数据手册

APT32F172

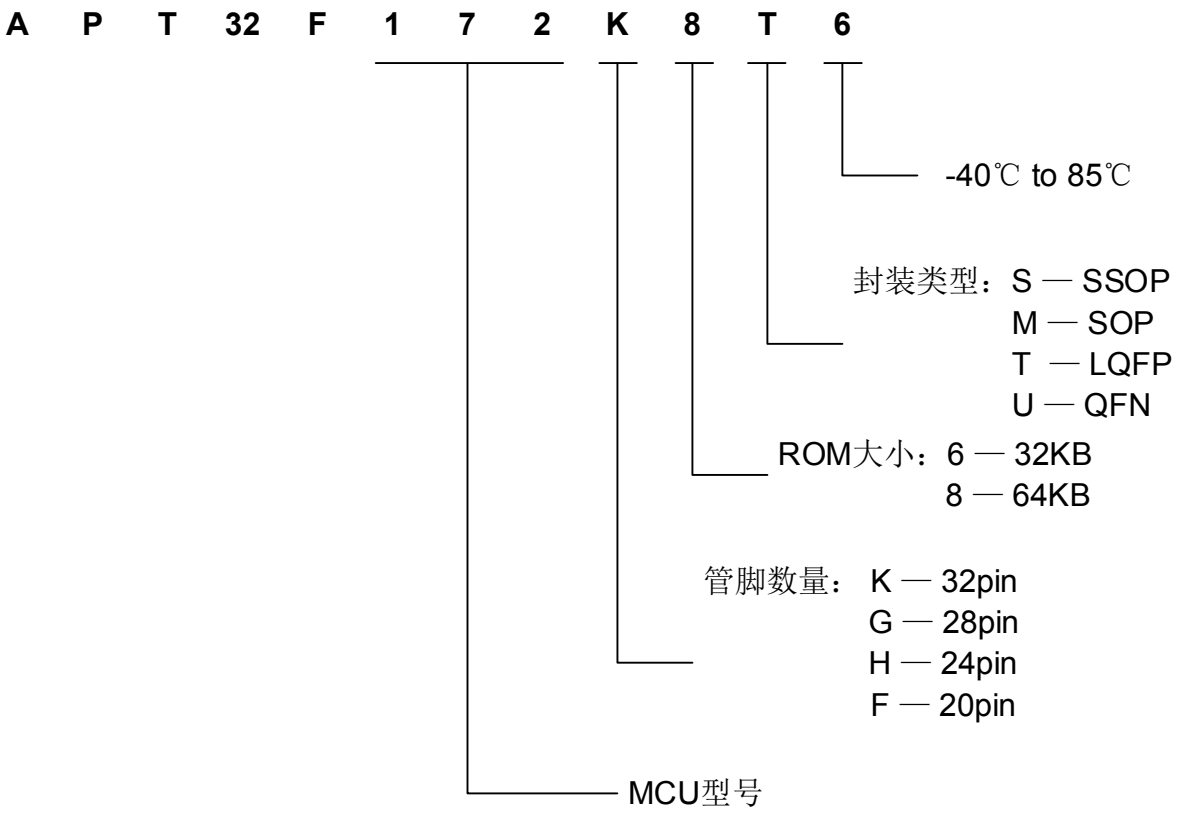
通用型 32 位微处理控制器

Version 1.5

Dec 2018

产品订购信息

型号	FLASH 大小	SRAM 大小	封装
APT32F172K8T6	64KB	6KB	LQFP32
APT32F172K8U6	64KB	6KB	QFN32
APT32F172G8M6	64KB	6KB	SOP28
APT32F172H8S6	64KB	6KB	SSOP24



历史版本说明

版本	修改日期	修改概要
V1.0	2018-02-08	初版
V1.1	2018-04-25	- TC1增加寄存器使用说明，PRDR/PULR不能设置为0 - 调整28PIN封装的管脚图，增加PC0.3，删除PD0.0 - 调整24PIN封装的管脚图，增加PC0.3/PA0.10，删除PA0.1/PA0.2 - 运放增加OPAxN_EXEN负向输入控制位 - 删除比较器中的BOOST/SPEED控制位，无特殊情况不建议客户修改此两位 - 增加EPWM章节中载波输出的相关描述说明 - 修正各章节一些描述错误
V1.2	2018-05-16	- 增加32QFN封装 - 修正EPWM里一些描述 - 电气特性章节增加上电和掉电特性
V1.3	2018-05-25	- 修正32QFN尺寸图，5x5(0.5)改为4x4(0.4)
V1.4	2018-08-21	- 增加FLASH烧录时电源去耦电容的建议 - 修正USART中不正确的波特率配置示例 - 增加32KHz晶振使用条件说明 - 修整格式，修正前后描述不一致的错误
V1.5	2018-12-7	- 修正内部参考电压精度精度范围为2%对应的电压值 - 修正SYSCON PCER/PCSR寄存器中GPIO相关的描述 - 修正QFN32封装描述中32脚的位置标识

1 概述

1.1 文档用途

本文档是APT32F172产品的一个完整，详细的规格书。

1.2 APT32F172介绍

APT32F172 基于 C-Sky Microsystems Co., Ltd 开发的 32 位单片机核的高性能低成本单片机。APT32F172 单片机面向的应用为变频控制，马达驱动，功率检测等应用。

- C-Sky 32 位 CPU 核
- 片载 60K(32Kbytes 可选)程序闪存，4Kbytes 数据闪存(大小可配置)
- 内含 6Kbytes SRAM，可用于堆栈，数据存储，代码存储
- 工作温度： – 40 to 85°C
- 工作电压范围： 2.4 to 5.5V
- 中断控制器：支持动态配置的可嵌套中断（NVIC）
- 用户友好的时钟和功耗控制器（SYSCON）
- 1 x 独立看门狗定时器（IWDT）
- 3 x 16 位同步定时/计数器，支持三路互补 PWM 输出（TC0）
- 1 × 32 位通用定时器/计数器，支持 PWM 功能（TC1）
- 1 × 16 位 2 路同步精简定时器/计数器，支持捕捉功能（TC2）、
- 1 x 16 位时钟定时器（TC3）
- 1 x 16 位增强型独立 PWM 模块，支持三路带死区控制的互补输出，支持与比较器联动工作（EPWM）
- 1 × I2C 和 1 × SPI
- 2 x UART，其中 1 路为增强型 USART。
- 多达 18 路的 12 位 ADC
- 多达 20 路的触摸按键控制器
- 支持 8x8 点阵(1/4 占空比)的自动扫描 LED 驱动
- 多达 5 个独立模拟比较器，2 个运算放大器
- 8 个大电流驱动的管脚(每个管脚最大 120mA)
- 支持 RUN，SLEEP，和 DEEP-SLEEP 模式

1.3 主要特性

1.3.1 CPU

- 32-bit RISC CPU核，指令长度16位
- 高效的2级执行流水线
- 单周期32位x32位的硬件整形乘法阵列
- SWD (Serial Wire Debug)调试接口

1.3.2 存储

- 多达60Kbytes的内部程序闪存，支持ISP保护，保护区域的大小可配置
- 多达4Kbytes的数据闪存，大小可通过User Option配置
- 闪存的User Option配置支持修改复位源，允许或者禁止看门狗，修改数据闪存大小以及使能程序代码保护功能
- 专用闪存烧录工具，支持快速的量产烧录
- 多达6Kbytes的内部SRAM
- 只支持小端(little-endian)存储方式

1.3.3 可嵌套中断控制器

- 多达32个中断源
- 32个可编程优先级，每个中断都有独立的优先级
- 每个中断都有独立的使能或者禁止控制
- 每个中断源都有固定的向量地址
- 支持陷阱功能
- 支持软件复位
- 可单独配置唤醒事件和中断的使能/禁止

1.3.4 系统控制器(SYSCON)

- 外部晶振 32.768K 到 24MHz (EMCLK: External Main Clock，外部主时钟)
- 内部晶振 20MHz或40MHz (1%偏差，典型值) (IMCLK: Internal Main Clock，内部主时钟)
- 内部副晶振 500KHz 或 3MHz (ISCLK: Internal Sub Clock，内部副时钟)
- 支持低功耗模式 (SLEEP/DEEP-SLEEP)
- 低功耗模式下支持可编程的功耗优化
- 可编程的时钟分频器
- 外部晶振失效监测
- 复位源检测和管理 (RSTID)

1.3.5 IWDT: 独立看门狗定时器

- 复位时间可配置

- 可配置定时提醒中断
- 使用内部副晶振的可编程18位递减计数器

1.3.6 TC0: 16位加强型定时器/计数器

- 内部总共三个16位多模式计数器，可分别独立工作，或者同步工作。
- 每个计数器支持3个16位输出比较值寄存器
- 可以工作在捕捉，匹配和溢出，输出翻转和PWM输出模式
- 支持三路PWM互补输出模式
- 多种可选择的时钟源

1.3.7 TC1：32位通用定制器/计数器

- 32位定时器/计数器
- 定时器的计数位数可配置成32以内的任意数
- 可以工作在捕捉，匹配和溢出，输出翻转和PWM输出模式
- 匹配和溢出中断
- 可选择内部或者外部时钟源

1.3.8 TC2：16位精简定制器/计数器

- 可配置16位自增定时器/计数器
- 两个捕捉通道
- 两种匹配输出模式

1.3.9 TC3：16位时钟定时器

- 16位定时器
- 计时和定时功能
- 可选的时钟源：外部晶振（支持32.768KHz），内部副振
- 蜂鸣器频率输出

1.3.10 EPWM：16位增强型脉宽调制模块

- 16位计数器
- 3路同步可配置PWM输出，支持双路独立输出模式，死区可调的互补输出模式和双路逐次触发输出模式
- 支持3路共用同一个计数器工作，或者3路计数器分开独立工作
- 支持比较器联动
- 支持紧急停车功能

1.3.11 USART: 同步异步收发器

- 1个通道

- 支持5、6、7和8位的数据长度
- 可编程的波特率
- 校验位，帧检测和缓存溢出错误报告
- 支持Loop-back模式
- 支持同步全双工模式
- 支持LIN总线协议：LIN1.2或者LIN2.0
- 支持智能卡协议：ISO7816-3兼容

1.3.12 UART: 通用异步收发器

- 1个通道
- 8位数据长度，支持校验位(奇偶校验，0/1校验)
- 可编程的波特率

1.3.13 I2C

- 1个通道
- 支持多主机I2C总线
- 兼容串行8位数据传输和双向数据传输
- 标准模式100Kbit/s，高速模式可达400Kbit/s

1.3.14 SPI

- 1个通道
- 可编程的数据帧长度：4到16位
- 支持主机和从机模式
- 单独的8x16位收发FIFO

1.3.15 ADC: 模数转换器

- 多达18个模拟输入通道供选择，参考电压支持选择内部或者外部，并且自带内部固定的电压参考源
- 12位模式支持最快500KSPS转换速度，10位模式支持最快1MSPS
- 支持连续转换模式和硬件比较转换结果
- 支持多序列转换模式， 最高可达16个转换序列， 可灵活配置转换通道，转换顺序，转换次数
- 支持连续采样或者单次采样，可灵活配置的采样优先级
- 启动转换支持外部管脚触发，定时器触发，EPWM触发或者比较器触发

1.3.16 TOUCH KEY：触摸按键

- 支持20个触摸按键扫描通道
- 每个通道都支持可编程的灵敏度调节
- 支持同一个管脚上的触摸扫描和LED驱动同时工作

- 可配置扫描时钟频率，用于优化功耗
- 多种扫描触发模式
- 内嵌独立的硬件算法模块，支持硬件自动按键检测和系统唤醒

1.3.17 LED驱动

- 支持8 x 8 点阵(1/8占空比) LED驱动
- 支持硬件自动扫描控制，占空比和扫描时间间隔可通过寄存器配置
- 可配置的COM通道数和亮度控制
- 段码驱动支持恒流模式
- 多达8个大电流驱动I/O口，可以直接驱动LED(每个120mA)
- 8段显示驱动的通道可以和触摸按键功能在不同时间段内共用同一个管脚

1.3.18 COMP：模拟比较器

- 支持5个独立的模拟比较器
- 可配置的比较器输出迟滞和数字去抖滤波
- 每个比较器可以独立选择内部248个参考电压
- 支持比较器输出通过特定事件窗口捕获功能
- 输入可以和AD以及运算放大器共用管脚

1.3.19 OPAMP：运算放大器

- 支持2个独立的运算放大器
- 可配置的内部增益控制
- 输出可以作为ADC的采样输入

1.3.20 通用IO (GPIO)

- 32管脚：30 个GPIO
- 28管脚：26 个GPIO
- 24管脚：22 个GPIO
- 推挽输出和开漏输出可配置
- 上下拉电阻可配置
- 支持输出状态监测
- 管脚复用功能简单易用；所有管脚都支持外部中断功能

1.3.21 两个低功耗模式

- SLEEP: 关闭选择的系统时钟和CPU时钟
- DEEP-SLEEP: 关闭所有系统时钟和CPU时钟

- 可配置的DEEP-SLEEP唤醒源：外部中断，iWDT中断，LVD中断或者触摸按键中断

1.3.22 POR: 上电复位

1.3.23 LVD: 低电压检测

- 可配置成低电压复位功能，可选4个电压值 (2.15V/2.75V/3.35V/3.75V).
- 可配置成低电压产生中断，可选4个电压值 (2.55V/3.00V/3.90V/4.10V).

1.3.24 工作电压范围

- 2.4V to 5.5V

1.3.25 工作频率范围

- 外部主晶振 24 MHz
- 内部主晶振 20 MHz，或者40MHz
- 内部副晶振 500KHz 或 3MHz

1.3.26 工作温度范围

- – 40 to 85°C

1.3.27 封装

- 32-LQFP
- 32-QFN
- 28-SOP
- 24-SSOP

1.4 模块框图

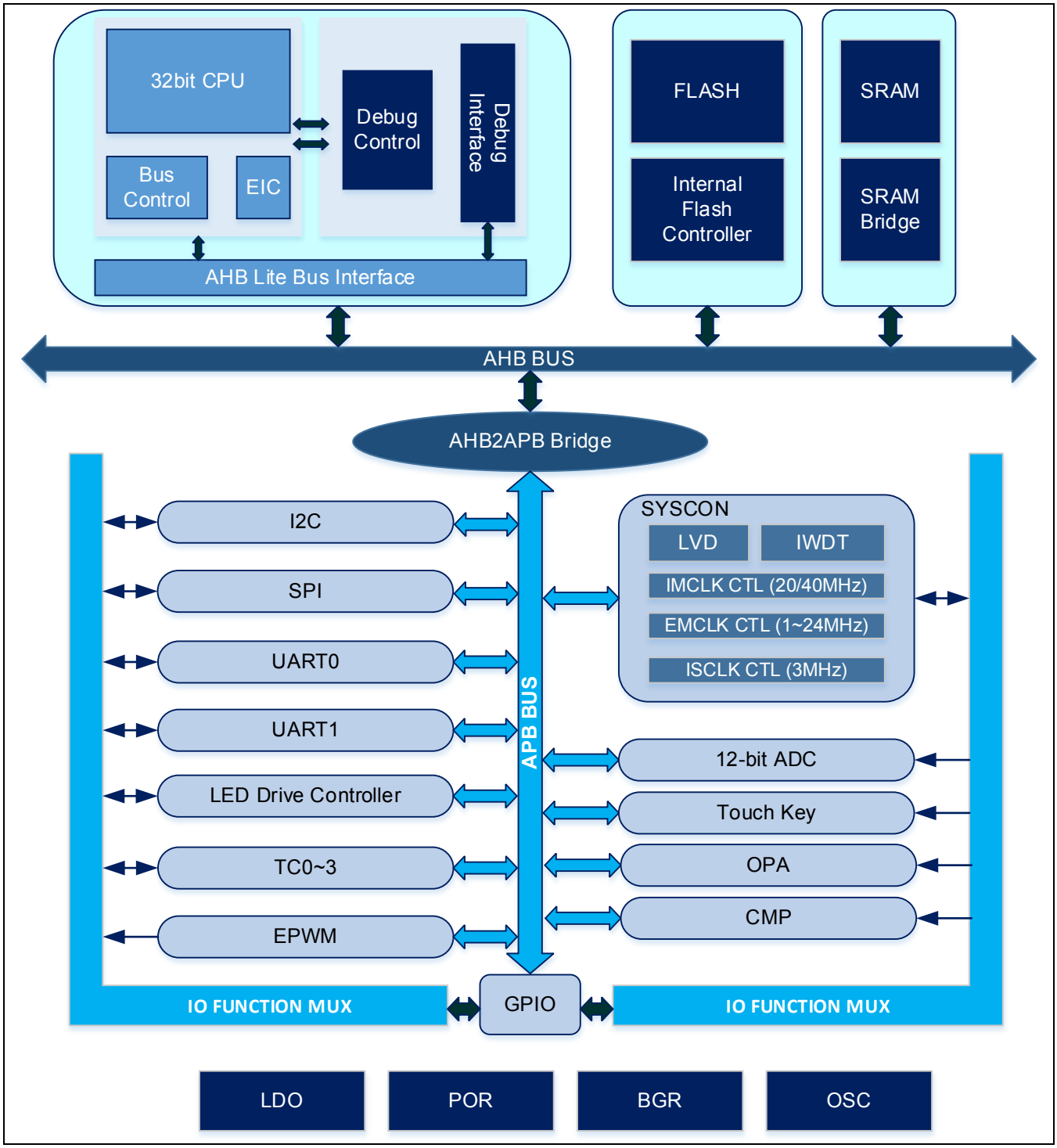


Figure 1-1 APT32F172模块框图

2 管脚配置

2.1 概要

本章节描述APT32F172产品的管脚功能信息。

包含：

- 管脚映射图
- 管脚分配表
- 管脚复用
- 管脚描述
- Pad电路类型

2.2 管脚定义图

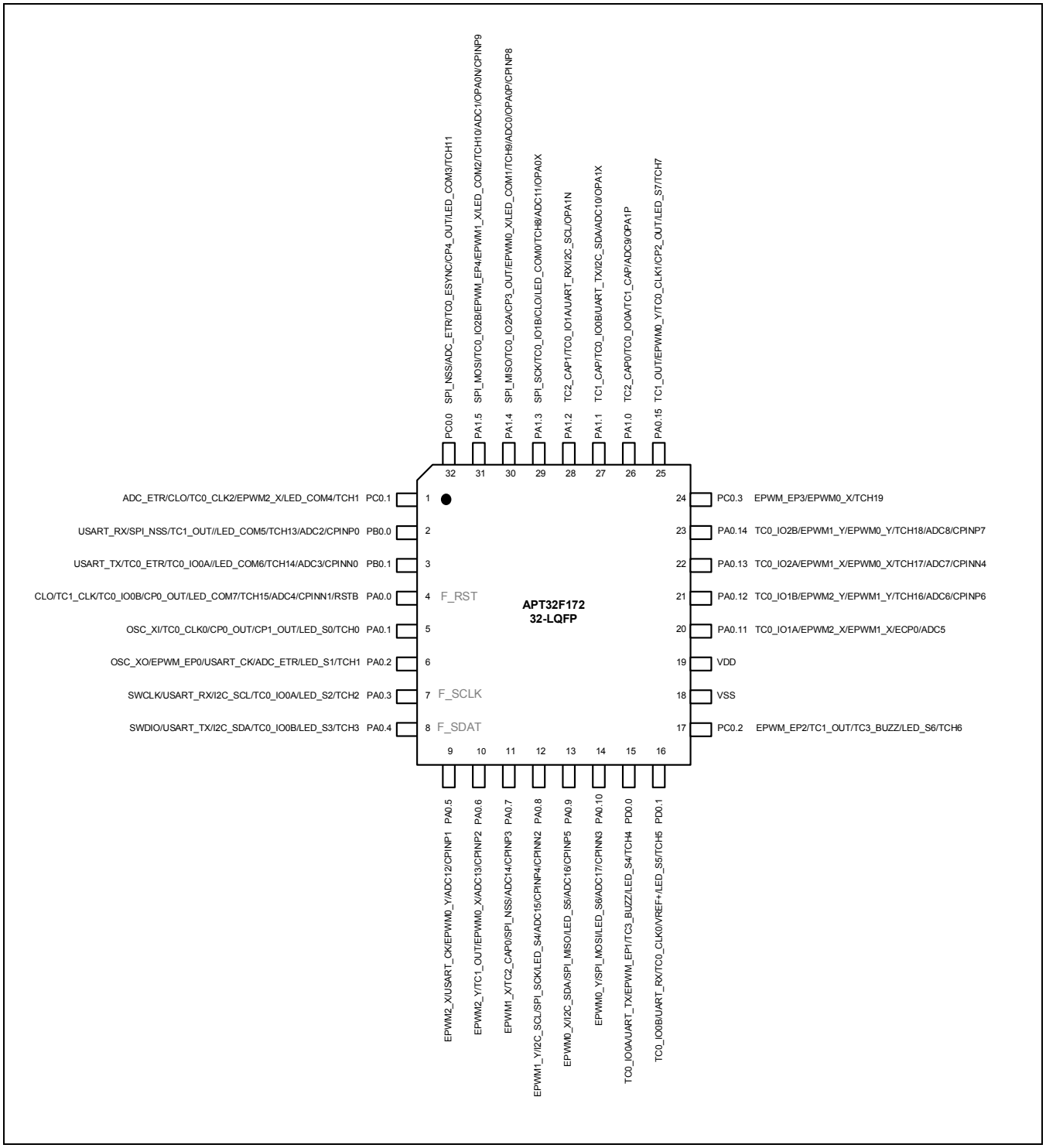


Figure 2-1 管脚定义图(32PIN)

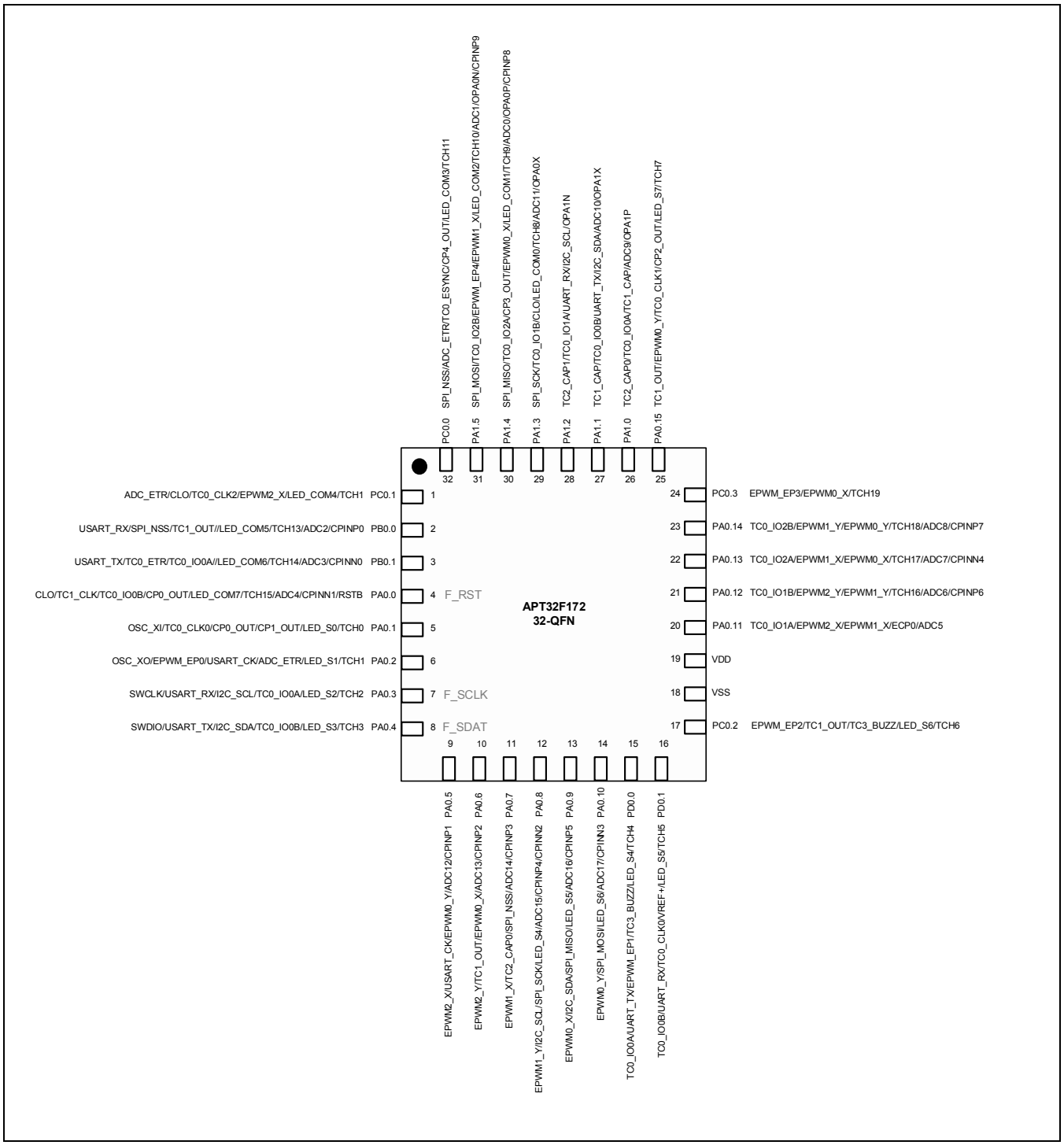


Figure 2-2 管脚定义图(32PIN)

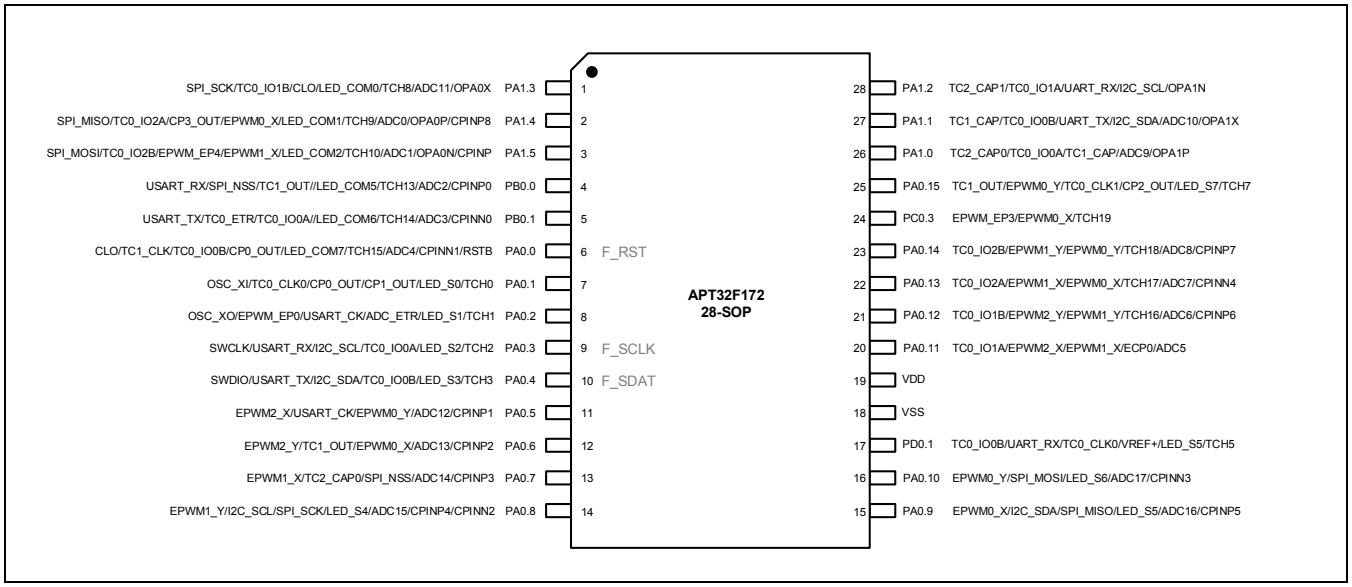


Figure 2-3 管脚定义图(28PIN)

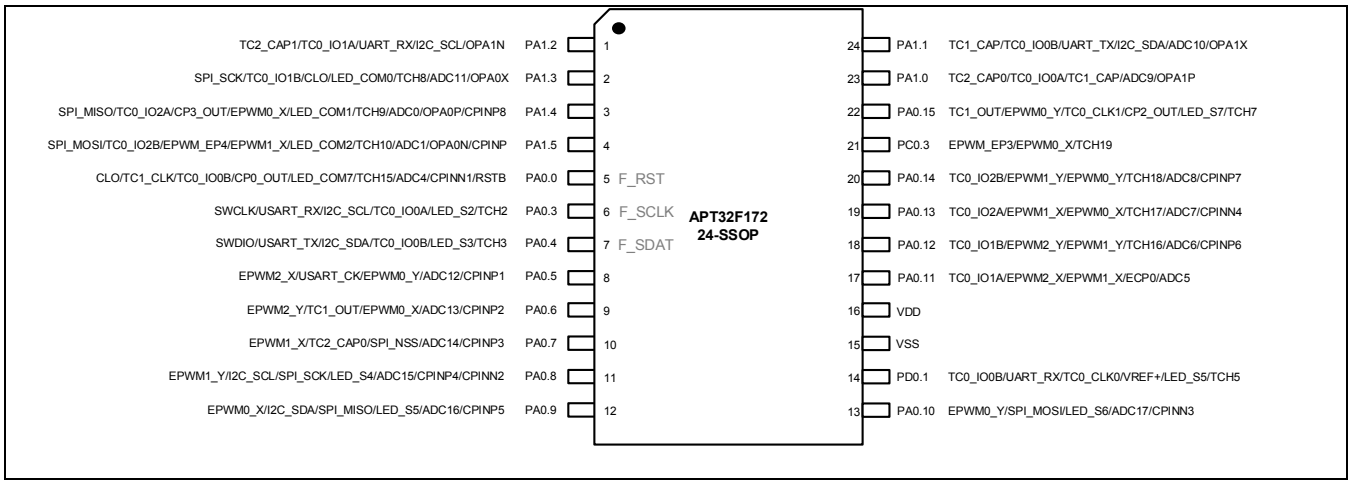


Figure 2-4 管脚定义图(24PIN)

2.3 管脚功能分配

Table 2-1 描述了管脚功能的详细分配。

- *UP*: 上拉使能; *DN*: 下拉使能
- *IO*: 双向; *I*: 输入; *O*: 输出; *P*: 电源; *G*: 地; *Z*: 高阻

Table 2-1 管脚功能分配，依照管脚号排序

Pin Number			Pin Name									Default	PU/PD	Reset Status
32PIN	28PIN	24PIN	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	EXI			
1	-	-	PC0.1	ADC_ETR	CLO	TC0_CLK2	EPWM2_X	**LED_COM4**	TCH12	-	EXI1	IO	-	Z
2	4	-	PB0.0	USART_RX	SPI_NSS	TC1_OUT		**LED_COM5**	TCH13	ADC2/CPINP0	EXI0	IO	-	Z
3	5	-	PB0.1	USART_TX	TC0_ETR	TC0_IO0A		**LED_COM6**	TCH14	ADC3/CPINN0	EXI1	IO	-	Z
4	6	5	PA0.0	CLO	TC1_CLK	TC0_IO0B	CP0_OUT	**LED_COM7**	TCH15	ADC4/CPINN1	EXI0	IO	-	Z
5	7	-	PA0.1	OSC_XI	TC0_CLK0	CP0_OUT	CP1_OUT	LED_S0	TCH0	-	EXI1	IO	-	Z
6	8	-	PA0.2	OSC_XO	EPWM_EP0	USART_CK	ADC_ETR	LED_S1	TCH1	-	EXI2	IO	-	Z
7	9	6	PA0.3	**SWCLK**	USART_RX	I2C_SCL	TC0_IO0A	LED_S2	TCH2	-	EXI3	SWCLK	UP	I
8	10	7	PA0.4	**SWDIO**	USART_TX	I2C_SDA	TC0_IO0B	LED_S3	TCH3	-	EXI4	SWDIO	UP	I
9	11	8	PA0.5	EPWM2_X	USART_CK	EPWM0_Y	-	-	-	ADC12/CPINP1	EXI5	IO	-	Z
10	12	9	PA0.6	EPWM2_Y	TC1_OUT	EPWM0_X	-	-	-	ADC13/CPINP2	EXI6	IO	-	Z
11	13	10	PA0.7	EPWM1_X	TC2_CAP0	SPI_NSS	-	-	-	ADC14/CPINP3	EXI7	IO	-	Z
12	14	11	PA0.8	EPWM1_Y	I2C_SCL	SPI_SCK	LED_S4	-	-	ADC15/CPINP4/CPINN2	EXI8	IO	-	Z
13	15	12	PA0.9	EPWM0_X	I2C_SDA	SPI_MISO	LED_S5	-	-	ADC16/CPINP5	EXI9	IO	-	Z
14	16	13	PA0.10	EPWM0_Y	-	SPI_MOSI	LED_S6	-	-	ADC17/CPINN3	EXI10	IO	-	Z
15	-	-	PD0.0	TC0_IO0A	UART_TX	EPWM_EP1	TC3_BUZZ	LED_S4	TCH4	-	EXI0	IO	-	Z
16	17	14	PD0.1	TC0_IO0B	UART_RX	TC0_CLK0	VREF+	LED_S5	TCH5	-	EXI1	IO	-	Z
17	-	-	PC0.2	EPWM_EP2	TC1_OUT	TC3_BUZZ	-	LED_S6	TCH6	-	EXI2	IO	-	Z
18	18	15	**VSS**	-	-	-	-	-	-	-	-	GND	-	P
19	19	16	**VDD**	-	-	-	-	-	-	-	-	POWER	-	P
20	20	17	PA0.11	TC0_IO1A	EPWM2_X	-	EPWM1_X	-	ECP0	ADC5	EXI11	IO	-	Z
21	21	18	PA0.12	TC0_IO1B	EPWM2_Y	-	EPWM1_Y	-	TCH16	ADC6/CPINP6	EXI12	IO		Z
22	22	19	PA0.13	TC0_IO2A	EPWM1_X	-	EPWM0_X	-	TCH17	ADC7/CPINN4	EXI13	IO	-	Z
23	23	20	PA0.14	TC0_IO2B	EPWM1_Y	-	EPWM0_Y	-	TCH18	ADC8/CPINP7	EXI14	IO	-	Z
24	24	21	PC0.3	EPWM_EP3	EPWM0_X	-	-	-	TCH19	-	EXI3	IO	-	Z
25	25	22	PA0.15	TC1_OUT	EPWM0_Y	TC0_CLK1	CP2_OUT	LED_S7	TCH7	-	EXI15	IO	-	Z
26	26	23	PA1.0	TC2_CAP0	TC0_IO0A	TC1_CAP	-	-	-	ADC9/OPA1P	EXI0	IO	-	Z
27	27	24	PA1.1	TC1_CAP	TC0_IO0B	UART_TX	I2C_SDA	-	-	ADC10/OPA1X	EXI1	IO	-	Z
28	28	1	PA1.2	TC2_CAP1	TC0_IO1A	UART_RX	I2C_SCL	-	-	OPA1N	EXI2	IO	-	Z
29	1	2	PA1.3	SPI_SCK	TC0_IO1B	CLO	-	**LED_COM0**	TCH8	ADC11/OPA0X	EXI3	IO	-	Z
30	2	3	PA1.4	SPI_MISO	TC0_IO2A	CP3_OUT	EPWM0_X	**LED_COM1**	TCH9	ADC0/OPA0P/CPINP8	EXI4	IO	-	Z
31	3	4	PA1.5	SPI_MOSI	TC0_IO2B	EPWM_EP4	EPWM1_X	**LED_COM2**	TCH10	ADC1/OPA0N/CPINP9	EXI5	IO	-	Z
32	-	-	PC0.0	SPI_NSS	ADC_ETR	TC0_ESYNC	CP4_OUT	**LED_COM3**	TCH11	-	EXI0	IO	-	Z

注意：

1) 外部复位功能和PA0.0管脚复用，可以使用User Option功能选择配置
2) F_SCLK(PA0.3), F_SDAT(PA0.4)为外部闪存烧录工具接口信号
3) 每个IO管脚只要配置成数字IO功能，都可以使用EXI功能来触发中断

2.4 管脚复用

下表归纳了各种功能的管脚复用，方便用户在各种不同应用下使用各种不同的功能。

Table 2-2 复用功能概要

功能模块	功能管脚	管脚分配
TIMER0	TC0_IO0A(B)	PD0.0(AF1) PA1.0(AF2) PB0.1(AF3) PA0.3(AF4)
	TC0_IO0B(B)	PD0.1(AF1) PA1.1(AF2) PA0.0(AF3) PA0.4(AF4)
	TC0_IO1A(B)	PA0.11(AF1) PA1.2(AF2)
	TC0_IO1B(B)	PA0.12(AF1) PA1.3(AF2)
	TC0_IO2A(B)	PA0.13(AF1) PA1.4(AF2)
	TC0_IO2B(B)	PA0.14(AF1) PA1.5(AF2)
	TC0_ETR(I)	PB0.1(AF2) PC0.0(AF3)
	TC0_CLK0(I)	PA0.1(AF2) PD0.1(AF3)
	TC0_CLK1(I)	PA0.15(AF3)
	TC0_CLK2(I)	PC0.1(AF3)
TIMER1	TC1_OUT(O)	PA0.15(AF1) PC0.2(AF2) PA0.6(AF2) PB0.0(AF3)
	TC1_CAP(I)	PA1.1(AF1) PA1.0(AF3)
	TC1_CLK(I)	PA0.0(AF2)
TIMER2	TC2_CAP0(I)	PA1.0(AF1) PA0.7(AF2)
	TC2_CAP1(I)	PA1.2(AF1)
CMP	CP0_OUT(O)	PA0.0(AF4) PA0.1(AF3)
	CP1_OUT(O)	PA0.1(AF4)

	CP2_OUT(O)	PA0.15(AF4)
	CP3_OUT(O)	PA1.4(AF3)
	CP4_OUT(O)	PC0.0(AF4)
EPWM	EPWM0_X	PA0.6(AF3) PA0.9(AF1) PA0.13(AF4) PC0.3(AF2)
	EPWM0_Y	PA0.5(AF3) PA0.10(AF1) PA0.14(AF4) PA0.15(AF2)
	EPWM1_X	PA0.7(AF1) PA0.13(AF2) PA0.11(AF4) PA1.5(AF4)
	EPWM1_Y	PA0.8(AF1) PA0.14(AF2) PA0.12(AF4)
	EPWM2_X	PA0.5(AF1) PA0.11(AF2) PC0.1(AF4)
	EPWM2_Y	PA0.6(AF1) PA0.12(AF1)
	EPWM_EP0	PA0.2(AF2)
	EPWM_EP1	PD0.0(AF3)
	EPWM_EP2	PC0.2(AF1)
	EPWM_EP3	PC0.3(AF1)
	EPWM_EP4	PA1.5(AF3)
SPI	SPI_NSS(B)	PC0.0(AF1) PB0.0(AF2) PA0.7(AF3)
	SPI_SCK(B)	PA1.3(AF1) PA0.8(AF3)
	SPI_MISO(B)	PA1.4(AF1) PA0.9(AF3)
	SPI_MOSI(B)	PA1.5(AF1) PA0.10(AF3)
ADC	ADC_ETR(I)	PC0.1(AF1) PC0.0(AF2) PA0.2(AF4)
	ADC0(A)	PA1.4(AF7)

	ADC1(A)	PA1.5(AF7)
	ADC2(A)	PB0.0(AF7)
	ADC3(A)	PB0.1(AF7)
	ADC4(A)	PA0.0(AF7)
	ADC5(A)	PA0.11(AF7)
	ADC6(A)	PA0.12(AF7)
	ADC7(A)	PA0.13(AF7)
	ADC8(A)	PA0.14(AF7)
	ADC9(A)	PA1.0(AF7)
	ADC10(A)	PA1.1(AF7)
	ADC11(A)	PA1.3(AF7)
	ADC12(A)	PA0.5(AF7)
	ADC13(A)	PA0.6(AF7)
	ADC14(A)	PA0.7(AF7)
	ADC15(A)	PA0.8(AF7)
	ADC16(A)	PA0.9(AF7)
	ADC17(A)	PA0.10(AF7)
I2C	SCL(B)	PA0.8(AF2) PA0.3(AF3) PA1.2(AF4)
	SDA(B)	PA0.9(AF2) PA0.4(AF3) PA1.1(AF4)
USART	USART_RX(I)	PB0.0(AF1) PA0.3(AF2)
	USART_TX (O)	PB0.1(AF1) PA0.4(AF2)
	USART_CK(B)	PA0.5(AF2) PA0.2(AF3)
UART	UART_RX(I)	PD0.1(AF2) PA1.2(AF3)
	UART_TX(O)	PD0.0(AF2) PA1.1(AF3)
LED	LED_S0(O)	PA0.1(AF5)
	LED_S1(O)	PA0.2(AF5)
	LED_S2(O)	PA0.3(AF5)
	LED_S3(O)	PA0.4(AF5)
	LED_S4(O)	PA0.8(AF4) PD0.0(AF5)

	LED_S5(O)	PA0.9(AF4) PD0.1(AF5)
	LED_S6(O)	PA0.10(AF4) PC0.2(AF5)
	LED_S7(O)	PA0.15(AF5)
	LED_COM0(O)	PA1.3(AF5)
	LED_COM1(O)	PA1.4(AF5)
	LED_COM2(O)	PA1.5(AF5)
	LED_COM3(O)	PA1.6(AF5)
	LED_COM4(O)	PC0.1(AF5)
	LED_COM5(O)	PB0.0(AF5)
	LED_COM6(O)	PB0.1(AF5)
	LED_COM7(O)	PA0.0(AF5)
OPA	OPA0P(A)	PA1.4(AF7)
	OPA0N(A)	PA1.5(AF7)
	OPA0X(A)	PA1.3(AF7)
	OPA1P(A)	PA1.0(AF7)
	OPA1N(A)	PA1.2(AF7)
	OPA1X(A)	PA1.1(AF7)
CMP	CMPINP0	PB0.0(AF7)
	CMPINN0	PB0.1(AF7)
	CMPINP1	PA0.5(AF7)
	CMPINN1	PA0.0(AF7)
	CMPINP2	PA0.6(AF7)
	CMPINN2	PA0.8(AF7)
	CMPINP3	PA0.7(AF7)
	CMPINN3	PA0.10(AF7)
	CMPINP4	PA0.8(AF7)
	CMPINN4	PA0.13(AF7)
	CMPINP5	PA0.9(AF7)
	CMPINP6	PA0.12(AF7)
	CMPINP7	PA0.14(AF7)
	CMPINP8	PA1.4(AF7)
	CMPINP9	PA1.5(AF7)
TOUCH	ECP0	PA0.11(AF6)
	TCH0	PA0.1(AF6)
	TCH1	PA0.2(AF6)
	TCH2	PA0.3(AF6)

	TCH3	PA0.4(AF6)
	TCH4	PD0.0(AF6)
	TCH5	PD0.1(AF6)
	TCH6	PC0.2(AF6)
	TCH7	PA0.15(AF6)
	TCH8	PA1.3(AF6)
	TCH9	PA1.4(AF6)
	TCH10	PA1.5(AF6)
	TCH11	PC0.0(AF6)
	TCH12	PC0.1(AF6)
	TCH13	PB0.0(AF6)
	TCH14	PB0.1(AF6)
	TCH15	PA0.0(AF6)
	TCH16	PA0.12(AF6)
	TCH17	PA0.13(AF6)
	TCH18	PA0.14(AF6)
	TCH19	PC0.3(AF6)
SYSTEM	CLO(O)	PA0.0(AF1) PC0.1(AF2) PA1.3(AF3)

注意：

1) 对于输出功能，如果多个管脚都被配置成同一个功能，那么所有这些管脚都会输出相同的信号。

2) 对于输入功能，如果多个管脚都被配置成同一个功能，那么AF编号小的管脚有更高的优先权呢。例如，当PD0.1和PA1.2都被配置成RX时，只有PD0.1(AF2)是RX，而PA1.2(AF3)的RX配置无效。

2.5 管脚功能说明

本段落描述了以下管脚的功能：

- 电源管脚
- 系统功能管脚
- 普通模块功能管脚
- 调试接口管脚
- 闪存烧录工具管脚

注意：
1) D: 数字; A: 模拟
2) I/O: 双向; I: 输入; O: 输出
3) P: 电源; G: 地
4) Z: 高阻

2.5.1 电源管脚

Table 2-3 电源管脚说明

模块	管脚名称	I/O	管脚说明	D/A
电源	VDD	-	芯片电源	-
	VSS	-	芯片地	-

2.5.2 系统功能管脚

Table 2-4 系统功能管脚说明

模块	管脚名称	I/O	管脚说明	D/A
系统	RSTB	I	硬件复位输入，当PA0.0选择RESETB时，内部没有上拉电阻，所以外部需要一个典型值为250K欧姆的上拉电阻。	D
	XIN	I	外部主晶振的输入	A
	XOUT	O	外部主晶振的输出	A
	CLO	O	内部系统时钟输出	D
	EXIx	I	外部中断输入通道	D

2.5.3 普通模块功能管脚

Table 2-5 普通模块功能管脚说明

模块	管脚名称	I/O	管脚说明	D/A
GPIO	PA0.x	I/O	通用IO A	D
	PB0.x	I/O	通用IO B	D

	PC0.x	I/O	通用IO C	D
	PD0.x	I/O	通用IO D	D
TIMER0	TC0_ETR	I	TC0外部触发输入	D
	TC0_CLK0	I	TC0的外部输入时钟通道0	D
	TC0_CLK1	I	TC0的外部输入时钟通道1	D
	TC0_CLK2	I	TC0的外部输入时钟通道2	D
	TC0_IO0A	I/O	TC0通道0的A相信号	D
	TC0_IO0B	I/O	TC0通道0的B相信号	D
	TC0_IO1A	I/O	TC0通道1的A相信号	D
	TC0_IO1B	I/O	TC0通道1的B相信号	D
	TC0_IO2A	I/O	TC0通道2的A相信号	D
	TC0_IO2B	I/O	TC0通道2的B相信号	D
TIMER1	TC1_CLK	I	TC1的外部时钟输入	D
	TC1_CAP	I	TC1的外部捕获输入	D
	TC1_OUT	O	TC1的输出	D
TIMER2	TC2_CAP0	I	TC2通道0的外部捕获输入	D
	TC2_CAP1	I	TC2通道1的外部捕获输入	D
EPWM	EPWM0_X	O	EPWM通道0的X相输出	D
	EPWM0_Y	O	EPWM通道0的Y相输出	D
	EPWM1_X	O	EPWM通道1的X相输出	D
	EPWM1_Y	O	EPWM通道1的Y相输出	D
	EPWM2_X	O	EPWM通道2的X相输出	D
	EPWM2_Y	O	EPWM通道2的Y相输出	D
I2C	I2C_SCL	I/O	I2C串行时钟	D
	I2C_SDA	I/O	I2C串行数据	D
SPI	SPI_NSS	I/O	帧或者从机片选输出（主机模式） 帧输入（从机模式）	D
	SPI_SCK	I/O	SPI串行时钟	D
	SPI_MISO	I/O	主机输入从机输出数据	D
	SPI_MOSI	I/O	主机输出从机输入数据	D
USART	USART_RX	I	串行数据接收	D
	USART_TX	O	串行数据发送	D
	USART_CK	I/O	串行时钟	D
UART	UART_RX	I	UART串行数据接收	D
	UART_TX	O	UART串行数据发送	D

LED	LED_Sx	O	LED的Segment输出	D
	LED_COMx	O	LED的COM扫描输出（大电流驱动I/O）	D
ADC	ADCx	I	ADC模拟输入通道	A
	ADC_ETR	I	ADC外部启动触发输入	D
TOUCH	TCHx	I/O	触摸按键扫描通道	A
	ECP0	I/O	触摸按键在外部电容模式中的外接电容管脚	A
CMP	CP0_OUT	O	比较器0输出端口	D
	CP1_OUT	O	比较器1输出端口	D
	CP2_OUT	O	比较器2输出端口	D
	CP3_OUT	O	比较器3输出端口	D
	CP4_OUT	O	比较器4输出端口	D
	CPINPx	I	比较器正向输入通道	A
	CPINNx	I	比较器负向输入通道	A
OPA	OPAxP	I	运算放大器的正向输入端	A
	OPAxN	I	运算放大器的负向输入端	A
	OPAxX	O	运算放大器的输出端	A

2.5.4 调试接口管脚

Table 2-6 调试接口管脚说明

模块	管脚名称	I/O	管脚说明	D/A
SWD	SWCLK	I	串行时钟，内部上拉	D
	SWDIO	I/O	串行数据输入/输出，内部上拉	D

2.5.5 闪存烧录工具管脚

Table 2-7 闪存烧录工具管脚说明

模块	管脚名称	I/O	管脚说明	D/A
FLASH	F_SCLK	I	串行时钟	D
	F_SDAT	I/O	串行数据	D
	F_RST	I	复位	D
	VDD	P	电源 (烧录时建议在电源和地之间连接0.1uF电容)	A
	VSS	G	地	A

3 系统存储空间

3.1 概述

本章节介绍了 APT32F172 系统存储空间管理。

本章包含内容如下：

- 存储地址表
- 特殊功能寄存器表，包含内容如下：
    - CPU特殊功能寄存器表
    - 外围设备特殊功能寄存器表

3.2 默认存储地址表

Table 3-1 存储地址

Address	Memory
Reserved	Reserved
0xE000_0000 to 0xE00F_FFFF	CPU内部寄存器
Reserved	Reserved
0x4000_0000 to 0x400F_FFFF	特殊功能寄存器 (SFR)
Reserved	Reserved
0x2000_0000 to 0x2001_FFFF	SRAM
Reserved	Reserved
0x1000_0000 to 0x1001_FFFF	数据闪存 (Data Flash)
Reserved	Reserved
0x0000_0000 to 0x0007_FFFF	程序闪存 (Program Flash)

3.3 特殊功能寄存器表

特殊功能寄存器表有如下两种形式

- CPU特殊功能寄存器表
- 外围设备特殊功能寄存器表

3.3.1 CPU特殊功能寄存器表

Table 3-2 CPU SFR 表

Address	Function Description
0xE000_EFA0 to 0xE00F_FFFF	Reserved
0xE000_EF90 to 0xE000_EF9F	电源管理控制器
0xE000_ED00 to 0xE000_EF8F	Reserved
0xE000_E100 to 0xE000_ECFF	VIC控制器
0xE000_E010 to oxE000_E0FF	系统定时器
0xE000_0000 to ox E000_E00F	Reserved

3.3.2 外围设备特殊功能寄存器表

Table 3-3 外围设备SFR表

Peripheral	Base Address	Function Description
OPA	0x400C_0000	运算放大器控制器 (OPA)
CMP	0x400B_4000	比较器4控制器 (CMP4)
	0x400B_3000	比较器3控制器 (CMP3)
	0x400B_2000	比较器2控制器 (CMP2)
	0x400B_1000	比较器1控制器 (CMP1)
	0x400B_0000	比较器0控制器 (CMP0)
I2C	0x400A_0000	I2C串行接口 (I2C)
SPI	0x4009_0000	同步并行接口 (SPI)
UART	0x4008_1000	通用异步收发器1 (UART1)
	0x4008_0000	通用异步/同步收发器0 (USART0)
LED	0x4006_0000	LED显示控制器 (LED)
TIMER	0x4005_4000	EPWM模块
	0x4005_3000	16位实时定时器 (TC3)
	0x4005_2000	16位两路同步简单定时器 (TC2)
	0x4005_1000	32位通用定时器/计数器 (TC1)
	0x4005_0000	16位三路同步通用定时器 (TC0)
GPIO	0x4004_4000	EXIGRP

	0x4004_3000	通用IO端口-D (GPIO D)
	0x4004_2000	通用IO端口-C (GPIO C)
	0x4004_1000	通用IO端口-B (GPIO B)
	0x4004_0000	通用IO端口-A (GPIO A)
ADC	0x4003_0000	模数转换器 (ADC)
TKEY	0x4002_0000	电容式触摸按键传感器 (TOUCH)
SYSTEM	0x4001_1000	系统控制器 (SYSCON)
	0x4001_0000	闪存控制器 (IFC)
RSVD	0x4000_0000	设备信息寄存器 (Device ID)

4 中断向量控制器(INTC)

4.1 概述

中断控制器是用于收集来自于多个中断源的中断请求，依据中断优先级对中断请求进行仲裁并提交给CPU的接口逻辑。CPU支持可嵌套的抢占式中断响应处理。在CPU处理当前中断的过程中，如果有更高优先级的中断请求，CPU将挂起当前中断而转入处理更高优先级的中断请求。当高优先级的中断处理完成以后，CPU将恢复被挂起的中断继续执行。中断控制器只允许高优先级的中断请求抢占低优先级的中断，但不允许同级别或者更低优先级的中断抢占。

4.1.1 特性

- 最大支持32个通道的中断源（IRQ[31:0]）
- 每个中断源具有独立的可编程的中断优先级设置和中断使能控制
- 在中断处理过程中，支持优先级的动态调整
- 独立的中断唤醒和中断使能配置（中断唤醒类似于Event事件）
- 每个中断源具有独立的中断向量号

4.2 中断向量表

Table 4-1 System Interrupt Vectors

Number	Address	Vector	Interrupt Sources
32	0x0000_0080	CORET	CPU Core Timer
33	0x0000_0084	SYSCON	System controller interrupt
34	0x0000_0088	IFC	Program flash controller interrupt
35	0x0000_008C	ADC	ADC Interrupt
36	0x0000_0090	TC0_0	GPT Sub Timer 0 Interrupt
37	0x0000_0094	TC0_1	GPT Sub Timer 1 Interrupt
38	0x0000_0098	TC0_2	GPT Sub Timer 2 Interrupt
39	0x0000_009C	EXI0	External interrupt 0
40	0x0000_00A0	EXI1	External interrupt 1
41	0x0000_00A4	EPWM	Enhanced PWM
42	0x0000_00A8	TC1	TC16
43	0x0000_00AC	TC2	Simple Timer
44	0x0000_00B0	TC3	Watch Timer
45	0x0000_00B4	UART0	USART interrupt
46	0x0000_00B8	UART1	UART interrupt
47	0x0000_00BC	-	Reserved
48	0x0000_00C0	-	Reserved
49	0x0000_00C4	I2C	I2C interrupt
50	0x0000_00C8	-	Reserved
51	0x0000_00CC	SPI	SPI interrupt
52	0x0000_00D0	-	Reserved
53	0x0000_00D4	EXI2	External Interrupt 2 ~ 3
54	0x0000_00D8	EXI3	External Interrupt 4 ~ 9
55	0x0000_00DC	EXI4	External Interrupt 10 ~ 15
56	0x0000_00E0	-	Reserved
57	0x0000_00E4	TKEY	Touch Key interrupt
58	0x0000_00E8	-	Reserved
59	0x0000_00EC	LED	LED interrupt
60	0x0000_00F0	CMP0	CMP0, CMP2 Interrupt
61	0x0000_00F4	CMP1	CMP1, CMP3, CMP4 Interrupt
62	0x0000_00F8	-	Reserved
63	0x0000_00FC	-	Reserved

中断向量号，是请求在异常表的位置编号。0～30号向量是用作处理器内部识别的向量；31号向量是留给软件的，用作指向系统描述符指针；从32号开始的向量是留给外设请求的。

4.3 工作原理

4.3.1 中断处理

矢量中断控制器（NVIC）协同其外围逻辑，用于中断的高效处理。控制器最大可支持 32 个中断源，每个中断源拥有独立的软件可编程优先级。矢量中断控制器支持中断嵌套。当处理器正在处理一个中断请求的同时来了一个更高优先级的中断请求，处理器将中断当前中断服务程序的处理，响应该更高优先级的中断请求。在更高优先级的中断请求处理结束时，CPU 返回被打断的中断服务程序继续执行。NVIC 支持独立的软件可编程唤醒设置和中断使能设置。

进入中断服务程序中，需要软件清除外设的中断有效信号，否则当中断退出时会重新请求 CPU。另外，矢量中断控制器支持软件中断，软件可以通过设置中断设置等待有效寄存器（VIC_ISPR）置高相应的中断等待状态位，向 CPU 发送中断请求。

当处理器响应中断请求后，矢量中断控制器会自动清除等待状态位，软件也可以通过设置中断清除等待寄存器（VIC_ICPR）清除正在等待中的中断。如果外设的中断请求持续有效，将无法通过 VIC_ICPR 的方式清除等待的中断。

中断的处理过程可以分以下几个步骤进行：

- 外设产生中断请求信号，置高相应 IRQ 被 NVIC 捕捉到。
- VIC 根据 IRQ 申请，设置相应的 Pending 状态位。
- 经过优先级仲裁后向 CPU 发起中断请求。
- CPU 在当期指令执行完成同时响应中断，返回中断响应给 VIC，然后更新 EPSR 和 EPC，更新 PSR 中的 VEC 为当前请求的中断向量号，清除 PSR.EE，最后取得中断程序入口地址；VIC 根据 CPU 返回的中断响应信号清除 Pending 状态位和设置 Active 状态位。
- 进行中断现场保存（保存中断控制寄存器器现场 EPSR 和 EPC，打开 PSR.EE 和 PSR.IE 使能中断嵌套，然后保存中断通用寄存器现场）。
- CPU 开始处理中断程序，程序中需清除中断源有效信号，否则中断退出后会重入该中断。
- 中断现场恢复和中断退出（恢复中断通用寄存器，然后回复中断控制寄存器，退出 ISR。VIC 接收到 CPU 的退出信号，清除 Active 状态位）。

中断现场的保存可以通过在中断服务程序的开头执行 NIE 和 IPUSH 指令来完成；中断现场的恢复和退出可以通过在中都服务程序结尾执行 IPOP 和 NIR 指令来完成。

4.3.2 中断优先级和中断抢占

中断的优先级可以通过VIC_IPR0 ~3这四个寄存器来设置。每个VIC_IPR寄存器对应四个中断源的优先级设置。

中断的优先级共分为4级设置，通过VIC_IPR寄存器的PRI_x中的最高两位来设置。数值越小，代表的优先级越高，所以设置为‘0’时代表最高优先级。如果优先级号相同，则根据中断源号来决定优先的顺序，号码越小，优先级越高。例如，IRQ0 和 IRQ1的优先级号设置为相同，当IRQ0和IRQ1同时提交中断，由于IRQ0的中断源号小于IRQ1，因此IRQ0先得到CPU的响应。

将所有中断的优先级设置为统一的优先级时，可以禁止中断的抢占响应。也可以通过设置VIC_IPTR寄存器来定义可以发起中断抢占的优先级临界值。当设置了VIC_IPTR的THDEN，等待中断处理的中断请求优先级必须高于VIC_IPTR的PRITHD中所设置的优先级阈值，才能发起中断抢占请求。

中断抢占的优先级条件可分为两种：

- 当中断优先级阈值未使能时，中断抢占的优先级必须高于当前 CPU 正在处理的中断的优先级；同级优先级不能进行抢占。
- 当中断优先级阈值使能时，中断抢占的优先级不仅要高于当前 CPU 正在处理的中断优先级，而且要高于中断优先级阈值寄存器设置的阈值。VIC 支持中断优先级的动态调整，当被嵌套的中断优先级需要调高或者调低时，在设置中断优先级设置寄存器的同时设置中断优先级阈值寄存器。

下图给出了中断抢占的示例。中断优先级设置为：IRQ0<IRQ1<IRQ2<IRQ3；中断源请求产生的顺序为：IRQ0>IRQ1>IRQ2>IRQ3。CPU 首先响应了 IRQ0，在 IRQ0 中断服务程序执行的过程中，来了更高优先级的 IRQ1，因此 IRQ0 被抢占，CPU 开始执行 IRQ1 的中断服务程序。同样，IRQ2 对 IRQ1 进行了抢占，并设置了中断优先级阈值寄存器（VIC_IPTR.VECTHD = IRQ0，IPTR.PRITHD = 0，IPTR.THDEN = 1）。当 IRQ3 到来时，尽管优先级高于 IRQ2，但没有高于 IPTR，因此 IRQ3 无法抢占 IRQ2。IRQ3 在 IRQ0 的中断服务程序执行结束，清除了 IPTR.THDEN 后，才得到 CPU 响应。

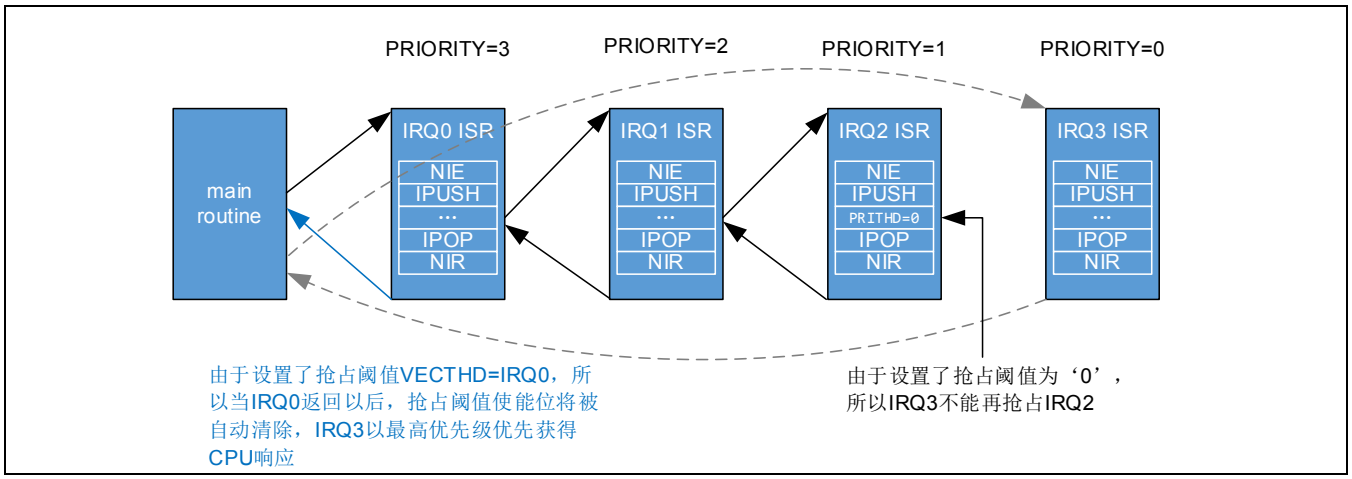


Figure 4-1 中断嵌套优先级示意图

4.3.3 中断响应时间和中断嵌套条件

一般中断在指令的边界上被确认，如下图所示。

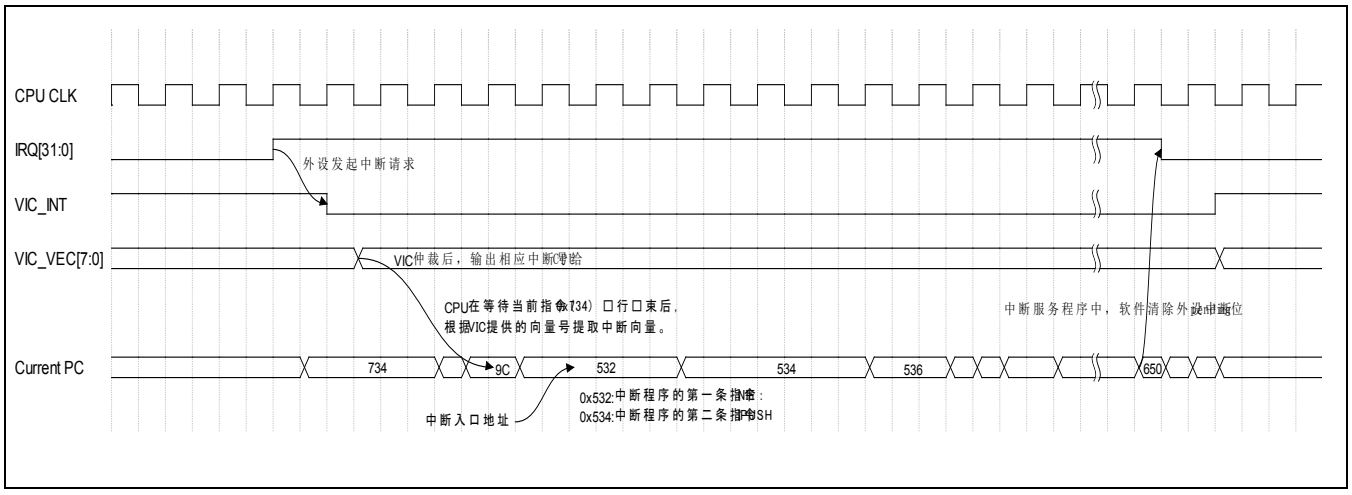


Figure 4-2 中断响应过程

当中断请求信号被置高，经过CPU的时钟采样以后触发VIC中断处理。VIC经过仲裁处理以后，向CPU发送相应的中断向量号，CPU内部收到中断后，根据向量号取得中断向量，进入中断服务程序。此过程需耗费时间为中断请求信号的同步时间，VIC的处理时间，CPU等待当前指令的执行完成的时间，以及CPU获取中断向量的时间总和。

中断响应时间不是具体固定的，而是和当前执行的指令所消耗的时间相关，如果中断的响应因为等待长指令周期的指令而延后，需要加速中断的响应时间，可以通过设置PSR.IC位，打断当前执行的指令。LDM、STM、PUSH、POP、IPUSH、IPOP等多周期指令可以被中断而不等它们完成，从而缩短中断响应延时。多周期指令NIE不可响应中断，NIR只在指令执行的末尾响应中断，不能被PSR（IC）位打断。

中断抢占在满足优先级的条件下，还需要判断当前中断响应的阶段。和中断嵌套相关的主要分为以下几个阶段：1）EPSR、EPC、PSR的更新和中断入口地址的读取；2）NIE指令；3）IPUSH指令；4）中断服务；5）IPOP指令；6）NIR指令。

为保证中断嵌套现场的保护和恢复，CPU在以下阶段不能被中断打断：

- 中断响应之后更新 EPSR、EPC、PSR 和获取中断入口地址的过程中。
- NIE 指令执行过程中。
- PSR.IC 位被关闭，IPUSH 和 IPOP 指令执行过程中。
- NIR 指令执行过程中。

CPU在以下阶段可以安全的响应新的中断：

- 正常程序的执行过程中，在中断响应之前。
- IPUSH、IPOP 指令执行完成。
- PSR.IC 位被打开，IPUSH、IPOP 指令执行过程中。
- NIR 指令执行完成。
- 中断服务处理过程中。

下图给出了IRQ0/IRQ1/IRQ2/IRQ3中断的嵌套过程。

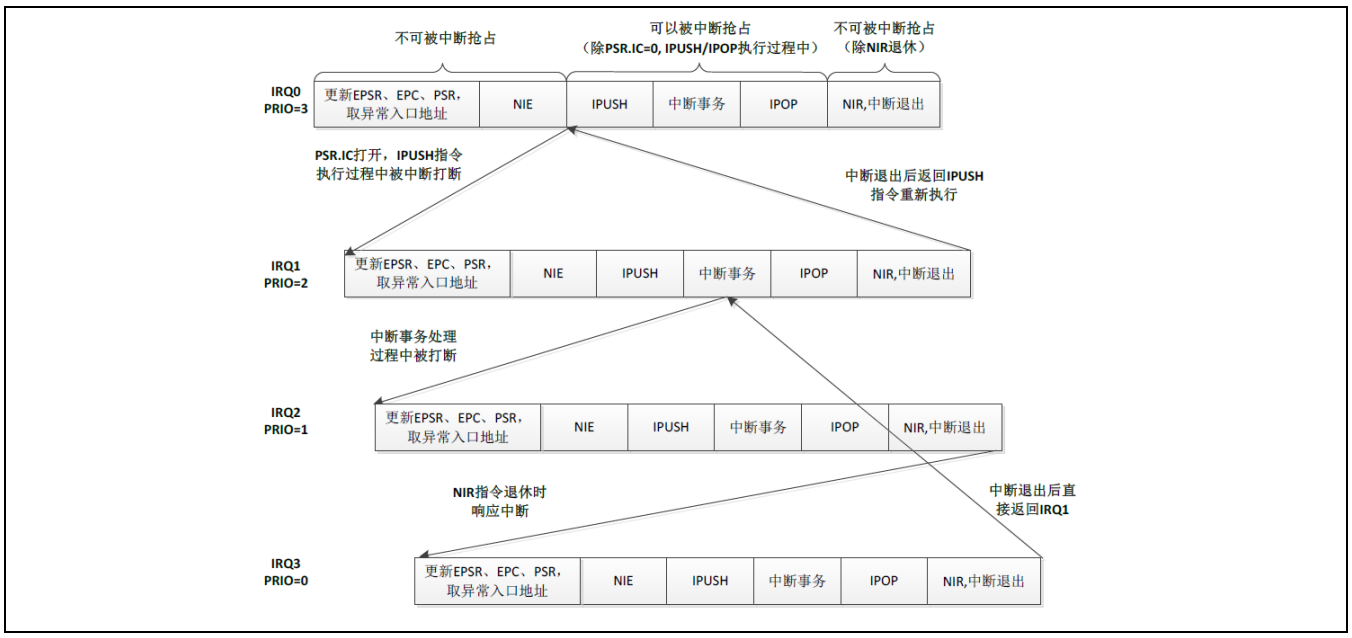


Figure 4-3 中断嵌套条件示例

在IRQ0被CPU响应后，产生了更高优先级的IRQ1，当PSR.IC打开时，在执行到IPUSH指令时响应IRQ1。IRQ2在IRQ1处理中断事务时产生，因此可立即被CPU响应。IRQ3在IRQ2执行NIR指令时产生，在NIR指令完成时响应IRQ3。当IRQ3处理完，退出中断服务程序时，直接返回到IRQ1被IRQ2打断的点。当IRQ1 返回IRQ0时，需要重新执行IPUSH指令。

4.3.4 中断唤醒

当CPU处于低功耗模式（DOZE、STOP）时，外设产生的中断可以将CPU从低功耗模式唤醒。如果一个中断的低功耗唤醒功能已经使能，而且该中断处于等待状态，VIC将产生低功耗唤醒请求。如果一个中断的低功耗唤醒功能未使能，即使该中断处于等待状态，VIC也不会产生低功耗唤醒请求。

需要注意，中断的使能（VIC_ISER）和中断的唤醒使能（VIC_IWER）分别控制中断的事务处理和唤醒功能。当两者都设置时，一个等待的中断请求既会产生中断事务处理请求，又会产生低功耗唤醒请求；当只有其中一个使能时，只激活对应设置的功能；两者都没有使能时，即使中断处于等待状态，VIC也不会产生任何请求。

在只使能了低功耗唤醒功能时，CPU被唤醒以后，由于没有进入相应中断服务程序，所以该中断在CPU内一直处于Pending状态。这会导致在下一次进入低功耗模式时，CPU立即退出低功耗模式。为保证在唤醒后，下一次能够正常进入低功耗模式，必须在CPU唤醒以后，通过软件清除CPU中该中断的Pending状态（VIC_ICPR）。

.

4.3.5 中断操作步骤

中断的配置基本分为两个级别，一个处于外设内部的配置，另外一个处于VIC内部的配置。要使能某个特定外设的中断，首先需要配置该外设内部的中断控制寄存器，使能外设的特定中断；再配置VIC内部的中断控制，首先设置VIC_IPR0~7，设置中断优先级，然后设置IVC_ISER，使能该外设所对应的中断号。CPU具有全局中断使能控制，在CPU响应VIC中断请求之前，必须使能PSR.IE/EE，否则CPU无法响应中断。当某个特定外设的中断发生以后，外设内部的中断pending位首先会置位，随之触发VIC内部相对应的中断源pending位置位。外设内部的中断pending位需要程序在软件中清除，而VIC中的pending位在处理器响应中断请求后，会自动清除。也可以通过设置中断清除等待寄存器（VIC_ICPR）强制清除还没有被处理器响应的中断请求。

VIC可以通过VIC_ISPR，软件触发相应的中断源。VIC为每个中断源提供两种状态查询，分别为：

- Pending：查询是否存在等待 CPU 处理的中断请求。
  0：表示该中断源没有等待的中断请求；
  1：表示该中断源有等待的中断请求。

- Active：查询 CPU 是否响应该中断源但是还没有处理完成该中断请求。
  0：表示该中断源没有被CPU响应；
  1：表示该中断源已经被CPU响应，但是还没有处理完成。

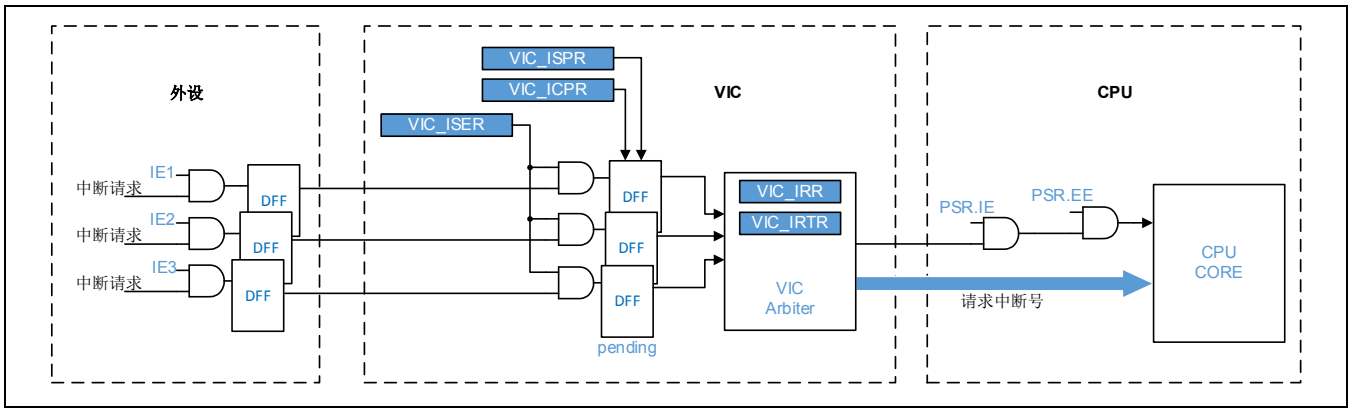


Figure 4-4 中断配置结构示意图

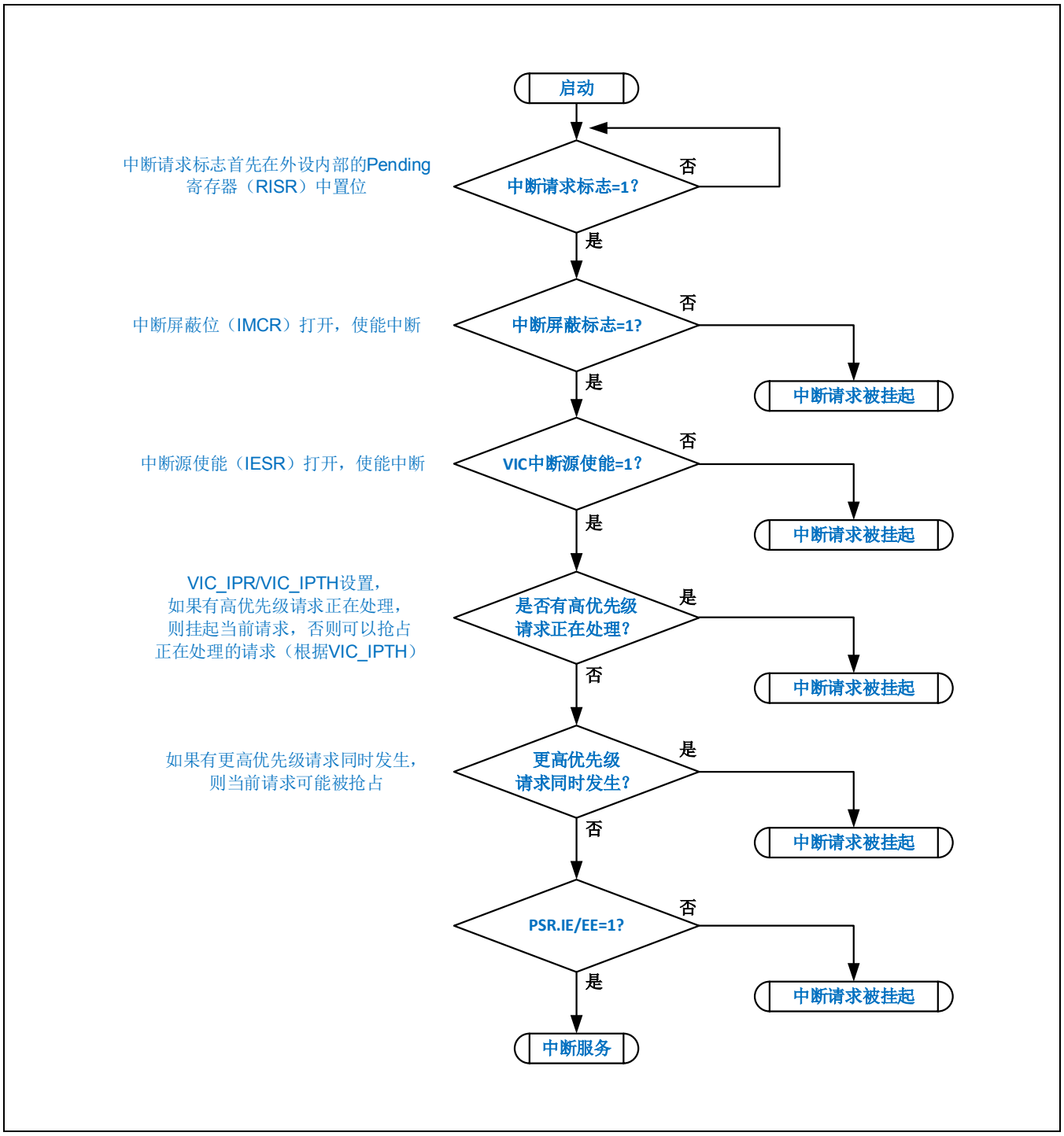


Figure 4-5 中断请求处理流程

4.4 寄存器说明

4.4.1 寄存器表

- Base Address: 0xE000_E000

Register	Offset	Description	Reset Value
VIC_ISER	0x100	Interrupt Set Enable Register	
RSVD	0x104 - 0x13F	Reserved	
VIC_IWER	0x140	Interrupt Wakeup Enable Register	0x0000_0000
RSVD	0x144 - 0x17F	Reserved	0x0000_0000
VIC_ICER	0x180	Interrupt Clear Enable Register	0x0000_0000
RSVD	0x184 - 0x1BF	Reserved	0x0000_0000
VIC_IWDR	0x1C0	Interrupt Wakeup Disable Register	0x0000_0000
RSVD	0x1C4 - 0x1FF	Reserved	-
VIC_ISPR	0x200	Interrupt Set Pending Register	
RSVD	0x204 - 0x27F	Reserved	0x0000_0000
VIC_ICPR	0x280	Interrupt Clear Pending Register	0x0000_0000
RSVD	0x284 - 0x2FF	Reserved	0x0000_0000
VIC_IABR	0x300	Interrupt Active Status Register	0x0000_0000
RSVD	0x304 - 0x3FF	Reserved	
VIC_IPR0	0x400	Interrupt Priority Register 0	
VIC_IPR1	0x404	Interrupt Priority Register 1	
VIC_IPR2	0x408	Interrupt Priority Register 2	
VIC_IPR3	0x40C	Interrupt Priority Register 3	
VIC_IPR4	0x410	Interrupt Priority Register 4	
VIC_IPR5	0x414	Interrupt Priority Register 5	
VIC_IPR6	0x418	Interrupt Priority Register 6	

VIC_IPR7	0x41C	Interrupt Priority Register 7	
RSVD	0x420 - 0xBFF	Reserved	
VIC_ISR	0xC00	Interrupt Status Register	
VIC_IPTR	0xC04	Interrupt Priority Threshold Register	
RSVD	0xC08 - 0xCFF	Reserved	

NOTE:

4.4.2 VIC_ISER (中断设置使能寄存器)

- Address = Base Address + 0x0100, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETENA31	SETENA30	SETENA29	SETENA28	SETENA27	SETENA26	SETENA25	SETENA24	SETENA23	SETENA22	SETENA21	SETENA20	SETENA19	SETENA18	SETENA17	SETENA16	SETENA15	SETENA14	SETENA13	SETENA12	SETENA11	SETENA10	SETENA9	SETENA8	SETENA7	SETENA6	SETENA5	SETENA4	SETENA3	SETENA2	SETENA1	SETENA0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
SETENAx	[31:0]	RW	中断向量号使能 读操作： 0： 对应中断未使能 1： 对应中断已使能 写操作： 0： 无效 1： 使能对应中断	0x0

4.4.3 VIC_IWER (中断低功耗唤醒使能寄存器)

- Address = Base Address + 0x0140, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETENA31	SETENA30	SETENA29	SETENA28	SETENA27	SETENA26	SETENA25	SETENA24	SETENA23	SETENA22	SETENA21	SETENA20	SETENA19	SETENA18	SETENA17	SETENA16	SETENA15	SETENA14	SETENA13	SETENA12	SETENA11	SETENA10	SETENA9	SETENA8	SETENA7	SETENA6	SETENA5	SETENA4	SETENA3	SETENA2	SETENA1	SETENA0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
SETENAx	[31:0]	RW	设置中断低功耗唤醒功能 读操作： 0： 对应中断的低功耗唤醒未使能 1： 对应中断的低功耗唤醒已使能 写操作： 0： 无效 1： 使能对应中断的低功耗唤醒功能	0x0

4.4.4 VIC_ICER (中断使能清除寄存器)

- Address = Base Address + 0x0180, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRENA31	CLRENA30	CLRENA29	CLRENA28	CLRENA27	CLRENA26	CLRENA25	CLRENA24	CLRENA23	CLRENA22	CLRENA21	CLRENA20	CLRENA19	CLRENA18	CLRENA17	CLRENA16	CLRENA15	CLRENA14	CLRENA13	CLRENA12	CLRENA11	CLRENA10	CLRENA9	CLRENA8	CLRENA7	CLRENA6	CLRENA5	CLRENA4	CLRENA3	CLRENA2	CLRENA1	CLRENA0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
CLRENAx	[31:0]	RW	清除中断使能 读操作： 0： 对应中断未使能 1： 对应中断已使能 写操作： 0： 无效 1： 清除对应中断的使能	0x0

4.4.5 VIC_IWDR (中断低功耗唤醒清除寄存器)

- Address = Base Address + 0x01C0, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRENA31	CLRENA30	CLRENA29	CLRENA28	CLRENA27	CLRENA26	CLRENA25	CLRENA24	CLRENA23	CLRENA22	CLRENA21	CLRENA20	CLRENA19	CLRENA18	CLRENA17	CLRENA16	CLRENA15	CLRENA14	CLRENA13	CLRENA12	CLRENA11	CLRENA10	CLRENA9	CLRENA8	CLRENA7	CLRENA6	CLRENA5	CLRENA4	CLRENA3	CLRENA2	CLRENA1	CLRENA0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
CLRENAx	[31:0]	RW	清除中断低功耗唤醒功能 读操作： 0： 对应中断的低功耗唤醒未使能 1： 对应中断的低功耗唤醒已使能 写操作： 0： 无效 1： 清除对应中断的低功耗唤醒功能	0x0

4.4.6 VIC_ISPR (中断等待设置寄存器)

- Address = Base Address + 0x0200, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETPEND31	SETPEND30	SETPEND29	SETPEND28	SETPEND27	SETPEND26	SETPEND25	SETPEND24	SETPEND23	SETPEND22	SETPEND21	SETPEND20	SETPEND19	SETPEND18	SETPEND17	SETPEND16	SETPEND15	SETPEND14	SETPEND13	SETPEND12	SETPEND11	SETPEND10	SETPEND9	SETPEND8	SETPEND7	SETPEND6	SETPEND5	SETPEND4	SETPEND3	SETPEND2	SETPEND1	SETPEND0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
SETPENDx	[31:0]	RW	更改中断的等待状态 读操作： 0： 对应中断未处于等待状态 1： 对应中断已处于等待状态 写操作： 0： 无效 1： 改变对应中断为等待状态	0x0

4.4.7 VIC_ICPR (中断等待清除寄存器)

- Address = Base Address + 0x0280, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRPEND31	CLRPEND30	CLRPEND29	CLRPEND28	CLRPEND27	CLRPEND26	CLRPEND25	CLRPEND24	CLRPEND23	CLRPEND22	CLRPEND21	CLRPEND20	CLRPEND19	CLRPEND18	CLRPEND17	CLRPEND16	CLRPEND15	CLRPEND14	CLRPEND13	CLRPEND12	CLRPEND11	CLRPEND10	CLRPEND9	CLRPEND8	CLRPEND7	CLRPEND6	CLRPEND5	CLRPEND4	CLRPEND3	CLRPEND2	CLRPEND1	CLRPEND0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
CLRPENDx	[31:0]	RW	清除中断的等待状态 读操作： 0： 对应中断未处于等待状态 1： 对应中断已处于等待状态 写操作： 0： 无效 1： 清除对应中断的等待状态	0x0

4.4.8 VIC_IABR (中断响应状态寄存器)

- Address = Base Address + 0x0300, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACTIVE31	ACTIVE30	ACTIVE29	ACTIVE28	ACTIVE27	ACTIVE26	ACTIVE25	ACTIVE24	ACTIVE23	ACTIVE22	ACTIVE21	ACTIVE20	ACTIVE19	ACTIVE18	ACTIVE17	ACTIVE16	ACTIVE15	ACTIVE14	ACTIVE13	ACTIVE12	ACTIVE11	ACTIVE10	ACTIVE9	ACTIVE8	ACTIVE7	ACTIVE6	ACTIVE5	ACTIVE4	ACTIVE3	ACTIVE2	ACTIVE1	ACTIVE0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
ACTIVEx	[31:0]	RW	指示对应的中断源是否已经被CPU响应但还没有处理完成。 读操作： 0： 没有被CPU响应 1： 已经被CPU响应，但还没有处理完 写操作： 0： 清除当前Active状态 1： 不允许（软件写1可能导致不可预期的错误）	0x0

4.4.9 VIC_IPR0 (中断优先级设置寄存器0)

- Address = Base Address + 0x0400, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
PRI_x	[7:6] [15:14] [23:22] [31:30]	RW	设置对应中断号的优先级，数值越小，优先级越高 PRI_0： 中断号0的优先级设置 PRI_1： 中断号1的优先级设置 PRI_2： 中断号2的优先级设置 PRI_3： 中断号3的优先级设置	0x0

4.4.10 VIC_IPR1 (中断优先级设置寄存器1)

- Address = Base Address + 0x0404, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
PRI_x	[7:6] [15:14] [23:22] [31:30]	RW	设置对应中断号的优先级，数值越小，优先级越高 PRI_4： 中断号4的优先级设置 PRI_5： 中断号5的优先级设置 PRI_6： 中断号6的优先级设置 PRI_7： 中断号7的优先级设置	0x0

4.4.11 VIC_IPR2 (中断优先级设置寄存器2)

- Address = Base Address + 0x0408, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
PRI_x	[7:6] [15:14] [23:22] [31:30]	RW	设置对应中断号的优先级，数值越小，优先级越高 PRI_8： 中断号8的优先级设置 PRI_9： 中断号9的优先级设置 PRI_10： 中断号10的优先级设置 PRI_11： 中断号11的优先级设置	0x0

4.4.12 VIC_IPR3 (中断优先级设置寄存器3)

- Address = Base Address + 0x040C, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
PRI_x	[7:6] [15:14] [23:22] [31:30]	RW	设置对应中断号的优先级，数值越小，优先级越高 PRI_12： 中断号12的优先级设置 PRI_13： 中断号13的优先级设置 PRI_14： 中断号14的优先级设置 PRI_15： 中断号15的优先级设置	0x0

4.4.13 VIC_IPR4 (中断优先级设置寄存器4)

- Address = Base Address + 0x0410, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
PRI_x	[7:6] [15:14] [23:22] [31:30]	RW	设置对应中断号的优先级，数值越小，优先级越高 PRI_16： 中断号16的优先级设置 PRI_17： 中断号17的优先级设置 PRI_18： 中断号18的优先级设置 PRI_19： 中断号19的优先级设置	0x0

4.4.14 VIC_IPR5 (中断优先级设置寄存器5)

- Address = Base Address + 0x0414, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
PRI_x	[7:6] [15:14] [23:22] [31:30]	RW	设置对应中断号的优先级，数值越小，优先级越高 PRI_20： 中断号20的优先级设置 PRI_21： 中断号21的优先级设置 PRI_22： 中断号22的优先级设置 PRI_23： 中断号23的优先级设置	0x0

4.4.15 VIC_IPR6 (中断优先级设置寄存器6)

- Address = Base Address + 0x0418, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
PRI_x	[7:6] [15:14] [23:22] [31:30]	RW	设置对应中断号的优先级，数值越小，优先级越高 PRI_24： 中断号24的优先级设置 PRI_25： 中断号25的优先级设置 PRI_26： 中断号26的优先级设置 PRI_27： 中断号27的优先级设置	0x0

4.4.16 VIC_IPR7 (中断优先级设置寄存器7)

- Address = Base Address + 0x041C, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
PRI_x	[7:6] [15:14] [23:22] [31:30]	RW	设置对应中断号的优先级，数值越小，优先级越高 PRI_28： 中断号28的优先级设置 PRI_29： 中断号29的优先级设置 PRI_30： 中断号30的优先级设置 PRI_31： 中断号31的优先级设置	0x0

4.4.17 VIC_ISR (中断状态寄存器)

- Address = Base Address + 0x0C00, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
VECACTIVE	[8:0]	RW	指示当前CPU正在处理的中断向量号	0x0
VECPENDING	[29:12]	RW	指示当前等待的最高优先级中断向量号	0x0

4.4.18 VIC_IPTR (中断优先级阈值寄存器)

- Address = Base Address + 0x0C04, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
PRITHD	[7:0]	RW	中断抢占的优先级阈值设置 仅最高两位[7:6]有效，剩下[5:0]保留。	0x0
VECTHD	[16:8]	RW	优先级阈值对应的中断向量号。当VIC发现CPU从VECTHD所设置的中断服务程序退出时，会硬件清除中断优先级阈值有效位（THDEN）	0x0
THDEN	[31]	RW	中断优先级阈值有效位 0： 中断抢占不需要高于优先级阈值 1： 中断抢占需要优先级高于阈值	0x0

5 系统定时器 (CORET)

5.1 概述

系统定时器是 CPU一个内部模块，它主要用于计时。系统定时器提供了一个简单易用的 24 位循环递减的计数器，当系统计时器使能时，计数器开始工作。当计数器递减到 0 时，会向中断控制器发起中断请求。

5.1.1 特性

- 可编程计数时钟
  - 系统时钟8分频后，作为 TIMER 的计数时钟
  - CPU 的工作时钟
- 24位递减计数器，自动重载功能
- 支持溢出中断

5.2 功能描述

5.2.1 模块框图

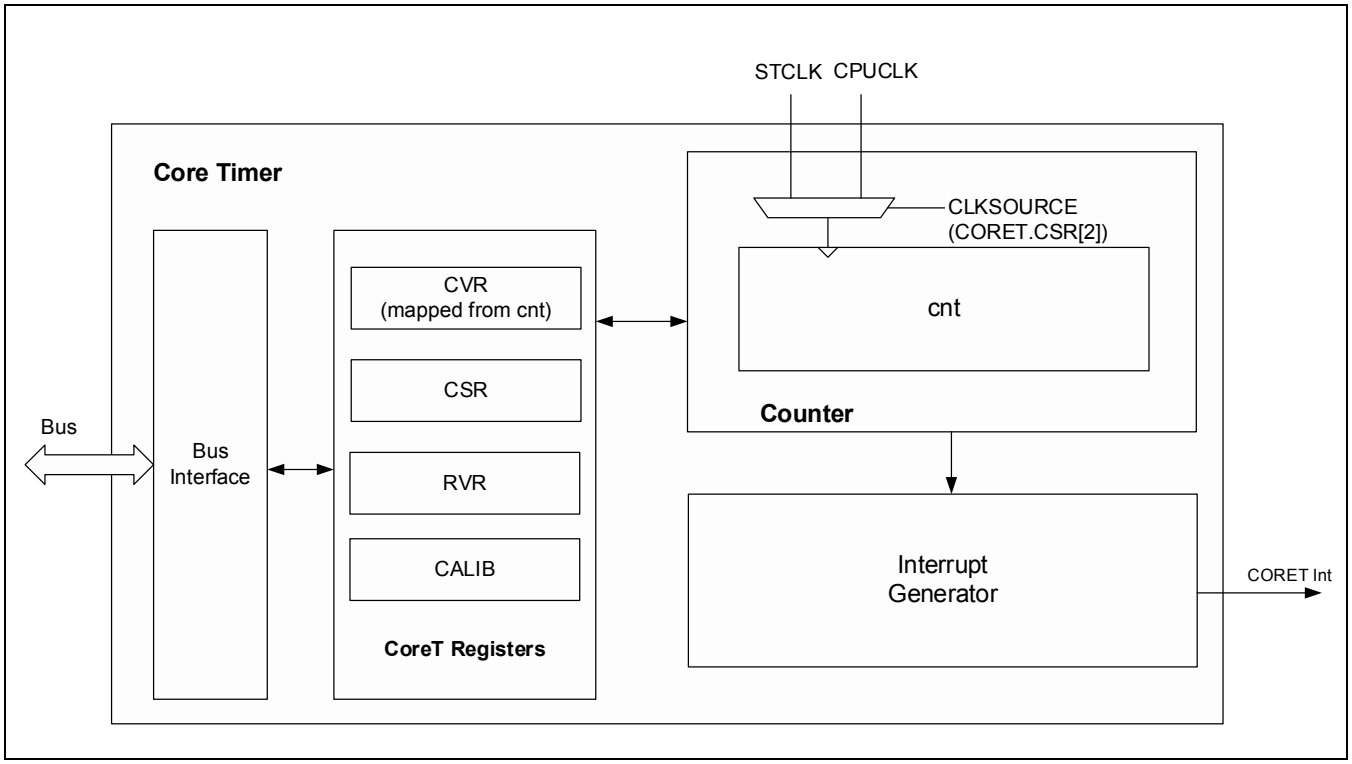


Figure 5-1 CORET模块框图

5.2.2 功能说明

5.2.2.1 定时器的时钟源

系统计时器内部由一个简单的 24 位循环递减的计数器构成，当系统计时器使能时，计数器开始工作。当计数器值递减到0 时，会向矢量中断控制器发起中断请求，申请获得处理器响应并处理系统计时器的事务。

CORET 定时器有两个可选时钟源：

- CPU 时钟 (CORECLK)
- 系统时钟的8分频 STCLK

时钟源的选择通过 CSR 寄存器的 bit2位 CLKSOURCE 来实现。当选择 CORET 的时钟为系统 STCLK 时，需要预先在 SYSCON 的 GCER 寄存器中使能该时钟。STCLK 时钟的使能/禁止和各种配置请参考 SYSCON 章节。

5.2.2.2 定时器的工作原理

CORET 定时器包含在 CPU Core 内部，产生的中断具有最高的优先级。CORET 定时器可以用作任何简单的计时，或者可以作为操作系统的 SYSTICK 定时器使用。当系统定时器使能 (CSR[0]=1) 时，计数器开始工作。计数

