 系列 MCU 中，BS16T0、BS16T1 为基本定时器。

3.8.3.2 基本定时器的时钟

基本定时器的总线时钟和模块时钟源为 PCLK1。

3.8.3.3 基本定时器的低功耗动作模式

低功耗模式	模块工作模式
STOP1	不工作
STOP2	不工作
STANDBY	掉电

表 3-17 基本定时器的低功耗动作模式

3.8.4 实时定时器（RTC）配置

3.8.4.1 RTC的时钟

下图为实时定时器的计数时钟源选择。

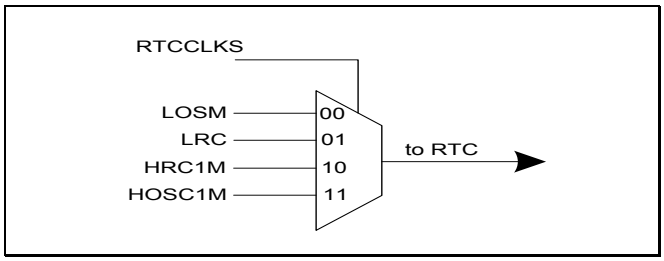


图 3-5 RTC 计数时钟

3.8.4.2 RTC的低功耗动作模式

低功耗模式	模块工作模式
STOP1	可工作
STOP2	可工作
STANDBY	可工作

表 3-18 RTC 的低功耗动作模式

3.9 通信配置

3.9.1 I2C接口配置

3.9.1.1 I2C接口的时钟

I2C 的总线时钟和模块时钟源为 PCLK1。

3.9.1.2 I2C接口的低功耗动作模式

低功耗模式	模块工作模式
STOP1	不工作
STOP2	不工作
STANDBY	掉电

表 3-19 I2C 接口的低功耗动作模式

3.9.2 串行外设接口（SPI/I2S）配置

3.9.2.1 串行外设接口（SPI/I2S）的时钟

SPI 的总线时钟和模块时钟源为 PCLK1。

3.9.2.2 串行外设接口（SPI/I2S）的低功耗动作模式

低功耗模式	模块工作模式
STOP1	不工作
STOP2	不工作
STANDBY	掉电

表 3-20 串行外设接口的低功耗动作模式

3.9.3 通用异步收发器（UART）

3.9.3.1 UART例化说明

ES32F36xx 系列 MCU 中，UART0~5 为通用 UART，其中 UART4~5 可额外支持 ISO7816 协议。

3.9.3.2 通用异步收发器（UART）的时钟

UART 的总线时钟和模块时钟源为 PCLK1。

3.9.3.3 通用异步收发器（UART）的低功耗动作模式

低功耗模式	模块工作模式
STOP1	不工作
STOP2	不工作
STANDBY	掉电

表 3-21 通用异步收发器的低功耗动作模式

3.9.4 控制器局域网络（CAN）

3.9.4.1 控制器局域网络（CAN）的时钟

CAN 的总线时钟和模块时钟源为 PCLK1。

3.9.4.2 控制区域网络（CAN）的低功耗动作模式

低功耗模式	模块工作模式
STOP1	不工作
STOP2	不工作
STANDBY	掉电

表 3-22 控制器局域网络的低功耗动作模式

3.9.5 通用串行总线（USB）

3.9.5.1 通用串行总线（USB）电源

通用串行总线（USB）在默认情况下电源关闭以节省功耗。在使用 USB 前，请开启 USB 电源，USB 电源可通过设置电源控制寄存器 PMU_PWRCR.USB 来开启。

另外需注意：如无 USB 应用需要，在芯片进入 STOP1/STOP2、STANDBY 等模式时，请确保 USB 电源已通过软件关闭，以免带来额外的待机功耗。

3.9.5.2 通用串行总线（USB）时钟

通用串行总线（USB）的总线时钟为 AHB 时钟（HCLK1）。USB 工作时须保证系统时钟 HCLK1 工作在 30MHz<f_{HCLK1}<=72MHz 的频率范围。

UTMI 接口通信时钟为 60MHz，由内部集成的 USB2.0 PHY 提供。

USB2.0 PHY 的参考时钟源可灵活配置，请参考下图：

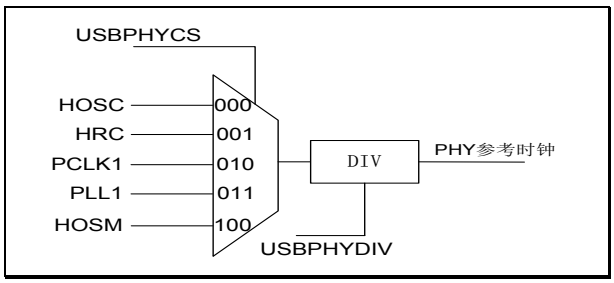


图 3-6 USB2.0 PHY 的参考时钟源

3.9.5.3 通用串行总线（USB）的低功耗动作模式

低功耗模式	模块工作模式
STOP1	通过软件设置电源控制寄存器 PMU_PWRCR.USB 关闭 USB 电源
STOP2	通过软件设置电源控制寄存器 PMU_PWRCR.USB 关闭 USB 电源
STANDBY	通过软件设置电源控制寄存器 PMU_PWRCR.USB 关闭 USB 电源

表 3-23 通用串行总线（USB）的低功耗动作模式

3. 10 模拟配置

3. 10. 1 ADC控制配置

3. 10. 1. 1 ADC模块例化

ES32F36xx 包含 2 路 ADC（ADC0、1）。

3. 10. 1. 2 ADC转换通道配置

ADC0、1 支持 16 个通道选择，具体分配如下表所示，每个 ADC 通道在管脚上的对应关系，请参考数据手册的管脚功能定义表格。

寄存器 ADC_CON0 的 AWDCH	ADC 通道	信号分配
00000	ADC 通道 0	ADC_IN0
00001	ADC 通道 1	ADC_IN1
00010	ADC 通道 2	ADC_IN2
00011	ADC 通道 3	ADC_IN3
00100	ADC 通道 4	ADC_IN4
00101	ADC 通道 5	ADC_IN5
00110	ADC 通道 6	ADC_IN6
00111	ADC 通道 7	ADC_IN7
01000	ADC 通道 8	ADC_IN8
01001	ADC 通道 9	ADC_IN9
01010	ADC 通道 10	ADC_IN10
01011	ADC 通道 11	ADC_IN11
01100	ADC 通道 12	ADC_IN12
01101	ADC 通道 13	ADC_IN13
01110	ADC 通道 14	ADC_IN14
01111	ADC 通道 15	ADC_IN15
10000~11111	—	—

表 3-24 ADC 转换通道配置

3. 10. 1. 3 ADC电源及参考电压

ADC 电源由 VDD 提供。ADC 的参考电压可选 VDD 或 VREFP（VREFP 可由内部参考产生或外部端口 VREFP 输入）。请注意 ADC 参考电压源在 ADC0 寄存器中进行配置。

3. 10. 1. 4 ADC的时钟

ADC 的总线时钟和模块时钟源为 PCLK2。

3. 10. 1. 5 ADC的低功耗动作模式

低功耗模式	模块工作模式
SLEEP	可工作

表 3-25 ADC 的低功耗动作模式

3. 10. 2 DAC控制配置

3. 10. 2. 1 DAC模块例化

ES32F36xx 包含 2 路 DAC，2 路 DAC 共用一个主控制器。

3. 10. 2. 2 DAC电源及参考电压

DAC 电源由 VDD 提供。

DAC 的参考电压可选 VDD 或 VREFP（VREFP 可由内部参考产生或外部端口 VREFP 输入）。请注意 DAC 参考电压源在 ADC0 寄存器中进行配置，即 DAC 和 ADC 的参考电压是共用的。

3. 10. 2. 3 DAC的时钟

DAC 的总线时钟和模块时钟源为 PCLK2。DAC 转换时钟为 PCLK2 的分频输出。

3. 10. 2. 4 DAC的低功耗动作模式

低功耗模式	模块工作模式
SLEEP	可工作

表 3-26 DAC 的低功耗动作模式

3. 10. 3 ACMP控制配置

3. 10. 3. 1 ACMP模块例化

ES32F36xx 包含 3 路 ACMP 模块（ACMP0、1、2）。

3. 10. 3. 2 ACMP比较通道配置

ACMP0、1、2 正端支持 8 个外部通道选择，负端支持 8 个外部通道和 5 个内部通道选择。具体分配如下表。ACMP 通道与管脚的对应关系请参考数据手册。

寄存器 ACMP_INPUTSEL 的 PSEL	ACMP 通道	信号分配
000	ACMP 外部通道 0	ACMP_IN0
001	ACMP 外部通道 1	ACMP_IN1
010	ACMP 外部通道 2	ACMP_IN2
011	ACMP 外部通道 3	ACMP_IN3
100	ACMP 外部通道 4	ACMP_IN4
101	ACMP 外部通道 5	ACMP_IN5
110	ACMP 外部通道 6	ACMP_IN6
111	ACMP 外部通道 7	ACMP_IN7

表 3-27 ACMP 正端通道选择

寄存器 ACMP_INPUTSEL 的 NSEL	ACMP 通道	信号分配
0000	ACMP 外部通道 0	ACMP_IN0
0001	ACMP 外部通道 1	ACMP_IN1
0010	ACMP 外部通道 2	ACMP_IN2
0011	ACMP 外部通道 3	ACMP_IN3
0100	ACMP 外部通道 4	ACMP_IN4
0101	ACMP 外部通道 5	ACMP_IN5
0110	ACMP 外部通道 6	ACMP_IN6
0111	ACMP 外部通道 7	ACMP_IN7
1000	ACMP 内部通道 0	内部参考电压 1V
1001	ACMP 内部通道 1	VREFP
1010	ACMP 内部通道 2	VDD 分压
1011~1101	—	—
1110	ACMP 内部通道 6	DAC 通道 0 输出
1111	ACMP 内部通道 7	DAC 通道 1 输出

表 3-28 ACMP 负端通道选择

3. 10. 3. 3 ACMP电源

ACMP 电源为 VDD。

3. 10. 3. 4 ACMP的时钟

ACMP 的总线时钟和模块时钟源为 PCLK2。

3. 10. 3. 5 ACMP的低功耗动作模式

低功耗模式	模块工作模式
STOP1	可工作
STOP2	可工作
STANDBY	掉电

表 3-29 ACMP 的低功耗动作模式

第4章　系统总线和存储器

4.1　概述

主系统由 32 位多层 AHB 总线矩阵构成，可实现以下部分的互连。

五条主控总线：

- Cortex-M3 内核 I 总线、D 总线和 S 总线
- DMA 存储器总线
- USB OTG DMA 总线

七条被控总线：

- 内部 Flash 总线
- 内部 SRAM 总线
- AHB 外设
- APB1 外设
- APB2 外设
- 外部存储器接口

借助总线矩阵，可以实现主控总线到被控总线的访问，这样即使在多个高速外设同时运行期间，系统也可以实现并发访问和高效运行。

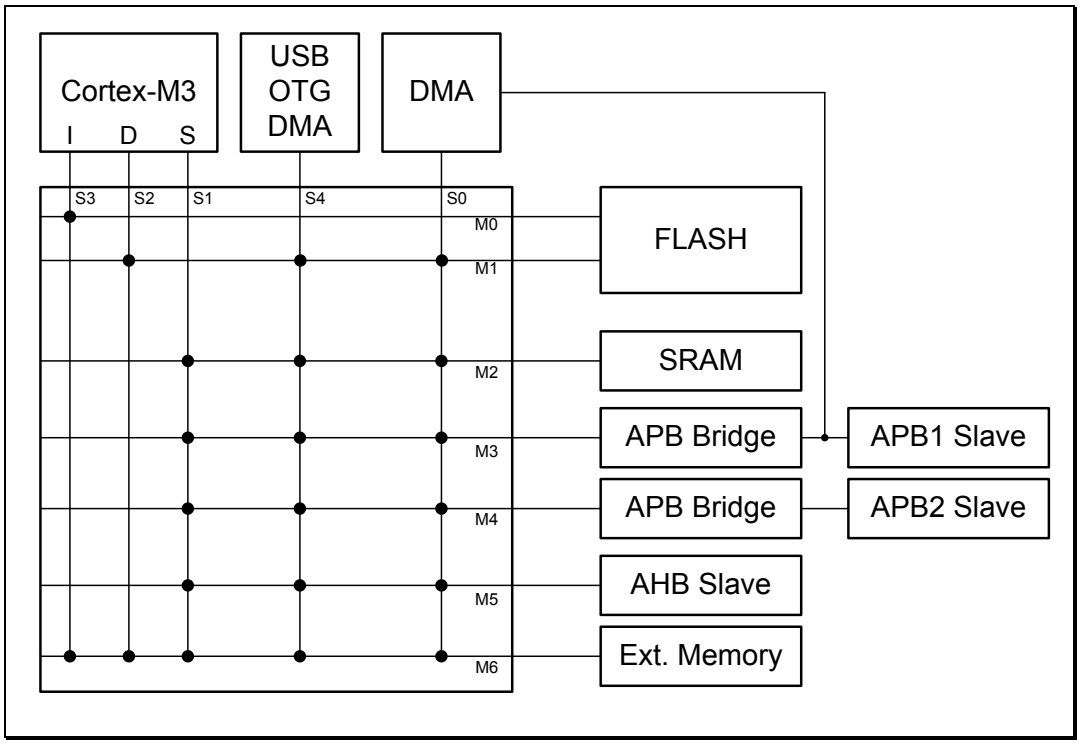


图 4-1　系统总线矩阵

4.2 系统总线

4.2.1 S0：DMA总线

此总线用于将 DMA 存储器总线主接口连接到总线矩阵。DMA 通过此总线访问外设或执行存储器间的数据传输。此总线访问的对象是 AHB 和 APB 外设以及数据存储器：内部 SRAM 以及通过 EBI/QSPI 的外部存储器。

4.2.2 S1：S总线

此总线用于将 Cortex-M3 内核的系统总线连接到总线矩阵。此总线用于访问位于外设或 SRAM 中的数据。也可通过此总线获取指令（效率低于 I 总线）。此总线访问的对象是内部 SRAM、AHB 外设、APB1 外设、APB2 外设以及通过 EBI/QSPI 的外部存储器。

4.2.3 S2：D总线

此总线用于将 Cortex-M3 内核的数据总线连接到总线矩阵。内核通过此总线进行立即数加载和调试访问。此总线访问的对象是包含代码或数据的存储器（内部 Flash）。

4.2.4 S3：I总线

此总线用于将 Cortex-M3 内核的指令总线连接到总线矩阵。内核通过此总线获取指令。此总线访问的对象是包含代码的存储器（内部 Flash）。

4.2.5 S4：USB OTG DMA总线

此总线用于将 USB OTG DMA 主接口连接到总线矩阵。USB OTG DMA 通过此总线向存储器加载或存储数据。此总线访问的对象是数据存储器：内部 SRAM 以及通过 EBI/QSPI 的外部存储器。

4.2.6 总线矩阵

总线矩阵用于主控总线之间的访问仲裁管理，仲裁采用循环调度算法。

4.2.7 AHB/APB总线桥

借助两个 AHB/APB 总线桥 APB1 和 APB2，可在 AHB 总线与两个 APB 总线之间实现完全同步的连接，从而可灵活选择外设频率。

4.3 存储器的组织结构

程序存储器、数据存储器、寄存器和 I/O 端口排列在同一个顺序的 4 GB 地址空间内。

各字节按小端格式在存储器中编码。字中编号最低的字节被视为该字的最低有效字节，而编号最高的字节被视为最高有效字节。

可寻址的存储空间分为 8 个主要块，每个块为 512MB。

未分配给片上存储器和外设的所有存储区域均视为“保留区”。请参见产品数据手册中的存储器映射图。

4.3.1 系统存储器映射

系统存储器映射图如下表所示：

地址范围	存储	备注
0x0000_0000~0x0007_FFFF	Flash 存储器	
0x0008_0000~0x1FFF_FFFF	Reserved	
0x2000_0000~0x2001_7FFF	SRAM	
0x2001_8000~0x3FFF_FFFF	Reserved	
0x4000_0000~0x4003_FFFF	APB1 外设	
0x4004_0000~0x4007_FFFF	APB2 外设	
0x4008_0000~0x4008_FFFF	AHB 外设	
0x4009_0000~0x5FFF_FFFF	Reserved	
0x6000_0000~0xDFFF_FFFF	外部存储	可通过 EBI 或 QSPI 扩展外部存储
0xE000_0000~0xE00F_FFFF	私有外设	
0xE010_0000~0xFFFF_FFFF	Reserved	

表 4-1 系统存储器映射

4.3.2 Flash存储器映射

Flash 接口可控制 CPU 通过 AHB I 总线和 D 总线对 Flash 进行的访问，该接口可对 Flash 执行擦除和编程操作，并实施读写保护机制。Flash 接口支持指令预取和缓存机制，以提高代码执行效率。

Flash 结构如下：

- ◇ 主存储区共 512K Bytes，分为 512 页，每页 1K Bytes，支持预取模式，一次可读 8 Bytes
- ◇ 信息区用于存储芯片配置字，通过更改配置字实现芯片读写保护，更改 BOR 级别，启用或关闭硬件开门狗，配置器件在待机或停止状态下的复位模式。

4.3.3 SRAM存储器映射

SRAM 容量为 96K Bytes，地址范围 0x2000_0000~0x2001_7FFF，单周期访问时间。

4.3.4 外设存储映射

ES32F36xx 产品外设存储映射请参考章节“芯片配置指引”的外设存储映射表。

4.3.5 私有外设存储器映射

地址范围	私有外设	备注
0xE000_0000~0xE000_0FFF	ITM	
0xE000_1000~0xE000_1FFF	DWT	
0xE000_2000~0xE000_2FFF	FPB	
0xE000_3000~0xE000_DFFF	Reserved	
0xE000_E000~0xE000_EFFF	NVIC	
0xE000_F000~0xE003_FFFF	Reserved	
0xE004_0000~0xE004_0FFF	TPIU	
0xE004_1000~0xE004_1FFF	Reserved	
0xE004_2000~0xE00F_EFFF	外部扩展私有外设	
0xE00F_F000~0xE00F_FFFF	Reserved	

表 4-2 私有外设存储器映射

4.3.6 位带（Bitband）

4.3.6.1 SRAM位带扩展

SRAM 支持位带扩展，可使用普通的加载和存储指令对单比特进行读写操作。当 SRAM 存储空间小于 1MB 时（地址范围：0x2000_0000 ~0x2001_7FFF），通过位带扩展，支持起始地址为 0x2000_0000 的空间访问 SRAM，支持起始地址为 0x2200_0000 的位带扩展区以单比特方式访问 SRAM。

位带扩展区把每个比特扩展为一个 32-bit 的字，即占用 4 个字节地址；一个 byte 占用 8x4=32 个地址。通过访问这些字可达到访问原始比特的目的。对于 SRAM 的某个 bit，如果它所在字节地址为 A，位序号为 N（0≤N≤7），则该 bit 在 SRAM 位带扩展后的地址为：

AliasAddress_A_N = 0x2200_0000 + (A – 0x2000_0000) x 32 + N x 4

例如，字节地址 A 为 0x2000_0001，访问该地址的 bit1，地址为：

AliasAddress_A_N = 0x2200_0000 + 1 x 32 + 1 x 4 = 0x2200_0024

4. 3. 6. 2 外设位带扩展

AliasAddress_A_N = 0x4200_0000 + (A – 0x4000_0000) x 32 + N x 4

利用外设位带访问对寄存器位置 1 和清 0

```
LDR     R0, = AliasAddress_A_N
MOVS    R1, #1
STR     R1, [R0]                ；对该位置 1

LDR     R0, = AliasAddress_A_N
MOVS    R1, #0
STR     R1, [R0]                ；对该位清 0
```

4.4 启动引导

芯片发生复位（包括上电复位、欠压复位、MRST 复位或软件复位）时，若配置字 BOOT（CFG_WORD0[12] =1）使能，程序将从 Boot Flash 启动，如果需要将程序引导至 App Flash，须先将寄存器位 BFRMPEN 改写为 0，然后通过软件跳转进行引导。

Boot Flash 为用户烧录的 Boot 程序，对应不同的 Flash 空间（512K/384K/256K/128K）起始地址分别为 0x0007_E000/0x0005_E000/0x0003_E000/0x0001_E000，共 8K 字节。APP Flash 为用户系统运行程序，起始地址为 0x0000_0000。

芯片发生复位（包括上电复位、欠压复位、MRST 复位或软件复位）时，若配置字 BOOT（CFG_WORD0[12] =0）禁止，程序将直接从 Flash 起始地址 0x0000_0000 启动。

Flash 启动引导如下图所示（示例中 Flash 容量为 512K）：

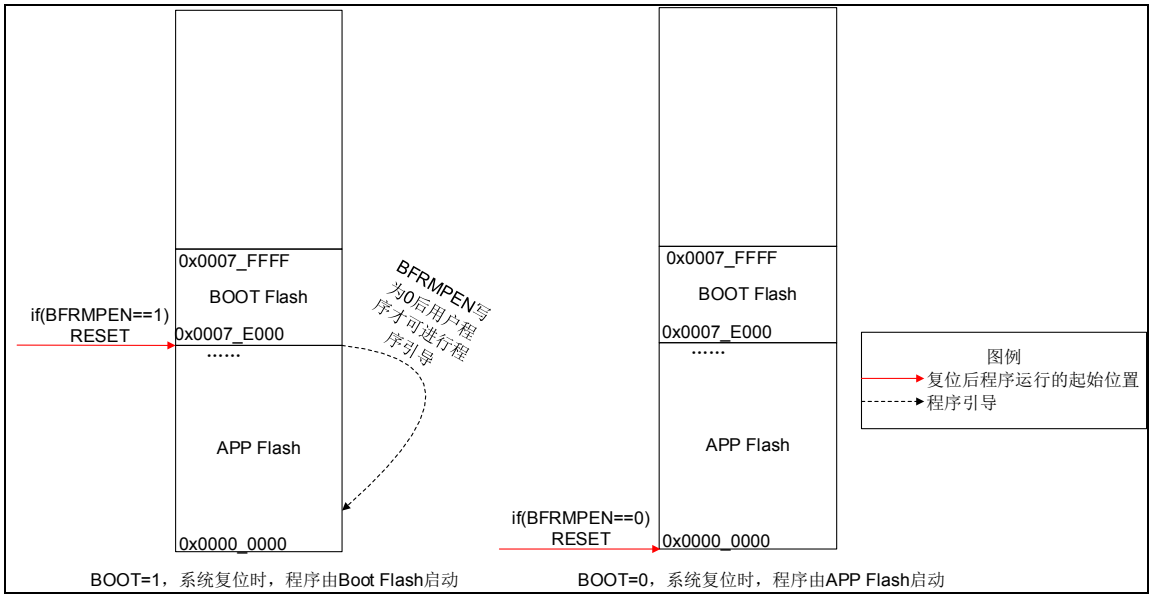


图 4-2 启动引导

第5章 存储器系统控制

5.1 概述

存储器系统控制（MSC）主要作用是控制系统程序编程 Flash 的各种操作，包括全擦除，页擦除，写操作，读操作以及对应的访问权限管理等，并实时反馈各种控制操作中的状态，以便于系统对 Flash 编程进行控制。

存储器系统控制（MSC）支持两个写保护分区，支持两个私有代码读保护区，以及 Flash 主程序区的全局读保护。

存储器系统控制（MSC）中可以划分独立 Data Flash 区域用于存放用户数据，用户可根据具体应用灵活选择。

写保护分区、私有代码读保护区、全局读保护以及数据 Flash 区域的配置请同时对照章节“Flash 信息区”。

5.2 特性

- 支持主程序区和信息区
- 支持对 Flash 的编程和擦除控制
  - 程序区全擦除
  - 页擦除
  - 字编程
  - Data Flash 页擦除
  - Data Flash 字编程
  - 可支持快速编程
- 存储器读取等待时间可配置
- 支持 2 个写保护分区，保护区域范围可配置
- 支持数据 Flash 区域配置
- 支持全局读保护，保护等级可配置
- 支持 2 个私有代码读保护分区，保护区域范围可配置

5.3 结构框图

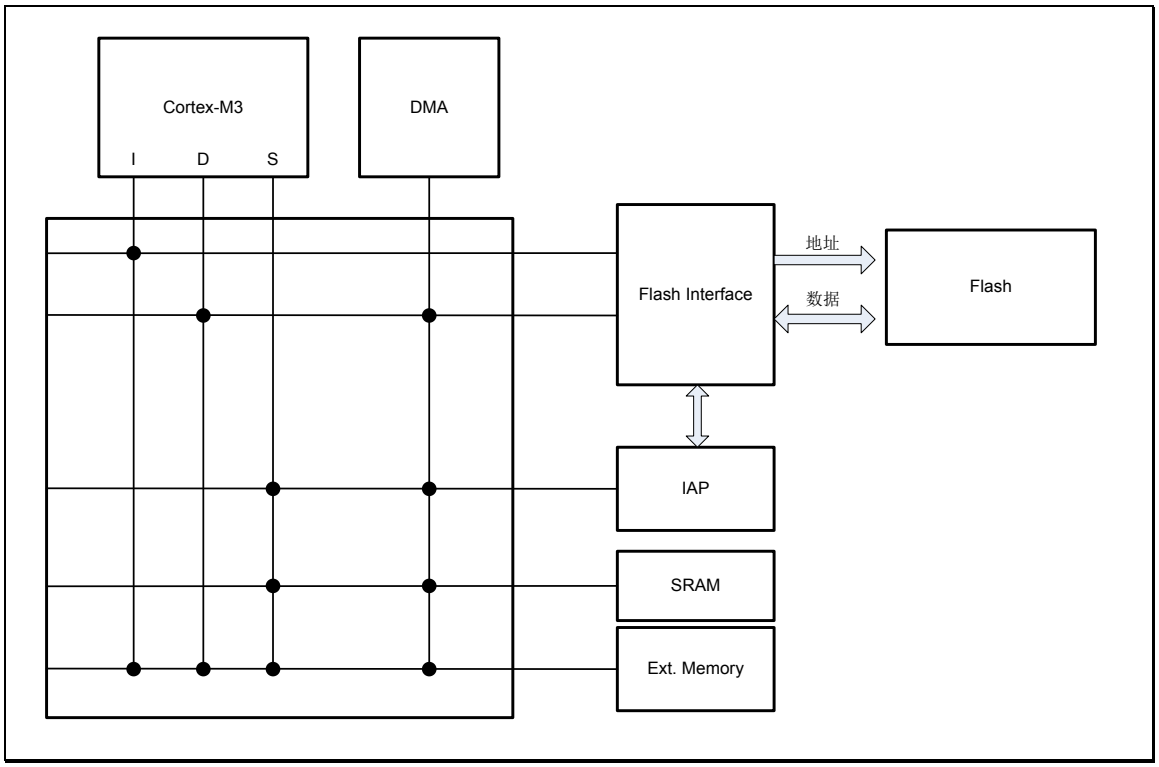


图 5-1 Flash 控制结构图

在上电时，系统从 Flash 中加载配置字，配置字加载过程检查成功后，当 BOOT（CFG_WORD0[12] =1）使能时，系统复位后可选从 Boot Flash 中启动，此时，若需要更新下载用户程序，则通过 IAP 对用户 Flash 区进行擦写，并读出校验，编程完成后，Boot Flash 再引导系统从 Flash 运行用户程序；若 BOOT（CFG_WORD0[12] =0）禁止时系统复位，则直接将系统引导到 Flash 中运行用户程序。

5.4 功能描述

5.4.1 Flash保护

5.4.1.1 IAP操作保护KEY

软件通过写 MSC_FLASHKEY 和 MSC_INFOKEY 寄存器，可解除对程序区和信息区的保护，处于保护状态时，无法进行擦除和编程的操作。

通过检查 MSC_FLASHKEY.STATUS 是否为 0，判断 Flash 是否处于保护状态。

5.4.1.2 Flash写保护区

Flash 存储器可以通过 WRP0_START、WRP0_END 和 WRP1_START、WRP1_END 设置两段写保护区域；通过 WRP0_ENB 和 WRP1_ENB 配置两段写保护区域使能。

Flash 页擦除和 Flash 字编程，无法对写保护区擦除和写入，Flash 全擦时，可以将写保护区数据清除。

写保护区域	使能	起始页号	结束页号
区域 1	WRP0_ENB	WRP0_START	WRP0_END
区域 2	WRP1_ENB	WRP1_START	WRP1_END

表 5-1 写保护区配置字对应表

5.4.1.3 Flash私有代码读保护区

Flash 存储器可以通过 PCROP0_START、PCROP0_END 和 PCROP1_START、PCROP1_END 配置两段私有代码读保护区；通过 PCROP0_ENB 和 PCROP1_ENB 配置使能。

Flash 对私有读保护区进行任何非法的读取或擦写，均会置位对应的错误标识；但是可以进行 Flash 全擦操作，且 Flash 全擦时，会将私有代码读保护区数据清除。

私有读保护区	使能	起始页号	结束页号
区域 1	PCROP0_ENB	PCROP0_START	PCROP0_END
区域 2	PCROP1_ENB	PCROP1_START	PCROP1_END

表 5-2 私有代码读保护区配置字对应表

5.4.1.4 Data Flash区

Flash 区域可以通过配置字 DAFLS 划分 Data Flash 区，通过 DAFLS_ENB 配置 Data Flash 的使能。

Data Flash	使能	起始页号	结束页号
区域 1	DAFLS_ENB	DAFLS_START	DAFLS_END

表 5-3 Data Flash 配置字对应表

5.4.1.5 Flash全局读保护

Flash 存储器可以进行全局读保护，保护等级分为 Level0，Level1，Level2。

当全局保护字为 32 位全 1 时，全局保护级别即为 Level0。

当全局保护字高 16 位为全 1 且低 16 位为非全 1 时，全局保护级别即为 Level1。

当全局保护字高 16 位为非全 1 且低 16 位也为非全 1 时，全局保护级别即为 Level2。

不同全局加密保护级别下的访问限制如下表：

存储区		全局保护级别	调试模式			用户模式					
						在 Flash 中运行			在 SRAM 中运行		
			擦	写	读[1]	擦[4]	写[4]	读[2]	擦	写	读[1]
Code Flash	非保护区	Level0	全擦/页擦	是	是	否	否	是	页擦[5]	是[5]	是
		Level1	全擦/页擦	否	否	否	否	是	页擦[5]	是[5]	否
		Level2	否	否	否	否	否	是	页擦[5]	是[5]	否
	私有代码读保护区	Level0	全擦	否	否	否	否	否	否	否	否
		Level1	全擦	否	否	否	否	否	否	否	否
		Level2	否	否	否	否	否	否	否	否	否
	写保护区	Level0	全擦	是	是	否	否	是	否	否	是
		Level1	全擦	否	否	否	否	是	否	否	否
		Level2	否	否	否	否	否	是	否	否	否
配置区（Info 1）		Level0	页擦	是	是	否	否	是	页擦	是[6]	是
		Level1	条件页擦[3]	是	是	否	否	是	否	是[6]	是
		Level2	否	否	否	否	否	否	否	是[6]	否
加密区[7]（Info 2）		Level0	页擦	是	是	否	否	是	页擦	是[6]	是
		Level1	条件页擦[3]	是	是	否	否	是	否	是[6]	是
		Level2	否	否	否	否	否	否	否	是[6]	否

表 5-4 不同全局保护级别下的访问限制表

注 1：调试模式或在 SRAM 中运行程序时，在 Level 1 和 Level 2 时，不能读取 Code Flash,也不能在 Level2 时读取 Info1，除此之外均可以正常读取。

注 2：用户模式下，Flash 中运行程序时，加密分区内程序可运行但不能被读取，其他区域均可以被读取。

注 3：在 Level 1 时，所有对于 Info1/Info2 的擦除操作必须紧跟在对 Code 区进行全擦操作之后才能正确擦除。

注 4：禁止所有 Flash 运行程序对 Flash 本身的擦写操作。

注 5：用户模式下，在 SRAM 或 IAPROM 中运行程序可擦写非保护非加密 Code Flash 区。

注 6：用户模式下，在 SRAM 或者 IAPROM 中可对 Info1/ Info2 写操作。

注 7： Level 2 等级禁止所有模式下对 Info2 擦写操作，Level 1 等级禁止用户模式下对 Info1 擦操作，而允许调试模式对 Info1/Info2 的写操作和有条件擦操作（如注 3）。

注 8：用户模式下，不支持对 Flash 的全擦和非私有代码区全擦命令。

5.4.2 Flash程序区全擦除

程序区全擦除可擦除全部程序区空间，一次全擦除耗时约 10ms。具体步骤如下：

1. 查 MSC_FLASHSR.BUSY 标志是否处于空闲状态；
2. 通过 MSC_FLASHKEY 解除 Flash 程序区保护状态；
3. 写入程序区的首地址；
4. 写入 MSC_FLASHCMD.CMD 命令触发全擦除；
5. 等待 MSC_FLASHSR.BUSY 标志再次变为空闲状态；
6. 判断 MSC_FLASHSR.MASE 标志位是否置起。

5.4.3 Flash页擦除

页擦除可擦除固定一页空间，其中程序区一页大小为 1024 Bytes，信息区一页大小为 512 Bytes，一次页擦除耗时约 2ms。具体步骤如下：

1. 检查 MSC_FLASHSR.BUSY 标志是否处于空闲状态；
2. 通过 MSC_FLASHKEY 解除 Flash 程序区或信息区保护状态；
3. 写入需擦除页的首地址；
4. 设置信息区是否需使能；
5. 写入 MSC_FLASHCMD.CMD 命令触发页擦除；
6. 等待 MSC_FLASHSR.BUSY 标志再次变为空闲状态；
7. 判断 MSC_FLASHSR.SERA 标志位是否置起。

注：Data Flash 页擦除流程与 Flash 页擦除流程一致，仅触发命令不同。

5.4.4 Flash字编程

程序区字编程可一次编程 8 Bytes 空间，信息区字编程可一次编程 4 Bytes 空间，一次字编程耗时约 30us。具体步骤如下：

1. 检查 MSC_FLASHSR.BUSY 标志是否处于空闲状态；
2. 通过 MSC_FLASHKEY 解除 Flash 程序区或信息区保护状态；
3. 写入需编程地址；
4. 设置信息区是否需使能；
5. 写入需编程数据，程序区需同时写入 MSC_FLASHDL.DATAL 和 MSC_FLASHDH.DATAH，信息区只需写入 MSC_FLASHDL.DATAL；
6. 写入 MSC_FLASHCMD.CMD 命令触发字编程；
7. 等待 MSC_FLASHSR.BUSY 标志再次变为空闲状态；

8. 判断 MSC_FLASHSR.PROG 标志位是否置起。

注：Data Flash 字编程流程与 Flash 字编程流程一致，仅触发命令不同，用户可根据擦除地址所在的区域选择相应的字编程命令字写入 FLASHCMD 寄存器触发字编程。区域类型分为普通 Flash 区和数据 Flash 区。数据 Flash 区通过数据 Flash 配置字（CFG_DAFLS）设置，若未设置，则整个 Flash 区域均为普通 Flash 区。若当前字编程地址所在区域和命令字对应区域不匹配时，触发无效，并产生 WAE 错误标志位。

5. 4. 5 Flash快速编程

快速编程可一次最多编程 512 Bytes 空间（每页 1KBytes），并且编程块的首末地址不可横跨相邻的两页或每一页上下半个区间，否则编程会产生异常。若有跨半页的编程需求，需要在半页的首地址启动新的快速编程操作。快速编程可比连续的字编程节约 40%的时间。

具体步骤如下：

1. 检查 BUSY 标志是否处于空闲状态；
2. 解除 Flash 程序区或信息区保护状态；
3. 写入需编程首地址；
4. 设置信息区是否需使能；
5. 设置编程数据长度；
6. 写入第一个编程数据，需同时写入 FLASHDL 和 FLASHDH；
7. 写入 MSC_FLASHCMD.CMD 命令触发快速编程；
8. 等待 MSC_FLASHSR.FASTPREQ 请求，同时判断无 FASTPERR；
9. 写入下一个编程数据；
10. 判断是否是最后一个数据，不是重复 8~10；
11. 等待 BUSY 标志再次变为空闲状态；
12. 判断 FASTP 标志位是否置起，判断有无 FASTPERR。

注：Data Flash 快速编程流程与 Flash 快速编程流程一致，仅触发命令不同，用户可根据擦除地址所在的区域选择相应的快速编程命令字写入 FLASHCMD 寄存器触发快速编程。区域类型分为普通 Flash 区和数据 Flash 区。数据 Flash 区通过数据 Flash 配置字（CFG_DAFLS）设置，若未设置，则整个 Flash 区域均为普通 Flash 区。若当前快速编程地址所在区域和命令字对应区域不匹配时，触发无效，并产生 WAE 错误标志位。

5. 4. 6 Flash编程数据FIFO

Flash 编程数据 FIFO 可通过 FIFOEN 使能，该 FIFO 为写入 FIFO，读取无效。当数据写入 FIFO 后，可在 FLASHDL 和 FLASHDH 寄存器中读取到写入的值，优先写入的是低位数据。在 FIFO 中写入 2 个字数据时，可触发一次编程。FIFO 中使用字节和半字写入无效。

若 FIFOFP 未使能，则写入 FIFO 后触发字编程，若 FIFOFP 使能，则写入 FIFO 后触发快速编程。字编程和快速编程的其他设置参考普通编程方式。

5.4.7 IAP自编程硬件固化模块

芯片内置 IAP 自编程固化模块，由硬件电路实现，在 IAP 自编程操作程序中可以调用这些自编程固化模块，以减少 SRAM 中的 IAP 操作代码量。

IAP 自编程硬件固化模块支持页擦，单字编程，双字编程和多字编程，分别由如下 IAP 操作函数来实现：

5.4.7.1 页擦函数

- 函数功能：擦除指定的页
- 入口地址：0x10000004
- 输入参数：R0-擦除页的首地址
- 返回值：R0-函数执行状态（R0=1 为成功，R0=0 为失败）

5.4.7.2 单字编程函数

- 函数功能：向 Flash 指定地址写入一个字(32 位)
- 入口地址：0x10000008
- 输入参数：R0-待编程的 Flash 地址，R1-待编程数据
- 返回值：R0-函数执行状态（R0=1 为成功，R0=0 为失败）

5.4.7.3 双字编程函数

- 函数功能：向 Flash 指定地址写入两个字(64 位)
- 入口地址：0x1000000C
- 输入参数：R0-待编程的 Flash 地址，R1-待编程数据低 32 位，R2-待编程数据高 32 位
- 返回值：R0-函数执行状态（R0=1 为成功，R0=0 为失败）

5.4.7.4 多字编程

- 函数功能：向 Flash 指定地址写入多个字
- 入口地址：0x10000000
- 输入参数：R0-待编程的 Flash 首地址，R1-放在 SRAM 空间的编程数据首地址，R2-编程数据长度，R3-当编程到页首时是否先进行页擦除（R3 非零为擦除，R3=0 为不擦除）
- 返回值：R0-函数执行状态（R0=1 为成功，R0=0 为失败）

5.5 特殊功能寄存器

5.5.1 寄存器列表

MSC 寄存器列表		
名称	偏移地址	描述
MSC_FLASHKEY	000_H	Flash 程序区操作关键码寄存器
MSC_INFOKEY	004_H	Flash 信息区操作关键码寄存器
MSC_FLASHADDR	008_H	Flash 擦除编程地址寄存器
MSC_FLASHFIFO	$00C_H$	Flash 编程数据写缓存寄存器
MSC_FLASHDL	010_H	Flash 编程数据低字寄存器
MSC_FLASHDH	014_H	Flash 编程数据高字寄存器
MSC_FLASHCMD	018_H	Flash 操作命令寄存器
MSC_FLASHCR	$01C_H$	Flash 控制寄存器
MSC_FLASHSR	020_H	Flash 状态寄存器
MSC_FLASHFPL	024_H	Flash 快速编程长度寄存器
MSC_MEMWAIT	028_H	存储器读取等待时间寄存器

5. 5. 2 寄存器描述

5. 5. 2. 1 Flash程序区关键码寄存器（MSC_FLASHKEY）

Flash 程序区关键码寄存器（MSC_FLASHKEY）																															
偏移地址：00_H																															
复位值：$00000000\_00000000\_00000000\_00000011_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														STATUS	

Reserved	Bit 31-2	—	保留
STATUS	Bit 1-0	R	**Flash 程序区状态位** 00：可擦除或编程 其他：被保护，不可擦除或编程 注：IAP复位可将该寄存器复位

注：对上述该寄存器连续写入 0x8ACE0246 和 0x9BDF1357 可去除保护，写入其他值或中间插入其他操作将失效。

5. 5. 2. 2 Flash信息区关键码寄存器（MSC_INFOKEY）

Flash 信息区关键码寄存器（MSC_INFOKEY）																															
偏移地址：04_H																															
复位值：$00000000\_00000000\_00000000\_00000011_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														STATUS	

Reserved	Bit 31-2	—	保留
STATUS	Bit 1-0	R	**Flash 信息区状态位** 00：可擦除或编程 其他：被保护，不可擦除或编程 注：IAP复位可将该寄存器复位

注：对上述该寄存器连续写入 0x7153BFD9 和 0x0642CEA8 可去除保护，写入其他值或中间插入其他操作将失效。

5. 5. 2. 3 Flash擦除编程地址寄存器（MSC_FLASHADDR）

Flash 擦除编程地址寄存器（MSC_FLASHADDR）																															
偏移地址：08_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												IFREN	ADDR																		

Reserved	Bit 31-20	—	**保留**
IFREN	Bit 19	R/W	**信息区使能** 0：禁止 1：使能
ADDR	Bit 18-0	R/W	**Flash 地址**

关于上述寄存器中的 ADDR 位：

注 1：低 3 位写入无效，读出始终为 0。

注 2：页擦除完成后，地址自动加 0x400。

注 3：字编程完成后，地址自动加 8。

注 4：快速编程完成后，地址自动更新到快速编程区域之后的下一个地址。

5. 5. 2. 4 Flash编程FIFO寄存器（MSC_FLASHFIFO）

Flash 编程 FIFO 寄存器（MSC_FLASHFIFO）																															
偏移地址：$0C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFO																															

FIFO	Bit 31-0	W	**Flash编程FIFO**

注 1：需通过 FIFOEN 使能 FIFO，写入编程数据，可适用于 DMA 传输数据，先写入低 32 位数据，再写入高 32 位数据。

注 2：当写入相应个数数据后，将自动触发字编程。

5. 5. 2. 5 Flash编程数据低字寄存器（MSC_FLASHDL）

Flash 编程数据低字寄存器（MSC_FLASHDL）																															
偏移地址：10_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAL																															

DATAL	Bit 31-0	R/W	**Flash编程数据低字**

5. 5. 2. 6 Flash编程数据高字寄存器（MSC_FLASHDH）

Flash 编程数据高字寄存器（MSC_FLASHDH）																															
偏移地址：14_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAH																															

DATAH	Bit 31-0	R/W	**Flash编程数据高字**

5. 5. 2. 7 Flash命令寄存器（MSC_FLASHCMD）

Flash 命令寄存器（MSC_FLASHCMD）																															
偏移地址：18_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMD																															

CMD	Bit 31-0	W	**Flash操作触发命令字** 0x000051AE：Flash全擦除 0x00005EA1：普通Flash区页擦除 0x00005DA2：普通Flash区字编程 0x00005CA3：普通Flash区快速编程 0x00005BA4：数据Flash区页擦除 0x00005AA5：数据Flash区字编程 0x000059A6：数据Flash区快速编程 0x000050AF：非私有代码保护区全擦除 其他：保留

注：私有代码保护区通过私有代码读出保护配置字（CFG_PCROPx）设置，如果未设置私有代码保护区，建议使用全擦除命令。

5. 5. 2. 8 Flash控制寄存器（MSC_FLASHCR）

Flash 控制寄存器（MSC_FLASHCR）																													
偏移地址：$1C_H$																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								FIFODFS	FIFOFP	FIFOEN	FLASHREQ	Reserved		IAPRST	IAPEN

Reserved	Bit 31-8	—	**保留**
FIFODFS	Bit 7	R/W	**FIFO 快速编程区域选择位** 0：普通 Flash 区 1：数据Flash区
FIFOFP	Bit 6	R/W	**FIFO 快速编程使能位** 0：禁止 1：使能
FIFOEN	Bit 5	R/W	**FIFO 使能** 0：禁止 1：使能
FLASHREQ	Bit 4	R/W	**Flash 操作请求使能** 0：禁止 1：使能
Reserved	Bit 3-2	—	**保留**
IAPRST	Bit 1	W1	**自编程复位** 0：无操作 1：复位自编程
IAPEN	Bit 0	R/W	**自编程使能** 0：禁止 1：使能

5. 5. 2. 9 Flash状态寄存器（MSC_FLASHSR）

Flash 状态寄存器（MSC_FLASHSR）																															
偏移地址：20_H																															
复位值：$000x0000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADERR	MEPRT	SEPRT	PGPRT	AERA	DAFLS	MERF	FERF	Reserved						FEEND	FEBUSY	Reserved					FASTP	FASTPERR	FASTPREQ	TIMEOUT	PROG	SERA	MASE	WAE	WPE	BUSY	FLASHACK

ADERR	Bit 31	R	**当前地址错误状态位** 0：无错误 1：错误 注：当前地址处于非Flash合法区域为错误
MEPRT	Bit 30	R	**当前全擦除保护状态位** 0：无保护 1：保护 注：当前处于全擦除保护状态时全擦除无效
SEPRT	Bit 29	R	**当前页擦除保护状态位** 0：无保护 1：保护 注：当前处于页擦除保护状态时页擦除无效
PGPRT	Bit 28	R	**当前编程保护状态位** 0：无保护 1：保护 注：当前处于编程保护状态时自编程和快速编程无效
AERA	Bit 27	R	**当前地址区域状态位** 0：普通 Flash 区域 1：数据 Flash 区域 注：数据Flash区域通过数据Flash配置字（CFG_DAFLS）设置，若未设置，则整个Flash区域均为普通Flash区域
DAFLS	Bit 26	R	**上一次操作区域状态位** 0：上一次操作普通 Flash 区域 1：上一次操作数据Flash区域
MERF	Bit 25	R	**全擦除状态位** 0：未执行过全擦除 1：已执行过全擦除 注：该位置1则表示硬件允许擦除信息区的第1页和第2页
FERF	Bit 24	R	**非私有代码保护区域全擦除状态位** 0：未执行过非私有代码保护区域全擦除

			1：已执行过非私有代码保护区域全擦除 注：该位置1则表示硬件允许擦除信息区的第1页
Reserved	Bit 23-18	—	**保留**
FEEND	Bit 17	R	**非私有代码保护区域全擦除完成标志位** 0：未进行或正在进行中 1：已完成 注：重新启动新的擦除或编程操作时自动清除
FEBUSY	Bit 16	R	**非私有代码保护区域全擦除运行状态位** 0：空闲 1：正在进行
Reserved	Bit 15-11	—	**保留**
FASTP	Bit 10	R	**快速编程完成标志位** 0：未进行或正在进行中 1：已完成 注：重新启动新的擦除或编程操作时自动清除
FASTPERR	Bit 9	R	**快速编程错误标志位** 0：无错误 1：发生错误 注1：未在规定时间响应数据请求后将停止自动编程，并产生错误标志 注2：重新启动新的擦除或编程操作时自动清除
FASTPREQ	Bit 8	R	**快速编程数据请求标志位** 0：无请求 1：请求下一次编程数据 注：填写数据后自动清除
TIMEOUT	Bit 7	R	**超时错误标志** 0：无错误 1：发生错误 注：未在规定时间内完成相应擦除或编程动作时产生错误标志，可能硬件发生了故障，需软件触发一次IAP复位
PROG	Bit 6	R	**字编程完成标志** 0：未进行或正在进行中 1：已完成 注：重新启动新的擦除或编程操作时自动清除
SERA	Bit 5	R	**页擦除完成标志** 0：未进行或正在进行中 1：已完成 注：重新启动新的擦除或编程操作时自动清除
MASE	Bit 4	R	**程序区全擦除完成标志** 0：未进行或正在进行中 1：已完成 注：重新启动新的擦除或编程操作时自动清除

WAE	Bit 3	R	**擦除编程地址错误标志位** 0：无错误 1：发生错误 注：在非法地址触发擦除和编程时产生错误标志，或使用了与当前区域不匹配（数据Flash区域）的擦除和编程命令也可产生错误标志
WPE	Bit 2	R	**擦除编程保护错误标志位** 0：无错误 1：发生错误 注：在被保护的地址触发擦除和编程时产生错误标志
BUSY	Bit 1	R	**自编程状态位** 0：空闲 1：正在进行
FLASHACK	Bit 0	R	**Flash 操作许可状态** 0：禁止操作 1：允许操作

5. 5. 2. 10 Flash快速编程长度寄存器（MSC_FLASHFPL）

Flash 快速编程长度寄存器（MSC_FLASHFPL）

偏移地址：24_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	**保留**
LEN	Bit 15-0	R/W	**Flash 快速编程数据长度** 以双字为单位配置快速编程数据的数目

5. 5. 2. 11 存储器读取等待时间寄存器（MSC_MEMWAIT）

存储器读取等待时间寄存器（MSC_MEMWAIT）

偏移地址：28_H

复位值：$00000000\_00000000\_00000000\_00000010_B$

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

Reserved	Bit 31-10	—	**保留**
SRAM_W	Bit 9-8	R/W	**SRAM 读取等待时间** 00：无等待 01：1 个 SYSCLK 10：2 个 SYSCLK 11：3 个 SYSCLK
Reserved	Bit 7-4	—	**保留**
FLASH_W	Bit 3-0	R/W	**Flash 读取等待时间选择** 0000：无等待 0001：1 个 SYSCLK 0010：2 个 SYSCLK …… 1111：15 个 SYSCLK 读取等待时间的选择决定于系统运行频率大小，具体选择如下： 0 ~ 34MHz：无需等待 34~65MHz：等待至少 1 个 SYSCLK 65~96MHz：等待至少 2 个 SYSCLK 注：当有低功耗需求时，也可设置较大的等待时间可降低运行功耗

第6章 系统配置控制器

6.1 概述

系统配置模块（SYSCFG）用于芯片的系统级功能配置。

6.2 特性

- 支持寄存器写保护功能
- 支持存储器重映射功能
- 支持 USB 配置功能
- 支持定时器刹车源配置功能

6.3 功能描述

6.3.1 系统寄存器写保护

为避免程序的异常运行对系统级模块的误操作，SYSCFG_PROT 寄存器用于防止程序对系统级模块的误操作。该寄存器保护范围为除自身外的 SYSCFG、PMU、CMU、RMU 模块的所有寄存器。

SYSCFG_PROT 寄存器为虚拟寄存器，要对系统级模块其它寄存器进行写操作时，需先对 SYSCFG_PROT 寄存器写 0x55AA6996（即移除写保护功能），之后可对系统级模块其它寄存器进行写操作；对 SYSCFG_PROT 寄存器写入其他值重新进入写保护状态，在写保护状态下，对系统寄存器进行的写操作将被忽略。

可以通过读取 SYSCFG_PROT 寄存器的值，来确认系统级模块是否处于写保护状态：读出值为 0x00000000，表示当前可对系统级模块寄存器进行写操作；读出值为 0x00000001 表示系统级模块处于写保护状态。

注：SYSCFG_PROT 寄存器仅能读出 0 和 1，无其它读出值。

6.3.2 存储器重映射

存储器重映射应用于系统的 2 种启动模式：

- ◇ Boot Flash 启动
- ◇ 用户 Flash 启动

系统在发生上电复位、低电压复位或外部端口复位时，会根据 BOOT（CFG_WORD0[12]）配置情况，决定启动地址，详见章节《系统总线和存储器》中“启动引导”的介绍。

通过配置芯片配置字中的 Flash 启动地址选择位，来决定跳转至 Boot Flash 还是用户 Flash。用户根据实际需要编写 Boot Flash 的程序内容，在启动完成后清除 SYSCFG_MEMRMP.BFRMPEN 位，并自行跳转至用户 Flash 区域。

6.3.3 USB配置

通过配置 SYSCFG_USBCFG 寄存器可实现对 USB 相关配置功能。

6.3.4 定时器刹车源配置

通过配置 SYSCFG_TBKCFG 寄存器可选择相应的定时器刹车事件。

- ◇ CPU 锁死
- ◇ LVD 事件
- ◇ 时钟安全事件

6.4 特殊功能寄存器

6.4.1 寄存器列表

SYSCFG 寄存器列表		
名称	偏移地址	描述
SYSCFG_PROT	000_H	系统写保护寄存器
SYSCFG_MEMRMP	004_H	存储器重映射寄存器
—	008_H	—
SYSCFG_USBCFG	$00C_H$	USB 配置寄存器
SYSCFG_TBKCFG	010_H	定时器刹车源配置寄存器

6. 4. 2 寄存器描述

6. 4. 2. 1 系统写保护寄存器（SYSCFG_PROT）

系统写保护寄存器（SYSCFG_PROT）																															
偏移地址：000_H																															
复位值：$00000000\_00000000\_00000000\_00000001_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY																															PROT

KEY	Bit 31-1	W	**保护关键码** 0x55AA6996：去除写保护 其他：开启写保护
PROT	Bit 0	R	**保护状态位** 0：无写保护 1：写保护

6. 4. 2. 2 存储器重映射寄存器（SYSCFG_MEMRMP）

存储器重映射寄存器（SYSCFG_MEMRMP）																															
偏移地址：004_H																															
复位值：$00000000\_00000000\_00000001\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							BFRMPEN	Reserved							

Reserved	Bit 31-9	—	**保留**
BFRMPEN	Bit 8	R/W	**Boot Flash 映射使能位** 0：禁止 1：使能
Reserved	Bit 7-0	—	**保留**

6. 4. 2. 3 USB配置寄存器（SYSCFG_USBCFG）

USB 配置寄存器（SYSCFG_USBCFG）

偏移地址：$00C_H$

复位值：$00000000\_10101000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CLKRDY	CLKRDYBP	CKEN	ITRM		CKDIVN						Reserved				CKDIVM				Reserved				HSDRV		Reserved	TXLBSE

Reserved	Bit 31-27	—	**保留**
CLKRDY	Bit 26	R	**内部 PLL 时钟稳定标志位** 0：未稳定 1：已稳定
CLKRDYBP	Bit 25	R/W	**内部 PLL 时钟稳定标志旁路选择位** 0：未旁路 1：旁路 注：旁路稳定标志位时，CLKRDY 一直为 1
CKEN	Bit 24	R/W	**USB PHY 时钟源使能位** 0：禁止 1：使能
ITRM	Bit 23-22	R/W	**电流调整位** 00：30uA 01：40uA 10：50uA 11：60uA
CKDIVN	Bit 21-16	R/W	**时钟分频 N 选择位** 00000x：保留 000010：2 分频 000011：3 分频 000100：4 分频 …… 111111：63 分频
Reserved	Bit 15-12	—	**保留**
CKDIVM	Bit 11-8	R/W	**时钟分频 M 选择位** 0000：不分频 0001：保留 0010：2 分频 0011：3 分频 …… 1111：15 分频
Reserved	Bit 7-4	—	**保留**
HSDRV	Bit 3-2	R/W	**高速模式驱动选择位**

			00：正常驱动 …… 11：最大驱动
Reserved	Bit 1	—	**保留**
TXLBSE	Bit 0	R/W	**发送数据低位数据填充使能位** 0：禁止 1：使能 注：对应数据位 7-0

6. 4. 2. 4 定时器刹车源配置寄存器（SYSCFG_TBKCFG）

定时器刹车源配置寄存器（SYSCFG_TBKCFG）

偏移地址：010_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-3	—	**保留**
CLUBKE	Bit 2	R/W	**CPU 锁死作为定时器刹车源使能位** 0：禁止 1：使能
LVDBKE	Bit 1	R/W	**LVD 事件作为定时器刹车源使能位** 0：禁止 1：使能
CSSBKE	Bit 0	R/W	**时钟安全事件作为定时器刹车源使能位** 0：禁止 1：使能

第7章 电源管理及低功耗模式

7.1 概述

电源管理单元（PMU）管理芯片的电源以及低功耗模式。在每个低功耗模式下，都有其对应的单元模块状态（使能、禁止或掉电）。芯片可支持各种功耗模式：RUN，SLEEP，STOP1，STOP2，STANDBY。其中 RUN 为芯片正常运行模式，所有的外设模块均可被使能。其余为低功耗模式。STOP2 为 CPU 最低可恢复模式，在 STOP2 模式时，CPU 和大部分外设被禁止，所有 RAM 中数据保持，唤醒之后外设继续运行，CPU 从暂停处继续运行。STANDBY 模式会禁止 CPU 和除 POR、MRST 和备份 RTC 外所有的外设，备份域 RAM 数据保持，GPIO 状态保持。

低功耗模式通过软件操作使能。SLEEP，STOP1，STOP2 可通过一系列中断或事件唤醒回到 RUN 模式，STANDBY 仅可通过上电复位、备份域 RTC 中断、外部 WKUP 引脚上升沿或外部 MRST 复位唤醒回到 RUN 模式。

PMU 也可将不需要使用的 RAM 模块关闭以降低芯片功耗。

7.2 特性

- 支持多种低功耗模式配置
- 支持多种唤醒源灵活配置
- 快速的唤醒时间

7.3 结构框图

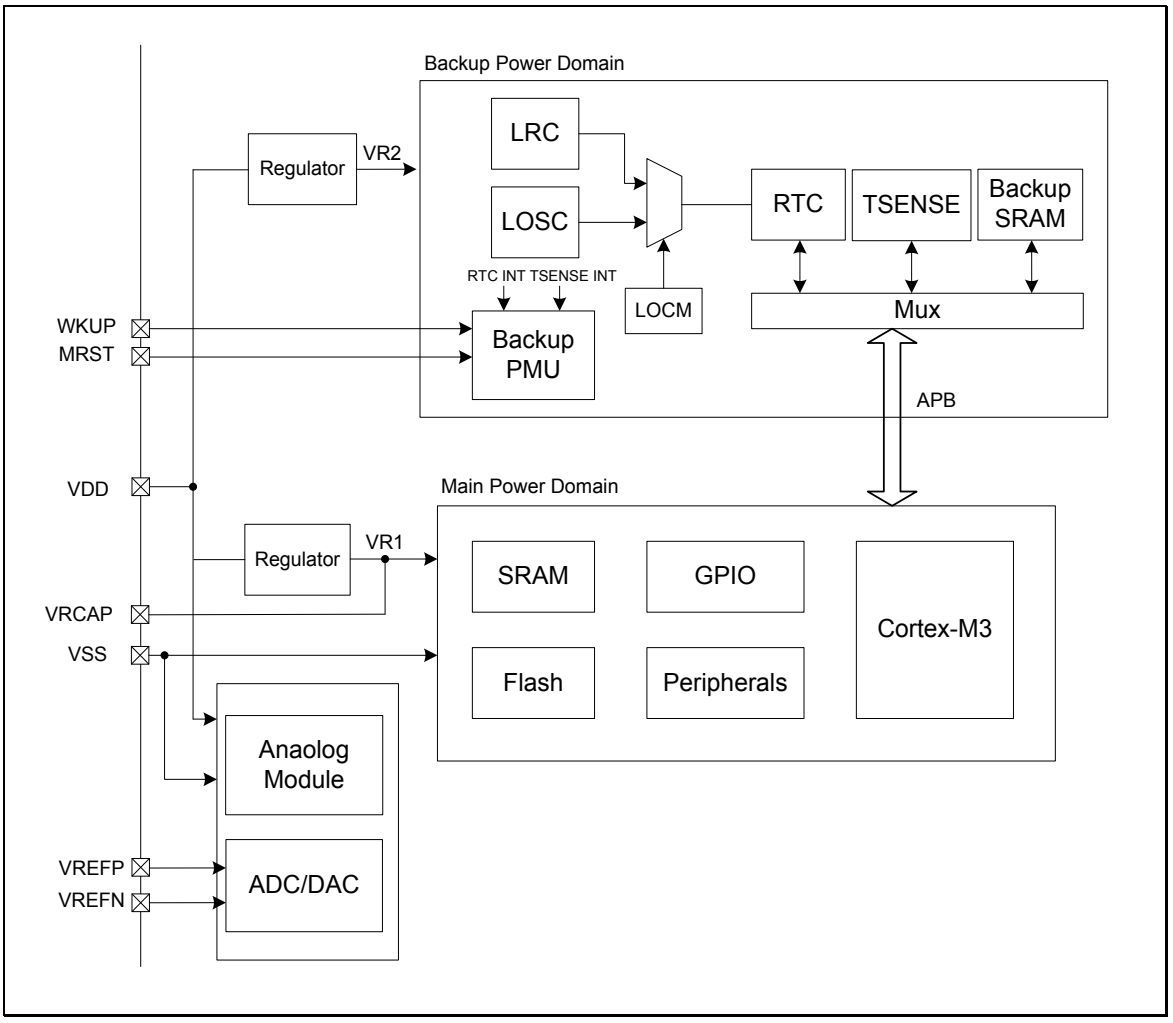


图 7-1 电源结构框图

7.4 功能描述

7.4.1 芯片电源

7.4.1.1 主系统电源域

芯片工作电压 VDD 要求介于 2.6V 到 5.5V 之间。片上调压稳压器用于提供内部 1.2V 数字电源，以及 Flash 编程和擦除专用的 1.8V 电源

7.4.1.2 独立的模拟模块电源和参考电压

ADC、DAC 电源由 VDD 提供。

ADC、DAC 参考电压可选 VDD 和 VREF（内部参考产生或者由 VREFP 端口输入）。

为了确保测量低电压时具有更高的精度，用户可以在 VREFP 上连接单独的 ADC 外部参考电压输入。VREFP 电压应介于 2.0 V 到 VDD 之间。

7.4.1.3 备份域电源

备份域电源为以下各模块供电：

- ◇ RTC
- ◇ LOSC
- ◇ LRC
- ◇ 温度传感器
- ◇ 备份域电源及时钟管理
- ◇ 备份 RAM

7. 4. 2 电源监视

7. 4. 2. 1 上电复位（POR）

芯片内部集成 POR 产生电路。

当 VDD 低于指定阈值 V_{POR} 时，器件无需外部复位电路便会保持复位状态。

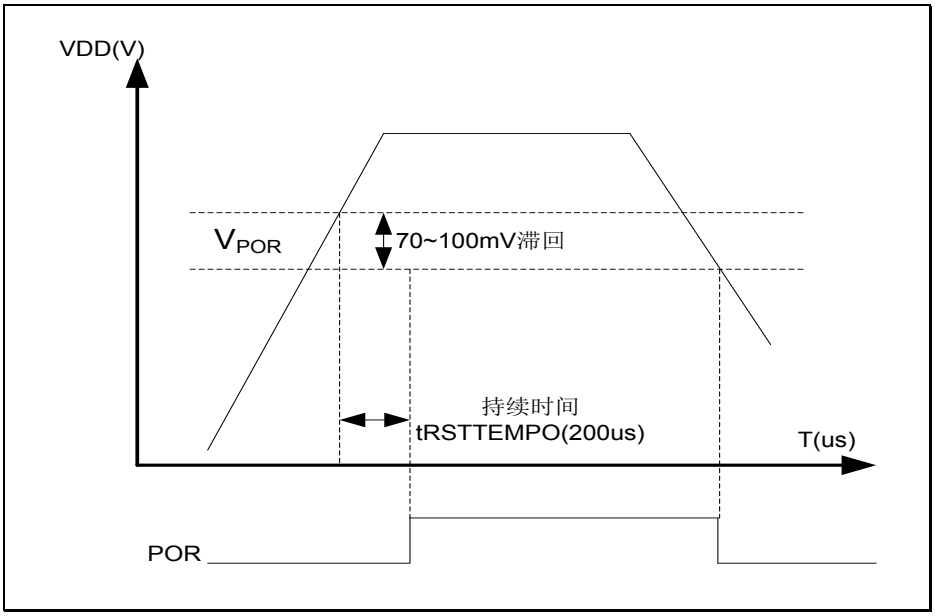


图 7-2 POR 示意图

7. 4. 2. 2 欠压复位（BOR）

上电期间，欠压复位（BOR）将使器件保持复位状态，直到电源电压达到 1.8V 以上。芯片默认 BOR 为开启状态，复位完成后，可通过软件选择 BOR 复位电压阈值 V_{BOR}，或可将 BOR 禁止。芯片支持 16 个 V_{BOR} 阈值选择。

当电源电压（VDD）降至所选 V_{BOR} 阈值以下时，将使器件复位。

BOR 阈值滞回电压约为 30 mV（电源电压的上升沿与下降沿之间）。

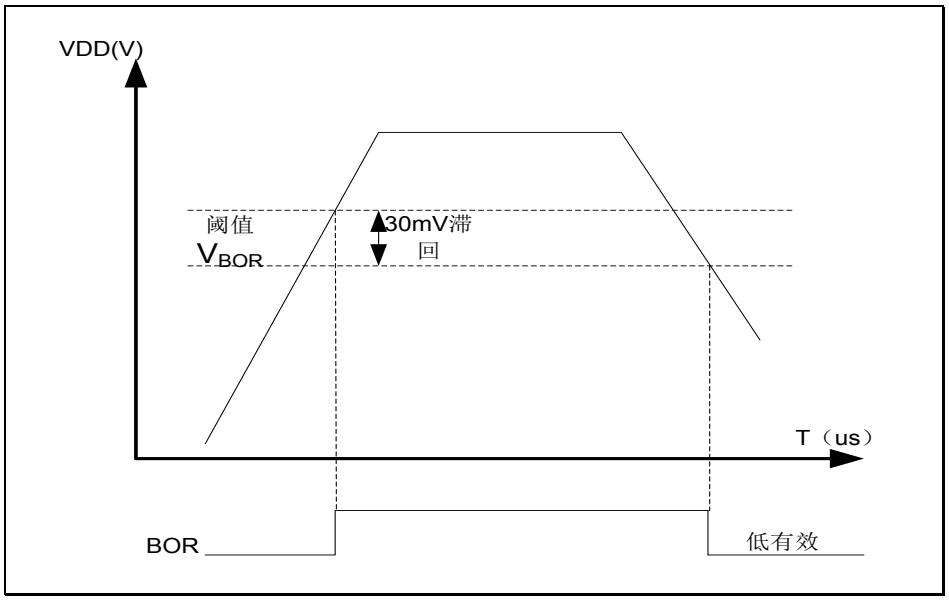


图 7-3 BOR 示意图

7. 4. 2. 3 低电压检测（LVD）

LVD 可用于监视 VDD 电源，通过设置 LVDEN 使能 LVD，将 VDD 电压和 LVDS 所选择的电压值进行比较，可粗略判断当前电源 VDD 的电压值。

LVD 也可检测外部引脚输入（LVDIN）的电压值。

LVD 提供了一个状态标志位 LVDO，用于指示 VDD 是大于还是小于 LVD 阈值。通过使能 LVDIE 可使能 LVD 中断，通过选择 LVDIFS 可选择 LVD 中断类型。当 VDD 降至 LVD 阈值以下或者当 VDD 升至 LVD 阈值以上时，可以产生 LVD 中断，具体取决于 LVDIFS 的中断类型配置。该功能的用处之一就是可以在 VDD 发生跌落时，立即进入中断服务程序中执行紧急关闭系统的任务，若外部有电池供电，则可进入低功耗模式并切换至电池供电。

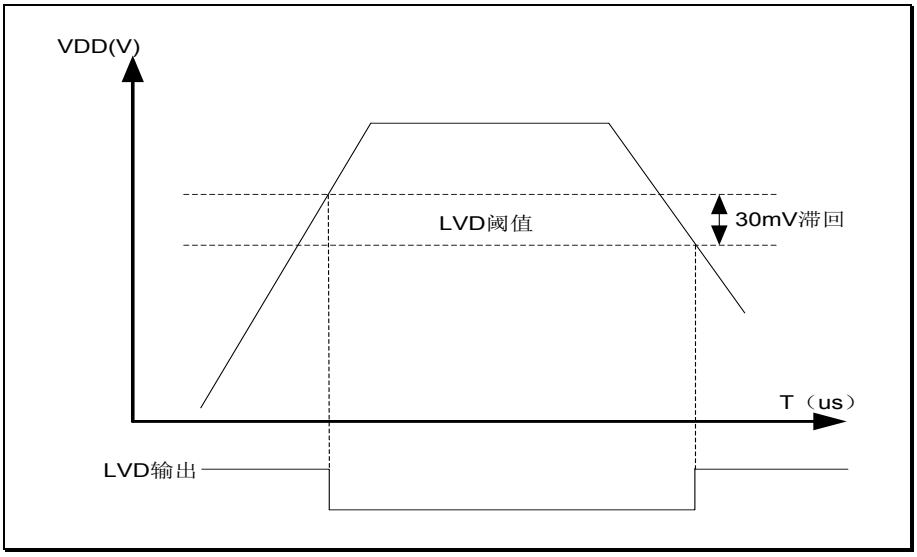


图 7-4 LVD 示意图

7. 4. 3 低功耗模式

7. 4. 3. 1 低功耗模式转换

默认情况下，系统复位或上电复位后，微控制器进入运行模式。在运行模式下，CPU 通过 HCLK 提供时钟，并执行程序代码。系统提供了多个低功耗模式，可在 CPU 不需要运行时（例如等待外部事件时）节省功耗。由用户根据应用选择具体的低功耗模式，以在低功耗、短启动时间和可用唤醒源之间寻求最佳平衡。

芯片支持以下低功耗模式：

◇ SLEEP 模式（Cortex-M3 内核停止，外设保持运行）

◇ STOP1 模式（DMA 仍可动作，可配合低功耗外设，PIS 等最小系统内动作）

◇ STOP2 模式（DMA 关闭，仅部分低功耗外设可工作）

◇ STANDBY 模式（1.2 V 主系统域断电）

此外，可通过下列方法之一降低运行模式的功耗：

◇ 降低系统时钟速度

◇ 不使用某个 APB 或 AHB 外设时，将对应的外设时钟关闭

进入低功耗模式的转换关系如下图所示：

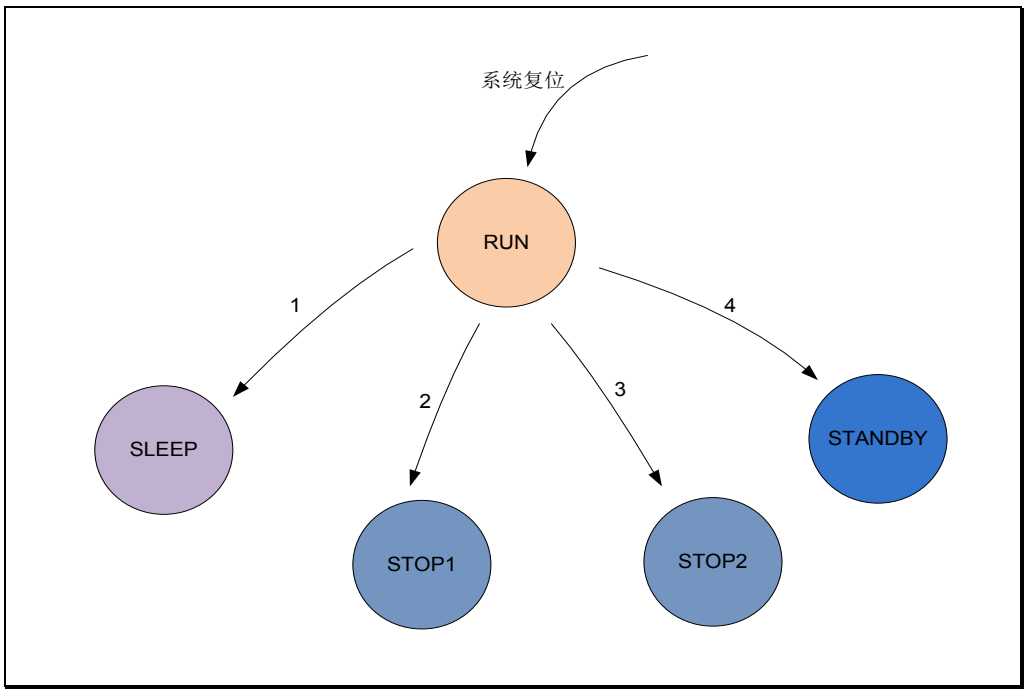


图 7-5 低功耗模式转换图

序号	模式	进入	唤醒	对逻辑电路时钟的影响	对时钟源的影响	主电源域1.2V稳压器
1	SLEEP	WFI WFE	任意中断 唤醒事件	CPU 时钟关闭	无	普通工作模式
2	STOP1	LPM 位=0 +DEEPSLEEP 位 +WFI/WFE	具体请参照章节“芯片配置指引”STOP1 低功耗模式的中断唤醒源	具体请参照表格：低功耗模式下各模块操作	PLL、HOSC 关闭 HRC 可使能	普通工作模式
3	STOP2	LPM 位=1 +DEEPSLEEP 位 +WFI/WFE	具体请参照章节“芯片配置指引”STOP2 低功耗模式的中断唤醒源		PLL、HOSC、HRC 关闭	普通工作模式或维持模式
4	STANDBY	LPM 位=2 +DEEPSLEEP 位 +WFI/WFE	具体请参照章节“芯片配置指引”STANDBY 低功耗模式的中断唤醒源		PLL、HOSC、HRC 关闭	关闭

表 7-1 低功耗模式说明

7.4.3.2 系统时钟速度

在运行模式下，可通过对预分频寄存器编程来降低系统时钟（SYSCLK、HCLK、PCLK1 和 PCLK2）速度。进入睡眠模式之前，也可以使用这些预分频器降低外设速度。也可将系统时钟切换至低速时钟源并关闭高速时钟源来降低功耗。

系统时钟速度的有关详细信息，请参见时钟管理。

7.4.3.3 外设时钟门控

在运行模式下，可通过设置时钟门控来停止各外设和存储器的总线时钟或模块工作时钟以降低功耗。

要进一步降低低功耗模式的功耗，可在执行 WFI 或 WFE 指令之前可通过门控关闭外设时钟。

外设时钟门控配置的有关详细信息，请参见时钟管理。

7.4.3.4 RUN模式

◇ 所有高速时钟源可使能

◇ 所有外设可使能

7. 4. 3. 5 SLEEP模式

- ◇ 所有高速时钟源可使能
- ◇ CPU 时钟被关断
- ◇ 所有外设可使能

7. 4. 3. 6 STOP1 模式

- ◇ 高速时钟源默认禁止
- ◇ CPU 时钟关闭
- ◇ DMA 可工作
- ◇ APB2 外设可工作
- ◇ SRAM 和寄存器值保持

7. 4. 3. 7 STOP2 模式

- ◇ 高速时钟源默认禁止
- ◇ CPU 时钟关闭
- ◇ ACMP，LVD，IWDT，WWDT，RTC，TSENSE 等可工作
- ◇ SRAM 和各寄存器数据保持

7. 4. 3. 8 STANDBY模式

- ◇ 主系统电源域掉电
- ◇ RTC，TSENSE 等可工作
- ◇ 备份域 SRAM 数据保持

7.4.3.9 低功耗模式下各模块操作

下表列举了各模块在低功耗模式下的操作可能性，为低功耗应用提供参考。

	SLEEP	STOP1	STOP2	STANDBY
内核				
NVIC	工作	停止	停止	掉电
调试	工作	工作	工作	掉电
存储器及存储器接口				
Flash	可配置为空闲模式或待机模式	待机模式	可配置为待机模式或掉电	掉电
SRAM	可配置是否掉电	可配置是否掉电	可配置是否掉电	掉电
EBI	可配置	停止	停止	掉电
QSPI	可配置	停止	停止	掉电
系统模块				
主域 1.2V 稳压器	普通模式	普通模式	可配置为普通模式或维持模式	掉电
主域 1.8V 稳压器	低功耗模式可配置	低功耗模式可配置	低功耗模式可配置	掉电
备份域稳压器	工作	工作	工作	工作
掉电检测	工作	工作	工作	工作
欠压检测	可配置	可配置	可配置	掉电
低电压检测	可配置	可配置	可配置	掉电
DMA 控制器	可配置	可配置	停止	掉电
外设互联	可配置	可配置	可配置	掉电
独立看门狗定时器	可配置	可配置	可配置	掉电
窗口看门狗定时器	可配置	可配置	可配置	掉电
时钟				
LOSC	可配置	可配置	可配置	可配置
HOSC	可配置	可配置	可配置	掉电
LRC	可配置	可配置	可配置	可配置
HRC	可配置	可配置	可配置	掉电
ULRC	工作	工作	工作	工作
内核时钟	停止	停止	停止	掉电
系统时钟	工作	工作	停止	掉电
外部接口				
GPIO	可配置	可配置	可配置	WAKEUP PIN 可唤醒
安全管理				
CRC	可配置	停止	停止	掉电
加密处理	可配置	停止	停止	掉电
真随机发生器	可配置	停止	停止	掉电
定时器				

	SLEEP	STOP1	STOP2	STANDBY
高级定时器	可配置	停止	停止	掉电
通用定时器	可配置	停止	停止	掉电
基本定时器	可配置	停止	停止	掉电
RTC	可配置	可配置	可配置	可配置
通信				
I2C 接口	可配置	停止	停止	掉电
串行外设接口 (SPI)	可配置	停止	停止	掉电
通用异步收发器 (UART)	可配置	停止	停止	掉电
控制区域网络 (CAN)	可配置	停止	停止	掉电
通用串行总线 (USB)	可配置	可配置	进入 STOP2 前需关闭 USB 电源	进入 STANDBY 前需关闭 USB 电源
模拟				
ADC	可配置	停止	停止	掉电
ACMP	可配置	可配置	可配置	掉电
DAC	可配置	可配置	停止	掉电

表 7-2　低功耗模式下各模块操作

7.5 特殊功能寄存器

7.5.1 寄存器列表

PMU 寄存器列表		
名称	偏移地址	描述
PMU_CR0	000_H	PMU 控制寄存器 0
PMU_CR1	004_H	PMU 控制寄存器 1
PMU_SR	008_H	PMU 状态寄存器
PMU_LVDCR	$00C_H$	LVD 控制寄存器
PMU_PWRCR	010_H	电源控制寄存器
PMU_VREFCR	020_H	VREF 控制寄存器

7. 5. 2 寄存器描述

7. 5. 2. 1 PMU控制寄存器 0（PMU_CR0）

PMU 控制寄存器 0（PMU_CR0）

偏移地址：00_H

复位值：$00000000\_01000011\_00000000\_00000000_B$

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

Reserved	Bit 31-22	—	**保留**
MTSTOP	Bit 21	R/W	**STOP2 模式主域 1.2V LDO 维持使能位** 0：禁止 1：使能 STOP1 模式下该位请写 0。
Reserved	Bit 20-16	—	**保留**
SFPMS	Bit 15	R/W	**SLEEP 模式下 Flash 功耗模式选择位** 0：Flash处于空闲模式 1：Flash处于待机模式
Reserved	Bit 14-4	—	**保留**
CSTANDBYF	Bit 3	W1	**STANDBY 标志清除位** 0：无操作 1：清除STANDBY标志
CWUF	Bit 2	W1	**WUF 标志清除位** 0：无操作 1：清除WUF标志
LPM	Bit 1-0	R/W	**低功耗模式选择位（CPU 配置为 Deepsleep 时有效）** 00：STOP1 01：STOP2 1x：STANDBY

7. 5. 2. 2 PMU控制寄存器 1（PMU_CR1）

PMU 控制寄存器 1（PMU_CR1）

偏移地址：04_H

复位值：$00000000\_00000000\_10000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																LDO18RDY	Reserved													LDO18MOD	

Reserved	Bit 31-16	—	**保留**
LDO18RDY	Bit 15	R	**主域1.8V LDO状态位** 0：1.8V LDO未稳定 1：1.8V LDO稳定
Reserved	Bit 14-2	—	**保留**
LDO18MOD	Bit 1-0	R/W	**SLEEP/STOP模式下主域1.8V LDO功耗模式配置位** 00：高驱动模式 01：低功耗模式 10：维持模式 11：关断模式 注：在STOP2维持模式使能即PMU_CR0.MTSTOP设置为1时，1.8V LDO须配置为维持模式或关断模式。

7. 5. 2. 3 PMU状态寄存器（PMU_SR）

PMU 状态寄存器（PMU_SR）																															
偏移地址：08_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													USBRDY	STANDBYF	WUF

Reserved	Bit 31-3	—	保留
USBRDY	Bit 2	R	**USB电源状态标志位** 0：USB 电源未准备好 1：USB 电源已准备好
STANDBYF	Bit 1	R	**STANDBY 标志位** 0：芯片未进入 STANDBY 模式 1：芯片复位之前已进入STANDBY模式 注：该位通过CSTANDBYF位来清零
WUF	Bit 0	R	**唤醒标志位** 0：未发生唤醒事件 1：使能 注：该位通过 CWUF 位来清零

7. 5. 2. 4 LVD控制寄存器（PMU_LVDCR）

LVD 控制寄存器（PMU_LVDCR）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																LVDO	Reserved			LVDFLT	LVDIFS			LVDS				LVDCIF	LVDIF	LVDIE	LVDEN

Reserved	Bit 31-16	—	**保留**
LVDO	Bit 15	R	**LVD 状态标志位** 0：大于阈值 1：小于阈值
Reserved	Bit 14-12	—	**保留**
LVDFLT	Bit 11	R/W	**LVD 滤波使能位** 0：禁止 1：使能 注：使能后 LVDO 稳定时间小于 100us 的变化将被忽略
LVDIFS	Bit 10-8	R/W	**LVD 中断标志产生模式选择位** 000：LVDO 上升沿产生中断 001：LVDO 下降沿产生中断 010：LVDO 高电平产生中断 011：LVDO 低电平产生中断 1xx：LVDO 变化（上升或下降沿）产生中断
LVDS	Bit 7-4	R/W	**LVD 电压阈值选择位** 0000：2.2V 0001：2.4V 0010：2.6V 0011：2.8V 0100：3.0V 0101：3.2V 0110：3.4V 0111：3.6V 1000：3.8V 1001：4.0V 1010：4.2V 1011：4.4V 1100：4.6V 1101：4.8V 111x：1.2V（使用 LVDIN 输入时固定为 1.2V） 其他：保留

LVDCIF	Bit 3	W1	**LVD 中断标志清除位** 0：无操作 1：清除LVD中断标志
LVDIF	Bit 2	R	**LVD 中断标志位** 0：LVDO 状态未变化 1：LVDO状态发生变化 注：该位由LVDCIF清除
LVDIE	Bit 1	R/W	**LVD 中断使能位** 0：禁止 1：使能
LVDEN	Bit 0	R/W	**LVD 使能位** 0：禁止 1：使能

7. 5. 2. 5 电源控制寄存器（PMU_PWRCR）

电源控制寄存器（PMU_PWRCR）

偏移地址：10_H

复位值：$00000000\_00000000\_00010111\_11111111_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			ROM	USB	QSPI	Reserved	BXCAN	Reserved		SRAM5	SRAM4	SRAM3	SRAM2	SRAM1	SRAM0

Reserved	Bit 31-13	—	**保留**
ROM	Bit 12	R/W	**ROM 电源使能位** 0：禁止 1：使能
USB	Bit 11	R/W	**USB 电源使能位** 0：禁止 1：使能
QSPI	Bit 10	R/W	**QSPI SRAM 电源使能位** 0：禁止 1：使能
Reserved	Bit 9	—	**保留**
BXCAN	Bit 8	R/W	**BXCAN SRAM 电源使能位** 0：禁止 1：使能
Reserved	Bit 7-6	—	**保留**
SRAM<y>	Bit 5-0	R/W	**SRAM<y>电源使能位** 0：禁止 1：使能

7. 5. 2. 6 VREF控制寄存器（PMU_VREFCR）

VREF 控制寄存器（PMU_VREFCR）

偏移地址：020_H

复位值：$00000000\_00000000\_01110100\_00000000_B$

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

Reserved	Bit 31-1	—	保留
VREFEN	Bit 0	R/W	**VREF 使能位** 0：禁止 1：使能

第8章 复位管理（RMU）

8.1 概述

系统复位可以由下面列出的任一事件触发。这些复位事件标志可以通过读取 RMU_RSTSR 寄存器来判断复位源。

8.2 特性

- 备份域支持 POR
- 主电源域支持 POR/BOR
- 支持外部端口复位 MRSTN
- 支持看门狗溢出复位
- 内核锁死（LOCKUP）复位
- 读取配置字复位标志位和配置字错误标志位
- 支持三种软件复位
  - 复位整个主电源域数字内核
  - CPU 复位
  - Cortex-M 内核请求系统复位
- 支持各外设模块的单独复位

8.3 结构框图

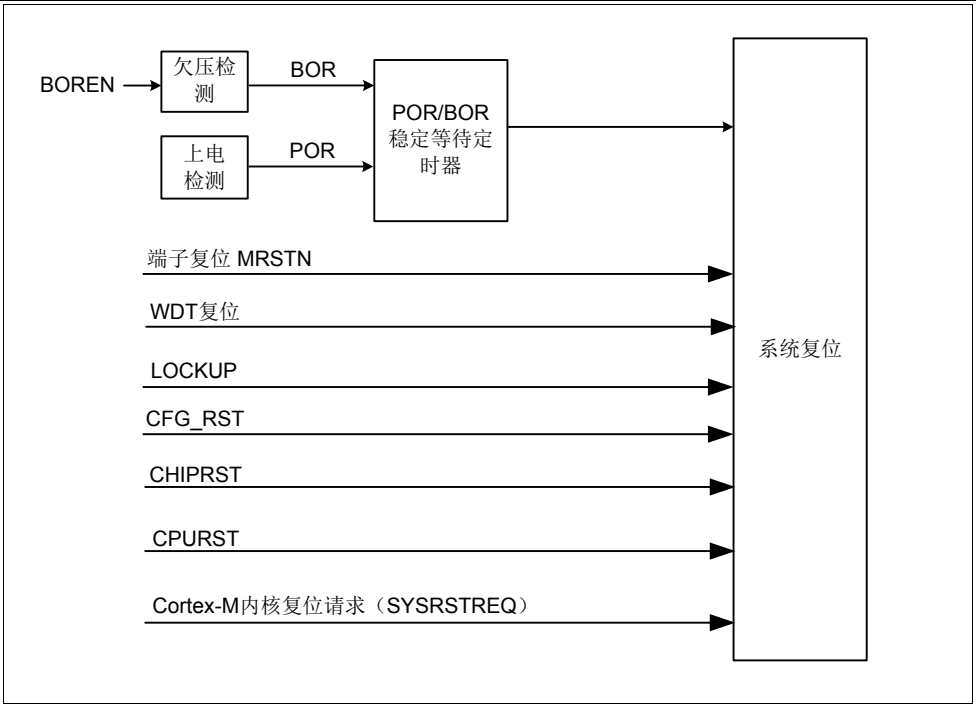


图 8-1 复位结构图

8.4 功能描述

ES32F36xx 微控制器支持 8 种复位源。CPURST 只复位 CPU 内核（不包含调试部分）；其他复位源则复位 CPU 内核和所有外设。各个复位源及寄存器关系如下表所示：

	POR	BOR
RMU_RSTSR	POR=1 WAKEUP=1	BOR=1
BOREN（RMU_CSR[0]）	0x1	0x1
LVDEN（LVDCR[0]）	0x0	0x0
BRRMPEN （SYSCFG_MEMRMP[0]）	0x1	0x1
BFRMPEN （SYSCFG_MEMRMP[8]）	0x1	0x1
SYS_STU（CMU_CSR）	0x1	0x1
CFT_STU（CMU_CSR）	0x0	0x0
HRCFSW（CMU_CFGR[24]）	0x0	0x0
系统和外设时钟分频	1 分频	1 分频
HOSCEN	0x0	0x0
LOSCEN	配置字	配置字
HRCEN	0x1	0x1
LRCEN	0x0	0x0
PLLEN	0x0	0x0
CMU_AHB1ENR	0x0000	0x0000
CMU_APB1ENR	0x0000_0000	0x0000_0000
CMU_APB2ENR	0x0000_0000	0x0000_0000
CPU 内核调试模块	复位值	复位值
备份域寄存器	备份域电源上电复位可复位； 主电源域上电复位不影响备份域	–
其他外设	复位值	

表 8-1 POR/BOR 复位与寄存器关系

	MRSTN	WDT
RMU_RSTSR	NMRST=1	WWDT=1 或 IWDT=1
BOREN（RMU_CSR[0]）	0x1	0x1
LVDEN（LVDCR[0]）	0x0	0x0
BRRMPEN（SYSCFG_MEMRMP[0]）	0x1	–
BFRMPEN（SYSCFG_MEMRMP[8]）	0x1	0x1
SYS_STU（CMU_CSR）	0x1	0x1
CFT_STU（CMU_CSR）	0x0	0x0
HRCFSW（CMU_CFGR[24]）	0x0	0x0
系统和外设时钟分频	1 分频	1 分频
HOSCEN	0x0	0x0
LOSCEN	配置字	配置字
HRCEN	0x1	0x1
LRCEN	0x0	0x0
PLLEN	0x0	0x0
CMU_AHB1ENR	0x0000	0x0000
CMU_APB1ENR	0x0000_0000	0x0000_0000
CMU_APB2ENR	0x0000_0000	0x0000_0000
CPU 内核调试模块	–	–
备份域寄存器	–	–
其他外设	复位值	

表 8-2 MRSTN/WDT 复位与寄存器关系

	LOCKUP	CHIPRST	SYSRSTREQ	CPURST
RMU_RSTSR	LOCKUP=1	CHIP=1	MCU=1	CPU=1
BOREN（RMU_CSR[0]）	0x1	0x1	0x1	–
LVDEN（LVDCR[0]）	0x0	0x0	0x0	–
BRRMPEN（SYSCFG_MEMRMP[0]）	–	0x1	–	–
BFRMPEN（SYSCFG_MEMRMP[8]）	0x1	0x1	0x1	–
SYS_STU（CMU_CSR）	–	0x1	–	–
CFT_STU（CMU_CSR）	–	0x0	–	–
HRCFSW（CMU_CFGR[24]）	–	0x0	–	–
系统和外设时钟分频	–	1 分频	–	–
HOSCEN	0x0	0x0	0x0	–
LOSCEN	配置字	配置字	配置字	–
HRCEN	0x1	0x1	0x1	–
LRCEN	0x0	0x0	0x0	–
PLLEN	0x0	0x0	0x0	–
CMU_AHB1ENR	0x0000	0x0000	0x0000	–
CMU_APB1ENR	0x0000_0000	0x0000_0000	0x0000_0000	–
CMU_APB2ENR	0x0000_0000	0x0000_0000	0x0000_0000	–
CPU 内核调试模块	–	复位值	–	–
备份域寄存器	–	–	–	–
其他外设	复位值			–

表 8-3　系统复位与寄存器关系

8.4.1 硬件复位

硬件复位包括上电复位，欠压复位，外部端口复位，LOCKUP 复位，WDT 复位和读取配置字错误复位。

8.4.1.1 上电复位

当 VDD 低于指定阈值 V_{POR} 时，器件无需外部复位电路便会保持复位状态。

8.4.1.2 欠压复位

上电期间，欠压复位（BOR）将使器件保持复位状态，直到电源电压达到 1.8V 以上。芯片默认 BOR 为开启状态，复位完成后，可通过软件选择 BOR 复位电压阈值 V_{BOR}，或可将 BOR 禁止。芯片支持 16 个 V_{BOR} 阈值选择。

当电源电压（VDD）降至所选 V_{BOR} 阈值以下时，将使器件复位。

8.4.1.3 端口复位

可复位除内核调试模块以外的芯片整体。

8.4.1.4 看门狗复位

详细描述请参照独立看门狗和窗口看门狗章节的说明。

可复位除内核调试模块以外的芯片整体，复位解除后，芯片正常从 Flash 启动。

8.4.1.5 LOCKUP复位

由不可恢复的异常导致的内核锁死，此时将产生复位信号来重新启动内核及系统。详细说明可参考 ARM 技术手册。

8.4.1.6 读取配置字错误标志位

在配置字加载或者程序运行过程中，由于异常干扰导致配置字的读取和控制出现错误，可能导致严重系统错误，此时将置位标志位，系统软件需要使芯片触发看门狗复位或芯片复位（CHIPRST）来重新加载配置字。

8.4.2 软件复位

8.4.2.1 芯片复位（CHIPRST）

芯片复位由寄存器 RMU_AHB2RSTR.CHIPRST 位控制，可复位整体芯片。

8.4.2.2 CPU复位（CPURST）

CPU 复位由寄存器 RMU_AHB2RSTR.CPURST 位控制，可复位 Cortex-M 内核（不包含调试部分）。

8.4.2.3 Cortex-M内核复位请求（SYSRSTREQ）

MCU 复位从 Cortex-M 内核产生。由应用中断和复位控制寄存器（Application Interrupt and Reset Control Register）的 SYSRESETREQ 位控制，将该位置 1 可对系统复位。详细可参考 ARM 相关技术手册。

8.4.2.4 外设软件复位

对应每个外设分别分配了一个软件复位。

AHB1 外设复位寄存器（RMU_AHB1RSTR）为 GPIO，CRC，CALC，CRYPT，TRNG，PIS，USB 等模块提供软件复位。

AHB2 外设复位寄存器（RMU_AHB2RSTR）作为 EBI，CPU 复位和芯片复位的控制寄存器。

APB1 外设复位寄存器（RMU_APB1RSTR）为定时器，UART，SPI，I2C，CAN 等 APB1 模块提供软件复位。

APB2 外设复位寄存器（RMU_APB2RSTR）为 ADC，ACMP，WWDT，IWDT，DAC，TSENSE 等 APB2 模块以及备份域相关寄存器等提供软件复位。

8.5 特殊功能寄存器

8.5.1 寄存器列表

RMU 寄存器列表		
名称	偏移地址	描述
RMU_CSR	000_H	RMU 控制状态寄存器
RMU_RSTSR	010_H	RMU 复位状态寄存器
RMU_CRSTSR	014_H	RMU 清复位状态寄存器
RMU_AHB1RSTR	020_H	AHB1 外设复位寄存器
RMU_AHB2RSTR	024_H	AHB2 外设复位寄存器
RMU_APB1RSTR	030_H	APB1 外设复位寄存器
RMU_APB1RSTR	034_H	APB2 外设复位寄存器

8. 5. 2 寄存器描述

8. 5. 2. 1 RMU控制状态寄存器（RMU_CSR）

RMU 控制状态寄存器（RMU_CSR）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000001_B$

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

Reserved	Bit 31-8	—	保留
BORVS	Bit 7-4	R/W	**BOR 电压点选择** 0000：2.0V 0001：2.0V 0010：2.2V 0011：2.4V 0100：2.6V 0101：2.8V 0110：3.0V 0111：3.2V 1000：3.4V 1001：3.6V 1010：3.8V 1011：4.0V 1100：4.2V 1101：4.4V 1110：4.6V 1111：4.8V
BORFLT	Bit 3-1	R/W	**BOR 滤波时钟选择** 00x：1 个 ULRC 周期 010：2个ULRC周期 011：3个ULRC周期 100：4个ULRC周期 101：5个ULRC周期 110：6个ULRC周期 111：7个ULRC周期
BOREN	Bit 0	R/W	**BOR 使能** 0：禁止 1：使能

8. 5. 2. 2 RMU复位状态寄存器（RMU_RSTSR）

RMU 复位状态寄存器（RMU_RSTSR）

偏移地址：10_H

复位值：$00000000\_00000000\_00000101\_00000011_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															CFGERR	Reserved					CFG	CPU	MCU	CHIP	LOCKUP	WWDT	IWDT	NMRST	BOR	WAKEUP	POR

Reserved	Bit 31-17	—	**保留**
CFGERR	Bit 16	R	**配置字错误状态标志** 0：无配置字错误 1：产生配置字错误 注：当程序运行过程中出现配置字错误时，软件需要触发看门狗复位或芯片全局复位来复位芯片
Reserved	Bit 15-11	—	**保留**
CFG	Bit 10	R	**配置字复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生
CPU	Bit 9	R	**软件 CPU 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生
MCU	Bit 8	R	**软件 MCU 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生 注：该复位从内核产生。由应用中断和复位控制寄存器（Application Interrupt and Reset Control Register）的SYSRESETREQ位控制，将该位置1可对系统复位。详细可参考ARM相关技术手册。
CHIP	Bit 7	R	**软件 CHIP 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生
LOCKUP	Bit 6	R	**LOCKUP 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生
WWDT	Bit 5	R	**WWDT 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生
IWDT	Bit 4	R	**IWDT 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生
NMRST	Bit 3	R	**NMRST 复位状态标志**

			0：无复位发生或标志位已被清除 1：有复位发生
BOR	Bit 2	R	**BOR 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生
WAKEUP	Bit 1	R	**唤醒复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生
POR	Bit 0	R	**POR 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生

8. 5. 2. 3 RMU清复位状态寄存器（RMU_CRSTSR）

RMU 清复位状态寄存器（RMU_CRSTSR）

偏移地址：14_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					CFG	CPU	MCU	CHIP	LOCKUP	WWDT	IWDT	NMRST	BOR	WAKEUP	POR

Reserved	Bit 31-11	—	**保留**
CFG	Bit 10	W1	**配置字复位标志清除** 0：无操作 1：清除标志
CPU	Bit 9	W1	**软件 CPU 复位标志清除** 0：无操作 1：清除标志
MCU	Bit 8	W1	**软件 MCU 复位标志清除** 0：无操作 1：清除标志
CHIP	Bit 7	W1	**软件 CHIP 复位标志清除** 0：无操作 1：清除标志
LOCKUP	Bit 6	W1	**LOCKUP 复位标志清除** 0：无操作 1：清除标志
WWDT	Bit 5	W1	**WWDT 复位标志清除** 0：无操作 1：清除标志
IWDT	Bit 4	W1	**IWDT 复位标志清除** 0：无操作 1：清除标志
NMRST	Bit 3	W1	**NMRST 复位标志清除** 0：无操作 1：清除标志
BOR	Bit 2	W1	**BOR 复位标志清除** 0：无操作 1：清除标志
WAKEUP	Bit 1	W1	**唤醒复位标志清除** 0：无操作 1：清除标志
POR	Bit 0	W1	**POR 复位标志清除** 0：无操作

			1：清除标志

8. 5. 2. 4 AHB1 外设复位寄存器（RMU_AHB1RSTR）

AHB1 外设复位寄存器（RMU_AHB1RSTR）

偏移地址：20_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-11	—	**保留**
USBRST	Bit 10	W1	**USB 复位** 0：无操作 1：复位
Reserved	Bit 9-6	—	**保留**
PISRST	Bit 5	W1	**PIS 复位** 0：无操作 1：复位
TRNGRST	Bit 4	W1	**TRNG 复位** 0：无操作 1：复位
CRYPTRST	Bit 3	W1	**CRYPT 复位** 0：无操作 1：复位
CALCRST	Bit 2	W1	**CALC 复位** 0：无操作 1：复位
CRCRST	Bit 1	W1	**CRC 复位** 0：无操作 1：复位
GPIORST	Bit 0	W1	**GPIO 复位** 0：无操作 1：复位

8. 5. 2. 5 AHB2 外设复位寄存器（RMU_AHB2RSTR）

AHB2 外设复位寄存器（RMU_AHB2RSTR）

偏移地址：24_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-9	—	**保留**
EBIRST	Bit 8	W1	**EBI 复位** 0：无操作 1：复位
Reserved	Bit 7-2	—	**保留**
CPURST	Bit 1	W1	**处理器内核复位** 0：无操作 1：复位 注：该复位只复位处理器内核（不包括 DEBUG 逻辑）
CHIPRST	Bit 0	W1	**芯片全局复位** 0：无操作 1：复位

8. 5. 2. 6 APB1 外设复位寄存器（RMU_APB1RSTR）

APB1 外设复位寄存器（RMU_APB1RSTR）

偏移地址：30_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						QSPIRST	CAN0RST	Reserved		I2C1RST	I2C0RST	Reserved	SPI2RST	SPI1RST	SPI0RST	Reserved		UART5RST	UART4RST	UART3RST	UART2RST	UART1RST	UART0RST	GP16C4T1RST	GP16C4T0RST	BS16T1RST	BS16T0RST	GP32C4T1RST	GP32C4T0RST	AD16C4T1RST	AD16C4T0RST

Reserved	Bit 31-26	—	**保留**
QSPIRST	Bit 25	W1	**QSPI 复位** 0：无操作 1：复位
CAN0RST	Bit 24	W1	**CAN0 复位** 0：无操作 1：复位
Reserved	Bit 23-22	—	**保留**
I2C1RST	Bit 21	W1	**I2C1 复位** 0：无操作 1：复位
I2C0RST	Bit 20	W1	**I2C0 复位** 0：无操作 1：复位
Reserved	Bit 19	—	**保留**
SPI2RST	Bit 18	W1	**SPI2 复位** 0：无操作 1：复位
SPI1RST	Bit 17	W1	**SPI1 复位** 0：无操作 1：复位
SPI0RST	Bit 16	W1	**SPI0 复位** 0：无操作 1：复位
Reserved	Bit 15-14	—	**保留**
UART5RST	Bit 13	W1	**UART5 复位** 0：无操作 1：复位
UART4RST	Bit 12	W1	**UART4 复位** 0：无操作 1：复位
UART3RST	Bit 11	W1	**UART3 复位**

			0：无操作 1：复位
UART2RST	Bit 10	W1	**UART2 复位** 0：无操作 1：复位
UART1RST	Bit 9	W1	**UART1 复位** 0：无操作 1：复位
UART0RST	Bit 8	W1	**UART0 复位** 0：无操作 1：复位
GP16C4T1RST	Bit 7	W1	**GP16C4T1 复位** 0：无操作 1：复位
GP16C4T0RST	Bit 6	W1	**GP16C4T0 复位** 0：无操作 1：复位
BS16T1RST	Bit 5	W1	**BS16T1 复位** 0：无操作 1：复位
BS16T0RST	Bit 4	W1	**BS16T0 复位** 0：无操作 1：复位
GP32C4T1RST	Bit 3	W1	**GP32C4T1 复位** 0：无操作 1：复位
GP32C4T0RST	Bit 2	W1	**GP32C4T0 复位** 0：无操作 1：复位
AD16C4T1RST	Bit 1	W1	**AD16C4T1 复位** 0：无操作 1：复位
AD16C4T0RST	Bit 0	W1	**AD16C4T0 复位** 0：无操作 1：复位

8. 5. 2. 7 APB2 外设复位寄存器（RMU_APB2RSTR）

APB2 外设复位寄存器（RMU_APB2RSTR）

偏移地址：34_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													BKPRAMRST	BKPCRST	TSENSERST	RTCRST	IWDTRST	Reserved	WWDTRST	ACMP2RST	Reserved	DACRST	Reserved	ACMP1RST	ACMP0RST	ADC1RST	ADC0RST	Reserved			

Reserved	Bit 31-19	—	**保留**
BKPRAMRST	Bit 18	W1	**BKPRAM 复位** 0：无操作 1：复位
BKPCRST	Bit 17	W1	**BKPC 复位** 0：无操作 1：复位
TSENSERST	Bit 16	W1	**TSENSE 复位** 0：无操作 1：复位
RTCRST	Bit 15	W1	**RTC 复位** 0：无操作 1：复位
IWDTRST	Bit 14	W1	**IWDT 复位** 0：无操作 1：复位
Reserved	Bit 13	—	**保留**
WWDTRST	Bit 12	W1	**WWDT 复位** 0：无操作 1：复位
ACMP2RST	Bit 11	—	ACMP2 **复位** 0：无操作 1：复位
Reserved	Bit 10	—	**保留**
DACRST	Bit 9	—	**DAC 复位** 0：无操作 1：复位
Reserved	Bit 8	—	**保留**
ACMP1RST	Bit 7	W1	**ACMP1 复位** 0：无操作 1：复位
ACMP0RST	Bit 6	W1	**ACMP0 复位**

			0：无操作 1：复位
ADC1RST	Bit 5	—	**ADC1 复位** 0：无操作 1：复位
ADC0RST	Bit 4	W1	**ADC0 复位** 0：无操作 1：复位
Reserved	Bit 3-0	—	**保留**

第9章 时钟管理（CMU）

9.1 概述

时钟管理单元（CMU）的作用是控制时钟和振荡器。MCU 各外设时钟可独立配置。外设时钟的灵活配置可以有效降低系统功耗。

9.2 特性

- 支持多种时钟源
  - 1~24MHz 外部高速晶体振荡器（HOSC）
  - 24MHz 或 2MHz 可配置内部高速 RC 振荡器（HRC）
  - 32.768KHz 外部低速晶体振荡器（LOSC）
  - 32.768KHz 内部低速 RC 振荡器（LRC）
  - 10KHz 内部超低速 RC 振荡器（ULRC）
  - 内部锁相环倍频时钟（PLL）
- 支持低功耗配置
- 快速启动时间
- AHB 外设、APB 外设和 CPU 可独立预分频
- 内核和外设均支持独立的时钟门控
- 支持系统时钟输出

9.3 结构框图

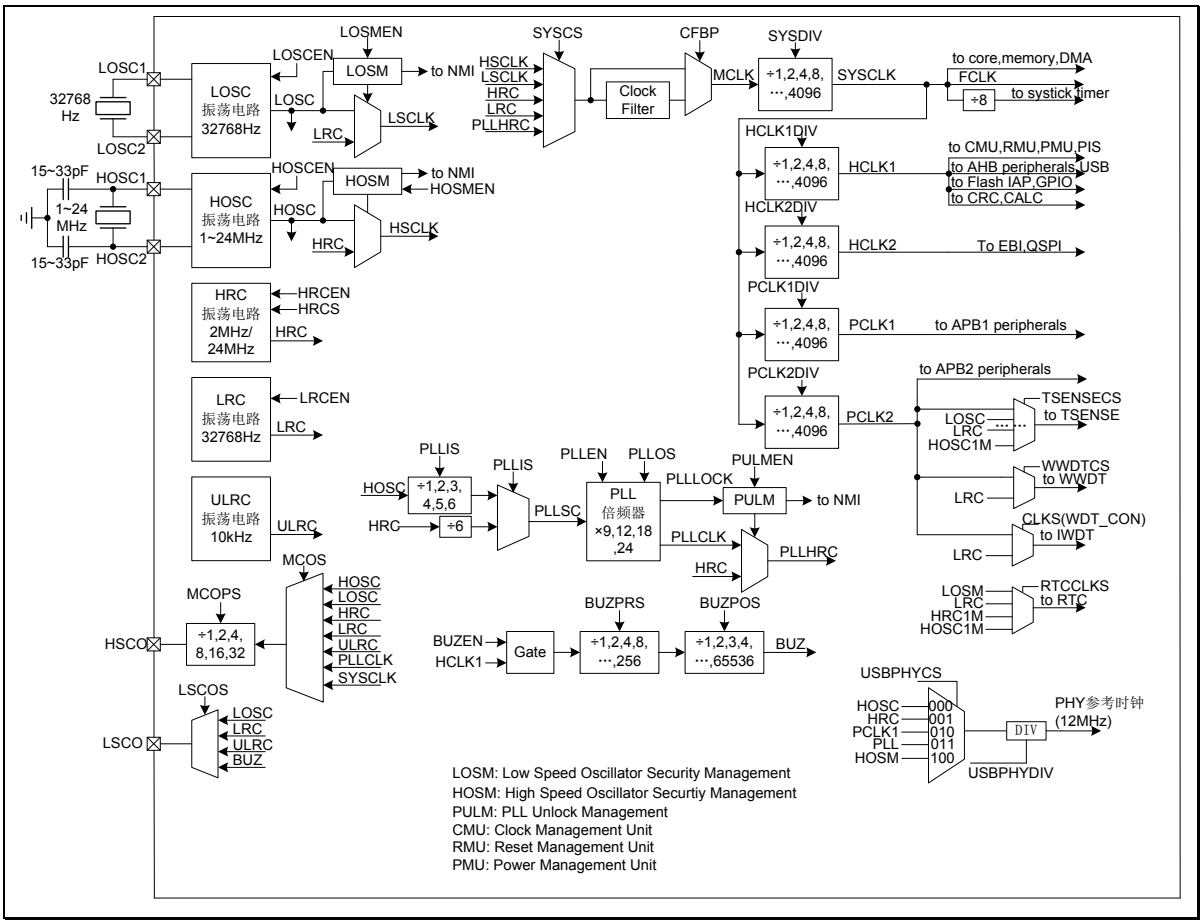


图 9-1 时钟管理结构图

9.4 功能描述

9.4.1 外部高速振荡器时钟（HOSC）

HOSC 高速振荡电路可驱动 1~24MHz 晶体振荡器和陶瓷振荡器。驱动晶体振荡器时需要匹配 15~33pF 电容，驱动陶瓷振荡器时不需要匹配电容。HOSC 内部自带反馈电阻，驱动外置振荡器不需要外接电阻。

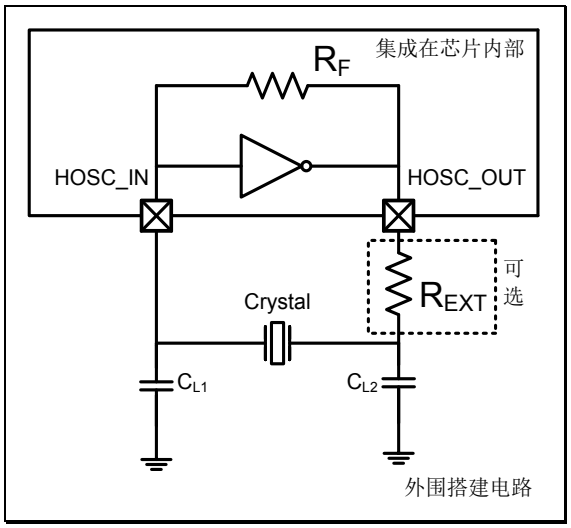


图 9-2 HOSC 电路图

9.4.2 内部高速RC振荡器时钟（HRC）

内部高速 RC 振荡器 HRC，可输出 2MHz（低功耗模式）或 24MHz 时钟（高速模式）。HRC 不受电源电压 VDD 的变化影响。

9.4.3 外部低速振荡器时钟（LOSC）

外部低速振荡器 LOSC，可驱动 32768Hz 的外部晶体振荡器。LOSC 内部自带匹配电容和反馈电阻，不需要外接电容和电阻。

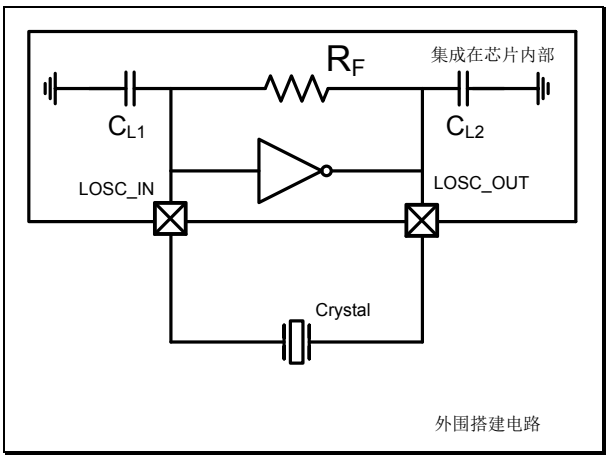


图 9-3 LOSC 电路图

9.4.4 内部低速RC振荡器时钟（LRC）

内部低速 RC 振荡器 LRC 可输 32768Hz 时钟。LRC 不受电源电压 VDD 的变化影响。

9.4.5 内部超低速RC振荡器时钟（ULRC）

内部超低速 RC 振荡器 ULRC 可输出约 10KHz 时钟。ULRC 不受电源电压 VDD 的变化影响。

9.4.6 内部倍频时钟（PLL）

4MHz 时钟源通过内部倍频器 PLL 可将时钟倍频至 36MHz 或 48MHz 或 72MHz 或 96MHz，PLL 时钟源有：

- ◇ HRC 分频至 4MHz
- ◇ HOSC 分频至 4MHz

9.4.7 系统时钟选择

通过 CMU 控制状态寄存器（CMU_CSR）的 SYS_CMD 位进行配置选择和切换时钟。

系统时钟源有：

- ◇ HRC
- ◇ LRC
- ◇ 带安全管理的 HOSC，可配置 HOSC 停振时自动切换至 HRC
- ◇ 带安全管理的 LOSC，可配置 LOSC 停振时自动切换至 LRC
- ◇ 带安全管理的 PLL，可配置 PLL 失锁时自动切换至 HRC
- ◇ 系统时钟可通过 CMU_CFGR 寄存器进行分频

9.4.8 时钟安全管理

高速振荡器安全管理 HOSM

HOSM 时钟安全系统可以通过软件使能（CMU_HOSMCR.EN = '1'）。HOSM 时钟安全监测机制在 HOSC 振荡器启动并稳定后被激活，当软件将 HOSC 振荡器关闭时，安全监测机制也将关闭。若 HOSC 时钟发生故障，系统时钟将切换至 HRC（内部高速振荡器时钟）。时钟故障事件可送到高级定时器的刹车输入端，且时钟失效中断标志位（CMU_HOSMCR.STPIF 位）被置位。如果中断使能（CMU_HOSMCR.STPIE=1），则产生时钟安全中断，允许软件完成营救操作。若 NMI 中断使能（CMU_HOSMCR.NMIE=1），则时钟失效事件将产生 NMI 中断（不可屏蔽中断）。

PLL 失锁安全管理 PULM

PULM 失锁安全管理通过将 CMU_PULMCR 的 EN 位设为 1 使能，一旦使能，PULM 失锁安全监测机制将在 PLL 启动后被激活，若软件将 PLL 关闭，则 PULM 安全监测机制关闭。如果 PLL 发生失锁，硬件对时钟的处理方式可通过 CMU_PULMCR 的 MODE 设定进行选择。PLL 发生失锁后，故障事件可送到高级定时器的刹车输入端，PLL 失锁中断标志位（CMU_PULMCR.ULKIF）被置起，如果失锁中断使能（CMU_PULMCR.ULKIE=1），则产生时钟安全中断，允许软件完成营救操作。若 NMI 中断使能（CMU_PULMCR.NMIE=1），则 PLL 失锁事件将产生 NMI 中断（不可屏蔽中断）。

低速振荡器安全管理 LOSM

LOSM 时钟安全系统可以通过软件使能(将 CMU_LOSMCR.EN 设为 1),一旦使能,LOSM 时钟安全监测机制将在 LOSC 振荡器启动并稳定后被激活,若软件将 LOSC 振荡器关闭,则安全监测机制关闭。如果 LOSC 时钟发生故障，输出时钟将切换至 LRC（内部低速振荡器时钟)，时钟失效中断标志位（CMU_LOSMCR.STPIF 位）被置起，如果中断使能（CMU_LOSMCR.STPIE=1)，则产生时钟安全中断，允许软件完成营救操作。若 NMI 中断使能(CMU_LOSMCR.NMIE=1),则时钟失效事件将产生 NMI 中断(不可屏蔽中断)。

9.5 特殊功能寄存器

9.5.1 寄存器列表

CMU 寄存器列表		
名称	偏移地址	描述
CMU_CSR	000_H	CMU 控制状态寄存器
CMU_CFGR	004_H	CMU 配置寄存器
Reserved	008_H~$00C_H$	保留
CMU_CLKENR	010_H	CMU 时钟使能寄存器
CMU_CLKSR	014_H	CMU 时钟状态寄存器
CMU_PLLCFG	018_H	PLL 配置寄存器
CMU_HOSCCFG	$01C_H$	HOSC 配置寄存器
CMU_HOSMCR	020_H	HOSC 安全管理控制寄存器
CMU_LOSMCR	024_H	LOSC 安全管理控制寄存器
CMU_PULMCR	028_H	PLL 失锁管理控制寄存器
Reserved	$02C_H$	保留
CMU_CLKOCR	030_H	CMU 时钟输出控制寄存器
CMU_BUZZCR	034_H	BUZZ 控制寄存器
Reserved	038_H~$03C_H$	保留
CMU_AHB1ENR	040_H	AHB1 外设时钟使能寄存器
Reserved	044_H~$04C_H$	保留
CMU_APB1ENR	050_H	APB1 外设时钟使能寄存器
CMU_APB2ENR	054_H	APB2 外设时钟使能寄存器
Reserved	058_H~$05C_H$	保留
CMU_LPENR	060_H	外设时钟低功耗模式使能寄存器
Reserved	064_H~$07C_H$	保留
CMU_PERICR	080_H	外设时钟控制寄存器
Reserved	084_H	保留
CMU_PERIDIVR	088_H	外设时钟分频控制寄存器

9. 5. 2 寄存器描述

9. 5. 2. 1 CMU控制状态寄存器（CMU_CSR）

CMU 控制状态寄存器（CMU_CSR）

偏移地址：000_H

复位值：$00000000\_00000000\_00000001\_00000000_B$

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

Reserved	Bit 31-26	—	**保留**
CFT_RDYN	Bit 25	R	**系统时钟滤波器切换状态位** 0：系统时钟滤波器切换完成或未发生切换动作 1：系统时钟滤波器正在切换 注：系统时钟滤波器在切换时仅需若干个系统时钟，软件读取时不保证能读到系统时钟滤波器正在切换的状态
CFT_STU	Bit 24	R	**系统时钟滤波器激活状态位** 0：系统时钟滤波器被选择 1：系统时钟滤波器被旁路
CFT_CMD	Bit 23-16	W	**系统时钟滤波切换命令位** 0x55：选择系统时钟滤波器 0xAA：旁路系统时钟滤波器 其他：无操作 注1：系统默认为选择系统时钟滤波器。 注2：当系统时钟滤波器正在切换时，该位写入无效。该位读出始终为0。
Reserved	Bit 15-13	—	**保留**
SYS_RDYN	Bit 12	R	**系统时钟切换状态位** 0：系统时钟切换完成或未发生切换动作 1：系统时钟正在切换 注：系统时钟在切换时仅需若干个系统时钟，软件读取时不保证能读到系统时钟正在切换的状态
Reserved	Bit 11	—	**保留**
SYS_STU	Bit 10-8	R	**当前系统时钟状态位** 000：保留 001：选择HRC 010：选择LRC 011：选择LOSC 100：选择PLL 101：选择HOSC 11x：保留

Reserved	Bit 7-3	—	**保留**
SYS_CMD	Bit 2-0	W	**系统时钟切换命令位** 000：无操作 001：选择HRC 010：选择LRC 011：选择LOSC 100：选择PLL 101：选择HOSC 11x：保留 注1：系统默认为选择HRC。 注2：当系统时钟正在切换时，该位写入无效。该位读出始终为0。

9. 5. 2. 2 CMU配置寄存器（CMU_CFGR）

CMU 配置寄存器（CMU_CFGR）

偏移地址：004_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					HRCFCS	HRCFST	HRCFSW	PCLK2DIV				PCLK1DIV				SYSDIV				Reserved				HCLK2DIV				HCLK1DIV			

Reserved	Bit 31-27	—	**保留**
HRCFCS	Bit 26	R/W	**HRC 频率配置选择位** 0：由配置字控制 1：由HRCFSW位控制
HRCFST	Bit 25	R	**HRC 频率选择状态位** 0：24MHz 1：2MHz 注：当HRC作为PLL的输入时钟源时，硬件固定为24MHz
HRCFSW	Bit 24	R/W	**HRC 频率软件选择位** 0：24MHz 1：2MHz
PCLK2DIV	Bit 23-20	R/W	**PCLK2 分频选择位** 0000：1分频 0001：2分频 0010：4分频 0011：8分频 …… 1100：4096分频 其他：保留
PCLK1DIV	Bit 19-16	R/W	**PCLK1 分频选择位** 0000：1分频 0001：2分频 0010：4分频 0011：8分频 …… 1100：4096分频 其他：保留
SYSDIV	Bit 15-12	R/W	**SYSCLK 分频选择位** 0000：1分频 0001：2分频 0010：4分频 0011：8分频

			…… 1100：4096分频 其他：保留
Reserved	Bit 11-8	—	**保留**
HCLK2DIV	Bit 7-4	R/W	**HCLK2 分频选择位** 0000：1分频 0001：2分频 0010：4分频 0011：8分频 …… 1100：4096分频 其他：保留
HCLK1DIV	Bit 3-0	R/W	**HCLK1 分频选择位** 0000：1分频 0001：2分频 0010：4分频 0011：8分频 …… 1100：4096分频 其他：保留

9. 5. 2. 3 CMU时钟使能寄存器（CMU_CLKENR）

CMU 时钟使能寄存器（CMU_CLKENR）

偏移地址：010_H

复位值：$00000000\_00000000\_00000000\_00010100_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														HOSC1MEN	HRC1MEN	Reserved							PLLEN	Reserved				LRCEN	HRCEN	LOSCEN	HOSCEN

Reserved	Bit 31-18	—	保留
HOSC1MEN	Bit 17	R/W	**HOSC 分频至 1MHz 使能位** 0：禁止 1：使能 注：需要根据实际外部的HOSC频率设置HOSC配置寄存器CMU_HOSCCFG
HRC1MEN	Bit 16	R/W	**HRC 分频至 1MHz 使能位** 0：禁止 1：使能
Reserved	Bit 15-9	—	保留
PLLEN	Bit 8	R/W	**PLL 软件使能位** 0：禁止 1：使能
Reserved	Bit 7-4	—	保留
LRCEN	Bit 3	R/W	**LRC 软件使能位** 0：禁止 1：使能
HRCEN	Bit 2	R/W	**HRC 软件使能位** 0：禁止 1：使能
LOSCEN	Bit 1	R/W	**LOSC 软件使能位** 0：禁止 1：使能
HOSCEN	Bit 0	R/W	**HOSC 软件使能位** 0：禁止 1：使能

9. 5. 2. 4 CMU时钟状态寄存器（CMU_CLKSR）

CMU 时钟状态寄存器（CMU_CLKSR）

偏移地址：014_H

复位值：$00000000\_0000xx00\_00000000\_00010100_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							PLLRDY	Reserved				LRCRDY	HRCRDY	LOSCRDY	HOSCRDY	Reserved							PLLACT	Reserved				LRCACT	HRCACT	LOSCACT	HOSCACT

Reserved	Bit 31-25	—	**保留**
PLLRDY	Bit 24	R	**PLL 稳定标志位** 0：未稳定或未激活 1：稳定
Reserved	Bit 23-20	—	**保留**
LRCRDY	Bit 19	R	**LRC 稳定标志位** 0：未稳定或未激活 1：稳定
HRCRDY	Bit 18	R	**HRC 稳定标志位** 0：未稳定或未激活 1：稳定
LOSCRDY	Bit 17	R	**LOSC 稳定标志位** 0：未稳定或未激活 1：稳定
HOSCRDY	Bit 16	R	**HOSC 稳定标志位** 0：未稳定或未激活 1：稳定
Reserved	Bit 15-9	—	**保留**
PLLACT	Bit 8	R	**PLL 激活状态位** 0：未激活 1：激活
Reserved	Bit 7-4	—	**保留**
LRCACT	Bit 3	R	**LRC 激活状态位** 0：未激活 1：激活
HRCACT	Bit 2	R	**HRC 激活状态位** 0：未激活 1：激活
LOSCACT	Bit 1	R	**LOSC 激活状态位** 0：未激活 1：激活
HOSCACT	Bit 0	R	**HOSC 激活状态位** 0：未激活

			1：激活

9. 5. 2. 5 PLL配置寄存器（CMU_PLLCFG）

PLL 配置寄存器（CMU_PLLCFG）

偏移地址：18_H

复位值：$00000000\_00000011\_00000000\_00000000_B$

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

Reserved	Bit 31-17	—	**保留**
PLLLCKN	Bit 16	R	**PLL 锁定标志位** 0：锁定 1：未锁定
Reserved	Bit 15-6	—	**保留**
PLLOS	Bit 5-4	R/W	**PLL 输出时钟选择位** 00：×9（36MHz） 01：×12（48MHz） 10：×18（72MHz） 11：×24（96MHz）
Reserved	Bit 3	—	**保留**
PLLRFS	Bit 2-0	R/W	**PLL 参考时钟选择位** 000：HRC/6 001：预留 010：HOSC 011：HOSC/2 100：HOSC/3 101：HOSC/4 110：HOSC/5 111：HOSC/6

9. 5. 2. 6 HOSC配置寄存器（CMU_HOSCCFG）

HOSC 配置寄存器（CMU_HOSCCFG）

偏移地址：$1C_H$

复位值：$00000000\_00000000\_00000000\_00010111_B$

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

Reserved	Bit 31-5	—	保留
FREQ	Bit 4-0	R/W	**HOSC 频率配置位** 0x00：1MHz 0x01：2MHz …… 0x16：23MHz 0x17：24MHz 其他：保留

9. 5. 2. 7 HOSC安全管理控制寄存器（CMU_HOSMCR）

HOSC 安全管理控制寄存器（CMU_HOSMCR）

偏移地址：020_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											NMIE	STPIF	STRIF	STPIE	STRIE	Reserved					FRQS			Reserved						CLKS	EN

Reserved	Bit 31-21	—	保留
NMIE	Bit 20	R/W	**HOSC安全事件不可屏蔽中断使能位** 0：禁止 1：使能
STPIF	Bit 19	R/C_W1	**HOSC停振标志位** 0：未发生 HOSC 停振或标志位已被清除 1：发生HOSC停振 注：该位写1清除，写0无效
STRIF	Bit 18	R/C_W1	**HOSC起振标志位** 0：未发生 HOSC 起振或标志位已被清除 1：发生HOSC起振 注：该位写1清除，写0无效
STPIE	Bit 17	R/W	**HOSC停振中断使能位** 0：禁止 1：使能
STRIE	Bit 16	R/W	**HOSC起振中断使能位** 0：禁止 1：使能
Reserved	Bit 15-11	—	保留
FRQS	Bit 10-8	R/W	**HOSC 频率配置位** 000：1~2MHz 001：2~4MHz 010：4~8MHz 011：8~16MHz 1xx：16~24MHz 注：当频率处在两个相邻档位的临界值时，这两个档位可任意选择
Reserved	Bit 7-2	—	保留
CLKS	Bit 1	R	**当前选择时钟状态位** 0：选择 HOSC 1：选择HRC
EN	Bit 0	R/W	**HOSC 安全管理使能位** 0：禁止 1：使能

9. 5. 2. 8 LOSC安全管理控制寄存器（CMU_LOSMCR）

LOSC 安全管理控制寄存器（CMU_LOSMCR）

偏移地址：024_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-21	—	保留
NMIE	Bit 20	R/W	**LOSC安全事件不可屏蔽中断使能位** 0：禁止 1：使能
STPIF	Bit 19	R/C_W1	**LOSC停振标志位** 0：未发生 LOSC 停振或标志位已被清除 1：发生LOSC停振 注：该位写1清除，写0无效
STRIF	Bit 18	R/C_W1	**LOSC起振标志位** 0：未发生 LOSC 起振或标志位已被清除 1：发生LOSC起振 注：该位写1清除，写0无效
STPIE	Bit 17	R/W	**LOSC停振中断使能位** 0：禁止 1：使能
STRIE	Bit 16	R/W	**LOSC起振中断使能位** 0：禁止 1：使能
Reserved	Bit 15-2	—	保留
CLKS	Bit 1	R	**当前选择时钟状态位** 0：选择 LOSC 1：选择LRC
EN	Bit 0	R	**LOSC 安全管理使能位** 0：禁止 1：使能

9. 5. 2. 9 PLL失锁管理控制寄存器（CMU_PULMCR）

PLL 失锁管理控制寄存器（CMU_PULMCR）

偏移地址：028_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											NMIE	ULKIF	LCKIF	ULKIE	LCKIE	Reserved						MODE		Reserved						CLKS	EN

Reserved	Bit 31-21	—	**保留**
NMIE	Bit 20	R/W	**PLL安全事件不可屏蔽中断使能位** 0：禁止 1：使能
ULKIF	Bit 19	R/C_W1	**PLL失锁标志位** 0：未发生 PLL 失锁或标志位已被清除 1：发生PLL失锁 注：该位写1清除，写0无效
LCKIF	Bit 18	R/C_W1	**PLL锁定标志位** 0：未发生 PLL 锁定或标志位已被清除 1：发生PLL锁定 注：该位写1清除，写0无效
ULKIE	Bit 17	R/W	**PLL失锁定中断使能位** 0：禁止 1：使能
LCKIE	Bit 16	R/W	**PLL锁定中断使能位** 0：禁止 1：使能
Reserved	Bit 15-10	—	**保留**
MODE	Bit 9-8	R/W	**PLL失锁管理模式选择位** 00：失锁后无操作 01：失锁后切换至HRC 1x：失锁后切换至HRC，待重新后锁定切回
Reserved	Bit 7-2	—	**保留**
CLKS	Bit 1	R	**当前选择时钟状态位** 0：选择 PLL 1：选择HRC
EN	Bit 0	R/W	**PLL 失锁管理使能位** 0：禁止 1：使能

9. 5. 2. 10 CMU时钟输出控制寄存器（CMU_CLKOCR）

CMU 时钟输出控制寄存器（CMU_CLKOCR）

偏移地址：030_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
LSCOS	Bit 26-24	R/W	**低速时钟输出源选择位** 000：LOSC 001：LRC 010：LOSM 011：BUZZ 100：ULRC 其余：保留
Reserved	Bit 23-17	—	**保留**
LSCOEN	Bit 16	R/W	**低速时钟输出使能位** 0：禁止 1：使能
Reserved	Bit 15	—	**保留**
HSCODIV	Bit 14-12	R/W	**高速时钟输出分频选择位** 000：1 分频 001：2 分频 010：4 分频 011：8 分频 100：16 分频 101：32 分频 110：64 分频 111：128 分频
Reserved	Bit 11	—	**保留**
HSCOS	Bit 10-8	R/W	**高速时钟输出源选择位** 000：HOSC 001：LOSC 010：HRC 011：LRC 100：HOSM 101：PLL 110：预留 111：SYSCLK
Reserved	Bit 7-1	—	**保留**

HSCOEN	Bit 0	R/W	**高速时钟输出使能位** 0：禁止 1：使能

9. 5. 2. 11 BUZZ控制寄存器（CMU_BUZZCR）

BUZZ 控制寄存器（CMU_BUZZCR）

偏移地址：034_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

DAT	Bit 31-16	R/W	**BUZZ频率数据值** $Freq_{BUZZ} = \frac{Freq_{SYSCLK}}{2^{DIV+1} \times (DAT+1)}$
Reserved	Bit 15-11	—	**保留**
DIV	Bit 10-8	R/W	**BUZZ 时钟分频选择位** 000：2 分频 001：4 分频 010：8 分频 011：16 分频 100：32 分频 101：64 分频 110：128 分频 111：256 分频
Reserved	Bit 7-1	—	**保留**
EN	Bit 0	R/W	**BUZZ 使能位** 0：禁止 1：使能

9. 5. 2. 12 AHB1 外设时钟使能寄存器（CMU_AHB1ENR）

AHB1 外设时钟使能寄存器（CMU_AHB1ENR）

偏移地址：040_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					USBEN	Reserved	DMAEN	QSPIEN	EBIEN	PISEN	TRNGEN	CRYPTEN	CALCEN	CRCEN	GPIOEN

Reserved	Bit 31-11	—	保留
USBEN	Bit 10	R/W	**USB 时钟使能位（初值为 0）** 0：禁止 1：使能
Reserved	Bit 9	—	保留
DMAEN	Bit 8	R/W	**DMA 时钟使能位** 0：禁止 1：使能
QSPIEN	Bit 7	R/W	**QSPI 的 AHB 时钟使能位** 0：禁止 1：使能
EBIEN	Bit 6	R/W	**EBI 时钟使能位** 0：禁止 1：使能
PISEN	Bit 5	R/W	**PIS 时钟使能位** 0：禁止 1：使能
TRNGEN	Bit 4	R/W	**TRNG 时钟使能位** 0：禁止 1：使能
CRYPTEN	Bit 3	R/W	**CRYPT 时钟使能位** 0：禁止 1：使能
CALCEN	Bit 2	R/W	**CALC 时钟使能位** 0：禁止 1：使能
CRCEN	Bit 1	R/W	**CRC 时钟使能位** 0：禁止 1：使能
GPIOEN	Bit 0	R/W	**GPIO 时钟使能位** 0：禁止 1：使能

9. 5. 2. 13 APB1 外设时钟使能寄存器（CMU_APB1ENR）

APB1 外设时钟使能寄存器（CMU_APB1ENR）

偏移地址：050_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						QSPIEN	CAN0EN	Reserved		I2C1EN	I2C0EN	Reserved	SPI2EN	SPI1EN	SPI0EN	Reserved		UART5EN	UART4EN	UART3EN	UART2EN	UART1EN	UART0EN	GP16C4T1EN	GP16C4T0EN	BS16T1EN	BS16T0EN	GP32C4T1EN	GP32C4T0EN	AD16C4T1EN	AD16C4T0EN

Reserved	Bit 31-26	—	**保留**
QSPIEN	Bit 25	R/W	**QSPI 的 APB 时钟使能位** 0：禁止 1：使能
CAN0EN	Bit 24	R/W	**CAN0 时钟使能位** 0：禁止 1：使能
Reserved	Bit 23-22	—	**保留**
I2C1EN	Bit 21	R/W	**I2C1 时钟使能位** 0：禁止 1：使能
I2C0EN	Bit 20	R/W	**I2C0 时钟使能位** 0：禁止 1：使能
Reserved	Bit 19	—	**保留**
SPI2EN	Bit 18	R/W	**SPI2 时钟使能位** 0：禁止 1：使能
SPI1EN	Bit 17	R/W	**SPI1 时钟使能位** 0：禁止 1：使能
SPI0EN	Bit 16	R/W	**SPI0 时钟使能位** 0：禁止 1：使能
Reserved	Bit 15-14	—	**保留**
UART5EN	Bit 13	R/W	**UART5 时钟使能位** 0：禁止 1：使能
UART4EN	Bit 12	R/W	**UART4 时钟使能位** 0：禁止 1：使能
UART3EN	Bit 11	R/W	**UART3 时钟使能位**

			0：禁止 1：使能
UART2EN	Bit 10	R/W	**UART2 时钟使能位** 0：禁止 1：使能
UART1EN	Bit 9	R/W	**UART1 时钟使能位** 0：禁止 1：使能
UART0EN	Bit 8	R/W	**UART0 时钟使能位** 0：禁止 1：使能
GP16C4T1EN	Bit 7	R/W	**GP16C4T1 时钟使能位** 0：禁止 1：使能
GP16C4T0EN	Bit 6	R/W	**GP16C4T0 时钟使能位** 0：禁止 1：使能
BS16T1EN	Bit 5	R/W	**BS16T1 时钟使能位** 0：禁止 1：使能
BS16T0EN	Bit 4	R/W	**BS16T0 时钟使能位** 0：禁止 1：使能
GP32C4T1EN	Bit 3	R/W	**GP32C4T1 时钟使能位** 0：禁止 1：使能
GP32C4T0EN	Bit 2	R/W	**GP32C4T0 时钟使能位** 0：禁止 1：使能
AD16C4T1EN	Bit 1	R/W	**AD16C4T1 时钟使能位** 0：禁止 1：使能
AD16C4T0EN	Bit 0	R/W	**AD16C4T0 时钟使能位** 0：禁止 1：使能

9. 5. 2. 14 APB2 外设时钟使能寄存器（CMU_APB2ENR）

APB2 外设时钟使能寄存器（CMU_APB2ENR）

偏移地址：054_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												DBGCEN	Reserved	BKPCEN	TSENSEEN	RTCEN	IWDTEN	Reserved	WWDTEN	ACMP2EN	Reserved	DACEN	Reserved	ACMP1EN	ACMP0EN	ADC1EN	ADC0EN	Reserved			

Reserved	Bit 31-20	—	**保留**
DBGCEN	Bit 19	R/W	**DBGC 时钟使能位** 0：禁止 1：使能
Reserved	Bit 18	—	**保留**
BKPCEN	Bit 17	R/W	**BKPC 时钟使能位** 0：禁止 1：使能
TSENSEEN	Bit 16	R/W	**TSENSE 时钟使能位** 0：禁止 1：使能
RTCEN	Bit 15	R/W	**RTC 时钟使能位** 0：禁止 1：使能
IWDTEN	Bit 14	R/W	**IWDT 时钟使能位** 0：禁止 1：使能
Reserved	Bit 13	R/W	**保留**
WWDTEN	Bit 12	R/W	**WWDT 时钟使能位** 0：禁止 1：使能
ACMP2EN	Bit 11	R/W	**ACMP2 时钟使能位** 0：禁止 1：使能
Reserved	Bit 10	—	**保留**
DACEN	Bit 9	R/W	**DAC 时钟使能位** 0：禁止 1：使能
Reserved	Bit 8	R/W	**保留**
ACMP1EN	Bit 7	R/W	**ACMP1 时钟使能位** 0：禁止 1：使能
ACMP0EN	Bit 6	R/W	**ACMP0 时钟使能位**

			0：禁止 1：使能
ADC1EN	Bit 5	R/W	**ADC1 时钟使能位** 0：禁止 1：使能
ADC0EN	Bit 4	R/W	**ADC0 时钟使能位** 0：禁止 1：使能
Reserved	Bit 3-0	—	**保留**

9. 5. 2. 15 外设时钟低功耗模式使能寄存器（CMU_LPENR）

外设时钟低功耗模式使能寄存器（CMU_LPENR）

偏移地址：060_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													STOP1CS			Reserved												HOSCEN	HRCEN	LOSCEN	LRCEN

Reserved	Bit 31-19	—	保留
STOP1CS	Bit 18-16	R/W	**STOP1 模式时钟选择位** 000：LRC 001：HRC（24MHz） 010：HRC分频至2MHz 011：HRC分频至1MHz 100：HOSC 101：HOSC分频至1MHz 110：LOSM 其他：保留
Reserved	Bit 15-4	—	保留
HOSCEN	Bit 3	R/W	**HOSC 时钟低功耗模式使能位** 0：禁止 1：使能
HRCEN	Bit 2	R/W	**HRC 时钟低功耗模式使能位** 0：禁止 1：使能
LOSCEN	Bit 1	R/W	**LOSC 时钟低功耗模式使能位** 0：禁止 1：使能
LRCEN	Bit 0	R/W	**LRC 时钟低功耗模式使能位** 0：禁止 1：使能

9. 5. 2. 16 外设时钟控制寄存器（CMU_PERICR）

外设时钟控制寄存器（CMU_PERICR）

偏移地址：080_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31	—	**保留**
USBPHYCS	Bit 30-28	R/W	**USB PHY 通信时钟源选择位** 000：HOSC 001：HRC 010：PCLK1 011：PLL 100：HOSM 其他：保留
Reserved	Bit 27	—	**保留**
QSPICS	Bit 26-24	R/W	**QSPI 通信时钟选择位** 000：PCLK1 001：HCLK2 010：HRC 011：HOSC 100：PLL 101：HOSM 其他：保留
Reserved	Bit 23-0	—	**保留**

9. 5. 2. 17 外设时钟分频控制寄存器（CMU_PERIDIVR）

外设时钟分频控制寄存器（CMU_PERIDIVR）																															
偏移地址：088_H																															
复位值：$00010000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBPHYDIV				Reserved																											

USBPHYDIV	Bit 31-28	W	**USB PHY 时钟分频选择位** 0000：1分频 0001：2分频（默认） 0010：4分频 0011：8分频 …… 1100：4096分频 其他：保留
Reserved	Bit 27-0	—	**保留**

第10章 备份电源域控制（BKPC）

10. 1 概述

本模块控制备份电源域（如 LRC、LOSC、LOSM、RTC、TSENSE 等）的工作状态， standby 模式下唤醒源的选择等。通过对寄存器的配置，可以达到在功耗和可靠性上对备份域的工作状态进行灵活控制与选择。

10. 2 特性

- 受保护的寄存器访问，防止误操作
- Standby 模式下唤醒源可选择
- RTC 与温感模块的工作时钟可选择

10. 3 特殊功能寄存器

10. 3. 1 寄存器列表

BKPC 寄存器列表		
名称	偏移地址	描述
BKPC_PROT	000_H	备份域保护寄存器
BKPC_CR	004_H	备份域控制寄存器
BKPC_PCCR	008_H	备份域外设时钟控制寄存器
—	$00C_H$	保留
BKPC_PDCR	010_H	备份域掉电控制寄存器

10. 3. 2 寄存器描述

10. 3. 2. 1 备份域保护寄存器（BKPC_PROT）

备份域保护寄存器（BKPC_PROT）																															
偏移地址：000_H																															
上电复位值：$00000000\_00000000\_00000000\_00000001_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY																															PROT

KEY	Bit 31-1	W	**保护关键码** 0x9669AA55：去除写保护 其他：开启写保护
PROT	Bit 0	R	**保护状态位** 0：无写保护 1：写保护

10. 3. 2. 2 备份域控制寄存器（BKPC_CR）

备份域控制寄存器（BKPC_CR）																															
偏移地址：004_H																															
上电复位值：$00000000\_00000100\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			WKPOL	WKPS			WKPEN	MRST_WKPEN	Reserved				LRCEN	LOSMEN	LOSCEN

Reserved	Bit 31-13	—	保留
WKPOL	Bit 12	RW	**STANDBY 唤醒端口极性选择位** 0：高电平唤醒 1：低电平唤醒
WKPS	Bit 11-9	R/W	**STANDBY 唤醒端口选择位** 000：PA0 001：PA1 010：PA2 011：PA3 100：PA4 101：PA5 110：PA6 111：PA7
WKPEN	Bit 8	R/W	**STANDBY 唤醒端口使能位** 0：禁止 1：使能
MRST_WKPEN	Bit 7	R/W	**MRST 端口唤醒 STANDBY 使能位** 0：禁止 1：使能
Reserved	Bit 6-3	—	保留
LRCEN	Bit 2	RW	**LRC 使能位** 0：禁止 1：使能
LOSMEN	Bit 1	RW	**LOSC 安全管理使能位** 0：禁止 1：使能
LOSCEN	Bit 0	RW	**LOSC 使能位** 0：禁止 1：使能

10. 3. 2. 3 备份域外设时钟控制寄存器（BKPC_PCCR）

备份域外设时钟控制寄存器（BKPC_PCCR）

偏移地址：008_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-6	—	保留
TSENSECS	Bit 5-4	R/W	**TSENSE 时钟选择位** 00：LOSM 01：LRC 10：HRC分频至1MHz 11：HOSC分频至1MHz
Reserved	Bit 3-2	—	保留
RTCCS	Bit 1-0	R/W	**RTC 时钟选择位** 00：LOSM 01：LRC 10：HRC分频至1MHz 11：HOSC分频至1MHz

10. 3. 2. 4 备份域掉电控制寄存器（BKPC_PDCR）

备份域掉电控制寄存器（BKPC_PDCR）

偏移地址：010_H

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-2	—	保留
ISOLATE	Bit 1	R	**备份域 ISO 状态** 0：ISO 无效 1：ISO 有效
Reserved	Bit 0	—	保留

第11章 DMA控制器（DMA）

11. 1 概述

DMA 控制器可独立于 CPU 对内部存储进行操作，利用 DMA 可很好的减轻 CPU 的负担并且节省功耗。每个 DMA 通道可选择芯片上的任意外设资源，实现数据的搬运、传输及控制。

11. 2 特性

该模块的控制逻辑提供以下特性：

- 支持 12 个独立的 DMA 通道
- 仲裁到达的请求
- 指示有效通道
- 指示通道完成
- 指示 AHB-Lite 接口上发生错误
- 使能低速外设，用来拖延 DMA 周期
- 完成一个 DMA 周期前，等待清零请求
- 进行多个或单个 DMA 传输
- 进行以下不同类型的 DMA 传输
  - 存储器到存储器
  - 存储器到外设
  - 外设到存储器

注：因每个通道只提供单个 DMA 请求接口，因此不支持外设到外设传输。

11. 3 结构框图

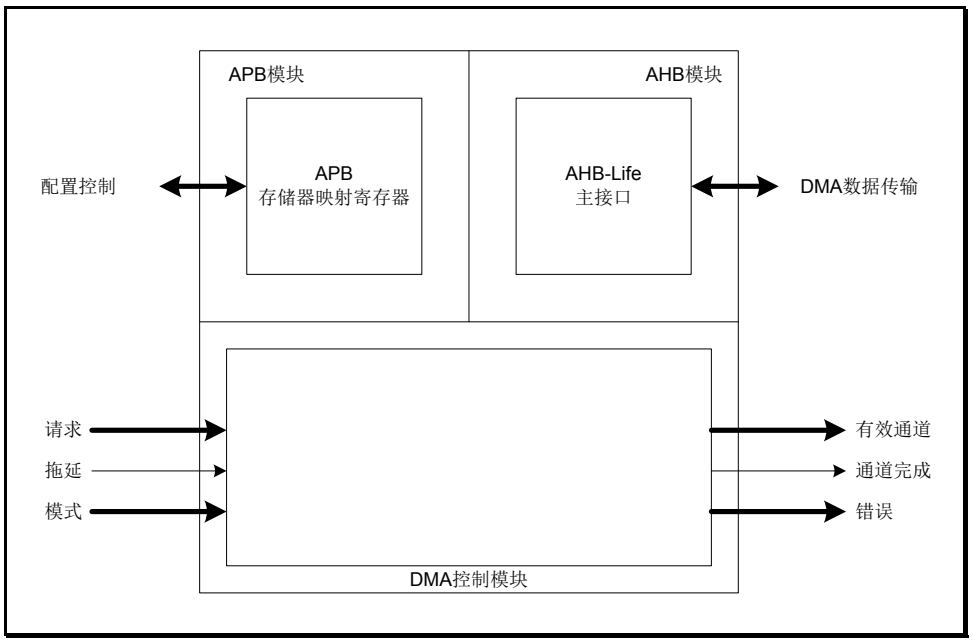


图 11-1 DMA 结构框图

11.4 功能描述

11.4.1 通道选择配置

通道选择模块可以选择要连接到每个 DMA 通道的外设请求线路。

该配置是由软件通过控制寄存器 DMA_CH0_SELCON-DMA_CH11_SELCON 完成的。这些控制寄存器都是由 MSEL 和 MSIGSEL 位组成的。MSEL 位选择要监听的外设，MSIGSEL 位从所选外设中选择要使用的输出信号。

注：当所选外设的时钟关闭时，DMA 通道不应该被使能。

11.4.2 DMA控制

11.4.2.1 DMA仲裁率

在 DMA 传输过程中，用户可配置控制器何时进行仲裁。该动作可缩短响应高优先通道的延时。

控制器提供 4 个比特位，R_power，用来配置在重新进行仲裁前，AHB 总线传输的次数。其中 R 为 2 的次方，2^R，决定了仲裁率。当 R = 4，则仲裁率为 2^4 = 16，即每进行 16 次 DMA 传输就进行一次仲裁。

R_power	x 次 DMA 传输后进行仲裁
b0000	x = 1
b0001	x = 2
b0010	x = 4
b0011	x = 8
b0100	x = 16
b0101	x = 32
b0110	x = 64
b0111	x = 128
b1000	x = 256
b1001	x = 512
b1010-b1111	x = 1024

表 11-1 仲裁设置

注：不要给优先级低的通道赋一个较大的 R_power 值，这样会导致在重新进行仲裁前，控制器不会响应优先级高的请求。

需要完成的 DMA 传输次数 N 是由用户指定的。当 N > 2^R 且 N 不是 2^R 的整数倍，控制器会依次先完成 2^R 次传输直到剩余的次数 N 小于 2^R。剩余的次数 N 会在 DMA 周期的最后完成。用户可将 R_power 的值保存在通道控制数据结构中。

11. 4. 2. 2 优先级

当控制器进行仲裁的时候，下一个进行仲裁的通道可由以下信息决定：

- 通道数
- 通道优先级

每个通道都可以通过 CHNL_PRIORITY_SET 寄存器设置为默认优先级或者高优先级。

通道 0 的优先级最高。随着通道数增高，优先级随之降低。

通道数	优先级设置	优先级（由高到低）
0	高	最高优先级
1	高	-
2	高	-
-	高	-
-	高	-
-	高	-
30	高	-
31	高	-
0	默认	-
1	默认	-
2	默认	-
-	默认	-
-	默认	-
-	默认	-
30	默认	-
31	默认	最低优先级

表 11-2 DMA 通道优先级

当完成一个 DMA 传输，控制器将轮询所有可用的 DMA 通道。下方为轮询流程图。

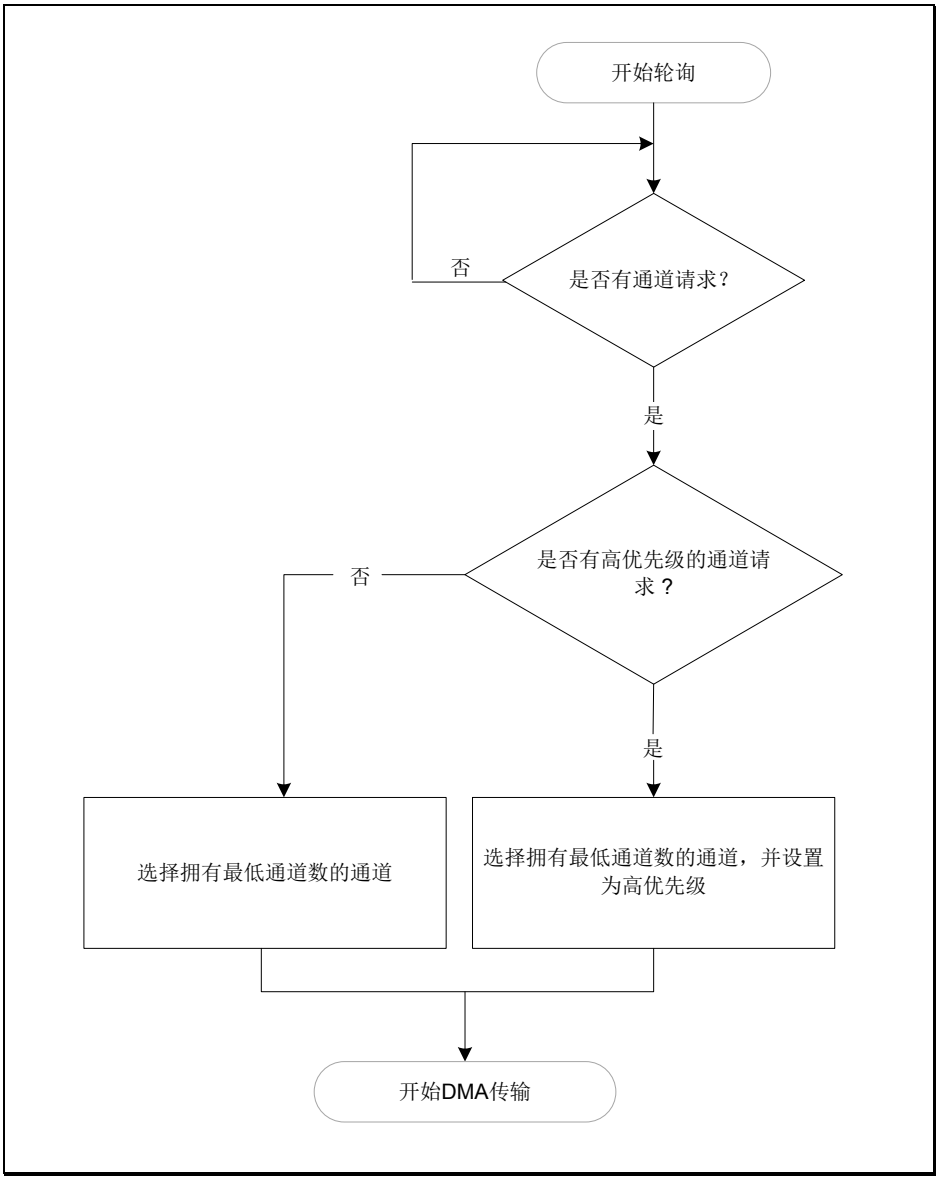


图 11-2 轮询流程图

11. 4. 2. 3 DMA周期类型

控制器可通过 cycle_ctrl 来选择 DMA 周期的类型，请参考下表。

cycle_ctrl	功能描述
b000	通道控制数据结构无效
b001	基础 DMA 传输
b010	自动请求
b011	乒乓
b100	存储器分散-聚集（主要数据结构）
b101	存储器分散-聚集（交替数据结构）
b110	外设分散-聚集（主要数据结构）
b111	外设分散-聚集（交替数据结构）

表 11-3 DMA 周期类型

注：cycle_ctrl 位于 channel_cfg 存储地址。

在所有的周期模式下，控制器在完成 2^R 次传输后进行仲裁。若一个低优先级通道被赋予了一个较大的 2^R 值，则到该通道完成 DMA 传输之前，所有其它的通道都不会进行 DMA 传输。因此，用户需谨慎设置 R_power 的值，避免增加高优先级通道的延时。

无效模式

当完成一个 DMA 传输后，控制器会将周期类型设置为无效，避免控制器发送同一个 DMA 周期。

基础模式

在此模式下，用户可以设置控制器使用主要数据结构或者交替数据结构。当通道使能后且控制器接收到该通道上的请求信号，接下来的流程如下

1. 控制器展开 2^R 次传输。如果剩余的传输次数为 0，则继续步骤 3。
2. 控制器仲裁如下：
   - 若高优先级通道有请求信号，则控制器先响应该通道
   - 若外设或者软件向控制器发出请求信号，则继续进行步骤 1
3. 控制器将 dma_done[C]置高一个 hclk 周期，向主控处理器表明该 DMA 周期已完成。

自动请求

在自动请求模式下，仅需要收到一次请求信号就可以完成整个 DMA 周期。无须大幅度增加回应高优先级请求的延时，也无须多次向处理器或者外设发出请求，就可以完成大数据的传输。

用户可以配置控制器使用主要数据结构或者交替数据结构。当通道使能后且控制器接收到该通道上的请求，则流程如下：

1. 控制器展开 2^R 次传输。如果剩余的传输次数为 0，则继续步骤 3。
2. 控制器进行仲裁。当通道 C 的优先级最高，则继续进行步骤 1。
3. 控制器将 dma_done[C]置高一个 hclk 周期，向主控处理器表明该 DMA 周期已完成。

乒乓

在乒乓模式下，控制器先使用其中一种数据结构完成一个 DMA 周期，接着再使用另外一种数据结构完成一次 DMA 周期。控制器将会继续转换这两种数据结构直到读到无效的数据结构或者通道被主控处理器禁止。

下图显示为在乒乓模式下的 DMA 传输。

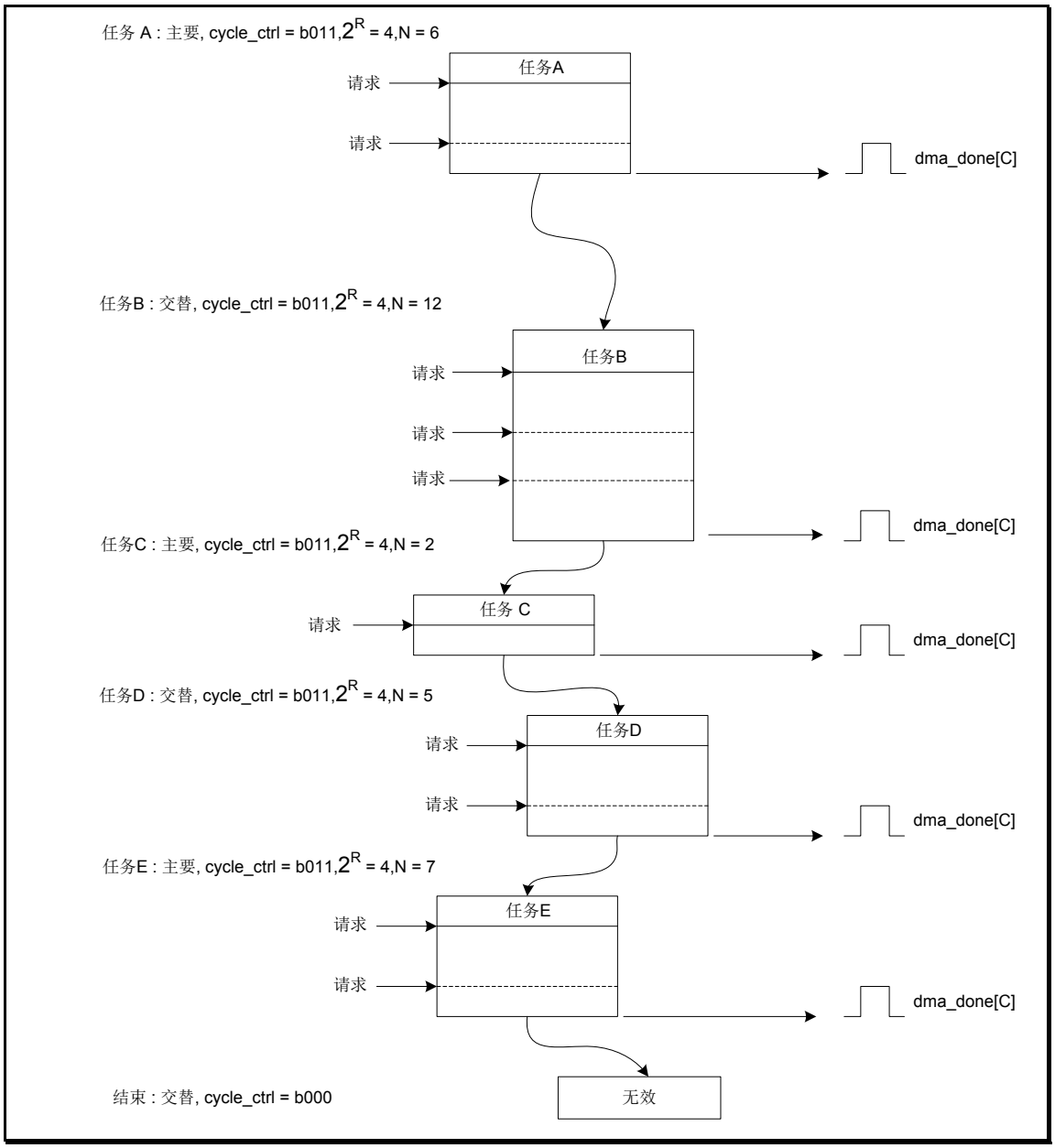


图 11-3 乒乓示例

任务 A

1. 主控处理器配置任务 A 为主要数据结构。
2. 主控处理器配置任务 B 为交替数据结构。当任务 A 完成时，控制器会立刻转换去任务 B，前提是没有高优先级通道提出请求。
3. 控制器接收到一个请求并完成 4 次 DMA 传输。
4. 控制器进行仲裁。当控制器接收到该通道上的请求，如果该通道具有最高优先级，则继续流程。
5. 控制器进行剩余的 2 次 DMA 传输。
6. 控制器将 dma_done[C]置高一个 hclk 周期，并进入仲裁流程。

当任务 A 完成后，主控处理器可配置任务 C 为主要数据结构。在任务 B 完成之后，控制器可立即转换去任务 C，前提是没有高优先级通道提出请求。

当控制器接收到该通道上新的请求且拥有最高优先级，则任务 B 开始执行：

任务 B

7. 控制器进行 4 次 DMA 传输。
8. 控制器进行仲裁。当控制器接收到该通道上的请求，如果该通道具有最高优先级，则继续流程。
9. 控制器进行 4 次 DMA 传输。
10. 控制器进行仲裁。当控制器接收到该通道上的请求，如果该通道具有最高优先级，则继续流程。
11. 控制器进行剩余的 4 次 DMA 传输。
12. 控制器将 dma_done[C]置高一个 hclk 周期，并进入仲裁流程。

当任务 B 完成后，主控处理器可将任务 D 配置为交替数据结构。

当控制器接收到该通道上新的请求且拥有最高优先级，则任务 C 开始执行：

任务 C

13. 控制器进行 2 次 DMA 传输。
14. 控制器将 dma_done[C]置高一个 hclk 周期，并进入仲裁流程。

当任务 C 完成后，主控处理器可将任务 E 配置为主要数据结构。

当控制器接收到该通道上新的请求且拥有最高优先级，则任务 D 开始执行：

任务 D

15. 控制器进行 4 次 DMA 传输。
16. 控制器进行仲裁。当控制器接收到该通道上的请求，如果该通道具有最高优先级，则继续流程。
17. 控制器进行剩余的 DMA 传输。

18. 控制器将 dma_done[C]置高一个 hclk 周期，并进入仲裁流程。

当控制器接收到该通道上新的请求且拥有最高优先级，则任务 E 开始执行：

任务 E

19. 控制器进行 4 次 DMA 传输。
20. 控制器进行仲裁。当控制器接收到该通道上的请求，如果该通道具有最高优先级，则继续流程。
21. 控制器进行剩余的 3 次 DMA 传输。
22. 控制器将 dma_done[C]置高一个 hclk 周期，并进入仲裁流程。

若控制器接收到该通道上的新请求且拥有最高优先级，则控制器会企图开始下一个任务。但由于主控处理器尚未配置交替数据结构，且当任务 D 完成的时候 cycle_ctrl 被设置为 b000，则会终止乒乓 DMA 传输。

注：用户可通过设置 cycle_ctrl 为 b001 将任务 E 配置为基础 DMA 周期，以用来终止乒乓 DMA 周期。

存储器分散-聚集

在存储器分散-聚集模式下，控制器先接收到一个初始请求，接着采用主要数据结构进行 4 次 DMA 传输。当传输完成时，会以交替数据结构开始一个新的 DMA 周期。当该周期完成时，控制器将会采用主要数据结构开始又一轮 4 次 DMA 传输。若发生以下任意一种情况，则控制器将停止主要数据结构和交替数据结构的转换：

- 主控处理器将交替数据结构配置为一个基础周期
- 读取到一个无效数据结构

注：当完成 N 次主要数据结构的传输后，可设置 cycle_ctrl 为 b000 使其变为无效的数据结构。

当完成一个自动请求周期的分散-聚集模式传输，控制器仅将 dma_done[C]置为有效。

在分散-聚集模式下，控制器利用主要数据结构来编程交替数据结构。下表列出了主要数据结构的 channel_cfg 配置，分为固定值配置和用户配置。

位	域	值	功能描述
固定值			
[31：30]	dst_inc	b10	配置控制器使用字作为地址增量
[29：28]	dst_size	b10	配置控制器使用字传输
[27：26]	src_inc	b10	配置控制器使用字作为地址增量
[25：24]	src_size	b10	配置控制器使用字传输
[17：14]	R_power	b1101	配置控制器进行 4 次 DMA 传输
[3]	next_useburst	0	当配置为存储器分散-聚集模式时，该位必须设置为 0
[2：0]	cycle_ctrl	b100	配置控制器执行存储器分散-聚集 DMA 周期
用户定义			
[23：21]	dst_prot_ctrl	-	当写入目标数据后，配置 HPROT 的状态
[20：18]	src_prot_ctrl	-	当读出源数据后，配置 HPROT 的状态
[13：4]	n_minus_1	N[a]	配置控制器进行 N 次 DMA 传输，其中 N 为 4 的倍数

表 11-4 存储器分散-聚集数据结构

注：由于 R_power 配置为 4，因此 N 必须设置为 4 的倍数。N/4 的值为配置交替数据结构的次数。

下图为存储器分散-聚集模式的示例：

初始化：

1. 配置主要数据结构来使能复制 A，B，C 和 D 的操作：cycle_ctrl = b100，2^R = 4，N = 16
2. 利用下表中的结构，将主要源数据写入存储器。

	src_data_end_ptr	dst_data_end_ptr	channel_cfg	Unused
任务 A 数据	0x0A000000	0x0AE00000	Cycle_ctrl = b101，2^R = 4，N = 3	0xXXXXXXXX
任务 B 数据	0x0B000000	0x0BE00000	Cycle_ctrl = b101，2^R = 2，N = 8	0xXXXXXXXX
任务 C 数据	0x0C000000	0x0CE00000	Cycle_ctrl = b101，2^R = 8，N = 5	0xXXXXXXXX
任务 D 数据	0x0D000000	0x0DE00000	Cycle_ctrl = b010，2^R = 4，N = 4	0xXXXXXXXX

表 11-5　各任务描述符配置示例

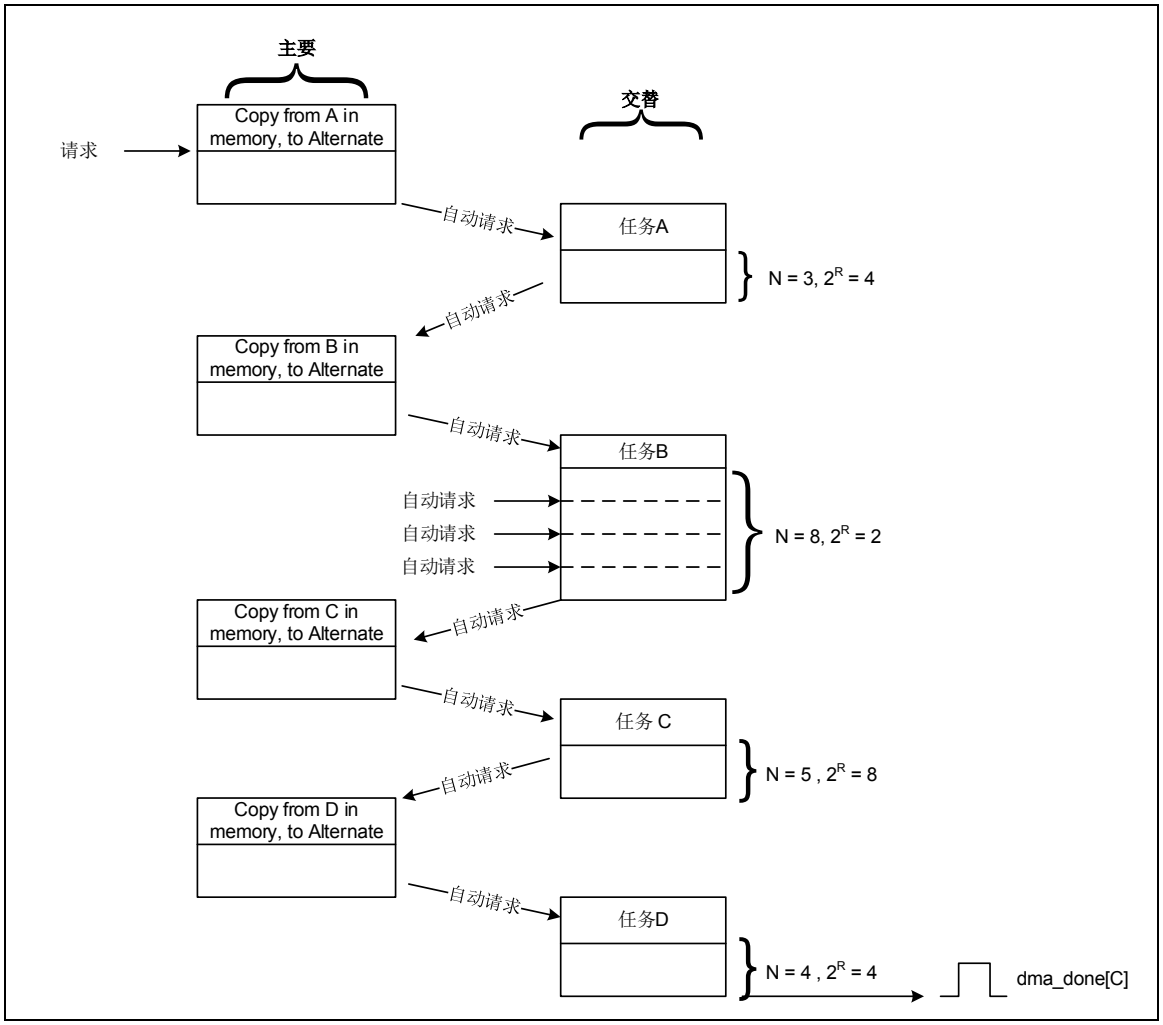


图 11-4　存储器分散-聚集示例

初始化

1. 主控处理器通过设置 cycle_ctrl 为 b100，使主要数据结构运行于存储器分散-聚集模式。由于单个通道的数据结构包含 4 个字，所以 2^R 必须设置为 4。在该示例中，有 4 个任务，因此 N 设为 16。

2. 主控处理器将任务 A，B，C 和 D 的数据结构写入由主要 src_data_end_ptr 指定的存储器地址中。

3. 主控处理器使能该通道。

当控制器接收到 dma_req[ ]请求或者来自主控处理器的手动请求，则存储器分散-聚集传输开始执行，流程如下：

主要，复制 A

1. 在接收到请求后，控制器进行 4 次 DMA 传输，并且任务 A 写为交替数据结构。

2. 控制器在该通道上生成一个自动请求接着进行仲裁。

任务 A

3. 控制器进行任务 A。当任务 A 完成，控制器在该通道上生成一个自动请求，接着进行仲裁。

主要，复制 B

4. 控制器进行 4 次 DMA 传输，且任务 B 写为交替数据结构。

5. 控制器在该通道上生成一个自动请求接着进行仲裁。

任务 B

6. 控制器进行任务 B。当任务 B 完成，控制器在该通道上生成一个自动请求，接着进行仲裁。

主要，复制 C

7. 控制器进行 4 次 DMA 传输，且任务 C 写为交替数据结构。

8. 控制器在该通道上生成一个自动请求接着进行仲裁。

任务 C

9. 控制器进行任务 C。当任务 C 完成，控制器在该通道上生成一个自动请求，接着进行仲裁。

主要，复制 D

10. 控制器进行 4 次 DMA 传输，且任务 D 写为交替数据结构。

11. 控制器设置主要数据结构的 cycle_ctrl 为 b000，表明该数据结构为无效。

12. 控制器在该通道上生成一个自动请求接着进行仲裁。

任务 D

13. 控制器使用自动请求周期执行任务 D。

14. 控制器将 dma_done[C]置高一个 hclk 周期，并进入仲裁流程。

外设分散-聚集

在外设分散-聚集模式下，控制器接收到一个来自外设的初始请求，接着使用主要数据结构执行 4 次 DMA 传输，然后立即使用交替数据结构启动一个新的 DMA 周期，无须重新仲裁。

注：仅在该状况下，当完成主要数据结构的传输后，控制器无须进入仲裁流程。

在该 DMA 周期完成后，控制器重新仲裁。如果控制器接收到外设请求且拥有最高优先级，则执行又一轮 4 次主要数据结构的 DMA 传输。接着立即启动一个交替数据结构的 DMA 周期，无须重新仲裁。若发生以下任意一种情况，则控制器将停止主要数据结构和交替数据结构的转换：

- 主控处理器将交替数据结构配置为一个基础周期
- 读取到一个无效数据结构

注：当完成 N 次主要数据结构的传输后，可设置 cycle_ctrl 为 b000 使其变为无效的数据结构。

当完成一个基础周期的分散-聚集模式传输，控制器将 dma_done[C]置为有效。

在分散-聚集模式下，控制器利用主要数据结构来编程交替数据结构。下表列出了主要数据结构的 channel_cfg 配置，分为固定值配置和用户配置。

位	域	值	功能描述
固定值			
[31：30]	dst_inc	b10	配置控制器使用字作为地址增量
[29：28]	dst_size	b10	配置控制器使用字传输
[27：26]	src_inc	b10	配置控制器使用字作为地址增量
[25：24]	src_size	b10	配置控制器使用字传输
[17：14]	R_power	b1101	配置控制器进行 4 次 DMA 传输
[2：0]	cycle_ctrl	b100	配置控制器执行外设分散-聚集 DMA 周期
用户定义			
[23：21]	dst_prot_ctrl	-	当写入目标数据后，配置 HPROT 的状态
[20：18]	src_prot_ctrl	-	当读出源数据后，配置 HPROT 的状态
[13：4]	n_minus_1	N[a]	配置控制器进行 N 次 DMA 传输，其中 N 为 4 的倍数
[3]	next_useburst	-	当设置为 1，在交替传输完成后，控制器会将 CHNL_USEBURST_SET[C]置 1。

表 11-6 外设分散-聚集数据结构

注：由于 R_power 配置为 4，因此 N 必须设置为 4 的倍数。N/4 的值为配置交替数据结构的次数。

下图为外设分散-聚集模式的示例：

初始化：

1. 配置主要数据结构来使能复制 A，B，C 和 D 的操作：cycle_ctrl = b110，2^R = 4，N = 16
2. 利用下表中的结构，将主要源数据写入存储器。

	src_data_end_ptr	dst_data_end_ptr	channel_cfg	Unused
任务 A 数据	0x0A000000	0x0AE00000	Cycle_ctrl = b111，2^R = 4，N = 3	0xXXXXXXXX
任务 B 数据	0x0B000000	0x0BE00000	Cycle_ctrl = b111，2^R = 2，N = 8	0xXXXXXXXX
任务 C 数据	0x0C000000	0x0CE00000	Cycle_ctrl = b111，2^R = 8，N = 5	0xXXXXXXXX
任务 D 数据	0x0D000000	0x0DE00000	Cycle_ctrl = b001，2^R = 4，N = 4	0xXXXXXXXX

表 11-7 各任务描述符配置示例

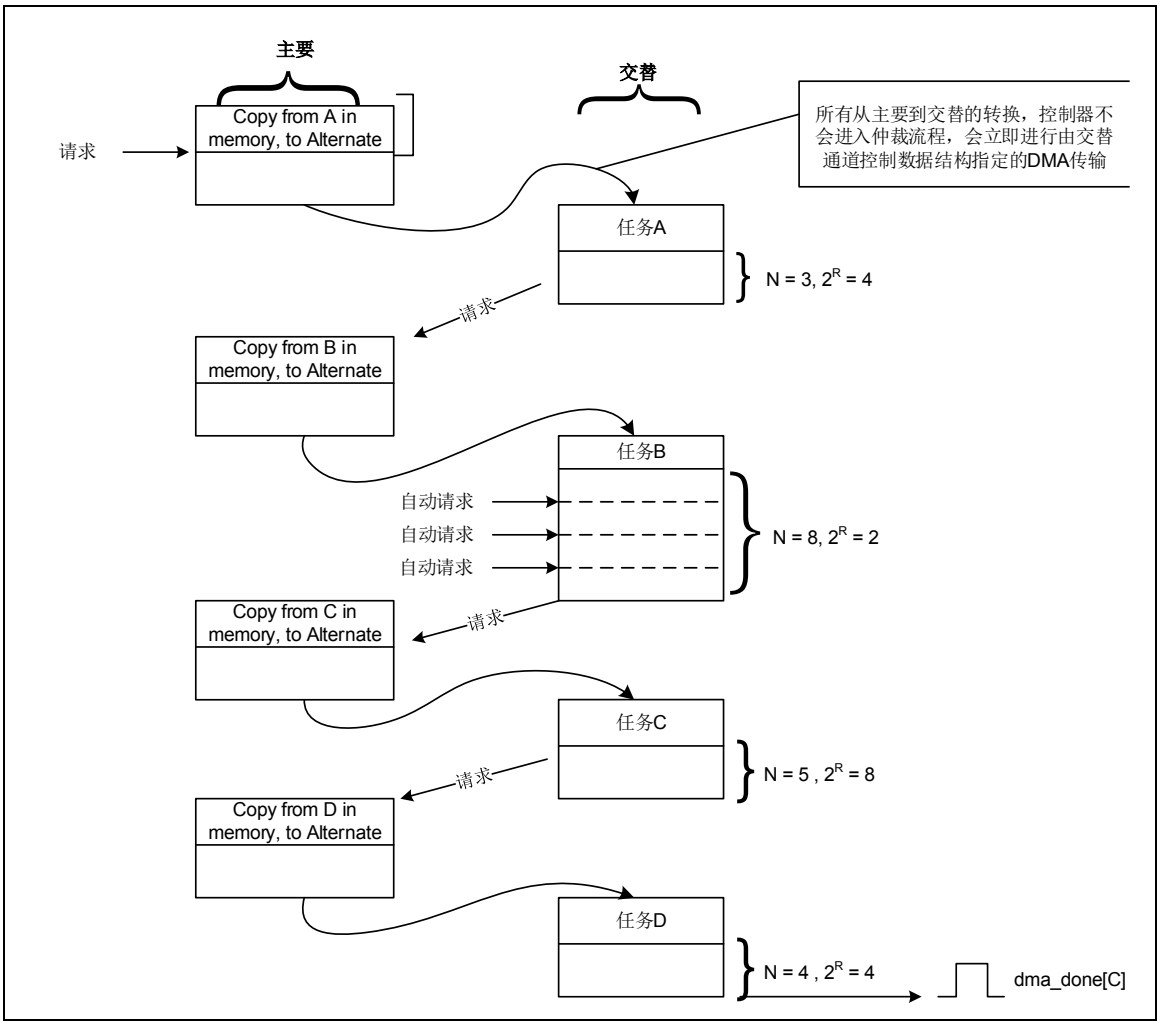


图 11-5 外设分散-聚集示例

初始化

1. 主控处理器通过设置 cycle_ctrl 为 b110，使主要数据结构运行于外设分散-聚集模式。由于单个通道的数据结构包含 4 个字，所以 2^R 必须设置为 4。在该示例中，有 4 个任务，因此 N 设为 16。

2. 主控处理器将任务 A，B，C 和 D 的数据结构写入由主要 src_data_end_ptr 指定的存储器地址中。

3. 主控处理器使能该通道。

当控制器接收到 dma_req[ ]请求，则外设分散-聚集传输开始执行，流程如下：

主要，复制 A

1. 在接收到请求后，控制器进行 4 次 DMA 传输，并且任务 A 写为交替数据结构。

任务 A

2. 控制器进行任务 A。

3. 当完成任务 A 后，控制器进入仲裁流程。

在外设发出一个新的请求且拥有最高优先级，则继续以下流程：

主要，复制 B

4. 控制器进行 4 次 DMA 传输，且任务 B 写为交替数据结构。

任务 B

5. 控制器执行任务 B。为使控制器完成该任务，外设还必须再发出 3 次请求。

6. 当任务 B 完成，控制器进入仲裁流程。

在外设发出一个新的请求且拥有最高优先级，则继续以下流程：

主要，复制 C

7. 控制器进行 4 次 DMA 传输，且任务 C 写为交替数据结构。

任务 C

8. 控制器执行任务 C。

9. 当任务 C 完成，控制器进入仲裁流程。

在外设发出一个新的请求且拥有最高优先级，则继续以下流程：

主要，复制 D

10. 控制器进行 4 次 DMA 传输，且任务 D 写为交替数据结构。

11. 控制器设置主要数据结构的 cycle_ctrl 为 b000，表明该数据结构为无效。

任务 D

12. 控制器使用基础 DMA 周期执行任务 D。

13. 控制器将 dma_done[C]置高一个 hclk 周期，并进入仲裁流程。

11. 4. 2. 4 错误信号

如果控制器在 AHB-Lite 主机接口上检测到一个错误回应，则：

◇ 禁止该错误信号对应的通道

◇ 置 dma_err 为高电平

在主控处理器检测到 dma_err 为高时，必须检查当错误发生时哪个通道被使能了。可以通过以下步骤来完成：

1. 读取 CHNL_ENABLE_SET 寄存器，获取一张禁止通道的列表。

当某通道将 dma_done[ ]置为有效，则控制器将禁止该通道。主控处理器上的程序必须记录哪些通道曾置高 dma_done[ ]输出。

2. 将步骤 1 中得到的列表和主程序上的记录作比较。若某通道没有 dma_done[C]被置高的记录，则就是该通道发生了错误信号。

11. 4. 3 通道控制数据结构

用户必须提供系统存储器空间用来包含通道控制数据结构。该系统存储器必须：

◇ 提供一个连续的存储空间，以便控制器和主控处理器可以访问

◇ 其基地址为通道控制数据结构总容量的整数倍

下图为当使用 16 个通道和交替数据结构时，控制器所需要的通道控制数据的存储器映射。

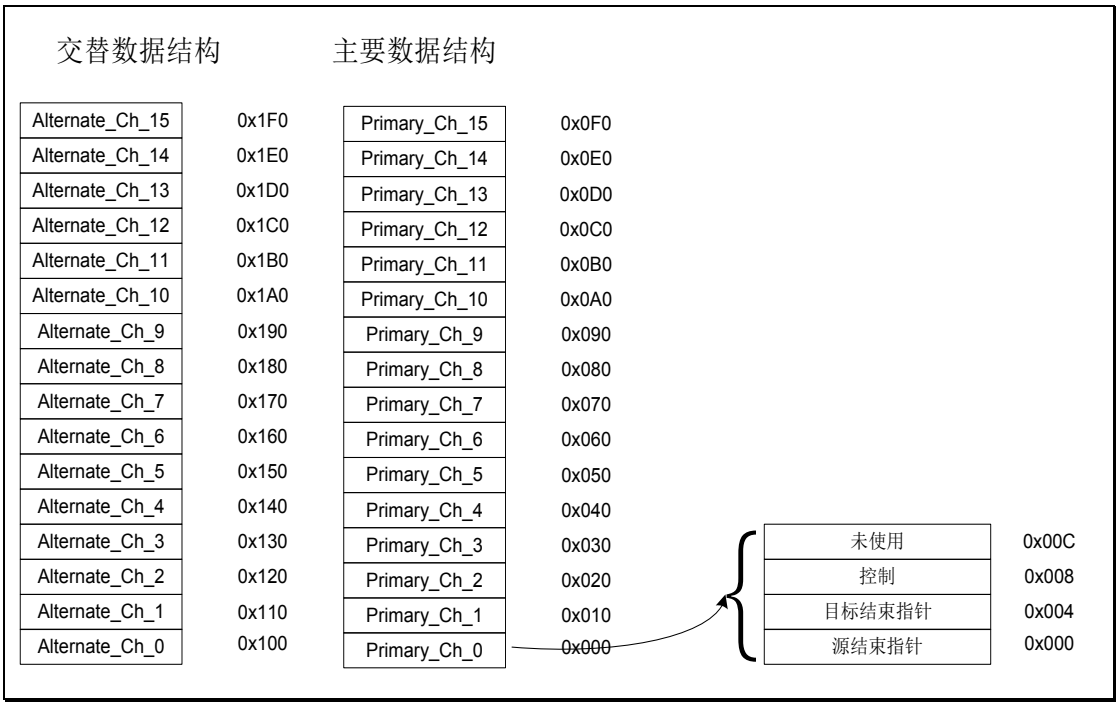


图 11-6 16 通道存储器映射（包括交替数据结构）

图中的通道数据结构使用了 448Bytes 的系统存储器。该示例中，控制器用地址低 9 位来访问数据结构中的所有地址，因此基地址必须为 0xXXXXX000，0xXXXXX200，0xXXXXX400 或者 0xXXXXX800。

通过准确设置 CTRL_BASE_PTR 寄存器，用户可配置主要数据结构的基地址。

控制器利用系统存储器来访问两个指针及每个通道的控制信息。以下内容详细描述了 32 位存储地址及 DMA 传输地址的计算方法。

- ◇ 源数据结束指针
- ◇ 目标数据结束指针
- ◇ 控制数据配置
- ◇ 地址计算

11. 4. 3. 1 源数据结束指针

src_data_end_ptr 存储地址包含一个指针，指向源数据的最后一个地址。

在执行 DMA 传输前，该存储地址写入源数据的结束地址。当启动 2^R 次 DMA 传输时，控制器读取 src_data_end_ptr。注意控制器不能写该存储器地址。

11. 4. 3. 2 目标数据结束指针

dst_data_end_ptr 存储地址包含一个指针，指向目标数据的最后一个地址。

在执行 DMA 传输前，该存储地址写入目标数据的结束地址。当启动 2^R 次 DMA 传输时，控制器读取 dst_data_end_ptr。注意控制器不能写该存储器地址。

11. 4. 3. 3 控制数据配置

channel_cfg 会向控制器提供每一次 DMA 传输的控制信息。

通道配置（channel_cfg）																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dst_inc		dst_size		src_inc		src_size		dst_prot_ctrl			src_prot_ctrl			R_power				n_minus_1										next_useburst		cycle_ctrl	

dst_inc	Bit 31-30	R/W	**目标地址增量** 地址增量取决于源数据的宽度： 源数据宽度 = 字节 b00 = 字节 b01 = 半字 b10 = 字 b11 = 无增量。地址仍然是 dst_data_end_ptr 中包含的地址。 源数据宽度 = 半字 b00 = 保留 b01 = 半字 b10 = 字 b11 = 无增量。地址仍然是 dst_data_end_ptr

			中包含的地址。 源数据宽度 = 字 b00 = 保留 b01 = 保留 b10 = 字 b11 = 无增量。地址仍然是 dst_data_end_ptr 中包含的地址。
dst_size	Bit 29-28	R/W	**目标数据大小** 需要注意的是 dst_size 和 src_size 的值必须一致。
src_inc	Bit 27-26	R/W	**源地址增量** 地址增量取决于源数据的宽度： 源数据宽度 = 字节 b00 = 字节 b01 = 半字 b10 = 字 b11 = 无增量。地址仍然是 src_data_end_ptr 中包含的地址。 源数据宽度 = 半字 b00 = 保留 b01 = 半字 b10 = 字 b11 = 无增量。地址仍然是 src_data_end_ptr 中包含的地址。 源数据宽度 = 字 b00 = 保留 b01 = 保留 b10 = 字 b11 = 无增量。地址仍然是 src_data_end_ptr 中包含的地址。
src_size	Bit 25-24	R/W	**设置该位段用来匹配源数据大小** b00 = 字节 b01 = 半字 b10 = 字 b11 = 保留
dst_prot_ctrl	Bit 23-21	R/W	**当控制器写目标数据的时候，HPROT 状态控制位** bit[23] 该位对 DMA 无效 bit[22] 该位对 DMA 无效 bit[21] HPROT 状态控制位 0=HPROT 为低，访问为无特权模式 1=HPROT 为高，访问为特权模式
src_prot_ctrl	Bit 20-18	R/W	**当控制器读取源数据的时候，HPROT 状态控制位** bit[20] 该位对 DMA 无效 bit[19] 该位对 DMA 无效 bit[18] HPROT 状态控制位

			0=HPROT 为低，访问为无特权模式 1=HPROT 为高，访问为特权模式
R_power	Bit 17-14	R/W	**在控制器重新仲裁前，该位段决定了 DMA 传输可以发生的次数** b0000：发生 1 次 DMA 传输后仲裁 b0001：发生 2 次 DMA 传输后仲裁 b0010：发生 4 次 DMA 传输后仲裁 b0011：发生 8 次 DMA 传输后仲裁 b0100：发生 16 次 DMA 传输后仲裁 b0101：发生 32 次 DMA 传输后仲裁 b0110：发生 64 次 DMA 传输后仲裁 b0111：发生 128 次 DMA 传输后仲裁 b1000：发生 256 次 DMA 传输后仲裁 b1001：发生 512 次 DMA 传输后仲裁 b1010-b1111：发生 1024 次 DMA 传输后仲裁。由于最大的传输次数为 1024，由此表明 DMA 传输中无仲裁发生。
n_minus_1	Bit 13-4	R/W	在 DMA 周期开始前，该位段代表的是 DMA 周期中包含的 DMA 传输的总次数。用户须根据所需要的 DMA 周期的大小来设置该位段。该 10 位值为 DMA 传输次数减 1。 b000000000：1 次 DMA 传输 b000000001：2 次 DMA 传输 b000000010：3 次 DMA 传输 b000000011：4 次 DMA 传输 b000000100：5 次 DMA 传输 ..... b111111111：1024 次 DMA 传输 在进入仲裁流程前，控制器会立即更新该位段，可使控制器保存该 DMA 周期中还需完成的剩下的 DMA 传输次数。
next_useburst	Bit 3	R/W	**当控制器在外设分散-聚集模式下，且使用交替数据结构，该位段控制 CHNL_USEBURST_SET[C]是否置 1。** 需要注意的是，在完成由交替数据结构指定的 DMA 周期前，如果剩余的传输次数小于 2^R，控制器会将 CHNL_USEBURST_SET[C]设置为 0。 Next_useburst 的设定控制了是否需要再次修改 CHNL_USEBURST_SET[C]。 在外设分散-聚集 DMA 周期模式下，当使用交替数据结构的 DMA 周期完成后，会发生以下任一情况： 0：控制器不改变 CHNL_USEBURST_SET[C]的值。 当进行使用交替数据结构的 DMA 周期时，如果 CHNL_USEBURST_SET[C]为 0，则对于所有在外设

			分散-聚集模式下的剩余的 DMA 周期，控制器将回应 dma_req[ ]和 dma_sreq[ ]的请求。 1：控制器将 CHNL_USEBURST_SET[C]置 1。因此，当进行使用交替数据结构的 DMA 周期时，对于所有在外设分散-聚集模式下的剩余的 DMA 周期，控制器仅回应 dma_req[ ]的请求。
cycle_ctrl	Bit 2-0	R/W	**DMA 周期的工作模式：** b000：停止。表明该数据结构无效。 b001：基础。在进入仲裁流程前，控制器必须接收到一个新的请求才能完成 DMA 周期。 b010：自动请求。在仲裁过程中，控制器在合适的通道上自动插入一个请求。这就意味着最初的请求已足够完成 DMA 周期。 b011：乒乓。控制器使用一种数据结构完成一个 DMA 周期。当该周期完成后，控制器使用另一种数据结构完成一个 DMA 周期。在该周期完成后，且如果主控处理器已更新了原始数据结构，则控制器使用原始数据结构执行一次 DMA 周期。控制器将会继续进行 DMA 周期，直到读取到无效数据结构或者主控处理器将 cycle_ctrl 改为 b001 或 b010。 b100：存储器分散-聚集。当控制器运行在存储器分散-聚集模式下，用户仅可在主要数据结构下使用该值。 b101：存储器分散-聚集。当控制器运行在存储器分散-聚集模式下，用户仅可在交替数据结构下使用该值。 b110：外设分散-聚集。当控制器运行在外设分散-聚集模式下，用户仅可在主要数据结构下使用该值。 b111：外设分散-聚集。当控制器运行在外设分散-聚集模式下，用户仅可在交替数据结构下使用该值。

表 11-8 控制信息格式描述

在 DMA 周期或者 2^R DMA 传输开始的时候，控制器会从系统存储器中获取 channel_cfg 的值。当完成 2^R 或 N 次传输后，新的 channel_cfg 值会被存储到系统存储器中。

控制器不支持 dst_size 和 src_size 拥有两个不同的值。如果检测到两个值不匹配，src_size 的值会被作为源数据和目标数据的大小。当下一次 n_minus_1 更新时，dst_size 将被设置为 src_size 的值。

当完成 N 次传输后，控制器会将 cycle_ctrl 设置为 b000，以此表明 channel_cfg 数据为无效，用来防止控制器重复相同的 DMA 传输。

11. 4. 3. 4 地址计算

为了计算 DMA 传输的源地址，控制器必须将 n_minus_1 的值向左移，移动值由 src_inc 定义。接着从源数据结束指针中减去移动后的 n_minus_1 的值。同样地，为了计算 DMA 传输的目标地址，控制器也必须将 n_minus_1 的值向左移，移动量由 dst_inc 定义，接着从目标数据结束指针中减去移动后的 n_minus_1 的值。

根据 src_inc 和 dst_inc 的值，源地址与目标地址可用以下等式计算：

Src_inc = b00，dst_inc = b00

源地址 = src_data_end_ptr – n_minus_1

目标地址 = dst_data_end_ptr - n_minus_1

Src_inc = b01，dst_inc = b01

源地址 = src_data_end_ptr –（n_minus_1<<1）

目标地址 = dst_data_end_ptr –（n_minus_1<<1）

Src_inc = b10，dst_inc = b10

源地址 = src_data_end_ptr –（n_minus_1<<2）

目标地址 = dst_data_end_ptr –（n_minus_1<<2）

Src_inc = b11，dst_inc = b11

源地址 = src_data_end_ptr

目标地址 = dst_data_end_ptr

下表为 6 个字的 DMA 周期，地址增量为 1 个字。

channel_cfg 初始值（DMA 周期前）				
src_size=b10，dst_inc=b10，n_minus_1=b101，cycle_ctrl=1				
DMA 传输	结束指针	次数	差值	地址
	0x2AC	5	0x14	0x298
	0x2AC	4	0x10	0x29C
	0x2AC	3	0xC	0x2A0
	0x2AC	2	0x8	0x2A4
	0x2AC	1	0x4	0x2A8
	0x2AC	0	0x0	0x2AC
channel_cfg 最终值（DMA 周期后）				
src_size=b10，dst_inc=b10，n_minus_1=0，cycle_ctrl=0				

表 11-9 传输过程中指针变化

注：上表中的“差值”为“次数”向左移动 dst_inc 对应位数后的结果。

channel_cfg 初始值（DMA 周期前）				
src_size=b00，dst_inc=b01，n_minus_1=b1011，cycle_ctrl=1，R_power=b11				
DMA 传输	结束指针	次数	差值	地址
	0x5E7	11	0x16	0x5D1
	0x5E7	10	0x14	0x5D3
	0x5E7	9	0x12	0x5D5
	0x5E7	8	0x10	0x5D7
	0x5E7	7	0xE	0x5D9
	0x5E7	6	0xC	0x5DB
	0x5E7	5	0xA	0x5DD
	0x5E7	4	0x8	0x5DF
2^R 次 DMA 传输完成后，channel_cfg 的值				
src_size=b00，dst_inc=b01，n_minus_1=b011，cycle_ctrl=1，R_power=b11				
DMA 传输	结束指针	次数	差值	地址
	0x5E7	3	0x6	0x5E1
	0x5E7	2	0x4	0x5E3
	0x5E7	1	0x2	0x5E5
	0x5E7	0	0x0	0x5E7
channel_cfg 最终值（DMA 周期后）				
src_size=b00，dst_inc=b01，n_minus_1=0，cycle_ctrl=0，R_power=b11				

表 11-10 传输过程中指针变化

注 1：上表中的“差值”为“次数”向左移动 dst_inc 对应位数后的结果。

注 2：当控制器完成 DMA 周期后，通过将 cycle_ctrl 清零使 channel_cfg 无效。

11. 5 特殊功能寄存器

该章节描述了 DMAC 寄存器及提供了编程控制器的信息。

- ◇ 编程器模型介绍
- ◇ 寄存器描述

11. 5. 1 编程器模型介绍

下列条件适用于控制器提供的寄存器：

- ◇ 控制器的基地址并非固定不变的，但任何特定寄存器的偏移地址都是固定的。
- ◇ 用户禁止访问任何保留的地址，否则可导致控制器产生不可预期的行为。
- ◇ 除非有其他相关说明，用户必须将保留的和未使用的寄存器位写 0，无视读取值。
- ◇ 除非有其他相关说明，系统复位或者上电复位会将所有寄存器位写 0。
- ◇ 除非有其他相关说明，所有寄存器都支持读写访问。写操作可更新寄存器内容，读操作则返回寄存器内容。

11. 5. 2 寄存器列表

DMA 寄存器列表		
名称	偏移地址	描述
DMA_STATUS	0000_H	DMA 状态寄存器
DMA_CFG	0004_H	DMA 配置寄存器
DMA_CTRLBASE	0008_H	DMA 通道控制数据基指针寄存器
DMA_ALTCTRLBASE	$000C_H$	DMA 通道交替控制数据基指针寄存器
DMA_CHWAITSTATUS	0010_H	DMA 通道等待请求状态寄存器
DMA_CHSWREQ	0014_H	DMA 通道软件请求寄存器
DMA_CHUSEBURSTSET	0018_H	DMA 通道使用突发设置寄存器
DMA_CHUSEBURSTCLR	$001C_H$	DMA 通道使用突发清除寄存器
DMA_CHREQMASKSET	0020_H	DMA 通道请求屏蔽设置寄存器
DMA_CHREQMASKCLR	0024_H	DMA 通道请求屏蔽清除寄存器
DMA_CHENSET	0028_H	DMA 通道使能设置寄存器
DMA_CHENCLR	$002C_H$	DMA 通道使能清除寄存器
DMA_CHPRIALTSET	0030_H	DMA 通道主要-交替设置寄存器
DMA_CHPRIALTCLR	0034_H	DMA 通道主要-交替清除寄存器
DMA_CHPRSET	0038_H	DMA 通道优先级设置寄存器
DMA_CHPRCLR	$003C_H$	DMA 通道优先级清除寄存器
Reserved	0040_H~0048_H	保留
DMA_ERRCLR	$004C_H$	DMA 总线错误清除寄存器
Reserved	0050_H~$0FFC_H$	保留
DMA_IFLAG	1000_H	DMA 中断标志寄存器
Reserved	1004_H	保留
DMA_ICFR	1008_H	DMA 中断标志清零寄存器
DMA_IER	$100C_H$	DMA 中断使能控制寄存器
Reserved	1010_H~$10FC_H$	保留
DMA_CH0_SELCON	1100_H	DMA 通道 0 复用选择寄存器
DMA_CH1_SELCON	1104_H	DMA 通道 1 复用选择寄存器
DMA_CH2_SELCON	1108_H	DMA 通道 2 复用选择寄存器
DMA_CH3_SELCON	$110C_H$	DMA 通道 3 复用选择寄存器
DMA_CH4_SELCON	1110_H	DMA 通道 4 复用选择寄存器
DMA_CH5_SELCON	1114_H	DMA 通道 5 复用选择寄存器
DMA_CH6_SELCON	1118H	DMA 通道 6 复用选择寄存器
DMA_CH7_SELCON	111CH	DMA 通道 7 复用选择寄存器
DMA_CH8_SELCON	1120H	DMA 通道 8 复用选择寄存器
DMA_CH9_SELCON	1124H	DMA 通道 9 复用选择寄存器
DMA_CH10_SELCON	1128H	DMA 通道 10 复用选择寄存器
DMA_CH11_SELCON	112CH	DMA 通道 11 复用选择寄存器

11. 5. 3 寄存器描述

11. 5. 3. 1 DMA状态寄存器（DMA_STATUS）

该寄存器可返回控制器的状态，为只读寄存器。当控制器处于复位状态时，用户不能读取该寄存器。

DMA 状态寄存器（DMA_STATUS）

偏移地址：000_H

复位值：$00010000\_00001011\_00000000\_00000000_B$

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

Reserved	Bit 31-8	-	**保留，写 0。**
STATUS	Bit 7-4	R	**当前状态控制位** b0000：空闲 b0001：读取通道控制器数据 b0010：读取源数据结束指针 b0011：读取目标数据结束指针 b0100：读取源数据 b0101：写目标数据 b0110：等待 DMA 请求清 0 b0111：写通道控制器数据 b1000：延迟 b1001：完成 b1010：外设分散-聚集转换 b1011-b1111：未定义
Reserved	Bit 3-1	-	**保留，写 0。**
MASTER_ENABLE	Bit 0	R	**使能控制器** 0：禁止控制器 1：使能控制器

11. 5. 3. 2 DMA配置寄存器（DMA_CFG）

该寄存器为只写寄存器，可对控制器进行配置。

DMA 配置寄存器（DMA_CFG）

偏移地址：004_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-6	-	**保留，写0。**
CHNL_PROT_CTRL	Bit 5	W	**通道保护控制** 控制 DMA 访问是否具有特权 0：HPROT 为低，访问为无特权模式 1：HPROT 为高，访问为特权模式
Reserved	Bit 4-1	-	**保留，写 0。**
MASTER_ENABLE	Bit 0	W	**使能控制器** 0：禁止控制器 1：使能控制器

11. 5. 3. 3 DMA通道控制数据基指针寄存器（DMA_CTRLBASE）

该寄存器为读写寄存器。用户须配置该寄存器用来指向系统存储器中的地址。
需要指出的是，控制器不提供内部存储器用来存储通道控制数据结构。
控制器中用户所需指定的系统存储器的大小取决于以下两个条件：DMA 通道的个数和是否使用交替数据结构。
当控制器处于复位状态时，用户不可读取该寄存器。

DMA 通道控制数据基指针寄存器（DMA_CTRLBASE）

偏移地址：008_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

CTRL_BASE_PTR	Bit 31-9	R/W	**通道控制数据基地址指针** 该位段为系统内存中保存通道控制数据结构的位置的基地址指针。在使用DMA之前，该寄存器必须写入指向具有通道控制数据结构的系统内存的某个位置。
Reserved	Bit 8-0	-	**保留，写 0。**

11. 5. 3. 4 DMA通道交替控制数据基指针寄存器（DMA_ALTCTRLBASE）

该寄存器为只读寄存器，可返回交替数据结构的基地址，因此用户无需在应用软件中计算交替数据结构的基地址。当控制器处于复位状态时，用户不可读取该寄存器。

DMA 通道交替控制数据基指针寄存器（DMA_ALTCTRLBASE）

偏移地址：$00C_H$

复位值：$00000000\_00000000\_00000001\_00000000_B$

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

ALT_CTRL_BASE_PTR	Bit 31-0	R	**通道交替控制数据基地址指针** 该位段为交替数据结构的基地址。该寄存器会被读取为DMA_CTRLBASE + 0x100。

11. 5. 3. 5 DMA通道等待请求状态寄存器（DMA_CHWAITSTATUS）

该寄存器为只读寄存器，可返回 dma_waitonreq[ ]的状态。当控制器处于复位状态时，用户不可读取该寄存器。Bit0 返回的为 dma_waitonreq[0]的状态，Bit1 返回的为 dma_waitonreq[1]的状态，以此类推，Bit31 返回的为 dma_waitonreq[31]的状态。

DMA 通道等待请求状态寄存器（DMA_CHWAITSTATUS）

偏移地址：010_H

复位值：$00000000\_00000000\_00001111\_11111111_B$

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

DMA_WAITONREQ_STATUS	Bit 31-0	R	**通道等待请求状态** 相应通道的等待请求状态。通道对应状态位如果为1表示该通道一旦存在DMA请求，DMA将一直处于工作状态直到请求被响应。

11. 5. 3. 6 DMA通道软件请求寄存器（DMA_CHSWREQ）

该寄存器为只写寄存器，每个对应的位都可在相应的通道上生产软件 DMA 请求。

DMA 通道软件请求寄存器（DMA_CHSWREQ）

偏移地址：014_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

CHSWREQ	Bit 31-0	W	**通道软件请求** 0：无效 1：为相应通道生成DMA请求。

11. 5. 3. 7 DMA通道使用突发设置寄存器（DMA_CHUSEBURSTSET）

该寄存器为可读写寄存器，可禁止单次请求 dma_sreq[ ]产生请求信号，因此仅有 dma_req[ ]产生请求。读取该寄存器可返回突发的使用状态。每个位都有相对应的通道，bit0 对应通道 0，bit1 对应通道 1，以此类推。

当完成倒数第二次 2^R 传输，如果剩余的次数 N 小于 2^R，控制器会将 CHNL_USEBURST_SET 复位成 0。剩余的传输次数可通过 dma_req[ ]或 dma_sreq[ ]完成。

注：当 channel_cfg 的设置值 N 小于 2^R，如果外设没有将 dma_req[ ]置为有效，则不应该将 CHNL_USEBURST_SET 置 1。

在外设分散-聚集模式下，当使用交替数据结构的 DMA 周期完成的时候，如果 channel_cfg 中的 next_useburst 已被设置为 1，控制器会将 CHNL_USEBURST_SET[C]置 1。

DMA 通道使用突发设置寄存器（DMA_CHUSEBURSTSET）

偏移地址：018_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

CHNL_USEBURST_SET	Bit 31-0	R/W	**通道使用突发设置** 读取： 0：相应通道响应单一的请求和突发请求 1：相应通道仅响应突发请求 写入： 0：无效 1：使能相应通道上的使用突发设置。

11. 5. 3. 8 DMA通道使用突发清除寄存器（DMA_CHUSEBURSTCLR）

DMA 通道使用突发清除寄存器（DMA_CHUSEBURSTCLR）																															
偏移地址：$01C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_USEBURST_CLR																															

CHNL_USEBURST_CLR	Bit 31-0	W	**通道使用突发清除** 0：无效 1：禁止相应通道上的使用突发设置。

11. 5. 3. 9 DMA通道请求屏蔽设置寄存器（DMA_CHREQMASKSET）

该寄存器为读写寄存器，可禁止 dma_req[ ]和 dma_sreq[ ]产生请求。读取时可返回 dma_req[ ]和 dma_sreq[ ]的屏蔽状态。

DMA 通道请求屏蔽设置寄存器（DMA_CHREQMASKSET）																															
偏移地址：020_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_REQ_MASK_SET																															

CHNL_REQ_MASK_SET	Bit 31-0	R/W	**通道请求屏蔽设置** 读取： 0：相应通道上的外部请求已使能。 1：相应通道上的外部请求已禁止。 写入： 0：无效 1：禁止相应通道上的外设请求

11. 5. 3. 10 DMA通道请求屏蔽清除寄存器（DMA_CHREQMASKCLR）

DMA 通道请求屏蔽清除寄存器（DMA_CHREQMASKCLR）																															
偏移地址：024_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_REQ_MASK_CLR																															

CHNL_REQ_MASK_CLR	Bit 31-0	W	**通道请求屏蔽清除** 0：无效 1：使能相应通道上的外设请求。

11. 5. 3. 11 DMA通道使能设置寄存器（DMA_CHENSET）

该寄存器为读写寄存器，设置该寄存器可使能通道。读取时，可返回该通道的状态。

DMA 通道使能设置寄存器（DMA_CHENSET）																															
偏移地址：028_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_ENABLE_SET																															

CHNL_ENABLE_SET	Bit 31-0	R/W	**通道使能设置** 读取： 0：相应通道禁止 1：相应通道使能 写入： 0：无效 1：使能相应通道。

11. 5. 3. 12 DMA通道使能清除寄存器（DMA_CHENCLR）

该寄存器为只写寄存器，可用来禁止相应的通道。

DMA 通道使能清除寄存器（DMA_CHENCLR）

偏移地址：$02C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

CHNL_ENABLE_CLR	Bit 31-0	W	**通道使能清除** 0：无效 1：禁止相应通道。

注：当发生以下任意一种情况时，控制器可通过设置相应 CHNL_ENABLE_CLR 来禁止通道：

1.控制器完成 DMA 周期

2.控制器读取 channel_cfg，其 cycle_ctrl 位段为 b000

3.AHB-Lite 总线上发生错误

11. 5. 3. 13 DMA通道主要-交替设置寄存器（DMA_CHPRIALTSET）

该寄存器为读写寄存器。通过该寄存器，用户可将 DMA 通道设置为使用交替数据结构。读取时，可返回相应通道的数据结构使用状态。

DMA 通道主要-交替设置寄存器（DMA_CHPRIALTSET）																															
偏移地址：030_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_PRI_ALT_SET																															

CHNL_PRI_ALT_SET	Bit 31-0	R/W	**通道交替结构设置** 读取： 0：相应DMA通道使用的是主要数据结构。 1：相应DMA通道使用的是交替数据结构。 写入： 0：无效。设置DMA_CHPRIALTCLR寄存器来选择主要数据结构 1：为相应通道选择交替数据结构

11. 5. 3. 14 DMA通道主要-交替清除寄存器（DMA_CHPRIALTCLR）

该寄存器为只写寄存器。通过该寄存器，用户可将 DMA 通道设置为使用主要数据结构。

DMA 通道主要-交替清除寄存器（DMA_CHPRIALTCLR）																															
偏移地址：034_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_PRI_ALT_CLR																															

CHNL_PRI_ALT_CLR	Bit 31-0	W	**通道交替结构清除** 0：无效。设置DMA_CHPRIALTSET寄存器用来选择交替数据结构 1：为相应通道选择主要数据结构

11. 5. 3. 15 DMA通道优先级设置寄存器（DMA_CHPRSET）

该寄存器为读写寄存器。通过该寄存器，用户可配置 DMA 通道为高优先级。读取时，可返回通道优先级的屏蔽状态。

DMA 通道优先级设置寄存器（DMA_CHPRSET）																															
偏移地址：038_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_PRIORITY_SET																															

CHNL_PRIORITY_SET	Bit 31-0	R/W	**通道高优先级设置** 读取： 0：相应通道为默认优先级 1：相应通道为高优先级 写入： 0：无效 1：设置相应通道为高优先级。

11. 5. 3. 16 DMA通道优先级清除寄存器（DMA_CHPRCLR）

该寄存器为只写寄存器。通过该寄存器，用户可将 DMA 通道设置为默认优先级。

DMA 通道优先级清除寄存器（DMA_CHPRCLR）																															
偏移地址：$03C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_PRIORITY_CLR																															

CHNL_PRIORITY_CLR	Bit 31-0	W	**通道高优先级清除** 0：无效 1：设置相应通道为默认优先级

11. 5. 3. 17 DMA总线错误清除寄存器（DMA_ERRCLR）

该读写寄存器可返回 dma_err 的状态，可将 dma_err 设置为低电平。

DMA 总线错误清除寄存器（DMA_ERRCLR）

偏移地址：$04C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-1	-	**保留，写0。**
ERR_CLR	Bit 0	R/W	**总线错误清除** 如果AHB总线错误发生，该位会被置1。向该位写1会清除该位。当AHB-Lite总线上发生错误的同时，若ERR_CLR被置为无效，则错误条件先发生，ERR_CLR保持有效。

11. 5. 3. 18 DMA中断标志寄存器（DMA_IFLAG）

DMA 中断标志寄存器（DMA_IFLAG）																															
偏移地址：1000_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAERRIF	Reserved																			CH11DONEIF	CH10DONEIF	CH9DONEIF	CH8DONEIF	CH7DONEIF	CH6DONEIF	CH5DONEIF	CH4DONEIF	CH3DONEIF	CH2DONEIF	CH1DONEIF	CH0DONEIF

DMAERRIF	Bit 31	R	**DMA 错误中断标志** 当 AHB 总线上发生错误时，该标志位置 1。
Reserved	Bit 30-12	–	**保留**
CH11DONEIF	Bit 11	R	**DMA 通道 11 结束中断标志** 当 DMA 通道完成传输时该位置 1。如果通道被禁止，当该通道上存在请求时该标志置 1。
CH10DONEIF	Bit 10	R	**DMA 通道 10 结束中断标志** 当 DMA 通道完成传输时该位置 1。如果通道被禁止，当该通道上存在请求时该标志置 1。
CH9DONEIF	Bit 9	R	**DMA 通道 9 结束中断标志** 当 DMA 通道完成传输时该位置 1。如果通道被禁止，当该通道上存在请求时该标志置 1。
CH8DONEIF	Bit 8	R	**DMA 通道 8 结束中断标志** 当 DMA 通道完成传输时该位置 1。如果通道被禁止，当该通道上存在请求时该标志置 1。
CH7DONEIF	Bit 7	R	**DMA 通道 7 结束中断标志** 当 DMA 通道完成传输时该位置 1。如果通道被禁止，当该通道上存在请求时该标志置 1。
CH6DONEIF	Bit 6	R	**DMA 通道 6 结束中断标志** 当 DMA 通道完成传输时该位置 1。如果通道被禁止，当该通道上存在请求时该标志置 1。
CH5DONEIF	Bit 5	R	**DMA 通道 5 结束中断标志** 当 DMA 通道完成传输时该位置 1。如果通道被禁止，当该通道上存在请求时该标志置 1。
CH4DONEIF	Bit 4	R	**DMA 通道 4 结束中断标志** 当 DMA 通道完成传输时该位置 1。如果通道被禁止，当该通道上存在请求时该标志置 1。
CH3DONEIF	Bit 3	R	**DMA 通道 3 结束中断标志** 当 DMA 通道完成传输时该位置 1。如果通道被禁止，当该通道上存在请求时该标志置 1。
CH2DONEIF	Bit 2	R	**DMA 通道 2 结束中断标志** 当 DMA 通道完成传输时该位置 1。如果通道被禁止，当该通道上存在请求时该标志置 1。

CH1DONEIF	Bit 1	R	**DMA 通道 1 结束中断标志** 当 DMA 通道完成传输时该位置 1。如果通道被禁止，当该通道上存在请求时该标志置 1。
CH0DONEIF	Bit 0	R	**DMA 通道 0 结束中断标志** 当 DMA 通道完成传输时该位置 1。如果通道被禁止，当该通道上存在请求时该标志置 1。

11. 5. 3. 19 DMA中断标志清零寄存器（DMA_ICFR）

DMA 中断标志清零寄存器（DMA_ICFR）

偏移地址：1008_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAERRC	Reserved																			CH11DONEC	CH10DONEC	CH9DONEC	CH8DONEC	CH7DONEC	CH6DONEC	CH5DONEC	CH4DONEC	CH3DONEC	CH2DONEC	CH1DONEC	CH0DONEC

DMAERRC	Bit 31	W1	**DMA 错误中断标志清零** 对该位写 1 有效。 注：如果有错误发生，必须使用总线错误清除寄存器来清除 DMA。
Reserved	Bit 30-12	–	**保留**
CH11DONEC	Bit 11	W1	**DMA 通道 11 结束中断标志清零** 对该位写 1 有效。
CH10DONEC	Bit 10	W1	**DMA 通道 10 结束中断标志清零** 对该位写 1 有效。
CH9DONEC	Bit 9	W1	**DMA 通道 9 结束中断标志清零** 对该位写 1 有效。
CH8DONEC	Bit 8	W1	**DMA 通道 8 结束中断标志清零** 对该位写 1 有效。
CH7DONEC	Bit 7	W1	**DMA 通道 7 结束中断标志清零** 对该位写 1 有效。
CH6DONEC	Bit 6	W1	**DMA 通道 6 结束中断标志清零** 对该位写 1 有效。
CH5DONEC	Bit 5	W1	**DMA 通道 5 结束中断标志清零** 对该位写 1 有效。
CH4DONEC	Bit 4	W1	**DMA 通道 4 结束中断标志清零** 对该位写 1 有效。
CH3DONEC	Bit 3	W1	**DMA 通道 3 结束中断标志清零** 对该位写 1 有效。
CH2DONEC	Bit 2	W1	**DMA 通道 2 结束中断标志清零** 对该位写 1 有效。
CH1DONEC	Bit 1	W1	**DMA 通道 1 结束中断标志清零** 对该位写 1 有效。
CH0DONEC	Bit 0	W1	**DMA 通道 0 结束中断标志清零** 对该位写 1 有效。

11. 5. 3. 20 DMA中断使能控制寄存器（DMA_IER）

DMA 中断使能控制寄存器（DMA_IER）

偏移地址：$100C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAERRIE	Reserved																			CH11DONEIE	CH10DONEIE	CH9DONEIE	CH8DONEIE	CH7DONEIE	CH6DONEIE	CH5DONEIE	CH4DONEIE	CH3DONEIE	CH2DONEIE	CH1DONEIE	CH0DONEIE

DMAERRIE	Bit 31	R/W	**DMA 错误中断使能** 0：无效 1：在 AHB 总线错误发生时使能中断
Reserved	Bit 30-12	–	**保留**
CH11DONEIE	Bit 11	R/W	**DMA 通道 11 结束中断使能** 0：中断禁止 1：中断使能
CH10DONEIE	Bit 10	R/W	**DMA 通道 10 结束中断使能** 0：中断禁止 1：中断使能
CH9DONEIE	Bit 9	R/W	**DMA 通道 9 结束中断使能** 0：中断禁止 1：中断使能
CH8DONEIE	Bit 8	R/W	**DMA 通道 8 结束中断使能** 0：中断禁止 1：中断使能
CH7DONEIE	Bit 7	R/W	**DMA 通道 7 结束中断使能** 0：中断禁止 1：中断使能
CH6DONEIE	Bit 6	R/W	**DMA 通道 6 结束中断使能** 0：中断禁止 1：中断使能
CH5DONEIE	Bit 5	R/W	**DMA 通道 5 结束中断使能** 0：中断禁止 1：中断使能
CH4DONEIE	Bit 4	R/W	**DMA 通道 4 结束中断使能** 0：中断禁止 1：中断使能
CH3DONEIE	Bit 3	R/W	**DMA 通道 3 结束中断使能** 0：中断禁止 1：中断使能
CH2DONEIE	Bit 2	R/W	**DMA 通道 2 结束中断使能**

			0：中断禁止 1：中断使能
CH1DONEIE	Bit 1	R/W	**DMA 通道 1 结束中断使能** 0：中断禁止 1：中断使能
CH0DONEIE	Bit 0	R/W	**DMA 通道 0 结束中断使能** 0：中断禁止 1：中断使能

11. 5. 3. 21 DMA通道 0 复用选择寄存器（DMA_CH0_SELCON）

DMA 通道 0 复用选择寄存器（DMA_CH0_SELCON）

偏移地址：1100_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-14	–	保留
MSEL	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO_EXTI 000010：CRYPT 000011：预留 000100：DAC0 000101：预留 000110：ADC0 000111：CRC 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：预留 010111：预留 011000：LPUART0 011001：预留 011010：SPI2 011011：BS16T0 011100：BS16T1 011101：GP16C4T0 011110：GP16C4T1

			011111：ADC1 100000：PIS 100001：TRNG 100010：QSPI 100011：USB 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	–	保留
MSIGSEL	Bit 3-0	R/W	**MSEL=000000** MSIGSEL 无效 **MSEL=000001** 0000：EXTI0 0001：EXTI1 0010：EXTI2 0011：EXTI3 0100：EXTI4 0101：EXTI5 0110：EXTI6 0111：EXTI7 1000：EXTI8 1001：EXTI9 1010：EXTI10 1011：EXTI11 1100：EXTI12 1101：EXTI13 1110：EXTI14 1111：EXTI15 **MSEL=000010（SRC=CRYPT）时** 0000：CRYPT DMA 写数据申请 0001：CRYPT DMA 读数据申请 **MSEL=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束申请 0001：DAC0 通道 1 转换结束申请 **MSEL=000110（SRC=ADC0）** ADC0 DMA 申请 **MSEL=000111（SRC=CRC）** CRC DMA 申请 **MSEL=001000，001001，001010，001011** **（SRC=UART0，UART1，UART2，UART3）** 0000：发送保持寄存器空申请 0001：接收可用数据申请 **MSEL=001100，001101（SRC=UART4，UART5）** 0000：接收可用数据申请 0001：发送保持寄存器空申请

			MSEL=001110，001111，011010（SRC=SPI0，SPI1，SPI2） 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010010，010011，010100，010101，011101，011110（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：捕捉比较通道 1 申请 0001：捕捉比较通道 2 申请 0010：捕捉比较通道 3 申请 0011：捕捉比较通道 4 申请 0100：TIM 触发申请 0101：TIM 比较匹配申请 0110：TIM 更新事件申请 **MSEL=011011，011100（SRC=BS16T0，BS16T1）** TIM 更新事件申请 **MSEL=011000（SRC=LPUART0）** 0000：接收可用数据申请 0001：发送缓冲器空申请 **MSEL=011111（SRC=ADC1）** ADC1 DMA 申请 **MSEL=100000（SRC=PIS）** 0000：PIS 通道 0 0001：PIS 通道 1 0010：PIS 通道 2 0011：PIS 通道 3 0100：PIS 通道 4 0101：PIS 通道 5 0110：PIS 通道 6 0111：PIS 通道 7 1000：PIS 通道 8 1001：PIS 通道 9 1010：PIS 通道 10 1011：PIS 通道 11 1100：PIS 通道 12 1101：PIS 通道 13 1110：PIS 通道 14 1111：PIS 通道 15 **MSEL=100001（SRC=TRNG）**

			TRNG DMA 申请 **MSEL=100010（SRC=QSPI）** 0000：QSPI 间接写申请 0001：QSPI 间接读申请 **MSEL=100011（SRC=USB）** 0000：USB 发送端点 1 0001：USB 发送端点 2 0010：USB 发送端点 3 0011：USB 发送端点 4 0100：USB 发送端点 5 0101：保留 0110：保留 0111：保留 1000：USB 接收端点 1 1001：USB 接收端点 2 1010：USB 接收端点 3 1011：USB 接收端点 4 1100：USB 接收端点 5 1101：保留 1110：保留 1111：保留 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1

11. 5. 3. 22 DMA通道 1 复用选择寄存器（DMA_CH1_SELCON）

DMA 通道 1 复用选择寄存器（DMA_CH1_SELCON）

偏移地址：1104_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-14	–	保留
MSEL	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO_EXTI 000010：CRYPT 000011：预留 000100：DAC0 000101：预留 000110：ADC0 000111：CRC 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：预留 010111：预留 011000：LPUART0 011001：预留 011010：SPI2 011011：BS16T0 011100：BS16T1 011101：GP16C4T0 011110：GP16C4T1

			011111：ADC1 100000：PIS 100001：TRNG 100010：QSPI 100011：USB 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	–	保留
MSIGSEL	Bit 3-0	R/W	**MSEL=000000** MSIGSEL 无效 **MSEL=000001** 0000：EXTI0 0001：EXTI1 0010：EXTI2 0011：EXTI3 0100：EXTI4 0101：EXTI5 0110：EXTI6 0111：EXTI7 1000：EXTI8 1001：EXTI9 1010：EXTI10 1011：EXTI11 1100：EXTI12 1101：EXTI13 1110：EXTI14 1111：EXTI15 **MSEL=000010（SRC=CRYPT）时** 0000：CRYPT DMA 写数据申请 0001：CRYPT DMA 读数据申请 **MSEL=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束申请 0001：DAC0 通道 1 转换结束申请 **MSEL=000110（SRC=ADC0）** ADC0 DMA 申请 **MSEL=000111（SRC=CRC）** CRC DMA 申请 **MSEL=001000，001001，001010，001011** **（SRC=UART0，UART1，UART2，UART3）** 0000：发送保持寄存器空申请 0001：接收可用数据申请 **MSEL=001100，001101（SRC=UART4，UART5）** 0000：接收可用数据申请 0001：发送保持寄存器空申请

			MSEL=001110，001111，011010（SRC=SPI0，SPI1，SPI2） 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010010，010011，010100，010101，011101，011110（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：捕捉比较通道 1 申请 0001：捕捉比较通道 2 申请 0010：捕捉比较通道 3 申请 0011：捕捉比较通道 4 申请 0100：TIM 触发申请 0101：TIM 比较匹配申请 0110：TIM 更新事件申请 **MSEL=011011，011100（SRC=BS16T0，BS16T1）** TIM 更新事件申请 **MSEL=011000（SRC=LPUART0）** 0000：接收可用数据申请 0001：发送缓冲器空申请 **MSEL=011111（SRC=ADC1）** ADC1 DMA 申请 **MSEL=100000（SRC=PIS）** 0000：PIS 通道 0 0001：PIS 通道 1 0010：PIS 通道 2 0011：PIS 通道 3 0100：PIS 通道 4 0101：PIS 通道 5 0110：PIS 通道 6 0111：PIS 通道 7 1000：PIS 通道 8 1001：PIS 通道 9 1010：PIS 通道 10 1011：PIS 通道 11 1100：PIS 通道 12 1101：PIS 通道 13 1110：PIS 通道 14 1111：PIS 通道 15 **MSEL=100001（SRC=TRNG）**

			TRNG DMA 申请 **MSEL=100010（SRC=QSPI）** 0000：QSPI 间接写申请 0001：QSPI 间接读申请 **MSEL=100011（SRC=USB）** 0000：USB 发送端点 1 0001：USB 发送端点 2 0010：USB 发送端点 3 0011：USB 发送端点 4 0100：USB 发送端点 5 0101：保留 0110：保留 0111：保留 1000：USB 接收端点 1 1001：USB 接收端点 2 1010：USB 接收端点 3 1011：USB 接收端点 4 1100：USB 接收端点 5 1101：保留 1110：保留 1111：保留 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1

11. 5. 3. 23 DMA通道 2 复用选择寄存器（DMA_CH2_SELCON）

DMA 通道 2 复用选择寄存器（DMA_CH2_SELCON）

偏移地址：1108_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-14	–	保留
MSEL	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO_EXTI 000010：CRYPT 000011：预留 000100：DAC0 000101：预留 000110：ADC0 000111：CRC 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：预留 010111：预留 011000：LPUART0 011001：预留 011010：SPI2 011011：BS16T0 011100：BS16T1 011101：GP16C4T0 011110：GP16C4T1

			011111：ADC1 100000：PIS 100001：TRNG 100010：QSPI 100011：USB 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	–	保留
MSIGSEL	Bit 3-0	R/W	**MSEL=000000** MSIGSEL 无效 **MSEL=000001** 0000：EXTI0 0001：EXTI1 0010：EXTI2 0011：EXTI3 0100：EXTI4 0101：EXTI5 0110：EXTI6 0111：EXTI7 1000：EXTI8 1001：EXTI9 1010：EXTI10 1011：EXTI11 1100：EXTI12 1101：EXTI13 1110：EXTI14 1111：EXTI15 **MSEL=000010（SRC=CRYPT）时** 0000：CRYPT DMA 写数据申请 0001：CRYPT DMA 读数据申请 **MSEL=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束申请 0001：DAC0 通道 1 转换结束申请 **MSEL=000110（SRC=ADC0）** ADC0 DMA 申请 **MSEL=000111（SRC=CRC）** CRC DMA 申请 **MSEL=001000，001001，001010，001011** **（SRC=UART0，UART1，UART2，UART3）** 0000：发送保持寄存器空申请 0001：接收可用数据申请 **MSEL=001100，001101（SRC=UART4，UART5）** 0000：接收可用数据申请 0001：发送保持寄存器空申请

			MSEL=001110，001111，011010（SRC=SPI0，SPI1，SPI2） 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010010，010011，010100，010101，011101，011110（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：捕捉比较通道 1 申请 0001：捕捉比较通道 2 申请 0010：捕捉比较通道 3 申请 0011：捕捉比较通道 4 申请 0100：TIM 触发申请 0101：TIM 比较匹配申请 0110：TIM 更新事件申请 **MSEL=011011，011100（SRC=BS16T0，BS16T1）** TIM 更新事件申请 **MSEL=011000（SRC=LPUART0）** 0000：接收可用数据申请 0001：发送缓冲器空申请 **MSEL=011111（SRC=ADC1）** ADC1 DMA 申请 **MSEL=100000（SRC=PIS）** 0000：PIS 通道 0 0001：PIS 通道 1 0010：PIS 通道 2 0011：PIS 通道 3 0100：PIS 通道 4 0101：PIS 通道 5 0110：PIS 通道 6 0111：PIS 通道 7 1000：PIS 通道 8 1001：PIS 通道 9 1010：PIS 通道 10 1011：PIS 通道 11 1100：PIS 通道 12 1101：PIS 通道 13 1110：PIS 通道 14 1111：PIS 通道 15 **MSEL=100001（SRC=TRNG）**

			TRNG DMA 申请 **MSEL=100010（SRC=QSPI）** 0000：QSPI 间接写申请 0001：QSPI 间接读申请 **MSEL=100011（SRC=USB）** 0000：USB 发送端点 1 0001：USB 发送端点 2 0010：USB 发送端点 3 0011：USB 发送端点 4 0100：USB 发送端点 5 0101：保留 0110：保留 0111：保留 1000：USB 接收端点 1 1001：USB 接收端点 2 1010：USB 接收端点 3 1011：USB 接收端点 4 1100：USB 接收端点 5 1101：保留 1110：保留 1111：保留 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1

11. 5. 3. 24 DMA通道 3 复用选择寄存器（DMA_CH3_SELCON）

DMA 通道 3 复用选择寄存器（DMA_CH3_SELCON）

偏移地址：$110C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-14	–	保留
MSEL	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO_EXTI 000010：CRYPT 000011：预留 000100：DAC0 000101：预留 000110：ADC0 000111：CRC 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：预留 010111：预留 011000：LPUART0 011001：预留 011010：SPI2 011011：BS16T0 011100：BS16T1 011101：GP16C4T0 011110：GP16C4T1

			011111：ADC1 100000：PIS 100001：TRNG 100010：QSPI 100011：USB 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	–	保留
MSIGSEL	Bit 3-0	R/W	**MSEL=000000** MSIGSEL 无效 **MSEL=000001** 0000：EXTI0 0001：EXTI1 0010：EXTI2 0011：EXTI3 0100：EXTI4 0101：EXTI5 0110：EXTI6 0111：EXTI7 1000：EXTI8 1001：EXTI9 1010：EXTI10 1011：EXTI11 1100：EXTI12 1101：EXTI13 1110：EXTI14 1111：EXTI15 **MSEL=000010（SRC=CRYPT）时** 0000：CRYPT DMA 写数据申请 0001：CRYPT DMA 读数据申请 **MSEL=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束申请 0001：DAC0 通道 1 转换结束申请 **MSEL=000110（SRC=ADC0）** ADC0 DMA 申请 **MSEL=000111（SRC=CRC）** CRC DMA 申请 **MSEL=001000，001001，001010，001011** **（SRC=UART0，UART1，UART2，UART3）** 0000：发送保持寄存器空申请 0001：接收可用数据申请 **MSEL=001100，001101（SRC=UART4，UART5）** 0000：接收可用数据申请 0001：发送保持寄存器空申请

			MSEL=001110，001111，011010（SRC=SPI0，SPI1，SPI2） 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010010，010011，010100，010101，011101，011110（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：捕捉比较通道 1 申请 0001：捕捉比较通道 2 申请 0010：捕捉比较通道 3 申请 0011：捕捉比较通道 4 申请 0100：TIM 触发申请 0101：TIM 比较匹配申请 0110：TIM 更新事件申请 **MSEL=011011，011100（SRC=BS16T0，BS16T1）** TIM 更新事件申请 **MSEL=011000（SRC=LPUART0）** 0000：接收可用数据申请 0001：发送缓冲器空申请 **MSEL=011111（SRC=ADC1）** ADC1 DMA 申请 **MSEL=100000（SRC=PIS）** 0000：PIS 通道 0 0001：PIS 通道 1 0010：PIS 通道 2 0011：PIS 通道 3 0100：PIS 通道 4 0101：PIS 通道 5 0110：PIS 通道 6 0111：PIS 通道 7 1000：PIS 通道 8 1001：PIS 通道 9 1010：PIS 通道 10 1011：PIS 通道 11 1100：PIS 通道 12 1101：PIS 通道 13 1110：PIS 通道 14 1111：PIS 通道 15 **MSEL=100001（SRC=TRNG）**

			TRNG DMA 申请 **MSEL=100010（SRC=QSPI）** 0000：QSPI 间接写申请 0001：QSPI 间接读申请 **MSEL=100011（SRC=USB）** 0000：USB 发送端点 1 0001：USB 发送端点 2 0010：USB 发送端点 3 0011：USB 发送端点 4 0100：USB 发送端点 5 0101：保留 0110：保留 0111：保留 1000：USB 接收端点 1 1001：USB 接收端点 2 1010：USB 接收端点 3 1011：USB 接收端点 4 1100：USB 接收端点 5 1101：保留 1110：保留 1111：保留 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1

11. 5. 3. 25 DMA通道 4 复用选择寄存器（DMA_CH4_SELCON）

DMA 通道 4 复用选择寄存器（DMA_CH4_SELCON）

偏移地址：1110_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-14	–	保留
MSEL	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO_EXTI 000010：CRYPT 000011：预留 000100：DAC0 000101：预留 000110：ADC0 000111：CRC 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：预留 010111：预留 011000：LPUART0 011001：预留 011010：SPI2 011011：BS16T0 011100：BS16T1 011101：GP16C4T0 011110：GP16C4T1

			011111：ADC1 100000：PIS 100001：TRNG 100010：QSPI 100011：USB 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	–	保留
MSIGSEL	Bit 3-0	R/W	**MSEL=000000** MSIGSEL 无效 **MSEL=000001** 0000：EXTI0 0001：EXTI1 0010：EXTI2 0011：EXTI3 0100：EXTI4 0101：EXTI5 0110：EXTI6 0111：EXTI7 1000：EXTI8 1001：EXTI9 1010：EXTI10 1011：EXTI11 1100：EXTI12 1101：EXTI13 1110：EXTI14 1111：EXTI15 **MSEL=000010（SRC=CRYPT）时** 0000：CRYPT DMA 写数据申请 0001：CRYPT DMA 读数据申请 **MSEL=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束申请 0001：DAC0 通道 1 转换结束申请 **MSEL=000110（SRC=ADC0）** ADC0 DMA 申请 **MSEL=000111（SRC=CRC）** CRC DMA 申请 **MSEL=001000，001001，001010，001011** **（SRC=UART0，UART1，UART2，UART3）** 0000：发送保持寄存器空申请 0001：接收可用数据申请 **MSEL=001100，001101（SRC=UART4，UART5）** 0000：接收可用数据申请 0001：发送保持寄存器空申请

			MSEL=001110，001111，011010（SRC=SPI0，SPI1，SPI2） 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010010，010011，010100，010101，011101，011110（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：捕捉比较通道 1 申请 0001：捕捉比较通道 2 申请 0010：捕捉比较通道 3 申请 0011：捕捉比较通道 4 申请 0100：TIM 触发申请 0101：TIM 比较匹配申请 0110：TIM 更新事件申请 **MSEL=011011，011100（SRC=BS16T0，BS16T1）** TIM 更新事件申请 **MSEL=011000（SRC=LPUART0）** 0000：接收可用数据申请 0001：发送缓冲器空申请 **MSEL=011111（SRC=ADC1）** ADC1 DMA 申请 **MSEL=100000（SRC=PIS）** 0000：PIS 通道 0 0001：PIS 通道 1 0010：PIS 通道 2 0011：PIS 通道 3 0100：PIS 通道 4 0101：PIS 通道 5 0110：PIS 通道 6 0111：PIS 通道 7 1000：PIS 通道 8 1001：PIS 通道 9 1010：PIS 通道 10 1011：PIS 通道 11 1100：PIS 通道 12 1101：PIS 通道 13 1110：PIS 通道 14 1111：PIS 通道 15 **MSEL=100001（SRC=TRNG）**

			TRNG DMA 申请 **MSEL=100010（SRC=QSPI）** 0000：QSPI 间接写申请 0001：QSPI 间接读申请 **MSEL=100011（SRC=USB）** 0000：USB 发送端点 1 0001：USB 发送端点 2 0010：USB 发送端点 3 0011：USB 发送端点 4 0100：USB 发送端点 5 0101：保留 0110：保留 0111：保留 1000：USB 接收端点 1 1001：USB 接收端点 2 1010：USB 接收端点 3 1011：USB 接收端点 4 1100：USB 接收端点 5 1101：保留 1110：保留 1111：保留 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1

11. 5. 3. 26 DMA通道 5 复用选择寄存器（DMA_CH5_SELCON）

DMA 通道 5 复用选择寄存器（DMA_CH5_SELCON）

偏移地址：1114_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-14	–	保留
MSEL	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO_EXTI 000010：CRYPT 000011：预留 000100：DAC0 000101：预留 000110：ADC0 000111：CRC 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：预留 010111：预留 011000：LPUART0 011001：预留 011010：SPI2 011011：BS16T0 011100：BS16T1 011101：GP16C4T0 011110：GP16C4T1

			011111：ADC1 100000：PIS 100001：TRNG 100010：QSPI 100011：USB 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	–	保留
MSIGSEL	Bit 3-0	R/W	**MSEL=000000** MSIGSEL 无效 **MSEL=000001** 0000：EXTI0 0001：EXTI1 0010：EXTI2 0011：EXTI3 0100：EXTI4 0101：EXTI5 0110：EXTI6 0111：EXTI7 1000：EXTI8 1001：EXTI9 1010：EXTI10 1011：EXTI11 1100：EXTI12 1101：EXTI13 1110：EXTI14 1111：EXTI15 **MSEL=000010（SRC=CRYPT）时** 0000：CRYPT DMA 写数据申请 0001：CRYPT DMA 读数据申请 **MSEL=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束申请 0001：DAC0 通道 1 转换结束申请 **MSEL=000110（SRC=ADC0）** ADC0 DMA 申请 **MSEL=000111（SRC=CRC）** CRC DMA 申请 **MSEL=001000，001001，001010，001011** **（SRC=UART0，UART1，UART2，UART3）** 0000：发送保持寄存器空申请 0001：接收可用数据申请 **MSEL=001100，001101（SRC=UART4，UART5）** 0000：接收可用数据申请 0001：发送保持寄存器空申请

			MSEL=001110，001111，011010（SRC=SPI0，SPI1，SPI2） 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010010，010011，010100，010101，011101，011110（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：捕捉比较通道 1 申请 0001：捕捉比较通道 2 申请 0010：捕捉比较通道 3 申请 0011：捕捉比较通道 4 申请 0100：TIM 触发申请 0101：TIM 比较匹配申请 0110：TIM 更新事件申请 **MSEL=011011，011100（SRC=BS16T0，BS16T1）** TIM 更新事件申请 **MSEL=011000（SRC=LPUART0）** 0000：接收可用数据申请 0001：发送缓冲器空申请 **MSEL=011111（SRC=ADC1）** ADC1 DMA 申请 **MSEL=100000（SRC=PIS）** 0000：PIS 通道 0 0001：PIS 通道 1 0010：PIS 通道 2 0011：PIS 通道 3 0100：PIS 通道 4 0101：PIS 通道 5 0110：PIS 通道 6 0111：PIS 通道 7 1000：PIS 通道 8 1001：PIS 通道 9 1010：PIS 通道 10 1011：PIS 通道 11 1100：PIS 通道 12 1101：PIS 通道 13 1110：PIS 通道 14 1111：PIS 通道 15 **MSEL=100001（SRC=TRNG）**

			TRNG DMA 申请 **MSEL=100010（SRC=QSPI）** 0000：QSPI 间接写申请 0001：QSPI 间接读申请 **MSEL=100011（SRC=USB）** 0000：USB 发送端点 1 0001：USB 发送端点 2 0010：USB 发送端点 3 0011：USB 发送端点 4 0100：USB 发送端点 5 0101：保留 0110：保留 0111：保留 1000：USB 接收端点 1 1001：USB 接收端点 2 1010：USB 接收端点 3 1011：USB 接收端点 4 1100：USB 接收端点 5 1101：保留 1110：保留 1111：保留 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1

11. 5. 3. 27 DMA通道 6 复用选择寄存器（DMA_CH6_SELCON）

DMA 通道 6 复用选择寄存器（DMA_CH6_SELCON）

偏移地址：1118_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-14	–	保留
MSEL	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO_EXTI 000010：CRYPT 000011：预留 000100：DAC0 000101：预留 000110：ADC0 000111：CRC 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：预留 010111：预留 011000：LPUART0 011001：预留 011010：SPI2 011011：BS16T0 011100：BS16T1 011101：GP16C4T0 011110：GP16C4T1

			011111：ADC1 100000：PIS 100001：TRNG 100010：QSPI 100011：USB 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	–	保留
MSIGSEL	Bit 3-0	R/W	**MSEL=000000** MSIGSEL 无效 **MSEL=000001** 0000：EXTI0 0001：EXTI1 0010：EXTI2 0011：EXTI3 0100：EXTI4 0101：EXTI5 0110：EXTI6 0111：EXTI7 1000：EXTI8 1001：EXTI9 1010：EXTI10 1011：EXTI11 1100：EXTI12 1101：EXTI13 1110：EXTI14 1111：EXTI15 **MSEL=000010（SRC=CRYPT）时** 0000：CRYPT DMA 写数据申请 0001：CRYPT DMA 读数据申请 **MSEL=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束申请 0001：DAC0 通道 1 转换结束申请 **MSEL=000110（SRC=ADC0）** ADC0 DMA 申请 **MSEL=000111（SRC=CRC）** CRC DMA 申请 **MSEL=001000，001001，001010，001011** **（SRC=UART0，UART1，UART2，UART3）** 0000：发送保持寄存器空申请 0001：接收可用数据申请 **MSEL=001100，001101（SRC=UART4，UART5）** 0000：接收可用数据申请 0001：发送保持寄存器空申请

			MSEL=001110，001111，011010（SRC=SPI0，SPI1，SPI2） 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010010，010011，010100，010101，011101，011110（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：捕捉比较通道 1 申请 0001：捕捉比较通道 2 申请 0010：捕捉比较通道 3 申请 0011：捕捉比较通道 4 申请 0100：TIM 触发申请 0101：TIM 比较匹配申请 0110：TIM 更新事件申请 **MSEL=011011，011100（SRC=BS16T0，BS16T1）** TIM 更新事件申请 **MSEL=011000（SRC=LPUART0）** 0000：接收可用数据申请 0001：发送缓冲器空申请 **MSEL=011111（SRC=ADC1）** ADC1 DMA 申请 **MSEL=100000（SRC=PIS）** 0000：PIS 通道 0 0001：PIS 通道 1 0010：PIS 通道 2 0011：PIS 通道 3 0100：PIS 通道 4 0101：PIS 通道 5 0110：PIS 通道 6 0111：PIS 通道 7 1000：PIS 通道 8 1001：PIS 通道 9 1010：PIS 通道 10 1011：PIS 通道 11 1100：PIS 通道 12 1101：PIS 通道 13 1110：PIS 通道 14 1111：PIS 通道 15 **MSEL=100001（SRC=TRNG）**

			TRNG DMA 申请 **MSEL=100010（SRC=QSPI）** 0000：QSPI 间接写申请 0001：QSPI 间接读申请 **MSEL=100011（SRC=USB）** 0000：USB 发送端点 1 0001：USB 发送端点 2 0010：USB 发送端点 3 0011：USB 发送端点 4 0100：USB 发送端点 5 0101：保留 0110：保留 0111：保留 1000：USB 接收端点 1 1001：USB 接收端点 2 1010：USB 接收端点 3 1011：USB 接收端点 4 1100：USB 接收端点 5 1101：保留 1110：保留 1111：保留 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1

11. 5. 3. 28 DMA通道 7 复用选择寄存器（DMA_CH7_SELCON）

DMA 通道 7 复用选择寄存器（DMA_CH7_SELCON）

偏移地址：$111C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-14	–	保留
MSEL	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO_EXTI 000010：CRYPT 000011：预留 000100：DAC0 000101：预留 000110：ADC0 000111：CRC 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：预留 010111：预留 011000：LPUART0 011001：预留 011010：SPI2 011011：BS16T0 011100：BS16T1 011101：GP16C4T0 011110：GP16C4T1

			011111：ADC1 100000：PIS 100001：TRNG 100010：QSPI 100011：USB 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	–	保留
MSIGSEL	Bit 3-0	R/W	**MSEL=000000** MSIGSEL 无效 **MSEL=000001** 0000：EXTI0 0001：EXTI1 0010：EXTI2 0011：EXTI3 0100：EXTI4 0101：EXTI5 0110：EXTI6 0111：EXTI7 1000：EXTI8 1001：EXTI9 1010：EXTI10 1011：EXTI11 1100：EXTI12 1101：EXTI13 1110：EXTI14 1111：EXTI15 **MSEL=000010（SRC=CRYPT）时** 0000：CRYPT DMA 写数据申请 0001：CRYPT DMA 读数据申请 **MSEL=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束申请 0001：DAC0 通道 1 转换结束申请 **MSEL=000110（SRC=ADC0）** ADC0 DMA 申请 **MSEL=000111（SRC=CRC）** CRC DMA 申请 **MSEL=001000，001001，001010，001011** **（SRC=UART0，UART1，UART2，UART3）** 0000：发送保持寄存器空申请 0001：接收可用数据申请 **MSEL=001100，001101（SRC=UART4，UART5）** 0000：接收可用数据申请 0001：发送保持寄存器空申请

			MSEL=001110，001111，011010（SRC=SPI0，SPI1，SPI2） 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010010，010011，010100，010101，011101，011110（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：捕捉比较通道 1 申请 0001：捕捉比较通道 2 申请 0010：捕捉比较通道 3 申请 0011：捕捉比较通道 4 申请 0100：TIM 触发申请 0101：TIM 比较匹配申请 0110：TIM 更新事件申请 **MSEL=011011，011100（SRC=BS16T0，BS16T1）** TIM 更新事件申请 **MSEL=011000（SRC=LPUART0）** 0000：接收可用数据申请 0001：发送缓冲器空申请 **MSEL=011111（SRC=ADC1）** ADC1 DMA 申请 **MSEL=100000（SRC=PIS）** 0000：PIS 通道 0 0001：PIS 通道 1 0010：PIS 通道 2 0011：PIS 通道 3 0100：PIS 通道 4 0101：PIS 通道 5 0110：PIS 通道 6 0111：PIS 通道 7 1000：PIS 通道 8 1001：PIS 通道 9 1010：PIS 通道 10 1011：PIS 通道 11 1100：PIS 通道 12 1101：PIS 通道 13 1110：PIS 通道 14 1111：PIS 通道 15 **MSEL=100001（SRC=TRNG）**

			TRNG DMA 申请 **MSEL=100010（SRC=QSPI）** 0000：QSPI 间接写申请 0001：QSPI 间接读申请 **MSEL=100011（SRC=USB）** 0000：USB 发送端点 1 0001：USB 发送端点 2 0010：USB 发送端点 3 0011：USB 发送端点 4 0100：USB 发送端点 5 0101：保留 0110：保留 0111：保留 1000：USB 接收端点 1 1001：USB 接收端点 2 1010：USB 接收端点 3 1011：USB 接收端点 4 1100：USB 接收端点 5 1101：保留 1110：保留 1111：保留 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1

11. 5. 3. 29 DMA通道 8 复用选择寄存器（DMA_CH8_SELCON）

DMA 通道 8 复用选择寄存器（DMA_CH8_SELCON）

偏移地址：1120_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-14	–	保留
MSEL	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO_EXTI 000010：CRYPT 000011：预留 000100：DAC0 000101：预留 000110：ADC0 000111：CRC 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：预留 010111：预留 011000：LPUART0 011001：预留 011010：SPI2 011011：BS16T0 011100：BS16T1 011101：GP16C4T0 011110：GP16C4T1

			011111：ADC1 100000：PIS 100001：TRNG 100010：QSPI 100011：USB 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	–	保留
MSIGSEL	Bit 3-0	R/W	**MSEL=000000** MSIGSEL 无效 **MSEL=000001** 0000：EXTI0 0001：EXTI1 0010：EXTI2 0011：EXTI3 0100：EXTI4 0101：EXTI5 0110：EXTI6 0111：EXTI7 1000：EXTI8 1001：EXTI9 1010：EXTI10 1011：EXTI11 1100：EXTI12 1101：EXTI13 1110：EXTI14 1111：EXTI15 **MSEL=000010（SRC=CRYPT）时** 0000：CRYPT DMA 写数据申请 0001：CRYPT DMA 读数据申请 **MSEL=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束申请 0001：DAC0 通道 1 转换结束申请 **MSEL=000110（SRC=ADC0）** ADC0 DMA 申请 **MSEL=000111（SRC=CRC）** CRC DMA 申请 **MSEL=001000，001001，001010，001011** **（SRC=UART0，UART1，UART2，UART3）** 0000：发送保持寄存器空申请 0001：接收可用数据申请 **MSEL=001100，001101（SRC=UART4，UART5）** 0000：接收可用数据申请 0001：发送保持寄存器空申请

			MSEL=001110，001111，011010（SRC=SPI0，SPI1，SPI2） 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010010，010011，010100，010101，011101，011110（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：捕捉比较通道 1 申请 0001：捕捉比较通道 2 申请 0010：捕捉比较通道 3 申请 0011：捕捉比较通道 4 申请 0100：TIM 触发申请 0101：TIM 比较匹配申请 0110：TIM 更新事件申请 **MSEL=011011，011100（SRC=BS16T0，BS16T1）** TIM 更新事件申请 **MSEL=011000（SRC=LPUART0）** 0000：接收可用数据申请 0001：发送缓冲器空申请 **MSEL=011111（SRC=ADC1）** ADC1 DMA 申请 **MSEL=100000（SRC=PIS）** 0000：PIS 通道 0 0001：PIS 通道 1 0010：PIS 通道 2 0011：PIS 通道 3 0100：PIS 通道 4 0101：PIS 通道 5 0110：PIS 通道 6 0111：PIS 通道 7 1000：PIS 通道 8 1001：PIS 通道 9 1010：PIS 通道 10 1011：PIS 通道 11 1100：PIS 通道 12 1101：PIS 通道 13 1110：PIS 通道 14 1111：PIS 通道 15 **MSEL=100001（SRC=TRNG）**

			TRNG DMA 申请 **MSEL=100010（SRC=QSPI）** 0000：QSPI 间接写申请 0001：QSPI 间接读申请 **MSEL=100011（SRC=USB）** 0000：USB 发送端点 1 0001：USB 发送端点 2 0010：USB 发送端点 3 0011：USB 发送端点 4 0100：USB 发送端点 5 0101：保留 0110：保留 0111：保留 1000：USB 接收端点 1 1001：USB 接收端点 2 1010：USB 接收端点 3 1011：USB 接收端点 4 1100：USB 接收端点 5 1101：保留 1110：保留 1111：保留 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1

11. 5. 3. 30 DMA通道 9 复用选择寄存器（DMA_CH9_SELCON）

DMA 通道 9 复用选择寄存器（DMA_CH9_SELCON）

偏移地址：1124_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-14	–	保留
MSEL	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO_EXTI 000010：CRYPT 000011：预留 000100：DAC0 000101：预留 000110：ADC0 000111：CRC 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：预留 010111：预留 011000：LPUART0 011001：预留 011010：SPI2 011011：BS16T0 011100：BS16T1 011101：GP16C4T0 011110：GP16C4T1

			011111：ADC1 100000：PIS 100001：TRNG 100010：QSPI 100011：USB 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	–	保留
MSIGSEL	Bit 3-0	R/W	**MSEL=000000** MSIGSEL 无效 **MSEL=000001** 0000：EXTI0 0001：EXTI1 0010：EXTI2 0011：EXTI3 0100：EXTI4 0101：EXTI5 0110：EXTI6 0111：EXTI7 1000：EXTI8 1001：EXTI9 1010：EXTI10 1011：EXTI11 1100：EXTI12 1101：EXTI13 1110：EXTI14 1111：EXTI15 **MSEL=000010（SRC=CRYPT）时** 0000：CRYPT DMA 写数据申请 0001：CRYPT DMA 读数据申请 **MSEL=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束申请 0001：DAC0 通道 1 转换结束申请 **MSEL=000110（SRC=ADC0）** ADC0 DMA 申请 **MSEL=000111（SRC=CRC）** CRC DMA 申请 **MSEL=001000，001001，001010，001011** **（SRC=UART0，UART1，UART2，UART3）** 0000：发送保持寄存器空申请 0001：接收可用数据申请 **MSEL=001100，001101（SRC=UART4，UART5）** 0000：接收可用数据申请 0001：发送保持寄存器空申请

			MSEL=001110，001111，011010（SRC=SPI0，SPI1，SPI2） 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010010，010011，010100，010101，011101，011110（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：捕捉比较通道 1 申请 0001：捕捉比较通道 2 申请 0010：捕捉比较通道 3 申请 0011：捕捉比较通道 4 申请 0100：TIM 触发申请 0101：TIM 比较匹配申请 0110：TIM 更新事件申请 **MSEL=011011，011100（SRC=BS16T0，BS16T1）** TIM 更新事件申请 **MSEL=011000（SRC=LPUART0）** 0000：接收可用数据申请 0001：发送缓冲器空申请 **MSEL=011111（SRC=ADC1）** ADC1 DMA 申请 **MSEL=100000（SRC=PIS）** 0000：PIS 通道 0 0001：PIS 通道 1 0010：PIS 通道 2 0011：PIS 通道 3 0100：PIS 通道 4 0101：PIS 通道 5 0110：PIS 通道 6 0111：PIS 通道 7 1000：PIS 通道 8 1001：PIS 通道 9 1010：PIS 通道 10 1011：PIS 通道 11 1100：PIS 通道 12 1101：PIS 通道 13 1110：PIS 通道 14 1111：PIS 通道 15 **MSEL=100001（SRC=TRNG）**

			TRNG DMA 申请 **MSEL=100010（SRC=QSPI）** 0000：QSPI 间接写申请 0001：QSPI 间接读申请 **MSEL=100011（SRC=USB）** 0000：USB 发送端点 1 0001：USB 发送端点 2 0010：USB 发送端点 3 0011：USB 发送端点 4 0100：USB 发送端点 5 0101：保留 0110：保留 0111：保留 1000：USB 接收端点 1 1001：USB 接收端点 2 1010：USB 接收端点 3 1011：USB 接收端点 4 1100：USB 接收端点 5 1101：保留 1110：保留 1111：保留 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1

11. 5. 3. 31 DMA通道 10 复用选择寄存器（DMA_CH10_SELCON）

DMA 通道 10 复用选择寄存器（DMA_CH10_SELCON）

偏移地址：1128_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-14	–	保留
MSEL	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO_EXTI 000010：CRYPT 000011：预留 000100：DAC0 000101：预留 000110：ADC0 000111：CRC 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：预留 010111：预留 011000：LPUART0 011001：预留 011010：SPI2 011011：BS16T0 011100：BS16T1 011101：GP16C4T0 011110：GP16C4T1

			011111：ADC1 100000：PIS 100001：TRNG 100010：QSPI 100011：USB 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	–	保留
MSIGSEL	Bit 3-0	R/W	**MSEL=000000** MSIGSEL 无效 **MSEL=000001** 0000：EXTI0 0001：EXTI1 0010：EXTI2 0011：EXTI3 0100：EXTI4 0101：EXTI5 0110：EXTI6 0111：EXTI7 1000：EXTI8 1001：EXTI9 1010：EXTI10 1011：EXTI11 1100：EXTI12 1101：EXTI13 1110：EXTI14 1111：EXTI15 **MSEL=000010（SRC=CRYPT）时** 0000：CRYPT DMA 写数据申请 0001：CRYPT DMA 读数据申请 **MSEL=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束申请 0001：DAC0 通道 1 转换结束申请 **MSEL=000110（SRC=ADC0）** ADC0 DMA 申请 **MSEL=000111（SRC=CRC）** CRC DMA 申请 **MSEL=001000，001001，001010，001011** **（SRC=UART0，UART1，UART2，UART3）** 0000：发送保持寄存器空申请 0001：接收可用数据申请 **MSEL=001100，001101（SRC=UART4，UART5）** 0000：接收可用数据申请 0001：发送保持寄存器空申请

			MSEL=001110，001111，011010（SRC=SPI0，SPI1，SPI2） 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010010，010011，010100，010101，011101，011110（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：捕捉比较通道 1 申请 0001：捕捉比较通道 2 申请 0010：捕捉比较通道 3 申请 0011：捕捉比较通道 4 申请 0100：TIM 触发申请 0101：TIM 比较匹配申请 0110：TIM 更新事件申请 **MSEL=011011，011100（SRC=BS16T0，BS16T1）** TIM 更新事件申请 **MSEL=011000（SRC=LPUART0）** 0000：接收可用数据申请 0001：发送缓冲器空申请 **MSEL=011111（SRC=ADC1）** ADC1 DMA 申请 **MSEL=100000（SRC=PIS）** 0000：PIS 通道 0 0001：PIS 通道 1 0010：PIS 通道 2 0011：PIS 通道 3 0100：PIS 通道 4 0101：PIS 通道 5 0110：PIS 通道 6 0111：PIS 通道 7 1000：PIS 通道 8 1001：PIS 通道 9 1010：PIS 通道 10 1011：PIS 通道 11 1100：PIS 通道 12 1101：PIS 通道 13 1110：PIS 通道 14 1111：PIS 通道 15 **MSEL=100001（SRC=TRNG）**

			TRNG DMA 申请 **MSEL=100010（SRC=QSPI）** 0000：QSPI 间接写申请 0001：QSPI 间接读申请 **MSEL=100011（SRC=USB）** 0000：USB 发送端点 1 0001：USB 发送端点 2 0010：USB 发送端点 3 0011：USB 发送端点 4 0100：USB 发送端点 5 0101：保留 0110：保留 0111：保留 1000：USB 接收端点 1 1001：USB 接收端点 2 1010：USB 接收端点 3 1011：USB 接收端点 4 1100：USB 接收端点 5 1101：保留 1110：保留 1111：保留 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1

11. 5. 3. 32 DMA通道 11 复用选择寄存器（DMA_CH11_SELCON）

DMA 通道 11 复用选择寄存器（DMA_CH11_SELCON）

偏移地址：$112C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-14	–	保留
MSEL	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO_EXTI 000010：CRYPT 000011：预留 000100：DAC0 000101：预留 000110：ADC0 000111：CRC 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：预留 010111：预留 011000：LPUART0 011001：预留 011010：SPI2 011011：BS16T0 011100：BS16T1 011101：GP16C4T0 011110：GP16C4T1

			011111：ADC1 100000：PIS 100001：TRNG 100010：QSPI 100011：USB 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	–	保留
MSIGSEL	Bit 3-0	R/W	**MSEL=000000** MSIGSEL 无效 **MSEL=000001** 0000：EXTI0 0001：EXTI1 0010：EXTI2 0011：EXTI3 0100：EXTI4 0101：EXTI5 0110：EXTI6 0111：EXTI7 1000：EXTI8 1001：EXTI9 1010：EXTI10 1011：EXTI11 1100：EXTI12 1101：EXTI13 1110：EXTI14 1111：EXTI15 **MSEL=000010（SRC=CRYPT）时** 0000：CRYPT DMA 写数据申请 0001：CRYPT DMA 读数据申请 **MSEL=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束申请 0001：DAC0 通道 1 转换结束申请 **MSEL=000110（SRC=ADC0）** ADC0 DMA 申请 **MSEL=000111（SRC=CRC）** CRC DMA 申请 **MSEL=001000，001001，001010，001011** **（SRC=UART0，UART1，UART2，UART3）** 0000：发送保持寄存器空申请 0001：接收可用数据申请 **MSEL=001100，001101（SRC=UART4，UART5）** 0000：接收可用数据申请 0001：发送保持寄存器空申请

			MSEL=001110，001111，011010（SRC=SPI0，SPI1，SPI2） 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空申请 0001：发送缓冲器空申请 **MSEL=010010，010011，010100，010101，011101，011110（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：捕捉比较通道 1 申请 0001：捕捉比较通道 2 申请 0010：捕捉比较通道 3 申请 0011：捕捉比较通道 4 申请 0100：TIM 触发申请 0101：TIM 比较匹配申请 0110：TIM 更新事件申请 **MSEL=011011，011100（SRC=BS16T0，BS16T1）** TIM 更新事件申请 **MSEL=011000（SRC=LPUART0）** 0000：接收可用数据申请 0001：发送缓冲器空申请 **MSEL=011111（SRC=ADC1）** ADC1 DMA 申请 **MSEL=100000（SRC=PIS）** 0000：PIS 通道 0 0001：PIS 通道 1 0010：PIS 通道 2 0011：PIS 通道 3 0100：PIS 通道 4 0101：PIS 通道 5 0110：PIS 通道 6 0111：PIS 通道 7 1000：PIS 通道 8 1001：PIS 通道 9 1010：PIS 通道 10 1011：PIS 通道 11 1100：PIS 通道 12 1101：PIS 通道 13 1110：PIS 通道 14 1111：PIS 通道 15 **MSEL=100001（SRC=TRNG）**

			TRNG DMA 申请 **MSEL=100010（SRC=QSPI）** 0000：QSPI 间接写申请 0001：QSPI 间接读申请 **MSEL=100011（SRC=USB）** 0000：USB 发送端点 1 0001：USB 发送端点 2 0010：USB 发送端点 3 0011：USB 发送端点 4 0100：USB 发送端点 5 0101：保留 0110：保留 0111：保留 1000：USB 接收端点 1 1001：USB 接收端点 2 1010：USB 接收端点 3 1011：USB 接收端点 4 1100：USB 接收端点 5 1101：保留 1110：保留 1111：保留 注：ES32F36xx 系列不支持 LPUART， ES32F336x 和 ES32F3936 系列不支持 ADC1

第12章 外设互联（PIS）

12. 1 概述

PIS（Peripheral Interaction System）在微控制器中作为外设互联的桥接口使用，利用 PIS 可实现外设之间的相互触发，控制及自动化工作，提高系统的实时性和快速响应能力，可避免占用过多的 CPU 资源并简化软件工作，为各种应用提供便捷。

12. 2 特性

- 最多支持 16 个 PIS 通道选择
- 支持同步和异步通道选择
- 支持信号有效边缘选择
- 支持通道输出到管脚
- UART 输出调制

12. 3 结构框图

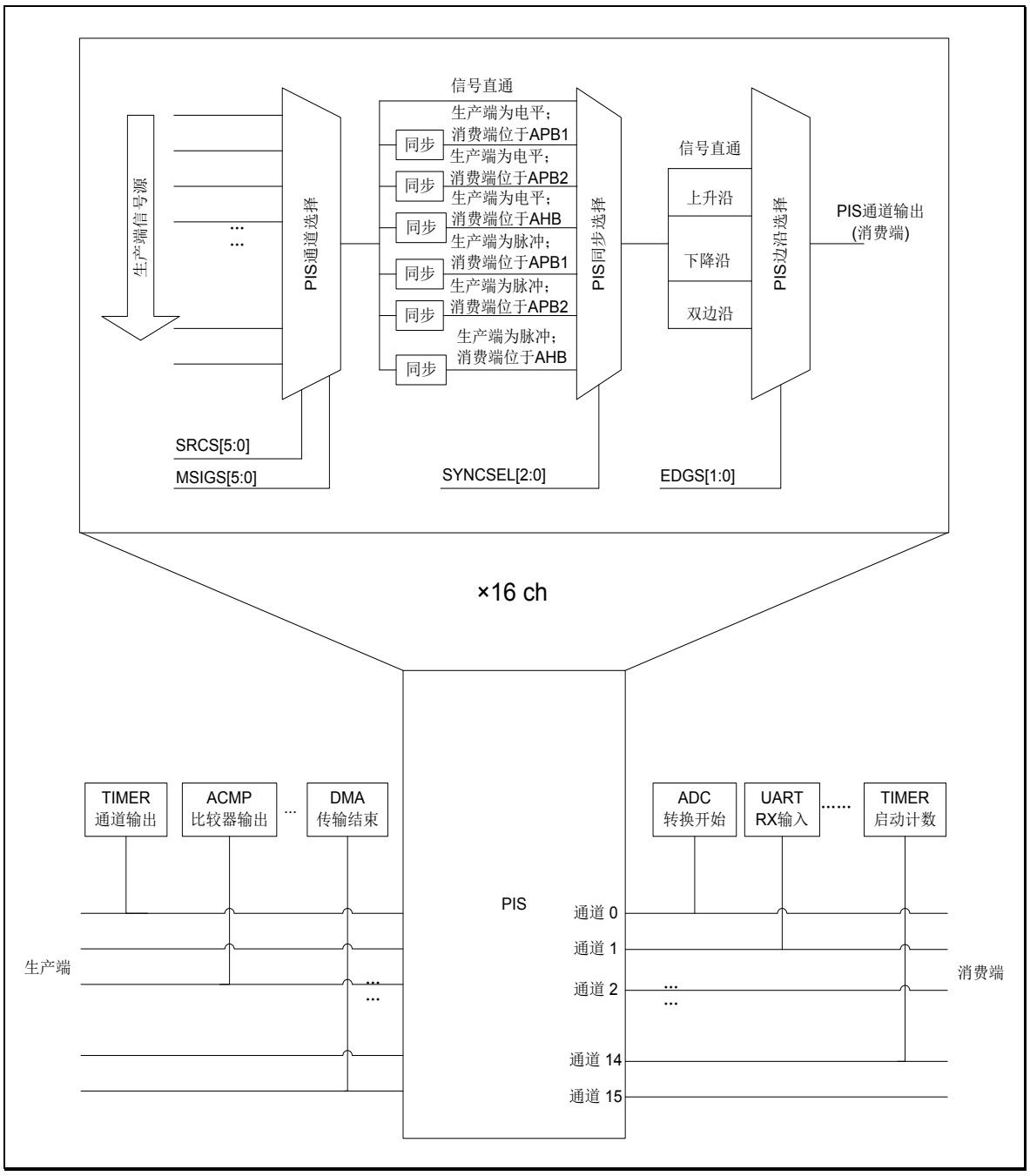


图 12-1 PIS 结构框图

12. 4 功能描述

外设互联可支持 16 个通道资源，每个通道均可对生产端信号进行多路复用。针对不同应用可灵活配置。

12. 4. 1 生产端信号

外设互联的生产端信号如下表所示：

（注意 ES32F36xx 不支持 LP16T 和 LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1;只有 ES32F36xx 可支持高级定时器，详细产品资源选型请参考数据手册）

	生产端	输出形式	异步支持	位置（APB1，APB2 或 AHB）
GPIO	PB0~PB15 输入	电平	是	AHB 外设
UART0/1/2/3	发送空状态中断	脉冲	—	APB1 外设
	接收数据中断	脉冲	—	
	IrDA 解码器输出	电平	—	
	RTS 输出	电平	—	
	TX 输出	电平	—	
UART4/5	接收数据中断	脉冲	—	APB1 外设
	发送空状态中断	脉冲	—	
	TX 输出	电平	—	
SPI	接收缓冲器非空	脉冲	—	APB1 外设
	发送缓冲器空	脉冲	—	
	片选输出	电平	—	
I2C	接收缓冲器非空	电平	—	APB1 外设
	发送缓冲器空	电平	—	
TIMER	更新事件	脉冲	—	APB1 外设
	触发事件	脉冲	—	
	输入捕获	脉冲	—	
	输出比较	脉冲	—	
RTC	亚秒、秒、分、时、日、月、年	脉冲	是	APB2 外设
	闹钟 A 和闹钟 B	脉冲	是	
ADC	标准转换组转换结束	脉冲	—	APB2 外设
	插入组转换结束	脉冲	—	
DAC	通道 0/1 转换结束	脉冲	—	APB2 外设
ACMP	比较器输出	电平	是	APB2 外设
LVD	比较器输出	电平	是	—
LP16T	更新事件	脉冲	是	APB2 外设
LPUART	接收缓冲器非空	脉冲	是	APB2 外设
	发送缓冲器空	脉冲	是	
DMA	DMA 通道完成	脉冲	是	AHB 外设

表 12-1 生产端信号

12. 4. 2 消费端信号

外设互联的消费端信号如下表所示：

	消费端	输入形式	异步支持	位置（APB1，APB2 或 AHB）
UART	RX 输入	电平	—	APB1 外设
	IrDA 编码器输入	电平	—	
SPI	RX 输入	电平	—	APB1 外设
	CLK 输入	电平	—	
TIMER	启动	脉冲	—	APB1 外设
	停止	脉冲	—	
	清零	脉冲	—	
	比较捕捉通道输入	电平或脉冲	—	
	通道输出清除	电平或脉冲	—	
	刹车输入	电平	—	
ADC	启动标准转换组转换	脉冲	—	APB2 外设
	启动插入组转换	脉冲	—	
DAC	触发转换	电平	是	APB2 外设
LP16T	启动	脉冲	是	APB2 外设
LPUART	RX 输入	电平	是	APB2 外设
DMA	DMA 通道请求	脉冲	是	AHB 外设

表 12-2 消费端信号

消费端信号的 PIS 通道分配如下表所示：

	消费端	源通道	备注
UART0	RX 输入或 IrDA 编码器输入	PIS 通道 9	由 PIS_TAR_CON1.UART0_RXD_SEL 设定
UART1	RX 输入或 IrDA 编码器输入	PIS 通道 10	由 PIS_TAR_CON1.UART1_RXD_SEL 设定
UART2	RX 输入或 IrDA 编码器输入	PIS 通道 11	由 PIS_TAR_CON1.UART2_RXD_SEL 设定
UART3	RX 输入或 IrDA 编码器输入	PIS 通道 12	由 PIS_TAR_CON1.UART3_RXD_SEL 设定
UART4	RX 输入	PIS 通道 13	由 PIS_TAR_CON1.UART4_RXD_SEL 设定
UART5	RX 输入	PIS 通道 14	由 PIS_TAR_CON1.UART5_RXD_SEL 设定
SPI0	RX 输入	PIS 通道 5	由 PIS_TAR_CON1.SPI0_RX_SEL 设定
	CLK 输入	PIS 通道 6	由 PIS_TAR_CON1.SPI0_CLK_SEL 设定
SPI1	RX 输入	PIS 通道 7	由 PIS_TAR_CON1.SPI1_RX_SEL 设定
	CLK 输入	PIS 通道 8	由 PIS_TAR_CON1.SPI1_CLK_SEL 设定
AD16C4T0	ITR0	PIS 通道	—

	消费端	源通道	备注
		12	
	ITR1	PIS 通道 13	—
	ITR2	PIS 通道 14	—
	ITR3	PIS 通道 15	—
	捕捉通道 1	PIS 通道 1	由 PIS_TAR_CON0.AD16C4T0_CH1IN_SEL 设定
	捕捉通道 2	PIS 通道 2	由 PIS_TAR_CON0.AD16C4T0_CH2IN_SEL 设定
	捕捉通道 3	PIS 通道 3	由 PIS_TAR_CON0.AD16C4T0_CH3IN_SEL 设定
	捕捉通道 4	PIS 通道 4	由 PIS_TAR_CON0.AD16C4T0_CH4IN_SEL 设定
	刹车输入	PIS 通道 0	由 PIS_TAR_CON0.AD16C4T0_BRKIN_SEL 设定 注意只有高级定时器可支持刹车输入，普通定时器不支持。请结合产品数据手册资源确定该功能是否能被有效支持。
	通道输出清除源 0	PIS 通道 0	—
	通道输出清除源 1	PIS 通道 1	—
	通道输出清除源 2	PIS 通道 2	—
	通道输出清除源 3	PIS 通道 3	—
AD16C4T1	ITR0	PIS 通道 12	—
	ITR1	PIS 通道 13	—
	ITR2	PIS 通道 14	—
	ITR3	PIS 通道 15	—
	捕捉通道 1	PIS 通道 1	由 PIS_TAR_CON0.AD16C4T1_CH1IN_SEL 设定
	捕捉通道 2	PIS 通道 2	由 PIS_TAR_CON0.AD16C4T1_CH2IN_SEL 设定
	捕捉通道 3	PIS 通道 3	由 PIS_TAR_CON0.AD16C4T1_CH3IN_SEL 设定
	捕捉通道 4	PIS 通道 4	由 PIS_TAR_CON0.AD16C4T1_CH4IN_SEL 设定
	刹车输入	PIS 通道 0	由 PIS_TAR_CON0.AD16C4T1_BRKIN_SEL 设定

	消费端	源通道	备注
			注意只有高级定时器可支持刹车输入，普通定时器不支持。请结合产品数据手册资源确定该功能是否能被有效支持。
	通道输出清除源 0	PIS 通道 0	—
	通道输出清除源 1	PIS 通道 1	—
	通道输出清除源 2	PIS 通道 2	—
	通道输出清除源 3	PIS 通道 3	—
GP32C4T0	ITR0	PIS 通道 12	—
	ITR1	PIS 通道 13	—
	ITR2	PIS 通道 14	—
	ITR3	PIS 通道 15	—
	捕捉通道 1	PIS 通道 5	由 PIS_TAR_CON0.GP32C4T0_CH1IN_SEL 设定
	捕捉通道 2	PIS 通道 6	由 PIS_TAR_CON0.GP32C4T0_CH2IN_SEL 设定
	捕捉通道 3	PIS 通道 7	由 PIS_TAR_CON0.GP32C4T0_CH3IN_SEL 设定
	捕捉通道 4	PIS 通道 8	由 PIS_TAR_CON0.GP32C4T0_CH4IN_SEL 设定
	通道输出清除源 0	PIS 通道 0	—
	通道输出清除源 1	PIS 通道 1	—
	通道输出清除源 2	PIS 通道 2	—
	通道输出清除源 3	PIS 通道 3	—
GP32C4T1	ITR0	PIS 通道 12	—
	ITR1	PIS 通道 13	—
	ITR2	PIS 通道 14	—
	ITR3	PIS 通道 15	—
	捕捉通道 1	PIS 通道 5	由 PIS_TAR_CON0.GP32C4T1_CH1IN_SEL 设定
	捕捉通道 2	PIS 通道 6	由 PIS_TAR_CON0.GP32C4T1_CH2IN_SEL 设定
	捕捉通道 3	PIS 通道 7	由 PIS_TAR_CON0.GP32C4T1_CH3IN_SEL 设定
	捕捉通道 4	PIS 通道 8	由 PIS_TAR_CON0.GP32C4T1_CH4IN_SEL

	消费端	源通道	备注
			设定
	通道输出清除源 0	PIS 通道 0	—
	通道输出清除源 1	PIS 通道 1	—
	通道输出清除源 2	PIS 通道 2	—
	通道输出清除源 3	PIS 通道 3	—
ADC0	启动标准转换组转换	PIS 通道 6	—
	启动插入组转换	PIS 通道 7	—
ADC1	启动标准转换组转换	PIS 通道 0	—
	启动插入组转换	PIS 通道 1	—
DAC	启动通道 0/1 转换	PIS 通道 0	—
		PIS 通道 1	—
		PIS 通道 2	—
		PIS 通道 3	—
		PIS 通道 4	—
		PIS 通道 5	—
		PIS 通道 6	—
		PIS 通道 7	—
		PIS 通道 8	—
		PIS 通道 9	—
		PIS 通道 10	—
		PIS 通道 11	—
LP16T0	ext_trig0	PIS 通道 0	—
	ext_trig1	PIS 通道 1	—
	ext_trig2	PIS 通道 2	—
	ext_trig3	PIS 通道 3	—
	ext_trig4	PIS 通道 4	—
	ext_trig5	PIS 通道 5	—
	ext_trig6	PIS 通道 6	—
	ext_trig7	PIS 通道 7	—
LPUART	RX 输入	PIS 通道 15	由 PIS_TAR_CON1.LPUART0_RXD_SEL 设定
DMA	DMA 申请	PIS 通道 7	—

表 12-3 消费端的 PIS 通道分配

12. 4. 3 PIS通道选择

PIS 的源端定义为生产端信号，PIS 的输出信号用于消费端。消费端可根据应用需要，来选择合适的生产端信号，并通过选择同步路径以保证正确采样到生产端信号。PIS 的生产端信号选择由 PIS 通道控制寄存器（PIS_CHx_CON，x=0~15）配置。PIS 通道控制寄存器的 PIS_CHx_CON.SRCS 位（x=0~15）用来选择生产端模块，配置

PIS_CHx_CON.MSIGS 位（x=0~15）则可从生产端模块的多路信号中选择一路作为生产端信号。

以下从几种情况来举例说明 PIS 的配置。

12.4.3.1 同一时钟域互联

以 LP16T0 和 ADC 为例，在 LP16T0 使用 PCLK2 作为计数时钟的情况下，产生与 PCLK2 同步的更新事件去触发 ADC 转换动作。可参照如下配置：

1. 通过上表可知 ADC 标准转换组转换的启动信号为 PIS 通道 6。
2. 设定寄存器 PIS_CH6_CON.SRCS 为 010111 选择 LP16T0 为生产端模块。
3. 设定寄存器 PIS_CH6_CON.MSIGS 为 0000，选择 LP16T0 的同步更新事件（PCLK2 同步）作为生产端信号。
4. 设定寄存器 PIS_CH6_CON.SYNCSEL 为 000（信号直通），并将 PIS_CH6_CON.EDGS 设为 00（不输出边沿），此时 TSCKS 可任意设定。
5. 配置 ADC 选择外部触发方式进行标准转换组转换。
6. 配置 LP16T0 进行计数，产生更新事件后将触发 ADC 开始 AD 转换。

对采用异步时钟（非 PCLK1，PCLK2 等）工作的两个模块之间相互触发，也可参照以上流程进行配置。

12.4.3.2 APB1 和APB2 外设之间互联

以 AD16C4T0（位于 APB1）和 ADC（位于 APB2）为例，AD16C4T0 使用 PCLK1 作为计数时钟，产生与 PCLK1 同步的更新事件去触发 ADC 转换动作。可参照如下配置：

1. 通过表 12-3 消费端的 PIS 通道分配可知 ADC 标准转换组转换的启动信号为 PIS 通道 6。
2. 设定寄存器 PIS_CH6_CON.SRCS 为 010010 选择 AD16C4T0 为生产端模块。
3. 设定寄存器 PIS_CH6_CON.MSIGS 为 0000，选择 AD16C4T0 的同步更新事件（PCLK1 同步）作为生产端信号。
4. 设定寄存器 PIS_CH6_CON.SYNCSEL 为 101（生产端为脉冲信号，消费端位于 APB2 时钟域），并将 PIS_CH6_CON.EDGS 设为 00（不输出边沿），此时 TSCKS 可任意设定。
5. 配置 ADC 选择外部触发方式进行标准转换组转换。
6. 配置 AD16C4T0 进行计数，产生更新事件后将触发 ADC 开始 AD 转换。

12.4.3.3 生产端信号为异步信号

以 RTC 和 ADC（位于 APB2）为例，RTC 使用 LOSC 作为计数时钟，产生 32.768KHz 的事件脉冲去触发 ADC 转换动作。可参照如下配置：

1. 通过上表 12-3 消费端的 PIS 通道分配可知 ADC 标准转换组转换的启动信号为 PIS 通道 6。

2. 设定寄存器 PIS_CH6_CON.SRCS 为 010110 选择 RTC 为生产端模块。
3. 设定寄存器 PIS_CH6_CON.MSIGS 为 0000，选择 RTC 的亚秒、秒、分、时、日、月、年事件脉冲作为生产端信号。
4. 设定寄存器 PIS_CH6_CON.SYNCSEL 为 010（生产端为电平信号，消费端位于 APB2 时钟域），并将 PIS_CH6_CON.EDGS 设为 01（上升沿），此时 TSCKS 可任意设定。
5. 配置 ADC 选择外部触发方式进行标准转换组转换。
6. 配置 RTC 进行计数，产生事件脉冲后触发 ADC 开始 AD 转换。

12. 4. 4 UART输出调制

UART 的输出调制功能利用定时器 PWM 波或 BUZ 信号对 UART 的 TX 调制后发送到端口。

调制方式如下图所示：

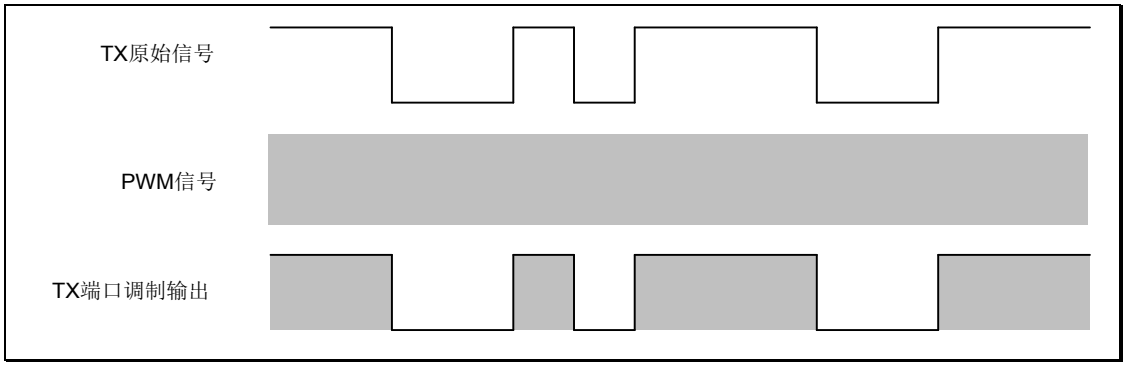


图 12-2 高电平调制输出波形图

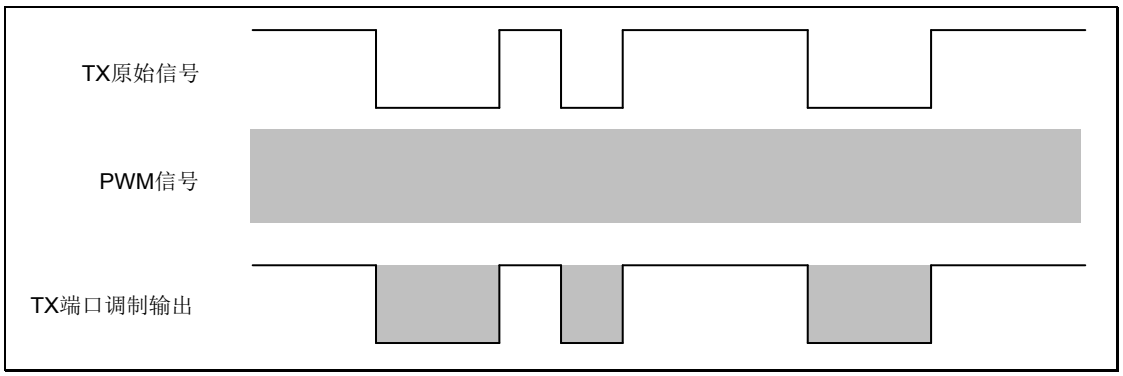


图 12-3 低电平调制输出波形图

以下为 UART 输出调制的参考配置流程（以 AD16C4T0 调制 UART0 为例）：

1. 设置寄存器 UART0_TXMCR.TXMSS 为 0001，选择 AD16C4T0 作为调制源。
2. 设置寄存器 UART0_TXMCR.TXSIGS 为 0000，选择 AD16C4T0 通道 1 输出作为调制信号。
3. 设置 UART0_TXMCR.TXMLVLS 选择调制电平。

4. 配置 AD16C4T0 进行计数。
5. 配置 UART 发送数据。

12. 5 特殊功能寄存器

12. 5. 1 寄存器列表

PIS 寄存器列表		
名称	偏移地址	描述
PIS_CH0_CON	0000_H	PIS 通道 0 控制寄存器
PIS_CH1_CON	0004_H	PIS 通道 1 控制寄存器
PIS_CH2_CON	0008_H	PIS 通道 2 控制寄存器
PIS_CH3_CON	$000C_H$	PIS 通道 3 控制寄存器
PIS_CH4_CON	0010_H	PIS 通道 4 控制寄存器
PIS_CH5_CON	0014_H	PIS 通道 5 控制寄存器
PIS_CH6_CON	0018_H	PIS 通道 6 控制寄存器
PIS_CH7_CON	$001C_H$	PIS 通道 7 控制寄存器
PIS_CH8_CON	0020_H	PIS 通道 8 控制寄存器
PIS_CH9_CON	0024_H	PIS 通道 9 控制寄存器
PIS_CH10_CON	0028_H	PIS 通道 10 控制寄存器
PIS_CH11_CON	$002C_H$	PIS 通道 11 控制寄存器
PIS_CH12_CON	0030_H	PIS 通道 12 控制寄存器
PIS_CH13_CON	0034_H	PIS 通道 13 控制寄存器
PIS_CH14_CON	0038_H	PIS 通道 14 控制寄存器
PIS_CH15_CON	$003C_H$	PIS 通道 15 控制寄存器
PIS_CH_OER	0040_H	PIS 通道端口输出使能寄存器
PIS_TAR_CON0	0044_H	PIS 消费端通道控制寄存器 0
PIS_TAR_CON1	0048_H	PIS 消费端通道控制寄存器 1
Reserved	$004C_H$~$005C_H$	保留
UART0_TXMCR	0060_H	UART0 输出调制控制寄存器
UART1_TXMCR	0064_H	UART1 输出调制控制寄存器
UART2_TXMCR	0068_H	UART2 输出调制控制寄存器
UART3_TXMCR	$006C_H$	UART3 输出调制控制寄存器
LPUART0_TXMCR	0070_H	LPUART0 输出调制控制寄存器

12. 5. 2 寄存器描述

12. 5. 2. 1 PIS通道 0 控制寄存器（PIS_CH0_CON）

PIS 通道 0 控制寄存器（PIS_CH0_CON）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
SYNCSEL	Bit 26-24	R/W	**信号同步选择：** 000：信号直通。 在以下场合可设置信号直通： 在一些低功耗应用的场合需采用异步的电平或脉冲； 生产端信号源与消费端处于同一时钟域，无需同步。 001：生产端为电平信号，消费端位于 APB1 时钟域 010：生产端为电平信号，消费端位于 APB2 时钟域 011：生产端为电平信号，消费端位于 AHB 时钟域 100：生产端为脉冲信号，消费端位于 APB1 时钟域 101：生产端为脉冲信号，消费端位于 APB2 时钟域 110：生产端为脉冲信号，消费端位于 AHB 时钟域 111：预留
Reserved	Bit 23-20	—	**保留**
TSCKS	Bit 19-18	R/W	**触发采样时钟选择** 在 SYNCSEL=000，且触发信号源与触发目标处于同一时钟域时，请根据触发目标时钟域进行设置，比如触发目标为 ADC，请将 TSCKS 设为 01 选择 PCLK2。其余情况下，触发采样时钟是确定的，该设定无效。 00：PCLK1 01：PCLK2 10：HCLK 11：预留

EDGS	Bit 17-16	R/W	**边沿选择：** 在输入信号为电平时使用，低功耗应用场合下的异步通道请设置为 00。 00：不输出边沿 01：上升沿 10：下降沿 11：双边沿
Reserved	Bit 15-14	—	**保留**
SRCS	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO 000010：- 000011：ACMP 000100：DAC0 000101：- 000110：ADC0 000111：LVD 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：RTC 010111：LP16T0 011000：LPUART0 011001：DMA 011010：ADC1 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=000000** MSIGS 无效 **SRCS=000001（SRC=GPIO）** 0000：PB0 0001：PB1

			0010：PB2 …… 1110：PB14 1111：PB15 **SRCS=000011（SRC=ACMP）** 0000：ACMP_OUT0 0001：ACMP_OUT1 **SRCS=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束 0001：DAC0 通道 1 转换结束 **SRCS=000110（SRC=ADC0）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 **SRCS=000111（SRC=LVD）** LVD 比较器输出 **SRCS=001000，001001，001010，001011（SRC=UART0，UART1，UART2，UART3）** 0000：保留 0001：保留 0010：IrDA 输出电平 0011：RTS 输出 0100：TX 输出 0101：发送空状态中断脉冲 0110：接收数据中断脉冲 **SRCS=001100，001101（SRC=UART4，UART5）** 0000：接收数据中断脉冲 0001：发送空状态中断脉冲 0010：TX 输出 **SRCS=001110，001111（SRC=SPI0，SPI1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=010010，010011，010100，010101（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1）** 0000: 更新事件脉冲 0001: 触发事件脉冲 0010: 通道 1 输入捕获脉冲 0011: 通道 1 输出比较脉冲 0100: 通道 2 输入捕获脉冲

			0101: 通道 2 输出比较脉冲 0110: 通道 3 输入捕获脉冲 0111: 通道 3 输出比较脉冲 1000: 通道 4 输入捕获脉冲 1001: 通道 4 输出比较脉冲 **SRCS=010110（SRC=RTC）** 0000：亚秒、秒、分、时、日、月、年脉冲 0001：闹钟 A 和闹钟 B **SRCS=010111（SRC=LP16T0）** 0000：保留 0001：异步更新事件脉冲 **SRCS=011000（SRC=LPUART0）** 0000：接收缓冲器非空异步脉冲 0001：发送缓冲器空异步脉冲 0010：接收缓冲器非空同步脉冲 0011：发送缓冲器空同步脉冲 **SRCS=011001（SRC=DMA）** DMA 通道完成脉冲 **SRCS=011010（SRC=ADC1）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1

12. 5. 2. 2 PIS通道 1 控制寄存器（PIS_CH1_CON）

PIS 通道 1 控制寄存器（PIS_CH1_CON）

偏移地址：04_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
SYNCSEL	Bit 26-24	R/W	**信号同步选择：** 000：信号直通。 在以下场合可设置信号直通： 在一些低功耗应用的场合需采用异步的电平或脉冲； 生产端信号源与消费端处于同一时钟域，无需同步。 001：生产端为电平信号，消费端位于 APB1 时钟域 010：生产端为电平信号，消费端位于 APB2 时钟域 011：生产端为电平信号，消费端位于 AHB 时钟域 100：生产端为脉冲信号，消费端位于 APB1 时钟域 101：生产端为脉冲信号，消费端位于 APB2 时钟域 110：生产端为脉冲信号，消费端位于 AHB 时钟域 111：预留
Reserved	Bit 23-20	—	**保留**
TSCKS	Bit 19-18	R/W	**触发采样时钟选择** 在 SYNCSEL=000，且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置，比如触发目标为 ADC，请将 TSCKS 设为 01 选择 PCLK2。其余情况下，触发采样时钟是确定的，该设定无效。 00：PCLK1 01：PCLK2 10：HCLK 11：预留
EDGS	Bit 17-16	R/W	**边沿选择：**

			在输入信号为电平时使用，低功耗应用场合下的异步通道请设置为 00。 00：不输出边沿 01：上升沿 10：下降沿 11：双边沿
Reserved	Bit 15-14	—	**保留**
SRCS	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO 000010：- 000011：ACMP 000100：DAC0 000101：- 000110：ADC0 000111：LVD 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：RTC 010111：LP16T0 011000：LPUART0 011001：DMA 011010：ADC1 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=000000** MSIGS 无效 **SRCS=000001（SRC=GPIO）** 0000：PB0 0001：PB1 0010：PB2

			…… 1110：PB14 1111：PB15 **SRCS=000011（SRC=ACMP）** 0000：ACMP_OUT0 0001：ACMP_OUT1 **SRCS=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束 0001：DAC0 通道 1 转换结束 **SRCS=000110（SRC=ADC0）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 **SRCS=000111（SRC=LVD）** LVD 比较器输出 **SRCS=001000，001001，001010，001011（SRC=UART0，UART1，UART2，UART3）** 0000：保留 0001：保留 0010：IrDA 输出电平 0011：RTS 输出 0100：TX 输出 0101：发送空状态中断脉冲 0110：接收数据中断脉冲 **SRCS=001100，001101（SRC=UART4，UART5）** 0000：接收数据中断脉冲 0001：发送空状态中断脉冲 0010：TX 输出 **SRCS=001110，001111（SRC=SPI0，SPI1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=010010，010011，010100，010101（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1）** 0000: 更新事件脉冲 0001: 触发事件脉冲 0010: 通道 1 输入捕获脉冲 0011: 通道 1 输出比较脉冲 0100: 通道 2 输入捕获脉冲 0101: 通道 2 输出比较脉冲