器从预设的值 (RVR 寄存器) 开始递减，当计数器递减到0时，如果使能了 CORET 中断 (CSR[1]=1) ，计数器会向中断控制器发起中断请求。

RELOAD 的正常取值范围在0x1 ~ 0x00FFFFFF 之间。RELOAD 值可以被赋值为0，但这不会产生任何中断，因为系统计数器中断以及 COUNTFLAG 位只有在计数器值由1变成0时才起作用。当需要产生一个周期为 N 个计数时钟周期的计时器，RELOAD 的值需要被赋值为 N-1。比如要在每100个计数时钟周期产生一个 CORET 的中断，需要将 RELOAD 赋值为99。

由于 RVR 和 CVR 两个寄存器不存在复位值，在系统计时器工作前，需要按照如下方式进行初始化操作：

1）向 CORET_RVR 寄存器中写入需要的 RELOAD 值。

2）向 CORET_CVR 写任意值，从而使得 RELOAD 被加载。

3）操作 CORET_CSR 寄存器，使能系统计数器。

CORET 计数溢出后（计数值由1变成0），CORET_CSR 寄存器中的 COUNTFLAG 位将被置位。通过读取该标志位对中断标志进行清除操作。

5.3 寄存器说明

5.3.1 寄存器表

Base Address: 0xE000_E000

Offset Address	Name	Description	R/W	Reset State
0x010	CORET_CSR	控制寄存器	R/W	0x00000000
0x014	CORET_RVR	回填值寄存器	R/W	0x00000000
0x018	CORET_CVR	当前值寄存器	R/W	0x00000000

5.3.1.1 CORET_CSR (控制寄存器)

- Address = Base Address + 0x0010

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

Name	Bit	Type	Description	Reset Value
COUNTFLAG	[16]	R	中断标志位，表示在上一次读此寄存器后计数器是否计数到 0： 0：计数器还没有计数到0 1：计数器已经计数到0 在计数器的值由1变到0时，COUNTFLAG会被置位。 读CSR寄存器以及任何写CVR寄存器会使COUNTFLAG清零。	0
CLKSOURCE	[2]	RW	系统定时器的时钟源选择： 0：时钟源为STCLK (SYSCLK/8) 1：时钟源为CORECLK	1
TICKINT	[1]	RW	中断使能： 0：禁止计数到0的中断 1：使能计数到0的中断 写CVR寄存器会使计数器清零，但不会导致系统计时器的中断状态位发生改变。	0
ENABLE	[0]	RW	定时器的使能控制： 0：禁止定时器 1：使能定时器	0

5.3.1.2 CORET_RVR (回填值寄存器)

- Address = Base Address + 0x0014

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

Name	Bit	Type	Description	Reset Value
RELOAD	[23:0]	RW	在计数器计数到0时，RELOAD值会被赋给CORET_CVR寄存器。 向CORET_RVR寄存器写0会使计数器在下一次循环时停止工作，此后计数器的值将一直保持为0。当使用外部参考时钟使能计数器后，必须等到计数器正常计数开始后(即CORET_CVR 变为非0值时)，才可以将 CORET_RVR置为0以让计数器在下一次循环时停止工作，否则计数器无法开始第一次计数。	0x0

5.3.1.3 CORET_CVR (当前值寄存器)

- Address = Base Address + 0x0008

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

Name	Bit	Type	Description	Reset Value
CURRENT	[23:0]	R/W	计数器的当前值。 写CORET_CVR寄存器会同时使此寄存器和COUNTFLAG状态位清零，并且会导致下一个时钟周期开始时，系统计时器取出寄存器CORET_RVR里的值并赋给CORET_CVR。 注意写CORET_CVR不会导致系统计时器的中断状态位发生改变。 读 CORET_CVR 会返回访问寄存器时计数器的值。	0x0

6 闪存控制器(IFC)

6.1 概述

本章节描述用来控制内部程序存储的闪存控制器。APT32F172 系列片上带有 60K/32K 字节的闪存(PROM)，支持通过 ISP 来更新闪存内容。有了 ISP (In System Programming)功能，用户可以在芯片被焊在 PCB 板上的情况下更新程序。芯片上电后，CPU 从 PROM 取指令并且执行。APT32F172 系列还支持额外的数据闪存(DROM)存储空间，让用户在掉电之前存储一些应用程序需要的数据。数据闪存(DROM)的大小可以通过 User Option 配置。

6.1.1 主要特性

- 程序闪存(PROM)大小: 60K/32K Bytes
- 数据闪存(DROM)大小: 4K/2K/1K Bytes
- 编程支持ISP模式和专用的工具模式
- 页大小: 1K Bytes
- 可擦除单元: 页
- 可靠性: PROM和DROM都为200,000次
- 可自定义的选项(称为User Option)支持iWDT使能和禁止，配置复位管脚
- 支持各种保护：调试接口保护，硬件保护和读保护

6.2 功能描述

6.2.1 模块框图

闪存控制器由 AHB 和 APB 接口模块，ISP 控制逻辑和时序控制逻辑组成。模块框图如下图所示：

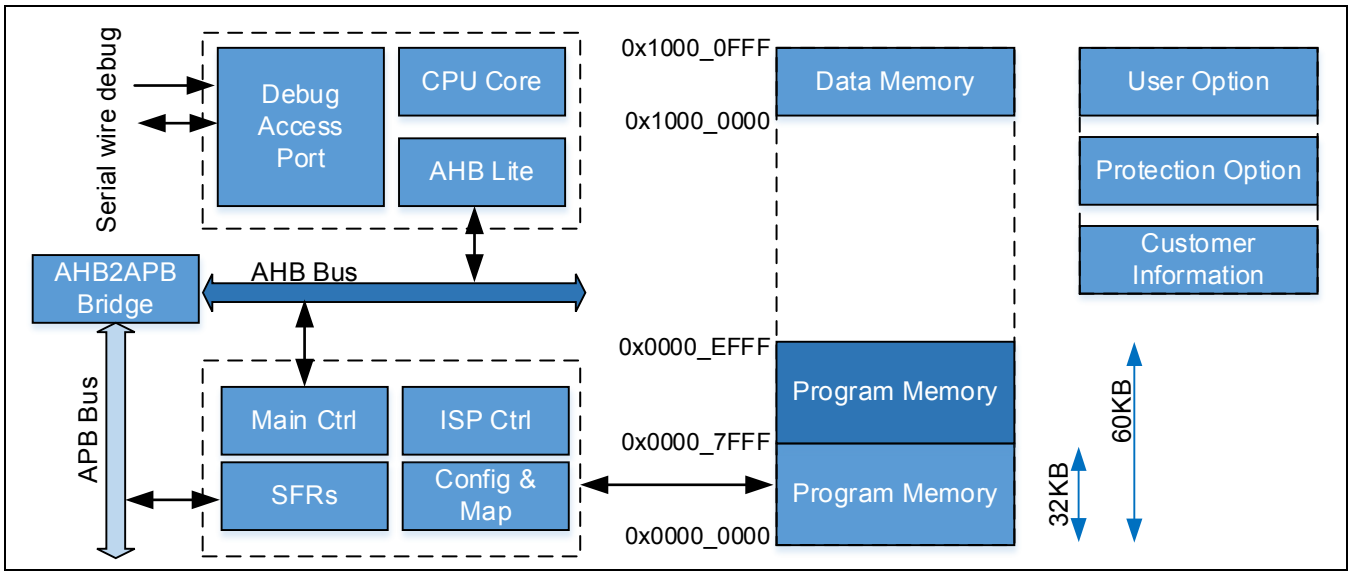


Figure 6-1 IFC模块框图

6.2.2 模块结构

APT32F172 系列闪存由程序存储单元(PROM)，数据存储单元(DROM)，用户配置单元(User Option)，保护选项和客户信息区域构成。PROM 有 64/63/61/60/32 个页空间，每页有 1K 字节。最小的擦除单元为页空间，用户可以每次指定一个页空间的单位地址进行页擦除操作，指定一个字(Word)的单位地址进行写操作。

区域	页名称	大小	起始地址	结束地址
PROM Within 32KB	Page 0	1KB	0x0000_0000	0x0000_03FF
	Page 1	1KB	0x0000_0400	0x0000_07FF
	Page 2	1KB	0x0000_0800	0x0000_0BFF
	Page 3	1KB	0x0000_0C00	0x0000_0FFF
	Page 4	1KB	0x0000_1000	0x0000_13FF
	Page 5	1KB	0x0000_1400	0x0000_17FF
	Page 6	1KB	0x0000_1800	0x0000_1BFF
	Page 7	1KB	0x0000_1C00	0x0000_1FFF
	:	:	:	:

	Page 30	1KB	0x0000_7800	0x0000_7BFF
	Page 31	1KB	0x0000_7C00	0x0000_7FFF
PROM Within 60KB	Page 32	1KB	0x0000_8000	0x0000_83FF
	Page 33	1KB	0x0000_8400	0x0000_87FF
	Page 34	1KB	0x0000_8800	0x0000_8BFF
	Page 35	1KB	0x0000_8C00	0x0000_8FFF
	:	:	:	:
	Page 58	1KB	0x0000_E800	0x0000_EBFF
	Page 59	1KB	0x0000_EC00	0x0000_EFFF
62KB	Page 60	1KB	0x0000_F000	0x0000_F3FF
	Page 61	1KB	0x0000_F400	0x0000_F7FF
63KB	Page 62	1KB	0x0000_F800	0x0000_FBFF
64KB	Page 63	1KB	0x0000_FC00	0x0000_FFFF
DROM	Page 0	1KB	0x1000_0000	0x1000_03FF
	Page 1	1KB	0x1000_0400	0x1000_07FF
	Page 2	1KB	0x1000_0800	0x1000_0BFF
	Page 3	1KB	0x1000_0C00	0x1000_0FFF

Table 6-1 闪存地址映射

闪存结构如下图所示：

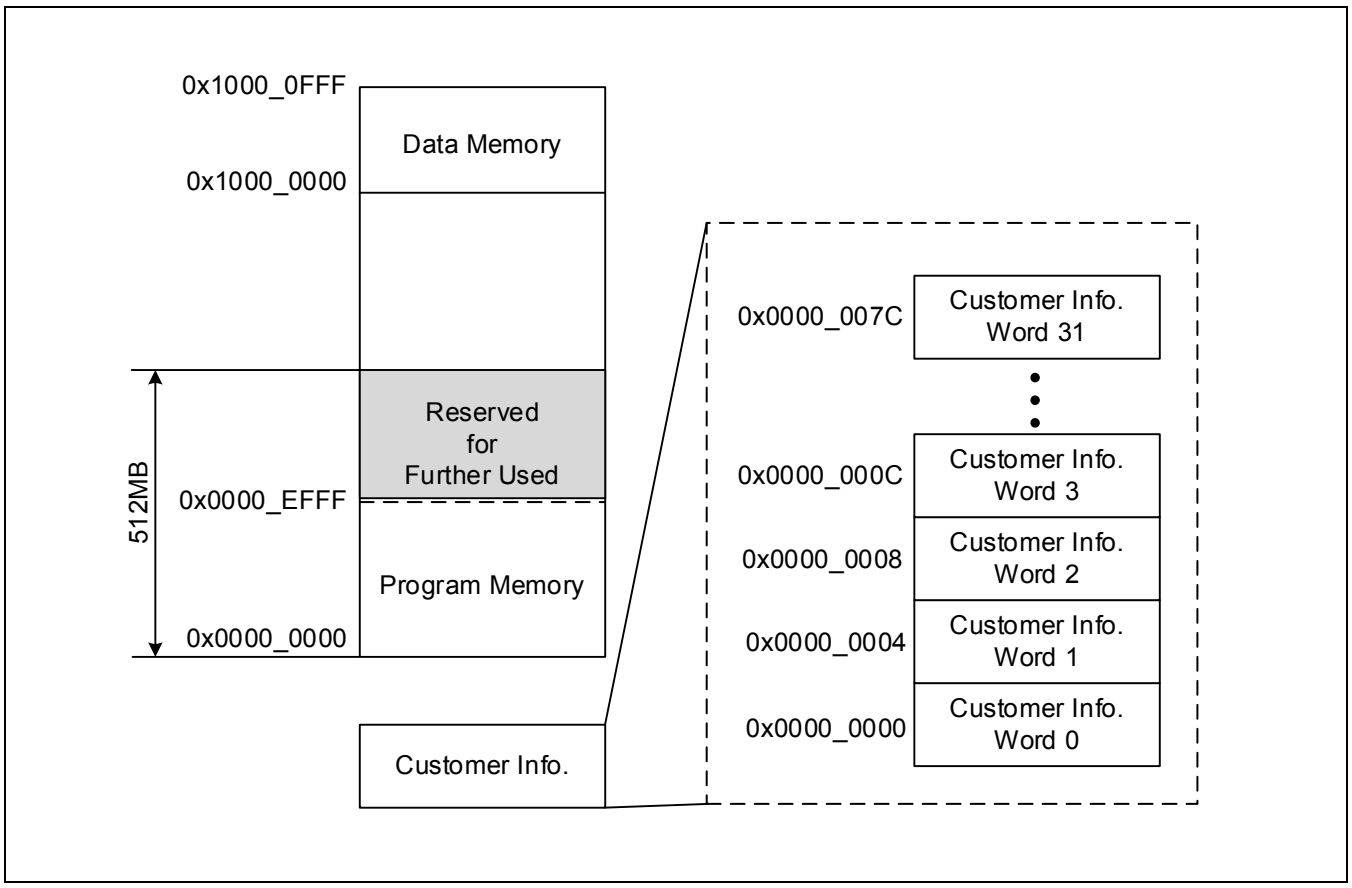


Figure 6-2 闪存地址空间结构

6.2.3 数据闪存

APT32F172 系列支持数据闪存，给用户存储普通数据。数据闪存可以通过 ISP 编程进行读写。擦除的最小单位为页。当需要改变某个字节时，该页中所有 256 个字节都需要提前复制到另一页里或者 SRAM 里暂存，或者使用特殊的软件算法在多页中轮循，模拟 EEPROM 的操作。尽管 PROM 也支持 ISP 功能，但为了数据的安全性和程序代码的完成性，我们强烈建议使用数据闪存空间来存放应用所需要存储的信息，而不是使用程序闪存。在进行全芯片擦除操作时，数据闪存和程序闪存一样都会被擦除掉。

在 60KB 闪存的产品里，用户可以选择数据闪存的大小，支持的选项为 4KB/2KB/1KB 和没有数据闪存(0KB)。最大位 4KB。如果不需要 4KB 数据闪存，可以选择 2KB/1KB/0KB，这样剩下的数据闪存空间则可以被用来当做程序闪存空间。参考 User Option 选项里的 DSIZE 位。

6.2.4 自定义选项 (User Option，保护选项，客户信息区域，工厂信息区域)

闪存中有一些可以自定义的空间，用来设置 User Option，使能各种保护功能和给客户存储一些自定义的信息。除工厂信息区域，所有自定义空间的内容在 CHIP ERASE 时，都会被擦除。

自定义空间的内容在芯片上电后会被自动读取到相应的寄存器中。即使芯片以及被焊在 PCB 上，用户仍然可以根据需要，使用 ISP 功能或者专用的烧录工具来设置这些选项。

User Option 用来配置各种不同应用所需的功能，这个选项是地址为 0x0000_0100 的一个字(4 个字节)。用户如果需要配置 User Option，只需要保证在烧写进闪存的程序代码中，地址 0x0000_0100 的值为所需的 User Option 值即可。（该功能需要配合专用的烧录器使用）

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

名称	位	描述
IWDT	[31:16]	独立看门狗电路使能/禁止 **IWDT[15:0]** / **功能** 0x5555 / 禁止 Other / 使能
DSIZE	[9:8]	数据闪存大小控制 **DSIZE[1:0]** / **数据闪存大小** / **程序闪存大小** 11 / 4KB / 60KB 10 / 2KB / 62KB 01 / 1KB / 63KB 00 / 0KB / 64KB
EXTRST	[7:4]	外部复位管脚功能 **EXTRST[3:0]** / **功能** 0x5 / 在PA0.0上使能外部复位功能 其它值 / 禁用PA0.0的外部复位功能，当作IO使用
HSFREQ	[3:2]	HSOSC 输出频率选择 **HSFREQ[1:0]** / **功能** 0x2 / HSOSC 96MHz 其它值 / HSOSC 48MHz

<table>
<tr><td>ISFREQ</td><td>[1:0]</td><td>ISOSC 输出频率选择
<table><tr><th>ISFREQ[1:0]</th><th>功能</th></tr><tr><td>0x2</td><td>ISOSC 500KHz</td></tr><tr><td>其它值</td><td>ISOSC 3MHz</td></tr></table></td></tr>
</table>

保护选项是为了保护代码的安全。闪存控制器支持三种不同的保护机制，可以通过操作相应的保护位来使能。

- 硬件(Hard-lock)保护

  如果使能了硬件保护，用户不能在已经设置了保护的PROM区域中执行页擦除和写操作。用户可以通过软件的HDPEN或者IFERASE功能锁住或者解锁PROM。使用烧写工具时，在执行了全芯片擦除后，硬件保护会被解锁。DROM的ISP操作不会受硬件保护锁的影响。硬件保护锁可以增加闪存的可靠性和抗干扰能力，避免PROM闪存数据由于代码错误造成丢失或改变。

  硬件保护功能可以保护整个或者部分PROM，软件中可以在用户特权模式下通过软件使能，或者使用外部的烧录工具使能。具体参考外部烧录工具的手册或者IFC指令寄存器(IFC_CMR)，可用的选择如下所示。

  IFC_FULL: 保护闪存全部PROM区域的内容

  IFC_4K: 保护从0x0地址开始的4K字节 (0x0 ~ 0xFFF)

  IFC_2K: 保护从0x0地址开始的2K字节 (0x0 ~ 0x7FF)

  IFC_1K: 保护从0x0地址开始的1K字节 (0x0 ~ 0x3FF)

  这个选项可以在固件更新功能中使用。例如，可以将用户启动代码(booting code)放在0x0 ~ 0x7FF区域内，然后选择IFC_2K选项使能硬件保护锁。当固件需要更新时，启动代码可以擦除所有没有被保护的PROM区域并且将新的固件烧写进去，同时启动代码本身不会被擦除，保持不变。

- 读保护

  大多数用户都不希望闪存中的程序代码被其他人读出来。所以为了用户的代码安全，读保护功能可以禁止外部烧录工具读取闪存中的数据。这个功能被使能后，只有自定义选项区域和客户自定义信息区域可以被正常读取，其它所有闪存区域读出来都是0.

客户信息区域由 32 个字(128 字节)组成，可以根据客户所需存储应用 ID 或者序列号等等。这个区域不支持通过 ISP 编程，而且跟其它自定义选项一样在 CHIP ERASE 时会被擦除掉。这个区域必须通过外部烧录工具进行烧写。

工厂信息区域跟客户信息区域一样，也是由 32 个字(128 字节)组成，不同的是，这个区域客户不能自己通过 ISP 功能或者烧录工具进行烧写，只能委托工厂在芯片出厂的时候一次性写入。工厂在写入后，该区域存储的内容不会被 ISP 或者烧录工具擦除。

所有自定义选项的闪存单元都不能直接通过总线被 CPU 读取出来，而是在 SYSCON 模块中有相应的镜像寄存器供查询，具体请参考 SYSCON 中 OPT0 寄存器。客户信息区域中只有前 2 个字(8 字节)能通过 SYSCON 中 CINF0~CINF1 寄存器读取，剩下的字节都只能通过外部烧录工具读取。工厂信息区域中也只有前 2 个字(8 字节)能通过 SYSCON 中 FINF0~FINF1 寄存器读取，剩下的字节都只能通过外部烧录工具读取。

6.2.5 读操作

闪存控制器支持最大 25MHz 系统频率下的 0-wait 读取。当频率超过 25MHz 时，读取闪存时需要增加额外的等待周期。APT32F172 系列只支持最大 40MHz 的系统频率，所以当系统频率超过 25MHz 时，0-wait 需要设置为 1。

6.2.6 烧写方法

用户可以通过下面几种方法将数据或者代码(烧)写进闪存

- 用户编程模式 (AHB接口)
- SWD接口
- 烧录工具 (专用串行接口)

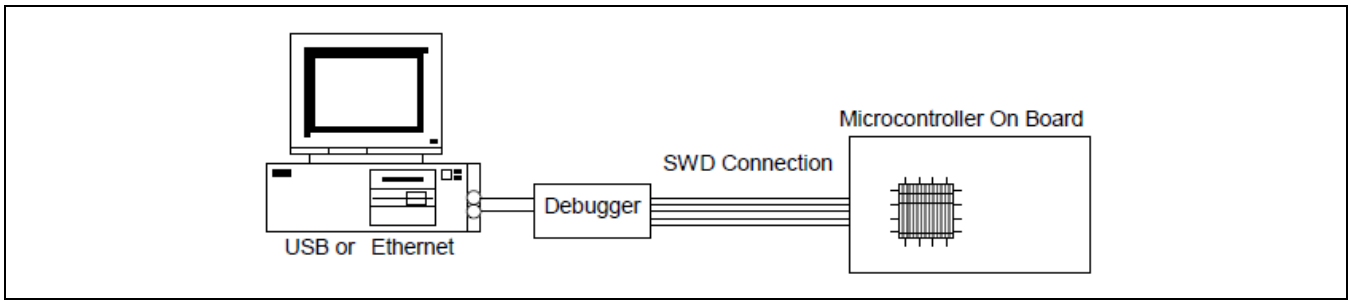


Figure 6-3 通过调试接口的烧写

通过程序代码或者 SWD 接口来擦除和烧写闪存的方式，一般通常被叫做 ISP(In System Program)方式。它支持当芯片在工作时，或者芯片已经被焊在 PCB 版上时，用户也能够修改闪存的内容。如果 SWD 接口被调试保护功能禁止，那么 SWD 烧写的方式就不再可用，这时候最好的办法就是通过硬件烧录工具来烧录了。

APT 硬件烧录模式跟 ISP 模式类似，只用了两根线作为通讯信号，能减少量产阶段产品所需的上市时间。烧写所需的信号如下表所示。

但是，如果在交付给终端客户后仍然有固件更新的需求，那么建议在代码中加入自定义的 ISP 功能用于固件更新。

信号	管脚名称	I/O	描述
VDD	VDD	P	芯片电源 **(烧录时建议在电源和地之间连接0.1uF电容)**
VSS	VSS	G	芯片地
SDAT	PA0.4	I/O	串行双向数据管脚
SCLK	PA0.3	I	串行时钟输入管脚，内部默认上拉
RESET	PA0.0	I	复位

Table 6-2 闪存烧写信号

6.2.7 ISP功能

闪存 ISP 功能通过 IFC 中的一些控制寄存器来实现。ISP 操作中会检查一些错误情况，如果遇到某些特定的错误，那么 ISP 操作会失败。

6.2.7.1 写闪存操作

写(烧录)操作会在 FM_ADDR 寄存器所包含的地址中写入一个字(4 个字节)。也就是说一次写操作会写入 32 位 4 个字节，所以 FM_ADDR 中的最低 2 位会被忽略并且自动进行字对齐。

为了成功写入数据，目标地址的闪存必须先进行页擦除或者全芯片擦除。在 ISP 操作前，用户必须将秘钥 0x5A5A_5A5A 写入 IFC_KEY 寄存器以禁止闪存模块的擦除/烧写保护。之后，用户需要将烧写的地址写入 IFC_FM_ADDR 寄存器，并将 IFC_CMR 里的指令 CMD[3:0]写为 0x1(写操作)，最后将 IFC_CR 的 START 位置 1 启动该操作的执行。在 ISP 操作完成后，IFC_RISR 里的 END 位会置 1。用户同时也可以查询 IFC_CR 里的 START 位来判断 ISP 操作是否完成。

示例：

```
CSP_IFC_SET_KR(IFC, USER_KEY);          // Write Key
CSP_IFC_SET_CMR(IFC, PROGRAM);          // Program
CSP_IFC_SET_AR(IFC, 0x00007C00);        // Program address
CSP_IFC_SET_DR(IFC, 0x87654321);        // Program data
CSP_IFC_SET_CR(IFC, START);             // Start Program
while ( CSP_IFC_GET_CR(IFC) != 0x0 );   // Wait for operation done
```

6.2.7.2 页擦除操作

每页闪存中有 1K 字节。页擦除操作会擦除 IFC_FM_ADDR 中地址所在的那一页闪存。在 ISP 操作前，用户必须将秘钥 0x5A5A_5A5A 写入 IFC_KEY 寄存器以禁止闪存模块的擦除/烧写保护。之后，用户需要将烧写的地址写入 IFC_FM_ADDR 寄存器，并将 IFC_CMR 里的指令 CMD[3:0]写为 0x2(页擦除操作)，最后将 IFC_CR 的 START 位置 1 启动该操作的执行。在 ISP 操作完成后，IFC_RISR 里的 END 位会置 1。用户同时也可以查询 IFC_CR 里的 START 位来判断 ISP 操作是否完成。

示例：

```
CSP_IFC_SET_KR(IFC, USER_KEY);          // Write Key
CSP_IFC_SET_CMR(IFC, PAGE_ERASE);       // Page Erase
CSP_IFC_SET_AR(IFC, 0x00007C00);        // Program address
CSP_IFC_SET_CR(IFC, START);             // Start Page Erase
while ( CSP_IFC_GET_CR(IFC) != 0x0 );   // Wait for operation done
```

6.2.7.3 片擦除操作

片擦除操作会擦除整个闪存的程序存储和数据存储区域，但不会擦除自定义选项的区域。片擦除操作只能在用户特权模式下才能执行。在片擦除中，不需要指定 ISP 操作相关的地址和数据寄存器。在 ISP 操作前，用户必须将秘钥 0x5A5A_5A5A 写入 IFC_KEY 寄存器以禁止闪存模块的擦除/烧写保护。之后，将 IFC_CMR 里的指令 CMD[3:0]写为 0x3(片擦除操作)，HMODE[1:0]写为 0x1(用户特权模式)，最后将 IFC_CR 的 START 位置 1 启动该操作的执行。在 ISP 操作完成后，IFC_RISR 里的 END 位会置 1。用户同时也可以查询 IFC_CR 里的 START 位来判断 ISP 操作是否完成。

示例：

```
CSP_IFC_SET_KR(IFC, USER_KEY);              // Write Key
CSP_IFC_SET_CMR(IFC, HIDM1|CHIP_ERASE);     // Chip Erase
CSP_IFC_SET_CR(IFC, START);                 // Start Erase
while ( CSP_IFC_GET_CR(IFC) != 0x0 );       // Wait for operation done
```

6.2.7.4 烧写自定义选项操作

共有 4 种自定义的选项支持通过 ISP 操作(HDP, RDP, DBP, User Option)。在 ISP 操作前，用户必须将秘钥 0x5A5A_5A5A 写入 IFC_KEY 寄存器以禁止闪存模块的擦除/烧写保护。之后，将 IFC_CMR 里的指令 CMD[3:0]写为 0x09/0x0A/0x0B/0x0C/0x0D/0x0E/0x0F，HMODE[1:0]写为 0x1(用户特权模式)，最后将 IFC_CR 的 START 位

置 1 启动该操作的执行。在 ISP 操作完成后，IFC_RISR 里的 END 位会置 1。用户同时也可以查询 IFC_CR 里的 START 位来判断 ISP 操作是否完成。在 HDP/RDP/DBP 操作中，不需要设置 ISP 的地址和数据寄存器。在写 User Option 的操作中，不需要设置地址寄存器，但需要将 User Option 的值写入数据寄存器。

示例：

```
CSP_IFC_SET_KR(IFC, USER_KEY);                  // Write Key
CSP_IFC_SET_CMR(IFC, HIDM1|USER_OPTION);        // Write User Option
CSP_IFC_SET_DR(IFC, 0x5555FF5F);                // User Option data,
                                                // Disable WDT, enable EXRST
CSP_IFC_SET_CR(IFC, START);                     // Start Program
while ( CSP_IFC_GET_CR(IFC) != 0x0 );           // Wait for operation done
```

6.2.7.5 擦除自定义选项区域

这个自定义选项擦除操作会擦除所有的 User Option，保护选项和客户信息区域。在 ISP 操作前，用户必须将秘钥 0x5A5A_5A5A 写入 IFC_KEY 寄存器以禁止闪存模块的擦除/烧写保护。之后，将 IFC_CMR 里的指令 CMD[3:0]写为 0x4(自定义选项擦除操作)，HMODE[1:0]写为 0x1(用户特权模式)，最后将 IFC_CR 的 START 位置 1 启动该操作的执行。在 ISP 操作完成后，IFC_RISR 里的 END 位会置 1,。用户同时也可以查询 IFC_CR 里的 START 位来判断 ISP 操作是否完成。

示例：

```
CSP_IFC_SET_KR(IFC, USER_KEY);                  // Write Key
CSP_IFC_SET_CMR(IFC, HIDM1|IF0_ERASE);          // IF0 Erase
CSP_IFC_SET_CR(IFC, START);                     // Start Erase
while ( CSP_IFC_GET_CR(IFC) != 0x0 );           // Wait for operation done
```

6.2.8 闪存控制器的中断

闪存操作有 5 个中断源，如下所示。

中断	描述
END	指令执行完成中断
PROT_ERR	保护错误；当硬件保护锁使能，仍然进行写操作或擦除操作
UDEF_ERR	未定义指令错误；CMD中定义的操作指令非法或者不允许在当前模式中执行
ADDR_ERR	地址错误；FM_ADDR中定义的地址超出了最大地址范围 (注意)
OVW_ERR	非法操作错误；当ISP操作正在进行时，尝试修改CMD，FM_ADDR，FM_DR，START寄存器

Table 6-3 中断源描述

当中断发生时，RISR 寄存器中的相应位会被置 1。RISR 的置 1 并不受 ICR 设置的影响。如果 ICR 中相应的中断位被置 1，而且该中断发生了(RISR 相应位置 1)，那么该中断会被送至 CPU 处理，进入中断子程序。用户可以在中断子程序中用 ICLR 寄存器清除相应的中断状态位。

注意：ADDR_ERR 只提供在 RISR 中的查询功能，不提供 CPU 的中断功能。

6.2.9 闪存控制流程图

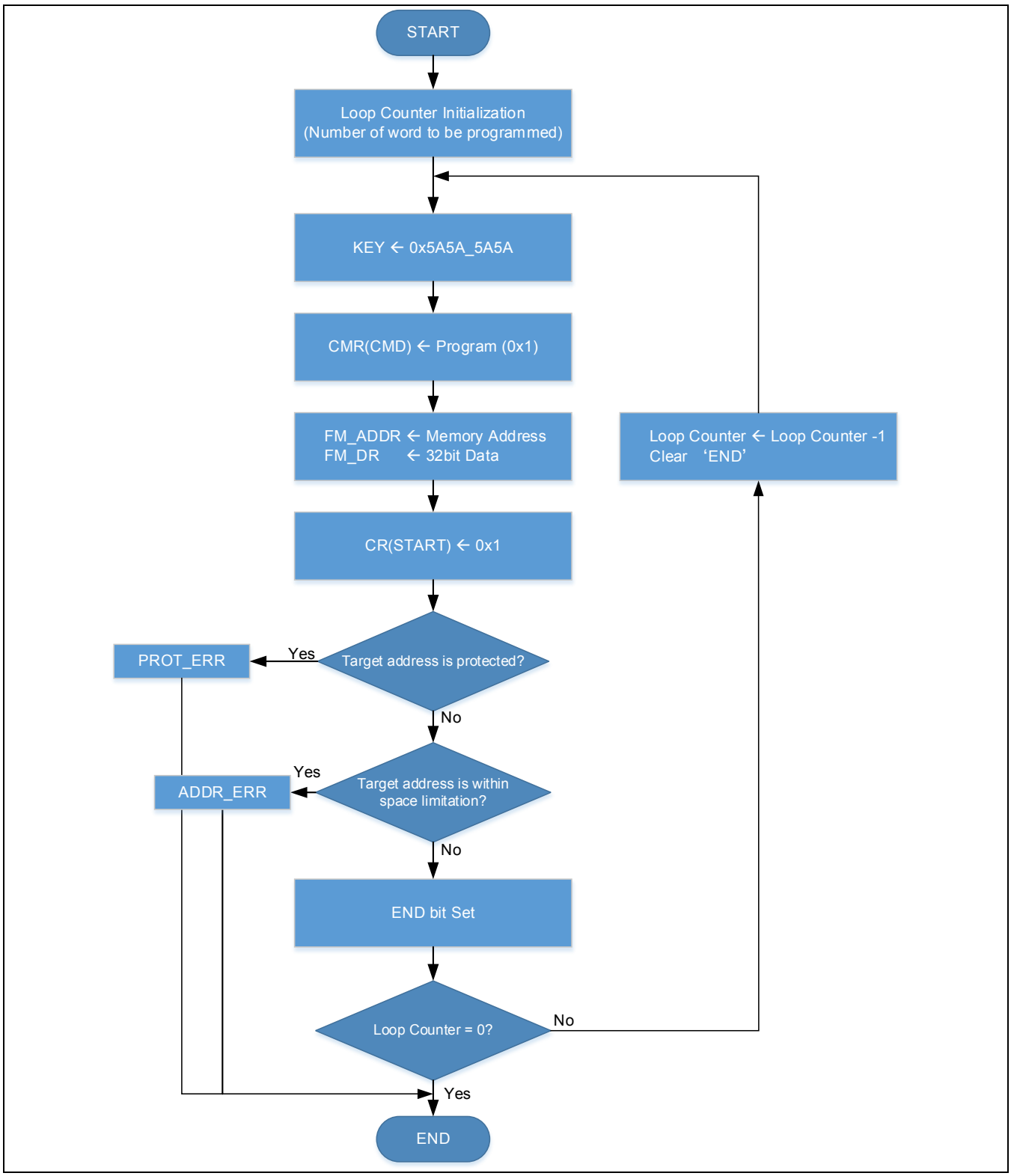


Figure 6-4 写操作流程图

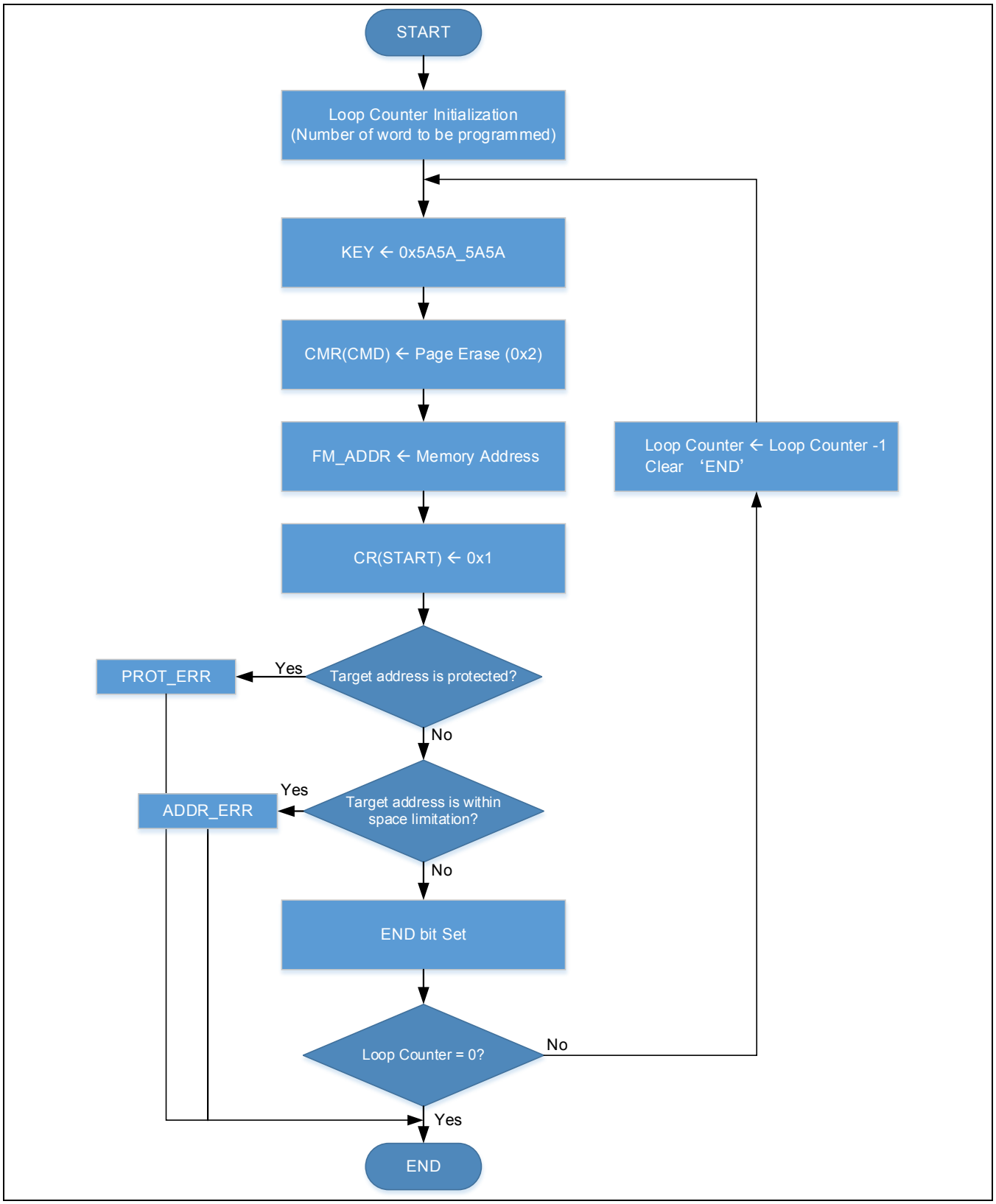


Figure 6-5 页擦除操作流程图

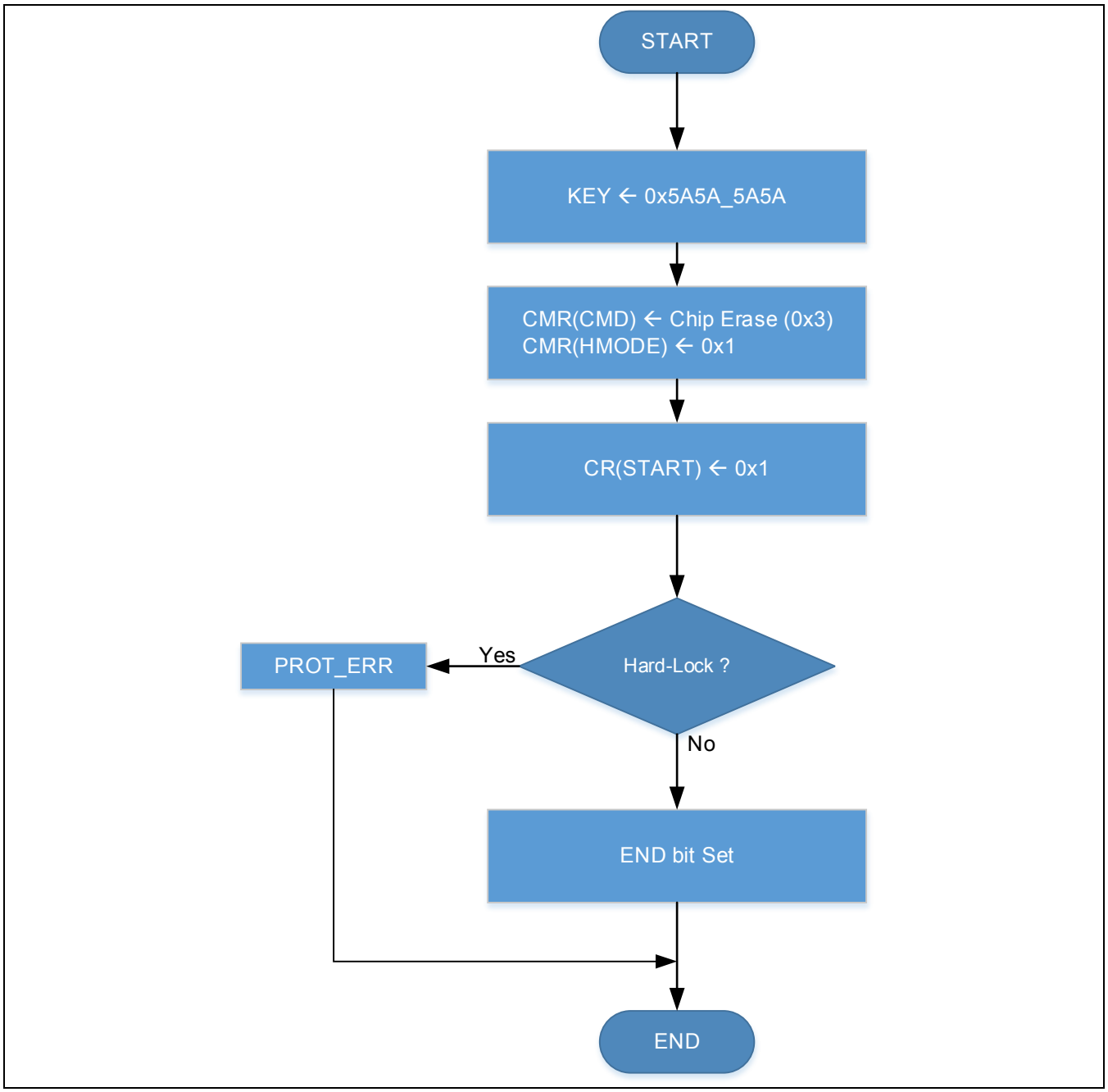


Figure 6-6 片擦除操作流程图

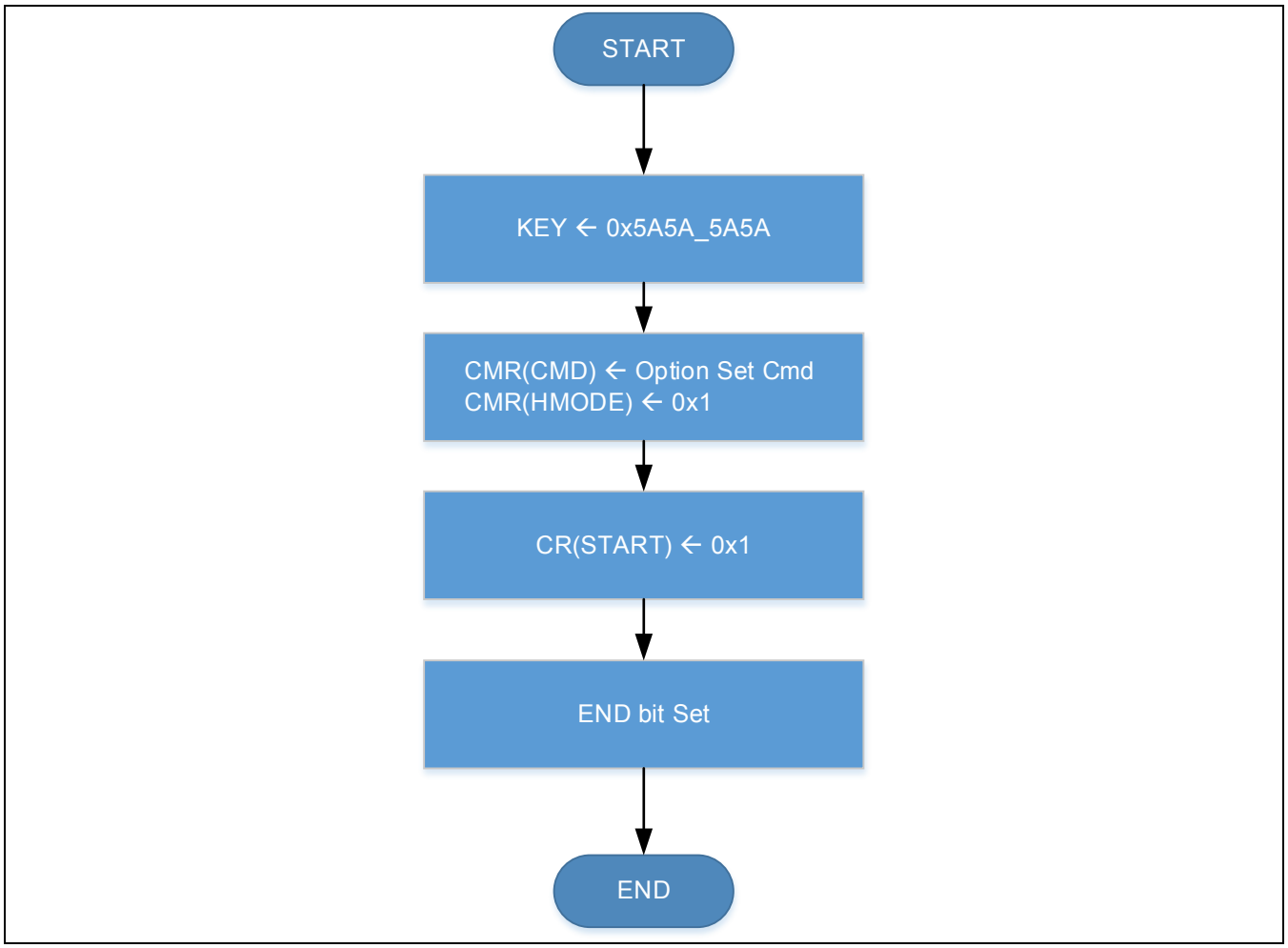


Figure 6-7　自定义选项写操作流程图

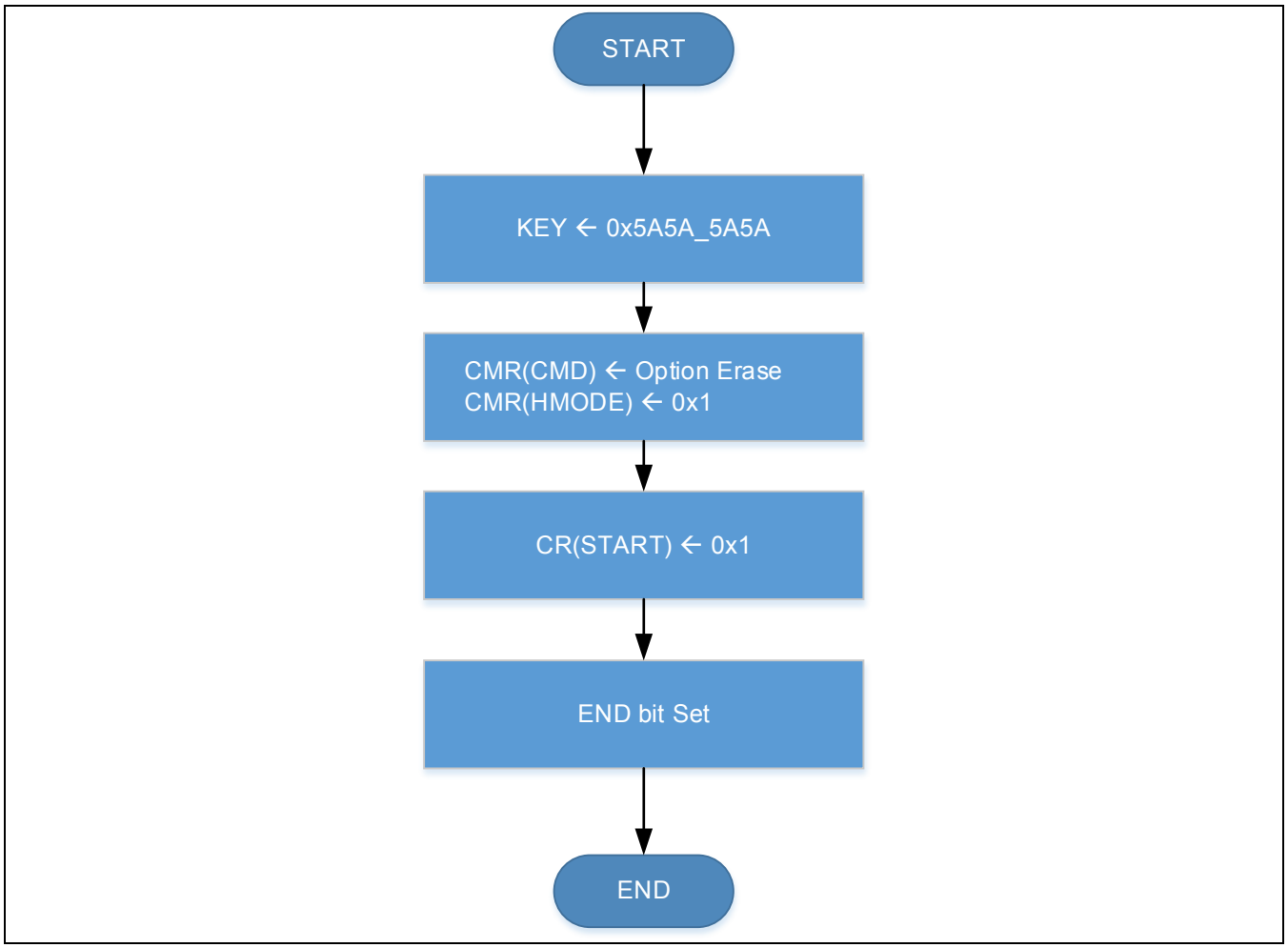


Figure 6-8 自定义选项擦除操作流程图

6.3 寄存器说明

6.3.1 寄存器表

- Base Address: 0x4001_0000

Register	Offset	Description	Reset Value
IFC_IDR	0x00	闪存控制器 ID 寄存器	
IFC_CEDR	0x04	时钟使能/禁止寄存器	0x0000_0000
IFC_SRR	0x08	软件复位寄存器	0x0000_0000
IFC_CMR	0x0C	指令寄存器	0x0000_0000
IFC_CR	0x10	控制寄存器	0x0000_0000
IFC_MR	0x14	工作模式寄存器	0x0000_0000
IFC_FM_ADDR	0x18	ISP 地址寄存器	0x0000_0000
IFC_FM_DR	0x1C	ISP 数据寄存器	0x0000_0000
IFC_KR	0x20	ISP 秘钥寄存器	0x0000_0000
IFC_ICR	0x24	中断控制寄存器	0x0000_0000
IFC_RISR	0x28	中断原始状态寄存器	0x0000_0000
IFC_MISR	0x2C	中断状态寄存器	0x0000_0000
IFC_ICLR	0x30	中断状态清楚寄存器	0x0000_0000

6.3.2 IFC_IDR（ID寄存器）

- Address = Base Address + 0x0000, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
IDCODE	[31:8]	R	**ID** 代码

6.3.3 IFC_CEDR (时钟使能/禁止寄存器)

- Address = Base Address + 0x0004, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
CLKEN	[0]	RW	**时钟使能/禁止寄存器** 0: 禁止闪存控制器的时钟 1: 使能闪存控制器的时钟 软件复位 (IFC_SRR)不会影响 CLKEN 的状态

6.3.4 IFC_SRR (软件复位寄存器)

- Address = Base Address + 0x0008, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
SWRST	[0]	RW	**软件复位** 0: 无效 1: 执行软件复位操作 除 CEDR 外的所有寄存器都会恢复初始值

6.3.5 IFC_CMR (指令寄存器)

- Address = Base Address + 0x000C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
CMD	[3:0]	RW	**写/擦除指令寄存器** **CMD[3:0] / 指令** 0x1 写操作 0x2 页擦除 0x3 片擦除 0x4 自定义选项擦除 0x9 硬件保护1K(HDP_1K)使能 0xA 硬件保护2K(HDP_2K)使能 0xB 硬件保护4K(HDP_4K)使能 0xC 全范围硬件保护(HDP_FULL)使能 0xD RDP读保护使能 0xE DBP调试保护使能 0xF 写User Option操作 注意： 1. 当执行 ISP 操作时，禁止读取闪存内容 2. 当操作完成后，IFC_CMR 寄存器会自动清零 3. 如果 IFC_KR 的秘钥值不对，那么指令不会被执行
HMODE	[9:8]	RW	**操作模式寄存器**

			00: 普通模式 01: 用户特权模式 10: 保留 11: 保留 在普通模式下，只有页擦除和写操作有效。其它指令都必须在用户特权模式下执行。

6.3.6 IFC_CR (控制寄存器)

- Address = Base Address + 0x0010, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
START	[0]	RW	**操作启动位** 0: 无效 1: 根据 CMR 设置的值开始执行指令 注意： 1. 当操作完成后，START 位会被自动清零 2. 指令的执行过程中，禁止对这位再进行写操作

6.3.7 IFC_MR (工作模式寄存器)

- Address = Base Address + 0x0014, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
WAIT	[2:0]	RW	**闪存读等待周期** 0: 闪存读取中等待 0 个周期 n: 闪存读取中等待 n 个周期 注意： 1. 工作频率在 0~25MHz 时，使用 0 等待周期 2. 工作频率在 25~48MHz 时，使用 1 个等待周期

6.3.8 IFC_FM_ADDR (ISP地址寄存器)

- Address = Base Address + 0x0018, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
FM_ADDR	[31:0]	RW	**ISP 地址寄存器** 写操作和页擦除操作中的目标闪存地址 注意： 1. 操作完成后，这个寄存器会自动清零。 2. 除了写操作和页擦除操作，其它指令执行时都不需要设置该寄存器

6.3.9 IFC_FM_DR (ISP数据寄存器)

- Address = Base Address + 0x001C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
FM_DATA	[31:0]	RW	**ISP 数据寄存器** 当执行写操作时，需要写进闪存的数据

6.3.10 IFC_KR (ISP秘钥寄存器)

- Address = Base Address + 0x0020, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	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																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Name	Bit	Type	Description
KEY	[31:0]	W	**ISP 安全秘钥寄存器** 秘钥寄存器用来保证 ISP 操作的安全，必须将该寄存器写 0x5A5A_5A5A，所有闪存控制器的指令才会被执行。该寄存器在 ISP 操作完成后会被自动清零。

6.3.11 IFC_ICR (中断控制寄存器)

- Address = Base Address + 0x0024, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
END	[0]	RW	**指令执行完成中断使能/禁止** ISP 操作完成 0: 禁止中断 1: 使能中断
PROT_ERR	[12]	RW	**保护错误中断使能/禁止** 当硬件保护锁使能，仍然进行写操作或擦除操作 0: 禁止中断 1: 使能中断
UDEF_ERR	[13]	RW	**未定义指令错误中断使能/禁止** CMD 中定义的操作指令非法或者不允许在当前模式中执行 0: 禁止中断 1: 使能中断
OVW_ERR	[15]	RW	**非法操作错误中断使能/禁止** 当 ISP 操作正在进行时，尝试修改 CMD，FM_ADDR，FM_DR，START 寄存器

			0: 禁止中断 1: 使能中断

6.3.12 IFC_RISR (中断原始状态寄存器)

- Address = Base Address + 0x0028, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
END	[0]	R	**指令执行完成中断的原始状态** 0: 该状态没有发生 1: 该状态发生
PROT_ERR	[12]	R	**保护错误中断的原始状态** 0: 该状态没有发生 1: 该状态发生
UDEF_ERR	[13]	R	**未定义指令错误中断的原始状态** 0: 该状态没有发生 1: 该状态发生
ADDR_ERR	[14]	R	**地址错误中断的原始状态** 0: 该状态没有发生 1: 该状态发生
OVW_ERR	[15]	R	**非法操作错误中断的原始状态**

			0: 该状态没有发生 1: 该状态发生

6.3.13 IFC_MISR（中断状态寄存器）

- Address = Base Address + 0x002C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
END	[0]	R	**指令执行完成中断的状态** 0: 该中断没有发生 1: 该中断发生
PROT_ERR	[12]	R	**保护错误中断的状态** 0: 该中断没有发生 1: 该中断发生
UDEF_ERR	[13]	R	**未定义指令错误中断的状态** 0: 该中断没有发生 1: 该中断发生
OVW_ERR	[15]	R	**非法操作错误中断的状态** 0: 该中断没有发生 1: 该中断发生

6.3.14 IFC_ICLR（中断状态清除寄存器）

- Address = Base Address + 0x0030, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																OVW_ERR	ADDR_ERR	UDEF_ERR	PROT_ERR	RSVD											END
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	R	R	R	R	R	R	R	R	R	R	R	W

Name	Bit	Type	Description
END	[0]	W	**指令执行完成中断状态清除** 0: 无效 1: 清除中断
PROT_ERR	[12]	W	**保护错误中断状态清除** 0: 无效 1: 清除中断
UDEF_ERR	[13]	W	**未定义指令错误中断状态清除** 0: 无效 1: 清除中断
ADDR_ERR	[14]	W	**地址错误中断状态清除** 0: 无效 1: 清除中断
OVW_ERR	[15]	W	**非法操作错误中断状态清除**

			0: 无效 1: 清除中断

7 系统控制器（SYSCON）

7.1 概述

系统控制器用于管理和配置整个芯片的时钟和系统相关的工作状态，包括不同工作模式下的具体时钟配置，功耗优化控制，系统异常处理（RESET 源历史记录，外部晶振失效监测，低电压报警和复位，看门狗设置，以及外部中断）。

系统运行的时钟可以从外部时钟（EMOSC），内部主时钟（IMOSC）和内部副时钟（ISOSC）三个时钟源中选择任意一个作为系统时钟。

通过系统控制器还可以对系统的配置状态（看门狗使能状态，调试口使能状态，Flash 硬件写保护状态，Flash 读保护，用户信息数据）进行查询，并可以通过系统控制器实现对系统内部时钟频率的微调。

7.1.1 特性

- 系统时钟频率管理
  - 可编程系统时钟（SYSCLK）和外设时钟（PCLK）
  - 外部时钟失效监测（Clock Fail Monitor）
  - 可选择的系统内部时钟源输出（COP）
- 可配置的系统运行时钟源
  - EMCLK：外部主时钟，通过外部直接加载或者晶振工作
  - IMCLK：内部主时钟，内部20/40MHz 的 RC 振荡器
  - ISCLK：内部副时钟，内部500KHz 或者3MHz 可选的 RC 振荡器
- 分立的基础时钟控制区域
  - SYSCLK：支持系统工作的时钟（例如，CPU 时钟，AHB 总线时钟）
  - PCLK：外设工作的基础时钟
  - IWDTCLK：看门狗的工作时钟（只能由 ISOSC 提供）
  - TKEYCLK：触控检测模拟部分的工作时钟（只能由 ISOSC 提供）
- 支持多种复位源
  - POR 上电复位
  - EXTRSTB：外部按键复位
  - LVDRST：低电压复位
  - SWRST：软件复位
  - CMRST：外部时钟异常复位
  - IWDRST：独立看门狗复位

  - SYSRST：CPU 申请的复位
- 功耗控制和工作模式
  - RUN 模式：CPU 和所有的外设均处于工作状态
  - SLEEP 模式：CPU 处于挂起状态
  - DEEP-SLEEP 模式：所有的时钟停止（除了 ISOSC，ISOSC 可以配置为在此工作模式下仍旧工作）
- 从 DEEP-SLEEP 模式唤醒
  - 灵活选择唤醒源，包括周期的 IWDT 中断，或者任意的外部中断
  - CPU 支持两种唤醒方式：中断唤醒或者事件唤醒
  - 外部中断源的触发方式可配置为：上升沿、下降沿、上升下降两个沿
- 独立看门狗
  - 异步工作的（ISOSC 时钟）高可靠性独立看门狗
  - 可以在程序中配置使能或者通过 User Option 配置缺省使能

7.2 功能描述

7.2.1 时钟管理和控制

系统控制器最重要的一个功能之一是对芯片的工作时钟进行管理。芯片的工作时钟结构如下图所示。

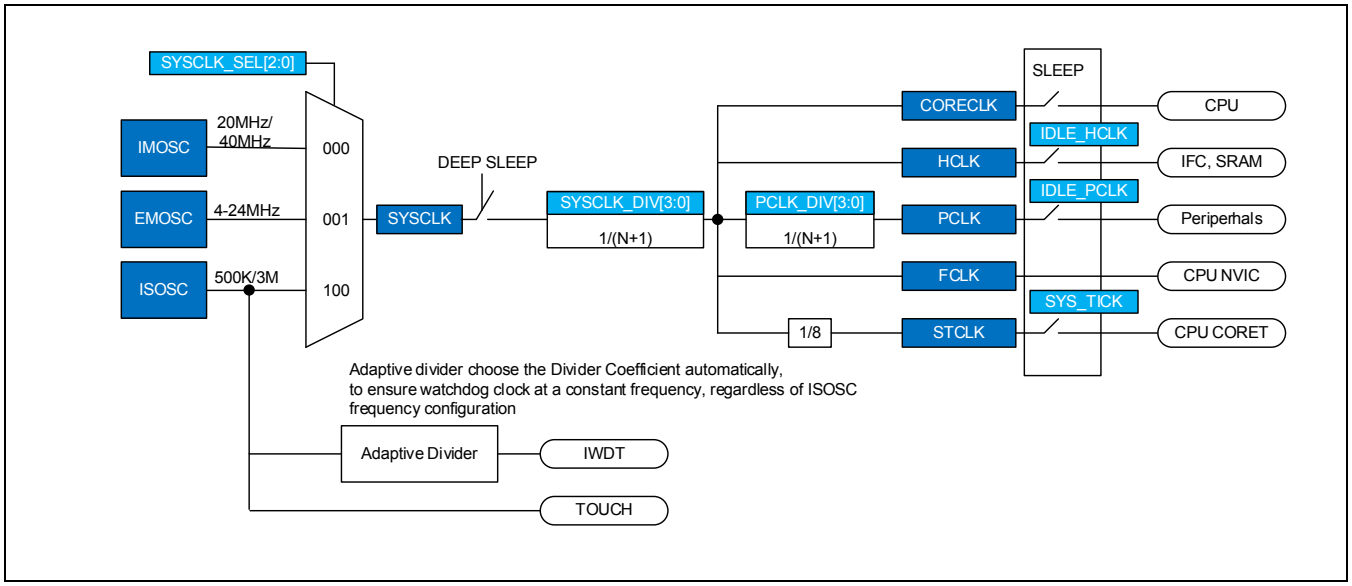


Figure 7-1 时钟结构示意图

NOTE:

1) 在 POR 完成以后，IMOSC 为系统的缺省时钟源。
2) 外部时钟（EMOSC）可以由软件使能、关闭。
3) 在系统时钟切换时，IMOSC 必须使能，切换完成后可以关闭 IMOSC。
4) IWDT 和 TOUCH 模块的时钟源始终为 ISOSC。

外部时钟源（EMOSC）可以通过芯片内置的晶体振荡器，为系统提供稳定和精准的时钟。外部时钟精度高，但是相比于内部振荡器（IMOSC），它的功耗更高。内部振荡器可以提供较好精度的系统工作时钟，相比于外部晶振，它的功耗更低，而且稳定时间更短。系统在缺省时选择内部时钟作为系统时钟，保证了芯片在上电后短时间内可以开始工作。当系统工作时，对计时和时间控制精度要求不是非常高时，推荐使用 IMOSC 作为系统时钟，这样既节省了外部 IO 和晶振的开销，也节省了系统整体的功耗。

芯片内所有的时钟，除了 IWDT 和 TOUCH 的模拟部分时钟，都由系统时钟供给（SYSCLK）。SYSCLK 在芯片处于 DEEP-SLEEP 模式时，将自动断开。CPU 的时钟（CORECLK），AHB 总线的时钟（HCLK），Flash 控制器和 SRAM 控制器的时钟（HCLK）由系统时钟（SYSCLK）派生。所有外设的时钟统称为 PCLK，PCLK 也由系统时钟派生。在 SLEEP 模式下，CPU 的时钟和 HCLK 将会被断开，而 PCLK 的状态，可以通过 GCER/GCDR 寄存器的 IDLE_PCLK 位来配置（缺省状态下，PCLK 会在 SLEEP 模式下继续工作）。

STCLK 是 CPU 内部的 CORET 计数器的时钟（CORET 可以作为产生间隔时序中断的简单计数器使用），STCLK 的频率为系统时钟的 1/8，STCLK 的使能和断开，可以通过 GCER/GCDR 寄存器的 SYSTICK 位来配置。

ISOSC 是芯片内部的低速振荡器，用于给 IWDT 和 TOUCH 的模式部分提供时钟。ISOSC 在 IWDT 使能时，会自动打开。

7.2.2 工作时钟切换

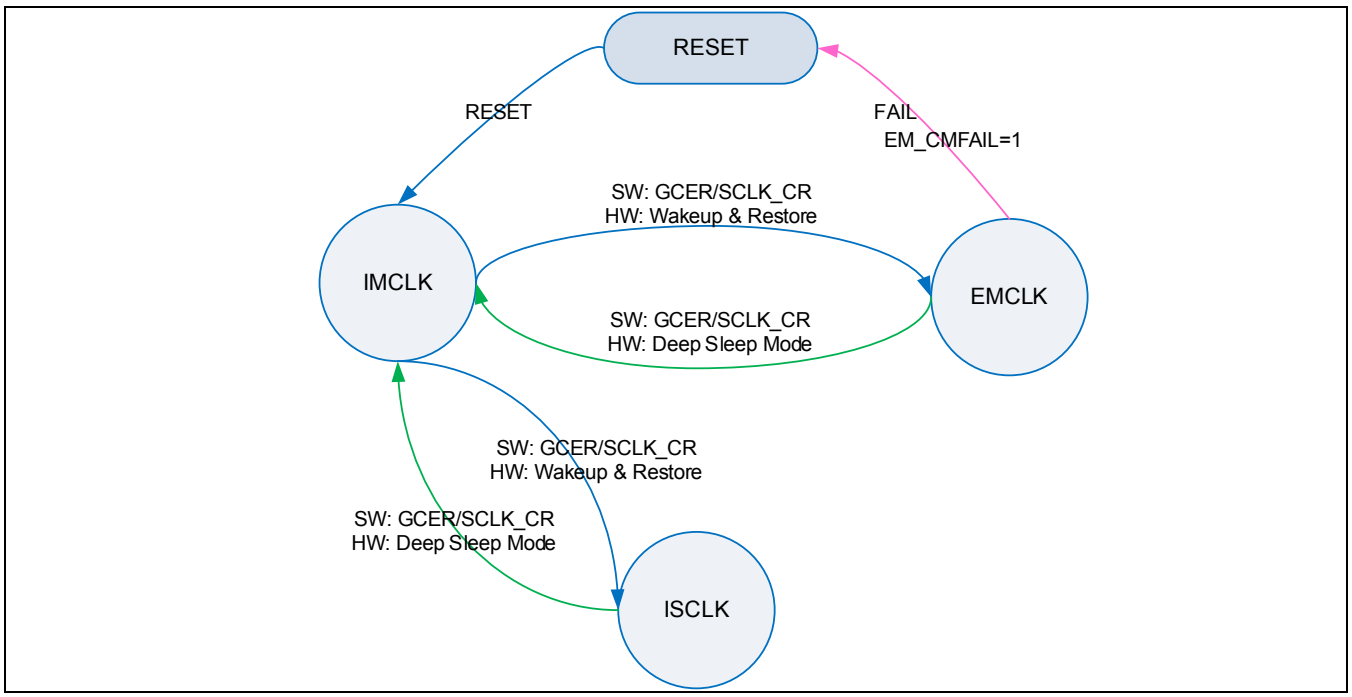


Figure 7-2 时钟切换状态机

系统时钟在不同条件下，可以通过硬件控制或软件切换不同的时钟源。内部高速时钟（IMOSC 的输出为 IMCLK，定义为内部高速时钟）在芯片上电初始化或芯片复位以后，作为芯片的缺省系统时钟工作。在软件中可以通过使能 GCER 寄存器中的相应位，使能需要切换到的时钟源，然后通过设置 SCLKCR 寄存器，软件切换需要的系统工作时钟。当系统时钟进行切换时，目标时钟源必须满足稳定条件（即 STABLE 已经被检测到），否则切换不能成功。目标时钟源的稳定标志可以通过查询 RISR 寄存器得到。

当芯片执行到 STOP 指令（工作模式切换到 DEEP-SLEEP 模式）时，当前的时钟配置将会被自动保存，然后系统硬件自动切换系统时钟到 IMCLK（若 IMOSC 此时被关闭，系统将自动打开 IMOSC），由 IMCLK 作为系统时钟，控制 DEEP-SLEEP 的初始化过程（包括当前系统时钟设置的备份，EMOSC，ISOSC 的停止，功耗模式的切换），在完成所有初始化后，IMOSC 会自动停止。芯片自此进入 DEEP-SLEEP 模式。

系统从 DEEP-SLEEP 唤醒的初始化过程根据系统时钟设置的不同会有差异，整个初始化的过程包括 IMOSC 的稳定，恢复 DEEP-SLEEP 前系统各个时钟模块的配置，恢复系统运行时钟，切换系统时钟到 DEEP-SLEEP 前的时钟源。其中系统时钟配置恢复时间大约在4.6us 左右(20MHz 情况下)，ISOSC 的稳定时间为9个 ISCLK 周期，IMOSC 的稳定时间缺省为256个 IMCLK 周期，IMOSC 的稳定时间可以通过 PWRCR 中的 WKUPTIM0进行调整，在系统复位时 IMOSC 的稳定时间固定为512个 IMCLK 周期。所有的这些 DEEP-SLEEP 前后的时钟切换都由硬件自动完成，对用户程序是完全透明的。

通过设置 GCER 寄存器的 EM_CM 位，可以使能 EMOSC（外部晶振）失效监测功能。在 EMOSC 失效时，系统会自动复位，并置位相应的 RESET ID 标志位。系统的初始化程序，可以通过检查 RESET ID 标志位，判断 EMOSC 的失效，并且选择合适的系统时钟源进行工作。

7.2.3 外部主时钟（EMOSC，EMCLK）

外部主时钟振荡器（EMOSC）可以通过外接晶振，或者直接在 XIN 管脚输入 CLOCK 信号来实现时钟引入。外部晶振和负载电容需要尽可能靠近芯片以保证时钟的稳定，和减少起振时间。对于不同类型的晶振，需要调整振荡器的增益控制 CYOSC_GM 和振荡器的驱动控制 CYOSC_CD 位，以满足不同起振条件。

Table 7-1 CYOSC_GM 设置说明

EMOSC 频率	CYO_GM[2:0]	CYO_CD
20MHz	111	1
32.768KHz	000	0

如果当前系统工作时钟选择 EMOSC 作为系统时钟源，那么在芯片 SLEEP 模式下，EMOSC 不会改变工作状态。在芯片进入 DEEP-SLEEP 模式后，EMOSC 会自动停止工作，并在系统被唤醒以后，被硬件自动使能。在芯片上电或者复位后，EMOSC 缺省处于关闭状态。

EMOSC 的使能时通过对 GCER 的 EMOSC 位置高来实现的。当使能 EMOSC 以后，内部的一个时钟稳定计数器将自动开始计数，该计数器为一个递减计数器，计数器初始值可以通过 OSTR 寄存器中 EM_CNT 来设置。一旦计数器的计数值变为‘0’，则表示 EMOSC 稳定的振荡已经建立，相应的，RISR 寄存器中 EMOSC_ST 位将被置高。当对 GCDR 的 EMOSC 位置高，则 EMOSC 会被关闭，并且相应的 GCSR 寄存器中的 EMOSC 使能状态位会被清除。

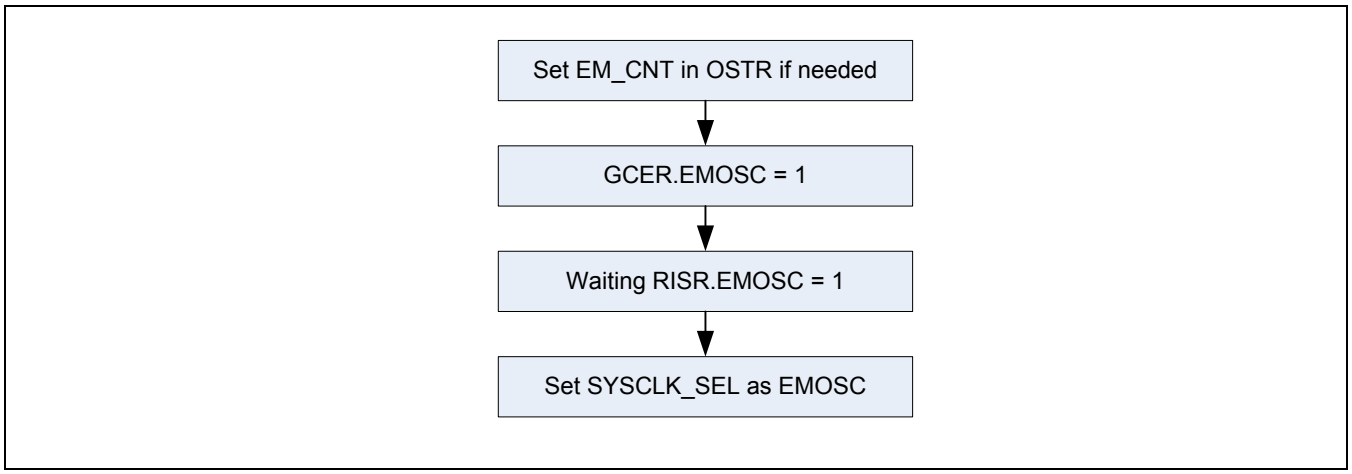


Figure 7-3 EMOSC 使能和系统时钟选择设置流程

当外部时钟稳定被检测到以后，RISR 寄存器中相应的 EMOSC_ST 位会被置起。如果相应的 IMSR 寄存器中的 EMOSC 位处于使能状态，则系统会产生外部时钟稳定中断。在中断服务程序中，可以通过检测 ISR 寄存器中的 EMOSC_ST 标志位来判断该中断的产生。RISR 中的标志位，无论中断是否使能，在外部时钟稳定后都会置位。

GCSR 寄存器中的状态位表示使能请求是否置位，但是通过该标志位并不能判断请求使能的震荡器已经正确打开。只有当该震荡器的稳定标志位（RISR 中的相应标志位）置位后，才能确认震荡器已经工作。RISR 中相应标志位一旦置位不会自动清除，需要通过软件清除。

7.2.4 内部主时钟振荡器（IMOSC，IMCLK）

芯片内部有一个高速的 RC 振荡器，可以作为系统工作的主要时钟源。当 IMOSC 使能时，GCSR 寄存器的 IMOSC 状态标志位将始终为‘1’。在芯片上电后复位以后，IMOSC 是芯片的缺省工作时钟。由于 IMOSC 的稳定

时间非常短（相比较于外部晶振），所以系统从 DEEP-SLEEP 或者 POR 以后进入正常工作模式的时间可以大幅的缩短，有效的提升系统响应时间。

如果当前系统工作时钟选择 IMOSC 作为系统时钟源，那么在芯片 SLEEP 模式下，IMOSC 不会改变工作状态。在芯片进入 DEEP-SLEEP 模式后，IMOSC 会自动停止工作，并在系统被唤醒以后，被硬件自动使能。内部主时钟震荡器支持两种频率选择，在系统复位后，缺省为低频率模式，通过 CLCR 寄存器可以进行频率的切换。在改变 IMOSC 的运行频率时，必须保证 IMCLK 不作为当前的系统时钟运行。

7.2.5 内部副时钟振荡器（ISOSC，ISCLK）

芯片内部有一个低速的 RC 振荡器，这个振荡器可以作为系统的工作时钟。在不需要非常精准的计时应用时，选择低速振荡器以减少系统的整体功耗。同时 ISOSC 也是内部独立看门狗电路的唯一时钟源，和内部 TOUCH 模块模拟电路的时钟源。当 IWDT 或者 TOUCH 被使能时，ISOSC 在 DEEP-SLEEP 模式下将不会被硬件自动停止。

ISOSC 同时也作为 EMOSC 失效监测时的参考时钟，所以当使能 EMOSC 失效监测时，ISOSC 必须是使能的。

7.2.6 外部时钟可靠性监测

外部时钟可靠性监测是对外部振荡器（EMOSC）可用性的一种监测。当外部时钟监测被使能时，内部副时钟振荡器（ISOSC）作为参考时钟源，必须同时使能。一个内部的8位递减计数器在 EMOSC 的时钟控制下进行计数，每一个 ISOSC 的时钟周期，硬件都会自动比较上一次的计数值和当前的计数值，如果连续三次的比较值都保持一致，则认为外部时钟可能失效（芯片保证 ISOSC 和 EMOSC 的频率不是整数倍，避免监测错误）。

外部时钟失效监测通过设置 GCER 寄存器中的 EM_CM 位来使能。当失效被检测到，系统根据不同设置做出如下的操作：

- 当 CMRST 使能，系统产生 CMRST 信号，芯片复位。
- 当 CMRST 未使能，并且当前系统时钟未选择 EMOSC 时，系统产生 EM_CMFAIL 事件。
- 当 CMRST 未使能，且当前系统时钟选择为 EMOSC 时，自动切换系统时钟到 ISOSC 并产生 EM_CMFAIL 事件。

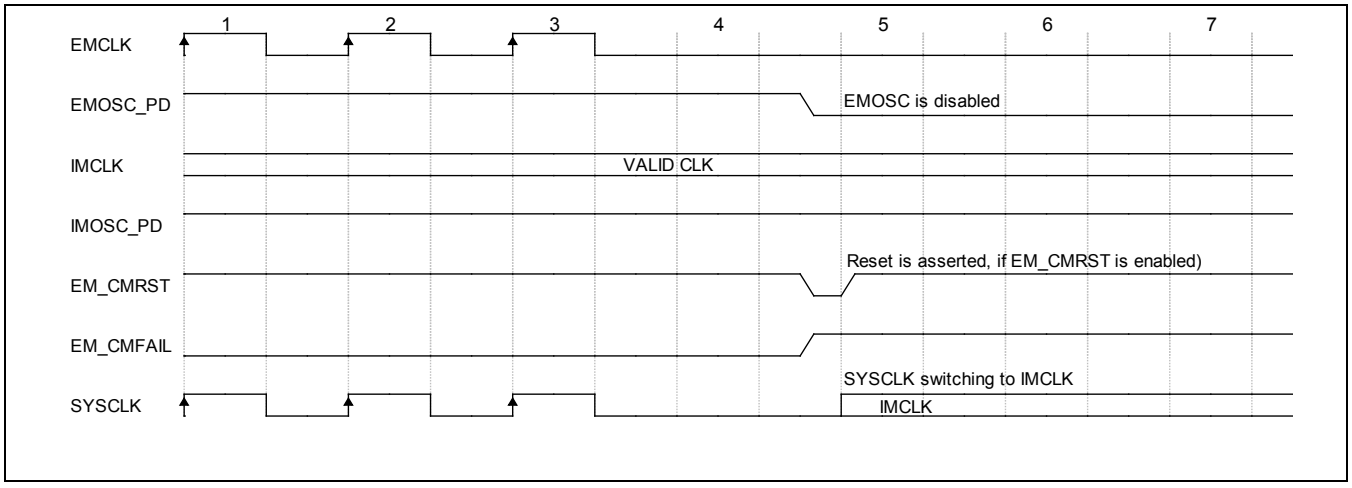


Figure 7-4 EMOSC 失效并产生复位

外部时钟失效监测需要在 EMOSC 稳定以后才可以使能，EM_CMRCV 中断表示 EMOSC 的时钟失效已经恢复。由于外部时钟失效而引发的系统复位，不会清除 EM_CMFAIL 标志（此标志位为异常事件，所以必须软件强制清除，但是断电后上电复位，外部复位，低电压复位和看门狗复位可以清除这个标志）。在 CMRST 触发后，程序再次使能 EM 功能时，由于硬件中存在 EMOSC 异常复位记录，一旦检测到 EMOSC 已经恢复，EM_CMRCV 标志会自动置位，表示从上次的异常中恢复，可以通过软件强制清除。

在 EM_CMFAIL 发生后，如果未指定系统复位，系统将当前的工作时钟自动切换到 ISOSC，程序可以继续执行并自动产生 EMOSC 的复位信号，尝试重新启动 EMOSC。一旦 EMOSC 的工作恢复，EM_CMRCV 标志将被置位，软件可以通过该标志位或者相应中断，尝试恢复到 EMOSC 作为系统时钟工作。恢复的工作可以参考如下顺序进行：

- 清除 RISR 寄存器中 EM_CMFAIL 标志位和 SYSCLK_ST 标志位。
- 配置 SCLKCR 寄存器，选择 EMOSC 作为系统时钟。

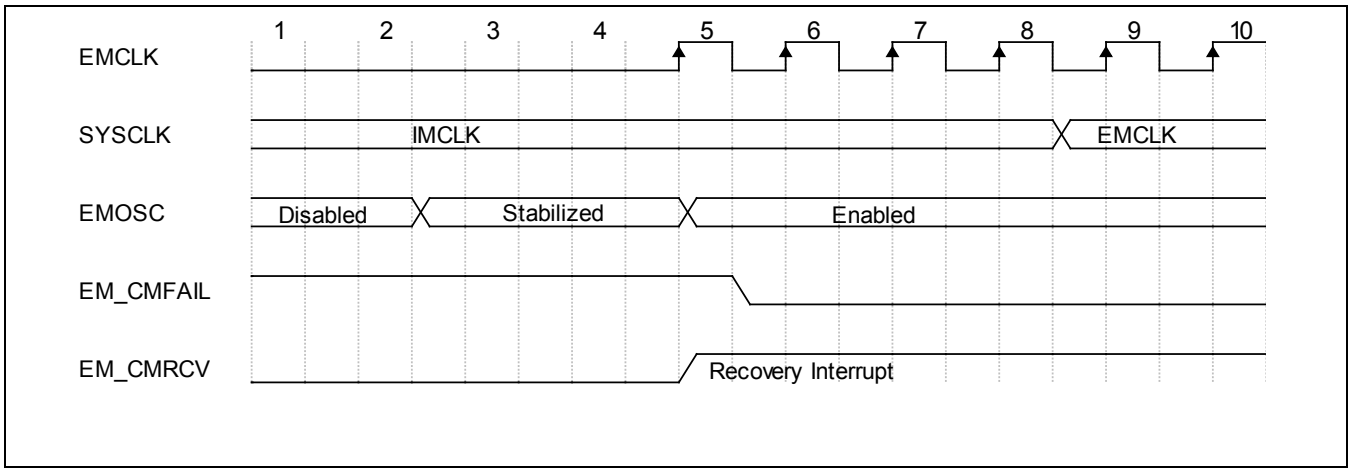


Figure 7-5 EMOSC 从 CM Fail 恢复以后重新使能

7.2.7 功耗管理

芯片支持三种不同的工作模式：普通（RUN）、睡眠（SLEEP）、深睡眠（DEEP-SLEEP）。工作模式的切换使用汇编指令：“stop”指令进入深睡眠工作模式，“doze”指令进入睡眠工作模式，当相应的唤醒中断发生后，切换为普通模式。针对不同的工作模式，可以设置不同的功耗调整策略。在某些特定应用场合，需要特别优化功耗时，可以通过设置 PWRCR 寄存器来设置不同工作模式下的驱动能力，以减少额外的电流消耗。例如在使用 ISOSC 运行时，由于系统工作在非常低的时钟下，可以通过设置 PWRCR 的 RUNCFG 位来关闭内部高精度参考源和切换内部整流电路的驱动能力，而从达到大幅降低系统整体功耗的需求。

调整运行状态功耗配置时，必须先将系统运行功耗调低以后再进行功耗配置切换，否则将造成因为驱动不足而导致系统异常。对低功耗模式的配置，必须在切换工作模式前进行配置，一旦进入低功耗模式，系统将根据 PWRCR 的设置提供供电。在低功耗模式下，不能再对功耗进行配置。在正常模式下，也可以对系统 RUN 模式下的功耗进行优化，但是由于系统一直处于运行中，在减小驱动前，必须先将系统的时钟降低，以降低系统负荷，然后再进行驱动切换，否则会造成由于驱动不足而逻辑错误。

在系统从低功耗模式（SLEEP 或者 DEEP-SLEEP）切换为 NORM 模式时，系统会自动插入稳定时间，以保证切换的稳定。当在模式切换时，如果存在驱动或者参考源切换，需要适当增加稳定的时间，以保证内部供电的稳定。此稳定时间可以通过 PWRCR 的 WKUPTIM 位进行设置。

7.2.7.1 RUN 模式

RUN 模式，通常也被称作普通模式，是芯片工作的最基本模式。在上电复位以后，芯片即进入此模式工作。所有的逻辑模块在 PCLK 使能的前提下，都可以正常工作。CPU 在该模式下全速工作。缺省时，除了 IFC 和 SYSCON 模块，其他模块的 PCLK 都处于 DISABLE 状态，可以通过设置寄存器 PCER0 和 PCER1 来使能相应功能模块的 PCLK。所有功能模块的寄存器都必须在 PCLK 使能以后才可以修改。

7.2.7.2 SLEEP 模式

在 SLEEP 模式下，系统控制器将挂起 CORE 模块的时钟（CPU 将不会工作）。缺省模式下，所有的逻辑外设控制时钟（PCLK）也将被停止。如果 SLEEP 模式通过指定外设唤醒，而指定外设的工作时钟为 PCLK，例如某计时器是通过 PCLK 来进行工作的，则在 SLEEP 进入前，必须通过设置 IDLE_PCLK 位来修改 SLEEP 模式下，PCLK 不被挂起，从而保证在 SLEEP 模式下，该外设仍旧可以工作。所有振荡器的工作状态在该模式下不会做任何改变。

任何外设事件或者中断都可以触发系统从该模式退出。如果 PCLK 被配置为在 SLEEP 模式下挂起（IDLE_PCLK 未使能），则不能产生任何外设的事件或者中断。

SLEEP 模式相比于 DEEP-SLEEP 模式，由于不存在时钟源的使能切换，有更快的唤醒响应时间，可以在系统应用需要快速唤醒，并对功耗有一定要求的应用中采用。

7.2.7.3 DEEP-SLEEP 模式

在 DEEP-SLEEP 模式下，系统控制器将挂起所有的时钟源，但是内部逻辑电源仍将保持不变。在 IWDT 或 TOUCH 模块使能前提下，ISOSC 可以在该模式下继续工作。IMOSC 和 EMOSC 在进入 DEEP-SLEEP 模式以后，会被自动关闭。当处理器从 DEEP-SLEEP 模式退出时，系统工作时钟（SYSCLK）将会自动恢复到之前的状态。

当处理器进入 DEEP-SLEEP 模式后，由于系统所有的时钟都被关闭，只有特定的几类中断源可以唤醒处理器：

Table 7-2 可以唤醒 DEEP-SLEEP 的中断源

PERIPHERAL	中断类型
GPIO	所有外部中断(EXI)
IWDT	Alert中断
TOUCH	通过TCH_IMCR使能的中断
LVD	低电压检测中断

特定中断触发后，IMOSC 会被使能，并从 DEEP-SLEEP 模式退出。中断的唤醒配置需要通过 SYSCON 和 CPU 的中断控制器设置（CPU 的中断设置，参考中断控制器章节的描述），EXI，IWDT 和 LVD 外设的中断设置在 SYSCON 寄存器中，TOUCH 外设的中断参考 TOUCH 章节中的说明。

7.2.8 外部供电监测 （LVD）

LVD 提供外部电源的监测功能。该模块可以根据设置，在外部供电电压低于设置值时，产生系统中断或者芯片

复位信号。

LVD 在初次上电时，缺省使能，通过配置 LVDCR 的 LVD 使能控制位，可以通过软件禁止 LVD 模块。当 LVD 模块使能以后，处理器将在外部供电电压低于 RSTDET_LVL 的设置值时，产生硬件复位信号。当中断使能时（通过 LVD_INT 控制位设置），处理器会在外部供电电压低于 INTDET_LVL 设置值时，产生中断请求（IER、IDR 寄存器中的 LVD_INT 位可以设置或者清除中断标志）。当前外部供电电压的状态，可以通过 LVDCR 的 LVDFLAG 位检测到。当外部供电电压低于检测 level 时，该标志位为‘1’，当高于检测 level 时，该标志位为‘0’。

由 LVD 产生的系统复位信号，不会清除 LVD 的使能状态。LVD 模块可以在芯片处于 Deep-Sleep 模式时使能，通过 LVD 的中断唤醒系统。

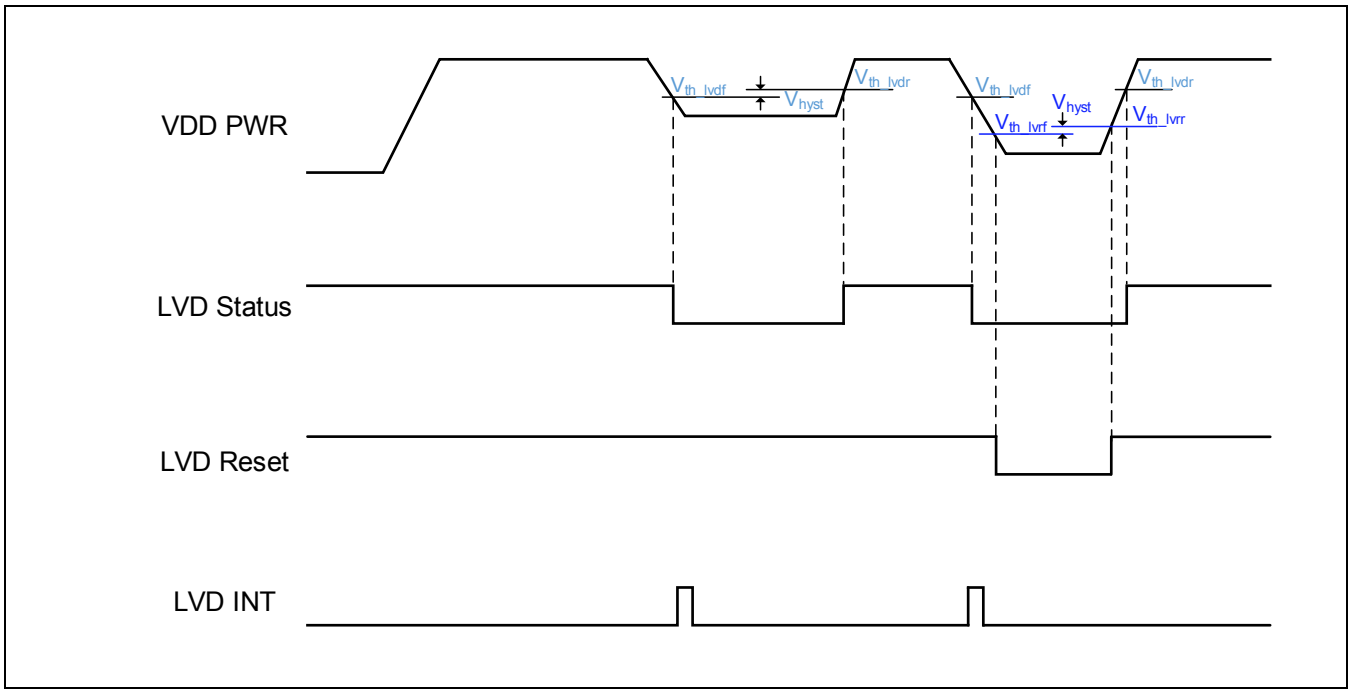


Figure 7-6 LVD 工作时序图

7.2.9 复位管理 （RESET ID）

处理器内嵌一个复位历史记录控制器，专门用于记录引起系统复位的 RESET 源。通过判读该寄存器，可以定位系统的异常复位，并根据需要作出相应的软件处理。下表中描述了处理器所有可能的复位信号源。

Table 7-3 处理器复位信号源表

信号名	描述
EXTRST	外部输入的硬件 RESET 信号（低电平有效）。在外部复位脚有效时可用。
CMRST	由内部产生的 EMOSC 时钟异常复位信号。可以通过软件使能或关闭该功能。
LVDRST	由低电压监测模块（LVD）产生的系统复位信号。可以通过软件势能或者关闭该功能。
IWDTRST	由内部看门狗电路产生的复位信号。
SWRST	由系统控制器产生的软件复位信号。(IDCCR 寄存器中的 SWRST 控制位)

SYSRSTREQ	由 CPU 产生的系统复位请求（通过 MTCR 指令对 CPU 中的 SRCR 寄存器写入 0xABCD1234）。
POR	上电复位

每一个复位信号都对应 RSR 寄存器中的一个状态位。可以通过软件读取该寄存器鉴别处理器的复位信号源。该寄存器中的所有位信息在电源上电复位（POR）以后，会自动清除。有效的复位在芯片复位成功后自动清除 RSR 寄存器中的其他标志位，并记录当前复位的触发源。

7.2.10 外部中断管理 （EXI）

处理器的所有 GPIO 都可以设置为外部中断输入。在该芯片中，最多有16个外部中断信号线。同一个组的管脚可以被配置为该组中断信号线的触发源。外部中断管脚分组的定义如下图所示。详细的管脚设置和分组设置可以参考 GPIO 章节。外部中断有别于 SYSCON 中的其他中断，外部中断在 CPU 中有独立的中断源。

外部中断的中断线控制在 SYSCON 中通过一组寄存器实现。EXIECR/EXIEDR 寄存器可以使能或者关闭指定的外部中断信号线，当前的中断线设置状态可以通过 EXIMR 寄存器获得。外部中断线的 pending 状态可以通过 EXICR 寄存器查询，对 EXICR 寄存器相应位写入‘1’，可以清除该中断线的 pending 状态。中断的原始 pending 状态可以通过 EXIRS 寄存器查询。外部中断可以支持软件触发，通过设置 EXIAR 可以在没有外部硬件触发的情况下，直接由软件触发外部中断。

外部中断的触发可以选择输入信号上、下边沿中的任意一个，或者同时两个类型，通过 EXIRT 和 EXIFT 寄存器进行设置。只要相应的管脚处于输入状态，不论是否设置成 GPIO 输入模式，都支持外部中断触发。详细参考 GPIO 的外部中断章节。

Table 7-4 外部中断源分组

EXI 中断信号线	中断源
EXI0	PA0.0 or PA1.0 or PB0.0 or PC0.0 or PD0.0
EXI1	PA0.1 or PA1.1 or PB0.1 or PC0.1 or PD0.1
EXI2	PA0.2 or PA1.2 or PC0.2
EXI3	PA0.3 or PA1.3 or PC0.3
EXI4	PA0.4 or PA1.4
EXI5	PA0.5 or PA1.5
EXI6	PA0.6
EXI7	PA0.7
EXI8	PA0.8
EXI9	PA0.9
EXI10	PA0.10
EXI11	PA0.11
EXI12	PA0.12
EXI13	PA0.13
EXI14	PA0.14
EXI15	PA0.15

7.2.11 独立的看门狗定时器 （IWDT）

看门狗定时器的作用是在系统运行中，由于程序干扰或者错误运行，导致处理器运行到一个未知的状态中时，产生一个系统复位请求，把处理器重新置位到初始化状态。他可以保证系统不会因为程序运行错误导致永久挂起。除外，看门狗定时器中断还可以作为处理器在 DEEP-SLEEP 模式下定时唤醒的中断源，间隔唤醒系统工作，大幅降低系统的整体功耗。IWDT 是一个独立工作的看门狗模块，和系统的工作状态无关。它内部通过一个18位的 Free Running 递减计数器控制定时时间。