			0110: 通道 3 输入捕获脉冲 0111: 通道 3 输出比较脉冲 1000: 通道 4 输入捕获脉冲 1001: 通道 4 输出比较脉冲 **SRCS=010110（SRC=RTC）** 0000：亚秒、秒、分、时、日、月、年脉冲 0001：闹钟 A 和闹钟 B **SRCS=010111（SRC=LP16T0）** 0000：保留 0001：异步更新事件脉冲 **SRCS=011000（SRC=LPUART0）** 0000：接收缓冲器非空异步脉冲 0001：发送缓冲器空异步脉冲 0010：接收缓冲器非空同步脉冲 0011：发送缓冲器空同步脉冲 **SRCS=011001（SRC=DMA）** DMA 通道完成脉冲 **SRCS=011010（SRC=ADC1）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1

12. 5. 2. 3 PIS通道 2 控制寄存器（PIS_CH2_CON）

PIS 通道 2 控制寄存器（PIS_CH2_CON）

偏移地址：08_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
SYNCSEL	Bit 26-24	R/W	**信号同步选择：** 000：信号直通。 在以下场合可设置信号直通： 在一些低功耗应用的场合需采用异步的电平或脉冲； 生产端信号源与消费端处于同一时钟域，无需同步。 001：生产端为电平信号，消费端位于 APB1 时钟域 010：生产端为电平信号，消费端位于 APB2 时钟域 011：生产端为电平信号，消费端位于 AHB 时钟域 100：生产端为脉冲信号，消费端位于 APB1 时钟域 101：生产端为脉冲信号，消费端位于 APB2 时钟域 110：生产端为脉冲信号，消费端位于 AHB 时钟域 111：预留
Reserved	Bit 23-20	—	**保留**
TSCKS	Bit 19-18	R/W	**触发采样时钟选择** 在 SYNCSEL=000，且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置，比如触发目标为 ADC，请将 TSCKS 设为 01 选择 PCLK2。其余情况下，触发采样时钟是确定的，该设定无效。 00：PCLK1 01：PCLK2 10：HCLK 11：预留
EDGS	Bit 17-16	R/W	**边沿选择：**

			在输入信号为电平时使用，低功耗应用场合下的异步通道请设置为 00。 00：不输出边沿 01：上升沿 10：下降沿 11：双边沿
Reserved	Bit 15-14	—	**保留**
SRCS	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO 000010：- 000011：ACMP 000100：DAC0 000101：- 000110：ADC0 000111：LVD 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：RTC 010111：LP16T0 011000：LPUART0 011001：DMA 011010：ADC1 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=000000** MSIGS 无效 **SRCS=000001（SRC=GPIO）** 0000：PB0 0001：PB1 0010：PB2

			…… 1110：PB14 1111：PB15 **SRCS=000011（SRC=ACMP）** 0000：ACMP_OUT0 0001：ACMP_OUT1 **SRCS=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束 0001：DAC0 通道 1 转换结束 **SRCS=000110（SRC=ADC0）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 **SRCS=000111（SRC=LVD）** LVD 比较器输出 **SRCS=001000，001001，001010，001011（SRC=UART0，UART1，UART2，UART3）** 0000：保留 0001：保留 0010：IrDA 输出电平 0011：RTS 输出 0100：TX 输出 0101：发送空状态中断脉冲 0110：接收数据中断脉冲 **SRCS=001100，001101（SRC=UART4，UART5）** 0000：接收数据中断脉冲 0001：发送空状态中断脉冲 0010：TX 输出 **SRCS=001110，001111（SRC=SPI0，SPI1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=010010，010011，010100，010101（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1）** 0000: 更新事件脉冲 0001: 触发事件脉冲 0010: 通道 1 输入捕获脉冲 0011: 通道 1 输出比较脉冲 0100: 通道 2 输入捕获脉冲 0101: 通道 2 输出比较脉冲

			0110: 通道 3 输入捕获脉冲 0111: 通道 3 输出比较脉冲 1000: 通道 4 输入捕获脉冲 1001: 通道 4 输出比较脉冲 **SRCS=010110（SRC=RTC）** 0000：亚秒、秒、分、时、日、月、年脉冲 0001：闹钟 A 和闹钟 B **SRCS=010111（SRC=LP16T0）** 0000：保留 0001：异步更新事件脉冲 **SRCS=011000（SRC=LPUART0）** 0000：接收缓冲器非空异步脉冲 0001：发送缓冲器空异步脉冲 0010：接收缓冲器非空同步脉冲 0011：发送缓冲器空同步脉冲 **SRCS=011001（SRC=DMA）** DMA 通道完成脉冲 **SRCS=011010（SRC=ADC1）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1

12. 5. 2. 4 PIS通道 3 控制寄存器（PIS_CH3_CON）

PIS 通道 3 控制寄存器（PIS_CH3_CON）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
SYNCSEL	Bit 26-24	R/W	**信号同步选择：** 000：信号直通。 在以下场合可设置信号直通： 在一些低功耗应用的场合需采用异步的电平或脉冲； 生产端信号源与消费端处于同一时钟域，无需同步。 001：生产端为电平信号，消费端位于 APB1 时钟域 010：生产端为电平信号，消费端位于 APB2 时钟域 011：生产端为电平信号，消费端位于 AHB 时钟域 100：生产端为脉冲信号，消费端位于 APB1 时钟域 101：生产端为脉冲信号，消费端位于 APB2 时钟域 110：生产端为脉冲信号，消费端位于 AHB 时钟域 111：预留
Reserved	Bit 23-20	—	**保留**
TSCKS	Bit 19-18	R/W	**触发采样时钟选择** 在 SYNCSEL=000，且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置，比如触发目标为 ADC，请将 TSCKS 设为 01 选择 PCLK2。其余情况下，触发采样时钟是确定的，该设定无效。 00：PCLK1 01：PCLK2 10：HCLK 11：预留
EDGS	Bit 17-16	R/W	**边沿选择：**

			在输入信号为电平时使用，低功耗应用场合下的异步通道请设置为 00。 00：不输出边沿 01：上升沿 10：下降沿 11：双边沿
Reserved	Bit 15-14	—	**保留**
SRCS	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO 000010：- 000011：ACMP 000100：DAC0 000101：- 000110：ADC0 000111：LVD 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：RTC 010111：LP16T0 011000：LPUART0 011001：DMA 011010：ADC1 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=000000** MSIGS 无效 **SRCS=000001（SRC=GPIO）** 0000：PB0 0001：PB1 0010：PB2

			…… 1110：PB14 1111：PB15 **SRCS=000011（SRC=ACMP）** 0000：ACMP_OUT0 0001：ACMP_OUT1 **SRCS=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束 0001：DAC0 通道 1 转换结束 **SRCS=000110（SRC=ADC0）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 **SRCS=000111（SRC=LVD）** LVD 比较器输出 **SRCS=001000，001001，001010，001011（SRC=UART0，UART1，UART2，UART3）** 0000：保留 0001：保留 0010：IrDA 输出电平 0011：RTS 输出 0100：TX 输出 0101：发送空状态中断脉冲 0110：接收数据中断脉冲 **SRCS=001100，001101（SRC=UART4，UART5）** 0000：接收数据中断脉冲 0001：发送空状态中断脉冲 0010：TX 输出 **SRCS=001110，001111（SRC=SPI0，SPI1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=010010，010011，010100，010101（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1）** 0000: 更新事件脉冲 0001: 触发事件脉冲 0010: 通道 1 输入捕获脉冲 0011: 通道 1 输出比较脉冲 0100: 通道 2 输入捕获脉冲 0101: 通道 2 输出比较脉冲

			0110: 通道 3 输入捕获脉冲 0111: 通道 3 输出比较脉冲 1000: 通道 4 输入捕获脉冲 1001: 通道 4 输出比较脉冲 **SRCS=010110（SRC=RTC）** 0000：亚秒、秒、分、时、日、月、年脉冲 0001：闹钟 A 和闹钟 B **SRCS=010111（SRC=LP16T0）** 0000：保留 0001：异步更新事件脉冲 **SRCS=011000（SRC=LPUART0）** 0000：接收缓冲器非空异步脉冲 0001：发送缓冲器空异步脉冲 0010：接收缓冲器非空同步脉冲 0011：发送缓冲器空同步脉冲 **SRCS=011001（SRC=DMA）** DMA 通道完成脉冲 **SRCS=011010（SRC=ADC1）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1

12. 5. 2. 5 PIS通道 4 控制寄存器（PIS_CH4_CON）

PIS 通道 4 控制寄存器（PIS_CH4_CON）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
SYNCSEL	Bit 26-24	R/W	**信号同步选择：** 000：信号直通。 在以下场合可设置信号直通： 在一些低功耗应用的场合需采用异步的电平或脉冲； 生产端信号源与消费端处于同一时钟域，无需同步。 001：生产端为电平信号，消费端位于 APB1 时钟域 010：生产端为电平信号，消费端位于 APB2 时钟域 011：生产端为电平信号，消费端位于 AHB 时钟域 100：生产端为脉冲信号，消费端位于 APB1 时钟域 101：生产端为脉冲信号，消费端位于 APB2 时钟域 110：生产端为脉冲信号，消费端位于 AHB 时钟域 111：预留
Reserved	Bit 23-20	—	**保留**
TSCKS	Bit 19-18	R/W	**触发采样时钟选择** 在 SYNCSEL=000，且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置，比如触发目标为 ADC，请将 TSCKS 设为 01 选择 PCLK2。其余情况下，触发采样时钟是确定的，该设定无效。 00：PCLK1 01：PCLK2 10：HCLK 11：预留
EDGS	Bit 17-16	R/W	**边沿选择：**

			在输入信号为电平时使用，低功耗应用场合下的异步通道请设置为 00。 00：不输出边沿 01：上升沿 10：下降沿 11：双边沿
Reserved	Bit 15-14	—	**保留**
SRCS	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO 000010：- 000011：ACMP 000100：DAC0 000101：- 000110：ADC0 000111：LVD 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：RTC 010111：LP16T0 011000：LPUART0 011001：DMA 011010：ADC1 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=000000** MSIGS 无效 **SRCS=000001（SRC=GPIO）** 0000：PB0 0001：PB1 0010：PB2

			…… 1110：PB14 1111：PB15 **SRCS=000011（SRC=ACMP）** 0000：ACMP_OUT0 0001：ACMP_OUT1 **SRCS=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束 0001：DAC0 通道 1 转换结束 **SRCS=000110（SRC=ADC0）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 **SRCS=000111（SRC=LVD）** LVD 比较器输出 **SRCS=001000，001001，001010，001011（SRC=UART0，UART1，UART2，UART3）** 0000：保留 0001：保留 0010：IrDA 输出电平 0011：RTS 输出 0100：TX 输出 0101：发送空状态中断脉冲 0110：接收数据中断脉冲 **SRCS=001100，001101（SRC=UART4，UART5）** 0000：接收数据中断脉冲 0001：发送空状态中断脉冲 0010：TX 输出 **SRCS=001110，001111（SRC=SPI0，SPI1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=010010，010011，010100，010101（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1）** 0000: 更新事件脉冲 0001: 触发事件脉冲 0010: 通道 1 输入捕获脉冲 0011: 通道 1 输出比较脉冲 0100: 通道 2 输入捕获脉冲 0101: 通道 2 输出比较脉冲

			0110: 通道 3 输入捕获脉冲 0111: 通道 3 输出比较脉冲 1000: 通道 4 输入捕获脉冲 1001: 通道 4 输出比较脉冲 **SRCS=010110（SRC=RTC）** 0000：亚秒、秒、分、时、日、月、年脉冲 0001：闹钟 A 和闹钟 B **SRCS=010111（SRC=LP16T0）** 0000：保留 0001：异步更新事件脉冲 **SRCS=011000（SRC=LPUART0）** 0000：接收缓冲器非空异步脉冲 0001：发送缓冲器空异步脉冲 0010：接收缓冲器非空同步脉冲 0011：发送缓冲器空同步脉冲 **SRCS=011001（SRC=DMA）** DMA 通道完成脉冲 **SRCS=011010（SRC=ADC1）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1

12. 5. 2. 6 PIS通道 5 控制寄存器（PIS_CH5_CON）

PIS 通道 5 控制寄存器（PIS_CH5_CON）

偏移地址：14_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
SYNCSEL	Bit 26-24	R/W	**信号同步选择：** 000：信号直通。 在以下场合可设置信号直通： 在一些低功耗应用的场合需采用异步的电平或脉冲； 生产端信号源与消费端处于同一时钟域，无需同步。 001：生产端为电平信号，消费端位于 APB1 时钟域 010：生产端为电平信号，消费端位于 APB2 时钟域 011：生产端为电平信号，消费端位于 AHB 时钟域 100：生产端为脉冲信号，消费端位于 APB1 时钟域 101：生产端为脉冲信号，消费端位于 APB2 时钟域 110：生产端为脉冲信号，消费端位于 AHB 时钟域 111：预留
Reserved	Bit 23-20	—	**保留**
TSCKS	Bit 19-18	R/W	**触发采样时钟选择** 在 SYNCSEL=000，且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置，比如触发目标为 ADC，请将 TSCKS 设为 01 选择 PCLK2。其余情况下，触发采样时钟是确定的，该设定无效。 00：PCLK1 01：PCLK2 10：HCLK 11：预留
EDGS	Bit 17-16	R/W	**边沿选择：**

			在输入信号为电平时使用，低功耗应用场合下的异步通道请设置为 00。 00：不输出边沿 01：上升沿 10：下降沿 11：双边沿
Reserved	Bit 15-14	—	**保留**
SRCS	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO 000010：- 000011：ACMP 000100：DAC0 000101：- 000110：ADC0 000111：LVD 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：RTC 010111：LP16T0 011000：LPUART0 011001：DMA 011010：ADC1 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=000000** MSIGS 无效 **SRCS=000001（SRC=GPIO）** 0000：PB0 0001：PB1 0010：PB2

			…… 1110：PB14 1111：PB15 **SRCS=000011（SRC=ACMP）** 0000：ACMP_OUT0 0001：ACMP_OUT1 **SRCS=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束 0001：DAC0 通道 1 转换结束 **SRCS=000110（SRC=ADC0）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 **SRCS=000111（SRC=LVD）** LVD 比较器输出 **SRCS=001000，001001，001010，001011（SRC=UART0，UART1，UART2，UART3）** 0000：保留 0001：保留 0010：IrDA 输出电平 0011：RTS 输出 0100：TX 输出 0101：发送空状态中断脉冲 0110：接收数据中断脉冲 **SRCS=001100，001101（SRC=UART4，UART5）** 0000：接收数据中断脉冲 0001：发送空状态中断脉冲 0010：TX 输出 **SRCS=001110，001111（SRC=SPI0，SPI1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=010010，010011，010100，010101（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1）** 0000: 更新事件脉冲 0001: 触发事件脉冲 0010: 通道 1 输入捕获脉冲 0011: 通道 1 输出比较脉冲 0100: 通道 2 输入捕获脉冲 0101: 通道 2 输出比较脉冲

			0110: 通道 3 输入捕获脉冲 0111: 通道 3 输出比较脉冲 1000: 通道 4 输入捕获脉冲 1001: 通道 4 输出比较脉冲 **SRCS=010110（SRC=RTC）** 0000：亚秒、秒、分、时、日、月、年脉冲 0001：闹钟 A 和闹钟 B **SRCS=010111（SRC=LP16T0）** 0000：保留 0001：异步更新事件脉冲 **SRCS=011000（SRC=LPUART0）** 0000：接收缓冲器非空异步脉冲 0001：发送缓冲器空异步脉冲 0010：接收缓冲器非空同步脉冲 0011：发送缓冲器空同步脉冲 **SRCS=011001（SRC=DMA）** DMA 通道完成脉冲 **SRCS=011010（SRC=ADC1）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1

12. 5. 2. 7 PIS通道 6 控制寄存器（PIS_CH6_CON）

PIS 通道 6 控制寄存器（PIS_CH6_CON）

偏移地址：18_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
SYNCSEL	Bit 26-24	R/W	**信号同步选择：** 000：信号直通。 在以下场合可设置信号直通： 在一些低功耗应用的场合需采用异步的电平或脉冲； 生产端信号源与消费端处于同一时钟域，无需同步。 001：生产端为电平信号，消费端位于 APB1 时钟域 010：生产端为电平信号，消费端位于 APB2 时钟域 011：生产端为电平信号，消费端位于 AHB 时钟域 100：生产端为脉冲信号，消费端位于 APB1 时钟域 101：生产端为脉冲信号，消费端位于 APB2 时钟域 110：生产端为脉冲信号，消费端位于 AHB 时钟域 111：预留
Reserved	Bit 23-20	—	**保留**
TSCKS	Bit 19-18	R/W	**触发采样时钟选择** 在 SYNCSEL=000，且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置，比如触发目标为 ADC，请将 TSCKS 设为 01 选择 PCLK2。其余情况下，触发采样时钟是确定的，该设定无效。 00：PCLK1 01：PCLK2 10：HCLK 11：预留
EDGS	Bit 17-16	R/W	**边沿选择：**

			在输入信号为电平时使用，低功耗应用场合下的异步通道请设置为 00。 00：不输出边沿 01：上升沿 10：下降沿 11：双边沿
Reserved	Bit 15-14	—	**保留**
SRCS	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO 000010：- 000011：ACMP 000100：DAC0 000101：- 000110：ADC0 000111：LVD 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：RTC 010111：LP16T0 011000：LPUART0 011001：DMA 011010：ADC1 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=000000** MSIGS 无效 **SRCS=000001（SRC=GPIO）** 0000：PB0 0001：PB1 0010：PB2

			…… 1110：PB14 1111：PB15 **SRCS=000011（SRC=ACMP）** 0000：ACMP_OUT0 0001：ACMP_OUT1 **SRCS=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束 0001：DAC0 通道 1 转换结束 **SRCS=000110（SRC=ADC0）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 **SRCS=000111（SRC=LVD）** LVD 比较器输出 **SRCS=001000，001001，001010，001011（SRC=UART0，UART1，UART2，UART3）** 0000：保留 0001：保留 0010：IrDA 输出电平 0011：RTS 输出 0100：TX 输出 0101：发送空状态中断脉冲 0110：接收数据中断脉冲 **SRCS=001100，001101（SRC=UART4，UART5）** 0000：接收数据中断脉冲 0001：发送空状态中断脉冲 0010：TX 输出 **SRCS=001110，001111（SRC=SPI0，SPI1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=010010，010011，010100，010101（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1）** 0000: 更新事件脉冲 0001: 触发事件脉冲 0010: 通道 1 输入捕获脉冲 0011: 通道 1 输出比较脉冲 0100: 通道 2 输入捕获脉冲 0101: 通道 2 输出比较脉冲

			0110: 通道 3 输入捕获脉冲 0111: 通道 3 输出比较脉冲 1000: 通道 4 输入捕获脉冲 1001: 通道 4 输出比较脉冲 **SRCS=010110（SRC=RTC）** 0000：亚秒、秒、分、时、日、月、年脉冲 0001：闹钟 A 和闹钟 B **SRCS=010111（SRC=LP16T0）** 0000：保留 0001：异步更新事件脉冲 **SRCS=011000（SRC=LPUART0）** 0000：接收缓冲器非空异步脉冲 0001：发送缓冲器空异步脉冲 0010：接收缓冲器非空同步脉冲 0011：发送缓冲器空同步脉冲 **SRCS=011001（SRC=DMA）** DMA 通道完成脉冲 **SRCS=011010（SRC=ADC1）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1

12. 5. 2. 8 PIS通道 7 控制寄存器（PIS_CH7_CON）

PIS 通道 7 控制寄存器（PIS_CH7_CON）

偏移地址：$1C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
SYNCSEL	Bit 26-24	R/W	**信号同步选择：** 000：信号直通。 在以下场合可设置信号直通： 在一些低功耗应用的场合需采用异步的电平或脉冲； 生产端信号源与消费端处于同一时钟域，无需同步。 001：生产端为电平信号，消费端位于 APB1 时钟域 010：生产端为电平信号，消费端位于 APB2 时钟域 011：生产端为电平信号，消费端位于 AHB 时钟域 100：生产端为脉冲信号，消费端位于 APB1 时钟域 101：生产端为脉冲信号，消费端位于 APB2 时钟域 110：生产端为脉冲信号，消费端位于 AHB 时钟域 111：预留
Reserved	Bit 23-20	—	**保留**
TSCKS	Bit 19-18	R/W	**触发采样时钟选择** 在 SYNCSEL=000，且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置，比如触发目标为 ADC，请将 TSCKS 设为 01 选择 PCLK2。其余情况下，触发采样时钟是确定的，该设定无效。 00：PCLK1 01：PCLK2 10：HCLK 11：预留
EDGS	Bit 17-16	R/W	**边沿选择：**

			在输入信号为电平时使用，低功耗应用场合下的异步通道请设置为 00。 00：不输出边沿 01：上升沿 10：下降沿 11：双边沿
Reserved	Bit 15-14	—	**保留**
SRCS	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO 000010：- 000011：ACMP 000100：DAC0 000101：- 000110：ADC0 000111：LVD 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：RTC 010111：LP16T0 011000：LPUART0 011001：DMA 011010：ADC1 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=000000** MSIGS 无效 **SRCS=000001（SRC=GPIO）** 0000：PB0 0001：PB1 0010：PB2

			…… 1110：PB14 1111：PB15 **SRCS=000011（SRC=ACMP）** 0000：ACMP_OUT0 0001：ACMP_OUT1 **SRCS=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束 0001：DAC0 通道 1 转换结束 **SRCS=000110（SRC=ADC0）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 **SRCS=000111（SRC=LVD）** LVD 比较器输出 **SRCS=001000，001001，001010，001011（SRC=UART0，UART1，UART2，UART3）** 0000：保留 0001：保留 0010：IrDA 输出电平 0011：RTS 输出 0100：TX 输出 0101：发送空状态中断脉冲 0110：接收数据中断脉冲 **SRCS=001100，001101（SRC=UART4，UART5）** 0000：接收数据中断脉冲 0001：发送空状态中断脉冲 0010：TX 输出 **SRCS=001110，001111（SRC=SPI0，SPI1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=010010，010011，010100，010101（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1）** 0000: 更新事件脉冲 0001: 触发事件脉冲 0010: 通道 1 输入捕获脉冲 0011: 通道 1 输出比较脉冲 0100: 通道 2 输入捕获脉冲 0101: 通道 2 输出比较脉冲

			0110: 通道 3 输入捕获脉冲 0111: 通道 3 输出比较脉冲 1000: 通道 4 输入捕获脉冲 1001: 通道 4 输出比较脉冲 **SRCS=010110（SRC=RTC）** 0000：亚秒、秒、分、时、日、月、年脉冲 0001：闹钟 A 和闹钟 B **SRCS=010111（SRC=LP16T0）** 0000：保留 0001：异步更新事件脉冲 **SRCS=011000（SRC=LPUART0）** 0000：接收缓冲器非空异步脉冲 0001：发送缓冲器空异步脉冲 0010：接收缓冲器非空同步脉冲 0011：发送缓冲器空同步脉冲 **SRCS=011001（SRC=DMA）** DMA 通道完成脉冲 **SRCS=011010（SRC=ADC1）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1

12. 5. 2. 9 PIS通道 8 控制寄存器（PIS_CH8_CON）

PIS 通道 8 控制寄存器（PIS_CH8_CON）

偏移地址：20_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
SYNCSEL	Bit 26-24	R/W	**信号同步选择：** 000：信号直通。 在以下场合可设置信号直通： 在一些低功耗应用的场合需采用异步的电平或脉冲； 生产端信号源与消费端处于同一时钟域，无需同步。 001：生产端为电平信号，消费端位于 APB1 时钟域 010：生产端为电平信号，消费端位于 APB2 时钟域 011：生产端为电平信号，消费端位于 AHB 时钟域 100：生产端为脉冲信号，消费端位于 APB1 时钟域 101：生产端为脉冲信号，消费端位于 APB2 时钟域 110：生产端为脉冲信号，消费端位于 AHB 时钟域 111：预留
Reserved	Bit 23-20	—	**保留**
TSCKS	Bit 19-18	R/W	**触发采样时钟选择** 在 SYNCSEL=000，且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置，比如触发目标为 ADC，请将 TSCKS 设为 01 选择 PCLK2。其余情况下，触发采样时钟是确定的，该设定无效。 00：PCLK1 01：PCLK2 10：HCLK 11：预留
EDGS	Bit 17-16	R/W	**边沿选择：**

			在输入信号为电平时使用，低功耗应用场合下的异步通道请设置为 00。 00：不输出边沿 01：上升沿 10：下降沿 11：双边沿
Reserved	Bit 15-14	—	**保留**
SRCS	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO 000010：- 000011：ACMP 000100：DAC0 000101：- 000110：ADC0 000111：LVD 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：RTC 010111：LP16T0 011000：LPUART0 011001：DMA 011010：ADC1 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=000000** MSIGS 无效 **SRCS=000001（SRC=GPIO）** 0000：PB0 0001：PB1 0010：PB2

			…… 1110：PB14 1111：PB15 **SRCS=000011（SRC=ACMP）** 0000：ACMP_OUT0 0001：ACMP_OUT1 **SRCS=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束 0001：DAC0 通道 1 转换结束 **SRCS=000110（SRC=ADC0）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 **SRCS=000111（SRC=LVD）** LVD 比较器输出 **SRCS=001000，001001，001010，001011（SRC=UART0，UART1，UART2，UART3）** 0000：保留 0001：保留 0010：IrDA 输出电平 0011：RTS 输出 0100：TX 输出 0101：发送空状态中断脉冲 0110：接收数据中断脉冲 **SRCS=001100，001101（SRC=UART4，UART5）** 0000：接收数据中断脉冲 0001：发送空状态中断脉冲 0010：TX 输出 **SRCS=001110，001111（SRC=SPI0，SPI1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=010010，010011，010100，010101（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1）** 0000: 更新事件脉冲 0001: 触发事件脉冲 0010: 通道 1 输入捕获脉冲 0011: 通道 1 输出比较脉冲 0100: 通道 2 输入捕获脉冲 0101: 通道 2 输出比较脉冲

			0110: 通道 3 输入捕获脉冲 0111: 通道 3 输出比较脉冲 1000: 通道 4 输入捕获脉冲 1001: 通道 4 输出比较脉冲 **SRCS=010110（SRC=RTC）** 0000：亚秒、秒、分、时、日、月、年脉冲 0001：闹钟 A 和闹钟 B **SRCS=010111（SRC=LP16T0）** 0000：保留 0001：异步更新事件脉冲 **SRCS=011000（SRC=LPUART0）** 0000：接收缓冲器非空异步脉冲 0001：发送缓冲器空异步脉冲 0010：接收缓冲器非空同步脉冲 0011：发送缓冲器空同步脉冲 **SRCS=011001（SRC=DMA）** DMA 通道完成脉冲 **SRCS=011010（SRC=ADC1）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1

12. 5. 2. 10 PIS通道 9 控制寄存器（PIS_CH9_CON）

PIS 通道 9 控制寄存器（PIS_CH9_CON）

偏移地址：24_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
SYNCSEL	Bit 26-24	R/W	**信号同步选择：** 000：信号直通。 在以下场合可设置信号直通： 在一些低功耗应用的场合需采用异步的电平或脉冲； 生产端信号源与消费端处于同一时钟域，无需同步。 001：生产端为电平信号，消费端位于 APB1 时钟域 010：生产端为电平信号，消费端位于 APB2 时钟域 011：生产端为电平信号，消费端位于 AHB 时钟域 100：生产端为脉冲信号，消费端位于 APB1 时钟域 101：生产端为脉冲信号，消费端位于 APB2 时钟域 110：生产端为脉冲信号，消费端位于 AHB 时钟域 111：预留
Reserved	Bit 23-20	—	**保留**
TSCKS	Bit 19-18	R/W	**触发采样时钟选择** 在 SYNCSEL=000，且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置，比如触发目标为 ADC，请将 TSCKS 设为 01 选择 PCLK2。其余情况下，触发采样时钟是确定的，该设定无效。 00：PCLK1 01：PCLK2 10：HCLK 11：预留
EDGS	Bit 17-16	R/W	**边沿选择：**

			在输入信号为电平时使用，低功耗应用场合下的异步通道请设置为 00。 00：不输出边沿 01：上升沿 10：下降沿 11：双边沿
Reserved	Bit 15-14	—	**保留**
SRCS	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO 000010：- 000011：ACMP 000100：DAC0 000101：- 000110：ADC0 000111：LVD 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：RTC 010111：LP16T0 011000：LPUART0 011001：DMA 011010：ADC1 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=000000** MSIGS 无效 **SRCS=000001（SRC=GPIO）** 0000：PB0 0001：PB1 0010：PB2

			…… 1110：PB14 1111：PB15 **SRCS=000011（SRC=ACMP）** 0000：ACMP_OUT0 0001：ACMP_OUT1 **SRCS=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束 0001：DAC0 通道 1 转换结束 **SRCS=000110（SRC=ADC0）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 **SRCS=000111（SRC=LVD）** LVD 比较器输出 **SRCS=001000，001001，001010，001011（SRC=UART0，UART1，UART2，UART3）** 0000：保留 0001：保留 0010：IrDA 输出电平 0011：RTS 输出 0100：TX 输出 0101：发送空状态中断脉冲 0110：接收数据中断脉冲 **SRCS=001100，001101（SRC=UART4，UART5）** 0000：接收数据中断脉冲 0001：发送空状态中断脉冲 0010：TX 输出 **SRCS=001110，001111（SRC=SPI0，SPI1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=010010，010011，010100，010101（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1）** 0000: 更新事件脉冲 0001: 触发事件脉冲 0010: 通道 1 输入捕获脉冲 0011: 通道 1 输出比较脉冲 0100: 通道 2 输入捕获脉冲 0101: 通道 2 输出比较脉冲

			0110: 通道 3 输入捕获脉冲 0111: 通道 3 输出比较脉冲 1000: 通道 4 输入捕获脉冲 1001: 通道 4 输出比较脉冲 **SRCS=010110（SRC=RTC）** 0000：亚秒、秒、分、时、日、月、年脉冲 0001：闹钟 A 和闹钟 B **SRCS=010111（SRC=LP16T0）** 0000：保留 0001：异步更新事件脉冲 **SRCS=011000（SRC=LPUART0）** 0000：接收缓冲器非空异步脉冲 0001：发送缓冲器空异步脉冲 0010：接收缓冲器非空同步脉冲 0011：发送缓冲器空同步脉冲 **SRCS=011001（SRC=DMA）** DMA 通道完成脉冲 **SRCS=011010（SRC=ADC1）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1

12. 5. 2. 11 PIS通道 10 控制寄存器（PIS_CH10_CON）

PIS 通道 10 控制寄存器（PIS_CH10_CON）

偏移地址：28_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
SYNCSEL	Bit 26-24	R/W	**信号同步选择：** 000：信号直通。 在以下场合可设置信号直通： 在一些低功耗应用的场合需采用异步的电平或脉冲； 生产端信号源与消费端处于同一时钟域，无需同步。 001：生产端为电平信号，消费端位于 APB1 时钟域 010：生产端为电平信号，消费端位于 APB2 时钟域 011：生产端为电平信号，消费端位于 AHB 时钟域 100：生产端为脉冲信号，消费端位于 APB1 时钟域 101：生产端为脉冲信号，消费端位于 APB2 时钟域 110：生产端为脉冲信号，消费端位于 AHB 时钟域 111：预留
Reserved	Bit 23-20	—	**保留**
TSCKS	Bit 19-18	R/W	**触发采样时钟选择** 在 SYNCSEL=000，且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置，比如触发目标为 ADC，请将 TSCKS 设为 01 选择 PCLK2。其余情况下，触发采样时钟是确定的，该设定无效。 00：PCLK1 01：PCLK2 10：HCLK 11：预留
EDGS	Bit 17-16	R/W	**边沿选择：**

			在输入信号为电平时使用，低功耗应用场合下的异步通道请设置为 00。 00：不输出边沿 01：上升沿 10：下降沿 11：双边沿
Reserved	Bit 15-14	—	**保留**
SRCS	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO 000010：- 000011：ACMP 000100：DAC0 000101：- 000110：ADC0 000111：LVD 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：RTC 010111：LP16T0 011000：LPUART0 011001：DMA 011010：ADC1 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=000000** MSIGS 无效 **SRCS=000001（SRC=GPIO）** 0000：PB0 0001：PB1 0010：PB2

			…… 1110：PB14 1111：PB15 **SRCS=000011（SRC=ACMP）** 0000：ACMP_OUT0 0001：ACMP_OUT1 **SRCS=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束 0001：DAC0 通道 1 转换结束 **SRCS=000110（SRC=ADC0）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 **SRCS=000111（SRC=LVD）** LVD 比较器输出 **SRCS=001000，001001，001010，001011（SRC=UART0，UART1，UART2，UART3）** 0000：保留 0001：保留 0010：IrDA 输出电平 0011：RTS 输出 0100：TX 输出 0101：发送空状态中断脉冲 0110：接收数据中断脉冲 **SRCS=001100，001101（SRC=UART4，UART5）** 0000：接收数据中断脉冲 0001：发送空状态中断脉冲 0010：TX 输出 **SRCS=001110，001111（SRC=SPI0，SPI1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=010010，010011，010100，010101（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1）** 0000: 更新事件脉冲 0001: 触发事件脉冲 0010: 通道 1 输入捕获脉冲 0011: 通道 1 输出比较脉冲 0100: 通道 2 输入捕获脉冲 0101: 通道 2 输出比较脉冲

			0110: 通道 3 输入捕获脉冲 0111: 通道 3 输出比较脉冲 1000: 通道 4 输入捕获脉冲 1001: 通道 4 输出比较脉冲 **SRCS=010110（SRC=RTC）** 0000：亚秒、秒、分、时、日、月、年脉冲 0001：闹钟 A 和闹钟 B **SRCS=010111（SRC=LP16T0）** 0000：保留 0001：异步更新事件脉冲 **SRCS=011000（SRC=LPUART0）** 0000：接收缓冲器非空异步脉冲 0001：发送缓冲器空异步脉冲 0010：接收缓冲器非空同步脉冲 0011：发送缓冲器空同步脉冲 **SRCS=011001（SRC=DMA）** DMA 通道完成脉冲 **SRCS=011010（SRC=ADC1）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1

12. 5. 2. 12 PIS通道 11 控制寄存器（PIS_CH11_CON）

PIS 通道 11 控制寄存器（PIS_CH11_CON）

偏移地址：$2C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
SYNCSEL	Bit 26-24	R/W	**信号同步选择：** 000：信号直通。 在以下场合可设置信号直通： 在一些低功耗应用的场合需采用异步的电平或脉冲； 生产端信号源与消费端处于同一时钟域，无需同步。 001：生产端为电平信号，消费端位于 APB1 时钟域 010：生产端为电平信号，消费端位于 APB2 时钟域 011：生产端为电平信号，消费端位于 AHB 时钟域 100：生产端为脉冲信号，消费端位于 APB1 时钟域 101：生产端为脉冲信号，消费端位于 APB2 时钟域 110：生产端为脉冲信号，消费端位于 AHB 时钟域 111：预留
Reserved	Bit 23-20	—	**保留**
TSCKS	Bit 19-18	R/W	**触发采样时钟选择** 在 SYNCSEL=000，且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置，比如触发目标为 ADC，请将 TSCKS 设为 01 选择 PCLK2。其余情况下，触发采样时钟是确定的，该设定无效。 00：PCLK1 01：PCLK2 10：HCLK 11：预留
EDGS	Bit 17-16	R/W	**边沿选择：**

			在输入信号为电平时使用，低功耗应用场合下的异步通道请设置为 00。 00：不输出边沿 01：上升沿 10：下降沿 11：双边沿
Reserved	Bit 15-14	—	**保留**
SRCS	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO 000010：- 000011：ACMP 000100：DAC0 000101：- 000110：ADC0 000111：LVD 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：RTC 010111：LP16T0 011000：LPUART0 011001：DMA 011010：ADC1 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=000000** MSIGS 无效 **SRCS=000001（SRC=GPIO）** 0000：PB0 0001：PB1 0010：PB2

			…… 1110：PB14 1111：PB15 **SRCS=000011（SRC=ACMP）** 0000：ACMP_OUT0 0001：ACMP_OUT1 **SRCS=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束 0001：DAC0 通道 1 转换结束 **SRCS=000110（SRC=ADC0）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 **SRCS=000111（SRC=LVD）** LVD 比较器输出 **SRCS=001000，001001，001010，001011（SRC=UART0，UART1，UART2，UART3）** 0000：保留 0001：保留 0010：IrDA 输出电平 0011：RTS 输出 0100：TX 输出 0101：发送空状态中断脉冲 0110：接收数据中断脉冲 **SRCS=001100，001101（SRC=UART4，UART5）** 0000：接收数据中断脉冲 0001：发送空状态中断脉冲 0010：TX 输出 **SRCS=001110，001111（SRC=SPI0，SPI1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=010010，010011，010100，010101（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1）** 0000: 更新事件脉冲 0001: 触发事件脉冲 0010: 通道 1 输入捕获脉冲 0011: 通道 1 输出比较脉冲 0100: 通道 2 输入捕获脉冲 0101: 通道 2 输出比较脉冲

			0110: 通道 3 输入捕获脉冲 0111: 通道 3 输出比较脉冲 1000: 通道 4 输入捕获脉冲 1001: 通道 4 输出比较脉冲 **SRCS=010110（SRC=RTC）** 0000：亚秒、秒、分、时、日、月、年脉冲 0001：闹钟 A 和闹钟 B **SRCS=010111（SRC=LP16T0）** 0000：保留 0001：异步更新事件脉冲 **SRCS=011000（SRC=LPUART0）** 0000：接收缓冲器非空异步脉冲 0001：发送缓冲器空异步脉冲 0010：接收缓冲器非空同步脉冲 0011：发送缓冲器空同步脉冲 **SRCS=011001（SRC=DMA）** DMA 通道完成脉冲 **SRCS=011010（SRC=ADC1）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1

12. 5. 2. 13 PIS通道 12 控制寄存器（PIS_CH12_CON）

PIS 通道 12 控制寄存器（PIS_CH12_CON）

偏移地址：30_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
SYNCSEL	Bit 26-24	R/W	**信号同步选择：** 000：信号直通。 在以下场合可设置信号直通： 在一些低功耗应用的场合需采用异步的电平或脉冲； 生产端信号源与消费端处于同一时钟域，无需同步。 001：生产端为电平信号，消费端位于 APB1 时钟域 010：生产端为电平信号，消费端位于 APB2 时钟域 011：生产端为电平信号，消费端位于 AHB 时钟域 100：生产端为脉冲信号，消费端位于 APB1 时钟域 101：生产端为脉冲信号，消费端位于 APB2 时钟域 110：生产端为脉冲信号，消费端位于 AHB 时钟域 111：预留
Reserved	Bit 23-20	—	**保留**
TSCKS	Bit 19-18	R/W	**触发采样时钟选择** 在 SYNCSEL=000，且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置，比如触发目标为 ADC，请将 TSCKS 设为 01 选择 PCLK2。其余情况下，触发采样时钟是确定的，该设定无效。 00：PCLK1 01：PCLK2 10：HCLK 11：预留
EDGS	Bit 17-16	R/W	**边沿选择：**

			在输入信号为电平时使用，低功耗应用场合下的异步通道请设置为 00。 00：不输出边沿 01：上升沿 10：下降沿 11：双边沿
Reserved	Bit 15-14	—	**保留**
SRCS	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO 000010：- 000011：ACMP 000100：DAC0 000101：- 000110：ADC0 000111：LVD 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：RTC 010111：LP16T0 011000：LPUART0 011001：DMA 011010：ADC1 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=000000** MSIGS 无效 **SRCS=000001（SRC=GPIO）** 0000：PB0 0001：PB1 0010：PB2

			…… 1110：PB14 1111：PB15 **SRCS=000011（SRC=ACMP）** 0000：ACMP_OUT0 0001：ACMP_OUT1 **SRCS=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束 0001：DAC0 通道 1 转换结束 **SRCS=000110（SRC=ADC0）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 **SRCS=000111（SRC=LVD）** LVD 比较器输出 **SRCS=001000，001001，001010，001011（SRC=UART0，UART1，UART2，UART3）** 0000：保留 0001：保留 0010：IrDA 输出电平 0011：RTS 输出 0100：TX 输出 0101：发送空状态中断脉冲 0110：接收数据中断脉冲 **SRCS=001100，001101（SRC=UART4，UART5）** 0000：接收数据中断脉冲 0001：发送空状态中断脉冲 0010：TX 输出 **SRCS=001110，001111（SRC=SPI0，SPI1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=010010，010011，010100，010101（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1）** 0000: 更新事件脉冲 0001: 触发事件脉冲 0010: 通道 1 输入捕获脉冲 0011: 通道 1 输出比较脉冲 0100: 通道 2 输入捕获脉冲 0101: 通道 2 输出比较脉冲

			0110: 通道 3 输入捕获脉冲 0111: 通道 3 输出比较脉冲 1000: 通道 4 输入捕获脉冲 1001: 通道 4 输出比较脉冲 **SRCS=010110（SRC=RTC）** 0000：亚秒、秒、分、时、日、月、年脉冲 0001：闹钟 A 和闹钟 B **SRCS=010111（SRC=LP16T0）** 0000：保留 0001：异步更新事件脉冲 **SRCS=011000（SRC=LPUART0）** 0000：接收缓冲器非空异步脉冲 0001：发送缓冲器空异步脉冲 0010：接收缓冲器非空同步脉冲 0011：发送缓冲器空同步脉冲 **SRCS=011001（SRC=DMA）** DMA 通道完成脉冲 **SRCS=011010（SRC=ADC1）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1

12. 5. 2. 14 PIS通道 13 控制寄存器（PIS_CH13_CON）

PIS 通道 13 控制寄存器（PIS_CH13_CON）

偏移地址：34_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
SYNCSEL	Bit 26-24	R/W	**信号同步选择：** 000：信号直通。 在以下场合可设置信号直通： 在一些低功耗应用的场合需采用异步的电平或脉冲； 生产端信号源与消费端处于同一时钟域，无需同步。 001：生产端为电平信号，消费端位于 APB1 时钟域 010：生产端为电平信号，消费端位于 APB2 时钟域 011：生产端为电平信号，消费端位于 AHB 时钟域 100：生产端为脉冲信号，消费端位于 APB1 时钟域 101：生产端为脉冲信号，消费端位于 APB2 时钟域 110：生产端为脉冲信号，消费端位于 AHB 时钟域 111：预留
Reserved	Bit 23-20	—	**保留**
TSCKS	Bit 19-18	R/W	**触发采样时钟选择** 在 SYNCSEL=000，且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置，比如触发目标为 ADC，请将 TSCKS 设为 01 选择 PCLK2。其余情况下，触发采样时钟是确定的，该设定无效。 00：PCLK1 01：PCLK2 10：HCLK 11：预留
EDGS	Bit 17-16	R/W	**边沿选择：**

			在输入信号为电平时使用，低功耗应用场合下的异步通道请设置为 00。 00：不输出边沿 01：上升沿 10：下降沿 11：双边沿
Reserved	Bit 15-14	—	**保留**
SRCS	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO 000010：- 000011：ACMP 000100：DAC0 000101：- 000110：ADC0 000111：LVD 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：RTC 010111：LP16T0 011000：LPUART0 011001：DMA 011010：ADC1 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=000000** MSIGS 无效 **SRCS=000001（SRC=GPIO）** 0000：PB0 0001：PB1 0010：PB2

			…… 1110：PB14 1111：PB15 **SRCS=000011（SRC=ACMP）** 0000：ACMP_OUT0 0001：ACMP_OUT1 **SRCS=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束 0001：DAC0 通道 1 转换结束 **SRCS=000110（SRC=ADC0）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 **SRCS=000111（SRC=LVD）** LVD 比较器输出 **SRCS=001000，001001，001010，001011（SRC=UART0，UART1，UART2，UART3）** 0000：保留 0001：保留 0010：IrDA 输出电平 0011：RTS 输出 0100：TX 输出 0101：发送空状态中断脉冲 0110：接收数据中断脉冲 **SRCS=001100，001101（SRC=UART4，UART5）** 0000：接收数据中断脉冲 0001：发送空状态中断脉冲 0010：TX 输出 **SRCS=001110，001111（SRC=SPI0，SPI1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=010010，010011，010100，010101（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1）** 0000: 更新事件脉冲 0001: 触发事件脉冲 0010: 通道 1 输入捕获脉冲 0011: 通道 1 输出比较脉冲 0100: 通道 2 输入捕获脉冲 0101: 通道 2 输出比较脉冲

			0110: 通道 3 输入捕获脉冲 0111: 通道 3 输出比较脉冲 1000: 通道 4 输入捕获脉冲 1001: 通道 4 输出比较脉冲 **SRCS=010110（SRC=RTC）** 0000：亚秒、秒、分、时、日、月、年脉冲 0001：闹钟 A 和闹钟 B **SRCS=010111（SRC=LP16T0）** 0000：保留 0001：异步更新事件脉冲 **SRCS=011000（SRC=LPUART0）** 0000：接收缓冲器非空异步脉冲 0001：发送缓冲器空异步脉冲 0010：接收缓冲器非空同步脉冲 0011：发送缓冲器空同步脉冲 **SRCS=011001（SRC=DMA）** DMA 通道完成脉冲 **SRCS=011010（SRC=ADC1）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1

12. 5. 2. 15 PIS通道 14 控制寄存器（PIS_CH14_CON）

PIS 通道 14 控制寄存器（PIS_CH14_CON）

偏移地址：38_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
SYNCSEL	Bit 26-24	R/W	**信号同步选择：** 000：信号直通。 在以下场合可设置信号直通： 在一些低功耗应用的场合需采用异步的电平或脉冲； 生产端信号源与消费端处于同一时钟域，无需同步。 001：生产端为电平信号，消费端位于 APB1 时钟域 010：生产端为电平信号，消费端位于 APB2 时钟域 011：生产端为电平信号，消费端位于 AHB 时钟域 100：生产端为脉冲信号，消费端位于 APB1 时钟域 101：生产端为脉冲信号，消费端位于 APB2 时钟域 110：生产端为脉冲信号，消费端位于 AHB 时钟域 111：预留
Reserved	Bit 23-20	—	**保留**
TSCKS	Bit 19-18	R/W	**触发采样时钟选择** 在 SYNCSEL=000，且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置，比如触发目标为 ADC，请将 TSCKS 设为 01 选择 PCLK2。其余情况下，触发采样时钟是确定的，该设定无效。 00：PCLK1 01：PCLK2 10：HCLK 11：预留
EDGS	Bit 17-16	R/W	**边沿选择：**

			在输入信号为电平时使用，低功耗应用场合下的异步通道请设置为 00。 00：不输出边沿 01：上升沿 10：下降沿 11：双边沿
Reserved	Bit 15-14	—	**保留**
SRCS	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO 000010：- 000011：ACMP 000100：DAC0 000101：- 000110：ADC0 000111：LVD 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：RTC 010111：LP16T0 011000：LPUART0 011001：DMA 011010：ADC1 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=000000** MSIGS 无效 **SRCS=000001（SRC=GPIO）** 0000：PB0 0001：PB1 0010：PB2

			…… 1110：PB14 1111：PB15 **SRCS=000011（SRC=ACMP）** 0000：ACMP_OUT0 0001：ACMP_OUT1 **SRCS=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束 0001：DAC0 通道 1 转换结束 **SRCS=000110（SRC=ADC0）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 **SRCS=000111（SRC=LVD）** LVD 比较器输出 **SRCS=001000，001001，001010，001011（SRC=UART0，UART1，UART2，UART3）** 0000：保留 0001：保留 0010：IrDA 输出电平 0011：RTS 输出 0100：TX 输出 0101：发送空状态中断脉冲 0110：接收数据中断脉冲 **SRCS=001100，001101（SRC=UART4，UART5）** 0000：接收数据中断脉冲 0001：发送空状态中断脉冲 0010：TX 输出 **SRCS=001110，001111（SRC=SPI0，SPI1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=010010，010011，010100，010101（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1）** 0000: 更新事件脉冲 0001: 触发事件脉冲 0010: 通道 1 输入捕获脉冲 0011: 通道 1 输出比较脉冲 0100: 通道 2 输入捕获脉冲 0101: 通道 2 输出比较脉冲

			0110: 通道 3 输入捕获脉冲 0111: 通道 3 输出比较脉冲 1000: 通道 4 输入捕获脉冲 1001: 通道 4 输出比较脉冲 **SRCS=010110（SRC=RTC）** 0000：亚秒、秒、分、时、日、月、年脉冲 0001：闹钟 A 和闹钟 B **SRCS=010111（SRC=LP16T0）** 0000：保留 0001：异步更新事件脉冲 **SRCS=011000（SRC=LPUART0）** 0000：接收缓冲器非空异步脉冲 0001：发送缓冲器空异步脉冲 0010：接收缓冲器非空同步脉冲 0011：发送缓冲器空同步脉冲 **SRCS=011001（SRC=DMA）** DMA 通道完成脉冲 **SRCS=011010（SRC=ADC1）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1

12. 5. 2. 16 PIS通道 15 控制寄存器（PIS_CH15_CON）

PIS 通道 15 控制寄存器（PIS_CH15_CON）

偏移地址：$3C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
SYNCSEL	Bit 26-24	R/W	**信号同步选择：** 000：信号直通。 在以下场合可设置信号直通： 在一些低功耗应用的场合需采用异步的电平或脉冲； 生产端信号源与消费端处于同一时钟域，无需同步。 001：生产端为电平信号，消费端位于 APB1 时钟域 010：生产端为电平信号，消费端位于 APB2 时钟域 011：生产端为电平信号，消费端位于 AHB 时钟域 100：生产端为脉冲信号，消费端位于 APB1 时钟域 101：生产端为脉冲信号，消费端位于 APB2 时钟域 110：生产端为脉冲信号，消费端位于 AHB 时钟域 111：预留
Reserved	Bit 23-20	—	**保留**
TSCKS	Bit 19-18	R/W	**触发采样时钟选择** 在 SYNCSEL=000，且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置，比如触发目标为 ADC，请将 TSCKS 设为 01 选择 PCLK2。其余情况下，触发采样时钟是确定的，该设定无效。 00：PCLK1 01：PCLK2 10：HCLK 11：预留
EDGS	Bit 17-16	R/W	**边沿选择：**

			在输入信号为电平时使用，低功耗应用场合下的异步通道请设置为 00。 00：不输出边沿 01：上升沿 10：下降沿 11：双边沿
Reserved	Bit 15-14	—	**保留**
SRCS	Bit 13-8	R/W	**输入源选择** 000000：无输入 000001：GPIO 000010：- 000011：ACMP 000100：DAC0 000101：- 000110：ADC0 000111：LVD 001000：UART0 001001：UART1 001010：UART2 001011：UART3 001100：UART4 001101：UART5 001110：SPI0 001111：SPI1 010000：I2C0 010001：I2C1 010010：AD16C4T0 010011：AD16C4T1 010100：GP32C4T0 010101：GP32C4T1 010110：RTC 010111：LP16T0 011000：LPUART0 011001：DMA 011010：ADC1 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=000000** MSIGS 无效 **SRCS=000001（SRC=GPIO）** 0000：PB0 0001：PB1 0010：PB2

			…… 1110：PB14 1111：PB15 **SRCS=000011（SRC=ACMP）** 0000：ACMP_OUT0 0001：ACMP_OUT1 **SRCS=000100（SRC=DAC0）** 0000：DAC0 通道 0 转换结束 0001：DAC0 通道 1 转换结束 **SRCS=000110（SRC=ADC0）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 **SRCS=000111（SRC=LVD）** LVD 比较器输出 **SRCS=001000，001001，001010，001011（SRC=UART0，UART1，UART2，UART3）** 0000：保留 0001：保留 0010：IrDA 输出电平 0011：RTS 输出 0100：TX 输出 0101：发送空状态中断脉冲 0110：接收数据中断脉冲 **SRCS=001100，001101（SRC=UART4，UART5）** 0000：接收数据中断脉冲 0001：发送空状态中断脉冲 0010：TX 输出 **SRCS=001110，001111（SRC=SPI0，SPI1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=010000，010001（I2C0，I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=010010，010011，010100，010101（SRC=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1）** 0000: 更新事件脉冲 0001: 触发事件脉冲 0010: 通道 1 输入捕获脉冲 0011: 通道 1 输出比较脉冲 0100: 通道 2 输入捕获脉冲 0101: 通道 2 输出比较脉冲

			0110: 通道 3 输入捕获脉冲 0111: 通道 3 输出比较脉冲 1000: 通道 4 输入捕获脉冲 1001: 通道 4 输出比较脉冲 **SRCS=010110（SRC=RTC）** 0000：亚秒、秒、分、时、日、月、年脉冲 0001：闹钟 A 和闹钟 B **SRCS=010111（SRC=LP16T0）** 0000：保留 0001：异步更新事件脉冲 **SRCS=011000（SRC=LPUART0）** 0000：接收缓冲器非空异步脉冲 0001：发送缓冲器空异步脉冲 0010：接收缓冲器非空同步脉冲 0011：发送缓冲器空同步脉冲 **SRCS=011001（SRC=DMA）** DMA 通道完成脉冲 **SRCS=011010（SRC=ADC1）** 0000：插入组转换结束 0001：标准转换组转换结束 0010：保留 注：ES32F36xx 不支持 LP16T、LPUART，ES32F336x 和 ES32F3936 系列不支持 ADC1

12. 5. 2. 17 PIS通道端口输出使能寄存器（PIS_CH_OER）

PIS 通道端口输出使能寄存器（PIS_CH_OER）

偏移地址：40_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												CH3OE	CH2OE	CH1OE	CH0OE

Reserved	Bit 31-4	—	保留
CH3OE	Bit 3	R/W	**PIS 通道 3 输出至端口使能** 0：输出到端口禁止 1：输出到端口使能
CH2OE	Bit 2	R/W	**PIS 通道 2 输出至端口使能** 0：输出到端口禁止 1：输出到端口使能
CH1OE	Bit 1	R/W	**PIS 通道 1 输出至端口使能** 0：输出到端口禁止 1：输出到端口使能
CH0OE	Bit 0	R/W	**PIS 通道 0 输出至端口使能** 0：输出到端口禁止 1：输出到端口使能

12. 5. 2. 18 PIS消费端通道控制寄存器 0（PIS_TAR_CON0）

PIS 消费端通道控制寄存器 0（PIS_TAR_CON0）

偏移地址：44_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				GP32C4T1_CH4IN_SEL	GP32C4T1_CH3IN_SEL	GP32C4T1_CH2IN_SEL	GP32C4T1_CH1IN_SEL	Reserved				GP32C4T0_CH4IN_SEL	GP32C4T0_CH3IN_SEL	GP32C4T0_CH2IN_SEL	GP32C4T0_CH1IN_SEL	Reserved			AD16C4T1_BRKIN_SEL	AD16C4T1_CH4IN_SEL	AD16C4T1_CH3IN_SEL	AD16C4T1_CH2IN_SEL	AD16C4T1_CH1IN_SEL	Reserved			AD16C4T0_BRKIN_SEL	AD16C4T0_CH4IN_SEL	AD16C4T0_CH3IN_SEL	AD16C4T0_CH2IN_SEL	AD16C4T0_CH1IN_SEL

Reserved	Bit 31-28	—	保留
GP32C4T1_CH4IN_SEL	Bit 27	R/W	**GP32C4T1 输入捕捉通道 4 输入选择** 0：从端口输入 1：将 PIS 通道 8 输出作为输入
GP32C4T1_CH3IN_SEL	Bit 26	R/W	**GP32C4T1 输入捕捉通道 3 输入选择** 0：从端口输入 1：将 PIS 通道 7 输出作为输入
GP32C4T1_CH2IN_SEL	Bit 25	R/W	**GP32C4T1 输入捕捉通道 2 输入选择** 0：从端口输入 1：将 PIS 通道 6 输出作为输入
GP32C4T1_CH1IN_SEL	Bit 24	R/W	**GP32C4T1 输入捕捉通道 1 输入选择** 0：从端口输入 1：将 PIS 通道 5 输出作为输入
Reserved	Bit 23-20	—	保留
GP32C4T0_CH4IN_SEL	Bit 19	R/W	**GP32C4T0 输入捕捉通道 4 输入选择** 0：从端口输入 1：将 PIS 通道 8 输出作为输入
GP32C4T0_CH3IN_SEL	Bit 18	R/W	**GP32C4T0 输入捕捉通道 3 输入选择** 0：从端口输入 1：将 PIS 通道 7 输出作为输入
GP32C4T0_CH2IN_SEL	Bit 17	R/W	**GP32C4T0 输入捕捉通道 2 输入选择** 0：从端口输入 1：将 PIS 通道 6 输出作为输入
GP32C4T0_CH1IN_SEL	Bit 16	R/W	**GP32C4T0 输入捕捉通道 1 输入选择** 0：从端口输入 1：将 PIS 通道 5 输出作为输入
Reserved	Bit 15-13	—	保留
AD16C4T1_BRKIN_SEL	Bit 12	R/W	**AD16C4T1 刹车输入选择** 0：从端口输入 1：将 PIS 通道 0 输出作为输入

AD16C4T1_CH4IN_SEL	Bit 11	R/W	**AD16C4T1 输入捕捉通道 4 输入选择** 0：从端口输入 1：将 PIS 通道 4 输出作为输入
AD16C4T1_CH3IN_SEL	Bit 10	R/W	**AD16C4T1 输入捕捉通道 3 输入选择** 0：从端口输入 1：将 PIS 通道 3 输出作为输入
AD16C4T1_CH2IN_SEL	Bit 9	R/W	**AD16C4T1 输入捕捉通道 2 输入选择** 0：从端口输入 1：将 PIS 通道 2 输出作为输入
AD16C4T1_CH1IN_SEL	Bit 8	R/W	**AD16C4T1 输入捕捉通道 1 输入选择** 0：从端口输入 1：将 PIS 通道 1 输出作为输入
Reserved	Bit 7-5	—	保留
AD16C4T0_BRKIN_SEL	Bit 4	R/W	**AD16C4T0 刹车输入选择** 0：从端口输入 1：将 PIS 通道 0 输出作为输入
AD16C4T0_CH4IN_SEL	Bit 3	R/W	**AD16C4T0 输入捕捉通道 4 输入选择** 0：从端口输入 1：将 PIS 通道 4 输出作为输入
AD16C4T0_CH3IN_SEL	Bit 2	R/W	**AD16C4T0 输入捕捉通道 3 输入选择** 0：从端口输入 1：将 PIS 通道 3 输出作为输入
AD16C4T0_CH2IN_SEL	Bit 1	R/W	**AD16C4T0 输入捕捉通道 2 输入选择** 0：从端口输入 1：将 PIS 通道 2 输出作为输入
AD16C4T0_CH1IN_SEL	Bit 0	R/W	**AD16C4T0 输入捕捉通道 1 输入选择** 0：从端口输入 1：将 PIS 通道 1 输出作为输入

12. 5. 2. 19 PIS消费端通道控制寄存器 1（PIS_TAR_CON1）

PIS 消费端通道控制寄存器 1（PIS_TAR_CON1）

偏移地址：48_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																SPI1_CLK_SEL	SPI1_RX_SEL	SPI0_CLK_SEL	SPI0_RX_SEL	Reserved			LPUART0_RXD_SEL	UART5_RXD_SEL	UART4_RXD_SEL	Reserved		UART3_RXD_SEL	UART2_RXD_SEL	UART1_RXD_SEL	UART0_RXD_SEL

Reserved	Bit 31-16	—	保留
SPI1_CLK_SEL	Bit 15	R/W	**SPI1 CLK 输入选择** 0：从端口输入 1：将 PIS 通道 8 输出作为输入
SPI1_RX_SEL	Bit 14	R/W	**SPI1 RX 输入选择** 0：从端口输入 1：将 PIS 通道 7 输出作为输入
SPI0_CLK_SEL	Bit 13	R/W	**SPI0 CLK 输入选择** 0：从端口输入 1：将 PIS 通道 6 输出作为输入
SPI0_RX_SEL	Bit 12	R/W	**SPI0 RX 输入选择** 0：从端口输入 1：将 PIS 通道 5 输出作为输入
Reserved	Bit 11-9	—	保留
LPUART0_RXD_SEL	Bit 8	R/W	**LPUART0 RXD 输入选择** 0：从端口 RXD 输入 1：将 PIS 通道 15 输出作为输入 注：ES32F36xx 不支持 LPUART
UART5_RXD_SEL	Bit 7	R/W	**UART5 RXD 输入选择** 0：从端口 RXD 输入 1：将 PIS 通道 14 输出作为输入
UART4_RXD_SEL	Bit 6	R/W	**UART4 RXD 输入选择** 0：从端口 RXD 输入 1：将 PIS 通道 13 输出作为输入
Reserved	Bit 5-4	—	保留
UART3_RXD_SEL	Bit 3	R/W	**UART3 RXD 输入选择** 0：从端口 RXD 输入 1：将 PIS 通道 12 输出作为输入
UART2_RXD_SEL	Bit 2	R/W	**UART2 RXD 输入选择** 0：从端口 RXD 输入 1：将 PIS 通道 11 输出作为输入

UART1_RXD_SEL	Bit 1	R/W	**UART1 RXD 输入选择** 0：从端口 RXD 输入 1：将 PIS 通道 10 输出作为输入
UART0_RXD_SEL	Bit 0	R/W	**UART0 RXD 输入选择** 0：从端口 RXD 输入 1：将 PIS 通道 9 输出作为输入

12. 5. 2. 20 UART0 输出调制控制寄存器（UART0_TXMCR）

UART0 输出调制控制寄存器（UART0_TXMCR）

偏移地址：60_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-9	—	保留
TXMLVLS	Bit 8	R/W	**TX 调制电平选择** 0：低电平调制（TX 与所选取的调制信号进行硬件或操作） 1：高电平调制（TX 与所选取的调制信号进行硬件与操作）
TXMSS	Bit 7-4	R/W	**TX 调制源选择** 0000：调制禁止 0001：AD16C4T0 0010：AD16C4T1 0011：GP32C4T0 0100：GP32C4T1 0101：GP16C4T0 0110：GP16C4T1 0111：LP16T0 1000：BUZ 其余：无调制 注：ES32F36xx 不支持 LP16T
TXSIGS	Bit 3-0	R/W	**TX 调制信号选择** **TXMSS=0000** TXSIGS 无效 **TXMSS=0001，0010，0011，0100，0101，0110（调制源=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：TIMER 通道 1 0001：TIMER 通道 2 0010：TIMER 通道 3 0011：TIMER 通道 4 **TXMSS=0111（调制源=LP16T0）** 调制信号为 LP16T0 输出 **TXMSS=1000（调制源=BUZ）** 调制信号为 BUZ 输出 注：ES32F36xx 不支持 LP16T

12. 5. 2. 21 UART1 输出调制控制寄存器（UART1_TXMCR）

UART1 输出调制控制寄存器（UART1_TXMCR）

偏移地址：64_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-9	—	**保留**
TXMLVLS	Bit 8	R/W	**TX 调制电平选择** 0：低电平调制（TX 与所选取的调制信号进行硬件或操作） 1：高电平调制（TX 与所选取的调制信号进行硬件与操作）
TXMSS	Bit 7-4	R/W	**TX 调制源选择** 0000：调制禁止 0001：AD16C4T0 0010：AD16C4T1 0011：GP32C4T0 0100：GP32C4T1 0101：GP16C4T0 0110：GP16C4T1 0111：LP16T0 1000：BUZ 其余：无调制 注：ES32F36xx 不支持 LP16T
TXSIGS	Bit 3-0	R/W	**TX 调制信号选择** **TXMSS=0000** TXSIGS 无效 **TXMSS=0001，0010，0011，0100，0101，0110（调制源=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：TIMER 通道 1 0001：TIMER 通道 2 0010：TIMER 通道 3 0011：TIMER 通道 4 **TXMSS=0111（调制源=LP16T0）** 调制信号为 LP16T0 输出 **TXMSS=1000（调制源=BUZ）** 调制信号为 BUZ 输出 注：ES32F36xx 不支持 LP16T

12. 5. 2. 22 UART2 输出调制控制寄存器（UART2_TXMCR）

UART2 输出调制控制寄存器（UART2_TXMCR）

偏移地址：68_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-9	—	**保留**
TXMLVLS	Bit 8	R/W	**TX 调制电平选择** 0：低电平调制（TX 与所选取的调制信号进行硬件或操作） 1：高电平调制（TX 与所选取的调制信号进行硬件与操作）
TXMSS	Bit 7-4	R/W	**TX 调制源选择** 0000：调制禁止 0001：AD16C4T0 0010：AD16C4T1 0011：GP32C4T0 0100：GP32C4T1 0101：GP16C4T0 0110：GP16C4T1 0111：LP16T0 1000：BUZ 其余：无调制 注：ES32F36xx 不支持 LP16T
TXSIGS	Bit 3-0	R/W	**TX 调制信号选择** **TXMSS=0000** TXSIGS 无效 **TXMSS=0001，0010，0011，0100，0101，0110（调制源=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：TIMER 通道 1 0001：TIMER 通道 2 0010：TIMER 通道 3 0011：TIMER 通道 4 **TXMSS=0111（调制源=LP16T0）** 调制信号为 LP16T0 输出 **TXMSS=1000（调制源=BUZ）** 调制信号为 BUZ 输出 注：ES32F36xx 不支持 LP16T

12. 5. 2. 23 UART3 输出调制控制寄存器（UART3_TXMCR）

UART3 输出调制控制寄存器（UART3_TXMCR）

偏移地址：$6C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-9	—	保留
TXMLVLS	Bit 8	R/W	**TX 调制电平选择** 0：低电平调制（TX 与所选取的调制信号进行硬件或操作） 1：高电平调制（TX 与所选取的调制信号进行硬件与操作）
TXMSS	Bit 7-4	R/W	**TX 调制源选择** 0000：调制禁止 0001：AD16C4T0 0010：AD16C4T1 0011：GP32C4T0 0100：GP32C4T1 0101：GP16C4T0 0110：GP16C4T1 0111：LP16T0 1000：BUZ 其余：无调制 注：ES32F36xx 不支持 LP16T
TXSIGS	Bit 3-0	R/W	**TX 调制信号选择** **TXMSS=0000** TXSIGS 无效 **TXMSS=0001，0010，0011，0100，0101，0110（调制源=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：TIMER 通道 1 0001：TIMER 通道 2 0010：TIMER 通道 3 0011：TIMER 通道 4 **TXMSS=0111（调制源=LP16T0）** 调制信号为 LP16T0 输出 **TXMSS=1000（调制源=BUZ）** 调制信号为 BUZ 输出 注：ES32F36xx 不支持 LP16T

12. 5. 2. 24 LPUART0 输出调制控制寄存器（LPUART0_TXMCR）

LPUART0 输出调制控制寄存器（LPUART0_TXMCR）

偏移地址：70_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-9	—	保留
TXMLVLS	Bit 8	R/W	**TX 调制电平选择** 0：低电平调制（TX 与所选取的调制信号进行硬件或操作） 1：高电平调制（TX 与所选取的调制信号进行硬件与操作）
TXMSS	Bit 7-4	R/W	**TX 调制源选择** 0000：调制禁止 0001：AD16C4T0 0010：AD16C4T1 0011：GP32C4T0 0100：GP32C4T1 0101：GP16C4T0 0110：GP16C4T1 0111：LP16T0 1000：BUZ 其余：无调制 注：ES32F36xx 不支持 LP16T、LPUART
TXSIGS	Bit 3-0	R/W	**TX 调制信号选择** **TXMSS=0000** TXSIGS 无效 **TXMSS=0001，0010，0011，0100，0101，0110（调制源=AD16C4T0，AD16C4T1，GP32C4T0，GP32C4T1，GP16C4T0，GP16C4T1）** 0000：TIMER 通道 1 0001：TIMER 通道 2 0010：TIMER 通道 3 0011：TIMER 通道 4 **TXMSS=0111（调制源=LP16T0）** 调制信号为 LP16T0 输出 **TXMSS=1000（调制源=BUZ）** 调制信号为 BUZ 输出 注：ES32F36xx 不支持 LP16T、LPUART

第13章 独立看门狗（IWDT）

13. 1 概述

独立看门狗可用来检测软件和硬件异常引起的故障，如主时钟停振、用户程序异常无法喂狗等，当计数器达到给定的超时值时，将产生系统复位。

当硬件使能独立看门狗时，IWDT 时钟强制变为独立的 32.768KHz 的 LRC 时钟，且用户无法通过软件来关闭。

独立看门狗最适合独立于主程序之外，并且对时间精度要求较低的场合。

13. 2 特性

- 支持硬件使能和关闭
  - 配置字中的 IWDTEN 位配置为 1，则硬件使能 IWDT
  - 配置字中的 IWDTEN 位配置为 0，则硬件关闭 IWDT，但可以软件使能 IWDT
  - 硬件使能后，不可通过软件关停
  - 硬件使能后，IWDT 时钟强制变为 32.768KHz 的 LRC 时钟
- 溢出时间可设定
  - 写入 IWDT_LOAD 寄存器将重新加载看门狗
  - 溢出时产生 IWDT 复位
- 中断可唤醒 STOP1、STOP2，不能唤醒 Standby 模式

13. 3 功能描述

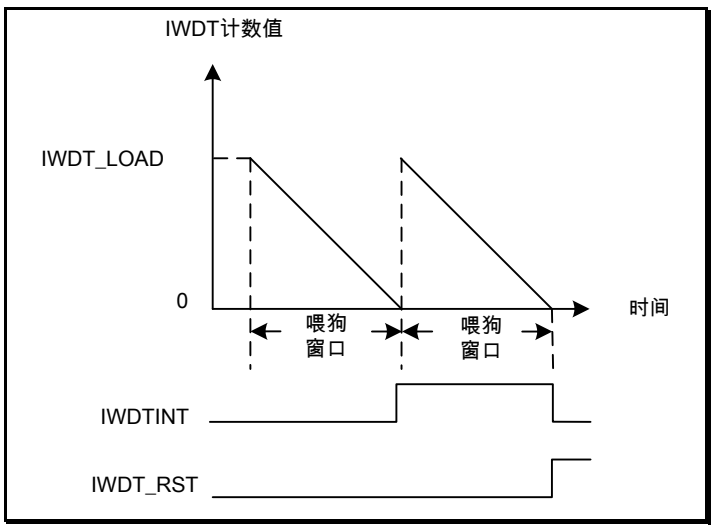


图 13-1 独立看门狗时序图

13.3.1 硬件看门狗

IWDT 可用于检测软件和硬件异常，用户可通过使能配置字中的 IWDTEN 配置位启用硬件看门狗功能，以提高系统健壮性；硬件看门狗使能后，时钟强制变为 32.768KHz 的 LRC 时钟，即使系统时钟失效，IWDT 仍可正常工作。

当硬件看门狗使能时，在系统上电后看门狗立即运行（时钟固定为 32.768KHz LRC 时钟）；IWDT 载入 IWDT_LOAD 寄存器的默认值 0x8000（约 1s），并从该值开始递减计数。

计数器计数到 0 时，IWDT 产生中断标志，并在下一计数时钟到来时，计数器再次载入 IWDT_LOAD 的设定值，并继续递减计数；当计数器再次计数到 0 时，如果 IWDT 中断标志未被清零，IWDT 将产生复位信号。

IWDT_INTCLR 寄存器写入任意值，则清除中断标志位，并重新载入计数初值，进行递减计数。

操作流程

1. 开启 IWDT_IRQn 中断服务，并使能 IWDT 中断（IWDT_CON.IE=1）；
2. 修改 IWDT_LOAD.LOAD 值，以确定喂狗间隔；
3. 在中断服务中检查中断标志位（IWDT_RIS.WDTIF）是否被置起；
4. 如果中断标志位置起，则执行喂狗操作；

13.3.2 软件看门狗

当硬件看门狗禁止时，系统上电看门狗不运行，但可通过软件配置 IWDT_CON.EN=1 使能看门狗，俗称软件看门狗。

当软件看门狗使能时，计数器载入 IWDT_LOAD 的设定值，并开始递减计数；当计数到 0 时，IWDT 产生中断标志，并在下一个计数时钟到来时，计数器再次载入 IWDT_LOAD 的设定值，并继续递减计数；当计数器再次计数到 0 时，如果 IWDT 中断标志未被清除，IWDT 将产生复位信号。

IWDT_INTCLR 写入任意值，则清除中断标志位，并重新载入计数初值，进行递减计数。

操作流程

1. 开启 IWDT 中断服务，并使能 IWDT 中断（IWDT_CON.IE=1）；
2. 修改 IWDT_LOAD.LOAD 值，以确定喂狗间隔；
3. 配置 IWDT_CON.CLKS，选择 IWDT 时钟源；
4. 配置 IWDT_CON.EN=1，使能软件看门狗；
5. 在中断服务中检查中断标志位（IWDT_RIS.WDTIF）是否被置起；
6. 如果中断标志位置起，则执行喂狗操作；

注：软件看门狗可以通过配置 IWDT_CON.CLKS 选择时钟源，系统提供了 LRC 和 PCLK2 两种时钟源供选择；

13. 3. 3 调试模式

当系统进入调试模式时，通过配置 DBG_APB2FZ .IWDT_STOP，可以在内核停止时暂停 IWDT 计数，以便查询系统内部状态，查询完成后，恢复程序运行。

13. 3. 4 寄存器访问保护

IWDT_LOAD、IWDT_CON、IWDT_INTCLR 这三个寄存器具有写保护功能，对其修改时，必须先向 IWDT_LOCK 寄存器写入 0x1ACCE551 来移除写保护；当 IWDT_LOCK 写入其他任意值时，寄存器重新被保护。

13. 4 特殊功能寄存器

13. 4. 1 寄存器列表

IWDT 寄存器列表		
名称	偏移地址	描述
IWDT_LOAD	0000_H	IWDT 计数器装载值寄存器
IWDT_VALUE	0004_H	IWDT 计数器当前值寄存器
IWDT_CON	0008_H	IWDT 控制寄存器
IWDT_INTCLR	$000C_H$	IWDT 中断标志清除寄存器
IWDT_RIS	0010_H	IWDT 中断标志寄存器
IWDT_LOCK	0100_H	IWDT 锁定寄存器

13. 4. 2 寄存器描述

13. 4. 2. 1 IWDT计数器装载值寄存器（IWDT_LOAD）

IWDT 计数器装载值寄存器（IWDT_LOAD）																															
偏移地址：00_H																															
复位值：00000000_00000000_10000000_00000000$_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD																															

LOAD	Bit 31-0	R/W	**IWDT 计数器重载值** 计数范围 0x0000_0001~0xFFFF_FFFF。如果为 0，IWDT 不计数。

13. 4. 2. 2 IWDT计数器当前值寄存器（IWDT_VALUE）

IWDT 计数器当前值寄存器（IWDT_VALUE）																															
偏移地址：04_H																															
复位值：11111111_ 11111111_ 11111111 _ 11111111$_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE																															

VALUE	Bit 31-0	R	**IWDT 计数器当前值** 读取时返回 IWDT 计数器的当前计数值

13. 4. 2. 3 IWDT控制寄存器（IWDT_CON）

IWDT 控制寄存器（IWDT_CON）

偏移地址：08_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-4	—	保留
CLKS	Bit 3	R/W	**IWDT 计数时钟选择位** 0：PCLK2 1：32KHz LRC 注：时钟源配置参考时钟分配说明
RSTEN	Bit 2	R/W	**IWDT 复位使能位** 0：禁止 1：使能，IWDT 计数到 0 时，产生复位信号，将芯片复位
IE	Bit 1	R/W	**IWDT 中断使能位** 0：禁止 1：使能，IWDT 计数到 0 时，产生中断标志
EN	Bit 0	R/W	**IWDT 模块使能位** 0：禁止 1：使能

13. 4. 2. 4 IWDT中断标志清除寄存器（IWDT_INTCLR）

IWDT 中断标志清除寄存器（IWDT_INTCLR）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

INTCLR	Bit 31-0	W	**IWDT 中断标志清 0 位** 对 IWDT_INTCLR 寄存器进行任意写操作，IWDT 中断标志位均被清零，计数器重载 IWDT_LOAD 寄存器值，继续递减计数

13. 4. 2. 5 IWDT中断标志寄存器（IWDT_RIS）

IWDT 中断标志寄存器（IWDT_RIS）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-1	—	保留
WDTIF	Bit 0	R	**IWDT 中断标志位** 0：未产生中断 1：IWDT 计数器计数到 0 时，产生中断 写寄存器 IWDT_INTCLR，可清除 IWDT 中断标志位

13. 4. 2. 6 IWDT锁定寄存器（IWDT_LOCK）

IWDT 锁定寄存器（IWDT_LOCK）

偏移地址：100_H

复位值：$00000000\_00000000\_00000000\_00000001_B$

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

Reserved	Bit 31-1	—	保留
LOCK	Bit 0	R/W	**IWDT 寄存器保护状态位** 0：IWDT 寄存器处于未保护状态 1：IWDT 寄存器处于保护状态 对 IWDT_LOCK 寄存器写入 0x1ACCE551，被保护的寄存器处于未保护状态；写入其它值，处于保护状态

第14章 窗口看门狗（WWDT）

14. 1 概述

窗口看门狗通常用来监视由外部干扰或不可预见的逻辑条件造成的应用程序背离运行序列而产生的软件故障。

窗口看门狗对于过早或过晚喂狗都将产生 WWDT 复位，可用于检测软件没有喂狗或在喂狗禁止区内喂狗，防止程序运行至不可控状态。

14. 2 特性

- 可编程的递减计数器
- 支持设定喂狗禁止区
  - 通过配置 WWDT_CON.WWDTWIN 设置喂狗禁止区
  - 窗口外喂狗产生 WWDT 复位
  - 窗口内产生 WWDT 中断
- 安全可靠
  - 当配置字中的 WWDTEN 为 1 时，一旦软件使能，则只能通过复位关断
- 计数长度可设定
  - 可通过配置 WWDT_LOAD 寄存器设定计数长度
  - 下溢时产生 WWDT 复位
- 中断可用作喂狗请求，可唤醒 STOP1 和 STOP2 模式

14. 3 功能描述

14. 3. 1 窗口看门狗

对于窗口看门狗，过早或过晚喂狗都将产生 WWDT 复位，可用于检测软件的错误喂狗行为，防止程序运行至不可控状态，可通过 WWDT 复位消除不可控状态。如中断异常时，程序不断进入一个带喂狗指令的子程序的情况。

用户可根据程序正常执行的时间设定喂狗窗口，可检测到程序未按正常次序执行，跳过某些程序的异常情况。

窗口看门狗时钟源

窗口看门狗对时间窗口有一定的要求，系统内提供了 LRC 和 PCLK2 作为 WWDT 时钟，用户可根据自己喂狗精度的需要，通过配置 WWDT_CON.CLKS 来选择合适的时钟源。

操作流程

- 系统上电后，窗口看门狗不启动，配置 WWDT_LOAD 寄存器设置计数初值，配置 WWDT_CON.CLKS 选择时钟源，配置 WWDT_CON.IE 使能中断，配置 WWDT_CON.EN 使能窗口看门狗；
- WWDT 计数器载入 WWDT_LOAD 寄存器值的 1/4，并开始递减计数，当计数到 0 时，窗口计数器加 1；
- 在下一个计数时钟到来时，计数器再次载入 WWDT_LOAD 寄存器值的 1/4，并开始递减计数，当计数到 0 时，窗口计数器加 1，重复此过程直到窗口计数器计到 4；
- 若 WWDT_CON.WWDTWIN 设置为 00，则窗口计数器计到 1 时，WWDT 产生中断标志；
- 若 WWDT_CON.WWDTWIN 设置为 01，则窗口计数器计到 2 时，WWDT 产生中断标志；
- 若 WWDT_CON.WWDTWIN 设置为 10，则窗口计数器计到 3 时，WWDT 产生中断标志；
- 若 WWDT 产生中断后，直至窗口计数器计数到 4 之前（即累计计数等于 WWDT_LOAD 寄存器的值），如果没有在喂狗窗口内进行喂狗，则 WWDT 模块将产生复位信号，如下图所示；
- 若喂狗窗口内寄存器 WWDT_INTCLR 写入任意值，将清除中断标志位，WWDT 计数器重新载入计数初值并进行递减计数；
- 若在喂狗禁止区内进行喂狗操作，则会产生 WWDT 复位，如下图所示。

注：若配置字中的 WWDTEN 位配置为 1，则软件使能窗口看门狗之后，不可再通过软件关闭窗口看门狗。

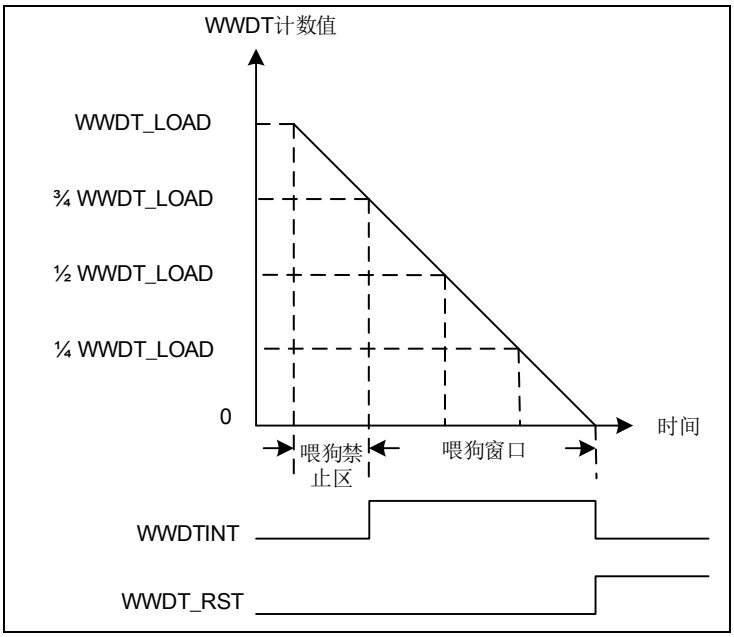


图 14-1 窗口看门狗中断和下溢复位产生时序图（WWDTWIN 设定为 00）

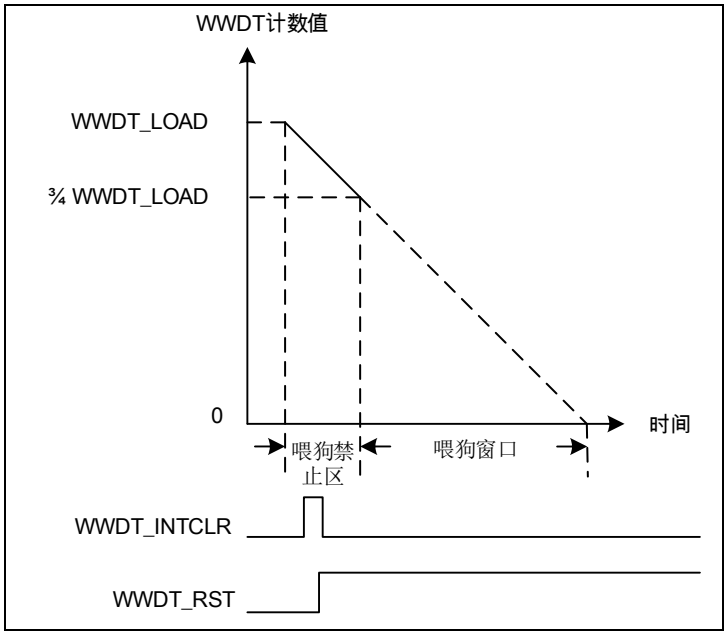


图 14-2 错误的喂狗时序图（WWDTWIN 设定为 00）

14. 3. 2 调试模式

当系统进入调试模式时，通过配置 DBG_APB2FZ.WWDT_STOP，可以在内核停止时，暂停 WWDT 计数，以便查询系统内部状态，查询完成后，恢复程序运行。

14. 3. 3 寄存器访问保护

WWDT_LOAD、WWDT_CON、WWDT_INTCLR 这三个寄存器具有写保护功能，对其修改时，必须先向 WWDT_LOCK 寄存器写入 0x1ACCE551 来移除写保护；当 WWDT_LOCK 写入其它任意值时，寄存器重新被保护。

14. 4 特殊功能寄存器

14. 4. 1 寄存器列表

WWDT 寄存器列表		
名称	偏移地址	描述
WWDT_LOAD	0000_H	WWDT 计数器装载值寄存器
WWDT_VALUE	0004_H	WWDT 计数器当前值寄存器
WWDT_CON	0008_H	WWDT 控制寄存器
WWDT_INTCLR	$000C_H$	WWDT 中断标志清除寄存器
WWDT_RIS	0010_H	WWDT 中断标志寄存器
WWDT_LOCK	0100_H	WWDT 锁定寄存器

14. 4. 2 寄存器描述

14. 4. 2. 1 WWDT计数器装载值寄存器（WWDT_LOAD）

WWDT 计数器装载值寄存器（WWDT_LOAD）																															
偏移地址：00_H																															
复位值：$00000000\_00000010\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD																															

LOAD	Bit 31-0	R/W	**WWDT 计数器重载值** 计数范围 0x0000_0001~0xFFFF_FFFF。如果为 0，WWDT 不计数。

14. 4. 2. 2 WWDT计数器当前值寄存器（WWDT_VALUE）

WWDT 计数器当前值寄存器（WWDT_VALUE）																															
偏移地址：04_H																															
复位值：$00111111\_\ 11111111\_\ 11111111\ \_\ 11111111_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE																															

VALUE	Bit 31-0	R	**WWDT 计数器当前值** 读取时返回 WWDT 计数器的当前计数值，其中高两位为窗口计数器当前值

14. 4. 2. 3 WWDT控制寄存器（WWDT_CON）

WWDT 控制寄存器（WWDT_CON）

偏移地址：08_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-6	—	保留
WWDTWIN	Bit 5-4	R/W	**WWDT 禁止喂狗窗口选择位** 00：25%窗口内禁止喂狗，窗口内喂狗产生复位 01：50%窗口内禁止喂狗，窗口内喂狗产生复位 10：75%窗口内禁止喂狗，窗口内喂狗产生复位 11：不禁止喂狗，喂狗将使看门狗计数器重载
CLKS	Bit 3	R/W	**WWDT 计数时钟选择位** 0：PCLK2 1：LRC 时钟（32768Hz）
RSTEN	Bit 2	R/W	**WWDT 复位使能位** 0：禁止 1：使能，WWDT 计数到 0 时，产生复位信号，将芯片复位
IE	Bit 1	R/W	**WWDT 中断使能位** 0：禁止 1：使能，WWDT 计数到 0 时，产生中断标志
EN	Bit 0	R/W	**WWDT 模块使能位** 0：禁止 1：使能

14. 4. 2. 4 WWDT中断标志清除寄存器（WWDT_INTCLR）

WWDT 中断标志清除寄存器（WWDT_INTCLR）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

INTCLR	Bit 31-0	W	**WWDT 中断标志清 0 位** 对 WWDT_INTCLR 寄存器进行任意写操作，WWDT 中断标志位均被清零，计数器重载 WWDT_LOAD 寄存器值，继续递减计数

14. 4. 2. 5 WWDT中断标志寄存器（WWDT_RIS）

WWDT 中断标志寄存器（WWDT_RIS）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-1	—	**保留**
WWDTIF	Bit 0	R	**WWDT 中断标志位** 0：未产生中断 1：WWDT 计数器计数到 0 时，产生中断 写寄存器 WWDT_INTCLR，可清除 WWDT 中断标志位

14. 4. 2. 6 WWDT锁定寄存器（WWDT_LOCK）

WWDT 锁定寄存器（WWDT_LOCK）

偏移地址：100_H

复位值：$00000000\_00000000\_00000000\_00000001_B$

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

Reserved	Bit 31-1	—	保留
LOCK	Bit 0	R/W	**WWDT 寄存器保护状态位** 0：WWDT 寄存器处于未保护状态 1：WWDT 寄存器处于保护状态 对 WWDT_LOCK 寄存器写入 0x1ACCE551，被保护的寄存器处于未保护状态；写入其它值，处于保护状态

第15章 通用端口及端口控制（GPIO）

15. 1 概述

每组通用端口包含 16 个独立的引脚。这些引脚可单独配置为输入或输出，每个引脚输出输入功能中可配置为开漏输出、推挽输出以及浮空输入、带滤波输入模式，配置为输出模式时还可选择每个引脚的驱动强度，拉电流和灌电流驱动能力分别可配。

GPIO 引脚可复用为外设功能端口，例如 PWM 输出口、UART 通信口或模拟输入，每个外设均支持复用到多个引脚上。GPIO 端口支持最多 16 个异步外部中断，可被配置到任何一个 IO 引脚上。并且 GPIO 端口支持通过 PIS 触发其他外设。

15. 2 特性

- 可配置为输入或输出
- 输出模式可配置
  - 推挽/开漏
  - 上拉/下拉
- 输入模式
  - 端口浮空
  - 上拉/下拉
  - 模拟端口
- 支持端口输出数据的复位、置位或取反，可按位操作
- 支持复用为外设功能端口
- 推挽输出 NMOS 和 PMOS 驱动能力可分别配置：四种驱动能力选择
- 支持 16 个外部输入中断
- 支持端口配置写保护功能

15. 3 结构框图

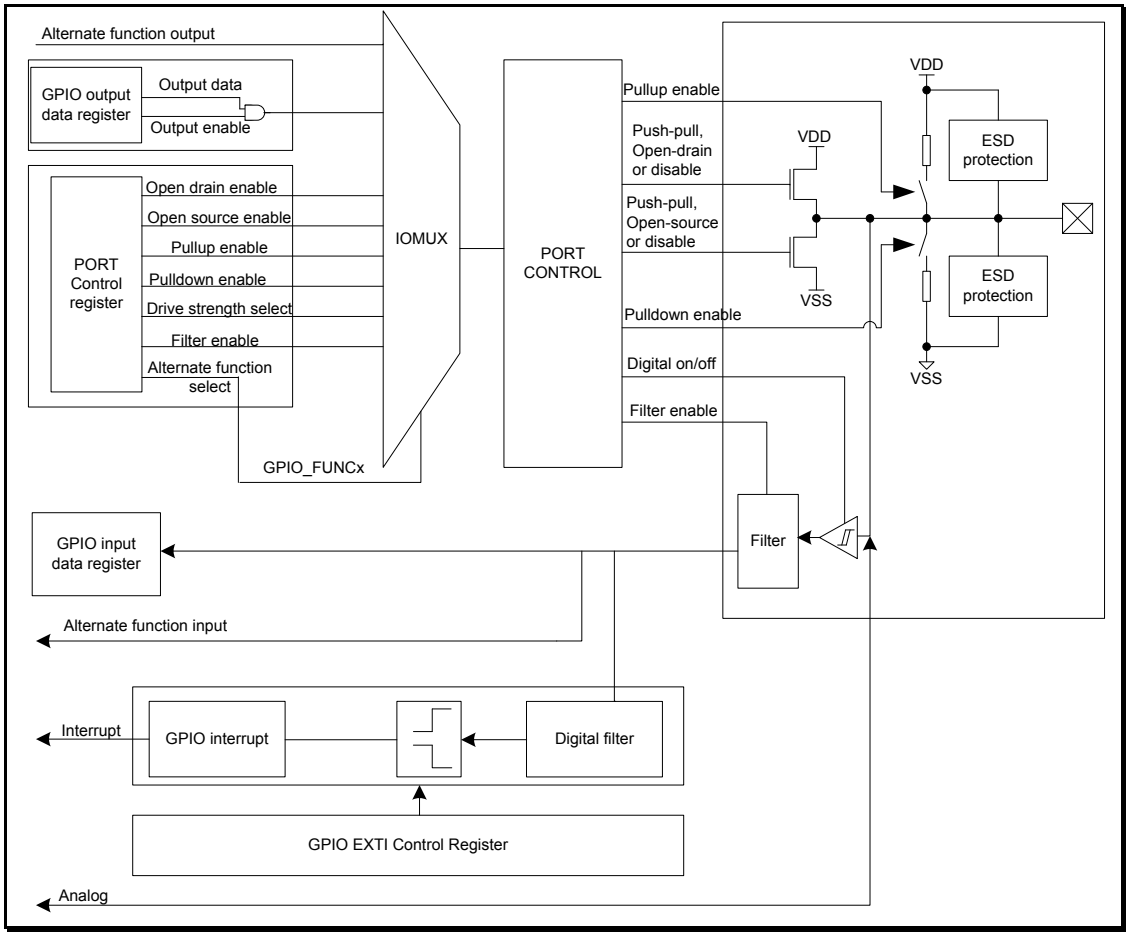


图 15-1 GPIO 结构框图

15.4 功能描述

15.4.1 端口控制寄存器

每个 GPIO 有 16 个相对独立的引脚，每个引脚都可以通过寄存器自由配置。

通过配置 GPIO_MODE，可选择相应端口的模式，可配置为输入模式，输出模式和端口关闭模式。选择输出模式时，端口状态可以通过 GPIO_DIN 寄存器读取。选择端口关闭模式时，相应端口可被复用为模拟功能。

通过配置 GPIO_PUPD，可使能相应端口的上拉或/和下拉电阻。

通过配置 GPIO_ODOS，可将相应端口输出方式配置为推挽、源极开路和漏极开路三种模式。在配置为源极开路或漏极开路模式时，可使能相应端口的上拉或下拉，以保证正常输出，也可在端口外部连接上拉或下拉电阻。

通过分别配置 GPIO_PODRV（拉电流）和 GPIO_NODRV（灌电流），可选择相应端口的输出驱动能力，以满足不同的负载要求。

通过配置 GPIO_FLT，可使能相应端口的输入滤波功能，可滤除外部引线上高频信号干扰或毛刺。若输入需要较高的实时性，建议关闭输入滤波功能。

通过配置 GPIO_TYPE，可选择相应端口的输入类型，可选择 TTL 或 CMOS 两种模式。

通过配置 GPIO_FUNC，可选择相应 GPIO 的复用功能，可选择 FUNC_ALT0 ~ FUNC_ALT7（细节可参考对应产品数据手册的管脚功能定义）。复用功能 GPIO 为输入时必须配置为输入模式，且需外部驱动；复用功能 GPIO 为输出时必须配置为输出模式，推挽或开漏；复用功能 GPIO 为双向模式时，端口必须配置为输出模式，推挽或开漏。

通过配置 GPIO_LOCK，可锁定相应端口的控制寄存器数值。直到下一次 CPU 复位锁定才可被解除。端口数据寄存器不受锁定的控制。

15.4.2 端口数据寄存器

软件可通过读取 GPIO_DIN 来获知端口的电平状态，若相应端口输入滤波被使能，则读到的是端口滤波之后的状态。

通过配置 GPIO_DOUT，可选择端口输出电平值，若端口模式已配置为输出，则该值所对应的电平会在管脚上立即生效。

通过配置 GPIO_BSRR，可按位改写端口输出电平值。对置位寄存器某些位进行写入 1 可置位相应端口，写入 0 的位不会影响相应端口的输出电平。对复位寄存器某些位进行写入 1 可复位相应端口，写入 0 的位不会影响相应端口的输出电平。若同时将某位置位和复位，则置位的优先级更高。

通过配置 GPIO_BIR，可按位翻转端口输出电平值。对翻转寄存器某些位进行写入 1 可将相应端口电平值翻转，写入 0 的位不会影响相应端口的输出电平。

15. 4. 3 外部端口中断

通过配置 GPIO_EXTIRER，可设置外部中断上升沿触发使能或禁止，其中低 16bit 为上升沿中断触发使能位，高 16bit 为保留位。

通过配置 GPIO_EXTIFER，可设置外部中断下降沿触发使能或禁止，其中低 16bit 为下降沿中断触发使能位，高 16bit 为保留位。

通过配置 GPIO_EXTIEN，可设置外部中断使能或禁止，其中低 16bit 为中断使能设置位，外部中断对应 bit 位配置为 1 表示中断使能，配置为 0 表示中断禁止，高 16bit 为保留位。

通过 GPIO_EXTIFLAG 寄存器，可检测当前有效中断。外部中断对应 bit 位为 1 表示检测到有效中断，为 0 表示未检测到有效中断。该寄存器为只读。

通过配置 GPIO_EXTISFR，可将外部端口中断标志位置位。其中低 16bit 为中断标志位置位位，为 1 表示置位中断标志位，为 0 无操作，高 16bit 为保留位。

通过配置 GPIO_EXTICFR，可将外部端口中断标志位清除。其中低 16bit 为中断标志位清除位，为 1 表示清零中断标志位，为 0 无操作，高 16bit 为保留位

通过配置 GPIO_EXTIPSR0 和 GPIO_EXTIPSR1 可选择外部中断的 GPIO 端口，每个外部中断可选择 8 个不同的 GPIO 口（PAn~PHn）。

通过配置 GPIO_EXTIFLTCR，可设置外部中断滤波参数。GPIO_EXTIFLTCR.FLTEN 位可使能相应外部端口中断滤波功能，配置 GPIO_EXTIFLTCR.FLTSEL 位可设置滤波时间，配置 GPIO_EXTIFLTCR.FLTCKS 位可选滤波时钟。

15. 4. 4 通用GPIO配置

每个 GPIO 都可以由软件设置为输出模式或输入模式，也可以复用为外设功能接口。所有 GPIO 端口都有内部上拉或下拉，应用中可以激活也可以断开。

端口配置表				
配置模式		MODEn[1:0]	PUPDn[1:0]	ODOSn[1:0]
输出	推挽输出	1x	xx	0x
	PMOS 开漏输出	1x	xx	10
	NMOS 开漏输出	1x	xx	11
输入	浮空输入	01	00	xx
	上拉输入	01	01	xx
	下拉输入	01	10	xx
	模拟输入	01	禁止	xx

表 15-1 端口配置表

在输出模式中，还可以软件方式设置每个 GPIO 的驱动能力。输入模式中，可以软件方式设置每个 GPIO 的滤波特性。还可以软件选择 GPIO 管脚类型，有 CMOS 和 TTL 两种选择。

ODRVn[1:0]	意义
00	驱动 level 1（0.5mA）
01	驱动 level 2（10mA）
10	驱动 level 3（30mA）
11	驱动 level 4（30mA）

表 15-2 端口拉电流驱动表

ODRVn[1:0]	意义
00	驱动 level 1（0.5mA）
01	驱动 level 2（10mA）
10	驱动 level 3（50mA）
11	驱动 level 4（120mA）

表 15-3 端口灌电流驱动表

15. 4. 5 外部中断与唤醒

外部中断可以配到每个 GPIO，且配置为外部中断的 GPIO 必须设置为输入模式。

每个外部中断通道可以独立的配置输入类型和对应的触发事件（上升沿触发、下降沿触发或双边触发），都可以独立的设置触发或禁止。

EXTI 主要特性如下：

- ◇ 每个外部中断都有独立的触发或禁止设置
- ◇ 每个中断通道有独立的状态标识位
- ◇ 支持多达 16 个外部中断请求
- ◇ 独立设置外部中断滤波特性

外部中断映像如下：

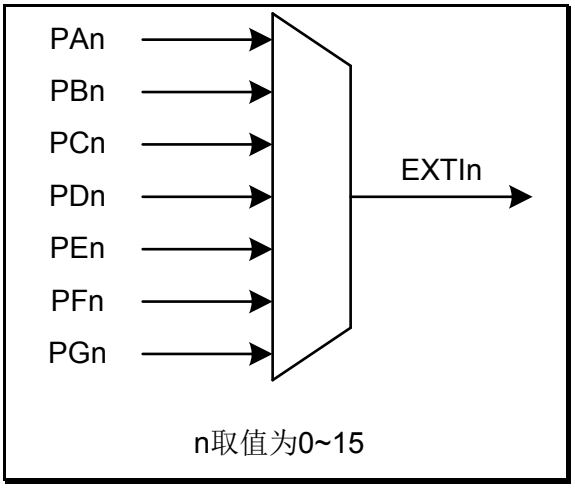


图 15-2 外中断 GPIO 映像

应用说明：

要产生中断，必须配置好中断端口并使能，然后根据需要通过触发使能寄存器来设置边沿检测，另外根据应用需求可以设置输入滤波。当外部端口发生了预期的边沿时将产生一个中断请求，外部中断标志寄存器中对应的位随之被置 1，此时在外部中断标志清零寄存器对应标志位置 1 可以清除中断请求及对应标志位。

外部中断配置步骤如下：

1. 将对应 GPIO 端口配置为输入模式。
2. 配置外部中断端口选择寄存器 0 和外部中断端口选择寄存器 1 选择相应的 GPIO 端口。
3. 配置外部中断上升沿触发使能寄存器设置上升沿触发事件，配置外部中断下降沿触发使能寄存器设置下降沿触发事件。
4. 配置外部中断滤波控制寄存器设置需求对应的滤波特性。
5. 配置外部中断使能寄存器对应的中断使能位，使得外部中断可以有效响应。

15. 4. 6 外设功能端口复用

使用复用功能时必须先对 GPIO 端口复用功能寄存器 0 或 GPIO 端口复用功能寄存器 1 进行配置，对应的配置参数请查看数据手册管脚功能复用表。

- ◇ 当复用功能为输入时，端口必须配置为输入模式（浮空、上拉或下拉），且输入引脚由外部驱动。
- ◇ 当复用功能为输出时，端口必须配置为输出模式（推挽、开漏）。在配置开漏模式时，GPIO 输入功能有效，此时可以用作双向模式。

如果端口复用功能为输出，则引脚会和片上外设的输出信号连接，如果对应外设没有被激活，GPIO 的输出信号将不确定。

15. 4. 7 GPIO锁定

芯片 GPIO 锁定机制的处理是将端口配置冻结。当一个端口执行了锁定程序后，对应 GPIO 控制寄存器数值将被锁定，在下一次复位之前，不能再被更改。

15. 4. 8 GPIO输入配置

当 GPIO 端口配置为输入模式时

- ◇ 输出缓冲器被禁止。
- ◇ 根据输入模式配置参数的不同，连接内部上拉、下拉。
- ◇ 输入数据采样到 GPIO 端口输入数据寄存器
- ◇ 访问 GPIO 端口输入数据寄存器即可得到 GPIOn 状态数据

15. 4. 9 GPIO输出配置

当 GPIO 端口配置为输出模式时：

- ◇ 输出缓冲器被激活
- ◇ 根据配置参数的不同，连接内部上拉、下拉
- ◇ 输入数据采样到 GPIO 端口输入数据寄存器
- ◇ 在开漏模式下，访问输入数据寄存器得到当前 GPIOn 的状态数据。
- ◇ 在推挽模式下，访问输出数据寄存器得到最后一次写的值。

15. 4. 10 模拟输入配置

当 GPIO 端口配置为模拟输入模式时：

- ◇ 输出缓冲器被禁止
- ◇ 施密特输入触发禁止
- ◇ 内部上拉或下拉禁止
- ◇ 访问输入数据寄存器得到的数据为全 0

15. 5 特殊功能寄存器

15. 5. 1 寄存器列表

GPIO 寄存器列表		
名称	偏移地址	描述
基地址： GPIOA_BASE（000_H） GPIOB_BASE（040_H） GPIOC_BASE（080_H） GPIOD_BASE（$0C0_H$） GPIOE_BASE（100_H） GPIOF_BASE（140_H） GPIOG_BASE（180_H） GPIOH_BASE（$1C0_H$）		
GPIO_DIN	000_H	GPIO 端口输入数据寄存器
GPIO_DOUT	004_H	GPIO 端口输出数据寄存器
GPIO_BSRR	008_H	GPIO 端口置位和复位寄存器
GPIO_BIR	$00C_H$	GPIO 端口翻转寄存器
GPIO_MODE	010_H	GPIO 端口模式寄存器
GPIO_ODOS	014_H	GPIO 端口开漏输出选择寄存器
GPIO_PUPD	018_H	GPIO 端口上拉和下拉寄存器
GPIO_PODRV	$01C_H$	GPIO 端口 PMOS 输出驱动寄存器
GPIO_NODRV	020_H	GPIO 端口 NMOS 输出驱动寄存器
GPIO_FLT	024_H	GPIO 端口滤波寄存器
GPIO_TYPE	028_H	GPIO 端口类型寄存器
GPIO_FUNC0	$02C_H$	GPIO 端口复用功能寄存器 0
GPIO_FUNC1	030_H	GPIO 端口复用功能寄存器 1
GPIO_LOCK	034_H	GPIO 端口锁定寄存器
基地址：EXTI_BASE（300_H）		
GPIO_EXTIRER	000_H	GPIO 外部中断上升沿触发使能寄存器
GPIO_EXTIFER	008_H	GPIO 外部中断下降沿触发使能寄存器
GPIO_EXTIEN	010_H	GPIO 外部中断使能寄存器
GPIO_EXTIFLAG	018_H	GPIO 外部中断标志寄存器
GPIO_EXTISFR	020_H	GPIO 外部中断标志置位寄存器
GPIO_EXTICFR	028_H	GPIO 外部中断标志清零寄存器
GPIO_EXTIPSR0	030_H	GPIO 外部中断端口选择寄存器 0
GPIO_EXTIPSR1	034_H	GPIO 外部中断端口选择寄存器 1
GPIO_EXTIFLTCR	040_H	GPIO 外部中断滤波控制寄存器

15. 5. 2 寄存器描述

15. 5. 2. 1 GPIO端口输入数据寄存器（GPIO_DIN）

GPIO 端口输入数据寄存器（GPIO_DIN）																															
偏移地址：00_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DIN15	DIN14	DIN13	DIN12	DIN11	DIN10	DIN9	DIN8	DIN7	DIN6	DIN5	DIN4	DIN3	DIN2	DIN1	DIN0

Reserved	Bit 31-16	—	保留
DIN<y>	Bit 15-0	R	**GPIO 输入数据**

15. 5. 2. 2 GPIO端口输出数据寄存器（GPIO_DOUT）

GPIO 端口输出数据寄存器（GPIO_DOUT）																															
偏移地址：04_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DOUT15	DOUT14	DOUT13	DOUT12	DOUT11	DOUT10	DOUT9	DOUT8	DOUT7	DOUT6	DOUT5	DOUT4	DOUT3	DOUT2	DOUT1	DOUT0

Reserved	Bit 31-16	—	保留
DOUT<y>	Bit 15-0	R/W	**GPIO 输出数据**

15. 5. 2. 3 GPIO端口置位和复位寄存器（GPIO_BSRR）

GPIO 端口置位和复位寄存器（GPIO_BSRR）																															
偏移地址：08_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR15	BRR14	BRR13	BRR12	BRR11	BRR10	BRR9	BRR8	BRR7	BRR6	BRR5	BRR4	BRR3	BRR2	BRR1	BRR0	BSR15	BSR14	BSR13	BSR12	BSR11	BSR10	BSR9	BSR8	BSR7	BSR6	BSR5	BSR4	BSR3	BSR2	BSR1	BSR0

BRR<y>	Bit 31-16	W	**GPIO 复位控制位** 0：无操作 1：相应位复位 注：如果同时对 GPIO 置位和复位，置位的优先级更高
BSR<y>	Bit 15-0	W	**GPIO 置位控制位** 0：无操作 1：相应位置位

15. 5. 2. 4 GPIO端口翻转寄存器（GPIO_BIR）

GPIO 端口翻转寄存器（GPIO_BIR）																															
偏移地址：$0C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																BIR15	BIR14	BIR13	BIR12	BIR11	BIR10	BIR9	BIR8	BIR7	BIR6	BIR5	BIR4	BIR3	BIR2	BIR1	BIR0

Reserved	Bit 31-16	—	**保留**
BIR<y>	Bit 15-0	W	**GPIO 翻转控制位** 0：无操作 1：相应位翻转

15. 5. 2. 5 GPIO端口模式寄存器（GPIO_MODE）

GPIO 端口模式寄存器（GPIO_MODE）																															
偏移地址：10_H																															
复位值：00000000_00000000_00000000_00000000$_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE15		MODE14		MODE13		MODE12		MODE11		MODE10		MODE9		MODE8		MODE7		MODE6		MODE5		MODE4		MODE3		MODE2		MODE1		MODE0	

MODE<y>	Bit 31-0	R/W	**GPIO 模式控制位** 00：预留 01：端口输入 1x：端口输出

15. 5. 2. 6 GPIO端口开漏输出选择寄存器（GPIO_ODOS）

GPIO 端口开漏输出选择寄存器（GPIO_ODOS）																															
偏移地址：14_H																															
复位值：00000000_00000000_00000000_00000000$_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODOS15		ODOS14		ODOS13		ODOS12		ODOS11		ODOS10		ODOS9		ODOS8		ODOS7		ODOS6		ODOS5		ODOS4		ODOS3		ODOS2		ODOS1		ODOS0	

ODOS<y>	Bit 31-0	R/W	**GPIO 开漏输出选择位** 0x：推挽输出 10：PMOS 开漏输出 11：NMOS 开漏输出

15. 5. 2. 7 GPIO端口上拉和下拉寄存器（GPIO_PUPD）

GPIO 端口上拉和下拉寄存器（GPIO_PUPD）																															
偏移地址：18_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD15		PUPD14		PUPD13		PUPD12		PUPD11		PUPD10		PUPD9		PUPD8		PUPD7		PUPD6		PUPD5		PUPD4		PUPD3		PUPD2		PUPD1		PUPD0	

PUPD<y>	Bit 31-0	R/W	**GPIO 上拉和下拉控制位** 00：无上拉和下拉 01：上拉 10：下拉 11：同时上拉和下拉

15. 5. 2. 8 GPIO端口PMOS输出驱动寄存器（GPIO_PODRV）

GPIO 端口 PMOS 输出驱动寄存器（GPIO_PODRV）																															
偏移地址：$1C_H$																															
复位值：$10101010\_10101010\_10101010\_10101010_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PODRV15		PODRV14		PODRV13		PODRV12		PODRV11		PODRV10		PODRV9		PODRV8		PODRV7		PODRV6		PODRV5		PODRV4		PODRV3		PODRV2		PODRV1		PODRV0	

PODRV<y>	Bit 31-0	R/W	**GPIO PMOS 输出驱动控制位** 即端口输出高电平时的驱动能力选择，驱动能力从 0 至 3 依次增强，具体参数请参考数据手册中端口电气特性。 00：驱动 0 01：驱动 1 10：驱动 2 11：驱动 3

15. 5. 2. 9 GPIO端口NMOS输出驱动寄存器（GPIO_NODRV）

GPIO 端口 NMOS 输出驱动寄存器（GPIO_NODRV）																															
偏移地址：20_H																															
复位值：$10101010\_10101010\_10101010\_10101010_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NODRV15		NODRV14		NODRV13		NODRV12		NODRV11		NODRV10		NODRV9		NODRV8		NODRV7		NODRV6		NODRV5		NODRV4		NODRV3		NODRV2		NODRV1		NODRV0	

NODRV<y>	Bit 31-0	R/W	**GPIO NMOS 输出驱动控制位** 即端口输出低电平时的驱动能力选择，驱动能力从 0 至 3 依次增强，具体参数请参考数据手册中端口电气特性。 00：驱动 0 01：驱动 1 10：驱动 2 11：驱动 3

15. 5. 2. 10 GPIO端口滤波寄存器（GPIO_FLT）

GPIO 端口滤波寄存器（GPIO_FLT）																															
偏移地址：24_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																FLT15	FLT14	FLT13	FLT12	FLT11	FLT10	FLT9	FLT8	FLT7	FLT6	FLT5	FLT4	FLT3	FLT2	FLT1	FLT0

Reserved	Bit 31-16	—	**保留**
FLT<y>	Bit 15-0	R/W	**GPIO 滤波控制位** 0：输入滤波禁止 1：输入滤波使能

15. 5. 2. 11 GPIO端口类型寄存器（GPIO_TYPE）

GPIO 端口类型寄存器（GPIO_TYPE）																															
偏移地址：28_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																TYPE15	TYPE14	TYPE13	TYPE12	TYPE11	TYPE10	TYPE9	TYPE8	TYPE7	TYPE6	TYPE5	TYPE4	TYPE3	TYPE2	TYPE1	TYPE0

Reserved	Bit 31-16	—	保留
TYPE<y>	Bit 15-0	R/W	**GPIO 类型选择位** 0：CMOS 1：TTL

15. 5. 2. 12 GPIO端口复用功能寄存器 0（GPIO_FUNC0）

GPIO 端口复用功能寄存器 0（GPIO_FUNC0）																															
偏移地址：$2C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSEL_IO7				FSEL_IO6				FSEL_IO5				FSEL_IO4				FSEL_IO3				FSEL_IO2				FSEL_IO1				FSEL_IO0			

FSEL_IO7	Bit 31-28	R/W	**GPIO<7>功能复用选择位** 请查看相应数据手册中的管脚功能定义
FSEL_IO6	Bit 27-24	R/W	**GPIO<6>功能复用选择位** 请查看相应数据手册中的管脚功能定义
FSEL_IO5	Bit 23-20	R/W	**GPIO<5>功能复用选择位** 请查看相应数据手册中的管脚功能定义
FSEL_IO4	Bit 19-16	R/W	**GPIO<4>功能复用选择位** 请查看相应数据手册中的管脚功能定义
FSEL_IO3	Bit 15-12	R/W	**GPIO<3>功能复用选择位** 请查看相应数据手册中的管脚功能定义
FSEL_IO2	Bit 11-8	R/W	**GPIO<2>功能复用选择位** 请查看相应数据手册中的管脚功能定义
FSEL_IO1	Bit 7-4	R/W	**GPIO<1>功能复用选择位** 请查看相应数据手册中的管脚功能定义
FSEL_IO0	Bit 3-0	R/W	**GPIO<0>功能复用选择位** 请查看相应数据手册中的管脚功能定义

15. 5. 2. 13 GPIO端口复用功能寄存器 1（GPIO_FUNC1）

GPIO 端口复用功能寄存器 1（GPIO_FUNC1）																															
偏移地址：30_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSEL_IO15				FSEL_IO14				FSEL_IO13				FSEL_IO12				FSEL_IO11				FSEL_IO10				FSEL_IO9				FSEL_IO8			

FSEL_IO15	Bit 31-28	R/W	**GPIO<15>功能复用选择位** 请查看相应数据手册中的管脚功能定义
FSEL_IO14	Bit 27-24	R/W	**GPIO<14>功能复用选择位** 请查看相应数据手册中的管脚功能定义
FSEL_IO13	Bit 23-20	R/W	**GPIO<13>功能复用选择位** 请查看相应数据手册中的管脚功能定义
FSEL_IO12	Bit 19-16	R/W	**GPIO<12>功能复用选择位** 请查看相应数据手册中的管脚功能定义
FSEL_IO11	Bit 15-12	R/W	**GPIO<11>功能复用选择位** 请查看相应数据手册中的管脚功能定义
FSEL_IO10	Bit 11-8	R/W	**GPIO<10>功能复用选择位** 请查看相应数据手册中的管脚功能定义
FSEL_IO9	Bit 7-4	R/W	**GPIO<9>功能复用选择位** 请查看相应数据手册中的管脚功能定义
FSEL_IO8	Bit 3-0	R/W	**GPIO<8>功能复用选择位** 请查看相应数据手册中的管脚功能定义

15. 5. 2. 14 GPIO端口锁定寄存器（GPIO_LOCK）

GPIO 端口锁定寄存器（GPIO_LOCK）

偏移地址：34_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY																LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0

KEY	Bit 31-16	W	**GPIO 锁定寄存器关键码** 注：检测到写入关键码 0x55AA，才可对 LOCK<y> 进行写操作，读该位始终为 0
LOCK<y>	Bit 15-0	R/W	**GPIO<y>锁定控制位** 0：相应端口未锁定 1：相应端口锁定 注：被锁定相应端口只允许改变输出数据，LOCK<y>一旦被置位，必须等到下一次 CPU 复位才被清除。

15. 5. 2. 15 GPIO外部中断上升沿触发使能寄存器（GPIO_EXTIRER）

GPIO 外部中断上升沿触发使能寄存器（GPIO_EXTIRER）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTIRER15	EXTIRER14	EXTIRER13	EXTIRER12	EXTIRER11	EXTIRER10	EXTIRER9	EXTIRER8	EXTIRER7	EXTIRER6	EXTIRER5	EXTIRER4	EXTIRER3	EXTIRER2	EXTIRER1	EXTIRER0

Reserved	Bit 31-16	—	**保留**
EXTIRER<y>	Bit 15-0	R/W	**EXTI<y>上升沿触发使能位** 0：禁止 1：使能

15. 5. 2. 16 GPIO外部中断下降沿触发使能寄存器（GPIO_EXTIFER）

GPIO 外部中断下降沿触发使能寄存器（GPIO_EXTIFER）

偏移地址：08_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTIFER15	EXTIFER14	EXTIFER13	EXTIFER12	EXTIFER11	EXTIFER10	EXTIFER9	EXTIFER8	EXTIFER7	EXTIFER6	EXTIFER5	EXTIFER4	EXTIFER3	EXTIFER2	EXTIFER1	EXTIFER0

Reserved	Bit 31-16	—	保留
EXTIFER<y>	Bit 15-0	R/W	**EXTI<y>下降沿触发使能位** 0：禁止 1：使能

15. 5. 2. 17 GPIO外部中断使能寄存器（GPIO_EXTIEN）

GPIO 外部中断使能寄存器（GPIO_EXTIEN）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTIEN15	EXTIEN14	EXTIEN13	EXTIEN12	EXTIEN11	EXTIEN10	EXTIEN9	EXTIEN8	EXTIEN7	EXTIEN6	EXTIEN5	EXTIEN4	EXTIEN3	EXTIEN2	EXTIEN1	EXTIEN0

Reserved	Bit 31-16	—	保留
EXTIEN<y>	Bit 15-0	R/W	**EXTI<y>中断使能位** 0：禁止 1：使能

15. 5. 2. 18 GPIO外部中断标志寄存器（GPIO_EXTIFLAG）

GPIO 外部中断标志寄存器（GPIO_EXTIFLAG）

偏移地址：18_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTIFLAG15	EXTIFLAG14	EXTIFLAG13	EXTIFLAG12	EXTIFLAG11	EXTIFLAG10	EXTIFLAG9	EXTIFLAG8	EXTIFLAG7	EXTIFLAG6	EXTIFLAG5	EXTIFLAG4	EXTIFLAG3	EXTIFLAG2	EXTIFLAG1	EXTIFLAG0

Reserved	Bit 31-16	—	保留
EXTIFLAG<y>	Bit 15-0	R	**EXTI<y>中断状态位** 0：未检测到有效中断 1：检测到有效中断

15. 5. 2. 19 GPIO外部中断标志置位寄存器（GPIO_EXTISFR）

GPIO 外部中断标志置位寄存器（GPIO_EXTISFR）

偏移地址：20_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTISFR15	EXTISFR14	EXTISFR13	EXTISFR12	EXTISFR11	EXTISFR10	EXTISFR9	EXTISFR8	EXTISFR7	EXTISFR6	EXTISFR5	EXTISFR4	EXTISFR3	EXTISFR2	EXTISFR1	EXTISFR0

Reserved	Bit 31-16	—	保留
EXTISFR<y>	Bit 15-0	W1	**EXTI<y>中断标志位置位** 0：无操作 1：置位中断标志位 注：对该位写 1 将中断标志置位，写 0 无效

15. 5. 2. 20 GPIO外部中断标志清零寄存器（GPIO_EXTICFR）

GPIO 外部中断标志清零寄存器（GPIO_EXTICFR）

偏移地址：28_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTICFR15	EXTICFR14	EXTICFR13	EXTICFR12	EXTICFR11	EXTICFR10	EXTICFR9	EXTICFR8	EXTICFR7	EXTICFR6	EXTICFR5	EXTICFR4	EXTICFR3	EXTICFR2	EXTICFR1	EXTICFR0

Reserved	Bit 31-16	—	**保留**
EXTICFR<y>	Bit 15-0	W1	**EXTI<y>中断标志位清零** 0：无操作 1：清零中断标志位 注：对该位写 1 将中断标志复位，写 0 无效

15. 5. 2. 21 GPIO外部中断端口选择寄存器 0（GPIO_EXTIPSR0）

GPIO 外部中断端口选择寄存器 0（GPIO_EXTIPSR0）

偏移地址：30_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	EXTIS7			Reserved	EXTIS6			Reserved	EXTIS5			Reserved	EXTIS4			Reserved	EXTIS3			Reserved	EXTIS2			Reserved	EXTIS1			Reserved	EXTIS0		

Reserved	Bit 31	—	保留
EXTIS7	Bit 30-28	R/W	**EXTI<7>端口选择位** 000：PA7 001：PB7 010：PC7 011：PD7 100：PE7 101：PF7 110：PG7 111：PH7
Reserved	Bit 27	—	保留
EXTIS6	Bit 26-24	R/W	**EXTI<6>端口选择位** 000：PA6 001：PB6 010：PC6 011：PD6 100：PE6 101：PF6 110：PG6 111：PH6
Reserved	Bit 23	—	保留
EXTIS5	Bit 22-20	R/W	**EXTI<5>端口选择位** 000：PA5 001：PB5 010：PC5 011：PD5 100：PE5 101：PF5 110：PG5 111：PH5
Reserved	Bit 19	—	保留
EXTIS4	Bit 18-16	R/W	**EXTI<4>端口选择位** 000：PA4

			001：PB4 010：PC4 011：PD4 100：PE4 101：PF4 110：PG4 111：PH4
Reserved	Bit 15	—	保留
EXTIS3	Bit 14-12	R/W	**EXTI<3>端口选择位** 000：PA3 001：PB3 010：PC3 011：PD3 100：PE3 101：PF3 110：PG3 111：PH3
Reserved	Bit 11	—	保留
EXTIS2	Bit 10-8	R/W	**EXTI<2>端口选择位** 000：PA2 001：PB2 010：PC2 011：PD2 100：PE2 101：PF2 110：PG2 111：PH2
Reserved	Bit 7	—	保留
EXTIS1	Bit 6-4	R/W	**EXTI<1>端口选择位** 000：PA1 001：PB1 010：PC1 011：PD1 100：PE1 101：PF1 110：PG1 111：PH1
Reserved	Bit 3	—	保留
EXTIS0	Bit 2-0	R/W	**EXTI<0>端口选择位** 000：PA0 001：PB0 010：PC0 011：PD0 100：PE0

			101：PF0 110：PG0 111：PH0

15. 5. 2. 22 GPIO外部中断端口选择寄存器 1（GPIO_EXTIPSR1）

GPIO 外部中断端口选择寄存器 1（GPIO_EXTIPSR1）

偏移地址：34_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	EXTIS15			Reserved	EXTIS14			Reserved	EXTIS13			Reserved	EXTIS12			Reserved	EXTIS11			Reserved	EXTIS10			Reserved	EXTIS9			Reserved	EXTIS8		

Reserved	Bit 31	—	保留
EXTIS15	Bit 30-28	R/W	**EXTI<15>端口选择位** 000：PA15 001：PB15 010：PC15 011：PD15 100：PE15 101：PF15 110：PG15 111：PH15
Reserved	Bit 27	—	保留
EXTIS14	Bit 26-24	R/W	**EXTI<14>端口选择位** 000：PA14 001：PB14 010：PC14 011：PD14 100：PE14 101：PF14 110：PG14 111：PH14
Reserved	Bit 23	—	保留
EXTIS13	Bit 22-20	R/W	**EXTI<13>端口选择位** 000：PA13 001：PB13 010：PC13 011：PD13 100：PE13 101：PF13 110：PG13 111：PH13
Reserved	Bit 19	—	保留
EXTIS12	Bit 18-16	R/W	**EXTI<12>端口选择位** 000：PA12

			001：PB12 010：PC12 011：PD12 100：PE12 101：PF12 110：PG12 111：PH12
Reserved	Bit 15	—	保留
EXTIS11	Bit 14-12	R/W	**EXTI<11>端口选择位** 000：PA11 001：PB11 010：PC11 011：PD11 100：PE11 101：PF11 110：PG11 111：PH11
Reserved	Bit 11	—	保留
EXTIS10	Bit 10-8	R/W	**EXTI<10>端口选择位** 000：PA10 001：PB10 010：PC10 011：PD10 100：PE10 101：PF10 110：PG10 111：PH10
Reserved	Bit 7	—	保留
EXTIS9	Bit 6-4	R/W	**EXTI<9>端口选择位** 000：PA9 001：PB9 010：PC9 011：PD9 100：PE9 101：PF9 110：PG9 111：PH9
Reserved	Bit 3	—	保留
EXTIS8	Bit 2-0	R/W	**EXTI<8>端口选择位** 000：PA8 001：PB8 010：PC8 011：PD8 100：PE8

			101：PF8 110：PG8 111：PH8

15. 5. 2. 23 GPIO外部中断滤波控制寄存器（GPIO_EXTIFLTCR）

GPIO 外部中断滤波控制寄存器（GPIO_EXTIFLTCR）

偏移地址：40_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						FLTCKS		FLTSEL								FLTEN15	FLTEN14	FLTEN13	FLTEN12	FLTEN11	FLTEN10	FLTEN9	FLTEN8	FLTEN7	FLTEN6	FLTEN5	FLTEN4	FLTEN3	FLTEN2	FLTEN1	FLTEN0

Reserved	Bit 31-26	—	**保留**
FLTCKS	Bit 25-24	R/W	**EXTI 端口中断去抖滤波时钟选择** 00：ULRC（约 10KHz） 01：LRC（约 32KHz） 10：预留 11：预留
FLTSEL	Bit 23-16	R/W	**EXTI 端口中断去抖滤波选择** 滤波时间 = (FLTSEL<7:0> + 1)×2 个时钟周期
FLTEN<y>	Bit 15-0	R/W	**EXTI<y>端口中断去抖滤波使能** 0：禁止 1：使能 注：该位需在中断使能之前配置，使能之后禁止更改

第16章 循环冗余校验（CRC）

16. 1 概述

循环冗余校验（CRC）发生器可以执行带可编程多项式设定的 CRC 计算，用于对数据传输的完整性和正确性进行校验 。

16. 2 特性

- 支持四个常用的多项式：CRC-CCITT，CRC-8，CRC-16 和 CRC-32
  - CRC-CCITT：$X^{16} + X^{12} + X^5 + 1$
  - CRC-8：$X^8 + X^2 + X + 1$
  - CRC-16：$X^{16} + X^{15} + X^2 + 1$
  - CRC-32：$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- 支持可编程的种子值
- 支持对输入数据和 CRC 校验值的可编程的反序设定
- 支持对输入数据和 CRC 校验值的可编程的反码设定
- 支持 8/16/32 位数据宽度
  - 8-bit 写模式：1 个 AHB 时钟周期操作
  - 16-bit 写模式：2 个 AHB 时钟周期操作
  - 32-bit 写模式：4 个 AHB 时钟周期操作
- 支持使用 DMA 写数据执行 CRC 操作

16. 3 结构框图

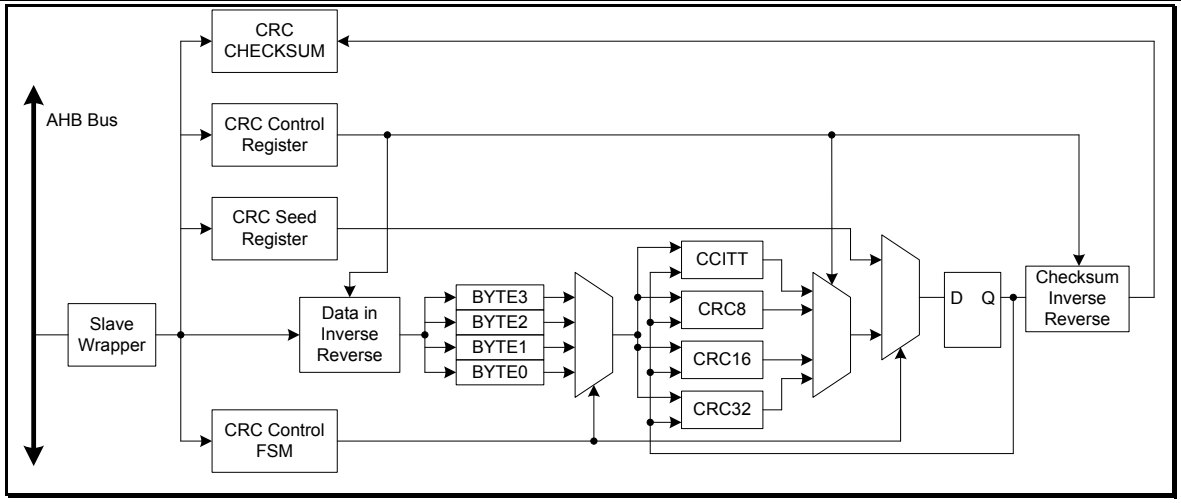


图 16-1 CRC 结构框图

16. 4 功能描述

16. 4. 1 常规操作

CRC 发生器可以执行带可编程多项式设定的 CRC 运算。多项式操作包括 CRC-CCITT，CRC-8，CRC-16 和 CRC-32。用户可以通过设置 MODE 选择 CRC 多项式操作模式。

操作示例：

1. 通过设置 CRC_CR.EN 使能 CRC 发生器
2. CRC 运算初始化设置
   a. 通过设置 CRC_CR.CHSINV 配置 CRC 校验值反码
   b. 通过设置 CRC_CR.CHSREV 配置 CRC 校验值位反序
   c. 通过设置 CRC_CR.DATINV 配置 CRC 写入数据反码
   d. 通过设置 CRC_CR.DATREV 配置 CRC 写入数据位反序
   e. 通过设置 CRC_CR.MODE 配置 CRC 校验模式
   f. 通过设置 CRC_CR.DATLEN 配置 CRC 写入数据长度
3. 通过设置 CRC_CR.RST 执行 CRC 复位，CRC 复位将装载初始种子值到 CRC 运算电路
4. 写数据到 CRC_DATA 寄存器来计算 CRC 校验值
5. 通过读 CRC_CHECKSUM 寄存器来获得 CRC 校验结果

16. 4. 2 DMA请求

DMA 请求（在使能后）仅用于数据传输，当需要减少 MCU 负荷时可以使用 DMA 功能，在计算完 CRC 后，DMA 模块将 CRC_CHECKSUM 里面的计算值传输到用户提供的存储区中。

操作示例：

1. 通过设置 CRC_CR.EN 使能 CRC 发生器
2. CRC 运算初始化设置
   a. 通过设置 CRC_CR.CHSINV 配置 CRC 校验值反码
   b. 通过设置 CRC_CR.CHSREV 配置 CRC 校验值位反序
   c. 通过设置 CRC_CR.DATINV 配置 CRC 写入数据反码
   d. 通过设置 CRC_CR.DATREV 配置 CRC 写入数据位反序
   e. 通过设置 CRC_CR.MODE 配置 CRC 校验模式
   f. 通过设置 CRC_CR.DATLEN 配置 CRC 写入数据长度
3. 通过设置 CRC_CR.RST 执行 CRC 复位，CRC 复位将装载初始种子值到 CRC 运算

电路

4. 将 CRC_DATA 寄存器的地址填入 DMA 模块的目标地址，表示每发生一次 CRC 的 DMA 申请事件，数据搬运到目标地址，此配置参考 DMA 章节
5. 将用户定义的数据存储区的地址填入 DMA 的原地址，表示每发生一次 CRC 的 DMA 申请事件后数据搬运到源地址，此配置参考 DMA 章节
6. 用户需要传输的总字节数需要在 DMA 中配置好，每发生一次 CRC 的 DMA 申请事件后，该值都会递减，注意最大字节数为 1024 个字节，此配置参考 DMA 章节
7. 设置 DMA 的触发源，选择触发 DMA 的输入源为 CRC
8. DMA 通道的优先级需要正确配置
9. 以上步骤都执行完毕后，初始化阶段完成了，需要激活该 DMA 通道
10. 使能 CRC_CR.DMAEN 位。

16. 5 特殊功能寄存器

16. 5. 1 寄存器列表

CRC 寄存器列表		
名称	偏移地址	描述
CRC_CR	0000_H	CRC 控制寄存器
CRC_DATA	0004_H	CRC 写数据寄存器
CRC_SEED	0008_H	CRC 种子寄存器
CRC_CHECKSUM	$000C_H$	CRC 校验值寄存器

16. 5. 2 寄存器描述

16. 5. 2. 1 CRC控制寄存器（CRC_CR）

CRC 控制寄存器（CRC_CR）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							BYTORD	DATLEN		MODE		CHSINV	DATINV	CHSREV	DATREV	Reserved											DMAEN	CWERR	WERR	RST	EN

Reserved	Bit 31-25	—	**保留**
BYTORD	Bit 24	R/W	**校验值字节顺序选择位** 0：优先校验低字节 1：优先校验高字节 注：优先校验低字节即优先校验Bit 7-0；优先校验高字节，16-bit模式即优先校验Bit 15-8，32-bit模式优先校验Bit 31-24；8-bit模式校验顺序无区别
DATLEN	Bit 23-22	R/W	**数据长度选择位** 00：通过写 CRC_DATA 寄存器方式自动判断 01：数据为 8-bit（CRC_DATA[7:0]有效） 10：数据为 16-bit（CRC_DATA[15:0]有效） 11：数据为 32-bit（CRC_DATA[31:0]有效）
MODE	Bit 21-20	R/W	**模式选择位** 00：CRC-CCITT 多项式模式 01：CRC-8 多项式模式 10：CRC-16 多项式模式 11：CRC-32 多项式模式
CHSINV	Bit 19	R/W	**校验值反码使能位** 0：禁止 1：使能
DATINV	Bit 18	R/W	**写数据反码使能位** 0：禁止 1：使能
CHSREV	Bit 17	R/W	**校验值反序使能位** 0：禁止 1：使能
DATREV	Bit 16	R/W	**写数据反序使能位** 0：禁止 1：使能
Reserved	Bit 15-5	—	**保留**
DMAEN	Bit 4	R/W	**DMA 使能位** 0：禁止

			1：使能
CWERR	Bit 3	W1	**CRC 写数据错误标志清除位** 0：无操作 1：清除标志位
WERR	Bit 2	R	**CRC 写数据错误标志位** 0：无错误 1：发生写数据错误 注：写入格式与 DATLEN 所选择不相符时会将标志位置位，未在最低字节或低半字写数据时也会将标志位置位。
RST	Bit 1	W1	**CRC 复位位** 0：无操作 1：复位 注：该位复位CRC内部状态机、DMAEN和缓存以及初始化种子值，但不会复位寄存器值
EN	Bit 0	R/W	**CRC 使能位** 0：禁止 1：使能

16. 5. 2. 2 CRC写数据寄存器（CRC_DATA）

CRC 写数据寄存器（CRC_DATA）

偏移地址：04_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

DATA	Bit 31-0	R/W	**CRC 写入数据位** 用户看可以通过 CPU 或 DMA 直接写数据到该位来执行 CRC 操作。 注 1：当写数据长度是 8-bit 模式，该位有效数据为 DATA[7:0]，如果数据长度是 16-bit 模式，该位有效数据为 DATA[15:0]。如果自动检测数据长度，则可通过最低字节、低半字或字写入方式任意写入数值。 注 2：当写入到错误的字节或半字时，硬件会置位写数据错误标志位 WERR。

16. 5. 2. 3 CRC种子寄存器（CRC_SEED）

CRC 种子寄存器（CRC_SEED）																															
偏移地址：08_H																															
复位值：$11111111\_11111111\_11111111\_11111111_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEED																															

SEED	Bit 31-0	R/W	**CRC 种子值** 该位表示 CRC 校验种子值

16. 5. 2. 4 CRC校验值寄存器（CRC_CHECKSUM）

CRC 校验值寄存器（CRC_CHECKSUM）																															
偏移地址：$0C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHECKSUM																															

CHECKSUM	Bit 31-0	R	**CRC 校验值** 该位表示 CRC 校验结果

第17章 硬件加密（CRYPT）

17.1 概述

硬件加密模块主要用于由硬件对数据进行加密或解密操作，支持的标准有 AES、DES。

AES（Advanced Encryption Standard）是最新的分组对称密码算法，兼容联邦信息处理标准出版物（FIPS PUB 197，2001 年 11 月 26 日）规定的高级加密标准（AES）。

DES（Data Encryption Standard）是应用非常广泛的对称密码算法，兼容联邦信息处理标准出版物（FIPS PUB 46-3，1999 年 10 月 25 日）规定的数据加密标准（DES）和三重 DES（TDES），遵循美国国家标准协会（ANSI）X9.52 标准。

17.2 特性

- 适用于 AES、DES 和 TDES 加密和解密操作
- AES
  - 支持 128、192 和 256 位的密钥
  - 支持 ECB、CBC、CTR 和 GCM 模式
  - 支持 1、8、16 或 32 位数据交换
- DES/TDES
  - 支持 ECB、CBC 模式
  - 支持 64、128 和 192 位密钥
  - 支持在 CBC 模式下使用的 2×32 位初始化向量（IV）
  - 支持 1、8、16 或 32 位数据交换
- 支持直接存储器访问（DMA）（用于传入数据和读出已处理数据）
- 支持产生 CPU 中断请求

17. 3 AES功能描述

17. 3. 1 AES-ECB模式加密

AES 是以 128bit 作为一个数据单元进行加密，下图介绍了 AES 电子密码本（AES-ECB）模式加密。一个数据单元（128bit 明文 P）经过位/字节/半字交换后作为一个输入单元（I）。输入单元通过 AES 算法（AEA）在加密状态下使用 128 位、192 位或 256 位密钥进行处理。得到的处理结果再执行位/字节/半字交换后，得到 128bit 输出密文单元（C）。

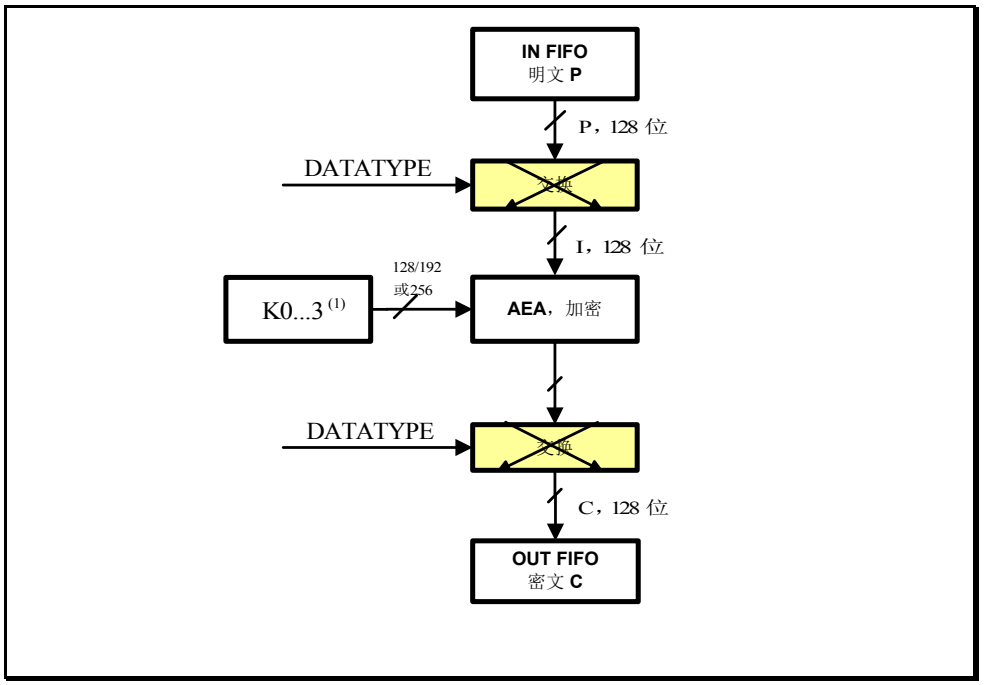


图 17-1 AES-ECB 模式加密

注 1：K = 密钥；C = 密文；I = 输入块；O = 输出块；P = 明文。
注 2：如果密钥大小 = 128：密钥 = [K3 K2];
如果密钥大小 = 192：密钥 = [K3 K2 K1];
如果密钥大小 = 256：密钥 = [K3 K2 K1 K0]。

操作示例：

1. 设置 CRYPT_CON.CRYSEL = 0，选择算法类型为 AES
2. 设置 CRYPT_CON.ENCS = 1，选择加密
3. 设置 CRYPT_CON.AESKS，选择密钥长度 128bit、192bit 或者 256bit
4. 设置 CRYPT_CON.MODE = 0，选择 ECB 模式
5. 向 CRYPT_KEY0...CRYPT_KEY7 中填入密钥， 128bit 长度填入 KEY3/2/1/0,192bit 长度填入 KEY5/4/3/2/1/0，256bit 长度填入 KEY7/6/5/4/3/2/1/0
6. 向 CRYPT_DATA0/1/2/3 中写入需要加密的明文
7. 设置 CRYPT_CON.GO = 1，启动加密

8. 通过 CRYPT_IF.DONE 判断加密是否完成，如果 CRYPT_IF.DONE = 0，则加密完成
9. 从 CRYPT_DATA0/1/2/3 中读出已经加密的密文

17. 3. 2 AES-ECB模式解密

AES 是以 128bit 作为一个数据单元进行解密，下图介绍了 AES 电子密码本（AES-ECB）模式解密。一个数据单元（128bit 密文 C）经过位/字节/半字交换后作为一个输入单元（I）。输入单元通过 AES 算法（AEA）在解密状态下使用 128 位、192 位或 256 位密钥进行处理。得到的处理结果再执行位/字节/半字交换后，得到 128bit 输出明文单元（P）。

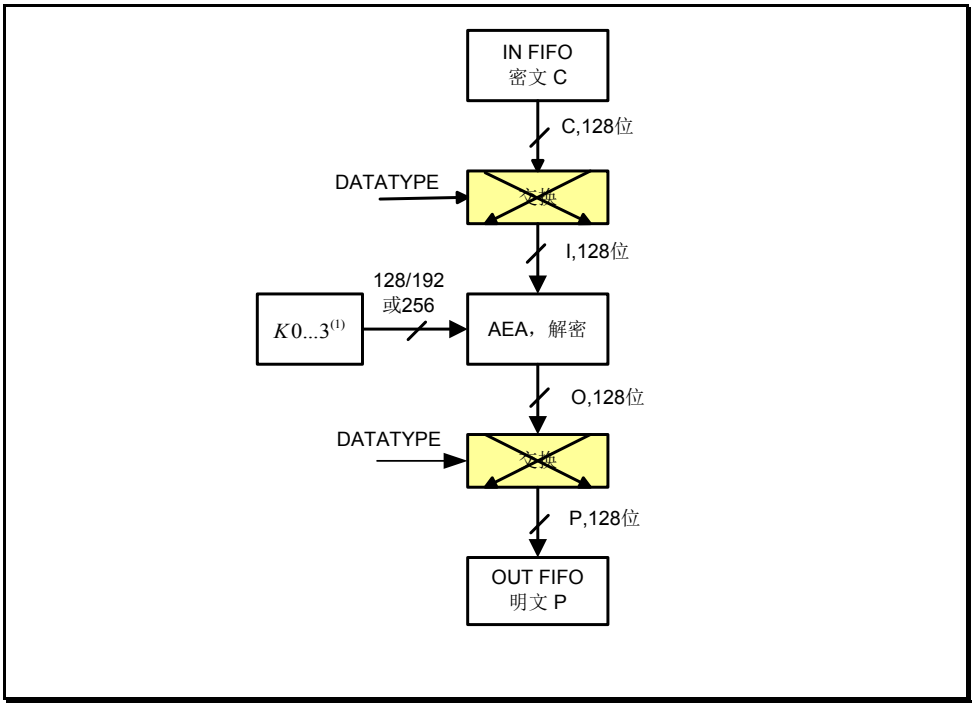


图 17-2 AES-ECB 模式解密

注 1：K = 密钥；C = 密文；I = 输入块；O = 输出块；P = 明文。
注 2：如果密钥大小 = 128：密钥 = [K3 K2];
如果密钥大小 = 192：密钥 = [K3 K2 K1];
如果密钥大小 = 256：密钥 = [K3 K2 K1 K0]。

操作示例：

1. 设置 CRYPT_CON.CRYSEL = 0，选择算法类型为 AES
2. 设置 CRYPT_CON.ENCS = 0，选择解密
3. 设置 CRYPT_CON.AESKS，选择密钥长度 128bit、192bit 或者 256bit
4. 设置 CRYPT_CON.MODE = 0，选择 ECB 模式
5. 向 CRYPT_KEY0...CRYPT_KEY7 中填入密钥， 128bit 长度填入 KEY3/2/1/0,192bit 长度填入 KEY5/4/3/2/1/0，256bit 长度填入 KEY7/6/5/4/3/2/1/0
6. 向 CRYPT_DATA0/1/2/3 中写入需要解密的密文

7. 设置 CRYPT_CON.GO = 1，启动解密
8. 通过 CRYPT_IF.DONE 判断解密是否完成，如果 CRYPT_IF.DONE = 0，则解密完成
9. 从 CRYPT_DATA0/1/2/3 中读出已经解密的明文

17. 3. 3 AES-CBC模式加密

AES 是以 128 位作为一个数据单元进行加密，下图介绍了 AES 加密分组链接（AES-CBC）模式加密。该模式首先将明文消息分成多个 128 位数据明文单元。在 CBC 加密过程中，一个数据单元经过执行位/字节/半字交换后作为明文单元（P1），通过与一个 128 位初始化向量（IV）进行异或运算（IV^P1），作为第一个输入单元（I1）。该输入单元通过 AES 算法（AEA）在加密状态下使用 128 位、192 位或 256 位密钥进行加密处理。生成的 128 位输出单元（O1）将直接用作密文（C1），即 C1 = O1。然后，第一个密文单元与第二个明文数据单元进行异或运算（C1^P2），从而生成第二个输入单元（I2）。第二个输入单元通过以上 AES 处理而生成第二个密文单元。加密处理会不断将后续密文单元和明文单元链接到一起，直到消息中所有的明文单元都加密完成为止。如果消息中最后的数据单元不是一个 128 位数据单元，则由应用程序按照一定规则对不完整数据单元进行加密。

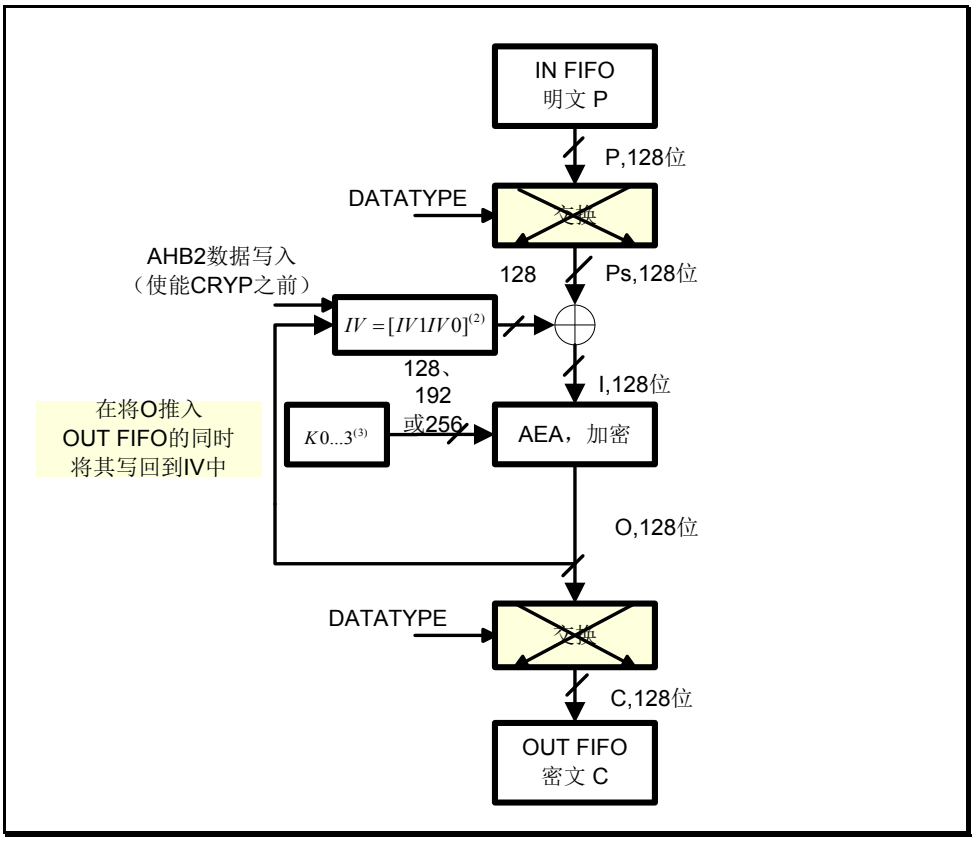


图 17-3 AES-CBC 模式加密

注 1：K = 密钥；C = 密文；I = 输入块；Ps = 交换前（解码时）或 交换后（编码时）的明文；P = 明文；IV = 初始化向量。
注 2：IVx = [IVxR IVxL], R = 右，L = 左。
注 3：如果密钥大小 = 128: 密钥 = [K3 K2];
如果密钥大小 = 192: 密钥 = [K3 K2 K1];

如果密钥大小 = 256: 密钥 = [K3 K2 K1 K0]。

操作示例：

1. 设置 CRYPT_CON.CRYSEL = 0，选择算法类型为 AES
2. 设置 CRYPT_CON.ENCS = 1，选择加密
3. 设置 CRYPT_CON.AESKS，选择密钥长度 128bit、192bit 或者 256bit
4. 设置 CRYPT_CON.MODE = 1，选择 CBC 模式
5. 向 CRYPT_KEY0...CRYPT_KEY7 中填入密钥， 128bit 长度填入 KEY3/2/1/0,192bit 长度填入 KEY5/4/3/2/1/0，256bit 长度填入 KEY7/6/5/4/3/2/1/0
6. 向 CRYPT_IV0/1/2/3 填入初始向量
7. 向 CRYPT_DATA0/1/2/3 中写入需要加密的明文
8. 设置 CRYPT_CON.GO = 1，启动加密
9. 通过 CRYPT_IF.DONE 判断加密是否完成，如果 CRYPT_IF.DONE = 0，则加密完成
10. 从 CRYPT_DATA0/1/2/3 中读出已经加密的密文
11. 回到步骤 7 继续进行加密
12. 所有加密完成

17. 3. 4 AES-CBC模式解密

AES 是以 128 位作为一个数据单元进行解密，下图介绍了 AES 密码分组链接（AES-CBC）模式解密。该模式首先将密文消息分成多个 128 位数据密文单元，第一个密文单元（C1）经过位/字节/半字交换后作为一个输入单元（I1），该输入单元（I1）通过 AES 算法（AEA）在解密状态下使用 128 位、192 位或 256 位密钥进行解密处理，生成的 128 位输出单元（O1），再与 IV 进行异或运算（O1^IV），从而生成第一个明文单元。然后，第二个密文单元作为下一个输入单元，经过以上 AES 解密处理，生成的输出单元再与第一个密文单元进行异或运算（O2^C1），从而生成第二个明文数据单元（P2）。依次进行解密，直到消息中所有的密文单元都解密完成为止。

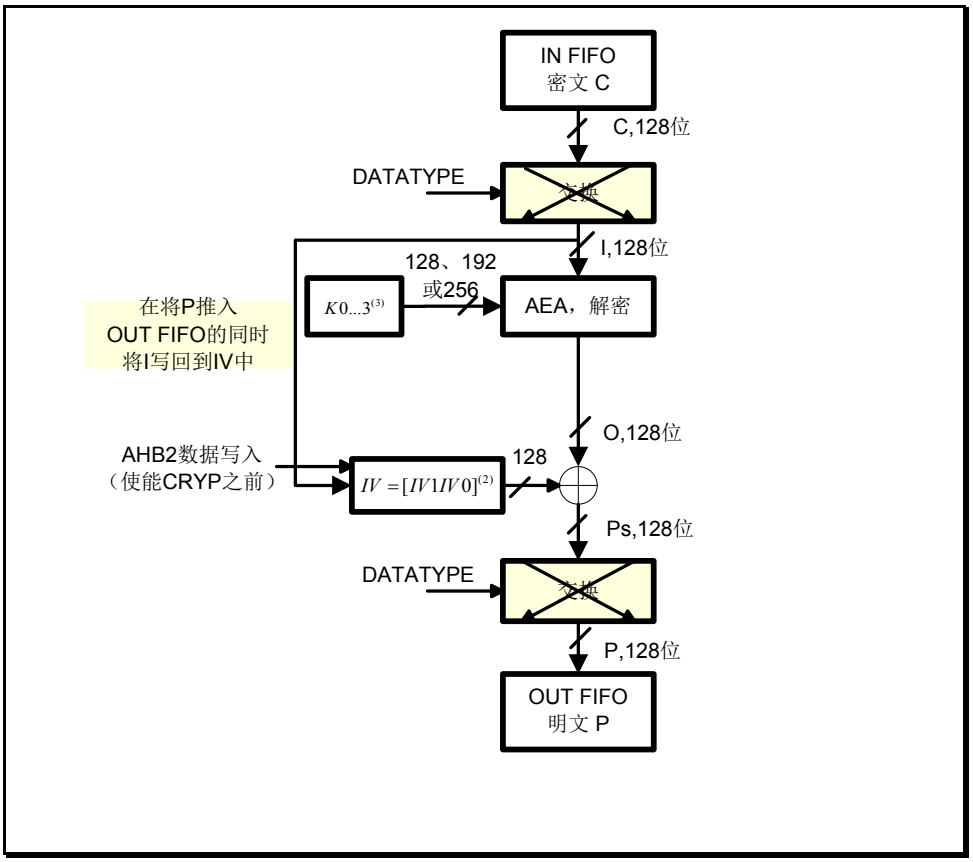


图 17-4 AES-CBC 模式解密

注 1：K = 密钥；C = 密文；I = 输入块；Ps = 交换前（解码时）或 交换后（编码时）的明文；P = 明文；IV = 初始化向量。

注 2：IVx = [IVxR IVxL], R = 右，L = 左。

注 3：如果密钥大小 = 128: 密钥 = [K3 K2];
如果密钥大小 = 192: 密钥 = [K3 K2 K1];
如果密钥大小 = 256: 密钥 = [K3 K2 K1 K0]。

操作示例：

1. 设置 CRYPT_CON.CRYSEL = 0，选择算法类型为 AES
2. 设置 CRYPT_CON.ENCS = 0，选择解密
3. 设置 CRYPT_CON.AESKS，选择密钥长度 128bit、192bit 或者 256bit
4. 设置 CRYPTCON.MODE = 1，选择 CBC 模式
5. 向 CRYPT_KEY0...CRYPT_KEY7 中填入密钥， 128bit 长度填入 KEY3/2/1/0,192bit 长度填入 KEY5/4/3/2/1/0，256bit 长度填入 KEY7/6/5/4/3/2/1/0
6. 向 CRYPT_IV0/1/2/3 填入初始向量
7. 向 CRYPT_DATA0/1/2/3 中写入需要解密的密文
8. 设置 CRYPT_CON.GO = 1，启动解密

9. 通过 CRYPT_IF.DONE 判断解密是否完成，如果 CRYPT_IF.DONE = 0，则解密完成
10. 从 CRYPT_DATA0/1/2/3 中读出已经解密的明文
11. 回到步骤 7 继续进行解密
12. 所有解密完成

17. 3. 5 AES-CTR模式加解密

AES 计数模式（AES-CTR）即把一个 32bit 的计数器和一个随机数作为 AES 加解密模块的输入，输出的密文流作为密钥流，对明文或密文进行异或从而得到相应的密文或明文。在此模式下，解密操作与加密操作的操作方式完全相同。图 17-5 和图 17-6 分别给出了加密流程和解密流程。

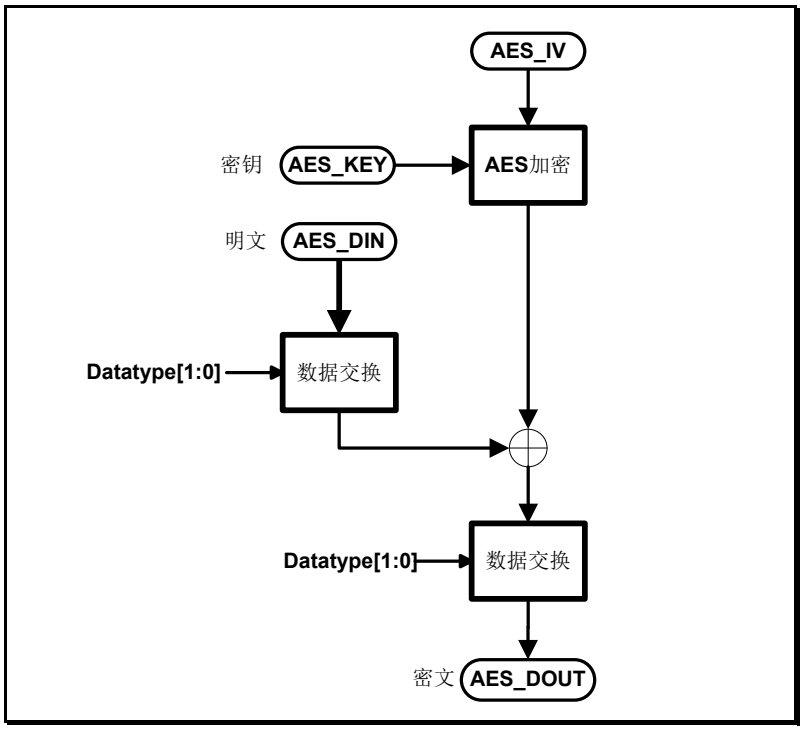


图 17-5 CTR 加密流程

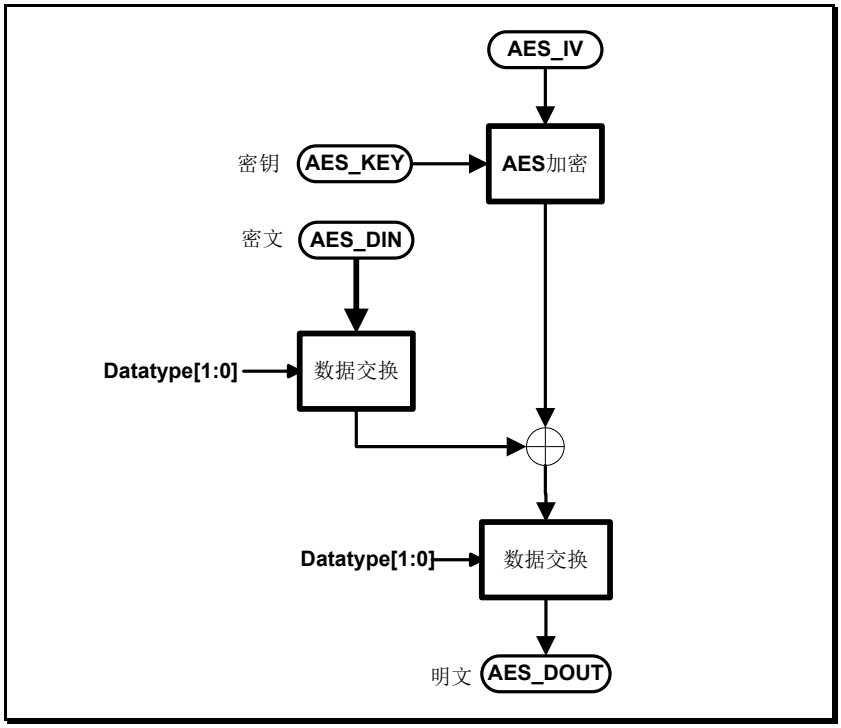


图 17-6 CTR 解密流程

随机数(nonce)和 32 位计数器存储在 IV 寄存器中，如图 17-7 所示：

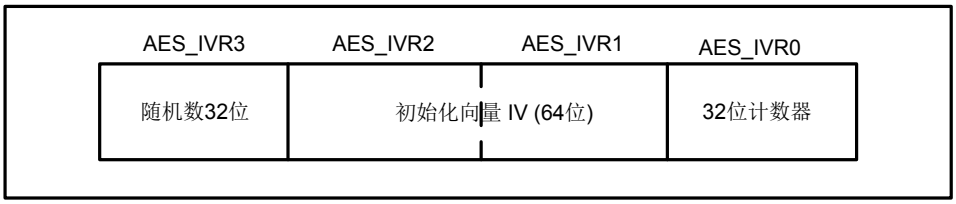


图 17-7 计数器模式下的初始计数器块结构

◇ 随机数是一个 32 位的一次性值。每个不同的通信都应得到最新的随机值。

◇ 初始化向量（IV）是一个 64 位的值，为确保一个给定值只用于一个给定的密钥，必须选择 IV。

◇ 32 位的计数器是一个大端模式的整数，随着块的加密次数增加而增加。计数器的初始值应设置为“1”。

◇ 为保持计数器块的最高有效 96 位不变，其使用最低有效 32 位进行增量的计数。

操作示例：

1. 设置 CRYPT_CON.CRYSEL = 0，选择算法类型为 AES
2. 设置 CRYPT_CON.ENCS = 1/0，选择加密/解密
3. 设置 CRYPT_CON.AESKS，选择密钥长度 128bit、192bit 或者 256bit
4. 设置 CRYPT_CON.MODE = 2，选择 CTR 模式
5. 向 CRYPT_KEY0...CRYPT_KEY7 中填入密钥， 128bit 长度填入 KEY3/2/1/0,192bit 长度填入 KEY5/4/3/2/1/0，256bit 长度填入 KEY7/6/5/4/3/2/1/0

6. 向 CRYPT_IV0/1/2/3 填入初始向量为初始计数器块
7. 向 CRYPT_DATA0/1/2/3 中写入需要加密的明文/解密的密文
8. 设置 CRYPT_CON.GO = 1，启动加密/解密
9. 通过 CRYPT_IF.DONE 判断加密/解密是否完成，如果 CRYPT_IF.DONE = 0，则加密/解密完成
10. 从 CRYPT_DATA0/1/2/3 中读出已经加密/的密文/解密的明文
11. 回到步骤 7 继续进行加密/解密
12. 所有加密/解密完成

17. 3. 6 AES-GCM模式加解密

AES Galois/计数器模式 (GCM) 不仅可以加密和验证明文，还能够生成对应的密文(消息验证码)和标记(消息完整性检查)。该算法基于 AES 计数器模式，可确保保密性。它针对固定有限的字段使用乘法器来生成标记。开始执行算法时需要使用初始化向量。

要处理的消息分为 2 个部分：

◇ 信息头（附加认证数据）：需要验证但不被保护的数据（类似路由数据包的信息）

◇ 信息（有效负载）：经过验证和加密的消息本身

注：附加认证数据必须在信息的前面，并且两部分不能混合。

依据 GCM 标准，在消息结束时，必须传递完由信息头大小（64 位）和信息大小（64 位）构成的特定 128 位块。在计算过程中，必须将信息头块与信息块区分开。

GCM 的加密按照以下公式定义：

$$H = E(K, 0^{128})$$

$$Y_0 = \begin{cases} IV \,||\, 0^{31}1 & if\ len(IV) = 96 \\ GHASH(H, \{\}, IV) & otherwise \end{cases}$$

$$Y_i = incr(Y_{i-1})\ for\ i = 1, \ldots, n$$

$$C_i = P_i \oplus E(K, Y_i)\ for\ i = 1, \ldots, n-1$$

$$C_n^* = P_n^* \oplus MSB_u(E(K, Y_n))$$

$$T = MSB_t(GHASH(H, A, C) \oplus E(K, Y_0))$$

其中 GHASH 函数的定义为 $GHASH(H, A, C) = X_{m+n+1}$，其中 X 的定义为

$$X_i = \begin{cases} 0 & for\ i = 0 \\ (X_{i-1} \oplus A_i) \bullet H & for\ i = 1,\ldots,m-1 \\ (X_{m-1} \oplus (A_m^* \| 0^{128-v})) \bullet H & for\ i = m \\ (X_{i-1} \oplus C_i) \bullet H & for\ i = m+1,\ldots,m+n-1 \\ (X_{m+n-1} \oplus (C_m^* \| 0^{128-u})) \bullet H & for\ i = m+n \\ (X_{m+n} \oplus (len(A) \| len(C))) \bullet H & for\ i = m+n+1. \end{cases}$$

GCM 模式的加解密流程如图 17-8、图 17-9 所示。

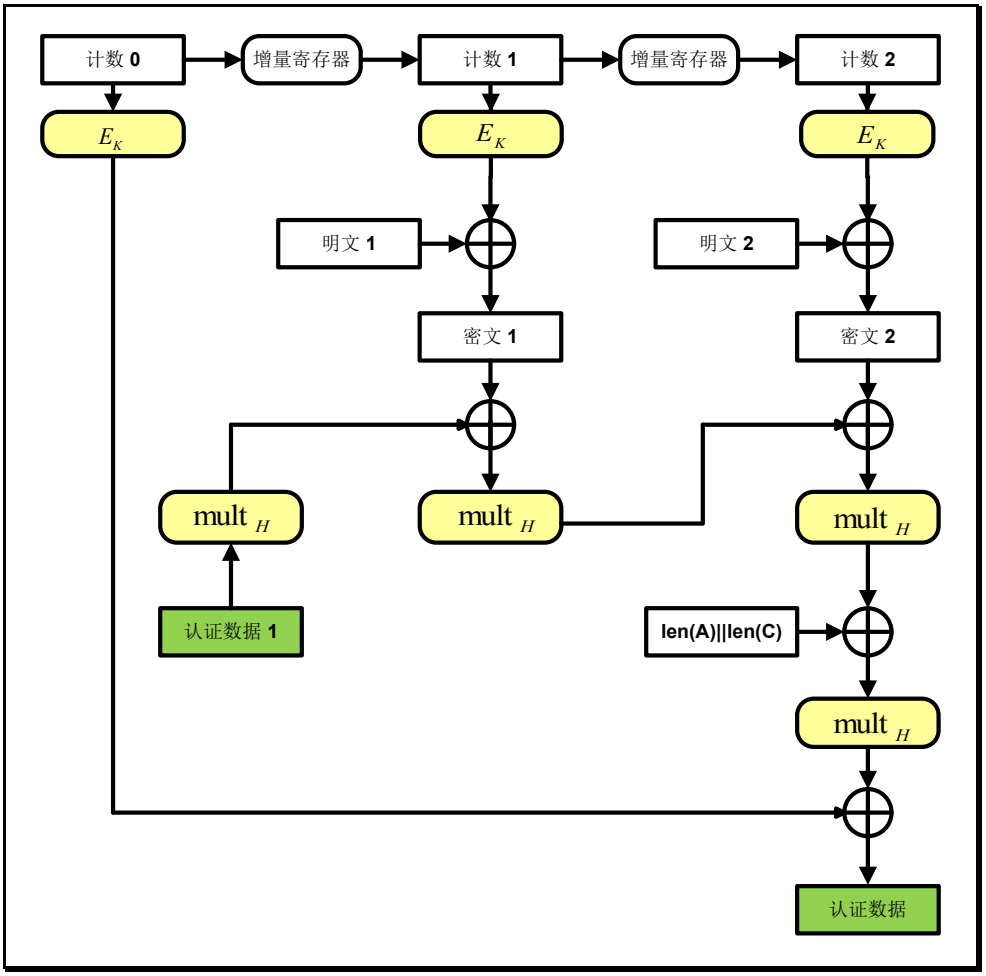


图 17-8 GCM 加密流程

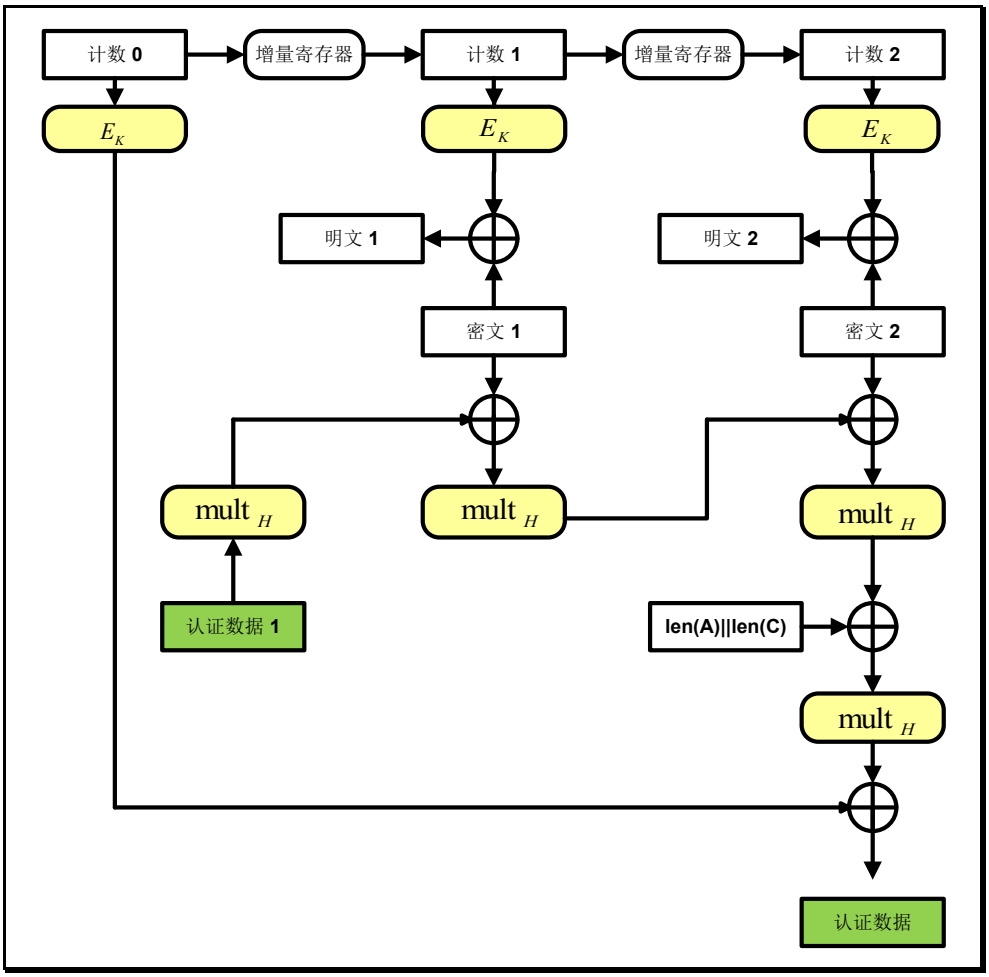


图 17-9 GCM 解密流程

图中 EK 表示 AES 加密模块。MULTH 模块是一个 GF(2^128)域上的乘法。增量寄存器表示计数器加 1。GCM 模式由软件配合实现，硬件提供一个 AES 模块和 MULTH 模块供软件调度。GCM 模式加解密的过程与 CTR 模式相同。认证过程需执行 GHASH 算法，可通过软件调度硬件模块（MUL 模式）实现。

MUL 模式使用说明

算法描述

Algorithm 1: X •Y

Input:

blocks X, Y

Output:

block X •Y

Steps:

1). Let x0x1...x127 denote the sequence of bits in X.。

2). Let Z0 = 0128 and V0 = Y..。

3). For i = 0 to 127, calculate blocks Zi+1 and Vi+1 as follows:

$$Z_{i+1} = \begin{cases} Z_i & if\ x_i = 0; \\ Z_i \oplus V_i & if\ x_i = 1; \end{cases}$$

$$V_{i+1} = \begin{cases} V_i >> 1 & if\ LSB_1(V_i) = 0; \\ V_i >> 1 \oplus R_i & if\ LSB_1(V_i) = 1; \end{cases}$$

4). Return Z128。

实现方式

在 MUL 模式下，执行加密/解密需要以下 3 个步骤：

第 1 步：CRYPT_CON. MODE=3 选择 MUL 模式选择），CRYPT_CON.RESCLR=1，清空 CRYPT_ RESx 内容, CRYPT_CON.GO = 1 使能 MUL 运算；

第 2 步： 将 X 放入 CRYPT_DATAx 寄存器，Y 放入 CRYPT_IVx 寄存器，Z 存放在 CRYPT_RESx 寄存器；

第 3 步：读取输出结果：在 CRYPT_IF.MULTHIF 为高时，读取 CRYPT_RESx 的值，读取后，置 CRYPT_IFC.MULTHIFC = 1。

在 GCM 模式下，执行加密/解密需要以下 3 个步骤

第 1 步：GCM 初始化

操作示例：

1. 将 CRYPT_CON.GO 清零来禁止加密处理器。
2. 初始化密钥寄存器 CRYPT_KEYx（128/192/256 位）及初始向量寄存器 CRYPT_IVx。

第 2 步：GCM 加密/解密

操作示例：

1. 将明文/密文写入数据寄存器 CRYPT_DATAx，并使用 CRYPT_CON 中的 FIFOEN 位判断 FIFO 是否使能，且通过 FIFOODR 位确定 FIFO 的顺序。
2. 配置 CRYPT_CON.ENCS 位选择加密或解密。设置 CRYPT_CON.MODE = 2， 设置 CRYPT_CON.GO = 1 启动运算。
3. 等待运算结束后，从 CRYPT_RESx 中读取结果。
4. 重复之前的步骤，直到所有的明文/密文加解密完成，此过程可以使用 DMA。

第 3 步：GCM 签名验证

该阶段需要用到 HASH 函数。HASH 函数的输入数据包括需要验证但不受保护的数据 A（128 位的整数倍，不足以 0 补充）、密文 C（128 位的整数倍，不足以 0 补充）、数据 A 的长度（64 位）和密文 C 的长度（64 位）。

操作示例：

1. 配置 CRYPT_CON.MODE = 0，计算 H 值。
2. 配置 CRYPT_CON.RESCLR = 1，将寄存器 CRYPT_RESx 清 0。
3. 将 HASH 函数的输入数据以 4 字一组与 CRYPT_RESx 异或后写入寄存器 CRYPT_DATAx，将 H 值写入寄存器 CRYPT_IVx。
4. 配置 CRYPT_CON.MODE = 3，设置 CRYPT_CON.GO = 1 启动运算。
5. 重复 2)-5)步，直到所有数据运算完成。读取 CRYPT_RESx 为 HASH 值。
6. 对初始向量进行一次 ECB 运算，将得到的结果与 HASH 值异或，以此得到签名。

17. 4 DES功能描述

17. 4. 1 DES/TDES-ECB模式加密

DES 或 TDES 是以 64bit 作为一个数据单元进行加密，下图介绍 DES 和 TDES 电子密码本（DES/TDES-ECB）模式中的加密。一个数据单元（64bit 明文 P）经过位/字节/半字交换后作为一个输入单元（I）。该输入单元通过 DES 算法（DEA）在加密状态下使用 K1 进行加密处理。在 DES 算法处理后的输出会作为下一次 DES 算法（DEA）的输入，在解密状态下使用 K2 执行 DES 算法。在 DES 算法处理后的输出再次作为 DES 算法（DEA）的输入，在加密状态下使用 K3 再次执行 DES 算法。最后生成的 64bit 输出块（O），再执行位/字节/半字交换之后，以密文（C）形式输出到数据寄存器。

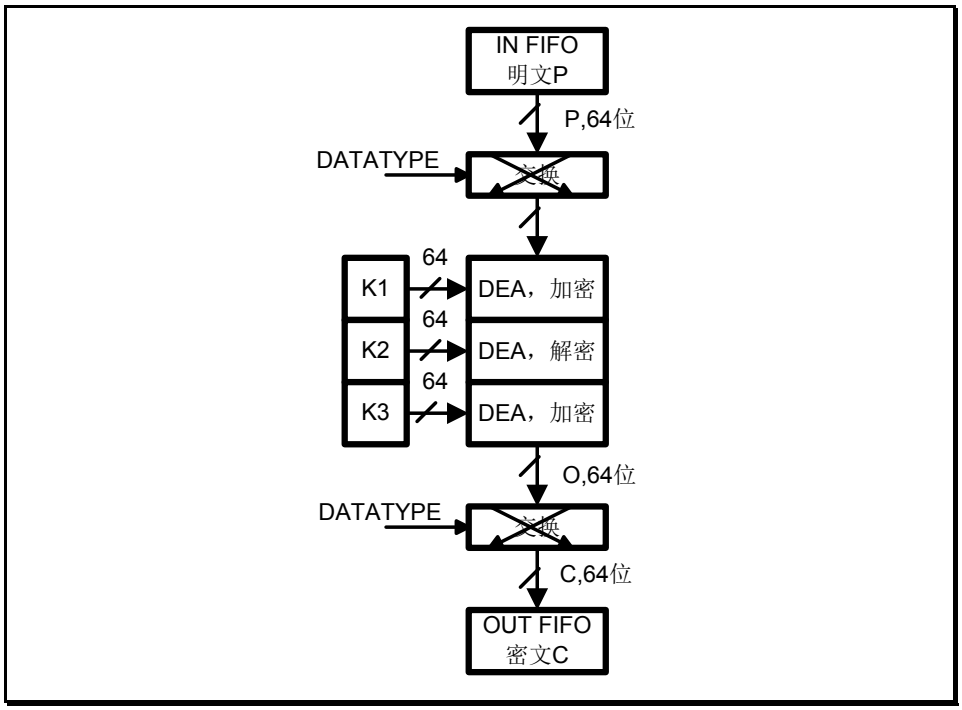


图 17-10 DES/TDES-ECB 模式加密

注：K = 密钥；C = 密文；I = 输入块；O = 输出块；P = 明文。

操作示例：

1. 设置 CRYPT_CON.CRYSEL = 1，选择算法类型为 DES/TDES
2. 设置 CRYPT_CON.ENCS = 1，选择加密
3. 设置 CRYPT_CON.TDES = 0 选择 DES，设置 CRYPT_CON.TDES = 1 选择 TDES，用户还可以设置 CRYPT_CON.DESKS 选择 TDES 使用 2 或者 3 个密钥。
4. 设置 CRYPT_CON.MODE = 0，选择 ECB 模式
5. 向 CRYPT_KEY0...CRYPT_KEY7 中填入密钥，其中 KEY0/1 = K1,KEY2/3 = K2 KEY4/5 = K3.
6. 向 CRYPT_DATA0/1 中写入需要加密的明文

7. 设置 CRYPT_CON.GO = 1，启动加密
8. 通过 CRYPT_IF.DONE 判断加密是否完成，如果 CRYPT_IF.DONE = 0，则加密完成
9. 从 CRYPT_DATA0/1 中读出已经加密的密文

17. 4. 2 DES/TDES-ECB模式解密

DES 或 TDES 是以 64bit 作为一个数据单元进行解密，下图介绍 DES 和 TDES 电子密码本（DES/TDES-ECB）模式中的解密。一个数据单元（64bit 密文 C）经过位/字节/半字交换后作为一个输入单元(I)，作为解密的密钥顺序是与加密过程中使用的密钥顺序是相反。该输入单元通过 DES 算法（DEA）在解密状态下使用 K3 进行解密处理。在 DES 算法处理后的输出会作为下一次 DES 算法（DEA）的输入，在加密状态下使用 K2 执行 DES 算法。在 DES 算法处理后的输出再次作为 DES 算法(DEA)的输入，在解密状态下使用 K1 再次执行 DES 算法。最后生成的 64bit 输出块(O)，再执行位/字节/半字交换之后，以明文（P）形式输出到数据寄存器。

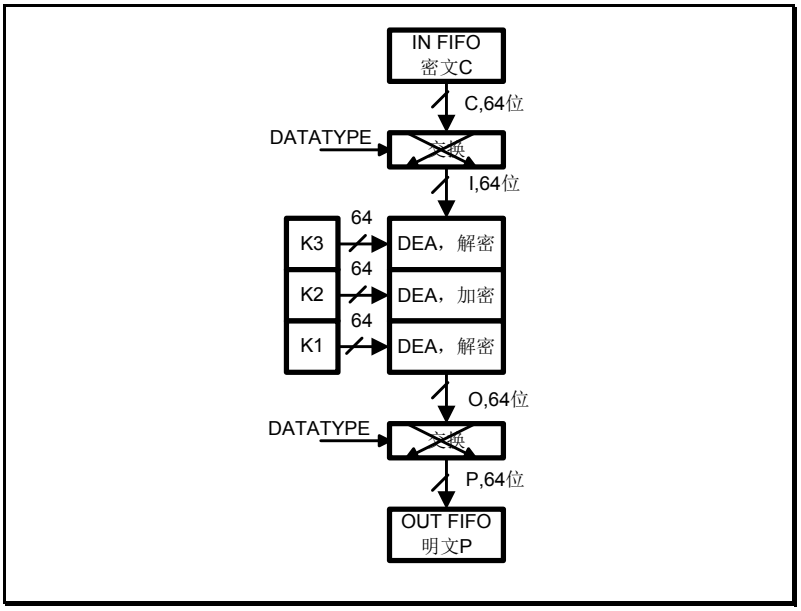


图 17-11 DES/TDES-ECB 模式解密

注：K = 密钥；C = 密文；I = 输入块；O = 输出块；P = 明文

操作示例：

1. 设置 CRYPT_CON.CRYSEL = 1，选择算法类型为 DES/TDES
2. 设置 CRYPT_CON.ENCS = 0，选择解密
3. 设置 CRYPT_CON.TDES = 0 选择 DES，设置 CRYPT_CON.TDES = 1 选择 TDES，用户还可以设置 CRYPT_CON.DESKS 选择 TDES 使用 2 或者 3 个密钥。
4. 设置 CRYPT_CON.MODE = 0，选择 ECB 模式
5. 向 CRYPT_KEY0...CRYPT_KEY7 中填入密钥，其中 KEY0/1 = K1,KEY2/3 = K2 KEY4/5 = K3
6. 向 CRYPT_DATA0/1 中写入需要解密的密文

7. 设置 CRYPT_CON.GO = 1，启动解密
8. 通过 CRYPT_IF.DONE 判断解密是否完成，如果 CRYPT_IF.DONE = 0，则解密完成
9. 从 CRYPT_DATA0/1 中读出已经解密的明文

17. 4. 3 DES/TDES-CBC模式加密

DES 或 TDES 是以 64 位作为一个数据单元进行加密，下图介绍了 DES 和三重 TDES 密码分组链接（DES/TDES-CBC）模式加密。该模式首先将明文消息分成多个 64bit 数据明文单元。在 CBC 加密过程中，一个数据单元经过执行位/字节/半字交换后作为明文单元（P1），通过与一个 64 位初始化向量（IV）进行异或运算（IV^P1 ），作为第一个输入单元 （I1）。该输入单元通过 DES 算法（DEA）在加密状态下使用 K1 进行加密处理。在 DES 算法处理后的输出会作为下一次 DES 算法（DEA）的输入，在解密状态下使用 K2 执行 DES 算法。在 DES 算法处理后的输出再次作为 DES 算法（DEA）的输入，在加密状态下使用 K3 再次执行 DES 算法。生成的 64 位输出单元（O1）将直接用作密文（C1），即 C1 = O1。然后，第一个密文单元与第二个明文数据单元进行异或运算（C1^P2），从而生成第二个输入单元(I2)。第二个输入单元通过以上 CBC 处理而生成第二个密文单元。加密处理会不断将后续密文单元和明文单元链接到一起，直到消息中所有的明文单元都加密完成为止。如果消息中最后的数据单元不是一个 64 位数据单元，则由应用程序按照一定规则对不完整数据单元进行加密。

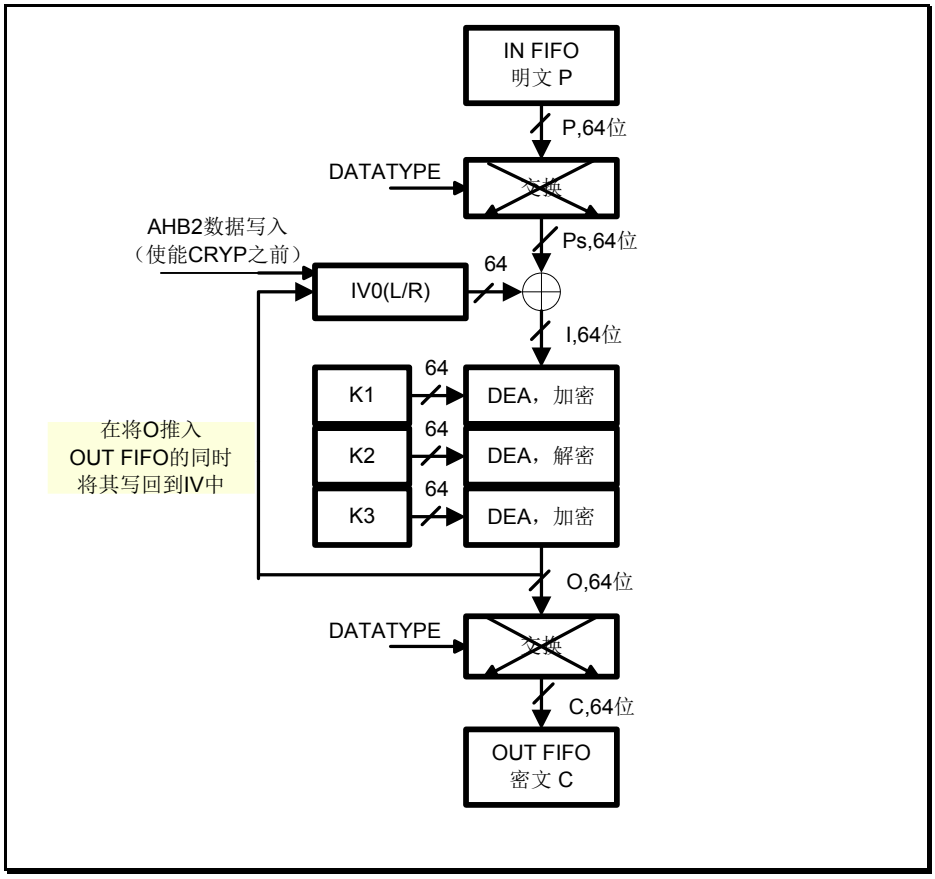


图 17-12 DES/TDES-CBC 模式加密

注：K = 密钥；C = 密文；I = 输入块；Ps = 交换前（解码时）或交换后（编码时）的明文；P = 明文；IV = 初始化向量。

操作示例：

1. 设置 CRYPT_CON.CRYSEL = 1，选择算法类型为 DES/TDES
2. 设置 CRYPT_CON.ENCS = 1，选择加密
3. 设置 CRYPT_CON.TDES = 0 选择 DES，设置 CRYPT_CON.TDES = 1 选择 TDES，用户还可以设置 CRYPT_CON.DESKS 选择 TDES 使用 2 或者 3 个密钥。
4. 设置 CRYPT_CON.MODE = 1，选择 CBC 模式
5. 向 CRYPT_KEY0...CRYPT_KEY7 中填入密钥，其中 KEY0/1 = K1,KEY2/3 = K2 KEY4/5 = K3.
6. 向 CRYPT_IV0/1 填入初始向量
7. 向 CRYPT_DATA0/1 中写入需要加密的明文
8. 设置 CRYPT_CON.GO = 1，启动加密
9. 通过 CRYPT_IF.DONE 判断加密是否完成，如果 CRYPT_IF.DONE = 0，则加密完成
10. 从 CRYPT_DATA0/1 中读出已经加密的密文
11. 回到步骤 7 继续进行加密
12. 所有加密完成

17. 4. 4 DES/TDES-CBC模式解密

DES 或 TDES 是以 64 位作为一个数据单元进行解密，下图介绍了 DES 和三重 TDES 密码分组链接（DES/TDES-CBC）模式解密。该模式首先将密文消息分成多个 64 位数据密文单元，第一个密文单元（C1）经过位/字节/半字交换后作为一个输入单元（I1），而作为解密的密钥顺序是与加密过程中使用的密钥顺序是相反。该输入单元（I1）通过 DES 算法（DEA）在解密状态下使用 K3 进行解密处理。在 DES 算法处理后的输出会作为下一次 DES 算法（DEA）的输入，在加密状态下使用 K2 执行 DES 算法。在 DES 算法处理后的输出再次作为 DES 算法（DEA）的输入，在解密状态下使用 K1 再次执行 DES 算法。最后生成的 64 位输出单元（O1），再与 IV 进行异或运算（O1^IV），从而生成第一个明文单元。然后，第二个密文单元作为下一个输入单元，经过以上 DES 解密处理，生成的输出单元再与第一个密文单元进行异或运算（O2^C1），从而生成第二个明文数据单元（P2）。依次进行解密，直到消息中所有的密文单元都解密完成为止。

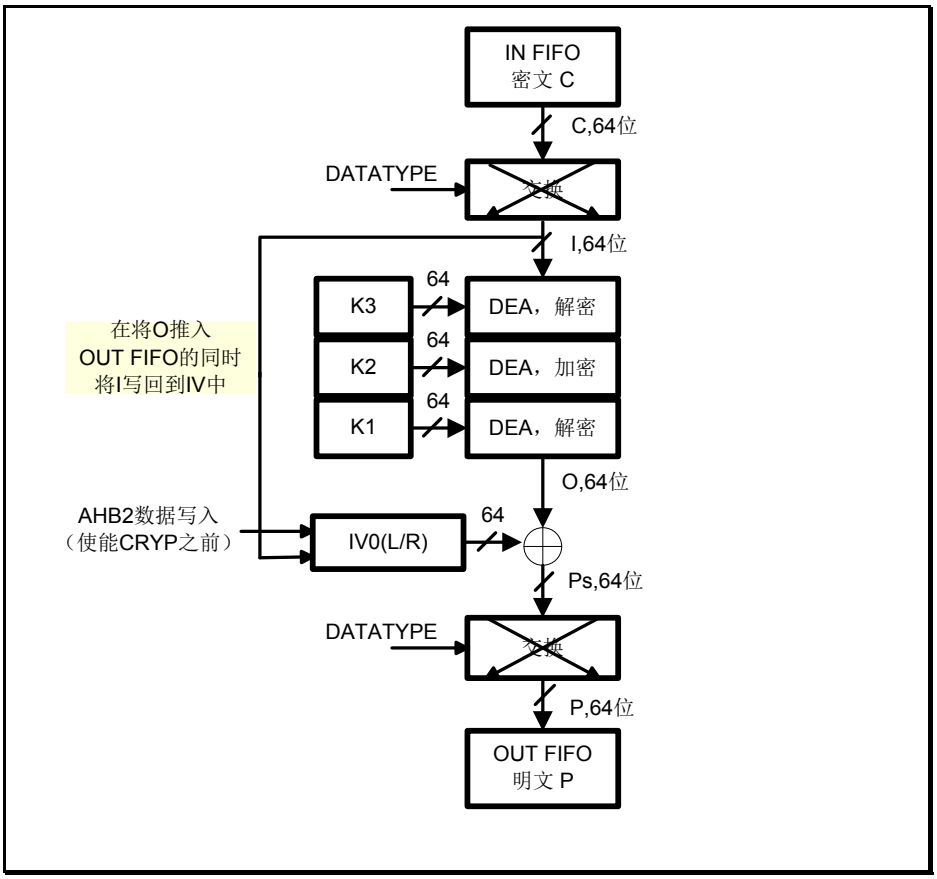


图 17-13 DES/TDES-CBC 模式解密

注：K = 密钥；C = 密文；I = 输入块；Ps = 交换前（解码时）或交换后（编码时）的明文；P = 明文；IV = 初始化向量。

操作示例：

1. 设置 CRYPT_CON.CRYSEL = 1，选择算法类型为 DES/TDES
2. 设置 CRYPT_CON.ENCS = 0，选择解密
3. 设置 CRYPT_CON.TDES = 0 选择 DES，设置 CRYPT_CON.TDES = 1 选择 TDES，用户还可以设置 CRYPT_CON.DESKS 选择 TDES 使用 2 或者 3 个密钥。
4. 设置 CRYPT_CON.MODE = 0，选择 ECB 模式
5. 向 CRYPT_KEY0...CRYPT_KEY7 中填入密钥，其中 KEY0/1 = K1,KEY2/3 = K2 KEY4/5 = K3
6. 向 CRYPT_IV0/1 填入初始向量
7. 向 CRYPT_DATA0/1 中写入需要解密的密文
8. 设置 CRYPT_CON.GO = 1，启动解密
9. 通过 CRYPT_IF.DONE 判断解密是否完成，如果 CRYPT_IF.DONE = 0，则解密完成
10. 从 CRYPT_DATA0/1 中读出已经解密的明文
11. 回到步骤 7 继续进行解密
12. 所有解密完成

17. 4. 5 FIFO使用

为了用户使用方便以及在加解密过程中使用 DMA，在 CRYPT 模块中添加了 FIFO 模式，在配置成 FIFO 模式后，可以通过寄存器 CRYPT_FIFO 进行数据输入输出操作，当 AES 运算时，通过 FIFO 写入 4 个 Word 后自动触发 AES 运算；当 DES 运算时，FIFO 写入 2 个 Word 后自动触发 DES/TDES 运算。

另外配置 FIFOODR 可选择 FIFO 写入和读出的顺序。AES 模式时，FIFOODR 为 0 则优先读出 RES0 或写入 DATA0，FIFOODR 为 1 则优先读出 RES3 或写入 DATA3。DES 和 TDES 模式时，FIFOODR 为 0 则优先读出 RES0 或写入 DATA0，FIFOODR 为 1 则优先读出 RES1 或写入 DATA1。

17. 4. 6 使用DMA传输数据

在使用 DMA 传输数据前需配置成 FIFO 模式，并且在将 DMAEN 置 1 的同时触发 DMA 传输。DMA 传输需将读数据和写数据配置在不同的通道上，硬件自动分别控制读数据和写数据的 DMA 请求信号，直至所需数据数量全部转换完成。

使用 DMA 传输必须配置数据总量为单次转换字数的整数倍。

17. 5 特殊功能寄存器

17. 5. 1 寄存器列表

CRYPT 寄存器列表		
名称	偏移地址	描述
CRYPT_DATA0	000_H	CRYPT 数据寄存器 0
CRYPT_DATA1	004_H	CRYPT 数据寄存器 1
CRYPT_DATA2	008_H	CRYPT 数据寄存器 2
CRYPT_DATA3	$00C_H$	CRYPT 数据寄存器 3
CRYPT_KEY0	010_H	CRYPT 密钥寄存器 0
CRYPT_KEY1	014_H	CRYPT 密钥寄存器 1
CRYPT_KEY2	018_H	CRYPT 密钥寄存器 2
CRYPT_KEY3	$01C_H$	CRYPT 密钥寄存器 3
CRYPT_KEY4	020_H	CRYPT 密钥寄存器 4
CRYPT_KEY5	024_H	CRYPT 密钥寄存器 5
CRYPT_KEY6	028_H	CRYPT 密钥寄存器 6
CRYPT_KEY7	$02C_H$	CRYPT 密钥寄存器 7
CRYPT_IV0	030_H	CRYPT 初始向量寄存器 0
CRYPT_IV1	034_H	CRYPT 初始向量寄存器 1
CRYPT_IV2	038_H	CRYPT 初始向量寄存器 2
CRYPT_IV3	$03C_H$	CRYPT 初始向量寄存器 3
CRYPT_RES0	040_H	CRYPT 结果寄存器 0
CRYPT_RES1	044_H	CRYPT 结果寄存器 1
CRYPT_RES2	048_H	CRYPT 结果寄存器 2
CRYPT_RES3	$04C_H$	CRYPT 结果寄存器 3
CRYPT_CON	050_H	CRYPT 控制寄存器
CRYPT_IF	054_H	CRYPT 中断标志寄存器
CRYPT_IFC	058_H	CRYPT 中断标志清零寄存器
CRYPT_FIFO	$05C_H$	CRYPT FIFO 寄存器

17. 5. 2 寄存器描述

17. 5. 2. 1 CRYPT数据寄存器X（CRYPT_DATAx）（x = 0..3）

CRYPT 数据寄存器 X（CRYPT_DATAx）（x = 0..3）																															
偏移地址：000_H ~$00C_H$																															
复位值：00000000_00000000_00000000_00000000$_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAx																															

DATAx	Bit 31-0	R/W	**CRYPT 待处理数据** AES 数据寄存器共 4 个字（16 个字节）； DES 数据寄存器共 2 个字（8 个字节）； 加密前，写入明文数据；解密前，写入密文数据

17. 5. 2. 2 CRYPT密钥寄存器X（CRYPT_KEYx）（x = 0..7）

CRYPT 密钥寄存器 X（CRYPT_KEYx）（x = 0..7）																															
偏移地址：010_H ~$02C_H$																															
复位值：00000000_00000000_00000000_00000000$_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYx																															

KEYx	Bit 31-0	R/W	**CRYPT 密钥** AES 密钥寄存器可以是 4 个字（128 位） AES 密钥寄存器可以是 6 个字（192 位） AES 密钥寄存器可以是 8 个字（256 位） DES/3DES 密钥寄存器可以是 2 个字（56 位） 3DES 2Keys 密钥寄存器可以是 4 个字（112 位） 3DES 3Keys 密钥寄存器可以是 6 个字（168 位）

17. 5. 2. 3 CRYPT初始向量寄存器X（CRYPT_IVx）（x = 0..3）

CRYPT 初始向量寄存器 X（CRYPT_IVx）（x = 0..3）																															
偏移地址：030_H ~$03C_H$																															
复位值：00000000_00000000_00000000_00000000$_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVx																															

IVx	Bit 31-0	R/W	**CRYPT 初始向量** AES 在 CBC 模式时用 4 个初始向量寄存器，共 4 个字，DES 在 CBC 模式时用 2 个初始向量寄存器，共 2 个字 AES 在 CTR、GCM 模式用作累加器。

17. 5. 2. 4 CRYPT结果寄存器X（CRYPT_RESx）（x = 0..3）

CRYPT 结果寄存器 X（CRYPT_RESx）（x = 0..3）																															
偏移地址：040_H ~$04C_H$																															
复位值：00000000_00000000_00000000_00000000$_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESx																															

RESx	Bit 31-0	R	**CRYPT 结果** AES 结果寄存器共 4 个字（16 个字节）； DES 结果寄存器共 2 个字（8 个字节）； 加密后，读出密文数据；解密后，读出明文数据

17. 5. 2. 5 CRYPT控制寄存器（CRYPT_CON）

CRYPT 控制寄存器（CRYPT_CON）

偏移地址：50_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRYSEL	Reserved															RESCLR	DMAEN	FIFOODR	FIFOEN	DESKS	TDES	TYPE		IE	IVEN	MODE		AESKS		ENCS	GO

CRYSEL	Bit 31	R/W	**算法标准选择位** 0：AES 1：DES/TDES
Reserved	Bit 30-16	—	**保留**
RESCLR	Bit 15	W1	**结果寄存器清零位** 0：无操作 1：结果寄存器清零
DMAEN	Bit 14	R/W	**DMA 使能位** 0：禁止 1：使能
FIFOODR	Bit 13	R/W	**FIFO 顺序选择位** 0：DATA0/RES0 优先 1：DATA3/RES3 优先（AES）；DATA1/RES1 优先（DES/TDES）
FIFOEN	Bit 12	R/W	**FIFO 使能位** 0：禁止 1：使能 注1：使用FIFO后当写入数据量达到预定数量时可自动触发加解密，同时处理完成后可从FIFO中读出结果。 注2：FIFO写入的数据可在DATA寄存器中体现，读出的数据可在RES寄存器中体现
DESKS	Bit 11	R/W	**TDES 密钥选择位** 0：2 个密钥 1：3 个密钥
TDES	Bit 10	R/W	**TDES 使能位** 0：禁止 1：使能
TYPE	Bit 9-8	R/W	**数据类型选择位** 00：32 位数据，不交换 01：16 位数据或半字交换 10：8 位数据或字节交换 11：1 位数据或位串交换

IE	Bit 7	R/W	**中断使能位** 0：禁止 1：使能
IVEN	Bit 6	R/W	**IV 使能位** 0：禁止 1：使能 注：使能后应用于CBC模式的首次运算，完成后硬件自动清0
MODE	Bit 5-4	R/W	**模式选择位** 00：ECB 01：CBC 10：CTR（基于AES） 11：GCM（使用MULTH）
AESKS	Bit 3-2	R/W	**AES 密钥选择位** 00：128 位 01：192位 10：256位 11：保留
ENCS	Bit 1	R/W	**加解密选择位** 0：解密 1：加密
GO	Bit 0	W1	**CRYPT 启动触发位** 0：无操作 1：启动加密或解密

17. 5. 2. 6 CRYPT中断标志寄存器（CRYPT_IF）

CRYPT 中断标志寄存器（CRYPT_IF）

偏移地址：54_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-9	—	**保留**
DONE	Bit 8	R	**加解密完成标志位** 0：未开始或已完成 1：未完成
Reserved	Bit 7-3	—	**保留**
MULTHIF	Bit 2	R	**MULTH 中断标志位** 0：未完成 MULTH 运算或标志位已被清除 1：完成MULTH运算 注：该位由硬件自动置1，软件操作MULTHIFC清零。
DESIF	Bit 1	R	**DES 中断标志位** 0：未完成 DES 运算或标志位已被清除 1：完成DES运算 注：该位由硬件自动置1，软件操作DESIFC清零。
AESIF	Bit 0	R	**AES 中断标志位** 0：未完成 AES 运算或标志位已被清除 1：完成AES运算 注：该位由硬件自动置1，软件操作AESIFC清零。

17. 5. 2. 7 CRYPT中断标志清零寄存器（CRYPT_IFC）

CRYPT 中断标志清零寄存器（CRYPT_IFC）

偏移地址：58_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-3	—	**保留**
MULTHIFC	Bit 2	W1	**MULTH 中断标志位清零** 0：无操作 1：MULTHIF清零操作
DESIFC	Bit 1	W1	**DES 中断标志位清零** 0：无操作 1：DESIF清零操作
AESIFC	Bit 0	W1	**AES 中断标志位清零** 0：无操作 1：AESIF清零操作

17. 5. 2. 8 CRYPT FIFO寄存器（CRYPT_FIFO）

CRYPT FIFO 寄存器（CRYPT_FIFO）

偏移地址：$5C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

FIFO	Bit 31-0	R/W	**数据缓冲** 需通过FIFOEN位使能该寄存器的读写功能，运算前写入待处理数据，运算后读取结果

第18章 真随机数发生器（TRNG）

18. 1 概述

真随机数发生器（TRNG）可生产 1 位串行真随机数或 8/16/32 位并行真随机数。

18. 2 特性

- 支持可编程的随机数位宽
- 支持可编程的种子值
- 支持随机性修正模式
- 支持随机序列错误检测

18. 3 结构框图

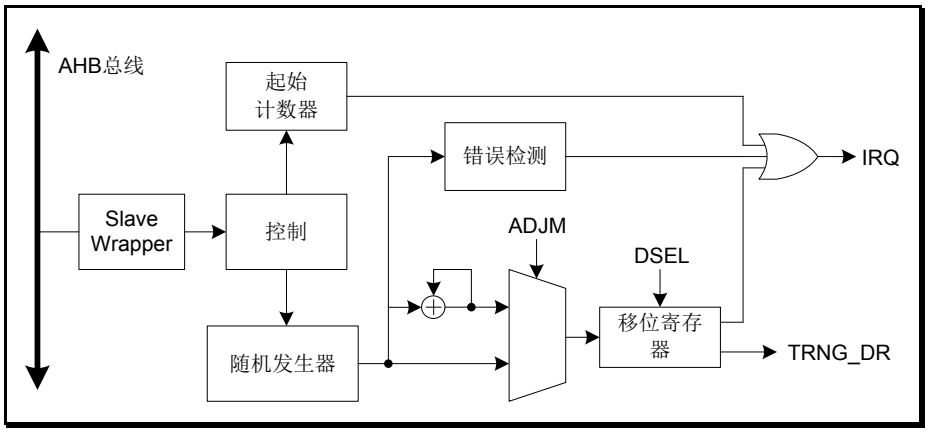


图 18-1 TRNG 结构框图

18.4 功能描述

18.4.1 初始化时间

TRNG 使能后需要经过一段时间初始化才可产生新的随机数，初始化时间通过 TRNG_CFGR 寄存器的 TSTART 位配置。根据当前 HCLK 时钟频率，计算出 TSTART 的值，使初始化时间不低于 1ms。

计算公式为：

$$T_{START} = T_{HCLK1} \times 2^{TSTART+1}$$

18.4.2 错误管理

出现随机序列错误时，TRNG_SR 的 SERR 位被硬件置 1，随机数产生被中断。若使能了序列错误中断，则产生该中断。如果 TRNG_DR 寄存器中有新产生的随机数，则不能使用该随机数。

应执行以下操作：置位 TRNG_IFCR 寄存器的 SERR 位，将 TRNGEN 位清零并置 1，以便重新初始化和重新启动 TRNG。

18.4.3 随机数生成时间

随机数生成时间由以下位共同决定：TRNG_CR 寄存器中的 DSEL 位、ADJM 位、POSTEN 位和 ADJC 位。

随机数生成时间公式为：

$T_{RNG} \approx 5 \times D \times P \times C^{M}$（单位 us）（误差范围+66.66%/-33.33%）

DSEL	系数 D	ADJM	系数 M	POSTEN	系数 P	ADJC	系数 C
00	1	0	0	0	1	00	2
01	8					01	4
10	16	1	1	1	2	10	6
11	32					11	8

表 18-1 随机数生成时间系数表

18.4.4 随机数种子

因为随机序列发生器来自模拟信号，电源和地上的干扰或环境温度的变化等都会导致随机性变差、输出序列的相邻位的自相关系数升高。通过设置一种随机的种子值，可能使随机序列的随机性能更加优化，降低自相关性。

种子的类型可选择为使用上一次产生的随机数或使用 TRNG_SEED 寄存器值。若使用 TRNG_SEED 寄存器值，则在每次读取新的随机数时同时更新 TRNG_SEED 寄存器。种子的类型也可选择固定为 0 或 1。

18.4.5 操作流程

要运行 TRNG，请按以下步骤操作：

1. 如果需要，使能中断（TRNG_IER 相应位置 1）。准备好随机数时或出现错误时生成中断。

2. 通过将 TRNG_CR 寄存器中的 TRNGEN 位置 1 使能随机数产生。这会激活随机发生器、移位寄存器和错误检测器。如果需要随机性更高的数据，可以将 POSTEN 位置 1，使能后处理模式、也可以将 ADJM 位置 1，使能修正模式，但相应的随机数生成时间会变长。在使能修正模式时，配置 ADJC 可改变修正模式系数、系数越大，随机性越高，但随机数生成时间也越长。

3. 每次中断时，检查确认未出现错误（TRNG_SR 寄存器中的 SERR 位应为 0，TRNG_IFR 寄存器中的 SERR 也为 0），并且随机数已准备就绪（TRNG_SR 寄存器中的 DAVLD 位为 1）。然后即可读取 TRNG_DR 寄存器中的内容。

按照 FIPS PUB（联邦信息处理标准出版物）140-2 的要求，将 TRNGEN 位置 1 后产生的第一个随机数不应使用，但应保存起来，与产生的下一个随机数进行比较。随后产生的每个随机数都需要与产生的上一个随机数进行比较。如果任何一对进行比较的数字相等，则测试失败（连续随机数发生器测试）。

操作流程示意图如下所示。

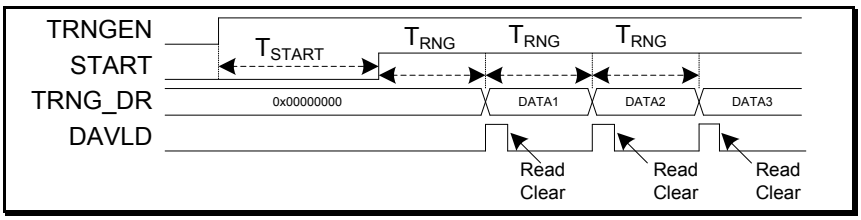


图 18-2 操作流程示意图

18. 5 特殊功能寄存器

18. 5. 1 寄存器列表

TRNG 寄存器列表		
名称	偏移地址	描述
TRNG_CR	0000_H	TRNG 控制寄存器
TRNG_SR	0004_H	TRNG 状态寄存器
TRNG_DR	0008_H	TRNG 数据寄存器
TRNG_SEED	$000C_H$	TRNG 种子寄存器
TRNG_CFGR	0010_H	TRNG 配置寄存器
TRNG_IER	0014_H	TRNG 中断使能寄存器
TRNG_IFR	0018_H	TRNG 中断标志寄存器
TRNG_IFCR	$001C_H$	TRNG 中断标志清零寄存器
TRNG_ISR	0020_H	TRNG 中断状态寄存器

18. 5. 2 寄存器描述

18. 5. 2. 1 TRNG控制寄存器（TRNG_CR）

TRNG 控制寄存器（TRNG_CR）																															
偏移地址：00_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														ADJC		Reserved				SDSEL		DSEL		Reserved				POSTEN	TRNGSEL	ADJM	TRNGEN

Reserved	Bit 31-18	—	**保留**
ADJC	Bit 17-16	R/W	**TRNG 修正系数** 注：仅在修正模式使能时有效，修正系数越大，数据的随机性越高，但随机数据生成时间会越长
Reserved	Bit 15-12	—	**保留**
SDSEL	Bit 11-10	R/W	**TRNG 种子类型选择位** 00：固定为 0 01：固定为 1 10：使用上一次产生随机数 11：使用 SEED 寄存器值
DSEL	Bit 9-8	R/W	**TRNG 数据类型选择位** 00：1 位 01：8 位 10：16 位 11：32 位 注：数据位宽越短，随机数生成时间越短，但累积生成相同 Bit 量的时间是固定的
Reserved	Bit 7-4	—	**保留**
POSTEN	Bit 3	R/W	**数据后处理使能位** 0：禁止 1：使能
TRNGSEL	Bit 2	R/W	**TRNG 源选择位** 0：TRNG0 1：TRNG1
ADJM	Bit 1	R/W	**TRNG 修正模式使能位** 0：禁止 1：使能 注：使能修正模式可提高数据的随机性，但相应地随机数据的生成时间会变长
TRNGEN	Bit 0	R/W	**TRNG 使能位** 0：禁止 1：使能

18. 5. 2. 2 TRNG状态寄存器（TRNG_SR）

TRNG 状态寄存器（TRNG_SR）

偏移地址：04_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-4	—	保留
OVER	Bit 3	R	**序列溢出状态位（仅用于 TRNG0）** 0：当前序列无溢出 1：当前序列有溢出
SERR	Bit 2	R	**序列错误状态位** 0：当前序列无错误 1：当前序列有错误 注：当硬件检测到超过64个连续位具有相同值（0或1），或者超过32个连续交替的0和1（01010101……01）则表示序列出错
DAVLD	Bit 1	R	**当前数据有效状态位** 0：数据未生成或已被读取 1：数据有效 注：该位在模块关闭或随机数被读取时清零
START	Bit 0	R	**初始化状态位** 0：初始化未完成 1：初始化完成 注：该位在模块关闭时被清零

18. 5. 2. 3 TRNG数据寄存器（TRNG_DR）

TRNG 数据寄存器（TRNG_DR）																															
偏移地址：08_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA																															

DATA	Bit 31-0	R	**生成的随机数** 注1：当状态寄存器中数据有效位为有效时才可读取 注2：该数据实际有效宽度为数据类型选择的宽度

18. 5. 2. 4 TRNG种子寄存器（TRNG_SEED）

TRNG 种子寄存器（TRNG_SEED）																															
偏移地址：$0C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEED																															

SEED	Bit 31-0	R/W	**种子值** 注1：种子值可在每次生成随机数时进行更新，也可固定为某值 注2：该数据实际有效宽度为数据类型选择的宽度

18. 5. 2. 5 TRNG配置寄存器（TRNG_CFGR）

TRNG 配置寄存器（TRNG_CFGR）

偏移地址：10_H

复位值：$00000001\_11111111\_00000111\_00000111_B$

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

Reserved	Bit 31-25	—	**保留**
TOPLMT	Bit 24-16	R/W	**随机数上限配置（仅用于 TRNG0）** 默认值 0x1FF
Reserved	Bit 15-12	—	**保留**
CKDIV	Bit 11-8	R/W	**时钟分频配置位（仅用于 TRNG0）** $Freq_{CK} = Freq_{HCLK1} / 2^{CKDIV+1}$
Reserved	Bit 7-3	—	**保留**
TSTART	Bit 2-0	R/W	**初始化时间配置位** $T_{START} = T_{RNG} \times 2^{TSTART+1}$ 注：根据当前TRNG数据流的频率值配置初始化时间，需保证初始化时间不少于1ms

18. 5. 2. 6 TRNG中断使能寄存器（TRNG_IER）

TRNG 中断使能寄存器（TRNG_IER）

偏移地址：14_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-3	—	**保留**
SERR	Bit 2	R/W	**序列错误中断使能位** 0：禁止 1：使能
DAVLD	Bit 1	R/W	**数据有效中断使能位** 0：禁止 1：使能
START	Bit 0	R/W	**初始化完成中断使能位** 0：禁止 1：使能

18. 5. 2. 7 TRNG中断标志寄存器（TRNG_IFR）

TRNG 中断事件寄存器（TRNG_IFR）

偏移地址：18_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-3	—	保留
SERR	Bit 2	R	**序列错误状态位** 0：未检测到序列错误或标志位已被清除 1：已检测到序列错误 注1：当硬件检测到超过64个连续位具有相同值（0或1），或者超过32个连续交替的0和1（01010101……01）则表示序列出错 注2：该位通过操作TRNG_ICFR清零。
DAVLD	Bit 1	R	**数据有效标志位** 0：数据无效或标志位已被清除 1：数据有效 注：该位通过操作 TRNG_ICFR 清零。若使用该位的 DMA 功能，DMA 应答后自动将该位清零。
START	Bit 0	R	**初始化完成标志位** 0：初始化未完成或标志位已被清除 1：初始化完成 注：该位通过操作TRNG_ICFR清零。

18. 5. 2. 8 TRNG中断标志清零寄存器（TRNG_IFCR）

TRNG 中断事件清零寄存器（TRNG_IFCR）

偏移地址：$1C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-3	—	保留
SERR	Bit 2	W1	**序列错误状态位清零** 0：无操作 1：序列错误状态位清零 注：该位写1清除，写0无效
DAVLD	Bit 1	W1	**数据有效标志位清零** 0：无操作 1：数据有效标志位清零 注：该位写 1 清除，写 0 无效
START	Bit 0	W1	**初始化完成标志位清零** 0：无操作 1：初始化完成标志位清零 注：该位写1清除，写0无效

18. 5. 2. 9 TRNG中断状态寄存器（TRNG_ISR）

TRNG 中断状态寄存器（TRNG_ISR）

偏移地址：20_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-3	—	保留
SERR	Bit 2	R	**序列错误中断状态位** 0：未发生中断 1：已发生中断
DAVLD	Bit 1	R	**数据有效中断状态位** 0：未发生中断 1：已发生中断。
START	Bit 0	R	**初始化完成中断状态位** 0：未发生中断 1：已发生中断

第19章 运算加速器（CALC）

19. 1 概述

运算加速器（CALC）可以执行平方根运算加速。

19. 2 特性

- 支持最大 32 位无符号数平方根运算

19. 3 结构框图

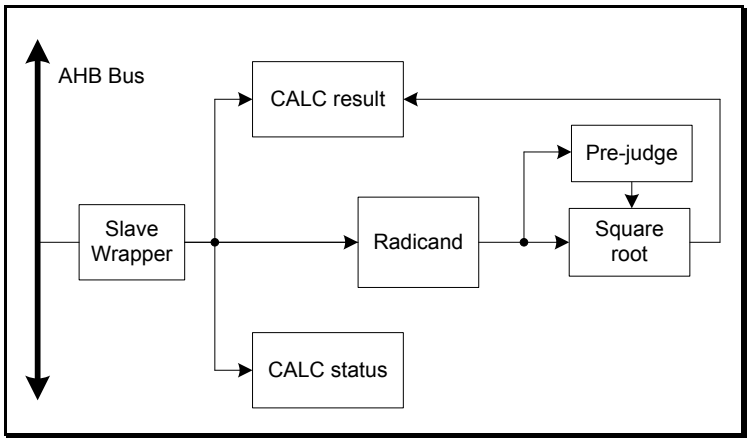


图 19-1 CALC 结构框图

19.4 功能描述

19.4.1 平方根运算

19.4.1.1 算法概述

无符号数平方根算法可对最大 32 位的无符号数进行平方根运算，运算结果最大为 16 位无符号数。若理论平方根值中含有小数部分，则硬件在计算时会舍去小数，即向 0 的方向取最大整数。

硬件运算电路中带有预判决功能，可根据被开方数 RCND 的数量级，自动选取最小的计算时间。

19.4.1.2 使用说明

当被开方数寄存器 RDCND 发生写入动作时，平方根运算即开始，开始时平方根运算标志位 CALC_SQRTSR.BUSY 被置位。当软件检测到 CALC_SQRTSR.BUSY 被硬件清零时，表示平方根运算已经完成。通过读取平方根运算结果寄存器 SQRTRES 可获得被开方数的平方根近似值。

若当运算还未完成，被开方数寄存器 RDCND 发生写入动作时，硬件会重新开始新的运算，原先的运算结果将被丢弃。

为使得运算的结果更精确，在操作上可采取移位的方式来减小运算的误差。

例如，需要运算 X 的平方根，由于 X 较小，若直接写入 RDCND 中，产生的结果误差较大。可以先将 X 进行左移 n 位（n 需为偶数），将 X 左移后可得新的被开方数即 $X' = X * 2^n$，将 X' 写入被开方数寄存器 RDCND，得到运算结果 Y'，可知 $Y' = \sqrt{X'} = \sqrt{(X * 2^n)} = 2^{n/2} * \sqrt{X}$，可将 SQRTRES 中的结果右移 n/2 位后，即得到 X 的平方根 $Y = \sqrt{X} = Y' / 2^{n/2}$ 。

原先 X 允许的位数 m 最大可至 32 位，当 X 的实际位数没有 32 位时，可适当的调整 n 的位数以最大程度的利用平方根运算器的性能。n 值越大，运算结果越精确。

以计算 2 的平方根为例。$\sqrt{2}$ = 1.4142135623731。

Radicand (Hex)	Radicand 格式	Result(Hex)	Result(Dec)	误差（%）
0x0000 0002	m=32, n=0	0x0000 0001	1.0	-29.289
0x0000 0020	m=28, n=4	0x0000 0005	1.25	-11.612
0x0000 0200	m=24, n=8	0x0000 0016	1.3750	-2.773
0x0000 2000	m=20, n=12	0x0000 005A	1.406250	-0.563
0x0002 0000	m=16, n=16	0x0000 016A	1.41406250	-0.011
0x0020 0000	m=12, n=20	0x0000 05A8	1.41406250	-0.011
0x0200 0000	m=8, n=24	0x0000 16A0	1.41406250	-0.011
0x2000 0000	m=4, n=28	0x0000 5A82	1.4141845703	-0.002
0x8000 0000	m=2, n=30	0x0000 B504	1.4141845703	-0.002

表 19-1 平方根运算误差示例

19. 4. 1. 3 完成时间

平方根算法可根据被开方数 RDCND 中输入的数值进行预判决，硬件针对被开方数不同量级自动决定运算时间，不同量级的运算时间可根据下表所示。

RDCND[31:0]	运算时间（BUSY=1 的时钟个数）
1xxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx 01xx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	17
001x_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx 0001_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	16
0000_1xxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx 0000_01xx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	15
0000_001x_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx 0000_0001_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	14
0000_0000_1xxx_xxxx_xxxx_xxxx_xxxx_xxxx 0000_0000_01xx_xxxx_xxxx_xxxx_xxxx_xxxx	13
0000_0000_001x_xxxx_xxxx_xxxx_xxxx_xxxx 0000_0000_0001_xxxx_xxxx_xxxx_xxxx_xxxx	12
0000_0000_0000_1xxx_xxxx_xxxx_xxxx_xxxx 0000_0000_0000_01xx_xxxx_xxxx_xxxx_xxxx	11
0000_0000_0000_001x_xxxx_xxxx_xxxx_xxxx 0000_0000_0000_0001_xxxx_xxxx_xxxx_xxxx	10
0000_0000_0000_0000_1xxx_xxxx_xxxx_xxxx 0000_0000_0000_0000_01xx_xxxx_xxxx_xxxx	9
0000_0000_0000_0000_001x_xxxx_xxxx_xxxx 0000_0000_0000_0000_0001_xxxx_xxxx_xxxx	8
0000_0000_0000_0000_0000_1xxx_xxxx_xxxx 0000_0000_0000_0000_0000_01xx_xxxx_xxxx	7
0000_0000_0000_0000_0000_001x_xxxx_xxxx 0000_0000_0000_0000_0000_0001_xxxx_xxxx	6
0000_0000_0000_0000_0000_0000_1xxx_xxxx 0000_0000_0000_0000_0000_0000_01xx_xxxx	5
0000_0000_0000_0000_0000_0000_001x_xxxx 0000_0000_0000_0000_0000_0000_0001_xxxx	4
0000_0000_0000_0000_0000_0000_0000_1xxx 0000_0000_0000_0000_0000_0000_0000_01xx	3
0000_0000_0000_0000_0000_0000_0000_00xx	2

表 19-2 平方根运算时间表

19. 5 特殊功能寄存器

19. 5. 1 寄存器列表

CALC 寄存器列表		
名称	偏移地址	描述
CALC_SQRTSR	000_H	平方根运算状态寄存器
CALC_RDCND	004_H	被开方数寄存器
CALC_SQRTRES	008_H	平方根运算结果寄存器

19. 5. 2 寄存器描述

19. 5. 2. 1 平方根运算状态寄存器（CALC_SQRTSR）

平方根运算状态寄存器（CALC_SQRTSR）																															
偏移地址：00_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															BUSY

Reserved	Bit 31-1	—	**保留**
BUSY	Bit 0	R	**平方根运算状态位** 0：完成 1：进行中

19. 5. 2. 2 被开方数寄存器（CALC_RDCND）

被开方数寄存器（CALC_RDCND）																															
偏移地址：04_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RADICAND																															

RADICAND	Bit 31-0	R/W	**被开方数** 32 位无符号被开方数

19. 5. 2. 3 平方根运算结果寄存器（CALC_SQRTRES）

平方根运算结果寄存器（CALC_SQRTRES）																															
偏移地址：08_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																RESULT															

Reserved	Bit 31-16	—	**保留**
RESULT	Bit 15-0	R	**平方根运算结果值** 16 位平方根运算结果

第20章 高级控制定时器（AD16C4T）

20. 1 概述

高级控制定时器（AD16C4T）是一个功能强大、配置灵活的定时器模块，它包含一个 16-bit，定时器,具有定时、计数、脉冲输入信号测量（输入捕获）、产生特定 PWM 波（输出比较）等功能。

20. 2 特性

- 16 位递增，递减，递增/递减自动加载计数器
- 16 位可编程预分频器，可在定时器运行中对计数器工作时钟进行 1 到 65536 间的任意分频
- 带有四个独立通道，每个通道支持以下功能
  - 输入捕获
  - 输出比较
  - PWM 输出
  - 单脉冲输出
- 通道 1~3 支持互补输出，死区时间可配
- 在给定数目的计数周期之后更新重复计数寄存器
- 支持刹车功能，刹车后定时器输出状态可控
- 支持中断/DMA：
  - 更新事件：计数器上溢/下溢，计数器初始化
  - 触发事件
  - 换相事件
  - 输入捕获
  - 输出比较
  - 刹车输入
- 支持增量（正交）编码及霍尔电路
- 通过外设互联（PIS）可支持与片上其他定时器的互联工作

20. 3 结构框图

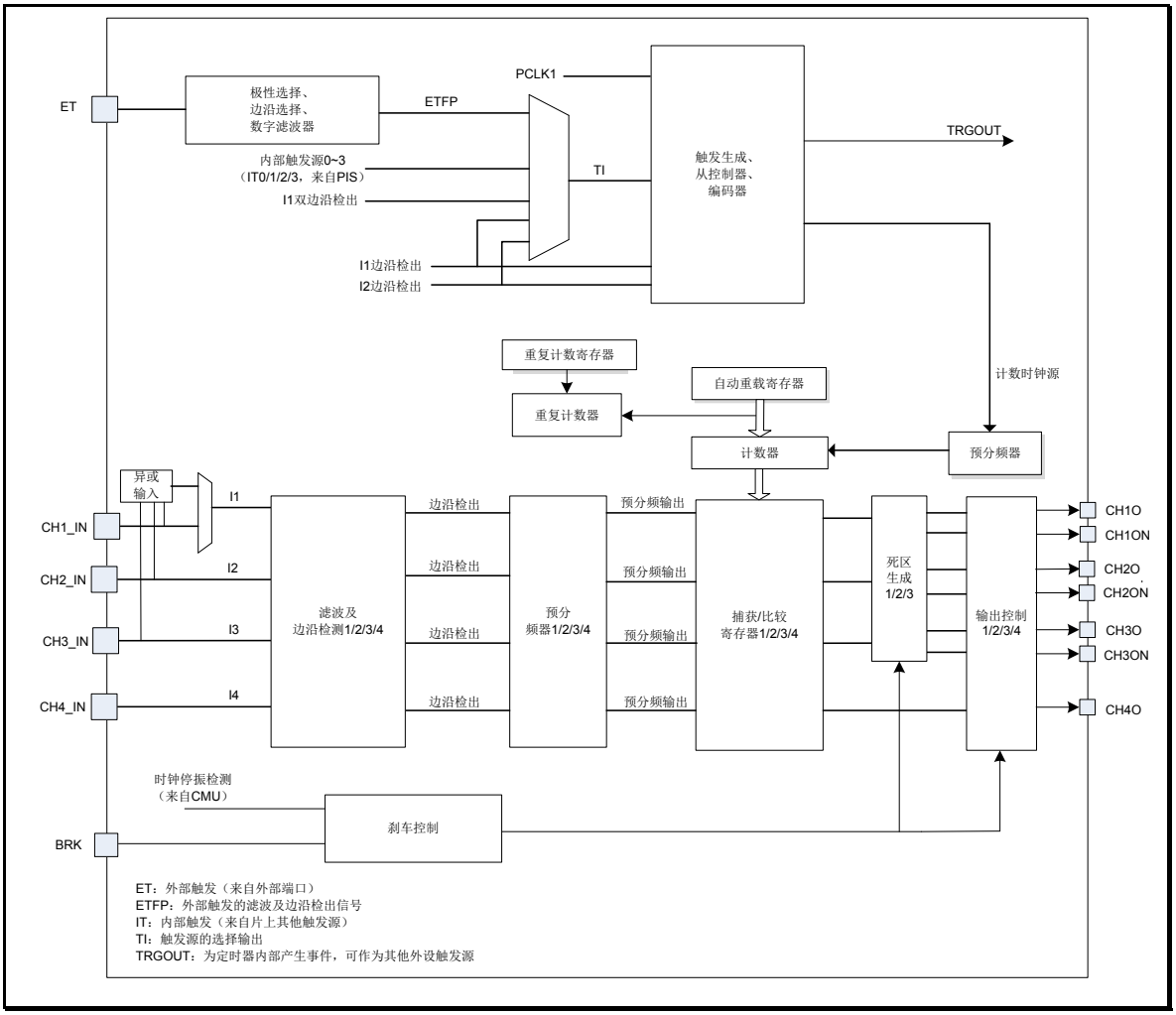


图 20-1　高级定时器结构框图

20.4 功能描述

20.4.1 预分频器

定时器包含一个 16-bit 的计数器（AD16C4Tn_COUNT），计数时钟由预分频寄存器（AD16C4Tn_PRES）进行分频。计数周期由自动重载计数器（AD16C4Tn_AR）设定。重复计数寄存器则可指定计数周期数目（AD16C4Tn_REPAR）。

自动重载寄存器（AD16C4Tn_AR）是一个可缓存的寄存器。当 AD16C4Tn_CON1 寄存器的 ARPEN 位复位时，AD16C4Tn_AR 寄存器重载功能失效，AD16C4Tn_AR 就是有效寄存器；ARPEN 置位时，AD16C4Tn_AR 寄存器具有重载功能，产生更新事件（UEV）时，加载值（AD16C4Tn_AR 寄存器值）更新到影子寄存器后才生效。

当 AD16C4Tn_CON1 寄存器中 DISUE 位为 0 时，计数器计数上溢（或递减下溢）时会产生更新事件（UEV）。同样，软件方式也可产生更新事件。AD16C4Tn_CON1 寄存器的 CNTEN 置位时，计数器开始计数。

注：计数器在 CNTEN 位置位 1 个时钟周期后开始计数。

预分频器可对定时器工作时钟进行 AD16C4Tn_PRES 寄存器值+1 次分频。由于 AD16C4Tn_PRES 是一个可重载寄存器，因此，定时器工作时可以对该寄存器进行修改，修改值在下次更新事件（UEV）后有效。

下图给出了定时器运行过程中改变预分频值时计数器的计数情况。

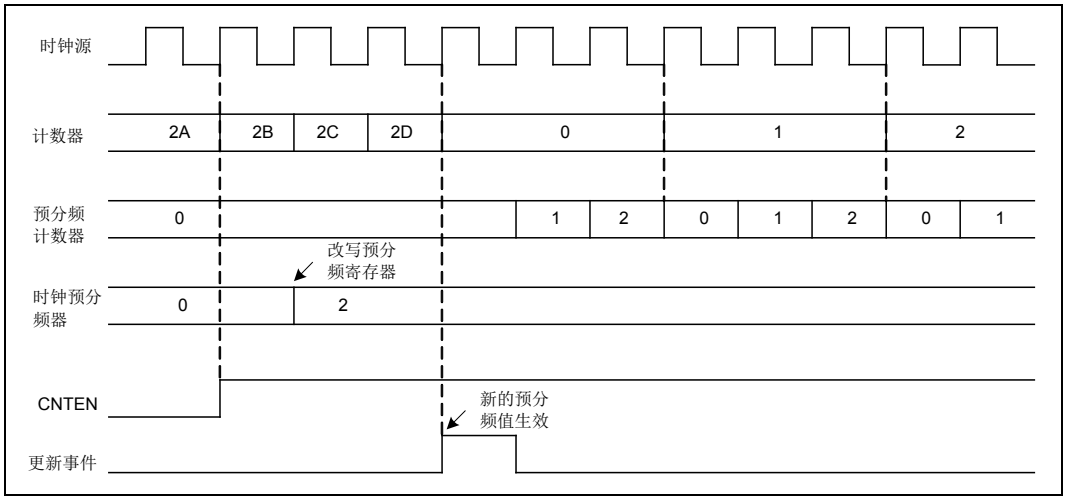


图 20-2 预分频值计数时序图

20. 4. 2 重复计数器

重复计数器用于控制发生多少次上溢或下溢后产生更新事件。

重复计数器递减：

- ◇ 递增模式的每次上溢
- ◇ 递减模式的每次下溢
- ◇ 中心对齐模式时，计数器上溢与下溢。中心对齐模式限制了最大重复次数为 128 个 PWM 周期，每个 PWM 周期内可更新两次占空比。

AD16C4Tn_REPAR 寄存器是一个可缓存寄存器。软件（置位 AD16C4Tn_SGE 寄存器中的 SGU 位）或硬件从机模式控制方式产生更新事件时，无论重复计数器为何值，AD16C4Tn_REPAR 寄存器中值会立即更新到重复计数器的影子寄存器中。

中心对齐模式下，REPAR 中值为奇数时，更新事件是在上溢或下溢时产生，取决于何时写 REPAR 寄存器及何时开始计数。若在启动计数器前写 REPAR，则上溢时产生 UEV，反之则在下溢时产生 UEV。

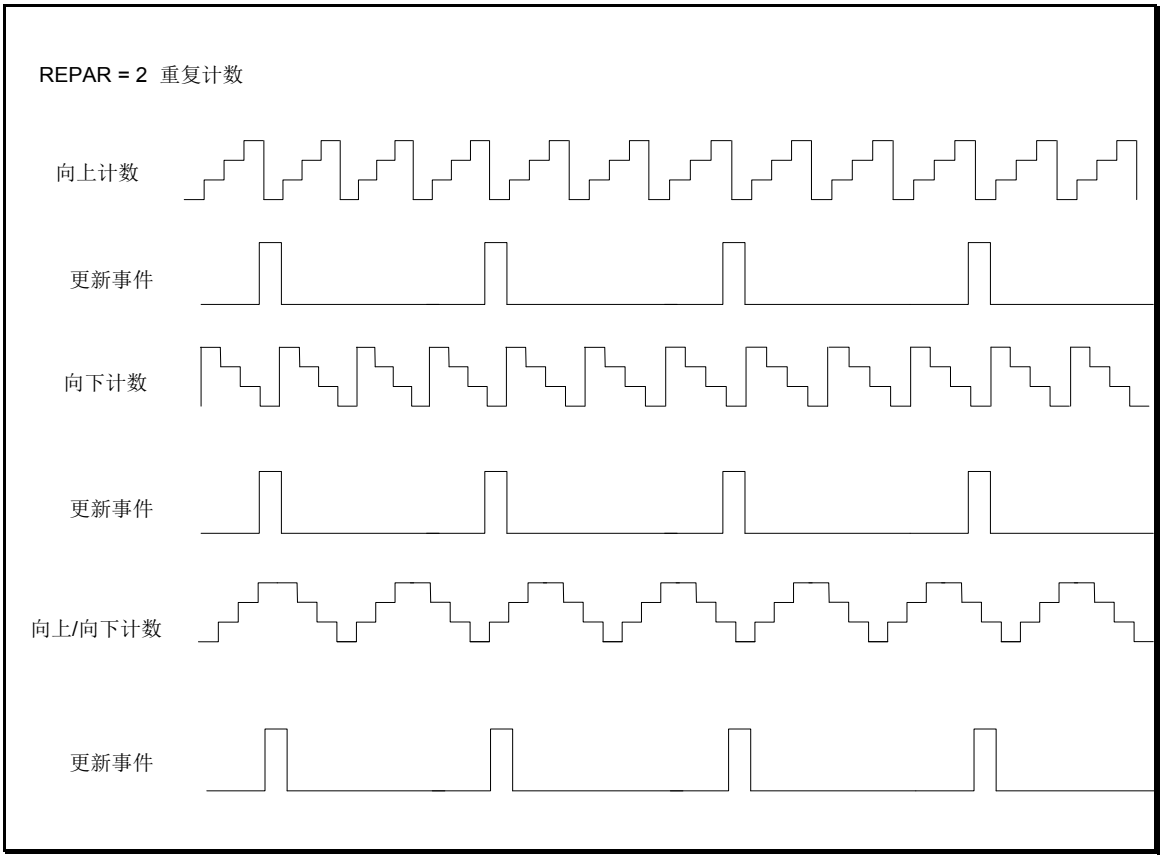


图 20-3 重复计数器工作模式

注意：置位 AD16C4Tn_SGE 寄存器中的 SGU 位也可以产生更新事件。

20. 4. 3 时钟源

计数器工作时钟可以选择内部时钟(INT_CLK)、外部时钟源 1(I1、I2)、外部时钟源 2(ET)，内部触发输入（IT1、IT2、IT3、IT4）

20. 4. 3. 1 内部时钟源（INT_CLK）

若从模式控制器被关闭(AD16C4Tn_SMCON 寄存器内，SMODS= "000")，则 CNTEN，AD16C4Tn_CON1.DIRSEL 与 AD16C4Tn_SGE.SGU 位为实际控制位，这些位只能软件修改（SGU 位除外，仍硬件自动清除）。一旦 CNTEN 位被写为'1'，预分频器就由内部 INT_CLK 提供时钟。

下图给出了通常模式下控制电路和递增计数的情况，没有分频。

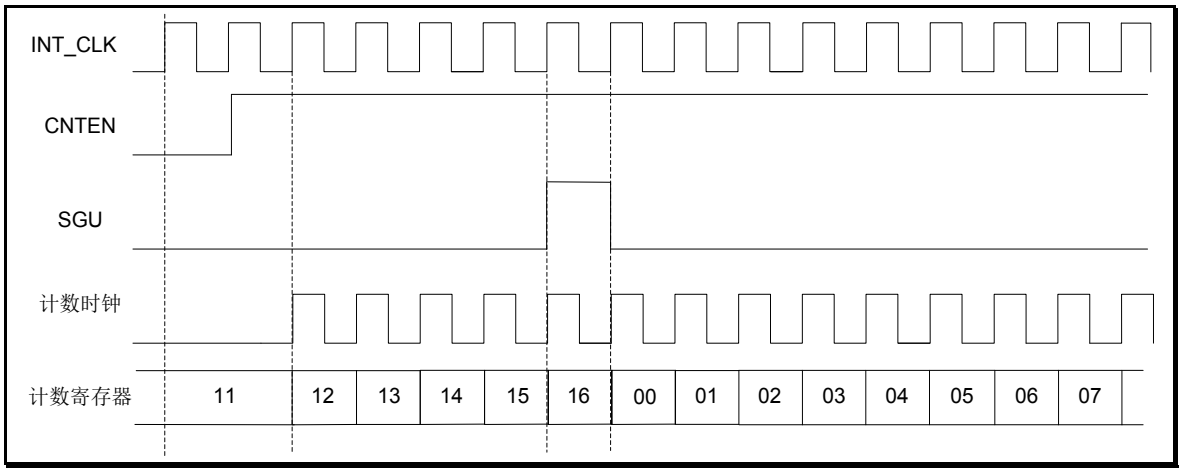


图 20-4 采用内部时钟计数

20. 4. 3. 2 外部时钟源 1

AD16C4Tn_SMCON 寄存器的 SMODS= "111"时，可选择外部时钟源 1。计数器可根据选定的上升沿或下降沿计数。

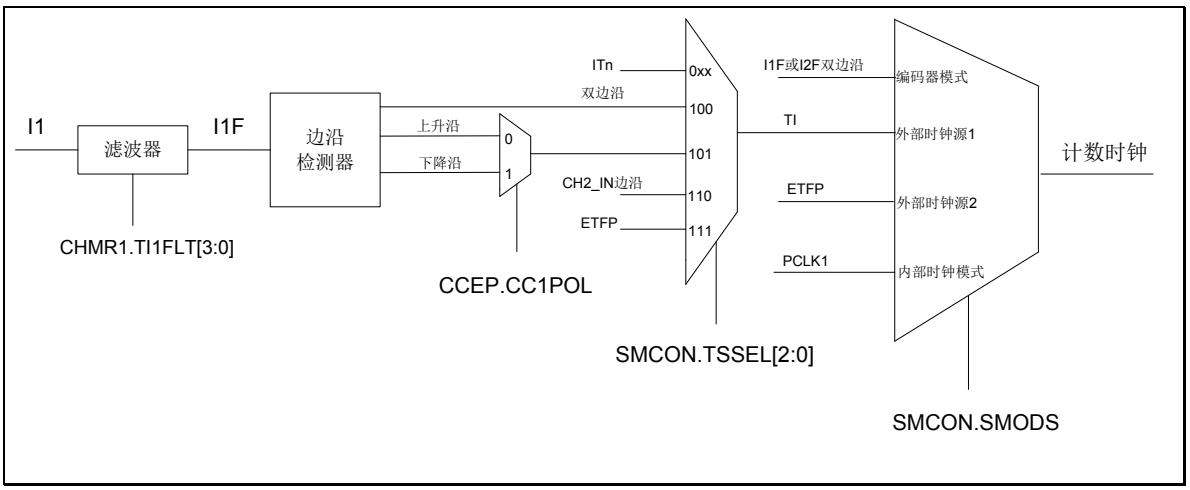


图 20-5 I1 外部时钟连接

配置计数器为外部时钟源 1，步骤如下：

1. AD16C4Tn_SMCON 寄存器中 SMODS = "111"，配置定时器外部时钟模式 1。
2. 设置 AD16C4Tn_SMCON 寄存器中的 TSSEL 选择外部时钟源。
3. 如外部时钟源为 I1，可配置 AD16C4Tn_CHMR1 寄存器 CC1SSEL = "01"，配置通道 1 检测 I1 输入的上升沿；设置 AD16C4Tn_CCEP 寄存器中 CC1POL = '0'，选择极性为上升沿。
4. 写 AD16C4Tn_CHMR1 寄存器的 I1FLT[3：0]位，配置输入滤波器时间（若没有滤波器需求，维持 I1FLT = "0000"）。
5. AD16C4Tn_CON1 寄存器中 CNTEN = '1'，使能计数器。

当 I1 上出现一次上升沿时，计数器计数一次且 TRGIF 标志位置位。

20. 4. 3. 3 外部时钟源 2

置位 AD16C4Tn_SMCON 寄存器的 ECM2EN 位选定外部时钟源 2。

计数器可对外部触发输入 ET 进行上升沿或下降沿计数。

下图给出了外部输入模块的概况。

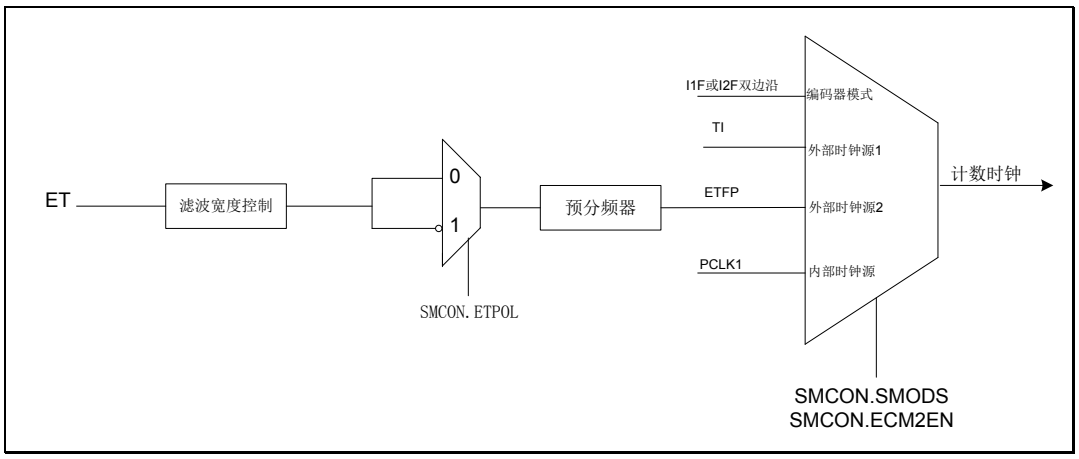


图 20-6 外部触发输入模块

配置计数器为外部时钟源 2，配置过程如下：

1. 设置 AD16C4Tn_SMCON 寄存器的 ETFLT[3：0]，配置输入滤波时间。
2. 设置 AD16C4Tn_SMCON 寄存器中 ETPSEL[1：0]，设置预分频器。
3. 设置 AD16C4Tn_SMCON 寄存器中 ETPOL，检测 ET 引脚上升沿或下降沿。
4. 设置 AD16C4Tn_SMCON 寄存器中 ECM2EN = '1'，使能外部时钟模式 2。
5. 设置 AD16C4Tn_CON1 寄存器的 CNTEN = '1'，使能计数器。

计数器每两个上升沿计一次数。

20. 4. 3. 4 内部触发输入（ITn）

当 AD16C4Tn_SMCON 寄存器的 SMODS= "111"，选定内部触发模式。计数器根据选定的内部输入端的上升或下降沿计数。

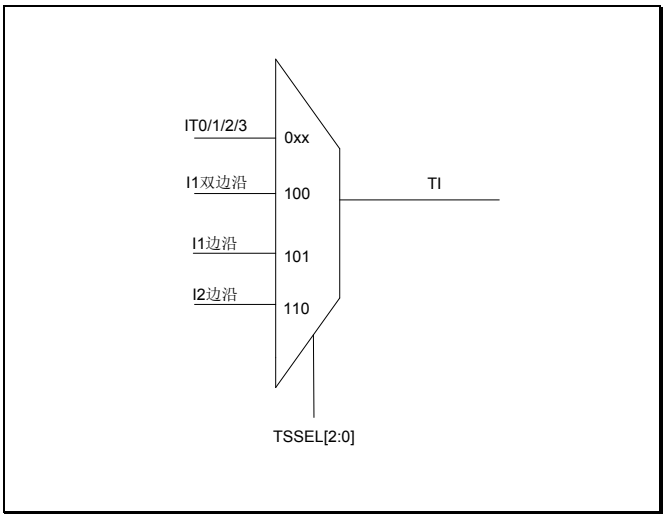


图 20-7 ITn 外部时钟连接

配置计数器在 ITn 输入端的上升沿递增计数，步骤如下：

1. AD16C4Tn_SMCON 寄存器中 SMODS = "111"，配置外部时钟模式 1。
2. AD16C4Tn_SMCON 寄存器的 TSSEL = "0xx"，选定 ITn 作为触发输入源。
3. AD16C4Tn_CON1 寄存器的 CNTEN = '1'，使能计数器。

ITn 产生上升沿时，计数器计数一次。ITn 上升沿与实际时钟间的延时，取决于 ITn 输入的再同步电路，一般为 2~3 个定时器模块时钟周期。

20. 4. 4 计数模式

20. 4. 4. 1 递增计数模式

在递增计数模式下，当 AD16C4Tn_REPAR 寄存器值为 0 时，计数器从 0 开始递增，直至 AD16C4Tn_AR 寄存器值；然后从 0 重新开始计数并产生一个更新事件（UEV）。当 AD16C4Tn_REPAR 寄存器不为 0 时，则在 AD16C4Tn_REPAR+1 次计数后产生更新事件。

当有更新事件（UEV）产生时，预装载寄存器会更新到影子寄存器，更新标志位（AD16C4Tn_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）：

- ◇ 更新 AD16C4Tn_REPAR 寄存器的值到影子寄存器
- ◇ 更新 AD16C4Tn_AR 寄存器的值到影子寄存器
- ◇ 更新 AD16C4Tn_PRES 寄存器的值到影子寄存器

下图为 AD16C4Tn_REPAR=0x0，AD16C4Tn_AR = 0x16，预分频设为 2 分频时的计数器时序。

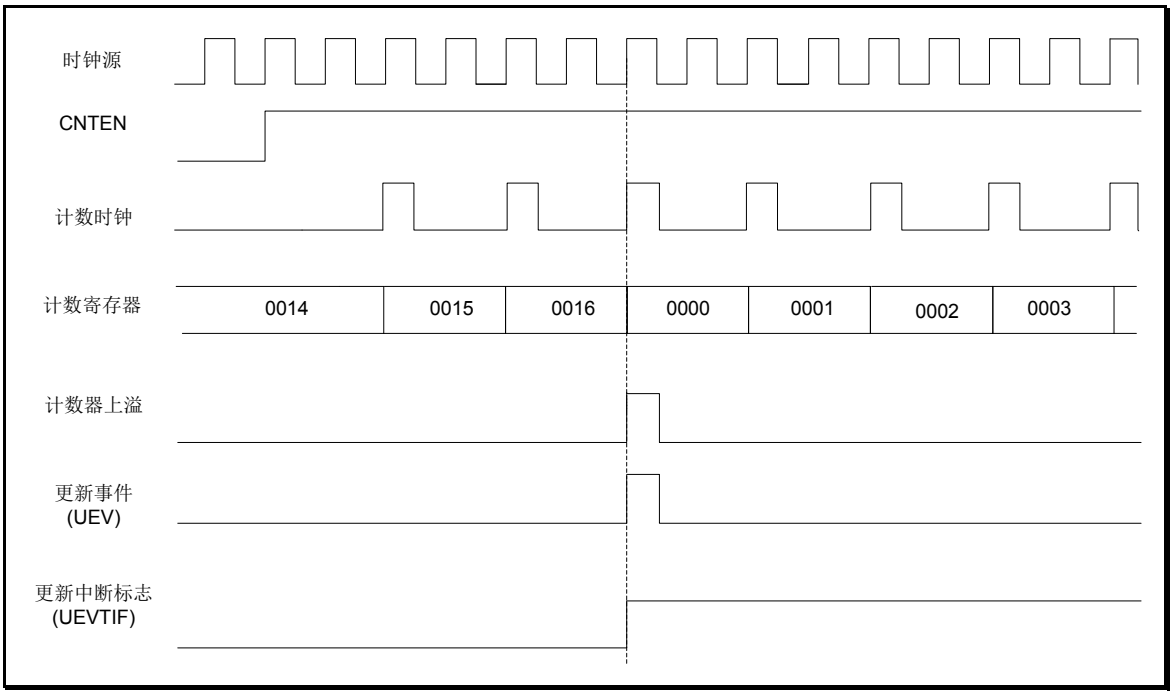


图 20-8 计数器递增计数时序图

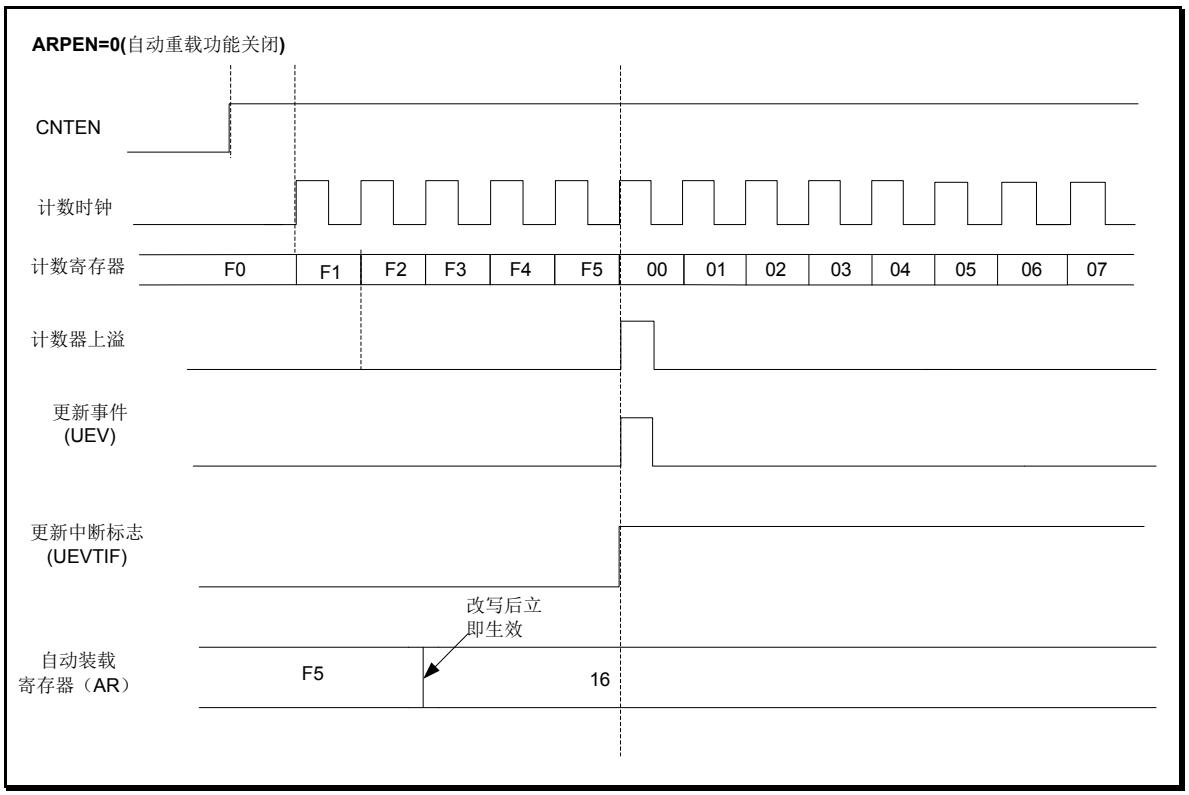


图 20-9　当 ARPEN=0 时计数器时序图

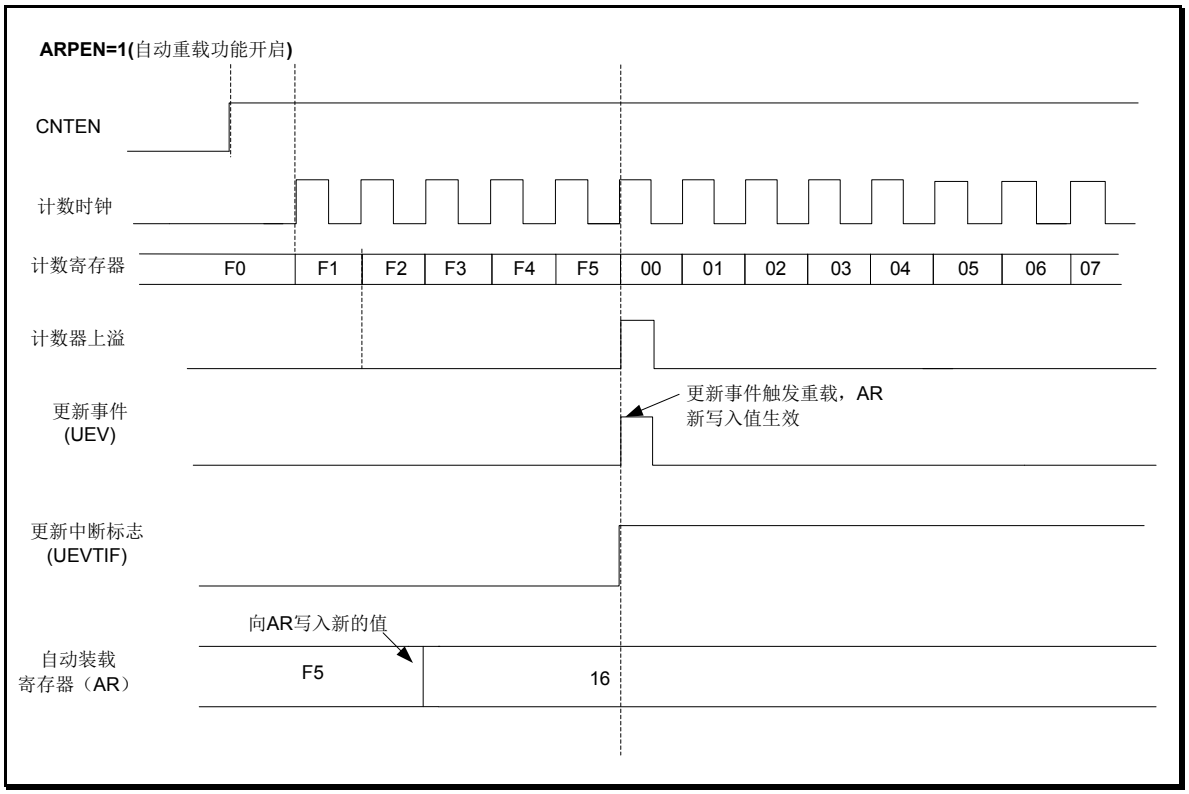


图 20-10　当 ARPEN=1 时计数器时序图

20. 4. 4. 2　递减计数模式

在递减计数模式下，当 AD16C4Tn_REPAR 寄存器值为 0 时，计数器从 AD16C4Tn_AR

寄存器值开始递减至 0；然后重复递减并产生更新事件（UEV）。当 AD16C4Tn_REPAR 寄存器不为 0 时，则在 AD16C4Tn_REPAR+1 次后产生更新事件。

置位 AD16C4Tn_SGE 寄存器中的 SGU 位（通过软件或使用从机模式控制器）同样会产生更新事件。

当有更新事件（UEV）产生时，预载寄存器值会更新到影子寄存器，更新标志位（AD16C4Tn_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）。

下图为 AD16C4Tn_REPAR=0x0，AD16C4Tn_AR = 0x27，预分频设为 1 分频时的计数器时序。

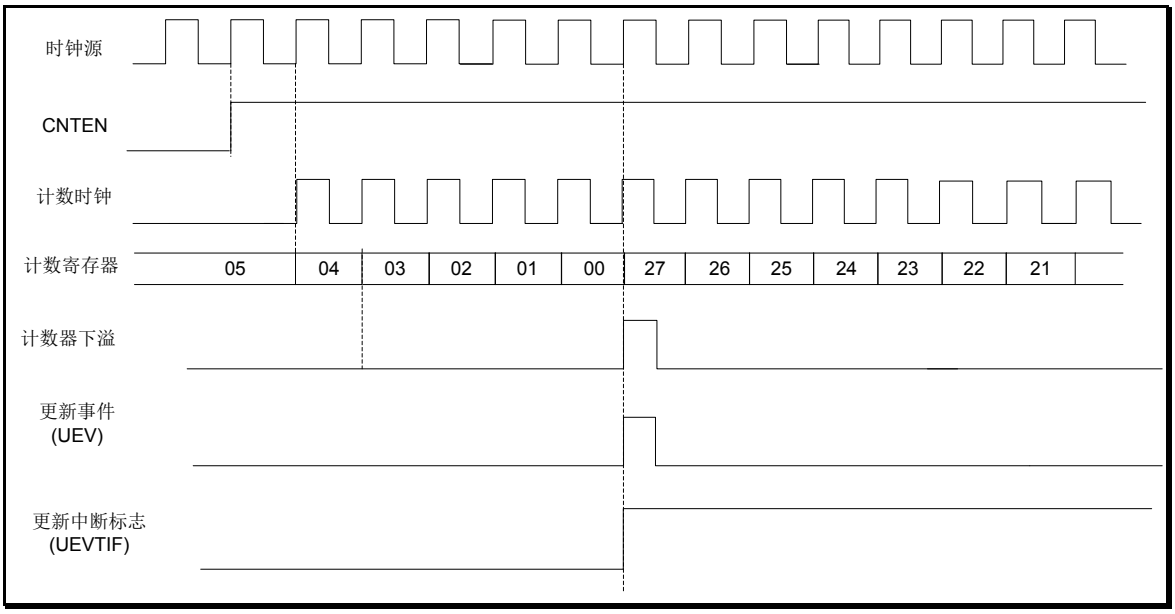


图 20-11　定时器递减计数时序图

20. 4. 4. 3　中心对齐模式

当 AD16C4Tn_CON1 寄存器的 CMSEL 位的值不等于"00"时，定时器工作在中心对齐模式。定时器配置为中心对齐模式时，计数器先从 0 开始递增至 AD16C4Tn_AR 寄存器值-1，并产生更新事件（UEV）；接着计数器从 AD16C4Tn_AR 寄存器值递减至 1，并产生下溢事件。如此循环计数。在计数器递减计数（中心对称模式 1，CMSEL="01"）、计数器递增计数（中心对称模式 2，CMSEL="10"）、计数器递增和递减计数（中心对称模式 3，CMSEL="11"）模式下，当通道配置为输出模式时，其输出比较中断标志位会置位。

在中心对齐模式下，AD16C4Tn_CON1 寄存器的 DIRSEL 位无法进行写操作，该位由硬件自动更新指示当前计数方向。

计数上溢、下溢或者置位 AD16C4Tn_SGE 寄存器的 SGU 位（通过软件或使用从模式控制器）都会产生更新事件。因此，计数器和预分频器都会从 0 开始计数。

软件置位 AD16C4Tn_CON1 寄存器中的 DISUE 位可关闭更新事件（UEV）的产生。更新事件（UEV）关闭时，可避免向预载寄存器写新值时更新影子寄存器。DISUE 复位之前都不会产生更新事件。而在正常产生更新事件时，计数器仍然从 0 开始，同样预分频计数也是从 0 开始(但预分频值没有改变)。此外，若置位 AD16C4Tn_CON1 寄存器中的 UERSEL

位（更新请求选择），置位 SGU 位时会产生一次更新事件（UEV），但 UEVTIF 标志位不会置位（因此，不会触发中断或 DMA 请求）。这就避免了在捕获事件时，清除计数器值时产生更新和捕获中断。

当有更新事件（UEV）产生时，预载寄存器值会更新到影子寄存器，更新标志位（AD16C4Tn_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）。

注：若更新源为计数器上溢，自动重载会在计数器重载前更新。因此，下一周期即为预期值（计数器载入新值）。

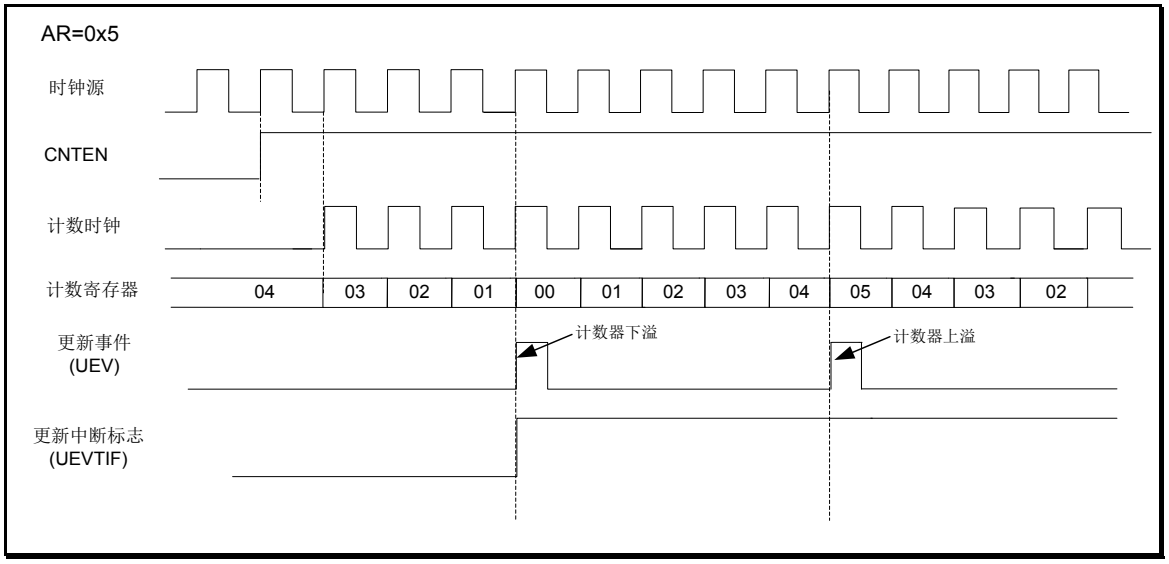


图 20-12　增减计数器时序图

20. 4. 5 捕获/比较通道

下方 3 张图为捕获/比较通道的概述。

输入电路对 In 输入端的信号进行采样，产生一个经过滤波的信号 InF。之后，一个可极性选择的边沿检测器产生 In 边沿检出信号，该信号可作为从模式控制器的触发输入或作为捕获控制命令，且信号经过分频后进入捕获寄存器。

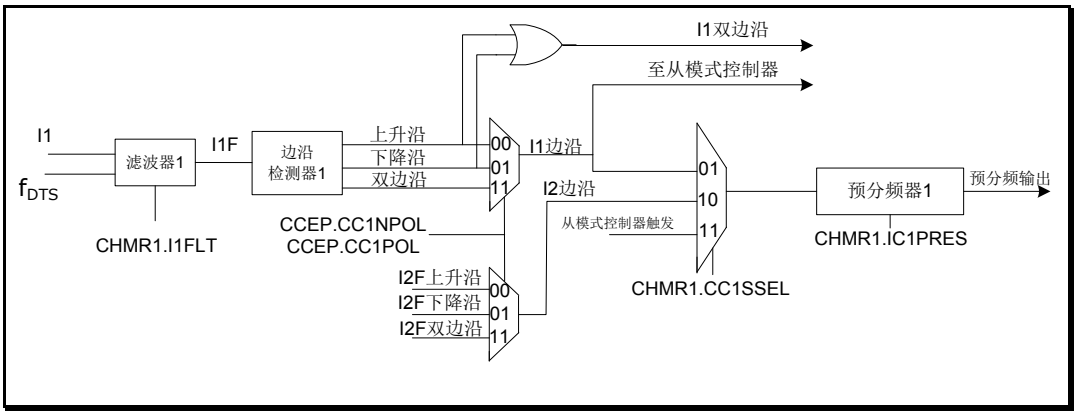


图 20-13 捕获/比较通道

输出部分产生一个中间波形（高有效）作为基准，在输出末端决定最终输出信号的极性。

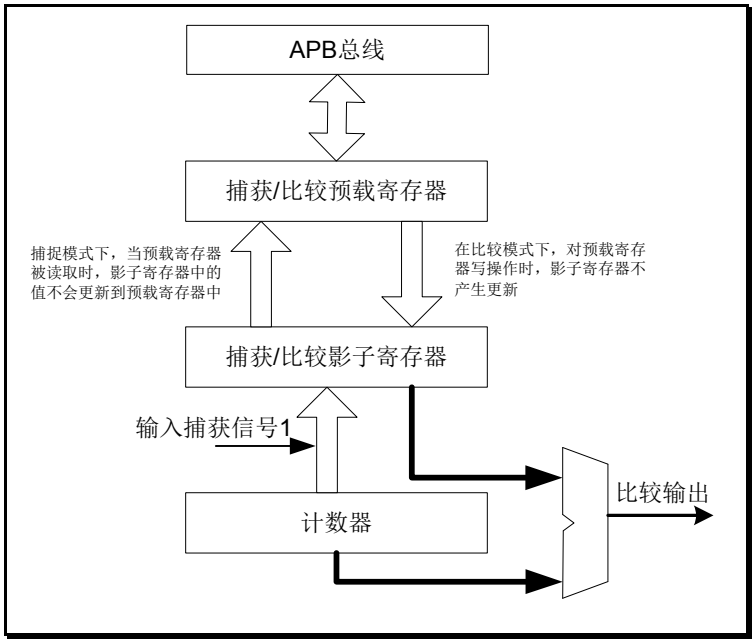


图 20-14 捕获/比较通道 1 结构图

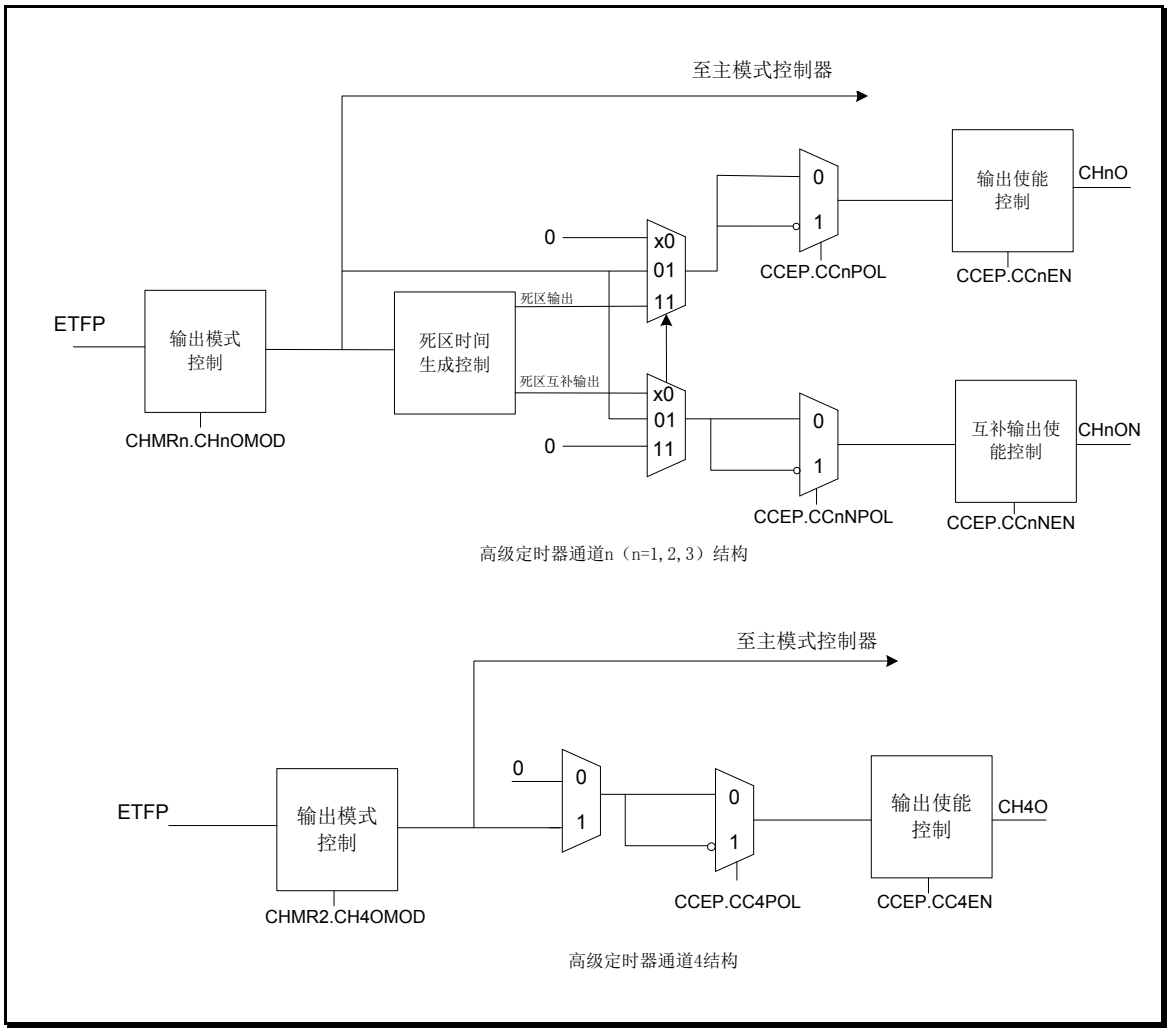


图 20-15　捕获/比较信道的输出部分

20.4.6 输入捕获模式

在输入捕获模式下，当检测到 In 上相应信号变化时，计数器的值就会被锁存到捕获/比较寄存器（AD16C4Tn_CCVALn）中。当捕获发生时，相应的 CHnCCIF 标志位（AD16C4Tn_RIF）会置位，同时会触发中断或 DMA（如果使能）请求。若发生捕获时，CHnCCIF 标志位已经置位，则过捕获 CHnOVIF 标志位（AD16C4Tn_RIF）置位。软件写'0'或读取 AD16C4Tn_CCVALn 寄存器中的捕获值都可以复位 CHnCCIF 标志位。对 CHnOVIF 位写'0'可清空该标志位。

以下为以 I1 输入上升沿作为捕获输入时的流程：

1. 选择有效输入端：AD16C4Tn_CCVAL1 必须连接到 I1 输入端，因此需将 AD16C4Tn_CHMR1 寄存器中的 CC1SSEL 位写"01"。只要 CC1SSEL 不为"00"，通道被配置为输入且 AD16C4Tn_CCVAL1 寄存器为只读。
2. 根据定时器连接的输入信号，配置输入滤波器的持续时间。当输入信号翻转时，前 5 个内部时钟信号是不稳定的，因此必须配置滤波器的时间大于 5 个时钟周期。当 I1 检测到新的电平，连续 8 次采样可确认电平变化有效。
3. 选择 I1 信道的有效边沿变换。AD16C4Tn_CCEP 寄存器中的 CC1POL 写'0'（上升沿）。
4. 配置输入预分频器。
5. 置位 AD16C4Tn_CCEP 寄存器中的 CC1EN 位，使能捕获计数器的值到捕获寄存器。
6. 如有需要，置位 AD16C4Tn_IER 寄存器中的 CC1IT 位，使能中断请求。置位 AD16C4Tn_DMAEN 寄存器中的 CC1DMA 位，使能 DMA 请求。

当发生输入捕获时：

1. 有效边沿产生，AD16C4Tn_CCVAL1 寄存器获取计数器的值。
2. CH1CCIF 标志位置位（中断标志）。若至少 2 个连续的捕获发生，但标志位没有及时清除，则 CH1OVIF 也会置位。
3. 中断的产生取决于 AD16C4Tn_IVS 寄存器中的 CC1IT 位。
4. DMA 请求的产生取决于 CC1DMA。

为了处理捕获溢出，建议在读出捕获溢出标志位之前先读取捕获数据。这可以避免丢失在读出捕获标志位之后与读取数据之前可能重复产生的捕获信息。

注：捕获中断请求可由软件设置 AD16C4Tn_SGE 寄存器中 SGCCnE 位产生。

20. 4. 6. 1 PWM输入模式

测量 I1 上 PWM 信号的周期和占空比的过程如下：

1. 为 AD16C4Tn_CCVAL1 选择有效的输入：AD16C4Tn_CHMR1 寄存器中的 CC1SSEL 位写"01"（I1 被选择）。

2. 为 I1 边沿检出选择有效的极性（用于捕获数据到 AD16C4Tn_CCVAL1 寄存器和计数器清零）：CC1POL 位写'0'（上升沿有效）。

3. 为 AD16C4Tn_CCVAL2 选择有效输入：AD16C4Tn_CHMR1 寄存器的 CC2SSEL 位写"10"（I1 被选择）。

4. 为 I1 边沿检出选择有效极性（用于捕获数据到 AD16C4Tn_CCVAL2）：CC2POL 位写'1'（下降沿有效）。

5. 选择有效的触发输入：AD16C4Tn_SMCON 寄存器的 TSSEL 位写"101"（I1 边沿检出被选择）。

6. 配置从机模式控制器为复位模式：AD16C4Tn_SMCON 寄存器的 SMODS 位写"100"。

7. 使能捕获：AD16C4Tn_CCEP 寄存器的 CC1EN 位和 CC2EN 位写'1'。

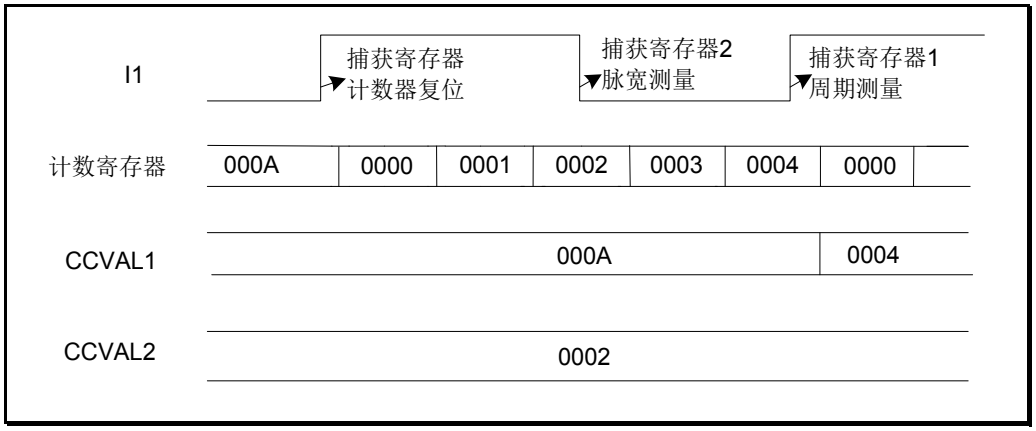


图 20-16 PWM 输入模式时序

20.4.7 PWM模式

脉宽调制模式可以产生一个 AD16C4Tn_AR 寄存器值确定频率，AD16C4Tn_CCVALn 寄存器值确定占空比的信号。

每个通道的 PWM 模式是相互独立的（每个 CHnO 输出一个 PWM），AD16C4Tn_CHMRn 寄存器的 CHnOMOD 位写"110"（PWM 模式 1）或写"111"（PWM 模式 2）。必须通过置位 AD16C4Tn_CHMRn 寄存器的 CHnOPREN 位来使能相应的预载寄存器，最后还需置位 AD16C4Tn_CON1 寄存器的 ARPEN 位来使能自动重装预载功能。

只有当更新事件发生时预载寄存器中的值才会传到影子寄存器，因此，在使能计数前，必须通过置位 AD16C4Tn_SGE 寄存器的 SGU 位来初始化所有的寄存器。

CHnO 的极性可通过 AD16C4Tn_CCEP 寄存器的 CCnPOL 位配置，有效极性可配置为高或低。CHnO 的输出使能由 CCnEN、CCnNEN、GOEN、OFFSSI 和 OFFSSR 位（AD16C4Tn_CCEP 和 AD16C4Tn_BDCFG 寄存器）组合控制。

在 PWM 模式（1 或 2）中，AD16C4Tn_COUNT 和 AD16C4Tn_CCVALn 寄存器的值会持续的比较，确定 AD16C4Tn_CCVALn<= AD16C4Tn_COUNT 或 AD16C4Tn_CCVALn >= AD16C4Tn_COUNT（取决于计数器的计数方向）。

定时器产生 PWM 波形是边沿对齐或中心对齐，取决于 AD16C4Tn_CON1 寄存器的 CMSEL 位。

20.4.7.1 PWM边沿对齐模式

1. 递增计数配置

当 AD16C4Tn_CON1 寄存器的 DIRSEL 位为低时，计数器递增计数。

下图给出了 AD16C4Tn_AR = 8 时的边沿对齐 PWM 波形。

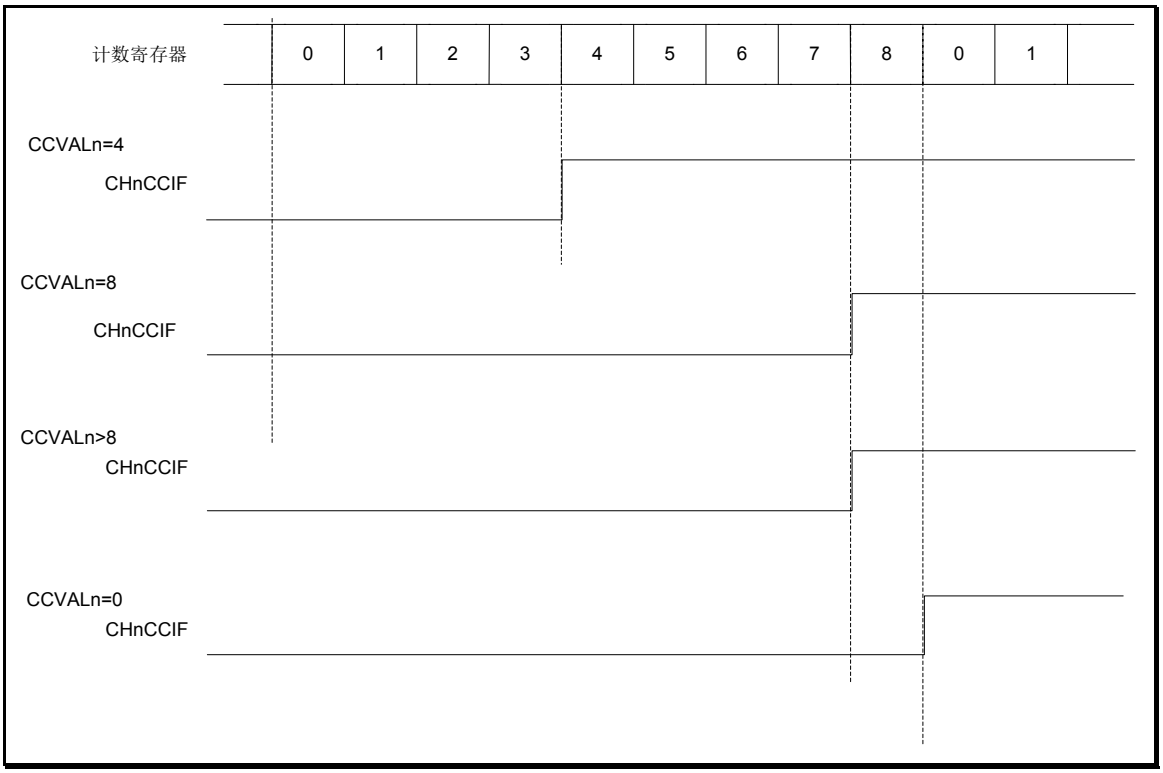


图 20-17　边沿对齐 PWM 波形（AR=8）

2.　递减计数配置

当 AD16C4Tn_CON1 寄存器的 DIRSEL 位为高时，计数器递减计数。

20. 4. 7. 2　PWM中心对齐模式

当 AD16C4Tn_CON1 寄存器中的 CMSEL 位不为"00"时，中心对齐模式有效。计数器是递增、递减计数分别置比较标志位或递增递减都置比较标志位，取决于 CMSEL 位的配置。AD16C4Tn_CON1 寄存器的方向位（DIRSEL）是由硬件更新的，软件无法修改。

下图为中心对齐方式产生的 PWM 波形的例子：

◇　AD16C4Tn_AR=0x3F，AD16C4Tn_CCVALn=0x3D

◇　PWM 模式 1

- AD16C4Tn_CON1 寄存器的 CMSEL= "01"，在中心对齐模式 1 下，计数器向下计数时会置位比较标志位。

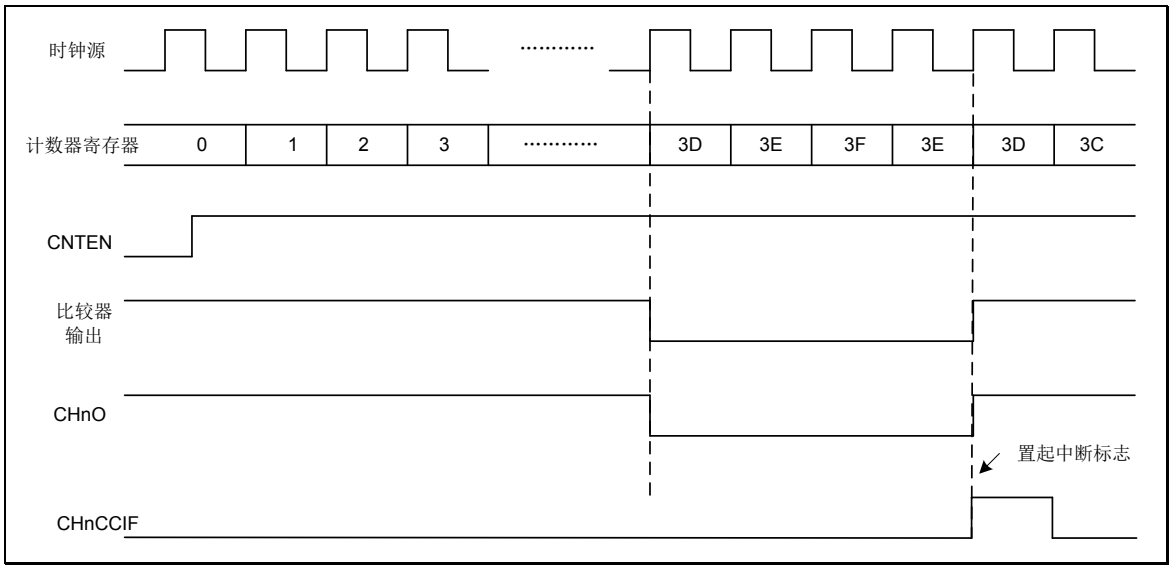


图 20-18 中心对齐 PWM 波形（AR=0x3F）

中心对齐模式的使用技巧：

◇ 当进入中心对齐模式后，当前递增或递减配置生效。计数器递增或递减计数取决于 AD16C4Tn_CON1 寄存器的 DIRSEL 位的值。

◇ 计数器在中心对齐模式下运行时，对计数器写操作可能导致不可预知的结果。特别是：

- 若向计数器写入的值大于自动重载值（AD16C4Tn_COUNT>AD16C4Tn_AR），计数方向不更新。例如，如果计数器递增计数，写入值后仍旧递增计数。
- 若向计数器写 0 或 AD16C4Tn_AR 中的重载值，则计数方向更新，但并没有产生 UEV。

◇ 使用中心对齐模式最安全的方式是计数器开始计数前通过软件产生更新事件（置位 AD16C4Tn_SGE 寄存器中的 SGU 位）且在计数器运行过程中不对计数器写值。

20. 4. 8 输出比较模式

该功能用于控制输出波形或指示周期时间的结束。

当捕获/比较寄存器和计数器值匹配时，输出比较功能：

- ◇ 输出比较模式（AD16C4Tn_CHMRn 寄存器中的 CHnOMOD 位）和输出极性（AD16C4Tn_CCEP 寄存器中的 CCnPOL 位）的配置值输出到对应的引脚上。
- ◇ 中断状态寄存器中的标志位置位（AD16C4Tn_RIF 寄存器的 CHnCCIF 位）。
- ◇ 若相应的中断掩码置位，则产生中断（AD16C4Tn_IER 寄存器的 CCnIT 位）。
- ◇ 若相应的使能位置位（AD16C4Tn_DMAEN 寄存器的 CCnDMA 位，AD16C4Tn_CON2 寄存器的 CCDMASEL 位用于 DMA 请求的选择），则发送 DMA 请求。

AD16C4Tn_CHMRn 寄存器中 CHnOPREN 位的值可决定 AD16C4Tn_CCVALn 寄存器是否带有预装载寄存器。

在输出比较模式中，更新事件 UEV 对 CHnO 的输出没有影响。计时分辨率为计数器的一次计数。输出比较模式同样可以用来输出单个脉冲（单脉冲模式）。

输出比较的配置过程：

1. 选定计数器时钟（内部，外部，预分频）。
2. AD16C4Tn_AR 与 AD16C4Tn_CCVALn 寄存器中写入预期值。
3. 若需要产生中断请求，置位 AD16C4Tn_IER 寄存器中的 CCnIT 位。
4. 选择输出模式，例如：

   - CHnOMOD = "011"，当 CNTV 与 CCRVn 匹配时，CHnO 输出翻转。
   - CHnOPREN = '0'，关闭预载寄存器。
   - CCnPOL = '0'，选择有效极性为高。
   - CCnEN = '1'，使能输出。
5. AD16C4Tn_CON1 寄存器中的 CNTEN 位置位，使能计数器。

通过配置 AD16C4Tn_CHMR1 寄存器的 CHnOPREN 位可将 AD16C4Tn_CCVALn 配置为是否带预装载寄存器。通过软件方式，AD16C4Tn_CCVALn 寄存器的值可随时更新控制输出波形。

20. 4. 8. 1 外部事件清除比较输出

ETFP 输入端（AD16C4Tn_CHMRn 寄存器的 CHnOCLREN 位写'1'）上的高电平，可将给定通道的比较输出信号拉低。在下次更新事件（UEV）发生前，比较输出会一直保持为低。该功能只能应用在输出比较和 PWM 模式中，强制输出模式中不起作用。

ET 信号可以接到电流控制比较器的输出端。该例中，ET 须按如下流程配置：

1. 外部触发预分频器应该关闭：AD16C4Tn_SMCON 寄存器的 ETPSEL[1：0]位应该写"00"

2. 外部时钟源 2 关闭：AD16C4Tn_SMCON 寄存器的 ECM2EN 位写'0'

3. 外部触发极性（ETPOL）和外部触发滤波器（ETFLT）可根据用户需要配置

20. 4. 8. 2 强制输出模式

在输出模式中（AD16C4Tn_CHMRn 寄存器中 CCnSSEL = "00"），软件可强制将每个输出比较信号（CHnO/CHnON）改为有效或无效状态，这种修改独立于输出比较寄存器和计数器的比较结果。

为了将某输出比较信号（CHnO）强制为有效状态，需将相应的 AD16C4Tn_CHMRn 寄存器中 CHnOMOD 位写"101"。因此，比较输出被强制为高（高时为有效状态）且 CHnO 的值为 CCnPOL 极性位的相反值。

例如：CCnPOL= '0'（CHnO 高电平有效），则 CHnO 被强制为高电平。

对 AD16C4Tn_CHMRn 寄存器的 CHnOMOD 位写"100"，比较输出可被置低。

无论怎样，AD16C4Tn_CCVALn 影子寄存器和计数器之间的比较仍然进行，相应的标志位仍可置位。

20.4.9 单脉冲模式

单脉冲模式下，响应某个触发后，定时器的输出通道在可配置的延迟时间后产生一个脉冲，脉冲长度可配。从模式控制器可控制计数器的启动。脉冲波形可在输出比较模式和 PWM 模式下产生。置位 AD16C4Tn_CON1 寄存器的 SPMEN 位可选择单脉冲模式。计数器会在下次更新事件 UEV 产生时自动停止。

只有比较值不同于计数器初始值时，单脉冲才可以正确的产生。计数器开始计数前（定时器等待触发），必须如下配置：

- ◇ 递增计数：CNT < CCVALn≤ AR（特别地，0 < CCVALn）
- ◇ 递减计数：CNT > CCVALn

基于 PWM 模式设置单脉冲输出波形的步骤如下：

- ◇ 设置 AD16C4Tn_CHMRn 寄存器的 CHnOMOD 位，选择 PWM 模式 1 或 2；
- ◇ 设置 AD16C4Tn_CCEP 寄存器的 CCnPOL 位，选择通道端口 CHnO 的输出极性；
- ◇ 设置 AD16C4Tn_CON1 寄存器的 DIRSEL，CMSEL，SPMEN 位，配置为递增或递减计数，PWM 普通波形模式，单脉冲模式使能；
- ◇ 设置 AD16C4Tn_CHMR1 寄存器的 CH1OPREN =1，AD16C4Tn_CON1 寄存器的 ARPEN =1，使能比较寄存器和计数重载寄存器的缓冲功能（也可以根据实际情况不使能缓冲）；
- ◇ 设置 AD16C4Tn_CCVALn 寄存器和 AD16C4Tn_AR 寄存器，配置单脉冲输出延时和脉宽时间；
- ◇ 设置 AD16C4Tn_SGE 寄存器的 SGU=1 来产生一个更新事件；
- ◇ 设置 AD16C4Tn_CON1 寄存器的 CNTEN=1 来启动计数器，也可以在触发模式下，通过外部触发输入信号来触发硬件自动设置 CNTEN=1。

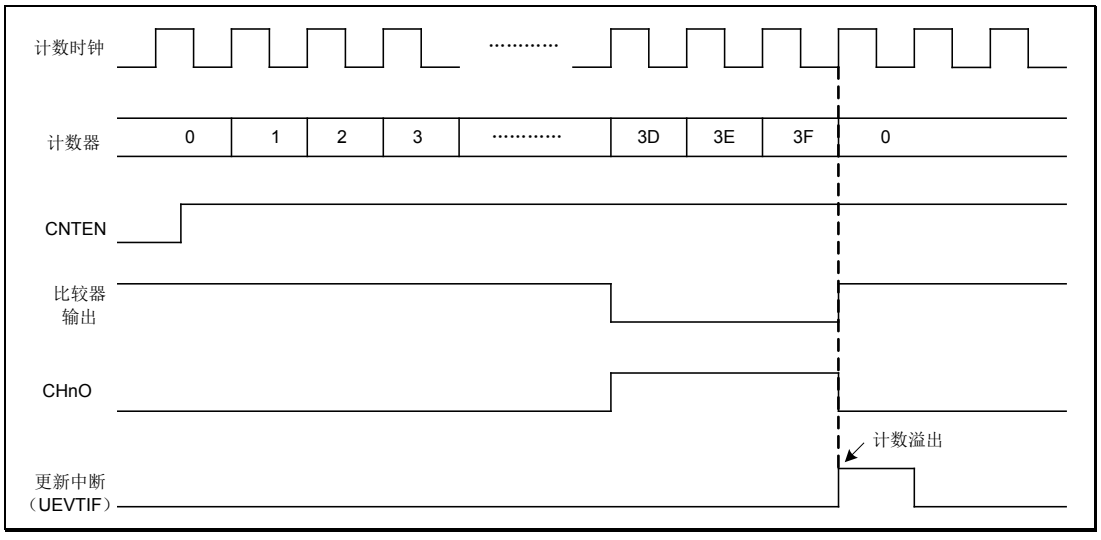


图 20-19 单脉冲模式

20. 4. 10 互补输出与死区时间

两个互补的通道输出信号，可以用来控制输出的瞬时开关。死区时间可配置。

每个输出可独立选择输出极性（主输出 CHnO 或互补输出 CHnON），该操作可通过写 AD16C4Tn_CCEP 寄存器的 CCnPOL 和 CCnNPOL 位完成。

互补信号 CHnO 和 CHnON 由几个控制位共同控制，分别是 AD16C4Tn_CCEP 寄存器中的 CCnEN 和 CCnNEN 位，AD16C4Tn_BDCFG 和 AD16C4Tn_CON2 寄存器中的 GOEN、OISSn、OISSnN、OFFSSI 及 OFFSSR 位。特别是死区时间使能后的空闲状态的切换（GOEN 变为 0）。

置位 CCnEN 和 CCnNEN 位，使能死区时间插入，若有刹车电路，同样需要置位 GOEN 位。AD16C4Tn_BDCFG 寄存器的 DT[7：0]可以控制所有通道的死区时间的产生。根据比较输出波形，产生 CHnO 和 CHnON 两路输出。若 CHnO 和 CHnON 有效电平为高：

- ◇ CHnO 的输出信号与参考信号一致。上升沿除外，相对参考信号的上升沿，CHnO 输出会有延迟。
- ◇ CHnON 的输出信号与参考信号相反。上升沿除外，相对参考信号的下降沿，CHnON 输出会有延迟。

若延迟时间大于有效输出的宽度（CHnO 或 CHnON），则相应的脉冲不会产生。

下图给出了死区时间输出信号和比较输出波形之间的关系。假设 CCnPOL = 0，CCnNPOL = 0，GOEN = 1，CCnEN = 1，和 CCnNEN = 1

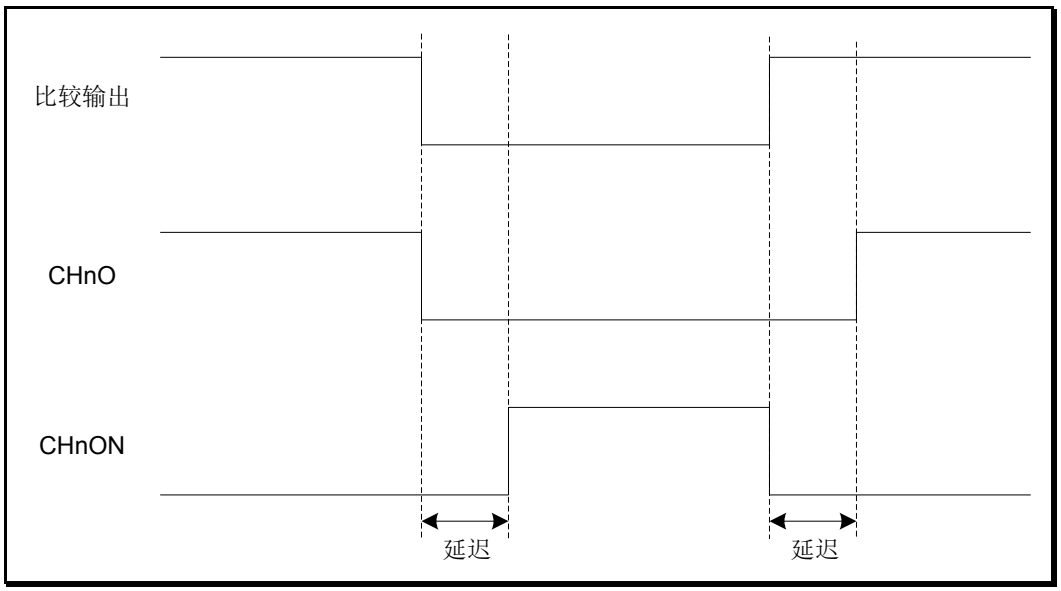


图 20-20　互补输出含死区时间插入

当 PWM 通道配置为互补输出时，如下寄存器控制位都会有缓冲：CHnOMOD、CCnEN 和 CCnNEN。发生互补通道更新事件时，这些寄存器位才会真正生效，这样就可以预先设置好下一步的配置，并同时对所有互补通道的配置进行更新。互补通道更新事件可以通过设置 AD16C4Tn_SGE 寄存器的 SGCOM=1 产生，或由触发信号产生（由 AD16C4Tn_SMCON 寄存器的 TSSEL 位选择触发信号）。

20. 4. 11 刹车功能

刹车功能模式由以下几个控制位进行设置：AD16C4Tn_BDCFG 寄存器中的 GOEN、OFFSSI 和 OFFSSR 位，AD16C4Tn_CON2 寄存器中的 OISSn 和 OISSnN 位，输出使能信号和无效电平都会被修改。

刹车源可以是刹车输入引脚、系统安全事件（包括时钟失败事件，低电压检出事件和 CPU 锁死事件，可在寄存器 SYSCFG_TBKCFG 进行配置使能）以及软件控制 AD16C4Tn_SGE 寄存器的 SGBRK 位。时钟失败事件由时钟管理单元（CMU）中时钟安全机制产生。时钟安全机制详细信息可参考时钟管理单元（CMU）章节。

系统复位后，刹车电路被禁止且 GOEN 位被复位。置位 AD16C4Tn_BDCFG 寄存器的 BRKEN 位可使能刹车功能，同样寄存器中，BRKEN 位可选择刹车输入信号的极性。BRKEN 和 BRKP 位可同时修改。对 BRKEN 和 BRKP 位写操作后，1 个 APB 时钟周期延时后写入值才会生效。因此，写操作后，需等待 1 个 APB 时钟周期后才能正确读回写入值。

由于 GOEN 的下降沿可以是异步的，在实际信号（作用在输出端）和同步控制位（AD16C4Tn_BDCFG 寄存器中）之间插入了一个同步电路。这也导致了异步和同步信号之间会产生一些延迟。特别是 GOEN 之前为低时对 GOEN 写 1 操作后，要读取正确值，必须先插入一个延时（空指令）。这是因为写入的是异步信号，而读取的是同步信号。

当发生刹车请求时（刹车输入端有刹车电平）：

- ◇ GOEN 位被异步清除，输出端进入无效状态，空闲状态或复位状态（OFFSSI 位选择）。即使 MCU 的振荡器关闭，该功能仍然有效。
- ◇ 一旦 GOEN=0，每个通道输出预先配置的电平。AD16C4Tn_CON2 寄存器中的 OISSn 位配置该电平。如果 OFFSSI=0，则定时器释放使能输出，否则使能输出一直为高。
- ◇ 当使用互补输出时：
  - 如果定时器时钟仍然存在，则死区时间生成器会重新生效，这样在死区时间后，OISSn 和 OISSnN 位的配置电平可驱动输出。这种情况下，CHnO 和 CHnON 无法驱动输出端都为有效电平。
  - 输出首先被置于复位状态即无效的状态（取决于极性）。这是异步操作，即使定时器没有时钟时，此功能也有效。注，由于对 GOEN 的重新同步，死区时间的周期会比通常情况下长一些（大约 2 个 TIMER 模块时钟周期）。
  - 如果 OFFSSI = 0，则定时器释放使能输出，否则使能输出保持或变高（一旦 CCnEN 和 CCnNEN 有一个变高时）。
- ◇ 当刹车状态标志位（AD16C4Tn_RIF 寄存器中的 BRKIF 位）置位时，若 AD16C4Tn_IER 寄存器中的 BRKIT 位置位，可触发中断。
- ◇ 当 AD16C4Tn_BDCFG 寄存器中的 AOEN 位置位时，在下次更新事件（UEV）发生时，GOEN 位会自动置位。例如，该功能可用来整形。否则，GOEN 位会保持为低，直到对其写'1'操作，该特性可用于安全方面的应用，可以将刹车输入端接到一个电源驱动的报警端、热敏传感器或其他安全器件上。

注：刹车输入为电平有效。所以，当刹车输入有效时，不能置位（自动地或者通过软件）GOEN。同时，状态标志 BRKIF 不能被清除。

除刹车输入和输出管理，为保证应用程序的安全，内部刹车电路具有写保护功能。用户可冻结几个配置参数（死区时间，CHnO/CHnON 极性和失能时状态，CHnOMOD 配置，刹车使能和极性）。通过 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位，可从三个保护等级中选择一种保护等级。MCU 复位后，LOCKLVL 位只能写一次。

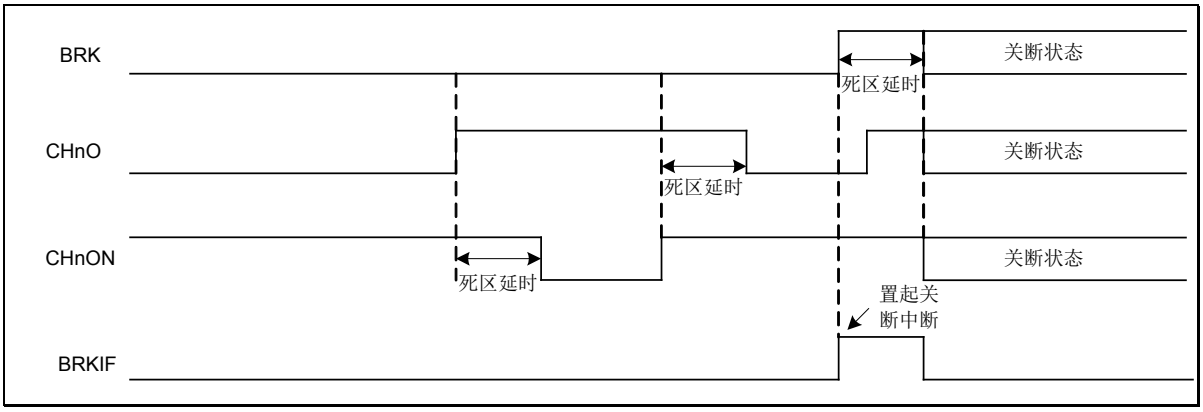


图 20-21　刹车输出行为

20. 4. 12 编码器接口模式

编码器接口模式的三种配置：若计数器只根据 I2 上的边沿计数，则 AD16C4Tn_SMCON 寄存器中的 SMODS = "001"；若计数器只根据 I1 上的边沿计数，则 AD16C4Tn_SMCON 寄存器中的 SMODS = "010"；若计数器同时根据 I1 和 I2 上的边沿计数，则 AD16C4Tn_SMCON 寄存器中的 SMODS = "011"。

配置 AD16C4Tn_CCEP 寄存器中的 CC1POL 和 CC2POL 位的值可选择 I1 和 I2 的极性。如果需要，也可以配置输入滤波器。

CH1_IN 和 CH2_IN 端口作为增量编码器的接口。当计数器使能时，计数器根据 I1 或 I2 上滤波后的有效电平变化时钟计数。I1 和 I2 滤波后的有效信号顺序会产生计数脉冲及方向信号。计数器是递增或递减计数由信号的跳变顺序决定，AD16C4Tn_CON1 寄存器中的 DIRSEL 计数方向位由自动硬件更新。

编码器接口模式的工作方式类似于一个带有方向选择的外部时钟。计数器在 0 到 AD16C4Tn_AR 寄存器中的自动重载值之间连续计数。因此，必须在开始计数前配置 AD16C4Tn_AR 寄存器。同样的，捕获器、预分频器、重复计数器、触发输出的特性正常工作。设定编码模式和选择外部时钟源 2 不兼容，不可以同时选择。

该模式下，计数器会根据增量式编码器的速度和方向自动修改，计数器的值反应的是编码器的位置。计数方向对应着连接传感器的旋转方向。

下表列出了所有的可能组合，假设 I1 和 I2 不同时变换。

有效边沿	有效边沿相对信号的电平（I1 滤波信号对应 I2,I2 滤波信号对应 I1）	I1 滤波信号边沿		I2 滤波信号边沿	
		上升	下降	上升	下降
仅在 I1 计数	高	下降	上升	不计数	不计数
	低	上升	下降	不计数	不计数
仅在 I2 计数	高	不计数	不计数	上升	下降
	低	不计数	不计数	下降	上升
在 I1 和 I2 上计数	高	下降	上升	上升	下降
	低	上升	下降	下降	上升

表 20-1 计数方向与编码器信号的关系

外部增量编码器可直接与 MCU 连接，无需外部接口逻辑。而比较器通常用于将编码器的差分输出转换为数字信号，这极大地增加了抗噪声能力。编码器的第三个输出端用于指示机械零点，可以连接到外部中断输入引脚以触发一次计数复位。

下图给出了计数器操作的例子，给出了计数信号产生和方向控制。同样给出了选择双边沿时，输入抖动如何被补偿。输入抖动可能发生在传感器靠近切换点处。

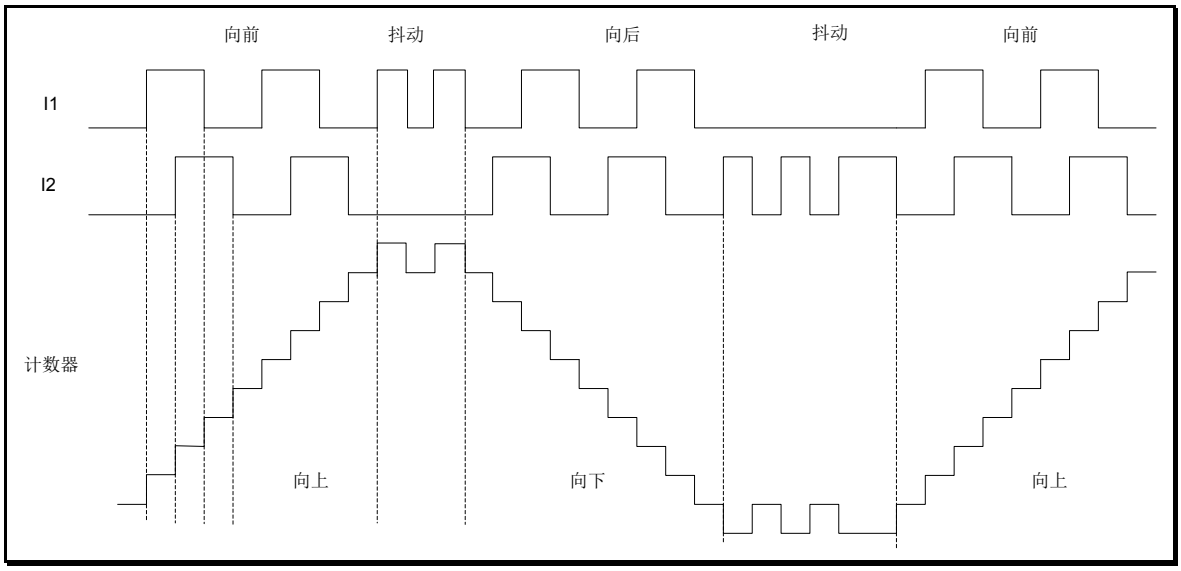


图 20-22 编码器接口模式下的计数操作

下图给出了计数器在 I1 滤波信号极性反相时的计数过程（除了 CC1POL = '1'，其他配置与上面一致）

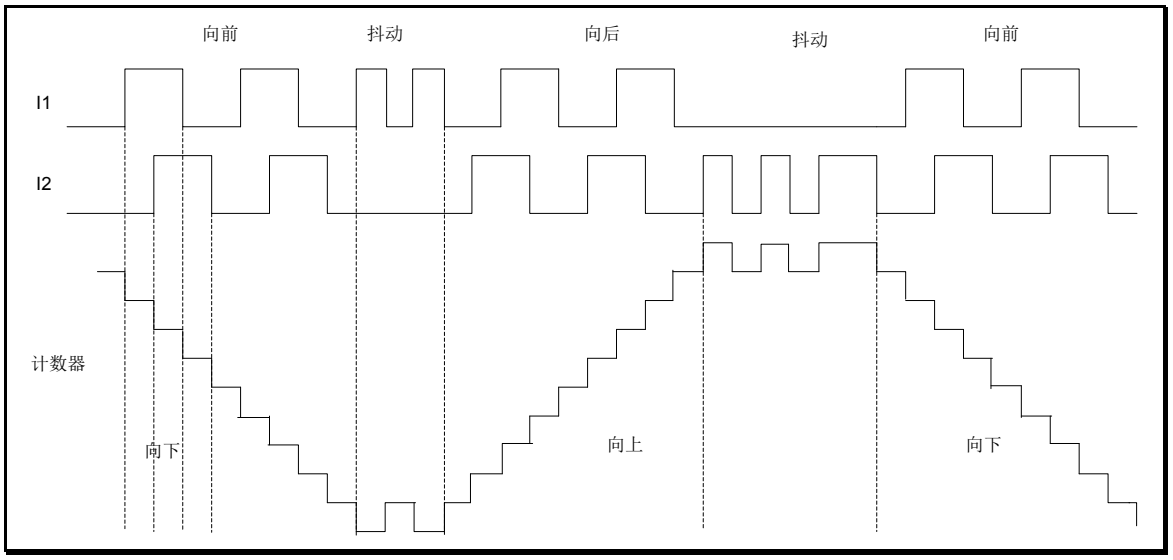


图 20-23 I1 滤波后极性反相时编码器接口例子

当配置为编码器接口模式时，定时器可提供传感器的当前位置信息。配置一个额外定时器为捕获模式，用于测量两个编码器事件的间隔，根据间隔时常获取动态信息（速度、加速度、减速度）。编码器用于指示机械零点的输出就是此用处。根据编码器两个事件间隔，可以周期性的读取计数器的值。如果允许，可以将计数器值锁存到第三个输入捕获寄存器（捕获信号必须是周期性的且可由其它定时器产生）。条件允许时，可通过实时时钟产生 DMA 请求的方式读取计数器值。

20. 4. 13 输入异或功能

通过 AD16C4Tn_CON2 寄存器中 I1FSEL 位，可将通道 1 的输入滤波器连接到 XOR 门的输出端，XOR 门联合了 CH1_IN、CH2_IN 和 CH3_IN 三个输入引脚。

XOR 输出用于定时器的所有输入功能，如触发或输入捕获。该功能参见下节的霍尔传感器接口。

20. 4. 14 霍尔传感器接口

高级控制定时器产生 PWM 信号驱动电机，另外一个定时器作为“接口定时器”。“接口定时器”捕获 3 个引脚输入（CH1_IN，CH2_IN，CH3_IN），引脚信号经过一个异或门后连接到 I1 的输入通道（置位 AD16C4Tn_CON2 寄存器中的 I1SEL 位选择）。

从模式控制器配置为复位模式，3 个输入后的异或信号作为从输入。因此，只要 3 个输入信号有一个变化，计数器重新从 0 开始计数。霍尔输入端的任何变化都可触发创建一个时钟基准。

20. 4. 15 外部触发的同步

AD16C4Tn 定时器可在多种模式下与外部触发同步：复位模式、门控模式及触发模式。

20. 4. 15. 1 复位模式

计数器及其预分频器可以在响应触发输入事件时重新初始化。此外，若 AD16C4Tn_CON1 寄存器的 UERSEL 位为低时会产生一次更新事件 UEV。所有预载寄存器（AD16C4Tn_AR，AD16C4Tn_CCVALn）都会因更新事件 UEV 而被更新。

在下面例子中，I1 输入端的上升沿让递增计数被清空：

◇ 配置通道 1 上检测 I1 上的上升沿。配置输入滤波周期（本例无需滤波器，故 I1FLT = "0000"）。触发捕获分频器没有使用，无需配置。CC1SSEL 位只选择输入捕获源，AD16C4Tn_CHMR1 寄存器中 CC1SSEL = "01"。AD16C4Tn_CCEP 寄存器中 CC1POL = 0 以确定极性（只检测上升沿）。

◇ 定时器配置位复位模式：AD16C4Tn_SMCON 寄存器中 SMODS = "100"。选择 I1 作为输入源：AD16C4Tn_SMCON 寄存器中 TSSEL = "101"。

◇ 启动计数器：AD16C4Tn_CON1 寄存器中 CNTEN = '1'。

计数器依据内部时钟开始计数，正常计数直到 I1 上出现上升沿。当 I1 上出现上升沿时，计数器会被清零且从 0 重新开始计数。同时，标志位置位（AD16C4Tn_RIF 寄存器中 TRGIF 位），如果中断及 DMA 使能（取决于 AD16C4Tn_IER 寄存器中的 TRGIT 和 AD16C4Tn_DMAEN 寄存器中的 TRGDMA 位），会发送中断及 DMA 请求。

下图给出了当自动重载寄存器 AD16C4Tn_AR = 0x36 时的信号变化。由于 I1 输入的再同步电路，I1 上的上升沿和计数器实际复位之间会存在延时（包含 2~3 个模块时钟周期的同步延时）。

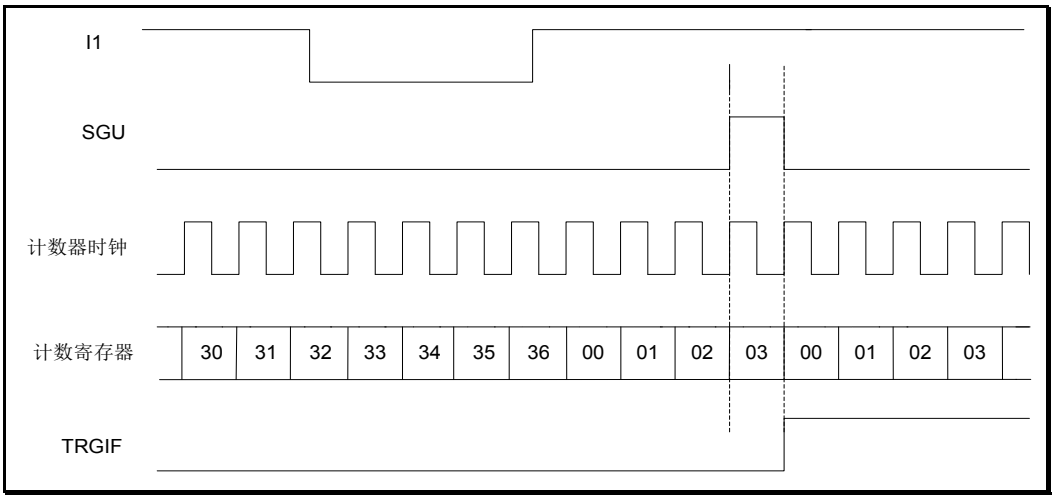


图 20-24　复位模式控制电路

20. 4. 15. 2　门控模式

计数器根据选中的输入电平被使能。

下面的例子中，计数器只在 I1 输入为低电平时才递增计数：

◇　配置通道 1 在 I1 上检测低电平。配置输入滤波周期（本例不需要滤波器，I1FLT = "0000"）。触发捕获分频器没有使用，无需配置。AD16C4Tn_CHMR1 寄存器中的 CC1SSEL = "01"，选择输入捕获源。AD16C4Tn_CCEP 寄存器中 CC1POL = '1'，确认极性（只检测低电平）。

◇　配置定时器为门控模式：AD16C4Tn_SMCON 寄存器中 SMODS = "101"。选择 I1 作为输入源：AD16C4Tn_SMCON 寄存器中 TSSEL = "101"。

◇　使能计数器：AD16C4Tn_CON1 寄存器中 CNTEN = '1'（门控模式中，如果 CNTEN = '0'，无论触发输入为何电平，计数器都不会启动）。

只要 I1 为低电平，计数器依据内部时钟开始计数，一旦 I1 为高则停止计数。由于 I1 输入端再同步电路的原因，I1 上出现上升沿和计数器实际停止之间会有一定的延时。

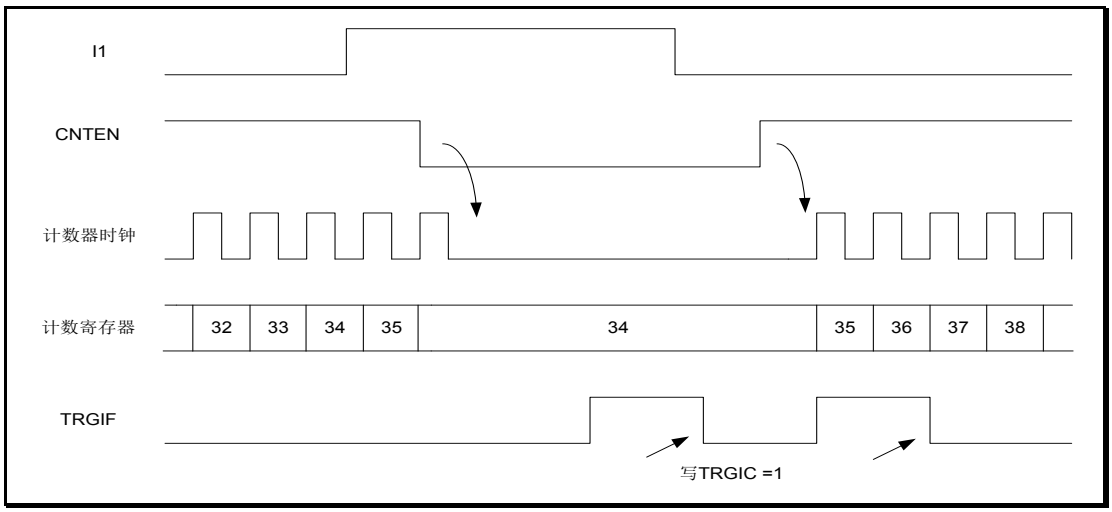


图 20-25　门控模式控制电路

20. 4. 15. 3 触发模式

输入端选中的事件可以使能计数器。

下面的例子中，I2 输入端上的上升沿可以启动递增计数：

◇ 配置通道 2 可以检测 I2 上的上升沿。配置滤波时间（本例不需要滤波，I2FLT = "0000"）。触发捕获分频器没有使用，无需配置。AD16C4Tn_CHMR1 寄存器中 CC2SSEL = "01",用于选择捕获源。AD16C4Tn_CCEP 寄存器中 CC2POL = '1'，确认极性（只检测低电平）。

◇ 配置定时器为触发模式：AD16C4Tn_SMCON 寄存器中 SMODS = "110"。AD16C4Tn_SMCON 寄存器中 TSSEL = "110"，用于选择输入源。

I2 上出现上升沿时，计数器开始依据内部时钟计数并置位 TRGIF 标志位。

由于 I2 输入的再同步原因，I2 上出现上升沿和计数器实际停止之间会有一定的延时。

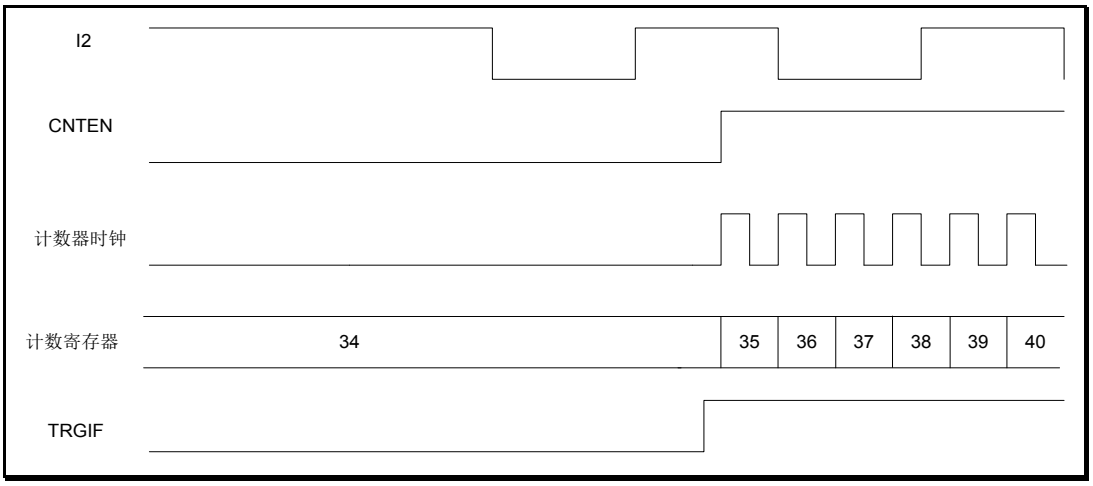


图 20-26 触发模式控制电路

20. 4. 15. 4 选择外部时钟源 2 的触发模式

外部时钟源 2 可和其他模式一起使用（除编码模式）。ET 信号可作为外部时钟输入，另一个输入可选择为触发输入（复位模式、门控模式或触发模式）。不推荐用 AD16C4Tn_SMCON 寄存器的 TSSEL 位选择 ET 作为 TI。

下面的例子中，一旦 I1 上出现上升沿时，计数器会依据 ET 信号的每个上升沿递增计数。

◇ 通过 AD16C4Tn_SMCON 寄存器，配置外部触发输入电路，过程如下：

ETFLT = "000"：无滤波

ETPSEL = "00"：禁止分频

ETPOL = '0'：检测 ET 的上升沿，ECM2EN = '1'使能外部时钟模式 2

◇ 配置通道 1 检测 I 的上升沿，过程如下：

I1FLT = "0000"：无滤波。

触发捕获分频器没有使用，无需配置。

AD16C4Tn_CHMR1 寄存器中 CC1SSEL = "01"选择输入捕获源，
AD16C4Tn_CCEP 寄存器的 CC1POL = '0'确认极性（只检测上升沿）。

◇ 配置定时器为触发模式：AD16C4Tn_SMCON 寄存器中 SMODS = "110"。
AD16C4Tn_SMCON 寄存器中 TSSEL = "101"选择 I1 作为输入源。

I1 上出现上升沿时，计数器使能且 TRGIF 标志位置位，然后计数器根据 ET 上的上升沿开始计数。

由于 ETFP 输入再同步电路的原因，ET 信号的上升沿和实际计数器的复位会有延时。

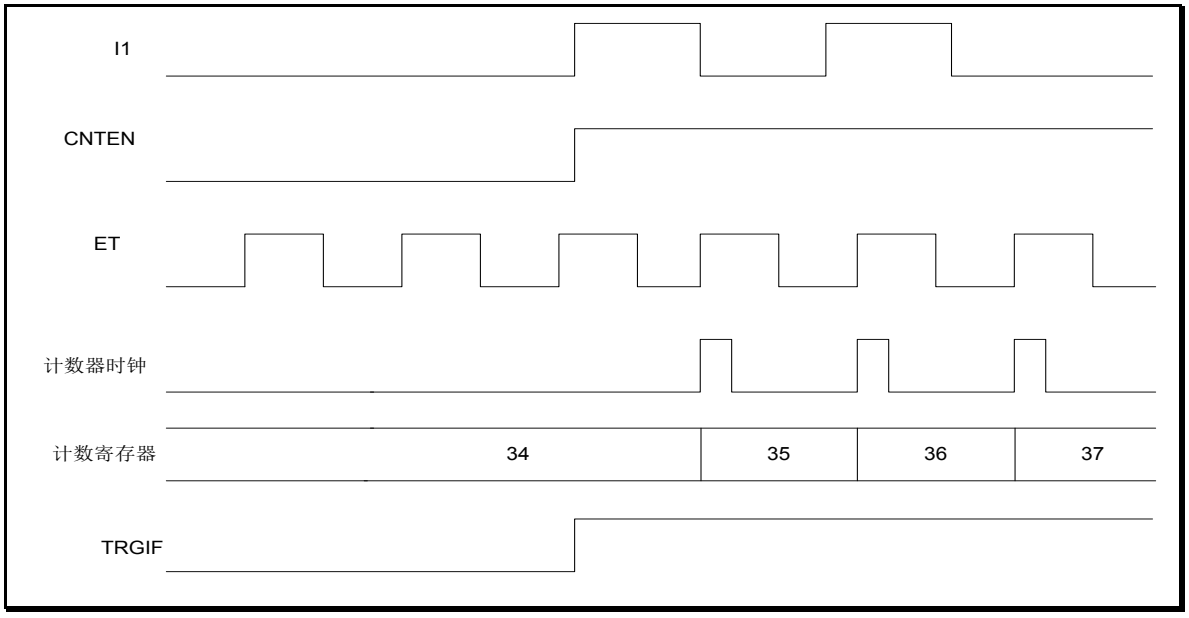


图 20-27 外部时钟源 2＋触发模式下的控制电路

20. 4. 16 调试模式

当微控制器进入调试模式（Cortex™-M 内核终止），AD16C4Tn 计数器可停止计数。

20. 4. 17 6 步PWM生成

当通道配置为互补输出时，CHnOMOD,CCnEN 及 CCnNEN 位都是预装载位。发生换相事件时，预装载位传递给寄存器的影子位。因此，可预先配置下一步，同时也可修改所有通道的配置。软件置位 AD16C4Tn_SGE 寄存器的 COM 位可以产生 COM 换相事件，硬件 TRGI 上升沿也可产生。

COM 事件发生时标志位（AD16C4Tn_IFM 寄存器的 COMIM 位）置位可触发中断（AD16C4Tn_IVS 寄存器的 COMIT 位置位）或 DMA 请求（AD16C4Tn_DMAEN 寄存器的 COMDMA 位置位）。

下图描述了当发生 COM 事件时，3 种不同配置，CHnO 和 CHnON 输出情况。

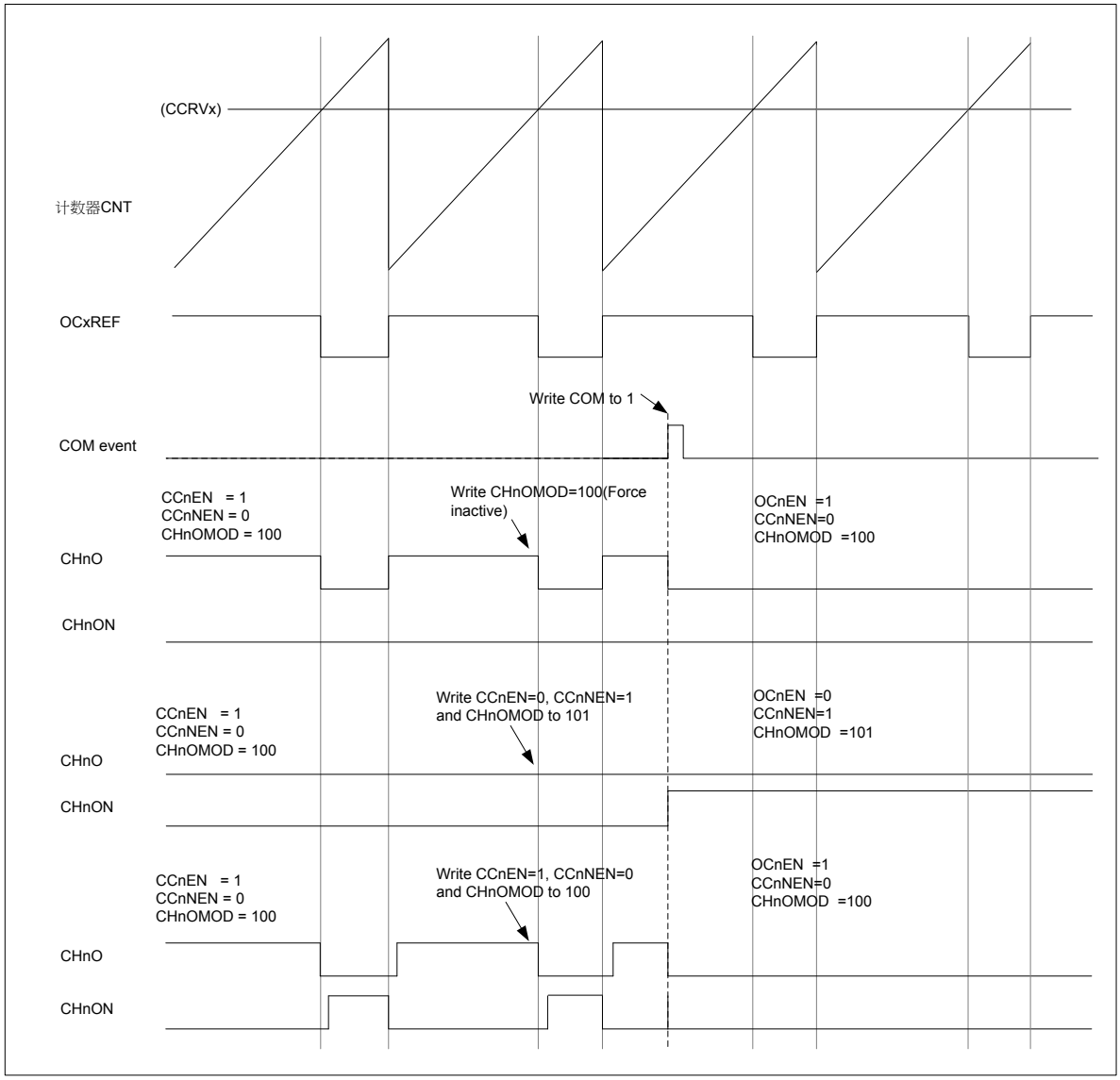


图 20-28　6 步 PWM 波形示例

20. 5 特殊功能寄存器

20. 5. 1 寄存器列表

AD16C4T 寄存器列表		
寄存器名称	偏移地址	描述
AD16C4Tn_CON1	000_H	控制寄存器 1
AD16C4Tn_CON2	004_H	控制寄存器 2
AD16C4Tn_SMCON	008_H	从模式控制寄存器
AD16C4Tn_IER	$00C_H$	中断使能寄存器
AD16C4Tn_IDR	010_H	中断禁止寄存器
AD16C4Tn_IVS	014_H	中断有效状态寄存器
AD16C4Tn_RIF	018_H	原始中断标志寄存器
AD16C4Tn_IFM	$01C_H$	中断标志屏蔽寄存器
AD16C4Tn_ICR	020_H	中断清零寄存器
AD16C4Tn_SGE	024_H	软件生成事件寄存器
AD16C4Tn_CHMR1	028_H	捕获/比较模式寄存器 1
AD16C4Tn_CHMR2	$02C_H$	捕获/比较模式寄存器 2
AD16C4Tn_CCEP	030_H	捕获/比较使能寄存器
AD16C4Tn_COUNT	034_H	计数寄存器
AD16C4Tn_PRES	038_H	预分频寄存器
AD16C4Tn_AR	$03C_H$	自动重载寄存器
AD16C4Tn_REPAR	040_H	重复计数寄存器
AD16C4Tn_CCVAL1	044_H	捕获/比较寄存器 1
AD16C4Tn_CCVAL2	048_H	捕获/比较寄存器 2
AD16C4Tn_CCVAL3	$04C_H$	捕获/比较寄存器 3
AD16C4Tn_CCVAL4	050_H	捕获/比较寄存器 4
AD16C4Tn_BDCFG	054_H	刹车和死区时间寄存器
AD16C4Tn_DMAEN	058_H	DMA 使能寄存器

20. 5. 2 寄存器描述

20. 5. 2. 1 控制寄存器 1（AD16C4Tn_CON1）

控制寄存器 1（AD16C4Tn_CON1）

偏移地址：000_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DBGSEL	Reserved	OCCISS			OCCISP	DFCKSEL		APREN	CMSEL		DIRSEL	SPMEN	UERSEL	DISUE	CNTEN

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
DBGSEL	Bit 15	R/W	**调试时通道输出状态选择：** 0：通道输出为高阻态 1：通道输出保持
Reserved	Bit 14	-	**保留，必须保持为复位值**
OCCISS	Bit 13-11	R/W	**通道输出清除内部触发源选择：** 000：无效 001：清除触发源通道 0 010：清除触发源通道 1 011：清除触发源通道 2 100：清除触发源通道 3
OCCISP	Bit 10	R/W	**通道输出清除内部触发源极性：** 0：低电平有效 1：高电平有效
DFCKSEL	Bit 9-8	R/W	**死区发生器和数字滤波器工作时钟频率 Fdfck 选择位** 该位为计数器时钟（INT_CLK）频率，显示了死区时间和由死区发生器与数字滤波器（ET，In）所用的采样时钟（t_{DTS}）之间的分频关系。 00：$t_{DTS}=t_{INT\_CLK}$ 01：$t_{DTS}=2*t_{INT\_CLK}$ 10：$t_{DTS}=4*t_{INT\_CLK}$ 11：预留,不允许编程该值。
ARPEN	Bit 7	R/W	**计数器自动重载寄存器预载** 0：AD16C4Tn_AR 寄存器未缓冲 1：AD16C4Tn_AR 寄存器被装入缓冲器
CMSEL	Bit 6-5	R/W	**中心对齐模式选择** 00：**边沿对齐模式**。计数器根据方向为（DIRSEL）来向上或向下计数。 01：**中央对齐模式 1**。计数器以交替方式向上或向下计数。仅当计数器向下计数时，配置为输出的通道（AD16C4Tn_CHMRn 寄存器中

			CCnSSEL=00）的输出比较中断标志位才会被设置。 10：**中央对齐模式 2**。计数器以交替方式向上或向下计数。仅当计数器向上计数时，配置为输出的通道（AD16C4Tn_CHMRn 寄存器中 CCnSSEL=00）的输出比较中断标志位才会被设置。 11：**中央对齐模式 3**。计数器以交替方式向上或向下计数。当计数器向上或向下计数时，配置为输出的通道（AD16C4Tn_CHMRn 寄存器中 CCnSSEL=00）的输出比较中断标志位均会被设置。 注意：当计数器使能时（CNTEN=1），不允许从边沿对齐模式转换到中央对齐模式。
DIRSEL	Bit 4	R/W	**计数器方向选择** 0：计数器向上计数 1：计数器向下计数 注意：当计数器配置为中央对齐模式或者编码器模式时，该位只读。
SPMEN	Bit 3	R/W	**使能单脉冲模式** 0：当发生更新事件时，计数器不停止。 1：当发生下一次更新事件（CNTEN 位清零）时，计数器停止。
UERSEL	Bit 2	R/W	**选择更新事件请求** 该位由软件置 1 或清零，来选择 UEV 事件源。 0：如果更新中断或 DMA 请求使能，则下述任一时间都可产生更新中断或 DMA 请求： – 计数器上溢/下溢 – 设置 SGU 位 – 从模式控制器产生的更新 1：如果更新中断或 DMA 请求使能，仅计数器上溢/下溢才能产生更新中断或 DMA 请求中断
DISUE	Bit 1	R/W	**禁止更新事件** 该位由软件置 1 或清零来使能/禁止 UEV 事件的产生。 0：UEV 使能。更新事件（UEV）由下列任一事件产生： – 计数器上溢/下溢 –设置 SGU 位 –从模式控制器产生的更新 缓冲寄存器载入他们的预载值。 1：UEV 禁止。不产生更新事件，影子寄存器保持他们的值（ARRV, PSCV, CCRVx）。如果从从模式控制器接收到硬件复位，计数器和预分频器将

			被重新初始化。
CNTEN	Bit 0	R/W	**使能计数器** 0:计数器禁止 1:计数器使能 注意：如果软件设置了 CNTEN 位，外部时钟，门控模式和编码器模式才能工作。触发模式可由硬件自动设置 CNTEN 位。

20. 5. 2. 2 控制寄存器 2（AD16C4Tn_CON2）

控制寄存器 2（AD16C4Tn_CON2）																															
偏移地址：004_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	OISS4	OISS3N	OISS3	OISS2N	OISS2	OISS1N	OISS1	I1FSEL	TRGOSEL			CCDMASEL	CCUSEL	Reserved	CCPCEN

Reserved	Bit 31-15	-	**保留，必须保持为复位值**
OISS4	Bit 14	R/W	**通道 4 输出的空闲状态选择位** 参考 OISS1 描述
OISS3N	Bit 13	R/W	**通道 3 互补输出的空闲状态选择位** 参考 OISS1N 描述
OISS3	Bit 12	R/W	**通道 3 输出的空闲状态 3 选择位** 参考 OISS1 描述
OISS2N	Bit 11	R/W	**通道 2 互补输出的空闲状态选择位** 参考 OISS1N 描述
OISS2	Bit 10	R/W	**通道 2 输出的空闲状态选择位** 参考 OISS1 描述
OISS1N	Bit 9	R/W	**通道 1 互补输出的空闲状态选择位** 0：当 GOEN=0，在一段死区时间后，CH1ON=0 1：当 GOEN=0，在一段死区时间后，CH1ON=1 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 1,2,或 3 后，OISS1N 不可更改。
OISS1	Bit 8	R/W	通道 1 输出的空闲状态选择位 0：当 GOEN=0，如果 CH1ON 已实现，在一段死区时间后，CH1O=0 1：当 GOEN=0，如果 CH1ON 已实现，在一段死区时间后，CH1O=1 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 1,2,或 3 后，OISS1N 不可更改。
I1FSEL	Bit 7	R/W	**选择 I1 引脚功能** 0：AD16C4Tn_CH1 引脚与 I1 输入连接 1：AD16C4Tn_CH1, CH2 和 CH3 引脚与 I1 输入（XOR）连接。
TRGOSEL	Bit 6-4	R/W	**选择主模式 TRGOUT 输出** 为同步（TRGOUT），该位可选择在主模式下发送至从计数器的信息。 000：**复位**–AD16C4Tn_SGE 寄存器中的 SGU 位

			被采用为触发输出（TRGOUT）。如果复位由触发输入生成（从模式控制器配置复位模式），则相较与实际复位，TRGOUT 上的信号将会延迟。 001：**使能**–计数器使能信号被用作触发输出（TRGOUT）。在从计数器使能的情况下，该设置用于在同一时间启动数次或者用来控制窗口。计数器使能信号是由 CNTEN 控制位与配置为门控模式的触发输入进行 OR 操作产生的。当计数器使能信号由触发输入控制，TRGOUT 上会产生延迟，除非被选为主/从模式（参考 AD16C4Tn_SMCON 寄存器中的 MSCFG 位的描述）。 010：**更新**–更新事件被选为触发输出（TRGOUT）。举例，主计数器可被用作从计数器的预分频器。 011：**比较脉冲**–一旦捕获或者比较匹配发生，当 CH1CCIF 标志位被置起（即便已为高电平），触发输出会发送一个正脉冲。 100：**比较**- 通道 1 比较输出信号用作触发输出 TRGOUT 101：**比较**- 通道 2 比较输出信号用作触发输出 TRGOUT 110：**比较**- 通道 3 比较输出信号用作触发输出 TRGOUT 111：**比较**- 通道 4 比较输出信号用作触发输出 TRGOUT
CCDMASEL	Bit 3	R/W	**使能捕获/比较 DMA** 0：当 CCn 事件发生，会发出 CCn DMA 请求。 1：当发生更新时间，会发出 CCn DMA 请求。
CCUSEL	Bit 2	R/W	**选择捕获比较更新** 0：当捕获/比较控制位为预载值（CCPCEN=1），则当设置 SGCOM 位时才会被更新。 1：当捕获/比较控制位为预载值（CCPCEN=1），则当设置 SGCOM 位或者当 TI 边沿上升时，均会被更新。 注意：该位只有用作于通道时才有互补输出。
Reserved	Bit 1	-	**保留，必须保持为复位值**
CCPCEN	Bit 0	R/W	**使能捕获比较预载控制** 0：CCnEN, CCnNEN 和 CHnOMOD 不为预载值。 1：CCnEN, CCnNEN 和 CHnOMOD 为预载值。当写入预载值后，仅当发生换相事件（COM）时（SGCOM 位设置或者 TI 上检测到上升沿由 CCUSEL 位决定），才会被更新）。 注意：该位只有用作于通道时才有互补输出。

20. 5. 2. 3 从模式控制寄存器（AD16C4Tn_SMCON）

从模式控制寄存器（AD16C4Tn_SMCON）

偏移地址：008_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ETPOL	ECM2EN	ETPSEL		ETFLT				MSCFG	TSSEL			OCCS	SMODS		

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
ETPOL	Bit 15	R/W	**选择外部触发信号边沿** 0：正向 ET,高电平有效或上升沿有效。 1：反正 ET, 低电平有效或下降沿有效。
ECM2EN	Bit 14	R/W	**使能外部时钟模式 2** 该位使能外部时钟模式 2 0：禁止外部时钟模式 2 1：使能外部时钟模式 2。计数器由 ETFP 信号上的有效边沿计数。 注意： 1. 设置 ECM2EN 位与选择外部时钟模式 1 且 TI 与 ETFP 相连接（SMODS=111 和 TSSEL=111）具有相同的效果。 2. 可同时使用外部时钟模式 2 与下列从模式：复位模式，门控模式和除法模式。在这种情况下，TI 不能与 ETFP 相连接（TSSEL 不能设置为 111。 3. 如果外部时钟模式 1 和外部时钟模式 2 同时使能，外部时钟输入为 ETFP。
ETPSEL	Bit 13-12	R/W	**选择外部触发分频** 外部触发信号频率最大为 AD16C4Tn CLK 频率的 1/4。可使能预分频器来减小 ETFP 频率。该位有效用于输入高速外部时钟的情况。 00：预分频器关闭 01：ETFP 频率 2 分频 10：ETFP 频率 4 分频 11：ETFP 频率 8 分频
ETFLT	Bit 11-8	R/W	**选择外部触发滤波采样时钟** 该位定义了对 ET 信号的采样频率和数字滤波器的滤波长度。 数字滤波器由一个事件计数器组成，每 N 个连续事件才视为一个有效边沿。 0000：无滤波器，采样频率为 f_{DTS} 0001：$f_{SAMPLING} = f_{INT\_CLK}$, N = 2

			0010：$f_{SAMPLING} = f_{INT\_CLK}$, N = 4 0011：$f_{SAMPLING} = f_{INT\_CLK}$, N = 8 0100：$f_{SAMPLING} = f_{DTS}$ / 2, N = 6 0101：$f_{SAMPLING} = f_{DTS}$ / 2, N = 8 0110：$f_{SAMPLING} = f_{DTS}$ / 4, N = 6 0111：$f_{SAMPLING} = f_{DTS}$ / 4, N = 8 1000：$f_{SAMPLING} = f_{DTS}$ / 8, N = 6 1001：$f_{SAMPLING} = f_{DTS}$ / 8, N = 8 1010：$f_{SAMPLING} = f_{DTS}$ / 16, N = 5 1011：$f_{SAMPLING} = f_{DTS}$ / 16, N = 6 1100：$f_{SAMPLING} = f_{DTS}$ / 16, N = 8 1101：$f_{SAMPLING} = f_{DTS}$ / 32, N = 5 1110：$f_{SAMPLING} = f_{DTS}$ / 32, N = 6 1111：$f_{SAMPLING} = f_{DTS}$ / 32, N = 8 注意：当 ETFLT[3：0] = 1, 2 or 3 时，公式中的 f_{DTS} 由 INT_CLK 取代。
MSCFG	Bit 7	R/W	**主/从模式配置** 0：无动作 1：延迟触发输入（In）上的事件来允许当前计时器和其从器件之间的同步。该设置有效用于使用单个外部事件来同步多个计时器。
TSSEL	Bit 6-4	R/W	**选择从模式触发源** 该位用来选择不同的触发输入来同步计数器。 000：内部触发 0（IT0），PIS 通道 12 001：内部触发 1（IT1），PIS 通道 13 010：内部触发 2（IT2），PIS 通道 14 011：内部触发 3（IT3），PIS 通道 15 100：I1 双边沿边沿检出（I1F_ED） 101：I1 滤波输入 1 110：I2 滤波输入 2 111：外部触发输入 注意：为了避免错误边沿检测，该位在不使用时（SMODS=000）才能改变。
OCCS	Bit 3	R/W	**输出通道清除源选择** 0：OCCISS 选择的内部通道 1：ETFP
SMODS	Bit 2-0	R/W	**选择从模式功能** 当选择外部信号，触发信号 TI 的有效边沿与外部输入的极性有关系（详见输入控制寄存器和控制寄存器描述） 000：**禁止从模式**–如果 CNTEN = ‘1’，则预分频器直接由内部时钟计数。 001 **编码器模式 1**–计数器向上/向下计数 I2 边沿，取决于 I1 电平。

			010：**编码器模式 2** -计数器向上/向下计数 I1 边沿，取决于 I2 电平 011 ：**编码器模式 3** -计数器向上/向下计数 I1 和 I2 边沿，取决于另一个输入的电平。 100：**复位模式–**选中的触发输入的上升沿重新初始化计数器，生成寄存器的更新 101：**门控模式–**当触发输入 TI 为高电平，计数器时钟使能。一旦触发变为低电平，计数器停止计数（并非复位）。计数器的启动和停止均受控制。 110：**触发模式–**计数器在触发信号 TI 的上升沿处启动（不复位）。仅寄存器的启动受控制。 111：**外部时钟源 1–**计数器在 TI 的上升沿计数 注意：如果 I1 双边沿检出被选为触发输入（TSSEL='100'），不能使用门控模式。I1 每一次转换，I1 双边沿检出就会输出 1 个脉冲，而门控模式则是检查触发信号的电平。 注意：在发生来自主计时器的接收事件之前，从计时器的时钟必须先使能，且在接收来自主计时器的触发过程中，从计数器时钟不能即时更改。

20. 5. 2. 4 中断使能寄存器（AD16C4Tn_IER）

中断使能寄存器（AD16C4Tn_IER）																													
偏移地址：$00C_H$																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CC4OIT	CC3OIT	CC2OIT	CC1OIT	Reserved	BRKIT	TRGIT	COMIT	CC4IT	CC3IT	CC2IT	CC1IT	UIT

Reserved	Bit 31-13	-	保留，必须保持复位值。
CC4OIT	Bit 12	W	**使能捕获/比较 4 捕获溢出中断** 0：无效 1：使能
CC3OIT	Bit 11	W	**使能捕获/比较 3 捕获溢出中断** 0：无效 1：使能
CC2OIT	Bit 10	W	**使能捕获/比较 2 捕获溢出中断** 0：无效 1：使能
CC1OIT	Bit 9	W	**使能捕获/比较 1 捕获溢出中断** 0：无效 1：使能
Reserved	Bit 8	-	保留，必须保持为复位值
BRKIT	Bit 7	W	**使能刹车中断** 0：无效 1：使能
TRGIT	Bit 6	W	**使能触发中断** 0：无效 1：使能
COMIT	Bit 5	W	**使能 COM 中断** 0：无效 1：使能
CC4IT	Bit 4	W	**使能捕获/比较 4 中断** 0：无效 1：使能
CC3IT	Bit 3	W	**使能捕获/比较 3 中断** 0：无效 1：使能
CC2IT	Bit 2	W	**使能捕获/比较 2 中断** 0：无效 1：使能
CC1IT	Bit 1	W	**使能捕获/比较 1 中断** 0：无效 1：使能
UIT	Bit 0	W	**使能更新事件中断** 0：无效 1：使能

20. 5. 2. 5 中断禁止寄存器（AD16C4Tn_IDR）

中断禁止寄存器（AD16C4Tn_IDR）																													
偏移地址：0010_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CC4OIT	CC3OIT	CC2OIT	CC1OIT	Reserved	BRKIT	TRGIT	COMIT	CC4IT	CC3IT	CC2IT	CC1IT	UIT

Reserved	Bit 31-13	-	**保留，必须保持复位值。**
CC4OIT	Bit 12	W	**使能捕获/比较 4 捕获溢出中断** 0：无效 1：使能
CC3OIT	Bit 11	W	**使能捕获/比较 3 捕获溢出中断** 0：无效 1：使能
CC2OIT	Bit 10	W	**使能捕获/比较 2 捕获溢出中断** 0：无效 1：使能
CC1OIT	Bit 9	W	**使能捕获/比较 1 捕获溢出中断** 0：无效 1：使能
Reserved	Bit 8	-	**保留，必须保持为复位值**
BRKIT	Bit 7	W	**禁止刹车中断** 0：无效 1：禁止
TRGIT	Bit 6	W	**禁止触发中断** 0：无效 1：禁止
COMIT	Bit 5	W	**禁止 COM 中断** 0：无效 1：禁止
CC4IT	Bit 4	W	**禁止捕获/比较 4 中断** 0：无效 1：禁止
CC3IT	Bit 3	W	**禁止捕获/比较 3 中断** 0：无效 1：禁止
CC2IT	Bit 2	W	**禁止捕获/比较 2 中断** 0：无效 1：禁止
CC1IT	Bit 1	W	**禁止捕获/比较 1 中断**

			0：无效 1：禁止
UIT	Bit 0	W	**禁止更新中断** 0：无效 1：禁止

20. 5. 2. 6 中断有效状态寄存器（AD16C4Tn_IVS）

中断有效状态寄存器（AD16C4Tn_IVS）																													
偏移地址：0014_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CC4OIT	CC3OIT	CC2OIT	CC1OIT	Reserved	BRKIT	TRGIT	COMIT	CC4IT	CC3IT	CC2IT	CC1IT	UIT

Reserved	Bit 31-13	-	**保留**
CC4OIT	Bit 12	W	**使能捕获/比较 4 捕获溢出中断** 0：无效 1：使能
CC3OIT	Bit 11	W	**使能捕获/比较 3 捕获溢出中断** 0：无效 1：使能
CC2OIT	Bit 10	W	**使能捕获/比较 2 捕获溢出中断** 0：无效 1：使能
CC1OIT	Bit 9	W	**使能捕获/比较 1 捕获溢出中断** 0：无效 1：使能
Reserved	Bit 8	-	**保留，必须保持为复位值**
BRKIT	Bit 7	R	**刹车中断状态** 0：禁止 1：使能
TRGIT	Bit 6	R	**触发中断状态** 0：禁止 1：使能
COMIT	Bit 5	R	**COM中断状态** 0：禁止 1：使能
CC4IT	Bit 4	R	**通道 4 捕获/比较中断状态** 0：禁止 1：使能
CC3IT	Bit 3	R	**通道3捕获/比较中断状态** 0：禁止 1：使能
CC2IT	Bit 2	R	**通道 2 捕获/比较中断状态** 0：禁止 1：使能
CC1IT	Bit 1	R	**通道2捕获/比较中断状态**

			0：禁止 1：使能
UIT	Bit 0	R	**更新事件中断状态** 0：禁止 1：使能

20. 5. 2. 7 原始中断标志寄存器（AD16C4Tn_RIF）

中断标志寄存器（AD16C4Tn_RIF）																															
偏移地址：0018_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CH4OVIF	CH3OVIF	CH2OVIF	CH1OVIF	Reserved	BRKIF	TRGIF	COMIF	CH4CCIF	CH3CCIF	CH2CCIF	CH1CCIF	UEVTIF

Reserved	Bit 31-13	-	**保留**
CH4OVIF	Bit 12	R	**通道 4 捕获/比较捕获溢出中断标志** 仅当相应的通道配置为捕获输入状态时，该标志位才由硬件设置。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未检测到捕获溢出 1：当 CH4CCIF 标志为置起时，捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
CH3OVIF	Bit 11	R	**通道 3 捕获/比较捕获溢出中断标志** 仅当相应的通道配置为捕获输入状态时，该标志位才由硬件设置。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未检测到捕获溢出 1：当 CH3CCIF 标志为置起时，捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
CH2OVIF	Bit 10	R	**通道 2 捕获/比较捕获溢出中断标志** 仅当相应的通道配置为捕获输入状态时，该标志位才由硬件设置。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未检测到捕获溢出 1：当 CH2CCIF 标志为置起时，捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
CH1OVIF	Bit 9	R	**通道 1 捕获/比较捕获溢出中断标志** 仅当相应的通道配置为捕获输入状态时，该标志位才由硬件设置。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未检测到捕获溢出 1：当 CH1CCIF 标志为置起时，捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
Reserved	Bit 8	R	**保留**
BRKIF	Bit 7	R	**刹车中断标志** 如果刹车中断使能，一旦刹车输入有效，该标志

			位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生刹车事件。 1：在刹车输入上检测到有效电平
TRGIF	Bit 6	R	**触发中断标志** 如果触发中断使能，当从模式控制器在门控模式以外的所有模式下使能，发生触发事件时（TI 上检测到有效边沿），该标志位被硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生触发事件 1：触发中断被挂起
COMIF	Bit 5	R	**COM 中断标志** 如果 COM 中断使能，当发生 COM 事件（当捕获/比较控制位 CCnEN, CCnNEN, CHnOMOD 更新后），该标志位被硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生 COM 事件 1：COM 中断被挂起
CH4CCIF	Bit 4	R	**通道 4 捕获/比较中断标志** 参考 CH1CCIF 描述
CH3CCIF	Bit 3	R	**通道 4 捕获/比较中断标志** 参考 CH1CCIF 描述
CH2CCIF	Bit 2	R	**通道 2 捕获/比较中断标志** 参考 CH1CCIF 描述
CH1CCIF	Bit 1	R	**通道 1 捕获/比较 1 中断标志** **如果 CC1 通道配置为输出：** 如果中断使能，除去中央对齐模式的情况（参考 AD16C4Tn_CON1 寄存器中 CMSEL 的描述），当计数值与比较值匹配，该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：不匹配。 1：AD16C4Tn_COUNT 计数值与 AD16C4Tn_CCVAL1 值匹配。当 AD16C4Tn_CCVAL1 寄存器值大于 AD16C4Tn_AR 值，发生计数器上溢时（递增模式和递增/递减模式）或下溢时（递减模式），CH1CCIF 为被置起。 **如果 CC1 通道配置为输入：** 发生捕获时，该位由硬件置起。该位可通过软件或者读取 AD16C4Tn_CCVAL1 寄存器来清零。 0：未发生输入捕获 1：计数值捕获至 AD16C4Tn_CCVAL1 寄存器（I1 上检测到与选中极性匹配的边沿）
UEVTIF	Bit 0	R	**更新事件中断标志**

			如果更新中断使能，当发生更新事件，该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生更新。 1：更新中断被挂起。当寄存器更新时，该位被硬件置起： –当重复计数器值发生上溢或者下溢（若重复计数器=0，则更新）和当 AD16C4Tn_CON1 寄存器中 DISUE=0 –当使用 AD16C4Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时，如果 AD16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0 –当 CNT 由触发事件来重新初始化，如果 AD16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0

20. 5. 2. 8 中断标志屏蔽寄存器（AD16C4Tn_IFM）

中断标志屏蔽寄存器（AD16C4Tn_IFM）																															
偏移地址：$001C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CH4OVIM	CH3OVIM	CH2OVIM	CH1OVIM	Reserved	BRKIM	TRGIM	COMIM	CH4CCIM	CH3CCIM	CH2CCIM	CH1CCIM	UEVTIM

Reserved	Bit 31-13	-	**保留**
CH4OVIM	Bit 12	R	**屏蔽通道 4 捕获/比较捕获溢出中断标志** 0：未检测到捕获溢出 1：当 CH4CCIF 标志为置起时，捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
CH3OVIM	Bit 11	R	**屏蔽通道 3 捕获/比较捕获溢出中断标志** 0：未检测到捕获溢出 1：当 CH3CCIF 标志为置起时，捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
CH2OVIM	Bit 10	R	**屏蔽通道 2 捕获/比较捕获溢出中断标志** 0：未检测到捕获溢出 1：当 CH2CCIF 标志为置起时，捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
CH1OVIM	Bit 9	R	**屏蔽通道 1 捕获/比较捕获溢出中断标志** 0：未检测到捕获溢出 1：当 CH1CCIF 标志为置起时，捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
Reserved	Bit 8	R	**保留**
BRKIM	Bit 7	R	**刹车中断标志屏蔽** 如果刹车中断使能，一旦刹车输入有效，该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生刹车事件。 1：在刹车输入上检测到有效电平
TRGIM	Bit 6	R	**屏蔽触发中断标志** 如果触发中断使能，当从模式控制器在门控模式以外的所有模式下使能，发生触发事件时（TI 上检测到有效边沿），该标志位被硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生触发事件 1：触发中断被挂起
COMIM	Bit 5	R	**屏蔽 COM 中断标志** 如果 COM 中断使能，当发生 COM 事件（当捕获

			/比较控制位 CCnEN, CCnNEN, CHnOMOD 更新后），该标志位被硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生 COM 事件 1：COM 中断被挂起
CH4CCIM	Bit 4	R	**屏蔽通道 4 捕获/比较中断标志** 参考 CH1CCIM 描述
CH3CCIM	Bit 3	R	**屏蔽通道 3 捕获/比较中断标志** 参考 CH1CCIM 描述
CH2CCIM	Bit 2	R	**屏蔽通道 2 捕获/比较中断标志** 参考 CH1CCIM 描述
CH1CCIM	Bit 1	R	**屏蔽通道 1 捕获/比较中断标志** **如果通道 1 配置为输出：** 如果中断使能，除去中央对齐模式的情况（参考 AD16C4Tn_CON1 寄存器中 CMSEL 的描述），当计数值与比较值匹配，该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：不匹配。 1：AD16C4Tn_COUNT 计数值与 AD16C4Tn_CCVAL1 值匹配。当 AD16C4Tn_CCVAL1 寄存器值大于 AD16C4Tn_AR 值，发生计数器上溢时（递增模式和递增/递减模式）或下溢时（递减模式），CH1CCIF 为被置起。 **如果通道配置为输入：** 发生捕获时，该位由硬件置起。该位可通过软件或者读取 AD16C4Tn_CCVAL1 寄存器来清零。 0：未发生输入捕获 1：计数值捕获至 AD16C4Tn_CCVAL1 寄存器（I1 上检测到与选中极性匹配的边沿）
UEVTIM	Bit 0	R	**屏蔽更新事件中断标志** 如果更新中断使能，当发生更新事件，该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生更新。 1：更新中断被挂起。当寄存器更新时，该位被硬件置起： –当重复计数器值发生上溢或者下溢（若重复计数器=0，则更新）和当 AD16C4Tn_CON1 寄存器中 DISUE=0 –当使用 AD16C4Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时，如果 AD16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0

			–当 CNT 由触发事件来重新初始化，如果 AD16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0

20. 5. 2. 9 中断清零寄存器（AD16C4Tn_ICR）

中断清零寄存器（AD16C4Tn_ICR）																													
偏移地址：0020_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CH4OVIC	CH3OVIC	CH2OVIC	CH1OVIC	Reserved	BRKIC	TRGIC	COMIC	CH4CCIC	CH3CCIC	CH2CCIC	CH1CCIC	UEVTIC

Reserved	Bit 31-13	-	**保留**
CH4OVIC	Bit 12	R	**通道 4 捕获/比较捕获溢出中断标志清除** 0：无效 1：CH4OVIF 清除
CH3OVIC	Bit 11	R	**通道 3 捕获/比较捕获溢出中断标志** 0：无效 1：CH3OVIF 清除
CH2OVIC	Bit 10	R	**通道 2 捕获/比较捕获溢出中断标志** 0：无效 1：CH2OVIF 清除
CH1OVIC	Bit 9	R	**通道 1 捕获/比较捕获溢出中断标志** 0：无效 1：CH1OVIF 清除
Reserved	Bit 8	R	**保留**
BRKIC	Bit 7	C_W1	**清刹车中断** 0：无效 1：清零（AD16C4Tn_RIF）
TRGIC	Bit 6	C_W1	**清触发中断** 0：无效 1：清零（AD16C4Tn_RIF）
COMIC	Bit 5	C_W1	**清 COM 中断清零** 0：无效 1：清零（AD16C4Tn_RIF）
CH4CCIC	Bit 4	C_W1	**通道 4 捕获/比较中断清零** 参考 CH1CCIC 描述
CH3CCIC	Bit 3	C_W1	**通道 3 捕获/比较中断清零** 参考 CH1CCIC 描述
CH2CCIC	Bit 2	C_W1	**通道 2 捕获/比较中断清零** 参考 CH1CCIC 描述
CH1CCIC	Bit 1	C_W1	**通道 1 捕获/比较中断清零** 0：无效 1：清零（AD16C4Tn_RIF）
UEVTIC	Bit 0	C_W1	**更新事件中断清零**

			0：无效 1：清零（AD16C4Tn_RIF）

20. 5. 2. 10 软件生成事件寄存器（AD16C4Tn_SGE）

软件生成事件寄存器（AD16C4Tn_SGE）

偏移地址：0024_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								SGBRK	SGTRG	SGCOM	SGCC4E	SGCC3E	SGCC2E	SGCC1E	SGU

Reserved	Bit 31-8	-	保留
SGBRK	Bit 7	W	**软件生成刹车事件** 该位由软件设置来生成刹车事件，可由硬件自动清零。 0：无动作 1：产生刹车事件。GOEN 清零，BRKIF 标志位置起，产生相关中断或 DMA 传输。
SGTRG	Bit 6	W	**软件生成触发事件** 该位由软件设置来生成触发事件，可由硬件自动清零。 0：无动作 1：AD16C4Tn_RIF 寄存器中的 TRGIF 被置起，产生相关中断或 DMA 传输
SGCOM	Bit 5	W	**软件生成换相事件捕获** 该位由软件设置，由硬件自动清零。 0：无动作 1：当 CCPCEN 被置 1，则可更新 CCnEN, CCnNEN 和 CHnOMOD 注意：该位只有用作于通道时才有互补输出
SGCC4E	Bit 4	W	**软件触发通道 4 捕获/比较事件** 参考 SGCC1E 描述
SGCC3E	Bit 3	W	**软件触发通道 3 捕获/比较事件** 参考 SGCC1E 描述
SGCC2E	Bit 2	W	**软件触发通道 2 捕获/比较事件** 参考 SGCC1E 描述
SGCC1E	Bit 1	W	**软件触发通道 1 捕获/比较事件** 该位由软件设置来生成事件，可由硬件自动清零。 0：无动作 1：通道 1 上产生捕获/比较事件： **如果通道 1 配置为输出：** CH1CCIF 标志位被置起，产生相应中断或 DMA 请求发送。 **如果通道 1 配置为输入：**

			当前计数值捕获至 AD16C4Tn_CCVAL1 寄存器。CH1CCIF 标志位被置起，产生相应中断或 DMA 请求发送。CH1OVIF 标志位置起如果 CH1CCIF 标志位为高电平。
SGU	Bit 0	W	**软件触发更新事件** 该位由软件设置，可由硬件自动清零。 0：无动作 1：重新初始化计数器，更新寄存器。注意，预分频器也会被清零（但预分频比不会受到影响）。如果使用中央对齐模式或者 DIRSEL=0（递增），则计数器将清零；否则如果 DIRSEL=1（递减），则将使用自动重载入值。

20. 5. 2. 11 通道捕获模式寄存器 1（AD16C4Tn_CHMR1）

◆ 输出比较模式

通道捕获模式寄存器 1（AD16C4Tn_CHMR1）

偏移地址：0028_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CH2OCLREN	CH2OMOD			CH2OPREN	CH2OHSEN	CC2SSEL		CH1OCLREN	CH1OMOD			CH1OPREN	CH1OHSEN	CC1SSEL	

Reserved	Bit 31-16	-	**保留**
CH2OCLREN	Bit 15	R/W	**使能通道 2 输出比较清零** 参考 CH1OCLREN 描述
CH2OMOD	Bit 14-12	R/W	**通道 2 输出比较模式** 参考 CH1OMOD 描述
CH2OPREN	Bit 11	R/W	**使能通道 2 输出比较预载入** 参考 CH1OPREN 描述
CH2OHSEN	Bit 10	R/W	**使能通道 2 输出比较高速模式** 参考 CH1OHSEN 描述
CC2SSEL	Bit 9-8	R/W	**选择通道 2 输出比较** 该位定义了通道以及使用的输入的方向（输入/输出） 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I2** 10：通道配置为输入，捕获源为 **I1** 11：通道配置为输入，捕获源为 **ITn 或 I1 的双边沿检出**。 仅当内部触发输入通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才能工作。 注意：当通道为关闭状态时（AD16C4Tn_CCEP 中 CC2EN = '0'），CC2SSEL 为只写。
CH1OCLREN	Bit 7	R/W	**使能通道 1 输出比较清零** 0：通道 1 比较输出不会受到 ETFP 输入影响 1：当 ETFP 输入上检测到高电平时，通道 1 比较输出将被清零
CH1OMOD	Bit 6-4	R/W	**输出比较 1 模式** 该位定义了输出参考信号通道 1 比较输出的行为。通道 1 比较输出为高有效，CH1O 和 CH1ON 的有效电平由 CC1POL 和 CC1NPOL 位决定。 000 :**冻结**–输出比较寄存器 AD16C4Tn_CCVAL1

			寄存器和 AD16C4Tn_COUNT 计数器之间的比较对输出无效。 001：**发生匹配时设置通道 1 为有效电平**–当计数器 AD16C4Tn_COUNT 与捕获/比较寄存器 1AD16C4Tn_CCVAL1 发生匹配后，通道 1 比较输出信号强制为高电平。 010：**发生匹配时设置通道 1 为无效电平**–当计数器 AD16C4Tn_COUNT 与捕获/比较寄存器 1AD16C4Tn_CCVAL1 发生匹配后，通道 1 比较输出信号强制为低电平。 011：**翻转** -当 AD16C4Tn_COUNT=AD16C4Tn_CCVAL1，通道 1 比较输出发生翻转。 100：**强制为无效电平** - 通道 1 比较输出强制为低电平。 101：**强制为有效电平**- 通道 1 比较输出强制为高电平。 110：**PWM 模式 1** -在递增模式下，当 AD16C4Tn_COUNT<AD16C4Tn_CCVAL1，通道 1 为有效电平，否则，通道 1 为无效电平。在递减模式下，当 AD16C4Tn_COUNT>AD16C4Tn_CCVAL1，通道 1 为无效电平（通道 1 比较输出='0'），否则通道 1 为有效电平（通道 1 比较输出='1'）。 111：**PWM 模式 2** -在递增模式下，当 AD16C4Tn_COUNT<AD16C4Tn_CCVAL1，通道 1 为无效电平，否则，通道 1 为有效电平。在递减模式下，当 AD16C4Tn_COUNT>AD16C4Tn_CCVAL1，通道 1 为有效电平，否则通道 1 为无效电平。 注意： 1：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 3，且 CC1SSEL=00（通道为输出模式），该位将不能更改。 2：在 PWM 模式 1 和 2 中，仅当比较结果更改或当输出比较模式从冻结模式转换成 PWM 模式，比较输出电平才会更改。 3：对于有互补输出的通道，该位设置为预载值。如果 AD16C4Tn_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CH1OMOD 有效位才会设置为预载值中新的值。
CH1OPREN	Bit 3	R/W	**输出比较 1 预载使能** 0：AD16C4Tn_CCVAL1 的预载寄存器禁止。AD16C4Tn_CCVAL1 在任何时候都可写，新写入

			的值将立刻生效。 1：AD16C4Tn_CCVAL1 的预载寄存器使能。读/写操作可访问预载寄存器。每当发生一次更新事件，AD16C4Tn_CCVAL1 预载入值将会被填入有效寄存器。 注意： 1：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 3，且 CC1SSEL=00（通道为输出模式），该位将不能更改。 2：仅在单脉冲模式下（AD16C4Tn_CON1 寄存器中的 SPMEN 设置为 1），PWM 模式可在不经过验证预载寄存器的情况下使用。其他情况下的行为不做保证。
CH1OHSEN	Bit 2	R/W	**使能通道 1 输出比较高速** 该位用来加速在 CC 输出上的输入触发事件的效应。 0：当触发开启，通道 1 运作正常取决于计数器和 CCRV1 的值。当触发输入上发现边沿时，至少需要 5 个时钟周期来激活通道 1 输出。 1：触发输入上的有效沿类似于通道 1 输出上的比较匹配。设置 OC 为 1 用来比较电平，采样触发输入和激活通道 1 输出的延时将会减少至 3 个时钟周期。只有当通道配置为 PWM1 或 PWM2 模式，CH1OHSEN 才会起作用。
CC1SSEL	Bit 1-0	R/W	**捕获/比较 1 选择** 该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I1** 10：通道配置为输入，捕获源为 **I2** 11：通道配置为输入，捕获源为 **ITn 或 I1 的双边沿检出**。 只有当内部触发输入是通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才运行。 注意：当通道关闭（AD16C4Tn_CCEP 寄存器中的 CC1EN = ‘0’），CC1SSEL 为只写。

◆ 输入捕获模式

通道捕获模式寄存器 1（AD16C4Tn_CHMR1）																													
偏移地址：0028_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																I2FLT				IC2PRES		CC2SSEL		I1FLT				IC1PRES		CC1SSEL	

Reserved	Bit 31-16	-	**保留**
I2FLT	Bit 15-12	R/W	**通道 2 输入捕获滤波器** 参考 I1FLT 描述
IC2PRES	Bit 11-10	R/W	**通道 2 输入捕获预分频器** 参考 IC1PRES 描述
CC2SSEL	Bit 9-8	R/W	**选择通道 2 输入捕获源** 该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I2** 10：通道配置为输入，捕获源为 **I1** 11：通道配置为输入，捕获源为 **ITn 或 I1 的双边沿检出** 只有当内部触发输入是通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才运行 注意：当通道关闭（AD16C4Tn_CCEP 寄存器中的 CC2EN = '0'），CC2SSEL 为只写。
I1FLT	Bit 7-4	R/W	**I1 滤波器** 该位定义了 I1 输入的采样频率和数字滤波器的长度。 数字滤波器由一个事件计数器组成，每 N 个连续事件才视为一个有效边沿： 0000：无滤波器，采样频率为 f_{DTS} 0001：$f_{SAMPLING} = f_{INT\_CLK}$, N = 2 0010：$f_{SAMPLING} = f_{INT\_CLK}$, N = 4 0011：$f_{SAMPLING} = f_{INT\_CLK}$, N = 8 0100：$f_{SAMPLING} = f_{DTS} / 2$, N = 6 0101：$f_{SAMPLING} = f_{DTS} / 2$, N = 8 0110：$f_{SAMPLING} = f_{DTS} / 4$, N = 6 0111：$f_{SAMPLING} = f_{DTS} / 4$, N = 8 1000：$f_{SAMPLING} = f_{DTS} / 8$, N = 6 1001：$f_{SAMPLING} = f_{DTS} / 8$, N = 8 1010：$f_{SAMPLING} = f_{DTS} / 16$, N = 5 1011：$f_{SAMPLING} = f_{DTS} / 16$, N = 6

			1100：$f_{SAMPLING} = f_{DTS} / 16$, N = 8 1101：$f_{SAMPLING} = f_{DTS} / 32$, N = 5 1110：$f_{SAMPLING} = f_{DTS} / 32$, N = 6 1111：$f_{SAMPLING} = f_{DTS} / 32$, N = 8 注意：当 ICxF [3：0] = 1, 2 or 3 时，公式中的 f_{DTS} 由 INT_CLK 取代。
IC1PRES	Bit 3-2	R/W	**捕获比较器 1 输入预分频器** 该位定义了作用在 CC1 输入（I1）上的预分频比。当 CC1EN='0' （AD16C4Tn_CCEP 寄存器），预分频器将复位。 00：无预分频器。每当捕获输入上检测到边沿时，发生捕获动作。 01：每发生 2 次事件，执行一次捕获 10：每发生 4 次事件，执行一次捕获 11：每发生 8 次事件，执行一次捕获
CC1SSEL	Bit 1-0	R/W	**捕获比较器 1 输入源选择** 该位定义了通道和使用的输入的方向。 00：CC1 通道配置为**输出** 01：CC1 通道配置为输入，捕获源为 **I1** 10：CC1 通道配置为输入，捕获源为 **I2** 11：CC1 通道配置为输入，捕获源为 **ITn 或 I1 的双边沿检出**。 只有当内部触发输入是通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才运行 注意：当通道关闭（AD16C4Tn_CCEP 寄存器中的 CC1EN = '0'），CC1SSEL 为只写。

20. 5. 2. 12 通道捕获模式寄存器 2（AD16C4Tn_CHMR2）

◆ 输出比较模式

通道捕获模式寄存器 2（AD16C4Tn_CHMR2）																															
偏移地址：$002C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CH4OCLREN	CH4OMOD			CH4OPREN	CH4OHSEN	CC4SSEL		CH3OCLREN	CH3OMOD			CH3OPREN	CH3OHSEN	CC3SSEL	

Reserved	Bit 31-16	-	**保留**
CH4OCLREN	Bit 15	R/W	**通道 4 输出比较清零使能** 参考 CH1OCLREN 描述
CH4OMOD	Bit 14-12	R/W	**通道 4 输出比较模式** 参考 CH1OMOD 描述
CH4OPREN	Bit 11	R/W	**通道 4 输出比较预载入使能** 参考 CH1OPREN 描述
CH4OHSEN	Bit 10	R/W	**通道 4 输出比较高速使能** 参考 CH1OHSEN 描述
CC4SSEL	Bit 9-8	R/W	**通道 4 输出比较选择** 该位定义了通道以及使用的输入的方向（输入/输出）。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I4** 10：通道配置为输入，捕获源为 **I3** 11：通道配置为输入，捕获源为 **ITn 或 I1 的双边沿检出** 仅当内部触发输入通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才能工作。 注意：当通道为关闭状态时（AD16C4Tn_CCEP 中 CC4EN = '0'），CC4SSEL 为只写。
CH3OCLREN	Bit 7	R/W	**通道 3 输出比较清零使能** 参考 CH1OCLREN 描述
CH3OMOD	Bit 6-4	R/W	**通道 3 输出比较模式** 参考 CH1OMOD 描述
CH3OPREN	Bit 3	R/W	**通道 3 输出比较预载使能** 参考 CH1OPREN 描述
CH3OHSEN	Bit 2	R/W	**通道 3 输出比较高速使能** 参考 CH1OHSEN 描述
CC3SSEL	Bit 1-0	R/W	**通道 3 捕获/比较选择**

			该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I3** 10：通道配置为输入，捕获源为 **I4** 11：通道配置为输入，捕获源为 **ITn 或 I1 的双边沿检出**。 只有当内部触发输入是通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才运行。 注意：当通道关闭（AD16C4Tn_CCEP 寄存器中的 CC3EN = '0'），CC3SSEL 为只写

◆ 输入捕获模式

通道捕获模式寄存器 2（AD16C4Tn_CHMR2）																													
偏移地址：$002C_H$																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																I4FLT				IC4PRES		CC4SSEL		I3FLT				IC3PRES		CC3SSEL	

Reserved	Bit 31-16	-	**保留**
I4FLT	Bit 15-12	R/W	**通道 4 输入捕获滤波器** 参考 I1FLT 描述
IC4PRES	Bit 11-10	R/W	**通道 4 输入捕获预分频器** 参考 IC1PRES 描述
CC4SSEL	Bit 9-8	R/W	**选择通道 4 输入捕获源** 该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I4** 10：通道配置为输入，捕获源为 **I3** 11：通道配置为输入，捕获源为 **ITn 或 I1 的双边沿检出** 只有当内部触发输入是通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才运行 注意：当通道关闭（AD16C4Tn_CCEP 寄存器中的 CC4EN = '0'），CC4SSEL 为只写。
I3FLT	Bit 7-4	R/W	**通道 3 输入捕获滤波器** 参考 I1FLT 描述
IC3PRES	Bit 3-2	R/W	**通道 3 输入捕获预分频器** 参考 IC1PRES 描述
CC3SSEL	Bit 1-0	R/W	**选择通道 3 输入捕获源** 该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I3** 10：通道配置为输入，捕获源为 **I4** 11：通道配置为输入，捕获源为 **ITn 或 I1 的双边沿检出** 只有当内部触发输入是通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才运行 注意：当通道关闭（AD16C4Tn_CCEP 寄存器中的 CC3EN = '0'），CC3SSEL 为只写。

20. 5. 2. 13 捕获/比较使能极性寄存器（AD16C4Tn_CCEP）

通道捕获/比较使能寄存器（AD16C4Tn_CCEP）

偏移地址：0030_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		CC4POL	CC4EN	CC3NPOL	CC3NE	CC3POL	CC3EN	CC2NPOL	CC2NE	CC2POL	CC2EN	CC1NPOL	CC1NEN	CC1POL	CC1EN

Reserved	Bit 31-14	-	**保留**
CC4POL	Bit 13	R/W	**通道 4 捕获/比较输出极性** 参考 CC1POL 描述
CC4EN	Bit 12	R/W	**使能通道 4 捕获/比较输出** 参考 CC1EN 描述
CC3NPOL	Bit 11	R/W	**通道 3 捕获/比较互补输出极性** 参考 CC1NPOL 描述
CC3NEN	Bit 10	R/W	**使能通道 3 捕获/比较互补输出** 参考 CC1NEN 描述
CC3POL	Bit9	R/W	**通道 3 捕获/比较输出极性** 参考 CC1POL 描述
CC3EN	Bit 8	R/W	**使能通道 3 捕获/比较输出** 参考 CC1EN 描述
CC2NPOL	Bit 7	R/W	**通道 2 捕获/比较 2 互补输出极性** 参考 CC1NPOL 描述
CC2NEN	Bit 6	R/W	**使能通道 2 捕获/比较互补输出** 参考 CC1NEN 描述
CC2POL	Bit 5	R/W	**通道 2 捕获/比较输出极性** 参考 CC1POL 描述
CC2EN	Bit 4	R/W	**使能通道 2 捕获/比较输出** 参考 CC1EN 描述
CC1NPOL	Bit 3	R/W	**通道 1 捕获/比较互补输出极性** **通道配置为输出：** 0：CH1ON 高有效。 1：CH1ON 低有效。 **通道配置为输入：** 该位需和 CC1POL 一起使用来定义输入边沿的极性。参考 CC1POL 描述。 注意：对于有互补输出的通道，该位设置为预载值。如果 AD16C4Tn_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1NPOL 有效位才会设置为预载值中新的值。 注意：当 AD16C4Tn_BDCFG 寄存器中的

			LOCKLVL 位被设置为锁定级别 2 或 3，且 CC1SSEL=00（通道为输出模式），该位将不可写。
CC1NEN	Bit 2	R/W	**使能通道 1 捕获/比较互补输出** 0：关闭 - CH1ON 无效。CH1ON 电平取决于 GOEN, OFFSSI, OFFSSR, OISS1, OISS1N 和 CC1EN 的功能 1：开启 - CH1ON 为对应输出引脚上的输出信号，由 GOEN, OFFSSI, OFFSSR, OISS1, OISS1N 和 CC1EN 决定。 注意：对于有互补输出的通道，该位设置为预载值。如果 AD16C4Tn_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1NEN 有效位才会设置为预载值中新的值
CC1POL	Bit 1	R/W	**通道 1 捕获/比较输出极性** **通道配置为输出：** 0：CH1O 高有效 1：CH1O 低有效 **通道配置为输入：** CC1NPOL/CC1POL 为触发和捕获操作选择 I1 边沿检出和 I2 边沿检出的有效极性。 00：正向/上升沿 电路对 In 边沿检出的上升沿敏感（在复位，外部时钟或触发模式下，进行捕获或触发），In 边沿检出不反向（门控模式或编码器模式下，进行触发）。 01：反向/下降沿 电路对 In 边沿检出的下降沿敏感（在复位，外部时钟或触发模式下，进行捕获或触发），In 边沿检出反向（门控模式或编码器模式下，进行触发）。 10：保留，请勿使用该配置 11：正向/双边沿 电路对 In 边沿检出的上升沿和下降沿均敏感（在复位，外部时钟或触发模式下，进行捕获或触发），In 边沿检出不反向（门控模式下，进行触发）。在编码器接口模式下勿使用该配置。 注意：对于有互补输出的通道，该位设置为预载值。如果 AD16C4Tn_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1POL 有效位才会设置为预载值中新的值。 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 2 或 3，且 CC1SSEL=00（通道为输出模式），该位将不可写。
CC1EN	Bit 0	R/W	**使能通道 1 捕获/比较输出** **通道配置为输出：** 0：关闭 - CH1O 无效。CH1ON 电平取决于

			GOEN, OFFSSI, OFFSSR, OISS1, OISS1N 和 CC1EN 的功能 1：开启 - CH1O 为对应输出引脚上的输出信号，由 GOEN, OFFSSI, OFFSSR, OISS1, OISS1N 和 CC1EN 决定 **通道配置为输入：** 该位决定了计数值是否能捕获到输入捕获/比较寄存器 1（AD16C4Tn_CCVAL1）。 0：禁止捕获。 1：使能捕获。 注意：对于有互补输出的通道，该位设置为预载值。如果 AD16C4Tn_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1EN 有效位才会设置为预载值中新的值。

20. 5. 2. 14 计数寄存器（AD16C4Tn_COUNT）

计数寄存器（AD16C4Tn_COUNT）

偏移地址：0034_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	-	**保留**
CNTV	Bit 15-0	R/W	**计数值**

20. 5. 2. 15 预分频寄存器（AD16C4Tn_PRES）

预分频寄存器（AD16C4Tn_PRES）																															
偏移地址：0038_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PSCV															

Reserved	Bit 31-16	-	保留
PSCV	Bit 15-0	R/W	**时钟预分频器值** 计数器时钟频率（CK_CNT）=f_{CK_PSC} /（PSCV[15：0] + 1）。 每发生一次更新事件（包括当计数器由 AD16C4Tn_SGE 寄存器中的 SGU 位清零或当配置为复位模式时，通过触发控制器清零），PSCV 包含的值需填入到有效的预分频寄存器内。

20. 5. 2. 16 计数器自动装载寄存器（AD16C4Tn_AR）

计数器自动装载寄存器（AD16C4Tn_AR）																															
偏移地址：$003C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ARRV															

Reserved	Bit 31-16	-	保留
ARRV	Bit 15-0	R/W	**计数器自动装载值** ARRV 中的值将被载入实际的自动重载寄存器中。当自动重载值为空，计数器被屏蔽。

20. 5. 2. 17 重复计数寄存器（AD16C4Tn_REPAR）

重复计数寄存器（AD16C4Tn_REPAR）																															
偏移地址：0040_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								REPV							

Reserved	Bit 31-8	-	保留
REPV	Bit 7-0	R/W	**重复计数值** 当预载寄存器使能，该位允许用户设置比较寄存器的更新率（例如：预载到有效寄存器的周期性传输），同样也可以设置更新中断生成率。每次当 REPV_CNT 的相关递减计数器递减至 0，会产生更新事件，会从 REPV 值重新计数。因为只有当发生重复更新事件 U_RC 时，REPV_CNT 才会重新载入 REPV 值，所以只有在发生下一次重复更新事件时，写入 AD16C4Tn_REPAR 寄存器的值才会生效。 即，在 PWM 模式下，（REPV+1）相当于： –在边沿对齐模式下，（REPV+1）对应的是 PWM 的周期数 –在中央对齐模式下，（REPV+1）对应的是 1/2 PWM 的周期数

20. 5. 2. 18 通道捕获/比较寄存器 1（AD16C4Tn_CCVAL1）

通道捕获/比较寄存器 1 （AD16C4Tn_CCVAL1）																															
偏移地址：0044_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV1															

Reserved	Bit 31-16	-	保留
CCRV1	Bit 15-0	R/W	**捕获/比较值 1** **如果通道 CCn 配置为输出：** CCRVn 中的值将被载入实际的捕获/比较寄存器中（预载值）。 如果在 AD16C4Tn_CHMRn 寄存器中的预载功能没有选中，CCRVn 中的值将被永久载入；否则，每当发生更新事件，预载值将会复制到有效的捕获/比较寄存器中。有效捕获/比较寄存器中包含的值将会与 AD16C4Tn_COUNT 中的值进行比较，并在 OCn 上输出。 **如果通道 CCn 配置为输入：** CCRVn 为由上一个输入捕获事件（ICn）传输的计数值。

20. 5. 2. 19 通道捕获/比较寄存器 2（AD16C4Tn_CCVAL2）

通道捕获/比较寄存器 2（AD16C4Tn_CCVAL2）																															
偏移地址：0044_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV2															

Reserved	Bit 31-16	-	保留
CCRV2	Bit 15-0	R/W	**通道捕获/比较值 2** 参考 CCRV1 描述

20. 5. 2. 20 通道捕获/比较寄存器 3（AD16C4Tn_CCVAL3）

捕获/比较寄存器 3（AD16C4Tn_CCVAL3）																															
偏移地址：$004C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV3															

Reserved	Bit 31-16	-	**保留**
CCRV3	Bit 15-0	R/W	**捕获/比较值 3** 参考 CCRV1 描述

20. 5. 2. 21 通道捕获/比较寄存器 4（AD16C4Tn_CCVAL4）

通道捕获/比较寄存器 4（AD16C4Tn_CCVAL4）																															
偏移地址：0050_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV4															

Reserved	Bit 31-16	-	**保留**
CCRV4	Bit 15-0	R/W	**通道捕获/比较值 4** 参考 CCRV1 描述

20. 5. 2. 22 刹车和死区配置寄存器（AD16C4Tn_BDCFG）

刹车和死区配置寄存器（AD16C4Tn_BDCFG）																															
偏移地址：0054_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																GOEN	AOEN	BRKP	BRKEN	OFFSSR	OFFSSI	LOCKLVL		DT							

Reserved	Bit 31-16	-	保留
GOEN	Bit 15	R/W	**通道主要输出使能** 一旦刹车输入有效，该位会由硬件异步清零。该位可由软件置 1 或自动置 1，取决于 AOEN 位。该位仅作用于配置为输出的通道。 0：OC 和 OCN 输出禁止或强制为空闲状态。 1：如果 OC 和 OCN 各自的使能位都置 1（AD16C4Tn_CCEP 寄存器中的 CCnEN, CCnNEN），则 OC 和 OCN 输出使能。
AOEN	Bit 14	R/W	**通道自动输出使能** 0：GOEN 仅可由软件置位 1：在下一个更新事件发生时（如果刹车输入无效），GOEN 可由软件或自动置位。 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1，则该位不可更改。
BRKP	Bit 13	R/W	**选择通道刹车极性** 0：刹车输入 BRKP 为低有效 1：刹车输入 BRKP 为高有效 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1，则该位不可更改 注意：任何对该位的写操作都要延时 1 APB 时钟周期后才变为有效。
BRKEN	Bit 12	R/W	**使能刹车** 0：刹车输入（BRKP 和 CCS 时钟失效事件）禁止 1：刹车输入（BRKP 和 CCS 时钟失效事件）使能 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1，则该位不可更改 注意：任何对该位的写操作都要延时 1 APB 时钟周期后才变为有效。

OFFSSR	Bit 11	R/W	运行模式下的无效状态选择位 该位使用于，当 GOEN=1 时，被配置为输出并使用互补输出的通道。如果计时器中没有使用互补输出，则 OFFSSR 不使用。 0：无效状态时, OC/OCN 输出禁止（OC/OCN 使能输出信号=0）。 1：无效状态时，当 CCnEN=1 或 CCnNEN=1 时，便使能 OC/OCN 输出并将其设为无效电平。（OC/OCN 使能输出信号=1） 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 2，则该位不可更改。
OFFSSI	Bit 10	R/W	空闲模式下的空闲状态选择位该位使用于，当 GOEN=0 时，被配置为输出通道 0：无效状态时, OC/OCN 输出禁止（OC/OCN 使能输出信号=0）。 1：无效状态时，当 CCnEN=1 或 CCnNEN=1，便将 OC/OCN 输出首先强制为其空闲电平。（OC/OCN 使能输出信号=1） 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 2，则该位不可更改。
LOCKLVL	Bit 9-8	R/W	**锁定级别配置** 针对软件错误，该位提供写保护。 00：**锁定关闭**–不提供写保护 01：**锁定级别 1** = AD16C4Tn_BDCFG 寄存器中的 DT，AD16C4Tn_CON2 寄存器中的 OISSn 和 OISSnN，和 AD16C4Tn_BDCFG 寄存器中的 BRKEN/BRKP/AOEN 不再可写。 10：**锁定级别 2** = 锁定级别 1 + CC 极性位（AD16C4Tn_CCEP 寄存器中的 CCnPOL/CCnNPOL，只要相关通道由 CCnSSEL 配置为输出）以及 OFFSSR 和 OFFSSI 都不再可写。 11：**锁定级别 3** = 锁定级别 2 + CC 控制位（AD16C4Tn_CHMRn 寄存器中的 CHnOMOD 和 CHnOPREN，只要相关通道由 CCnSSEL 配置为输出）都不再可写。 注意：锁定配置为仅在复位后可写。一旦 AD16C4Tn_BDCFG 已写，其设置内容在下一个复位前都处于冻结状态。
DT	Bit 7-0	R/W	**死区延时设置值** 该位定义了互补输出之间插入的死区时间。DT 对应的就是该时间段。

			DT[7：5]=0xx => DT=DT[7:0]x t_{dtg}，式中 $t_{dtg}=t_{DTS}$ **DT[7：5]=10x** => DT=（64+DT[5:0]）xt_{dtg}，式中 $t_{dtg}=2xt_{DTS}$ **DT[7：5]=110**=> DT=（32+DT[4:0]）xt_{dtg}，式中 $t_{dtg}=8xt_{DTS}$ **DT[7：5]=111** => DT=（32+DT[4：0]）xt_{dtg}，式中 $t_{dtg}=16xt_{DTS}$ 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1，2 或 3，则该位不可更改

20. 5. 2. 23 DMA使能寄存器（AD16C4Tn_DMAEN）

DMA 使能寄存器（AD16C4Tn_DMAEN）																												
偏移地址：058_H																												
复位值：$00000000\_00000000\_00000000\_00000000_B$																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									TRGDMA	COMDMA	CC4DMA	CC3DMA	CC2DMA	CC1DMA	UDMA

Reserved	Bit 31-7	-	**保留，必须保持为复位值**
TRGDMA	Bit 6	R/W	**触发DMA请求使能** 0：DMA请求禁止 1：DMA请求使能
COMDMA	Bit 5	R/W	**COM DMA访问使能** 0：DMA请求禁止 1：DMA请求使能
CC4DMA	Bit 4	R/W	**捕获/比较值 4 DMA 访问使能** 0：DMA 请求禁止 1：DMA 请求使能
CC3DMA	Bit 3	R/W	**捕获/比较值 3 DMA 访问使能** 0：DMA 请求禁止 1：DMA 请求使能
CC2DMA	Bit 2	R/W	**捕获/比较值 2 DMA 访问使能** 0：DMA 请求禁止 1：DMA 请求使能
CC1DMA	Bit 1	R/W	**捕获/比较值 1 DMA 访问使能** 0：DMA 请求禁止 1：DMA 请求使能
UDMA	Bit 0	R/W	**更新 DMA 请求使能** 0：DMA 请求禁止 1：DMA 请求使能

第21章 通用定时器（GP16C4T）

21.1 概述

通用定时器（GP16C4T）包含一个 16 位自动重载计数器，该计数器由可配置的预分频器驱动。

通用定时器（GP16C4T）的用途广泛，可测量信号脉冲长度（输入捕获）或输出脉冲波形（比较输出、PWM）。

21.2 特性

- 16 位递增，递减，递增/递减自动加载计数器
- 16 位可编程预分频器，可对计数器工作时钟进行 1 到 65536 间的任意分频
- 多达四个独立信道
  - 输入捕获
  - 输出比较
  - PWM 产生（边沿与中央对齐模式）
  - 单脉冲输出模式
- 同步电路用于外部信号控制定时器及内部互联多个定时器
- 以下事件中产生中断/DMA：
  - 更新事件：计数器上溢/下溢，计数器初始化（通过软件或内/外部触发）
  - 触发事件（计数器起始、停止、初始化或内/外触发计数）
  - 输入捕获
  - 输出比较
- 支持增量（正交）编码及霍尔电路进行定位
- 触发输入可对外部时钟管理

21. 3 结构框图

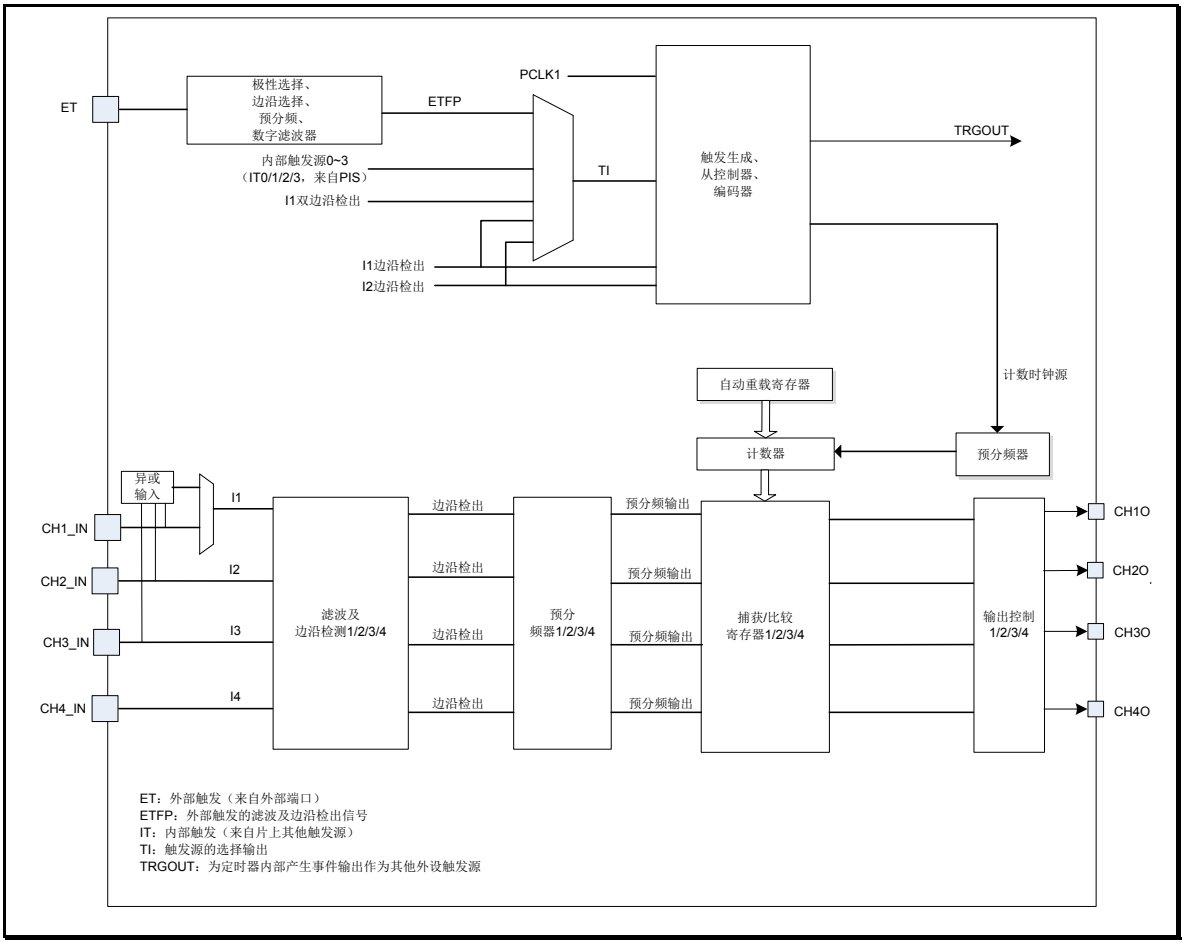


图 21-1　通用定时器结构框图

21.4 功能描述

21.4.1 预分频器

定时器包含一个 16-bit 的计数器（GP16C4Tn_COUNT），计数时钟由预分频寄存器（GP16C4Tn_PRES）进行分频。计数周期由自动重载计数器（GP16C4Tn_AR）设定。

自动重载寄存器（GP16C4Tn_AR）是一个可缓存的寄存器。当 GP16C4Tn_CON1 寄存器的 ARPEN 位复位时，GP16C4Tn_AR 寄存器重载功能失效，GP16C4Tn_AR 就是有效寄存器；ARPEN 置位时，GP16C4Tn_AR 寄存器具有重载功能，产生更新事件（UEV）时，加载值（GP16C4Tn_AR 寄存器值）更新到影子寄存器后才生效。

当 GP16C4Tn_CON1 寄存器中 DISUE 位为 0 时，计数器计数上溢（或递减下溢）时会产生更新事件（UEV）。同样，软件方式也可产生更新事件。GP16C4Tn_CON1 寄存器的 CNTEN 置位时，计数器开始计数。

注：计数器在 CNTEN 位置位 1 个时钟周期后开始计数。

预分频器可对定时器工作时钟进行 GP16C4Tn_PRES 寄存器值+1 次分频。由于 GP16C4Tn_PRES 是一个可重载寄存器，因此，定时器工作时可以对该寄存器进行修改，修改值在下次更新事件（UEV）后有效。

下图给出了定时器运行过程中改变预分频值时计数器的计数情况。

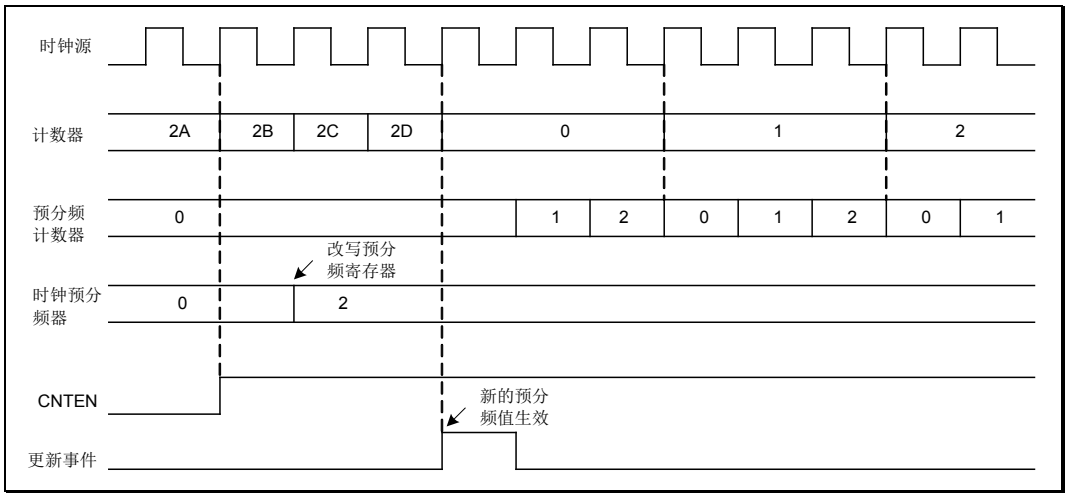


图 21-2 预分频值计数时序图

21. 4. 2 时钟源

计数器工作时钟可以选择内部时钟(INT_CLK)、外部时钟源 1（I1、I2、I3、I4）、外部时钟源 2（ET），内部触发输入（IT1、IT2、IT3、IT4）

21. 4. 2. 1 内部时钟源（INT_CLK）

若从模式控制器被关闭（GP16C4Tn_SMCON 寄存器内，SMODS= "000"），则 CNTEN，GP16C4Tn_CON1.DIRSEL 与 GP16C4Tn_SGE.SGU 位为实际控制位，这些位只能软件修改（SGU 位除外，仍硬件自动清除）。一旦 CNTEN 位被写为'1'，预分频器就由内部 INT_CLK 提供时钟。

下图给出了通常模式下控制电路和递增计数的情况，没有分频。

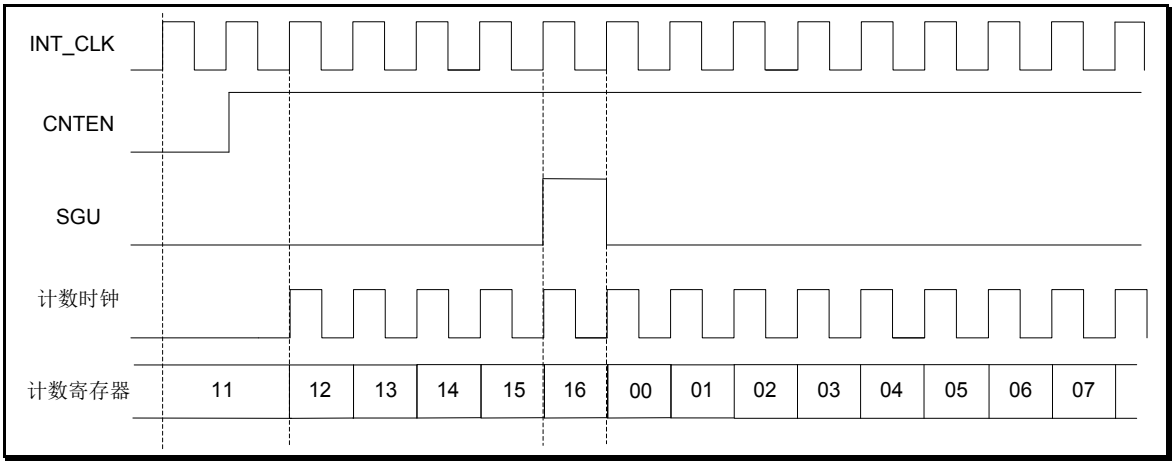


图 21-3 采用内部时钟计数

21. 4. 2. 2 外部时钟源 1

GP16C4Tn_SMCON 寄存器的 SMODS= "111"时，可选择外部时钟源 1。计数器可根据选定的上升沿或下降沿计数。

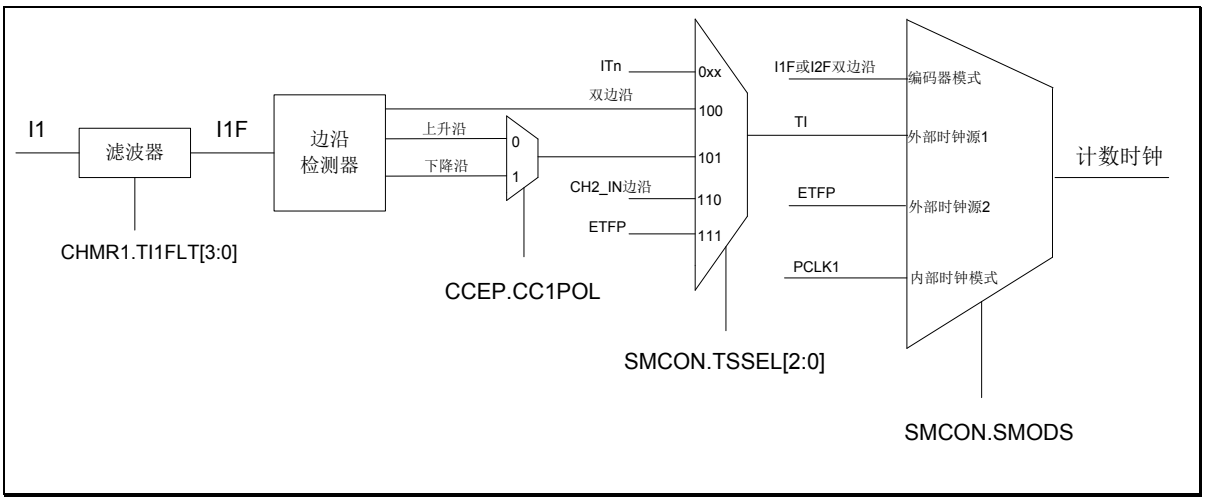


图 21-4 I1 外部时钟连接

配置计数器为外部时钟源 1，步骤如下：

6. GP16C4Tn_SMCON 寄存器中 SMODS = "111"，配置定时器外部时钟模式 1。
7. 设置 GP16C4Tn_SMCON 寄存器中的 TSSEL 选择外部时钟源。
8. 如外部时钟源为 I1，可配置 GP16C4Tn_CHMR1 寄存器 CC1SSEL = "01"，配置通道 1 检测 I1 输入的上升沿；设置 GP16C4Tn_CCEP 寄存器中 CC1POL = '0'，选择极性为上升沿。
9. 写 GP16C4Tn_CHMR1 寄存器的 I1FLT[3：0]位，配置输入滤波器时间（若没有滤波器需求，维持 I1FLT = "0000"）。
10. GP16C4Tn_CON1 寄存器中 CNTEN = '1'，使能计数器。

当 I1 上出现一次上升沿时，计数器计数一次且 TRGIF 标志位置位。

21. 4. 2. 3 外部时钟源 2

置位 GP16C4Tn_SMCON 寄存器的 ECM2EN 位选定外部时钟源 2。

计数器可对外部触发输入 ET 进行上升沿或下降沿计数。

下图给出了外部输入输入模块的概况。

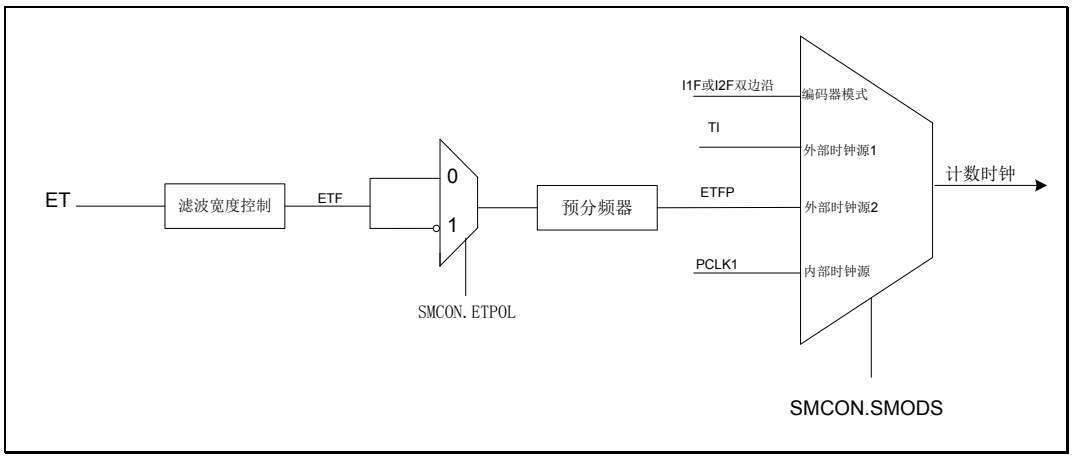


图 21-5 外部触发输入模块

配置计数器为外部时钟源 2，配置过程如下：

6. 设置 GP16C4Tn_SMCON 寄存器的 ETFLT[3：0]，配置输入滤波时间。
7. 设置 GP16C4Tn_SMCON 寄存器中 ETPSEL[1：0]，设置预分频器。
8. 设置 GP16C4Tn_SMCON 寄存器中 ETPOL，检测 ET 引脚上升沿或下降沿。
9. 设置 GP16C4Tn_SMCON 寄存器中 ECM2EN = '1'，使能外部时钟模式 2。
10. 设置 GP16C4Tn_CON1 寄存器的 CNTEN = '1'，使能计数器。

计数器每两个上升沿计一次数。

21. 4. 2. 4 内部触发输入（ITn）

当 GP16C4Tn_SMCON 寄存器的 SMODS= "111"，选定内部触发模式。计数器根据选定的内部输入端的上升或下降沿计数。

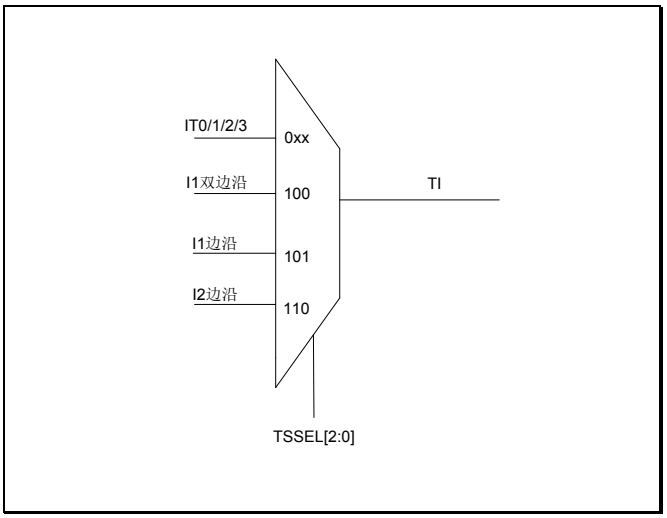


图 21-6 ITn 外部时钟连接

配置计数器在 ITn 输入端的上升沿递增计数，步骤如下：

4. GP16C4Tn_SMCON 寄存器中 SMODS = "111"，配置外部时钟模式 1。
5. GP16C4Tn_SMCON 寄存器的 TSSEL = "0xx"，选定 ITn 作为触发输入源。
6. GP16C4Tn_CON1 寄存器的 CNTEN = '1'，使能计数器。

ITn 产生上升沿时，计数器计数一次。ITn 上升沿与实际时钟间的延时，取决于 ITn 输入的再同步电路，一般为 2~3 个定时器模块时钟周期。

21.4.3 计数器模式

21.4.3.1 递增计数模式

在递增模式下，当 GP16C4Tn_REPAR 寄存器值为 0 时，计数器从 0 开始递增，直至 GP16C4Tn_AR 寄存器值；然后从 0 重新开始计数并产生一个更新事件（UEV）。当 GP16C4Tn_REPAR 寄存器不为 0 时，则在 GP16C4Tn_REPAR+1 次计数后产生更新事件。

当有更新事件（UEV）产生时，预装载寄存器会更新到影子寄存器，更新标志位（GP16C4Tn_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）：

- ◇ 更新 GP16C4Tn_REPAR 寄存器的值到影子寄存器
- ◇ 更新 GP16C4Tn_AR 寄存器的值到影子寄存器
- ◇ 更新 GP16C4Tn_PRES 寄存器的值到影子寄存器

下图为 GP16C4Tn_REPAR=0x0，GP16C4Tn_AR = 0x16，预分频设为 2 分频时的计数器时序。

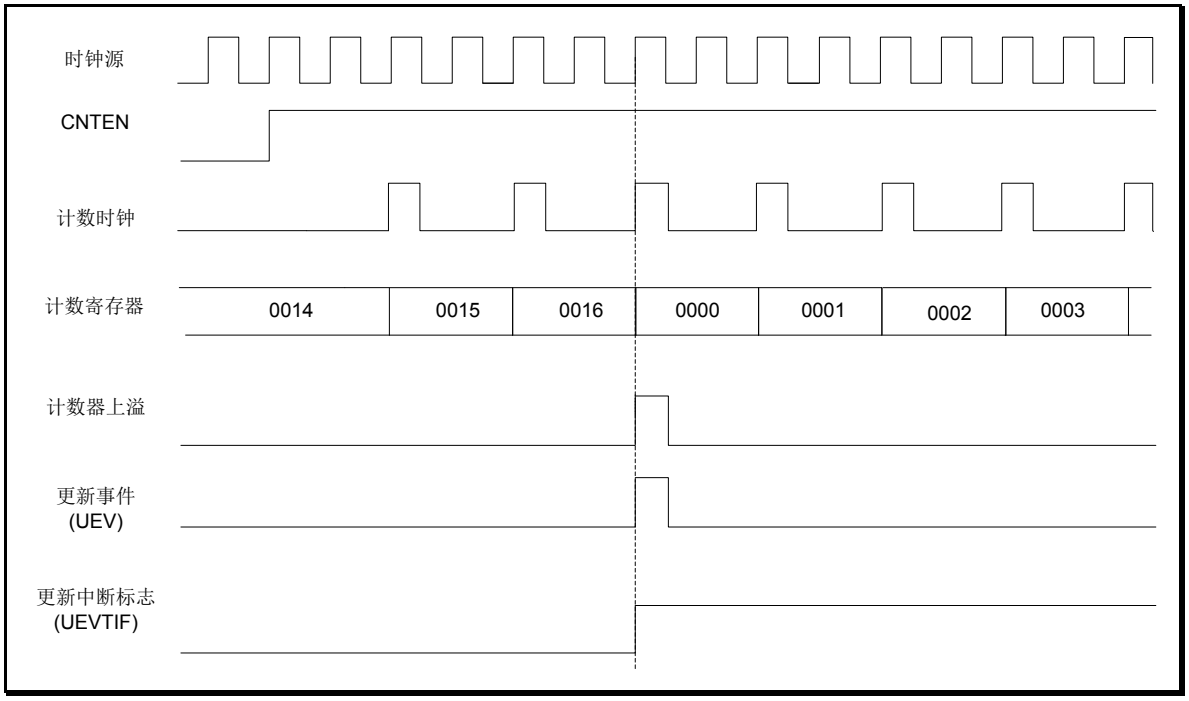


图 21-7 计数器时序图，内部时钟除以 1

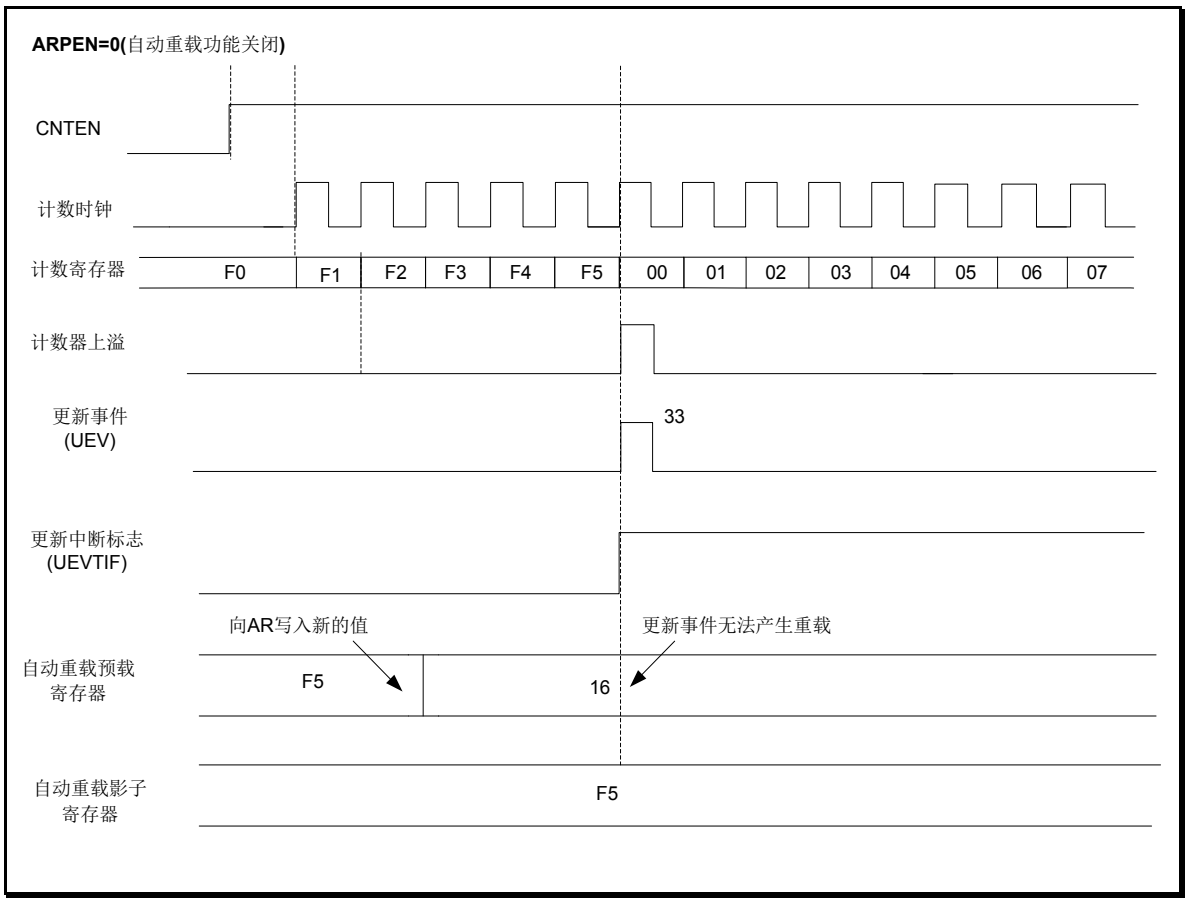


图 21-8　当 ARPEN=0 时计数器时序图

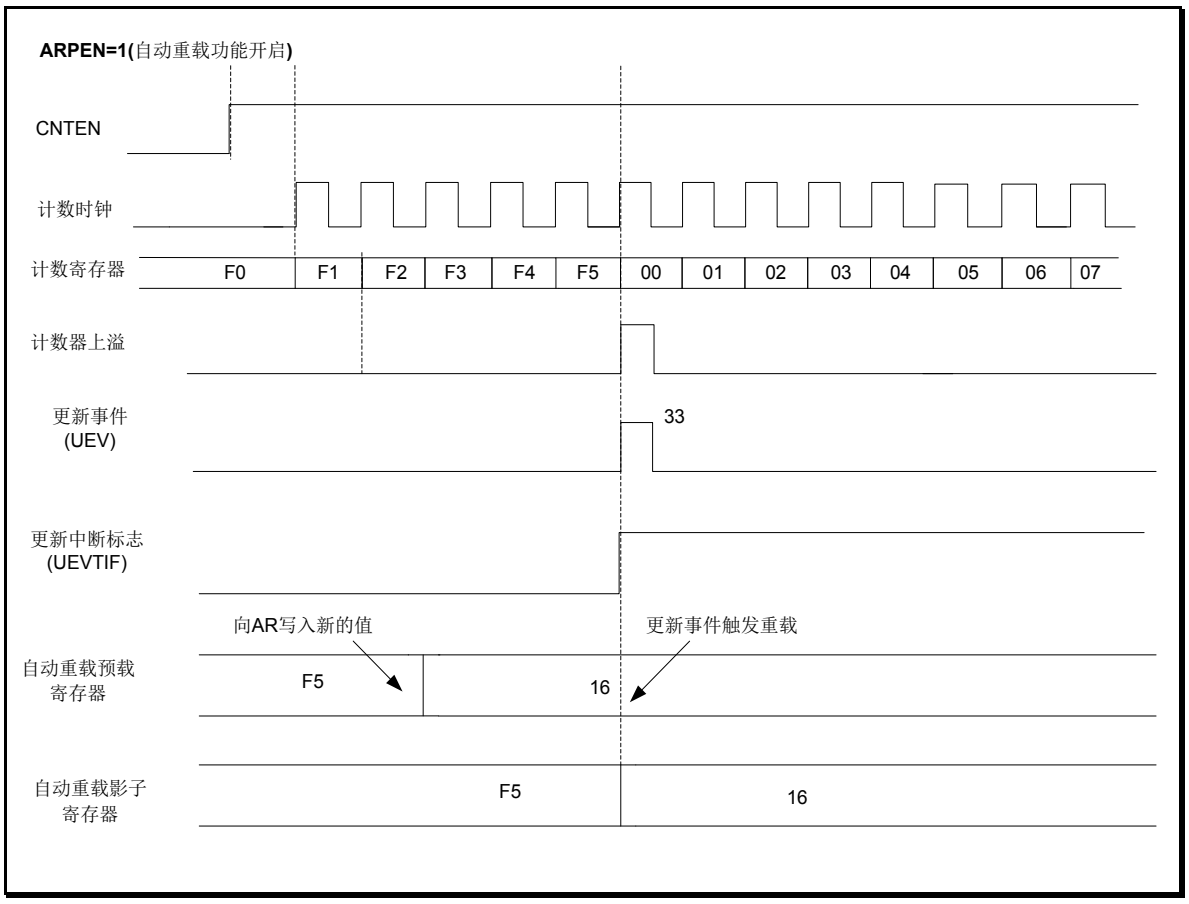


图 21-9　当 ARPEN=1 时计数器时序图

21. 4. 3. 2　　递减计数模式

在递增模式下，当 GP16C4Tn_REPAR 寄存器值为 0 时，计数器从 GP16C4Tn_AR 寄存器值开始递减至 0；然后重复递减并产生更新事件（UEV）。当 GP16C4Tn_REPAR 寄存器不为 0 时，则在 GP16C4Tn_REPAR+1 次后产生更新事件。

置位 GP16C4Tn_SGE 寄存器中的 SGU 位（通过软件或使用从机模式控制器）同样会产生更新事件。

当有更新事件（UEV）产生时，预载寄存器值会更新到影子寄存器，更新标志位（GP16C4Tn_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）。

下图为 GP16C4Tn_REPAR=0x0，GP16C4Tn_AR = 0x27，预分频设为 1 分频时的计数器时序。

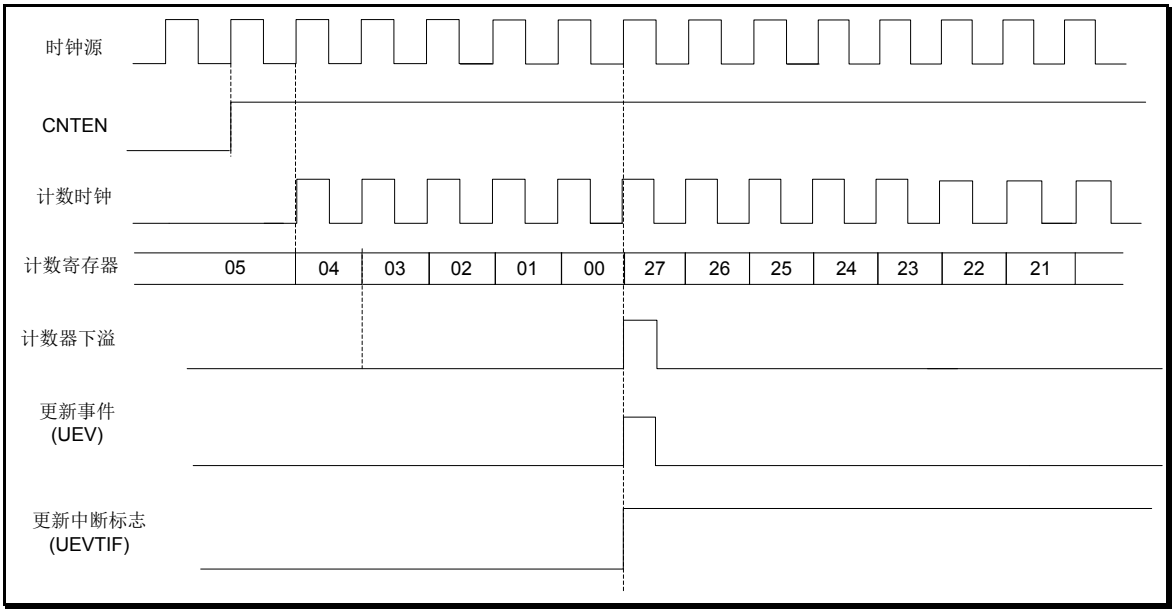


图 21-10　定时器递减计数时序图

21. 4. 3. 3　　中心对齐模式

当 GP16C4Tn_CON1 寄存器的 CMSEL 位的值不等于"00"时，定时器工作在中心对齐模式。定时器配置为中心对齐模式时，计数器先从 0 开始递增至 GP16C4Tn_AR 寄存器值-1，并产生更新事件（UEV）；接着计数器从 GP16C4Tn_AR 寄存器值递减至 1，并产生下溢事件。如此循环计数。在计数器递减计数（中心对称模式 1，CMSEL="01"）、计数器递增计数（中心对称模式 2，CMSEL="10"）、计数器递增和递减计数（中心对称模式 3，CMSEL="11"）模式下，当通道配置为输出模式时，其输出比较中断标志位会置位。

在中心对齐模式下，GP16C4Tn_CON1 寄存器的 DIRSEL 位无法进行写操作，该位由硬件自动更新指示当前计数方向。

计数上溢、下溢或者置位 GP16C4Tn_EGR 寄存器的 SGU 位（通过软件或使用从模式控制器）都会产生更新事件。因此，计数器和预分频器都会从 0 开始计数。

软件置位 GP16C4Tn_CON1 寄存器中的 DISUE 位可关闭更新事件（UEV）的产生。更新事件（UEV）关闭时，可避免向预载寄存器写新值时更新影子寄存器。DISUE 复位之前都不会产生更新事件。而在正常产生更新事件时，计数器仍然从 0 开始，同样预分频计数也是从 0 开始（但预分频值没有改变）。此外，若置位 GP16C4Tn_CON1 寄存器中的 UERSEL 位（更新请求选择），置位 SGU 位时会产生一次更新事件（UEV），但 UEVTIF 标志位不会置位（因此，不会触发中断或 DMA 请求）。这就避免了在捕获事件时，清除计数器值时产生更新和捕获中断。

当有更新事件（UEV）产生时，预载寄存器值会更新到影子寄存器，更新标志位（GP16C4Tn_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）。

注：若更新源为计数上溢，自动重载会在计数器重载前更新。因此，下一周期即为预期值（计数器载入新值）。

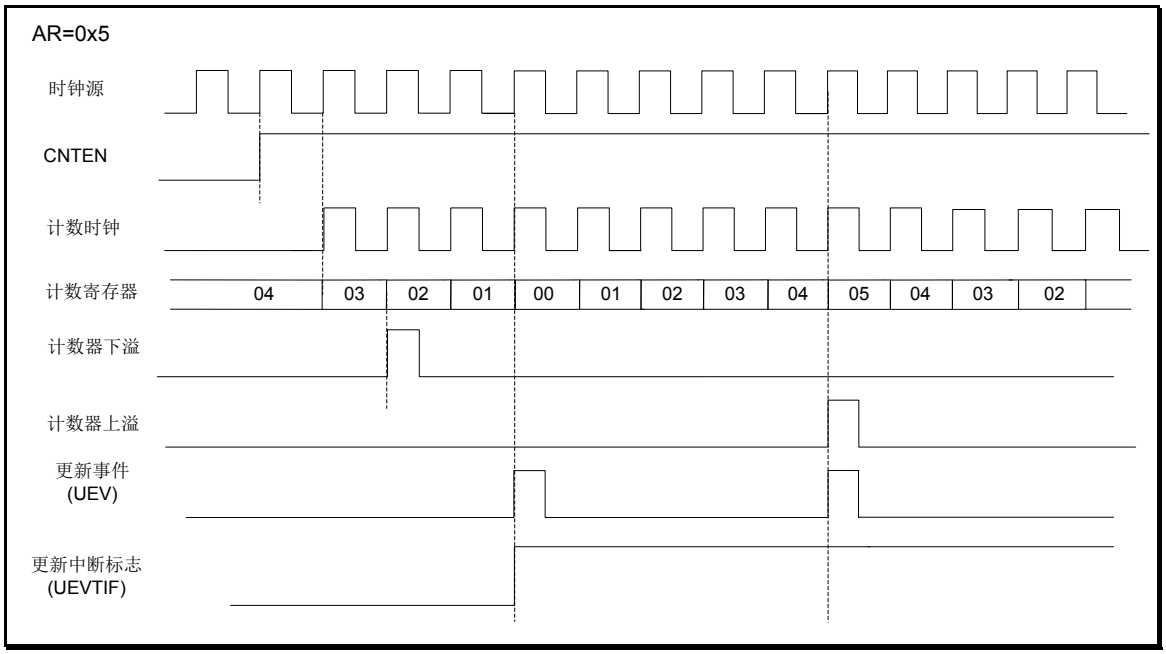


图 21-11　增减计数器时序图

21. 4. 4　捕获/比较通道

以下各图为捕获/比较通道的概述。

输入电路对 In 输入端的信号进行采样，产生一个经过滤波的信号 InF。之后，一个可极性选择的边沿检测器产生 In 边沿检出信号，该信号可作为从模式控制器的触发输入或作为捕获控制命令，且信号经过分频后进入捕获寄存器。

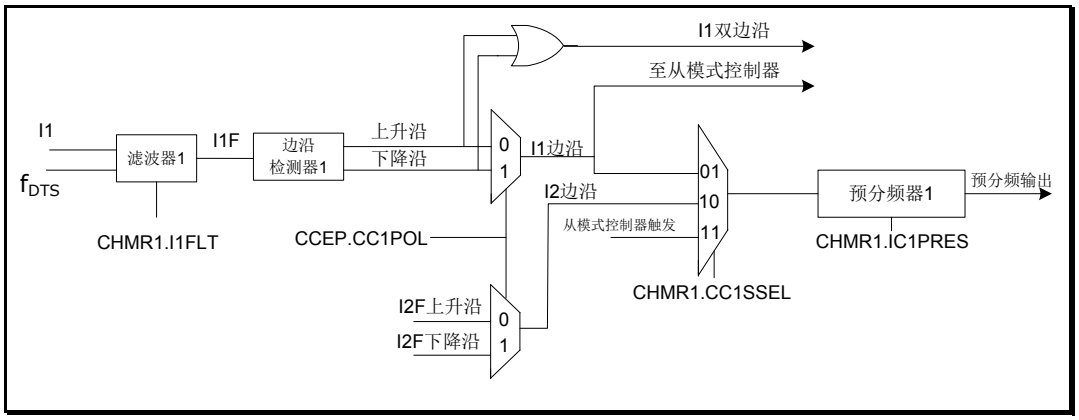


图 21-12　捕获/比较通道

输出部分产生一个中间波形（高有效）作为基准，在输出末端决定最终输出信号的极性。

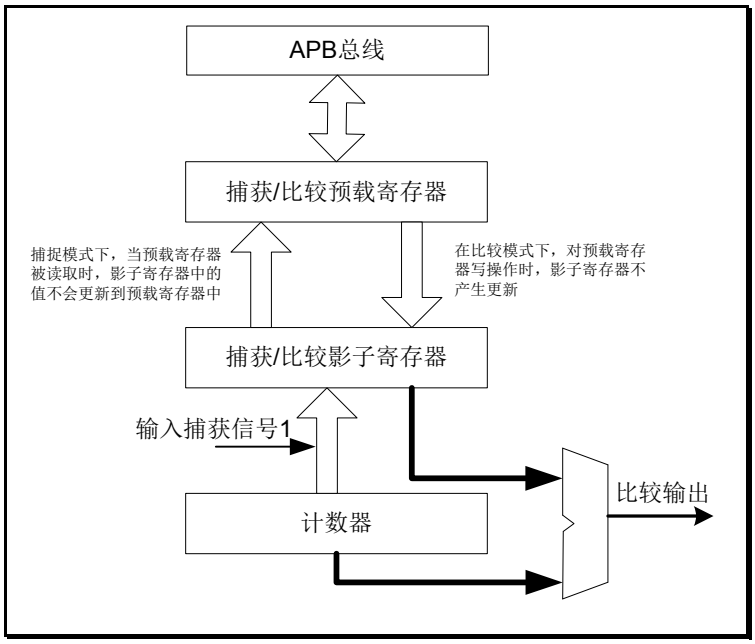


图 21-13　捕获/比较信道 1 主电路

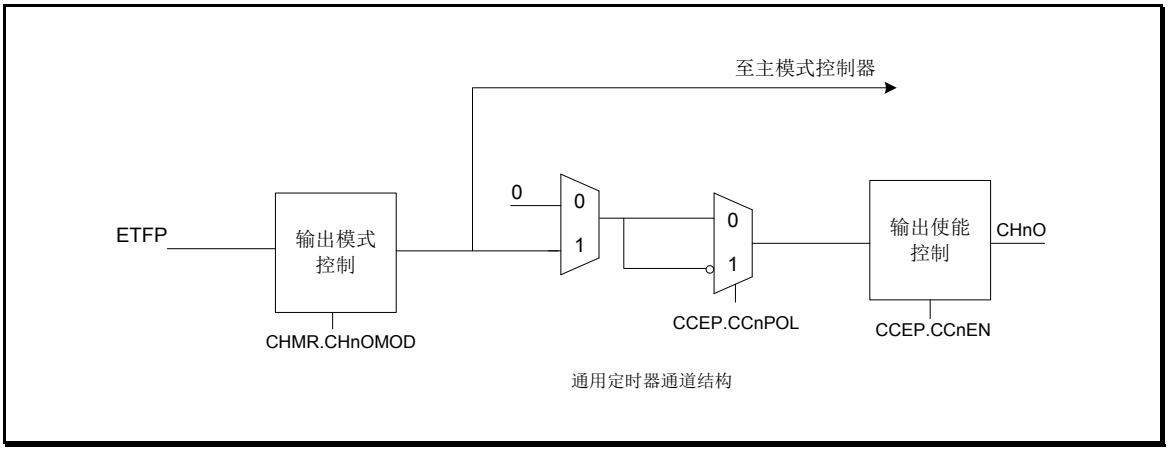


图 21-14　捕获/比较通道的输出阶段

21.4.5 输入捕获模式

在输入捕获模式下，当检测到 In 上相应信号变化时，计数器的值就会被锁存到捕获/比较寄存器（GP16C4Tn_CCVALn）寄存器中。当捕获发生时，相应的 CHnCCIF 标志位（GP16C4Tn_RIF）会置位，同时会触发中断或 DMA（如果使能）请求。若发生捕获时，CHnCCIF 标志位已经置位，则过捕获 CHnOVIF 标志位（GP16C4Tn_RIF）置位。软件写'0'或读取 GP16C4Tn_CCVALn 寄存器中的捕获值都可以复位 CHnCCIF 标志位。对 CHnOVIF 位写'0'可清空该标志位。

以下为以 I1 输入上升沿作为捕获输入时的流程：

7. 选择有效输入端：GP16C4Tn_CCVAL1 必须连接到 I1 输入端，因此需将 GP16C4Tn_CHMR1 寄存器中的 CC1SSEL 位写"01"。只要 CC1SSEL 不为"00"，通道被配置为输入且 GP16C4Tn_CCVAL1 寄存器为只读。
8. 根据定时器连接的输入信号，配置输入滤波器的持续时间。当输入信号翻转时，前 5 个内部时钟信号是不稳定的，因此必须配置滤波器的时间大于 5 个时钟周期。当 I1 检测到新的电平，连续 8 次采样可确认电平变化有效。
9. 选择 I1 信道的有效边沿变换。GP16C4Tn_CCEP 寄存器中的 CC1POL 写'0'（上升沿）。
10. 配置输入预分频器。
11. 置位 GP16C4Tn_CCEP 寄存器中的 CC1EN 位，使能捕获计数器的值到捕获寄存器。
12. 如有需要，置位 GP16C4Tn_IER 寄存器中的 CC1IT 位，使能中断请求。置位 GP16C4Tn_DMAEN 寄存器中的 CC1DMA 位，使能 DMA 请求。

当发生输入捕获时：

5. 有效边沿产生，GP16C4Tn_CCVAL1 寄存器获取计数器的值。
6. CH1CCIF 标志位置位（中断标志）。若至少 2 个连续的捕获发生，但标志位没有及时清除，则 CH1OVIF 也会置位。
7. 中断的产生取决于 GP16C4Tn_IVS 寄存器的 CC1IT 位。
8. DMA 请求的产生取决于 CC1DMA。

为了处理捕获溢出，建议在读出捕获溢出标志位之前先读取捕获数据。这可以避免丢失在读出捕获标志位之后与读取数据之前可能重复产生的捕获信息。

注：捕获中断请求可由软件设置 GP16C4Tn_SGE 寄存器中 SGCCnE 位产生。

21. 4. 5. 1 PWM输入模式

测量 I1 上 PWM 信号的周期和占空比的过程如下：

1. 为 GP16C4Tn_CCVAL1 选择有效的输入：GP16C4Tn_CHMR1 寄存器中的 CC1SSEL 位写"01"（I1 被选择）。

2. 为 I1 边沿检出选择有效的极性（用于捕获数据到 GP16C4Tn_CCVAL1 寄存器和计数器清零）：CC1POL 位写'0'（上升沿有效）。

3. 为 GP16C4Tn_CCVAL2 选择有效输入：GP16C4Tn_CHMR1 寄存器的 CC2SSEL 位写"10"（I1 被选择）。

4. 为 I1 边沿检出选择有效极性（用于捕获数据到 GP16C4Tn_CCVAL2）：CC2POL 位写'1'（下降沿有效）。

5. 选择有效的触发输入：GP16C4Tn_SMCON 寄存器的 TSSEL 位写"101"（I1 边沿检出被选择）。

6. 配置从机模式控制器为复位模式：GP16C4Tn_SMCON 寄存器的 SMODS 位写"100"。

7. 使能捕获：GP16C4Tn_CCEP 寄存器的 CC1EN 位和 CC2EN 位写'1'。

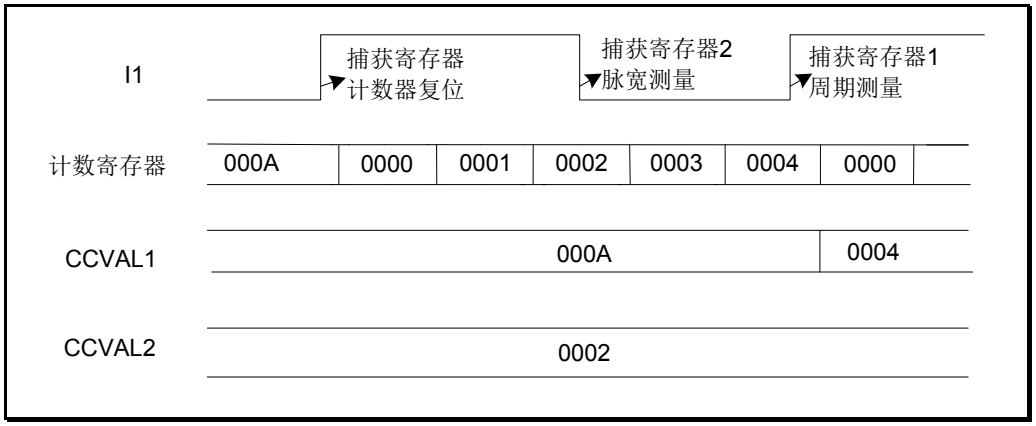


图 21-15 PWM 输入模式时序

21. 4. 6 PWM模式

脉宽调制模式可以产生一个 GP16C4Tn_AR 寄存器值确定频率，GP16C4Tn_CCVALn 寄存器值确定占空比的信号。

每个通道的 PWM 模式是相互独立的（每个 CHnO 输出一个 PWM），GP16C4Tn_CHMRn 寄存器的 CHnOMOD 位写"110"（PWM 模式 1）或写"111"（PWM 模式 2）。必须通过置位 GP16C4Tn_CHMRn 寄存器的 CHnOPREN 位来使能相应的预载寄存器，最后还需置位 GP16C4Tn_CON1 寄存器的 ARPEN 位来使能自动重装预载功能。

只有当更新事件发生时预载寄存器中的值才会传到影子寄存器，因此，在使能计数前，必须通过置位 GP16C4Tn_SGE 寄存器的 SGU 位来初始化所有的寄存器。

CHnO 的极性可通过 GP16C4Tn_CCEP 寄存器的 CCnPOL 位配置，有效极性可配置为高或低。CHnO 的输出使能由 CCnEN 位（GP16C4Tn_CCEP 寄存器）控制。

在 PWM 模式（1 或 2）中，GP16C4Tn_COUNT 和 GP16C4Tn_CCVALn 寄存器的值会持续的比较，确定 GP16C4Tn_CCVALn<= GP16C4Tn_COUNT 或 GP16C4Tn_CCVALn>= GP16C4Tn_COUNT（取决于计数器的计数方向）。

定时器产生 PWM 波形是边沿对齐或中心对齐，取决于 GP16C4Tn_CON1 寄存器的 CMSEL 位。

21. 4. 6. 1 PWM边沿对齐模式

1. 递增计数配置

当 GP16C4Tn_CON1 寄存器的 DIRSEL 位为低时，计数器递增计数。

下图给出了 GP16C4Tn_AR = 8 时的边沿对齐 PWM 波形。

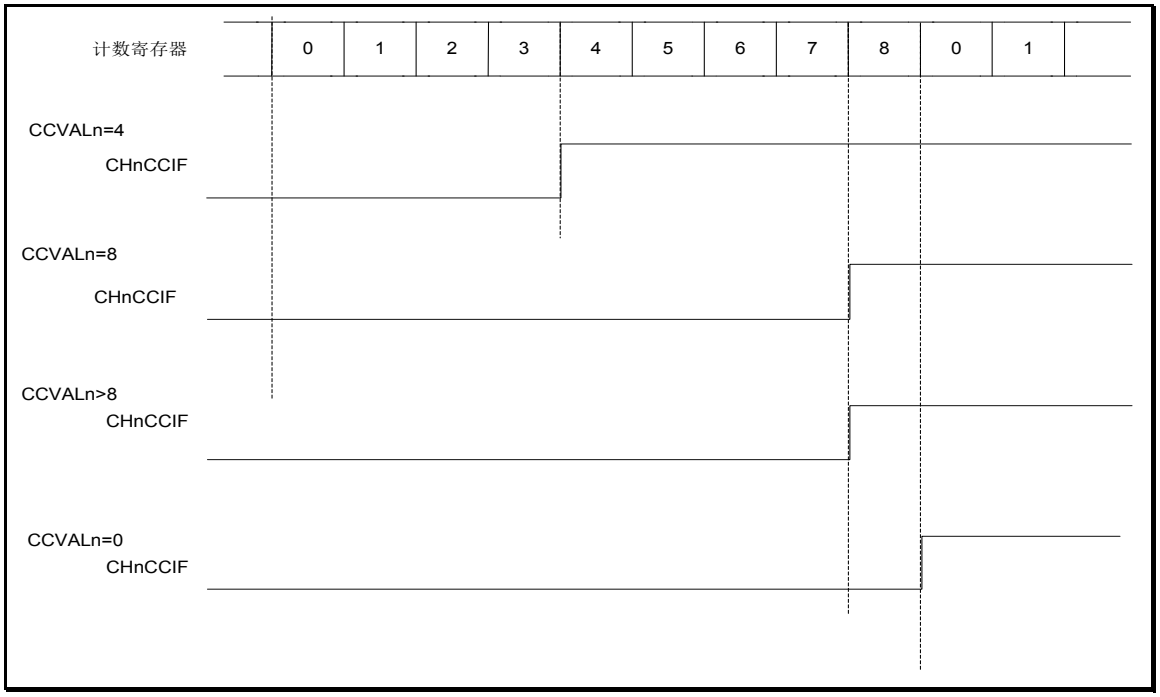


图 21-16 边沿对齐 PWM 波形（AR=8）

2. 递减计数配置

当 GP16C4Tn_CON1 寄存器的 DIRSEL 位为高时，计数器递减计数。

21. 4. 6. 2 PWM中心对齐模式

当 GP16C4Tn_CON1 寄存器中的 CMSEL 位不为"00"时，中心对齐模式有效。计数器是递增、递减计数分别置比较标志位或递增递减都置比较标志位，取决于 CMSEL 位的配置。GP16C4Tn_CON1 寄存器的方向位（DIRSEL）是由硬件更新的，软件无法修改。

下图为中心对齐方式产生的 PWM 波形的例子：

◇ GP16C4Tn_AR=0x3F，GP16C4Tn_CCVALn=0x3D

◇ PWM 模式 1

- GP16C4Tn_CON1 寄存器的 CMSEL= "01"，在中心对齐模式 1 下，计数器向下计数时会置位比较标志位。

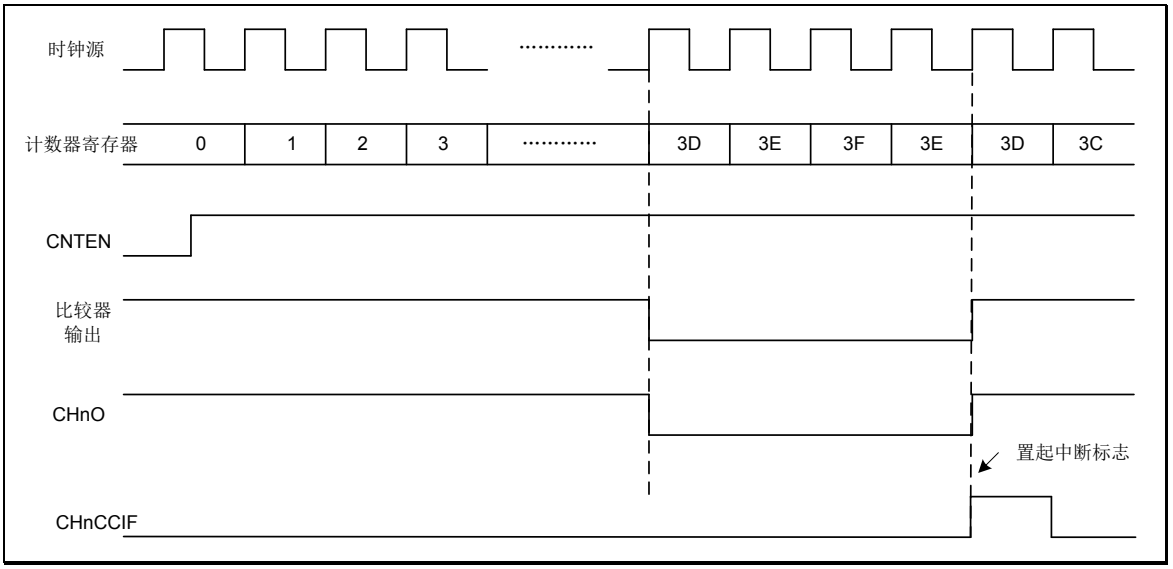


图 21-17 边沿对齐 PWM 波形（AR=0x3F）

中心对齐模式的使用技巧：

◇ 当进入中心对齐模式后，当前递增或递减配置生效。计数器递增或递减计数取决于 GP16C4Tn_CON1 寄存器的 DIRSEL 位的值。

◇ 计数器在中心对齐模式下运行时，对计数器写操作可能导致不可预知的结果。特别是：

- 若向计数器入的值大于自动重载值（GP16C4Tn_COUNT>GP16C4Tn_AR），计数方向不更新。例如，如果计数器递增计数，写入值后仍旧递增计数。
- 若向计数器写 0 或 GP16C4Tn_AR 中的重载值，则计数方向更新，但并没有产生 UEV。

◇ 使用中心对齐模式最安全的方式是计数器开始计数前通过软件产生更新事件（置位 GP16C4Tn_SGE 寄存器中的 SGU 位）且在计数器运行过程中不对计数器写值。

21.4.7 输出比较模式

该功能用于控制输出波形或指示周期时间的结束。

当捕获/比较寄存器和计数器值匹配时，输出比较功能：

- ◇ 输出比较模式（GP16C4Tn_CHMRn 寄存器中的 CHnOMOD 位）和输出极性（GP16C4Tn_CCEP 寄存器中的 CCnPOL 位）的配置值输出到对应的引脚上。
- ◇ 中断状态寄存器中的标志位置位（GP16C4Tn_RIF 寄存器的 CHnCCIF 位）。
- ◇ 若相应的中断掩码置位，则产生中断（GP16C4Tn_IER 寄存器的 CCnIT 位）。
- ◇ 若相应的使能位置位（GP16C4Tn_DMAEN 寄存器的 CCnDMA 位，GP16C4Tn_CON2 寄存器的 CCDMASEL 位用于 DMA 请求的选择），则发送 DMA 请求。

GP16C4Tn_CHMRn 寄存器中 CHnOPREN 位的值可决定 GP16C4Tn_CCVALn 寄存器是否带有预装载寄存器。

在输出比较模式中，更新事件 UEV 对 CHnO 的输出没有影响。计时分辨率为计数器的一次计数。输出比较模式同样可以用来输出单个脉冲（单脉冲模式）。

输出比较的配置过程：

1. 选定计数器时钟（内部，外部，预分频）。
2. GP16C4Tn_AR 与 GP16C4Tn_CCVALn 寄存器中写入预期值。
3. 若需要产生中断请求，置位 GP16C4Tn_IER 寄存器中的 CCnIT 位。
4. 选择输出模式，例如：

   - CHnOMOD = "011"，当 CNTV 与 CCRVn 匹配时，CHnO 输出翻转。
   - CHnOPREN = '0'，关闭预载寄存器。
   - CCnPOL = '0'，选择有效极性为高。
   - CCnEN = '1'，使能输出。
5. GP16C4Tn_CON1 寄存器中的 CNTEN 位置位，使能计数器。

通过配置 GP16C4Tn_CHMR1 寄存器的 CHnOPREN 位可将 GP16C4Tn_CCVALn 配置为是否带预装载寄存器。通过软件方式，GP16C4Tn_CCVALn 寄存器的值可随时更新控制输出波形。

21.4.7.1 外部事件清除比较输出

ETFP 输入端（GP16C4Tn_CHMRn 寄存器的 CHnOCLREN 位写'1'）上的高电平，可将给定通道的比较输出信号拉低。在下次更新事件（UEV）发生前，比较输出会一直保持为低。该功能只能应用在输出比较和 PWM 模式中，强制输出模式中不起作用。

ET 信号可以接到电流控制比较器的输出端。该例中，ET 须按如下流程配置：

1. 外部触发预分频器应该关闭：GP16C4Tn_SMCON 寄存器的 ETPSEL[1：0]位应该写"00"

2. 外部时钟源 2 关闭：GP16C4Tn_SMCON 寄存器的 ECM2EN 位写'0'

3. 外部触发极性（ETPOL）和外部触发滤波器（ETFLT）可根据用户需要配置

21. 4. 8 单脉冲模式

单脉冲模式下，响应某个触发后，定时器的输出通道在可配置的延迟时间后产生一个脉冲，脉冲长度可配。从模式控制器可控制计数器的启动。脉冲波形可在输出比较模式和 PWM 模式下产生。置位 GP16C4Tn_CON1 寄存器的 SPMEN 位可选择单脉冲模式。计数器会在下次更新事件 UEV 产生时自动停止。

只有比较值不同于计数器初始值时，单脉冲才可以正确的产生。计数器开始计数前（定时器等待触发），必须如下配置：

- ◇ 递增计数：CNT < CCVALn≤ AR（特别地，0 < CCVALn）
- ◇ 递减计数：CNT > CCVALn

基于 PWM 模式设置单脉冲输出波形的步骤如下：

- ◇ 设置 GP16C4Tn_CHMRn 寄存器的 CHnOMOD 位，选择 PWM 模式 1 或 2；
- ◇ 设置 GP16C4Tn_CCEP 寄存器的 CCnPOL 位，选择通道端口 CHnO 的输出极性；
- ◇ 设置 GP16C4Tn_CON1 寄存器的 DIRSEL，CMSEL，SPMEN 位，配置为递增或递减计数，PWM 普通波形模式，单脉冲模式使能；
- ◇ 设置 GP16C4Tn_CHMR1 寄存器的 CH1OPREN =1，GP16C4Tn_CON1 寄存器的 ARPEN =1，使能比较寄存器和计数重载寄存器的缓冲功能（也可以根据实际情况不使能缓冲）；
- ◇ 设置 GP16C4Tn_CCVALn 寄存器和 GP16C4Tn_AR 寄存器，配置单脉冲输出延时和脉宽时间；
- ◇ 设置 GP16C4Tn_SGE 寄存器的 SGU=1 来产生一个更新事件；
- ◇ 设置 GP16C4Tn_CON1 寄存器的 CNTEN=1 来启动计数器，也可以在触发模式下，通过外部触发输入信号来触发硬件自动设置 CNTEN=1。

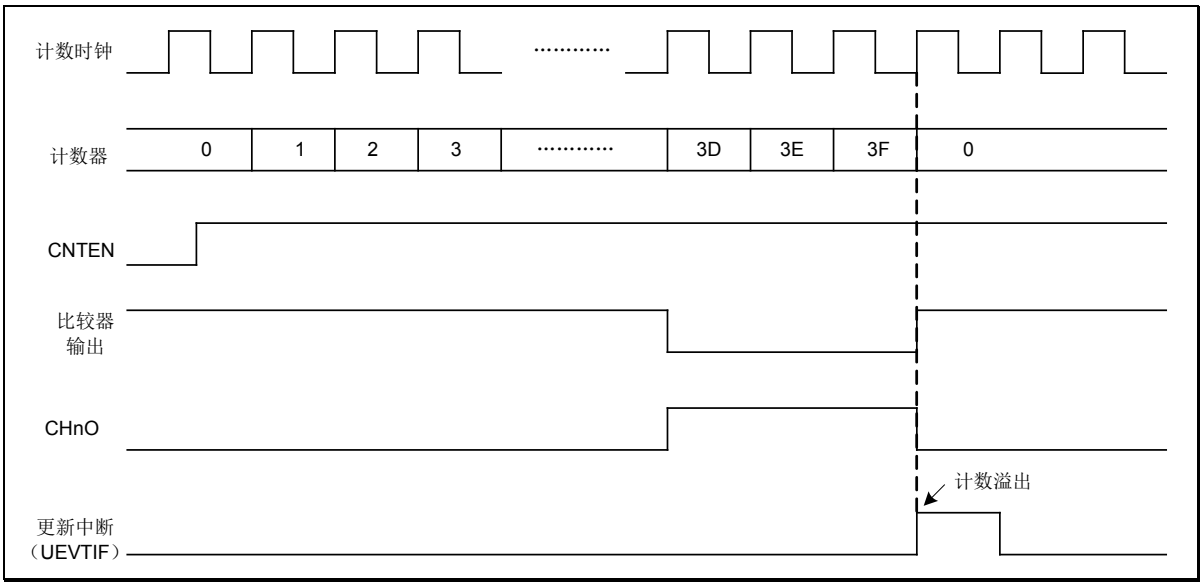


图 21-18 单脉冲模式

21.4.9 编码器接口模式

编码器接口模式的三种配置：若计数器只根据 I2 上的边沿计数，则 GP16C4Tn_SMCON 寄存器中的 SMODS = "001"；若计数器只根据 I1 上的边沿计数，则 GP16C4Tn_SMCON 寄存器中的 SMODS = "010"；若计数器同时根据 I1 和 I2 上的边沿计数，则 GP16C4Tn_SMCON 寄存器中的 SMODS = "011"。

配置 GP16C4Tn_CCEP 寄存器中的 CC1POL 和 CC2POL 位的值可选择 I1 和 I2 的极性。如果需要，也可以配置输入滤波器。

CH1_IN 和 CH2_IN 端口作为增量编码器的接口。当计数器使能时，计数器根据 I1 或 I2 上滤波后的有效电平变化时钟计数。I1 和 I2 滤波后的有效信号顺序会产生计数脉冲及方向信号。计数器是递增或递减计数由信号的跳变顺序决定，GP16C4Tn_CON1 寄存器中的 DIRSEL 计数方向位由自动硬件更新。

编码器接口模式的工作方式类似于一个带有方向选择的外部时钟。计数器在 0 到 GP16C4Tn_AR 寄存器中的自动重载值之间连续计数。因此，必须在开始计数前配置 GP16C4Tn_AR 寄存器。同样的，捕获器、预分频器、重复计数器、触发输出的特性正常工作。设定编码模式和选择外部时钟源 2 不兼容，不可以同时选择。

该模式下，计数器会根据增量式编码器的速度和方向自动修改，计数器的值反应的是编码器的位置。计数方向对应着连接传感器的旋转方向。

下表列出了所有的可能组合，假设 I1 和 I2 不同时变换。

有效边沿	有效边沿相对信号的电平（I1 滤波信号对应 I2,I2 滤波信号对应 I1）	I1 滤波信号边沿		I2 滤波信号边沿	
		上升	下降	上升	下降
仅在 I1 计数	高	下降	上升	不计数	不计数
	低	上升	下降	不计数	不计数
仅在 I2 计数	高	不计数	不计数	上升	下降
	低	不计数	不计数	下降	上升
在 I1 和 I2 上计数	高	下降	上升	上升	下降
	低	上升	下降	下降	上升

表 21-1 计数方向与编码器信号的关系

外部增量编码器可直接与 MCU 连接，无需外部逻辑接口逻辑。而比较器通常用于将编码器的差分输出转换为数字信号，这极大地增加了抗噪声能力。编码器的第三个输出端用于指示机械零点，可以连接到外部中断输入引脚以触发一次计数复位。

下图给出了计数器操作的例子，给出了计数信号了产生和方向控制。同样给出了选择双边沿时，输入抖动如何被补偿。输入抖动可能发生在传感器靠近切换点处。

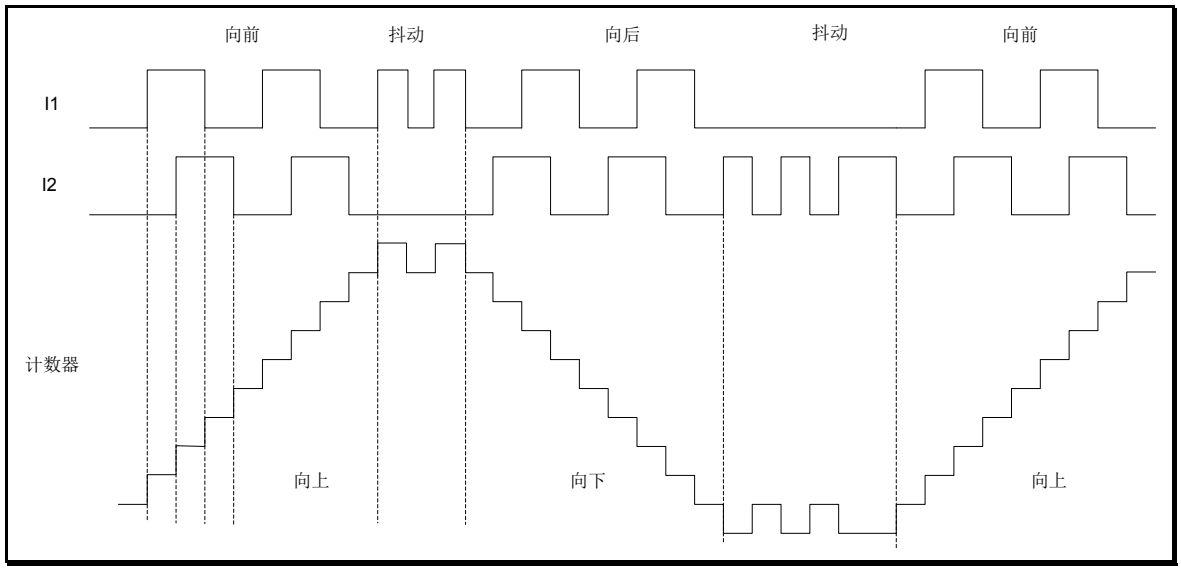


图 21-19　编码器接口模式下的计数操作

下图给出了计数器在 I1 滤波信号极性反相时的计数过程（除了 CC1POL = '1'，其他配置与上面一致）

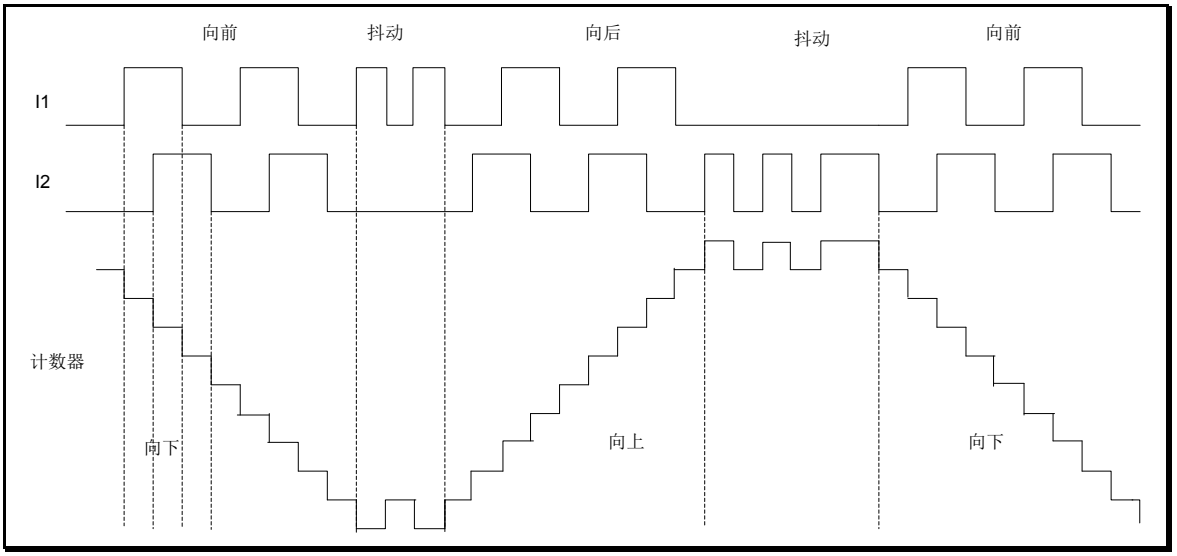


图 21-20　滤波后极性反相时编码器接口

当配置为编码器接口模式时，定时器可提供传感器的当前位置信息。配置一个额外定时器为捕获模式，用于测量两个编码器事件的间隔，根据间隔时长获取动态信息（速度、加速度、减速度）。编码器用于指示机械零点的输出就是此用处。根据编码器两个事件间隔，可以周期性的读取计数器的值。如果允许，可以将计数器值锁存到第三个输入捕获寄存器（捕获信号必须是周期性的且可由其它定时器产生）。条件允许时，可通过实时时钟产生 DMA 请求的方式读取计数器值。

21. 4. 10 输入异或功能

通过 GP16C4Tn_CON2 寄存器中 I1FSEL 位，可将通道 1 的输入滤波器连接到 XOR 门的输出端，XOR 门联合了 CH1_IN、CH2_IN 和 CH3_IN 三个输入引脚。

XOR 输出用于定时器的所有输入功能，如触发或输入捕获。

21. 4. 11 定时器和外部触发的同步

GP16C4Tn 定时器可在多种模式下与外部触发同步：复位模式、门控模式及触发模式。

21. 4. 11. 1 复位模式

计数器及其预分频器可以在响应触发输入事件时重新初始化。此外，若 GP16C4Tn_CON1 寄存器的 UERSEL 位为低时会产生一次更新事件 UEV。所有预载寄存器（GP16C4Tn_AR，GP16C4Tn_CCVALn）都会因更新事件 UEV 而被更新。

在下面例子中，I1 输入端的上升沿让递增计数被清空：

◇ 配置通道 1 上检测 I1 上的上升沿。配置输入滤波周期（本例无需滤波器，故 I1FLT = "0000"）。触发捕获分频器没有使用，无需配置。CC1SSEL 位只选择输入捕获源，GP16C4Tn_CHMR1 寄存器中 CC1SSEL = "01"。GP16C4Tn_CCEP 寄存器中 CC1POL = 0 以确定极性（只检测上升沿）。

◇ 定时器配置为复位模式：GP16C4Tn_SMCON 寄存器中 SMODS = "100"。选择 I1 作为输入源：GP16C4Tn_SMCON 寄存器中 TSSEL = "101"。

◇ 启动计数器：GP16C4Tn_CON1 寄存器中 CNTEN = '1'。

计数器依据内部时钟开始计数，正常计数直到 I1 上出现上升沿。当 I1 上出现上升沿时，计数器会被清零且从 0 重新开始计数。同时，标志位置位（GP16C4Tn_RIF 寄存器中 TRGIF 位），如果中断及 DMA 使能（取决于 GP16C4Tn_IER 寄存器中的 TRGIT 和 GP16C4Tn_DMAEN 的 TRGDMA 位），会发送中断及 DMA 请求。

下图给出了当自动重载寄存器 GP16C4Tn_AR = 0x36 时的信号变化。由于 I1 输入的再同步电路，I1 上的上升沿和计数器实际复位之间会存在延时（包含 2~3 个模块时钟周期的同步延时）。

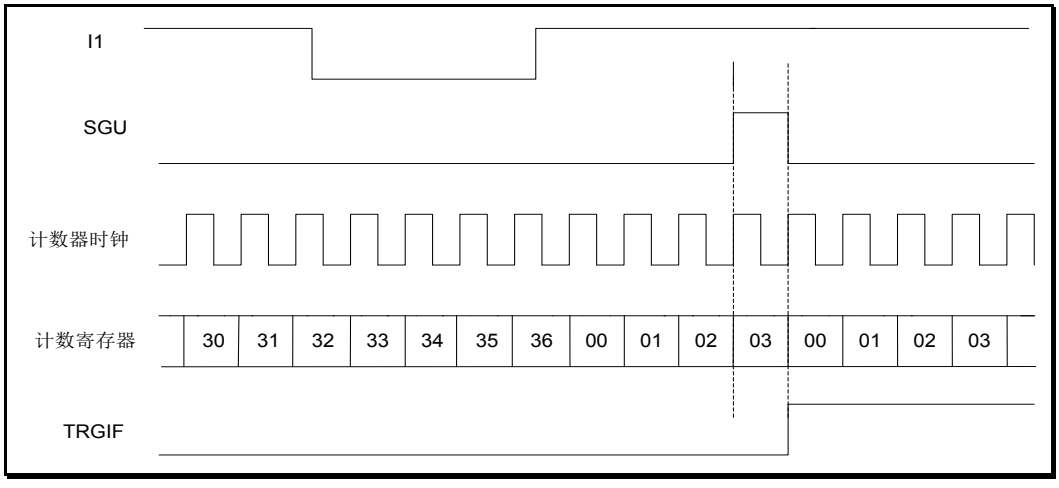


图 21-21 复位模式控制电路

21. 4. 11. 2 门控模式

计数器根据选中的输入电平被使能。

下面的例子中，计数器只在 I1 输入为低电平时才递增计数：

◇ 配置通道 1 在 I1 上检测低电平。配置输入滤波周期（本例不需要滤波器，I1FLT = "0000"）。触发捕获分频器没有使用，无需配置。GP16C4Tn_CHMR1 寄存器中的 CC1SSEL = "01"，选择输入捕获源。GP16C4Tn_CCEP 寄存器中 CC1POL = '1'，确认极性（只检测低电平）。

◇ 配置定时器为门控模式：GP16C4Tn_SMCON 寄存器中 SMODS = "101"。选择 I1 作为输入源：GP16C4Tn_SMCON 寄存器中 TSSEL = "101"。

◇ 使能计数器：GP16C4Tn_CON1 寄存器中 CNTEN = '1'（门控模式中，如果 CNTEN = '0'，无论触发输入为何电平，计数器都不会启动）。

只要 I1 为低电平，计数器依据内部时钟开始计数，一旦 I1 为高则停止计数。由于 I1 输入端再同步电路的原因，I1 上出现上升沿和计数器实际停止之间会有一定的延时。

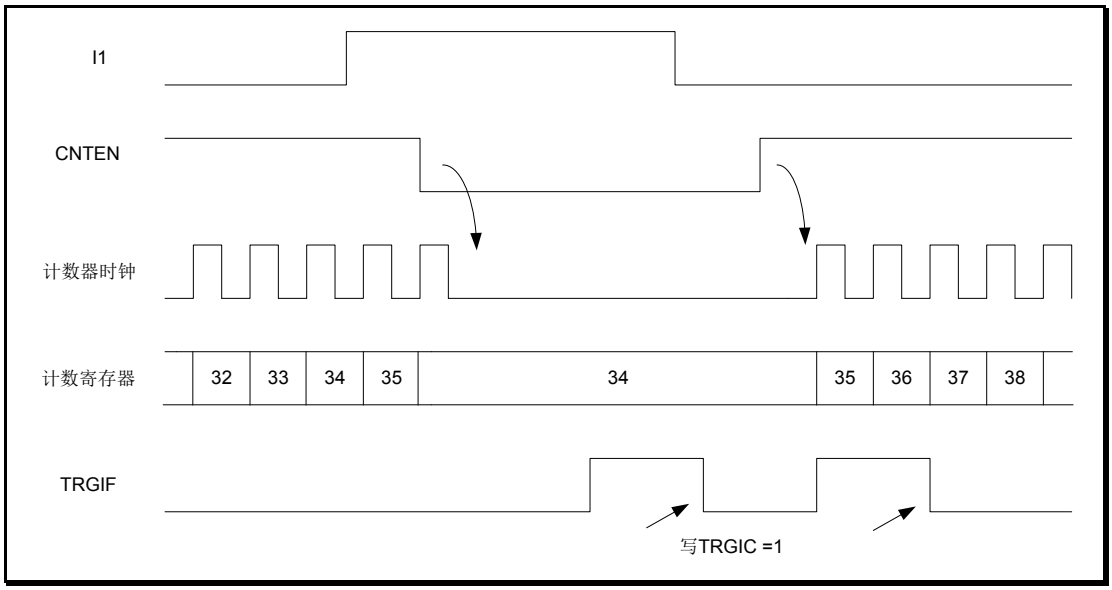


图 21-22 门控模式控制电路

21. 4. 11. 3 触发模式

输入端选中的事件可以使能计数器。

下面的例子中，I2 输入端上的上升沿可以启动递增计数：

◇ 配置通道 2 可以检测 I2 上的上升沿。配置滤波时间（本例不需要滤波，I2FLT = "0000"）。触发捕获分频器没有使用，无需配置。GP16C4Tn_CHMR1 寄存器中 CC2SSEL = "01",用于选择捕获源。GP16C4Tn_CCEP 寄存器中 CC2POL = '1'，确认极性（只检测低电平）。

◇ 配置定时器为触发模式：GP16C4Tn_SMCON 寄存器中 SMODS = "110"。GP16C4Tn_SMCON 寄存器中 TSSEL = "110"，用于选择输入源。

I2 上出现上升沿时，计数器开始依据内部时钟计数并置位 TRGIF 标志位。

由于 I2 输入的再同步原因，I2 上出现上升沿和计数器实际停止之间会有一定的延时。

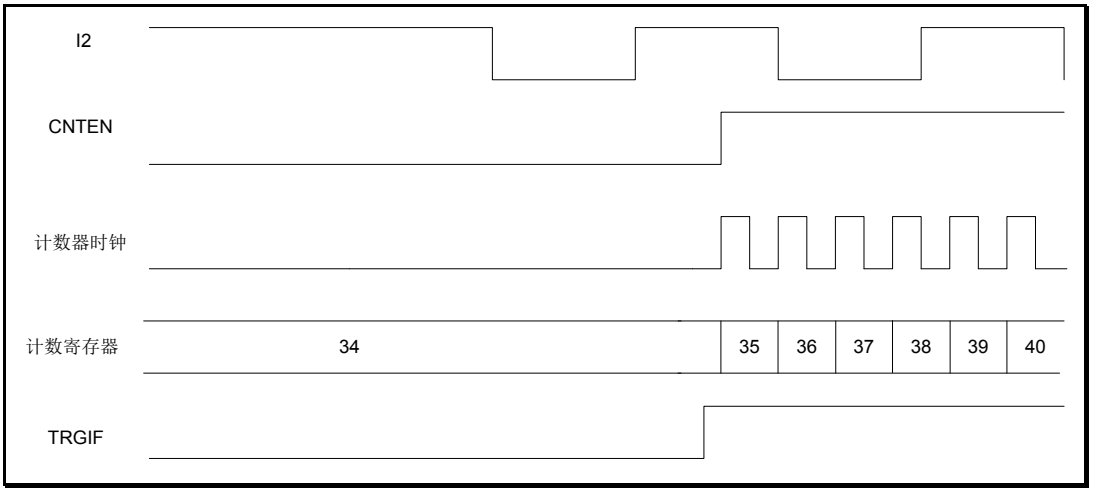


图 21-23　触发模式控制电路

21. 4. 11. 4　选择外部时钟源 2 的触发模式

外部时钟源 2 可和其他模式一起使用（除编码模式）。ET 信号可作为外部时钟输入，另一个输入可选择为触发输入（复位模式、门控模式或触发模式）。不推荐用 GP16C4Tn_SMCON 寄存器的 TSSEL 位选择 ET 作为 TI。

下面的例子中，一旦 I1 上出现上升沿时，计数器会依据 ET 信号的每个上升沿递增计数。

◇　通过 GP16C4Tn_SMCON 寄存器，配置外部触发输入电路，过程如下：

ETFLT = "000"：无滤波

ETPSEL = "00"：禁止分频

ETPOL = '0'：检测 ET 的上升沿，ECM2EN = '1'使能外部时钟模式 2

◇　配置通道 1 检测 I 的上升沿，过程如下：

I1FLT = "0000"：无滤波。

触发捕获分频器没有使用，无需配置。

GP16C4Tn_CHMR1 寄存器中 CC1SSEL = "01"选择输入捕获源，
GP16C4Tn_CCEP 寄存器的 CC1POL = '0'确认极性（只检测上升沿）。

◇　配置定时器为触发模式：GP16C4Tn_SMCON 寄存器中 SMODS = "110"。
GP16C4Tn_SMCON 寄存器中 TSSEL = "101"选择 I1 作为输入源。

I1 上出现上升沿时，计数器使能且 TRGIF 标志位置位，然后计数器根据 ET 上的上升沿开始计数。

由于 ETFP 输入再同步电路的原因，ET 信号的上升沿和实际计数器的复位会有延时。

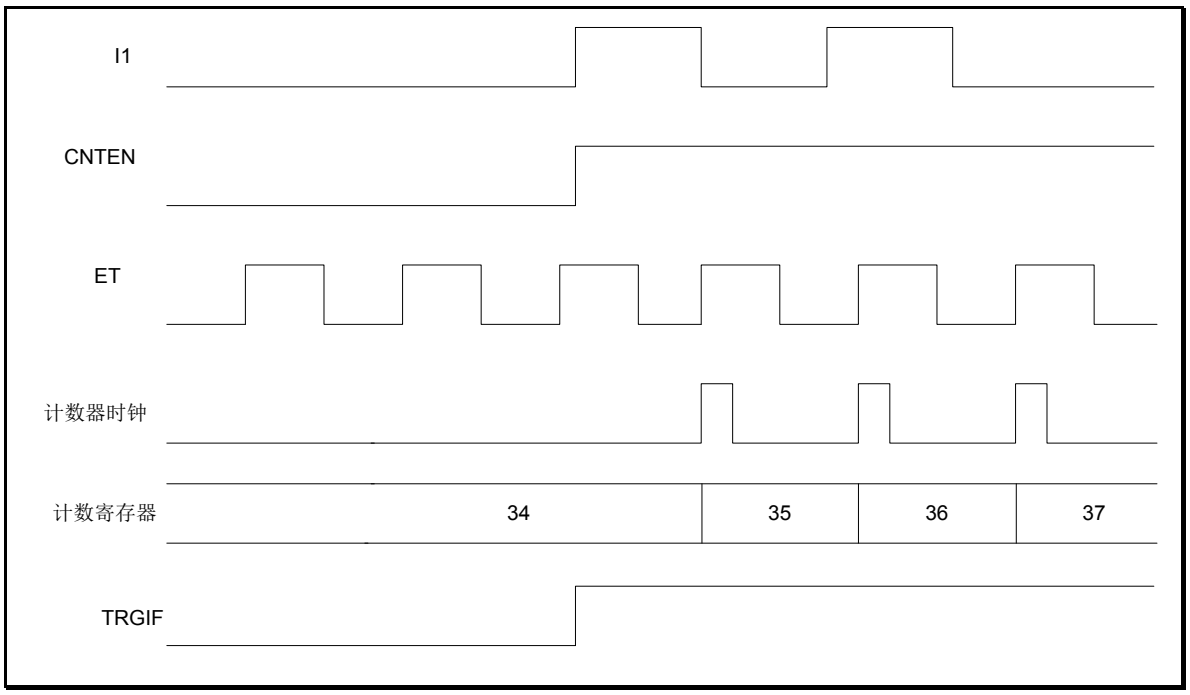


图 21-24　外部时钟源 2＋触发模式下的控制电路

21. 4. 12 调试模式

当微控制器进入调试模式（Cortex™-M3 内核停止），GP16C4Tn 计数器停止计数。

21. 5 特殊功能寄存器

21. 5. 1 寄存器列表

名称	偏移地址	描述
GP16C4Tn_CON1	000_H	控制寄存器 1
GP16C4Tn_CON2	004_H	控制寄存器 2
GP16C4Tn_SMCON	008_H	从模式控制寄存器
GP16C4Tn_IER	$00C_H$	中断使能寄存器
GP16C4Tn_IDR	010_H	中断禁止寄存器
GP16C4Tn_IVS	014_H	中断有效状态寄存器
GP16C4Tn_RIF	018_H	原始中断标志寄存器
GP16C4Tn_IFM	$01C_H$	中断标志屏蔽寄存器
GP16C4Tn_ICR	020_H	中断清零寄存器
GP16C4Tn_SGE	024_H	软件生成事件计算器
GP16C4Tn_CHMR1	028_H	捕获/比较模式寄存器 1
GP16C4Tn_CHMR2	$02C_H$	捕获/比较模式寄存器 2
GP16C4Tn_CCEP	030_H	捕获/比较使能寄存器
GP16C4Tn_COUNT	034_H	计数器寄存器
GP16C4Tn_PRES	038_H	预分频寄存器
GP16C4Tn_AR	$03C_H$	自动重载寄存器
GP16C4Tn_CCVAL1	044_H	捕获/比较寄存器 1
GP16C4Tn_CCVAL2	048_H	捕获/比较寄存器 2
GP16C4Tn_CCVAL3	$04C_H$	捕获/比较寄存器 3
GP16C4Tn_CCVAL4	050_H	捕获/比较寄存器 4
GP16C4Tn_DMAEN	058_H	DMA 使能寄存器

21. 5. 2 寄存器描述

21. 5. 2. 1 控制寄存器 1（GP16C4Tn_CON1）

控制寄存器 1（GP16C4Tn_CON1）

偏移地址：000_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						DFCKSEL		ARPEN	CMSEL		DIRSEL	SPMEN	UERSEL	DISUE	CNTEN

Reserved	Bit 31-10	-	**保留，必须保持为复位值**
DFCKSEL	Bit 9-8	R/W	**时钟分频** 该时钟分频为定时器（INT_CLK）频率与死区时间生成器和数字滤波器（ET，In）采用的死区时间和采样时钟（tDTS）之间的分频比。 00：$t_{DTS}=t_{INT\_CLK}$ 01：$t_{DTS}=2*tI_{NT\_CLK}$ 10：$t_{DTS}=4*t_{INT\_CLK}$ 11：保留
ARPEN	Bit 7	R/W	**自动重载预载使能** 0：GP16C4Tn_AR 寄存器未缓冲 1：GP16C4Tn_AR 寄存器被装入缓冲器
CMSEL	Bit 6-5	R/W	**中央对齐模式选择** 00：边沿对齐模式。计数器根据方向为（DIRSEL）来向上或向下计数。 01：**中央对齐模式 1。**计数器以交替方式向上或向下计数。仅当计数器向下计数时，配置为输出的通道（GP16C4T_CHMRn 寄存器中 CCnSSEL=00）的输出比较中断标志位才会被设置。 10：**中央对齐模式 2。**计数器以交替方式向上或向下计数。仅当计数器向上计数时，配置为输出的通道（GP16C4T_CHMRn 寄存器中 CCnSSEL=00）的输出比较中断标志位才会被设置。 11：**中央对齐模式 3。**计数器以交替方式向上或向下计数。当计数器向上或向下计数时，配置为输出的通道（GP16C4T_CHMRn 寄存器中 CCnSSEL=00）的输出比较中断标志位均会被设置。 注意：当计数器使能时（CNTEN=1），不允许从边沿对齐模式转换到中央对齐模式

DIRSEL	Bit 4	R/W	**计数器方向选择** 0：计数器向上计数 1：计数器向下计数 注意：当计数器配置为中央对齐模式或者编码器模式时，该位只读。
SPMEN	Bit 3	R/W	**单脉冲模式** 0：当发生更新事件时，计数器不停止。 1：当发生下一次更新事件（CNTEN 位清零）时，计数器停止。
UERSEL	Bit 2	R/W	**更新请求源** 该位由软件置 1 或清零，来选择 UEV 事件源。 0：如果更新中断或 DMA 请求使能，则下述任一事件都可产生更新中断或 DMA 请求： – 计数器上溢/下溢 – 设置 SGU 位 – 从模式控制器产生的更新 1：如果更新中断或 DMA 请求使能，仅计数器上溢/下溢才能产生更新中断或 DMA 请求中断
DISUE	Bit 1	R/W	**更新禁止** 该位由软件置 1 或清零来使能/禁止 UEV 事件的产生。 0：UEV 使能. 更新事件（UEV）由下列任一事件产生： – 计数器上溢/下溢 – 设置 SGU 位 – 从模式控制器产生的更新 缓冲寄存器载入他们的预载值。 1：UEV 禁止. 不产生更新事件，影子寄存器保持他们的值（ARRV， PSCV， CCRVx）. 如果从从模式控制器接收到硬件复位，计数器和预分频器将被重新初始化。
CNTEN	Bit 0	R/W	**计数器使能** 0：计数器禁止 1：计数器使能 注意：如果软件设置了 CNTEN 位，外部时钟，门控模式和编码器模式才能工作。触发模式可由硬件自动设置 CNTEN 位。

21. 5. 2. 2 控制寄存器 2（GP16C4Tn_CON2）

控制寄存器 2（GP16C4Tn_CON2）

偏移地址：004_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								I1FSEL	TRGOSEL			CCDMASEL	Reserved		

Reserved	Bit 31-8	-	**保留，必须保持为复位值**
I1FSEL	Bit 7	R/W	**I1 选择** 0：GP16C4Tn_CH1 引脚与 I1 输入连接 1：GP16C4Tn_CH1， CH2 和 CH3 引脚与 I1 输入（XOR）连接
TRGOSEL	Bit 6-4	R/W	**选择主模式 TRGOUT 输出** 为同步（TRGOUT），该位可选择在主模式下发送至从计数器的信息。 000：**复位**–GP16C4Tn_SGE 寄存器中的 SGU 位被采用为触发输出（TRGOUT）。如果复位由触发输入生成（从模式控制器配置复位模式），则相较与实际复位，TRGOUT 上的信号将会延迟。 001：**使能**–计数器使能信号被用作触发输出（TRGOUT）。在从计数器使能的情况下，该设置用于在同一时间启动数次或者用来控制窗口。计数器使能信号是由 CNTEN 控制位与配置为门控模式的触发输入进行 OR 操作产生的。当计数器使能信号由触发输入控制，TRGOUT 上会产生延迟，除非被选为主/从模式（参考 GP16C4Tn_SMCON 寄存器中的 MSCFG 位的描述）。 010：**更新**–更新事件被选为触发输出（TRGOUT）。举例，主计数器可被用作从计数器的预分频器。 011：**比较脉冲**–一旦捕获或者比较匹配发生，当 CH1CCIF 标志位被置起（即便已为高电平），触发输出会发送一个正脉冲。 100：**比较**- 通道 1 比较输出信号用作触发输出 TRGOUT 101：**比较**- 通道 2 比较输出信号用作触发输出 TRGOUT 110：**比较**- 通道 3 比较输出信号用作触发输出 TRGOUT

			111：**比较**- 通道 4 比较输出信号用作触发输出 TRGOUT
CCDMASEL	Bit 3	R/W	**捕获/比较 DMA 选择** 0：当 CCn 事件发生，会发出 CCn DMA 请求。 1：当发生更新时间，会发出 CCn DMA 请求。
Reserved	Bit 2-0	-	**保留，必须保持为复位值**

21. 5. 2. 3 从模式控制寄存器（GP16C4Tn_SMCON）

从模式控制寄存器（GP16C4Tn_SMCON）

偏移地址：008_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ETPOL	ECM2EN	ETPSEL		ETFLT				MSCFG	TSSEL			Reserved	SMODS		

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
ETPOL	Bit 15	R/W	**外部触发极性** 0：正向 ET，高电平有效或上升沿有效 1：反正 ET，低电平有效或下降沿有效
ECM2EN	Bit 14	R/W	**使能外部时钟模式 2** 该位使能外部时钟模式 2 0：禁止外部时钟模式 2 1：使能外部时钟模式 2。计数器由 ETFP 信号上的有效边沿计数。 注意： 1. 设置 ECM2EN 位与选择外部时钟模式 1 且 TI 与 ETFP 相连接（SMODS=111 和 TSSEL=111）具有相同的效果。 2. 可同时使用外部时钟模式 2 与下列从模式：复位模式，门控模式和除法模式。在这种情况下，TI 不能与 ETFP 相连接（TSSEL 不能设置为 111。 3. 如果外部时钟模式 1 和外部时钟模式 2 同时使能，外部时钟输入为 ETFP。
ETPSEL	Bit 13-12	R/W	**外部触发预分频器** 外部触发信号频率最大为 GP16C4TnCLK 频率的 1/4。可使能预分频器来减小 ETFP 频率。该位有效用于输入高速外部时钟的情况。 00：预分频器关闭 01：ETFP 频率 2 分频 10：ETFP 频率 4 分频 11：ETFP 频率 8 分频
ETFLT	Bit 11-8	R/W	**外部触发滤波器** 该位定义了ETFP信号的采样频率和数字滤波器的滤波长度。 数字滤波器由一个事件计数器组成，每N个连续事件才视为一个有效边沿。 0000：无滤波器，采样频率为f_{DTS} 0001：$f_{SAMPLING} = f_{INT\_CLK}$，N = 2

			0010：$f_{SAMPLING} = f_{INT\_CLK}$，N = 4 0011：$f_{SAMPLING} = f_{INT\_CLK}$，N = 8 0100：$f_{SAMPLING} = f_{INT\_CLK} / 2$，N = 6 0101：$f_{SAMPLING} = f_{INT\_CLK} / 2$，N = 8 0110：$f_{SAMPLING} = f_{INT\_CLK} / 4$，N = 6 0111：$f_{SAMPLING} = f_{INT\_CLK} / 4$，N = 8 1000：$f_{SAMPLING} = f_{INT\_CLK} / 8$，N = 6 1001：$f_{SAMPLING} = f_{INT\_CLK} / 8$，N = 8 1010：$f_{SAMPLING} = f_{INT\_CLK} / 16$，N = 5 1011：$f_{SAMPLING} = f_{INT\_CLK} / 16$，N = 6 1100：$f_{SAMPLING} = f_{INT\_CLK} / 16$，N = 8 1101：$f_{SAMPLING} = f_{INT\_CLK} / 32$，N = 5 1110：$f_{SAMPLING} = f_{INT\_CLK} / 32$，N = 6 1111：$f_{SAMPLING} = f_{INT\_CLK} / 32$，N = 8 注意： 当ETFLT[3:0] = 1， 2或3时，公式中的f_{DTS}由INT_CLK 取代。
MSCFG	Bit 7	R/W	**主/从模式** 0：无动作 1：延迟触发输入（In）上的事件来允许当前计时器和其从器件之间的同步。该设置有效用于使用单个外部事件来同步多个计时器。
TSSEL	Bit 6-4	R/W	**触发选择** 该位用来选择不同的触发输入来同步计数器。 000：内部触发 0（IT0） 001：内部触发 1（IT1） 010：内部触发 2（IT2） 011：内部触发 3（IT3） 100：I1 边沿检测器（I1F_ED） 101：滤波计时器输入 1 110：滤波计时器输入 2 111：外部触发输入 注意：为了避免错误边沿检测，该位在不使用时（SMODS=000）才能改变。
Reserved	Bit 3	-	保留，必须保持为复位值
SMODS	Bit 2-0	R/W	**选择从模式功能** 当选择外部信号，触发信号TI的有效边沿与外部输入的极性有关系（详见输入控制寄存器和控制寄存器描述） 000：**禁止从模式**–如果CNTEN = ‘1’，则预分频器直接由内部时钟计数。 001：**编码器模式1**–计数器向上/向下计数I2边沿，取决于I1电平。 010：**编码器模式2** -计数器向上/向下计数I1边沿，取决于I2电平

			011：**编码器模式3** -计数器向上/向下计数I1边沿检出和I2边沿检出边沿，取决于另一个输入的电平。 100：**复位模式–**选中的触发输入的上升沿重新初始化计数器，生成寄存器的更新 101：**门控模式–**当触发输入TI为高电平，计数器时钟使能。一旦触发变为低电平，计数器停止计数（并非复位）。计数器的启动和停止均受控制。 110：**触发模式–**计数器在触发信号TI的上升沿处启动（不复位）。仅寄存器的启动受控制。 111：**外部时钟模式1–**计数器在TI的上升沿计数 注意：如果I1双边沿检出被选为触发输入（TSSEL='100'），不能使用门控模式。I1每一次转换，I1双边沿检出就会输出1个脉冲，而门控模式则是检查触发信号的电平。 注意：在发生来自主计时器的接收事件之前，从计时器的时钟必须先使能，且在接收来自主计时器的触发过程中，从计数器时钟不能即时更改。

21. 5. 2. 4 中断使能寄存器（GP16C4Tn_IER）

中断使能寄存器（GP16C4Tn_IER）																													
偏移地址：$00C_H$																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CC4OIT	CC3OIT	CC2OIT	CC1OIT	Reserved		TRGIT	Reserved	CC4IT	CC3IT	CC2IT	CC1IT	UIT

Reserved	Bit31-13	-	保留，必须保持复位值。
CC4OIT	Bit12	W	使能捕获/比较 4 捕获溢出中断 0：无效 1：使能
CC3OIT	Bit11	W	使能捕获/比较 3 捕获溢出中断 0：无效 1：使能
CC2OIT	Bit10	W	使能捕获/比较 2 捕获溢出中断 0：无效 1：使能
CC1OIT	Bit9	W	使能捕获/比较 1 捕获溢出中断 0：无效 1：使能
Reserved	Bit 8-7	-	保留，必须保持为复位值
TRGIT	Bit6	W	使能触发中断 0：无效 1：使能
Reserved	Bit 5	-	保留，必须保持为复位值
CC4IT	Bit4	W	使能捕获/比较 4 中断 0：无效 1：使能
CC3IT	Bit3	W	使能捕获/比较 3 中断 0：无效 1：使能
CC2IT	Bit2	W	使能捕获/比较 2 中断 0：无效 1：使能
CC1IT	Bit1	W	使能捕获/比较 1 中断 0：无效 1：使能
UIT	Bit0	W	使能更新事件中断 0：无效 1：使能

21. 5. 2. 5 中断禁止寄存器（GP16C4Tn_IDR）

中断禁止寄存器（GP16C4Tn_IDR）

偏移地址：0010_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CC4OIT	CC3OIT	CC2OIT	CC1OIT	Reserved		TRGIT	Reserved	CC4IT	CC3IT	CC2IT	CC1IT	UIT

Reserved	Bit 31-13	-	**保留，必须保持复位值。**
CC4OIT	Bit12	W	**使能捕获/比较 4 捕获溢出中断** 0：无效 1：使能
CC3OIT	Bit11	W	**使能捕获/比较 3 捕获溢出中断** 0：无效 1：使能
CC2OIT	Bit10	W	**使能捕获/比较 2 捕获溢出中断** 0：无效 1：使能
CC1OIT	Bit9	W	**使能捕获/比较 1 捕获溢出中断** 0：无效 1：使能
Reserved	Bit 8-7	-	**保留，必须保持为复位值**
TRGIT	Bit 6	W	**禁止触发中断** 0：无效 1：禁止
Reserved	Bit 5	-	**保留，必须保持为复位值**
CC4IT	Bit 4	W	**禁止捕获/比较 4 中断** 0：无效 1：禁止
CC3IT	Bit 3	W	**禁止捕获/比较 3 中断** 0：无效 1：禁止
CC2IT	Bit 2	W	**禁止捕获/比较 2 中断** 0：无效 1：禁止
CC1IT	Bit 1	W	**禁止捕获/比较 1 中断** 0：无效 1：禁止
UIT	Bit 0	W	**禁止更新中断** 0：无效 1：禁止

21. 5. 2. 6 中断有效状态寄存器（GP16C4Tn_IVS）

中断有效状态寄存器（GP16C4Tn_IVS）

偏移地址：0014_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CC4OIT	CC3OIT	CC2OIT	CC1OIT	Reserved		TRGIT	Reserved	CC4IT	CC3IT	CC2IT	CC1IT	UIT

Reserved	Bit 31-13	-	**保留**
CC4OIT	Bit12	W	**使能捕获/比较 4 捕获溢出中断** 0：无效 1：使能
CC3OIT	Bit11	W	**使能捕获/比较 3 捕获溢出中断** 0：无效 1：使能
CC2OIT	Bit10	W	**使能捕获/比较 2 捕获溢出中断** 0：无效 1：使能
CC1OIT	Bit9	W	**使能捕获/比较 1 捕获溢出中断** 0：无效 1：使能
Reserved	Bit 8-7	-	**保留，必须保持为复位值**
TRGIT	Bit 6	R	**触发中断状态** 0：禁止 1：使能
Reserved	Bit 5	-	**保留，必须保持为复位值**
CC4IT	Bit 4	R	**通道 4 捕获/比较中断状态** 0：禁止 1：使能
CC3IT	Bit 3	R	**通道3捕获/比较中断状态** 0：禁止 1：使能
CC2IT	Bit 2	R	**通道 2 捕获/比较中断状态** 0：禁止 1：使能
CC1IT	Bit 1	R	**通道2捕获/比较中断状态** 0：禁止 1：使能
UIT	Bit 0	R	**更新事件中断状态** 0：禁止 1：使能

21. 5. 2. 7 原始中断标志寄存器（GP16C4Tn_RIF）

原始中断标志寄存器（GP16C4Tn_RIF）

偏移地址：018_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CH4OVIF	CH3OVIF	CH2OVIF	CH1OVIF	Reserved		TRGIF	Reserved	CH4CCIF	CH3CCIF	CH2CCIF	CH1CCIF	UEVTIF

Reserved	Bit 31-13	-	**保留，必须保持为复位值**
CH4OVIF	Bit 12	R	**捕获/比较 4 捕获溢出中断标志** 仅当相应的通道配置为捕获输入状态时，该标志位才由硬件设置。对 GP16C4Tn_ICR 写 1 来清除原始中断。 0：未检测到捕获溢出 1：当 CH4CCIF 标志位置起时，捕获计数器值至 GP16C4Tn_CCVAL1 寄存器
CH3OVIF	Bit 11	R	**捕获/比较 3 捕获溢出中断标志** 仅当相应的通道配置为捕获输入状态时，该标志位才由硬件设置。对 GP16C4Tn_ICR 写 1 来清除原始中断。 0：未检测到捕获溢出 1：当 CH3CCIF 标志位置起时，捕获计数器值至 GP16C4Tn_CCVAL1 寄存器
CH2OVIF	Bit 10	R	**捕获/比较 2 捕获溢出中断标志** 仅当相应的通道配置为捕获输入状态时，该标志位才由硬件设置。对 GP16C4Tn_ICR 写 1 来清除原始中断。 0：未检测到捕获溢出 1：当 CH2CCIF 标志位置起时，捕获计数器值至 GP16C4Tn_CCVAL1 寄存器
CH1OVIF	Bit 9	R	**捕获/比较 1 捕获溢出中断标志** 仅当相应的通道配置为捕获输入状态时，该标志位才由硬件设置。对 GP16C4Tn_ICR 写 1 来清除原始中断。 0：未检测到捕获溢出 1：当 CH1CCIF 标志位置起时，捕获计数器值至 GP16C4Tn_CCVAL1 寄存器
Reserved	Bit 8-7	-	保留，必须保持为复位值
TRGIF	Bit 6	R	**触发中断标志** 如果触发中断使能，当从模式控制器在门控模式

			以外的所有模式下使能，发生触发事件时（In 上检测到有效边沿），该标志位被硬件置起。对 GP16C4Tn_ICR 写 1 来清除原始中断. 0: 未发生触发事件 1: 触发中断被挂起
Reserved	Bit 5	-	**保留，必须保持为复位值**
CH4CCIF	Bit 4	R	**捕获/比较 4 中断标志** 参考 CH1CCIF 描述
CH3CCIF	Bit 3	R	**捕获/比较 3 中断标志** 参考 CH1CCIF 描述
CH2CCIF	Bit 2	R	**捕获/比较 2 中断标志** 参考 CH1CCIF 描述
CH1CCIF	Bit 1	R	**捕获/比较 1 中断标志** **如果 CC1 通道配置为输出:** 如果中断使能，除去中央对齐模式的情况（参考 GP16C4Tn_CON1 寄存器中 CMSEL 的描述），当计数值与比较值匹配，该标志位由硬件置起。对 GP16C4Tn_ICR 写 1 来清除原始中断. 0: 不匹配. 1: GP16C4Tn_COUNT 计数值与 GP16C4Tn_CCVAL1 值匹配。当 GP16C4Tn_CCVAL1 寄存器值大于 GP16C4Tn_AR 值，发生计数器上溢时（递增模式和递增/递减模式）或下溢时（递减模式），CH1CCIF 为被置起 **如果 CC1 通道配置为输入:** 发生捕获时，该位由硬件置起。该位可通过软件或者读取 GP16C4Tn_CCVAL1 寄存器来清零。 0: 未发生输入捕获 1: 计数值捕获至 GP16C4Tn_CCVAL1 寄存器（IC1 上检测到与选中极性匹配的边沿）
UEVTIF	Bit 0	R	**更新中断标志** 如果更新中断使能，当发生更新事件，该标志位由硬件置起。对 GP16C4Tn_ICR 写 1 来清除原始中断。 0：未发生更新。 1：更新中断被挂起。当寄存器更新时，该位被硬件置起： –当重复计数器值发生上溢或者下溢（若重复计数器=0，则更新）和当 GP16C4Tn_CON1 寄存器中 DISUE=0 –当使用 GP16C4Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时，如果 GP16C4Tn_CON1 寄存中的 UERSEL=0 和

			DISUE=0 –当 CNT 由触发事件来重新初始化，如果 GP16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0

21. 5. 2. 8 中断标志屏蔽寄存器（GP16C4Tn_IFM）

中断标志屏蔽寄存器（GP16C4Tn_IFM）																															
偏移地址：$001C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CH4OVIM	CH3OVIM	CH2OVIM	CH1OVIM	Reserved		TRGIM	Reserved	CH4CCIM	CH3CCIM	CH2CCIM	CH1CCIM	UEVTIM

Reserved	Bit 31-13	-	**保留**
CH4OVIM	Bit 12	R	**屏蔽通道 4 捕获/比较捕获溢出中断标志** 0：未检测到捕获溢出 1：当 CH4CCIF 标志为置起时，捕获计数器值至 GP16C4Tn_CCVAL1 寄存器
CH3OVIM	Bit 11	R	**屏蔽通道 3 捕获/比较捕获溢出中断标志** 0：未检测到捕获溢出 1：当 CH3CCIF 标志为置起时，捕获计数器值至 GP16C4Tn_CCVAL1 寄存器
CH2OVIM	Bit 10	R	**屏蔽通道 2 捕获/比较捕获溢出中断标志** 0：未检测到捕获溢出 1：当 CH2CCIF 标志为置起时，捕获计数器值至 GP16C4Tn_CCVAL1 寄存器
CH1OVIM	Bit 9	R	**屏蔽通道 1 捕获/比较捕获溢出中断标志** 0：未检测到捕获溢出 1：当 CH1CCIF 标志为置起时，捕获计数器值至 GP16C4Tn_CCVAL1 寄存器
Reserved	Bit 8-7	-	**保留**
TRGIM	Bit 6	R	**屏蔽触发中断标志** 如果触发中断使能，当从模式控制器在门控模式以外的所有模式下使能，发生触发事件时（TI 上检测到有效边沿），该标志位被硬件置起。对 GP16C4Tn_ICR 写 1 来清除原始中断。 0：未发生触发事件 1：触发中断被挂起
Reserved	Bit 5	-	**保留**
CH4CCIM	Bit 4	R	**屏蔽通道 4 捕获/比较中断标志** 参考 CH1CCIM 描述
CH3CCIM	Bit 3	R	**屏蔽通道 3 捕获/比较中断标志** 参考 CH1CCIM 描述
CH2CCIM	Bit 2	R	**屏蔽通道 2 捕获/比较中断标志** 参考 CH1CCIM 描述
CH1CCIM	Bit 1	R	**屏蔽通道 1 捕获/比较中断标志如果通道 1 配置为**

			输出： 如果中断使能，除去中央对齐模式的情况（参考 GP16C4Tn_CON1 寄存器中 CMSEL 的描述），当计数值与比较值匹配，该标志位由硬件置起。对 GP16C4Tn_ICR 写 1 来清除原始中断。 0：不匹配。 1：GP16C4Tn_COUNT 计数值与 GP16C4Tn_CCVAL1 值匹配。当 GP16C4Tn_CCVAL1 寄存器值大于 GP16C4Tn_AR 值，发生计数器上溢时（递增模式和递增/递减模式）或下溢时（递减模式），CH1CCIF 为被置起。 **如果通道配置为输入：** 发生捕获时，该位由硬件置起。该位可通过软件或者读取 GP16C4Tn_CCVAL1 寄存器来清零。 0：未发生输入捕获 1：计数值捕获至 GP16C4Tn_CCVAL1 寄存器（I1 上检测到与选中极性匹配的边沿）
UEVTIM	Bit 0	R	**屏蔽更新事件中断标志** 如果更新中断使能，当发生更新事件，该标志位由硬件置起。对 GP16C4Tn_ICR 写 1 来清除原始中断。 0：未发生更新。 1：更新中断被挂起。当寄存器更新时，该位被硬件置起： –当重复计数器值发生上溢或者下溢（若重复计数器=0，则更新）和当 GP16C4Tn_CON1 寄存器中 DISUE=0 –当使用 GP16C4Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时，如果 GP16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0 –当 CNT 由触发事件来重新初始化，如果 GP16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0

21. 5. 2. 9 中断清零寄存器（GP16C4Tn_ICR）

中断清零寄存器（GP16C4Tn_ICR）																															
偏移地址：020_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CH4OVIC	CH3OVIC	CH2OVIC	CH1OVIC	Reserved		TRGIC	Reserved	CH4CCIC	CH3CCIC	CH2CCIC	CH1CCIC	UEVTIC

Reserved	Bit 31-13	-	保留，必须保持为复位值
CH4OVIC	Bit 12	R	**通道 4 捕获/比较捕获溢出中断标志清除** 0：无效 1：CH4OVIF 清除
CH3OVIC	Bit 11	R	**通道 3 捕获/比较捕获溢出中断标志** 0：无效 1：CH3OVIF 清除
CH2OVIC	Bit 10	R	**通道 2 捕获/比较捕获溢出中断标志** 0：无效 1：CH2OVIF 清除
CH1OVIC	Bit 9	R	**通道 1 捕获/比较捕获溢出中断标志** 0：无效 1：CH1OVIF 清除
Reserved	Bit 8-7	-	保留，必须保持为复位值
TRGIC	Bit 6	C_W1	**触发中断清零** 0：无效 1：触发中断清零（GP16C4Tn_RIF）
Reserved	Bit 5	-	保留，必须保持为复位值
CH4CCIC	Bit 4	C_W1	**捕获/比较 4 中断清零** 参考 CH1CCIC 描述
CH3CCIC	Bit 3	C_W1	**捕获/比较 3 中断清零** 参考 CH1CCIC 描述
CH2CCIC	Bit 2	C_W1	**捕获/比较 2 中断清零** 参考 CH1CCIC 描述
CH1CCIC	Bit 1	C_W1	**捕获/比较 1 中断清零** 0：无效 1：捕获/比较中断清零（GP16C4Tn_RIF）
UEVTIC	Bit 0	C_W1	**更新中断清零** 0：无效 1：更新中断清零（GP16C4Tn_RIF）

21. 5. 2. 10 软件生成事件寄存器（GP16C4Tn_SGE）

软件生成事件寄存器（GP16C4Tn_SGE）

偏移地址：024_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									SGTRG	Reserved	SGCC4E	SGCC3E	SGCC2E	SGCC1E	SGU

Reserved	Bit 31-7	-	**保留，必须保持为复位值**
SGTRG	Bit 6	W	**触发生成** 该位由软件设置来生成触发事件，可由硬件自动清零. 0：无动作 1：GP16C4Tn_RIF 寄存器中的 TRGIF 被置起，产生相关中断或 DMA 传输
Reserved	Bit 5	-	**保留，必须保持为复位值**
SGCC4E	Bit 4	W	**捕获/比较 4 生成** 参考 SGCC1E 描述
SGCC3E	Bit 3	W	**捕获/比较 3 生成** 参考 SGCC1E 描述
SGCC2E	Bit 2	W	**捕获/比较 2 生成** 参考 SGCC1E 描述
SGCC1E	Bit 1	W	**捕获/比较 1 生成** 该位由软件设置来生成事件，可由硬件自动清零. 0：无动作 1：通道 1 上产生捕获/比较事件： **如果通道 1 配置为输出：** CH1CCIF 标志位被置起，产生相应中断或 DMA 请求发送 **如果通道 1 配置为输入：** 当前计数值捕获至 GP16C4Tn_CCVAL1 寄存器. CH1CCIF 标志位被置起，产生相应中断或 DMA 请求发送。CH1OVIF 标志位置起如果 CH1CCIF 标志位为高电平。
SGU	Bit 0	W	**更新生成** 该位由软件设置，可由硬件自动清零. 0：无动作 1：重新初始化计数器，更新寄存器。注意，预分频器也会被清零（但预分频比不会受到影响）。如果使用中央对齐模式或者 DIRSEL=0（递增），则计数器将清零；否则如果 DIRSEL=1（递减），则将使用自动重载入值。

21. 5. 2. 11 捕获/比较模式寄存器 1（GP16C4Tn_CHMR1）

◆ 输出比较模式

捕获/比较模式寄存奇 1（GP16C4Tn_CHMR1）																															
偏移地址：028_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CH2OCLREN	CH2OMOD			CH2OPREN	CH2OHSEN	CC2SSEL		CH1OCLREN	CH1OMOD			CH1OPREN	CH1OHSEN	CC1SSEL	

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
CH2OCLREN	Bit 15	R/W	**输出比较 2 清零使能** 参考 CH1OCLREN 描述
CH2OMOD	Bit 14-12	R/W	**输出比较 2 模式** 参考 CH1OMOD 描述
CH2OPREN	Bit 11	R/W	**输出比较 2 预载使能** 参考 CH1OPREN 描述
CH2OHSEN	Bit 10	R/W	**输出比较 2 高速使能** 参考 CH1OHSEN 描述
CC2SSEL	Bit 9-8	R/W	**输出比较 2 选择** 该位定义了通道以及使用的输入的方向（输入/输出） 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I2** 10：通道配置为输入，捕获源为 **I1** 11：通道配置为输入，捕获源为 **ITn 或 I1** 的双边沿检出。 仅当内部触发输入通过 TSSEL 位（GP16C4Tn_SMCON 寄存器）选择时，该模式才能工作。 注意：当通道为关闭状态时（GP16C4Tn_CCEP 中 CC2EN = ‘0’），CC2SSEL 为只写。
CH1OCLREN	Bit 7	R/W	**输出比较 1 清零使能** 0：通道 1 比较输出不会受到 ETFP 输入影响 1：当 ETFP 输入上检测到高电平时，通道 1 比较输出将被清零
CH1OMOD	Bit 6-4	R/W	**输出比较 1 模式** 该位定义了输出参考信号通道 1 比较输出的行为。通道 1 比较输出为高有效，CH1O 的有效电平由 CC1POL 位决定。 000 :**冻结**–输出比较寄存器 GP16C4Tn_CCVAL1

			寄存器和 GP16C4Tn_COUNT 计数器之间的比较对输出无效。 001：**发生匹配时设置通道 1 为有效电平**–当计数器 GP16C4Tn_COUNT 与捕获/比较寄存器 1GP16C4Tn_CCVAL1 发生匹配后，通道 1 比较输出信号强制为高电平。 010：**发生匹配时设置通道 1 为无效电平**-当计数器 GP16C4Tn_COUNT 与捕获/比较寄存器 1GP16C4Tn_CCVAL1 发生匹配后，通道 1 比较输出信号强制为低电平。 011：**翻转** -当 GP16C4Tn_COUNT=GP16C4Tn_CCVAL1，通道 1 比较输出发生翻转。 100：**强制为无效电平** - 通道 1 比较输出强制为低电平。 101：**强制为有效电平**- 通道 1 比较输出强制为高电平。 110：**PWM 模式 1** -在递增模式下，当 GP16C4Tn_COUNT<GP16C4Tn_CCVAL1，通道 1 为有效电平，否则，通道 1 为无效电平。在递减模式下，当 GP16C4Tn_COUNT>GP16C4Tn_CCVAL1，通道 1 为无效电平（通道 1 比较输出='0'），否则通道 1 为有效电平（通道 1 比较输出='1'）。 111：**PWM 模式 2** -在递增模式下，当 GP16C4Tn_COUNT<GP16C4Tn_CCVAL1，通道 1 为无效电平，否则，通道 1 为有效电平。在递减模式下，当 GP16C4Tn_COUNT>GP16C4Tn_CCVAL1，通道 1 为有效电平，否则通道 1 为无效电平。 注意： 在 PWM 模式 1 和 2 中，仅当比较结果更改或当输出比较模式从冻结模式转换成 PWM 模式，比较输出电平才会更改。
CH1OPREN	Bit 3	R/W	**输出比较 1 预载使能** 0：GP16C4Tn_CCVAL1 的预载寄存器禁止。GP16C4Tn_CCVAL1 在任何时候都可写，新写入的值将立刻生效。 1：GP16C4Tn_CCVAL1 的预载寄存器使能。读/写操作可访问预载寄存器。每当发生一次更新事件，GP16C4Tn_CCVAL1 预载入值将会被填入有效寄存器。 注意： 仅在单脉冲模式下（GP16C4Tn_CON1 寄存器中

			的 SPMEN 设置为 1），PWM 模式可在不经过验证预载寄存器的情况下使用。其他情况下的行为不做保证。
CH1OHSEN	Bit 2	R/W	**输出比较 1 高速使能** 该位用来加速在 CC 输出上的输入触发事件的效应。 0：当触发开启，通道 1 运作正常取决于计数器和 CCRV1 的值。当触发输入上发现边沿时，至少需要 5 个时钟周期来激活通道 1 输出。 1：触发输入上的有效沿类似于通道 1 输出上的比较匹配。设置 OC 为 1 用来比较电平，采样触发输入和激活通道 1 输出的延时将会减少至 3 个时钟周期。只有当通道配置为 PWM1 或 PWM2 模式，CH1OHSEN 才会起作用。
CC1SSEL	Bit 1-0	R/W	**捕获/比较 1 选择** 该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I1** 10：通道配置为输入，捕获源为 **I2** 11：通道配置为输入，捕获源为 ITn 或 I1 的双边沿检出。 只有当内部触发输入是通过 TSSEL 位（GP16C4Tn_SMCON 寄存器）选择时，该模式才运行。 注意：当通道关闭（GP16C4Tn_CCEP 寄存器中的 CC1EN = ‘0’），CC1SSEL 为只写。

◆ 输入捕获模式

捕获/比较模式寄存奇 1（GP16C4Tn_CHMR1）																													
偏移地址：028_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																I2FLT				IC2PRES		CC2SSEL		I1FLT				IC1PRES		CC1SSEL	

Reserved	Bit 31-16	-	保留，必须保持为复位值
I2FLT	Bit 15-12	R/W	**输入捕获 2 滤波器** 参考 I1FLT 描述
IC2PRES	Bit 11-10	R/W	**输入捕获 2 预分频器** 参考 IC1PRES 描述
CC2SSEL	Bit 9-8	R/W	**输入捕获 2 选择** 该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I2** 10：通道配置为输入，捕获源为 **I1** 11：通道配置为输入，捕获源为 **ITn 或 I1 的双边沿检出** 只有当内部触发输入是通过 TSSEL 位（GP16C4Tn_SMCON 寄存器）选择时，该模式才运行 注意：当通道关闭（GP16C4Tn_CCEP 寄存器中的 CC2EN = '0'），CC2SSEL 为只写。
I1FLT	Bit 7-4	R/W	**输入捕获 1 滤波器** 该位定义了 I1 输入的采样频率和数字滤波器的长度。 数字滤波器由一个事件计数器组成，每 N 个连续事件才视为一个有效边沿： 0000：无滤波器，采样频率为 f_{DTS} 0001：$f_{SAMPLING} = f_{INT\_CLK}$, N = 2 0010：$f_{SAMPLING} = f_{INT\_CLK}$, N = 4 0011：$f_{SAMPLING} = f_{INT\_CLK}$, N = 8 0100：$f_{SAMPLING} = f_{DTS} / 2$, N = 6 0101：$f_{SAMPLING} = f_{DTS} / 2$, N = 8 0110：$f_{SAMPLING} = f_{DTS} / 4$, N = 6 0111：$f_{SAMPLING} = f_{DTS} / 4$, N = 8 1000：$f_{SAMPLING} = f_{DTS} / 8$, N = 6 1001：$f_{SAMPLING} = f_{DTS} / 8$, N = 8 1010：$f_{SAMPLING} = f_{DTS} / 16$, N = 5 1011：$f_{SAMPLING} = f_{DTS} / 16$, N = 6

			1100：$f_{SAMPLING} = f_{DTS} / 16$, N = 8 1101：$f_{SAMPLING} = f_{DTS} / 32$, N = 5 1110：$f_{SAMPLING} = f_{DTS} / 32$, N = 6 1111：$f_{SAMPLING} = f_{DTS} / 32$, N = 8 注意：当 ICxF [3：0] = 1, 2 or 3 时，公式中的 f_{DTS} 由 INT_CLK 取代。
IC1PRES	Bit 3-2	R/W	**输入捕获 1 预分频器** 该位定义了作用在 CC1 输入（I1）上的预分频比。当 CC1EN='0' （GP16C4Tn_CCEP 寄存器），预分频器将复位。 00：无预分频器。每当捕获输入上检测到边沿时，发生捕获动作。 01：每发生 2 次事件，执行一次捕获 10：每发生 4 次事件，执行一次捕获 11：每发生 8 次事件，执行一次捕获
CC1SSEL	Bit 1-0	R/W	**输入捕获 1 选择** 该位定义了通道和使用的输入的方向 00：CC1 通道配置为**输出** 01：CC1 通道配置为输入，IC1 映射到 **I1** 10：CC1 通道配置为输入，IC1 映射到 **I2** 11：CC1 通道配置为输入，捕获源为 ITn 或 I1 的双边沿检出。只有当内部触发输入是通过 TSSEL 位（GP16C4Tn_SMCON 寄存器）选择时，该模式才运行 注意： 当通道关闭（GP16C4Tn_CCEP 寄存器中的 CC1EN = '0'），CC1SSEL 为只写。

21. 5. 2. 12 捕获/比较模式寄存器 2（GP16C4Tn_CHMR2）

◆ 输出比较模式

捕获/比较模式寄存器 2（GP16C4Tn_CHMR2）																															
偏移地址：$02C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CH4OCLREN	CH4OMOD			CH4OPREN	CH4OHSEN	CC4SSEL		CH3OCLREN	CH3OMOD			CH3OPREN	CH3OHSEN	CC3SSEL	

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
CH4OCLREN	Bit 15	R/W	**输出比较 4 清零使能** 参考 CH1OCLREN 描述
CH4OMOD	Bit 14-12	R/W	**输出比较 4 模式** 参考 CH1OMOD 描述
CH4OPREN	Bit 11	R/W	**输出比较 4 预载使能** 参考 CH1OPREN 描述
CH4OHSEN	Bit 10	R/W	**输出比较 4 高速使能** 参考 CH1OHSEN 描述
CC4SSEL	Bit 9-8	R/W	**输出比较 4 选择** 该位定义了通道以及使用的输入的方向（输入/输出）。 00：通道配置为输出 01：通道配置为输入，捕获源为 I4 10：通道配置为输入，捕获源为 I3 11：通道配置为输入，捕获源为 ITn 或 I1 的双边沿检出 仅当内部触发输入通过 TSSEL 位（GP16C4Tn_SMCON 寄存器）选择时，该模式才能工作。 注意：当通道为关闭状态时（GP16C4Tn_CCEP 中 CC4EN = '0'），CC4SSEL 为只写。
CH3OCLREN	Bit 7	R/W	**输出比较 3 清零使能** 参考 CH1OCLREN 描述
CH3OMOD	Bit 6-4	R/W	**输出比较 3 模式** 参考 CH1OMOD 描述
CH3OPREN	Bit 3	R/W	**输出比较 3 预载使能** 参考 CH1OPREN 描述
CH3OHSEN	Bit 2	R/W	**输出比较 3 高速使能** 参考 CH1OHSEN 描述
CC3SSEL	Bit 1-0	R/W	**捕获/比较 3 选择**

			该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I3** 10：通道配置为输入，捕获源为 **I4** 11：通道配置为输入，捕获源为 ITn 或 I1 的双边沿检出。 只有当内部触发输入是通过 TSSEL 位（GP16C4Tn_SMCON 寄存器）选择时，该模式才运行。 注意：当通道关闭（GP16C4Tn_CCEP 寄存器中的 CC3EN = '0'），CC3SSEL 为只写

◆ 输入捕获模式

捕获/比较模式寄存器 2（GP16C4Tn_CHMR2）																													
偏移地址：$02C_H$																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																I4FLT				IC4PRES		CC4SSEL		IC3FLT				IC3PRES		CC3SSEL	

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
I4FLT	Bit 15-12	R/W	**输入捕获 4 滤波器** 参考 I1FLT 描述
IC4PRES	Bit 11-10	R/W	**输入捕获 4 预分频器** 参考 IC1PRES 描述
CC4SSEL	Bit 9-8	R/W	**输入捕获 4 选择** 该位定义了通道和使用的输入的方向。 00：CC4 通道配置为**输出** 01：CC4 通道配置为输入，IC4 映射到 **TI4** 10：CC4 通道配置为输入，IC4 映射到 **TI3** 11：CC4 通道配置为输入，IC4 映射到 **TRC** 只有当内部触发输入是通过 TSSEL 位（GP16C4Tn_SMCON 寄存器）选择时，该模式才运行 注意： 当通道关闭（GP16C4Tn_CCEP 寄存器中的 CC4EN = ‘0’），CC4SSEL 为只写。
IC3FLT	Bit 7-4	R/W	**输入捕获 3 滤波器** 参考 I1FLT 描述
IC3PRES	Bit 3-2	R/W	**输入捕获 3 预分频器** 参考 IC1PRES 描述
CC3SSEL	Bit 1-0	R/W	**输入捕获 3 选择** 该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I3** 10：通道配置为输入，捕获源为 **I4** 11：通道配置为输入，捕获源为 ITn 或 I1 的双边沿检出 只有当内部触发输入是通过 TSSEL 位（GP16C4Tn_SMCON 寄存器）选择时，该模式才运行 注意：当通道关闭（GP16C4Tn_CCEP 寄存器中的 CC3EN = ‘0’），CC3SSEL 为只写。

21. 5. 2. 13 捕获/比较使能寄存器（GP16C4Tn_CCEP）

捕获/比较使能寄存器（GP16C4Tn_CCEP）																													
偏移地址：030_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		CC4POL	CC4EN	Reserved		CC3POL	CC3EN	Reserved		CC2POL	CC2EN	Reserved		CC1POL	CC1EN

Reserved	Bit 31-14	-	**保留，必须保持为复位值**
CC4POL	Bit 13	R/W	**捕获/比较 4 输出极性** 参考 CC1POL 描述
CC4EN	Bit 12	R/W	**捕获/比较 4 输出使能** 参考 CC1EN 描述
Reserved	Bit 11-10	-	**保留，必须保持为复位值**
CC3POL	Bit 9	R/W	**捕获/比较 3 输出极性** 参考 CC1POL 描述
CC3EN	Bit 8	R/W	**捕获/比较 3 输出使能** 参考 CC1EN 描述
Reserved	Bit 7-6	-	**保留，必须保持为复位值**
CC2POL	Bit 5	R/W	**捕获/比较 2 输出极性** 参考 CC1POL 描述
CC2EN	Bit 4	R/W	**捕获/比较 2 输出使能** 参考 CC1EN 描述
Reserved	Bit 3-2	-	**保留，必须保持为复位值**
CC1POL	Bit 1	R/W	**捕获/比较 1 输出极性** **通道配置为输出：** 0：CH1O 高有效 1：CH1O 低有效 **通道配置为输入：** CC1POL 为触发和捕获操作选择 I1 边沿检出和 I2 边沿检出的有效极性。 0：正向/上升沿 电路对 In 边沿检出的上升沿敏感（在复位，外部时钟或触发模式下，进行捕获或触发），In 边沿检出不反向（门控模式或编码器模式下，进行触发）。 1：反向/下降沿 电路对 In 边沿检出的下降沿敏感（在复位，外部时钟或触发模式下，进行捕获或触发），In 边沿检出反向（门控模式或编码器模式下，进行触发）。
CC1EN	Bit 0	R/W	**捕获/比较 1 输出使能** **通道配置为输出：**

			0：关闭 - CH1O 无效。 1：开启 - CH1O 为对应输出引脚上的输出信号，由 CC1EN 决定 **通道配置为输入：** 该位决定了计数值是否能捕获到输入捕获/比较寄存器 1（GP16C4Tn_CCVAL1）。 0：禁止捕获。 1：使能捕获。

21. 5. 2. 14 计数器寄存器（GP16C4Tn_COUNT）

计数器寄存器（GP16C4Tn_COUNT）

偏移地址：034_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
CNTV	Bit 15-0	R/W	**计数值**

21. 5. 2. 15 预分频寄存器（GP16C4Tn_PRES）

预分频寄存器（GP16C4Tn_PRES）																															
偏移地址：038_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PSCV															

Reserved	Bit 31-16	-	保留，必须保持为复位值
PSCV	Bit 15-0	R/W	**预分频器值** 计数器时钟频率（CK_CNT）= f_{CK_PSC} /（PSCV [15:0] + 1） 每发生一次更新事件（包括当计数器由 GP16C4Tn_SGE 寄存器中的 SGU 位清零或当配置为复位模式时，通过触发控制器清零），PSCV 包含的值将被填入到有效的预分频寄存器内。

21. 5. 2. 16 自动重载寄存器（GP16C4Tn_AR）

自动重载寄存器（GP16C4Tn_AR）																															
偏移地址：$03C_H$																															
复位值：$00000000\_00000000\_11111111\_11111111_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ARRV															

Reserved	Bit 31-16	-	保留，必须保持为复位值
ARRV	Bit 15-0	R/W	**自动重载值** ARRV 中的值将被载入实际的自动重载寄存器中。当自动重载值为空，计数器被屏蔽。

21. 5. 2. 17 捕获/比较寄存器 1（GP16C4Tn_CCVAL1）

捕获/比较寄存器 1（GP16C4T_CCVAL1）																															
偏移地址：044_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV1															

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
CCRV1	Bit 15-0	R/W	**捕获/比较值 1** **如果通道 CCn 配置为输出：** CCRVn 中的值将被载入实际的捕获/比较寄存器中（预载值）。 如果在 GP16C4Tn_CHMRn 寄存器中的预载功能没有选中，CCRVn 中的值将被永久载入；否则，每当发生更新事件，预载值将会复制到有效的捕获/比较寄存器中。有效捕获/比较寄存器中包含的值将会与 GP16C4Tn_COUNT 中的值进行比较，并在 OCn 上输出。 **如果通道 CCn 配置为输入：** CCRVn 为由上一个输入捕获事件（ICn）传输的计数值。

21. 5. 2. 18 捕获/比较寄存器 2（GP16C4Tn_CCVAL2）

捕获/比较寄存器 2（GP16C4T_ CCVAL 2）																															
偏移地址：048_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV2															

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
CCRV2	Bit 15-0	R/W	**捕获/比较值 2** 参考 CCRV1 描述

21. 5. 2. 19 捕获/比较寄存器 3（GP16C4Tn_CCVAL3）

捕获/比较寄存器 3（GP16C4T_ CCVAL 3）																															
偏移地址：$04C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CRRV3															

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
CCRV3	Bit 15-0	R/W	**捕获/比较值 3** 参考 CCRV1 描述

21. 5. 2. 20 捕获/比较寄存器 4（GP16C4Tn_CCVAL4）

捕获/比较寄存器 4（GP16C4T_ CCVAL 4）																															
偏移地址：050_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV4															

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
CCRV4	Bit 15-0	R/W	**捕获/比较值 4** 参考 CCRV1 描述

21. 5. 2. 21 DMA使能寄存器（GP16C4Tn_DMAEN）

DMA 使能寄存器（GP16C4Tn_DMAEN）																												
偏移地址：058_H																												
复位值：$00000000\_00000000\_00000000\_00000000_B$																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									TRGDMA	COMDMA	CC4DMA	CC3DMA	CC2DMA	CC1DMA	UDMA

Reserved	Bit 31-7	-	**保留，必须保持为复位值**
TRGDMA	Bit 6	R/W	**触发DMA请求使能** 0：DMA请求禁止 1：DMA请求使能
COMDMA	Bit 5	R/W	**COM DMA访问使能** 0：DMA请求禁止 1：DMA请求使能
CC4DMA	Bit 4	R/W	**捕获/比较值 4 DMA 访问使能** 0：DMA 请求禁止 1：DMA 请求使能
CC3DMA	Bit 3	R/W	**捕获/比较值 3 DMA 访问使能** 0：DMA 请求禁止 1：DMA 请求使能
CC2DMA	Bit 2	R/W	**捕获/比较值 2 DMA 访问使能** 0：DMA 请求禁止 1：DMA 请求使能
CC1DMA	Bit 1	R/W	**捕获/比较值 1 DMA 访问使能** 0：DMA 请求禁止 1：DMA 请求使能
UDMA	Bit 0	R/W	**更新 DMA 请求使能** 0：DMA 请求禁止 1：DMA 请求使能

第22章 通用定时器（GP32C4T）

22. 1 概述

通用定时器（GP32C4T）包含一个 32 位自动重载计数器，该计数器由可配置的预分频器驱动。

通用定时器（GP32C4T）的用途广泛，可测量信号脉冲长度（输入捕获）或输出脉冲波形（比较输出、PWM）。

22. 2 特性

- 32 位递增，递减，递增/递减自动加载计数器
- 32 位可编程预分频器，可对计数器工作时钟进行 1 到 65536 间的任意分频
- 多达四个独立信道
  - 输入捕获
  - 输出比较
  - PWM 产生（边沿与中央对齐模式）
  - 单脉冲输出模式
- 同步电路用于外部信号控制定时器及内部互联多个定时器
- 以下事件中产生中断/DMA：
  - 更新事件：计数器上溢/ 下溢，计数器初始化（通过软件或内/外部触发）
  - 触发事件（计数器起始、停止、初始化或内/外触发计数）
  - 输入捕获
  - 输出比较
- 支持增量（正交）编码及霍尔电路进行定位
- 触发输入可对外部时钟管理

22. 3 结构框图

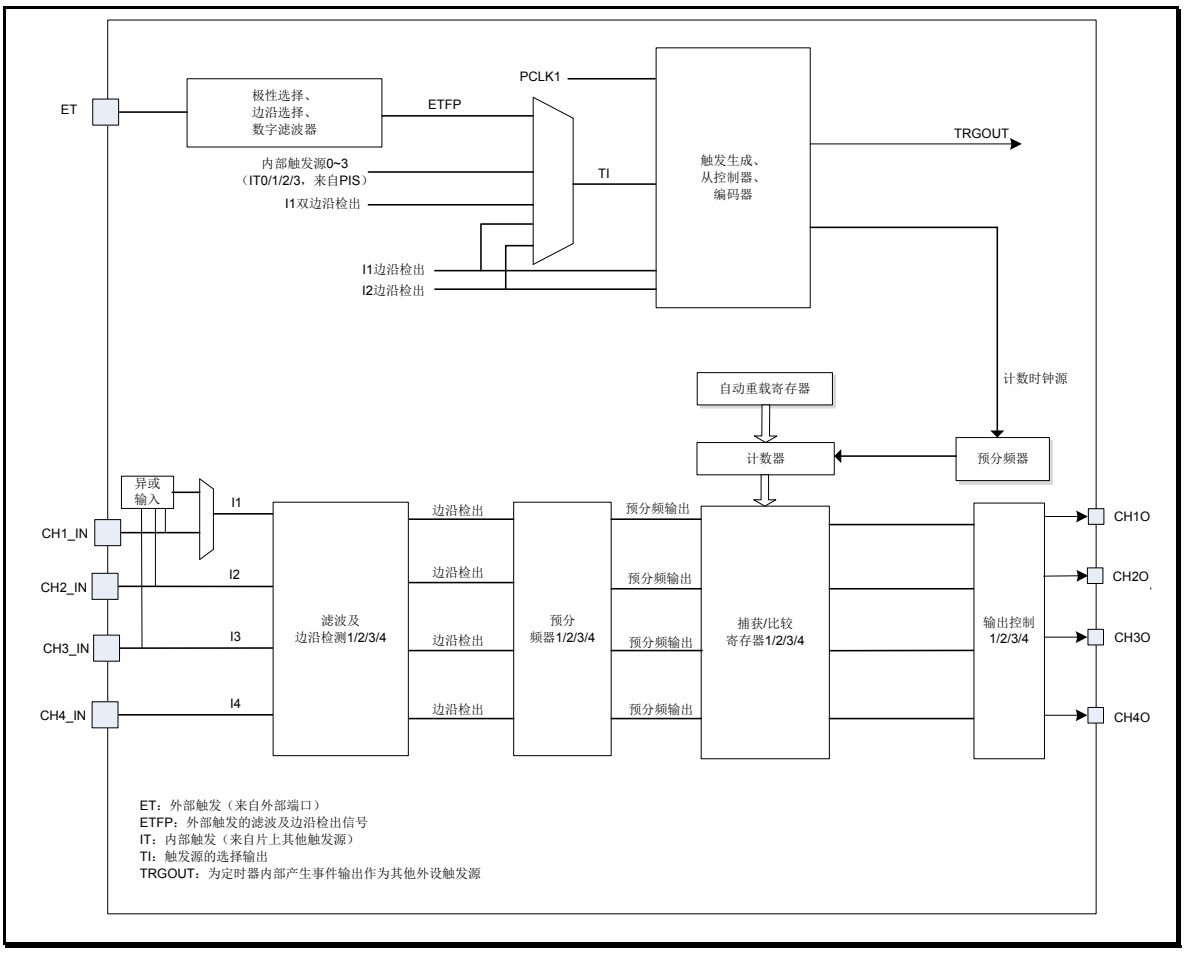


图 22-1　通用定时器结构框图

22. 4 功能描述

22. 4. 1 预分频器

定时器包含一个 32-bit 的计数器（GP32C4Tn_COUNT），计数时钟由预分频寄存器（GP32C4Tn_PRES）进行分频。计数周期由自动重载计数器（GP32C4Tn_AR）设定。

自动重载寄存器（GP32C4Tn_AR）是一个可缓存的寄存器。当 GP32C4Tn_CON1 寄存器的 ARPEN 位复位时，GP32C4Tn_AR 寄存器重载功能失效，GP32C4Tn_AR 就是有效寄存器；ARPEN 置位时，GP32C4Tn_AR 寄存器具有重载功能，产生更新事件（UEV）时，加载值（GP32C4Tn_AR 寄存器值）更新到影子寄存器后才生效。

当 GP32C4Tn_CON1 寄存器中 DISUE 位为 0 时，计数器计数上溢（或递减下溢）时会产生更新事件（UEV）。同样，软件方式也可产生更新事件。GP32C4Tn_CON1 寄存器的 CNTEN 置位时，计数器开始计数。

注：计数器在 CNTEN 位置位 1 个时钟周期后开始计数。

预分频器可对定时器工作时钟进行 GP32C4Tn_PRES 寄存器值+1 次分频。由于 GP32C4Tn_PRES 是一个可重载寄存器，因此，定时器工作时可以对该寄存器进行修改，修改值在下次更新事件（UEV）后有效。

下图给出了定时器运行过程中改变预分频值时计数器的计数情况。

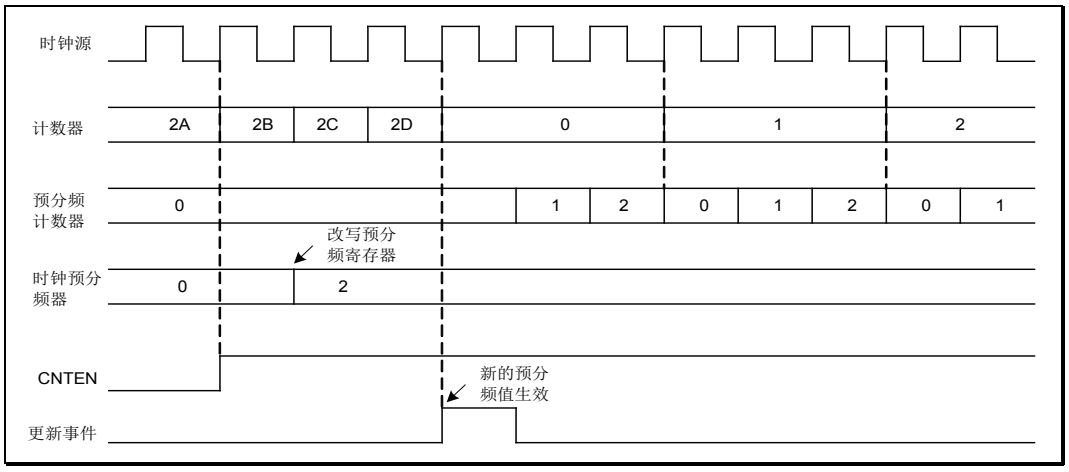


图 22-2 预分频值计数时序图

22. 4. 2 时钟源

计数器工作时钟可以可以选择内部时钟(INT_CLK)、外部时钟源 1（I1、I2）、外部时钟源 2（ET），内部触发输入（IT1、IT2、IT3、IT4）

22. 4. 2. 1 内部时钟源（INT_CLK）

若从模式控制器被关闭（GP32C4Tn_SMCON 寄存器内，SMODS= "000"），则 CNTEN，GP32C4Tn_CON1.DIRSEL 与 GP32C4Tn_SGE.SGU 位为实际控制位，这些位只能软件修改（SGU 位除外，仍硬件自动清除）。一旦 CNTEN 位被写为'1'，预分频器就由内部 INT_CLK 提供时钟。

下图给出了通常模式下控制电路和递增计数的情况，没有分频。

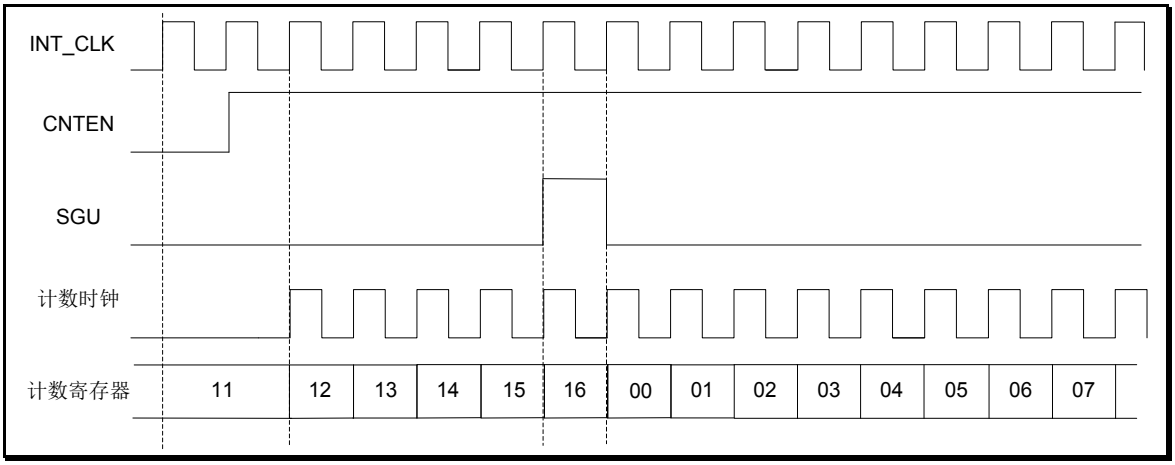


图 22-3 采用内部时钟计数

22. 4. 2. 2 外部时钟源 1

GP32C4Tn_SMCON 寄存器的 SMODS= "111"时，可选择外部时钟源 1。计数器可根据选定的上升沿或下降沿计数。

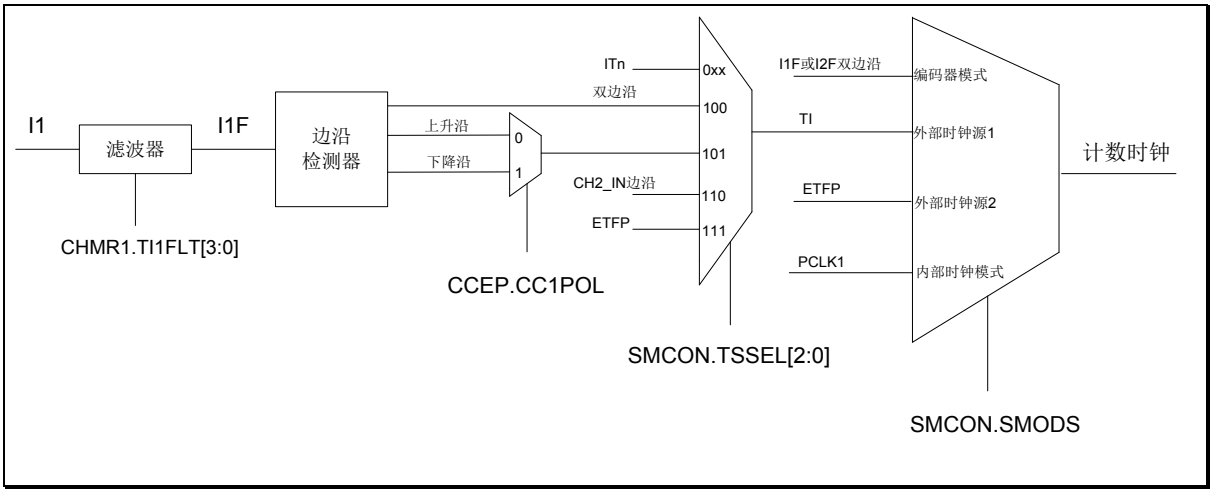


图 22-4 I1 外部时钟连接

配置计数器为外部时钟源 1，步骤如下：

11. GP32C4Tn_SMCON 寄存器中 SMODS = "111"，配置定时器外部时钟模式 1。
12. 设置 GP32C4Tn_SMCON 寄存器中的 TSSEL 选择外部时钟源。
13. 如外部时钟源为 I1，可配置 GP32C4Tn_CHMR1 寄存器 CC1SSEL = "01"，配置通道 1 检测 I1 输入的上升沿；设置 GP32C4Tn_CCEP 寄存器中 CC1POL = '0'，选择极性为上升沿。
14. 写 GP32C4Tn_CHMR1 寄存器的 I1FLT[3：0]位，配置输入滤波器时间（若没有滤波器需求，维持 I1FLT = "0000"）。
15. GP32C4Tn_CON1 寄存器中 CNTEN = '1'，使能计数器。

当 I1 上出现一次上升沿时，计数器计数一次且 TRGIF 标志位置位。

22. 4. 2. 3 外部时钟源 2

置位 GP32C4Tn_SMCON 寄存器的 ECM2EN 位选定外部时钟源 2。

计数器可对外部触发输入 ET 进行上升沿或下降沿计数。

下图给出了外部触发输入模块的概况。

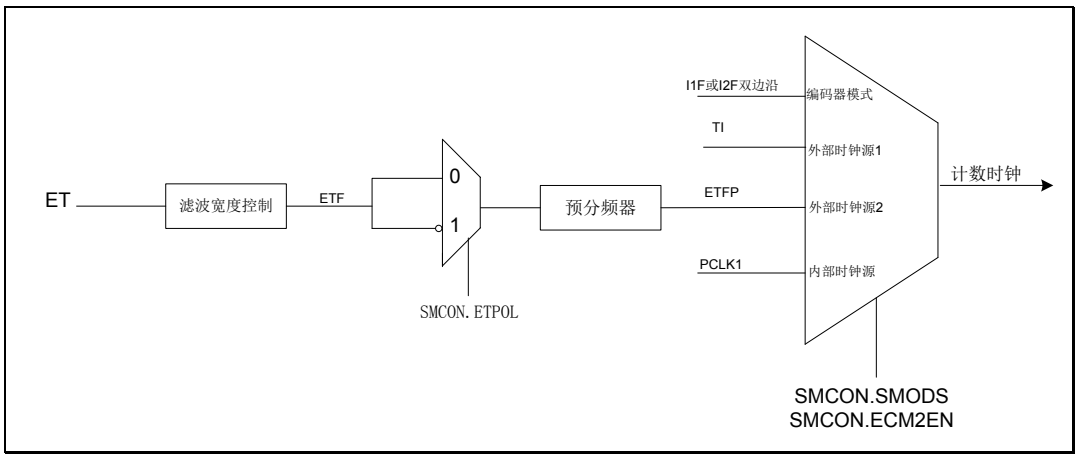


图 22-5 外部触发输入模块

配置计数器为外部时钟源 2，配置过程如下：

11. 设置 GP32C4Tn_SMCON 寄存器的 ETFLT[3：0]，配置输入滤波时间。
12. 设置 GP32C4Tn_SMCON 寄存器中 ETPSEL[1：0]，设置预分频器。
13. 设置 GP32C4Tn_SMCON 寄存器中 ETPOL，检测 ET 引脚上升沿或下降沿。
14. 设置 GP32C4Tn_SMCON 寄存器中 ECM2EN = '1'，使能外部时钟模式 2。
15. 设置 GP32C4Tn_CON1 寄存器的 CNTEN = '1'，使能计数器。

计数器每两个上升沿计一次数。

22. 4. 2. 4 内部触发输入（ITn）

当 GP32C4Tn_SMCON 寄存器的 SMODS= "111"，选定内部触发模式。计数器根据选定的内部输入端的上升或下降沿计数。

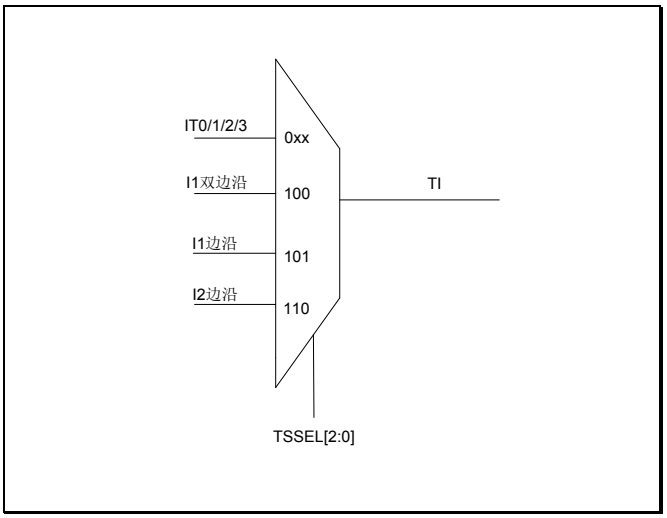


图 22-6 ITn 外部时钟连接

配置计数器在 ITn 输入端的上升沿递增计数，步骤如下：

7. GP32C4Tn_SMCON 寄存器中 SMODS = "111"，配置外部时钟模式 1。
8. GP32C4Tn_SMCON 寄存器的 TSSEL = "0xx"，选定 ITn 作为触发输入源。
9. GP32C4Tn_CON1 寄存器的 CNTEN = '1'，使能计数器。

ITn 产生上升沿时，计数器计数一次。ITn 上升沿与实际时钟间的延时，取决于 ITn 输入的再同步电路，一般为 2~3 个定时器模块时钟周期。

22. 4. 3 计数器模式

22. 4. 3. 1 递增计数模式

在递增计数模式下，当 GP32C4Tn_REPAR 寄存器值为 0 时，计数器从 0 开始递增，直至 GP32C4Tn_AR 寄存器值；然后从 0 重新开始计数并产生一个更新事件（UEV）。当 GP32C4Tn_REPAR 寄存器不为 0 时，则在 GP32C4Tn_REPAR+1 次计数后产生更新事件。

当有更新事件（UEV）产生时，预装载寄存器会更新到影子寄存器，更新标志位（GP32C4Tn_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）：

- ◇ 更新 GP32C4Tn_REPAR 寄存器的值到影子寄存器
- ◇ 更新 GP32C4Tn_AR 寄存器的值到影子寄存器
- ◇ 更新 GP32C4Tn_PRES 寄存器的值到影子寄存器

下图为 GP32C4Tn_REPAR=0x0，GP32C4Tn_AR = 0x16，预分频设为 2 分频时的计数器时序。

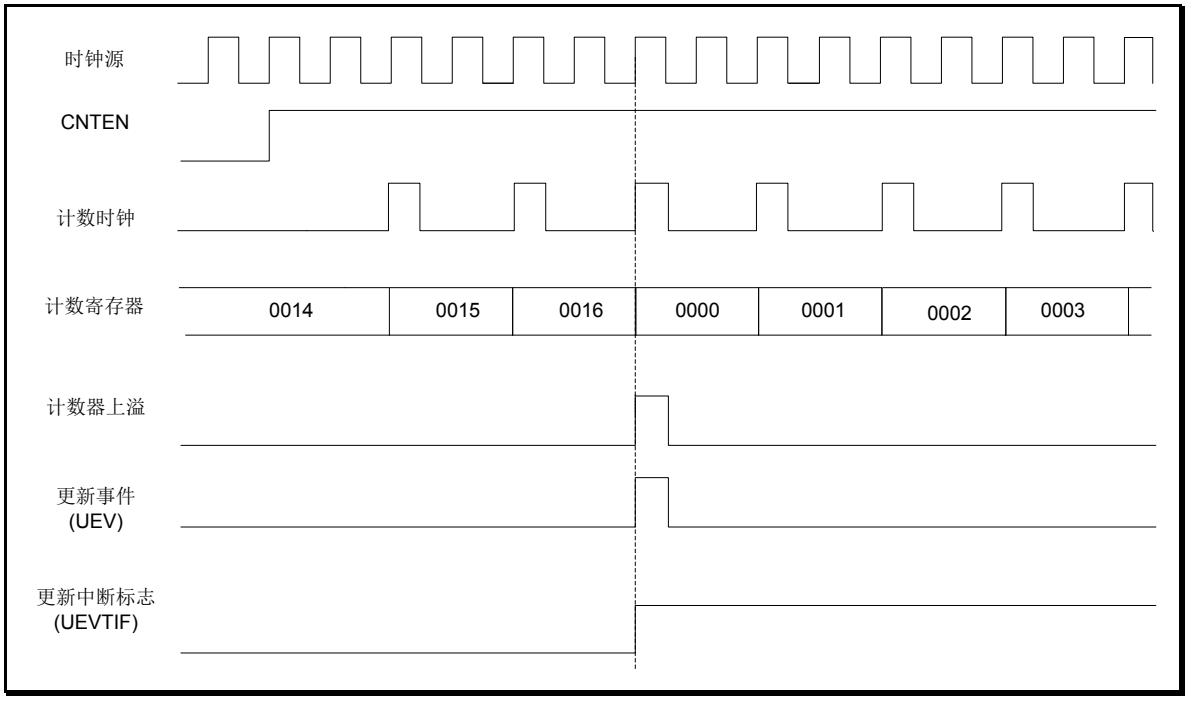


图 22-7 计数器时序图，内部时钟除以 1

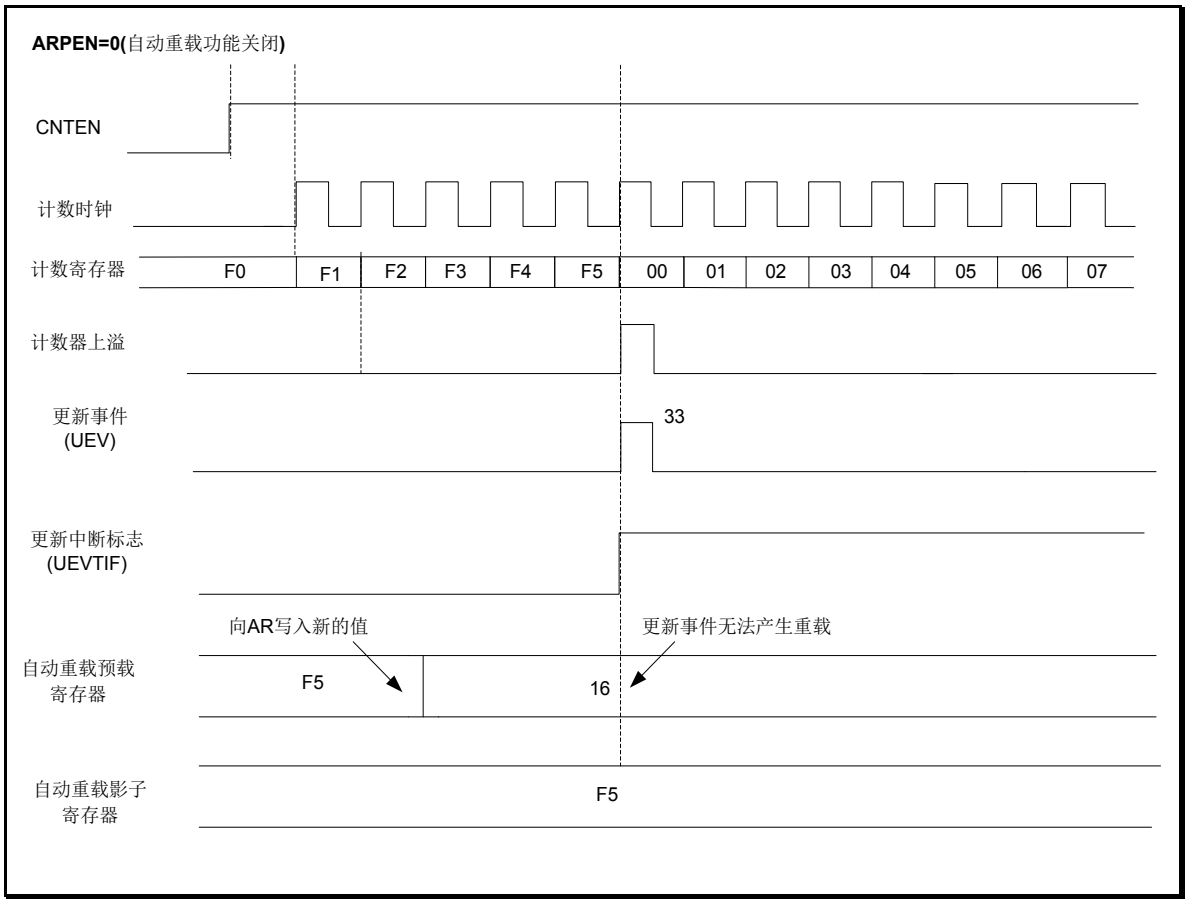


图 22-8　当 ARPEN=0 时计数器时序图

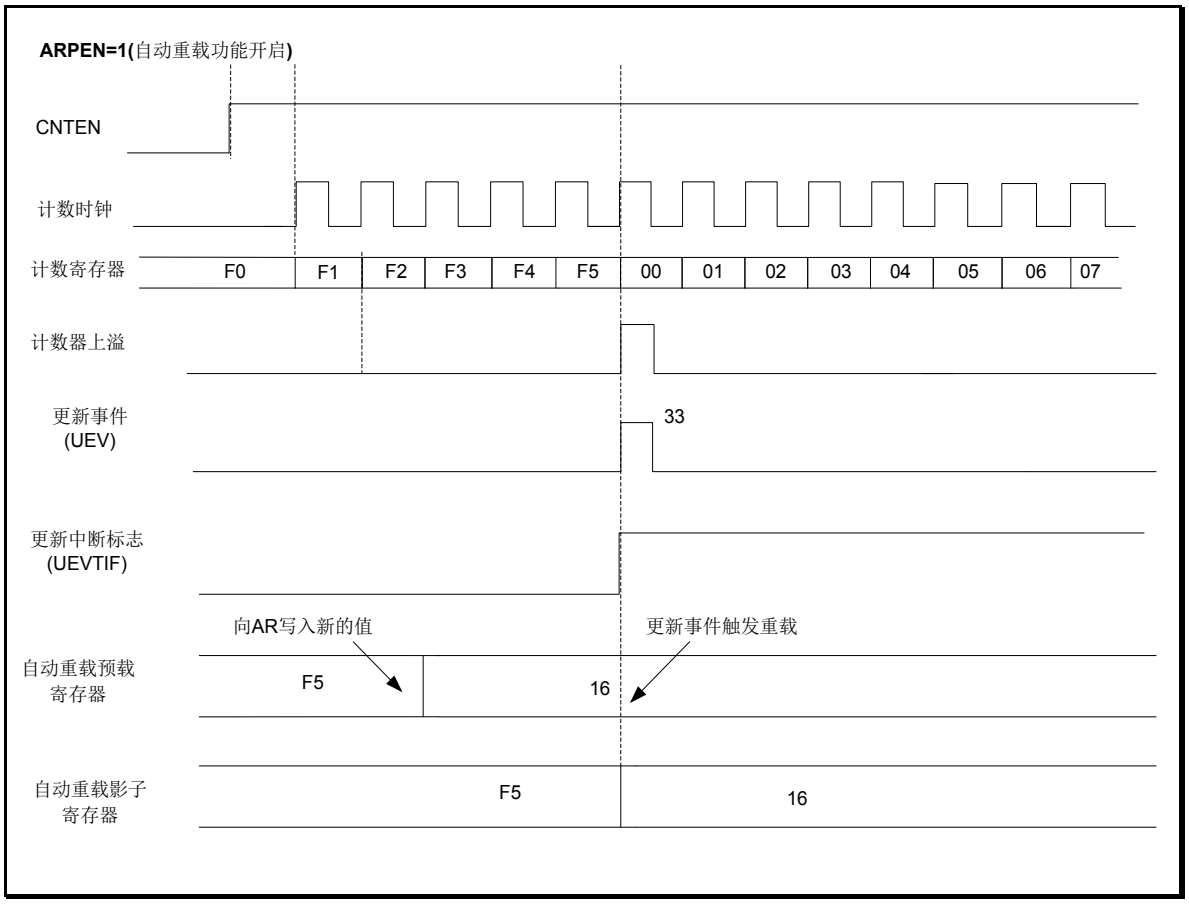


图 22-9　当 ARPEN=1 时计数器时序图

22. 4. 3. 2　　递减计数模式

在递减计数模式下，当 GP32C4Tn_REPAR 寄存器值为 0 时，计数器从 GP32C4Tn_AR 寄存器值开始递减至 0；然后重复递减并产生更新事件（UEV）。当 GP32C4Tn_REPAR 寄存器不为 0 时，则在 GP32C4Tn_REPAR+1 次后产生更新事件。

置位 GP32C4Tn_SGE 寄存器中的 SGU 位（通过软件或使用从机模式控制器）同样会产生更新事件。

当有更新事件（UEV）产生时，预载寄存器值会更新到影子寄存器，更新标志位（GP32C4Tn_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）。

下图为 GP32C4Tn_REPAR=0x0，GP32C4Tn_AR = 0x27，预分频设为 1 分频时的计数器时序。

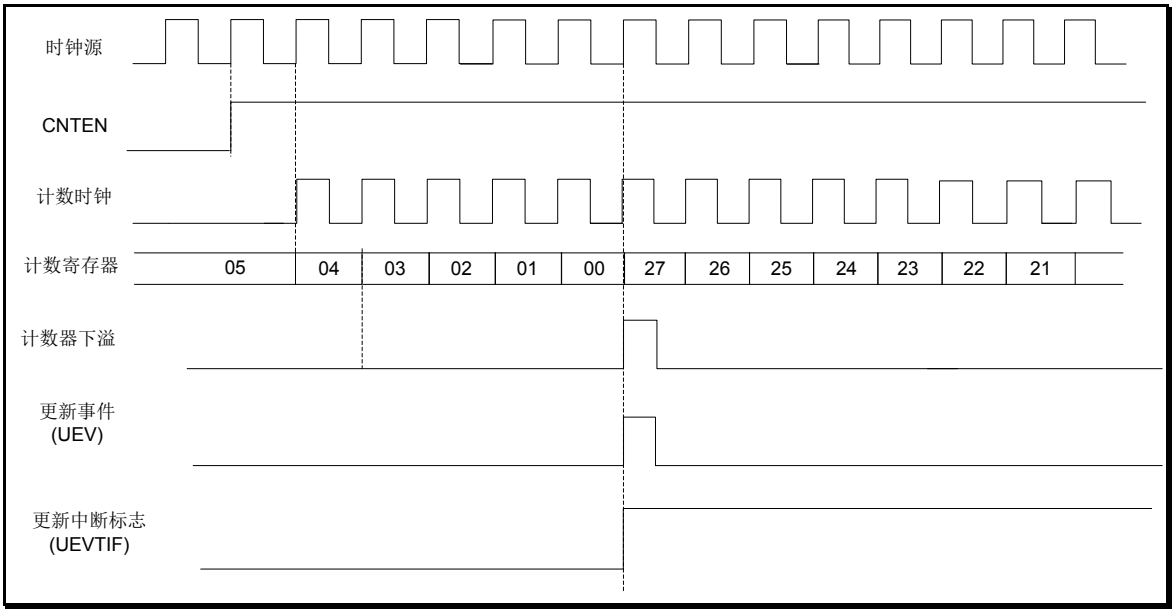


图 22-10　定时器递减计数时序图

22. 4. 3. 3　　中心对齐模式

当 GP32C4Tn_CON1 寄存器的 CMSEL 位的值不等于"00"时，定时器工作在中心对齐模式。定时器配置为中心对齐模式时，计数器先从 0 开始递增至 GP32C4Tn_AR 寄存器值-1，并产生更新事件（UEV）；接着计数器从 GP32C4Tn_AR 寄存器值递减至 1，并产生下溢事件。如此循环计数。在计数器递减计数（中心对称模式 1，CMSEL="01"）、计数器递增计数（中心对称模式 2，CMSEL="10"）、计数器递增和递减计数（中心对称模式 3，CMSEL="11"）模式下，当通道配置为输出模式时，其输出比较中断标志位会置位。

在中心对齐模式下，GP32C4Tn_CON1 寄存器的 DIRSEL 位无法进行写操作，该位由硬件自动更新指示当前计数方向。

计数上溢、下溢或者置位 GP32C4Tn_SGE 寄存器的 SGU 位（通过软件或使用从模式控制器）都会产生更新事件。因此，计数器和预分频器都会从 0 开始计数。

软件置位 GP32C4Tn_CON1 寄存器中的 DISUE 位可关闭更新事件（UEV）的产生。更新事件（UEV）关闭时，可避免向预载寄存器写新值时更新影子寄存器。DISUE 复位之前都不会产生更新事件。而在正常产生更新事件时，计数器仍然从 0 开始，同样预分频计数也是从 0 开始（但预分频值没有改变）。此外，若置位 GP32C4Tn_CON1 寄存器中的 UERSEL 位（更新请求选择），置位 SGU 位时会产生一次更新事件（UEV），但 UEVTIF 标志位不会置位（因此，不会触发中断或 DMA 请求）。这就避免了在捕获事件时，清除计数器值时产生更新和捕获中断。

当有更新事件（UEV）产生时，预载寄存器值会更新到影子寄存器，更新标志位（GP32C4Tn_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）。

注：若更新源为计数上溢，自动重载会在计数器重载前更新。因此，下一周期即为预期值（计数器载入新值）。

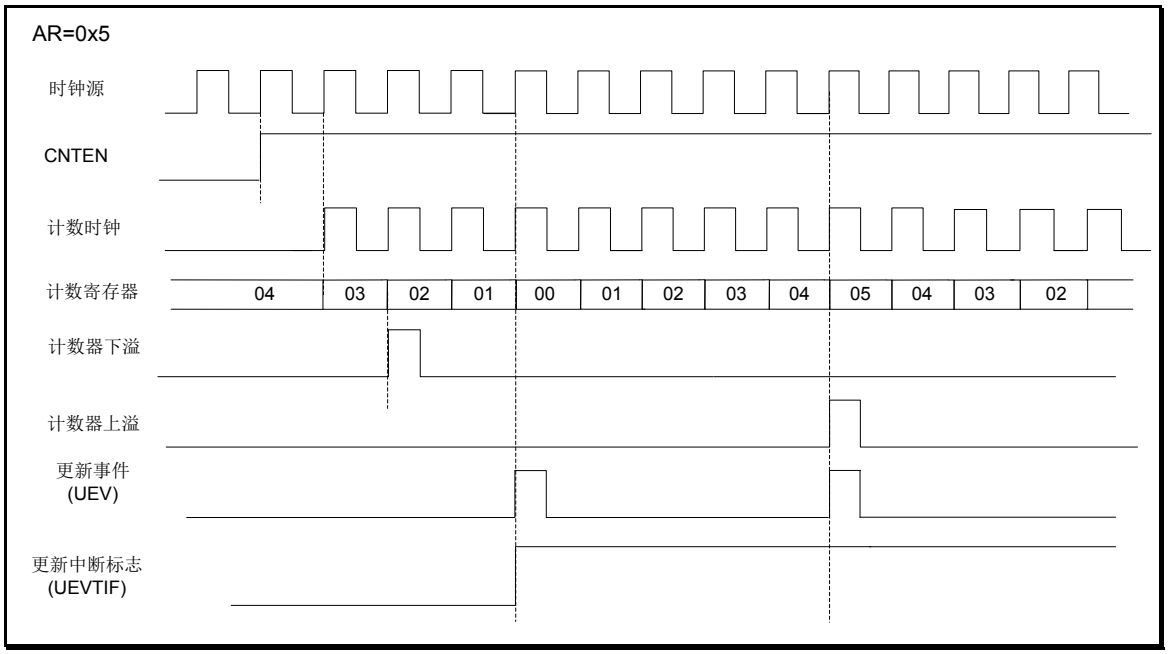


图 22-11　增减计数器时序图

22. 4. 4 捕获/比较通道

以下各图为捕获/比较通道的概述。

输入电路对 In 输入端的信号进行采样，产生一个经过滤波的信号 InF。之后，一个可极性选择的边沿检测器产生 In 边沿检出信号，该信号可作为从模式控制器的触发输入或作为捕获控制命令，且信号经过分频后进入捕获寄存器。

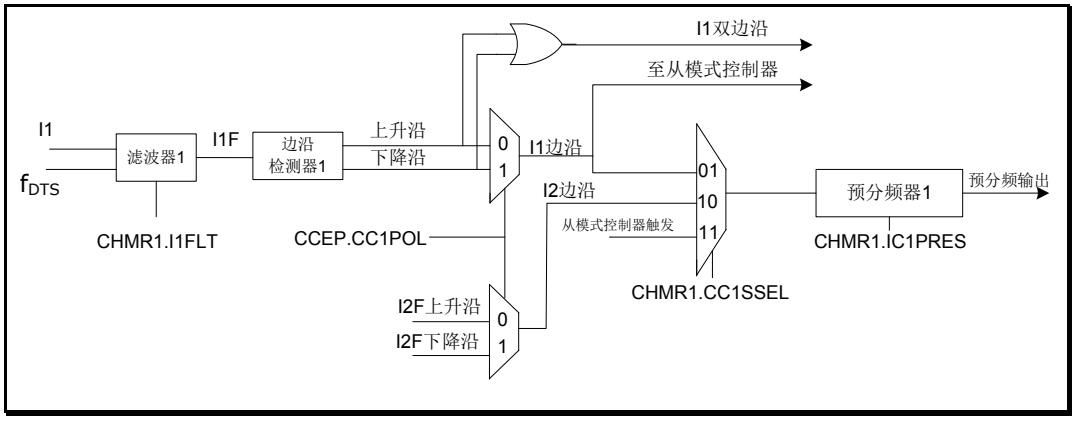


图 22-12 捕获/比较通道

输出部分产生一个中间波形（高有效）作为基准，在输出末端决定最终输出信号的极性。

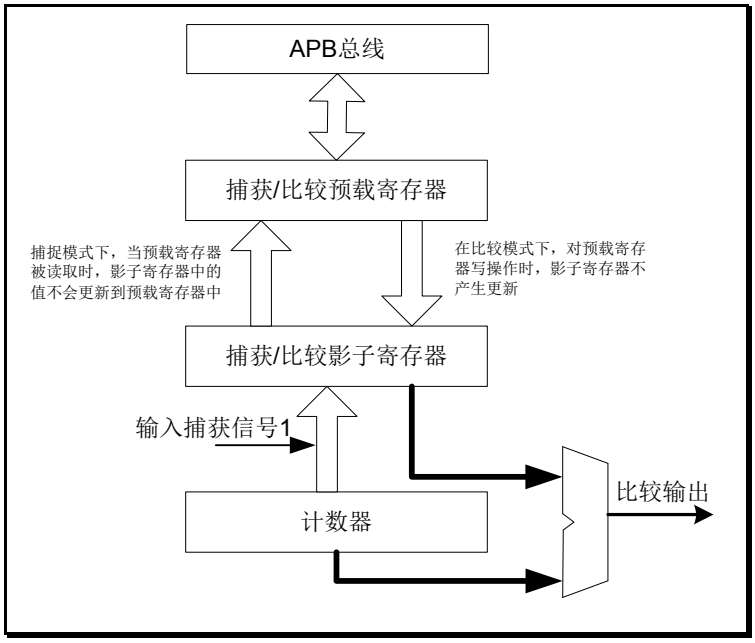


图 22-13 捕获/比较信道 1 主电路

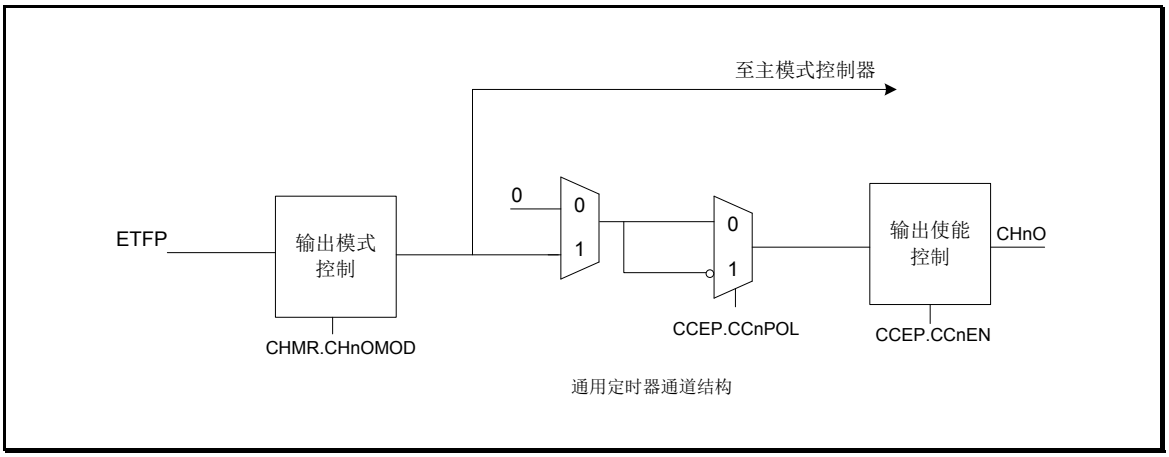


图 22-14 捕获/比较通道的输出阶段

22. 4. 5 输入捕获模式

在输入捕获模式下，当检测到 In 上相应信号变化时，计数器的值就会被锁存到捕获/比较寄存器（GP32C4Tn_CCVALn）寄存器中。当捕获发生时，相应的 CHnCCIF 标志位（GP32C4Tn_RIF）会置位，同时会触发中断或 DMA（如果使能）请求。若发生捕获时，CHnCCIF 标志位已经置位，则过捕获 CHnOVIF 标志位（GP32C4Tn_RIF）置位。软件写'0'或读取 GP32C4Tn_CCVALn 寄存器中的捕获值都可以复位 CHnCCIF 标志位。对 CHnOVIF 位写'0'可清空该标志位。

以下为以 I1 输入上升沿作为捕获输入时的流程：

13. 选择有效输入端：GP32C4Tn_CCVAL1 必须连接到 I1 输入端，因此需将 GP32C4Tn_CHMR1 寄存器中的 CC1SSEL 位写"01"。只要 CC1SSEL 不为"00"，通道被配置为输入且 GP32C4Tn_CCVAL1 寄存器为只读。
14. 根据定时器连接的输入信号，配置输入滤波器的持续时间。当输入信号翻转时，前 5 个内部时钟信号内信号是不稳定的，因此必须配置滤波器的时间大于 5 个时钟周期。当 I1 检测到新的电平，连续 8 次采样可确认电平变化有效。
15. 选择 I1 信道的有效边沿变换。GP32C4Tn_CCEP 寄存器中的 CC1POL 写'0'（上升沿）。
16. 配置输入预分频器。
17. 置位 GP32C4Tn_CCEP 寄存器中的 CC1EN 位，使能捕获计数器的值到捕获寄存器。
18. 如有需要，置位 GP32C4Tn_IER 寄存器中的 CC1IT 位，使能中断请求。置位 GP32C4Tn_DMAEN 寄存器中的 CC1DMA 位，使能 DMA 请求。

当发生输入捕获时：

9. 有效边沿产生，GP32C4Tn_CCVAL1 寄存器获取计数器的值。
10. CH1CCIF 标志位置位（中断标志）。若至少 2 个连续的捕获发生，但标志位没有及时清除，则 CH1OVIF 也会置位。
11. 中断的产生取决于 GP32C4Tn_IVS 寄存器中的 CC1IT 位。
12. DMA 请求的产生取决于 CC1DMA。

为了处理捕获溢出，建议在读出捕获溢出标志位之前先读取捕获数据。这可以避免丢失在读取捕获标志位之后与读取数据之前可能重复产生的捕获信息。

注：捕获中断请求可由软件设置 GP32C4Tn_SGE 寄存器中 SGCCnE 位产生。

22. 4. 5. 1 PWM输入模式

测量 I1 上 PWM 信号的周期和占空比的过程如下：

1. 为 GP32C4Tn_CCVAL1 选择有效的输入：GP32C4Tn_CHMR1 寄存器中的 CC1SSEL 位写"01"（I1 被选择）。

2. 为 I1 边沿检出选择有效的极性（用于捕获数据到 GP32C4Tn_CCVAL1 寄存器和计数器清零）：CC1POL 位写'0'（上升沿有效）。

3. 为 GP32C4Tn_CCVAL2 选择有效输入：GP32C4Tn_CHMR1 寄存器的 CC2SSEL 位写"10"（I1 被选择）。

4. 为 I1 边沿检出选择有效极性（用于捕获数据到 GP32C4Tn_CCVAL2）：CC2POL 位写'1'（下降沿有效）。

5. 选择有效的触发输入：GP32C4Tn_SMCON 寄存器的 TSSEL 位写"101"（I1 边沿检出被选择）。

6. 配置从机模式控制器为复位模式：GP32C4Tn_SMCON 寄存器的 SMODS 位写"100"。

7. 使能捕获：GP32C4Tn_CCEP 寄存器的 CC1EN 位和 CC2EN 位写'1'。

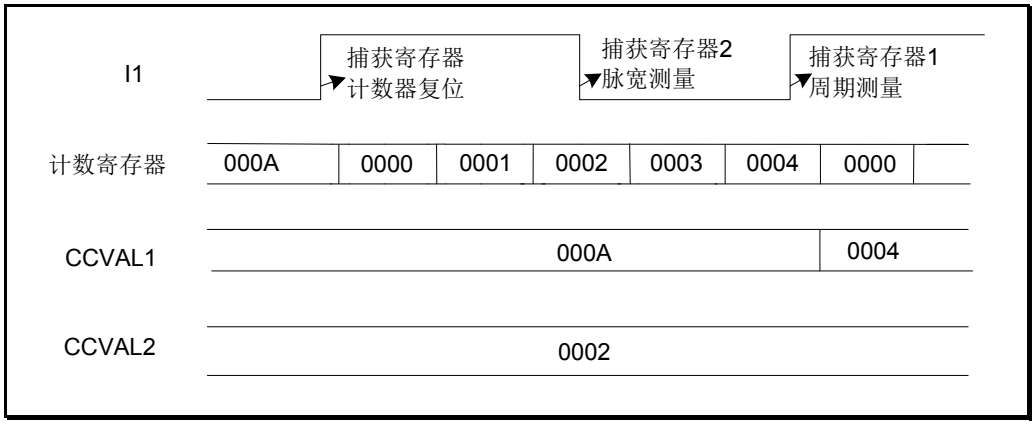


图 22-15 PWM 输入模式时序

22. 4. 6 PWM模式

脉宽调制模式可以产生一个 GP32C4Tn_AR 寄存器值确定频率，GP32C4Tn_CCVALn 寄存器值确定占空比的信号。

每个通道的 PWM 模式是相互独立的（每个 CHnO 输出一个 PWM），GP32C4Tn_CHMRn 寄存器的 CHnOMOD 位写"110"（PWM 模式 1）或写"111"（PWM 模式 2）。必须通过置位 GP32C4Tn_CHMRn 寄存器的 CHnOPREN 位来使能相应的预载寄存器，最后还需置位 GP32C4Tn_CON1 寄存器的 ARPEN 位来使能自动重装预载功能。

只有当更新事件发生时预载寄存器中的值才会传到影子寄存器，因此，在使能计数前，必须通过置位 GP32C4Tn_SGE 寄存器的 SGU 位来初始化所有的寄存器。

CHnO 的极性可通过 GP32C4Tn_CCEP 寄存器的 CCnPOL 位配置，有效极性可配置为高或低。CHnO 的输出使能由 CCnEN 位（GP32C4Tn_CCEP 寄存器）控制。

在 PWM 模式（1 或 2）中，GP32C4Tn_COUNT 和 GP32C4Tn_CCVALn 寄存器的值会持续的比较，确定 GP32C4Tn_CCVALn <= GP32C4Tn_COUNT 或 GP32C4Tn_CCVALn >= GP32C4Tn_COUNT（取决于计数器的计数方向）。

定时器产生 PWM 波形是边沿对齐或中心对齐，取决于 GP32C4Tn_CON1 寄存器的 CMSEL 位。

22. 4. 6. 1 PWM边沿对齐模式

递增计数配置

当 GP32C4Tn_CON1 寄存器的 DIRSEL 位为低时，计数器递增计数。

下图给出了 GP32C4Tn_AR = 8 时的边沿对齐 PWM 波形。

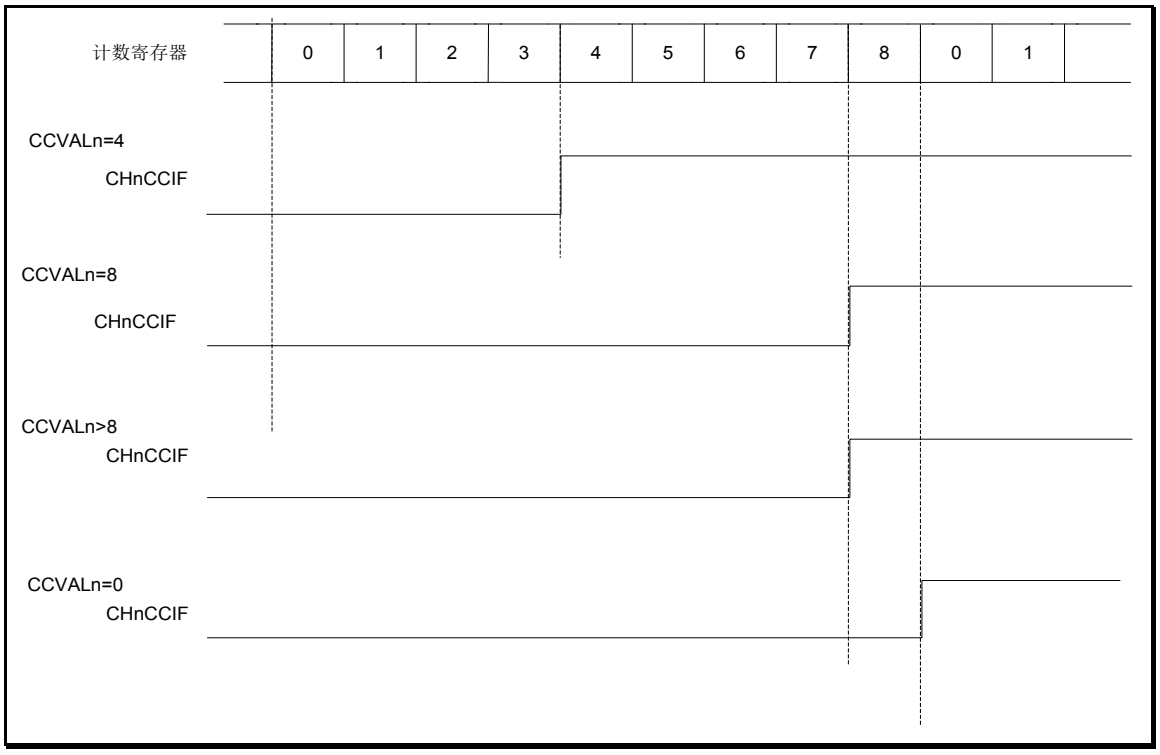


图 22-16 边沿对齐 PWM 波形（AR=8）

递减计数配置

当 GP32C4Tn_CON1 寄存器的 DIRSEL 位为高时，计数器递减计数。

22. 4. 6. 2 PWM中心对齐模式

当 GP32C4Tn_CON1 寄存器中的 CMSEL 位不为"00"时，中心对齐模式有效。计数器是递增、递减计数分别置比较标志位或递增递减都置比较标志位，取决于 CMSEL 位的配置。GP32C4Tn_CON1 寄存器的方向位（DIRSEL）是由硬件更新的，软件无法修改。

下图为中心对齐方式产生的 PWM 波形的例子：

◇ GP32C4Tn_AR=0x3F，GP32C4Tn_CCVALn=0x3D

◇ PWM 模式 1

- GP32C4Tn_CON1 寄存器的 CMSEL= "01"，在中心对齐模式 1 下，计数器向下计数时会置位比较标志位。

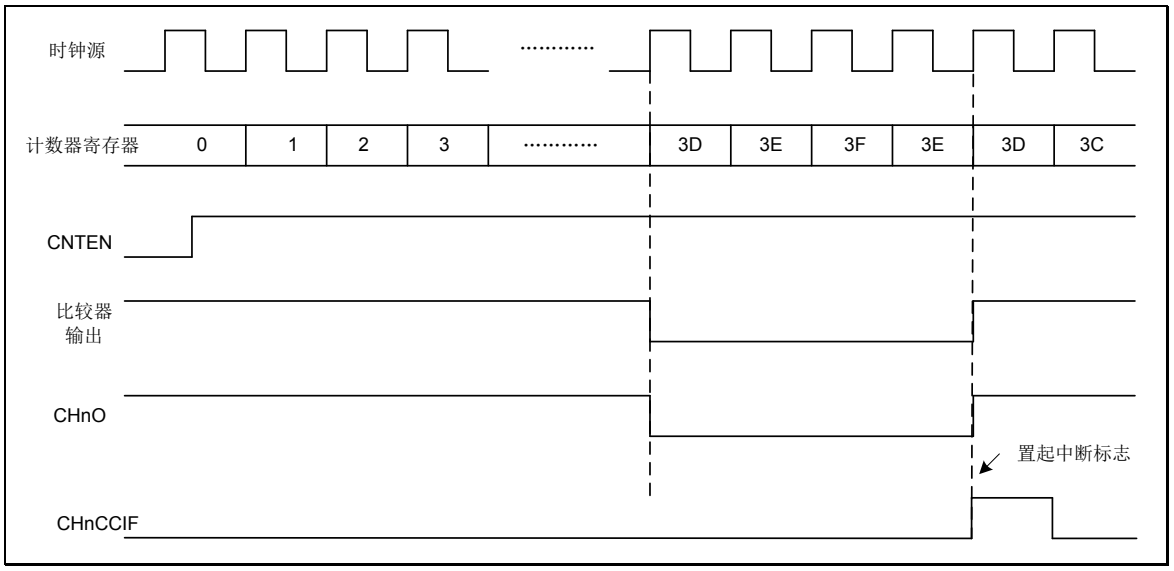


图 22-17 中心对齐 PWM 波形（AR=0x3F）

中心对齐模式的使用技巧：

◇ 当进入中心对齐模式后，当前递增或递减配置生效。计数器递增或递减计数取决于 GP32C4Tn_CON1 寄存器的 DIRSEL 位的值。

◇ 计数器在中心对齐模式下运行时，对计数器写操作可能导致不可预知的结果。特别是：

- 若向计数器入的值大于自动重载值（GP32C4Tn_COUNT>GP32C4Tn_AR），计数方向不更新。例如，如果计数器递增计数，写入值后仍旧递增计数。
- 若向计数器写 0 或 GP32C4Tn_AR 中的重载值，则计数方向更新，但并没有产生 UEV。

◇ 使用中心对齐模式最安全的方式是计数器开始计数前通过软件产生更新事件（置位 GP32C4Tn_SGE 寄存器中的 SGU 位）且在计数器运行过程中不对计数器写值。

22. 4. 7 输出比较模式

该功能用于控制输出波形或指示周期时间的结束。

当捕获/比较寄存器和计数器值匹配时，输出比较功能：

- ◇ 输出比较模式（GP32C4Tn_CHMRn 寄存器中的 CHnOMOD 位）和输出极性（GP32C4Tn_CCEP 寄存器中的 CCnPOL 位）的配置值输出到对应的引脚上。
- ◇ 中断状态寄存器中的标志位置位（GP32C4Tn_RIF 寄存器的 CHnCCIF 位）。
- ◇ 若相应的中断掩码置位，则产生中断（GP32C4Tn_IER 寄存器的 CCnIT 位）。
- ◇ 若相应的使能位置位（GP32C4Tn_DMAEN 寄存器的 CCnDMA 位，GP32C4Tn_CON2 寄存器的 CCDMASEL 位用于 DMA 请求的选择），则发送 DMA 请求。

GP32C4Tn_CHMRn 寄存器中 CHnOPREN 位的值可决定 GP32C4Tn_CCVALn 寄存器是否带有预装载寄存器。

在输出比较模式中，更新事件 UEV 对 CHnO 的输出没有影响。计时分辨率为计数器的一次计数。输出比较模式同样可以用来输出单个脉冲（单脉冲模式）。

输出比较的配置过程：

6. 选定计数器时钟（内部，外部，预分频）。
7. GP32C4Tn_AR 与 GP32C4Tn_CCVALn 寄存器中写入预期值。
8. 若需要产生中断请求，置位 GP32C4Tn_IER 寄存器中的 CCnIT 位。
9. 选择输出模式，例如：