IWDT 的缺省状态可以通过 Flash 内部的 User Option 设置。在软件中，可以通过设置 IWDEDR 寄存器中的 IWDT_EDC 位来打开或者关闭 IWDT。当清除 IWDT 操作时，IWDT 中计数器的预置数会被重新置位。清除 IWDT 通过对 IWDCNT 寄存器的 IWDT_CLR 位写入0x5A 实现。

IWDT 在 ISOSC 控制下，会一直从预置数递减计数值。当计数器值变为零时，会自动产生系统复位信号。预置值通过 IWDCR 寄存器的 IWDT_TIME 位设置。IWDT_TIME 一共为3位，对应8种定时时间设置。最小定时时间为 128ms。

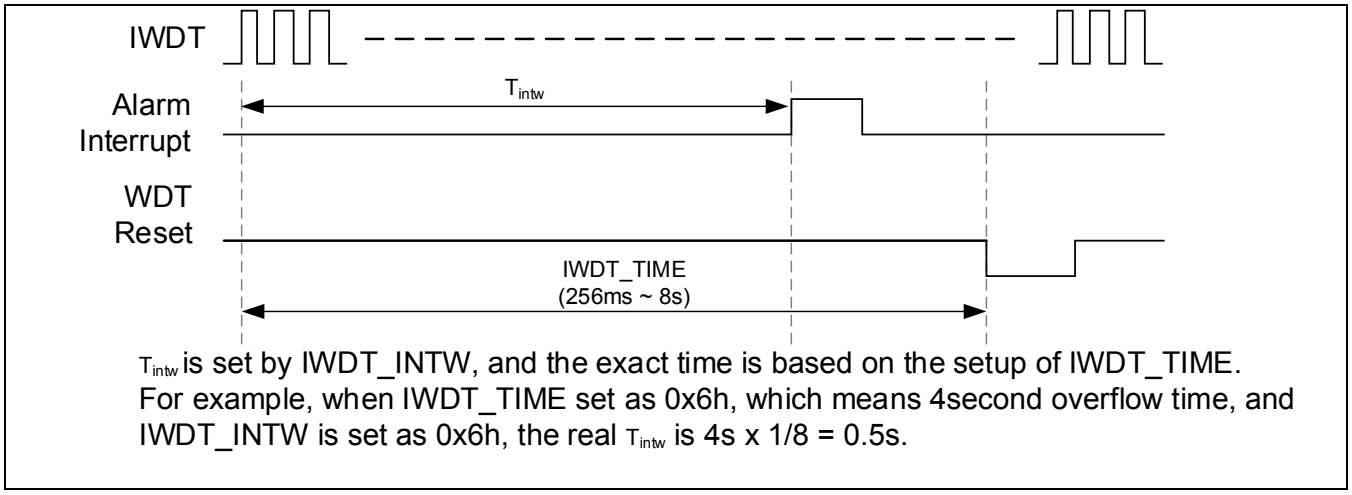


Figure 7-7 IWDT 溢出和中断间隔

IWDT 还支持报警功能。在计数器计数过程中，当计数值达到 IWDT_INTW 设置值时，会产生一个中断信号。IWDT_INTW 的设置值表示中断发生的时间点在整个计数周期的百分比位置。例如，当 IWDT_TIME 设置看门狗定时溢出时间为4秒，如果 IWDT_INTW 设置为3'b101，则表示在计数器计时到 4 x 2/8 = 1秒时，系统会产生一个报警中断。

7.2.12 错误命令处理

系统控制器提供了一种自动检测由于错误设置寄存器，而导致系统挂起或者功能错误的机制。当发现当前寄存器操作会导致系统挂起或者错误风险时，系统控制器会忽略当前操作，并产生错误中断。当有错误中断发生时，可以通过检查 SYSCON_ERRINF 寄存器获得详细错误信息。

例如，当程序试图通过设置 SCLKCR 寄存器，将系统时钟切换到 EMOSC，而此时 EMOSC 未被使能，或者已经使能，但是振荡未达到稳定时，控制器将会忽略这次切换操作，并给出命令错误中断。

7.3 寄存器说明

7.3.1 寄存器表

- Base Address: 0x4001_1000

Table 7-5 寄存器表

Register	Offset	Description	Reset Value
SYSCON_IDCCR	0x000	ID 和控制器模块时钟控制寄存器	0x0000_0001
SYSCON_GCER	0x004	通用使能控制寄存器	0x0000_0000
SYSCON_GCDR	0x008	通用禁止控制寄存器	0x0000_0000
SYSCON_GCSR	0x00C	通用状态寄存器	0x0000_0003
RSVD	0x010	保留	0x0000_0000
RSVD	0x014	保留	0x0000_0000
RSVD	0x018	保留	0x0000_0000
SYSCON_SCLKCR	0x01C	系统时钟控制寄存器	0x0000_0800
SYSCON_PCLKCR	0x020	外设时钟控制寄存器	0x0000_0100
RSVD	0x024	保留	0x0000_0000
SYSCON_PCER0	0x028	外设时钟使能寄存器0	0x0000_0000
SYSCON_PCDR0	0x02C	外设时钟禁止寄存器0	0x0000_0000
SYSCON_PCSR0	0x030	外设时钟状态寄存器0	0x0000_0001
SYSCON_PCER1	0x034	外设时钟使能寄存器1	0x0000_0000
SYSCON_PCDR1	0x038	外设时钟禁止寄存器1	0x0000_0000
SYSCON_PCSR1	0x03C	外设时钟状态寄存器1	0x0000_0000
SYSCON_OSTR	0x040	外部振荡器稳定时间配置寄存器	0x00FF_03FF
RSVD	0x044	保留	0x0000_03FF
RSVD	0x048	保留	0x0000_0000
SYSCON_LVDCR	0x04C	低电压检测控制寄存器	0x0000_0000
SYSCON_CLCR	0x050	CLO 配置寄存器	0x0000_01FF
SYSCON_PWRCR	0x054	功耗控制寄存器	0x0000_1F09
SYSCON_OPT4	0x058	系统配置寄存器4 (TRIM value for OPA OFFSET) [1]	
SYSCON_OPT3	0x05C	系统配置寄存器3 (TRIM value for CMP OFFSET) [1]	
SYSCON_OPT2	0x060	系统配置寄存器2 (TRIM value for CMP OFFSET) [1]	
SYSCON_OPT1	0x064	系统配置寄存器1 (TRIM value for OSC) [1]	0x0000_XXXX
SYSCON_OPT0	0x068	系统配置寄存器0 [2]	-
RSVD	0x06C	保留	-
RSVD	0x070	保留	-
SYSCON_IECR	0x074	中断使能控制寄存器	0x0000_0000

Register	Offset	Description	Reset Value
SYSCON_IDCR	0x078	中断禁止控制寄存器	0x0000_0000
SYSCON_IMSR	0x07C	中断使能/禁止状态寄存器	0x0000_0000
SYSCON_IAR	0x080	中断软件触发寄存器	0x0000_0000
SYSCON_ICR	0x084	中断清除寄存器	0x0000_0000
SYSCON_RISR	0x088	原始中断标志状态寄存器	0x0000_0000
SYSCON_ISR	0x08C	中断标志状态寄存器	0x0000_0000
SYSCON_RSR	0x090	复位记录状态寄存器	-
SYSCON_EXIRT	0x094	外部中断上升沿选择寄存器	0x0000_0000
SYSCON_EXIFT	0x098	外部中断下降沿选择寄存器	0x0000_0000
SYSCON_EXIER	0x09C	外部中断使能寄存器	0x0000_0000
SYSCON_EXIDR	0x0A0	外部中断禁止寄存器	0x0000_0000
SYSCON_EXIMR	0x0A4	外部中断使能/禁止状态寄存器	0x0000_0000
SYSCON_EXIAR	0x0A8	外部中断软件触发寄存器	0x0000_0000
SYSCON_EXICR	0x0AC	外部中断清除寄存器（中断状态标志位寄存器）	0x0000_0000
SYSCON_EXIRS	0x0B0	外部中断原始标志状态寄存器	0x0000_0000
SYSCON_IWDCR	0x0B4	看门狗控制寄存器	0x0000_070C
SYSCON_IWDCNT	0x0B8	看门狗控制计数器值	0x0003_FFFF
SYSCON_IWDEDR	0x0BC	看门狗使能寄存器	0x0000_XXXX
SYSCON_CINF0	0x0C0	客户信息区0[3]	-
SYSCON_CINF1	0x0C4	客户信息区1	-
SYSCON_FINF0	0x0C8	工程信息区0[4]	-
SYSCON_FINF1	0x0CC	工程信息区1	-
SYSCON_ERRINF	0x0E0	错误命令信息查询寄存器	0x0000_0000

NOTE:

1. 内部主振荡器的频率在出厂时已经经过校准，但可以在软件中通过寄存器再次调整。
2. 存储于 Flash 内部的保护状态信息，可以通过这个寄存器查看。
3. 存储于 Flash 中的客户信息区的第一个 Word（32bit）的内容被自动映射到客户信息区0，第二个 Word 的内容被映射到客户信息区1。
4. 存储于 Flash 中的工程信息区的第一个 Word（32bit）的内容被自动映射到工程信息区0，第二个 Word 的内容被映射到工程信息区1。

7.3.2 SYSCON_IDCCR (ID 和控制器模块时钟控制寄存器)

- Address = Base Address + 0x0000, Reset Value = 0x0000_0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID_KEY																IDCODE								SWRST	RSVD						CLKEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	W	R	R	R	R	R	R	RW

Name	Bit	Type	Description
CLKEN	[0]	R/W	使能 SYSCON 模块的 APB 时钟。
SWRST	[7]	W	软件复位。 0：没有效果 1：执行软件复位操作
IDCODE	[31:8]	R	ID Code 寄存器。 这个区域保存了相应 IP 的 IDCODE。
ID_KEY	[31:16]	W	对本寄存器进行写操作时，需要填入对应的 KEY 值。 只有在 ID_KEY 等于0xE11E 时，对本寄存器的写入才有效。

7.3.3 SYSCON_GCER/GCDR (通用使能/禁止控制寄存器)

- GCER：Address = Base Address + 0x0004, Reset Value = 0x0000_0000
- GCDR：Address = Base Address + 0x0008, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												EM_CMRST	EM_CM	RSVD						SYSTICK	RSVD		IDLE_PCLK	RSVD				EMOSC	RSVD	IMOSC	ISOSC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	W	W	R	R	R	R	R	R	W	R	R	W	R	R	R	R	W	W	W	W

Name	Bit	Type	Description
ISOSC	[0]	W	使能/禁止 ISOSC 振荡器。
IMOSC	[1]	W	使能/禁止 IMOSC 振荡器。
EMOSC	[3]	W	使能/禁止 EMOSC 振荡器。
IDLE_PCLK	[8]	W	使能/禁止在 SLEEP 模式下的 PCLK。
SYSTICK	[11]	W	使能/禁止 STCLK（CPU 内部 CORET 计时器时钟）。
EM_CM	[18]	W	使能/禁止外部晶振监测功能。
EM_CMRST	[19]	W	使能/禁止外部晶振失效时产生系统复位。 当 SYSCLK=EMCLK 时，一旦使能 EM_CM 功能，EM_CMRST 就会强制使能。

NOTE:

GCER 和 GCDR 寄存器只有对写入的‘1’敏感，写入‘0’时无效。

7.3.4 SYSCON_GCSR (通用状态寄存器)

- Address = Base Address + 0x000C, Reset Value = 0x0000_0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												EM_CMRST	EM_CM	RSVD						SYSTICK	RSVD		IDLE_PCLK	RSVD				EMOSC	RSVD	IMOSC	ISOSC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
ISOSC	[0]	R	ISOSC 振荡器使能状态。 0：ISOSC 振荡器被禁止。 1：ISOSC 振荡器被使能。
IMOSC	[1]	R	IMOSC 振荡器使能状态。 0：IMOSC 振荡器被禁止。 1：IMOSC 振荡器被使能。
EMOSC	[3]	R	EMOSC 振荡器使能状态。 0：EMOSC 振荡器被禁止。 1：EMOSC 振荡器被使能。
IDLE_PCLK	[8]	R	SLEEP 模式下的 PCLK 使能/禁止状态。 0：在 SLEEP 模式下，PCLK 被禁止。 1：在 SLEEP 模式下，PCLK 被使能。 如果在 SLEEP 模式下，禁止了 PCLK，外设将不能产生中断。
SYSTICK	[11]	R	STCLK 时钟使能状态。 0：STCLK 被禁止。 1：STCLK 被使能。
EM_CM	[18]	R	外部时钟监测功能使能状态。 0：外部时钟监测被禁止。 1：外部时钟监测被使能。
EM_CMRST	[19]	R	使能/禁止外部时钟失效时的系统复位。 0：时钟失效时，复位禁止。 1：时钟失效时，复位使能。

7.3.5 SYSCON_SCLKCR (系统时钟控制寄存器)

- Address = Base Address + 0x001C, Reset Value = 0x0000_0400

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

Name	Bit	Type	Description
SYSCLK_SEL	[2:0]	R/W	系统时钟选择控制位。 000：选择 IMOSC 作为系统时钟。 001：选择 EMOSC 作为系统时钟。 100：选择 ISOSC 作为系统时钟。
SYSCLK_DIV	[11:8]	R/W	CPU 时钟分频设置。 0000(0)：不分频，等于系统时钟。 0001(1)：不分频，等于系统时钟。 0010(2)：2分频。 0011(3)：3分频。 0100(4)：4分频。 0101(5)：5分频。 0110(6)：6分频。 0111(7)：7分频。 1000(8)：8分频。 1001(9)：12分频。 1010(10)：16分频。 1011(11)：24分频。 1100(12)：32分频。 1101(13)：64分频。 1110(14)：128分频。 1111(15)：256分频。
SYSCLK_KEY	[31:16]	W	对本寄存器进行写操作时，需要填入对应的 KEY 值。 只有在 KEY 等于0xD22D 时，对本寄存器的写入才有效。

7.3.6 SYSCON_PCLKCR (外设时钟控制寄存器)

- Address = Base Address + 0x0020, Reset Value = 0x0000_0100

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

Name	Bit	Type	Description
PCLK_DIV	[11:8]	R/W	PCLK 时钟分频设置。 0000：不分频，等于系统时钟。 0001：2分频。 001x：4分频。 01xx：8分频。 1xxx：16分频。
PCLK_KEY	[31:16]	W	对本寄存器进行写操作时，需要填入对应的 KEY 值。 只有在 KEY 等于0xC33C 时，对本寄存器的写入才有效。

7.3.7 SYSCON_PCER0/PCDR0 (外设时钟使能/禁止寄存器0)

- PCER0：Address = Base Address + 0x0028, Reset Value = 0x0000_0000
- PCDR0：Address = Base Address + 0x002C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								I2C														UART1	UART0	RSVD	TKEY	RSVD	ADC	RSVD			IFC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	W	R	R	R	R	R	R	R	R	R	R	R	R	W	W	R	W	R	W	R	R	R	W

Name	Bit	Type	Description
IFC ADC TKEY UART0 UART1 I2C	[-]	W	使能/禁止相应外设模块的 PCLK 时钟。 只有对相应位写‘1’时才有效，写‘0’时无效 PCER 相应位写‘1’时，使能相应模块 PCLK 时钟， PCDR 相应位写‘1’时，禁止相应模块 PCLK 时钟。

7.3.8 SYSCON_PCSR0 (外设时钟状态寄存器0)

- Address = Base Address + 0x0030, Reset Value = 0x0000_0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									I2C													UART1	UART0	RSVD	TKEY	RSVD	ADC	RSVD			IFC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
IFC ADC TKEY UART0 UART1 I2C	[-]	R	相应外设模块的 PCLK 时钟的使能/禁止状态。 0：该对应模块的时钟被禁止。 1：该对应模块的时钟被使能。

7.3.9 SYSCON_PCER1/PCDR1 (外设时钟使能/禁止寄存器1)

- PCER0：Address = Base Address + 0x0034, Reset Value = 0x0000_0000
- PCDR0：Address = Base Address + 0x0038, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				GPIOD	GPIOC	GPIOB	GPIOA	RSVD	OPA	CMP4	CMP3	CMP2	CMP1	CMP0	LED	RSVD		RSVD	EPWM	TC3	TC2	TC1	TC0	RSVD							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	W	W	W	R	R	R	R	R	R	R	W	R	R	W	R	W	W	W	W	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
TC0 TC1 TC2 TC3 EPWM LED CMP0 CMP1 CMP2 CMP3 CMP4 OPA GPIOA GPIOB GPIOC GPIOD	[-]	W	使能/禁止相应外设模块的 PCLK 时钟。 只有对相应位写‘1’时才有效，写‘0’时无效 PCER 相应位写‘1’时，使能相应模块 PCLK 时钟， PCDR 相应位写‘1’时，禁止相应模块 PCLK 时钟。

7.3.10 SYSCON_PCSR1 (外设时钟状态寄存器1)

- Address = Base Address + 0x003C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				GPIOD	GPIOC	GPIOB	GPIOA	RSVD							LED	RSVD		CNTA	RSVD	GTC3	GTC2	GTC1	GTC0	RSVD							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
GTC0 GTC1 GTC2 GTC3 CNTA LED GPIOA GPIOB GPIOC GPIOD	[-]	R	相应外设模块的 PCLK 时钟的使能/禁止状态。 0：该对应模块的时钟被禁止。 1：该对应模块的时钟被使能。

7.3.11 SYSCON_OSTR (外部振荡器稳定时间配置寄存器)

- Address = Base Address + 0x0040, Reset Value = 0x00FF_03FF

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

Name	Bit	Type	Description
EM_CNT	[9:0]	R/W	外部晶振的时钟稳定计数器。 该计数器值可以在 EMOSC 禁止时进行修改。EMOSC 使能时，时钟稳定计数器开始递减计数，当计数值达到零，RISR 状态寄存器中的 EMOSC_ST 位被置位。 时钟稳定计数器的计数时钟为外部时钟的256分频，所以在缺省状态下，当外部晶振为8MHz 时，稳定计数时间为： 0x3FF x 256 x 125ns = 32.7ms

7.3.12 SYSCON_LVDCR (低电压检测控制寄存器)

- Address = Base Address + 0x004C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LVD_KEY																LVDFLAG	RSTDET_LVL			LVD_INT	INTDET_LVL			FLTBP	RSVD			LVDEN			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
LVDEN	[3:0]	R/W	使能/禁止 LVD 模块。 0Ah：禁止 LVD 模块。 其他：使能 LVD 模块。
FLTBP	[7]	R/W	LVR 输出滤波使能控制 0：使能输出滤波。 1：禁止输出滤波。
INTDET_LVL	[10:8]	R/W	LVD 中断触发检测电平 **INTDET_LVL[2:0]** / **VDD(v)** x00 / 2.55 x01 / 3.00 x10 / 3.90 x11 / 4.10
LVD_INT	[11]	R/W	使能/禁止 LVD 中断功能。 0：LVD 中断禁止。 1：LVD 中断使能。 在需要产生 LVD 中断时，还必须使能中断使能寄存器 IECR 中的 LVD_INT 位。
RSTDET_LVL	[14:12]	R/W	LVD 复位触发检测电平 **RSTDET_LVL[2:0]** / **VDD(v)** x00 / 2.15 x01 / 2.75 x10 / 3.35 x11 / 3.65

LVDFLAG	[15]	R/W	LVD 的检测状态。 0：VDD 的当前电压高于 INTDET_LVL 设置的检测阈值。 1：VDD 的当前电压低于 INTDET_LVL 设置的检测阈值。
LVD_KEY	[31:16]	R/W	对本寄存器进行写操作时，需要填入对应的 KEY 值。 只有在 ID_KEY 等于0xB44B 时，对本寄存器的写入才有效。

NOTE:

由 LVD 模块产生的处理器复位，不会复位 LVD 控制寄存器。

7.3.13 SYSCON_CLCR (CLO 配置寄存器)

- Address = Base Address + 0x0050, Reset Value = 0x0000_01FF

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

Name	Bit	Type	Description
CLOMX	[19:16]	R/W	CLO 输出选择。 **CLOMX** / **CLOCK** / **CLOMX** / **CLOCK** 0x0 / ISCLK / 0x8 / FRNCLK 0x1 / IMCLK / 0x9 / DAPCLK 0x2 / RSVD / 0xA / CPUCLK 0x3 / EMCLK / 0xB / AHBCLK 0x4 / TKEYCLK / 0xC / APBCLK 0x5 / TKEYCLK_DV / Others / RSVD 0x6 / IWDTCLK 0x7 / SYSCLK
HFOSCEN	[20]	R/W	内部高频振荡器使能控制。 0：禁止内部高频振荡器 1：使能内部高频振荡器 该振荡器输出96MHz 时钟 HFCLK，专门供给 TC0使用，其它模块无法使用。
HFO_ST	[21]	R	内部高频振荡器状态位。 0：振荡器时钟未稳定 1：振荡器时钟已稳定
IMO_FSEL	[22]	R/W	内部主振荡器频率选择。(1) 0：振荡器时钟输出为20MHz 1：振荡器时钟输出为40MHz

NOTE: 1）对内部主振荡器的频率切换，必须在该振荡器禁止时进行。当系统使用 IMOSC 工作时，先将系统工作时钟切换到其他时钟源（例如，ISOSC），然后禁止 IMOSC，切换 IMOSC 频率后，再使能 IMOSC，最后重新选择系统工作时钟到 IMOSC。

7.3.14 SYSCON_PWRCR (功耗控制寄存器)

- Address = Base Address + 0x0054, Reset Value = 0x0000_1F09

31	30	29	28	27	26	25	24	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																WKUPTIM1		WKUPTIM0		CYO_CD	CYO_GM			RUNCFG			SLPCFG			DSLPCFG	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	1	0	1	1
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
DSLPCFG	[1:0]	R/W	在 DEEP-SLEEP 模式下功耗控制。(1) 00b：内部精准参考源使能，REG 低驱动能力（较高功耗）。 01b：内部精准参考源使能，REG 极低驱动能力。 10b：内部精准参考源关闭，REG 低驱动能力。 11b：内部精准参考源关闭，REG 极低驱动能力（最低功耗）。
SLPCFG	[4:2]	R/W	在 SLEEP 模式下功耗控制。(1) 00xb：内部精准参考源使能，REG 正常驱动能力。 010b：内部精准参考源使能，REG 低驱动能力。 011b：内部精准参考源使能，REG 极低驱动能力。 10xb：内部精准参考源关闭，REG 正常驱动能力。 110b：内部精准参考源关闭，REG 低驱动能力。 111b：内部精准参考源关闭，REG 极低驱动能力。
RUNCFG	[7:5]	R/W	在 RUN 模式下功耗控制 00xb：内部精准参考源使能，REG 正常驱动能力。 010b：内部精准参考源使能，REG 低驱动能力。 011b：内部精准参考源使能，REG 极低驱动能力。 10xb：内部精准参考源关闭，REG 正常驱动能力。 110b：内部精准参考源关闭，REG 低驱动能力。 111b：内部精准参考源关闭，REG 极低驱动能力。
CYO_GM	[10:8]	R/W	外部晶振增益调整。 高频: 111 低频: 000 可根据实际外围电路起振情况调整。
CYO_CD	[11]	R/W	外部晶振驱动调整。 高频: 1 低频: 0

			可根据实际外围电路起振情况调整。 使用32KHz 晶振时，该位必须是0才能起振。
WKUPTIM0	[13:12]	R/W	由 DEEP-SLEEP 唤醒时，内部主晶振控制下的系统稳定时间。 Tstable = Timoclk x CNTst，Timoclk 为 IMCLK 的低频周期。 CNTst 的设置值如下： 00: 0x0040 01: 0x0100 10: 0x0400 11: 0x1000
WKUPTIM1	[15:14]	R/W	由 SLEEP 唤醒时，系统稳定时间设置。当 SLEEP 模式切换到 NORM 模式时，发生参考源变化或者驱动变化时，有必要适当增加系统稳定时间。 Tstable = Timoclk x CNTst，Timoclk 为 IMCLK 的低频周期。 CNTst 的设置值如下： 00: 0x0F 01: 0x5F 10: 0xBF 11: 0xFF

NOTE: 1）在低功耗模式下，不允许切换模式配置。改变模式配置，必须在 RUN 模式下进行。

2）对该寄存器操作时，必须同时写入相应的KEY值，KEY值为：0xA66A

7.3.15 SYSCON_OPT4 (系统配置寄存器4)

- Address = Base Address + 0x0058, Reset Value = 0x0000_XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																	OPA1_OSTR							RSVD	OPA0_OSTR						
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
OPA0_OSTR	[6:0]	R/W	OPA0的输入失调电压调整。 在上电时，FLASH 内的工厂校准值会被自动加载到此寄存器中。通过调整此寄存器，程序可以再次校准运放的输入失调电压。
OPA1_OSTR	[14:8]	R/W	OPA1的输入失调电压调整。 在上电时，FLASH 内的工厂校准值会被自动加载到此寄存器中。通过调整此寄存器，程序可以再次校准运放的输入失调电压。

7.3.16 SYSCON_OPT3 (系统配置寄存器3)

- Address = Base Address + 0x005C, Reset Value = 0x0000_XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																	CMP4_OSTR							RSVD	CMP3_OSTR						
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CMP3_OSTR	[6:0]	R/W	CMP3的输入失调电压调整。 在上电时，FLASH 内的工厂校准值会被自动加载到此寄存器中。通过调整此寄存器，程序可以再次校准比较器的输入失调电压。
CMP4_OSTR	[14:8]	R/W	CMP4的输入失调电压调整。 在上电时，FLASH 内的工厂校准值会被自动加载到此寄存器中。通过调整此寄存器，程序可以再次校准比较器的输入失调电压。

7.3.17 SYSCON_OPT2 (系统配置寄存器2)

- Address = Base Address + 0x0060, Reset Value = 0x00XX_XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									CMP2_OSTR							RSVD				CMP1_OSTR				RSVD				CMP0_OSTR			
R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CMP0_OSTR	[6:0]	R/W	CMP0的输入失调电压调整。 在上电时，FLASH 内的工厂校准值会被自动加载到此寄存器中。通过调整此寄存器，程序可以再次校准比较器的输入失调电压。
CMP1_OSTR	[14:8]	R/W	CMP1的输入失调电压调整。 在上电时，FLASH 内的工厂校准值会被自动加载到此寄存器中。通过调整此寄存器，程序可以再次校准比较器的输入失调电压。
CMP2_OSTR	[22:16]	R/W	CMP2的输入失调电压调整。 在上电时，FLASH 内的工厂校准值会被自动加载到此寄存器中。通过调整此寄存器，程序可以再次校准比较器的输入失调电压。

7.3.18 SYSCON_OPT1 (系统配置寄存器1)

- Address = Base Address + 0x0064, Reset Value = 0xXXXX_XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HFIMO_TRM1								HFIMO_TRM0								NFIMO_TRM1								NFIMO_TRM0							
R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
NFIMO_TRM0	[7:0]	R/W	低频 IMOSC（20MHz）的输出频率调整。 在上电时，FLASH 内的工厂校准值会被自动加载到此寄存器中。通过调整此寄存器，程序可以再次校准 IMOSC 的输出。
NFIMO_TRM1	[15:8]	R/W	高频 IMOSC（40MHz）的输出频率调整。 在上电时，FLASH 内的工厂校准值会被自动加载到此寄存器中。通过调整此寄存器，程序可以再次校准 IMOSC 的输出。
HFIMO_TRM0	[23:16]	R/W	超高频 IMOSC（48MHz）的输出频率调整。 在上电时，FLASH 内的工厂校准值会被自动加载到此寄存器中。通过调整此寄存器，程序可以再次校准 IMOSC 的输出。
HFIMO_TRM0	[31:24]	R/W	超高频 IMOSC（96MHz）的输出频率调整。 在上电时，FLASH 内的工厂校准值会被自动加载到此寄存器中。通过调整此寄存器，程序可以再次校准 IMOSC 的输出。

7.3.19 SYSCON_OPT0 (系统配置寄存器0)

- Address = Base Address + 0x0068, Reset Value = 0xXXXX_XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				RDP	RSVD							HDP_4K	HDP_2K	HDP_1K	HDP_ALL	RSVD							SWDP	RSVD						EXTRST	IWDT_EN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
IWDT_EN	[0]	R	看门狗模块缺省状态。 0：缺省关闭看门狗。 1：缺省使能看门狗。
EXTRST	[1]	R	外部复位管脚使能状态。 0：外部复位管脚禁用。 1：外部复位管脚使能。
SWDP	[8]	R	SWD 调试接口保护状态。 0：保护未使能（可以通过 SWD 连接） 1：保护使能（不能通过 SWD 连接）
HDP_ALL	[16]	R	Hardware Protection（Flash Erase/Write）。 0：未设置保护。 1：启动全区域保护（禁止对 Flash 程序区的页擦除和写操作）。
HDP_1K	[17]	R	Hardware Protection（Flash Erase/Write）。 0：未设置保护。 1：启动 Flash 程序区起始1K 地址区间保护。
HDP_2K	[18]	R	Hardware Protection（Flash Erase/Write）。 0：未设置保护。 1：启动 Flash 程序区起始2K 地址区间保护。
HDP_4K	[19]	R	Hardware Protection（Flash Erase/Write）。 0：未设置保护。 1：启动 Flash 程序区起始4K 地址区间保护。
RDP	[27]	R	Flash 读保护状态。 0：未设置保护。 1：Flash 内容不能通过外部烧写器读取。

7.3.20 SYSCON_IECR/IEDR/IAR/ICR (中断使能/禁止/软件触发/清除寄存器)

- IECR：Address = Base Address + 0x0074, Reset Value = 0x0000_0000
- IEDR：Address = Base Address + 0x0078, Reset Value = 0x0000_0000
- IAR：Address = Base Address + 0x0080, Reset Value = 0x0000_0000
- ICR：Address = Base Address + 0x0084, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		CMD_ERR	RSVD									EM_CMRCV	EM_CMFAIL	RSVD						LVD_INT	RSVD		IWDT_INT	SYSCLK_ST	RSVD			EMOSC_ST	RSVD	IMOSC_ST	ISOSC_ST
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	W	R	R	R	R	R	R	R	R	R	W	W	R	R	R	R	R	R	W	R	R	W	W	R	R	R	W	R	W	W

Name	Bit	Type	Description
ISOSC_ST	[0]	W	ISOSC 时钟稳定中断。
IMOSC_ST	[1]	W	IMOSC 时钟稳定中断。
EMOSC_ST	[3]	W	EMOSC 时钟稳定中断。
SYSCLK_ST	[7]	W	系统工作时钟稳定中断。
IWDT_INT	[8]	W	看门狗报警中断。
LVD_INT	[11]	W	低电压检测中断。
EM_CMFAIL	[18]	W	外部时钟失效中断。
EM_CMRCV	[19]	W	外部时钟失效恢复中断。
CMD_ERR	[29]	W	命令错误中断。

NOTE:

1） 寄存器只有在写入‘1’时有效，写入‘0’时无效。

2） IECR 在对应位写入‘1’时，对应中断使能；IEDR 在对应位写入‘1’时，对应中断禁止。

3） IAR 在对应位写入‘1’时，对应中断被触发。

4） ICR 在对应位写入‘1’时，对应中断状态标示被清除。

7.3.21 SYSCON_IMSR/RISR/ISR (中断使能/禁止状态/原始标志状态/标志状态寄存器)

- IMSR：Address = Base Address + 0x007C, Reset Value = 0x0000_0000
- RISR：Address = Base Address + 0x0088, Reset Value = 0x0000_0000
- ISR：Address = Base Address + 0x008C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		CMD_ERR	RSVD									EM_CMRCV	EM_CMFAIL	RSVD						LVD_INT	RSVD		IWDT_INT	SYSCLK_ST	RSVD			EMOSC_ST	RSVD	IMOSC_ST	ISOSC_ST
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
ISOSC_ST	[0]	R	ISOSC 时钟稳定中断。
IMOSC_ST	[1]	R	IMOSC 时钟稳定中断。
EMOSC_ST	[3]	R	EMOSC 时钟稳定中断。
SYSCLK_ST	[7]	R	系统工作时钟稳定中断。
IWDT_INT	[8]	R	看门狗报警中断。
LVD_INT	[11]	R	低电压检测中断。
EM_CMFAIL	[18]	R	外部时钟失效中断。
EM_CMRCV	[19]	R	外部时钟失效恢复中断。
CMD_ERR	[29]	R	命令错误中断。
标志状态说明： 0：中断禁止，或者中断未发生。 1：中断使能，或者中断请求发生。			

NOTE:

1） IMSR 寄存器表示中断的使能或者禁止状态，可以通过 IECR 和 IDCR 来设置。

2） RISR 是原始中断标志位，一旦有中断发生，无论该是否在 IECR 中使能，RISR 中相应位都会置位。通过 ICR 清除。

3） ISR 是中断标志位，逻辑上只有在 IECR 中使能的中断，在 RISR 相应位置位时，对应 ISR 中的相应位才会置位。

7.3.22 SYSCON_RSR (复位记录状态寄存器)

- Address = Base Address + 0x0090, Reset Value = 0xXXXX_XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																							SWRST	CPURST	EM_CMRST	RSVD	IWDTRST	RSVD	EXTRST	LVRST	POR
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	R	RW	R	RW	RW	RW

Name	Bit	Type	Description
POR	[0]	R/W	处理器上电复位
LVRST	[1]	R/W	低电压检测复位
EXTRST	[2]	R/W	外部管脚复位
IWDTRST	[4]	R/W	看门狗复位
EM_CMRST	[6]	R/W	外部时钟失效复位
CPURST	[7]	R/W	CPU 复位请求
SWRST	[8]	R/W	软件复位（通过系统控制器的 IDCCR）

NOTE:

对相应位写‘1’来清除。

7.3.23 SYSCON_EXIRT/EXIFT（外部中断上升沿/下降沿选择寄存器）

- EXIRT：Address = Base Address + 0x0094, Reset Value = 0x0000_0000
- EXIFT：Address = Base Address + 0x0098, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																EXI15	EXI14	EXI13	EXI12	EXI11	EXI10	EXI9	EXI8	EXI7	EXI6	EXI5	EXI4	EXI3	EXI2	EXI1	EXI0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
EXI0 ~ EXI15	[15:0]	R/W	外部中断上升沿/下降沿使能。 0：上升沿（EXIRT）/ 下降沿（EXIFT）未使能。 1：上升沿（EXIRT）/ 下降沿（EXIFT）使能。

NOTE:

1） EXIRT 是上升沿选择寄存器。

2） EXIFT 是下降沿选择寄存器。

3） 当 EXIRT 或者 EXIFT 中对应位选择使能时，对应外部中断线由上升沿，或者下降沿触发；当 EXIRT 和 EXIFT 中对应位都使能时，对应外部中断线为双边沿触发。

7.3.24 SYSCON_EXIER/EXIDR/EXIAR/EXICR(EXIMSR)（外部中断使能/禁止/软件触发/清除寄存器）

- EXIER：Address = Base Address + 0x009C, Reset Value = 0x0000_0000
- EXIDR：Address = Base Address + 0x00A0, Reset Value = 0x0000_0000
- EXIAR：Address = Base Address + 0x00A8, Reset Value = 0x0000_0000
- EXICR：Address = Base Address + 0x00AC, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																EXI15	EXI14	EXI13	EXI12	EXI11	EXI10	EXI9	EXI8	EXI7	EXI6	EXI5	EXI4	EXI3	EXI2	EXI1	EXI0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
EXI0 ~ EXI15	[15:0]	R/W	IER/IDR：使能/禁止外部中断 IAR/ICR：软件触发/清除外部中断

NOTE:

1） 寄存器只有在写入‘1’时有效，写入‘0’时无效。

2） EXIER 在对应位写入‘1’时，对应外部中断使能；EXIDR 在对应位写入‘1’时，对应外部中断禁止。

3） EXIAR 在对应位写入‘1’时，对应外部中断被触发。

4） EXICR 在对应位写入‘1’时，对应外部中断状态标示被清除。改寄存器在读取时，返回 Masked Interrupt Status 的状态。

7.3.25 SYSCON_EXIMR/EXIRS（外部中断使能/禁止状态/原始状态寄存器）

- EXIMR：Address = Base Address + 0x00A4, Reset Value = 0x0000_0000
- EXIRS：Address = Base Address + 0x00B0, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																EXI15	EXI14	EXI13	EXI12	EXI11	EXI10	EXI9	EXI8	EXI7	EXI6	EXI5	EXI4	EXI3	EXI2	EXI1	EXI0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
EXI0 ~ EXI15	[15:0]	R	EXIMR：外部中断使能、禁止状态寄存器。 0：该外部中断被禁止。 1：该外部中断被使能。 EXIRS：外部中断原始标志位 0：该中断未发生。 1：该中断发生。

7.3.26 SYSCON_IWDCR（看门狗控制寄存器）

- Address = Base Address + 0x00B4, Reset Value = 0x0000_070C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWDT_KEY																RSVD			IWDT_BUSY	RSVD	IWDT_TIME			IWDT_INTW						RSVD	IWDT_SHT
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	1	0	0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	RW

Name	Bit	Type	Description
IWDT_SHT	[0]	R/W	IWDT 的 SHORT 模式，在 debug 时使用。 0：禁止 SHORT 模式。 1：使能 SHORT 模式。
IWDT_INTW	[7:2]	R/W	看门狗的报警中断窗口时间。当看门狗计时到总溢出时间一定比例时，发生报警中断。 **IWDT_INTW** / **比例** 0x0 / IWDT溢出时间的1/8 0x1 / IWDT溢出时间的2/8 0x2 / IWDT溢出时间的3/8 0x3 / IWDT溢出时间的4/8 0x4 / IWDT溢出时间的5/8 0x5 / IWDT溢出时间的6/8 0x6 / IWDT溢出时间的7/8 Others / IWDT溢出时间的7/8 例如：当看门狗溢出时间设置为8S 时（IWDT_TIME=7），若设置 IWDT_INTW 为0，则中断时间为1S
IWDT_TIME	[10:8]	R/W	看门狗的总溢出时间。 **IWDT_TIME** / **溢出时间** 0x0 / 0.128 秒 0x1 / 0.256 秒 0x2 / 0.5 秒 0x3 / 1 秒 0x4 / 2 秒

<table>
<tr><td></td><td></td><td></td><td><table>
<tr><td>0x5</td><td>3 秒</td></tr>
<tr><td>0x6</td><td>4 秒</td></tr>
<tr><td>0x7</td><td>8 秒</td></tr>
</table></td></tr>
<tr><td>IWDT_BUSY</td><td>[12]</td><td>R</td><td>看门狗的工作状态。
0：看门狗未工作。
1：看门狗工作。</td></tr>
<tr><td>IWDT_KEY</td><td>[31:16]</td><td>W</td><td>对本寄存器进行写操作时，需要填入对应的 KEY 值。
只有在 IWDT_KEY 等于0x8778时，对本寄存器的写入才有效。</td></tr>
</table>

NOTE:

在看门狗工作时，任何尝试关闭 ISOSC 的操作将导致命令错误中断发生。

7.3.27 SYSCON_IWDCNT（看门狗控制计数器值）

- Address = Base Address + 0x00B8, Reset Value = 0x0003_FFFF

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

Name	Bit	Type	Description
IWDT_CNT	[17:0]	R	IWDT 的当前计数值。
IWDT_CLR	[30:24]	W	看门狗计时器清除请求。当写入‘0x5A’时有效。
CLR_BUSY	[31]	R	清除请求状态位。 0：没有挂起的清除请求。 1：正在清除中。

7.3.28 SYSCON_IWDEDR（看门狗使能寄存器）

- Address = Base Address + 0x00BC, Reset Value = 0x0000_XXXX

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

Name	Bit	Type	Description
IWDT_EDC	[15:0]	R/W	IWDT 的使能控制。 0x5555：关闭看门狗。 Others: 启动看门狗。 这个寄存器在处理器复位以后，只能写一次。
IWDTE_KEY	[31:16]	W	对本寄存器进行写操作时，需要填入对应的 KEY 值。 只有在 IWDTE_KEY 等于0x7887时，对本寄存器的写入才有效。

7.3.29 SYSCON_CINF0/CINF1/FINF0/FINF1（客户信息区、工程信息区）

- CINF0：Address = Base Address + 0x00C0, Reset Value = 0xXXXX_XXXX
- CINF1：Address = Base Address + 0x00C4, Reset Value = 0xXXXX_XXXX
- FINF0：Address = Base Address + 0x00C8, Reset Value = 0xXXXX_XXXX
- FINF1：Address = Base Address + 0x00CC, Reset Value = 0xXXXX_XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CINF0 CINF1 FINF0 FINF1																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
CINF0 CINF1 FINF0 FINF1	[31:0]	R	通过烧写器写入客户信息区的第一和第二个 word 内容，在上电时，自动映射到 CINF 寄存器中。 工程信息区的信息由芯片制造商在出厂前写入。不能修改。

7.3.30 SYSCON_ERRINF（错误命令信息查询寄存器）

- Address = Base Address + 0x00E0, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																ER_IWDCNT	ER_CMEN			ER_CMDIS			ER_EMDIS			ER_ISODIS	ER_AHBCLK				ER_PWRCFG
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
ER_PWRCFG	[0]	R	在低功耗模式下，尝试切换功耗配置(1)
ER_AHBCLK	[4]	R	尝试切换系统时钟到一个未稳定的时钟源
ER_ISODIS	[5]	R	IWDT 使能时，尝试关闭 ISOSC
ER_EMDIS	[8]	R	当外部时钟监测使能时，尝试关闭 EMOSC
ER_CMDIS	[11]	R	当外部时钟未稳定，或者未使能时，尝试关闭外部时钟监测。
ER_CMEN	[14]	R	当外部时钟未稳定，或者未使能时，尝试启动外部时钟监测。
ER_IWDCNT	[15]	R	ISOSC 未使能或者未稳定，尝试修改 IWDT 的计数器（清除操作）

NOTE: 1）低功耗模式包括 DEEP-SLEEP 模式和 SLEEP 模式。

8 模数转换器（ADC)

8.1 概述

本章节描述ADC控制器的功能，从用户的角度详细说明如何操作ADC。

8.1.1 主要特性

12位模数转换器(ADC)模块使用一个逐次逼近电路将模拟电平转换为一个12位的数字值。输入的模拟电平值必须在AVREF和AVSS的值之间。

- 带逐次逼近逻辑的模拟比较器
- 参考电压(AVREF)支持选择内部或者外部
- 自带固定电压参考源
- 支持多路外部模拟输入AIN[17:0]，内部固定电压参考源输入，以及1/4VDD输入
- 支持多序列转换模式，可灵活配置转换通道，转换顺序，转换次数
- 每个转换序列都有一个12位转换结果寄存器(ADC_DR)
- 支持多个外部触发源，可以触发转换序列
- 最大转换速度: 1MSPS
- 模拟输入范围：AVSS 到 AVREF
- 可配置采样时间，并且可以配置成10位的ADC以加快转换速度

8.1.2 管脚描述

Table 8-1 ADC管脚描述

管脚名称	功能	I/O类型	有效电平	说明
AVREF	模拟参考电压	模拟	-	-
AIN0 to AIN17	模拟信号输入	模拟	-	-

8.1.3 模块框图

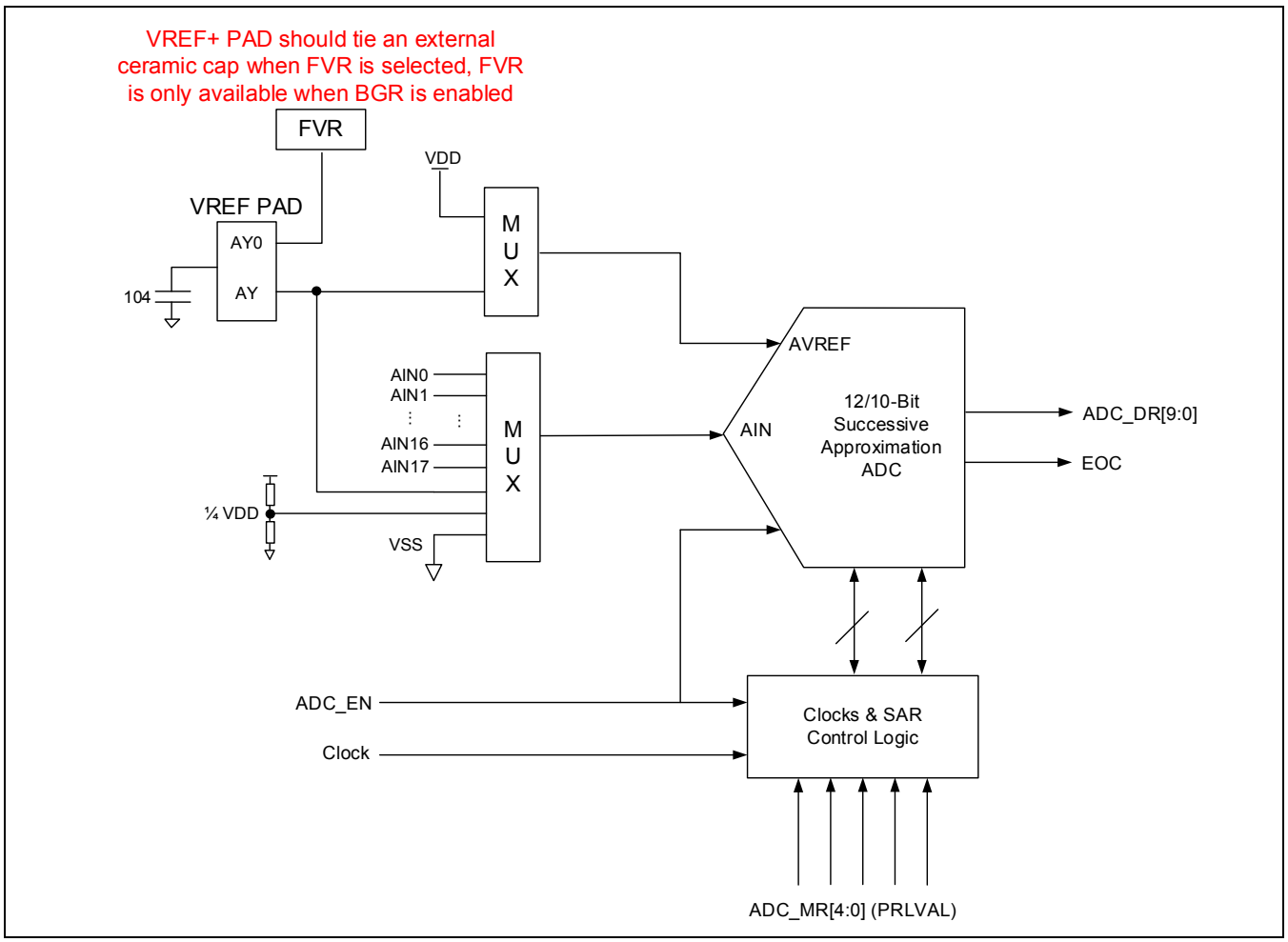


Figure 8-1 ADC模块框图

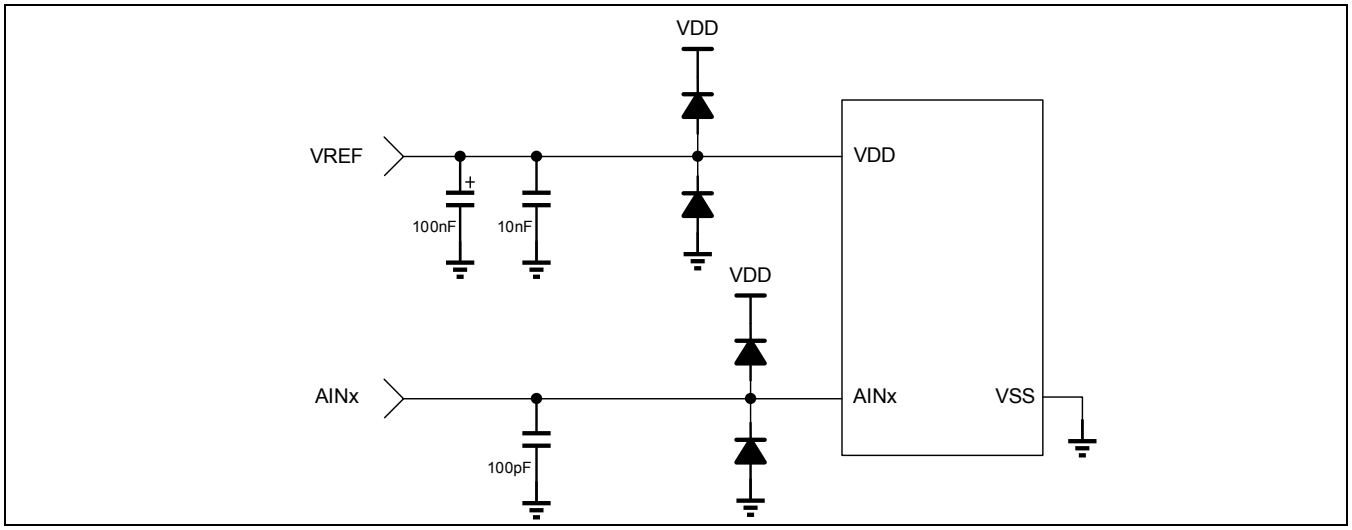


Figure 8-2 参考电路

8.1.4 输入和输出

ADC的功能是将通过AIN输入的模拟信号转换成数字值。有效的输入信号如下。电压范围从0V到电源电压。

Input Voltage Range: 0.0 V ~ 5.0 V

Reference Bottom Voltage: 0.0 V

Reference Top Voltage: 5.0 V

$$Dout = \frac{V_{IN}}{V_{FS}} = \frac{b_{N-1}}{2} + \frac{b_{N-2}}{2^2} + ... + \frac{b_0}{2^N}$$

$$1\ LSB = \frac{Reference\ Top - Reference\ Bottom}{2^{Resolution}} = \frac{5.0V - 0.0V}{2^{12}} = \frac{5.0V}{4096} \approx 1.22mV \quad (12\text{-bit})$$

$$1\ LSB = \frac{Reference\ Top - Reference\ Bottom}{2^{Resolution}} = \frac{5.0V - 0.0V}{2^{10}} = \frac{5.0V}{1024} \approx 4.88mV \quad (10\text{-bit})$$

Table 8-2 12位模式的输入和输出范围 (VREF = 5V)

Index	AINx Input Voltage (V)	Digital Output (Binary)	Digital Output (Hex)
0	0.00000 to 0.00122	0000 0000 0000	0x000
1	0.00122 to 0.00244	0000 0000 0001	0x001
…	…	…	…
2047	2.49878 to 2.50000	0111 1111 1111	0x7FF
2048	2.50000 to 2.50122	1000 0000 0000	0x800
…	…	…	…
4094	4.99756 to 4.99878	1111 1111 1110	0xFFE
4095	4.99878 to 5.00000	1111 1111 1111	0xFFF

Table 8-3 10位模式的输入和输出范围 (VREF = 5V)

Index	AINx Input Voltage (V)	Digital Output (Binary)	Digital Output (Hex)
0	0.00000 to 0.00488	00 0000 0000	0x000
1	0.00488 to 0.00976	00 0000 0001	0x001
…	…	…	…
511	2.49512 to 2.50000	01 1111 1111	0x1FF
512	2.50000 to 2.50488	10 0000 0000	0x200
…	…	…	…
1022	4.99023 to 4.99512	11 1111 1110	0x3FE
1023	4.99512 to 5.00000	11 1111 1111	0x3FF

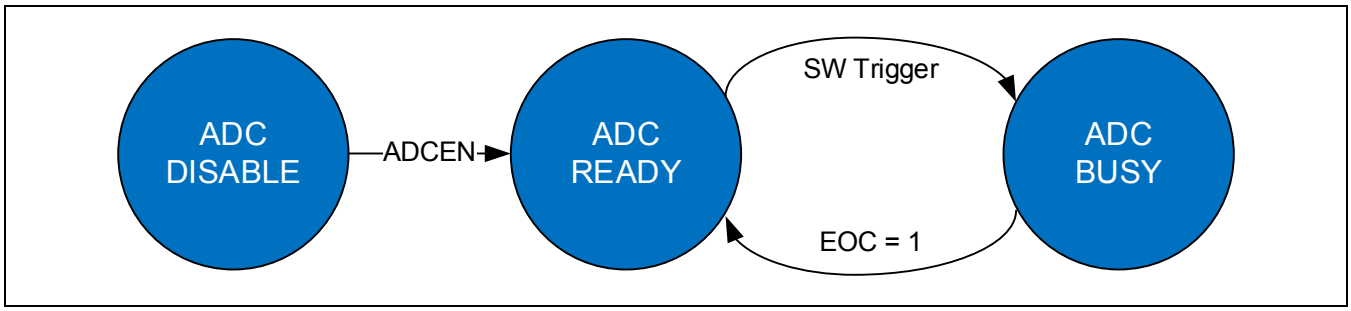


Figure 8-3 ADC状态机

8.1.5 参考电压源(AVREF)选择

ADC的参考电压源支持选择内部(VDD)或者外部(VREF)，由ADC_CR寄存器中的VREF位控制。如果该位为0(默认)，ADC的参考电压源为芯片的电源VDD；如果该位为1，则ADC的电压参考源可以由外部VREF管脚提供，也可以由一个固定电压源(FVR)模块提供。如果希望使用固定电压源提供参考电压，则必须使用ADC_CR中的FVR_EN位打开电压基准源模块。电压基准源模块提供两个不会随着芯片电源VDD变化的固定电压2.048V和4.096V，可以用过ADC_CR中的FVR_LVL位进行选择。

注意，如果使用FVR作为参考，需要在VREF管脚上增加一个0.1uF的电容，参考 Figure 8-1 ADC模块框图。

8.1.6 时钟频率和转换时间

ADC 工作的时钟是从 PCLK 获得的。AD 转换的过程需要总共(setup+12/10)个时钟周期。setup 时间可以通过 ADC_SEQx 寄存器里的“SAMPLE”位(ADC_SEQx[7:6])设置。ADC 模块提供一个时钟分频器，该分频器是一个 6 位计数器，由模式寄存器里的 PRLVAL 控制。 下面的表达式给出了系统频率和 ADC 模拟模块时钟频率之间的关系。

如果 PRLVAL 是 0，那么 F_ANA = PCLK

否则 PRLVAL 是其它任何值的话，F_ANA = PCLK / (2*PRLVAL)

PRLVAL 的值必须保证采样速度不超过手册规定的最大值(1MSPS)。如果 PCLK 频率是 20MHz，并且 PCLK/2 被选择位转换时钟，那么一个时钟周期就是 100ns。转换速度计算如下(假设 setup 时间为默认值 6 个周期)：

12 位 – (6 个 setup 时钟周期) + (每位 1 个时钟转换周期 x 12 位) + (3 个同步和结果处理时钟周期) = 21 个周期

21 x 100ns = 2.1us (476ksps)

10 位 – (6 个 setup 时钟周期) + (每位 1 个时钟转换周期 x 10 位) + (3 个同步和结果处理时钟周期) = 19 个周期

19 x 100ns = 1.9us (526ksps)

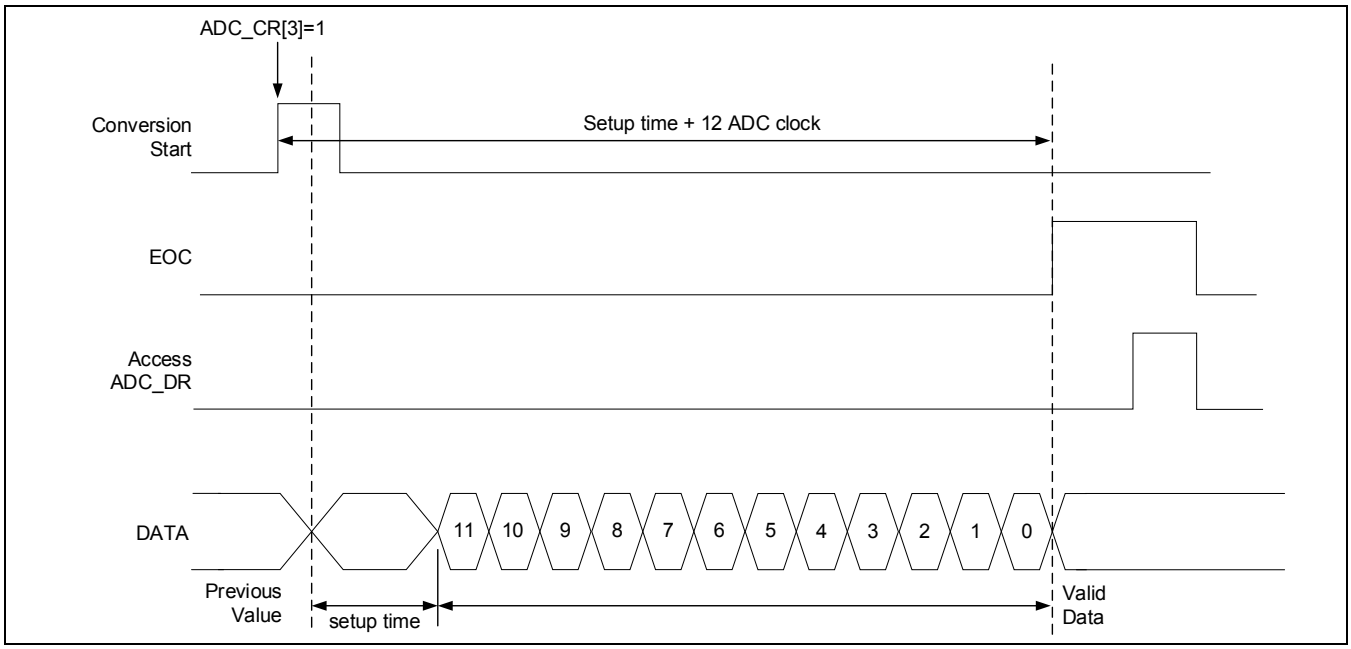


Figure 8-4 ADC工作时序图

8.1.7 转换序列定义

转换序列是指若干个需要转换的模拟输入的组合。用户可以设置ADC模块以任意顺序转换18个通道中的某几个选择好的输入信号。序列的长度(个数)可以由ADC_MR(ADC模式寄存器)中的NBRCH位定义。下表列出了NBRCH和转换次数的关系：

Table 8-4 NBRCH[3:0]的值和转换序列个数

NBRCH[3:0]	转换序列个数
0000	1
0001	2
0010	3
0011	4
0100	5
0101	6
0110	7
0111	8
1000	9
1001	10
1010	11
1011	12

要注意的是，即使是在单次转换(one shot)模式下，ADC 也会在启动后转换设置好的转换序列。12 个转换结果寄存器会保存每个序列的转换结果供读取。

序列中转换的通道由 ADC_SEQx 寄存器设定。下表列出了 ADC_SEQx 中 AIN_SEL 值和输入通道选择的关系：

Table 8-5 AIN_SEL值和输入选择通道

AIN_SEL值	选择的输入通道	选择的管脚
0_0000	Input 0	AIN0
0_0001	Input 1	AIN1
0_0010	Input 2	AIN2
0_0011	Input 3	AIN3
0_0100	Input 4	AIN4
0_0101	Input 5	AIN5
0_0110	Input 6	AIN6
0_0111	Input 7	AIN7
0_1000	Input 8	AIN8
0_1001	Input 9	AIN9
0_1010	Input 10	AIN10
0_1011	Input 11	AIN11
0_1100	Input 12	AIN12
0_1101	Input 13	AIN13
0_1110	Input 14	AIN14
0_1111	Input 15	AIN15
1_0000	Input 16	AIN16
1_0001	Input 17	AIN17
…	No Input (input floating)	N/A
1_1100	Input 28	FVR
1_1101	Input 29	¼ VDD
1_1110	Input 30	VSS
1_1111	No Input (input floating)	N/A

例如，假设：

NBRCH = 0x2,

ADC_SEQ0.AIN_SEL = 0x5, ADC_SEQ1.AIN_SEL = 0x2 and ADC_SEQ2.AIN_SEL = 0x0

在转换开始后，ADC 先转换输入通道 5(AIN5)，然后转换通道 2(AIN2)，最后再以转换通道 0(AIN0)结束。

8.1.8 单次转换或者连续转换模式

ADC 可以配置成两种模式：单次转换模式和连续转换模式。

将模式寄存器中的 CONTCV 位设 0 为单次转换模式。这个模式下，转换开始后，ADC 只进行一次完整的(序列)转换，之后就停止并且等待下一个开始转换的请求。

在序列转换完成前，ADC 不可以被停止。

将模式寄存器中的 CONTCV 位设 1 则为连续转换模式。这个模式下，转换开始后，ADC 不停的循环转换(序列)，从序列 0 到序列 11 循环，直到被停止。要停止转换，CPU 必须将控制寄存器中的 STOP 位写 1。

当收到停止的请求后，ADC 会完成当前的转换，并且将转换结果寄存器更新为最后一次转换的结果。即使序列中其它转换没有完成，ADC 也会立即停止不会再进行其它转换。

用户必须注意，因为在连续转换模式中的停止命令不会让 ADC 立即停止，而是要完成当前进行中的转换，所以可能看起来像是多转换了一次。

当前正在转换的序列号可以由 ADC_SR 中的 SEQ_INDEX 位查看。

8.1.9 重复转换和平均值计算

在某一个转换序列中，可以设置 ADC 重复转换的次数(重复采样)，并且可以计算多次采样的平均值。这个功能由 ADC_SEQx 寄存器中的 CV_CNT 位和 AVG_CAL 位实现。

Table 8-6 CV_CNT[2:0]的值和重复采样次数

CV_CNT[2:0]	重复采样次数
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

如果使能计算平均值功能(AVG_CAL=1)，那么 ADC_DRx 寄存器将会保存多次转换的平均值，否则 ADC_DRx 保存的是最后一次转换的结果。

例如，在 ADC_SEQ0 中设置 CV_CNT=3'b011，AVG_CAL=1，那么该 SEQ0 序列中，ADC 会转换 8 次，假设转换结果为 DATA0~DATA7，在转换结束后，ADC_DR0 的值为 (DATA0+DATA1+…+DATA7)/8。如果设置 AVG_CAL=0，那么 SEQ0 序列转换结束后，ADC_DR0 的值为 DATA7。

8.1.10 ADC的比较功能

ADC 的比较功能可以让 ADC 在转换结果达到某个设定的值时触发一个中断。将 ADC_CMPx 设定为想要的阈值，一旦转换完成后，ADC 就会将转换结果和这个阈值比较，如果转换结果比 ADC_CMPx 寄存器的值大(电压高)，那么 CMPxH 状态位会被置 1，并且出发相应的中断；如果转换结果比 ADC_CMPx 寄存器的值小(电压低)，那么 CMPxL 状态位会被置 1，并且出发相应的中断。

在连续转换模式下，需要比较的转换可以通过 ADC_MR 寄存器中的 NBRCMPx 位设置。

在该 ADC 中，可以用来比较的阈值寄存器有两个：ADC_CMP0 和 ADC_CMP1。在连续转换模式下，也有两个相应的 NBRCMP0(ADC_MR[19:16])和 NBRCMP1(ADC_MR[25:22])寄存器。

Table 8-7 NBRCMPx[3:0]的值和需要比较的转换次数

NBRCMPx[3:0]	需要比较的转换次数
0000	1
0001	2

0010	3
0011	4
0100	5
0101	6
0110	7
0111	8
1000	9
…	…
1110	15
1111	16

例如，假设：

NBRCH = 0x4,

ADC_SEQ0.AIN_SEL = 0x5, ADC_SEQ1.AIN_SEL = 0x2,

ADC_SEQ2.AIN_SEL = 0x0, ADC_SEQ3.AIN_SEL = 0x5,

ADC_SEQ4.AIN_SEL = 0x2

NBRCMP0 = 0x1, NBRCMP1 = 0x3

ADC_CMP0 = 0x200, ADC_CMP1 = 0x700

(ADC_IER) CMP0H = 1, CMP1L = 1

那么 ADC 会进行 5 次转换。

1. ADC 将 SEQ1 (AIN2)的转换结果和 0x200 (ADC_CMP0)比较，如果结果大于 0x200，那么 CMP0H 中断产生。
2. ADC 将 SEQ3 (AIN5)的转换结果和 0x700 (ADC_CMP1)比较，如果结果小于 0x700，那么 CMP1L 中断产生。

8.1.11 ADC转换启动的触发源和触发优先级

ADC转换序列可以选择各种事件作为触发源，如下表格所示：

Table 8-8 TRG_SRC[2:0]的值和选择的触发源

TRG_SRC[2:0]	触发源
000	无触发
001	软件触发(ADC_CR中的SWTRG位)
010	TC1 脉冲匹配中断
011	EPWM触发 (EPWM模块里可以选择具体的触发源)
100	CMP 触发 (CMP 模块里可以选择具体的触发源)
101	外部管脚，上升沿
110	外部管脚，下降沿
111	外部管脚，上升和下降沿

ADC在使能触发(TRG_SRC不等于0)并且接收到该触发源后，会立即按预设的配置开始进行转换，也就是触发源和ADC_CR寄存器的开始转换位(START)功能一致。

ADC的触发功能还能设置延时，也就是在收到触发后，并不会马上开始ADC转换，而是延时一段时间，然后再开始转换，以避免转换到不想要的值。延时的时长在ADC_TDL0/1寄存器中设置。注意如果ADC_TDL0/1寄存器的值为0，那么触发延时功能为关闭状态，只有设置大于0的值，才会打开触发延时功能。

在连续转换模式下，如果转换序列选择的触发源产生了触发事件，那么该序列会被提升至下一个转换序列。下图为触发工作原理的示意图。

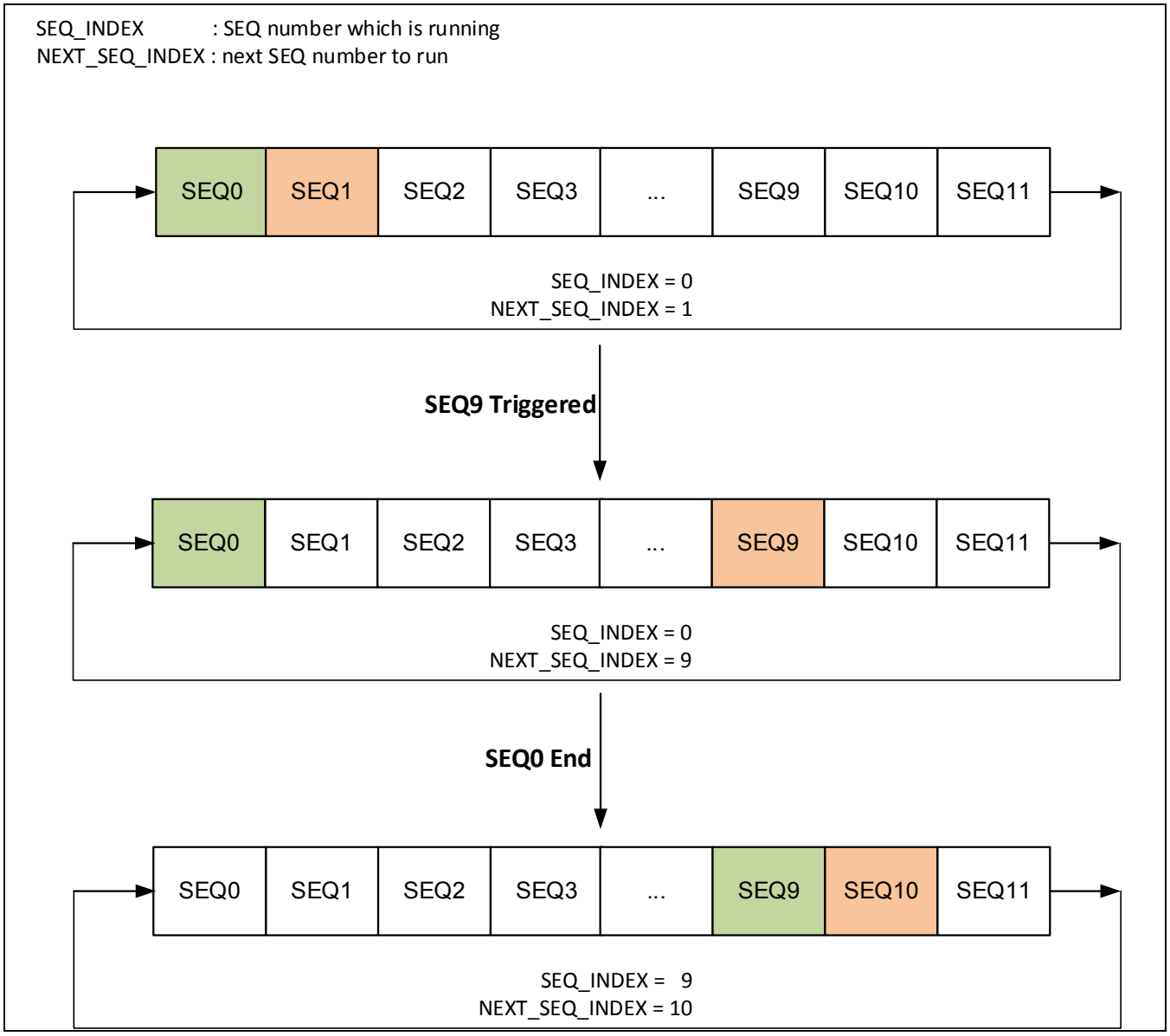


Figure 8-5 触发原理示意图

如果两个序列的触发事件同时产生，那么序列号低(小)的优先级高。

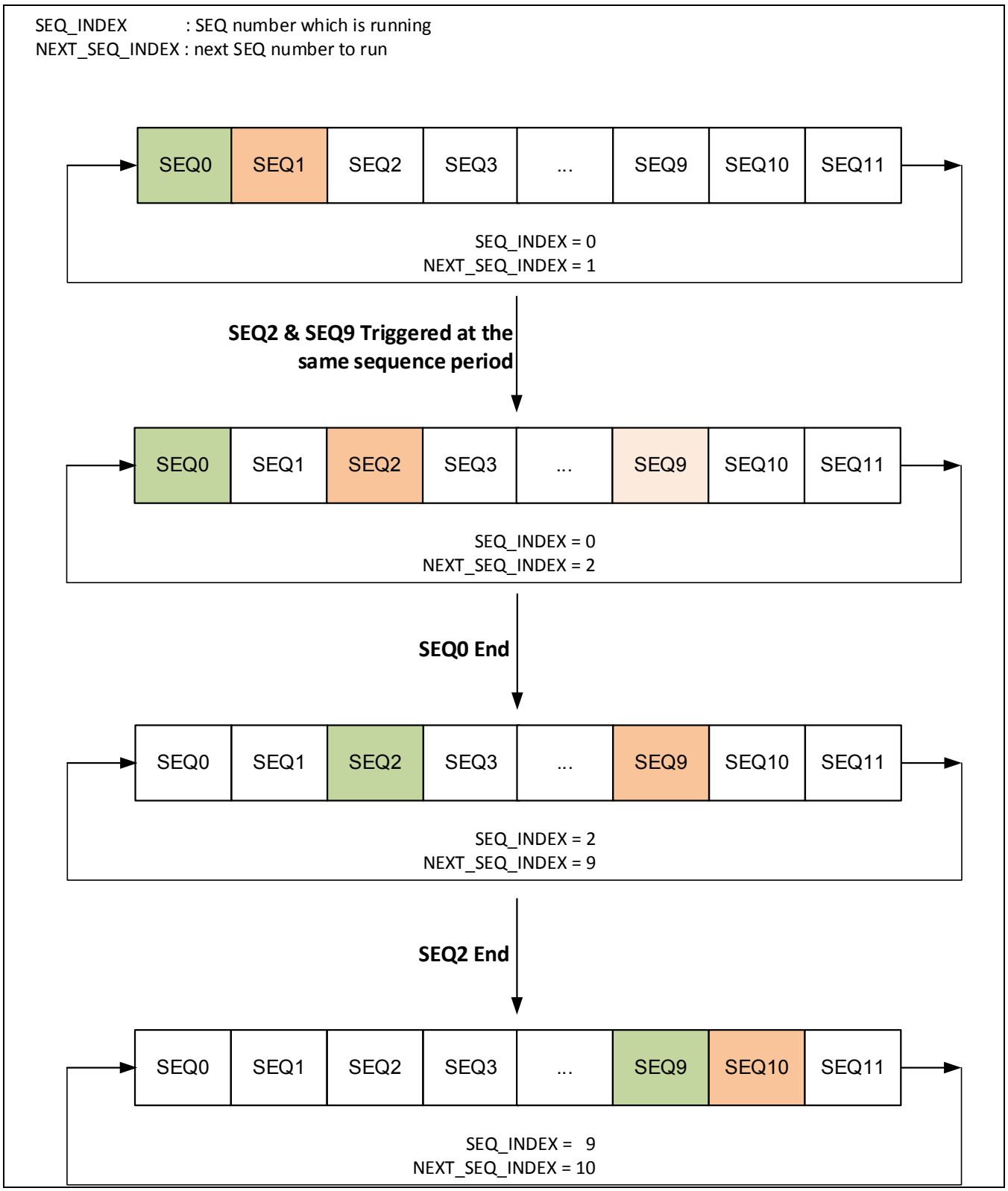


Figure 8-6 同时触发示意图

当某些特殊的转换序列不需要连续转换，而又比普通序列需要有更高的触发转换优先级时，可以使用ADC_PRI寄存器来设置优先级。比ADC_PRI寄存器中设置的值小的序列，会从转换序列中剔除，并且有更高的触发优先级。参考下图的例子。

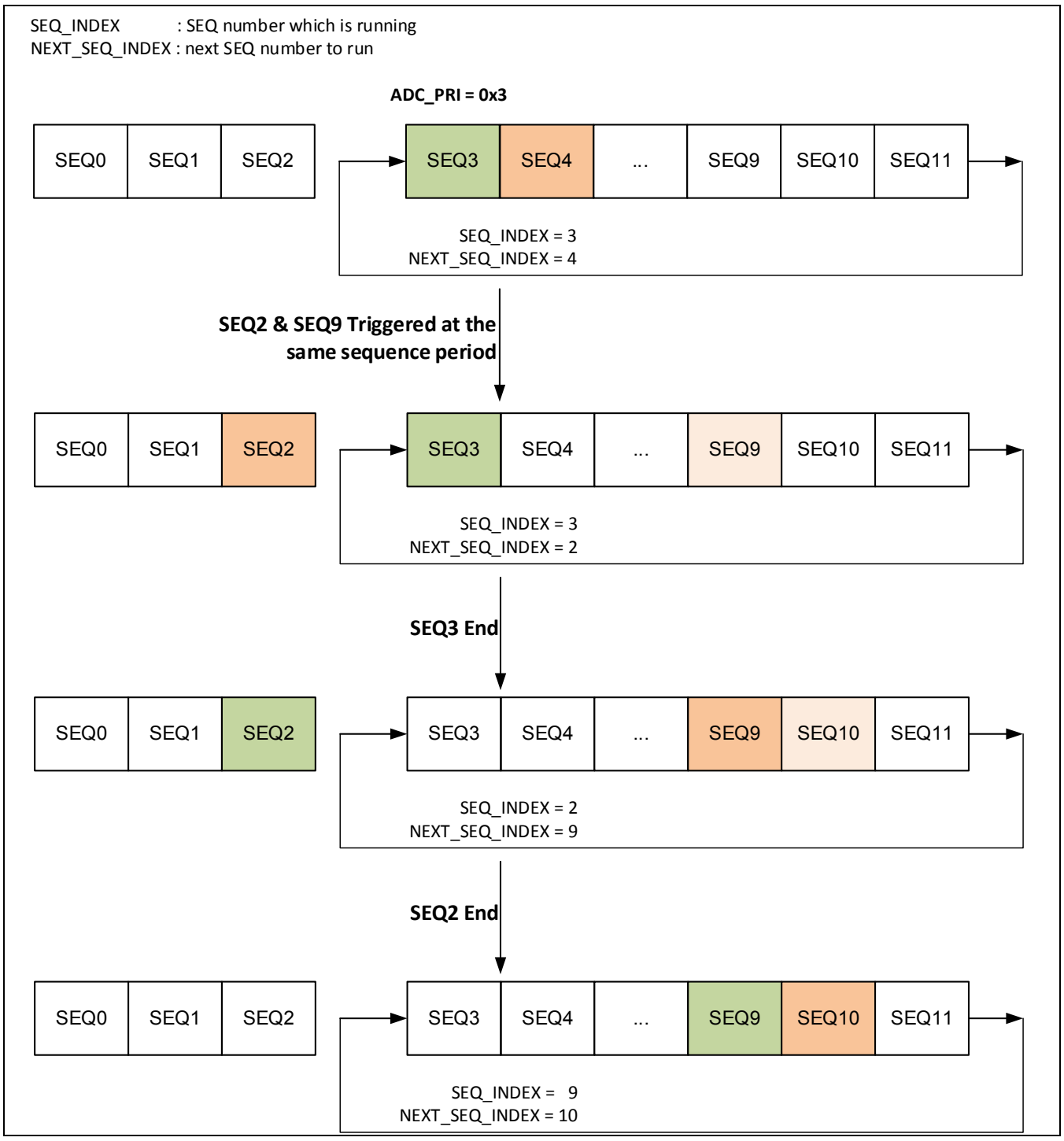


Figure 8-7 触发优先级示意图

注意：如果某个触发源需要触发两个或多个序列 (需要连续转换两个或多个通道的值)，那么需要这些序列必须是连续的序列，否则按照序列号低优先级高的原则，多个序列将不会被连续转换。

例如，如果设置PWM触发SEQ4, SEQ10, SEQ11，CMP触发SEQ5，那么当PWM和CMP触发同时发生时，CMP的SEQ5会抢在SEQ10和SEQ11之前转换。所以如果要设置PWM触发三个序列，那么必须设置触发SEQ4, SEQ5, SEQ6，CMP触发SEQ7，这样当PWM和CMP触发同时发生时，会先转换PWM的连续3个序列SEQ4, SEQ5, SEQ6，然后再转换SEQ7。

8.1.12 功耗管理

ADC 模块含有功耗管理功能，用以减少模块的功耗。功耗可以从两方面减少：模拟和数字。

- 减少模拟功耗：为了降低模拟功耗，CPU 需要禁用 ADC 模块(写 ADC_CR 中的 ADCDIS 位)，让 ADC 处于待机模式。
- 减少数字功耗：为了降低数字功耗，CPU 需要关闭 ADC 时钟(写 ADC_DCR 中的 ADC 位)，让 ADC 的数字模块没有输入时钟，这样数字功耗就降到几乎为 0 了。注意当时钟被关闭时，除了“时钟使能寄存器”以外的所有寄存器的写操作都无效，但是读操作仍然可以。

所以，为了让 ADC 模块处于最低功耗状态，必须先关闭 ADC 模拟模块(写 ADC_CR 中的 ADCDIS 位)，然后再关闭时钟(写 ADC_DCR 中的 ADC 位)。另一方面，为了让 ADC 退出最低功耗状态，必须先打开时钟(写 ADC_ECR 中的 ADC 位)，然后再打开 ADC 模拟模块(写 ADC_CR 中的 ADCEN 位)。

下表列出了功耗管理的各种状态：

Table 8-9 功耗管理的状态位

寄存器中的状态位	状态位为1时	状态位为0时
ADC_PMSR中的ADC位	时钟被使能	时钟被禁止，降低数字功耗
ADC_SR中的ADCENS位	模拟模块处于工作状态	模拟模块处于待机状态，降低模拟功耗

8.1.13 EOC标志 (End of Conversion)

状态寄存器中的 EOC 位表示转换结果寄存器中有新的值。

- 如果 EOC 是 0，表示自从这位清零后，或者上一个转换的结果被 CPU 读取后，还没有完成过任何转换。
- 如果 EOC 是 1，表示有 AD 转换完成，并且转换结果寄存器中的新数据还没有被读取。

注意：每次读转换结果寄存器(ADC_DR)都会将 EOC 标志位清零。

8.1.14 Ready标志

状态寄存器中的 READY 位表示 ADC 已经做好准备，可以开始进行转换。当 ADC 正在进行转换的时候，读取这位会返回 0。

8.1.15 OVR标志 (转换溢出)

这个标志表示某个转换完成的数据还没有被读取，就被新的数据覆盖了。

OVR 标志可以被 CPU 清除(在状态清除寄存器里写 OVR 位)。

8.1.16 CMPxH/L标志

这个标志表示某个选择的通道的转换结果比预设的值(ADC_CMPx)高或者低。

CMPxH/L 标志可以被 CPU 清除(在状态清除寄存器里写 CMPxH/L 位)。

8.1.17 SEQ_ENDx标志

这个标志表示序列 x 的转换已经完成。

SEQ_ENDx 标志可以被 CPU 清除(在状态清除寄存器里写 SEQ_END[x]位)。

8.1.18 工作流程

当 ADC 转换被启动后，ADC 转换开始。当转换结束时，EOC 位(ADC_SR[0])会自动被置 1，并且转换的结果被存入到 ADC_DR 寄存器中以供读取。然后 ADC 进入等待状态。在开始另一个转换前，记住要先读取 ADC_DR 寄存器的内容，否则下个转换结果将会覆盖前一个结果。

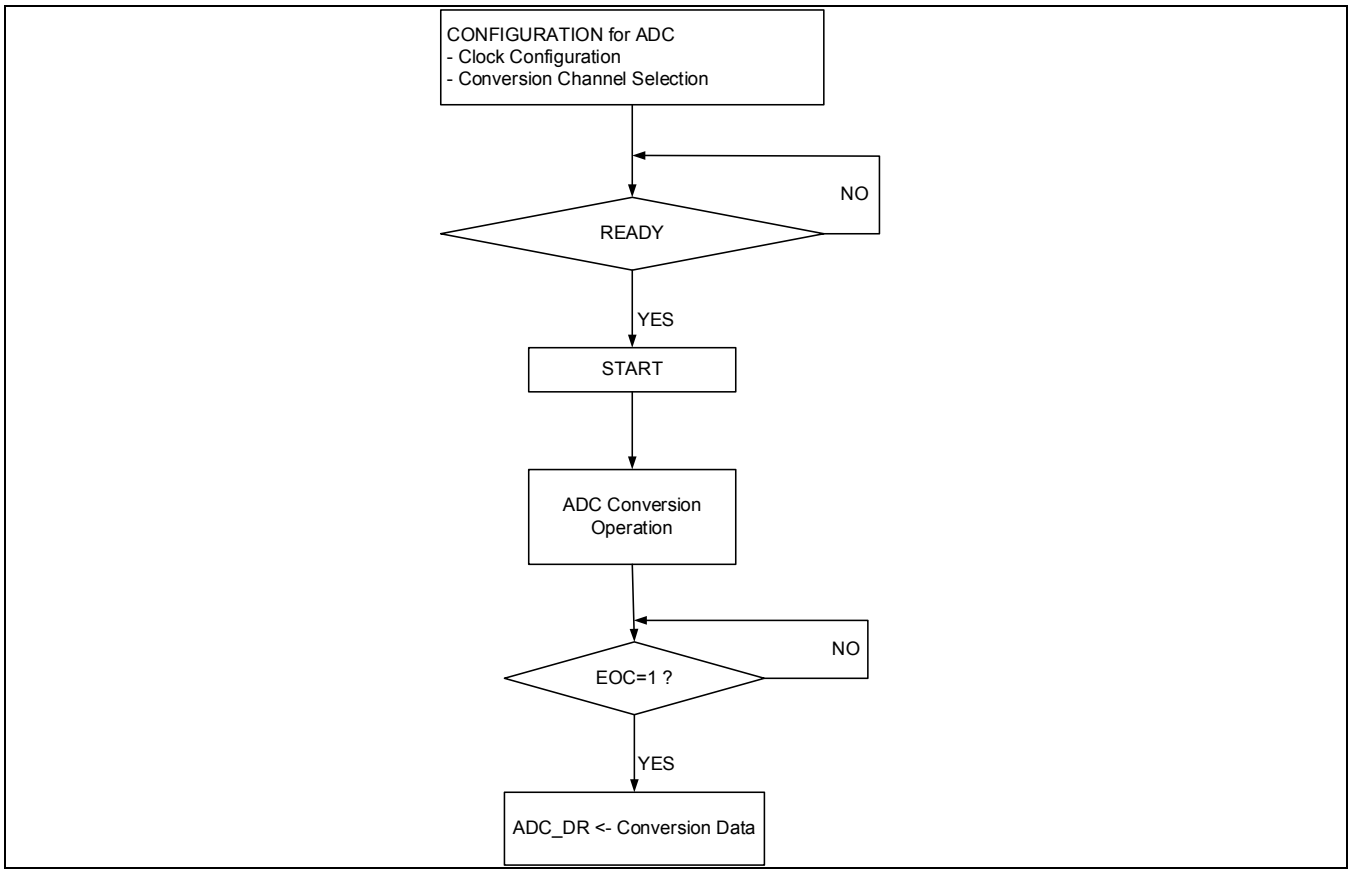


Figure 8-8 ADC工作流程图

8.1.19 转换的软件操作流程

下面描述了在复位后使用 ADC 模块的基本操作流程：

1. 在 ADC_ECR 中使能时钟

2. 在 ADC_MR 中设置 ADC 工作模式。PRLVAL 的值不能让模拟模块的工作时钟频率超过 10MHz。设置单次转换还是连续转换模式。定义转换序列：转换序列个数(NBRCH)和哪些输入通道需要被转换(ADC_SEQx 中的 AIN_SEL)
3. 使能 ADC 模块(ADC_CR 中的 ADC_EN)
4. 等待 ADC_SR 中的 READY 位。只有当这个标志位被置 1 后，ADC 才能正常的开始转换。如果 ADC_IMR 中的相应中断被使能，那么当 READY 标志置起的时候，会产生一个中断
5. 通过写 ADC_CR 中的 START 位，开始转换
6. ADC 选择转换序列中的第一个模拟输入通道
7. 模拟输入电压被采样并且在 21/19 个时钟周期后，转换完成。12 位/10 位的数字转换结果被存入到 ADC_DR 中，并且 ADC_SR 中的 EOC 位被置 1。如果 EOC 标志已经是 1，那么 OVR 位会被置 1。
8. 然后 CPU 就可以读取 ADC_DR 中的数字值，并且自动清除 EOC。在连续转换模式中，如果 CPU 判断不需要更多的转换了，那么它可以写 STOP 位停止转换。这样 ADC 就会停止工作并且等待下一个开始转换的请求。注意在单次转换模式，ADC 不可以被停止，它会转换完所有的序列后自己停止。
9. 如果 NBRCH 不是 0，那么 ADC 会选择下一个需要转换的模拟输入通道，然后从上面第 6 步重新开始。
10. 如果 CONTCV 是 1，那么 ADC 会从第 5 步重新开始另一个转换序列。

8.2 寄存器说明

8.2.1 寄存器表

- Base Address: 0x4003_0000

Register	Offset	Description	Reset Value
ADC_ECR	0x0000	时钟使能寄存器	–
ADC_DCR	0x0004	时钟禁止寄存器	–
ADC_PMSR	0x0008	功耗管理状态寄存器	–
–	0x000C	Reserved	–
ADC_CR	0x0010	控制寄存器	0x80000000
ADC_MR	0x0014	模式寄存器	0x00000000
–	0x0018	Reserved	–
ADC_CSR	0x001C	状态清除寄存器	–
ADC_SR	0x0020	状态寄存器	0x00000000
ADC_IER	0x0024	中断使能寄存器	–
ADC_IDR	0x0028	中断禁止寄存器	–
ADC_IMR	0x002C	中断使能状态寄存器	0x00000000
ADC_SEQx	0x0030 ~ 0x006C	转换序列寄存器x (x=0~15)	0x00000080
ADC_PRI	0x0070	转换序列优先级寄存器	0x00000000
–	0x0074 ~ 0x00FC	Reserved	
ADC_DRx	0x0100	转换结果寄存器x (x=0~15)	0x00000000
ADC_CMP0	0x0140	比较数据0寄存器	0x00000000
ADC_CMP1	0x0144	比较数据1寄存器	0x00000000

8.2.2 ADC_ECR (时钟使能寄存器)

- Address = Base Address + 0x0000, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
ADC	[1]	W	**ADC: ADC时钟使能** 0: 无效 1: 使能ADC时钟	-
DBGEN	[31]	W	**DBGEN: ADC调试模式使能** 0: 无效 1: 使能ADC调试模式	-

8.2.3 ADC_DCR (时钟禁止寄存器)

- Address = Base Address + 0x0004, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
ADC	[1]	W	**ADC: ADC时钟禁止** 0: 无效 1: 禁止ADC时钟	-
DBGEN	[31]	W	**DBGEN: ADC调试模式禁止** 0: 无效 1: 禁止ADC调试模式	-

8.2.4 ADC_PMSR (功耗管理状态寄存器)

- Address = Base Address + 0x0008, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
ADC	[1]	R	**ADC：ADC 时钟状态** 0: ADC 时钟被禁止 1: ADC 时钟被使能	0
IPICODE	[29:4]	R	**IPICODE[25:0]：IP 识别码** 模块的版本号，共 26 位	-
DBGEN	[31]	R	**DBGEN： 调试模式** 0: ADC 在调试模式下不停止 1: ADC 在调试模式下停止工作	0

8.2.5 ADC_CR (控制寄存器)

- Address = Base Address + 0x0010, Reset Value = 0x0000_0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCURACY	RSVD																				FVR_LVL	FVR_EN	VREF			SWTRG	STOP	START	ADCDIS	ADCEN	SWRST
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	R	R	W	W	W	W	W	W

Name	B	Type	Description	Reset Value
SWRST	[0]	W	**SWRST : ADC 软件复位** 0: 无效 1: 复位 ADC 模块 当软件复位发生时，除了 ADC_PMSR 寄存器以外，其它所有寄存器都会恢复初始值。	-
ADCEN	[1]	W	**ADCEN : ADC 使能** 0: 无效 1: 使能 ADC 模块	-
ADCDIS	[2]	W	**ADCDIS : ADC 禁止** 0: 无效 1: 关闭 ADC 模块(待机模式) 如果 ADCEN 和 ADCDIS 都写 1，那么 ADC 会被禁用。	-
START	[3]	W	**START : 开始转换** 0: 无效 1: 开始模数转换，清除 EOC 标志位 **注意：**在开始转换前，用户必须保证 ADC 已经处于准备好转换的状态(ADC_SR 中的 READY 位必须为 1)	-
STOP	[4]	W	**STOP: 在连续转换模式下停止转换** 0: 无效 1: 停止连续转换	-
SWTRG	[5]	W	**SWTRG : 软件触发** 0: 无效 1: 触发转换序列	-
VREF	[8]	RW	**VREF: ADC 电压参考源选择** 0: VDD	0

			1: 外部 VREF 管脚或 FVR 电压值	
FVR_EN	[9]	RW	**FVR_EN: 使能固定电压参考源** 0: 禁止 1: 使能	0
FVR_LVL	[10]	RW	**FVR_LVL: 固定电压参考源的电压值选择** 0: 2.048V 1: 4.096V	0
ACCURACY	[31]	R/W	**ACCURACY: ADC 转换精度选择位** 0: 10 位 1: 12 位	1

8.2.6 ADC_MR (模式寄存器)

- Address = Base Address + 0x0014, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONTCV	RSVD						NBRCMP1			RSVD			NBRCMP0			RSVD		NBRCH				RSVD					PRLVAL				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
RW	R	R	R	R	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW	R	R	RW	RW	RW	RW	R	R	R	R	R	RW	RW	RW	RW	RW

Name	B	Type	Description	Reset Value
PRLVAL	[4:0]	R/W	**PRLVAL[4:0]: 分频设置** 将 PCLK 分频，给 ADC 模拟模块作为时钟。 如果 PRLVAL == 0，那么 FADC = PCLK 否则 FADC = PCLK / (2*PRLVAL) **注意：** - ADC 模拟模块的时钟频率不能超过 24MHz - 如果系统时钟为 40MHz，那么 PRLVAL 至少为 1	00001
NBRCH	[13:10]	R/W	**NBRCH[3:0]: 转换序列个数** **NBRCH[3:0]** / **转换序列的个数** 0000b / 1 0001b / 2 … / … 1111b / 16 **注意：** 即使在单次转换模式，如果 NBRCH[3:0]的值大于 0，ADC 也会进行多次转换。	000000
NBRCMP0	[19:16]	R/W	**NBRCMP0[3:0]: 需要比较的转换序列** 当该次转换结果大于或者小于 ADC_CMP0 寄存器时，将产生一个 CMPxH/CMPxL 中断	000000
NBRCMP1	[25:22]	R/W	**NBRCMP1[3:0]: 需要比较的转换序列** 当该次转换结果大于或者小于 ADC_CMP1 寄存器时，将产生一个 CMPxH/CMPxL 中断	000000
CONTCV	[31]	R/W	**CONTCV: 连续转换** 0: 单次转换模式。ADC 根据 NBRCH[3:0]中设置的值转换输入的信号并且停止 1: 连续转换模式。ADC 根据 NBRCH[3:0]中设置的值转换输入的信号并且重复不停的循环转换。	0

			该位初始值为 0. **注意：**在连续转换模式下，ADC 收到停止指令后，仍然会完成当前正在进行的转换，看起来像是多转换了一次。	

8.2.7 ADC_CSR (状态清除寄存器)

- Address = Base Address + 0x001C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQ_END[15]	SEQ_END[14]	SEQ_END[13]	SEQ_END[12]	SEQ_END[11]	SEQ_END[10]	SEQ_END[9]	SEQ_END[8]	SEQ_END[7]	SEQ_END[6]	SEQ_END[5]	SEQ_END[4]	SEQ_END[3]	SEQ_END[2]	SEQ_END[1]	SEQ_END[0]	RSVD								CMP1L	CMP1H	CMP0L	CMP0H	RSVD	OVR	READY	RSVD
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W

Name	B	Type	Description	Reset Value
READY	[1]	W	**READY：ADC 已准备好可以转换中断** 0: 无效 1: 清除该中断	-
OVR	[2]	W	**OVR： 转换溢出中断** 0: 无效 1: 清除该中断	-
CMP0H	[4]	W	**CMP0H： 转换结果大于 ADC_CMP0 中断** 0: 无效 1: 清除该中断	-
CMP0L	[5]	W	**CMP0L： 转换结果小于 ADC_CMP0 中断** 0: 无效 1: 清除该中断	-
CMP1H	[6]	W	**CMP1H： 转换结果大于 ADC_CMP1 中断** 0: 无效 1: 清除该中断	-
CMP1L	[7]	W	**CMP1L： 转换结果小于 ADC_CMP1 中断** 0: 无效 1: 清除该中断	-
SEQ_END[x]	[31:16]	W	**SEQ_END[x]：SEQx 序列转换完成中断** 0: 无效 1: 清除该中断 SEQ_END[16]对应 SEQ0 序列，以此类推	-