   - CHnOMOD = "011"，当 CNTV 与 CCRVn 匹配时，CHnO 输出翻转。
   - CHnOPREN = '0'，关闭预载寄存器。
   - CCnPOL = '0'，选择有效极性为高。
   - CCnEN = '1'，使能输出。
10. GP32C4Tn_CON1 寄存器中的 CNTEN 位置位，使能计数器。

通过配置 GP32C4Tn_CHMR1 寄存器的 CHnOPREN 位 可将 GP32C4Tn_CCVALn 配置为是否带预装载寄存器。。通过软件方式，GP32C4Tn_CCVALn 寄存器的值可随时更新控制输出波形。

22. 4. 7. 1 外部事件清除比较输出

ETFP 输入端（GP32C4Tn_CHMRn 寄存器的 CHnOCLREN 位写'1'）上的高电平，可将给定通道的比较输出信号拉低。在下次更新事件（UEV）发生前，比较输出会一直保持为低。该功能只能应用在输出比较和 PWM 模式中，强制输出模式中不起作用。

ET 信号可以接到电流控制比较器的输出端。该例中，ET 须按如下流程配置：

a) 外部触发预分频器应该关闭：GP32C4Tn_SMCON 寄存器的 ETPSEL[1：0]位应该写"00"

b) 外部时钟源 2 关闭：GP32C4Tn_SMCON 寄存器的 ECM2EN 位写'0'

c) 外部触发极性（ETPOL）和外部触发滤波器（ETFLT）可根据用户需要配置

22. 4. 8 单脉冲模式

单脉冲模式下，响应某个触发后，定时器的输出通道在可配置的延迟时间后产生一个脉冲，脉冲长度可配。从模式控制器可控制计数器的启动。脉冲波形可在输出比较模式和 PWM 模式下产生。置位 GP32C4Tn_CON1 寄存器的 SPMEN 位可选择单脉冲模式。计数器会在下次更新事件 UEV 产生时自动停止。

只有比较值不同于计数器初始值时，单脉冲才可以正确的产生。计数器开始计数前（定时器等待触发），必须如下配置：

- ◇ 递增计数：CNT < CCVALn≤ AR（特别地，0 < CCVALn）
- ◇ 递减计数：CNT > CCVALn

基于 PWM 模式设置单脉冲输出波形的步骤如下：

- ◇ 设置 GP32C4Tn_CHMRn 寄存器的 CHnOMOD 位，选择 PWM 模式 1 或 2；
- ◇ 设置 GP32C4Tn_CCEP 寄存器的 CCnPOL 位，选择通道端口 CHnO 的输出极性；
- ◇ 设置 GP32C4Tn_CON1 寄存器的 DIRSEL，CMSEL，SPMEN 位，配置为递增或递减计数，PWM 普通波形模式，单脉冲模式使能；
- ◇ 设置 GP32C4Tn_CHMR1 寄存器的 CH1OPREN =1，GP32C4Tn_CON1 寄存器的 ARPEN =1，使能比较寄存器和计数重载寄存器的缓冲功能（也可以根据实际情况不使能缓冲）；
- ◇ 设置 GP32C4Tn_CCVALn 寄存器和 GP32C4Tn_AR 寄存器，配置单脉冲输出延时和脉宽时间；
- ◇ 设置 GP32C4Tn_SGE 寄存器的 SGU=1 来产生一个更新事件；
- ◇ 设置 GP32C4Tn_CON1 寄存器的 CNTEN=1 来启动计数器，也可以在触发模式下，通过外部触发输入信号来触发硬件自动设置 CNTEN=1。

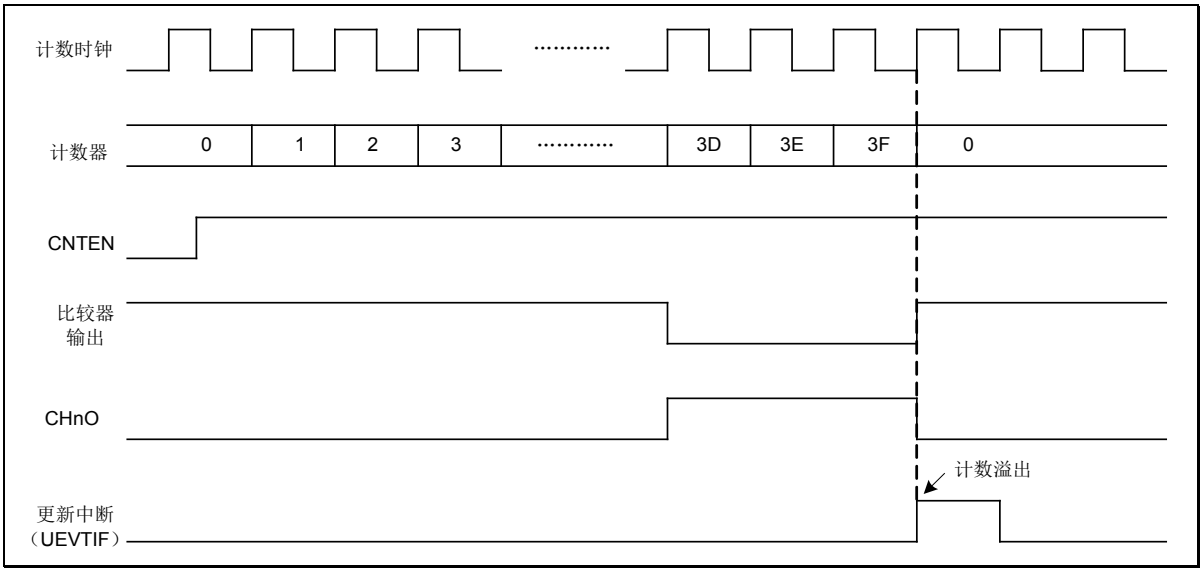


图 22-18 单脉冲模式

22. 4. 9 编码器接口模式

编码器接口模式的三种配置：若计数器只根据 I2 上的边沿计数，则 GP32C4Tn_SMCON 寄存器中的 SMODS = "001"；若计数器只根据 I1 上的边沿计数，则 GP32C4Tn_SMCON 寄存器中的 SMODS = "010"；若计数器同时根据 I1 和 I2 上的边沿计数，则 GP32C4Tn_SMCON 寄存器中的 SMODS = "011"。

配置 GP32C4Tn_CCEP 寄存器中的 CC1POL 和 CC2POL 位的值可选择 I1 和 I2 的极性。如果需要，也可以配置输入滤波器。

CH1_IN 和 CH2_IN 端口作为增量编码器的接口。当计数器使能时，计数器根据 I1 或 I2 上滤波后的有效电平变化时钟计数。I1 和 I2 滤波后的有效信号顺序会产生计数脉冲及方向信号。计数器是递增或递减计数由信号的跳变顺序决定，GP32C4Tn_CON1 寄存器中的 DIRSEL 计数方向位由自动硬件更新。

编码器接口模式的工作方式类似于一个带有方向选择的外部时钟。计数器在 0 到 GP32C4Tn_AR 寄存器中的自动重载值之间连续计数。因此，必须在开始计数前配置 GP32C4Tn_AR 寄存器。同样的，捕获器、预分频器、重复计数器、触发输出的特性正常工作。设定编码模式和选择外部时钟源 2 不兼容，不可以同时选择。

该模式下，计数器会根据增量式编码器的速度和方向自动修改，计数器的值反应的是编码器的位置。计数方向对应着连接传感器的旋转方向。

下表列出了所有的可能组合，假设 I1 和 I2 不同时变换。

有效边沿	有效边沿相对信号的电平（I1 滤波信号对应 I2,I2 滤波信号对应 I1）	I1 滤波信号边沿		I2 滤波信号边沿	
		上升	下降	上升	下降
仅在 I1 计数	高	下降	上升	不计数	不计数
	低	上升	下降	不计数	不计数
仅在 I2 计数	高	不计数	不计数	上升	下降
	低	不计数	不计数	下降	上升
在 I1 和 I2 上计数	高	下降	上升	上升	下降
	低	上升	下降	下降	上升

表 22-1 计数方向与编码器信号的关系

外部增量编码器可直接与 MCU 连接，无需外部逻辑接口逻辑。而比较器通常用于将编码器的差分输出转换为数字信号，这极大地增加了抗噪声能力。编码器的第三个输出端用于指示机械零点，可以连接到外部中断输入引脚以触发一次计数复位。

下图给出了计数器操作的例子，给出了计数信号了产生和方向控制。同样给出了选择双边沿时，输入抖动如何被补偿。输入抖动可能发生在传感器靠近切换点处。

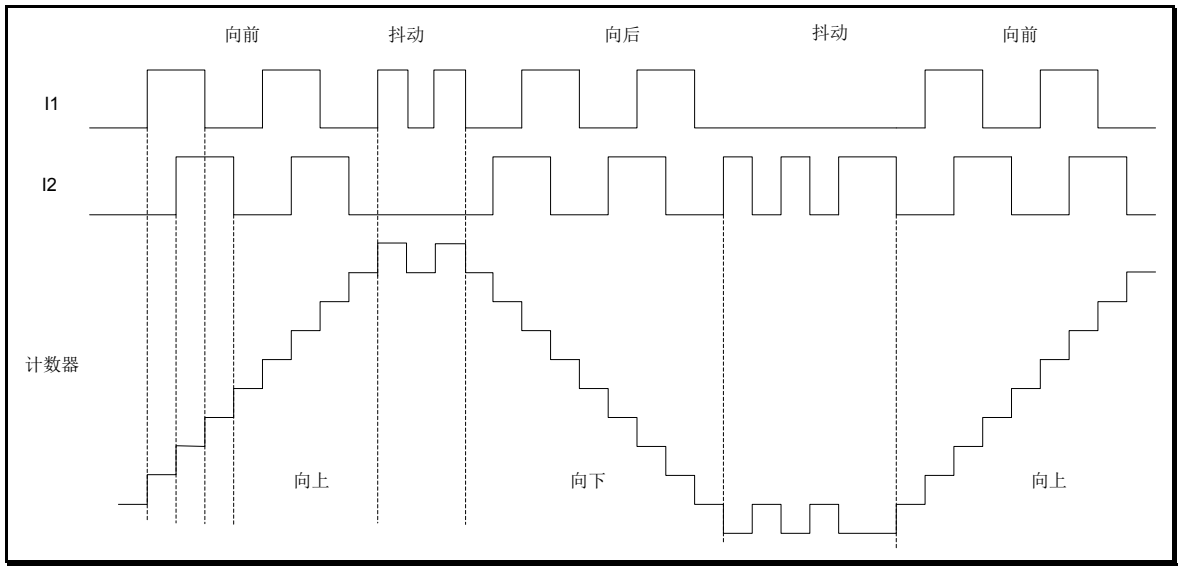


图 22-19　编码器接口模式下的计数操作

下图给出了计数器在 I1 滤波信号极性反相时的计数过程（除了 CC1POL = '1'，其他配置与上面一致）

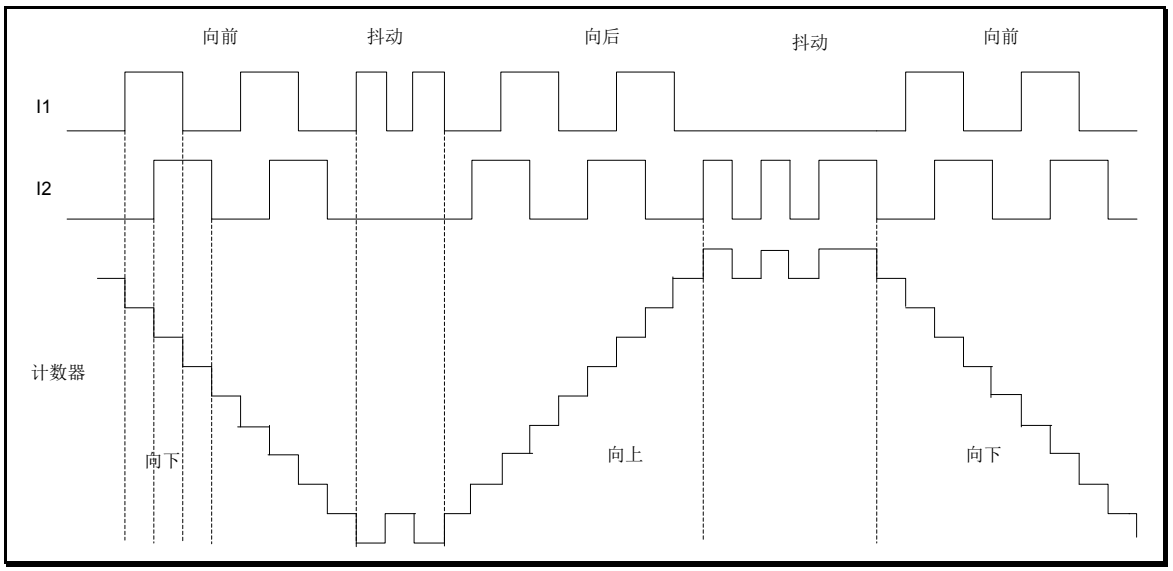


图 22-20　滤波后极性反相时编码器接口

当配置为编码器接口模式时，定时器可提供传感器的当前位置信息。配置一个额外定时器为捕获模式，用于测量两个编码器事件的间隔，根据间隔时常获取动态信息（速度、加速度、减速度）。编码器用于指示机械零点的输出就是此用处。根据编码器两个事件间隔，可以周期性的读取计数器的值。如果允许，可以将计数器值锁存到第三个输入捕获寄存器（捕获信号必须是周期性的且可由其它定时器产生）。条件允许时，可通过实时时钟产生 DMA 请求的方式读取计数器值。

22. 4. 10 输入异或功能

通过 GP32C4Tn_CON2 寄存器中 I1FSEL 位，可将通道 1 的输入滤波器连接到 XOR 门的输出端，XOR 门联合了 CH1_IN、CH2_IN 和 CH3_IN 三个输入引脚。

XOR 输出用于定时器的所有输入功能，如触发或输入捕获。

22. 4. 11 定时器和外部触发的同步

GP32C4Tn 定时器可在多种模式下与外部触发同步：复位模式、门控模式及触发模式。

22. 4. 11. 1 复位模式

计数器及其预分频器可以在响应触发输入事件时重新初始化。此外，若 GP32C4Tn_CON1 寄存器的 UERSEL 位为低时会产生一次更新事件 UEV。所有预载寄存器（GP32C4Tn_AR，GP32C4Tn_CCVALn）都会因更新事件 UEV 而被更新。

在下面例子中，I1 输入端的上升沿让递增计数被清空：

◇ 配置通道 1 上检测 I1 上的上升沿。配置输入滤波周期（本例无需滤波器，故 I1FLT = "0000"）。触发捕获分频器没有使用，无需配置。CC1SSEL 位只选择输入捕获源，GP32C4Tn_CHMR1 寄存器中 CC1SSEL = "01"。GP32C4Tn_CCEP 寄存器中 CC1POL = 0 以确定极性（只检测上升沿）。

◇ 定时器配置位复位模式：GP32C4Tn_SMCON 寄存器中 SMODS = "100"。选择 I1 作为输入源：GP32C4Tn_SMCON 寄存器中 TSSEL = "101"。

◇ 启动计数器：GP32C4Tn_CON1 寄存器中 CNTEN = '1'。

计数器依据内部时钟开始计数，正常计数直到 I1 上出现上升沿。当 I1 上出现上升沿时，计数器会被清零且从 0 重新开始计数。同时，标志位置位（GP32C4Tn_RIF 寄存器中 TRGIF 位），如果中断及 DMA 使能（取决于 GP32C4Tn_IER 寄存器中的 TRGIT 和 GP32C4Tn_DMAEN 的 TRGDMA 位），会发送中断及 DMA 请求。

下图给出了当自动重载寄存器 GP32C4Tn_AR = 0x36 时的信号变化。由于 I1 输入的再同步电路，I1 上的上升沿和计数器实际复位之间会存在延时（包含 2~3 个模块时钟周期的同步延时）。

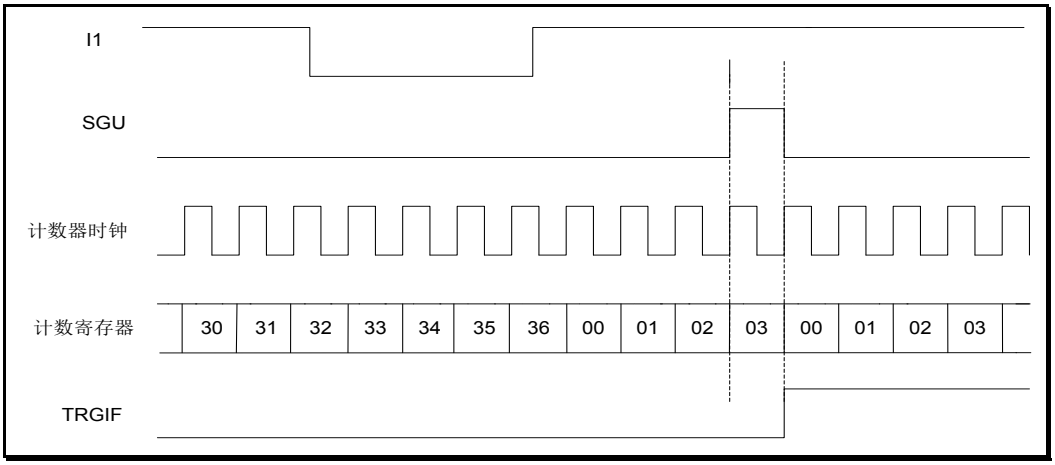


图 22-21 复位模式控制电路

22. 4. 11. 2 门控模式

计数器根据选中的输入电平被使能。

下面的例子中，计数器只在 I1 输入为低电平时才递增计数：

◇ 配置通道 1 在 I1 上检测低电平。配置输入滤波周期（本例不需要滤波器，I1FLT = "0000"）。触发捕获分频器没有使用，无需配置。GP32C4Tn_CHMR1 寄存器中的 CC1SSEL = "01"，选择输入捕获源。GP32C4Tn_CCEP 寄存器中 CC1POL = '1'，确认极性（只检测低电平）。

◇ 配置定时器为门控模式：GP32C4Tn_SMCON 寄存器中 SMODS = "101"。选择 I1 作为输入源：GP32C4Tn_SMCON 寄存器中 TSSEL = "101"。

◇ 使能计数器：GP32C4Tn_CON1 寄存器中 CNTEN = '1'（门控模式中，如果 CNTEN = '0'，无论触发输入为何电平，计数器都不会启动）。

只要 I1 为低电平，计数器依据内部时钟开始计数，一旦 I1 为高则停止计数。由于 I1 输入端再同步电路的原因，I1 上出现上升沿和计数器实际停止之间会有一定的延时。

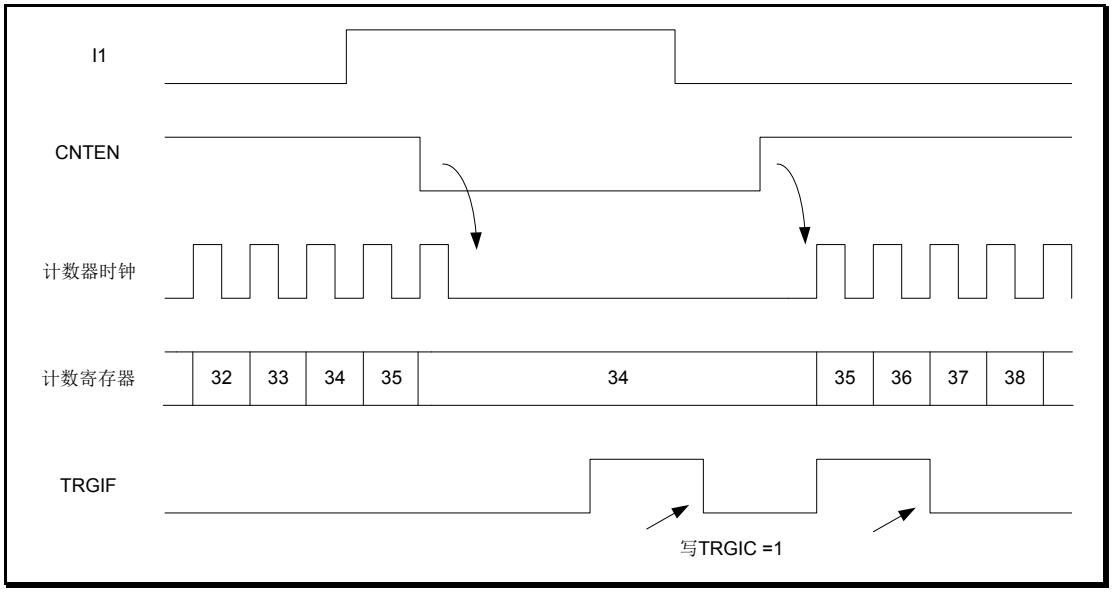


图 22-22 门控模式控制电路

22. 4. 11. 3 触发模式

输入端选中的事件可以使能计数器。

下面的例子中，I2 输入端上的上升沿可以启动递增计数：

◇ 配置通道 2 可以检测 I2 上的上升沿。配置滤波时间（本例不需要滤波，I2FLT = "0000"）。触发捕获分频器没有使用，无需配置。GP32C4Tn_CHMR1 寄存器中 CC2SSEL = "01",用于选择捕获源。GP32C4Tn_CCEP 寄存器中 CC2POL = '1'，确认极性（只检测低电平）。

◇ 配置定时器为触发模式：GP32C4Tn_SMCON 寄存器中 SMODS = "110"。GP32C4Tn_SMCON 寄存器中 TSSEL = "110"，用于选择输入源。

I2 上出现上升沿时，计数器开始依据内部时钟计数并置位 TRGIF 标志位。

由于 I2 输入的再同步原因，I2 上出现上升沿和计数器实际停止之间会有一定的延时。

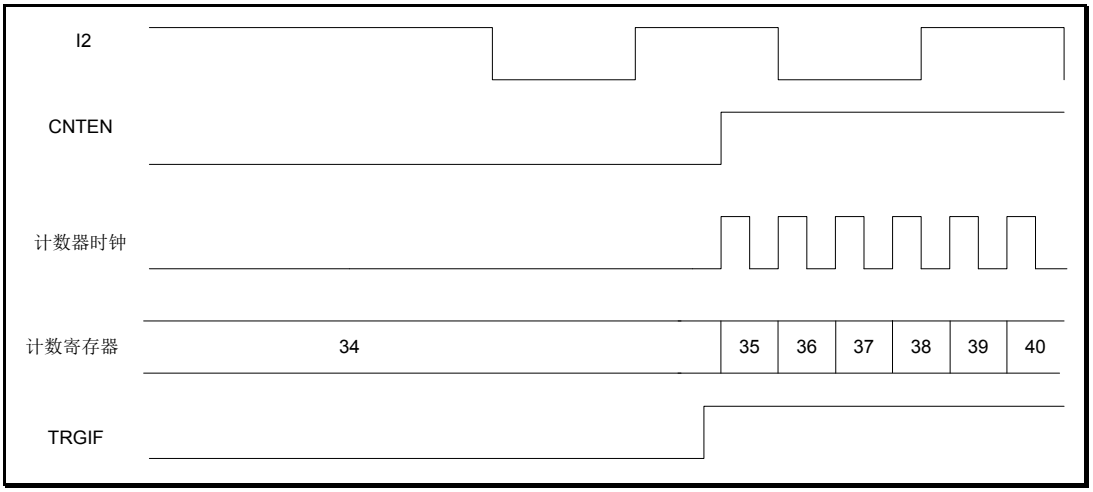


图 22-23　触发模式控制电路

22. 4. 11. 4　选择外部时钟源 2 的触发模式

外部时钟源 2 可和其他模式一起使用（除编码模式）。ET 信号可作为外部时钟输入，另一个输入可选择为触发输入（复位模式、门控模式或触发模式）。不推荐用 GP32C4Tn_SMCON 寄存器的 TSSEL 位选择 ET 作为 TI。

下面的例子中，一旦 I1 上出现上升沿时，计数器会依据 ET 信号的每个上升沿递增计数。

◇　通过 GP32C4Tn_SMCON 寄存器，配置外部触发输入电路，过程如下：

ETFLT = "000"：无滤波

ETPSEL = "00"：禁止分频

ETPOL = '0'：检测 ET 的上升沿，ECM2EN = '1'使能外部时钟模式 2

◇　配置通道 1 检测 I 的上升沿，过程如下：

I1FLT = "0000"：无滤波。

触发捕获分频器没有使用，无需配置。

GP32C4Tn_CHMR1 寄存器中 CC1SSEL = "01"选择输入捕获源，
GP32C4Tn_CCEP 寄存器的 CC1POL = '0'确认极性（只检测上升沿）。

◇　配置定时器为触发模式：GP32C4Tn_SMCON 寄存器中 SMODS = "110"。
GP32C4Tn_SMCON 寄存器中 TSSEL = "101"选择 I1 作为输入源。

I1 上出现上升沿时，计数器使能且 TRGIF 标志位置位，然后计数器根据 ET 上的上升沿开始计数。

由于 ETFP 输入再同步电路的原因，ET 信号的上升沿和实际计数器的复位会有延时。

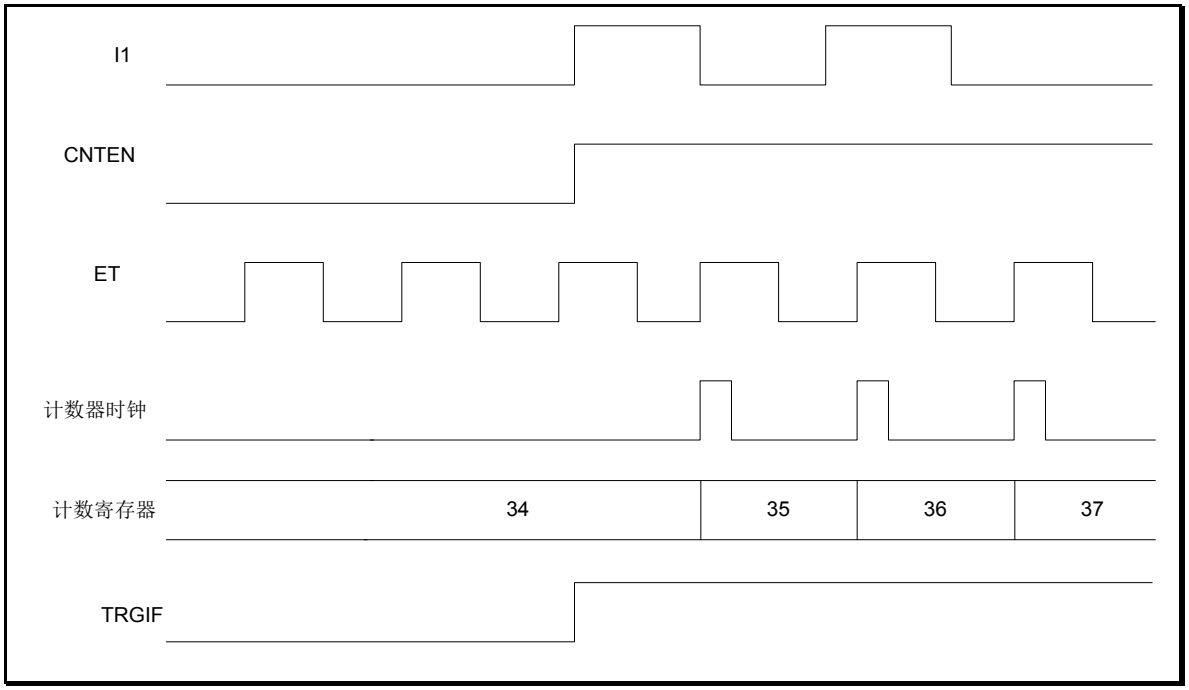


图 22-24 外部时钟源 2＋触发模式下的控制电路

22. 4. 12 调试模式

当微控制器进入调试模式（Cortex™-M3 内核停止），GP32C4Tn 计数器停止计数。

22. 5 特殊功能寄存器

22. 5. 1 寄存器列表

名称	偏移地址	描述
GP32C4Tn_CON1	000_H	控制寄存器 1
GP32C4Tn_CON2	004_H	控制寄存器 2
GP32C4Tn_SMCON	008_H	从模式控制寄存器
GP32C4Tn_IER	$00C_H$	中断使能寄存器
GP32C4Tn_IDR	010_H	中断禁止寄存器
GP32C4Tn_IVS	014_H	中断有效状态寄存器
GP32C4Tn_RIF	018_H	原始中断标志寄存器
GP32C4Tn_IFM	$01C_H$	中断标志屏蔽寄存器
GP32C4Tn_ICR	020_H	中断清零寄存器
GP32C4Tn_SGE	024_H	软件生成事件寄存器
GP32C4Tn_CHMR1	028_H	捕获/比较模式寄存器 1
GP32C4Tn_CHMR2	$02C_H$	捕获/比较模式寄存器 2
GP32C4Tn_CCEP	030_H	捕获/比较使能寄存器
GP32C4Tn_COUNT	034_H	计数器寄存器
GP32C4Tn_PRES	038_H	预分频寄存器
GP32C4Tn_AR	$03C_H$	自动重载寄存器
GP32C4Tn_CCVAL1	044_H	捕获/比较寄存器 1
GP32C4Tn_CCVAL2	048_H	捕获/比较寄存器 2
GP32C4Tn_CCVAL3	$04C_H$	捕获/比较寄存器 3
GP32C4Tn_CCVAL4	050_H	捕获/比较寄存器 4
GP32C4Tn_DMAEN	058_H	DMA 使能寄存器

22. 5. 2 寄存器描述

22. 5. 2. 1 控制寄存器 1（GP32C4Tn_CON1）

控制寄存器 1（GP32C4Tn_CON1）

偏移地址：000_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DBGSEL	Reserved	OCCISS			OCCISP	DFCKSEL		ARPEN	CMSEL		DIRSEL	SPMEN	UERSEL	DISUE	CNTEN

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
DBGSEL	Bit 15	R/W	**调试时通道输出状态选择：** 0：通道输出为高阻态 1：通道输出保持
Reserved	Bit 14	-	**保留，必须保持为复位值**
OCCISS	Bit 13-11	R/W	**通道输出清除内部触发源选择：** 000：无效 001：清除触发源通道 0 010：清除触发源通道 1 011：清除触发源通道 2 100：清除触发源通道 3
OCCISP	Bit 10	R/W	**通道输出清除内部触发源极性：** 0：低电平有效 1：高电平有效
DFCKSEL	Bit 9-8	R/W	**时钟分频** 该时钟分频为定时器（INT_CLK）频率与死区时间生成器和数字滤波器（ET，In）采用的死区时间和采样时钟（tDTS）之间的分频比。 00：$t_{DTS}=t_{INT\_CLK}$ 01：$t_{DTS}=2*t_{INT\_CLK}$ 10：$t_{DTS}=4*t_{INT\_CLK}$ 11：保留
ARPEN	Bit 7	R/W	**自动重载预载使能** 0：GP32C4Tn_AR 寄存器未缓冲 1：GP32C4Tn_AR 寄存器被装入缓冲器
CMSEL	Bit 6-5	R/W	**中央对齐模式选择** 00：**边沿对齐模式。**计数器根据方向为（DIRSEL）来向上或向下计数。 01：**中央对齐模式 1。**计数器以交替方式向上或向下计数。仅当计数器向下计数时，配置为输出的通道（GP32C4T_CHMRx 寄存器中 CCnSSEL =00）的输出比较中断标志位才会被设置。

			10：**中央对齐模式 2。**计数器以交替方式向上或向下计数。仅当计数器向上计数时，配置为输出的通道（GP32C4T_CHMRx 寄存器中 CCnSSEL =00）的输出比较中断标志位才会被设置。 11：**中央对齐模式 3。**计数器以交替方式向上或向下计数。当计数器向上或向下计数时，配置为输出的通道（GP32C4T_CHMRx 寄存器中 CCnSSEL =00）的输出比较中断标志位均会被设置。 注意：当计数器使能时（CNTEN=1）， 不允许从边沿对齐模式转换到中央对齐模式
DIRSEL	Bit 4	R/W	**计数器方向选择** 0：计数器向上计数 1：计数器向下计数 注意：当计数器配置为中央对齐模式或者编码器模式时，该位只读。
SPMEN	Bit 3	R/W	**单脉冲模式** 0：当发生更新事件时，计数器不停止。 1：当发生下一次更新事件（CNTEN 位清零）时，计数器停止。
UERSEL	Bit 2	R/W	**更新请求源** 该位由软件置 1 或清零，来选择 UEV 事件源。 0：如果更新中断或 DMA 请求使能，则下述任一事件都可产生更新中断或 DMA 请求： – 计数器上溢/下溢 – 设置 SGU 位 – 从模式控制器产生的更新 1：如果更新中断或 DMA 请求使能，仅计数器上溢/下溢才能产生更新中断或 DMA 请求中断
DISUE	Bit 1	R/W	**更新禁止** 该位由软件置 1 或清零来使能/禁止 UEV 事件的产生。 0：UEV 使能。更新事件（UEV）由下列任一事件产生： – 计数器上溢/下溢 – 设置 SGU 位 – 从模式控制器产生的更新 缓冲寄存器载入他们的预载值。 1：UEV 禁止。不产生更新事件，影子寄存器保持他们的值（ARRV， PSCV， CCRVx）。如果从模式控制器接收到硬件复位，计数器和预分频器将被重新初始化。
CNTEN	Bit 0	R/W	**计数器使能**

			0：计数器禁止 1：计数器使能 注意：如果软件设置了 CNTEN 位，外部时钟，门控模式和编码器模式才能工作。触发模式可由硬件自动设置 CNTEN 位。

22. 5. 2. 2 控制寄存器 2（GP32C4Tn_CON2）

控制寄存器 2（GP32C4Tn_CON2）																															
偏移地址：004_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								I1FSEL	TRGOSEL			CCDMASEL	Reserved		

Reserved	Bit 31-8	-	**保留，必须保持为复位值**
I1FSEL	Bit 7	R/W	**I1 选择** 0：GP32C4Tn_CH1 引脚与 I1 输入连接 1：GP32C4Tn_CH1， CH2 和 CH3 引脚与 I1 输入（XOR）连接
TRGOSEL	Bit 6-4	R/W	**选择主模式 TRGOUT 输出** 为同步（TRGOUT），该位可选择在主模式下发送至从计数器的信息。 000：**复位**–GP32C4Tn_SGE 寄存器中的 SGU 位被采用为触发输出（TRGOUT）。如果复位由触发输入生成（从模式控制器配置复位模式），则相较与实际复位，TRGOUT 上的信号将会延迟。 001：**使能**–计数器使能信号被用作触发输出（TRGOUT）。在从计数器使能的情况下，该设置用于在同一时间启动数次或者用来控制窗口。计数器使能信号是由 CNTEN 控制位与配置为门控模式的触发输入进行 OR 操作产生的。当计数器使能信号由触发输入控制，TRGOUT 上会产生延迟，除非被选为主/从模式（参考 GP32C4Tn_SMCON 寄存器中的 MSCFG 位的描述）。 010：**更新**–更新事件被选为触发输出（TRGOUT）。举例，主计数器可被用作从计数器的预分频器。 011：**比较脉冲**–一旦捕获或者比较匹配发生，当 CH1CCIF 标志位被置起（即便已为高电平），触发输出会发送一个正脉冲。 100：**比较**- 通道 1 比较输出信号用作触发输出 TRGOUT 101：**比较**- 通道 2 比较输出信号用作触发输出 TRGOUT 110：**比较**- 通道 3 比较输出信号用作触发输出 TRGOUT

			111：**比较**- 通道 4 比较输出信号用作触发输出 TRGOUT
CCDMASEL	Bit 3	R/W	**捕获/比较 DMA 选择** 0：当 CCn 事件发生，会发出 CCn DMA 请求。 1：当发生更新时间，会发出 CCn DMA 请求。
Reserved	Bit 2-0	-	**保留，必须保持为复位值**

22. 5. 2. 3 从模式控制寄存器（GP32C4Tn_SMCON）

从模式控制寄存器（GP32C4Tn_SMCON）																															
偏移地址：008_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ETPOL	ECM2EN	ETPSEL		ETFLT				MSCFG	TSSEL			OCCS	SMODS		

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
ETPOL	Bit 15	R/W	**外部触发极性** 0：正向 ET，高电平有效或上升沿有效 1：反正 ET，低电平有效或下降沿有效
ECM2EN	Bit 14	R/W	**使能外部时钟模式 2** 该位使能外部时钟模式 2 0：禁止外部时钟模式 2 1：使能外部时钟模式 2。计数器由 ETFP 信号上的有效边沿计数。 注意： 1. 设置 ECM2EN 位与选择外部时钟模式 1 且 TI 与 ETFP 相连接（SMODS=111 和 TSSEL=111）具有相同的效果。 2. 可同时使用外部时钟模式 2 与下列从模式：复位模式，门控模式和除法模式。在这种情况下，TI 不能与 ETFP 相连接（TSSEL 不能设置为 111。 3. 如果外部时钟模式 1 和外部时钟模式 2 同时使能，外部时钟输入为 ETFP。
ETPSEL	Bit 13-12	RW	**外部触发预分频器** 外部触发信号频率最大为 GP32C4TnCLK 频率的 1/4。可使能预分频器来减小 ETFP 频率。该位有效用于输入高速外部时钟的情况。 00：预分频器关闭 01：ETFP 频率 2 分频 10：ETFP 频率 4 分频 11：ETFP 频率 8 分频
ETFLT	Bit 11-8	R/W	**外部触发滤波器** 该位定义了对 ET 信号的采样频率和数字滤波器的滤波长度。 数字滤波器由一个事件计数器组成，每 N 个连续事件才视为一个有效边沿。 0000：无滤波器，采样频率为 f_{DTS} 0001：$f_{SAMPLING} = f_{INT\_CLK}$, N = 2

			0010：$f_{SAMPLING} = f_{INT\_CLK}$, N = 4 0011：$f_{SAMPLING} = f_{INT\_CLK}$, N = 8 0100：$f_{SAMPLING} = f_{DTS}$ / 2, N = 6 0101：$f_{SAMPLING} = f_{DTS}$ / 2, N = 8 0110：$f_{SAMPLING} = f_{DTS}$ / 4, N = 6 0111：$f_{SAMPLING} = f_{DTS}$ / 4, N = 8 1000：$f_{SAMPLING} = f_{DTS}$ / 8, N = 6 1001：$f_{SAMPLING} = f_{DTS}$ / 8, N = 8 1010：$f_{SAMPLING} = f_{DTS}$ / 16, N = 5 1011：$f_{SAMPLING} = f_{DTS}$ / 16, N = 6 1100：$f_{SAMPLING} = f_{DTS}$ / 16, N = 8 1101：$f_{SAMPLING} = f_{DTS}$ / 32, N = 5 1110：$f_{SAMPLING} = f_{DTS}$ / 32, N = 6 1111：$f_{SAMPLING} = f_{DTS}$ / 32, N = 8 注意： 当ETFLT[3:0] = 1， 2或3时，公式中的f_{DTS}由INT_CLK 取代。
MSCFG	Bit 7	R/W	**主/从模式** 0：无动作 1：延迟触发输入（In）上的事件来允许当前计时器和其从器件之间的同步。该设置有效用于使用单个外部事件来同步多个计时器。
TSSEL	Bit 6-4	R/W	**触发选择** 该位用来选择不同的触发输入来同步计数器。 000：内部触发 0（IT0） 001：内部触发 1（IT1） 010：内部触发 2（IT2） 011：内部触发 3（IT3） 100：I1 边沿检测器（I1F_ED） 101：滤波计时器输入 1 110：滤波计时器输入 2 111：外部触发输入 注意：为了避免错误边沿检测，该位在不使用时（SMODS=000）才能改变。
OCCS	Bit 3	R/W	**输出通道清除源选择** 0：OCCISS 选择的内部通道 1：ETFP
SMODS	Bit 2-0	R/W	**选择从模式功能** 当选择外部信号，触发信号TI的有效边沿与外部输入的极性有关系（详见输入控制寄存器和控制寄存器描述） 000：**禁止从模式**–如果CNTEN = ‘1’，则预分频器直接由内部时钟计数。 001：**编码器模式1**–计数器向上/向下计数I2边沿，

			取决于I1电平。 010：**编码器模式2** -计数器向上/向下计数I1边沿，取决于I2电平 011：**编码器模式3** -计数器向上/向下计数I1边沿检出和I2边沿检出边沿，取决于另一个输入的电平。 100：**复位模式**–选中的触发输入的上升沿重新初始化计数器，生成寄存器的更新 101：**门控模式**–当触发输入TI为高电平，计数器时钟使能。一旦触发变为低电平，计数器停止计数（并非复位）。计数器的启动和停止均受控制。 110：**触发模式**–计数器在触发信号TI的上升沿处启动（不复位）。仅寄存器的启动受控制。 111：**外部时钟模式1**–计数器在TI的上升沿计数 注意：如果I1双边沿检出被选为触发输入（TSSEL='100'），不能使用门控模式。I1每一次转换，I1双边沿检出就会输出1个脉冲，而门控模式则是检查触发信号的电平。 注意：在发生来自主计时器的接收事件之前，从计时器的时钟必须先使能，且在接收来自主计时器的触发过程中，从计数器时钟不能即时更改。

22. 5. 2. 4 中断使能寄存器（GP32C4Tn_IER）

中断使能寄存器（GP32C4Tn_IER）																															
偏移地址：$00C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CC4OIT	CC3OIT	CC2OIT	CC1OIT	Reserved		TRGIT	Reserved	CC4IT	CC3IT	CC2IT	CC1IT	UIT

Reserved	Bit31-13	-	保留，必须保持复位值。
CC4OIT	Bit12	W	**使能捕获/比较 4 捕获溢出中断** 0：无效 1：使能
CC3OIT	Bit11	W	**使能捕获/比较 3 捕获溢出中断** 0：无效 1：使能
CC2OIT	Bit10	W	**使能捕获/比较 2 捕获溢出中断** 0：无效 1：使能
CC1OIT	Bit9	W	**使能捕获/比较 1 捕获溢出中断** 0：无效 1：使能
Reserved	Bit 8-7	-	保留，必须保持为复位值
TRGIT	Bit6	W	**使能触发中断** 0：无效 1：使能
Reserved	Bit 5	-	保留，必须保持为复位值
CC4IT	Bit4	W	**使能捕获/比较 4 中断** 0：无效 1：使能
CC3IT	Bit3	W	**使能捕获/比较 3 中断** 0：无效 1：使能
CC2IT	Bit2	W	**使能捕获/比较 2 中断** 0：无效 1：使能
CC1IT	Bit1	W	**使能捕获/比较 1 中断** 0：无效 1：使能
UIT	Bit0	W	**使能更新事件中断** 0：无效 1：使能

22. 5. 2. 5 中断禁止寄存器（GP32C4Tn_IDR）

中断禁止寄存器（GP32C4Tn_IDR）																														
偏移地址：0010_H																														
复位值：$00000000\_00000000\_00000000\_00000000_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CC4OIT	CC3OIT	CC2OIT	CC1OIT	Reserved		TRGIT	Reserved	CC4IT	CC3IT	CC2IT	CC1IT	UIT

Reserved	Bit 31-13	-	**保留，必须保持复位值。**
CC4OIT	Bit12	W	**使能捕获/比较 4 捕获溢出中断** 0：无效 1：使能
CC3OIT	Bit11	W	**使能捕获/比较 3 捕获溢出中断** 0：无效 1：使能
CC2OIT	Bit10	W	**使能捕获/比较 2 捕获溢出中断** 0：无效 1：使能
CC1OIT	Bit9	W	**使能捕获/比较 1 捕获溢出中断** 0：无效 1：使能
Reserved	Bit 8-7	-	**保留，必须保持为复位值**
TRGIT	Bit 6	W	**禁止触发中断** 0：无效 1：禁止
Reserved	Bit 5	-	**保留，必须保持为复位值**
CC4IT	Bit 4	W	**禁止捕获/比较 4 中断** 0：无效 1：禁止
CC3IT	Bit 3	W	**禁止捕获/比较 3 中断** 0：无效 1：禁止
CC2IT	Bit 2	W	**禁止捕获/比较 2 中断** 0：无效 1：禁止
CC1IT	Bit 1	W	**禁止捕获/比较 1 中断** 0：无效 1：禁止
UIT	Bit 0	W	**禁止更新中断** 0：无效 1：禁止

22. 5. 2. 6 中断有效状态寄存器（GP32C4Tn_IVS）

中断有效状态寄存器（GP32C4Tn_IVS）																															
偏移地址：0014_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CC4OIT	CC3OIT	CC2OIT	CC1OIT	Reserved		TRGIT	Reserved	CC4IT	CC3IT	CC2IT	CC1IT	UIT

Reserved	Bit 31-13	-	**保留**
CC4OIT	Bit12	W	**使能捕获/比较 4 捕获溢出中断** 0：无效 1：使能
CC3OIT	Bit11	W	**使能捕获/比较 3 捕获溢出中断** 0：无效 1：使能
CC2OIT	Bit10	W	**使能捕获/比较 2 捕获溢出中断** 0：无效 1：使能
CC1OIT	Bit9	W	**使能捕获/比较 1 捕获溢出中断** 0：无效 1：使能
Reserved	Bit 8-7	-	**保留，必须保持为复位值**
TRGIT	Bit 6	R	**触发中断状态** 0：禁止 1：使能
Reserved	Bit 5	-	**保留，必须保持为复位值**
CC4IT	Bit 4	R	**通道 4 捕获/比较中断状态** 0：禁止 1：使能
CC3IT	Bit 3	R	**通道3捕获/比较中断状态** 0：禁止 1：使能
CC2IT	Bit 2	R	**通道 2 捕获/比较中断状态** 0：禁止 1：使能
CC1IT	Bit 1	R	**通道2捕获/比较中断状态** 0：禁止 1：使能
UIT	Bit 0	R	**更新事件中断状态** 0：禁止 1：使能

22. 5. 2. 7 原始中断标志寄存器（GP32C4Tn_RIF）

原始中断标志（GP32C4Tn_RIF）

偏移地址：018_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CH4OVIF	CH3OVIF	CH2OVIF	CH1OVIF	Reserved		TRGIF	Reserved	CH4CCIF	CH3CCIF	CH2CCIF	CH1CCIF	UEVTIF

Reserved	Bit 31-13	-	**保留，必须保持为复位值**
CH4OVIF	Bit 12	R	**捕获/比较 4 捕获溢出中断标志** 仅当相应的通道配置为捕获输入状态时，该标志位才由硬件设置。对 GP32C4Tn_ICR 写 1 来清除原始中断。 0：未检测到捕获溢出 1：当 CH4CCIF 标志位置起时，捕获计数器值至 GP32C4Tn_CCVAL1 寄存器
CH3OVIF	Bit 11	R	**捕获/比较 3 捕获溢出中断标志** 仅当相应的通道配置为捕获输入状态时，该标志位才由硬件设置。对 GP32C4Tn_ICR 写 1 来清除原始中断。 0：未检测到捕获溢出 1：当 CH3CCIF 标志位置起时，捕获计数器值至 GP32C4Tn_CCVAL1 寄存器
CH2OVIF	Bit 10	R	**捕获/比较 2 捕获溢出中断标志** 仅当相应的通道配置为捕获输入状态时，该标志位才由硬件设置。对 GP32C4Tn_ICR 写 1 来清除原始中断。 0：未检测到捕获溢出 1：当 CH2CCIF 标志位置起时，捕获计数器值至 GP32C4Tn_CCVAL1 寄存器
CH1OVIF	Bit 9	R	**捕获/比较 1 捕获溢出中断标志** 仅当相应的通道配置为捕获输入状态时，该标志位才由硬件设置。对 GP32C4Tn_ICR 写 1 来清除原始中断。 0：未检测到捕获溢出 1：当 CH1CCIF 标志位置起时，捕获计数器值至 GP32C4Tn_CCVAL1 寄存器
Reserved	Bit 8-7	-	保留，必须保持为复位值
TRGIF	Bit 6	R	**触发中断标志** 如果触发中断使能，当从模式控制器在门控模式

			以外的所有模式下使能，发生触发事件时（In 上检测到有效边沿），该标志位被硬件置起。对 GP32C4Tn_ICR 写 1 来清除原始中断. 0：未发生触发事件 1：触发中断被挂起
Reserved	Bit 5	-	**保留，必须保持为复位值**
CH4CCIF	Bit 4	R	**捕获/比较 4 中断标志** 参考 CH1CCIF 描述
CH3CCIF	Bit 3	R	**捕获/比较 3 中断标志** 参考 CH1CCIF 描述
CH2CCIF	Bit 2	R	**捕获/比较 2 中断标志** 参考 CH1CCIF 描述
CH1CCIF	Bit 1	R	**捕获/比较 1 中断标志** **如果 CC1 通道配置为输出:** 如果中断使能，除去中央对齐模式的情况（参考 GP32C4Tn_CON1 寄存器中 CMSEL 的描述），当计数值与比较值匹配，该标志位由硬件置起。对 GP32C4Tn_ICR 写 1 来清除原始中断。 0：不匹配 1：GP32C4Tn_COUNT 计数值与 GP32C4Tn_CCVAL1 值匹配。当 GP32C4Tn_CCVAL1 寄存器值大于 GP32C4Tn_AR 值，发生计数器上溢时（递增模式和递增/递减模式）或下溢时（递减模式），CH1CCIF 为被置起。 **如果 CC1 通道配置为输入:** 发生捕获时，该位由硬件置起。该位可通过软件或者读取 GP32C4Tn_CCVAL1 寄存器来清零。 0：未发生输入捕获 1：计数值捕获至 GP32C4Tn_CCVAL1 寄存器（I1 上检测到与选中极性匹配的边沿）
UEVTIF	Bit 0	R	**更新中断标志** 如果更新中断使能，当发生更新事件，该标志位由硬件置起。对 GP32C4Tn_ICR 写 1 来清除原始中断。 0：未发生更新。 1：更新中断被挂起。当寄存器更新时，该位被硬件置起： –当重复计数器值发生上溢或者下溢（若重复计数器=0，则更新）和当 GP32C4Tn_CON1 寄存器中 DISUE=0 –当使用 GP32C4Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时，如果 GP32C4Tn_CON1 寄存中的 UERSEL=0 和

			DISUE=0 –当 CNT 由触发事件来重新初始化，如果 GP32C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0

22. 5. 2. 8 中断标志屏蔽寄存器（GP32C4Tn_IFM）

中断标志屏蔽寄存器（GP32C4Tn_IFM）																															
偏移地址：$001C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CH4OVIM	CH3OVIM	CH2OVIM	CH1OVIM	Reserved		TRGIM	Reserved	CH4CCIM	CH3CCIM	CH2CCIM	CH1CCIM	UEVTIM

Reserved	Bit 31-13	-	**保留**
CH4OVIM	Bit 12	R	**屏蔽通道 4 捕获/比较捕获溢出中断标志** 0：未检测到捕获溢出 1：当 CH4CCIF 标志为置起时，捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
CH3OVIM	Bit 11	R	**屏蔽通道 3 捕获/比较捕获溢出中断标志** 0：未检测到捕获溢出 1：当 CH3CCIF 标志为置起时，捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
CH2OVIM	Bit 10	R	**屏蔽通道 2 捕获/比较捕获溢出中断标志** 0：未检测到捕获溢出 1：当 CH2CCIF 标志为置起时，捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
CH1OVIM	Bit 9	R	**屏蔽通道 1 捕获/比较捕获溢出中断标志** 0：未检测到捕获溢出 1：当 CH1CCIF 标志为置起时，捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
Reserved	Bit 8-7	-	**保留**
TRGIM	Bit 6	R	**屏蔽触发中断标志** 如果触发中断使能，当从模式控制器在门控模式以外的所有模式下使能，发生触发事件时（TI 上检测到有效边沿），该标志位被硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生触发事件 1：触发中断被挂起
Reserved	Bit 5	-	**保留**
CH4CCIM	Bit 4	R	**屏蔽通道 4 捕获/比较中断标志** 参考 CH1CCIM 描述
CH3CCIM	Bit 3	R	**屏蔽通道 3 捕获/比较中断标志** 参考 CH1CCIM 描述
CH2CCIM	Bit 2	R	**屏蔽通道 2 捕获/比较中断标志** 参考 CH1CCIM 描述
CH1CCIM	Bit 1	R	**屏蔽通道 1 捕获/比较中断标志如果通道 1 配置为**

			输出： 如果中断使能，除去中央对齐模式的情况（参考 AD16C4Tn_CON1 寄存器中 CMSEL 的描述），当计数值与比较值匹配，该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：不匹配。 1：AD16C4Tn_COUNT 计数值与 AD16C4Tn_CCVAL1 值匹配。当 AD16C4Tn_CCVAL1 寄存器值大于 AD16C4Tn_AR 值，发生计数器上溢时（递增模式和递增/递减模式）或下溢时（递减模式），CH1CCIF 为被置起。 **如果通道配置为输入：** 发生捕获时，该位由硬件置起。该位可通过软件或者读取 AD16C4Tn_CCVAL1 寄存器来清零。 0：未发生输入捕获 1：计数值捕获至 AD16C4Tn_CCVAL1 寄存器（I1 上检测到与选中极性匹配的边沿）
UEVTIM	Bit 0	R	**屏蔽更新事件中断标志** 如果更新中断使能，当发生更新事件，该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生更新。 1：更新中断被挂起。当寄存器更新时，该位被硬件置起： –当重复计数器值发生上溢或者下溢（若重复计数器=0，则更新）和当 AD16C4Tn_CON1 寄存器中 DISUE=0 –当使用 AD16C4Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时，如果 AD16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0 –当 CNT 由触发事件来重新初始化，如果 AD16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0

22. 5. 2. 9 中断清零寄存器（GP32C4Tn_ICR）

中断清零寄存器（GP32C4Tn_ICR）																															
偏移地址：020_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CH4OVIC	CH3OVIC	CH2OVIC	CH1OVIC	Reserved		TRGIC	Reserved	CH4CCIC	CH3CCIC	CH2CCIC	CH1CCIC	UEVTIC

Reserved	Bit 31-13	-	**保留，必须保持为复位值**
CH4OVIC	Bit 12	R	**通道 4 捕获/比较捕获溢出中断清零** 0：无效 1：CH4OVIF 清除
CH3OVIC	Bit 11	R	**通道 3 捕获/比较捕获溢出中断清零** 0：无效 1：CH3OVIF 清除
CH2OVIC	Bit 10	R	**通道 2 捕获/比较捕获溢出中断清零** 0：无效 1：CH2OVIF 清除
CH1OVIC	Bit 9	R	**通道 1 捕获/比较捕获溢出中断清零** 0：无效 1：CH1OVIF 清除
Reserved	Bit 8-7	-	**保留，必须保持为复位值**
TRGIC	Bit 6	C_W1	**触发中断清零** 0：无效 1：触发中断清零（GP32C4Tn_RIF）
Reserved	Bit 5	-	**保留，必须保持为复位值**
CH4CCIC	Bit 4	C_W1	**捕获/比较 4 中断清零** 参考 CH1CCIC 描述
CH3CCIC	Bit 3	C_W1	**捕获/比较 3 中断清零** 参考 CH1CCIC 描述
CH2CCIC	Bit 2	C_W1	**捕获/比较 2 中断清零** 参考 CH1CCIC 描述
CH1CCIC	Bit 1	C_W1	**捕获/比较 1 中断清零** 0：无效 1：捕获/比较中断清零（GP32C4Tn_RIF）
UEVTIC	Bit 0	C_W1	**更新中断清零** 0：无效 1：更新中断清零（GP32C4Tn_RIF）

22. 5. 2. 10 软件生成事件寄存器（GP32C4Tn_SGE）

软件生成事件寄存器（GP32C4Tn_SGE）																												
偏移地址：024_H																												
复位值：$00000000\_00000000\_00000000\_00000000_B$																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									SGTRG	Reserved	SGCC4E	SGCC3E	SGCC2E	SGCC1E	SGU

Reserved	Bit 31-7	-	**保留，必须保持为复位值**
SGTRG	Bit 6	W	**触发生成** 该位由软件设置来生成触发事件，可由硬件自动清零。 0：无动作 1：GP32C4Tn_RIF 寄存器中的 TRGIF 被置起，产生相关中断或 DMA 传输
Reserved	Bit 5	-	**保留，必须保持为复位值**
SGCC4E	Bit 4	W	**捕获/比较 4 生成** 参考 SGCC1E 描述
SGCC3E	Bit 3	W	**捕获/比较 3 生成** 参考 SGCC1E 描述
SGCC2E	Bit 2	W	**捕获/比较 2 生成** 参考 SGCC1E 描述
SGCC1E	Bit 1	W	**捕获/比较 1 生成** 该位由软件设置来生成事件，可由硬件自动清零。 0：无动作 1：通道 1 上产生捕获/比较事件： **如果通道 1 配置为输出：** CH1CCIF 标志位被置起，产生相应中断或 DMA 请求发送 **如果通道 1 配置为输入：** 当前计数值捕获至 GP32C4Tn_CCVAL1 寄存器。CH1CCIF 标志位被置起，产生相应中断或 DMA 请求发送。CH1OVIF 标志位置起如果 CH1CCIF 标志位为高电平。
SGU	Bit 0	W	**更新生成** 该位由软件设置，可由硬件自动清零. 0：无动作 1：重新初始化计数器，更新寄存器。注意，预分频器也会被清零（但预分频比不会受到影响）。如果使用中央对齐模式或者 DIRSEL=0（递增），则计数器将清零；否则如果 DIRSEL=1（递减），则将使用自动重载入值。

22. 5. 2. 11 捕获/比较模式寄存器 1（GP32C4Tn_CHMR1）

◆ 输出比较模式

捕获/比较模式寄存奇 1（GP32C4Tn_CHMR1）																													
偏移地址：028_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CH2OCLREN	CH2OMOD			CH2OPREN	CH2OHSEN	CC2SSEL		CH1OCLREN	CH1OMOD			CH1OPREN	CH1OHSEN	CC1SSEL	

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
CH2OCLREN	Bit 15	R/W	**输出比较 2 清零使能** 参考 CH1OCLREN 描述
CH2OMOD	Bit 14-12	R/W	**输出比较 2 模式** 参考 CH1OMOD 描述
CH2OPREN	Bit 11	R/W	**输出比较 2 预载使能** 参考 CH1OPREN 描述
CH2OHSEN	Bit 10	R/W	**输出比较 2 高速使能** 参考 CH1OHSEN 描述
CC2SSEL	Bit 9-8	R/W	**输出比较 2 选择** 该位定义了通道以及使用的输入的方向（输入/输出） 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I2** 10：通道配置为输入，捕获源为 **I1** 11：通道配置为输入，捕获源为 **ITn 或 I1** 的双边沿检出。 仅当内部触发输入通过 TSSEL 位（GP32C4Tn_SMCON 寄存器）选择时，该模式才能工作。 注意：当通道为关闭状态时（GP32C4Tn_CCEP 中 CC2EN = '0'），CC2SSEL 为只写。
CH1OCLREN	Bit 7	R/W	**输出比较 1 清零使能** 0：通道 1 比较输出不会受到 ETFP 输入影响 1：当 ETFP 输入上检测到高电平时，通道 1 比较输出将被清零
CH1OMOD	Bit 6-4	R/W	**输出比较 1 模式** 该位定义了输出参考信号通道 1 比较输出的行为。通道 1 比较输出为高有效，CH1O 的有效电平由 CC1POL 位决定。 000 :**冻结**–输出比较寄存器 GP32C4Tn_CCVAL1

			寄存器和 GP32C4Tn_COUNT 计数器之间的比较对输出无效。 001：**发生匹配时设置通道 1 为有效电平**–当计数器 GP32C4Tn_COUNT 与捕获/比较寄存器 1GP32C4Tn_CCVAL1 发生匹配后，通道 1 比较输出信号强制为高电平。 010：**发生匹配时设置通道 1 为无效电平**-当计数器 GP32C4Tn_COUNT 与捕获/比较寄存器 1GP32C4Tn_CCVAL1 发生匹配后，通道 1 比较输出信号强制为低电平。 011：**翻转** -当 GP32C4Tn_COUNT=GP32C4Tn_CCVAL1，通道 1 比较输出发生翻转。 100：**强制为无效电平** - 通道 1 比较输出强制为低电平。 101：**强制为有效电平**- 通道 1 比较输出强制为高电平。 110：**PWM 模式 1** -在递增模式下，当 GP32C4Tn_COUNT<GP32C4Tn_CCVAL1，通道 1 为有效电平，否则，通道 1 为无效电平。在递减模式下，当 GP32C4Tn_COUNT>GP32C4Tn_CCVAL1，通道 1 为无效电平（通道 1 比较输出='0'），否则通道 1 为有效电平（通道 1 比较输出='1'）。 111：**PWM 模式 2** -在递增模式下，当 GP32C4Tn_COUNT<GP32C4Tn_CCVAL1，通道 1 为无效电平，否则，通道 1 为有效电平。在递减模式下，当 GP32C4Tn_COUNT>GP32C4Tn_CCVAL1，通道 1 为有效电平，否则通道 1 为无效电平。 注意： 在 PWM 模式 1 和 2 中，仅当比较结果更改或当输出比较模式从冻结模式转换成 PWM 模式，比较输出电平才会更改。
CH1OPREN	Bit 3	R/W	**输出比较 1 预载使能** 0：GP32C4Tn_CCVAL1 的预载寄存器禁止。GP32C4Tn_CCVAL1 在任何时候都可写，新写入的值将立刻生效。 1：GP32C4Tn_CCVAL1 的预载寄存器使能。读/写操作可访问预载寄存器。每当发生一次更新事件，GP32C4Tn_CCVAL1 预载入值将会被填入有效寄存器。 注意： 仅在单脉冲模式下（GP32C4Tn_CON1 寄存器中

			的 SPMEN 设置为 1），PWM 模式可在不经过验证预载寄存器的情况下使用。其他情况下的行为不做保证。
CH1OHSEN	Bit 2	R/W	**输出比较 1 高速使能** 该位用来加速在 CC 输出上的输入触发事件的效应。 0：当触发开启，通道 1 运作正常取决于计数器和 CCRV1 的值。当触发输入上发现边沿时，至少需要 5 个时钟周期来激活通道 1 输出。 1：触发输入上的有效沿类似于通道 1 输出上的比较匹配。设置 OC 为 1 用来比较电平，采样触发输入和激活通道 1 输出的延时将会减少至 3 个时钟周期。只有当通道配置为 PWM1 或 PWM2 模式，CH1OHSEN 才会起作用。
CC1SSEL	Bit 1-0	R/W	**捕获/比较 1 选择** 该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I1** 10：通道配置为输入，捕获源为 **I2** 11：通道配置为输入，捕获源为 **ITn 或 I1 的双边沿检出。** 只有当内部触发输入是通过 TSSEL 位（GP32C4Tn_SMCON 寄存器）选择时，该模式才运行。 注意：当通道关闭（GP32C4Tn_CCEP 寄存器中的 CC1EN = ‘0’），CC1SSEL 为只写。

◆ 输入捕获模式

捕获/比较模式寄存奇 1（GP32C4Tn_CHMR1）																													
偏移地址：028_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																I2FLT				IC2PRES		CC2SSEL		I1FLT				IC1PRES		CC1SSEL	

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
I2FLT	Bit 15-12	R/W	**输入捕获 2 滤波器** 参考 I1FLT 描述
IC2PRES	Bit 11-10	R/W	**输入捕获 2 预分频器** 参考 IC1PRES 描述
CC2SSEL	Bit 9-8	R/W	**输入捕获 2 选择** 该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I2** 10：通道配置为输入，捕获源为 **I1** 11：通道配置为输入，捕获源为 **ITn 或 I1 的双边沿检出** 只有当内部触发输入是通过 TSSEL 位（GP32C4Tn_SMCON 寄存器）选择时，该模式才运行 注意：当通道关闭（GP32C4Tn_CCEP 寄存器中的 CC2EN = '0'），CC2SSEL 为只写。
I1FLT	Bit 7-4	R/W	**输入捕获 1 滤波器** 该位定义了 I1 输入的采样频率和数字滤波器的长度。 数字滤波器由一个事件计数器组成，每 N 个连续事件才视为一个有效边沿： 0000：无滤波器，采样频率为 f_{DTS} 0001：$f_{SAMPLING} = f_{INT\_CLK}$, N = 2 0010：$f_{SAMPLING} = f_{INT\_CLK}$, N = 4 0011：$f_{SAMPLING} = f_{INT\_CLK}$, N = 8 0100：$f_{SAMPLING} = f_{DTS} / 2$, N = 6 0101：$f_{SAMPLING} = f_{DTS} / 2$, N = 8 0110：$f_{SAMPLING} = f_{DTS} / 4$, N = 6 0111：$f_{SAMPLING} = f_{DTS} / 4$, N = 8 1000：$f_{SAMPLING} = f_{DTS} / 8$, N = 6 1001：$f_{SAMPLING} = f_{DTS} / 8$, N = 8 1010：$f_{SAMPLING} = f_{DTS} / 16$, N = 5 1011：$f_{SAMPLING} = f_{DTS} / 16$, N = 6

			1100：$f_{SAMPLING} = f_{DTS} / 16$, N = 8 1101：$f_{SAMPLING} = f_{DTS} / 32$, N = 5 1110：$f_{SAMPLING} = f_{DTS} / 32$, N = 6 1111：$f_{SAMPLING} = f_{DTS} / 32$, N = 8 注意：当 ICxF [3:0] = 1， 2 或 3 时，公式中的 f_{DTS} 由 INT_CLK 取代
IC1PRES	Bit 3-2	R/W	**输入捕获 1 预分频器** 该位定义了作用在 CC1 输入（I1）上的预分频比。当 CC1EN='0' （GP32C4Tn_CCEP 寄存器），预分频器将复位。 00：无预分频器。每当捕获输入上检测到边沿时，发生捕获动作。 01：每发生 2 次事件，执行一次捕获 10：每发生 4 次事件，执行一次捕获 11：每发生 8 次事件，执行一次捕获
CC1SSEL	Bit 1-0	R/W	**输入捕获 1 选择** 该位定义了通道和使用的输入的方向 00：CC1 通道配置为**输出** 01：CC1 通道配置为输入，IC1 映射到 **I1** 10：CC1 通道配置为输入，IC1 映射到 **I2** 11：CC1 通道配置为输入，捕获源为 ITn 或 I1 的双边沿检出。只有当内部触发输入是通过 TSSEL 位（GP32C4Tn_SMCON 寄存器）选择时，该模式才运行 注意： 当通道关闭（GP32C4Tn_CCEP 寄存器中的 CC1EN = '0'），CC1SSEL 为只写。

22. 5. 2. 12 捕获/比较模式寄存器 2（GP32C4Tn_CHMR2）

◆ 输出比较模式

捕获/比较模式寄存器 2（GP32C4Tn_CHMR2）																															
偏移地址：$02C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CH4OCLREN	CH4OMOD			CH4OPREN	CH4OHSEN	CC4SSEL		CH3OCLREN	CH3OMOD			CH3OPREN	CH3OHSEN	CC3SSEL	

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
CH4OCLREN	Bit 15	R/W	**输出比较 4 清零使能** 参考 CH1OCLREN 描述
CH4OMOD	Bit 14-12	R/W	**输出比较 4 模式** 参考 CH1OMOD 描述
CH4OPREN	Bit 11	R/W	**输出比较 4 预载使能** 参考 CH1OPREN 描述
CH4OHSEN	Bit 10	R/W	**输出比较 4 高速使能** 参考 CH1OHSEN 描述
CC4SSEL	Bit 9-8	R/W	**输出比较 4 选择** 该位定义了通道以及使用的输入的方向（输入/输出）。 00：通道配置为输出 01：通道配置为输入，捕获源为 I4 10：通道配置为输入，捕获源为 I3 11：通道配置为输入，捕获源为 ITn 或 I1 的双边沿检出 仅当内部触发输入通过 TSSEL 位（GP32C4Tn_SMCON 寄存器）选择时，该模式才能工作。 注意：当通道为关闭状态时（GP32C4Tn_CCEP 中 CC4EN = '0'），CC4SSEL 为只写。
CH3OCLREN	Bit 7	R/W	**输出比较 3 清零使能** 参考 CH1OCLREN 描述
CH3OMOD	Bit 6-4	R/W	**输出比较 3 模式** 参考 CH1OMOD 描述
CH3OPREN	Bit 3	R/W	**输出比较 3 预载使能** 参考 CH1OPREN 描述
CH3OHSEN	Bit 2	R/W	**输出比较 3 高速使能** 参考 CH1OHSEN 描述
CC3SSEL	Bit 1-0	R/W	**捕获/比较 3 选择**

			该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I3** 10：通道配置为输入，捕获源为 **I4** 11：通道配置为输入，捕获源为 ITn 或 I1 的双边沿检出。 只有当内部触发输入是通过 TSSEL 位（GP32C4Tn_SMCON 寄存器）选择时，该模式才运行。 注意：当通道关闭（GP32C4Tn_CCEP 寄存器中的 CC3EN = '0'），CC3SSEL 为只写

◆ 输入捕获模式

捕获/比较模式寄存器 2（GP32C4Tn_CHMR2）																													
偏移地址：$02C_H$																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																I4FLT				IC4PRES		CC4SSEL		IC3FLT				IC3PRES		CC3SSEL	

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
I4FLT	Bit 15-12	R/W	**输入捕获 4 滤波器** 参考 I1FLT 描述
IC4PRES	Bit 11-10	R/W	**输入捕获 4 预分频器** 参考 IC1PRES 描述
CC4SSEL	Bit 9-8	R/W	**输入捕获 4 选择** 该位定义了通道和使用的输入的方向。 00：CC4 通道配置为**输出** 01：CC4 通道配置为输入，IC4 映射到 **TI4** 10：CC4 通道配置为输入，IC4 映射到 **TI3** 11：CC4 通道配置为输入，IC4 映射到 **TRC** 只有当内部触发输入是通过 TSSEL 位（GP32C4Tn_SMCON 寄存器）选择时，该模式才运行 注意： 当通道关闭（GP32C4Tn_CCEP 寄存器中的 CC4EN = '0'），CC4SSEL 为只写。
IC3FLT	Bit 7-4	R/W	**输入捕获 3 滤波器** 参考 I1FLT 描述
IC3PRES	Bit 3-2	R/W	**输入捕获 3 预分频器** 参考 IC1PRES 描述
CC3SSEL	Bit 1-0	R/W	**输入捕获 3 选择** 该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I3** 10：通道配置为输入，捕获源为 **I4** 11：通道配置为输入，捕获源为 ITn 或 I1 的双边沿检出 只有当内部触发输入是通过 TSSEL 位（GP32C4Tn_SMCON 寄存器）选择时，该模式才运行 注意：当通道关闭（GP32C4Tn_CCEP 寄存器中的 CC3EN = '0'），CC3SSEL 为只写。

22. 5. 2. 13 捕获/比较使能寄存器（GP32C4Tn_CCEP）

捕获/比较使能寄存器（GP32C4Tn_CCEP）																													
偏移地址：030_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		CC4POL	CC4EN	Reserved		CC3POL	CC3EN	Reserved		CC2POL	CC2EN	Reserved		CC1POL	CC1EN

Reserved	Bit 31-14	-	**保留，必须保持为复位值**
CC4POL	Bit 13	R/W	**捕获/比较 4 输出极性** 参考 CC1POL 描述
CC4EN	Bit 12	R/W	**捕获/比较 4 输出使能** 参考 CC1EN 描述
Reserved	Bit 11-10	-	保留，必须保持为复位值
CC3POL	Bit 9	R/W	**捕获/比较 3 输出极性** 参考 CC1POL 描述
CC3EN	Bit 8	R/W	**捕获/比较 3 输出使能** 参考 CC1EN 描述
Reserved	Bit 7-6	-	保留，必须保持为复位值
CC2POL	Bit 5	R/W	**捕获/比较 2 输出极性** 参考 CC1POL 描述
CC2EN	Bit 4	R/W	**捕获/比较 2 输出使能** 参考 CC1EN 描述
Reserved	Bit 3-2	-	保留，必须保持为复位值
CC1POL	Bit 1	R/W	**捕获/比较 1 输出极性** **通道配置为输出：** 0：CH1O 高有效 1：CH1O 低有效 **通道配置为输入：** CC1POL 为触发和捕获操作选择 I1 边沿检出和 I2 边沿检出的有效极性。 0：正向/上升沿 电路对 In 边沿检出的上升沿敏感（在复位，外部时钟或触发模式下，进行捕获或触发），In 边沿检出不反向（门控模式或编码器模式下，进行触发）。 1：反向/下降沿 电路对 In 边沿检出的下降沿敏感（在复位，外部时钟或触发模式下，进行捕获或触发），In 边沿检出反向（门控模式或编码器模式下，进行触发）。
CC1EN	Bit 0	R/W	**捕获/比较 1 输出使能** 0：关闭 - CH1O 无效。

			1：开启 - CH1O 为对应输出引脚上的输出信号，由 CC1EN 决定 **通道配置为输入：** 该位决定了计数值是否能捕获到输入捕获/比较寄存器 1（GP32C4Tn_CCVAL1）。 0：禁止捕获。 1：使能捕获。

22. 5. 2. 14 计数器寄存器（GP32C4Tn_COUNT）

计数器寄存器（GP32C4Tn_COUNT）																															
偏移地址：034_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTV																															

CNTV	Bit 31-0	R/W	计数值

22. 5. 2. 15 预分频寄存器（GP32C4Tn_PRES）

预分频寄存器（GP32C4Tn_PRES）																															
偏移地址：038_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PSCV															

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
PSCV	Bit 15-0	R/W	**预分频器值** 计数器时钟频率（CK_CNT）= f_{CK_PSC} /（PSCV [15:0] + 1） 每发生一次更新事件（包括当计数器由 GP32C4Tn_SGE 寄存器中的 SGU 位清零或当配置为复位模式时，通过触发控制器清零），PSCV 包含的值将被填入到有效的预分频寄存器内。

22. 5. 2. 16 自动重载寄存器（GP32C4Tn_AR）

自动重载寄存器（GP32C4Tn_AR）																															
偏移地址：$03C_H$																															
复位值：$11111111\_11111111\_11111111\_11111111_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARRV																															

ARRV	Bit 31-0	R/W	**自动重载值** ARRV 中的值将被载入实际的自动重载寄存器中。当自动重载值为空，计数器被屏蔽。

22. 5. 2. 17 捕获/比较寄存器 1（GP32C4Tn_CCVAL1）

捕获/比较寄存器 1（GP32C4T_ CCVAL 1）																															
偏移地址：044_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCRV1																															

CCRV1	Bit 31-0	R/W	**捕获/比较值 1** **如果通道 CCn 配置为输出：** CCRVn 中的值将被载入实际的捕获/比较寄存器中（预载值）。 如果在 GP32C4Tn_CHMRn 寄存器中的预载功能没有选中，CCRVn 中的值将被永久载入；否则，每当发生更新事件，预载值将会复制到有效的捕获/比较寄存器中。有效捕获/比较寄存器中包含的值将会与 GP32C4Tn_COUNT 中的值进行比较，并在 OCn 上输出。 **如果通道 CCn 配置为输入：** CCRVn 为由上一个输入捕获事件（ICn）传输的计数值。

22. 5. 2. 18 捕获/比较寄存器 2（GP32C4Tn_CCVAL2）

捕获/比较寄存器 2（GP32C4T_ CCVAL 2）																															
偏移地址：048_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCRV2																															

CCRV2	Bit 31-0	R/W	**捕获/比较值 2** 参考 CCRV1 描述

22. 5. 2. 19 捕获/比较寄存器 3（GP32C4Tn_CCVAL3）

捕获/比较寄存器 3（GP32C4T_ CCVAL 3）																															
偏移地址：$04C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRRV3																															

CCRV3	Bit 31-0	R/W	**捕获/比较值 3** 参考 CCRV1 描述

22. 5. 2. 20 捕获/比较寄存器 4（GP32C4Tn_CCVAL4）

捕获/比较寄存器 4（GP32C4T_ CCVAL 4）																															
偏移地址：050_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCRV4																															

CCRV4	Bit 31-0	R/W	**捕获/比较值 4** 参考 CCRV1 描述

22. 5. 2. 21 DMA使能寄存器（GP32C4Tn_DMAEN）

DMA 使能寄存器（GP32C4Tn_DMAEN）																												
偏移地址：058_H																												
复位值：$00000000\_00000000\_00000000\_00000000_B$																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									TRGDMA	COMDMA	CC4DMA	CC3DMA	CC2DMA	CC1DMA	UDMA

Reserved	Bit 31-7	-	**保留，必须保持为复位值**
TRGDMA	Bit 6	R/W	**触发DMA请求使能** 0：DMA请求禁止 1：DMA请求使能
COMDMA	Bit 5	R/W	**COM DMA访问使能** 0：DMA请求禁止 1：DMA请求使能
CC4DMA	Bit 4	R/W	**捕获/比较值 4 DMA 访问使能** 0：DMA 请求禁止 1：DMA 请求使能
CC3DMA	Bit 3	R/W	**捕获/比较值 3 DMA 访问使能** 0：DMA 请求禁止 1：DMA 请求使能
CC2DMA	Bit 2	R/W	**捕获/比较值 2 DMA 访问使能** 0：DMA 请求禁止 1：DMA 请求使能
CC1DMA	Bit 1	R/W	**捕获/比较值 1 DMA 访问使能** 0：DMA 请求禁止 1：DMA 请求使能
UDMA	Bit 0	R/W	**更新 DMA 请求使能** 0：DMA 请求禁止 1：DMA 请求使能

第23章 基本定时器（BS16T）

23. 1 概述

基本定时器（BS16T）包含一个 16 位自动重载计数器，该计数器由可配置的预分频器驱动。

通过使用定时器的分频器和 APB 时钟控制器的预分频功能，可对脉冲长度和波形周期进行数微妙到几毫秒的调整。

23. 2 主要特点

- 16 位自动加载递增计数器
- 16 位可编程预分频器，可对计数器工作时钟进行 1 到 65536 的任意分频（运行中也可以）
- 计数上溢更新事件产生中断/DMA 请求

23. 3 结构框图

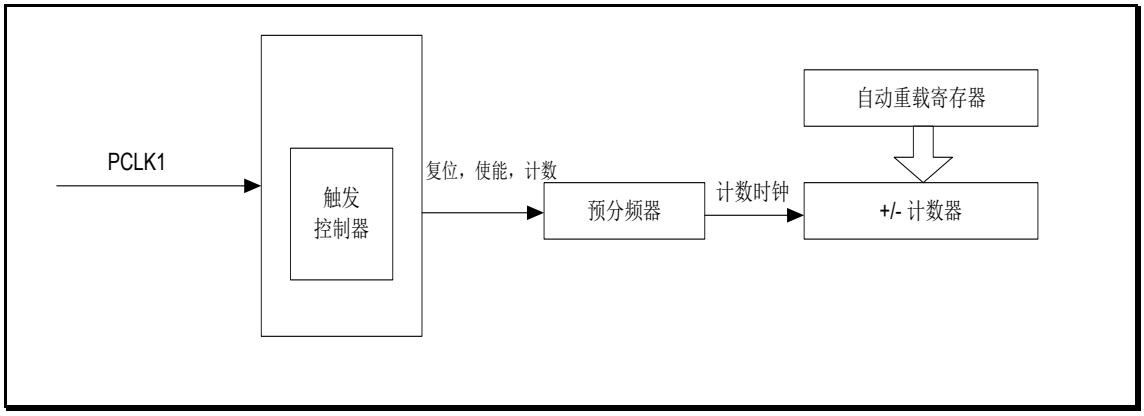


图 23-1 基本定时器结构框图

23. 4 功能描述

23. 4. 1 预分频器

定时器包含一个 16-bit 的计数器（BS16Tn_COUNT），计数时钟由预分频寄存器（BS16Tn_PRES）进行分频。计数周期由自动重载寄存器（BS16Tn_AR）设定。

自动重载寄存器（BS16Tn_AR）是一个可缓存的寄存器。当 BS16Tn_CON1 寄存器的 ARPEN 位复位时，BS16Tn_AR 寄存器重载功能失效，BS16Tn_AR 就是有效寄存器；ARPEN 置位时，BS16Tn_AR 寄存器具有重载功能，产生更新事件（UEV）时，加载值（BS16Tn_AR 寄存器值）更新到影子寄存器才有效。

当BS16Tn_CON1 寄存器中 DISUE 位为 0 时，计数器计数上溢时会产生更新事件（UEV）。同样，软件方式也可产生更新事件。BS16Tn_CON1 寄存器的 CNTEN 置位时，计数器开始计数。

注：计数器在 CNTEN 位置位 1 个时钟周期后开始计数。

预分频器可对定时器工作时钟进行 BS16Tn_PRES 寄存器值+1 次分频。由于 BS16Tn_PRES 是一个可重载寄存器，因此，定时器工作时可以对该寄存器进行修改，修改值在下次更新事件（UEV）后有效。

下图给出了定时器运行过程中改变预分频值时计数器的计数情况。

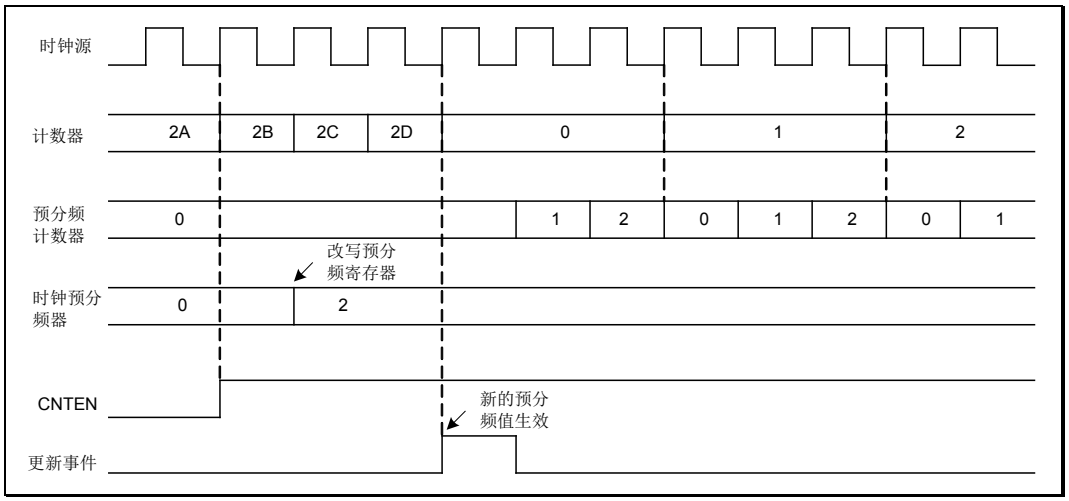


图 23-2 预分频值计数时序图

23.4.2 时钟源

计数时钟由内部时钟源（PCLK1）提供。

CNTEN 位（BS16Tn_CON1 寄存器）与 SGU 位（BS16Tn_SGE 寄存器）为实际控制位，这两个位只能软件修改（SGU 位除外，仍硬件自动清除）。一旦 CNTEN 位被写为'1'，预分频器就由内部 PCLK1 提供时钟。

23.4.3 递增计数模式

在递增计数模式中，计数器由 0 开始计数至自动重载值（BS16Tn_AR 寄存器中的值），然后从 0 开始重新计数并产生一个计数溢出事件。

软件置位 BS16Tn_CON1 寄存器中的 DISUE 位可关闭更新事件（UEV）的产生。更新事件（UEV）关闭，可避免向预载寄存器写新值的过程中更新影子寄存器。这种情况下，DISUE 位在写'0'之前都不会产生更新事件。正常产生更新事件后，计数器和预载计数器都是从 0 重新开始（但预分频值没有改变）。此外，若置位 BS16Tn_CON1 寄存器中的 UERSEL 位（更新请求选择），置位 SGU 位时会产生一次更新事件（UEV），但 UEVTIF 标志位不会置位（因此，不会触发中断或 DMA 请求）。

当更新事件发生时，所有寄存器都会被更新且更新标志位（BS16Tn_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）：

- ◇ 更新 BS16Tn_AR 寄存器的值到影子寄存器
- ◇ 更新 BS16Tn_PRES 寄存器的值到影子寄存器

23.4.4 调试模式

当微控制器进入调试模式（Cortex™-M 内核终止）,计数器可被设定停止计数。

23. 5 特殊功能寄存器

23. 5. 1 寄存器列表

BS16T 寄存器列表		
名称	偏移地址	描述
BS16Tn_CON1	000_H	控制寄存器 1
Reserved	004_H	保留
BS16Tn_IER	$00C_H$	中断使能寄存器
BS16Tn_IDR	010_H	中断禁止寄存器
BS16Tn_IVS	014_H	中断有效状态寄存器
BS16Tn_RIF	018_H	原始中断标志寄存器
BS16Tn_IFM	$01C_H$	中断标志屏蔽寄存器
BS16Tn_ICR	020_H	中断清零寄存器
BS16Tn_SGE	024_H	软件生成事件寄存器
BS16Tn_COUNT	034_H	计数器寄存器
BS16Tn_PRES	038_H	预分频寄存器
BS16Tn_AR	$03C_H$	自动重载寄存器
BS16Tn_DMAEN	058_H	DMA 使能寄存器

23. 5. 2 寄存器描述

23. 5. 2. 1 控制寄存器 1（BS16Tn_CON1）

控制寄存器 1（BS16Tn_CON1）

偏移地址：000_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-8	-	**保留，必须保持为复位值**
APREN	Bit7	R/W	**自动重载预载使能** 0：BS16Tn_AR 寄存器未缓冲 1：BS16Tn_AR 寄存器被装入缓冲器
Reserved	Bit 6-4	-	**保留，必须保持为复位值**
SPMEN	Bit 3	R/W	**单脉冲模式** 0：当发生更新事件时，计数器不停止。 1：当发生下一次更新事件（CNTEN 位清零）时，计数器停止。
UERSEL	Bit 2	R/W	**更新请求源** 该位由软件置 1 或清零，来选择 UEV 事件源。 0：如果更新中断或 DMA 请求使能，则下述任一事件都可产生更新中断或 DMA 请求： –计数器上溢 –设置 SGU 位 1：如果更新中断或 DMA 请求使能，仅计数器上溢才能产生更新中断或 DMA 请求中断
DISUE	Bit1	R/W	**更新禁止** 该位由软件置 1 或清零来使能/禁止 UEV 事件的产生。 0：UEV 使能。更新事件（UEV）由下列任一事件产生： – 计数器上溢 –设置 SGU 位 缓冲寄存器载入他们的预载值。 1：UEV 禁止。不产生更新事件，影子寄存器保持他们的值（ARRV， PSCV）。如果从从模式控制器接收到硬件复位，计数器和预分频器将被重新初始化。
CNTEN	Bit0	R/W	**计数器使能** 0：计数器禁止 1：计数器使能

23. 5. 2. 2 中断使能寄存器（BS16Tn_IER）

中断使能寄存器（BS16Tn_IER）																															
偏移地址：$00C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														UIT	

Reserved	Bit 31-1	-	保留，必须保持为复位值
UIT	Bit 0	W	更新中断使能 0：无效 1：使能

23. 5. 2. 3 中断禁止寄存器（BS16Tn_IDR）

中断禁止寄存器（BS16Tn_IDR）																															
偏移地址：010_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														UI	

Reserved	Bit 31-1	-	保留，必须保持为复位值
UI	Bit 0	W	更新中断禁止 0：无效 1：禁止

23. 5. 2. 4 中断有效状态寄存器（BS16Tn_IVS）

中断有效状态寄存器（BS16Tn_IVS）																															
偏移地址：014_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															UEI

Reserved	Bit 31-1	-	保留，必须保持为复位值
UEI	Bit 0	R	**更新中断有效状态** 0：禁止更新中断 1：使能更新中断 IER/IDR 写 1 来使能或禁止该位。

23. 5. 2. 5 原始中断标志寄存器（BS16Tn_RIF）

原始中断标志寄存器（BS16Tn_RIF）																															
偏移地址：018_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															UEVTIF

Reserved	Bit 31-1	-	保留，必须保持为复位值
UEVTIF	Bit 0	R	**更新中断标志** 如果更新中断使能，当发生更新事件，该标志位由硬件置起。对 BS16Tn_ICR 写 1 来清除原始中断 0：未发生更新 1：更新中断被挂起。当寄存器更新时，该位被硬件置起： –当计数器值发生上溢（若计数器=0，则更新）和当 BS16Tn_CON1 寄存器中 DISUE=0 –当使用 BS16Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时，如果 BS16Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0 –当 CNT 由触发事件来重新初始化，如果 BS16Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0。

23. 5. 2. 6 中断标志屏蔽寄存器（BS16Tn_IFM）

中断标志屏蔽寄存器（BS16Tn_IFM）																															
偏移地址：$01C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															UEI

Reserved	Bit 31-1	-	保留，必须保持为复位值
UEI	Bit 0	R	**更新中断标志屏蔽** 如果更新中断使能，当发生更新事件，该标志位由硬件置起。对 BS16Tn_ICR 写 1 来清除原始中断 0：未发生更新 1：更新中断被挂起。当寄存器更新时，该位被硬件置起： –当计数器值发生上溢（若计数器=0，则更新）和当 BS16Tn_CON1 寄存器中 DISUE=0 –当使用 BS16Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时，如果 BS16Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0 –当 CNT 由触发事件来重新初始化，如果 BS16Tn_CON1 寄存器中的 UERSEL=0 和 DISUE=0。

23. 5. 2. 7 中断清零寄存器（BS16Tn_ICR）

中断清零寄存器（BS16Tn_ICR）																															
偏移地址：020_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															UEIC

Reserved	Bit 31-1	-	**保留，必须保持为复位值**
UEIC	Bit 0	C_W1	**更新中断清零** 0：无效 1：更新中断清零（BS16Tn_RIF）

23. 5. 2. 8 软件生成事件寄存器（BS16Tn_SGE）

软件生成事件寄存器（BS16Tn_SGE）

偏移地址：024_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-1	-	**保留，必须保持为复位值**
SGU	Bit 0	W	**更新生成** 该位由软件设置，可由硬件自动清零. 0：无动作 1：重新初始化计数器，更新寄存器。注意，预分频器也会被清零（但预分频比不会受到影响）。

23. 5. 2. 9 计数器寄存器（BS16Tn_COUNT）

计数器寄存器（BS16Tn_COUNT）

偏移地址：034_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
CNTV	Bit 15-0	R/W	**计数值**

23. 5. 2. 10 预分频寄存器（BS16Tn_PRES）

预分频寄存器（BS16Tn_PRES）																															
偏移地址：038_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PSCV															

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
PSCV	Bit 15-0	R/W	**预分频器值** 计数器时钟频率（CK_CNT）= f_{CK_PSC} /（PSCV[15:0] + 1） 每发生一次更新事件（包括当计数器由 BS16Tn_SGE 寄存器中的 SGU 位清零或当配置为复位模式时，通过触发控制器清零），PSCV 包含的值将被填入到有效的预分频寄存器内。

23. 5. 2. 11 自动重载寄存器（BS16Tn_AR）

自动重载寄存器（BS16Tn_AR）																															
偏移地址：$03C_H$																															
复位值：$00000000\_00000000\_11111111\_11111111_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ARRV															

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
ARRV	Bit 15-0	R/W	**自动重载值** AR 中的值将被载入实际的自动重载寄存器中。 当自动重载值为空，计数器被屏蔽。

23. 5. 2. 12 DMA使能寄存器（BS16Tn_DMAEN）

DMA 使能寄存器（BS16Tn_DMAEN）

偏移地址：058_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-1	-	保留，必须保持为复位值
UDEN	Bit 0	R/W	**DMA 访问使能** 0：DMA 访问禁止 1：DMA 访问使能

第24章 实时时钟（RTC）

24. 1 概述

实时时钟（RTC）是一个独立的 BCD 码定时器，提供实时时钟和日历的计数功能。两个可编程闹钟和一个唤醒定时器可实现周期性的中断功能。

上电并软件使能后，无论芯片工作在何种模式，只要电源电压保持在正常工作范围内，RTC 可一直提供高精度计时工作。

24. 2 特性

- 仅上电复位有效，支持寄存器写保护，有效避免软件误操作
- 时钟源支持 LOSC、LRC、HOSC、HRC
- 提供时钟和日历功能：年、月、日、时、分、秒、星期
- 自动闰年识别，有效期 100 年（00-99）
- 12 小时和 24 小时模式设置可选
- 支持可编程的夏令时调整功能
- 两个可编程闹钟，支持闹钟匹配字段配置
- 一个可编程的定时器，并支持定时唤醒功能
- 可进行高精度数字校准，最高精度±0.0254 ppm
- 支持时间戳功能，在发生时间戳事件时保持时间戳时间和日期
- 支持两路侵入检测功能
- 支持 128Bytes 备份寄存器，在侵入事件发生时复位所有备份寄存器
- 低功耗设计：在电源 STANDBY 模式下能保证时钟和日历的精度

24. 3 结构框图

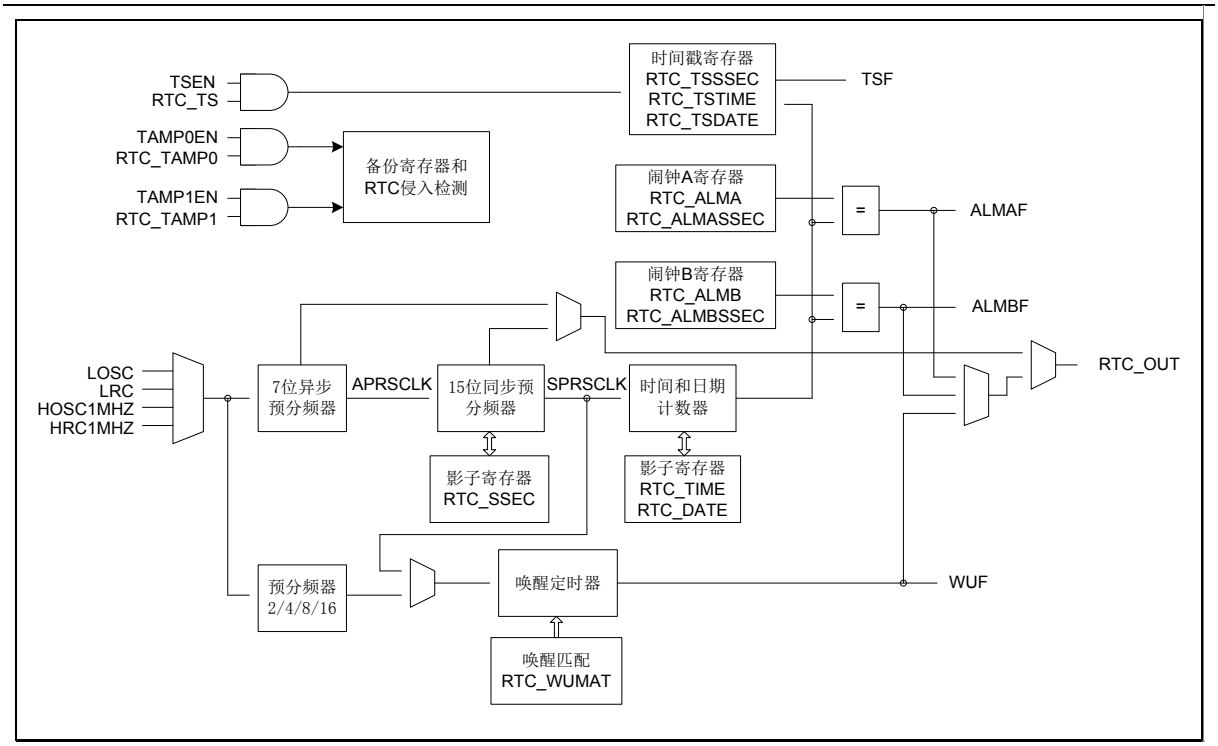


图 24-1　电路结构框图

24. 4 功能描述

24. 4. 1 时钟和预分频

RTC 时钟源通过备份域外设时钟控制寄存器 BKPC_PCCR.RTCCS 进行选择，可选择为：

- ◇ 32768Hz 低速时钟 LRC
- ◇ 32768Hz 低速时钟 LOSC（停振自动切换至 LRC）
- ◇ 高速时钟 HRC 分频至 1MHz
- ◇ 高速时钟 HOSC 分频至 1MHz

RTC 支持工作时钟预分频，预分频器分为两个可编程的预分频器：

- ◇ 7 位异步预分频器
- ◇ 15 位同步预分频器

两个预分频器可灵活使用，使用较高的异步分频系数可降低 RTC 运行功耗，但使用较高的同步分频系数可提升数字校准的精度。

APRSCLK 时钟频率计算公式为：

$$F_{APRSCLK} = \frac{F_{RTCCLK}}{APRS+1}$$

APRSCLK 时钟用于亚秒寄存器 RTC_SSEC 计数器提供时钟。当该计数器计数匹配同步分频数时会清零并重新开始计数。

SPRSCLK 时钟频率计算公式为：

$$F_{SPRSCLK} = \frac{F_{RTCCLK}}{(APRS+1)\times(SPRS+1)}$$

SPRSCLK 时钟用于更新日历，也可以用作 16 位唤醒定时器的计数时钟。

使用 32768Hz 频率的时钟获得频率为 1Hz 的时钟 SPRSCLK，可将异步分频系数设置为 1，并将同步分频系数设置为 32768。使用 1MHz 频率的时钟获得频率为 1Hz 的时钟 SPRSCLK，可将异步分频系数设置为 32，并将同步分频系数设置为 31250。

24.4.2 时钟和日历

RTC 时间和日历寄存器可通过对应的影子寄存器进行访问，或可设置为直接访问时间和日历寄存器以避免时钟同步造成的延时。

时间和日历寄存器包括：

◇ 亚秒寄存器 RTC_SSEC

◇ 时间寄存器 RTC_TIME

◇ 日期寄存器 RTC_DATE

每两个 RTCCLK 时钟周期便将时间和日历寄存器复制至相应的影子寄存器中。在 STOP 和 STANDBY 模式下不会执行该操作，直至退出这两种模式后最迟两个 RTCCLK 时钟周期后更新。

在默认情况下，当读取时间和日历寄存器时，会访问到影子寄存器内容。也可通过将 RTC_CON.SHDBP 置 1 来旁路影子寄存器而直接访问到时间和日历寄存器。读取影子寄存器时，APB2 时钟频率必须大于 RTCCLK 时钟频率的 4 倍以上。

24.4.3 可编程闹钟

RTC 提供两个可编程闹钟 A 和 B。

通过将寄存器 RTC_CON.ALMAEN 和寄存器 RTC_CON.ALMBEN 置 1 来使能闹钟 A 和闹钟 B。如果时间和日历的值分别与闹钟寄存器 RTC_ALMA/RTC_ALMASSEC 和 RTC_ALMB/RTC_ALMBSSEC 中的配置值相匹配，则标志位 RTC_IFR. ALMAF 和 RTC_IFR. ALMBF 会相应的置起。

通过闹钟寄存器 RTC_ALMA 和 RTC_ALMB 中的 xMSK 位，以及 RTC_ALMASSEC 和 RTC_ALMBSSEC 中的 SSECM 位单独选择屏蔽相应的字段。

闹钟 A 和闹钟 B 可连接到 RTCO 端口输出，用户可通过配置 RTC_CON.EOS 进行使能，通过配置 RTC_CON. POL 选择输出的极性。

24.4.4 周期性唤醒

RTC 提供一个 16 位周期性唤醒定时器，并可通过配置扩展至 17 位。可通过将 RTC_CON. WUTE 置 1 使能唤醒功能。

唤醒定时器的时钟输入可选择为两种：

◇ RTCCLK 的分频时钟，可选择为 2/4/8/16 分频

- 当 RTCCLK 选择为 32768Hz 时，可配置的唤醒周期范围可选择为 122us 至 32s，分辨率为 61us

◇ SPRSCLK 时钟（通常为 1Hz 内部时钟）

- 当 SPRSCLK 为 1Hz，并且 RTC_CON.WUCKS[2:1]配置为 0b10 时，可配置的唤醒周期范围可选择为 1s 至 18h，分辨率为 1s
- 当 SPRSCLK 为 1Hz，并且 RTC_CON.WUCKS[2:1]配置为 0b11 时，可配置的唤醒周期范围可选择为 18h 至 36h，分辨率为 1s

如果唤醒定时器计数与 RTC_WUMAT 寄存器值匹配后，标志位 RTC_IFR. WUF 被置起，并且定时器清零后重新开始计数。系统复位和低功耗模式（SLEEP、STOP 和 STANDBY）

对唤醒定时器均没有任何影响。

定时器溢出标志可连接到 RTCO 端口输出，用户可通过配置 RTC_CON.EOS 进行使能，通过配置 RTC_CON. POL 选择输出的极性。

24. 4. 5 数字校准

RTC 提供了一种数字校准的方法，通过增加或减少同步分频器的系数，可对 RTC 时钟周期的偏差进行补偿。

通过将寄存器 RTC_CALCON.CALEN 置 1 使能 RTC 数字校准功能，通过配置 RTC_CALCON.CALP 选择数字校准的间隔周期。数字校准将在所选间隔周期的最后一秒进行补偿。

通过配置寄存器 RTC_CALDR.VAL 来选择数字校准时增加或减少同步分频器的系数值，寄存器 RTC_CALDR.VAL 为 16 位补码形式存放，其中 Bit15 为符号位，为 0 时会增加同步分频器的系数，RTC 时间会相应的变慢，为 1 时会减少同步分频器的系数，RTC 时间会相应的变快。

例如：RTC 时钟源选择 32768Hz 时钟，每秒比标准时间慢 150us，选择每隔 20s 校准

RTC 时钟实际周期为 $T_{RTCCLK} = \frac{10^6 + 150}{32768} \approx 30.522156\ \text{us}$

需校准的周期数 $T = -\frac{150 \times 20}{T_{RTCCLK}} = -98.3 \approx -98$（对应补码为 0xFF9E）

校准寄存器 RTC_CALDR. VAL 需要配置为 0xFF9E

校准前偏差 $e = \frac{T_{RTCCLK} \times 32768 \times 20}{20 \times 10^6} - 1 = +150\text{ppm}$

校准后偏差 $e' = \frac{T_{RTCCLK} \times (32768 \times 20 + T)}{20 \times 10^6} - 1 = +0.441\text{ppm}$

24. 4. 6 时间戳功能

RTC 提供了时间和日历的时间戳功能，通过将寄存器 RTC_CON.TSEN 置 1 可使能时间戳功能。

当时间戳功能复用端口上检测到时间戳事件时，实时的时间和日历（包括亚秒、时间和日期寄存器）可被保存到时间戳寄存器中。时间戳寄存器包括：

- ◇ 时间戳亚秒寄存器 RTC_TSSSEC
- ◇ 时间戳时间寄存器 RTC_TSTIME
- ◇ 时间戳日期寄存器 RTC_TSDATE

发生时间戳事件时，标志位 RTC_IFR.TSF 将被置起，通过软件可将该标志位清零。若该标志位为 1 期间又检测到新的时间戳事件时，时间戳溢出标志位 RTC_IFR.TSOVF 将被置起。

侵入事件的发生也可将时间和日历记录到时间戳寄存器中，同时也会置起时间戳标志位。

24. 4. 7 侵入检测功能

RTC 提供了侵入检测的功能，通过对侵入检测复用的端口电平边沿或带滤波的电平检测，可产生侵入事件。将寄存器 RTC_TAMPCON.TAMPxEN 置 1 可使能侵入检测的功能，配置寄存器 RTC_TAMPCON.TAMPxLV 选择侵入电平的极性。

可通过配置寄存器 RTC_TAMPCON.TAMPFLT 选择是否需要对侵入电平进行滤波，并且选择滤波的采样次数，通过配置寄存器 RTC_TAMPCON.TAMPCKS 选择采样时钟的频率。

侵入检测事件也可配置为同时触发时间戳事件，可通过将寄存器 RTC_TAMPCON.TAMPTS 置 1 来使能。

发生侵入检测事件时，标志位 RTC_IFR.TAMPxF 将被置起，通过软件可将该标志位清零。

侵入检测事件可同时将 RTC 备份寄存器 RTC_BKPxR 全部清零。

24. 4. 8 时钟输出

RTC 可提供 RTC 时钟分频连接到 RTCO 端口输出，用户可将寄存器 RTC_CON.CKOE 置 1 进行使能，同时需要将寄存器 RTC_CON.EOS 配置为 0。通过配置寄存器 RTC_CON.CKOS 选择输出时钟的频率。

当选择输出为精确 1Hz 时，需先使能 PLL2 并等待其稳定。

24. 5 基本配置

24. 5. 1 RTC写保护

为避免程序的异常运行对 RTC 的误操作，RTC 写保护寄存器 RTC_WPR 用于阻止程序对 RTC 其它寄存器的误写入。该寄存器保护范围为除 RTC_WPR 寄存器外的 RTC 模块所有寄存器。

RTC_WPR 寄存器为虚拟寄存器。要对 RTC 其它寄存器进行写操作时，需先对 RTC_WPR 寄存器写 0x55AAAA55，之后可对 RTC 其它寄存器进行写操作。对 RTC_WPR 寄存器写入其他值重新进入写保护状态，写保护状态下对 RTC 寄存器进行的写操作将被忽略。

可以通过读取 RTC_WPR 寄存器确认 RTC 是否处于写保护状态，读出值为 0x00000000，表示当前可对 RTC 寄存器进行写操作；读出值为 0x00000001 表示 RTC 处于写保护状态。RTC_WPR 寄存器无其它读出值。

24. 5. 2 RTC校准写保护

为避免程序的异常运行对 RTC 校准的误操作，RTC 校准写保护寄存器 RTC_CALWPR 用于阻止程序对 RTC 校准寄存器的误写入。该寄存器保护范围为除 RTC_CALWPR 寄存器外的 RTC 校准相关的所有寄存器。

RTC_CALWPR 寄存器为虚拟寄存器。要对 RTC 其它寄存器进行写操作时，需先对 RTC_CALWPR 寄存器写 0x699655AA，之后可对 RTC 其它寄存器进行写操作。对 RTC_CALWPR 寄存器写入其他值重新进入校准写保护状态，校准写保护状态下对 RTC 寄存器进行的写操作将被忽略。

可以通过读取 RTC_CALWPR 寄存器确认 RTC 是否处于校准写保护状态，读出值为 0x00000000，表示当前可对 RTC 寄存器进行写操作；读出值为 0x00000001 表示 RTC 处于校准写保护状态。RTC_CALWPR 寄存器无其它读出值。

注：校准相关寄存器需要同时解除 RTC 写保护和 RTC 校准写保护后才可进行正常写入操作

24. 5. 3 时间和日历初始化

由于 APB2 总线时钟与 RTC 时钟异步，因此 RTC 时间和日历计数器需要通过影子寄存器进行读写操作。

时间和日历初始化配置步骤如下：

1. 配置异步预分频系数和同步预分频系数
2. 将寄存器 RTC_CON. GO 置 1 使能 RTC 计数器和分频器
3. 配置寄存器 RTC_CON. HFM 选择时间格式（12 或 24 小时制）
4. 在时间和日期影子寄存器（RTC_TIME 和 RTC_DATE）中加载初始的时间和日期值
5. 通过判断标志位 RTC_CON.BUSY 等待时间和日期寄存器写入同步完成

若 RTC 运行期间需要修改时间和日期，可重复以上 3~5 步骤操作。

时间和日期寄存器数据格式采用 BCD 编码。其计数范围为：

◇ 秒计数范围从 00 到 59，进位到分钟后从 59 变为 00。

◇ 分钟计数范围从 00 到 59，进位到小时后从 59 变为 00。

◇ 小时计数范围根据配置寄存器 RTC_CON.HFM 的设置选择 12 或 24 小时制。

- 选择 12 小时制时，分钟进位后小时从 PM11 更新到 AM12 或 AM11 更新到 PM12。
- 选择 24 小时制时，分钟进位后小时从 23 到更新 00。

◇ 星期计数器为 3 位计数器，数值为 0-6，初始值可配置。

◇ 日计数按照每月最后一天加 1 进位到下月，日计数范围按月分为：

- 一、三、五、七、八、十、十二月从 1 到 31；
- 四、六、九、十一月从 1 到 30；
- 二月（普通年份）从 1 到 28；二月（闰年）从 1 到 29；

◇ 月计数范围从 1 到 12，进位到年后从 12 变为 1；

◇ 年计数范围从 00 到 99（00，04，08，…，92，96 为闰年），99 后进位到 00。

12 或 24 小时制小时格式对照表如下：

24 小时模式	12 小时模式	24 小时模式	12 小时模式
00	12(AM12)	12	12(PM12)
01	01(AM1)	13	01(PM1)
02	02(AM2)	14	02(PM2)
03	03(AM3)	15	03(PM3)
04	04(AM4)	16	04(PM4)
05	05(AM5)	17	05(PM5)
06	06(AM6)	18	06(PM6)
07	07(AM7)	19	07(PM7)
08	08(AM8)	20	08(PM8)
09	09(AM9)	21	09(PM9)
10	10(AM10)	22	10(PM10)
11	11(AM11)	23	11(PM11)

表 24-1　小时格式对照表

24. 5. 4　夏令时

RTC 支持夏令时的软件触发调整。通过将寄存器 RTC_CON.SUB1H 和 RTC_CON.ADD1H 置 1 可触发对当前时间减少或增加 1 小时进行调整，而无需进行重新初始化时间操作。寄存器 RTC_CON.SUB1H 和 RTC_CON.ADD1H 触发后硬件自动清零。

软件可实现配置夏令时选择寄存器 RTC_CON.DSTS 选择当前是否为夏令时时间。触发完成后需要更新夏令时选择寄存器，以便之后软件可查询该寄存器值判断是否操作过夏令时时间。

当选择为冬季时间时，软件可触发时间增加 1 小时，触发时间减少 1 小时功能被禁止。当选择为夏季时间时，软件可触发时间减少 1 小时，触发时间增加 1 小时功能被禁止。

若不需要启用夏令时，则无需对这些寄存器进行操作。

24. 5. 5 亚秒调整

RTC 可实现与远程高精度的时钟同步。在读取亚秒寄存器 RTC_SSEC 后，即可计算 RTC 与远程时钟的时间偏差。使用亚秒调整寄存器 RTC_SSECTR 可对 RTC 进行亚秒级修正，修正的最大偏差为 1s，修正的精度为 ASPRCLK 周期值。

亚秒寄存器 RTC_SSEC 的值实际为同步预分频器的计数值，当对亚秒调整寄存器 RTC_SSECTR 进行写入操作时触发亚秒调整。

24. 6 RTC中断

RTC 模块支持以下中断源

- ◇ 6 个周期中断：年、月、日、时、分、秒中断
- ◇ 2 个闹钟中断：闹钟 A 和闹钟 B 中断
- ◇ 时间戳和时间戳溢出中断
- ◇ 2 个侵入检测中断：侵入检测 0 和侵入检测 1
- ◇ 周期性唤醒中断
- ◇ 寄存器同步完成中断和亚秒调整完成中断

每个中断源都有独立的使能位，使能位影响该中断是否产生 IRQ 中断请求，而不影响中断功能。即关闭相应中断使能，标志位仍可用于相应功能查询。当有多个中断使能时，各中断经过“或”逻辑产生 IRQ 中断请求。即任何一个被使能的中断产生中断事件时，均产生 IRQ 中断请求，且只有将所有的产生中断事件的中断标志清零后，IRQ 中断请求才解除。

24. 7 特殊功能寄存器

24. 7. 1 寄存器列表

RTC 寄存器列表		
名称	偏移地址	描述
RTC_WPR	000_H	RTC 写保护寄存器
RTC_CON	004_H	RTC 控制寄存器
RTC_PSR	008_H	RTC 预分频寄存器
RTC_TAMPCON	$00C_H$	RTC 侵入控制寄存器
RTC_TIME	010_H	RTC 时间寄存器
RTC_DATE	014_H	RTC 日期寄存器
RTC_SSEC	018_H	RTC 亚秒寄存器
RTC_WUMAT	$01C_H$	RTC 唤醒匹配寄存器
RTC_ALMA	020_H	RTC 闹钟 A 寄存器
RTC_ALMB	024_H	RTC 闹钟 B 寄存器
RTC_ALMASSEC	028_H	RTC 闹钟 A 亚秒寄存器
RTC_ALMBSSEC	$02C_H$	RTC 闹钟 B 亚秒寄存器
RTC_TSTIME	030_H	RTC 时间戳时间寄存器
RTC_TSDATE	034_H	RTC 时间戳日期寄存器
RTC_TSSSEC	038_H	RTC 时间戳亚秒寄存器
RTC_SSECTR	$03C_H$	RTC 亚秒调整寄存器
RTC_IER	040_H	RTC 中断使能寄存器
RTC_IFR	044_H	RTC 中断标志寄存器
RTC_IFCR	048_H	RTC 中断标志清零寄存器
RTC_ISR	$04C_H$	RTC 中断状态寄存器
RTC_CALWPR	050_H	RTC 校准写保护寄存器
RTC_CALCON	054_H	RTC 校准控制寄存器
RTC_CALDR	058_H	RTC 校准值寄存器
Reserved	$05C_H$~$0FC_H$	—
RTC_BKPxR	100_H~$17C_H$	RTC 备份寄存器 0~31

24. 7. 2 寄存器描述

24. 7. 2. 1 RTC写保护寄存器（RTC_WPR）

RTC 写保护寄存器（RTC_WPR）																															
偏移地址：000_H																															
复位值：$00000000\_00000000\_00000000\_00000001_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															WP

Reserved	Bit 31-1	—	保留
WP	Bit 0	R	**写保护状态位** 0：写保护解除 1：写保护有效

注：对该寄存器写入 0x55AAAA55 解除写保护，写入其他值开启写保护。该寄存器保护除自身外的 RTC 所有区域。

24. 7. 2. 2 RTC控制寄存器（RTC_CON）

RTC 控制寄存器（RTC_CON）																															
偏移地址：004_H																															
上电复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						SSEC	BUSY	Reserved	POL	EOS		CKOS			CKOE	WUCKS			WUTE	Reserved	DSTS	SUB1H	ADD1H	TSPIN	TSSEL	TSEN	SHDBP	HFM	ALMBEN	ALMAEN	GO

Reserved	Bit 31-26	—	**保留**
SSEC	Bit 25	R	**亚秒调整状态位** 0：亚秒调整完成 1：亚秒调整正在进行
BUSY	Bit 24	R	**寄存器同步状态位** 0：寄存器同步完成 1：寄存器同步正在进行
Reserved	Bit 23	—	**保留**
POL	Bit 22	R/W	**输出极性选择位** 0：高电平有效（正常极性） 1：低电平有效（相反极性） 注：极性选择只影响闹钟和唤醒输出，对校准输出无效
EOS	Bit 21-20	R/W	**事件输出选择位** 00：禁止输出 01：使能闹钟 A 输出 10：使能闹钟 B 输出 11：使能唤醒输出
CKOS	Bit 19-17	R/W	**时钟输出选择位** 000：32768Hz 001：1024Hz 010：32Hz 011：1Hz 100：数字校准后 1Hz 101：精确 1Hz 其他：保留 注 1：RTCCLK 使用 LOSC（32768Hz），且 APRS=0x0，SPRS=0x7FFF 注 2:选择精确 1Hz 前需使能 PLL2 并等待其稳定
CKOE	Bit 16	R/W	**时钟输出使能位** 0：禁止 1：使能 注：当 EOS 配置为非 00 时，校准输出被硬件强

			制禁止
WUCKS	Bit 15-13	R/W	**唤醒定时器时钟选择位** 000：RTCCLK/16 001：RTCCLK/8 010：RTCCLK/4 011：RTCCLK/2 10x：选择 SPRSCLK 11x：选择 SPRSCLK 并将 WUT 计数值增加 2^{16}
WUTE	Bit 12	R/W	**唤醒定时器使能位** 0：禁止 1：使能
Reserved	Bit 11	—	**保留**
DSTS	Bit 10	R/W	**夏令时选择位** 0：当前为冬季或不启用夏令时 1：当前为夏季
SUB1H	Bit 9	W1	**冬季时间更改位** 0：无操作 1：当前时钟减少1小时 注：当DSTS=0时，硬件强制禁止
ADD1H	Bit 8	W1	**夏季时间更改位** 0：无操作 1：当前时钟增加1小时 注：当DSTS=1时，硬件强制禁止
TSPIN	Bit 7	R/W	**时间戳信号管脚选择位** 0：TAMPER0 管脚 1：TAMPER1 管脚
TSSEL	Bit 6	R/W	**时间戳边沿选择位** 0：上升沿 1：下降沿
TSEN	Bit 5	R/W	**时间戳使能位** 0：禁止 1：使能
SHDBP	Bit 4	R/W	**影子寄存器模式选择位** 0：读取时间和日期时直接从影子寄存器读取 1：读取时间和日期时旁路影子寄存器 注 1：涉及到的寄存器有 RTC_TIME、RTC_DATE 和 RTC_SSEC 注 2：如果 APB 时钟频率小于 RTCCLK 的 8 倍时，必须将此位配置为 1
HFM	Bit 3	R/W	**小时格式选择位** 0：24 小时制 1：12 小时制（AM/PM）
ALMBEN	Bit 2	R/W	**闹钟 B 使能位**

			0：禁止 1：使能
ALMAEN	Bit 1	R/W	**闹钟 A 使能位** 0：禁止 1：使能
GO	Bit 0	R/W	**RTC 运行配置位** 0：停止 1：运行 注：该位可由软件置 1，RTC 开始运行，软件无法对其清 0。该位仅在上电时被复位。

24. 7. 2. 3 RTC预分频寄存器（RTC_PSR）

RTC 预分频寄存器（RTC_PSR）

偏移地址：008_H

上电复位值：$00000000\_00000000\_01111111\_11111111_B$

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

Reserved	Bit 31-23	—	保留
APRS	Bit 22-16	R/W	**RTC 异步预分频系数**
Reserved	Bit 15	—	保留
SPRS	Bit 14-0	R/W	**RTC 同步预分频系数**

24. 7. 2. 4 RTC侵入控制寄存器（RTC_TAMPCON）

RTC 侵入控制寄存器（RTC_TAMPCON）

偏移地址：$00C_H$

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										TAMPFLT		TAMPCKS			TAMPTS	Reserved						TAMP1LV	TAMP1EN	Reserved						TAMP0LV	TAMP0EN

Reserved	Bit 31-22	—	**保留**
TAMPFLT	Bit 21-20	R/W	**侵入电平滤波选择位** 00：有效电平直接激活侵入事件 01：在有效电平上连续 2 次采样后激活侵入事件 10：在有效电平上连续 4 次采样后激活侵入事件 11：在有效电平上连续 8 次采样后激活侵入事件
TAMPCKS	Bit 19-17	R/W	**侵入采样时钟选择位** 000：RTCCLK/32768 001：RTCCLK/16384 010：RTCCLK/8192 011：RTCCLK/4096 100：RTCCLK/2048 101：RTCCLK/1024 110：RTCCLK/512 111：RTCCLK/256
TAMPTS	Bit 16	R/W	**侵入事件触发时间戳选择位** 0：发生侵入事件时不触发时间戳 1：发生侵入事件时触发时间戳
Reserved	Bit 15-10	—	**保留**
TAMP1LV	Bit 9	R/W	**侵入 1 有效电平选择位** 0：低电平 1：高电平
TAMP1EN	Bit 8	R/W	**侵入 1 检测使能位** 0：禁止 1：使能
Reserved	Bit 7-2	—	**保留**
TAMP0LV	Bit 1	R/W	**侵入 0 有效电平选择位** 0：低电平 1：高电平
TAMP0EN	Bit 0	R/W	**侵入 0 检测使能位** 0：禁止 1：使能

24. 7. 2. 5 RTC时间寄存器（RTC_TIME）

RTC 时间寄存器（RTC_TIME）

偏移地址：010_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-23	—	保留
PM	Bit 22	R/W	**AM/PM 符号位** 0：AM 或 24 小时制 1：PM
HRT	Bit 21-20	R/W	**小时十位** 注：表示为 BCD 格式
HRU	Bit 19-16	R/W	**小时个位** 注：表示为 BCD 格式
Reserved	Bit 15	—	保留
MINT	Bit 14-12	R/W	**分钟十位** 注：表示为 BCD 格式
MINU	Bit 11-8	R/W	**分钟个位** 注：表示为 BCD 格式
Reserved	Bit 7	—	保留
SECT	Bit 6-4	R/W	**秒十位** 注：表示为 BCD 格式
SECU	Bit 3-0	R/W	**秒个位** 注：表示为 BCD 格式

24. 7. 2. 6 RTC日期寄存器（RTC_DATE）

RTC 日期寄存器（RTC_DATE）

偏移地址：014_H

复位值：$00000000\_00000000\_00000001\_00000001_B$

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

Reserved	Bit 31-27	—	**保留**
WD	Bit 26-24	R/W	**星期** 000：周日 001：周一 …… 110：周六 111：禁止
YRT	Bit 23-20	R/W	**年十位** 注：表示为 BCD 格式
YRU	Bit 19-16	R/W	**年个位** 注：表示为 BCD 格式
Reserved	Bit 15-13	—	**保留**
MONT	Bit 12	R/W	**月十位** 注：表示为 BCD 格式
MONU	Bit 11-8	R/W	**月个位** 注：表示为 BCD 格式
Reserved	Bit 7-6	—	**保留**
DAYT	Bit 5-4	R/W	**日十位** 注：表示为 BCD 格式
DAYU	Bit 3-0	R/W	**日个位** 注：表示为 BCD 格式

24. 7. 2. 7 RTC亚秒寄存器（RTC_SSEC）

RTC 亚秒寄存器（RTC_SSEC）

偏移地址：018_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	**保留**
VAL	Bit 15-0	R	**亚秒值** 该位表示同步预分频计数器的值

24. 7. 2. 8 RTC唤醒匹配寄存器（RTC_WUMAT）

RTC 唤醒匹配寄存器（RTC_WUMAT）

偏移地址：$01C_H$

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	**保留**
VAL	Bit 15-0	R/W	**RTC 唤醒定时器匹配值**

24. 7. 2. 9 RTC闹钟A寄存器（RTC_ALMA）

RTC 闹钟 A 寄存器（RTC_ALMA）

偏移地址：020_H

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDS	DAWD							HRMSK	PM	HRT		HRU				MINMSK	MINT			MINU				SECMSK	SECT			SECU			

WDS	Bit 31	R/W	**闹钟匹配星期选择位** 0：闹钟与日期匹配，Bit29-24（DAYU/DAYT）表示日期，Bit30（DAYMSK）表示日期掩码 1：闹钟与星期匹配，Bit30-24（DAWD）表示星期匹配使能位
DAWD	Bit 30-24	R/W	**闹钟日期匹配（当 WDS=0 时）** Bit 27-24 DAYU：日期个位（表示为 BCD 格式） Bit 29-28 DAYT：日期十位（表示为 BCD 格式） Bit 30 DAYMSK：闹钟日期掩码 0：闹钟匹配时日期有效 1：闹钟匹配时与日期无关 **闹钟星期匹配（当 WDS=1 时）** Bit 30 WDE0：星期日匹配使能位 Bit 29 WDE1：星期一匹配使能位 Bit 28 WDE2：星期二匹配使能位 Bit 27 WDE3：星期三匹配使能位 Bit 26 WDE4：星期四匹配使能位 Bit 25 WDE5：星期五匹配使能位 Bit 24 WDE6：星期六匹配使能位 注：当相应位为 1 时，闹钟将匹配相应的星期
HRMSK	Bit 23	R/W	**闹钟小时掩码** 0：闹钟匹配时小时有效 1：闹钟匹配时与小时无关
PM	Bit 22	R/W	**AM/PM 符号位** 0：AM 或 24 小时制 1：PM
HRT	Bit 21-20	R/W	**小时的十位** 注：表示为 BCD 格式
HRU	Bit 19-16	R/W	**小时的个位** 注：表示为 BCD 格式
MINMSK	Bit 15	R/W	**闹钟分钟掩码** 0：闹钟匹配时分钟有效 1：闹钟匹配时与分钟无关

MINT	Bit 14-12	R/W	**分钟的十位** 注：表示为 BCD 格式
MINU	Bit 11-8	R/W	**分钟的个位** 注：表示为 BCD 格式
SECMSK	Bit 7	R/W	**闹钟秒掩码** 0：闹钟匹配时秒有效 1：闹钟匹配时与秒无关
SECT	Bit 6-4	R/W	**秒的十位** 注：表示为 BCD 格式
SECU	Bit 3-0	R/W	**秒的个位** 注：表示为 BCD 格式

24. 7. 2. 10 RTC闹钟B寄存器（RTC_ALMB）

RTC 闹钟 B 寄存器（RTC_ALMB）

偏移地址：024_H

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDS	DAWD							HRMSK	PM	HRT		HRU				MINMSK	MINT			MINU				SECMSK	SECT			SECU			

WDS	Bit 31	R/W	**闹钟匹配星期选择位** 0：闹钟与日期匹配，Bit29-24（DAYU/DAYT）表示日期，Bit30（DAYMSK）表示日期掩码 1：闹钟与星期匹配，Bit30-24（DAWD）表示星期匹配使能位
DAWD	Bit 30-24	R/W	**闹钟日期匹配（当 WDS=0 时）** Bit 27-24 DAYU：日期个位（表示为 BCD 格式） Bit 29-28 DAYT：日期十位（表示为 BCD 格式） Bit 30 DAYMSK：闹钟日期掩码 0：闹钟匹配时日期有效 1：闹钟匹配时与日期无关 **闹钟星期匹配（当 WDS=1 时）** Bit 30 WDE0：星期日匹配使能位 Bit 29 WDE1：星期一匹配使能位 Bit 28 WDE2：星期二匹配使能位 Bit 27 WDE3：星期三匹配使能位 Bit 26 WDE4：星期四匹配使能位 Bit 25 WDE5：星期五匹配使能位 Bit 24 WDE6：星期六匹配使能位 注：当相应位为 1 时，闹钟将匹配相应的星期
HRMSK	Bit 23	R/W	**闹钟小时掩码** 0：闹钟匹配时小时有效 1：闹钟匹配时与小时无关
PM	Bit 22	R/W	**AM/PM 符号位** 0：AM 或 24 小时制 1：PM
HRT	Bit 21-20	R/W	**小时的十位** 注：表示为 BCD 格式
HRU	Bit 19-16	R/W	**小时的个位** 注：表示为 BCD 格式
MINMSK	Bit 15	R/W	**闹钟分钟掩码** 0：闹钟匹配时分钟有效 1：闹钟匹配时与分钟无关

MINT	Bit 14-12	R/W	**分钟的十位** 注：表示为 BCD 格式
MINU	Bit 11-8	R/W	**分钟的个位** 注：表示为 BCD 格式
SECMSK	Bit 7	R/W	**闹钟秒掩码** 0：闹钟匹配时秒有效 1：闹钟匹配时与秒无关
SECT	Bit 6-4	R/W	**秒的十位** 注：表示为 BCD 格式
SECU	Bit 3-0	R/W	**秒的个位** 注：表示为 BCD 格式

24. 7. 2. 11 RTC闹钟A亚秒寄存器（RTC_ALMASSEC）

RTC 闹钟 A 亚秒寄存器（RTC_ALMASSEC）

偏移地址：028_H

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-28	—	**保留**
SSECM	Bit 27-24	R/W	**亚秒匹配掩码** 0000：亚秒不参与闹钟匹配 0001：亚秒值 Bit0 匹配 SSEC[0] 0010：亚秒值 Bit1-0 匹配 SSEC[1:0] 0011：亚秒值 Bit2-0 匹配 SSEC[2:0] …… 1101：亚秒值 Bit12-0 匹配 SSEC[12:0] 1110：亚秒值 Bit13-0 匹配 SSEC[13:0] 1111：亚秒值 Bit14-0 匹配 SSEC[14:0] 注：亚秒的其他位均与 0 匹配
Reserved	Bit 23-15	—	**保留**
SSEC	Bit 14-0	R/W	**亚秒值** 该值与同步预分频计数器的值匹配产生闹钟

24. 7. 2. 12 RTC闹钟B亚秒寄存器（RTC_ALMBSSEC）

RTC 闹钟 B 亚秒寄存器（RTC_ALMBSSEC）

偏移地址：$02C_H$

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-28	—	**保留**
SSECM	Bit 27-24	R/W	**亚秒匹配掩码** 0000：亚秒不参与闹钟匹配 0001：亚秒值 Bit0 匹配 SSEC[0] 0010：亚秒值 Bit1-0 匹配 SSEC[1:0] 0011：亚秒值 Bit2-0 匹配 SSEC[2:0] …… 1101：亚秒值 Bit12-0 匹配 SSEC[12:0] 1110：亚秒值 Bit13-0 匹配 SSEC[13:0] 1111：亚秒值 Bit14-0 匹配 SSEC[14:0] 注：亚秒的其他位均与 0 匹配
Reserved	Bit 23-15	—	**保留**
SSEC	Bit 14-0	R/W	**亚秒值** 该值与同步预分频计数器的值匹配产生闹钟

24. 7. 2. 13 RTC时间戳时间寄存器（RTC_TSTIME）

RTC 时间戳时间寄存器（RTC_TSTIME）

偏移地址：030_H

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-23	—	**保留**
PM	Bit 22	R	**AM/PM 符号位** 0：AM 或 24 小时制 1：PM
HRT	Bit 21-20	R	**小时计数十位** 注：表示为 BCD 格式
HRU	Bit 19-16	R	**小时计数个位** 注：表示为 BCD 格式
Reserved	Bit 15	—	**保留**
MINT	Bit 14-12	R	**分钟计数十位** 注：表示为 BCD 格式
MINU	Bit 11-8	R	**分钟计数个位** 注：表示为 BCD 格式
Reserved	Bit 7	—	**保留**
SECT	Bit 6-4	R	**秒计数十位** 注：表示为 BCD 格式
SECU	Bit 3-0	R	**秒计数个位** 注：表示为 BCD 格式

24. 7. 2. 14 RTC时间戳日期寄存器（RTC_TSDATE）

RTC 时间戳日期寄存器（RTC_TSDATE）

偏移地址：034_H

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
WD	Bit 26-24	R	**星期的个位** 000：周日 001：周一 …… 110：周六 111：禁止
YRT	Bit 23-20	R	**年份的十位** 注：表示为 BCD 格式
YRU	Bit 19-16	R	**年份的个位** 注：表示为 BCD 格式
Reserved	Bit 15-13	—	**保留**
MONT	Bit 12	R	**月份的十位** 注：表示为 BCD 格式
MONU	Bit 11-8	R	**月份的个位** 注：表示为 BCD 格式
Reserved	Bit 7-6	—	**保留**
DAYT	Bit 5-4	R	**日期的十位** 注：表示为 BCD 格式
DAYU	Bit 3-0	R	**日期的个位** 注：表示为 BCD 格式

24. 7. 2. 15 RTC时间戳亚秒寄存器（RTC_TSSSEC）

RTC 时间戳亚秒寄存器（RTC_TSSSEC）

偏移地址：038_H

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	**保留**
SSEC	Bit 15-0	R	**亚秒值** 该位表示发生时间戳事件时同步预分频计数器的值

24. 7. 2. 16 RTC亚秒调整寄存器（RTC_SSECTR）

RTC 亚秒调整寄存器（RTC_SSECTR）

偏移地址：$03C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

INC	Bit 31	W	**秒修正位** 0：无修正 1：增加 1 秒 注：该位读出始终为 0
Reserved	Bit 30-15	—	**保留**
TRIM	Bit 14-0	W	**亚秒修正位** 该位表示减小相应的亚秒数，该位与 INC 配合使用可任意对时钟进行 1 秒之内的调整 注：该位读出始终为 0

注：该寄存器不支持字节操作。

24. 7. 2. 17 RTC中断使能寄存器（RTC_IER）

RTC 中断使能寄存器（RTC_IER）

偏移地址：040_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													WU	SSTC	RSC	Reserved		TAMP1	TAMP0	TSOV	TS	ALMB	ALMA	Reserved		YR	MON	DAY	HR	MIN	SEC

Reserved	Bit 31-19	—	**保留**
WU	Bit 18	R/W	**唤醒中断使能位** 0：禁止 1：使能
SSTC	Bit 17	R/W	**亚秒调整完成中断使能位** 0：禁止 1：使能
RSC	Bit 16	R/W	**寄存器同步完成中断使能位** 0：禁止 1：使能
Reserved	Bit 15-14	—	**保留**
TAMP1	Bit 13	R/W	**侵入检测 1 中断使能位** 0：禁止 1：使能
TAMP0	Bit 12	R/W	**侵入检测 0 中断使能位** 0：禁止 1：使能
TSOV	Bit 11	R/W	**时间戳溢出中断使能位** 0：禁止 1：使能
TS	Bit 10	R/W	**时间戳中断使能位** 0：禁止 1：使能
ALMB	Bit 9	R/W	**闹钟 B 中断使能位** 0：禁止 1：使能
ALMA	Bit 8	R/W	**闹钟 A 中断使能位** 0：禁止 1：使能
Reserved	Bit 7-6	—	**保留**
YR	Bit 5	R/W	**年份中断使能位** 0：禁止 1：使能

MON	Bit 4	R/W	**月份中断使能位** 0：禁止 1：使能
DAY	Bit 3	R/W	**日中断使能位** 0：禁止 1：使能
HR	Bit 2	R/W	**小时中断使能位** 0：禁止 1：使能
MIN	Bit 1	R/W	**分钟中断使能位** 0：禁止 1：使能
SEC	Bit 0	R/W	**秒中断使能位** 0：禁止 1：使能

24. 7. 2. 18 RTC中断标志寄存器（RTC_IFR）

RTC 中断标志寄存器（RTC_IFR）

偏移地址：044_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													WUF	SSTCF	RSCF	Reserved		TAMP1F	TAMP0F	TSOVF	TSF	ALMBF	ALMAF	Reserved		YRF	MONF	DAYF	HRF	MINF	SECF

Reserved	Bit 31-19	—	保留
WUF	Bit 18	R	**唤醒中断标志** 0：未发生事件 1：已发生事件
SSTCF	Bit 17	R	**亚秒调整完成中断标志** 0：未发生事件 1：已发生事件
RSCF	Bit 16	R	**寄存器同步完成中断标志** 0：未发生事件 1：已发生事件
Reserved	Bit 15-14	—	保留
TAMP1F	Bit 13	R	**侵入检测 1 中断标志** 0：未发生事件 1：已发生事件
TAMP0F	Bit 12	R	**侵入检测 0 中断标志** 0：未发生事件 1：已发生事件
TSOVF	Bit 11	R	**时间戳溢出中断标志** 0：未发生事件 1：已发生事件
TSF	Bit 10	R	**时间戳中断标志** 0：未发生事件 1：已发生事件
ALMBF	Bit 9	R	**闹钟 B 中断标志** 0：未发生事件 1：已发生事件
ALMAF	Bit 8	R	**闹钟 A 中断标志** 0：未发生事件 1：已发生事件
Reserved	Bit 7-6	—	保留
YRF	Bit 5	R	**年份中断标志** 0：未发生事件 1：已发生事件

MONF	Bit 4	R	**月份中断标志** 0：未发生事件 1：已发生事件
DAYF	Bit 3	R	**日中断标志** 0：未发生事件 1：已发生事件
HRF	Bit 2	R	**小时中断标志** 0：未发生事件 1：已发生事件
MINF	Bit 1	R	**分钟中断标志** 0：未发生事件 1：已发生事件
SECF	Bit 0	R	**秒中断标志** 0：未发生事件 1：已发生事件

注：通过操作 RTC_IFCR 寄存器清零以上标志

24. 7. 2. 19 RTC中断标志清零寄存器（RTC_IFCR）

RTC 中断标志清零寄存器（RTC_IFCR）

偏移地址：048_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													WUFC	SSTCFC	RSCFC	Reserved		TAMP1FC	TAMP0FC	TSOVFC	TSFC	ALMBFC	ALMAFC	Reserved		YRFC	MONFC	DAYFC	HRFC	MINFC	SECFC

Reserved	Bit 31-19	—	**保留**
WUFC	Bit 18	W1	**唤醒中断标志清零** 0：无操作 1：中断标志清零
SSTCFC	Bit 17	W1	**亚秒调整完成中断标志清零** 0：无操作 1：亚秒调整完成中断标志清零
RSCFC	Bit 16	W1	**寄存器同步完成中断标志清零** 0：无操作 1：寄存器同步完成中断标志清零
Reserved	Bit 15-14	—	**保留**
TAMP1FC	Bit 13	W1	**侵入检测 1 中断标志清零** 0：无操作 1：侵入检测 1 中断标志清零
TAMP0FC	Bit 12	W1	**侵入检测 0 中断标志清零** 0：无操作 1：侵入检测 0 中断标志清零
TSOVFC	Bit 11	W1	**时间戳溢出中断标志清零** 0：无操作 1：时间戳溢出中断标志清零
TSFC	Bit 10	W1	**时间戳中断标志清零** 0：无操作 1：时间戳中断标志清零
ALMBFC	Bit 9	W1	**闹钟 B 中断标志清零** 0：无操作 1：闹钟 B 中断标志清零
ALMAFC	Bit 8	W1	**闹钟 A 中断标志清零** 0：无操作 1：闹钟 A 中断标志清零
Reserved	Bit 7-6	—	**保留**
YRFC	Bit 5	W1	**年份中断标志清零** 0：无操作 1：年份中断标志清零

MONFC	Bit 4	W1	**月份中断标志清零** 0：无操作 1：月份中断标志清零
DAYFC	Bit 3	W1	**日中断标志清零** 0：无操作 1：日中断标志清零
HRFC	Bit 2	W1	**小时中断标志清零** 0：无操作 1：小时中断标志清零
MINFC	Bit 1	W1	**分钟中断标志清零** 0：无操作 1：分钟中断标志清零
SECFC	Bit 0	W1	**秒中断标志清零** 0：无操作 1：秒中断标志清零

24. 7. 2. 20 RTC中断状态寄存器（RTC_ISR）

RTC 中断状态寄存器（RTC_ISR）

偏移地址：$04C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													WUF	SSTCF	RSCF	Reserved		TAMP1F	TAMP0F	TSOVF	TSF	ALMBF	ALMAF	Reserved		YRF	MONF	DAYF	HRF	MINF	SECF

Reserved	Bit 31-19	—	保留
WUF	Bit 18	R	**唤醒中断状态标志** 0：未发生中断 1：已发生中断
SSTCF	Bit 17	R	**亚秒调整完成中断状态标志** 0：未发生中断 1：已发生中断
RSCF	Bit 16	R	**寄存器同步完成中断状态标志** 0：未发生中断 1：已发生中断
Reserved	Bit 15-14	—	保留
TAMP1F	Bit 13	R	**侵入检测 1 中断状态标志** 0：未发生中断 1：已发生中断
TAMP0F	Bit 12	R	**侵入检测 0 中断状态标志** 0：未发生中断 1：已发生中断
TSOVF	Bit 11	R	**时间戳溢出中断状态标志** 0：未发生中断 1：已发生中断
TSF	Bit 10	R	**时间戳中断状态标志** 0：未发生中断 1：已发生中断
ALMBF	Bit 9	R	**闹钟 B 中断状态标志** 0：未发生中断 1：已发生中断
ALMAF	Bit 8	R	**闹钟 A 中断状态标志** 0：未发生中断 1：已发生中断
Reserved	Bit 7-6	—	保留
YRF	Bit 5	R	**年份中断状态标志** 0：未发生中断 1：已发生中断

MONF	Bit 4	R	**月份中断状态标志** 0：未发生中断 1：已发生中断
DAYF	Bit 3	R	**日中断状态标志** 0：未发生中断 1：已发生中断
HRF	Bit 2	R	**小时中断状态标志** 0：未发生中断 1：已发生中断
MINF	Bit 1	R	**分钟中断状态标志** 0：未发生中断 1：已发生中断
SECF	Bit 0	R	**秒中断状态标志** 0：未发生中断 1：已发生中断

24. 7. 2. 21 RTC校准写保护寄存器（RTC_CALWPR）

RTC 校准写保护寄存器（RTC_CALWPR）

偏移地址：050_H

复位值：$00000000\_00000000\_00000000\_00000001_B$

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

Reserved	Bit 31-1	—	**保留**
WP	Bit 0	R	**写保护状态位** 0：写保护解除 1：写保护有效

注：对该寄存器写入 0x699655AA 解除写保护，写入其他值开启写保护。该寄存器保护 054_H~088_H 所有地址区域。

24. 7. 2. 22 RTC校准控制寄存器（RTC_CALCON）

RTC 校准控制寄存器（RTC_CALCON）

偏移地址：054_H

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-4	—	保留
CALP	Bit 3-1	R/W	**校准模式周期选择位** 000：每隔10秒校准一次 001：每隔20秒校准一次 010：每隔1分钟校准一次 011：每隔2分钟校准一次 100：每隔5分钟校准一次 101：每隔10分钟校准一次 110：每隔20分钟校准一次 111：每隔 1 秒校准一次 注：RTC 时钟输出选择精确 1Hz 时不可选择 CALP=111
CALEN	Bit 0	R/W	**校准使能位** 0：禁止 1：使能

24. 7. 2. 23 RTC校准值寄存器（RTC_CALDR）

RTC 校准值寄存器（RTC_CALDR）

偏移地址：058_H

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

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

DATA	Bit 31-16	R	**RTC 实时校准值** 注：读该寄存器时需连续读两次，两次值相同时该值有效
VAL	Bit 15-0	R/W	**RTC 数字校准值** 0x7FFF：正向补偿 32767 …… 0x0001：正向补偿 1 0x0000：无补偿 0xFFFF：负向补偿 1 …… 0x8001：负向补偿 32767 0x8000：预留 注 1：写入该寄存器之前需先使能 CALEN，否则写入后补偿无效 注 2：写入校准值后最多延时 1 秒后生效

24. 7. 2. 24 RTC备份寄存器（RTC_BKPxR）

RTC 备份寄存器（RTC_BKPxR）																															
偏移地址：100_H ~ $17C_H$																															
上电复位值：xxxxxxxx_xxxxxxxx _xxxxxxxx _xxxxxxxx$_B$ 系统复位：不受影响																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKP																															

BKP	Bit 31-0	R/W	**备份寄存器** 可以对该寄存器进行读取或写入操作。 该寄存器由备份域电源供电，当主电源域电源掉电时，寄存器的值仍可保持，并且系统复位不会影响寄存器的值。 发生侵入检测事件时该寄存器会被复位，并且只要RTC_IFR.TAMPxF=1，该寄存器就一直保持复位。

第25章 串行总线（I2C）

25. 1 概述

I2C 是两线双向的串行传输总线，提供了一种简单有效的方法来实现设备之间的数据交换。

I2C 是一种包括冲突检测与仲裁机制的多主机总线，如果两个或两个以上的主机同时试图控制总线时，仲裁机制可以防止数据损坏。芯片提供了标准模式（Sm）、快速模式（Fm）与极快速模式（Fm+）供用户选择。并且也提供 SMBus（系统管理总线）与 PMBus（电源管理总线）。

25. 2 特性

- 可配置为主机或从机
- 多主机模式（Multimaster capability）
- 标准模式（最高 100 kHz）
- 快速模式（最高 400 kHz）
- 极快速模式（最高 1 MHz）
- 7 位与 10 位地址模式
- 提供 2 组 7 位从机地址（2 个地址，其中一个包括屏蔽）
- 提供所有 7 位地址应答模式
- 提供广播模式（General call）
- 可编程的设置时间和保持时间
- 可选择时钟延长（Optional clock stretching）
- 提供深度为 16 字节的 TX / RX FIFOs
- 提供 DMA 传输
- 提供 SMBus 标准
- 硬件 PEC（封包错误检查）产生与 ACK 控制
- 命令与数据应答控制
- 提供地址解析协议（Address resolution protocol（ARP）support）
- 提供可选择为主控者或设备（Host and Device support）
- 提供 SMBus 警报
- 提供侦测超时与闲置功能
- 提供 PMBus V1.1 标准

25. 3 结构框图

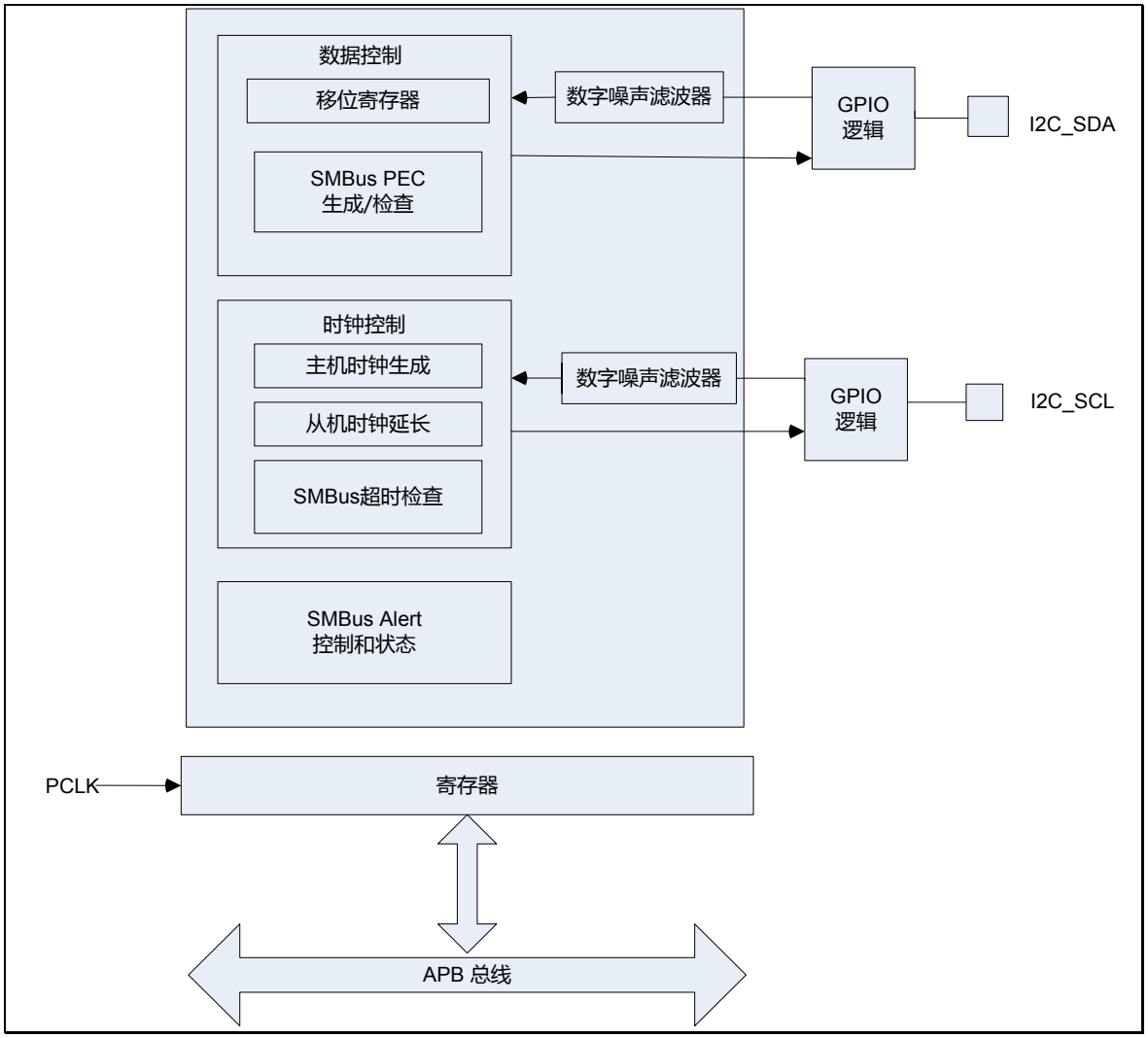


图 25-1 I2C 结构框图

25. 4 功能描述

除了接收和发送数据外，I2C 接口还完成接收和发送数据的串并转换。I2C 接口可以通过数据引脚（SDA）和时钟引脚（SCL）连接标准模式（最高 100 kHz），快速模式（最高 400 kHz）或极快速模式（最高 1 MHz）的 I2C 总线。I2C 的中断由软件开启和关闭。

I2C 接口也可通过数据引脚（SDA）和时钟引脚（SCL）连接到 SMBus，如果支持 SMBus 功能，还可以使用可选的 SMBus Alert 引脚（SMBA）。

25. 4. 1 I2C总线协议

I2C 是一种双线双向串行总线，可在设备之间提供简单有效的数据交换方法。I2C 标准是真正的多主机总线，包含冲突检测和仲裁机制，如果两个或多个主机同时尝试控制总线，则可防止数据损坏。数据在主机和从机之间逐字节同步传输到串行数据（SDA）和串行时钟（SCL）线上，每个数据字节长度为 8 位。

25. 4. 1. 1 START和STOP条件协议

I2C 规范将启动条件定义为在 SCL 时钟线为高电平时 SDA 数据线从高电平到低电平的转换。启动条件始终由主机生成，表示总线从空闲状态转换为活动状态。每个数据位对应一个 SCL 时钟脉冲，数据传输时先传输 MSB。每个传输的字节后面都有一个对应的应答位。当 SCL 为高电平时，SDA 线上的电平转换被解释为命令（START 或 STOP）。规定在 SCL 的高电平期间对每个数据位进行采样，因此 SDA 线上的数据只可以在 SCL 的低电平期间改变，并且必须在 SCL 的高电平期间保持稳定。

当总线空闲时，SCL 和 SDA 信号都通过总线上的外部上拉电阻拉高。当主机想要在总线上开始传输时，主机发出 START 信号，即 SCL 为 1 时，SDA 从高到低转换。当主机想要终止传输时，主机发出 STOP 条件，即 SCL 为 1 时，SDA 从低到高转换。下图为 START 和 STOP 条件的产生时序。当数据在总线上传输时，SCL 为 1 期间，SDA 线上的数据必须稳定。

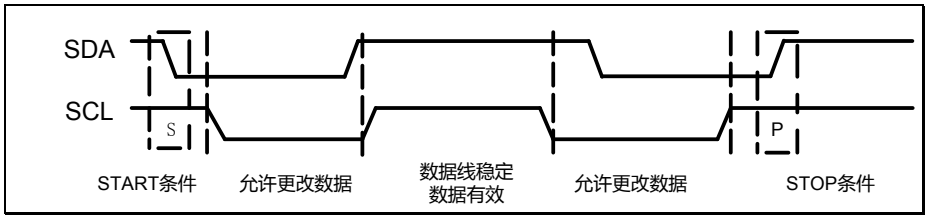


图 25-2 START 和 STOP 条件

25. 4. 1. 2 应答位

数据传输时要求带有应答位，应答位相关的时钟脉冲由主机产生。主机在应答时钟脉冲期间释放 SDA 线（HIGH）。从机必须在应答时钟脉冲低电平期间下拉 SDA 线，以便在时钟脉冲的高电平期间保持稳定的低电平。同时还必须考虑建立和保持时间。通常情况下，除非消息以 CBUS 地址开始，已经被寻址的从机必须在接收到每个字节后生成应答位。

当从机不应答从机地址时（例如从机正在执行某些实时功能而无法接收或发送），从机必须将数据线保持为高电平。然后，主机可以生成 STOP 条件以中止传输，或者重复 START 条件以启动新传输。

如果从机应答了从机地址，但是，传输后的某个时间不能再接收数据字节，则主机必须中止传输。这由从机在随后的第一个字节上产生非应答来指示。从机使数据线保持高电平，主机产生 STOP 或重复 START 条件。

如果主机进行接收传输，它必须通过在从机输出的最后一个字节上产生非应答来向从机发送数据结束信号。从机必须在应答时钟脉冲期间释放数据线，以允许主机产生 STOP 或重复 START 条件。

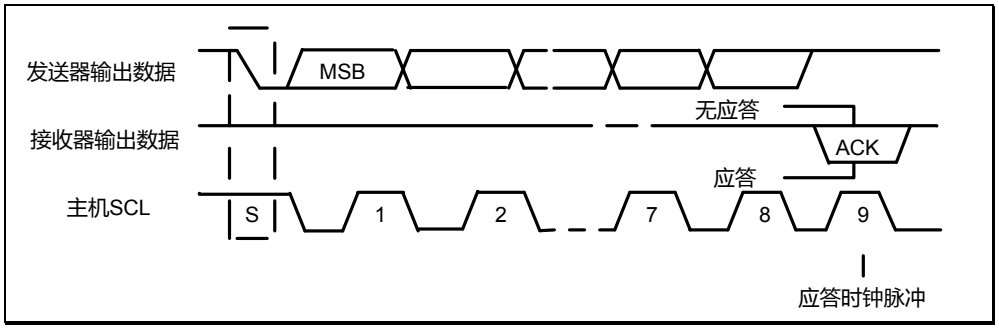


图 25-3 I2C 总线上的应答

25. 4. 1. 3 I2C寻址协议

从机有两种地址格式：7 位地址格式和 10 位地址格式。7 位地址格式下，第一个字节的前 7 位（bit7~bit1）设置从地址，LSB 位（bit0）是 R/W 位。当 bit0 置 0 时，主机向从机写数据。当 bit0 置 1 时，主机从从机读数据。数据首先传输最高有效位（MSB）。10 位地址格式下，主机传输两个字节以设置 10 位地址。第一个字节传输时，前 5 位（bit7~bit3）通知从机这是一个 10 位传输，接着是后 2 位（bit2~bit1），它为从机地址的第 9 位和第 8 位，LSB 位（bit0）是 R/W 位。传输的第二个字节设置从地址的 bit7~bit0。

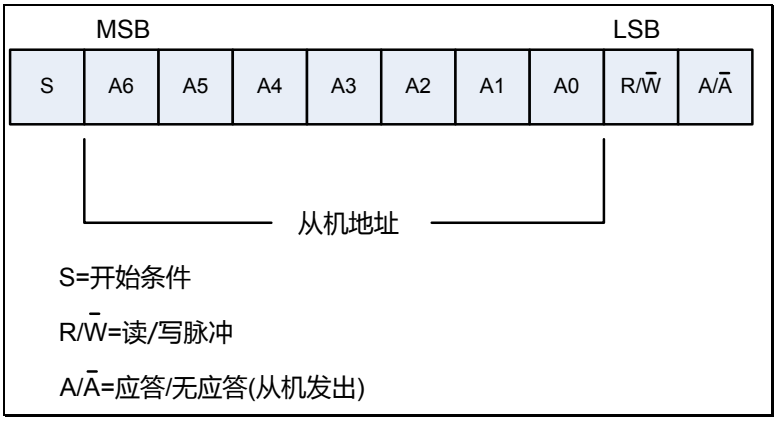


图 25-4 7 位地址格式

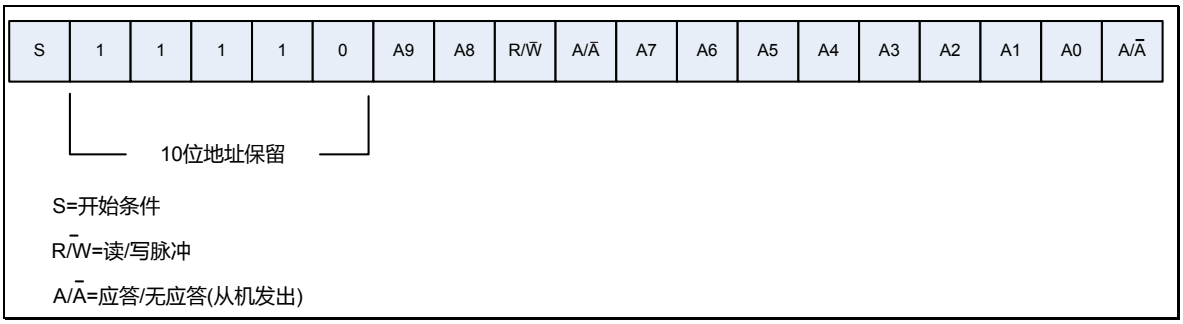


图 25-5 10 位地址格式

从地址	R/W位	描述
0000 000	0	广播地址
0000 000	1	START字节
0000 001	X	CBUS地址
1111 0XX	X	10位从机寻址

表 25-1 10 位地址格式第一个字节中位的定义

25. 4. 1. 4 I2C发送和接收协议

所有数据都以字节格式传输，每次传输的字节数没有限制。在主机发送地址和 R/W 位或主机向从机发送一个字节数据后，接收的从机必须响应应答脉冲。当接收的从机没有响应应答脉冲时，主机通过发出 STOP 条件来中止传输。从机应使 SDA 线保持高电平，以便主机可以中止传输。如果主机正在发送数据，则接收的从机在接收到每个数据字节后都应该响应应答脉冲。传输格式如下：

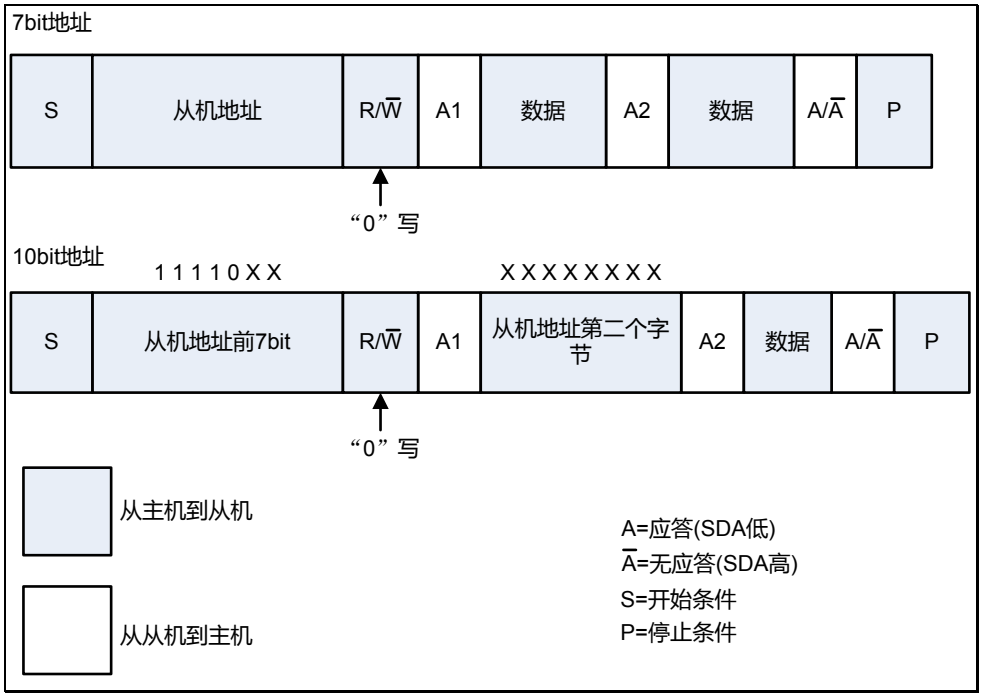


图 25-6 主机-发送协议

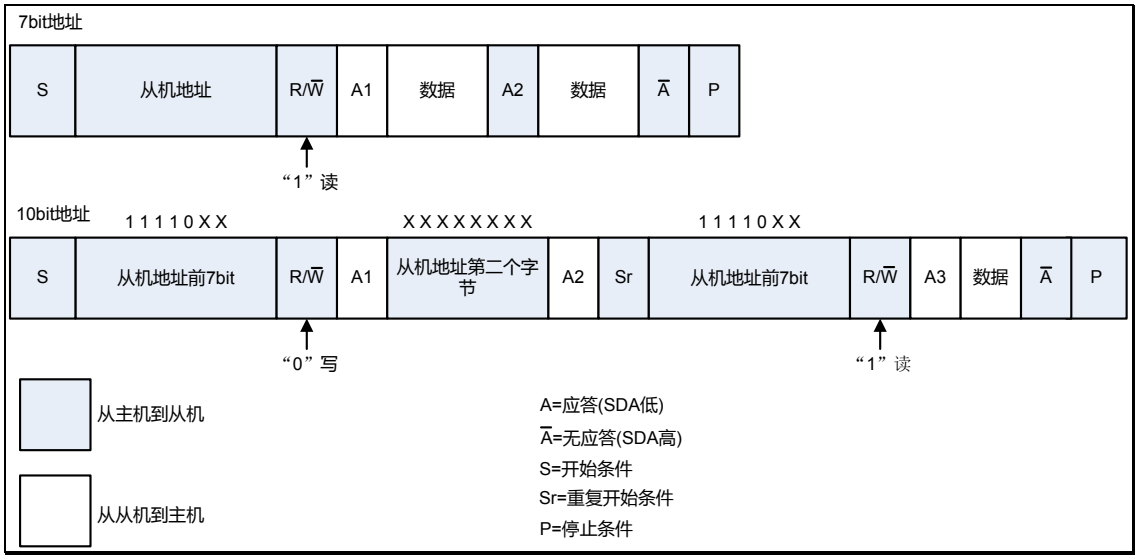


图 25-7 主机-接收协议

25. 4. 2 I2C时钟要求

I2C 内核由 I2CCLK 提供时钟。

I2CCLK 周期 T_{I2CCLK} 必须符合以下条件：

$T_{I2CCLK}<T_{LOW}/4$ 且 $T_{I2CCLK}<T_{HIGH}$

T_{LOW}：SCL 低电平时间。

T_{HIGH}：SCL 高电平时间。

PCLK 时钟周期 T_{PCLK} 必须符合以下条件：

$T_{PCLK}<1/8\ T_{SCL}$

T_{SCL}：SCL 周期。

25. 4. 3 数据传输

SDA 线上的每个字节必须为 8 位长。每次传输可以传输的字节数不受限制。每个字节后面都必须有一个应答位。首先使用最高有效位（MSB）传输数据。如果从机不能接收或发送另一个完整的数据字节，则它可以将时钟线 SCL 保持为低电平以强制主机进入等待状态，直到它执行完了某些其他功能，例如服务于内部中断。当从机准备好接收另一个数据字节并释放时钟线 SCL 时，数据传输继续。

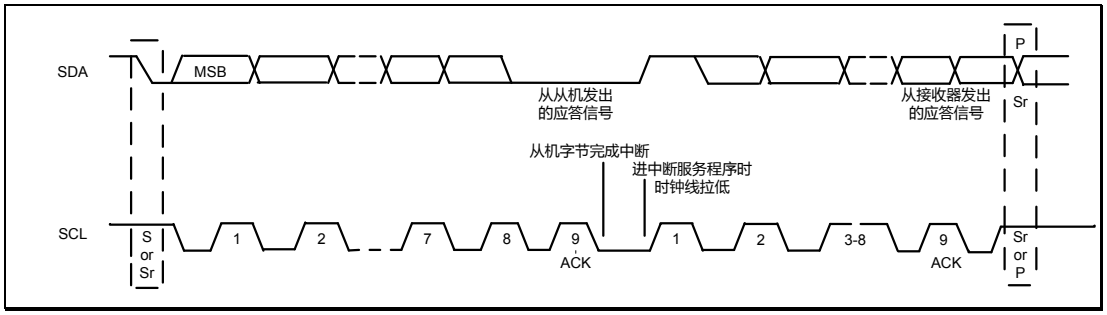


图 25-8 I2C 总线上的数据传输

接收

SDA 的输入填充移位寄存器。在第 8 个 SCL 脉冲之后（当接收到完整的数据字节时），如果接收 FIFO 尚未满（RXF = 0），则将其复制到接收 FIFO 中。如果 RXF = 1，意味着尚未读取先前接收的数据字节，此时 SCL 线会在第 9 个 SCL 脉冲之后延长其低电平，直到读取 I2Cx_RXDATA，使得接收 FIFO 未满（RXF = 0）。

传输

如果发送 FIFO 不为空（TXE = 0），则在第 9 个 SCL 脉冲（包括应答脉冲）之后将其内容复制到移位寄存器中。然后移位寄存器的内容会在 SDA 线上发送。如果 TXE = 1，意味着发送 FIFO 尚未写入数据，SCL 线将被拉低，直到有数据通过 I2Cx_TXDATA 写入到发送 FIFO。

硬件传输管理

I2C 具有嵌入在硬件中的字节计数器，以便管理字节传输并在各种模式下关闭通信，例如：在主模式下生成 NACK、STOP 和 RESTART，从接收器模式下的 ACK 控制，支持 SMBus

功能时的 PEC 生成和检查。

在主模式下，字节计数器始终被使能。在从模式下，字节计数器默认被禁止，但可以通过软件设置 I2Cx_CON1 寄存器中的 SBC（从机字节控制）位来使能。要传输的字节数在 I2Cx_CON2 寄存器的 NBYTES<7:0>位字段中设置。如果要传输的字节数（NBYTES）大于 255，或者接收时想要控制接收数据字节的应答值，则必须通过将 I2Cx_CON2 寄存器中的 RELOAD 位置 1 来选择重载模式。在此模式下，当达到 NBYTES 中所设置的字节数时，TCR 标志置位，如果 TCRIE 置位，则产生中断。只要设置了 TCR 标志，SCL 就会被延长。当软件对 NBYTES 写入非零值时，会清除 TCR。

当 NBYTES 计数器重新加载最后一个字节数时，必须清除 RELOAD 位。

当主模式下 RELOAD = 0 时，计数器可用于 2 种模式：

- ◇ 自动结束模式（I2Cx_CON2 寄存器中的 AUTOEND =1）。在此模式下，一旦传输了 NBYTES<7:0>位字段中编程的字节数，主机就会自动发送 STOP 条件。
- ◇ 软件结束模式（I2Cx_CON2 寄存器中的 AUTOEND ='0'）。在此模式下，一旦传输了 NBYTES<7:0>位字段中编程的字节数，就会产生软件操作; 如果 TCIE 位置 1，则 TC 标志置 1，产生中断。只要 TC 标志置位，SCL 信号就会被拉长。当 I2Cx_CON2 寄存器中的 START 或 STOP 位置 1 时，TC 标志由软件清零。当主机要发送 RESTART 条件时，必须使用此模式。

25. 4. 4 I2C主机模式

I2C 主机初始化

在启用外设之前，必须通过将 I2Cx_TIMINGR 寄存器中的 SCLH 和 SCLL 位置 1 来配置 I2C 主时钟,实现时钟同步机制以支持多主机环境和从机时钟延长。

为了允许时钟同步：

- ◇ 从 SCL 低电平内部检测开始，使用 SCLL 计数器计数低电平时钟。
- ◇ 从 SCL 高电平内部检测开始，使用 SCLH 计数器计数高电平时钟。

根据 SCL 下降沿，主机在 T_{SYNC} 延迟后检测到自己的 SCL 低电平。一旦 SCLL 计数器达到 I2Cx_TIMINGR 寄存器中 SCLL<7:0>位中设置的值，主机就会将 SCL 释放为高电平，SCL 输入数字噪声滤波器且与 I2Cx 时钟同步。

在 T_{SYNC} 产生延迟后，主机会检测自己的 SCL 高电平，具体取决于 SCL 上升沿，SCL 输入数字噪声滤波器且与 I2Cx 时钟同步。

一旦 SCLH 计数器达到 I2Cx_TIMINGR 寄存器中的 SCLH<7:0>位编程的值，主机就会将 SCL 置为低电平。

因此，主时钟周期为：

$T_{SCL} = T_{SYNC1} + T_{SYNC2} + \{[（SCLH+1）+（SCLL+1）]*（PRESC+1）*T_{I2CCLK}\}$

T_{SYNC1} 的持续时间取决于以下参数：

- ◇ SCL 下降斜率
- ◇ 由于 SCL 与 I2CCLK 时钟同步（2 至 3 个 I2CCLK 周期）导致的延迟

T_{SYNC2} 的持续时间取决于以下参数：

- ◇ SCL 上升斜率
- ◇ 由于 SCL 与 I2CCLK 时钟同步（2 至 3 个 I2CCLK 周期）导致的延迟

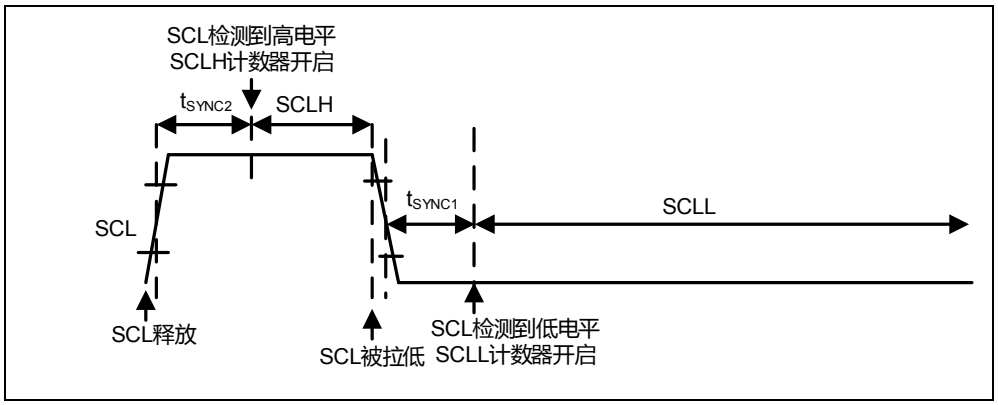


图 25-9 主时钟产生

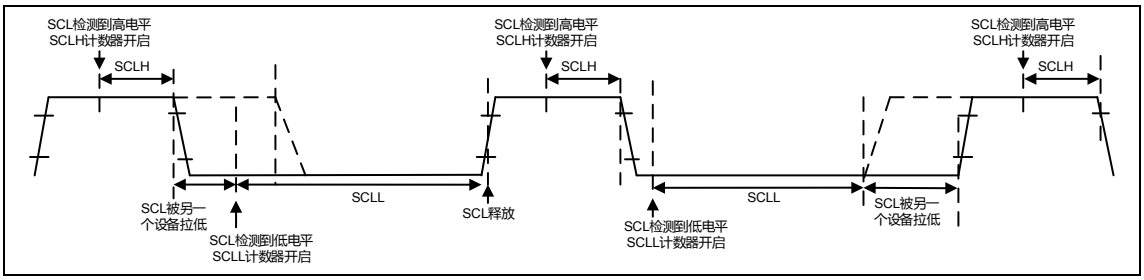


图 25-10　SCL 主时钟同步

主机通信初始化（地址阶段）

为了启动通信，用户必须为 I2Cx_CON2 寄存器中寻址的从机编程以下参数：

- ◇ 寻址模式（7 位或 10 位）：ADD10
- ◇ 要发送的从站地址：SADD<9:0>
- ◇ 传输方向：RD_WRN
- ◇ 如果是 10 位地址读取：HEAD10R 位。必须配置 HEAD10R 以指示是否必须发送完整的地址序列，或者仅在方向改变时发送地址标头。
- ◇ 要传输的字节数：NBYTES<7:0>。如果字节数等于或大于 255 个字节，则 NBYTES<7:0>最初必须填充 0xFF。

然后，用户必须将 I2Cx_CON2 寄存器中的 START 位置 1。当 START 位置 1 时，不允许更改所有上述位。一旦检测到总线空闲（BUSY = 0）后，主机就会自动发送 START 条件，再发送从机地址。

在仲裁丢失的情况下，主机自动切换回从机模式，如果它作为寻址从机，则可以应答自己的地址。

注 1：无论接收到的应答值是什么，当总线上的从机地址发送时，START 位由硬件复位。如果发生仲裁丢失，则 START 位也由硬件复位。在 10 位寻址模式下，当从机地址前 7 位被从机 NACK 时，主机将自动重新启动从机地址传输，直到收到 ACK。如果在 START 位置 1 时 I2C 被寻址为从机（ADDRRI = 1），则 I2C 切换到从机模式，当 ADDRRI 位置 1 时，START 位清零。

注 2：对重复启动条件应用相同的过程。在这种情况下，BUSY = 1。

初始化主机 10 位从地址寻址接收模式

- ◇ 如果从机地址采用 10 位格式，用户可以通过清零 I2Cx_CON2 寄存器中的 HEAD10R 位来选择发送完整的读序列。在这种情况下，主机在 START 位置 1 后自动发送以下完整序列：启动+从机地址 10 位标头写+从机地址第 2 字节+重新启动+从机地址 10 位标头读取；

如果主机寻址 10 位地址从机，将数据发送到该从机，然后从同一从机读取数据，则必须先完成主机传输流程，然后使用 HEAD10R = 1 配置的 10 位从地址设置重复启动。在这种情况下，主机发送此序列：重新启动+从机地址 10 位标头读取。

主机传送

在主机开始传送之前，需先配置 I2C_CON2 寄存器中 NBYTES 位，当传输字节大于 255 时，需要额外配置 RELOAD 位。在此配置下，当 NBYTES 配置的字节数传送完成后，I2C_RIF 寄存器中 TCRRI 位会置位，此时 SCL 时钟会保持低电平，直到 NBYTES 位重新写入新的数值以及对 I2C_TXDATA 寄存器写入发送数据后，才会继续进行传输。

当从机回应 NACK，I2C_RIF 寄存器中 NACKRI 位会置位，并且接下来主机会自动发送停止信号 STOP。

当从机响应 ACK，且 RELOAD = 0、NBYTES 所配置的笔数都已经传完时，会有以下情况：

◇ 若 AUTOEND = 1，此时会自动发送停止讯号 STOP。

◇ 若 AUTOEND = 0，此时主机会将 SCL 时钟保持低电平，等待软件控制后续操作：

RESTART：对 I2C_CON2 寄存器中 START 置位，此时会发送重启信号。

STOP：对 I2C_CON2 寄存器中 STOP 置位，此时会发送停止信号。

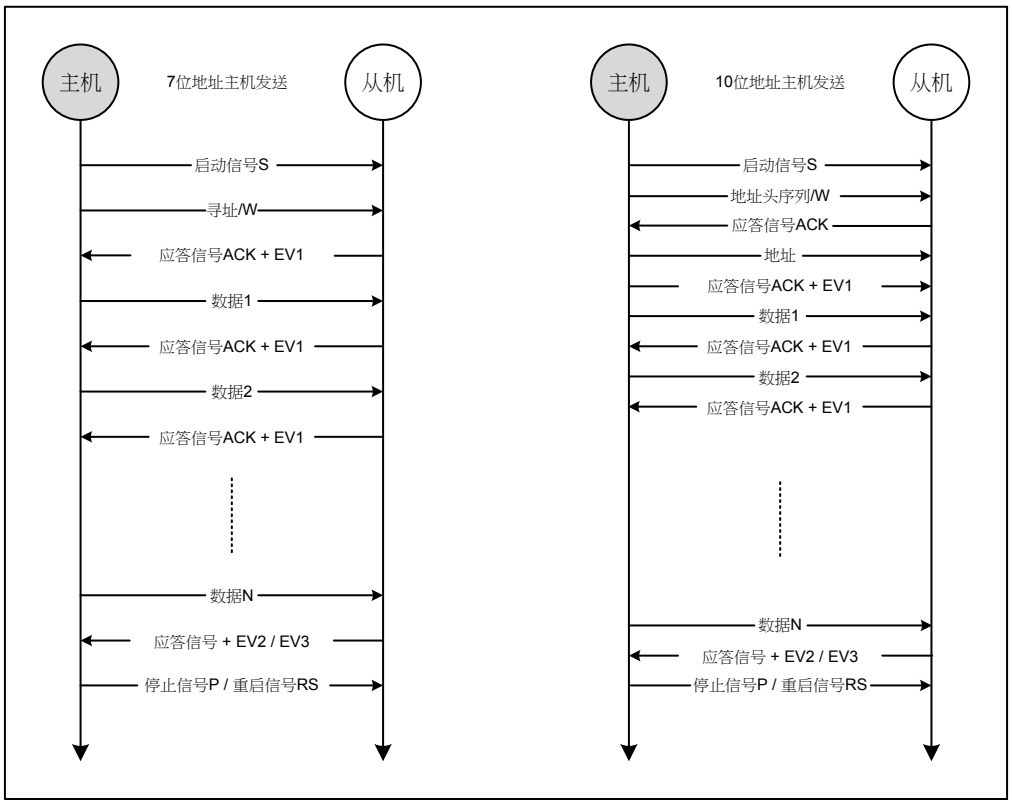


图 25-11　主机发送的传输序列图

注 1：S=起始位，RS=重复起始位，P=停止位，ACK=应答，NACK=非应答

注 2：EV1 = 传输尚未完成事件，判断 **I2C_STAT**.TXTH，若低于水平则对 **I2C_TXDATA** 写入数据。

注 3：EV2 = 当应答信号为 ACK 时，若传输已完成，后续操作为停止信号 STOP 或重启信号 RESTART。

注 4：EV3 = 当应答信号为 NACK 时，后续操作为停止信号 STOP。

注 5：如果软件序列在当前传送字节传输结束之前尚未写入下一个字节，导致 TX FIFO 为空时，EV1 事件将会延长 SCL 时钟低电平时间。

主机接收

在主机开始接收之前，需先配置 I2C_CON2 寄存器中 NBYTES 位，当传输字节大于 255 时，需要额外配置 RELOAD 位。在此配置下，当 NBYTES 配置的字节数传送完成后，I2C_RIF 寄存器中 TCRRI 位会置位，此时 SCL 时钟会保持低电平，直到 NBYTES 位重新写入新的数值后，才会继续进行传输。

当 RELOAD = 0 并且 NBYTES 所配置的笔数都已经传完时，会有以下情况：

◇ 若 AUTOEND = 1，此时会自动发送非应答信号 NACK 与停止讯号 STOP。

◇ 若 AUTOEND = 0，此时会自动发送非应答信号 NACK，并将 SCL 时钟保持低电平，等待软件控制后续操作：

RESTART：对 I2C_CON2 寄存器中 START 置位，此时会发送重启信号。

STOP：对 I2C_CON2 寄存器中 STOP 置位，此时会发送停止信号。

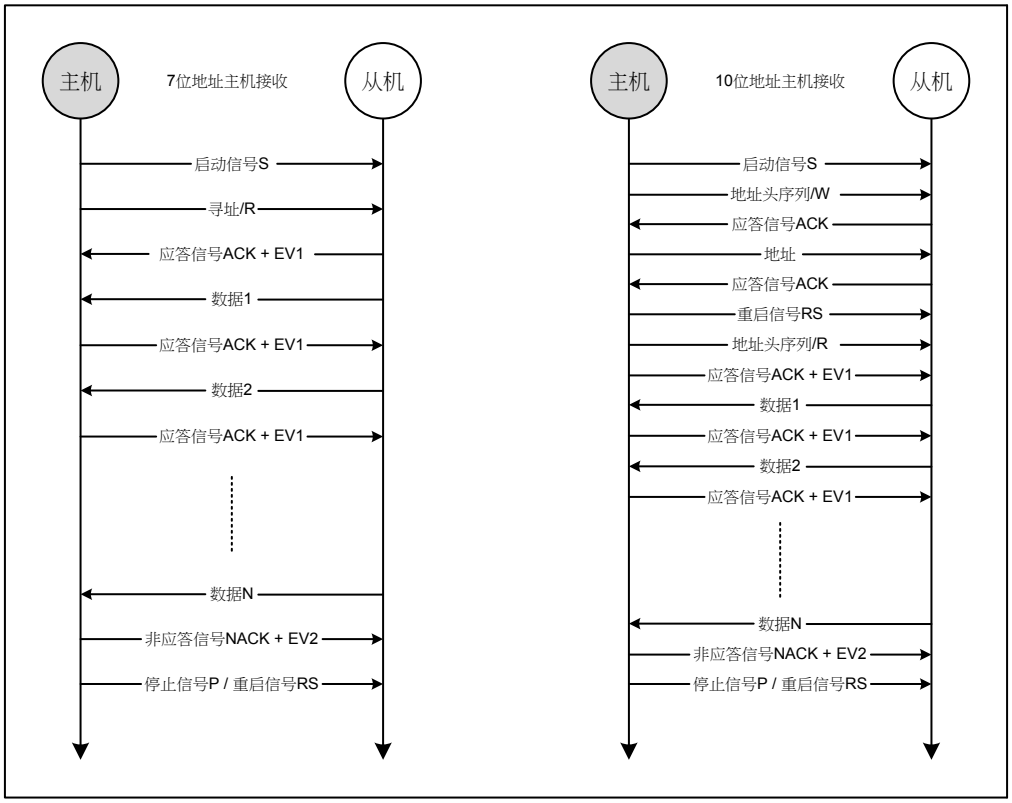


图 25-12 主机接收的传输序列图

注 1：S=起始位，RS=重复起始位，P=停止位，ACK=应答，NACK=非应答

注 2：EV1 = 传输尚未完成事件，判断 **I2C_STAT**.RXTH，若高于水平则对 **I2C_RXDATA** 读出数据。

注 3：EV2 = 传输完成事件，此时会自动发送非应答信号 NACK，后续根据软件配置来决定是发送停止信号 STOP 还是重启信号 RESTART。

注 4：如果软件序列在当前传送字节传输结束之前尚未将数据读出，导致 RX FIFO 满时，EV1 事件将会延长 SCL 时钟低电平时间。

25. 4. 5 I2C从机模式

I2C 从机初始化

为了在从机模式下工作，用户必须至少启用一个从机地址。两个寄存器 I2Cx_ADDR1 和 I2Cx_ADDR2 可用于设置从机自身地址 OA1 和 OA2。

- ◇ 通过将 I2Cx_ADDR1 寄存器中的 OA1MODE 位置 1，可以在 7 位模式（默认情况下）或 10 位寻址模式下配置 OA1。通过将 I2Cx_ADDR1 寄存器中的 OA1EN 位置 1 来启用 OA1。
- ◇ 如果需要额外的从地址，可以配置第二个从地址 OA2。通过配置 I2Cx_ADDR2 寄存器中的 OA2MSK<2:0>位，最多可以屏蔽 OA2<6:0>。因此，对于配置为 1 至 6 的 OA2MSK，仅 OA2<6:1>，OA2<6:2>，OA2<6:3>，OA2<6:4>，OA2<6:5>或 OA2<6>与收到的地址相比较。一旦 OA2MSK 不等于 0，OA2 的地址比较器就会排除未被应答的 I2C 保留地址（0000XXX 和 1111XXX）。如果 OA2MSK = 7，则应答所有接收的 7 位地址（保留地址除外）。OA2 始终是 7 位地址。

  当 OA2MSK = 0 的情况下且在 I2Cx_ADDR1 或 I2Cx_ADDR2 寄存器中设置了特定启用位置，则可应答这些保留地址。通过将 I2Cx_ADDR2 寄存器中的 OA2EN 位置 1 启用 OA2。
- ◇ 通过将 I2Cx_CON1 寄存器中的 GCEN 位置 1 使能广播地址。

当启用地址选择时，地址匹配中断标志状态置 1，如果 ADDRIE 位置 1，则产生中断。

默认情况下，从机使用其时钟延长功能，这意味着它在需要时将 SCL 信号拉到低电平，以执行软件操作。如果主机不支持时钟延长，则必须在 I2Cx_CON1 寄存器中将 I2C 配置为 NOSTRETCH = 1。

收到地址匹配中断后，如果启用了多个地址，用户必须读取 I2Cx_STAT 寄存器中的 ADDCODE<6:0>位，以检查匹配的地址。还必须检查 DIR 标志以了解传输方向。

从机时钟延长（NOSTRETCH = 0）

在默认模式下，I2C 从机在以下情况下延长 SCL 时钟：

- ◇ 在传输过程中，如果先前的数据传输完成且没有在 I2Cx_TXDATA 寄存器中写入新数据。当数据写入 I2Cx_TXDATA 寄存器时，将释放此延长。
- ◇ 在接收时，I2Cx_RXDATA 寄存器尚未读取且接收 FIFO 已满。读取 I2Cx_RXDATA 且接收 FIFO 未满时，将释放此延长。
- ◇ 从机字节控制模式下，当发生 TCR = 1 时，重载模式（SBC = 1 且 RELOAD = 1），表示数据字节已被传输。此时通过在 NBYTES<7:0>字段中写入非零值清除 TCR 时，释放该延长。
- ◇ 从机应答控制模式下，当收到地址或数据时，每个字节的第 8 和第 9 个 SCL 脉冲之间会将 SCL 延长。当设置应答更新时，将会释放该延长并回应应答脉冲。
- ◇ 在 SCL 下降沿检测后，I2C 拉低 SCL 延长单位为

  [（SDADEL+SCLDEL+1）*（PRESC+1） + 1] * T_{I2CCLK}

从机没有时钟延长（NOSTRETCH = 1）

当 I2Cx_CON1 寄存器中的 NOSTRETCH = 1 时，I2C 从机不会延长 SCL 信号。

◇ 发送时，必须在与传输相对应的第一个 SCL 脉冲之前将数据写入 I2Cx_TXDATA 寄存器。如果不是，则发生欠载运算，在 I2Cx_STAT 寄存器中设置 TXUD 标志，如果 I2Cx_IER 寄存器中的 TXUDIE 位置 1，则会产生中断。

◇ 接收时，如果接收 FIFO 满时，必须在下一个数据字节的第 9 个 SCL 脉冲（应答脉冲）之前从 I2Cx_RXDATA 寄存器中读取数据。如果发生溢出，则在 I2Cx_STAT 寄存器中设置 RXOV 标志，如果 I2Cx_IER 寄存器中的 RXOVIE 位置 1，则会产生中断。

从机字节控制模式

为了在从机接收模式下允许字节 ACK 控制，必须通过将 I2Cx_CON1 寄存器中的 SBC 位置 1 来启用从机字节控制模式。这需要符合 SMBus 标准。

必须选择重载模式，以便在从机接收模式（RELOAD = 1）中允许字节控制。要控制每个字节，必须在 ADDR 中断子程序中将 NBYTES 初始化为 0x1，并在每个接收到的字节后重新加载到 0x1。当接收到该字节时，TCR 位置 1，在第 9 个 SCL 脉冲之后将 SCL 信号拉低。用户可以从 I2Cx_RXDATA 寄存器读取数据，然后通过配置 I2Cx_CON2 寄存器中的 NACK 位决定下一个数据是否应答。通过将 NBYTES 设置为非零值来释放 SCL 延伸：发送应答或不应答。

NBYTES 可以加载大于 0x1 的值，在这种情况下，接收流程在 NBYTES 数据接收期间是连续的。

注释：当禁止 I2C 时，才可配置 SBC 位。当 TCR = 1 时，此时可以更改 RELOAD 位值。

警告：从机字节控制模式与 NOSTRETCH 模式不兼容。不允许在 NOSTRETCH = 1 时设置 SBC。

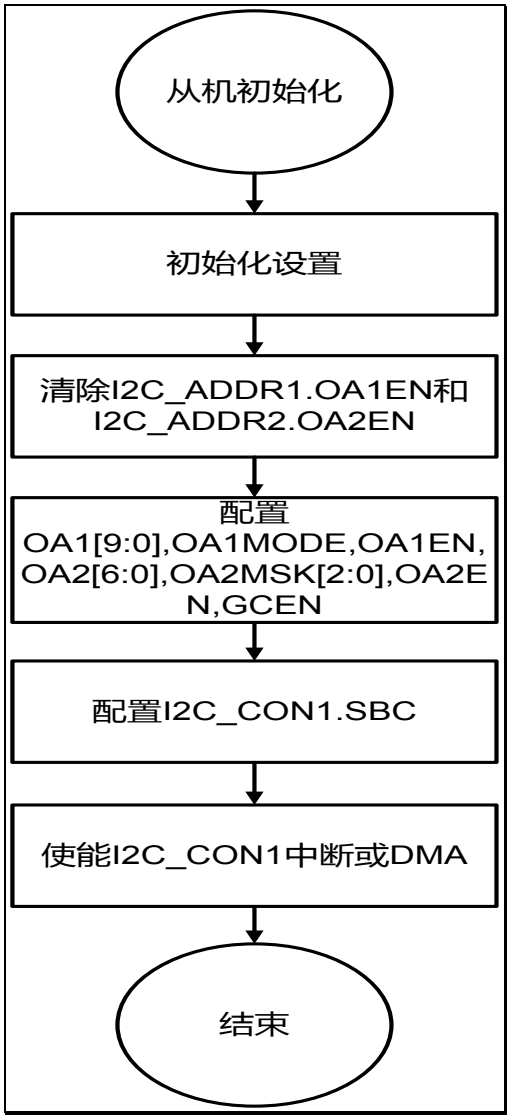


图 25-13 从机初始化流程图

从机传送

在接收到匹配的地址时，I2C_RIF 寄存器中 ADDRRI 位会置位，此时若 TX FIFO 中已准备好要发送的数据时，从机会通过内部移位寄存器将 TX FIFO 中的字节发送到 SDA 线。若 TX FIFO 此时为空，从机会延长 SCL 低电平时间，直到 TX FIFO 透过 I2C_TXDATA 寄存器写入发送数据为止。

当发送成功，主机会回应答信号，当主机回应 NACK 时，此时 I2C_RIF 寄存器中 NACKRI 位会置位，此时从机会自动释放 SCL 与 SDA 总线让主机能够发送后续的 STOP 或 RESTART 命令。

当主机回应 STOP 时，此时 I2C_RIF 寄存器中 STOPRI 位会置位，并结束通信，等待下一次收到匹配的地址。然而，若 TX FIFO 内还有尚未传送的字节，使用者可以选择下一次地址匹配时，继续传送 FIFO 中的数据，或是对 I2C_FCON 寄存器中 TXFRST 位置位，来清除 FIFO 中的字节，下一次传送新的字节。

当从机字节控制启动时，需要传送多少字节需在 I2C_CON2 寄存器中 NBYTES 内配置。

警告：当 NOSTRETCH 模式启动时，由于 SCL 时钟无法被延长，因此需要传送的字节需要被提前写入 TX FIFO 内。

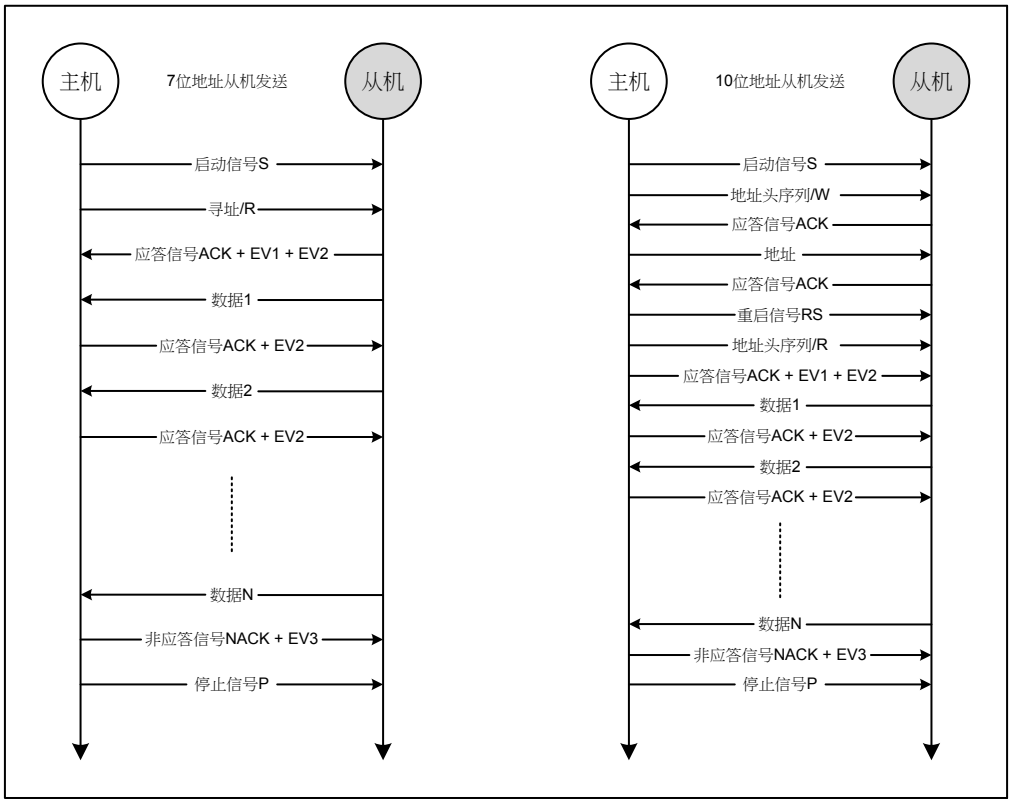


图 25-14　从机发送的传输序列图

注 1：S=起始位，RS=重复起始位，P=停止位，ACK=应答，NACK=非应答

注 2：EV1 = 当收到匹配地址时，I2C_RIF. ADDRRI = 1，对 I2C_ICR.ADDRIC 置位来清除中断。

注 3：EV2 = 判断 I2C_STAT.TXTH，若低于水平则对 I2C_TXDATA 写入数据。

注 4：EV3 = 当收到 NACK 时，I2C_RIF. NACKRI = 1，对 I2C_ICR.NACKIC 置位来清除中断。此时还需判断 TX FIFO 内是否还有尚未发送的字节，若有则软件需要额外处理。

注 5：如果软件序列在当前传送字节传输结束之前尚未写入下一个字节，导致 TX FIFO 为空时，EV2 事件将会延长 SCL 时钟低电平时间。

从机接收

在接收到匹配的地址时，I2C_RIF 寄存器中 ADDRRI 位会置位，从机此时会通过内部移位寄存器将 SDA 总线上的字节保存到 RX FIFO 中，无论 RX FIFO 是否满，从机都会自动发送应答信号，差异仅在应答信号发送后，FIFO 满时会延长 SCL 时钟低电平时间，等待软件将 RX FIFO 中的字节取出后，才会释放 SCL 时钟。

当主机发送 STOP 时，此时 I2C_RIF 寄存器中 STOPRI 位会置位，并结束通信，等待下一次收到匹配的地址。

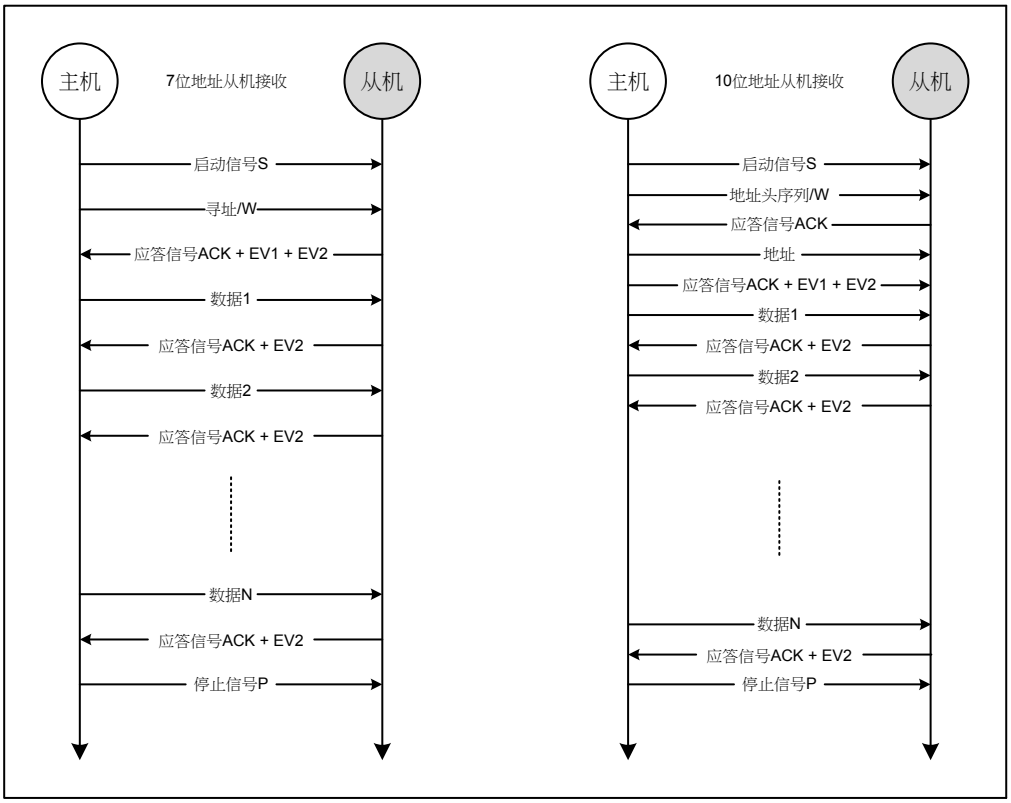


图 25-15　从机接收的传输序列图

注 1：S=起始位，RS=重复起始位，P=停止位，ACK=应答，NACK=非应答

注 2：EV1 = 当收到匹配地址时，I2C_RIF. ADDRRI = 1，对 I2C_ICR.ADDRIC 置位来清除中断。

注 3：EV2 = 判断 I2C_STAT.RXTH，若高于水平则对 I2C_RXDATA 取出数据。

注 4：如果软件序列在当前传送字节传输结束时尚未将数据读出，导致 RX FIFO 满，EV2 事件将会延长 SCL 时钟低电平时间。

25. 4. 6 I2Cx_TIMINGR寄存器的配置的例子

下表提供了如何对 I2Cx_TIMINGR 进行编程以获得符合 I2C 规范的时序的示例。

参数	标准模式（Sm）		快速模式（Fm）	极快速模式（Fm+）
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC	1	1	0	0
SCLL	0xC7	0x13	0x9	0x6
T_{SCLL}	200 * 250 ns = 50 µs	20 * 250 ns = 5.0 µs	10 * 125 ns = 1250 ns	7 * 125 ns = 875 ns
SCLH	0xC3	0xF	0x3	0x3
T_{SCLH}	196 * 250 ns = 49 µs	16 * 250 ns = 4.0µs	4 * 125ns = 500ns	4 * 125ns = 500ns
T_{SCL} (1)	~100 µs (2)	~10 µs (2)	~2500 ns (3)	~2000 ns (4)
SDADEL	0x2	0x2	0x1	0x0
T_{SDADEL}	2 * 250 ns = 500 ns	2 * 250 ns = 500 ns	1 * 125 ns = 125 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
T_{SCLDEL}	5 * 250 ns = 1250 ns	5 * 250 ns = 1250 ns	4 * 125 ns = 500 ns	2 * 125 ns = 250 ns

表 25-2 F_{I2CCLK} = 8 MHz 的时序设置示例

注 1：由于 SCL 内部检测延迟，SCL 周期 T_{SCL} 大于 T_{SCLL} + T_{SCLH}。为 T_{SCL} 提供的值仅为示例。
注 2：T_{SYNC1} + T_{SYNC2} 最小值为 4 * T_{I2CCLK} = 500 ns. T_{SYNC1} + T_{SYNC2} = 1000 ns 的示例。
注 3：T_{SYNC1} + T_{SYNC2} 最小值为 4 * T_{I2CCLK} = 500 ns. T_{SYNC1} + T_{SYNC2} = 750 ns 的示例。
注 4：T_{SYNC1} + T_{SYNC2} 最小值为 4 * T_{I2CCLK} = 500 ns. T_{SYNC1} + T_{SYNC2} = 655 ns 的示例。

参数	标准模式（Sm）		快速模式（Fm）	极快速模式（Fm+）
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	3	3	1	0
SCLL	0xC7	0x13	0x9	0x4
T_{SCLL}	200 * 250 ns = 50 µs	20 * 250 ns = 5.0 µs	10 * 125 ns = 1250 ns	6 * 62.5 ns = 312.5 ns
SCLH	0xC3	0xF	0x3	0x2
T_{SCLH}	196 * 250 ns = 49 µs	16 * 250 ns = 4.0µs	4 * 125ns = 500ns	3 * 62.5ns = 187.5ns
T_{SCL} (1)	~100 µs (2)	~10 µs (2)	~2500 ns (3)	~1000 ns (4)
SDADEL	0x2	0x2	0x2	0x0
T_{SDADEL}	2 * 250 ns = 500 ns	2 * 250 ns = 500 ns	2 * 125 ns = 250 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x2
T_{SCLDEL}	5 * 250 ns = 1250 ns	5 * 250 ns = 1250 ns	4 * 125 ns = 500 ns	3 * 62.5 ns = 187.5 ns

表 25-3 F_{I2CCLK} = 16 MHz 的时序设置示例

注 1：由于 SCL 内部检测延迟，SCL 周期 T_{SCL} 大于 T_{SCLL} + T_{SCLH}。为 T_{SCL} 提供的值仅为示例。
注 2：T_{SYNC1} + T_{SYNC2} 最小值为 4 * T_{I2CCLK} = 250 ns. T_{SYNC1} + T_{SYNC2} = 1000 ns 的示例。
注 3：T_{SYNC1} + T_{SYNC2} 最小值为 4 * T_{I2CCLK} = 250 ns. T_{SYNC1} + T_{SYNC2} = 750 ns 的示例。
注 4：T_{SYNC1} + T_{SYNC2} 最小值为 4 * T_{I2CCLK} = 250 ns. T_{SYNC1} + T_{SYNC2} = 500 ns 的示例。

参数	标准模式（Sm）		快速模式（Fm）	极快速模式（Fm+）
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	0xB	0xB	5	5
SCLL	0xC7	0x13	0x9	0x3
T_{SCLL}	200 * 250 ns = 50 µs	20 * 250 ns = 5.0 µs	10 * 125 ns = 1250 ns	4 * 125 ns = 500 ns
SCLH	0xC3	0xF	0x3	0x1
T_{SCLH}	196 * 250 ns = 49 µs	16 * 250 ns = 4.0µs	4 * 125ns = 500ns	2 * 125ns = 250ns
T_{SCL} (1)	~100 µs (2)	~10 µs (2)	~2500 ns (3)	~875 ns (4)
SDADEL	0x2	0x2	0x3	0x0
T_{SDADEL}	2 * 250 ns = 500 ns	2 * 250 ns = 500 ns	3 * 125 ns = 375 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
T_{SCLDEL}	5 * 250 ns = 1250 ns	5 * 250 ns = 1250 ns	4 * 125 ns = 500 ns	2 * 125ns = 250ns

表 25-4 F_{I2CCLK} = 48 MHz 的时序设置示例

注 1：由于 SCL 内部检测延迟，SCL 周期 T_{SCL} 大于 T_{SCLL} + T_{SCLH}。为 T_{SCL} 提供的值仅为示例。

注 2：T_{SYNC1} + T_{SYNC2} 最小值为 4 * T_{I2CCLK} = 83.3 ns. T_{SYNC1} + T_{SYNC2} = 1000 ns 的示例。

注 3：T_{SYNC1} + T_{SYNC2} 最小值为 4 * T_{I2CCLK} = 83.3 ns. T_{SYNC1} + T_{SYNC2} = 750 ns 的示例。

注 4：T_{SYNC1} + T_{SYNC2} 最小值为 4 * T_{I2CCLK} = 83.3 ns. T_{SYNC1} + T_{SYNC2} = 250 ns 的示例。

25. 4. 7 SMBus具体功能

系统管理总线（SMBus）是一个双线接口，各种设备可以通过该接口相互通信并与系统的其余部分通信。它基于 I2C 操作原理。SMBus 为系统和电源管理相关任务提供控制总线。

该外设与 SMBUS 规范 rev 2.0（http://smbus.org）兼容。

系统管理总线规范指的是三种类型的设备。

- ◇ 从机是接收或响应命令的设备。
- ◇ 主设备是发出命令，生成时钟并终止传输的设备。
- ◇ 主控者是专用主机，为系统的 CPU 提供主接口。主控者必须可当作是主机或从机，并且必须支持 SMBus 主机通信协议。

系统中只允许一个主控者。该外设可以配置为主设备或从设备，也可以配置为主控者。

SMBUS 基于 I2C 规范 2.1 版。

总线协议

对于任何给定的设备，有 11 种可能的命令协议。设备可以使用 11 个协议中的任何一个或全部来进行通信。协议包括快速命令，发送字节，接收字节，写入字节，写入字，读取字节，读取字，进程调用，块读取，块写入和块写入块读取进程调用。这些协议应由用户软件实现。有关这些协议的更多详细信息，请参阅 SMBus 规范 2.0 版（http://smbus.org）。

地址解析协议（ARP）

可以通过为每个从设备动态分配新的唯一地址来解决 SMBus 从地址冲突。为了提供隔离每个设备以便进行地址分配的机制，每个设备必须实现唯一的设备标识符（UDID）。这个 128 位数字由软件实现。该外设支持地址解析协议（ARP）。通过将 I2Cx_CON1 寄存器中的 SMBDEN 位置 1 启用 SMBus 从设备默认地址（0b1100001）。ARP 命令应由用户软件实现。仲裁也在从属模式下执行以支持 ARP。

有关 SMBus 地址解析协议的更多详细信息，请参阅 SMBus 规范 2.0 版（http://smbus.org）。

命令接收和数据应答控制

SMBus 接收器必须能够 NACK 每个接收到的命令或数据。为了在从机模式下允许 ACK 控制，必须通过将 I2Cx_CON1 寄存器中的 SBC 位置 1 来启用从机字节控制模式。

主控者通知协议

该外设通过将 I2Cx_CON1 寄存器中的 SMBHEN 位置 1 来支持主控者通知协议。在这种情况下，主控者将应答 SMBus 主控者地址（0b0001 000）。使用此协议时，设备充当主机，主控者充当从机。

SMBus 警报

支持 SMBus ALERT 可选信号。仅从设备可以通过它想要通话的 SMBALERT＃引脚向主控者发送信号。主控者处理中断并同时通过警报响应地址（0b0001100）访问所有 SMBALERT＃设备。只有拉低 SMBALERT＃的设备才会应答警报响应地址。当配置为从设备（SMBHEN = 0）时，通过将 I2Cx_CON1 寄存器中的 ALERTEN 位置 1，可将 SMBA 引脚拉低。警报响应地址同时启用。当配置为主控者（SMBHEN = 1）时，如果在 SMBA 引脚上检测到下降沿且 ALERTEN = 1，则在 I2Cx_RIF 寄存器中设置 ALERTRI 标志。如果 I2Cx_IER 寄存器中的 ALERTIE 位置 1，则会产生中断。当 ALERTEN = 0 时，即使外

部 SMBA 引脚为低电平，ALERT 线也会被视为高电平。如果不需要 SMBus ALERT 引脚，若 ALERTEN = 0，则 SMBA 引脚可用作标准 GPIO。

数据包错误检查

SMBus 规范中引入了一种数据包错误检查机制，以提高可靠性和通信稳健性。通过在每次消息传输结束时附加分组错误代码（PEC）来实现分组错误检查。通过在所有消息字节（包括地址和读/写位）上使用 C（x） = X8 + X2 + X + 1 CRC-8 多项式来计算 PEC。外设嵌入了硬件 PEC 计算器，当接收到的字节与硬件计算的 PEC 不匹配时，允许自动发送非应答。

超时

该外设嵌入了硬件定时器，以符合 SMBus 规范 2.0 版中定义的 3 个超时

标记	参数	范围		单位
		最小	最大	
$T_{TIMEOUT}$	检测时钟低超时	25	35	ms
$T_{LOW:SEXT}$ (1)	累积时钟低延长时间（从设备）	-	25	ms
$T_{LOW:MEXT}$ (2)	累积时钟低延长时间（主设备）	-	10	ms

表 25-5 SMBus 超时规格

注 1：$T_{LOW:SEXT}$ 是允许给定从设备在从初始 START 到 STOP 的一条消息中延长时钟周期的累积时间。另一个从设备或主设备也可能延长时钟，导致组合时钟低延长时间大于 $T_{LOW:SEXT}$。因此，该参数是在从设备作为全速主设备的唯一目标的情况下测量的。

注 2：$T_{LOW:MEXT}$ 是允许主设备在从 START 到 ACK，ACK 到 ACK 或 ACK 到 STOP 定义的消息的每个字节内延长其时钟周期的累积时间。从设备或另一个主设备也可能延长时钟，导致组合时钟低电平时间大于给定字节上的 $T_{LOW:MEXT}$。因此，使用全速从设备作为主设备的唯一目标来测量该参数。

总线空闲检测

如果总线检测到时钟和数据信号已经持续高电平并且 T_{IDLE} 大于 $T_{HIGH.MAX}$，则主设备可以假设总线是空闲的。

该时序参数涵盖了主机已动态添加到总线并且可能未检测到 SMBCLK 或 SMBDAT 线路上的状态转换的情况。在这种情况下，主设备必须等待足够长的时间以确保当前没有进行传输。外设支持硬件总线空闲检测。

25. 4. 8 SMBus初始化

除了 I2C 初始化之外，还必须进行一些其它特定的初始化以进行 SMBus 通信：

接收命令和数据应答控制（从机模式）

SMBus 接收器必须能够 NACK 每个接收到的命令或数据。为了在从机模式下允许 ACK 控制，必须通过将 I2Cx_CON1 寄存器中的 SBC 位置 1 来启用从机字节控制模式。

特定地址（从机模式）

如果需要，应启用特定的 SMBus 地址。

通过将 I2Cx_CON1 寄存器中的 SMBDEN 位置 1 启用 SMBus 设备从机地址（0b1100 001）。

通过将 I2Cx_CON1 寄存器中的 SMBHEN 位置 1 启用 SMBus 主控者从机地址（0b0001 000）。

通过将 I2Cx_CON1 寄存器中的 ALERTEN 位置 1 启用报警响应地址（0b0001100）。

数据包错误检查

通过将 I2Cx_CON1 寄存器中的 PECEN 位置 1 启用 PEC 计算。然后在硬件字节计数器的帮助下管理 PEC 传输：I2Cx_CON2 寄存器中的 NBYTES<7:0>。必须先配置 PECEN 位，然后再启用 I2C。

PEC 传输由硬件字节计数器管理，因此在从机模式下连接 SMBus 时必须设置 SBC 位。在 PECBYTE 位置 1 且 RELOAD 位清零后，在传输 NBYTES-1 数据后传输 PEC。如果设置了 RELOAD，则 PECBYTE 无效。

注意：启用 I2C 时，不允许更改 PECEN 配置。

超时检测

通过将 I2Cx_TIMEOUTR 寄存器中的 TIMEOUTEN 和 TEXTEN 位置 1 来启用超时检测。定时器必须以这样的方式编程，即它们在 SMBus 规范 2.0 版中给出的最大时间之前检测到超时。

◇ $T_{TIMEOUT}$ 检查

为了启用 $T_{TIMEOUT}$ 检查，必须对 12 位 TIMEOUTA 定时器进行编程，以检查 $T_{TIMEOUT}$ 参数。必须将 TIDLE 位配置为“0”才能检测 SCL 低电平超时。然后通过设置 I2Cx_TIMEOUTR 寄存器中的 TIMEOUTEN 来启用定时器。如果 SCL 在大于（TIMEOUTA + 1）* 2048 * T_{I2CCLK} 的时间内被拉低，则在 I2Cx_RIF 寄存器中设置 TOUTRI 标志。

注意：当 TIMEOUTEN 位置 1 时，不允许更改 TIMEOUTA<11:0>位和 TIDLE 位配置。

◇ $T_{LOW:SEXT}$ 和 $T_{LOW:MEXT}$ 检查

根据外设是配置为主机还是从机，必须配置 12 位 TIMEOUTB 定时器，以便检查从机的 $T_{LOW:SEXT}$ 和主机的 $T_{LOW:MEXT}$。由于标准仅指定最大值，因此用户可以为两者选择相同的值。然后，通过将 I2Cx_TIMEOUTR 寄存器中的 TEXTEN 位置 1 来启用定时器。

注意：设置 TEXTEN 位时，不允许更改 TIMEOUTB 配置。

总线空闲检测

为了启用 TIDLE 检查，必须对 12 位 TIMEOUTA 定时器进行编程，以获得 TIDLE 参数。必须将 TIDLE 位配置为 1 才能检测 SCL 和 SDA 高电平超时。

然后，通过将 I2Cx_TIMEOUTR 寄存器中的 TIMEOUTEN 位置 1 来启用定时器。如果 SCL 和 SDA 线都保持高电平的时间大于（TIMEOUTA + 1）* 4 * T_{I2CCLK}，则在 I2Cx_RIF 寄存器中设置 TOUTRI 标志。

注意：设置 TIMEOUTEN 时，不允许更改 TIMEOUTA 和 TIDLE 配置。

25. 4. 9 SMBus：I2Cx_TIMEOUTR寄存器配置的例子

◇ 将 $T_{TIMEOUT}$ 的最大持续时间配置为 25 ms

F_{I2CCLK}	TIMEOUTA<11:0>	TIDLE	TIMEOUTEN	$T_{TIMEOUT}$
8 MHz	0x61	0	1	98 * 2048 * 125 ns = 25 ms
16 MHz	0xC3	0	1	196 * 2048 * 62.5 ns = 25 ms
32 MHz	0x186	0	1	391 * 2048 * 31.25 ns = 25 ms

表 25-6 各种 I2CCLK 频率的 TIMEOUTA 设置示例（最大值 $T_{TIMEOUT}$ = 25 ms）

◇ 将 $T_{LOW:SEXT}$ 和 $T_{LOW:MEXT}$ 的最大持续时间配置为 8 ms

F_{I2CCLK}	TIMEOUTB<11:0>	TIMEOUTEN	$T_{LOW:SEXT}$
8 MHz	0x1F	1	32 * 2048 * 125 ns = 8 ms
16 MHz	0x3F	1	64 * 2048 * 62.5 ns = 8 ms
32 MHz	0x7C	1	125 * 2048 * 31.25 ns = 8 ms

表 25-7 各种 I2CCLK 频率的 TIMEOUTB 设置示例（最大值 $T_{TIMEOUT}$ = 8 ms）

◇ 将 T_{IDLE} 的最大持续时间配置为 50μs

F_{I2CCLK}	TIMEOUTA<11:0>	TIDLE	TIMEOUTEN	T_{IDLE}
8 MHz	0x63	1	1	100 * 4 * 125 ns = 50 μs
16 MHz	0xC7	1	1	200 * 4 * 62.5 ns = 50 μs
32 MHz	0x18F	1	1	400 * 4 * 31.25 ns = 50 μs

表 25-8 各种 I2CCLK 频率的 TIMEOUTA 设置示例（最大 T_{IDLE} =50μs）

25. 4. 10 DMA请求

使用 DMA 发送

通过将 I2Cx_CON1 寄存器中的 TXDMAEN 位置 1，可以启用 DMA 进行发送。只要 TXTH 位置 1，就会从使用 DMA 外设配置的 SRAM 区域加载数据到 I2Cx_TXDATA 寄存器。只有数据通过 DMA 发送。

使用 DMA 接收

通过将 I2Cx_CON1 寄存器中的 RXDMAEN 位置 1，可以启用 DMA 以进行接收。只要 RXTH 位置 1，就会将数据从 I2Cx_RXDATA 寄存器加载到使用 DMA 外设配置的 SRAM 区域。仅使用 DMA 传输数据（包括 PEC）。

25. 4. 11 错误情况

以下是可能导致通信失败的错误情况。

总线错误（BERR）

总线错误是指在总线上不在 9 个 SCL 时钟脉冲的倍数之后检测到 START 或 STOP 条件。仅当 I2C 作为主机或寻址从机进行传输时（不在从机模式下的地址阶段），才会设置总线错误标志。

如果在从机模式下检测到错误的 START 或 RESTART，则 I2C 会进入地址识别状态，就像正确的 START 条件一样。

检测到总线错误时，BERRRI 标志位在 I2Cx_RIF 寄存器中，如果 I2Cx_IER 寄存器中的 BERRIE 位置 1，则产生中断。

仲裁丢失（ARLO）

仲裁丢失为当在 SDA 线上发送高电平时，但在 SCL 上升沿上采样到低电平。

- ◇ 在主机模式下，会在地址阶段，数据阶段和数据应答阶段检测仲裁丢失。在这种情况下，SDA 和 SCL 线被释放，START 控制位由硬件清零，主机自动切换到从机模式。
- ◇ 在从机模式下，会在数据阶段和数据应答阶段检测仲裁丢失。在这种情况下，传输停止，SCL 和 SDA 线被释放。当检测到仲裁丢失时，ARLORI 标志位在 I2Cx_RIF 寄存器中，并且如果在 I2Cx_IER 寄存器中设置了 ARLOIE 位，则会产生中断。

接收溢出/ 发送欠载错误（RXOV / TXUD）

当 NOSTRETCH = 1 时，在从机模式下检测到溢出或欠载错误：

- ◇ 当接收到新字节且接收 FIFO 满时。新接收的字节将会丢失，并且自动发送 NACK 作为对新字节的应答。
- ◇ 在传输中：当应发送新字节且尚未写入发送 FIFO 时，将发送 0xFF。

当检测到接收溢出或发送欠载错误时，TXUD 与 RXOV 标志位在 I2Cx_STAT 寄存器中，如果 I2Cx_IER 寄存器中的 TXUDIE 或 RXOVIE 位置 1，则会产生中断

25. 4. 12 I2C中断

I2C 中的中断由一组六个寄存器控制。

◇ 中断控制（IER，IDR，IVS）

I2C 中断使能寄存器（I2Cx_IER）通过写 1 来启用中断请求线。同样，I2C 中断禁止寄存器（I2Cx_IDR）通过写“1”来禁止中断。IER 和 IDR 是只写寄存器，上述寄存器的总结果可由中断有效状态寄存器（I2Cx_IVS）显示。IVS 是一个只读寄存器，使用“1”或“0”来指示中断请求线是否有效。

◇ 原始中断标志状态读取（RIF）

I2C 原始中断状态（I2Cx_RIF）是一个只读寄存器，用于读取模块的中断状态。该寄存器中的位显示 I2C 中断的真实状态。当观察到以下条件时，I2C 可以产生中断：

- SMBus 警报
- 发生超时
- PEC 错误
- 仲裁丢失
- 总线错误
- 传送完成并重新加载
- 传送完成
- 检测检测 STOP
- 收到非应答
- 地址匹配
- 接收 FIFO 为满
- 发送 FIFO 为空
- 接收 FIFO 超过阈值
- 发送 FIFO 低于阈值
- 发生接收溢出或发送欠载

◇ 中断标志屏蔽状态（IFM）

I2C 中断标志屏蔽状态寄存器（I2Cx_IFM）用于读取模块的屏蔽中断状态，显示屏蔽了哪个中断。IFM 中的每个位是 IVS 和 RIF 中各个位的逻辑 AND。

◇ 中断清除（ICR）

向该寄存器的位写“1”可以清除相应的中断。

I2C 中断可根据通信状态分为事件中断和错误中断两种类型，用户可参考中断向量分配表的入口地址进行中断服务程序划分和编写。如果产品只为 I2C 分配 1 个中断向量，则 I2C 所有中断类型须在同一中断服务程序中处理。每个中断源对应的中断类型请参考下图。

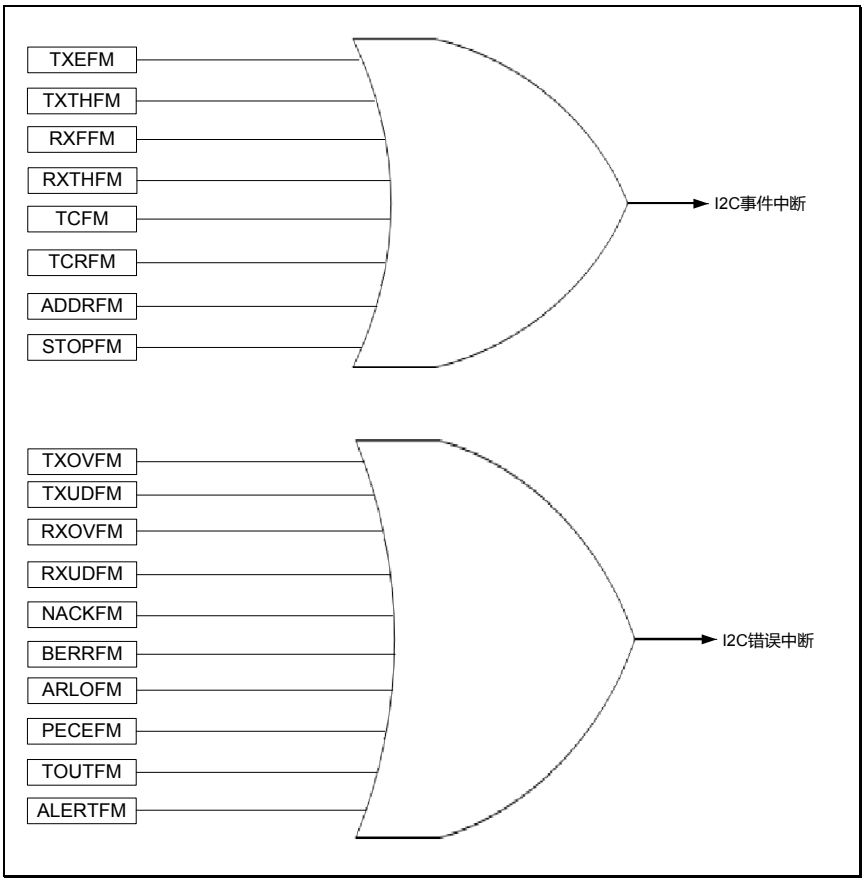


图 25-16 I2C 中断映射图

25. 5 特殊功能寄存器

25. 5. 1 寄存器列表

I2C 寄存器列表		
名称	偏移地址	描述
I2Cx_CON1	000_H	I2C 控制寄存器 1
I2Cx_CON2	004_H	I2C 控制寄存器 2
I2Cx_ADDR1	008_H	I2C 本机地址寄存器 1
I2Cx_ADDR2	$00C_H$	I2C 本机地址寄存器 2
I2Cx_TIMINGR	010_H	I2C 时序寄存器
I2Cx_TIMEOUTR	014_H	I2C 超时寄存器
I2Cx_STAT	018_H	I2C 状态寄存器
I2Cx_FCON	$01C_H$	I2C FIFO 控制寄存器
I2Cx_PECR	020_H	I2C PEC 寄存器
I2Cx_RXDATA	024_H	I2C 接收数据寄存器
I2Cx_TXDATA	028_H	I2C 发送数据寄存器
I2Cx_IER	$02C_H$	I2C 中断使能寄存器
I2Cx_IDR	030_H	I2C 中断禁止寄存器
I2Cx_IVS	034_H	I2C 中断有效状态寄存器
I2Cx_RIF	038_H	I2C 原始中断标志寄存器
I2Cx_IFM	$03C_H$	I2C 中断标志屏蔽寄存器
I2Cx_ICR	040_H	I2C 中断清除寄存器

25. 5. 2 寄存器描述

25. 5. 2. 1 I2C控制寄存器 1（I2Cx_CON1）

I2C 控制寄存器 1（I2Cx_CON1）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	Reserved	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Reserved													PE

Reserved	Bit 31-24	—	**保留**
PECEN	Bit 23	R/W	**PEC 使能** 0：禁止 PEC 计算 1：使能 PEC 计算 注意：如果不支持 SMBus 功能，则该位保留并由硬件强制为“0”。
ALERTEN	Bit 22	R/W	**SMBus 警报使能** 设备模式（SMBHEN = 0）： 0：释放 SMBA 引脚为高电平且禁止警报响应地址标头： 0001100x 并回应 NACK。 1：将 SMBA 引脚驱动为低电平且使能警报响应地址标头：0001100x 并回应 ACK。 主控者模式（SMBHEN = 1）： 0：不支持 SMBus 警报引脚（SMBA）。 1：支持 SMBus 警报引脚（SMBA）。 注：当 ALERTEN = 0 时，SMBA 引脚可用作标准 GPIO。如果不支持 SMBus 功能，则该位保留并由硬件强制为“0”。
SMBDEN	Bit 21	R/W	**SMBus 设备从机地址使能** 0：禁止设备从机地址。地址 0b1100001x 无应答。 1：使能设备从机地址。地址 0b1100001x 被应答。 注意：如果不支持 SMBus 功能，则该位保留并由硬件强制为“0”。
SMBHEN	Bit 20	R/W	**SMBus 主控者地址使能** 0：禁止主控者地址。地址 0b0001000x 无应答。 1：使能主控者地址。地址 0b0001000x 被应答。 注意：如果不支持 SMBus 功能，则该位保留并由

			硬件强制为“0”。
GCEN	Bit 19	R/W	**广播呼叫使能** 0：禁止广播呼叫。地址 0b00000000 无应答。 1：使能广播呼叫。地址 0b00000000 被应答。
Reserved	Bit 18	—	**保留**
NOSTRETCH	Bit 17	R/W	**时钟延长禁止** 该位用于禁止从机模式下的时钟延长。必须在主机模式下保持清除状态。 0：使能时钟延长 1：禁止时钟延长 注：只有在 I2C 被禁止（PE = 0）时才能对该位进行编程。
SBC	Bit 16	R/W	**从机字节控制** 该位用于在从机模式下使能硬件字节控制。 0：禁止从机字节控制 1：使能从机字节控制
RXDMAEN	Bit 15	R/W	**使能 DMA 接收请求** 0：禁止 DMA 模式进行接收 1：使能 DMA 模式进行接收
TXDMAEN	Bit 14	R/W	**使能 DMA 发送请求** 0：禁止 DMA 模式进行发送 1：使能 DMA 模式进行发送
Reserved	Bit 13-1	—	**保留**
PE	Bit 0	R/W	**I2C 使能** 0：I2C 禁止 1：I2C 使能 注：当 PE = 0 时，I2C SCL 和 SDA 线被释放。内部状态机和状态位将恢复为其复位值。清零后，PE 必须保持低电平至少 3 个 APB 时钟周期。

25. 5. 2. 2 I2C控制寄存器 2（I2Cx_CON2）

I2C 控制寄存器 2（I2Cx_CON2）

偏移地址：04_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		ACK_UPD	HOLDACK	Reserved	PECBYTE	AUTOEND	RELOAD	NBYTES								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD									

Reserved	Bit 31-30	—	**保留**
ACK_UPD	Bit 29	R/C_W1	**ACK 更新** 0：不更新 ACK 状态。 1：更新 ACK 状态并释放时钟。
HOLDACK	Bit 28	R/W	**等待回应 ACK** 使能时，第 8 和第 9 个 SCL 脉冲之间会将 SCL 延长，并等待用户设置 ACK / NACK。在设置 ACK / NACK 之后，设置 ACK_UPD 以发出 ACK / NACK 并释放 SCL。 0：自动应答 ACK。 1：使能手动回应 ACK。
Reserved	Bit 27	—	**保留**
PECBYTE	Bit 26	R/C_W1	**数据包错误检查字节** 该位由软件置 1，当传输 PEC 时，或者当接收到 STOP 条件或匹配的地址时，由硬件清零，当 PE = 0 时也是如此。 0：没有 PEC 传输。 1：请求 PEC 发送/接收 注意：向该位写“0”无效。 设置 RELOAD 时，该位无效。 当 SBC = 0 时，该位对从机模式无效。 如果不支持 SMBus 功能，则该位保留并由硬件强制为“0”。
AUTOEND	Bit 25	R/W	**自动结束模式（主机模式）** 该位由软件置位和清除。 0：软件结束模式：发送完 NBYTES 数据时设置 TC 标志，将 SCL 拉低。 1：自动结束模式：发送完 NBYTES 数据时自动发送 STOP 条件。

			注：该位在从机模式或 RELOAD 位置 1 时无效。
RELOAD	Bit 24	R/W	**NBYTES 重新加载模式** 该位由软件置位和清除。 0：在 NBYTES 数据传输之后完成传输（接下来是 STOP 或 RESTART）。 1：NBYTES 数据传输后将继续传输（NBYTES 将重新加载）。发送完 NBYTES 数据后设置 TCR 标志，将 SCL 拉低。
NBYTES	Bit 23-16	R/W	**字节数** 在此编程要发送/接收的字节数。 在 SBC = 0 的从机模式下，该字段无关紧要。 注：不允许在设置 START 位时更改这些位。
NACK	Bit 15	R/C_W1	**NACK 生成（从机模式）** 该位由软件置位，在发送 NACK 时由硬件清零，或者在接收到 STOP 条件或地址匹配时，或者当 PE = 0 时由硬件清零。 0：在下一个接收的字节发送 ACK。 1：在下一个接收的字节发送 NACK。 注意：向该位写“0”无效。 该位仅用于从机模式：在主机接收模式下，无论 NACK 位值如 何，在 STOP 或 RESTART 条件之前的最后一个字节之后自动生成 NACK。当从机接收 NOSTRETCH 模式发生溢出时，无论 NACK 位值如何，都会自动生成 NACK。当使能硬件 PEC 检查（PECBYTE = 1）时，PEC 确认值不依赖于 NACK 值。
STOP	Bit 14	R/C_W1	**STOP 生成（主机模式）** 该位由软件置位，当检测到停止条件时，或者当 PE = 0 时由硬件清零。 在主模式下： 0：无 STOP 生成。 1：当前字节传输后 STOP 生成。 注意：向该位写“0”无效。
START	Bit 13	R/C_W1	**START 生成** 该位由软件置位，并在启动后跟随地址序列发送，仲裁丢失，超时错误检测或 PE = 0 时由硬件清零。 0：无 START 生成。 1：RESTART / START 生成：

			- 如果 I2C 已经处于主模式且 AUTOEND = 0，则在 NBYTES 传输结束后，当 RELOAD = 0 时，将该位置 1 会产生重复启动条件。 - 否则，一旦总线空闲，将该位置 1 产生 START 条件。 注意：向该位写“0”无效。 即使总线忙，也可以设置 START 位。 在 10 位寻址模式下，如果在地址的第一部分接收到 NACK，则 START 位不会被硬件清零，主机将重新发送地址序列，除非 START 位被软件清零
HEAD10R	Bit 12	R/W	**10 位地址标头仅读取方向（主机接收模式）** 0：主机发送完整的 10 位从机地址读取序列：在写入方向上启动+ 2 字节 10 位地址+ 在读取方向上重新启动+ 10 位地址的前 7 位，然后发送读取方向。。 1：主机仅发送 10 位地址的前 7 位，然后发送读取方向。 注：不允许在设置 START 位时更改该位。
ADD10	Bit 11	R/W	**10 位寻址模式（主机模式）** 0：主机工作在 7 位寻址模式 1：主机工作在 10 位寻址模式 注：不允许在设置 START 位时更改该位。
RD_WRN	Bit 10	R/W	**传输方向（主机模式）** 0：主机请求写入传输。 1：主机请求读取传输。 注：不允许在设置 START 位时更改该位。
SADD	Bit 9-0	R/W	**从机地址位 0（主机模式）** 在 7 位寻址模式下（ADD10 = 0）： 此位不在乎 在 10 位寻址模式下（ADD10 = 1）： 该位应写入要发送的从地址的第 0 位 **从机地址位 7：1（主机模式）** 在 7 位寻址模式下（ADD10 = 0）： 应使用要发送的 7 位从地址写入这些位 在 10 位寻址模式下（ADD10 = 1）： 应使用要发送的从地址的位 7：1 写入这些位。 **从机地址位 9：8（主机模式）** 在 7 位寻址模式下（ADD10 = 0）： 这些位都不在乎

			在 10 位寻址模式下（ADD10 = 1）： 应使用要发送的从地址的第 9：8 位写入这些位 注：不允许在设置 START 位时更改这些位。

25. 5. 2. 3 I2C本机地址寄存器 1（I2Cx_ADDR1）

I2C 本机地址寄存器 1（I2Cx_ADDR1）

偏移地址：08_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																OA1EN	Reserved				OA1MODE	OA1									

Reserved	Bit 31-16	—	**保留**
OA1EN	Bit 15	R/W	**本机地址 1 使能** 0：禁止本机地址 1。接收到的从机地址 OA1 无应答。 1：使能本机地址 1。接收到的从机地址 OA1 被应答。
Reserved	Bit 14-11	—	**保留**
OA1MODE	Bit 10	R/W	**本机地址 1，10 位地址模式使能** 0：本机地址 1 是 7 位地址。 1：本机地址 1 是 10 位地址。 注：仅当 OA1EN = 0 时才能写入该位。
OA1	Bit 9-0	R/W	**OA1<0>：本机地址 1** 7 位寻址模式：不在乎 10 位寻址模式：地址的第 0 位 **OA1<7:1>：本机地址 1** 7 位寻址模式：7 位地址 10 位寻址模式：10 位地址的 7：1 位 **OA1<9:8>：本机地址 1** 7 位寻址模式：不在乎 10 位寻址模式：地址的 9：8 位 注：仅当 OA1EN = 0 时才能写入该位。

25. 5. 2. 4 I2C本机地址寄存器 2（I2Cx_ADDR2）

I2C 本机地址寄存器 2（I2Cx_ADDR2）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																OA2EN	Reserved				OA2MSK			OA2							Reserved

Reserved	Bit 31-16	—	**保留**
OA2EN	Bit 15	R/W	**本机地址 2 使能** 0：禁止本机地址 2。接收到的从机地址 OA2 无应答。 1：使能本机地址 2。接收到的从机地址 OA2 被应答。
Reserved	Bit 14-11	—	**保留**
OA2MSK	Bit 10-8	R/W	**本机地址 2 屏蔽** 000：没有屏蔽 001：OA2<0>被屏蔽并且忽略。仅比较 OA2<6:1>。 010：OA2<1:0>被屏蔽并且忽略。仅比较 OA2<6:2>。 011：OA2<2:0>被屏蔽并且忽略。仅比较 OA2<6:3>。 100：OA2<3:0>被屏蔽并且忽略。仅比较 OA2<6:4>。 101：OA2<4:0>被屏蔽并且忽略。仅比较 OA2<6:5>。 110：OA2<5:0>被屏蔽并且忽略。仅比较 OA2<6>。 111：OA2<6:0>被屏蔽并且忽略。不进行比较，并且应答所有（除了保留的）7 位接收地址。 注意：仅当 OA2EN = 0 时才能写入这些位。 一旦 OA2MSK 不等于 0，则 I2C 即使比较匹配，也不会应答保留地址（0b0000xxx 和 0b1111xxx）。
OA2	Bit 7-1	R/W	**本机地址 2** 7 位寻址模式：7 位地址 注意：仅当 OA2EN = 0 时才能写入这些位。
Reserved	Bit 0	—	**保留**

25. 5. 2. 5 I2C时钟寄存器（I2Cx_TIMINGR）

I2C 时序寄存器（I2Cx_TIMINGR）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

PRESC	Bit 31-28	R/W	**时钟预分频器** 该字段用于预分频 I2CCLK，以产生用于数据设置和保持计数器以及 SCL 高电平和低电平计数器的时钟周期 T_{PRESC}。 T_{PRESC} =（PRESC + 1）* T_{I2CCLK} 禁止 I2C（PE = 0）时，必须配置该寄存器。
Reserved	Bit 27-24	—	**保留**
SCLDEL	Bit 23-20	R/W	**数据设置时间** 该字段用于在 SDA 边沿和 SCL 上升沿之间生成延迟 T_{SCLDEL}。在主机和从机模式下，NOSTRETCH = 0 时，SCL 线在 T_{SCLDEL} 期间拉低。 T_{SCLDEL} =（SCLDEL + 1）* T_{PRESC} 注意：T_{SCLDEL} 用于生成 $T_{SU:\ DAT}$ 时序。 禁止 I2C（PE = 0）时，必须配置该寄存器。
SDADEL	Bit 19-16	R/W	**数据保持时间** 该字段用于在 SCL 下降沿和 SDA 边沿之间生成延迟 T_{SDADEL}。在主机和从机模式下，NOSTRETCH = 0 时，SCL 线在 T_{SDADEL} 期间拉低。 T_{SDADEL}= SDADEL * T_{PRESC} 注意：SDADEL 用于生成 $T_{HD:\ DAT}$ 时序。 禁止 I2C（PE = 0）时，必须配置该寄存器。
SCLH	Bit 15-8	R/W	**SCL 高电平期间（主机模式）** 该字段用于在主机模式下生成 SCL 高电平周期。 T_{SCLH} =（SCLH + 1）* T_{PRESC} 注意：SCLH 还用于生成 $T_{SU:\ STO}$ 和 $T_{HD:\ STA}$ 时序。 禁止 I2C（PE = 0）时，必须配置该寄存器。
SCLL	Bit 7-0	R/W	**SCL 低电平周期（主机模式）** 该字段用于在主机模式下生成 SCL 低电平周期. T_{SCLL} =（SCLL + 1）* T_{PRESC}

			注意：SCLL 还用于生成 T_{BUF} 和 $T_{SU:\ STA}$ 时序。 禁止 I2C（PE = 0）时，必须配置该寄存器。

25. 5. 2. 6 I2C超时寄存器（I2Cx_TIMEOUTR）

I2C 超时寄存器（I2Cx_TIMEOUTR）

偏移地址：14_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

TEXTEN	Bit 31	R/W	**累积时钟延长超时使能** 0：禁止累积时钟延长超时检测 1：使能累积时钟延长超时检测。 当 I2C 接口完成累积 SCL 延长超过 $T_{LOW:\ EXT}$ 时，会检测到超时错误（TOUTRI = 1）。
Reserved	Bit 30-28	—	**保留**
TIMEOUTB	Bit 27-16	R/W	**总线超时 B** 该字段用于配置累积时钟延长超时： 在主机模式下，检测到主机累积时钟低延长时间（$T_{LOW\ :\ MEXT}$）。 在从机模式下，检测到从机累积时钟低延长时间（$T_{LOW\ :\ SEXT}$）。 $T_{LOW\ :\ EXT}$ =（TIMEOUTB + 1）* 2048 * T_{I2CCLK} 注意：仅当 TEXTEN = 0 时才能写入这些位。
TIMEOUTEN	Bit 15	R/W	**时钟超时使能** 0：禁止 SCL 超时检测 1：使能 SCL 超时检测：当 SCL 为低电平超过 $T_{TIMEOUT}$（TIDLE = 0）或高电平超过 T_{IDLE}（TIDLE = 1）时，检测到超时错误（TOUTRI = 1）。
Reserved	Bit 14-13	—	**保留**
TIDLE	Bit 12	R/W	**空闲时钟超时检测** 0：TIMEOUTA 用于检测 SCL 低超时 1：TIMEOUTA 用于检测 SCL 和 SDA 高超时（总线空闲状态） 注：仅当 TIMEOUTEN = 0 时才能写入该位。
TIMEOUTA	Bit 11-0	R/W	**总线超时 A** 该字段用于配置： - 当 TIDLE = 0 时，SCL 低超时条件 $T_{TIMEOUT}$

			$T_{TIMEOUT}$ =（TIMEOUTA + 1）* 2048 * T_{I2CCLK} - 当 TIDLE = 1 时，总线空闲状态（SCL 和 SDA 均为高电平） T_{IDLE} =（TIMEOUTA + 1）* 4 * T_{I2CCLK} 注意：仅当 TIMEOUTEN = 0 时才能写入这些位。

25. 5. 2. 7 I2C状态寄存器（I2Cx_STAT）

I2C 状态寄存器（I2Cx_STAT）

偏移地址：18_H

复位值：$00000000\_00000000\_00000000\_00110001_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ADDCODE							DIR	BUSY	Reserved			TCR	TC	RXTH	RXUD	RXOV	RXF	RXE	TXTH	TXUD	TXOV	TXF	TXE

Reserved	Bit 31-24	—	**保留**
ADDCODE	Bit 23-17	R	**地址匹配代码（从机模式）** 当发生地址匹配事件（ADDR = 1）时，使用接收的地址更新这些位。在 10 位地址的情况下，ADDCODE 提供 10 位标头与地址的 2 个 MSB。
DIR	Bit 16	R	**传输方向（从机模式）** 发生地址匹配事件（ADDRRI = 1）时更新此标志。 0：写入传输，从机进入接收模式。 1：读取传输，从机进入发送模式。
BUSY	Bit 15	R	**总线忙** 该标志表示总线上正在进行通信。检测到 START 条件时由硬件设置。当检测到停止条件或 PE = 0 时，它由硬件清零。
Reserved	Bit 14-12	—	**保留**
TCR	Bit 11	R	**传输完成并重新加载** 当 RELOAD = 1 且已将 NBYTES 数据传输完成时，此标志由硬件设置。当 NBYTES 写入非零值时，它由硬件清零。 注：当 PE = 0 时，该位由硬件清零。 该标志仅用于主模式，或者用于从模式 SBC 位置 1 时。
TC	Bit 10	R	**传输完成（主机模式）** 当 RELOAD = 0，AUTOEND = 0 且已将 NBYTES 数据传输完成时，该标志由硬件设置。当 START 位或 STOP 位置 1 时，由硬件清零。 注：当 PE = 0 时，该位由硬件清零。
RXTH	Bit 9	R	**接收 FIFO 水平超出阈值** 该位提供接收 FIFO 水平超过阈值。

			0：接收 FIFO 水平没有超过阀值。 1：接收 FIFO 水平超过阀值。
RXUD	Bit 8	R	**接收 FIFO 下溢** 0：接收 FIFO 没有下溢 1：接收 FIFO 下溢
RXOV	Bit 7	R	**接收 FIFO 溢出** 0：接收 FIFO 没有溢出 1：接收 FIFO 溢出
RXF	Bit 6	R	**接收 FIFO 满** 0：接收 FIFO 没有满 1：接收 FIFO 满
RXE	Bit 5	R	**接收 FIFO 空** 0：接收 FIFO 没有空 1：接收 FIFO 空
TXTH	Bit 4	R	**发送 FIFO 水平低于阈值** 该位提供发送 FIFO 水平低于阈值。 0：发送 FIFO 水平高于阀值。 1：发送 FIFO 水平低过阀值。
TXUD	Bit 3	R	**发送 FIFO 下溢** 0：发送 FIFO 没有下溢 1：发送 FIFO 下溢
TXOV	Bit 2	R	**发送 FIFO 溢出** 0：发送 FIFO 没有溢出 1：发送 FIFO 溢出
TXF	Bit 1	R	**发送 FIFO 满** 0：发送 FIFO 没有满 1：发送 FIFO 满
TXE	Bit 0	R	**发送 FIFO 空** 0：发送 FIFO 没有空 1：发送 FIFO 空

25. 5. 2. 8 I2C FIFO控制寄存器（I2Cx_FCON）

I2CFIFO 控制寄存器（I2Cx_FCON）

偏移地址：$1C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	保留
RXFTH	Bit 15-14	R/W	**接收 FIFO 触发阈值** 这用于选择接收 FIFO 生成接收 FIFO 触发阈值标志的阈值电平。 00：当接收 FIFO 中有 1 个字符时触发 01：当接收 FIFO 中有 4 个字符时触发 10：当接收 FIFO 中有 8 个字符时触发 11：当接收 FIFO 中有 14 个字符时触发
RXFRST	Bit 13	C_W1	**接收 FIFO 复位** 当 FIFO 置 1 时，接收 FIFO 中的所有字节都被清除，并将 FIFO 视为空。请注意，该位将在下一个时钟周期内返回 0。 0：没有 FIFO 接收复位 1：重置接收指针 该位由硬件自动清除。
RXFLV	Bit 12-8	R	**接收 FIFO 电平** 该位由硬件设置。这用于指示接收 FIFO 中的数据条目数。 它从 0 到 16。
TXFTH	Bit 7-6	R/W	**发送 FIFO 触发阈值** 这用于选择发送 FIFO 产生发送 FIFO 触发阈值标志的阈值电平。 00：当发送 FIFO 空时触发 01：当发送 FIFO 中有 2 个字符时触发 10：当发送 FIFO 中有 4 个字符时触发 11：当发送 FIFO 中有 8 个字符时触发
TXFRST	Bit 5	C_W1	**发送 FIFO 复位** 当 FIFO 置 1 时，发送 FIFO 中的所有字节都被清除，并将 FIFO 视为空。请注意，该位将在下一个时钟周期内返回 0。 0：无 FIFO 发送复位

			1：重置发送指针 该位由硬件自动清除。
TXFLV	Bit 4-0	R	**发送 FIFO 电平** 该位由硬件设置。这用于指示发送 FIFO 中的数据条目数。 它从 0 到 16。

25. 5. 2. 9 I2C PEC寄存器（I2Cx_PECR）

I2CPEC 寄存器（I2Cx_PECR）

偏移地址：20_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-8	—	**保留**
PEC	Bit 7-0	R	**数据包错误检查寄存器** 当 PECEN = 1 时，该字段包含内部 PEC。 当 PECEN = 0 时，PEC 由硬件清零。

25. 5. 2. 10 I2C接收数据寄存器（I2Cx_RXDATA）

I2C 接收数据寄存器（I2Cx_RXDATA）

偏移地址：24_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-8	—	**保留**
RXDATA	Bit 7-0	R	**8 位接收数据** 从 I2C 总线接收的数据字节。

25. 5. 2. 11 I2C发送数据寄存器（I2Cx_TXDATA）

I2C 发送数据寄存器（I2Cx_TXDATA）

偏移地址：28_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-8	—	**保留**
TXDATA	Bit 7-0	W	**8 位发送数据** 要传输到 I2C 总线的数据字节。

25. 5. 2. 12 I2C中断使能寄存器（I2Cx_IER）

I2C 中断使能寄存器（I2Cx_IER）

偏移地址: $01C_H$

复位值: $00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											ALERTIE	TOUTIE	PECEIE	ARLOIE	BERRIE	Reserved	STOPIE	NACKIE	ADDRIE	TCRIE	TCIE	RXTHIE	RXUDIE	RXOVIE	RXFIE	Reserved	TXTHIE	TXUDIE	TXOVIE	Reserved	TXEIE

Reserved	Bit 31-21	—	保留
ALERTIE	Bit 20	W1	**SMBus 报警中断使能** 0：无效 1：使能中断
TOUTIE	Bit 19	W1	**超时中断使能** 0：无效 1：使能中断
PECEIE	Bit 18	W1	**PEC 错误中断使能** 0：无效 1：使能中断
ARLOIE	Bit 17	W1	**仲裁丢失中断使能** 0：无效 1：使能中断
BERRIE	Bit 16	W1	**总线错误中断使能** 0：无效 1：使能中断
Reserved	Bit 15	—	**保留**
STOPIE	Bit 14	W1	**停止检测中断使能** 0：无效 1：使能中断
NACKIE	Bit 13	W1	**NACK 接收中断使能** 0：无效 1：使能中断
ADDRIE	Bit 12	W1	**地址匹配中断使能** 0：无效 1：使能中断
TCRIE	Bit 11	W1	**传输完成并重新加载中断使能** 0：无效 1：使能中断
TCIE	Bit 10	W1	**传输完成中断使能** 0：无效 1：使能中断

RXTHIE	Bit 9	W1	**接收 FIFO 超过阈值中断使能** 0：无效 1：使能中断
RXUDIE	Bit 8	W1	**接收 FIFO 欠载中断使能** 0：无效 1：使能中断
RXOVIE	Bit 7	W1	**接收 FIFO 溢出中断使能** 0：无效 1：使能中断
RXFIE	Bit 6	W1	**接收 FIFO 满中断使能** 0：无效 1：使能中断
Reserved	Bit 5	—	**保留**
TXTHIE	Bit 4	W1	**发送 FIFO 低于阈值中断使能** 0：无效 1：使能中断
TXUDIE	Bit 3	W1	**发送 FIFO 欠载中断使能** 0：无效 1：使能中断
TXOVIE	Bit 2	W1	**发送 FIFO 溢出中断使能** 0：无效 1：使能中断
Reserved	Bit 1	—	**保留**
TXEIE	Bit 0	W1	**发送 FIFO 空中断使能** 0：无效 1：使能中断

25. 5. 2. 13 I2C中断禁止寄存器（I2Cx_IDR）

I2C 中断禁止寄存器（I2Cx_IDR）

偏移地址：30_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											ALERTID	TOUTID	PECEID	ARLOID	BERRID	Reserved	STOPID	NACKID	ADDRID	TCRID	TCID	RXTHID	RXUDID	RXOVID	RXFID	Reserved	TXTHID	TXUDID	TXOVID	Reserved	TXEID

Reserved	Bit 31-21	—	保留
ALERTID	Bit 20	W1	**SMBus 警报中断禁止** 0：无效 1：禁止中断
TOUTID	Bit 19	W1	**超时中断禁止** 0：无效 1：禁止中断
PECEID	Bit 18	W1	**PEC 错误中断禁止** 0：无效 1：禁止中断
ARLOID	Bit 17	W1	**仲裁丢失中断禁止** 0：无效 1：禁止中断
BERRID	Bit 16	W1	**总线错误中断禁止** 0：无效 1：禁止中断
Reserved	Bit 15	—	保留
STOPID	Bit 14	W1	**停止检测中断禁止** 0：无效 1：禁止中断
NACKID	Bit 13	W1	**NACK 接收中断禁止** 0：无效 1：禁止中断
ADDRID	Bit 12	W1	**地址匹配中断禁止** 0：无效 1：禁止中断
TCRID	Bit 11	W1	**传输完成并重新加载中断禁止** 0：无效 1：禁止中断
TCID	Bit 10	W1	**传输完成中断禁止** 0：无效 1：禁止中断
RXTHID	Bit 9	W1	**接收 FIFO 超过阈值中断禁止**

			0：无效 1：禁止中断
RXUDID	Bit 8	W1	**接收 FIFO 欠载中断禁止** 0：无效 1：禁止中断
RXOVID	Bit 7	W1	**接收 FIFO 溢出中断禁止** 0：无效 1：禁止中断
RXFID	Bit 6	W1	**接收 FIFO 满中断禁止** 0：无效 1：禁止中断
Reserved	Bit 5	—	**保留**
TXTHID	Bit 4	W1	**发送 FIFO 低于阈值中断禁止** 0：无效 1：禁止中断
TXUDID	Bit 3	W1	**发送 FIFO 欠载中断禁止** 0：无效 1：禁止中断
TXOVID	Bit 2	W1	**发送 FIFO 溢出中断禁止** 0：无效 1：禁止中断
Reserved	Bit 1	—	**保留**
TXEID	Bit 0	W1	**发送 FIFO 空中断禁止** 0：无效 1：禁止中断

25. 5. 2. 14 I2C中断有效状态寄存器（I2Cx_IVS）

I2C 中断有效状态寄存器（I2Cx_IVS）

偏移地址：34_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											ALERTIV	TOUTIV	PECEIV	ARLOIV	BERRIV	Reserved	STOPIV	NACKIV	ADDRIV	TCRIV	TCIV	RXTHIV	RXUDIV	RXOVIV	RXFIV	Reserved	TXTHIV	TXUDIV	TXOVIV	Reserved	TXEIV

Reserved	Bit 31-21	—	保留
ALERTIV	Bit 20	R	**SMBus 警报中断有效** 0：无效 1：中断有效
TOUTIV	Bit 19	R	**超时中断有效** 0：无效 1：中断有效
PECEIV	Bit 18	R	**PEC 错误中断有效** 0：无效 1：中断有效
ARLOIV	Bit 17	R	**仲裁丢失中断有效** 0：无效 1：中断有效
BERRIV	Bit 16	R	**总线错误中断有效** 0：无效 1：中断有效
Reserved	Bit 15	—	保留
STOPIV	Bit 14	R	**停止检测中断有效** 0：无效 1：中断有效
NACKIV	Bit 13	R	**NACK 接收中断有效** 0：无效 1：中断有效
ADDRIV	Bit 12	R	**地址匹配中断有效** 0：无效 1：中断有效
TCRIV	Bit 11	R	**传输完成并重新加载中断有效** 0：无效 1：中断有效
TCIV	Bit 10	R	**传输完成中断有效** 0：无效 1：中断有效

RXTHIV	Bit 9	R	**接收 FIFO 超过阈值中断有效** 0：无效 1：中断有效
RXUDIV	Bit 8	R	**接收 FIFO 欠载中断有效** 0：无效 1：中断有效
RXOVIV	Bit 7	R	**接收 FIFO 溢出中断有效** 0：无效 1：中断有效
RXFIV	Bit 6	R	**接收 FIFO 满中断有效** 0：无效 1：中断有效
Reserved	Bit 5	—	**保留**
TXTHIV	Bit 4	R	**发送 FIFO 低于阈值中断有效** 0：无效 1：中断有效
TXUDIV	Bit 3	R	**发送 FIFO 欠载中断有效** 0：无效 1：中断有效
TXOVIV	Bit 2	R	**发送 FIFO 溢出中断有效** 0：无效 1：中断有效
Reserved	Bit 1	—	**保留**
TXEIV	Bit 0	R	**发送 FIFO 空中断有效** 0：无效 1：中断有效

25. 5. 2. 15 I2C原始中断标志寄存器（I2Cx_RIF）

I2C 原始中断标志寄存器（I2Cx_RIF）

偏移地址：38_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											ALERTRI	TOUTRI	PECERI	ARLORI	BERRRI	Reserved	STOPRI	NACKRI	ADDRRI	TCRRI	TCRI	RXTHRI	RXUDRI	RXOVRI	RXFRI	Reserved	TXTHRI	TXUDRI	TXOVRI	Reserved	TXERI

Reserved	Bit 31-21	—	保留
ALERTRI	Bit 20	R	**SMBus 警报中断标志状态** 0：无效 1：原始中断标志状态
TOUTRI	Bit 19	R	**超时中断标志状态** 0：无效 1：原始中断标志状态
PECERI	Bit 18	R	**PEC 错误中断标志状态** 0：无效 1：原始中断标志状态
ARLORI	Bit 17	R	**仲裁丢失中断标志状态** 0：无效 1：原始中断标志状态
BERRRI	Bit 16	R	**总线错误中断标志状态** 0：无效 1：原始中断标志状态
Reserved	Bit 15	—	保留
STOPRI	Bit 14	R	**停止检测中断标志状态** 0：无效 1：原始中断标志状态
NACKRI	Bit 13	R	**NACK 接收中断标志状态** 0：无效 1：原始中断标志状态
ADDRRI	Bit 12	R	**地址匹配中断标志状态** 0：无效 1：原始中断标志状态
TCRRI	Bit 11	R	**传输完成并重新加载中断标志状态** 0：无效 1：原始中断标志状态
TCRI	Bit 10	R	**传输完成中断标志状态** 0：无效 1：原始中断标志状态

RXTHRI	Bit 9	R	**接收 FIFO 超过阈值中断标志状态** 0：无效 1：原始中断标志状态
RXUDRI	Bit 8	R	**接收 FIFO 欠载中断标志状态** 0：无效 1：原始中断标志状态
RXOVRI	Bit 7	R	**接收 FIFO 溢出中断标志状态** 0：无效 1：原始中断标志状态
RXFRI	Bit 6	R	**接收 FIFO 满中断标志状态** 0：无效 1：原始中断标志状态
Reserved	Bit 5	—	**保留**
TXTHRI	Bit 4	R	**发送 FIFO 低于阈值中断标志状态** 0：无效 1：原始中断标志状态
TXUDRI	Bit 3	R	**发送 FIFO 欠载中断标志状态** 0：无效 1：原始中断标志状态
TXOVRI	Bit 2	R	**发送 FIFO 溢出中断标志状态** 0：无效 1：原始中断标志状态
Reserved	Bit 1	—	**保留**
TXERI	Bit 0	R	**发送 FIFO 空中断标志状态** 0：无效 1：原始中断标志状态

25. 5. 2. 16 I2C中断标志屏蔽寄存器（I2Cx_IFM）

I2C 中断标志屏蔽寄存器（I2Cx_IFM）

偏移地址：$3C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											ALERTFM	TOUTFM	PECEFM	ARLOFM	BERRFM	Reserved	STOPFM	NACKFM	ADDRFM	TCRFM	TCFM	RXTHFM	RXUDFM	RXOVFM	RXFFM	Reserved	TXTHFM	TXUDFM	TXOVFM	Reserved	TXEFM

Reserved	Bit 31-21	—	保留
ALERTFM	Bit 20	R	**SMBus 报警中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
TOUTFM	Bit 19	R	**超时中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
PECEFM	Bit 18	R	**PEC 错误中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
ARLOFM	Bit 17	R	**仲裁丢失中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
BERRFM	Bit 16	R	**总线错误中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
Reserved	Bit 15	—	保留
STOPFM	Bit 14	R	**停止检测中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
NACKFM	Bit 13	R	**NACK 接收中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
ADDRFM	Bit 12	R	**地址匹配中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
TCRFM	Bit 11	R	**传输完成并重新加载中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
TCFM	Bit 10	R	**传输完成中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态

RXTHFM	Bit 9	R	**接收 FIFO 超过阈值中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
RXUDFM	Bit 8	R	**接收 FIFO 欠载中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
RXOVFM	Bit 7	R	**接收 FIFO 溢出中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
RXFFM	Bit 6	R	**接收 FIFO 满中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
Reserved	Bit 5	—	**保留**
TXTHFM	Bit 4	R	**发送 FIFO 低于阈值中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
TXUDFM	Bit 3	R	**发送 FIFO 欠载中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
TXOVFM	Bit 2	R	**发送 FIFO 溢出中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
Reserved	Bit 1	—	**保留**
TXEFM	Bit 0	R	**发送 FIFO 空中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态

25. 5. 2. 17 I2C中断清除寄存器（I2Cx_ICR）

I2C 中断清除寄存器（I2Cx_ICR）																															
偏移地址：40_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											ALERTIC	TOUTIC	PECEIC	ARLOIC	BERRIC	Reserved	STOPIC	NACKIC	ADDRIC	TCRIC	TCIC	RXTHIC	RXUDIC	RXOVIC	RXFIC	Reserved	TXTHIC	TXUDIC	TXOVIC	Reserved	TXEIC	

Reserved	Bit 31-21	—	保留
ALERTIC	Bit 20	C_W1	**SMBus 警报中断清除** 0：无效 1：中断清除
TOUTIC	Bit 19	C_W1	**超时中断清除** 0：无效 1：中断清除
PECEIC	Bit 18	C_W1	**PEC 错误中断清除** 0：无效 1：中断清除
ARLOIC	Bit 17	C_W1	**仲裁丢失中断清除** 0：无效 1：中断清除
BERRIC	Bit 16	C_W1	**总线错误中断清除** 0：无效 1：中断清除
Reserved	Bit 15	—	保留
STOPIC	Bit 14	C_W1	**停止检测中断清除** 0：无效 1：中断清除
NACKIC	Bit 13	C_W1	**NACK 接收中断清除** 0：无效 1：中断清除
ADDRIC	Bit 12	C_W1	**地址匹配中断清除** 0：无效 1：中断清除
TCRIC	Bit 11	C_W1	**传输完成并重新加载中断清除** 0：无效 1：中断清除
TCIC	Bit 10	C_W1	**传输完成中断清除** 0：无效 1：中断清除

RXTHIC	Bit 9	C_W1	**接收 FIFO 超过阈值中断清除** 0：无效 1：中断清除
RXUDIC	Bit 8	C_W1	**接收 FIFO 欠载中断清除** 0：无效 1：中断清除
RXOVIC	Bit 7	C_W1	**接收 FIFO 溢出中断清除** 0：无效 1：中断清除
RXFIC	Bit 6	C_W1	**接收 FIFO 满中断清除** 0：无效 1：中断清除
Reserved	Bit 5	—	**保留**
TXTHIC	Bit 4	C_W1	**发送 FIFO 低于阈值中断清除** 0：无效 1：中断清除
TXUDIC	Bit 3	C_W1	**发送 FIFO 欠载中断清除** 0：无效 1：中断清除
TXOVIC	Bit 2	C_W1	**发送 FIFO 溢出中断清除** 0：无效 1：中断清除
Reserved	Bit 1	—	**保留**
TXEIC	Bit 0	C_W1	**发送 FIFO 空中断清除** 0：无效 1：中断清除

第26章 串行外设接口（SPI）

26. 1 概述

SPI/I2S 接口可用于使用 SPI 协议或 I2S 音频协议来与外部设备通信的情形。SPI 或 I2S 模式可通过软件选择。IP 复位后，默认选择 SPI 模式。

串行外设接口（SPI）可与外部器件进行半双工/全双工的同步串行通信。该接口可配置为主机模式，在这种情况下，它可为外部从机提供通信时钟（SCK）。该接口还能够在多主模式配置下工作。

此接口能运行多种模式，如基于双线的单工同步传输，其中一条线用于双向数据传输，同时可用 CRC 校验提高通信可靠性。

Inter-IC Sound（I2S）协议也是同步串行通信接口。它可以通过半双工通信在主机模式下运行。它可以满足四种不同的音频标准，包括 Philips I2S 标准，MSB 和 LSB 对齐标准以及 PCM 标准。

26. 2 特性

SPI 的主要特点

- 主机或从机模式
- 基于三条线的全双工同步传输
- 基于双线的半双工同步传输（使用双向数据线）
- 基于双线的单工同步传输（使用单向数据线）
- 8 位或 16 位传输帧格式选择
- 支持多主模式功能
- 8 个主机模式波特率预分频器，最高可达 $f_{PCLK}/2$
- 对于主机模式和从机模式都可通过硬件或软件进行 NSS 控制：动态切换主机/从机操作
- 时钟极性和相位可编程
- 数据顺序可编程，如最先移位 MSB 或 LSB
- 具有专用的发送和接收中断标志功能
- 支持 SPI 总线忙状态标志
- 支持标准 SPI 通信协议、兼容 TI SPI 通信协议
- 用于确保可靠通信的硬件 CRC 功能：
  - ◇ 在发送模式下可将 CRC 值作为最后一个字节数据发送
  - ◇ 根据收到的最后一个字节自动进行 CRC 错误校验
- 提供发送和接收 FIFO，深度为 16
- 支持 DMA 传输

I2S 的主要特点

- 半双工通信（仅发送器或接收器）
- 仅支持主机模式
- 8 位可编程线性预分频器，可达到精确的音频采样频率（从 8kHz 到 96kHz）
- 数据格式可以是 16 位，24 位或 32 位
- 通过音频通道将通道帧固定为 16 位（16 位数据帧）或 32 位（16 位，24 位，32 位数据帧）
- 可编程时钟极性（稳态）
- 提供接收与发送的下溢与溢出标志
- 16 位寄存器，用于发送和接收
- 支持的 I2S 协议：
  - I2S 飞利浦标准
  - MSB 对齐标准（左对齐）
  - LSB 对齐标准（右对齐）
  - PCM 标准(16 位通道帧上的短帧和长帧同步或扩展到 32 位通道帧的 16 位数据帧)
- 数据方向始终为 MSB 优先
- 发送和接收的 DMA 功能（16 位宽）
- 可输出主时钟以驱动外部音频组件。比率固定为 256×FS（FS 为音频采样频率）
- 提供发送和接收 FIFO，深度为 16
- 提供两个可选的外部时钟输入引脚

26. 3 SPI结构框图

SPI 允许 MCU 与外部设备之间进行同步串行通信。应用软件可以通过轮询状态标志或使用专用 SPI 中断来管理通信。SPI 的主要模块及其相互关系如下面的框图所示。

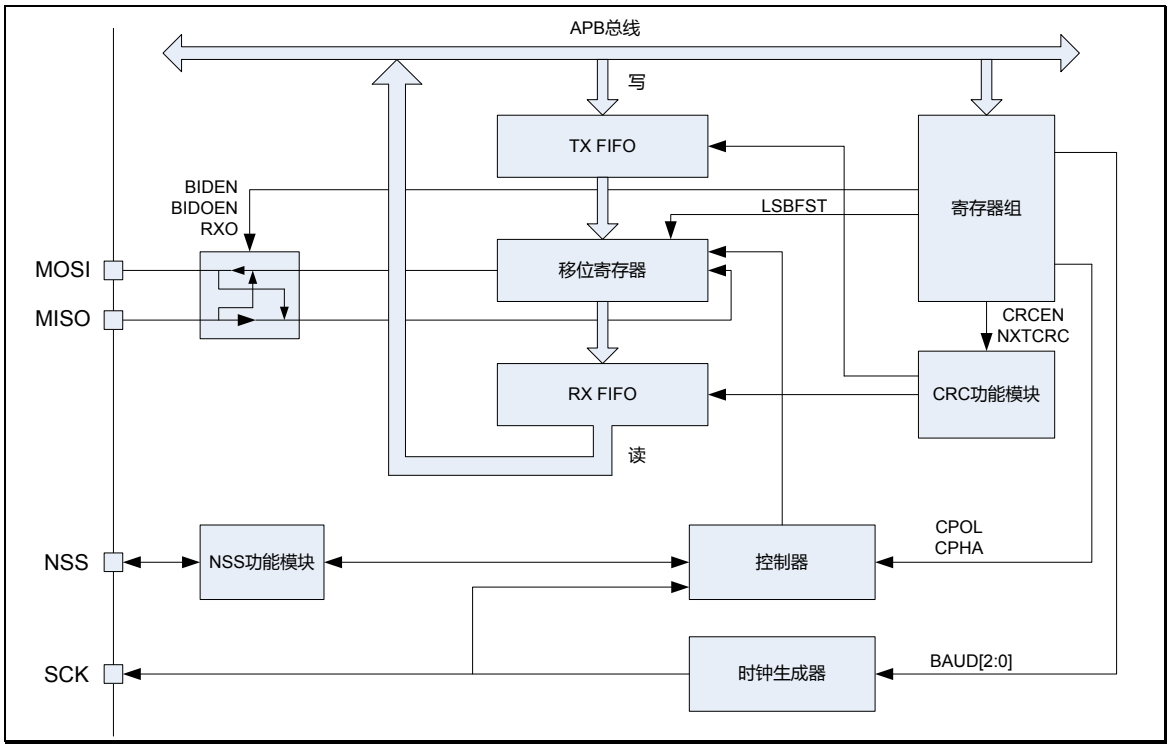


图 26-1　SPI 电路结构框图

通常，SPI 使用四个 I/O 引脚来与外部设备进行通信。

- ◇ MISO：主机输入/从机输出数据
- ◇ MOSI：主机输出/从机输入数据
- ◇ SCK：用于 SPI 主机的串行时钟输出以及 SPI 从机的串行时钟输入。
- ◇ NSS：从机选择引脚。这是用于选择从机的可选引脚。此引脚用作“片选”，可让 SPI 主机与从机进行单独通信，从而避免数据线上的竞争。从机的 NSS 输入可由主机上的标准 IO 端口驱动。NSS 引脚在使能（SPI_CON2.NSSOE 位）时还可用作输出，并可在 SPI 处于主机模式配置时驱动为低电平。通过这种方式，只要器件配置成 NSS 硬件控制模式，所有连接到该主机 NSS 引脚的其它器件 NSS 引脚都将呈现低电平，并因此而作为从机。当配置为主机模式，且 NSS 配置为输入（SPI_CON1.MSTREN=1 且 SPI_CON2.NSSOE=0）时，如果 NSS 拉至低电平，SPI 将进入模式错误状态：SPI_CON1.MSTREN 位自动清零，并且器件配置为从机模式（参见章节 SPI 错误标志）

26. 4 SPI功能描述

26. 4. 1 通信模式

在 SPI 通信期间，接收和发送操作同时执行。串行时钟（SCK）同步数据线上信息的移位和采样。通信格式取决于时钟相位，时钟极性和数据帧格式。为了能够同时通信，主机和从机必须遵循相同的通信格式。

注 1：在作为从机通讯时，从机所使用的系统频率需大于总线上的传输频率 16 倍以上才可正常工作。
注 2：在作为从机通讯时，需要在主机发起传输前写入数据。

26. 4. 1. 1 时钟相位和极性控制

通过配置 SPI_CON1.CPOL 和 SPI_CON1.CPHA 位，可以用软件选择四种可能的时序关系。SPI_CON1.CPOL（时钟极性）位控制空闲时时钟线上的电平状态，此位对主机和从机都有作用。如果复位 SPI_CON1.CPOL 位，SCK 引脚在空闲状态时处于低电平。如果将 SPI_CON1.CPOL 位置 1，SCK 引脚在空闲状态时处于高电平。

如果将 SPI_CON1.CPHA 位置 1，则 SCK 引脚上的第二个边沿（如果 SPI_CON1.CPOL 位配置为 0，则为下降沿；如果 SPI_CON1.CPOL 位配置为 1，则为上升沿）对 MSB 采样。即，在第二个时钟边沿锁存数据。如果复位 CPHA 位，则 SCK 引脚上的第一个边沿（如果 SPI_CON1.CPOL 位配置为 0，则为上升沿；如果 SPI_CON1.CPOL 位配置为 1，则为下降沿）对 MSB 采样。即在第一个时钟边沿锁存数据。

用户通过组合 SPI_CON1.CPOL 和 SPI_CON1.CPHA 位来选择数据捕获的时钟边沿。

下图显示了在 SPI_CON1.CPHA 和 SPI_CON1.CPOL 位的四种组合下的 SPI 传输。可以将该图解释为主机或从机时序图，其中 SCK 引脚、MISO 引脚、MOSI 引脚直接连接在主机和从机之间。

注：在切换 SPI_CON1.CPOL/SPI_CON1.CPHA 位之前，必须通过复位 SPI_CON1.SPIEN 位来关闭 SPI。

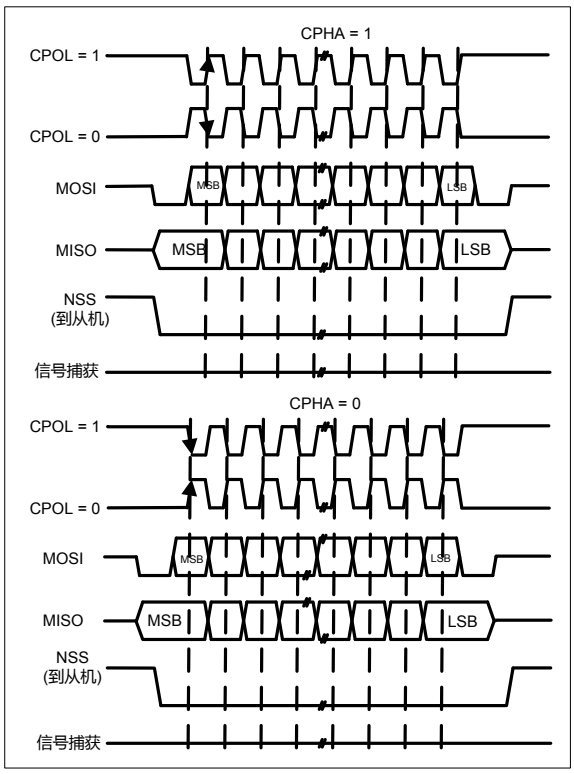


图 26-2 SPI 模式

注：数据位的顺序取决于 LSBFST 位的设置。

26. 4. 1. 2 数据帧格式

SPI 移位寄存器可以设置为移出 MSB 优先或 LSB 优先，具体取决于 SPI_CON1.LSBFST 位的值。每个数据帧为 8 或 16 位长，具体取决于 SPI_CON1.FLEN 位的配置。所选的数据帧长度适用于发送和/或接收。

26. 4. 2 从机选择（NSS）引脚管理

可以使用 SPI_CON1.SSEN 位设置硬件或软件控制从机选择。

◇ 软件控制 NSS（SPI_CON1.SSEN = 1）从机的选择信息在内部由 SPI_CON1.SSOUT 位的值驱动。

◇ 硬件管理 NSS（SPI_CON1.SSEN = 0）根据 NSS 输出配置（SPI_CON2.NSSOE 位），硬件管理 NSS 有两种模式。

- NSS 输出使能（SPI_CON1.SSEN = 0，SPI_CON2.NSSOE = 1）仅当器件在主机模式下工作时才使用此配置。当主机开始传输数据时，NSS 信号驱动为低电平，并保持到数据传输结束为止。
- NSS 输出禁止（SPI_CON1.SSEN = 0，SPI_CON2.NSSOE = 0）对于在主机模式下工作的器件，此配置支持多主模式功能。对于设置为从机模式的器件，NSS 引脚用作传统 NSS 输入：在 NSS 为低电平时片选该从机，在 NSS 为高电平时取消对它的片选。

26. 4. 3 单对单应用

SPI 允许 MCU 使用不同的配置进行通信，具体取决于所针对的设备和应用要求。这些配置使用 2 或 3 线（使用软件 NSS 管理）或者是 3 或 4 线（使用硬件 NSS 管理）。通信始终由主机启动。

26. 4. 3. 1 全双工通信

默认情况下，SPI 配置为全双工通信。在此配置中，MOSI 引脚连接在一起，MISO 引脚连接在一起。通过这种方式，主机和从机之间以串行方式传输数据（最高有效位在前）。

通信始终由主机发起。当主机通过 MOSI 引脚向从机发送数据时，从机同时通过 MISO 引脚发出准备好的数据。这是一个数据输出和数据输入都由同一时钟进行同步的全双工通信过程，时钟信号由主机的 SCK 引脚发出提供给从机。

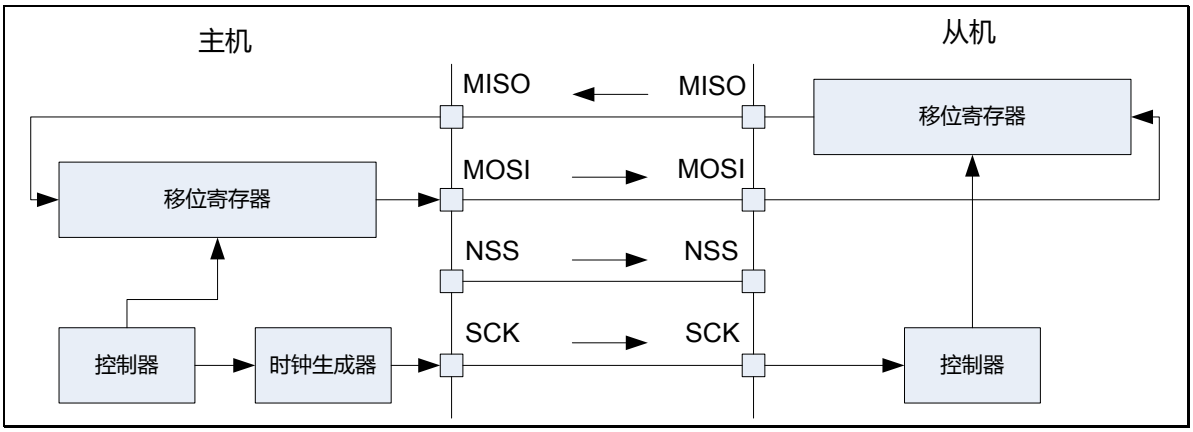


图 26-3 全双工通信

26. 4. 3. 2 半双工通信

SPI 能够在以下两种配置中以半双工模式工作。

1 个时钟和 1 条双向数据线（SPI_CON1.BIDEN=1）

可将 SPI_CON1.BIDEN 位置 1 来使能此模式。在此模式下，SCK 作为时钟信号输出引脚，MOSI（主机模式下）或 MISO（从机模式下）作为数据通信引脚。通过 SPI_CON1.BIDOEN 位来选择传输方向（输入/输出）。当该位置 1 时，数据线为输出，否则为输入。

1 个时钟和 1 条单向数据线（SPI_CON1.BIDEN=0）

在此模式下，应用程序可使用 SPI 的只发送或只接收功能。

◇ 只发送模式类似于全双工模式：在发送引脚（主机模式下的 MOSI 或从机模式下的 MISO）上发送数据，不再接收数据。

◇ 只接收模式下，应用程序可将 SPI_CON1.RXO 位置 1 来关闭 SPI 输出功能。

当 SPI 进入只读模式后：

◇ 一旦在主机模式下使能 SPI 后，主机会等待用户写入数据之后开始从 SCK 引脚发送时钟，需要读回多少个数据由写入无效数据的个数来决定，当 TX FIFO 中的无效的数据发送完毕后，通信也立即停止。

◇ 在从机模式下，只要 NSS 引脚被拉低（或在 NSS 软件模式下将 SPI_CON1.SSOUT 位清零），意味着从机被选中，同时一直有来自主机的 SCK 输入，SPI 就会继续接收。

26. 4. 4 数据发送和接收

接收和发送 FIFO 缓存

所有 SPI 数据传输都通过嵌入式 16 级深度的 FIFO 缓存。使 SPI 能够连续传输工作，并在数据帧长度较短时防止接收溢出。发送和接收都有自己的 FIFO 缓存。

对 SPI_DATA 寄存器的读访问将返回存储在接收 FIFO 缓存中但尚未读取的最旧的值。对 SPI_DATA 的写访问将已写数据存储在发送队列末尾的发送 FIFO 缓存中。SPI_STAT 寄存器中 RXFLV 和 TXFLV 位域指示两个 FIFO 缓存的有效数据个数。

对 SPI_DATA 寄存器的读访问必须由 RXTH 事件管理。当数据存储在接收 FIFO 缓存中并且大于或等于阈值（由 SPI_CON2 寄存器中 RXFTH 位域定义）时，触发此事件。当 RXTH 被清除时，表示接收 FIFO 缓存中的有效数据个数小于阈值。以类似的方式，要发送的数据帧的写访问由 TXTH 事件管理。当发送 FIFO 缓存有效数据个数小于或等于阈值（由 SPI_CON2 寄存器中 TXFTH 位域定义）时将触发此事件。

在主机模式下启动通信序列

◇ 在全双工通信（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0）

- 将数据写入到 SPI_DATA 寄存器（发送 FIFO 缓存）后，启动通信序列。
- 随后在第一个位的发送期间，将数据从发送 FIFO 缓存并行加载到 8 位移位寄存器中，然后以串行方式将其移出到 MOSI 引脚。
- 同时，将 MISO 引脚上接收的数据以串行方式移入 8 位移位寄存器，然后并行加载到 SPI_DATA 寄存器（接收 FIFO 缓存）中。

◇ 在单工通信-只接收模式（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=1）

- 将无效数据写入到 SPI_DATA 寄存器（发送 FIFO 缓存）后，启动通信序列。
- 随后在第一个位的发送期间，将数据从发送 FIFO 缓存并行加载到 8 位移位寄存器中，然后以串行方式将其移出到 MOSI 引脚。
- 同时，将 MISO 引脚上接收的数据以串行方式移入 8 位移位寄存器，然后并行加载到 SPI_DATA 寄存器（接收 FIFO 缓存）中。

◇ 在半双工通信-发送模式（SPI_CON1.BIDEN=1 且 SPI_CON1.BIDOEN=1）

- 将数据写入到 SPI_DATA 寄存器（发送 FIFO 缓存）时，通信序列启动。
- 随后在第一个位的发送期间，将数据从发送缓冲区并行加载到 8 位移位寄存器中，然后以串行方式将其移出到 MOSI 引脚。
- 不接收任何数据。

◇ 在半双工通信-接收模式（SPI_CON1.BIDEN=1 且 SPI_CON1.BIDOEN=0）

- SPI_CON1.SPIEN=1 且 SPI_CON1.BIDOEN=0。
- 将无效数据写入到 SPI_DATA 寄存器（发送 FIFO 缓存）后，启动通信序列。
- 随后在第一个位的发送期间，将数据从发送 FIFO 缓存并行加载到 8 位移位寄存器中，然后以串行方式将其移出到 MOSI 引脚。
- 同时，将 MOSI 引脚上接收的数据以串行方式移入 8 位移位寄存器，然后并行加载到 SPI_DATA 寄存器（接收 FIFO 缓存）中。

在从机模式下启动通信序列

◇ 在全双工模式（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0）

- 当从机收到时钟信号，并在其 MOSI 引脚上收到数据的第一个位时，通信序列开始。其余 7 个位将加载到移位寄存器中。
- 同时，在第一个位的发送期间，将数据从发送缓冲区并行加载到 8 位移位寄存器中，然后以串行方式将其移出到 MISO 引脚。在 SPI 主机启动传输前，软件必须已把要从机发送的数据写入发送 FIFO 缓存。

◇ 在单工通信-只接收模式（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=1）

- 当从机收到时钟信号，并在其 MOSI 引脚上收到数据的第一个位时，通信序列开始。其余 7 个位将加载到移位寄存器中。
- 由于发送器没有激活，因此不会有数据以串行方式移出 MISO 引脚。

◇ 在半双工通信-发送模式（SPI_CON1.BIDEN=1 且 SPI_CON1.BIDOEN=1）

- 当从机收到时钟信号，并且 MISO 引脚上发出发送 FIFO 缓存中的第一位数据时，通信序列开始。
- 随后在第一个位的发送期间，将数据从发送 FIFO 缓存并行加载到 8 位移位寄存器中，然后以串行方式将其移出到 MISO 引脚。在 SPI 主机启动传输前，软件必须已把要从机发送的数据写入发送 FIFO 缓存。
- 不接收任何数据。

◇ 在半双工通信-接收模式（SPI_CON1.BIDEN=1 且 SPI_CON1.BIDOEN=0）

- 当从机收到时钟信号，并在其 MISO 引脚上收到数据的第一个位时，通信序列开始。
- 在 MISO 引脚上接收的数据以串行方式移入 8 位移位寄存器，然后并行加载到 SPI_DATA 寄存器（接收 FIFO 缓存）中。
- 由于发送器没有被激活，因此不会有数据以串行方式移出 MISO 引脚。

处理数据发送与接收

全双工通信（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0），发送和接收数据的处理过程

直接存取操作模式：

1. 通过将 SPI_CON1.SPIEN 位置 1 来使能 SPI，将第一个要发送的数据项写入 SPI_DATA 寄存器（此操作会将 SPI_STAT.TXE 位清零）。
2. 等待 SPI_STAT.TXE=1，然后写入要发送的第二个数据项。然后等待 SPI_STAT.RXE=0，读取 SPI_DATA 以获取第一个接收到的数据（此操作会将 SPI_STAT.RXE 位置 1）。对每个要发送和接收的数据项重复此操作，直到发送并接收完最后的数据。
3. 检查 SPI_STAT.TXE=1，然后等待至 SPI_STAT.BUSY=0，再关闭 SPI。
4. 此外，还可以使用 TXE 或 RXTH（将 SPI_CON2.RXFTH 位域置 1）中断事件对应的各个中断子程序来实现该过程。

FIFO 缓存操作模式：

1. 通过将 SPI_CON1.SPIEN 位置 1 来使能 SPI。
2. 配置 SPI_CON2.TXFTH 与 SPI_CON2.RXFTH。

3. 当 SPI_STAT.TXTH=1，将要发送的数据写入 SPI_DATA 寄存器（写入的数据个数必须大于 SPI_CON2.TXFTH 设定的阀值），当 SPI_STAT.RXTH=1，读取 SPI_DATA 寄存器以获取接收到的数据(读取的数据个数必须为 SPI_CON2.RXFTH 设定的阀值)，重复此操作直到写入最后要发送的数据。
4. 等待至 SPI_STAT.BUSY=0，读取 SPI_DATA 寄存器以获取接收到的数据直到 SPI_STAT.RXFLV 位置 0，再关闭 SPI。
5. 此外，还可以使用 TXTH 或 RXTH 中断事件对应的各个中断子程序来实现该过程。

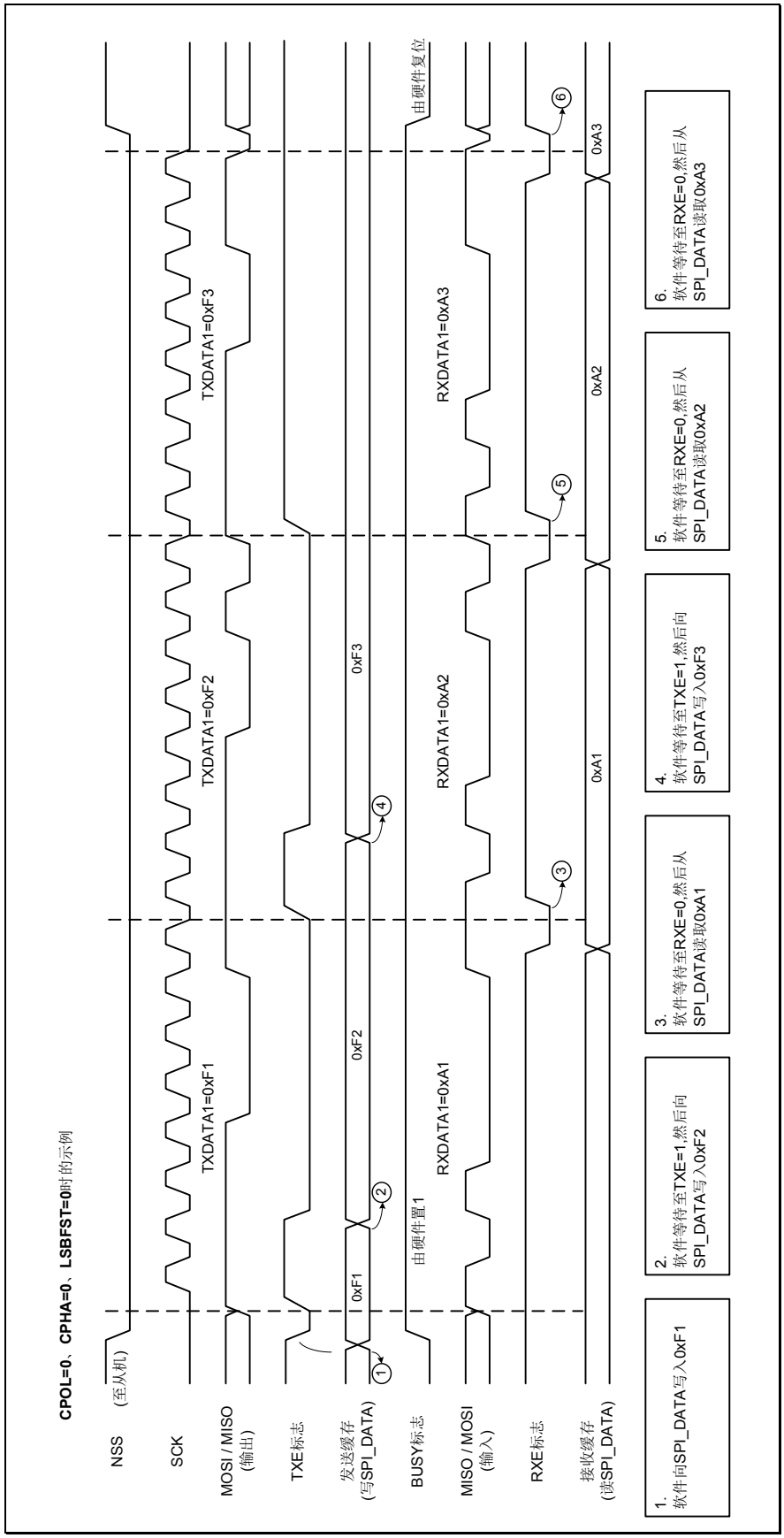


图 26-4　全双工通信（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0）的 TXE、RXE、BUSY 行为

（直接存取操作模式在连续传输的情况下）

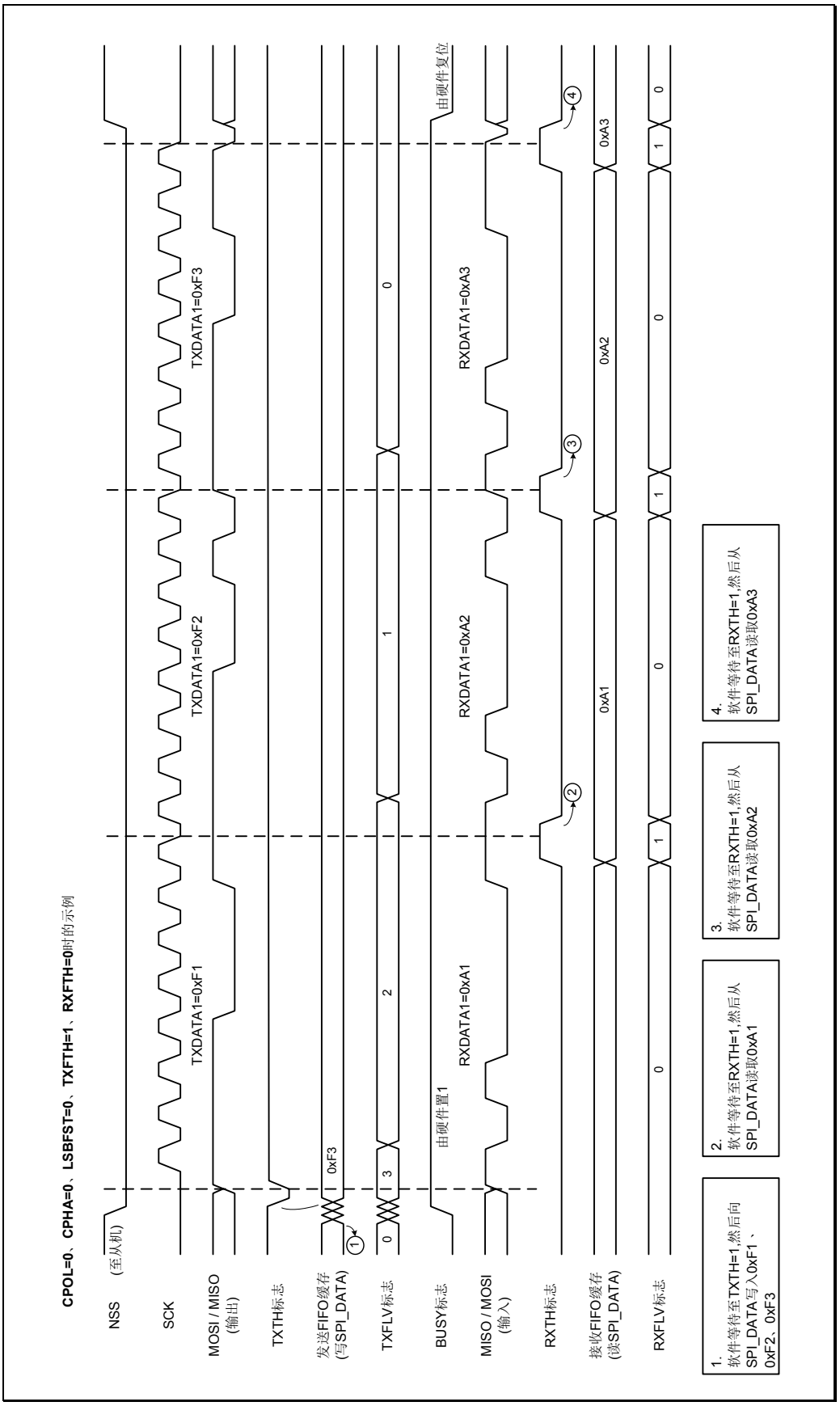


图 26-5　全双工通信（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0）的 TXTH、RXTH、TXFLV、RXFLV、BUSY 行为（FIFO 缓存操作模式在连续传输的情况下）