8.2.8 ADC_SR (状态寄存器)

- Address = Base Address + 0x0020, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQ_END[15]	SEQ_END[14]	SEQ_END[13]	SEQ_END[12]	SEQ_END[11]	SEQ_END[10]	SEQ_END[9]	SEQ_END[8]	SEQ_END[7]	SEQ_END[6]	SEQ_END[5]	SEQ_END[4]	SEQ_END[3]	SEQ_END[2]	SEQ_END[1]	SEQ_END[0]	RSVD		SEQ_INDEX				CTCVS	ADCENS	CMP1L	CMP1H	CMP0L	CMP0H	RSVD	OVR	READY	EOC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	B	Type	Description	Reset Value
EOC	[0]	R	**EOC：转换结束** 0: 转换未结束，仍在进行中 1: 转换完成，ADC_DR 中的数据有效。当 ADC_DR 被读取时该位自动清零	0
READY	[1]	R	**READY：ADC 已准备好可以转换** 0: ADC 忽略开始或者停止指令：因为它还没有准备好或者转换未结束它还在工作中 1: ADC 已经准备好，可以开始一个转换	0
OVR	[2]	R	**OVR：转换溢出** 0: 最后一次读 ADC_DR 时，ADC 没有完成任何转换或者只完成了 1 次转换 1: 最后一次读 ADC_DR 时，ADC 完成了 2 次或者 2 次以上的转换	0
CMP0H	[4]	R	**CMP0H：比较功能的状态** 0: ADC 转换的结果比 ADC_CMP0 小 1: ADC 转换的结果比 ADC_CMP0 大	0
CMP0L	[5]	R	**CMP0L：比较功能的状态** 0: ADC 转换的结果比 ADC_CMP0 大 1: ADC 转换的结果比 ADC_CMP0 小	0
CMP1H	[6]	R	**CMP1H：比较功能的状态** 0: ADC 转换的结果比 ADC_CMP1 小 1: ADC 转换的结果比 ADC_CMP1 大	0
CMP1L	[7]	R	**CMP1L：比较功能的状态** 0: ADC 转换的结果比 ADC_CMP1 大 1: ADC 转换的结果比 ADC_CMP1 小	0
ADCENS	[8]	R	**ADCENS：ADC 使能状态**	0

			0: ADC 被禁止 1: ADC 被使能	
CTCVS	[9]	R	**CTCVS：连续转换模式状态** 0: 单次模式 1: 连续模式	0
SEQ_INDEX	[3:0]	R	**SEQ_INDEX：当前转换序列** 该寄存器的值为当前转换的序列号	0
SEQ_END[x]	[31:16]	R	**SEQ_END[x]：SEQx 序列转换完成中断** 0: 该转换序列没完成 1: 该转换序列已完成	0

为了更好的解释 READY 标志位，让我们把 ADC 正在转换数据的状态叫做“正在工作”状态，当模拟模块被禁用或者还在初始化时，把“模拟模块是否准备好”事件标记为 0 状态。

Table 8-10 是否准备好进行转换

模拟模块是否准备好	正在工作	READY
0	0	0
0	1	0
1	0	1
1	1	0

8.2.9 ADC_IER (中断使能寄存器)

- Address = Base Address + 0x0024, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQ_END[15]	SEQ_END[14]	SEQ_END[13]	SEQ_END[12]	SEQ_END[11]	SEQ_END[10]	SEQ_END[9]	SEQ_END[8]	SEQ_END[7]	SEQ_END[6]	SEQ_END[5]	SEQ_END[4]	SEQ_END[3]	SEQ_END[2]	SEQ_END[1]	SEQ_END[0]	RSVD								CMP1L	CMP1H	CMP0L	CMP0H	RSVD	OVR	READY	EOC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W

Name	B	Type	Description	Reset Value
EOC	[0]	W	**EOC：转换结束中断** 0: 无效 1: 使能该中断	-
READY	[1]	W	**READY：ADC READY 中断** 0: 无效 1: 使能该中断	-
OVR	[2]	W	**OVR：转换溢出中断** 0: 无效 1: 使能该中断	-
CMP0H	[4]	W	**CMP0H：转换结果高于 ADC_CMP0 中断** 0: 无效 1: 使能该中断	-
CMP0L	[5]	W	**CMP0L：转换结果低于 ADC_CMP0 中断** 0: 无效 1: 使能该中断	-
CMP1H	[6]	W	**CMP1H：转换结果高于 ADC_CMP1 中断** 0: 无效 1: 使能该中断	-
CMP1L	[7]	W	**CMP1L：转换结果低于 ADC_CMP1 中断** 0: 无效 1: 使能该中断	-
SEQ_END[x]	[31:16]	W	**SEQ_END[x]：SEQx 序列转换完成中断** 0: 无效 1: 使能该中断	-

注意： 对于 CMPxH 和 CMPxL 中断，请勿将“H”和”L”中断同时使能。

8.2.10 ADC_IDR (中断禁止寄存器)

- Address = Base Address + 0x0028, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQ_END[15]	SEQ_END[14]	SEQ_END[13]	SEQ_END[12]	SEQ_END[11]	SEQ_END[10]	SEQ_END[9]	SEQ_END[8]	SEQ_END[7]	SEQ_END[6]	SEQ_END[5]	SEQ_END[4]	SEQ_END[3]	SEQ_END[2]	SEQ_END[1]	SEQ_END[0]	RSVD								CMP1L	CMP1H	CMP0L	CMP0H	RSVD	OVR	READY	EOC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W

Name	B	Type	Description	Reset Value
EOC	[0]	W	**EOC : 转换结束中断** 0: 无效 1: 禁止该中断	-
READY	[1]	W	**READY : ADC READY 中断** 0: 无效 1: 禁止该中断	-
OVR	[2]	W	**OVR : 转换溢出中断** 0: 无效 1: 禁止该中断	-
CMP0H	[4]	W	**CMP0H : 转换结果高于 ADC_CMP0 中断** 0: 无效 1: 禁止该中断	-
CMP0L	[5]	W	**CMP0L : 转换结果低于 ADC_CMP0 中断** 0: 无效 1: 禁止该中断	-
CMP1H	[6]	W	**CMP1H : 转换结果高于 ADC_CMP1 中断** 0: 无效 1: 禁止该中断	-
CMP1L	[7]	W	**CMP1L : 转换结果低于 ADC_CMP1 中断** 0: 无效 1: 禁止该中断	-
SEQ_END[x]	[31:16]	W	**SEQ_END[x] : SEQx 序列转换完成中断** 0: 无效 1: 禁止该中断	-

8.2.11 ADC_IMR (中断使能状态寄存器)

- Address = Base Address + 0x002C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQ_END[15]	SEQ_END[14]	SEQ_END[13]	SEQ_END[12]	SEQ_END[11]	SEQ_END[10]	SEQ_END[9]	SEQ_END[8]	SEQ_END[7]	SEQ_END[6]	SEQ_END[5]	SEQ_END[4]	SEQ_END[3]	SEQ_END[2]	SEQ_END[1]	SEQ_END[0]	RSVD								CMP1L	CMP1H	CMP0L	CMP0H	RSVD	OVR	READY	EOC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	B	Type	Description	Reset Value
EOC	[0]	R	**EOC： 转换结束中断** 0: 该中断没有发生 1: 该中断发生	0
READY	[1]	R	**READY : ADC READY 中断** 0: 该中断没有发生 1: 该中断发生	0
OVR	[2]	R	**OVR： 转换溢出中断** 0: 该中断没有发生 1: 该中断发生	0
CMP0H	[4]	R	**CMP0H： 转换结果高于 ADC_CMP0 中断** 0: 该中断没有发生 1: 该中断发生	0
CMP0L	[5]	R	**CMP0L： 转换结果低于 ADC_CMP0 中断** 0: 该中断没有发生 1: 该中断发生	0
CMP1H	[6]	R	**CMP1H： 转换结果高于 ADC_CMP1 中断** 0: 该中断没有发生 1: 该中断发生	0
CMP1L	[7]	R	**CMP1L： 转换结果低于 ADC_CMP1 中断** 0: 该中断没有发生 1: 该中断发生	0
SEQ_END[x]	[31:16]	R	**SEQ_END[x] : SEQx 序列转换完成中断** 0: 该中断没有发生 1: 该中断发生	0

8.2.12 ADC_SEQx (转换序列寄存器)

- ADC_SEQ0 Address = Base Address + 0x0030, Reset Value = 0x0000_0000
- ADC_SEQ1 Address = Base Address + 0x0034, Reset Value = 0x0000_0000
- ……
- ADC_SEQ15 Address = Base Address + 0x006C, Reset Value = 0x0000_0000

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

Name	B	Type	Description	Reset Value
AIN_SEL	[4:0]	RW	**模拟输入通道选择** **AIN_SEL 值** / **输入选择** **BIN** / **DEC** 00000 / 0 / AIN0 00001 / 1 / AIN1 00010 / 2 / AIN2 …… / / …… 10001 / 17 / AIN17 …… / …… / N/A 11100 / 28 / FVR 11101 / 29 / 1/4VDD 11110 / 30 / VSS 11111 / 31 / N/A	-
SAMPLE	[7:6]	R/W	**SAMPLE: ADC 采样周期 (setup 时间)** 00: 3 个周期 01: 4 个周期 10: 6 个周期 11: 8 个周期 **注意：**采样时间不能小于(TBD) us	10
CV_CNT	[10:8]	RW	**CV_CNT: 连续重复采样次数** 000 : 1 001 : 2	000

			010 : 4 011 : 8 100 : 16 101 : 32 110 : 64 111 : 128	
AVG_CAL	[12]	RW	**AVG_CAL：平均值计算** 0：禁用 1：使能 当这位使能时，ADC 转换结果寄存器 ADC_DRx 将保存若干次数转换后的平均值，这个次数由 CV_CNT 设定。否则，ADC_DRx 将保存最后一次转换的值。	0
TRG_SRC	[15:13]	RW	**TRG_SRC：触发源选择** 000：无触发 001：软件触发(ADC_CR 中的 SWTRG 位) 010 : TC1 脉冲匹配中断 011 : EPWM 触发 (EPWM 模块里可以选择具体的触发源) 100 : CMP 触发 (CMP 模块里可以选择具体的触发源) 101：外部管脚，上升沿 110：外部管脚，下降沿 111：外部管脚，上升和下降沿	000

8.2.13 ADC_PRI (ADC转换序列优先权寄存器)

- Address = Base Address + 0x0070, Reset Value = 0x0000_0000

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

Name	B	Type	Description	Reset Value
PRI	[3:0]	RW	**PRI：转换序列优先权选择** 比这个寄存器数值低的序列有更高的优先权	0x0

8.2.14 ADC_TDL0 (触发延时寄存器0)

- Address = Base Address + 0x0074, Reset Value = 0x0000_0000

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

Name	B	Type	Description	Reset Value
TC_TDL	[7:0]	RW	**TC_TDL : TC 触发延时控制** 触发时，使用计数器延时一段时间后，才开始 ADC 转换。 延时 = (TC_TDL+1) x 4 x PCLK 周期	0
EPWM_TDL	[15:8]	RW	**EWPM_TDL： 触发延时控制** 触发时，使用计数器延时一段时间后，才开始 ADC 转换。 延时 = (EPWM_TDL+1) x 4 x PCLK 周期	0
CMP_TDL	[23:16]	RW	**CMP_TDL： 触发延时控制** 触发时，使用计数器延时一段时间后，才开始 ADC 转换。 延时 = (CMP_TDL+1) x 4 x PCLK 周期	0

注意：延时寄存器(xxx_TDL)如果等于0，那么延时功能为关闭状态，只有在不等于0的时候，才会开启延时功能。

8.2.15 ADC_TDL1 (触发延时寄存器1)

- Address = Base Address + 0x0078, Reset Value = 0x0000_0000

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

Name	B	Type	Description	Reset Value
RISE_TDL	[7:0]	RW	**RISE_TDL：外部管脚上升沿触发延时控制** 触发时，使用计数器延时一段时间后，才开始 ADC 转换。 延时 = (RISE_TDL+1) x 4 x PCLK 周期	0
FALL_TDL	[15:8]	RW	**FALL_TDL：外部管脚下降沿触发延时控制** 触发时，使用计数器延时一段时间后，才开始 ADC 转换。 延时 = (FALL_TDL+1) x 4 x PCLK 周期	0

注意：延时寄存器(xxx_TDL)如果等于0，那么延时功能为关闭状态，只有在不等于0的时候，才会开启延时功能。

8.2.16 ADC_DRx (转换结果寄存器)

- ADC_DR0 Address = Base Address + 0x0100, Reset Value = 0x0000_0000
- ADC_DR1 Address = Base Address + 0x0104, Reset Value = 0x0000_0000
- ……
- ADC_DR2 Address = Base Address + 0x013C, Reset Value = 0x0000_0000

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

Name	B	Type	Description	Reset Value
DATA	[11:0]	R	**DATA[11:0]：转换结果** 模数转换的结果在转换结束后，锁存在该寄存器，直到下一个转换完成前一直有效可供读取。 当该寄存器被读取后，ADC_SR 的 EOC 位会被自动清零。x **注意 1：** 在调试的时候，为了避免清除 EOC 位，可以使用 ADC_DR 的镜像寄存器读取转换结果。镜像寄存器的地址为 Base Address + 0x0900 **注意 2：** 在 10 位模式下(ADC_CR[31]=1)，该寄存器最高两位(ADC_DR[11:10])的值保持为 00。	0x0

8.2.17 ADC_CMP0 (比较数据0寄存器)

- Address = Base Address + 0x0140, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																				CMP0											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	B	Type	Description	Reset Value
CMP0	[11:0]	RW	**CMP0[11:0]：比较阈值** 模数转换结束后，转换结果会和这个寄存器的值进行比较，根据比较的结果触发相应的中断。	0x0

8.2.18 ADC_CMP1 (比较数据1寄存器)

- Address = Base Address + 0x0144, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																				CMP1											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	B	Type	Description	Reset Value
CMP1	[11:0]	RW	**CMP1[11:0]：比较阈值** 模数转换结束后，转换结果会和这个寄存器的值进行比较，根据比较的结果触发相应的中断。	0x0

9 模拟比较器（CMP）

9.1 概述

模拟比较器通过比较两个模拟输入电压量，输出一个数字标志用以表示两个输入量的幅值大小，是模拟电路和数字电路的的一种转换接口。模拟比较器在数模混合电路中作为非常重要的一种构建单元，可以提供独立于程序运行的模拟功能。

9.1.1 特性

- 支持最大5个独立模拟比较器。
- 每个模拟比较器支持正负端外部输入，比较结果直接 IO 输出。
  - 比较器的正向端可以选择外部输入，或者运放输出作为输入。
  - 比较器的负向端可以选择外部输入，或者内部参考电压。
- 247个内部参考电压
  - 支持246个从 VDD 分压得到的内部参考电压（最小分辨率：VDD/256）。
  - 支持 FVR 作为比较器输入参考电压源
  - 每个模拟比较器可以独立设置不同的参考电压输入。
- 可配置的比较器输出极性。
- 可配置的输出迟滞特性。
- 可配置的比较器输出数字滤波选择。
- 可配置的捕获滤波器选择。
- 中断触发模式选择（上升沿、下降沿）。
- 支持中断硬件自动触发 ADC 转换。
- 支持比较器触发 EPWM
- 支持比较器和 TC1联动
  - 触发 TC1计数。
  - 计较器输出作为 TC1的时钟，对比较器翻转个数进行计数。
  - 比较器输出作为 TC1的 Capture 输入，对比较器的输出电平宽度进行测量。

9.1.2 管脚描述

Table 9-1 CMP 管脚描述

管脚名称	功能	I/O类型	有效状态	备注
CPINP[9:0]	比较器模拟输入正向端通道	A	-	-

CPINN[3:0]	比较器模拟输入负向端通道	A	-	-
CPx_OUT	比较器输出输出	O	-	-

9.2 功能描述

9.2.1 比较器功能

处理器中内嵌 5 个一致的模拟比较器，其数字输出和模拟输入的关系如下图所示。当 VIN+的电压低于 VIN-的时候，比较器的数字输出为低电平。反之，则为高电平。比较器 0 和比较器 1 的数字处理部分一致，比较器 2、比较器 3 和比较器 4 的数字处理部分，增加了比较器输出可由特定事件触发捕获的功能。

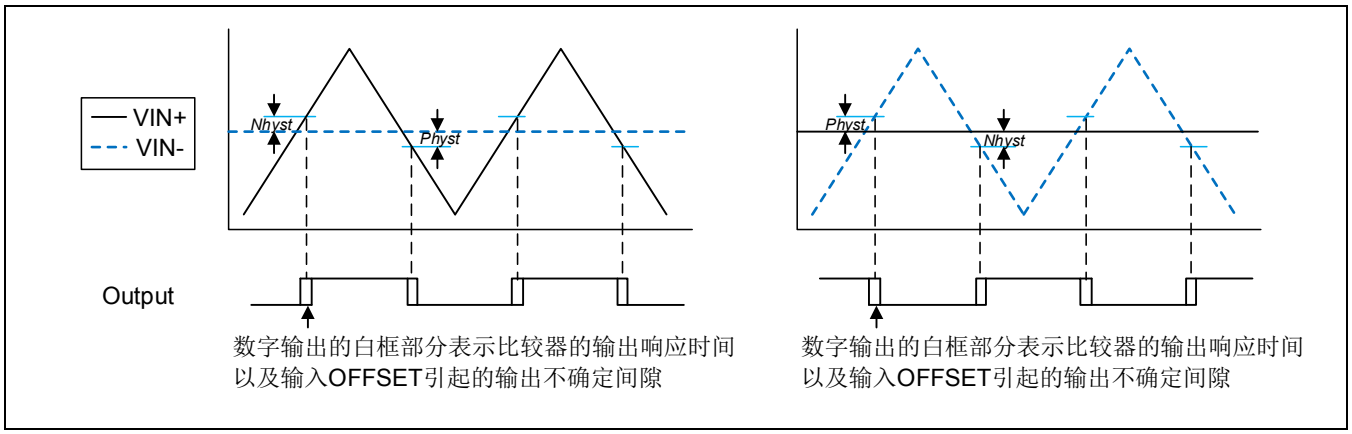


Figure 9-1 单一比较器示意图

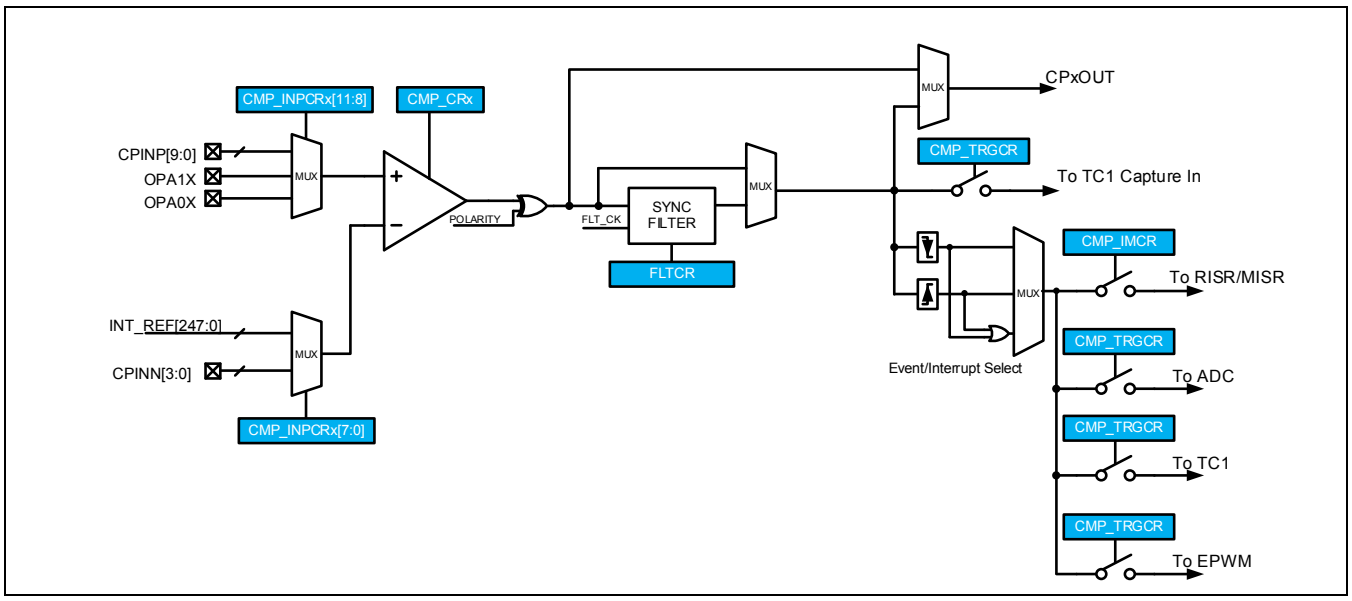


Figure 9-2 CMP0、CMP1比较器结构示意图

NOTE:

1) 所有比较器的模拟输入通道都是共享的，每个通道不是单一对应某一个比较器的模拟输入端。
2) 每个比较器的内部参考电压可以分别独立设置为247个内部参考点中的任意一个。
3) 比较器的控制时钟为比较器模块 PCLK 的时钟频率。

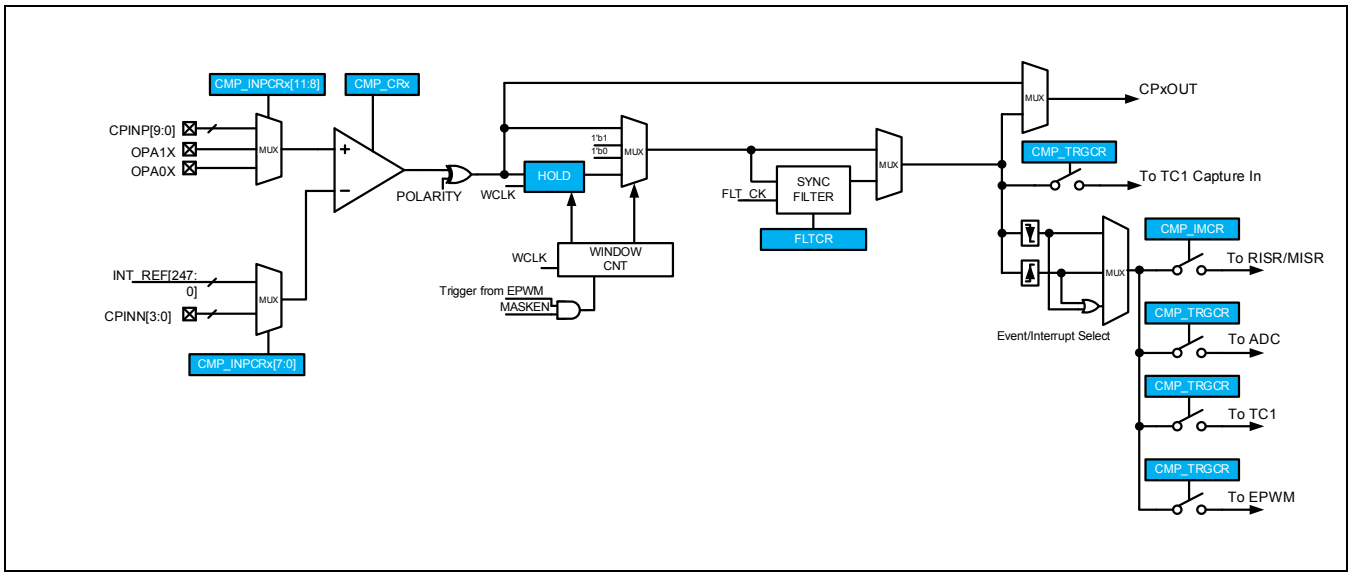


Figure 9-3 CMP2、CMP3、CMP4比较器结构示意图

NOTE:

1) 所有比较器的模拟输入通道都是共享的，每个通道不是单一对应某一个比较器的模拟输入端。
2) 每个比较器的内部参考电压可以分别独立设置为247个内部参考点中的任意一个。
3) 比较器的控制时钟为比较器模块 PCLK 的时钟频率。

9.2.2 比较器控制

每一个模拟比较器都有独立的控制寄存器来进行模式配置：CMP_CRx。控制寄存器中，可以设置比较器的使能，输出极性，响应速度以及相关的输入输出特性。对于比较器的输入模拟通道的选择，可以通过相对应的通道选择控制寄存器进行设置：CMP_INPCRx。比较器有两个模拟输入端，正向和负向输入端。正向模拟输入端，可以支持外部模拟信号的直接输入或者从模拟运算放大器的输出端得到模拟输入。负向输入端，可以支持外部模拟信号的直接输入或者从内部的247个电压参考源中选择任意一个作为输入。每个模拟比较器支持比较结果直接输出到外部管脚。

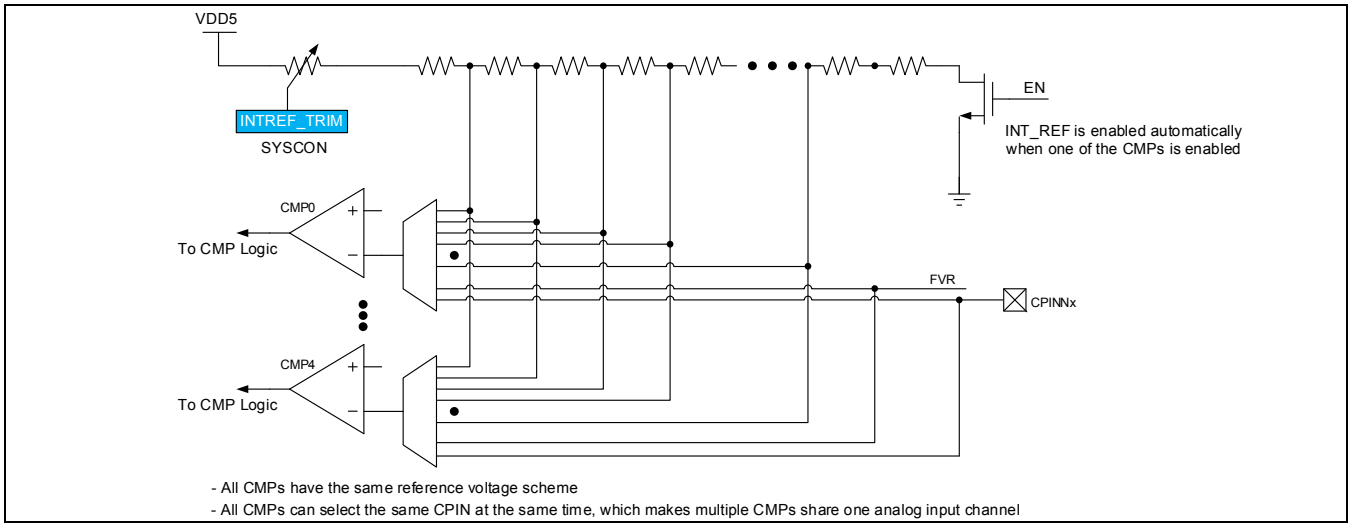


Figure 9-4 比较器内部参考电压

比较器负向输入支持从内部的电压参考源接入。内部参考电压源提供247个电压（其中246个为电阻从VDD分压获得，1个为内部FVR）可供选择，每个模拟比较器可以通过通道设置控制寄存器独立选择其中一个作为输入。

比较器的输出极性可以进行配置，极性设置必须在比较器使能之前进行，在比较器使能控制位使能以后不能再进行更改，但使能配置和极性设置可以在同一次写操作内完成。对比较器的输出取反输出，相当于交换比较器的正负向输入端口。比较器的输出反向可以通过控制寄存器中的 POLARITY 位进行设置。比较器的输出状态，可以通过控制寄存器的 CMPOUT 获得。

Table 9-2 比较器输出状态和输入关系

输入状态	极性设置	CPxOUT
CxVIN- > CxVIN+	0	0
CxVIN- < CxVIN+	0	1
CxVIN- > CxVIN+	1	1
CxVIN- < CxVIN+	1	0

比较器内建模拟输入迟滞滤波功能，可以通过控制器中的PHYST和NHYST控制位，分别设置正向和负向输入的迟滞滤波等级。

9.2.3 比较器响应时间

在比较器的输入端发生改变时，比如通道切换、参考源变化、或者输入变化等，都会使得比较器的输出在一定时间内的不确定性输出。这个时间长度为比较器的响应时间。此响应时间包含除比较器自身内部电路引起的延时以外，还包括参考电压源的稳定时间。因此在应用设计时，确定响应时间时，需要充分考虑两种延时的影响。

9.2.4 比较器输出数字滤波

比较器的输出端可以选择数字滤波输出，数字滤波的工作时钟为比较器的工作时钟（PCLK）。在使能 CR 控制寄存器的 FLTEN 后，该数字滤波器被打开，否则滤波器将被旁路。

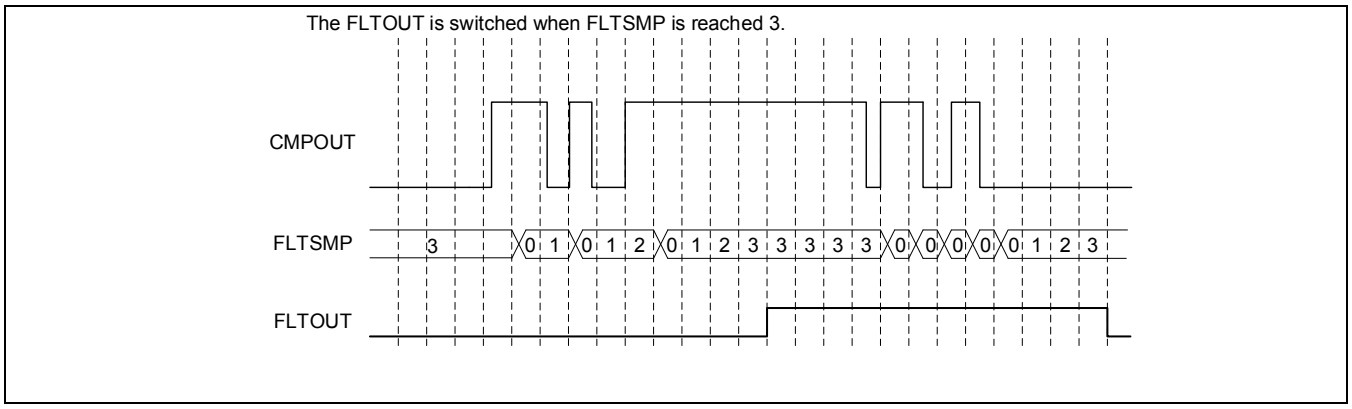


Figure 9-5 数字滤波机制

数字输出滤波采用数字去抖算法，滤波器在每个采样周期结束时对输入值进行采样。当输入电压从低电平到高电平切换时，在连续三次采样结果均为高电平的情况下，输出会确认从低电平切换到高电平。反之，当输入电压从

高电平切换到低电平时，在连续三次采样结果均为低电平的情况下，滤波输出会确认从高电平切换到低电平。在三次连续采样中，如果出现一次采样结果和前一次采样不一致时，滤波计数将会被清除，并重新开始三次采样。滤波器的采样周期通过 FLTCR 中的分频系数进行设置。

$$F_{FLT\_CK} = \frac{PCLK}{(DIVM+1) \times 2^{DIVN}}$$

在应用中，如果在使能滤波器后，需要再次修改数字滤波器的分频配置，则必须先停止滤波器（设置 BYPASS），修改分频配置后，再使能滤波器。这样才能确保新的分频配置立即有效。否则，新的配置需要在上一次配置的滤波周期结束时才会被更新到滤波器中。

9.2.5 比较器捕获滤波器

比较器的捕获功能指的是在特定的触发窗口内，比较器的输出将通过一个采样寄存器输出（该寄存器的采样时钟为 PCLK）。一旦该窗口关闭，寄存器将保持该输出状态或者以指定逻辑输出，直到下次的窗口有效。捕获窗口可以设置相应的延时时间，在触发信号有效后，延时特定的时间，再使能捕获窗口。捕获滤波器适用于在某些强干扰环境中，对比较器输出在特定时间内进行分析的应用。滤波器在初次使能时的输出极性（在滤波器使能后，但是未有触发前），可以通过 HLS 控制位进行设置。捕获窗口的触发支持外部特定的 EPWM 事件，该事件可以通过 WCNT 寄存器选择。捕获窗口的计时时钟基于比较器的工作时钟（PCLK），可以通过 CLKDIV 进行分频设置。

捕获窗口内，只有对特定的比较器输出沿敏感。根据当前 DPHS 的设置，捕获窗口选择下降沿或者上升沿敏感。当 DPHS 设置为低电平输出，则捕获窗口只对上升沿敏感；当 DPHS 设置为高电平输出，则捕获窗口只对下降沿敏感。

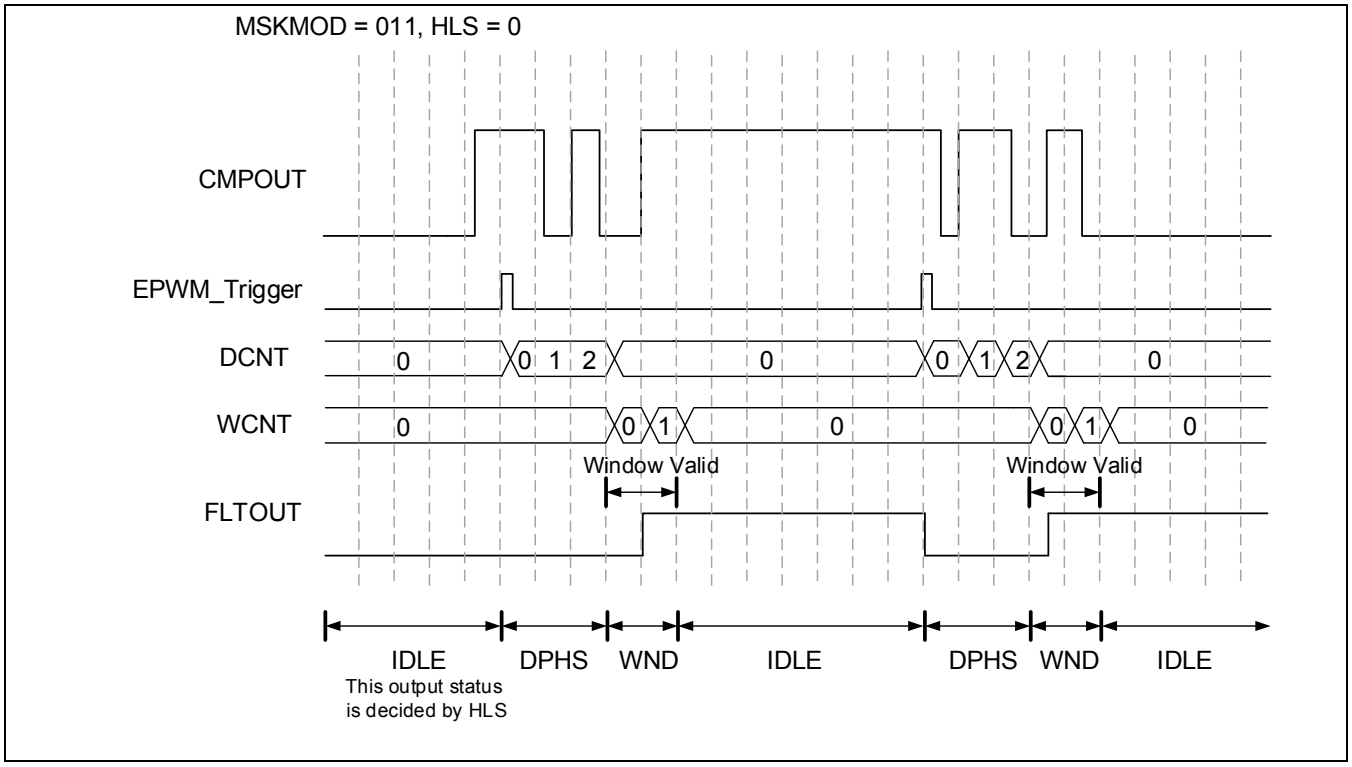


Figure 9-6 捕获滤波机制

9.2.6 比较器中断

比较器的中断事件触发可以选择上升沿触发，或者下降沿触发。每个比较器的中断可以通过IMCR寄存器来设置使能。中断的状态可以通过MISR和RISR寄存器查询。RISR中的标志位无论中断是否使能，一旦比较器的输出状态发生改变，都会自动置位。MISR中的标志位只有在IMCR中的相应位使能以后，才会在中断发生后置位。

比较器模块一共占用两个CPU的中断源，其中两个比较器可以触发CMP_INT0中断（CMP0/CMP2），三个比较器可以触发CMP_INT1中断（CMP1/CMP3/CMP4）。

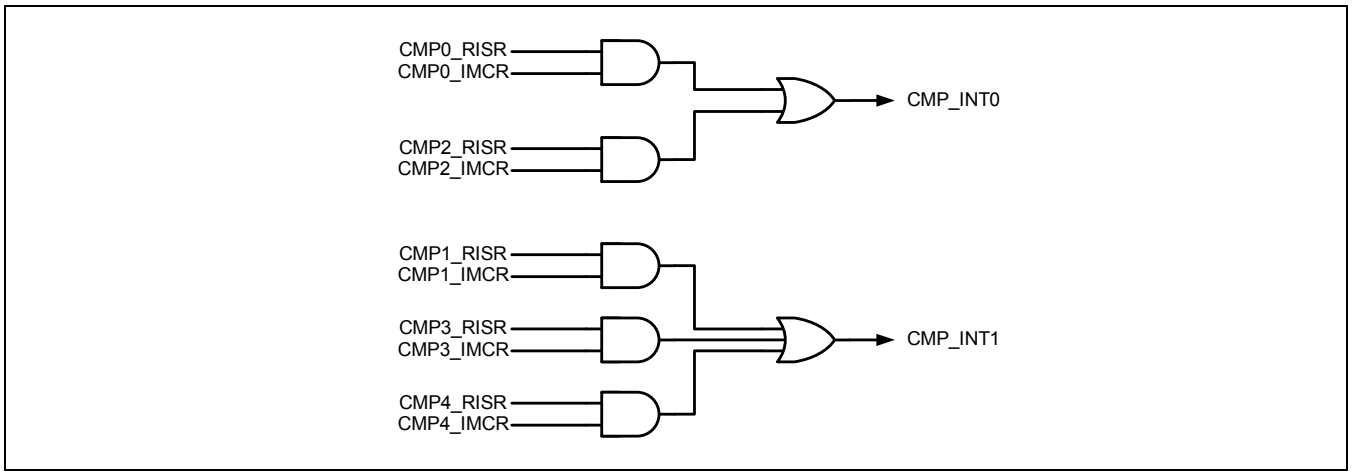


Figure 9-7 中断分配

9.3 寄存器说明

9.3.1 寄存器表

- Base Address: 0x400B_0000

Table 9-3 寄存器表

Register	Offset	Description	Reset Value
CMP_CEDR	0x00	ID 和时钟使能控制寄存器	0x0209_0000
CMP_CR0	0x04	比较器0的控制寄存器	0x0000_0040
CMP_CR1	0x08	比较器1的控制寄存器	0x0000_0040
CMP_CR2	0x0C	比较器2的控制寄存器	0x0000_0040
CMP_CR3	0x10	比较器3的控制寄存器	0x0000_0040
CMP_CR4	0x14	比较器4的控制寄存器	0x0000_0040
CMP_FLTCR0	0x18	比较器0的数字滤波控制器	0x0000_0000
CMP_FLTCR1	0x1C	比较器1的数字滤波控制器	0x0000_0000
CMP_FLTCR2	0x20	比较器2的数字滤波控制器	0x0000_0000
CMP_FLTCR3	0x24	比较器3的数字滤波控制器	0x0000_0000
CMP_FLTCR4	0x28	比较器4的数字滤波控制器	0x0000_0000
CMP_WCNT0	0x2C	比较器2的捕捉窗口控制寄存器	0x0000_0000
CMP_WCNT1	0x30	比较器3的捕捉窗口控制寄存器	0x0000_0000
CMP_WCNT2	0x34	比较器4的捕捉窗口控制寄存器	0x0000_0000
CMP_INPCR0	0x38	比较器0的输入控制寄存器	0x0000_0000
CMP_INPCR1	0x3C	比较器1的输入控制寄存器	0x0000_0000
CMP_INPCR2	0x40	比较器2的输入控制寄存器	0x0000_0000
CMP_INPCR3	0x44	比较器3的输入控制寄存器	0x0000_0000
CMP_INPCR4	0x48	比较器4的输入控制寄存器	0x0000_0000
CMP_TRGCR	0x4C	比较器触发输出控制寄存器	0x0000_0000
CMP_IMCR	0x50	中断使能禁止控制寄存器	0x0000_0000
CMP_RISR	0x54	原始中断状态寄存器	0x0000_0000
CMP_MISR	0x58	中断状态寄存器	0x0000_0000
CMP_ICR	0x5C	中断标志清除寄存器	0x0000_0000

9.3.2 CMP_CEDR (ID和时钟使能控制寄存器)

- Address = Base Address + 0x0000, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDCODE																					RSVD			SWRST	RSVD		CLKEN4	CLKEN3	CLKEN2	CLKEN1	CLKEN0
0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	R	R	R W	R W	R W	R W	R W

Name	Bit	Type	Description
CLKENx	[4:0]	R/W	CMPx 的时钟使能/禁止控制位。 0： 停止控制时钟 1： 使能控制时钟
SWRST	[7]	W	软件复位。 0：没有效果 1：执行软件复位操作
IDCODE/ID_KEY	[31:11]	R	ID Code 寄存器。 这个区域保存了相应 IP 的 IDCODE。

9.3.3 CMP_CR0 (比较器0的控制寄存器)

- Address = Base Address + 0x0004, Reset Value = 0x0000_0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPOUT4	CMPOUT3	CMPOUT2	CMPOUT1	CMPOUT0	RSVD											CPOS	RSVD				FLTEN	EVE_SEL		POLARITY	RSVD	RSVD	PHYST		NHYST		CMPEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

名称	位	类型	描述
CMPEN	[0]	R/W	使能/禁止模拟比较器。 0：禁止模拟比较器 1：使能模拟比较器
NHYST	[2:1]	R/W	比较器负向输入迟滞。 0：0mV 1：75mV 2：100mV 3：150mV
PHYST	[4:3]	R/W	比较器正向输入迟滞。 0：0mV 1：75mV 2：100mV 3：150mV
POLARITY	[7]	R/W	比较器输出极性选择。(1) 0：输出不反向 1：输出反向
EVE_SEL	[9:8]	R/W	事件触发边沿选择。 0：下降沿 1：上升沿 2：下降沿和上升沿 3：下降沿和上升沿
FLTEN	[10]	R/W	输出数字滤波计数器设置。 0：跳过滤波器 1：滤波器使能
CPOS	[15]	R/W	比较器输出管脚输出选择

			0：比较器输出状态 1：比较器滤波后输出状态
CMPOUT0	[27]	R	比较器0输出状态 当 POLARITY=0时（非取反模式） 0：VIN+ < VIN- 1：VIN+ > VIN- 当 POLARITY=1时（取反模式） 0：VIN+ > VIN- 1：VIN+ < VIN-
CMPOUT1	[28]	R	比较器1输出状态
CMPOUT2	[29]	R	比较器2输出状态
CMPOUT3	[30]	R	比较器3输出状态
CMPOUT4	[31]	R	比较器4输出状态

NOTE: 1) 只有在 CMPEN 为‘0’时，才可以改变比较器的极性。当比较器未使能时，对 CMPEN 的写 1 操作和极性配置可以同时进行。

9.3.4 CMP_CR1 (比较器1的控制寄存器)

- Address = Base Address + 0x0008, Reset Value = 0x0000_0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPOUT4	CMPOUT3	CMPOUT2	CMPOUT1	CMPOUT0	RSVD											CPOS	RSVD				FLTEN	EVE_SEL		POLARITY	RSVD	RSVD	PHYST		NHYST		CMPEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

名称	位	类型	描述
CMPEN	[0]	R/W	使能/禁止模拟比较器。 0：禁止模拟比较器 1：使能模拟比较器
NHYST	[2:1]	R/W	比较器负向输入迟滞。 0：0mV 1：75mV 2：100mV 3：150mV
PHYST	[4:3]	R/W	比较器正向输入迟滞。 0：0mV 1：75mV 2：100mV 3：150mV
POLARITY	[7]	R/W	比较器输出极性选择。(1) 0：输出不反向 1：输出反向
EVE_SEL	[9:8]	R/W	事件触发边沿选择。 0：下降沿 1：上升沿 2：下降沿和上升沿 3：下降沿和上升沿
FLTEN	[10]	R/W	输出数字滤波计数器设置。 0：跳过滤波器 1：滤波器使能
CPOS	[15]	R/W	比较器输出管脚输出选择

			0：比较器输出状态 1：比较器滤波后输出状态
CMPOUT0	[27]	R	比较器0输出状态 当 POLARITY=0时（非取反模式） 0：VIN+ < VIN- 1：VIN+ > VIN- 当 POLARITY=1时（取反模式） 0：VIN+ > VIN- 1：VIN+ < VIN-
CMPOUT1	[28]	R	比较器1输出状态
CMPOUT2	[29]	R	比较器2输出状态
CMPOUT3	[30]	R	比较器3输出状态
CMPOUT4	[31]	R	比较器4输出状态

NOTE: 1) 只有在 CMPEN 为‘0’时，才可以改变比较器的极性。当比较器未使能时，对 CMPEN 的写 1 操作和极性配置可以同时进行。

9.3.5 CMP_CR2 (比较器2的控制寄存器)

- Address = Base Address + 0x000C, Reset Value = 0x0000_0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPOUT4	CMPOUT3	CMPOUT2	CMPOUT1	CMPOUT0	RSVD											CPOS	RSVD				FLTEN	EVE_SEL		POLARITY	RSVD	RSVD	PHYST		NHYST		CMPEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

名称	位	类型	描述
CMPEN	[0]	R/W	使能/禁止模拟比较器。 0：禁止模拟比较器 1：使能模拟比较器
NHYST	[2:1]	R/W	比较器负向输入迟滞。 0：0mV 1：75mV 2：100mV 3：150mV
PHYST	[4:3]	R/W	比较器正向输入迟滞。 0：0mV 1：75mV 2：100mV 3：150mV
POLARITY	[7]	R/W	比较器输出极性选择。(1) 0：输出不反向 1：输出反向
EVE_SEL	[9:8]	R/W	事件触发边沿选择。 0：下降沿 1：上升沿 2：下降沿和上升沿 3：下降沿和上升沿
FLTEN	[10]	R/W	输出数字滤波计数器设置。 0：跳过滤波器 1：滤波器使能
CPOS	[15]	R/W	比较器输出管脚输出选择

			0：比较器输出状态 1：比较器滤波后输出状态
CMPOUT0	[27]	R	比较器0输出状态 当 POLARITY=0时（非取反模式） 0：VIN+ < VIN- 1：VIN+ > VIN- 当 POLARITY=1时（取反模式） 0：VIN+ > VIN- 1：VIN+ < VIN-
CMPOUT1	[28]	R	比较器1输出状态
CMPOUT2	[29]	R	比较器2输出状态
CMPOUT3	[30]	R	比较器3输出状态
CMPOUT4	[31]	R	比较器4输出状态

NOTE: 1) 只有在 CMPEN 为‘0’时，才可以改变比较器的极性。当比较器未使能时，对 CMPEN 的写 1 操作和极性配置可以同时进行。

9.3.6 CMP_CR3 (比较器3的控制寄存器)

- Address = Base Address + 0x0010, Reset Value = 0x0000_0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPOUT4	CMPOUT3	CMPOUT2	CMPOUT1	CMPOUT0	RSVD											CPOS	RSVD				FLTEN	EVE_SEL		POLARITY	RSVD	RSVD	PHYST		NHYST		CMPEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

名称	位	类型	描述
CMPEN	[0]	R/W	使能/禁止模拟比较器。 0：禁止模拟比较器 1：使能模拟比较器
NHYST	[2:1]	R/W	比较器负向输入迟滞。 0：0mV 1：75mV 2：100mV 3：150mV
PHYST	[4:3]	R/W	比较器正向输入迟滞。 0：0mV 1：75mV 2：100mV 3：150mV
POLARITY	[7]	R/W	比较器输出极性选择。(1) 0：输出不反向 1：输出反向
EVE_SEL	[9:8]	R/W	事件触发边沿选择。 0：下降沿 1：上升沿 2：下降沿和上升沿 3：下降沿和上升沿
FLTEN	[10]	R/W	输出数字滤波计数器设置。 0：跳过滤波器 1：滤波器使能
CPOS	[15]	R/W	比较器输出管脚输出选择

			0：比较器输出状态 1：比较器滤波后输出状态
CMPOUT0	[27]	R	比较器0输出状态 当 POLARITY=0时（非取反模式） 0：VIN+ < VIN- 1：VIN+ > VIN- 当 POLARITY=1时（取反模式） 0：VIN+ > VIN- 1：VIN+ < VIN-
CMPOUT1	[28]	R	比较器1输出状态
CMPOUT2	[29]	R	比较器2输出状态
CMPOUT3	[30]	R	比较器3输出状态
CMPOUT4	[31]	R	比较器4输出状态

NOTE: 1) 只有在 CMPEN 为‘0’时，才可以改变比较器的极性。当比较器未使能时，对 CMPEN 的写 1 操作和极性配置可以同时进行。

9.3.7 CMP_CR4 (比较器4的控制寄存器)

- Address = Base Address + 0x0014, Reset Value = 0x0000_0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPOUT4	CMPOUT3	CMPOUT2	CMPOUT1	CMPOUT0	RSVD											CPOS	RSVD				FLTEN	EVE_SEL		RSVD	RSVD	BOOST	PHYST		NHYST		CMPEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

名称	位	类型	描述
CMPEN	[0]	R/W	使能/禁止模拟比较器。 0：禁止模拟比较器 1：使能模拟比较器
NHYST	[2:1]	R/W	比较器负向输入迟滞。 0：0mV 1：75mV 2：100mV 3：150mV
PHYST	[4:3]	R/W	比较器正向输入迟滞。 0：0mV 1：75mV 2：100mV 3：150mV
POLARITY	[7]	R/W	比较器输出极性选择。(1) 0：输出不反向 1：输出反向
EVE_SEL	[9:8]	R/W	事件触发边沿选择。 0：下降沿 1：上升沿 2：下降沿和上升沿 3：下降沿和上升沿
FLTEN	[10]	R/W	输出数字滤波计数器设置。 0：跳过滤波器 1：滤波器使能
CPOS	[15]	R/W	比较器输出管脚输出选择

			0：比较器输出状态 1：比较器滤波后输出状态
CMPOUT0	[27]	R	比较器0输出状态 当 POLARITY=0时（非取反模式） 0：VIN+ < VIN- 1：VIN+ > VIN- 当 POLARITY=1时（取反模式） 0：VIN+ > VIN- 1：VIN+ < VIN-
CMPOUT1	[28]	R	比较器1输出状态
CMPOUT2	[29]	R	比较器2输出状态
CMPOUT3	[30]	R	比较器3输出状态
CMPOUT4	[31]	R	比较器4输出状态

NOTE: 1) 只有在 CMPEN 为‘0’时，才可以改变比较器的极性。当比较器未使能时，对 CMPEN 的写 1 操作和极性配置可以同时进行。

9.3.8 CMP_FLTCR0（比较器0的数字滤波控制寄存器)

- Address = Base Address + 0x00018, Reset Value = 0x0000_0000

9.3.9 CMP_FLTCR1（比较器1的数字滤波控制寄存器)

- Address = Base Address + 0x001C, Reset Value = 0x0000_0000

9.3.10 CMP_FLTCR2（比较器2的数字滤波控制寄存器)

- Address = Base Address + 0x0020, Reset Value = 0x0000_0000

9.3.11 CMP_FLTCR3（比较器3的数字滤波控制寄存器)

- Address = Base Address + 0x0024, Reset Value = 0x0000_0000

9.3.12 CMP_FLTCR4（比较器4的数字滤波控制寄存器)

- Address = Base Address + 0x0028, Reset Value = 0x0000_0000

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

名称	位	类型	描述
CKSRC	[2:0]	R/W	数字滤波器时钟源选择。 0：PCLK 1：TC1 PEND 触发 2：TC2 PEND 触发 其他：无效
DIVN	[7:3]	R/W	数字滤波器时钟分频系数 N (1)
DIVM	[15:8]	R/W	数字滤波器时钟分频系数 M (1)

NOTE: 数字滤波器的时钟 FLT_CK = PCLK/(M+1)/2^N

9.3.13 CMP_WCNT0 (比较器2的捕捉窗口控制寄存器)

- Address = Base Address + 0x002C, Reset Value = 0x0000_0000

9.3.14 CMP_WCNT1 (比较器3的捕捉窗口控制寄存器)

- Address = Base Address + 0x0030, Reset Value = 0x0000_0000

9.3.15 CMP_WCNT2 (比较器4的捕捉窗口控制寄存器)

- Address = Base Address + 0x0034, Reset Value = 0x0000_0000

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

名称	位	类型	描述
WCNT	[9:0]	R/W	设置捕捉窗口宽度计数器。在触发后，当延时计数器计数完成后，内部10位计数器根据 CLKDIV 的设置频率计数。窗口的宽度为：(WCNT+1)x Twcnt
CLKDIV	[15:10]	R/W	设置捕捉滤波器的频率 Fwcnt。

Value	DIV	Value	DIV
0	不分频	28	Div208
1	Div2	29	Div224
2	Div3	30	Div240
3	Div4	31	Div256
4	Div5	32	Div288
……	……	33	Div320
15	Div16	34	Div352
16	Div24	35	Div384
17	Div32	36	Div416
18	Div40	37	Div448
19	Div48	38	Div480
20	Div56	39	Div512
21	Div64	40	Div640
22	Div72	41	Div720
23	Div128	42	Div1024

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td>24</td><td>Div144</td><td>43</td><td>Div2048</td></tr>
<tr><td>25</td><td>Div160</td><td>Other</td><td>不分频</td></tr>
<tr><td>26</td><td>Div176</td><td></td><td></td></tr>
<tr><td>27</td><td>Div192</td><td></td><td></td></tr>
</table>
Fwcnt 的分频是基于比较器模块 PCLK 时钟的分频。
</td></tr>
<tr><td>DCNT</td><td>[23:16]</td><td>R/W</td><td>设置捕捉窗口延时计数器。在触发后，内部8位递减计数器根据 CLKDIV 的设置频率计数。延时的时间为(DCNT+1) x Twcnt。
注意：当 DCNT 为零时，延时窗口被关闭，而不是1个 Twcnt。</td></tr>
<tr><td>HLS</td><td>[24]</td><td>R/W</td><td>滤波器初次触发前的缺省输出相位。
0：低电平输出
1：高电平输出</td></tr>
<tr><td>MSKMOD</td><td>[27:25]</td><td>R/W</td><td>捕获滤波器控制
<table>
<tr><td>Value</td><td>IDLE</td><td>DPHS</td></tr>
<tr><td>000</td><td colspan="2">Skip filter</td></tr>
<tr><td>001</td><td>LOW</td><td>LOW</td></tr>
<tr><td>010</td><td>HIGH</td><td>LOW</td></tr>
<tr><td>011</td><td>HOLD</td><td>LOW</td></tr>
<tr><td>100</td><td colspan="2">Skip filter</td></tr>
<tr><td>101</td><td>LOW</td><td>HIGH</td></tr>
<tr><td>110</td><td>HIGH</td><td>HIGH</td></tr>
<tr><td>111</td><td>HOLD</td><td>HIGH</td></tr>
</table>
IDLE：在窗口结束后，但未有被触发前的区间。该区间的初始极性电平可以通过 WCNT 中的 HLS 位进行设置。</td></tr>
<tr><td>TRGSEL</td><td>[31:28]</td><td>R/W</td><td>捕获滤波器触发信号选择（来自 EPWM 模块）。
0：PWM_START 事件触发
1：PWM_STOP 事件触发
2：PWM_PEND 事件触发
3：PWM_CENTER 事件触发
4：PWM0_CMPAUM 事件触发
5：PWM0_CMPADM 事件触发
6：PWM0_CMPBUM 事件触发
7：PWM0_CMPBDM 事件触发
8：PWM1_CMPAUM 事件触发
9：PWM1_CMPADM 事件触发
10：PWM1_CMPBUM 事件触发
11：PWM1_CMPBDM 事件触发
12：PWM2_CMPAUM 事件触发
13：PWM2_CMPADM 事件触发
14：PWM2_CMPBUM 事件触发
15：PWM2_CMPBDM 事件触发</td></tr>
</table>

9.3.16 CMP_INPCR0 (比较器0的输入控制寄存器)

- Address = Base Address + 0x0038, Reset Value = 0x0000_0000

9.3.17 CMP_INPCR1 (比较器1的输入控制寄存器)

- Address = Base Address + 0x003C, Reset Value = 0x0000_0000

9.3.18 CMP_INPCR2 (比较器2的输入控制寄存器)

- Address = Base Address + 0x0040, Reset Value = 0x0000_0000

9.3.19 CMP_INPCR3 (比较器3的输入控制寄存器)

- Address = Base Address + 0x0044, Reset Value = 0x0000_0000

9.3.20 CMP_INPCR4 (比较器4的输入控制寄存器)

- Address = Base Address + 0x0048, Reset Value = 0x0000_0000

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

名称	位	类型	描述
NSEL	[7:0]	R/W	比较器负向输入选择。 0：GND 1：INT_REF = VDD/256 * 1 2：INT_REF = VDD/256 * 2 …… 246：INT_REF = VDD/256 * 246 247：FVR 248：CPINN0 249：CPINN1 250：CPINN2〉 251：CPINN3〉 252：CPINN4 253：Not used 254：Not used 255：RSVD〈1〉

PSEL	[11:8]	R/W	比较器正向输入选择。 0：CPINP0 1：CPINP1 …… 9：CPINP9 Others：Not used 13：OPA1X 14：OPA0X 15：RSVD（1）

NOTE:

1）工程模式预留

9.3.21 CMP_TRGCR (比较器触发输出控制寄存器)

- Address = Base Address + 0x004C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																			AD_TRG4	AD_TRG3	AD_TRG2	AD_TRG1	AD_TRG0	TC_CIN4	TC_CIN3	TC_CIN2	TC_CIN1	TC_CIN0	TC_TRG		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
TC_TRG	[2:0]	R/W	选择比较器的输出作为 TC1的启动触发输入。 0：不输出 1：CMPOUT0作为 TC1的启动触发 2：CMPOUT1作为 TC1的启动触发 3：CMPOUT2作为 TC1的启动触发 4：CMPOUT3作为 TC1的启动触发 5：CMPOUT4作为 TC1的启动触发 Others：不输出
TC_CINx	[7:3]	R/W	TCIN0: CMP0输出作为 TC1的 Capture/CLK 输入。 TCIN1: CMP1输出作为 TC1的 Capture/CLK 输入。 TCIN2: CMP2输出作为 TC1的 Capture/CLK 输入。 TCIN3: CMP3输出作为 TC1的 Capture/CLK 输入。 TCIN4: CMP4输出作为 TC1的 Capture/CLK 输入。 0：不输出（保持低电平） 1：输出使能
AD_TRGx	[12:8]	R/W	CMPx 的 ADC 转换硬件触发使能控制。 0：禁止 CMP 硬件触发 ADC 1：使能 CMP 硬件触发 ADC

9.3.22 CMP_IMCR (中断使能禁止控制寄存器)

- Address = Base Address + 0x0050, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																											EDGEDET4	EDGEDET3	EDGEDET2	EDGEDET1	EDGEDET0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW

Name	Bit	Type	Description
EDGEDET0	[0]	R/W	CMP0的输出边沿检测中断使能控制。 0：禁止中断发生 1：使能中断发生
EDGEDET1	[1]	R/W	CMP1的输出边沿检测中断使能控制。 0：禁止中断发生 1：使能中断发生
EDGEDET2	[2]	R/W	CMP2的输出边沿检测中断使能控制。 0：禁止中断发生 1：使能中断发生
EDGEDET3	[3]	R/W	CMP3的输出边沿检测中断使能控制。 0：禁止中断发生 1：使能中断发生
EDGEDET4	[4]	R/W	CMP4的输出边沿检测中断使能控制。 0：禁止中断发生 1：使能中断发生

9.3.23 CMP_RISR (原始中断状态寄存器)

- Address = Base Address + 0x0054, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																											EDGEDET4	EDGEDET3	EDGEDET2	EDGEDET1	EDGEDET0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
EDGEDET0	[0]	R	CMP0的输出边沿检测中断原始状态位。 0：中断未发生 1：中断发生
EDGEDET1	[1]	R	CMP1的输出边沿检测中断原始状态位。 0：中断未发生 1：中断发生
EDGEDET2	[2]	R	CMP2的输出边沿检测中断原始状态位。 0：中断未发生 1：中断发生
EDGEDET3	[3]	R	CMP3的输出边沿检测中断原始状态位。 0：中断未发生 1：中断发生
EDGEDET4	[4]	R	CMP4的输出边沿检测中断原始状态位。 0：中断未发生 1：中断发生

9.3.24 CMP_MISR (中断状态寄存器)

- Address = Base Address + 0x0058, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																											EDGEDET4	EDGEDET3	EDGEDET2	EDGEDET1	EDGEDET0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
EDGEDET0	[0]	R	CMP0的输出边沿检测中断状态位。 0：中断未发生 1：中断发生
EDGEDET1	[1]	R	CMP1的输出边沿检测中断状态位。 0：中断未发生 1：中断发生
EDGEDET2	[2]	R	CMP2的输出边沿检测中断状态位。 0：中断未发生 1：中断发生
EDGEDET3	[3]	R	CMP3的输出边沿检测中断状态位。 0：中断未发生 1：中断发生
EDGEDET4	[4]	R	CMP4的输出边沿检测中断状态位。 0：中断未发生 1：中断发生

9.3.25 CMP_ICR (中断标志清除寄存器)

- Address = Base Address + 0x005C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																											EDGEDET4	EDGEDET3	EDGEDET2	EDGEDET1	EDGEDET0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	W

Name	Bit	Type	Description
EDGEDET0	[0]	W	CMP0的输出边沿检测中断状态清除位。（只有写入时有效） 0：无效果 1：清除 CMP0中断标志位
EDGEDET1	[1]	W	CMP1的输出边沿检测中断状态清除位。（只有写入时有效） 0：无效果 1：清除 CMP1中断标志位
EDGEDET2	[2]	W	CMP2的输出边沿检测中断状态清除位。（只有写入时有效） 0：无效果 1：清除 CMP2中断标志位
EDGEDET3	[3]	W	CMP3的输出边沿检测中断状态清除位。（只有写入时有效） 0：无效果 1：清除 CMP3中断标志位
EDGEDET4	[4]	W	CMP4的输出边沿检测中断状态清除位。（只有写入时有效） 0：无效果 1：清除 CMP4中断标志位

10 模拟运算放大器（OP-AMP）

10.1 概述

芯片内部集成了两个运算放大器（OPA0 和 OPA1），可用于特定模拟信号的处理。通过内部寄存器设置，可以配置运算放大器工作在不同的工作方式。例如单位增益缓冲器，同相放大器，反向放大器和各种滤波器等。当运算放大器的三个端口全部使能时，需要增加外部反馈电路才能使得运算放大器正常工作；如果采用同相放大模式，则可以选择内部增益控制。无论何种模式工作，运算放大器的输出端必须使能。

10.1.1 特性

- 支持最大2个独立运算放大器。
- 每个运算放大器可以支持外部独立工作，或者内部增益控制模式。
- 可配置的内部增益放大系数
  - OPA0：X1 / X2 / X3 / X4 / X5 / X6 / X7 / X8。
  - OPA1：X1 / X10 / X20 / X40 / X60 / X80 / X100 / X120。

10.1.2 管脚描述

Table 10-1 OPA 管脚描述

管脚名称	功能	I/O类型	有效状态	备注
OPA0X	运算放大器0的输出通道	A	-	-
OPA0P	运算放大器0的正向输入通道	A	-	-
OPA0N	运算放大器0的负向输入通道	A	-	-
OPA1X	运算放大器1的输出通道	A	-	-
OPA1P	运算放大器1的正向输入通道	A	-	-
OPA1N	运算放大器1的负向输入通道	A	-	-

10.2 功能描述

10.2.1 运算放大器功能模块

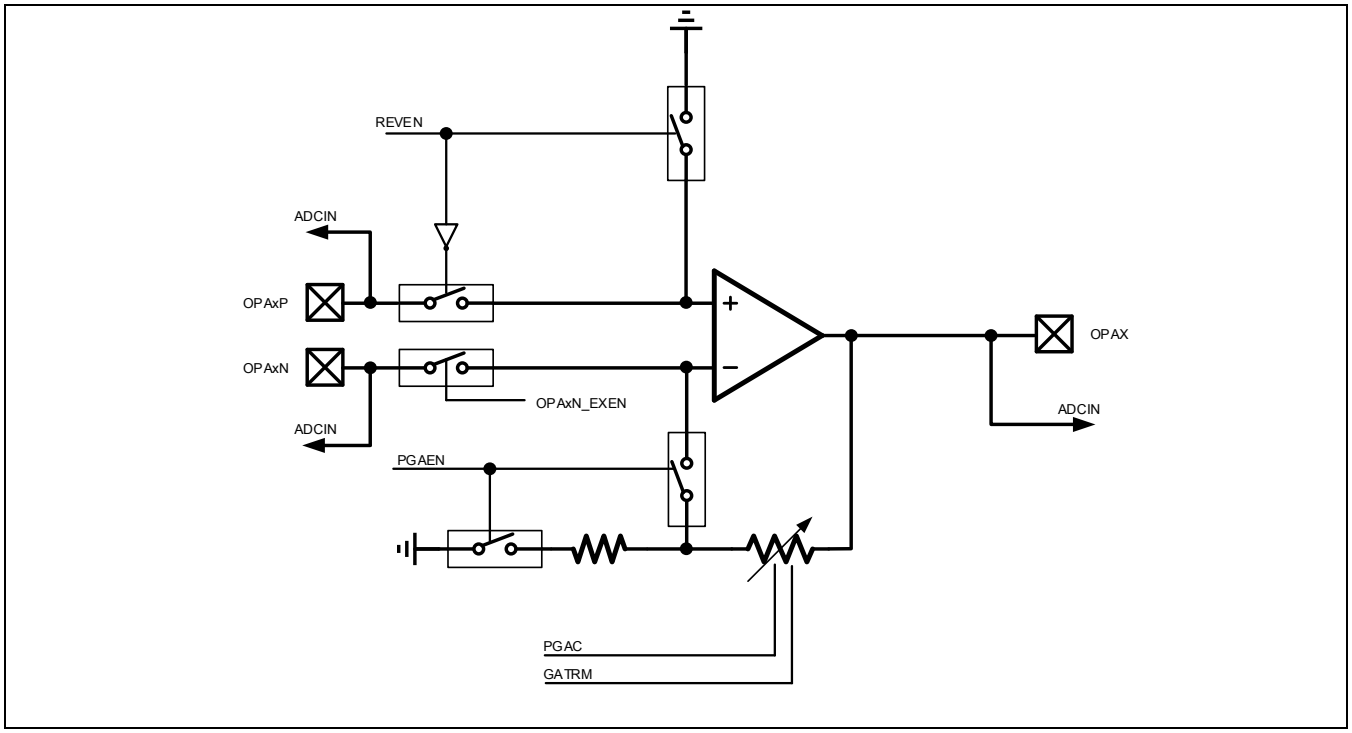


Figure 10-1 运算放大器框图

10.2.2 运算放大器工作模式

芯片内建两个高性能运算放大器，当不使用时，可以通过控制寄存器（OPA_CR）中的OPAEN位来实现软件关闭功能。每个运放提供三个外接引脚：OPAxP（正输入端口），OPAxN（负输入端口），OPAxX（输出端口）。运放使能需要先设置对应的输出引脚，只有在对应的输出端口（OPAxX）引脚选择为AF7（模拟信号通路）后，运放才能使能。在OPAxX功能未正确设置前，即使OPAEN控制位已经被置位，运放仍处于关闭状态。

运放的输入端口和ADC以及模拟比较器的正向输入端复用，当对应的GPIO设置为AF7时，可以同时使用该引脚上的所有模拟功能（输入端口的模拟信号，可以同时作为运放，ADC或者比较器的输入）。运放的输出端口和ADC复用，当对应的GPIO设置为AF7时，运放的输出可以通过选择该引脚上的ADC通道使能ADC采样。在使用ADC对运放输出进行采样时，如果噪声较大，可以在对应的GPIO管脚上增加一个对地的辅助电容，以保证ADC采样的准确性，电容的建议值为：1~10nF，可根据运放输出的频率进行调整。

运放支持内部增益控制模式工作（PGAEN=1），当选择内部增益控制时，可以节省运放的一个输入引脚。内部增益只支持同相放大模式。内部增益模式下，如果希望使用负向输入管脚，也可以将OPAxN_EXEN置1，来使能IO上的负向输入功能。

运放也支持外部部增益控制模式工作（PGAEN=0，且OPAxN_EXEN=1使能负向输入端），当选择外部增益控制时，可以通过外部电阻网络来控制环路增益。

选择外部增益时还可以支持反向放大应用，运放的正向输入端口可以在内部接地（REVEN=1）。增益调节由外部电阻网络决定。

在低功耗模式下，运放被强制关闭。

10.2.3 运算放大器作为ADC输入的典型应用

如下图，运算放大器使用内部增益模式，并且运算放大器的输出作为ADC的输入，同时对应的GPIO管脚外接一个1~10nF的电容（可选）。

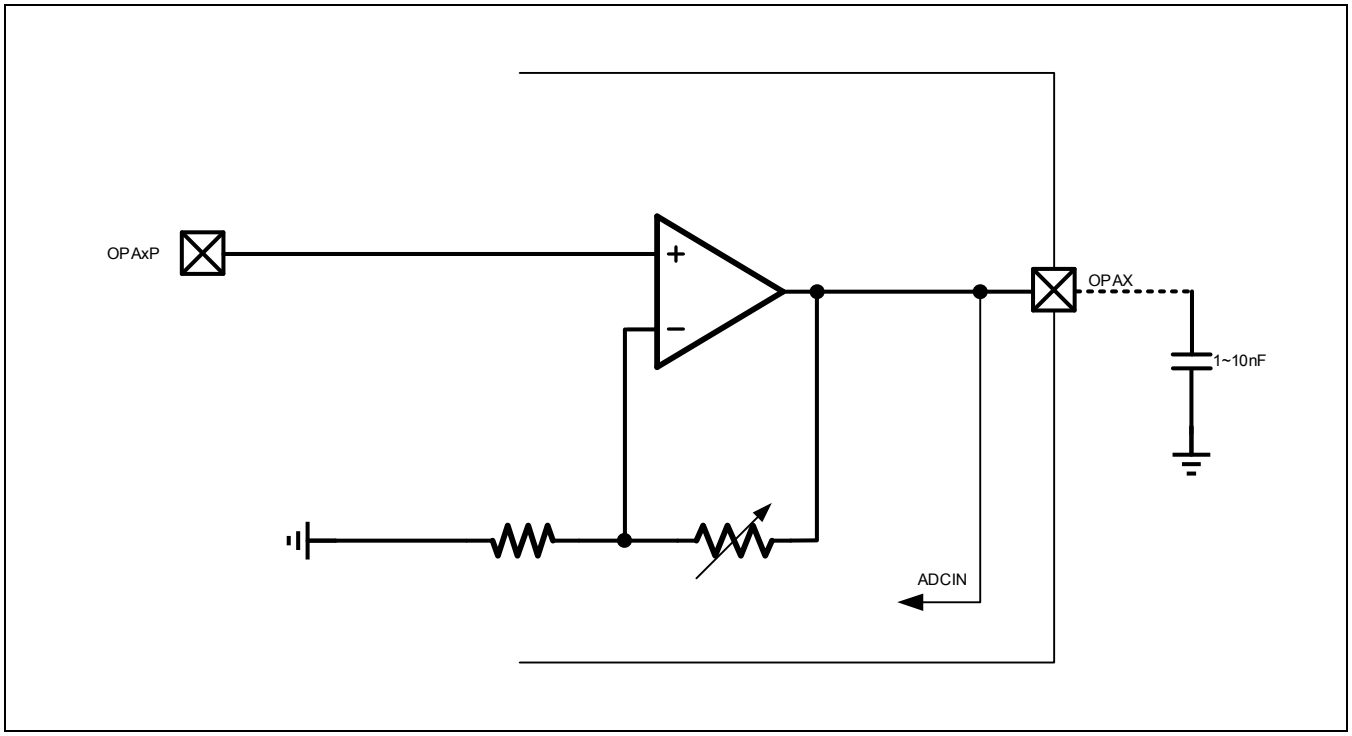


Figure 10-2 运算放大器作为 ADC 输入

需要进行的配置为：

1. 相应的GPIO设置为AF7模拟信号输入输出功能(运放输出以及ADC输入)。
2. 相应的运算放大器使能内部增益 (PGAEN=1)，选择需要的放大倍数。
3. ADC选择对应的ADCIN输入通道 (ADCIN10/11)，并且根据需求选择合适的转换时钟和转换模式。

10.3 寄存器说明

10.3.1 寄存器表

- Base Address: 0x400C_0000

Table 10-2 寄存器表

Register	Offset	Description	Reset Value
OPA0_CR	0x00	运算放大器0控制寄存器	0x0000_0000
OPA1_CR	0x04	运算放大器1控制寄存器	0x0000_0000
OPA0_IGCR	0x08	运算放大器0内部增益控制寄存器	0x0000_0007
OPA1_IGCR	0x0C	运算放大器1内部增益控制寄存器	0x0000_0007

NOTE:

1. OPA 的输入失调电压在出厂时已经进行了校准，用户还可以根据应用需求通过 SYSCON 内的相应寄存器进行微调。
2. OPA 的内部增益可以通过软件进行微调，出厂时不进行校准。

10.3.2 OPA0_CR (运算放大器0控制寄存器)

- Address = Base Address + 0x0000, Reset Value = 0x0000_0000

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

名称	位	类型	描述
OPAEN	[0]	R/W	OPA 模块的使能控制。 0： 禁止 OPA 模块 1： 使能 OPA 模块
PGAEN	[1]	RW	内部增益控制使能。(1) 0：禁止内部增益控制 1：使能内部增益控制
PGAC	[4:2]	RW	内部增益控制设置。 0：X 1 1：X 2 2：X 3 3：X 4 4：X 5 5：X 6 6：X 7 7：X 8
IPSEL	[9:8]	RW	INP 输入端控制，选择正向输入端信号源。(2) 0：外部输入端口 OPA0P 1：RSVD 2：BGR 输出（1.5V） 3：内部模拟地

NOTE:

1） 内部增益控制只支持正向输入的情况。

2） 当使用运放负向输入时，内部增益控制必须关闭，且 IPSEL 选择内部模拟地，OPAx_IGCR 寄存器中的 OPAxN_EXEN 必须写1。同时外部必须增加电阻进行增益控制。

10.3.3 OPA1_CR (运算放大器1控制寄存器)

- Address = Base Address + 0x0004, Reset Value = 0x0000_0000

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

名称	位	类型	描述
OPAEN	[0]	R/W	OPA 模块的使能控制。 0： 禁止 OPA 模块 1： 使能 OPA 模块
PGAEN	[1]	RW	内部增益控制使能。(1) 0：禁止内部增益控制 1：使能内部增益控制
PGAC	[4:2]	RW	内部增益控制设置。 0：X 1 1：X 10 2：X 20 3：X 40 4：X 60 5：X 80 6：X 100 7：X 120
REVEN	[8]	RW	负向输入模式控制。(2) 0：正向输入模式。 1：负向输入模式，正向输入端（INP）内部接地。

NOTE:

1）内部增益控制只支持正向输入的情况。当使用运放负向输入时，内部增益控制必须关闭，且使能 REVEN。

2）当使用运放负向输入时，内部增益控制必须关闭，且使能 REVEN，OPAx_IGCR 寄存器中的 OPAxN_EXEN 必须写1。同时外部必须增加电阻进行增益控制。

10.3.4 OPA0_IGCR (运算放大器0内部增益控制寄存器)

- Address = Base Address + 0x0008, Reset Value = 0x0000_0007

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRM_KEY																RSVD												OPA0N_EXEN	GATRM		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW

名称	位	类型	描述
GATRM	[2:0]	R/W	增益放大微调。
OPA0N_EXEN	[3]	R/W	使用内部增益时，是否将运算放大器的负向输入接到 IO 端口 0: 禁止 1: 使能 在使用内部增益时，如果需要在输出端和负向输入端增加外部电阻或者电容，必须将该位写1。
TRM_KEY	[31:16]	W	对本寄存器进行写操作时，需要填入对应的 KEY 值。只有在 KEY 等于0xA77A 时，对本寄存器的写入才有效。

10.3.5 OPA1_IGCR (运算放大器1内部增益控制寄存器)

- Address = Base Address + 0x000C, Reset Value = 0x0000_0007

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRM_KEY																RSVD												OPA1N_EXEN	GATRM		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW

名称	位	类型	描述
GATRM	[2:0]	R/W	增益放大微调。
OPA1N_EXEN	[3]	R/W	使用内部增益时，是否将运算放大器的负向输入接到 IO 端口 0: 禁止 1: 使能 在使用内部增益时，如果需要在输出端和负向输入端增加外部电阻或者电容，必须将该位写1。
TRM_KEY	[31:16]	W	对本寄存器进行写操作时，需要填入对应的 KEY 值。只有在 KEY 等于0xA77A 时，对本寄存器的写入才有效。

11 I/O

11.1 概述

本章节介绍了通用 I/O 口的使用。所有的 I/O 口都可以通过相应的 GPIO 寄存器进行模块化配置。GPIO 寄存器也提供中断信号的配置。每一个通用 I/O 管脚 (GPIOs) 都有如下特征。

- Port A0: 16 位输入/输出端口, PA0.0 ~ PA0.15
- Port A1: 6 位输入/输出端口, PA1.0 ~ PA1.5
- Port B0: 2 位输入/输出端口, PB0.0 ~ PB0.1
- Port C0: 4 位输入/输出端口, PC0.0 ~ PC0.3
- Port D0: 2 位输入/输出端口, PD0.0 ~ PD0.1

每个端口都可通过软件配置以符合不同种类系统和设计的需求。用户应在应用程序之前完成相应端口配置。如果不需要复用各个引脚，也可配置成简易 I/O 模式。

11.1.1 主要特性

- 5种 I/O 模式:
    - 禁止输入 & 输出模式（高阻）
    - 输入模式.
    - 输出模式(禁止输入)
    - 带输入监测的输出模式（输出的同时输入路径也使能）
    - 多功能复用模式
- 在各个模式下，都可对上拉/下拉进行使能/禁用
- 输出模式下，推挽式输出和开漏式输出可选（输出模式和复用功能无关，可单独配置）
- 每一个 I/O 口都可被配置成外部中断源
- 每一个管脚可以独立设置驱动能力
- 其中11个管脚可以支持恒流源驱动模式
- 其中 8 个管脚（PA0.0, PB0.0, PB0.1, PC0.0, PC0.1, PA1.3 ~ PA1.5）支持灌入大电流

11.1.2 管脚描述

Table 11-1 管脚描述

Pin Name	Function	I/O Type	Active Level	Comments
PA0[15:0]	通用 I/O 口 A0	I/O	-	-
PA1[5:0]	通用 I/O 口 A1	I/O	-	-
PB0[1:0]	通用 I/O 口 B0	I/O	-	-
PC0[3:0]	通用 I/O 口 C0	I/O	-	-
PD0[1:0]	通用 I/O 口 D0	I/O	-	-

注意：

1)大部分 I/O 口在复位后处于禁用状态，SWD 调试的管脚默认为输入且弱上拉使能。

11.2 功能描述

11.2.1 电路图

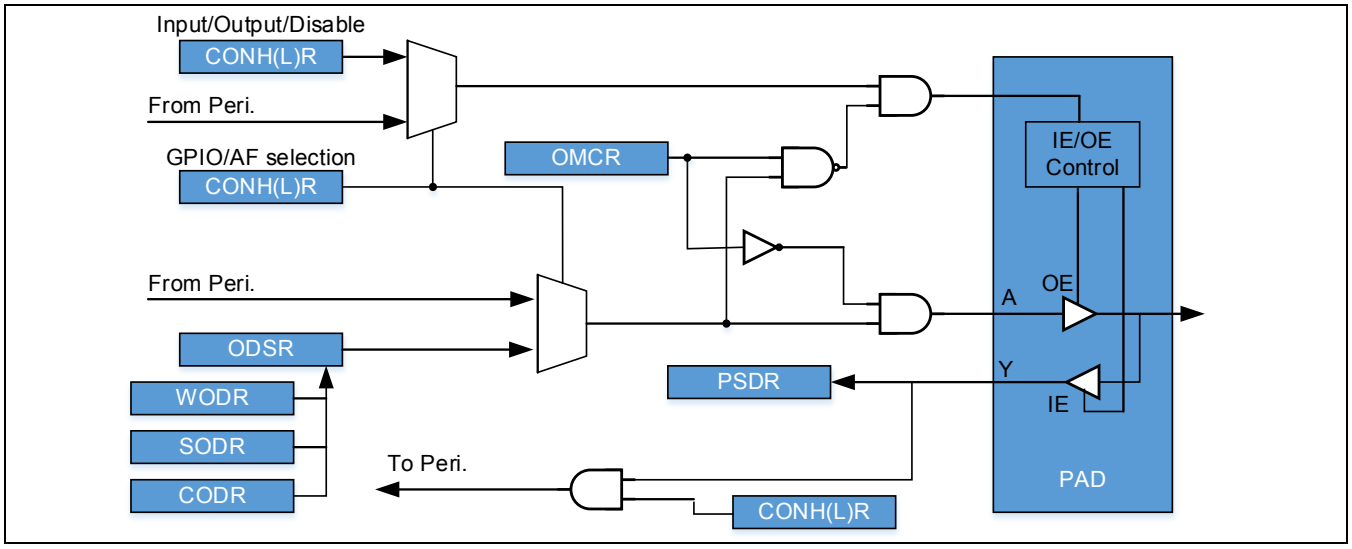


Figure 11-1 GPIO原理图

11.2.2 工作原理

11.2.2.1 功能性描述

I/O 管脚共有 0 ~ 15 种复用功能，可通过 CONLR 和 CONHR 寄存器进行配置。作为 GPIO 功能使用之前，需通过相应 I/O 口的寄存器（CONLR 和 CONHR）配置成 GPIO 模式。

各个管脚功能都可以通过寄存器(CONLR 和 CONHR)按位或者整体进行配置，例如使能，禁止或复用功能。

11.2.2.2 输入与输出的配置

当 I/O 口处于输出状态下，被设置成 GPIO 功能，其输入功能也可被使能或禁止；反之亦然。I/O 口可被配置成如下 4 种模式:

- 输入模式（禁止输出）。这是最常用的输入模式，输入施密特触发器被使能，输入数据可从寄存器PSDR 中被读取
- 输出模式（禁止输入）。这是最常用的输出模式，输出数据被移入寄存器 ODSR 中，并且寄存器 PSDR 被置 0；与此同时，输入路径也被禁止。
- 带输入监测的输出模式（输出的同时输入路径使能）。在这种模式下，寄存器 PSDR 会实时监测管脚的状态。在某些特殊应用中，该模式可用于软件对输出数据的错误校验。
- 禁止输入和输出模式。在这种模式下，管脚处于高阻状态。该模式为芯片复位后多数管脚的默认模式。

当 IO 处于输出模式中，有如下两种方式可用于设置或清除输出数据。第一种是直接写输出值方式，任何写入寄存器 GPIO_WODR 中的值将会被映射到输出寄存器 GPIO_ODSR 中（不论是高电平还是低电平）。第二种方式是通过设置 GPIO_SODR 和 GPIO_CODR 这组寄存器来设置或者清除 GPIO_ODSR 中的相应位。由于将清除和置位两种操作独立，所以可以容易实现对整个 GPIO 组中的单独位进行控制，以弥补 CPU 不能直接支持位操作的不足。对于频繁改变某一个 IO 输出值的场合，使用这种方法，可以有效缩减代码长度。

寄存器 GPIO_ODSR 存储着输出数据，寄存器 GPIO_PSDR 存储着输入数据。

当 GPIO 输出时，大电流灌入使能功能可通过寄存器 GPIO_DSCR 设置。对于输出模式可以通过 GPIO_OMCR 进行配置。每个 I/O 口上都带有内部弱上拉和弱下拉功能，可以通过 GPIO_PUDR 寄存器进行设置。

11.2.3 工作模式

GPIO 只有在工作状态下才可以进行操作和配置。 GPIO 由时钟 PCLK 驱动。可以通过断开 GPIO 的时钟来减少功耗。当系统工作模式改变时（进入或退出 SLEEP/DEEP-SLEEP 模式），I/O 上的配置和状态都不会改变。

11.2.4 外部中断与唤醒功能

通过设置寄存器 GPIO_IECR 和 GPIO_IGRP，任何一个 GPIO 管脚都可以设置成外部中断源。当指定 GPIO 的 EXI 功能被使能，即使当前 GPIO 设置为 AF 复用功能，只要该 GPIO 的 GPIO_IECR 设置位被使能，该 GPIO 的 IO 输入变化也可以触发外部中断。例如：特定 GPIO 被程序设置为 RXD 复用功能，当该 GPIO 的 IECR 被使能后，该 GPIO 口可以通过 RXD 的变化触发外部中断。

中断触发方式可由与 SYSCON EXI 相关的控制寄存器来进行设置。中断路径中的自适应噪声滤波器能对输入信号进行去抖处理。去抖处理不依赖于系统时钟，当系统处于 DEEP-SLEEP 模式时，仍旧有效。所有的 GPIO 按后缀进行分组。每组中 4 个管脚中的其中一个都可以通过配置寄存器 GPIO_IGRP 来被设置成 EXI。

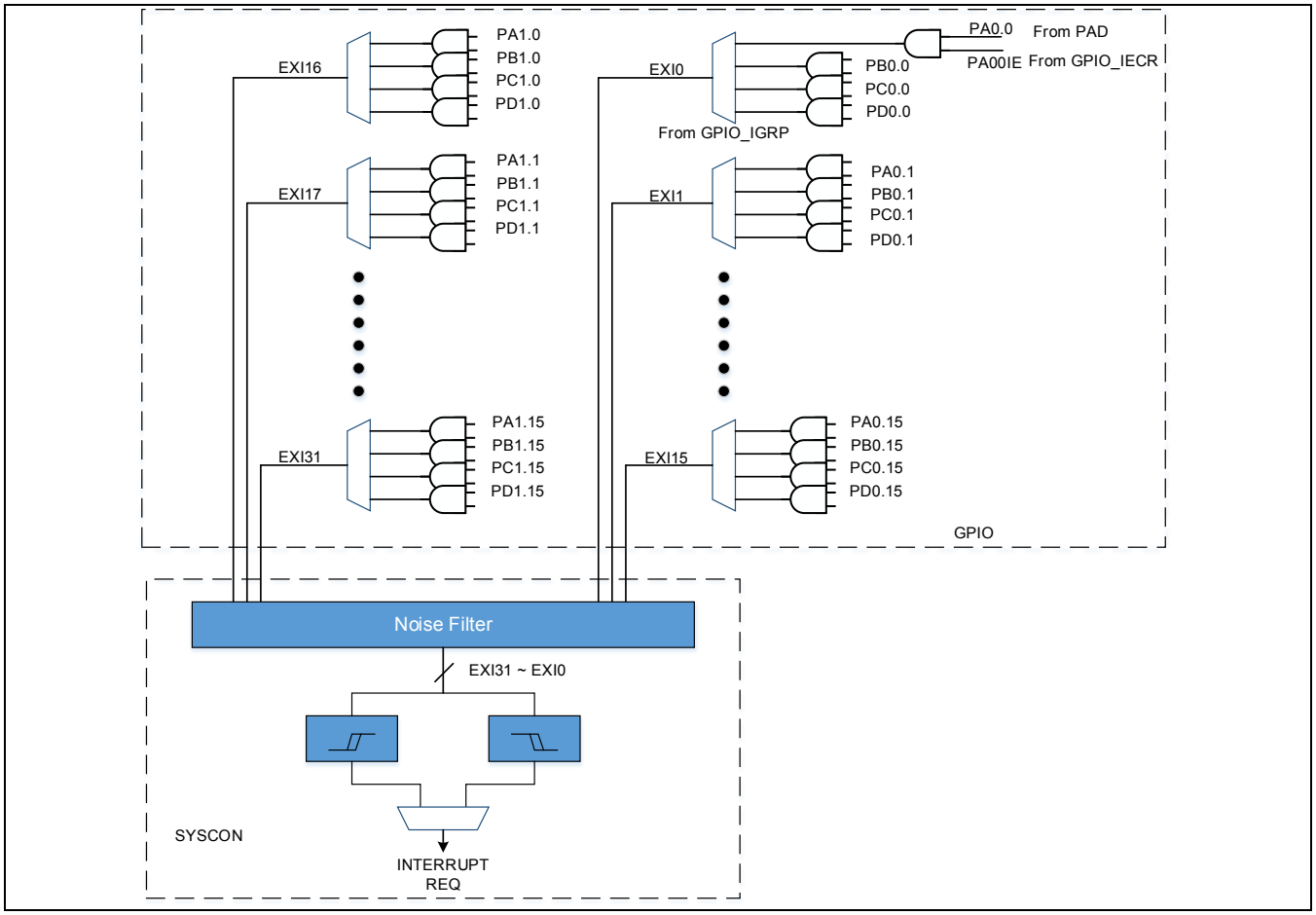


Figure 11-2 GPIO外部中断原理图

当芯片处于 SLEEP 模式或 DEEP-SLEEP 模式下，GPIO 可以被作为唤醒源使用。当需要使能 GPIO 外部中断功能时， GPIO 外部中断功能应该通过 GPIO_IECR 寄存器来使能，并且相应的 EXI 组需要通过 SYSCON_EXIER 寄存器设置为中断使能。相对应的 IRQ 需要在 CPU 中使能为唤醒源。

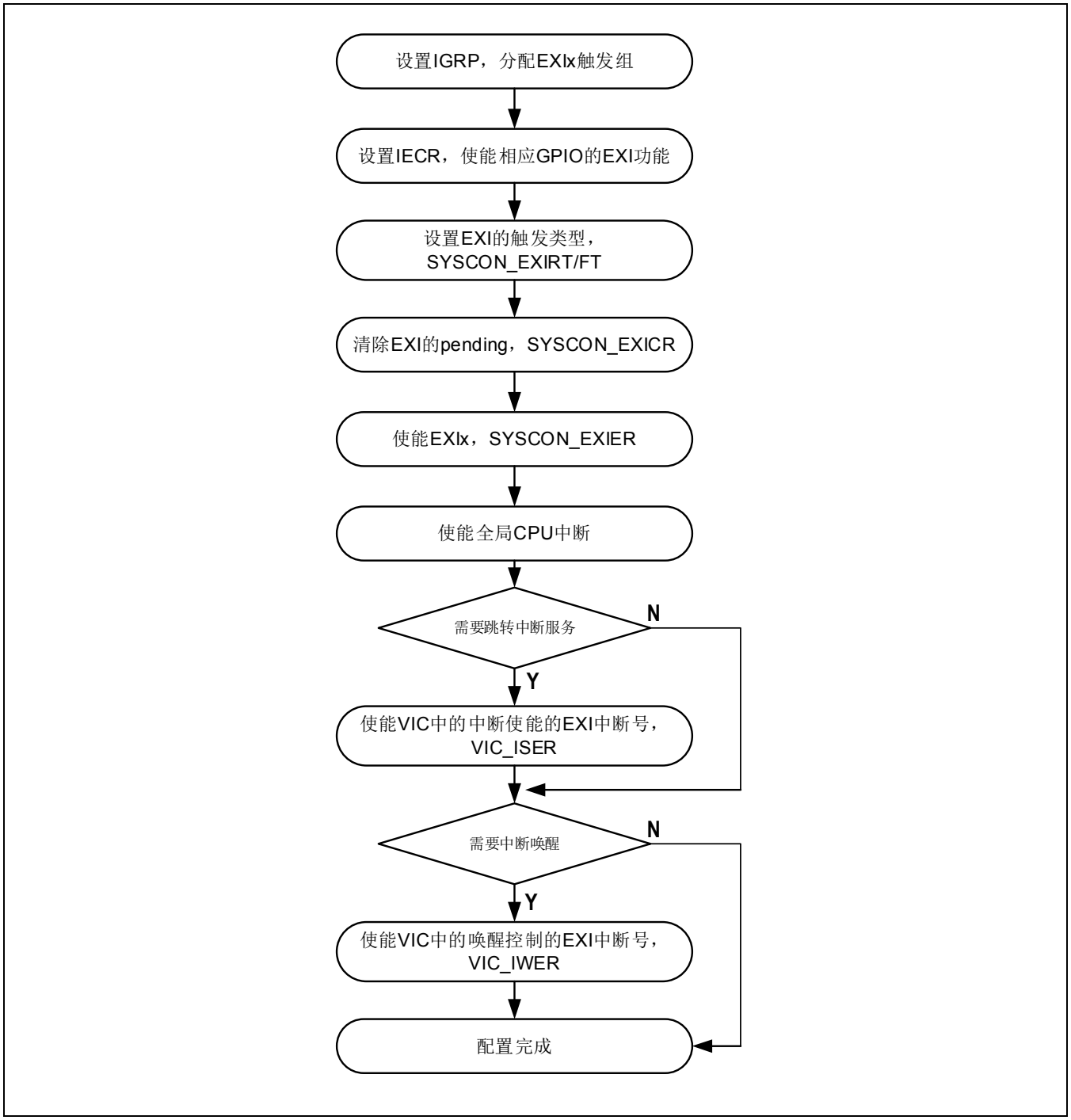


Figure 11-3 GPIO外部中断配置流程

11.3 寄存器说明

11.3.1 寄存器表

- Base Address of A0: 0x4004_0000
- Base Address of A1: 0x4004_0100
- Base Address of B0: 0x4004_1000
- Base Address of C0: 0x4004_2000
- Base Address of D0: 0x4004_3000

Register	Offset	Description	Reset Value
GPIO_CONLR	0x00	低位控制寄存器	-
GPIO_CONHR	0x04	高位控制寄存器	-
GPIO_WODR	0x08	输出数据寄存器	0x0000_0000
GPIO_SODR	0x0C	输出置位寄存器	0x0000_0000
GPIO_CODR	0x10	输出清除寄存器	0x0000_0000
GPIO_ODSR	0x14	输出状态寄存器	0x0000_0000
GPIO_PSDR	0x18	管脚状态寄存器	0x0000_0000
RSVD	0x1C	Reserved	-
GPIO_PUDR	0x20	上拉/下拉配置寄存器	-
GPIO_DSCR	0x24	驱动强度配置寄存器	0x0000_0000
GPIO_OMCR	0x28	输出模式配置寄存器	0x0000_0000
GPIO_IECR	0x2C	外部中断使能寄存器	0x0000_0000
GPIO_IEER	0x30	外部中断使能设置寄存器	0x0000_0000
GPIO_IEDR	0x34	外部中断使能清除寄存器	0x0000_0000

注意：

- GPIO_IGRP 跟 GPIOA0 使用同一个 PCLK

- Base Address of GPIO_IGRP: 0x4004_4000

Register	Offset	Description	Reset Value
GPIO_IGRPL	0x00	外部中断组配置寄存器	0x0000_0000
GPIO_IGRPH	0x04	外部中断组配置寄存器	0x0000_0000