单工通信-只发送模式（SPI_CON1.BIDEN=0、SPI_CON1.RXO=0），发送数据的处理过程

直接存取操作模式：

1. 通过将 SPI_CON1.SPIEN 位置 1 来使能 SPI。
2. 等待 SPI_STAT.TXE=1 然后写入要发送的数据。对每个要发送的数据项重复此步骤。
3. 将最后一个数据写入 SPI_DATA 寄存器后，等待至 SPI_STAT.TXE=1，然后等待至 SPI_STAT.BUSY=0 再关闭 SPI，这表示最后的数据发送完成。
4. 此外，还可以使用在 TXE 中断事件对应的中断子程序来实现该过程。

FIFO 缓存操作模式：

1. 通过将 SPI_CON1.SPIEN 位置 1 来使能 SPI。
2. 配置 SPI_CON2.TXFTH。
3. 当 SPI_STAT.TXTH=1，将要发送的数据写入 SPI_DATA 寄存器（写入的数据个数必须大于 SPI_CON2.TXFTH 设定的阀值），重复此操作直到写入最后要发送的数据。
4. 等待至 SPI_STAT.BUSY=0 再关闭 SPI。
5. 此外，还可以使用 TXTH 中断事件对应的中断子程序来实现该过程。

注 1：在不连续通信期间，在对 SPI_DATA 寄存器执行写操作与 SPI_STAT.BUSY 位置 1 之间有延迟。因此在只发送模式下，写入最后的数据后，必须先等待 SPI_STAT.TXE 位置 1，然后等待 SPI_STAT.BUSY 位清零。

注 2：在只发送模式下，发送 17 个数据项后，SPI_STAT.RXOV 标志将置 1，因为始终不会读取接收的数据。

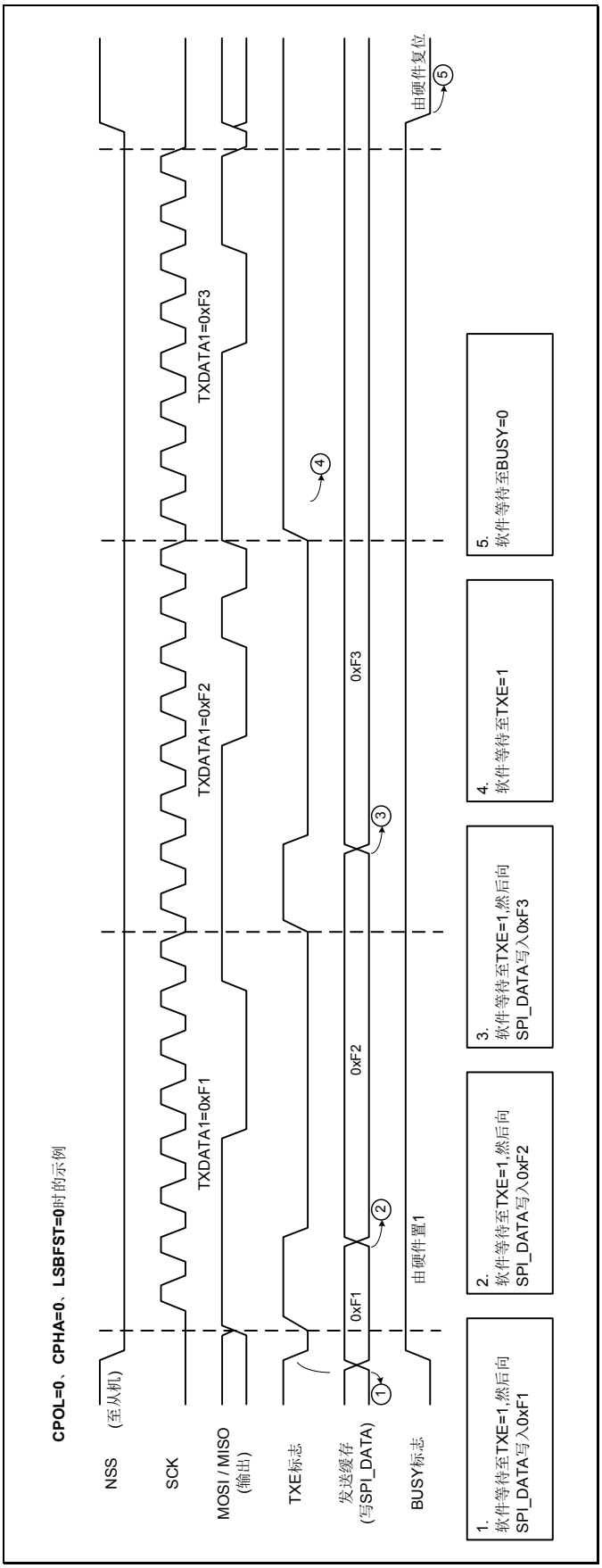


图 26-6 单工通信-只发送模式（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0）的 TXE、BUSY

行为（直接存取操作模式在连续传输的情况下）

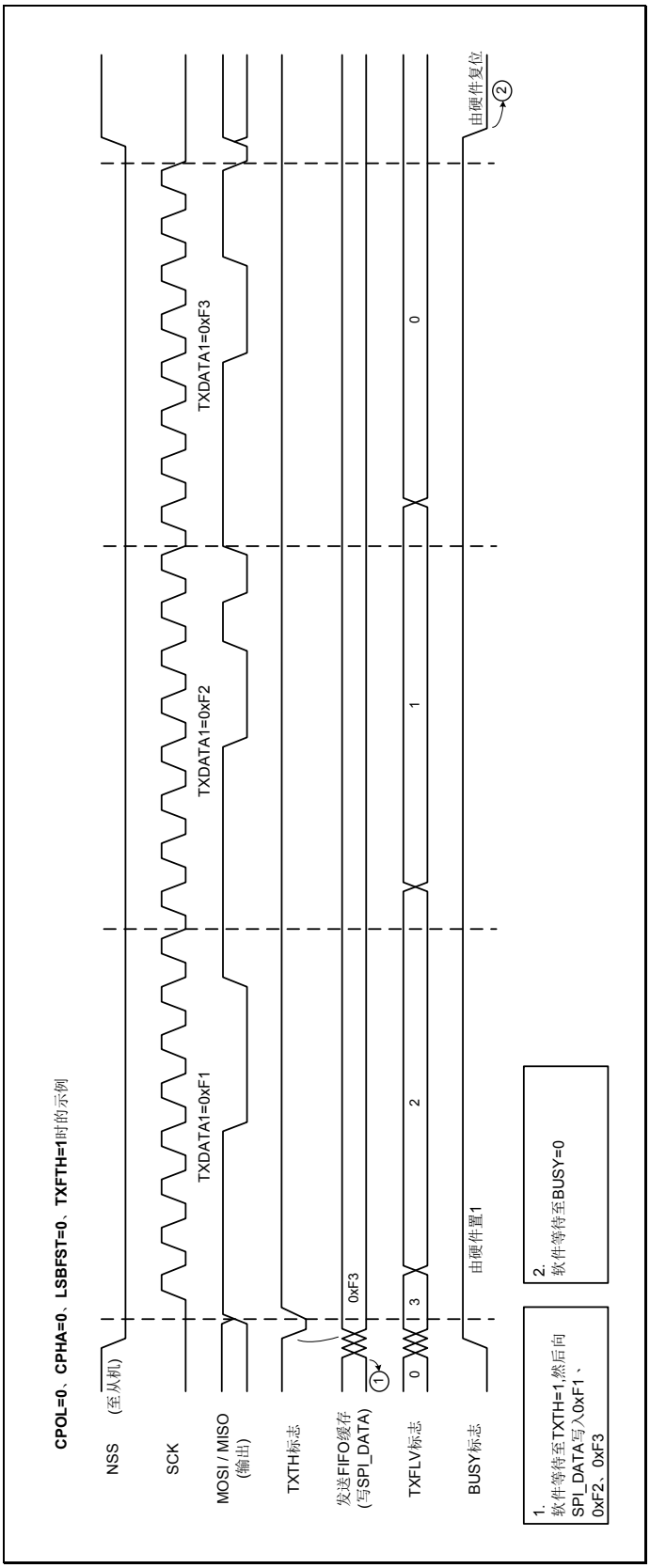


图 26-7　单工通信-只发送模式（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0）的 TXTH、TXFLV、BUSY 行为（FIFO 缓存操作模式在连续传输的情况下）

半双工通信-发送模式（SPI_CON1.BIDEN=1 且 SPI_CON1.BIDOEN=1），发送数据的处理过程

此模式与单工通信-只发送模式数据的处理过相似，但是在 SPI 模块使能前，必须将 SPI_CON1.BIDEN 位和 SPI_CON1.BIDOEN 位置 1。

单工通信-只接收模式（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=1），接收数据的处理过程

直接存取操作模式：

1. 将 SPI_CON1.RXO 位置 1。
2. 通过将 SPI_CON1.SPIEN 位置 1 使能 SPI。
3. 在主机模式下。等待 SPI_STAT.TXE=1，然后将无效数据写入到 SPI_DATA 寄存器。等待 SPI_STAT.RXE=0，然后读取 SPI_DATA 寄存器以获取接收的数据（此操作会将 SPI_STAT.RXE 位置 1）。对每个要接收的数据项重复此操作。
4. 在从机模式下。等待 SPI_STAT.RXE=0，然后读取 SPI_DATA 寄存器以获取接收的数据（此操作会将 SPI_STAT.RXE 位置 1）。对每个要接收的数据项重复此操作。
5. 此外，还可以使用 RXTH（将 SPI_CON2.RXFTH 位域置 1）中断事件对应的中断子程序来实现该过程。

FIFO 缓存操作模式：

1. 将 SPI_CON1.RXO 位置 1。
2. 配置 SPI_CON2.RXFTH。
3. 通过将 SPI_CON1.SPIEN 位置 1 使能 SPI。
4. 在主机模式下。当 SPI_STAT.TXTH=1，将无效数据写入 SPI_DATA 寄存器（写入的数据个数必须大于 SPI_CON2.TXFTH 设定的阀值），当 SPI_STAT.RXTH=1，读取 SPI_DATA 寄存器以获取接收到的数据（读取的数据个数必须为 SPI_CON2.RXFTH 设定的阀值），重复此操作直到获取最后要接收的数据。
5. 在从机模式下。当 SPI_STAT.RXTH=1，读取 SPI_DATA 寄存器以获取接收到的数据（读取的数据个数必须为 SPI_CON2.RXFTH 设定的阀值），重复此操作直到获取最后要接收的数据。
6. 此外，还可以使用 RXTH 中断事件对应的中断子程序来实现该过程。

注 1：在主机模式下，一旦 SPI 使能后 SCK 会立即发送时钟，从机接收到时钟后会发送数据，直到主机关闭 SPI 功能结束通信。

注 2：在从机模式下，当 NSS 被拉低并且接收到 SCK 时钟后开始接收数据。

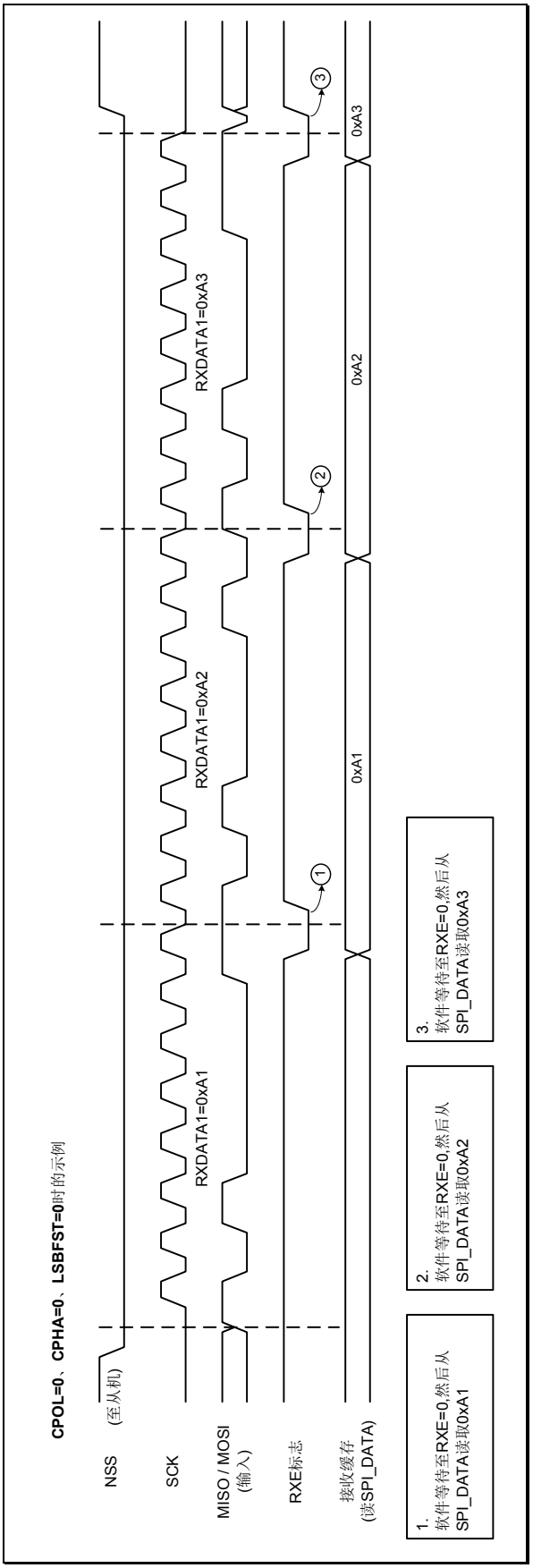


图 26-8　单工通信-只接收模式（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=1）的 RXE 行为（直接存取操作模式在连续传输的情况下）

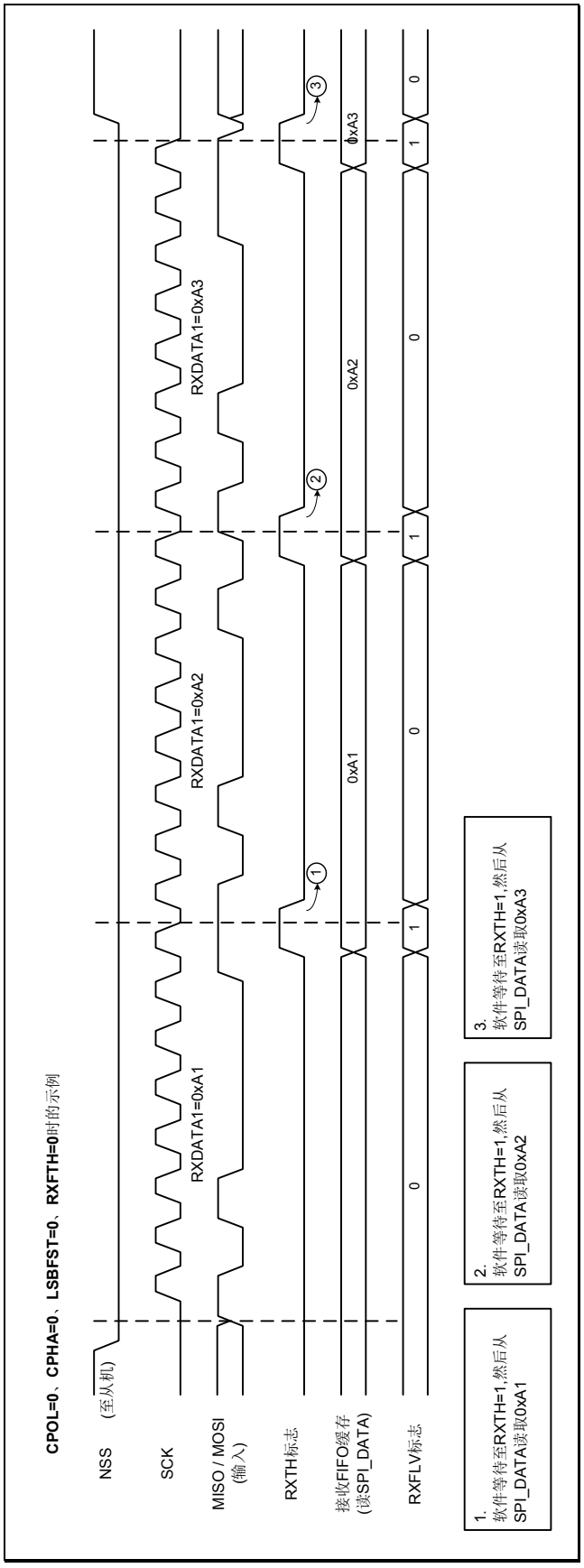


图 26-9　单工通信-只接收模式（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=1）的 RXTH、RXFLV 行为（FIFO 缓存操作模式在连续传输的情况下）

半双工通信-接收模式（SPI_CON1.BIDEN=1 和 SPI_CON1.BIDOEN=0），接收数据的处理过程

此模式与单工通信-只接收模式数据的处理过程相似，但是在 SPI 模块使能之前，需要将 SPI_CON1.BIDEN 位置 1，并将 SPI_CON1.BIDOEN 与 SPI_CON1.RXO 位清 0。

连续传输和间断传输

在主机模式下发送数据时，如果软件处理速度足够快，可以在检测到 SPI_STAT.TXE=1（或发生 TXE 中断事件），并且当前数据传输未结束，立即将下一次的数据写入 SPI_DATA 寄存器，则能实现连续的通信。或者配置 SPI_CON2.TXFTH 检测当 SPI_STAT.TXTH=1（或发生 TXTH 中断事件），将要发送的数据写入 SPI_DATA 寄存器（写入的数据个数必须大于 SPI_CON2.TXFTH 设定的阀值），实现连续的通信。观察到的现象是 SPI_STAT.BUSY 位一直为 1 不被清除，并且每个数据的 SPI 时钟保持连续。

相反，如果软件速度不够快，则可能导致通信中断。在这种情况下，各数据传输之间会清零 SPI_STAT.BUSY 位。

在主机或从机模式下的单工通信-只接收模式（SPI_CON1.RXO=1），通信始终是连续的，且 SPI_STAT.BUSY 位始终为 1。

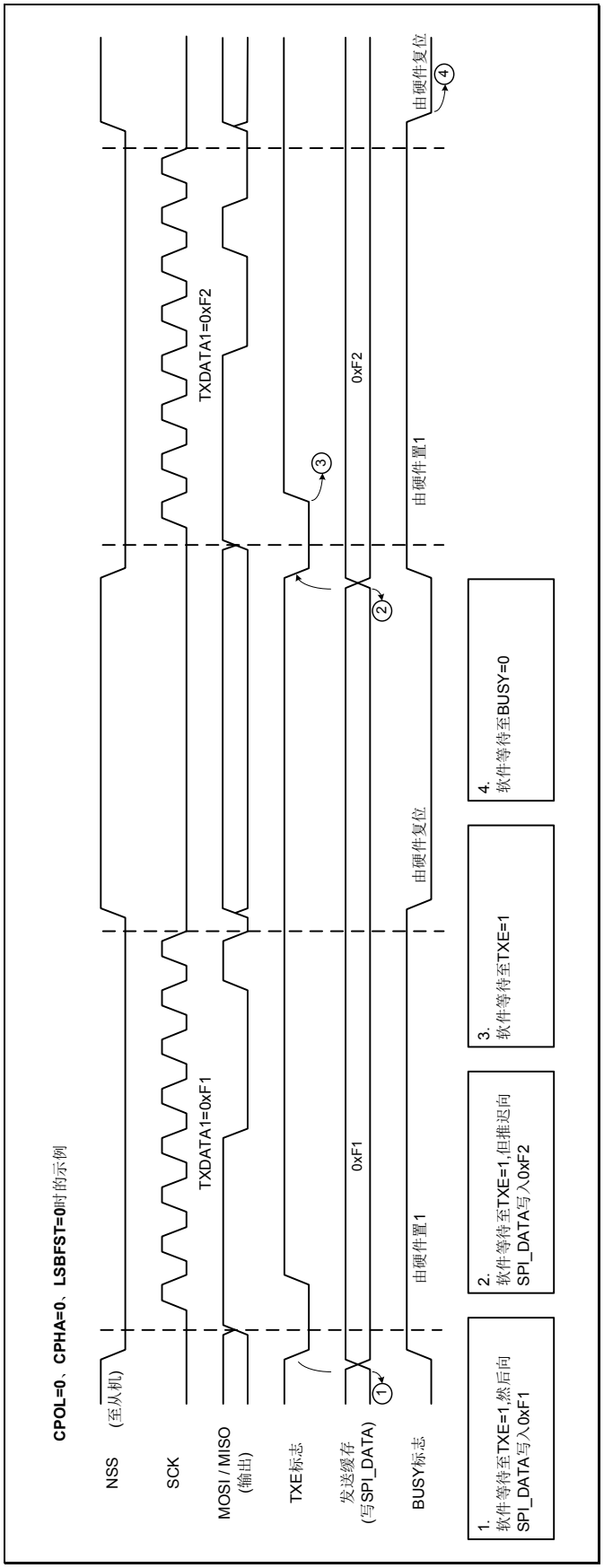


图 26-10　发送时（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0）的 TXE/BUSY 行为（在间断传输的情况下）

26. 4. 5 DMA请求

为了更方便的实现高速通信，SPI 提供了 DMA 功能。DMA 请求条件是根据 SPI_CON2 寄存器中的 TXFTH 与 RXFTH 位域配置，当使能 SPI_CON2 寄存器中相应的 DMA 使能位时，将请求 DMA 访问。发送 FIFO 缓存和接收 FIFO 缓存会发出各自的 DMA 请求（参见以下两张图）：

- ◇ 在发送过程中，当 SPI_STAT.TXTH 位置 1 时会发出 DMA 请求。DMA 随后对 SPI_DATA 寄存器执行写操作（此操作会将 SPI_STAT.TXTH 位清零）。
- ◇ 在接收过程中，当 SPI_STAT.RXTH 位置 1 时会发出 DMA 请求。DMA 随后对 SPI_DATA 寄存器执行读操作（此操作会将 SPI_STAT.RXTH 位清零）。

当 SPI 仅用于发送数据时，可以只使能 SPI TXDMA 通道。在这种情况下，SPI_STAT.RXOV 位会置 1，因为未读取接收的数据。

在发送模式下，DMA 完成了所有要发送数据的传输后，会产生相对应通道的传输完成状态标志，使用者可以对 BUSY 标志进行监视，以确保 SPI 通信已完成。在关闭 SPI 或进入停止模式前必须等待 SPI_STAT.BUSY=0。

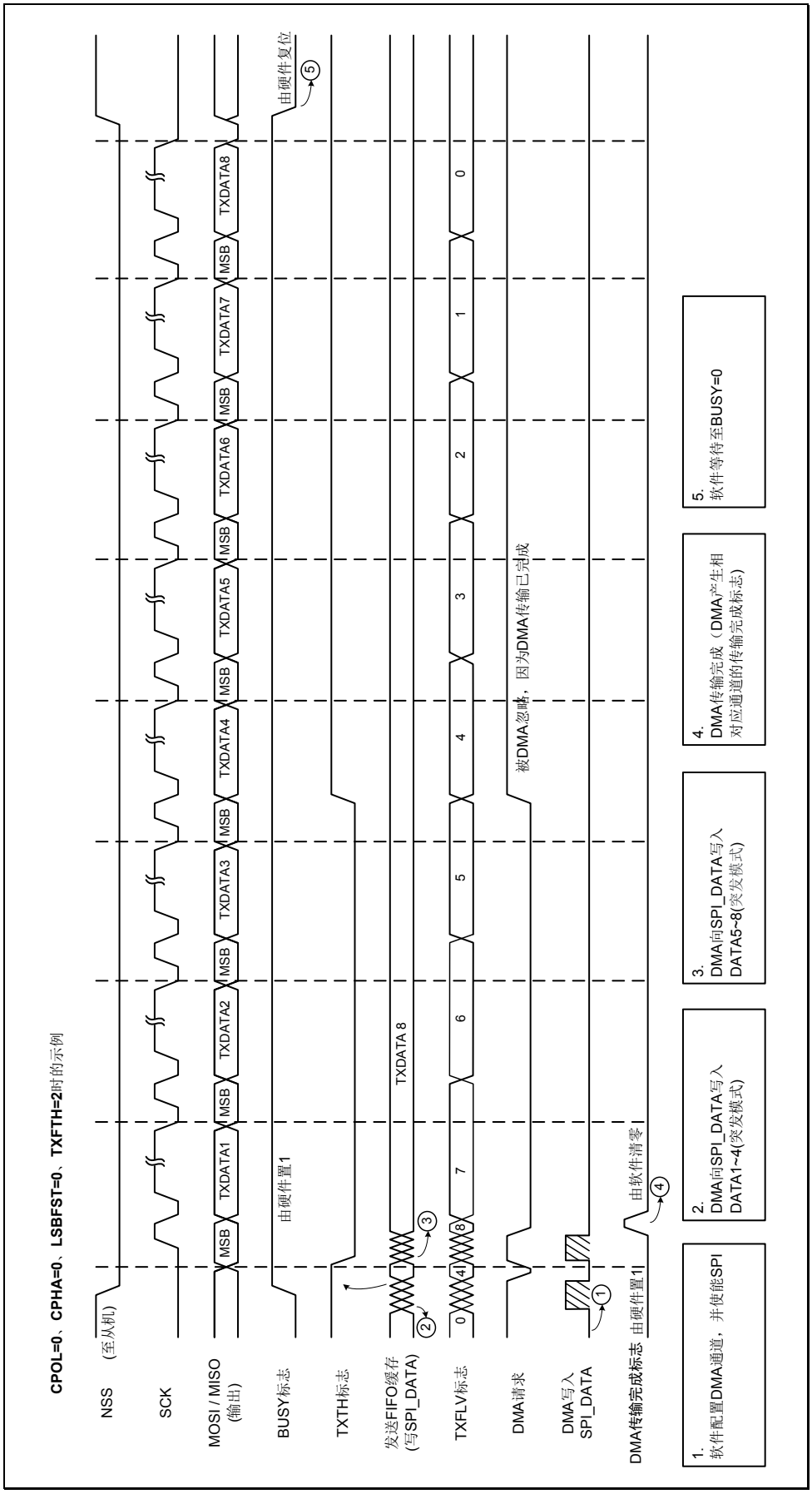


图 26-11 使用 DMA 进行发送

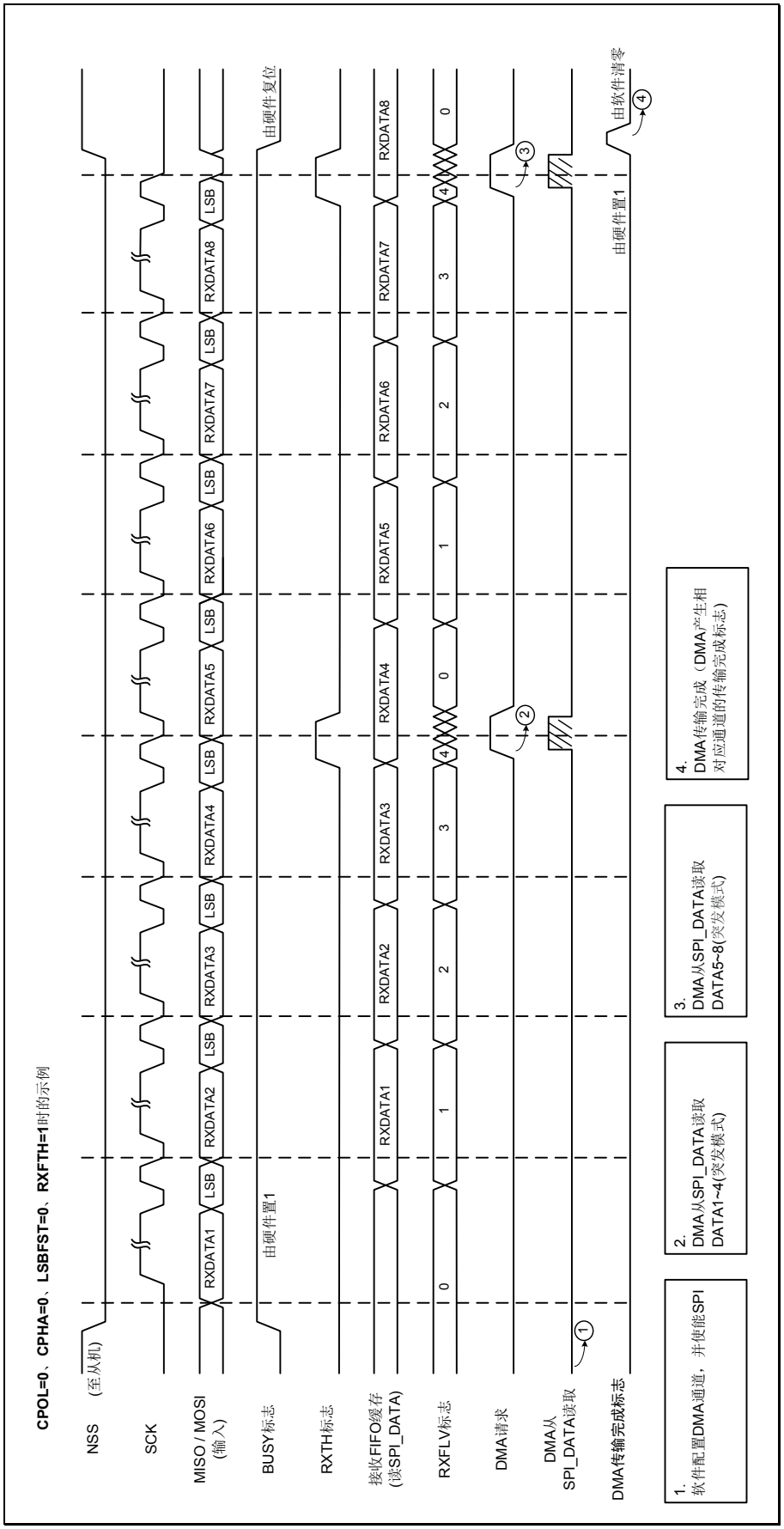


图 26-12　使用 DMA 进行接收

26. 4. 6 SPI兼容模式

该模式 SPI 接口与 TI SPI 通信协议兼容。SPI_CON2 寄存器的 FRF 位可用于配置 SPI 以符合此协议。

无论 SPI_CON1 寄存器中设置的值如何，时钟极性和相位都必须符合 TI 协议要求。NSS 管理也特定于 TI 协议，在这种情况下，通过 SPI_CON1 和 SPI_CON2 寄存器（SSEN，SSOUT，NSSOE）无法配置 NSS 管理。

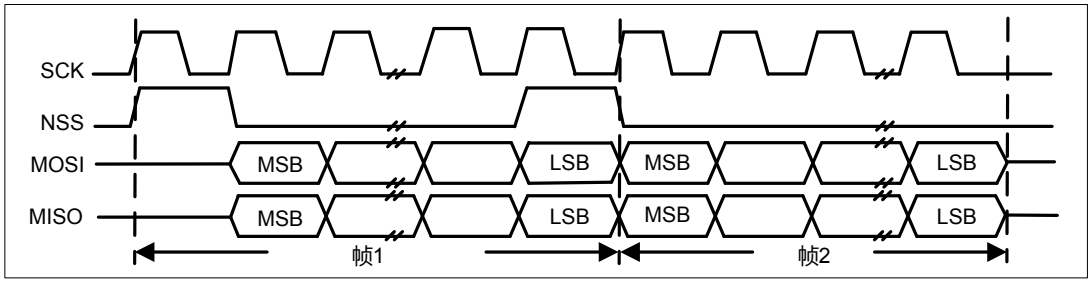


图 26-13 SPI 兼容模式通信波形

26. 4. 7 CRC计算

26. 4. 7. 1 CRC功能描述

为确保通信的可靠性，SPI 模块实现了硬件 CRC 功能。针对发送或接收的数据帧宽度有 8 位和 16 位的选择，硬件 CRC 计算也提供了两种计算标准，分别为 8 位数据的 CRC8 和 16 位数据的 CRC16，具体取决于通过 FLEN 位选择的数据格式。使用 SPI_CRCPOLY 寄存器中编程的多项式串行计算 CRC。目前 CRC 功能不支持 BAUD 设置为 0 与 1 的情况。

通过 SPI_CON1.CRCEN 位置 1 来使能 CRC 计算功能，此操作会复位 CRC 寄存器（SPI_RXCRC 和 SPI_TXCRC）。使用每个位上的奇数可编程多项式计算 CRC 值。计算在 SPI_CON1 寄存器中 CPHA 和 CPOL 位定义的采样时钟边沿处理。在数据块的末尾自动检查计算的 CRC 值以及由 CPU 管理的传输。当在接收数据内部计算的 CRC 与发送器发送的 CRC 之间检测到不匹配时，设置 CRCERRRI 位以指示数据损坏错误。处理 CRC 计算的正确过程取决于 SPI 配置和所选的传输管理。

注：多项式值应该只是奇数。不支持偶数值。

26.4.7.2 CRC传输管理

在此可分为 2 种方式来传送 CRC 帧：

◇ 跟随数据之后传输 CRC 帧

可在写入数据后设置 SPI_CON1.NXTCRC 位，以指示在当前处理的数据帧传输完成之后将跟随 CRC 帧传输。在传输 CRC 帧时，CRC 会停止计算，并同时检测接收数据的 CRC 是否正确。当传输完全结束后，可在 SPI_TXCRC 和 SPI_RXCRC 寄存器中确认发送与接收的 CRC 值。

◇ 单独传输 CRC 帧

若数据帧已传输结束，此时需要单独发送 CRC 帧时，可直接设置 SPI_CON1.NXTCRC 位，此时会直接发送已计算好的 CRC 数值，并同时检测接收数据的 CRC 是否正确。

注：接收的 CRC 帧会如同一般数据一样存入接收 FIFO 中。软件必须检查 SPI_RIF 寄存器中的 CRCERRRI 位，以确定数据传输是否已损坏。软件通过向 SPI_ICR 写入'1'来清除 CRCERRRI 位。

26.4.7.3 复位CRC方式

启用 CRC 计算时，SPI_TXCRC 和 SPI_RXCRC 值将自动清零。

在从机禁用（NSS 上的高电平）和新的从机使能（NSS 上的低电平）之间，应在主机和从机侧清除 CRC 值，以重新同步主机和从机各自的 CRC 计算。

要清除 CRC，请遵循以下顺序：

1. 禁用 SPI
2. 清零 CRCEN 位
3. 启用 CRCEN 位
4. 启用 SPI

26.4.8 SPI状态标志

为应用程序提供了五个状态标志，以完全监视 SPI 总线的状态。

26.4.8.1 发送FIFO空标志（TXE）

设置时，TXE 标志指示发送 FIFO 为空，并且下一个要发送的数据可以加载到 FIFO 中。通过写 SPI_DATA 寄存器清零 TXE 标志。

26.4.8.2 发送FIFO满标志（TXF）

设置时，TXF 标志指示发送 FIFO 已满，并且下一个要发送的数据无法加载到 FIFO 中。TXF 标志由要发送的下一个数据清除。

26.4.8.3 接收FIFO空标志（RXE）

设置时，RXE 标志指示接收 FIFO 为空，并且下一个要接收的数据可以加载到 FIFO 中。RXE 标志由下一个数据接收到 FIFO 中清除。

26. 4. 8. 4 接收FIFO满标志（RXF）

设置后，RXF 标志指示接收 FIFO 已满，并且下一个要接收的数据无法加载到 FIFO 中。通过对 SPI_DATA 寄存器的读访问来清除 RXF 标志。

26. 4. 8. 5 通信忙（BUSY）

BUSY 标志用于指示 SPI 通信的状态。此标志由硬件置 1 和清零。

SPI_STAT.BUSY 位置 1 时，表示 SPI 正在通信中。在通信结束前，用户可检测 SPI_STAT.BUSY 位是否为 0，表示通信已结束，此时关闭 SPI 模块，停止通信。

BUSY 标志还可用于避免在多主模式系统中发生写冲突。

在以下情况硬件将清零该标志：

◇ 传输完成时（主机模式下的连续通信除外）

◇ 关闭 SPI 时

◇ 发生模式错误时（SPI_RIF.MODFRI=1）

当通信不连续时，BUSY 标志在各通信之间处于低电平。

当通信连续时：

◇ 在主机模式下，BUSY 标志在所有传输期间均保持高电平

◇ 在从机模式下，BUSY 标志在各传输之间的一个 SPI 时钟周期内变为低电平

注：建议始终使用 TXTH 和 RXTH 标志（而不是 BUSY 标志）来处理数据传输或接收操作。

26.4.9 SPI错误标志

如果设置了以下错误标志之一并通过将该位置 1 启用中断，则会产生 SPI 中断。

26.4.9.1 发送FIFO溢出标志（TXOV）

发生发送 FIFO 溢出的条件是发送 FIFO 已满，但用户正在执行写操作。在这种情况下，新写入的数据不会加载到发送 FIFO 中。通过对 SPI_ICR 寄存器的写访问来清除 TXOV 位，或者在新传输开始时它将自动清零。

26.4.9.2 接收FIFO溢出标志（RXOV）

当主机或从机完成下一个数据帧的接收而接收 FIFO 的前 16 帧的读操作尚未完成时（设置了 RXF 标志的情况），发生接收溢出条件。

在这种情况下，接收 FIFO 的内容不会将接收的新数据存入。SPI_DATA 寄存器的读操作返回先前接收的帧。所有其他在 RXF 标志设置的情况下，接收的新数据都将丢失。

通过对 SPI_ICR 寄存器的写访问来清除 RXOV 位，或者在新的传输开始时它将自动清零。

26.4.9.3 接收FIFO下溢标志（RXUD）

当接收 FIFO 为空时，但用户正在执行读操作，会发生接收下溢条件。在这种情况下，读取操作不会从接收 FIFO 中读到有效的数据。

通过对 SPI_ICR 寄存器的写访问来清除 RXUD 位，或者在新的传输开始时它将自动清零。

26.4.9.4 主机模式故障（MODF）

当主机模式的内部 NSS 信号清零（NSS 硬件模式下的 NSS 引脚拉低或 NSS 软件模式下的 SSOUT 位为 0）时，发生主机模式故障，这会自动将 MODFRI 位置 1。主机模式故障通过以下方式影响 SPI 接口：

◇ MODFRI 位自动置 1，如果 MODFIV 位为 1，则产生 SPI 中断请求。

◇ MSTREN 位清零，强制 SPI 进入从机模式。

通过软件对 SPI_ICR.MODFIC 写 1 来清除 MODFRI 位。

为避免在包含多个 MCU 的系统中发生任何从机冲突，必须在 MODFRI 位清零前将 NSS 引脚拉高。在 MODFRI 位清零之后，MSTREN 位若要恢复到其原始状态需软件重新对其写入。

在从机模式中，MODFRI 位不会被置 1。但发生主机模式故障时，器件会自动切换至从机模式，MODFRI 位被置 1，此时表示系统可能存在多主模式冲突。

26.4.9.5 CRC错误（CRCERR）

该标志用于验证 SPI_CON1 寄存器中的 CRCEN 位置 1 时接收到的值的有效性。如果移位寄存器中接收的值与接收器 SPI_RXCRC 值不匹配，则 SPI_RIF 寄存器中的 CRCERRRI 标志置 1。该标志由软件清除。

26. 4. 9. 6 SPI兼容模式帧格式错误（FRE）

当 SPI 在从机模式下工作并配置为兼容 TI 模式协议时，在正在进行的通信期间发生 NSS 脉冲时，会检测到兼容模式帧格式错误。发生此错误时，将在 SPI_RIF 寄存器中设置 FRERI 标志。否则发生错误时，SPI 不会被禁用，NSS 脉冲会被忽略，SPI 在开始新的传输之前等待下一个 NSS 脉冲。数据可能被破坏，因为错误检测可能导致两个数据字节的丢失。

如果 FREIE 位置 1，则在 NSS 错误检测时产生中断。在这种情况下，应禁用 SPI，因为不再保证数据一致性，并且当再次启用从 SPI 时，主机应重新启动通信。

26. 5 I2S结构图

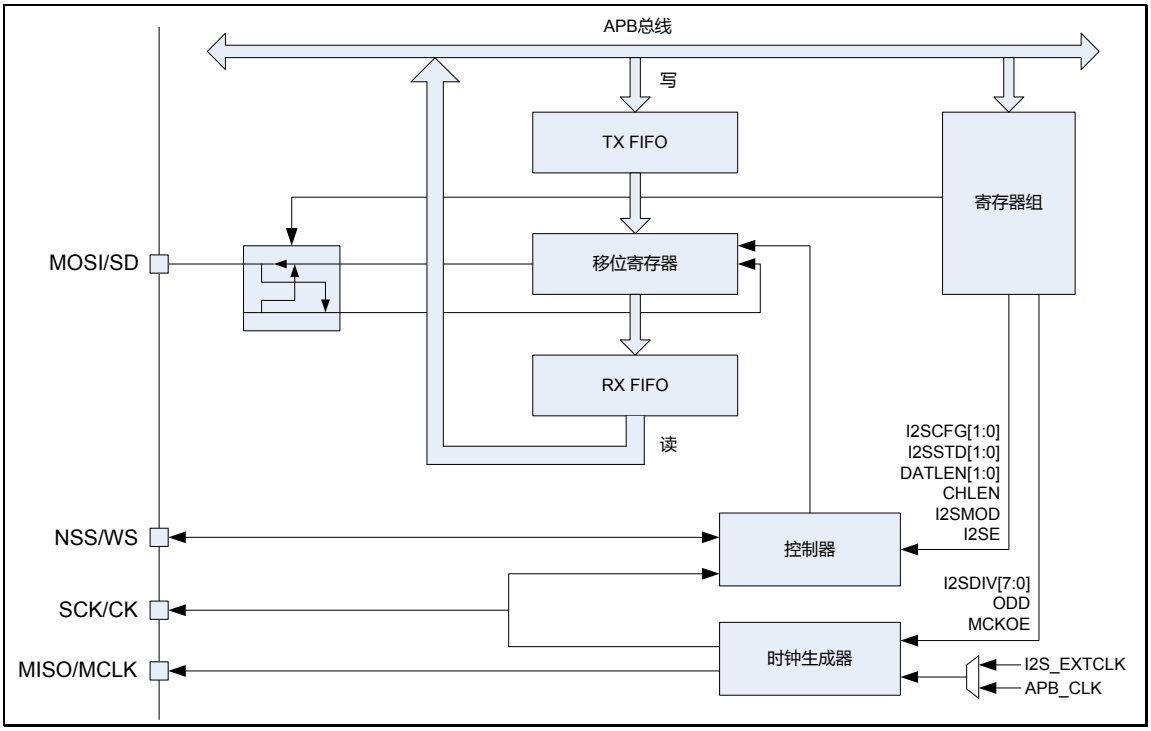


图 26-14 I2S 电路结构框图

当 I2S 功能使能时（通过设置 SPI_I2SCFG.I2SMOD 位），SPI 可用作音频 I2S 接口。该接口主要使用与 SPI 相同的引脚，标志和中断。I2S 与 SPI 共享四个公共引脚：

- ◇ SD：串行数据（映射在 MOSI 引脚上），用于发送或接收两个时间复用数据通道。
- ◇ WS：声道选择（映射在 NSS 引脚上）是主机输出的数据控制信号。
- ◇ CK：串行时钟（映射在 SCK 引脚上）是主机的串行时钟输出。
- ◇ MCLK：当 SPI_I2SPR.MCKOE 位置 1 时，使用主时钟（映射在 MISO 引脚上），输出以预先配置的频率（等于 256×FS）生成的附加时钟，其中 FS 是音频采样频率。

I2S 使用自己的时钟发生器产生通信时钟。该时钟发生器也是主时钟输出的源。在 I2S 模式下有两个额外的寄存器。一个是 I2S 预分频寄存器 SPI_I2SPR，另一个是通用 I2S 配置寄存器 SPI_I2SCFG（音频标准，数据帧，通道帧，时钟极性等）。

在 I2S 模式下不使用 SPI_CON1 寄存器和所有 CRC 寄存器。同样，不使用 SPI_CON2 寄存器中的 NSSOE 位、中断相关寄存器中的 MODF 和 CRCERR 位。

I2S 在 16/32 位宽模式下使用相同的 SPI 寄存器（SPI_DATA）进行数据传输。

26. 6 I2S功能描述

26. 6. 1 音频协议

三线总线通常在处理两个通道上进行时间复用的音频数据：右声道和左声道。但是，只有一个 16 位寄存器用于发送或接收。因此，由软件向数据寄存器写入与每个通道侧相对应的适当值，或者从数据寄存器读取数据。始终先发送左声道，然后右声道（CHSIDE 对 PCM 协议没有意义）。

有四个数据和通道帧可供使用。数据可以采用以下格式发送：

- ◇ 16 位数据帧装在一个 16 位通道帧
- ◇ 16 位数据帧装在一个 32 位通道帧
- ◇ 24 位数据帧装在一个 32 位通道帧
- ◇ 32 位数据帧装在一个 32 位通道帧

当使用 16 位数据帧装在一个 32 位通道帧上时，前 16 位（MSB）是有效位，后 16 位 LSB 被强制为 0，无需任何软件操作或 DMA 请求（只有一个读/写操作）。

如果 DMA 是应用程序的首选，则 24 位和 32 位数据帧需要对 SPI_DATA 寄存器进行两次 CPU 读或写操作，或者需要两次 DMA 操作。具体而言，对于 24 位数据帧，8 个非有效位通过 0 位扩展为 32 位（通过硬件）。

对于所有数据格式和通信标准，始终首先发送最高有效位（MSB 优先）。

I2S 接口支持四种音频标准，可通过 SPI_I2SCFG.I2SSTD 位进行配置。

26. 6. 1. 1 I2S飞利浦标准

对于该标准，WS 信号用于指示正在传输哪个信道。它在第一位（MSB）可用之前的一个 CK 时钟周期被激活。

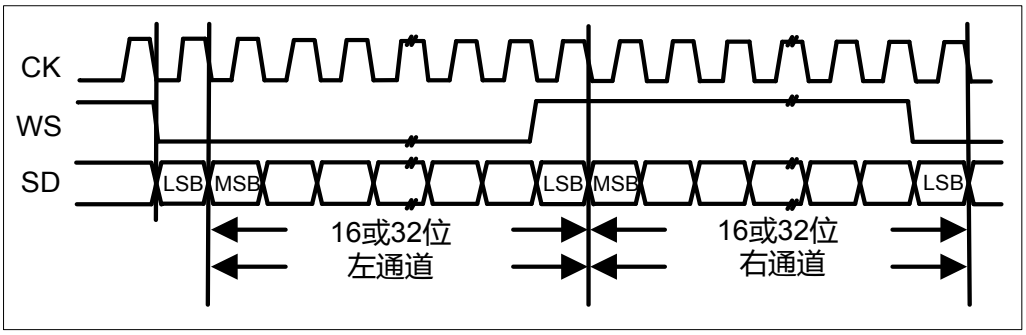


图 26-15 I2S 飞利浦标准波形（16/32 位数据帧，CPOL = 0）

数据在 CK 的下降沿（对于发送器）被锁存，并在上升沿（对于接收器）被读取。WS 信号也在 CK 的下降沿锁存。

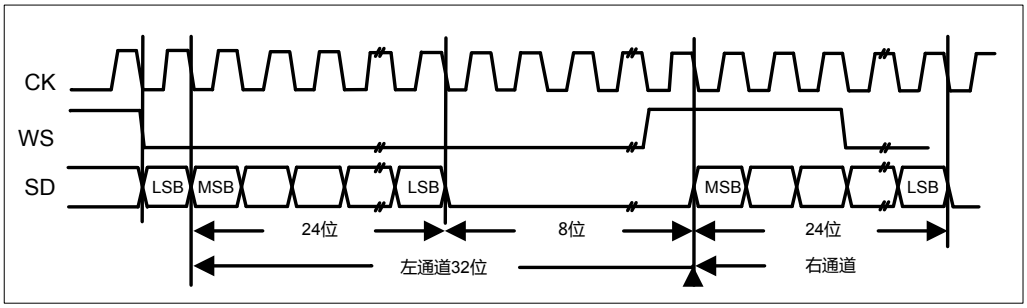


图 26-16 I2S 飞利浦标准波形（24 位数据帧，CPOL = 0）

该模式需要对 SPI_DATA 寄存器进行两次写或读操作。

◇ 在传输模式下

如果必须发送 0x123456（24 位）

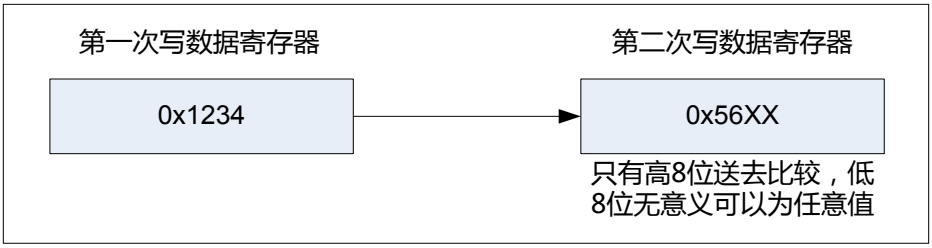


图 26-17 发送 0x123456

◇ 在接收模式下

如果收到数据 0x123456

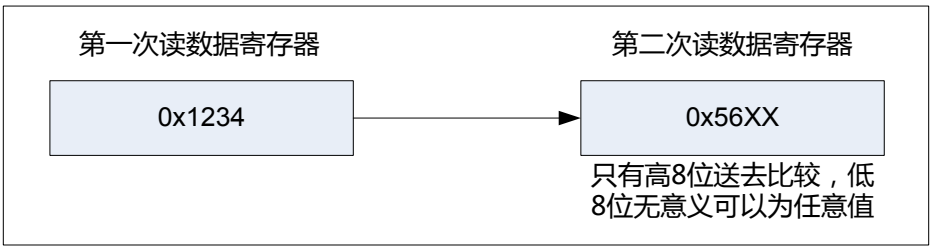


图 26-18 接收 0x123456

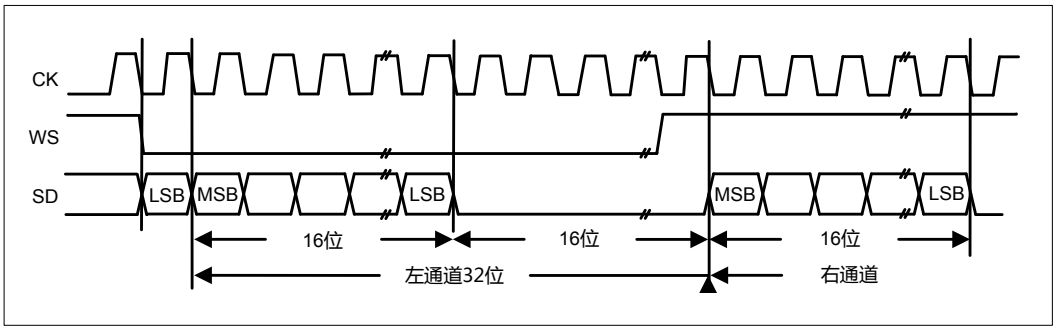


图 26-19 I2S 飞利浦标准波形（16 位数据帧扩展到 32 位通道帧，CPOL = 0）

当在 I2S 配置阶段选择将 16 位数据帧扩展到 32 位通道帧时，只需要访问一次 SPI_DATA 寄存器。其余 16 位由硬件强制为 0x0000，以将数据扩展为 32 位格式。

如果要传输的数据或接收的数据是 0x4567（0x45670000 扩展到 32 位），则需要执行下图所示的操作。

图 26-20 16 位数据帧扩展到 32 位通道帧的示例

26. 6. 1. 2 MSB对齐标准

对于该标准，WS 信号与第一个数据位（MSB）同时生成。

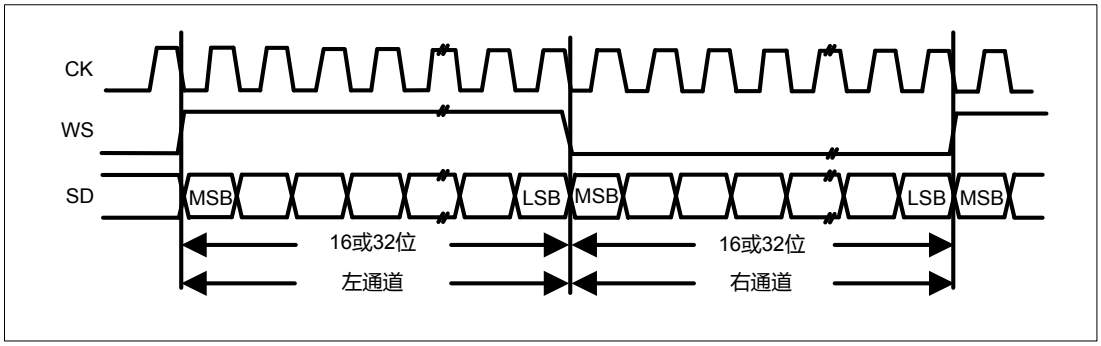


图 26-21 MSB 对齐标准波形（16/32 位数据帧，CPOL = 0）

数据在 CK 的下降沿（发送器）被锁存，并在上升沿（对于接收器）读取。

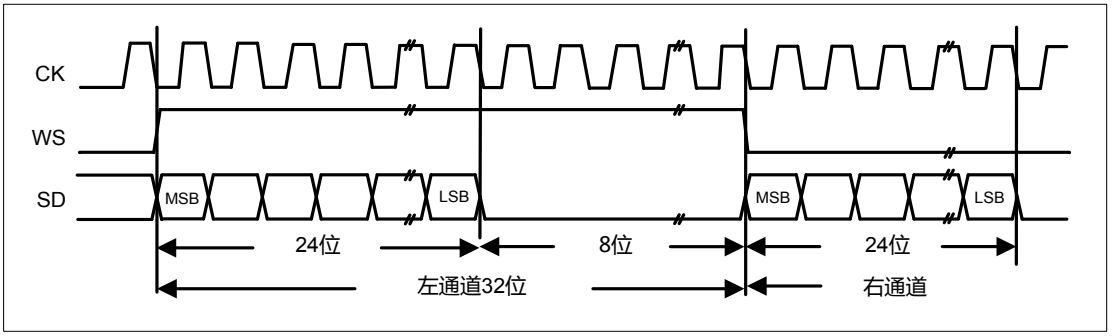


图 26-22 MSB 对齐标准波形（24 位数据帧，CPOL = 0）

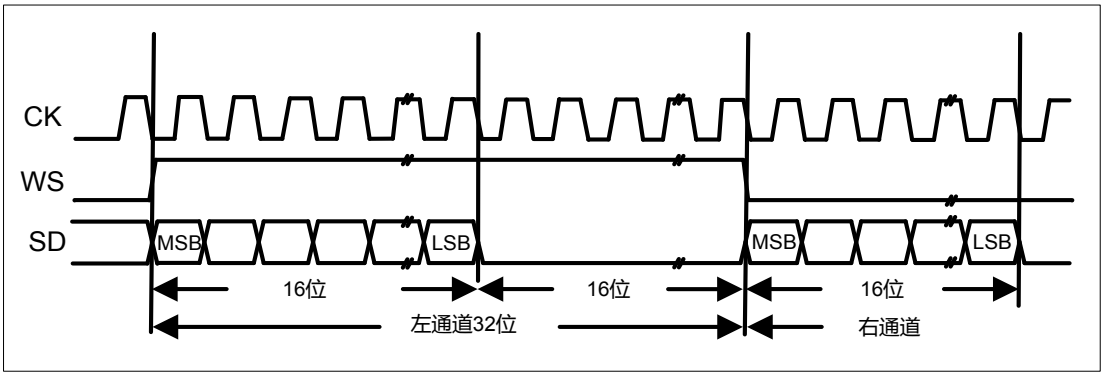


图 26-23 MSB 对齐标准波形（16 位数据帧扩展到 32 位通道帧，CPOL = 0）

26. 6. 1. 3 LSB对齐标准

该标准类似于 MSB 对齐标准（16 位和 32 位通道帧格式没有区别）。

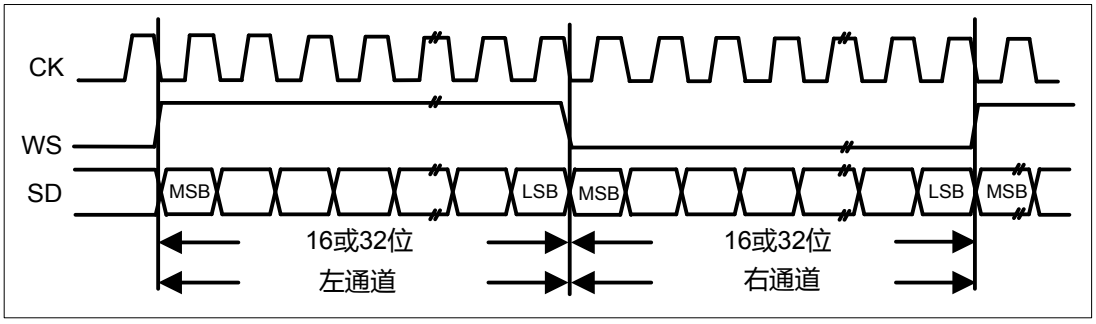


图 26-24 LSB 对齐标准波形（16/32 位数据帧，CPOL = 0）

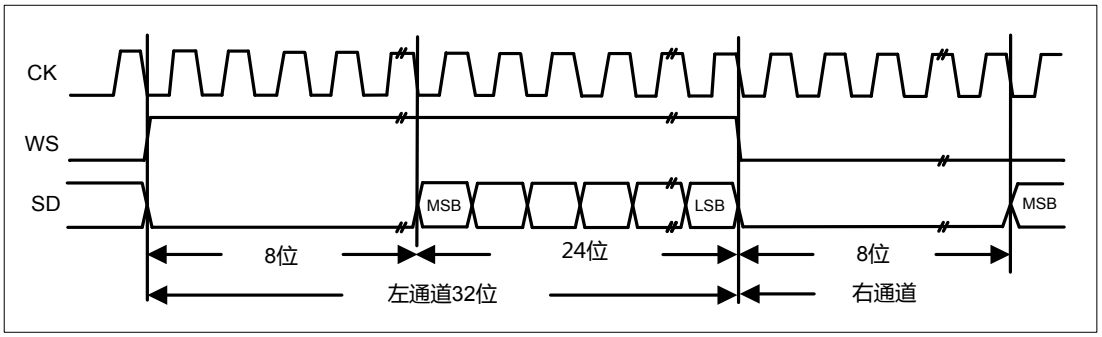


图 26-25 LSB 对齐标准波形（24 位数据帧，CPOL = 0）

◇ 在传输模式下

如果必须发送数据 0x123456，则软件或 DMA 需要对 SPI_DATA 寄存器进行两次写操作。操作如下所示。

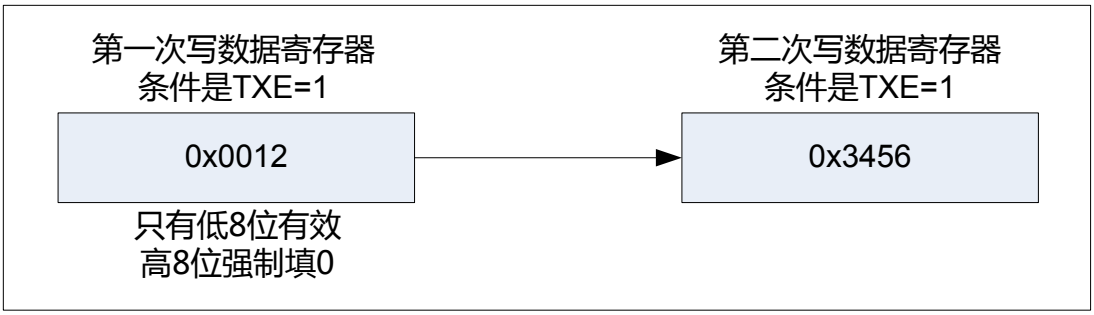


图 26-26 传输 0x123456 所需的操作

◇ 在接收模式下

如果接收到数据 0x123456，则每个 RXE 事件都需要从 SPI_DATA 寄存器执行两次连续的读操作。

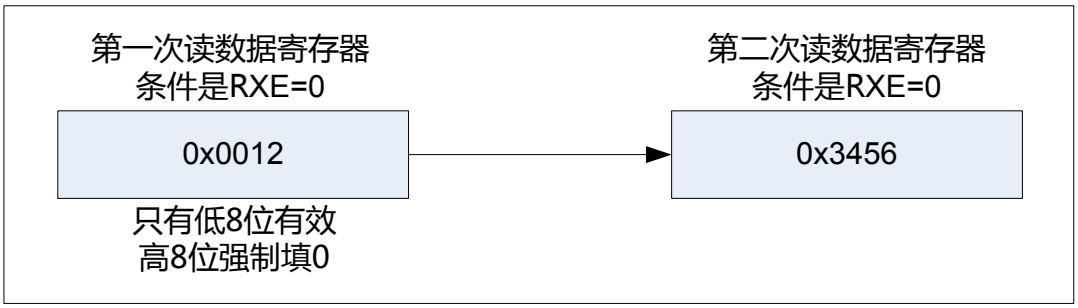


图 26-27 接收 0x123456 所需的操作

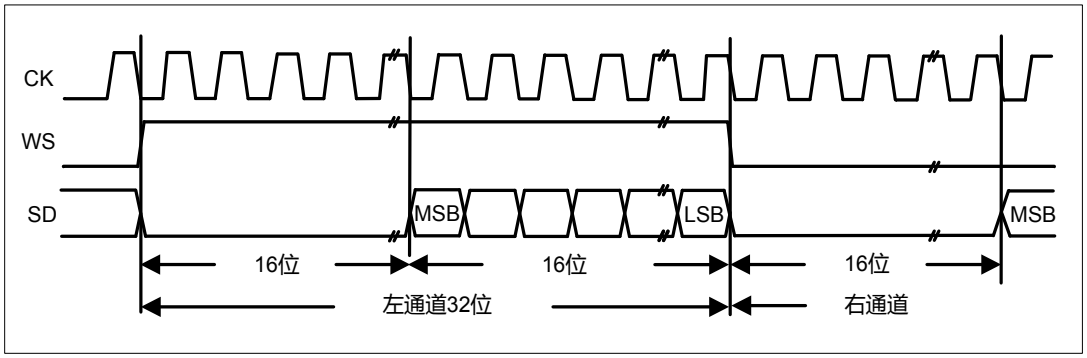


图 26-28 LSB 对齐标准波形（16 位数据帧扩展到 32 位通道帧，CPOL = 0）

在 I2S 配置阶段选择扩展到 32 位通道帧的 16 位数据帧时，只需要访问一次 SPI_DATA 寄存器。其余 16 位由硬件强制为 0x0000，以将数据扩展为 32 位格式。在这种情况下，它对应于半字 MSB。

如果要发送的数据或接收的数据是 0x4567（0x0000 4567 扩展到 32 位），则需要执行下图所示的操作。

图 26-29 16 位数据帧扩展到 32 位通道帧的示例

在传输模式下，当发生 TXE 事件时，应用程序必须写入要传输的数据（在本例中为 0x4567）。首先发送 0x0000 字段（32 位扩展）。

在接收模式下，一旦接收到有效的半字（而不是 0x0000 字段），RXE 就会被清零。

以这种方式，在两次写入或读取操作之间提供更多时间以防止下溢或溢出条件。

26. 6. 1. 4 PCM标准

对于 PCM 标准，不需要使用信道侧信息。有两种 PCM 模式（短帧和长帧）可用，使用 SPI_I2SCFG.PCMSYNC 位进行配置。

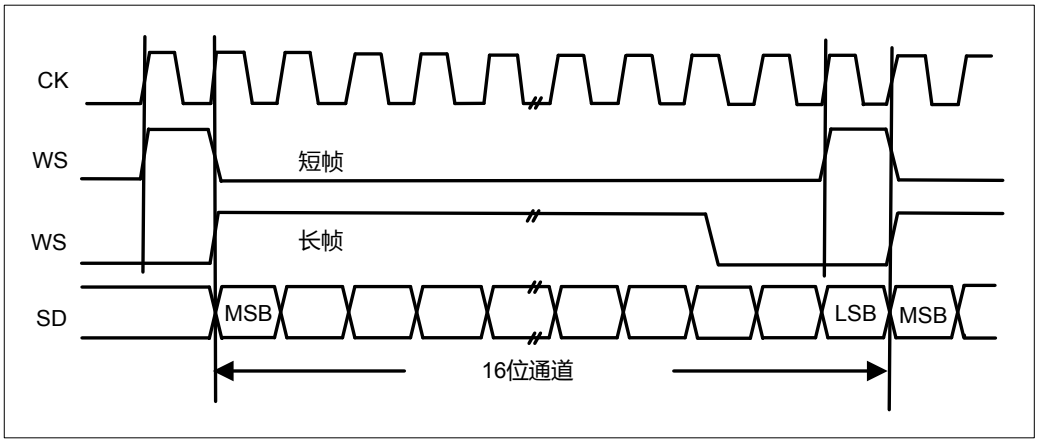


图 26-30 PCM 标准波形（16 位）

对于长帧同步，WS 信号置入时间固定为 13 位。

对于短帧同步，WS 同步信号只有一个周期。

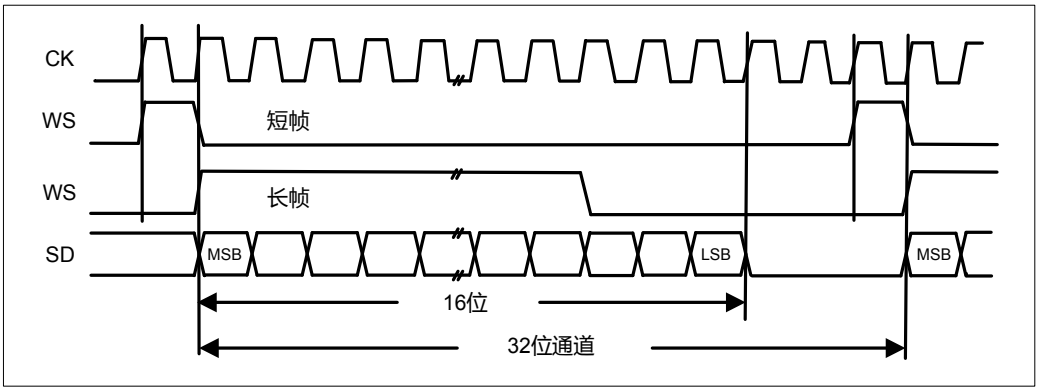


图 26-31 PCM 标准波形（16 位数据帧扩展到 32 位通道帧）

注：对于两种同步（短帧和长帧），需要指定两个连续数据（以及两个同步信号）之间的位数（SPI_I2SCFG.DATLEN 和 SPI_I2SCFG.CHLEN 位）。

26. 6. 2 时钟产生器

I2S 比特率决定了 I2S 数据线上的数据流和 I2S 时钟信号频率。

I2S 比特率=每通道的比特数×通道数×采样音频

对于 16 位音频，左右声道，I2S 比特率计算如下：

I2S 比特率=16×2×F_S

如果数据包长度为 32 位宽，则 I2S 比特率=32×2×F_S。

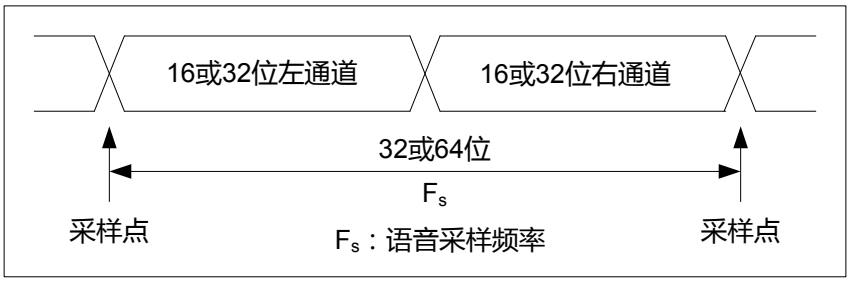


图 26-32 音频采样频率定义

音频采样频率可以是 192kHz，96kHz，48kHz，44.1kHz，32kHz，22.05kHz，16kHz，11.025kHz 或 8kHz（或该范围内的任何其他值）。为了达到所需频率，需要根据以下公式对线性分频器进行编程：

在 I2S，MSB，LSB 模式中，CH 为 2。

在 PCM 模式下，CH 为 1。

当产生主时钟时（SPI_I2SPR 寄存器中的 MCKOE 位置 1）：

当通道帧为 16 位宽时，FS = I2S_CLK / [（16 * 2）*（（CH * I2SDIV）+ ODD）* 8）]

当通道帧为 32 位宽时，FS = I2S_CLK / [（32 * 2）*（（CH * I2SDIV）+ ODD）* 4）]

禁用主时钟时（SPI_I2SPR 寄存器中的 MCKOE 位清零）：

当通道帧为 16 位宽时，FS = I2S_CLK / [（16 * 2）*（（CH * I2SDIV）+ ODD））]

当通道帧为 32 位宽时，FS = I2S_CLK / [（32 * 2）*（（CH * I2SDIV）+ ODD））]

下表提供了不同时钟配置的示例精度值。

APB_CLK（MHz）	Data length	I2SDIV	I2SODD	MCLK	Target fs（Hz）	Real fs（Hz）	Error
32	16	5	0	No	96000	100000	4.1667%
32	32	2	0	No	96000	100000	4.1667%
32	16	10	1	No	48000	47619	0.7937%
32	32	5	0	No	48000	50000	4.1667%
32	16	11	1	No	44100	43478	1.4098%
32	32	5	1	No	44100	45454	3.0715%
32	16	15	1	No	32000	32258	0.8065%
32	32	8	0	No	32000	31250	2.3430%
32	16	22	1	No	22050	22222	0.7811%

APB_CLK（MHz）	Data length	I2SDIV	I2SODD	MCLK	Target fs（Hz）	Real fs（Hz）	Error
32	32	11	1	No	22050	21739	1.4098%
32	16	31	1	No	16000	15873	0.7937%
32	32	15	1	No	16000	16129	0.8065%
32	16	45	1	No	11025	10989	0.3264%
32	32	22	1	No	11025	11111	0.7811%
32	16	62	1	No	8000	8000	0.0000%
32	32	31	1	No	8000	7936	0.7937%
32	16	2	0	Yes	32000	31250	2.3430%
32	32	2	0	Yes	32000	31250	2.3430%
32	16	3	0	Yes	22050	20833	5.5170%
32	32	3	0	Yes	22050	20833	5.5170%
32	16	4	0	Yes	16000	15625	2.3428%
32	32	4	0	Yes	16000	15625	2.3428%
32	16	5	1	Yes	11025	11363	3.0715%
32	32	5	1	Yes	11025	11363	3.0715%
32	16	8	0	Yes	8000	7812	2.3428%
32	32	8	0	Yes	8000	7812	2.3428%

表 26-1　不同时钟配置的示例精度值

注：其他配置可能允许最佳时钟精度。

26. 6. 3　I2S主机模式

使用 I2S 功能时只能工作在主机模式。这意味着在 CK 引脚上生成串行时钟以及字选择信号 WS。主时钟（MCK）的输出与否，则由 SPI_I2SPR.MCKOE 位控制。

26. 6. 3. 1　设置流程

1. 配置 SPI_I2SPR.I2SDIV 位，定义串行时钟波特率，以达到正确的音频采样频率。同时必须配置 SPI_I2SPR.ODD 位。
2. 配置 CKPOL 位以定义通信时钟的稳定电平。如果需要将主时钟 MCK 提供给外部 DAC/ADC 音频组件，则将 SPI_I2SPR.MCKOE 位置 1（I2SDIV 和 ODD 值应根据 MCK 输出的状态进行计算，更多详细信息，请参见：时钟产生器）。
3. 将 SPI_I2SCFG.I2SMOD 位置 1 以激活 I2S 功能，并通过 SPI_I2SCFG.I2SSTD 和 SPI_I2SCFG.PCMSYNC 位选择 I2S 标准，通过 SPI_I2SCFG.DATLEN 位选择数据长度，通过配置 SPI_I2SCFG.CHLEN 位来选择通道长度。通过 SPI_I2SCFG.I2SCFG 位选择方向（发送器或接收器）。
4. 如果需要，通过配置 SPI_CON2 寄存器选择 DMA 功能。
5. 如果需要，通过配置 SPI_IER 寄存器选择所有可能的中断源。
6. 必须配置 SPI_I2SCFG.I2SE 位。WS 和 CK 配置为输出模式。如果 SPI_I2SPR.MCKOE 位置 1，MCK 也是输出。

26. 6. 3. 2 传输序列

当半字写入 TX FIFO 时，传输序列开始。

写入 TX FIFO 的第一个数据应对应于左声道数据。当左声道的数据写入 TX FIFO 后必须紧跟着写入对应于右声道数据。CHSIDE 标志指示当前发送的声道。

必须将全帧视为左右声道数据传输。不会有仅发送左声道的部分帧。

有关写入操作的更多详细信息，具体取决于所选的 I2S 标准模式，请参见：音频协议。

为了确保连续的音频数据传输，必须在当前传输结束之前将对 SPI_DATA 寄存器写入下一个要传输的数据。若要禁用 I2S，必须等待 TXE = 1 且 BUSY = 0 时，通过清除 SPI_I2SCFG.I2SE 位来关闭 I2S。

26. 6. 3. 3 接收序列

操作模式与传输模式相同，除了第 3 点（参见：I2S 主机模式中描述的过程）之外，其中配置应通过 SPI_I2SCFG.I2SCFG 位来设置接收模式。

RX FIFO 满时，RXF 标志置 1，如果 SPI_IER.RXFIE 位置 1，则产生中断。根据数据和通道长度配置，右声道或左声道接收的音频值可能是由于进入 RX FIFO 的一次或两次接收所致。

通过读 SPI_DATA 寄存器来清零 RXF 位。

有关根据所选 I2S 标准模式进行读取操作的更多详细信息，请参见：音频协议。

如果在尚未读取先前接收的数据的同时接收到新的数据，则生成溢出并设置 RXOV 标志。如果 SPI_IER 寄存器中的 RXOVIE 位置 1，则会产生中断以指示错误。

26. 6. 4 I2S状态标志

为应用程序提供了六个状态标志，以完全监视 I2S 总线的状态。

26. 6. 4. 1 发送FIFO空标志（TXE）

设置时，TXE 标志指示发送 FIFO 为空，并且下一个要发送的数据可以加载到 FIFO 中。通过写 SPI_DATA 寄存器清零 TXE 标志。

26. 6. 4. 2 发送FIFO满标志（TXF）

设置时，TXF 标志指示发送 FIFO 已满，并且下一个要发送的数据无法加载到 FIFO 中。TXF 标志由要发送的下一个数据清除。

26. 6. 4. 3 接收FIFO空标志（RXE）

设置时，RXE 标志指示接收 FIFO 为空，并且下一个要接收的数据可以加载到 FIFO 中。RXE 标志由下一个数据接收到 FIFO 中清除。

26. 6. 4. 4 接收FIFO满标志（RXF）

设置后，RXF 标志指示接收 FIFO 已满，并且下一个要接收的数据无法加载到 FIFO 中。通过对 SPI_DATA 寄存器的读访问来清除 RXF 标志。

26. 6. 4. 5 BUSY标志（BUSY）

BUSY 标志由硬件置位来清除（写入此标志无效）。当 BUSY 标志置 1 时，表示 I2S 上正在进行数据传输（I2S 总线忙）。BUSY 标志可以用于检测传输的结束，从而防止最后一次传输损坏。

在以下任一条件下清除 BUSY 标志：

◇ 当 I2S 正确地禁用

◇ 传输完成时

◇ 在发送模式下，BUSY 标志在所有传输期间保持高电平

注：建议始终使用 TXTH 和 RXTH 标志（而不是 BUSY 标志）来处理数据传输或接收操作。

26. 6. 4. 6 声道标志（CHSIDE）

此标志是提供使用者判断此时总线上正在传输的声道为左声道还右声道使用，只有在 BUSY 标志被设置时有效。该标志在 PCM 标准中没有意义（短帧和长帧模式）。

26. 6. 5 I2S错误标志

I2S 单元有四个错误标志。

26. 6. 5. 1 发送FIFO溢出标志（TXOV）

发生发送溢出条件的情况是发送 FIFO 已满，但用户正在执行写操作。在这种情况下，新写入的数据不会加载到发送 FIFO 中。通过对 SPI_ICR 寄存器的写访问来清除 TXOV 位，或者在新传输开始时它将自动清零。

26. 6. 5. 2 发送FIFO下溢标志（TXUD）

发生发送下溢运行条件是当发送 FIFO 为空时，但主机想要从发送 FIFO 拿取数据。在这种情况下，发送 FIFO 的内容不会被发送至总线上。通过对 SPI_ICR 寄存器的写访问来清除 TXUD 位，或者在新的传输开始时它将自动清零。

26. 6. 5. 3 接收FIFO溢出标志（RXOV）

当完成下一个数据帧的接收而接收 FIFO 的前 16 帧的读操作尚未完成时（设置了 RXF 标志的情况），发生接收溢出条件。

在这种情况下，接收 FIFO 的内容不会将接收的新数据存入。SPI_DATA 寄存器的读操作返回先前接收的帧。所有其他在 RXF 标志设置的情况下，接收的新数据都将丢失。

通过对 SPI_ICR 寄存器的写访问来清除 RXOV 位，或者在新的传输开始时它将自动清零。

26. 6. 5. 4 接收FIFO下溢标志（RXUD）

当接收 FIFO 为空时，但用户正在执行读操作，会发生接收下溢条件。在这种情况下，读取操作不会从接收 FIFO 中读到有效的数据。

通过对 SPI_ICR 寄存器的写访问来清除 RXUD 位，或者在新的传输开始时它将自动清零。

26. 7 特殊功能寄存器

26. 7. 1 寄存器列表

外设寄存器可支持半字（16 位）或字（32 位）访问。

SPI 寄存器列表		
名称	偏移地址	描述
SPI_CON1	0000_H	SPI 控制寄存器 1
SPI_CON2	0004_H	SPI 控制寄存器 2
SPI_STAT	0008_H	SPI 状态寄存器
SPI_DATA	$000C_H$	SPI 数据寄存器
SPI_CRCPOLY	0010_H	SPI CRC 多项式寄存器
SPI_RXCRC	0014_H	SPI RX CRC 寄存器
SPI_TXCRC	0018_H	SPI TX CRC 寄存器
SPI_I2SCFG	$001C_H$	SPI I2S 配置寄存器
SPI_I2SPR	0020_H	SPI I2S 预分频寄存器
SPI_IER	0024_H	SPI 中断启用寄存器
SPI_IDR	0028_H	SPI 中断禁用寄存器
SPI_IVS	$002C_H$	SPI 中断有效状态寄存器
SPI_RIF	0030_H	SPI 原始中断标志状态寄存器
SPI_IFM	0034_H	SPI 中断标志屏蔽状态寄存器
SPI_ICR	0038_H	SPI 中断清除寄存器

26.7.2 寄存器描述

26.7.2.1 SPI控制寄存器 1（SPI_CON1）

SPI 控制寄存器 1（SPI_CON1）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																BIDEN	BIDOEN	CRCEN	NXTCRC	FLEN	RXO	SSEN	SSOUT	LSBFST	SPIEN	BAUD			MSTREN	CPOL	CPHA

Reserved	Bit 31-16	—	**保留**
BIDEN	Bit 15	R/W	**启用双向数据模式** 该位使用通用单双向数据线实现半双工通信。双向模式激活时，保持 RXO 位清零。 0：选择双线单向通信数据模式 1：选择单线双向通信数据模式 注：该位不用于 I2S 模式。
BIDOEN	Bit 14	R/W	**双向模式下的输出使能** 该位与 BIDEN 位组合在一起选择双向模式下的传输方向 0：禁用输出（仅接收模式） 1：启用输出（仅发送模式） 注：在主机模式下，使用 MOSI 引脚，在从机模式，使用 MISO 引脚。在 I2S 模式下不使用该位。
CRCEN	Bit 13	R/W	**硬件 CRC 计算使能** 0：禁用 CRC 计算 1：启用 CRC 计算 注：为确保正确操作，只应在禁止 SPI（SPIEN =“0”）时对此位执行写操作。I2S 模式中不使用该位。
NXTCRC	Bit 12	R/W	**下一次传输 CRC** 0：数据传输结束时不传输 CRC 1：数据传输结束时传输 CRC 注：若当前传输需要传送 CRC 时，在写入数据后即可将该位置 1。若为接收时，则在写入无效数据后将该位置 1。I2S 模式中不使用该位。
FLEN	Bit 11	R/W	**数据帧长度** 0：选择 8 位数据帧格式进行发送/接收 1：选择 16 位数据帧格式进行发送/接收 注：为确保正确操作，只应在禁止 SPI（SPIEN =“0”）时对此位执行写操作。I2S 模式中不使用该

			位。
RXO	Bit 10	R/W	**启用仅接收模式** 该位允许使用单个单向线进行单工通信，以专门接收数据。当仅接收模式处于活动状态时，保持 BIDEN 位清零。该位在多区域系统中也很有用，在该系统中不访问此特定从机，来自被访问从机的输出不会被破坏。 0：全双工（发送和接收） 1：禁用输出（仅接收模式） 注：该位不用于 I2S 模式。
SSEN	Bit 9	R/W	**软件从机选择管理** 当 SSEN 位置 1 时，NSS 引脚输入将被 SSOUT 位的值替换。 0：禁用软件从机选择管理 1：启用软件从机选择管理 注：该位不用于 I2S 模式和 SPI 兼容模式。
SSOUT	Bit 8	R/W	**软件从机选择，由该位决定模块内部片选** 0：内部片选 NSS 为 0，模块选中 1：内部片选 NSS 为 1，模块未选中 仅当 SSEN 位置 1 时，该位才有效。此位的值将作用到模块内部片选 NSS 上，并忽略外部 NSS 管脚的输入值。 注：该位不用于 I2S 模式和 SPI 兼容模式。
LSBFST	Bit 7	R/W	**先发送 LSB** 0：首先使用 MSB 发送/接收数据 1：首先使用 LSB 发送/接收数据 注 1：通信正在进行时，不应更改此位。 注 2：该位不用于 I2S 模式和 SPI 兼容模式。
SPIEN	Bit 6	R/W	**SPI 启用** 0：禁用 SPI 外设 1：启用 SPI 外设 注：该位不用于 I2S 模式。
BAUD	Bit 5-3	R/W	**波特率控制** 000：F_{PCLK} / 2 001：F_{PCLK} / 4 010：F_{PCLK} / 8 011：F_{PCLK} / 16 100：F_{PCLK} / 32 101：F_{PCLK} / 64 110：F_{PCLK} / 128 111：F_{PCLK} / 256 注：通信正在进行时，不应更改这些位。在 I2S 模式下不使用该位。
MSTREN	Bit 2	R/W	**主机选择**

			0：从机模式 1：主机模式 注：通信正在进行时，不应更改此位。在 I2S 模式下不使用该位。
CPOL	Bit 1	R/W	**时钟极性** 0：在空闲的状态下，SCK 引脚保持低电平输出 1：在空闲的状态下，SCK 引脚保持高电平输出 注：通信正在进行时，不应更改此位。该位不用于 I2S 模式和 SPI 兼容模式。
CPHA	Bit 0	R/W	**时钟相位** 0：从第一个时钟边沿开始采样数据 1：从第二个时钟边沿开始采样数据 注：通信正在进行时，不应更改此位。该位不用于 I2S 模式和 SPI 兼容模式。

26. 7. 2. 2 SPI控制寄存器 2（SPI_CON2）

SPI 控制寄存器 2（SPI_CON2）

偏移地址：04_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	**保留**
RXFTH	Bit15-14	R/W	**接收 FIFO 触发阈值** 这用于选择接收 FIFO 生成接收 FIFO 触发阈值标志的阈值。 00：当接收 FIFO 中有 1 个字符时触发 01：当接收 FIFO 中有 4 个字符时触发 10：当接收 FIFO 中有 8 个字符时触发 11：当接收 FIFO 中有 14 个字符时触发 注：如果 RX DMA 使能，该位也提供 RX DMA 请求的条件。
TXFTH	Bit 13-12	R/W	**发送 FIFO 触发阈值** 这用于选择发送 FIFO 产生发送 FIFO 触发阈值标志的阈值。 00：当发送 FIFO 空时触发 01：当发送 FIFO 中有 2 个字符时触发 10：当发送 FIFO 中有 4 个字符时触发 11：当发送 FIFO 中有 8 个字符时触发 注：如果 TX DMA 使能，该位也提供 TX DMA 请求的条件。
Reserved	Bit 11-5	—	**保留**
FRF	Bit 4	R/W	**通信模式选择** 0：SPI 标准模式（摩托罗拉通信协议） 1：SPI 兼容模式（兼容 TI 通信协议） 注：只有在禁止 SPI（SPIEN = 0）时才能写入该位。在 I2S 模式下不使用该位。
NSSP	Bit 3	R/W	**NSS 脉冲管理** 该位仅用于主机模式。它允许 SPI 在进行连续传输时在两个连续数据之间产生 NSS 脉冲。在单次数据传输的情况下，它会在传输后强制 NSS 引脚为高电平。如果 CPHA =1 或 FRF =1，则没有意义。 0：没有 NSS 脉冲 1：产生 NSS 脉冲 注 1：只有在禁止 SPI（SPIEN = 0）时才能写入

			该位。 注 2：该位不用于 I2S 模式和 SPI 兼容模式。
NSSOE	Bit 2	R/W	**NSS 引脚输出使能** 0：在主机模式下禁止 NSS 输出，可在多主模式配置下工作 1：在主机模式下使能 NSS 输出，不能在多主模式环境下工作 注：该位不用于 I2S 模式和 SPI 兼容模式。
TXDMA	Bit 1	R/W	**TX DMA 使能** 该位置 1 时，只要 TXTH 标志置 1，就会产生 DMA 请求。 0：禁用 TXDMA 1：启用 TXDMA
RXDMA	Bit 0	R/W	**RX DMA 使能** 该位置 1 时，只要 RXTH 标志置 1，就会产生 DMA 请求。 0：禁用 RXDMA 1：启用 RXDMA

26. 7. 2. 3 SPI状态寄存器（SPI_STAT）

SPI 状态寄存器（SPI_STAT）

偏移地址：08_H

复位值：$00000000\_00000000\_00000000\_01010001_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			RXFLV					Reserved			TXFLV					BUSY	CHSIDE	Reserved	RXTH	RXUD	RXOV	RXF	RXE	Reserved			TXTH	TXUD	TXOV	TXF	TXE

Reserved	Bit 31-29	—	**保留**
RXFLV	Bit 28-24	R	**RX FIFO 的水平** 该位提供 RX FIFO 的水平值。
Reserved	Bit 23-21	—	**保留**
TXFLV	Bit 20-16	R	**TX FIFO 水平** 该位提供 TX FIFO 的水平值。
BUSY	Bit 15	R	**忙标志** 0：SPI（或 I2S）不繁忙 1：SPI（或 I2S）忙于通信 该标志由硬件设置和清除。
CHSIDE	Bit 14	R	**声道侧** 0：当前正在发送或接收左声道 1：当前正在传输或接收右声道 注：该位不用于 SPI 模式。它在 PCM 模式下没有意义。
Reserved	Bit 13	—	**保留**
RXTH	Bit 12	R	**接收 FIFO 水平超出阈值** 该位提供接收FIFO水平超过阈值。 0：接收 FIFO 水平没有超过阀值。 1：接收FIFO水平超过阀值。
RXUD	Bit 11	R	**接收 FIFO 下溢** 0：接收 FIFO 没有下溢 1：接收FIFO下溢
RXOV	Bit 10	R	**接收 FIFO 溢出** 0：接收 FIFO 没有溢出 1：接收FIFO溢出
RXF	Bit 9	R	**接收 FIFO 满** 0：接收 FIFO 没有满 1：接收FIFO满
RXE	Bit 8	R	**接收 FIFO 空** 0：接收 FIFO 没有空

			1：接收FIFO空
Reserved	Bit 7-5	—	**保留**
TXTH	Bit 4	R	**发送 FIFO 水平低于阈值** 该位提供发送FIFO水平低于阈值。 0：发送 FIFO 水平高于阀值。 1：发送FIFO水平低过阀值。
TXUD	Bit 3	R	**发送 FIFO 下溢** 0：发送 FIFO 没有下溢 1：发送FIFO下溢
TXOV	Bit 2	R	**发送 FIFO 溢出** 0：发送 FIFO 没有溢出 1：发送FIFO溢出
TXF	Bit 1	R	**发送 FIFO 满** 0：发送 FIFO 没有满 1：发送FIFO满
TXE	Bit 0	R	**发送 FIFO 空** 0：发送 FIFO 没有空 1：发送FIFO空

26. 7. 2. 4 SPI数据寄存器（SPI_DATA）

SPI 数据寄存器（SPI_DATA）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	**保留**
DATA	Bit 15-0	R/W	**数据寄存器** 收到或将要传输的数据 数据寄存器用作RX FIFO和TX FIFO之间的接口。读取数据寄存器时，访问RX FIFO，而写入数据寄存器访问TX FIFO。 注：数据始终是右对齐的。写入寄存器时忽略未使用的位，读取寄存器时读取为0。RX阈值设置必须始终与当前使用的读取访问权限相对应。

26. 7. 2. 5 SPI CRC多项式寄存器（SPI_CRCPOLY）

SPI CRC 多项式寄存器（SPI_CRCPOLY）																															
偏移地址：10_H																															
复位值：00000000_00000000_00000000_00000111																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CRCPOLY															

Reserved	Bit 31-16	—	保留
CRCPOLY	Bit 15-0	R/W	**CRC多项式寄存器** 该寄存器包含CRC计算的多项式。 CRC多项式（0007h）是该寄存器的复位值。可以根据需要配置另一个多项式。 注1：多项式值应仅为奇数。没有支持偶数。 注2：当使用I2S时，这些位应配置为默认值。

26. 7. 2. 6 SPI RX CRC寄存器（SPI_RXCRC）

SPI RX CRC 寄存器（SPI_RXCRC）																															
偏移地址：14_H																															
复位值：00000000_00000000_00000000_00000000_B																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																RXCRC															

Reserved	Bit 31-16	—	保留
RXCRC	Bit 15-0	R	**接收 CRC 值** 使能 CRC 计算后，RXCRC<15:0>位将包含后续接收字节在计算后所得到的 CRC 值。当 SPI_CON1 寄存器中的 CRCEN 位写入 1 时，此寄存器复位。CRC 通过 SPI_CRCPOLY 寄存器中编程的多项式连续计算。数据帧长度设置为 8 位数据（SPI_CON1 的 FLEN 位清零）时，仅考虑 8 个 LSB 位。CRC 计算依据任意 CRC8 标准进行。选择 16 位数据帧长度（SPI_CON1 寄存器的 FLEN 位置 1）时，考虑此寄存器的全部 16 个位。CRC 计算依据任意 CRC16 标准进行。 注 1：当 BUSY 标志置 1 时，读取此寄存器可能返

			回一个不正确的值。 注 2：当使用 I2S 时，这些位应配置为默认值。

26. 7. 2. 7 SPI TX CRC寄存器（SPI_TXCRC）

SPI TX CRC 寄存器（SPI_TXCRC）																													
偏移地址：18_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																TXCRC															

Reserved	Bit 31-16	—	保留
TXCRC	Bit 15-0	R	**发送 CRC 值** 使能 CRC 计算后，TXCRC<15:0>位将包含后续发送字节在计算后所得到的 CRC 值。当 SPI_CON1 寄存器中的 CRCEN 位写入 1 时，此寄存器复位。CRC 通过 SPI_CRCPOLY 寄存器中编程的多项式连续计算。数据帧长度设置为 8 位数据（SPI_CON1 的 FLEN 位清零）时，仅考虑 8 个 LSB 位。CRC 计算依据任意 CRC8 标准进行。选择 16 位数据帧长度（SPI_CON1 寄存器的 FLEN 位置 1）时，考虑此寄存器的全部 16 个位。CRC 计算依据任意 CRC16 标准进行。 注 1：当 BUSY 标志置 1 时，读取此寄存器可能返回一个不正确的值。 注 2：当使用 I2S 时，这些位应配置为默认值。

26. 7. 2. 8 SPI I2S配置寄存器（SPI_I2SCFG）

SPI I2S 配置寄存器（SPI_I2SCFG）

偏移地址：$1C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				I2SMOD	I2SE	I2SCFG		PCMSYNC	Reserved	I2SSTD		CKPOL	DATLEN		CHLEN

Reserved	Bit 31-12	—	**保留**
I2SMOD	Bit 11	R/W	**I2S 模式选择** 0：选择 SPI 模式 1：选择 I2S 模式 注：当 SPI 或 I2S 禁止时，应配置该位为默认值。
I2SE	Bit 10	R/W	**I2S 启用** 0：禁用 I2S 外设 1：启用 I2S 外设 注：该位不用于 SPI 模式。
I2SCFG	Bit9-8	R/W	**I2S 模式配置** 0x：保留 10：发送 11：接收 注：当 I2S 禁止时应配置这些位为默认值。它们不用于 SPI 模式。
PCMSYNC	Bit 7	R/W	**PCM 帧同步** 0：短帧同步 1：长帧同步 注：仅当 I2SSTD = 11（使用 PCM 标准）时，该位才有意义。它不用于 SPI 模式。
Reserved	Bit 6	—	**保留**
I2SSTD	Bit 5-4	R/W	**I2S 标准选择** 00：I2S 飞利浦标准 01：MSB 对齐标准（左对齐） 10：LSB 对齐标准（右对齐） 11：PCM 标准 注：为了正确操作，当 I2S 禁止时应配置这些位为默认值。它们不用于 SPI 模式。
CKPOL	Bit 3	R/W	**非活动状态时钟极性** 0：I2S 时钟非活动状态为低电平 1：I2S 时钟非活动状态是高电平 注：为了正确操作，当 I2S 禁止时应配置该位为默认值。它不用于 SPI 模式。CKPOL 位不会影响用

			于接收或发送 SD 和 WS 信号的 CK 边沿灵敏度。
DATLEN	Bit 2-1	R/W	**要传输的数据长度** 00：16 位数据长度 01：24 位数据长度 10：32 位数据长度 11：不允许 注：为了正确操作，当 I2S 禁止时应配置这些位为默认值它们不用于 SPI 模式。
CHLEN	Bit 0	R/W	**通道长度**（每个音频通道的位数） 0：16 位宽 1：32 位宽 当 DATLEN 大于 00 时，通道长度需固定为 32 位。 注：为了正确操作，当 I2S 禁止时应配置该位为默认值。它不用于 SPI 模式。

26. 7. 2. 9 SPI I2S预分频寄存器（SPI_I2SPR）

SPI I2S 预分频寄存器（SPI_I2SPR）																															
偏移地址：20_H																															
复位值：$00000000\_00000000\_00000000\_00000010_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					EXTCKEN	MCKOE	ODD	I2SDIV							

Reserved	Bit 31-11	—	保留
EXTCKEN	Bit 10	R/W	**外部 I2S 时钟使能** 0：选择 APB 时钟 1：选择外部时钟 注：当 I2S 禁止时应配置该位为默认值。该位不用于 SPI 模式。
MCKOE	Bit 9	R/W	**主时钟输出使能** 0：禁用主时钟输出 1：启用主时钟输出 注：当 I2S 禁止时应配置该位为默认值。它不用于 SPI 模式。
ODD	Bit 8	R/W	**预分频器的奇数系数** 0：实际分频器值= I2SDIV * 2 1：实际分频器值=（I2SDIV * 2）+ 1 注：当 I2S 禁止时应配置该位为默认值。它不用于 SPI 模式。
I2SDIV	Bit 7-0	R/W	**I2S 线性预分频器** I2SDIV<7:0> = 0 是禁用值。 注：当 I2S 禁止时应配置这些位为默认值。它们不用于 SPI 模式。

26. 7. 2. 10 SPI中断启用寄存器（SPI_IER）

SPI 中断启用寄存器（SPI_IER）																															
偏移地址：24_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													FREIE	MODFIE	CRCERRIE	Reserved			RXTHIE	RXUDIE	RXOVIE	RXFIE	Reserved				TXTHIE	TXUDIE	TXOVIE	Reserved	TXEIE

Reserved	Bit 31-19	—	保留
FREIE	Bit 18	W1	**帧格式错误中断启用** 0：无效 1：启用中断
MODFIE	Bit 17	W1	**模式故障中断启用** 0：无效 1：启用中断
CRCERRIE	Bit 16	W1	**CRC 错误中断启用** 0：无效 1：启用中断
Reserved	Bit 15-13	—	保留
RXTHIE	Bit 12	W1	**接收 FIFO 超过阈值中断启用** 0：无效 1：启用中断
RXUDIE	Bit 11	W1	**接收 FIFO 欠载中断启用** 0：无效 1：启用中断
RXOVIE	Bit 10	W1	**接收 FIFO 溢出中断启用** 0：无效 1：启用中断
RXFIE	Bit 9	W1	**接收 FIFO 满中断启用** 0：无效 1：启用中断
Reserved	Bit 8-5	—	保留
TXTHIE	Bit 4	W1	**发送 FIFO 低于阈值中断启用** 0：无效 1：启用中断
TXUDIE	Bit 3	W1	**发送 FIFO 欠载中断启用** 0：无效 1：启用中断
TXOVIE	Bit 2	W1	**发送 FIFO 溢出中断启用** 0：无效

			1：启用中断
Reserved	Bit 1	—	保留
TXEIE	Bit 0	W1	发送 FIFO 空中断启用 0：无效 1：启用中断

26. 7. 2. 11 SPI中断禁用寄存器（SPI_IDR）

SPI 中断禁用寄存器（SPI_IDR）

偏移地址：28_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													FREID	MODFID	CRCERRID	Reserved			RXTHID	RXUDID	RXOVID	RXFID	Reserved				TXTHID	TXUDID	TXOVID	Reserved	TXEID

Reserved	Bit 31-19	—	保留
FREID	Bit 18	W1	**帧格式错误中断禁用** 0：无效 1：禁用中断
MODFID	Bit 17	W1	**模式故障中断禁用** 0：无效 1：禁用中断
CRCERRID	Bit 16	W1	**CRC 错误中断禁用** 0：无效 1：禁用中断
Reserved	Bit 15-13	—	保留
RXTHID	Bit 12	W1	**接收 FIFO 超过阈值中断禁用** 0：无效 1：禁用中断
RXUDID	Bit 11	W1	**接收 FIFO 欠载中断禁用** 0：无效 1：禁用中断
RXOVID	Bit 10	W1	**接收 FIFO 溢出中断禁用** 0：无效 1：禁用中断
RXFID	Bit 9	W1	**接收 FIFO 满中断禁用** 0：无效 1：禁用中断
Reserved	Bit 8-5	—	保留
TXTHID	Bit 4	W1	**发送 FIFO 低于阈值中断禁用** 0：无效 1：禁用中断
TXUDID	Bit 3	W1	**发送 FIFO 欠载中断禁用** 0：无效 1：禁用中断
TXOVID	Bit 2	W1	**发送 FIFO 溢出中断禁用** 0：无效

			1：禁用中断
Reserved	Bit 1	—	**保留**
TXEID	Bit 0	W1	**发送 FIFO 空中断禁用** 0：无效 1：禁用中断

26. 7. 2. 12 SPI中断有效状态寄存器（SPI_IVS）

SPI 中断有效状态寄存器（SPI_IVS）

偏移地址：$2C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													FREIV	MODFIV	CRCERRIV	Reserved			RXTHIV	RXUDIV	RXOVIV	RXFIV	Reserved				TXTHIV	TXUDIV	TXOVIV	Reserved	TXEIV

Reserved	Bit 31-19	—	保留
FREIV	Bit 18	R	**帧格式错误中断有效** 0：无效 1：中断有效
MODFIV	Bit 17	R	**模式故障中断有效** 0：无效 1：中断有效
CRCERRIV	Bit 16	R	**CRC 错误中断有效** 0：无效 1：中断有效
Reserved	Bit 15-13	—	保留
RXTHIV	Bit 12	R	**接收 FIFO 超过阈值中断有效** 0：无效 1：中断有效
RXUDIV	Bit 11	R	**接收 FIFO 欠载中断有效** 0：无效 1：中断有效
RXOVIV	Bit 10	R	**接收 FIFO 溢出中断有效** 0：无效 1：中断有效
RXFIV	Bit 9	R	**接收 FIFO 满中断有效** 0：无效 1：中断有效
Reserved	Bit 8-5	—	保留
TXTHIV	Bit 4	R	**发送 FIFO 低于阈值中断有效** 0：无效 1：中断有效
TXUDIV	Bit 3	R	**发送 FIFO 欠载中断有效** 0：无效 1：中断有效
TXOVIV	Bit 2	R	**发送 FIFO 溢出中断有效** 0：无效

			1：中断有效
Reserved	Bit 1	—	保留
TXEIV	Bit 0	R	发送 FIFO 空中断有效 0：无效 1：中断有效

26. 7. 2. 13 SPI原始中断标志状态寄存器（SPI_RIF）

SPI 原始中断标志状态寄存器（SPI_RIF）

偏移地址：30_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													FRERI	MODFRI	CRCERRRI	Reserved			RXTHRI	RXUDRI	RXOVRI	RXFRI	Reserved				TXTHRI	TXUDRI	TXOVRI	Reserved	TXERI

Reserved	Bit 31-19	—	保留
FRERI	Bit 18	R	**帧格式错误中断标志状态** 0：无效 1：原始中断标志状态
MODFRI	Bit 17	R	**模式故障中断标志状态** 0：无效 1：原始中断标志状态
CRCERRRI	Bit 16	R	**CRC 错误中断标志状态** 0：无效 1：原始中断标志状态
Reserved	Bit 15-13	—	保留
RXTHRI	Bit 12	R	**接收 FIFO 超过阈值中断标志状态** 0：无效 1：原始中断标志状态
RXUDRI	Bit 11	R	**接收 FIFO 欠载中断标志状态** 0：无效 1：原始中断标志状态
RXOVRI	Bit 10	R	**接收 FIFO 溢出中断标志状态** 0：无效 1：原始中断标志状态
RXFRI	Bit 9	R	**接收 FIFO 满中断标志状态** 0：无效 1：原始中断标志状态
Reserved	Bit 8-5	—	保留
TXTHRI	Bit 4	R	**发送 FIFO 低于阈值中断标志状态** 0：无效 1：原始中断标志状态
TXUDRI	Bit 3	R	**发送 FIFO 欠载中断标志状态** 0：无效 1：原始中断标志状态
TXOVRI	Bit 2	R	**发送 FIFO 溢出中断标志状态** 0：无效

			1：原始中断标志状态
Reserved	Bit 1	—	保留
TXERI	Bit 0	R	发送 FIFO 空中断标志状态 0：无效 1：原始中断标志状态

26. 7. 2. 14 SPI中断标志屏蔽状态寄存器（SPI_IFM）

SPI 中断标志屏蔽状态寄存器（SPI_IFM）

偏移地址：34_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													FREFM	MODFFM	CRCERRFM	Reserved			RXTHFM	RXUDFM	RXOVFM	RXFFM	Reserved				TXTHFM	TXUDFM	TXOVFM	Reserved	TXEFM

Reserved	Bit 31-19	—	保留
FREFM	Bit 18	R	**帧格式错误中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
MODFFM	Bit 17	R	**模式故障中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
CRCERRFM	Bit 16	R	**CRC 错误中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
Reserved	Bit 15-13	—	保留
RXTHFM	Bit 12	R	**接收 FIFO 超过阈值中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
RXUDFM	Bit 11	R	**接收 FIFO 欠载中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
RXOVFM	Bit 10	R	**接收 FIFO 溢出中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
RXFFM	Bit 9	R	**接收 FIFO 满中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
Reserved	Bit 8-5	—	保留
TXTHFM	Bit 4	R	**发送 FIFO 低于阈值中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
TXUDFM	Bit 3	R	**发送 FIFO 欠载中断标志屏蔽状态** 0：无效 1：中断标志屏蔽状态
TXOVFM	Bit 2	R	**发送 FIFO 溢出中断标志屏蔽状态** 0：无效

			1：中断标志屏蔽状态
Reserved	Bit 1	—	保留
TXEFM	Bit 0	R	发送 FIFO 空中断标志屏蔽状态 0：无效 1：中断标志屏蔽状态

26. 7. 2. 15 SPI中断清除寄存器（SPI_ICR）

SPI 中断清除寄存器（SPI_ICR）																															
偏移地址：38_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													FREIC	MODFIC	CRCERRIC	Reserved			RXTHIC	RXUDIC	RXOVIC	RXFIC	Reserved				TXTHIC	TXUDIC	TXOVIC	Reserved	TXEIC

Reserved	Bit 31-19	—	保留
FREIC	Bit 18	C_W1	**帧格式错误中断清除** 0：无效 1：中断清除
MODFIC	Bit 17	C_W1	**模式故障中断清除** 0：无效 1：中断清除
CRCERRIC	Bit 16	C_W1	**CRC 错误中断清除** 0：无效 1：中断清除
Reserved	Bit 15-13	—	保留
RXTHIC	Bit 12	C_W1	**接收 FIFO 超过阈值中断清除** 0：无效 1：中断清除
RXUDIC	Bit 11	C_W1	**接收 FIFO 欠载中断清除** 0：无效 1：中断清除
RXOVIC	Bit 10	C_W1	**接收 FIFO 溢出中断清除** 0：无效 1：中断清除
RXFIC	Bit 9	C_W1	**接收 FIFO 满中断清除** 0：无效 1：中断清除
Reserved	Bit 8-5	—	保留
TXTHIC	Bit 4	C_W1	**发送 FIFO 低于阈值中断清除** 0：无效 1：中断清除
TXUDIC	Bit 3	C_W1	**发送 FIFO 欠载中断清除** 0：无效 1：中断清除
TXOVIC	Bit 2	C_W1	**发送 FIFO 溢出中断清除** 0：无效

			1：中断清除
Reserved	Bit 1	—	保留
TXEIC	Bit 0	C_W1	发送 FIFO 空中断清除 0：无效 1：中断清除

第27章 通用异步收发器（UART）

27.1 概述

通用异步收发器（UART）提供了一个灵活的通信方式，使 MCU 可以与外部设备通过工业标准 NRZ 的形式实现全双工异步串行数据通讯。UART 可以使用小数波特率发生器，提供了超宽的波特率设置范围。

UART 支持异步通讯模式和单线半双工通讯，也支持 LIN（本地互联网络）、智能卡协议、IrDA（红外数据协会）SIR ENDEC 规范和 modem 流控操作（CTSn/RTSn），同时还支持多机通讯方式。

可以使用 DMA 实现多缓冲区设置，从而能够支持高速数据通讯。

27.2 特性

- 全双工异步通信
- 16-byte 接收和发送 FIFO
- 兼容 16C550 标准
- 可软件控制接收 FIFO 触发点
- 通信波特率可设置
- 十七个中断源
- 可与 DMA 使用
  - 利用 DMA 功能将收/发字节缓冲到保留的 SRAM 空间
- 内置小数波特率发生器，覆盖范围广，不需要特定值的外部晶体
  - 在时钟频率为 48 MHz 下，可编程收发波特率高达 3Mbps，最低可达 732.4bps
  - 在时钟频率为 4 MHz 下，可编程收发波特率高达 250Kbps，最低可达 61bps
- 支持自动硬件流控制/流控制功能（CTSn、RTSn），RTSn 控制流触发点可程序设计
  - Modem 硬件自动控制
  - RS485 发送使能控制
- 支持 CTSn 唤醒功能
- 支持 IrDA SIR 模式
  - 支持 3/16 位周期调制
- 支持 RS-485
  - 支持 9-位模式
  - 多处理器通信
- 完全可程序设计的串行接口特性
  - 可程序设计数据位个数，即 5，6，7，8，9 位，9 位用于 RS485 模式
  - 校验位：奇、偶、无校验
  - 停止位长度可程序设计：1，2 位，在智能卡模式中支持 0.5，1.5 位

- ◇ 可设置高位在前或低位在前

- ◆ 单线半双工通讯
- ◆ LIN 主机的断开信号发送能力和 LIN 从机的断开信号检测能力
  - ◇ 将 UART 设置为 LIN 模式时，有 13 位的断开信号发生器与断开信号检测功能
- ◆ 智能卡模式
  - ◇ 支持 ISO/IEC7816-3 标准定义的 T=0 和 T=1 智能卡异步协议
  - ◇ 智能卡使用的 1.5 停止位长度
- ◆ 噪声检测
- ◆ 支持 Modbus 协议

27. 3 结构框图

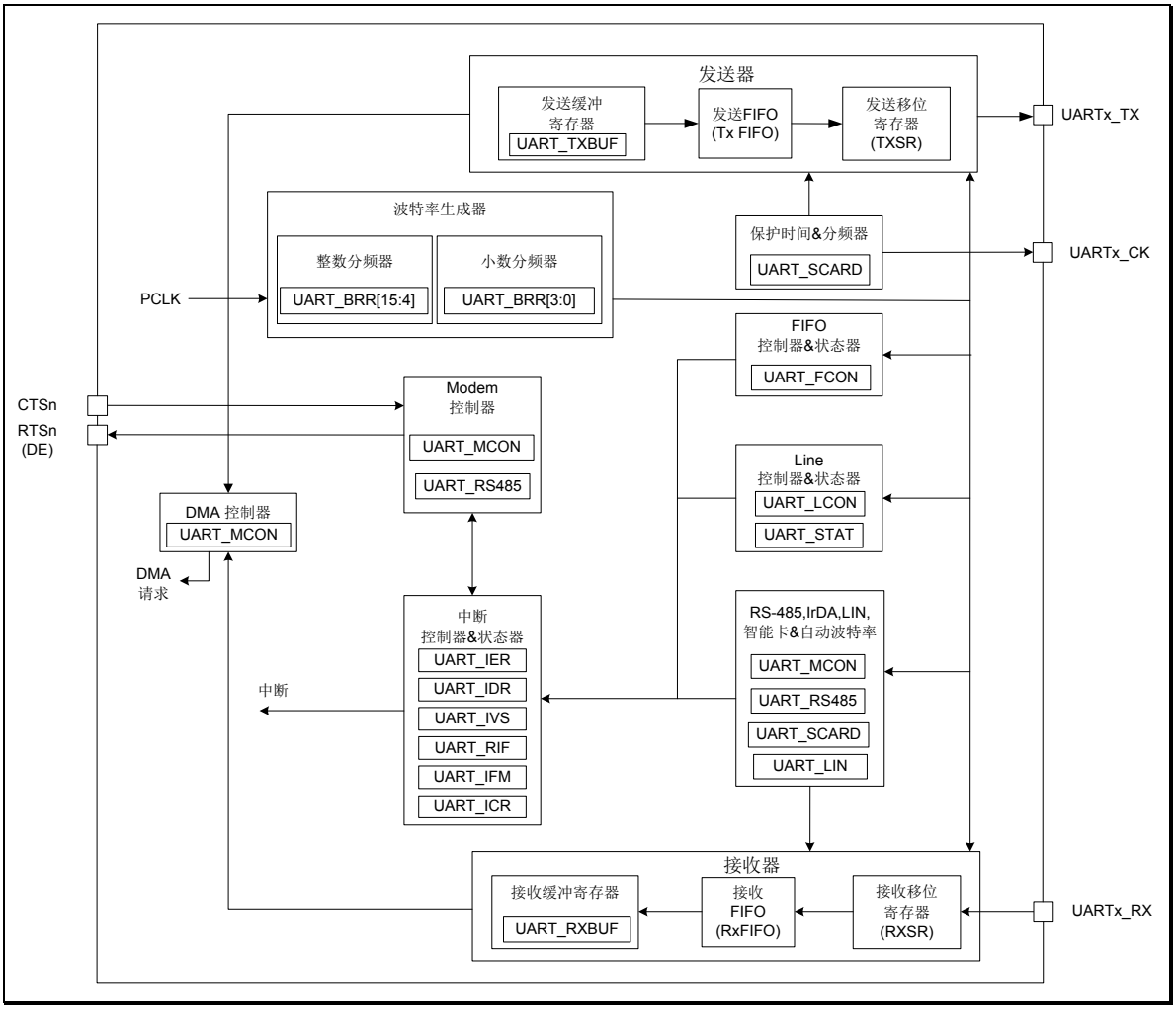


图 27-1 UART 结构框图

27. 4 功能描述

接口与外部设备通过三个引脚相连。任何 UART 双向通讯要求最少有两个引脚：接收数据输入（RX）和发送数据输出（TX）。

RX：接收数据输入,是串行数据的输入口。使用过采样技术来完成数据采集，以区别输入数据和噪声。

TX：数据发送输出。当发送器被禁止，输出脚回到其 I/O 口配置状态。当发送器被使能，但不发送数据时，TX 脚为高电平输出。在单线和智能卡模式中，这个引脚既用于发送数据也用于接收数据。

通过这些引脚，串行数据用数据帧的形式发送和接收：

- ◇ 在发送和接收之前为空闲状态
- ◇ 起始位
- ◇ 数据可通过 UART_LCON 寄存器的 MSB 位设定
- ◇ 1, 2 个停止位表明帧的结束（0.5, 1.5 个停止位用于智能卡模式）
- ◇ 一个状态寄存器（UART_STAT）
- ◇ 分开的接收和发送数据寄存器（UART_RXBUF, UART_TXBUF）
- ◇ 一个波特率寄存器（UART_BRR）12 位整数和 4 位小数。
- ◇ 一个智能卡寄存器（UART_SCARD）用于智能卡模式。
- ◇ 一个接收超时寄存器（UART_RTOR）检测输入信号时间并产生中断

下面的引脚在智能卡模式中会用到：

CK：时钟输出。智能卡模式中，CK 引脚会向智能卡提供时钟。

下列引脚用于支持硬件流控制模式：

CTSn：低电平发送，当高电平时作为发送阻塞信号。

RTSn：请求发送，表明 UART 已经准备好接收数据（低电平的时候）。

下列引脚在 RS485 驱动使能控制的时候会用到：

DE：驱动使能将外部收发器的发送模式激活。

注：DE 和 RTSn 共享同一个外部引脚。

27. 4. 1 数据长度

配置 LCON 寄存器中的 DLS 位可选择 8-5 位字长。

默认设置中，发送和接收的起始位都是低电平。而停止位都是高电平。

这个逻辑可以在 LCON 寄存器的 TXINV 与 RXINV 位设置为反向。

- ◇ 8 位字符宽度：DLS<1:0>=00
- ◇ 7 位字符宽度：DLS<1:0>=01
- ◇ 6 位字符宽度：DLS<1:0>=10
- ◇ 5 位字符宽度：DLS<1:0>=11

注：第 9 位使用于 RS485 多处理器模式。

空闲符号被视为完全由'1'组成的完整的数据帧，后面跟着包含了数据的下一帧的开始位（'1'的位数也包括了停止位的位数）。

断开符号被视为在一个帧周期内全部收到'0'（包括停止位期间，也是'0'）。在断开帧结束时，发送器会再插入 2 个停止位。

发送和接收由一个共享的波特率发生器驱动，当发送器和接收器的使能位分别置 1 时，分别为其产生时钟。

下面是每个字长的详细说明。

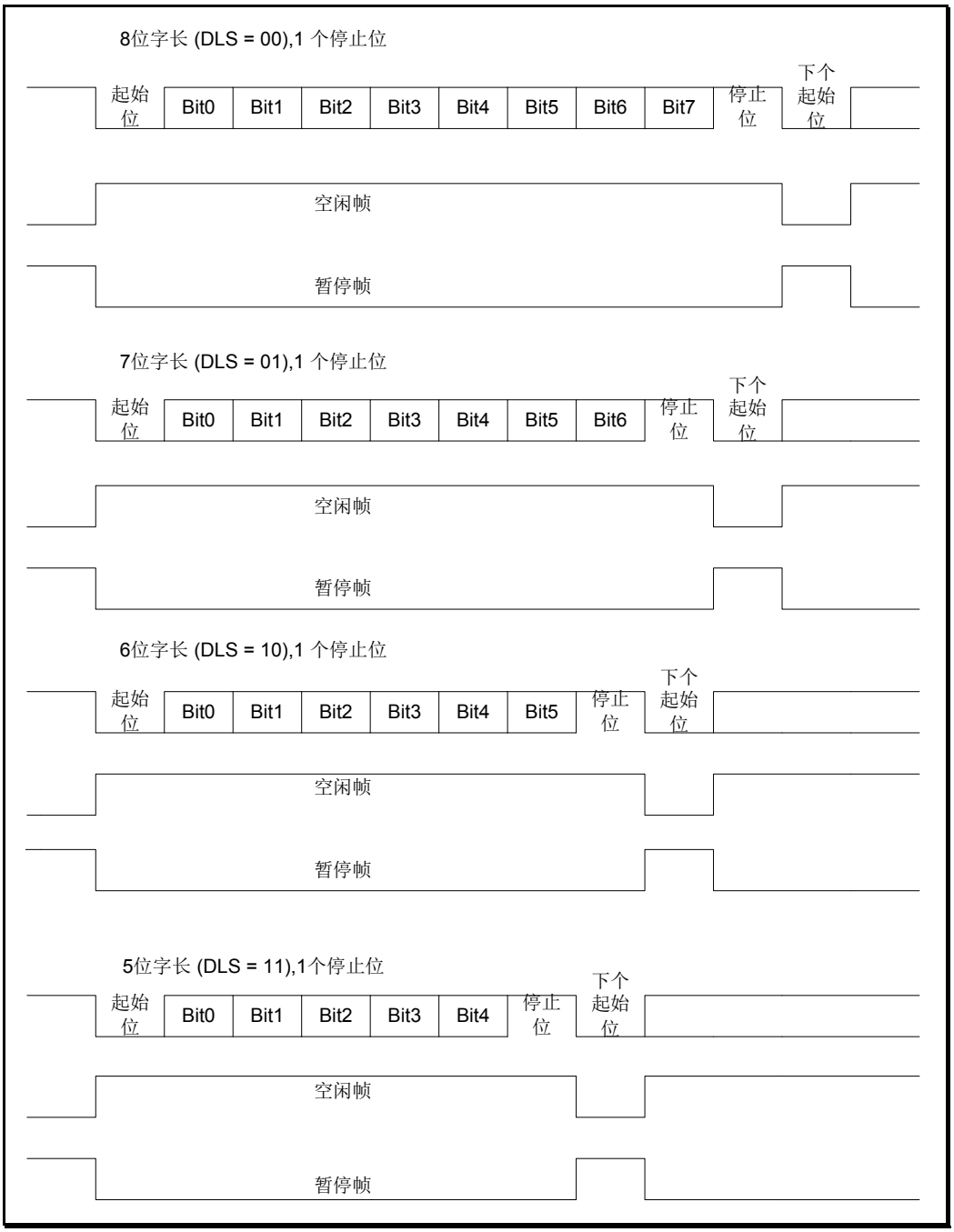


图 27-2 数据宽度设置

27. 4. 2 发送器

发送器根据 LCON 寄存器中的 DLS 位的状态发送 8-5 位的数据字。

当写入 UART_TXBUF 后，发送移位寄存器中的数据在 TX 脚上输出。

在 UART 发送期间，在 TX 引脚上首先移出数据的最低有效位。在此模式里，TXBUF 寄存器充当了一个内部总线和发送移位寄存器之间的缓冲器（TXSR）。

每个字符之前都有一个低电平的起始位，字符结束有停止位，停止位的数目可配置。

UART 支持多种停止位的选择：0.5,1,1.5 和 2 个停止位。

注 1：在写入 TXBUF 寄存器数据前必须先令 STAT 寄存器中的 TFFULL 位为 0。
注 2：打开 TX 开关后，数据才能在 TX 脚上输出可配置的停止位,随每个字符发送的停止位的位数可以通过 LCON 寄存器中的 STOP 位进行编程。

- ◇ 0.5 个停止位：在智能卡模式下发送和接收数据时使用。
- ◇ 1 个停止位：停止位的位数的默认值。
- ◇ 1.5 个停止位：在智能卡模式下发送和接收数据时使用。
- ◇ 2 个停止位：可用于常规 UART 模式、以及调制解调器模式。

空闲帧包括了停止位。

断开帧可通过 MCON 寄存器的 BKREQ 位产生 10 位低电平（当 DLS=00 时），9 位低电平（当 DLS=01 时），8 位低电平（当 DLS=10 时）或者 7 位低电平（当 DLS=11 时），后跟 2 个停止位。

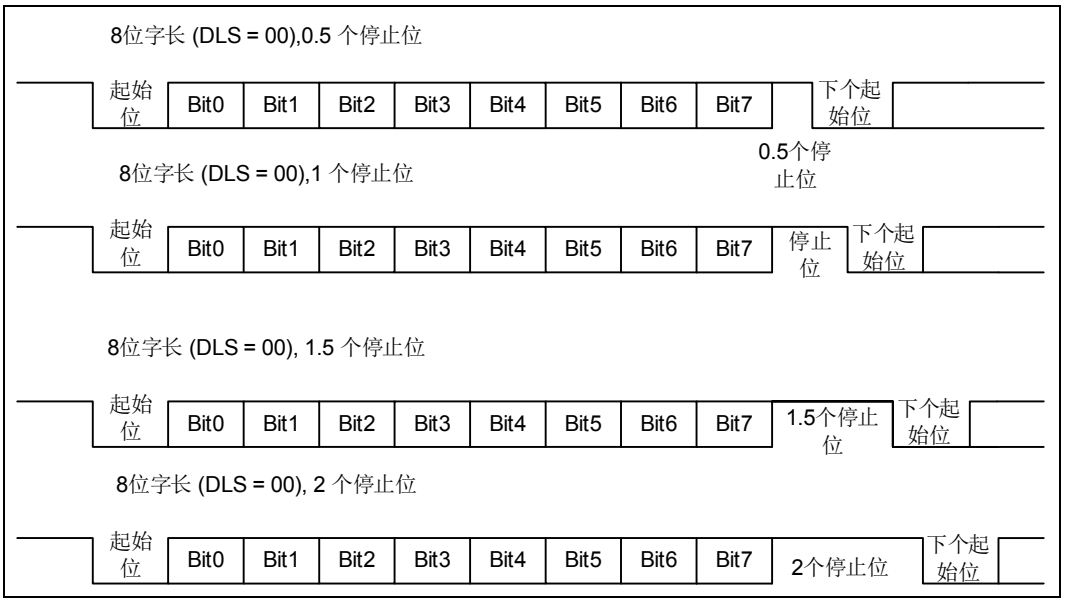


图 27-3 配置停止位

TX FIFO 阈值设置

设置 FCON 寄存器的 TXTH 位，可设定 FIFO 的阈值为 0,2,4,8，当 FIFO 内的数据个数小于阈值会使得 STAT 寄存器的 TFTH 位为 1，告知用户需要再填入数据，以避免数据传送中断。

开启 IER 寄存器的 TFTH 位为 1，UART 会判断 FIFO 内的个数是否小于阈值，使得 RIF 寄存器的 TFTH 位为 1，产生中断。在产生中断的期间设置 ICR 寄存器的 TFTH 位为 1，使得 RIF 寄存器的 TFTH 位清除，若使用者未写入新的数据至 FIFO 中，FIFO 内的数据个数仍然小于阈值则不会再次产生中断，需要重新开启 IER 寄存器的 TFTH 位。

注：一开始 RIF 寄存器的 TFTH 与 TFEMPTY 位为 0，当产生 FIFO 内的数据个数小于阈值事件与 FIFO 空事件时，使得 TFTH 与 TFEMPTY 位为 1。STAT 寄存器的 TFTH 与 TFEMPTY 位则是反映 TX FIFO 状态。

配置步骤：

1. 设置 LCON 寄存器中的 DLS 位来定义字长。
2. 设置 LCON 寄存器中的 STOP 位设置停止位的位数。
3. 设置 LCON 寄存器中的 PE 与 PS 位设置校验控制开关与极性。
4. 设置 BRR 寄存器选择希望的波特率。
5. 如果采用多缓冲器通信，配置 MCON 寄存器中的 TXDMAEN 位为 1。按多缓冲器通信中的描述配置 DMA 寄存器。
6. 设置 LCON 寄存器中的 TXEN 位，使能发送器。
7. 把要发送的数据写进 TXBUF 寄存器(此动作将清除 STAT 寄存器中的 TFEMPTY 位)。
8. 在 TXBUF 寄存器中写入数据字时，要等待 STAT 寄存器中的 TFFULL 位为 0，它表示 FIFO 中未满足 16byte。当需要关闭 UART，需要确认传输结束 STAT 寄存器中的 TSBUSY 位为 0，避免破坏最后一次传输。

注：当 LCON 寄存器的 TXEN 与 RXEN 位为 1 时，无法写入 LCON 寄存器与 BRR 寄存器。

27. 4. 3 接收器

27. 4. 3. 1 防抖电路

在UART_RX引脚上配置了一个防抖电路，设置LCON寄存器中的DBCEN位开启功能，输入信号须维持至少 8 个模块时钟的高电平或低电平，才能使得信号反映至 UART 中，反之则被忽略，如下叙述。

在下图中，SYNC0 是指输入信号由模块时钟一次采样；SYNC1 是指 SYNC0 被模块时钟一次采样；SYNC2 表示 SYNC1 由模块时钟一次采样。SYNC0、SYNC1 和 SYNC2 可以表示成 x[n]×Ts，x[n+1]×Ts 和 x[n+2]×Ts。Ts 是模块时钟周期，n 是采样时间。如果关闭防抖模块，时间就可以表示为 x[n+1]×Ts。

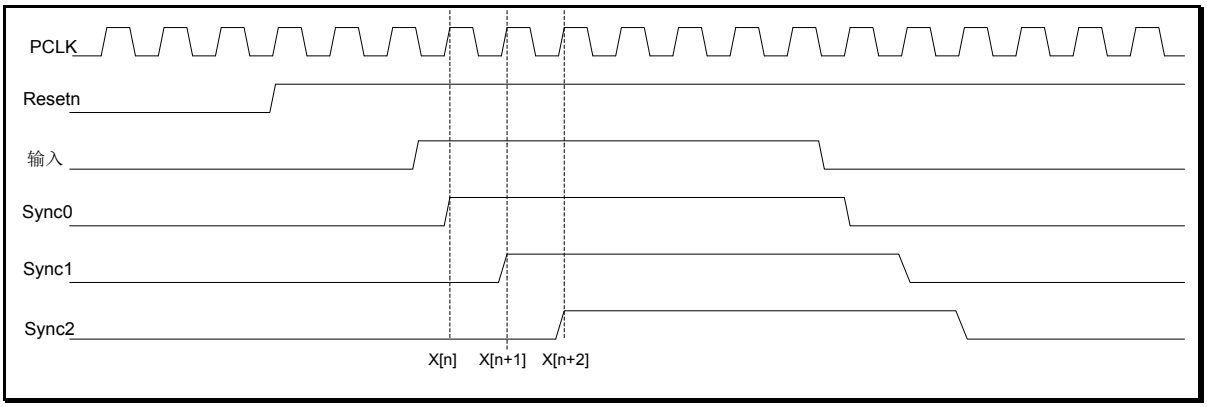


图 27-4 防抖动波形

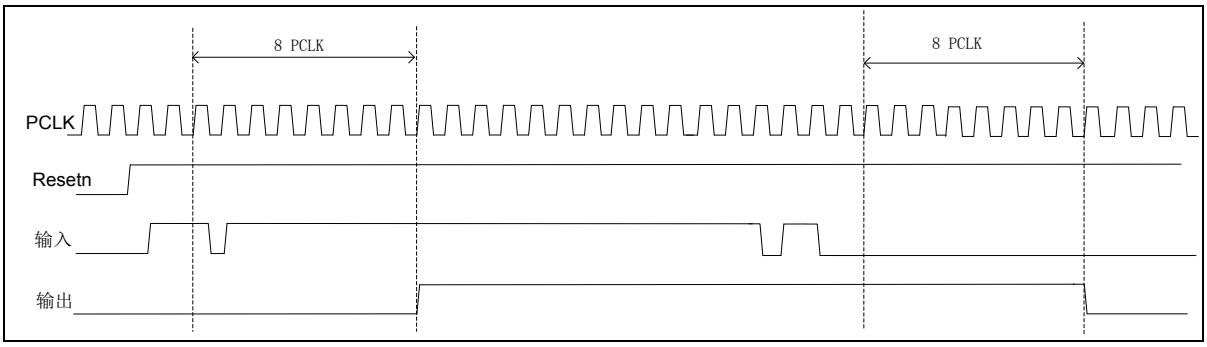


图 27-5 防抖动输出

27. 4. 3. 2 起始位检测

接收器根据 LCON 寄存器中 DLS 位的状态接收 8-5 位的数据字。

起始位检测

在 UART 中，如果辨认出一个特殊的采样序列，那么就认为检测到一个起始位。

该序列为：1 1 1 0 X 0 X 0 X 0 0 0 0 X X X X X X X

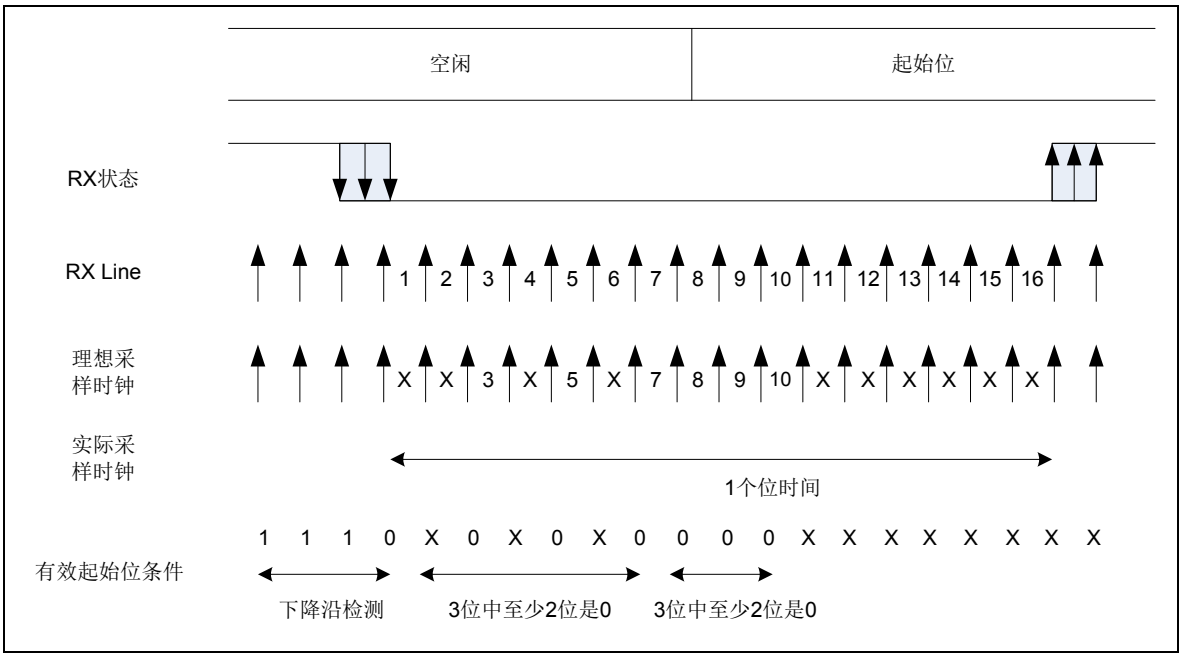


图 27-6 起始位检测

注 1：如果该序列不完整，那么接收端将退出起始位检测并回到空闲状态（不设置标志位）开始等待下降沿。

注 2：如果 3 个采样点都为'0'（在第 3、5、7 位的第一次采样，和在第 8、9、10 的第二次采样都为'0'），则确认收到起始位。

注 3：如果两次 3 个采样点上有 2 个是'0'（第 3、5、7 位的采样点和第 8、9、10 位的采样点），那么起始位仍然是有效的。如果不能满足这个条件，则中止起始位的检测过程，接收器会回到空闲状态。

注 4：如果两次 3 个采样点上有 2 个是'1'（第 3、5、7 位的采样点和第 8、9、10 位的采样点），那么起始位是无效的，将退出起始位检测并回到空闲状态，并会设置噪声标志位。

RX FIFO 阈值设置

设置 FCON 寄存器的 RXTH 位，可设定 FIFO 的阈值为 1,4,8,14，当 FIFO 内的数据个数大于或等于阈值会使得 STAT 寄存器的 RFTH 位为 1，告知用户需要读取数据，以避免数据遗失。

开启 IER 寄存器的 RFTH 位为 1，UART 会判断 FIFO 内的个数是否大于或等于阈值，使得 RIF 寄存器的 RFTH 位为 1,产生中断。在产生中断的期间设置 ICR 寄存器的 RFTH 位为 1，使得 RIF 寄存器的 RFTH 位清除，若用户未读取数据，FIFO 内的数据个数仍然大于或等于阈值则不会再次产生中断，需要重新开启 IER 寄存器的 RFTH 位。

注：一开始 RIF 寄存器的 RFTH 位为 0，当产生 FIFO 内的数据个数大于或等于阈值事件时，使得 RFTH 位为 1。STAT 寄存器的 RFTH 位则是反映 RX FIFO 状态。

配置步骤：

1. 设置 LCON 寄存器中的 DLS 位来定义字长。
2. 设置 LCON 寄存器中的 STOP 位设置停止位的位数。
3. 设置 LCON 寄存器中的 PE 与 PS 位设置校验控制开关与极性。
4. 设置 BRR 寄存器选择希望的波特率。
5. 如果采用多缓冲器通信，配置 MCON 寄存器中的 RXDMAEN 位为 1。按多缓冲器通信中的描述配置 DMA 寄存器。
6. 设置 LCON 寄存器中的 RXEN 位，这将启动接收器，使它开始寻找起始位。

当一个字符被接收到时，STAT 寄存器的 RFEMPTY 位被置 0。它表明移位寄存器的内容被转移到 RX FIFO 中。换句话说，数据已经被接收并且可以被读出（包括与之有关的错误标志）。

这时如果 IER 寄存器的 RFTH 位是 1，且 RX FIFO 阈值为 1，将会引起中断请求。

在接收期间如果检测到帧错误，噪音或溢出错误，错误标志将被置起。RXBERR 标志也会和 RFTH 一起被置 1。

在多缓冲器通信时，RFEMPTY 在每个字节接收后被清零，并由 DMA 对数据寄存器的读操作而置起。

RFFULL 位必须在下一字符接收结束前被清零，以避免溢出错误。

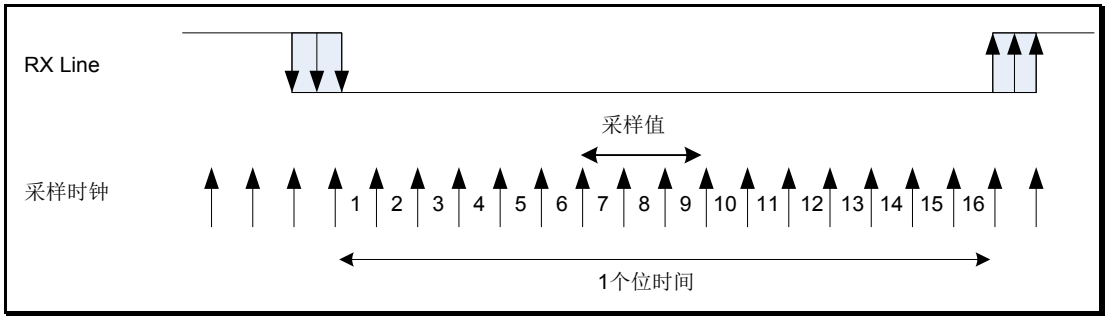


图 27-7 数值采样

帧错误

当以下情况发生时检测到帧错误：

由于没有同步上或大量噪音的原因，停止位没有在预期的时间上接收和识别出来。

当帧错误被检测到时：

1. FERR 位被硬件置 1
2. 此时读取的 RXBUF 寄存器数据可能有错。
3. 寄存器 STAT 中的 FERR 位显示当前从 FIFO 读取的 RXBUF 寄存器是否为帧错误。
4. 寄存器 RIF 中的 RXBERR 位则会在接收的过程中被置起，若 IER 中的 RXBERR 位为 1 则会产生中断。

奇偶位错误

当以下情况发生时检测到奇偶性错误：

由于没有同步上或大量噪音的原因，奇偶位没有在预期的时间上接收和识别出来。

当奇偶位错误被检测到时：

1. PERR 位被硬件置 1
2. 此时读取的 RXBUF 寄存器数据可能有错。
3. 寄存器 STAT 中的 PERR 位显示当前从 FIFO 读取的 RXBUF 寄存器是否为校验错误。
4. 寄存器 RIF 中的 RXBERR 位则会在接收的过程中被置起，若 IER 中的 RXBERR 位为 1 则会产生中断。

断开错误

当以下情况发生时检测到断开错误：

由于没有同步上或大量噪音的原因，字符与字符停止位为 0 时没有在预期的时间上接收和识别出来。

当断开错误被检测到时：

1. BKERR 位被硬件置 1
2. 此时读取的 RXBUF 寄存器数据可能有错。
3. 寄存器 STAT 中的 BKERR 位显示当前从 FIFO 读取的 RXBUF 寄存器是否为断开错误。
4. 这个错误并不会产生中断。

溢出错误

当以下情况发生时检测到溢出错误：

由于 FIFO 已满 16byte 还没有被读取，而又接收到一个字符，则发生溢出错误。

当溢出错误被检测到时：

1. RFOERR 位被硬件置 1
2. RXBUF 内容将不会丢失，读取 RXBUF 寄存器仍能得到先前的数据。
3. 移位寄存器中以前的内容将被覆盖，随后接收的数据将丢失。
4. 寄存器 RIF 中的 RFOERR 位则会在接收的过程中被置起，若 IER 中的 RFOERR 位为 1 则会产生中断。

采样值	接收到的值
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	1

表 27-1　来自采样数据的噪音检测

27. 4. 4　状态寄存器

在 UART 中配置了 15 种 UART 状态提供使用者使用，叙述如下

◇ PERR（Parity error）

- 当所接收的字符没有正确的校验字节，将生成一个奇偶校验错误。该错误与 FIFO 顶部的字符有关，即读取 RXBUF 寄存器的字符。

◇ FERR（Framing error）

- 当接收到的字符停止位为 0 时，产生帧错误。该错误与 FIFO 顶部的字符有关，即读取 RXBUF 寄存器的字符。

◇ BKERR（Break error）

- 当接收到的字符与字符停止位为 0 时，产生断开错误。该错误与 FIFO 顶部的字符有关，即读取 RXBUF 寄存器的字符。

◇ CTSSTA（CTSn status error）

- 清除发送。此位为 CTSn pin 上的状态。当 CTSSTA 位为 0，这是一个迹象表明调制解调器和数据设备已准备好与 UART 进行数据交换。

◇ RSBUSY（RX shift register busy）

- 当此位为 1 表示接收器正在接收字符，为 0 则为接收完成。

◇ RFTH（RX FIFO 阈值）

- 当此位为 1 表示 RX FIFO 内数据大于或等于阈值（设置 FCON 寄存器的 RXTH，

可设定阈值为 1,4,8,14），告知使用者需要读取 RXBUF 寄存器。

◇ RFEMPTY（RX FIFO empty）

- 当此位为 1 表示 RX FIFO 内无任何字符，为 0 则为已接收 1 个以上的字符。

◇ RFFULL（RX FIFO full）

- 当此位为 1 表示 RX FIFO 内已有 16 个字符，需要被读取。

◇ RFOERR（RX FIFO overrun）

- 当此位为 1 表示 RX FIFO 内已有 16 个字符，且又再接收 1 个字符，此时 FIFO 内字符不会丢失，接收的字符则会被丢失。

◇ RFUERR（RX FIFO underrun）

- 当此位为 1 表示 RX FIFO 内无任何字符，且又被读取。

◇ TSBUSY（TX shift register busy）

- 当此位为 1 表示发送器正在传送字符，为 0 则为传送完成，当写入第一个数据至 TXBUF 寄存器就会使得 TSBUSY 位为 1。

◇ TFTH（TX FIFO 阈值）

- 当此位为 1 表示 TX FIFO 内数据小于阈值（设置 FCON 寄存器的 TXTH，可设定阈值为 0,2,4,8），告知使用者需要写入 TXBUF 寄存器。

◇ TFEMPTY（TX FIFO empty）

- 当此位为 1 表示 TX FIFO 内无任何字符，为 0 则为准备发送 1 个以上的字符。

◇ TFFULL（TX FIFO full）

- 当此位为 1 表示 TX FIFO 内已有 16 个字符，准备发送。

◇ TFOERR（TX FIFO overrun）

- 当此位为 1 表示 TX FIFO 内已有 16 个字符，且又在写入 1 个字符，此时 FIFO 内字符不会丢失，写入的字符则会被丢失。

27. 4. 5 波特率生成器

接收器和发送器的波特率在 UARTDIV 的整数和小数寄存器中的值应设置成相同。

UARTDIV=UART_BRR.BRR.

TX/RX baud = PCLK / UARTDIV

注 1：当 LCON 寄存中的 RXEN 与 TXEN 为 1 时，BRR 寄存器无法被写入。 注 2：当 BRR<15:4>=0 时，无法运行。

从 UART_BRR 寄存器值中推断出 UART 波特率

◇ 在 4 MHz 下，为了得到 115200 波特率

- UARTDIV = 4000000/115200 = 34.7
- BRR<15:0> = UARTDIV = 35d = 23h（四舍五入）

◇ 在 8 MHz 下，为了得到 9600 波特率

- UARTDIV = 8000000/9600 = 833.33
- BRR<15:0> = UARTDIV = 833d = 341h（四舍五入）

◇ 在 16 MHz 下，为了得到 1200 波特率

- UARTDIV = 16000000/1200 = 13333.33
- BRR<15:0> = UARTDIV = 13333d = 3415h（四舍五入）

◇ 在 24 MHz 下，为了得到 460800 波特率

- UARTDIV = 24000000/460800 = 52.08
- BRR<15:0> = UARTDIV = 52d = 34h（四舍五入）

◇ 在 48 MHz 下，为了得到 115200 波特率

- UARTDIV = 48000000/115200 = 416.66
- BRR<15:0> = UARTDIV = 417d = 1A1h（四舍五入）

◇ 在 48 MHz 下，为了得到 921600 波特率

- UARTDIV = 48000000/921600 = 52.08
- BRR<15:0> = UARTDIV = 52d = 34h（四舍五入）

波特率	16 倍过采样			
序号	预期值	实际值	BRR	%误差
1	734Bps	733Bps	0xFFCC	0.14
2	1.2KBps	1.2KBps	0x9C40	0
3	2.4KBps	2.4KBps	0x4E20	0
4	4.8KBps	4.8KBps	0x2710	0
5	9.6KBps	9.6KBps	0x1388	0
6	19.2KBps	19.2KBps	0x9C4	0
7	38.4KBps	38.4KBps	0x4E2	0
8	57.6KBps	57.62KBps	0x341	0.03
9	115.2KBps	115.11KBps	0x1A1	0.08
10	230.4KBps	230.77KBps	0xD0	0.16
11	460.8KBps	461.54KBps	0x68	0.16
12	921.6KBps	923.07KBps	0x34	0.16
13	1.5MBps	1.5MBps	0x20	0
14	2MBps	2MBps	0x18	0
15	3MBps	3MBps	0x10	0

表 27-2 时钟为 48MHz 下，设置波特率时的误差计算

27.4.6 自动波特率检测

UART 可以根据接收到的一个字符来检测和自动设置 BRR 寄存器的值。自动波特率检测在两种情况下有用：

- ◇ 通讯速度不可知的情况下
- ◇ 使用低精度时钟源，需要在不测量时钟偏差的条件下纠正波特率的时候。

时钟源的频率必须和预期的波特率保持相对的稳定（过采样率为 16，并且波特率处于 PCLK/65535 和 PCLK/16 之间）。

在打开自动波特率检测之前，字符的内容必须先确认。有三个可能的字符内容，能够通过 MCON 寄存器中的 ABRMOD 位进行选择。具体是：

- ◇ 模式 0（0x00）：波特率在 UART 的 RX 引脚的两个连续的下降沿上测量（起始位的下降沿和最低数据位的下降沿（LSB））
- ◇ 模式 1（0x01）：波特率在 UART 的 RX 引脚下降沿和后续上升沿之间（起始位的长度）测量。
- ◇ 模式 2（0x10）：波特率在 UART 的 RX 引脚下降沿和后续上升沿之间（起始位的长度加上 bit<0>的长度）测量（e.g. 0xFE）。

将 MCON 寄存器中的 ABREN 位置 1，开启自动波特率检测功能。UART 在 RX 线上等待第一个字符过来。当自动波特率操作结束后，RIF 寄存器中的 ABEND 标志会被硬件自动设置 1，若 IER 寄存器中的 ABEND 位为 1 则会产生中断。

如果线路噪声严重，不能保证得到的波特率是准确的。这时 BRR 值可能是错的或者 ABEND 错误标志会被置 1。在通讯速度超出自动波特率检测范围（位长度不在 0x10 到 0x1FFFF 个时钟周期之间）时也会发生这种情况。

在此模式中配置了两个中断，分别为检测超时与检测结束。在软件并未清除 MCON 寄存器中的 ABREN 的情况下，UART 计数器会持续计数直到 0x1FFFF 后，硬件会自动将 ABREN 清除，并将 RIF 寄存器中的 ABTO 位置起，如果开启中断，则会产生 ABTO 中断。

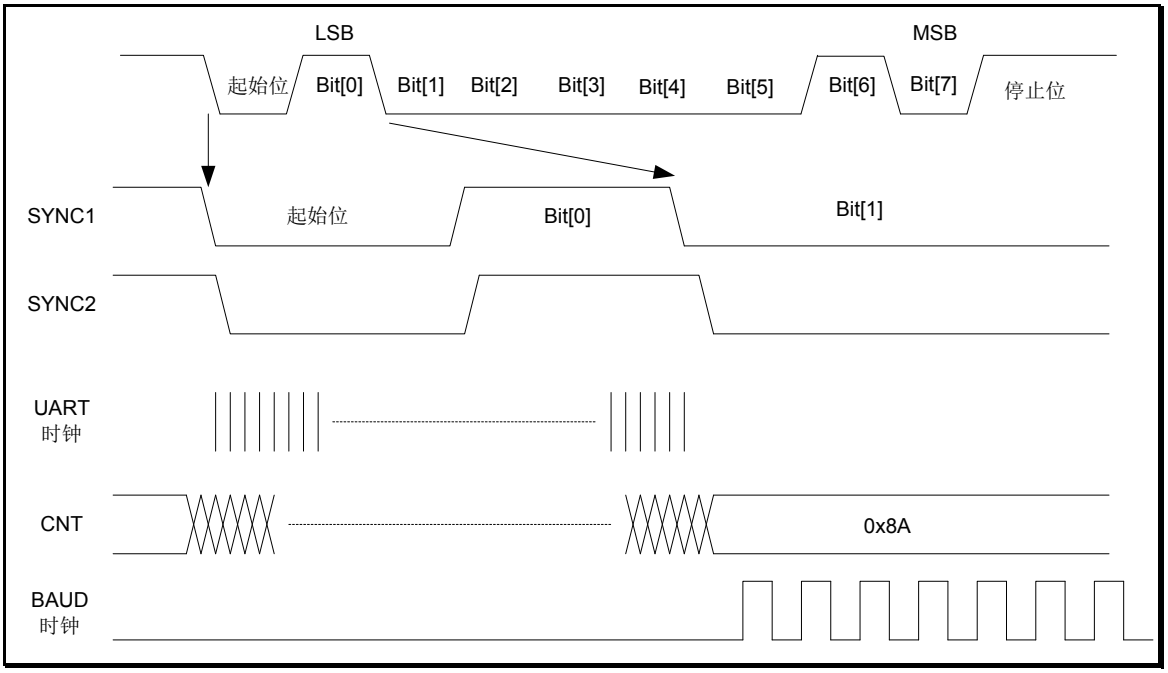


图 27-8　自动波特率检测模式 0

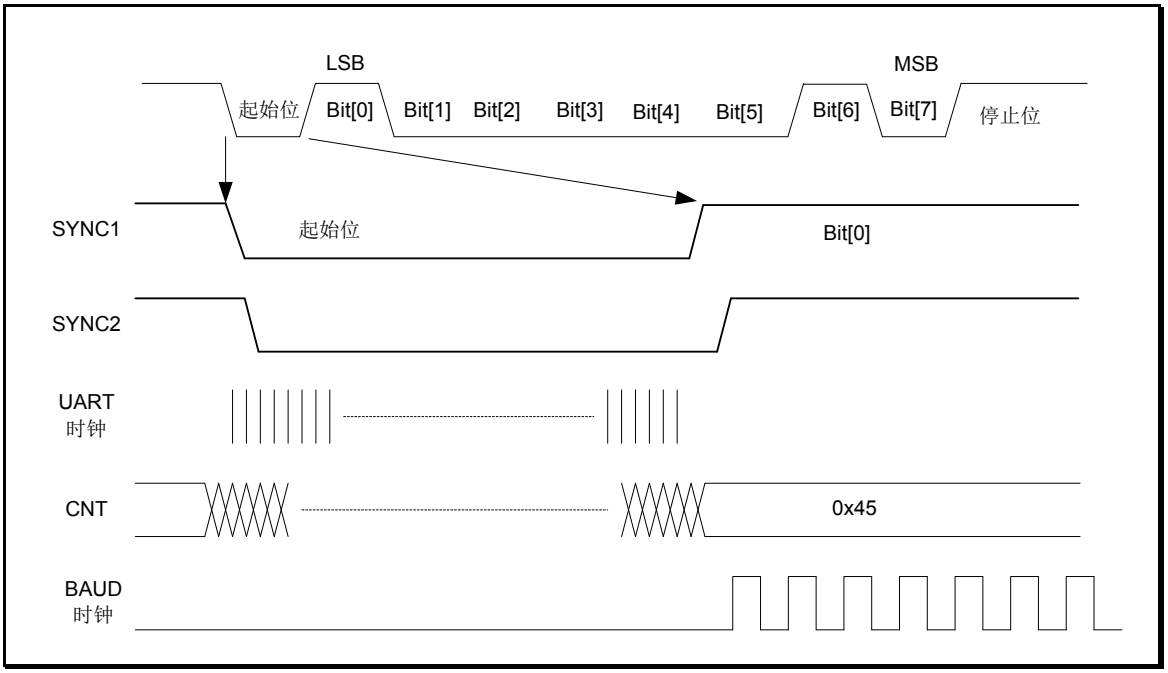


图 27-9　自动波特率检测模式 1

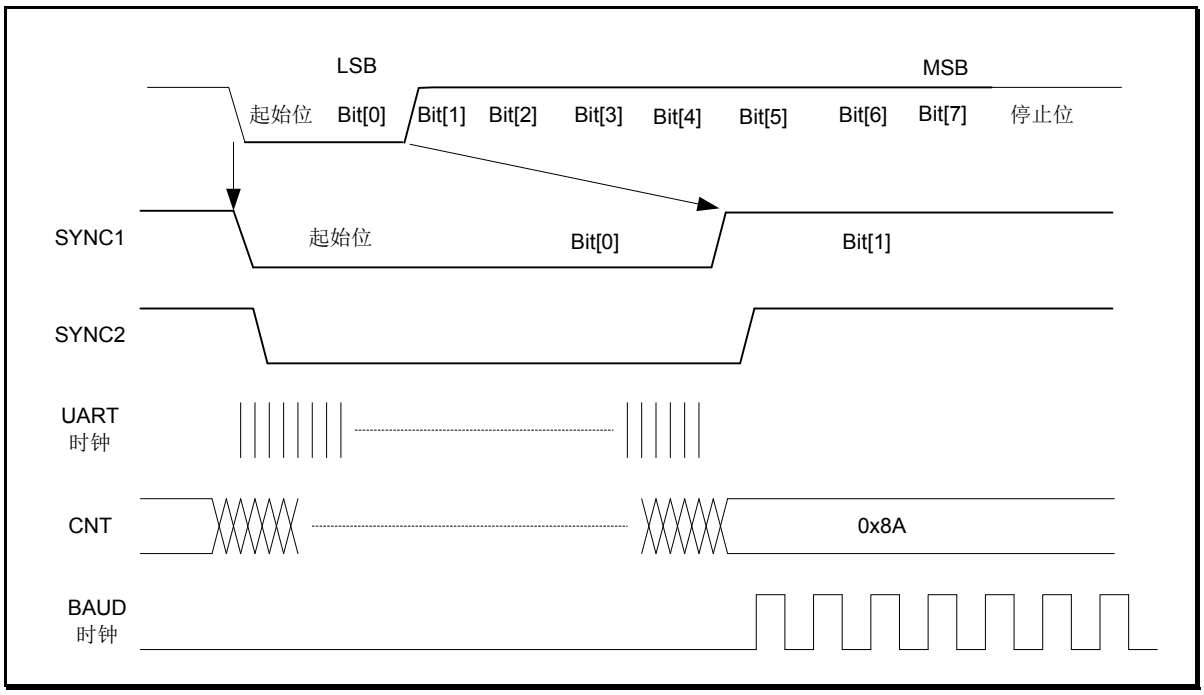


图 27-10　自动波特率检测模式 2

27.4.7 自动流控制

如果使能自动流控制，接收 FIFO 和发送 FIFO 会通过 UARTx_RTSn 和 UARTx_CTSn 引脚去控制 UART 的接收（RX）和发送（TX）。

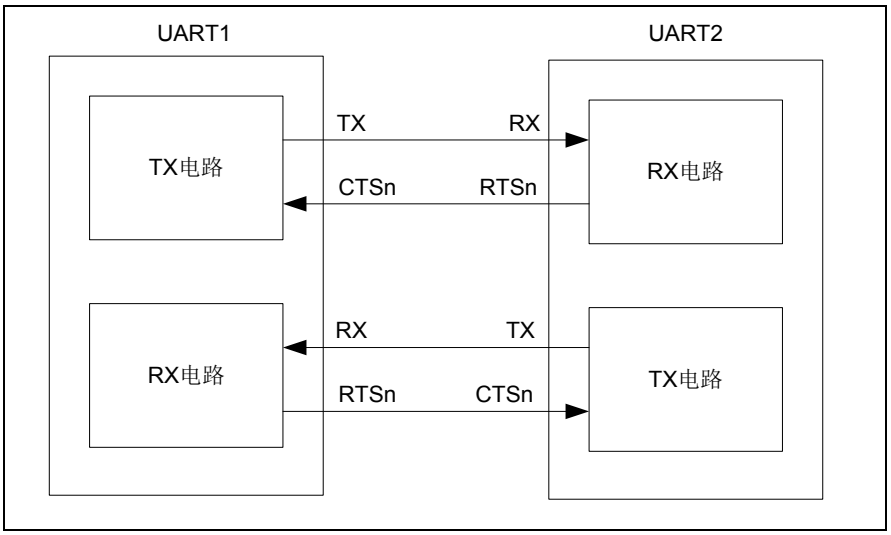


图 27-11 自动流控制框图

27.4.7.1 RTSn控制

当自动 RTS 被使能时，接收 FIFO 达到由 UART_FCON 寄存器设置的阈值，UART_RTSn 输出会拉为高电平。当 UART_RTSn 连接到另一个 UART 设备的 UART_CTSn 输入，另一个 UART 会停止发送串行数据，直到接收 FIFO 完全是空的。

可选择接收 FIFO 阈值的值是：1，4，8，14 个字符。一个额外的字符有可能会在 UART_RTSn 成为无效后传送到 UART（由于从 UART 进入发送器的数据尚未发送完成），阈值设置为 14 个字符能最大限度的使用 FIFO。

硬件通过读取接收缓冲寄存器 UART_RXBUF，一旦得知接收 FIFO 完全为空时，UART_RTSn 变低电平，通知其他 UART 继续发送数据。

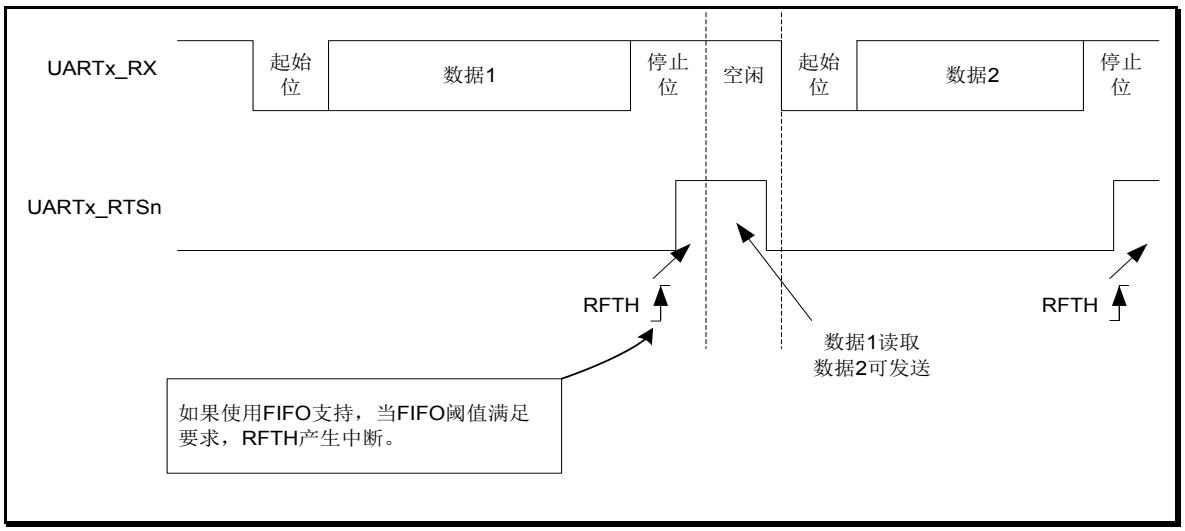


图 27-12 自动 RTSn 控制

27. 4. 7. 2 CTSn控制

当自动 CTS 启用，每当 UART_CTSn 输入变为高电平时，UART 发送器被禁用。这可以防止接收端 UART 的 FIFO 溢出。在最后一个停止位发出时，假设 UART_CTSn 依然为低电平，则发送器会在禁用前继续传输一个字符。在发送器被禁用时，发送 FIFO 仍然可以被写入，甚至溢出。

只要 UART_CTSn 输入一变换状态，硬件就自动设置 RIF 寄存器中的 DCTS 位。它表明接收器是否准备好进行通信。如果开启中断，则会产生 DCTS 中断。

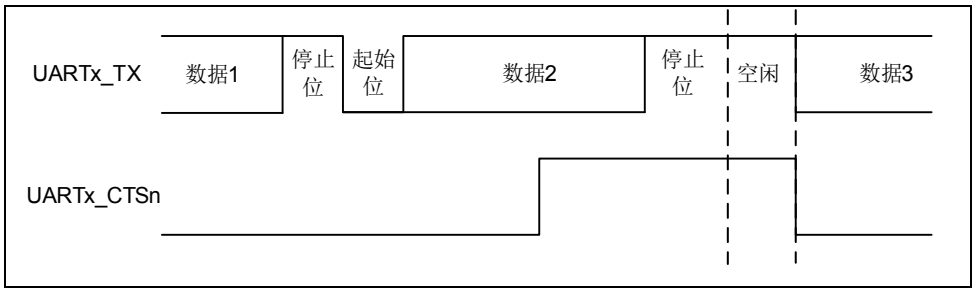


图 27-13 自动 CTSn 控制

自动流控制可以减少系统中断，当自动流控制使能，CTSn 状态不会触发系统中断。因为是设备自动地控制其收发器。

27. 4. 7. 3 RS485 驱动使能（DE）

当 RS485 驱动使能功能启用。它允许用户通过 DE（驱动使能）信号来激活外部收发器的控制端。滞后时间是一个发送消息的最后一个字节的停止位和释放 DE 信号之间的时间间隔，这个时间可以在 RS485 寄存器中的 DLY 位设置。DE 信号的极性则可以通过 RS485 寄存器中的 AADINV 位中设置并进行选择。

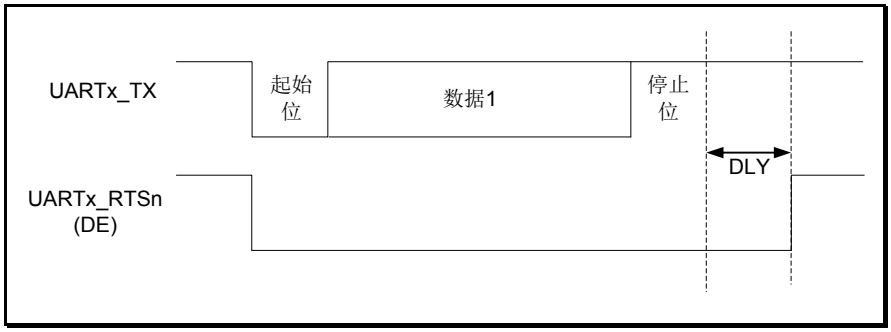


图 27-14 驱动使能当 AADINV=0

27.4.8 校验控制

设置 LCON 寄存器中的 PE 位，可以使能校验控制（发送时生成一个校验位，接收时进行校验检查）。根据 DLS 位定义的帧长度，可能的 UART 帧格式列在下表中。

DLS<1:0>	PE	UART 帧
00	0	起始位→8 位数据→停止位
00	1	起始位→8 位数据→校验位→停止位
01	0	起始位→7 位数据→停止位
01	1	起始位→7 位数据→校验位→停止位
10	0	起始位→6 位数据→停止位
10	1	起始位→6 位数据→校验位→停止位
11	0	起始位→5 位数据→停止位
11	1	起始位→5 位数据→校验位→停止位

表 27-3 帧格式

◇ 奇校验

校验位的内容使得一帧中的 8，7，6 或 5 个 LSB 数据以及校验位中 1 的个数为奇数。

例如：数据=00110101，有 4 个 1，如果选择奇校验（在 LCON 寄存器中的 PS＝0），校验位将是 1。

◇ 偶校验

校验位的内容使得一帧中的 8，7，6 或 5 个 LSB 数据以及校验位中 1 的个数为偶数。

例如：数据=00110101，有 4 个 1，如果选择偶校验（在 LCON 寄存器中的 PS＝1），校验位将是 0。

◇ 接收时的校验检查

如果校验检查失败，STAT 寄存器中的 PERR 标志会被置 1，如果 IER 寄存器中的 RXBERR 为 1，将引发相应中断。

◇ 发送时的校验生成

如果 LCON 寄存器的 PE 位被置 1，写进数据寄存器的数据的 MSB 位被校验位替换后发送出去（如果选择奇校验奇数个 1，如果选择偶校验偶数个 1）。

27.4.9 多处理器通信

设置 DLS 位为 8 位字长（第 9 位为判断地址或数据）

设置 RS485 寄存器的 AADEN 位为 1 以进入模式。

设置 RS485 寄存器的 ADDR 位配置匹配地址

可以将多个 UART 连接成一个网络来实现多机通讯。例如某个 UART 设备可以是主，它的 TX 输出和其他 UART 从设备的 RX 输入相连接；UART 从设备各自的 TX 输出逻辑地与在一起，并且和主设备的 RX 输入相连接。

在多处理器配置中，通常希望只有被寻址的接收者才被激活，来接收随后的数据，这样就可以减少由未被寻址的接收器的参与带来的多余的 UART 服务开销。

未被寻址的设备可启用其静默功能进入静默模式。要使用静默模式功能，RS485 寄存器的 AADEN 位必须被置 1。

在这个模式里，如果 MSB 是 1，该字节被认为是地址，否则被认为是数据。在一个地址字节中，目标接收器的地址被放在 RS485 寄存器的 ADDR 位。

如果接收到的字节与它的编程地址不匹配时，UART 进入静默模式。当 UART 进到静默模式后，接收字节时既不会改变 RIF 寄存器的 RFTH 位标志也不会产生中断或发出 DMA 请求。

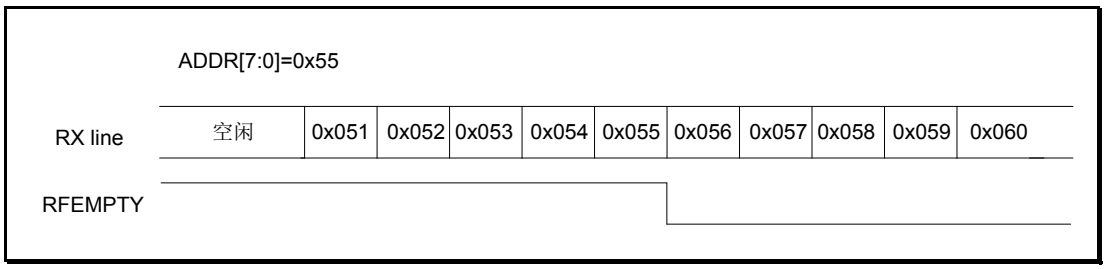


图 27-15 使用地址标示检测模式

27. 4. 10 LIN模式

- LIN 发送

  和常规的 UART 发送相同，但包含下列区别：

  - 设置 DLS 位为 8 位字长
  - 设置 LIN 寄存器的 LINEN 位为 1 以进入 LIN 模式。这时，置 LIN 寄存器的 LINBKREQ 位为 1 将发送 13 位 0 作为断开符号。然后发两位 1，以允许对下一个开始位的检测。

- LIN 接收

  当 LIN 模式被使能时（LIN 寄存器的 LINEN 位为 1），断开符号检测电路被激活。该检测完全独立于 UART 接收器。不管是在总线空闲时还是在发送某数据帧期间，断开符号只要一出现就能检测到。

  一旦接收器被激活（LCON 寄存器的 RXEN 位为 1），电路就开始监测 RX 上的起始信号。监测起始位的方法同检测断开符号或数据是一样的。当起始位被检测到后，电路对每个接下来的位，在每个位的第 8，9，10 个过采样时钟点上进行采样，就像针对数据一样。如果 10 个（当 LIN 寄存器中的 LINBDL=0）或 11 个（当 LIN 寄存器中的 LINBDL=1）连续位都是 0，并且又跟着一个定界符，RIF 寄存器的 LINBK 位标志就会被置 1。如果 IER 寄存器的 LINBK 位为 1，还会产生中断。在确认断开符号前，要检查定界符，因为它表示 RX 线已经回到高电平。如果在第 10 或 11 个采样点之前采样到了 1，检测电路取消当前检测并重新寻找起始位。如果 LIN 模式被禁止，接收器继续如正常 UART 那样工作，不再考虑检测断开符号。如果 LIN 模式被激活（LINEN=1），只要一发生帧错误（例如：停止位检测到 0，这种情况出现在断开符号被接收到的时候），接收器就停止，直到断开符号检测电路接收到一个 1（这种情况发生于断开符号没有完整的发出来），或一个定界符（这种情况发生于已经检测到一个完整的断开符号）。

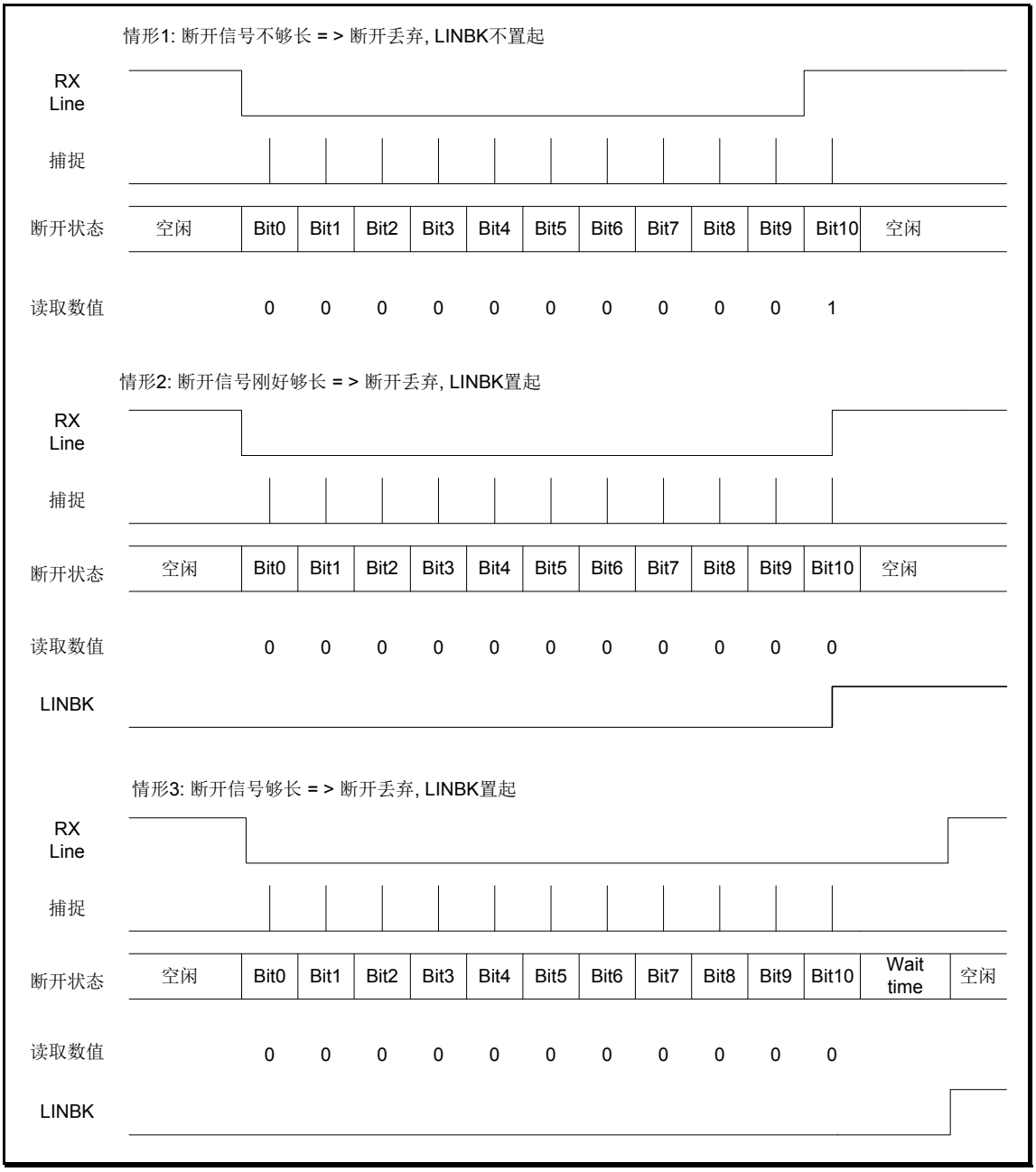


图 27-16　LIN 模式下断开信号检测（11 位断开长度–LBDL 位为 1）

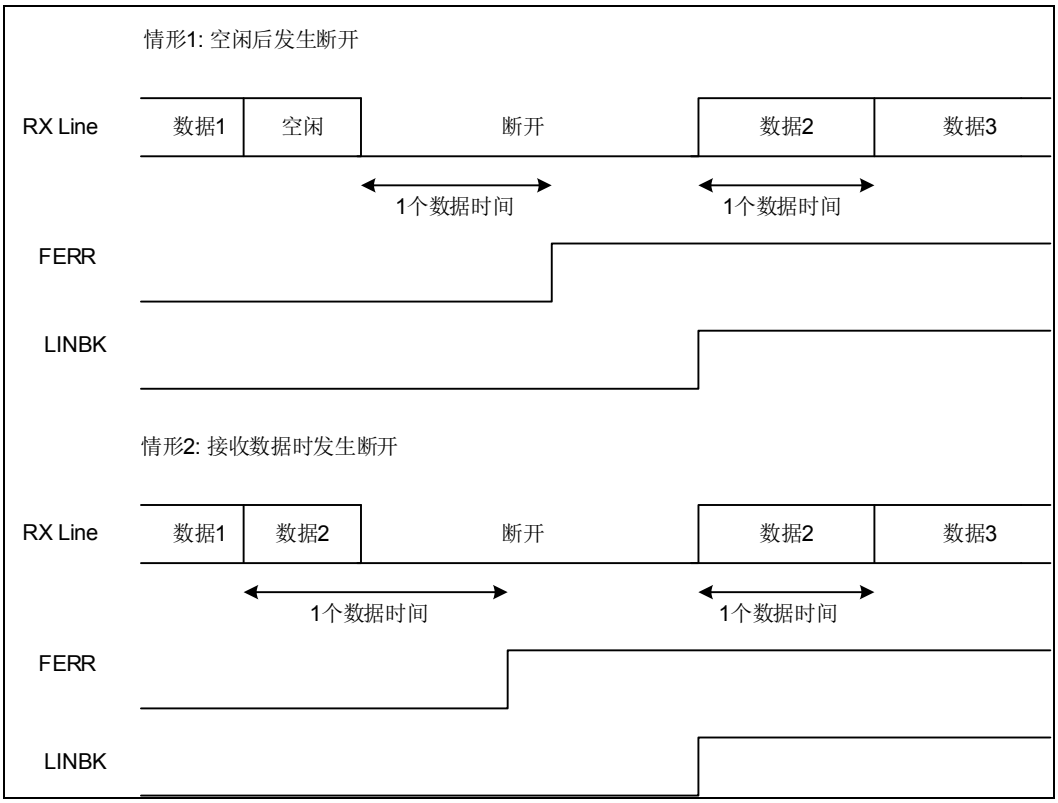


图 27-17 LIN 模式下的断开检测与帧错误的检测

27. 4. 11 单线半双工通讯

UART 可以配置成遵循单线半双工协议。在单线半双工模式下，TX 和 RX 引脚在芯片内部是连在一起的。使用控制位（MCON 寄存器中的 HDEN 位）选择半双工或全双工通信。

◇ 当 HDEN 为 1 时

- TX 和 RX 引脚在芯片内部是连在一起的
- RX 不再被使用
- 当没有数据传输时，TX 总是被释放。因此，它在空闲状态或接收状态时表现为一个标准 I/O 口。这就意味该 I/O 在不被 UART 驱动时，必须配置成悬空输入（或开漏的输出高）。

除此以外，通信与正常 UART 模式类似。由软件来管理线上的冲突（例如通过使用一个中央仲裁器）。特别的是，发送从不会被硬件所阻碍。当 LCON 寄存器的 TXEN 为 1，只要数据一写到数据寄存器中，发送就会开始。

27. 4. 12 智能卡模式

设置 8 位数据位加校验位：即 LCON 寄存器中 DLS=00，PE=1

设置 0.5/1.5 个停止位：即 LCON 寄存器的 STOP=0/1

设置 SCARD 寄存器的 SCEN 为 1 以进入 Smart card 模式

在 T=0（字符）模式中，保护时间内，校验错误在字符发送完毕后被提出。

所示为在数据线上有校验错误和没有校验错误时的情形。

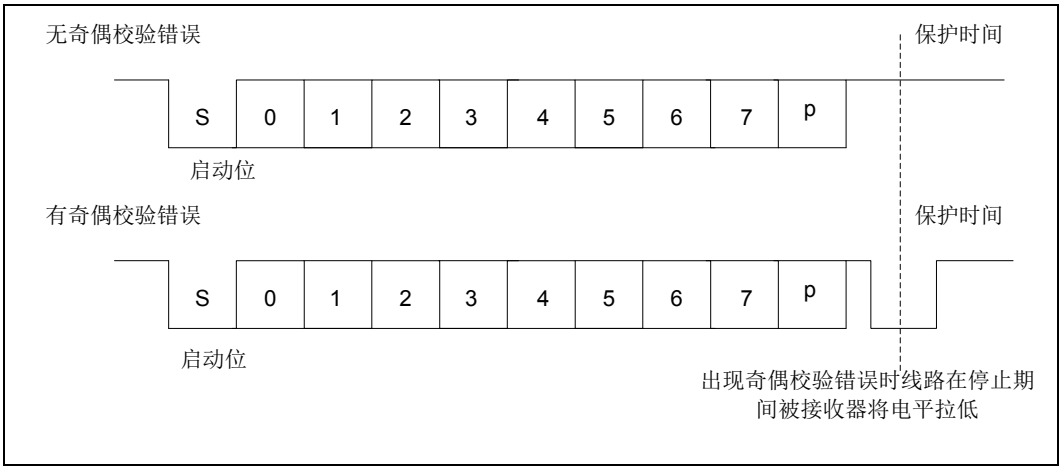


图 27-18 ISO 7816-3 异步协定

当连接到智能卡时，UART 的 TX 管脚和智能卡数据管脚通过同一根双向数据线进行通讯。所以 TX 引脚必须配置成开漏状态。

智能卡是一个单线半双工通信协议：

- ◇ 从发送移位寄存器发送数据会经过至少 1/2 个时钟周期的延迟。正常工作时，已满的发送移位寄存器会在下一个时钟边沿开始移位。在智能卡模式下，此发送过程还会进一步经过 1/2 波特时钟周期的延迟。
- ◇ 如果在接收一个使用 0.5 或 1.5 个停止位编程的帧期间检测到奇偶校验错误，则在完成接收帧后，发送线会被拉低一个时钟周期。这是为了向智能卡指出发送到 UART 的数据尚未正确接收。此 NACK 信号（将发送线拉低 1 个时钟周期）会导致发送器端（配置为 1.5 个停止位）出现帧错误。应用程序可根据协议重新发送数据。如果 NACK 控制位置 1，则接收器会发送“NACK”信号；否则不会发送 NACK 信号。
- ◇ 通过对保护时间寄存器进行编程，可以延迟 RIF 寄存器的 TBC 标志的置位。正常工作时，当发送移位寄存器为空时，会对 TBC 标志进行置位。在智能卡模式下，空的发送移位寄存器会触发保护时间计数器，使其递增计数至保护时间寄存器中的值。在此期间，TBC 标志被强制为低电平。当保护时间计数器达到设置值时，TBC 置位为高电平。
- ◇ 对 TBC 标志的释放不受智能卡模式的影响。
- ◇ 如果在发送端检测到帧错误（由来自接收器的 NACK 信号引起），则发送端的接收器不会将 NACK 作为起始位进行检测。根据 ISO 协议，接收到的 NACK 信号的持

续时间可以是 1 或 2 个时钟周期。

◇ 在接收端，如果检测到奇偶校验错误并发送了 NACK 信号，则接收端不会将 NACK 作为起始位进行检测。

注：在智能卡模式下带有帧错误的 0x00 数据将被视为数据，而非中断。

下图详细介绍了 UART 如何对 NACK 信号采样。在本例中，UART 正在发送数据并配置了 1.5 个停止位。UART 的接收部分已被使能，以检查数据的完整性和 NACK 信号。

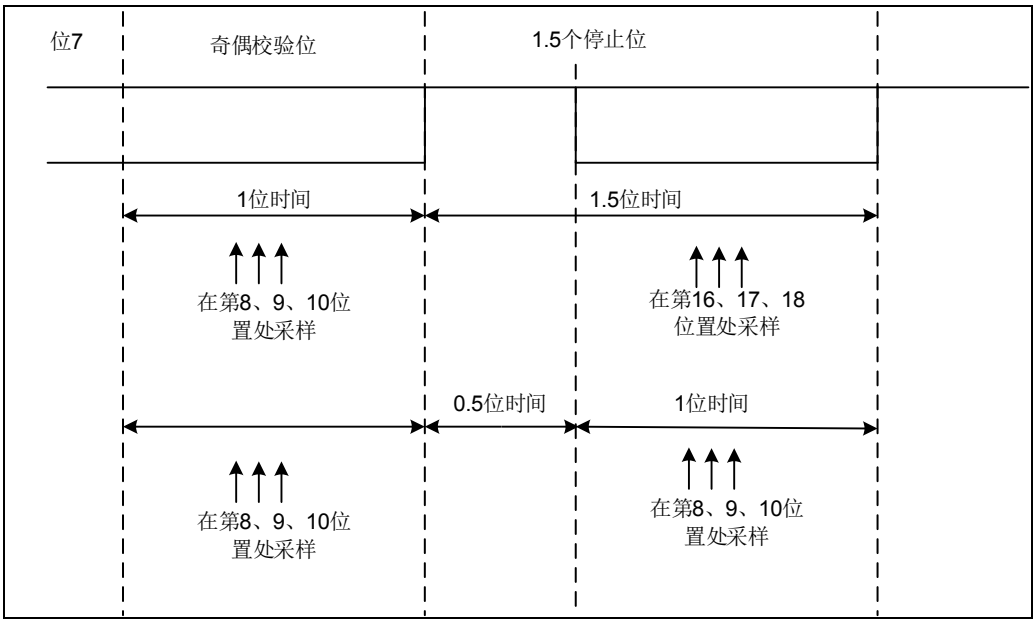


图 27-19 用 1.5 位停止位时检测校验错误

UART 可以通过 CK 脚向智能卡提供时钟。智能卡模式中，CK 和通讯没有关系，只是通过一个 5 位的预分频器从内部外设时钟源得到时钟信号。这个分频系数在 SCARD 寄存器的 PSC 位设置。CK 频率可以设置在 $f_{CK}/2$ 到 $f_{CK}/64$ 之间，f_{CK} 指外设输入时钟。

27. 4. 13 IrDA SIR模块

设置 MCON 寄存器的 IREN 为 1 以进入 IrDA 模式。

IrDA SIR 物理层规定使用反相归零（RZI）调制方案，它以红外光脉冲表示逻辑 0。

SIR 发送编码器用于调制 UART 发出的非归零（NRZ）位流。输出脉冲流会发送到外部输出驱动器和红外线 LED。UART 支持的 SIR 编码比特率最高为 115.2Kbps。在正常模式下，所发送的脉冲宽度规定为一个位周期的 3/16。

SIR 接收解码器用于解调由红外探测器发出的归零位流，并将接收到的 NRZ 串行位流输出到 UART。在空闲状态下，解码器输入通常为高电平（标记状态）。发送编码器输出的极性与解码器输入相反。当解码器输入为低电平时，会检测到起始位。

- ◇ IrDA 是一个半双工通信协议。如果发送器忙，例如 UART 正在向 IrDA 编码器发送数据，则 IrDA 解码器会忽略 IrDA 接收线上的所有数据；如果接收器忙，例如 UART 正在接收来自 RX 引脚上的数据，则 IrDA 不会对 UART 发送到 IrDA 的 TX 数据进行编码。在接收数据时，应避免同时进行发送，因为这样做可能会破坏要发送的数据。
- ◇ SIR 发送逻辑把 0 作为高脉冲发送，把 1 作为低电平发送。脉冲的宽度规定为所选位周期的 3/16
- ◇ SIR 解码器用于将兼容 IrDA 的接收信号转换为 UART 的位流。
- ◇ SIR 接收逻辑把高电平状态解释为 1，把低脉冲解释为 0。
- ◇ 发送编码器输出的极性与解码器输入相反。SIR 输出在空闲时处于低电平状态。
- ◇ 在 IrDA 模式里，LCON 寄存器中的 STOP 位必须配置成 1 个停止位。

接收器的建立时间应由软件进行管理。IrDA 物理层规范规定发送和接收之间至少要经过 10ms 的延迟（IrDA 是一个半双工协议）。

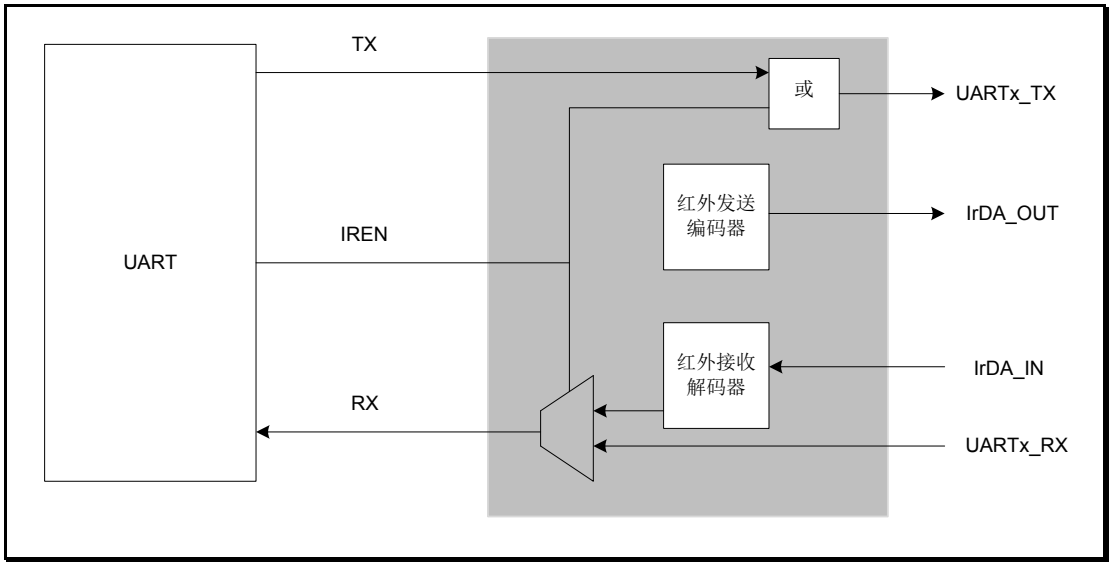


图 27-20　红外收发框图

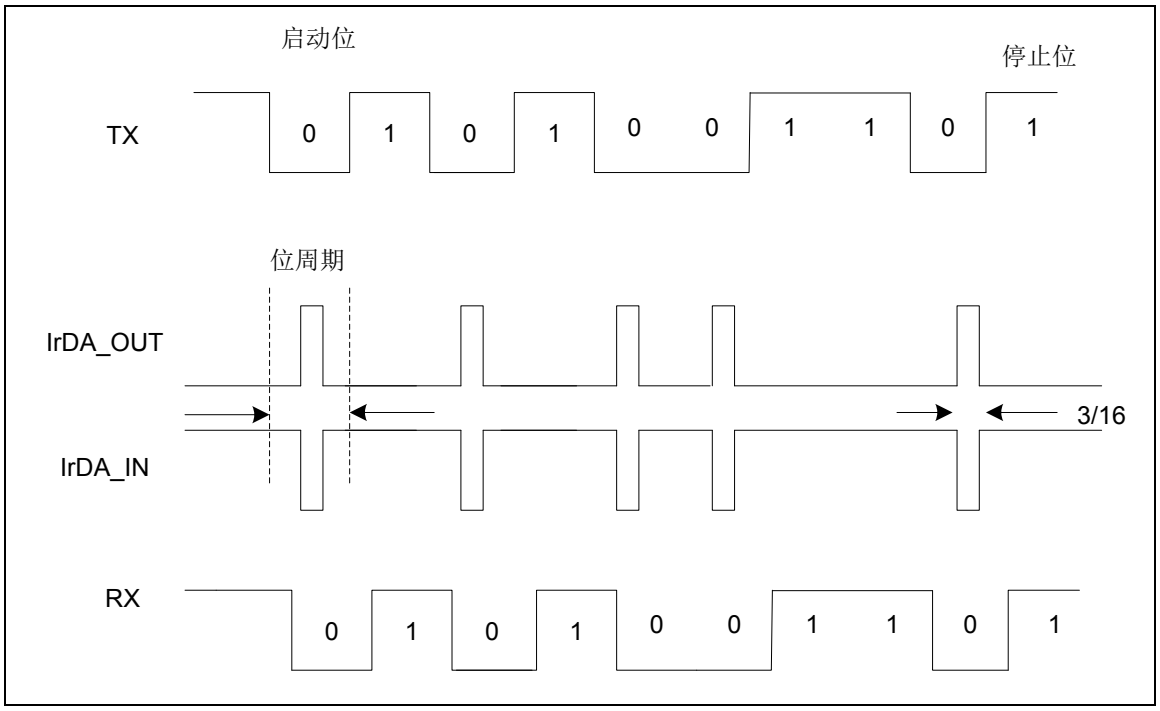


图 27-21 IrDA 数据调制（3/16）–正常模式

27. 4. 14 使用DMA连续通讯

设置 MCON 的 RXDMAEN 为 1 使能 RX DMA 或 TXDMAEN 为 1 使能 TX DMA。

UART 可以利用 DMA 连续通信。RX 缓冲器和 TX 缓冲器的 DMA 请求是分别产生的。

利用 DMA 发送

使用 DMA 进行发送，可以通过设置 MCON 寄存器中的 TXDMAEN 位使能。在数据被预先放到 DMA 外设所设定的 SRAM 区域，设置一个 DMA 通道给 UART 发送，要使用下列步骤（x 指通道号）：

- ◇ 在 DMA 控制寄存器上将 TXBUF 寄存器地址配置成 DMA 传输的目的地址。在每个发送事件后，数据将被传送到这个地址。
- ◇ 在 DMA 控制寄存器上将内存地址配置成 DMA 传输的源地址。在每个 TFTH 事件后，将从此存储器区读出数据并传送到 TXBUF 寄存器中。
- ◇ 在 DMA 控制寄存器中配置要传输的总的字节数。
- ◇ 在 DMA 控制寄存器上配置通道优先级。
- ◇ 根据应用程序的要求，配置在传输完成一半还是全部完成时产生 DMA 中断。
- ◇ 将 ICR 寄存器的 TFTH 位置 1 以清除 RIF 寄存器的 TFTH 标志。
- ◇ 在 DMA 控制寄存器上激活该通道。

注：若使用 FIFO，可一次写入多个传输字节数，并设定 FCON 寄存器中的 TXTH 位，可设定 FIFO 深度为 0,2,4,8 个字符时，新的数据将被传送到 TXBUF 寄存器中。

在发送模式下，当 DMA 传输完所有要发送的数据时，DMA 控制器设置 IFM 寄存器的 TFEMPTY 标志；监视 RIF 寄存器的 TFEMPTY 标志可以确认 UART 通信是否结束。这样可以在关闭 UART 或进入停机模式之前避免破坏最后一次传输的数据。软件必须等待 TBC 被置 1。TBC 标志在全部数据发送期间会是零，并且在最后一帧数据发送出去之后会由硬件置 1。

利用 DMA 接收

使用 DMA 进行接收，可以通过设置 MCON 寄存器中的 RXDMAEN 使能。当收到一个字节数据时，从 RXBUF 寄存器取出来的数据会被转移到 DMA 外设中指向的 SRAM 区域，将一个 DMA 通道设置给 UART 接收，要按照下列步骤：

- ◇ 在 DMA 控制寄存器上将 RXBUF 寄存器地址配置成 DMA 传输的源地址。在每个 RFTH 事件后，数据将从这个地址取走。
- ◇ 在 DMA 控制寄存器上将内存地址配置成 DMA 传输的目标地址。在每个 RFTH 事件后，数据将从 RXBUF 寄存器取出，放入这个目标地址。
- ◇ 在 DMA 控制寄存器中配置要传输的总的字节数。
- ◇ 在 DMA 控制寄存器上配置通道优先级。
- ◇ 根据应用程序的要求，配置在传输完成一半还是全部完成时产生 DMA 中断。
- ◇ 在 DMA 控制寄存器上激活该通道。

需要注意的是，若使用 FIFO，可一次接收多笔传输字节数，并设定 FCON 寄存器中的 RXTH，可设定 FIFO 深度为 1,4,8,14 个字符时，将新的数据将被传送到 RXBUF 寄存器中。

当传输完成时，若使能了 DMA 中断，DMA 控制器会产生中断。

多缓冲器通信中的错误标志和中断产生：在多缓冲器通信的情况下，通信期间如果发生任何错误，会在当前字节传输后将错误标志置 1。如果中断使能位被置 1，将产生中断。在单个字节接收的情况下，和 IFM 寄存器中的 RXBERR 和 RFOERR 一起被置起的帧错误和溢出错误，有单独的错误标志中断使能位；如果被置 1 了，会在当前字节传输结束后，产生中断。

27. 4. 15 中断配置

- UART_IER 中断使能寄存器

  中断使能寄存器，此位设定 1 时，表示开启中断功能，并且同时反映在 IVS 寄存器。此寄存器属于只写，并且只允许写入 1，无法写 0 取消使能中断设定。

- UART_IDR 中断禁止寄存器

  中断禁止寄存器，此位设定 1 时，表示关闭中断功能，并且同时反映在 IVS 寄存器。此寄存器属于只写，并且只允许写入 1，无法写 0 取消禁止中断设定。

- UART_IVS 中断有效状态寄存器

  中断有效状态寄存器，反映 IER 与 IDR 寄存器所设定的结果。0：中断禁止 1：中断使能

- UART_RIF 原始中断标志寄存器

  原始中断标志寄存器，反映所有发生中断事件的状态，无论 IVS 是否有使能中断，皆会反映在此寄存器中，主要提供用户监控无屏蔽的中断位，是否有错误事件发生。

- UART_IFM 中断标志屏蔽寄存器

  中断标志屏蔽寄存器，记录中断使能位所发生中断事件。0：无中断事件 1：发生中断事件。

- UART_ICR 中断清除寄存器

  中断清除寄存器，此位设定 1 时，清除中断标志 RIF 与 IFM，此寄存器属于写一清零，并且只允许写入 1 清除，无法写 0 清除。

  在 UART 中，配置了 17 种中断，分别为下表。

中断事件	中断标志
接收器字节格式错误	RXBERR
自动波特率检测结束	ABEND
自动波特率检测超时	ABTO
CTSn 引脚电平改变	DCTS
接收超时	RXTO
地址匹配	ADDRM
LIN 断开检测	LINBK
块结束	EOB
噪声位检测	NOISE
接收器 FIFO 触发阈值	RFTH
接收器 FIFO 满	RFFULL
接收器 FIFO 溢出	RFOERR
接收器 FIFO 下溢	RFUERR
发送器字节完成	TBC
发送器 FIFO 触发阈值	TFTH
发送器 FIFO 空	TFEMPTY
发送器 FIFO 溢出	TFOVER

表 27-4　中断配置表

27. 5 特殊功能寄存器

27. 5. 1 寄存器列表

UART 寄存器列表		
名称	偏移地址	描述
UART_RXBUF	000_H	UART 接收缓冲寄存器
UART_TXBUF	004_H	UART 发送缓冲寄存器
UART_BRR	008_H	UART 波特率寄存器
UART_LCON	$00C_H$	UART 线控寄存器
UART_MCON	010_H	UART 模式控制寄存器
UART_RS485	014_H	UART RS485 控制寄存器
UART_SCARD	018_H	UART 智能卡控制寄存器
UART_LIN	$01C_H$	UART LIN 控制寄存器
UART_RTOR	020_H	UART 接收超时寄存器
UART_FCON	024_H	UART FIFO 控制寄存器
UART_STAT	028_H	UART 状态寄存器
UART_IER	$02C_H$	UART 中断使能寄存器
UART_IDR	030_H	UART 中断禁止寄存器
UART_IVS	034_H	UART 中断有效位状态寄存器
UART_RIF	038_H	UART 原始中断标志寄存器
UART_IFM	$03C_H$	UART 中断标志屏蔽寄存器
UART_ICR	040_H	UART 中断清除寄存器

27. 5. 2 寄存器描述

27. 5. 2. 1 UART接收缓冲寄存器（UART_RXBUF）

UART 接收缓冲寄存器（UART_RXBUF）																															
偏移地址：00_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							RXBUF								

Reserved	Bit 31-9	—	保留
RXBUF	Bit 8-0	R	**接收缓冲寄存器** 包含接收到的字节。 RXBUF寄存器提供接收移位寄存器和内部总线间的并行接口。 注：Bit 8用于RS485寻址模式

27. 5. 2. 2 UART发送缓冲寄存器（UART_TXBUF）

UART 发送缓冲寄存器（UART_TXBUF）																															
偏移地址：04_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							TXBUF								

Reserved	Bit 31-9	—	保留
TXBUF	Bit 8-0	R/W	**发送缓冲寄存器** 用于写入要发送的数据字节。 TXBUF寄存器提供发送移位寄存器与内部总线间的并行接口。 注：Bit 8用于RS485寻址模式

27. 5. 2. 3 UART波特率寄存器（UART_BRR）

UART 波特率寄存器（UART_BRR）

偏移地址：08_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	保留
BRR	Bit 15-0	R/W	**波特率寄存器** 整数部分BRR<15:4> 小数部分BRR<3:0> 此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 注：使用自动波特率功能时则可自动写入

27. 5. 2. 4 UART线控寄存器（UART_LCON）

UART 线控寄存器（UART_LCON）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																TXEN	RXEN	DBCEN	Reserved		BREAK	Reserved	TXINV	RXINV	DATAINV	MSB	PS	PE	STOP	DLS	

Reserved	Bit 31-16	—	**保留**
TXEN	Bit 15	R/W	**发送器使能** 使能发送器，此位由软件设置1和清除。 0：发送器禁止 1：发送器使能
RXEN	Bit 14	R/W	**接收器使能** 使能接收器，此位由软件设置1和清除。 0：接收器禁止 1：接收器使能
DBCEN	Bit 13	R/W	**防抖动使能** 使能防抖动功能，此位由软件设置1和清除。 0：防抖动禁止 1：防抖动使能，在RX线上须维持8个模块时钟（PCLK）的电平
Reserved	Bit 12-11	—	**保留**
BREAK	Bit 10	R/W	**断开使能** 使能断开功能，此位由软件设置1和清除。 此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 0：断开禁止 1：断开使能，会使得TX输出为0
Reserved	Bit 9	—	**保留**
TXINV	Bit 8	R/W	**TX引脚电平反向** TX引脚反向功能，此位由软件设置1和清除。 此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 0：TX引脚信号工作于标准逻辑电平（VDD=1/idle，Gnd=0/mark） 1：TX引脚信号反向（VDD=0/mark，Gnd=1/idle）。 此功能可用于TX线上带有外部反向器时
RXINV	Bit 7	R/W	**RX引脚电平反向** RX引脚反向功能，此位由软件设置1和清除。 此位在LCON寄存器中的RXEN与TXEN位为0时

			才可以写入。 0：RX引脚信号工作于标准逻辑电平（VDD=1/idle，Gnd=0/mark） 1：RX引脚信号反向（VDD=0/mark，Gnd=1/idle）。此功能可用于RX线上带有外部反向器时
DATAINV	Bit 6	R/W	**二进制反向使能** 二进制反向功能，此位由软件设置1和清除。 此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 0：缓冲寄存器中的逻辑数据在接收的时候，采用正/直接逻辑。（1=H，0=L） 1：缓冲寄存器中的逻辑数据在接收的时候，采用负/反向逻辑。（1=L，0=H）
MSB	Bit 5	R/W	**高位在前使能** 高位在前功能，此位由软件设置1和清除。 此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 0：数据在发送和接收的时候，采用起始位后面跟着第0位的顺序 1：数据在发送和接收的时候，采用起始位后面跟着的最高位的顺序
PS	Bit 4	R/W	**校验位奇偶选择** 当使能校验功能时，选择校验位为奇校验或偶校验，此位由软件设置1和清除。 此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 0：奇校验 1：偶校验
PE	Bit 3	R/W	**校验使能** 使能校验功能，计算好的校验位被插入到最高位，并检测接收数据的校验位（接收与发送功能），此位由软件设置1和清除。 此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 0：校验位禁止 1：校验位使能
STOP	Bit 2	R/W	**停止位选择** 此位由软件设置1和清除。 此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 普通模式下 0：1个停止位 1：2个停止位（在5字长模式为1.5个停止位） 智能卡模式

			0：0.5个停止位 1：1.5个停止位
DLS	Bit 1-0	R/W	**数据字长选择** 此位由软件设置1和清除。 此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 00：8位字长 01：7位字长 10：6位字长 11：5位字长

27. 5. 2. 5 UART模式控制寄存器（UART_MCON）

UART 模式控制寄存器（UART_MCON）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																TXDMAEN	RXDMAEN	Reserved		ABRREPT	ABRMOD		ABREN	Reserved		BKREQ	HDEN	IREN	AFCEN	RTSSET	LPBKEN

Reserved	Bit 31-16	—	**保留**
TXDMAEN	Bit 15	R/W	**发送器DMA使能** 此位由软件设置1和清除。 0：发送器DMA通讯禁止 1：发送器DMA通讯使能
RXDMAEN	Bit 14	R/W	**接收器DMA使能** 此位由软件设置1和清除。 0：接收器DMA通讯禁止 1：接收器DMA通讯使能
Reserved	Bit 13-12	—	**保留**
ABRREPT	Bit 11	R/W	**重复自动波特率检测** 在使能自动波特率检测时，当产生波特率超时时并不会关闭自动波特率检测功能，并在下一个下降沿时重复自动波特率检测，此位由软件设置1和清除。 0：重复自动波特率检测禁止 1：重复自动波特率检测使能
ABRMOD	Bit 10-9	R/W	**自动波特率模式选择** 此位由软件设置1和清除。 00：模式0，检测第一个下降沿到第二个下降沿时间（检测2位） 01：模式1，检测第一个下降沿到第一个上升沿时间（检测1位） 10：模式2，检测第一个下降沿到第一个上升沿时间（检测2位） 11：保留
ABREN	Bit 8	R/W	**自动波特率使能** 此位在使能后并完成自动波特率检测后会自动清除，也可由软件设置1和清除。 0：自动波特率禁止 1：自动波特率使能
Reserved	Bit 7-6	—	**保留**
BKREQ	Bit 5	W	**断开请求** 此位在写入后的下一个时钟会自动清除。

			0：断开请求禁止 1：断开请求使能，根据设定的N位长（8/7/6/5）产生N个低脉冲信号
HDEN	Bit 4	R/W	**单线半双工使能** 此位由软件设置1和清除。 0：单线半双工禁止 1：单线半双工使能
IREN	Bit 3	R/W	**IrDA红外线模式使能** 此位由软件设置1和清除。 0：IrDA红外线模式禁止 1：IrDA红外线模式使能
AFCEN	Bit 2	R/W	**自动流量控制使能** 此位由软件设置1和清除。 0：自动流量控制禁止 1：自动流量控制使能
RTSSET	Bit 1	R/W	**RTSn设置控制** 此位由软件设置1和清除。 0：自动流量控制禁止时，RTSn引脚输出高电平 1：自动流量控制禁止时，RTSn引脚输出低电平
LPBKEN	Bit 0	R/W	**回送模式使能** 此模式是用于UART测试项目的诊断模式，在UART普通模式下运行，TX引脚输出为高电平，串行的数据在内部回送至RX。在此模式下，所有中断都是正常运行的，此位由软件设置1和清除。 0：回送模式禁止 1：回送模式使能

27. 5. 2. 6 UART RS485 控制寄存器（UART_RS485）

UART RS485 控制寄存器（UART_RS485）

偏移地址：14_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-24	—	**保留**
DLY	Bit 23-16	R/W	**延迟数值** 用于设置延迟RTSn的输出时间，是由一个8位计数器计数，此位由软件设置和清除
ADDR	Bit 15-8	R/W	**地址匹配值** 用于多机通讯时地址标记的检测。 接收器在RS485自动检测模式时，当接收数据的最高位为1且匹配ADDR，数据才允许接收至FIFO中，否则舍弃此数值，此位由软件设置和清除。
Reserved	Bit 7-4	—	**保留**
AADINV	Bit 3	R/W	**驱动使能反向** 在自动流量控制模式时，设置驱动使能引脚（RTSn/DE）的输出电平，此位由软件设置和清除。 0：当发送器开始发送数据时，驱动使能引脚输出0，发送完成且FIFO内无数据时，驱动使能引脚输出1 1：当发送器开始发送数据时，驱动使能引脚输出1，发送完成且FIFO内无数据时，驱动使能引脚输出0
AADACEN	Bit 2	R/W	**自动流量控制模式使能** 此位由软件设置1和清除。 0：自动流量控制模式禁止 1：自动流量控制模式使能
AADNEN	Bit 1	R/W	**普通模式使能** 此位由软件设置1和清除。 0：普通模式禁止 1：普通模式使能，接收数据的地址位为第8位（UART_RXBUF）
AADEN	Bit 0	R/W	**自动地址检测模式使能** 在普通模式时，设置此位无效，此位由软件设置1和清除。 0：自动地址检测模式禁止 1：自动地址检测模式使能，当接收数据的地址位为1且匹配ADDR时，才会将数据接收至FIFO中

27. 5. 2. 7 UART智能卡控制寄存器（UART_SCARD）

UART 智能卡控制寄存器（UART_SCARD）

偏移地址：18_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

BLEN	Bit 31-24	R/W	**块长度** 设置了智能卡模式T=1的接收时的块长度，此位由软件设置1和清除。 例如 BLEN = 0 -> 0个信号字符 BLEN = 1 -> 1个信号字符 BLEN = 255 -> 255个信号字符 这个功能也可以在其他模式中使用，当LCON寄存器的RXEN位清除时，块长度计数器会重新计数
GT	Bit 23-16	R/W	**保护时间** 设置保护时间长度，是使用波特时钟为单位。 在智能卡模式中使用，完成标志（RIF寄存器TBC位）在保护时间过后置起，此位由软件设置1和清除
PSC	Bit 15-8	R/W	**分频器数值** 此位由软件设置1和清除。 智能卡模式： PSC<4:0>：输出时钟分频数值 用于设定UART时钟的分频数，得到智能卡输出时钟，由五个有效位组成，寄存器值加1后，乘以2得到的数值作为分频 00000：2分频 00001：4分频 00010：6分频 00011：8分频
Reserved	Bit 7-6	—	—
SCCNT	Bit 5-3	R/W	**智能卡重试计数器** 设置智能卡模式中接收和发送的重试次数。此位由软件设置1和清除。 在发送模式下，在产生帧错误前重试发送的次数 在接收模式下，在接收到NACK后重试接收的次数 0x0：重试功能关闭，在发送与接收模式下不进行自动重试

			0x1至0x7：在产生错误前自动重试的次数
SCLKEN	Bit 2	R/W	**智能卡时钟使能** 此位由软件设置1和清除。 0：CK引脚禁止 1：CK引脚使能
SCNACK	Bit 1	R/W	**智能卡NACK发送使能** 此位由软件设置1和清除。 0：出现校验错误时禁止发送NACK信号 1：出现校验错误时使能发送NACK信号
SCEN	Bit 0	R/W	**智能卡模式使能** 此位由软件设置1和清除。 0：智能卡模式禁止 1：智能卡模式使能

27. 5. 2. 8 UART LIN控制寄存器（UART_LIN）

UART LIN 控制寄存器（UART_LIN）

偏移地址：$1C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-3	—	**保留**
LINBKREQ	Bit 2	W1	**LIN模式断开请求** 在LIN模式下，发送器将发送13位0作为断开符号后，发送2位1用于对下一个开始位的检测。此位由软件设置1并在下一个时钟后自动清除。 0：LIN模式断开请求禁止 1：LIN模式断开请求使能
LINBDL	Bit 1	R/W	**LIN模式断开字长** 此位由软件设置1和清除。 0：10位断开字符检测 1：11位断开字符检测
LINEN	Bit 0	R/W	**LIN模式使能** 此位由软件设置1和清除。 0：LIN模式禁止 1：LIN模式使能

27. 5. 2. 9 UART接收超时寄存器（UART_RTOR）

UART 超时接收寄存器（UART_RTOR）

偏移地址：20_H

复位值：$00000000\_00000000\_00000000\_11111111_B$

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

Reserved	Bit 31-25	—	**保留**
RTOEN	Bit 24	R/W	**接收器超时使能** 此位由软件设置1和清除。 0：接收器超时禁止 1：接收器超时使能
RTO	Bit 23-0	R/W	**接收器超时数值** 设置接收超时时间，使用波特率时钟的字长为单位。在标准模式下，接收最后一个字节后，在超时时间内未检测到新的起始位，将置起 RIF 寄存器的 RXTO 位，此位由软件设置和清除。

27. 5. 2. 10 UART FIFO控制寄存器（UART_FCON）

UART FIFO 控制寄存器（UART_FCON）

偏移地址：24_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	保留
TXFL	Bit 15-11	R	**发送器FIFO中数据个数** 显示发送器FIFO内准备发送的个数（0-16），此位由硬件设置且只能读取。
TXTH	Bit 10-9	R/W	**发送器触发阈值** 设置发送器触发阈值，当TXFL个数小于TXTH设定的个数时，将置起RIF寄存器的TFTH位，与STAT寄存器的TFTH位 00：TX FIFO内无字符 01：TX FIFO有2个字符 10：TX FIFO有4个字符 11：TX FIFO有8个字符
TFRST	Bit 8	W1	**发送器FIFO重置使能** 设置清除TX FIFO内字符，此位由软件设置1且在下一个时钟自动清除。 0：发送器FIFO重置禁止 1：发送器FIFO重置使能
RXFL	Bit 7-3	R	**接收器FIFO中数据个数** 显示接收器FIFO内准备接收的个数（0-16），此位由硬件设置且只能读取
RXTH	Bit 2-1	R/W	**接收器触发阈值** 设置接收器触发阈值，当RXFL个数大于或等于RXTH设定的个数时，将置起RIF寄存器的RFTH位，与STAT寄存器的RFTH位 00：RX FIFO有1字符 01：RX FIFO有4个字符 10：RX FIFO有8个字符 11：RX FIFO有14个字符
RFRST	Bit 0	W1	**接收器FIFO重置** 设置清除RX FIFO内字符，此位由软件设置1且在下一个时钟自动清除。 0：接收器FIFO重置禁止 1：接收器FIFO重置使能

27. 5. 2. 11 UART状态寄存器（UART_STAT）

UART 状态寄存器（UART_STAT）

偏移地址：28_H

复位值：$00000000\_00000001\_10000100\_00001000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TFOERR	TFFULL	TFEMPTY	TFTH	TSBUSY	RFUERR	RFOERR	RFFULL	RFEMPTY	RFTH	RSBUSY	Reserved				CTSSTA	BKERR	FERR	PERR

Reserved	Bit 31-19	—	保留
TFOERR	Bit 18	R	**发送器FIFO溢出错误** 当发送器FIFO内已有16个数据时，有新数据再次写入FIFO中时，将会置起此位并舍弃新数据。此位由硬件设置1，在发送数据时清除或设置ICR寄存器的TFOVER位清除 0：发送器FIFO溢出错误未产生 1：发送器FIFO溢出错误产生
TFFULL	Bit 17	R	**发送器FIFO满** 当发送器FIFO内有16个数据时，此位由硬件设置1，在FIFO内未满足16个数据时清除。 0：发送器FIFO未满足16个数据 1：发送器FIFO满足16个数据
TFEMPTY	Bit 16	R	**发送器FIFO空** 当发送器FIFO内无任何数据时，此位由硬件设置1，在FIFO内写入数据时清除。 0：发送器FIFO有数据 1：发送器FIFO无数据
TFTH	Bit 15	R	**发送器FIFO触发阈值** 当FCON寄存器的TXFL位所指示的FIFO字节数小于FCON寄存器的TXTH设定的阈值，将会置起此位。此位由硬件设置1和在未满足阈值水平时清除 0：发送器FIFO未小于阈值水平 1：发送器FIFO小于阈值水平
TSBUSY	Bit 14	R	**发送器移位寄存器忙碌** 当写入数据至FIFO中由硬件设置1在发送最后一个数据完成后清除。 0：发送器FIFO内无数据等待传送 1：发送器FIFO内有数据等待传送且未发送完最后一个数据
RFUERR	Bit 13	R	**接收器FIFO下溢错误** 当接收器FIFO内无数据时，又再次读取接收器FIFO时，将会置起此位。此位由硬件设置1，在接

			收数据时清除或设置ICR寄存器的RFUERR位清除 0：接收器FIFO下溢错误未产生 1：接收器FIFO下溢错误产生
RFOERR	Bit 12	R	**接收器FIFO溢出错误** 当接收器FIFO内已有16个数据时，有新数据再次接收至FIFO中时，将会置起此位并舍弃新数据。此位由硬件设置1，在读取数据时清除或设置ICR寄存器的RFOERR位清除 0：接收器FIFO溢出错误未产生 1：接收器FIFO溢出错误产生
RFFULL	Bit 11	R	**接收器FIFO满** 当接收器FIFO内有16个数据时，此位由硬件设置1，在FIFO内未满足16个数据时清除。 0：接收器FIFO未满足16个数据 1：接收器FIFO满足16个数据
RFEMPTY	Bit 10	R	**接收器FIFO空** 当接收器FIFO内无任何数据时，此位由硬件设置1，接收数据至FIFO内时清除。 0：接收器FIFO有数据 1：接收器FIFO无数据
RFTH	Bit 9	R	**接收器FIFO触发阈值** 当FCON寄存器的RXFL位所指示的FIFO字节数大于或等于FCON寄存器的RXTH设定的阈值，将会置起此位。此位由硬件设置1，在未满足阈值水平时清除 0：接收器FIFO未大于或等于阈值水平 1：接收器FIFO大于或等于阈值水平
RSBUSY	Bit 8	R	**接收移位寄存器忙碌** 当接收数据时，由硬件设置1在完成接收数据后清除 0：接收器未接收数据 1：接收器正在接收数据
Reserved	Bits 7-4	—	**保留**
CTSSTA	Bit 3	R	**CTSn状态** 此位显示CTSn输入引脚状态，由硬件设置1和清除。 0：CTSn输入引脚为0 1：CTSn输入引脚为1
BKERR	Bit 2	R	**断开错误** 当接收数据与停止位皆为0时，将由硬件设置1。此位为显示当前读取接收器FIFO的数值。 0：断开错误未产生 1：断开错误产生

FERR	Bit 1	R	**帧错误** 当接收数据的停止位为0时，将由硬件设置1。此位为显示当前读取接收器FIFO的数值。 0：帧错误未产生 1：帧错误产生
PERR	Bit 0	R	**校验错误** 当接收数据的校验位接收错误时，将由硬件设置1。此位为显示当前读取接收器FIFO的数值。 0：校验错误未产生 1：校验错误产生

27. 5. 2. 12 UART中断使能寄存器（UART_IER）

UART 中断使能寄存器（UART_IER）

偏移地址：$2C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TFOVER	Reserved	TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	Reserved	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR

Reserved	Bit 31-19	—	**保留**
TFOVER	Bit 18	W1	**发送器FIFO溢出中断使能** 0：写入0无效 1：发送器FIFO溢出中断使能
Reserved	Bit 17	—	**保留**
TFEMPTY	Bit 16	W1	**发送器FIFO空中断使能** 0：写入0无效 1：发送器FIFO空中断使能
TFTH	Bit 15	W1	**发送器FIFO触发阈值中断使能** 0：写入0无效 1：发送器FIFO触发阈值中断使能
TBC	Bit 14	W1	**发送器字节完成中断使能** 0：写入0无效 1：发送器字节完成中断使能
RFUERR	Bit 13	W1	**接收器FIFO下溢中断使能** 0：写入0无效 1：接收器FIFO下溢中断使能
RFOERR	Bit 12	W1	**接收器FIFO溢出中断使能** 0：写入0无效 1：接收器FIFO溢出中断使能
RFFULL	Bit 11	W1	**接收器FIFO满中断使能** 0：写入0无效 1：接收器FIFO满中断使能
Reserved	Bit 10	—	**保留**
RFTH	Bit 9	W1	**接收器FIFO触发阈值中断使能** 0：写入0无效 1：接收器FIFO触发阈值中断使能
NOISE	Bit 8	W1	**噪声位检测中断使能** 0：写入0无效 1：噪声位检测中断使能
EOB	Bit 7	W1	**块结束中断使能** 0：写入0无效

			1：块结束中断使能
LINBK	Bit 6	W1	**LIN断开检测中断使能** 0：写入0无效 1：LIN断开检测中断使能
ADDRM	Bit 5	W1	**地址匹配中断使能** 0：写入0无效 1：地址匹配中断使能
RXTO	Bit 4	W1	**接收超时中断使能** 0：写入0无效 1：接收超时中断使能
DCTS	Bit 3	W1	**CTSn引脚电平改变中断使能** 0：写入0无效 1：CTSn引脚电平改变中断使能
ABTO	Bit 2	W1	**自动波特率检测超时中断使能** 0：写入0无效 1：自动波特率检测超时中断使能
ABEND	Bit 1	W1	**自动波特率检测结束中断使能** 0：写入0无效 1：自动波特率检测结束中断使能
RXBERR	Bit 0	W1	**接收器字节格式错误中断使能** 0：写入0无效 1：接收器字节格式错误中断使能

27. 5. 2. 13 UART中断禁止寄存器（UART_IDR）

UART 中断禁止寄存器（UART_IDR）

偏移地址：30_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TFOVER	Reserved	TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	Reserved	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR

Reserved	Bit 31-19	—	**保留**
TFOVER	Bit 18	W1	**发送器FIFO溢出中断禁止** 0：写入0无效 1：发送器FIFO溢出中断禁止
Reserved	Bit 17	—	**保留**
TFEMPTY	Bit 16	W1	**发送器FIFO空中断禁止** 0：写入0无效 1：发送器FIFO空中断禁止
TFTH	Bit 15	W1	**发送器FIFO触发阈值中断禁止** 0：写入0无效 1：发送器FIFO触发阈值中断禁止
TBC	Bit 14	W1	**发送器字节完成中断禁止** 0：写入0无效 1：发送器字节完成中断禁止
RFUERR	Bit 13	W1	**接收器FIFO下溢中断禁止** 0：写入0无效 1：接收器FIFO下溢中断禁止
RFOERR	Bit 12	W1	**接收器FIFO溢出中断禁止** 0：写入0无效 1：接收器FIFO溢出中断禁止
RFFULL	Bit 11	W1	**接收器FIFO满中断禁止** 0：写入0无效 1：接收器FIFO满中断禁止
Reserved	Bit 10	—	**保留**
RFTH	Bit 9	W1	**接收器FIFO触发阈值中断禁止** 0：写入0无效 1：接收器FIFO触发阈值中断禁止
NOISE	Bit 8	W1	**噪声位检测中断禁止** 0：写入0无效 1：噪声位检测中断禁止
EOB	Bit 7	W1	**块结束中断禁止** 0：写入0无效

			1：块结束中断禁止
LINBK	Bit 6	W1	**LIN断开检测中断禁止** 0：写入0无效 1：LIN断开检测中断禁止
ADDRM	Bit 5	W1	**地址匹配中断禁止** 0：写入0无效 1：地址匹配中断禁止
RXTO	Bit 4	W1	**接收超时中断禁止** 0：写入0无效 1：接收超时中断禁止
DCTS	Bit 3	W1	**CTSn引脚电平改变中断禁止** 0：写入0无效 1：CTSn引脚电平改变中断禁止
ABTO	Bit 2	W1	**自动波特率检测超时中断禁止** 0：写入0无效 1：自动波特率检测超时中断禁止
ABEND	Bit 1	W1	**自动波特率检测结束中断禁止** 0：写入0无效 1：自动波特率检测结束中断禁止
RXBERR	Bit 0	W1	**接收器字节格式错误中断禁止** 0：写入0无效 1：接收器字节格式错误中断禁止

27. 5. 2. 14 UART中断有效状态寄存器（UART_IVS）

UART 中断有效状态寄存器（UART_IVS）

偏移地址：34_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TFOVER	Reserved	TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	Reserved	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR

Reserved	Bit 31-19	—	保留
TFOVER	Bit 18	R	**发送器FIFO溢出中断有效位** 0：发送器FIFO溢出中断禁止 1：发送器FIFO溢出中断使能
Reserved	Bit 17	—	保留
TFEMPTY	Bit 16	R	**发送器FIFO空中断有效位** 0：发送器FIFO空中断禁止 1：发送器FIFO空中断使能
TFTH	Bit 15	R	**发送器FIFO触发阈值中断有效位** 0：发送器FIFO触发阈值中断禁止 1：发送器FIFO触发阈值中断使能
TBC	Bit 14	R	**发送器字节完成中断有效位** 0：发送器字节完成中断禁止 1：发送器字节完成中断使能
RFUERR	Bit 13	R	**接收器FIFO下溢中断有效位** 0：接收器FIFO下溢中断禁止 1：接收器FIFO下溢中断使能
RFOERR	Bit 12	R	**接收器FIFO溢出中断有效位** 0：接收器FIFO溢出中断禁止 1：接收器FIFO溢出中断使能
RFFULL	Bit 11	R	**接收器FIFO满中断有效位** 0：接收器FIFO满中断禁止 1：接收器FIFO满中断使能
Reserved	Bit 10	—	保留
RFTH	Bit 9	R	**接收器FIFO触发阈值中断有效位** 0：接收器FIFO触发阈值中断禁止 1：接收器FIFO触发阈值中断使能
NOISE	Bit 8	R	**噪声位检测中断有效位** 0：噪声位检测中断禁止 1：噪声位检测中断使能
EOB	Bit 7	R	**块结束中断有效位** 0：块结束中断禁止

			1：块结束中断使能
LINBK	Bit 6	R	**LIN断开检测中断有效位** 0：LIN断开检测中断禁止 1：LIN断开检测中断使能
ADDRM	Bit 5	R	**地址匹配中断有效位** 0：地址匹配中断禁止 1：地址匹配中断使能
RXTO	Bit 4	R	**接收超时中断有效位** 0：接收超时中断禁止 1：接收超时中断使能
DCTS	Bit 3	R	**CTSn引脚电平改变中断有效位** 0：CTSn引脚电平改变中断禁止 1：CTSn引脚电平改变中断使能
ABTO	Bit 2	R	**自动波特率检测超时中断有效位** 0：自动波特率检测超时中断禁止 1：自动波特率检测超时中断使能
ABEND	Bit 1	R	**自动波特率检测结束中断有效位** 0：自动波特率检测结束中断禁止 1：自动波特率检测结束中断使能
RXBERR	Bit 0	R	**接收器字节格式错误中断有效位** 0：接收器字节格式错误中断禁止 1：接收器字节格式错误中断使能

27. 5. 2. 15 UART原始中断标志寄存器（UART_RIF）

UART 原始中断标志寄存器（UART_RIF）

偏移地址：38_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TFOVER	Reserved	TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	Reserved	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR

Reserved	Bit 31-19	—	**保留**
TFOVER	Bit 18	R	**发送器FIFO溢出中断标志** 当发送器FIFO内已有16个数据时，有新数据再次写入FIFO中时，将会置起此位并舍弃新数据。此位由硬件设置1，设置ICR寄存器的TFOVER位清除 0：无中断产生 1：发送器FIFO溢出中断产生
Reserved	Bit 17	—	**保留**
TFEMPTY	Bit 16	R	**发送器FIFO空中断标志** 发送器FIFO由一个数值转为空时置起，或设置IER寄存器的TFEMPTY位时，UART会判断当前发送器FIFO是否为空将此位置起。此位由硬件设置1，设置ICR寄存器的TFEMPTY位清除 0：无中断产生 1：发送器FIFO空中断产生
TFTH	Bit 15	R	**发送器FIFO触发阈值中断标志** 发送器FIFO个数达到发送器设定的阈值时置起，或设置IER寄存器的TFTH位时，UART会判断当前发送器FIFO个数达到发送器设定的阈值将此位置起。此位由硬件设置1，设置ICR寄存器的TFTH位清除 0：无中断产生 1：发送器FIFO触发阈值中断产生
TBC	Bit 14	R	**发送器字节完成中断标志** 发送器完成单个字节时置起。此位由硬件设置1，设置ICR寄存器的TBC位清除 0：无中断产生 1：发送器字节完成中断产生
RFUERR	Bit 13	R	**接收器FIFO下溢中断标志** 当接收器FIFO内无数据时，又再次读取接收器FIFO时置起。此位由硬件设置1，设置ICR寄存器的RFUERR位清除

			0：无中断产生 1：接收器FIFO下溢中断产生
RFOERR	Bit 12	R	**接收器FIFO溢出中断标志** 当接收器FIFO内已有16个数据时，有新数据再次接收至FIFO中时置起并舍弃新数据。此位由硬件设置1，设置ICR寄存器的RFOERR位清除 0：无中断产生 1：接收器FIFO溢出中断产生
RFFULL	Bit 11	R	**接收器FIFO满中断标志** 当接收器FIFO内由15个数据接收至16个数据时置起，或设置IER寄存器的RFFULL位时，UART会判断当前接收器FIFO是否为16个数据而置起。此位由硬件设置1，设置ICR寄存器的RFFULL位清除 0：无中断产生 1：接收器FIFO满中断产生
Reserved	Bit 10	—	**保留**
RFTH	Bit 9	R	**接收器FIFO触发阈值中断标志** 接收器FIFO个数达到接收器设定的阈值时置起，或设置IER寄存器的RFTH位时，UART会判断当前接收器FIFO个数达到接收器设定的阈值将此位置起。此位由硬件设置1，设置ICR寄存器的RFTH位清除 0：无中断产生 1：接收器FIFO触发阈值中断产生
NOISE	Bit 8	R	**噪声位检测中断标志** 此位由硬件设置1，设置ICR寄存器的NOISE位清除 0：无中断产生 1：噪声位检测中断产生
EOB	Bit 7	R	**块结束中断标志** 接收器接收个数等于SCARD寄存器的BLEN位时置起。此位由硬件设置1，设置ICR寄存器的EOB位清除 0：无中断产生 1：块结束中断产生
LINBK	Bit 6	R	**LIN断开检测中断标志** 在LIN模式中，当接收器检测到断开字符时置起。此位由硬件设置1，设置ICR寄存器的LINBK位清除 0：无中断产生 1：LIN断开检测中断产生
ADDRM	Bit 5	R	**地址匹配中断标志** 接收器接收到的字符匹配RS485寄存器的ADDR位时置起。此位由硬件设置1，设置ICR寄存器的

			ADDRM位清除 0：无中断产生 1：地址匹配中断产生
RXTO	Bit 4	R	**接收超时中断标志** 接收最后一个字节后，在超时时间内未检测到新的起始位时置起。此位由硬件设置1，设置ICR寄存器的RXTO位清除 0：无中断产生 1：接收超时中断产生
DCTS	Bit 3	R	**CTSn引脚电平改变中断标志** CTSn引脚电平改变时置起。此位由硬件设置1，设置ICR寄存器的DCTS位清除 0：无中断产生 1：CTSn引脚电平改变中断产生
ABTO	Bit 2	R	**自动波特率检测超时中断标志** 自动波特率在检测到下降沿后，在超时时间内未检测到上升沿时置起。此位由硬件设置1，设置ICR寄存器的ABTO位清除 0：无中断产生 1：自动波特率检测超时中断产生
ABEND	Bit 1	R	**自动波特率检测结束中断标志** 自动波特率检测完成时置起。此位由硬件设置1，设置ICR寄存器的ABEND位清除 0：无中断产生 1：自动波特率检测结束中断产生
RXBERR	Bit 0	R	**接收器字节格式错误中断标志** 接收器接收到的字符发生校验错误与帧错误时置起。此位由硬件设置1，设置ICR寄存器的RXBERR位清除 0：无中断产生 1：接收器字节格式错误中断产生

27. 5. 2. 16 UART中断标志屏蔽寄存器（UART_IFM）

UART 中断标志屏蔽寄存器（UART_IFM）

偏移地址：$3C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TFOVER	Reserved	TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	Reserved	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR

Reserved	Bit 31-19	—	保留
TFOVER	Bit 18	R	**发送器FIFO溢出中断屏蔽标志** 当IVS寄存器的TFOVER位为1时，发送器FIFO内已有16个数据时，有新数据再次写入FIFO中时，将会置起此位并舍弃新数据。此位由硬件设置1，设置ICR寄存器的TFOVER位清除 0：无中断产生 1：发送器FIFO溢出中断产生
Reserved	Bit 17	—	保留
TFEMPTY	Bit 16	R	**发送器FIFO空中断屏蔽标志** 当IVS寄存器的TFEMPTY位为1时，发送器FIFO由一个数值转为空时置起，或设置IER寄存器的TFEMPTY位时，UART会判断当前发送器FIFO是否为空将此位置起。此位由硬件设置1，设置ICR寄存器的TFEMPTY位清除 0：无中断产生 1：发送器FIFO空中断产生
TFTH	Bit 15	R	**发送器FIFO触发阈值中断屏蔽标志** 当IVS寄存器的TFTH位为1时，发送器FIFO个数达到发送器设定的阈值时置起，或设置IER寄存器的TFTH位时，UART会判断当前发送器FIFO个数达到发送器设定的阈值将此位置起。此位由硬件设置1，设置ICR寄存器的TFTH位清除 0：无中断产生 1：发送器FIFO触发阈值中断产生
TBC	Bit 14	R	**发送器字节完成中断屏蔽标志** 当IVS寄存器的TBC位为1时，发送器完成单个字节时置起。此位由硬件设置1，设置ICR寄存器的TBC位清除 0：无中断产生 1：发送器字节完成中断产生
RFUERR	Bit 13	R	**接收器FIFO下溢中断屏蔽标志** 当IVS寄存器的RFUERR位为1时，接收器FIFO内

			无数据时，又再次读取接收器FIFO时置起。此位由硬件设置1，设置ICR寄存器的RFUERR位清除 0：无中断产生 1：接收器FIFO下溢中断产生
RFOERR	Bit 12	R	**接收器FIFO溢出中断屏蔽标志** 当IVS寄存器的RFOERR位为1时，接收器FIFO内已有16个数据时，有新数据再次接收至FIFO中时置起并舍弃新数据。此位由硬件设置1，设置ICR寄存器的RFOERR位清除 0：无中断产生 1：接收器FIFO溢出中断产生
RFFULL	Bit 11	R	**接收器FIFO满中断屏蔽标志** 当IVS寄存器的RFFULL位为1时，接收器FIFO内由15个数据接收至16个数据时置起，或设置IER寄存器的RFFULL位时，UART会判断当前接收器FIFO是否为16个数据而置起。此位由硬件设置1，设置ICR寄存器的RFFULL位清除 0：无中断产生 1：接收器FIFO满中断产生
Reserved	Bit 10	—	**保留**
RFTH	Bit 9	R	**接收器FIFO触发阈值中断屏蔽标志** 当IVS寄存器的RFTH位为1时，接收器FIFO个数达到接收器设定的阈值时置起，或设置IER寄存器的RFTH位时，UART会判断当前接收器FIFO个数达到接收器设定的阈值将此位置起。此位由硬件设置1，设置ICR寄存器的RFTH位清除 0：无中断产生 1：接收器FIFO触发阈值中断产生
NOISE	Bit 8	R	**噪声位检测中断屏蔽标志** 当IVS寄存器的NOISE位为1时，如噪声位检测中断产生该位置起，此位由硬件设置1，设置ICR寄存器的NOISE位清除 0：无中断产生 1：噪声位检测中断产生
EOB	Bit 7	R	**块结束中断屏蔽标志** 当IVS寄存器的EOB位为1时，接收器接收个数等于SCARD寄存器的BLEN位时置起。此位由硬件设置1，设置ICR寄存器的EOB位清除 0：无中断产生 1：块结束中断产生
LINBK	Bit 6	R	**LIN断开检测中断屏蔽标志** 当IVS寄存器的LINBK位为1时，LIN模式中，当接收器检测到断开字符时置起。此位由硬件设置1，设置ICR寄存器的LINBK位清除

			0：无中断产生 1：LIN断开检测中断产生
ADDRM	Bit 5	R	**地址匹配中断屏蔽标志** 当IVS寄存器的ADDRM位为1时，接收器接收到的字符匹配RS485寄存器的ADDR位时置起。此位由硬件设置1，设置ICR寄存器的ADDRM位清除 0：无中断产生 1：地址匹配中断产生
RXTO	Bit 4	R	**接收超时中断屏蔽标志** 当IVS寄存器的RXTO位为1时，接收最后一个字节后，在超时时间内未检测到新的起始位时置起。此位由硬件设置1，设置ICR寄存器的RXTO位清除 0：无中断产生 1：接收超时中断产生
DCTS	Bit 3	R	**CTSn引脚电平改变中断屏蔽标志** 当IVS寄存器的DCTS位为1时，CTSn引脚电平改变时置起。此位由硬件设置1，设置ICR寄存器的DCTS位清除 0：无中断产生 1：CTSn引脚电平改变中断产生
ABTO	Bit 2	R	**自动波特率检测超时中断屏蔽标志** 当IVS寄存器的ABTO为1时，自动波特率在检测到下降沿后，在超时时间内未检测到上升沿时置起。此位由硬件设置1，设置ICR寄存器的ABTO位清除 0：无中断产生 1：自动波特率检测超时中断产生
ABEND	Bit 1	R	**自动波特率检测结束中断屏蔽标志** 当IVS寄存器的ABEND位为1时，自动波特率检测完成时置起。此位由硬件设置1，设置ICR寄存器的ABEND位清除 0：无中断产生 1：自动波特率检测结束中断产生
RXBERR	Bit 0	R	**接收器字节格式错误中断屏蔽标志** 当IVS寄存器的RXBERR位为1时，接收器接收到的字符发生校验错误与帧错误时置起。此位由硬件设置1，设置ICR寄存器的RXBERR位清除 0：无中断产生 1：接收器字节格式错误中断产生

27. 5. 2. 17 UART中断清除寄存器（UART_ICR）

UART 中断清除寄存器（UART_ICR）

偏移地址：40_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TFOVER	Reserved	TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	Reserved	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR

Reserved	Bit 31-19	—	保留
TFOVER	Bit 18	C_W1	**发送器FIFO溢出中断清除** 0：写入0无效 1：发送器FIFO溢出中断清除
Reserved	Bit 17	—	保留
TFEMPTY	Bit 16	C_W1	**发送器FIFO空中断清除** 0：写入0无效 1：发送器FIFO空中断清除
TFTH	Bit 15	C_W1	**发送器FIFO触发阈值中断清除** 0：写入0无效 1：发送器FIFO触发阈值中断清除
TBC	Bit 14	C_W1	**发送器字节完成中断清除** 0：写入0无效 1：发送器字节完成中断清除
RFUERR	Bit 13	C_W1	**接收器FIFO下溢中断清除** 0：写入0无效 1：接收器FIFO下溢中断清除
RFOERR	Bit 12	C_W1	**接收器FIFO溢出中断清除** 0：写入0无效 1：接收器FIFO溢出中断清除
RFFULL	Bit 11	C_W1	**接收器FIFO满中断清除** 0：写入0无效 1：接收器FIFO满中断清除
Reserved	Bit 10	—	保留
RFTH	Bit 9	C_W1	**接收器FIFO触发阈值中断清除** 0：写入0无效 1：接收器FIFO触发阈值中断清除
NOISE	Bit 8	C_W1	**噪声位检测中断清除** 0：写入0无效 1：噪声位检测中断清除
EOB	Bit 7	C_W1	**块结束中断清除** 0：写入0无效

			1：块结束中断清除
LINBK	Bit 6	C_W1	**LIN断开检测中断清除** 0：写入0无效 1：LIN断开检测中断清除
ADDRM	Bit 5	C_W1	**地址匹配中断清除** 0：写入0无效 1：地址匹配中断清除
RXTO	Bit 4	C_W1	**接收超时中断清除** 0：写入0无效 1：接收超时中断清除
DCTS	Bit 3	C_W1	**CTSn引脚电平改变中断清除** 0：写入0无效 1：CTSn引脚电平改变中断清除
ABTO	Bit 2	C_W1	**自动波特率检测超时中断清除** 0：写入0无效 1：自动波特率检测超时中断清除
ABEND	Bit 1	C_W1	**自动波特率检测结束中断清除** 0：写入0无效 1：自动波特率检测结束中断清除
RXBERR	Bit 0	C_W1	**接收器字节格式错误中断清除** 0：写入0无效 1：接收器字节格式错误中断清除

第28章 基本扩展控制器局域网（BxCAN）

28. 1 概述

基本扩展 CAN（Basic Extended Controller Area Network）外设又称 bxCAN，可与 CAN 网络进行交互。该外设支持 2.0A 和 2.0B Active 版本的 CAN 协议规范，2.0A 版本的协议规范支持 11 位标准标识符，2.0B Active 版本的协议规范支持 11 位标准标识符和 29 位扩展标识符。

CAN 广泛应用于安全性较高的汽车电子和工业控制中，CAN 控制器支持 3 个优先级可配置的发送邮箱，2 个可以存储三级邮箱深度的接收 FIFO，同时硬件支持时间触发通信方案。

28. 2 特性

- 支持 2.0A 及 2.0B Active 版本的 CAN 协议规范
- 比特率高达 1Mbps
- 支持时间触发通信方案
- 在唯一地址空间通过软件实现高效的邮箱映射

发送

- 三个发送邮箱
- 可配置的发送优先级
- 支持发送中断

接收

- 两个具有三级邮箱深度的接收 FIFO
- 14 个可调整的筛选器组：
- 标识符列表功能
- 可配置的 FIFO 上溢
- 支持接收中断

时间触发通信方案

- 禁止自动重发模式
- 专用的 16 位定时器
- 支持发送时间戳和接收时间戳

28. 3 结构框图

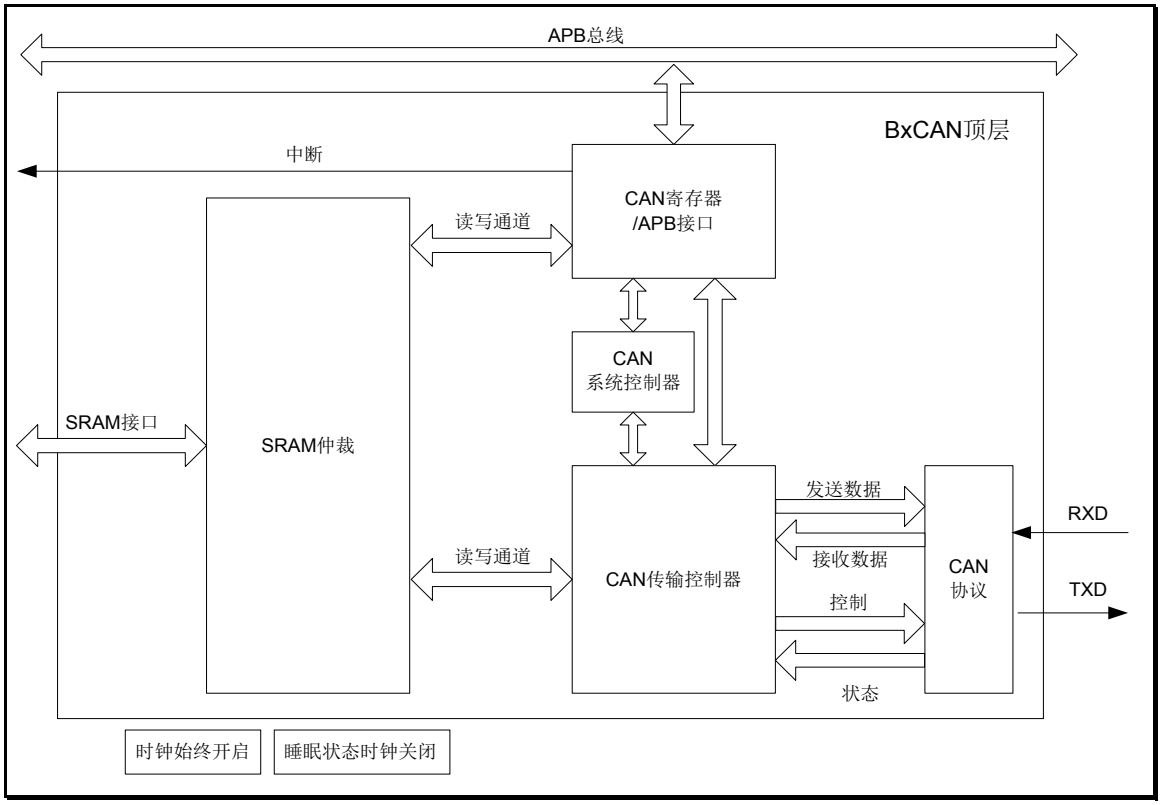


图 28-1 BxCAN 电路结构框图

28. 4 功能描述

28. 4. 1 简介

在如今的 CAN 应用中，网络节点数量日益增多，经常需要通过网关将数个网络连接在一起，使得系统中的消息数量（以及各个节点需要处理的消息）有了显著增加。除应用程序的消息外，还引入了网络管理和诊断消息。

此外，应用程序任务需要更多的 CPU 时间，因此必须减少因消息接收而对实时处理造成的限制。

◇ 需要一个增强的筛选机制对各种类型的消息进行处理。

◇ 接收 FIFO 方案使 CPU 能够长时间专门处理应用程序任务，又不致丢失消息。基于标准 CAN 驱动程序的标准 HLP（更高层协议）需要一个高效接口来与 CAN 控制器连接。

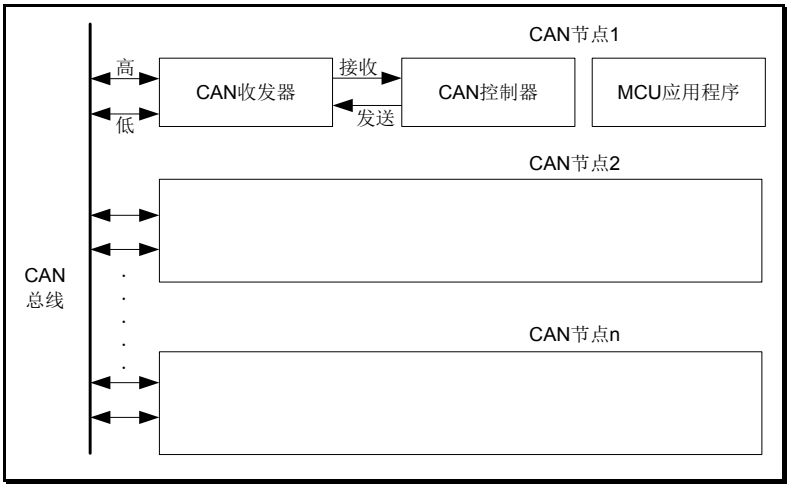


图 28-2 CAN 网络拓扑结构

28. 4. 1. 1 CAN 2.0B

bxCAN2.0B 规范要求 CAN 收发器支持扩展格式的标识符(29 位)，同时能够兼容标准标识符（11 位），且接收器可以自动识别出接收的是扩展帧还是标准帧。

28. 4. 1. 2 CAN消息存储

CAN 消息的软件与硬件之间的接口通过邮箱实现。邮箱中包含所有与消息相关的信息：标识符、数据、控制、状态和时间戳信息。

发送邮箱

软件在空发送邮箱中设置将要发送的消息。发送状态由硬件在 CAN_TSTAT 寄存器中指示。

与发送邮箱基地址之间的偏移	寄存器名称
0	CAN_TXIDx
4	CAN_TXFCONx
8	CAN_TXDLx
12	CAN_TXDHx

表 28-1 发送邮箱映射

接收邮箱

消息在接收到后，将放在接收 FIFO 邮箱中供软件使用。一旦软件对消息进行了处理（例如读取），则必须通过将 CAN_RXFx.FREE 位置 1 释放 FIFO 接收邮箱，以接收下一条传入消息。筛选器匹配索引存储在 CAN_RXFxINF.FLTIDX 中。16 位时间戳值存储在 CAN_RXFxINF.STAMP 字段中。

与接收邮箱基地址之间的偏移（字节）	寄存器名称
0	CAN_RXFxID
4	CAN_RXFxINF
8	CAN_RXFxDL
12	CAN_RXFxDH

表 28-2 接收邮箱映射

CAN 邮箱 SRAM 存储

CAN 的发送邮箱和接收邮箱都是存储在 SRAM 中，可以存储 3 个发送邮箱。硬件使用两个接收 FIFO 来存储传入消息。每个 FIFO 中可以存储三条完整消息，FIFO 完全由硬件管理，接收邮箱寄存器访问的是最早存入 FIFO 的消息。

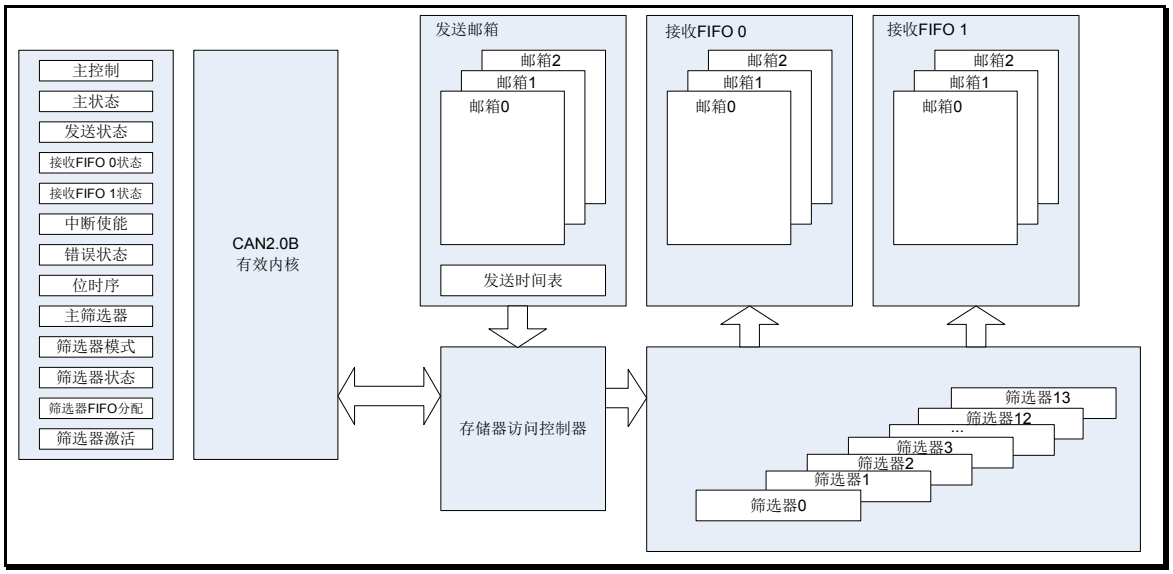


图 28-3 CAN SRAM 存储

28. 4. 1. 3 错误管理

如 CAN 协议所述，节点的故障状态分为错误主动、错误被动、总线关闭三种。

错误管理完全由硬件通过发送错误计数器（CAN_ERRSTAT.TXERRC 值）和接收错误计数器（CAN_ERRSTAT.RXERRC 值）来处理，这两个计数器根据错误状况进行递增或递减。

两者均可由软件读取，用以确定网络的稳定性。此外，CAN 硬件还将在 CAN_ERRSTAT 寄存器中提供当前错误状态的详细信息。通过 CAN_IE.ERRIE 位，软件可以非常灵活地配置在检测到错误时生成的中断。

总线关闭恢复

当 TXERRC 大于 255 时，节点变为“总线关闭”状态，该状态由 CAN_ERRSTAT.BOFF 位指示。在“总线关闭”状态下，bxCAN 不能再发送和接收消息。

bxCAN 可以自动或者由软件请求从“总线关闭”状态恢复为“错误主动”状态，具体取决于 CAN_CON.ABOFFEN 位。但在这两种情况下，bxCAN 都必须等待至少一个 CAN 标准恢复序列（在 CAN 总线上监测到 128 次 11 个连续隐性位）。

如果 CAN_CON.ABOFFEN=1，bxCAN 将在进入“总线关闭”状态后自动启动恢复序列。

如果 CAN_CON.ABOFFEN=0，则软件必须请求 bxCAN 先进入初始化模式再退出初始化（CAN_CON.INIREQ 置 1 再清零），从而启动恢复序列。

注：在初始化模式下，bxCAN 不会监视 CANRX 信号，因此无法完成恢复序列。要进行恢复，bxCAN 必须处于正常模式。

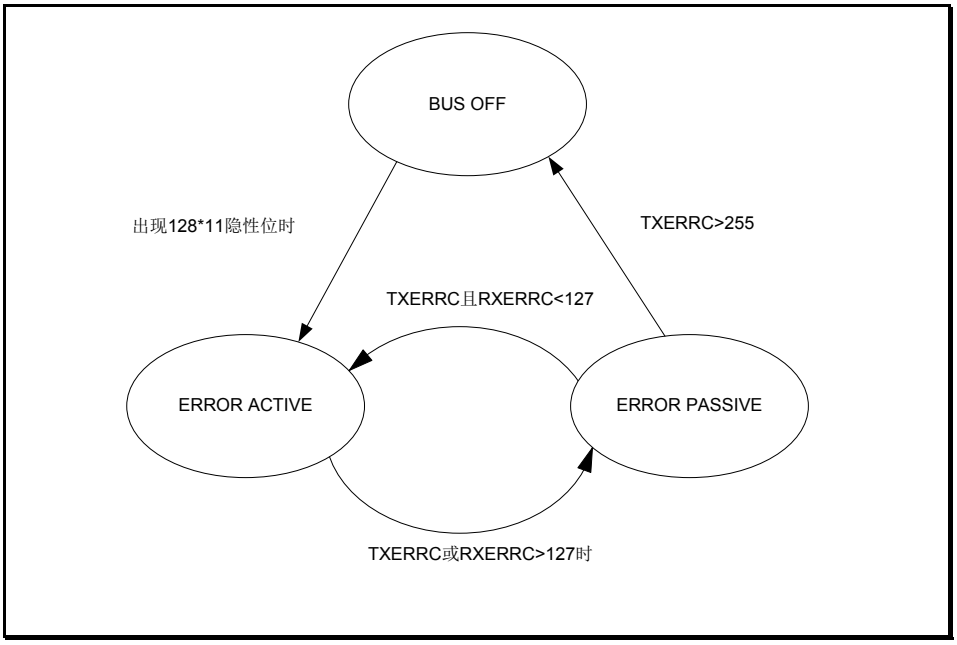


图 28-4 CAN 错误状态图

28. 4. 1. 4 位时序

位时序逻辑将监视 CAN 总线，执行采样并调整采样点，在调整采样点时，需要在起始位边沿进行同步并在后续的边沿进行再同步。

通过将标称位时间划分为以下三段，即可解释其工作过程：

- ◇ 同步段（**SYNC_SEG**）：位变化应该在此时间段内发生。它只有一个时间片的固定长度（1xt_{CAN}）。
- ◇ 位段 **1**（**SEG1**）：定义采样点的位置。它包括 CAN 标准的 PROP_SEG 和 PHASE_SEG1。其持续长度可以在 1 到 16 个时间片之间调整，但也可以自动加长，以补偿由不同网络节点的频率差异所导致的正相位漂移。
- ◇ 位段 **2**（**SEG2**）：定义发送点的位置。它代表 CAN 标准的 PHASE_SEG2。其持续长度可以在 1 到 8 个时间片之间调整，但也可以自动缩短，以补偿负相位漂移。

再同步跳转宽度（RESJW）定义位段加长或缩短的上限。它可以在 1 到 4 个时间片之间调整。

有效边沿是指一个位时间内总线电平从隐性到显性的第一次转换（前提是控制器本身不发送隐性位）。

如果在 SEG1 而不是 SYNC_SEG 中检测到有效边沿，则 SEG1 会延长最多 RESJW，以便延迟采样点。

相反地，如果在 SEG2 而不是 SYNC_SEG 中检测到有效边沿，则 SEG2 会缩短最多 RESJW，以便提前发送点。

为了避免编程错误，位时序寄存器（CAN_BTIME）只能在器件处于初始化模式时进行配置。

注：有关 CAN 位时序和再同步机制的详细说明，请参见 ISO11898 标准。

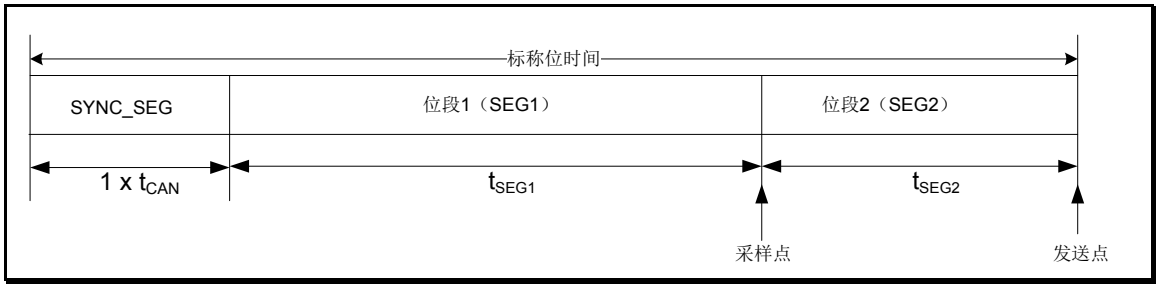


图 28-5 位时序

$$波特率 = \frac{1}{标称位时间}$$

$$标称位时间 = 1 \times t_{CAN} + t_{SEG1} + t_{SEG2}$$

其中：

$t_{SEG1} = t_{CAN} \times (SEG1+1)$

$t_{SEG2} = t_{CAN} \times (SEG2+1)$

$t_{CAN} = (BPSC[9:0]+1) \times t_{PCLK}$

BPSC[9:0]、SEG1 和 SEG2 在 CAN_BTIME 寄存器中的定义。

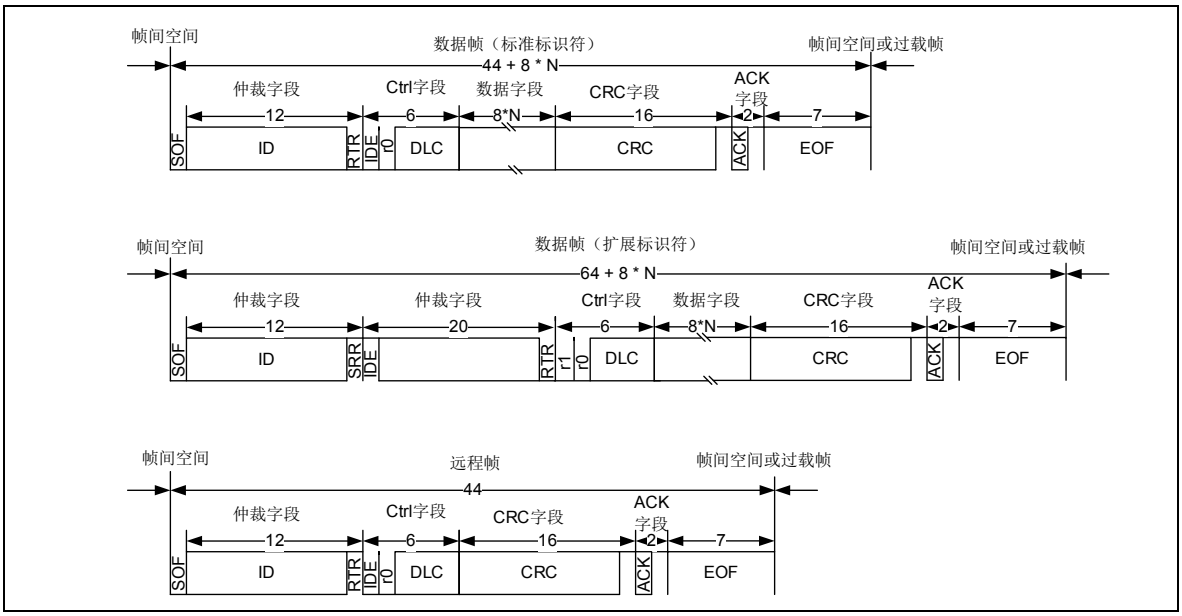


图 28-6 CAN 帧

注：0<=N<=8，SOF=帧起始，ID=标识符，RTR=远程传输请求，IDE=标识符扩展位，r0=保留位，DLC=数据长度代码，CRC=循环冗余代码，EOF=帧结束，ACK=确认位，Ctrl=控制，SRR=替代远程请求位，r1=保留位

28. 4. 2 工作模式

bxCAN 有三种主要的工作模式：初始化模式、正常模式和睡眠模式。

28. 4. 2. 1 初始化模式

在初始化模式下，所有从 CAN 总线传入和传出的消息都将停止，并且 CAN 总线输出 CANTX 的状态为隐性（高）。

通过将 CAN_CON.INIREQ 置 1 以进入初始化模式，进入初始化模式不会更改任何配置寄存器。

为初始化与 CAN 筛选器组相关的寄存器（模式、宽度、FIFO 分配、激活和筛选器值），软件必须将 CAN_FLTCON.FLTINI 置 1。当进入筛选器初始化模式时，CAN 接收停用。筛选器的初始化也可以在初始化模式之外进行。

筛选器值也可通过停用（CAN_FLTGO 寄存器）相关筛选器激活位来修改。

如果某个筛选器组未使用，建议将其保持为未激活状态（将相应 CAN_FLTGO.GO 位保持清零）。

28. 4. 2. 2 正常模式

一旦初始化完成，软件必须向硬件请求进入正常模式，这样才能在 CAN 总线上进行同步，并开始接收和发送。

进入正常模式的请求可通过 CAN_CON.INIREQ 清 0 来实现。bxCAN 进入正常模式，并与 CAN 总线上的数据传输实现同步后（RX 上检测到 11 个连续隐性位），即可参与总线活动。硬件通过将 CAN_STAT.INISTAT 清 0，来确认切换到正常模式。

筛选器的相关配置必须在进入正常模式之前完成。

28. 4. 2. 3 睡眠模式

为降低能耗功耗，bxCAN 具有低功耗模式，称为睡眠模式。在正常模式下软件通过将 CAN_CON.SLPREQ 置 1 发出请求，即可进入该模式。该模式下，bxCAN 时钟停止，但软件仍可访问 bxCAN 邮箱。

在 bxCAN 处于睡眠模式时，如果软件通过将 CAN_CON.INIREQ 置 1 来请求进入初始化模式，则必须同时将 CAN_CON.SLPREQ 位清零。

软件将 CAN_CON.SLPREQ 位清零或是检测到 CAN 总线活动时，bxCAN 即被唤醒（退出睡眠模式）。

检测到 CAN 总线活动后，如果 CAN_CON.AWKEN=1，硬件将通过清零 CAN_CON.SLPREQ 位来自动执行唤醒序列。如果 CAN_CON.AWKEN=0，在发生唤醒中断时，软件必须将 CAN_CON.SLPREQ 位清零才能退出睡眠模式。

注：如果使能唤醒中断（CAN_IE.WKIE=1），一旦检测到 CAN 总线活动，即使 bxCAN 自动执行唤醒序列，也会发生唤醒中断。

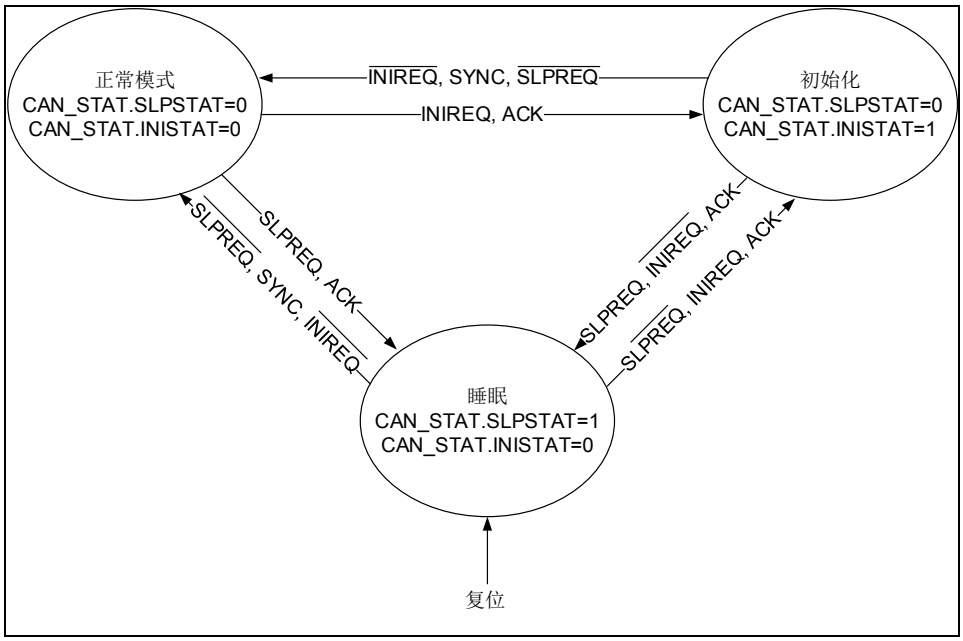


图 28-7 BxCAN 工作模式

◇ ACK 为硬件通过将 CAN_STAT 寄存器的 INISTAT 或 SLPSTAT 位置 1 来确认请求的等待状态。

◇ SYNC 为 bxCAN 等待 CAN 总线变为空闲（即在 CANRX 上监测到连续 11 个隐性位）的状态。

由睡眠模式进入初始化模式

1. 将寄存器 CAN_CON.SLPREQ=0
2. 将寄存器 CAN_CON.INIREQ=1
3. 等待寄存器 CAN_STAT.INISTAT=1

由初始化进入正常模式

1. 将寄存器 CAN_CON.INIREQ=0
2. 等待总线同步，CANRX 上检测到连续 11 个隐性位

由正常模式进入睡眠模式

1. 将寄存器 CAN_CON.SLPREQ=1
2. 等待寄存器 CAN_STAT.SLPSTAT=1

28. 4. 3 发送处理

28. 4. 3. 1 发送处理

bxCAN 提供三个发送邮箱，为了发送消息，应用程序必须在请求发送（CAN_TXIDx.TXMREQ=1）前，选择一个空发送邮箱，并设置标识符类型、数据长度（DLC）和要发送的数据。CAN_TXIDx.TXMREQ=1 时，邮箱变为非空状态，立即进入挂起态，软件不再具有对发送邮箱寄存器的写访问权限。接着该邮箱等待成为优先级最高的邮箱，请参见章节：发送优先级。一旦邮箱拥有最高优先级，即被安排发送。CAN 总线变为空闲后，被安排好的邮箱中的消息即开始发送（进入发送状态）。邮箱一旦发送成功，即恢复空状态。硬件通过将 CAN_TXSTAT 寄存器的 MxREQC 和 MxTXC 位置 1，来表示发送成功。

如果发送失败，失败原因将由 CAN_TXSTAT 寄存器的 MxARBLST 位（仲裁丢失）或 MxTXERR 位（检测到发送错误）指示。

28. 4. 3. 2 发送优先级

当多个发送邮箱挂起时，发送顺序可以按标识符决定或按发送请求顺序决定，具体取决于寄存器 CAN 主控制寄存器 CAN_CON.TXMP 位。

◇ 按标识符

当 CAN_CON.TXMP=0 时，发送顺序由邮箱中所存储消息的标识符来确定。根据 CAN 协议的仲裁，标识符值最小的消息具有最高的优先级。如果标识符值相等，则首先安排发送编号较小的邮箱。

◇ 按发送请求顺序

当 CAN_CON.TXMP=1 时，发送优先级顺序按照发送请求顺序来确定，先请求的邮箱优先发送。

28. 4. 3. 3 发送停止

处于挂起态、已安排状态或发送态的邮箱可以通过软件将其终止发送。

处于挂起态或已安排状态下的邮箱，将 CAN_TXSTAT.MxSTPREQ 置 1，发送请求即被终止。

处于发送态的邮箱，将 CAN_TSTAT.MxSTPREQ=1，可能会出现两种结果：如果邮箱发送成功，将变为空状态，同时 CAN_TSTAT.MxTXC=1；如果发送失败，邮箱变为已安排状态，发送中止并变为空状态，CAN_TSTAT.MxTXC=0。

28. 4. 3. 4 禁止自动重发送模式

该模式旨在满足 CAN 标准的时间触发通信方案的要求。要将硬件配置为此模式，必须将 CAN_CON. ARTXDIS 置 1。

在此模式下，每个发送仅启动一次。如果由于仲裁丢失或错误导致第一次尝试失败，硬件将不会自动重新启动消息发送。

第一次发送尝试结束时，硬件将认为请求已完成，并将 CAN_TXSTAT.MxREQC 置 1。发送结果由 CAN_TXSTAT 寄存器的 MxTXC、MxARBLST 和 MxTXERR 位来指示。

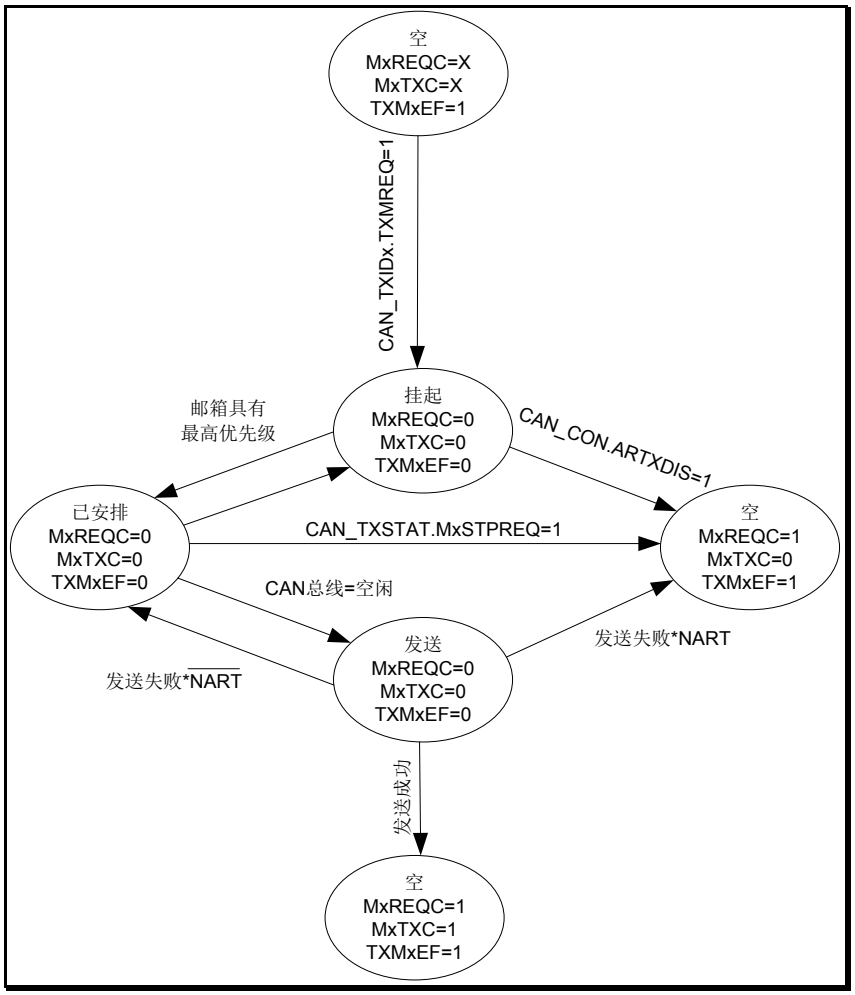


图 28-8 发送邮箱状态

28. 4. 3. 5 时间触发通信模式

在此模式下，CAN 硬件的内部计数器激活，用于为接收和发送邮箱生成时间戳值，这些值分别存储在寄存器 CAN_TXFCONx.STAMP 和 CAN_RXFxINF.STAMP 寄存器中。内部计数器在每个 CAN 位时间加 1（请参见章节：位时序）。在接收和发送时，都会在帧起始位的采样点捕获内部计数器。

28. 4. 3. 6 发送消息流程

1. 选择一个空发送邮箱，查找 CAN_TXSTAT.TXMxEF 为 1 的邮箱
2. 配置 4 个发送邮箱寄存器 CAN_TXIDx、CAN_TXFCONx、CAN_TXDLx、CAN_TXDHx
3. 将 CAN_TXIDx.TXMREQ 的位置 1
4. 检查发送状态寄存器 TXSTAT 的 MxREQC、MxTXC 和 TXMxEF 是否为 1

28. 4. 4 接收处理

bxCAN 提供了两个三级邮箱深度的接收 FIFO，为了节约 CPU 负载，简化软件并保证数据一致性，FIFO 完全由硬件进行管理。应用程序通过读取接收 FIFO 来获得最先存入 FIFO 的消息。

28. 4. 4. 1 有效消息

当消息依据 CAN 协议正确接收（没有发送错误），并且成功通过了标识符筛选后，该消息被视为有效。有关标识符筛选请参见章节：标识符筛选器。

28. 4. 4. 2 FIFO管理

FIFO 开始时处于空状态，在接收的第一条有效消息存储在其中后，变为 Pending_1 状态。硬件通过将 CAN_RXFx.PEND 置 1 来指示该事件。消息将放在接收 FIFO 中供软件读取。软件读取邮箱内容后，通过将 CAN_RXFx.FREE 置 1 将邮箱释放，该接收 FIFO 便会恢复空状态。如果同时接收到新的有效消息，FIFO 将保持 Pending_1 状态，新消息将在输出邮箱中供读取。

如果应用程序未释放邮箱，下一条有效消息将继续存储在 FIFO 中，使其进入 Pending_2 状态（CAN_RXFx.PEND=2）。下一条有效消息会重复该存储过程，同时将 FIFO 变为 Pending_3 状态（CAN_RXFx.PEND=3）。此时，软件必须通过将 CAN_RXFx.FREE 置 1 来释放输出邮箱，从而留出一个空邮箱来存储下一条有效消息。否则，下一次接收到有效消息时，将导致消息丢失。

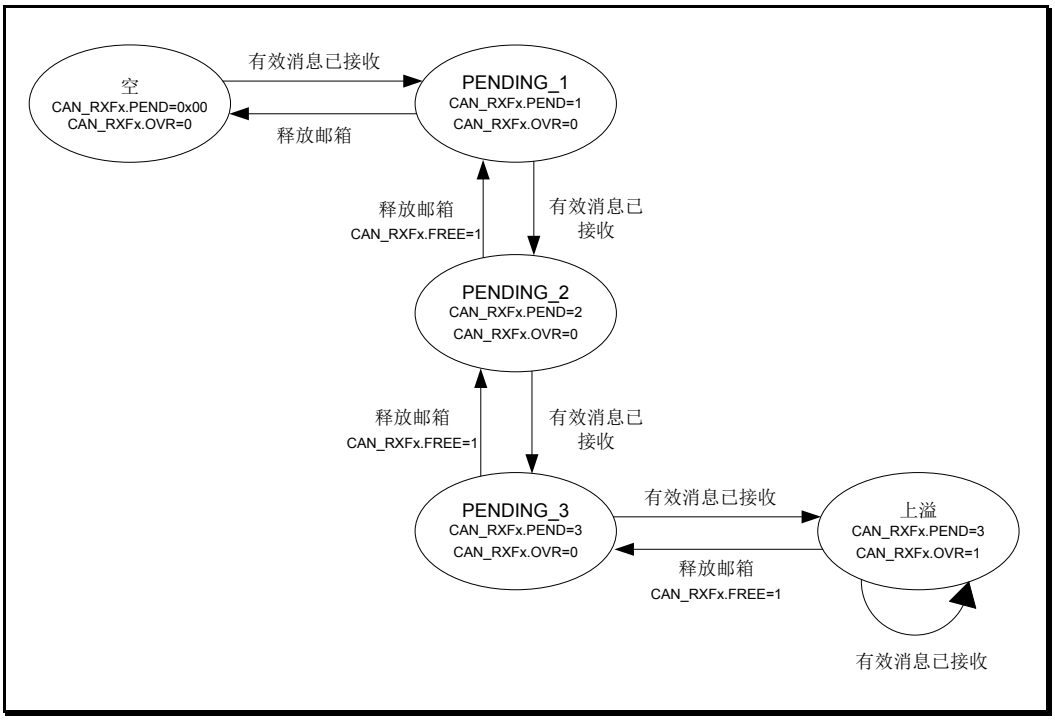


图 28-9 接收 FIFO 状态

28. 4. 4. 3 上溢

一旦 FIFO 处于 Pending_3 状态（即三个邮箱均已满），则下一次接收到有效消息时，将导致上溢并丢失一条消息。硬件通过将 CAN_RXFx.OVR 置 1 来指示上溢状态。具体丢失哪一条消息取决于 FIFO 的配置：

◇ 如果禁止 FIFO 锁定功能（CAN_CON.RXFOPM=0），则新传入的消息将覆盖 FIFO 中存储的最后一条消息。在这种情况下，应用程序将始终能访问到最新的消息。

◇ 如果使能 FIFO 锁定功能（CAN_CON.RXFOPM=1），则将丢弃最新的消息，软件将提供 FIFO 中最早的三条消息。

28. 4. 4. 4 接收FIFO中断

当使能消息挂起中断 CAN_IE.FxPIE 后，收到消息存储到 FIFO 中，CAN_RXFx.PEND 位非 0，即产生中断请求。

当使能 FIFO 满中断 CAN_IE.FxFULIE 后，FIFO 存满消息（即存储了第三条消息）后，CAN_RXFx.FULL 为 1，将产生满中断。

当使能 FIFIO 溢出中断 CAN_IE.FxOVRIE 后，出现上溢时，CAN_RXFx.OVR 位置 1，将产生溢出中断。

28. 4. 4. 5 接收消息流程

1. 等待接收 FIFO 寄存器 CAN_RXFx.PEND 非 0

2. 读取相关接收 FIFO 邮箱寄存器：CAN_RXFxID、CAN_RXFxINF、CAN_RXFxDL 和 CAN_RXFxDH

3. 置位 CAN_RXFx.FREE 释放接收 FIFO

28. 4. 5 标识符筛选器

在 CAN 协议中，消息的标识符与节点地址无关，但与消息内容有关。因此，发送器将消息广播给所有接收器。在接收到消息时，接收器节点会根据标识符的值来确定软件是否需要该消息。如果需要，该消息将复制到 SRAM 中。如果不需要，硬件自动丢弃该消息。

为了实现这一功能，bxCAN 控制器为应用程序提供了 14 个可配置且可调整的硬件筛选器组（0-13），以便接收器仅接收软件需要的消息。此硬件筛选功能可以节省软件筛选所需的 CPU 资源。每个筛选器组 x 均包含两个 32 位寄存器，分别是 CAN_FLTxR1 和 CAN_FLTxR2。

28. 4. 5. 1 可调整的宽度

为了根据应用程序的需求来优化和调整筛选器，每个筛选器组可分别进行伸缩调整。根据筛选器宽度不同，一个筛选器组可以：

◇ 为 STDID[10:0]、EXTID[17:0]、IDE 和 RTR 位提供一个 32 位筛选器。

◇ 为 STDID[10:0]、RTR、IDE 和 EXTID[17:15]位提供两个 16 位筛选器。

此外，筛选器还可配置为掩码模式或标识符列表模式。

28. 4. 5. 2 掩码模式

在掩码模式下，标识符寄存器与掩码寄存器关联，用以指示标识符的哪些位“必须匹配”，哪些位“无关”，CAN_FLTxR1 设定匹配 ID，CAN_FLTxR2 设定哪些位无需匹配。接收器可以接收标识符中“必须匹配位”所匹配的所有消息。

28. 4. 5. 3 标识符列表模式

在标识符列表模式下，CAN_FLTxR1 和 CAN_FLTxR2 都用于设定匹配 ID，传入标识符的所有位都必须与筛选器寄存器匹配才可以被接收器接收。

28. 4. 5. 4 筛选器组宽度和模式配置

筛选器组通过相应的 CAN_FLTCON 寄存器进行配置。为了配置筛选器组，必须通过将筛选器启用寄存器 CAN_FLTGO 的对应位清零而将其停用。筛选器宽度通过筛选器宽度选择寄存器 CAN_FLTWS 的对应位进行配置。相应掩码/标识符寄存器的标识符列表或标识符掩码模式通过筛选器模式寄存器 CAN_FLTM 的对应位进行配置。

要筛选一组标识符，应将掩码/标识符寄存器配置为掩码模式。

要选择单个标识符，应将掩码/标识符寄存器配置为标识符列表模式。

未由应用程序使用的筛选器应保持停用。

筛选器组中的每个筛选器将按从 0 到最大值的顺序进行编号（称为筛选器索引号），具体取决于每个筛选器组的模式和宽度。

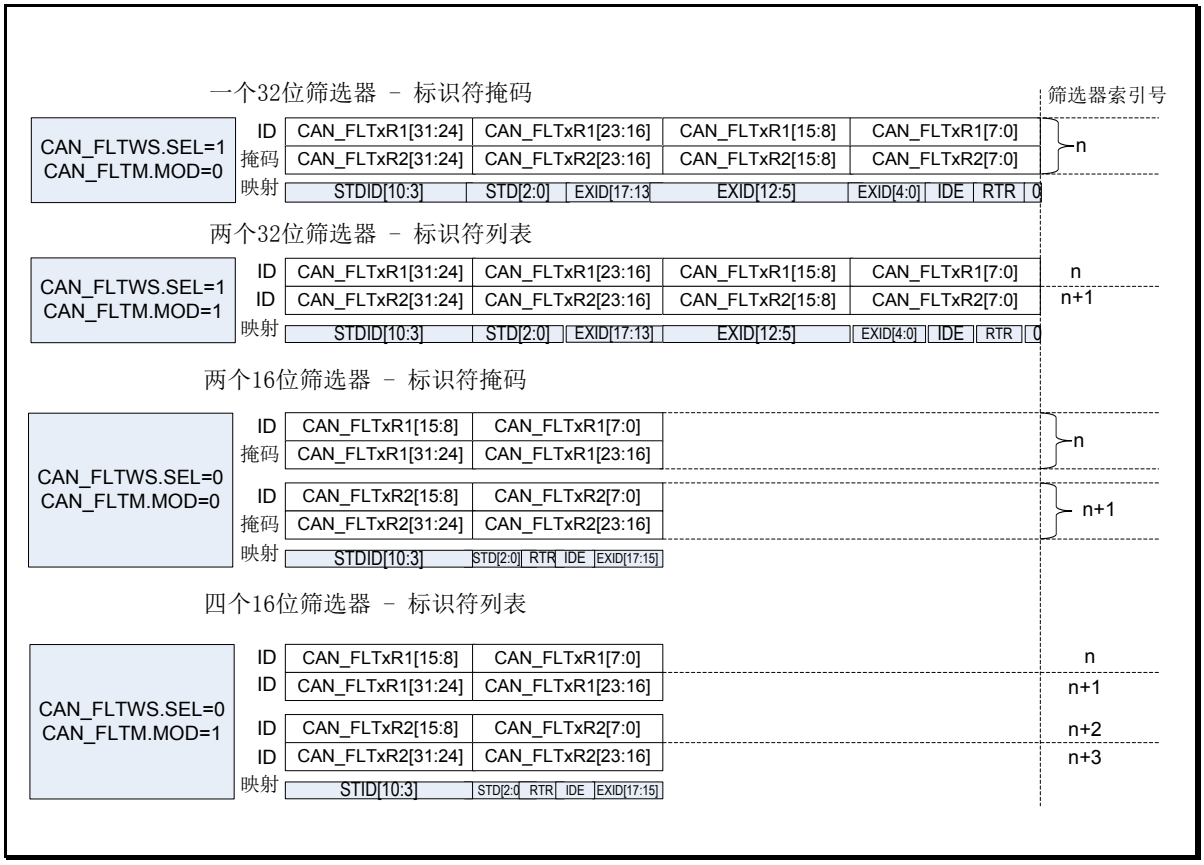


图 28-10 筛选器组宽度配置寄存器构成

注 1：上图中，x=筛选器组编号，ID=标识符。

注 2：筛选标准 ID 时需要注意：若工作在标识符列表模式时，EXID 部分需要设为 0；若工作在掩码模式时，如果掩码没有屏蔽 EXID 部分时，EXID 部分需要设为 0

28. 4. 5. 5 筛选器匹配索引

消息接收到 FIFO 中后，即可供应用程序使用。应用程序通常会复制到 SRAM 中的位置。为了将数据复制到正确的位置，应用程序必须通过标识符来识别数据。为了方便访问 SRAM 位置，CAN 控制器提供了一个筛选器匹配索引。

该索引根据筛选器优先级规则与消息一同存储在邮箱中。因此，每条收到的消息都有相关联的筛选器匹配索引。

筛选器匹配索引的使用方法有两种：

◇ 将筛选器匹配索引与预期值列表进行比较。

◇ 将筛选器匹配索引用作阵列索引，以访问数据目标位置。

对于标识符列表筛选器，软件不再需要比较标识符。

对于掩码模式筛选器，软件则只需比较屏蔽位。

筛选器编号的索引值与筛选器组的激活状态无关。此外，两个 FIFO 使用两个独立的编号方案，每个 FIFO 各一个。

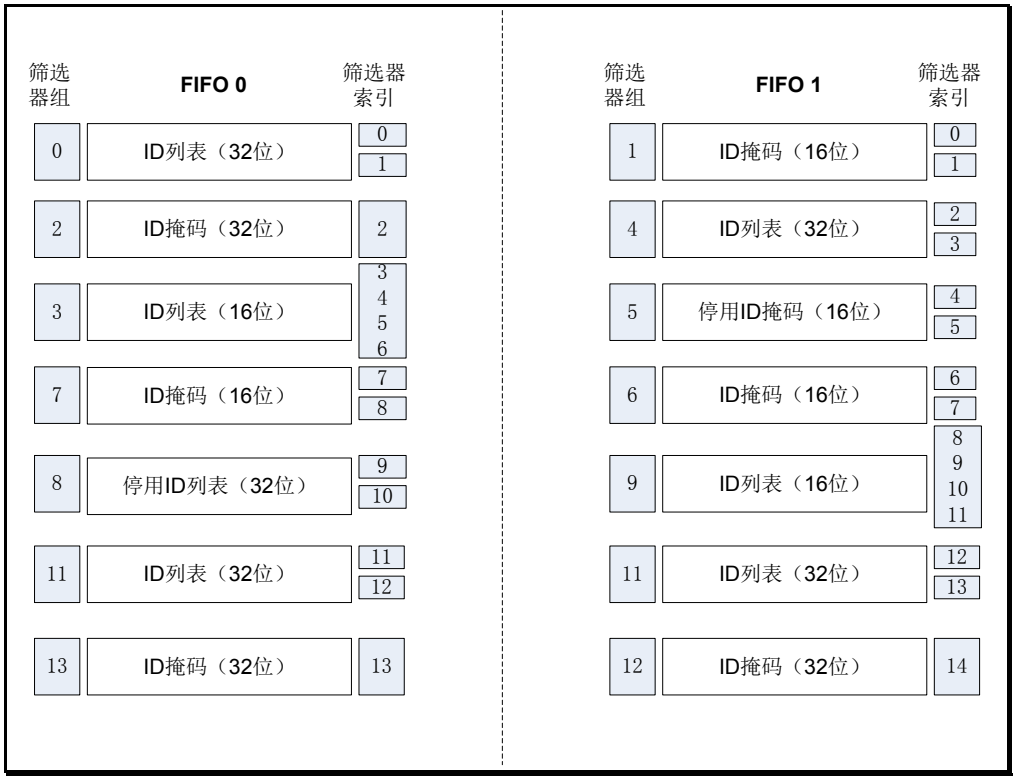


图 28-11 筛选器编号示例

28. 4. 5. 6 筛选器优先级规则

根据筛选器组合，可能会出现一个标识符成功通过数个筛选器的情况。这种情况下，将根据以下优先级规则选择接收邮箱中存储的筛选器匹配值：

- ◇ 32 位筛选器优先于 16 位筛选器。
- ◇ 对于宽度相等的筛选器，标识符列表模式优先于标识符掩码模式。
- ◇ 对于宽度和模式均相等的筛选器，则按筛选器编号确定优先级（编号越低，优先级越高）。

28. 4. 6 测试模式

测试模式用于 CAN 设备的调试与自检，bxCAN 在初始化模式时设置 CAN_BTTIME 寄存器中的 SILENT 和 LOOP 位来进入测试模式。选择测试模式后，必须清除 CAN_CON 寄存器中的 INIREQ 位才能进入正常模式。

28. 4. 6. 1 静默模式

在初始化模式时将 CAN_BTIME 寄存器的 SILENT 位置 1，bxCAN 进入静默模式。

在静默模式下，bxCAN 可以有效接收总线上的数据帧和远程帧。但 CAN 发送通道与 CAN 总线断开，无法向 CAN 总线发出显性位，bxCAN 发送的显性位仍可以被 CAN 内核监视。静默模式下，bxCAN 发送对总线保持隐性状态，通常用于分析 CAN 总线上的流量。

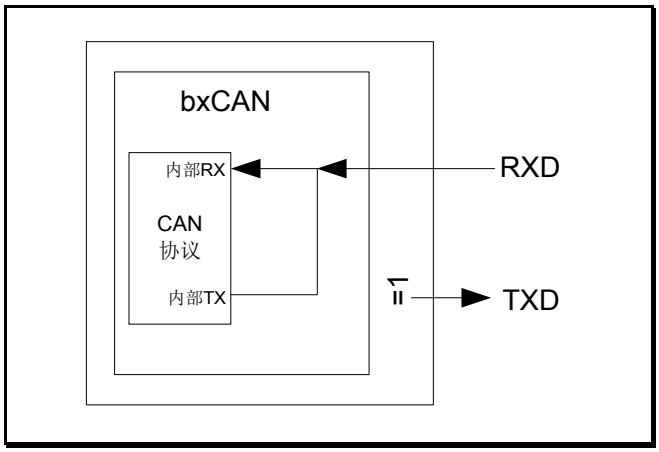


图 28-12 静默模式下的 bxCAN

28. 4. 6. 2 回环模式

在初始化模式时将 CAN_BTIME.LOOP 位置 1，bxCAN 进入回环模式。

在回环模式下，bxCAN 将其自身发送的消息作为接收的消息来处理并存储（如果这些消息通过了验收筛选）在接收 FIFO 中，同时发送的消息会进入 CAN 总线网络。

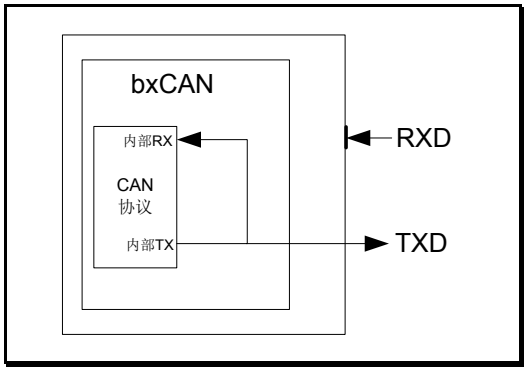


图 28-13 回环模式下的 bxCAN

28. 4. 6. 3 回环与静默组合模式

在初始化模式时将 CAN_BTIME 寄存器的 LOOP 和 SILENT 位置 1，bxCAN 进入回环与静默组合模式。

该模式可用于“热自检”，也就是说，bxCAN 的发送通道和接收通道与总线完全断开，bxCAN 即不受 CAN 总线影响，也不影响 CAN 总线。bxCAN 发送的数据可以被 CAN 内核自己接收。

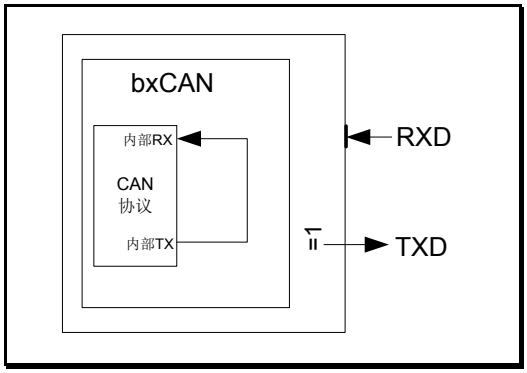


图 28-14 回环与静默组合模式下的 bxCAN

28. 4. 7 调试模式

当微控制器进入调试模式时，bxCAN 可以继续正常工作，也可以停止工作，具体取决于如下寄存器的指定位的值：

- ◇ DBG 模块中 APB1 外设调试冻结寄存器的 DBG_APB1FZ 的 CAN_STOP 位。
- ◇ CAN_CON.DBGSTP 位。

28.4.8 中断

bxCAN 共有四个专用的中断向量：发送中断、FIFO0 中断、FIFO1 中断和状态改变错误中断。每个中断源均可通过 CAN 中断使能寄存器（CAN_IE）来单独地使能或禁止。

◇ 发送中断可由以下事件产生：

- 发送邮箱 0 变为空，CAN_TXSTAT.M0REQC 位置 1
- 发送邮箱 1 变为空，CAN_TXSTAT.M1REQC 位置 1
- 发送邮箱 2 变为空，CAN_TXSTAT.M2REQC 位置 1

◇ **FIFO0** 中断可由以下事件产生：

- 接收到新消息，CAN_RXF0.PEND 位非 0
- FIFO0 满，CAN_RXF0.FULL 位置 1
- FIFO0 上溢，CAN_RXF0.OVR 位置 1

◇ **FIFO1** 中断可由以下事件产生：

- 接收到新消息，CAN_RXF1.PEND 位非 0
- FIFO1 满，CAN_RXF1.FULL 位置 1
- FIFO1 上溢，CAN_RXF1.OVR 位置 1

◇ 状态改变和错误中断可由以下事件产生：

- 唤醒状况，CAN Rx 信号上监测到 SOF
- 进入睡眠模式
- 错误状况，有关错误状况的更多详细信息，请参见 CAN 错误状态寄存器（CAN_ERRSTAT）

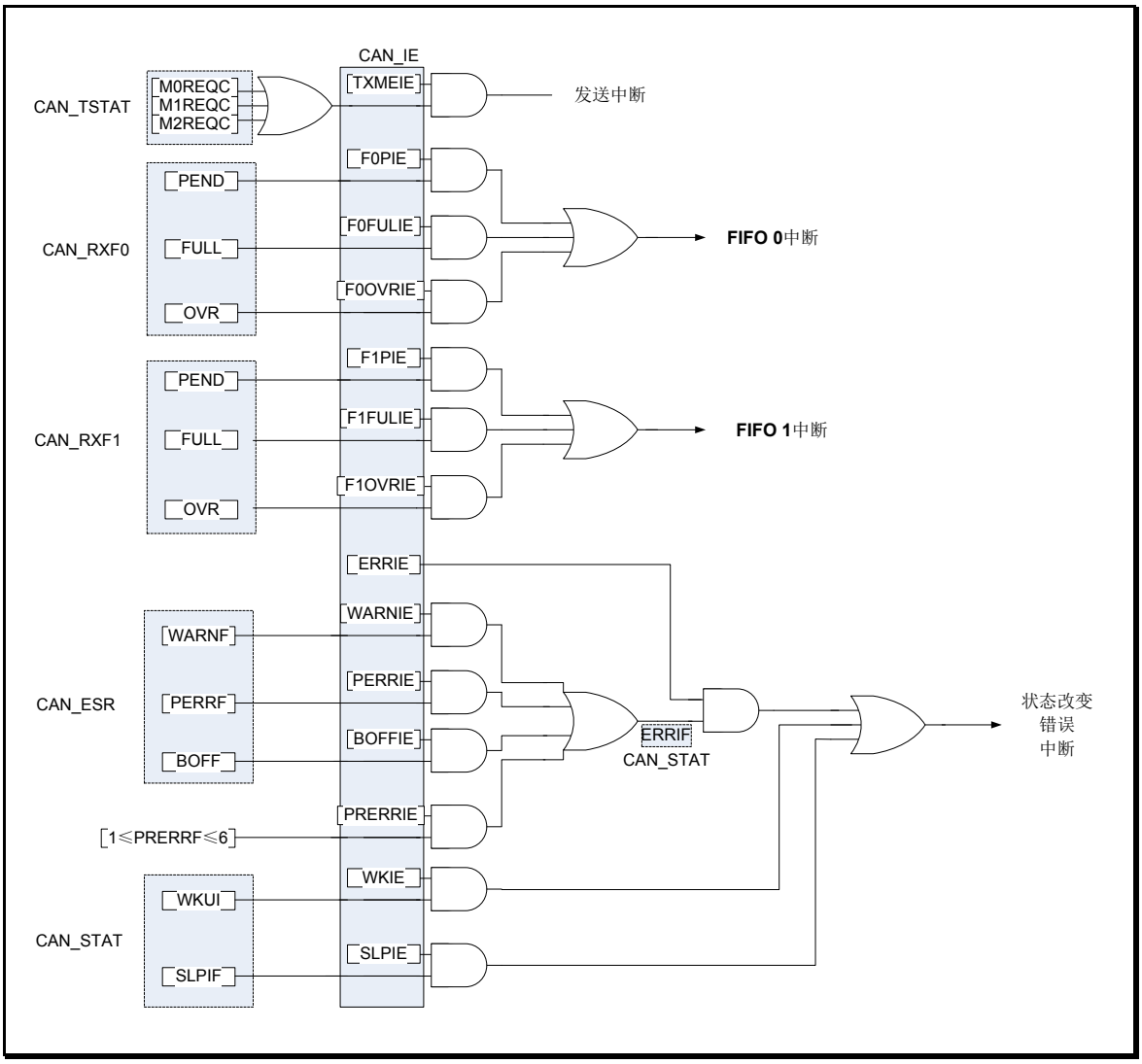


图 28-15　事件标志与中断产生

28. 5 特殊功能寄存器

28. 5. 1 寄存器列表

CAN 寄存器列表			
名称	偏移地址	类型	描述
CAN_CON	0000_H	R/W	CAN 控制寄存器
CAN_STAT	0004_H	R	CAN 状态寄存器
CAN_IFC	0008_H	W	CAN 中断标志清零寄存器
CAN_TXSTAT	$000C_H$	R/W	CAN 发送状态寄存器
CAN_TXSTATC	0010_H	W	CAN 发送状态清零寄存器
CAN_RXF0	0014_H	R/W	CAN 接收 FIFO0 寄存器
CAN_RXF0C	0018_H	W	CAN 接收 FIFO0 状态清零寄存器
CAN_RXF1	$001C_H$	R/W	CAN 接收 FIFO1 寄存器
CAN_RXF1C	0020_H	R	CAN 接收 FIFO1 状态清零寄存器
CAN_IE	0024_H	R/W	CAN 中断使能寄存器
CAN_ERRSTAT	0028_H	R/W	CAN 错误状态寄存器
CAN_BTIME	$002C_H$	R/W	CAN 位时序寄存器
Reserved	0030_H~$017F_H$	-	保留
CAN_TXID0	0180_H	R/W	CAN 发送邮箱标识符寄存器 0
CAN_TXFCON0	0184_H	R/W	CAN 发送邮箱帧控制寄存器 0
CAN_TXDL0	0188_H	R/W	CAN 发送邮箱数据低位寄存器 0
CAN_TXDH0	$018C_H$	R/W	CAN 发送邮箱数据高位寄存器 0
CAN_TXID1	0190_H	R/W	CAN 发送邮箱标识符寄存器 1
CAN_TXFCON1	0194_H	R/W	CAN 发送邮箱帧控制寄存器 1
CAN_TXDL1	0198_H	R/W	CAN 发送邮箱数据低位寄存器 1
CAN_TXDH1	$019C_H$	R/W	CAN 发送邮箱数据高位寄存器 1
CAN_TXID2	$01A0_H$	R/W	CAN 发送邮箱标识符寄存器 2
CAN_TXFCON2	$01A4_H$	R/W	CAN 发送邮箱帧控制寄存器 2
CAN_TXDL2	$01A8_H$	R/W	CAN 发送邮箱数据低位寄存器 2
CAN_TXDH2	$01AC_H$	R/W	CAN 发送邮箱数据高位寄存器 2
CAN_RXF0ID	$01B0_H$	R	CAN 接收 FIFO0 邮箱标识符寄存器
CAN_RXF0INF	$01B4_H$	R	CAN 接收 FIFO0 邮箱数据信息寄存器
CAN_RXF0DL	$01B8_H$	R	CAN 接收 FIFO0 邮箱数据低位寄存器
CAN_RXF0DH	$01BC_H$	R	CAN 接收 FIFO0 邮箱数据高位寄存器
CAN_RXF1ID	$01C0_H$	R	CAN 接收 FIFO1 邮箱标识符寄存器
CAN_RXF1INF	$01C4_H$	R	CAN 接收 FIFO1 邮箱信息寄存器
CAN_RXF1DL	$01C8_H$	R	CAN 接收 FIFO1 邮箱数据低位寄存器
CAN_RXF1DH	$01CC_H$	R	CAN 接收 FIFO1 邮箱数据高位寄存器
Reserved	$01D0_H$~$01FF_H$	-	保留
CAN_FLTCON	0200_H	R/W	CAN 筛选器控制寄存器
CAN_FLTM	0204_H	R/W	CAN 筛选器模式寄存器
Reserved	0208_H	-	保留

CAN 寄存器列表			
名称	偏移地址	类型	描述
CAN_FLTWS	$020C_H$	R/W	CAN 筛选器宽度选择寄存器
Reserved	0210_H	-	保留
CAN_FLTAS	0214_H	R/W	CAN 筛选器分配寄存器
Reserved	0218_H	-	保留
CAN_FLTGO	$021C_H$	R/W	CAN 筛选器启用寄存器
Reserved	0220_H~$023F_H$	-	保留
CAN_FLT0R1	0240_H	R/W	筛选器组 0 寄存器 1
CAN_FLT0R2	0244_H	R/W	筛选器组 0 寄存器 2
CAN_FLT1R1	0248_H	R/W	筛选器组 1 寄存器 1
CAN_FLT1R2	$024C_H$	R/W	筛选器组 1 寄存器 2
CAN_FLT2R1	0250_H	R/W	筛选器组 2 寄存器 1
CAN_FLT2R2	0254_H	R/W	筛选器组 2 寄存器 2
CAN_FLT3R1	0258_H	R/W	筛选器组 3 寄存器 1
CAN_FLT3R2	$025C_H$	R/W	筛选器组 3 寄存器 2
CAN_FLT4R1	0260_H	R/W	筛选器组 4 寄存器 1
CAN_FLT4R2	0264_H	R/W	筛选器组 4 寄存器 2
CAN_FLT5R1	0268_H	R/W	筛选器组 5 寄存器 1
CAN_FLT5R2	$026C_H$	R/W	筛选器组 5 寄存器 2
CAN_FLT6R1	0270_H	R/W	筛选器组 6 寄存器 1
CAN_FLT6R2	0274_H	R/W	筛选器组 6 寄存器 2
CAN_FLT7R1	0278_H	R/W	筛选器组 7 寄存器 1
CAN_FLT7R2	$027C_H$	R/W	筛选器组 7 寄存器 2
CAN_FLT8R1	0280_H	R/W	筛选器组 8 寄存器 1
CAN_FLT8R2	0284_H	R/W	筛选器组 8 寄存器 2
CAN_FLT9R1	0288_H	R/W	筛选器组 9 寄存器 1
CAN_FLT9R2	$028C_H$	R/W	筛选器组 9 寄存器 2
CAN_FLT10R1	0290_H	R/W	筛选器组 10 寄存器 1
CAN_FLT10R2	0294_H	R/W	筛选器组 10 寄存器 2
CAN_FLT11R1	0298_H	R/W	筛选器组 11 寄存器 1
CAN_FLT11R2	$029C_H$	R/W	筛选器组 11 寄存器 2
CAN_FLT12R1	$02A0_H$	R/W	筛选器组 12 寄存器 1
CAN_FLT12R2	$02A4_H$	R/W	筛选器组 12 寄存器 2
CAN_FLT13R1	$02A8_H$	R/W	筛选器组 13 寄存器 1
CAN_FLT13R2	$02AC_H$	R/W	筛选器组 13 寄存器 2

28. 5. 2 寄存器描述

28. 5. 2. 1 CAN控制寄存器（CAN_CON）

CAN 主控制寄存器（CAN_CON）

偏移地址：00_H

复位值：$00000000\_00000001\_00000000\_00000010_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															DBGSTP	RST	Reserved								TTCEN	ABOFFEN	AWKEN	ARTXDIS	RXFOPM	TXMP	SLPREQ	INIREQ

Reserved	Bit 31-17	—	**保留**
DBGSTP	Bit 16	R/W	**调试停止** 0：调试期间 CAN 处于工作状态。 1：调试期间 CAN 处于停止状态。接收 FIFO 仍可正常访问/控制。
RST	Bit 15	R/W	**bxCAN 软件复位** 0：正常工作。 1：bxCAN 进行软件复位，复位后激活睡眠模式。此位自动复位为 0。
Reserved	Bit 14-8	—	**保留**
TTCEN	Bit 7	R/W	**时间触发通信使能** 0：禁止时间触发通信模式。 1：使能时间触发通信模式。
ABOFFEN	Bit 6	R/W	**自动退出总线关闭使能** 此位控制 CAN 硬件在退出总线关闭状态时的行为。 0：一旦监测到 128 次连续 11 个隐性位，并且软件将 CAN_CON.INIREQ 位先置 1 再清零，退出总线关闭状态。 1：一旦监测到 128 次连续 11 个隐性位，即通过硬件自动退出总线关闭状态。
AWKEN	Bit 5	R/W	**自动唤醒使能** 此位控制 CAN 硬件在睡眠模式下接收到消息时的行为。 0：在软件通过将 CAN_CON.SLPREQ 位清零发出请求后，退出睡眠模式。 1：一旦监测到 CAN 消息，即通过硬件自动退出睡眠模式。CAN_CON.SLPREQ 位和 CAN_STAT.SLPSTAT 位由硬件清零。
ARTXDIS	Bit 4	R/W	**自动重发禁止** 0：CAN 硬件将自动重发送消息，直到消息发送成功。

			1：无论发送结果如何（成功、错误或仲裁丢失），消息均只发送一次。
RXFOPM	Bit 3	R/W	**接收 FIFO 溢出处理模式** 0：接收 FIFO 装满后，下一条传入消息将覆盖前一条消息。 1：接收 FIFO 装满后，下一条传入消息将被丢弃。
TXMP	Bit 2	R/W	**发送邮箱优先级** 此位用于控制在几个邮箱同时挂起时的发送顺序。 0：优先级由消息标识符确定 1：优先级由请求顺序（时间顺序）确定
SLPREQ	Bit 1	R/W	**睡眠请求** 此位由软件置 1，用于请求 CAN 硬件进入睡眠模式。一旦当前 CAN 活动（发送或接收 CAN 帧）结束，即进入睡眠模式。 此位由软件清 0 时，将退出睡眠模式。 当 CAN_CON.AWKEN 位置 1 以及在 CAN RX 信号上检测到帧起始符（SOF）位时，硬件即将此位清零。 复位后，此位被置 1，CAN 启动睡眠模式。
INIREQ	Bit 0	R/W	**初始化请求** 软件通过将此位清零，来将硬件切换到正常模式。CAN 可由总线激活，一旦在总线信号上监测到连续 11 个隐性位 CAN 硬件即完成同步并准备进行发送和接收。 软件通过将此位置 1 来请求 CAN 硬件进入初始化模式。一旦当前 CAN 活动（发送或接收）结束，即进入初始化模式。 硬件通过将 CAN_STAT.INISTAT 位置 1 指示此事件。

28. 5. 2. 2 CAN状态寄存器（CAN_STAT）

CAN 主状态寄存器 （CAN_STAT）

偏移地址：04_H

复位值：$00000000\_00000000\_00001100\_00000010_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				RX	PRESMP	RXSTAT	TXSTAT	Reserved			SLPIF	WKIF	ERRIF	SLPSTAT	INISTAT

Reserved	Bit 31-12	—	**保留**
RX	Bit 11	R	**CAN Rx 信号** 监视 CAN_RX 引脚的实际值。
PRESMP	Bit 10	R	**前采样点** 上一个采样点的 RX 值（当前接收的位值）。
RXSTAT	Bit 9	R	**接收状态** CAN 硬件当前为接收器。
TXSTAT	Bit 8	R	**发送状态** CAN 硬件当前为发送器。
Reserved	Bit 7-5	—	**保留**
SLPIF	Bit 4	R	**睡眠中断标志** 当 CAN 进入睡眠工作模式时，该位标志被置 1。 如果 CAN_IE.SLPIE=1 时，则当此位置 1 后将产生中断。此位通过 CAN_IFC.SLPIFC=1 清零。
WKIF	Bit 3	R	**唤醒中断标志** 此位由硬件置 1，用于指示在 CAN 硬件处于睡眠模式期间检测到一个帧起始（SOF）位。 如果 CAN_IE.WKIE=1 时，则当此位置 1 后将产生中断。此位通过 CAN_IFC.WKIFC=1 清零。
ERRIF	Bit 2	R	**错误中断标志** 当错误中断使能 CAN_IE.ERRIE=1 时，并且 CAN_ERRSTAT 中某一位被置 1，该位被置 1，此位通过 CAN_IFC.ERRIFC=1 清零。
SLPSTAT	Bit 1	R	**睡眠状态** 此位由硬件置 1，用于向软件指示 CAN 此时处于睡眠模式。此位可确认软件的睡眠模式请求（CAN_CON.SLPREQ=1）。 当 CAN 退出睡眠模式(CAN_CON.SLPREQ=0)时此位由硬件清零。
INISTAT	Bit 0	R	**初始化状态** 此位由硬件置 1，用于向软件指示 CAN 硬件此时处于初始化模式。此位可确认软件的初始化请求（CAN_CON.INIREQ=1）。

			当 CAN 退出初始化模式(CAN_CON.INIREQ=0)时，此位由硬件清零。

28. 5. 2. 3 CAN中断标志清零寄存器（CAN_IFC）

CAN 主状态清零寄存器（CAN_IFC）

偏移地址：08_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-5	—	**保留**
SLPIFC	Bit 4	W1	**睡眠中断标志清零** 0：无操作 1：睡眠确认中断标志清零
WKIFC	Bit 3	W1	**唤醒中断标志清零** 0：无操作 1：唤醒中断标志清零
ERRIFC	Bit 2	W1	**错误中断标志清零** 0：无操作 1：错误中断标志清零
Reserved	Bit 1-0	—	**保留**

28. 5. 2. 4 CAN发送状态寄存器（CAN_TXSTAT）

CAN 发送状态寄存器 （CAN_TXSTAT）

偏移地址：$0C_H$

复位值：$00011100\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXM2LPF	TXM1LPF	TXM0LPF	TXM2EF	TXM1EF	TXM0EF	CODE		M2STPREQ	Reserved			M2TXERR	M2ARBLST	M2TXC	M2REQC	M1STPREQ	Reserved			M1TXERR	M1ARBLST	M1TXC	M1REQC	M0STPREQ	Reserved			M0TXERR	M0ARBLST	M0TXC	M0REQC

TXM2LPF	Bit 31	R	**发送邮箱 2 最低优先级标志** 当多个邮箱处于挂起态且邮箱 2 优先级最低时，此位由硬件置 1。
TXM1LPF	Bit 30	R	**发送邮箱 1 最低优先级标志** 当多个邮箱处于挂起态且邮箱 1 优先级最低时，此位由硬件置 1。
TXM0LPF	Bit 29	R	**发送邮箱 0 最低优先级标志** 当多个邮箱处于挂起态且邮箱 0 优先级最低时，此位由硬件置 1。
TXM2EF	Bit 28	R	**发送邮箱 2 空标志** 当邮箱 2 没有挂起的发送请求时，此位由硬件置 1。
TXM1EF	Bit 27	R	**发送邮箱 1 空标志** 当邮箱 1 没有挂起的发送请求时，此位由硬件置 1。
TXM0EF	Bit 26	R	**发送邮箱 0 空标志** 当邮箱 0 没有挂起的发送请求时，此位由硬件置 1。
CODE	Bit 25-24	R	**邮箱代码** 如果至少一个发送邮箱空闲，代码值等于下一个空闲发送邮箱的编号。 如果所有发送邮箱均挂起，则代码值等于优先级最低的发送邮箱的编号。
M2STPREQ	Bit 23	R/W	**邮箱 2 停止请求** 由软件置 1，用于中止相应邮箱的发送请求。 邮箱变为空后，此位由硬件清零。 未处于挂起态的邮箱，将此位置 1 没有任何作用。
Reserved	Bit 22-20	—	**保留**
M2TXERR	Bit 19	R	**邮箱 2 发送错误** 如果上一次发送因错误而失败，此位将置 1。 该位通过 CAN_TXSTATC.M2TXERR=1 清零。
M2ARBLST	Bit 18	R	**邮箱 2 仲裁丢失** 如果上一次发送因仲裁丢失而失败，此位将置 1。 该位通过 CAN_TXSTATC.M2ARBLST=1 清零。
M2TXC	Bit 17	R	**邮箱 2 发送完成** 每次发送尝试后，硬件都将更新此位。

			0：上一次发送失败 1：上一次发送成功 该位通过 CAN_TXSTATC.M2TXC=1 清零。
M2REQC	Bit 16	R	**邮箱 2 请求完成** 最后一个请求（发送或中止）执行完毕时，由硬件置 1。 该位通过 CAN_TXSTATC.M2REQC=1 清零。当产生发送请求（CAN_TXID2.TXMREQ=1）时由硬件清零。 如果将此位清零，邮箱 2 的所有状态位（M2TXC、M2ARBLST、M2TXERR）都将清零。
M1STPREQ	Bit 15	R/W	**邮箱 1 停止请求** 由软件置 1，用于中止相应邮箱的发送请求。 邮箱变为空后，此位由硬件清零。 未处于挂起态的邮箱，将此位置 1 没有任何作用。
Reserved	Bit 14-12	—	**保留**
M1TXERR	Bit 11	R	**邮箱 1 发送错误** 如果上一次发送因错误而失败，此位将置 1。 该位通过 CAN_TXSTATC.M1TXERR=1 清零。
M1ARBLST	Bit 10	R	**邮箱 1 仲裁丢失** 如果上一次发送因仲裁丢失而失败，此位将置 1。 该位通过 CAN_TXSTATC.M1ARBLST=1 清零。
M1TXC	Bit 9	R	**邮箱 1 发送完成** 每次发送尝试后，硬件都将更新此位。 0：上一次发送失败 1：上一次发送成功 该位通过 CAN_TXSTATC.M1TXC=1 清零。
M1REQC	Bit 8	R	**邮箱 1 请求完成** 最后一个请求（发送或中止）执行完毕时，由硬件置 1。 该位通过 CAN_TXSTATC.M1REQC=1 清零。当产生发送请求（CAN_TXID1.TXMREQ=1）时由硬件清零。 如果将此位清零，邮箱 1 的所有状态位（M1TXC、M1ARBLST、M1TXERR）都将清零。
M0STPREQ	Bit7	R/W	**邮箱 0 停止请求** 由软件置 1，用于中止相应邮箱的发送请求。 邮箱变为空后，此位由硬件清零。 未处于挂起态的邮箱，将此位置 1 没有任何作用。
Reserved	Bit 6-4	—	**保留**
M0TXERR	Bit 3	R	**邮箱 0 发送错误** 如果上一次发送因错误而失败，此位将置 1。 该位通过 CAN_TXSTATC.M0TXERR=1 清零。
M0ARBLST	Bit 2	R	**邮箱 0 仲裁丢失** 如果上一次发送因仲裁丢失而失败，此位将置 1。 该位通过 CAN_TXSTATC.M0ARBLST=1 清零。
M0TXC	Bit 1	R	**邮箱 0 发送完成**

			每次发送尝试后，硬件都将更新此位。 0：上一次发送失败 1：上一次发送成功 该位通过 CAN_TXSTATC.M0TXC=1 清零。
M0REQC	Bit 0	R	**邮箱 0 请求完成** 最后一个请求（发送或中止）执行完毕时，由硬件置 1。 该位通过 CAN_TXSTATC.M0REQC=1 清零。当产生发送请求（CAN_TXID0.TXMREQ=1）时由硬件清零。 如果将此位清零，邮箱 0 的所有状态位（M0TXC、M0ARBLST、M0TXERR）都将清零。

28. 5. 2. 5 CAN发送状态清零寄存器（CAN_TXSTATC）

CAN 发送状态清零寄存器（CAN_TXSTATC）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												M2TXERR	M2ARBLST	M2TXC	M2REQC	Reserved				M1TXERR	M1ARBLST	M1TXC	M1REQC	Reserved				M0TXERR	M0ARBLST	M0TXC	M0REQC

Reserved	Bit 31-20	—	保留
M2TXERR	Bit 19	W1	**邮箱 2 发送错误标志清零** 0：无操作 1：邮箱 2 发送错误标志清零
M2ARBLST	Bit 18	W1	**邮箱 2 仲裁丢失标志清零** 0：无操作 1：邮箱 2 仲裁丢失标志清零
M2TXC	Bit 17	W1	**邮箱 2 发送完成标志清零** 0：无操作 1：邮箱 2 发送成功标志清零
M2REQC	Bit 16	W1	**邮箱 2 请求完成标志清零** 0：无操作 1：邮箱 2 请求完成标志清零
Reserved	Bit 15-12	—	保留
M1TXERR	Bit 11	W1	**邮箱 1 发送错误标志清零** 0：无操作 1：邮箱 1 发送错误标志清零
M1ARBLST	Bit 10	W1	**邮箱 1 仲裁丢失标志清零** 0：无操作 1：邮箱 1 仲裁丢失标志清零
M1TXC	Bit 9	W1	**邮箱 1 发送完成标志清零** 0：无操作 1：邮箱 1 发送成功标志清零
M1REQC	Bit 8	W1	**邮箱 1 请求完成标志清零** 0：无操作 1：邮箱 1 请求完成标志清零
Reserved	Bit 7-4	—	保留
M0TXERR	Bit 3	W1	**邮箱 0 发送错误标志清零** 0：无操作 1：邮箱 0 发送错误标志清零
M0ARBLST	Bit 2	W1	**邮箱 0 仲裁丢失标志清零** 0：无操作 1：邮箱 0 仲裁丢失标志清零

M0TXC	Bit 1	W1	**邮箱 0 发送完成标志清零** 0：无操作 1：邮箱 0 发送成功标志清零
M0REQC	Bit 0	W1	**邮箱 0 请求完成标志清零** 0：无操作 1：邮箱 0 请求完成标志清零

28. 5. 2. 6 CAN接收FIFO0 寄存器（CAN_RXF0）

CAN 接收 FIFO0 寄存器（CAN_RXF0）

偏移地址：14_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-6	—	**保留**
FREE	Bit 5	R/W	**释放 FIFO0 接收邮箱** 由软件置 1，用于释放 FIFO0 的接收邮箱。FIFO0 至少有一条消息挂起时，才能释放输出邮箱。FIFO 为空时，将此位置 1 没有任何作用。如果 FIFO 中至少有两个邮箱，软件必须释放当前接收邮箱，才能访问下一个接收邮箱。接收邮箱全部释放后，此位由硬件清零。
OVR	Bit 4	R	**FIFO0 上溢** FIFO 三级深度邮箱填满时，如果接收到新消息并且通过筛选器，此位将由硬件置 1。 此位通过 CAN_RXF0C.OVRC=1 清零。
FULL	Bit 3	R	**FIFO0 满** FIFO 三级深度邮箱填满时，由硬件置 1。 此位通过 CAN_RXF0C.FULLC=1 清零。
Reserved	Bit 2	—	**保留**
PEND	Bit 1-0	R	**FIFO0 挂起接收邮箱数** 这些位用于指示接收 FIFO 中挂起接收邮箱数。硬件每向 FIFO 存储一条新消息，CAN_RXF0.PEND 即会增加。软件每次通过将 CAN_RXF0.FREE=1 来释放接收邮箱，CAN_RXF0.PEND 即会减小。

28. 5. 2. 7 CAN接收FIFO0 状态清零寄存器（CAN_RXF0C）

CAN 接收 FIFO0 状态清零寄存器（CAN_RXF0C）

偏移地址：18_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-5	—	**保留**
OVRC	Bit 4	W1	**FIFO0 上溢标志清零** 0：无操作 1：FIFO0 上溢标志清零
FULLC	Bit 3	W1	**FIFO0 满标志清零** 0：无操作 1：FIFO0 满标志清零
Reserved	Bit 2-0	—	**保留**

28. 5. 2. 8 CAN接收FIFO1 寄存器（CAN_RXF1）

CAN 接收 FIFO1 寄存器（CAN_RXF1）

偏移地址：$1C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-6	—	**保留**
FREE	Bit 5	R/W	**释放 FIFO1 接收邮箱** 由软件置 1，用于释放 FIFO1 的接收邮箱。FIFO1 至少有一条消息挂起时，才能释放输出邮箱。FIFO 为空时，将此位置 1 没有任何作用。如果 FIFO 中至少有两个邮箱，软件必须释放当前接收邮箱，才能访问下一个接收邮箱。接收邮箱全部释放后，此位由硬件清零。
OVR	Bit 4	R	**FIFO1 上溢** FIFO 三级深度邮箱填满时，如果接收到新消息并且通过筛选器，此位将由硬件置 1。 此位通过 CAN_RXF1C.OVRC=1 清零。
FULL	Bit 3	R	**FIFO1 满** FIFO 三级深度邮箱填满时，由硬件置 1。 此位通过 CAN_RXF1C.FULLC=1 清零。
Reserved	Bit 2	—	**保留**
PEND	Bit 1-0	R	**FIFO1 挂起接收邮箱数** 这些位用于指示接收 FIFO 中挂起接收邮箱数。硬件每向 FIFO 存储一条新消息，CAN_RXF1.PEND 即会增加。软件每次通过将 CAN_RXF1.FREE=1 来释放接收邮箱，CAN_RXF1.PEND 即会减小。

28. 5. 2. 9 CAN接收FIFO1 状态清零寄存器（CAN_RXF1C）

CAN 接收 FIFO1 状态清零寄存器（CAN_RXF1C）

偏移地址：20_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-5	—	**保留**
OVRC	Bit 4	W1	**FIFO1 上溢标志清零** 0：无操作 1：FIFO1 上溢标志清零
FULLC	Bit 3	W1	**FIFO1 满标志清零** 0：无操作 1：FIFO1 满标志清零
Reserved	Bit 2-0	—	**保留**

28. 5. 2. 10 CAN中断使能寄存器（CAN_IE）

CAN 中断使能寄存器（CAN_IE）																															
偏移地址：24_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														SLPIE	WKIE	ERRIE	Reserved			PRERRIE	BOFFIE	PERRIE	WARNIE	Reserved	F1OVRIE	F1FULIE	F1PIE	F0OVRIE	F0FULIE	F0PIE	TXMEIE

Reserved	Bit 31-18	—	保留
SLPIE	Bit 17	R/W	**睡眠中断使能** 0：当 CAN_STAT.SLPIF=1 时，不产生中断。 1：当 CAN_STAT.SLPIF=1 时，产生中断。
WKIE	Bit 16	R/W	**唤醒中断使能** 0：当 CAN_STAT.WKIF=1 时，不产生中断。 1：当 CAN_STAT.WKIF=1 时，产生中断。
ERRIE	Bit 15	R/W	**错误中断使能** 0：当 CAN_STAT.ERRIF=1 时，不会产生中断。 1：当 CAN_STAT.ERRIF=1 时，会产生中断。
Reserved	Bit 14-12	—	保留
PRERRIE	Bit 11	R/W	**上一个错误代码中断使能** 0：当 CAN_ERRSTAT.PRERRF 非 0 时， CAN_STAT.ERRIF 不会置 1。 1：当 CAN_ERRSTAT.PRERRF 非 0 时， CAN_STAT.ERRIF 会置 1。
BOFFIE	Bit 10	R/W	**总线关闭中断使能** 0：当 CAN_ERRSTAT.BOFF=1 时， CAN_STAT.ERRIF 不会置 1。 1：当 CAN_ERRSTAT.BOFF=1 时， CAN_STAT.ERRIF 会置 1。
PERRIE	Bit 9	R/W	**错误被动中断使能** 0：当 CAN_ERRSTAT.PERRF=1 时， CAN_STAT.ERRIF 不会置 1。 1：当 CAN_ERRSTAT.PERRF=1 时， CAN_STAT.ERRIF 会置 1。
WARNIE	Bit 8	R/W	**警告中断使能** 0：当 CAN_ERRSTAT.WARNF=1 时， CAN_STAT.ERRIF 不会置 1。 1：当 CAN_ERRSTAT.WARNF=1 时， CAN_STAT.ERRIF 会置 1。
Reserved	Bit 7	—	保留
F1OVRIE	Bit 6	R/W	**FIFO1 上溢中断使能**

			0：当 CAN_RXF1.OVR=1 时，不产生中断。 1：当 CAN_RXF1.OVR=1 时，产生中断。
F1FULIE	Bit 5	R/W	**FIFO1 满中断使能** 0：当 CAN_RXF1.FULL=1 时，不产生中断。 1：当 CAN_RXF1.FULL=1 时，产生中断。
F1PIE	Bit 4	R/W	**FIFO1 消息挂起中断使能** 0：当 CAN_RXF1.PEND 非 0 时，不产生中断。 1：当 CAN_RXF1.PEND 非 0 时，产生中断。
F0OVRIE	Bit 3	R/W	**FIFO0 上溢中断使能** 0：当 CAN_RXF0.OVR=1 时，不产生中断。 1：当 CAN_RXF0.OVR=1 时，产生中断。
F0FULIE	Bit 2	R/W	**FIFO0 满中断使能** 0：当 CAN_RXF0.FULL=1 时，不产生中断。 1：当 CAN_RXF0.FULL=1 时，产生中断。
F0PIE	Bit 1	R/W	**FIFO0 消息挂起中断使能** 0：当 CAN_RXF0.PEND 非 0 时，不产生中断。 1：当 CAN_RXF0.PEND 非 0 时，产生中断。
TXMEIE	Bit 0	R/W	**发送邮箱空中断使能** 0：CAN_TXSTAT.MxREQC=1 时，不产生中断。 1：CAN_TXSTAT.MxREQC=1 时，产生中断。

28. 5. 2. 11 CAN错误状态寄存器（CAN_ERRSTAT）

CAN 错误状态寄存器（CAN_ERRSTAT）																															
偏移地址：28_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXERRC								TXERRC								Reserved									PRERRF			Reserved	BOFF	PERRF	WARNF

RXERRC	Bit 31-24	R	**8 位接收错误计数器** CAN 协议故障隔离机制的实施部分。如果接收期间发生错误，该计数器按 1 或 8 递增，具体取决于 CAN 标准所定义的错误状况。每次成功接收后，该计数器按 1 递减，如果其数值大于 128，则复位为 120。计数器值超过 127 时，CAN 控制器进入错误被动状态。
TXERRC	Bit 23-16	R	**9 位发送错误计数器的低八位** 同上类似。
Reserved	Bit 15-7	—	**保留**
PRERRF	Bit 6-4	R/W	**上一个错误代码** 该字段由硬件置位，其中的代码指示 CAN 总线上检测到的上一个错误的错误状况。 000：无错误 001：填充错误 010：格式错误 011：确认错误 100：位隐性错误 101：位显性错误 110：CRC 错误 111：由软件置位
Reserved	Bit 3	—	**保留**
BOFF	Bit 2	R	**总线关闭标志** 此位由硬件置 1。当 CAN_ERRSTAT.TXERRC 上溢（超过 255）时，进入总线关闭状态。
PERRF	Bit 1	R	**错误被动标志** 达到错误被动状态（接收错误计数器或发送错误计数器>127）时，此位由硬件置 1。
WARNF	Bit 0	R	**警告错误标志** 达到错误警告极限时，此位由硬件置 1（接收错误计数器或发送错误计数器≥96）。

28. 5. 2. 12 CAN位时序寄存器（CAN_BTIME）

CAN 位时序寄存器 （CAN_BTIME）																															
偏移地址：$2C_H$																															
复位值：$00000001\_00100011\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SILENT	LOOP	Reserved				RESJW		Reserved	SEG2			SEG1				Reserved						BPSC									

SILENT	Bit 31	R/W	**静默模式（调试）** 0：正常工作 1：静默模式
LOOP	Bit 30	R/W	**回环模式（调试）** 0：禁止回环模式 1：使能回环模式
Reserved	Bit 29-26	—	**保留**
RESJW	Bit 25-24	R/W	**再同步跳转宽度** 这些位定义 CAN 硬件在执行再同步时最多可以将位加长或缩短的时间片数目。 $t_{RESJW}=t_{CAN}x$（RESJW+1）
Reserved	Bit 23	—	**保留**
SEG2	Bit 22-20	R/W	**时间段 2** 这些位定义时间段 2 中的时间片数目。 $T_{SEG2}=t_{CAN}x$（SEG2+1）
SEG1	Bit 19-16	R/W	**时间段 1** 这些位定义时间段 1 中的时间片数目。 $T_{SEG1}=t_{CAN}x$（SEG1+1）
Reserved	Bit 15-10	—	**保留**
BPSC	Bit 9-0	R/W	**波特率预分频器** 这些位定义一个时间片的长度。 T_{CAN}=（BPSC[9:0]+1）xt_{PCLK}

28. 5. 2. 13 CAN发送邮箱标识符寄存器 0（CAN_TXID0）

CAN 发送邮箱标识符寄存器 0（CAN_TXID0）

偏移地址：180_H

复位值：xxxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxx0$_B$

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

STDID[10: 0]/ EXID[28: 18]	Bit 31-21	R/W	**标准标识符或扩展标识符** 标准标识符或扩展标识符的 MSB（取决于 IDE 位的值）。
EXID[17: 0]	Bit 20-3	R/W	**扩展标识符** 扩展标识符的 LSB。
IDE	Bit 2	R/W	**标识符扩展** 此位用于定义邮箱中消息的标识符类型。 0：标准标识符。 1：扩展标识符。
RTR	Bit 1	R/W	**帧类型** 0：数据帧 1：远程帧
TXMREQ	Bit 0	R/W	**发送邮箱请求** 由软件置 1，用于请求发送相应邮箱的内容。邮箱变为空后，此位由硬件清零。

28. 5. 2. 14 CAN发送邮箱帧控制寄存器 0（CAN_TXFCON0）

CAN 发送邮箱帧控制寄存器 0（CAN_TXFCON0）

偏移地址：184_H

复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$

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

STAMP	Bit 31-16	R/W	**消息时间戳** 此字段包含在进行帧起始（SOF）发送时所捕获的 16 位定时器值。
Reserved	Bit 15-9	R/W	**保留**
TXGT	Bit 8	R/W	**发送全局时间** 只有硬件处于时间触发通信模式（CAN_CON.TTCEN=1）时，此位才会激活。 0：不发送时间戳 STAMP。 1：在 8 字节消息的最后两个数据字节中发送时间戳 STAMP 的值。数据字节 6 写入 STAMP[7:0]，数据字节 7 写入 STAMP[15:8]。且 DLEN 必须编程为 8，才能通过 CAN 总线发送这两个字节。
Reserved	Bit 7-4	—	**保留**
DLEN	Bit 3-0	R/W	**数据长度** 该字段定义数据帧或远程帧请求中的数据字节数。 一条消息可以包含 0 到 8 个数据字节。

28. 5. 2. 15 CAN发送邮箱数据低位寄存器 0（CAN_TXDL0）

CAN 邮箱数据低位寄存器 0 （CAN_TXDL0）																															
偏移地址：188_H																															
复位值：$xxxxxxxx\_xxxxxxxx\_xxxxxxxx\_xxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE3								BYTE2								BYTE1								BYTE0							

BYTE3	Bit 31-24	R/W	**数据字节 3** 消息的数据字节 3。
BYTE2	Bit 23-16	R/W	**数据字节 2** 消息的数据字节 2。
BYTE1	Bit 15-8	R/W	**数据字节 1** 消息的数据字节 1。
BYTE0	Bit 7-0	R/W	**数据字节 0** 消息的数据字节 0。

注：当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

28. 5. 2. 16 CAN发送邮箱数据高位寄存器 0（CAN_TXDH0）

CAN 邮箱数据高位寄存器 0（CAN_TXDH0）																															
偏移地址：$18C_H$																															
复位值：$xxxxxxxx\_xxxxxxxx\_xxxxxxxx\_xxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE7								BYTE6								BYTE5								BYTE4							

BYTE7	Bit 31-24	R/W	**数据字节 7** 消息的数据字节 7。
BYTE6	Bit 23-16	R/W	**数据字节 6** 消息的数据字节 6。
BYTE5	Bit 15-8	R/W	**数据字节 5** 消息的数据字节 5。
BYTE4	Bit 7-0	R/W	**数据字节 4** 消息的数据字节 4。

注：当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

28. 5. 2. 17 CAN发送邮箱标识符寄存器 1（CAN_TXID1）

CAN 发送邮箱标识符寄存器 1（CAN_TXID1）

偏移地址：190_H

复位值：xxxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxx0$_B$

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

STDID[10: 0]/ EXID[28: 18]	Bit 31-21	R/W	**标准标识符或扩展标识符** 标准标识符或扩展标识符的 MSB（取决于 IDE 位的值）。
EXID[17: 0]	Bit 20-3	R/W	**扩展标识符** 扩展标识符的 LSB。
IDE	Bit 2	R/W	**标识符扩展** 此位用于定义邮箱中消息的标识符类型。 0：标准标识符。 1：扩展标识符。
RTR	Bit 1	R/W	**帧类型** 0：数据帧 1：远程帧
TXMREQ	Bit 0	R/W	**发送邮箱请求** 由软件置 1，用于请求发送相应邮箱的内容。邮箱变为空后，此位由硬件清零。

28. 5. 2. 18 CAN发送邮箱帧控制寄存器 1（CAN_TXFCON1）

CAN 发送邮箱帧控制寄存器 1 （CAN_TXFCON1）																															
偏移地址：194_H																															
复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STAMP																Reserved							TXGT	Reserved				DLEN			

STAMP	Bit 31-16	R/W	**消息时间戳** 此字段包含在进行帧起始（SOF）发送时所捕获的 16 位定时器值。
Reserved	Bit 15-9	R/W	**保留**
TXGT	Bit 8	R/W	**发送全局时间** 只有硬件处于时间触发通信模式（CAN_CON.TTCEN=1）时，此位才会激活。 0：不发送时间戳 STAMP。 1：在 8 字节消息的最后两个数据字节中发送时间戳 STAMP 的值。数据字节 6 写入 STAMP[7:0]，数据字节 7 写入 STAMP[15:8]。且 DLEN 必须编程为 8，才能通过 CAN 总线发送这两个字节。
Reserved	Bit 7-4	—	**保留**
DLEN	Bit 3-0	R/W	**数据长度代码** 该字段定义数据帧或远程帧请求中的数据字节数。 一条消息可以包含 0 到 8 个数据字节。

28. 5. 2. 19 CAN发送邮箱数据低位寄存器 1（CAN_TXDL1）

CAN 邮箱数据低位寄存器 1 （CAN_TXDL1）																															
偏移地址：198_H																															
复位值：$xxxxxxxx\_xxxxxxxx\_xxxxxxxx\_xxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE3								BYTE2								BYTE1								BYTE0							

BYTE3	Bit 31-24	R/W	**数据字节 3** 消息的数据字节 3。
BYTE2	Bit 23-16	R/W	**数据字节 2** 消息的数据字节 2。
BYTE1	Bit 15-8	R/W	**数据字节 1** 消息的数据字节 1。
BYTE0	Bit 7-0	R/W	**数据字节 0** 消息的数据字节 0。

注：当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

28. 5. 2. 20 CAN发送邮箱数据高位寄存器 1（CAN_TXDH1）

CAN 邮箱数据高位寄存器 1（CAN_TXDH1）																															
偏移地址：$19C_H$																															
复位值：$xxxxxxxx\_xxxxxxxx\_xxxxxxxx\_xxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE7								BYTE6								BYTE5								BYTE4							

BYTE7	Bit 31-24	R/W	**数据字节 7** 消息的数据字节 7。
BYTE6	Bit 23-16	R/W	**数据字节 6** 消息的数据字节 6。
BYTE5	Bit 15-8	R/W	**数据字节 5** 消息的数据字节 5。
BYTE4	Bit 7-0	R/W	**数据字节 4** 消息的数据字节 4。

注：当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

28. 5. 2. 21 CAN发送邮箱标识符寄存器 2（CAN_TXID2）

CAN 发送邮箱标识符寄存器 2（CAN_TXID2）

偏移地址：$1A0_H$

复位值：xxxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxx0$_B$

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

STDID[10: 0] /EXID[28: 18]	Bit 31-21	R/W	**标准标识符或扩展标识符** 标准标识符或扩展标识符的 MSB（取决于 IDE 位的值）。
EXID[17: 0]	Bit 20-3	R/W	**扩展标识符** 扩展标识符的 LSB。
IDE	Bit 2	R/W	**标识符扩展** 此位用于定义邮箱中消息的标识符类型。 0：标准标识符。 1：扩展标识符。
RTR	Bit 1	R/W	**帧类型** 0：数据帧 1：远程帧
TXMREQ	Bit 0	R/W	**发送邮箱请求** 由软件置 1，用于请求发送相应邮箱的内容。邮箱变为空后，此位由硬件清零。

28. 5. 2. 22 CAN发送邮箱帧控制寄存器 2（CAN_TXFCON2）

CAN 邮箱数据长度控制和时间戳寄存器 2 （CAN_TXFCON2）																															
偏移地址：$1A4_H$																															
复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STAMP																Reserved							TXGT	Reserved				DLEN			

STAMP	Bit 31-16	R/W	**消息时间戳** 此字段包含在进行帧起始（SOF）发送时所捕获的 16 位定时器值。
Reserved	Bit 15-9	R/W	**保留**
TXGT	Bit 8	R/W	**发送全局时间** 只有硬件处于时间触发通信模式（CAN_CON.TTCEN=1）时，此位才会激活。 0：不发送时间戳 STAMP。 1：在 8 字节消息的最后两个数据字节中发送时间戳 STAMP 的值。数据字节 6 写入 STAMP[7:0]，数据字节 7 写入 STAMP[15:8]。且 DLEN 必须编程为 8，才能通过 CAN 总线发送这两个字节。
Reserved	Bit 7-4	—	**保留**
DLEN	Bit 3-0	R/W	**数据长度代码** 该字段定义数据帧或远程帧请求中的数据字节数。 一条消息可以包含 0 到 8 个数据字节。

28. 5. 2. 23 CAN发送邮箱数据低位寄存器 2（CAN_TXDL2）

CAN 邮箱数据低位寄存器 2（CAN_TXDL2）																															
偏移地址：$1A8_H$																															
复位值：$xxxxxxxx\_xxxxxxxx\_xxxxxxxx\_xxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE3								BYTE2								BYTE1								BYTE0							

BYTE3	Bit 31-24	R/W	**数据字节 3** 消息的数据字节 3。
BYTE2	Bit 23-16	R/W	**数据字节 2** 消息的数据字节 2。
BYTE1	Bit 15-8	R/W	**数据字节 1** 消息的数据字节 1。
BYTE0	Bit 7-0	R/W	**数据字节 0** 消息的数据字节 0。

注：当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

28. 5. 2. 24 CAN发送邮箱数据高位寄存器 2（CAN_TXDH2）

CAN 邮箱数据高位寄存器 2（CAN_TXDH2）																															
偏移地址：$1AC_H$																															
复位值：$xxxxxxxx\_xxxxxxxx\_xxxxxxxx\_xxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE7								BYTE6								BYTE5								BYTE4							

BYTE7	Bit 31-24	R/W	**数据字节 7** 消息的数据字节 7。
BYTE6	Bit 23-16	R/W	**数据字节 6** 消息的数据字节 6。
BYTE5	Bit 15-8	R/W	**数据字节 5** 消息的数据字节 5。
BYTE4	Bit 7-0	R/W	**数据字节 4** 消息的数据字节 4。

注：当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

28. 5. 2. 25 CAN接收FIFO0 邮箱标识符寄存器（CAN_RXF0ID）

CAN 接收 FIFO0 邮箱标识符寄存器（CAN_RXF0ID）

偏移地址：$1B0_H$

复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$

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

STDID[10:0] /EXID[28:18]	Bit 31-21	R	**标准标识符或扩展标识符** 标准标识符或扩展标识符的 MSB（取决于 IDE 位的值）。
EXID[17:0]	Bit 20-3	R	**扩展标识符** 扩展标识符的 LSB。
IDE	Bit 2	R	**标识符扩展** 此位用于定义邮箱中消息的标识符类型。 0：标准标识符。 1：扩展标识符。
RTR	Bit 1	R	**帧类型** 0：数据帧 1：远程帧
Reserved	Bit 0	—	**保留**

注：所有接收寄存器均受到写保护。

28. 5. 2. 26 CAN接收FIFO0 邮箱数据信息寄存器（CAN_RXF0INF）

CAN 接收 FIFO0 邮箱数据信息寄存器（CAN_RXF0INF）

偏移地址：$1B4_H$

复位值：$xxxxxxxx\_xxxxxxxx\_xxxxxxxx\_xxxxxxxx_B$

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

STAMP	Bit 31-16	R	**消息时间戳** 此字段包含在接收帧起始（SOF）时所捕获的 16 位定时器值。
FLTIDX	Bit 15-8	R	**筛选器匹配索引** 该寄存器包含筛选器索引，邮箱中存储的消息需要经过筛选器。
Reserved	Bit 7-4	—	**保留**
DLEN	Bit 3-0	R	**数据长度代码** 该字段定义一个数据帧所包含的数据字节数（0 到 8）。如果是远程帧请求，则为 0。

28. 5. 2. 27 CAN接收FIFO0 邮箱数据低位寄存器（CAN_RXF0DL）

CAN 接收 FIFO0 邮箱数据低位寄存器（CAN_RXF0DL）																															
偏移地址：$1B8_H$																															
复位值：$xxxxxxxx\_xxxxxxxx\_xxxxxxxx\_xxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE3								BYTE2								BYTE1								BYTE0							

BYTE3	Bit 31-24	R	**数据字节 3** 消息的数据字节 3。
BYTE2	Bit 23-16	R	**数据字节 2** 消息的数据字节 2。
BYTE1	Bit 15-8	R	**数据字节 1** 消息的数据字节 1。
BYTE0	Bit 7-0	R	**数据字节 0** 消息的数据字节 0。

28. 5. 2. 28 CAN接收FIFO0 邮箱数据高位寄存器（CAN_RXF0DH）

CAN 接收 FIFO0 邮箱数据高位寄存器（CAN_RXF0DH）																															
偏移地址：$1BC_H$																															
复位值：$xxxxxxxx\_xxxxxxxx\_xxxxxxxx\_xxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE7								BYTE6								BYTE5								BYTE4							

BYTE7	Bit 31-24	R	**数据字节 7** 消息的数据字节 7。
BYTE6	Bit 23-16	R	**数据字节 6** 消息的数据字节 6。
BYTE5	Bit 15-8	R	**数据字节 5** 消息的数据字节 5。
BYTE4	Bit 7-0	R	**数据字节 4** 消息的数据字节 4。

28. 5. 2. 29 CAN接收FIFO1 邮箱标识符寄存器（CAN_RXF1ID）

CAN 接收 FIFO1 邮箱标识符寄存器（CAN_RXF1ID）

偏移地址：$1C0_H$

复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$

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

STDID[10:0] /EXID[28:18]	Bit 31-21	R	**标准标识符或扩展标识符** 标准标识符或扩展标识符的 MSB（取决于 IDE 位的值）。
EXID[17:0]	Bit 20-3	R	**扩展标识符** 扩展标识符的 LSB。
IDE	Bit 2	R	**标识符扩展** 此位用于定义邮箱中消息的标识符类型。 0：标准标识符。 1：扩展标识符。
RTR	Bit 1	R	**帧类型** 0：数据帧 1：远程帧
Reserved	Bit 0	—	**保留**

注：所有接收寄存器均受到写保护。

28. 5. 2. 30 CAN接收FIFO1 邮箱数据信息寄存器（CAN_RXF1INF）

CAN 接收 FIFO1 邮箱数据信息寄存器（CAN_RXF1INF）

偏移地址：$1C4_H$

复位值：$xxxxxxxx\_xxxxxxxx\_xxxxxxxx\_xxxxxxxx_B$

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

STAMP	Bit 31-16	R	**消息时间戳** 此字段包含在接收帧起始（SOF）时所捕获的 16 位定时器值。
FLTIDX	Bit 15-8	R	**筛选器匹配索引** 该寄存器包含筛选器索引，邮箱中存储的消息需要经过筛选器。
Reserved	Bit 7-4	—	**保留**
DLEN	Bit 3-0	R	**数据长度代码** 该字段定义一个数据帧所包含的数据字节数（0 到 8）。如果是远程帧请求，则为 0。

28. 5. 2. 31 CAN接收FIFO1 邮箱数据低位寄存器（CAN_RXF1DL）

CAN 接收 FIFO1 邮箱数据低位寄存器（CAN_RXF1DL）

偏移地址：$1C8_H$

复位值：$xxxxxxxx\_xxxxxxxx\_xxxxxxxx\_xxxxxxxx_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE3								BYTE2								BYTE1								BYTE0							

BYTE3	Bit 31-24	R	**数据字节 3** 消息的数据字节 3。
BYTE2	Bit 23-16	R	**数据字节 2** 消息的数据字节 2。
BYTE1	Bit 15-8	R	**数据字节 1** 消息的数据字节 1。
BYTE0	Bit 7-0	R	**数据字节 0** 消息的数据字节 0。

28. 5. 2. 32 CAN接收FIFO1 邮箱数据高位寄存器（CAN_RXF1DH）

CAN 接收 FIFO1 邮箱数据高位寄存器（CAN_RXF1DH）

偏移地址：$1CC_H$

复位值：$xxxxxxxx\_xxxxxxxx\_xxxxxxxx\_xxxxxxxx_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE7								BYTE6								BYTE5								BYTE4							

BYTE7	Bit 31-24	R	**数据字节 7** 消息的数据字节 7。
BYTE6	Bit 23-16	R	**数据字节 6** 消息的数据字节 6。
BYTE5	Bit 15-8	R	**数据字节 5** 消息的数据字节 5。
BYTE4	Bit 7-0	R	**数据字节 4** 消息的数据字节 4。

28. 5. 2. 33 CAN筛选器控制寄存器（CAN_FLTCON）

CAN 筛选器主寄存器（CAN_FLTCON）

偏移地址：200_H

复位值：$00101010\_00011100\_00001110\_00000001_B$

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

Reserved	Bit 31-1	—	保留
FLTINI	Bit 0	R/W	**筛选器初始化模式** 筛选器组的初始化模式。（在配置筛选器寄存器时，需要将该位置 1） 0：筛选器工作模式。 1：筛选器初始化模式。

注：此寄存器的所有位均由软件置 1 和清零。

28. 5. 2. 34 CAN筛选器模式寄存器（CAN_FLTM）

CAN 筛选器模式寄存器（CAN_FLTM）

偏移地址：204_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		MOD13	MOD12	MOD11	MOD10	MOD9	MOD8	MOD7	MOD6	MOD5	MOD4	MOD3	MOD2	MOD1	MOD0

Reserved	Bit 31-14	—	保留
MOD<x>	Bit 13-0	R/W	**筛选器模式** 筛选器 x 的模式 0：筛选器组 x 的两个 32 位寄存器处于掩码模式。 1：筛选器组 x 的两个 32 位寄存器处于标识符列表模式。

注：仅当 CAN_FLTCON 寄存器中设置了筛选器初始化模式（FLTINI=1）时，才能对此寄存器执行写操作。

28. 5. 2. 35 CAN筛选器宽度选择寄存器（CAN_FLTWS）

CAN 筛选器宽度寄存器（CAN_FLTWS）

偏移地址：$20C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		SEL13	SEL12	SEL11	SEL10	SEL9	SEL8	SEL7	SEL6	SEL5	SEL4	SEL3	SEL2	SEL1	SEL0

Reserved	Bit 31-14	R/W	保留
SEL<x>	Bit 13-0	R/W	**宽度选择** 这些位定义了筛选器 0-13 的宽度选择。 0：双 16 位宽度 1：单 32 位宽度

注：仅当 CAN_FLTCON 寄存器中设置了筛选器初始化模式（FLTINI=1）时，才能对此寄存器执行写操作。

28. 5. 2. 36 CAN筛选器分配寄存器（CAN_FLTAS）

CAN 筛选器 FIFO 分配寄存器（CAN_FLTAS）

偏移地址：214_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		ASSIGN13	ASSIGN12	ASSIGN11	ASSIGN10	ASSIGN9	ASSIGN8	ASSIGN7	ASSIGN6	ASSIGN5	ASSIGN4	ASSIGN3	ASSIGN2	ASSIGN1	ASSIGN0

Reserved	Bit 31-14	—	保留
ASSIGN<x>	Bit 13-0	R/W	**筛选器 x 的 FIFO 分配** 通过此筛选器的消息将存储在指定的 FIFO 中。 0：筛选器分配到 FIFO0 1：筛选器分配到 FIFO1

注：仅当 CAN_FLTCON 寄存器中设置了筛选器初始化模式（FLTINI=1）时，才能对此寄存器执行写操作。

28. 5. 2. 37 CAN筛选器启用寄存器（CAN_FLTGO）

CAN 筛选器激活寄存器（CAN_FLTGO）

偏移地址：$21C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		GO13	GO12	GO11	GO10	GO9	GO8	GO7	GO6	GO5	GO4	GO3	GO2	GO1	GO0

Reserved	Bit 31-14	—	**保留**
GO<x>	Bit 13-0	R/W	**筛选器激活** 软件将此位置 1 可启用筛选器 x。要修改筛选器相关寄存器，必须将筛选器关闭或将筛选器设为初始化模式。 0：筛选器 x 关闭 1：筛选器 x 启用

28. 5. 2. 38 筛选器组 0 寄存器 1（CAN_FLT0R1）

筛选器组 0 寄存器 1（CAN_FLT0R1）

偏移地址：240_H

复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 39 筛选器组 0 寄存器 2（CAN_FLT0R2）

筛选器组 0 寄存器 2（CAN_FLT0R2）																															
偏移地址：244_H																															
复位值：xxxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxxx$_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 40 筛选器组 1 寄存器 1（CAN_FLT1R1）

筛选器组 1 寄存器 1（CAN_FLT1R1）																															
偏移地址：248_H																															
复位值：xxxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxxx$_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 41 筛选器组 1 寄存器 2（CAN_FLT1R2）

筛选器组 1 寄存器 2（CAN_FLT1R2）																														
偏移地址：$24C_H$																														
复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 42 筛选器组 2 寄存器 1（CAN_FLT2R1）

筛选器组 2 寄存器 1（CAN_FLT2R1）																														
偏移地址：250_H																														
复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 43 筛选器组 2 寄存器 2（CAN_FLT2R2）

筛选器组 2 寄存器 2（CAN_FLT2R2）																															
偏移地址：254_H																															
复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 44 筛选器组 3 寄存器 1（CAN_FLT3R1）

筛选器组 3 寄存器 1（CAN_FLT3R1）																															
偏移地址：258_H																															
复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 45 筛选器组 3 寄存器 2（CAN_FLT3R2）

筛选器组 3 寄存器 2（CAN_FLT3R2）																														
偏移地址：$25C_H$																														
复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 46 筛选器组 4 寄存器 1（CAN_FLT4R1）

筛选器组 4 寄存器 1（CAN_FLT4R1）																														
偏移地址：260_H																														
复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 47 筛选器组 4 寄存器 2（CAN_FLT4R2）

筛选器组 4 寄存器 2（CAN_FLT4R2）																															
偏移地址：264_H																															
复位值：xxxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxxx$_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 48 筛选器组 5 寄存器 1（CAN_FLT5R1）

筛选器组 5 寄存器 1（CAN_FLT5R1）																															
偏移地址：268_H																															
复位值：xxxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxxx$_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 49　筛选器组 5 寄存器 2（CAN_FLT5R2）

筛选器组 5 寄存器 2（CAN_FLT5R2）																														
偏移地址：$26C_H$																														
复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 50　筛选器组 6 寄存器 1（CAN_FLT6R1）

筛选器组 6 寄存器 1（CAN_FLT6R1）																														
偏移地址：270_H																														
复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 51 筛选器组 6 寄存器 2（CAN_FLT6R2）

筛选器组 6 寄存器 2（CAN_FLT6R2）																														
偏移地址：274_H																														
复位值：xxxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxxx$_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 52 筛选器组 7 寄存器 1（CAN_FLT7R1）

筛选器组 7 寄存器 1（CAN_FLT7R1）																														
偏移地址：278_H																														
复位值：xxxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxxx$_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 53 筛选器组 7 寄存器 2（CAN_FLT7R2）

筛选器组 7 寄存器 2（CAN_FLT7R2）																														
偏移地址：$27C_H$																														
复位值：xxxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxxx$_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 54 筛选器组 8 寄存器 1（CAN_FLT8R1）

筛选器组 8 寄存器 1（CAN_FLT8R1）																														
偏移地址：280_H																														
复位值：xxxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxxx$_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 55 筛选器组 8 寄存器 2（CAN_FLT8R2）

筛选器组 8 寄存器 2（CAN_FLT8R2）																														
偏移地址：284_H																														
复位值：xxxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxxx$_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 56 筛选器组 9 寄存器 1（CAN_FLT9R1）

筛选器组 9 寄存器 1（CAN_FLT9R1）																														
偏移地址：288_H																														
复位值：xxxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxxx$_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 57 筛选器组 9 寄存器 2（CAN_FLT9R2）

筛选器组 9 寄存器 2（CAN_FLT9R2）																															
偏移地址：$28C_H$																															
复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 58 筛选器组 10 寄存器 1（CAN_FLT10R1）

筛选器组 10 寄存器 1（CAN_FLT10R1）																															
偏移地址：290_H																															
复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 59 筛选器组 10 寄存器 2（CAN_FLT10R2）

筛选器组 10 寄存器 2（CAN_FLT10R2）																															
偏移地址：294_H																															
复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 60 筛选器组 11 寄存器 1（CAN_FLT11R1）

筛选器组 11 寄存器 1（CAN_FLT11R1）																															
偏移地址：298_H																															
复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 61 筛选器组 11 寄存器 2（CAN_FLT11R2）

筛选器组 11 寄存器 2（CAN_FLT11R2）																														
偏移地址：$29C_H$																														
复位值：xxxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxxx$_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 62 筛选器组 12 寄存器 1（CAN_FLT12R1）

筛选器组 12 寄存器 1（CAN_FLT12R1）																														
偏移地址：$2A0_H$																														
复位值：xxxxxxxxx_xxxxxxxxx_xxxxxxxxx_xxxxxxxxx$_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 63 筛选器组 12 寄存器 2（CAN_FLT12R2）

筛选器组 12 寄存器 2（CAN_FLT12R2）																														
偏移地址：$2A4_H$																														
复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 64 筛选器组 13 寄存器 1（CAN_FLT13R1）

筛选器组 13 寄存器 1（CAN_FLT13R1）																														
偏移地址：$2A8_H$																														
复位值：$xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx\_xxxxxxxxx_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

28. 5. 2. 65 筛选器组 13 寄存器 2（CAN_FLT13R2）

筛选器组 13 寄存器 2（CAN_FLT13R2）																														
偏移地址：$2AC_H$																														
复位值：$xxxxxxxx\_\ xxxxxxxx\ \_\ xxxxxxxx\ \_\ xxxxxxxx_B$																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLTB31	FLTB30	FLTB29	FLTB28	FLTB27	FLTB26	FLTB25	FLTB24	FLTB23	FLTB22	FLTB21	FLTB20	FLTB19	FLTB18	FLTB17	FLTB16	FLTB15	FLTB14	FLTB13	FLTB12	FLTB11	FLTB10	FLTB9	FLTB8	FLTB7	FLTB6	FLTB5	FLTB4	FLTB3	FLTB2	FLTB1	FLTB0

FLTB<x>	Bit 31-0	R/W	**筛选器位** **标识符** 寄存器的每一位极性。 0：需要显性位 1：需要隐性位 **掩码** 寄存器的对应位是否需要匹配。 0：屏蔽，该位无需比较。 1：匹配，该位必须与对应的标识符寄存器位极性相同。

第29章 通用串行总线（USB2.0）

29. 1 概述

该 USB 控制器作为高速/全速 USB 设备的控制器，兼容 USB2.0 协议规范中高速数据传输（480Mbps）、全速数据传输（12Mbps）和嵌入式设备（On-The-GO）标准。支持点对点通信时工作于主机或设备两种模式，多点通信时工作于 USB 主机模式。支持在 OTG 模式下与一个或多个高速/全速/低速设备通信。

支持会话请求协议 SRP（Session Request Protocol）和主机协商协议 HNP（Host Negotiation Protocol）通信协议；支持 4 种数据传输类型：控制传输/同步传输/中断传输/批量传输；支持 DMA 对端点 FIFO 的访问。

29. 2 特性

- USB 设备控制器支持高速（480Mbps）/全速（12Mbps）的数据传输模式
- 支持点对点通信时工作于主机或设备两种模式，多点通信时工作于 USB 主机模式
- 兼容 USB2.0 协议规范中高速数据传输（480Mbps）、全速数据传输（12Mbps）和嵌入式设备（On-The-GO）标准
- 支持在 OTG 模式下与一个或多个高速/全速/低速设备通信
- 支持会话请求协议 SRP（Session Request Protocol）和主机协商协议 HNP（Host Negotiation Protocol）通信协议
- 支持 4 种数据传输类型：控制传输/同步传输/中断传输/批量传输
- 支持挂起状态和恢复功能
- USB 设备模式下支持软连接和断开功能
- 内置 4KB SRAM 端点 FIFO
- 支持 11 个端点，端点大小最大支持 1024 字节数据：
  - EP0IN/OUT：支持控制传输
  - EP1IN~EP5IN（Rx Endpoints）：支持同步传输、中断传输、批量传输
  - EP1OUT~EP5OUT（Tx Endpoints）：支持同步传输、中断传输、批量传输
- 支持使用 DMA 对端点 FIFO 进行访问

29. 3 结构框图

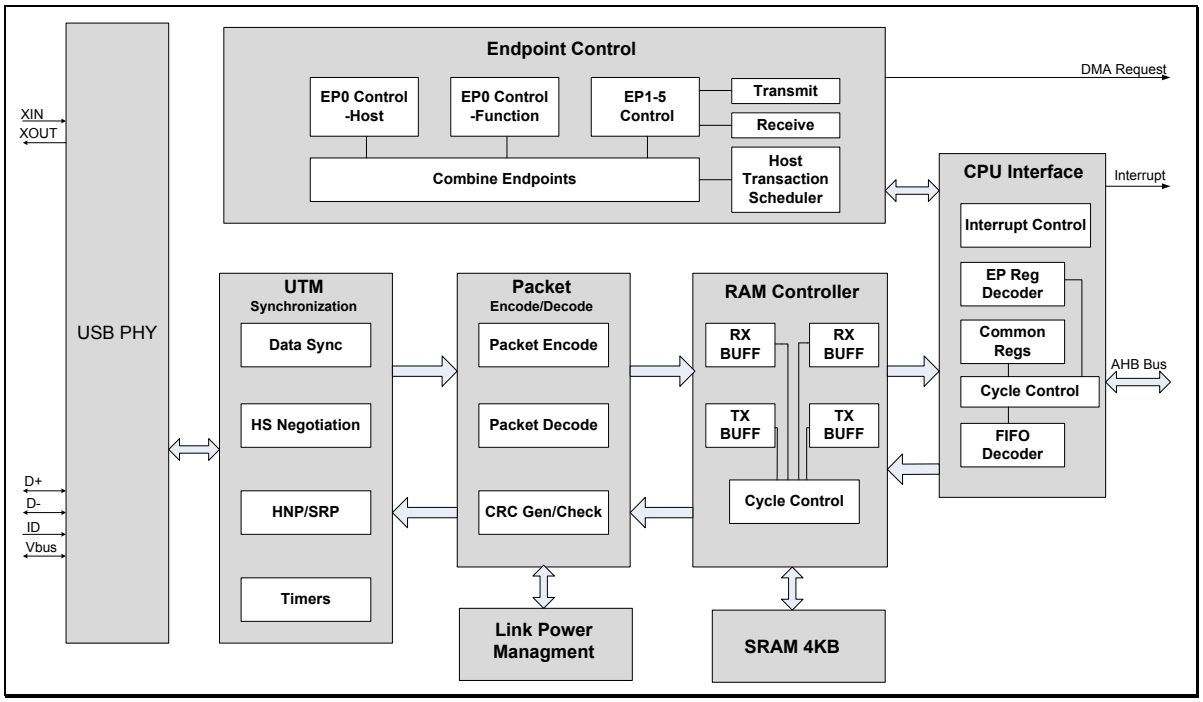


图 29-1 USB 电路结构框图

29.4 功能描述

29.4.1 功能概述

USB 控制器有两种主要的工作模式：USB 设备模式和 USB 主机模式。

当USB控制器工作于设备模式，USB控制器完成USB数据包的发送接收以及数据的编码、解码、校验等工作。对于 IN 事务的处理是通过发送端点（EP0OUT/EP1OUT~EP5OUT）完成，OUT 事务的处理是通过接收端点（EP0IN/EP1IN~EP5IN）完成。可支持控制传输、同步传输、中断传输、批量传输四种数据传输方式。

当 USB 控制器工作于主机模式，USB 控制器的工作行为取决于以下两种连接方式：与另一个 USB 设备进行点对点通信以及连接 USB 集线器。当与另一个 USB 设备进行点对点通信时，USB 控制器作为主机可以提供大部分 USB 主机功能。当连接 USB 集线器，作为主机可连接多个 USB 设备同时进行通信。

当 USB 控制器工作于主机模式，与另一个 USB 设备进行点对点通信时，USB 控制器可支持控制传输、同步传输、中断传输、批量传输。对于 IN 事务的处理是通过接收端点（EP0IN/EP1IN~EP5IN）完成，OUT 事务的处理是通过发送端点（EP0OUT/EP1OUT~EP5OUT）完成。USB 控制器完成 USB 数据包的发送接收以及数据的编码、解码、校验等工作，同时对于同步端点和中断端点 USB 控制器会自动的根据端点的配置在 USB 帧/小帧中安排事务传输。

当 USB 控制器工作于主机模式连接 USB 集线器时，需要进一步的完成如下细节配置：

- ◇ 目标通信设备的功能地址（配置寄存器 TX/RXFUNCADDR）；
- ◇ 目标通信设备的传输速率（配置寄存器 IND_TXTYPE / IND_RXTYPE）；
- ◇ 当通过高速集线器与全速或者低速 USB 设备进行通信时，需要新增集线器功能地址和集线器端口设置（配置寄存器 TX/RXxHUBADDR、TX/RXxHUBPORT）；

当 USB 控制器作为 A 类设备（主机模式）连接其它 USB 设备时，需要向 VBUS 端口提供 5V 电压。当作为 B 类设备（设备模式）时，应该能够在 VBUS 端口提供 2V 电压用于唤醒 A 类设备。USB 控制器初始连接是工作于主机模式还是工作于设备模式，是由 ID 管脚电平决定。ID 管脚为低电平时，表示工作于主机模式（A 类设备）；ID 管脚为高电平时，表示工作于设备模式（B 类设备）。当 USB 控制器作为 B 类设备工作于设备模式时，检测到 USB 总线处于空闲状态时，软件可以配置寄存器 USB_DEVCTL 的控制位“HOSTREQ”请求工作于主机模式。

29.4.2 设备模式

当主机模式位（USB_DEVCTL.HOSTMODE）清零，则 USB 控制器运行在设备模式下。

USB 控制器的工作模式发生改变时，即从设备模式切换到主机模式或从主机模式切换到设备模式时，软件必须复位 USB 控制器，通过将寄存器 SOFTRST 的 bit0 置 1 来实现。

在设备模式下，IN 事务由端点传输接口控制，并使用发送端点完成。OUT 事务使用端点接收接口控制，并使用接收端点完成。要注意端点 FIFO 大小要适合端点最大数据包的大小。

批量传输

批量传输端点的 FIFO 必须支持最大数据包的大小（最多 64 个字节）或最大值的两倍包大小（如果使用双缓冲）。

中断传输

中断传输端点的 FIFO 必须支持最大数据包的大小（最多 64 个字节）或最大值的两倍包大小（如果使用双缓冲）。

同步传输

同步传输端点的 FIFO 比较灵活，最大可支持 1023 字节。

控制传输

通常，USB 设备应使用 USB 控制器端点 0 作为专用控制端点。

29. 4. 2. 1 端点

USB 控制器支持 11 个端点：EP0IN/OUT、EP1IN~EP5IN、EP1OUT~EP5OUT。EP0 端点仅支持控制传输，其他端点同时支持同步传输、中断传输、批量传输。各个端点的 FIFO 大小是固定的，不可配置。端点特性如下表所示。

端点 0 是专用的控制端点，用于在枚举过程中所有的对端点 0 的控制传输事务，或者是对端点 0 的任意其它的控制请求。

配对的 IN 端点和 OUT 端点可以配置成不同的类型。比如，OUT 端点对配置成批量端点，而该端点对应的 IN 端点可以配置成中断端点。

端点号	端点大小（字节）	端点类型	双缓冲
EP0（IN/OUT）	64	控制	无
EP1 IN	64	同步、中断、块	有
EP1 OUT	64	同步、中断、块	有
EP2 IN	64	同步、中断、块	有
EP2 OUT	64	同步、中断、块	有
EP3 IN	64	同步、中断、块	有
EP3 OUT	64	同步、中断、块	有
EP4 IN	128	同步、中断、块	有
EP4 OUT	128	同步、中断、块	有
EP5 IN	128	同步、中断、块	有
EP5 OUT	128	同步、中断、块	有

表 29-1 端点特性

注 1：Tx Endpoints 对应端点 EP* OUT，Rx Endpoints 对应端点 EP* IN，都是相对于主机而言；
注 2：USB 控制器工作于设备模式时，IN 事务处理通过 Tx Endpoints，OUT 事务处理通过 Rx Endpoints；
注 3：USB 控制器工作于主机模式时，IN 事务处理通过 Rx Endpoints，OUT 事务处理通过 Tx Endpoints；

29. 4. 2. 2 设备模式下的IN传输

IN 事务传输的数据通过 OUT 端点的发送 FIFO 进行处理。

单缓冲

如果发送端点的 FIFO 大小小于该端点最大数据包长（由寄存器 TXMAXP 设定，或者在支持动态 FIFO 深度功能时由 TXFIFOSIZE 设定）的两倍，只有一个数据包可以被缓冲到 FIFO 中。当每个数据包加载到发送 FIFO 中去时，USB 端点发送控制/状态寄存器 USB_TXxCSRL 的 TXPKTRDY 位必须置 1。如果 USB 端点控制/状态寄存器 USB_TXxCSRH 的 AUTOSET 位为 1，当最大大小的数据包被加载到 FIFO 中去时，TXPKTRDY 位会被自动置 1。对于包长小于最大包长的数据包，TXPKTRDY 位必须手动置 1。当 TXPKTRDY 位被置 1 时，数据包已准备好发送，且 FIFONE 也置位。当数据包成功发送后，TXPKTRDY 和 FIFONE 位会被自动清零，并产生相应的发送端点中断。此时，下一个数据包可以被加载到 FIFO 中去。

双缓冲

如果发送端点的 FIFO 大小至少是此端点的最大包长（由寄存器 TXMAXP 设定，或者在支持动态 FIFO 深度功能时由 TXFIFOSIZE 设定）的两倍时，这时允许双缓冲，即两个数据包可以同时缓存在 FIFO 中。当每个数据包加载到发送 FIFO 中去时，USB 端点发送控制/状态寄存器 USB_TXxCSRL 的 TXPKTRDY 位必须置 1。如果 USB 端点发送控制/状态寄存器 TXxCSRH 的 AUTOSET 位为 1，当最大包长的数据包被加载到 FIFO 中去时，TXPKTRDY 位会被自动置 1。对于小于最大包长的数据包或高带宽同步传输/中断传输的场合，TXPKTRDY 位必须手动置 1。当 TXPKTRDY 位被置 1 时，数据包已准备好发送，且 FIFONE 也置位。当第一个数据包加载后，TXPKTRDY 位会被清零，并产生相应的发送端点中断。此时，第二个数据包可以被加载到 FIFO 中去，TXPKTRDY 位再次被置 1。这时，二个数据包都已准备好被发送。当二个数据包都已成功发送后，TXPKTRDY 位会被清零，并产生相应的发送端点中断。此时，下一个数据包可以被加载到 FIFO 中去。

需要注意的是，双包缓存发送禁止寄存器 USB_TXDPKTBUFDIS 的 EPx 位默认值为 1，这时相应端点禁止双包缓存。如果要使用双包缓存发送，需要将此位清零。

29. 4. 2. 3 设备模式下的OUT传输

在设备模式下，OUT 事务传输通过 USB 控制器的接收 FIFO 处理。当双缓冲使能时，二个数据包都可以缓存到 FIFO 中。当双缓冲不使能时，只有一个数据包可以缓存到 FIFO 中。

单缓冲

如果接收端点的 FIFO 大小小于该端点最大包长的两倍时，只能使用单缓冲，只有一个数据包可以被缓冲到 FIFO 中。当一个数据包接收到并缓存到 FIFO 中去时，USB 端点接收控制/状态寄存器 USB_RXxCSRL 的 RXPKTRDY 位和 FIFOFULL 位都会被置 1，告知用户可以从 FIFO 中去取接收到的数据。当接收的数据从 FIFO 中取出后，用户必须将 RXPKTRDY 位清零，以便允许接收更多数据包。此操作还会产生 ACK 信号给到到主机控制器。当 USB 端点接收控制/状态寄存器 USB_RXxCSRH 的 AUTOCLR 位被置 1 时，最大包长的数据包被从 FIFO 中取出后，RXPKTRDY 位和 FIFOFULL 位会被自动清

零。如果数据包不是最大包长，RXPKTRDY 位必须手动清零。

双缓冲

如果接收端点的 FIFO 大小至少是此端点最大包大小的两倍，这时允许双缓冲，则两包数据可以缓存在 FIFO 中。当第一包数据被接收并缓存在 FIFO 中后，USB 端点控制/状态寄存器 USB_RXxCSRL 的 RXPKTRDY 位被置 1，相应的接收端点中断会产生，告知用户可以从 FIFO 中去取接收到的数据。需要注意的是，当第一个数据包被接收到后，USB 端点接收控制/状态寄存器 USB_RXxCSRL 的 FIFOFULL 位不会被置 1。只有当第二个数据包被接收到后，USB 端点接收控制/状态寄存器 USB_RXxCSRL 的 FIFOFULL 位才会被置 1。

当所有数据包都从 FIFO 中取出后，用户必须将 RXPKTRDY 位清零，以便允许接收更多数据包。如果 USB 端点接收控制/状态寄存器 USB_RXxCSRH 的 AUTOCLR 位为 1，当最大包长的数据包被从 FIFO 中读出后，RXPKTRDY 位会被自动清零。对于小于最大包长的数据包，RXPKTRDY 位必须手动清零。

需要注意的是，双包缓存接收禁止寄存器 USB_RXDPKTBUFDIS 的 EPx 位默认值为 1，这时相应端点禁止双包缓存。如果要使用双包缓存接收，需要将此位清零。

29. 4. 2. 4 调度

设备控制器不能控制传输事务的调度，因为调度由主机控制器确定。USB 控制器可以随时建立传输事务。设备控制器等待来自主机控制器的请求，并在事务完成时产生中断。如果主机控制器发出请求，而设备控制器没有就绪，设备控制器向所有请求发送忙响应（NAK），直到它准备就绪。

29. 4. 2. 5 其他操作

USB 控制器自动响应 USB 总线上的某些状况或来自主机控制器的操作。

暂停控制传输

USB 控制器检测到下面这些情况会自动发出暂停（STALL）控制传输握手信号，情况如下：

在控制传输的输出数据过程中，主机发送的数据比建立过程中设备请求的数据多。当设备接收到最后一个输出数据包，并置位 USB 端点 0 控制和状态寄存器 USB_CSR0L 的 DATAEND 位后，主机却发送了输出令牌包（应该为输入令牌）。

在控制传输的输入数据过程中，主机请求的数据比建立过程中设备请求的数据多。当控制器收到最后一个数据包时，将 USB 端点发送控制/状态寄存器 USB_TXxCSRL 的 TXPKTRDY 位清零、USB 控制/状态寄存器 USB_CSR0L 的 DATAEND 位置 1 来指示当前是最后一个数据包，等待主机发出 ACK 信号。主机却发送了输入令牌包（应该为输出令牌包）。

主机发送大于 RXMAXP 字节的数据。

主机在输出阶段发送多于一个的零长度数据包。

零长度数据包

一个零长度的输出数据包用于指示控制传输的结束。在正常情况下，这个零长度数据包应该只在设备请求的全部数据已经传输完成之后才会被接收。但是，如果主机在设备请求的全部数据发送完成之前就发送零长度的 OUT 数据包，就表示此次数据传输提前结束。在这种情况下，USB 控制器自动清空 FIFO 中 IN 令牌阶段收到的全部数据，并将 USB_CSR0L 寄存器中的 DATAEND 位置 1。

设备地址设置

当主机尝试枚举 USB 设备时，主机会请求设备将其地址从零改到其他值。通过将主机请求的值写入 USB 设备寄存器 USB_FADDR 来实现地址的更改。但是在写入时应该小心，避免在事务完成之前更改地址。只能在 SET_ADDRESS 命令完成后才能去设置寄存器 USB_FADDR。SET_ADDRESS 命令下，设备在主机输入请求时发送零长度数据包来响应主机，表明 SET_ADDRESS 命令已完成。设备一旦响应输入请求，寄存器 USB_FADDR 需要立刻改写为新的值，以避免丢失发送到新地址的任何新命令。

29. 4. 2. 6 设备模式挂起

当 USB 总线空闲超过 3 ms，USB 控制器自动进入挂起模式，这时会立刻产生挂起中断（挂起中断已使能）。当处于挂起模式时，USB 的物理层也会进入挂起模式。当检测到恢复信号时，USB 控制器退出挂起模式。这时会产生恢复中断。通过设置 USB 电源寄存器 USB_POWER 的 RESUME 位，也可以强制 USB 控制器退出挂起模式。当该位置 1 时，USB 控制器退出挂起模式，并将恢复信号驱动到总线上。RESUME 位必须在 10 ms（最多 15 ms）之后清零，以结束恢复信号。

为满足电源功耗需求，USB 控制器可进入深睡眠模式。

29. 4. 2. 7 起始帧

当 USB 控制器工作在设备模式下时，它会接收到一个来至主机的起始帧数据包（SOF），起始帧数据包每毫秒发送一次。当接收到起始帧数据包时，包中包含的 11 位帧编号被写入寄存器 USB_FRAME，并且会产生一个 SOF 中断。一旦 USB 控制器已经开始接收起始帧数据包，它每一毫秒就需要成功接收到一次。如果在 1.00358ms 内没有接收到起始帧数据包，则认为该数据包已丢失，并且寄存器 USB_FRAME 不会被更新。当 SOF 包重新成功接收时，USB 控制器继续工作，并重新同步这些脉冲。

29. 4. 2. 8 USB复位

当 USB 控制器工作在器件模式下，在 USB 总线上检测到满足复位条件时，USB 控制器自动执行以下操作：

- ◇ 清零寄存器 USB_FADDR
- ◇ 清零寄存器 USB_INDEX
- ◇ 清空全部端点的 FIFO
- ◇ 清零全部控制/状态寄存器
- ◇ 使能全部端点中断
- ◇ 产生复位中断

29. 4. 2. 9 连接和断开

USB 控制器与 USB 总线的连接由软件处理。当控制器工作在设备模式下时，通过将寄存器 USB_POWER 中的 SOFTCONN 位置 1 或清零，可以将 USB 物理层在正常模式和非驱动模式间切换。USB 控制器的默认状态是非驱动模式。

将寄存器 USB_POWER 中的 SOFTCONN 位置 1 后，USB 物理层处在正常模式下，同时 USBDP 和 USBDM 差分线是使能的。这时和其他设备是连接的。将寄存器 USB_POWER 中的 SOFTCONN 位清 0 后，USB 物理层处在非驱动模式下，USBDP 和 USBDM 差分线为高阻态。这时和其他设备是断开连接的。

在产生系统复位后，寄存器 USB_POWER 中的 SOFTCONN 位被清零，在软件将该位置 1 前，控制器一直处于断开状态。应用软件可选择何时将物理层设置成正常模式。需较长初始化过程的系统，可采用此方法来确保初始化过程已结束且系统已准备好进行 USB 连接前的枚举。

当设备连接到主机时，USB 控制器不会产生中断。但是主机终止会话时，USB 控制器会产生中断。

29. 4. 3 主机模式

当 USB 控制器在主机模式下运行时，它可以于与另一个 USB 设备进行点对点通信，也可以通过集线器与多个设备通信，并且同时支持高速、全速、低速传输。USB 控制器的工作模式发生改变时，即从设备模式切换到主机模式或从主机模式切换到设备模式时，软件必须复位 USB 控制器，通过将寄存器 USB_SOFTRST 的 bit0 置 1 来实现。主机模式支持控制，批量，同步和中断传输。

当处于主机模式时，输入事务由端点的接收接口控制。所有输入事务使用接收端点寄存器，并且所有输出事务使用端点的发送端点寄存器。端点的 FIFO 应考虑端点最大数据包的大小。

批量传输

批量传输端点必须支持最大数据包的大小（最多 64 个字节），或两倍于最大包长（如果使用双缓冲）。

中断传输

中断传输端点必须支持最大数据包的大小（最多 64 个字节），或两倍于最大包长（如果使用双缓冲）。

同步传输

同步端点更灵活，最多可支持 1023 字节。

控制传输

控制传输可以指定单独的控制端点以与设备通信。在大多数情况下，USB 控制器应使用控制端点 0 作为专用的控制端点与设备的端点 0 进行通信。

29. 4. 3. 1 端点

端点寄存器用于控制 USB 端点接口，通过这些接口可与连接的设备进行通信。端点由一个专用控制输入\输出端点、5 个输出端点和 5 个输入端点组成。

专用的控制接口只能用于与设备的端点 0 之间的控制传输。他们用于设备枚举或其他使用设备端点的控制功能。控制端点的输入和输出事务共享 USB 控制器 FIFO 内存的前 64 字节。其余输入和输出接口可配置为与控制端点、批量端点、中断端点或同步端点通信。

这些 USB 接口可同时调度，用于与任何设备的任何端点的 7 个独立的输出事务和 7 个独立的输入事务。输入和输出控制有成对的三组寄存器。通过配置，它们可以与不同类型的端点以及不同设备的不同端点进行通讯。例如，第一对端点可分开控制，输出部分与设备的批量输出端点 1 通信，同时输入部分与设备的中断输出端点 2 通信。

无论用于点对点通信还是通过集线器通信，在访问设备之前，必须设置相关的 USB 端点接收功能地址寄存器 USB_RXxFUNCADDR 或 USB 端点发送功能地址寄存器 USB_TXxFUNCADDR，以记录每个接收和发送端点将被访问的设备地址。

USB 控制器支持通过 USB 集线器连接设备。

29. 4. 3. 2 主机模式下的IN传输

输入事务的处理，采用与设备模式处理输出事务类似的方式，但传输事务必须通过设置寄存器 USB_CSR0L 中的 REQPKT 位开始，向事务调度表明此端点存在一个正在运行的传输。此时事务调度向目标设备发送一个输入令牌包。

当接收到数据包且存到相应接收 FIFO 中去时，USB_CSR0L 寄存器中的 RXPKTRDY 位置 1，同时产生相应的接收端点中断信号，指示有一个数据包需要从 FIFO 中读出。

当数据包被读出时，必须将 RXPKTRDY 位清零。USB_RXxCSRH 寄存器中的 AUTOCLR 位可用于当最大包长的数据包从 FIFO 中读出时将 RXPKTRDY 位自动清零。

当 RXPKTRDY 位清零时，USB_RXxCSRH 寄存器中的 AUTOREQ 位将 REQPKT 位自动置 1。当将 RXPKTRDY 位清零时，控制器会向设备发生一条确认消息。当所要传输的数据包数量确定时，与该端点向关联的 USB 端点请求包数量寄存器 USB_EPx_RQPKTCOUNT 会被设置为要传输的数据包数量。每次请求前，USB 控制器都将 USB_EPx_RQPKTCOUNT 寄存器中的值减 1。当 USB_EPx_RQPKTCOUNT 寄存器中的值减到 0 时，将 AUTOREQ 位清零以阻止后面试图进行的事务。如果传输的数量未知，则应当将寄存器 USB_EPx_RQPKTCOUNT 清零。AUTOREQ 位保持置 1 状态，直到接收到一个短包（小于寄存器 USB_RXxMAXP 中的 MAXLOAD 值）才清零，此种情形只在批量传输的末尾才可出现。

如果设备用 NAK 响应批量或中断传输的输入令牌，USB 主机控制器将重试，直到达到设置的 NAK 限制次数。如果目标设备用 STALL 握手响应，USB 主机将不重试传输，而将寄存器 USB_RXxCSRL 中的 RXSTALL 位置位来产生中断。如果目标设备在规定的时间内不响应输入令牌包，或者包存在 CRC 或位填充错误，USB 主机将重试传输。如果三次重试，目标设备仍无响应，USB 控制器将 REQPKT 位清零，将 USB_RXxCSRL 寄存器中的 ERROR 位置 1 来产生中断。

29. 4. 3. 3 主机模式下的OUT传输

当数据包装载到发送 FIFO 中时，USB_TXxCSRL 寄存器中的 TXPKTRDY 位必须置 1。如果将 USB_TXxCSRH 寄存器中的 AUTOSET 位置 1，当最大包长的数据包装载到 FIFO 中，TXPKTRDY 位自动置 1。

如果目标设备用 NAK 响应输出令牌包，USB 主机控制器将重试，直到达到设置的 NAK 限制次数。如果目标设备用 STALL 握手响应，USB 主机将不重试传输，而通过将 USB_TXxCSRL 寄存器中的 RXSTALL 位置 1 来中断主处理器。如果目标设备在需要的时间内不响应输出令牌包，或者存在 CRC 或位填充错误，USB 主机将重试传输。如果三次重试，目标设备仍无响应，USB 控制器将清空 FIFO，置 1 寄存器 USB_TXxCSRL 的 ERROR 位。

29. 4. 3. 4 事务调度

事务调度由 USB 主机控制器自动处理。主机控制器允许根据端点事务类型配置端点通讯调度。中断传输可以是每 1 帧进行一次，也可以每 255 帧进行一次，可以在 1 帧到 255 之间以 1 帧增量调度。批量端点不允许调度参数，但在设备的端点不响应时，允许 NAK 超时。同步端点可以在每帧到每 216 帧之间调度。

USB 控制器维持帧计数。如果目标设备为全速设备，控制器在每帧开始时自动发送 SOF 包，同时帧计数加 1。如果目标设备为低速设备，将在总线上发送 K 状态来保持总线活动，防止低速设备进入挂起模式。

在 SOF 包发送后，USB 主机控制器巡检所有配置好的端点，寻找激活的传输事务。REQPKT 位置 1 的接收端点或 TXPKTRDY 位置 1 的发送端点，被视为存在激活的传输事务。

如果传输建立在一帧的第一个调度周期，而且端点的间隔计数器减到 0，则同步传输和中断传输开始。所以每个端点的中断传输和同步传输每 x 帧才发生一次，其中 x 是通过 USB 端点 x 主机发送间隔（USB_TXxINTERVAL）或 USB 端点 x 主机接收间隔（USB_RXxINTERVAL）寄存器设置的间隔。

如果在帧中下一个 SOF 包之前有足够的时间完成传输，则激活的批量传输立即开始。如果传输需要重发时（例如，收到 NAK 或设备未响应），需要在调度器先检查完其他所有端点是否有其它激活的传输之后，传输才能重传。这保证了一个发送大量 NAK 响应的端点不阻塞总线上的其他传输正常进行。控制器同样允许用户设置目标设备发送 NAK 的端点的超时限制。

29. 4. 3. 5 USB集线器

当低速设备或全速设备通过 USB 2.0 集线器连接到 USB 主机，集线器的地址和端口的详细信息必须记录在相应的 USB 端点 x 接收集线器地址寄存器 USB_RXxHUBADDR 和 USB 端点 x 接收集线器端口寄存器 USB_RXxHUBPORT 或 USB 端点 x 发送集线器地址寄存器 USB_TXxHUBADDR 和 USB 端点 x 发送集线器端口寄存器 USB_TXxHUBPORT 中。此外，设备的运行速度（全速或低速）必须记录在 USB 端点 0 发送配置寄存器 USB_TYPE0、USB 端点 x 主机配置发送类型寄存器 USB_TXxTYPE，或者 USB 端点 x 主机配置接收类型寄存器 USB_RXxTYPE 中。

对于集线器通信，这些寄存器的设置记录了接入的 USB 设备当前相应端点的配置。为了

支持更多数量的设备，USB 主机控制器允许通过简单的更新记录在这些寄存器中的地址和速度信息来动态改变这些配置。设备功能端点配置的改变必须在相关端点正在进行的传输完成后进行。

29. 4. 3. 6 干扰

只有总线至少空闲一个最小间隔包的时间，USB 主机控制器才会开始传输。USB 控制器不会发起事务传输，除非它能在结束帧前完成。如果在结束帧时 USB 总线上仍有活动，USB 主机将假定连接的目标设备发生故障，同时 USB 控制器挂起所有传输事务，并产生干扰（Babble）中断信号。

29. 4. 3. 7 主机挂起

如果 USB_POWER 寄存器中的 SUSPEND 位置 1，USB 主机控制器完成当前的传输事务，然后停止事务调度和帧计数。此时，不再启动事务传输，不再产生 SOF 包。

要想离开挂起模式，可以将 RESUME 位置 1 并将 SUSPEND 位清零。当 RESUME 位置 1 时，USB 主机将在总线上产生恢复信号。但 20 ms 之后，必须将 RESUME 位清零，此时，帧计数和事务调度开始。主机支持远程恢复检测。

29. 4. 3. 8 USB复位

如果 USB_POWER 寄存器中的 RESET 位置 1，USB 主机控制器将在总线上产生 USB 复位信号。RESET 位需要保持置位至少 20 ms，以确保目标设备的正确复位。软件清除此位后，USB 主机控制器开始帧计数和事务调度。

29. 4. 3. 9 连接/断开

通过将 USB 设备控制寄存器 USB_DEVCTL 中的 SESSION 位置 1 来启动会话。当检测到设备时，将产生连接中断信号。连接的设备的速度，通过读寄存器 USB_DEVCTL 来确定。如果 FSDEV 位置 1，连接的设备为全速设备；如果 LSDEV 位置 1，连接的设备为低速设备。USB 控制器必须对设备发出一个复位信号，此时 USB 主机开始设备枚举。如果会话过程中设备断开连接，将产生断开中断。

29. 4. 4 OTG模式

为了节省电源，USB OTG 可以当需要时才给 VBUS 上电，当不使用 USB 总线时，则关断 VBUS。VBUS 的供电总是由总线上的 A 设备提供。OTG 控制器通过物理层采样 ID 输入来决定哪个是 A 设备哪个是 B 设备。ID 信号拉低时，检测到插入 A 设备（表示 OTG 控制器作为 A 设备角色）；ID 信号为高时，检测到插入 B 设备（表示 OTG 控制器作为 B 设备角色）。注意当在 OTG A 和 OTG B 之间切换时，控制器保留所有的寄存器内容。

29. 4. 4. 1 开始会话

当 USB OTG 控制器准备开始会话时，USB_DEVCTL 寄存器中的 SESSION 位必须置 1。此时 OTG 控制器使能 ID 管脚检测。当检测到 A 类型连接时，ID 输入为低；当检测到 B 类型连接时，ID 输入为高。同时将 USB_DEVCTL 寄存器中的 BDEVICE 位置 1，表明 USB OTG 控制器用作 A 设备还是 B 设备。该 USB OTG 控制器提供中断信号，表明 ID 脚已完成检测，USB_DEVCTL 存器中的 BDEVICE 位设置是有效的。此中断通过设置寄存器 USB_USBIE 启用，可以通过寄存器 USB_USBIS 查看其状态。当 USB 控制器检测到处于线缆的 A 端，必须尽快在 100ms 内启用 VBUS 电源。

如果 USB OTG 控制器是 A 设备，则它进入主机模式（A 设备总是默认为主机），打开 VBUS 电源，等待 VBUS 达到有效门限以上（USB_DEVCTL 寄存器中的 VBUS 位域值为 0x3）。此时，OTG 控制器等待外设接入。当检测到外设接入，则产生一个连接中断信号，USB_DEVCTL 寄存器中的 FSDEV 或 LSDEV 位置 1（取决于接入的全速设备还是低速设备）。这时，USB 控制器向接入的设备发送一个复位信号。可以通过将 USB_DEVCTL 寄存器中的 SESSION 位清零来结束会话。如果发生干扰或 VBUS 掉到会话有效电压以下，OTG 控制器将自动结束会话。

需要注意的是，当接入大电流设备，可能会造成 USB OTG 控制器不能保留在主机模式。一些设备吸收大量电流会暂时将 VBUS 拉低到有效电压之下，造成控制器退出主机模式。保持控制器在主机模式，唯一的方法是允许 VBUS 电压低于会话结束电平。此时，设备引起 VBUS 掉到有效电压下后，又恢复正常。

此外，当接入设备被告之可以启动它的激活配置时，USB OTG 控制器可能不能保留在主机模式。此时，接入设备吸取大量的电流，可能造成 VBUS 电压跌落到有效电压以下。

如果 OTG 控制器用作 B 设备，它使用 USB OTG 规范中定义的会话请求协议来请求会话。首先它释放 VBUS，然后当 VBUS 电平低于会话结束门限（USB_DEVCTL 寄存器中的 VBUS 位域值为 0x0），总线保持 SE0 电平状态时间大于 2 ms，OTG 控制器将依次向数据线和 VUBS 线发送脉冲。会话结束时，SESSION 位可通过 OTG 控制器或应用软件清零。OTG 控制器使物理层切断 USBDP 上的上拉电阻，向 A 设备发送会话结束信号。

29. 4. 4. 2 活动性检测

当 OTG 的其他设备希望发起会话时，如果它是 A 设备，将 VBUS 电平提升到会话有效电压之上；如果它是 B 设备，它将向数据线和 VBUS 线发送脉冲。USB 控制器依据这些动作，确定发起建立会话的是 A 设备还是 B 设备。如果 VBUS 电平提升到会话有效电平之上，则 USB 控制器用作 B 设备。USB 控制器将 USB_DEVCTL 寄存器中的 SESSION 位置 1。当检测到总线上有复位信号，将产生复位中断信号，作为会话的起始信号。

USB 控制器的默认模式是用作 B 设备时的设备模式。会话结束时，A 设备关断 VBUS 电源。当 VBUS 下降到会话有效电平之下时，USB 控制器将检测到此下降信号，并将 SESSION 位清零来指出会话已经结束，同时引发一个断开连接中断信号。如果检测到数据线和 VBUS 上的脉冲，则 USB 控制器用作 A 设备。该控制器产生会话请求中断信号来指示 B 设备在请求会话。只有将 USB_DEVCTL 寄存器中的 SESSION 位置 1 才可启动会话。

29. 4. 4. 3 主机协商

当 USB 控制器是 A 设备，ID 信号为低，当会话发起时它将自动进入主机模式。当 USB 控制器是 B 设备，ID 信号为高，当会话发起时它将自动进入设备状态。但是页可以通过软件方式将寄存器 USB_DEVCTL 中的 HOSTREQ 位置 1 使 USB 控制器从设备模式变为主机模式。此位可以在通过置 1 寄存器 USB_DEVCTL 中的 SESSION 位发起会话请求的同时置位，也可以在发起请求之后任意时刻置位。当 USB 控制器下次进入挂起模式时，如果 HOSTREQ 位保持置 1，控制器进入主机模式，并开始主机协商，引发物理层断开 D+线路上的上拉电阻，触发之前的 A 设备切换到设备模式，并连接设备模式的上拉电阻。当 USB 控制器检测到此情况，将产生连接中断信号，并将寄存器 USB_POWER

中的 RESET 位置 1 使之前的 A 设备置位。USB 控制器自动开始复位序列，确保复位在之前的 A 设备连接上拉电阻后的 1ms 内开始。主处理器应等待至少 20 ms，然后将 RESET 位清零，开始枚举之前的 A 设备。

当 USB OTG 控制器 B 设备使用完总线，它将寄存器 USB_POWER 中的 SUSPEND 位置 1 进入挂起模式。A 设备检测此情况，则结束会话或恢复到主机模式。如果 A 设备是 OTG 控制器，将产生一个连接断开的中断信号。

29. 4. 5 DMA控制器

该 DMA 控制器支持两个 DMA 模式，为 DMA 模式 0 和 DMA 模式 1。它能处理的最大数据包大小为 8K。除此之外，该控制器可被编程为使用 INCR4，INCR8 和 INCR16 4/8/16 节拍递增的突发而不是使用未声明长度的突发来进行传输。

当工作在 DMA 模式 0 下时，DMA 控制器只能被编程为加载/卸载一个数据包，因此每个通过 USB 传输的数据包都需要处理器干预。该模式可用于任何端点，无论该端点使用的是控制，批量，同步或中断事务（也就是说，同样包括端点 0）。

当工作在 DMA 模式 1 下时，DMA 控制器可以被编程为加载/卸载一个完整的批量传输（可以是多个数据包）。一旦设置好，DMA 控制器将加载/卸载传输的所有数据包，只有在传输完成时才中断处理器。DMA 模式 1 只能用于使用批量事务的端点。

每一个通道都可以根据所选的工作模式独立编程。

29. 4. 5. 1 DMA寄存器

该 DMA 控制器有一个中断寄存器用来指示哪个通道有一个挂起的中断和一组用于每个配置通道的三个控制寄存器。关于 DMA 寄存器的完整描述在寄存器描述章节中。

地址*	寄存器	描述
200h	USB_DMA_INTR	DMA 中断寄存器
204h + (n-1)*10h	USB_DMAx_CNTL	DMA 通道控制寄存器
208h + (n-1)*10h	USB_DMAx_ADDR	DMA 通道地址寄存器
20Ch + (n-1)*10h	USB_DMAx_COUNT	DMA 通道计数寄存器

*n = 通道号 1 到 8

29. 4. 5. 2 DMA总线周期

DMA 控制器在 AHB 总线上使用递增的突发。当 DMA 控制器首先被授予总线主控权（无论是在 USB 数据包的开始还是当通过数据包的一部分被丢弃后重新获得总线时），并且当 AHB 地址启动一个新的 1K 的字节块时，DMA 控制器开启一个新的突发。需要注意的是，在 1K 边界开启新的突发的需求是由 DMA 控制器自动处理的。用户不需要在编程 DMA 控制器时考虑这些边界。

根据 DMA 通道的编程方式，传输的数据包的大小和相对于下一个 1K 边界的位置，这些突发可能是 4 拍，8 拍，16 拍或者是未声明的长度。USB_DMAx_CNTL 寄存器的 Bit 10-9 位选择了突发模式 0-3，这些模式反过来定义了可以使用哪些突发类型（参考 DMA 控制器章节）。比如，选择突发模式 2，可以使用 8 拍（INCR8），4 拍（INCR4）和未声明长度的突发，但是不能使用 16 拍（INCR16）的突发。在 DMA 模式 0 和 DMA 模式 1 下，

传输选择的突发模式没有限制。

每一个数据包的传输通常是使用字传输（32 位）进行的，但是在数据包传输结束的时候可能会有额外的字节或者半字传输。由于起始地址（写入 USB_DMAx_ADDR 寄存器）必须是字对齐的，因此数据包传输会以字传输开始，但是在传输结束时可能会增加半字和/或字节传输来处理残留数据。支持 AHB 拆分事务和重试。

29. 4. 5. 3 总线错误

如果当 DMA 控制器访问 AHB 总线上的存储器时发生总线错误，DMA 控制器将立刻终止 DMA 传输并且将 USB_DMAx_CNTL 寄存器中的 DMA_ERR 位（Bit 8）置位来中断处理器。

注：该中断的生成不受 USB_DMAx_CNTL 寄存器中的 DMAIE 位（bit 3）置位的影响。该中断在 USB_DMAx_CNTL.DMAIE=0 时仍然会生成。

29. 4. 5. 4 传输数据包

使用内置的 DMA 控制器访问 USB 控制器 FIFOs 需要对 DMA 控制器和 USB 控制器端点进行适当的编程。很多变化都是有可能的。以下部分详细介绍了用于传输单个数据包和多个数据包的基本操作的标准设置。

单个数据包：RX 端点

单个数据包的传输通常使用 DMA 模式 0 进行。

在这种情况下，USB 控制器 Rx 端点应该被编程为如下：

1. USB_RXIE 寄存器中相关的中断使能位置为 1
2. 适当的 USB_IND_RXCSRH 或 USB_RXxCSRH 寄存器中的 DMAREQENAB 位（Bit 5）置为 0（注：在此操作中，不需要设置 USB 控制器来支持 DMA）。

当 USB 控制器接收到一个数据包时，会生成适当的端点中断。然后处理器应该对 DMA 控制器选择的通道进行如下编程：

ADDR：存储数据包的存储器地址

COUNT：数据包的大小（通过读 USB_IND_RXCOUNT 或 USB_IND_COUNT0 寄存器确定）

CNTL：DMA_EN 位（Bit 0）为 1；DMA_DIR 位（Bit 1）为 0；DMAMODE 位（Bit 2）为 0；DMAIE 位（Bit 3）为 1；所需的 DMA_BRSTM 位（Bit 10-9）。

然后 DMA 控制器将请求总线主控权，并将数据包传输到存储器中。当传输完成时，会生成 DMA 中断（DMA_NINT 取低）。然后处理器应该清除 USB_IND_RXCSRL 或 USB_RXxCSRL 寄存器中的 RXPKTRDY 位。

单个数据包：TX 端点

要使用 DMA 模式 0 进行该操作，USB 控制器 Tx 端点应该被编程为如下：

1. USB_TXIE 寄存器中相关的中断使能位置为 1
2. 适当的 USB_IND_TXCSRH 或 USB_TXxCSRH 寄存器中的 DMAREQENAB 位（Bit

4）置为 0（注：在此操作中，不需要设置 USB 控制器来支持 DMA）。

当 USB 控制器中的 FIFO 变为可用时，USB 控制器将会使用适当的 Tx 端点中断来中断处理器。然后处理器应该按如下编程 DMA 控制器：

ADDR：要发送的数据包的存储器地址

COUNT：要发送的数据包的大小

CNTL：DMA_EN 位（Bit 0）为 1；DMA_DIR 位（Bit 1）为 1；DMAMODE 位（Bit 2）为 0；DMAIE 位（Bit 3）为 1；所需的 DMA_BRSTM 位（Bit 10-9）。

然后 DMA 控制器将请求总线主控权，并将数据包传输到 USB 控制器 FIFO 中。当传输完成时，会生成 DMA 中断。然后处理器应该将 USB_IND_TXCSRL 或 USB_TXxCSRL 寄存器中的 TXPKTRDY 位置位。

多个数据包：RX 端点

多个数据包的传输通常使用 DMA 模式 1 进行。

当使用 DMA 模式 1 接收多个数据包时，DMA 控制器应该按如下编程：

ADDR：存储传输的缓冲区存储器地址

COUNT：数据缓冲区的最大大小

CNTL：DMA_EN 位（Bit 0）为 1；DMA_DIR 位（Bit 1）为 0；DMAMODE 位（Bit 2）为 1；DMAIE 位（Bit 3）为 1；所需的 DMA_BRSTM 位（Bit 10-9）。

并且 USB 控制器 Rx 端点应该按如下编程：

1. USB_RXIE 寄存器中相关的中断使能位应该置为 1
2. 适当的 USB_IND_RXCSRH 或 USB_RXxCSRH 寄存器中的 AUTOCLR 位（bit 7），DMAREQENAB 位（bit 5）和 DMAREQMODE 位（bit 3）应该置为 1。在主模式下，AUTOREQ 位（bit 6）也应该置为 1，USB_EPx_RQPKTCOUNT 寄存器应该使用传输中的数据包数目来编程。

随着 USB 控制器接收到每一个数据包，DMA 控制器将请求总线主控权并将数据包传输到存储器中。在自动清零设置下，USB 控制器会自动清除 RXPKTRDY 位。

在外设模式下或者当 USB_EPx_RQPKTCOUNT 寄存器为 0 时，该流程会自动继续进行直到 USB 控制器接收到一个“短的数据包”（小于端点设定的最大数据包大小），表示传输结束。DMA 控制器不会传输该数据包，反而，USB 控制器会通过生成适当的端点中断来中断处理器。然后处理器读取 USB_IND_RXCOUNT 或 USB_IND_COUNT0 寄存器来查看该包的大小，该数据包需要通过手动卸载或者在模式 0 中重新编程 DMA 控制器来卸载。

在主机模式 AUTOREQ 位置位和 USB_EPx_RQPKTCOUNT 寄存器非零的情况下，内核会在每次请求之后递减 USB_EPx_RQPKTCOUNT 寄存器中的值。当值从 1 递减至 0，AUTOREQ 位会被清零，以避免尝试任何进一步的事务。

USB_DMAx_ADDR 寄存器会随着数据包的卸载而递增，因此，处理器可以通过比较 USB_DMAx_ADDR 寄存器当前的值与存储器缓冲区中的初始地址来决定传输的大小。

注：如果传输的大小超过了数据缓冲区的大小，DMA 控制器会停止卸载 FIFO 并且通过 DMA_NINT 线来中断处理器。

多个数据包：TX 端点

要使用 DMA 模式 1 进行该操作，DMA 控制器应该按如下编程：

ADDR：要发送的数据块的存储器地址

COUNT：数据块大小

CNTL：DMA_EN 位（Bit 0）为 1；DMA_DIR 位（Bit 1）为 1；DMAMODE 位（Bit 2）为 1；DMAIE 位（Bit 3）为 1；所需的 DMA_BRSTM 位（Bit 10-9）。

并且 USB 控制器 Tx 端点应该按如下编程：

1. USB_TXIE 寄存器中相关的中断使能位应该置为 1（这样就可以检测到错误）
2. 适当的 USB_IND_TXCSRH 或 USB_TXxCSRH 寄存器中的 AUTOSET 位（Bit7），DMAREQENAB 位（Bit4）和 DMAREQMODE 位（Bit2）应该置为 1。

当 USB 控制器中的 FIFO 变为可用时，DMA 控制器将请求总线主控权，并将数据包传输到 FIFO 中。置位了 AUTOSET 位，USB 控制器将自动将 TXPKTRDY 位置位。该流程会继续进行直到整个数据块被传输到 USB 控制器。然后 DMA 控制器会通过对 DMA_NINT 取低中断处理器。如果最后一个加载的数据包小于端点的最大数据包大小，TXPKTRDY 位不会因为这个数据包被置位：因此处理器会通过将 TXPKTRDY 位置位来应答 DMA 中断，使得最后一个“短的数据包”被发送。如果最后一个加载的数据包就是最大数据包大小，那么，要采取的行动取决于传输是否在一个应用程序的控制之下，例如 Windows 系统上的海量存储软件，该软件记录发送的各个数据包的数量。如果传输不在这样的控制之下，处理器仍然会通过将 TXPKTRDY 位置位来应答 DMA 中断。这样做的效果是发送一个空的数据包，以便接收软件将其解释为表示传输结束。

29. 5 特殊功能寄存器

29. 5. 1 寄存器列表

注意，USB 时钟启动 8 个时钟后，才能访问 USB 控制器的寄存器。

USB 寄存器划分如下所示：

基础寄存器（0x00~0x0F）

这部分寄存器提供对 USB 内核的控制和状态描述。

索引端点寄存器（0x10~0x1F）

这部分寄存器提供对当前选中的端点进行控制和状态描述。

FIFO 寄存器（0x20~0x3F）

这部分寄存器是端点数据访问寄存器。

附加控制和状态寄存器（0x60~0x7F）

这部分寄存器是附加的设备状态和控制寄存器。

目标端点控制寄存器（0x80~0xBF）

这部分寄存器提供每个端点的目标功能和集线器地址详细信息。只有使能了多端点功能后，才能访问这部分寄存器。

未索引端点寄存器（0x100~0x17F）

这部分寄存器功能同索引端点寄存器完全一致，这部分寄存器描述的未被索引的端点。比如，偏移地址为 0x100~0x10F 的寄存器描述端点 0 的信息，偏移地址为 0x110~0x11F 的寄存器描述端点 1 的信息。

DMA 控制寄存器（0x200~0x300）

这部分寄存器用于配置 USB 内置 DMA 控制器。

数据包请求计数寄存器（0x304~0x33C）

这部分寄存器在主机模式下与自动请求（autoreq）一起使用。

双缓冲禁止寄存器（0x340~0x343）

这部分寄存器提供禁用双缓冲的直接用户控制。

LPM 管理寄存器（0x360~0x365）

这部分寄存器用于设置低功耗管理。

USB 寄存器列表		
基础寄存器（0x00~0x0F）		
名称	偏移地址	描述
USB_FADDR	0000_H	设备功能地址寄存器
USB_POWER	0001_H	设备电源控制寄存器
USB_TXIS	0002_H	端点发送中断寄存器
USB_RXIS	0004_H	端点接收中断寄存器
USB_TXIE	0006_H	端点发送中断使能寄存器
USB_RXIE	0008_H	端点接收中断使能寄存器
USB_USBIS	$000A_H$	通用 USB 中断寄存器
USB_USBIE	$000B_H$	通用 USB 中断使能寄存器
USB_FRAME	$000C_H$	USB 帧值寄存器
USB_INDEX	$000E_H$	端点索引寄存器
USB_TEST	$000F_H$	USB 测试模式寄存器
索引端点寄存器（设备）（0x10~0x1F）		
USB_IND_TXMAXP	0010_H	索引端点发送最大传输数据寄存器
USB_IND_CSR0L/ USB_IND_TXCSRL	0012_H	索引端点 0 控制和状态低字节寄存器/ 索引端点发送控制和状态低字节寄存器
USB_IND_CSR0H/ USB_IND_TXCSRH	0013_H	索引端点 0 控制和状态高字节寄存器/ 索引端点发送控制和状态高字节寄存器
USB_IND_RXMAXP	0014_H	索引端点接收最大传输数据寄存器
USB_IND_RXCSRL	0016_H	索引端点接收控制和状态低字节寄存器
USB_IND_RXCSRH	0017_H	索引端点接收控制和状态高字节寄存器
USB_IND_COUNT0/ USB_IND_RXCOUNT	0018_H	索引端点 0 接收字节数量寄存器/ 索引端点接收字节数量寄存器
Reserved	$001A_H$~$001E_H$	保留
USB_IND_CONFIG0/ USB_IND_FIFOSIZE	$001F_H$	索引端点 0 的控制和状态高字节寄存器/ 索引端点 FIFO 长度寄存器
索引端点寄存器（主机）（0x10~0x1F）		
USB_IND_TXMAXP	0010_H	索引端点发送最大传输数据寄存器
USB_IND_CSR0L/ USB_IND_TXCSRL	0012_H	索引端点 0 控制和状态低字节寄存器/ 索引端点发送控制和状态低字节寄存器
USB_IND_CSR0H/ USB_IND_TXCSRH	0013_H	索引端点 0 控制和状态高字节寄存器/ 索引端点发送控制和状态高字节寄存器
USB_IND_RXMAXP	0014_H	索引端点接收最大传输数据寄存器
USB_IND_RXCSRL	0016_H	索引端点接收控制和状态低字节寄存器
USB_IND_RXCSRH	0017_H	索引端点接收控制和状态高字节寄存器
USB_IND_COUNT0/ USB_IND_RXCOUNT	0018_H	索引端点 0 接收字节数量寄存器/ 索引端点接收字节数量寄存器
USB_IND_TYPE0/ USB_IND_TXTYPE	$001A_H$	索引端点 0 主机发送配置寄存器/ 索引端点主机发送配置寄存器
USB_IND_NAK/	$001B_H$	索引端点 0 无应答超时时间设置寄存器/

USB_IND_TXINTERVAL		索引端点主机发送轮询间隔寄存器
USB_IND_RXTYPE	$001C_H$	索引端点主机接收配置寄存器
USB_IND_RXINTERVAL	$001D_H$	索引端点主机接收轮询间隔寄存器
Reserved	$001E_H$	保留
USB_IND_CONFIG0/ USB_IND_FIFOSIZE	$001F_H$	索引端点 0 的控制和状态高字节寄存器/ 索引端点 FIFO 长度寄存器
FIFO 寄存器（0x20~0x3F）		
USB_FIFO0	0020_H	端点 0 FIFO 访问寄存器
USB_FIFO1	0024_H	端点 1 FIFO 访问寄存器
USB_FIFO2	0028_H	端点 2 FIFO 访问寄存器
USB_FIFO3	$002C_H$	端点 3 FIFO 访问寄存器
USB_FIFO4	0030_H	端点 4 FIFO 访问寄存器
USB_FIFO5	0034_H	端点 5 FIFO 访问寄存器
Reserved	0038_H~$003F_H$	保留
附加控制和状态寄存器（0x60~0x7F）		
USB_DEVCTL	0060_H	USB 设备控制寄存器
USB_DMACFG	0061_H	USB DMA 配置寄存器
USB_TXFIFOSIZE	0062_H	端点发送 FIFO 长度寄存器
USB_RXFIFOSIZE	0063_H	端点接收 FIFO 长度寄存器
USB_TXFIFOADD	0064_H	发送端点 FIFO 起始地址寄存器
USB_RXFIFOADD	0066_H	接收端点 FIFO 起始地址寄存器
Reserved	0068_H~0077_H	保留
USB_EPINFO	0078_H	发送、接收端点信息寄存器
USB_RAMINFO	0079_H	RAM 宽度、DMA 通道信息寄存器
USB_LINKINFO	$007A_H$	连接时序寄存器
USB_VPLEN	$007B_H$	VBUS 脉冲时序寄存器
USB_HS_EOF1	$007C_H$	高速传输时间缓冲寄存器
USB_FS_EOF1	$007D_H$	全速传输时间缓冲寄存器
USB_LS_EOF1	$007E_H$	低速传输时间缓冲寄存器
USB_SOFTRST	$007F_H$	软件复位寄存器
目标端点控制寄存器（0x80~0xBF）		
USB_TX0FUNCADDR	0080_H	发送端点 0 功能地址寄存器
USB_TX1FUNCADDR	0088_H	发送端点 1 功能地址寄存器
USB_TX2FUNCADDR	0090_H	发送端点 2 功能地址寄存器
USB_TX3FUNCADDR	0098_H	发送端点 3 功能地址寄存器
USB_TX4FUNCADDR	$00A0_H$	发送端点 4 功能地址寄存器
USB_TX5FUNCADDR	$00A8_H$	发送端点 5 功能地址寄存器
USB_TX0HUBADDR	0082_H	发送端点 0 集线器地址寄存器
USB_TX1HUBADDR	$008A_H$	发送端点 1 集线器地址寄存器
USB_TX2HUBADDR	0092_H	发送端点 2 集线器地址寄存器
USB_TX3HUBADDR	$009A_H$	发送端点 3 集线器地址寄存器
USB_TX4HUBADDR	$00A2_H$	发送端点 4 集线器地址寄存器
USB_TX5HUBADDR	$00AA_H$	发送端点 5 集线器地址寄存器

USB_TX0HUBPORT	0083_H	发送端点 0 集线器端口寄存器
USB_TX1HUBPORT	$008B_H$	发送端点 1 集线器端口寄存器
USB_TX2HUBPORT	0093_H	发送端点 2 集线器端口寄存器
USB_TX3HUBPORT	$009B_H$	发送端点 3 集线器端口寄存器
USB_TX4HUBPORT	$00A3_H$	发送端点 4 集线器端口寄存器
USB_TX5HUBPORT	$00AB_H$	发送端点 5 集线器端口寄存器
USB_RX1FUNCADDR	$008C_H$	接收端点 1 功能地址寄存器
USB_RX2FUNCADDR	0094_H	接收端点 2 功能地址寄存器
USB_RX3FUNCADDR	$009C_H$	接收端点 3 功能地址寄存器
USB_RX4FUNCADDR	$00A4_H$	接收端点 4 功能地址寄存器
USB_RX5FUNCADDR	$00AC_H$	接收端点 5 功能地址寄存器
USB_RX1HUBADDR	$008E_H$	接收端点 1 集线器地址寄存器
USB_RX2HUBADDR	0096_H	接收端点 2 集线器地址寄存器
USB_RX3HUBADDR	$009E_H$	接收端点 3 集线器地址寄存器
USB_RX4HUBADDR	$00A6_H$	接收端点 4 集线器地址寄存器
USB_RX5HUBADDR	$00AE_H$	接收端点 5 集线器地址寄存器
USB_RX1HUBPORT	$008F_H$	接收端点 1 集线器端口寄存器
USB_RX2HUBPORT	0097_H	接收端点 2 集线器端口寄存器
USB_RX3HUBPORT	$009F_H$	接收端点 3 集线器端口寄存器
USB_RX4HUBPORT	$00A7_H$	接收端点 4 集线器端口寄存器
USB_RX5HUBPORT	$00AF_H$	接收端点 5 集线器端口寄存器
未索引端点寄存器（0x100~0x17F）		
USB_CSR0L	0102_H	端点 0 控制和状态低字节寄存器
USB_CSR0H	0103_H	端点 0 控制和状态高字节寄存器
USB_COUNT0	0108_H	端点 0 接收字节数量寄存器
USB_TYPE0	$010A_H$	端点 0 发送配置寄存器
USB_NAK	$010B_H$	端点 0 无应答超时时间设置寄存器
USB_CONFIG0	$010F_H$	端点 0 的控制和状态高字节寄存器
USB_TX1MAXP	0110_H	端点 1 发送最大传输数据寄存器
USB_TX2MAXP	0120_H	端点 2 发送最大传输数据寄存器
USB_TX3MAXP	0130_H	端点 3 发送最大传输数据寄存器
USB_TX4MAXP	0140_H	端点 4 发送最大传输数据寄存器
USB_TX5MAXP	0150_H	端点 5 发送最大传输数据寄存器
USB_TX1CSRL	0112_H	端点 1 发送控制和状态低字节寄存器
USB_TX2CSRL	0122_H	端点 2 发送控制和状态低字节寄存器
USB_TX3CSRL	0132_H	端点 3 发送控制和状态低字节寄存器
USB_TX4CSRL	0142_H	端点 4 发送控制和状态低字节寄存器
USB_TX5CSRL	0152_H	端点 5 发送控制和状态低字节寄存器
USB_TX1CSRH	0113_H	端点 1 发送控制和状态高字节寄存器
USB_TX2CSRH	0123_H	端点 2 发送控制和状态高字节寄存器
USB_TX3CSRH	0133_H	端点 3 发送控制和状态高字节寄存器
USB_TX4CSRH	0143_H	端点 4 发送控制和状态高字节寄存器
USB_TX5CSRH	0153_H	端点 5 发送控制和状态高字节寄存器

USB_RX1MAXP	0114_H	端点 1 接收最大传输数据寄存器
USB_RX2MAXP	0124_H	端点 2 接收最大传输数据寄存器
USB_RX3MAXP	0134_H	端点 3 接收最大传输数据寄存器
USB_RX4MAXP	0144_H	端点 4 接收最大传输数据寄存器
USB_RX5MAXP	0154_H	端点 5 接收最大传输数据寄存器
USB_RX1CSRL	0116_H	端点 1 接收控制和状态低字节寄存器
USB_RX2CSRL	0126_H	端点 2 接收控制和状态低字节寄存器
USB_RX3CSRL	0136_H	端点 3 接收控制和状态低字节寄存器
USB_RX4CSRL	0146_H	端点 4 接收控制和状态低字节寄存器
USB_RX5CSRL	0156_H	端点 5 接收控制和状态低字节寄存器
USB_RX1CSRH	0117_H	端点 1 接收控制和状态高字节寄存器
USB_RX2CSRH	0127_H	端点 2 接收控制和状态高字节寄存器
USB_RX3CSRH	0137_H	端点 3 接收控制和状态高字节寄存器
USB_RX4CSRH	0147_H	端点 4 接收控制和状态高字节寄存器
USB_RX5CSRH	0157_H	端点 5 接收控制和状态高字节寄存器
USB_RX1COUNT	0118_H	端点 1 接收字节数量寄存器
USB_RX2COUNT	0128_H	端点 2 接收字节数量寄存器
USB_RX3COUNT	0138_H	端点 3 接收字节数量寄存器
USB_RX4COUNT	0148_H	端点 4 接收字节数量寄存器
USB_RX5COUNT	0158_H	端点 5 接收字节数量寄存器
USB_TX1TYPE	$011A_H$	端点 1 主机发送配置寄存器
USB_TX2TYPE	$012A_H$	端点 2 主机发送配置寄存器
USB_TX3TYPE	$013A_H$	端点 3 主机发送配置寄存器
USB_TX4TYPE	$014A_H$	端点 4 主机发送配置寄存器
USB_TX5TYPE	$015A_H$	端点 5 主机发送配置寄存器
USB_TX1INTERVAL	$011B_H$	端点 1 主机发送轮询间隔寄存器
USB_TX2INTERVAL	$012B_H$	端点 2 主机发送轮询间隔寄存器
USB_TX3INTERVAL	$013B_H$	端点 3 主机发送轮询间隔寄存器
USB_TX4INTERVAL	$014B_H$	端点 4 主机发送轮询间隔寄存器
USB_TX5INTERVAL	$015B_H$	端点 5 主机发送轮询间隔寄存器
USB_RX1TYPE	$011C_H$	端点 1 主机接收配置寄存器
USB_RX2TYPE	$012C_H$	端点 2 主机接收配置寄存器
USB_RX3TYPE	$013C_H$	端点 3 主机接收配置寄存器
USB_RX4TYPE	$014C_H$	端点 4 主机接收配置寄存器
USB_RX5TYPE	$015C_H$	端点 5 主机接收配置寄存器
USB_RX1INTERVAL	$011D_H$	端点 1 主机接收轮询间隔寄存器
USB_RX2INTERVAL	$012D_H$	端点 2 主机接收轮询间隔寄存器
USB_RX3INTERVAL	$013D_H$	端点 3 主机接收轮询间隔寄存器
USB_RX4INTERVAL	$014D_H$	端点 4 主机接收轮询间隔寄存器
USB_RX5INTERVAL	$015D_H$	端点 5 主机接收轮询间隔寄存器
DMA 控制寄存器（0x200~0x300）		
USB_DMA_INTR	0200_H	DMA 中断寄存器
USB_DMAx_CNTL	$0204_H+(x-1)*10_H$	DMA 通道控制寄存器

USB_DMAx_ADDR	0208_H+(x-1)*10_H	DMA 通道地址寄存器
USB_DMAx_COUNT	$020C_H$+(x-1)*10_H	DMA 通道计数寄存器
数据包请求计数寄存器（0x304~0x33C）		
USB_EP1_RQPKTCOUNT	0304_H	端点 1 批量传输请求数据包数量寄存器
USB_EP2_RQPKTCOUNT	0308_H	端点 2 批量传输请求数据包数量寄存器
USB_EP3_RQPKTCOUNT	$030C_H$	端点 3 批量传输请求数据包数量寄存器
USB_EP4_RQPKTCOUNT	0310_H	端点 4 批量传输请求数据包数量寄存器
USB_EP5_RQPKTCOUNT	0314_H	端点 5 批量传输请求数据包数量寄存器
双缓冲禁止寄存器（0x340~0x343）		
USB_RXDPKTBUFDIS	0340_H	接收双包缓冲禁用寄存器
USB_TXDPKTBUFDIS	0342_H	发送双包缓冲禁用寄存器
LPM 管理寄存器（0x360~0x365）		
USB_LPM_ATTR	0360_H	LPM 属性寄存器
USB_LPM_CNTRL	0362_H	LPM 控制寄存器
USB_LPM_INTREN	0363_H	LPM 中断使能寄存器
USB_LPM_INTR	0364_H	LPM 中断状态寄存器
USB_LPM_FADDR	0365_H	LPM 功能地址寄存器

29. 5. 2 寄存器描述

29. 5. 2. 1 设备功能地址寄存器（USB_FADDR）

USB_FADDR 是一个 8 位寄存器，其中 7 位用作传输设备地址。当 USB 控制器用于设备模式时，通过 SET_ADDRESS 命令将收到的地址写入该寄存器，用于随后的令牌包功能地址解码。

设备模式：

设备功能地址寄存器（USB_FADDR）							
偏移地址：000_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved	FADDR						

Reserved	Bit 7	-	**保留**
FADDR	Bit 6-0	R/W	**USB 设备功能地址设置位** 设备地址。

29. 5. 2. 2 设备电源控制寄存器（USB_POWER）

USB_POWER 是用于控制挂起和恢复信号，以及一些基本操作的 8 位寄存器

- 主机模式：

设备电源控制寄存器（USB_POWER）							
偏移地址：001_H							
复位值：00100000_B							
7	6	5	4	3	2	1	0
Reserved	Reserved	HS_EN	HS_M	RESET	RESUME	SUSPEND	EN_SPDM

Reserved	Bit 7~6	-	**保留**
HS_EN	Bit 5	R/W	**高速传输使能位** 0：禁止 1：使能 当处于使能状态时，在 USB 复位期间会进行高速传输模式的识别和握手处理。
HS_M	Bit 4	R	**高速传输模式标志位** 0：非高速传输模式 1：高速传输模式； 在主机模式时，RESET 信号被清零之后此信号有效。在会话请求期间此信号保持有效。
RESET	Bit 3	R/W	**复位信号** 0：USB D+/D-不产生复位信号 1：USB D+/D-产生复位信号
RESUME	Bit 2	R/W	**唤醒控制位** 0：不产生唤醒信号 1：产生唤醒信号 在主机模式时，置 1 之后应该在 20ms 之后清零此控制位。
SUSPEND	Bit 1	W	**挂起模式设置** 0：正常工作模式 1：进入挂起模式 此控制位在主机模式下写 1 有效。
EN_SPDM	Bit 0	R/W	**挂起模式使能位** 0：禁止 1：使能

◆ 设备模式：

设备电源控制寄存器（USB_POWER）							
偏移地址：001_H							
复位值：00100000_B							
7	6	5	4	3	2	1	0
ISOUPDATA	SOFTCONN	HS_EN	HS_M	RESET	RESUME	SUSPEND	EN_SPDM

ISOUPDATA	Bit 7	R/W	**同步端点更新控制位** 0：当信号“TXPKTRDY”信号置位后，发送数据包之前无需等待 SOF 令牌包； 1：当信号“TXPKTRDY”信号置位后，要完成 SOF 令牌包的接收才能发送数据包。如果在收到 SOF 令牌之前收到了 IN 令牌包，控制器将会发送数据长度为 0 的数据包。 此控制位仅对同步传输有效
SOFTCONN	Bit 6	R/W	**软连接控制位** 0：断开 1：连接
HS_EN	Bit 5	R/W	**高速传输使能位** 0：禁止 1：使能 当处于使能状态时，在 USB 复位期间会进行高速传输模式的识别和握手处理。
HS_M	Bit 4	R	**高速传输模式标志位** 0：非高速传输模式 1：高速传输模式 在设备模式时，当 USB 复位完成（可通过 USB 复位中断标志判断）之后，此标志位有效。
RESET	Bit 3	R	**复位信号** 0：D+/D-无复位信号产生 1：D+/D-有复位信号产生
RESUME	Bit 2	R/W	**唤醒控制位** 0：不产生唤醒信号 1：产生唤醒信号 在设备模式时，置 1 之后应该在 10ms（最大 15ms）之后清零此控制位。
SUSPEND	Bit 1	R	**挂起模式标志位** 0：处于非挂起状态 1：处于挂起状态 当软件读取中断寄存器或者设置唤醒控制位“RESUME”时，清零此控制位
EN_SPDM	Bit 0	R/W	**挂起模式使能位**

			0：禁止 1：使能

29. 5. 2. 3 端点发送中断寄存器（USB_TXIS）

USB_TXIS 是一个 8 位的只读寄存器，用于指示端点 0 和发送端点 1~5 的哪个中断是有效的。寄存器中每一位的含意取决于设备的模式。EP1 到 EP5 位总是指示 USB 控制器正在发送数据；在主机模式，这些位适用于输出端点；而在设备模式，这些位适用于输入端点。EP0 位比较特殊，在主机和设备模式中，该位指示一个控制输入或控制输出端点产生了中断。

与这些端点相关的位如果没有配置，其返回值一直为 0。对该寄存器的读操作，会清除所有已激活的中断。

- 主机模式/设备模式：

端点发送中断寄存器（USB_TXIS）							
偏移地址：002_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved	Reserved	EP5TX_IF	EP4TX_IF	EP3TX_IF	EP2TX_IF	EP1TX_IF	EP0_IF

Reserved	Bit 7~6	-	**保留**
EP5TX_IF	Bit 5	R	**发送端点 5 中断标志位** 0：无中断 1：产生中断 读中断标志位时此位自动清零
EP4TX_IF	Bit 4	R	**发送端点 4 中断标志位** 0：无中断 1：产生中断 读中断标志位时此位自动清零
EP3TX_IF	Bit 3	R	**发送端点 3 中断标志位** 0：无中断 1：产生中断 读中断标志位时此位自动清零
EP2TX_IF	Bit 2	R	**发送端点 2 中断标志位** 0：无中断 1：产生中断 读中断标志位时此位自动清零
EP1TX_IF	Bit 1	R	**发送端点 1 中断标志位** 0：无中断 1：产生中断 读中断标志位时此位自动清零
EP0_IF	Bit 0	R	**端点 0 中断标志位** 0：无中断 1：产生中断 读中断标志位时此位自动清零

29. 5. 2. 4 端点接收中断寄存器（USB_RXIS）

USB_RXIS 是一个 8 位的只读寄存器，用于指示接收端点 1~5 的哪个中断是有效的。
与这些端点相关的位如果没有配置，其返回值一直为 0。对该寄存器的读操作，会清除所有已激活的中断。

- 主机模式/设备模式：

端点接收中断寄存器（USB_RXIS）							
偏移地址：004_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved	Reserved	EP5RX_IF	EP4RX_IF	EP3RX_IF	EP2RX_IF	EP1RX_IF	Reserved

Reserved	Bit 7~6	-	**保留**
EP5RX_IF	Bit 5	R	**接收端点 5 中断标志位** 0：无中断 1：产生中断 读中断标志位时此位自动清零
EP4RX_IF	Bit 4	R	**接收端点 4 中断标志位** 0：无中断 1：产生中断 读中断标志位时此位自动清零
EP3RX_IF	Bit 3	R	**接收端点 3 中断标志位** 0：无中断 1：产生中断
EP2RX_IF	Bit 2	R	**接收端点 2 中断标志位** 0：无中断 1：产生中断 读中断标志位时此位自动清零
EP1RX_IF	Bit 1	R	**接收端点 1 中断标志位** 0：无中断 1：产生中断 读中断标志位时此位自动清零
Reserved	Bit 0	-	**保留**

29. 5. 2. 5 端点发送中断使能寄存器（USB_TXIE）

USB_TXIE 是一个 8 位寄存器，用于为 USB_TXIS 寄存器中的中断配置中断使能位。复位时，所有的发送中断是启用的。

- 主机模式/设备模式：

端点发送中断使能寄存器（USB_TXIE）							
偏移地址：006_H							
复位值：11111111_B							
7	6	5	4	3	2	1	0
Reserved	Reserved	EP5TX_IE	EP4TX_IE	EP3TX_IE	EP2TX_IE	EP1TX_IE	EP0_IE

Reserved	Bit 7~6	-	保留
EP5TX_IE	Bit 5	R/W	**发送端点 5 中断使能位** 0：禁止 1：使能
EP4TX_IE	Bit 4	R/W	**发送端点 4 中断使能位** 0：禁止 1：使能
EP3TX_IE	Bit 3	R/W	**发送端点 3 中断使能位** 0：禁止 1：使能
EP2TX_IE	Bit 2	R/W	**发送端点 2 中断使能位** 0：禁止 1：使能
EP1TX_IE	Bit 1	R/W	**发送端点 1 中断使能位** 0：禁止 1：使能
EP0_IE	Bit 0	R/W	**端点 0 中断使能位** 0：禁止 1：使能

29. 5. 2. 6 端点接收中断使能寄存器（USB_RXIE）

USB_RXIE 是一个 8 位寄存器，用于为 USB_RXIS 寄存器的中断提供中断使能位。复位时，所有的接收中断是启用的。

- ◆ 主机模式/设备模式：

端点接收中断使能寄存器（USB_RXIE）							
偏移地址：008_H							
复位值：11111110_B							
7	6	5	4	3	2	1	0
Reserved	Reserved	EP5RX_IE	EP4RX_IE	EP3RX_IE	EP2RX_IE	EP1RX_IE	Reserved

Reserved	Bit 7~6	R/W	保留
EP5RX_IE	Bit 5	R/W	**接收端点 5 中断使能位** 0：禁止 1：使能
EP4RX_IE	Bit 4	R/W	**接收端点 4 中断使能位** 0：禁止 1：使能
EP3RX_IE	Bit 3	R/W	**接收端点 3 中断使能位** 0：禁止 1：使能
EP2RX_IE	Bit 2	R/W	**接收端点 2 中断使能位** 0：禁止 1：使能
EP1RX_IE	Bit 1	R/W	**接收端点 1 中断使能位** 0：禁止 1：使能
Reserved	Bit 0	-	保留

29. 5. 2. 7 通用USB中断寄存器（USB_USBIS）

USB_USBIS 是一个 8 位只读寄存器，用来指示哪个 USB 中断有效。读此寄存器，所有的中断将被清除。

- 主机模式：

通用 USB 中断寄存器（USB_USBIS）							
偏移地址：$00A_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
VBUSERR_IF	SESREQ_IF	DISCON_IF	CONN_IF	SOF_IF	BABBLE_IF	RESUME_IF	Reserved

VBUSERR_IF	Bit 7	R	**VBUS 错误中断标志位** 0：未产生 VBUS 错误中断 1：产生 VBUS 错误中断 读中断标志位时此位自动清零
SESREQ_IF	Bit 6	R	**会话请求中断标志位** 0：未产生会话请求中断 1：产生会话请求中断 读中断标志位时此位自动清零
DISCON_IF	Bit 5	R	**会话断开中断标志位** 0：未产生会话断开中断 1：产生会话断开中断 读中断标志位时此位自动清零
CONN_IF	Bit 4	R	**会话连接中断标志位** 0：未产生会话连接中断 1：产生会话连接中断 读中断标志位时此位自动清零
SOF_IF	Bit 3	R	**起始帧中断标志位** 0：未产生起始帧中断 1：产生起始帧中断 读中断标志位时此位自动清零
BABBLE_IF	Bit 2	R	**超时干扰中断标志位** 0：未产生超时干扰中断 1：产生超时干扰中断。 读中断标志位时此位自动清零
RESUME_IF	Bit 1	R	**唤醒信号中断标志位** 0：未产生唤醒中断 1：产生唤醒中断（挂起模式下） 读中断标志位时此位自动清零
Reserved	Bit 0	-	**保留**

◆ 设备模式：

通用 USB 中断寄存器（USB_USBIS）							
偏移地址：$00A_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved		DISCON_IF	Reserved	SOF_IF	RESET_IF	RESUME_IF	SUSPEND_IF

Reserved	Bit 7-6	-	**保留**
DISCON_IF	Bit 5	R	**会话断开中断标志位** 0：未产生会话断开中断 1：产生会话断开中断（设备已从主机断开） 读中断标志位时此位自动清零
Reserved	Bit 4	-	**保留**
SOF_IF	Bit 3	R	**起始帧中断标志位** 0：未产生起始帧中断 1：产生起始帧中断 读中断标志位时此位自动清零
RESET_IF	Bit 2	R	**复位信号中断标志位** 0：未产生复位信号中断 1：产生复位信号中断。 读中断标志位时此位自动清零
RESUME_IF	Bit 1	R	**唤醒信号中断标志位** 0：未产生唤醒信号中断 1：产生唤醒信号中断（挂起模式下） 读中断标志位时此位自动清零
SUSPEND_IF	Bit 0	R	**挂起信号中断标志位** 0：未产生挂起信号中断 1：产生挂起信号中断 读中断标志位时此位自动清零

29. 5. 2. 8 通用USB中断使能寄存器（USB_USBIE）

USB_USBIE 是一个 8 位寄存器，为 USB_USBIS 寄存器中的每个中断提供中断使能位。设备模式中，复位时中断 1 和中断 2 是使能的。

- 主机模式：

通用 USB 中断使能寄存器（USB_USBIE）							
偏移地址：$00B_H$							
复位值：00000110_B							
7	6	5	4	3	2	1	0
VBUSERR_IE	SESREQ_IE	DISCON_IE	CONN_IE	SOF_IE	BABBLE_IE	RESUME_IE	Reserved

VBUSERR_IE	Bit 7	R/W	**VBUS 错误中断使能位** 0：禁止 1：使能
SESREQ_IE	Bit 6	R/W	**会话请求信号中断使能位** 0：禁止 1：使能
DISCON_IE	Bit 5	R/W	**设备断开连接中断使能位** 0：禁止 1：使能
CONN_IE	Bit 4	R/W	**设备连接中断使能位** 0：禁止 1：使能
SOF_IE	Bit 3	R/W	**起始帧产生中断使能位** 0：禁止 1：使能
BABBLE_IE	Bit 2	R/W	**超时干扰产生中断使能位** 0：禁止 1：使能
RESUME_IE	Bit 1	R/W	**唤醒信号产生中断使能位** 0：禁止 1：使能
Reserved	Bit 0	-	**保留**

◆ 设备模式：

通用 USB 中断使能寄存器（USB_USBIE）							
偏移地址：$00B_H$							
复位值：00000110_B							
7	6	5	4	3	2	1	0
Reserved		DISCON_IE	Reserved	SOF_IE	RESET_IE	RESUME_IE	SUSPEND_IE

Reserved	Bit 7-6	-	保留
DISCON_IE	Bit 5	R/W	设备断开连接中断使能位 0：禁止 1：使能
Reserved	Bit 4	-	保留
SOF_IE	Bit 3	R/W	起始帧产生中断使能位 0：禁止 1：使能
RESET_IE	Bit 2	R/W	复位信号产生中断使能位 0：禁止 1：使能
RESUME_IE	Bit 1	R/W	唤醒信号产生中断使能位 0：禁止 1：使能
SUSPEND_IE	Bit 0	R/W	挂起信号产生中断使能位 0：禁止 1：使能

29. 5. 2. 9 USB帧值寄存器（USB_FRAME）

USB_FRAME 是一个 8 位寄存器，用于保存接收到最后帧编号。

- 主机模式/设备模式

USB 帧值寄存器（USB_FRAME）															
偏移地址：$00C_H$															
复位值：00000000_B															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					FRAME										

Reserved	Bit 15-11	-	**保留**
FRAME	Bit 10-0	R	**帧编号** 接收到的最后帧编号

29. 5. 2. 10 端点索引寄存器（USB_INDEX）

USB_INDEX 是一个 8 位寄存器。当读写 USB 控制器采用索引方式时，此寄存器设置要访问的端点号。

- 主机模式/设备模式

端点索引寄存器（USB_INDEX）							
偏移地址：$00E_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved					EP_INDEX		

Reserved	Bit 7-3	-	**保留**
EP_INDEX	Bit 2-0	R/W	**端点号** 索引端点号

29. 5. 2. 11 USB测试模式寄存器（USB_TEST）

USB_TEST 是一个 8 位寄存器。执行 TESTMODE 命令后，USB 控制器根据该寄存器的值选择“USB 2.0 规范”中所描述的四种测试模式中的一种模式。此寄存器不用于正常操作。

- 主机模式/设备模式

USB 测试模式寄存器（USB_TEST）							
偏移地址：$00F_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
FORCE_HST	FIFO_ACS	FORCE_FS	FORCE_HS	TEST_PKT	TEST_K	TEST_J	TEST_SENA

FORCE_HST	Bit 7	R/W	**强制进入主机模式控制位** 0：非主机模式 1：强制进入主机模式 此控制位置 1 同时控制位 SESSION（DEVCTL<0>）置 1 时强制进入主机模式，此模式有效时，主机断开状态信号是通过控制位“BDEVICE”（DEVCTL<7>）表示。 操作速度是通过 FORCE_FS 和 FORCE_HS 决定的：

FORCE_HS	FORCE_FS	Oper Speed
0	0	LS
0	1	FS
1	0	HS
1	1	Undefined

FIFO_ACS	Bit 6	W	**FIFO 测试控制位** 0：FIFO 访问测试无效 1：FIFO 访问测试有效 此测试模式有效时，USB 控制器会将 Endpoint0 TX FIFO 的数据传送到 Endpoint0 RX FIFO。此控制位会自动清零
FORCE_FS	Bit 5	R/W	**全速模式测试控制位** 0：测试无效 1：测试有效 此控制位需要与控制位 FORCE_HST 同时置 1 或者当收到 USB 复位信号时强制 USB 控制器进入全速模式
FORCE_HS	Bit 4	R/W	**高速模式测试控制位** 0：测试无效 1：测试有效 此控制位需要与控制位 FORCE_HST 同时置 1 或者当收到 USB 复位信号时强制 USB 控制器进入高速模式
TEST_PKT	Bit 3	R/W	**数据包测试模式控制位（高速传输模式）** 0：数据包测试模式无效

			1：数据包测试模式有效 此测试模式有效时，USB 控制器发送长度为 53 字节测试数据包（测试数据包的格式是固定的，在进入此测试模式之前，需要先将测试数据包加载到 Endpoint0 FIFO 中。）
TEST_K	Bit 2	R/W	**K 状态测试模式控制位（高速传输模式）** 0：K 状态测试无效 1：K 状态测试有效 此测试模式有效时，USB 控制器连续的发送 K 状态信号
TEST_J	Bit 1	R/W	**J 状态测试模式控制位（高速传输模式）** 0：J 状态测试无效 1：J 状态测试有效 此测试模式有效时，USB 控制器连续的发送 J 状态信号
TEST_SENA	Bit 0	R/W	**SE0_NAK 测试模式控制位（高速传输模式）** 0：SE0_NAK 测试无效 1：SE0_NAK 测试有效 此测试模式有效时，USB 控制器应该处于高速传输模式，对于任何有效的 IN 令牌包都仅回复 NAK 握手包。

29. 5. 2. 12 索引端点 0 控制和状态低字节寄存器（USB_IND_CSR0L）

USB_IND_CSR0L 是一个 8 位寄存器，是索引端点 0 控制和状态的低字节寄存器。

◆ 主机模式

索引端点 0 控制和状态低字节寄存器（USB_IND_CSR0L）							
偏移地址：012_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
NAKTOUT	STATUSPKT	REQPKT	ERROR	SETUPPKT	RXSTALL	TXPKTRDY	RXPKTRDY

NAKTOUT	Bit 7	R/W	**NAK 超时标志位** 当端点 0 收到 NAK 握手包的时间大于寄存器 USB_IND_NAK 设置值时此标志位被 USB 控制器置 1。当此标志位置 1 时，应通过软件写 0 清零此标志位让端点 0 继续进行事务处理。
STATUSPKT	Bit 6	R/W	**Status stage 事务处理控制位** 控制位 TXPKTRDY 或 REQPKT 置 1 的同时软件置 1 此控制位将会执行状态阶段事务处理。在置 1 此控制的同时要确保 DTOG 置 1 即采用 DATA1 奇数包进行状态阶段事务处理。
REQPKT	Bit 5	R/W	**IN 事务处理请求控制位** 0：无 IN 事务处理请求 1：请求 IN 事务处理 当 RXPKTRDY 置 1 时，会自动清零此控制位。
ERROR	Bit 4	R/W	**传输错误标志位** USB 控制器对某一事务处理重试 3 次，USB 设备无任何应答时，此标志位置 1 表明事务处理产生错误。软件写 0 清零此标志位，写 1 无效。
SETUPPKT	Bit 3	R/W	**SETUP 令牌包发送控制位** 软件置 1 控制位 TXPKTRDY 的同时置 1 此控制位，将会以 SETUP 令牌包事务处理取代 OUT 令牌包事务处理。软件置 1 此控制位的同时会清零 DTOG 控制位。
RXSTALL	Bit 2	R/W	**接收 STALL 握手信号** 当完成 STALL 握手信号接收时，USB 控制器自动置 1 此控制位。此控制位仅可通过软件写 0 清，软件写 1 无效。
TXPKTRDY	Bit 1	R/W	**发送数据包 Ready 标志位** 0：端点 0 数据包加载未完成 1：端点 0 数据包加载完成（软件写 1 有效，写 0 无效） 当软件完成端点 0 数据包加载时，软件需要将此控制位置 1。当完成数据包发送时，此控制位自动清零，同时端点 0 产生中断（中断使能时）。

RXPKTRDY	Bit 0	R/W	**数据包接收完成标志位** 0：数据包接收未完成 1：数据包接收完成 当此控制位置 1 时，如果中断使能，端点 0 会产生中断。 软件写 0 清零此控制位（软件写 1 无效）。

◆ 设备模式

索引端点 0 控制和状态低字节寄存器（USB_IND_CSR0L）							
偏移地址：012_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
SSTPEND	SRXPKTRDY	SENDSTALL	SETUPEND	DATAEND	SENTSTALL	TXPKTRDY	RXPKTRDY

SSTPEND	Bit 7	W	**ServicedSetupEnd 清零控制位** 软件置 1 此位，清零此标志位（软件写 0 无效）
SRXPKTRDY	Bit 6	W	**ServicedRxPktRdy 清零控制位** 软件置 1 此控制位，清零此标志位（软件写 0 无效）
SENDSTALL	Bit 5	W	**发送 STALL 控制位** 软件置 1 此控制位时会中止当前事务处理，并且发送 STALL 握手包。此控制位自动清零（软件写 0 无效）。
SETUPEND	Bit 4	R	**SETUP 事务处理完成标志位** 0：控制事务处理未完成 1：控制事务处理完成 在 DATAEND 控制位置 1 之前完成控制事务处理此控制位置 1，同时会产生相应的中断和 FIFO Flush 操作。控制位 SSTPEND 写 1 清零此标志位。
DATAEND	Bit 3	W	**数据完成控制位** 当产生如下情况时，软件需要将此位置 1（软件写 0 无效）： 1）完成最后一包数据加载置位 TXPKTRDY； 2）完成最后一包数据读取清零 RXPKTRDY； 3）发送长度为 0 的数据包置位 TXPKTRDY； 此控制位自动清零。
SENTSTALL	Bit 2	R/W	**发送 STALL 握手信号** 当完成 STALL 握手信号传输时，USB 控制器置 1 此控制位。此控制位仅可通过软件写 0 清，软件写 1 无效。
TXPKTRDY	Bit 1	R/W	**发送数据包 Ready 标志位** 0：端点 0 数据包加载未完成 1：端点 0 数据包加载完成（软件写 1 有效，写 0 无效） 当软件完成端点 0 数据包加载时，软件需要将此控制位置 1。当完成数据包发送时，此控制位自动清零，同时端点 0

			产生中断（中断使能时）。
RXPKTRDY	Bit 0	R	**数据包接收完成标志位** 0：数据包接收未完成 1：数据包接收完成 当此控制位置 1 时，如果中断使能端点 0 会产生中断。软件置 1 控制位 SRXPKTRDY 时清零此控制位。

29. 5. 2. 13 索引端点 0 控制和状态高字节寄存器（USB_IND_CSR0H）

USB_IND_CSR0H 是一个 8 位寄存器，提供对索引端点 0 的控制和状态获取的高字节寄存器。

◆ 主机模式

索引端点 0 控制和状态高字节寄存器（USB_IND_CSR0H）							
偏移地址：013_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved				DISPING	DTOG_WN	DTOG	FLUSHFIFO

Reserved	Bit 7~4	-	**保留**
DISPING	Bit 3	R/W	**PING 令牌包禁止控制位** 0：高速控制传输使能 PING 事务处理 1：高速控制传输禁止 PING 事务处理
DTOG_WN	Bit 2	W	**数据触发位写使能** 软件置 1 此控制位时，端点 0 的 DTOG 数据触发位可写。当新的值写入 DTOG 数据触发位后，此控制位自动清零。
DTOG	Bit 1	R/W	**数据触发位** 0：数据触发位为 0 1：数据触发位为 1 仅当 DTOG_WN 置 1 时，才可以写此控制位，否则只可读不可写。
FLUSHFIFO	Bit 0	W	**FIFO Flush 控制位** 软件置 1 此控制位时，会触发从端点 0 FIFO 中发送或读取下一包数据，同时复位 FIFO 指针、清零 TXPKTRDY/RXPKTRDY 标志位。注意，仅当 TXPKTRDY/RXPKTRDY 为 1 时才可使用此控制位，否则可能会引起数据被破坏。 此控制位自动清零（软件写 0 无效）

◆ 设备模式

索引端点 0 控制和状态高字节寄存器（USB_IND_CSR0H）							
偏移地址：013_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved							FLUSHFIFO

Reserved	Bit 7-1	-	保留
FLUSHFIFO	Bit 0	W	**FIFO Flush 控制位** 软件置 1 此控制位时，会触发从端点 0 FIFO 中发送或读取下一包数据，同时复位 FIFO 指针清零 TXPKTRDY/RXPKTRDY 标志位。仅当 TXPKTRDY/RXPKTRDY 为 1 时才可使用此控制位，否则可能会引起数据被破坏。 此控制位自动清零（软件写 0 无效）

29. 5. 2. 14 索引端点 0 接收字节数量寄存器（USB_IND_COUNT0）

USB_IND_COUNT0 是一个 8 位只读寄存器，用来指示索引端点 0 的 FIFO 中收到数据的字节数量。此值伴随 FIFO 中内容变化而变化，只有 RXPKTRDY 位置 1 时有效。

◆ 主机模式/设备模式

索引端点 0 接收字节数量寄存器（USB_IND_COUNT0）							
偏移地址：018_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved	COUNT						

Reserved	Bit 7	-	保留
COUNT	Bit 6-0	R	**指示端点 0 的 FIFO 中收到的数据的字节数量。** 此值伴随 FIFO 中内容变化而变化。

29. 5. 2. 15 索引端点 0 主机发送配置寄存器（USB_IND_TYPE0）

USB_IND_TYPE0 是一个 8 位寄存器，用于存储索引端点 0 通讯的目标设备的运行速度。

◆ 主机模式

索引端点 0 主机发送配置寄存器（USB_IND_TYPE0）							
偏移地址：$01A_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
SPEED		Reserved					

SPEED	Bit 7-6	R/W	目标设备的运行速度控制位 00：NC 01：高速 10：全速 11：低速
Reserved	Bit 5-0	-	保留

29. 5. 2. 16 索引端点 0 无应答超时时间设置寄存器（USB_IND_NAK）

USB_IND_NAK 是一个 8 位寄存器，用于设置在接收到一连串的 NAK 响应时，索引端点 0 应设置多少个帧的超时。

所选的帧数为 $2^{(m-1)}$（其中，m 是该寄存器中设定的值，其有效数值为 2-16）。如果主机从目标设备接收的 NAK 大于此寄存器中代表的限度值，端点将暂停。

注意：寄存器值为 0 或 1 时将禁止 NAK 超时功能

- 主机模式

索引端点 0 无应答超时时间设置寄存器（USB_IND_NAK）							
偏移地址：$01B_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved			NAKLIMIT				

Reserved	Bit 7-5	-	保留
NAKLIMIT	Bit 4-0	R/W	**NAK** 延时帧数

29. 5. 2. 17 索引端点 0 的控制和状态高字节寄存器（USB_IND_CONFIG0）

USB_IND_CONFIG0 是一个 8 位只读寄存器，存储索引端点配置为端点 0 时的 USB 内核配置信息。

- 主机模式/设备模式

索引端点 0 的控制和状态高字节寄存器（USB_IND_CONFIG0）							
偏移地址：$01F_H$							
复位值：11011010_B							
7	6	5	4	3	2	1	0
MPRXE	MPTXE	BIGENDIAN	HBRXE	HBTXE	FIFOSIZE	SOFTCONE	UTMI DATAWIDTH

MPRXE	Bit 7	R	MPRXE 位始终为 1，表示批量传输模式下支持接收数据包拆分。
MPTXE	Bit 6	R	MPTXE 位始终为 1，表示批量传输模式下支持发送数据包拆分。
BIGENDIAN	Bit 5	R	BIGENDIAN 位始终为 0，表示数据为小端模式。
HBRXE	Bit 4	R	HBRXE 位始终为 1，表示使能宽带接收端点。
HBTXE	Bit 3	R	HBTXE 位始终为 1，表示使能宽带发送端点。
FIFOSIZE	Bit 2	R	FIFOSIZE 位始终为 0，表示 FIFO 大小固定。
SOFTCONE	Bit 1	R	SOFTCONE 位始终为 1，表示软件连接和断开。
UTMI DATAWIDTH	Bit 0	R	DATAWIDTH 位始终为 0，表示数据位宽为 8bit。

29. 5. 2. 18 索引端点发送控制和状态低字节寄存器（USB_IND_TXCSRL）

USB_IND_TXCSRL 是一个 8 位寄存器，是索引端点发送控制和状态低字节寄存器。

◆ 主机模式

索引端点发送控制和状态低字节寄存器（USB_IND_TXCSRL）							
偏移地址：012_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
NAKTOUT	CLRDATAT	RXSTALL	SETUPPKT	FLUSHFIFO	ERROR	FIFONE	TXPKTRDY

NAKTOUT	Bit 7	R/W	**NAK 超时** 0：无超时 1：只用于批量传输端点接收 NAK 的次数超过了寄存器 USB_IND_NAK 设置的次数。软件需要清除此位来允许端点继续传输。 只用于宽带中断端点如果没有收到已发送数据包的应答，此位会被置 1。
CLRDATAT	Bit 6	R/W	**清除数据转换** 0：清除数据转换 1：向此位写 1 可将寄存器 USB_IND_TXCSRH 的 DATAT 位清零。
RXSTALL	Bit 5	R/W	**端点挂起** 0：没有接收到 STALL 握手信号 1：收到 STALL 握手信号。此位需要软件清零。
SETUPPKT	Bit 4	R/W	**建立令牌包** 0：不发送建立令牌包 1：为此传输事务发送建立令牌包而不是输出令牌包。在置 1 TXPKTRDY 位的同时，也应将此位置位。 向此位写 1 会将寄存器 USB_IND_TXCSRH 的 FDATAT 位清零。
FLUSHFIFO	Bit 3	R/W	**清空 FIFO** 0：无影响 1：清空发送端点 FIFO 中最后的包。同时，FIFO 指针复位，TXPKTRDY 位清零。此时 USB_TXIS 寄存器中的 EPnTX_IF 位也要置位。 此位与 TXPKTRDY 位同时置位，将放弃当前正在装载到 FIFO 的数据包。仅当 TXPKTRDY 为 1 时才可置 1 此控制位，否则可能会引起数据被破坏。当端点为双缓存 FIFO 时，应该置 1 此控制位两次，完成 FIFO 中全部数据清除。
ERROR	Bit 2	R/W	**错误：** 0：无错误

			1：尝试三次发送包，都未接收到握手包。此时 ERROR 位会被自动置 1，TXPKTRDY 位清零，FIFO 完全清空。 注意，此位仅在批量传输和中断传输模式下有效。
FIFONE	Bit 1	R/W	**FIFO 不空** 0：FIFO 空 1：FIFO 不空
TXPKTRDY	Bit 0	R/W	**发送包准备好** 0：未准备好发送包。 1：在装载数据包到发送 FIFO 中后，软件置 1 此位。 当数据包发送完成时，此位自动清零，并产生中断。向双缓存的 FIFO 中装载第二个数据包时，TXPKTRDY 位也将自动清零。

◆ 设备模式

索引端点发送控制和状态低字节寄存器（USB_IND_TXCSRL）							
偏移地址：012_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
INCOMPTX	CLRDATAT	SENTSTALL	SENDSTALL	FLUSHFIFO	UNDERRUN	FIFONE	TXPKTRDY

INCOMPTX	Bit 7	R/W	**数据包发送不完全** 0：数据包完全发送 1：数据包不完全发送 当发送端点进行宽带同步传输时，大的数据包会被分割成 2 或 3 个数据包进行传输，但是当收到的 IN 令牌包不足发送完所有数据包时，此标志位被置 1 表明数据发送不完全。 软件写 0 清零此标志位（软件写 1 无效）
CLRDATAT	Bit 6	W	**数据转换清除** 软件置 1 此控制位，清零端点 DTOG（软件写 0 无效）
SENTSTALL	Bit 5	R/W	**发送 STALL** 0：未发送 STALL 握手包 1：发送 STALL 握手包 当发送 STALL 握手包后 USB 控制器置 1 此控制位，同时会执行 FIFO Flush 和清零 TXPKTRDY。软件写 0 清零此标志位（软件写 1 无效）。
SENDSTALL	Bit 4	R/W	**发送 STALL** 对于 IN 令牌包，软件置 1 此控制位会触发 STALL 握手包发送，软件清 0 此控制位终止 STALL 握手包。 此控制位对同步传输无效。
FLUSHFIFO	Bit 3	W	**清空 FIFO**

			软件置 1 此控制位时，会触发从端点 FIFO 中发送下一包数据，同时复位 FIFO 指针、清零 TXPKTRDY 标志位。 仅当 TXPKTRDY 为 1 时才可置 1 此控制位，否则可能会引起数据被破坏。当端点为双缓存 FIFO 时，应该置 1 此控制位两次，完成 FIFO 中全部数据清除。 此控制位自动清零（软件写 0 无效）
UNDERRUN	Bit 2	R/W	**下溢** 当 TXPKTRDY 为 0 时，收到 IN 令牌包，USB 控制器置 1 此标志位。当软件检测到此标志位为 0 时应该写 0 清零此标志位（软件写 1 无效）
FIFONE	Bit 1	R/W	**FIFO 不空** 0：发送 FIFO 空 1：发送 FIFO 非空 当发送 FIFO 中至少存在一包数据时，USB 控制器置 1 此标志位
TXPKTRDY	Bit 0	R/W	**数据包准备好** 0：数据包加载未完成 1：数据包加载完成（软件写 1 有效，写 0 无效） 当软件完成端点 FIFO 数据包加载时，软件需要将此控制位置 1。当完成数据包发送时，此控制位自动清零，同时端点产生中断（中断使能时）。当端点 FIFO 为双缓存 FIFO 时，在加载第二个数据包到 FIFO 时 TXPKTRDY 会自动清零。

29. 5. 2. 19 索引端点发送控制和状态高字节寄存器（USB_IND_TXCSRH）

USB_IND_TXCSRH 是一个 8 位寄存器，是索引端点发送控制和状态高字节寄存器。

◆ 主机模式

索引端点发送控制和状态高字节寄存器（USB_IND_TXCSRH）							
偏移地址：013_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
AUTOSET	Reserved	MODE	DMAREQENAB	FDATAT	DMAREQMODE	DATATWE	DATAT

AUTOSET	Bit 7	R/W	**自动置位** 当软件将此位置 1 时，当最大包长中（USB_IND_TXMAXP 中的值）的数据装载到发送 FIFO 时，TXPKTRDY 位可自动置 1。如果装载的数据包小于最大包大小，TXPKTRDY 位必须手动置 1。 注意：宽带同步传输端点和宽带中断传输端点时，不能将 AUTOSET 位置 1。
Reserved	Bit 6	-	**保留**
MODE	Bit 5	R/W	**端点方向设置** 0：设置端点方向为接收。 1：设置端点方向为发送。 注意：此位只在发送和接收传输使用相同端点 FIFO 时起作用。
DMAREQENAB	Bit 4	-	软件设置该位来使能 TX 端点的 DMA 请求。
FDATAT	Bit 3	R/W	**强制数据切换** 0：无影响 1：软件将此位置 1 时，无论是否收到 ACK 确认，强制端点切换 DATAT 位，清空 FIFO 中的数据包。中断传输端点可以将此位用于反馈同步端点的通信速率。
DMAREQMODE	Bit 2	-	**软件置位该位来选择 DMA 请求模式 1，清除该位来选择 DMA 请求模式 0。** 注：该位不能在上述 DMAREQENAB 位被清零之前或者同一周期内清零。
DATATWE	Bit 1	W	**数据切换启用** 0：不能对 DATAT 位进行写操作。 1：使能对 DATAT 位进行写操作。 一旦写入新值，此位自动清零。
DATAT	Bit 0	R/W	**数据切换** 此位指示发送端点数据切换的当前状态。如果 DATATWE 位为 1，此位可根据数据切换的需求设置写值。如果

			DATATWE 位为 0，写到该位的任何值都将被忽略。

◆ 设备模式

索引端点发送控制和状态高字节寄存器（USB_IND_TXCSRH）							
偏移地址：013_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
AUTOSET	ISO	MODE	Reserved	FDATAT	Reserved		

AUTOSET	Bit 7	R/W	**自动置位** 当软件将此位置 1 时，当最大包长中（USB_IND_TXMAXP 中的值）的数据装载到发送 FIFO 时，TXPKTRDY 位可自动置 1。如果装载的数据包小于最大包大小，TXPKTRDY 位必须手动置 1。 注意：宽带同步传输端点和宽带中断传输端点时，不能将 AUTOSET 位置 1。
ISO	Bit 6	R/W	**同步传输** 0：启用发送端点的批量或中断传输。 1：启用发送端点的同步传输。
MODE	Bit 5	R/W	**端点方向设置** 0：设置端点方向为接收。 1：设置端点方向为发送。 注意：此位只在发送和接收传输使用相同端点 FIFO 时起作用。
Reserved	Bit 4	-	**Reserved**
FDATAT	Bit 3	R/W	**强制数据切换** 0：无影响 1：软件将此位置 1 时，无论是否收到 ACK 确认，强制端点切换 DATAT 位，清空 FIFO 中的数据包。中断传输端点可以将此位用于反馈同步端点的通信速率。
Reserved	Bit 2-0	-	**保留**

29. 5. 2. 20 索引端点发送最大传输数据寄存器（USB_IND_TXMAXP）

USB_IND_TXMAXP 是一个 16 位寄存器，定义了通过索引发送端点单次可传输的最大数据的长度。该长度最大为 1024 字节，但必须服从“USB 规范”里批量传输、中断传输和全速模式下同步传输的包长限制。

写入寄存器中的值代表的数据总大小不能超过发送端点的 FIFO 大小，如果支持双包缓存，不能超过 FIFO 大小的一半。

如果此寄存器在端点发送包后发生改变，在寄存器写入新值之前，发送端点 FIFO 必须完全被清空（使用 USB_IND_TXCSRL 寄存器中的 FLUSHFIFO 位）。

此寄存器仅适用于索引端点 1~5。

- 主机模式/设备模式

索引端点发送最大传输数据寄存器（USB_IND_TXMAXP）															
偏移地址：010_H															
复位值：$00000000\_00000000_B$															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					MAXLOAD										

Reserved	Bit 15-11	-	保留
MAXLOAD	Bit 10-0	R/W	发送端点单次可传输的最大数据的长度

29. 5. 2. 21 索引端点接收最大传输数据寄存器（USB_IND_RXMAXP）

USB_IND_RXMAXP 是一个 16 位寄存器，定义了通过接收端点单次可传输的最大数据的长度。该长度最大为 1024 字节，但必须服从“USB 规范”里传输的包长限制。

写入寄存器中的值代表的数据总大小不能超过接收端点的 FIFO 大小，如果支持双包缓存，不能超过 FIFO 大小的一半。

此寄存器仅适用于索引端点 1~5。

- 主机模式/设备模式

索引端点接收最大传输数据寄存器（USB_IND_RXMAXP）															
偏移地址：014_H															
复位值：$00000000\_00000000_B$															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					MAXLOAD										

Reserved	Bit 15-11	-	保留
MAXLOAD	Bit 10-0	R/W	接收端点单次可传输的最大数据的长度

29. 5. 2. 22 索引端点接收控制和状态低字节寄存器（USB_IND_RXCSRL）

USB_IND_RXCSRL 是一个 8 位寄存器，是索引端点接收控制和状态低字节寄存器。

◆ 主机模式：

索引端点接收控制和状态低字节寄存器（USB_IND_RXCSRL）							
偏移地址：016_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
CLRDATAT	RXSTALL	REQPKT	FLUSHFIFO	DER_NAKO	ERROR	FIFOFULL	RXPKTRDY

CLRDATAT	Bit 7	W	**清除数据转换** 向此位写 1，可以清除数据转换。
RXSTALL	Bit 6	R/W	**端点挂起** 0：没有接收到 STALL 握手信号。 1：接收到 STALL 握手信号，此时会产生中断。 此位需要软件清零。
REQPKT	Bit 5	R/W	**请求包** 0：无请求。 1：请求输入事务。 当 RXPKTRDY 位置 1 时，此位清零。
FLUSHFIFO	Bit 4	R/W	**清空 FIFO** 0：无影响 1:清空下一个将从端点 FIFO 中读出的数据包。同时，FIFO 指针复位，RXPKTRDY 位清零。注意，如果 FIFO 是双缓存的，FLUSHFIFO 位需被置 1 两次才可清空 FIFO。 只有在 RXPKTRDY 位为 1 时，此位才能清零。否则，可能会导致数据损坏。
DER_NAKO	Bit 3	R/W	**数据错误/NAK 超时** 同步模式下此位和 RXPKTRDY 位同时为 1，表示数据包有 CRC 或位填充错误。RXPKTRDY 位清零时，此位也会清零。 批量模式下接收 NAK 响应的次数超过了 USB_RXxINTERVAL 寄存器中设置的次数，接收端点暂停。需软件清除此位允许端点继续工作。
ERROR	Bit 2	R/W	**错误：** 0：无错误 1：三次接收数据包，都未接收到数据包。此时 ERROR 位会被自动置 1。 注意，此位仅在批量传输或中断传输模式下有效。
FIFOFULL	Bit 1	R	**FIFO 满**

			0：FIFO 未满 1：FIFO 满
RXPKTRDY	Bit 0	R/W	**接收包准备好** 0：未接收到数据包。 1：接收到数据包。 如果 USB_RXxCSRH 寄存器中的 AUTOCLR 位置 1，则当 USB_RXxMAXP 字节的数据包从接收 FIFO 读出时，该位会自动清零。如果 AUTOCLR 位为零或者读出了比最大包长小的数据包，则软件必须在数据包已从接收 FIFO 中读出时将该位清零。

◆ 设备模式：

索引端点接收控制和状态低字节寄存器（USB_IND_RXCSRL）							
偏移地址：016_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
CLRDATAT	SENTSTALL	SENDSTALL	FLUSHFIFO	DATAERROR	OVERRUN	FIFOFULL	RXPKTRDY

CLRDATAT	Bit 7	W	**清除数据转换** 向此位写 1，可以清除数据转换。
SENTSTALL	Bit 6	R/W	**端点挂起** 0：未发送 STALL 握手。 1：已发送 STALL 握手。 必须通过软件清除此位。
SENDSTALL	Bit 5	R/W	**发送 STALL** 0：无影响 1：发送 STALL 握手包 注意，此位同步传输下无效。
FLUSHFIFO	Bit 4	R/W	**清空 FIFO** 0：无影响 1：清空从端点接收 FIFO 中读出的下一个数据包。同时，FIFO 指针复位，RXPKTRDY 位清零。 当端点为双缓存 FIFO 时，应该置 1 此控制位两次，完成 FIFO 中全部数据清除。
DATAERROR	Bit 3	R	**数据错误** 0：正常工作。 1：表示 RXPKTRDY 位置 1 且数据包有 CRC 或位填充错误。 注意，只有在同步模式下，此位才有效。
OVERRUN	Bit 2	R/W	**上溢**

			0：无错误。 1：表示输出数据包不能装载到接收 FIFO 中。 注意，只有在同步模式下，此位才有效。
FIFOFULL	Bit 1	R	**FIFO 满** 0：接收 FIFO 未满 1：接收 FIFO 满
RXPKTRDY	Bit 0	R/W	**接收包准备好** 0：未接收到数据包。 1：已接收到数据包。 如果 USB_RXxCSRH 寄存器中的 AUTOCLR 位置 1，则当 USB_RXxMAXP 字节的数据包从接收 FIFO 读出时，该位会自动清零。如果 AUTOCLR 位为零或者读出了比最大包长小的数据包，则软件必须在数据包已从接收 FIFO 中读出时将该位清零。

29. 5. 2. 23 索引端点接收控制和状态高字节寄存器（USB_IND_RXCSRH）

USB_IND_RXCSRH 是一个 8 位寄存器，是索引端点接收控制和状态高字节寄存器。

◆ 主机模式：

索引端点接收控制和状态高字节寄存器（USB_IND_RXCSRH）							
偏移地址：017_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
AUTOCLR	AUTOREQ	DMAREQENAB	PIDERROR	DMAREQMODE	DATATWE	DATAT	INCOMPRX

AUTOCLR	Bit 7	R/W	**自动清零** 0：不使能自动清零 1：当 USB_RXxMAXP 定义的数据包已从接收 FIFO 读出时，RXPKTRDY 位可自动清零。如果读出的包长小于最大包长时，必须手动将 RXPKTRDY 位置 1。 注意，宽带同步模式端点情况下，此位不能置 1。
AUTOREQ	Bit 6	R/W	**自动请求** 0：无自动请求。 1：当 RXPKTRDY 位清零时，REQPKT 位可自动置 1。 注意，当收到一个短包时，此位自动清零。
DMAREQENAB	Bit 5	-	**软件设置该位来使能 RX 端点的 DMA 请求。**
PIDERROR	Bit 4	R	**PID 错误** 0：无错误 1：同步传输的接收包中存在 PID 错误。 在批量或中断传输时，此位忽略。
DMAREQMODE	Bit 3	-	**软件置位该位来选择 DMA 请求模式 1，清除该位来选择 DMA 请求模式 0。**
DATATWE	Bit 2	R	**数据切换写启用** 0：不能对 DATAT 位进行写操作。 1：可以对 DATAT 位进行写操作。
DATAT	Bit 1	R	**数据切换** 此位指示端点 0 数据切换的当前状态。 如果 DATATWE 位为高电平，此位可根据数据切换的需求设置写值。如果 DATATWE 位为低电平，写到该位的任何值都将被忽略。
INCOMPRX	Bit 0	R/W	**接收包完成** 0：接收数据包完成。 1：表示在宽带同步模式或中断模式下，数据包接收未完成。当 RXPKTRDY 位被清零时，此位也会被清零。 USB 工作正常情况下，此位不会被置 1。

◆ 设备模式：

索引端点接收控制和状态高字节寄存器（USB_IND_RXCSRH）							
偏移地址：017_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
AUTOCLR	ISO	Reserved	DN_PIDERR	Reserved			INCOMPRX

AUTOCLR	Bit 7	R/W	**自动清零** 0：不使能自动清零 1：当 USB_RXxMAXP 定义的数据包已从接收 FIFO 读出时，RXPKTRDY 位可自动清零。如果读出的包长小于最大包长时，必须手动将 RXPKTRDY 位置 1。 注意，宽带同步模式端点情况下，此位不能置 1。
ISO	Bit 6	R/W	**同步传输** 0：启用接收端点的批量/中断传输。 1：启用接收端点的同步传输。
Reserved	Bit 5	-	**保留**
DN_PIDERR	Bit 4	R	**禁止 NYET/PID 错误** 此位为 1，表明对于批量或中断传输：禁止 NYET 握手的发送。当此位置 1，所有成功接收的包都将被确认，即使此时 FIFO 变满。 对于同步时传输：表明接收到的数据包存在 PID 错误。
Reserved	Bit 3-1	-	**保留**
INCOMPRX	Bit 0	R/W	**接收包完成** 0：接收数据包完成。 1：表示在宽带同步模式或中断模式下，数据包接收未完成。当 RXPKTRDY 位被清零时，此位也会被清零。 USB 工作正常情况下，此位不会被置 1。

29. 5. 2. 24 索引端点接收字节数量寄存器（USB_IND_RXCOUNT）

USB_IND_RXCOUNT 是一个 16 位寄存器，用于指示在要从接收 FIFO 读出的，当前行的数据包中保留的数据字节数。如果被发送的是多个批量数据包，则给出的数字适用于组合的数据包。
注意，返回的值随着读 FIFO 而改变，此值只在寄存器 USB_RXxCSRL 中的 RXPKTRDY 位为 1 时有效。

- ◆ 主机模式/设备模式

索引端点接收字节数量寄存器（USB_IND_RXCOUNT）															
偏移地址：018_H															
复位值：$00000000\_00000000_B$															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			COUNT												

Reserved	Bit 15-13	-	**保留**
COUNT	Bit 12-0	R	**指示索引端点的 FIFO 中收到的数据的字节数量**。 此值伴随 FIFO 中内容变化而变化。

29. 5. 2. 25 索引端点主机发送配置寄存器（USB_IND_TXTYPE）

USB_IND_TXTYPE 是一个 8 位寄存器，用来说明当前选中发送端点的目标端点号，传输协议，以及其运行速度。

- ◆ 主机模式

索引端点主机发送配置寄存器（USB_IND_TXTYPE）							
偏移地址：$01A_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
SPEED		PROTOCOL		TEN			

SPEED	Bit 7-6	R/W	**目标设备的运行速度选择** 00：目标设备假定与 USB 控制器具有相同的连接速度。 01：高速 10：全速 11：低速
PROTOCOL	Bit 5-4	R/W	**协议** 00：控制 01：同步 10：批量 11：中断 必须软件配置此二位，用来选择发送端点需要的协议
TEN	Bit 3-0	R/W	**目标端点号** 此端点号必须软件配置，包含在设备枚举过程中返回 USB 控制器的描述符中。

29. 5. 2. 26 索引端点主机发送轮询间隔寄存器（USB_IND_TXINTERVAL）

USB_IND_TXINTERVAL 是一个 8 位寄存器，为当前选中的发送端点的中断传输和同步传输定义了轮询间隔。为批量传输定义了接收到多少帧的 NAK 响应将超时。

USB_IND_TXINTERVAL 寄存器的值定义了帧数，如下表：

传输类型	速度	有效值（m）	说明
中断传输	低速或全速	1~255	轮询间隔是 m 帧
	高速	1~16	轮询间隔是 $2^{(m-1)}$ microframes
同步传输	全速或高速	1~16	轮询间隔是 $2^{(m-1)}$ 帧/microframes
批量传输	全速或高速	2~16	NAK 限制是 $2^{(m-1)}$ 帧。值为 0 或 1 时将禁止 NAK 超时功能。

◆ 主机模式

索引端点主机发送轮询间隔寄存器（USB_IND_TXINTERVAL）							
偏移地址：$01B_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
TXPOLL/NAKLMT							

TXPOLL/NAKLMT	Bit 7-0	R/W	**发送轮询间隔/NAK 限制时间** 见上表

29. 5. 2. 27 索引端点主机接收配置寄存器（USB_IND_RXTYPE）

USB_IND_RXTYPE 是一个 8 位寄存器，用来说明当前选中接收端点的目标端点号，传输协议，以及其运行速度。

- ◆ 主机模式

索引端点主机接收配置寄存器（USB_IND_RXTYPE）							
偏移地址：$01C_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
SPEED		PROTOCOL		TEN			

SPEED	Bit 7-6	R/W	**目标设备的运行速度选择** 00：目标设备假定与 USB 控制器具有相同的连接速度。 01：高速 10：全速 11：低速
PROTOCOL	Bit 5-4	R/W	**协议** 00：控制 01：同步 10：批量 11：中断 必须软件配置此二位，用来选择接收端点需要的协议
TEN	Bit 3-0	R/W	**目标端点号** 此端点号必须软件配置，包含在设备枚举过程中返回 USB 控制器的描述符中。

29. 5. 2. 28 索引端点主机接收轮询间隔寄存器（USB_IND_RXINTERVAL）

USB_IND_RXINTERVAL 是一个 8 位寄存器，为当前选中的接收端点的中断传输和同步传输定义了轮询间隔。为批量传输定义了接收到多少帧的 NAK 响应将超时。

USB_IND_RXINTERVAL 寄存器的值定义了帧数，如下表：

传输类型	速度	有效值（m）	说明
中断传输	低速或全速	1~255	轮询间隔是 m 帧
	高速	1~16	轮询间隔是 $2^{(m-1)}$ microframes
同步传输	全速或高速	1~16	轮询间隔是 $2^{(m-1)}$ 帧/microframes
批量传输	全速或高速	2~16	NAK 限制是 $2^{(m-1)}$ 帧。值为 0 或 1 时将禁止 NAK 超时功能。

◆ 主机模式

索引端点主机接收轮询间隔寄存器（USB_IND_RXINTERVAL）							
偏移地址：$01D_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
RXPOLL/NAKLMT							

RXPOLL/NAKLMT	Bit 7-0	R/W	**接收轮询间隔/NAK 限制时间** 见上表

29. 5. 2. 29　索引端点FIFO长度寄存器（USB_IND_FIFOSIZE）

USB_IND_FIFOSIZE 是一个 8 位只读寄存器，存储索引端点（非端点 0）的 TX/RX 的 FIFO 长度信息。

- ◆ 主机模式/设备模式

索引端点 FIFO 长度寄存器（USB_IND_FIFOSIZE）							
偏移地址：$01F_H$							
复位值：根据配置决定							
7	6	5	4	3	2	1	0
RXFIFOSIZE				TXFIFOSIZE			

RXFIFOSIZE	Bit 7-4	R	**RX 的 FIFO 长度** 取值范围是 0~10 RX 的 FIFO 长度是 $8*2^{RXFIFOSIZE}$ bytes
TXFIFOSIZE	Bit 3-0	R	**TX 的 FIFO 长度** 取值范围是 0~10 TX 的 FIFO 长度是 $8*2^{TXFIFOSIZE}$ bytes

29. 5. 2. 30 端点FIFO访问寄存器组（USB_FIFOx）

偏移地址	寄存器名称	复位值
0x020	USB_FIFO0	00000000_00000000_00000000_00000000$_B$
0x024	USB_FIFO1	00000000_00000000_00000000_00000000$_B$
0x028	USB_FIFO2	00000000_00000000_00000000_00000000$_B$
0x02C	USB_FIFO3	00000000_00000000_00000000_00000000$_B$
0x030	USB_FIFO4	00000000_00000000_00000000_00000000$_B$
0x034	USB_FIFO5	00000000_00000000_00000000_00000000$_B$

这组 32 位寄存器为软件访问每个端点的 FIFO 提供地址。对这些地址进行写操作，将向相应端点的发送 FIFO 中写入数据；对这些地址进行读操作，将从相应端点的接收 FIFO 中读出数据。
与 FIFO 交换数据的宽度，可以根据需要设置为 8 位、16 位或 32 位，任何允许访问的组合提供的数据存取是连续的。与一个包相关联的数据传送必须是相同宽度的，也就是说这些数据始终是字节对齐、半字对齐或字对齐的。然而，为了完成奇数字节或奇数字的传送，最后一次传送可能比先前的数据传送少一些字节。
根据 FIFO 的大小和预期的最大包大小，FIFO 支持单包或双包缓存。
端点 1－5 发生 STALL 握手响应或发送错误时，清空相应的 FIFO

USB_FIFO0

- ◆ 主机模式/设备模式 ：

端点 0 FIFO 访问寄存器（USB_FIFO0）															
偏移地址：020_H															
复位值：00000000_00000000_00000000_00000000$_B$															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VALUE<31:16>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE<15:0>															

VALUE<31:0>	Bit 31-0	R/W	**端点数据** 对此寄存器进行写操作，将向发送 FIFO 中写入数据， 对此寄存器进行读操作，将从接收 FIFO 中读出数据。

29. 5. 2. 31 USB设备控制寄存器（USB_DEVCTL）

USB_DEVCTL 是用于控制和监测 USB VBUS 线路的 8 位寄存器。如果设备挂起，无设备时钟可以接收，则不对 VBUS 进行采样。此外，在主机模式中，USB_DEVCTL 提供 USB 控制器当前运行模式的状态信息。如果 USB 控制器处于主机模式，此寄存器可指示接入的是全速设备还是低速设备。

◆ 主机模式

USB 设备控制寄存器（USB_DEVCTL）							
偏移地址：060_H							
复位值：10000000_B							
7	6	5	4	3	2	1	0
BDEVICE	FSDEV	LSDEV	VBUS		HOSTMODE	HOSTREQ	SESSION

BDEVICE	Bit 7	R	**设备模式** 0：USB 控制器运行在电缆的 OTG A 端。 1：USB 控制器运行在电缆的 OTG B 端。 此位只在会话过程中有效。
FSDEV	Bit 6	R	**检测到全速设备** 0：没有检测到全速设备。 1：检测到全速设备。
LSDEV	Bit 5	R	**检测到低速设备** 0：没有检测到低速设备。 1：检测到低速设备。
VBUS	Bit 4~3	R	**VBUS 电平** 00：低于 SessionEnd 电平。检测到 VBUS 低于 0.5 V。 01：高于 SessionEnd 电平，低于 AValid 电平。检测到 VBUS 高于 0.5 V，低于 1.5 V。 10：高于 AValid 电平，低于 VBUSValid 电平检测到 VBUS 高于 1.5 V，低于 4.75 V。 11：高于 VBUSValid 电平检测到 VBUS 高于 4.75 V。
HOSTMODE	Bit 2	R	**主机模式** 0：USB 控制器未作为主机。 1：USB 控制器作为主机。
HOSTREQ	Bit 1	R/W	**主机请求** 此位置 1，表示当进入挂起模式时，发起主机协商。 当主机协商完成后，此位清零。
SESSION	Bit 0	R/W	**会话开始/结束（作为 OTG A Device 运行）** 0：通过软件清零来结束会话。 1：通过软件置 1 来开始会话。 **会话开始/结束（作为 OTG B Device 运行）** 0：USB 控制器结束一次会话。如果 USB 控制器处于挂

			起模式，软件清零，连接将软断开。 1：USB 控制器开始一次会话。软件置 1，将发起会话请求协议。 注意，当 USB 控制器由于未定义的行为造成处于未挂起状态时，清零此位。

29. 5. 2. 32 USB DMA配置寄存器（USB_DMACFG）

USB DMA 配置寄存器是一个 8 位的寄存器，它包含多种常见的配置位。这些位包括 RX/TX 早期 DMA 使能位。

USB DMA 配置寄存器（USB_DMACFG）							
偏移地址：061_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved						TXEDMA	RXEDMA

Reserved	Bit 7-2	R	保留
TXEDMA	Bit 1	R/W	0：当 MAXP 字节被写入端点时，所有输入端点的 DMA_REQ 信号都将变为无效，这是晚期模式。 1：当 MAXP-8 字节被写入端点时，所有输入端点的 DMA_REQ 信号都将变为无效，这是早期模式。
RXEDMA	Bit 0	R/W	0：当 MAXP 字节被读出端点时，所有输出端点的 DMA_REQ 信号都将变为无效，这是晚期模式。 1：当 MAXP-8 字节被读出端点时，所有输出端点的 DMA_REQ 信号都将变为无效，这是早期模式。

29. 5. 2. 33 端点发送FIFO长度寄存器（USB_TXFIFOSIZE）

USB_TXFIFOSIZE 是一个 8 位只读寄存器，存储端点（非端点 0）发送的 FIFO 长度信息。

- 主机模式/设备模式

端点发送 FIFO 长度寄存器（USB_TXFIFOSIZE）							
偏移地址：62_H							
复位值：00000_B							
7	6	5	4	3	2	1	0
Reserved			DPB	TXFIFOSIZE			

Reserved	Bit 7-5	-	**保留**
DPB	Bit 4	R	**是否支持双包缓存** 0：仅支持单包缓存。 1：支持双包缓存。
TXFIFOSIZE	Bit 3-0	R	**TX 的 FIFO 长度** TXFIFOSIZE 的取值范围是 0~9。 TX 的 FIFO 长度是 $8*2^{TXFIFOSIZE}$ bytes。当 DPB 位为 1 时，TX 的 FIFO 长度要大一倍。

29. 5. 2. 34 端点接收FIFO长度寄存器（USB_RXFIFOSIZE）

USB_RXFIFOSIZE 是一个 8 位只读寄存器，存储端点（非端点 0）接受的 FIFO 长度信息。

- 主机模式/设备模式

端点接收 FIFO 长度寄存器（USB_RXFIFOSIZE）							
偏移地址：063_H							
复位值：00000_B							
7	6	5	4	3	2	1	0
Reserved			DPB	RXFIFOSIZE			

Reserved	Bit 7-5	-	**保留**
DPB	Bit 4	R	**是否支持双包缓存** 0：仅支持单包缓存。 1：支持双包缓存。
RXFIFOSIZE	Bit 3-0	R	**RX 的 FIFO 长度** RXFIFOSIZE 的取值范围是 0~9。 RX 的 FIFO 长度是 $8*2^{RXFIFOSIZE}$ bytes。当 DPB 位为 1 时，RX 的 FIFO 长度要大一倍。

29. 5. 2. 35 发送端点FIFO起始地址寄存器（USB_TXFIFOADD）

USB_TXFIFOADD 是 16 位寄存器，用于控制所选的发送端点 FIFO 的起始地址。

◆ 主机模式/设备模式

发送端点 FIFO 起始地址寄存器（USB_TXFIFOADD）															
偏移地址：064_H															
复位值：$00000000\_00000000_B$															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							ADDR								

Reserved	Bit 15-9	-	保留
ADDR	Bit 8-0	R/W	发送端点 FIFO 起始地址

ADDR[8:0]	起始地址
0000H	0000H
0001H	0008H
0002H	0010H
...	...
1FFH	0FF8H

29. 5. 2. 36 接收端点FIFO起始地址寄存器（USB_RXFIFOADD）

USB_RXFIFOADD 是 16 位寄存器，用于控制所选接收端点 FIFO 的起始地址。

◆ 主机模式/设备模式

接收端点 FIFO 起始地址寄存器（USB_RXFIFOADD）															
偏移地址：066_H															
复位值：$00000000\_00000000_B$															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							ADDR								

Reserved	Bit 15-9	-	保留
ADDR	Bit 8-0	R/W	接收端点 FIFO 起始地址

ADDR[8:0]	起始地址
0000H	0000H
0001H	0008H
0002H	0010H
...	...
1FFH	0FF8H

29. 5. 2. 37 发送、接收端点信息寄存器（USB_EPINFO）

该 8 位只读寄存器允许读回设计中包含的发送端点和接收端点数目。

发送、接收端点信息寄存器（USB_EPINFO）							
偏移地址：078_H							
复位值：01010101_B							
7	6	5	4	3	2	1	0
RXENDPOINTS				TXENDPOINTS			

RXENDPOINTS	Bit 7-4	R	在设计中实现的接收端点数目
TXENDPOINTS	Bit 3-0	R	在设计中实现的发送端点数目

29. 5. 2. 38 RAM宽度、DMA通道信息寄存器（USB_RAMINFO）

该 8 位只读寄存器提供了有关 RAM 宽度的信息。

RAM 宽度、DMA 通道信息寄存器（USB_RAMINFO）							
偏移地址：079_H							
复位值：01001100_B							
7	6	5	4	3	2	1	0
DMACHANS				RAMBITS			

DMACHANS	Bit 7-4	R	在设计中实现的 **DMA** 通道数目
RAMBITS	Bit 3-0	R	**RAM** 地址总线的宽度

29. 5. 2. 39 连接时序寄存器（USB_LINKINFO）

USB_LINKINFO 是一个用于配置连接和协商延时的 8 位寄存器。

- 主机模式/设备模式

连接时序寄存器（USB_LINKINFO）							
偏移地址：$07A_H$							
复位值：01011100_B							
7	6	5	4	3	2	1	0
WTCON				WTID			

WTCON	Bit 7-4	R/W	**连接等待** 按需求配置等待延时，用于满足用户连接/断开的滤波需求；单位为 533.3 ns。默认为 2.667 μs。
WTID	Bit 3-0	R/W	**等待 ID** 根据需要来配置等待 ID 的延时，等待 ID 值有效时才启用 OTG 的 ID 检测。单位为 4.369 ms。默认为 52.43 ms。

29. 5. 2. 40 VBUS脉冲时序寄存器（USB_VPLEN）

USB_VPLEN 是一个用于配置 VBUS 脉冲充电的持续时间的 8 位寄存器。

- OTG 模式

VBUS 脉冲时序寄存器（USB_VPLEN）							
偏移地址：$07B_H$							
复位值：00111100_B							
7	6	5	4	3	2	1	0
VPLEN							

VPLEN	Bit 7-0	R/W	**VBUS 脉宽** 配置 VBUS 脉冲充电的持续时间，单位 546.1 μs。默认为 32.77 ms。

29. 5. 2. 41 高速传输时间缓冲寄存器（USB_HS_EOF1）

USB_HS_EOF1 是一个 8 位寄存器，用于配置高速模式下最后一次传输开始与帧结束（EOF）之间允许的最小时间间隔。

- 主机模式/设备模式

高速传输时间缓冲寄存器（USB_HS_EOF1）							
偏移地址：$07C_H$							
复位值：10000000_B							
7	6	5	4	3	2	1	0
HSEOF1							

HSEOF1	Bit 7-0	R/W	**高速模式帧结束 EOF 间隙** 在高速传输过程中，设置最后的传输与帧结束（EOF）之间的时间间隔，单位 133.3 ns。默认为 17.07 μs。

29. 5. 2. 42 全速传输时间缓冲寄存器（USB_FS_EOF1）

USB_FS_EOF1 是一个 8 位寄存器，用于配置全速模式下最后一次传输开始与帧结束（EOF）之间允许的最小时间间隔。

- 主机模式/设备模式

全速传输时间缓冲寄存器（USB_FS_EOF1）							
偏移地址：$07D_H$							
复位值：01110111_B							
7	6	5	4	3	2	1	0
FSEOF1							

FSEOF1	Bit 7-0	R/W	**全速模式帧结束 EOF 间隙** 在全速传输过程中，设置最后的传输与帧结束（EOF）之间的时间间隔，单位 533.3 ns。默认为 63.46 μs。

29. 5. 2. 43 低速传输时间缓冲寄存器（USB_LS_EOF1）

USB_LS_EOF1 是一个 8 位寄存器，用于配置低速模式下最后一次传输开始与帧结束（EOF）之间允许的最小时间间隔。

- 主机模式/设备模式

低速传输时间缓冲寄存器（USB_LS_EOF1）							
偏移地址：$07E_H$							
复位值：01110010_B							
7	6	5	4	3	2	1	0
LSEOF1							

LSEOF1	Bit 7-0	R/W	**低速模式帧结束 EOF 间隙** 在低速传输过程中，设置最后的传输与帧结束（EOF）之间的时间间隔，单位 1.067 ns。默认为 121.6 μs。

29. 5. 2. 44 软件复位寄存器（USB_SOFTRST）

该 8 位寄存器的输出复位信号 NRSTO 和 NRSTOX 为低有效。该寄存器自身会清零并由输入 NRST 复位。

软件复位寄存器（USB_SOFTRST）							
偏移地址：$07F_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved						NRSTX	NRST

Reserved	Bit 7-2	R	**保留**
NRSTX	Bit 1	R/W	该位的默认值为 0，当 1 写入该位时，输出 NRSTOX 会在 7 个时钟输入周期的最小延迟内变为有效（低电平）。 对于 XCLK，输出 NRSTOX 是异步有效，同步无效的。 该寄存器自身会清零并由输入 NRST 复位。
NRST	Bit 0	R/W	该位的默认值为 0，当 1 写入该位时，输出 NRSTO 会在 7 个时钟输入周期的最小延迟内变为有效（低电平）。 对于 CLK，输出 NRSTO 是异步有效，同步无效的。 该寄存器自身会清零并由输入 NRST 复位。

29. 5. 2. 45 发送端点功能地址寄存器组（USB_TXxFUNCADDR）

偏移地址	寄存器名称	复位值
0x080	USB_TX0FUNCADDR	00000000_B
0x088	USB_TX1FUNCADDR	00000000_B
0x090	USB_TX2FUNCADDR	00000000_B
0x098	USB_TX3FUNCADDR	00000000_B
0x0A0	USB_TX4FUNCADDR	00000000_B
0x0A8	USB_TX5FUNCADDR	00000000_B

USB_TXxFUNCADDR 是一组 8 位寄存器，用于记录通过相关端点访问的目标功能地址。
注意：USB_TX0FUNCADDR 可用于端点 0 的接收和发送。

USB_TX0FUNCADDR

- 主机模式

发送端点 0 功能地址寄存器（USB_TX0FUNCADDR）							
偏移地址：080_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved	ADDR						

Reserved	Bit 7	-	保留
ADDR	Bit 6-0	R/W	目标设备的 USB 总线地址。

29. 5. 2. 46 发送端点集线器地址寄存器组（USB_TXxHUBADDR）

偏移地址	寄存器名称	复位值
0x082	USB_TX0HUBADDR	00000000_B
0x08A	USB_TX1HUBADDR	00000000_B
0x092	USB_TX2HUBADDR	00000000_B
0x09A	USB_TX3HUBADDR	00000000_B
0x092	USB_TX4HUBADDR	00000000_B
0x09A	USB_TX5HUBADDR	00000000_B

USB_TXxHUBADDR 是一组 8 位寄存器，只有当 USB 设备通过 USB 2.0 集线器连接到发送端点 EP 时才被设置。此寄存器记录 USB 2.0 集线器的地址，通过该地址可访问与端点相关联的目标。注意：USB_TX0HUBADDR 可用于端点 0 的接收和发送。

USB_TX0HUBADDR

- ◆ 主机模式

发送端点 0 集线器地址寄存器（USB_TXxHUBADDR）							
偏移地址：082_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
MULTTRAN	ADDR						

MULTTRAN	Bit 7	R/W	**多路转换器** 0：指示集线器具有单路转换器。 1：指示集线器具有多路转换器。
ADDR	Bit 6-0	R/W	**USB 2.0 集线器的 USB 总线地址。**

29. 5. 2. 47 发送端点集线器端口寄存器组（USB_TXxHUBPORT）

偏移地址	寄存器名称	复位值
0x083	USB_TX0HUBPORT	00000000_B
0x08B	USB_TX1HUBPORT	00000000_B
0x093	USB_TX2HUBPORT	00000000_B
0x09B	USB_TX3HUBPORT	00000000_B
0x0A3	USB_TX4HUBPORT	00000000_B
0x0AB	USB_TX5HUBPORT	00000000_B

USB_TXxHUBPORT 是一组 8 位寄存器，只有当低速设备或全速设备通过 USB 2.0 集线器连接到发送端点 EP 时才必须写入。

注意：USB_TX0HUBPORT 可用于端点 0 的接收和发送。

USB_TX0HUBPORT

- ◆ 主机模式

发送端点 0 集线器端口寄存器（USB_TX0HUBPORT）							
偏移地址：083_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved	HUBPORT						

Reserved	Bit 7	-	保留
HUBPORT	Bit 6-0	R/W	**USB 2.0 集线器的端口，通过该端口可访问与端点相关联的目标**

29. 5. 2. 48 接收端点功能地址寄存器组（USB_RXxFUNCADDR）

偏移地址	寄存器名称	复位值
0x08C	USB_RX1FUNCADDR	00000000_B
0x094	USB_RX2FUNCADDR	00000000_B
0x09C	USB_RX3FUNCADDR	00000000_B
0x0A4	USB_RX4FUNCADDR	00000000_B
0x0AC	USB_RX5FUNCADDR	00000000_B

USB_RXxFUNCADDR 是一组 8 位寄存器，用于记录通过相关端点访问的目标功能地址。
注意：USB_TX0FUNCADDR 可用于端点 0 的接收和发送。

USB_RX1FUNCADDR

- ◆ 主机模式

接收端点 1 功能地址寄存器（USB_RX0FUNCADDR）							
偏移地址：$08C_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved	ADDR						

Reserved	Bit 7	-	保留
ADDR	Bit 6-0	R/W	目标设备的 USB 总线地址。

29. 5. 2. 49 接收端点集线器地址寄存器组（USB_RXxHUBADDR）

偏移地址	寄存器名称	复位值
0x08E	USB_RX1HUBADDR	00000000_B
0x096	USB_RX2HUBADDR	00000000_B
0x09E	USB_RX3HUBADDR	00000000_B
0x0A6	USB_RX4HUBADDR	00000000_B
0x0AE	USB_RX5HUBADDR	00000000_B

USB_RXxHUBADDR 是一组 8 位寄存器，只有当 USB 设备通过 USB 2.0 集线器连接到接收端点 EP 时才被设置。此寄存器记录 USB 2.0 集线器的地址，通过该地址可访问与端点相关联的目标。注意：USB_TX0HUBADDR 可用于端点 0 的接收和发送。

USB_RX1HUBADDR

- ◆ 主机模式

接收端点 1 集线器地址寄存器（USB_RXxHUBADDR）							
偏移地址：$08E_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
MULTTRAN	ADDR						

MULTTRAN	Bit 7	R/W	**多路转换器** 0：指示集线器具有单路转换器。 1：指示集线器具有多路转换器。
ADDR	Bit 6-0	R/W	**USB 2.0 集线器的 USB 总线地址。**

29. 5. 2. 50 接收端点集线器端口寄存器组（USB_RXxHUBPORT）

偏移地址	寄存器名称	复位值
0x08F	USB_RX1HUBPORT	00000000_B
0x097	USB_RX2HUBPORT	00000000_B
0x09F	USB_RX3HUBPORT	00000000_B
0x0A7	USB_RX4HUBPORT	00000000_B
0x0AF	USB_RX5HUBPORT	00000000_B

USB_RXxHUBPORT 是一组 8 位寄存器，只有当低速设备或全速设备通过 USB 2.0 集线器连接到接收端点 EP 时才必须写入。此寄存器记录 USB 2.0 集线器的端口，通过该端口可访问与端点相关联的目标。

注：USB_TX0HUBPORT 可用于端点 0 的接收和发送。

USB_RX1HUBPORT

- 主机模式：

接收端点 1 集线器端口寄存器（USB_RX1HUBPORT）							
偏移地址：$08F_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved	HUBPORT						

Reserved	Bit 7	-	保留
HUBPORT	Bit 6-0	R/W	集线器端口定义

29. 5. 2. 51 端点发送最大传输数据寄存器组（USB_TXxMAXP）

偏移地址	寄存器名称	复位值
0x110	USB_TX1MAXP	$00000000\_00000000_B$
0x120	USB_TX2MAXP	$00000000\_00000000_B$
0x130	USB_TX3MAXP	$00000000\_00000000_B$
0x140	USB_TX4MAXP	$00000000\_00000000_B$
0x150	USB_TX5MAXP	$00000000\_00000000_B$

USB_TX1MAXP 是一个 16 位寄存器，定义了通过发送端点 1 单次可传输的最大数据的长度。该长度最大为 1024 字节，但必须服从“USB 规范”里批量传输、中断传输和全速模式下同步传输的包长限制。

写入寄存器中的值代表的数据总大小不能超过发送端点的 FIFO 大小，如果支持双包缓存，不能超过 FIFO 大小的一半。

如果此寄存器在端点发送包后发生改变，在寄存器写入新值之前，发送端点 FIFO 必须完全被清空（使用 USB_TXxCSRL 寄存器中的 FLUSHFIFO 位）。

- 主机模式/设备模式

端点 1 发送最大传输数据寄存器（USB_TX1MAXP）

偏移地址：110_H

复位值：$00000000\_00000000_B$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					MAXLOAD										

Reserved	Bit 15-11	-	保留
MAXLOAD	Bit 10-0	R/W	发送端点 1 单次可传输的最大数据的长度

29. 5. 2. 52 端点 0 控制和状态低字节寄存器（USB_CSR0L）

USB_CSR0L 是一个 8 位寄存器，是端点 0 控制和状态的低字节寄存器。

◆ 主机模式：

端点 0 控制和状态低字节寄存器（USB_CSR0L）							
偏移地址：102_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
NAKTOUT	STATUSPKT	REQPKT	ERROR	SETUPPKT	RXSTALL	TXPKTRDY	RXPKTRDY

NAKTOUT	Bit 7	R/W	**NAK 超时标志位** 当端点 0 收到 NAK 握手包的时间大于寄存器 USB_IND_NAK 设置值时此标志位被 USB 控制器置 1。当此标志位置 1 时，应通过软件写 0 清零此标志位让端点 0 继续进行事务处理。
STATUSPKT	Bit 6	R/W	**Status stage 事务处理控制位** 控制位 TXPKTRDY 或 REQPKT 置 1 的同时软件置 1 此控制位将会执行状态阶段事务处理。在置 1 此控制的同时要确保 DTOG 置 1 即采用 DATA1 奇数包进行状态阶段事务处理。
REQPKT	Bit 5	R/W	**IN 事务处理请求控制位** 0：无 IN 事务处理请求 1：请求 IN 事务处理 当 RXPKTRDY 置 1 时，会自动清零此控制位。
ERROR	Bit 4	R/W	**传输错误标志位** USB 控制器对某一事务处理重试 3 次，USB 设备无任何应答时，此标志位置 1 表明事务处理产生错误。软件写 0 清零此标志位，写 1 无效。
SETUPPKT	Bit 3	R/W	**SETUP 令牌包发送控制位** 软件置 1 控制位 TXPKTRDY 的同时置 1 此控制位，将会以 SETUP 令牌包事务处理取代 OUT 令牌包事务处理。软件置 1 此控制位的同时会清零 DTOG 控制位。
RXSTALL	Bit 2	R/W	**接收 STALL 握手信号** 当完成 STALL 握手信号接收时，USB 控制器自动置 1 此控制位。此控制位仅可通过软件写 0 清，软件写 1 无效。
TXPKTRDY	Bit 1	R/W	**发送数据包 Ready 标志位** 0：端点 0 数据包加载未完成 1：端点 0 数据包加载完成（软件写 1 有效，写 0 无效） 当软件完成端点 0 数据包加载时，软件需要将此控制位置 1。当完成数据包发送时，此控制位自动清零，同时端点 0 产生中断（中断使能时）。

RXPKTRDY	Bit 0	R/W	**数据包接收完成标志位** 0：数据包接收未完成 1：数据包接收完成 当此控制位置 1 时，如果中断使能，端点 0 会产生中断。 软件写 0 清零此控制位（软件写 1 无效）。

◆ 设备模式：

端点 0 控制和状态低字节寄存器（USB_CSR0L）							
偏移地址：102_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
SSTPEND	SRXPKTRDY	SENDSTALL	SETUPEND	DATAEND	SENTSTALL	TXPKTRDY	RXPKTRDY

SSTPEND	Bit 7	W	**ServicedSetupEnd 清零控制位** 软件写 1 清零此标志位（写 0 无效）
SRXPKTRDY	Bit 6	W	**ServicedRxPktRdy 清零控制位** 软件写 1 清零此标志位（写 0 无效）
SENDSTALL	Bit 5	W	**发送 STALL 控制位** 软件置 1 此控制位时会中止当前事务处理，并且发送 STALL 握手包。此控制位自动清零（软件写 0 无效）。
SETUPEND	Bit 4	R	**SETUP 事务处理完成标志位** 0：控制事务处理未完成 1：控制事务处理完成 在 DATAEND 控制位置 1 之前完成控制事务处理此控制位置 1，同时会产生相应的中断和 FIFO Flush 操作。控制位 SSTPEND 写 1 清零此标志位。
DATAEND	Bit 3	W	**数据完成控制位** 当产生如下情况时，软件需要将此位置 1（软件写 0 无效）： 1）完成最后一包数据加载置位 TXPKTRDY； 2）完成最后一包数据读取清零 RXPKTRDY； 3）发送长度为 0 的数据包置位 TXPKTRDY； 此控制位自动清零。
SENTSTALL	Bit 2	R/W	**发送 STALL 握手信号** 当完成 STALL 握手信号传输时，USB 控制器置 1 此控制位。此控制位仅可通过软件写 0 清，软件写 1 无效。
TXPKTRDY	Bit 1	R/W	**发送数据包 Ready 标志位** 0：端点 0 数据包加载未完成 1：端点 0 数据包加载完成（软件写 1 有效，写 0 无效） 当软件完成端点 0 数据包加载时，软件需要将此控制位置 1。当完成数据包发送时，此控制位自动清零，同时端点 0 产生中断（中断使能时）。
RXPKTRDY	Bit 0	R	**数据包接收完成标志位** 0：数据包接收未完成 1：数据包接收完成 当此控制位置 1 时，如果中断使能端点 0 会产生中断。软件置 1 控制位 SRXPKTRDY 时清零此控制位。

29. 5. 2. 53 端点 0 控制和状态高字节寄存器（USB_CSR0H）

USB_CSR0H 是一个 8 位寄存器，提供对端点 0 的控制和状态的高字节寄存器。

◆ 主机模式：

端点 0 控制和状态高字节寄存器（USB_CSR0H）							
偏移地址：103_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved				DISPING	DTOG_WN	DTOG	FLUSHFIFO

Reserved	Bit 7-4	-	**保留**
DISPING	Bit 3	R/W	**PING 令牌包禁止控制位** 0：高速控制传输使能 PING 事务处理 1：高速控制传输禁止 PING 事务处理
DTOG_WN	Bit 2	W	**数据触发位写使能** 软件置 1 此控制位时，端点 0 的 DTOG 数据触发位可写。当新的值写入 DTOG 数据触发位后，此控制位自动清零。
DTOG	Bit 1	R/W	**数据触发位** 0：数据触发位为 0 1：数据触发位为 1 仅当 DTOG_WN 置 1 时，才可以写此控制位，否则只可读不可。
FLUSHFIFO	Bit 0	W	**FIFO Flush 控制位** 软件置 1 此控制位时，会触发从端点 0 的 FIFO 中发送或读取下一包数据，同时复位 FIFO 指针、清零 TXPKTRDY/RXPKTRDY 标志位。注意，仅当 TXPKTRDY/RXPKTRDY 为 1 时才可使用此控制位，否则可能会引起数据被破坏。 此控制位自动清零（软件写 0 无效）

◆ 设备模式 ：

端点 0 控制和状态高字节寄存器（USB_CSR0H）							
偏移地址：103_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved							FLUSHFIFO

Reserved	Bit 7-1	-	**保留**
FLUSHFIFO	Bit 0	W	**FIFO Flush 控制位** 软件置 1 此控制位时，会触发从端点 0 FIFO 中发送或读取下一包数据，同时复位 FIFO 指针清零 TXPKTRDY/RXPKTRDY 标志位。仅当 TXPKTRDY/RXPKTRDY 为 1 时才可使用此控制位，否则可能会引起数据被破坏。 此控制位自动清零（软件写 0 无效）

29. 5. 2. 54 端点 0 接收字节数量寄存器（USB_COUNT0）

USB_COUNT0 是一个 8 位寄存器，用来指示端点 0 的 FIFO 中收到数据的字节数量。此值伴随 FIFO 中内容变化而变化，只有 RXPKTRDY 位置 1 时有效。

- ◆ 主机模式/设备模式

端点 0 接收字节数量寄存器（USB_COUNT0）							
偏移地址：108_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved	COUNT						

Reserved	Bit 7	-	**保留**
COUNT	Bit 6-0	R	**指示端点 0 的 FIFO 中收到的数据的字节数量。** 此值伴随 FIFO 中内容变化而变化。

29. 5. 2. 55 端点 0 发送配置寄存器（USB_TYPE0）

USB_TYPE0 是一个 8 位寄存器，用于存储与端点 0 通讯的目标设备的运行速度。

- ◆ 主机模式

端点 0 发送配置寄存器（USB_TYPE0）							
偏移地址：$10A_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
SPEED		Reserved					

SPEED	Bit 7-6	R/W	**目标设备的运行速度控制位** 00：NC 01：高速 10：全速 11：低速
Reserved	Bit 5-0	-	**保留**

29. 5. 2. 56 端点 0 的控制和状态高字节寄存器（USB_CONFIG0）

USB_CONFIG0 是一个 8 位只读寄存器，存储端点 0 的 USB 内核配置信息。

- 主机模式/设备模式

端点 0 的控制和状态高字节寄存器（USB_CONFIG0）							
偏移地址：$10F_H$							
复位值：11011010_B							
7	6	5	4	3	2	1	0
MPRXE	MPTXE	BIGENDIAN	HBRXE	HBTXE	FIFOSIZE	SOFTCONE	UTMI DATAWIDTH

MPRXE	Bit 7	R	MPRXE 位始终为 1，表示批量传输模式下支持接收数据包拆分。
MPTXE	Bit 6	R	MPTXE 位始终为 1，表示批量传输模式下支持发送数据包拆分。
BIGENDIAN	Bit 5	R	BIGENDIAN 位始终为 0，表示数据为小端模式。
HBRXE	Bit 4	R	HBRXE 位始终为 1，表示使能宽带接收端点。
HBTXE	Bit 3	R	HBTXE 位始终为 1，表示使能宽带发送端点。
FIFOSIZE	Bit 2	R	FIFOSIZE 位始终为 0，表示 FIFO 大小固定。
SOFTCONE	Bit 1	R	SOFTCONE 位始终为 1，表示软件连接和断开。
UTMI DATAWIDTH	Bit 0	R	DATAWIDTH 位始终为 0，表示数据位宽为 8bit。

29. 5. 2. 57　端点 0 无应答超时时间设置寄存器（USB_NAK）

USB_NAK 是一个 8 位寄存器，用于设置在接收到一连串的 NAK 响应时，端点 0 应设置多少个帧的超时。

所选的帧数为 $2^{(m-1)}$（其中，m 是该寄存器中设定的值，其有效数值为 2-16）。如果主机从目标设备接收的 NAK 相应大于此寄存器中代表的限度值，端点将暂停。

注意：寄存器值为 0 或 1 时将禁止 NAK 超时功能

- ◆ 主机模式

端点 0 无应答超时时间设置寄存器（USB_NAK）							
偏移地址：$10B_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved			NAKLIMIT				

Reserved	Bit 7-5	-	保留
NAKLIMIT	Bit 4-0	R/W	**NAK 延时帧数**

29. 5. 2. 58 端点发送控制和状态低字节寄存器组（USB_TXxCSRL）

偏移地址	寄存器名称	复位值
0x112	USB_TX1CSRL	00000000_B
0x122	USB_TX2CSRL	00000000_B
0x132	USB_TX3CSRL	00000000_B
0x142	USB_TX4CSRL	00000000_B
0x152	USB_TX5CSRL	00000000_B

这组 8 位寄存器组是端点发送控制和状态低字节寄存器。

USB_TX1CSRL

◆ 主机模式：

端点 1 发送控制和状态低字节寄存器（USB_TX1CSRL）							
偏移地址：112_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
NAKTOUT	CLRDATAT	RXSTALL	SETUPPKT	FLUSHFIFO	ERROR	FIFONE	TXPKTRDY

NAKTOUT	Bit 7	R/W	**NAK 超时** 0：无超时 1：只用于批量传输端点接收 NAK 的次数超过了寄存器 USB_IND_NAK 设置的次数。软件需要清除此位来允许端点继续传输。 只用于宽带中断端点如果没有收到已发送数据包的应答，此位会被置 1。
CLRDATAT	Bit 6	R/W	**清除数据转换** 0：清除数据转换 1：向此位写 1 可将寄存器 USB_IND_TXCSRH 的 FDATAT 位清零。
RXSTALL	Bit 5	R/W	**端点挂起** 0：没有接收到 STALL 握手信号 1：收到 STALL 握手信号。此位需要软件清零。
SETUPPKT	Bit 4	R/W	**建立令牌包** 0：不发送建立令牌包 1：为此传输事务发送建立令牌包而不是输出令牌包。在置 1 TXPKTRDY 位的同时，也应将此位置位。 向此位写 1 会将寄存器 USB_IND_TXCSRH 的 FDATAT 位清零。
FLUSHFIFO	Bit 3	R/W	**清空 FIFO** 0：无影响

			1：清空发送端点 FIFO 中最后的包。同时，FIFO 指针复位，TXPKTRDY 位清零。此时 USB_TXIS 寄存器中的 EPnTX_IF 位也要置位。 此位与 TXPKTRDY 位同时置位，将放弃当前正在装载到 FIFO 的数据包。仅当 TXPKTRDY 为 1 时才可置 1 此控制位，否则可能会引起数据被破坏。当端点为双缓存 FIFO 时，应该置 1 此控制位两次，完成 FIFO 中全部数据清除。
ERROR	Bit 2	R/W	**错误：** 0：无错误 1：尝试三次发送包，都未接收到握手包。此时 ERROR 位会被自动置 1，TXPKTRDY 位清零，FIFO 完全清空。 注意，此位仅在批量传输和中断传输模式下有效。
FIFONE	Bit 1	R/W	**FIFO 不空** 0：FIFO 空 1：FIFO 不空
TXPKTRDY	Bit 0	R/W	**发送包准备好** 0：未准备好发送包。 1：在装载数据包到发送 FIFO 中后，软件置 1 此位。 当数据包发送完成时，此位自动清零，并产生中断。向双缓存的 FIFO 中装载第二个数据包时，TXPKTRDY 位也将自动清零。

◆ 设备模式：

端点 1 发送控制和状态低字节寄存器（USB_TX1CSRL）							
偏移地址：112_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
INCOMPTX	CLRDATAT	SENTSTALL	SENDSTALL	FLUSHFIFO	UNDERRUN	FIFONE	TXPKTRDY

INCOMPTX	Bit 7	R/W	**数据包发送不完全** 0：数据包完全发送 1：数据包不完全发送 当发送端点进行宽带同步传输时，大的数据包会被分割成 2 或 3 个数据包进行传输，但是当收到的 IN 令牌包不足发送完所有数据包时，此标志位被置 1 表明数据发送不完全。 软件写 0 清零此标志位（软件写 1 无效）
CLRDATAT	Bit 6	W	**数据转换清除** 软件置 1 此控制位，清零端点 DTOG（软件写 0 无效）
SENTSTALL	Bit 5	R/W	**发送 STALL** 0：未发送 STALL 握手包

			1：发送 STALL 握手包 当发送 STALL 握手包时 USB 控制器置 1 此控制位，同时会执行 FIFO Flush 和清零 TXPKTRDY。软件写 0 清零此标志位（软件写 1 无效）。
SENDSTALL	Bit 4	R/W	**发送 STALL** 对于 IN 令牌包，软件置 1 此控制位会触发 STALL 握手包发送，软件清 0 此控制位终止 STALL 握手包。 此控制位对同步传输无效。
FLUSHFIFO	Bit 3	W	**清空 FIFO** 软件置 1 此控制位时，会触发从端点 FIFO 中发送下一包数据，同时复位 FIFO 指针、清零 TXPKTRDY 标志位。 仅当 TXPKTRDY 为 1 时才可置 1 此控制位，否则可能会引起数据被破坏。当端点为双缓存 FIFO 时，应该置 1 此控制位两次，完成 FIFO 中全部数据清除。 此控制位自动清零（软件写 0 无效）
UNDERRUN	Bit 2	R/W	**下溢** 当 TXPKTRDY 为 0 时，收到 IN 令牌包，USB 控制器置 1 此标志位。当软件检测到此标志位为 0 时应该写 0 清零此标志位（软件写 1 无效）
FIFONE	Bit 1	R/W	**FIFO 不空** 0：发送 FIFO 空 1：发送 FIFO 非空 当发送 FIFO 中至少存在一包数据时，USB 控制器置 1 此标志位
TXPKTRDY	Bit 0	R/W	**数据包准备好** 0：数据包加载未完成 1：数据包加载完成（软件写 1 有效，写 0 无效） 当软件完成端点 FIFO 数据包加载时，软件需要将此控制位置 1。当完成数据包发送时，此控制位自动清零，同时端点产生中断（中断使能时）。当端点 FIFO 为双缓存 FIFO 时，在加载第二个数据包到 FIFO 时 TXPKTRDY 会自动清零。

29. 5. 2. 59 端点发送控制和状态高字节寄存器组（USB_TXxCSRH）

偏移地址	寄存器名称	复位值
0x113	USB_TX1CSRH	00000000_B
0x123	USB_TX2CSRH	00000000_B
0x133	USB_TX3CSRH	00000000_B
0x143	USB_TX4CSRH	00000000_B
0x153	USB_TX5CSRH	00000000_B

这组 8 位寄存器组是端点发送控制和状态低字节寄存器。

USB_TX1CSRH

- 主机模式：

端点 1 发送控制和状态高字节寄存器（USB_TX1CSRH）							
偏移地址：113_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
AUTOSET	Reserved	MODE	DMAREQENAB	FDATAT	DMAREQMODE	DATATWE	DATAT

AUTOSET	Bit 7	R/W	**自动置位** 当软件将此位置 1 时，当最大包长中（USB_TXxMAXP 中的值）的数据装载到发送 FIFO 时，TXPKTRDY 位可自动置 1。如果装载的数据包小于最大包大小，TXPKTRDY 位必须手动置 1。 注意：宽带同步传输端点和宽带中断传输端点时，不能将 AUTOSET 位置 1。
Reserved	Bit 6	-	**保留**
MODE	Bit 5	R/W	**端点方向设置** 0：设置端点方向为接收。 1：设置端点方向为发送。 注意：此位只在发送和接收传输使用相同端点 FIFO 时起作用。
DMAREQENAB	Bit 4	-	**软件设置该位来使能 TX 端点的 DMA 请求。**
FDATAT	Bit 3	R/W	**强制数据切换** 0：无影响 1：软件将此位置 1 时，无论是否收到 ACK 确认，强制端点切换 DATAT 位，清空 FIFO 中的数据包。中断传输端点可以将此位用于反馈同步端点的通信速率。
DMAREQMODE	Bit 2	-	**软件置位该位来选择 DMA 请求模式 1，清除该位来选择 DMA 请求模式 0。**

			注：该位不能在上述 DMAREQENAB 位被清零之前或者同一周期内清零。
DATATWE	Bit 1	W	**数据切换启用** 0：不能对 DATAT 位进行写操作。 1：使能对 DATAT 位进行写操作。 一旦写入新值，此位自动清零。
DATAT	Bit 0	R/W	**数据切换** 此位指示发送端点数据切换的当前状态。如果 DATATWE 位为 1，此位可根据数据切换的需求设置写值。如果 DATATWE 位为 0，写到该位的任何值都将被忽略。

◆ 设备模式：

端点 1 发送控制和状态高字节寄存器（USB_TX1CSRH）							
偏移地址：113_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
AUTOSET	ISO	MODE	Reserved	FDATAT	Reserved		

AUTOSET	Bit 7	R/W	**自动置位** 当软件将此位置 1 时，当最大包长中（USB_TXxMAXP 中的值）的数据装载到发送 FIFO 时，TXPKTRDY 位可自动置 1。如果装载的数据包小于最大包大小，TXPKTRDY 位必须手动置 1。 注意：宽带同步传输端点和宽带中断传输端点时，不能将 AUTOSET 位置 1。
ISO	Bit 6	R/W	**同步传输** 0：启用发送端点的批量或中断传输。 1：启用发送端点的同步传输。
MODE	Bit 5	R/W	**端点方向设置** 0：设置端点方向为接收。 1：设置端点方向为发送。 注意：此位只在发送和接收传输使用相同端点 FIFO 时起作用。
Reserved	Bit 4	R/W	**保留**
FDATAT	Bit 3	R/W	**强制数据切换** 0：无影响 1：软件将此位置 1 时，无论是否收到 ACK 确认，强制端点切换 DATAT 位，清空 FIFO 中的数据包。中断传输端点可以将此位用于反馈同步端点的通信速率。
Reserved	Bit 2-0	-	**保留**

29. 5. 2. 60 端点接收最大传输数据寄存器组（USB_RXxMAXP）

偏移地址	寄存器名称	复位值
0x114	USB_RX1MAXP	$00000000\_00000000_B$
0x124	USB_RX2MAXP	$00000000\_00000000_B$
0x134	USB_RX3MAXP	$00000000\_00000000_B$
0x144	USB_RX4MAXP	$00000000\_00000000_B$
0x154	USB_RX5MAXP	$00000000\_00000000_B$

这组 16 位寄存器 USB_RXxMAXP 定义了通过接收端点单次可传输的最大数据的长度。
Bit10~0 定义了单次传输的最大发送数据长度。该值最大为 1024 字节，但必须服从“USB 规范”里传输的包长限制。
写入寄存器中的值代表的数据总大小不能超过接收端点的 FIFO 大小，如果支持双包缓存，不能超过 FIFO 大小的一半。

USB_RX1MAXP

- 主机模式/设备模式：

端点 1 接收最大传输数据寄存器（USB_RX1MAXP）															
偏移地址：114_H															
复位值：$00000000\_00000000_B$															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					MAXLOAD										

Reserved	Bit 15-11	-	保留
MAXLOAD	Bit 10-0	R/W	接收端点单次可传输的最大数据的长度

29. 5. 2. 61 端点接收控制和状态低字节寄存器组（USB_RXxCSRL）

偏移地址	寄存器名称	复位值
0x116	USB_RX1CSRL	00000000_B
0x126	USB_RX2CSRL	00000000_B
0x136	USB_RX3CSRL	00000000_B
0x146	USB_RX4CSRL	00000000_B
0x156	USB_RX5CSRL	00000000_B

这组 8 位寄存器组是端点接收控制和状态低字节寄存器。

USB_RX1CSRL

- 主机模式：

端点 1 接收控制和状态低字节寄存器（USB_RX1CSRL）							
偏移地址：116_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
CLRDATAT	RXSTALL	REQPKT	FLUSHFIFO	DER_NAKO	ERROR	FIFOFULL	RXPKTRDY

CLRDATAT	Bit 7	W	**清除数据转换** 向此位写 1，可以清除数据转换。
RXSTALL	Bit 6	R/W	**端点挂起** 0：没有接收到 STALL 握手信号。 1：接收到 STALL 握手信号，此时会产生中断。 此位需要软件清零。
REQPKT	Bit 5	R/W	**请求包** 0：无请求。 1：请求输入事务。 当 RXPKTRDY 位置 1 时，此位清零。
FLUSHFIFO	Bit 4	R/W	**清空 FIFO** 0：无影响 1:清空下一个将从端点 FIFO 中读出的数据包。同时，FIFO 指针复位，RXPKTRDY 位清零。注意，如果 FIFO 是双缓存的，FLUSHFIFO 位需被置 1 两次才可清空 FIFO。 只有在 RXPKTRDY 位为 1 时，此位才能清零。否则，可能会导致数据损坏。
DER_NAKO	Bit 3	R/W	**数据错误/NAK 超时** 同步模式下此位和 RXPKTRDY 位同时为 1，表示数据包有 CRC 或位填充错误。RXPKTRDY 位清零时，此位也会清零。 批量模式下接收 NAK 响应的次数超过了

			USB_RXxINTERVAL 寄存器中设置的次数，接收端点暂停。需软件清除此位允许端点继续工作。
ERROR	Bit 2	R/W	**错误：** 0：无错误 1：三次接收数据包，都未接收到数据包。此时 ERROR 位会被自动置 1。 注意，此位仅在批量传输或中断传输模式下有效。
FIFOFULL	Bit 1	R	**FIFO 满** 0：FIFO 未满 1：FIFO 满
RXPKTRDY	Bit 0	R/W	**接收包准备好** 0：未接收到数据包。 1：接收到数据包。 如果 USB_RXxCSRH 寄存器中的 AUTOCLR 位置 1，则当 USB_RXxMAXP 字节的数据包从接收 FIFO 读出时，该位会自动清零。如果 AUTOCLR 位为零或者读出了比最大包长小的数据包，则软件必须在数据包已从接收 FIFO 中读出时将该位清零。

◆ 设备模式：

端点 1 接收控制和状态低字节寄存器（USB_RX1CSRL）							
偏移地址：116_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
CLRDATAT	SENTSTALL	SENDSTALL	FLUSHFIFO	DATAERROR	OVERRUN	FIFOFULL	RXPKTRDY

CLRDATAT	Bit 7	W	**清除数据转换** 向此位写 1，可以清除数据转换。
SENTSTALL	Bit 6	R/W	**端点挂起** 0：未发送 STALL 握手。 1：已发送 STALL 握手。 必须通过软件清除此位。
SENDSTALL	Bit 5	R/W	**发送 STALL** 0：无影响 1：发送 STALL 握手包 注意，此位同步传输下无效。
FLUSHFIFO	Bit 4	R/W	**清空 FIFO** 0：无影响 1：清空从端点接收 FIFO 中读出的下一个数据包。同时，FIFO 指针复位，RXPKTRDY 位清零。 当端点为双缓存 FIFO 时，应该置 1 此控制位两次，完成 FIFO 中全部数据清除。
DATAERROR	Bit 3	R	**数据错误**

			0：正常工作。 1：表示 RXPKTRDY 位置 1 且数据包有 CRC 或位填充错误。 注意，只有在同步模式下，此位才有效。
OVERRUN	Bit 2	R/W	**上溢** 0：无错误。 1：表示输出数据包不能装载到接收 FIFO 中。 注意，只有在同步模式下，此位才有效。
FIFOFULL	Bit 1	R	**FIFO 满** 0：接收 FIFO 未满 1：接收 FIFO 满
RXPKTRDY	Bit 0	R/W	**接收包准备好** 0：未接收到数据包。 1：已接收到数据包。 如果 USB_RXxCSRH 寄存器中的 AUTOCLR 位置 1，则当 USB_RXxMAXP 字节的数据包从接收 FIFO 读出时，该位会自动清零。如果 AUTOCLR 位为零或者读出了比最大包长小的数据包，则软件必须在数据包已从接收 FIFO 中读出时将该位清零。

29. 5. 2. 62 端点接收控制和状态高字节寄存器组（USB_RXxCSRH）

偏移地址	寄存器名称	复位值
0x117	USB_RX1CSRH	00000000_B
0x127	USB_RX2CSRH	00000000_B
0x137	USB_RX3CSRH	00000000_B
0x147	USB_RX4CSRH	00000000_B
0x157	USB_RX5CSRH	00000000_B

这组 8 位寄存器组是端点接收控制和状态高字节寄存器。

USB_RX1CSRH

- 主机模式：

端点 1 接收控制和状态高字节寄存器（USB_RX1CSRH）							
偏移地址：117_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
AUTOCLR	AUTOREQ	DMAREQENAB	PIDERROR	DMAREQMODE	DATATWE	DATAT	INCOMPRX

AUTOCLR	Bit 7	R/W	**自动清零** 0：不使能自动清零 1：当 USB_RXxMAXP 定义的数据包已从接收 FIFO 读出时，RXPKTRDY 位可自动清零。如果读出的包长小于最大包长时，必须手动将 RXPKTRDY 位置 1。 注意，宽带同步模式端点情况下，此位不能置 1。
AUTOREQ	Bit 6	R/W	**自动请求** 0：无自动请求。 1：当 RXPKTRDY 位清零时，REQPKT 位可自动置 1。 注意，当收到一个短包时，此位自动清零。
DMAREQENAB	Bit 5	-	**软件置位该位来使能 RX 端点的 DMA 请求。**
PIDERROR	Bit 4	R	**PID 错误** 0：无错误 1：同步传输的接收包中存在 PID 错误。 在批量或中断传输时，此位忽略。
DMAREQMODE	Bit 3	-	**软件置位该位来选择 DMA 请求模式 1，清除该位来选择 DMA 请求模式 0。**
DATATWE	Bit 2	R	**数据切换写启用** 0：不能对 DATAT 位进行写操作。 1：可以对 DATAT 位进行写操作。
DATAT	Bit 1	R	**数据切换**

			此位指示端点 0 数据切换的当前状态。 如果 DATATWE 位为高电平，此位可根据数据切换的需求设置写值。如果 DATATWE 位为低电平，写到该位的任何值都将被忽略。
INCOMPRX	Bit 0	R/W	**接收包完成** 0：接收数据包完成。 1：表示在宽带同步模式或中断模式下，数据包接收未完成。 当 RXPKTRDY 位被清零时，此位也会被清零。 USB 工作正常情况下，此位不会被置 1。

◆ 设备模式：

端点 1 接收控制和状态高字节寄存器（USB_RXCSRH）							
偏移地址：117_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
AUTOCLR	ISO	Reserved	DN_PIDERR	Reserved			INCOMPRX

AUTOCLR	Bit 7	R/W	**自动清零** 0：不使能自动清零 1：当 USB_RXxMAXP 定义的数据包已从接收 FIFO 读出时，RXPKTRDY 位可自动清零。如果读出的包长小于最大包长时，必须手动将 RXPKTRDY 位置 1。 注意，宽带同步模式端点情况下，此位不能置 1。
ISO	Bit 6	R/W	**同步传输** 0：启用接收端点的批量/中断传输。 1：启用接收端点的同步传输。
Reserved	Bit 5	-	**保留**
DN_PIDERR	Bit 4	R	**禁止 NYET/PID 错误** 此位为 1，表明对于批量或中断传输：禁止 NYET 握手的发送。当此位置 1，所有成功接收的包都将被确认，即使此时 FIFO 变满。 对于同步时传输：表明接收到的数据包存在 PID 错误。
Reserved	Bit 3-1	-	**保留**
INCOMPRX	Bit 0	R/W	**接收包完成** 0：接收数据包完成。 1：表示在宽带同步模式或中断模式下，数据包接收未完成。 当 RXPKTRDY 位被清零时，此位也会被清零。 USB 工作正常情况下，此位不会被置 1。

29. 5. 2. 63 端点接收字节数量寄存器组（USB_RXxCOUNT）

偏移地址	寄存器名称	复位值
0x118	USB_RX1COUNT	00000000_00000000$_B$
0x128	USB_RX2COUNT	00000000_00000000$_B$
0x138	USB_RX3COUNT	00000000_00000000$_B$
0x148	USB_RX4COUNT	00000000_00000000$_B$
0x158	USB_RX5COUNT	00000000_00000000$_B$

端点接收字节数量寄存器组是一组 16 位寄存器，用于指示在要从接收 FIFO 读出的，当前行的数据包中保留的数据字节数。如果被发送的是多个批量数据包，则给出的数字适用于组合的数据包。注意，返回的值随着读 FIFO 而改变，此值只在寄存器 USB_RXxCSRL 中的 RXPKTRDY 位为 1 时有效。

USB_RX1COUNT

- 主机模式/设备模式：

端点 1 接收字节数量寄存器（USB_RX1COUNT）															
偏移地址：118_H															
复位值：00000000_00000000$_B$															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			COUNT												

Reserved	Bit 15-13	-	**保留**
COUNT	Bit 12-0	R	**端点 1 的 FIFO 中收到数据的字节数量**。 此值伴随 FIFO 中内容变化而变化。

29. 5. 2. 64 端点主机发送配置寄存器组（USB_TXxTYPE）

偏移地址	寄存器名称	复位值
0x11A	USB_TX1TYPE	00000000_B
0x12A	USB_TX2TYPE	00000000_B
0x13A	USB_TX3TYPE	00000000_B
0x14A	USB_TX4TYPE	00000000_B
0x15A	USB_TX5TYPE	00000000_B

端点主机发送配置寄存器组是一组 8 位寄存器，用来说明当前选中发送端点的目标端点号，传输协议，以及其运行速度。

USB_TX1TYPE

◆ 主机模式：

端点 1 主机发送配置寄存器（USB_TX1TYPE）							
偏移地址：$11A_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
SPEED		PROTOCOL		TEN			

SPEED	Bit 7-6	R/W	**目标设备的运行速度选择** 00：目标设备假定与 USB 控制器具有相同的连接速度。 01：高速 10：全速 11：低速
PROTOCOL	Bit 5-4	R/W	**协议** 00：控制 01：同步 10：批量 11：中断 必须软件配置此二位，用来选择发送端点需要的协议
TEN	Bit 3-0	R/W	**目标端点号** 此端点号必须软件配置，包含在设备枚举过程中返回 USB 控制器的描述符中。

29. 5. 2. 65 端点主机发送轮询间隔寄存器组（USB_TXxINTERVAL）

偏移地址	寄存器名称	复位值
0x11B	USB_TX1INTERVAL	00000000_B
0x12B	USB_TX2INTERVAL	00000000_B
0x13B	USB_TX3INTERVAL	00000000_B
0x14B	USB_TX4INTERVAL	00000000_B
0x15B	USB_TX5INTERVAL	00000000_B

端点主机发送间隔寄存器组是一组 8 位寄存器，为当前选中的发送端点的中断传输和同步传输定义了轮询间隔。为批量传输定义了接收到多少帧的 NAK 响应将超时。

USB_TX1INTERVAL

USB_TX1INTERVAL 寄存器的值定义了帧数，如下表：

传输类型	速度	有效值（m）	说明
中断传输	低速或全速	1~255	轮询间隔是 m 帧
	高速	1~16	轮询间隔是 $2^{(m-1)}$ microframes
同步传输	全速或高速	1~16	轮询间隔是 $2^{(m-1)}$ 帧/microframes
批量传输	全速或高速	2~16	NAK 限制是 $2^{(m-1)}$ 帧。值为 0 或 1 时将禁止 NAK 超时功能。

◆ 主机模式：

端点 1 主机发送轮询间隔寄存器（USB_TX1INTERVAL）

偏移地址：$11B_H$

复位值：00000000_B

7	6	5	4	3	2	1	0
TXPOLL/NAKLMT							

TXPOLL/NAKLMT	Bit 7-0	R/W	**发送轮询间隔/NAK 限制时间** 见上表

29. 5. 2. 66 端点主机接收配置寄存器组（USB_RXxTYPE）

偏移地址	寄存器名称	复位值
0x11C	USB_RX1TYPE	00000000_B
0x12C	USB_RX2TYPE	00000000_B
0x13C	USB_RX3TYPE	00000000_B
0x14C	USB_RX4TYPE	00000000_B
0x15C	USB_RX5TYPE	00000000_B

端点主机接收配置寄存器组是一组 8 位寄存器，用来说明当前选中接收端点的目标端点号，传输协议，以及其运行速度。

USB_RX1TYPE

◆ 主机模式：

端点 1 主机接收配置寄存器（USB_RX1TYPE）							
偏移地址：$11C_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
SPEED		PROTOCOL			TEN		

SPEED	Bit 7-6	R/W	**目标设备的运行速度选择** 00：目标设备假定与 USB 控制器具有相同的连接速度。 01：高速 10：全速 11：低速
PROTOCOL	Bit 5-4	R/W	**协议** 00：控制 01：同步 10：批量 11：中断 必须软件配置此二位，用来选择接收端点需要的协议
TEN	Bit 3-0	R/W	**目标端点号** 此端点号必须软件配置，包含在设备枚举过程中返回 USB 控制器的描述符中。

29. 5. 2. 67 端点主机接收轮询间隔寄存器组（USB_RXxINTERVAL）

偏移地址	寄存器名称	复位值
0x11D	USB_RX1INTERVAL	00000000_B
0x12D	USB_RX2INTERVAL	00000000_B
0x13D	USB_RX3INTERVAL	00000000_B
0x14D	USB_RX4INTERVAL	00000000_B
0x15D	USB_RX5INTERVAL	00000000_B

端点主机接收轮询间隔寄存器组是一组 8 位寄存器，为当前选中的接收端点的中断传输和同步传输定义了轮询间隔。为批量传输定义了接收到多少帧的 NAK 响应将超时。

USB_RX1INTERVAL

USB_RX1INTERVAL 寄存器的值定义了帧数，如下表：

传输类型	速度	有效值（m）	说明
中断传输	低速或全速	1~255	轮询间隔是 m 帧
	高速	1~16	轮询间隔是 $2^{(m-1)}$ microframes
同步传输	全速或高速	1~16	轮询间隔是 $2^{(m-1)}$ 帧/microframes
批量传输	全速或高速	2~16	NAK 限制是 $2^{(m-1)}$ 帧。值为 0 或 1 时将禁止 NAK 超时功能。

- 主机模式：

端点 1 主机接收轮询间隔寄存器（USB_RX1INTERVAL）							
偏移地址：$11D_H$							
复位值：00000000_B							
7	6	5	4	3	2	1	0
RXPOLL/NAKLMT							

RXPOLL/NAKLMT	Bit 7-0	R/W	**接收轮询间隔/NAK 限制时间** 见上表

29. 5. 2. 68 DMA中断寄存器（USB_DMA_INTR）

该寄存器为每一个 DMA 通道提供中断，该中断寄存器是读清零的。当该寄存器的任意位被置位了，输出 DMA_NINT 会变为低有效。可选 DMA 控制器章节中描述了引起中断置位的事件。该寄存器位仅在 DMA 相应通道的中断使能位使能时置 1（寄存器 DMA_CNTL.DMAIE）。

DMA 中断寄存器（USB_DMA_INTR）							
偏移地址：200_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved				CH4_DMA_INTR	CH3_DMA_INTR	CH2_DMA_INTR	CH1_DMA_INTR

Reserved	Bit 7-4	R	保留
CH4_DMA_INTR	Bit 3	R/W	**通道 4 DMA 中断**
CH3_DMA_INTR	Bit 2	R/W	**通道 3 DMA 中断**
CH2_DMA_INTR	Bit 1	R/W	**通道 2 DMA 中断**
CH1_DMA_INTR	Bit 0	R/W	**通道 1 DMA 中断**

29. 5. 2. 69 DMA通道控制寄存器组（USB_DMAx_CNTL）

偏移地址	寄存器名称	复位值
0x204	USB_DMA1_CNTL	00000000_B
0x214	USB_DMA2_CNTL	00000000_B
0x224	USB_DMA3_CNTL	00000000_B
0x234	USB_DMA4_CNTL	00000000_B

只有当 USB 控制器被配置为至少使用一个内部 DMA 通道时，该寄存器才可用。该寄存器为每个通道提供 DMA 传输控制。使能，传输方向，传输模式，DMA 突发模式都是由该寄存器控制的。

DMA 通道 1 控制寄存器（USB_DMA1_CNTL）

偏移地址：204_H

复位值：00000000_B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					DMA_BRSTM		DMA_ERR	DMAEP				DMAIE	DMAMODE	DMA_DIR	DMA_EN

Reserved	Bit 15-11	R	**保留**
DMA_BRSTM	Bit 10-9	R/W	**突发模式** 00 = 突发模式 0：未声明长度突发 01 = 突发模式 1：INCR4 或未声明长度 10 = 突发模式 2：INCR8，INCR4 或未声明长度 11 = 突发模式 3：INCR16，INCR8，INCR4 或未声明长度
DMA_ERR	Bit 8	R/W	**总线错误位。** 表示观察到总线错误。该位由软件清零。
DMAEP	Bit 7-4	R/W	指定此通道的端点号。
DMAIE	Bit 3	R/W	**DMA 中断使能。**
DMAMODE	Bit 2	R/W	**DMA 传输模式。** 0 = DMA 模式 0 传输 1 = DMA 模式 1 传输
DMA_DIR	Bit 1	R/W	**DMA 传输方向。** 0 = DMA 写（RX 端） 1 = DMA 读（TX 端）
DMA_EN	Bit 0	R/W	**该位使能 DMA 传输并开启传输。**

29. 5. 2. 70 DMA通道地址寄存器组（USB_DMAx_ADDR）

偏移地址	寄存器名称	复位值
0x208	USB_DMA1_ADDR	00000000_B
0x218	USB_DMA2_ADDR	00000000_B
0x228	USB_DMA3_ADDR	00000000_B
0x238	USB_DMA4_ADDR	00000000_B

该寄存器识别相应 DMA 通道的当前存储器地址。写入该寄存器的初始存储器地址必须是一个模 4 的值为 0 的数。也就是说，DMA_ADDR[1:0]必须为 2’b00。该寄存器的低 2 位只能读并且不能被软件置位。随着 DMA 传输的进展，存储器地址会随着字节的传输而增加。

DMA 通道 1 地址寄存器（USB_DMA1_ADDR）

偏移地址：208_H

复位值：00000000_B

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

DMA_ADDR	Bit 31-0	R/W	**DMA 存储器地址**

29. 5. 2. 71 DMA通道计数寄存器组（USB_DMAx_COUNT）

偏移地址	寄存器名称	复位值
0x20C	USB_DMA1_COUNT	00000000_B
0x21C	USB_DMA2_COUNT	00000000_B
0x22C	USB_DMA3_COUNT	00000000_B
0x23C	USB_DMA4_COUNT	00000000_B

该寄存器定义了传输的当前 DMA 计数值。软件将设置传输的初始计数值，该计数值定义了整个传输长度。在计数的进行过程中，随着字节的传输，该计数值将递减。

DMA 通道 1 计数寄存器（USB_DMA1_COUNT）

偏移地址：$20C_H$

复位值：00000000_B

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

DMA_COUNT	Bit 31-0	R/W	当前 **DMA** 计数值

29. 5. 2. 72 端点批量传输请求数据包数量寄存器组（USB_EPx_RQPKTCOUNT）

偏移地址	寄存器名称	复位值
0x304	USB_EP1_RQPKTCOUNT	00000000_00000000$_B$
0x308	USB_EP2_RQPKTCOUNT	00000000_00000000$_B$
0x30C	USB_EP3_RQPKTCOUNT	00000000_00000000$_B$
0x310	USB_EP4_RQPKTCOUNT	00000000_00000000$_B$
0x314	USB_EP5_RQPKTCOUNT	00000000_00000000$_B$

端点批量传输请求数据包数量寄存器组是一组 16 位寄存器，用于主机模式时说明批量传输中可以向接收端点传输的批量数据包的数量。USB 控制器使用此寄存器记录的值来决定寄存器 USB_RXxCSRH 中的 AUTOREQ 位置 1 时发起请求包的数量。
注意：多个包合并到单批量包中作为一个数据包。

USB_EP1_RQPKTCOUNT

- 主机模式：

端点 1 批量传输请求数据包数量寄存器（USB_EP1_RQPKTCOUNT）															
偏移地址：304_H															
复位值：00000000_00000000$_B$															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT															

COUNT	Bit 15-0	R/W	**批量传输中可以向接收端点 1 传输的批量数据包的数量。**

29. 5. 2. 73 接收双包缓冲禁用寄存器（USB_RXDPKTBUFDIS）

USB_RXDPKTBUFDIS 是一个 16 位寄存器，用来指示哪个接收端点禁用双包缓存功能。

- 主机模式/设备模式：

接收双包缓冲禁用寄存器（USB_RXDPKTBUFDIS）															
偏移地址：340_H															
复位值：$00000000\_00000000_B$															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										EP5	EP4	EP3	EP2	EP1	Reserved

Reserved	Bit 15-6	-	**保留**
EP5	Bit 5	R/W	**禁用 EP5 的接收双包缓存** 0：使用双包缓存。 1：禁用双包缓存。
EP4	Bit 4	R/W	**禁用 EP4 的接收双包缓存** 0：使用双包缓存。 1：禁用双包缓存。
EP3	Bit 3	R/W	**禁用 EP3 的接收双包缓存** 0：使用双包缓存。 1：禁用双包缓存。
EP2	Bit 2	R/W	**禁用 EP2 的接收双包缓存** 0：使用双包缓存。 1：禁用双包缓存。
EP1	Bit 1	R/W	**禁用 EP1 的接收双包缓存** 0：使用双包缓存。 1：禁用双包缓存。
Reserved	Bit 0	-	**保留**

29. 5. 2. 74 发送双包缓冲禁用寄存器（USB_TXDPKTBUFDIS）

USB_TXDPKTBUFDIS 是一个 16 位寄存器，用来指示哪个发送端点禁用双包缓存功能。

- 主机模式/设备模式：

发送双包缓冲禁用寄存器（USB_TXDPKTBUFDIS）															
偏移地址：342_H															
复位值：$00000000\_00000000_B$															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										EP5	EP4	EP3	EP2	EP1	Reserved

Reserved	Bit 15-6	-	**保留**
EP5	Bit 5	R/W	**禁用 EP5 的发送双包缓存** 0：使用双包缓存。 1：禁用双包缓存。
EP4	Bit 4	R/W	**禁用 EP4 的发送双包缓存** 0：使用双包缓存。 1：禁用双包缓存。
EP3	Bit 3	R/W	**禁用 EP3 的发送双包缓存** 0：使用双包缓存。 1：禁用双包缓存。
EP2	Bit 2	R/W	**禁用 EP2 的发送双包缓存** 0：使用双包缓存。 1：禁用双包缓存。
EP1	Bit 1	R/W	**禁用 EP1 的发送双包缓存** 0：使用双包缓存。 1：禁用双包缓存。
Reserved	Bit 0	-	**保留**

29. 5. 2. 75 LPM属性寄存器（USB_LPM_ATTR）

USB_LPM_ATTR 是一个 8 位寄存器，用来定义 LPM 事务和睡眠属性。在主机模式和设备模式下，它的值含义是一样的，但数据来源不同。

- 主机模式

  在主机模式下，软件将设置此寄存器中的值以定义将要传输的下一次 LPM 事务。

- 设备模式

  在设备模式下，此寄存器中的值包含在接受的最后一个 LPM 事务中。如果对最后一个 LPM 事务的响应是 ACK，就用此 LPM 事务中的内容来更新此寄存器。

LPM 属性寄存器（USB_LPM_ATTR）

偏移地址：360_H

复位值：$00000000\_00000000_B$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDPNT				Reserved			RMTWAK	HIRD				LINKSTATE			

ENDPNT	Bit 15-12	R	**LPM 事务令牌包端点**
Reserved	Bit 11-9	-	**保留**
RMTWAK	Bit 8	R	**远程唤醒使能位** 0：远程唤醒不使能 1：远程唤醒使能
HIRD	Bit 7-4	R	**主机启动恢复的持续时间**。 这个值是主机驱动在总线上的恢复最小时间。此寄存器的值对应实际恢复时间： 恢复时间 = 50μs+ HIRD*75。 恢复时间的范围是：50~1200μs。
LINKSTATE	Bit 3-0	R	**此值由主机提供给外部设备以指示在 LPM 事务的接收和接受之后外部设备必须切换到什么状态。** 0001：睡眠状态 其它值：保留

29. 5. 2. 76 LPM控制寄存器（USB_LPM_CNTRL）

USB_LPM_CNTRL 是一个 8 位寄存器，用来控制 LPM 事务。

- 主机模式：

LPM 控制寄存器（USB_LPM_CNTRL）							
偏移地址：362_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved						LPMRES	LPMXMT

Reserved	Bit 7-2	-	保留
LPMRES	Bit 1	R/W	此位由软件用于从睡眠状态进行唤醒。此位不同于寄存器 USB_POWER 中的 RESUME 位，RESUME 信号定时由硬件控制。软件写 LPMRES 位时，恢复时间将由 HIRD 指定。 LPMRES 位是自清零的。
LPMXMT	Bit 0	R/W	软件应将此位置 1 以发送 LPM 事务。该位自清零。该位将在接受到任何令牌或三次超时的情况下自清零。

◆ 设备模式：

LPM 控制寄存器（USB_LPM_CNTRL）							
偏移地址：362_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved			LPMNAK	LPMEN		LPMRES	LPMXMT

Reserved	Bit 7-5	-	保留
LPMNAK	Bit 4	R/W	该位用于将所有端点置于一种状态，使得对除 LPM 事务之外的其它事务都是响应 NAK。该位仅在已暂停 LPM 后才生效。这种情况下，将持续 NAK，直到该位由软件清零。
LPMEN	Bit 3-2	R/W	**用于使能 USB 控制器的 LPM。** 有以下三个级别： 00 和 10：不支持 LPM 和扩展事务。在这种情况下，USB 控制器不会响应 LPM 事务，事务将超时。 01：不支持 LPM，但支持扩展事务。在这种情况下，USB 控制器将使用 STALL 响应 LPM 事务。 11：支持 LPM 和扩展事务。
LPMRES	Bit 1	R/W	此位由软件用于从睡眠状态进行唤醒。此位不同于寄存器 USB_POWER 中的 RESUME 位，RESUME 信号定时由硬件控制。软件写 LPMRES 位时，恢复时间将由 HIRD 指定。 LPMRES 位是自清零的。
LPMXMT	Bit 0	R/W	此位由软件置 1，指示 USB 控制器在接收到下一个 LPM 事务时切换到睡眠状态。此位仅在 LPMEN 为 11B 时有效。如果没有数据待处理（所有 TX FIFO 为空），USB 控制器将以 ACK 响应。在这种情况下，该位将自清零，并产生软件中断。 如果有数据待处理（至少有一个 TX FIFO 不空），USB 控制器将回应一个 NYET。在这种情况下，该位不会自清零，将产生软件中断。

29. 5. 2. 77 LPM中断使能寄存器（USB_LPM_INTREN）

USB_LPM_INTREN 是一个 8 位寄存器，为 USB_LPM_INTR 寄存器中的中断提供使能位。

- 主机模式/设备模式：

LPM 中断使能寄存器（USB_LPM_INTREN）							
偏移地址：363_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved		LPMERREN	LPMRESEN	LPMNCEN	LPMACKEN	LPMNYEN	LPMSTEN

Reserved	Bit 7-6	-	保留
LPMERREN	Bit 5	R	**LPMERR 中断使能位：** 0：LPMERR 中断不使能。 1：LPMERR 中断使能。
LPMRESEN	Bit 4	R	**LPMRES 中断使能位：** 0：LPMRES 中断不使能。 1：LPMRES 中断使能。
LPMNCEN	Bit 3	R/W	**LPMNC 中断使能位：** 0：LPMNC 中断不使能。 1：LPMNC 中断使能。
LPMACKEN	Bit 2	R/W	**LPMACK 中断使能位：** 0：LPMACK 中断不使能。 1：LPMACK 中断使能。
LPMNYEN	Bit 1	R/W	**LPMNY 中断使能位：** 0：LPMNY 中断不使能。 1：LPMNY 中断使能。
LPMSTEN	Bit 0	R/W	**LPMST 中断使能位：** 0：LPMST 中断不使能。 1：LPMST 中断使能。

29. 5. 2. 78 LPM中断状态寄存器（USB_LPM_INTR）

USB_LPM_INTR 是一个 8 位寄存器，当其中一位置 1，相应使能位也置 1 时，将产生中断。

◆ 主机模式

LPM 中断状态寄存器（USB_LPM_INTR）							
偏移地址：364_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved		LPMERR	LPMRES	LPMNC	LPMACK	LPMNY	LPMST

Reserved	Bit 7-6	-	保留
LPMERR	Bit 5	R	接收到对 LPM 事务的响应有位填充错误或 PID 错误时，此位会被置 1。这种情况下，设备不会发生挂起，并且设备现在的状态是未知的。
LPMRES	Bit 4	R	当 USB 控制器由于任何原因恢复时，这一位和 USB_POWER 寄存器的 RESUME 位是互斥的。
LPMNC	Bit 3	R	当 LPM 事务已经发送，但未能完成时，此位将被置 1。
LPMACK	Bit 2	R	当发送 LPM 事务，并且设备以 ACK 响应时，该位置 1。
LPMNY	Bit 1	R	当发送 LPM 事务，并且设备以 NYET 响应时，该位置 1。
LPMST	Bit 0	R	当发送 LPM 事务，并且设备以 STALL 响应时，该位置 1。

◆ 设备模式

LPM 中断状态寄存器（USB_LPM_INTR）							
偏移地址：364_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved		LPMERR	LPMRES	LPMNC	LPMACK	LPMNY	LPMST

Reserved	Bit 7-6	-	保留
LPMERR	Bit 5	R	当接收到具有不是 LinkState 字段的 LPM 事务，此位将被置 1。在这种情况下，对事务的响应将是 STALL。但是，USB_LPM_ATTR 寄存器将被更新，以便软件可以观察不兼容的 LPM 数据包。
LPMRES	Bit 4	R	当 USB 控制器由于任何原因恢复时，这一位和 USB_POWER 寄存器的 RESUME 位是互斥的。
LPMNC	Bit 3	R	当 LPM 事务已经接收，此位将被置 1。USB 控制器以 NYET 响应 RX FIFO 中的数据。

LPMACK	Bit 2	R	当接收到 LPM 事务，并且 USB 控制器以 ACK 响应时，该位置 1。
LPMNY	Bit 1	R	当接收到 LPM 事务，并且 USB 控制器以 NYET 响应时，该位置 1。
LPMST	Bit 0	R	当接收到 LPM 事务，并且 USB 控制器以 STALL 响应时，该位置 1。

29. 5. 2. 79 LPM功能地址寄存器（USB_LPM_FADDR）

USB_LPM_FADDR 是一个 8 位寄存器，用于存放将被放置在 LPM 的有效数据的功能地址。

- 主机模式

LPM 功能地址寄存器（USB_LPM_FADDR）							
偏移地址：365_H							
复位值：00000000_B							
7	6	5	4	3	2	1	0
Reserved	LPMFADDR						

Reserved	Bit 7	-	保留
LPMFADDR	Bit 6-0	R/W	**LPM 数据功能地址**

第30章 扩展总线接口（EBI）

30. 1 概述

EBI 模块的主要用途为：

1）将 AHB 传输转换成合适的外部器件协议；

2）满足外部器件的访问时序要求。

所有的外部存储器都共用相同的地址，数据和控制信号。每个外部器件都通过一个独一无二的片选访问。EBI 模块仅一次访问一个外部器件。

30. 2 特性

- 具有静态存储器接口的器件包含：
  - 静态随机存储器 SRAM
  - 只读存储器 ROM
  - NOR Flash
  - PSRAM（3 个存储器组）
- NAND Flash 中的 2 组存储器支持 ECC 硬件，可检查高达 8K 字节数据
- 支持对同步器件的 Burst 模式访问（NOR Flash 和 PSRAM）
- 8 位或 16 位数据总线
- 每个存储器组都有独立片选控制，并可单独配置
- 具有可配置时序，能够支持各种器件：
  - 可编程等待状态（高达 15）
  - 可编程总线恢复周期（高达 15）
  - 可编程输出使能和写使能延时（高达 15）
  - 为了支持各种不同的存储器和时序，具有独立的写/读时序及协议
- PSRAM 和 SRAM 器件使用的写使能和字节选择输出
- 可将 32 位 AHB 传输转换成连续的 16 位或 8 位传输，对外部器件进行访问
- 支持 1 个写 FIFO，2 个字长，每个字 32 位，仅存储数据，不存储地址。因此，该 FIFO 只缓冲 AHB 写 burst 传输，可缓慢写入存储器，更快的释放 AHB。一次仅缓冲一个 burst：当运行过程中发生一个新的 AHB burst，FIFO 将被耗尽。EBI 模块将插入等待状态直到当前存储器访问结束。
- 外部异步等待协议
- EBI 寄存器定义了外部器件类型和相关的特性，尽管在任意时刻寄存器设置都可被更改，但通常用法是这些寄存器在 boot 时设置完成，在下一个复位或上电发生前，不要改变寄存器设置。

30. 3 结构框图

EBI 模块包含了 4 个主要模块：AHB 接口， NOR Flash/PSRAM 控制器，NAND Flash 控制器，和外部器件接口。

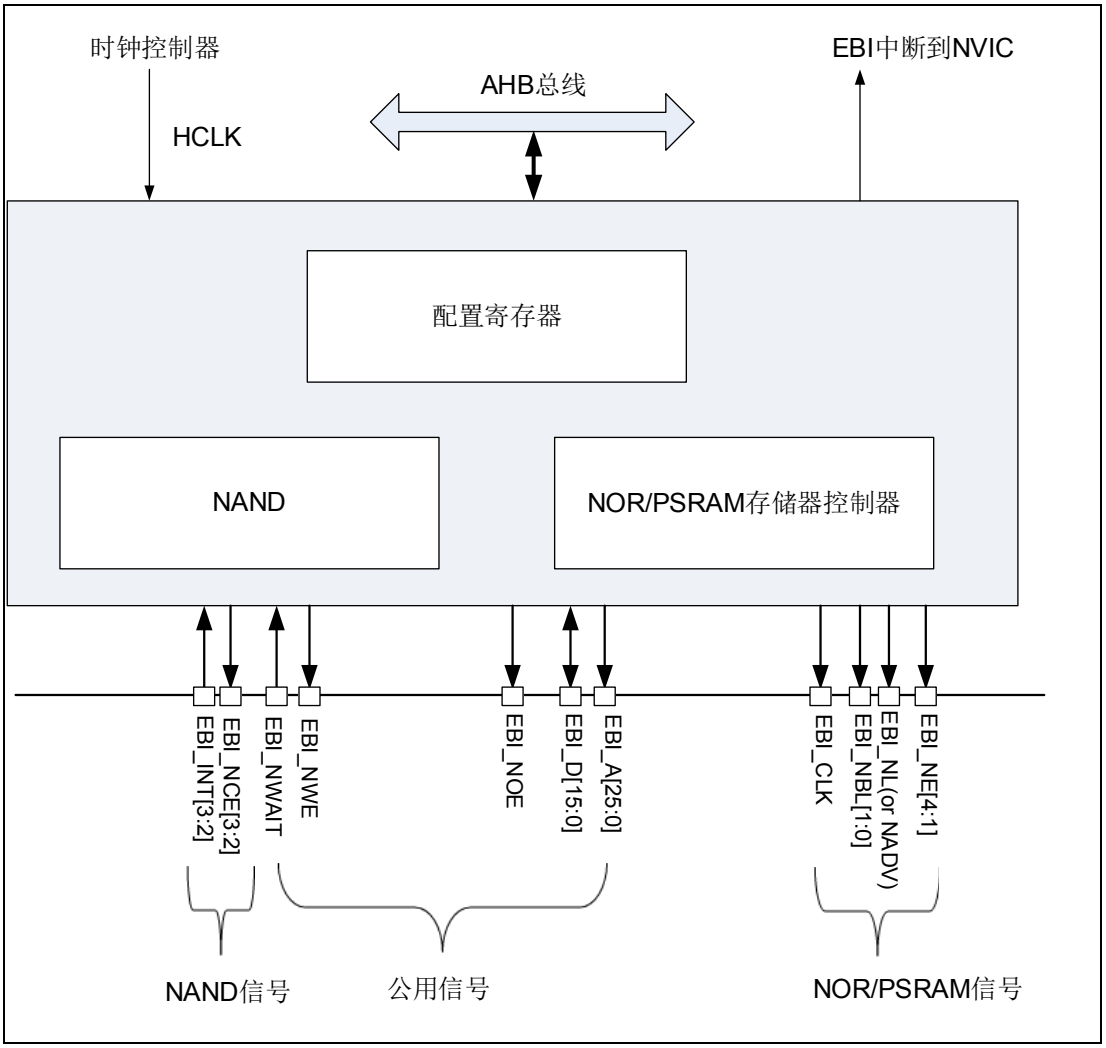


图 30-1 EBI 结构框图

30.4 功能描述

内部 CPU 和外部总线主外设可通过 AHB 从接口访问外部静态存储器。

AHB 传输将转换成外部器件协议。尤其是，如果外部存储器为 16 或 8 位宽，AHB 上的 32 位传输将分割成连续的 16 或 8 位进行访问。当分别进行 32 位对齐或不对齐的访问时，片选将保持低电平或在两个连续访问之间翻转。

发生以下情况时，EBI 会产生 AHB 错误：

◇ 对未使能的 EBI 组进行读或写操作

◇ 在 EBI_BCTRLRx 寄存器中的 FLASHACCEN 复位的情况下，对 NOR Flash 组进行读或写操作

AHB 时钟（HCLK）为 EBI 模块的参考时钟。

30.4.1 存储器和传输

30.4.1.1 通用传输规格

请求 AHB 传输的数据宽度可以为 8 位，16 位或 32 位，而访问的外部器件的数据宽度则是固定的，因此可造成传输不一致的问题。

执行下方的传输规则可保证传输的一致性：

◇ AHB 传输大小和存储器数据大小相同

◇ AHB 传输大小大于存储器大小。在这个情况下，EBI 将 AHB 传输分割成连续的较小数据宽度。

◇ AHB 传输小于存储器大小。异步数据传输有可能不一致，这取决于外部器件的类型。

- 异步访问具有字节选择功能的器件（SRAM，ROM，PSRAM）
  a) EBI 通过字节通道 NBL[1:0]允许写传输访问正确的数据
  b) 允许读传输。读取所有的存储器字节，丢弃掉无用的字节。读传输过程中，NBL[1:0]保持低电平。
- 异步访问不具有字节选择功能的器件（NOR 和 NAND Flash 16 位）
  a) 允许写传输
  b) 允许读传输。读取所有的存储器字节，丢弃掉无用的字节。读传输过程中，NBL[1:0]保持低电平。

30.4.1.2 寄存器配置

EBI 模块可通过寄存器组进行配置。NOR Flash/PSRAM 控制寄存器和 NAND Flash 寄存器的具体描述，请参考文中相关章节。

30.4.2 外部器件地址映射

以 EBI 的角度来看，外部存储器被分为 3 个固定大小的存储器组，每个存储器组为 256M 字节。

存储器组 1 可寻址高达 4 个 NOR Flash 或 PSRAM 存储器器件。存储器组 1 可再通过 4 个专门的片选分成 4 个 NOR/PSRAM 子组：NOR/PSRAM1，NOR/PSRAM2，NOR/PSRAM3 和 NOR/PSRAM4。

存储器组 2 和存储器组 3 用来寻址 NAND Flash 器件（一个组对应一个器件）。

每个组的存储器类型用户可在配置寄存器中进行自定义。

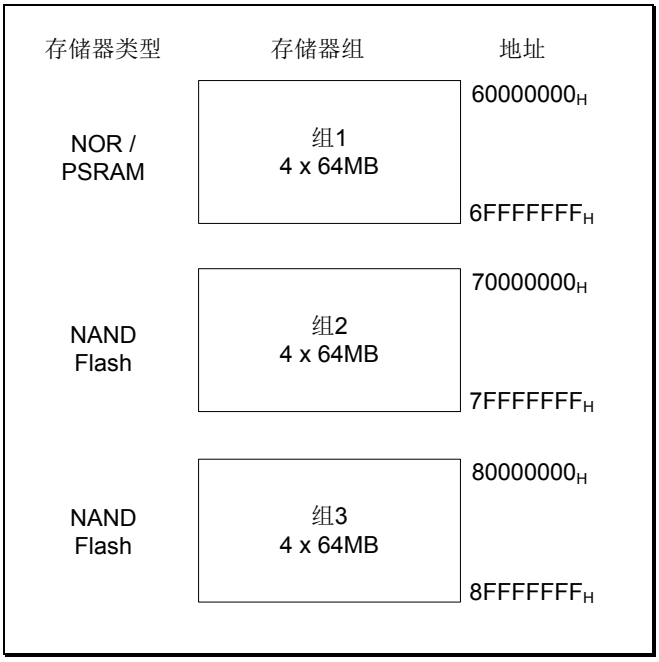


图 30-2 外部器件存储器地址映射

30. 4. 2. 1 NOR/PSRAM地址映射

HADDR[27:26]位用来选择 4 个存储器组中的 1 个。HADDR 为需要转换到外部存储器的内部 AHB 地址。

HADDR	存储器组
00	NOR/PSRAM1（存储器组 1）
01	NOR/PSRAM2（存储器组 1）
10	NOR/PSRAM3（存储器组 1）
11	NOR/PSRAM4（存储器组 1）

表 30-1 NOR/PSRAM 存储器组选择

HADDR[25:0]包含了外部存储器地址。HADDR 为字节地址，而存储器地址为字，所以实际发送到存储器的地址会根据存储器数据宽度而变化。注意，当外部存储器宽度为 16 位，EBI 会使用内部 HADDR[25:1]来为外部存储器 EBI_A[24:0]生成。无论外部存储器宽度是 16 还是 8 位，EBI_A[0]需要连接到外部存储器地址 A[0]。

存储器宽度	发送到存储器的数据地址	最大存储器容量（Bits）
8 位	HADDR[25:0]	64MByte x 8 = 512 Mbit
16 位	HADDR[25:1]>>1	64MByte/2 x 8 = 512 Mbit

表 30-2 外部存储器地址

不支持同步存储器 WRAP BURST 模式。存储器必须配置为线性 BURST 模式。

30. 4. 2. 2 NAND地址映射

2 个存储器组可用，每个组都可划分成几个存储器空间，如下表所示。

起始地址	结束地址	EBI 存储器组	存储器空间	时序寄存器
0x88000000	0x8BFFFFFF	组 3 – NAND Flash	属性	EBI_PATTR3 (0x8C)
0x80000000	0x83FFFFFF		通用	EBI_PMEMR3 (0x88)
0x78000000	0x7BFFFFFF	组 2 –NAND Flash	属性	EBI_PATTR2 (0x6C)
0x70000000	0x73FFFFFF		通用	EBI_PMEMR2 (0x68)

表 30-3 存储器映射和时序寄存器

对于 NAND Flash 存储器，通用和属性存储器空间又可在低 256 字节部分再分成 3 个区：

- ◇ 数据区（通用/属性存储器空间的第 1 个 64K 字节）
- ◇ 通用区（通用/属性存储器空间的第 2 个 64K 字节）
- ◇ 地址区（通用/属性存储器空间的余下的 128K 字节）

区	HADDR[17:6]	地址范围
地址区	1X	0x020000 – 0x03FFFF
命令区	01	0x010000 – 0x01FFFF
数据区	00	0x000000 – 0x0FFFF

表 30-4 NAND 组选择

应用软件使用这 2 个区来访问 NAND Flash 存储器：

- ◇ 向 NAND Flash 存储器发送命令：软件必须向命令区的任意地址写入命令。
- ◇ 读取或写入特定 NAND Flash 地址：软件必须向地址区的任意地址写入地址。由于地址长度为 4 或 5 个字节（取决于实际存储器数据宽度），需向地址区连续写才能输出完整地址。
- ◇ 读取或写入数据：软件向数据区的任意地址进行读数据或写数据。

由于 NAND Flash 存储器自动进行地址递增，连续访问存储器地址时无需递增数据区的地址。

30. 4. 3 NOR Flash/PSRAM控制器

EBI 模块可生成不同的信号时序来驱动以下存储器类型：

- ◇ 异步 SRAM 和 ROM（8/16/32 位）
- ◇ PSRAM（Cellular RAM）：支持异步模式，同步访问 BURST 模式
- ◇ NOR Flash：支持异步模式，同步访问 BURST 模式，复用/非复用模式

EBI 模块对每个存储器组输出为一个唯一的片选信号 NE[4:1]，其他所有信号（包括地址、数据和控制）均共享。

关于同步访问，EBI 仅在读/写操作过程中，向选中的外部器件发送时钟 CLK。该时钟的频率是 HCLK 时钟的整除因子。每个存储器组为固定 64M 字节。每个存储器由专门的寄存器配置。

可配置的存储器参数包括访问时序和等待管理支持（针对 BURST 模式下的 PSRAM 和 NOR Flash）。

参数	功能	访问模式	单位	最小值	最大值
地址建立	地址建立阶段的时间	异步	AHB 时钟周期（HCLK）	1	16
地址保持	地址保持阶段的时间	异步，复用 I/O	AHB 时钟周期（HCLK）	2	16
数据建立	数据建立阶段的时间	异步	AHB 时钟周期（HCLK）	2	256
总线恢复	总线恢复阶段的时间	异步和同步读/写	AHB 时钟周期（HCLK）	1	16
时钟分频	建立一个存储器时钟周期（CLK）所需的 AHB 时钟周期（HCLK）数	同步	AHB 时钟周期（HCLK）	2	16
数据延时	发送在第一个 BURST 数据前所需时钟周期数	同步	存储器时钟周期（CLK）	2	17

表 30-5 可配置 NOR/PSRAM 访问参数

30. 4. 3. 1 外部存储器接口信号

下面 3 张表格列出了与 NOR Flash，SRAM 和 PSRAM 相连接的典型信号。

EBI 信号名	I/O 类型	功能
CLK	输出	时钟（同步访问）
A[25:0]	输出	地址总线
D[15:0]	输入/输出	双向数据总线
NE[x]	输出	片选，x=1…4
NOE	输出	输出使能
NWE	输出	写使能
NL(=NADV)	输出	锁存使能（某些 NOR Flash 器件命名该信号地址有效，NADV）
NWAIT	输入	NOR Flash 要求 EBI 等待的输入信号

表 30-6 NOR Flash 非复用 I/O

EBI 信号名	I/O 类型	功能
CLK	输出	时钟（同步访问）
A[25:16]	输出	地址总线
D[15:0]	输入/输出	16 位复用，双向数据总线
NE[x]	输出	片选，x=1…4
NOE	输出	输出使能
NWE	输出	写使能
NL(=NADV)	输出	锁存使能（某些 NOR Flash 器件命名该信号地址有效，NADV）
NWAIT	输入	NOR Flash 要求 EBI 等待的输入信号

表 30-7 NOR Flash 复用 I/O

EBI 信号名	I/O 类型	功能
CLK	输出	时钟（PSRAM 同步访问）
A[25:0]	输出	地址总线
D[15:0]	输入/输出	双向数据总线
NE[x]	输出	片选，x=1…4
NOE	输出	输出使能
NWE	输出	写使能
NL(=NADV)	输出	PSRAM 数据有效地址（存储器信号：NDAV）
NWAIT	输入	NOR Flash 要求 EBI 等待的输入信号
NBL[1]	输出	高字节使能（存储器信号：NUB）
NBL[1]	输出	低字节使能（存储器信号：NLB）

表 30-8 PSRAM/SRAM 非复用 I/O

注：上述 3 个表中，带有前缀 N 的信号为低有效。

30. 4. 3. 2 支持的存储器和传输

下表列出了，当存储器数据总线为 16 位 NOR，PSRAM 和 SRAM 时，所支持的器件，访问模式和操作。

器件	模式	读/写	AHB 数据宽度	存储器数据宽度	是否支持	注释
NOR Flash（复用 I/O 和非复用 I/O）	异步	R	8	16	Y	
	异步	W	8	16	**N**	不支持
	异步	R	16	16	Y	
	异步	W	16	16	Y	
	异步	R	32	16	Y	分成 2 个 EBI 访问
	异步	W	32	16	Y	分成 2 个 EBI 访问
	异步页	R	-	16	**N**	不支持
	同步	R	8	16	**N**	不支持
	同步	R	16	16	Y	
	同步	R	32	16	Y	
PSRAM（复用 I/O 和非复用 I/O）	异步	R	8	16	Y	
	异步	W	8	16	Y	使用字节通道 NBL[1:0]
	异步	R	16	16	Y	
	异步	W	16	16	Y	
	异步	R	32	16	Y	分成 2 个 EBI 访问
	异步	W	32	16	Y	分成 2 个 EBI 访问
	异步页	R	-	16	**N**	不支持
	同步	R	8	16	**N**	不支持
	同步	R	16	16	Y	
	同步	R	32	16	Y	
	同步	W	8	16	Y	使用字节通道 NBL[1:0]
	同步	W	16/32	16	Y	
SRAM 和 ROM	异步	R	8/16	16	Y	
	异步	W	8/16	16	Y	使用字节通道 NBL[1:0]
	异步	R	32	16	Y	分成 2 个 EBI 访问
	异步	W	32	16	Y	分成 2 个 EBI 访问 使用字节通道 NBL[1:0]

表 30-9 NOR Flash/PSRAM 控制器：支持的存储器示例

30. 4. 3. 3 时序规则

信号同步

◇ 所有的控制器输出信号在内部时钟 HCLK 的上升沿变化

◇ 在同步模式下（读或写），所有输出信号在 HCLK 上升沿变化。

无论 CLKDIV 值为多少，所有输出变化如下：

- NOEL/NWEL/NEL/NADVL/NADVH/NBLL/地址有效输出在 EBI_CLK 时钟下降沿变化。
- NOEH/NWEH/NEH/NBLH 地址有效输出在 EBI_CLK 时钟上升沿变化。

30. 4. 3. 4 NOR Flash/RSRAM控制器同步传输

异步静态存储器（NOR Flash 存储器，PSRAM，SRAM）

◇ 信号由内部时钟 HCLK 同步，该时钟不发送给存储器。

◇ EBI 时钟在 NOE 信号无效前对数据采样。这么做可保证满足存储器数据保持时序约束（片选失效至数据失效的间隔，通常最小为 0ns）

◇ 如果扩展模式使能（EBI_BCTRLR 寄存器中的 EXTMODEN 置 1），可使用高达 4 个扩展模式（A,B,C 和 D）。读写操作时可混合使用着 4 个模式。例如，在 A 模式下进行读操作，在 B 模式下进行写操作。

◇ 如果扩展模式禁止（EBI_BCTRLR 寄存器中的 EXTMOD 复位），EBI 可在模式 1 或模式 2 下运行：

- 当选中 SRAM/PSRAM 存储器时（EBI_BCTRLRx 寄存器中的 MEMTYP [0:1]=0x0 或 0x01），模式 1 为默认模式
- 当选中 NOR 存储器时（EBI_BCTRLRx 寄存器中的 MEMTYP [0:1]=0x10），模式 2 为默认模式

模式 1 – SRAM/PSRAM（CRAM）

根据 EBI_BCTRLRx 和 EBI_BTRx/EBI_BWRTRx 寄存器的配置，下图展示了所支持模式下读写操作。

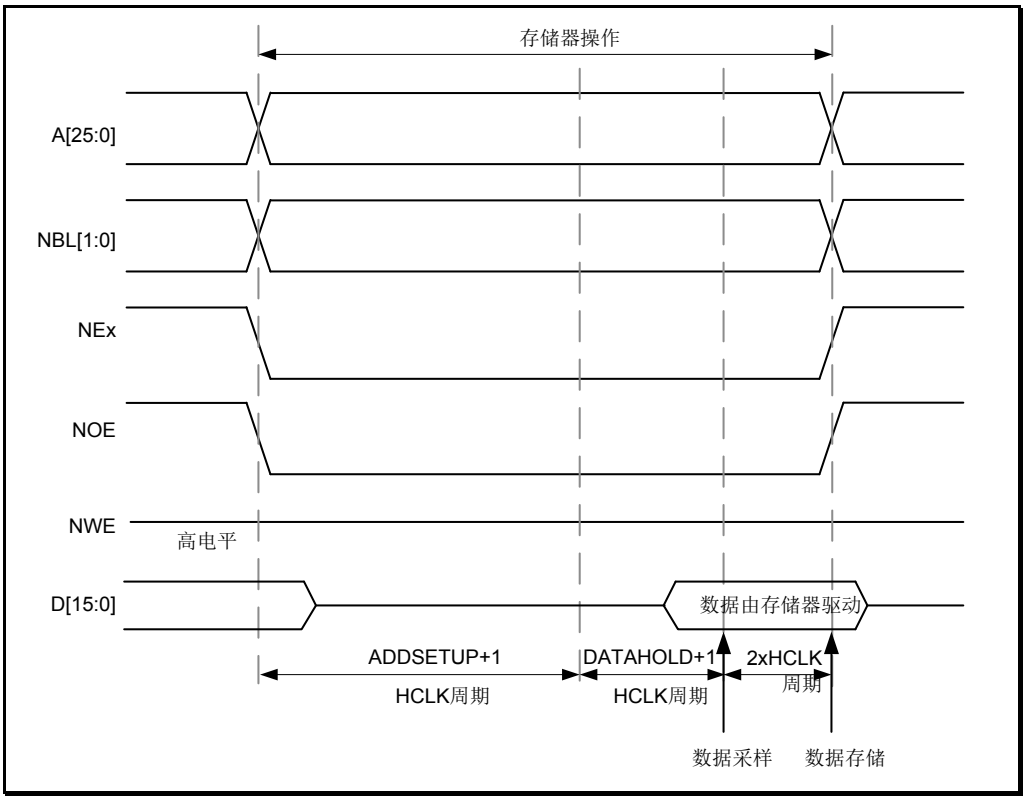


图 30-3　模式 1 读访问

注：在读访问过程中，NBL[1:0]为低电平。

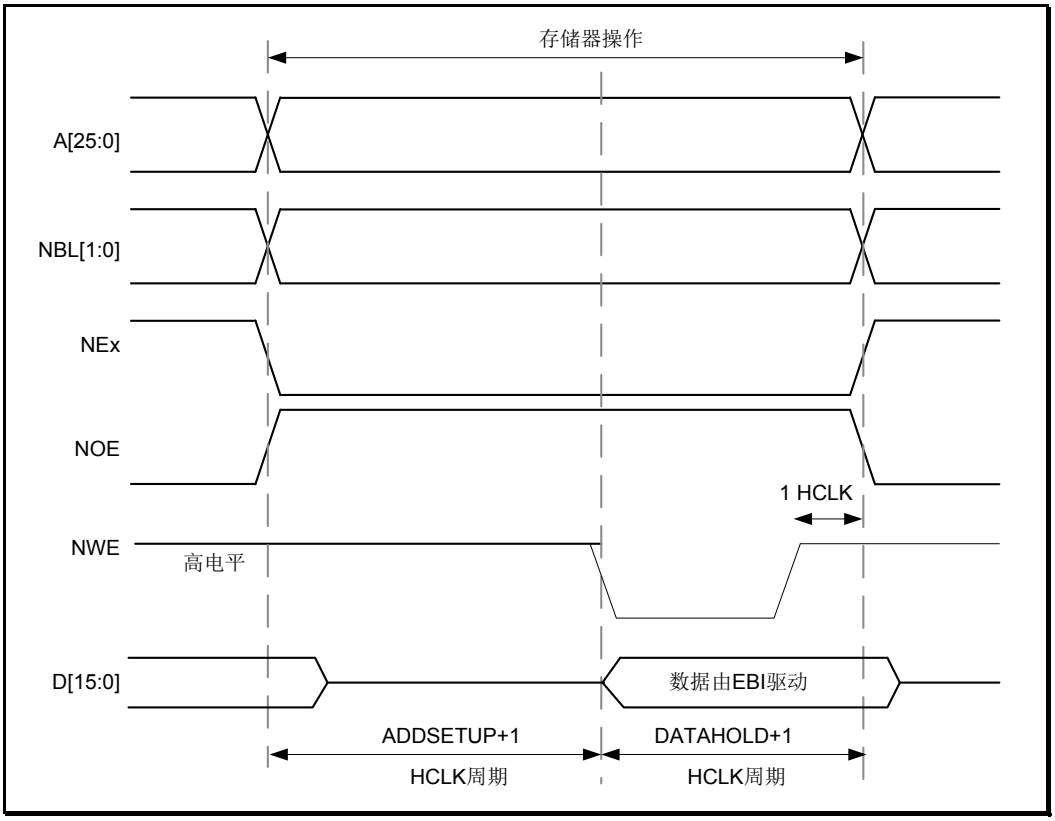


图 30-4 模式 1 写访问

写操作过程中最后的 1 个 HCLK 保证了在 NWE 上升沿过后地址和数据的保持时间。由于该 HCLK 周期的存在，DATAHOLD 的值必须大于 0。

Bit	位名	设置值
31-20	保留	0x000
19	RWCBURSTEN	0x0（对异步模式无效）
18-16	CPAGESIZE	0x0（对异步模式无效）
15	ASYNCWAIT	若存储器支持该功能，则置 1，否则保持 0
14	EXTMODEN	0x0
13	WAITEN	0x0（对异步模式无效）
12	WREN	如有需要，则设置
11	WAITCFG	无关位
10	WARPMOD	0x0
9	WAITPOL	仅当 Bit15 为 1 时，该位才有意义
8	BURSTEN	0x0
7	保留	0x1
6	FLASHACCEN	无关位
5-4	MEMWID	如有需要，则设置
3-2	MEMTYP[0:1]	如有需要，则设置，不包括 0x2（NOR Flash）
1	MUXE	0x0
0	MEMBKEN	0x1

表 30-10 EBI_BCTRLRx 寄存器设置

Bit	位名	设置值
31-30	保留	0x0
29-28	ACCMODE	无关位
27-24	DATALAT	无关位
23-20	CLKDIV	无关位
19-16	BUSTURN	NEx 高电平和 NEx 低电平之间的时间（BUSTURN HCLK）
15-8	DATAHOLD	第二次访问阶段的时间 写访问：（DATAHOLD+1）HCLK 周期 读访问：（DATAHOLD+3）HCLK 周期 该位不能设置为 0，最小值为 1
7-4	ADDHOLD	无关位
3-0	ADDSET[3:0]	第一次访问阶段的时间（ADDSET+1）HCLK 周期

表 30-11 EBI_BTRx 寄存器设置

模式 A – SRAM/PSRAM (CRAM) OE 翻转

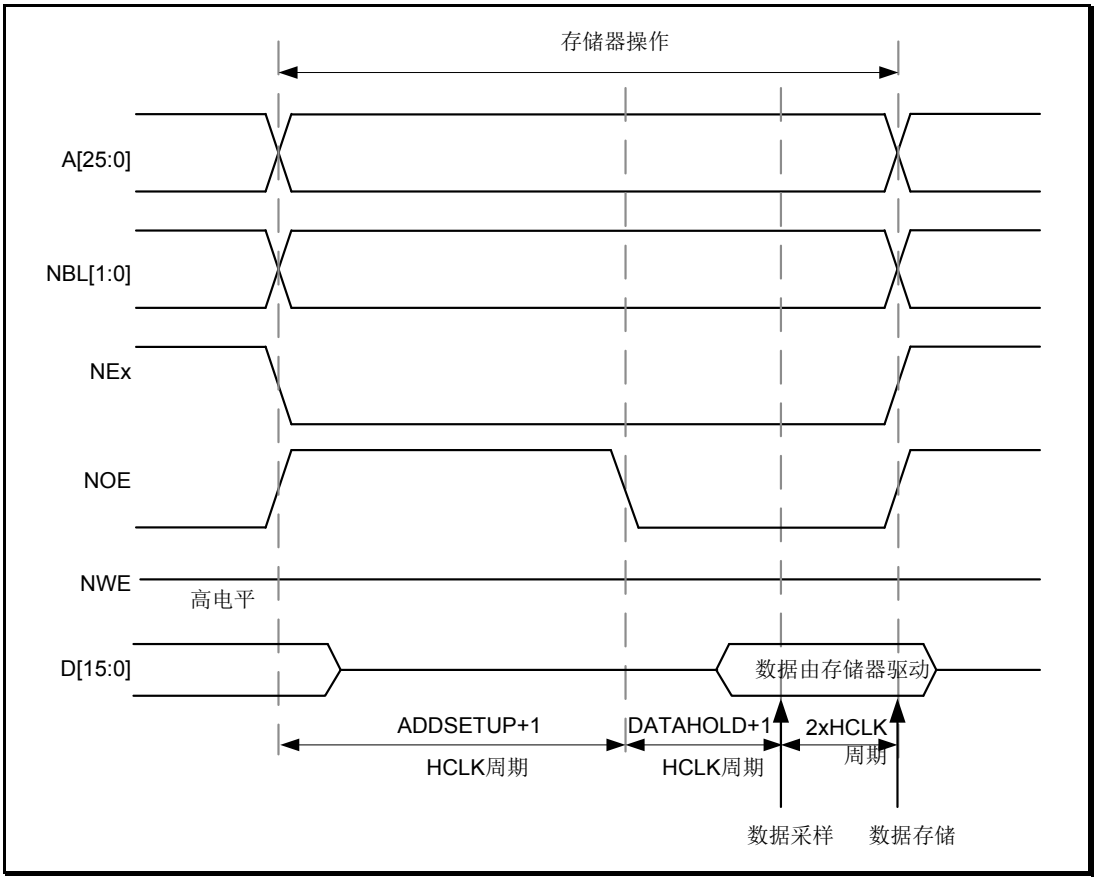


图 30-5 模式 A 读访问

注：在读访问过程中，NBL[1:0]为低电平。

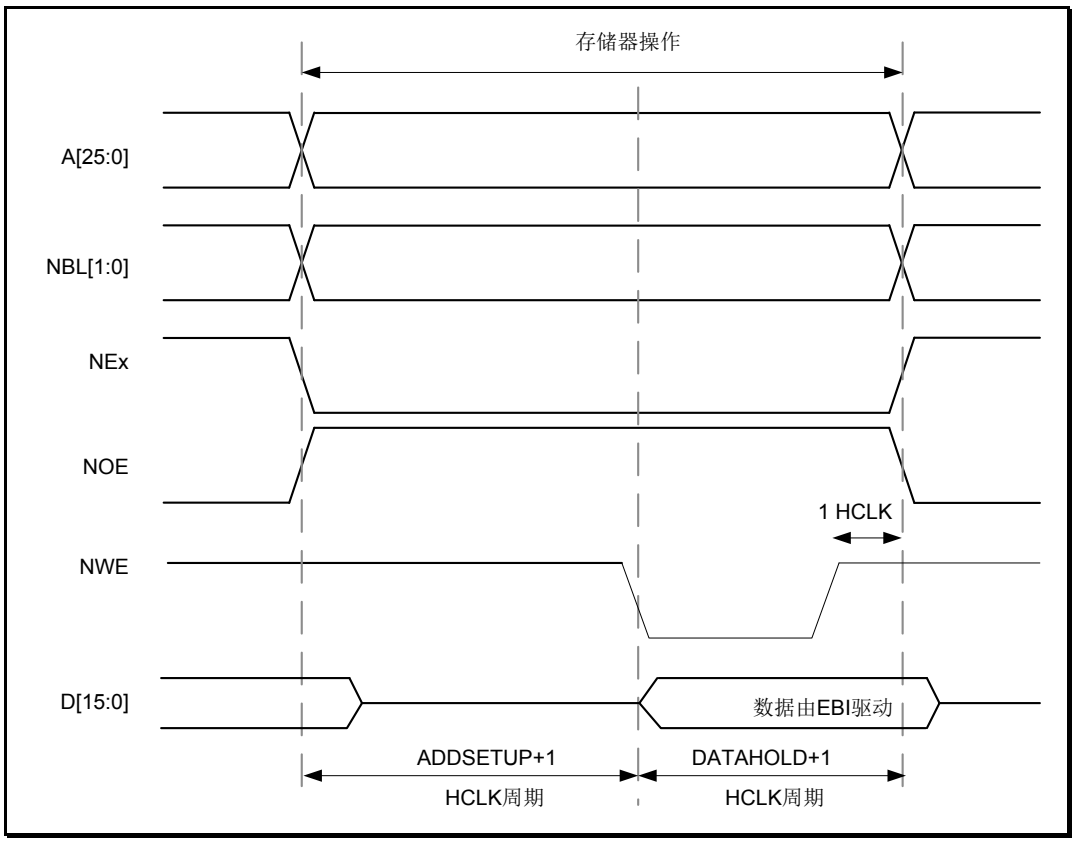


图 30-6 模式 A 写访问

模式 A 与模式 1 相比不同的是，NOE 的翻转及独立的读写时序。

Bit	位名	设置值
31-20	保留	0x000
19	RWCBURSTEN	0x0（对异步模式无效）
18-16	CPAGESIZE	0x0（对异步模式无效）
15	ASYNCWAIT	若存储器支持该功能，则置 1，否则保持 0
14	EXTMODEN	0x1
13	WAITEN	0x0（对异步模式无效）
13	WREN	如有需要，则设置
11	WAITCFG	无关位
10	WARPMOD	0x0
9	WAITPOL	仅当 Bit15 为 1 时，该位才有意义
8	BURSTEN	0x0
7	保留	0x1
6	FLASHACCEN	无关位
5-4	MEMWID	如有需要，则设置
3-2	MEMTYP [0:1]	如有需要，则设置，不包括 0x2（NOR Flash）
1	MUXE	0x0
0	MEMBKEN	0x1

表 30-12 EBI_BCTRLRx 寄存器设置

Bit	位名	设置值
31-30	保留	0x0
29-28	ACCMODE	0x0
27-24	DATALAT	无关位
23-20	CLKDIV	无关位
19-16	BUSTURN	NEx 高电平和 NEx 低电平之间的时间（BUSTURN HCLK）
15-8	DATAHOLD	第二次访问阶段的时间 读访问：（DATAHOLD+3）HCLK 周期 该位不能设置为 0，最小值为 1
7-4	ADDHOLD	无关位
3-0	ADDSET[3:0]	第一次读访问阶段的时间（ADDSET+1）HCLK 周期

表 30-13 EBI_BTRx 寄存器设置

Bit	位名	设置值
31-30	保留	0x0
29-28	ACCMODE	0x0
27-24	DATALAT	无关位
23-20	CLKDIV	无关位
19-16	BUSTURN	NEx 高电平和 NEx 低电平之间的时间（BUSTURN HCLK）
15-8	DATAHOLD	第二次访问阶段的时间 写访问：（DATAHOLD+1）HCLK 周期 读访问：（DATAHOLD+3）HCLK 周期 该位不能设置为 0，最小值为 1
7-4	ADDHOLD	无关位
3-0	ADDSET[3:0]	第一次写访问阶段的时间（ADDSET+1）HCLK 周期