11.3.2 GPIO_CONLR (低位控制寄存器)

- Address = Base Address + 0x0000, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P7				P6				P5				P4				P3				P2				P1				P0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
P7	[31:28]	RW	IO 管脚 7 的功能模式	0x0
P6	[27:24]	RW	IO 管脚 6 的功能模式	0x0
P5	[23:20]	RW	IO 管脚 5 的功能模式	0x0
P4	[19:16]	RW	IO 管脚 4 的功能模式	0x0
P3	[15:12]	RW	IO 管脚 3 的功能模式	0x0
P2	[11:8]	RW	IO 管脚 2 的功能模式	0x0
P1	[7:4]	RW	IO 管脚 1 的功能模式	0x0
P0	[3:0]	RW	IO 管脚 0 的功能模式	0x0
0: GPD, GPIO 输入输出禁止模式 (默认模式) 1: GPI, GPIO 输入使能模式 2: GPO, GPIO 输出使能模式，输入禁止 3: GPO, GPIO 带输入监测的输出模式 4 ~15: AFx (x 从‘1’开始)，功能复用模式 (参见管脚配置)				

11.3.3 GPIO_CONHR (高位控制寄存器)

- Address = Base Address + 0x0004, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15				P14				P13				P12				P11				P10				P9				P8			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
P15	[31:28]	RW	IO 管脚 15 的功能模式	0x0
P14	[27:24]	RW	IO 管脚 14 的功能模式	0x0
P13	[23:20]	RW	IO 管脚 13 的功能模式	0x0
P12	[19:16]	RW	IO 管脚 12 的功能模式	0x0
P11	[15:12]	RW	IO 管脚 11 的功能模式	0x0
P10	[11:8]	RW	IO 管脚 10 的功能模式	0x0
P9	[7:4]	RW	IO 管脚 9 的功能模式	0x0
P8	[3:0]	RW	IO 管脚 8 的功能模式	0x0
0: GPD, GPIO 输入输出禁止模式 (默认模式) 1: GPI, GPIO 输入使能模式 2: GPO, GPIO 输出使能模式，输入禁止 3: GPO, GPIO 带输入监测的输出模式 4 ~15: AFx (x 从‘1’开始)，功能复用模式 (参见管脚配置)				

11.3.4 GPIO_WODR (输出数据寄存器)

- Address = Base Address + 0x0008, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description	Reset Value
Px	[x]	W	端口 x 输出数据控制位 0 = 对应管脚置‘0’，低电平。 1 = 对应管脚置‘1’，高电平。 该寄存器用途与寄存器 GPIO_SODR (输出置位寄存器) 和 GPIO_CODR (输出清除寄存器)一致。但是，不同的地方在于所有的输出数据都在同一时间被设置(1 和 0)。这个功能是与寄存器 GPIO_SODR 和 GPIO_CODR 不一致的。 只有当功能模式在寄存器 CONLR 或 CONHR 中被设置成 GPIO ，输出的数据才是有效的。	0

11.3.5 GPIO_SODR (输出置位寄存器)

- Address = Base Address + 0x000C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description	Reset Value
Px	[x]	W	端口 x 输出置 1 0 = 无效果 1 = 相应 GPIO 管脚的输出数据被置 1，高电平 只有当功能模式在寄存器 CONLR 或 CONHR 中被设置成 GPIO ，输出的数据才是有效的。	0

11.3.6 GPIO_CODR (输出清除寄存器)

- Address = Base Address + 0x0010, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description	Reset Value
Px	[x]	W	端口 x 的输出清零 0 = 无效果 1 = 相应 GPIO 管脚的输出数据被清零，变成低电平 只有当功能模式在寄存器 CONLR 或 CONHR 中被设置成 GPIO ，清除数据才是有效的。	0

11.3.7 GPIO_ODSR (输出状态寄存器)

- Address = Base Address + 0x0014, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description	Reset Value
Px	[x]	R	端口 x 输出状态 0 = 对应管脚置为‘0’，低电平。 1 = 对应管脚置为‘1’，高电平。	0

11.3.8 GPIO_PSDR (管脚状态寄存器)

- Address = Base Address + 0x0018, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description	Reset Value
Px	[x]	R	端口 x 输入状态 0 = 对应管脚为‘0’，低电平。 1 = 对应管脚为‘1’，高电平。	0

11.3.9 GPIO_PUDR (上拉/下拉配置寄存器)

- Address = Base Address + 0x0020, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15		P14		P13		P12		P11		P10		P9		P8		P7		P6		P5		P4		P3		P2		P1		P0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
P15	[31:30]	RW	上拉/下拉 IO 管脚 15	'b00
P14	[29:28]	RW	上拉/下拉 IO 管脚 14	'b00
P13	[27:26]	RW	上拉/下拉 IO 管脚 13	'b00
P12	[25:24]	RW	上拉/下拉 IO 管脚 12	'b00
P11	[23:22]	RW	上拉/下拉 IO 管脚 11	'b00
P10	[21:20]	RW	上拉/下拉 IO 管脚 10	'b00
P9	[19:18]	RW	上拉/下拉 IO 管脚 9	'b00
P8	[17:16]	RW	上拉/下拉 IO 管脚 8	'b00
P7	[15:14]	RW	上拉/下拉 IO 管脚 7	'b00
P6	[13:12]	RW	上拉/下拉 IO 管脚 6	'b00
P5	[11:10]	RW	上拉/下拉 IO 管脚 5	'b00
P4	[9:8]	RW	上拉/下拉 IO 管脚 4	'b00
P3	[7:6]	RW	上拉/下拉 IO 管脚 3	'b00
P2	[5:4]	RW	上拉/下拉 IO 管脚 2	'b00
P1	[3:2]	RW	上拉/下拉 IO 管脚 1	'b00
P0	[1:0]	RW	上拉/下拉 IO 管脚 0	'b00

'b00: 上拉禁止，下拉禁止
'b01: 上拉使能，下拉禁止
'b10: 上拉禁止，下拉使能
'b11: 上拉禁止，下拉禁止

即使在寄存器 CONLR 或 CONHR 中配置成 GPD，寄存器 PUDR 中的改动也仍然有效。

11.3.10 GPIO_DSCR (驱动强度配置寄存器)

- Address = Base Address + 0x0024, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15		P14		P13		P12		P11		P10		P9		P8		P7		P6		P5		P4		P3		P2		P1		P0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
P15	[31:30]	RW	IO 管脚 15 驱动强度配置	'b00
P14	[29:28]	RW	IO 管脚 14 驱动强度配置	'b00
P13	[27:26]	RW	IO 管脚 13 驱动强度配置	'b00
P12	[25:24]	RW	IO 管脚 12 驱动强度配置	'b00
P11	[23:22]	RW	IO 管脚 11 驱动强度配置	'b00
P10	[21:20]	RW	IO 管脚 10 驱动强度配置	'b00
P9	[19:18]	RW	IO 管脚 9 驱动强度配置	'b00
P8	[17:16]	RW	IO 管脚 8 驱动强度配置	'b00
P7	[15:14]	RW	IO 管脚 7 驱动强度配置	'b00
P6	[13:12]	RW	IO 管脚 6 驱动强度配置	'b00
P5	[11:10]	RW	IO 管脚 5 驱动强度配置	'b00
P4	[9:8]	RW	IO 管脚 4 驱动强度配置	'b00
P3	[7:6]	RW	IO 管脚 3 驱动强度配置	'b00
P2	[5:4]	RW	IO 管脚 2 驱动强度配置	'b00
P1	[3:2]	RW	IO 管脚 1 驱动强度配置	'b00
P0	[1:0]	RW	IO 管脚 0 驱动强度配置	'b00

每个 IO 通过两个 bit 分别设置驱动能力和驱动模式，在普通 IO 模式下，调整驱动能力可以设置不同的驱动电流。在恒流源模式下，调整驱动能力可以设置不同的恒流限值。

BIT1	驱动模式设置	BIT0	驱动强度设置
0	普通 IO 模式	0	弱驱动能力
1	恒流源模式	1	强驱动能力

NOTE: 恒流源驱动模式只有 LED 的 SEG 管脚才支持，其他管脚无此功能。LED 的 COM 管脚大电流驱动模式下可以支持 120mA 驱动。ECP0 管脚不支持驱动能力调整。

11.3.11 GPIO_OMCR (输出模式配置寄存器)

- Address = Base Address + 0x0028, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																ODP15	ODP14	ODP13	ODP12	ODP11	ODP10	ODP9	ODP8	ODP7	ODP6	ODP5	ODP4	ODP3	ODP2	ODP1	ODP0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
ODPx	[x]	RW	端口 x 开漏使能/禁止 0 = GPIO 管脚 x 不处于开漏输出模式 (推挽输出模式) 1 = GPIO 管脚 x 处于开漏输出模式	0

NOTE: 如果开漏使能，相应的管脚只能驱动“低“电平。当需要它驱动高电平时，管脚上需连接上拉电阻。

11.3.12 GPIO_IECR (外部中断使能寄存器)

- Address = Base Address + 0x002C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																IEN15	IEN14	IEN13	IEN12	IEN11	IEN10	IEN9	IEN8	IEN7	IEN6	IEN5	IEN4	IEN3	IEN2	IEN1	IEN0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
IENx	[x]	RW	端口 x 外部中断使能/禁止 0 = 外部中断禁止 1 = 外部中断使能	0

NOTE:

11.3.13 GPIO_IEER (外部中断使能设置寄存器)

- Address = Base Address + 0x0030, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																IEE15	IEE14	IEE13	IEE12	IEE11	IEE10	IEE9	IEE8	IEE7	IEE6	IEE5	IEE4	IEE3	IEE2	IEE1	IEE0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
IEEx	[x]	W	端口 x 外部中断使能设置寄存器 0：写‘0’时无效 1：写‘1’时设置该 GPIO 外部中断有效	0

NOTE:
该寄存器为只写寄存器

11.3.14 GPIO_IEDR (外部中断使能清除寄存器)

- Address = Base Address + 0x0034, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																IED15	IED14	IED13	IED12	IED11	IED10	IED9	IED8	IED7	IED6	IED5	IED4	IED3	IED2	IED1	IED0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
IEDx	[x]	W	端口 x 外部中断使能清除寄存器 0：写‘0’时无效 1：写‘1’时设置该 GPIO 外部中断无效	0

NOTE:
该寄存器为只写寄存器

11.3.15 GPIO_IGRPL (外部中断组配置寄存器)

- Address = Base Address + 0x0000, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	GRP7			RSVD	GRP6			RSVD	GRP5			RSVD	GRP4			RSVD	GRP3			RSVD	GRP2			RSVD	GRP1			RSVD	GRP0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	RW	RW	RW	R	RW	RW	RW	R	RW	RW	RW	R	RW	RW	RW	R	RW	RW	RW	R	RW	RW	RW	R	RW	RW	RW	R	RW	RW	RW

Name	Bit	Type	Description	Reset Value
GRP7	[30:28]	RW	选择外部中断组 7	0
GRP6	[26:24]	RW	选择外部中断组 6	0
GRP5	[22:20]	RW	选择外部中断组 5	0
GRP4	[18:16]	RW	选择外部中断组 4	0
GRP3	[14:12]	RW	选择外部中断组 3	0
GRP2	[10:8]	RW	选择外部中断组 2	0
GRP1	[7:4]	RW	选择外部中断组 1	0
GRP0	[3:0]	RW	选择外部中断组 0	0

0000: GPIOA0.x 被选中
0001: GPIOA1.x 被选中
0010: GPIOB0.x 被选中
0011: GPIOC0.x 被选中
0100: GPIOD0.x 被选中
Other: GPIOA0.x 被选中

'x' 表示组数

11.3.16 GPIO_IGRPH (外部中断组配置寄存器)

- Address = Base Address + 0x0004, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	GRP15			RSVD	GRP14			RSVD	GRP13			RSVD	GRP12			RSVD	GRP11			RSVD	GRP10			RSVD	GRP9			RSVD	GRP8		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	RW	RW	RW	R	RW	RW	RW	R	RW	RW	RW	R	RW	RW	RW	R	RW	RW	RW	R	RW	RW	RW	R	RW	RW	RW	R	RW	RW	RW

Name	Bit	Type	Description	Reset Value
GRP15	[30:28]	RW	选择外部中断组 15	0
GRP14	[26:24]	RW	选择外部中断组 14	0
GRP13	[22:20]	RW	选择外部中断组 13	0
GRP12	[18:16]	RW	选择外部中断组 12	0
GRP11	[14:12]	RW	选择外部中断组 11	0
GRP10	[10:8]	RW	选择外部中断组 10	0
GRP9	[6:4]	RW	选择外部中断组 9	0
GRP8	[2:0]	RW	选择外部中断组 8	0
0000: GPIOA0.x 被选中 0001: GPIOA1.x 被选中 0010: GPIOB0.x 被选中 0011: GPIOC0.x 被选中 0100: GPIOD0.x 被选中 Other: GPIOA0.x 被选中 ‘x’ 表示组数				

12 通用定时器 (GPT/TC0)

12.1 概述

通用定时器 0 (GPT/TC0)是一个 16 位的定时/计数模块，包含功耗控制器和可选的并行 IO 控制器。定时器 0 有 3 个输入输出通道，2 种工作模式(捕捉模式和波形发生器模式)，用来实现各种功能，例如频率测量，事件计数，时长测量，脉冲发生，产生延时，脉冲宽度调制(PWM)等。

12.1.1 主要特性

TC0 有三个通道。每个通道，都有如下功能：

- 1 个 16 位的可复位计数器: GPT_CV
- 3 个 16 位的比较值寄存器:

  1 个简单比较寄存器: GPT_RC

  2 个捕捉/比较寄存器: GPT_RA 和 GPT_RB
- 1 个选择器，可以选择 6 个时钟源，其中 5 个内部，1 个外部。

  支持将 2 个时钟源组合在一起，产生一个时钟脉冲群。

  外部时钟可以当作外部的触发源。
- 可编程的中断选择，支持下列中断:
  - 计数器溢出: **COVFS**
  - 寄存器 A 被载入数据或寄存器 B 被载入数据: **LDRAS or LDRBS**
  - 比较寄存器 A，B 或 C 与计数器相等: **CPAS, CPBS, or CPS**
  - 外部沿检测: **ETRGS**
  - 寄存器覆盖: **LOVRS**
- 1 个软件触发源，可以用作软件复位。
- 1 个软件复位可以复位所有通道以及其相关寄存器(PMC 寄存器除外)。3 个 I/O 管脚可以用作各种不同的功能(根据不同工作模式)。
  - 在捕捉模式，TIOA 是一个事件输入，用来载入寄存器 A，寄存器 B，或者作为输入触发源。在波形发生器模式，TIOA 用来输出波形。
  - TIOB, 在捕捉模式，是一个输入触发源，而在波形发生器模式，即可以当做波形输出(双波形输出模式)，也可以当做输入触发源(单波形输出模式)。
  - TCLK, 在捕捉模式和波形发生器模式，都是外部时钟输入。

12.1.2 管脚描述

下表列出了不同模式下的管脚定义。

Table 12-1 不同模式下的管脚描述

管脚名称	捕捉模式	波形发生器： 单波形输出模式	波形发生器： 双波形输出模式
TIOA[2:0]	捕捉输入或者输入触发	输出波形	输出波形
TIOB[2:0]	输入触发	输入触发 (GPT_MR.EEVT = 00)	输出波形 (GPT_MR.EEVT = 01, 10, 11)
TCLK[2:0]	外部时钟输入	外部时钟输入	外部时钟输入
ETR	外部同步(上升沿)	外部同步(上升沿)	外部同步(上升沿)

12.2 功能描述

12.2.1 模块框图

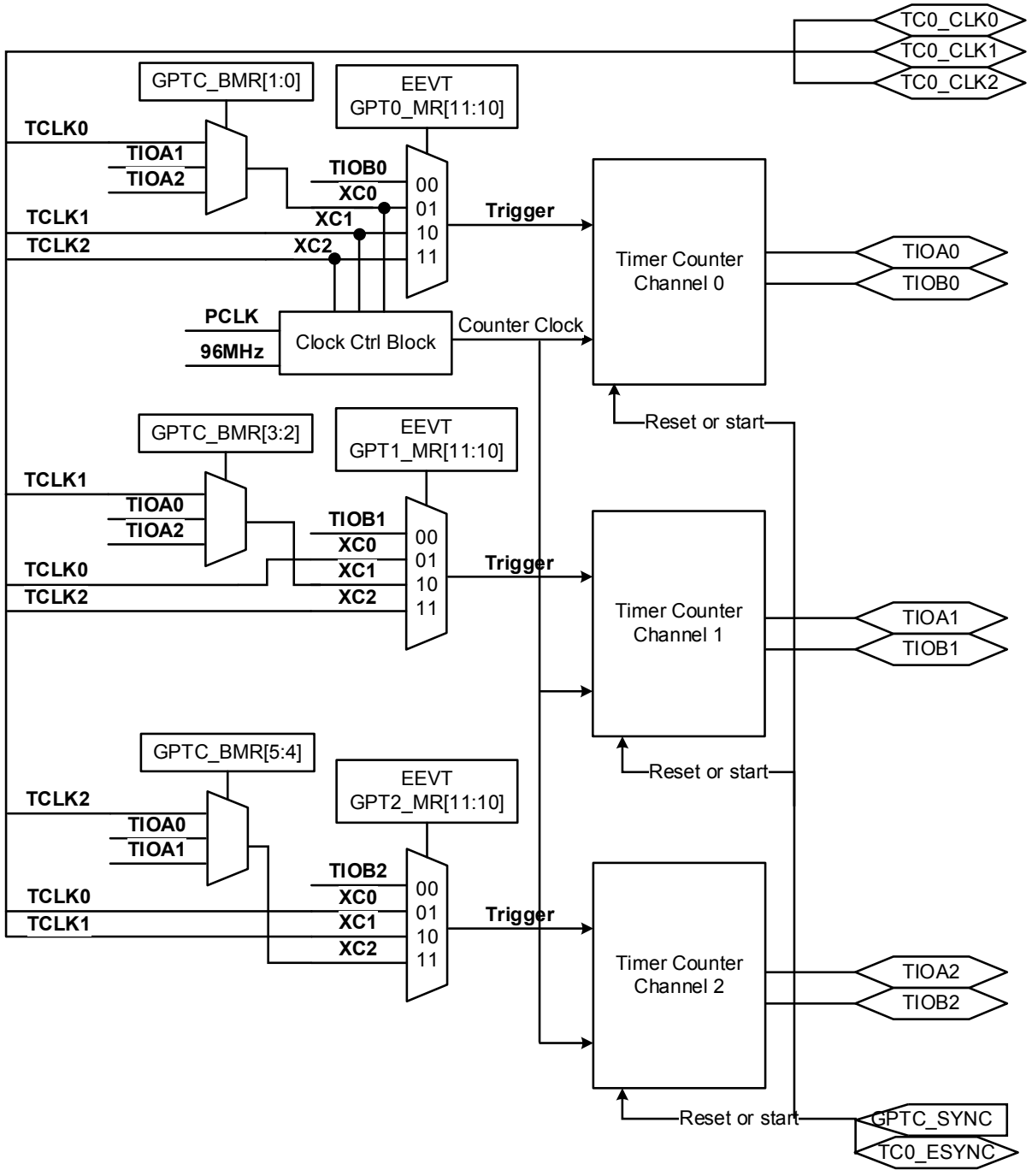


Figure 12-1 通用定时器模块框图

注意：上图中作为 **Trigger** 的 **TIOAx/TIOBx** 为定时器模块的输出，用来级联，并非外部管脚输入。

12.3 基本功能描述

12.3.1 时钟源

12.3.1.1 概述

通用定时器 TC0 可以配置使用各种不同的时钟。模式寄存器 GPT_MR 的 CLKS[2:0]这 3 位用来选择计数器的时钟是 5 个内部时钟源(C_CLK/x)还是外部时钟(TCLK)。 同时，C_CLK 这个计数器时钟又可以选择为 TC0 的 PCLK 或者一个系统为 TC0 专门提供的 96MHz 高速时钟 HFCLK。

计数器的时钟源可以是以下任意一个：

- 外部事件输入：XC0, XC1, 和 XC2
- 5 个内部时钟中的任意一个： C_CLK/2, C_CLK/8, C_CLK/32, C_CLK/128, 和 C_CLK/1024
- 群脉冲时钟 (参见“群脉冲时钟”的说明)

当选择内部时钟时，最大计数时长由内部时钟频率 (参考下图) 和分频数决定。

最大计数时长(秒) = 2^{16}/CLK (CLK 单位为 Hz). 计数精度= 1/CLK.

该定时器支持 96MHz 的高速时钟工作，可以实现高精度的 PWM 波形输出。如需使用高速时钟，首先需要在 SYSCON 模块中先将高速时钟使能(SYSCON_CLCR 寄存器中的 HFOSCEN 位)，并且等待该时钟稳定(SYSCON_CLCR 寄存器中的 HFO_ST 位)，然后再将 GPT_BMR 中的 HSPD_EN 位置 1。

12.3.1.2 时钟模块框图

下图为时钟选择模块的示意图：

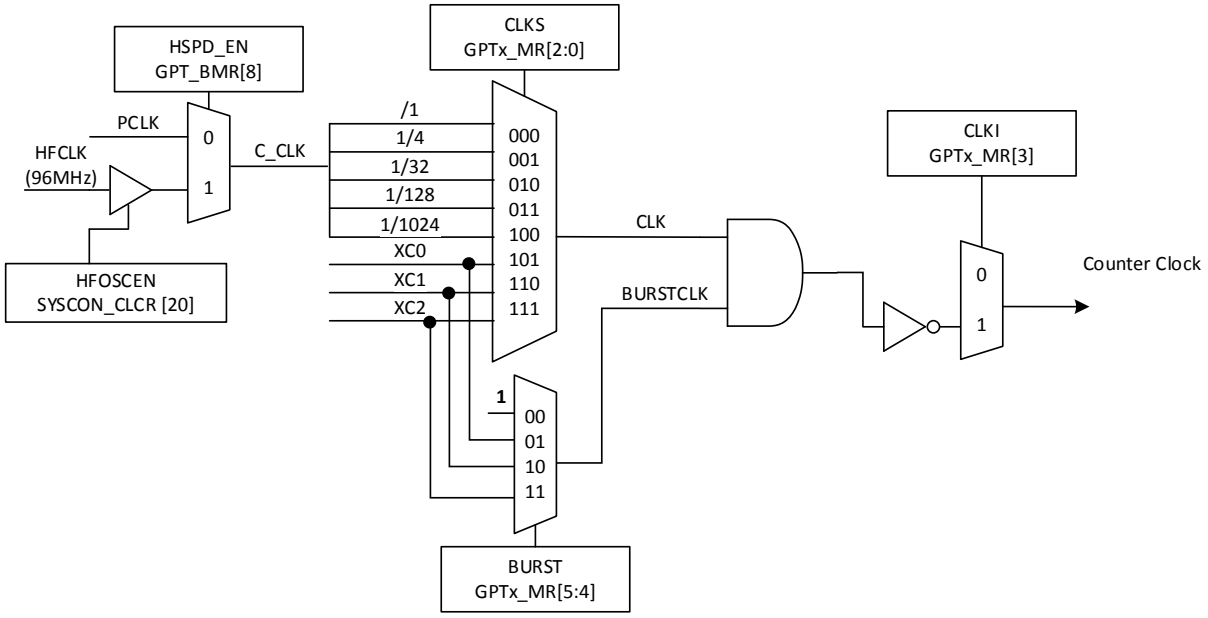


Figure 12-2 时钟选择模块框图

12.3.1.3 外部时钟

当选择外部时钟的时候，16 位计数器可以用作一个 16 位的事件计数器。外部的电平变化(上升还是下降由模式寄存器的第 3 位 CLKI 决定)会让计数器加 1.

如果使用外部时钟，必须保证每个时钟脉冲的宽度都大于 PCLK 的周期。

12.3.1.4 群脉冲时钟

如果 BURST (模式寄存器的[5:4]位) 选择了外部时钟，那么这个外部时钟会跟内部 CLK 进行与操作。

所以，这个通用定时器的时钟只有当 CLKBURST 为高的时候才发挥作用，如下图所示：

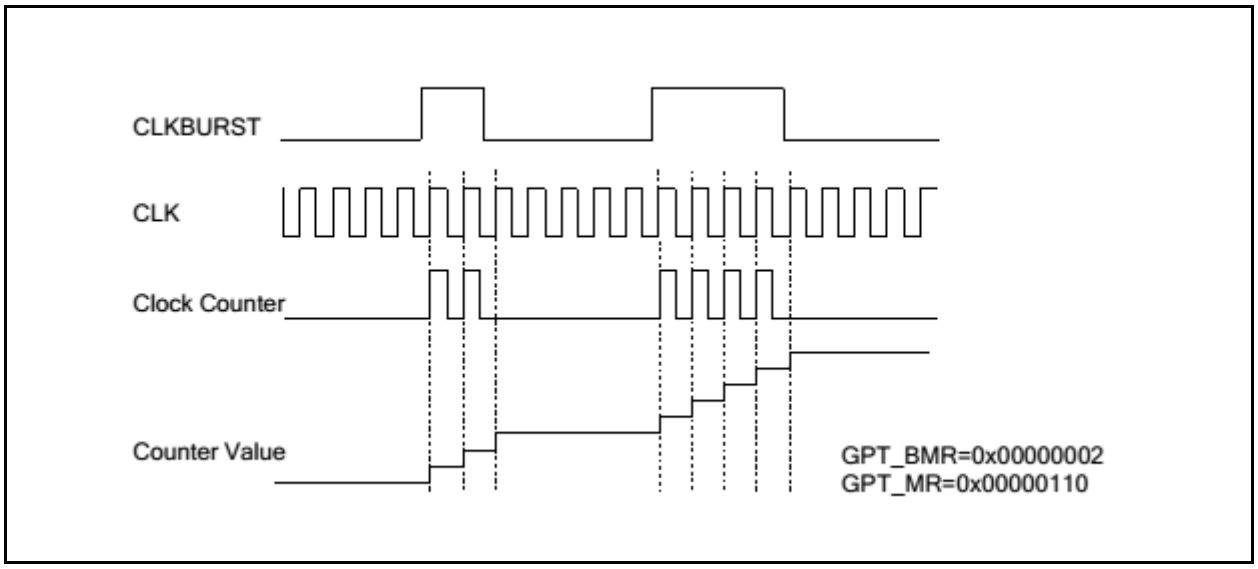


Figure 12-3 群脉冲时钟的时序图

12.3.1.5 16位计数器

16 位计数器的计数支持 8 个时钟源：5 个内部和 3 个外部。

程序可以通过寄存器 GPT_CV 实时读取计数器的计数值。

计数器发生复位时，计数值被重置为 0x0000，如果时钟使能，那么计数器会从 0x0000 开始计数。时钟的使能或者禁止由控制寄存器 GPT_CR 的第 2 位 CLKDIS 和第 1 位 CLKEN 控制。

当计数达到最大计数值(0xFFFF)时，计数器回滚到 0x0000，并且将溢出标志位(COVFS, 状态寄存器 GPT_SR 的第 0 位) 置 1，同时可以产生一个中断(通过中断使能寄存器 GPT_IER 的第 0 位 COVFS 使能，通过中断禁止寄存器 GPT_IDR 的第 0 位 COVFS 禁止)，然后继续开始计数。

计数器的复位：

当计数器在工作时，计数值可以通过下列操作重置为 0x0000：

- 软件触发 (GPT_CR 的 SWTRG)
- 外部触发
- 比较值 C 的匹配
- 模块控制寄存器 GPT_BCR 的第 1 位，同步位 TCSYNC

每当重置发生时，计数值会在下一个有效的时钟沿变成 0x0000，如下图所示：

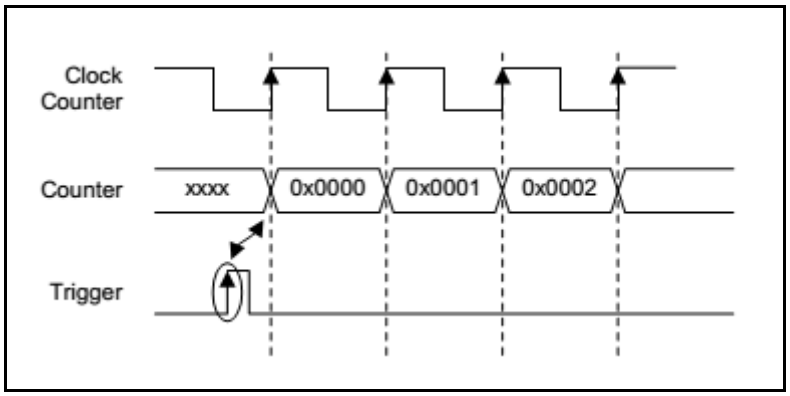


Figure 12-4 计数器重置框图

12.3.1.6 16位寄存器

该通用定时器的每个通道都有 3 个用于捕捉/比较的 16 位寄存器。

模式的选择可以决定该寄存器是用于捕捉寄存器还是比较寄存器。

在捕捉模式，寄存器 A 和 B 为捕捉寄存器，并且会在 TIOA 的变化沿捕捉计数值。

在波形发生模式，寄存器 A 和 B 为比较寄存器。

寄存器 C 则永远是一个比较寄存器。

当计数器的值跟比较寄存器的预置值相同时，比较寄存器可以产生一个计数器重置(RC)，或者产生需要的波形(RA, RB 和 RC)。

在实际应用中，需要比较的值必须通过下面的公式计算：

比较值 = (t x CLK) - 1

- t = 希望产生的时长 (以秒为单位)
- CLK = 计数器时钟 (以赫兹为单位)

例如：

计数器 PCLK = 30MHz 时，让计数器在 0.1 秒后发生一个计数值等于比较寄存器的事件。

比较值 = 0.1 x (PCLK/1024) - 1 = 0.1 x (30000000/1024) - 1= 2928.69

取整值 2929 (0x0B71) 产生约 0.01% 的偏差。

12.3.1.7 外部沿检测

该通用定制器包含有很多的外部沿检测选项。

工作模式决定了它们的功能：

- 捕捉模式：
  - TIOA 作为捕捉触发和外部触发
  - TIOB 作为外部触发
- 波形发生模式：双波形模式
  - XC0, XC1, XC2 作为外部触发
- 波形发生模式：单波形模式
  - TIOB 作为外部触发

每个外部沿检测，都可以选择上升沿，下降沿或者上升下降沿。

注意每次复位都是发生在下一个有效的时钟沿。

如果使用了外部触发，必须保证该触发源的输入脉冲宽度大于 PCLK 周期。

12.3.1.8 中断

通用定时器的每个通道都有 8 个中断。它们可以通过 GPT_IER 和 GPT_IDR 寄存器使能或者禁用。

工作模式决定了哪些中断可用，如下表所示：

Table 12-2 可用中断

中断名称	捕捉模式	波形发生模式
计数值溢出中断 COVFS	O	O
计数值载入溢出中断 LOVRS	O	X
比较寄存器 A 匹配中断 CPAS	X	O
比较寄存器 B 匹配中断 CPBS	X	O
比较寄存器 C 匹配中断 CPCS	O	O
载入捕捉寄存器 A 中断 LDRAS	O	X
载入捕捉寄存器 B 中断 LDRBS	O	X
外部触发中断 ETRGS	O	O

12.3.2 捕捉模式

12.3.2.1 概述

当 WAVE 位(模式寄存器的第 15 位)是 0 的时候，定时器工作为捕捉模式(可用于波形检测)。

捕捉模式为硬件复位后默认的工作模式。该模式强制将 TIOA 和 TIOB 当作输入管脚。

捕捉模式可以用来检测 2 个事件发生的间隔时间。

事件可以是外部在 TIOA 或者 TIOB 上的输入信号，也可以是内部的事件(软件触发或者预设比较寄存器的匹配)。

TIOA 上的外部事件(上升或者下降沿)可以让计数值载入寄存器 A，让计数值载入寄存器 B，或者触发功能(重置并且重新开始计数)。

寄存器 C 可以设置一个预先定义好的比较值(16 位)。

当捕捉寄存器 B 被载入时，可以在此时禁止计数器的时钟或者停止计数。

用户可以随意选择内部时钟源 (PCLK/2, PCLK/8, PCLK/32, PCLK/128 或 PCLK/1024)，或者外部时钟源 (XC0, XC1, XC2)。

群脉冲模式可以用来产生特殊的时钟群，详细信息请参考“时钟源”章节。

定时器在以下情况会产生中断：

- 检测到外部触发
- RA 载入
- RB 载入
- 计数值溢出 (当计数值从 0xFFFF 计到 0x0000)
- 载入溢出 (当 RA 或者 RB 还没被读取时又被载入值)
- RC 匹配 (计数器计数到寄存器 C 设定的值)

最后，同步寄存器可以用来产生一个软件触发，让所有通道的计数器重置并且同时开始计数。

下面有一些例子，演示了捕捉模式的用途。

12.3.2.2 检测TIOA脉冲宽度，以及TIOB和TIOA的相位

TIOB 上升沿复位并且开始计数。

TIOA 上升沿载入 RA，下降沿载入 RB。

当 RB 载入时，计数停止。

外部触发重新开始一个捕捉周期。

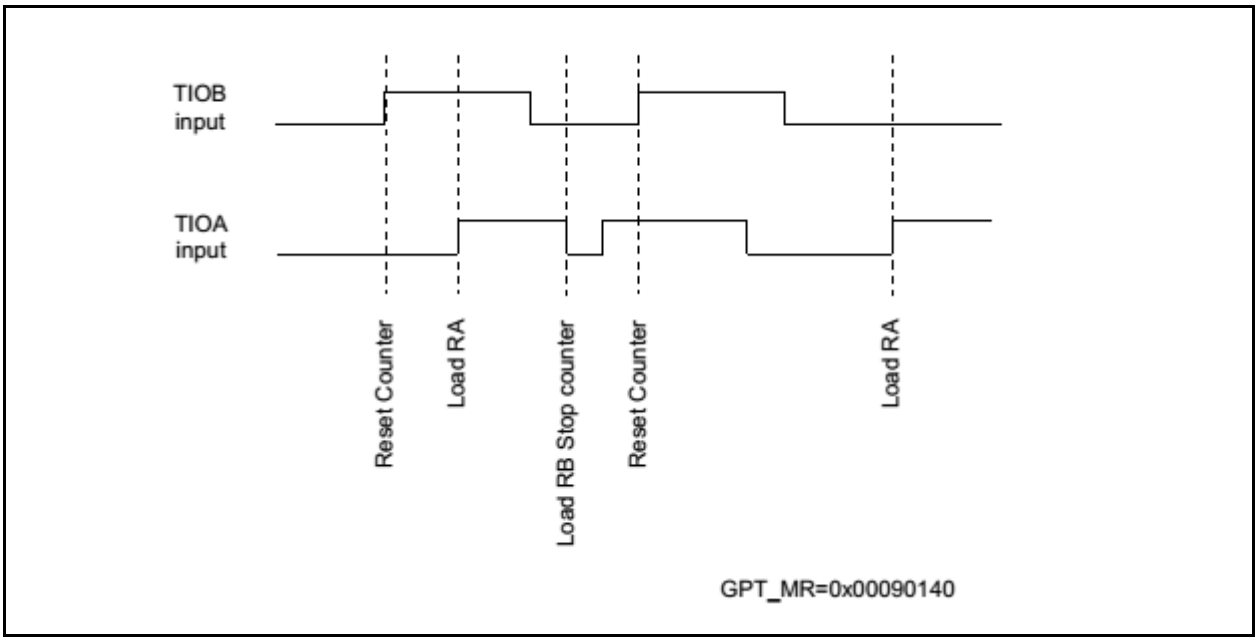


Figure 12-5 TIOA 脉冲宽度

RA 的值为 TIOB 和 TIOA 的相位差。

(RB-RA) 为 TIOA 脉冲的宽度

注意计数器重置后，TIOA的下降沿不会让计数值载入RB。RA总是会被先载入，也就是说在RA没有被载入的前提下，RB是不会被载入计数值的。

12.3.2.3 检测TIOA上两个连续上升沿的间隔时间

TIOB 上升沿复位并且开始计数。

开始计数后的第一个 TIOA 上升沿载入 RA，第二个 TIOA 上升沿载入 RB。

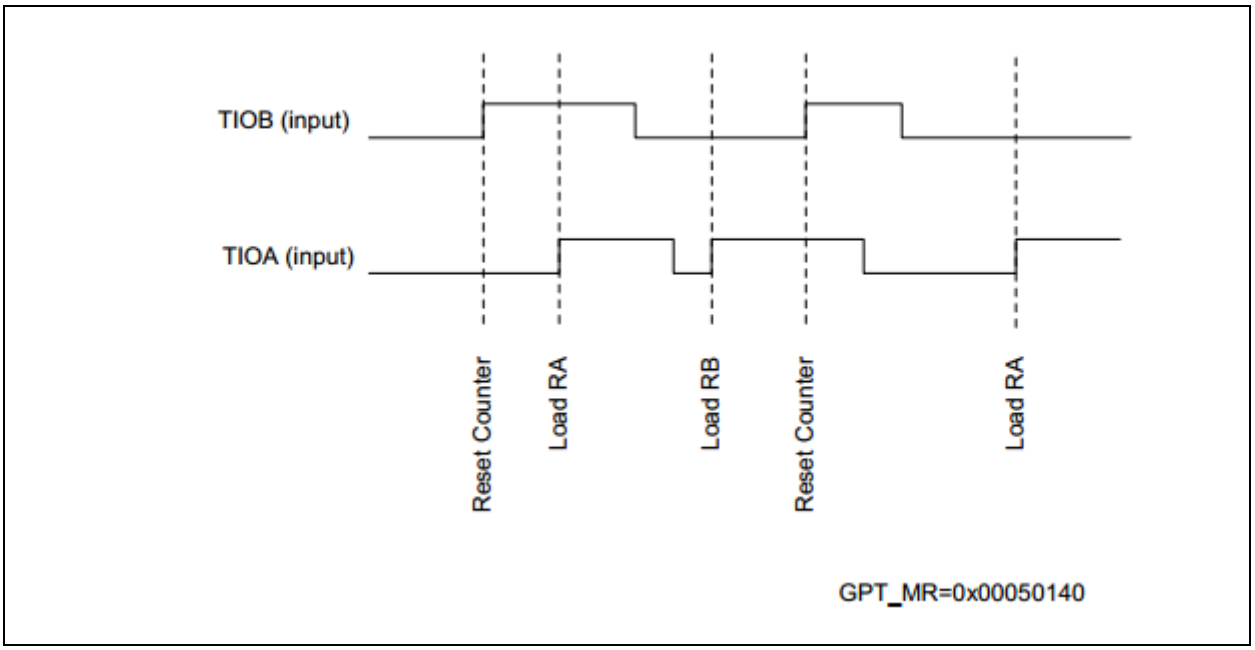


Figure 12-6 TIOA 上升沿

RA 的值为 TIOB 和 TIOA 的相位差。

(RB-RA) 为TIOA脉冲的周期

12.3.2.4 检测TIOA脉冲宽度或者周期 (TIOB未使用)

TIOA 下降沿复位并且开始计数，如果 RA 已经被载入，那么 TIOA 下降沿也载入 RB。

TIOA 上升沿载入 RA。

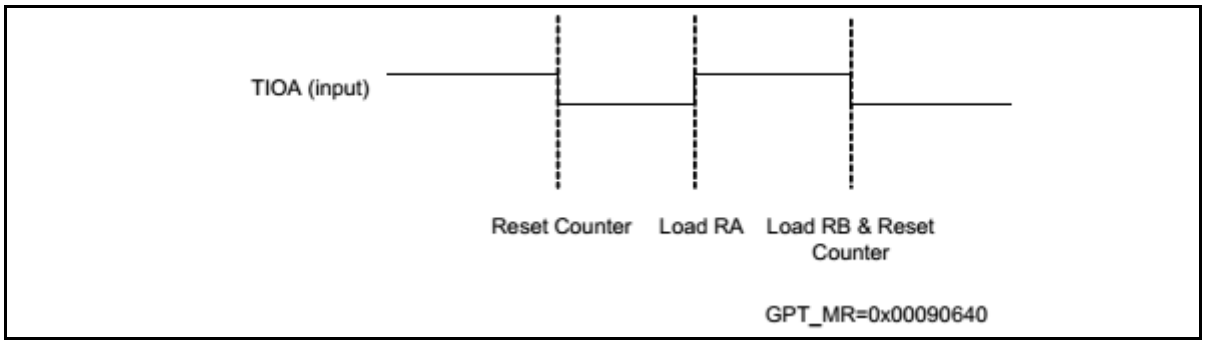


Figure 12-7 TIOA 脉冲宽度或者周期

RA 的值为 TIOA 脉冲宽度(低电平) 时长。
RB 的值为 TIOA 的周期时长。

12.3.2.5 外部时钟TCLK的事件计数

每个 TCLK 上升沿，计数值加 1。

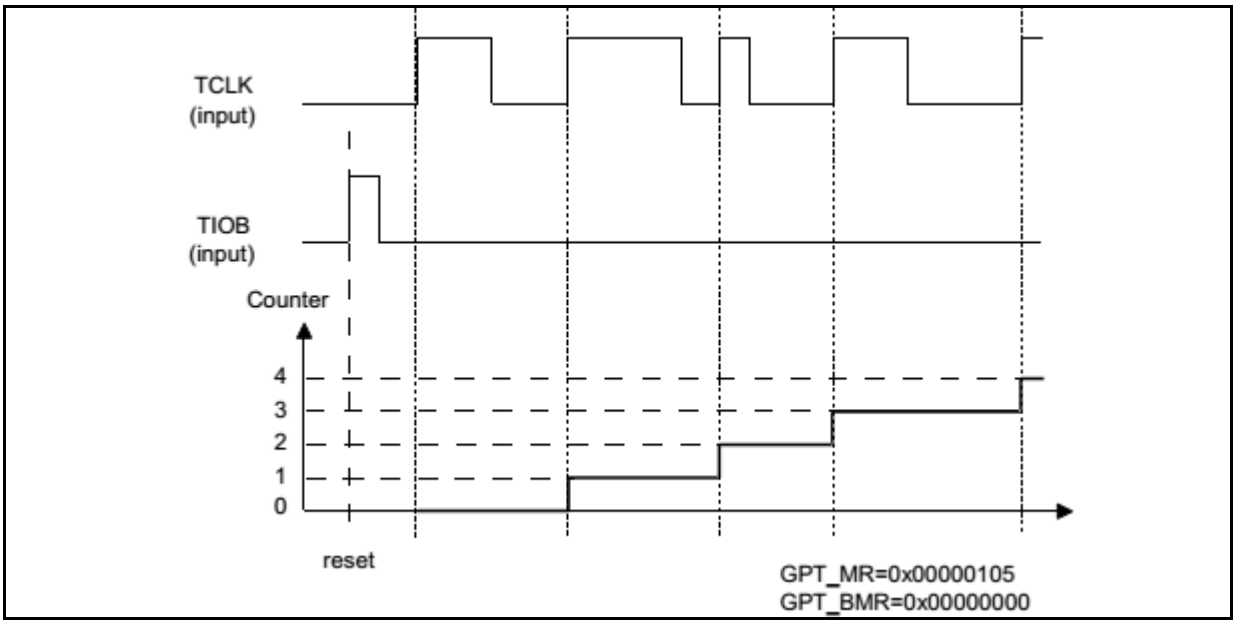


Figure 12-8 TCLK的计数

12.3.3 计数值为检测到的TCLK上升沿的个数。

波形发生模式

12.3.3.1 概述

当 WAVE 位(模式寄存器的第 15 位)是 1 的时候，定时器工作为波形发生模式。该模式强制将 TIOA 当作输出管脚。TIOB 则可以当做输出(双波形输出模式)或者输入(单波形输出模式)。

波形发生模式可以用来产生对称的或者可变的周期波形。

TIOA 管脚的输出波形(变 1，变 0，或者翻转) 由 4 个事件控制：

- 软件触发
- 外部事件的变化沿 (上升，下降或者上升下降)
- 计数器和比较寄存器 A 的值匹配
- 计数器和比较寄存器 C 的值匹配

作为输出，TIOB 管脚的输出波形(变 1，变 0，或者翻转) 由 4 个事件控制：

- 软件触发
- 外部事件的变化沿 (上升，下降或者上升下降)
- 计数器和比较寄存器 B 的值匹配
- 计数器和比较寄存器 C 的值匹配

当 TIOB 用作外部触发源时，比较寄存器 B 不起作用。

当比较寄存器 C 的值匹配时，可以触发 3 个事件：

- 计数器重置并且在下个时钟沿开始计数
- 计数器停止计数
- 计数器停止计数并且禁用计数器的时钟。

如果寄存器 C 匹配重新开始计数，那么可以产生一个连续的波形，周期为寄存器 C 的值+1。如果不重新开始计数，那么也可以产生一个连续的波形，只是周期为最大计数值(0xFFFF)能产生的周期。
用户可以随意选择内部时钟源 (PCLK/2, PCLK/8, PCLK/32, PCLK/128 或 PCLK/1024)，或者外部时钟源 (TCLK)。支持群脉冲模式，可以用来产生特殊的时钟群，详细信息请参考“时钟源”章节。

下列事件会产生中断：

- 检测到外部触发
- 计数值溢出 (当计数值从 0xFFFF 变为 0x0000)
- RA 匹配 (计数值跟比较寄存器 A 的值相等)
- RB 匹配 (计数值跟比较寄存器 B 的值相等)
- RC 匹配 (计数值跟比较寄存器 C 的值相等)

最后，同步寄存器可以用来产生一个软件触发，让所有通道的计数器重置并且同时开始计数。

下面有一些例子，演示了波形发生模式的用途。

12.3.3.2 产生双脉冲宽度调制 (PWM) (双波形输出模式)

TIOA由RA和RC的值决定翻转，TIOB由RB和RC决定。

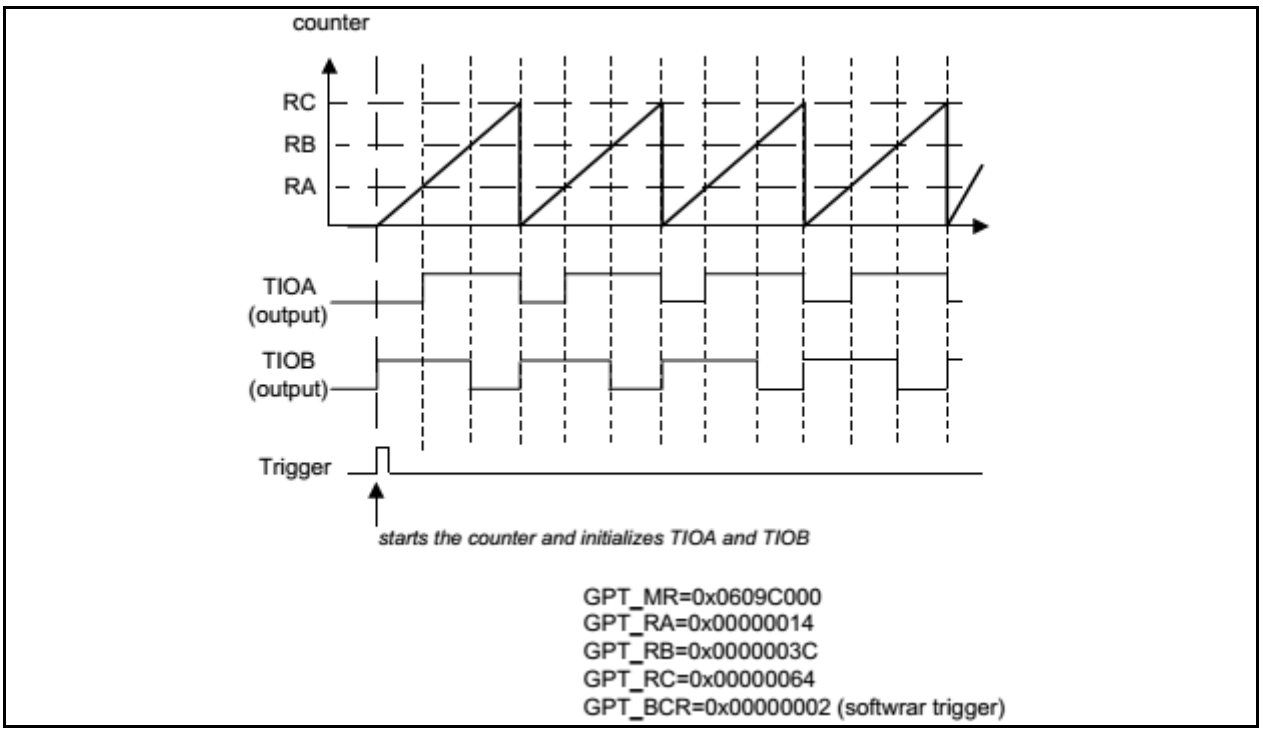


Figure 12-9 双脉冲宽度调制

RC 的值决定两个信号的频率，RA 决定 TIOA 的占空比，RB 决定 TIOB 的占空比。

12.3.3.3 产生2个波形相同的但相位不同的方波信号 (双波形输出模式)

每次计数到 RA 值的时候 TIOA 翻转，计数到 RC 值的时候 TIOB 翻转。某个触发(外部或者软件)启动计数并且初始化 TIOA 和 TIOB。

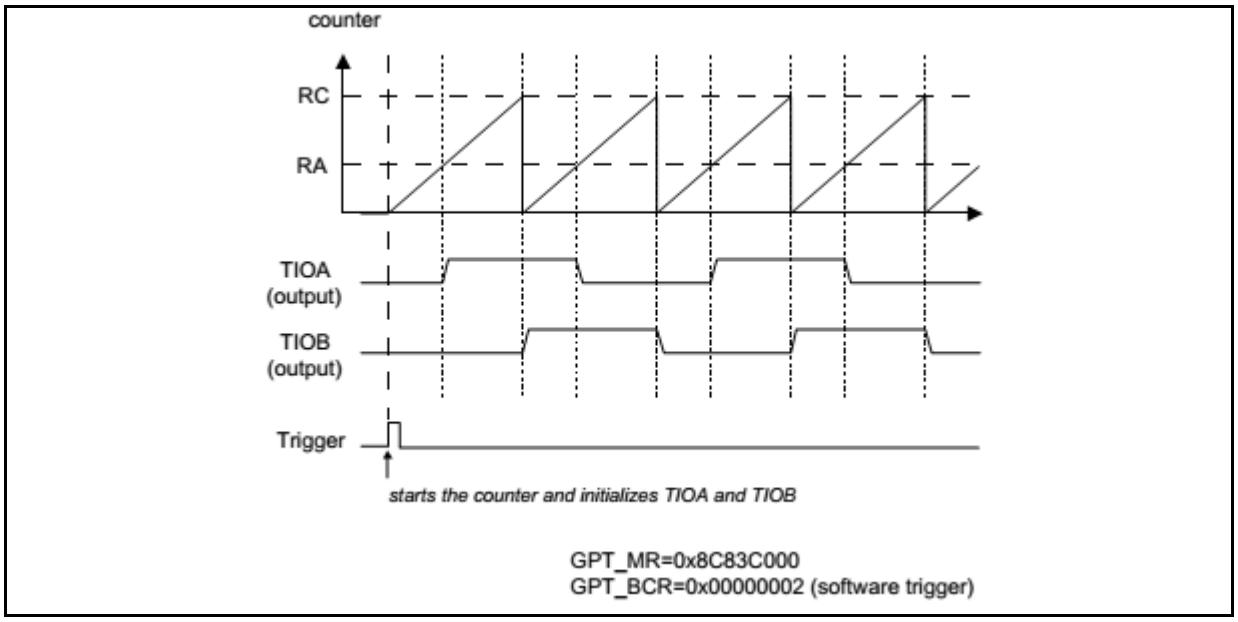


Figure 12-10 2 Square Signals

RC决定两个信号的频率，RA则决定两个信号的相位延时。

12.3.3.4 产生特定脉冲 (双输出波形模式)

只有在某个触发源触发后，才在 TIOA 和 TIOB 上产生脉冲。

TIOA 脉冲的宽度由 RA 的值决定，TIOB 脉冲的宽度由 RB 的值决定。

只有每当新的触发出现后，才会产生新的输出脉冲。

当触发来的时候，计数器被复位并且在下一个有效的时钟上升沿开始计数。

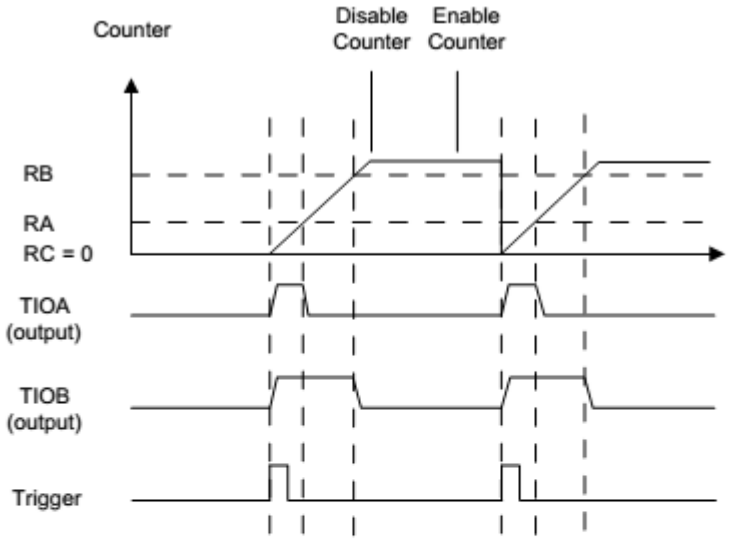
如果想在计数器重置的时刻马上产生脉冲，必须设置 RC=0,另外 MR 寄存器的

CPCTRG 需设置为 0，即 RC 匹配不作为触发。

所以，产生脉冲的原因是 RC=0 的匹配事件，而不是触发事件。

在这种情况下，用户必须设置比较寄存器 RB 匹配时禁止计数时钟，从而停止计数。

想要接收新的触发事件，用户必须重新将计数器的时钟使能。




```
GPT_ECR=0x00000002 (enable the channel clock)
GPT_IER=0x00000008 (enable interrupt CPBS)
GPT_MR=0x06068400
GPT_RA=0x00000002 (waveform mode)
GPT_RB=0x00000005 (waveform mode)
GPT_RC=0x00000000 (waveform mode)
GPT_CR=0x0000000A (reset and enable counter)

(wait for interrupt)

GPT_CR=0x00000004 (disable counter)
GPT_CSR=0x00000008 (clear interrupt)
GPT_CR=0x0000000A (reset and enable counter)
```

Figure 12-11 产生脉冲

12.3.3.5 TIOB输入管脚触发 (单波形输出模式)

上面的一些例子中，TIOB 都是作为输出，但是 TIOB 也是可以作为触发输入的，并且可以用来产生 TIOA 的输出信号。

下面的应用跟第一个例子一样 (双脉冲宽度调制)，但 TIOB 不是作为波形输出，而是作为一个触发输入。

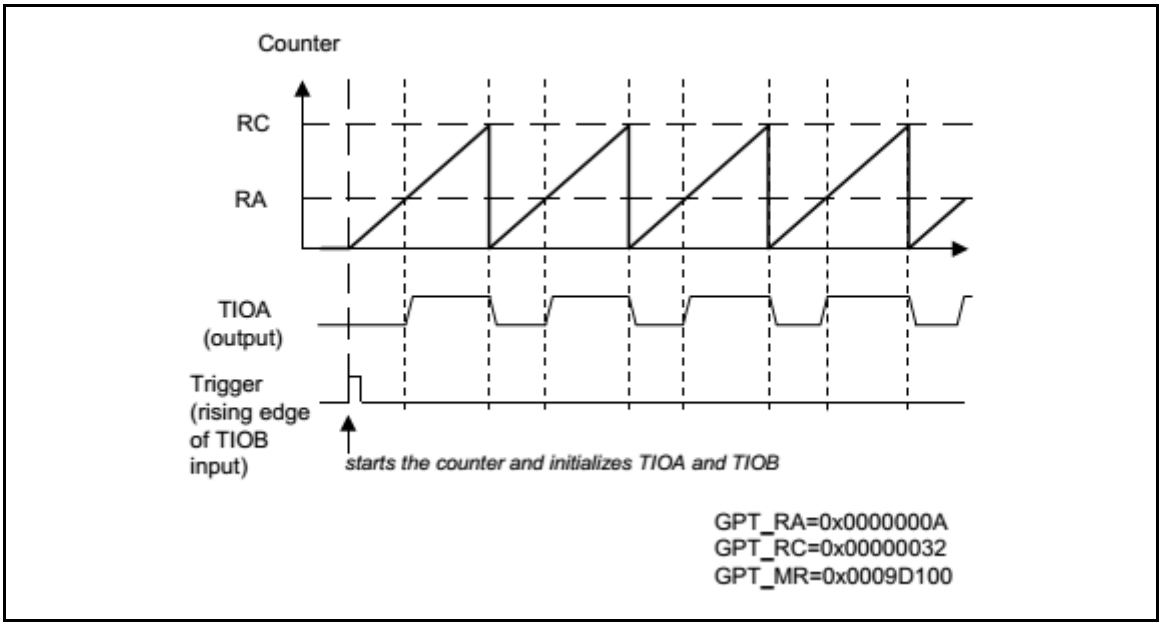


Figure 12-12 TIOB作为触发的单波形输出

12.3.3.6 外部时钟 (TCLK0) 的事件计数

在每个 TCLK 的上升沿，计数值加 1。(参考 GPT_MR 寄存器的 CLKI，可以设置上升或者下降沿)

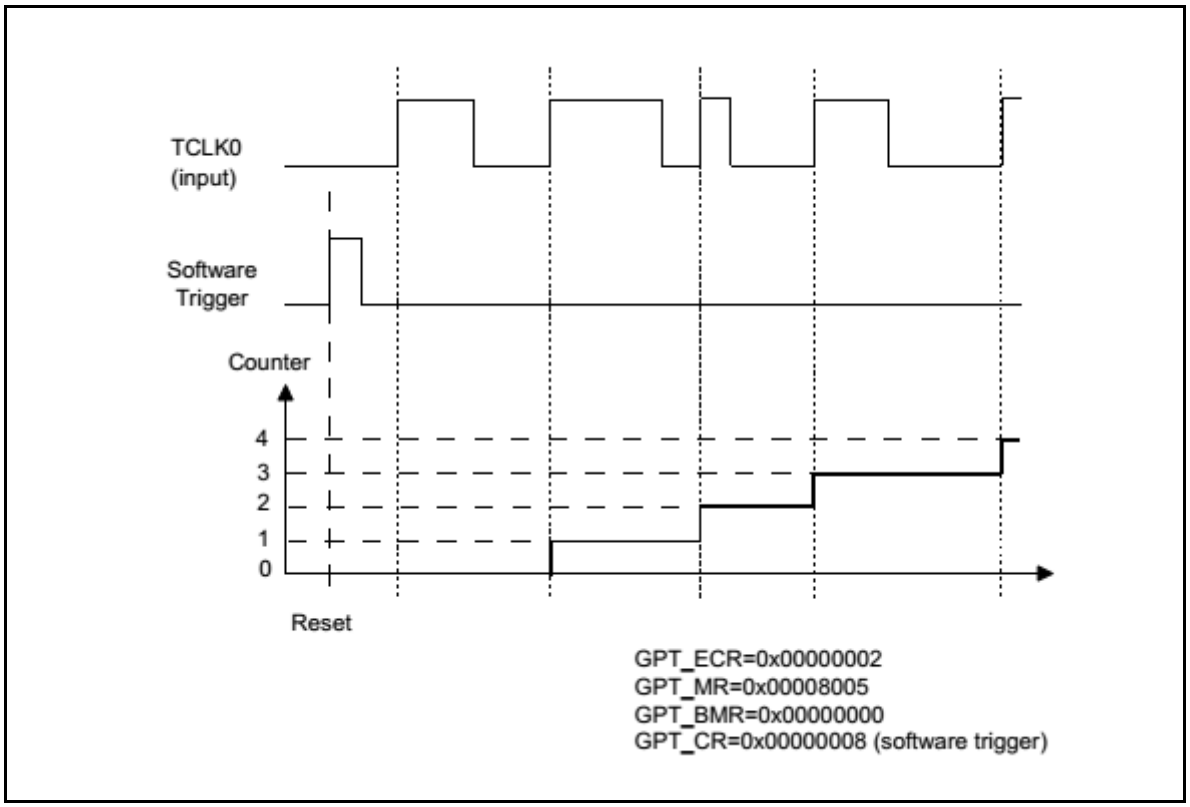


Figure 12-13 事件计数

12.4 整个定时器控制模块的级联和编程

12.4.1 概述

这个控制模块用来控制通用定时器 TC0 所有的 3 个通道。

它有 2 个主要功能：

- 能让 3 个定时器通道同步。它能产生一个软件触发信号，让 3 个通道同时开始工作。
- 能让 2 个或者 3 个通道组成菊花链，也就说可以提高计数的能力。

下图为控制模块框图：

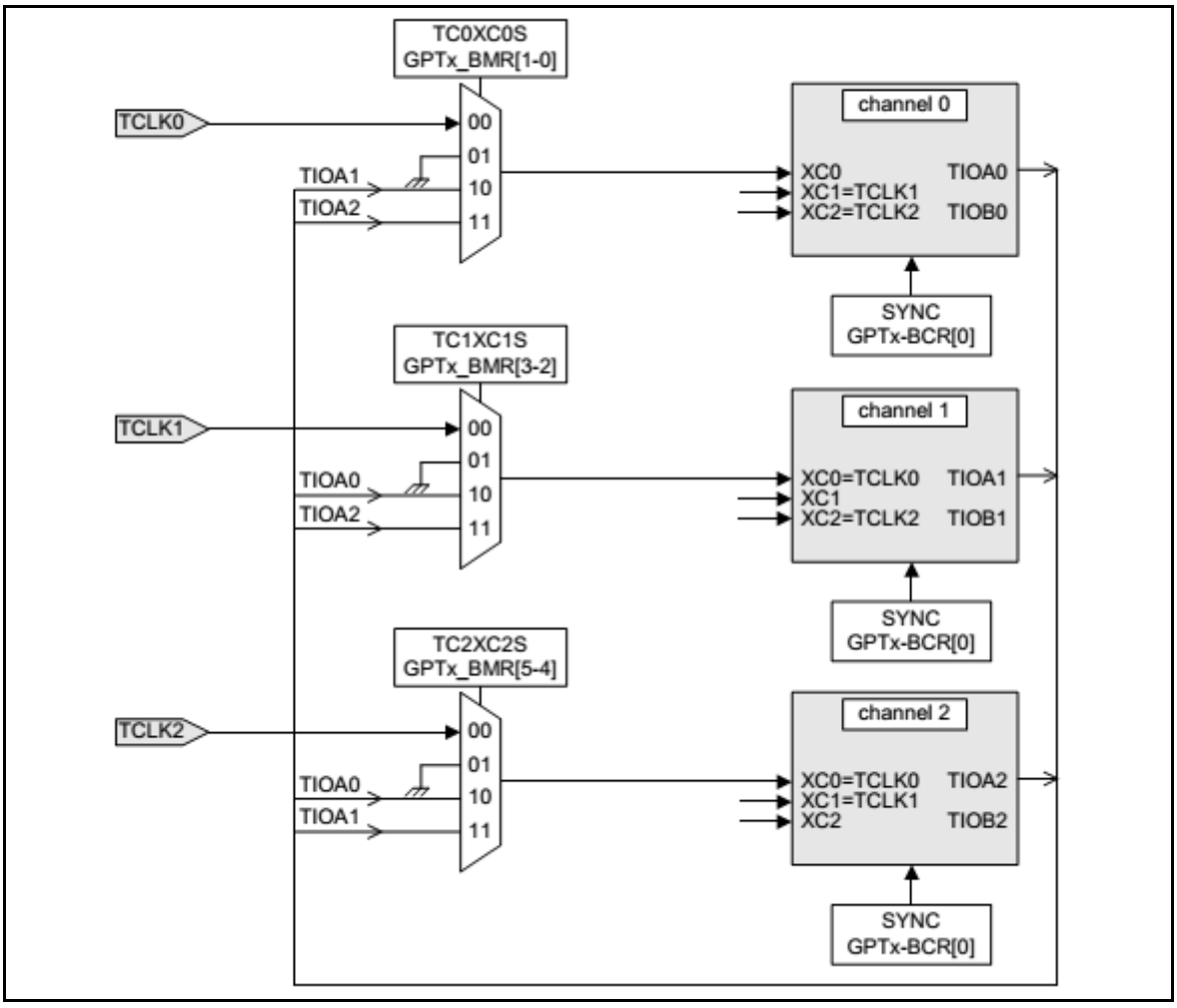


Figure 12-14 通用定时器控制模块编程

12.4.1 外部同步管脚

ETR管脚为外部同步输入管脚，只有上升沿有效。当ETR管脚收到一个上升沿信号时，计数器会被重置并且在下一个有效时钟开始计数，也就是说ETR的上升沿与BCR寄存器的TCSYNC功能一致。

12.4.1 使用通道0，给通道1产生一个外部时钟

使用 GPT0_RA 和 GPT0_RC，通道 0 产生一个脉冲宽度调制(PWM)输出波形，用来给通道 1 作为外部的时钟输入。

注意：

如果 RC 不使用，这个可以用来做一个 32 位的计数器。

每次通道 0 计数器计数到 0xFFFF(溢出)的时候，通道 1 计数器的值加 1。

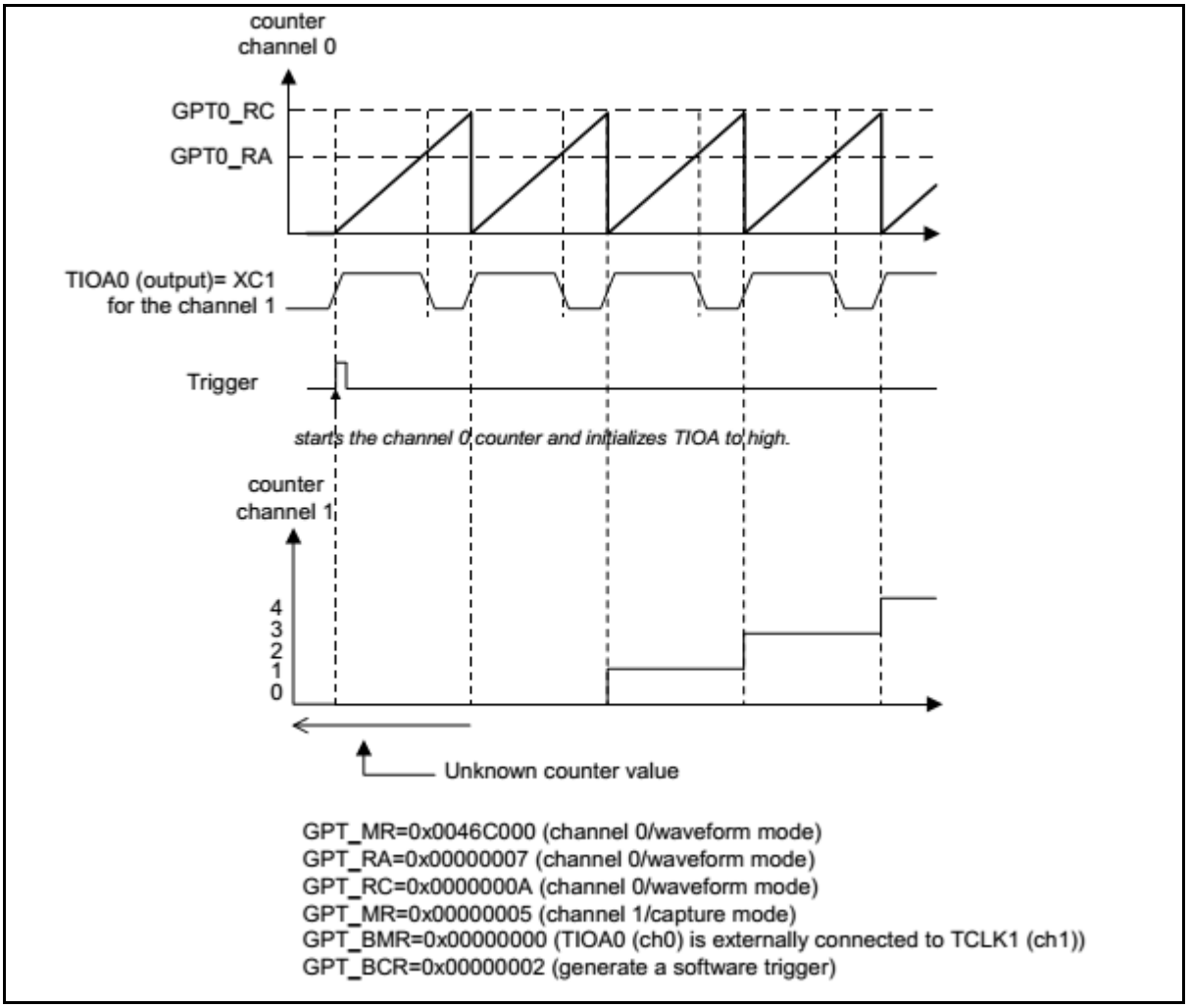


Figure 12-15 通用定时器整体控制模块的应用

12.4.1.1 编程举例

通用定时器 TC0 的一个例子。

我们希望产生一个 10ms 的定时(RTOS 里常用)。在中断里计数器重新开始计数，PCLK 频率为 40MHz。

配置：

- 写 GPT_ECR 寄存器里的 GPT 位，使能 GPT 的时钟。
- 写 GPT_CR 里的 SWRST 位，给 GPT 一个复位，初始化 GPT。
- 配置 GPT_MR 寄存器。选择 1024 分频，波形发生模式，选择 CPCTRG，当 RC 寄存器匹配时计数器会重新开始计数(从 0 开始)，而选择 CPCSTOP 的话，当 RC 寄存器匹配时计数器会停止工作。
- 配置 GPT_RC：控制周期为 10ms，周期 = PCLK/(GPT 时钟分频 * 定时的频率)，也就是 40MHz/(1024*100)。
- 配置 GPT_IER：将 GPT_IER 的 CPCS 位置 1，当 RC 寄存器匹配时，系统会产生一个中断。注意 CPU 的相应中断也需要使能。
- 将 GPT_CR 寄存器里的 CLKEN 位和 SWTRG 位置 1，计数器开始计数，10ms 后会产生中断。

中断处理：

- IRQ 进入，调用中断处理函数。
- 读取 GPT_SR 寄存器并且判断中断源，注意这个寄存器读取后会自动清除。
- 中断处理：将 GPT_CR 寄存器里的 CLKEN 位和 SWTRG 位置 1 重启计数，然后 10ms 后会再次进入中断。

IRQ退出。

12.5 寄存器说明

12.5.1 寄存器

GPT基地址： 0x4005_0000

GPT 通道 0 基地址：0x4005_0000

GPT 通道 1 基地址：0x4005_0100

GPT 通道 2 基地址：0x4005_0200

Table 12-3 GPT 特殊功能寄存器表

Offset Address	Name	Description	R/W	Reset State
0x000 ~ 0x04C	–	保留	–	–
0x050	GPT_ECR	时钟使能寄存器	W	–
0x054	GPT_DCR	时钟禁止寄存器	W	–
0x058	GPT_PMSR	时钟状态寄存器	R	(注意)
0x05C	–	保留	–	–
0x060	GPT_CR	控制寄存器	W	–
0x064	GPT_MR	模式寄存器	R/W	0x00000000
0x068	–	保留	–	–
0x06C	GPT_CSR	状态清除寄存器	W	–
0x070	GPT_SR	状态寄存器	R	0x00000X00
0x074	GPT_IER	中断使能寄存器	W	–
0x078	GPT_IDR	中断禁止寄存器	W	–
0x07C	GPT_IMR	中断使能状态寄存器	R	0x00000000
0x080	GPT_CV	计数值	R	0x00000000
0x084	GPT_RA	捕捉或者比较寄存器 A	R/W	0x00000000
0x088	GPT_RB	捕捉或者比较寄存器 B	R/W	0x00000000
0x08C	GPT_RC	比较寄存器 C	R/W	0x00000000

注：以上是通道 0 的寄存器描述，通道 1、2 与通道 0 寄存器分布相同

Table 12-4 多通道控制寄存器

Offset Address	Name	Description	R/W	Reset State
0x300	GPT_BCR	整体模块控制寄存器	W	–
0x304	GPT_BMR	整体模块工作模式寄存器	R/W	0x00000000

注意： 该寄存器的复位值跟复位后外部管脚的状态有关。

12.5.2 GPT_ECR (时钟使能寄存器)

- Address = Base Address + 0x0050, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
GPT	[1]	W	GPT 时钟使能 0: 无效 1: 使能时钟	0
DBGEN	[31]	W	调试功能使能 0: 无效 1: 使能调试功能 调试功能使能后，在进入调试模式时，计数器将停止计数	0

12.5.3 GPT_DCR (时钟禁止寄存器)

- Address = Base Address + 0x0054, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
GPT	[1]	W	GPT 时钟禁止 0: 无效 1: 禁止时钟	0
DBGEN	[31]	W	调试功能禁止 0: 无效 1: 禁止调试功能	0

12.5.4 GPT_PMSR (时钟状态寄存器)

- Address = Base Address + 0x0058, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
GPT	[1]	R	GPT: 时钟状态 0: 时钟被禁止 1: 时钟被使能	0
IPIDCODE	[29:4]	R	模块的版本号	0x2AAAAAA
DBGEN	[31]	R	DBGEN: 调试功能的状态 0: 调试功能被禁止 1: 调试功能被使能	0

12.5.5 GPT_CR (控制寄存器)

- Address = Base Address + 0x0060, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
SWRST	[0]	W	SWRST: 软件复位 0: 无效 1: 产生软件复位 (GPT_PMSR 寄存器不会被复位)	0
CLKEN	[1]	W	CLKEN: 计数器时钟使能 0: 无效 1: 使能计数时钟如果 CLKDIS=0	0
CLKDIS	[2]	W	CLKDIS: 计数器时钟禁止 0: 无效 1: 禁止计数器时钟	0
SWTRG	[3]	W	SWTRG: 软件触发 0: 无效 1: 产生一个软件触发 在计数时钟使能的前提下，写 1 会产生一个软件触发，让计数器在下一个有效的时钟上升沿开始计数。	0

12.5.6 GPT_MR (模式寄存器，捕捉模式)

- Address = Base Address + 0x0064, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												LDRB		LDRA		WAVE=0	CPCTRG				ABETRG	ETRGEDG		LDBDIS	LDBSTOP	BURST		CLKI	CLKS		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
CLKS	[2:0]	R/W	CLKS[2:0]: 时钟选择 TC0_MCLK 为内部时钟信号，其时钟源可选，具体参照 BMR 寄存器的 HSPD_EN 位说明 XC0, XC1 和 XC2 是外部时钟 000 : TC0_MCLK/1 001 : TC0_MCLK /4 010 : TC0_MCLK /32 011 : TC0_MCLK /128 100 : TC0_MCLK /1024 101 : XC0 110 : XC1 111 : XC2	0
CLKI	[3]	R/W	CLKI: 时钟反向 0: 普通时钟(上升沿计数加 1) 1: 反向时钟(下降沿计数加 1)	0
BURST	[5:4]	R/W	BURST[1:0]: 群脉冲设置 这个选择的信号会跟时钟进行一个与操作，详细请参考“时钟源”章节。 00 : None 01 : XC0 10 : XC1 11 : XC2	00

LDBSTOP	[6]	R/W	LDBSTOP: 载入 RB 停止计数 0: 当 RB 被载入计数值时不停止计数 1: 当 RB 被载入计数值时停止计数 计数停止后，可以通过触发源再次启动计数(从 0x0000 开始)。 如果 TIOA 同时触发启动计数和载入寄存器 RB，那么触发没有效果，也就是载入 RB 优先。	0
LDBDIS	[7]	R/W	LDBDIS: 载入 RB 禁止时钟 0: 当 RB 被载入计数值时不禁止计数器的时钟 1: 当 RB 被载入计数值时禁止计数器的时钟并且停止计数 计数器的时钟被禁止后，可以通过控制寄存器的第 1 位 CLKEN 再次使能	0
ETRGEDG	[9:8]	R/W	ETRGEDG: 外部触发沿选择 外部触发源可以是 TIOA 或者 TIOB，由模式寄存器的第 10 位 ABETRG 选择。当外部触发有效时，会发生 3 个事件： – 复位并且重启计数器(在计数时钟有效的前提下) – 状态寄存器中的 ETRGS 标志位被置位 – 产生 ETRGS 中断(如果该中断被使能的话) 00：无效 01：上升沿 10：下降沿 11：上升下降沿	00
ABETRG	[10]	R/W	ABETRG: 选择 TIOA 或者 TIOB 作为外部触发源 0: 选择 TIOB 作为外部触发 1: 选择 TIOA 作为外部触发 注意：计数器必须在时钟使能的情况下才能开始计数	0
CPCTRG	[14]	R/W	CPCTRG: RC 匹配后的触发 0: RC 匹配不作为触发 1: RC 匹配作为触发 注意：计数器必须在时钟使能的情况下才能开始计数	0
WAVE	[15]	R/W	WAVE: 波形输出模式 0: 捕捉模式 1: 波形输出模式 注意：硬件复位后的默认模式为捕捉模式	0

LDRA	[17:16]	R/W	LDRA[1:0]: 载入 RA RA 寄存器的计数值载入事件选择 00 : 无 01 : TIOA 的上升沿 10 : TIOA 的下降沿 11 : TIOA 的上升下降沿都可以 注意：应用中必须保证 RA 的载入发生在 LDRA 配置完成后的下一个有效时钟沿之后，因为 LDRA 配置完成后的下一个有效时钟沿会让计数器重置，复位后才能正常开始工作。	00
LDRB	[19:18]	R/W	LDRB[1:0]: 载入 RB RB 寄存器的计数值载入事件选择 00 : 无 01 : TIOA 的上升沿 10 : TIOA 的下降沿 11 : TIOA 的上升下降沿都可以	00

12.5.7 GPT_MR (模式寄存器，波形输出模式)

- Address = Base Address + 0x0064, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BSWTRG		BEEVT		BCPC		BCPB		ASWTRG		AEEVT		ACPC		ACPA		WAVE=1	CPCTRG		ENETRG	EEVT		EEVTEDG		CPCDIS	CPCSTOP	BURST		CLKI	CLKS		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
CLKS	[2:0]	R/W	CLKS[2:0]: 时钟选择 TC0_MCLK 为内部时钟信号，其时钟源可选，具体参照 MBR 寄存器的 HSPD_EN 位说明 XC0, XC1 和 XC2 是外部时钟 000 : TC0_MCLK/1 001 : TC0_MCLK /4 010 : TC0_MCLK /32 011 : TC0_MCLK /128 100 : TC0_MCLK /1024 101 : XC0 110 : XC1 111 : XC2	0
CLKI	[3]	R/W	CLKI: 时钟反向 0: 普通时钟(上升沿计数加 1) 1: 反向时钟(下降沿计数加 1)	0
BURST	[5:4]	R/W	BURST[1:0]: 群脉冲设置 这个选择的信号会跟时钟进行一个与操作，详细请参考“时钟源”章节。 00 : None 01 : XC0 10 : XC1 11 : XC2	00

CPCSTOP	[6]	R/W	CPCSTOP: 比较寄存器 RC 匹配后停止计数 0: 当 RC 匹配时不停止计数 1: 当 RC 匹配时停止计数 计数停止后，可以通过触发源再次启动计数(从 0x0000 开始)。 如果 RC 匹配同时触发启动计数和停止计数，那么触发没有效果，也就是会停止计数。	0
CPCDIS	[7]	R/W	CPCDIS: 比较寄存器 RC 匹配后禁止时钟 0: 当 RC 匹配时不禁止计数器的时钟 1: 当 RC 匹配时禁止计数器的时钟并且停止计数 计数器的时钟被禁止后，可以通过控制寄存器的第 1 位 CLKEN 再次使能	0
EEVTEDG	[9:8]	R/W	EEVTEDG: 外部事件沿选择 外部事件的源由模式寄存器的[11:10]位 EEVT 选择。 当外部触发产生时，会有下面 5 个事件发生： 状态寄存器里的 ETRGS 标志被置位 产生 ETRGS 中断(如果该中断被使能) 如果模式寄存器的第 12 位 ENETRG 是高，在下一个有效的时钟上升沿，计数器被复位并且重新开始计数 根据 AEEVT(模式寄存器[21:20]位)的设置，TIOA 管脚可以变高，变低，翻转或者不变 根据 BEEVT(模式寄存器[29:28]位)的设置，TIOB 管脚可以变高，变低，翻转或者不变 00：无 01：上升沿 10：下降沿 11：上升下降沿	00
EEVT	[11:10]	R/W	EEVT: 外部事件选择 从下面 4 个管脚选择外部事件源： 00 : TIOB 01 : XC0 10 : XC1 11 : XC2 如果选择了 TIOB，那么模式将变成单波形输出模式，TIOA 为输出，TIOB 为输入，并且下面的寄存器设置为无效设置： – BSWTRG，模式寄存器的[31:30]位 – BEEVT，模式寄存器的[29:28]位 – BCPC，模式寄存器的[27:26]位	00

			– BCPB, 模式寄存器的[25:24]位 – 比较寄存器 B 如果选择了外部时钟，那么模式则为双波形输出模式，TIOA 和 TIOB 都为输出。	
ENETRG	[12]		ENETRG: 使能外部触发 该位决定外部事件是否作为复位并且重启计数的触发。 外部事件源由模式寄存器的[11:10]位 EEVT 选择。 0: 外部事件不会复位并且重启计数，只能控制 TIOA 和 TIOB 1: 外部触发会复位并且重启计数 注意：计数器必须在时钟使能的情况下才能开始计数	0
CPCTRG	[14]	R/W	CPCTRG: RC 匹配后的触发 该位决定 RC 匹配是否作为复位并且重启计数的触发。 0: RC 匹配不作为触发 1: RC 匹配作为触发 注意：计数器必须在时钟使能的情况下才能开始计数	0
WAVE	[15]	R/W	WAVE: 波形输出 0: 捕捉模式 1: 波形输出模式	0
ACPA	[17:16]	R/W	ACPA[1:0]: TIOA 比较寄存器 A 匹配 计数值和比较寄存器 A 匹配后，TIOA 输出状态的变化 00：不变 01：变高 10：变低 11：翻转 注意：如果多个能控制 TIOA 输出的事件同时发生，那么只有一个事件能控制 TIOA 的输出，事件优先级如下： – ASWTRG (优先级最高) – AEEVT – ACPC – ACPA	00

ACPC	[19:18]	R/W	ACPC[1:0]: TIOA 比较寄存器 C 匹配 计数值和比较寄存器 C 匹配后，TIOA 输出状态的变化 00： 不变 01： 变高 10： 变低 11： 翻转	00
AEEVT	[21:20]	R/W	AEEVT[1:0]: TIOA 外部事件 外部事件发生后，TIOA 输出状态的变化。外部事件的源由模式寄存器的[11:10]位 EEVT 选择。 00： 不变 01： 变高 10： 变低 11： 翻转	00
ASWTRG	[23:22]	R/W	ASWTRG[1:0]: TIOA 软件触发 软件触发后，TIOA 输出状态的变化 00： 不变 01： 变高 10： 变低 11： 翻转	00
BCPB	[25:24]	R/W	BCPB[1:0]: TIOB 比较寄存器 B 匹配 计数值和比较寄存器 B 匹配后，TIOB 输出状态的变化 00： 不变 01： 变高 10： 变低 11： 翻转 注意：如果多个能控制 TIOB 输出的事件同时发生，那么只有一个事件能控制 TIOB 的输出，事件优先级如下： – BSWTRG (优先级最高) – BEEVT – BCPC – BCPB	00

BCPC	[27:26]	R/W	BCPC[1:0]: TIOB 比较寄存器 C 匹配 计数值和比较寄存器 C 匹配后，TIOB 输出状态的变化。该寄存器只有在 TIOB 不是作为输入的时候有效(参考模式寄存器的[11:10]位 EEVT) 00： 不变 01： 变高 10： 变低 11： 翻转	00
BEEVT	[29:28]	R/W	BEEVT[1:0]: TIOB 外部事件 外部事件发生后，TIOB 输出状态的变化。外部事件的源由模式寄存器的[11:10]位 EEVT 选择。该寄存器只有在 TIOB 不是作为输入的时候有效(参考模式寄存器的[11:10]位 EEVT) 00： 不变 01： 变高 10： 变低 11： 翻转	00
BSWTRG	[31:30]	R/W	BSWTRG[1:0]: TIOB 软件触发 软件触发后，TIOB 输出状态的变化。该寄存器只有在 TIOB 不是作为输入的时候有效(参考模式寄存器的[11:10]位 EEVT) 00： 不变 01： 变高 10： 变低 11： 翻转	00

12.5.8 GPT_CSR (状态清除寄存器)

- Address = Base Address + 0x006C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description	Reset Value
COVFS	[0]	W	COVFS: 计数器溢出中断 0: 无效 1: 清除 COVFS 中断	0
LOVRS	[1]	W	LOVRS: 载入溢出中断 0: 无效 1: 清除 LOVRS 中断	0
CPAS	[2]	W	CPAS: 比较寄存器 A 匹配中断 0: 无效 1: 清除 CPAS 中断.	0
CPBS	[3]	W	CPBS: 比较寄存器 B 匹配中断 0: 无效 1: 清除 CPBS 中断.	0
CPCS	[4]	W	CPCS: 比较寄存器 C 匹配中断 0: 无效 1: 清除 CPCS 中断.	0
LDRAS	[5]	W	LDRAS: 载入寄存器 A 中断 0: 无效 1: 清除 LDRAS 中断	0

LDRBS	[6]	W	LDRBS: 载入寄存器 B 中断 0: 无效 1: 清除 LDRBS 中断	0
ETRGS	[7]	W	ETRGS: 外部触发中断 0: 无效 1: 清除 ETRGS 中断	0

12.5.9 GPT_SR (状态寄存器)

- Address = Base Address + 0x0070, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													TCLKS	TIOAS	TIOBS						MTIOB	MTIOA	CLKSTA	ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

注意: 这个寄存器为“读-清除”寄存器，读取这个寄存器后，下面这些位如果被是 1，那么读取后会自动被清除：COVFS, CPAS, CPBS, CPCS, ETRGS, TIOBS, TIOAS 和 TCLKS。在调试的时候，为了避免仿真器在暂停运行时对寄存器的遍历读取，用户可以使用镜像寄存器。

Name	Bit	Type	Description	Reset Value
COVFS	[0]	R	COVFS: 计数器溢出状态 当计数器计数溢出时，该位会被置 1，也就是当计数值从最大值 0xFFFF 变为 0x0000 时，计数值溢出。 0: 没有检测到溢出 1: 在上一次读取 GPT_SR 后，检测到计数器溢出	0
LOVRS	[1]	R	LOVRS: 载入溢出状态 检测到载入溢出时，该位会被置 1。如果捕捉寄存器 A 或者 B 在读取前又被再次载入，那么载入溢出发生。 0: 没有检测到载入溢出 1: 在上一次读取 GPT_SR 后，检测到载入溢出	0
CPAS	[2]	R	CPAS: 比较寄存器 A 的匹配状态 当计数值计数到比较寄存器 A 的值时，该位会被置 1. 0: 在上一次读取 GPT_SR 后，比较寄存器 A 没有匹配 1: 在上一次读取 GPT_SR 后，比较寄存器 A 发生了匹配	0
CPBS	[3]	R	CPBS: 比较寄存器 B 的匹配状态 当计数值计数到比较寄存器 B 的值时，该位会被置 1. 0: 在上一次读取 GPT_SR 后，比较寄存器 B 没有匹配	0

			1: 在上一次读取 GPT_SR 后，比较寄存器 B 发生了匹配	
CPCS	[4]	R	CPCS: 比较寄存器 C 的匹配状态 当计数值计数到比较寄存器 C 的值时，该位会被置 1. 0: 在上一次读取 GPT_SR 后，比较寄存器 C 没有匹配 1: 在上一次读取 GPT_SR 后，比较寄存器 C 发生了匹配	0
LDRAS	[5]	R	LDRAS: 寄存器 A 的载入状态 0: 寄存器 A 没有被载入 1: 在上一次读取 GPT_SR 后，寄存器 A 被载入了计数值	0
LDRBS	[6]	R	LDRBS: 寄存器 B 的载入状态 0: 寄存器 B 没有被载入 1: 在上一次读取 GPT_SR 后，寄存器 B 被载入了计数值	0
ETRGS	[7]	R	ETRGS: 外部触发状态 外部触发被检测到时，该位会被置 1. 0: 外部触发没有被检测到 1: 在上一次读取 GPT_SR 后，检测到了外部触发 在捕捉模式： 触发沿的极性由模式寄存器的[9:8]位 ETRGEDG 选择，触发源由模式寄存器的第 10 位 ABETRG 选择。 在波形发生模式： 触发沿的极性由模式寄存器的[9:8]位 ETRGEDG 选择，触发方式由模式寄存器的第 12 位 ENETRG 设置。	0
CLKSTA	[8]	R	CLKSTA: 时钟状态 0: 时钟禁止 1: 时钟使能	0
MTIOA	[9]	R	MTIOA: TIOA 镜像 该位直接镜像了 TIOA 管脚的值 由于硬件复位后 TIOA 输入状态不定，所以复位值不定。	0

MTIOB	[10]	R	MTIOB: TIOB 镜像 该位直接镜像了 TIOB 管脚的值 由于硬件复位后 TIOB 输入状态不定，所以复位值不定。	0
TIOBS	[16]	R	TIOBS: TIOB 状态 0: 在上一次读取该寄存器后，TIOB 管脚发生了至少一次的电平变化 1: 在上一次读取该寄存器后，TIOB 管脚没有发生电平变化	0
TIOAS	[17]	R	TIOAS: TIOA 状态 0: 在上一次读取该寄存器后，TIOA 管脚发生了至少一次的电平变化 1: 在上一次读取该寄存器后，TIOA 管脚没有发生电平变化	0
TCLKS	[18]	R	TCLKS: TCLK 状态 0: 在上一次读取该寄存器后，TCLK 管脚发生了至少一次的电平变化 1: 在上一次读取该寄存器后，TCLK 管脚没有发生电平变化	0

12.5.10 GPT_IER (中断使能寄存器)

- Address = Base Address + 0x0074, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description	Reset Value
COVFS	[0]	W	COVFS: 计数器溢出中断 0: 无效 1: 使能 COVFS 中断.	0
LOVRS	[1]	W	LOVRS: 载入溢出中断 0: 无效 1: 使能 LOVRS 中断.	0
CPAS	[2]	W	CPAS: 比较寄存器 A 匹配中断 0: 无效 1: 使能 CPAS 中断.	0
CPBS	[3]	W	CPBS: 比较寄存器 B 匹配中断 0: 无效 1: 使能 CPBS 中断.	0
CPCS	[4]	W	CPCS: 比较寄存器 C 匹配中断 0: 无效 1: 使能 CPCS 中断.	0
LDRAS	[5]	W	LDRAS: 载入寄存器 A 中断 0: 无效 1: 使能 LDRAS 中断.	0

LDRBS	[6]	W	LDRBS: 载入寄存器 B 中断 0: 无效 1: 使能 LDRBS 中断.	0
ETRGS	[7]	W	ETRGS: 外部触发中断 0: 无效 1: 使能 ETRGS 中断.	0

12.5.11 GPT_IDR (中断禁止寄存器)

- Address = Base Address + 0x0078, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description	Reset Value
COVFS	[0]	W	COVFS: 计数器溢出中断 0: 无效 1: 禁止 COVFS 中断.	0
LOVRS	[1]	W	LOVRS: 载入溢出中断 0: 无效 1: 禁止 LOVRS 中断.	0
CPAS	[2]	W	CPAS: 比较寄存器 A 匹配中断 0: 无效 1: 禁止 CPAS 中断.	0
CPBS	[3]	W	CPBS: 比较寄存器 B 匹配中断 0: 无效 1: 禁止 CPBS 中断.	0
CPCS	[4]	W	CPCS: 比较寄存器 C 匹配中断 0: 无效 1: 禁止 CPCS 中断.	0
LDRAS	[5]	W	LDRAS: 载入寄存器 A 中断 0: 无效 1: 禁止 LDRAS 中断.	0

LDRBS	[6]	W	LDRBS: 载入寄存器 B 中断 0: 无效 1: 禁止 LDRBS 中断.	0
ETRGS	[7]	W	ETRGS: 外部触发中断 0: 无效 1: 禁止 ETRGS 中断.	0

12.5.12 GPT_IMR (中断使能状态寄存器)

- Address = Base Address + 0x007C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description	Reset Value
COVFS	[0]	R	COVFS: 计数器溢出中断 0: COVFS 中断处于禁止状态. 1: COVFS 中断处于使能状态.	0
LOVRS	[1]	R	LOVRS: 载入溢出中断 0: LOVRS 中断处于禁止状态. 1: LOVRS 中断处于使能状态.	0
CPAS	[2]	R	CPAS: 比较寄存器 A 匹配中断 0: CPAS 中断处于禁止状态. 1: CPAS 中断处于使能状态.	0
CPBS	[3]	R	CPBS: 比较寄存器 B 匹配中断 0: CPBS 中断处于禁止状态. 1: CPBS 中断处于使能状态.	0
CPCS	[4]	R	CPCS: 比较寄存器 C 匹配中断 0: CPCS 中断处于禁止状态. 1: CPCS 中断处于使能状态.	0
LDRAS	[5]	R	LDRAS: 载入寄存器 A 中断 0: LDRAS 中断处于禁止状态. 1: LDRAS 中断处于使能状态.	0

LDRBS	[6]	R	LDRBS: 载入寄存器 B 中断 0: LDRBS 中断处于禁止状态. 1: LDRBS 中断处于使能状态.	0
ETRGS	[7]	R	ETRGS: 外部触发中断 0: ETRGS 中断处于禁止状态. 1: ETRGS 中断处于使能状态.	0

12.5.13 GPT_CV (计数值寄存器)

- Address = Base Address + 0x0080, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
CV	[15:0]	R	CV[15:0]: 计数器的计数值 该寄存器为计数器的实时计数值，最大值为 0xFFFF = 65535. 当触发事件发生时，在下一个有效的时钟上升沿，计数器会被复位到 0x0000。	0x0000

12.5.14 GPT_RA (寄存器A，捕捉模式)

- Address = Base Address + 0x0084, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
RA	[15:0]	R	RA: 寄存器 A 的值 当 TIOA 上有效沿被检测到时，这个寄存器会被载入当前计数器的值。有效沿由模式寄存器的[17:16]位 LDRA 决定。 当这个寄存器被载入时，会发生 2 个事件： – 状态寄存器中 LDRAS 标志被置 1 – 产生 LDRAS 中断(如果该中断使能) 如果计数器停止或者时钟被禁止，那么这个寄存器不能被载入值。	0x0000

12.5.15 GPT_RB (寄存器B，捕捉模式)

- Address = Base Address + 0x0088, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
RB	[15:0]	R	RB: 寄存器 B 的值 当 TIOB 上有效沿被检测到时，这个寄存器会被载入当前计数器的值。有效沿由模式寄存器的[19:18]位 LDRB 决定。 当这个寄存器被载入时，会发生 4 个事件： – 状态寄存器中 LDRBS 标志被置 1 – 产生 LDRBS 中断(如果该中断使能) – 计数器时钟可以被禁止，由模式寄存器的第 7 位 LDBDIS 决定 – 计数器可以停止计数，由模式寄存器的第 6 位 LDBSTOP 决定	0x0000