表 30-14 EBI_BWRTRx 寄存器设置

Mode 2/B – NOR Flash

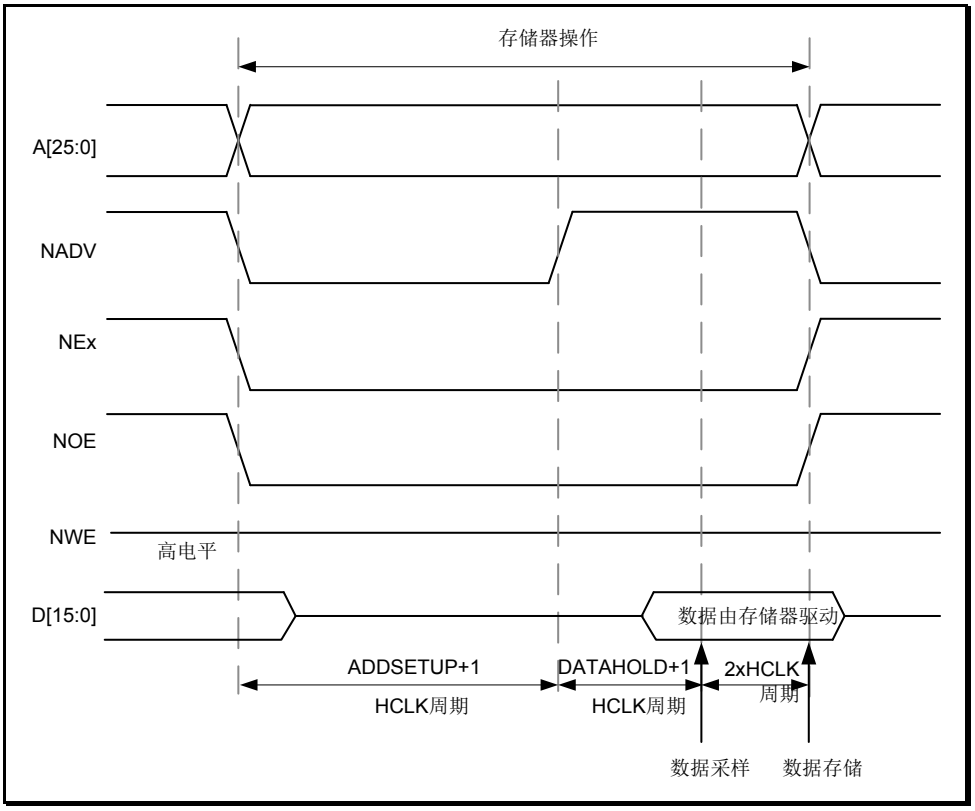


图 30-7 模式 2 和模式 B 读访问

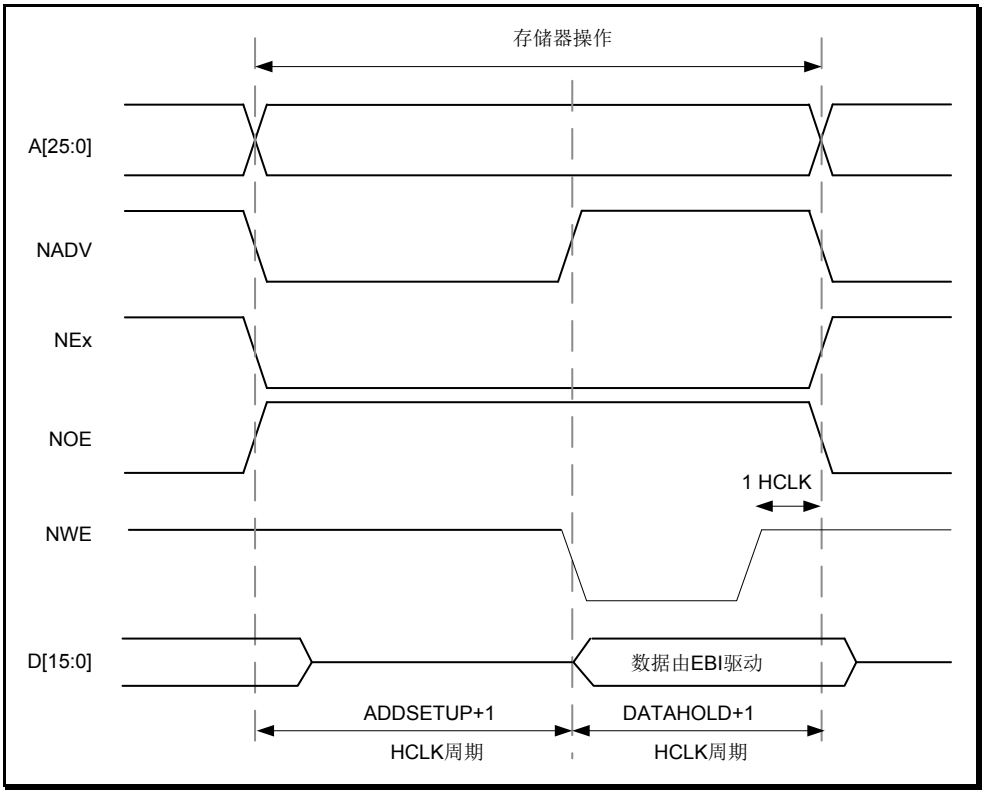


图 30-8 模式 2 写访问

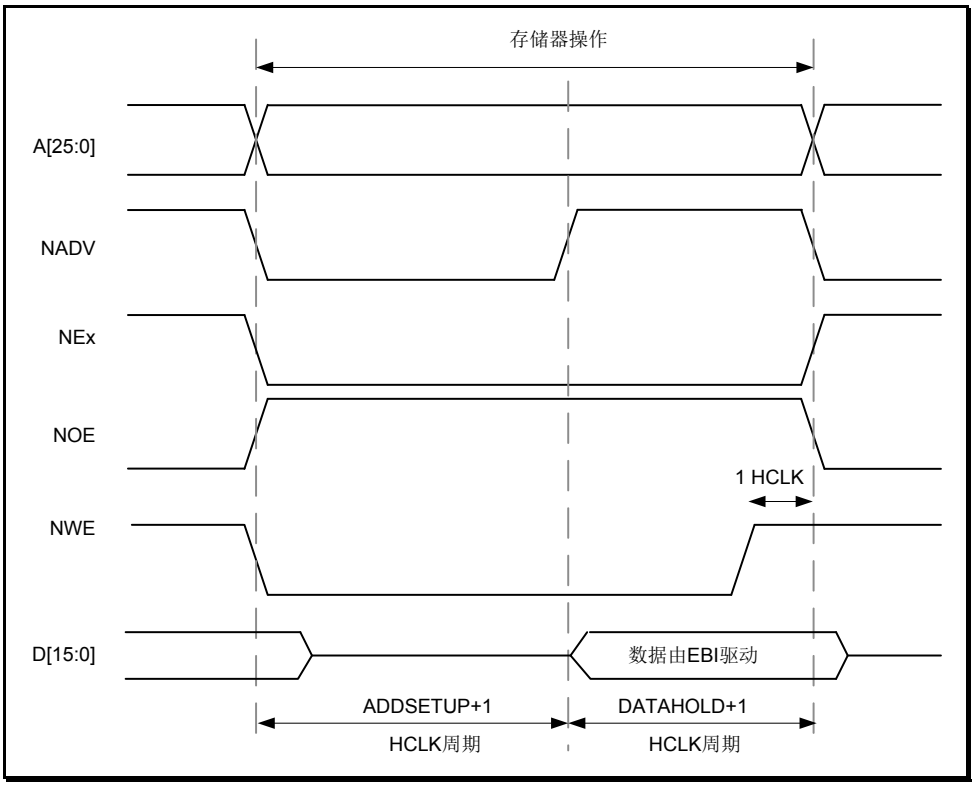


图 30-9 模式 B 写访问

模式 2/B 与模式 1 相比不同的是，NWE 的翻转及扩展模式 B 下读写时序相互独立。

Bit	位名	设置值
31-20	保留	0x000
19	RWCBURSTEN	0x0（对异步模式无效）
18-16	CPAGESIZE	0x0（对异步模式无效）
15	ASYNCWAIT	若存储器支持该功能，则置 1，否则保持 0
14	EXTMODEN	0x1
13	WAITEN	0x0（模式 2），0x1（模式 B）
13	WREN	0x0（对异步模式无效）
11	WAITCFG	无关位
10	WARPMOD	0x0
9	WAITPOL	仅当 Bit15 为 1 时，该位才有意义
8	BURSTEN	0x0
7	保留	0x1
6	FLASHACCEN	0x1
5-4	MEMWID	如有需要，则设置
3-2	MEMTYP [0:1]	0x2（NOR Flash）
1	MUXE	0x0
0	MEMBKEN	0x1

表 30-15 EBI_BCTRLRx 寄存器设置

Bit	位名	设置值
31-30	保留	0x0
29-28	ACCMODE	0x1
27-24	DATALAT	无关位
23-20	CLKDIV	无关位
19-16	BUSTURN	NEx 高电平和 NEx 低电平之间的时间（BUSTURN HCLK）
15-8	DATAHOLD	第二次访问阶段的时间 读访问：（DATAHOLD+3）HCLK 周期 该位不能设置为 0，最小值为 1
7-4	ADDHOLD	无关位
3-0	ADDSET[3:0]	第一次读访问阶段的时间（ADDSET+1）HCLK 周期

表 30-16 EBI_BTRx 寄存器设置

Bit	位名	设置值
31-30	保留	0x0
29-28	ACCMODE	0x1
27-24	DATALAT	无关位
23-20	CLKDIV	无关位
19-16	BUSTURN	NEx 高电平和 NEx 低电平之间的时间（BUSTURN HCLK）
15-8	DATAHOLD	第二次访问阶段的时间 写访问：（DATAHOLD+1）HCLK 周期 读访问：（DATAHOLD+3）HCLK 周期 该位不能设置为 0，最小值为 1
7-4	ADDHOLD	无关位
3-0	ADDSET[3:0]	第一次写访问阶段的时间（ADDSET+1）HCLK 周期

表 30-17 EBI_BWRTRx 寄存器设置

注：EBI_BWRTRx 寄存器仅在扩展模式 B 下有效，否则为无效。

模式 C – NOR Flash – OE 翻转

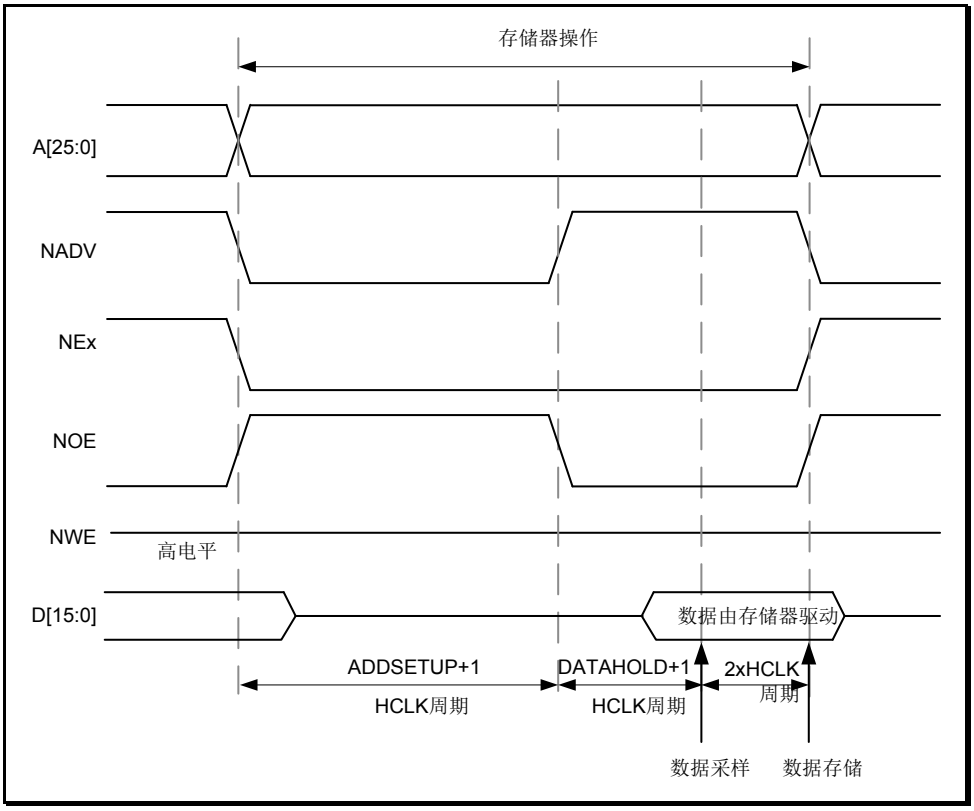


图 30-10 模式 C 读访问

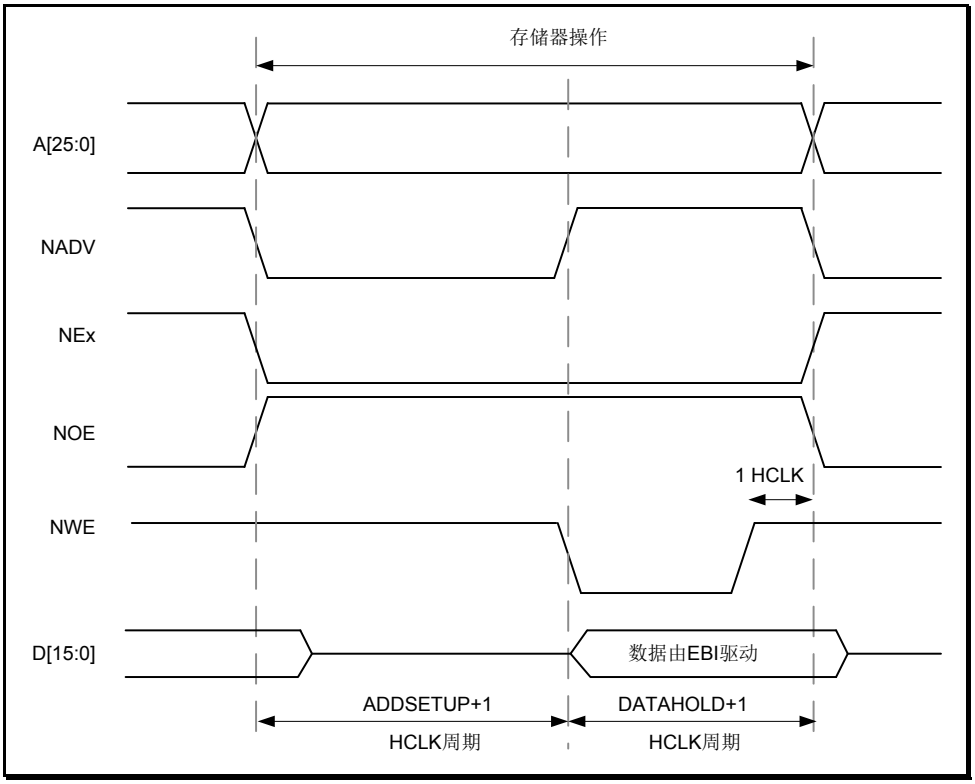


图 30-11 模式 C 写访问

模式 C 与模式 1 相比不同的是，NOE 的翻转及读写时序相互独立。

Bit	位名	设置值
31-20	保留	0x000
19	RWCBURSTEN	0x0（对异步模式无效）
18-16	CPAGESIZE	0x0（对异步模式无效）
15	ASYNCWAIT	若存储器支持该功能，则置 1，否则保持 0
14	EXTMODEN	0x1
13	WAITEN	0x0（对异步模式无效）
13	WREN	如有需要，则设置
11	WAITCFG	无关位
10	WARPMOD	0x0
9	WAITPOL	仅当 Bit15 为 1 时，该位才有意义
8	BURSTEN	0x0
7	保留	0x1
6	FLASHACCEN	0x1
5-4	MEMWID	如有需要，则设置
3-2	MEMTYP [0:1]	0x2（NOR Flash）
1	MUXE	0x0
0	MEMBKEN	0x1

表 30-18 EBI_BCTRLRx 寄存器设置

Bit	位名	设置值
31-30	保留	0x0
29-28	ACCMODE	0x2
27-24	DATALAT	0x0
23-20	CLKDIV	0x0
19-16	BUSTURN	NEx 高电平和 NEx 低电平之间的时间（BUSTURN HCLK）
15-8	DATAHOLD	第二次访问阶段的时间 读访问：（DATAHOLD+3）HCLK 周期 该位不能设置为 0，最小值为 1
7-4	ADDHOLD	无关位
3-0	ADDSET[3:0]	第一次读访问阶段的时间（ADDSET+1）HCLK 周期

表 30-19 EBI_BTRx 寄存器设置

Bit	位名	设置值
31-30	保留	0x0
29-28	ACCMODE	0x2
27-24	DATALAT	无关位
23-20	CLKDIV	无关位
19-16	BUSTURN	NEx 高电平和 NEx 低电平之间的时间（BUSTURN HCLK）
15-8	DATAHOLD	第二次访问阶段的时间 写访问：（DATAHOLD+1）HCLK 周期

Bit	位名	设置值
		读访问：（DATAHOLD+3）HCLK 周期 该位不能设置为 0，最小值为 1
7-4	ADDHOLD	无关位
3-0	ADDSET[3:0]	第一次写访问阶段的时间（ADDSET+1）HCLK 周期

表 30-20 EBI_BWRTRx 寄存器设置

模式 D – 扩展地址异步访问

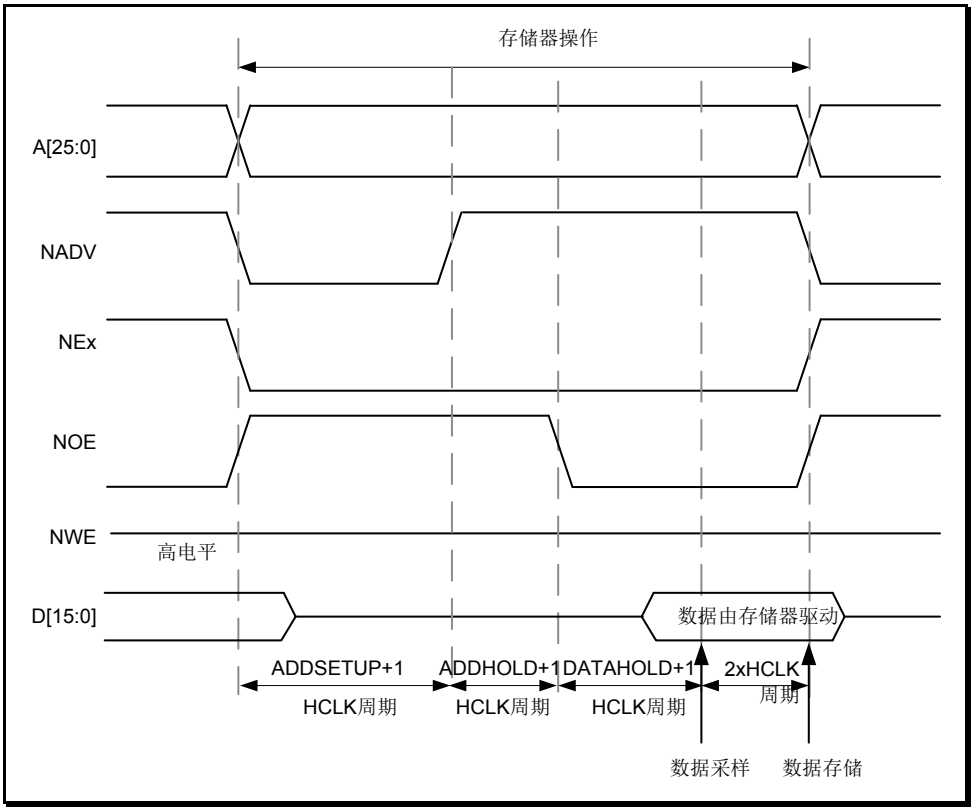


图 30-12 模式 D 读访问

模式 D 与模式 1 相比不同的是，NOE 的翻转在 NADV 变化后出现，还有读写的相互独立。

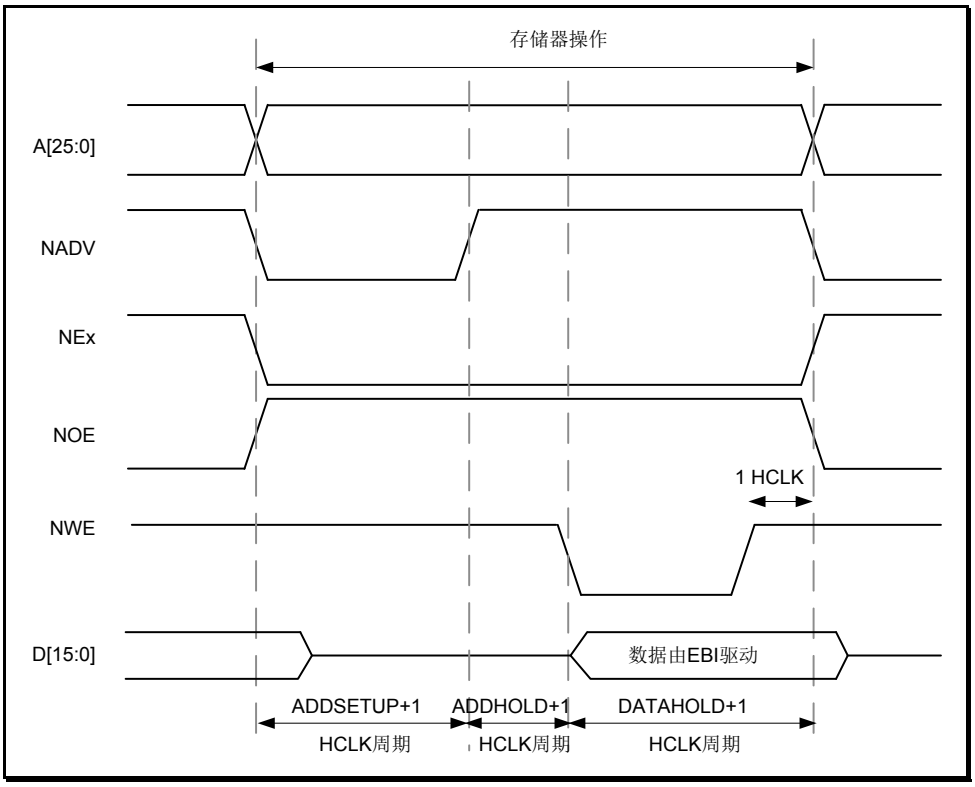


图 30-13 模式 D 写访问

Bit	位名	设置值
31-20	保留	0x000
19	RWCBURSTEN	0x0（对异步模式无效）
18-16	CPAGESIZE	0x0（对异步模式无效）
15	ASYNCWAIT	若存储器支持该功能，则置 1，否则保持 0
14	EXTMODEN	0x1
13	WAITEN	0x0（对异步模式无效）
13	WREN	如有需要，则设置
11	WAITCFG	无关位
10	WARPMOD	0x0
9	WAITPOL	仅当 Bit15 为 1 时，该位才有意义
8	BURSTEN	0x0
7	保留	0x1
6	FLASHACCEN	根据存储器来设置
5-4	MEMWID	如有需要，则设置
3-2	MEMTYP [0:1]	如有需要，则设置
1	MUXE	0x0
0	MEMBKEN	0x1

表 30-21 EBI_BCTRLRx 寄存器设置

Bit	位名	设置值
31-30	保留	0x0
29-28	ACCMODE	0x3
27-24	DATALAT	无关位
23-20	CLKDIV	无关位
19-16	BUSTURN	NEx 高电平和 NEx 低电平之间的时间（BUSTURN HCLK）
15-8	DATAHOLD	第二次访问阶段的时间 读访问：（DATAHOLD+3）HCLK 周期 该位不能设置为 0，最小值为 1
7-4	ADDHOLD	读访问的中间阶段的时间（ADDHOLD+1）HCLK 周期
3-0	ADDSET[3:0]	第一次读访问阶段的时间（ADDSET+1）HCLK 周期

表 30-22 EBI_BTRx 寄存器设置

Bit	位名	设置值
31-30	保留	0x0
29-28	ACCMODE	0x3
27-24	DATALAT	0x0
23-20	CLKDIV	0x0
19-16	BUSTURN	NEx 高电平和 NEx 低电平之间的时间（BUSTURN HCLK）
15-8	DATAHOLD	第二次访问阶段的时间 写访问：（DATAHOLD+3）HCLK 周期 该位不能设置为 0，最小值为 1
7-4	ADDHOLD	写访问的中间阶段的时间（ADDHOLD+1）HCLK 周期
3-0	ADDSET[3:0]	第一次写访问阶段的时间（ADDSET+1）HCLK 周期

表 30-23 EBI_BWRTRx 寄存器设置

复用模式 – 复用异步访问 NOR Flash 存储器

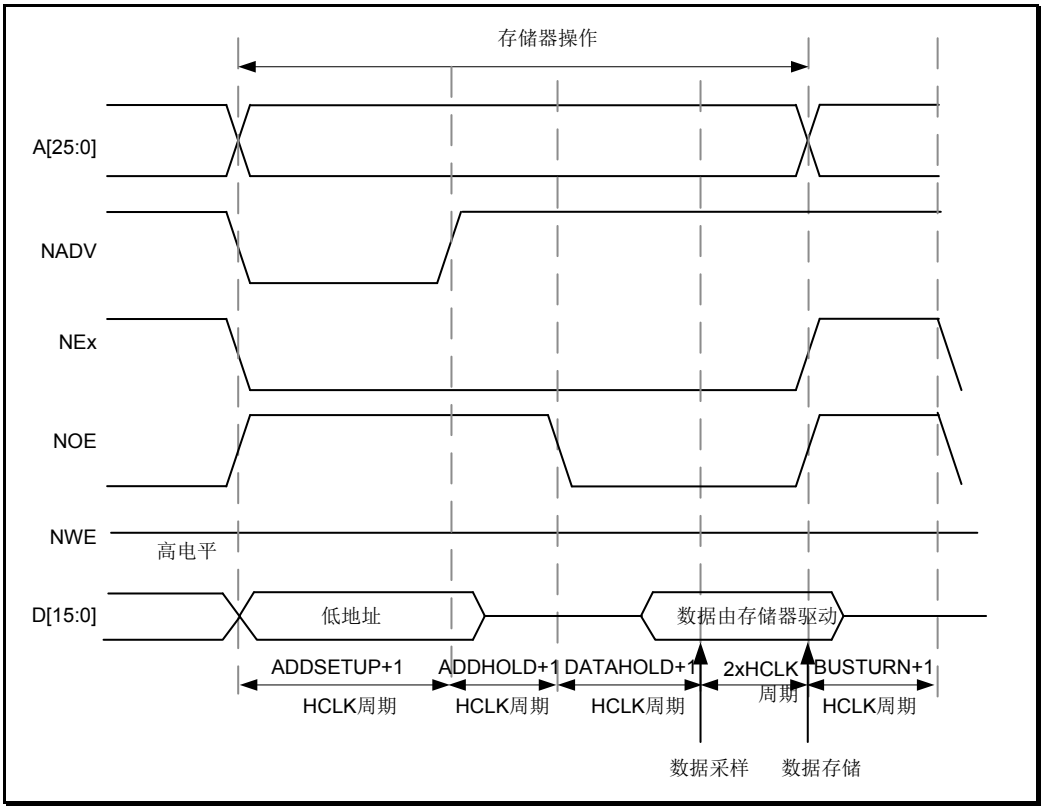


图 30-14 模式 D 读访问

总线恢复延时（BUSTURN+1）和连续 2 次操作之间的延时有重叠，所有当 BUSTURN ≤5 时无影响。

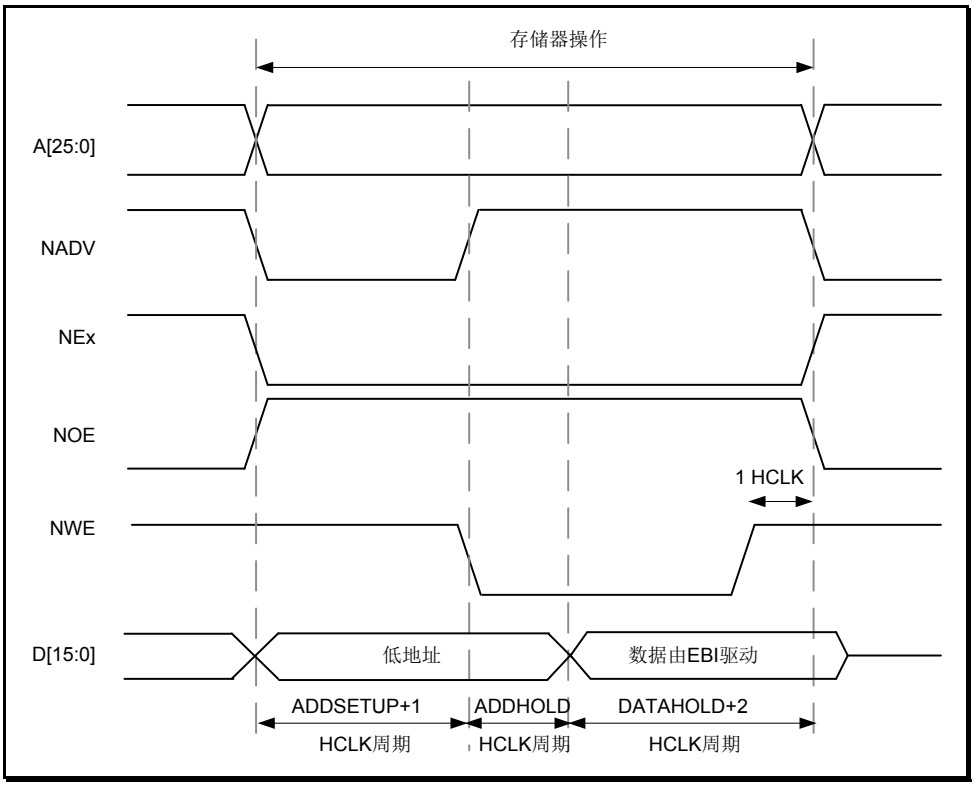


图 30-15　模式 D 写访问

模式 D 与模式 1 相比不同的是，低地址出现在数据总线上。

Bit	位名	设置值
31-20	保留	0x000
19	RWCBURSTEN	0x0（对异步模式无效）
18-16	CPAGESIZE	0x0（对异步模式无效）
15	ASYNCWAIT	若存储器支持该功能，则置 1，否则保持 0
14	EXTMODEN	0x0
13	WAITEN	0x0（对异步模式无效）
13	WREN	如有需要，则设置
11	WAITCFG	无关位
10	WARPMOD	0x0
9	WAITPOL	仅当 Bit15 为 1 时，该位才有意义
8	BURSTEN	0x0
7	保留	0x1
6	FLASHACCEN	0x1
5-4	MEMWID	如有需要，则设置
3-2	MEMTYP [0:1]	0x2（NOR Flash 存储器）
1	MUXE	0x1
0	MEMBKEN	0x1

表 30-24　EBI_BCTRLRx 寄存器设置

Bit	位名	设置值
31-30	保留	0x0
29-28	ACCMODE	0x3
27-24	DATALAT	无关位
23-20	CLKDIV	无关位
19-16	BUSTURN	NEx 高电平和 NEx 低电平之间的时间（BUSTURN HCLK）
15-8	DATAHOLD	第二次访问阶段的时间 读访问：（DATAHOLD+3）HCLK 周期 该位不能设置为 0，最小值为 1
7-4	ADDHOLD	读访问的中间阶段的时间（ADDHOLD+1）HCLK 周期
3-0	ADDSET[3:0]	第一次读访问阶段的时间（ADDSET+1）HCLK 周期

表 30-25 EBI_BTRx 寄存器设置

Bit	位名	设置值
31-30	保留	0x0
29-28	ACCMODE	0x0
27-24	DATALAT	无关位
23-20	CLKDIV	无关位
19-16	BUSTURN	NEx 高电平和 NEx 低电平之间的时间（BUSTURN HCLK）
15-8	DATAHOLD	第二次访问阶段的时间 写访问：（DATAHOLD+1）HCLK 周期 读访问：（DATAHOLD+3）HCLK 周期 该位不能设置为 0，最小值为 1
7-4	ADDHOLD	访问的中间阶段的时间（ADDHOLD+1）HCLK 周期 该位不能设置为 0，最小值为 1
3-0	ADDSET[3:0]	第一次访问阶段的时间（ADDSET+1）HCLK 周期

表 30-26 EBI_BWRTRx 寄存器设置

异步访问中等待管理

如果异步存储器将等待信号设置为有效，表明存储器还未准备好接受或提供数据，EBI_BCTRLRx 寄存中的 ASYNWAIT 需置 1。

如果等待信号有效（高有效或低有效取决于 WAITPOL 位），延长第二个访问阶段（数据建立阶段，由 DATAHOLD 设置）直到等待信号变为无效。不同于数据建立阶段，第一个访问阶段（地址建立和地址保持阶段，由 ADDSET[3:0]和 ADDHOLD 设置），不会受等待信号影响，因此不会延长。

为使等待信号满足以下条件，数据建立阶段（EBI_BTRx 寄存器中的 DATAHOLD）需按要求设置：

◇ 读访问：等待信号可在数据采样前 4 个周期或在 NOE 变为无效前的 6 个 HCLK 周期检测到。

◇ 写访问：等待信号可在 NWE 变为无效前 4 个周期检测到。

下面两张图显示了在异步存储器释放等待信号后，访问存储器时需要添加的 HCLK 时钟

周期数。

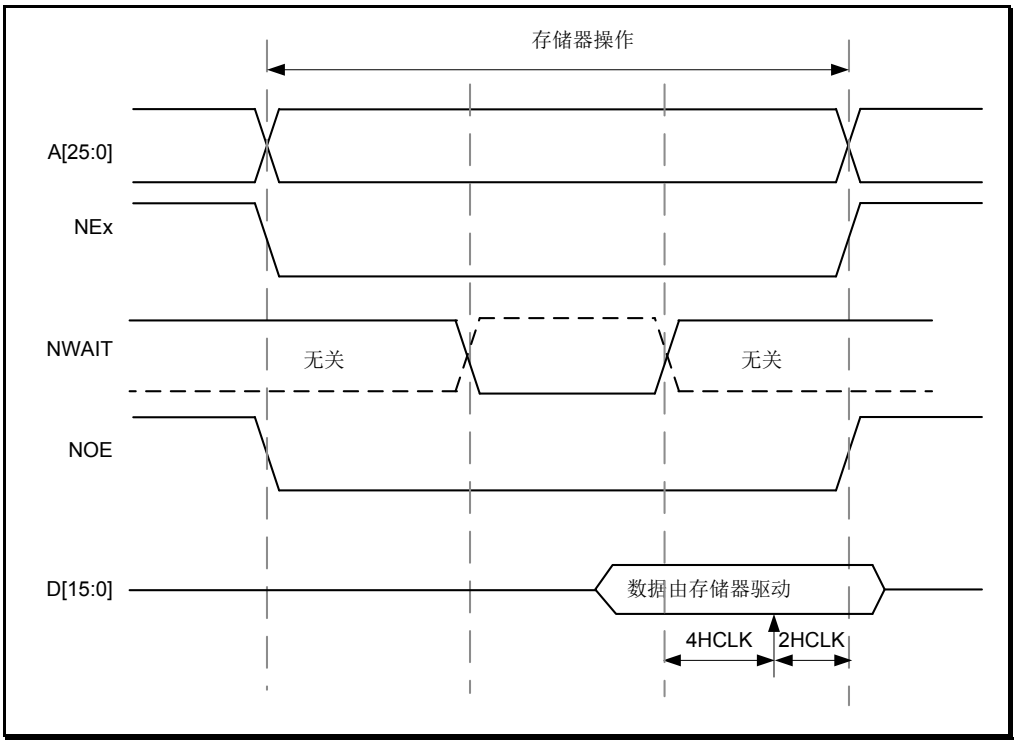


图 30-16　读访问过程中的异步等待

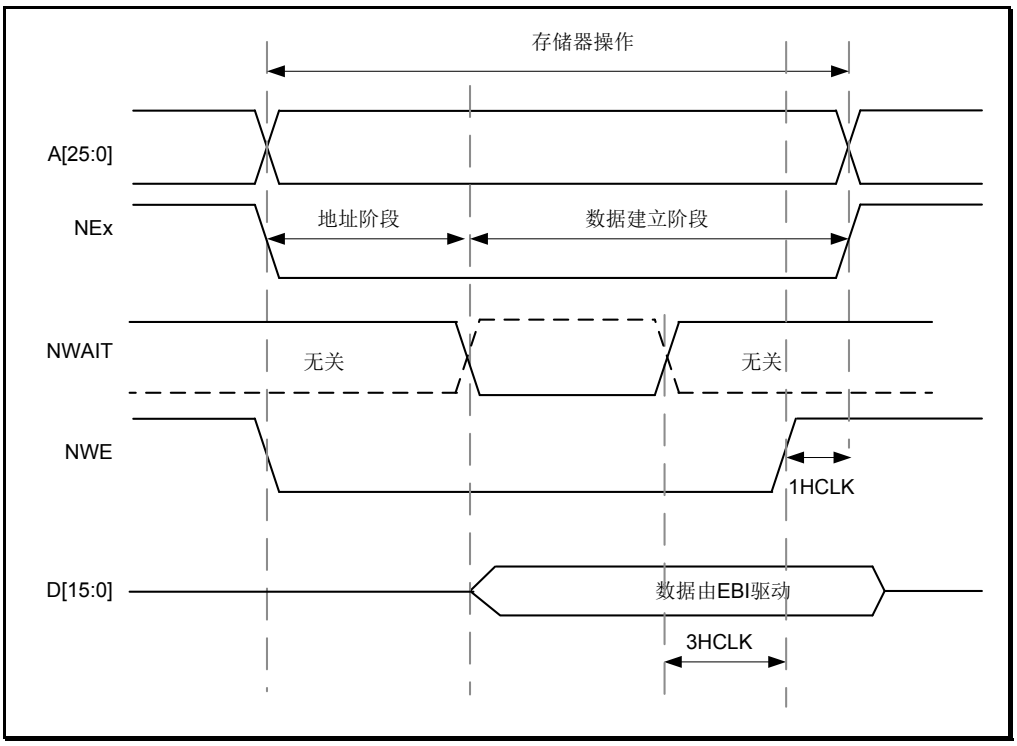


图 30-17　写访问过程中的异步等待

注：NWAIT 极性取决月 EBI_BCTRLRx 寄存器中的 WAITPOL 位。

30. 4. 3. 5 同步传输

存储器时钟 EBI_CLK 是 HCLK 的整除因子，由 CLKDIV 参数值决定。

NOR Flash 存储器定义了从 NADV 有效到 CLK 高电平之间的最小时间。为满足该约束，EBI 在同步访问的第一个内部时钟周期内（NADV 变为有效前）不会向存储器发送时钟，确保了存储器时钟的上升沿发生在 NADV 低脉冲的中间。

数据延时 VS NOR Flash 延时

数据延时是指数据采样前需要等待的周期数。DATALAT 值必须与 NOR Flash 配置寄存器中设置的延时值一致。当 NADV 在数据延时期间为低电平，则 EBI 模块不会将这些时钟周期包含在数据延时中。

需要注意的是，在数据延时期间，某些 NOR Flash 存储器包含 NADV 低电平周期，所以 NOR Flash 延时和 EBI DATALAT 参数之间确切的关系可以为下面任意一种：

◇ NOR Flash 延时 = (DATALAT + 2) CLK 时钟周期

◇ NOR Flash 延时 = (DATALAT + 3) CLK 时钟周期

延时阶段期间，某些存储器将会将 NWAIT 置为有效。在这种情况下，DATALAT 可被设置为最小值。如果数据有效，EBI 会对这些数据进行采样，并长时间等待后才进行评估。因此，当存储器结束延时后，EBI 才会检测并获取真实数据。

而有些存储器在延时期间不会将 NWAIT 置为有效。在这种情况下，EBI 和存储器的延时必须准确设置，否则无效数据会被误认为是有效数据，或在最初阶段造成有效数据丢失。

Single-burst 传输

同步访问时，当选中的存储器组配置为 BURST 模式，如果 AHB 需要传输 16 位数据，要求使用 AHB single-burst 传输，则 EBI 执行长度为 1 的 burst 传输（如果 AHB 传输为 16 位）或执行长度为 2 的 burst 传输（如果 AHB 传输为 32 位），当最后一个数据存好后，片选置为无效。很明显，这种传输方式不是最有效的（对比异步读访问而言）。但是，一个随机异步访问需要重新设置存储器访问模式，会导致用时更久。

Cellular RAM 1.5 跨页

Cellular RAM 1.5 不允许进行跨页 BURST 访问。当存储器页容量达到了 EBI_BCTRLR1 寄存器中 CPAGESIZE 的设置值，EBI 控制器会自动划分开 BURST 访问。

等待管理

对于同步 NOR Flash 存储器，在设置的延时时间：（DATALT+2）CLK 时钟周期后，需检测 NWAIT 信号。

如果检测到 NWAIT 为有效（WAITPOL=0 时为低有效，WAITPOL=1 时为高有效），插入等待状态直到 NWAIT 被检测到无效（WAITPOL=0 时为高无效，WAITPOL=1 时为低无效）。

当 NWAIT 为无效，数据即刻（WAITCFG=1）或在下个时钟边沿（WAITCFG=1）被认为有效。

在等待状态期间，控制器继续向存储器发送时钟脉冲，保持片选和输出使能信号有效，在这期间，数据认为是无效数据。

BURST 模式下，NOR Flash NWAIT 信号有 2 种时序配置：

◇ 在等待前一个数据周期，Flash 存储器将 NWAIT 设置为有效（复位后的默认设置）

◇ 在等待状态期间，Flash 存储器将 NWAIT 设置为有效。

通过 EBI_BCTRLRx 寄存中的 WAITCFG 位，EBI 在每个片选上都支持以上两种 NOR Flash 等待状态配置。

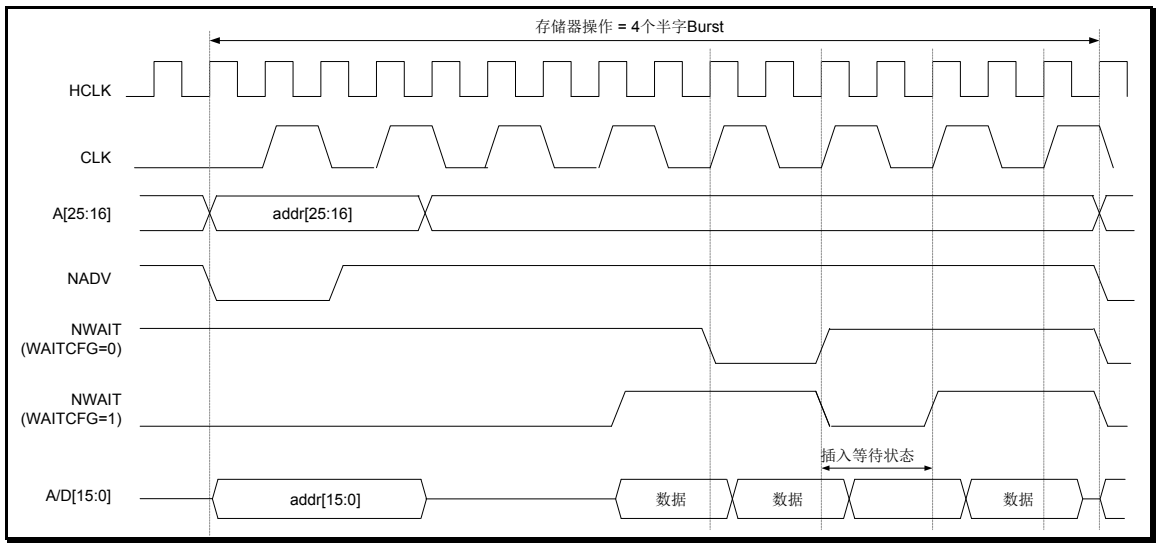


图 30-18 等待配置

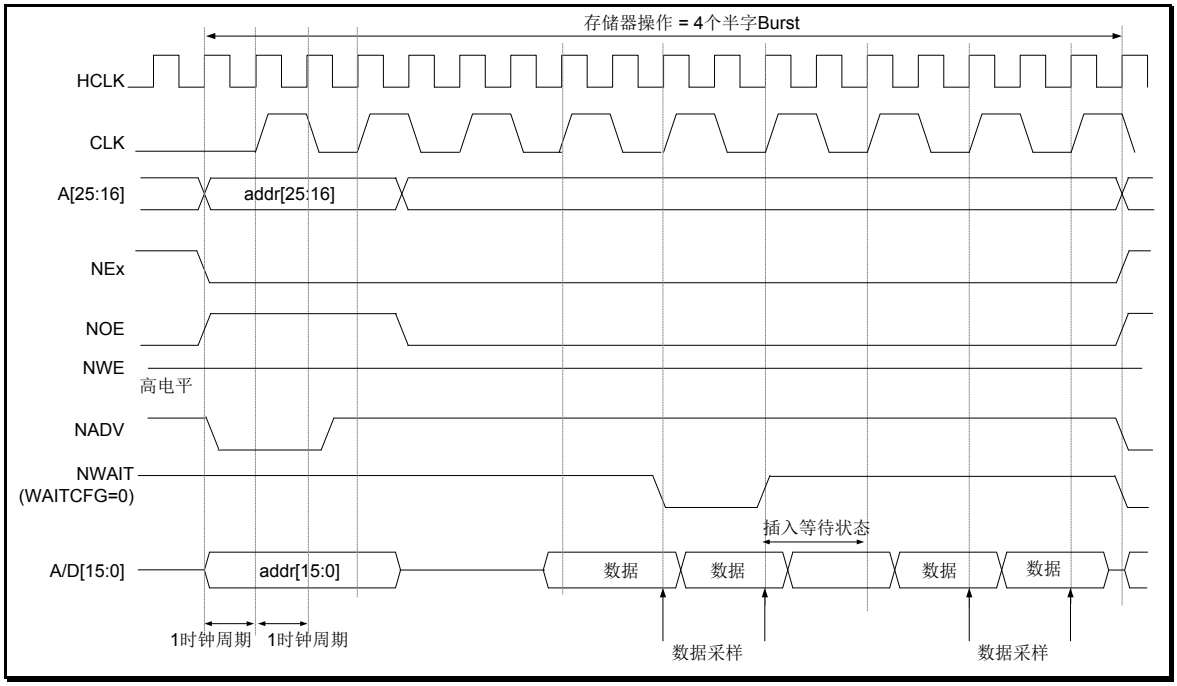


图 30-19 同步复用读模式 – NOR, PSRAM (CRAM)注 1：字节通道和输出 BL 并未在上图中显示对于 NOR 访问，字节通道和输出 BL 保持高电平；对于 PSRAM（CRAM）访问，保持低电平。

注 2：NWAIT 极性设置为 0。

Bit	位名	设置值
31-20	保留	0x000
19	RWCBURSTEN	对同步读无效
18-16	CPAGESIZE	如有需要，则设置（使用 CRAM1.5 时，设置为 0x1）
15	ASYNCWAIT	0x0
14	EXTMODEN	0x0
13	WAITEN	如果存储器支持该功能，设置为 1；否则设置为 0
13	WREN	对同步读无效
11	WAITCFG	根据存储器来设置
10	WARPMOD	0x0
9	WAITPOL	根据存储器来设置
8	BURSTEN	0x1
7	保留	0x1
6	FLASHACCEN	根据存储器来设置（NOR Flash 存储器）
5-4	MEMWID	如有需要，则设置
3-2	MEMTYP [0:1]	0x1，0x2
1	MUXE	如有需要，则设置
0	MEMBKEN	0x1

表 30-27 EBI_BCTRLRx 寄存器设置

Bit	位名	设置值
31-30	保留	0x0
29-28	ACCMODE	0x0
27-24	DATALAT	数据延时
23-20	CLKDIV	0x0 使得 CLK=CLK（不支持） 0x1 使得 CLK=2xCLK
19-16	BUSTURN	无效
15-8	DATAHOLD	无关位
7-4	ADDHOLD	无关位
3-0	ADDSET[3:0]	无关位

表 30-28 EBI_BTRx 寄存器设置

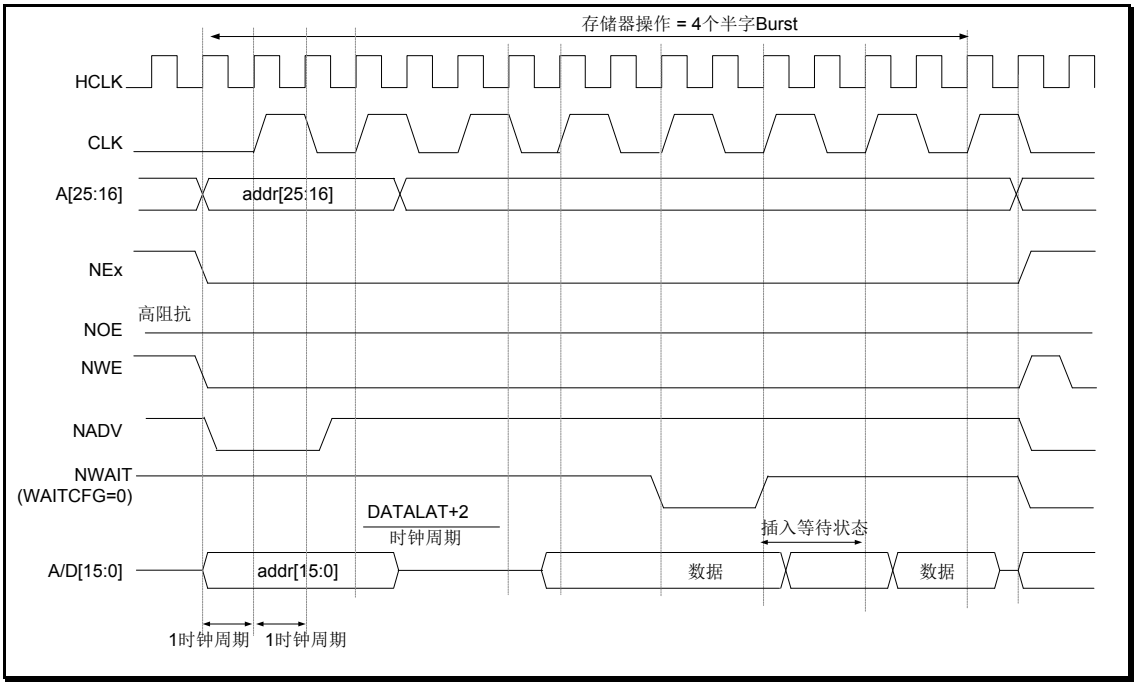


图 30-20 同步复用写模式 – PSRAM (CRAM)

注 1：存储器必须提前 1 个周期发送 NWAIT 信号，相应地，WAITCFG 必须设置为 0。
注 2：NWAIT 极性设置为 0。
注 3：字节通道 NBL 输出未显示在上图中，当 NEx 有效时，NBL 输出保持低电平。

Bit	位名	设置值
31-20	保留	0x000
19	RWCBURSTEN	0x1
18-16	CPAGESIZE	如有需要，则设置（使用 CRAM1.5 时，设置为 0x1）
15	ASYNCWAIT	0x0
14	EXTMODEN	0x0
13	WAITEN	如果存储器支持该功能，设置为 1；否则设置为 0
13	WREN	0x1
11	WAITCFG	0x0
10	WARPMOD	0x0
9	WAITPOL	根据存储器来设置
8	BURSTEN	对同步写无效
7	保留	0x1
6	FLASHACCEN	根据存储器来设置
5-4	MEMWID	如有需要，则设置
3-2	MEMTYP [0:1]	0x1
1	MUXE	如有需要，则设置
0	MEMBKEN	0x1

表 30-29 EBI_BCTRLRx 寄存器设置

Bit	位名	设置值
31-30	保留	0x0
29-28	ACCMODE	0x0
27-24	DATALAT	数据延时
23-20	CLKDIV	0x0 使得 CLK=CLK（不支持） 0x1 使得 CLK=2xCLK
19-16	BUSTURN	NEx 高电平和 NEx 低电平之间的时间（BUSTURN HCLK）
15-8	DATAHOLD	无关位
7-4	ADDHOLD	无关位
3-0	ADDSET[3:0]	无关位

表 30-30 EBI_BTRx 寄存器设置

1.

30. 4. 4 NAND Flash

EBI 可生成不同的信号时序来驱动以下类型的器件：

◇ NAND Flash (8/16 位)

NAND 控制器可控制 2 个外部存储组。存储组 2 和存储组 3 支持 NAND Flash 器件。

每个存储组有专门的寄存器配置。可设置的存储器参数包括访问时序（如下表）和 ECC 配置。

参数	功能	访问模式	单位	最小值	最大值
存储器建立时间	发出命令前建立地址的（HCLK）时钟周期数	读/写	AHB 时钟周期（HCLK）	1	255
存储器等待时间	发出命令的最短持续时间（HCLK 时钟周期）	读/写		2	256
存储器保持时间	发送命令结束后保持地址（在写访问时，为保持数据）的（HCLK）时钟周期数	读/写		1	254
存储器数据总线高阻抗	开始写访问后，数据总线保持为高阻抗的（HCLK）时钟周期数	写		0	255

表 30-31 可配置的 NAND 访问参数

30. 4. 4. 1 外部存储器接口信号

下表列出了用于接口 NAND Flash 的典型信号线。

EBI 信号名	I/O	功能
A[17]	O	NAND Flash 地址锁存使能（ALE）信号
A[16]	O	NAND Flash 命令锁存使能（CLE）信号
D[7:0]	I/O	8 位复用，双向地址/数据总线
NCE[x]	O	片选，x = 2，3
NOE(= NRE)	O	输出使能（存储器信号名：读使能，NRE）
NWE	O	写使能
NWAIT/INT[3:2]	I	NAND Flash 就绪/忙碌输入信号至 EBI

表 30-32 8 位 NAND Flash

EBI 可以根据需要产生多个地址周期，理论上 EBI 不限制可以访问的 NAND 容量。

EBI 信号名	I/O	功能
A[17]	O	NAND Flash 地址锁存使能（ALE）信号
A[16]	O	NAND Flash 命令锁存使能（CLE）信号
D[15:0]	I/O	16 位复用，双向地址/数据总线
NCE[x]	O	片选，x = 2，3
NOE(= NRE)	O	输出使能（存储器信号名：读使能，NRE）
NWE	O	写使能
NWAIT/INT[3:2]	I	NAND Flash 就绪/忙碌输入信号至 EBI

表 30-33 16 位 NAND Flash

30. 4. 4. 2 NAND Flash操作

下表列出了 NAND Flash 控制器支持的器件，访问模式和操作，不支持的也同样列出。

器件	模式	R/W	AHB 数据大小	存储器 数据大小	支持与否	备注
NAND 8 位	异步	R	8	8	支持	
	异步	W	8			
	异步	R	16			分为 2 个 EBI 访问
	异步	W	16			分为 2 个 EBI 访问
	异步	R	32			分为 4 个 EBI 访问
	异步	W	32			分为 4 个 EBI 访问
NAND 16 位	异步	R	8	16	支持	
	异步	W	8		**不支持**	
	异步	R	16		支持	
	异步	W	16			
	异步	R	32			分为 2 个 EBI 访问
	异步	W	32			分为 2 个 EBI 访问

表 30-34 支持的存储器和操作

NAND Flash 不支持从 AHB 总线写 8 位的操作到 16 位宽的存储器。

30. 4. 4. 3 NAND时序图

每个 NAND Flash 存储器组都由一组寄存器控制：

◇ 控制寄存器：EBI_PCTRLRx

◇ 中断状态寄存器：EBI_STARx

◇ ECC 寄存器：EBI_ECCRESULTx

◇ 通用存储空间的时序寄存器：EBI_PMEMRx

◇ 属性存储空间的时序寄存器：EBI_PATTRx

◇ I/O 空间的时序寄存器：EBI_PIORx

每个时序配置寄存器包含了 3 个参数用来定义 NAND Flash 操作中 3 个阶段的 HCLK 周

期数，还有一个定义了写操作中 EBI 开始驱动数据总线时基的参数。下图为 NAND 控制器通用存储空间的访问时序。

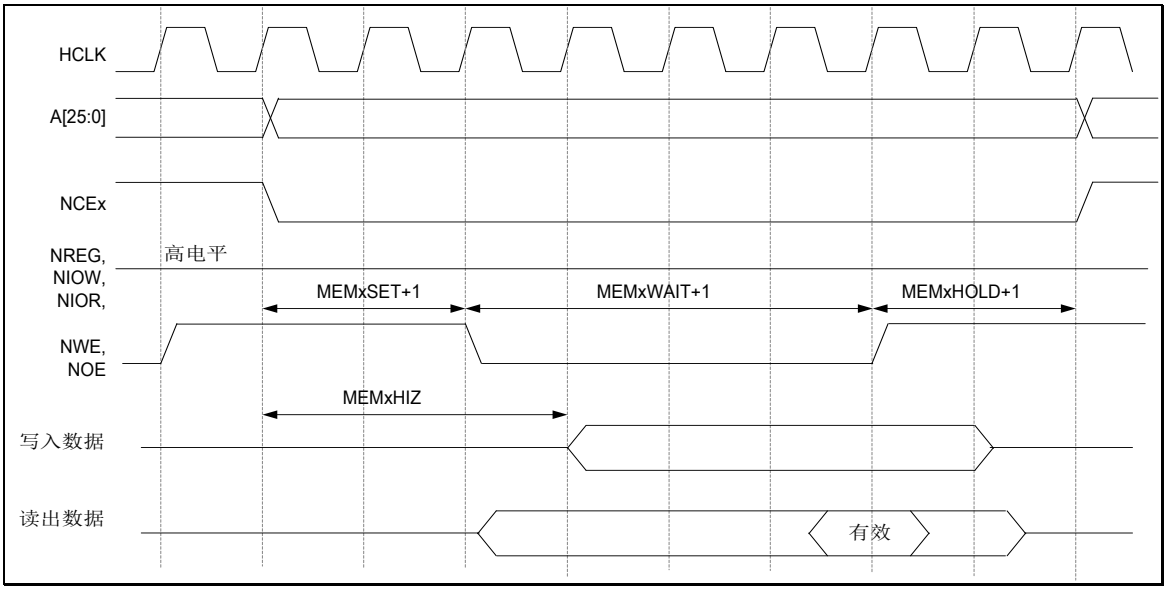


图 30-21　NAND 控制器通用存储器访问时序

注 1：在写访问过程中，NOE 保持高电平（无效）。写访问过程中，NWE 保持高电平（无效）。
注 2：一旦请求 NAND 访问，NCEx 变为并保持低电平直到访问不同的存储器组。

30. 4. 4. 4　NAND Flash操作

NAND Flash 的命令锁存使能（CLE）和地址锁存使能（ALE）信号由 EBI 控制器的地址信号驱动。向 NAND Flash 存储器发送命令和地址时，CPU 需在其存储空间的某特定地址进行写操作。

从 NAND Flash 器件进行典型页读的步骤如下：

1. 根据 NAND Flash 特性（NAND Flash 数据总线宽度由 PDATBUSWID 控制，PMEMTYP=1，根据需求 PWAITEN 可设置为 0 或 1），通过配置 EBI_PCTRLRx 和 EBI_PMEMRx（对于某些器件，需配置 EBI_PATTRx）设置和使能相应的存储器组。

2. CPU 在通用存储器空间执行字节的写操作，一个数据字节等于一个 Flash 命令字节。在写信号有效期间（NWE 为低脉冲），NAND Flash 的 CLE 输入有效，因此写入的字节被 NAND Flash 识别为一个命令。一旦该命令被 NAND Flash 器件锁存，后面的页读操作无需再发送相同的命令。

3. 在通用存储器或属性空间写入所需的字节，以 64Mbx8bit NAND Flash 为例，写入 STARTAD[7:0]，STARTAD[15:8]，STARTAD[23:16]和 STARTAD[25:24]，以此 CPU 可发送读操作的起始地址（STARTAD）。在写信号有效期间（NWE 为低脉冲），NAND Flash 器件的 ALE 输入为有效，因此写入的字节被识别为读操作的起始地址。利用属性空间可使 EBI 产生不同的时序配置，该功能可用来实现某些 NAND Flash 存储器的预等待功能。

4. 在启动新的访问之前（同一个或不同的存储器组），控制器会一直等待直到 NAND Flash 变为有效（R/NB 信号为高电平）。等待期间，NCE 信号保持低有效。

5. 此时，CPU 可在通用存储器空间执行字节读操作来读取 NAND Flash 页。

6. 无需 CPU 命令或地址写操作，可用以下 3 种不同的方式来读取 NAND Flash 下一页

- 通过执行步骤 5
- 返回步骤 3 重新输入新的地址
- 返回步骤 3 重新输入新的命令

30. 4. 4. 5 NAND Flash预等待功能

在写入最后一个地址字节后，有些 NAND Flash 器件会要求控制器等待 R/NB 信号变为低电平。

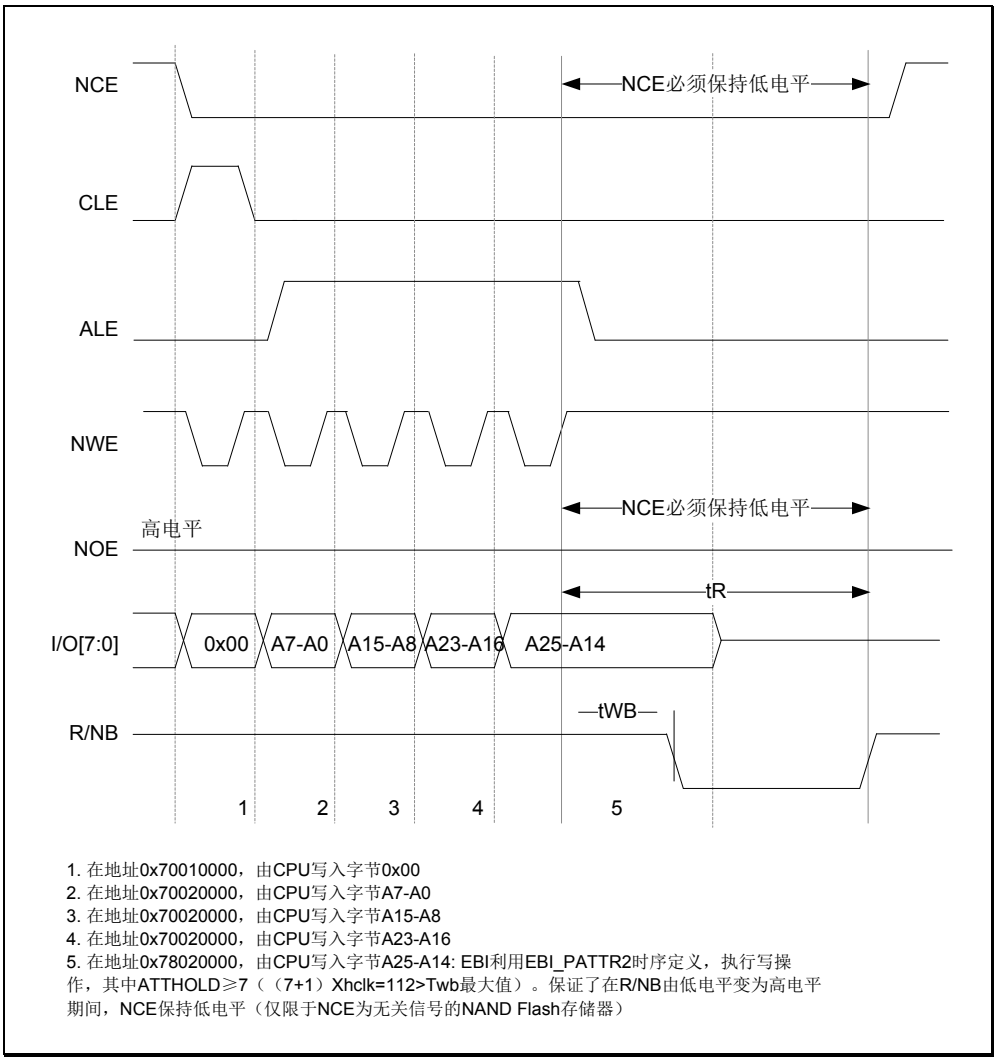


图 30-22 访问非“CE 无关”NAND Flash

当使用该功能时，可通过配置 MEMHOLD 值来满足 tWB 的时序，但是任何虽有的 CPU 对 NAND Flash 的读或写操作中，控制器都会在 NWE 信号的上升沿至下一次操作之间插入一个保持延时，延时长度为（MEMHOLD+1）个 HCLK 周期。

为克服该时序限制，这里使用了属性存储空间配置 ATTHOLD 值使其符合 tWB 的时序，同时保持 MEMHOLD 为其最小值。此时，CPU 必须在所有 NAND Flash 的读写操作时使用通用存储空间。只有在写入 NAND Flash 地址的最后一个字节时，CPU 才需要写入属性存储空间。

30. 4. 4. 6 NAND Flash存储器纠错码ECC计算

EBI 控制器包含 2 个纠错码计算硬件模块，每个模块对应一个存储器组。当软件在系统中处理纠错码时，可减少主机 CPU 的工作负荷。

有 2 个相同的寄存器，分别对应存储器组 2 和存储器组 3。

EBI 中的纠错码算法可在读或写 NAND Flash 时，在每 256，512，1024，2048，4096 或 8192 个字节中，纠正 1 个比特位的错误并检测出 2 个比特位的错误。

每次当 NAND Flash 存储器组处于有效状态时，ECC 模块可监测 NAND Flash 数据总线和读/写信号（NCE 和 NWE）。

该功能的操作如下：

◇ 当访问 NAND Flash 的存储器组 2 或存储器组 3 时，D[15:0]总线上的数据被锁存并用于 ECC 计算。

◇ 当访问 NAND Flash 的任何其他地址时，ECC 逻辑处于空闲状态，不会执行任何操作。因此，定义 NAND Flash 命令和地址的写操作不会参与 ECC 计算。

当所需的字节数由主机 CPU 从 NAND Flash 读取或写入，必须读取 EBI_ECCRESULT2/3 寄存器以获得计算结果。一旦读取，通过设置 ECCEN 为 0 将寄存器清零。计算新的数据块时，EBI_PCTRLR2/3.ECCEN 必须设置为 1。

以下为 ECC 计算步骤：

1. 使能 EBI_PCTRLR.ECCEN
2. 在 NAND Flash 存储器页中写入数据。当 NAND 页写入数据后，ECC 模块计算出 ECC 值。
3. 在 EBI_ECCRESULT2/3 寄存器中读取 ECC 值，以变量形式存储。
4. 将 ECCEN 清零。接着在从 NAND 页读回写入的数据前，再重新使能 ECCEN。当 NAND 页写入数据后，ECC 模块计算出 ECC 值。
5. 在 EBI_ECCRESULT2/3 寄存器中读取新的 ECC 值。
6. 如果两次读取的 ECC 值一样，无需纠正，否则存在 ECC 错误，软件纠正程序会返回信息告知该错误是否可纠正。

30. 5 特殊功能寄存器

30. 5. 1 寄存器列表

EBI 寄存器列表		
寄存器名称	偏移地址	寄存器描述
EBI 基地址：$A000\_0000_H$		
EBI_BCTRLR1	000_H	EBI SRAM/NOR Flash 片选控制寄存器 1
EBI_BCTRLR2	008_H	EBI SRAM/NOR Flash 片选控制寄存器 2
EBI_BCTRLR3	010_H	EBI SRAM/NOR Flash 片选控制寄存器 3
EBI_BCTRLR4	018_H	EBI SRAM/NOR Flash 片选控制寄存器 4
EBI_BTR1	004_H	EBI SRAM/NOR Flash 片选时序寄存器 1
EBI_BTR2	$00C_H$	EBI SRAM/NOR Flash 片选时序寄存器 2
EBI_BTR3	014_H	EBI SRAM/NOR Flash 片选时序寄存器 3
EBI_BTR4	$01C_H$	EBI SRAM/NOR Flash 片选时序寄存器 4
EBI_BWRTR1	104_H	EBI SRAM/NOR Flash 写时序寄存器 1
EBI_BWRTR2	$10C_H$	EBI SRAM/NOR Flash 写时序寄存器 2
EBI_BWRTR3	114_H	EBI SRAM/NOR Flash 写时序寄存器 3
EBI_BWRTR4	$11C_H$	EBI SRAM/NOR Flash 写时序寄存器 4
Reserved	200_H~$23C_H$	保留
EBI_PCTRLR2	060_H	EBI NAND Flash 控制寄存器 2
EBI_PCTRLR3	080_H	EBI NAND Flash 控制寄存器 3
Reserved	$0A0_H$	保留
EBI_STAR2	064_H	EBI FIFO 状态和中断寄存器 2
EBI_STAR3	084_H	EBI FIFO 状态和中断寄存器 3
Reserved	$0A4_H$	保留
EBI_PMEMR2	068_H	EBI 通用存储器空间时序寄存器 2
EBI_PMEMR3	088_H	EBI 通用存储器空间时序寄存器 3
Reserved	$0A8_H$	保留
EBI_PATTR2	$06C_H$	EBI 属性存储器空间时序寄存器 2
EBI_PATTR3	$08C_H$	EBI 属性存储器空间时序寄存器 3
Reserved	$0AC_H$	保留
Reserved	$0B0_H$	保留
EBI_ECCRESULT2	074_H	EBI ECC 结果寄存器 2
EBI_ECCRESULT3	094_H	EBI ECC 结果寄存器 3

30. 5. 2 寄存器描述

30. 5. 2. 1 EBI SRAM/NOR-Flash片选控制寄存器（EBI_BCTRLR1~4）

EBI SRAM/NOR-Flash 片选控制寄存器（EBI_BCTRLRx, x=1~4）

EBI_BCTRLR1 偏移地址：00_H

EBI_BCTRLR2 偏移地址：08_H

EBI_BCTRLR3 偏移地址：10_H

EBI_BCTRLR4 偏移地址：18_H

EBI_BCTRLR1 复位值：$00000000\_00000000\_00110000\_11011011_B$

EBI_BCTRLR2~4 复位值：$00000000\_00000000\_00110000\_11010010_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												RWCBURSTEN	CPAGESIZE			ASYNCWAIT	EXMODEN	WAITEN	WREN	WAITCFG	WRAPMODE	WAITPOL	BURSTEN	Reserved	FLASHACCEN	MEMWID		MEMTYP		MUXEN	MEMBKEN

Reserved	Bit 31-20	—	**保留**
RWCBURSTEN	Bit 19	R/W	**写 BUSRT 使能** 针对 Cellular RAM (PSRAM)存储器，在写操作中，该位使能同步 burst 协议。同步读操作的使能位是 EBI_BCTRLRx.BURSTEN 0：在异步模式下，始终执行写操作 1：在同步模式下执行写操作
CPAGESIZE[2:0]	Bit 18-16	R/W	**CRAM 页大小** 该位用于 Celluar RAM1.5。Cellular RAM 1.5 不支持页与页之间的跨页 burst 访问。当配置该位以后，在达到存储器页容量时，EBI 控制器会自动划分 burst 操作。 000：跨页时，无 burst 划分（复位后默认状态） 001：128 字节 010：256 字节 011：512 字节 100：1024 字节 其他：保留
ASYNCWAIT	Bit 15	R/W	**异步操作中的等待信号** 0：当运行异步协议时，NWAIT 信号不参与其中（复位后默认状态） 1：当运行异步协议时，NWAIT 信号参与其中。
EXTMODEN	Bit 14	R/W	**扩展模式使能** 该位对于非复用异步操作，可在 EBI_BWRTR 寄存器内设置不同的写时序，因此可产生不同的读写时序。 0：EBI_BWRTR 寄存器值不参与其中（默认）

			1：EBI_BWRTR 寄存器值参与其中 注意：当扩展模式禁止，EBI 能运行于模式 1 和模式 2，如下： 当选择 SRAM/PSRAM 存储器形式时（MEMTYP [0:1]=0x0 或 0x01），模式 1 为默认模式。 当选择 NOR 存储器形式时（MEMTYP [0:1]=0x10），模式 2 为默认模式。
WAITEN	Bit 13	R/W	**等待使能** 在同步模式下访问 Flash 存储器时，该位通过 NWAIT 信号使能/禁止插入等待状态。 0：禁止 NWAIT 信号（在设置的 Flash 延时周期后，不插入等待状态） 1：使能 NWAIT 信号（在设置的 Flash 延时周期后插入等待状态）。复位后默认状态。
WREN	Bit 12	R/W	**写使能** 0：禁止 EBI 在存储器组上进行写操作，否则产生一个 AHB 错误。 1：使能 EBI 在存储器组上进行写操作。复位后默认状态。
WAITCFG	Bit 11	R/W	**等待时序配置** NWAIT 信号表明存储器数据是否有效，或者当在同步模式下访问 Flash 存储器时，是否需要插入等待状态。该配置位决定了，存储器是在等待状态前 1 个时钟周期，还是在等待状态期间产生 NWAIT 信号。 0：在等待状态前 1 个数据周期产生 NWAIT 有效信号（复位后默认状态）。 1：在等待状态期间产生 NWAIT 有效信号。
WRAPMODE	Bit 10	R/W	**支持 Wrapped Burst 模式** 该位决定控制器是否支持将 AHB burst wrap 操作分成 2 次线性操作。仅在 burst 模式下有效。 0：禁止直接 wrap burst（复位后默认状态） 1：使能直接 wrap burst 注意：因 CPU 和 DMA 不能产生 wrap burst 操作，因此该位无效。
WAITPOL	Bit 9	R/W	**等待信号极性** 仅在 burst 模式下有效。 0：NWAIT 为低有效（复位后默认状态） 1：NWAIT 为高有效
BURSTEN	Bit 8	R/W	**Burst 使能** 仅限于同步存储器在 burst 模式下时有效。 0：禁止 Burst 模式（复位后默认状态）。在异步模

			式下进行读操作。 1：使能 Burst 模式。在同步模式下进行读操作。
Reserved	Bit 7	R/W	**保留**
FLASHACCEN	Bit 6	R/W	**Flash 操作使能** 0：禁止相应的 NOR Flash 存储器操作 1：使能相应的 NOR Flash 存储器操作（复位后默认状态）
MEMWID[1:0]	Bit 5-4	R/W	**存储器数据总线宽度** 定义了外部存储器设备宽度，对所有的存储器类型都有效。 00：8 位 01：16 位（复位后默认状态） 10：保留 11：保留
MEMTYP[1:0]	Bit 3-2	R/W	**存储器类型** 定义了连接着相应存储器组的外部存储器类型 00:SRAM（复位后默认状态,针对存储器组 2~4） 01：PSRAM（CRAM） 10：NOR Flash （复位后默认状态，针对存储器组 1） 11：保留
MUXEN	Bit 1	R/W	**地址/数据复用使能** 当该位设置为 1 时，地址和数据将共用数据总线，该位仅对 NOR 和 PSRAM 存储器有效。 0：地址/数据不复用 1：地址/数据复用数据总线。（复位后默认状态）
MEMBKEN	Bit 0	R/W	**存储器组使能** 使能对应的存储器组。复位后，存储器组 1 为使能状态，其他存储器组为禁止状态。访问禁止的存储器组将在 AHB 总线产生错误。 0：禁止对应的存储器组 1：使能对应的存储器组

30. 5. 2. 2 EBI SRAM/NOR-Flash片选时序寄存器（EBI_BTR1~4）

EBI SRAM/NOR-Flash 片选时序寄存器（EBI_BTRx, x=1~4）

EBI_BTR1 偏移地址：04_H

EBI_BTR2 偏移地址：$0C_H$

EBI_BTR3 偏移地址：14_H

EBI_BTR4 偏移地址：$1C_H$

EBI_BTR1~4 复位值：$00001111\_11111111\_11111111\_11111111_B$

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

Reserved	Bit 31-30	—	**保留**
ACCMODE	Bit 29-28	R/W	**访问模式** 定义异步访问模式，该位只在 EBI_BCTRLRx.EXTMODEN=1 时才起作用。 00：访问模式 A 01：访问模式 B 10：访问模式 C 11：访问模式 D
DATALAT	Bit 27-24	R/W	**同步 NOR Flash 存储器数据延时** 当同步 NOR Flash 存储器 burst 模式使能，该位定义了在读/写第一个数据前，发送存储器时钟的周期数（+2） 该时序参数表现在 EBI_CLK 周期中，而非 HCLK 周期中。对于 PSRAM（CRAM），该位必须设置为 0。在异步 NOR Flash 或 SRAM 或 PSRAM 中，该位无效。 0000：第一个 burst 访问的数据延迟为 2 个 CLK 时钟周期。 1111：第一个 burst 访问的数据延迟为 17 个 CLK 时钟周期。（复位后的默认状态）
CLKDIV	Bit 23-20	R/W	**时钟分频比（EBI_CLK 信号）** 该位定义了 EBI_CLK 时钟输出信号的周期，由 HCLK 周期数表示。在异步 NOR Flash，SRAM 或 PSRAM 访问中，该位无效。 0000：保留 0001：EBI_CLK 周期 = 2 x HCLK 周期 0010：EBI_CLK 周期 = 3 x HCLK 周期 1111：EBI_CLK 周期 = 16 x HCLK 周期 （复位后的默认状态
BUSTURN	Bit 19-16	R/W	**总线恢复时间**

			该位由软件设置，在 write-to-read 和 read-to-write 操作最后，插入一段延时。设置好的总线恢复时间插入在异步读（复用或 D 模式）或写操作和其他异步/同步读或写之间（读操作，存储器组可以是相同的也可以是不同的；写操作，除了复用或 D 模式，存储器组可以是不同的）。 0000：1 个 HCLK 时钟周期 … 1111：16 个 HCLK 时钟周期（复位后的默认状态） 在有些情况下，总线恢复延时是固定的（详见该寄存器下的备注）
DATAHOLD	Bit 15-8	R/W	**数据保持时间** 适用于异步访问。 0000000：保留 00000001：2 个 HCLK 时钟周期 00000010：3 个 HCLK 时钟周期 … 11111111：256 个 HCLK 时钟周期（复位后的默认状态） 举例：模式 1、读操作、DATADUR=1：数据保持时间=DATADUR+3=4 个 HCLK 时钟周期
ADDHOLD	Bit 7-4	R/W	**地址保持时间** 适用于模式 D 和复用访问。在同步访问时，地址保持时间始终保持 1 个存储器时钟周期。 0000：保留 0001：2 个 HCLK 时钟周期 0010：3 个 HCLK 时钟周期 … 1111：16 个 HCLK 时钟周期（复位后的默认状态） 举例：模式 D、读操作、ADDHOLD=1：地址保持时间=ADDHOLD+1=2 个 HCLK 时钟周期
ADDATASETUP[3:0]	Bit 3-0	R/W	**地址建立时间** 适用于 SRAM，ROM，和异步 NOR Flash 及 PSRAM 访问。在同步 NOR Flash 和 PSRAM 中，该位无效。 0000：1 个 HCLK 时钟周期 … 1111：16 个 HCLK 时钟周期（复位后的默认状态） 举例：模式 2、读操作、ADDSETDUR=1：地址建立时间=ADDSETDUR+1=2 个 HCLK 时钟周期

注 1：总线恢复延时在以下情况下是固定的：

- 对相同的静态存储器组进行两个连续的异步写操作时（除了复用模式和 D 模式），无需插入总线恢复延时。
- 在以下情况中，插入 1 个 EBI 时钟周期的总线恢复延时：

1. 对不同的静态存储器组进行两个连续同步写操作时（burst 模式或 single 模式下）
2. 对相同的或不同的存储器组进行同步写操作（burst 模式或 single 模式）和同步读操作
3. 除复用和 D 模式外，向任何静态或动态的存储器组进行异步写和同步写
4. 从不同的静态存储器组进行异步读（模式 1，2，A，B 或 C）和读操作

- 在以下情况中，插入 2 个 EBI 时钟周期的总线恢复延时：

1. 对相同的存储器组进行两个连续的同步写操作（burst 模式或 single 模式）
2. 对静态存储器组（读操作时，可以是相同也可以是不同的存储器组）进行同步写（burst 模式和 single 模式）和异步写或读操作
3. 两个连续的同步读操作（burst 模式和 single 模式），随后对另一个静态存储器组进行同步/异步读或写

- 在以下情况中，插入 3 个 EBI 时钟周期的总线恢复延时：

1. 对不同的静态存储器组进行两个连续同步写操作时（burst 模式或 single 模式下）
2. 对相同的或不同的存储器组进行同步写操作（burst 模式或 single 模式）和同步读操作

注 2：因为内部刷新，PSRAM (CRAM)具有可变的保持延迟，因此这样的存储器会在数据保持期间输出 NWAIT 信号来延长数据的保持时间。使用 PSRAM（CRAM）的 DATALAT 应设置为 0，这样 EBI 可以及时的退出自己的保持阶段并开始对存储器发出的 NWAIT 信号进行采样，然后在存储器准备就绪时开始读或写操作。

30. 5. 2. 3 EBI SRAM/NOR-Flash写时序寄存器（EBI_BWRTR1~4）

EBI SRAM/NOR-Flash 写时序寄存器（EBI_BWRTRx, x=1~4）

EBI_BWRTR1 偏移地址：104_H

EBI_BWRTR2 偏移地址：$10C_H$

EBI_BWRTR3 偏移地址：114_H

EBI_BWRTR4 偏移地址：$11C_H$

EBI_BWRTR1~4 复位值：$00001111\_11111111\_11111111\_11111111_B$

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

Reserved	Bit 31-30	—	**保留**
ACCMODE	Bit 29-28	R/W	**访问模式** 定义异步访问模式，该位只在 EBI_BCTRLRx.EXTMODEN=1 时才起作用。 00：访问模式 A 01：访问模式 B 10：访问模式 C 11：访问模式 D
Reserved	Bit 27-20	R/W	**保留**
BUSTURN	Bit 19-16	R/W	**总线恢复时间** 设置好的总线恢复时间插入在异步写和异步/同步读或写操作之间（读操作，存储器组可以是相同的也可以是不同的；写操作，除了复用或 D 模式，存储器组可以是不同的）。 0000：1 个 HCLK 时钟周期 … 1111：16 个 HCLK 时钟周期（复位后的默认状态） 在有些情况下，总线恢复延时是固定的（详见该寄存器下的备注）
DATAHOLD	Bit 15-8	R/W	**数据保持时间** 适用于异步 SRAM，PSRAM 和 NOR Flash 访问。同步模式下，该位无效。 0000000：保留 00000001：2 个 HCLK 时钟周期 00000010：3 个 HCLK 时钟周期 … 11111111：256 个 HCLK 时钟周期（复位后的默认状态）

ADDHOLD	Bit 7-4	R/W	**地址保持时间** 适用于异步复用访问。在同步访问时，地址保持时间始终保持 1 个 Flash 时钟周期。 0000：保留 0001：2 个 HCLK 时钟周期 0010：3 个 HCLK 时钟周期 … 1111：16 个 HCLK 时钟周期（复位后的默认状态）
ADDATASETUP	Bit 3-0	R/W	**地址建立时间** 适用于异步访问。在同步 NOR Flash 和 PSRAM 中，该位无效。 0000：1 个 HCLK 时钟周期 … 1111：16 个 HCLK 时钟周期（复位后的默认状态）

注：总线恢复延时在以下情况下是固定的：

- 对相同的静态存储器组进行两个连续的异步写操作时（除了复用模式和 D 模式），无需插入总线恢复延时。
- 在以下情况中，插入 2 个 EBI 时钟周期的总线恢复延时：

1. 对相同的存储器组进行两个连续的同步写操作（burst 模式或 single 模式）
2. 对静态存储器组（读操作时，可以是相同也可以是不同的存储器组）进行同步写（burst 模式和 single 模式）和异步写或读操作
3. 两个连续的同步读操作（burst 模式和 single 模式），随后对另一个静态存储器组进行同步/异步读或写

- 在以下情况中，插入 3 个 EBI 时钟周期的总线恢复延时：

1. 对不同的静态存储器组进行两个连续同步写操作时（burst 模式或 single 模式下）
2. 对相同的或不同的存储器组进行同步写操作（burst 模式或 single 模式）和同步读操作

30. 5. 2. 4 EBI NAND Flash控制寄存器（EBI_PCTRLR2~3）

EBI NAND Flash 控制寄存器（EBI_PCTRLRx, x=2~3）

EBI_PCTRLR2 偏移地址：060_H

EBI_ PCTRLR3 偏移地址：080_H

EBI_ PCTRLR2~3 复位值：$00000000\_00000000\_00000000\_00011000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												ECCPSIZE			ARDLY				CRDLY				TFUNSEL	Reserved	ECCEN	PDATBUSWID		PMEMTYP	PMEMBKEN	PWAITEN	Reserved

Reserved	Bit 31-20	—	保留
ECCPSIZE	Bit 19-17	R/W	**ECC 页字节数** 000：256 字节 001：512 字节 010：1024 字节 011：2048 字节 100：4096 字节 101：8192 字节
ARDLY	Bit 16-13	R/W	**ALE 到 RE 的延时** 设置从 ALE 低电平到 RE 低电平之间的时间，单位为 AHB 时钟周期（HCLK） t_ar =（ARDLY +SET+2）x THCLK，其中 THCLK 为 HCLK 时钟周期 0000：1 个 HCLK 周期（默认） 1111：16 个 HCLK 周期 注意，根据寻址空间，SET 可以是 MEMSETUP 或 ATTSETUP。
CRDLY	Bit 12-9	R/W	**CLE 到 RE 的延时** 设置从 CLE 低电平到 RE 低电平之间的时间，单位为 AHB 时钟周期（HCLK） t_clr =（CRDLY +SET+2）x THCLK，其中 THCLK 为 HCLK 时钟周期 0000：1 个 HCLK 周期（默认） 1111：16 个 HCLK 周期 注意，根据寻址空间，SET 可以是 MEMSETUP 或 ATTSETUP。
TFUNSEL	Bit 8	R/W	**ARDLY & CRDLY 功能选择** 选择 ARDLY 和 CRDLY 其中较为大的延时（默认） 1：如果上一个访问发生在地址段，则 ARDLY 有效；或如果上一个访问发生在命令段，则 CRDLY

			有效。
Reserved	Bit 7	R/W	**保留**
ECCEN	Bit 6	R/W	**ECC 计算逻辑使能** 0：禁止和复位 ECC 逻辑（复位后的默认状态） 1：使能 ECC 逻辑
PDATBUSWID	Bit 5-4	R/W	**数据总线宽度** 定义外部存储器器件宽度 00：8 位 01：16 位（复位后的默认状态）。 10：保留 11：保留
PMEMTYP	Bit 3	R/W	**存储器类型** 定义了连接相应存储器组的器件类型。 0：预留 1：NAND Flash （复位后的默认状态）
PMEMBKEN	Bit 2	R/W	**NAND Flash 存储器组使能** 使能存储器组。访问一个禁止的存储器组会在 AHB 总线上产生错误。 0：禁止相应的存储器组（复位后的默认状态） 1：使能相应的存储器组
PWAITEN	Bit 1	R/W	**等待功能使能** 使能 NAND Flash 存储器组的等待功能 0：禁止 1：使能
Reserved	Bit 0	—	**保留**

30. 5. 2. 5 EBI FIFO状态和中断寄存器（EBI_STAR2~3）

EBI FIFO 状态和中断寄存器（EBI_STARx, x=2~3）

EBI_STAR2 偏移地址：064_H

EBI_STAR3 偏移地址：084_H

EBI_STAR2~3 复位值：$00000000\_00000000\_00000000\_01000000_B$

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

Reserved	Bit 31-7	—	保留
FIFOEMPT	Bit 6	R	**FIFO 空** 0：FIFO 非空 1：FIFO 空
IFALLEN	Bit 5	R/W	**中断下降沿检测使能** 0：禁止中断下降沿检测请求 1：使能中断下降沿检测请求
IHILEN	Bit 4	R/W	**中断高电平检测使能** 0：禁止中断高电平检测请求 1：使能中断高电平检测请求
IRISEEN	Bit 3	R/W	**中断上升沿检测使能** 0：禁止中断上升沿检测请求 1：使能中断上升沿检测请求
IFALLS	Bit 2	R/W	**中断下降沿状态** 该标志位由硬件置 1，软件复位。 0：未产生中断下降沿 1：产生中断下降沿
IHILS	Bit 1	R/W	**中断高电平状态** 该标志位由硬件置 1，软件复位。 0：未产生中断高电平 1：产生中断高电平
IRISES	Bit 0	R/W	**中断上升沿状态** 该标志位由硬件置 1，软件复位。 0：未产生中断上升沿 1：产生中断上升沿

30. 5. 2. 6 EBI 通用存储器空间时序寄存器（EBI_PMEMR2~3）

EBI 通用存储器空间时序寄存器（EBI_PMEMRx, x=2~3）

EBI_PMEMR2 偏移地址：068_H

EBI_ PMEMR3 偏移地址：088_H

EBI_ PMEMR2~3 复位值：$11111100\_11111100\_11111100\_11111100_B$

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

MEMHIZTx	Bit 31-24	R/W	**通用存储器 x 数据总线高阻抗时间** 设置了 HCLK 的时钟周期数(针对 NAND 时，+1)。当 NAND Flash 在 socketx 上开始对通用存储空间进行写操作后，在设置的该段时钟周期内，数据总线保持高阻抗。仅对写操作有效。 00000000：1 个 HCLK 周期 11111110：255 个 HCLK 周期 11111111：保留
MEMHOLDx	Bit 23-16	R/W	**通用存储器 x 保持时间** 当 NAND Flash 在 socket x 上对通用存储空间进行读或写操作时，该位设置了当发送命令（NWE，NOE）后，地址（和写操作数据）的保持时间。 00000000：保留 00000001：1 个 HCLK 周期 11111110：254 个 HCLK 周期 11111111：保留
MEMWAITx	Bit 15-8	R/W	**通用存储器 x 等待时间** 当 NAND Flash 在 socket x 上对通用存储空间进行读或写操作时，该位设置了保持命令（NWE，NOE）的最小时钟周期 HCLK（+1）。当该设置值结束时，如果等待信号（NWAIT）仍有效（低电平），则延长命令的保持时间。 00000000：保留 00000001：2 个 HCLK 周期（+加上由 NWAIT 信号变低引入的等待周期） … 11111111：256 个 HCLK 周期（+加上由 NWAIT 信号变低引入的等待周期）（复位后的默认状态）
MEMSETUPx	Bit 7-0	R/W	**通用存储器 x 建立时间** 当 NAND Flash 在 socket x 上对通用存储空间进行读或写操作时，该位设置了在发送命令（NWE，NOE）之前的 HCLK 时钟周期数（NAND+2）来

			建立地址。 00000000：1 个 HCLK 周期 11111110：255 个 HCLK 周期 11111111：保留

30. 5. 2. 7 EBI 属性存储器空间时序寄存器（EBI_PATTR2~3）

EBI 属性存储器空间时序寄存器（EBI_PATTRx, x=2~3）

EBI_PATTR2 偏移地址：$06C_H$

EBI_ PATTR3 偏移地址：$08C_H$

EBI_ PATTR2~3 复位值：$11111100\_11111100\_11111100\_11111100_B$

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

ATTHIZTx	Bit 31-24	R/W	**属性存储器 x 数据总线高阻抗时间** 设置了 HCLK 的时钟周期数。当 NAND Flash 在 socketx 上开始对属性存储空间进行写操作后，在设置的该段时钟周期内，数据总线保持高阻抗。仅对写操作有效。 00000000：0 个 HCLK 周期 11111110：255 个 HCLK 周期 11111111：保留
ATTHOLDx	Bit 23-16	R/W	**属性存储器 x 保持时间** 当 NAND Flash 在 socket x 上对属性存储空间进行读或写操作时，该位设置了当发送命令（NWE，NOE）后，地址（和写操作数据）的保持时间。 00000000：保留 00000001：1 个 HCLK 周期 11111110：254 个 HCLK 周期 11111111：保留
ATTWAITx	Bit 15-8	R/W	**属性存储器 x 等待时间** 当 NAND Flash 在 socket x 上对属性存储空间进行读或写操作时，该位设置了保持命令（NWE，NOE）的最小时钟周期 HCLK（+1）。当该设置值结束时，如果等待信号（NWAIT）仍有效（低电平），则延长命令的保持时间。 00000000：保留 00000001：2 个 HCLK 周期（+加上由 NWAIT 信号变低引入的等待周期） … 11111111：256 个 HCLK 周期（+加上由 NWAIT

			信号变低引入的等待周期）（复位后的默认状态）
ATTSETUPx	Bit 7-0	R/W	**属性存储器 x 建立时间** 当 NAND Flash 在 socket x 上对属性存储空间进行读或写操作时， 该位设置了在发送命令（NWE，NOE）之前的 HCLK 时钟周期数（+1）来建立地址。 00000000：1 个 HCLK 周期 11111110：255 个 HCLK 周期 11111111：保留

30. 5. 2. 8 EBI ECC结果寄存器 2/3（EBI_ECCRESULT2/3）

EBI ECC 结果寄存器 2/3（EBI_ECCRESULT2/3）

EBI_ECRRESULT2 偏移地址：074_H

EBI_ ECRRESULT3 偏移地址：094_H

EBI_ ECRRESULT2~3 复位值：$00000000\_00000000\_00000000\_00000000_B$

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

ECCRESULTx	Bit 31-0	R/W	**ECC 结果** 该寄存器为 ECC 计算逻辑得出的计算结果。 下表显示了该寄存器得有效位。

ECCPSIZE[2:0]	页尺寸	ECC 有效位
000	256	ECC[21:0]
001	512	ECC[23:0]
010	1024	ECC[25:0]
011	2048	ECC[27:0]
100	4096	ECC[29:0]
101	8192	ECC[31:0]

表 30-35 ECC 结果有效位

第31章 QSPI Flash控制器

31. 1 概述

QSPI Flash 控制器可向串行 Flash 器件提供访问权限。支持高性能双线 SPI 和四线 SPI 标准接口。

31. 2 特性

- 存储器映射的直接操作模式，用来进行 Flash 数据传输和执行 Flash 存储器中的编码
- 软件触发的间接操作模式，用来进行延迟时间短和非处理器密集型的 Flash 数据传输
- 间接状态模式下与外部 DMA 通信的 DMA 外设接口可选
- 局部 SRAM 大小可配置，在间接传输过程中可用来减小 AHB 开销和缓冲 Flash 数据
- 软件 APB 可访问的 Flash 控制寄存器组可执行任何 Flash 命令，包括一次数据传输可高达 8 个字节
- 支持任意时钟频率，包括当前市场上的 133MHz SDR 和 80MHz DDR
- 支持 XIP (Execute in Place)，有时也指连续模式
- 支持 DDR 模式和 DTR 协议
- 支持单线，双线，四线 I/O 指令
- 器件大小可编程
- 写保护区域可编程，可阻止系统写入生效
- 传输事务之间的延时可编程
- 传统模式下允许软件直接访问底层发送和接收 FIFOs，旁路较高层流程
- 可支持独立的异步的 SPI 通信参考时钟
- 串行时钟可配置极性
- 可编程波特率发生器支持生成不同器件时钟
- 包含可提升高速读数据捕捉机制的特性
- 可选择使用调整时钟来进一步提升读数据捕捉
- 可编程中断生成
- 高达 4 个外部器件选择
- 可编程 AHB 解码器，支持每个已连接器件的连续寻址模式和器件之间边界的自动检测
- 支持 BOOT 模式

31. 3 结构框图

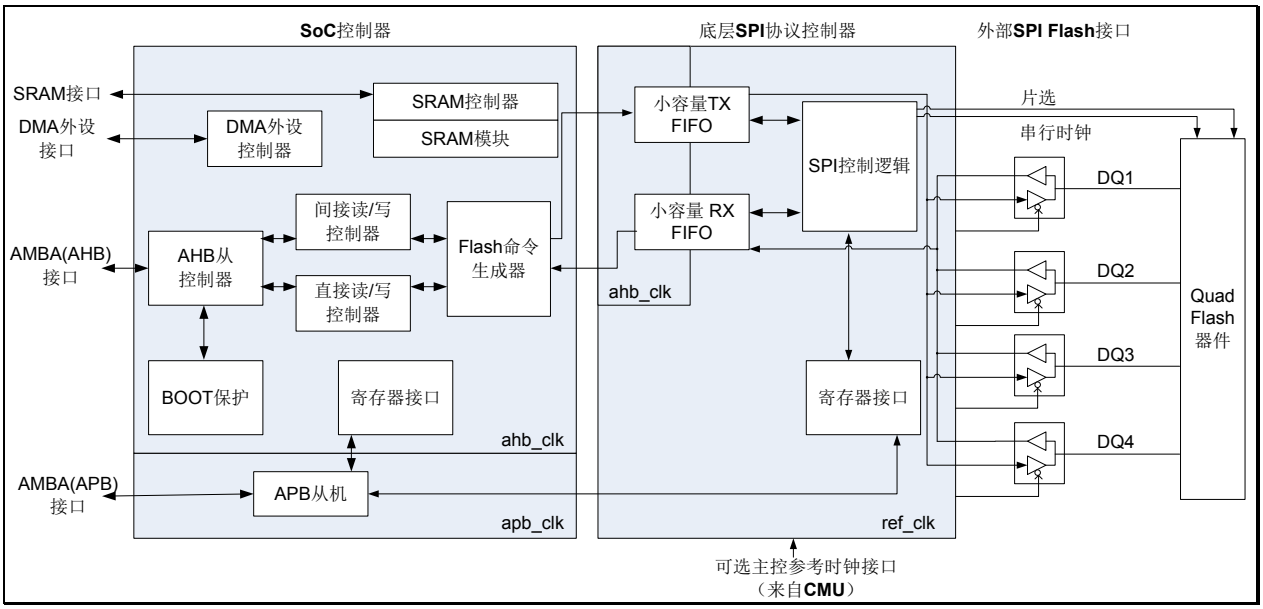


图 31-1 QSPI Flash 控制器结构框图

31.4 功能描述

31.4.1 AHB控制接口

AHB 从控制器可验证接收到的 AHB 访问；对无效请求作出响应；进行任意字节或者半字的重排序；屏蔽违反写保护规则的写操作（仅限直接访问）；向直接访问控制器或者间接访问控制器转发传输请求。

31.4.1.1 AHB接口

AHB 接口遵循 ARM 的 AMBA 3 AHB-Lite 协议规格。不支持传输锁定（HMASLOCK）和采用保护控制信号（HPROT）的传输。

AHB 的数据位宽为 32 位。因此，只允许字节，半字和字访问。关于写操作，只支持递增突发（incrementing bursts），接收到的 wrapping 写突发将会产生错误。支持 INCR16，INCR8，INCR4，INCR 和 SINGLE 的突发类型。关于读操作，所有突发类型，包括 WRAPS，均支持。不论是在突发传输因互连突发终止而造成提前终止，还是在突发传输内从机发生错误访问，从机都将正确运行。

31.4.1.2 AHB地址重映射

QSPI 控制器不会对接收到的错误地址进行具体的地址解码，但有 AHB 地址解码器。如果使能，QSPI Flash 控制器能够检测到每一个独立器件的地址范围，基于 AHB 的地址，做出有效器件自动切换的选择。接收到的 AHB 地址默认直接映射到串行发送至 Flash 器件上的地址。如果 Flash 器件有 24 位地址，将会发送 AHB 地址的低 24 位。

重映射功能将 AHB 总线上的地址映射为 ADDRESS+N，N 为地址重映射寄存器（QSPI_RAR）中的值。重映射功能可通过 QSPI 配置寄存器（QSPI_CR）使能。当软件需要将 BOOT 代码移动至另一个 Flash 区域时，需要用到重映射功能。

31.4.1.3 写保护

为保护 Flash 器件，写保护功能可通过硬件实现，也可通过软件控制。任何检测到的 AHB 写操作，指向受保护的 Flash 区域，将通过 HRESP 错误输出产生一个错误信号。

Flash 器件上的区域，若由多个 Flash“块”定义而成，且起始“块”有特定编号，则该区域可受到写保护。提供 3 个可编程寄存器。QSPI 写保护低位寄存器（QSPI_WPLR）定义了需要保护区域底部的 Flash 块。QSPI 写保护高位寄存器（QSPI_WPHR）定义了需要保护区域顶部的 Flash 块。QSPI 写保护配置寄存器（QSPI_WPCR）为一个 2 位控制寄存器。QSPI_WPCR.WPINV 可将受保护的区域进行反转，使得该区域成为 Flash 存储器中唯一不受保护的区域。QSPI_WPCR.WPEN 为写保护使能位。当 QSPI_WPCR.WPEN 为低电平，Flash 器件不受保护。

AHB 控制器需将接收到的 AHB 地址映射到相关 Flash 块。一个块的大小可通过器件容量配置寄存器（QSPI_DSCR）设置，范围为 1~65K 字节。

31.4.1.4 访问转发

为了合法访问，AHB 控制器会将所有的 AHB 访问转发给 2 个访问控制器中的一个：直接访问控制器或间接访问控制器。假设直接访问控制器已通过 QSPI 配置寄存器（QSPI_CR）使能，则默认所有 AHB 访问将会被转发至直接访问控制器。在转发任何访问至间接访问控制器前，必须由软件先配置。该过程在间接访问控制器章节进行了全面的讲解。如果直接访问控制器禁止，任何不能转发至间接访问控制器的 AHB 访问将被立即停止，并产生 HRESP 错误。如果直接访问控制器使能，同样的访问将由直接访问控制器转发和服务。

31.4.1.5 顺序访问检测和突发长度

为了使性能最大化，将不使用 AHB 接口信号 htrans 来识别顺序和非顺序访问。Htrans 的问题在于它将一个新的 AHB 突发传输启动时的所有访问都识别为非顺序访问，1K 边界以后的所有访问也识别为非顺序访问。取而代之的方法是，通过比较当前地址和上一个地址来检测非顺序访问。如果访问方向（写/读）已改变，则视为非顺序访问。如果读访问的大小（字节/半字/字）已改变，则同样视为非顺序访问。如果 AHB 地址解码器使能并检测到器件之间的切换，则需生成新的 SPI 传输，因为需要初始化当前选中的器件。该情况需视为非顺序访问，尽管地址为连续地址。否则，如果当前传输地址与上一个传输地址为连续地址，则视为顺序访问。

另外，控制器不采用 AHB 的 hburst 信号来识别突发长度。突发传输将继续，直到收到非顺序访问。

31.4.1.6 AHB地址解码器

控制器可连接最多 4 个 Flash 器件。新兴的存储器已达到 2Gb 的容量并仍在上升。连接多个 Flash 器件时，为确保高性能的连续数据传输，可使能 AHB 地址解码器。AHB 总线发出访问的地址范围被整合成连续的格式。无需利用额外步骤来检测器件之间的边界，就可使软件驱动来管理访问。基于 AHB 接口的实际地址，控制器可自动进行切换。地址边界由硬件进行计算。每个连接器件可支持的容量大小为 512Mb 到 4Gb。

AHB 地址解码器仅限直接读传输时使用。如果之前有写访问发出，为避免切换后轮询各个器件，要求需读取的数据必须稳定。为确保数据的稳定性，推荐在对每个器件进行第一次读操作时不要使用 AHB 解码器。

31. 4. 2 直接访问控制器（DAC）

直接访问是指 AHB 访问直接触发对 Flash 存储器进行读或写操作。该操作为存储器映射，可访问并直接执行外部 Flash 存储器中的编码。任何接收到的不在可编程间接触发区域内的 AHB 访问都被假想为直接访问，并由直接访问控制器响应。注意，使用直接访问控制器的访问将不使用嵌入 SRAM。当进行读或写突发操作时，AHB 将会节流，等待状态的数量取决于控制器的延时。延时已设计的尽可能小。当 XIP 读指令使能时，延时将保持最小值。

当响应 AHB 读操作时，DAC 将会发送一个额外的下游访问，而不仅仅是对 AHB 单个 burst 的转发。该访问将不会显现于系统接口。该操作为一个预读操作，确保底层 SPI 核运行于最大带宽。这里定义的 AHB 突发不同于由 AMBA 定义的 AHB 突发。

这里的 AHB 突发定义如下：

1. AHB 突发的第一次访问由以下定义：
   a) 非连续 AHB 访问（基于地址比较来判断是否为非连续，而非基于 htrans）
   b) 非连续或连续 AHB 访问（当下游模块处于空闲状态）
2. AHB 突发的最后一次访问由连续 AHB 访问定义，优先于上面 1 中定义的新突发。
3. AHB 突发的大小为 AHB 访问的数量，从第一次访问到最后一次访问。
4. 每个 AHB 突发的 DAC 请求的数量 = AHB 突发大小 + 1

关于 AHB 写操作，直接访问控制器会触发一系列的写命令（类似于读操作的处理方式），尽管 DAC 写请求的数量等于接收到的 AHB 写请求的数量。写操作时，AHB 控制器将确保 Flash 突发不会超出 Flash 页边界。当检测到页边界时，只有到边界为止的字节访问数会被转发。一个跨页边界的连续直接写请求必须检测为到下游模块的非连续请求，使得下游控制器强制 Flash 器件进入自定时页程序周期。

内核支持跨页分开写，仅限于字对齐的地址。

如果系统发送连续写操作时延迟太久，Flash 写周期可能提前启动，减少器件的有效寿命。注意，如若系统不能保证及时提供需要写入的数据，则采用间接写操作来避免这个问题。

Flash 擦操作由软件使用编程接口触发。

一旦页编程周期启动，在允许后续 AHB 访问完成之前，QSPI Flash 控制器将自动轮询写周期直到完成。该操作通过将后续 AHB 直接访问控制在等待状态来实现。

31. 4. 3 间接访问控制器（INDAC）

31. 4. 3. 1 间接读控制器

间接操作的目的在于，无需 AHB 访问触发的情况下，从 Flash 存储器读取有效字节数。间接操作由软件通过间接读传输控制寄存器（QSPI_IRTR），间接读传输数据深度阈值寄存器（QSPI_IRTWR），间接读传输起始地址寄存器（QSPI_IRTSAR）和间接读传输字节数寄存器（QSPI_IRTNR）控制和触发。该模块与底层 SPI 协议状态机模块通信，展开有效优化的 Flash 读突发操作，将读数据放在本地 SRAM 模块中，为和外部 AHB 主机进行快速及低延时传输做准备。

默认状态下，间接读控制器禁止。在使能前，软件必须配置读取数据的多少和起始地址。间接读传输起始地址寄存器（QSPI_IRTSAR）定义起始地址，间接读传输字节数寄存器（QSPI_IRTNR）定义字节数。一次可同时配置两个间接操作。第二个间接操作可在第一个操作过程中被触发。上一次间接操作完成与下一次间接操作开始之间允许一段短时间的周转。

间接操作读取的字节总数不受 SRAM 大小限制。SRAM 的大小仅会限制发送至 DMA 的请求数量（在 DMA 外设接口模块使能的情况下）。发生 SRAM 上溢时，控制器将压回 Flash 读直到 SRAM 有空余空间。通过完成当前读突发，在 SPI 接口处回压读操作，等待直到 SRAM 有剩余空间后，在上一个突发操作终止的地址发送新的读突发操作。

通过发送 AHB 读到 QSPI 控制器，总线主机能够获取控制器从外部 Flash 存储器读取的数据。接收的读访问 AHB 地址必须在 AHB 间接触发地址（通过间接 AHB 地址触发寄存器（QSPI_IATR）配置）到（AHB 间接触发地址 + 2**（间接 AHB 触发地址范围）- 1）的范围内。该范围的默认值等于 16，可有效地处理 16 次及以下的突发操作。实际的 AHB 地址必须在间接范围内同意 SRAM 为源。每个有效的 AHB 间接读将会引起内部 SRAM 出栈，从而将 AHB 地址从 Flash 地址中分离出来非直接映射。因此，AHB 间接触发地址与 Flash 地址毫无任何关系。触发任何有效间接读之后，SRAM 被视为源，而非 Flash 存储器阵列。间接读的 Flash 地址由间接读传输起始地址寄存器（QSPI_IRTSAR）获取。假设请求读取的数据已经在 SRAM 时，QSPI 控制器一旦收到 AHB 访问地址，SRAM 中数据将被取走，这就实现了读突发的最小延时响应。当数据从 SRAM 中读取，QSPI 控制器将释放 SRAM 中的相关资源。

如果接收到的 AHB 读访问的地址不在上述范围内，则该访问将不会由间接控制器完成，而是由直接访问控制器完成。

如果接收到的 AHB 读访问的地址在上述范围内，但读取的数据不存在于 SRAM 中，则 AHB 将处于等待状态直到从 Flash 读取到数据并压栈至 SRAM 中。

如果接收到的 AHB 读突发的访问元素横跨 AHB 间接触发范围，则间接触发范围内的访问将由间接访问控制器处理，其余部分由直接访问控制器处理。注意，这可能是软件配置错误。

总线主机仅允许发送 32 位 AHB 读，直到间接传输结束，这可以减少 SRAM 控制逻辑的复杂程度。在读最后非 32 位对齐数据时，总线主机可发送 16 位（半字）或者字节访问来完成此次传输。同样总线主机可以始终发送 32 位读，末尾非 32 位对齐的数据高位用 0 填充。

理想状态下，在执行读操作时，SRAM 将保持满的状态。SRAM 的数据深度状态（Fill level）可直接由软件直接读取（SRAM 数据深度状态寄存器（QSPI_SFLR））。如果 DMA 外设接口控制器使能，将自动请求外部 DMA 通过 AHB 以有效数据块（chunks of data）的形式从 SRAM 获取数据，每个数据块皆是整体间接传输的一部分。

通过设置 QSPI_IRTR.RDDIS，可随时取消间接操作。

任何总线主机可发起间接访问。DMA 总线外设接口可分担部分软件开销，有效管理数据传输。另外一个选择是：通过 APB 寄存器，软件直接访问 SRAM 数据深度状态，决定何时从 SRAM 获取数据。当 DMA 外设接口禁止，提供了可通过间接读传输数据深度阈值寄存器（QSPI_IRTWR）访问的数据深度阈值。当 SRAM 数据深度状态超过该数据深度阈值（watermark），会产生一个中断。如果数据深度阈值大于 0，当数据的最后一个字节被 QSPI 控制器读取并放置于 SRAM 中时，即便实际的 SRAM 数据深度状态还未超过数据深度阈值，同样会产生数据深度阈值中断。该功能有助于避免软件追踪已被读取的数据量和复位间接读传输中最后几个字节的数据深度阈值。注意，该数据深度阈值寄存器是一个两用寄存器。当 DMA 外设接口使能时，由硬件来控制 DMA 请求发送的速率。当 DMA 外设接口禁止时，其行为如上文中描述。

为了解间接操作的状态，另外还提供两个中断源。第一个：当完成一个间接操作时，会产生一个中断。第二个：如果发出间接读操作请求，但由于 QSPI 控制器中已缓存 2 个间接操作，而未被接受时，会产生中断。

设置 QSPI_IRTR.RDST 来启动间接读操作，QSPI_IRTR.RDPS 可用来检查状态。

间接读传输流程

◇ 当 DMA 外设控制器使能，需遵循下列流程：

1. 设置 QSPI 配置寄存器（QSPI_CR）
2. 通过间接读传输数据深度阈值寄存器（QSPI_IRTWR）设置 SRAM 数据深度阈值
3. 通过间接读传输起始地址寄存器（QSPI_IRTSAR）设置间接传输的 Flash 起始地址
4. 在间接读传输字节数寄存器（QSPI_IRTNR）中设置需传输字节的数量
5. 在间接 AHB 地址触发寄存器（QSPI_IATR）中设置间接传输 AHB 触发地址
6. 在间接触发地址范围寄存器（QSPI_ITARR）中设置间接传输 AHB 触发地址范围
7. 在 DMA 外设配置寄存器（QSPI_DMACR）中设置单次和突发传输的字节数
8. 设置 QSPI_IRTR.RDST = 1 来触发间接读访问
9. 通过间接读传输控制寄存器（QSPI_IRTR）来轮询间接读操作的完成状态（QSPI_IRTR.RDCS）。注意，该位为写清零（write-to-clear）。或者通过产生间接完成中断来判断间接读操作是否完成。
10. 在相同的寄存器中读取已完成的间接读操作数量。

◇ 当 DMA 外设控制器禁止，需遵循下列流程：

1. 设置 QSPI 配置寄存器（QSPI_CR）
2. 通过间接读传输起始地址寄存器（QSPI_IRTSAR）设置间接传输的 Flash 起始地址

3. 在间接读传输字节数寄存器（QSPI_IRTNR）中设置需传输字节的数量
4. 在间接 AHB 地址触发寄存器（QSPI_IATR）中设置间接传输 AHB 触发地址
5. 在间接触发地址范围寄存器（QSPI_ITARR）中设置间接传输 AHB 触发地址范围
6. 如果使用数据深度阈值中断功能，当数据深度状态超出间接读传输数据深度阈值寄存器（QSPI_IRTWR）设置的阈值，会产生中断。设置数据深度阈值有助于提示软件何时读取间接读传输的下一个部分。注意，如果数据深度阈值设置为一个非 0 值，即便数据深度阈值高于实际的数据深度状态，一旦获取间接传输的最后一个字节并存放于 SRAM 中，就会产生数据深度阈值中断。
7. 设置 QSPI_IRTR.RDST = 1 来触发间接读访问
8. 如果使用数据深度阈值中断功能，则等待数据深度阈值中断发生。否则，轮询 SRAM 的数据深度状态来决定什么时候 SRAM 中有足够的数据来触发获取 AHB 数据。
9. 从 SRAM 中读取期望的数据量。如果还要获取更多数据才能完成间接读传输，则回到步骤 8。否则，继续到 10。
10. 通过间接读传输控制寄存器（QSPI_IRTR）来轮询间接读操作的完成状态。
11. 当间接读操作完成，将产生间接完成中断。

31. 4. 3. 2 间接写控制器

间接写操作的目的是以最有效的方式将大量的数据从处理器或 DMA 写到 Flash 存储器。间接传输在 Flash 器件内部执行尽可能少的写周期，因此可最大化器件的寿命。

间接写操作在软件上可视为间接读的相反过程，通过间接写传输控制寄存器（QSPI_IWTR），间接写传输数据深度阈值寄存器（QSPI_IWTWR），间接写传输起始地址寄存器（QSPI_IWTSAR），间接写传输字节数寄存器（QSPI_IWTNR）来控制和触发。该模块通过外部 AHB 主机来等待写数据的传输，在与现有的传统 SPI IP 内核进行通信前，写数据会存放在自带的 SRAM 中。

默认状态下，间接写控制器禁止。在使能前，软件必须配置数据的多少和起始地址。间接写传输起始地址寄存器（QSPI_IWTSAR）定义起始地址，间接写传输字节数寄存器（QSPI_IWTNR）定义字节数。一次性可同时编程两个间接操作。第二个间接操作可在第一个操作过程中被触发。上一次间接操作完成与下一次间接操作开始之间允许一段短时间的周转。间接写序列类似于间接读序列。

间接操作中，写的字节总数不受 SRAM 的大小限制。SRAM 的大小仅会限制能从外部 AHB 主机接收的数据数量。当总线主机为 DMA，控制器通过 DMA 外设接口要求的数据量不会超出 SRAM 的现有 Fill level。但是，这并不保证 DMA 或者其他总线主机不会试图发送更多的数据，超出 SRAM 所能接收的量。当发生 SRAM 上溢时，控制器会在等待状态下回压 AHB。注意，通过 SRAM 数据深度状态寄存器（QSPI_SFLR）可读取 SRAM 的数据深度状态，避免发生上溢。

总线主机会提供写数据，并通过发送 AHB 写操作发送至 QSPI。接收的写访问 AHB 地址必须在 AHB 间接触发地址（通过间接 AHB 地址触发寄存器（QSPI_IATR）配置）到（AHB 间接触发地址 + 2**（间接 AHB 触发地址范围）- 1）的范围内。该范围的默认

值等于 16，可有效地处理 16 次及以下的突发操作。另外，对于压栈连续地址序列也没有严格要求。实际的 AHB 地址必须在间接范围内同意 SRAM 为源。每个有效的 AHB 间接写将会引起内部 SRAM 出栈，从而将 AHB 地址从 Flash 地址中分离出来。因此，AHB 间接触发地址与 Flash 地址毫无任何关系。触发任何有效间接写之后，SRAM 被视为源，而非 Flash 存储器阵列。间接写的 Flash 地址由间接写传输起始地址寄存器（QSPI_IWTSAR）获取。假设在 QSPI 控制器接收 AHB 访问的时候，SRAM 的状态为非满，则会以最小延时向 SRAM 中压栈数据。

如果接收到的 AHB 写访问的地址不在上述范围内，则该访问将不会由间接控制器完成，而是由直接访问控制器完成。

如果接收到的 AHB 写访问在上述范围内，但 SRAM 状态为满，则 AHB 将处于等待状态直到一部分数据或者全部数据从 SRAM 压栈到 Flash 中。

如果接收到的 AHB 写突发的访问元素横跨 AHB 间接触发范围，则间接触发范围内的访问将由间接访问控制器处理，其余部分由直接访问控制器处理。注意，这可能是软件配置错误。

总线主机仅允许发送 32 位 AHB 写，直到间接传输结束，用来减少 SRAM 控制逻辑的复杂程度。在写最后一个字时，总线主机可发送一个 32 位，16 位（半字）或者一个字节来结束此次传输。如果最后一次传输少于 4 个字节，主机依旧可发送 32 位的数据传输。多余的字节将被丢弃。

当 SRAM 中的字节数量等于或大于一页 Flash 的容量（默认 256 字节）或当 SRAM 中的字节为当前执行的间接传输的所有剩余字节，控制器会向命令发生器发起写突发操作。

通过设置 QSPI_IWTR.WRDIS，可随时取消间接操作。

任何总线主机可发起间接访问。DMA 总线外设接口可分担部分软件开销，有效管理数据传输。另外一个选择是：通过 APB 寄存器，软件直接访问 SRAM 数据深度状态，决定何时向 SRAM 写数据。当 DMA 外设接口禁止，提供了可通过间接写传输数据深度阈值寄存器（QSPI_IWTWR）访问的数据深度阈值。当 SRAM 数据深度状态低于该深度阈值，会产生一个中断。注意，该数据深度阈值寄存器为一个两用寄存器。当 DMA 外设接口使能，由硬件来控制 DMA 请求发送的速率。在该模式下，因为 QSPI 不会发起写操作除非 SRAM 中有至少一页 Flash，或者间接传输的剩余字节在 SRAM 中，DMA 模式下的数据深度阈值需设置为〉=1 个 Flash 页。

为了解间接操作的状态，另外还提供两个中断源。第一个：当完成一个间接操作时，会产生一个中断。第二个：如果发出间接写操作请求，但由于 QSPI 控制器中已缓存 2 个间接操作，而未被接受时，会产生中断。

设置 QSPI_IWTR.WRST 来启动间接写操作，QSPI_IWTR.WRPS 可用来检查状态。

间接写传输流程

◇ 当 DMA 外设控制器使能，需遵循下列流程：

1. 通过间接写传输起始地址寄存器（QSPI_IWTSAR）设置间接传输的 Flash 起始地址
2. 在间接写传输字节数寄存器（QSPI_IWTNR）中设置需传输字节的数量

3. 在间接 AHB 地址触发寄存器（QSPI_IATR）中设置间接传输 AHB 触发地址
4. 在间接触发地址范围寄存器（QSPI_ITARR）中设置间接传输 AHB 触发地址范围
5. 在 DMA 外设配置寄存器（QSPI_DMACR）中设置 DMA 单发和突发传输的字节数
6. 可选：设置间接写传输数据深度阈值寄存器（QSPI_IWTWR）来控制 DMA 请求的发送速率
7. 设置 QSPI_IWTR.WRST = 1 来触发间接写访问
8. QSPI 控制器将利用 DMA 请求接口来要求 DMA 传输数据
9. 通过间接写传输控制寄存器（QSPI_IWTR）来轮询间接写操作的完成状态（QSPI_IWTR.WRCS）。注意，该位为写清零（write-to-clear）。在相同的寄存器中读取已完成的间接写操作数量
10. 当间接写操作完成，会产生间接完成中断。

◇ 当 DMA 外设控制器禁止，需遵循下列流程：

1. 通过间接写传输起始地址寄存器（QSPI_IWTSAR）设置间接传输的 Flash 起始地址
2. 在间接写传输字节数寄存器（QSPI_IWTNR）中设置需传输字节的数量
3. 在间接 AHB 地址触发寄存器（QSPI_IATR）中设置间接传输 AHB 触发地址
4. 在间接触发地址范围寄存器（QSPI_ITARR）中设置间接传输 AHB 触发地址范围
5. 功能上，软件可简单的在一个块内将所有数据写到 SRAM 中。但是，如果写字节数超过 SRAM 的大小，SRAM 极有可能会满，导致 QSPI 花大量时间回压系统 AHB 总线。该时间基于 Flash 的数据速率和器件的页写时间。为避免所有写数据在一个块内发送，软件可利用数据深度阈值中断在合适的时间一次发送一页数据。也可选择，软件直接轮询 SRAM 数据深度状态寄存器（QSPI_SFLR）来识别 SRAM 的空状态，再决定下一部分传输发送的最合适时间。
6. 如果使用数据深度阈值中断，当数据深度状态低于深度阈值，将产生中断。数据深度阈值的设置范围为 0 ~ 页容量。举例：如果页容量为 256 字节，数据深度阈值的合理设置为 10 到 250 之间。当数据深度状态下降到设定值以下，将触发中断。设置数据深度阈值有助于提示软件何时向 SRAM 写入间接写传输的下一页数据。
7. 设置 QSPI_IWTR.WRST = 1 来触发间接写访问
8. 如果当前间接传输剩余的字节数大于一个 Flash 页，则向 SRAM 中写入一个 Flash 页数据；否则，将所有剩余数据发送至 SRAM。
9. 如果间接传输中所有的数据都已发送给 SRAM，跳至步骤 11，等待完成状态。如果还有更多的数据需要传输，则：
   a) 如果使用数据深度阈值中断，则等到中断发生
   b) 使用 SRAM 数据深度状态来判断发送数据的合适时间
10. 循环回步骤 8

11. 可选：间接写操作的完成状态可通过间接写传输控制寄存器（QSPI_IWTR）查询。
12. 当间接写操作完成，会产生间接完成中断。

31.4.3.3 间接访问队列

软件允许间接写控制器和间接读控制器有 2 个间接传输排列等候。支持前一个间接操作完成和后一个操作开始前有短暂周转时间。如果等候队列超出 2 个，则会产生中断。

软件上，间接访问队列通过短时间内触发两次 QSPI_IRTR.RDST 或 QSPI_IWTR.WRST 来实现。在每一次触发 QSPI_IRTR.RDST 或 QSPI_IWTR.WRST 前，间接传输起始地址寄存器（QSPI_IRTSAR/QSPI_IWTSAR）和间接传输字节数寄存器（QSPI_IRTNR/QSPI_IWTNR）必须完成设置。因为这些寄存器将进行周期性更新，在间接传输的周期过程中，硬件需要对这些寄存器保持采样的状态。

内部寄存器块一次只发送一个间接起始触发给关键数据路径块。间接访问控制器中有 2 个独立的数据路径块，可接收和独立采样间接起始触发信息。第一个数据路径块存在于 SRAM 的 AHB 端。当进行间接读时，为读接口；当进行间接写时，为写接口。第二个数据路径块在 SRAM 的 Flash 端。当进行间接读时，为写接口；当进行间接写时，为读接口。这两个块可在不同的时间处理间接传输。举例：进行间接读操作时，当第一个传输中的最后一个字节已写入 SRAM，SRAM 中的 Flash 上的数据路径块即可开始处理第二个队列传输。开始第二个传输前，必须对间接传输起始地址寄存器（QSPI_IRTSAR/QSPI_IWTSAR）和间接传输字节数寄存器（QSPI_IRTNR/QSPI_IWTNR）进行重采样。同样地，当第一个间接传输的所有的 Flash 数据都已从 SRAM 发送至 AHB，AHB 上的数据路径块也将重采样相同的寄存器。

31.4.3.4 间接传输：连续写读

间接写操作过程中，软件允许触发间接读操作。相似地，间接读操作过程中，也允许触发间接写操作。间接写操作优先级更高。

31.4.3.5 访问SRAM

物理上，SRAM 是一个单个端口模块。SRAM 深度可配置。SRAM 被划分为两块区域，低区域部分预留给间接读使用。高区域部分仅作为间接写使用。两部分的容量可通过 SRAM 分块配置寄存器（QSPI_SPR）配置，用户可选择分配 SRAM 地址总线中多少的 bit 给间接读操作。默认地，该寄存器设置 SRAM 的一半给间接读控制器使用。为确保 AHB 读数据总线不是直接由 SRAM 读数据通过组合逻辑提供，间接读数据路径中包含了另一组寄存器。

以下示例展示了在间接读和间接写之间，SRAM（和另一组寄存器）的分配方式。以下示例中，SRAM 深度为 8 位，等同于 256 地址。

- 如果 SRAM 分块配置寄存器（QSPI_SPR）设置为 0x00，256 个地址分配给间接写，1 个地址分配给间接读
- 如果 SRAM 分块配置寄存器（QSPI_SPR）设置为 0x01，255 个地址分配给间接写，2 个地址分配给间接读
- 如果 SRAM 分块配置寄存器（QSPI_SPR）设置为 0x02，254 个地址分配给间接写，3 个地址分配给间接读
- 依次类推，直到.....
- 如果 SRAM 分块配置寄存器（QSPI_SPR）设置为 0xfd，3 个地址分配给间接写，254 个地址分配给间接读
- 如果 SRAM 分块配置寄存器（QSPI_SPR）设置为 0xfe，2 个地址分配给间接写，

255 个地址分配给间接读

- 如果 SRAM 分块配置寄存器（QSPI_SPR）设置为 0xff，1 个地址分配给间接写，256 个地址分配给间接读

注：避免将 SRAM 分块配置寄存器（QSPI_SPR）设置为 0xFF 或者 0x00，因为仅 SRAM 数据深度状态底部的 8 位才能通过软件访问。如果间接读或间接写的数据深度达到 256，当读取数据深度时，会显示为 0。

有 4 个 SRAM 源，它们都在单个 SRAM 端口上进行仲裁和复用。高达 3 个源可在任意时间访问该端口。

- 间接写，写源，位于 SRAM 的 AHB 上。
- 间接写，读源，位于 SRAM 的 Flash 上。
- 间接读，写源，位于 SRAM 的 Flash 上。
- 间接读，读源，位于 SRAM 的 AHB 上。

优先级的仲裁方案如下：

		SRAM 访问优先级
间接写	写入 SRAM（从系统 AHB）	第 3 优先级（不包括 AHB 读请求）
	读取 SRAM（从 QSPI 控制器）	第 2 优先级
间接读	写入 SRAM（从 QSPI 控制器）	第 1 优先级
	读取 SRAM（从系统 AHB）	第 3 优先级（不包括 AHB 写请求）

表 31-1 SRAM 访问优先级

在间接读操作过程中（SRAM 中的 Flash）除写端口外，驱动 4 个源的逻辑不能被认为是单个周期完成。为避免数据丢失，在间接读过程中，必须允许写 SRAM 可以立即完成。因此，该端口拥有最大优先级。

31. 4. 4 DMA外设控制器

外设接口用来触发外部 DMA 进行短延时 AHB 数据突发访问。DMA 外设接口仅用于间接操作模式，数据缓存于自带的 SRAM 可更快速的响应 AHB 请求，并使内核在一个较长的周期内进行底层 Flash 传输。支持 2 个 DMA 请求，一个用于间接读控制器，另一个用于间接写控制器。关于间接读控制器，当数据已从 Flash 获取并写入自有 SRAM 后，QSPI 控制器仅发送 DMA 请求。关于间接写控制器，当触发传输后，QSPI 控制器会立即发送 DMA 请求并一直持续发送直到整个间接写传输完成。间接传输数据深度阈值寄存器（QSPI_IRTWR/QSPI_IWTWR）可改变发送请求的速率。

31. 4. 4. 1 操作顺序

当间接操作被触发，DMA 外设控制器可见所有 Flash 存储器的传输数据（以字节为单位）。控制器会将这些数据分为 DMA 突发请求和单次请求：所有字节数除以突发请求中设置的字节数，余下的除以单次请求中的字节数。软件需确保在这些除法之后无余数。举例，如果从 Flash 要读取的数据总字节为 512，SRAM 固定容量为 256 字节，软件配置的突发传输字节数为 256，则当前 256 个字节已缓冲，SPI 控制器将会触发 DMA 突发请求。当又有 256 个字节在 SRAM 中缓冲完成后才会触发第二次突发请求。由于 SRAM 本身容量为 256 字节，意味着在发送下一个请求前，DMA 会从 SRAM 中获取所有内容。

SRAM 数据深度状态对 DMA 外设控制器可设计为可见。关于间接读，会基于以下状态机发送请求给外部 DMA。

1. 等待 START 触发
2. 如果下一个需发送 BURST 请求，等待直到下面两个条件均为 TRUE。
   a) SRAM 数据深度状态大于或等于 QSPI_DMACR.BNUMB 设置的字节数
   b) SRAM 数据深度状态大于或等于间接传输数据深度阈值寄存器（QSPI_IRTWR/QSPI_IWTWR）中的值

   当 TRUE 时，执行以下：

   - 发送 BURST
   - 计算数据深度状态期望值 = SRAM fill level – num_bytes_requested_to_DMA
   - 如果下一个需发送的为 BURST 请求，跳至 4
   - 如果下一个需发送的为 SINGLE 请求，跳至 5
   - 其他，跳至 1
3. 如果下一个需发送的为 SINGLE 请求，等待直到下面条件均为 TRUE。
   a) SRAM 数据深度状态大于或等于 QSPI_DMACR.SNUMB 设置的字节数

   当 TRUE 时，执行以下：

   - 发送 SINGLE
   - 计算数据深度状态期望值 = SRAM fill level – num_bytes_requested_to_DMA
   - 如果下一个需发送的为 SINGLE 请求，跳至 5
   - 其他，跳至 1
4. 下一个需发送的为 BURST 请求。如果（SRAM fill level – num_bytes_requested_to_DMA）小于 QSPI_DMACR.BNUMB 设置的字节数，跳

至 2。其他，执行以下：

- 发送 BURST
- 计算数据深度状态期望值 = SRAM fill level – num_bytes_requested_to_DMA
- 如果下一个需发送的为 BURST 请求，跳至 4
- 如果下一个需发送的为 SINGLE 请求，跳至 5
- 其他，跳至 1

5. 下一个需发送的为 SINGLE 请求。如果（SRAM fill level – num_bytes_requested_to_DMA）小于 QSPI_DMACR.SNUMB 设置的字节数，跳至 3。其他，执行以下：

- 发送 SINGLE
- 计算数据深度状态期望值 = SRAM fill level – num_bytes_requested_to_DMA
- 如果下一个需发送的为 SINGLE 请求，跳至 5
- 其他，跳至 1

关于间接写，会基于以下状态机发送请求给外部 DMA。

1. 等待 START 触发
2. 如果下一个需发送的为 BURST 请求，等待直到下面两个条件均为 TRUE。

   a) SRAM 的剩余空间大于或等于 QSPI_DMACR.BNUMB 设置的字节数

   b) SRAM 数据深度状态小于或等于间接传输数据深度阈值寄存器（QSPI_IRTWR/QSPI_IWTWR）中的值

   当 TRUE 时，执行以下：

- 发送 BURST
- 计算数据深度状态期望值 = SRAM fill level + num_bytes_requested_to_DMA
- 如果下一个需发送的为 BURST 请求，跳至 4
- 如果下一个需发送的为 SINGLE 请求，跳至 5
- 其他，跳至 1

3. 如果下一个需发送的为 SINGLE 请求，等待直到下面条件均为 TRUE。

   a) SRAM 的剩余空间大于或等于 QSPI_DMACR.SNUMB 设置的字节数

   当 TRUE 时，执行以下：

- 发送 SINGLE
- 计算数据深度状态期望值 = SRAM fill level + num_bytes_requested_to_DMA
- 如果下一个需发送的为 SINGLE 请求，跳至 5
- 其他，跳至 1

4. 下一个需发送的为 BURST 请求。如果（SRAM 剩余空间 – num_bytes_requested_to_DMA）小于 QSPI_DMACR.BNUMB 设置的字节数，跳至 2。其他，执行以下：

- 发送 BURST
- 计算数据深度状态期望值 = SRAM fill level + num_bytes_requested_to_DMA

- 如果下一个需发送的为 BURST 请求，跳至 4
- 如果下一个需发送的为 SINGLE 请求，跳至 5
- 其他，跳至 1

5. 下一个需发送的为 SINGLE 请求。如果（SRAM 剩余空间 – num_bytes_requested_to_DMA）小于 QSPI_DMACR.SNUMB 设置的字节数，跳至 3。其他，执行以下：

   - 发送 SINGLE
   - 计算 fill level 期望值 = SRAM fill level + num_bytes_requested_to_DMA
   - 如果下一个需发送的为 SINGLE 请求，跳至 5
   - 其他，跳至 1

关于间接读操作，间接读传输数据深度阈值寄存器（QSPI_IRTWR）可定义控制器发送第一个 DMA 请求的最小数据深度状态，设置的值越大，在 DMA 获取之前，缓存在 SRAM 中的数据就越多。关于间接写操作，间接写传输数据深度阈值寄存器（QSPI_IWTWR）可定义控制器发送第一个 DMA 突发/单次请求的最大数据深度状态。间接传输数据深度阈值寄存器（QSPI_IRTWR/ QSPI_IWTWR）可使系统在短周期内选择性的集中于 AHB 传输。默认地，间接传输数据深度阈值寄存器（QSPI_IRTWR/QSPI_IWTWR）被复位为 0，意味着外设控制器可尽快发送 DMA 请求。在间接读的情况下，意味着 SRAM 中有足够的数据进行突发请求或单次请求（如果剩余字节小于突发请求的设置值）。注意，如果 SRAM 为空，DMA 外设不可发送间接读 DMA 请求；如果 SRAM 为满，DMA 外设不可发送间接写请求。

DMA 外设接口可通过软件禁止。当接口禁止，将不会发送任何通道请求。注意，如果除了 DMA 以外的 AHB 主机将进行间接数据传输，DMA 外设接口必须被禁止。

31.4.5 软件触发指令生成器（STIG）

直接和间接访问控制器用来进行数据传输。为进行擦除和访问易失性和非易失性配置寄存器，传统 SPI 状态寄存器，其他的状态/保护寄存器，需要有一个独立的软件控制器。软件触发指令生成器 STIG 由 Flash 命令控制寄存器（QSPI_FCR）通过设置命令发送去 Flash 器件来控制。这是一个通用寄存器，可用来执行 Flash 器件支持的扩展 SPI 协议的任何指令。指令如果不符合 Flash 器件规格可导致未知的控制器行为。读时，当命令已响应（QSPI_FCR.CMDS 从 1 翻转为 0），最多 8 个数据可放置于 Flash 命令读数据寄存器（QSPI_FCRLR/QSPI_FCRHR）。写时，写数据应该放置在 Flash 命令写数据寄存器（QSPI_FCWLR/QSPI_FCWHR）。

实现以上功能的代码保存在 QSPI 的 APB 寄存器块内。

31.4.5.1 响应STIG请求

STIG 请求会致使 QSPI Flash 控制器询问 Flash 命令控制寄存器（QSPI_FCR），决定该以什么方式发送多少字节去 Flash 器件。QSPI_FCR.OPCODE 指示要发送的指令，始终首先压栈。地址发送紧接在指令后面，地址大小同样在此寄存器中配置。地址本身保存在 Flash 命令地址寄存器（QSPI_FCAR）。Dummy 字节此后发送（字节数量仍在 Flash 命令控制寄存器（QSPI_FCR）中配置）。写/读数据的字节数同样在 Flash 命令控制寄存器（QSPI_FCR）中配置。写时，最多可发送 8 个字节（保存于 Flash 命令写数据寄存器（QSPI_FCWLR/QSPI_FCWHR））。读时，当读数据已从 Flash 器件获取，QSPI Flash 控制器会将读数据保存在 Flash 命令读数据寄存器（QSPI_FCRLR/QSPI_FCRHR）。

当 QSPI Flash 控制器开始响应 STIG 请求，QSPI_FCR.CMDS 置 1，表示正在执行命令中。

当 QSPI Flash 控制器处于自动轮询状态，响应 STIG 请求会有一点不同。在一个程序操作过后，大部分器件都不可访问直到写操作完成。它们中的一些器件还有可能会停止编程页。QSPI_PFSR.PSV 指示了有效自动轮询状态。当发生 STIG 请求后，QSPI Flash 控制器会立即发送合适的 OPCODE 去存储器。在响应 STIG（自动轮询中）过程中，命令执行的状态位保持，其他位（比如 ADDRESS 或 DUMMY 位）都被禁止。

另外还有一个可配置选项：在每个重复的轮询操作之间加入延时（延时由 QSPI_WCR.PREPD 定义）。该功能的作用是用来释放 SPI 带宽。

31.4.6 直接/间接访问控制器和STIG间的仲裁

当多个寄存器同时有效，一个简单的固定优先级仲裁方案可用来对每个接口进行仲裁，访问外部 Flash。固定优先级定义如下，1 为最高优先级。

1. 间接访问写
2. 直接访问写
3. STIG
4. 直接访问读
5. 间接访问读

等待响应时，每个控制器都会被回压。

31.4.7 SPI命令转换

由直接访问控制器，间接访问控制器或 STIG 发送的请求会转换成字节传输序列，发送去下游。下例显示的是一个字节无序列读：

- INSTRUCTION OPCODE → ADDRESS → Mode Byte → Dummy Bytes → 1 byte of don't care

当进行序列访问时，每一次读，都会在上面的序列最后向 Flash 器件压栈一个额外的字节。目的是为了每个传输字节之间没有空隙。

实际的发送去 Flash 器件的序列由要求的传输决定，是否该传输是非序列的还是序列的，是否器件已在 XIP 模式下配置完成，以及主要器件指令类型寄存器（器件读指令寄存器（QSPI_DRIR）/器件写指令寄存器（QSPI_DWIR））的状态。

写时，在发送写序列前，Flash 器件内的写使能锁存（WEL）必须为高电平。在通过直接或间接访问控制器（DAC/INDAC）触发写命令前，QSPI Flash 控制器将自动发送写使能锁存命令。为提高灵活性和性能，用户可设置 QSPI_DWIR.WELD 来关闭该功能。WREN 的 OPCODE 为 0x06，在器件之间共用。

当不再接收来自直接或间接访问控制器的写请求，并且所有请求都已发出，则 Flash 器件将自动开始页编程写周期。此时接收到的请求将会保持在等待状态，直到写周期完成。QSPI Flash 控制器会自动轮询 Flash 器件的传统 SPI 状态寄存器，来判断写周期是否完成：发送 RDSR OPCODE 至 Flash 器件，一直等待直到器件本身显示写周期已完成（直到 Write In Progress bit[0]已清零，写使能锁存位也已清零）。WREN 和 RDSR 器件指令是仅有的两个由控制器在底层发送的指令。任何其他的具体指令都应发送至器件，通过 STIG 发送 Flash 命令来单独处理。

31.4.8 Flash指令类型选择

为了发送正确的读写 OPCODE，软件应初始化 QSPI 器件读指令寄存器（QSPI_DRIR）和 QSPI 器件写指令寄存器（QSPI_DWIR）。这两个寄存器中，可设置指令 OPCODE，指令类型，边沿模式（DRR 或 SDR）和选择单个，双个还是 4 个引脚来进行地址和数据传输。为确保控制器可在复位状态中运行，寄存器会复位到兼容 SIO 器件的 OPCODE。

器件读指令寄存器（QSPI_DRIR）中包含指令类型字段，器件写指令寄存器（QSPI_DWIR）无指令类型字段。如果软件将指令类型设置为非 0 值，器件读指令类型寄存器（QSPI_DRIR）和器件写指令类型寄存器（QSPI_DWIR）中的地址传输类型和数据传输类型位无关。软件支持不太常见的 Flash 指令，在 2 或 4 通道发送 OPCODE，地址和数据。注意，对于那些支持 2 或 4 通道发送 OPCODE 的指令的器件，这些指令的名字与 Flash 数据手册上的命名不一致。其中一个支持这些指令的器件为 Numonyx（Micron）N25Q128。其他的读指令被称为 DCFR 和 QCFR。写指令为 DCPP 和 QCPP。下表以 N25Q128 作为参考，展示了软件该如何配置 QSPI。

◇ 读

OPCODE	OPCODE 发送通道数	ADDRESS/ DUMMY MODE 发送通道数	DATA Bytes 发送字节数	QSPI 指令类型（器件读指令寄存器）	QSPI 地址 Xfer 类型（器件读指令寄存器）	QSPI 数据 Xfer 类型（器件读指令寄存器）
READ	1	1	1	0	0	0
FAST_READ	1	1	1	0	0	0
DOFR (Dual O/P Fast Read)	1	1	2	0	0	1
DIOFR (Dual O/P Fast Read)	1	2	2	0	1	1
QOFR (Quad O/P Fast Read)	1	1	4	0	0	2
QIOFR (Quad O/P Fast Read)	1	4	4	0	2	2
DCFR (Dual Command Fast Read)	2	2	2	1	Don’t Care	Don’t Care
QCFR (Quad Command Fast Read)	4	4	4	2	Don’t Care	Don’t Care

◇ 写

OPCODE	OPCODE 发送通道数	ADDRESS/ DUMMY MODE 发送通道数	DATA Bytes 发送字节数	QSPI 指令类型（器件写指令寄存器）	QSPI 地址 Xfer 类型（器件写指令寄存器）	QSPI 数据 Xfer 类型（器件写指令寄存器）
PP	1	1	1	0	0	0
DIFP (Dual Input Fast Program)	1	1	2	0	0	1
DIEFP (Dual Input Extended Fast Program)	1	2	2	0	1	1
QIFP (Quad Input Fast Program)	1	1	4	0	0	2
QIEFP (Quad Input Extended Fast Program)	1	4	4	0	2	2
DCPP (Dual Command Fast Program)	2	2	2	1	Don't Care	Don't Care
QCPP (Quad Command Fast Program)	4	4	4	2	Don't Care	Don't Care

有几组器件（例如：Numonyx（Micro）N25Q512A）能够在双数据速率（DDR，也可叫做双传输速率模式（DTR））模式下处理读操作。在专有命令类型工作过程中，这些器件均可在上升沿和下降沿发送数据。这使得控制器可在低于 2 倍 spi_clk 频率的情况下保持流量。QSPI_DRIR.DDRM 可告知 QSPI Flash 控制器，写入读 OPCODE 位段的 OPCODE 可处理 DDR 命令类型。读数据捕捉寄存器（QSPI_RDCR）能在 DDR 模式下移动传输数据。默认地，数据移位 1 个时钟周期可确保在 DDR 传输中保持时间大于 0。对于高 ref_clk 频率来说，位移 1 个时钟可能不够。DDR 命令的配置与读表中列出的 SDR 命令兼容。

Micron 器件中的 MT25 系列还加入了 DTR 协议，可处理 DTR 模式中的所有命令。根据专门的 OPCODE，DTR 读命令可检测 DTR 模式。因此，OPCODE 必须作为 STR 发送。当 DTR 协议使能，器件无需 OPCODE 来检测边沿模式，因为可从配置寄存器中（QSPI_CR）的易失性和非易失性位来识别。

31. 4. 9 APB接口与寄存器模块

APB 接口利用 Flash 命令控制寄存器（QSPI_FCR）来进行由软件控制的 Flash 访问。APB 接口提供单个寄存器块，包含了可配置的寄存器组。该寄存器组由 APB 时钟定时。

31. 4. 10 SRAM模块

间接操作模式需要存储器模块。存储器的引脚连接在顶层，使得积分器可选择使用 SRAM

存储器，寄存器阵列或 FLOP 组。存储器模块仅有一个端口。存储器模块的深度可选且由具体应用决定。

31.5 编程指引

在与 Flash 器件通信前，软件负责配置 QSPI 控制器。

在 QSPI 控制器通过 QSPI_CR.EN 使能前，控制器静态配置位需设置完成。该目的是：在未实施亚稳定保护的路径上避免时钟边界问题。如果用户希望更改控制器配置，在重新配置前，建议将 QSPI 的使能位设置为禁止。

31.5.1 复位后配置QSPI控制器

QSPI 控制器可在合适的状态唤醒后，使用直接访问控制器进行基本读写。所有目标器件均支持基本读（opcode 0x03）和基本写（opcode 0x02）操作。控制器唤醒时，波特率分频为 32。如果目标器件不使用 3 个地址字节，器件容量配置寄存器（QSPI_DSCR）必须更改为一个合适的值。所有在该项目中测试的器件，其页容量为 256 字节，无需再更新寄存器。

若非硬性要求，在使能 QSPI 控制前，软件应慎重使能写保护功能。需设置写保护低位寄存器（QSPI_WPLR），写保护高位寄存器（QSPI_WPHR）和写保护配置寄存器（QSPI_WPCR），器件容量配置寄存器（QSPI_DSCR）中的每个器件块的字节数也需设置。

初始化后，当控制器和 DAC 使能后，软件可读取和写入 Flash 器件。简单对配置寄存器（QSPI_CR）写入就可使能控制器。注意，不要更改该寄存器中波特率分频和 CPOL/CPHA 的默认值。建议写入值为 0x00780081。

31.5.2 QSPI控制器配置优化

为更优化地访问 Flash，软件需准确配置控制器。

1. 等待直到挂起的 STIG 或 INDAC 操作都已完成或轮询 QSPI_CR.IDLES
2. 禁止 DAC（QSPI_CR.DACEN）。不必要完全禁止 QSPI 控制器（QSPI_CR.EN）
3. 根据用户希望使用的指令类型来更新器件指令配置寄存器（QSPI_DRIR/QSPI_DWIR）
4. 如果器件指令配置寄存器中的模式位（QSPI_DRIR.MBEN）使能，更新模式位寄存器（QSPI_MBR）
5. 如果内容不正确，更新器件容量配置寄存器（QSPI_DSCR）。注意，初始化后，部分或整个寄存器都有可能已更新。进行读写时，地址字节数是一个关键配置。进行任何写操作时，需要设置每页的字节数。只有在使用写保护功能的情况下，才需要每个器件块的字节数。如果目标器件的默认值正确，或者如果某些值（不包括地址字节数）不正确，并不允许器件写操作。
6. 更新 QSPI 器件延时寄存器（QSPI_DDLR）。该寄存器允许用户在每次 Flash 访问后，对如何驱动片选进行调整。原因是每个器件可能有不同的时序要求。当串行时钟频率增加，时序要求将变得越来越重要。注意，该寄存器中设置的值基于 ref_clk 周期。举例：在 CS 无效后重新设置为有效前，ATMEL 器件需要至少 50ns 的时间。默认情况下，控制器仅会提供最小 1 SCLK 周期。当器件运行于 100MHz，SCLK 周期仅为 10ns，因此需要额外的 40ns。当 ref_clk 运行于 400MHz（2.5ns 周期），该寄存器中的 CSDA 位至少要设置为 16。在自动轮询阶段，可增加该延时。写完成控制寄存器

（QSPI_WCR）中可定义轮询重复延时。

7. 如有需要，更新地址重映射寄存器（QSPI_RAR）。仅影响 DAC 路径。
8. 如有需要，如果在过往的初始化中还未设置，则设置和使能写保护低位寄存器（QSPI_WPLR），写保护高位寄存器（QSPI_WPHR）和写保护配置寄存器（QSPI_WPCR）。
9. 通过中断屏蔽寄存器（QSPI_IMR）使能需要的中断。
10. 在配置寄存器（QSPI_CR）中设置波特率分频，定义目标器件所需的时钟频率。
11. 更新读数据捕捉寄存器（QSPI_RDCR）。当捕捉读数据时，该寄存器可延时。当由器件到控制器的读数据路径很长且器件时钟频率很高，该寄存器的设置也会有所帮助。
12. 通过配置寄存器（QSPI_CR）使能 QSPI 控制器和 DAC。

31.5.3 Flash命令控制寄存器的使用（STIG操作）

Flash 命令控制寄存器（QSPI_FCR）提供了一种灵活可编程的方式访问 Flash 器件，被称为 STIG 操作。指令 OPCODE，地址字节数，地址本身，dummy 字节数，写数据字节数，写数据本身和读数据字节数，均可设置。一旦设置完成，软件可通过 QSPI_FCR.CMDT 触发命令，轮询 QSPI_FCR.CMDS 等待被接受。当该位变为无效，就可触发另一个 STIG。软件可用此种访问 Flash 的典型方法来访问 Flash 器件寄存器，也可进行擦除操作。尽管最多 8 个数据字节（由 Flash 命令读和写数据寄存器（QSPI_FCRLR/QSPI_FCRHR/QSPI_FCWLR/QSPI_FCWHR）定义）可一次读或写，该方法同样可用来访问 Flash 阵列本身。

对比 AHB 的所有其他读访问，该接口发送的命令拥有更高优先级。

31.5.4 SPI传统模式

SPI 传统模式允许软件直接访问内部 TX-FIFO 和 RX-FIFO，因此可绕过直接，间接和 STIG 控制器。

传统模式允许用户发送任意 Flash 指令，但会花费太多的软件开销，确保了 FIFO fill level 的有效管理。传统 SPI 内核是双向传输，当片选使能，数据可在任一方向连续传输。即使驱动器只希望从 Flash 器件读取数据，dummy 数据必须写出确保片选保持有效。对于写传输，反之亦然。这意味着，向一个器件（3 个地址字节）读 4 个字节，软件需向 TX FIFO 中写入一共 8 个字节。第一个字节为指令 OPCODE，接下来的 3 个字节为地址，最后 4 个字节为 dummy 数据。类似的，因 TX FIFO 中写入了 8 个字节，软件会期望有 8 个字节返回至 RX FIFO 中。前 4 个字节会被丢弃，后 4 个字节为所需的读数据。

由于 TX FIFO 和 RX FIFO 深度有限，软件有责任保持 FIFO 的深度，确保在指令执行过程中 TX FIFO 不会下溢，RX FIFO 不会上溢。这可能会增加软件开销。当数据深度超过深度阈值时，将会产生中断。

当传统模式使能，软件可访问 TX FIFO，通过 AHB 接口向 QSPI 控制器的任意地址内写入任意值。当传统模式使能，软件可访问 RX FIFO，软件可通过 AHB 接口向 QSPI 控制器读取任意地址。

31. 5. 5 进入和退出XIP模式

31. 5. 5. 1 从POR进入XIP模式

如果 XIP 以非易失性配置的使能，则 XIP 模式可以非易失性方式进入。Flash 器件中仅一小部分支持该功能。由于 Flash 器件唯一能识别的操作为 XIP 读操作，软件无法通过轮询 Flash 状态寄存器（QSPI_PFSR）读发现 POR 后 XIP 的状态。

如果已得知器件会从 POR 进入 XIP，QSPI_MBR 寄存器和 QSPI_CR.XIPIM 应在初始 BOOT 中设置完成。

如果不知道器件会从 POR 进入 XIP，且连接的 Flash 器件支持从 POR 进入 XIP，则软件通过 STIG 命令发送 XIP 退出命令来尝试退出 XIP 模式。软件需了解器件的模式位要求，因为每个器件的 XIP 进入和退出都不同。Micron N25Q 和 MT25 支持 XIP 从 POR 进入/退出。退出 XIP 模式时，需设置 OPCODE 为 8'h00，地址字节数为 3，dummy 周期数为 16，读字节数为 1。地址需配置为{8 模式位，16'h0000}。Micron 器件模式位需设置为 8'b10000000。

注意，通过 Flash 命令控制寄存器（QSPI_FCR）发送 NVCR 命令可使能从 POR 进入 XIP。直到下一个 POR 序列时，才会生效。

31. 5. 5. 2 其他进入XIP模式的情况

大部分 Flash 器件支持 XIP 模式。但 Flash 生厂商并没有一致的标准。大部分是在地址字节后直接发送签名位。其他，像 Micron 器件，采用 signature bits 的同时，也要求写 Flash 器件配置寄存器来使能 XIP。对于必须服从该控制器的 Flash 器件，需遵循以下步骤进入 XIP 模式。

Micron N25Q 和 MT25（不支持基本 XIP）

通过设置 Flash 器件内的 VCR[3]来使能 XIP 模式。利用 Flash 命令控制寄存器（QSPI_FCR）写 VCR 来发送 VCR 写命令，步骤如下：

- 禁止直接访问控制器和间接访问控制器，确保没有新的 AHB 读访问发送到 Flash 器件。
- 配置 Flash 命令控制寄存器（QSPI_FCR），发送 VCR 写至 Flash 存储器。
- 设置模式位寄存器（QSPI_MBR）中的 XIP 模式位为 8'b0000000。
- 通过设置 QSPI_CR.XIPNX 来使能自有控制器的 XIP 模式。
- 如有需要，重新使能直接访问控制器，间接访问控制器。

其他 Micron 器件（支持基本 XIP）

- 禁止直接访问控制器和间接访问控制器，确保没有新的 AHB 读访问发送到 Flash 器件。
- 设置模式位寄存器（QSPI_MBR）中的 XIP 模式位为 8'b1000000。
- 通过设置 QSPI_CR.XIPNX 来使能自有控制器的 XIP 模式。
- 如有需要，重新使能直接访问控制器，间接访问控制器。

Winbond 器件

- 禁止直接访问控制器和间接访问控制器，确保没有新的 AHB 读访问发送到 Flash 器件。
- 设置模式位寄存器（QSPI_MBR）中的 XIP 模式位为 8'b0010000。

- 通过设置 QSPI_CR.XIPNX 来使能自有控制器的 XIP 模式。
- 如有需要，重新使能直接访问控制器，间接访问控制器。

Spansion 器件

- 禁止直接访问控制器和间接访问控制器，确保没有新的 AHB 读访问发送到 Flash 器件。
- 设置模式位寄存器（QSPI_MBR）中的 XIP 模式位为 8'b1010000。
- 通过设置 QSPI_CR.XIPNX 来使能自有控制器的 XIP 模式。
- 如有需要，重新使能直接访问控制器，间接访问控制器。

31. 5. 5. 3 退出XIP模式

为退出 XIP 模式，软件需禁止直接访问控制器和间接访问控制器，确保没有新的 AHB 读访问发送到 Flash 器件。然后将模式位设置为其他任何值，而不是相应 Flash 器件规格上的特定模式位。接着软件需复位 QSPI_CR.XIPNX。

注意，在禁止内部 XIP 模式状态前，Flash 器件必须见到读指令，意味着，XIP 模式将一直保持有效直到下一个读指令被响应。需确保 XIP 模式在读序列结束前禁止。

31. 5. 6 间接数据传输模式

当使用间接读控制器时，必须遵循软件序列，包括 DMA 外设接口的设置（可参考间接写控制器和间接读控制器章节。）

31. 5. 7 AHB地址重映射

当访问经过直接访问控制器时，QSPI 的重映射功能定义了如何转换接收到的 AHB 地址。默认为 1：1 映射。当 QSPI_CR.AREN 使能，接收到的 AHB 地址重映射至 address + N，其中 N 的值保存在地址重映射寄存器（QSPI_RAR）中。建议在配置地址重映射寄存器（QSPI_RAR）前，先禁止 QSPI。

31. 5. 8 中断响应

QSPI 提供一个高有效的中断引脚。关于所有中断源的信息，请参考中断标志寄存器（QSPI_IFR）。

当软件读中断标志寄存器（QSPI_IFR）时，中断源清零。QSPI 控制器也可配置成在写中断标志寄存器（QSPI_IFR）时中断清零。默认的清零方式为：写清零（write-to-clear）。

中断屏蔽寄存器（QSPI_IMR）可屏蔽所有中断。写 0 使能，写 1 清零。

31. 5. 9 AHB保护寄存器

发生 POR 时，AHB 保护机制处于禁止状态。软件可利用保护寄存器来阻止 AHB 写入特定的寄存器。在配置保护寄存器前，建议禁止 QSPI。

31. 6 特殊功能寄存器

31. 6. 1 寄存器列表

QSPI 控制寄存器列表		
寄存器名称	偏移地址	寄存器描述
QSPI 基地址：4000_D400_H		
QSPI_CR	000_H	QSPI 配置寄存器
QSPI_DRIR	004_H	QSPI 器件读指令寄存器
QSPI_DWIR	008_H	QSPI 器件写指令寄存器
QSPI_DDLR	$00C_H$	QSPI 器件延时寄存器
QSPI_RDCR	010_H	QSPI 读数据捕捉寄存器
QSPI_DSCR	014_H	QSPI 器件容量配置寄存器
QSPI_SPR	018_H	QSPI SRAM 分块配置寄存器
QSPI_IATR	$01C_H$	QSPI 间接 AHB 地址触发寄存器
QSPI_DMACR	020_H	QSPI DMA 外设配置寄存器
QSPI_RAR	024_H	QSPI 地址重映射寄存器
QSPI_MBR	028_H	QSPI 模式位寄存器
QSPI_SFLR	$02C_H$	QSPI SRAM 数据深度状态寄存器
QSPI_TXHR	030_H	QSPI 发送阈值寄存器
QSPI_RXHR	034_H	QSPI 接收阈值寄存器
QSPI_WCR	038_H	QSPI 写完成控制寄存器
QSPI_PER	$03C_H$	QSPI 轮询结束寄存器
QSPI_IFR	040_H	QSPI 中断标志寄存器
QSPI_IMR	044_H	QSPI 中断屏蔽寄存器
Reserved	048_H	保留
Reserved	$04C_H$	保留
QSPI_WPLR	050_H	QSPI 写保护低位寄存器
QSPI_WPHR	054_H	QSPI 写保护高位寄存器
QSPI_WPCR	058_H	QSPI 写保护配置寄存器
Reserved	$05C_H$	保留
QSPI_IRTR	060_H	QSPI 间接读传输控制寄存器
QSPI_IRTWR	064_H	QSPI 间接读传输数据深度阈值寄存器
QSPI_IRTSAR	068_H	QSPI 间接读传输起始地址寄存器
QSPI_IRTNR	$06C_H$	QSPI 间接读传输字节数寄存器
QSPI_IWTR	070_H	QSPI 间接写传输控制寄存器
QSPI_IWTWR	074_H	QSPI 间接写传输数据深度阈值寄存器
QSPI_IWTSAR	078_H	QSPI 间接写传输起始地址寄存器
QSPI_IWTNR	$07C_H$	QSPI 间接写传输字节数寄存器
QSPI_ITARR	080_H	QSPI 间接触发地址范围寄存器
Reserved	0084_H ~$008C_H$	保留
QSPI_FCR	090_H	QSPI Flash 命令控制寄存器
QSPI_FCAR	094_H	QSPI Flash 命令地址寄存器

QSPI 控制寄存器列表		
寄存器名称	偏移地址	寄存器描述
QSPI 基地址：4000_D400$_H$		
Reserved	0098$_H$~009C$_H$	保留
QSPI_FCRLR	0A0$_H$	QSPI Flash 命令读数据低位寄存器
QSPI_FCRHR	0A4$_H$	QSPI Flash 命令读数据高位寄存器
QSPI_FCWLR	0A8$_H$	QSPI Flash 命令写数据低位寄存器
QSPI_FCWHR	0AC$_H$	QSPI Flash 命令写数据高位寄存器
QSPI_PFSR	00B0$_H$	QSPI 轮询 Flash 状态寄存器
Reserved	0B4$_H$~00F8$_H$	保留
QSPI_MIDR	00FC$_H$	QSPI 模块 ID 寄存器

31. 6. 2 寄存器描述

31. 6. 2. 1 QSPI配置寄存器（QSPI_CR）

QSPI 配置寄存器（QSPI_CR）

偏移地址：00_H

复位值：$10000000\_01111000\_00000000\_10000001_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDLES	Reserved						DTRM	ADEN	BAUD				XIPIM	XIPNX	AREN	DMAEN	SWPP	PSL				PSD	LIMEN	DACEN	Reserved				CPHA	CPOL	EN

IDLES	Bit 31	R	**串行接口和 QSPI PIPELINE 处于空闲** 该位是一个重新计时的信号，所以生产该状态位信号时会有一些延时。 在设置 IDLE 位有效和切换配置域之间，建议等待至少 4 个 ref_clk 周期。该延时可确保底层的同步，FIFO 为空，控制器可引入任何操作模式。
Reserved	Bit 30-25	R	**保留**
DTRM	Bit 24	R/W	**DTR 协议使能** 如果器件使用 DTR 协议，则置位该位。 注意，DTR 协议在 DTR 模式下提供所有命令。有些 DTR 读命令可在 STR 协议中处理（在 DTR 模式下）。将该位设置为 0 来执行这些命令。STR 协议中的 DTR 命令由 QSPI_DRIR.DDRM 来控制。
ADEN	Bit 23	R/W	**AHB 解码器使能（仅限直接访问模式）** 当设置为 1 时，PSL 位为无关位。 有效从机取决于实际的 AHB 地址。 每个器件的配置基于 QSPI_DSCR.CSxSIZE 来计算。
BAUD	Bit 22-19	R/W	**主模式波特率分频** SPI 波特率=（主参考时钟）/BD 其中 BD 可配置为以下： 0000 = /2 0001 = /4 0010 = /6 0011 = /8 0100 = /10 0101 = /12 0110 = /14 0111 = /16 1000 = /18 … 1111 = /32

			使能 QSPI 控制器前，优先设置好该寄存器。
XIPIM	Bit 18	R/W	**即刻进入 XIP 模式** 0：如果 XIP 使能，当下一个读指令完成后，控制器将退出 XIP 模式。 1：即刻进入 XIP 模式 当外部器件从 XIP 模式唤醒时，使用该寄存器。控制器假定下一个读指令将作为 XIP 指令传递给器件，因此不会要求传输读操作码。 注意：为退出 XIP 模式，该位需设置为 0。只有在下一个读指令执行后，才能在连接的器件中生效。因此，确定退出 XIP 模式前，在复位该位后，软件必须确保至少有一个读指令。 该位由硬件同步。控制器运行过程中，可对该位进行设置或清零。
XIPNX	Bit 17	R/W	**在下一个读指令进入 XIP 模式** 0：如果 XIP 使能，当下一个读指令完成后，控制器将退出 XIP 模式。 1：如果 XIP 禁止，该位置 1 可通知控制器，器件已就绪，可在下一个读指令进入 XIP 模式。接着，控制器将发送合适的命令序列，包括让器件进入 XIP 模式的模式位。 当控制器已确保接连的 Flash 器件已准备就绪可进入 XIP 模式后，使用该寄存器。 注意：为退出 XIP 模式，该位需设置为 0。只有在下一个读指令执行后，才能在连接的器件中生效。因此，确定退出 XIP 模式前，在复位该位后，软件必须确保至少有一个读指令。 该位由硬件同步。控制器运行过程中，可对该位进行设置或清零。
AREN	Bit 16	R/W	**AHB 地址重映射使能（仅限直接访问模式）** 当该位置 1，接收到的 AHB 地址将更新，并将（address+N）发送至 Flash 器件，其中 N 的值保存在地址重映射寄存器（QSPI_RAR）中。 该位由硬件同步。控制器运行过程中，可对该位进行设置或清零。
DMAEN	Bit 15	R/W	**DMA 外设接口使能** 0：禁止 1：使能 DMA 握手逻辑。使能后，QSPI 会通过 DMA 外设接口触发 DMA 传输请求。 该位由硬件同步。控制器运行过程中，可对该位进行设置或清零。
SWPP	Bit 14	R/W	**写保护引脚设置** 设置该位来驱动 Flash 器件的写保护引脚，与生成的存储器时钟重新同步。

			注意，WP 引脚仅在 SINGLE 或 DUAL 传输模式下有效。在 QUAD 传输模式下，WP 引脚用来传输数据，因此该寄存器位的设置将被无视。 该位由硬件同步。控制器运行过程中，可对该位进行设置或清零。
PSL	Bit 13-10	R/W	**外设片选线** 如果寄存器中的 PSD 位设置为 0，ss[3:0]为输出： ss[3:0] n_ss_out[3:0] xxx0 1110 xx01 1101 x011 1011 0111 0111 1111 1111（无外设选中） 其他情况下，ss[3:0]直接驱动 n_ss_out[3:0]
PSD	Bit 9	R/W	**外设选择解码** 0：4 个片选信号中只有 1 个有效 1：允许外部 4/16 译码
LIMEN	Bit 8	R/W	**传统 IP 模式使能** 0：使用直接访问控制器/间接访问控制器或 STIG 接口用作数据传输 1：使能传统模式。在传统模式下，任何 AHB 接口写操作都是串行发送至 Flash 器件。任何有效的 AHB 读会从内部 RX-FIFO 出栈，在 SPI 线上获取有外部 Flash 器件转发的数据，通过 HSIZE 输入控制和传输 4，2 或 1 个字节。 该位由硬件同步。控制器运行过程中，可对该位进行设置或清零。
DACEN	Bit 7	R/W	**直接访问控制器使能** 0：一旦当前数据字（FF_W）传输完成，禁止直接访问控制器。 1：使能直接访问控制器 当直接访问控制器和间接访问控制器均禁止，所有 AHB 请求完成时都会产生错误反应。 该位由硬件同步。控制器运行过程中，可对该位进行设置或清零。
Reserved	Bit 6-3	R	**保留**
CPHA	Bit 2	R/W	**时钟相位** 时钟相位映射到标准 SPI 时钟相位传输格式。 注意：当运行于 DDR 模式或者 DDR 协议时，保持该位为低电平。
CPOL	Bit 1	R/W	**SPI 字外的时钟极性** 映射到标准 SPI 时钟极性传输格式。 注意：当运行于 DDR 模式或者 DDR 协议时，保持该位为低电平。

EN	Bit 0	R/W	**QSPI 使能** 0：一旦当前数据字（FF_W）传输完成，禁止 QSPI。 1：使能 QSPI 当 spi_enable=0，所有的输出使能为无效，且所有引脚设置为输入模式。 该位由硬件同步。

31. 6. 2. 2 QSPI器件读指令寄存器（QSPI_DRIR）

QSPI 器件读指令寄存器（QSPI_DRIR）

偏移地址：004_H

复位值：$00000000\_00000000\_00000000\_00000011_B$

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

Reserved	Bit 31-29	R	**保留**
DCYC	Bit 28-24	R/W	**读指令所需的 Dummy 时钟周期数**
Reserved	Bit 23-21	R	**保留**
MBEN	Bit 20	R/W	**模式位使能** 置 1 确保模式位寄存器（QSPI_MBR）中定义的模式位跟在地址字节后发出。
Reserved	Bit 19-18	R	**保留**
DMODE	Bit 17-16	R/W	**标准 SPI 模式下的数据传输类型（如 IMODE 位所定义）** 0：SIO 模式 – 仅在 DQ0 上发送数据去器件，DQ1 上接收器件上的数据。 1：用作 DUAL 输入/输出指令。数据传输时，DQ0 和 DQ1 均用做为输入和输出。 2：用作 QUAD 输入/输出指令。数据传输时，DQ0，DQ1，DQ2 和 DQ3 均用做为输入和输出。
Reserved	Bit 15-14	R	**保留**
ADMODE	Bit 13-12	R/W	**标准 SPI 模式下的地址传输类型（如 IMODE 位所定义）** 0：地址仅可在 DQ0 上发送去器件。 1：地址仅可在 DQ0 和 DQ1 上发送去器件。 2：地址可在 DQ0，DQ1，DQ2 和 DQ3 上发送去器件。
Reserved	Bit 11	R	**保留**
DDRM	Bit 10	R/W	**DDR 位使能** 当 QSPI_WCR.OPCODE 符合 DDR 命令，该位置 1。
IMODE	Bit 9-8	R/W	**指令类型** 0：标准 SPI 模式（始终仅在 DQ0 上发送指令去器件） 1：DIO-SPI 模式（始终在 DQ0 和 DQ1 上发送指令，地址和数据） 2：QIO-SPI 模式（始终在 DQ0，DQ1，DQ2 和 DQ3 上发送指令，地址和数据）

			注意：该位不仅仅只与读传输相关。该位为全局设置，将影响 DAC 和 INDAC 读，写和任何 STIG 传输。
RINST	Bit 7-0	R/W	**在非 XIP 模式下，读操作码**

31. 6. 2. 3 QSPI器件写指令寄存器（QSPI_DWIR）

QSPI 器件写指令寄存器（QSPI_DWIR）

偏移地址：008_H

复位值：$00000000\_00000000\_00000000\_00000010_B$

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

Reserved	Bit 31-29	R	**保留**
DCYC	Bit 28-24	R/W	**写指令所需的 Dummy 时钟周期数**
Reserved	Bit 23-18	R	**保留**
DMODE	Bit 17-16	R/W	**标准 SPI 模式下的数据传输类型** 0：SIO 模式 – 仅在 DQ0 上发送数据去器件，DQ1 上接收器件上的数据。 1：用作 DUAL 输入/输出指令。数据传输时，DQ0 和 DQ1 均用做为输入和输出。 2：用作 QUAD 输入/输出指令。数据传输时，DQ0，DQ1，DQ2 和 DQ3 均用做为输入和输出。
Reserved	Bit 15-14	R	**保留**
ADMODE	Bit 13-12	R/W	**标准 SPI 模式下的地址传输类型** 0：地址仅可在 DQ0 上发送去器件。 1：地址仅可在 DQ0 和 DQ1 上发送去器件。 2：地址可在 DQ0，DQ1，DQ2 和 DQ3 上发送去器件。
Reserved	Bit 11-9	R	**保留**
WELD	Bit 8	R/W	**WEL 禁止** 在写操作前，该位允许关闭自动发送 WEL 命令
WINST	Bit 7-0	R/W	**写操作码**

31. 6. 2. 4 QSPI器件延时寄存器（QSPI_DDLR）

QSPI 器件延时寄存器（QSPI_DDLR）																															
偏移地址：$00C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSDA								CSDADS								CSEOT								CSSOT							

CSDA	Bit 31-24	R/W	**片选设为无效** 在两个传输之间，主模式片选输出设为无效的时间（主参考时钟 ref_clk 上的延时） 片选设为无效的最小延时： 1 sclk_out + 1 ref_clk 为确保片选不会在 1 个 sclk_out 周期内再次变为有效。 如果 CSDA=X，则片选无效时间为： 1 sclk_out + 1 ref_clk + X ref_clks
CSDADS	Bit 23-16	R/W	**不同从机的片选设为无效** 一个片选设为无效和另一个片选设为有效，其之间的延时（主参考时钟 ref_clk 上的延时）。该延时确保了两个不同从机切换之间的安静周期。当在两个外部 Flash 器件切换时，该位才有效。 举例：有 2 个 Flash 器件 A 和 B，作为不同的片选与控制器相连。用户有需求要在每个连续的非序列的传输之间，在特定时间段内，将片选设为无效。此时，用户可使用 CSDA 来进行控制。A 器件上传输的结束和 B 器件上传输的开始之间，将片选设为无效的时间要求可能不同。此时，用户可使用 CSDADS 来控制。 最小延时（CSDADS=0）： 1 sclk_out + 3 ref_clks 如果 CSDADS=X，则延时为： 1 sclk_out + 3 ref_clks + X ref_clks
CSEOT	Bit 15-8	R/W	**片选传输结束** 当前传输的最后一位与将片选（n_ss_out）设为无效之间的延时（主参考时钟 ref_clk 上的延时） 默认，当 CSEOT=0，在当前传输结束时，片选将在 sclk_out 的最后一个下降沿上设为无效。 如果 CSEOT=X，片选会在 sclk_out 的最后一个下降沿之后的 X 个 ref_clks 后变为无效。
CSSOT	Bit 7-0	R/W	**片选传输起始**

			设置 n_ss_out 为低电平和传输第一个比特位之间的延时。 默认，当 CSSOT=0，在 sclk_out 的第一个上升沿之前的半个 SCLK 周期，片选将被设为有效。 如果 CSSOT=X，在 sclk_out + X ref_clks 的第一个上升沿之前的半个 sclk_out 周期，片选将被设为有效。

31. 6. 2. 5 QSPI读数据捕捉寄存器（QSPI_RDCR）

QSPI 数据读捕捉寄存器（QSPI_RDCR）

偏移地址：010_H

复位值：$00000000\_00000000\_00000000\_00000001_B$

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

Reserved	Bit 31-20	R	**保留**
DLYT	Bit 19-16	R/W	**传输数据延时（延时时间为设定的 ref_clk 周期数）** （目的为在 DDR 模式中提高传输过程中的保持时间） 当 DDR 读命令执行时，该位才有效；否则，无视。 delay_value = 1 x ref_clk + <field_value>*ref_clk
Reserved	Bit 15-6	R	**保留**
SMES	Bit 5	R/W	**采样边沿选择（Flash 存储器数据输出）** 0：在 ref_clk 的下降沿对 Flash 存储器的数据输出进行采样 1：在 ref_clk 的上升沿沿对 Flash 存储器的数据输出进行采样
DLYR	Bit 4-1	R/W	**读数据捕捉延时（延时时间为设定的 ref_clk 周期数）**
Reserved	Bit 0	R/W	**保留，软件写 1。**

31. 6. 2. 6 QSPI器件容量配置寄存器（QSPI_DSCR）

QSPI 器件容量配置寄存器（QSPI_DSCR）

偏移地址：014_H

复位值：$00000000\_00010000\_00010000\_00000010_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CS3SIZE		CS2SIZE		CS1SIZE		CS0SIZE		BKSIZE					PASIZE												ADSIZE				

Reserved	Bit 31-29	R	保留
CS3SIZE	Bit 28-27	R/W	**连接 CS[3]引脚的 Flash 器件大小** 当 AHB 解码器使能时，该位有效。 00：512Mb 01：1Gb 10：2Gb 11：4Gb
CS2SIZE	Bit 26-25	R/W	**连接 CS[2]引脚的 Flash 器件大小** 当 AHB 解码器使能时，该位有效。 00：512Mb 01：1Gb 10：2Gb 11：4Gb
CS1SIZE	Bit 24-23	R/W	**连接 CS[1]引脚的 Flash 器件大小** 当 AHB 解码器使能时，该位有效。 00：512Mb 01：1Gb 10：2Gb 11：4Gb
CS0SIZE	Bit 22-21	R/W	**连接 CS[0]引脚的 Flash 器件大小** 当 AHB 解码器使能时，该位有效。 00：512Mb 01：1Gb 10：2Gb 11：4Gb
BKSIZE	Bit 20-16	R/W	**每块的字节数** 控制器需要该位来执行写保护逻辑。每块的字节数必须为 2 的 N 次方。 0：1 byte 1：2 bytes 2：4 bytes 3：8 bytes 16：65535 bytes

PASIZE	Bit 15-4	R/W	**每个器件页的字节数** 控制器需要该位来执行跨页的 Flash 写操作。
ADSIZE	Bit 3-0	R/W	**地址字节数** 0：1 byte …

31. 6. 2. 7 QSPI SRAM分块配置寄存器（QSPI_SPR）

QSPI SRAM 分块配置寄存器（QSPI_SPR）

偏移地址：018_H

复位值：$00000000\_00000000\_00000000\_00100000_B$

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

Reserved	Bit 31-8	R	**保留**
SRAMPS	Bit 7-0	R/W	定义了 SRAM 中间接读区域的大小。默认地，SRAM 中一半的大小预留给间接读操作，另一半分配给间接写操作。该寄存器大小可随着 SRAM 深度改变。 分配给间接读的地址数量 = SRAM_PARTITION_REG+1 分配给间接写的地址数量 = (2**8)-SRAM_PARTITION_REG 不要将 SRAM_PARTITION_REG 设置为 0x0000 或(2**8)-1

31. 6. 2. 8 QSPI间接AHB地址触发寄存器（QSPI_IATR）

QSPI 间接 AHB 地址触发寄存器（QSPI_IATR）

偏移地址：$01C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

INDTAD	Bit 31-0	R/W	**间接触发地址** 该地址为基地址。当接收到的 AHB 读访问地址与（从触发地址到触发地址+<配置范围>）内的地址匹配，则通过从间接控制器 SRAM 中获取数据来

			完成 AHB 请求。

31. 6. 2. 9 QSPI DMA外设配置寄存器（QSPI_DMACR）

QSPI DMA 外设配置寄存器（QSPI_DMACR）

偏移地址：020_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-12	R	**保留**
BNUMB	Bit 11-8	R/W	**DMA 外设请求中 Burst 类型的字节数** 设置为 0 时，代表一个字节。该位需在间接读/写开始前设置完成。实际的字节数为 2**（寄存器设置值） 突发传输的条件为： 1）SRAM 中空闲的存储空间（字为单位）大于等于 0x20->[11:8]（以字节为单位）的设置值； 2）间接访问总的数据量（间接读传输控制寄存器（QSPI_IRTR）/间接写传输控制寄存器（QSPI_IWTR））大于等于 0x20->[11:8]的设置值； 如果不满足上述条件，系统会自动采用单次传输。 例如：（SRAM 大小为 1KB）SRAM 分块配置寄存器（QSPI_SPR）配置为 0x80，第一次启动数据间接写传输时，SRAM 中空闲的存储空间为 0x80 字大小，如果此时设置 0x20->[11:8]的值大于等于 8 时，控制器会自动采用 single 方式传输。
Reserved	Bit 7-4	R	**保留**
SNUMB	Bit 3-0	R/W	**DMA 外设请求中 Single 类型的字节数** 设置为 0 时，代表一个字节。该位需在间接读/写开始前设置完成。实际的字节数为 2**（寄存器设置值），2**（寄存器设置值）表示一次 single 传输数据量的大小，如一次 single 传输一个字，则需要将此控制位设置为 2，如果设置成 0 则一次 single 传输传输的数据量为 1 个字节。 注意应用时大小需要设置成 Word 的整数倍。

31. 6. 2. 10 QSPI地址重映射寄存器（QSPI_RAR）

QSPI 地址重映射寄存器（QSPI_RAR）																															
偏移地址：024_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
READDR																															

READDR	Bit 31-0	R/W	该寄存器将接收到的 AHB 地址重映射到一个不同的地址。

31. 6. 2. 11 QSPI模式位寄存器（QSPI_MBR）

QSPI 模式位寄存器（QSPI_MBR）																															
偏移地址：028_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								MODEB							

Reserved	Bit 31-8	R	**保留**
MODEB	Bit 7-0	R/W	**模式位** 在地址字节后，向外部器件发送这 8 位模式位

31. 6. 2. 12 QSPI SRAM数据深度状态寄存器（QSPI_SFLR）

QSPI SRAM 数据深度状态寄存器（QSPI_SFLR）																															
偏移地址：$02C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INDWSFL																INDRSFL															

INDWSFL	Bit 31-16	R	SRAM 深度状态（间接写区域），单位为字（4 个字节）
INDRSFL	Bit 15-0	R	SRAM 深度状态（间接读区域），单位为字（4 个字节）

31. 6. 2. 13 QSPI发送阈值寄存器（QSPI_TXHR）

QSPI 发送阈值寄存器（QSPI_TXHR）

偏移地址：030_H

复位值：$00000000\_00000000\_00000000\_00000001_B$

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

Reserved	Bit 31-5	R	保留
TXTH	Bit 4-0	R/W	小容量 TX FIFO 不满中断阈值设置值 只有在传统模式下访问 Flash 器件，该位才有效。 其他情况下可无视。

31. 6. 2. 14 QSPI接收阈值寄存器（QSPI_RXHR）

QSPI 接收阈值寄存器（QSPI_RXHR）

偏移地址：034_H

复位值：$00000000\_00000000\_00000000\_00000001_B$

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

Reserved	Bit 31-5	R	保留
RXTH	Bit 4-0	R/W	小容量 RX FIFO 不空中断阈值设置值 只有在传统模式下访问 Flash 器件，该位才有效。 其他情况下可无视。

31. 6. 2. 15 QSPI写完成控制寄存器（QSPI_WCR）

QSPI 写完成控制寄存器（QSPI_WCR）

偏移地址：038_H

复位值：$00000000\_00000001\_00000000\_00000101_B$

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

PREPD	Bit 31-24	R/W	**轮询重复延时** 在自动轮询阶段，片选保持无效的额外延时时间
PCNT	Bit 23-16	R/W	**轮询次数** 定义轮询次数。当控制器吞吐量高时，有必要添加额外的周期。
Reserved	Bit 15	R	**保留**
PDIS	Bit 14	R/W	**轮询禁止** 关闭自动轮询功能
PPLT	Bit 13	R/W	**轮询极性** 0：如果轮询位为 0，则写传输完成。 1：如果轮询位为 1，则写传输完成。
Reserved	Bit 12-11	R	**保留**
PBIND	Bit 10-8	R/W	**轮询位检索** 设置为 010 时，表示轮询 QSPI_PFSR 寄存器的 bit2。 设置为 111 时，表示轮询 QSPI_PFSR 寄存器的 bit7。
OPCODE	Bit 7-0	R/W	**操作码** 当自动轮询器件编程的完成状态时，控制器必须发送操作码。该命令在所有的器件写操作后发送。默认使用操作码 0x05 来轮询标准器件状态寄存器。

31. 6. 2. 16 QSPI轮询结束寄存器（QSPI_PER）

QSPI 轮询结束寄存器（QSPI_PER）																															
偏移地址：$03C_H$																															
复位值：$11111111\_11111111\_11111111\_11111111_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCYCN																															

PCYCN	Bit 31-0	R/W	**轮询周期数** 轮询周期数后，产生轮询结束中断。

31. 6. 2. 17 QSPI中断标志寄存器（QSPI_IFR）

QSPI 中断标志寄存器（QSPI_IFR）

偏移地址：040_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		POLLF	INDRSFF	SRFFF	SRFNEF	STFFF	STFNFF	ROVF	INDTWF	AHBAEF	WPAF	INDRRF	INDCF	UDFF	MODFF

Reserved	Bit 31-14	R	保留
POLLF	Bit 13	R/C_W1	**最大的轮询周期数**
INDRSFF	Bit 12	R/C_W1	**SRAM 中的间接读区域满，无法立即完成间接操作。**
SRFFF	Bit 11	R/C_W1	**小容量 RX FIFO 满**（当前 FIFO 状态） 在非 SPI 传统模式下，可无视。 0：FIFO 非满 1：FIFO 满
SRFNEF	Bit 10	R/C_W1	**小容量 RX FIFO 非空**（当前 FIFO 状态） 在非 SPI 传统模式下，可无视。 0：FIFO < RX 阈值 1：FIFO ≥ 阈值
STFFF	Bit 9	R/C_W1	**小容量 TX FIFO 满**（当前 FIFO 状态） 在非 SPI 传统模式下，可无视。 0：FIFO 非满 1：FIFO 满
STFNFF	Bit 8	R/C_W1	**小容量 TX FIFO 非满**（当前 FIFO 状态） 在非 SPI 传统模式下，可无视。 0：FIFO > 阈值 1：FIFO ≤ 阈值
ROVF	Bit 7	R/C_W1	**接收上溢（仅在传统模式下发生）** 0：未检测到上溢 1：发生上溢
INDTWF	Bit 6	R/C_W1	**超出间接传输深度阈值** 0：未超出阈值 1：超出阈值
AHBAEF	Bit 5	R/C_W1	**检测到非法 AHB 访问** AHB 写 Wrapping Burst 和使用 SPLIT/RETRY 访问将触发该中断。 如果 DAC 禁止状态下进行 AHB 访问，也将触发该中断。
WPAF	Bit 4	R/C_W1	**企图写保护区域被拒绝**
INDRRF	Bit 3	R/C_W1	**不接收间接操作请求**

			已存在两个间接操作。
INDCF	Bit 2	R/C_W1	**控制器已完成上一个间接操作**
UDFF	Bit 1	R/C_W1	**下溢检测** 0：未检测到下溢 1：检测到下溢：当小容量 TX FIFO 为空时，企图发送数据。当 AHB 写数据太慢，跟不上写请求时，有可能发生下溢。
MODFF	Bit 0	R/C_W1	**模式失效** 不再使用传统模式。

31. 6. 2. 18 QSPI中断屏蔽寄存器（QSPI_IMR）

QSPI 中断屏蔽寄存器（QSPI_IMR）

偏移地址：044_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-14	R	**保留**
INTEN	Bit 13-0	R/W	0：中断标志寄存器（QSPI_IFR）中的相关中断位禁止 1：中断标志寄存器（QSPI_IFR）中的相关中断位使能

31. 6. 2. 19 QSPI写保护低位寄存器（QSPI_WPLR）

QSPI 写保护低位寄存器（QSPI_WPLR）

偏移地址：050_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

LBLKNUM	Bit 31-0	R/W	定义了写保护块范围的起始块，通过器件容量配置寄存器（QSPI_DSCR）可配置块的字节数。仅当写保护功能（QSPI_WPCR.WPEN）为低电平，可更改该位。

31. 6. 2. 20 QSPI写保护高位寄存器（QSPI_WPHR）

QSPI 写保护高位寄存器（QSPI_WPHR）																															
偏移地址：054_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HBLKNUM																															

HBLKNUM	Bit 31-0	R/W	定义了写保护块范围的结束块，通过器件容量配置寄存器（QSPI_DSCR）可配置块的字节数。 仅当写保护功能（QSPI_WPCR.WPEN）为低电平，可更改该位。

31. 6. 2. 21 QSPI写保护配置寄存器（QSPI_WPCR）

QSPI 写保护配置寄存器（QSPI_WPCR）																															
偏移地址：058_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														WPEN	WPINV

Reserved	Bit 31-2	R	**保留**
WPEN	Bit 1	R/W	**写保护使能** 1：拒绝访问任何 AHB 写访问（其写访问地址在写保护低位寄存器（QSPI_WPLR）和写保护高位寄存器（QSPI_WPHR）范围内）。产生一个 AHB 错误响应，触发中断源。 0：写保护禁止 该位由硬件内部同步。当改变该位状态时，无特别软件要求。
WPINV	Bit 0	R/W	**写保护取反控制** 1：写保护取反：系统允许写由写保护低位寄存器（QSPI_WPLR）和写保护高位寄存器（QSPI_WPHR）定义的保护区域的地址。 0：写保护不取反：系统不允许写由写保护低位寄存器（QSPI_WPLR）和写保护高位寄存器（QSPI_WPHR）定义的保护区域的地址。 仅当写保护功能（QSPI_WPCR.WPEN）为低电平，可更改该位。

31. 6. 2. 22 QSPI间接读传输控制寄存器（QSPI_IRTR）

QSPI 间接读传输控制寄存器（QSPI_IRTR）

偏移地址：060_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-8	R	**保留**
INDRNUM	Bit 7-6	R	**间接操作的完成数** 该位与 RDCS 位联合使用。 当完成一个间接操作时，由硬件递增。 当 RDCS 位写入 1 时，则递减。
RDCS	Bit 5	R/C_W1	**间接读完成状态** 当完成一个间接操作，该位由硬件置 1，写 1 清零。
RDQS	Bit 4	R	**已有两个间接读操作排队等候** 该位由硬件置 1 和清零。
SRAMFS	Bit 3	R/C_W1	**SRAM 满，无法立即完成间接操作** 硬件置 1，写 1 清零。
RDPS	Bit 2	R	**间接读操作进行中** 仅由硬件清零。
RDDIS	Bit 1	W1	**取消间接读** 写 1 取消进行中的间接读操作。同时取消当前所有间接操作。 该位由硬件内部同步。
RDST	Bit 0	W1	**开始间接读** 写 1 触发间接读操作。触发间接读操作前，先设置好间接读传输起始地址寄存器（QSPI_IRTSAR）和间接读传输字节数寄存器（QSPI_IRTNR）。 该位由硬件内部同步。

31. 6. 2. 23 QSPI间接读传输数据深度阈值寄存器（QSPI_IRTWR）

QSIP 间接读传输数据深度阈值寄存器（QSPI_IRTWR）																															
偏移地址：064_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE																															

VALUE	Bit 31-0	R/W	**数据深度** 在 DMA 外设访问前，该值代表的是 SRAM 的最小数据深度。当 SRAM 数据深度超越该值，则触发中断源。写全 0 可禁止。 单位为字节。

31. 6. 2. 24 QSPI间接读传输起始地址寄存器（QSPI_IRTSAR）

QSPI 间接读传输起始地址寄存器（QSPI_IRTSAR）																															
偏移地址：068_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															

ADDR	Bit 31-0	R/W	**间接访问起始地址**

31. 6. 2. 25 QSPI间接读传输字节数寄存器（QSPI_IRTNR）

QSPII 间接读传输字节数寄存器（QSPI_IRTNR）																															
偏移地址：$06C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUM																															

NUM	Bit 31-0	R/W	**间接字节数** 该设置值可大于 SRAM 大小。

31. 6. 2. 26 QSPI间接写传输控制寄存器（QSPI_IWTR）

QSP 间接写传输控制寄存器（QSPI_IWTR）																													
偏移地址：070_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								INDWNUM		WRCS	WRQS	Reserved	WRPS	WRDIS	WRST

Reserved	Bit 31-8	R	**保留**
INDWNUM	Bit 7-6	R	**间接操作的完成数** 该位与 WRCS 位联合使用。 当完成一个间接操作时，由硬件递增。 当 WRCS 位写入 1 时，则递减。
WRCS	Bit 5	R/C_W1	**间接写完成状态** 当完成一个间接操作，该位由硬件置 1，写 1 清零。
WRQS	Bit 4	R	**已有两个间接写操作排队等候** 该位由硬件置 1 和清零。
Reserved	Bit 3	R/C_W1	**保留**
WRPS	Bit 2	R	**间接写操作进行中** 仅由硬件清零。
WRDIS	Bit 1	W1	**取消间接写** 写 1 取消进行中的间接写操作。同时取消当前所有间接操作。 该位由硬件内部同步。
WRST	Bit 0	W1	**开始间接写** 写 1 触发间接写操作。触发间接写操作前，先设置好间接写传输起始地址寄存器（QSPI_IWTSAR）和间接写传输字节数寄存器（QSPI_IWTNR）。 该位由硬件内部同步。

31. 6. 2. 27 QSPI间接写传输数据深度阈值寄存器（QSPI_IWTWR）

QSPI 间接写传输数据深度阈值寄存器（QSPI_IWTWR）																															
偏移地址：074_H																															
复位值：$11111111\_11111111\_11111111\_11111111_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE																															

VALUE	Bit 31-0	R/W	**数据深度** 在 DMA 外设访问前，该值代表的是 SRAM 的最大数据深度。当 SRAM 数据深度低于该值，则触发中断源。写全 1 可禁止。 单位为字节。

31. 6. 2. 28 QSPI间接写传输起始地址寄存器（QSPI_IWTSAR）

QSPI 间接写传输起始地址寄存器（QSPI_IWTSAR）																															
偏移地址：078_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															

ADDR	Bit 31-0	R/W	**间接访问起始地址**

31. 6. 2. 29 QSPI间接写传输字节数寄存器（QSPI_IWTNR）

QSPI 间接写传输字节数寄存器（QSPI_IWTNR）																															
偏移地址：$07C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUM																															

NUM	Bit 31-0	R/W	**间接字节数** 该设置值可大于 SRAM 大小。

31. 6. 2. 30 QSPI间接触发地址范围寄存器（QSPI_ITARR）

QSPI 间接触发地址范围寄存器（QSPI_ITARR）																															
偏移地址：080_H																															
复位值：$00000000\_00000000\_00000000\_00000100_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												RNGW			

Reserved	Bit 31-4	R	保留
RNGW	Bit 3-0	R/W	**间接范围宽度** 该位为间接 AHB 地址触发寄存器（QSPI_IATR）的偏移地址。当触发间接访问且 AHB 地址在此范围内，访问请求就会发送至间接写/读控制器。 该值为 2 的 N 次方。默认 2^4=16，允许进行 16 字节的 Burst。该位段反映了范围宽度，有效地址的个数（单个地址 32 位）从间接触发地址到间接触发地址 + <间接范围宽度-1>（间接触发地址-> 间接触发地址 + 15

31. 6. 2. 31 QSPI Flash命令控制寄存器（QSPI_FCR）

QSPI Flash 命令控制寄存器（QSPI_FCR）

偏移地址：090_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE								RDEN	RDNUM			ADDREN	MODBEN	ADNUM		WREN	WDNUM			DUMNUM					Reserved					CMDS	CMDT

OPCODE	Bit 31-24	R/W	**命令操作码** 命令操作码需在触发命令前设置。举例：0x20 映射到 SubSector 擦除，写 CMDT 位可发送命令。 注意，利用该方式发送命令，将用到 QSPI_DRIR.IMODE 表示的指令类型。如果该位设置为 2'b00，则会连续传输命令操作码，命令地址，命令 dummy 字节，命令数据。如果该位设置为 2'b01，则会通过 DQ0 和 DQ1 引脚并行传输命令操作码，命令地址，命令 dummy 字节和命令数据。如果该位设置为 2'b10，则会通过 DQ0，DQ1，DQ2，DQ3 引脚并行传输命令操作码，命令地址，命令 dummy 字节和命令数据。
RDEN	Bit 23	R/W	**读数据使能** 如果 OPCODE 位中的命令要求接收读数据字节，将该位设置为 1。
RDNUM	Bit 22-20	R/W	**读数据字节数** 高达 8 个数据字节可读取 0：1 字节 1：2 字节 … 7：8 字节
ADDREN	Bit 19	R/W	**命令地址使能** 如果 OPCODE 位中的命令需要地址，将该位设置为 1。 通过 CMDT 位触发命令前，先设置好该位。
MODBEN	Bit 18	R/W	**模式位使能** 该位置 1 确保模式位寄存器（QSPI_MBR）中的模式位在地址字节后发送。
ADNUM	Bit 17-16	R/W	**地址字节数** 设置所需的地址字节数（地址本身在 Flash 命令地址寄存器（QSPI_FCAR）中设置） 通过 CMDT 位触发命令前，先设置好该位 00：1 地址字节

			01：2 地址字节 10：3 地址字节 11：4 地址字节
WREN	Bit 15	R/W	**写数据使能** 如果 OPCODE 位中的命令要求发送写数据字节，将该位设置为 1。
WDNUM	Bit 14-12	R/W	**写数据字节数** 高达 8 个数据字节可写 0：1 字节 1：2 字节 … 7：8 字节
DUMNUM	Bit 11-7	R/W	**Dummy 字节数** 设置所需的 dummy 字节数。 通过 CMDT 位触发命令前，先设置好该位。 注意，该位得设置不能大于 5'h03。设置过多的 dummy 字节有可能触发 TX FIFO 满或忽视掉读命令的一些数据，无法预知内核行为。 不要混淆 dummy 周期和 dummy 字节，仔细察看 Flash 器件规格中 dummy 周期的定义。
Reserved	Bit 6-2	R	**保留**
CMDS	Bit 1	R	**命令执行进行中**
CMDT	Bit 0	W1	**执行命令** 该位内部同步。

31. 6. 2. 32 QSPI Flash命令地址寄存器（QSPI_FCAR）

QSPI Flash 命令地址寄存器（QSPI_FCAR）

偏移地址：094_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

CMDADR	Bit 31-0	R/W	**命令地址** 在触发 QSPI_FCR.CMDT 前，先设置好该位。该地址为 QSPI_FCR.OPCODE 命令的地址。

31. 6. 2. 33 QSPI Flash命令读数据低位寄存器（QSPI_FCRLR）

QSPI Flash 命令读数据低位寄存器（QSPI_FCRLR）																															
偏移地址：$0A0_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDDL																															

CMDDL	Bit 31-0	R	**命令读数据（低字节）** 通过触发 Flash 命令控制寄存器（QSPI_FCR）中的事件，由 Flash 器件返回的任何状态或配置读操作的数据。 当 QSPI_FCR.CMDS 为低电平，该寄存器有效。

31. 6. 2. 34 QSPI Flash命令读数据高位寄存器（QSPI_FCRHR）

QSPI Flash 命令读数据高位寄存器（QSPI_FCRHR）																															
偏移地址：$0A4_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDDH																															

CMDDH	Bit 31-0	R	**命令读数据（高字节）** 通过触发 Flash 命令控制寄存器（QSPI_FCR）中的事件，由 Flash 器件返回的任何状态或配置读操作的数据。 当 QSPI_FCR.CMDS 为低电平，该寄存器有效。

31. 6. 2. 35 QSPI Flash命令写数据低位寄存器（QSPI_FCWLR）

QSPI Flash 命令写数据低位寄存器（QSPI_FCWLR）																															
偏移地址：$0A8_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDDL																															

CMDDL	Bit 31-0	R	**命令写数据（低字节）** 在触发 QSPI_FCR.CMDT 前，先设置好该位。通过触发 Flash 命令控制寄存器（QSPI_FCR）中的事件，向 Flash 器件写入的任何状态或配置写操作的数据。

31. 6. 2. 36 QSPI Flash命令写数据高位寄存器（QSPI_FCWHR）

QSPI Flash 命令写数据高位寄存器（QSPI_FCWHR）																															
偏移地址：$0AC_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDDH																															

CMDDH	Bit 31-0	R	**命令写数据（高字节）** 在触发 QSPI_FCR.CMDT 前，先设置好该位。通过触发 Flash 命令控制寄存器（QSPI_FCR）中的事件，由 Flash 器件写入的任何状态或配置写操作的数据。

31. 6. 2. 37 QSPI轮询Flash状态寄存器（QSPI_PFSR）

QSPI 轮询 Flash 状态寄存器（QSPI_PFSR）

偏移地址：$0B0_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-9	R	**保留**
PSV	Bit 8	R	**轮询状态有效** 当 FLSS 位有效，该位置 1。
FLSS	Bit 7-0	R	**Flash 状态** 定义了器件的实际状态寄存器。

31. 6. 2. 38 QSPI模块ID寄存器（QSPI_MIDR）

QSPI 模块 ID 寄存器（QSPI_MIDR）

偏移地址：$0FC_H$

复位值：$00000000\_00000001\_00000100\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID1								ID0																							

ID1	Bit 31-24	R	**固定号码** 与版本号（由该寄存器的 3 LSB 定义）相关
ID0	Bit 23-0	R	**模块/版本 ID 号**

第32章 模数转换器（ADC）

32. 1 概述

该 ADC 为逐次逼近型模数转换器，采样精度为 12 位，最多可以测量 16 个外部信号、两个内部参考电压和一个 1/2 VDD 电压。通道的转换可选择单次、连续、扫描或不连续等采样模式，其采样结果存储在 16 位数据寄存器，数据存储格式可以选择左对齐或右对齐存储。

ADC 模块具有模拟看门狗特性，允许应用程序检测输入电压是否在用户设定的阈值上限和下限范围内。

32. 2 特性

- 可配置的转换分辨率（6/8/10/12 位）
- 支持单次或连续工作模式
- 在标准转换、插入转换结束后以及发生模拟看门狗或溢出事件时产生中断
- 用于自动将通道“0”转换为通道“n”的扫描模式
- 可配置的数据对齐方式
- 可独立设置各通道采样时间
- 可配置外部触发器选项，可为标准转换和插入转换配置极性
- 支持不连续采样模式
- 可配置的参考源选择
- 可配置的转换时钟分频
- 支持标准数据转换的 DMA 请求

32. 3 结构框图

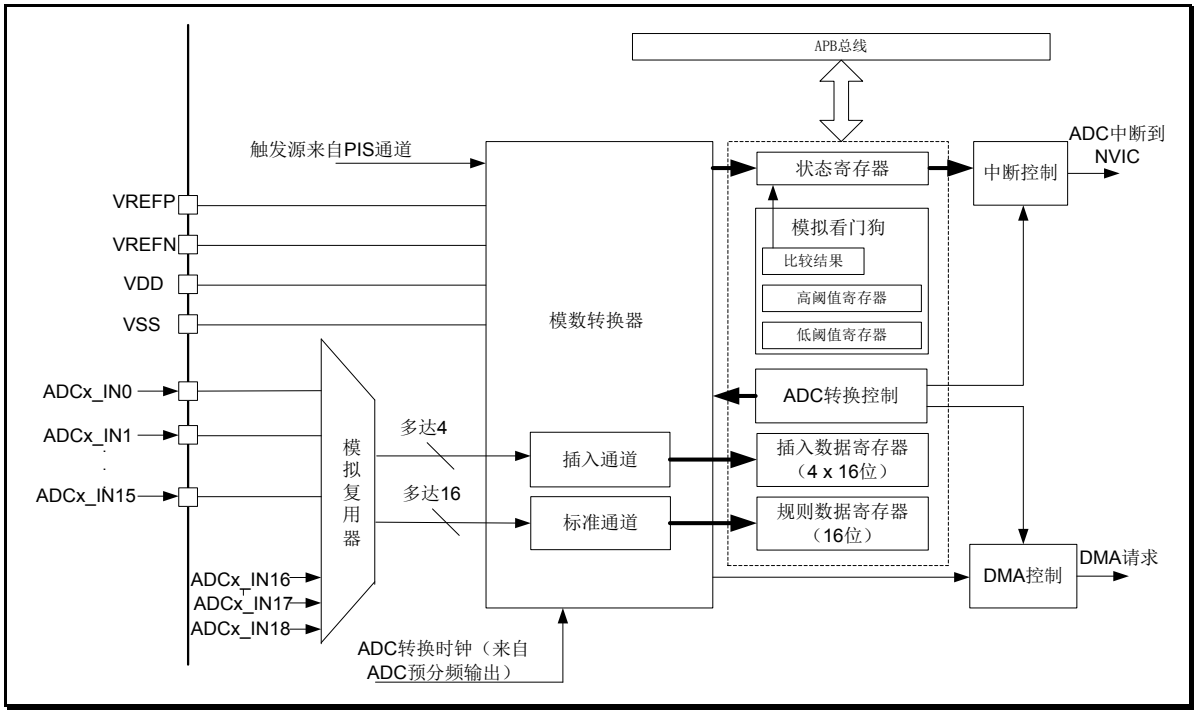


图 32-1 ADC 结构框图

32. 4 功能描述

32. 4. 1 ADC控制

通过将 ADC_CON1 寄存器中的 ADCEN 位置 1 来使能 ADC。

通过将 ADC_CON1 寄存器中的 NCHTRG 或 ICHTRG 位置 1 来启动 AD 转换。

通过将 ADC_CON1 寄存器中的 ADCEN 位清零来关闭 ADC。

32. 4. 2 ADC时钟

ADC 内部工作需要两路时钟：

◇ 用于模拟电路的时钟：ADCCLK

此时钟来自于经可编程预分频器分频的 APB 时钟，该预分频器可将 APB 时钟进行 1~128 分频，供 ADC 模拟电路使用。

◇ 用于数字接口的时钟（用于寄存器读/写访问）

此时钟为 APB 时钟。

32. 4. 3 通道控制

外部的 16 条复用通道可分为两组：标准转换和插入转换，每个组包含一个转换序列，该序列可按任意顺序在任意通道上完成。例如，可按以下顺序对序列进行转换：ADC_IN3、ADC_IN8、ADC_IN2、ADC_IN2、ADC_IN0、ADC_IN2、ADC_IN2、ADC_IN15。

◇ 一个标准转换组最多由 16 个转换构成。必须在 ADC_NCHSx 寄存器中选择转换序列的标准通道及其顺序。标准转换组中的转换总数必须写入 ADC_CHSL 寄存器中的 NSL 位。

◇ 一个插入转换组最多由 4 个转换构成。必须在 ADC_ICHS 寄存器中选择转换序列的插入通道及其顺序。插入转换组中的转换总数必须写入 ADC_CHSL 寄存器中的 ISL 位。

如果在转换期间修改 ADC_NCHSx 或 ADC_ICHS 寄存器，将复位当前转换并向 ADC 发送一个新的启动脉冲，以转换新选择的组。

内部通道 ADC_IN16~ADC_IN18 用于和芯片内部其它模块的输出相连，ES32F336x 系列 ADC_IN16~ADC_IN18 分别对应 OP0_ALTOUT0、OP1_ALTOUT1、OP2_MAINOUT，其余两个系列（ES32F36xx 和 ES32F39xx）ADC_IN16~ADC_IN18 保留不使用。

32. 4. 4 单次工作模式

单次工作模式是指 ADC 执行一次转换，其工作流程如下：

◇ 将 CON1.CM 位清 0，然后通过后续方式来启动此模式：

◇ 将 ADC_CON1 寄存器中的 NCHTRG 位置 1（仅适用于标准通道）

◇ 将 ADC_CON1 寄存器中的 ICHTRG 位置 1（仅适用于插入通道）

◇ 外部触发（适用于标准通道或插入通道）

完成所选通道的转换之后：

◇ 如果转换了标准通道组：

- 转换数据存储在 16 位 ADC_NCHDR 寄存器中
- 标准转换结束标志 ADC_STAT.NCHE 置 1
- 若标准转换完成中断使能位 ADC_CON0.NCHEIE 置 1，将产生中断

◇ 如果转换了插入通道组：

- 转换数据存储在 16 位 ADC_ICHDRx 寄存器中
- 插入转换结束标志 ADC_STAT.ICHE 置 1
- 若插入转换完成中断使能位 ADC_CON0.ICHEIE 置 1，将产生中断

然后，ADC 停止。

32. 4. 5 连续工作模式

连续工作模式是指 ADC 执行一个转换任务后，立即执行下一个转换任务，其工作流程如下：

◇ 将 CON1.CM 位置 1

◇ 通过外部触发或 ADC_CON1.NCHTRG 置 1，来启动此模式（仅适用于标准通道）

每次转换之后：

◇ 如果转换了标准通道组：

- 上次转换的数据存储在 16 位 ADC_NCHDR 寄存器中
- 标准转换结束标志 ADC_STAT.NCHE 置 1
- 若标准转换完成中断使能位 ADC_CON0.NCHEIE 置 1，将产生中断

注 1：连续转换模式不适用于插入组通道。

注 2：连续模式下唯一的例外情况是，在使能 ADC_CON0.IAUTO 时，插入通道组在标准通道之后自动转换。

32. 4. 6 时序图

在使能 ADC 后，需要一段稳定时间 t_{STAB}，然后再启动 ADC 转换，并经过 15 个时钟周期后 CHE 标志置 1，如下图所示。

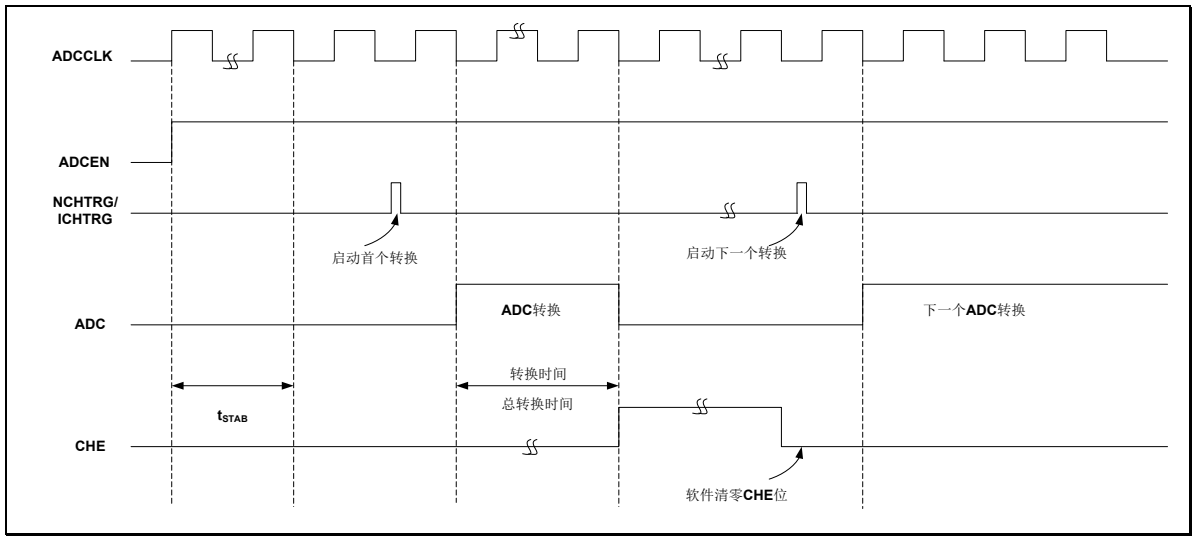


图 32-2 ADC 转换时序图

32. 4. 7 模拟看门狗

如果 ADC 转换的模拟电压低于阈值下限（寄存器 ADC_WDTL）或高于阈值上限（寄存器 ADC_WDTH），则模拟看门狗状态位 ADC_STAT.AWDF 位置 1。使能 ADC_CON0.AWDIE 位可以打开模拟看门狗中断。

上下限阈值寄存器是低 12 位有效的寄存器，而 ADC 结果寄存器可以左对齐和右对齐，但看门狗是在对齐之前将模拟电压与阈值上限和下限进行比较。

下表介绍了应如何配置 ADC_CON0 寄存器才能在一个或多个通道上使能模拟看门狗。

模拟看门狗保护通道	AWDSGL	NCHWDEN	ICHWDTEN
无	X	0	0
所有插入通道	0	0	1
所有标准通道	0	1	0
所有标准通道和插入通道	0	1	1
单个插入通道(AWDCH 选择)	1	0	1
单个标准通道(AWDCH 选择)	1	1	0
单个标准/插入通道(AWDCH 选择)	1	1	1

表 32-1 模拟看门狗通道选择

使用模拟看门狗流程

1. 设置 ADC_CON0.AWDSGL 位来选择看门狗作用于单个通道或一组通道，清 0 选择一组通道，置 1 选择单个通道。选择单个通道通过 AWDCH 选择哪个通道。
2. 置位 ADC_CON0.NCHWDEN 或 ADC_CON0.ICHWDTEN 选择使能标准组或插入组通道。

32. 4. 8 通道扫描

扫描模式是指连续依次转换标准组所有通道或者插入组所有通道。通过将 ADC_CON0.SCANEN 位置 1 来选择扫描模式。ADC 会扫描在 ADC_NCHSx 寄存器（对于标准通道）或 ADC_ICHS 寄存器（对于插入通道）中选择的所有通道，为组中的每个通道依次执行一次转换。如果将 CM 位置 1，并且是标准组扫描，ADC 会循环转换标准组所有通道。

如果将 ADC_CON1.DMA 位置 1，则在每次通道转换结束标志置位后，通过 DMA 控制器将转换自标准通道组的数据寄存器（ADC_NCHDR）中的数据传输到 SRAM。

在以下情况下，ADC_STAT 寄存器中的 NCHE 位置 1：

- ◇ 如果 CON1.NCHESEL 位清零，在每个标准组序列转换结束时
- ◇ 如果 CON1.NCHESEL 位置 1，在每个标准通道转换结束时

从插入通道转换的数据始终存储在 ADC_ICHDRx 寄存器中。

32. 4. 9 插入通道控制

触发插入

要使用触发插入，必须将 ADC_CON0.IAUTO 位清零。使用触发插入时，必须确保触发事件之间的间隔长于插入序列。

在插入转换期间出现标准通道转换事件，插入转换不会被中断，标准转换在插入转换结束时执行。

在标准组转换期间触发插入：

1. 通过外部触发或将 ADC_CON1.NCHTRG 位置 1 来启动标准通道组转换。
2. 如果在标准通道组转换期间出现外部插入触发或者 ADC_CON1.ICHTRG 位置 1，则当前的转换会打断，并且插入通道序列会切换为单次扫描模式。
3. 然后，标准通道组的标准转换会从上次中断的标准转换处恢复。

在插入转换期间触发插入

自动插入

如果将 IAUTO 位置 1，则插入组中的通道会在标准组通道之后自动转换。这可用于转换最多由 20 个 AD 转换构成的序列，这些转换在 ADC_NCHSx 和 ADC_ICHS 寄存器中编程。

在此模式下，必须禁止插入通道上的外部触发。

如果 CM 位和 IAUTO 位均已置 1，则在转换标准通道之后会继续转换插入通道。

32. 4. 10 不连续采样控制

标准组

可将 ADC_CON0.NCHDCEN 位置 1 来使能此模式。该模式可用于转换含有 n (n≤8)个转换的短序列，该短序列是在 ADC_NCHSx 寄存器中选择的转换序列的一部分。可通过写入 ADC_CON0.ETRGN 位来指定 n 的值。

出现外部触发时，将启动在 ADC_NCHSx 寄存器中选择的 n 个转换，直到序列中的所有转换均完成为止。通过 ADC_CHSL.NSL 位定义总序列长度。

示例：

n = 4，ADC_CHSL.NSL=9，要转换的通道 = 0、2、3、5、6、7、8、10、11

第 1 次触发：转换序列 0、2、3、5

第 2 次触发：转换序列 6、7、8、10

第 3 次触发：转换序列 11 并生成 NCHE 事件

第 4 次触发：转换序列 0、2、3、5

在不连续采样模式下转换标准组时，最后一次触发剩余的通道不足 ADC_CON0.ETRGN 指定的通道数时，不会接着从头转换，转换到序列最后一个通道就会停止，生成 NCHE 事件。转换完所有序列通道后，下一个触发信号将启动序列的第一个通道。在上述示例中，第 4 次触发 重新转换了序列的 0、2、3、5 通道。

插入组

可将 ADC_CON0 寄存器中的 ICHDCEN 位置 1 来使能此模式。在出现外部触发事件之后，可使用该模式逐通道（n=1）转换在 ADC_ICHS 寄存器中选择的序列。

出现外部触发时，将启动在 ADC_ICHS 寄存器中选择的下一个通道转换，直到序列中的所有转换均完成为止。通过 ADC_CHSL 寄存器中的 ISL 位定义总序列长度。

示例：

n = 1，ADC_CHSL.ISL=3，要转换的通道 = 1、3、4

第 1 次触发：转换通道 1

第 2 次触发：转换通道 3

第 3 次触发：转换通道 4 并生成 CHE 和 ICHE 事件

第 4 次触发：通道 1

转换完所有插入通道后，下一个触发信号将启动第一个插入通道的转换。在上述示例中，第 4 次触发重新转换了第 1 个插入通道。

不能同时使用自动插入和不连续采样模式。

不得同时为标准组和插入组设置不连续采样模式。只能选择其一进行不连续采样模式。

32. 4. 11 数据对齐

ADC_CON1.ALIGN 位用于选择转换后存储的数据的对齐方式,可选择左对齐和右对齐两种方式，如下图所示。

插入通道组的转换数据将加上 ADC_ICHOFFx 寄存器中写入的用户自定义偏移量,因此结果可以是一个负值，图中 EXS 位表示扩展的符号值。

对于标准组中的通道，不会减去任何偏移量，因此只有十二个位有效。

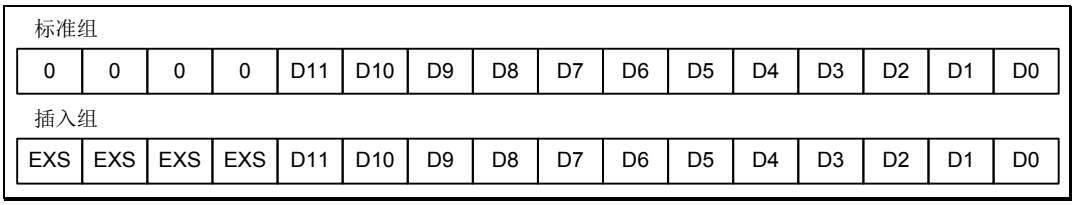


图 32-3 右对齐数据示意图

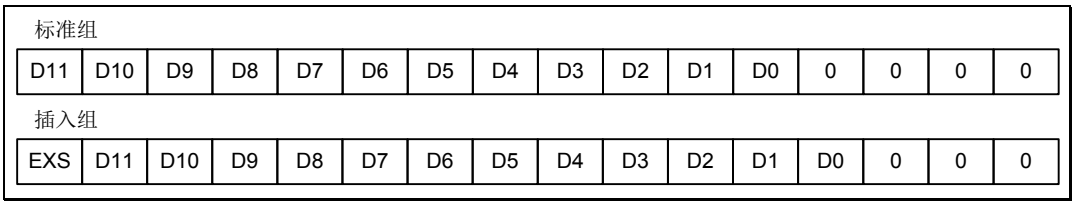


图 32-4 左对齐数据示意图

32. 4. 12 可独自设置各通道采样时间

ADC 会在数个 ADCCLK 周期内对输入电压进行采样，可使用 ADC_SMPT1 和 ADC_SMPT2 寄存器中的 CHTx 位修改周期数。每个通道均可以使用不同的采样时间进行采样。

总转换时间的计算公式如下：

Tconv = 采样时间 + 12 个周期

示例：

ADCCLK = 24MHz 且采样时间 = 2 个周期时：

Tconv = 2 + 12 = 14 个周期 = 0.583 μ s

32. 4. 13 外部触发转换和触发极性

可以通过配置 PIS 通道选择相应信号触发 ADC 转换，PIS 的配置请查看 PIS 相应章节描述。当 PIS 某一通道配置成 ADC 外部触发时，若相应的触发源事件发生，将自动触发标准序列或插入序列转换。

32. 4. 14 快速转换模式

可通过降低 ADC 分辨率来执行快速转换。ADC_CON0 寄存器中的 RSEL 位用于选择 ADC 转换的分辨率。每种分辨率的最小转换时间如下：

◇ 12 位：1 + 12 = 13 ADCCLK 周期

◇ 10 位：1 + 10 = 11 ADCCLK 周期

◇ 8 位：1 + 8 = 9 ADCCLK 周期

◇ 6 位：1 + 6 = 7 ADCCLK 周期

注：选择采样时间为一个周期。

32. 4. 15 数据管理

32. 4. 15. 1 使用DMA

标准组只有一个数据寄存器（ADC_NCHDR）用于存储 AD 转换结果值，所以，对于多个标准通道的转换，使用 DMA 可以快速存储数据，避免在上一次转换结果的值还未读出时新的 ADC 结果值又写入 ADC_NCHDR 寄存器，造成数据丢失。

每完成标准通道组中的一个通道转换后，都会生成一个 DMA 请求。这样便可将转换后的数据从 ADC_NCHDR 寄存器中传输到软件指定的目标内存位置。

32. 4. 15. 2 在不使用DMA的情况下管理转换序列

如果转换过程足够慢，则可使用软件来处理转换序列。在这种情况下，必须将 ADC_CON1 寄存器中的 NCHESEL 位置 1，才能使 NCHE 状态位在每次转换结束时置 1，而不仅是在转换序列结束时置 1。当 NCHESEL = 1 时，会自动使能溢出检测。因此，每当转换结束时，NCHE 都会置 1，并且可以读取 ADC_NCHDR 寄存器。如果数据丢失（溢出），则会将 ADC_STAT 寄存器中的 OVR 位置 1 并生成一个中断（如果 ADC_CON0.OVRIE 位已置 1）。

要在 NCHESEL 位置 1 时将 ADC 从 OVR 状态中恢复，请按以下步骤操作：

1. 将 ADC_STAT 寄存器中的 OVR 位清零

2. 触发 ADC 以开始转换

32. 4. 15. 3 在不使用DMA和溢出检测情况下进行转换

当 ADC 存在转换一个或多个通道时不需要每次读取数据的情况时，例如使用模拟看门狗，必须禁止 DMA (ADC_CON1.DMA = 0)并且仅在序列结束 (NCHESEL = 0) 时才将 NCHE 位置 1。这样溢出检测被禁止。

32. 4. 16 ADC中断

当模拟看门狗状态位和溢出状态位分别置 1 时，标准组和插入组在转换结束时可能会产生中断。可以使用单独的中断使能位以实现灵活性。

中断事件	事件标志位	使能控制位
结束标准组的转换	NCHE	NCHEIE
结束插入组的转换	ICHE	ICHEIE
发生模拟看门狗事件	AWDF	AWDIE
溢出	OVR	OVRIE

表 32-2 ADC 中断

32. 5 特殊功能寄存器

32. 5. 1 寄存器列表

ADC 寄存器列表		
名称	偏移地址	描述
ADC_STAT	000_H	ADC 状态寄存器
ADC_CLR	004_H	ADC 清零寄存器
ADC_CON0	008_H	ADC 控制寄存器 0
ADC_CON1	$00C_H$	ADC 控制寄存器 1
ADC_SMPT1	010_H	ADC 采样时间寄存器 1
ADC_SMPT2	014_H	ADC 采样时间寄存器 2
ADC_SMPT3	018_H	ADC 采样时间寄存器 3
—	$01C_H$	—
ADC_ICHOFF1	020_H	ADC 插入通道数据偏移寄存器 1
ADC_ICHOFF2	024_H	ADC 插入通道数据偏移寄存器 2
ADC_ICHOFF3	028_H	ADC 插入通道数据偏移寄存器 3
ADC_ICHOFF4	$02C_H$	ADC 插入通道数据偏移寄存器 4
ADC_NCHS1	030_H	ADC 标准通道序列寄存器 1
ADC_NCHS2	034_H	ADC 标准通道序列寄存器 2
ADC_NCHS3	038_H	ADC 标准通道序列寄存器 3
ADC_NCHS4	$03C_H$	ADC 标准通道序列寄存器 4
ADC_ICHS	040_H	ADC 插入通道序列寄存器
ADC_CHSL	044_H	ADC 通道序列长度寄存器
ADC_WDTH	048_H	ADC 看门狗高阈值寄存器
ADC_WDTL	$04C_H$	ADC 看门狗低阈值寄存器
ADC_ICHDR1	050_H	ADC 插入通道数据寄存器 1
ADC_ICHDR2	054_H	ADC 插入通道数据寄存器 2
ADC_ICHDR3	058_H	ADC 插入通道数据寄存器 3
ADC_ICHDR4	$05C_H$	ADC 插入通道数据寄存器 4
ADC_NCHDR	060_H	ADC 标准通道数据寄存器
ADC_CCR	064_H	ADC 通用控制寄存器

32. 5. 2 寄存器描述

32. 5. 2. 1 ADC状态寄存器（ADC_STAT）

ADC 状态寄存器（ADC_STAT）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-10	—	**保留**
ICHS	Bit 9	R	**插入通道转换开始标志位** 0：未开始插入转换或标志位已被清除 1：已开始插入转换 注：该位由硬件置1，通过操作ADC_CLR清零
NCHS	Bit 8	R	**标准通道转换开始标志位** 0：未开始标准转换或标志位已被清除 1：已开始标准转换 注：该位由硬件置1，通过操作ADC_CLR清零
Reserved	Bit 7-4	—	**保留**
OVR	Bit 3	R	**转换溢出标志位** 0：未发生溢出或标志位已被清除 1：发生溢出 注 1：溢出检测仅在 DMA=1 或 NCHESEL=1 时使能 注 2：该位由硬件置 1，通过操作 ADC_CLR 清零
ICHE	Bit 2	R	**插入通道转换结束标志位** 0：所有插入转换未完成或标志位已被清除 1：所有插入转换已完成 注：该位由硬件置 1，通过操作 ADC_CLR 清零
NCHE	Bit 1	R	**标准通道转换结束标志位** NCHESEL = 0时 0：标准转换序列未完成或标志位已被清除 1：标准转换序列已完成 NCHESEL = 1 时 0：单次标准转换未完成或标志位已被清除 1：单次标准转换已完成 注：该位由硬件置1，通过操作ADC_CLR清零
AWDF	Bit 0	R	**模拟看门狗标志位** 0：未发生看门狗事件或标志位已被清除 1：已发生看门狗事件 注：该位由硬件置1，通过操作ADC_CLR清零

32. 5. 2. 2 ADC清零寄存器（ADC_CLR）

ADC 清零寄存器（ADC_CLR）

偏移地址：04_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-10	—	**保留**
ICHS	Bit 9	W1	**插入通道开始标志位清零** 0：无操作 1：插入转换开始标志位清零
NCHS	Bit 8	W1	**标准通道开始标志位清零** 0：无操作 1：标准转换开始标志位清零
Reserved	Bit 7-4	—	**保留**
OVR	Bit 3	W1	**转换溢出标志位清零** 0：无操作 1：转换溢出标志位清零
ICHE	Bit 2	W1	**插入转换结束标志位清零** 0：无操作 1：插入转换结束标志位清零
NCHE	Bit 1	W1	**标准转换结束标志位清零** 0：无操作 1：标准转换结束标志位清零
AWDF	Bit 0	W1	**模拟看门狗标志位清零** 0：无操作 1：模拟看门狗标志位清零

32. 5. 2. 3 ADC控制寄存器 0（ADC_CON0）

ADC 控制寄存器 0（ADC_CON0）																															
偏移地址：08_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					OVRIE	RSEL		NCHWDEN	ICHWDTEN	Reserved						ETRGN			ICHDCEN	NCHDCEN	IAUTO	AWDSGL	SCANEN	ICHEIE	AWDIE	NCHEIE	AWDCH				

Reserved	Bit 31-27	—	**保留**
OVRIE	Bit 26	R/W	**溢出中断使能位** 0：禁止 1：使能
RSEL	Bit 25-24	R/W	**ADC 转换精度选择位** 00：6 位 01：8 位 10：10 位 11：12 位
NCHWDEN	Bit 23	R/W	**标准通道看门狗使能位** 0：禁止 1：使能
ICHWDTEN	Bit 22	R/W	**插入通道看门狗使能位** 0：禁止 1：使能
Reserved	Bit 21-16	—	**保留**
ETRGN	Bit 15-13	R/W	**外部触发不连续转换通道数** 000：1 个通道 001：2 个通道 …… 111：8 个通道
ICHDCEN	Bit 12	R/W	**插入通道不连续转换使能位** 0：禁止 1：使能
NCHDCEN	Bit 11	R/W	**标准通道不连续转换使能位** 0：禁止 1：使能
IAUTO	Bit 10	R/W	**插入组自动转换使能位** 0：禁止 1：使能
AWDSGL	Bit 9	R/W	**扫描模式单一通道模拟看门狗使能位** 0：禁止 1：使能

SCANEN	Bit 8	R/W	**扫描模式使能位** 0：禁止 1：使能
ICHEIE	Bit 7	R/W	**插入通道转换完成中断使能位** 0：禁止 1：使能
AWDIE	Bit 6	R/W	**模拟看门狗中断使能位** 0：禁止 1：使能
NCHEIE	Bit 5	R/W	**标准通道转换完成中断使能位** 0：禁止 1：使能
AWDCH	Bit 4-0	R/W	**模拟看门狗通道选择位** 00000：ADC 输入通道 0 00001：ADC 输入通道 1 …… 10010：ADC 输入通道 8 其他：保留

32. 5. 2. 4 ADC控制寄存器 1（ADC_CON1）

ADC 控制寄存器 1（ADC_CON1）																															
偏移地址：$0C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	NCHTRG	NETS		Reserved					ICHTRG	IETS		Reserved								ALIGN	NCHESEL	Reserved	DMA	Reserved						CM	ADCEN

Reserved	Bit 31	—	**保留**
NCHTRG	Bit 30	W1	**标准通道触发位** 0：无操作 1：触发开始标准通道转换
NETS	Bit 29-28	R/W	**标准转换外部触发极性选择位** 00：外部触发禁止 01：上升沿触发 10：下降沿触发 11：上升沿和下降沿触发
Reserved	Bit 27-23	—	**保留**
ICHTRG	Bit 22	W1	**插入通道触发位** 0：无操作 1：触发开始插入通道转换
IETS	Bit 21-20	R/W	**插入转换外部触发极性选择位** 00：外部触发禁止 01：上升沿触发 10：下降沿触发 11：上升沿和下降沿触发
Reserved	Bit 19-12	—	**保留**
ALIGN	Bit 11	R/W	**数据对齐方式位** 0：右对齐 1：左对齐
NCHESEL	Bit 10	R/W	**标准转换结束标志选择位** 0：每个标准转换序列结束时将 STAT.NCHE 位置 1 1：每个标准转换结束时将 STAT.NCHE 位置 1
Reserved	Bit 9	—	**保留**
DMA	Bit 8	R/W	**DMA 访问使能位** 0：禁止 1：使能
Reserved	Bit 7-2	—	**保留**
CM	Bit 1	R/W	**转换模式** 0：单次转换

			1：连续转换
ADCEN	Bit 0	R/W	**ADC 使能位** 0：禁止 1：使能

32. 5. 2. 5 ADC采样时间寄存器 1（ADC_SMPT1）

ADC 采样时间寄存器 1（ADC_SMPT1）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHT7				CHT6				CHT5				CHT4				CHT3				CHT2				CHT1				CHT0			

CHT<y>	Bit 31-0	R/W	**通道 y 采样时间选择位（y=0..7）** 0000：1 个周期 0001：2 个周期 0010：4 个周期 0011：15 个周期 其他：预留

32. 5. 2. 6 ADC采样时间寄存器 2（ADC_SMPT2）

ADC 采样时间寄存器 2（ADC_SMPT2）

偏移地址：14_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHT15				CHT14				CHT13				CHT12				CHT11				CHT10				CHT9				CHT8			

CHT<y>	Bit 31-0	R/W	**通道 y 采样时间选择位（y=8..15）** 0000：1 个周期 0001：2 个周期 0010：4 个周期 0011：15 个周期 其他：预留

32. 5. 2. 7 ADC采样时间寄存器 3（ADC_SMPT3）

ADC 采样时间寄存器 3（ADC_SMPT3）

偏移地址：18_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CHT19				CHT18				CHT17				CHT16			

Reserved	Bit 31-16	-	**保留**
CHT<y>	Bit 15-0	R/W	**通道 y 采样时间选择位（y=16..19）** 0000：1 个周期 0001：2 个周期 0010：4 个周期 0011：15 个周期 其他：预留

32. 5. 2. 8 ADC插入通道数据偏移寄存器 1（ADC_ICHOFF1）

ADC 插入通道数据偏移寄存器 1（ADC_ICHOFF1）

偏移地址：20_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				IOFF4				IOFF3				IOFF2				IOFF1			

Reserved	Bit 31-12	—	**保留**
IOFF<y>	Bit 11-0	R/W	**插入通道 1 数据偏移量** ADC_ICHDR1 中的数据为原始转换数据加上偏移量

32. 5. 2. 9 ADC插入通道数据偏移寄存器 2（ADC_ICHOFF2）

ADC 插入通道数据偏移寄存器 2（ADC_ICHOFF2）

偏移地址：24_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				IOFF4			IOFF3			IOFF2			IOFF1		

Reserved	Bit 31-12	—	保留
IOFF<y>	Bit 11-0	R/W	**插入通道 2 数据偏移量** ADC_ICHDR2 中的数据为原始转换数据加上偏移量

32. 5. 2. 10 ADC插入通道数据偏移寄存器 3（ADC_ICHOFF3）

ADC 插入通道数据偏移寄存器 3（ADC_ICHOFF3）

偏移地址：28_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				IOFF4			IOFF3			IOFF2			IOFF1		

Reserved	Bit 31-12	—	保留
IOFF<y>	Bit 11-0	R/W	**插入通道 3 数据偏移量** ADC_ICHDR3 中的数据为原始转换数据加上偏移量

32. 5. 2. 11 ADC插入通道数据偏移寄存器 4（ADC_ICHOFF4）

ADC 插入通道数据偏移寄存器 4（ADC_ICHOFF4）

偏移地址：$2C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				IOFF4			IOFF3			IOFF2			IOFF1		

Reserved	Bit 31-12	—	**保留**
IOFF<y>	Bit 11-0	R/W	**插入通道 4 数据偏移量** ADC_ICHDR4 中的数据为原始转换数据加上偏移量

32. 5. 2. 12 ADC标准通道序列寄存器 1（ADC_NCHS1）

ADC 标准通道序列寄存器 1（ADC_NCHS1）

偏移地址：30_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			NS4					Reserved			NS3					Reserved			NS2					Reserved			NS1				

Reserved	Bit 31-29	—	**保留**
NS4	Bit 28-24	R/W	**标准序列第 4 次转换通道编号** 00000~10010：通道 0~18 其他：预留
Reserved	Bit 23-21	—	**保留**
NS3	Bit 20-16	R/W	**标准序列第 3 次转换通道编号** 00000~10010：通道 0~18 其他：预留
Reserved	Bit 15-13	—	**保留**
NS2	Bit 12-8	R/W	**标准序列第 2 次转换通道编号** 00000~10010：通道 0~18 其他：预留
Reserved	Bit 7-5	—	**保留**
NS1	Bit 4-0	R/W	**标准序列第 1 次转换通道编号** 00000~10010：通道 0~18 其他：预留

32. 5. 2. 13 ADC标准通道序列寄存器 2（ADC_NCHS2）

ADC 标准通道序列寄存器 2（ADC_NCHS2）

偏移地址：34_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			NS8					Reserved			NS7					Reserved			NS6					Reserved			NS5				

Reserved	Bit 31-29	—	**保留**
NS8	Bit 28-24	R/W	**标准序列第 8 次转换通道编号** 00000~10010：通道 0~18 其他：预留
Reserved	Bit 23-21	—	**保留**
NS7	Bit 20-16	R/W	**标准序列第 7 次转换通道编号** 00000~10010：通道 0~18 其他：预留
Reserved	Bit 15-13	—	**保留**
NS6	Bit 12-8	R/W	**标准序列第 6 次转换通道编号** 00000~10010：通道 0~18 其他：预留
Reserved	Bit 7-5	—	**保留**
NS5	Bit 4-0	R/W	**标准序列第 5 次转换通道编号** 00000~10010：通道 0~18 其他：预留

32. 5. 2. 14 ADC标准通道序列寄存器 3（ADC_NCHS3）

ADC 标准通道序列寄存器 3（ADC_NCHS3）

偏移地址：38_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			NS12					Reserved			NS11					Reserved			NS10					Reserved			NS9				

Reserved	Bit 31-29	—	**保留**
NS12	Bit 28-24	R/W	**标准序列第 12 次转换通道编号** 00000~10010：通道 0~18 其他：预留
Reserved	Bit 23-21	—	**保留**
NS11	Bit 20-16	R/W	**标准序列第 11 次转换通道编号** 00000~10010：通道 0~18 其他：预留
Reserved	Bit 15-13	—	**保留**
NS10	Bit 12-8	R/W	**标准序列第 10 次转换通道编号** 00000~10010：通道 0~18 其他：预留
Reserved	Bit 7-5	—	**保留**
NS9	Bit 4-0	R/W	**标准序列第 9 次转换通道编号** 00000~10010：通道 0~18 其他：预留

32. 5. 2. 15 ADC标准通道序列寄存器 4（ADC_NCHS4）

ADC 标准通道序列寄存器 4（ADC_NCHS4）

偏移地址：$3C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			NS16					Reserved			NS15					Reserved			NS14					Reserved			NS13				

Reserved	Bit 31-29	—	**保留**
NS16	Bit 28-24	R/W	**标准序列第 16 次转换通道编号** 00000~10010：通道 0~18 其他：预留
Reserved	Bit 23-21	—	**保留**
NS15	Bit 20-16	R/W	**标准序列第 15 次转换通道编号** 00000~10010：通道 0~18 其他：预留
Reserved	Bit 15-13	—	**保留**
NS14	Bit 12-8	R/W	**标准序列第 14 次转换通道编号** 00000~10010：通道 0~18 其他：预留
Reserved	Bit 7-5	—	**保留**
NS13	Bit 4-0	R/W	**标准序列第 13 次转换通道编号** 00000~10010：通道 0~18 其他：预留

32. 5. 2. 16 ADC插入通道序列寄存器（ADC_ICHS）

ADC 插入通道序列寄存器（ADC_ICHS）																													
偏移地址：40_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			IS4					Reserved			IS3					Reserved			IS2					Reserved			IS1			

Reserved	Bit 31-29	—	**保留**
IS4	Bit 28-24	R/W	**插入序列第 4 次转换通道编号** 00000~10010：通道 0~18 其他：预留
Reserved	Bit 23-21	—	**保留**
IS3	Bit 20-16	R/W	**插入序列第 3 次转换通道编号** 00000~10010：通道 0~18 其他：预留
Reserved	Bit 15-13	—	**保留**
IS2	Bit 12-8	R/W	**插入序列第 2 次转换通道编号** 00000~10010：通道 0~18 其他：预留
Reserved	Bit 7-5	—	**保留**
IS1	Bit 4-0	R/W	**插入序列第 1 次转换通道编号** 00000~10010：通道 0~18 其他：预留

32. 5. 2. 17 ADC通道序列长度寄存器（ADC_CHSL）

ADC 通道序列长度寄存器（ADC_CHSL）

偏移地址：44_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-10	—	**保留**
ISL	Bit 9-8	R/W	**插入通道序列长度** 00：1 次转换 01：2 次转换 10：3 次转换 11：4 次转换
Reserved	Bit 7-4	—	**保留**
NSL	Bit 3-0	R/W	**标准通道序列长度** 0000：1 次转换 0001：2 次转换 …… 1111：16 次转换

32. 5. 2. 18 ADC看门狗高阈值寄存器（ADC_WDTH）

ADC 看门狗高阈值寄存器（ADC_WDTH）

偏移地址：48_H

复位值：$00000000\_00000000\_00001111\_11111111_B$

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

Reserved	Bit 31-12	—	**保留**
HT	Bit 11-0	R/W	**模拟看门狗高阈值** 当原始数据大于高阈值时产生看门狗事件

32. 5. 2. 19 ADC看门狗低阈值寄存器（ADC_WDTL）

ADC 看门狗低阈值寄存器（ADC_WDTL）

偏移地址：$4C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-12	—	**保留**
LT	Bit 11-0	R/W	**模拟看门狗低阈值** 当原始数据小于低阈值时产生看门狗事件

32. 5. 2. 20 ADC插入通道数据寄存器 1（ADC_ICHDR1）

ADC 插入通道数据寄存器 1（ADC_ICHDR1）

偏移地址：50_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	**保留**
VAL	Bit 15-0	R	**插入通道 1 转换数据** 该数据为对齐之后的数据

32. 5. 2. 21 ADC插入通道数据寄存器 2（ADC_ICHDR2）

ADC 插入通道数据寄存器 2（ADC_ICHDR2）

偏移地址：54_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	**保留**
VAL	Bit 15-0	R	**插入通道 2 转换数据** 该数据为对齐之后的数据

32. 5. 2. 22 ADC插入通道数据寄存器 3（ADC_ICHDR3）

ADC 插入通道数据寄存器 3（ADC_ICHDR3）

偏移地址：58_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	保留
VAL	Bit 15-0	R	**插入通道 3 转换数据** 该数据为对齐之后的数据

32. 5. 2. 23 ADC插入通道数据寄存器 4（ADC_ICHDR4）

ADC 插入通道数据寄存器 4（ADC_ICHDR4）

偏移地址：$5C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	保留
VAL	Bit 15-0	R	**插入通道 4 转换数据** 该数据为对齐之后的数据

32. 5. 2. 24 ADC标准通道数据寄存器（ADC_NCHDR）

ADC 标准通道数据寄存器（ADC_NCHDR）

偏移地址：60_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	保留
VAL	Bit 15-0	R	**标准通道转换数据** 该数据为对齐之后的数据

32. 5. 2. 25 ADC通用控制寄存器（ADC_CCR）

ADC 通用控制寄存器（ADC_CCR）

偏移地址：64_H

复位值：$00000000\_00000000\_11000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCH_PBUF_EN	ADCH_NBUF_EN	ADCH_INBUF_CALEN	TRMEN	Reserved								VREFOEN	VRNSEL	VRPSEL		PWRMODSEL	Reserved		DIFFEN	IREFEN	VRBUFEN	VCMBUFEN	VREFEN	Reserved						CKDIV	

ADCH_PBUF_EN	Bit 31	R/W	**ADC 通道正相输入 buffer 使能位** 使用 ADC 时必须置一 注：仅 ADC0 有效
ADCH_NBUF_EN	Bit 30	R/W	**ADC 通道负相输入 buffer 使能位** 使用 ADC 时必须置一 注：仅 ADC0 有效
ADCH_INBUF_CALEN	Bit 29	R/W	**ADC 输入 buffer 校准使能位** 0：禁止 1：使能 注：仅 ADC0 有效
TRMEN	Bit 28	R/W	**ADC 数据修调使能位** 0：禁止 1：使能
Reserved	Bit 27-20	—	**保留**
VREFOEN	Bit 19	R/W	**内部2.0V参考电压输出至引脚VREFP使能位** 0：禁止 1：使能 注：仅ADC0有效
VRNSEL	Bit 18	R/W	**负向参考电压选择位** 0：VSS 1：VREFN 管脚 注：仅 ADC0 有效
VRPSEL	Bit 17-16	R/W	**正向参考电压选择位** 00：VDD 01：内部 2.0V 参考电压 10：VREFP 管脚（不经过 Buffer） 11：VREFP 管脚（经过 Buffer） 注：仅 ADC0 有效
PWRMODSEL	Bit 15	R/W	**VREF 工作模式选择** 0：低速模式（ADC 时钟低于 1MHz） 1：高速模式（ADC 时钟高于 1MHz）

			注1：当ADC时钟小于1MHz需使能低速模式 注2：仅ADC0有效
Reserved	Bit 14-13	—	**保留**
DIFFEN	Bit 12	R/W	**差分模式使能位** 0：禁止 1：使能
IREFEN	Bit 11	R/W	**Buffer 参考电流使能位** 0：禁止 1：使能 注：仅 ADC0 有效
VRBUFEN	Bit 10	R/W	**参考电压 Buffer 使能位** 0：禁止 1：使能 注：仅 ADC0 有效
VCMBUFEN	Bit 9	R/W	**共模电压 Buffer 使能位** 0：禁止 1：使能 注：仅 ADC0 有效
VREFEN	Bit 8	R/W	**内部参考电压使能位** 0：禁止 1：使能 注：仅ADC0有效
Reserved	Bit 7-3	—	**保留**
CKDIV	Bit 2-0	R/W	**ADC 时钟分频选择位** 000：1 分频 001：2 分频 …… 111：128 分频

第33章 数模转换器（DAC）

33. 1 概述

DAC 模块可以将数值转化为相对应的模拟电压信号输出，该 DAC 支持 rail-to-rail 转换，12-bit 分辨率，可以设置两个单端输出缓冲，也可配置为差分输出，可以应用在传感器或者音频领域。

33. 2 特性

- 12-bit 分辨率
- 采样率 500k/s
- 可配置的参考源选择
- 两个输出通道，可配置为差分输出
- 可配置的转换时钟分频，分频比可选范围为 1~128
- 支持两种转换触发
  - 写数据
  - PIS 输入
- 支持定时自动刷新
  - 输入时钟预分频可设为 16~64，作为定时计数时钟
  - 每一个通道可独立配置刷新时间
- 转换完成产生中断，两通道有独立的中断标志位
- 转换完成后产生 PIS 脉冲输出，两通道独立控制
- 转换完成后产生 DMA 请求，两通道独立控制
- 支持偏移误差和增益误差校准
- 可输出给 ADC
- 可选波产生模式
- 可选高强度驱动

33. 3 结构框图

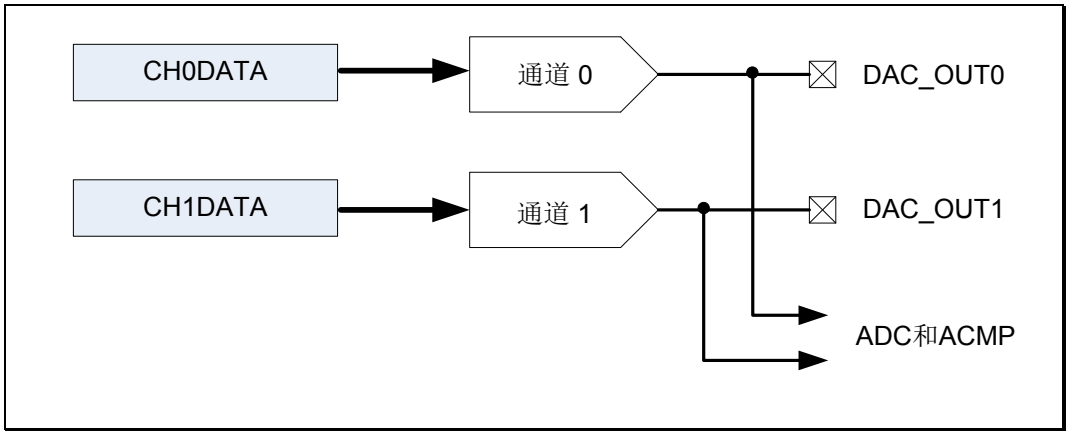


图 33-1 DAC 结构框图

数模转换器用来将输入的数字量转换为与之对应的模拟量输出，当输入数字量发生变化时，输出模拟量也会同步变化，其变化在一个给定的电压范围内，其参考电压可以在多个参考源中选择其一，所对应的转换电压范围会有不同。

33.4 功能描述

33.4.1 DAC控制

33.4.1.1 数字模拟转换

本模块支持两个通道的 DAC 转换，分别对应独立的输入数据寄存器（DAC_CH0DATA/DAC_CH1DATA），两个通道可独立工作输出两个单端的模拟量，也可以组合起来输出差分模拟量（此时仅使用 DAC_CH0DATA 作为输入数据寄存器）。支持三种工作方式：连续转换、采样保持以及采样关闭。

33.4.1.2 连续转换方式

在连续转换方式中，两个 DAC 通道连续地驱动对应模拟输出量，模拟量的变化取决于对应的数据缓存区的值变化，DAC 模拟输出一直被保持直到对应数据缓存区的值发生变化，因此，这种模式下无需刷新数据缓存区。

33.4.1.3 采样保持方式

在采样保持方式中，使能 DAC 并触发转换后才开始工作，当数据缓存被采样并且转换出模拟量后，数字控制器不工作，以此降低一部分的功耗消耗，同时在未收到刷新转换之前的一定时间内，由 DAC 模拟保持电路来维持着模拟量输出电压，采样周期是 DAC 时钟经预分频之后的一个时钟周期。

33.4.1.4 采样关闭方式

在采样关闭方式中，使能 DAC 并触发转换后才开始工作，当数据缓存被采样并且转换出模拟量后，数字控制器和 DAC 模拟电路都被关闭，参考源也被关闭，以此进一步降低该模块的功耗消耗，在下次开启转换，外围器件来维持着模拟量输出电压，例如电容等，这就意味着下次转换开启后需要经历一个上电稳定的过程。

33.4.1.5 开始转换

在 DAC 开始工作之前，必须先使能 DAC，然后向数据缓存区 DAC_CHxDATA 写入数据就会触发一次 DAC 的转换工作，当然，数据缓存区也可以映射到组合数据缓存寄存器 DAC_COMBDATA，向这个寄存器写入数据可以同时触发被使能的 DAC 通道。

当 PISEN 设置为 1 时，DAC 的转换工作就不会被写数据缓存区而触发，只有当接收到被 PISSEL 选为触发源的信号上有个高脉冲的时候，才会触发 DAC 开始工作。

DAC_STAT 中的 CH0BSY、CH1BSY 指示着对应的 DAC 数据缓存区是否已经被转换，当对应值为 1 时，说明还未被转换或者正在转换过程中，否则说明已经转换完成。

33.4.1.6 时钟预分频

DAC 内部集成了时钟预分频的电路，在系统时钟预分频配置为 1~128 之后，作为 DAC 工作的时钟频率，具体公式如下：

$$f_{DAC\_CLK} = fosc / 2^{PRES}$$

fosc 为系统时钟，f_{DAC_CLK} 为 DAC 转换时钟频率，f_{DAC_CLK} 的频率不能高于 500Khz。

正常情况下，DAC 内部的预分频逻辑在使能后会一直不停的计数，DAC 的转换是同步

于 f_{DAC_CLK}，所以从软件触发 DAC 转换到 DAC 真正开始转换，这段时间是不可预知的，然而，当配置 DAC_CON.CH0PRESRST=1 时，每次在 DAC 的 CH0 转换被触发时会对预分频计数进行复位，然后开始计数，这就可以预知 DAC 真正开始转换时间点，实现每次转换都开始于同一时刻。

33. 4. 2 参考源选择

DAC 参考源与 ADC 参考源共用。可参考 ADC0 中对参考源的寄存器配置。

只有当两个转换通道都被关闭时，DAC 的参考源才可以改变配置，且在使用某一参考源之前先要使能对应参考源，等参考源稳定后才开始正常转换，这段时间称为 warm-up 时间，选择不同参考源，对应的 warm-up 时间会不同，选内部 2.5v/1.5v 时 warm-up 时间约为 5 个 f_{DAC_CLK}，而选 VDD 时为 1 个 f_{DAC_CLK}。

33. 4. 3 输出模式

DAC 转换有三种工作模式，一种是单端转换输出，一种是双端差分转换输出，还有一种是输出固定正弦信号。

33. 4. 3. 1 单端转换输出模式

单端转换输出模式下，两个通道独立工作，DAC_CH0DATA.VAL 对应的是 DAC 通道 0 输出 V_{DAC_out0}，DAC_CH1DATA.VAL 对应的是 DAC 通道 1 输出 V_{DAC_out1}，输出的电压值理论计算值为：

$$V_{outx} = V_{DAC\_outx} - VSS = Vref * DAC\_CHxDATA.VAL/4095$$

其中，DAC_CHxDATA.VAL 是对应的 12-bit 无符号 DAC 转换数据。

33. 4. 3. 2 差分转换输出模式

差分转换输出模式下，两个通道联合工作，DAC 通道 1 为正端，DAC 通道 0 为负端，使用 DAC_CH0DATA.VAL 作为对应 DAC 通道 0 转换数据，而使用 DAC_CH1DATA.VAL 作为对应 DAC 通道 1 转换数据，输出的电压值理论计算值为：

$$V_{out} = V_{DAC\_out1} - V_{DAC\_out0} = Vref * DAC\_CHxDATA.VAL/2047$$

其中，DAC_CHxDATA.VAL 是对应的 12-bit 有符号 DAC 转换数据。

33. 4. 3. 3 正弦信号输出模式

正弦信号输出模式下，使用内部固化 16 个采样点的正弦数据作为转换数据，从零电平作为起点输出，若为单端输出，在 DAC 通道 0 输出正弦信号，若设置 DIFEN 为 1，则为差分输出模式，在通道 0 输出正弦信号，在通道 1 输出反相正弦信号。其正弦的频率为：

$$f_{sine} = f_{DAC\_CLK} / 32$$

33. 4. 4 中断和PIS触发

每个 DAC 通道都有独立的中断标志来指明转换已经结束，并且可以开始转换下一数据，所有 DAC 的中断源共享同一个中断入口向量，当中断使能打开时，向对应的中断标志写 1 也会产生中断，中断需要软件清零。

每个 DAC 通道转换完成后也会产生一个系统时钟脉冲宽度的 PIS 信号，作为 PIS 其他互联模块的触发源。

每个 DAC 通道转换完成后也会产生一个 DMA 请求信号，当该对应的通道数据缓存区被更新且请求被响应后，请求信号会自动清除。

33. 4. 5 模拟输出

每个 DAC 转换通道输出都可以独立地输出到端口上，也可作为内部信号进行处理，配置 DAC_CON.OUTMD 可以选择输出到 ACMP 或者 ADC 模块中，进行进一步处理。

当 DAC 被禁止时，DAC 通道输出均为高阻；当 DAC 被使能时，若配置 DAC_CON.OUTENPIS=1，输出使能由对应的通道寄存器的 PISSEL 位所选择的 PIS 信号源控制，当该 PIS 信号源为低时，DAC 通道输出也为高阻，反之，DAC 通道正常输出。

33. 4. 6 调校

原始 DAC 转换会有一定的转换偏差，经调校后可以精确转换，所以调校过程影响着 DAC 的工作性能。该 DAC 模块支持自动调校，配置 DAC_CAL.SELF_CALEN=1，开启 DAC 自动调校过程，软件等待 5us 待调校结束后将 DAC_CAL.SELF_CALEN 清零。

33. 5 特殊功能寄存器

33. 5. 1 寄存器列表

DAC 寄存器列表		
名称	偏移地址	描述
DAC_CON	000_H	DAC控制寄存器
DAC_STAT	004_H	DAC状态寄存器
DAC_CH0CTRL	008_H	DAC通道0控制寄存器
DAC_CH1CTRL	$00C_H$	DAC通道1控制寄存器
DAC_IES	010_H	DAC中断使能设置寄存器
DAC_IEC	014_H	DAC中断使能清除寄存器
DAC_IEV	018_H	DAC中断使能有效寄存器
DAC_RIF	$01C_H$	DAC 原始中断标志寄存器
DAC_IFM	020_H	DAC 中断标志屏蔽寄存器
DAC_IFC	024_H	DAC 中断标志清除寄存器
DAC_CH0DATA	028_H	DAC 通道 0 输入数据寄存器
DAC_CH1DATA	$02C_H$	DAC 通道 1 输入数据寄存器
DAC_COMBDATA	030_H	DAC 组合输入数据寄存器
DAC_CAL	034_H	DAC 校准寄存器

33. 5. 2 寄存器描述

33. 5. 2. 1 DAC控制寄存器（DAC_CON）

DAC 控制寄存器（DAC_CON）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RCYCLSEL		Reserved	PRES			Reserved								CH0PRESRST	OUTENPIS	OUTMD		CONVMD		SINEMD	DIFEN

Reserved	Bit 31-22	—	**保留**
RCYCLSEL	Bit 21-20	R/W	**自动刷新周期选择：** 可选 4~32 个周期刷新一次 00：4 个系统时钟周期 01：8 个系统时钟周期 10：16 个系统时钟周期 11：32 个系统时钟周期
Reserved	Bit 19	—	**保留**
PRES	Bit 18-16	R/W	**分频比设置** 设置时钟分频为 2^PRES
Reserved	Bit 15-8	—	**保留**
CH0PRESRST	Bit 7	R/W	**启动通道 0 时分频器复位控制位** 0：启动通道 0 时分频不复位 1：启动通道 0 时分频复位
OUTENPIS	Bit 6	R/W	**PIS 使能对 DAC 输出的控制位：** 0：DAC 输出一直使能 1：DAC 输出使能由 PIS 控制
OUTMD	Bit 5-4	R/W	**输出模式选择位：** 00：禁止 DAC 输出到 pin 和 ADC 01：PIN（使能 DAC 输出到 pin，禁止 DAC 输出到 ADC 和 ACMP） 10：ADC（禁止 DAC 输出到 pin，使能 DAC 输出到 ADC 和 ACMP） 11：PINADC（使能 DAC 输出到 pin、ADC 和 ACMP）
CONVMD	Bit 3-2	R/W	**转换模式配置：** 00：连续模式 01：采样保持 10：采样关闭
SINEMD	Bit 1	R/W	**正弦模式选择位：** 0：无效

			1：正弦模式
DIFEN	Bit 0	R/W	**差分模式使能位：** 0：无效 1：使能差分模式

33. 5. 2. 2 DAC状态寄存器（DAC_STAT）

DAC 状态寄存器（DAC_STAT）

偏移地址：04_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														CH1BSY	CH0BSY

Reserved	Bit 31-2	—	**保留**
CH1BSY	Bit 1	R	**通道 1 忙状态标志位：** 0：通道 1 转换空闲 1：通道 1 正忙于转换
CH0BSY	Bit 0	R	**通道 0 忙状态标志位：** 0：通道 0 转换空闲 1：通道 0 正忙于转换

33. 5. 2. 3 DAC通道 0 控制寄存器（DAC_CH0CTRL）

DAC 通道 0 控制寄存器（DAC_CH0CTRL）																															
偏移地址：08_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								PISSEL				Reserved	PISEN	RCYCLEN	EN

Reserved	Bit 31-8	—	保留
PISSEL	Bit 7-4	R/W	**PIS输入通道选择位：** 0：PISCH0触发通道0转换 1：PISCH1触发通道0转换 2：PISCH2触发通道0转换 3：PISCH3触发通道0转换 4：PISCH4触发通道0转换 5：PISCH5触发通道0转换 6：PISCH6触发通道0转换 7：PISCH7触发通道0转换 8：PISCH8触发通道0转换 9：PISCH9触发通道0转换 10：PISCH10触发通道0转换 11：PISCH11触发通道0转换 其它：保留
Reserved	Bit 3	—	保留
PISEN	Bit 2	R/W	**通道0转换触发使能位：** 0：无效 1：使能
RCYCLEN	Bit 1	R/W	**通道0自动刷新使能位：** 0：无效 1：使能 自动刷新周期由控制寄存器的RCYCLSEL位决定。
EN	Bit 0	R/W	**通道0使能位：** 0：无效 1：使能

33. 5. 2. 4 DAC通道 1 控制寄存器（DAC_CH1CTRL）

DAC 通道 1 控制寄存器（DAC_CH1CTRL）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-8	—	保留
PISSEL	Bit 7-4	R/W	**PIS输入通道选择位：** 0：PISCH0触发通道1转换 1：PISCH1触发通道1转换 2：PISCH2触发通道1转换 3：PISCH3触发通道1转换 4：PISCH4触发通道1转换 5：PISCH5触发通道1转换 6：PISCH6触发通道1转换 7：PISCH7触发通道1转换 8：PISCH8触发通道1转换 9：PISCH9触发通道1转换 10：PISCH10触发通道1转换 11：PISCH11触发通道1转换 其它：保留
Reserved	Bit 3	—	保留
PISEN	Bit 2	R/W	**通道1转换触发使能位：** 0：无效 1：使能
RCYCLEN	Bit 1	R/W	**通道1自动刷新使能位：** 0：无效 1：使能 自动刷新周期由控制寄存器的RCYCLSEL位决定。
EN	Bit 0	R/W	**通道1使能位：** 0：无效 1：使能

33. 5. 2. 5 DAC中断使能设置寄存器（DAC_IES）

DAC 中断使能设置寄存器（DAC_IES）																													
偏移地址：10_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										CH1UDF	CH0UDF	Reserved		CH1	CH0

Reserved	Bit 31-6	—	**保留**
CH1UDF	Bit 5	W	**通道 1 数据下溢中断使能位：** 0：禁止通道 1 数据下溢中断 1：使能通道 1 数据下溢中断
CH0UDF	Bit 4	W	**通道 0 数据下溢中断使能位：** 0：禁止通道 0 数据下溢中断 1：使能通道0数据下溢中断
Reserved -	Bit 3-2	—	**保留**
CH1	Bit 1	W	**通道1转换完成中断使能位：** 0：禁止通道1转换完成中断 1：使能通道1转换完成中断
CH0	Bit 0	W	**通道0转换完成中断使能位：** 0：禁止通道0转换完成中断 1：使能通道0转换完成中断

33. 5. 2. 6 DAC中断使能清除寄存器（DAC_IEC）

DAC 中断使能清除寄存器（DAC_IEC）

偏移地址：14_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										CH1UDF	CH0UDF	Reserved		CH1	CH0

Reserved	Bit 31-6	—	**保留**
CH1UDF	Bit 5	W	**通道 1 数据下溢中断清除使能位：** 0：无效 1：通道 1 数据下溢中断使能清除
CH0UDF	Bit 4	W	**通道 0 数据下溢中断清除使能位：** 0：无效 1：通道 0 数据下溢中断使能清除
Reserved	Bit 3-2	—	**保留**
CH1	Bit 1	W	**通道 1 转换完成中断清除使能位：** 0：无效 1：通道 1 转换完成中断使能清除
CH0	Bit 0	W	**通道 0 转换完成中断清除使能位：** 0：无效 1：通道 0 转换完成中断使能清除

33. 5. 2. 7 DAC中断使能有效寄存器（DAC_IEV）

DAC 中断使能有效寄存器（DAC_IEV）

偏移地址：18_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										CH1UDF	CH0UDF	Reserved		CH1	CH0

Reserved	Bit 31-6	—	**保留**
CH1UDF	Bit 5	R	**通道 1 数据下溢中断有效使能位：** 0：无效 1：通道 1 数据下溢中断使能有效
CH0UDF	Bit 4	R	**通道 0 数据下溢中断有效使能位：** 0：无效 1：通道0数据下溢中断使能有效
Reserved	Bit 3-2	—	**保留**
CH1	Bit 1	R	**通道1转换完成中断有效使能位：** 0：无效 1：通道 1 转换完成中断使能有效
CH0	Bit 0	R	**通道0转换完成中断有效使能位：** 0：无效 1：通道 0 转换完成中断使能有效

33. 5. 2. 8 DAC原始中断标志寄存器（DAC_RIF）

DAC 原始中断标志寄存器（DAC_RIF）

偏移地址：$1C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										CH1UDF	CH0UDF	Reserved		CH1	CH0

Reserved	Bit 31-6	—	**保留**
CH1UDF	Bit 5	R	**通道 1 数据下溢中断标志位：** 0：无效 1：发生通道 1 数据下溢
CH0UDF	Bit 4	R	**通道 0 数据下溢中断标志位：** 0：无效 1：发生通道 0 数据下溢
Reserved	Bit 3-2	—	**保留**
CH1	Bit 1	R	**通道 1 中断标志位：** 0：无效 1：通道 1 转换完成
CH0	Bit 0	R	**通道 0 中断标志位：** 0：无效 1：通道 0 转换完成

33. 5. 2. 9 DAC中断标志屏蔽寄存器（DAC_IFM）

DAC 中断标志屏蔽寄存器（DAC_IFM）

偏移地址：20_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										CH1UDF	CH0UDF	Reserved		CH1	CH0

Reserved	Bit 31-6	—	**保留**
CH1UDF	Bit 5	R	**屏蔽通道 1 数据下溢中断** 0：无效 1：屏蔽后的通道 1 数据下溢中断有效标志
CH0UDF	Bit 4	R	**屏蔽通道 0 数据下溢中断** 0：无效 1：屏蔽后的通道 0 数据下溢中断有效标志
Reserved	Bit 3-2	—	**保留**
CH1	Bit 1	R	**屏蔽通道 1 转换完成中断** 0：无效 1：屏蔽后的通道 1 转换完成中断有效标志
CH0	Bit 0	R	**屏蔽通道 0 转换完成中断** 0：无效 1：屏蔽后的通道 0 转换完成中断有效标志

33. 5. 2. 10 DAC中断标志清除寄存器（DAC_IFC）

DAC 中断标志清除寄存器（DAC_IFC）																															
偏移地址：24_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										CH1UDF	CH0UDF	Reserved		CH1	CH0

Reserved	Bit 31-6	—	**保留**
CH1UDF	Bit 5	W1	**软件写 1 清除通道 1 数据下溢中断** 0：无效 1：清除中断标志位
CH0UDF	Bit 4	W1	**软件写 1 清除通道 0 数据下溢中断** 0：无效 1：清除中断标志位
Reserved	Bit 3-2	—	**保留**
CH1	Bit 1	W1	**软件写 1 清除道 1 转换完成中断** 0：无效 1：清除中断标志位
CH0	Bit 0	W1	**软件写 1 清除道 0 转换完成中断** 0：无效 1：清除中断标志位

33. 5. 2. 11 DAC通道 0 输入数据寄存器（DAC_CH0DATA）

DAC 通道 0 输入数据寄存器（DAC_CH0DATA）																															
偏移地址：28_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				VAL											

Reserved	Bit 31-12	—	**保留**
VAL	Bit 11-0	R/W	**通道 0 输入数据**

33. 5. 2. 12 DAC通道 1 输入数据寄存器（DAC_CH1DATA）

DAC 通道 0 输入数据寄存器（DAC_CH0DATA）																															
偏移地址：$2C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				VAL											

Reserved	Bit 31-12	—	保留
VAL	Bit 11-0	R/W	通道 **1** 输入数据

33. 5. 2. 13 DAC组合输入数据寄存器（DAC_COMBDATA）

DAC 组合输入数据寄存器（DAC_COMBDATA）																															
偏移地址：30_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				CH1VAL												Reserved				CH0VAL											

Reserved	Bit 31-28	—	保留
CH1VAL	Bit 27-16	W	通道 **1** 输入数据
Reserved	Bit 15-12	—	保留
CH0VAL	Bit 11-0	W	通道 **0** 输入数据

33. 5. 2. 14 DAC校准寄存器（DAC_CAL）

DAC 校准寄存器（DAC_CAL）																															
偏移地址：34_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SELF_CALEN	Reserved																														

SELF_CALEN	Bit 31	R/W	**校准使能位：** 0：无效 1：校准使能（校准过程中自动清零）
Reserved	Bit 30-0	—	**保留**

第34章 模拟比较器（ACMP）

34. 1 概述

ACMP 模块可以将模拟电压信号进行比较，通过一个数字输出来指示哪一个模拟信号的电压更高，输入模拟信号可以从内部产生或者外部管脚输入，其工作消耗电流的大小或者响应时间可通过驱动电流的大小进行调节。

34. 2 特性

- 8 个可选的外部模拟信号输入到正端
- 8 个可选的外部模拟信号输入到负端
- 8 个可选的内部模拟信号输入到负端
- 可支持多种工作模式选择
- 可配置的迟滞选择，0 ~ 60mv 之间可选 8 个等级
- 可配置的响应时间
- 异步中断源可选为以下边沿触发
  - 上升沿
  - 下降沿
  - 上升下降沿
- 支持运行在芯片工作模式 RUN、SLEEP、STOP1、STOP2
- 还支持以下特性
  - 可调节的内部电阻
  - 可配置比较器反向输出
  - 可配置不活动时的比较器输出
  - 比较器输出直接给 PIS
  - 比较器输出可由 GPIO 输出，极性可选择

34. 3 结构框图

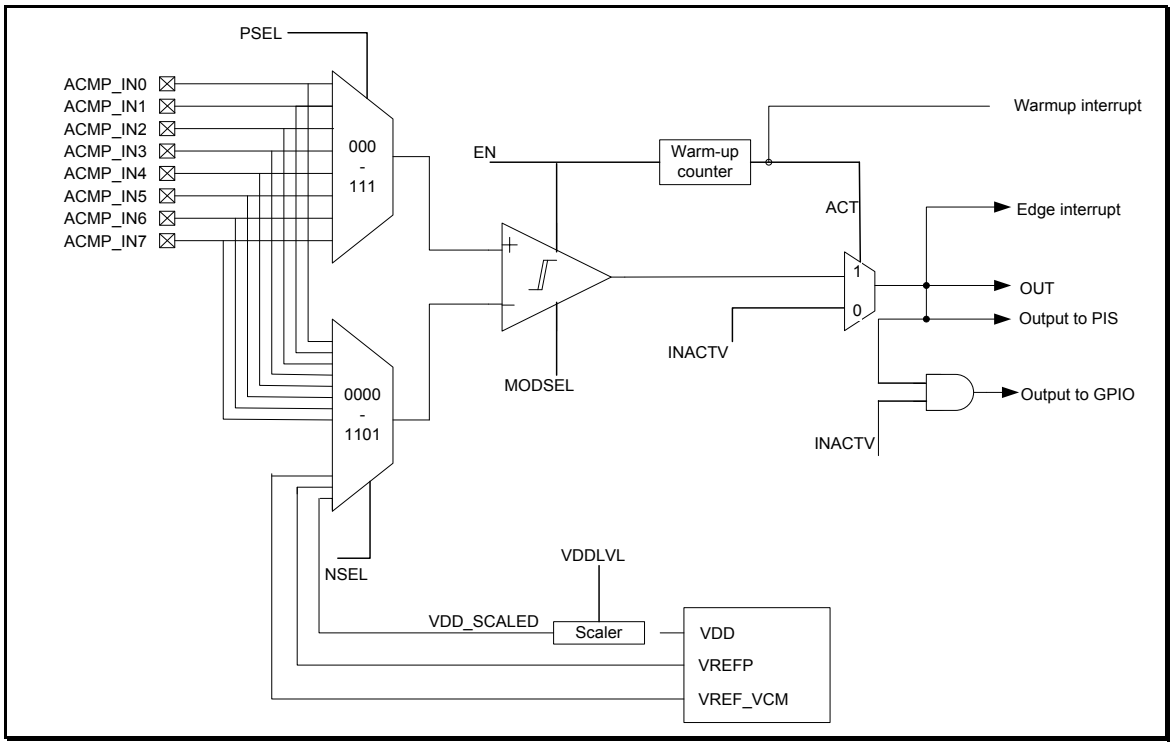


图 34-1 ACMP 结构框图

模拟比较器用来比较的是两个模拟输入量，一个正端，一个负端，通过比较器输出的数字信号来指示模拟输入量的电压高低情况，当输出为高时，表明正端输入电压高于负端输入电压，而当输出为低时，表明正端输入电压低于负端输入电压。

模拟比较器的输出数字信号保存在寄存器 OUT，该信号会随着两个模拟输入量的比较关系的变化而变化，除此之外其他对模拟比较器配置的改变要求在其不工作的情况下进行。

34.4 功能描述

34.4.1 ACMP控制

34.4.1.1 稳定时间

当寄存器 ACMP_CON.EN = 1 时，模拟比较器被使能，需要等待一段建立时间之后，才能正常工作，此时输出才有效，而这段建立时间称为稳定时间，该稳定时间可在寄存器 ACMP_CON.WARMUPT 位中配置在 1us ~ 50s 区间，以时钟计数值决定等待时间长短，当稳定时间结束后，寄存器 ACMP_STAT.ACT 位将置 1，表明模拟比较器进入有效状态开始正常工作。稳定时间期间的比较器处于无效状态，其输出信号配置在寄存器 ACMP_CON.INACTV 位。

若要在芯片工作模式 STOP1、STOP2 下使用模拟比较器，应该等到稳定时间之后（ACMP_STAT.ACT 为 1）才能配置进入对应模式，否则，将不能产生比较器中断；而要在芯片工作模式 SLEEP 下使用，稳定期间就可以配置进入。

34.4.1.2 响应

在模拟比较器正常工作时，当模拟输入量的比较极性变化时，需要等待一段时间之后，数字输出量才会对应变化，该时间称为响应时间，可以通过比较器工作模式选择位 MODE 调节 BIAS 的电流大小来改变响应时间，小的 BIAS 电流会降低工作功耗，但会带来较大的响应时间。

比较器工作模式	偏置电流（uA）	响应时间（us）
超低功耗模式	25	5.58
低功耗模式	125	1.17
普通模式	200	0.38
高速模式	500	0.11

表 34-1 响应时间与工作模式对应关系

当选择了高速模式时，应该选择适当的迟滞档位，以消除比较器输出信号的毛刺。

34.4.1.3 迟滞

通过配置寄存器 ACMP_CON.HYSTSEL 位，模拟比较器有 8 个可选的迟滞档位，包括档位 0 对应的迟滞是 0，当选为非 0 档位时，当正端和负端输入模拟信号电压的差值达到 ACMP_CON.HYSTSEL 选择的迟滞电压时，比较器输出信号才会对应变化，迟滞的作用是滤除那些在比较临界点的输出模拟信号的电压抖动变化，而只输出模拟信号压差大于迟滞范围的有效信号，所以迟滞的选取应该考虑输入量的抖动与范围。另外，迟滞的选取还会影响模拟比较器的工作功耗情况，一般是高的迟滞档位会降低工作的功耗。

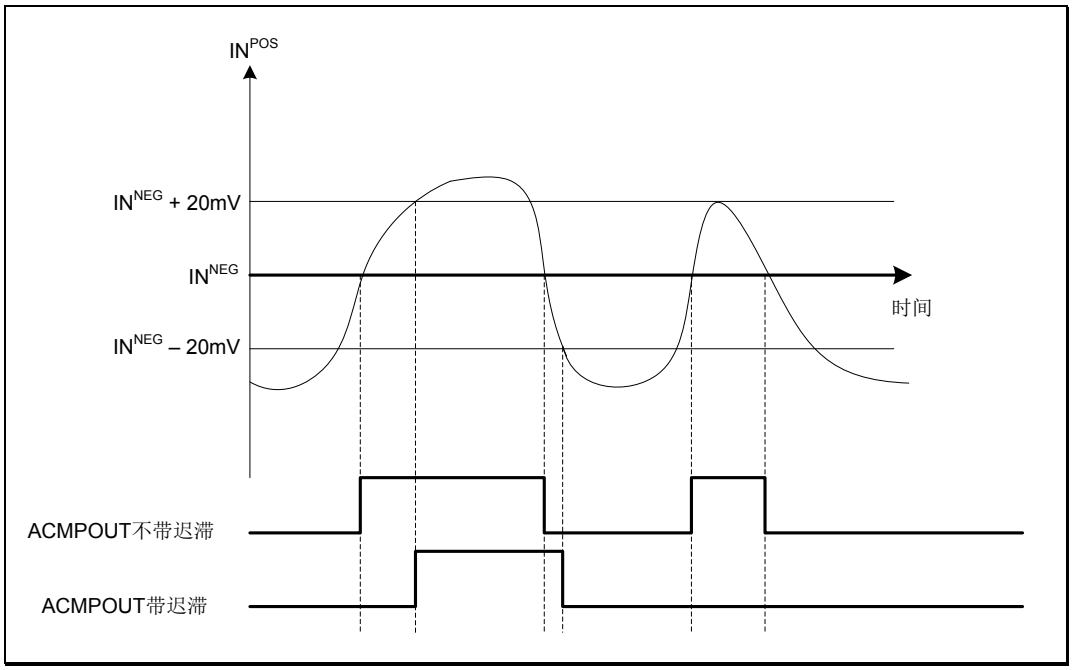


图 34-2 迟滞原理图

34. 4. 2 通道选择

模拟比较器有两个模拟输入量，正端输入可从 8 个外部输入中选择其一，负端输入可由 8 个外部输入、以及 VDD 分压和内部参考源等内部输入，其中 VDD 分压有 64 级分压，由 ACMP_INPUTSEL.VDDLVL 选择其电压值：

$$V_{DD\_SCALED} = V_{DD} * VDDLVL/63$$

34. 4. 3 数据管理

模拟比较器的输出数字信号保存在寄存器 ACMP_STAT.OUT，同时，当设置 ACMP_PORT.PEN = 1 时，作为 GPIO 可选输出功能之一，可以通过配置 ACMP_CON.OUTINV 来选择输出信号的极性。

34. 4. 4 中断与PIS触发

模拟比较器中断可选为边沿中断，包括上升沿或者下降沿，或双沿中断同时打开，寄存器 ACMP_RIF.EDGE 用于记录中断标志。当对应的 ACMP_IES.EDGE 中断使能被打开时，若此时产生边沿中断，则会产生中断请求，同时 ACMP_IFM.EDGE 会置 1，表明产生了中断请求，在系统响应了请求后，在对应 ACMP_IFC.EDGE 上置 1 会清掉 ACMP_IFM.EDGE 位。边沿中断可用于将系统从低功耗模式下唤醒，进入到正常工作模式。

当模拟比较器在开始的稳定工作后，也会产生一个稳定工作中断，并会被记录在 ACMP_RIF.WARMUP 中，当该中断在 ACMP_IES.WARMUP 被使能时，同样会产生中断请求。

模拟比较器的输出信号也连接到外设互联（PIS）系统中，作为其他外设的触发源。

34. 5 特殊功能寄存器

34. 5. 1 寄存器列表

名称	偏移地址	描述
ACMP_CON	000_H	ACMP 控制寄存器
ACMP_INPUTSEL	004_H	ACMP 输入选择寄存器
ACMP_STAT	008_H	ACMP 状态寄存器
ACMP_IES	$00C_H$	ACMP 中断使能设置寄存器
ACMP_IEC	010_H	ACMP 中断使能清除寄存器
ACMP_IEV	014_H	ACMP 中断使能有效寄存器
ACMP_RIF	018_H	ACMP 原始中断标志寄存器
ACMP_IFM	$01C_H$	ACMP 中断标志屏蔽寄存器
ACMP_IFC	020_H	ACMP 中断标志清除寄存器
ACMP_PORT	024_H	ACMP 端口寄存器

34. 5. 2 寄存器描述

34. 5. 2. 1 ACMP控制寄存器（ACMP _CON）

ACMP 控制寄存器（ACMP_CON）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														FALLEN	RISEEN	MODSEL		Reserved			WARMUPT			Reserved	HYSTSEL			OUTINV	INACTV	Reserved	EN

Reserved	Bit 31-18	—	**保留**
FALLEN	Bit 17	R/W	该位写 1，当比较输出出现下降沿时，可置起边沿中断标志。
RISEEN	Bit 16	R/W	该位写 1，当比较输出出现上升沿时，可置起边沿中断标志。
MODSEL	Bit 15-14	R/W	**模拟比较器运行模式选择** 00：超低功耗模式 01：低功耗模式 10：普通模式 11：高速模式
Reserved	Bit 13-11	—	**保留**
WARMUPT	Bit 10-8	R/W	**模拟比较器 Warm-up 时间设置** 000：4 PCLK2周期 001：8 PCLK2周期 010：16 PCLK2周期: 011：32 PCLK2周期 100：64 PCLK2周期 101：128 PCLK2周期 110：256 PCLK2周期 111：512 PCLK2 周期
Reserved	Bit 7	—	**保留**
HYSTSEL	Bit 6-4	R/W	**迟滞电平选择** 迟滞电平有可能会有变动，详情请参考电气特性 000：无迟滞 001：~15 mV迟滞 010：~22 mV迟滞 011：~29 mV迟滞 100：~36 mV迟滞 101：~43 mV迟滞 110：~50 mV迟滞 111：~57 mV 迟滞
OUTINV	Bit 3	R/W	**模拟比较器复用输出取反**

			0：比较输出不取反 1：比较输出取反
INACTV	Bit 2	R/W	**比较器的无效状态值** 0：无效值为0 1：无效值为 1
Reserved	Bit 1	—	**保留**
EN	Bit 0	R/W	**模拟比较器使能** 0：禁止 1：使能

34. 5. 2. 2 ACMP输入选择寄存器（ACMP_INPUTSEL）

ACMP 输入选择寄存器（ACMP_INPUTSEL）

偏移地址：04_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-14	—	**保留**
VDDLVL	Bit 13-8	R/W	VDD 参考电压分压比 VDD_SCALED = VDD*VDDLVL/63.
NSEL	Bit 7-4	R/W	**反向输入选择** 0000：通道 0 0001：通道 1 0010：通道 2 0011：通道 3 0100：通道 4 0101：通道 5 0110：通道 6 0111：通道 7 1000：内部参考电压 1V 1001：VREFP 1010：VDD 分压 1011：OP0_MAINOUT 1100：OP1_MAINOUT 1101：OP2_MAINOUT 1110：DAC 通道 0 输出 1111：DAC 通道 1 输出 注 1：对于不含 OPAMP 外设的产品系列，1011、1100、1101 为保留，具体某个产品是否含 OPAMP 外设可参考该产品的 Datasheet 注 2：对于不含 DAC 外设的产品系列，1110、1111 为保留，具体某个产品是否含 DAC 外设可参考该产品的 Datasheet
Reserved	Bit 3	—	**保留**
PSEL	Bit 2-0	R/W	**正向输入选择** 000：通道 0 001：通道 1 010：通道 2 011：通道 3 100：通道 4

			101：通道 5 110：通道 6 111：通道 7

34. 5. 2. 3 ACMP状态寄存器（ACMP_STAT）

ACMP 状态寄存器（ACMP_STAT）

偏移地址：08_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-2	—	保留
OUT	Bit 1	R	模拟比较器输出值
ACT	Bit 0	R	模拟比较器有效状态

34. 5. 2. 4 ACMP中断使能设置寄存器（ACMP_IES）

ACMP 中断使能设置寄存器（ACMP_IES）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-2	—	保留
WARMUP	Bit 1	W1	**Warm-up中断使能** 0：禁止 1：使能
EDGE	Bit 0	W1	**边沿触发中断使能** 0：禁止 1：使能

34. 5. 2. 5 ACMP中断使能清除寄存器（ACMP_IEC）

ACMP 中断使能清除寄存器（ACMP_IEC）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-2	—	保留
WARMUP	Bit 1	W1	**Warm-up中断使能清除** 0：无效 1：写1清除
EDGE	Bit 0	W1	**边沿触发中断使能清除** 0：无效 1：写1清除

34. 5. 2. 6 ACMP中断使能有效寄存器（ACMP_IEV）

ACMP 中断使能有效寄存器（ACMP_IEV）

偏移地址：14_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-2	—	保留
WARMUP	Bit 1	R	**Warm-up中断有效状态** 0：禁止 1：使能
EDGE	Bit 0	R	**边沿触发中断有效状态** 0：禁止 1：使能

34. 5. 2. 7 ACMP原始中断标志寄存器（ACMP_RIF）

ACMP 原始中断标志寄存器（ACMP_RIF）

偏移地址：18_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-2	—	保留
WARMUP	Bit 1	R	**原始Warm-up中断标志** 0：未发生 1：已发生
EDGE	Bit 0	R	**原始边沿触发中断标志** 0：未发生 1：已发生

34. 5. 2. 8 ACMP中断标志屏蔽寄存器（ACMP_IFM）

ACMP 中断标志屏蔽寄存器（ACMP_IFM）

偏移地址：$1C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-2	—	保留
WARMUP	Bit 1	R	**屏蔽Warm-up中断标志** 0：未发生 1：已发生
EDGE	Bit 0	R	**屏蔽边沿触发中断标志** 0：未发生 1：已发生

34. 5. 2. 9 ACMP中断标志清除寄存器（ACMP_IFC）

ACMP 中断标志清除寄存器（ACMP_IFC）

偏移地址：20_H

复位值：$00000000\_00000000\_00001111\_11111111_B$

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

Reserved	Bit 31-2	—	保留
WARMUP	Bit 1	W1	**Warm-up中断标志清除** 0：无效 1：写1清除
EDGE	Bit 0	W1	**边沿触发中断标志清除** 0：无效 1：写1清除

34. 5. 2. 10 ACMP端口寄存器（ACMP_PORT）

ACMP 端口寄存器（ACMP_PORT）

偏移地址：24_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-1	—	保留
PEN	Bit 0	R/W	**模拟比较器输出引脚使能** 0：禁止 1：使能

第35章 温度传感器模块（TSENSE）

35. 1 概述

温度传感器模块（TSENSE）是一个独立的温度传感器控制模块。TSENSE 可提供一个 16 位实时温度值，误差±1℃，最小分辨率约 0.004℃。

TSENSE 支持在低功耗模式下运行，同时可以给 RTC 模块提供自动温度补偿所需的实时温度值。

35. 2 特性

- 仅上电复位有效，支持寄存器写保护，有效避免软件误操作
- 时钟源支持 LOSC、LRC、HOSC、HRC
- 温度计量范围-40~85℃，计量误差±1℃，最小分辨率约 0.004℃
- 可配合 RTC 温补算法，为实时时钟校准提供高精度实时温度
- 低功耗设计：在 VDD 掉电时能保证温度检测精度

35. 3 功能描述

35. 3. 1 时钟频率

TSENSE 时钟源通过备份域外设时钟控制寄存器 BKPC_PCCR.TSENSECS 进行选择，可选择为：

- 32768Hz 低速时钟 LRC
- 32768Hz 低速时钟 LOSC（停振自动切换至 LRC）
- 高速时钟 HRC 分频至 1MHz
- 高速时钟 HOSC 分频至 1MHz

TSENSE 支持工作时钟分频，公式为：

$$F_{DIVCLK} = \frac{F_{TEMPCLK}}{PRS + 1}$$

注：工作时钟频率的选择，需基于速度和功耗之间的取舍，一般推荐使用的频率约为 16KHz。

35. 3. 2 温度输出

温度输出的时间通过建立时间和输出模式共同决定。温度刷新时钟频率公式为：

$$F_{TEMPO} = \frac{F_{DIVCLK}}{T_{OM}}$$

温度输出时间公式为：

$$t_{READY} = \frac{F_{TEMPO}}{T_{SU}}$$

输出时间计算示例如下表所示。

TSENSECLK	PRS[7:0]	DIVCLK	T_{OM}	T_{SU}	t_{READY}
32768Hz	1	16384Hz	200	7	85.5ms
32768Hz	1	16384Hz	400	6	146.5ms
32768Hz	1	16384Hz	800	5	244.1ms
32768Hz	1	16384Hz	1600	4	390.6ms
32768Hz	1	16384Hz	3200	3	585.9ms
1MHz	60	16393Hz	200	7	85.4ms
1MHz	60	16393Hz	400	6	146.4ms
1MHz	60	16393Hz	800	5	244.0ms
1MHz	60	16393Hz	1600	4	290.4ms
1MHz	60	16393Hz	3200	3	585.6ms

表 35-1 输出时间计算示例表

在 TSENSE 模块使能后，并经过 t_{READY} 时间，可从温度值寄存器 TSENSE_DR 中读到实时的温度数据。当读取温度时，确保当前温度输出值无错误，可从 TSENSE_DR.ERR 读取错误状态。

当温度计算完成后，TSENSE 使能位 TSENSE_CR.EN 被硬件自动清零。下一次需要获取温度时将其使能可再次获得实时温度。

当使能连续模式时，温度计算完成不会将 TSENSE 使能位 TSENSE_CR.EN 清零，硬件将会不停刷新实时温度，刷新频率为 $F_{TSENSEO}$。直到软件将 TSENSE 使能位 TSENSE_CR.EN 清零后停止刷新。

当 TSENSE 被使能后，不可改变工作时钟分频、建立时间和温度输出模式选择，否则会造成温度检测异常。

35. 3. 3 与RTC模块配合进行自动温度补偿

TSENSE 支持在低功耗模式下，被 RTC 模块自动调用进行实时温度检测。使能外部温度获取使能位 TSENSE_CR.REQEN，并关闭 TSENSE 使能位 TSENSE_CR.EN，在 RTC 硬件触发温度补偿操作时，将自动激活 TSENSE 模块进行实时温度的检测，在检测完成后将自动关闭。

35. 3. 4 寄存器写保护

为避免程序的异常运行对 TSENSE 的误操作，TSENSE 写保护寄存器 TSENSE_WPR 用于阻止程序对 TSENSE 其它寄存器的误写入。该寄存器保护范围为除 TSENSE_WPR 寄存器外的 TSENSE 模块所有寄存器。

TSENSE_WPR 寄存器为虚拟寄存器。要对 TSENSE 其它寄存器进行写操作时，需先对 TSENSE_WPR 寄存器写 0xA55A9669，之后可对 TSENSE 其它寄存器进行写操作。对 TSENSE_WPR 寄存器写入其他值重新进入写保护状态，写保护状态下对 TSENSE 寄存器进行的写操作将被忽略。

可以通过读取 TSENSE_WPR 寄存器确认 TSENSE 是否处于写保护状态，读出值为 0x00000000，表示当前可对 TSENSE 寄存器进行写操作；读出值为 0x00000001 表示 TSENSE 处于写保护状态。TSENSE_WPR 寄存器无其它读出值。

35. 3. 5 配置流程

1. 解除寄存器写保护
2. 配置预分频寄存器，使分频后时钟约为 16KHz
3. 配置低温增益和高温增益寄存器，配置温度边界寄存器，配置温感标定边界寄存器，配置值已在芯片出厂前存入 Flash 的信息区。
4. 配置建立时间选择位 TSENSE_CR.TSU 和温度输出模式选择位 TSENSE_CR.TOM
5. 配置是否使能连续模式，并使能 TSENSE
6. 等待温度检测中断，读取温度值

35. 3. 6 TSENSE中断源

TSENSE 模块共有 1 个中断源，即：

◇ 温度更新中断

每个中断源都有独立的使能位，使能位影响该中断是否产生 IRQ 中断请求，而不影响中断功能。即关闭相应中断使能，标志位仍可用于相应功能查询。当有多个中断使能时，各中断经过“或”逻辑产生 IRQ 中断请求。即任何一个被使能的中断产生中断事件时，均产生 IRQ 中断请求，且只有将所有的产生中断事件的中断标志清零后，IRQ 中断请求才解除。

35. 4 特殊功能寄存器

35. 4. 1 寄存器列表

TSENSE 寄存器列表		
名称	偏移地址	描述
TSENSE_WPR	000_H	TSENSE 写保护寄存器
TSENSE_CR	004_H	TSENSE 控制寄存器
TSENSE_DR	008_H	TSENSE 温度值寄存器
TSENSE_PSR	$00C_H$	TSENSE 预分频寄存器
TSENSE_IE	010_H	TSENSE 中断使能寄存器
TSENSE_IF	014_H	TSENSE 中断标志寄存器
TSENSE_IFCR	018_H	TSENSE 中断标志清零寄存器
TSENSE_LTGR	$01C_H$	TSENSE 低温增益系数寄存器
TSENSE_HTGR	020_H	TSENSE 高温增益系数寄存器
TSENSE_TBDR	024_H	TSENSE 温度边界寄存器
TSENSE_TCALBDR	028_H	TSENSE 温感标定边界寄存器
TSENSE_SR	$02C_H$	TSENSE 状态寄存器

35. 4. 2 寄存器描述

35. 4. 2. 1 TSENSE写保护寄存器（TSENSE_WPR）

TSENSE 写保护寄存器（TSENSE_WPR）

偏移地址：000_H

复位值：$00000000\_00000000\_00000000\_00000001_B$

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

Reserved	Bit 31-1	—	保留
WP	Bit 0	R	**写保护状态位** 0：写保护解除 1：写保护有效

注：对该寄存器写入 0xA55A9669 解除写保护，写入其他值开启写保护。该寄存器保护除自身外的 TSENSE 所有区域。

35. 4. 2. 2 TSENSE控制寄存器（TSENSE_CR）

TSENSE 控制寄存器（TSENSE_CR）

偏移地址：004_H

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

系统复位：不受影响

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

Reserved	Bit 31-15	—	**保留**
TSU	Bit 14-12	R/W	**建立时间选择位** 0xx：3 个温度更新周期 100：4 个温度更新周期 101：5 个温度更新周期 110：6 个温度更新周期 111：7 个温度更新周期 注：温度更新周期由 TOM 位配置
Reserved	Bit 11	—	**保留**
TOM	Bit 10-8	R/W	**温度输出模式选择位** 000：每 200 个 CLKDIV 时钟更新一次温度 001：每 400 个 CLKDIV 时钟更新一次温度 010：每 800 个 CLKDIV 时钟更新一次温度 011：每 1600 个 CLKDIV 时钟更新一次温度 1xx：每 3200 个 CLKDIV 时钟更新一次温度
Reserved	Bit 7-5	—	**保留**
CTN	Bit 4	R/W	**连续模式使能位** 0：禁止 1：使能
RST	Bit 3	W	**TSENSE 复位位** 0：无操作 1：复位 注：仅在 CTN 为 1 时有效
ENS	Bit 2	R	**TSENSE 使能状态位** 0：禁止 1：使能
REQEN	Bit 1	R/W	**外部温度获取使能位** 0：禁止 1：使能
EN	Bit 0	R/W	**TSENSE 使能位** 0：禁止 1：使能