12.5.16 GPT_RC (寄存器C)

- Address = Base Address + 0x008C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
RC	[15:0]	R	RC: 寄存器 C 的值 当计数器计到该寄存器设定的值，会发生 3 个事件： – 状态寄存器中的 CPCS 标志位会被置 1 – 产生 CPCS 中断(如果该中断使能) – 如果 CPCTRG (模式寄存器第 14 位)是 1，计数器重置并且重新开始计数。	0x0000

12.5.17 GPT_RA (寄存器A，波形发生模式)

- Address = Base Address + 0x0084, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
RA	[15:0]	RW	RA: 寄存器 A 的值 当计数器计到该寄存器设定的值，会发生 3 个事件： 状态寄存器中的 CPAS 标志被置 1。 产生 CPAS 中断(如果该中断使能) TIOA 管脚根据 ACPA(模式寄存器[17:16]位)的设定，变高，变低，翻转，或者不变。	0x0000

12.5.18 GPT_RB (寄存器B，波形发生模式)

- Address = Base Address + 0x0088, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
RB	[15:0]	RW	RB: 寄存器 B 的值 当计数器计到该寄存器设定的值，会发生 3 个事件： 状态寄存器中的 CPBS 标志被置 1。 产生 CPBS 中断(如果该中断使能) TIOB 管脚根据 BCPB(模式寄存器[25:24]位)的设定，变高，变低，翻转，或者不变。 该寄存器只有在 TIOB 不是作为输入的时候有效(参考模式寄存器的[11:10]位 EEVT)	0x0000

12.5.19 GPT_RC（寄存器C，波形发生模式)

- Address = Base Address + 0x008C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
RC	[15:0]	RW	RC: 寄存器 C 的值 当计数器计到该寄存器设定的值，会发生 7 个事件： – 状态寄存器中的 CPCS 标志被置 1。 – 产生 CPCS 中断(如果该中断使能) – 如果 CPCTRG (模式寄存器的第 14 位)是 1，计数器在下一个有效时钟沿复位并且重新开始计数 – 计数器时钟可以被禁止，由 CPCDIS (模式寄存器的第 7 位) 决定 – 计数器可以被停止，由 CPCSTOP (模式寄存器的第 6 位)决定 – TIOA 根据 ACPC(模式寄存器[19:18]位)的选择变高，变低，翻转，或者不变 – TIOB 根据 BCPC(模式寄存器[27:26]位)的选择变高，变低，翻转，或者不变	0x0000

12.5.20 GPT_BCR (整体模块控制寄存器)

- Address = Base Address + 0x0300, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
SWRST	[0]	W	SWRST: 软件复位 在 3 个定时器通道上同时产生软件复位 0: 无效 1: 产生软件复位	0
TCSYNC	[1]	W	TCSYNC: 同步位 给 3 个定时器通道同时产生软件触发信号，将计数器重置并且在下一个有效时钟沿开始计数。 0: 无效 1: 让 3 个通道的定时器同时重置并且开始计数	0

12.5.21 GPT_BMR (整体模块工作模式寄存器)

- Address = Base Address + 0x0304, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																							HSPD_EN			TC2XC2S		TC1XC1S		TC0XC0S	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
TC0XC0S	[1:0]	RW	TC0XC0S: TCK0 XC0 选择. 通道 0 外部时钟 XC0 的源选择 00 : TCLK0 01： 无 10 : TIOA1 (通道 1 的输出) 11 : TIOA2 (通道 2 的输出)	00
TC1XC1S	[3:2]	RW	TC1XC1S: TCK1 XC1 选择 通道 1 外部时钟 XC1 的源选择 00 : TCLK1 01： 无 10 : TIOA0 (通道 0 的输出) 11 : TIOA2 (通道 2 的输出)	00
TC2XC2S	[5:4]	RW	TC2XC2S: TCK2 XC2 选择 通道 2 外部时钟 XC2 的源选择 00 : TCLK2 01： 无 10 : TIOA0 (通道 0 的输出) 11 : TIOA1 (通道 1 的输出)	00
HSPD_EN	[8]	RW	C_CLK时钟源选择 0： 选择PCLK 1： 选择内部高速时钟96MHz（需要SYSCON的CLCR寄存器HFOSCEN位使能）	0

13 32位 定时器/计数器 (TC1)

13.1 概述

本章节介绍32位定时器/计数器的使用。定时器/计数器(简称TC1)有三种工作模式：比较匹配模式，输入捕捉模式，输出翻转或者称为PWM模式。TC1可以通过特定的管脚输出PWM信号，并且TC1的时钟源支持从外部引入。

13.1.1 主要特性

- 可编程的时钟源，支持从外部管脚输入
- 位数可编程的计数器，支持8位，10位，16位，32位配置
- 单次计数或者重复循环计数
- 计数值匹配和溢出
- 捕捉，支持捕捉比较器的输出脉冲
- 支持上升沿，下降沿和上升下降同时捕捉
- 两个捕捉寄存器，可捕捉两个沿
- 输出翻转功能
- PWM输出
- 周期和频率可编程
- 有效电平和空闲电平可编程
- 32位计数器可支持最高38位分辨率，16位计数器可支持最高22位分辨率，支持扩展功能
- 带Debug选项
- 支持作为ADC的触发源
- 支持比较器的输出信号作为定时器的启动信号

13.1.2 管脚描述

Table 13-1 管脚描述

Pin Name	Function	I/O Type	Comments
TCLK	外部时钟源	I	
TCAP	捕捉输入	I	
TPWM	PWM / 计数器输出管脚	O	

13.2 功能描述

13.2.1 模块框图

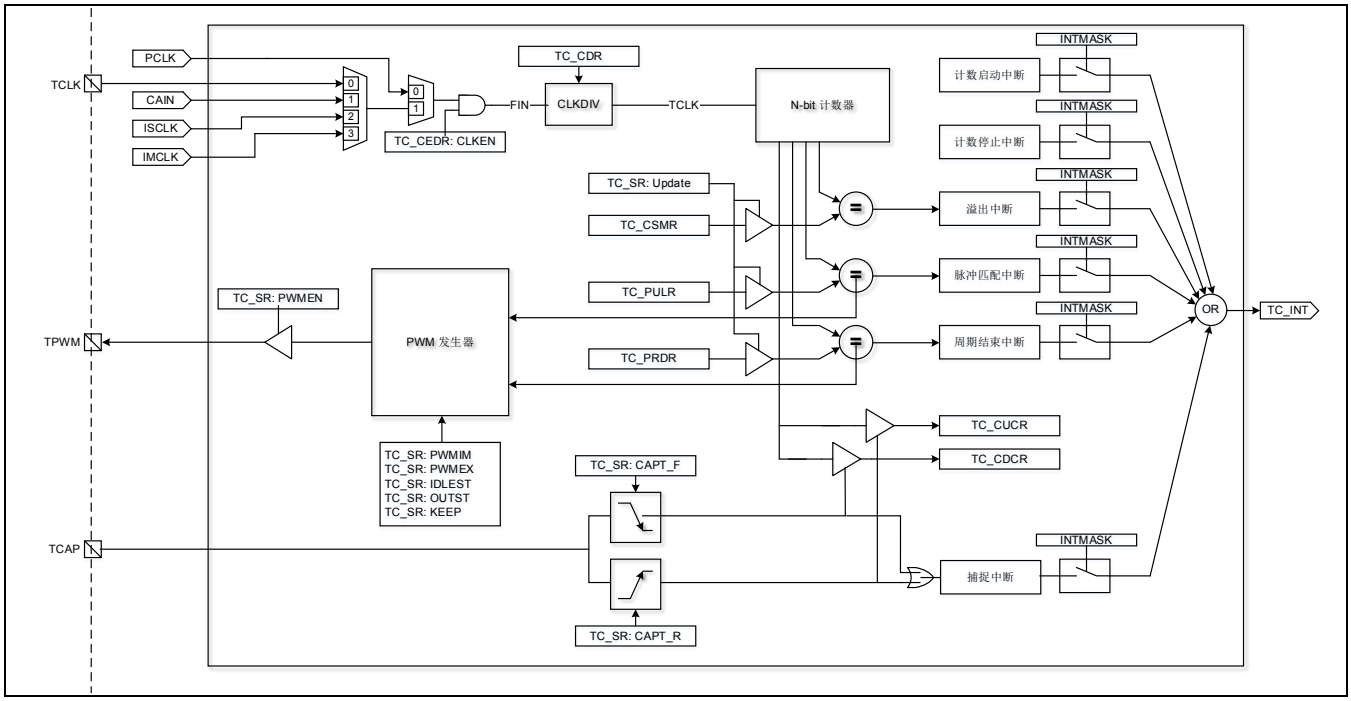


Figure 13-1 TC1模块框图

13.2.2 计数器位数

计数器的位数可以由一个寄存器控制，TC1_CSMR (Counter Size Mask Register). 如果SIZE[4:0]的值是n，那么TC将变成一个(n+1)位数的定时器，也就是说，定时器将从1计数到2^(n+1)-1为止。

13.2.3 计数器时钟

13.2.3.1 时钟源

TC可以使用两种不同的时钟(同步或者异步时钟)。同步时钟为PCLK，而异步时钟则是从下列源中任意选择的一个：晶振，TCLK管脚，比较器输出。如果要使用外部时钟源TCLK，那么在开始定时器之前，需要先将IO管脚配置好(配置成TCLK)。

13.2.3.2 计数器时钟

基于FIN的计数器时钟由TC1_CDR (Clock Divider Register)中的DIVM[7:0]和DIVN[3:0]决定。计数器时钟的频率TCCLK由FIN和内部时钟分频器(DIVM和DIVN)共同确定：

TCCLK = FIN / 2^DIVN / (DIVM +1) (例如，DIVN=3/DIVM=3，FIN=16MHz，那么TCCLK=500KHz)

计数器分辨率 = 1 / TCCLK

注意 ：在DIVN不等于0的时候，禁止将DIVM设为0

13.2.3.3 Debug选项

TC debug选项用来选择当CPU被调试器暂停的时候，计数器是否也同时停止计数。

13.2.4 芯片内部的触发源

当TC1_SR寄存器中的HWTRIG_OUT位被置位后，TC1可以作为ADC的触发源使用。TC1的脉冲比较中断可以作为ADC的转换开始触发源。详细内容请参考ADC章节。

当TC1_SR寄存器中的HWTRIG_IN位被置位后，比较器可以作为TC1启动计数的触发源，跟TC1_CSR寄存器中的START位作用相同，也就是当比较器出书了一个触发脉冲时，TC1可以开始计数。详细内容请参考比较器章节。

13.2.5 更新控制寄存器

所有跟计数器工作有关的寄存器值都会在定时器的开始位被置位的时候载入到计数的逻辑电路中，而当计数器在正常工作的时候，所有寄存器都会被冻结住无法更改，除非发生了软件复位或者操作了时钟使能控制位。如果需要在计数器工作的时候(on-the-fly)更改配置，那么请使用"UPDATE"位的更新机制。当TC1_SR中的UPDATE位被置位的时候才可以修改寄存器，当寄存器修改完成后，需要清除TC1_SR中的UPDATE位，告诉计数逻辑电路去载入寄存器的值。在内部逻辑中，所有的改动只有在循环模式中的计数开始或者周期结束事件发生后，才会生效。

注意时钟源选择寄存器是个例外，这个寄存器只有当计数器没有开始，并且时钟被禁止的情况下，才可以进行修改，否则修改无效。

13.2.6 连续计数模式

当TC1_SR中CNTM位被置位时，定时器工作在连续计数模式。在这个模式下，计数值逐次增加，直到2^(SIZE+1)-1为止，其中SIZE在TC1_CSMR (Counter Size Mask Register)寄存器中设置。当TC1_SR寄存器中的REPEAT位是0的时候，计数值在计数到2^(SIZE+1)-1后会被清零，这时定时器会产生停止中断和溢出中断。

13.2.6.1 匹配和溢出

定时器有两个比较值可以分别用来产生两种匹配中断，一个叫脉冲匹配中断，另一个叫周期结束中断。

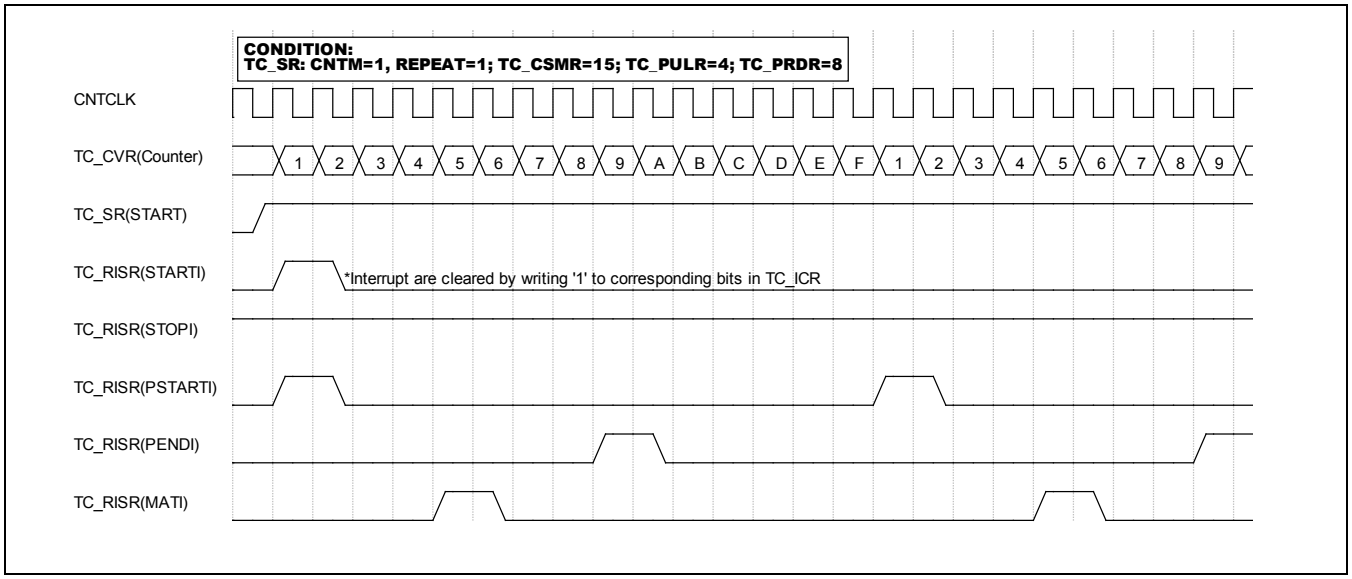


Figure 13-2 匹配和溢出时序

当定时器开始计数，立即产生计数启动中断和周期开始中断。当计数值跟TC1_PULR中PULSE的值相等的时候，定时器产生脉冲匹配中断，而当计数值跟TC1_PRDR中PERIOD值相等的时候，产生周期结束中断。当计数器溢出的时候，则产生溢出中断。TC1_SR中REPEAT位用来配置定时器的工作模式是单次(one-short)还是循环重复。如果REPEAT位被置位，那么计数溢出后计数器的值会被重置到1，然后自动重新开始计数。

将TC1_CCR寄存器的START位写1后，可以清除TC1_SR中的START启动状态，并且同时停止定时器。定时器停止的方式由TC1_SR寄存器中STOPHOLD和STOPCLEAR位决定。

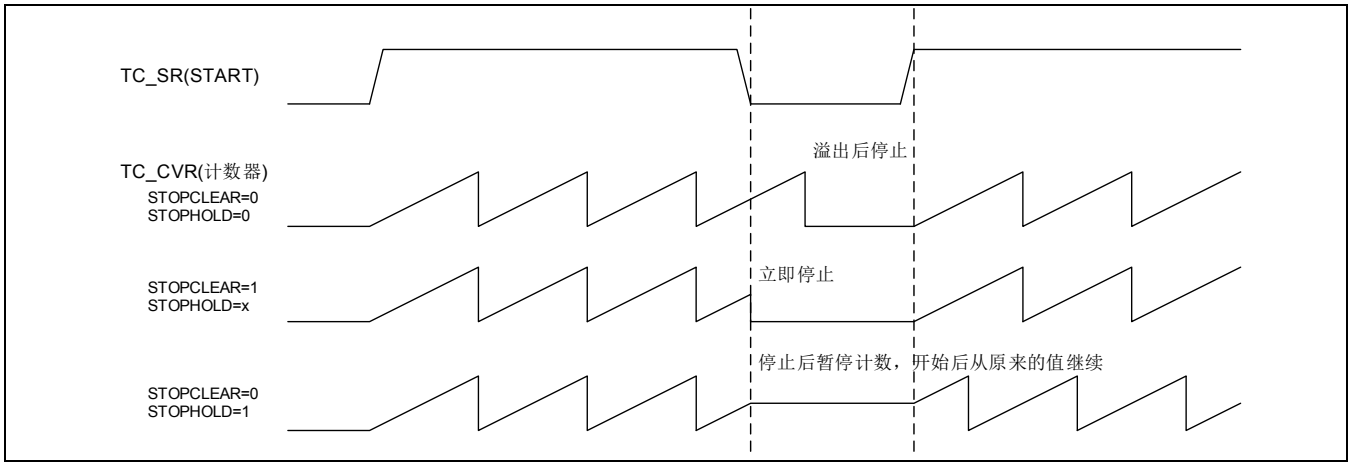


Figure 13-3 基于STOP, STOPCLEAR, STOPHOLD的计数控制

如果STOPHOLD和STOPCLEAR都是0，清除TC1_SR的START位后，计数器会计数到溢出后再停止，并且计数值自动清零。如果STOPCLEAR被置位，清除TC1_SR的START位后，计数器会立即停止。如果需要暂停计数，需要将STOPHOLD置1并且保持STOPCLEAR为0 (STOPCLEAR位的优先权高于STOPHOLD)，这种情况下停止计数后，计数器将保留计数值，再重新开始时会从保留的值开始继续计数。

13.2.6.2 输入捕捉

TC可以捕捉外部的输入信号，将计数值存入捕捉寄存器TC1_CUCR (Capture Up Counter Register)和TC1_CDCR (Capture Down Counter Register)中。这个工作模式用来计算外部信号的时间差异。外部输入触发信号的上升或者下降沿可以由寄存器预先定义好的值进行选择，当检测到预设好的上升或下降沿的时候，相应的计数器计数值会被复制到寄存器TC1_CUCR或TC1_CDCR中。

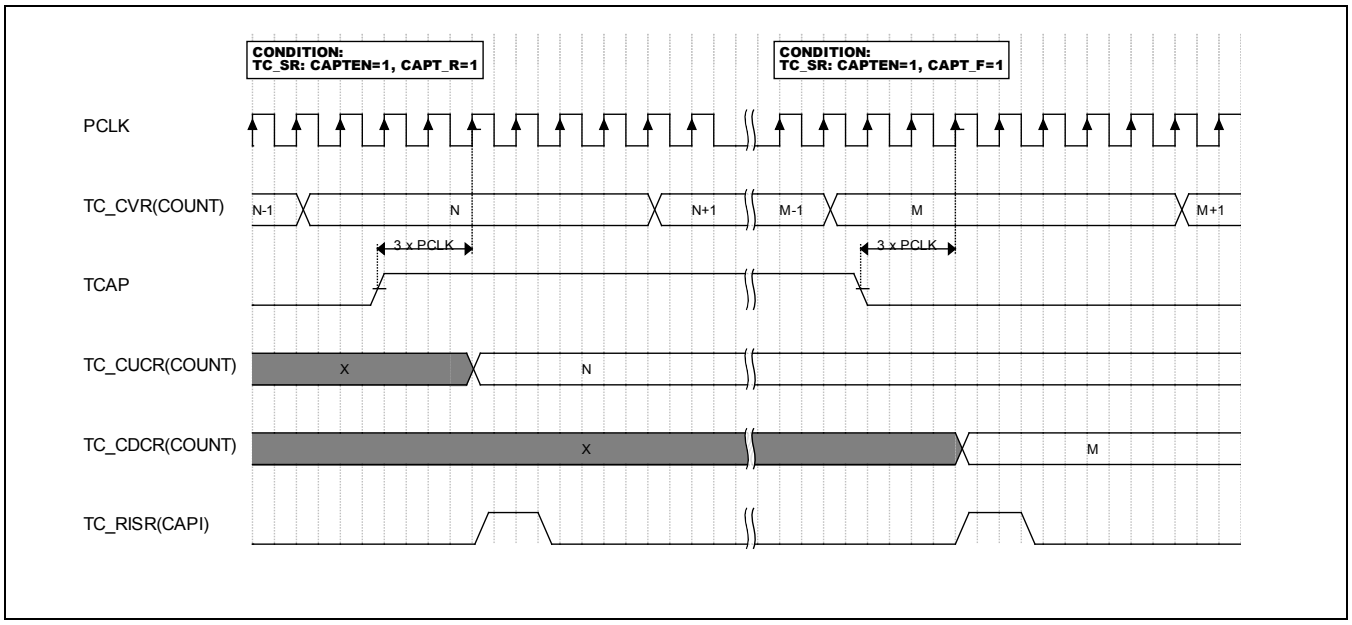


Figure 13-4 输入信号捕捉时序

对于输入捕捉模式，定时器的时钟必须选择PCLK，并且TCAP管脚输入的外部信号必须至少保持3个PCLK的时钟周期长度，以免被当作毛刺信号过滤掉。

13.2.7 周期匹配模式

当TC1_SR寄存器中的CNTM位为0，TC工作在周期匹配模式。这个模式下，计数器从1一直计数到TC1_PRDR中PERIOD设置的值。当计数到PERIOD值时，定时器产生周期结束中断。如果TC1_SR寄存器中REPEAT位为1，计数器在周期结束后会自动从1开始继续循环计数，如果REPEAT为0，计数器则被清零并且停止计数。

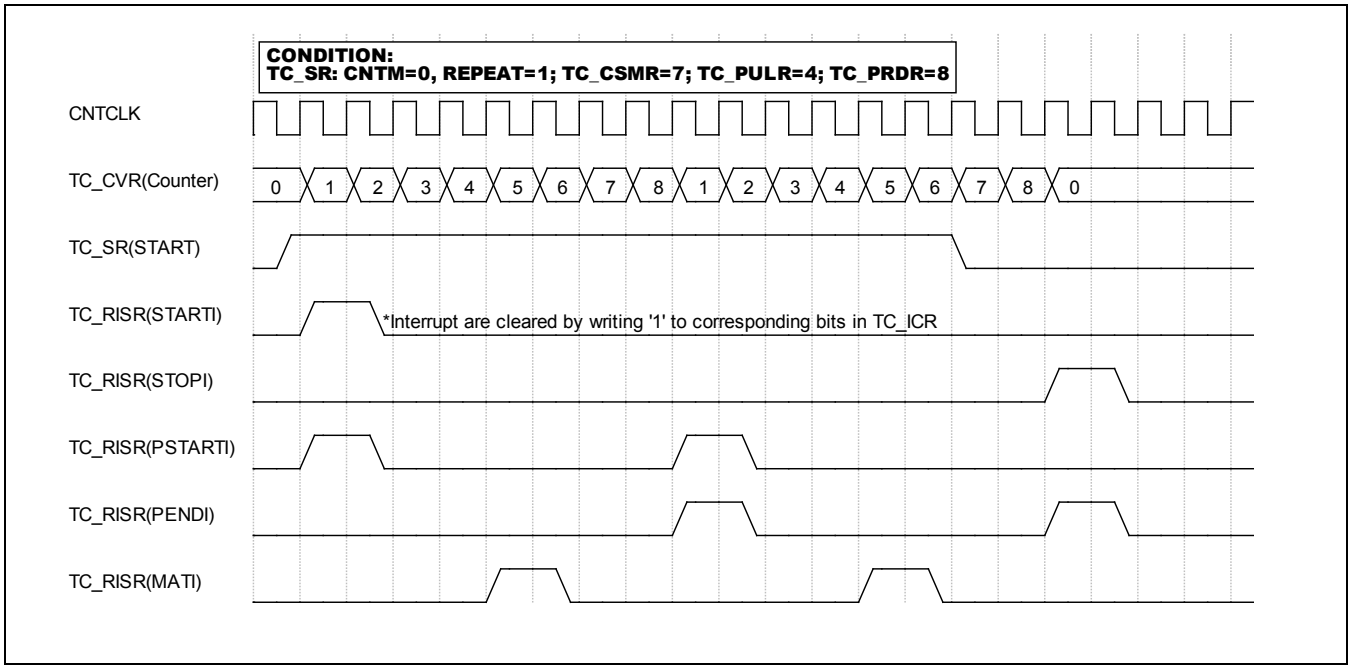


Figure 13-5 周期匹配模式时序

该模式下，当计数器的值跟TC1_PULR和TC1_PRDR相等时，定时器会分别产生脉冲匹配中断和周期结束中断。

基于TC1_SR的PWMIM位，TC可以产生两种类型的输出信号，输出翻转信号和PWM信号。注意为了将信号输出到芯片的管脚上，TC1_SR中的PWMEN位必须为1，并且管脚的复用功能设置也必须正确。

13.2.7.1 输出翻转功能

使能输出翻转功能后，定时器的输出会在检测到周期结束时翻转。比如，写0x08到TC1_PRDR，计数值会一直增加直到0x08，并且在此时翻转，所以输出的波形周期为：

2 x TCCLK x PERIOD

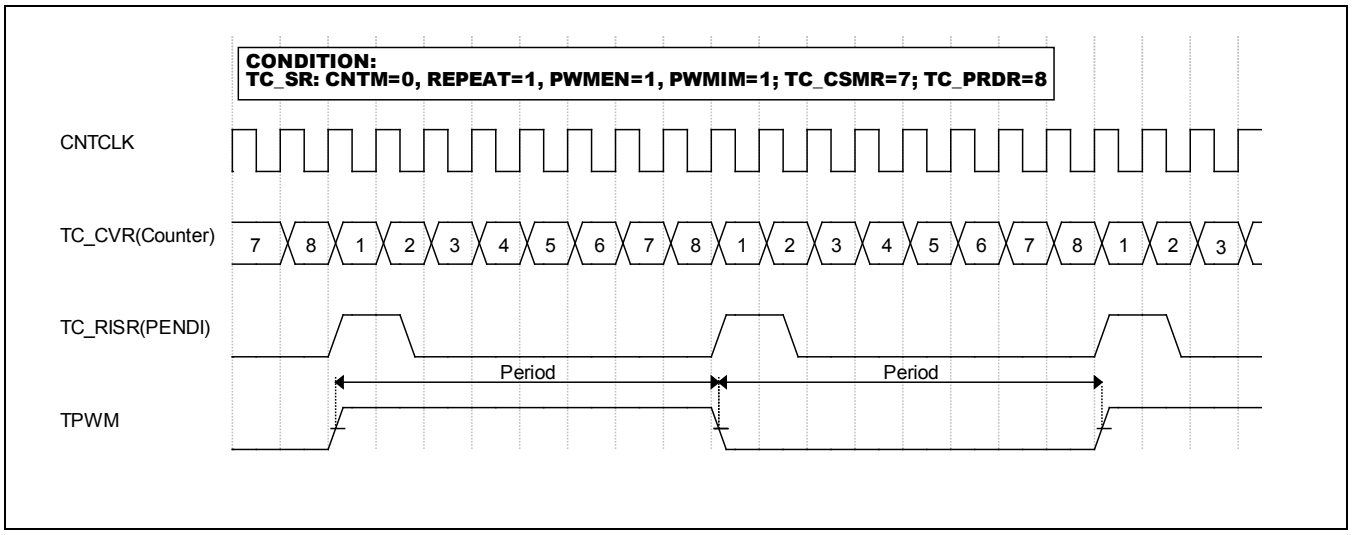


Figure 13-6 输出翻转功能的时序

13.2.7.2 PWM功能

TC可以用来产生PWM (Pulse Width Modulation)输出。在这模式下，当定时器启动时，TC1_PULR寄存器的PULSE会被载入到PWM逻辑电路中。 TC1_PULR的值必须小于或等于TC1_PRDR的值。

PWM输出信号的初始值由TC1_SR寄存器中的OUTST位决定。当OUTST为1的时候，输出信号为高；当OUTST为0的时候，输出信号为低。当计数器的值等于TC1_PULR的时候，定时器产生脉冲匹配事件，并且此时PWM输出翻转成与OUTST位相反的状态。之后计数器继续工作，直到等于TC1_PRDR中的PERIOD值，此时定时器产生周期结束事件，并且PWM输出再次翻转。这时如果REPEAT位为1，计数器则从1开始重新计数，如果REPEAT为0，那么定时器停止并且计数器清零。

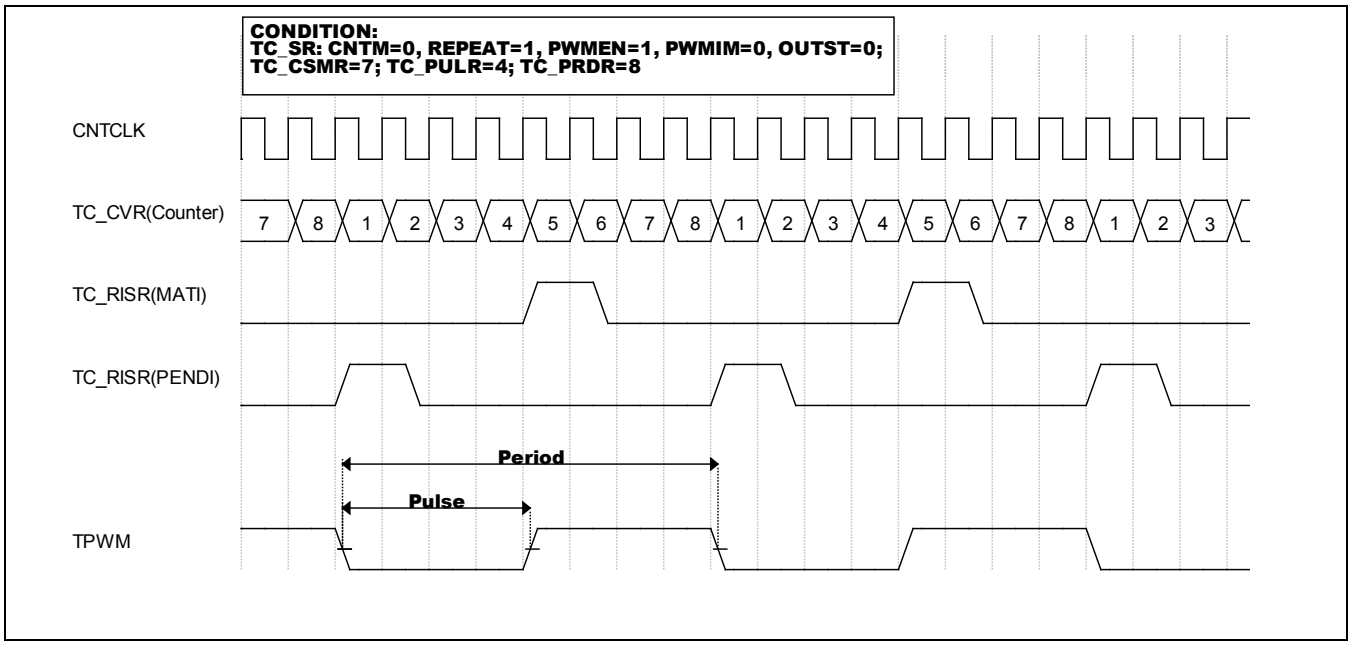


Figure 13-7 PWM时序

13.2.7.2.1 PWM扩展位

扩展计数器的值与扩展逻辑，PWMEX寄存器的值一起，用来“拉伸”PWM输出的周期。“拉伸”的值为1个时钟周期。

由于PWMEX是一个6位的寄存器，所以每64个PWM周期为一个扩展周期。扩展周期中，有一个或多个PWM输出会被“拉伸”一个时钟周期的长度，也就是比普通周期多一个时钟周期。扩展的周期位置由TC1_SR中的PWMEX位决定。由于PWMEX有6位，所以PWM输出能够扩展到约38位的分辨率(32位计数器+6位扩展)。

Table 13-2 PWM扩展位

PWMEX Bit	扩展周期(第x个周期拉伸1个周期)
0	32
1	16, 48
2	8, 24, 40, 56
3	4, 12, 20, 28, 36, 44, 52, 60
4	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
5	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63

下图演示了基于PWMEX的PWM输出信号。

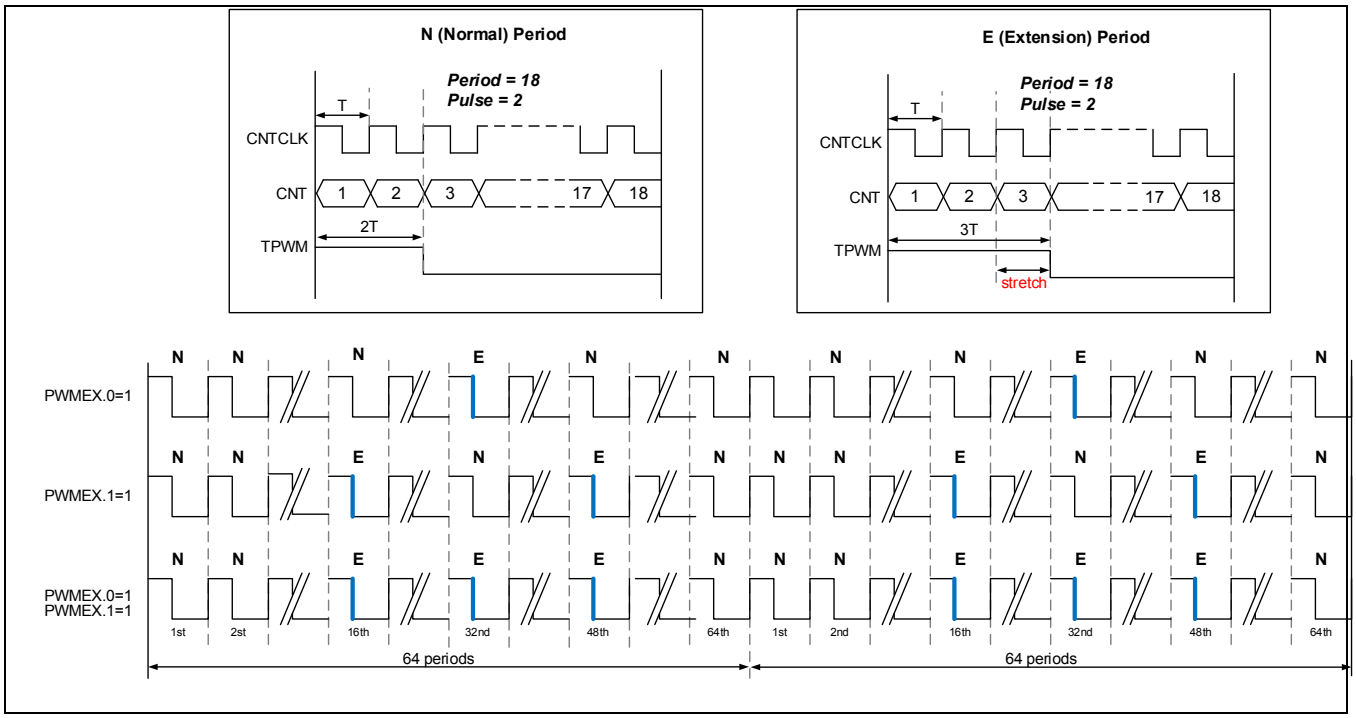


Figure 13-8 扩展PWM的波形

PWM 信号周期可以由下面的公式计算得出。

$$\text{Duty (\%)} = \left(\frac{(\text{PULSE} \times (64 - \text{E})) + (\text{PULSE} + 1) \times \text{E}}{\text{PERIOD} \times 64} \right) \text{X } 100 = \left(\frac{\text{PULSE}}{\text{PERIOD}} + \frac{\text{E}}{\text{PERIOD} \times 64} \right) \text{X } 100$$

，‘E’是64个周期中设置了扩展的周期数

例如，正常情况下，当PERIOD为100，PULSE为50的时候，PWM输出占空比位50%。如果PWMEX只有bit0为1，那么第32个和第64个脉冲周期的周期宽度为51个计数时钟周期。这个情况下，PWM输出的占空比为50.015625%，因为1/64是0.015625。如果只有PWMEX的bit5为1，那么每64个周期中的32个周期为扩展周期，这样占空比则为50.5%。

13.2.7.2.2 PWM波形

下图展示了基于PERIOD和PULSE的PWM波形。正常周期中，当PULSE为0的时候占空比为0%，当PULSE等于PERIOD的时候占空比100%。

PWM的输出电平由OUTST位决定。当OUTST为0的时候，PWM输出电平从低开始。可以看出，PWM扩展的电平为高还是低可以用OUTST位来控制。

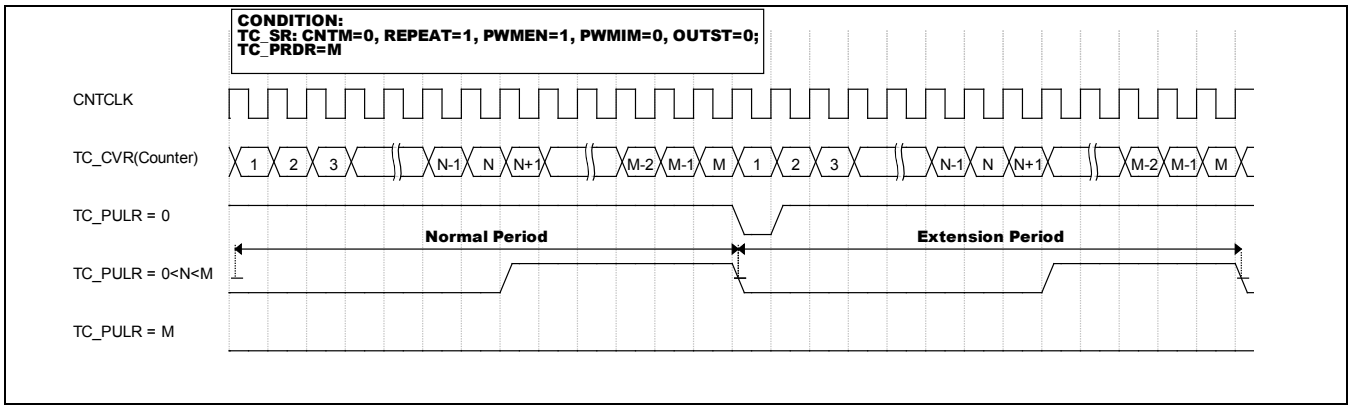


Figure 13-9 OUTST=0时PWM波形

如果OUTST为1，PWM输出信号从高电平开始。

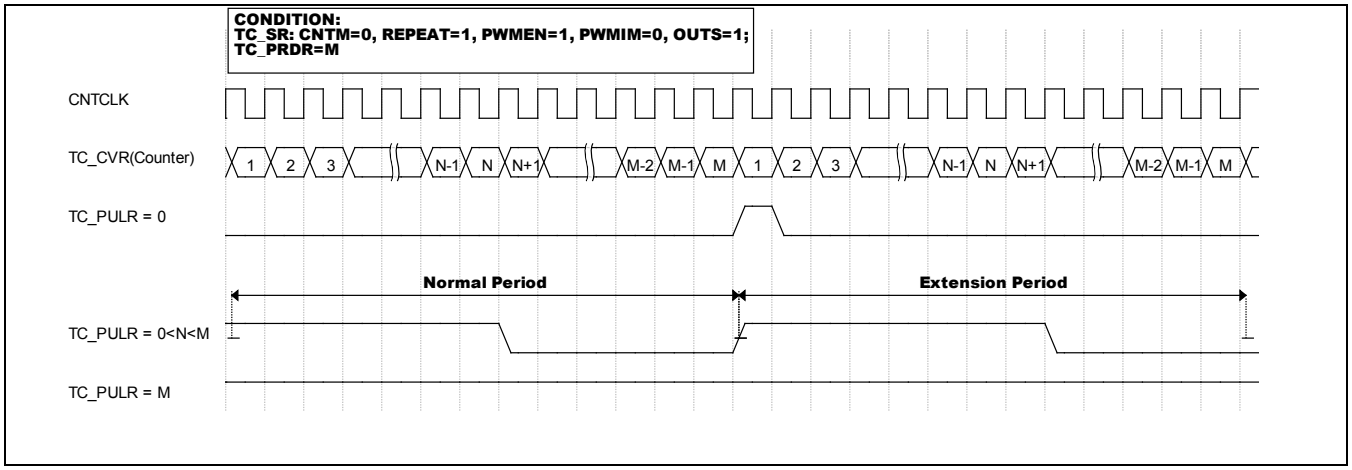


Figure 13-10 OUTST=1时PWM波形

13.2.7.2.3 PWM输出极性

当定时器停止时，PWM输出的状态由TC1_SR寄存器中的STOPCLEAR, STOPHOLD, KEEP, IDLEST和OUTST各个控制位决定。下表列出了各个控制位下输出极性的不同。

Table 13-3 PWM输出极性

STOP CLEAR	STOP HOLD	KEEP	IDLEST	OUTST	TPWM	说明
0	0	0	0	X	L	定时器在周期结束时停止，输出状态由IDLEST决定
0	0	0	1	X	H	
0	0	1	X	0	H	定时器在周期结束时停止，输出状态为OUTST的反
0	0	1	X	1	L	
0	1	X	X	X	L/H	定时器立即暂停计数，PWM输出保持为停止前的状态

STOP CLEAR	STOP HOLD	KEEP	IDLEST	OUTST	TPWM	说明
1	X	X	0	X	L	定时器立即停止计数，PWM输出由IDLEST决定
1	X	X	1	X	H	

注意：

控制位的优先权为：STOPCLEAR > STOPHOLD > KEEP > IDLEST.

如果在STOPHOLD和STOPCLEAR为0的情况下清除START位，那么定时器会在当前PERIOD周期结束后才停止。在这个状态下，如果TC1_SR中的KEEP位为1，那么TC保持PWM输出状态，也就是与OUTST相反的状态，如果KEEP位为0，那么PWM的输出状态则由IDLEST来决定。

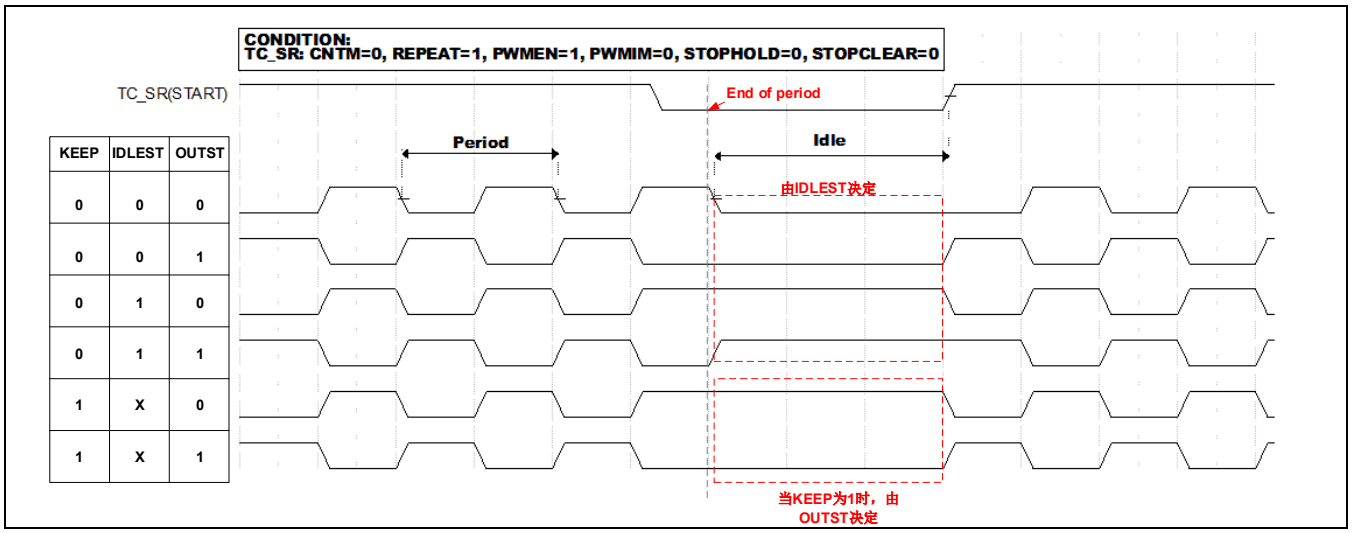


Figure 13-11 Idle状态下的PWM波形

如果STOPHOLD为1，但STOPCLEAR为0，当清除START的时候，定时器会停止并且计数器会保留停止前的值，PWM输出也保持停止前的状态。当再次给START置1的时候，定时器从之前保留的值重新开始计数。

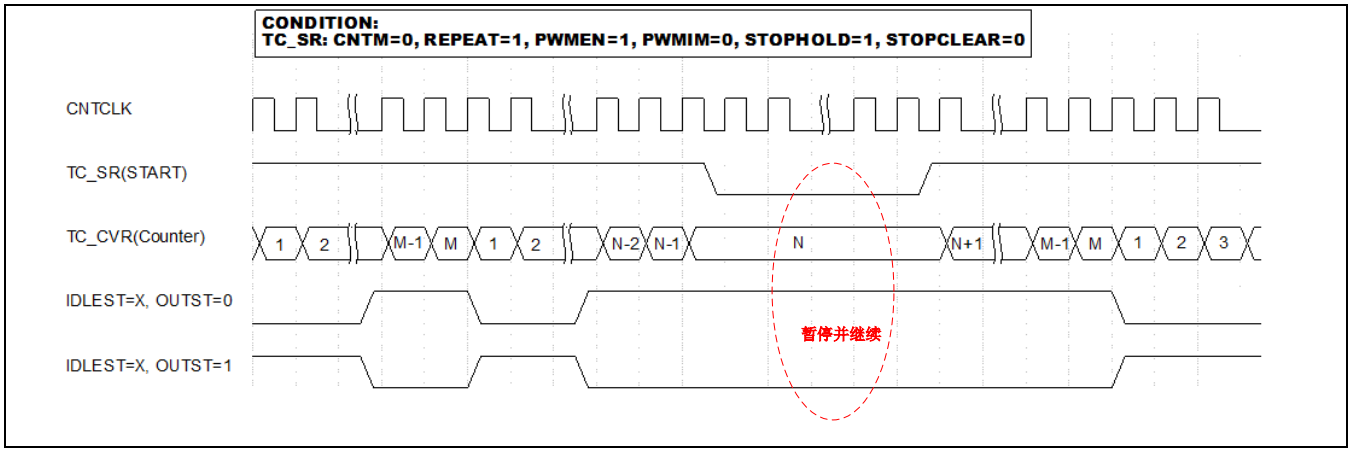


Figure 13-12 STOPHOLD=1, STOPCLEAR=0时PWM波形

如果STOPCLEAR为1，当清除START位时，定时器立即停止，并且输出电平由IDLEST决定。

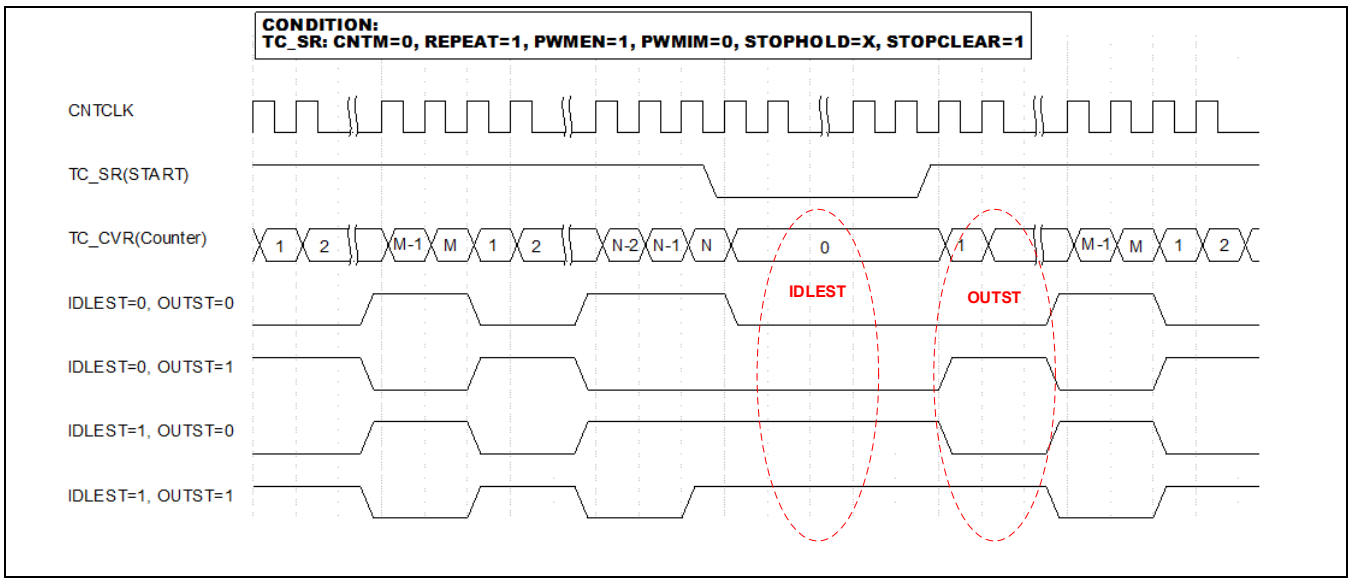


Figure 13-13 STOPCLEAR=1时PWM波形

定时器复位后，输出信号的初始值为低电平。当STOPCLEAR为1并且定时器不在工作的时候，通过改变IDLEST的值可以马上改变管脚的输出电平。

13.2.8 中断

定时器/计数器可以产生7种中断：STARTI, STOPI, PSTARTI, PENDI, MATI, OVFI 和 CAPTI。

13.2.8.1 计数启动中断

当定时器开始计数时，产生启动中断。

13.2.8.2 计数停止中断

当定时器停止计数时，产生停止中断。

13.2.8.3 周期开始中断

当计数周期开始时，产生周期开始中断。

13.2.8.4 周期结束中断

当计数周期结束时，产生周期结束中断。

13.2.8.5 脉冲匹配中断

当计数器的值等于脉冲寄存器TC1_PULSE设定的值时，产生脉冲匹配中断。

13.2.8.6 溢出中断

当定时器计数溢出时，产生溢出中断。

13.2.8.7 捕捉中断

捕捉模式下，当外部捕捉信号上升或下降沿触发时，产生捕捉中断。

13.2.8.8 中断处理

- 进入中断处理程序(ISR)并且调用C函数
- 读寄存器TC1_IMSR，确定中断源
- 写TC1_ICR清除相应中断的状态位
- 中断处理
- 退出中断服务程序

13.3 寄存器说明

13.3.1 寄存器表

- Base Address: 0x4005_1000

Table 13-4 寄存器表

Register	Offset	Description	Reset Value
TC1_IDR	0x000	ID寄存器	0x0011_000A
TC1_CSSR	0x004	时钟源选择寄存器	0x0000_0000
TC1_CEDR	0x008	时钟使能/禁止寄存器	0x0000_0000
TC1_SRR	0x00C	软件复位寄存器	0x0000_0000
TC1_CSR	0x010	控制使能寄存器	0x0000_0000
TC1_CCR	0x014	控制清除寄存器	0x0000_0000
TC1_SR	0x018	控制状态寄存器	0x0000_0002
TC1_IMSCR	0x01C	中断使能/禁止寄存器	0x0000_0000
TC1_RISR	0x020	原始中断状态寄存器	0x0000_0000
TC1_MISR	0x024	中断状态寄存器	0x0000_0000
TC1_ICR	0x028	中断状态清除寄存器	0x0000_0000
TC1_CDR	0x02C	时钟分频寄存器	0x0000_0000
TC1_CSMR	0x030	计数器位数控制寄存器	0x0000_001F
TC1_PRDR	0x034	周期值寄存器	0x0000_0000
TC1_PULR	0x038	脉冲比较值寄存器	0x0000_0000
TC1_CUCR	0x04C	捕捉上升沿计数寄存器	0x0000_0000
TC1_CDCR	0x050	捕捉下降沿计数寄存器	0x0000_0000
TC1_CVR	0x054	当前计数值寄存器	0x0000_0000

13.3.1.1 TC1_IDR (ID寄存器)

- Address = Base Address + 0x0000, Reset Value = 0x0011_000A

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

Name	Bit	Type	Description
IDCODE	[25:0]	R	ID Code寄存器 相应IP的ID code，无法更改

13.3.1.2 TC1_CSSR (时钟源选择寄存器)

- Address = Base Address + 0x0004, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
CLKSRC[3:0]	[3:0]	RW	时钟源选择寄存器 0x0: 计数器时钟源为PCLK 0x1: 计数器时钟源为外部TCLK管脚 0x2: 无效选择 0x3: 无效选择 0x4: 无效选择 0x5: 计数器时钟源为EMOSC 0x6: 计数器时钟源为ISOSC 0x7: 计数器时钟源为IMOSC 0x8: 计数器时钟源为比较器0的输出脉冲 0x9: 计数器时钟源为比较器1的输出脉冲 0xA: 计数器时钟源为比较器2的输出脉冲 0xB: 计数器时钟源为比较器3的输出脉冲 0xC: 计数器时钟源为比较器4的输出脉冲 0xD: 无效选择 0xE: 无效选择 0xF: 无效选择

注意：

- 定时器/计数器时钟使能后，用户不能修改CLKSRC位，所以在修改CLKSRC前，用户必须先禁止时钟。

13.3.1.3 TC1_CEDR (时钟使能/禁止寄存器)

- Address = Base Address + 0x0008, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
CLKEN	[0]	RW	时钟使能/禁止控制位 0: 禁止计数器时钟 1: 使能计数器时钟 SWRST不影响CLKEN位的状态
DBGEN	[31]	RW	Debug模式使能/禁止控制位 0: 禁止Debug模式 1: 使能Debug模式 如果DBGEN置1，当CPU在debug模式被暂停后，计数器计数将停止。

注意：

- 设置其它寄存器(除了CSSR)之前，必须要先将CLKEN置1。如果CLKEN为0，那么其它寄存器(除了CSSR)的写操作都无效。但读寄存器，写DBGEN和SWRST位跟CLKEN的状态无关。CSSR寄存器必须在CLKEN置1前设置。

13.3.1.4 TC1_SRR (软件复位寄存器)

- Address = Base Address + 0x000C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
SWRST	[0]	W	软件复位 0: 无效 1: 软件复位

13.3.1.5 TC1_CSR (控制使能寄存器)

- CSR: Address = Base Address + 0x0010, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		PWMEX5	PWMEX4	PWMEX3	PWMEX2	PWMEX1	PWMEX0	RSVD		CAPIN			CAPT_R	CAPT_F	HWTRIG_IN	HWTRIG_OUT	CNTM	REPEAT	PWMEN	PWMIM	KEEP	OUTST	IDLEST	RSVD				STOPCLEAR	STOPHOLD	UPDATE	START
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	W	W	W	W	W	W	R	R	R	R	R	W	W	W	W	W	W	W	W	W	W	W	R	R	R	R	W	W	W	W

Name	Bit	Type	Description
START	[0]	W	启动计数器
UPDATE	[1]	W	请求更新寄存器
STOPHOLD	[2]	W	停止计数后保留计数器值
STOPCLEAR	[3]	W	停止计数后清除计数器
IDLEST	[8]	W	IDLE时输出高电平
OUTST	[9]	W	输出状态为高电平
KEEP	[10]	W	停止后保持输出电平
PWMIM	[11]	W	使能输出翻转功能/禁止PWM输出功能
PWMEN	[12]	W	使能PWM输出功能
REPEAT	[13]	W	使能循环重复模式(溢出后重新开始计数)
CNTM	[14]	W	使能连续计数模式(计数到计数器的最大值)
HWTRIG_OUT	[15]	W	使能硬件触发其它模块功能
HWTRIG_IN	[16]	W	使能其它模块硬件触发定时器启动的功能
CAPT_F	[17]	W	当检测到外部输入信号的下降沿时，将当前计数值存入下降沿捕捉寄存器中。
CAPT_R	[18]	W	当检测到外部输入信号的上升沿时，将当前计数值存入上升沿捕捉寄存器中。
CAPIN	[21:19]	W	选择需要检测的外部信号 0x0: TCAP管脚输入 0x1: 无效选择 0x2: 无效选择 0x3: 比较器0的输出脉冲 0x4: 比较器1的输出脉冲 0x5: 比较器2的输出脉冲 0x6: 比较器3的输出脉冲 0x7: 比较器4的输出脉冲

PWMEX[5:0]	[29:24]	W	PWM扩展位

注意：

- 对该寄存器写0无效

13.3.1.6 TC1_CCR (控制清除寄存器)

- CCR: Address = Base Address + 0x0014, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		PWMEX5	PWMEX4	PWMEX3	PWMEX2	PWMEX1	PWMEX0	RSVD		CAPIN			CAPT_R	CAPT_F	HWTRIG_IN	HWTRIG_OUT	CNTM	REPEAT	PWMEN	PWMIM	KEEP	OUTST	IDLEST	RSVD				STOPCLEAR	STOPHOLD	UPDATE	START
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	W	W	W	W	W	W	R	R	R	R	R	W	W	W	W	W	W	W	W	W	W	W	R	R	R	R	W	W	W	W

Name	Bit	Type	Description
START	[0]	W	停止计数器
UPDATE	[1]	W	完成更新寄存器
STOPHOLD	[2]	W	停止计数后不保留计数器值
STOPCLEAR	[3]	W	停止计数后不清除计数器
IDLEST	[8]	W	IDLE时输出低电平
OUTST	[9]	W	输出状态为低电平
KEEP	[10]	W	停止后不保持输出电平
PWMIM	[11]	W	禁止输出翻转功能/ 使能PWM输出功能
PWMEN	[12]	W	禁止PWM输出功能
REPEAT	[13]	W	禁止循环重复模式
CNTM	[14]	W	禁止连续计数模式
HWTRIG_OUT	[15]	W	禁止硬件触发其它模块功能
HWTRIG_IN	[16]	W	禁止其它模块硬件触发定时器启动的功能
CAPT_F	[17]	W	禁止下降沿捕捉功能
CAPT_R	[18]	W	禁止上升沿捕捉功能
CAPIN	[21:19]	W	选择需要检测的外部信号 0x0: TCAP管脚输入 0x1: 无效选择 0x2: 无效选择 0x3: 比较器0的输出脉冲 0x4: 比较器1的输出脉冲 0x5: 比较器2的输出脉冲 0x6: 比较器3的输出脉冲 0x7: 比较器4的输出脉冲
PWMEX[5:0]	[29:24]	W	PWM扩展位

注意：

- 对该寄存器写0无效

13.3.1.7 TC1_SR (控制状态寄存器)

- CCR: Address = Base Address + 0x0018, Reset Value = 0x0000_0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		PWMEX5	PWMEX4	PWMEX3	PWMEX2	PWMEX1	PWMEX0	RSVD		CAPIN			CAPT_R	CAPT_F	HWTRIG_IN	HWTRIG_OUT	CNTM	REPEAT	PWMEN	PWMIM	KEEP	OUTST	IDLEST	RSVD				STOPCLEAR	STOPHOLD	UPDATE	START
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	W	W	W	W	W	W	R	R	R	R	R	W	W	W	W	W	W	W	W	W	W	W	R	R	R	R	W	W	W	W

Name	Bit	Type	Description
START	[0]	R	0: 计数器处于停止状态 1: 计数器处于工作状态
UPDATE	[1]	R	0: 寄存器更新被禁止 1: 允许更新寄存器
STOPHOLD	[2]	R	0: 停止后保留计数值模式被禁止 1: 停止后保留计数值模式被使能 当STOPHOLD置1时，计数器停止后会保留当前计数值和当前输出状态，所以当计数器再次启动后，计数会从保留的状态继续开始
STOPCLEAR	[3]	R	0: 停止后清除计数值模式被禁止 1: 停止后清除计数值模式被使能 当STOPCLEAR置1时，计数器停止后会清除计数值，输出电平由IDLEST决定
IDLEST	[8]	R	0: Idle状态下输出电平为低 1: Idle状态下输出电平为高
OUTST	[9]	R	0: 计数开始的时候输出电平为低 1: 计数开始的时候输出电平为高
KEEP	[10]	R	0: 保持状态模式被禁止 1: 保持状态模式被使能 不管IDLEST为0还是1，计数停止后都保持输出电平的状态
PWMIM	[11]	R	0: PWM输出 1: 翻转输出
PWMEN	[12]	R	0: 禁止PWM输出 1: 使能PWM输出
REPEAT	[13]	R	0: 禁止循环重复模式 1: 使能循环重复模式 使能该模式后，计数器在连续计数模式溢出或者周期模式周期结束后，会自动重新开始计数

<table>
<tr><td>CNTM</td><td>[14]</td><td>R</td><td>0: 工作在周期模式，计数值自增直到周期结束
1: 工作在连续计数模式，计数值自增直到溢出</td></tr>
<tr><td>HWTRIG_OUT</td><td>[15]</td><td>R</td><td>0: 硬件触发其它模块功能被禁止
1: 硬件触发其它模块功能被使能</td></tr>
<tr><td>HWTRIG_IN</td><td>[16]</td><td>R</td><td>0: 其它模块硬件触发定时器启动的功能被禁止
1: 其它模块硬件触发定时器启动的功能被使能</td></tr>
<tr><td>CAPT_F</td><td>[17]</td><td>R</td><td>下降沿触发捕捉
0: 外部输入信号下降沿捕捉被禁止
1: 外部输入信号下降沿捕捉被使能
当检测到外部输入信号的下降沿时，将当前计数值存入下降沿捕捉寄存器中。</td></tr>
<tr><td>CAPT_R</td><td>[18]</td><td>R</td><td>上升沿触发捕捉
0: 外部输入信号上升沿捕捉被禁止
1: 外部输入信号上升沿捕捉被使能
当检测到外部输入信号的上升沿时，将当前计数值存入上升沿捕捉寄存器中。</td></tr>
<tr><td>CAPIN</td><td>[21:19]</td><td>W</td><td>需要检测的外部信号选择
0x0: TCAP管脚输入
0x1: 无效选择
0x2: 无效选择
0x3: 比较器0的输出脉冲
0x4: 比较器1的输出脉冲
0x5: 比较器2的输出脉冲
0x6: 比较器3的输出脉冲
0x7: 比较器4的输出脉冲</td></tr>
<tr><td>PWMEX[5:0]</td><td>[29:24]</td><td>R</td><td>PWM扩展位
<table>
<tr><td>PWMEX</td><td>“拉伸”的周期数</td></tr>
<tr><td>PWMEX0</td><td>32</td></tr>
<tr><td>PWMEX1</td><td>16, 48</td></tr>
<tr><td>PWMEX2</td><td>8, 24, 40, 56</td></tr>
<tr><td>PWMEX3</td><td>4, 12, 20, 28, 36 , 44, 52, 60</td></tr>
<tr><td>PWMEX4</td><td>2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62</td></tr>
<tr><td>PWMEX5</td><td>1, 3, 5, 7, 9,11,13,15,17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63</td></tr>
</table></td></tr>
</table>

13.3.1.8 TC1_IMSCR (中断使能/禁止寄存器)

- Address = Base Address + 0x001C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																									CAPTI	OVFI	MATI	PENDI	PSTARTI	STOPI	STARTI
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W

Name	Bit	Type	Description
STARTI	[0]	RW	计数启动中断 0: 禁止该中断 1: 使能该中断
STOPI	[1]	RW	计数停止中断 0: 禁止该中断 1: 使能该中断
PSTARTI	[2]	RW	周期开始中断 0: 禁止该中断 1: 使能该中断
PENDI	[3]	RW	周期结束中断 0: 禁止该中断 1: 使能该中断
MATI	[4]	RW	脉冲匹配中断 0: 禁止该中断 1: 使能该中断
OVFI	[5]	RW	溢出中断 0: 禁止该中断 1: 使能该中断
CAPTI	[6]	RW	捕捉中断 0: 禁止该中断 1: 使能该中断

13.3.1.9 TC1_ RISR (原始中断状态寄存器)

- Address = Base Address + 0x0020, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																									CAPTI	OVFI	MATI	PENDI	PSTARTI	STOPI	STARTI
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
STARTI	[0]	R	计数启动中断的原始状态(即使该中断没有使能，也会置位)
STOPI	[1]	R	计数停止中断的原始状态(即使该中断没有使能，也会置位)
PSTARTI	[2]	R	周期开始中断的原始状态(即使该中断没有使能，也会置位)
PENDI	[3]	R	周期结束中断的原始状态(即使该中断没有使能，也会置位)
MATI	[4]	R	脉冲匹配中断的原始状态(即使该中断没有使能，也会置位)
OVFI	[5]	R	溢出中断的原始状态(即使该中断没有使能，也会置位)
CAPTI	[6]	R	捕捉中断的原始状态(即使该中断没有使能，也会置位)

注意：

- 0: 中断没有发生; 1: 中断已发生

13.3.1.10 TC1_ MISR (中断状态寄存器)

- Address = Base Address + 0x0024, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																									CAPTI	OVFI	MATI	PENDI	PSTARTI	STOPI	STARTI
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
STARTI	[0]	R	计数启动中断的状态(该中断使能后才会置位)
STOPI	[1]	R	计数停止中断的状态(该中断使能后才会置位)
PSTARTI	[2]	R	周期开始中断的状态(该中断使能后才会置位)
PENDI	[3]	R	周期结束中断的状态(该中断使能后才会置位)
MATI	[4]	R	脉冲匹配中断的状态(该中断使能后才会置位)
OVFI	[5]	R	溢出中断的状态(该中断使能后才会置位)
CAPTI	[6]	R	捕捉中断的状态(该中断使能后才会置位)

注意：

- TC1_MISR = TC1_IMSCR & TC1_RISR
- 读该寄存器返回当前相应中断的状态，写操作无效。
- 0: 中断没有发生; 1: 中断已发生

13.3.1.11 TC1_ ICR (中断状态清除寄存器)

- Address = Base Address + 0x0028, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																									CAPTI	OVFI	MATI	PENDI	PSTARTI	STOPI	STARTI
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W

Name	Bit	Type	Description
STARTI	[0]	W	0: 无效 1: 清除计数启动中断
STOPI	[1]	W	0: 无效 1: 清除计数停止中断
PSTARTI	[2]	W	0: 无效 1: 清除周期开始中断
PENDI	[3]	W	0: 无效 1: 清除周期结束中断
MATI	[4]	W	0: 无效 1: 清除脉冲匹配中断
OVFI	[5]	W	0: 无效 1: 清除溢出中断
CAPTI	[6]	W	0: 无效 1: 清除捕捉中断

13.3.1.12 TC1_CDR (时钟分频寄存器)

- Address = Base Address + 0x002C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
DIVN	[3:0]	RW	时钟分频器DIVN的值
DIVM	[11:4]	RW	时钟分频器DIVM的值

注意：

- 当DIVN不等于0的时候，禁止将DIVM设成0

13.3.1.13 TC1_CSMR (计数器位数控制寄存器)

- Address = Base Address + 0x0030, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
SIZE	[4:0]	RW	设置计数器位数，例如： 如果SIZE为0x07，那么定时器/计数器是一个8位的定时器/计数器 如果SIZE为0x09，那么定时器/计数器是一个10位的定时器/计数器 如果SIZE为0x0F，那么定时器/计数器是一个16位的定时器/计数器 如果SIZE为0x1F，那么定时器/计数器是一个32位的定时器/计数器

13.3.1.14 TC1_PRDR (周期值寄存器)

- Address = Base Address + 0x0034, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
PERIOD	[31:0]	RW	PERIOD的值

注意：由于该计数器在周期匹配或者溢出后，都是从1开始计数的，所以该寄存器的值不能设置为0，否则计数器无法正常工作。

13.3.1.15 TC1_PULR (脉冲比较值寄存器)

- Address = Base Address + 0x0038, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
PULSE	[31:0]	RW	PULSE的值

注意：由于该计数器在周期匹配或者溢出后，都是从1开始计数的，所以该寄存器的值不能设置为0，否则计数器无法正常工作。

13.3.1.16 TC1_CUCR (捕捉上升沿计数寄存器)

- Address = Base Address + 0x004C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
COUNT	[31:0]	R	检测到上升沿时的计数值

13.3.1.17 TC1_CDCR (捕捉下降沿计数寄存器)

- Address = Base Address + 0x0050, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
COUNT	[31:0]	R	检测到下降沿时的计数值

13.3.1.18 TC1_CVR (当前计数值寄存器)

- Address = Base Address + 0x0054, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
COUNT	[31:0]	R	当前计数值

注意：

当时钟源是一个异步时钟时，无法保证该寄存器的值为当前完全正确的值(由于同步逻辑的存在，可能有一定的误差

14 简易定时器 (TC2)

14.1 概述

TC2是一个简易定时器，定时器(以下称为STC)有2个功能通道，每个通道可以分别工作在两种模式，定时匹配和输入信号捕捉模式。

14.1.1 主要功能

- 可编程16位递增计数器
- 支持单次和重复计数
- 支持定时匹配功能(MATCH)
- 支持输入捕捉功能 (CAP)
  - 上升沿捕捉，下降沿捕捉和上升下降沿同时捕捉
- 支持外部触发源
  - EPWM输出信号可以启动计数

14.1.2 管脚描述

Table 14-1 TC2 管脚描述

管脚名称	功能	I/O类型	说明
CAP[1:0]	输入捕捉管脚	I	–

14.2 功能描述

14.2.1 模块框图

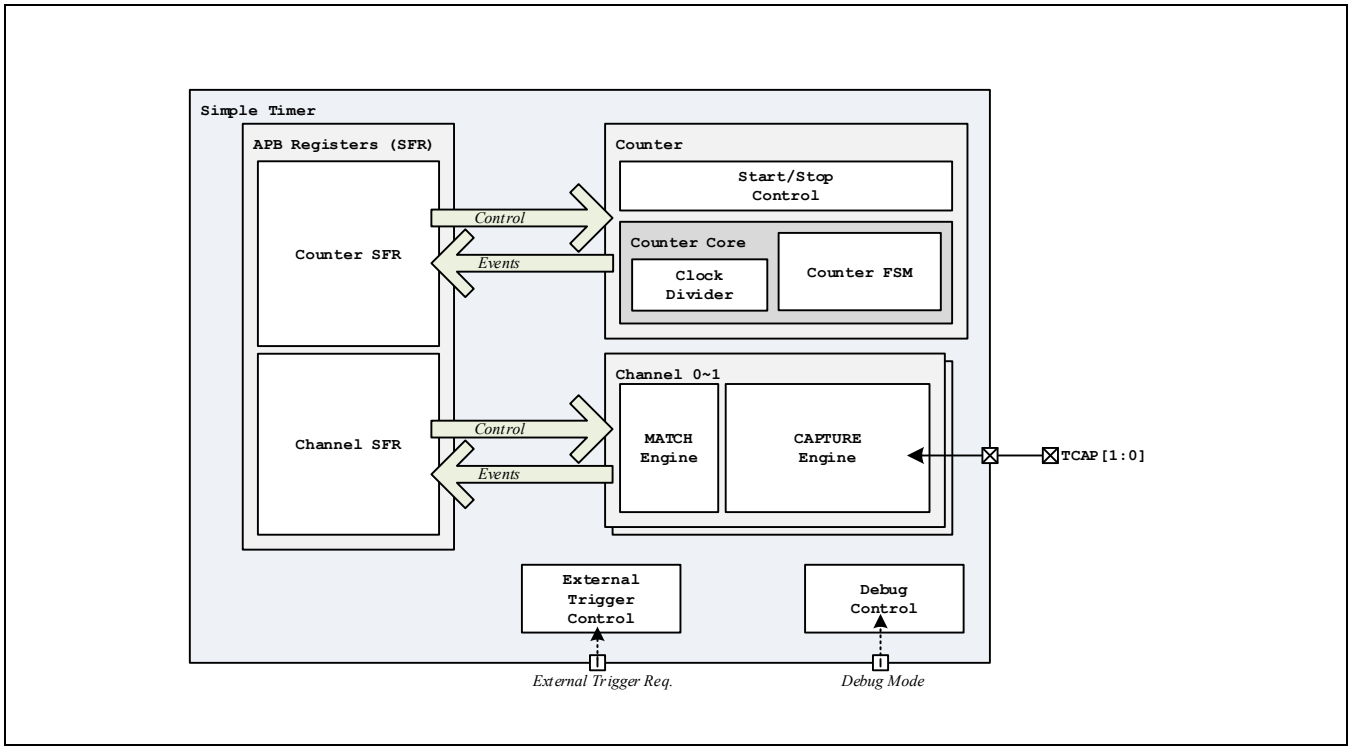


Figure 14-1 TC2模块框图

14.2.2 时钟控制

14.2.2.1 时钟控制模块框图

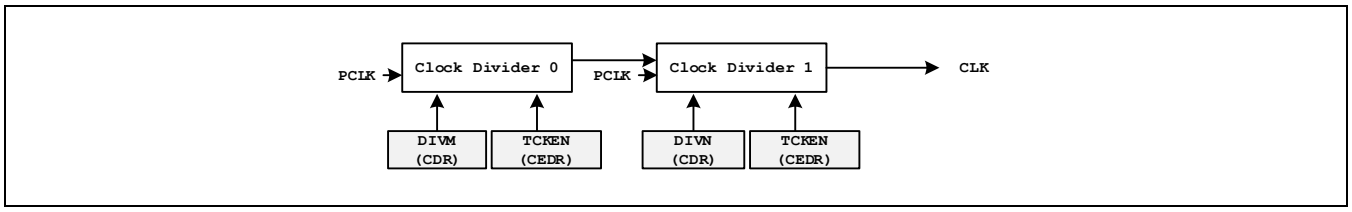


Figure 14-2 时钟控制模块框图

14.2.2.2 工作模式

时钟源

STC的时钟为系统的PCLK。

计数时钟分频

计数的快慢由时钟分频器(STC_CDR寄存器里的DIVM[10:0]和DIVN[3:0])决定。计数器的时钟TCCLK，由PCLK和STC_CDR共同决定。在计数器处于工作状态时，STC_CDR的值不允许改变。计数器的工作状态可以通过STC_SR寄存器中的BUSY位查询。

- TCCLK = PCLK / (DIVM + 1) / (2DIVN)

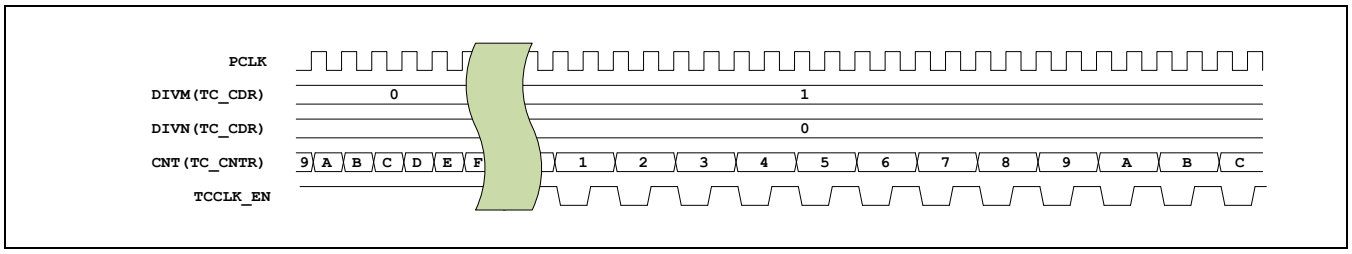
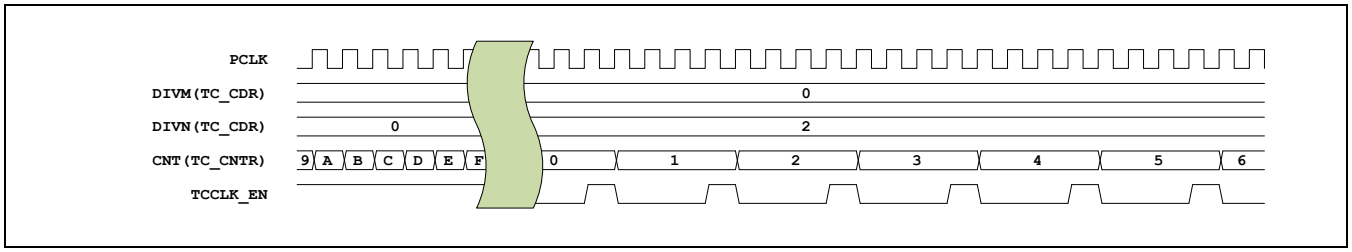


Figure 14-3 2分频的计数器时序图



14.2.3 Figure 14-4 4分频的计数器时序图

14.2.4 计数器模式控制

14.2.4.1 工作模式

计数器只有1种计数模式：**递增计数模式**

当定时器开始工作时，计数器从0开始计数到预设的值。当TCCLK_EN的值为1时，计数值在每个PCLK上升沿加1.

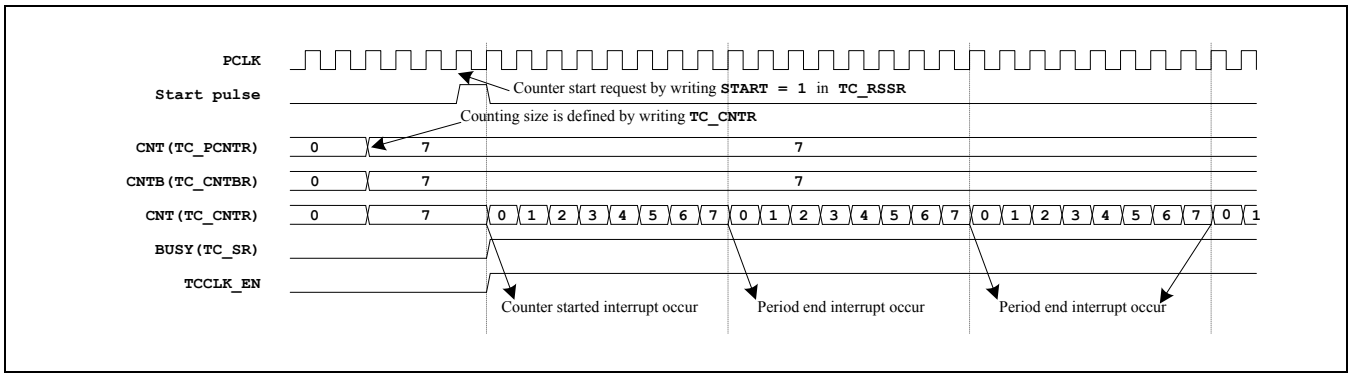


Figure 14-5 递增计数时序图，PCLK 1分频

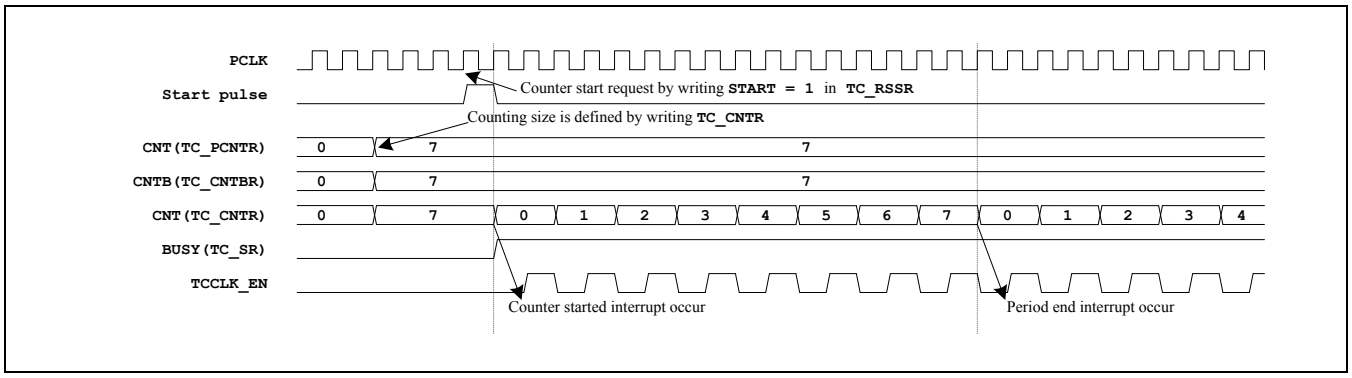


Figure 14-6 递增计数时序图，PCLK 2分频

14.2.5 自动重载

14.2.5.1 自动重载框图

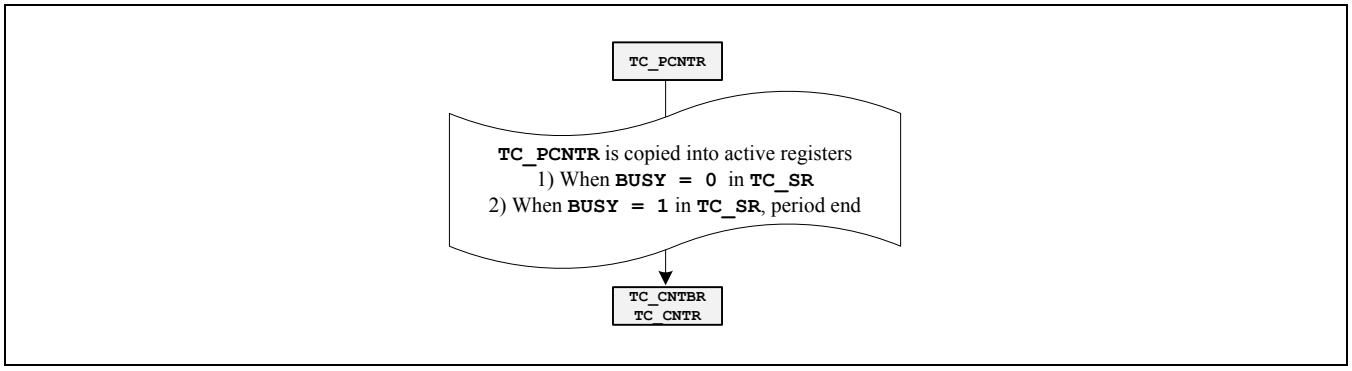


Figure 14-7 自动重载功能的发生条件

14.2.5.2 功能描述

当STC_SR中BUSY位等于1时，计数器工作状态可以周期性的更新。每当计数周期结束，计数器就会自动重载。

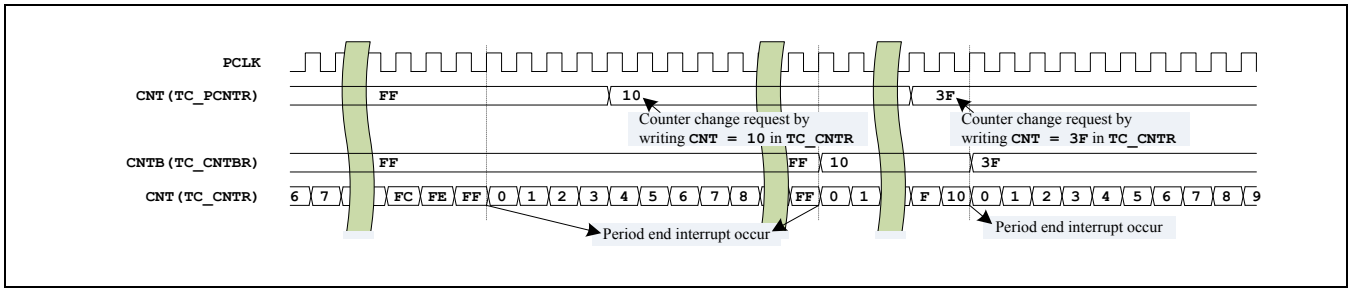


Figure 14-8 自动重载例子 (STC_CNTR)

Table 14-2 被自动重载功能影响的寄存器

Register	Comments
STC_CNTR	计数值寄存器

当STC_SR中BUSY等于0时，任何改变上面寄存器值的操作都会立即生效。

.

14.2.6 计数器启动控制

14.2.6.1 模块框图

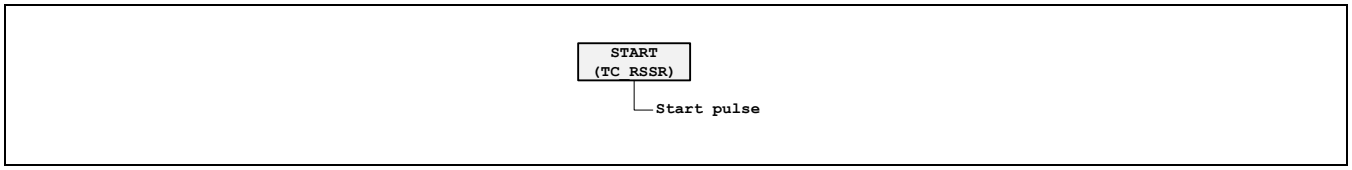


Figure 14-9 计数器启动控制

14.2.6.2 功能描述

通过将STC_RSSR寄存器中的START位置1，可以启动计数器。如果STC_SR中BUSY为1，那么对START位写1的启动请求会被忽略，因为计数器已经在工作中了。如果STC_CNTBR中的CNTB等于0，那么启动请求也会被忽略。

另外，STC还支持外部触发启动，参见外部触发章节。

14.2.7 计数器停止控制

14.2.7.1 模块框图

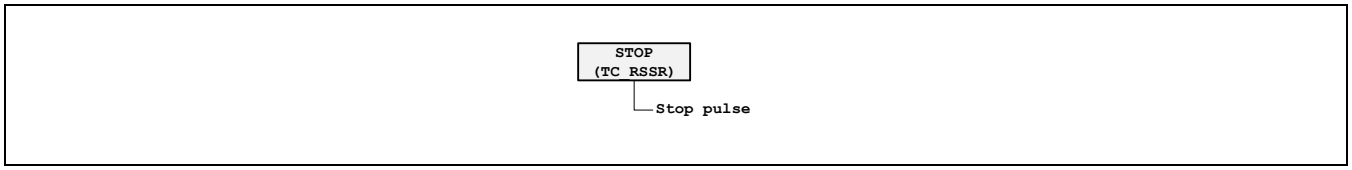


Figure 14-10 计数器停止控制

14.2.7.2 功能描述

通过将STC_RSSR寄存器中的STOP位置1，可以停止计数器。

通过设置STC_MR寄存器中的STOPTYPE位，计数器支持2种停止方式。

Table 14-3 计数器停止方式配置

STOPTYPE (STC_MR)	Stop Name	Operation
0	在周期结束时停止	停止请求跟周期结束事件同步
1	立即停止	停止请求立即生效，不管该计数周期是否结束。但是当STC_MR寄存器的SINGLE位是1，并且没有停止请求时，周期结束后会自动停止。

周期结束的停止

当STC_MR寄存器中的STOPTYPE是0时，只要收到任何计数器停止的请求，计数器都会在周期结束时自动停止。

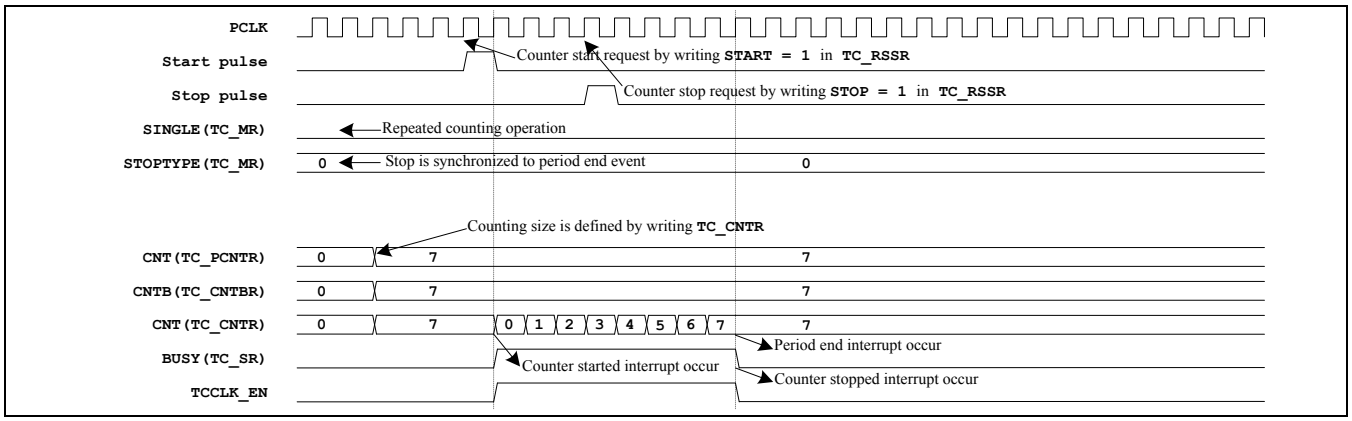


Figure 14-11 周期结束的停止时序图，例1 (重复计数，有停止请求)

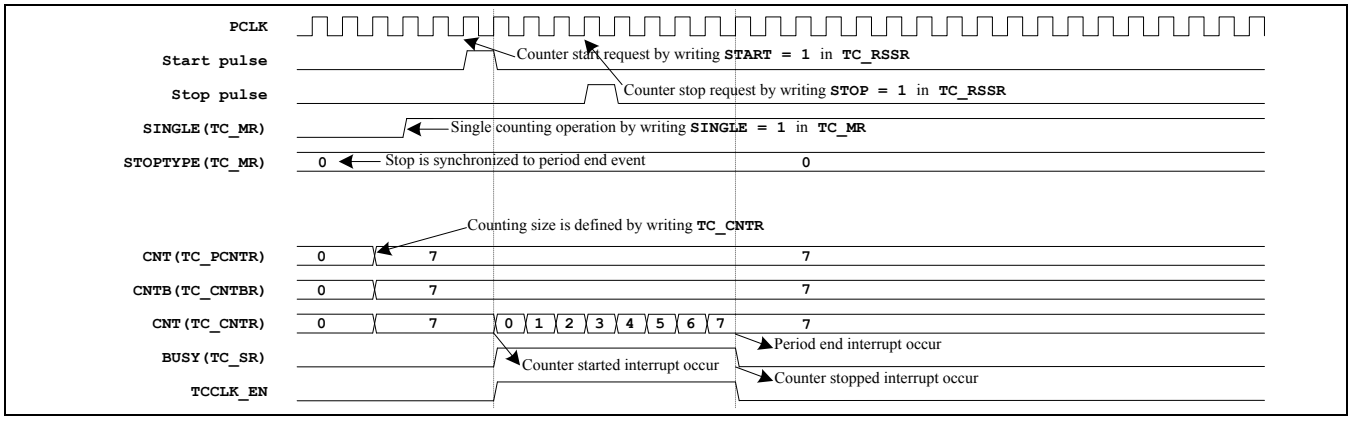


Figure 14-12 周期结束的停止时序图，例2 (单次计数，有停止请求)

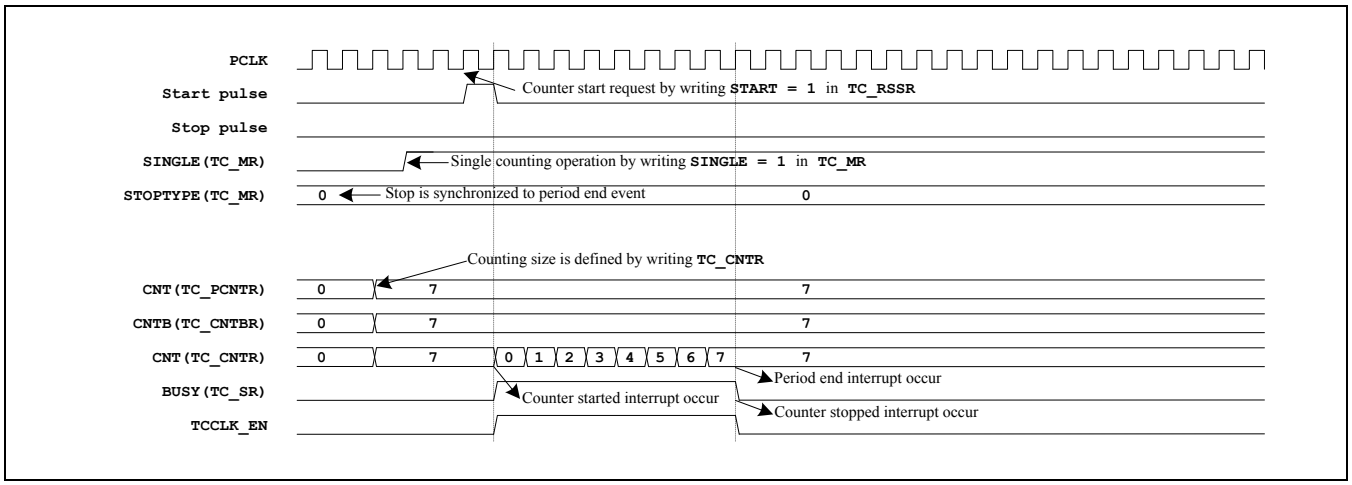


Figure 14-13 周期结束的停止时序图，例3 (单次计数)

立即停止

当STC_MR寄存器中的STOPTYPE是1时，只要收到任何计数器停止的请求，计数器都会立即停止。但是当STC_MR寄存器的SINGLE位是1，并且没有停止请求时，计数器会在周期结束后自动停止。

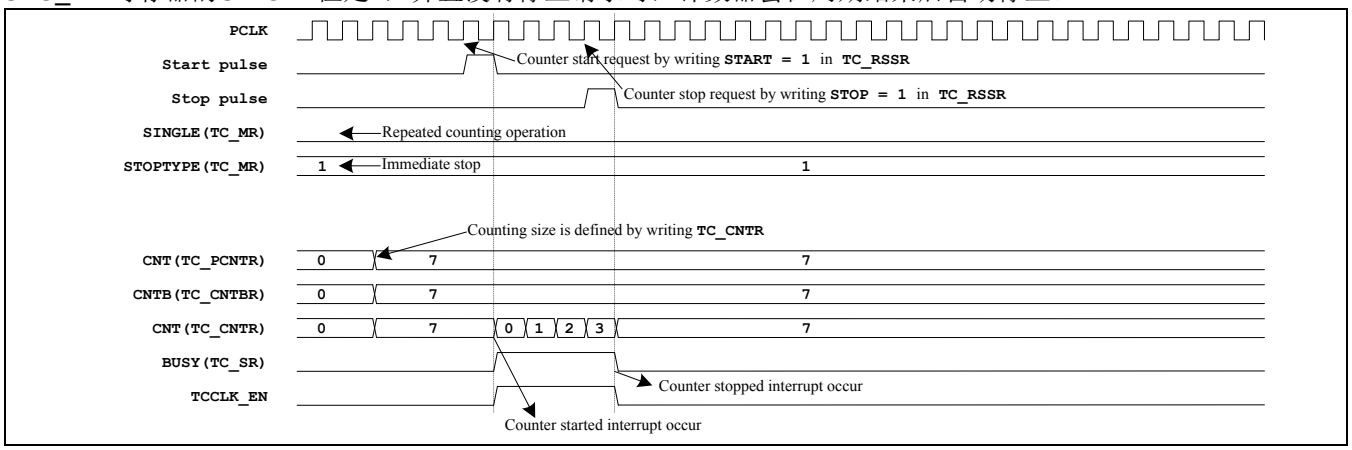


Figure 14-14 立即停止的时序图，例1 (重复计数，有停止请求)

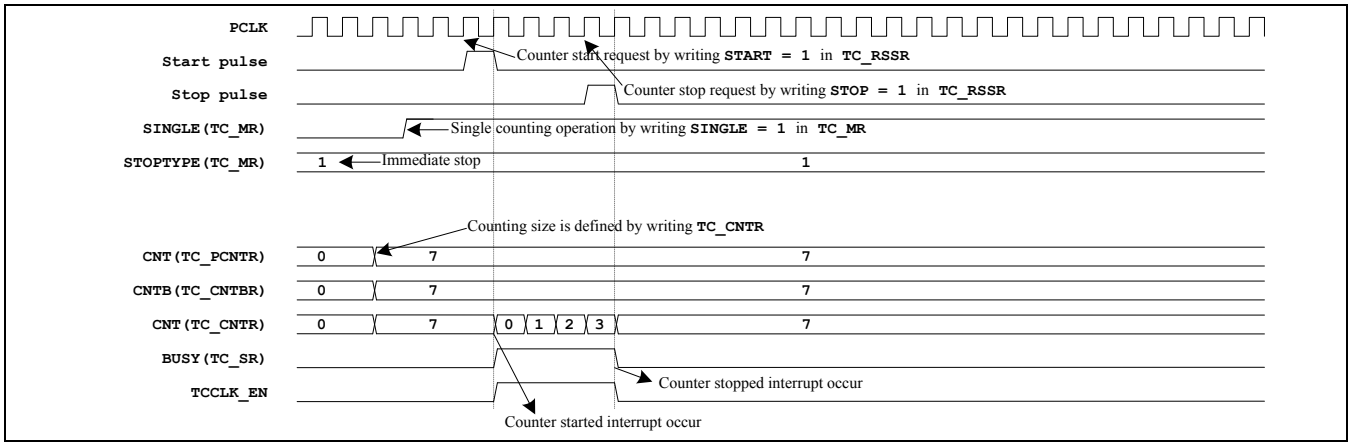


Figure 14-15 立即停止的时序图，例2 (单次计数，有停止请求)

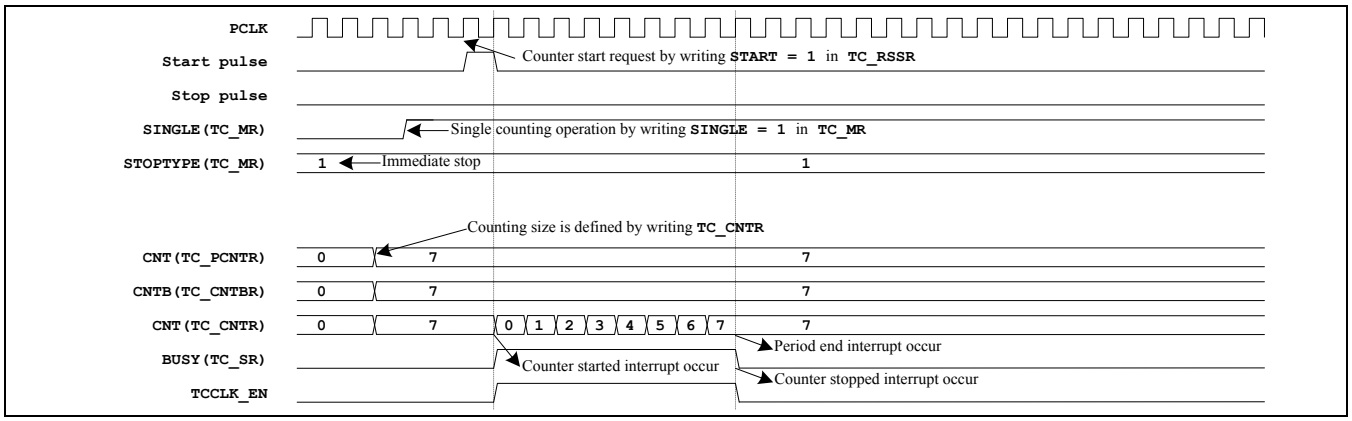


Figure 14-16 立即停止的时序图，例3 (单次计数)

14.2.8 计数通道的工作

14.2.8.1 模块框图

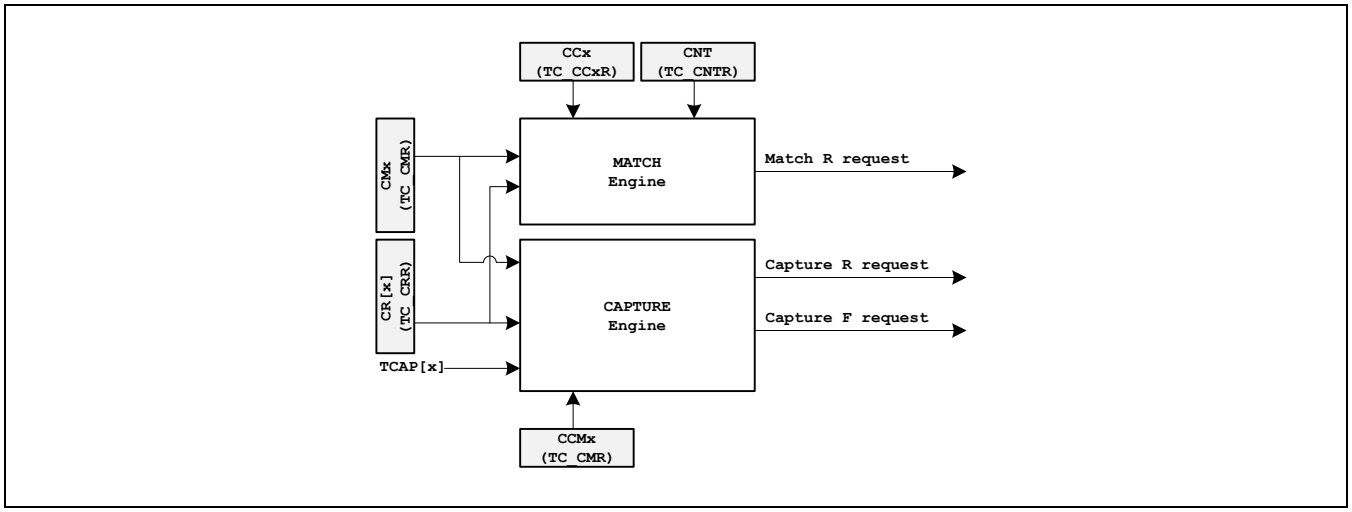


Figure 14-17 计数通道模块框图

14.2.8.2 功能描述

每个通道都有自己的匹配和捕捉引擎，可以通过STC_CMR寄存器里的CM位选择两种工作模式。所有工作模式只有当STC_CRR寄存器的CR[x]=1时才有效。

Table 14-4 计数通道工作模式

CM(STC_CMR)	工作模式
0	匹配模式
1	捕捉模式

匹配模式

- 当STC_SR中BUSY=1，STC_CRR中CR[x]=1时，只要STC_CNTR和STC_CCxR的值相等时，相应的R-type匹配中断就会产生。
- STC_CCxR代表STC_CC0R和STC_CC1R。如果STC_CCxR的值为0或者大于STC_CNTBR寄存器，匹配中断则不会产生。
- 匹配模式的要求
  - 0 < STC_CCxR ≤ STC_CNTBR

当某个通道被设置成匹配模式时，只要软件写入了STC_CCxR的值，相应的STC_CCxR寄存器就会被立即更新。在匹配模式，STC_PCCxR没有被使用。

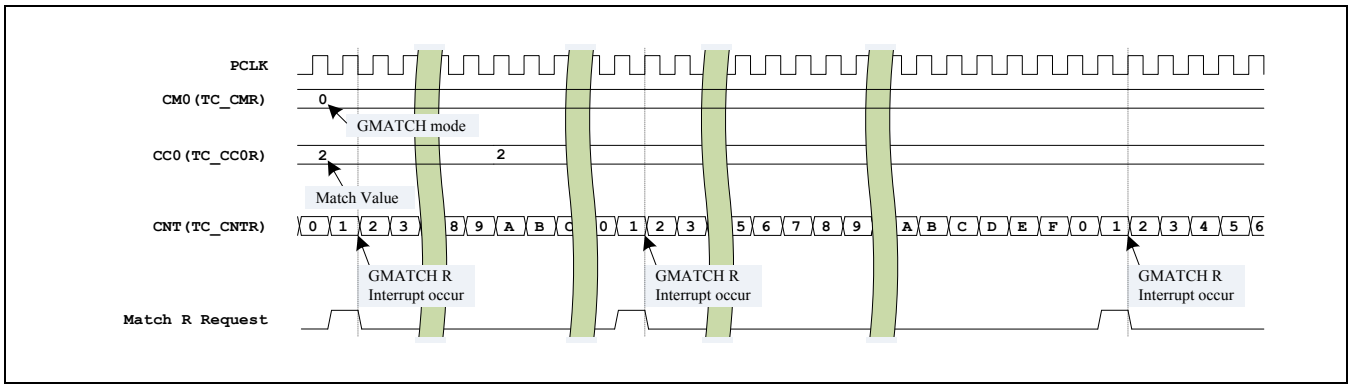


Figure 14-18 通道0的匹配模式时序图例

捕捉模式

当STC_SR中BUSY=1时，STCAP[x]输入的变化会将STC_CNTR中CNT的值复制到STC_CCxR寄存器。当某个通道被配置成捕捉模式时，软件也可以修改STC_CCxR的值。将STC_CRR中CR[x]写1会触发内部捕捉请求，这时CNT的值也会被复制到STC_CCxR。当改变STC_CMR中的CMx的值将x通道设置成捕捉模式时，系统需要4个PCLK的STCAP[x]处理时间，在这个时间内，系统不会响应内部捕捉请求。

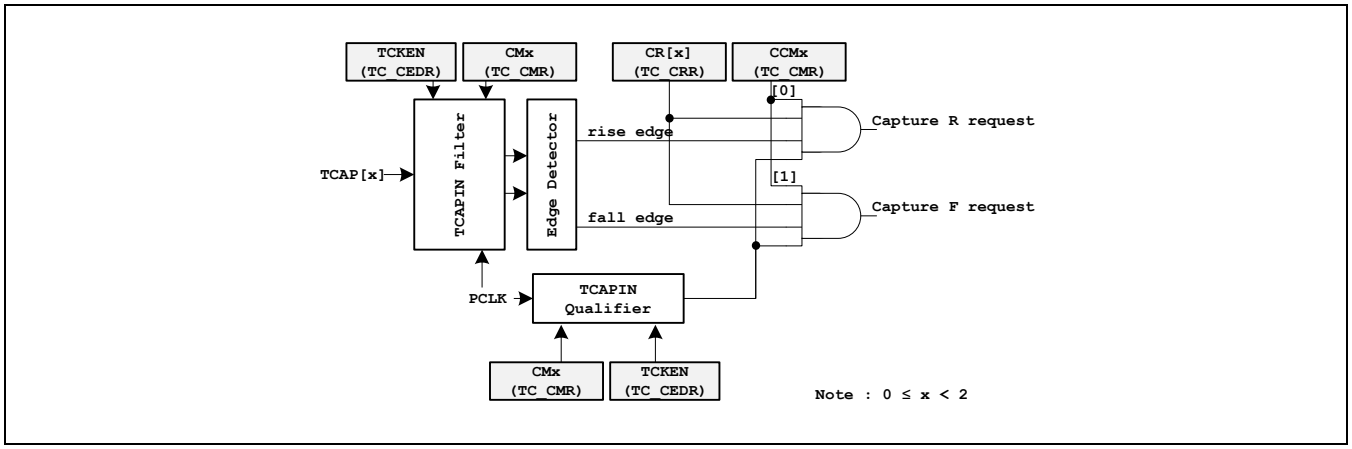


Figure 14-19 捕捉请求产生的模块框图

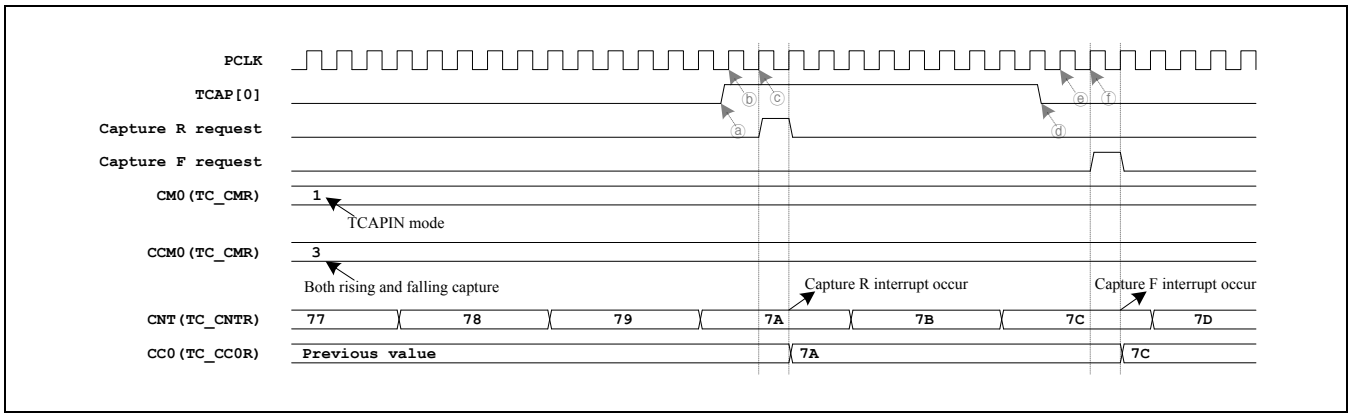


Figure 14-20 通道0的捕捉时序图例

14.2.9 中断

STC 中断根据中断源的不同，分为两类，一个是计数器中断，另一个是计数通道中断。

每当计数器中断发生时，中断状态被存储在 STC_SR 寄存器中。通过 STC_ICR 寄存器，可以清除中断。

每当计数通道中断发生时，中断状态被存储在 STC_CSR 寄存器中。通过 STC_CICR 寄存器，可以清除中断。

对 STC_SR 和 STC_CSR 的相应位写 0 也同样可以清除中断状态。

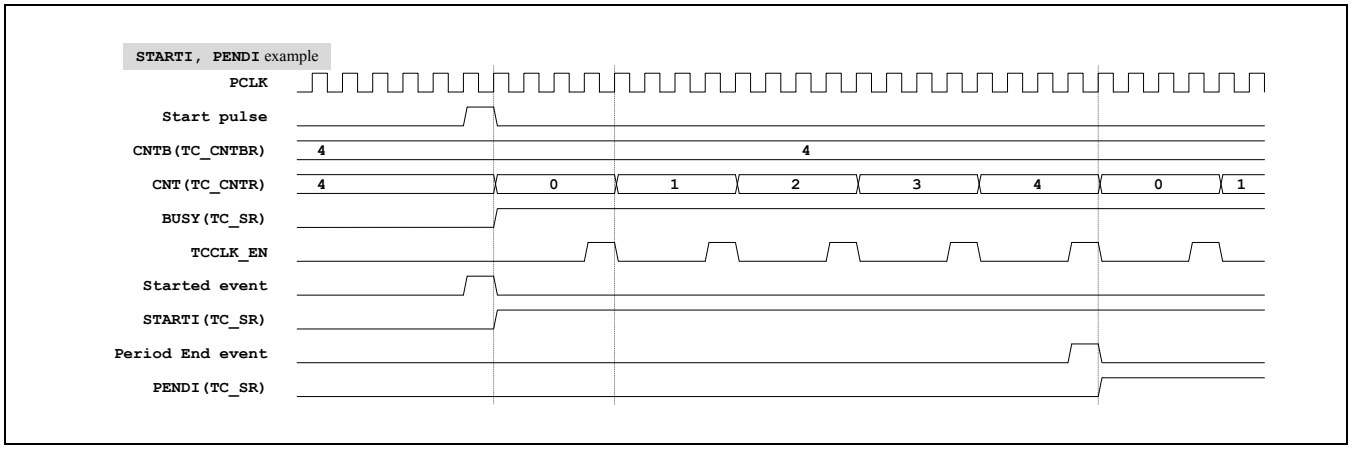


Figure 14-21 计数器中断时序图例

上面的时序图不包含STOPI的中断。对于STOPI，请参考“计数器停止控制”章节。对于计数通道的R和F中断，请参考“计数通道的工作”章节。

14.2.10 外部触发

从EPWM模块引入的外部触发源可以启动计数，跟STC_RSSR中START位写1有完全相同的效果。

14.3 寄存器说明

14.3.1 寄存器表 (基地址: 0x4005_2000)

Register	Offset	Description	Reset Value
STC_IDR	0x0000	IP版本代码	0x0041_000A
STC_CEDR	0x0004	时钟使能/禁止	0x0000_0000
STC_RSSR	0x0008	软件复位/启动/停止	0x0000_0000
STC_IMSCR	0x000C	中断使能/禁止	0x0000_0000
STC_RISR	0x0010	中断原始状态	0x0000_0000
STC_MISR	0x0014	中断状态	0x0000_0000
STC_ICR	0x0018	中断状态清除	0x0000_0000
STC_SR	0x001C	计数状态	0x0000_0000
	-	保留	-
STC_MR	0x0024	计数器模式控制	0x0000_0000
	-	保留	-
STC_CNTBR	0x0030	计数基值	0x0000_0000
STC_CNTR	0x0034	计数值	0x0000_0000
STC_CDR	0x0038	时钟分频控制	0x0000_0000
	-	保留	-
STC_PCNTR	0x0050	暂存计数值	0x0000_0000
	-	保留	-
STC_CRR	0x0080	通道启动控制	0x0000_0000
STC_CMR	0x0084	通道模式控制	0x0000_0000
STC_CIMSCR	0x0088	通道中断使能/禁止	0x0000_0000
STC_CRISR	0x008C	通道中断原始状态	0x0000_0000
STC_CMISR	0x0090	通道中断状态	0x0000_0000
STC_CICR	0x0094	通道中断状态清除	0x0000_0000
	-	保留	-
STC_CAPSR	0x009C	上一次捕捉状态	0x0000_0000
	-	保留	-
STC_CC0R	0x00C0	通道0捕捉/比较	0x0000_0000
STC_CC1R	0x00C4	通道1捕捉/比较	0x0000_0000

14.3.1.1 STC_IDR (版本代码寄存器)

- Address = Base Address + 0x0000, Reset Value = 0x0041_000A

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

Name	Bit	Type	Description	Reset Value
IDCODE	[23:0]	R	**版本代码寄存器** 存储该IP的版本代码	0x41_000A

14.3.1.2 STC_CEDR (时钟使能/禁止寄存器)

- Address = Base Address + 0x0004, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
TCKEN	[0]	R	时钟使能 0: 禁止时钟 1: 使能时钟	0

14.3.1.3 STC_RSSR (软件复位/启动/停止寄存器)

- Address = Base Address + 0x0008, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
SRR	[31]	W	软件复位 写1会对模块进行软件复位	0
STOP	[1]	W	软件停止 写1产生定时器/计数器的停止请求	0
START	[0]	W	软件启动 写1产生定时器/计数器的启动请求	0

注意： 上面 3 位有优先级： SRR > STOP > START.

注意： 软件复位需要1个PCLK时钟，所以如果在软件复位后马上读取寄存器的值，读回的值会是0x0000_0000.

14.3.1.4 STC_IMSCR (计数器中断控制寄存器)

- Address = Base Address + 0x000C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
PENDI	[2]	RW	周期结束中断控制 1: 中断使能 0: 中断禁止	0
STOPI	[1]	RW	定时器/计数器停止中断控制 1: 中断使能 0: 中断禁止	0
STARTI	[0]	RW	定时器/计数器启动中断控制 1: 中断使能 0: 中断禁止	0

14.3.1.5 STC_RISR (计数器中断原始状态寄存器)

- Address = Base Address + 0x0010, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
PENDI	[2]	R	周期结束中断控制 1: 中断发生 0: 中断没有发生	0
STOPI	[1]	R	定时器/计数器停止中断控制 1: 中断发生 0: 中断没有发生	0
STARTI	[0]	R	定时器/计数器启动中断控制 1: 中断发生 0: 中断没有发生	0

14.3.1.6 STC_MISR (计数器中断状态寄存器)

- Address = Base Address + 0x0014, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
PENDI	[2]	R	周期结束中断控制 1: 中断发生 0: 中断没有发生	0
STOPI	[1]	R	定时器/计数器停止中断控制 1: 中断发生 0: 中断没有发生	0
STARTI	[0]	R	定时器/计数器启动中断控制 1: 中断发生 0: 中断没有发生	0

14.3.1.7 STC_ICR (计数器中断状态清除寄存器)

- Address = Base Address + 0x0018, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
PENDI	[2]	W	周期结束中断控制 写1清除该中断状态	0
STOPI	[1]	W	定时器/计数器停止中断控制 写1清除该中断状态	0
STARTI	[0]	W	定时器/计数器启动中断控制 写1清除该中断状态	0

14.3.1.8 STC_SR (计数状态寄存器)

- Address = Base Address + 0x001C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
BUSY	[31]	R	计数器工作状态 1: 计数器正在工作 0: 计数器没有工作，处于闲置状态	0

14.3.1.9 STC_MR (计数器模式控制寄存器)

- Address = Base Address + 0x0024, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
SINGLE	[24]	RW	单次/重复计数模式 1 : 单次计数模式 0 : 重复计数模式 ※ STOPTYPE对单次计数模式有影响	0
STOPTYPE	[9]	RW	停止类型 STOPTYPE: 0 — 在自动重载的时候停止 STOPTYPE: 1 — 立即停止 – SINGLE=1时如果没有收到停止请求，那么会在自动重载时停止 – 收到停止请求时则立即停止	0

注意：在STC_SR中BUSY=1的情况下，不允许修改STC_MR的值。

14.3.1.10 STC_CNTBR (计数基值寄存器)

- Address = Base Address + 0x0030, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
CNTB	[15:0]	R	计数周期寄存器 该寄存器为当前计数器的计数目标值。 需通过写CNTR寄存器更新	0

14.3.1.11 STC_CNTR (计数值寄存器)

- Address = Base Address + 0x0034, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
CNT	[15:0]	RW	计数值以及周期设置 当STC_SR中BUSY=0时，读出值为计数目标值； 写入CNTR的值会立即更新到CNTBR。 当STC_SR中BUSY=1时，读出值为当前计数器的值。 写入CNTR的值，将缓存到PCNTR中，当周期结束后更新到CNTBR。	0

注意： 在STC_SR中BUSY=1时，修改STC_CNTR的值会将该值自动存入STC_PCNTR中。当周期结束时，STC_PCNTR的值会被更新到STC_CNTBR中。

14.3.1.12 STC_CDR (时钟分频寄存器)

- Address = Base Address + 0x0038, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
DIVM	[14:4]	RW	时钟分频M值 计数器时钟 = 时钟源 / (DIVM + 1)	0
DIVN	[3:0]	RW	时钟分频N值 计数器时钟 = 时钟源 / (2^N)	0

注意： 计数器时钟 = 时钟源 / (DIVM + 1) / (2^N)

注意： 在STC_SR中 BUSY = 1时，不允许修改STC_CDR。

14.3.1.13 STC_PCNTR (暂存计数值寄存器)

- Address = Base Address + 0x0050, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
CNT	[15:0]	R	暂存计数值	0

注意：当周期结束时，STC_PCNTR的值会被更新到STC_CNTBR中。

14.3.1.14 STC_CRR (通道启动寄存器)

- Address = Base Address + 0x0080, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
CR	[1:0]	RW	通道启动/停止控制 [1] : 通道1启动/停止控制 [0] : 通道0启动/停止控制 1 : 启动 0 : 停止	0

14.3.1.15 STC_CMR (通道工作模式寄存器)

- Address = Base Address + 0x0084, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												CCM1		CCM0		RSVD													CM1	RSVD	CM0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	RW

Name	Bit	Type	Description	Reset Value
CCM1	[19:18]	RW	通道1模式配置 CCM1 = 捕捉模式 [1] 下降沿输入捕捉 使能(1)/禁止(0) [0] 上升沿输入捕捉 使能(1)/禁止(0) 当STC_CRR中CR[1]是1时禁止修改CCM1	0
CCM0	[17:16]	RW	通道0模式配置 CCM0 = 捕捉模式 [1] 下降沿输入捕捉 使能(1)/禁止(0) [0] 上升沿输入捕捉 使能(1)/禁止(0) 当STC_CRR中CR[0]是1时禁止修改CCM0	0
CM1	[2]	RW	通道1工作模式 0: 匹配模式 1: 捕捉模式 当STC_CRR中CR[1]是1时禁止修改CM1	0
CM0	[0]	RW	通道0工作模式 0: 匹配模式 1: 捕捉模式 当STC_CRR中CR[0]是1时禁止修改CM0	0

注意： 当STC_SR中BUSY=0或者STC_CRR中CRx=0时，STC_CMR可以任意修改。

14.3.1.16 STC_CIMSCR (通道中断控制寄存器)

- Address = Base Address + 0x0088, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																						CC1FI	CC0FI	RSVD						CC1RI	CC0RI
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW

Name	Bit	Type	Description	Reset Value
CC1FI	[9]	RW	通道1 F中断控制 1: 中断使能 0: 中断禁止	0
CC0FI	[8]	RW	通道0 F中断控制 1: 中断使能 0: 中断禁止	0
CC1RI	[1]	RW	通道1 R中断控制 1: 中断使能 0: 中断禁止	0
CC0RI	[0]	RW	通道0 R中断控制 1: 中断使能 0: 中断禁止	0

14.3.1.17 STC_CRISR (通道中断原始状态寄存器)

- Address = Base Address + 0x008C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																						CC1FI	CC0FI	RSVD						CC1RI	CC0RI
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description	Reset Value
CC1FI	[9]	R	通道1 F中断状态 CM1 (STC_CMR) = CC1FI 捕捉模式 = 下降沿输入捕捉 其它模式 = 不使用 1: 中断发生 0: 中断没有发生	0
CC0FI	[8]	R	通道0 F中断状态 CM0 (STC_CMR) = CC0FI 捕捉模式 = 下降沿输入捕捉 其它模式 = 不使用 1: 中断发生 0: 中断没有发生	0
CC1RI	[1]	R	通道1 R中断状态 CM1 (STC_CMR) = CC1RI 捕捉模式 = 上升沿输入捕捉 其它模式 = 匹配 1: 中断发生 0: 中断没有发生	0
CC0RI	[0]	R	通道0 R中断状态 CM0 (STC_CMR) = CC0RI 捕捉模式 = 上升沿输入捕捉 其它模式 = 匹配 1: 中断发生 0: 中断没有发生	0

14.3.1.18 STC_CMISR (通道中断状态寄存器)

- Address = Base Address + 0x0090, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																						CC1FI	CC0FI	RSVD						CC1RI	CC0RI
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description	Reset Value
CC1FI	[9]	R	通道1 F中断状态 CM1 (STC_CMR) = CC1FI 捕捉模式 = 下降沿输入捕捉 其它模式 = 不使用 1: 中断发生 0: 中断没有发生	0
CC0FI	[8]	R	通道0 F中断状态 CM0 (STC_CMR) = CC0FI 捕捉模式 = 下降沿输入捕捉 其它模式 = 不使用 1: 中断发生 0: 中断没有发生	0
CC1RI	[1]	R	通道1 R中断状态 CM1 (STC_CMR) = CC1RI 捕捉模式 = 上升沿输入捕捉 其它模式 = 匹配 1: 中断发生 0: 中断没有发生	0
CC0RI	[0]	R	通道0 R中断状态 CM0 (STC_CMR) = CC0RI 捕捉模式 = 上升沿输入捕捉 其它模式 = 匹配 1: 中断发生 0: 中断没有发生	0

14.3.1.19 STC_CICR (通道中断状态清除寄存器)

- Address = Base Address + 0x0094, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																						CC1FI	CC0FI	RSVD						CC1RI	CC0RI
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	R	R	R	R	R	R	W	W

Name	Bit	Type	Description	Reset Value
CC1FI	[9]	W	通道1 F中断状态清除 写1清除该中断状态	0
CC0FI	[8]	W	通道0 F中断状态清除 写1清除该中断状态	0
CC1RI	[1]	W	通道1 R中断状态清除 写1清除该中断状态	0
CC0RI	[0]	W	通道0 R中断状态清除 写1清除该中断状态	0

14.3.1.20 STC_CAPSR (通道捕捉状态寄存器)

- Address = Base Address + 0x009C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																														C1SR	C0SR
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description	Reset Value
C1SR	[1]	R	通道1上一个输入捕捉状态 1: 捕捉到下降沿输入 0: 捕捉到上升沿输入	0
C0SR	[0]	R	通道0上一个输入捕捉状态 1: 捕捉到下降沿输入 0: 捕捉到上升沿输入	0

14.3.1.21 STC_CC0R (通道0捕捉/比较寄存器)

- Address = Base Address + 0x00C0, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																CC0															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
CC0	[15:0]	RW	通道0的捕捉/比较值 CM0 (STC_CMR): 匹配 — 当计数到CC0设定的值时发生匹配。写CC0立即生效。 CM0 (STC_CMR): 捕捉 — 当捕捉到输入信号变化时，STC_CNTR的值被复制到CC0。写CC0无效。	0

14.3.1.22 STC_CC1R (通道1捕捉/比较寄存器)

- Address = Base Address + 0x00C4, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																CC1															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
CC1	[15:0]	RW	通道1的捕捉/比较值 CM1 (STC_CMR): 匹配 — 当计数到CC1设定的值时发生匹配。写CC1立即生效。 CM1 (STC_CMR): 捕捉 — 当捕捉到输入信号变化时，STC_CNTR的值被复制到CC1。写CC1无效。	0

15 时钟定时器（TC3）

15.1 概述

时钟定时器的功能有实时时钟和计时。时钟定时器的时钟源可以设置为 EMOSC，ISOSC 或者 PCLK.

要启动时钟定时器，只需将控制寄存器(TC3_CR)中的 WTEN 位和 ITS 位置 1，之后时钟定时器开始工作，一段时间后，时钟定时的中断会自动产生，中断的间隔 PRDSEL 位控制。

时钟定时器还可以产生一个稳定的信号驱动蜂鸣器输出管脚 BUZ，控制寄存器(TC3_CR)中的 BCS 位可以选择蜂鸣器的输出频率，0.5KHz，1KHz，2KHz，或者 4KHz。

15.1.1 特性

- 32位内部计数器（在32.768KHz时钟工作时，最大溢出时间为36个小时）。
- 可选工作时钟源。
  - EMOSC，支持32.768KHz的晶振。
  - ISOSC，支持500KHz，或者3MHz。
- 定时器支持两种工作模式：周期计数模式，连续计数模式。
- 频率可配置的蜂鸣器载波输出，支持0.5KHz、1KHz、2KHz和4KHz可选。

15.1.2 管脚描述

Table 15-1 管脚描述

Pin Name	Function	I/O Type	Active Level	Comments
TC3_BUZZ	蜂鸣器频率输出	O	-	-

15.2 功能描述

15.2.1 模块框图

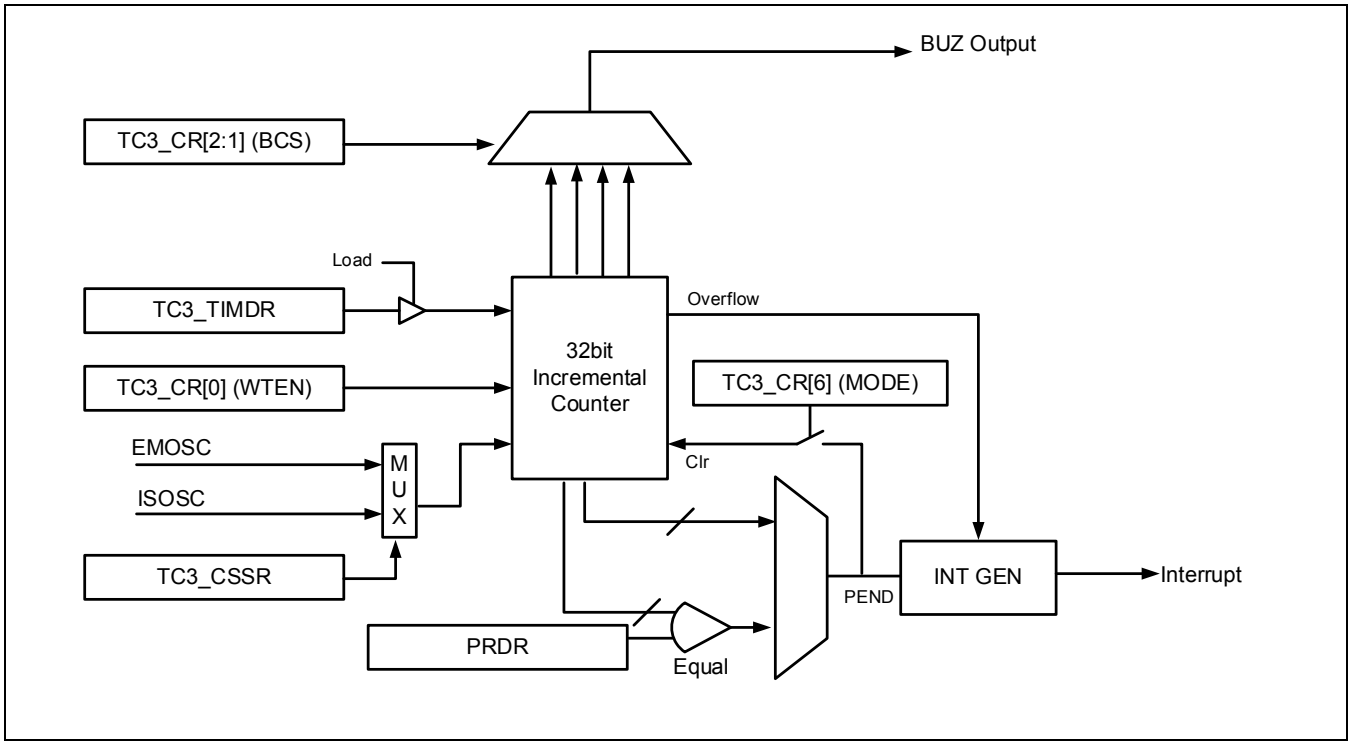


Figure 15-1 时钟定时器模块框图

15.2.2 工作原理

TC3 是一个 32 位长度的递增型计数器。工作时钟可以在 EMOSC 和 ISOSC 中进行选择，当 EMOSC 的外部晶振工作在 32.768KHz 时，此计数器可以作为 RTC 时钟使用。

TC3 可以支持两种工作模式，一种为普通计数模式（NORMAL MODE）。在此模式下，计数器一直递增，直到计数器溢出。溢出以后自动归零重新开始计数。在此模式下，当计数器值等于 PRDSEL 设置值时，会产生 PEND 中断。当计数器溢出时，会产生 OVF 中断。另外一种工作模式为周期计数模式（PERIOD MODE）。在此模式下，计数器一直递增，当发生 PEND 中断时，计数器会自动归零重新开始计数。

TC3 的计数在 WTEN 设置为高以后，开始计数。当 WTEN 被清除以后，计数器停止计数。在下一次 WTEN 置高时，计数器会被清零，重新开始计数。计数器的当前计数值可以通过读取 TC3_TIMDR 获得。当对 TC3_TIMDR 进行写操作时，写入值将会被直接载入计数器，计数器从下一个周期开始，会从更新后的计数值开始计数。

在周期计数模式时，通过设置 TC3_CR 寄存器的 PRDSEL 设置周期长度。在周期长度超过 2s 时，可以通过 PRDR 寄存器设置基于 2s 的计数周期数。

TC3 自带蜂鸣器载波发生功能，可以选择 0.5KHz、1KHz、2KHz 或者 4KHz 作为输出。

15.3 寄存器说明

15.3.1 寄存器表

- Base Address: 0x4005_3000

Register	Offset	Description	Reset Value
TC3_IDR	0x000	TC3 ID控制寄存器	0x0001_002D
TC3_CSSR	0x004	TC3时钟源选择寄存器	0x0000_0001
TC3_CEDR	0x008	TC3时钟使能/禁止控制寄存器	0x0000_0000
TC3_SRR	0x00C	TC3软件复位寄存器	0x0000_0000
TC3_CR	0x010	TC3控制寄存器	0x0000_0030
TC3_PRDR	0x014	TC3周期设置寄存器	0x0000_0001
TC3_TIMDR	0x018	TC3计数器数据寄存器	0x0000_0000
TC3_IMCR	0x01C	TC3中断使能控制寄存器	0x0000_0000
TC3_RISR	0x020	TC3中断原始状态寄存器	0x0000_0000
TC3_MISR	0x024	TC3中断状态寄存器	0x0000_0000
TC3_ICR	0x028	TC3中断状态清除寄存器	0x0000_0000

NOTE:

15.3.2 TC3_IDR (ID控制寄存器)

- Address = Base Address + 0x0000, Reset Value = 0xFFF0_0000

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

Name	Bit	RW	Description	Reset Value
IDR	[23:0]	R	ID Code寄存器 相应IP的ID Code，无法更改。	0x1002D

15.3.3 TC3_CSSR (时钟源选择寄存器)

- Address = Base Address + 0x0004, Reset Value = 0x0000_0001

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

Name	Bit	RW	Description	Reset Value
CSSR	[0]	RW	计数器时钟源选择： 0： 外部晶振 1： 内部低速振荡器 当外部晶振作为32.768KHz时钟的输入源，定时器可以做精准的时间计时。	0x1

15.3.4 TC3_CEDR (时钟使能/禁止控制寄存器)

- Address = Base Address + 0x0008, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
CLKEN	[0]	RW	计数器时钟使能/禁止控制位。 0：禁止计数器时钟。 1：使能计数器时钟。 SWRST不影响CLKEN位的状态。	0x0
DBGEN	[1]	RW	Debug模式使能/禁止控制位。 0： 禁止Debug模式。 1： 使能Debug模式。 如果DBGEN置1，当CPU在debug模式被暂停后，计数器也同时被挂起。否则，计数器则继续工作。	0x0

15.3.5 TC3_SRR (软件复位寄存器)

- Address = Base Address + 0x000C, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
SWRST	[1:0]	W	软件复位。 0：无效 1：软件复位	0x0

15.3.6 TC3_CR (计数器控制寄存器)

- Address = Base Address + 0x0010, Reset Value = 0x0000_0030

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																								MODE	PRDSEL			BCS		WTEN	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
WTEN	[0]	RW	计数器使能控制位 0： 停止计数 1： 启动计数	0x0
BCS	[2:1]	RW	蜂鸣器输出频率选择。 00：0.5KHz 01：1.0KHz 10：2.0KHz 11：4.0KHz	0x0
PRDSEL	[5:3]	RW	周期设置寄存器。(1) 000：3.91ms （2^7个时钟周期） 001：15.625ms （2^9个时钟周期） 010：62.5ms （2^{11}个时钟周期） 011：125ms （2^{12}个时钟周期） 100：250ms （2^{13}个时钟周期） 101：500ms （2^{14}个时钟周期） 110：1s （2^{15}个时钟周期） 111：使用PRDR寄存器设置值	0x6
MODE	[6]	W	计数器工作模式选择 (只可写)。 0：NORMAL模式。 1：PERIOD模式。	0x0

NOTE: 1) 标注的周期设置值是基于 32.768KHz 工作频率的。当选择其他工作频率时，需要进行换算。
2) 计数器工作模式选择位只可写不可读。

15.3.7 TC3_PRDR (周期设置寄存器)

- Address = Base Address + 0x0014, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
PRDR	[15:0]	RW	周期设置寄存器。 该寄存器值表示基于2s的周期长度。例如当该寄存器设置为4时，表示周期为4x2s=8s。 当该寄存器设置为0时，默认为1。	0x1

15.3.8 TC3_TIMDR (计数器数据寄存器)

- Address = Base Address + 0x0018, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
TIMDR	[31:0]	RW	计数器数据寄存器 当对该寄存器进行读操作时，返回当前计数器值。 当对该寄存器进行写操作时，计数值被TIMDR重置。	0x0

15.3.9 TC3_IMCR（中断使能控制寄存器）

- Address = Base Address + 0x001C, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
PEND	[0]	RW	周期结束中断标志位	0x0
OVF	[1]	RW	计数器溢出中断标志位	0x0

该寄存器用于使能中断。

0： 关闭中断
1： 打开中断

15.3.10 TC3_RISR (中断原始状态寄存器)

- Address = Base Address + 0x0020, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
PEND	[0]	R	周期结束中断原始标志位	0x0
OVF	[1]	R	计数器溢出中断原始标志位	0x0

15.3.11 TC3_MISR (中断状态寄存器)

- Address = Base Address + 0x0024, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
PEND	[0]	R	周期结束中断标志位	0x0
OVF	[1]	R	计数器溢出中断标志位	0x0

15.3.12 TC3_ICR (中断清除控制寄存器)

- Address = Base Address + 0x0028, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
PEND	[0]	W	清除周期结束中断标志位	0x0
OVF	[1]	W	清除计数器溢出中断标志位	0x0

该寄存器用于清除原始中断源标志位，该寄存器为只写寄存器

0： 无效果
1： 清除中断标志

16 EPWM (Enhanced PWM)

16.1 概述

此MCU内嵌了一个16位的EPWM模块。该PWM模块包含了3组6个独立的PWM通道，每组可以输出两路独立的，或者互补的PWM信号，该PWM模块还支持和模拟比较器的协同工作用以支持针对电磁加热以及电机的专门应用。

16.1.1 特性

- 16位计数方向可编程的计数器
    - 递增计数
    - 递减计数
    - 递增后递减计数
    - 递减后递增计数
- PWM 波形产生控制，支持双路独立输出模式，PWM互补输出模式和双路逐次触发输出模式
- 互补输出死区控制
- 数字PWM波形生成引擎
- 单计数器模式和独立计数器模式可选
- 比较器联动控制
    - 延时触发控制
    - 防误触发控制
    - 软锁止和硬锁止设定

16.1.2 管脚描述

Table 16-1 EPWM 管脚描述

Pin Name	Function	I/O Type	Active Level	Comments
EPWM0_X	PWM0 输出X端口	O	-	-
EPWM0_Y	PWM0 输出Y端口	O	-	-
EPWM1_X	PWM1 输出X端口	O	-	-
EPWM1_Y	PWM1 输出Y端口	O	-	-
EPWM2_X	PWM2 输出X端口	O	-	-
EPWM2_Y	PWM2 输出Y端口	O	-	-
EPWM_EP[4:0]	PWM的外部中断触发端口	I	下降沿	下降沿触发

16.2 功能描述

16.2.1 模块框图

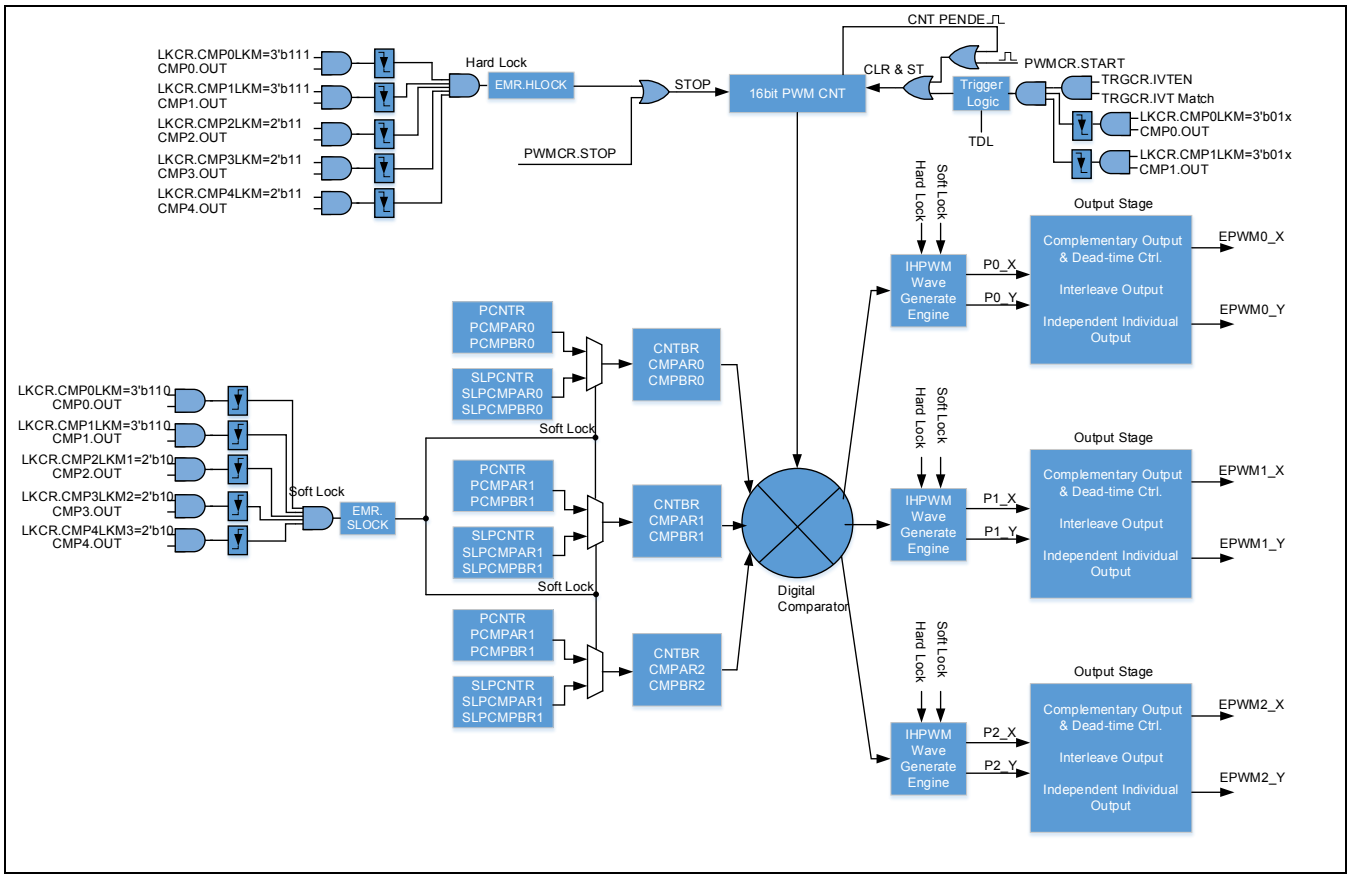


Figure 16-1 EPWM模块框图 – 单计数器模式

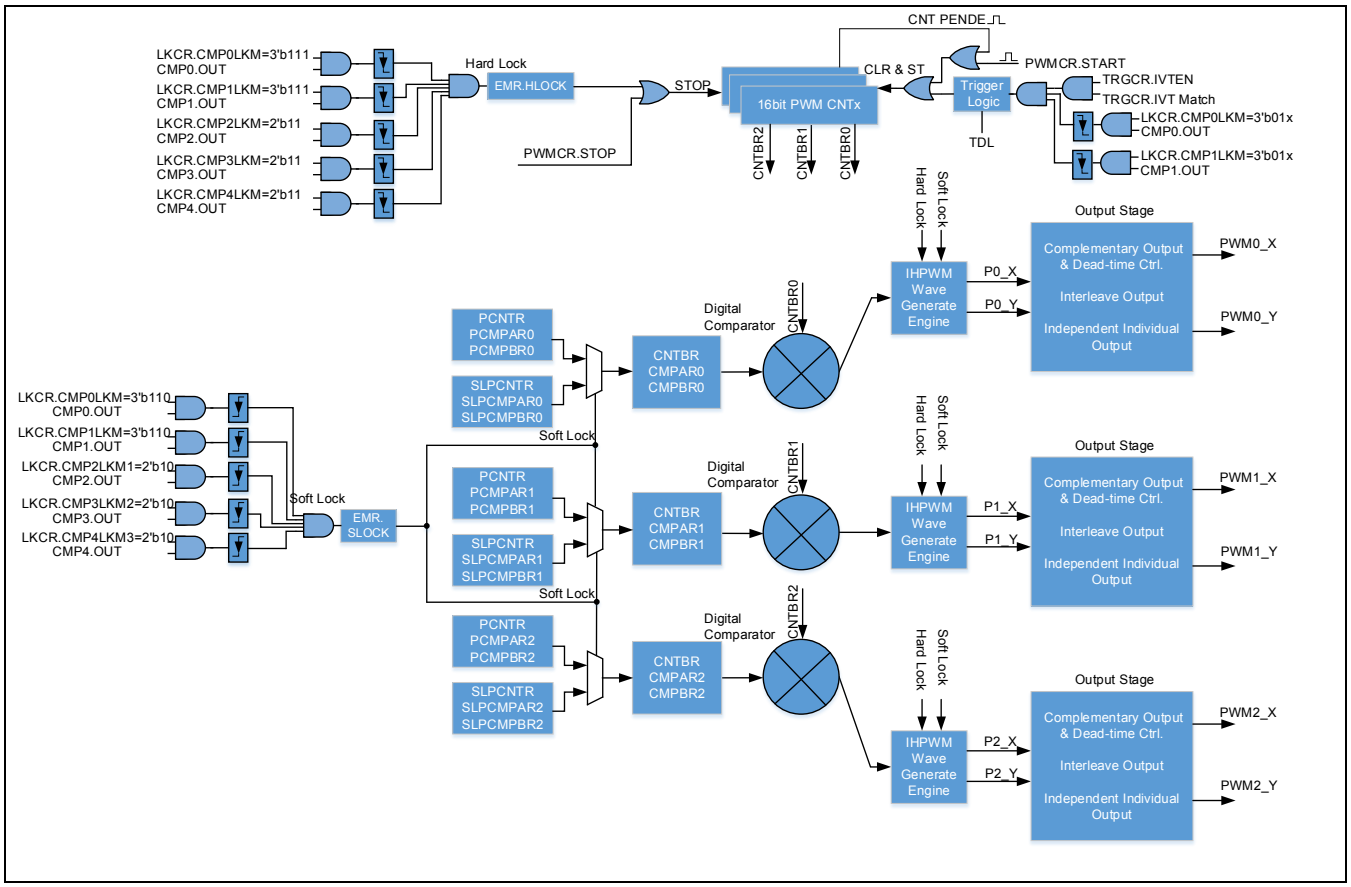


Figure 16-2 EPWM模块框图 – 独立计数器模式

16.2.2 工作原理

EPWM 模块作为专门为功率控制设计的数字 PWM 模块，可以提供非常便利和智能的 PWM 波形输出。该模块具有 3 组共 6 路输出，每组两路 PWM 信号输出（EPWM_X、EPWM_Y），两路 PWM 输出可以配置为独立的 PWM 信号，或者是互补带死区控制的一组 PWM 信号，又或者是间隔触发输出的一组 PWM 信号。

EPWM 内部具有三个 16 位的计数器，可以设置为单计数器模式(只有 1 个计数器工作)和独立计数模式(三个计数器分别独立工作)。计数器通过可分频的系统时钟控制，最高支持 40MHz 的工作频率。计数器支持 4 种不同的计数模式，在一个周期内支持三个比较值的比较，分别为 CMPA、CMPB 和 CNTB。在每个比较值匹配点可以对波形输出电平进行控制，从而可以实现非常灵活的生成自定义的 PWM 波形。

该 EPWM 模块还可以和 5 个比较器进行联动，其中 CMP0 和 CMP1 可以作为 PWM 模块的启动触发信号，触发延时可以通过寄存器进行调整，两次连续触发间的最小时间可以通过 TRGIVT 控制，以保证在触发以后的一段时间内不会被干扰而错误触发。CMP0~CMP4 中的任意一个输出都可以作为软锁止（Soft Lock）和硬锁止（Hard Lock）的触发信号。

16.2.3 计数器工作模式控制

16.2.3.1 计数器工作模式

EPWM的计数器可以在4种不同的计数方式下工作。

- 递增模式（Up Counting Mode）
- 递减模式（Down Counting Mode）
- 递增递减模式（Up-down Counting Mode）
- 递减递增模式（Down-up Counting Mode）

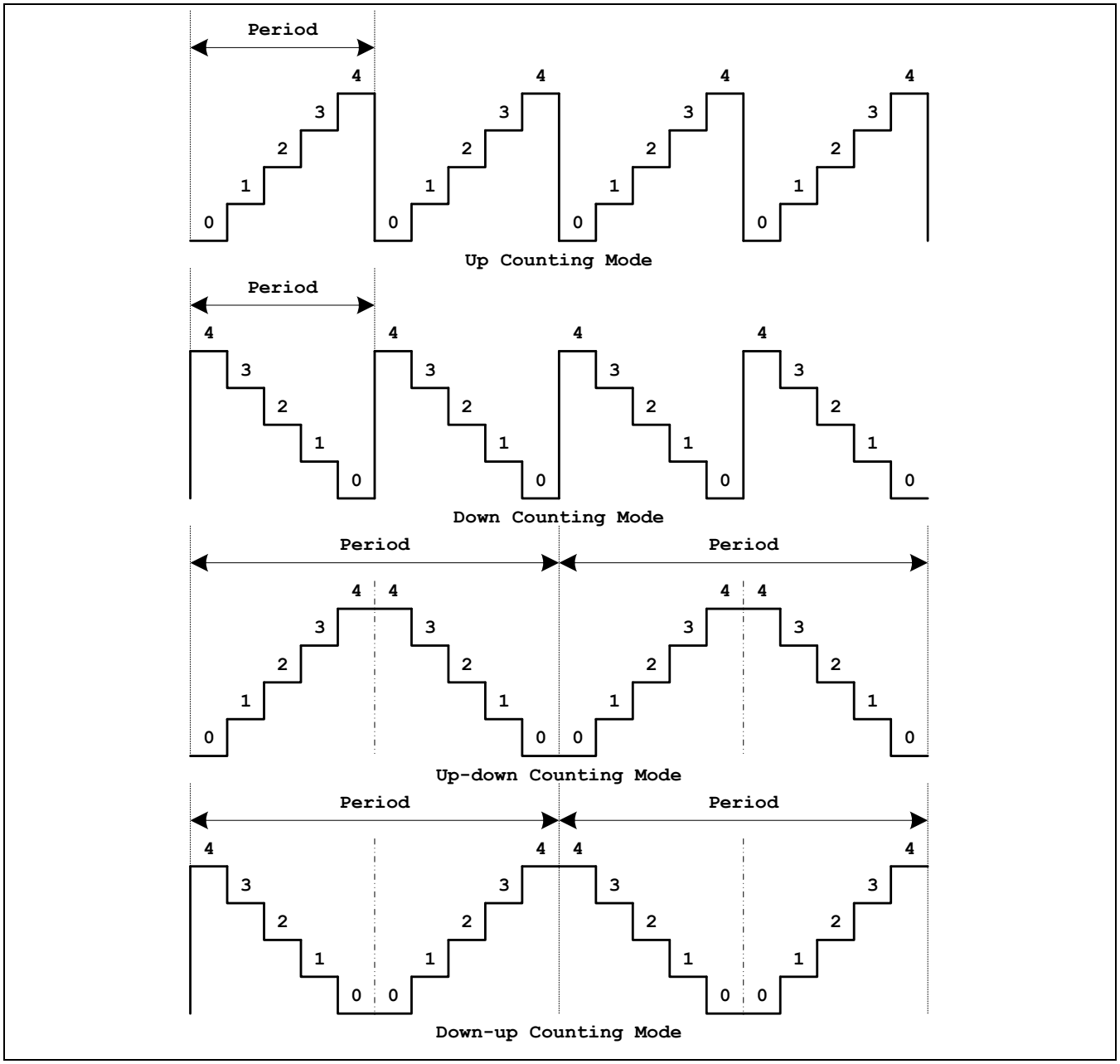


Figure 16-3 计数器工作模式

当PWM计数器在递增模式下工作时，计数器收到START信号以后，从’0’开始计数直到寄存器设置值（EPWM_CNTR）。计数器值在每个PCLK的上升沿增加1（SPCLK_EN置位时）。

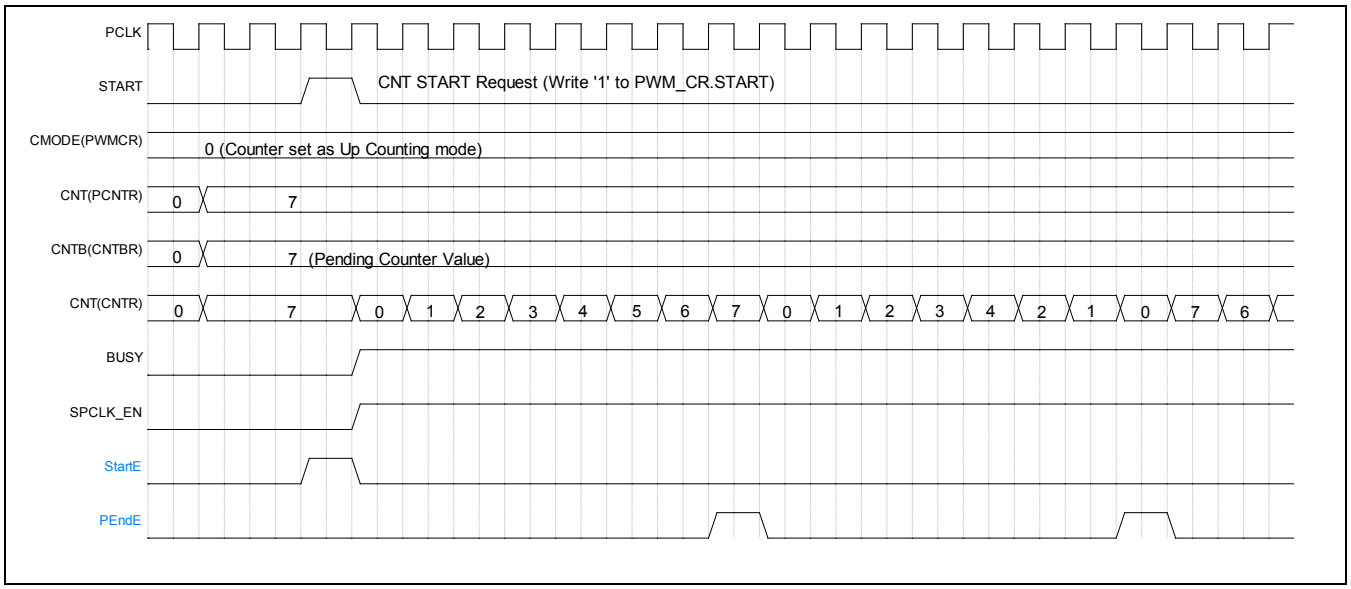


Figure 16-4 计数器递增模式

当PWM计数器在递减模式下工作时，计数器收到START信号以后，从寄存器设置值（EPWM_CNTR）开始递减直到’0’。计数器值在每个PCLK的上升沿减少1（SPCLK_EN置位时）。

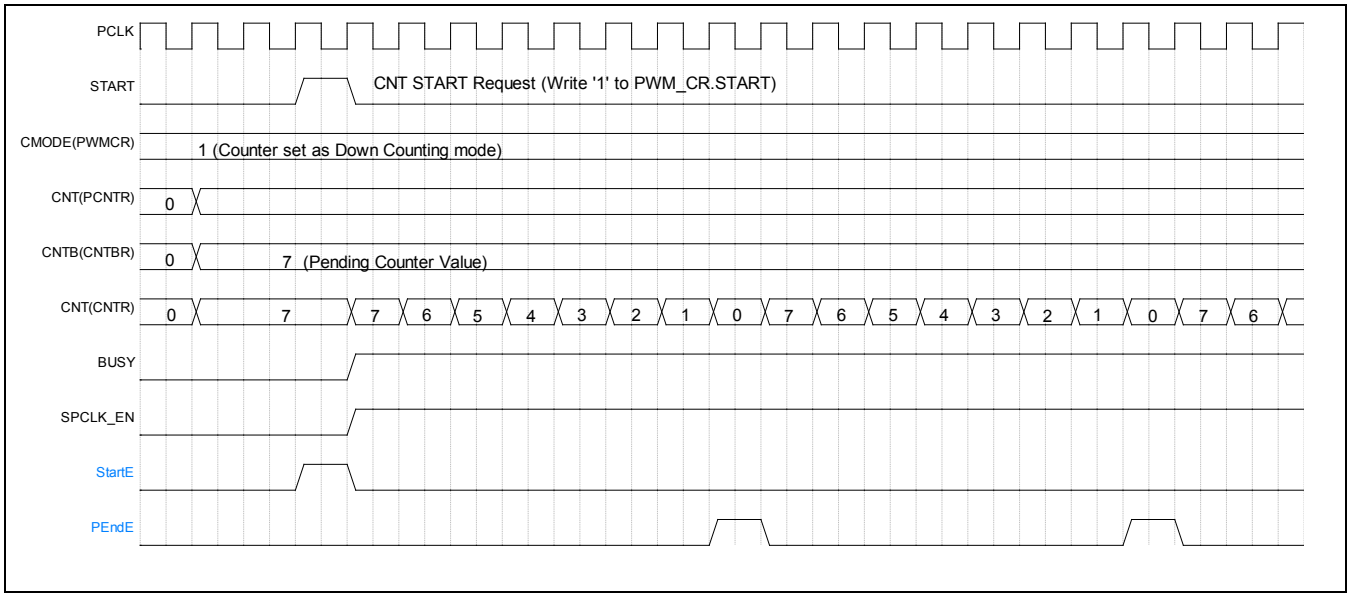


Figure 16-5 计数器递减模式

当PWM计数器在递增递减模式下工作时，计数器收到START信号以后，先从’0’开始计数直到寄存器设置值（EPWM_CNTBR），此时CENTERE事件将被触发，然后计数器从EPWM_CNTBR.CNTB值开始递减直到’0’。

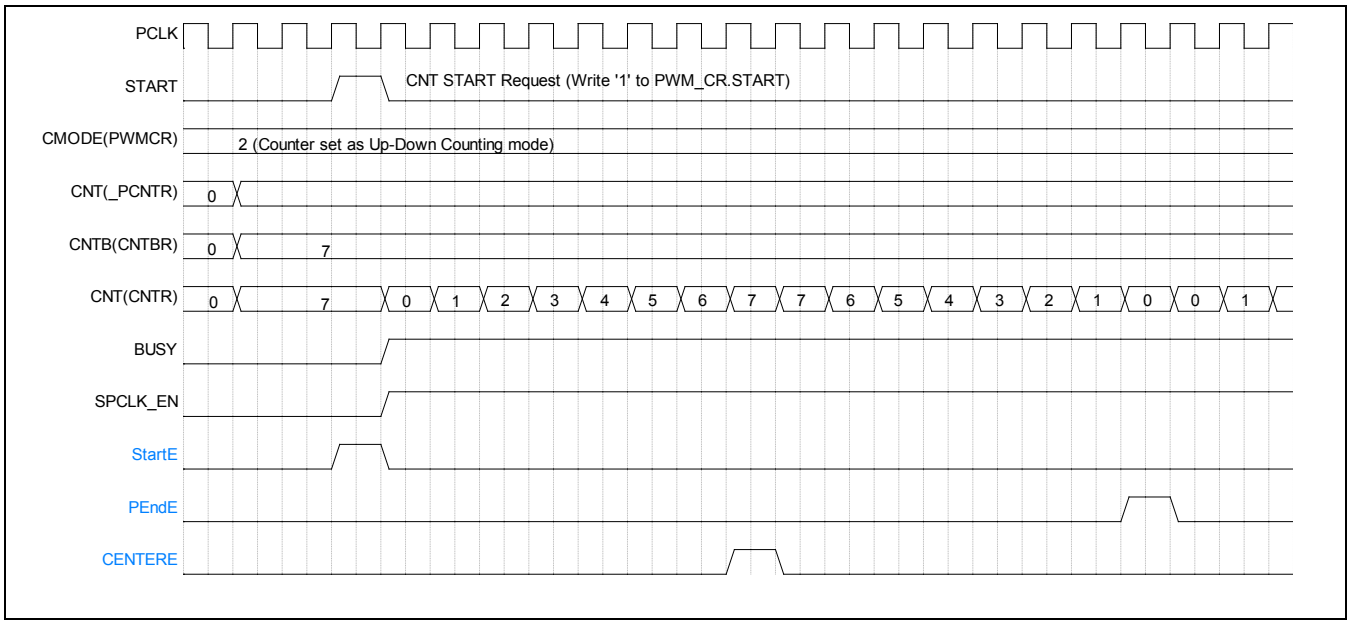


Figure 16-6 计数器递增递减模式

当PWM计数器在递减递增模式下工作时，计数器收到START信号以后，先从寄存器设置值（EPWM_CNTR）开始递减直到’0’，此时CENTERE事件将被触发，然后计数器从’0’开始递增计数直到寄存器设置值。

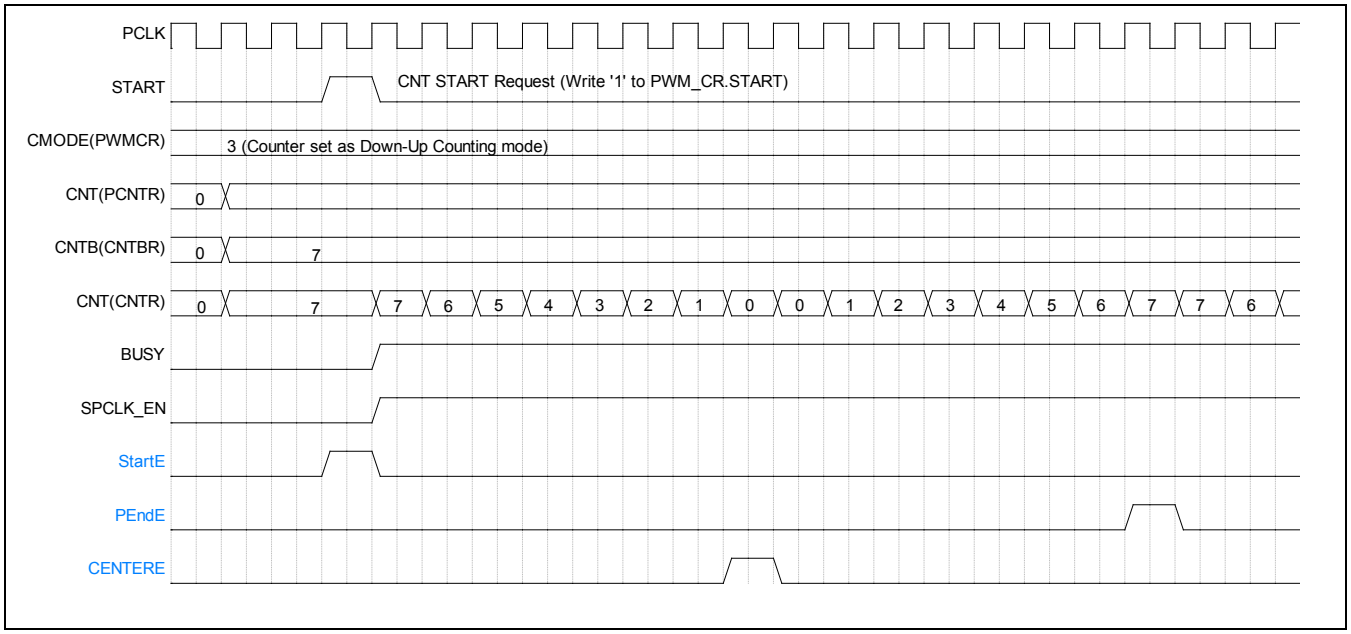


Figure 16-7 计数器递减递增模式

16.2.4 自动更新

16.2.4.1 自动更新框图

EPWM计数器一共具有7种比较值，CNTBR（基本比较值，控制PWM波形周期）、CMPAR0（PWM0通道比较值A）、CMPBR0（PWM0通道比较值B）、CMPAR1（PWM1通道比较值A）、CMPBR1（PWM1通道比较值B）、CMPAR2（PWM2通道比较值A）、CMPBR2（PWM2通道比较值B）。

每一个比较值寄存器都支持计数工作时更新。在计数器工作时，对这些比较值的更新都会被暂存在相对应的PENDING寄存器中（PCNTR、PCMPAR0/1/2、PCMPBR0/1/2），PENDING寄存器的值将在PENDE事件发生时，自动更新到相对应的比较值寄存器中。计数器未工作时，这些PENDING的值会立即更新到相对应的比较寄存器中。

在比较器联动模式下（CMP-LINK），如果检测到Soft-lock，则比较值会自动从相对应的SLPCNTR、SLPCMPAR0/1、SLPCMPBR0/1寄存器中更新。(PWM2通道不支持比较器联动功能)

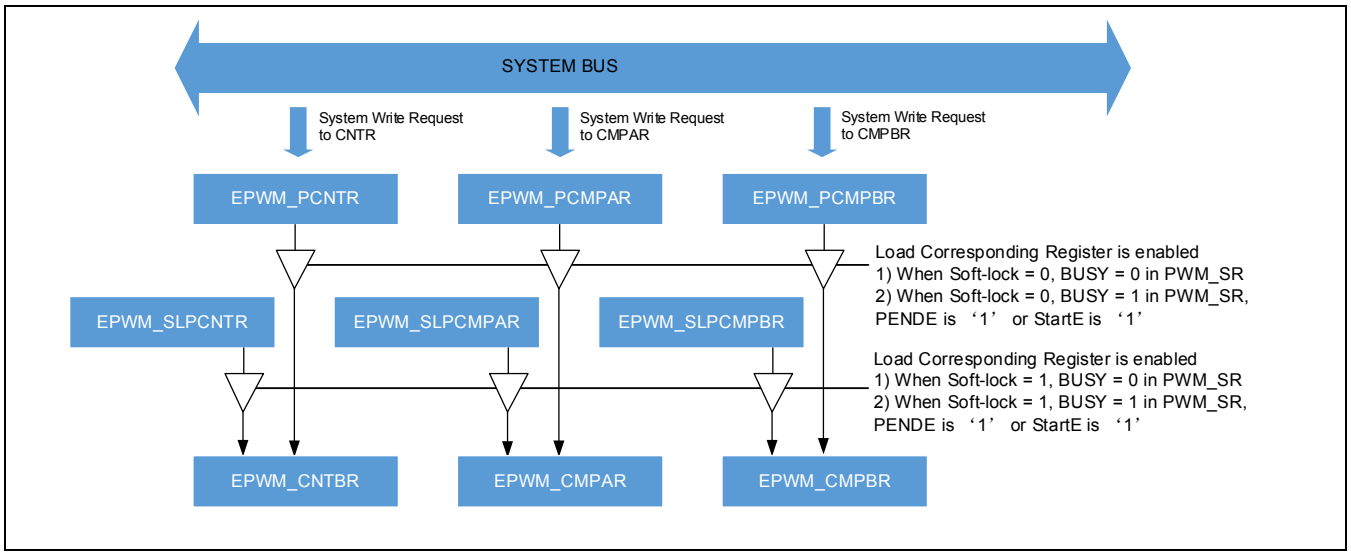


Figure 16-8 比较值自动更新条件

16.2.4.2 自动更新工作原理

当比较器工作时（EPWM_CR.BUSY = 1），PWM将周期性地自动更新比较寄存器，自动更新都发生在PENDE事件发生时。当计数器没有开始工作时，更新值会立即更新到相对应的寄存器中。

在比较器联动模式下，在Soft-lock事件被触发后，更新源将自动切换到Soft-lock相对应的寄存器中。

16.2.5 EPWM启动停止控制

16.2.5.1 启动EPWM的方式

EPWM 的启动可通过置位EPWM_CR.START软件启动，或者在使能比较器联动以后，由指定的比较器输出来触发。在BUSY=1条件下，如果START被置位，则PWM计数器和分频计数器都将重新开始计数。

通过软件写EPWM_EMR.SLOCK位，可以清除Soft-lock标志位。Hard-lock标志的清除必须通过设置EPWM_EMR.HLOCK 来实现。在Hard-lock发生以后，PWM的计数器会被停止，当该标志被清除以后，PWM计数器需要重新启动。

16.2.5.2 比较器联动和启动触发

EPWM支持比较器联动，通过设置EPWM_LKCR 可以使能比较器联动模式。在联动模式下，指定比较器的输出可以作为启动计数器的触发信号。为避免连续的错误触发，可以通过设置比较器的触发间隔来控制两次触发之间的时间间隔（EPWM_LKCR.TRGIVT）。

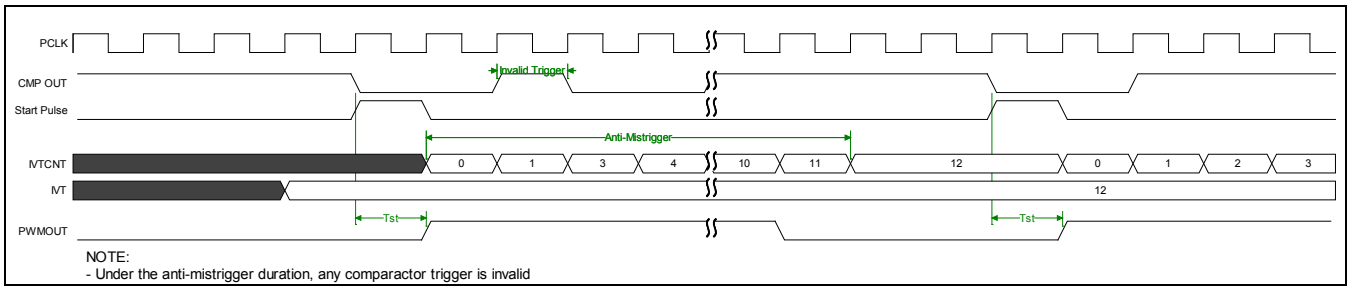


Figure 16-9 防误触发保护

比较器触发还支持延时触发功能，在比较器输出触发以后延时一定的时间再启动计数器，此延时可以通过EPWM_LKCR.TRGTDL设置。延时设置Tdly = Tpclk x 4 x TDL。

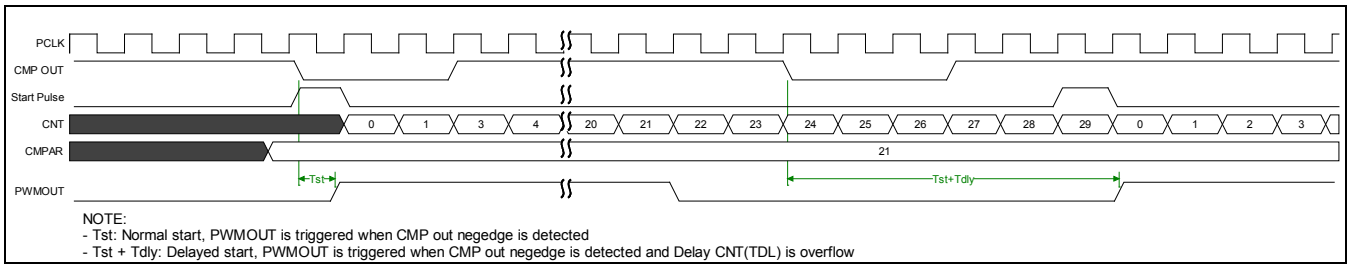


Figure 16-10 比较器延时触发启动

在联动模式下，EPWM有两种异常处理模式，一种为soft-lock模式，另外一种为hard-lock模式。通过EPWM_LKCR指定的比较器输出可以作为两种模式的触发源。当异常模式触发以后，相对应的标志位会被置位，可以通过状态寄存器（EPWM_EMR）查询。

关于soft-lock，EPWM提供两种工作模式：

1. 普通模式。当soft-lock发生时，当前周期的PWM输出被立即切换到EMR.SL_PxS设定的固定值，而在计数器重启以后的下一个周期自动切换到SLPCNTR、 SLPCMPAR、SLPCMPBR中设置的值作为当前比较值，并且输出PWM波形，直到soft-lock的状态被清除。在普通模式下，soft-lock只可以通过软件对EPWM_EMR.SLOCK位写1来清除，比较器触发的START条件不会清除soft-lock标志。soft-lock清除后的下

个PWM周期，PWM的输出将切换回非SLP寄存器(CNTR,CMPAR,CMPBR)中设置的值。

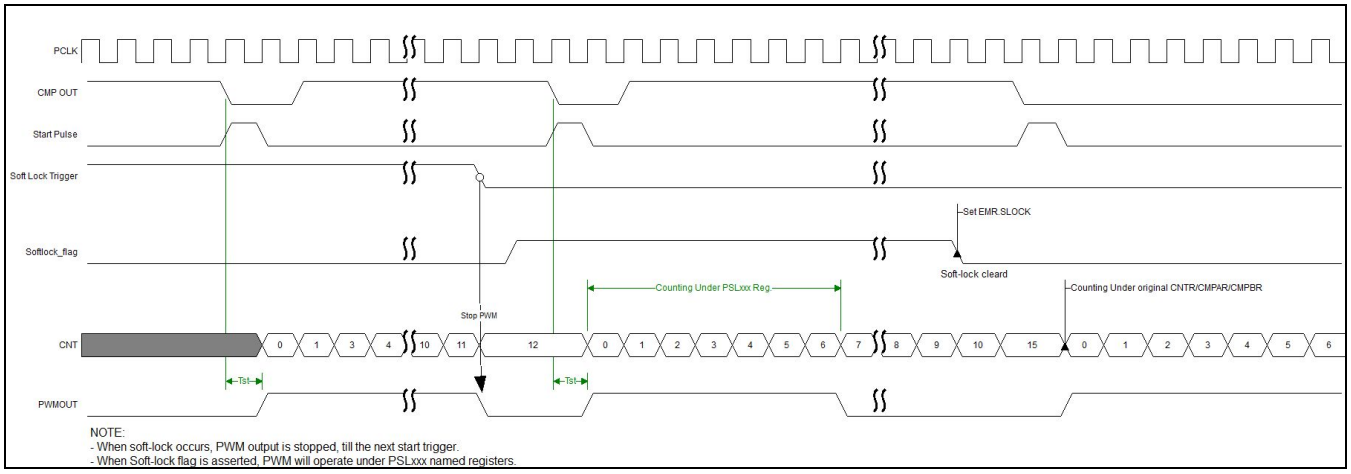


Figure 16-11 Soft Lock 普通模式

2. 自动调节模式。当soft-lock发生时，当前周期的PWM输出被切换到EMR.SL_PxS设定的固定值，而在计数器重启以后的下一个周期自动切换到SLPCNTR、 SLPCMPAR、SLPCMPBR中设置的值作为当前比较值，并且输出PWM波形，这个由SLP寄存器生成的周期如果继续发生soft-lock，那么这个周期中SLPCMPAR、SLPCMPBR的值将会自动以SLSTEP寄存器中设置的步长减小(或者增大)一次以供下个周期使用，SLPCNTR的值将会自动以SLSTEP寄存器值2倍的步长减小(或者增大)一次以供下个周期使用，直到不发生soft-lock的周期后，SLPCMPAR、SLPCMPBR的值又会自动以SLSTEP指定的值增大(或者减小)，SLPCNTR的值以SLSTEP值的2倍增大(或者减小)，直到SLPCMPAR、SLPCMPBR大于等于(或者小于等于)CMPAR、CMPBR的值，自动恢复到发生soft-lock之前的状态。使用自动调节模式需要注意：

   a. 自减和自增的功能可以分别打开或者关闭，但是只能选择SLPCMPAR和SLPCMPBR其中一个进行自减自增，不能同时让两者一起自减自增。

   b. 在实现所需PWM波形时，注意需要选择合适的比较值和soft-lock值。当设置的SLPCMPAR/SLPCMPBR 大于 CMPAR/CMPBR 时，自动调节功能会先自增再自减；而当设置的SLPCMPAR/SLPCMPBR 小于 CMPAR/CMPBR 时，自动调节功能会先自减再自增。

   c. 自动调节功能如果发生溢出，也就是SLPCMPAR/SLPCMPBR的值超过了整个周期的范围(0~CNTR)，那么EPWM将产生hard-lock，也即是关断所有输出，同时产生PWMx_SLPx_OVF中断。所以在使用自动调节功能时，必须注意不能让调节的范围溢出。

注意：
1. Soft-lock的自动调节功能只有在递减计数模式下可用(EPWM_CR寄存器里CMODE = 0x1)。
2. 自动调节模式只有在单计数器模式下(EPWM_CR.SINGLE=1)可用，独立计数器模式下不支持自动调节功能。

当hard-lock发生时，当前PWM计数器被紧急停止。如果hard-lock标志未清除，即使软件重置了START，计数器也无法启动。当hard-lock发生时，EPWM的输出状态可以设置为保持、高阻、高电平或者低电平输出（EPWM_EMR寄存器）。

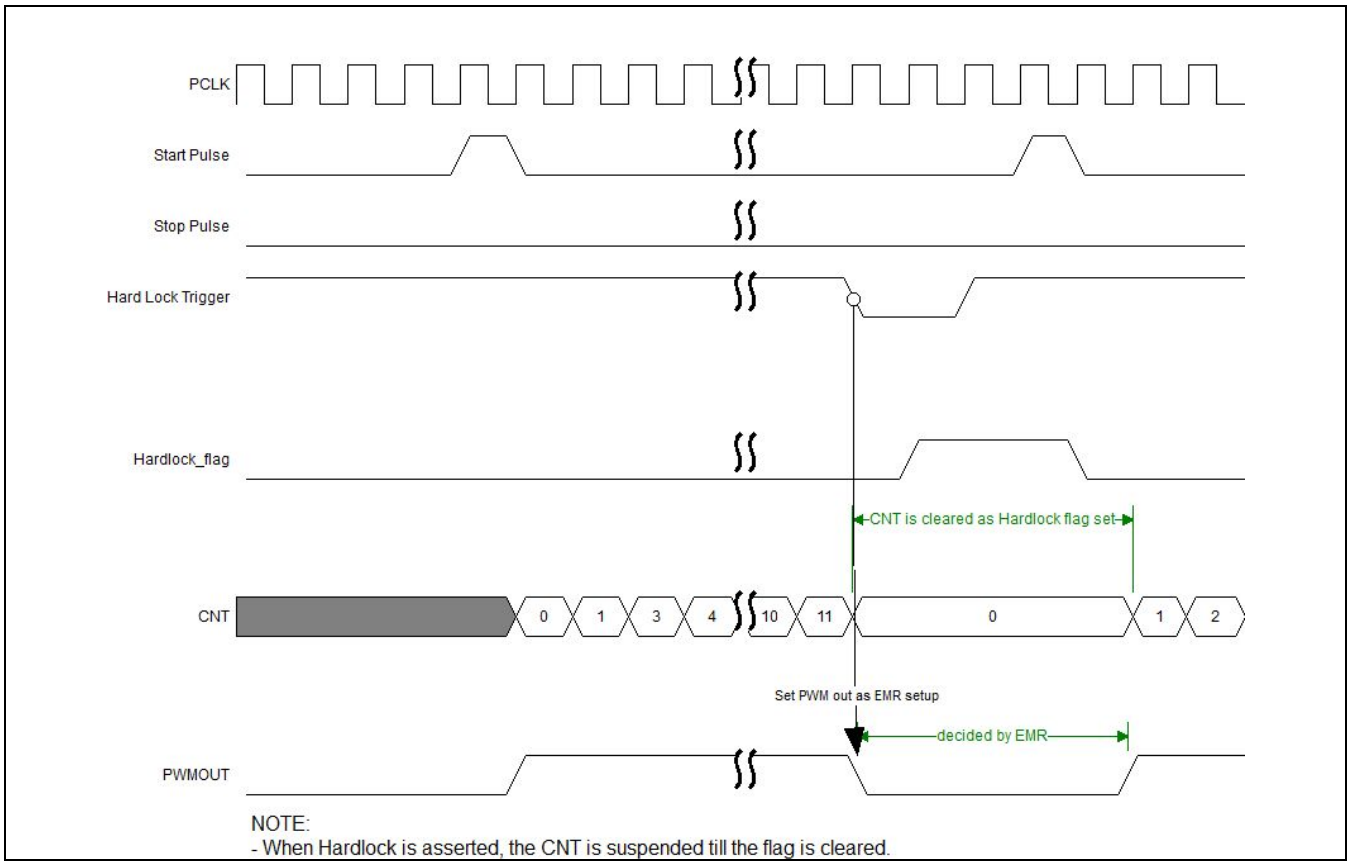


Figure 16-12 Hard Lock

16.2.5.3 EPWM停止控制

EPWM的停止可以通过软件置位EPWM_CR.STOP实现。在比较器联动模式下，如果hard-lock事件被检测到，计数器会自动停止工作。

16.2.6 比较事件生成器

16.2.6.1 比较事件生成模块框图

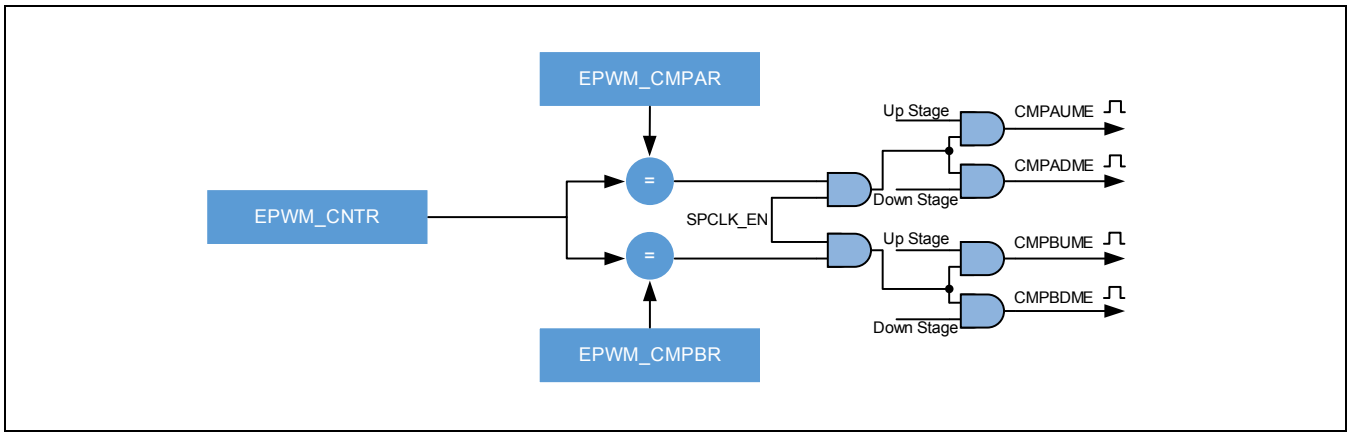


Figure 16-13 比较事件生成模块

16.2.6.2 比较事件生成

在EPWM中，有两个比较寄存器。通过比较比较寄存器的值和计数器的值，可以产生4种比较值匹配事件。当EPWM_CMPAR 等于0，或者EPWM_CMPAR的值等于EPWM_CNTBR时，将不会产生CMPAUME和CMPADME事件。当EPWM_CMPBR 等于0，或者EPWM_CMPBR的值等于EPWM_CNTBR时，将不会产生CMPBUME和CMPBDME事件。

当下列条件满足时，相应的比较值匹配事件将会触发。

Table 16-2 比较事件触发条件

事件	事件触发条件
CMPADME	当计数器处于递减阶段 EPWM_CMPAR 的值介于‘0’和 EPWM_CNTBR 之间 EPWM_CNTR 等于 EPWM_CMPAR
CMPAUME	当计数器处于递增阶段 EPWM_CMPAR 的值介于‘0’和 EPWM_CNTBR 之间 EPWM_CNTR 等于 EPWM_CMPAR
CMPBDME	当计数器处于递减阶段 EPWM_CMPBR 的值介于‘0’和 EPWM_CNTBR 之间 EPWM_CNTR 等于 EPWM_CMPBR
CMPBUME	当计数器处于递增阶段 EPWM_CMPBR 的值介于‘0’和 EPWM_CNTBR 之间 EPWM_CNTR 等于 EPWM_CMPBR

当 BUSY=0，不会有任何比较事件产生。

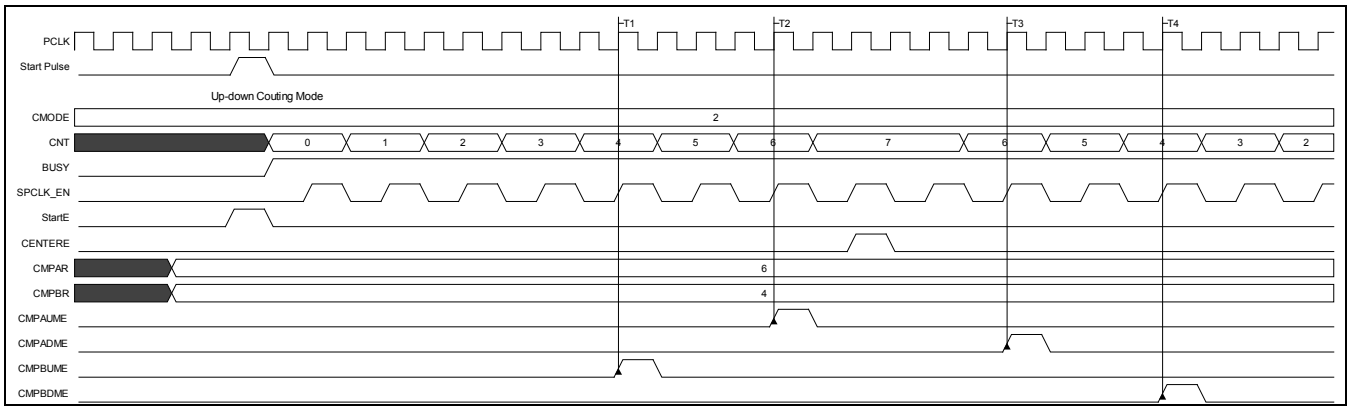


Figure 16-14 比较事件生成时序图

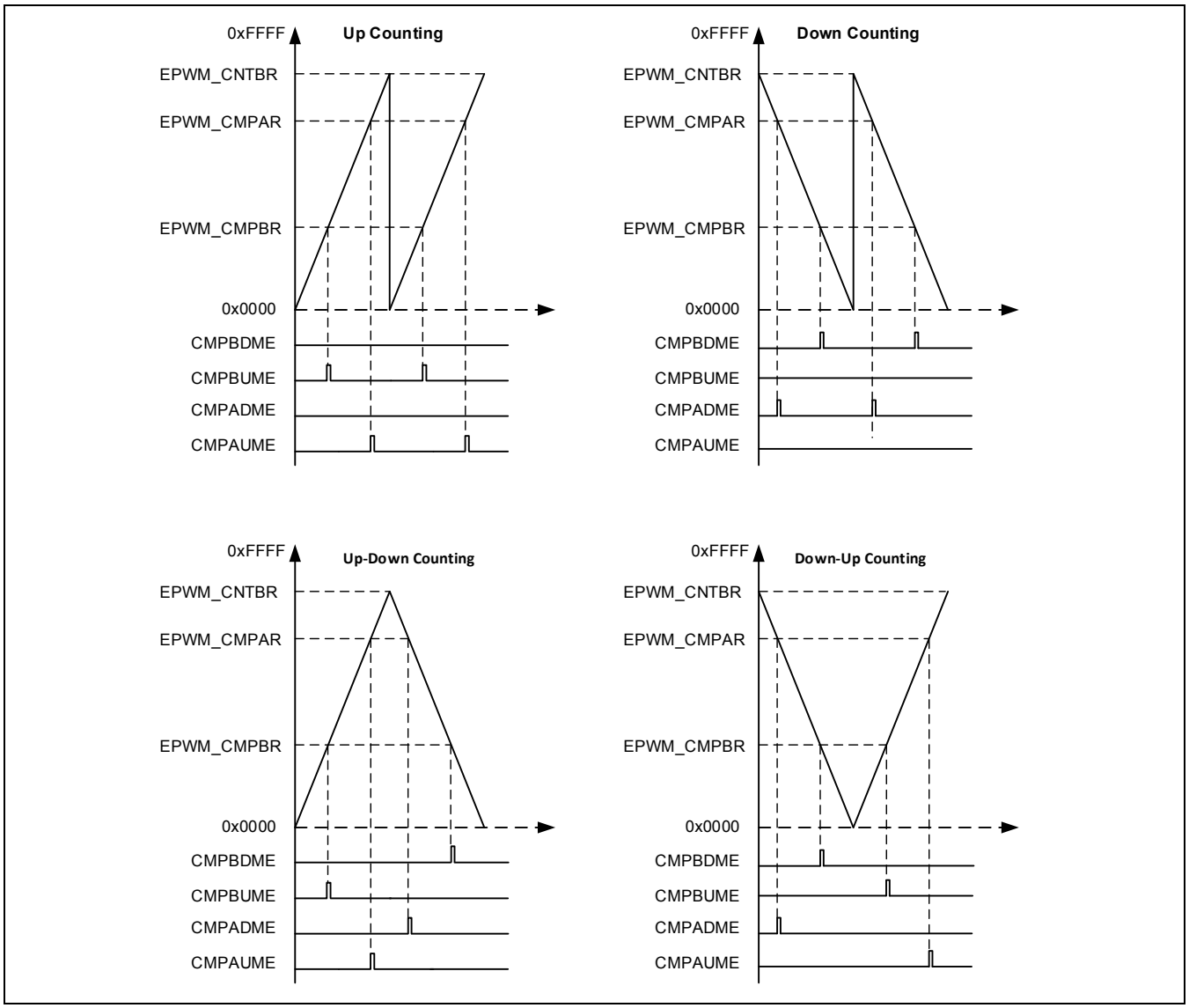


Figure 16-15 计数模式和比较事件相互关系

16.2.7 EPWM波形生成引擎

EPWM数字引擎模块可以产生两路基础PWM信号(PX、PY)。数字引擎根据计数器和数字比较器的各种事件，生成内部PX和PY信号。PX和PY信号可以作为两路独立的PWM数字信号输出，或者选择其中一路作为互补输出模块、间隔触发输出模块的输入信号，通过这些模块的后处理以后再输出。