			注：当 CTN 为 0 时，温度获取成功后硬件自动将该位清 0

35. 4. 2. 3 TSENSE温度值寄存器（TSENSE_DR）

TSENSE 温度值寄存器（TSENSE_DR）

偏移地址：008_H

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

系统复位：不受影响

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

ERR	Bit 31	R	**错误状态位** 0：无错误 1：错误 注：该位为1时表示硬件发生错误，此次输出的温度值为不可信
Reserved	Bit 30-16	—	**保留**
DATA	Bit 15-0	R	**温度值** 该位表示为 16 位补码格式，其中 1 位符号位（Bit15），7 位整数位（Bit14~8），8 位小数位（Bit7~0）

35. 4. 2. 4 TSENSE预分频寄存器（TSENSE_PSR）

TSENSE 预分频寄存器（TSENSE_PSR）

偏移地址：$00C_H$

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

系统复位：不受影响

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

Reserved	Bit 31-8	—	保留
PRS	Bit 7-0	R/W	预分频选择位 0x00：1:1 0x01：1:2 0x02：1:3 …… 0xFF：1:256

35. 4. 2. 5 TSENSE中断使能寄存器（TSENSE_IE）

TSENSE 中断使能寄存器（TSENSE_IE）

偏移地址：010_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-1	—	保留
TSENSE	Bit 0	R/W	温度更新中断使能位 0：禁止 1：使能

35. 4. 2. 6 TSENSE中断标志寄存器（TSENSE_IF）

TSENSE 中断标志寄存器（TSENSE_IF）

偏移地址：014_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-1	—	保留
TSENSE	Bit 0	R	**温度更新中断标志位** 0：温度未更新 1：温度已更新 该位只读，由操作 TSENSE_IFCR 清零

35. 4. 2. 7 TSENSE中断标志清零寄存器（TSENSE_IFCR）

TSENSE 中断标志清零寄存器（TSENSE_IFCR）

偏移地址：018_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-1	—	保留
TSENSE	Bit 0	R/W	**温度更新中断标志位清零** 0：无操作 1：温度更新中断标志位清零

35. 4. 2. 8 TSENSE低温增益寄存器（TSENSE_LTGR）

TSENSE 低温增益寄存器（TSENSE_LTGR）

偏移地址：$01C_H$

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

系统复位：不受影响

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

Reserved	Bit 31-21	—	保留
LTG	Bit 20-0	R/W	低温增益值

35. 4. 2. 9 TSENSE高温增益寄存器（TSENSE_HTGR）

TSENSE 高温增益寄存器（TSENSE_HTGR）

偏移地址：020_H

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

系统复位：不受影响

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

Reserved	Bit 31-21	—	保留
HTG	Bit 20-0	R/W	高温增益值

35. 4. 2. 10 TSENSE温度边界寄存器（TSENSE_TBDR）

TSENSE 温度边界寄存器（TSENSE_TBDR）

偏移地址：024_H

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

系统复位：不受影响

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

Reserved	Bit 31-16	—	保留
TBD	Bit 15-0	R/W	**边界温度值** 该位表示为 16 位补码格式，其中包含 1 位符号位（Bit15），7 为整数位（Bit14~8）和 8 位小数位（Bit7~0）

35. 4. 2. 11 TSENSE温感标定边界寄存器（TSENSE_TCALBDR）

TSENSE 温感标定边界寄存器（TSENSE_TCALBDR）

偏移地址：028_H

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

系统复位：不受影响

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

Reserved	Bit 31-17	—	保留
TCAL	Bit 16-0	R/W	**温感标定边界值** 该位表示为 17 位补码格式，其中包含 1 位符号位和 16 位整数位

35. 4. 2. 12 TSENSE状态寄存器（TSENSE_SR）

TSENSE 状态寄存器（TSENSE_SR）

偏移地址：$02C_H$

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

系统复位：不受影响

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

TSOUT	Bit 31	R	**温感输出值**
Reserved	Bit 30-26	—	**保留**
NVLD	Bit 25	R	**有效状态位** 0：有效 1：无效 注：当该位为0时，表示TCAL中的值为有效的当前温度值，软件可在该位为0时的任意时刻读取TCAL值，若该位值未发生改变，则TCAL值也不会改变
TCAL	Bit 24-0	R	**温感计算提取值** 该位表示为 25 位补码格式，其中 1 位符号位，24 位整数位

第36章 调试控制（DBGC）

36. 1 概述

ES32F36xx 系列使用的 Cortex™-M3 内核，该内核支持高级调试功能。利用这些调试功能，可以在取指（指令断点）或取访问数据（数据断点）时停止内核，查询内核的内部状态和系统的外部状态，查询完成后，可以恢复内核和系统运行。

当调试器与 MCU 相连并进行调试时，将使用内核的硬件调试模块。

ES32F36xx 系列 MCU 提供 SWD（Serial Wire Debug）调试接口。

参考文档：

ARM_debug_interface_v5.pdf（ADIv5 Architecture specification）

ARM_debug_interface_v5_supplement.pdf（ADIv5.1 Spec Supplement）

DDI0419C_arm_architecture_v6m_reference_manual.pdf（ARMv6 Architecture）

36. 2 特性

- 支持 SW-DP：调试端口电路，实现 DAP 电路和外部调试主机的通讯
- 支持 MEM-AP：访问端口电路，实现 DAP 电路与被调试单元的通讯
- 支持断点（Breakpoint）：4 个断点
- 支持数据观测和追踪（DWT）：2 个数据观测点

36. 3 结构框图

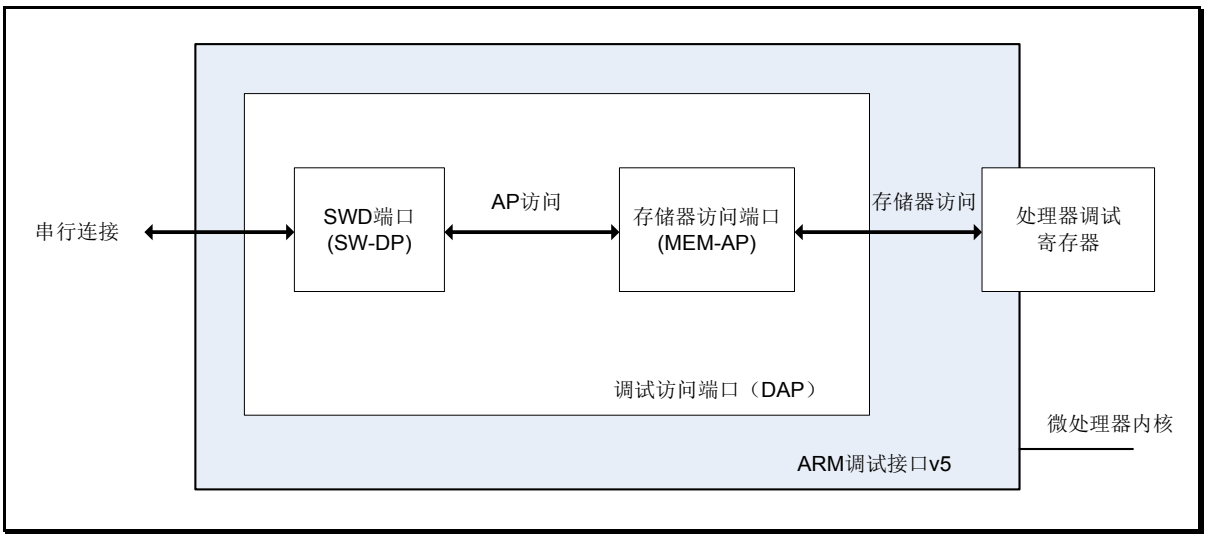


图 36-1 SWD 调试结构图

36.4 功能描述

36.4.1 调试端口

下表为 SWD 调试用端口，芯片上电后默认作为调试功能使用。

端口功能	输入/输出	说明
SWCLK	输入	调试时钟； 该端口在调试模式下为调试电路提供通信时钟，默认内部下拉。
SWDIO	输入/输出	调试数据输入/输出端口； 用于 SW-DP 与外部调试主机的数据交互，默认内部上拉。

表 36-1 SWD 端口描述

36.4.2 调试冻结

程序开发过程中，会经历反复调试，通过调试工具对运行程序进行暂停，实际上在暂停内核的时候，仍有部分硬件外设在工作（如：定时器、RTC、看门狗等），影响调试效果；DBGC 模块可以实现内核和外设同时暂停，实现高级调试功能。

操作示例：

1. 按照需求正确初始化使用的外设，如定时器 AD16C4T1
2. 设置 DBG_APB1FZ.AD16C4T1_STOP = 1，可实现调试过程中，内核与 AD16C4T1 同时暂停；
3. 如果未设置 DBG_APB1FZ.AD16C4T1_STOP = 1，程序在调试过程中暂停时，实际 AD16C4T1 仍然在计数。

36.4.3 调试复位

内核调试电路和调试控制寄存器只可被上电复位、欠压复位及软件复位中的芯片全局复位（RMU_AHB2RSTR.CHIPRST）所复位。

36. 4. 4 MEM-AP访问端口

MEM-AP 端口，用于访问被调试单元的存储器映射区域。

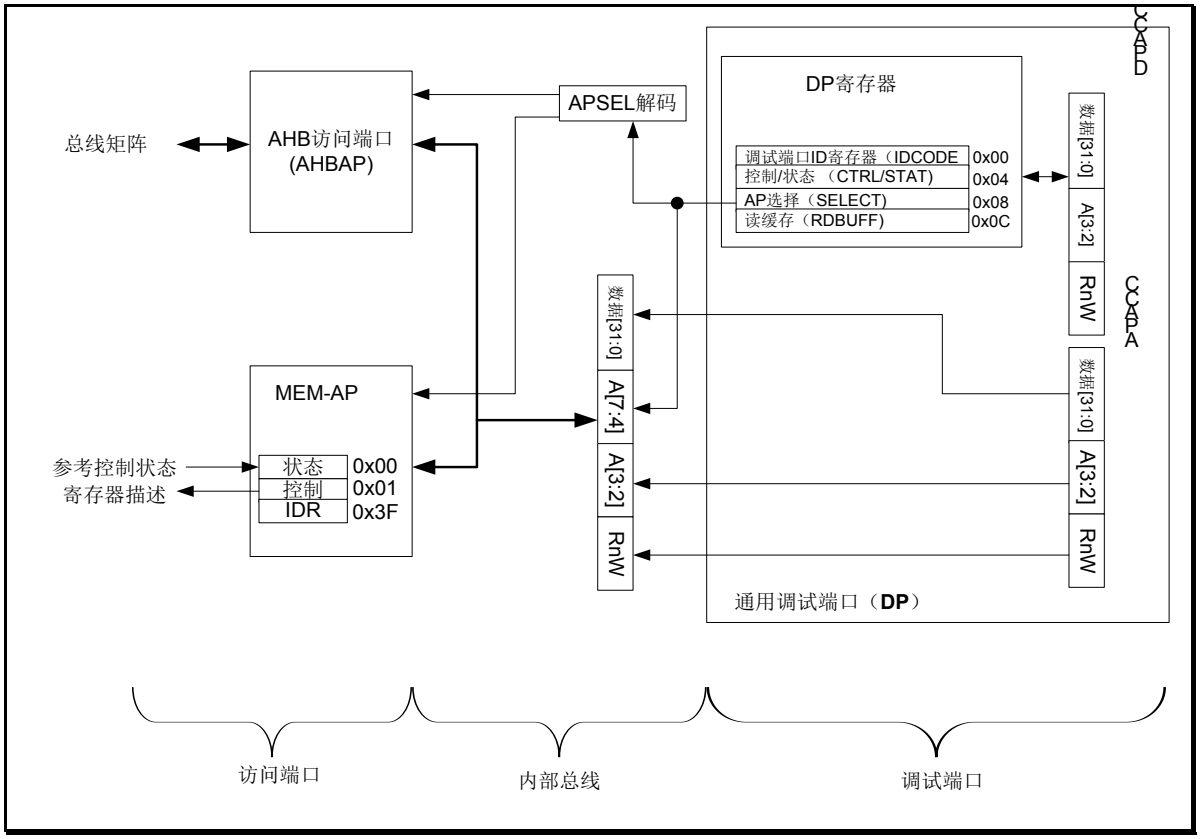


图 36-2 MEM-AP 地址映射

36. 5 特殊功能寄存器

36. 5. 1 寄存器列表

DBGC 寄存器列表		
名称	偏移地址	描述
DBG_IDCODE	000_H	DBG 器件识别码寄存器
Reserved	004_H	保留
DBG_APB1FZ	008_H	APB1 外设调试冻结寄存器
DBG_APB2FZ	$00C_H$	APB2 外设调试冻结寄存器

36. 5. 2 寄存器描述

36. 5. 2. 1 DBG器件识别码寄存器（DBG_IDCODE）

DBG 器件识别码寄存器（DBG_IDCODE）																															
偏移地址：000_H																															
上电复位值：xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxxxx $_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV_ID																CORE_ID				DEV_ID											

REV_ID	Bit 31-16	R	**版本识别码** 0x1000：版本 A 0x1001：版本 B
CORE_ID	Bit 15-12	R	**内核识别码** 0x3：Cortex-M3
DEV_ID	Bit 11-0	R	**器件识别码** 0x031：MCU 识别码

36. 5. 2. 2 APB1 外设调试冻结寄存器（DBG_APB1FZ）

APB1 外设调试冻结寄存器（DBG_APB1FZ）

偏移地址：008_H

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CAN_STOP	Reserved		I2C1_SMBUS_TO	I2C0_SMBUS_TO	GP16C4T1_STOP	GP16C4T0_STOP	BS16T1_STOP	BS16T0_STOP	GP32C4T1_STOP	GP32C4T0_STOP	AD16C4T1_STOP	AD16C4T0_STOP

Reserved	Bit 31-13	—	保留
CAN_STOP	Bit 12	R/W	**CAN 调试暂停选择位** 0：内核停止时，仍正常工作 1：内核停止时，暂停接收
Reserved	Bit 11-10	—	保留
I2C1_SMBUS_TO	Bit 9	R/W	**I2C1 SMBUS 超时定时器调试暂停选择位** 0：内核停止时，超时定时器仍正常工作 1：内核停止时，超时定时器暂停
I2C0_SMBUS_TO	Bit 8	R/W	**I2C0 SMBUS 超时定时器调试暂停选择位** 0：内核停止时，超时定时器仍正常工作 1：内核停止时，超时定时器暂停
GP16C4T1_STOP	Bit 7	R/W	**GP16C4T1 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数
GP16C4T0_STOP	Bit 6	R/W	**GP16C4T0 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数
BS16T1_STOP	Bit 5	R/W	**BS16T1 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数
BS16T0_STOP	Bit 4	R/W	**BS16T0 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数
GP32C4T1_STOP	Bit 3	R/W	**GP32C4T1 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数
GP32C4T0_STOP	Bit 2	R/W	**GP32C4T0 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数
AD16C4T1_STOP	Bit 1	R/W	**AD16C4T1 调试暂停选择位** 0：内核停止时，仍正常计数

			1：内核停止时，暂停计数
AD16C4T0_STOP	Bit 0	R/W	**AD16C4T0 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数

注：该寄存器仅支持按字写入。

36. 5. 2. 3 APB2 外设调试冻结寄存器（DBG_APB2FZ）

APB2 外设调试冻结寄存器（DBG_APB2FZ）

偏移地址：$00C_H$

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-11	—	**保留**
RTC_STOP	Bit 10	R/W	**RTC 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数
WWDT_STOP	Bit 9	R/W	**WWDT 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数
IWDT_STOP	Bit 8	R/W	**IWDT 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数
Reserved	Bit 7-0	—	**保留**

注：该寄存器仅支持按字写入。

第37章 Flash信息区

37. 1 概述

Flash 信息区用来存储芯片的只读信息和配置信息。

用户在程序中只能对只读信息和配置信息进行读操作。可以使用芯片烧录工具对配置信息进行修改，但无法改变只读信息。

37. 2 特性

- Flash 信息区存储的只读信息包括：
  - 产品识别码
  - 芯片唯一码
- Flash 信息区存储的配置信息包括
  - 芯片配置字
  - 写保护区域配置字
  - 数据区配置字
  - 全局读保护配置字
  - 私有代码读出保护区域配置字
  - 用户程序校验码

37. 3 功能描述

37. 3. 1 Flash信息区只读信息

Flash 信息区的基地址为 0x0008_0000，可以字读取。

37. 3. 1. 1 芯片唯一码（UID）

芯片唯一码 UID 为 96 位，每一颗芯片都是唯一的编码，可以用做：

◇ 终端产品序列号

◇ 通过特定的加密算法生成安全密钥

寄存器名称	芯片唯一码 0（UID0）	
地址偏移	$03E0_H$	
UID0	Bit 31-0	芯片唯一码 0

寄存器名称	芯片唯一码 1（UID1）	
地址偏移	$03E8_H$	
UID1	Bit 31-0	芯片唯一码 1

寄存器名称	芯片唯一码 2（UID2）	
地址偏移	$03EC_H$	
UID2	Bit 31-0	芯片唯一码 2

37. 3. 1. 2 芯片产品识别码（CHIPID）

CHIPID 用来区分芯片产品型号，为 32 位编码。

寄存器名称	芯片产品识别码（CHIPID）	
地址偏移	$03F8_H$	
CHIPID	Bit 31-0	CHIPID

37. 3. 2 Flash信息区配置信息

Flash 信息区的配置信息在芯片程序运行前生效。用户程序不能对其进行修改，但是有部分配置位是复位后加载到寄存器中生效的。用户可以修改这些寄存器再软件触发相应的复位来临时改变其配置，详细内容请参考“复位管理单元（RMU）”中对各种复位源和寄存器关系的描述。

Flash 信息区可以字读取。只能使用芯片烧录工具对配置信息进行修改。

37. 3. 2. 1 芯片配置字（CFG_WORD）

芯片的很多特性需要配置，而且这些配置需要在程序运行前生效，这就需要芯片配置字。

寄存器名称	芯片配置字 0（CFG_WORD0）
地址偏移	0400_H
低 16 位复位值	$0000\_1111\_0000\_0001_B$

寄存器名称	芯片配置字 0（CFG_WORD0）	
—	Bit 63-32	**保留未用**
—	Bit 31-16	**Bit 15-0 取反值（不满足取反时 Bit 15-0 强制为默认值）**
HRCSFS	Bit 15	HRC 启动频率选择： 0：24MHz 1：2MHz
IWDTEN	Bit 14	**IWDT 使能位** 0：由软件使能 1：硬件强制使能 注：硬件强制使能后，软件无法关闭；中断强制使能，软件无法关闭；复位强制使能，软件无法关闭；时钟源固定为 LRC，软件无法切换。
WWDTEN	Bit 13	**WWDT 使能位** 0：软件使能后可关闭 1：软件使能后无法关闭
BOOT	Bit 12	**Flash 启动地址选择位** 0：0x0000_0000 1：0x0007_E000（512KB Flash）或 0x0005_E000（384KB Flash）或 0x0003_E000（256KB Flash）或 0x0001_E000（128KB Flash） 0x0000_E000（64KB Flash）
BORVS	Bit 11-10	**BOR 电压点选择位** 00：3.8V 01：2.6V 10：2.0V 11：由软件控制
PWRTEN	Bit 9	**上电延时使能位** 0：禁止 1：使能
XTALS	Bit 8	**外部高速振荡器模式选择位** 0：1~8MHz 1：8~24MHz
XTFLTEB	Bit 7	**外部高速振荡器滤波禁止位** 0：使能 1：禁止
—	Bit 6-2	—
LOSMEN	Bit 1	**LOSC 安全管理使能位** 0：由软件使能或禁止 1：强制使能 注：硬件强制使能后，软件无法关闭
LOSCEN	Bit 0	**LOSC 使能位** 0：由软件使能或禁止

寄存器名称	芯片配置字 0（CFG_WORD0）	
		1：硬件强制使能 注：硬件强制使能后，软件无法关闭

注 1：复位值是指芯片配置字被从信息区读出之前的值。

37. 3. 2. 2 写保护区域配置字（CFG_WRP）

芯片支持 2 个保护区域，分别通过 CFG_WRP0 和 CFG_WRP1 来配置。设置为写保护的区域，用户程序将不能通过 IAP 对其进行擦写。

寄存器名称	写保护区域 x 配置字（CFG_WRPx）（x=0..1）	
地址偏移	0420_H ~0428_H	
低 16 位复位值	$0000\_0000\_0000\_0001_B$	
—	Bit 63-32	**保留未用**
—	Bit 31-16	**Bit 15-0 取反值（不满足取反时 Bit 15-0 强制为默认值）**
END	Bit 15-8	**保护结束页配置位** 0x0：Flash Page 1（默认） 0x1：Flash Page 3 0x2：Flash Page 5 …… 0xFF：Flash Page 511
START	Bit 7-1	**保护起始页配置位** 0x0：Flash Page 0（默认） 0x1：Flash Page 4 0x2：Flash Page 8 …… 0x7F：Flash Page 508
ENB	Bit 0	**保护禁止位** 0：使能 1：禁止（默认）

37. 3. 2. 3 数据区配置字（CFG_DAFLS）

芯片支持 1 个数据区域，通过 CFG_DAFLS 来配置。通过其配置可以将 Flash 空间分为程序区和数据区。程序区和数据区的 IAP 擦写命令不同。

寄存器名称	数据 Flash 配置字（CFG_DAFLS）	
地址偏移	0430_H	
低 16 位复位值	$0000\_0000\_0000\_0001_B$	
—	Bit 63-32	**保留未用**
—	Bit 31-16	**Bit 15-0 取反值（不满足取反时 Bit 15-0 强制为默认值）**
END	Bit 15-8	**数据 Flash 结束页配置位** 0x0：Flash Page 1（默认） 0x1：Flash Page 3

寄存器名称	数据 Flash 配置字（CFG_DAFLS）	
		0x2：Flash Page 5 …… 0xFF：Flash Page 511
START	Bit 7-1	**数据 Flash 起始页配置位** 0x0：Flash Page 0（默认） 0x1：Flash Page 4 0x2：Flash Page 8 …… 0x7F：Flash Page 508
ENB	Bit 0	**数据 Flash 禁止位** 0：使能 1：禁止（默认）

37. 3. 2. 4 用户程序校验码（CHKSUM）

烧录工具将用户程序校验码写入此区域。

寄存器名称	用户程序校验码（CHKSUM）	
地址偏移	$07A0_H$	
—	Bit 63-32	保留未用
CHKSUM	Bit 31-0	用户程序校验码

寄存器名称	用户程序校验码反码（CHKSUMN）	
地址偏移	$07A8_H$	
—	Bit 63-32	保留未用
CHKSUMN	Bit 31-0	用户程序校验码反码

37. 3. 2. 5 全局读保护配置字（CFG_GBRDP）

全局读保护分为 Level0~2 三个等级，Level0 为不保护，Level1 和 Level2 的详细说明请参考“存储器系统控制（MSC）”中“Flash 全局读保护”章节的描述。

寄存器名称	全局读保护配置字（CFG_GBRDP）	
地址偏移	$07F8_H$	
低 16 位复位值	$0000\_0000\_0000\_0000_B$	
—	Bit 63-32	保留未用
GBRDP	Bit 31-0	**全局读保护配置位** 0xFFFF_FFFF：读保护等级 Level 0 0xFFFF_XXXX：读保护等级 Level 1（XXXX 不为 FFFF） 0xYYYY_XXXX：读保护等级为 Level 2（YYYY 不为 FFFF）（默认）

37. 3. 2. 6 私有代码读出保护区域配置字（CFG_PCROP）

芯片支持 2 个私有代码读出保护区域，分别通过 CFG_PCROP0 和 CFG_PCROP1 来配

置。被设置为私有代码读出保护区域后，其中的代码可以被运行但无法读出。详细参考“存储器系统控制（MSC）”中“Flash 私有代码读出保护区域”章节的描述。

寄存器名称	私有代码读出保护区域 x 配置字（CFG_PCROPx）（x=0..1）	
地址偏移	0810_H ~0818_H	
低 16 位复位值	$1111\_1111\_0000\_0000_B$	
—	Bit 63-32	**保留未用**
—	Bit 31-16	**Bit 15-0 取反值（不满足取反时 Bit 15-0 强制为默认值）**
END	Bit 15-8	**保护结束页配置位** 0x0：Flash Page 1 0x1：Flash Page 3 0x2：Flash Page 5 …… 0xFF：Flash Page 511（默认）
START	Bit 7-1	**保护起始页配置位** 0x0：Flash Page 0（默认） 0x1：Flash Page 4 0x2：Flash Page 8 …… 0x7F：Flash Page 508
ENB	Bit 0	**保护禁止位** 0：使能（默认） 1：禁止

附录1 ARM Cortex-M3 参考资料

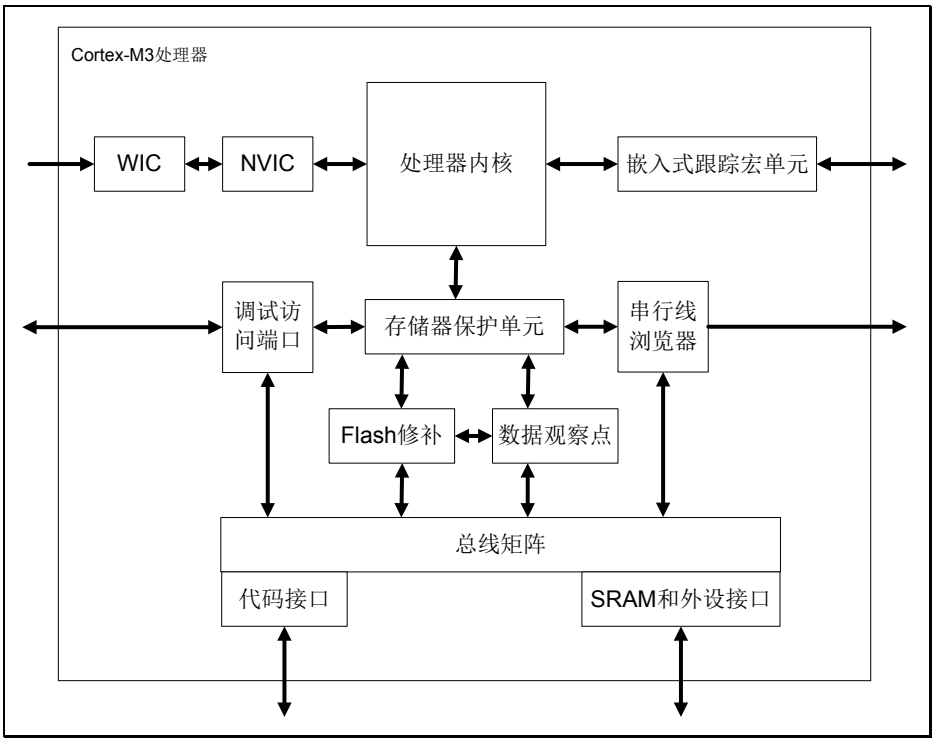
附录1. 1 ARM Cortex-M3 用户指南：简介

ARM Limited 提供本附录中的资料，并包括在含 Cortex-M3 CPU 的设备的用户手册中。为了反映只针对 ES32 微控制器的实现选项以及其他的差异，内容有轻微的改动。

附录1. 1. 1 关于处理器和内核外设

Cortex-M3 处理器是一种为微控制器市场设计的高性能 32 位处理器。它为开发人员提供了显著的益处，包括：

- ◇ 杰出的处理性能，以及快速的中断处理能力；
- ◇ 大量断点和跟踪能力，增强了系统调试能力；
- ◇ 高效的处理器内核、系统和存储器；
- ◇ 集成睡眠模式，具有超低功耗；
- ◇ 有可选的集成存储器保护单元（MPU），保证平台的安全性。



附录图 1-1 Cortex-M3 的典型应用

Cortex-M3 处理器建立在一个高性能处理器内核上，采用一个 3 级流水线哈佛架构，非常适合于高要求的嵌入式应用。该处理器使用了一个高效指令集和广泛优化的设计，具备了优异的能效，提供高端的处理硬件，包括单周期 32x32 乘法器和专用的硬件除法器。

为了降低设备的设计成本，Cortex-M3 处理器实现了紧密耦合的系统部件，在降低处理器面积的同时，还显著提高了中断处理和系统调试能力。Cortex-M3 处理器实现了一个 Thumb 指令集版本，从而确保了高代码密度，降低了对程序存储器的要求。Cortex-M3 指令集具有 8 位和 16 位微控制器的高代码密度，提供了现代 32 位架构理想的优异性能。

Cortex-M3 处理器紧密地集成了一个可配置的可嵌套向量中断控制器（NVIC），提供了业界领先的中断性能。 NVIC 包括一个不可屏蔽中断（NMI），并提供高达 256 个中断优先级。处理器内核和 NVIC 的紧密耦合实现了中断服务程序（ISR）的快速执行，大大缩短了中断延迟。其实现方式是通过寄存器的硬件堆栈，以及挂起多个加载（load-multiple）和多个存储(store-multiple)操作的能力。中断处理程序不需要任何的汇编程序插桩(stub)，消除了所有来自 ISR 的代码开销。末尾连锁(Tail-chaining)优化也显著降低了从一个 ISR 切换到另一个 ISR 时所产生的开销。

为了优化低功耗设计，NVIC 集成了睡眠模式，包括一个深度睡眠功能，能够使整个设备快速进入掉电模式。

ES32 还支持其他的降功耗模式，详细内容参见“电源管理及低功耗模式”章节。

附录1. 1. 1. 1 系统级接口

Cortex-M3 处理器采用 AMBA 技术提供了多个接口，以实现高速、低等待时间的存储器访问。它支持非对齐的数据访问，实现了原子位操作（atomic bit manipulation），能够实现更快的外设控制、系统自旋锁（spinlock），以及线程安全的 Boolean 数据处理。

Cortex-M3 处理器有一个可选的存储器保护单元（MPU），提供细粒度的存储器控制，使应用程序能够实现安全特权级别，在不同任务基础上划分代码、数据和堆栈。在很多嵌入式应用（如汽车）中，这类需求正变得越来越重要。ES32 中包括了 MPU。

附录1. 1. 1. 2 集成的可配置调试功能

Cortex-M3 处理器实现了完全的硬件调试解决方案。通过传统的 JTAG 端口或一个双管脚的串行线调试（SWD）端口提供了对处理器和存储器的系统高可视性，SWD 非常适用于微控制器和其它小封装设备。MCU 厂商决定了调试特性的配置，因此它在不同设备和产品系列上可能存在差异。

在系统跟踪方面，除数据观察点和一个分析单元外，处理器中还集成了一个指令跟踪宏单元（ITM）。为了能够简单而高效地分析这些生成的系统事件，串行线浏览器（SWV）可通过一个管脚输出一个软件生成消息、数据跟踪和数据分析信息流。

可选的嵌入式跟踪宏单元（ETM）的面积远小于传统跟踪单元的面积，但提供了无与伦比的指令跟踪捕获能力，从而使很多低成本 MCU 第一次实现了完全的指令跟踪。

附录1. 1. 1. 3 Cortex-M3 处理器特性和优点汇总

- ◇ 系统外设的紧密集成不但节约了空间还降低了开发成本；
- ◇ Thumb 指令集兼备了高代码密度和 32 位性能；
- ◇ 具备代码修补能力，支持系统更新；
- ◇ 系统部件的功率控制优化；
- ◇ 集成睡眠模式，实现低功耗；
- ◇ 快速的代码执行允许使用较慢的处理器时钟或增加睡眠模式的时间；
- ◇ 硬件除法器和快速乘法器；
- ◇ 确定性的高性能中断处理，可用于对时间要求严格的应用中；
- ◇ 可选的存储器保护单元（MPU），用于对安全性要求严格的应用中；

◆ 全面的调试和跟踪能力：

- ◇ 串行线调试和串行线跟踪，减少了调试和跟踪所需的管脚数。

附录1. 1. 1. 4 Cortex-M3 内核外设

Cortex-M3 内核外设有：

◇ 可嵌套向量中断控制器

可嵌套向量中断控制器（NVIC）是一种支持低等待时间中断处理的嵌入式中断控制器。

◇ 系统控制模块

系统控制模块（SCB）是编程模型与处理器之间的接口。它提供了系统实现信息和系统控制，包括配置、控制以及系统异常的报告。

◇ 系统定时器

系统定时器 SysTick 是一个 24 位的递减计数定时器。它可以用作实时操作系统（RTOS）的节拍定时器或一个普通计数器。

◇ 存储器保护单元

存储器保护单元（MPU）通过为不同存储区定义存储器属性，提高了系统的可靠性。它最多可以提供 8 个不同的区域，还有一个可选的预定义背景区。

附录1. 2 ARM Cortex-M3 用户指南：指令集

附录1. 2. 1 指令集汇总

该处理器实现了一个 Thumb 指令集版本。下表列出了支持的指令。

下表中：

- ◇ 尖括号<>中为操作数的代替形式
- ◇ 花括号{}中为可选操作数
- ◇ “操作数”栏并未列出全部
- ◇ Op2 是一个灵活的第二操作数，可以是一个寄存器或是一个常数
- ◇ 多数指令可使用一个可选的条件代码后缀。

有关指令和操作数的更多信息，请参见指令描述部分。

助记符	操作数	简要描述	标志	章节（已改为对应名称）
ADC, ADCS	{Rd,} Rn, Op2	带进位加法	N，Z，C，V	ADD、ADC、SUB、SBC 和 RSB
ADD, ADDS	{Rd,} Rn, Op2	加法	N，Z，C，V	ADD、ADC、SUB、SBC 和 RSB
ADD, ADDW	{Rd,} Rn, #imm12	加法	N，Z，C，V	ADD、ADC、SUB、SBC 和 RSB
ADR	Rd, label	加载相对 PC 地址	-	ADR
AND, ANDS	{Rd,} Rn, Op2	逻辑“与”	N，Z，C	AND，ORR，EOR，BIC 和 ORN
ASR, ASRS	Rd, Rm, <Rs | #n>	算术右移	N，Z，C	ASR，LSL，LSR，ROR 和 RRX
B	label	跳转	-	B、BL、BX 和 BLX
BFC	Rd, #lsb, #width	位域清零	-	BFC 和 BFI
BFI	Rd, Rn, #lsb, #width	位域插入	-	BFC 和 BFI
BIC, BICS	{Rd,} Rn, Op2	位清零	N，Z，C	AND，ORR，EOR，BIC 和 ORN
BKPT	#imm	断点	-	BKPT
BL	label	带链接的跳转	-	B、BL、BX 和 BLX
BLX	Rm	带链接的跳转并切换指令集	-	B、BL、BX 和 BLX
BX	Rm	跳转并切换指令集	-	B、BL、BX 和 BLX
CBNZ	Rn, label	比较，结果非零则跳转	-	CBZ 和 CBNZ
CBZ	Rn, label	比较，结果为零则跳转	-	CBZ 和 CBNZ
CLREX	-	独占清零	-	CLREX

助记符	操作数	简要描述	标志	章节（已改为对应名称）
CLZ	Rd, Rm	计算前导零数目	-	CLZ
CMN, CMNS	Rn, Op2	与负值比较	N，Z，C，V	CMP 和 CMN
CMP, CMPS	Rn, Op2	比较	N，Z，C，V	CMP 和 CMN
CPSID	iflags	更改处理器状态，禁能中断	-	CPS
CPSIE	iflags	更改处理器状态，使能中断	-	CPS
DMB	-	数据内存屏障	-	DMB
DSB	-	数据同步屏障	-	DSB
EOR, EORS	{Rd,} Rn, Op2	异或	N，Z，C	AND，ORR，EOR，BIC 和 ORN
ISB	-	指令同步屏障	-	ISB
IT	-	If-Then 条件模块	-	IT
LDM	Rn{ ! }, reglist	加载多个寄存器，之后递增	-	LDM 和 STM
LDMDB, LDMEA	Rn{ ! }, reglist	加载多个寄存器，之前递减	-	LDM 和 STM
LDMFD, LDMIA	Rn{ ! }, reglist	加载多个寄存器，之后递增	-	LDM 和 STM
LDR	Rt, [Rn, #offset]	加载字至寄存器	-	内存访问指令
LDRB, LDRBT	Rt, [Rn, #offset]	加载字节至寄存器	-	内存访问指令
LDRD	Rt, Rt2, [Rn, #offset]	加载双字节至寄存器	-	LDR 和 STR（直接偏移量）
LDREX	Rt, [Rn, #offset]	独占加载寄存器	-	LDREX 和 STREX
LDREXB	Rt, [Rn]	独占加载字节至寄存器	-	LDREX 和 STREX
LDREXH	Rt, [Rn]	独占加载半字至寄存器	-	LDREX 和 STREX
LDRH, LDRHT	Rt, [Rn, #offset]	加载半字至寄存器	-	内存访问指令
LDRSB, LDRSBT	Rt, [Rn, #offset]	加载经有符号扩展后的字节至寄存器	-	内存访问指令
LDRSH, LDRSHT	Rt, [Rn, #offset]	加载经有符号扩展后的半字至寄存器	-	内存访问指令
LDRT	Rt, [Rn, #offset]	加载字至寄存器	-	内存访问指令
LSL, LSLS	Rd, Rm, <Rs | #n>	逻辑左移	N，Z，C	ASR，LSL，LSR，ROR 和 RRX
LSR, LSRS	Rd, Rm, <Rs |	逻辑右移	N，Z，C	ASR，LSL，LSR，ROR

助记符	操作数	简要描述	标志	章节（已改为对应名称）
	#n>			和 RRX
MLA	Rd, Rn, Rm, Ra	乘加，结果为 32 位	-	MUL、MLA 和 MLS
MLS	Rd, Rn, Rm, Ra	乘减，结果为 32 位	-	MUL、MLA 和 MLS
MOV, MOVS	Rd, Op2	移动	N，Z，C	MOV 和 MVN
MOVT	Rd, #imm16	移动到顶部	-	MOVT
MOVW, MOV	Rd, #imm16	移动 16 位常数	N，Z，C	MOV 和 MVN
MRS	Rd, spec_reg	将专用寄存器的内容移到通用寄存器中	-	MRS
MSR	spec_reg,Rm	将通用寄存器的内容移到专用寄存器中	N，Z，C，V	MSR
MUL, MULS	{Rd,} Rn, Rm	乘法，结果为 32 位	N，Z	MUL、MLA 和 MLS
MVN, MVNS	Rd, Op2	取反移动	N，Z，C	MOV 和 MVN
NOP	-	不执行任何操作	-	NOP
ORN, ORNS	{Rd,} Rn, Op2	逻辑“或非”	N，Z，C	AND，ORR，EOR，BIC 和 ORN
ORR, ORRS	{Rd,} Rn, Op2	逻辑“或”	N，Z，C	AND，ORR，EOR，BIC 和 ORN
POP	reglist	从堆栈弹出寄存器	-	PUSH 和 POP
PUSH	reglist	将寄存器推入堆栈	-	PUSH 和 POP
RBIT	Rd, Rn	反转位	-	REV，REV16，REVSH 和 RBIT
REV	Rd, Rn	反转字中的字节顺序	-	REV，REV16，REVSH 和 RBIT
REV16	Rd, Rn	反转每个半字中的字节顺序	-	REV，REV16，REVSH 和 RBIT
REVSH	Rd, Rn	反转低半字中的字节顺序，并将符号扩展	-	REV，REV16，REVSH 和 RBIT
ROR, RORS	Rd, Rm, <Rs | #n>	向右循环移	N，Z，C	ASR，LSL，LSR，ROR 和 RRX
RRX, RRXS	Rd, Rm	带扩展向右循环移	N，Z，C	ASR，LSL，LSR，ROR 和 RRX
RSB, RSBS	{Rd,} Rn, Op2	反向减法	N，Z，C，V	ADD、ADC、SUB、SBC 和 RSB
SBC, SBCS	{Rd,} Rn, Op2	带进位减法	N，Z，C，V	ADD、ADC、SUB、SBC 和 RSB
SBFX	Rd, Rn, #lsb,	有符号位域提取	-	SBFX 和 UBFX

助记符	操作数	简要描述	标志	章节（已改为对应名称）
	#width			
SDIV	{Rd,} Rn, Rm	有符号除法	-	SDIV 和 UDIV
SEV	-	发送事件	-	SEV
SMLAL	RdLo, RdHi, Rn, Rm	有符号乘加（32x32+64），结果为 64 位		UMULL，UMLAL，SMULL 和 SMLAL
SMULL	RdLo, RdHi, Rn, Rm	有符号乘法（32x32），结果为 64 位	-	UMULL，UMLAL，SMULL 和 SMLAL
SSAT	Rd, #n, Rm {,shift #s}	有符号饱和	Q	SSAT 和 USAT
STM	Rn{ ! }, reglist	存储多个寄存器，之后递增	-	LDM 和 STM
STMDB, STMEA	Rn{ ! }, reglist	存储多个寄存器，之前递减	-	LDM 和 STM
STMFD, STMIA	Rn{ ! }, reglist	存储多个寄存器，之后递增	-	LDM 和 STM
STR	Rt, [Rn, #offset]	存储寄存器字	-	内存访问指令
STRB, STRBT	Rt, [Rn, #offset]	存储寄存器字节	-	内存访问指令
STRD	Rt, Rt2, [Rn, #offset]	存储寄存器双字	-	LDR 和 STR（直接偏移量）
STREX	Rd, Rt, [Rn, #offset]	独占存储寄存器	-	LDREX 和 STREX
STREXB	Rd, Rt, [Rn]	独占存储寄存器，字节	-	LDREX 和 STREX
STREXH	Rd, Rt, [Rn]	独占存储寄存器，半字	-	LDREX 和 STREX
STRH, STRHT	Rt, [Rn, #offset]	存储寄存器，半字	-	内存访问指令
STRT	Rt, [Rn, #offset]	存储寄存器，字	-	内存访问指令
SUB, SUBS	{Rd,} Rn, Op2	减法	N，Z，C，V	ADD、ADC、SUB、SBC 和 RSB
SUB, SUBW	{Rd,} Rn, #imm12	减法	N，Z，C，V	ADD、ADC、SUB、SBC 和 RSB
SVC	#imm	超级用户调用	-	BKPT0
SXTB	{Rd,} Rm {,ROR #n}	有符号扩展一个字节	-	SXT 和 UXT
SXTH	{Rd,} Rm {,ROR #n}	有符号扩展一个半字	-	SXT 和 UXT

助记符	操作数	简要描述	标志	章节（已改为对应名称）
TBB	[Rn, Rm]	表跳转字节	-	TBB 和 TBH
TBH	[Rn, Rm, LSL #1]	表跳转半字	-	TBB 和 TBH
TEQ	Rn, Op2	相等测试	N，Z，C	TST 和 TEQ
TST	Rn, Op2	测试	N，Z，C	TST 和 TEQ
UBFX	Rd, Rn, #lsb, #width	无符号位域提取	-	SBFX 和 UBFX
UDIV	{Rd,} Rn, Rm	无符号除法	-	SDIV 和 UDIV
UMLAL	RdLo, RdHi, Rn, Rm	无符号乘加（32x32+64），结果为 64 位		UMULL，UMLAL，SMULL 和 SMLAL
UMULL	RdLo, RdHi, Rn, Rm	无符号乘法（32x32），结果为 64 位	-	UMULL，UMLAL，SMULL 和 SMLAL
USAT	Rd, #n, Rm {,shift #s}	无符号饱和	Q	SSAT 和 USAT
UXTB	{Rd,} Rm {,ROR #n}	用零扩展一个字节	-	SXT 和 UXT
UXTH	{Rd,} Rm {,ROR #n}	用零扩展一个半字	-	SXT 和 UXT
WFE	-	等待事件	-	BKPT1
WFI	-	等待中断	-	BKPT2

附录表 1-1 Cortex-M3 指令

附录1. 2. 2 内在函数

ANSI 不能直接访问某些 Cortex-M3 指令。本节描述了可生成这些指令的内在函数，这些函数由 CMIS 提供，也可以由 C 编译器提供。如果 C 编译器不支持合适的内在函数，则可能必须要用内联汇编器来访问这些指令。

CMSIS 提供以下内在函数，用以生成 ANSI 不能直接访问的指令：

指令	CMSIS 内在函数
CPSIE I	void__enable_irq(void)
CPSID I	void__disable_irq(void)
CPSIE F	void__enable_fault_irq(void)
CPSID F	void__disable_fault_irq(void)
ISB	void__ISB(void)
DSB	void__DSB(void)
DMB	void__DMB(void)
REV	uint32_t__REV(uint32_t int value)
REV16	uint32_t__REV16(uint32_t int value)
REVSH	uint32_t__REVSH(uint32_t int value)
RBIT	uint32_t__RBIT(uint32_t int value)
SEV	void__SEV(void)
WFE	void__WFE(void)
WFI	void__WFI(void)

附录表 1-2 用来生成一些 Cortex-M3 指令的 CMSIS 内在函数

CMSIS 还提供了一些函数，用于访问使用 MRS 和 MSR 指令的专用寄存器：

专用寄存器	访问	CMSIS 函数
PRIMASK	读	uint32_t__get_PRIMASK(void)
	写	void__set_PRIMASK(uint32_tvalue)
FAULTMASK	读	uint32_t__get_FAULTMASK(void)
	写	void__set_FAULTMASK(uint32_tvalue)
BASEPRI	读	uint32_t__get_BASEPRI(void)
	写	void__set_BASEPRI(uint32_tvalue)
CONTROL	读	uint32_t__get_CONTROL(void)
	写	void__set_CONTROL(uint32_tvalue)
MSP	读	uint32_t__get_MSP(void)
	写	void__set_MSP(uint32_tTopOfMainStack)
PSP	读	uint32_t__get_PSP(void)
	写	Void__set_PSP(uint32_tTopOfProcStack)

附录表 1-3 用来访问专用寄存器的 CMSIS 内在函数

附录1. 2. 3 关于指令描述

以下章节提供了与使用指令相关的更多信息：

附录1. 2. 3. 1 操作数

指令操作数可以是一个 ARM 寄存器、一个常数或是其他指令特有的参数。指令按操作数执 行，并通常将结果保存在一个目标寄存器中。当指令中有一个目标寄存器时，它通常在操作数前指定。

某些指令中的操作数比较灵活，因为它即可以是一个寄存器，也可以是一个常数，参见小节“灵活的第二操作数”。

附录1. 2. 3. 2 使用 PC 或 SP 时的限制

很多指令对操作数或目标寄存器是否可使用程序计数器（PC）或堆栈指针（SP）有限制。更多信息请参见指令描述。

注：当使用 BX、BLX、LDM、LDR 或 POP 指令写 PC 时，任意写入地址的位[0]必须为 1 才能正确执行指令，因为此位表示所需的指令集，而 Cortex-M3 处理器只支持 Thumb 指令。

附录1. 2. 3. 3 灵活的第二操作数

很多通用数据处理指令都有一个灵活的第二操作数，在各个指令的语法描述中表示为操作数 2（Operand2）。

Operand2 可以是“常数”或“带可选移位的寄存器”。

常数

以下列形式指定一个 Operand2 常数：

#constant

其中“ constant” 可以是：

◇ 在一个 32 位的字内，将一个 8 位值左移任意位数可以得到的任何常数；

◇ 任何 0x00XY00XY 形式的常数；

◇ 任何 0xXY00XY00 形式的常数；

◇ 任何 0xXYXYXYXY 形式的常数。

注：在上述常数中，X 和 Y 均为 16 进制数字。

此外，在少数指令中，“constant”可以是更大范围的值。这在单个的指令描述中说明。

当将一个 Operand2 常数与指令 MOVS、MVNS、ANDS、ORRS、ORNS、EORS、BICS、TEQ 或 TST 一起使用时，如果常数大于 255 且可通过移位一个 8 位值产生，则进位标志被更新为常数的位[31]。如果 Operand2 为其他任何常数，则这些指令不影响进位标志。

指令替代：当指定了一个不被允许的常数时，所使用的汇编器可能会产生一条等效指令。例如，某个汇编器可能将指令 CMP Rd，#0xFFFFFFFE 汇编为等效指令 CMN Rd，

#0x2。

带可选移位的寄存器

以下列形式指定 Operand2 寄存器：

Rm {，shift}

其中：

Rm 为保存第二操作数数据的寄存器。

shift 为用于 Rm 的一个可选移位。它可以是以下内容之一：

ASR#n：算术右移 n 位，1≤n≤32。

LSL#n：逻辑左移 n 位，1≤n≤31。

LSR#n：逻辑右移 n 位，1≤n≤32。

ROR#n：向右循环移 n 位，1 ≤n≤31。

RRX：带扩展向右循环移 1 位。

—：如省略，则不发生移位，等效于 LSL#0。

如果省略移位或指定 LSL #0，则指令使用 Rm 中的值。

如果指定了一个移位，则将移位用于 Rm 中的值，得到的 32 位值由指令使用。但寄存器 Rm 中的内容保持不变。指定一个带移位的寄存器还会更新与某些指令一起使用的进位标志。有关移位操作及其对进位标志影响方式的信息，请参见“移位操作”小节。

附录1. 2. 3. 4 移位操作

寄存器移位操作是根据一个规定的位数（移位长度）将一个寄存器内的位向左移或向右移。 寄存器移位可以按照以下方式进行：

◇ 直接使用指令 ASR、LSR、LSL、ROR 和 RRX 进行寄存器移位，移位结果被写入目标寄存器。

◇ 当指令（指定第二操作数作为一个带移位的寄存器的指令）计算 Operand2 时进行寄存器移位，见“灵活的第二操作数”。移位结果被指令所使用。

允许的移位长度取决于移位类型和指令，请参见各指令的描述或“灵活的第二操作数”小节。如果移位长度 为 0，则不发生移位。除了指定的移位长度为 0 的情况之外，寄存器移位操作都会更新进位 标志。以下各小节描述了各种移位操作，以及它们对进位标志的影响方式。在这些描述中， Rm 为包含待移位值的寄存器，n 为移位长度。

ASR

算术右移 n 位，是将寄存器 Rm 左边 32-n 位向右移 n 个位置，成为结果中的右边 32-n 位。并且，它还将寄存器原来的位[31]复制到结果左边的 n 个位上。参见下图。

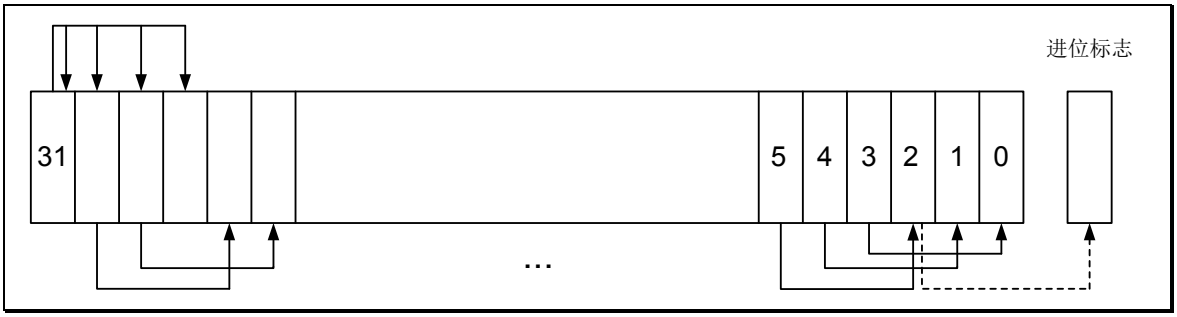
可使用 ASR #n 操作来将寄存器 Rm 中的值除以 2^n，得到的结果向负无穷方向舍入。

当指令为 ASRS，或指令 MOVS、MVNS、ANDS、ORRS、ORNS、EORS、BICS、TEQ 或 TST 中将 ASR #n 用于 Operand2 时，进位标志更新为寄存器 Rm 的最后一

个移出位，即位[n-1]。

◇ 如果 n 为 32 或更大，则结果中的所有位都设置为 Rm 位[31]的值。

◇ 如果 n 为 32 或更大且进位标志被更新，则其更新为 Rm 位[31]的值。



附录图 1-2 ASR #3

LSR

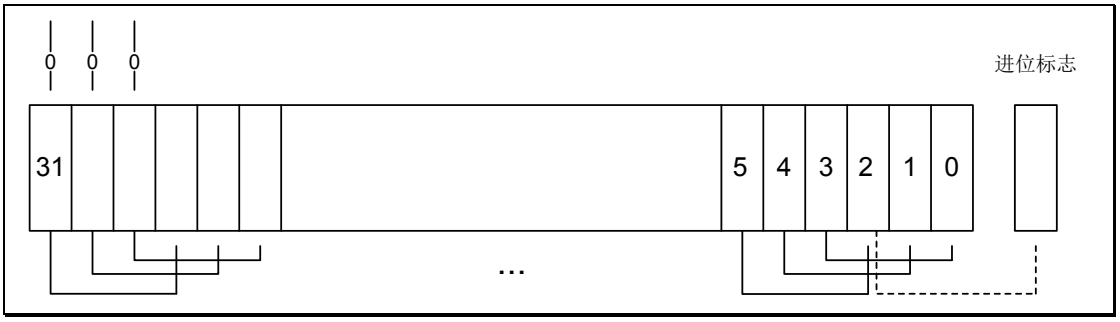
逻辑右移 n 位，是将寄存器 Rm 左边 32-n 位向右移 n 个位置，成为结果中的右边 32-n 位。并且，将结果的左边 n 位设置为 0。参见下图。

如果 Rm 中的值被视为一个无符号整数，则可以使用 LSR #n 操作将寄存器 Rm 中的值除以 2^n。

当指令为 LSRS，或将 LSR #n 用于指令 MOVS、MVNS、ANDS、ORRS、ORNS、EORS、BICS、TEQ 或 TST 的 Operand2 时，进位标志更新为寄存器 Rm 的最后一个移出位，即位[n-1]。

◇ 如果 n 为 32 或更大，则结果中的全部位都清零。

◇ 如果 n 为 33 或更大且进位标志被更新，则其更新为 0。



附录图 1-3 LSR #3

LSL

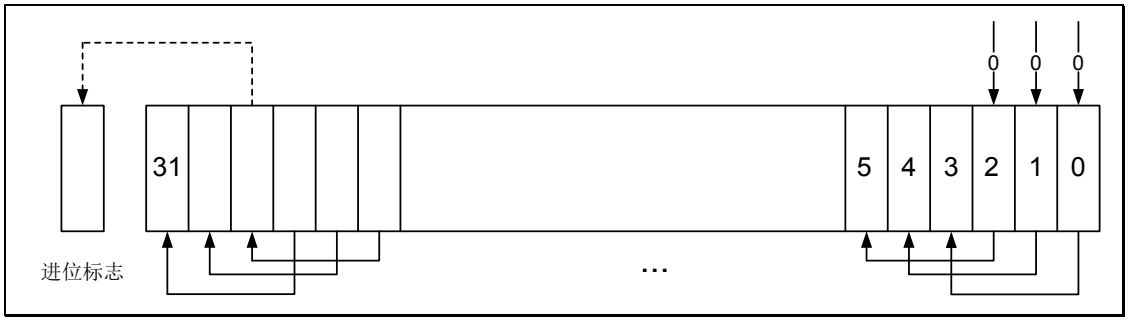
逻辑左移 n 位，是将寄存器 Rm 右边 32-n 位向左移 n 个位置，成为结果的左边 32-n 位。并且将结果的右边 n 位设置为 0。参见下图。

如果寄存器 Rm 中的值被视为一个无符号整数或一个 2 的补码有符号整数，则可使用 LSL #n 操作将寄存器 Rm 中的值乘以 2n。这可能会引发不带报警的溢出。

当指令为 LSLS，或将 LSL #n（n 非零）用于指令 MOVS、MVNS、ANDS、ORRS、ORNS、EORS、BICS、TEQ 或 TST 的 Operand2 时，进位标志更新为寄存器 Rm 的最后一个移出位，即位[32-n]。当使用 LSL #0 时，这些指令不影响进位标志。

◇ 如果 n 为 32 或更大，则结果中全部位都清零。

◇ 如果 n 为 33 或更大且进位标志被更新，则其更新为 0。



附录图 1-4 LSL #3

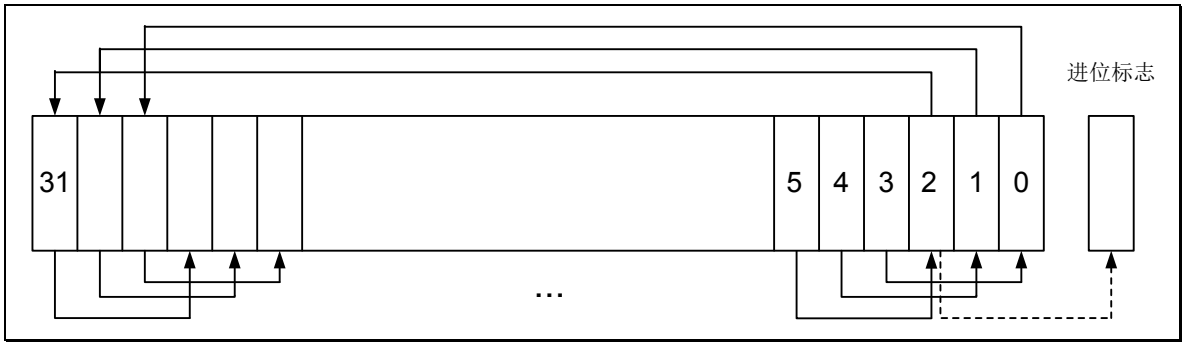
ROR

向右循环移 n 位，是将寄存器 Rm 左边 32-n 位向右移 n 个位置，成为结果的右边 32-n 位。并且将寄存器的右边 n 位移到结果的左边 n 位。参见下图。

当指令为 RORS，或将 ROR #n 用于指令 MOVS、MVNS、ANDS、ORRS、ORNS、EORS、BICS、TEQ 或 TST 的 Operand2 时，进位标志更新为寄存器 Rm 的最后一个循环位，即位[n-1]。

◇ 如果 n 为 32，则结果的值与 Rm 中的值相同，并且如果进位标志被更新，则它更新为 Rm 的位[31]。

◇ 移位长度 n 大于 32 的 ROR 与移位长度为 n-32 的 ROR 相同。

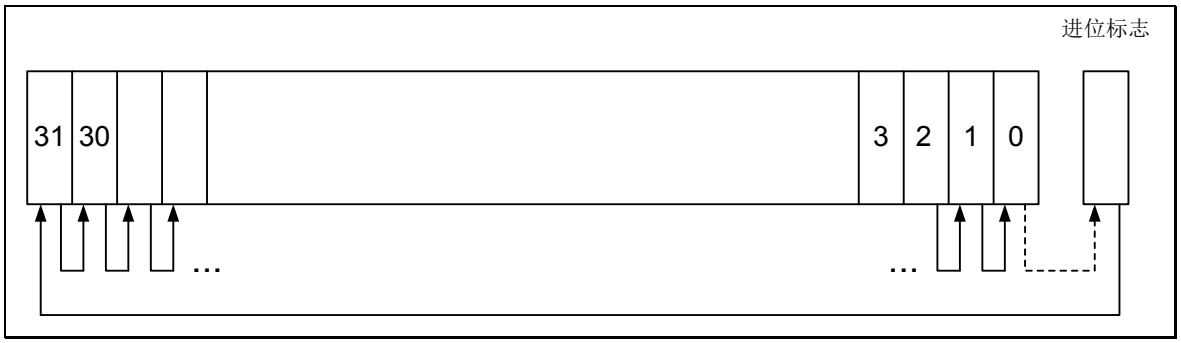


附录图 1-5 ROR #3

RRX

带扩展的向右循环移是将寄存器 Rm 中的位向右移一位。它还将进位标志复制到结果的位[31]中。参见下图。

当指令为 RRXS，或将 RRX 用于指令 MOVS、MVNS、ANDS、ORRS、ORNS、EORS、BICS、TEQ 或 TST 的 Operand2 时，进位标志更新为寄存器 Rm 的位[0]。



附录图 1-6 RRX

附录1. 2. 3. 5 地址对齐

对齐访问是将一个按字对齐的地址用于一个字、双字和多字的访问中，或将一个按半字对齐的地址用于半字访问中。字节访问总是对齐的。

Cortex-M3 处理器仅对下列指令支持非对齐的访问：

- ◇ LDR，LDRT
- ◇ LDRH，LDRHT
- ◇ LDRSH，LDRSHT
- ◇ STR，STRT
- ◇ STRH，STRHT

如果执行非对齐访问，则所有其他的加载和存储指令都会产生使用故障异常，因此，这些指令的访问都必须是地址对齐访问。有关使用故障的更多信息，见“故障处理”小节。

非对齐访问通常慢于对齐访问。另外，某些存储区可能不支持非对齐访问。因此，ARM 建议，程序员们应确保每次访问都是对齐访问。为了避免非对齐访问的偶然发生，应在配置和控制寄存器中使用 UNALIGN_TRP 位来捕获所有的非对齐访问，见“配置和控制寄存器”小节。

附录1. 2. 3. 6 相对 PC 的表达式

相对 PC 的表达式或标签是一个代表指令地址或立即数（literal data）的符号。在指令中，它表示为：PC 值加上或减去一个数值偏移量。汇编器根据标签和当前指令的地址计算出所需的偏移量。如果偏移量过大，则汇编器产生一个错误。

- ◇ 对于 B、BL、CBNZ 和 CBZ 指令，PC 值为当前指令地址加上 4 字节。
- ◇ 对于使用标签的所有其他指令，PC 值为当前指令地址加上 4 字节，结果的位[1]清零，使之按字对齐。
- ◇ 所用汇编器可能允许相对 PC 表达式的其他语法，例如一个标签加上或减去一个数字，或一个[PC，#number]形式的表达式。

附录1. 2. 3. 7 条件执行

大多数数据处理指令都具有一个选项，该选项可以根据操作结果来更新应用程序状态寄存器（SPAR）中的条件标志，见“程序状态寄存器”小节。有些指令会更新全部标志，有些指令则只更新一个子集。如果某个标志没有更新，则会保留其原始值。受影响的标志参见指令描述部分。

可以根据其他指令中设置的条件标志有条件地执行一条指令，执行时间为：

◇ 在更新标志的指令后立即执行；

◇ 在没有更新标志的任意数目的插入指令之后执行。

当使用条件跳转或为指令添加条件代码后缀时，可以实现有条件地执行。表“条件代码后缀”是一份后缀清单，这些后缀都可以添加到指令中使其可以有条件地执行。条件代码后缀使处理器能够根据标志来测试一个条件。如果某个条件指令的条件测试失败，则指令会：

◇ 不执行；

◇ 不向其目标寄存器写入任何值；

◇ 不影响任何标志；

◇ 不产生任何异常。

除了条件跳转外，条件指令必须处于一个 If-Then 指令模块中。有关使用 IT 指令的更多信息和限制，请参见 “IT”小节。如果在 IT 块外存在条件指令，汇编器可能自动插入一条 IT 指令（取决于厂商）。

使用 CBZ 和 CBNZ 指令将寄存器值与零比较，并根据结果执行跳转（branch）。

本节介绍以下内容：“条件标志”和“条件代码后缀”。

条件标志

APSR 包括以下条件标志：

N—操作结果为负值时设置为 1，否则清零。

Z—操作结果为零时设置为 1，否则清零。

C—操作导致一个进位出现时设置为 1，否则清零。

V—操作引起溢出时设置为 1，否则清零。

有关 APSR 的更多信息，请参见“ 程序状态寄存器”小节。以下情况中出现进位：

◇ 如果加法结果大于或等于 2^{32}；

◇ 如果减法结果为正值或零；

◇ 在一个移动或逻辑指令中，内联桶式（inline barrel）移位器操作的结果。

如果一个加法、减法或比较的结果大于或等于 2^{31}，或小于-2^{31}，则发生溢出。

注：多数指令仅在指定后缀 S 的情况下才更新状态标志。更多信息请参见指令描述。

条件代码后缀

可有条件地执行的指令具有一个可选条件代码，如{cond}中的语法描述所示。条件执行时要求之前有一条 IT 指令。仅当 APSR 中的条件代码标志满足指定的条件时，才会执行带有条件代码的指令。下表显示了可使用的条件代码。

可以通过 IT 指令使用有条件地执行来减少代码中跳转指令的数量。

下表还显示了条件代码后缀和 N、Z、C 和 V 标志之间的关系。

后缀	标志	含义
EQ	Z = 1	等于
NE	Z = 0	不等于
CS/HS	C = 1	进位置位/无符号大于或等于
CC/LO	C = 0	进位清零/无符号小于
MI	N = 1	减/负
PL	N = 0	加/正或零
VS	V = 1	溢出
VC	V = 0	无溢出
HI	C = 1 且 Z = 0	无符号大于
LS	C = 0 或 Z = 1	无符号小于或等于
GE	N = V	有符号大于或等于
LT	N！= V	有符号小于
GT	Z = 0 且 N = V	有符号大于
LE	Z = 1 且 N！= V	有符号小于或等于
AL	任意值	始终。无指定后缀时的缺省值

附录表 1-4　条件代码后缀

示例：绝对值　展示了使用一个条件指令来得到某个数的绝对值。R0 = ABS（R1）。

示例：比较和更新值　展示了当符号值 R0 大于 R1 且 R2 大于 R3 时，使用条件指令来更新 R4 的值。

示例：绝对值：

```
MOVS R0，R1   ; R0 = R1, setting flags
ITMI   ; IT instruction for the negative condition
RSBMI      R0，R1，#0  ; If negative, R0 = -R1
```

示例：比较和更新值：

```
CMP   R0, R1   ; Compare R0 and R1, setting flags
ITT GT ; IT instruction for the two GT conditions
CMPGT R2, R3 ; If 'greater than', compare R2 and R3, setting flags
MOVGT R4, R5 ; If still 'greater than', do R4 = R5
```

附录1. 2. 3. 8 指令宽度选择

很多指令既可产生 16 位编码，也可产生 32 位编码，这取决于指定的操作数和目标寄存器。对其中的某些指令，可使用一个指令宽度后缀来强制设置一个特定的指令尺寸。“.W ”后缀会强制使用 32 位指令编码。“.N ”后缀会强制使用 16 位指令编码。

如果指定了一个指令宽度后缀，但汇编器无法生成所要求宽度的指令编码，则产生一个错误。

注：某些情况下指定.W 后缀是非常有必要的，例如，当操作数为一个指令的标签或立即数时，如跳转指令的情况。这是因为汇编器可能无法自动生成大小正确的编码。

当使用指令宽度后缀时，将其紧跟在指令助记符和条件代码（如果有）后面。如下列出了带指令宽度后缀的指令。

示例：指令宽度选择

```
BCS.W Label          ; creates a 32-bit instruction even for a short branch
ADDS.W R0，R0，R1    ; creates a 32-bit instruction even though the same
                     ; operation can be done by a 16-bit instruction
```

附录1. 2. 4 内存访问指令

下表中列出了内存访问指令：

	简要描述	参见（已改为对应章节名称）
ADR	加载相对 PC 地址	ADR
CLREX	独占清零	CLREX
LDM{mode}	加载多个寄存器	LDM 和 STM
LDR{type}	利用直接偏移量加载寄存器	LDR 和 STR（直接偏移量）
LDR{type}	利用寄存器偏移量加载寄存器	LDR 和 STR（寄存器偏移量）
LDR{type}T	通过非特权访问加载寄存器	LDR 和 STR（非特权）
LDR	利用相对 PC 地址加载寄存器	LDR（相对 PC）
LDREX{type}	独占加载寄存器	LDREX 和 STREX
POP	从堆栈弹出寄存器	PUSH 和 POP
PUSH	将寄存器推入堆栈	PUSH 和 POP
STM{mode}	存储多个寄存器	LDM 和 STM
STR{type}	利用直接偏移量存储寄存器	LDR 和 STR（直接偏移量）
STR{type}	利用寄存器偏移量存储寄存器	LDR 和 STR（寄存器偏移量）
STR{type}T	通过非特权访问存储寄存器	LDR 和 STR（非特权）
STREX{type}	独占存储寄存器	LDREX 和 STREX

附录表 1-5 内存访问指令

附录1. 2. 4. 1 ADR

加载相对 PC 地址。

语法

ADR{cond} Rd，label

其中：

cond 为一个可选的条件代码，见“条件执行”小节。

Rd 为目标寄存器。

label 为一个相对 PC 表达式。参见“相对 PC 的表达式”小节。

操作

ADR 通过将一个立即值与 PC 值相加来确定地址，并将结果写入目标寄存器。

ADR 生成与位置无关的代码，因为地址相对于 PC。

如果使用 ADR 为 BX 或 BLX 指令生成目标地址，则必须确保将所生成地址的位[0]设为 1，才能保证正确执行。

label 的值必须在 PC 中地址的-4095 到+4095 范围内。

注：要得到最大偏移范围或生成未按字对齐的地址时可能必须使用到“.W”后缀。参见“指令宽度选择 ”小节。

限制

Rd 不得为 SP 且不得为 PC。

条件标志

此指令不改变标志。

示例

```
ADR   R1，TextMessage        ; Write address value of a location labelled as
                             ; TextMessage to R1
```

附录1. 2. 4. 2 LDR 和 STR（直接偏移量）

带有直接偏移量、前变址直接偏移量或后变址直接偏移量的加载和存储。

语法

```
op{type}{cond} Rt，[Rn {，#offset}]      ；直接偏移量
op{type}{cond} Rt，[Rn，#offset]!        ；前变址
op{type}{cond} Rt，[Rn]，#offset         ；后变址
opD{cond} Rt，Rt2，[Rn {，#offset}]   ；直接偏移量，双字
opD{cond} Rt，Rt2，[Rn，#offset]!     ；前变址，双字
opD{cond} Rt，Rt2，[Rn]，#offset      ；后变址，双字
```

其中：

Op 可为以下指令之一：

LDR：加载寄存器。

STR：存储寄存器。

Type 可以是下列项之一：

B：无符号字节，加载时零扩展到 32 位。

SB：有符号字节，符号扩展到 32 位（仅 LDR）。

H：无符号半字，加载时零扩展到 32 位。

SH：有符号半字，符号扩展到 32 位（仅 LDR）。

—：如果是字，则省略。

cond 为一个可选的条件代码，见 “条件执行”小节 。

Rt 为要加载或存储的寄存器。

Rn 为存储器地址所基于的寄存器。

offset 为一个 Rn 的偏移量。如果省略 offset，则该地址为 Rn 的内容。

Rt2 为附加寄存器，在双字操作时使用，用于加载或存储。

操作

LDR 指令将存储器中的一个值加载到一个或两个寄存器中。

STR 指令将一个或两个寄存器值存储到存储器中。

带直接偏移量的加载和存储指令可用于以下寻址模式：

◇ 偏移量寻址

在寄存器 Rn 获得的地址上加上或减去一个偏移量。结果用于存储器的访问地址。寄存器 Rn 不改变。此模式的汇编语言语法为：

[Rn，#offset]

◇ 前变址寻址

在寄存器 Rn 获得的地址上加上或减去一个偏移量。结果用于存储器的访问地址，并写回寄存器 Rn 中。此模式的汇编语言语法为：

[Rn，#offset]!

◇ 后变址寻址

使用来自寄存器 Rn 的地址作为存储器的访问地址。给地址加上或减去一个偏移量，并写回寄存器 Rn 中。此模式的汇编语言语法为：

[Rn]，#offset

要加载或存储的值可以是一个字节、半字、字或双字。字节和半字可以是有符号的，也可以是无符号的。参见“地址对齐”小节。

下表显示了直接、前变址和后变址形式的偏移量范围。

指令类型	直接偏移量	前变址偏移量	后变址偏移量
字、半字、有符号半字、 字节或有符号字节	-255~4095	-255~255	-255~255
双字	-1020~1020 范围内 4 的倍数	-1020~1020 范围内 4 的倍数	-1020~1020 范围内 4 的倍数

附录表 1-6 偏移量范围

限制

对于加载指令：

◇ 仅对字加载，Rt 可以是 SP 或 PC；

◇ 对双字加载，Rt 必须与 Rt2 不同；

◇ 前变址或后变址形式中，Rn 必须与 Rt 和 Rt2 不同。

◇ 当字加载指令中的 Rt 为 PC 时：

◇ 加载值的位[0]必须为 1 才能正确执行；

◇ 发生到当被加载值位[0]变为 0 时生成的一个地址的跳转；

◇ 如果指令是有条件的，则它必须是 IT 块中的最后一条指令。

◇ 对于存储指令：

◇ 仅对字存储，Rt 可以是 SP 或 PC；

◇ Rt 不得为 PC；

◇ Rn 不得为 PC；

◇ 前变址或后变址形式中，Rn 必须与 Rt 和 Rt2 不同。

条件标志

这些指令不改变标志。

示例

```
LDR      R8，[R10]                ; Loads R8 from the address in R10.
LDRNE    R2，[R5，#960]!          ; Loads (conditionally) R2 from a word
                                  ; 960 bytes above the address in R5, and
```

```
                                        ; increments R5 by 960.
STR     R2，[R9,#const-struc]           ; const-struc is an expression evaluating
                                        ; to a constant in the range 0-4095.
STRH    R3，[R4]，#4                    ; Store R3 as halfword data into address in
                                        ; R4, then increment R4 by 4
LDRD    R8，R9，[R3，#0x20]             ; Load R8 from a word 32 bytes above the
                                        ; address in R3, and load R9 from a word 36
                                        ; bytes above the address in R3
STRD    R0，R1，[R8]，#-16              ; Store R0 to address in R8, and store R1 to
                                        ; a word 4 bytes above the address in R8,
                                        ; and then decrement R8 by 16.
```

附录1. 2. 4. 3 LDR 和 STR（寄存器偏移量）

带有寄存器偏移量的加载与存储。

语法

op{type}{cond} Rt，[Rn，Rm {，LSL #n}]

其中：

op 为以下指令之一：

LDR：加载寄存器。

STR：存储寄存器。

type 为下列项之一：

B：无符号字节，加载时零扩展到 32 位。

SB：有符号字节，符号扩展到 32 位（仅 LDR）。

H：无符号半字，加载时零扩展到 32 位。

SH：有符号半字，符号扩展到 32 位（仅 LDR）。

—：如果是字，则省略。

cond 为一个可选的条件代码，见“条件执行”小节 。

Rt 为要加载或存储的寄存器。

Rn 为存储器地址所基于的寄存器。

Rm 为一个寄存器，包含要用作偏移量的值。

LSL #n 为一个可选的移位，n 的范围为 0~3。

操作

LDR 指令将存储器的一个值加载到寄存器。

STR 指令将一个寄存器值存入存储器。

加载或存入的存储器地址为来自寄存器 Rn 的一个偏移量。偏移量由寄存器 Rm 指定，并可使用 LSL 左移最多 3 位。

要加载或存储的值可以是一个字节、半字，或字。对于加载指令，字节和半字可以是有符号 的，也可以是无符号的。参见“地址对齐”小节。

限制

在这些指令中：

◇ Rn 不得为 PC；

◇ Rm 不得为 SP 且不得为 PC；

◇ Rt 仅在字加载或字存储时才可以是 SP；

◇ Rt 仅在字加载时才可以是 PC。

当字加载指令中的 Rt 为 PC 时：

◇ 被加载值位[0]必须为 1 才能正确执行，并且发生到这个以半字对齐的地址的跳转；

◇ 如果指令是有条件的，它必须是 IT 块中的最后一条指令。

条件标志

这些指令不改变标志。

示例

```
STR   R0，[R5，R1]          ; Store value of R0 into an address equal to
                            ; sum of R5 and R1
LDRSB R0，[R5，R1，LSL #1]  ; Read byte value from an address equal to
                            ; sum of R5 and two times R1, sign extended it
                            ; to a word value and put it in R0
STR   R0，[R1，R2，LSL #2]  ; Stores R0 to an address equal to sum of R1
                            ; and four times R2
```

附录1. 2. 4. 4 LDR 和 STR（非特权）

带非特权访问的加载和存储。

语法

op{type}T{cond} Rt，[Rn {，#offset}] ; 直接偏移量

其中：

op 为下列指令之一：

LDR：加载寄存器。

STR：存储寄存器。

type 为下列项之一：

B：无符号字节，加载时零扩展到 32 位。

SB：有符号字节，符号扩展到 32 位（仅 LDR）。

H：无符号半字，加载时零扩展到 32 位。

SH：有符号半字，符号扩展到 32 位（仅 LDR）。

—：如果是字，则省略。

cond 为一个可选条件代码，见“条件执行”小节。

Rt 为要加载或存储的寄存器。

Rn 为存储器地址所基于的寄存器。

offset 为 Rn 的偏移量，取值范围为 0~255。如果省略 offset，则该地址为 Rn 中的值。

操作

这些加载和存储指令完成与带直接偏移量的存储器访问指令相同的功能。差别是，即使在特权软件中，这些指令也只能进行非特权访问。

当在非特权软件中使用时，这些指令与带直接偏移量的正常存储器访问指令完全一样。

限制

在这些指令中：

◇ Rn 不得为 PC；

◇ Rt 不得为 SP 且不得为 PC。

条件标志

这些指令不改变标志。

示例

```
STRBTEQ      R4，[R7]     ;   Conditionally store least significant byte in
                          ;   R4 to an address in R7, with unprivileged access
LDRHT    R2，[R2，#8]     ;   Load halfword value from an address equal to
                          ;   sum of R2 and 8 into R2, with unprivileged access
```

附录1. 2. 4. 5 LDR（相对 PC）

从存储器加载寄存器。

语法

```
LDR{type}{cond} Rt，label
LDRD{cond} Rt，Rt2，label   ；加载双字
```

type 为下列项之一：

B：无符号字节，加载时零扩展到 32 位。

SB：有符号字节，符号扩展到 32 位（仅 LDR）。

H：无符号半字，加载时零扩展到 32 位。

SH：有符号半字，符号扩展到 32 位（仅 LDR）。

—：如果是字，则省略。

cond 为一个可选条件代码，见“条件执行”小节。

Rt 为要加载或存储的寄存器。

Rt2 为第二个要加载或存储的寄存器 。

label 为一个相对 PC 表达式。参见“相对 PC 的表达式”小节。

操作

LDR 用一个相对 PC 存储器地址的值加载寄存器。存储器地址由标签指定，或由一个 PC 偏移量指定。

加载或存储的值可以是一个字节、半字或字。对于加载指令，字节和半字可以是有符号的，也可以是无符号的。参见“地址对齐”小节。

label 必须在当前指令的限制范围内。下表显示了标签和 PC 之间的可能的偏移量。

指令类型	偏移量范围
字、半字、有符号半字、字节、有符号字节	-4095~4095
双字	-1020~1020

附录表 1-7 偏移量范围

注：要获得最大偏移量范围可能需要使用“.W”后缀。见“指令宽度选择”小节。

限制

在这些指令中：

◇ 仅对字加载，Rt 可以是 SP 或 PC；

◇ Rt2 不得为 SP 且不得为 PC；

◇ Rt 必须与 Rt2 不同。

当一个字加载指令中的 Rt 为 PC 时：

◇ 被加载值位[0]必须为 1 才能正确执行，并且发生到这个以半字对齐的地址上的跳转。

◇ 如果指令是有条件的，它必须是 IT 块中的最后一条指令。

条件标志

这些指令不改变标志。

示例

```
LDR   R0，LookUpTable        ;   Load R0 with a word of data from an address
                             ;   labelled as LookUpTable
LDRSB    R7，localdata       ;   Load a byte value from an address labelled
                             ;   as localdata, sign extend it to a word
                             ;   value, and put it in R7
```

附录1. 2. 4. 6 LDM 和 STM

加载和存储多个寄存器。

语法

op{addr_mode}{cond} Rn{!}，reglist

其中：

op 为下列指令之一：

LDM：加载多个寄存器。

STM：存储多个寄存器。

addr_mode 为下列项之一：

IA：每次访问后递增地址。这是默认设置。

DB：每次访问前递减地址。

cond 为一个可选条件代码，见“条件执行”小节。

Rn 为存储器地址所基于的寄存器 。

“!”为一个可选的写回后缀。如果有“!”，则要加载或存储到的最终地址被写回到 Rn。

reglist 为要加载或存储的一个或多个寄存器列表，括在大括号内。它可以包含寄存器范围。 如果包含一个以上的寄存器或寄存器范围，则必须以逗号隔开。

LDM 和 LDMFD 与 LDMIA 同义。LDMFD 指它用于从满降序堆栈（full descending stack）弹出（pop） 数据。

LDMEA 与 LDMDB 同义，表示它用于从空升序堆栈（empty ascending stack）弹出数据。

STM 和 STMEA 与 STMIA 同义，STMEA 表示它用于向空升序堆栈推入（push）数据。

STMFD 与 STMDB 同义，STMFD 表示它用于向满降序堆栈推入数据。

操作

LDM 指令将基于 Rn 的存储器地址的字值加载到 reglist 中的寄存器。

STM 指令将 reglist 中寄存器的字值存储到基于 Rn 的存储器地址。

对于 LDM、LDMIA、LDMFD、STM、STMIA 和 STMEA，用于访问的存储器地址以 4 个字节为间隔，范围从 Rn 到 Rn + 4x（n-1），其中 n 为 reglist 中的寄存器序号。访问按照寄存器序号递增的顺序进行，最低序号的寄存器使用最低的存储器地址，最高序号的寄存器使用最高的存储器地址。如果指定了写回后缀，则 Rn + 4x（n-1）的值被写回到 Rn。

对于 LDMDB、LDMEA、STMDB 和 STMFD，用于访问的存储器地址以 4 个字节为间隔，范围为从 Rn 到 Rn - 4x（n-1），其中 n 为 reglist 中的寄存器序号。访问是按照寄存器序号递减的顺序进 行，最高序号的寄存器使用最高的存储器地址，最低序号的

寄存器使用最低的存储器地址。 如果指定了写回后缀，则 Rn - 4x（n-1）的值被写回到 Rn。

PUSH 和 POP 指令可以用此形式表示。详情参见“PUSH 和 POP”小节。

限制

在这些指令中：

◇ Rn 不得为 PC；

◇ reglist 不得包括 SP；

◇ 在任何 STM 指令中，reglist 不得包括 PC；

◇ 在任何 LDM 指令中，如果 reglist 包括 LR，则它不得包括 PC；

◇ 如果指定了写回后缀，则 reglist 不得包括 Rn。

STM 指令中，当 PC 包括在 reglist 中时：

◇ 加载到 PC 值的位[0]必须为 1 才能正确执行，并且发生到此半字对齐地址的跳转。

◇ 如果指令是有条件的，则它必须是 IT 块中的最后一条指令。

条件标志

这些指令不改变标志。

示例

```
LDM   R8,{R0,R2,R9}          ; LDMIA is a synonym for LDM
STMDB R1!,{R3-R6,R11,R12}
```

不正确示例

```
STM   R5!,{R5,R4,R9}         ; Value stored for R5 is unpredictable
LDM   R2, {}                 ; There must be at least one register in the list
```

附录1. 2. 4. 7 PUSH 和 POP

向一个满降序堆栈推入和弹出寄存器。

语法

PUSH{cond} reglist POP{cond} reglist

其中：

cond 为一个可选条件代码，见“条件执行”小节。

reglist 为一个非空的寄存器列表，括在大括号内。它可以包括寄存器范围。如果包含了一个以上寄存器或寄存器范围，则必须以逗号隔开。

PUSH 和 POP 与 STMDB 和 LDM（或 LDMIA）同义，以存储器地址进行基于 SP 的访问，访问的最终地址被写回 SP。在这些情况下，PUSH 和 POP 是首选助记符。

操作

PUSH 按照寄存器序号递减的顺序将寄存器存储到堆栈上，最高序号的寄存器使用最高的存 储器地址，最低序号的寄存器使用最低的存储器地址。

POP 按照寄存器序号递增的顺序从堆栈加载寄存器，最低序号的寄存器使用最低的存储器地 址，最高序号的寄存器使用最高的存储器地址。

更多信息请参见“LDM 和 STM”小节。

限制

在这些指令中：

◇ reglist 不得包括 SP；

◇ 对于 PUSH 指令，reglist 不得包括 PC；

◇ 对于 POP 指令，如果 reglist 包括 LR，则它不得包括 PC。

在 POP 指令中，当 PC 包括在 reglist 中时：

◇ 被加载到 PC 值的位[0]必须为 1 才能正确执行，并且发生到此半字对齐地址的跳转。

◇ 如果指令是有条件的，它必须是 IT 块中的最后一条指令。

条件标志

这些指令不改变标志。

示例

```
PUSH    {R0,R4-R7}
PUSH    {R2,LR}
POP     {R0,R10,PC}
```

附录1. 2. 4. 8 LDREX 和 STREX

独占加载和存储寄存器。

语法

LDREX{cond} Rt，[Rn {，#offset}]
STREX{cond} Rd，Rt，[Rn {，#offset}]
LDREXB{cond} Rt，[Rn]
STREXB{cond} Rd，Rt，
[Rn] LDREXH{cond} Rt，
[Rn] STREXH{cond} Rd，Rt，[Rn]

其中：

cond 为一个可选的条件代码，见“条件执行”小节。

Rd 为存放返回状态的目标寄存器。

Rt 为要加载或存储的寄存器。

Rn 为存储器地址所基于的寄存器。

offset 为应用于 Rn 中的值的可选偏移量。如果省略 offset，则该地址为 Rn 中的值。

操作

LDREX、LDREXB 和 LDREXH 分别从一个存储器地址加载一个字、字节和半字。

STREX，STREXB 和 STREXH 分别尝试将一个字、字节和半字存储到某个存储器地址。独占存储指令中所用的地址必须与刚刚执行的独占加载指令中的地址相同。独占存储指令存储值的数据大小也必须与之前独占加载指令加载值的数据大小相同。这就是说，软件一定要用一条独占加载指令和一条匹配的独占存储指令来执行同步操作，见“同步原语”小节。

如果一条独占存储指令执行存储，则它将 0 写入其目标寄存器。如果它未执行存储，则将 1 写入其目标寄存器。如果独占存储指令将 0 写入目标寄存器，则可保证在独占加载和独占存 储指令之间，系统中没有其他进程访问过此存储单元。

出于性能考虑，尽量将相应独占加载和独占存储指令之间的指令数量降到最低。

注：如果执行一个独占存储指令的地址不同于之前独占加载指令的地址，则结果不可预知。

限制

在这些指令中：

◇ 不要使用 PC；

◇ Rd 和 Rt 不要使用 SP；

◇ 对于 STREX，Rd 必须与 Rt 和 Rn 都不同；

◇ offset 的值必须为 0~1020 范围内 4 的倍数。

条件标志

这些指令不改变标志。

示例

```
MOV   R1, #0x1                    ; Initialize the „lock taken‟ valuetry
LDREX     R0, [LockAddr]          ; Load the lock value
CMP   R0, #0                      ; Is the lock free?
ITT     EQ                        ; IT instruction for STREXEQ and CMPEQ
STREXEQ  R0, R1, [LockAddr        ; Try and claim the lock
CMPEQ     R0, #0                  ; Did this succeed?
BNE   try                         ; No – try again
....                              ; Yes – we have the lock
```

附录1. 2. 4. 9 CLREX

独占清零。

语法

CLREX{cond}

其中：cond 为一个可选的条件代码，见“条件执行”小节。

操作

使用 CLREX 使下一条 STREX、STREXB 或 STREXH 指令把 1 写入其目标寄存器且不执行存储。在异常处理程序代码中，当一个同步操作中独占加载指令和相应的独占存储指令之间发生异常时，它可用于强制独占存储失败。

更多信息请参见“同步原语”小节。

条件标志

这些指令不改变标志。

示例

```
CLREX
```

附录1. 2. 5 通用数据处理指令

下表列出了数据处理指令：

助记符	简要描述	参见（已改为对应章节名称）
ADC	带进位加法	ADD、ADC、SUB、SBC 和 RSB
ADD	加法	ADD、ADC、SUB、SBC 和 RSB
ADDW	加法	ADD、ADC、SUB、SBC 和 RSB
AND	逻辑“与”	AND，ORR，EOR，BIC 和 ORN
ASR	算术右移	ASR，LSL，LSR，ROR 和 RRX
BIC	位清零	AND，ORR，EOR，BIC 和 ORN
CLZ	计算前导零数目	CLZ
CMN	与负值比较	CMP 和 CMN
CMP	比较	CMP 和 CMN
EOR	异或	AND，ORR，EOR，BIC 和 ORN
LSL	逻辑左移	ASR，LSL，LSR，ROR 和 RRX
LSR	逻辑右移	ASR，LSL，LSR，ROR 和 RRX
MOV	移动	MOV 和 MVN
MOVT	移动到顶部	MOVT
MOVW	移动 16 位常数	MOV 和 MVN
MVN	取反移动	MOV 和 MVN
ORN	逻辑“或非”	AND，ORR，EOR，BIC 和 ORN
ORR	逻辑“或”	AND，ORR，EOR，BIC 和 ORN
RBIT	反转位	REV，REV16，REVSH 和 RBIT
REV	反转字中的字节顺序	REV，REV16，REVSH 和 RBIT
REV16	反转每个半字中的字节顺序	REV，REV16，REVSH 和 RBIT
REVSH	反转低半字中的字节顺序，并将符号扩展	REV，REV16，REVSH 和 RBIT
ROR	向右循环移	ASR，LSL，LSR，ROR 和 RRX
RRX	带扩展向右循环移	ASR，LSL，LSR，ROR 和 RRX
RSB	反向减法	ADD、ADC、SUB、SBC 和 RSB
SBC	带进位减法	ADD、ADC、SUB、SBC 和 RSB
SUB	减法	ADD、ADC、SUB、SBC 和 RSB
SUBW	减法	ADD、ADC、SUB、SBC 和 RSB
TEQ	相等测试	TST 和 TEQ
TST	测试	TST 和 TEQ

附录表 1-8 数据处理指令

附录1. 2. 5. 1 ADD、ADC、SUB、SBC 和 RSB

加法、带进位加法、减法、带进位减法和反向减法。

语法

op{S}{cond} {Rd,} Rn，Operand 2
op{cond} {Rd,} Rn，#imm12 ；仅 ADD 和 SUB

其中：

op 为下列项之一：

ADD：加法。

ADC：带进位加法。

SUB：减法。

RSB：反向减法。

S：是一个可选后缀。如果指定了 S，则会更新操作结果的条件代码标志，见“条件执行”小节。

cond：是一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。如果 Rd 被省略，则目标寄存器为 Rn。

Rn 为存放第一个操作数的寄存器。

Operand2 为一个灵活的第二操作数。此选项详细说明请参见“灵活的第二操作数”小节。

imm12 可为 0~4095 范围内的任意值。

操作

ADD 指令将 Rn 中的值与 Operand2 或 imm12 中的值相加。

ADC 指令将 Rn 中的值与 Operand2 相加（带进位标志）。

SUB 指令从 Rn 中的值减去 Operand2 或 imm12 的值。

SBC 指令从 Rn 中的值减去 Operand2 的值。如果进位标志被清除，则结果减 1。

RSB 指令将 Operand2 的值减去 Rn 中的值。这是很有用的，因为有了该指令，Operand2

的选项范围就会更大。

可以使用 ADC 和 SBC 进行合成多字运算，请参见“多字算术操作示例”小节。

亦可参见“ADR”小节。

注：ADDW 与使用 imm12 操作数的 ADD 语法等效。SUBW 与使用 imm12 操作数的 SUB 语法等效。

限制

◇ Operand2 不得为 SP 且不得为 PC；

◇ 仅在 ADD 和 SUB 中，Rd 可以是 SP，并有附加的限制：

- Rn 必须也是 SP；
- Operand2 中的任何移位都限于用 LSL 最多移 3 位；

◇ 仅在 ADD 和 SUB 中，Rn 可以是 SP；

◇ 仅在 cond 指令中，Rd 可以是 PC，且其中：

- 不得指定 S 后缀；
- Rm 不得为 PC 且不得为 SP；
- 如果指令是有条件的，则必须是 IT 块中的最后一条指令。

◇ cond 指令有异常时，仅在 ADD 和 SUB 中，Rn 可以是 SP，且仅在下列附加限制下：

- 不得指定 S 后缀；
- 第二操作数必须是 0~4095 范围内的一个常数。

◇ 当使用 PC 进行加法或减法时，计算前要将 PC 的位[1:0]舍入到 b00，从而让计算的基 址按字对齐。

◇ 如果想产生某条指令的地址，则必须根据 PC 的值调整常数。ARM 建议使用 ADR 指令代 替 Rn 等于 PC 的 ADD 或 SUB 指令，因为汇编器会自动为 ADR 指令计算正确的常数

当 cond 指令中的 Rd 为 PC 时：

◇ 写入 PC 的值的位[0]被忽略；

◇ 发生到将该值的位[0]强制为 0 生成的地址的跳转。

条件标志

如果指定了 S，这些指令根据结果更新 N、Z、C 和 V 标志。

示例

```
ADD   R2, R1, R3
SUBS R8, R6, #240      ;Sets the flags on the result
RSB   R4, R4, #1280    ; Subtracts contents of R4 from 1280
ADCHI   R11, R0, R3    ; Only executed if C flag set and Z
                       ; flag clear
```

多字算术操作示例

64 位加法中的两条指令将 R2 和 R3 中包含的一个 64 位整数与 R0 和 R1 中包含的另 一个 64 位整数相加，并将结果存入 R4 和 R5。

多字值不一定非要使用连续的寄存器。96 位减法中的指令从 R6、R2 和 R8 中包含的一个 96 位整数中减去 R9、R1 和 R11 中包含的一个 96 位整数。示例将结果存储在 R6、R9 和 R2 中。

64 位加法：

```
ADDS R4, R0, R2    ; add the least significant words
ADC   R5, R1, R3    ; add the most significant words with carry
```

96 位减法：

```
SUBS R6, R6, R9    ; subtract the least significant words
SBCS R9, R2, R1    ; subtract the middle words with carry
SBC  R2, R8, R11   ; subtract the most significant words with carry
```

附录1. 2. 5. 2 AND，ORR，EOR，BIC 和 ORN

逻辑“与”、“或”、“异或”、位清零和“或非”。

语法

op{S}{cond} {Rd,} Rn，Operand2

其中：

op 为下列项之一：

AND：逻辑“与”。

ORR：逻辑“或”，或置位。

EOR：逻辑“异或”。

BIC：逻辑“与非”，或位清零。

ORN：逻辑“或非”。

S：是一个可选后缀。如果指定了 S，则会更新操作结果的条件代码标志，见“条件执行”小节。

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。

Rn 为保存第一个操作数的寄存器。

Operand2 是灵活的第二操作数。各选项详细说明请参见“灵活的第二操作数”小节。

操作

AND、EOR 和 ORR 指令可对 Rn 和 Operand2 中的值按位进行“与”、“异或”和“或”操作。

BIC 指令对 Rn 中的位和 Operand2 值中相应位的补码进行“与”操作。

ORN 指令对 Rn 中的位和 Operand2 值中相应位的补码进行“或”操作。

限制

不使用 SP 且不使用 PC。

条件标志

如果指定了 S，则这些指令：

◇ 根据结果更新 N 和 Z 标志；

◇ 可以在 Operand2 运算期间更新 C 标志，见“灵活的第二操作数”小节；

◇ 不影响 V 标志。

示例

```
AND R9, R2, #0xFF00
ORREQ R2, R0, R5
ANDS R9, R8, #0x19
EORS R7, R11, #0x18181818
BIC R0, R1, #0xab
ORN R7, R11, R14, ROR #4
ORNS R7, R11, R14, ASR #32
```

附录1. 2. 5. 3 ASR，LSL，LSR，ROR 和 RRX

算术右移、逻辑左移、逻辑右移、向右循环移和带扩展向右循环移。

语法

```
op{S}{cond} Rd，Rm，Rs
op{S}{cond} Rd，Rm，#n
RRX{S}{cond} Rd，Rm
```

其中：

op 为下列项之一：

ASR：算术右移。

LSL：逻辑左移。

LSR：逻辑右移。

ROR：向右循环移。

S：是一个可选后缀。如果指定了 S，则会更新操作结果的条件代码标志，见“条件执行”小节。

Rd 为目标寄存器。

Rm 为存储将被移位数值的寄存器。

Rs 为存放移位长度的寄存器，所存放的移位长度应用于 Rm 中的值。仅使用最低有效字节，范围为 0~255。

n 为移位长度。不同指令的移位长度的范围为：

ASR：移位长度为 1~32；

LSL：移位长度为 0~31；

LSR：移位长度为 1~32；

ROR：移位长度为 1~31。

对于 LSL{S}{cond} Rd，Rm，#0， 首选语法为 MOV{S}{cond} Rd，Rm。

操作

ASR、LSL、LSR 和 ROR 将寄存器 Rm 中的位左移或右移，移位的数量由常数 n 或寄存器 Rs 指定。

RRX 将寄存器 Rm 中的位向右移 1 位。

在所有这些指令中，结果写到 Rd，但寄存器 Rm 中的值保持不变。欲详细了解不同指令产生的结果，请参见“移位操作”小节。

限制

不使用 SP 且不使用 PC。

条件标志

如果指定了 S，则：

◇ 这些指令根据结果更新 N 和 Z 标志；

◇ 如果移位长度为 0，则不会影响 C 标志。否则，C 标志会更新为移出的最后一位见 “移位操作 ”小节。

示例

```
ASR   R7, R8, #9   ;    Arithmetic shift right by 9 bits
LSLS  R1, R2, #3   ;    Logical shift left by 3 bits with flag update
LSR   R4, R5, #6   ;    Logical shift right by 6 bits
ROR   R4, R5, R6   ;    Rotate right by the value in the bottom byte of  R6
RRX   R4, R5       ;    Rotate right with extend
```

附录1. 2. 5. 4 CLZ

计算前导零数目。

语法

CLZ{cond} Rd，Rm

其中：

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。

Rm 为操作数寄存器。

操作

CLZ 指令对 Rm 中值中的前导零数目进行计算，并将结果返回到 Rd 中。如果未在源寄存器中设置任何位，则该结果值为 32，如果位[31]被置位，则结果值为零。

限制

不使用 SP 且不使用 PC。

条件标志

此指令不改变标志。

示例

```
CLZ     R4,R9
CLZNE   R2,R3
```

附录1. 2. 5. 5 CMP 和 CMN

比较和与负值比较。

语法

CMP{cond} Rn，Operand2
CMN{cond} Rn，Operand2

其中：

cond 为一个可选条件代码，见“条件执行”小节。

Rn 为存放第一个操作数的寄存器。

Operand2 是灵活的第二操作数。各选项详细说明请参见关于灵活的第二操作数的 内容。

操作

这些指令可比较寄存器中的值与 Operand2 比较。它们更新结果的条件标志，但不将结果写到寄存器。

CMP 指令从 Rn 中的值减去 Operand2 的值。这与 SUBS 指令功能相同，只是结果被丢弃。

CMN 指令给 Rn 中的值加上 Operand2 的值。这与 ADDS 指令功能相同，只是结果被丢弃。

限制

在这些指令中：

◇ 不使用 PC；

◇ Operand2 不得为 SP。

条件标志

这些指令根据结果更新 N、Z、C 和 V 标志。

示例

```
CMP   R2，R9
CMN   R0，#6400
CMPGT    SP，R7，LSL #2
```

附录1. 2. 5. 6 MOV 和 MVN

移动和取反移动。

语法

MOV{S}{cond} Rd，Operand2
MOV{cond} Rd，#imm16
MVN{S}{cond} Rd，Operand2

其中：

S 为一个可选后缀。如果指定了 S，则会更新操作结果的条件代码标志，见“条件执行”小节。

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。

Operand2 是一个灵活的第二操作数。各选项详细说明请参见关于灵活的第二操作 数的内容。

imm16 可为 0~65535 范围内的任意值。

操作

MOV 指令可将 Operand2 的值复制到 Rd 中。

当 MOV 指令中的 Operand2 为一个带移位而非 LSL #0 的寄存器时，首选语法为相应的移位指令：

◇ 对于 MOV{S}{cond} Rd，Rm，ASR #n，ASR{S}{cond} Rd，Rm，#n 为首选语法
◇ 对于 MOV{S}{cond} Rd，Rm，LSL #n，LSL{S}{cond} Rd，Rm，#n 为首选语法（n if !=0）
◇ 对于 MOV{S}{cond} Rd，Rm，LSR #n，LSR{S}{cond} Rd，Rm，#n 为首选语法
◇ 对于 MOV{S}{cond} Rd，Rm，ROR #n，ROR{S}{cond} Rd，Rm，#n 为首选语法
◇ 对于 MOV{S}{cond} Rd，Rm，RRX，RRX{S}{cond} Rd，Rm 为首选语法。

另外，作为移位指令的同义词，MOV 指令还允许其他形式的 Operand2：

◇ MOV{S}{cond} Rd，Rm，ASR Rs 与 SR{S}{cond} Rd，Rm，Rs 同义
◇ MOV{S}{cond} Rd，Rm，LSL Rs 与 LSL{S}{cond} Rd，Rm，Rs 同义
◇ MOV{S}{cond} Rd，Rm，LSR Rs 与 LSR{S}{cond} Rd，Rm，Rs 同义
◇ MOV{S}{cond} Rd，Rm，ROR Rs 与 ROR{S}{cond} Rd，Rm，Rs 同义

见“ASR，LSL，LSR，ROR 和 RRX” 小节。

MVN 指令先获取 Operand2 的值，然后对该值按位进行逻辑“非”操作，并将结果存入 Rd。

注：MOVW 指令提供与 MOV 相同的功能，但仅限于使用 imm16 操作数。

限制

仅在 MOV 指令中可使用 SP 和 PC，并有下列限制：

◇ 第二操作数必须为一个未带移位的寄存器；

◇ 不得指定 S 后缀。

当 MOV 指令中 Rd 为 PC 时：

◇ 写入 PC 的值的位[0]被忽略；

◇ 发生到强制使该值的位[0]为零产生的地址的跳转。

注：尽管可以将 MOV 作为一条跳转指令，但 ARM 强烈建议使用 BX 或 BLX 指令进行跳转，以保证软件对 ARM 指令集的可移植性。

条件标志

如果指定了 S，则这些指令：

◇ 根据结果更新 N 和 Z 标志；

◇ Operand2 运算过程中可以更新 C 标志，见“灵活的第二操作数”小节；

◇ 不影响 V 标志。

示例

```
MOVS    R11, #0x000B     ; Write value of 0x000B to R11, flags get updated
MOV     R1, #0xFA05      ; Write value of 0xFA05 to R1, flags are not updated
MOVS    R10, R12         ; Write value in R12 to R10, flags get updated
MOV     R3, #23          ; Write value of 23 to R3
MOV     R8, SP           ; Write value of stack pointer to R8
MVNS    R2, #0xF         ; Write value of 0xFFFFFFF0 (bitwise inverse of 0xF)
                         ; to the R2 and update flags
```

附录1. 2. 5. 7 MOVT

移动到顶部。

语法

MOVT{cond} Rd，#imm16

其中：

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。

imm16 为一个 16 位立即常数。

操作

MOVT 可将一个 16 位立即值 imm16 写入其目标寄存器的高半字，Rd[31:16]中。该写操作不会影响 Rd[15:0]。

可使用 MOV、MOVT 指令对生成任意的 32 位常数。

限制

Rd 不得为 SP 且不得为 PC。

条件标志

此指令不改变标志。

示例

```
MOVT   R3, #0xF123          ; Write 0xF123 to upper halfword of R3, lower halfword
                            ; and APSR are unchanged
```

附录1. 2. 5. 8 REV，REV16，REVSH 和 RBIT

在字或半字内反转字节或位的顺序。

语法

op{cond} Rd，Rn

其中：

op 为下列项之一：

REV　　反转字中的字节顺序。

REV16　　独立地反转每个半字中的字节顺序。

REVSH　　反转低半字中的字节顺序，并将符号扩展

RBIT　　反转 32 位字中的位的顺序。

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。

Rn 为存放操作数的寄存器。

操作

可使用这些指令来更改数据的字节序（endianness）：

REV：将 32 位大端数据转换为小端数据，或将 32 位小端数据转换为大端数据。

REV16 将 16 位大端数据转换为小端数据，或将 16 位小端数据转换为大端数据。

REVSH 可完成以下任一转换：

将 16 位带符号大端数据转换为 32 位带符号小端数据；

将 16 位带符号小端数据转换为 32 位带符号大端数据。

限制

不使用 SP 且不使用 PC。

条件标志

这些指令不改变标志。

示例

```
REV     R3,    R7       ; Reverse byte order of value in R7 and write it to R3
REV16   R0,  R0         ;Reverse byte order of each 16-bit halfword in R0
```

```
REVSH    R0, R5      ;Reverse Signed Halfword
REVHS    R3, R7      ;Reverse with Higher or Same condition
RBIT  R7, R8         ; Reverse bit order of value in R8 and write the result to R7
```

附录1. 2. 5. 9 TST 和 TEQ

位测试和相等测试。

语法

TST{cond} Rn，Operand2

TEQ{cond} Rn，Operand2

其中：

cond 为一个可选条件代码，见“条件执行”小节。

Rn 为存放第一个操作数的寄存器。

Operand2 是一个灵活的第二操作数。各选项详细说明请参见“灵活的第二操作数”小节。

操作

这些指令可利用 Operand2 来测试寄存器中的值。它们会更新结果的条件标志，但不会将结果写入任何寄存器中。

TST 指令对 Rn 中的值和 Operand2 的值按位进行“与”操作。除了结果会被丢弃以外，这 与 ANDS 指令功能相同。

如要测试 Rn 的某个位是否为 0 或 1，使用带一个 Operand2 常数的 TST 指令，会使该位置位 为 1，并清零所有其他位。

TEQ 指令对 Rn 中的值和 Operand2 的值按位进行“异或”操作。除了结果会被丢弃以外，这与 EORS 指令功能相同。

使用 TEQ 指令可在不影响 V 或 C 标志的情况下，测试两个值是否相等。

TEQ 也可用来测试值的符号。比较完毕后，两个操作数的符号位的逻辑“异或”运算结果将成为 N 标志。

限制

不得使用 SP 且不得使用 PC。

条件标志

这些指令：

◇ 根据结果更新 N 和 Z 标志；

◇ 在 Operand2 运算过程中能够更新 C 标志；

◇ 不影响 V 标志。

示例

```
TST    R0, #0x3F8          ; Perform bitwise AND of R0 value to 0x3F8,
                           ; APSR is updated but result is discarded
TEQEQ     R10, R9          ; Conditionally   test if value in R10 is equal to
                           ; value in R9, APSR is updated but result is discarded
```

附录1. 2. 6 乘法和除法指令

下表列出了乘法和除法指令：

助记符	简要描述	参见（已改为对应章节名称）
MLA	乘加，结果为 32 位	MUL、MLA 和 MLS
MLS	乘减，结果为 32 位	MUL、MLA 和 MLS
MUL	乘法，结果为 32 位	MUL、MLA 和 MLS
SDIV	有符号除法	SDIV 和 UDIV
SMLAL	有符号乘加（32x32+64），结果为 64 位	UMULL，UMLAL，SMULL 和 SMLAL
SMULL	有符号乘法（32 x 32），结果为 64 位	UMULL，UMLAL，SMULL 和 SMLAL
UDIV	无符号除法	SDIV 和 UDIV
UMLAL	无符号乘加（32x32+64），结果为 64 位	UMULL，UMLAL，SMULL 和 SMLAL
UMULL	无符号乘法（32 x 32），结果为 64 位	UMULL，UMLAL，SMULL 和 SMLAL

附录表 1-9 乘法和除法指令

附录1. 2. 6. 1 MUL、MLA 和 MLS

使用 32 位操作数的乘法、乘加和乘减，并产生一个 32 位的结果。

语法

MUL{S}{cond} {Rd,} Rn，Rm ；乘法
MLA{cond} Rd，Rn，Rm，Ra ；乘加
MLS{cond} Rd，Rn，Rm，Ra ；乘减

其中：

cond 为一个可选条件代码，见“条件执行”小节。

S 是一个可选后缀。如果指定了 S，则会更新操作结果的条件代码标志，见“条件执行”小节。

Rd 为目标寄存器。如果 Rd 被省略，则目标寄存器为 Rn。

Rn，Rm 为存放乘数的寄存器。

Ra 为存放被加数或被减数的寄存器。

操作

MUL 指令可将 Rn 和 Rm 中的值相乘，并将所得结果的低 32 位存入 Rd。

MLA 指令可将 Rn 和 Rm 中的值相乘，然后再将乘积与 Ra 的值相加，最后将所得结果的低 32 位存入 Rd

MLS 指令可将 Rn 和 Rm 中的值相乘，然后再从 Ra 中的值减去乘积，最后将所得结果的低 32 位存入 Rd。

这些指令的结果与操作数是有符号还是无符号无关。

限制

在这些指令中，不得使用 SP 且不得使用 PC。 如果 MUL 指令使用 S 后缀：

◇ Rd、Rn 和 Rm 必须全部都在 R0~R7 范围内；

◇ Rd 必须与 Rm 相同；

◇ 不得使用 cond 后缀。

条件标志

如果指定了 S，则 MUL 指令：

◇ 根据结果更新 N 和 Z 标志；

◇ 不影响 C 和 V 标志。

示例

```
MUL   R10, R2, R5          ; Multiply, R10 = R2 x R5
MLA   R10, R2, R1, R5      ; Multiply with accumulate,  R10 = (R2 x R1) + R5
MULS      R0, R2, R2       ; Multiply with flag update, R0 = R2 x R2
MULLT     R2, R3, R2       ; Conditionally  multiply, R2 = R3 x R2
MLS   R4, R5, R6, R7       ; Multiply with subtract, R4 = R7 - (R5 x R6)
```

附录1. 2. 6. 2 UMULL，UMLAL，SMULL 和 SMLAL

使用 32 位操作数的有符号长乘法和无符号长乘法，产生一个 64 位结果。

语法

op{cond} RdLo，RdHi，Rn，Rm

其中：

op 为下列项之一：

UMULL：无符号长乘法。

UMLAL：无符号长乘法，带累加。

SMULL：有符号长乘法。

SMLAL：有符号长乘法，带累加。

cond 为一个可选条件代码，见“条件执行”小节。

RdHi、RdLo 为目标寄存器。对于 UMLAL 和 SMLAL，它们还用于存放累加值。

Rn、Rm 为存放操作数的寄存器。

操作

UMULL 指令会将 Rn 和 Rm 中的值解释为无符号整数。它会先求这两个整数的乘积，然后将结果的低 32 位存入 RdLo，高 32 位存入 RdHi。

UMLAL 指令会将 Rn 和 Rm 中的值解释为无符号整数。它会先求这两个整数的乘积，然后将 64 位结果与 RdHi 和 RdLo 中的 64 位无符号整数相加，并将结果写回到 RdHi 和 RdLo。

SMULL 指令会将 Rn 和 Rm 中的值解释为有符号整数的二进制补码。它会先求这两个整数的乘积，然后将结果的低 32 位存入 RdLo，高 32 位存入 RdHi。

SMLAL 指令会将 Rn 和 Rm 中的值解释为有符号整数的二进制补码。它会先求这两个

整数的乘积，然后将 64 位结果与 RdHi 和 RdLo 中的 64 位有符号整数相加，并将结果写回到 RdHi 和 RdLo。

限制

在这些指令中：

◇ 不得使用 SP 且不得使用 PC；

◇ RdHi 和 RdLo 必须为不同的寄存器。

条件标志

这些指令不影响条件代码标志。

示例

```
UMULL    R0, R4, R5, R6     ; Unsigned (R4,R0) = R5 x R6
SMLAL    R4, R5, R3, R8     ; Signed (R5,R4) = (R5,R4) + R3 x R8
```

附录1. 2. 6. 3 SDIV 和 UDIV

有符号除法和无符号除法。

语法

```
SDIV{cond} {Rd,} Rn，Rm
UDIV{cond} {Rd,} Rn，Rm
```

其中：

cond 为一个可选条件代码，见 “条件执行”小节。

Rd 为目标寄存器。如果 Rd 被省略，则目标寄存器为 Rn。

Rn 为存放被除数的寄存器。

Rm 为存放除数的寄存器。

操作

SDIV 对 Rn 中的值用 Rm 中的值进行有符号的整数除法。

UDIV 对 Rn 中的值用 Rm 中的值进行无符号的整数除法。

对于这两条指令来说，如果 Rn 中的值不能被 Rm 中的值整除，则结果向零舍入。

限制

不得使用 SP 且不得使用 PC。

条件标志

这些指令不改变标志。

示例

```
SDIV   R0, R2, R4     ; Signed divide, R0 = R2/R4
UDIV   R8, R8, R1     ; Un sig ned    divide, R8 = R8/R1
```

附录1. 2. 7 饱和指令

本节介绍饱和指令，SSAT 和 USAT。

附录1. 2. 7. 1 SSAT 和 USAT

有符号饱和到任何位位置和无符号饱和到任何位位置，可选择在饱和前进行移位。

语法

op{cond} Rd，#n，Rm {，shift #s}

其中：

op 为下列项之一：

SSAT 可将一个有符号值饱和到一个有符号范围内。

USAT 可将一个有符号值饱和到一个无符号范围内。

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。

n 指定要饱和到的位位置：

◇ 对于 SSAT， n 的范围为 1~32。

◇ 对于 USAT， n 的范围为 0~31。

Rm 为存放待饱和值的寄存器。

shift #s 为一个可选的饱和前的移位，应用于 Rm。它必须为下列项之一：

ASR #s：其中，s 的范围为 1~31；

LSL #s：其中，s 的范围为 0~31。

操作

这些指令饱和到一个有符号或无符号的 n 位值。

SSAT 指令会先进行指定的移位，然后将结果饱和到有符号范围$-2^{n-1} \leqslant x \leqslant 2^{n-1}-1$。

USAT 指令会先进行指定的移位，然后将结果饱和到无符号范围 $0 \leqslant x \leqslant 2^{n}-1$。

对于使用 SSAT 的有符号 n 位饱和，这意味着：

◇ 如果待饱和的值小于-2^{n-1}，则返回结果为-2^{n-1}；

◇ 如果待饱和的值大于 $2^{n-1}-1$，则返回结果为 $2^{n-1}-1$；

◇ 否则，返回结果与待饱和的值相同。

对于使用 USAT 的无符号 n 位饱和，这意味着：

◇ 如果待饱和的值小于 0，则返回结果为 0；

◇ 如果待饱和的值大于 $2^{n}-1$，则返回结果为 $2^{n}-1$；

◇ 否则，返回结果与待饱和的值相同。

如果返回结果与待饱和的值不同，则称为饱和。如果发生了饱和，该指令在 APSR 中

将 Q 标志设为 1。否则，它将保持 Q 标志不变。如要清零 Q 标志，必须使用 MSR 指令，见“MSR”小节。

如要读取 Q 标志的状态，可使用 MRS 指令，见“MRS”小节。

限制

不得使用 SP 且不得使用 PC。

条件标志

这些指令不影响条件代码标志。 如果发生饱和，这些指令将 Q 标志设置为 1。

示例

```
SSAT    R7, #16, R7, LSL #4    ; Logical shift left value in R7 by 4, then
                               ; saturate it as a signed 16-bit value and
                               ; write it back to R7
USATNE  R0, #7, R5             ; Conditionally  saturate value in R5 as an
                               ; unsigned 7 bit value and write it to R0
```

附录1. 2. 8 位域指令

下表列出了对寄存器中位的相邻集（或称位域）操作的指令：

助记符	简要描述	参见（已改为对应章节名称）
BFC	位域清零	BFC 和 BFI
BFI	位域插入	BFC 和 BFI
SBFX	有符号位域提取	SBFX 和 UBFX
SXTB	有符号扩展一个字节	SXT 和 UXT
SXTH	有符号扩展一个半字	SXT 和 UXT
UBFX	无符号位域提取	SBFX 和 UBFX
UXTB	用零扩展一个字节	SXT 和 UXT
UXTH	用零扩展一个半字	SXT 和 UXT

附录表 1-10 组合和分离指令

附录1. 2. 8. 1 BFC 和 BFI

位域清零和位域插入。

语法

BFC{cond} Rd，#lsb，#width

BFI{cond} Rd，Rn，#lsb，#width

其中：

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。

Rn 为源寄存器。

lsb 为位域最低有效位的位置。lsb 必须在 0~31 范围之内。

width 为位域宽度，必须在 1~32-lsb 范围内。

操作

BFC 清零寄存器中的一个位域。它从低位位置 lsb 开始，清零 Rd 中的 width 位。Rd 中的其他位不变。

BFI 将一个位域从一个寄存器复制到另一个寄存器。它用 Rn 从位[0]开始的 width 位替换 Rd 中从低位位置 lsb 开始的 width 位。Rd 中的其他位不变。

限制

不得使用 SP 且不得使用 PC。

条件标志

这些指令不影响标志。

示例

```
BFC   R4, #8, #12          ; Clear bit 8 to bit 19 (12 bits) of R4 to 0
BFI   R9, R2, #8, #12      ; Replace bit 8 to bit 19 (12 bits) of R9 with
                           ; bit 0 to bit 11 from R2
```

附录1. 2. 8. 2 SBFX 和 UBFX

有符号位域提取和无符号位域提取。

语法

SBFX{cond} Rd，Rn，#lsb，#width
UBFX{cond} Rd，Rn，#lsb，#width
其中：

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。

Rn 为源寄存器。

lsb 为位域中的最低有效位的位置。lsb 必须在 0~31 范围内。

width 为位域宽度，必须在 1~32-lsb 范围内。

操作

SBFX 从一个寄存器中提取一个位域，用符号将其扩展到 32 位，并将结果写到目标寄存器。

UBFX 从一个寄存器中提取一个位域，用零将其扩展到 32 位，并将结果写到目标寄存器。

限制

不得使用 SP 且不得使用 PC。

条件标志

这些指令不影响标志。

示例

```
SBFX     R0, R1, #20, #4    ; Extract bit 20 to bit 23 (4 bits) from R1 and sign
                            ; extend to 32 bits and then write the result to R0.
UBFX     R8, R11, #9, #10   ; Extract bit 9 to bit 18 (10 bits) from R11 and zero
                            ; extend to 32 bits and then write the result to R8
```

附录1. 2. 8. 3 SXT 和 UXT

符号扩展和零扩展。

语法

SXTextend{cond} {Rd,} Rm {，ROR #n}
UXTextend{cond} {Rd}，Rm {，ROR #n}
其中：

extend 为下列项之一：

B：将一个 8 位值扩展为 32 位值。

H：将一个 16 位值扩展为 32 位值。

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。

Rm 为存放待扩展值的寄存器。

ROR #n 为下列项之一：

ROR #8：将 Rm 中的值向右循环移 8 位。

ROR #16：将 Rm 中的值向右循环移 16 位。

ROR #24： 将 Rm 中的值向右循环移 24 位。

如果 ROR #n 被省略，则不执行循环移位。

操作

这些指令执行以下操作：

1. 将 Rm 中的值向右循环移 0、8、16 或 24 位。
2. 从结果值中提取位：

◇ SXTB 提取位[7:0]，并用符号扩展到 32 位。

◇ UXTB 提取位[7:0]，并用零扩展到 32 位。

◇ SXTH 提取位[15:0]，并用符号扩展到 32 位。

◇ UXTH 提取位[15:0]，并用零扩展到 32 位。

限制

不得使用 SP 且不得使用 PC。

条件标志

这些指令不影响标志。

示例

```
SXTH    R4, R6, ROR #16    ; Rotate R6 right by 16 bits, then obtain the lower
                           ; halfword of the result and then sign extend to
                           ; 32 bits and write the result to R4.
UXTB    R3, R10            ; Extract lowest byte of the value in R10 and zero
                           ; extend it, and write the result to R3.
```

附录1. 2. 9 跳转和控制指令

下表列出了跳转和控制指令：

助记符	简要描述	参见（已改为对应章节名称）
B	跳转	B、BL、BX 和 BLX
BL	带链接的跳转	B、BL、BX 和 BLX
BLX	带链接的跳转并切换指令集	B、BL、BX 和 BLX
BX	跳转并切换指令集	B、BL、BX 和 BLX
CBNZ	比较，结果非零则跳转	CBZ 和 CBNZ
CBZ	比较，结果为零则跳转	CBZ 和 CBNZ
IT	If-Then	IT
TBB	表跳转字节	TBB 和 TBH
TBH	表跳转半字	TBB 和 TBH

附录表 1-11 跳转和控制指令

附录1. 2. 9. 1 B、BL、BX 和 BLX

跳转指令。

语法

B{cond} label
BL{cond} label
BX{cond} Rm
BLX{cond} Rm

其中：

B 为跳转（立即）。

BL 为带链接跳转（立即）。

BX 为跳转并切换指令集（寄存器）。

BLX 为带链接的跳转并切换指令集（寄存器）。

cond 为一个可选条件代码，见“条件执行”小节。

label 为一个相对 PC 表达式。见“相对 PC 的表达式”小节。

Rm 为一个寄存器，指示要跳转到的目标地址。Rm 中值的位[0]必须为 1，但跳转到的目标地址是通过将位[0]变为 0 生成的。

操作

所有这些指令都会引发跳转，或跳转到 label，或跳转到 Rm 所指示的地址处。此外：

◇ BL 和 BLX 指令可将下一条指令的地址复制到链接寄存器（LR）R14 中。

◇ 如果 Rm 位[0]为 0，BX 和 BLX 指令会导致 UsageFault 异常。

Bcond 标签是唯一一个既可在 IT 块内又可在 IT 块外的条件指令。所有其他跳转指令在 IT 块内都必须是有条件的，在 IT 块外必须是无条件的。

下表列出了各种跳转指令的范围。

指令	跳转范围
B label	-16MB~+16MB
Bcond label (IT 块外)	-1MB~+1MB
Bcond label (IT 块内)	-16MB~+16MB
BL{cond} label	-16MB~+16MB
BX{cond} Rm	寄存器中的任意值
BLX{cond} Rm	寄存器中的任意值

附录表 1-12 跳转范围

注：要获得最大跳转范围，可能需要使用.W 后缀。参见 “指令宽度选择”小节。

限制

限制如下：

◇ BLX 指令中不使用 PC；

◇ 对于 BX 和 BLX，Rm 的位[0]必须为 1 才能正确执行指令，但会跳转到 Rm 的位[0]为 0 所生成的目标地址。

◇ 如果这些指令中的任意一个处于一个 IT 块内，则它必须是 IT 块中的最后一条指令。

Bcond 是唯一一条不需要在 IT 块内的条件指令。但是，当它在 IT 块内时，具有较大的跳转范围。

条件标志

这些指令不改变标志。

示例

```
B      loopA          ; Branch to loopA
BLE    ng             ; Conditionally branch to label ng
B.W    target         ; Branch to target within 16MB range
BEQ    target         ; Conditionally branch to target
BEQ.W      target     ; Conditionally branch to target within 1MB
BL          funC      ; Branch with link (Call) to function funC, return address
                      ; stored in LR
BX     LR             ; Return from function call
BXNE       R0         ; Conditionally branch to address stored in R0
BLX    R0             ; Branch with link and exchange (Call) to a address stored
                       ; in R0
```

附录1. 2. 9. 2 CBZ 和 CBNZ

比较，结果为零则跳转；比较，结果非零则跳转。

语法

CBZ Rn ，label
CBNZ Rn，label
其中：

Rn 为存放操作数的寄存器。

label 为跳转目标。

操作

可以使用 CBZ 或 CBNZ 指令避免改变条件代码标志并减少指令数目。

除了不改变条件标志外，CBZ Rn， label 与下列指令序列功能相同：

```
CMP   Rn，#0
      BEQ     label
```

除了不改变条件标志外，CBNZ Rn，label 与下列指令序列功能相同：

```
CMP   Rn，#0
      BNE     label
```

限制

限制如下：

◇ Rn 必须在 R0~R7 范围内；

◇ 跳转目标必须在指令之后的 4 到 130 个字节之内；

◇ 这些指令一定不能在一个 IT 块内使用。

条件标志

这些指令不改变标志。

示例

```
CBZ   R5, target          ; Forward branch if R5 is zero
CBNZ      R0, target      ; Forward branch if R0 is not zero
```

附录1. 2. 9. 3 IT

If-Then（IT）条件指令。

语法

IT{x{y{z}}} cond

其中：

◇ x 指定 IT 块中第二条指令的条件开关。

◇ y 指定 IT 块中第三条指令的条件开关。

◇ z 指定 IT 块中第四条指令的条件开关。

◇ cond 指定 IT 块中第一条指令的条件。

IT 块中第二、第三和第四条指令的条件开关可以是下列项之一：

◇ T：Then。将条件 cond 应用于指令。

◇ E：Else。将 cond 的相反条件应用于指令。

注：在 IT 指令中，cond 可以使用 AL（“总是（满足）”条件）。如果这样，则 IT 块中所有指令都必须是无条件的，且每个 x、y 和 z 都必须为 T（不能是 E）或省略。

操作

IT 指令最多可以构成 4 条后续条件指令。条件可以全部相同，或一部分条件可以是其他条件的逻辑反相。IT 指令之后的条件指令构成了 IT 块。

IT 块中的指令，包括所有跳转，都必须在其语法的{cond}部分指定条件。

注：汇编器也可以自动地为条件指令生成所需 IT 指令，从而不需要自己编写。详细情况请参见所用汇编器的文档。

IT 块中的 BKPT 指令总会得到执行，即使无法满足其条件也是如此。

在一条 IT 指令和相应的 IT 块之间（或一个 IT 块内）都会发生异常。如果发生此异常，则会进入到相应的异常处理程序，同时将适用的返回信息存入 LR 和压入堆栈的 PSR 中。

可像平常一样利用异常返回指令从异常中返回，IT 块将会继续正常执行。这是 PC 修改指令跳转到一个 IT 块中的指令的唯一方法。

限制

在 IT 块中不允许使用下列指令：

◇ IT

◇ CBZ 和 CBNZ

◇ CPSID 和 CPSIE

◇ MOVS.N Rd,Rm

使用 IT 块的其他限制有：

◇ 跳转指令或修改 PC 的任何指令必须处于 IT 块外，或者必须是 IT 块中的最后一

条指令。这些指令有：

- ADD PC、PC、Rm
- MOV PC、Rm
- B、BL、BX、BLX
- LDM、LDR 或任何写入 PC 的 POP 指令
- TBB 和 TBH

◇ 无法跳转到 IT 块内的任何指令，除非是从一个异常处理程序返回时。

◇ 除 Bcond 外，所有其他条件指令都必须在 IT 块内。Bcond 可以在 IT 块外，也可以在 IT 块内，但它在 IT 块内时具有较大的跳转范围。

◇ IT 块内每条指令都必须指定一个条件代码后缀。这些条件代码后缀可以是相同的，也可以是与块中其他指令的条件的逻辑反相。

注：所用的汇编器可能对 IT 块的使用有额外的限制，例如禁止在块内使用汇编器指令。

条件标志

此指令不改变标志。

示例

```
ITTE    NE              ; Next 3 instructions    are conditional
ANDNE   R0, R0, R1      ; ANDNE does not update condition flags
ADDSNE  R2, R2, #1      ; ADDSNE updates condition flags
MOVEQ   R2, R3          ; Conditional move
CMP     R0, #9          ; Convert R0 hex value (0 to 15) into ASCII
                        ; ('0'-'9', 'A'-'F')
ITE     GT              ; Next 2 instructions    are conditional
ADDGT   R1, R0, #55     ; Convert 0xA -> 'A'
ADDLE   R1, R0, #48     ; Convert 0x0 -> '0'
IT      GT              ; IT block with only one conditional instruction
ADDGT   R1, R1, #1      ; Increment R1 conditionally
ITTEE   EQ              ; Next 4 instructions    are conditional
MOVEQ   R0, R1          ; Conditional move
ADDEQ   R2, R2, #10     ; Conditional add
ANDNE   R3, R3, #1      ; Conditional AND
BNE.W   dloop           ; Branch instruction can only be used in the last
                        ; instruction of an IT block
IT      NE              ; Next instruction is conditional
ADD     R0, R0, R1      ; Syntax error: no condition code used in IT block
```

附录1. 2. 9. 4 TBB 和 TBH

表跳转字节和表跳转半字。

语法

```
TBB [Rn，Rm]
TBH [Rn，Rm，LSL #1]
```

其中：

Rn 为寄存器，包含跳转长度表的地址。

如果 Rn 为 PC，则表格地址为紧跟在 TBB 或 TBH 指令后的字节的地址。

Rm 为索引寄存器。用于存放到表格的索引。对于半字表，LSL #1 将 Rm 中的值加倍，以形成表中的正确偏移量。

操作

这些指令可使用一个单字节偏移表（TBB）或半字偏移表（TBH）来产生一个相对 PC 的向前跳转。Rn 可提供一个表的指针，而 Rm 可提供表的索引。对于 TBB，跳转偏移量是从表返回的无符号字节值的两倍；对于 TBH，跳转偏移量是从表返回的无符号半字值的两倍。跳转到的地址是紧跟 TBB 或 TBH 指令后字节地址偏移量的地址。

限制

限制如下：

◇ Rn 不得为 SP；

◇ Rm 不得为 SP 且不得为 PC；

◇ 这些指令中的任意一个在 IT 块中使用时，它必须是 IT 块中的最后一条指令。

条件标志

这些指令不改变标志。

示例

```
ADR.W     R0, BranchTable_Byte
TBB       [R0, R1]          ; R1 is the index, R0 is the base address of the
                            ; branch table
Case1; an instruction sequence follows
Case2; an instruction sequence follows
Case3; an instruction sequence follows
BranchTable_Byte
DCB       0                 ; Case1 offset calculation
DCB       ((Case2-Case1)/2) ; Case2 offset calculation
DCB       ((Case3-Case1)/2) ; Case3 offset calculation
TBH       [PC, R1, LSL #1]  ; R1 is the index, PC is used as base of the
                            ; branch table
BranchTable_H
DCI       ((CaseA - BranchTable_H)/2)     ; CaseA offset calculation
DCI       ((CaseB - BranchTable_H)/2)     ; CaseB offset calculation
DCI       ((CaseC - BranchTable_H)/2)     ; CaseC offset calculation
CaseA; an instruction sequence follows
CaseB; an instruction sequence follows
CaseC; an instruction sequence follows
```

附录1. 2. 10 其他指令

下表列出了 Cortex-M3 的其他指令：

助记符	简要描述	参见（已改为对应章节名称）
BKPT	断点	BKPT
CPSID	更改处理器状态，禁能中断	CPS
CPSIE	更改处理器状态，使能中断	CPS
DMB	数据内存屏障	DMB
DSB	数据同步屏障	DSB
ISB	指令同步屏障	ISB
MRS	将专用寄存器的内容移到通用寄存器中	MRS
MSR	将通用寄存器的内容移到专用寄存器中	MSR
NOP	不执行任何操作	NOP
SEV	发送事件	SEV
SVC	超级用户调用	BKPT0
WFE	等待事件	BKPT1
WFI	等待中断	BKPT2

附录表 1-13 其他指令

附录1. 2. 10. 1 BKPT

断点。

语法

BKPT #imm

其中：

imm 为一个表达式，其值为 0~255 范围内的一个整数（8 位值）。

操作

BKPT 指令可使处理器进入调试状态。当指令到达某个特定地址时，调试工具可以使用此指令来检查系统的状态。

imm 被处理器忽略。如果需要，调试器可用它来存储关于断点的附加信息。

BKPT 指令可放置在一个 IT 块内，但它会无条件执行，不受 IT 指令指定条件的影响。

条件标志

此指令不改变标志。

示例

```
BKPT 0xAB       ; Breakpoint with immediate value set to 0xAB (debugger can
                ; extract the immediate value by locating it using the PC)
```

附录1. 2. 10. 2 CPS

更改处理器状态。

语法

CPSeffect iflags

其中：

effect 为下列项之一：

IE 清零专用寄存器。

ID 置位专用寄存器。

iflags 为一个或多个标志组成的序列：

i 置位或清零 PRIMASK。

f 置位或清零 FAULTMASK。

操作

CPS 会更改 PRIMASK 和 FAULTMASK 专用寄存器值。有关这些寄存器的更多信息，请参见“异常屏蔽寄存器”小节。

限制

限制如下：

◇ 仅可通过特权软件使用 CPS，在非特权软件中使用该指令无效。

◇ CPS 不能是有条件指令，因此无法用于 IT 块内。

条件标志

此指令不改变条件标志。

示例

```
CPSID i         ; Disable interrupts and configurable fault handlers (set PRIMASK)
CPSID f         ; Disable interrupts and all fault handlers (set FAULTMASK)
CPSIE i         ; Enable interrupts and configurable fault handlers (clear PRIMASK)
CPSIE f         ; Enable interrupts and fault handlers (clear FAULTMASK)
```

附录1. 2. 10. 3 DMB

数据内存屏障。

语法

DMB{cond}

其中：

cond 为一个可选的条件代码，见“条件执行”小节。

操作

DMB 用作一种数据内存屏障。它可确保会先检测到程序中位于 DMB 指令前的所有显式内存访问指令，然后再检测到程序中位于 DMB 指令后的显式内存访问指令。DMB 不影响其他不访问存储器的指令的执行顺序。

条件标志

此指令不改变标志。

示例

```
DMB   ; Data Memory Barrier
```

附录1. 2. 10. 4 DSB

数据同步屏障。

语法

DSB{cond}

其中：

cond 为一个可选的条件代码，见“条件执行”小节。

操作

DSB 是一种特殊的数据同步内存屏障。只有当 DSB 指令执行完毕后，才会执行程序中位于此指令后的指令。当 DSB 指令前的所有显式内存访问完成时，DSB 指令才会完成。

条件标志

此指令不改变标志。

示例

```
DSB       ; Data Synchronisation Barrier
```

附录1. 2. 10. 5 ISB

指令同步屏障。

语法

ISB{cond}

其中：

cond 为一个可选的条件代码，见“条件执行”小节。

操作

ISB 用作指令同步屏障。它可刷新处理器中的流水线，以便在 ISB 指令完成后，才从缓存或存储器中提取位于该指令后的所有其他指令。

条件标志

此指令不改变标志。

示例

```
ISB     ; Instruction Synchronisation Barrier
```

附录1. 2. 10. 6 MRS

将一个专用寄存器的内容移到通用寄存器。

语法

MRS{cond} Rd，spec_reg

其中：

cond 为一个可选的条件代码，见“条件执行”小节。

Rd 为目标寄存器。

spec_reg 可以是以下任意一个：APSR、IPSR、EPSR、IEPSR、IAPSR、EAPSR、PSR、MSP、PSP、PRIMASK、BASEPRI、BASEPRI_MAX、FAULTMASK 或 CONTROL。

操作

可将 MRS 与 MSR 配合使用作为一个更新 PSR 的读取-修改-写入序列的一部分，例如清除 Q 标志。

在进程交换代码中，必须保存换出进程的编程模型状态，包括 PSR 的相关内容。同样，也必须恢复换入进程的状态。这些操作将 MRS 用于状态保存指令序列，将 MSR 用于状态恢复指令序列。

注：与 MRS 指令共同使用时，BASEPRI_MAX 为 BASEPRI 的一个别名。参见“MSR”小节。

限制

Rd 不得为 SP 且不得为 PC。

条件标志

此指令不改变标志。

示例

```
MRS   R0, PRIMASK       ; Read PRIMASK value and write it to R0
```

附录1. 2. 10. 7 MSR

将一个通用寄存器的内容移到指定的专用寄存器。

语法

MSR{cond} spec_reg，Rn

其中：

cond 为一个可选的条件代码，见“条件执行”小节。

Rn 为源寄存器。

spec_reg 可以是以下任意一个：APSR、IPSR、EPSR、IEPSR、IAPSR、EAPSR、PSR、MSP、PSP、PRIMASK、BASEPRI、BASEPRI_MAX、FAULTMASK 或 CONTROL。

操作

MSR 中寄存器访问操作取决于特权级别。非特权软件只能访问 APSR，见表“APSR 位分配”。特权软件可访问所有专用寄存器。

在非特权软件中，对 PSR 中未分配状态位或执行状态位的写入被忽略。

当写入 BASEPRI_MAX 时，指令仅在以下情况之一时才写入 BASEPRI：

◇ Rn 为非零值且当前 BASEPRI 值为 0；

◇ Rn 为非零值且小于当前 BASEPRI 值。 参见“MRS”小节。

限制

Rn 不得为 SP 且不得为 PC。

条件标志

此指令根据 Rn 中的值显式更新标志。

示例

```
MSR   CONTROL, R1     ; Read R1 value and write it to the CONTROL register
```

附录1. 2. 10. 8 NOP

不执行任何操作。

语法

NOP{cond}

其中：

cond 为一个可选条件代码，见“条件执行”小节。

操作

NOP 不执行任何操作。NOP 未必一定是消耗时间的 NOP。也许在该指令执行前，处理器就会将其从流水线中删除。

可以使用 NOP 进行填充（padding），例如将后续指令置于 64 位边界上。

条件标志

此指令不改变标志。

示例

```
NOP   ; No operation
```

附录1. 2. 10. 9 SEV

发送事件。

语法

```
SEV{cond}
```

其中：

cond 为一个可选的条件代码，见“条件执行”小节。

操作

SEV 是一条提示指令，向一个多处理器系统中的所有处理器发送事件信号。它还会将本地事件寄存器设置为 1，见“电源管理”小节。

条件标志

此指令不改变标志。

示例

```
SEV       ; Send Event
```

附录1. 2. 10. 10 SVC

超级用户调用。

语法

```
SVC{cond} #imm
```

其中：

cond 为一个可选条件代码，见“条件执行”小节 。

imm 为一个表达式，其值为 0~255 范围内的一个整数（8 位值）。

操作

SVC 指令会引发一个 SVC 异常。

Imm 会被处理器忽略。如果需要，异常处理程序会将其恢复，借以确定所请求的服务。

条件标志

此指令不改变标志。

示例

```
SVC   0x32      ; Supervisor Call (SVC handler can extract the immediate value
                ; by locating it via the stacked PC)
```

附录1. 2. 10. 11 WFE

等待事件。

语法

WFE{cond}

其中：

cond 为一个可选条件代码，见“条件执行”小节。

操作

WFE 为一条提示指令。

如果事件寄存器为 0，WFE 会挂起程序执行，直到发生任一以下事件后再恢复执行：

◇ 一个异常，除非被异常屏蔽寄存器或当前优先级所屏蔽；

◇ 一个异常进入挂起状态，前提是系统控制寄存器中的 SEVONPEND 位置位；

◇ 一个调试进入请求，前提是调试已使能。

◇ 一个外设或多处理器系统中的另一个处理器使用 SEV 指令发出了一个事件信号。

如果事件寄存器为 1，WFE 将其清零并立即返回。

更多信息参见“电源管理”小节。

条件标志

此指令不改变标志。

示例

```
WFE  ; Wait for event
```

附录1. 2. 10. 12 WFI

等待中断。

语法

WFI{cond}

其中：

cond 为一个可选条件代码，见“条件执行”小节。

操作

WFI 是一条提示指令，它会挂起程序执行，直到发生以下事件之一：

一个异常；
一个调试进入请求，无论是否使能了调试。

条件标志

此指令不改变标志。

示例

```
WFI ; Wait for interrupt
```

附录1. 3 ARM Cortex-M3 用户指南：处理器

附录1. 3. 1 编程模型

本节描述了 Cortex-M3 的编程模型。除了对各个内核寄存器的描述外，它还包括关于处理器模式以及软件执行和堆栈的特权等级的信息。

附录1. 3. 1. 1 处理器模式和软件执行的特权等级

处理器模式包括：

◇ 线程模式

用于执行应用软件。处理器在退出复位时进入线程模式。

◇ 处理模式

用于处理异常。处理器在完成异常处理后返回线程模式。

软件执行的特权等级有：

◇ 非特权

软件：

- 对 MSR 和 MRS 指令的访问权限有限，且不能使用 CPS 指令；
- 不能访问系统定时器、NVIC 或系统控制模块；
- 可能限制对存储器或外设的访问。

非特权软件在非特权等级执行。

◇ 特权

软件可使用所有指令，并可访问全部资源。

特权软件在特权等级执行。

在线程模式中，控制寄存器控制着软件执行是有特权的还是非特权的，见表“控制寄存器的位分配”。在处理模式中，软件执行总是有特权的。

只有特权软件才可写入控制寄存器，以改变线程模式下软件执行的特权等级。非特权软件可使用 SVC 指令执行超级用户调用，以将控制权转移给特权软件。

附录1. 3. 1. 2 堆栈

处理器使用满降序堆栈。也就是说，堆栈指针指示最后推入堆栈存储器的项。当处理器将一个新项推入堆栈时，堆栈指针会首先减小，然后将该项写入新的存储单元。处理器实现了两个堆栈：主堆栈和进程堆栈，它们有独立的堆栈指针寄存器，见“堆栈指针”小节。

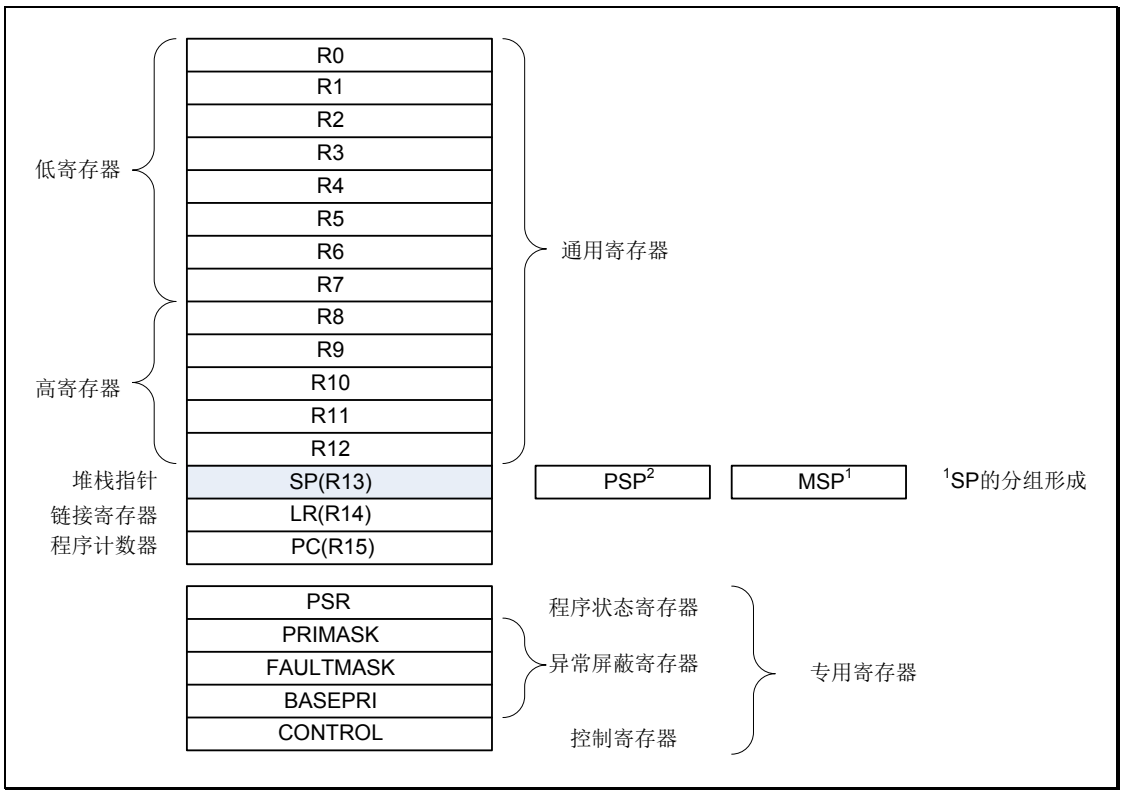
在线程模式下，控制寄存器控制着处理器是使用主堆栈还是进程堆栈，见“控制寄存器的位分配”。在处理模式下，处理器总是使用主堆栈。处理器工作的选项有：

处理器模式	用于执行	软件执行的特权级别	使用的堆栈
线程	应用程序	特权或非特权	主堆栈或进程堆栈
处理	异常处理程序	总是特权	主堆栈

附录表 1-14 处理器模式、执行特权级别和堆栈使用选择汇总

附录1. 3. 1. 3 内核寄存器

处理器的内核寄存器有：



附录图 1-7 内核寄存器

名称	类型[1]	所需特权[2]	复位值	描述（已改为对应章节名称）
R0-R12	RW	两者皆可	未定义	通用寄存器
MSP	RW	特权	参见描述章节	主堆栈指针
PSP	RW	两者皆可	未定义	进程堆栈指针
LR	RW	两者皆可	0xFFFFFFFF	链接寄存器
PC	RW	两者皆可	参见描述章节	程序计数器

名称	类型[1]	所需特权[2]	复位值	描述（已改为对应章节名称）
PSR	RW	特权	0x01000000	程序状态寄存器
APSR	RW	两者皆可	0x00000000	见表“APSR 位分配”
IPSR	R	特权	0x00000000	见表“IPSR 位分配”
EPSR	R	特权	0x01000000	见表“EPSR 位分配”
PRIMASK	RW	特权	0x00000000	见表“PRIMASK 寄存器的位分配”
FAULTMASK	RW	特权	0x00000000	见表“FAULTMASK 寄存器的位分配”
BASEPRI	RW	特权	0x00000000	见表“BASEPRI 寄存器的位分配”
CONTROL	RW	特权	0x00000000	见表“控制寄存器的位分配”

附录表 1-15 内核寄存器集汇总

注[1]：描述线程模式和处理模式下程序执行过程中的访问类型。调试访问可以不同。
注[2]：“两者皆可”意味着特权或非特权软件都可以访问该寄存器。

通用寄存器

R0-R12 为用于数据操作的 32 位通用寄存器。

堆栈指针

堆栈指针（SP）为寄存器 R13。在线程模式下，控制寄存器中的位[1]表示所使用的堆栈指针：

复位时，处理器将地址 0x00000000 的值加载到 MSP。

◇ 0 = 主堆栈指针（MSP）。这是复位值。

◇ 1 = 进程堆栈指针（PSP）。

链接寄存器

链接寄存器（LR）为寄存器 R14。它存储子程序、函数调用和异常的返回信息。复位时，处理器加载 LR 值 0xFFFFFFFF。

程序计数器

程序计数器（PC）为寄存器 R15。它包含当前的程序地址。位[0]总是为 0，因为指令的取指必须按半字对齐。复位时，处理器用复位向量的值加载 PC，复位向量在地址 0x00000004。

程序状态寄存器

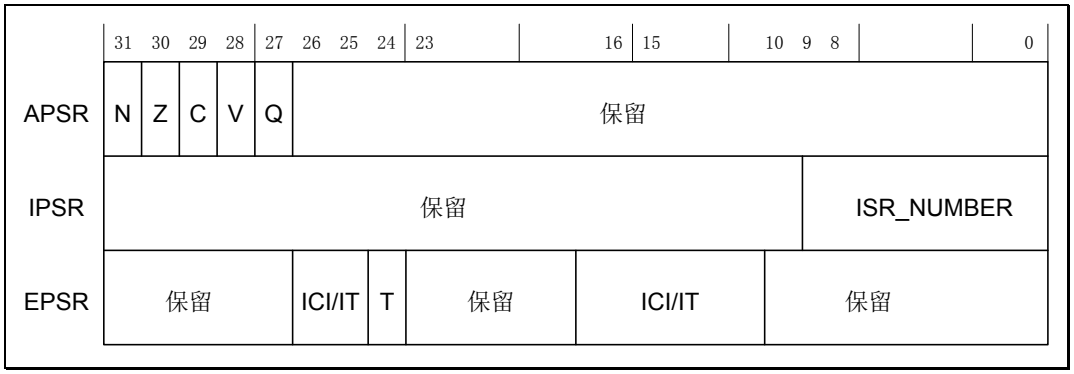
程序状态寄存器（PSR）组合了：

◇ 应用程序状态寄存器（APSR）

◇ 中断程序状态寄存器（IPSR）

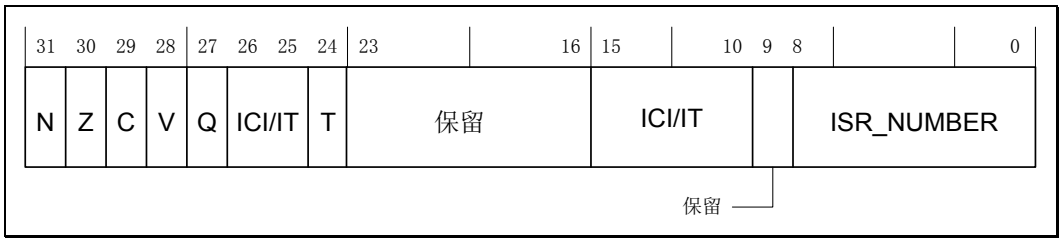
◇ 执行程序状态寄存器（EPSR）。

这些寄存器在 32 位 PSR 中为互斥的位域。位的分配如下：



附录图 1-8 APSR、IPSR 和 EPSR

PSR 位分配如下：



附录图 1-9 组合 xPSR

单独地访问这些寄存器，或访问任意两或三个寄存器的组合，可用寄存器名作为 MSR 或 MRS 指令的一个参数。例如：

◇ 使用带有 MRS 指令的 PSR 读取全部寄存器；

◇ 使用带有 MSR 指令的 APSR 写入 APSR。

PSR 组合和属性如下：

寄存器	类型	组合
PSR	RW[1][2]	APSR、EPSR 与 IPSR
IEPSR	R	EPSR 与 IPSR
IAPSR	RW[1]	APSR 与 IPSR
EAPSR	RW[2]	APSR 与 EPSR

附录表 1-16 PSR 寄存器组合

注[1]：处理器忽略 IPSR 位的写入值。
注[2]：对 EPSR 位进行读操作返回结果为零且处理器忽略这些位的写入值。

关于如何访问程序状态寄存器的更多信息，请参见指令描述“MRS”和“MSR”小节。

应用程序状态寄存器：APSR 包含了前面指令执行的条件标志的当前状态。其属性请参见 “内核寄存器集汇总”表中寄存器汇总。位分配如下：

位	名称	功能
[31]	N	负数或小于标志： 0：操作结果为正数、零、大于或等于

位	名称	功能
		1：操作结果为负数或小于
[30]	Z	零标志： 0：操作结果非零 1：操作结果为零
[29]	C	进位或借位标志： 0：加法操作未产生进位或减法操作产生了一个借位 1：加法操作产生了一个进位或减法操作未产生借位
[28]	V	溢出标志 0：操作未导致溢出发生 1：操作导致溢出发生
[27]	Q	粘着饱和（sticky saturation）标志： 0：复位或该位被最终清零后没有出现饱和 1：SSAT 或 USAT 指令导致了饱和的出现 该位由软件使用 MRS 指令清零。
[26:0]	-	保留

附录表 1-17 APSR 位分配

中断程序状态寄存器：IPSR 包含当前中断服务程序（ISR）的异常类型号。其属性请参见“内核寄存器集汇总”表中寄存器汇总。位分配如下：

位	名称	功能
[31:9]	-	保留
[8:0]	ISR_NUMBER	该位为当前异常的序号。 0 = 线程模式 1 = 保留 2 = NMI 3 = HardFault 4 = 存储器管理故障 5 = 总线故障 6 = 使用故障 7~10 = 保留 11 = SVCall 12 = 保留供调试使用 13 = 保留 14 = PendSV 15 = SysTick 16 = IRQ0 17 = IRQ1（第一个设备专用中断） ..... 255 = IRQ243（最后执行的中断，取决于设备） 更多信息请参见 “异常类型”小节。

附录表 1-18 IPSR 位分配

执行程序状态寄存器：EPSR 包含以下所列项之一的 Thumb 状态位和执行状态位：

◇ If-Then（IT）指令

◇ 用于一条中断多个加载或多个存储指令的可中断-可继续指令（ICI）域。

EPSR 属性，请参见“内核寄存器集汇总”表中寄存器汇总。位分配为：

位	名称	功能
[31:27]	-	保留。
[26:25]，[15:10]	ICI	可中断-可继续的指令位，参见“Cortex 微控制器软件接口标准”小节。
[26:25]，[15:10]	IT	IT 指令的执行状态位，参见“IT”小节。
[24]	T	一直设置为 1。
[23:16]	-	保留。
[9:0]	-	保留。

附录表 1-19 EPSR 位分配

如下图用 MSR 指令，直接通过应用软件读取 EPSR，则永远返回零。试图在应用软件中使用 MSR 指令写 EPSR，则会被忽略。故障处理程序可检查推入堆栈的 PSR 中 EPSR 的值，以指示出有故障的操作。参见异常进入和返回”小节。

可中断-可继续指令：当执行 LDM 或 STM 指令期间发生中断时，处理器执行下列操作：

处理中断以后，处理器：

◇ 暂时停止多个加载或多个存储指令的运行。

◇ 将多个操作中的下一个寄存器操作数保存到 EPSR 的位[15:12]。

◇ 返回位[15:12]指向的寄存器。

◇ 恢复执行多个加载或存储指令。

当 EPSR 保存有 ICI 执行状态时，位[26:25，11:10]为零。

If-Then 块：If-Then 块在一个 16 位 IT 指令后可跟随最多 4 条指令。块中的每条指令都是有条件的。指令条件既可以全部相同，也可以是其它条件的逻辑反。更多信息请参见 “ IT”小节。

异常屏蔽寄存器

异常屏蔽寄存器禁止处理器处理异常。当异常可能影响到时序关键任务时禁用异常。

用 MSR 和 MRS 指令访问异常屏蔽寄存器，或使用 CPS 指令来改变 PRIMASK 或 FAULTMASK 的值。更多信息请参见“MRS”、“MSR”和“CPS”小节。

优先级屏蔽寄存器：PRIMASK 寄存器能阻止所有可配置优先级的异常的激活。其属性见“内核寄存器集汇总”表中寄存器汇总。位分配见下表。

位	名称	功能
[31:1]	-	保留
[0]	PRIMA	0：无影响
	SK	1：阻止所有可配置优先级的异常的激活

附录表 1-20 PRIMASK 寄存器的位分配

故障屏蔽寄存器：故障屏蔽寄存器阻止除不可屏蔽中断（NMI）外所有异常的激活。其属性见“内核寄存器集汇总”表中寄存器汇总。位分配见下表。

位	名称	功能
[31:1]	-	保留
[0]	FAULTMASK	0：无影响
		1：阻止除 NMI 外其他异常的激活

附录表 1-21 FAULTMASK 寄存器的位分配

当从任意异常处理程序退出时（NMI 处理程序除外），处理器将 FAULTMASK 位清零。

基本优先级屏蔽寄存器：BASEPRI 寄存器定义了针对异常处理的最低优先级。当 BASEPRI 设置为一个非零值时，它阻止激活所有与 BASEPRI 值相同或更低优先级的异常。其属性见“内核寄存器集汇总”表中寄存器汇总。位分配见下表。

位	名称	功能
[31:8]	-	保留
[7:0]	BASEPRI[1]	优先级屏蔽位： 0x0000：无影响 非零值：定义了异常处理的基本优先级 处理器不处理那些优先级值大于或等于 BASEPRI 值的异常。

附录表 1-22 BASEPRI 寄存器的位分配

注[1]：该域与中断优先级寄存器中的优先级域相似。处理器仅执行该域的位[7:M]，位[M-1:0]被读为 0 且忽略写入值。M 值取决于所选设备。更多信息参见“中断优先级寄存器”小节。请记住，较高优先级域值对应较低异常优先级。

控制寄存器

当处理器处于线程模式时，控制寄存器控制所用堆栈和软件执行的特权等级。其属性见“内核寄存器集汇总”表中寄存器汇总。位分配见下表。

位	名称	功能
[31:2]	-	保留
[1]	有效堆栈指针	定义了当前堆栈： 0：MSP 是当前堆栈指针 1：PSP 是当前堆栈指针。在处理模式下，该位读取为 0 且忽略写入值。
[0]	线程模式特权级别	定义了线程模式下的特权级别： 0：特权 1：非特权

附录表 1-23 控制寄存器的位分配

处理模式始终使用 MSP，因此，当在处理模式下时，处理器会忽略对控制寄存器有效堆栈指针位的显式写入。异常进入和返回机制会更新控制寄存器。

在一个 OS 环境中，ARM 建议线程模式下运行的线程使用进程堆栈与内核，而异常处

理程序使用主堆栈。

默认情况下，线程模式使用 MSP。如要将线程模式下使用的堆栈指针切换到 PSP，使用 MSR 指令将“有效”堆栈指针位设置为 1，见“MSR”小节。

注：改变堆栈指针时，软件必须在 MSR 指令后立即使用一条 ISB 指令。这样可确保 ISB 后的指令使用新的堆栈指针执行操作。参见“ISB”小节。

附录1. 3. 1. 4 异常和中断

Cortex-M3 处理器支持中断和系统异常。处理器和可嵌套向量中断控制器（NVIC）确定所有异常的优先级并进行处理。一个异常会改变软件控制的正常流程。处理器使用处理模式来处理除复位外的所有异常。更多信息参见“异常进入”和“异常返回”小节。

NVIC 寄存器控制中断处理。更多信息参见“可嵌套向量中断控制器 ”小节。

附录1. 3. 1. 5 数据类型

处理器：

◇ 支持以下数据类型：

- 32 位字
- 16 位半字
- 8 位字节

◇ 支持 64 位数据传输指令。

◇ 以小端形式管理所有的数据内存访问。更多信息参见“存储区、类型和属性”小节。

附录1. 3. 1. 6 Cortex 微控制器软件接口标准

对于 Cortex-M3 微控制器系统，Cortex 微控制器软件接口标准（CMSIS）定义了：

◇ 一种通用方式，用来：

- 访问外设寄存器；
- 定义异常向量。

◇ 以下项的名称：

- 内核外设的寄存器；
- 内核异常向量。

◇ 用于 RTOS 内核的设备独立接口，包括一个调试通道。

CMSIS 包括 Cortex-M3 处理器中内核外设的地址定义和数据结构。它还包括可选的中间件组件接口，中间件组件包括 TCP/IP 堆栈和 Flash 文件系统。

CMSIS 能够重用各个中间件厂商的模板代码，以及符合 CMSIS 要求的软件组件组合，从而简化了软件开发工作。软件厂商可扩充 CMSIS，包括加入其外设定义，以及这些外设的访问函数。

本文件包括了 CMSIS 中定义的寄存器名称，并简要描述了用于访问处理器内核和内核外设 的 CMSIS 函数。

注：本文件使用了 CMSIS 定义的寄存器的简称。在个别情况下，这些名称可能不同于其他文件中使用的结构上的简称。

以下各节给出了关于 CMSIS 的更多信息：

◇ “电源管理编程提示”

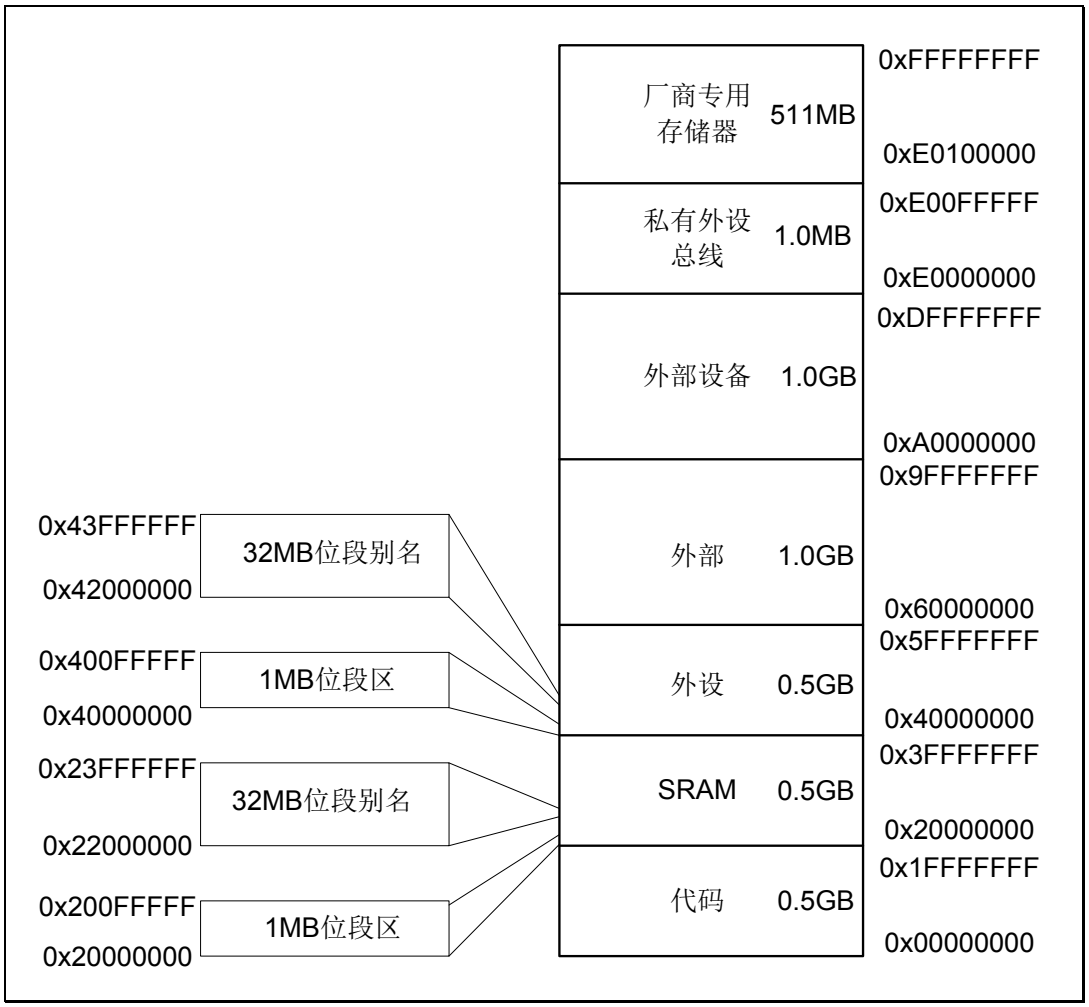
◇ “内在函数”

◇ “Cortex-M3 NVIC 寄存器的 CMSIS 映射”

◇ “ NVIC 编程提示”。

附录1. 3. 2 存储器模型

本节描述了处理器存储器映射、存储器访问行为，以及位段（bit-banding）特性。处理器具有固定存储器映射，可提供高达 4GB 可寻址存储器。存储器映射为：



附录图 1-10 Cortex-M3 处理器预定义的存储器映射

SRAM 和外设区包括了位段区。位段提供了对位数据的原子操作，见“位段”小节。

处理器将私有外设总线（PPB）区保留用于内核外设寄存器，见“关于 Cortex-M3 外设”小节。

附录1. 3. 2. 1 存储区、类型和属性

MPU 的存储器映射和编程将存储器映射分隔为区。每个区都有一个确定的存储器类型，有些区有附加的存储器属性。存储器类型和属性决定了对该区访问的行为。

存储器类型有：

◇ 正常型：处理器可将事务重新排序以改善效率，或执行猜读（speculative read）。

◇ 设备型：处理器保持相对于其他设备或非常有序型存储器的事务顺序。

◇ 非常有序型：处理器保持相对于全部其他事务的事务顺序。

设备型和非常有序型存储器有不同的排序要求，这意味着存储器系统可以缓冲对设备型存储器的写入，但不得缓冲对非常有序型存储器的写入。

其他存储器属性包括：

◇ 可共享

对于一个可共享存储区，存储器系统可在一个系统的总线主机与多个总线主机之间（例如处理器与 DMA 控制器之间）提供数据同步。

非常有序型存储器总是可共享的。

如果多个总线主机可访问一个不可共享的存储区，则软件必须保证多个总线主机之间的数据一致性。

◇ 从不执行（XN）

表示处理器阻止指令的访问。任何从 XN 区获取指令的尝试都会导致一个存储器管理故障异常。

附录1. 3. 2. 2 存储器系统访问的排序

对于多数显式存储器访问指令引起的存储器访问，只要不影响指令序列的行为，存储器系统就不会保证访问完成的顺序与指令的程序顺序相匹配。通常，如果程序正确执行依赖于两个存储器访问按程序的顺序完成，则软件必须在存储器访问指令之间插入一条存储器屏障指令，见“存储器访问的软件排序”小节。

但是，存储器系统一定要保证对设备型和非常有序型存储器的一定访问次序。对于两条存储器访问指令 A1 和 A2，如果在程序顺序中，A1 在 A2 之前发生，则两条指令产生的存储器访问次序为：

A2 / A1	正常访问	设备访问		非常有序访问
		非共享	可共享	
正常访问	-	-	-	-
设备访问，非共享	-	<	-	<
设备访问，可共享	-	-	<	<
非常有序访问	-	<	<	<

附录表 1-24 存储器排序限制

其中：

“-”表示存储器系统不保证访问次序。

“<”表示访问依照了程序顺序，即，A1 总在 A2 之前执行。

附录1. 3. 2. 3 存储器访问行为

存储器映射中各区的访问行为如下：

地址范围	存储区	存储器类型	XN	描述
0x00000000~ 0x1FFFFFFF	代码	正常[1]	-	编程代码的可执行区。也可在此放置数据。
0x20000000~ 0x3FFFFFFF	SRAM	正常[1]	-	数据的可执行区。也可在此放置代码。该区包含位段和位段别名区，参见表“SRAM 存储器位段区”
0x40000000~ 0x5FFFFFFF	外设	设备[1]	XN[1]	该区包含位段区和位段别名区，参见表“外设存储器位段区”
0x60000000~ 0x9FFFFFFF	外部 RAM	正常[1]	-	数据的可执行区。
0xA0000000~ 0xDFFFFFFF	外部设备	设备[1]	XN[1]	外部设备存储器。
0xE0000000~ 0xE00FFFFF	私有外设总线	非常有序[1]	XN[1]	该区包含 NVIC、系统定时器和系统控制模块。
0xE0100000~ 0xFFFFFFFF	厂商专用设备	设备[1]	XN[1]	不用于 ES32 设备。

附录表 1-25 存储器访问行为

注[1]：详细信息请参见“存储区、类型和属性”小节。

代码、SRAM 和外部 RAM 区可保存程序。然而，最有效的程序访问是从代码区。这是因为处理器有单独的总线，能够同时进行指令获取和数据访问。

MPU 可以覆盖本节所述的默认存储器访问行为。更多信息见“存储器保护单元”小节。

附录1. 3. 2. 4 存储器访问的软件排序

程序流中指令顺序并不是总能保证对应存储器事务的顺序。这是因为：

◇ 只要不影响指令序列的行为，处理器可将一些存储器访问重新排序来提高效率。

◇ 处理器有多个总线接口。

◇ 存储器映射中的存储器或设备具有不同的等待状态。

◇ 有些存储器访问是缓冲的或者是猜测的。

“存储器系统访问的排序”小节描述了存储器系统保证存储器访问顺序的情况。否则，如果存储器访问顺序很关键，软件必须包括存储器屏障指令以强制排序。处理器提供以下存储器屏障指令：

◇ DMB

数据内存屏障（DMB）指令确保未完成的存储器事务先于后续存储器事务完成。见“DMB”小节。

◇ DSB

数据同步屏障（DSB）指令确保未完成的存储器事务在执行后续指令前完成。请参见“DSB”小节。

◇ ISB

指令同步屏障（ISB）确保全部完成的存储器事务的效果可以被后续指令识别。请参见“ISB”小节。

在下列示例情况下可使用存储器屏障指令：

对非常有序型存储器（例如系统控制模块）的存储器访问不需要使用 DMB 指令。

◇ MPU 编程：

- 使用一个 DSB 指令，确保在上下文切换结束时 MPU 立即生效。
- 如果用一个跳转或调用访问了 MPU 配置代码，则使用一个 ISB 指令，以确保在对一个或多个 MPU 区编程后，新的 MPU 设置立即生效。如果是通过异常机制输入 MPU 配置代码，则不需要 ISB 指令。

◇ 向量表。如果程序改变了一个向量表项，然后使能了对应的异常，则要在两个操作间使用 DMB 指令。这样可以确保如果异常在使能后被立即获取，处理器使用新的异常向量。

◇ 自修改代码。如果程序包含自修改代码，则在程序中的代码修改之后要立即使用一条 ISB 指令。这样可确保后续指令执行时使用更新后的程序。

◇ 存储器映射切换。如果系统包含一个存储器映射切换机制，则在切换程序中的存储器映射后使用 DSB 指令。这样可确保后续指令执行时使用更新后的存储器映射。

◇ 动态异常优先级更改。如果在异常挂起或有效期间，必须更改一个异常的优先级，则在更改后使用 DSB 指令。这样可确保更改在 DSB 指令完成时生效。

◇ 在多主机系统中使用一个信号量。如果系统包含一个以上的总线主机（例如系统中有另一个处理器），则每个处理器都必须在任何信号量指令后使用一个 DMB 指令，才能确保其他总线主机能按照执行的顺序看到存储器事务。

附录1. 3. 2. 5 位段

位段区将一个位段别名区中的每个字映射到位段区中的单个位。位段区占用 SRAM 和外设存储区的最低 1MB。

存储器映射中有两个 32MB 的别名区，它们映射到两个 1MB 的位段区：

◇ 对 32MB SRAM 别名区的访问映射到 1MB SRAM 位段区，如下表“SRAM 存储器位段区”所示；

◇ 对 32MB 外设别名的访问映射到 1MB 外设位段区，如下表“外设存储器位段区”所示。

地址范围	存储区	指令与数据访问
0x20000000-0x200FFFFF	SRAM 位段区	直接访问此存储区与访问 SRAM 存储器一样，但是也可以通过位段别名区对此区进行位寻址。
0x22000000-0x23FFFFF0	SRAM 位段别名区	对此区的数据访问被重映射到位段区。写操作的效果相当于读取-修改-写入。指令访问不能重映射。

附录表 1-26 SRAM 存储器位段区

地址范围	存储区	指令与数据访问
0x40000000~0x400FFFFF	外设位段别名区	对此存储区的直接访问与访问外设存储器一样，但是也可以通过位段别名区对此区进行位寻址。
0x42000000~0x44FFFFFF	外设位段区	对此区的数据访问被重映射到位段区。写操作的效果相当于读取-修改-写入。不允许指令访问。

附录表 1-27 外设存储器位段区

注：对 SRAM 或外设位段别名区的字访问映射到 SRAM 或外设位段区中的单个位。

以下公式给出了别名区映射到位段区的方式：

bit_word_offset = (byte_offset x 32) + (bit_number x 4)

bit_word_addr = bit_band_base + bit_word_offset

其中：

◇ bit_word_offset 是位段存储区中目标位的位置。

◇ bit_word_addr 是映射到目标位的字在别名存储区中的地址。

◇ bit_band_base 是别名区的起始地址。

◇ byte_offset 是包含目标位的位段区的字节号。

◇ bit_number 是目标位的位置 0-7。

下图给出了 SRAM 位段别名区和 SRAM 位段区之间的位段映射示例：

◇ 0x23FFFFE0 处的别名字映射到 0x200FFFFF 处位段字节的位[0]：

◇ 0x23FFFFE0 = 0x22000000 +（0xFFFFF*32） +（0*4）。

◇ 0x23FFFFFC 处的别名字映射到 0x200FFFFF 处位段字节的位[7]：

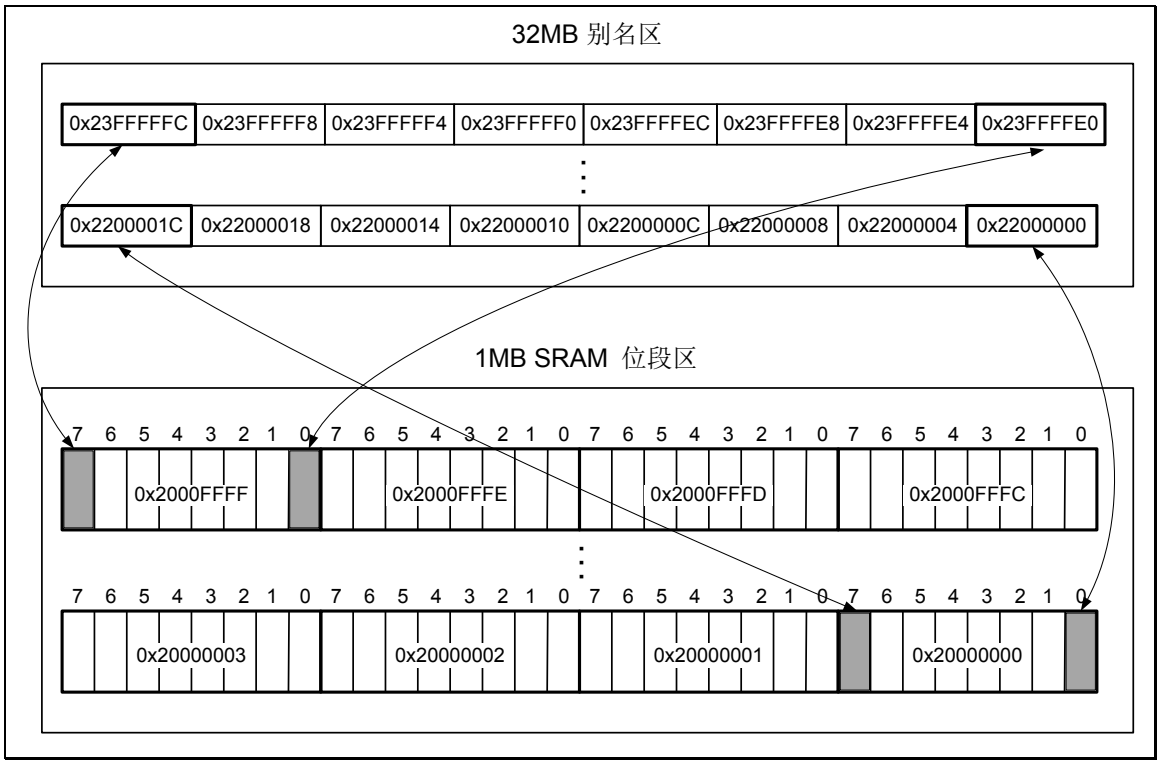
◇ 0x23FFFFFC = 0x22000000 +（0xFFFFF*32） +（7*4）。

◇ 0x22000000 处的别名字映射到 0x20000000 处位段字节的位[0]：

◇ 0x22000000 = 0x22000000 +（0*32） +（0 *4）。

◇ 0x2200001C 处的别名字映射到 0x20000000 处位段字节的位[7]：

◇ 0x2200001C = 0x22000000+（0*32） +（7*4）。



附录图 1-11 位段映射

直接访问一个别名区

向别名区中一个字执行写操作会更新位段区中的单个位。

向别名区中一个字写入值的位[0]决定了写入位段区中目标位的值。写入一个位[0]设为 1 的值，则会向位段位写入一个 1，写入一个位[0]设为 0 的值，则会向位段位写入一个 0。

别名字的位[31:1]对位段位无影响。写入 0x01 与写入 0xFF 效果相同。写入 0x00 与写入 0x0E 效果相同。

读取别名区中一个字：

◇ 0x00000000 表示位段区中目标位被设为 0；

◇ 0x00000001 表示位段区中目标位被设为 1。

直接访问一个位段区

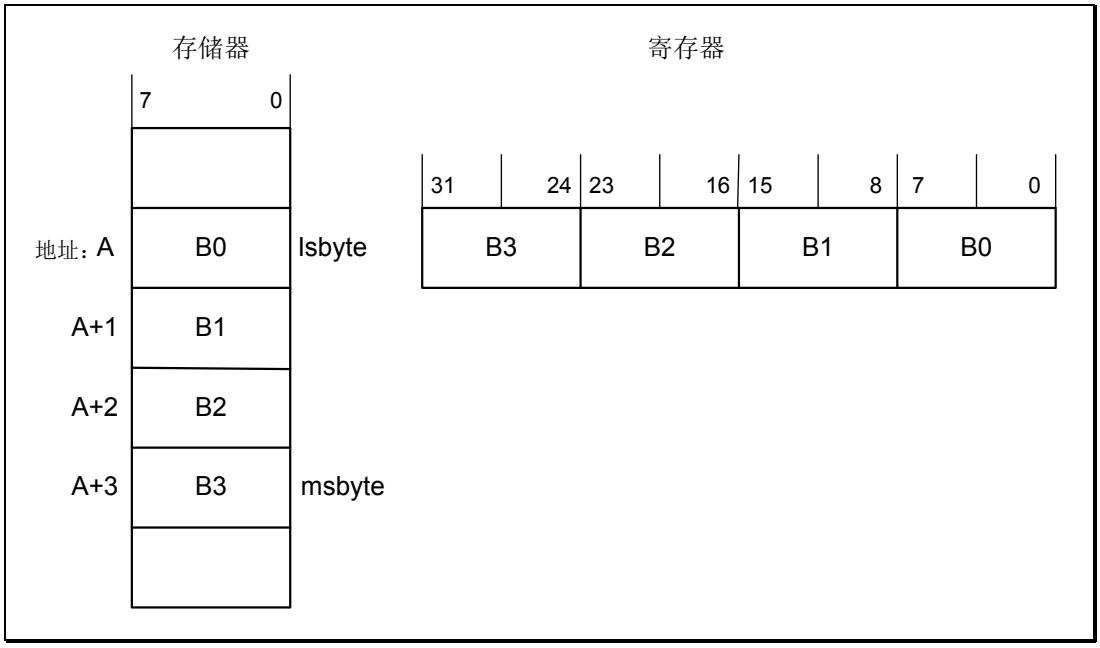
“存储器访问行为”小节描述了对位段区的直接字节、半字或字访问的行为。

附录1. 3. 2. 6 存储器字节序

处理器将存储器视为一个线性的字节集合，从零开始递增序号。例如，字节 0-3 保存最先存储的字，字节 4-7 保存第二个存储的字。“小端（Little-endian）格式”小节描述了数据字如何存储在存储器中。

小端（Little-endian）格式

在小端格式中，处理器将一个字的最低（有效）位字节存储在序号最小的字节，最高位（有效）位字节存储在序号最大的字节。例如：



附录图 1-12 小端格式

附录1. 3. 2. 7 同步原语

Cortex-M3 指令集包括多对同步原语。这些原语提供了一种可以为线程或进程使用的非阻塞机制，从而获得某个存储单元的独占访问权限。软件可用它们来执行有保障的读取-修改-写入存储器更新序列，或实现一个信号量（semaphore）机制。一个同步原语对包括：

◇ 一条“独占加载”指令

用于读取某个存储单元的值，申请对该单元的独占访问。

◇ 一条“独占存储”指令

用于尝试写入相同的存储单元，返回一个状态位到寄存器。如果此位为：

- 0：表示线程或进程获得对存储器的独占访问权限，写入成功；
- 1：表示线程或进程未获得对存储器的独占访问权限，未进行写入。

“独占加载”和“独占存储”指令对有：

◇ 字指令 LDREX 和 STREX

◇ 半字指令 LDREXH 和 STREXH

◇ 字节指令 LDREXB 和 STREXB。

对于“独占存储”指令，软件必须使用相应的“独占加载”指令。

为了对某个存储单元进行有保障的读取-修改-写入，软件必须：

1. 使用一条“独占加载”指令，读取某个单元的值。
2. 按需要更新值。
3. 使用一条“独占存储”指令，尝试将新的值写回存储单元，并测试返回的状态位。如果 此位为：
   - 0：读取-修改-写入成功完成；
   - 1：未完成读取-修改-写入。这表示步骤 1 中返回的值可能已过期。软件必须重试读取-修改-写入序列。

软件可使用同步原语来实现一个信号量，如下：

1. 使用一条“独占加载”指令从信号量地址读取，检查信号量是否空闲。
2. 如果信号量空闲，则使用“独占存储”将声明值（claim value）写入信号量地址。
3. 如果步骤 2 返回的状态位指示“独占存储”成功，则软件已获得了信号量。但是，如果“独占存储”失败，则可能在软件执行步骤 1 后，有其他进程获得了信号量。

Cortex-M3 包括一个独占访问监控器，它会标注处理器已执行“独占加载”指令的情况。

如有以下情况，处理器会删除独占访问标签：

◇ 执行了一条 CLREX 指令。

◇ 无论写入是否成功，执行了一条“独占存储”指令。

◇ 发生一个异常。这表示处理器可解决不同线程间的信号量冲突。

有关同步原语指令的更多信息，请“LDREX 和 STREX”和“CLREX” 小节。

附录1. 3. 2. 8 同步原语的编程提示

ANSI C 不能直接产生独占访问指令。有些 C 编译器提供了可产生这些指令的内在函数：

指令	内在函数
LDREX，LDREXH 或 LDREXB	unsigned int ldrex(volatile void *ptr)
STREX，STREXH 或 STREXB	int strex(unsigned int val, volatile void *ptr)
CLREX	void clrex(void)

附录表 1-28 独占访问指令的 C 编译器内在函数

实际产生的独占访问指令取决于传递给内在函数的指针数据类型。例如，以下 C 代码产生所需的 LDREXB 操作：

```
__ldrex（(volatile  char *） 0xFF） ;
```

附录1. 3. 3 异常模型

本节描述了异常模型。

附录1. 3. 3. 1 异常状态

每个异常都是以下几个状态之一：

- 无效

  异常无效且未被挂起。

- 挂起

  异常等待处理器的处理。 来自外设或软件的中断请求可将相应中断的状态改为挂起。

- 有效

  异常正在由处理器处理，但尚未完成。

注：一个异常处理程序可以中断其他异常处理程序的执行。在此情况下，两个异常都处于有效状态。

- 有效和挂起

  处理器正在处理一个异常，且有一个相同来源的处于挂起状态的异常。

附录1. 3. 3. 2 异常类型

异常类型如下：

- 复位

  上电或热复位时，复位启动。异常模型将复位作为一种特殊的异常形式。当复位使能时，处理器可能在一条指令的任何位置停止操作。当复位禁能时，由向量表中的复位表项提供重新开始执行的地址。在线程模式下，重新执行是特权执行。

- NMI

  不可屏蔽中断（NMI）可由一个外设发出信号或由软件触发。这是除复位外的最高优先级异常。它被永久性使能，具有固定的优先级-2。NMI 不能：

  - 因任何其他异常的激活而被屏蔽或阻止
  - 被除复位外的其他任何异常抢占。

- HardFault

  HardFault 是一种异常，发生原因是异常处理期间出错，或异常无法被任何其他异常机制管理。HardFault 具有固定优先级-1，意味着它们的优先级高于任何具有可配置优先级的异常。

- 存储器管理故障

  存储器管理故障是一种由存储器保护相关故障而引发的异常。对于指令和数据存储事务，MPU 或固定的存储器保护约束条件决定此故障。此故障用于中止对从不执行（XN）存 储区的指令访问，即使 MPU 被禁能也是如此。

- 总线故障

  总线故障是一种异常，由一个指令或数据存储器事务的存储器相关故障引发。这可能源

于从一个存储器系统总线上检测出的错误。

- 使用故障

  使用故障是指令执行相关故障导致的一种异常。这包括：

  - ◇ 未定义的指令
  - ◇ 非法的非对齐访问
  - ◇ 指令执行时的无效状态
  - ◇ 异常返回时的错误。

  当内核被配置为报告使用故障时，以下情况可以导致一个使用故障：

  - ◇ 字和半字存储器访问时的非对齐地址；
  - ◇ 除以零。

- SVCall

  一次超级用户调用（SVC）是由 SVC 指令触发的异常。在 OS 环境中，应用程序可使用 SVC 指令来访问 OS 内核函数和设备驱动程序。

- PendSV

  PendSV 是一个对系统级服务的中断驱动请求。在 OS 环境中，当无其他有效的异常时，使 用 PendSV 进行上下文切换。

- SysTick

  SysTick 异常是当系统定时器达到零时产生的异常。软件也可以产生一个 SysTick 异常。在 OS 环境中，处理器可将此异常用作系统节拍。

- 中断（IRQ）

  中断（IRQ）是由一个外设发出信号，或由一个软件请求而产生的异常。所有中断都与指令执行异步。在系统中，外设使用中断与处理器进行通信。

异常序号[1]	IRQ 序号[1]	异常类型	优先级	向量地址或偏移量[2]	激活
1	-	复位	-3，最高	0x00000004	异步
2	-14	NMI	-2	0x00000008	异步
3	-13	HardFault	-1	0x0000000C	-
4	-12	存储器管理故障	可配置[3]	0x00000010	同步
5	-11	总线故障	可配置[3]	0x00000014	精确时是同步，不精确时是异步
6	-10	使用故障	可配置[3]	0x00000018	同步
7-10	-	-	-	保留	-
11	-5	SVCall	可配置[3]	0x0000002C	同步
12-13	-	-	-	保留	-
14	-2	PendSV	可配置[3]	0x00000038	异步
15	-1	SysTick	可配置[3]	0x0000003C	异步
16 及以上	0 及以上	中断（IRQ）	可配置[4]	0x00000040 及以上[5]	异步

附录表 1-29 不同异常类型的属性

注[1]：为简化软件层级，CMSIS 仅使用 IRQ 序号，因此针对异常（而非中断）使用负值。IPSR 返回异常序号，参见表“IPSR 位分配”。
注[2]：详细信息请参见“向量表”小节。
注[3]：参见“系统处理程序优先级寄存器” 小节。
注[4]：参见“中断优先级寄存器”小节。
注[5]：以 4 为单位递增。

对于异步的异常，除复位外，在异常被触发和处理器进入异常处理程序之间，处理器也可以执行其他指令。

特权软件可以禁能上表“不同异常类型的属性”中列出的具有可配置优先级的异常，见：

◇ “系统处理程序控制和状态寄存器”

◇ “中断清零使能寄存器”。

有关 HardFault、存储器管理故障、总线故障和使用故障的更多信息，请参见“故障处理”小节。

附录1. 3. 3. 3 异常处理程序

处理器使用以下程序来处理异常：

◇ 中断服务程序（ISR）

IRQ0 和更高级别的中断是由 ISR 处理的异常。

◇ 故障处理程序

HardFault、存储器管理故障、使用故障、总线故障是由故障处理程序处理的异常。

◇ 系统处理程序

NMI、PendSV、SVCall SysTick 和故障异常都是系统异常，由系统处理程序处理。

附录1. 3. 3. 4 向量表

向量表包含了堆栈指针复位值和全部异常处理程序的起始地址，又称异常向量。如下图“向量表”列出了向量表中异常向量的顺序。每个向量的最低有效位必须为 1，这表示异常处理程序为 Thumb 代码。请注意 IRQ 序号的上限因设备而异。

异常序号	IRQ序号	偏移量	向量
127	111	0x1FC	IRQ111
⋮		⋮	⋮
18		0x004C	IRQ2
17	2	0x0048	IRQ1
16	1	0x0044	IRQ0
15	0	0x0040	Systick
14	-1	0x003C	PendSV
13	-2	0x0038	保留 保留供调试使用
12			
11			SVCall
10	-5	0x002C	保留 使用
9			
8			
7			
6	-10	0x0018	
5	-11	0x0014	
4	-12	0x0010	
3	-13	0x000C	
2	-14	0x0008	
1		0x0004	
		0x0000	

附录图 1-13 向量表

系统复位时，向量表固定在地址 0x00000000。特权软件可通过写入 VTOR，将向量表起始地址重新定位到不同的存储单元，范围为 0x00000080 至 0x3FFFFF80，见“向量表偏移量寄存器 ”小节。

附录1. 3. 3. 5 异常优先级

如表“不同异常类型的属性”所示，所有异常都有一个相关的优先级，且：

◇ 用一个较低的优先级值表示一个较高的优先级；

◇ 除复位、HardFault 和 NMI 外的其他异常都有可配置的优先级。

如果软件未配置任何优先级，则所有具有可配置优先级的异常都具有 0 优先级。有关异常优

先级配置的信息，请参见：

◇ “系统处理程序优先级寄存器”

◇ “中断优先级寄存器”。

注：可配置的优先级值范围为 0 到 31。这就是说，具有固定的负数优先级值的复位、HardFault 和 NMI 异常总是具有比其他异常更高的优先级。

例如，如果给 IRQ[0]赋以较高的优先级值并给 IRQ[1]赋以较低的优先级值，那么这就意味着 IRQ[1]的优先级高于 IRQ[0]。如果 IRQ[1]和 IRQ[0]都有效，则 IRQ[1]先于 IRQ[0]处理。

如果多个挂起的异常具有相同的优先级，则最先处理具有最低异常序号的异常。例如，如果 IRQ[0]和 IRQ[1]都挂起并具有相同的优先级，则 IRQ[0]先于 IRQ[1]被处理。

当处理器正在执行一个异常处理程序时，如果发生更高优先级的异常，则异常处理程序被抢占。如果发生了与正在处理的异常优先级相同的异常，无论异常序号如何，处理程序不会被抢占。但是，新中断的状态变为挂起。

附录1. 3. 3. 6 中断优先级分组

为了对具有中断的系统加强优先级控制，NVIC 支持优先级分组。这将每个中断优先级寄存器项划分为两个域：

◇ 上区域定义了组优先级；

◇ 下区域定义了组内的子优先级。

只有组优先级才能决定中断异常的抢占。当处理器正在执行一个中断异常处理程序时，另一个与正在处理中的中断具有相同组优先级的中断不会抢占处理程序。

如果多个挂起的中断具有相同的组优先级，子优先级域决定了它们的处理顺序。如果多个挂 起的中断具有相同的组优先级和子优先级，则具有最低 IRQ 序号的中断最先被处理。

有关中断优先级域拆分为组优先级和子优先级的相关信息，见“ 应用中断和复位控 制寄存器”小节。

附录1. 3. 3. 7 异常进入和返回

使用以下术语来描述异常处理：

◇ 抢占

当处理器正在执行一个异常处理程序时，如果某个异常的优先级高于正在处理中异常的优先级，则这个异常可抢占异常处理程序。有关中断抢占的更多信息，请参见“中断优先级分组”小节。

当某个异常抢占另一个异常的处理程序时，这些异常被称为嵌套异常。更多信息请参见“异常进入”小节。

◇ 返回

当异常处理程序完成且满足下列条件时，发生返回：

- 不存在挂起的异常，并且它们有被处理的足够优先级；
- 完成的异常处理程序并未在处理一个迟来（late-arriving）的异常。

处理器弹出堆栈，并将处理器状态恢复为中断发生前的状态。更多信息，请参见“异常返回”小节。

◇ 末尾连锁（Tail-chaining）

此机制可提高异常服务的速度。当一个异常处理程序完成时，如果有一个挂起的异常符合异常进入要求，则跳过堆栈弹出将控制权转移给新的异常处理程序。

◇ 迟来（late-arriving）

此机制可提高抢占速度。如果在前一异常状态保存期间发生了较高优先级的异常，处理器就会切换去处理较高优先级的异常，并启动对此异常的向量获取。状态保存不受迟来异常的影响，因为对这两个异常来说，保存的状态都是一样的。因此，状态保存可以不中断地继续进行。处理器可以接受迟来异常，直到初始异常的异常处理程序中的第一条指令进入处理器的执行阶段。在迟来异常的异常处理程序返回时，采用正常的末尾连锁规则。

异常进入

当存在一个有足够优先级的挂起异常，且满足以下条件之一时，发生异常进入：

◇ 处理器处于线程模式；

◇ 新的异常比正在处理中的异常具有更高优先级，在此情况下新的异常抢占原来的异常。

当一个异常抢占另一个异常时，异常被嵌套。

足够优先级指异常具有高于屏蔽寄存器所设置限值的优先级，见“异常屏蔽寄存器”小节。优先级低于它的异常被挂起，但不被处理器处理。

当处理器取得一个异常时，除非异常为末尾连锁或迟来异常，否则，处理器会将信息推入 当前堆栈。此操作被称为入栈（stacking），而 8 个数据字的结构被称为堆栈帧。堆栈帧包含以下信息：

◇ R0-R3，R12；

◇ 返回地址；

◇ PSR；

◇ LR。

紧接着入栈之后，堆栈指针表示堆栈帧中的最低地址。除非禁能了堆栈对齐，否则，堆栈帧按双字地址对齐。如果配置控制寄存器（CCR）的 STKALIGN 位被设为 1，则在入栈期间要进行堆栈对齐调节。

堆栈帧包括返回地址。这是中断的程序中下一条指令的地址。在异常返回时，此值被恢复到 PC，以便让中断的程序继续运行。

入栈操作的同时，处理器执行向量取址，即从向量表读取异常处理程序的起始地址。当入栈完成时，处理器开始执行异常处理程序。同时，处理器将一个 EXC_RETURN 值写入 LR。用以指示哪个堆栈指针对应于堆栈帧，以及在异常进入发生前，处理器处于何种操作模式。

如果异常进入期间没有发生更高优先级的异常，则处理器开始执行异常处理程序，并自动地将相应挂起中断的状态改变为有效。

如果异常进入期间发生了更高优先级的异常，则处理器开始执行此异常的异常处理程序，且不改变前一异常的挂起状态。这就是迟来异常的情况。

异常返回

当处理器处于处理模式，并执行下列指令之一将 EXC_RETURN 值加载到 PC 时，发生异常返回：

◇ 一条包括 PC 的 POP 指令；

◇ 一条带任意寄存器的 BX 指令；

◇ 一条以 PC 为目标的 LDR 或 LDM 指令。

EXC_RETURN 为异常进入时加载到 LR 中的值。异常机制靠此值来检测处理器何时完成一个异常处理程序。此值的最低 4 位提供关于返回堆栈和处理器模式的信息。下表列出了 EXC_RETURN[3:0]的值以及异常返回行为的描述。

处理器将 EXC_RETURN 的位[31:4]设为 0xFFFFFFF。当此值被加载到 PC 中时，它向处理器指示异常已完成，处理器就会启动异常返回序列操作。

EXC_RETURN[3:0]	描述
bXXX0	保留。
b0001	返回处理模式。异常返回，获得来自 MSP 的状态。返回后指令执行使用 MSP。
b0011	保留。
b01X1	保留。
b1001	返回线程模式。异常返回，获得来自 MSP 的状态。返回后指令执行使用 MSP。
b1101	返回线程模式。 异常返回，获得来自 PSP 的状态。返回后指令执行使用 PSP。
b1X11	保留。

附录表 1-30 异常返回行为

附录1. 3. 4 故障处理

故障为异常的一个子集，见“异常模型”小节。以下情况下会产生一个故障：

◇ 以下情况中的一个总线错误：

- 指令取指或向量表加载
- 数据访问

◇ 一个内部检测到的错误，例如未定义的指令，或试图用 BX 指令改变状态；

◇ 试图从标记为不可执行（XN）的存储区执行一个指令；

◇ 因特权违犯或试图访问一个未管理（unmanaged）区所引起的 MPU 故障。

附录1. 3. 4. 1 故障类型

下表列出了故障类型、故障的处理程序、相应的故障状态寄存器和指示发生故障的寄存器位。关于故障状态寄存器的更多信息，请参见“可配置故障状态寄存器”小节。

故障		处理程序	位名称	故障状态寄存器
读向量总线错误		HardFault	VECTTBL	HardFault 状态寄存器
升级为 HardFault 的故障			FORCED	
MPU 不匹配	由于指令访问	存储器管理故障	IACCVIOL [1]	存储器管理故障地址寄存器
	由于数据访问		DACCVIOL	
	异常入栈期间		MSTKERR	
	异常出栈期间		MUNSKERR	
总线错误	异常入栈期间	总线故障	STKERR	总线故障地址寄存器
	异常出栈期间		UNSTKERR	
	指令预取期间		IBUSERR	
	精确数据总线错误		PRECISERR	
	非精确数据总线错误		IMPRECISERR	
试图访问一个协处理器		使用故障	NOCP	使用故障状态寄存器
未定义指令			UNDEFINSTR	
试图进入一个无效的指令集状态 [2]			INVSTATE	
无效的 EXC_RETURN 值			INVPC	
非法的未对齐加载或存储			UNALIGNED	
除以 0			DIVBYZERO	

附录表 1-31 故障

注[1]：即使 MPU 被禁能，该操作也会在每次访问 XN 区时发生。
注[2]：试图使用一个非 Thumb 指令集。

附录1. 3. 4. 2 故障升级和HardFault

除了 HardFault，所有故障异常都具有可配置的异常优先级，见“系统处理程序优先级寄存器”小节。软件可禁止执行这些故障的处理程序，见“系统处理程序控制和状态寄存器”小节。

通常，异常优先级与异常屏蔽寄存器的值共同决定了处理器是否进入故障处理程序，以

及一个故障处理程序是否可抢占另一个故障处理程序。如 “异常模型”小节所述。

在某些情况下，有可配置优先级的故障可作为一个 HardFault。这被称为优先级升级，该故障被描述为升级到 HardFault。出现以下情况时发生升级到 HardFault：

◇ 故障处理程序会产生与它正在处理的故障相同类型的故障。升级到 HardFault 的原因是故障处理程序不能抢占自身，因为它必须与当前优先级具有相同的优先级。

◇ 故障处理程序导致一个与它正在处理的故障具有相同或更低优先级的故障。这是因为新故障的处理程序不能抢占当前正在执行的故障处理程序。

◇ 异常处理程序导致一个与正在执行的异常具有相同或更低优先级的故障。

◇ 发生了一个故障，但此故障的处理程序未被使能。

如果在进入总线故障处理程序时，堆栈推入期间发生了一个总线故障，则总线故障不会升级到 HardFault。这就是说，如果被破坏的堆栈导致一个故障发生，则即使处理程序的堆栈推入失败，故障处理程序也会执行。此故障处理程序可以工作，但堆栈内容被破坏。

注：只有复位和 NMI 可抢占固定优先级的 HardFault。HardFault 可抢占除复位、NMI 或另一 HardFault 外的任何异常。

附录1. 3. 4. 3 故障状态寄存器和故障地址寄存器

故障状态寄存器指示故障的起因。对于总线故障和存储器管理故障来说，故障地址寄存器指示导致故障发生的操作所访问的地址，如下表所示。

处理程序	状态寄存器名称	地址寄存器名称	寄存器描述
HardFault	HFSR	-	“HardFault 状态寄存器”小节
存储器管理故障	MMFSR	MMFAR	“存储器管理故障地址 寄存器”小节 “存储器管理故障状态 寄存器”小节
总线故障	BFSR	BFAR	“总线故障状态寄存器 ”小节 “总线故障地址寄存器 ”小节
使用故障	UFSR	-	“使用故障状态寄存器 ”小节

附录表 1-32 故障状态和故障地址寄存器

附录1. 3. 4. 4 锁定

执行 NMI 或 HardFault 处理程序时，如果发生 HardFault，则处理器进入一个锁定状态。当处理器处于锁定状态时，它不执行任何指令。处理器保持在锁定状态，直到发生以下情况之一：

◇ 处理器被复位；

◇ 发生一个 NMI 故障。

注：如果 NMI 处理程序使处理器进入锁定状态，则后续 NMI 也不会改变处理器的锁定状态。

附录1. 3. 5 电源管理

注：基于 Cortex-M3 处理器的 ES32 微控制器都支持额外的降功率模式。 有关全部可用的低功率模式的信息，请参见 正文“电源管理及低功耗模式”小节。

Cortex-M3 处理器的睡眠模式可降低功耗：

◇ 睡眠模式会停止处理器时钟；

◇ 深度睡眠模式会停止系统时钟，并关闭 PLL 和 Flash 存储器的电源。

SCR 的 SLEEPDEEP 位用来选择使用哪种睡眠模式，见“系统控制寄存器”小节。更多有关睡眠模式行为的信息，请参见“功率控制”小节。

本节叙述了进入睡眠模式的机制和从睡眠模式唤醒的条件。

附录1. 3. 5. 1 进入睡眠模式

本节叙述了软件用于将处理器置于睡眠模式的机制。

系统可能发生伪唤醒事件，例如，一个调试操作唤醒处理器。因此，软件必须能够在该类事件发生后，将处理器返回睡眠模式。程序可以有一个空闲循环（idle loop）来将处理器置回睡眠模式。

等待中断

等待中断指令 WFI 会使处理器立即进入睡眠模式。当处理器执行 WFI 指令时，处理器会停止执行指令并进入睡眠模式。更多信息，请参见“WFI”小节。

等待事件

注：基于 Cortex-M3 处理器的 ES32 微控制器不执行外部事件。

等待事件指令 WFE 使处理器根据一个单一位事件寄存器的值，有条件地进入睡眠模式。当处理器执行 WFE 指令时，它会检查该寄存器的下列内容：

◇ 如果该寄存器为 0，处理器停止执行指令并进入睡眠模式；

◇ 如果该寄存器为 1，处理器将该寄存器清零，不进入睡眠模式并继续执行指令。

更多信息，请参见“WFE”小节。

如果事件寄存器为 1，就表示当执行 WFE 指令时，处理器不得进入睡眠模式。通常，这是因为使能了一个外部事件信号，或系统中的一个处理器执行了一条 SEV 指令，请参见 “SEV”小节。软件不能直接访问此寄存器。

退出时睡眠（Sleep-on-exit）

如果 SCR 的 SLEEPONEXIT 位设为 1，则当处理器执行完一个异常处理程序时会返回到线程模式，并立即进入睡眠模式。在发生异常时，只在那些需要处理器运行的应用中使用此机制。

附录1. 3. 5. 2 从睡眠模式唤醒

处理器唤醒的条件取决于导致处理器进入睡眠模式的机制。

从“WFI”或“退出时睡眠”唤醒

正常情况下，只有在处理器检测到一个具有足够优先级且可造成异常进入的异常时，处理器才会唤醒。

某些嵌入式系统可能必须在处理器唤醒后和执行一个中断处理程序前，先执行系统恢复任务。为了实现该操作，将 PRIMASK 位设为 1，并将 FAULTMASK 位设为 0。如果有一个中断到达，该中断被使能且具有比当前异常优先级更高的优先级，则处理器唤醒，但不执行中断处理程序，直到处理器将 PRIMASK 设为零。有关 PRIMASK 和 FAULTMASK 的更多信息，请参见“异常屏蔽寄存器”小节。

从“WFE”唤醒

如有以下情况，处理器被唤醒：

◇ 处理器探测到一个有足够优先级的异常，可产生异常进入。

另外，如果 SCR 中 SEVONPEND 位被设为 1，则任何新挂起的中断都会触发一个事件并唤醒处理器，哪怕该中断被禁止或没有足够高的优先级来产生异常进入。更多关于 SCR 的信息，请参见“系统控制寄存器”小节。

附录1. 3. 5. 3 唤醒中断控制器

唤醒中断控制器（WIC）是一个外设，它可以检测一个中断并将处理器从深度睡眠模式、掉电或深度掉电模式唤醒。只有在 SCR 中 DEEPSLEEP 位被设为 1 时，WIC 才被使能。见“系统控制寄存器”小节。

注：ES32 微控制器扩展了降功率模式的数量，使之超出了 Cortex-M3 直接支持的模式数量。关于全部可用降功率模式和对 ES32 微控制器唤醒可能性的详细说明，请参见“功率控制”小节。

WIC 不可编程，且没有任何寄存器或用户接口。它完全依靠来自硬件的信号工作。

当 WIC 被使能，且处理器进入深度睡眠模式或掉电模式时，系统中电源管理单元可能会关闭 Cortex-M3 处理器的大部分电源。产生的副作用是会停止 SysTick 定时器。当 WIC 收到一个中断时，它要消耗一些时钟周期来唤醒处理器并恢复其状态，然后才能处理中断。这就意味着，在深度睡眠模式下，中断等待时间变长。从掉电模式唤醒需要启动设备的很多其他部分，需要消耗更长的时间。从深度掉电模式唤醒还要加上重新建立片上稳压器电压的时间。

注：如果处理器检测到一个与调试器的连接，它会禁能 WIC。

附录1. 3. 5. 4 电源管理编程提示

ANSI C 不能直接产生 WFI 和 WFE 指令。CMSIS 为这些指令提供了以下内在函数：

```
void __WFE(void)   // Wait for Event
void __WFI(void)   // Wait for Interrupt
```

附录1. 4 ARM Cortex-M3 用户指南：外设

附录1. 4. 1 关于 Cortex-M3 外设

私有外设总线（PPB）的地址映射如下：

地址	内核外设	描述
0xE000E008~0xE000E00F	系统控制模块	见表“系统控制模块寄存器汇总”
0xE000E010~0xE000E01F	系统定时器	见表“系统定时器寄存器汇总”
0xE000E100~0xE000E4EF	可嵌套向量中断控制器	见表“NVIC 寄存器汇总”
0xE000ED00~0xE000ED3F	系统控制模块	见表“系统控制模块寄存器汇总”
0xE000ED90~0xE000EDB8	存储器保护单元	见表“MPU 寄存器汇总”
0xE000EF00~0xE000EF03	可嵌套向量中断控制器	见表“NVIC 寄存器汇总”

附录表 1-33 内核外设寄存器区

在寄存器描述中：

- 寄存器类型描述如下：
  - RW：读写。
  - R：只读。
  - W：只写。
- 所需特权给出访问寄存器所需的下列特权等级：
  - 特权：仅特权软件可访问寄存器。
  - 非特权：非特权和特权软件均可访问寄存器。

附录1. 4. 2 可嵌套向量中断控制器

本节叙述了可嵌套向量中断控制器（NVIC）和它所使用的寄存器。NVIC 支持：

- ◇ 多达 112 个中断。实现的中断数量因设备而异。
- ◇ 每个中断有可编程的 0 到 31 优先等级。较高等级对应于较低的优先级，因此 0 级为最高的中断优先级。
- ◇ 中断信号的电平和脉冲检波。
- ◇ 中断优先级的动态重设。
- ◇ 优先级值分组为组优先级和子优先级域。
- ◇ 中断末尾连锁。
- ◇ 一个外部不可屏蔽中断（NMI）。

处理器在异常进入时将其状态自动入栈，并在异常退出时将其状态自动出栈，没有指令开销。 这样就提供了低延迟的异常处理。NVIC 寄存器的硬件实现为：

地址	名称	类型	所需特权	复位值	描述
0xE000E100~0xE000E10C	ISER0 - ISER3	RW	特权	0x00000000	见表“ISER 位分配”
0XE000E180~0xE000E18C	ICER0 - ICER3	RW	特权	0x00000000	见表“ICER 位分配”
0XE000E200~0xE000E20C	ISPR0 - ISPR3	RW	特权	0x00000000	见表“ISPR 位分配”
0XE000E280~0xE000E28C	ICPR0 - ICPR3	RW	特权	0x00000000	见表“ICPR 位分配”
0xE000E300~0xE000E30C	IABR0 - IABR3	R	特权	0x00000000	见表“IABR 位分配”
0xE000E400~0xE000E46C	IPR0 - IPR27	RW	特权	0x00000000	见表 “IPR 位分配”
0xE000EF00	STIR	W	可配置[1]	0x00000000	见表“STIR 位分配”

附录表 1-34 NVIC 寄存器汇总

注[1]：每个数组元对应一个 NVIC 寄存器。例如，数组元 ICER[1]对应 ICER1 寄存器。

附录1. 4. 2. 1 Cortex-M3 NVIC 寄存器的 CMSIS 映射

为了提高软件效率，CMSIS 简化了 NVIC 寄存器的表达。在 CMSIS 中：

- 置位使能、清零使能、置位挂起、清零挂起和有效位寄存器映射到 32 位整数的数组，从而使：
  - ◇ 数组 ISER[0]到 ISER[3]对应于寄存器 ISER0 - ISER3；
  - ◇ 数组 ICER[0]到 ICER[3]对应于寄存器 ICER0 - ICER3；
  - ◇ 数组 ISPR[0]到 ISPR[3]对应于寄存器 ISPR0 - ISPR3；
  - ◇ 数组 ICPR[0]到 ICPR[3]对应于寄存器 ICPR0 - ICPR3；
  - ◇ 数组 IABR[0]到 IABR[3]对应于寄存器 IABR0 - IABR3。
- 8 位的“中断优先级寄存器”域映射到一个 8 位整数数组，从而使数组 IP[0]到 IP[111]对应于寄存器 IPR0 - IPR27，数组表项 IP[n]保存中断 n 的中断优先级。

CMSIS 提供线程安全代码，它实现了对“中断优先级寄存器”的原子访问。更多信息，请参见“NVIC 编程提示”小节中 NVIC_SetPriority 函数的描述。下表列出了中断、 或 IRQ 序号是如何映射到中断寄存器和对相应的 CMSIS 变量，这些变量对每个中断有一个位。

中断	CMSIS 数组元[1]				
	置位-使能	清零-使能	置位-挂起	清零-挂起	有效位
0-31	ISER[0]	ICER[0]	ISPR[0]	ICPR[0]	IABR[0]
32-63	ISER[1]	ICER[1]	ISPR[1]	ICPR[1]	IABR[1]
64-95	ISER[2]	ICER[2]	ISPR[2]	ICPR[2]	IABR[2]
96-127	ISER[3]	ICER[3]	ISPR[3]	ICPR[3]	IABR[3]

附录表 1-35 中断到中断变量的映射

注[1]：每个数组元对应一个 NVIC 寄存器。例如，数组元 ICER[1]对应 ICER1 寄存器。

附录1. 4. 2. 2 中断置位使能寄存器

ISER0-ISER3 寄存器使能中断，并显示哪些中断被使能。请参见：

- ◇ “NVIC 寄存器汇总”表中寄存器汇总的寄存器属性；
- ◇ “中断到中断变量的映射”表中了解每个寄存器都控制哪些中断。

位分配见下表。

位	名称	功能
[31:0]	SETENA	中断置位-使能位。 写： 0：无影响 1：使能中断。 读： 0：中断被禁能 1：中断被使能。

附录表 1-36 ISER 位分配

如果某个挂起中断被使能，NVIC 根据其优先级激活中断。如果一个中断没有被使能，则这个中断信号的发出会将中断状态变为挂起，但无论其优先级如何，NVIC 永远不会激活此中断。

附录1. 4. 2. 3 中断清零使能寄存器

ICER0-ICER3 寄存器禁止中断，并显示哪些中断被使能。请参见：

◇ “NVIC 寄存器汇总”表中寄存器汇总的寄存器属性；

◇ “中断到中断变量的映射”表中了解每个寄存器都控制哪些中断。

位分配见下表。

位	名称	功能
[31:0]	CLRENA	中断清零-使能位。 写： 0：无影响 1：禁能中断。 读： 0：中断被禁能 1：中断被使能。

附录表 1-37 ICER 位分配

附录1. 4. 2. 4 中断置位挂起寄存器

ISPR0-ISPR3 寄存器强制中断进入挂起状态，并显示哪些中断正在挂起。请参见：

◇ “NVIC 寄存器汇总”表中寄存器汇总的寄存器属性；

◇ “中断到中断变量的映射”表中了解每个寄存器都控制哪些中断。

位分配见下表。

位	名称	功能
[31:0]	SETPEND	中断置位-挂起位。 写： 0：无影响 1：改变中断状态为挂起状态 读： 0：中断未挂起 1：中断挂起

附录表 1-38 ISPR 位分配

注：将 1 写到对应于以下项的 ISPR 位：
一个挂起的中断，无效果；
一个被禁用的中断，设置该中断的状态为挂起。

附录1. 4. 2. 5 中断清零挂起寄存器

ICPR0-ICPR3 寄存器移除中断的挂起状态，并显示有哪些中断在挂起。请参见：

◇ “NVIC 寄存器汇总”表中寄存器汇总的寄存器属性；

◇ “中断到中断变量的映射”表中了解每个寄存器都控制哪些中断。

位分配见下表。

位	名称	功能
[31:0]	CLRPEND	中断清零-挂起位。 写： 0：无影响 1：清除挂起状态的一个中断 读： 0：中断未挂起 1：中断挂起

附录表 1-39 ICPR 位分配

注：将 1 写到一个 ICPR 位不影响相应中断的行为状态。

附录1. 4. 2. 6 中断有效位寄存器

IABR0-IABR3 寄存器指示哪些中断是有效的。请参见：

◇ “NVIC 寄存器汇总”表中寄存器汇总的寄存器属性；

◇ “中断到中断变量的映射”表中了解每个寄存器都控制哪些中断。

位分配见下表。

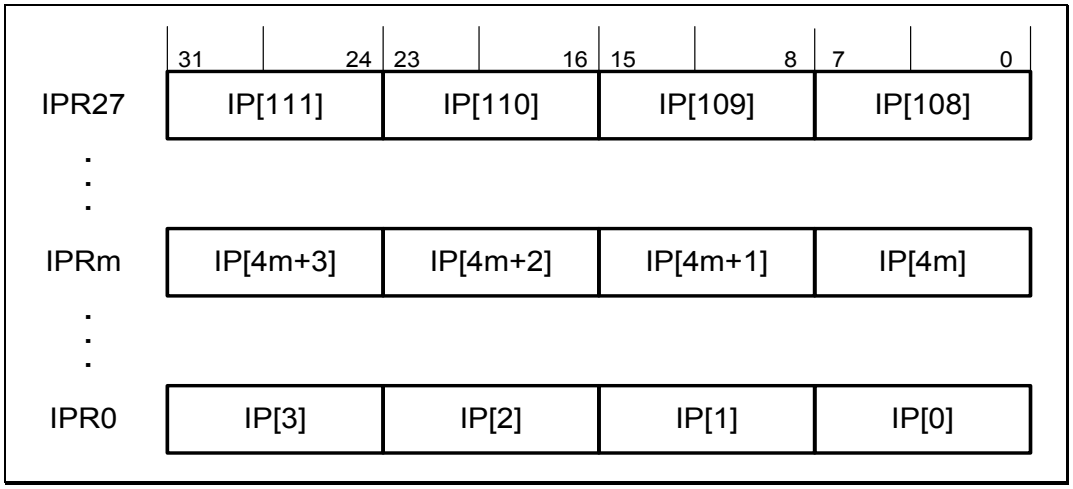
位	名称	功能
[31:0]	ACTIVE	中断有效标志： 0：中断无效 1：中断有效。

附录表 1-40 IABR 位分配

如果相应中断的状态为有效或有效并挂起，则该位读取值为 1。

附录1. 4. 2. 7　中断优先级寄存器

IPR0-IPR27 寄存器为每个中断提供一个 5 位的优先级域。这些寄存器可按字节访问。其属性请参见“NVIC 寄存器汇总”表中寄存器汇总。每个寄存器保存 4 个优先级域，它映射到 CMSIS 中断优先级数组 IP[0]到 IP[111]中的 4 个元素，如下所示：



附录图 1-14　中断优先级寄存器

位	名称	功能
[31:24]	优先级，字节偏移量 3	每个优先级域承载一个优先级值，0 至 31。这个值越低，相应中断的优先级就越高。处理器仅执行每个域的位[7:3]，位 [2:0]读为零且忽略写入值。
[23:16]	优先级，字节偏移量 2	
[15:8]	优先级，字节偏移量 1	
[7:0]	优先级，字节偏移量 0	

附录表 1-41　IPR 位分配

更多关于 IP[0]到 IP[111]中断优先级数组的信息，请参见“中断到中断变量的映射”表，该数组提供了中断优先级的软件视图。

按以下方式查找中断 N 的 IPR 号和字节偏移量：

◇　相应的 IPR 号 M，用 M = N 除以 4 得到；

◇　此寄存器中所需“优先级”域的字节偏移量为 N MOD 4，其中：

- 字节偏移量 0 引用寄存器位[7:0]；
- 字节偏移量 1 引用寄存器位[15:8]；
- 字节偏移量 2 引用寄存器位[23:16]；
- 字节偏移量 3 引用寄存器位[31:24]。

附录1. 4. 2. 8 软件触发中断寄存器

写入 STIR 产生一个软件产生的中断（SGI）。请参见“NVIC 寄存器汇总”表中寄存器汇总了解 STIR 属性。

当 CCR 中 USERSETMPEND 位设置为 1 时，非特权软件可访问 STIR，见“配置和控制寄存器”小节。

注：只有特权软件才能使能对 STIR 的非特权访问。

位分配见下表。

位	域	功能
[31:9]	-	保留。
[8:0]	INTID	所需 SGI 的中断 ID 范围（0~111）。例如，b000000011 指定的是中断 IRQ3。

附录表 1-42 STIR 位分配

附录1. 4. 2. 9 电平触发和脉冲中断

处理器同时支持电平触发中断和脉冲中断。脉冲中断也称为边沿触发中断。

电平触发中断会保持有效状态，直到外设撤销中断信号。通常这是因为 ISR 访问外设，导致它清除中断请求。脉冲中断是一个在处理器时钟上升沿同步采样的中断信号。为了确保 NVIC 检测到中断，外设必须使中断信号保持有效至少一个时钟周期，在此期间 NVIC 可检测到脉冲并锁存中断。

当处理器进入 ISR 时，它自动从中断移除挂起状态，见“中断的软/硬件控制” 小节。对于电平触发中断，如果在处理器从 ISR 返回前信号没有撤销，中断重新变为挂起状态，则处理器必须再次执行 它的 ISR。这就是说，外设可以保持中断一直有效，直到不再需要处理为止。

中断的软/硬件控制

Cortex-M3 锁存全部中断。一个外设中断会因以下原因之一而变为挂起状态：

◇ NVIC 检测到中断信号为高电平且中断为无效状态；

◇ NVIC 检测到一个中断信号的上升沿；

◇ 软件写入相应的中断置位挂起寄存器位，见表“ISPR 位分配”，或写入 STIR，造成一个 SGI 挂起，见表“STIR 位分配”。

一个挂起中断会保持在挂起状态，直到发生以下情况之一：

◇ 处理器进入该中断的 ISR。使中断的状态从挂起变为有效。然后：

- 对于电平触发中断，当处理器从 ISR 返回时，NVIC 采样中断信号。如果信号被使能，则中断变为挂起状态，这可能使处理器立即重新进入 ISR。否则，中断状态变为无效。
- 对于脉冲中断，NVIC 持续监测中断信号，如果有信号脉冲，则中断状态变为挂起和有效。在此情况下，当处理器从 ISR 返回时，中断状态变为挂起，这可能使处理器立即重新进入 ISR。
- 如果处理器在 ISR 中时没有中断信号脉冲，则当处理器从 ISR 返回时，中断状态

变为无效。

◇ 软件写入相应的中断清零挂起寄存器位。对于电平触发中断，如果中断信号仍然有效，则中断状态不变。否则中断状态变为无效。

对于脉冲中断，中断状态变为：

- 无效，如果原状态为挂起；
- 有效，如果原状态为有效和挂起。

附录1. 4. 2. 10 NVIC 设计提示和建议

确保软件使用了正确对齐的寄存器访问。处理器不支持对 NVIC 寄存器的非对齐访问。请参见各个寄存器的描述了解它所支持的访问大小。

即使某个中断被禁用，它也可以进入挂起状态。

在对 VTOR 编程以重新定位向量表以前，要确保对故障处理程序、NMI 和所有使能的异常（如中断）都建立了新向量表的向量表项。更多信息，请参见表“VTOR 位分配”。

NVIC 编程提示

软件使用 CPSIE I 和 CPSID I 指令来使能和禁能中断。CMSIS 提供为这些指令提供了以下内在函数：

```
void __disable_irq(void)  // Disable Interrupts
void __enable_irq(void)  // Enable Interrupts
```

另外，CMSIS 提供了一些 NVIC 控制函数，包括：

CMSIS 中断控制函数	描述
void NVIC_SetPriorityGrouping(uint32_t priority_grouping)	设置优先级分组
void NVIC_EnableIRQ(IRQn_t IRQn)	使能 IRQn
void NVIC_DisableIRQ(IRQn_t IRQn)	禁能 IRQn
uint32_t NVIC_GetPendingIRQ (IRQn_t IRQn)	如果 IRQn 挂起，返回结果为真（IRQ-Number 即 IRQ 序号）
void NVIC_SetPendingIRQ (IRQn_t IRQn)	设置 IRQn 挂起
void NVIC_ClearPendingIRQ (IRQn_t IRQn)	清除 IRQn 挂起状态
uint32_t NVIC_GetActive (IRQn_t IRQn)	返回有效中断的 IRQ 序号
void NVIC_SetPriority (IRQn_t IRQn, uint32_t priority)	设置 IRQn 优先级
uint32_t NVIC_GetPriority (IRQn_t IRQn)	读取 IRQn 优先级
void NVIC_SystemReset (void)	复位系统

附录表 1-43 用于 NVIC 控制的 CMSIS 函数

有关这些函数的更多信息，请参见 CMSIS 文档。

附录1. 4. 3 系统控制模块

系统控制模块（SCB）提供系统执行信息和系统控制。这包括对系统异常的配置、控制和报告。系统控制模块寄存器有：

地址	名称	类型	所需特权	复位值
0xE000E008	ACTLR	RW	特权	0x00000000
0xE000ED00	CPUID	R	特权	0x412FC230
0xE000ED04	ICSR	RW[1]	特权	0x00000000
0xE000ED08	VTOR	RW	特权	0x00000000
0xE000ED0C	AIRCR	RW[1]	特权	0xFA050000
0xE000ED10	SCR	RW	特权	0x00000000
0xE000ED14	CCR	RW	特权	0x00000200
0xE000ED18	SHPR1	RW	特权	0x00000000
0xE000ED1C	SHPR2	RW	特权	0x00000000
0xE000ED20	SHPR3	RW	特权	0x00000000
0xE000ED24	SHCRS	RW	特权	0x00000000
0xE000ED28	CFSR	RW	特权	0x00000000
0xE000ED28	MMSR[2]	RW	特权	0x00
0xE000ED29	BFSR[2]	RW	特权	0x00
0xE000ED2A	UFSR[2]	RW	特权	0x0000
0xE000ED2C	HFSR	RW	特权	0x00000000
0xE000ED34	MMFAR	RW	特权	未定义
0xE000ED38	BFAR	RW	特权	未定义

附录表 1-44 系统控制模块寄存器汇总

注[1]：更多信息，参见寄存器描述。
注[2]：CFSR 的一个子寄存器。

附录1. 4. 3. 1 Cortex-M3 SCB 寄存器的 CMSIS 映射

为了提高软件效率，CMSIS 简化了 SCB 寄存器的表述。在 CMSIS 中，字节数组 SHP[0] 到 SHP[12]对应于寄存器 SHPR1 至 SHPR3。

附录1. 4. 3. 2 辅助控制寄存器（ACTLR）

ACTLR 提供了以下处理器功能的禁用位：

◇ IT 折叠（folding）；
◇ 用于访问默认存储器映射的写缓冲；
◇ 多周期指令的中断。

请参见表“系统控制模块寄存器汇总”中寄存器汇总了解 ACTLR 属性。位分配见表“ACTLR 位分配”。

位	名称	功能
[31:3]	-	保留
[2]	DISFOLD	该位为 1 时，禁能 IT 折叠。更多信息请参见“关于 IT 折叠”小节。
[1]	DISDEFWBUF	该位为 1 时，在默认存储器映射访问期间禁能写缓冲使用。这使得所有的总线故障都是精确总线故障，但却降低了性能，因为存储器的所有存储操作都必须在处理器能够执行下一指令前完成。 注：该位仅影响在 Cortex-M3 处理器中执行的写缓冲。
[0]	DISMCYCINT	该位为 1 时，禁能多个加载和多个存储指令的中断。这增加了处理器的中断延迟，因为所有的 LDM 或 STM 都必须在处理器能够堆栈当前状态并进入中断处理程序前完成。

附录表 1-45 ACTLR 位分配

关于 IT 折叠

在某些情况下，处理器可以在执行 IT 指令时，开始执行某个 IT 块中的第一条指令。此行为 称为 IT 折叠，可提高性能，然而，IT 折叠可能导致循环中的抖动（jitter）。如果某个任务必 须避免抖动，则要在执行任务前将 DISFOLD 位设为 1，禁用 IT 折叠。

附录1. 4. 3. 3 CPUID 基址寄存器

CPUID 寄存器包括处理器部件号、版本和执行信息。其属性请参见表“系统控制模块寄存器汇总”中的寄存器汇总。 位分配见下表。

位	名称	功能
[31:24]	Implementer	实现者代码：0x41 = ARM
[23:20]	Variant	变量号，rnpn 产品修订标识符中的 r 值：0x2 = r2p0
[19:16]	Constant	读为 0xF
[15:4]	PartNo	处理器的部件序号：0xC23 = Cortex-M3
[3:0]	Revision	修订号，rnpn 产品修订标识符中的 p 值：0x0 = r2p0

附录表 1-46 CPUID 寄存器的位分配

附录1. 4. 3. 4 中断控制和状态寄存器

ICSR：

◇ 提供：

- 用于不可屏蔽中断（NMI）异常的置位挂起位；
- 用于 PendSV 和 SysTick 异常的置位挂起位和清零挂起位。

◇ 指示：

- 正在被处理的异常的异常号；
- 有无被抢占的有效异常；
- 具有最高优先级的挂起异常的异常号；

- 是否有挂起中断。

ICSR 属性请参见表“系统控制模块寄存器汇总”中寄存器汇总和表“ICSR 位分配”中的类型描述。位分配见下表。

位	名称	类型	功能
[31]	NMIPENDSET	RW	NMI 置位-挂起位。 写： 0：无影响 1：改变 NMI 异常状态为挂起。 读： 0：NMI 异常未挂起 1：NMI 异常挂起。 因为 NMI 是最高优先级的异常，通常情况下处理器在该位被写入 1 后马上进入 NMI 异常处理程序，进入处理程序将该位清零。当处理器执行某个 NMI 异常处理程序时，只有当 NMI 信号再次有效时，该处理程序对该位的读取结果才返回为 1。
[30:29]	-	-	保留。
[28]	PENDSVSET	RW	PendSV 置位-挂起位。 写： 0：无影响 1：改变 PendSV 异常状态为挂起。 读： 0：PendSV 异常未挂起 1：PendSV 异常挂起。 向该位写入 1 是将 PendSV 异常状态设为挂起的唯一途径。
[27]	PENDSVCLR	W	PendSV 清零-挂起位。 写： 0：无影响 1：移除 PendSV 异常中的挂起状态
[26]	PENDSTSET	RW	SysTick 异常置位-挂起位。 写： 0：无影响 1：改变 SysTick 异常状态为挂起。 读： 0：SysTick 异常未挂起 1：SysTick 异常挂起。
[25]	PENDSTCLR	W	SysTick 异常清零-挂起位。 写： 0：无影响 1：移除 SysTick 异常中的挂起状态。该位为只写位。该位在寄存器上读取值未知。
[24]	-	-	保留。

位	名称	类型	功能
[23]	Reserved for Debug use	R	该位保留用于调试，当处理器不在调试模式时该位读取为零。
[22]	ISRPENDING	R	中断挂起标志，不包括 NMI 和故障： 0：中断未挂起 1：中断挂起。
[21:18]	-	-	保留。
[17:12]	VECTPENDING	R	该位表示最高优先级挂起使能异常的异常序号： 0：无挂起异常 非零值：最高优先级挂起使能异常的异常序号。 该域值包括 BASEPRI 和 FAULTMASK 寄存器的效果，但不包括 PRIMASK 寄存器的任何效果。
[11]	RETTOBASE	R	该位表示有无被抢占的有效异常： 0：有待执行的被抢占的有效异常 1：无有效异常，或当前执行的异常是唯一有效异常。
[10:9]	-	-	保留。
[8:0]	VECTACTIVE[1]	R	包含有效异常序号： 0：线程模式 非零值：当前有效异常的异常序号[1]。 注：该值减 16 可以得到用来索引到中断清零使能、置位使能、清零挂起、置位挂起或优先级寄存器所需的 IRQ 序号，见表“IPSR 位分配”。

附录表 1-47 ICSR 位分配

注[1]：这与 IPSR 位[8:0]的值相同，见表“IPSR 位分配”。

当写入 ICSR 时，如有以下情况，则后果不可预知：

◇ 写入 1 到 PENDSVSET 位并写入 1 到 PENDSVCLR 位；

◇ 写入 1 到 PENDSTSET 位并写入 1 到 PENDSTCLR 位。

附录1. 4. 3. 5 向量表偏移量寄存器

VTOR 表示从存储器地址 0x00000000 开始的向量表基址偏移量。其属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。

位分配见下表。

位	名称	功能
[31:30]	-	保留。
[29:8]	TBLOFF	向量表的基址偏移域。它包含向量表基址与存储器映射底部的偏移量的位[29:8]。 注：位[29]决定向量表是否在编码或 SRAM 存储区： 位[29]有时被称为位 TBLBASE。 0：编码

		1：SRAM。
[7:0]	-	保留。

附录表 1-48 VTOR 位分配

当置位 TBLOFF 时，必须使偏移对准向量表中异常进入的序号。建议的对齐方式为 256 个字，可用于 128 个中断。

注：表对齐要求的含义是表偏移的位[7:0]始终为零。

附录1. 4. 3. 6 应用中断和复位控制寄存器

AIRCR 提供了异常模型的优先级分组控制、数据访问的字节序状态，以及系统的复位控制。其属性请参见表“系统控制模块寄存器汇总”中寄存器汇总和表“AIRCR 位分配”。

如要对此寄存器执行写操作，必须将 0x5FA 写入 VECTKEY 域，否则处理器会忽略写入值。 位分配见下表。

	名称	类型	功能
[31:16]	写：VECTKEYSTAT	RW	寄存器密钥（register key）：读为 0x05FA
	读：VECTKEY		执行写操作时要求向 VECTKEY 写入 0x5FA，否则写入值被忽略。
[15]	ENDIANESS	R	数据字节序位： 0：小端。 1：大端
[14:11]	-	-	保留
[10:8]	PRIGROUP	R/W	中断优先级分组域。该域决定从子优先级中拆分组优先级，见“二进制点”小节。
[7:3]	-	-	保留。
[2]	SYSRESETREQ	W	系统复位请求： 0：无系统复位请求 1：让信号在外部系统有效，表示请求一个复位。该位用于强制对调试设备之外的所有主要设备进行一次大的系统复位。 注：ES32 微控制器支持 SYSRESETREQ。该位读为 0。
[1]	VECTCLRACTIVE	W	该位保留供调试使用。该位读为 0。当向寄存器执行写操作时，必须向该位写入 0，否则该行为不可预知。
[0]	VECTRESET	W	该位保留供调试使用。该位读为 0。当向寄存器执行写操作时，必须向该位写入 0，否则该行为不可预知。

附录表 1-49 AIRCR 位分配

二进制点

PRIGROUP 域指示了二进制点的位置，该点将“中断优先级寄存器”中 PRI_n 域分割为独立的组优先级和子优先级域。见下表列出了 PRIGROUP 值如何控制此分割。

	中断优先级级别值，PRI_N[7:0]			数目	
PRIGROUP	二进制点[1]	组优先级位	子优先级位	组优先级	子优先级

	中断优先级级别值，PRI_N[7:0]			数目	
PRIGROUP	二进制点[1]	组优先级位	子优先级位	组优先级	子优先级
b011	bxxxx.0000	[7:4]	无	16	1
b100	bxxx.y0000	[7:5]	[4]	8	2
b101	bxx.yy0000	[7:6]	[5:4]	4	4
b110	bx.yyy000	[7]	[6:4]	2	8
b111	b.yyyy000	无	[7:4]	1	16

附录表 1-50 优先级分组

注[1]：PRI_n[7:0]域显示二进制点。X 指示一个组优先级域位，y 指示一个子优先级域位。位[3:0]不在 ES32 微控制器中使用。

备注：确定一个异常是否抢占时，只使用组优先级域，见“中断优先级分组”小节。

附录1. 4. 3. 7 系统控制寄存器

SCR 控制着进入和离开低功耗状态的特性。其属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。位分配见下表。

位	名称	功能
[31:5]	-	保留。
[4]	SEVONPEND	在挂起位上发送事件： 0：仅使能的中断或事件可以唤醒处理器，禁能的中断不能唤醒处理器。 1：使能的事件和所有中断，包括禁能的中断，都可以唤醒处理器。 当一个事件或中断进入挂起状态，事件信号将处理器从 WFE 中唤醒。 如果处理器没有在等待一个事件，该事件被记录并影响下一 WFE。 该处理器也可以在执行 SEV 指令或一个外部事件时被唤醒。
[3]	-	保留。
[2]	SLEEPDEEP	控制处理器是否使用睡眠或深度睡眠作为其低功率模式： 0：睡眠 1：深度睡眠。
[1]	SLEEPONEXIT	表示当从处理模式返回到线程模式时开始“退出时睡眠”： 0：返回到线程模式时不进入到睡眠模式 1：从 ISR 返回时进入睡眠或深度睡眠模式。 将该位设为 1 会使能一个中断驱动应用程序以避免返回到空的主应用程序中。
[0]	-	保留。

附录表 1-51 SCR 位分配

附录1. 4. 3. 8 配置和控制寄存器

CCR 控制着线程模式的进入，并使能：

◇ 通过 FAULTMASK 升级的 NMI、HardFault 和故障处理程序，以忽略总线故障；

◇ 被零除和非对齐访问的捕获；

◇ 非特权软件对 STIR 的访问，见表“STIR 位分配”。

CCR 属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。

位分配见下表。

位	名称	功能
[31:10]	-	保留。
[9]	STKALIGN	表示进入异常时的堆栈对齐。 0：4 字节对齐 1：8 字节对齐 进入异常时，处理器使用压入堆栈的 PSR 位[9]来指示堆栈对齐。从异常返回时，这个堆栈位被用来恢复正确的堆栈对齐。
[8]	BFHFNMIGN	使能时，使得以优先级位-1 或-2 运行的处理程序忽略加载和存储指令引起的数据总线故障。它用于 HardFault、NMI 和 FAULTMASK 升级处理程序中： 0：加载和存储指令引起的数据总线故障会引起锁定。 1：以优先级-1 或-2 运行的处理程序忽略加载和存储指令引起的数据总线故障。 仅在处理程序和其数据处于绝对安全的存储器时将该位设为 1。一般将该位用于探测系统设备和桥接器以检测并纠正控制路径问题。
[7:5]	-	保留。
[4]	DIV_0_TRP	当处理器进行除 0 操作（SDIV 或 UDIV 指令）时，会导致故障或停止。 0：不捕获除以零故障 1：捕获除以零故障。 当该位设为 0 时，除以零返回的商数为 0。
[3]	UNALIGN_TRP	使能非对齐访问捕获： 0：不捕获非对齐半字和字访问 1：捕获非对齐半字和字访问。 如果该位设为 1，非对齐访问产生一个使用故障。无论 UNALIGN_TRP 是否设为 1，非对齐的 LDM、STM、LDRD 和 STRD 指令总是出错。
[2]	-	保留。
[1]	USERSETMPEND	使能对 STIR 的无特权软件访问，参见表“STIR 位分配”。 0：禁能 1：使能 指示处理器如何进入线程模式：
[0]	NONEBASETHRDENA	0：处理器仅在没有有效异常时才能够进入线程模式。 1：处理器可以从 EXC_RETURN 值控制下的任何级别进入线程模式，参见 “异常返回”小节。

附录表 1-52 CCR 位分配

附录1. 4. 3. 9　系统处理程序优先级寄存器

SHPR1-SHPR3 寄存器用于设置有可配置优先级异常处理程序的优先级 0 到 31。

SHPR1-SHPR3 可按字节访问。这些寄存器的属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。

系统故障处理程序，以及各处理程序的优先级域和寄存器如下：

处理程序	域	寄存器描述
存储器管理故障	PRI_4	见表“SHPR1 寄存器的位分配”
总线故障	PRI_5	
使用故障	PRI_6	
SVCall	PRI_11	见表“SHPR2 寄存器的位分配”
PendSV	PRI_14	见表“SHPR3 寄存器的位分配”
SysTick	PRI_15	

附录表 1-53　系统故障处理程序优先级域

每个 PRI_N 域为 8 位宽，但是处理器仅实现每个域的位[7:3]，位[2:0]读取值为零并忽略写入值。

系统处理程序优先级寄存器 1

位分配见下表。

位	名称	功能
[31:24]	PRI_7	保留
[23:16]	PRI_6	系统处理程序 6 的优先级，使用故障
[15:8]	PRI_5	系统处理程序 5 的优先级，总线故障
[7:0]	PRI_4	系统处理程序 4 的优先级，存储器管理故障

附录表 1-54　SHPR1 寄存器的位分配

系统处理程序优先级寄存器 2

位分配见下表。

位	名称	功能
[31:24]	PRI_11	系统处理程序 11 的优先级，SVCall
[23:0]	-	保留

附录表 1-55　SHPR2 寄存器的位分配

系统处理程序优先级寄存器 3

位分配见下表。

位	名称	功能
[31:24]	PRI_15	系统处理程序 15 的优先级，SysTick 异常
[23:16]	PRI_14	系统处理程序 14 的优先级，PendSV
[15:0]	-	保留

附录表 1-56 SHPR3 寄存器的位分配

附录1. 4. 3. 10 系统处理程序控制和状态寄存器

SHCSR 使能系统处理程序，并指示：

◇ 总线故障、存储器管理故障和 SVC 异常的挂起状态；

◇ 系统处理程序的有效状态。

SHCSR 属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。位分配见下表。

位	名称	功能
[31:19]	-	保留
[18]	USGFAULTENA	使用故障使能位，设为 1 时使能[1]
[17]	BUSFAULTENA	总线故障使能位，设为 1 时使能[1]
[16]	MEMFAULTENA	存储器管理故障使能位，设为 1 时使能[1]
[15]	SVCALLPENDED	SVC 调用挂起位，如果异常挂起，该位读为 1[2]
[14]	BUSFAULTPENDED	总线故障异常挂起位，如果异常挂起，该位读为 1[2]
[13]	MEMFAULTPENDED	存储器管理故障异常挂起位，如果异常挂起，该位读为 1[2]
[12]	USGFAULTPENDED	使用故障异常挂起位，如果异常挂起，该位读为 1[2]
[11]	SYSTICKACT	SysTick 异常有效位，如果异常有效，该位读为 1[3]
[10]	PENDSVACT	PendSV 异常有效位，如果异常有效，该位读为 1
[9]	-	保留
[8]	MONITORACT	调试监控有效位，如果调试监控有效，该位读为 1
[7]	SVCALLACT	SVC 调用有效位，如果 SVC 调用有效，该位读为 1
[6:4]	-	保留
[3]	USGFAULTACT	使用故障异常有效位，如果异常有效，该位读为 1
[2]	-	保留
[1]	BUSFAULTACT	总线故障异常有效位，如果异常有效，该位读为 1
[0]	MEMFAULTACT	存储器管理故障异常有效位，如果异常有效，该位读为 1

附录表 1-57 SHCSR 位分配

注[1]：使能位，设为 1 使能异常，或设为 0 禁能异常。

注[2]：挂起位，如果异常挂起，该位读为 1，或如果异常未挂起，该位读为 0。可以向这些位写入值以改变 异常的挂起状态。

注[3]：有效位，如果异常有效，该位读为 1，或如果异常无效，该位读为 0。可以向这些位写入值以改变异 常的有效状态，但需关注本章节中的的“注意”部分。

备注：如果禁用某个系统处理程序，并发生了对应的故障，则处理器将该故障作为 HardFault。

可写入此寄存器来改变系统异常的挂起或有效状态。OS 内核可以写入有效位进行上下文切换，从而改变当前异常的类型。

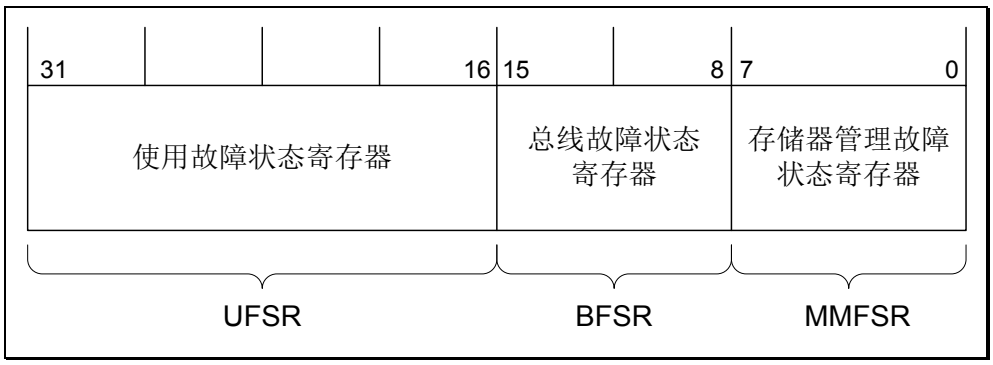
注意

◇ 如果软件改变了此寄存器的某个有效位的值，而未正确调整入栈的内容，则可能会使处理器产生一个故障异常。要确保写入此寄存器的软件继续并在随后恢复当前的有效状态。

◇ 使能了系统处理程序后，如果必须改变此寄存器中某个位的值，则必须采用读取-修改-写入步骤，以确保只改变需要的位。

附录1. 4. 3. 11 可配置故障状态寄存器

CFSR 指示存储器管理故障、总线故障或使用故障的原因。CFSR 属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。位分配为：



附录图 1-15 可配置故障状态寄存器

以下各小节描述了构成 CFSR 的子寄存器：

◇ 表“MMFSR 位分配”

◇ 表“BFSR 位分配”

◇ 表“UFSR 位分配”

CFSR 可按字节访问。可如下访问 CFSR 或其子寄存器：

◇ 通过到 0xE000ED28 的字访问来访问整个 CFSR；

◇ 通过到 0xE000ED28 的字节访问来访问 MMFSR；

◇ 通过到 0xE000ED28 的半字访问来访问 MMFSR 和 BFSR；

◇ 通过到 0xE000ED29 的字节访问来访问 BFSR；

◇ 通过到 0xE000ED2A 的半字访问来访问 UFSR。

存储器管理故障状态寄存器

MMFSR 中标志指示存储器访问故障的原因。位分配见下表。

位	名称	功能
[7]	MMARVALID	存储器管理故障地址寄存器（MMAR）有效标志： 0：MMAR 中的值不是一个有效故障地址 1：MMAR 中保留一个有效故障地址。 如果发生了一个存储器管理故障，并由于优先级的原因升级成一个 HardFault，那么 HardFault 处理程序必须将该位设为 0。 这样可以避免在返回到 MMAR 值已被覆写的压入堆栈的有效存储器管理故障处理程序时出现问题。
[6:5]	-	保留。
[4]	MSTKERR	进入异常时的入栈操作引起的存储器管理故障： 0：无入栈故障 1：进入异常时的入栈操作引起了一个或一个以上的访问违犯。 当该位设为 1 时，依然要对 SP 进行调节，并且堆栈的上下文区域的值可能不正确。处理器没有向 MMAR 中写入故障地址。
[3]	MUNSTKERR	异常返回时的出栈操作引起的存储器管理故障： 0：无出栈故障 1：异常返回时的出栈操作已引起一个或一个以上的访问违犯 该故障与处理程序相连，这意味着当该位为 1 时，原始的返回堆栈仍然存在。处理器不能对返回失败的 SP 进行调节，并且不会执行新的存储操作。 处理器没有向 MMAR 中写入故障地址。
[2]	-	保留
[1]	DACCVIOL	数据访问违犯标志： 0：无数据访问违犯故障 1：处理器试图在不允许执行操作的位置上进行加载和存储。 当该位为 1 时，异常返回的压入堆栈的 PC 值指向出错指令。 处理器已在 MMAR 中加载了目标访问的地址。
[0]	IACCVIOL	指令访问违犯标志： 0：无指令访问违犯错误 1：处理器试图从不允许执行操作的位置上进行指令获取。 即使 MPU 被禁能，这一故障也会在每次访问 XN 区时发生。 当该位为 1 时，异常返回的压入堆栈的 PC 值指向出错指令。处理器没有向 MMAR 中写入故障地址。

附录表 1-58 MMFSR 位分配

总线故障状态寄存器

BFSR 中标志指示总线访问故障原因。位分配见下表。

位	名称	功能
[7]	BFARVALID	总线故障地址寄存器（BFAR）有效标志： 0：BFAR 中的值不是有效故障地址 1：BFAR 中保留一个有效故障地址。

位	名称	功能
		在地址已知的总线故障发生后处理器将该位设为 1。该位可以被其他故障清零，例如之后发生的存储器管理故障。
		如果发生总线故障，并由于优先级原因升级为一个 HardFault，那么 HardFault 处理程序必须将该位设为 0。当返回至一个 BFAR 值已被覆写的压入堆栈的有效总线故障处理程序时，这样做可以防止问题的出现。
[6:5]	-	保留。
[4]	STKERR	进入异常时的入栈操作引起的总线故障：
		0：无入栈故障
		1：进入异常时的入栈操作已引起一个或一个以上的总线故障。
		当处理器将该位设为 1 时，依然要对 SP 进行调节，并且堆栈的上下文区域的值可能不正确。处理器没有向 BFAR 中写入故障地址。
[3]	UNSTKERR	异常返回时的出栈操作引起的总线故障：
		0：无出栈故障
		1：异常返回时的出栈操作已引起一个或一个以上的总线故障。
		该故障与处理程序相连， 这意味着当处理器将该位设为 1 时，原始的返回堆栈仍然存在。处理器不能对返回失败的 SP 进行调节，并且不会执行新的存储操作，也未向 BFAR 中写入故障地址。
[2]	IMPRECISERR	非精确数据总线错误：
		0：无非精确数据总线错误
		1：已发生一个数据总线错误，但是堆栈帧中的返回地址与引起错误的指令无关。
		当处理器将该位设为 1 时，不向 BFAR 中写入故障地址。
		这是一个异步故障。因此，如果在当前进程的优先级高于总线故障优先级时检测到该故障，总线故障被挂起并仅在处理器从所有更高优先级进程中返回时开始变为有效。如果在处理器进入非精确总线故障的处理程序前发生一个精确故障，那么处理程序同时对 IMPRECISERR 和其中一个精确故障状态位进行检测，判断它们是否置位为 1。
[1]	PRECISERR	精确数据总线错误：
		0：无精确数据总线错误
		1：已发生一个数据总线错误，且异常返回的压入堆栈的 PC 值指向引起故障的指令。
		当处理器将该位设为 1 时，向 BFAR 中写入故障地址。
[0]	IBUSERR	指令总线错误：
		0：无指令总线错误
		1：指令总线错误。
		处理器检测到预取指令时的指令总线错误，但仅在其试图签发故障指令时才将 IBUSERR 标志设为 1。
		当处理器将该位设为 1 时，不向 BFAR 中写入故障地址。

附录表 1-59 BFSR 位分配

使用故障状态寄存器

UFSR 指示使用故障的原因。位分配见下表。

位	名称	功能
[15:10]	-	保留。
[9]	DIVBYZERO	除以零使用故障： 0：无除以零故障或除以零捕获未使能 1：处理器已执行 SDIV 或 UDIV 指令（除以零）。 当处理器将该位设为 1 时，异常返回的压入堆栈的 PC 值指向执行除以零的指令。 通过将 CCR 中的 DIV_0_TRP 位设为 1 使能除以零捕获，参见表“CCR 位分配”。
[8]	UNALIGNED	非对齐访问使用故障： 0：无非对齐访问故障，或非对齐访问捕获未使能 1：处理器已进行了一次非对齐的存储器访问。 通过将 CCR 中的 UNALIGN_TRP 位设为 1 来使能非对齐访问捕获，参见表 “CCR 位分配”。 非对齐的 LDM、STM、LDRD 和 STRD 指令总是出错，与 UNALIGN_TRP 的设置无关。
[7:4]	-	保留。
[3]	NOCP	无协处理器使用故障。处理器不支持协处理器指令： 0：试图访问一个协处理器未引起使用故障 1：处理器已试图访问一个协处理器。
[2]	INVPC	EXC_RETURN 的无效 PC 加载引起的无效 PC 加载使用故障： 0：没有发生无效 PC 加载使用故障 1：处理器已试图将 EXC_RETURN 非法载入 PC，作为一个无效的上下文或一个无效的 EXC_RETURN 值。 当该位被设为 1 时，异常返回的压入堆栈的 PC 值指向尝试执行非法 PC 加载的指令。
[1]	INVSTATE	无效状态使用故障： 0：未发生无效状态使用故障 1：处理器已试图执行一个非法使用 EPSR 的指令。 当该位设为 1 时，异常返回的压入堆栈的 PC 值指向一个尝试非法使用 EPSR 的指令。 如果一个未定义的指令使用了 EPSR，则该位不被置位为 1。
[0]	UNDEFINSTR	未定义的指令使用故障： 0：无未定义的指令使用故障 1：处理器已试图执行一个未定义的指令。当该位设为 1 时，异常返回的压入堆栈的 PC 值指向未定义的指令。 未定义的指令是一条不能被处理器译码的指令。

附录表 1-60 UFSR 位分配

注：UFSR 位是粘性（sticky）位。这就是说，当一个或多个故障发生时，相关的位被设为 1。被设为 1 的位只能

通过向该位写入 1 或复位才能清零。

附录1. 4. 3. 12 HardFault状态寄存器

HFSR 提供关于激活 HardFault 处理程序的事件的信息。其属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。

此寄存器为可读并通过写入清零。也就是说，此寄存器中的位可正常读取，但向任何位写入 1 会将该位清零。位分配见下表。

位	名称	功能
[31]	DEBUGEVT	保留供调试使用。对寄存器执行写操作时，必须向该位写入 0；否则，该行为不可预知。
[30]	FORCED	指示一个强制的 HardFault，该 HardFault 由一个优先级可配置且无法处理的故障升级而来，这一故障无法处理是由于优先级或可配置故障被禁能的原因：
		0：无强制的 HardFault
		1：强制的 HardFault。
		当该位设为 1 时，HardFault 处理程序必须读其他故障状态寄存器以找出故障原因。
[29:2]	-	保留。
[1]	VECTTBL	指示一个在异常处理过程中读向量表而引起的总线故障：
		0：读向量表未引起总线故障
		1：读向量表引起了总线故障。
		这一错误通常情况下都由 HardFault 处理程序来处理。 当该位设为 1 时，异常返回的压入堆栈的 PC 值指向被异常抢占的指令。
[0]	-	保留。

附录表 1-61 HFSR 位分配

注：HFSR 位是粘性（sticky）位。这就是说，当一个或多个故障发生时，相关的位被设为 1。被设为 1 的位，只能通过向该位写入 1 或复位才能清零。

附录1. 4. 3. 13 存储器管理故障地址寄存器

MMFAR 包括产生存储器管理故障的位置的地址。其属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。位分配为：

位	名称	功能
[31:0]	ADDRESS	当 MMFSR 的位 MMARVALID 设为 1 时，该域中保存的是产生存储器管理故障的单元地址。

附录表 1-62 MMFAR 位分配

当出现非对齐访问故障时，地址就是实际出错的地址。由于单个读或写指令可能被分割为多个对齐的访问，故障地址可能是请求访问尺寸范围内的任何地址。

MMFSR 中的标志指示故障原因以及 MMFAR 中的值是否有效。请参见表“MMFSR 位

分配”。

附录1. 4. 3. 14 总线故障地址寄存器

BFAR 包括产生总线故障的位置的地址。其属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。位分配为：

位	名称	功能
[31:0]	ADDRESS	当 BFSR 的位 BFARVALID 设为 1 时，该域中保存的是产生总线故障的单元地址。

附录表 1-63 BFAR 位分配

当发生非对齐访问故障时，BFAR 中地址为指令所请求的地址，即使该地址不是故障的地址。

BFSR 中的标志用于指示故障原因以及 BFAR 中的值是否有效。请参见表“BFSR 位分配”。

附录1. 4. 3. 15 系统控制模块设计提示和建议

要确保软件使用正确尺寸的对齐访问来访问系统控制模块寄存器：

◇ 除了 CFSR 和 SHPR1-SHPR3，必须使用对齐的字访问；

◇ 对于 CFSR 和 SHPR1-SHPR3，可使用字节或对齐的半字或字访问。

处理器不支持对系统控制模块寄存器的非对齐访问。

在故障处理程序中，为了确定真实的出错地址：

1. 读取并保存 MMFAR 或 BFAR 值。
2. 读取 MMFSR 中 MMARVALID 位，或 BFSR 中 BFARVALID 位。仅在此位为 1 时，MMFAR 或 BFAR 地址才有效。

软件必须按照此顺序，因为可能会有另一个更高优先级的异常改变 MMFAR 或 BFAR 的值。 例如，如果更高优先级的处理程序抢占了当前故障处理程序，则其他故障可能会改变 MMFAR 或 BFAR 值。

附录1. 4. 4 系统定时器 SysTick

处理器有一个 24 位系统定时器 SysTick，它从重载值递减计数至零，在下一时钟边沿重载（绕回）加载寄存器中的值，然后在后续的时钟中递减计数。

注：当处理器停止调试时，计数器不会递减计数。

系统定时器寄存器有：

地址	名称	类型	所需特权	复位值
0xE000E010	CTRL	RW	特权	0x00000004
0xE000E014	LOAD	RW	特权	0x00000000
0xE000E018	VAL	RW	特权	0x00000000
0xE000E01C	CALIB	R	特权	0x000F423F[1]

附录表 1-64 系统定时器寄存器汇总

注[1]：SysTick 校准值。该值特用于 ES32 微控制器。

附录1. 4. 4. 1 SysTick 控制和状态寄存器

SysTick CTRL 寄存器用于使能 SysTick 特性。其属性请参见表“系统定时器寄存器汇总”中寄存器汇总。位分配见下表。

位	名称	功能
[31:17]	-	保留。
[16]	COUNTFLAG	从上次读取定时器开始，如果定时器计数到 0，则返回 1。
[15:3]	-	保留。
[2]	CLKSOURCE	指示时钟源： 0：外部时钟 1：处理器时钟。
[1]	TICKINT	使能 SysTick 异常请求： 0：向下计数至 0 不会使能 SysTick 异常请求 1：向下计数至 0 会使能 SysTick 异常请求。软件可以使用 COUNTFLAG 来判断 SysTick 是否曾计数到 0。
[0]	ENABLE	使能计数器： 0：计数器禁能 1：计数器使能。

附录表 1-65 SysTick 控制寄存器的位分配

当 ENABLE 设为 1 时，计数器从加载寄存器加载重载（RELOAD）值，然后递减计数。达到 0 时，它将 COUNTFLAG 设为 1，并可选择根据 TICKINT 的值使能 SysTick。然后它再次加载重载值并开始计数。

附录1. 4. 4. 2 SysTick 重载值寄存器

加载寄存器规定了载入 VAL 寄存器的起始值。其属性请参见表“系统定时器寄存器汇总”中寄存器汇总。位分配见下表。

位	名称	功能
[31:24]	-	保留。
[23:0]	RELOAD	当计数器使能且计数器值到 0 时要载入 VAL 寄存器的值，参见“计算重载值”小节。

附录表 1-66 加载寄存器的位分配

计算重载值

重载值可以是 0x00000001 至 0x00FFFFFF 范围内的任意值。可以以 0 为起始值，但这样没有效果，因为 SysTick 异常请求和 COUNTFLAG 在从 1 计数到 0 时激活。

根据其用途，计算得到重载值：

◇ 如要产生一个具有 N 个处理器时钟周期的连拍式（multi-shot）定时器，使用重载值 N-1。例如，如果每 100 时钟脉冲需要有一个 SysTick 中断，则将重载设到 99。

◇ 如要提供一个经过了 N 个处理器时钟周期延迟后的单次触发，则使用重载值 N。例如，如果要求 400 时钟脉冲后发生 SysTick 中断，则将重载设到 400。

附录1. 4. 4. 3 SysTick 当前值寄存器

VAL 寄存器包含 SysTick 计数器的当前值。其属性请参见表“系统定时器寄存器汇总”中寄存器汇总。位分配见下表。

位	名称	功能
[31:24]	-	保留。
[23:0]	CURRENT	读取返回 SysTick 计数器的当前值。 向该寄存器写入任意值都可以将该域清零，还会导致 SysTick CTRL.COUNTFLAG 位清零。

附录表 1-67 VAL 寄存器的位分配

附录1. 4. 4. 4 SysTick 校准值寄存器

CALIB 寄存器指示 SysTick 的校准性质。其属性请参见表“系统定时器寄存器汇总”中寄存器汇总。位分配见下表。

位	名称	功能
[31]	NOREF	是否有一个可用的单独参考时钟。该位的值在出厂时已预设，固定为 0。
[30]	SKEW	TENMS 的值是否精确。这会影响 SysTick 作为一个软件实时时钟的适合性。该位的值在出厂时已预设，固定为 1。
[29:24]	-	保留。
[23:0]	TENMS	该位的值在出厂时已预设，固定为 120000。

附录表 1-68 CALIB 寄存器的位分配

如果所用频率与工厂预设值的频率不同，则需要根据处理器时钟或外部时钟频率计算校

准值。

附录1. 4. 4. 5 SysTick 设计提示和建议

SysTick 计数器使用处理器时钟运行。如果 SysTick 计数器靠其运行时，为了低功率模式而停止了此时钟信号，则 SysTick 计数器将停止。

要确保软件使用对齐的字访问来访问 SysTick 寄存器。

附录1. 4. 5 存储器保护单元

本节介绍存储器保护单元（MPU）。

MPU 将存储器映射划分为多个区，并定义了每个区的位置、大小、访问权限和存储器属性。

它支持：

- ◇ 每个区独立的属性设置；
- ◇ 重叠区；
- ◇ 存储器属性导出到系统。

存储器属性影响对区的存储器访问的行为。Cortex-M3 MPU 定义了：

- ◇ 8 个单独的存储区 0-7；
- ◇ 1 个背景区。

当存储区重叠时，存储器访问受序号最大的区的属性影响。例如，区 7 的属性要优先于任何与区 7 重叠的区。

背景区具有与默认存储器映射相同的存储器访问属性，但只能从特权软件访问。

Cortex-M3 MPU 存储器映射是统一的。这就是说指令访问和数据访问具有相同的区设置。如果某个程序访问被 MPU 禁止的存储单元，则处理器产生一个存储器管理故障。这会导致故障异常，并可能导致 OS 环境中的进程终止。

在 OS 环境中，内核可根据要执行的进程，动态地更新 MPU 区设置。通常，嵌入式 OS 使 用 MPU 进行存储器保护。

MPU 区的配置基于存储器的类型，见“存储区、类型和属性”小节。

下表列出了可能的 MPU 区属性。它们包括与多数微控制器实现无关的“可共享性”和缓存行为属性。该类实现的编程指南见表“微控制器的存储器属性”。

存储器类型	可共享性	其他属性	描述
非常有序	-	-	所有对非常有序存储器的访问都按照 程序顺序进行。所有非常有序区都假设是可共享的。
设备	可共享	-	一些处理器共享的存储器映射的外设。
	非共享	-	仅供一个处理器使用的存储器映射的外设。
正常	可共享	不可缓存	几个处理器共享的正常存储器。
		直写可缓存	
		写回可缓存	
	非共享	不可缓存	仅供一个处理器使用的正常存储器。
		直写可缓存	
		写回可缓存	

附录表 1-69 存储器属性汇总

使用 MPU 寄存器来定义 MPU 区及其属性。MPU 寄存器有：

地址	名称	类型	所需特权	复位值	描述
0xE000ED90	TYPE	R	特权	0x00000800	见表“类型寄存器的位分配”
0xE000ED94	CTRL	RW	特权	0x00000000	见表“MPU 控制寄存器的位分配”
0xE000ED98	RNR	RW	特权	0x00000000	见表“RNR 位分配”
0xE000ED9C	RBAR	RW	特权	0x00000000	见表“RBAR 位分配”
0xE000EDA0	RASR	RW	特权	0x00000000	见表“RASR 位分配”
0xE000EDA4	RBAR_A1	RW	特权	0x00000000	RBAR 的别名，参见表“RBAR 位分配”
0xE000EDA8	RASR_A1	RW	特权	0x00000000	RASR 的别名，参见表“RASR 位分配”
0xE000EDAC	RBAR_A2	RW	特权	0x00000000	RBAR 的别名，参见表“RBAR 位分配”
0xE000EDB0	RASR_A2	RW	特权	0x00000000	RASR 的别名，参见表“RASR 位分配”
0xE000EDB4	RBAR_A3	RW	特权	0x00000000	RBAR 的别名，参见表“RBAR 位分配”
0xE000EDB8	RASR_A3	RW	特权	0x00000000	RASR 的别名，参见表“RASR 位分配”

附录表 1-70 MPU 寄存器汇总

附录1. 4. 5. 1 MPU 类型寄存器

类型（TYPE）寄存器指示是否存在 MPU，如果存在，它支持多少个区。该寄存器属性请参 见表“MPU 寄存器汇总”中的寄存器汇总。位分配见下表。

位	名称	功能
[31:24]	-	保留。
[23:16]	IREGION	指示支持的 MPU 指令区数目。 总是包含 0x00。 MPU 存储器映射是统一的，描述参见 DREGION 域。
[15:8]	DREGION	指示支持的 MPU 数据区数目：
		0x08：8 个 MPU 区。
[7:0]	-	保留。
[0]	SEPARATE	指示支持统一的或单独的指令和数据存储器映射：
		0：统一的。

附录表 1-71 类型寄存器的位分配

附录1. 4. 5. 2 MPU 控制寄存器

MPU CTRL 寄存器用于：

◇ 使能 MPU；

◇ 使能默认存储器映射的背景区；

◇ 在 HardFault、不可屏蔽中断（NMI）和 FAULTMASK 升级的处理程序中时，使能 MPU 的使用。

MPU CTRL 的属性请参见表“MPU 寄存器汇总”中的寄存器汇总。位分配见下表。

位	名称	功能
[31:3]	-	保留。
[2]	PRIVDEFENA	使能供特权软件访问使用的默认存储器映射：
		0：如果 MPU 使能，禁能默认存储器映射的使用。当任意存储器访问一个未被任何使能区覆盖的单元时会导致故障产生。
		1：如果 MPU 使能，使能默认存储器映射的使用，将其作为背景区供特权软件访问使用。
		使能时，背景区充当区号-1。所有已定义且使能的区的优先级都高于这个缺省映射。 如果 MPU 禁能，处理器忽略该位的写入值。
[1]	HFNMIENA	在处于 HardFault、NMI 和 FAULTMASK 处理程序时，该位使能 MPU。
		当 MPU 使能时：
		0：在处于 HardFault、NMI 和 FAULTMASK 处理程序时，MPU 被禁能，与 ENABLE 位的值无关。
		1：在处于 HardFault、NMI 和 FAULTMASK 处理程序时，MPU 被使能。
		当 MPU 禁能时，如果该位设为 1，那么行为不可预知。
[0]	ENABLE	使能 MPU：
		0：MPU 禁能
		1：MPU 使能。

附录表 1-72 MPU 控制寄存器的位分配

当 ENABLE 和 PRIVDEFENA 都设置为 1 时：

◇ 对于特权访问，默认存储器映射“存储器模型”小节所述。特权软件对任何不以使能的存储区为地址的访问，其行为都如默认存储器映射的定义。

◇ 非特权软件对任何不以使能的存储区为地址的访问都会导致一个存储器管理故障。

◇ 无论 ENABLE 位的值如何，XN 和非常有序规则总是适用于“系统控制空间”。

当 ENABLE 位设为 1 时，存储器映射的至少一个区必须被使能，以便让系统能工作，除非 PRIVDEFENA 位设为 1。如果 PRIVDEFENA 位设为 1，且没有区被使能，则只有特权软件可工作。

当 ENABLE 位设为 0 时，系统使用默认存储器映射。这与没有实施 MPU 的情况具有相同的存储器属性，见表“存储器访问行为”。默认存储器映射适用于特权软件和非特权软件的访问。

当使能 MPU 时，对“系统控制空间”和向量表的访问总是被允许。其他区域的访问许可要看存储区的情况，以及 PRIVDEFENA 是否设为 1。

除非 HFNMIENA 设为 1，否则当处理器正为一个优先级为-1 或-2 的异常执行处理程序时，MPU 不被使能。只有在处理一个 HardFault 或 NMI 异常，或当 FAULTMASK 被使能时，才可能有这样的优先级。设置 HFNMIENA 位到 1，会使能运行在这两个优先级的 MPU。

附录1. 4. 5. 3 MPU 区号寄存器

RNR 选择哪个存储区被 RBAR 和 RASR 寄存器引用。其属性请参见表“MPU 寄存器汇总”中的寄存器汇总。

位分配见下表。

位	名称	功能
[31:8]	-	保留。
[7:0]	REGION	指示被 RBAR 和 RASR 寄存器引用的 MPU 区。 MPU 支持 8 个存储区，因此，该域的允许值范围为 0~7。

附录表 1-73 RNR 位分配

通常，在访问 RBAR 或 RASR 前，要将所需的区号写入该寄存器。但是，可通过写入 RBAR 将 VALID 置位为 1 而改变区号，见表“RBAR 位分配”。此写入更新 REGION 域的值。

附录1. 4. 5. 4 MPU 区基址寄存器

RBAR 定义了 RNR 所选择 MPU 区的基址，并可以更新 RNR 的值。其属性请参见表“MPU 寄存器汇总”中的寄存器汇总。

将 VALID 置位为 1 写入 RBAR，可改变当前区号和更新 RNR。位分配见下表。

位	名称	功能
[31:N]	ADDR	区基址域。N 值取决于区域的大小。更多详情请参见 “ADDR 域”小节。
[(N-1):5]	-	保留。
[4]	VALID	MPU 区号有效位： 写： 0：RNR 保持不变且处理器： 更新 RNR 中规定区域的基址 忽略 REGION 域的值 1：处理器： 将 RNR 的值更新为 REGION 域的值 更新 REGION 域规定区域的基址。 读取值总是为 0。
[3:0]	REGION	MPU 区的域： 有关写操作行为，参见 VALID 域的描述。 读取该位返回由 RNR 设定的当前区号。

附录表 1-74 RBAR 位分配

ADDR 域

ADDR 域为 RBAR 的位[31:N]。这个区域的大小由 RASR 中的 SIZE 域指定，它定义了 N 的值：

N = Log2（单位为字节的区大小），

如果 RASR 中区的大小被配置为 4GB，则没有有效的 ADDR 域。在此情况下，区占用了全部的存储器映射，基址为 0x00000000。

基址按区的大小调整。例如一个 64 kB 的区必须对齐 64 kB 的倍数，例如，在 0x00010000 或 0x00020000。

附录1. 4. 5. 5 MPU 区的属性和大小寄存器

RASR 定义了区的大小和 RNR 所指定的 MPU 区的存储器属性，并使能该区及其子区。此寄存器属性请参见表“MPU 寄存器汇总”中的寄存器汇总。

RASR 可使用字或半字访问来访问：位分配见下表。

◇ 高位半字保存区的属性；

◇ 低位半字保存区的大小以及区和子区的使能位。

位	名称	功能
[31:29]	-	保留。
[28]	XN	指令访问禁能位： 0：指令获取使能 1：指令获取禁能。
[27]	-	保留。
[26:24]	AP	访问许可域，参见表“AP 编码”。
[23:22]	-	保留。
[21:19, 17, 16]	TEX，C，B	存储器访问属性，参见表“TEX、C、B 和 S 的编码”。
[18]	S	可共享位，参见表“TEX、C、B 和 S 的编码”。
[15:8]	SRD	子区禁能位。该域的每个位： 0：相应子区被使能 1：相应子区被禁能 更多详情请参见“子区”小节。 不支持 128 字节及更小的子区。对该域写属性时，向 SRD 域写入 0x00。
[7:6]	-	保留。
[5:1]	SIZE	指示 MPU 保护区的大小。最小值为 3 (b00010)，更多详情参见“SIZE 域的值”小节。
[0]	ENABLE	区域使能位。

附录表 1-75 RASR 位分配

有关访问权限的信息，见“MPU 访问权限属性”小节。

SIZE 域的值

SIZE 域定义了由 RNR 指定的 MPU 存储区的大小。定义如下：

（单位为字节的区的大小）=2（SIZE+1）

允许的最小区为 32B，对应的 SIZE 值是 4。下表给出了 SIZE 值示例，并列出了相应的区的大小和 RBAR 中 N 的值。

SIZE 值	区域大小	N 值[1]	注
b00100（4）	32 B	5	最小区域值
b01001（9）	1 kB	10	-
b10011（19）	1 MB	20	-
b11101（29）	1 GB	30	-
b11111（31）	4 GB	b01100	最大区域值

附录表 1-76 SIZE 域值示例

注[1]：RBAR 中，参见表“RBAR 位分配”。

附录1. 4. 5. 6 MPU 访问权限属性

本节描述了 MPU 访问权限属性。RASR 的访问权限位 TEX、C、B、S、AP 和 XN 控制对相应存储区的访问。如果对没有所需权限的存储区域进行访问，则 MPU 产生一个许可故障。

下表列出了 TEX、C、B、S 访问权限位的编码。

TEX	C	B	S	存储器类型	可共享性	其他属性
b000	0	0	x[1]	非常有序	可共享	-
		1	x[1]	设备	可共享	-
	1	0	0	正常	非共享	外部和内部直写，无写分配。
			1		可共享	
		1	0	正常	非共享	外部和内部写回，无写分配。
			1		可共享	
b001	0	0	0	正常	非共享	外部和内部不可缓存。
			1		可共享	
		1	x[1]	保留编码		
	1	0	x[1]	实现定义的属性。		
		1	0	正常	非共享	外部和内部写回，写和读分配。
			1		可共享	
b010	0	0	x[1]	设备	非共享	非共享设备。
		1	x[1]	保留编码		-
	1	x[1]	x[1]	保留编码		-
b1BB	A	A	0	正常	非共享	缓存的存储器[2]，BB=外部策略，AA=内部策略。
			1		可共享	

附录表 1-77 TEX、C、B 和 S 的编码

注[1]：MPU 忽略该位的值。
注[2]：有关 AA 和 BB 位的编码，参见下表“存储器属性编码的缓存策略”。

下表列出了存储器属性编码的缓存策略，TEX 值在 4~7 范围内。

编码，AA 或 BB	相应缓存策略
00	不可缓存
01	写回，写和读分配
10	直写，无写分配
11	写回，无写分配

附录表 1-78 存储器属性编码的缓存策略

下表了定义特权和非特权软件访问权限的 AP 编码。

AP[2:0]	特权许可	非特权许可	描述
000	不可访问	不可访问	所有的访问都会产生一个许可故障
001	RW	不可访问	只可进行特权软件访问
010	RW	R	非特权软件进行写操作时会产生一个许可故障
011	RW	RW	完全访问
100	不可预知	不可预知	保留
101	R	不可访问	只能进行特权软件读操作
110	R	R	只能进行特权或非特权软件读操作
111	R	R	只能进行特权或非特权软件读操作

附录表 1-79 AP 编码

附录1. 4. 5. 7 MPU 不匹配

当一个访问违反 MPU 权限时，处理器产生一个存储器管理故障，见“异常和中断”小节。MMFSR 指示故障的原因。更多信息见表“MMFSR 位分配”

附录1. 4. 5. 8 更新一个 MPU 区

如要更新一个 MPU 区的属性，需更新 RNR、RBAR 和 RASR 寄存器。可以对每个寄存器单独编程，也可使用一个多字写入，对全部这些寄存器编程。可使用 RBAR 和 RASR 别名利用一条 STM 指令同时对 4 个区编程。

使用单独的字更新一个 MPU 区

配置一个区的简单代码：

```
; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address
LDR R0,= MPU_RNR          ; 0xE000ED98, MPU region number register
STR   R1, [R0, #0x0]      ; Region Number
STR   R4, [R0, #0x4]      ; Region Base Address
STRH   R2, [R0, #0x8]     ; Region Size and Enable
STRH   R3, [R0, #0xA]     ; Region Attribute
```

如果已使能了要更改的区，则在写入新的区设置到 MPU 前要先禁用该区。例如：

```
; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address
LDR   R0,=MPU_RNR         ; 0xE000ED98, MPU region number register
STR   R1, [R0, #0x0]      ; Region Number
BIC   R2, R2, #1          ; Disable
STRH   R2, [R0, #0x8]     ; Region Size and Enable
STR   R4, [R0, #0x4]      ; Region Base Address
```

```
STRH   R3, [R0, #0xA]          ; Region Attribute
ORR    R2, #1                  ; Enable
STRH   R2, [R0, #0x8]          ; Region Size and Enable
```

以下情况下，软件必须使用存储器屏障指令：

◇ 如果可能有未完成的存储器传输（如缓冲写入），可能受到 MPU 设置更改的影响，则要在 MPU 设置前使用；

◇ 如果存储器传输必须使用新的 MPU 设置，则要在 MPU 设置后使用。

◇ 但是，如果 MPU 设置进程通过进入一个异常处理程序启动，或后边跟一个异常返回，则不需要存储器屏障指令，因为异常进入和异常返回机制会引起存储器屏障行为。

MPU 设置期间，软件不需要任何存储器屏障指令，因为它通过 PPB 访问 MPU，而 PPB 为一个非常有序型存储区。

例如，如果需要在编程序列后，让全部存储器访问行为立即生效，可以使用一条 DSB 指令和一条 ISB 指令。改变 MPU 设置后，例如上下文切换后，需要使用一个 DSB。如果利用跳转或调用进入一个或多个 MPU 区的编程代码，则需要使用 ISB。如果利用异常返回或是获取异常进入一个编程序列，则不需要 ISB。

使用多字写入更新一个 MPU 区

根据信息的分割方式，可使用多字写入直接编程。考虑以下再编程（reprogramming）：

```
; R1 = region number
; R2 = address
; R3 = size, attributes   in one
LDR   R0, =MPU_RNR          ; 0xE000ED98, MPU region number register
STR   R1, [R0, #0x0]        ; Region Number
STR   R2, [R0, #0x4]        ; Region Base Address
STR   R3, [R0, #0x8]        ; Region Attribute, Size and Enable
```

使用一条 ST M 指令来优化它：

```
; R1 = region number
; R2 = address
; R3 = size, attributes   in one
LDR   R0, =MPU_RNR          ; 0xE000ED98, MPU region number register
STM   R0, {R1-R3}           ; Region Number, address, attribute, size and enable
```

对于预封装的信息，可用双字实现此操作。这就是说，RBAR 包含所需的区号，并将 VALID 位设为 1，当数据是静态封装时，例如在一个启动加载程序中时，使用此方法：

```
; R1 = address and region number in one
; R2 = size and attributes   in one
LDR   R0, =MPU_RBAR         ; 0xE000ED9C, MPU Region Base register
STR   R1, [R0, #0x0]        ; Region base address and
                            ; region number combined with VALID (bit 4) set to 1
STR   R2, [R0, #0x4]        ; Region Attribute, Size and Enable
```

使用一条 STM 指令来优化它：

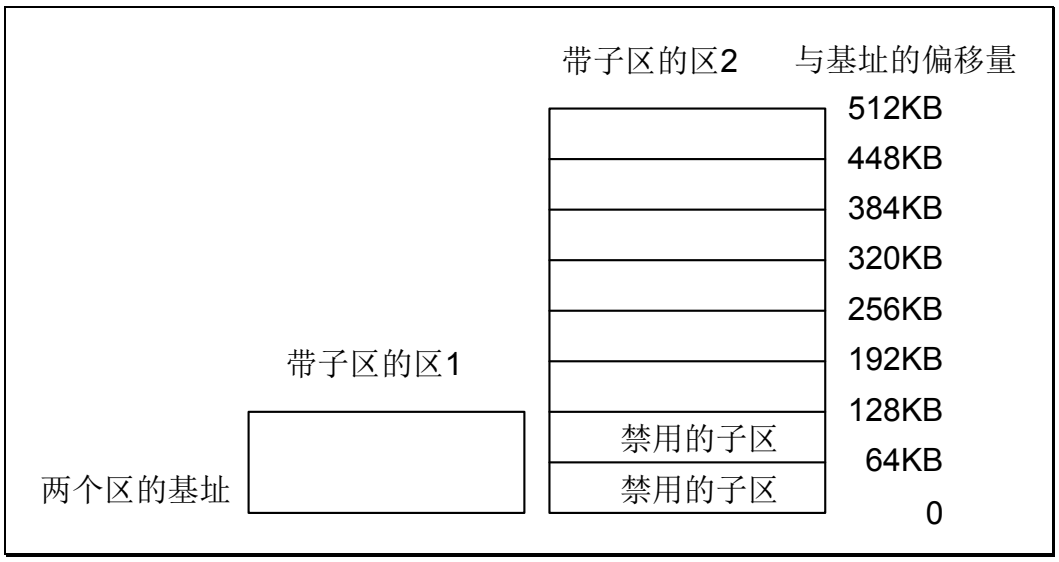
```
; R1 = address and region number in one
; R2 = size and attributes   in one
LDR   R0,= MPU_RBAR              ; 0xE000ED9C, MPU Region Base register
STM   R0, {R1-R2}                ; Region base address, region number and VALID bit,
                                 ; and Region Attribute, Size and Enable
```

子区

256 或更多字节的区可划分为 8 个相同大小的子区。可设置 RASR 中 SRD 域内相应的位来禁用某个子区，见表“RASR 位分配” 。SRD 的最低位控制第一个子区，最高位控制最后一个子区。禁用某个子区就意味着，另一个与禁用区重叠的区取代了它实现了匹配。如果没有其他使能的区与禁用的子区重叠，MPU 发出一个故障。

32、64 和 128 个字节的区不支持子区，使用这些大小的区时，必须将 SRD 域设置为 0x00， 否则 MPU 行为不可预知。

SRD 使用示例：两个具有相同基址的区重叠。区 1 为 128 kB，区 2 为 512 kB。为了保证区 1 的属性应用到第一个 128 kB 区，将区 2 的 SRD 域设置为 b00000011，禁用前两个子区， 如下图所示。



附录图 1-16 SRD 使用示例

附录1. 4. 5. 9 MPU 设计提示和建议

为了避免不期望的行为，在更新某个中断处理程序可能访问的区属性前，禁用中断。确保软件使用大小正确的对齐访问来访问 MPU 寄存器：

◇ 除 RASR 外，必须使用对齐字访问；

◇ 对于 RASR，可使用字节、对齐半字或字访问。

处理器不支持对 MPU 寄存器的非对齐访问。

当设置 MPU 时，如果 MPU 已经编程，则禁用未使用的区，以防止之前任何区的设置影响到新的 MPU 设置。

微控制器的 MPU 配置

通常，一个微控制器系统只有单个处理器且没有缓存。在该类系统中，MPU 编程如下：

存储区	TEX	C	B	S	存储器类型和属性
Flash 存储器	b000	1	0	0	正常存储器，非共享，直写
内部 SRAM	b000	1	0	1	正常存储器，可共享，直写
外部 SRAM	b000	1	1	1	正常存储器，可共享，写回，写分配
外设	b000	0	1	1	设备存储器，可共享

附录表 1-80 微控制器的存储器属性

在大多数微控制器执行中，可共享性和缓存策略的属性都不影响系统行为。然而，如果 MPU 区使用了这些设置，可使应用代码具有更好的可移植性。所列出的值是针对典型情况。在特殊系统中，例如多处理器设计或具有单独 DMA 引擎的设计中，可共享性属性可能会很重要。在这些情况下，请参考存储设备制造商的建议。

附录1.5 ARM Cortex-M3 用户指南：术语表

中止——向处理器表示一个存储器访问相关的值无效的一种机制。中止可能由外部或内部存储器系统引起，作为试图访问无效指令或数据存储器的结果。

对齐——如果保存在一个地址的数据项能够被定义数据大小的字节数整除，则此数据项被称为是对齐的。对齐的字和半字地址分别可被 4 和 2 整除。因此，术语字对齐和半字对齐规定，其地址能够分别被 4 和 2 整除。

分组寄存器——一个寄存器有多个物理副本，处理器的状态决定了使用哪个副本。堆栈指针 SP（R13）就是一个分组寄存器。

基址寄存器——在指令描述中，加载或存储指令指定一个寄存器，用于保持指令地址计算的基值。根据指令及其寻址模式，可在基址寄存器值上加或减一个偏移量，就形成了要发送到存储器的地址。亦见索引寄存器。

大端（big-endian）——字节定序方案，其中递减有效字节保存在存储器的递增地址中。亦见字节不变、字节序、小端。

大端存储器——具有以下特点的存储器：

- ◇ 字对齐地址上的字节或半字为此地址的字中的最高有效字节或半字。
- ◇ 半字对齐地址的字节为此地址的半字中的最高有效字节。

断点——断点是调试器提供的一个机制，用来识别程序执行被停止的指令。断点由程序员插入，以便能够检查程序执行中的寄存器内容，存储单元和某个固定点的变量值，从而检查程序工作是否正常。成功完成程序测试后，断点被移除。

字节不变（byte-invariant）——在字节不变系统中，当在小端和大端操作间切换时，存储器每个字节的地址保持不变。当把大于 1 个字节的数据项从存储器加载或存储到存储器时，根据存储器访问的字节序，将构成该数据项的字节安排为正确的顺序。

ARM 的字节不变实现还支持非对齐的半字和字存储器访问。多字访问按字对齐。

缓存——片上或片外快速访问存储单元模块，位于处理器和主存储器之间，用于存储和检索常用指令、数据或同时包括指令和数据的副本。这样可大大增加存储器访问的平均速度，改善处理器性能。

条件域——指令中一个 4 位的域，它指定了指令可以执行的条件。

上下文——对于一个多任务操作系统，各进程运行的环境。在 ARM 处理器中，它仅限于表示可访问的存储器物理地址范围，以及相关的存储器访问权限。

协处理器——协助主处理器的一种处理器。Cortex-M3 不支持任何协处理器。

调试器——一种包含某个程序的调试系统，用于检测、定位和纠正软件故障，还包括支持软件调试的定制硬件。

直接存储器访问（DMA）——无需处理器对相关数据执行任何访问，而直接访问主存的操作。

双字——一个 64 位数据项。除非另外说明，其内容被视为一个无符号整数。

双字对齐——存储器地址可被 8 整除的数据项。

字节序（endianness）——字节定序。该机制用来确定存储在存储器中的一个数据字的连续字节的顺序。是系统的存储器映射的一个方面。亦见小端和大端。

异常——中断程序执行的一个事件。当发生异常时，处理器挂起正常的程序流，并开始从相应异常向量所指示的地址执行。指示的地址存有异常处理程序的第一条指令。

一个异常可以是一个中断请求、一个故障，或软件产生的系统异常。故障包括试图进行非法的存储器访问、试图在非法的处理器状态下执行某条指令，以及试图执行一条未定义的指令。

异常服务程序——亦见中断处理程序。

异常向量——亦见中断向量。

平板地址映射（flat address mapping）——一种存储器组织体系，其中存储器空间中每个物理地址都与对应的虚拟地址相同。

半字——一个 16 位的数据项。

非法指令——结构上未定义的指令。

实现定义的（Implementation-defined）——结构上未定义，但在单独实现中定义并证明的。

实现特定的（Implementation-specific）—— 结构上未定义，且不必在单独实现中证明。

当有多个实现选项可供选择，且所选选项不影响软件兼容性时，使用该“实现特定的”。

索引寄存器——在某些加载和存储指令描述中，此寄存器的值被作为一个偏移量，将其加入基址寄存器值或从基址寄存器中减去，以形成发送到存储器的地址。某些寻址模式可以选择在加法或减法前，使能被移位的索引寄存器值。亦见基址寄存器。

指令周期数——指令占用流水线执行阶段所消耗的周期数。

中断处理程序——发生中断时，处理器的控制权传递到这个程序。

中断向量——存储器低地址空间或存储器高地址空间（如果配置了高位向量）内一些固定地址，存有相应中断处理程序的第一条指令。

小端（little-endian）——字节定序机制，其中一个数据字中的递增有效的字节保存在存储器的递增地址中。亦见大端、字节不变、字节序。

小端存储器——具有以下特点的存储器：

- ◇ 字对齐地址中的字节或半字为此地址字中的最低有效字节或半字。
- ◇ 半字对齐地址中的字节为此地址半字中的最低有效字节。

亦见大端存储器。

加载/存储架构——一种处理器架构，数据处理操作只针对寄存器的内容，而不直接对存储器内容。

存储器保护单元（MPU）——控制对存储器模块访问权限的硬件。MPU 不进行任何地址转换。

预取——在流水线处理器中，当前面的指令执行完成前，从存储器获取指令以填充流水线的过程。预取一条指令并不意味着该指令一定会执行。

读——读定义为具有一种加载语义的存储器操作。读包括 Thumb 指令 LDM、LDR、LDRSH、LDRH、LDRSB、LDRB 和 POP。

区——存储器空间的分区。

保留——在控制寄存器或指令格式中，如果某个域被实现定义了，或者在该域不为零时会产生不可预知的结果，则该域是保留域。这些域保留用于将来的结构扩展，或是由实现特定使用。所有未被实现使用的保留位都必须写为 0，且读取为 0。

应为 1（Should Be One，SBO）——软件写 1 或位域全写 1。写入 0 会产生不可预知的结果。

应为 0（Should Be Zero，SBZ）——软件写 0 或位域全写 0。写入 1 会产生不可预知的结果。

应为 0 或保留（Should Be Zero or Preserved，SBZP）——软件写 0 或位域全写 0，或通过将前面读出的相同处理器上相同域的值写回来保留。

线程安全（Thread-safe）——在多任务环境中，线程安全功能在访问共享资源时使用保护机制，以确保操作正确，而不会有共享访问冲突的风险。

Thumb 指令——指定处理器完成一个操作的一个或两个半字。Thumb 指令必须按半字对齐。

非对齐——存储在不能被定义的数据大小的字节数整除的地址的数据项被称为“非对齐”。例如，存储在不能被 4 整除的地址的一个字。

不可预知（UNP）——不能确知的行为。不可预知的行为不代表安全漏洞。不可预知的行为不得让处理器或系统的任何部分停止或挂起。

热复位——又称内核复位。它对处理器的大部分元件初始化，调试控制器和调试逻辑除外。

这种类型的复位在使用处理器的调试功能时非常有用。

WA——见写分配。

WB——见写回。

字——一个 32 位的数据项。

写——写定义为具有加载语义的存储器操作。写包括 Thumb 指令 STM、STR、STRH、STRB 和 PUSH。

写分配（write-allocate）——在一个写分配缓存中，存储数据的缓存未中会使一个缓存线被分配到缓存中。

写回（write-back）——在写回缓存中，仅在缓存未命中后行替换时，数据被强制清出缓存，才将数据写入主存储器。否则，处理器的写入只更新缓存。这又称回拷（copyback）。

写缓冲——位于数据缓存和主存之间的高速存储器模块，用作一个 FIFO 缓冲区，目的是优化到存储器的存储。

直写（write-through）——在直写缓存中，数据在缓存被更新的同时写入主存储器。

第38章 修订历史

版本	修订日期	修订内容
V1.0	2020-03-30	初版发布
V1.1	2020-08-03	1.该系列芯片配置字中无 BOR 使能位，删除 PMU/RMU 章节中相关描述。 2.SPI 补充主/机全双工收发时序图及对应文字描述，DMA 全双工收发时序图及对应文字描述；将摩托罗拉 SPI 协议改为标准 SPI 通信协议；修改“主机模式故障”章节中描述；调换部分章节顺序。 3.I2C 补充主/从发送/接收时序图及对应文字描述。 4.UART 章节中更改“防抖动输出时序图”及相关描述。 5.ADC 章节中对不同产品系列内部通道 ADC_IN16~ADC_IN18 与模块输出的连接关系补充明确说明。 6.存储器系统控制章节中添加“IAP 自编程硬件固化模块”小节；修改 Flash 快速编程长度寄存器中错误描述。 7.PIS 和 DMA 章节中涉及到 LPUART/LP16T 描述的寄存器添加“注：ES32F36xx 不支持 LP16T、LPUART”。 8.Flash 信息区章节中删除“用户产品序列码”相关描述。 9.AD16C4T 中“通信事件（COM）”更改为“换相事件（COM）”；增加章节“6 步 PWM 生成”。 10.附录添加 Cortex-M3 内核规格说明。
V1.2	2020-12-07	1.删除 EBI 模块中有关 LCD 的章节描述。 2.删除 UART 模块的交换 TX/RX pin 配置功能。 3.将 RTC 模块的"侵入检测 1/侵入检测 2"修改为"侵入检测 0/侵入检测 1"。 4.删除 CALC 模块除法运算章节及相关寄存器。 5.修改系统总线和存储器章节中 SRAM 容量大小为 96K Bytes 并修改相应的地址范围。 6.修改芯片配置指引中部分中断向量的优先级。 7.修改 IWDT 载入 IWDT_LOAD 寄存器的默认值为 0x8000（约 1s）。 8.修改 UART_SCARD 寄存器 PSC 位。 9.修改 ACMP_INPUTSEL 寄存器 NSEL 位。
V1.3	2021-05-19	1.删除文中所有关于 Boot Rom 功能的相关描述。 2.修改芯片配置指引章节中 ADC 和 ACMP 模块的寄存器名称。 3.修改寄存器 SPI_CON1.SSOUT 中的描述。 4.更新 CMU 模块中 HOSC 电路图与 Datasheet 保持一致并补充 LOSC 电路图。 5.表 5-4 中部分访问限制有误，根据设计更正，并修改部分注释。 6.修改附录表 1-50 中分组优先级的描述。

版本	修订日期	修订内容
		7.删除 I2C 模块的数字滤波功能。 8.修改第 37.1.1 章中 UID2 地址偏移值为 $03EC_H$。 9. USB 章节中出现重复的寄存器，删除“29.5.2.16 索引端点 0 的控制和状态高字节寄存器”