16.2.7.1 EPWM引擎模块框图

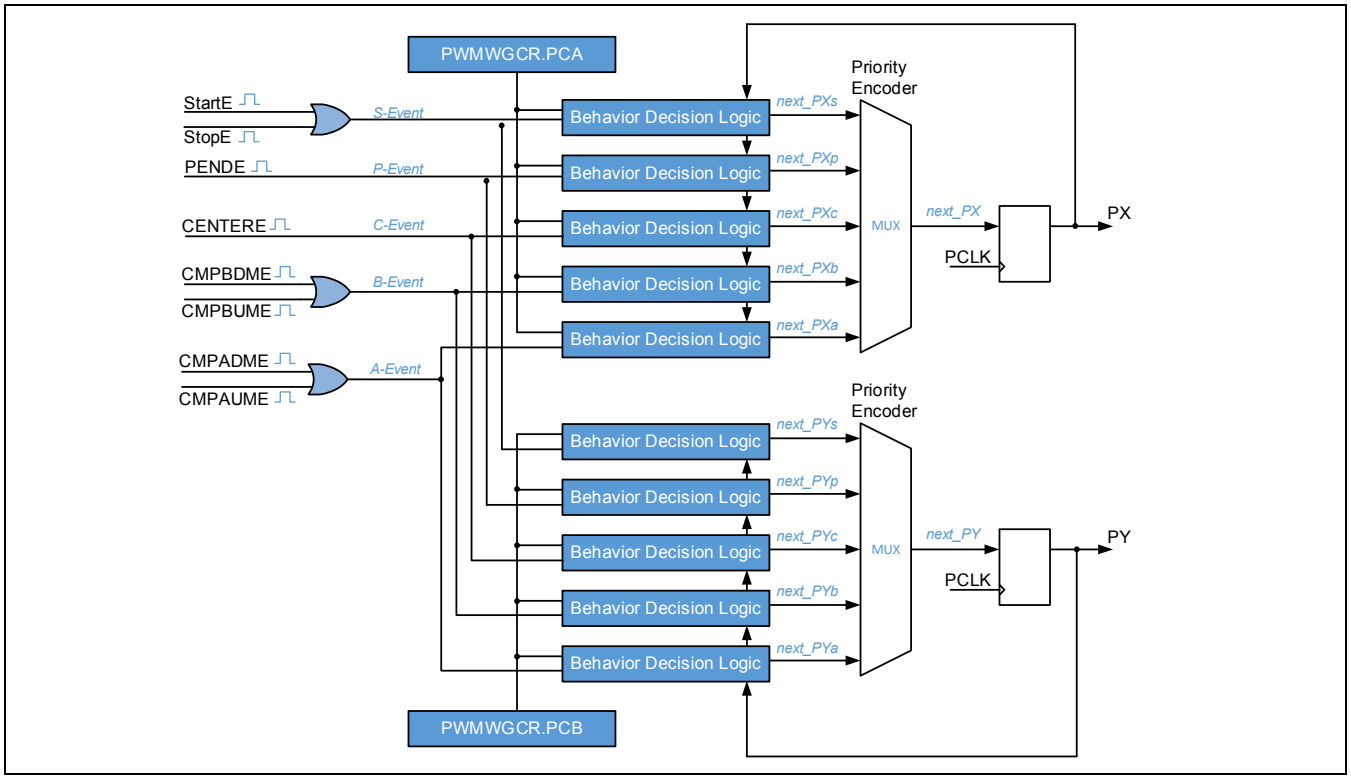


Figure 16-16 PWM波形生成引擎框图

16.2.7.2 EPWM引擎工作原理

EPWM引擎模块根据不同的事件产生内部PX和PY信号，PX和PY可以作为输出控制模块的信号源。根据EPWM_WGCR寄存器中PCX和PCY的设置，PX和PY的状态在5种不同的事件触发点可以作出改变，5种触发事件具有不同的优先级，当多个事件同时发生时，具有最高优先级的事件将作为决定因素来确定PX和PY的状态。

Table 16-3 事件的优先级

优先级	触发事件	可以触发该事件的信号
1	S-Event	StartE（软件使能，或者比较器触发）/ StopE （软件停止，或者Hard Lock）
2	P-Event	PendE（周期结束）
3	C-Event	CENTERE （递增递减计数中间点）
4	B-Event	CMPBDME 和 CMPBUME（计数值与CMPB相等）
5	A-Event	CMPADME 和 CMPAUME（计数值与CMPA相等）

NOTE: 优先级列数字越小，代表的优先级越高
EPWM_WGCR指定了在相应触发事件发生时，对PX和PY作出何种状态改变。

Table 16-4 PWM引擎各事件触发的动作设置

设定值	动作
0	没有动作 （保持原来状态）
1	PX 或 PY 变成低电平输出
2	PX 或 PY 变成高电平输出
3	PX 或 PY 翻转输出

下面给出几种基于不同计数模式，和引擎配置产生相应PX和PY信号的例子。

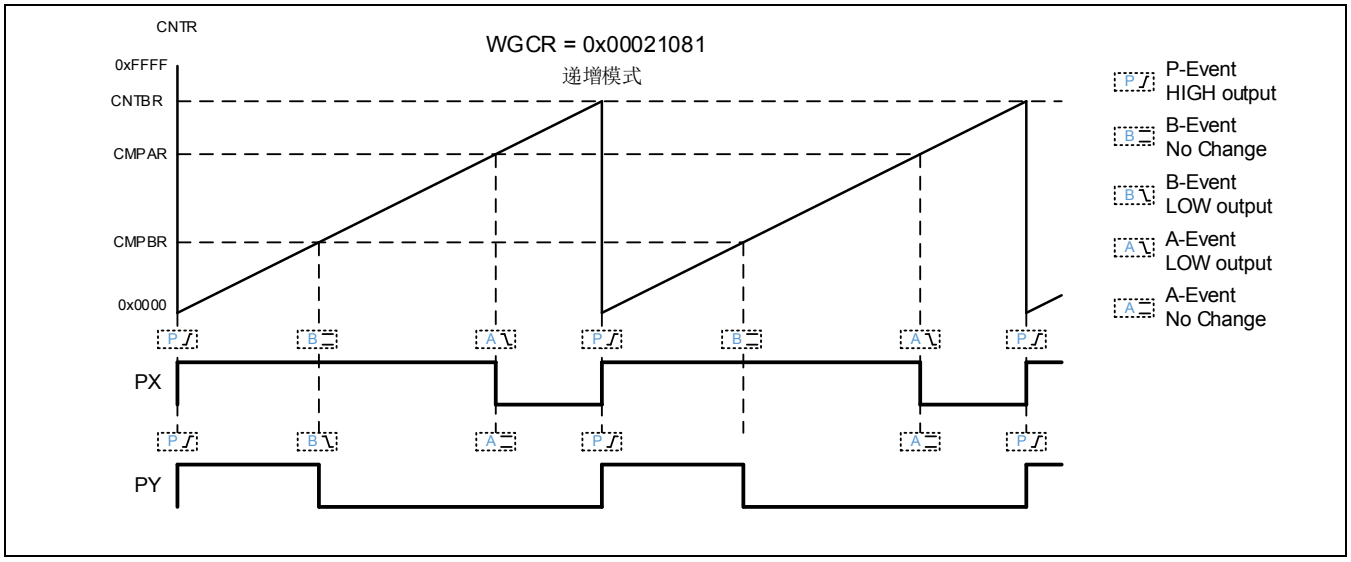


Figure 16-17 PWM 波形生成示例1

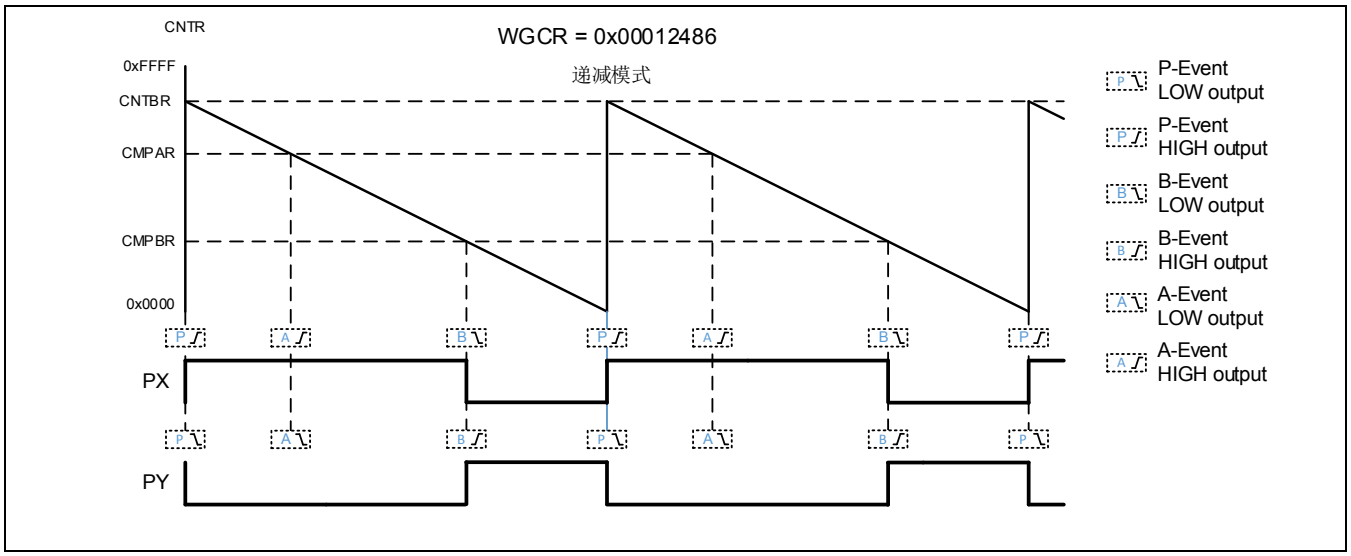


Figure 16-18 PWM 波形生成示例2

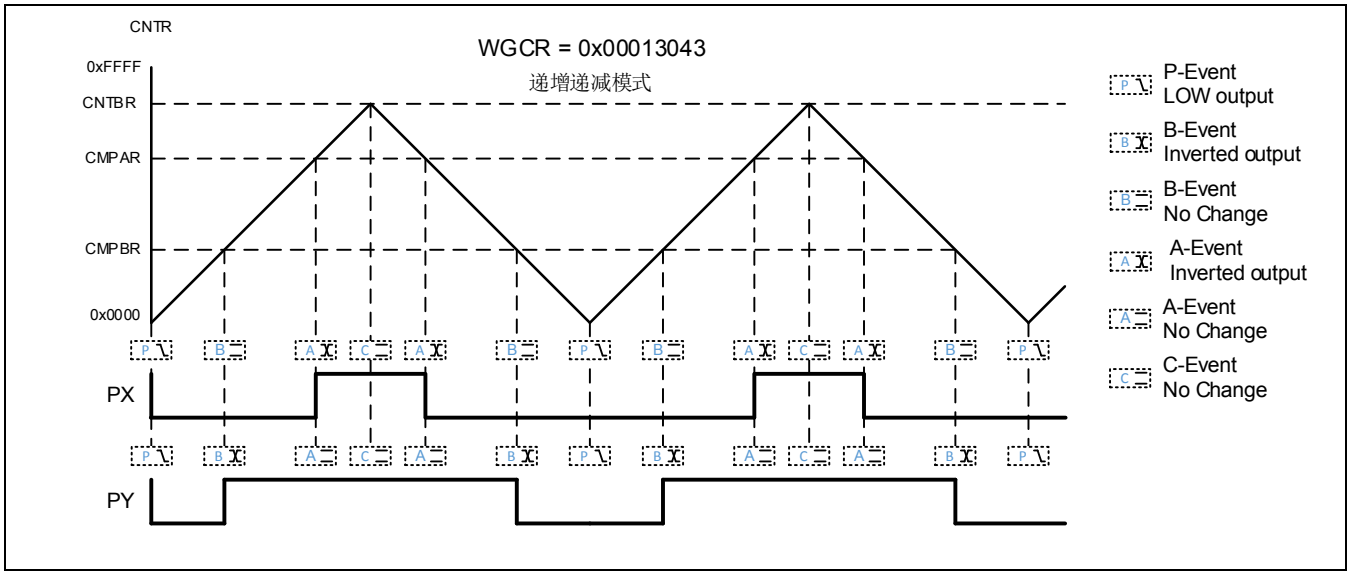


Figure 16-19 PWM 波形生成示例3

16.2.8 输出控制

EPWM模块具有3组共6个输出，每组有两个PWM信号输出端口，分别为EPWM_X和EPWM_Y。EPWM引擎生成的PX和PY信号可以作为输出控制模块的输入信号，经过输出模块的调整变成两路互补的EPWM信号，或者是间隔触发的信号从EPWM_X和EPWM_Y输出，PX和PY也可以跳过该输出控制模块，直接从EPWM_X和EPWM_Y输出。该设置可以通过EPWM_OUTCR来控制。同时EPWM模块还支持载波输出，在使能载波功能后，EPWM产生的波形为载波的包络，而且载波频率由PCLK分频产生。

Table 16-5 PWM的输出控制

EPWM_X	EPWM_Y	OUTSEL	SRCSEL
PX	PY	0, 3	X
PX互补模式的DX	PX互补模式的DY	1	0
PY互补模式的DX	PY互补模式的DY	1	1
PX间隔模式的SX	PX间隔模式的SY	2	0
PY间隔模式的SX	PY间隔模式的SY	2	1

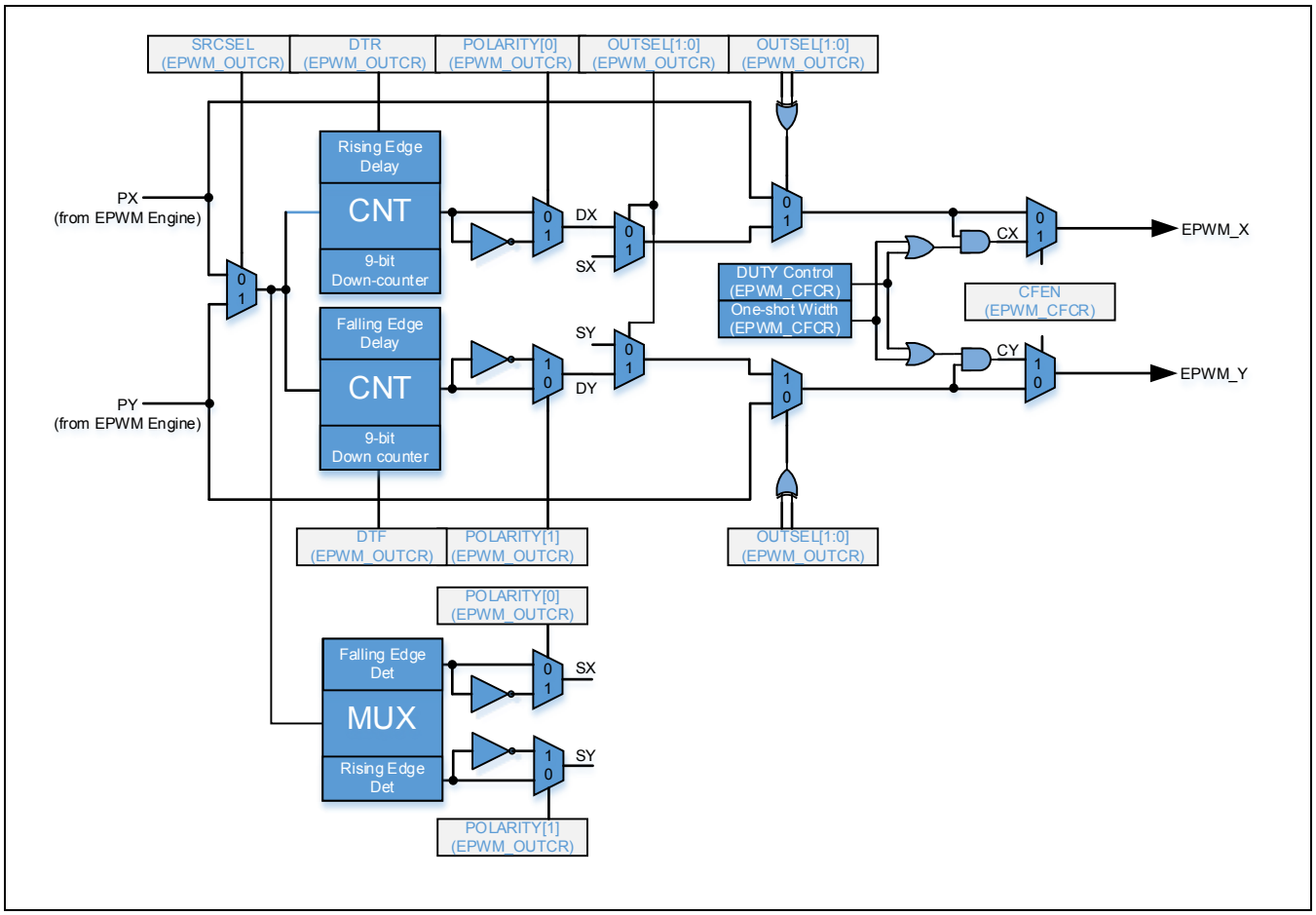


Figure 16-20 输出控制模块

16.2.8.1 互补输出模式及死区控制

互补输出模式通过对来自EPWM引擎的PX或者PY信号进行死区控制处理，产生一组具有互不相交跳变时间的同相或异相输出信号。死区时间由内部的一个9-bit递减计数器控制，该计数器的计数时钟为PWMCLK（PCLK分频后得到）。通过对信号上升沿和下降沿的delay控制，产生互不相交的死区时间。上升沿的delay时间可以通过EPWM_OUTCR.DTR来调节，下降沿的delay时间可以通过EPWM_OUTCR.DTF来调节。在delay产生以后，输出信号的极性可以通过POLARITY位来设置，两路输出可以分别设置不同的极性。在极性设置后的输出被定义为DX和DY信号，该信号通过OUTSEL的选择可以输出到最终的EPWM_X和EPWM_Y通道上。

死区控制的delay时间可以通过下面的公式计算得出：

$$RED = DTR \times (DIVM + 1) \times 2^{DIVN} \times PCLK$$
$$FED = DTF \times (DIVM + 1) \times 2^{DIVN} \times PCLK$$

其中：PCLK 代表 PWM模块输入系统时钟频率周期

Table 16-6 死区Delay时间（当PCLK=20MHz, uS）

DTR / DTF	DIVM=0, DIVN=0 (PCLK/1)	DIVM=1, DIVN=2 (PCLK/8)	DIVM=3, DIVN=4 (PCLK/64)
1	0.05	0.40	3.20
10	0.50	4.00	32.00
100	5.00	40.00	320.00
250	12.50	100.00	800.00
500	25.00	200.00	1600.00

在接下来的例子中，详细解释了几种不同case下的DX和DY输出。

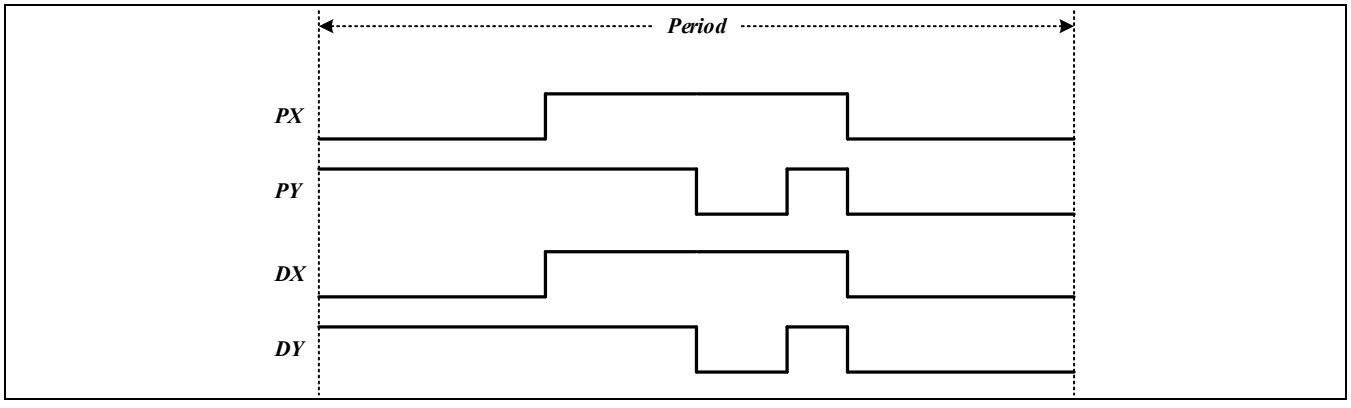


Figure 16-21 死区控制示例1：BYPASS 模式，PX和PY直接输出

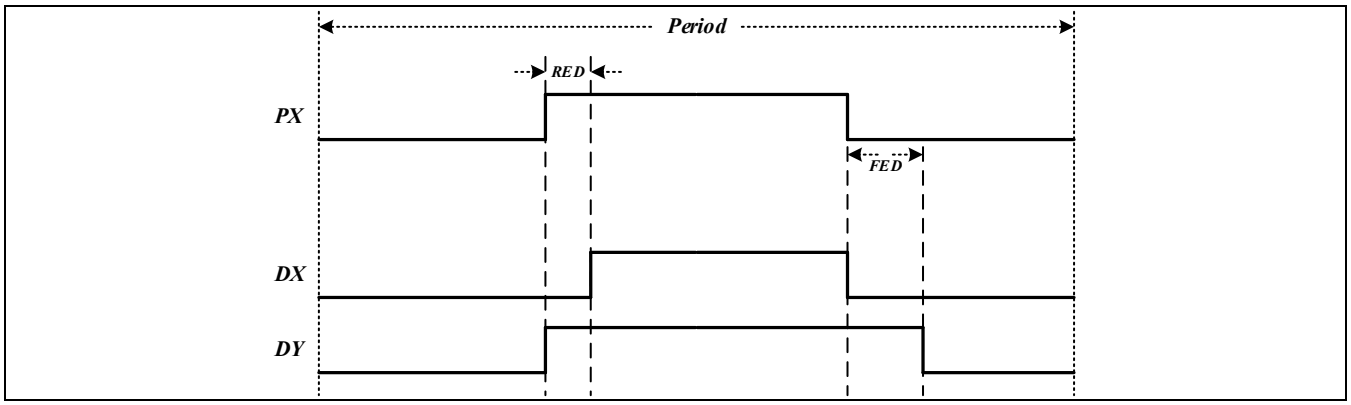


Figure 16-22 死区控制示例2：PX输入源，POLARITY=0

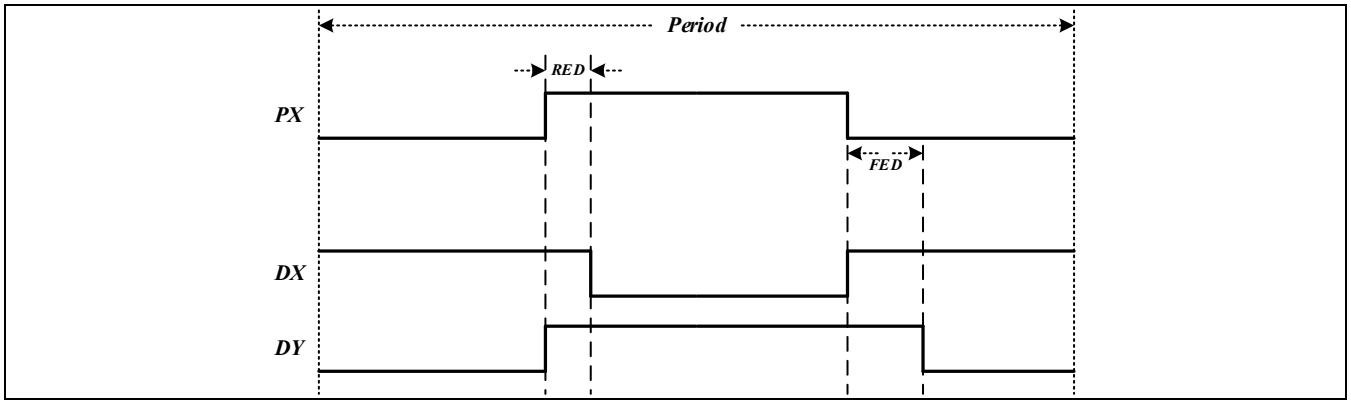


Figure 16-23 死区控制示例3：PX输入源，POLARITY=1

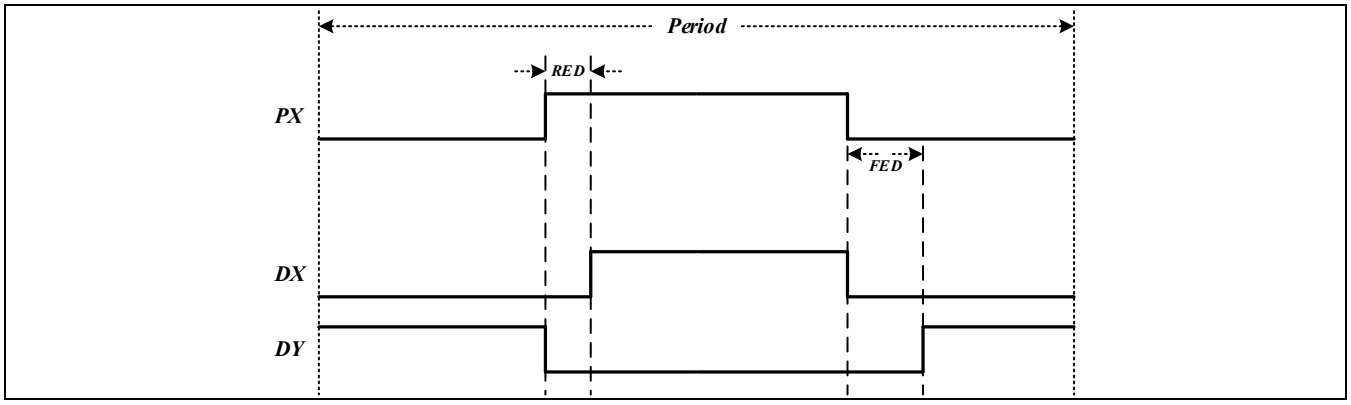


Figure 16-24 死区控制示例4：PX输入源，POLARITY=2

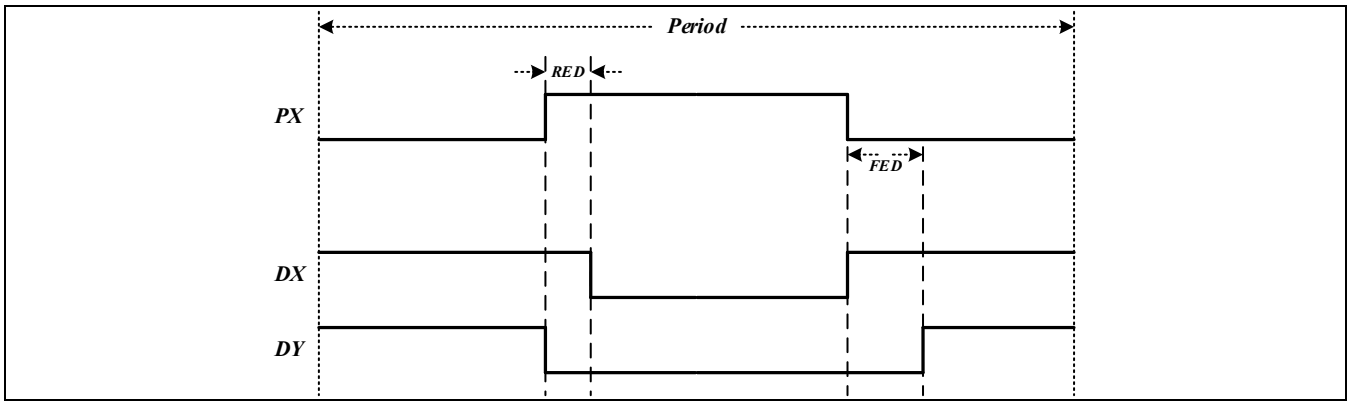


Figure 16-25 死区控制示例5：PX输入源，POLARITY=3

当PX的高电平持续时间小于DTR的设置值时，DX上的高电平输出将被滤去。

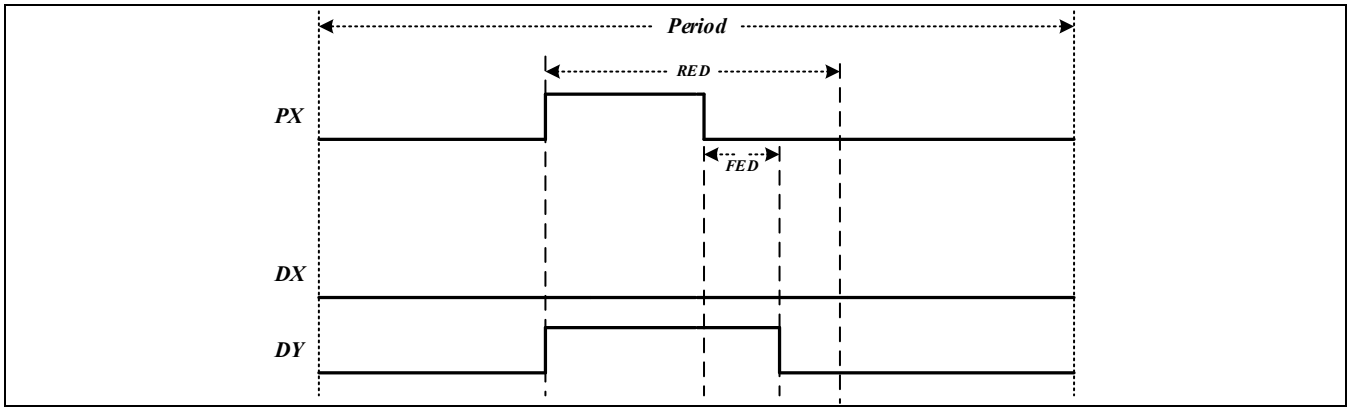


Figure 16-26 死区控制示例6：PX输入源，POLARITY=0

当PX的低电平持续时间小于DTF的设置值时，DX上的低电平输出将被滤去。

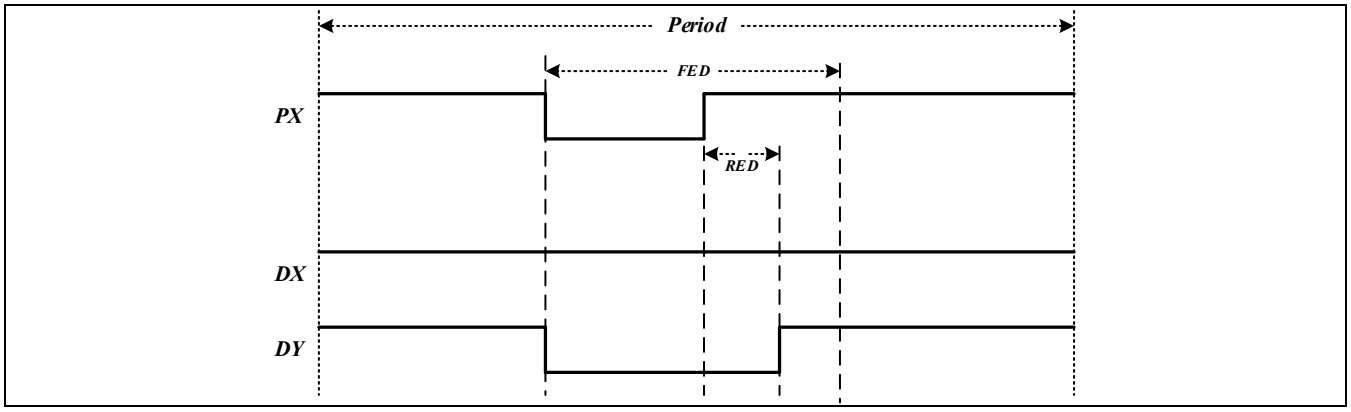


Figure 16-27 死区控制示例7：PX输入源，POLARITY=0

16.2.8.2 间隔触发输出模式

间隔触发模式是将PX或者PY信号的跳变状态间隔的分配到两路PWM输出上。由于边沿检测需要耗费一个系统时钟时间，所以输出的PWM信号会比原始信号（PX或PY）delay一个系统时钟周期时间。

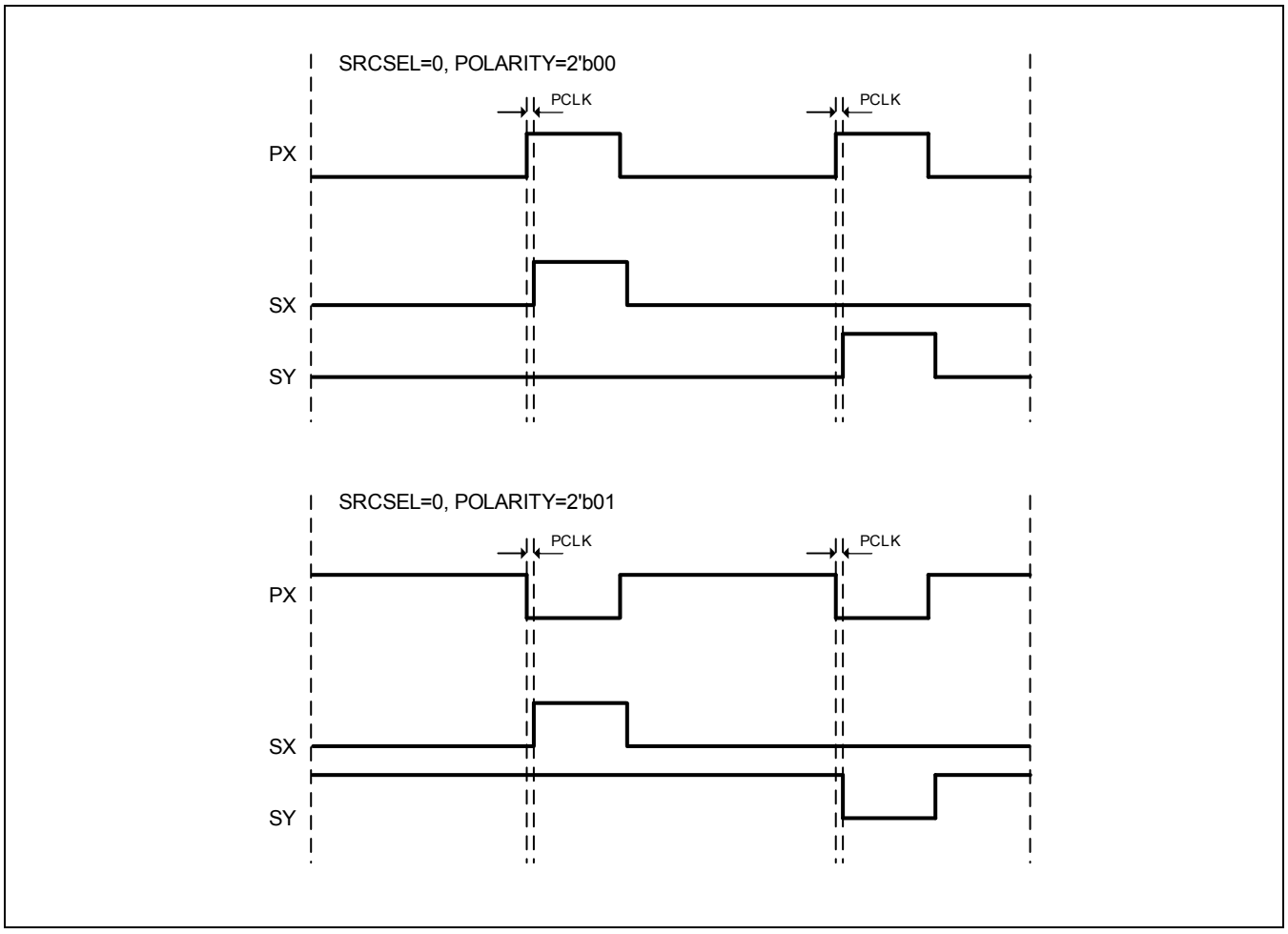


Figure 16-28 间隔触发输出模式

16.2.8.3 载波输出

EPWM支持载波输出功能，由互补输出或者间隔输出的信号作为包络，PCLK分频(EPWM_CFCR里的CDIV位)以及占空比控制(EPWM_CFCR里的DUTY位)输出的信号作为载波频率。

载波频率CFCLK由下面的公式计算：

CFCLK = PCLK / (CDIV+1) / 8

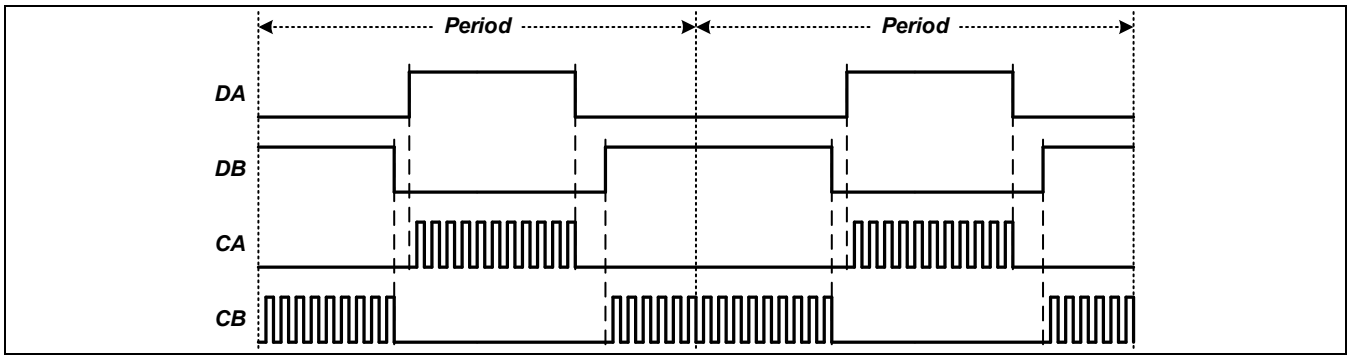


Figure 16-29 载波输出，基本波形 (EPWM_CFCR中OSW=0, CDIV=0)

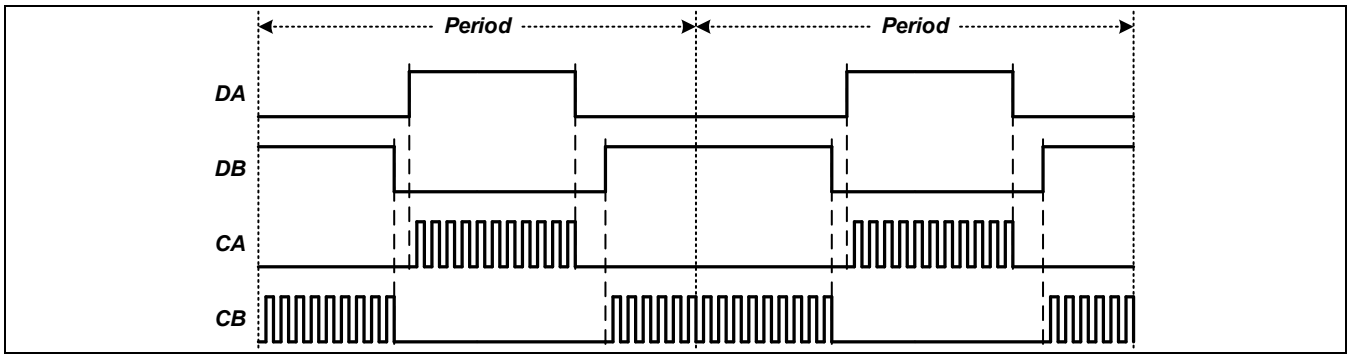


Figure 16-30 载波输出，单次扩展 (EPWM_CFCR中OSW=1, CDIV=0)

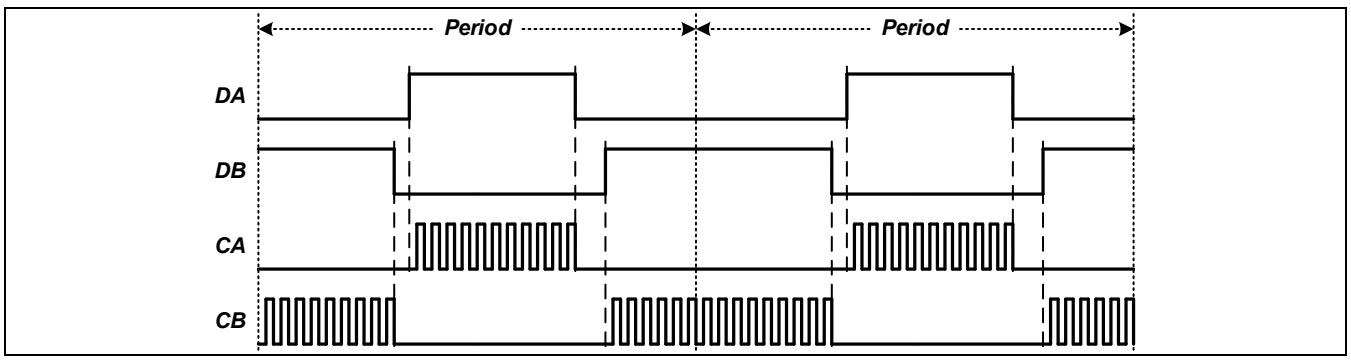


Figure 16-31 载波输出，单次扩展(EPWM_CFCR中OSW=2, CDIV=0)

载波输出功能支持在第一个载波周期扩展高电平输出，在这个扩展周期内，CDIV内部被强制设为0，CFCLK的公式变为：

CFCLK = PCLK / 8

(EPWM_CRCR中 OSW > 0 的情况下，即在扩展功能有效的情况下)

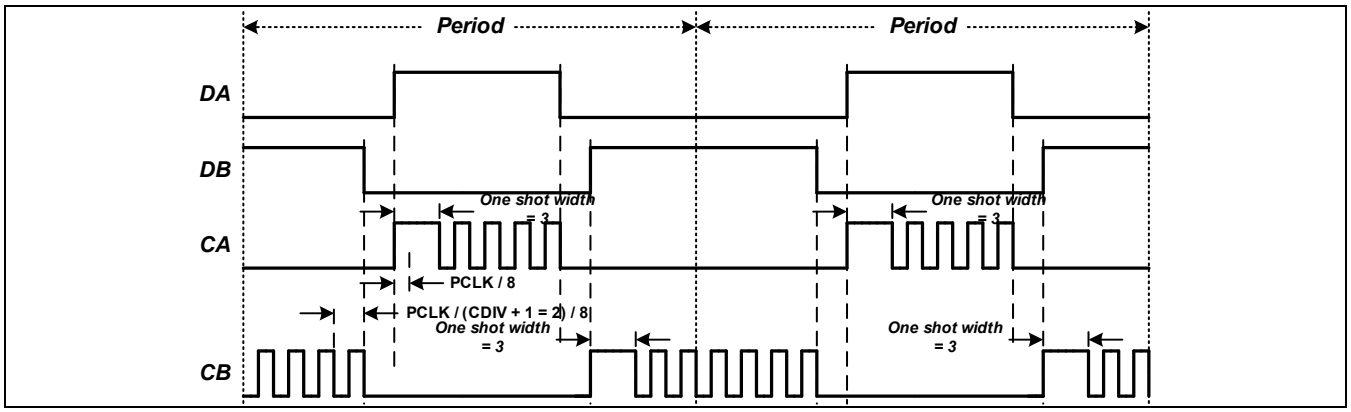


Figure 16-32 载波输出，单次扩展(EPWM_CFCR中OSW=3, CDIV=0)

Table 16-7 PCLK=40MHz时单次扩展脉冲宽度

OSW (hex)	Pulse Width (nS)
0	No expansion
1	200
2	400
3	600
4	800
5	1000
6	1200
7	1400
8	1600
9	1800
A	2000
B	2200
C	2400
D	2600
E	2800
F	3000

载波输出的占空比由EPWM_CR的DUTY位控制。当OSW大于0时，第一个脉冲的宽度不受DUTY位的影响。

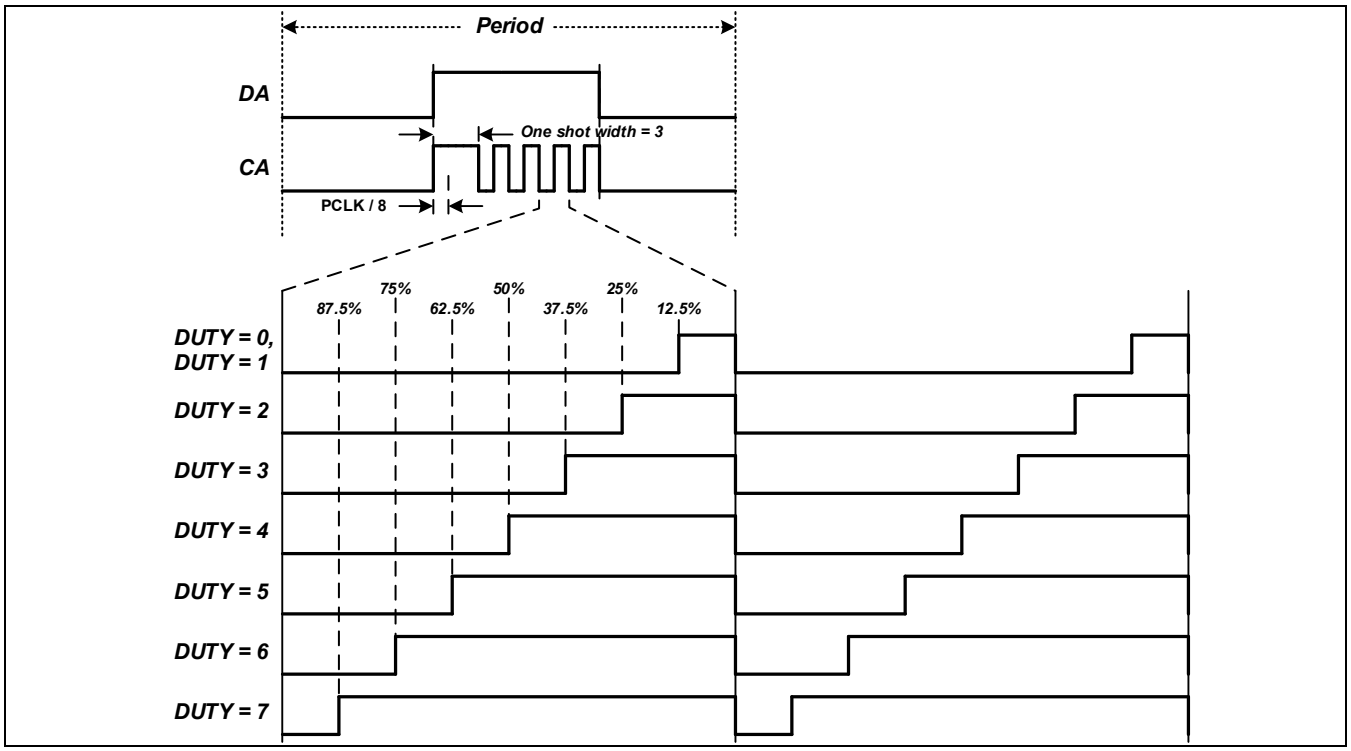


Figure 16-33 载波占空比控制

16.2.9 中断和触发

EPWM模块可以产生的中断如下表所示。

Table 16-8 中断列表

优先级	描述
STARTI0	PWM0 启动中断
STOPI0	PWM0 停止中断
PENDI0	PWM0 周期结束中断
CENTERI0	PWM0 计数器递增递减模式下，中间点匹配中断
STARTI1	PWM1 启动中断
STOPI1	PWM1 停止中断
PENDI11	PWM1 周期结束中断
CENTERI1	PWM1 计数器递增递减模式下，中间点匹配中断
STARTI2	PWM2 启动中断
STOPI2	PWM2 停止中断
PENDI2	PWM2 周期结束中断
CENTERI2	PWM2 计数器递增递减模式下，中间点匹配中断
PWM0_CMPAUMI	计数器递增阶段，当前计数器值和PWM0的CMPA匹配中断
PWM0_CMPADMI	计数器递减阶段，当前计数器值和PWM0的CMPA匹配中断
PWM0_CMPBUMI	计数器递增阶段，当前计数器值和PWM0的CMPB匹配中断
PWM0_CMPBDMI	计数器递减阶段，当前计数器值和PWM0的CMPB匹配中断
PWM1_CMPAUMI	计数器递增阶段，当前计数器值和PWM1的CMPA匹配中断
PWM1_CMPADMI	计数器递减阶段，当前计数器值和PWM1的CMPA匹配中断
PWM1_CMPBUMI	计数器递增阶段，当前计数器值和PWM1的CMPB匹配中断
PWM1_CMPBDMI	计数器递减阶段，当前计数器值和PWM1的CMPB匹配中断
PWM2_CMPAUMI	计数器递增阶段，当前计数器值和PWM2的CMPA匹配中断
PWM2_CMPADMI	计数器递减阶段，当前计数器值和PWM2的CMPA匹配中断
PWM2_CMPBUMI	计数器递增阶段，当前计数器值和PWM2的CMPB匹配中断
PWM2_CMPBDMI	计数器递减阶段，当前计数器值和PWM2的CMPB匹配中断
PWM0_SLPA_OVF	PWM0的SLPCMPAR自动调节溢出中断
PWM0_SLPB_OVF	PWM0的SLPCMPBR自动调节溢出中断
PWM1_SLPA_OVF	PWM1的SLPCMPAR自动调节溢出中断
PWM1_SLPB_OVF	PWM1的SLPCMPBR自动调节溢出中断

EPWM模块产生的事件同时可以作为其它模块的触发源，参考EPWM_EXTRGx寄存器的设置。

16.3 寄存器说明

16.3.1 寄存器表

- Base Address: 0x4005_4000

Register	Offset	Description	Reset Value
EPWM_CR	0x00	PWM 控制寄存器	0x0080_0000
EPWM_LKCR	0x04	PWM 联动控制寄存器	0x0000_0000
EPWM_LKTRG	0x08	PWM 联动延时以及防误触控制寄存器	0x0000_0000
EPWM_CNTR0	0x0C	PWM0 计数器寄存器	0x0000_0000
EPWM_CNTBR0	0x10	PWM0 计数器基数值寄存器	0x0000_0000
EPWM_PCNTR0	0x14	PWM0 计数器基数值暂存寄存器	0x0000_0000
EPWM_SLPCNTR0	0x18	PWM0 软锁止基数值暂存寄存器	0x0000_0000
EPWM_CNTR1	0x1C	PWM1 计数器寄存器	0x0000_0000
EPWM_CNTBR1	0x20	PWM1 计数器基数值寄存器	0x0000_0000
EPWM_PCNTR1	0x24	PWM1 计数器基数值暂存寄存器	0x0000_0000
EPWM_SLPCNTR1	0x28	PWM1 软锁止基数值暂存寄存器	0x0000_0000
EPWM_CNTR2	0x2C	PWM2 计数器寄存器	0x0000_0000
EPWM_CNTBR2	0x30	PWM2 计数器基数值寄存器	0x0000_0000
EPWM_PCNTR2	0x34	PWM2 计数器基数值暂存寄存器	0x0000_0000
EPWM_SLPCNTR2	0x38	PWM2 软锁止基数值暂存寄存器	0x0000_0000
EPWM_CMPAR0	0x3C	PWM0 比较值A寄存器	0x0000_0000
EPWM_PCMPAR0	0x40	PWM0 比较值A暂存寄存器	0x0000_0000
EPWM_SLPCMPAR0	0x44	PWM0 软锁止比较值A暂存寄存器	0x0000_0000
EPWM_CMPBR0	0x48	PWM0 比较值B寄存器	0x0000_0000
EPWM_PCMPBR0	0x4C	PWM0 比较值B暂存寄存器	0x0000_0000
EPWM_SLPCMPBR0	0x50	PWM0 软锁止比较值B暂存寄存器	0x0000_0000
EPWM_CMPAR1	0x54	PWM1 比较值A寄存器	0x0000_0000
EPWM_PCMPAR1	0x58	PWM1 比较值A暂存寄存器	0x0000_0000
EPWM_SLPCMPAR1	0x5C	PWM1 软锁止比较值A暂存寄存器	0x0000_0000
EPWM_CMPBR1	0x60	PWM1 比较值B寄存器	0x0000_0000
EPWM_PCMPBR1	0x64	PWM1 比较值B暂存寄存器	0x0000_0000
EPWM_SLPCMPBR1	0x68	PWM1 软锁止比较值B暂存寄存器	0x0000_0000
EPWM_CMPAR2	0x6C	PWM2 比较值A寄存器	0x0000_0000
EPWM_PCMPAR2	0x70	PWM2 比较值A暂存寄存器	0x0000_0000
EPWM_SLPCMPAR2	0x74	PWM2 软锁止比较值A暂存寄存器	0x0000_0000
EPWM_CMPBR2	0x78	PWM2 比较值B寄存器	0x0000_0000

EPWM_PCMPBR2	0x7C	PWM2 比较值B暂存寄存器	0x0000_0000
EPWM_SLPCMPBR2	0x80	PWM2 软锁止比较值B暂存寄存器	0x0000_0000
EPWM_WGCR0	0x84	PWM0 波形生成控制寄存器	0x0000_0000
EPWM_WGCR1	0x88	PWM1 波形生成控制寄存器	0x0000_0000
EPWM_WGCR2	0x8C	PWM2 波形生成控制寄存器	0x0000_0000
EPWM_OUTCR0	0x90	PWM0 输出控制寄存器	0x0000_0000
EPWM_OUTCR1	0x94	PWM1 输出控制寄存器	0x0000_0000
EPWM_OUTCR2	0x98	PWM2 输出控制寄存器	0x0000_0000
EPWM_CFCR0	0x9C	PWM0 载波频率控制寄存器	0x0000_0000
EPWM_CFCR1	0xA0	PWM1 载波频率控制寄存器	0x0000_0000
EPWM_CFCR2	0xA4	PWM2 载波频率控制寄存器	0x0000_0000
EPWM_EMR	0xA8	PWM 紧急模式控制寄存器	0x0000_0000
EPWM_SLCON	0xAC	PWM 软锁止控制寄存器	0x0000_0000
EPWM_SLSTEP0	0xB0	PWM0 软锁止自增自减步长寄存器	0x0000_0000
EPWM_SLSTEP1	0xB4	PWM1 软锁止自增自减步长寄存器	0x0000_0000
EPWM_IER	0xB8	PWM 中断使能寄存器	0x0000_0000
EPWM_ICR	0xBC	PWM 中断状态清除寄存器	0x0000_0000
EPWM_RISR	0xC0	PWM 中断原始状态寄存器	0x0000_0000
EPWM_MISR	0xC4	PWM 中断状态寄存器	0x0000_0000
EPWM_EXTRG0	0xC8	PWM 输出至外部触发源选择寄存器0	0x0000_0000
EPWM_EXTRG1	0xCC	PWM 输出至外部触发源选择寄存器1	0x0000_0000

16.3.2 EPWM_CR (PWM控制寄存器)

- Address = Base Address + 0x0000, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBGEN		STOP2	STOP1	STOP0	START2	START1	START0	SINGLE	CLKEN	OVFSTB	BUSY	DIVM												DIVN			CMODE		SWRST	STOP	START
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	W	W	W

Name	Bit	Type	Description	Reset Value
START	[0]	W	软件启动 (全局)	0x0
STOP	[1]	W	软件停止 (全局)	0x0
SWRST	[2]	W	软件复位	0x0
CMODE	[4:3]	RW	计数模式 0: 递增模式 1：递减模式 2：递增递减模式 3：递减递增模式	0x0
DIVN	[7:5]	RW	PWMCLK的分频系数N Divided Clock = PCLK / 2 ^{DIVN}	0x0
DIVM	[19:8]	RW	PWMCLK的分频系数M Divided Clock = PCLK / (DIVM + 1)	0x0
BUSY	[20]	RW	BUSY 标识位 0： 计数器未工作 1： 计数器正在工作	0x0
OVFSTB	[21]	RW	OVERFLOW 后，PWM 是否重新开始计数 0： PWM在计数器溢出以后自动重新开始计数 1： PWM在计数器溢出以后停止计数	0x0
CLKEN	[22]	RW	时钟使能控制 0： 计数时钟不使能 1： 计数时钟使能	0x0
SINGLE	[23]	RW	计数器控制 0： 3个通道分别使用3个不同的计数器 1： 3个通道使用同一个计数器(计数器0)	0x1
START0	[24]	W	启动计数器0	0x0

START1	[25]	W	启动计数器1 (必须SINGLE=0，否则无效)	0x0
START2	[26]	W	启动计数器2 (必须SINGLE=0，否则无效)	0x0
STOP0	[27]	W	停止计数器0	0x0
STOP1	[28]	W	停止计数器1 (必须SINGLE=0，否则无效)	0x0
STOP2	[29]	W	停止计数器2 (必须SINGLE=0，否则无效)	0x0
DBGEN	[31:30]	RW	调试使能 0xA: 进入调试模式后，PWM X/Y都输出0 0x5: 进入调试模式后，PWM X/Y都输出1 0xC: 进入调试模式后，PWM都输出z 其它: 关闭调试功能，PWM仍然输出	0x0

16.3.3 EPWM_LKCR (PWM联动控制寄存器)

- Address = Base Address + 0x0004, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD							EP4LKM	EP3LKM			EP2LKM			EP1LKM		EP0LKM			CMP4LKM			CMP3LKM			CMP2LKM			CMP1LKM			CMP0LKM	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
CMP0LKM	[2:0]	RW	比较器0的联动模式 000： 禁止联动模式 010： 正常启动触发 011： 延时的启动触发 110： 选择为软锁止的触发源 111： 选择为硬锁止的触发源 Other: 保留（禁止联动模式）	0x0
CMP1LKM	[5:3]	RW	比较器1的联动模式 000： 禁止联动模式 010： 正常启动触发 011： 延时的启动触发 110： 选择为软锁止的触发源 111： 选择为硬锁止的触发源 Other: 保留（禁止联动模式）	0x0
CMP2LKM	[7:6]	RW	比较器2的联动模式 0x：禁止联动模式 10：选择为软锁止的触发源 11：选择为硬锁止的触发源	0x0
CMP3LKM	[9:8]	RW	比较器3的联动模式 0x：禁止联动模式 10：选择为软锁止的触发源 11：选择为硬锁止的触发源	0x0
CMP4LKM	[11:10]	RW	比较器4的联动模式 00：禁止联动模式	0x0

			01：无效选择(勿使用) 10：选择为软锁止的触发源 11：选择为硬锁止的触发源	
EP0LKM	[14:12]	RW	外部中断0的联动模式 000： 禁止联动模式 010： 正常启动触发 011： 延时的启动触发 110： 选择为软锁止的触发源 111： 选择为硬锁止的触发源 Other: 保留（禁止联动模式）	0x0
EP1LKM	[17:15]	RW	外部中断1的联动模式 000： 禁止联动模式 010： 正常启动触发 011： 延时的启动触发 110： 选择为软锁止的触发源 111： 选择为硬锁止的触发源 Other: 保留（禁止联动模式）	0x0
EP2LKM	[20:18]	RW	外部中断2的联动模式 000： 禁止联动模式 010： 正常启动触发 011： 延时的启动触发 110： 选择为软锁止的触发源 111： 选择为硬锁止的触发源 Other: 保留（禁止联动模式）	0x0
EP3LKM	[23:21]	RW	外部中断3的联动模式 000： 禁止联动模式 010： 正常启动触发 011： 延时的启动触发 110： 选择为软锁止的触发源 111： 选择为硬锁止的触发源 Other: 保留（禁止联动模式）	0x0
EP4LKM	[25:24]	RW	外部中断4的联动模式 0x：禁止联动模式 10：选择为软锁止的触发源 11：选择为硬锁止的触发源	0x0

16.3.4 EPWM_LKTRG (PWM联动延时以及防误触控制寄存器)

- Address = Base Address + 0x0008, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	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								RSVD												TRGTDL				TRGIVT							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
TRGIVT	[7:0]	RW	防误触发时间间隔 同一个比较器两次触发间的最小时间间隔，在这个时间区间内，任何触发都被禁止。 IVT = TRGIVT x 4 x Tpwmclk 当TRGIVT等于0时，防误触发功能被禁止	0x0
TRGTDL	[11:8]	RW	触发延时控制 当比较器触发计数器时，计数器延时一定时间以后再开始工作。 TDL = (TRGTDL+1) x 4 x Tpwmclk TRGTDL只能设置成 0 ~ 14 中的值。	0x0
KEY	[31:24]	W	防误写KEY寄存器 必须写0xA5，写操作才有效	-

注意：TRGTDL位的设置范围为 0 ~ 14。

16.3.5 EPWM_CNTRx（PWMx计数器寄存器）

- EPWM_CNTR0 : Address = Base Address + 0x000C, Reset Value = 0x0000_0000
- EPWM_CNTR1 : Address = Base Address + 0x001C, Reset Value = 0x0000_0000
- EPWM_CNTR2 : Address = Base Address + 0x002C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	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								SLOCK_CLR	RSVD							CNT															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W	W	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description	Reset Value
CNT	[15:0]	RW	写：计数器周期值 读：当前计数器的计数值	0x0
SLOCK_CLR	[23]	W	清除Soft Lock状态	-
KEY	[31:24]	W	防误写KEY寄存器 必须写0xA5，写操作才有效	-

16.3.6 EPWM_CNTBRx (PWMx计数器基数值寄存器)

- EPWM_CNTBR0 : Address = Base Address + 0x0010, Reset Value = 0x0000_0000
- EPWM_CNTBR1 : Address = Base Address + 0x0020, Reset Value = 0x0000_0000
- EPWM_CNTBR2 : Address = Base Address + 0x0030, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
CNTB	[15:0]	R	计数器当前基数值（周期值）	0x0

16.3.7 EPWM_PCNTRx (PWMx计数器基数值暂存寄存器)

- EPWM_PCNTR0 : Address = Base Address + 0x0014, Reset Value = 0x0000_0000
- EPWM_PCNTR1 : Address = Base Address + 0x0024, Reset Value = 0x0000_0000
- EPWM_PCNTR2 : Address = Base Address + 0x0034, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
PCNT	[15:0]	R	计数器基数（周期）设置暂存值	0x0

16.3.8 EPWM_SLPCNTRx (PWMx软锁止基数值暂存寄存器)

- EPWM_SLPCNTR0 : Address = Base Address + 0x0018, Reset Value = 0x0000_0000
- EPWM_SLPCNTR1 : Address = Base Address + 0x0028, Reset Value = 0x0000_0000
- EPWM_SLPCNTR2 : Address = Base Address + 0x0038, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	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								RSVD								SLPCNT															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
SLPCNT	[15:0]	RW	软锁止时，计数器基数（周期）设置暂存值	0x0
KEY	[31:24]	W	防误写KEY寄存器 必须写0xA5，写操作才有效	-

注意：只有计数器 0 支持软锁止功能，也就是只有在 EPWM_CR 的 SINGLE 位为 1 的时候，才可以使用自动更新功能。

16.3.9 EPWM_CMPARx（PWMx比较值A寄存器）

- EPWM_CMPAR0 : Address = Base Address + 0x003C, Reset Value = 0x0000_0000
- EPWM_CMPAR1 : Address = Base Address + 0x0054, Reset Value = 0x0000_0000
- EPWM_CMPAR2 : Address = Base Address + 0x006C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	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								SLOCK_CLR	RSVD							CMPA															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W	W	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description	Reset Value
CMPA	[15:0]	R	当前比较寄存器A的值	0x0
SLOCK_CLR	[23]	W	清除Soft Lock状态	-
KEY	[31:24]	W	防误写KEY寄存器 必须写0xA5，写操作才有效	-

16.3.10 EPWM_PCMPARx (PWMx比较值A暂存寄存器)

- EPWM_PCMPAR0 : Address = Base Address + 0x0040, Reset Value = 0x0000_0000
- EPWM_PCMPAR1 : Address = Base Address + 0x0058, Reset Value = 0x0000_0000
- EPWM_PCMPAR2 : Address = Base Address + 0x0070, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
PCMPA	[15:0]	R	当前比较寄存器A的设置暂存值	0x0

16.3.11 EPWM_SLPCMPARx (PWMx软锁止比较值A暂存寄存器)

- EPWM_SLPCMPAR0 : Address = Base Address + 0x0044, Reset Value = 0x0000_0000
- EPWM_SLPCMPAR1 : Address = Base Address + 0x005C, Reset Value = 0x0000_0000
- EPWM_SLPCMPAR2 : Address = Base Address + 0x0074, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	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								RSVD								SLPCMPA															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
SLPCMPA	[15:0]	RW	软锁止时，比较寄存器A的设置暂存值	0x0
KEY	[31:24]	W	防误写KEY寄存器 必须写0xA5，写操作才有效	-

16.3.12 EPWM_CMPBRx (PWMx比较值B寄存器)

- EPWM_CMPBR0 : Address = Base Address + 0x0048, Reset Value = 0x0000_0000
- EPWM_CMPBR1 : Address = Base Address + 0x0060, Reset Value = 0x0000_0000
- EPWM_CMPBR2 : Address = Base Address + 0x0078, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	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								SLOCK_CLR	RSVD							CMPB															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W	W	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description	Reset Value
CMPB	[15:0]	R	当前比较寄存器B的值	0x0
SLOCK_CLR	[23]	W	清除Soft Lock状态	-
KEY	[31:24]	W	防误写KEY寄存器 必须写0xA5，写操作才有效	-

16.3.13 EPWM_PCMPBRx (PWMx比较值B暂存寄存器)

- EPWM_PCMPBR0 : Address = Base Address + 0x004C, Reset Value = 0x0000_0000
- EPWM_PCMPBR1 : Address = Base Address + 0x0064, Reset Value = 0x0000_0000
- EPWM_PCMPBR2 : Address = Base Address + 0x007C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
PCMPB	[15:0]	R	当前比较寄存器B的设置暂存值	0x0

16.3.14 EPWM_SLPCMPBRx (PWMx软锁止比较值B暂存寄存器)

- EPWM_SLPCMPBR0 : Address = Base Address + 0x0050, Reset Value = 0x0000_0000
- EPWM_SLPCMPBR1 : Address = Base Address + 0x0068, Reset Value = 0x0000_0000
- EPWM_SLPCMPBR2 : Address = Base Address + 0x0080, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	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								RSVD								SLPCMPB															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
SLPCMPB	[15:0]	RW	软锁止时，比较寄存器B的设置暂存值	0x0
KEY	[31:24]	W	防误写KEY寄存器 必须写0xA5，写操作才有效	-

16.3.15 EPWM_WGCRx (PWMx波形生成控制寄存器)

- EPWM_WGCR0 Address = Base Address + 0x0084, Reset Value = 0x0000_0000
- EPWM_WGCR1 Address = Base Address + 0x0088, Reset Value = 0x0000_0000
- EPWM_WGCR2 Address = Base Address + 0x008C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
PCX	[9:0]	RW	PX信号生成控制位 [9:8]： S事件触发控制 [7:6]： P事件触发控制 [5:4]： C事件触发控制 [3:2]： B事件触发控制 [1:0]： A事件触发控制 在所有的事件触发中，遵循如下优先级设置： S事件 > P事件 > C事件 > B事件 > A事件 控制位配置值： 0： 不变化 1： PX变成LOW输出 2： PX变成HIGH输出 3： PX取反输出	0x0
PCY	[19:10]	RW	PY信号生成控制位 [19:18]： S事件触发控制 [17:16]： P事件触发控制 [15:14]： C事件触发控制 [13:12]： B事件触发控制 [11:10]： A事件触发控制 控制位配置值： 0： 不变化 1： PY变成LOW输出 2： PY变成HIGH输出 3： PY取反输出	0x0

16.3.16 EPWM_OUTCRx (PWMx输出控制寄存器)

- EPWM_OUTCR0 Address = Base Address + 0x0090, Reset Value = 0x0000_0000
- EPWM_OUTCR1 Address = Base Address + 0x0094, Reset Value = 0x0000_0000
- EPWM_OUTCR2 Address = Base Address + 0x0098, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								DTF								DTR											SRCSEL	POLARITY		OUTSEL	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
OUTSEL	[1:0]	RW	输出选择位 0： 跳过输出控制模块，PX和PY直接输出 1： 互补输出 2： 间隔触发输出 3： 跳过输出控制模块，PX和PY直接输出	0x0
POLARITY	[3:2]	RW	输出极性控制位。 bit2: EPWM_X的极性控制 bit3: EPWM_Y的极性控制 0： 不改变极性 1： 反向	0x0
SRCSEL	[4]	RW	输出控制模块输入信号选择位 0： PX作为输入信号 1： PY作为输入信号	0x0
DTR	[14:6]	RW	上升沿死区时间延时计数值	0x0
DTF	[23:15]	RW	下降沿死区时间延时计数值	0x0

16.3.17 EPWM_CFCRx (PWMx载波频率控制寄存器)

- EPWM_CFCR0 Address = Base Address + 0x009C, Reset Value = 0x0000_0000
- EPWM_CFCR1 Address = Base Address + 0x00A0, Reset Value = 0x0000_0000
- EPWM_CFCR2 Address = Base Address + 0x00A4, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
CFEN	[0]	RW	载波控制 0： 禁止载波输出功能 1： 使能载波输出功能	0x0
OSW	[7:4]	RW	单次脉冲输出宽度控制 输出的宽度由下面的公式计算。 宽度 = OSW x (PCLK周期) x 8 OSW等于0相当于禁止单次脉冲输出。	0x0
CDIV	[10:8]	RW	载波频率的分频设置 0： PCLK / 8 1： PCLK / 16 2： PCLK / 24 3： PCLK / 32 4： PCLK / 40 5： PCLK / 48 6： PCLK / 56 7： PCLK / 64	0x0
DUTY	[14:12]	RW	载波占空比控制 0： 1 / 8 占空比 1： 1 / 8 占空比 2： 2 / 8 占空比 3： 3 / 8 占空比 4： 4 / 8 占空比 5： 5 / 8 占空比	0x0

			6： 6 / 8 占空比 7： 7 / 8 占空比	

16.3.18 EPWM_EMR (PWM紧急模式控制寄存器)

- Address = Base Address + 0x00A8, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD						SL_P2YS		SL_P2XS		HL_P2YS		HL_P2XS		SL_P1YS		SL_P1XS		HL_P1YS		HL_P1XS		SL_P0YS		SL_P0XS		HL_P0YS		HL_P0XS		SLOCK	HLOCK
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
HLOCK	[0]	RW	硬锁止指示位 当读操作时： 返回当前硬锁止状态 当写操作时： 0： 无效 1： 清除当前硬锁止状态	0x0
SLOCK	[1]	RW	软锁止指示位 当读操作时： 返回当前软锁止状态 当写操作时： 0： 无效 1： 清除当前软锁止状态	0x0
HL_P0XS	[3:2]	RW	硬锁止时，PWM0_X的状态设置 0： 输出LOW 1： 输出HIGH 2： 输出高阻 3： 保持上一个状态	0x0
HL_P0YS	[5:4]	RW	硬锁止时，PWM0_Y的状态设置 0： 输出LOW 1： 输出HIGH 2： 输出高阻 3： 保持上一个状态	0x0
SL_P0XS	[7:6]	RW	软锁止时，PWM0_X的状态设置 0： 输出LOW 1： 输出HIGH 2： 输出高阻 3： 保持上一个状态	0x0
SL_P0YS	[9:8]	RW	软锁止时，PWM0_Y的状态设置 0： 输出LOW 1： 输出HIGH 2： 输出高阻	0x0

			3： 保持上一个状态	
HL_P1XS	[11:10]	RW	硬锁止时，PWM1_X的状态设置 0： 输出LOW 1： 输出HIGH 2： 输出高阻 3： 保持上一个状态	0x0
HL_P1YS	[13:12]	RW	硬锁止时，PWM1_Y的状态设置 0： 输出LOW 1： 输出HIGH 2： 输出高阻 3： 保持上一个状态	0x0
SL_P1XS	[15:14]	RW	软锁止时，PWM1_X的状态设置 0： 输出LOW 1： 输出HIGH 2： 输出高阻 3： 保持上一个状态	0x0
SL_P1YS	[17:16]	RW	软锁止时，PWM1_Y的状态设置 0： 输出LOW 1： 输出HIGH 2： 输出高阻 3： 保持上一个状态	0x0
HL_P2XS	[19:18]	RW	硬锁止时，PWM2_X的状态设置 0： 输出LOW 1： 输出HIGH 2： 输出高阻 3： 保持上一个状态	0x0
HL_P2YS	[21:20]	RW	硬锁止时，PWM2_Y的状态设置 0： 输出LOW 1： 输出HIGH 2： 输出高阻 3： 保持上一个状态	0x0
SL_P2XS	[23:22]	RW	软锁止时，PWM2_X的状态设置 0： 输出LOW 1： 输出HIGH 2： 输出高阻 3： 保持上一个状态	0x0
SL_P2YS	[25:24]	RW	软锁止时，PWM2_Y的状态设置 0： 输出LOW 1： 输出HIGH 2： 输出高阻 3： 保持上一个状态	0x0

16.3.19 EPWM_SLCON (PWM软锁止控制寄存器)

- Address = Base Address + 0x00AC, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD							S_ONE	RSVD														SL_CNTR_INC_EN	SL_CNTR_DEC_EN	SL_INCB_EN1	SL_DECB_EN1	SL_INCA_EN1	SL_DECA_EN1	SL_INCB_EN0	SL_DECB_EN0	SL_INCA_EN0	SL_DECA_EN0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
SL_DECA_EN0	[0]	RW	0：禁止PWM0在SLOCK状态下的SLPCMPAR比较寄存器自动减少功能 1：使能PWM0在SLOCK状态下的SLPCMPAR比较寄存器自动减少功能	0x0
SL_INCA_EN0	[1]	RW	0：禁止PWM0在SLOCK状态下的SLPCMPAR比较寄存器自动增加功能 1：使能PWM0在SLOCK状态下的SLPCMPAR比较寄存器自动增加功能	0x0
SL_DECB_EN0	[2]	RW	0：禁止PWM0在SLOCK状态下的SLPCMPBR比较寄存器自动减少功能 1：使能PWM0在SLOCK状态下的SLPCMPBR比较寄存器自动减少功能	0x0
SL_INCB_EN0	[3]	RW	0：禁止PWM0在SLOCK状态下的SLPCMPBR比较寄存器自动增加功能 1：使能PWM0在SLOCK状态下的SLPCMPBR比较寄存器自动增加功能	0x0
SL_DECA_EN1	[4]	RW	0：禁止PWM1在SLOCK状态下的SLPCMPAR比较寄存器自动减少功能 1：使能PWM1在SLOCK状态下的SLPCMPAR比较寄存器自动减少功能	0x0
SL_INCA_EN1	[5]	RW	0：禁止PWM1在SLOCK状态下的SLPCMPAR比较寄存器自动增加功能 1：使能PWM1在SLOCK状态下的SLPCMPAR比较寄存器	0x0

			自动增加功能	
SL_DECB_EN1	[6]	RW	0：禁止PWM1在SLOCK状态下的SLPCMPBR比较寄存器自动减少功能 1：使能PWM1在SLOCK状态下的SLPCMPBR比较寄存器自动减少功能	0x0
SL_INCB_EN1	[7]	RW	0：禁止PWM1在SLOCK状态下的SLPCMPBR比较寄存器自动增加功能 1：使能PWM1在SLOCK状态下的SLPCMPBR比较寄存器自动增加功能	0x0
SL_CNTR_DEC_EN	[8]	RW	0：禁止PWM在SLOCK状态下的SLPCNTR比较寄存器自动减少功能 1：使能PWM在SLOCK状态下的SLPCNTR比较寄存器自动减少功能	
SL_CNTR_INC_EN	[9]	RW	0：禁止PWM在SLOCK状态下的SLPCNTR比较寄存器自动增加功能 1：使能PWM在SLOCK状态下的SLPCNTR比较寄存器自动增加功能	
S_ONE	[24]	RW	0：只输出一个周期SL_PxS设置的状态，下个周期继续输出PWM波形 1：一直输出SL_PxS设置的状态	0x0

注意：

1. A 和 B 的自动增减功能不可同时使能，只能选择其中一个进行使能。(A 自减自增，或者 B 自减自增)

2. SLCON 寄存器的设置必须在对 CNTRx / CMPARx / CMPBRx / SLPCMPARx / SLPCMPBRx 寄存器的初始化后再进行，否则 PWM 的输出可能出现异常情况。建议在 PWM 模块初始化代码的最末端设置该寄存器。

16.3.20 EPWM_SLSTEP0 (PWM0软锁止自增自减步长寄存器)

- Address = Base Address + 0x00B0, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
DEC_STEP	[11:0]	RW	Soft Lock发生后，比较寄存器自动减少的值	0x0
INC_STEP	[23:12]	RW	Soft Lock发生后，比较寄存器自动增加的值	0x0

16.3.21 EPWM_SLSTEP1 (PWM1软锁止自增自减步长寄存器)

- Address = Base Address + 0x00B4, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
DEC_STEP	[11:0]	RW	Soft Lock发生后，比较寄存器自动减少的值	0x0
INC_STEP	[23:12]	RW	Soft Lock发生后，比较寄存器自动增加的值	0x0

16.3.22 EPWM_IER (PWM 中断使能寄存器)

- Address = Base Address + 0x00B8, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				PMW1_SLPB_OVF	PMW1_SLPA_OVF	PMW0_SLPB_OVF	PMW0_SLPA_OVF	PWM2_CMPBDM	PWM2_CMPBUM	PWM2_CMPADM	PWM2_CMPAUM	PWM1_CMPBDM	PWM1_CMPBUM	PWM1_CMPADM	PWM1_CMPAUM	PWM0_CMPBDM	PWM0_CMPBUM	PWM0_CMPADM	PWM0_CMPAUM	PWM_CENTER2	PWM_PEND2	PWM_STOP2	PWM_START2	PWM_CENTER1	PWM_PEND1	PWM_STOP1	PWM_START1	PWM_CENTER0	PWM_PEND0	PWM_STOP0	PWM_START0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
PWM_START0	[0]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM_STOP0	[1]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM_PEND0	[2]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM_CENTER0	[3]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM_START1	[4]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM_STOP1	[5]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM_PEND1	[6]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM_CENTER1	[7]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM_START2	[8]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM_STOP2	[9]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM_PEND2	[10]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM_CENTER2	[11]	RW	1'b0：中断禁止 1'b1：中断使能	0x0

PWM0_CMPAUM	[12]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM0_CMPADM	[13]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM0_CMPBUM	[14]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM0_CMPBDM	[15]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM1_CMPAUM	[16]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM1_CMPADM	[17]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM1_CMPBUM	[18]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM1_CMPBDM	[19]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM2_CMPAUM	[20]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM2_CMPADM	[21]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM2_CMPBUM	[22]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PWM2_CMPBDM	[23]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PMW0_SLPA_OVF	[24]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PMW0_SLPB_OVF	[25]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PMW1_SLPA_OVF	[26]	RW	1'b0：中断禁止 1'b1：中断使能	0x0
PMW1_SLPB_OVF	[27]	RW	1'b0：中断禁止 1'b1：中断使能	0x0

16.3.23 EPWM_ICR (PWM 中断状态清除寄存器)

- Address = Base Address + 0x00BC, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				PMW1_SLPB_OVF	PMW1_SLPA_OVF	PMW0_SLPB_OVF	PMW0_SLPA_OVF	PWM2_CMPBDM	PWM2_CMPBUM	PWM2_CMPADM	PWM2_CMPAUM	PWM1_CMPBDM	PWM1_CMPBUM	PWM1_CMPADM	PWM1_CMPAUM	PWM0_CMPBDM	PWM0_CMPBUM	PWM0_CMPADM	PWM0_CMPAUM	PWM_CENTER2	PWM_PEND2	PWM_STOP2	PWM_START2	PWM_CENTER1	PWM_PEND1	PWM_STOP1	PWM_START1	PWM_CENTER0	PWM_PEND0	PWM_STOP0	PWM_START0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description	Reset Value
PWM_START0	[0]	W	1'b0：无效 1'b1：清除中断状态	0x0
PWM_STOP0	[1]	W	1'b0：无效 1'b1：清除中断状态	0x0
PWM_PEND0	[2]	W	1'b0：无效 1'b1：清除中断状态	0x0
PWM_CENTER0	[3]	W	1'b0：无效 1'b1：清除中断状态	0x0
PWM_START1	[4]	W	1'b0：无效 1'b1：清除中断状态	0x0
PWM_STOP1	[5]	W	1'b0：无效 1'b1：清除中断状态	0x0
PWM_PEND1	[6]	W	1'b0：无效 1'b1：清除中断状态	0x0
PWM_CENTER1	[7]	W	1'b0：无效 1'b1：清除中断状态	0x0
PWM_START2	[8]	W	1'b0：无效 1'b1：清除中断状态	0x0
PWM_STOP2	[9]	W	1'b0：无效 1'b1：清除中断状态	0x0
PWM_PEND2	[10]	W	1'b0：无效 1'b1：清除中断状态	0x0
PWM_CENTER2	[11]	W	1'b0：无效 1'b1：清除中断状态	0x0
PWM0_CMPAUM	[12]	W	1'b0：无效	0x0

			1'b1： 清除中断状态	
PWM0_CMPADM	[13]	W	1'b0： 无效 1'b1： 清除中断状态	0x0
PWM0_CMPBUM	[14]	W	1'b0： 无效 1'b1： 清除中断状态	0x0
PWM0_CMPBDM	[15]	W	1'b0： 无效 1'b1： 清除中断状态	0x0
PWM1_CMPAUM	[16]	W	1'b0： 无效 1'b1： 清除中断状态	0x0
PWM1_CMPADM	[17]	W	1'b0： 无效 1'b1： 清除中断状态	0x0
PWM1_CMPBUM	[18]	W	1'b0： 无效 1'b1： 清除中断状态	0x0
PWM1_CMPBDM	[19]	W	1'b0： 无效 1'b1： 清除中断状态	0x0
PWM2_CMPAUM	[20]	W	1'b0： 无效 1'b1： 清除中断状态	0x0
PWM2_CMPADM	[21]	W	1'b0： 无效 1'b1： 清除中断状态	0x0
PWM2_CMPBUM	[22]	W	1'b0： 无效 1'b1： 清除中断状态	0x0
PWM2_CMPBDM	[23]	W	1'b0： 无效 1'b1： 清除中断状态	0x0
PMW0_SLPA_OVF	[24]	W	1'b0： 无效 1'b1： 清除中断状态	0x0
PMW0_SLPB_OVF	[25]	W	1'b0： 无效 1'b1： 清除中断状态	0x0
PMW1_SLPA_OVF	[26]	W	1'b0： 无效 1'b1： 清除中断状态	0x0
PMW1_SLPB_OVF	[27]	W	1'b0： 无效 1'b1： 清除中断状态	0x0

16.3.24 EPWM_RISR (PWM 中断原始状态寄存器)

- Address = Base Address + 0x00C0, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				PMW1_SLPB_OVF	PMW1_SLPA_OVF	PMW0_SLPB_OVF	PMW0_SLPA_OVF	PWM2_CMPBDM	PWM2_CMPBUM	PWM2_CMPADM	PWM2_CMPAUM	PWM1_CMPBDM	PWM1_CMPBUM	PWM1_CMPADM	PWM1_CMPAUM	PWM0_CMPBDM	PWM0_CMPBUM	PWM0_CMPADM	PWM0_CMPAUM	PWM_CENTER2	PWM_PEND2	PWM_STOP2	PWM_START2	PWM_CENTER1	PWM_PEND1	PWM_STOP1	PWM_START1	PWM_CENTER0	PWM_PEND0	PWM_STOP0	PWM_START0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description	Reset Value
PWM_START0	[0]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM_STOP0	[1]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM_PEND0	[2]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM_CENTER0	[3]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM_START1	[4]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM_STOP1	[5]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM_PEND1	[6]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM_CENTER1	[7]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM_START2	[8]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM_STOP2	[9]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM_PEND2	[10]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM_CENTER2	[11]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM0_CMPAUM	[12]	R	1'b0： 该中断没有发生	0x0

			1'b1： 该中断发生	
PWM0_CMPADM	[13]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM0_CMPBUM	[14]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM0_CMPBDM	[15]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM1_CMPAUM	[16]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM1_CMPADM	[17]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM1_CMPBUM	[18]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM1_CMPBDM	[19]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM2_CMPAUM	[20]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM2_CMPADM	[21]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM2_CMPBUM	[22]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PWM2_CMPBDM	[23]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PMW0_SLPA_OVF	[24]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PMW0_SLPB_OVF	[25]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PMW1_SLPA_OVF	[26]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0
PMW1_SLPB_OVF	[27]	R	1'b0： 该中断没有发生 1'b1： 该中断发生	0x0

16.3.25 EPWM_MISR (PWM 中断状态寄存器)

- Address = Base Address + 0x00C4, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				PMW1_SLPB_OVF	PMW1_SLPA_OVF	PMW0_SLPB_OVF	PMW0_SLPA_OVF	PWM2_CMPBDM	PWM2_CMPBUM	PWM2_CMPADM	PWM2_CMPAUM	PWM1_CMPBDM	PWM1_CMPBUM	PWM1_CMPADM	PWM1_CMPAUM	PWM0_CMPBDM	PWM0_CMPBUM	PWM0_CMPADM	PWM0_CMPAUM	PWM_CENTER2	PWM_PEND2	PWM_STOP2	PWM_START2	PWM_CENTER1	PWM_PEND1	PWM_STOP1	PWM_START1	PWM_CENTER0	PWM_PEND0	PWM_STOP0	PWM_START0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description	Reset Value
PWM_START0	[0]	R	1'b0：该中断没有发生 1'b1：该中断被使能并且发生	0x0
PWM_STOP0	[1]	R	1'b0：该中断没有发生 1'b1：该中断被使能并且发生	0x0
PWM_PEND0	[2]	R	1'b0：该中断没有发生 1'b1：该中断被使能并且发生	0x0
PWM_CENTER0	[3]	R	1'b0：该中断没有发生 1'b1：该中断被使能并且发生	0x0
PWM_START1	[4]	R	1'b0：该中断没有发生 1'b1：该中断被使能并且发生	0x0
PWM_STOP1	[5]	R	1'b0：该中断没有发生 1'b1：该中断被使能并且发生	0x0
PWM_PEND1	[6]	R	1'b0：该中断没有发生 1'b1：该中断被使能并且发生	0x0
PWM_CENTER1	[7]	R	1'b0：该中断没有发生 1'b1：该中断被使能并且发生	0x0
PWM_START2	[8]	R	1'b0：该中断没有发生 1'b1：该中断被使能并且发生	0x0
PWM_STOP2	[9]	R	1'b0：该中断没有发生 1'b1：该中断被使能并且发生	0x0
PWM_PEND2	[10]	R	1'b0：该中断没有发生 1'b1：该中断被使能并且发生	0x0
PWM_CENTER2	[11]	R	1'b0：该中断没有发生 1'b1：该中断被使能并且发生	0x0
PWM0_CMPAUM	[12]	R	1'b0：该中断没有发生	0x0

			1'b1： 该中断被使能并且发生	
PWM0_CMPADM	[13]	R	1'b0： 该中断没有发生 1'b1： 该中断被使能并且发生	0x0
PWM0_CMPBUM	[14]	R	1'b0： 该中断没有发生 1'b1： 该中断被使能并且发生	0x0
PWM0_CMPBDM	[15]	R	1'b0： 该中断没有发生 1'b1： 该中断被使能并且发生	0x0
PWM1_CMPAUM	[16]	R	1'b0： 该中断没有发生 1'b1： 该中断被使能并且发生	0x0
PWM1_CMPADM	[17]	R	1'b0： 该中断没有发生 1'b1： 该中断被使能并且发生	0x0
PWM1_CMPBUM	[18]	R	1'b0： 该中断没有发生 1'b1： 该中断被使能并且发生	0x0
PWM1_CMPBDM	[19]	R	1'b0： 该中断没有发生 1'b1： 该中断被使能并且发生	0x0
PWM2_CMPAUM	[20]	R	1'b0： 该中断没有发生 1'b1： 该中断被使能并且发生	0x0
PWM2_CMPADM	[21]	R	1'b0： 该中断没有发生 1'b1： 该中断被使能并且发生	0x0
PWM2_CMPBUM	[22]	R	1'b0： 该中断没有发生 1'b1： 该中断被使能并且发生	0x0
PWM2_CMPBDM	[23]	R	1'b0： 该中断没有发生 1'b1： 该中断被使能并且发生	0x0
PMW0_SLPA_OVF	[24]	R	1'b0： 该中断没有发生 1'b1： 该中断被使能并且发生	0x0
PMW0_SLPB_OVF	[25]	R	1'b0： 该中断没有发生 1'b1： 该中断被使能并且发生	0x0
PMW1_SLPA_OVF	[26]	R	1'b0： 该中断没有发生 1'b1： 该中断被使能并且发生	0x0
PMW1_SLPB_OVF	[27]	R	1'b0： 该中断没有发生 1'b1： 该中断被使能并且发生	0x0

16.3.26 EPWM_EXTRG0 (PWM 输出至外部触发源选择寄存器0)

- Address = Base Address + 0x00C8, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								PWM2_CENTER		PWM2_PEND		PWM2_STOP		PWM2_START		PWM1_CENTER		PWM1_PEND		PWM1_STOP		PWM1_START		PWM0_CENTER		PWM0_PEND		PWM0_STOP		PWM0_START	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
PWM0_START事件 PWM0_STOP事件 PWM0_PEND事件 PWM0_CENTER事件 PWM1_START事件 PWM1_STOP事件 PWM1_PEND事件 PWM1_CENTER事件 PWM2_START事件 PWM2_STOP事件 PWM2_PEND事件 PWM2_CENTER事件	-	RW	00：不触发 01：该事件作为ADC触发源 10：该事件作为STIMER触发源 11：该事件作为ADC和STIMER的触发源	0x0

16.3.27 EPWM_EXTRG1 (PWM 输出至外部触发源选择寄存器1)

- Address = Base Address + 0x00CC, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								PWM2_CMPBDM		PWM2_CMPBUM		PWM2_CMPADM		PWM2_CMPAUM		PWM1_CMPBDM		PWM1_CMPBUM		PWM1_CMPADM		PWM1_CMPAUM		PWM0_CMPBDM		PWM0_CMPBUM		PWM0_CMPADM		PWM0_CMPAUM	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
PWM0 CMPAUM事件 PWM0 CMPADM事件 PWM0 CMPBUM事件 PWM0 CMPBDM事件 PWM1 CMPAUM事件 PWM1 CMPADM事件 PWM1 CMPBUM事件 PWM1 CMPBDM事件 PWM2 CMPAUM事件 PWM2 CMPADM事件 PWM2 CMPBUM事件 PWM2 CMPBDM事件	-	RW	00：不触发 01：该事件作为ADC触发源 10：该事件作为STIMER触发源 11：该事件作为ADC和STIMER的触发源	0x0

17 LED控制器（LED DRIVER）

17.1 概述

该型处理器内嵌一个 8x8 点的 LED 自动扫描控制器模块。它可以自动产生最大 8 个 COM 的扫描信号，通过不同时间槽分割来点亮 8 个 8 段数码管。在每个扫描时间槽（单独一个 COM）内，SEGMENT 输出口会将相应的预设值输出到 IO 口上，COM 扫描的通道可以自由定义。LED 扫描还可以和 TOUCH 模块进行时间上的复用。

17.1.1 特性

- 可配置的 COM 通道数（支持从1个通道到8个通道扫描模式）。
- 通过寄存器可配置 LED 的刷新率。
- 可选择的亮度调节功能。
- 支持 LED 闪烁扫描。
- 可配置 LED、TOUCH 的复用扫描工作模式。
  - 支持 LED 单独工作模式
  - 支持 TOUCH 单独工作模式
  - 支持 LED 和 TOUCH 复用扫描工作模式

17.2 功能描述

17.2.1 功能模块

LED 扫描控制模块功能图如下所示。

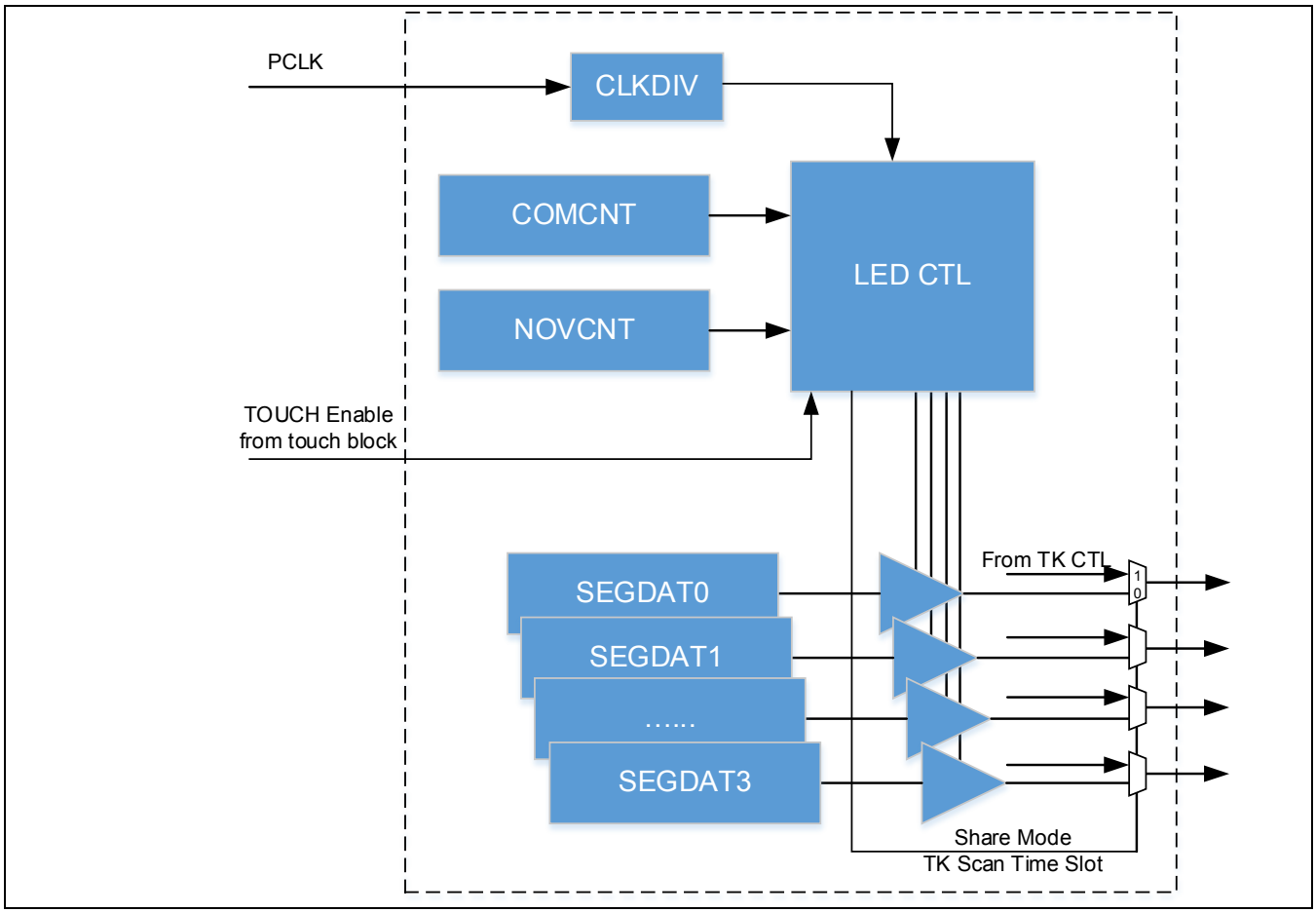


Figure 11-1 LED 扫描模块功能图

17.2.2 LED 扫描的工作模式

LED 自动扫描驱动器模块可以自动产生不互相叠加的扫描信号，用以驱动多个8段数码管。多个8段数码管的 SEG 端复用同一个驱动源。本驱动模块支持最大8个共阴极扫描的8段数码管进行并联扫描。在不同的时间段，通过使能 COM 端来点亮不同的数码管。每个 COM 管脚，均支持大电流驱动模式，可以在不外加驱动电路的前提下，直接驱动数码管（外部限流电阻任然需要）。扫描的 COM 数，可以通过 LED_CR 寄存器中的 COM_EN 来设置。

当需要点亮 LED 时，LED 的扫描时钟频率需要预先设置，而且每个 COM 的扫描有效时间长度也同样需要配置好。扫描的时序控制可以通过 LED_TIMCR 来设置。在每个 COM 使能的时间内，相对应的存储在 SEGDATA 中的数据将会从 SEG7~SEG0口输出。

启动 LED 的自动扫描，通过设置 LED_CR 寄存器中的 LIGHTON 来设置。

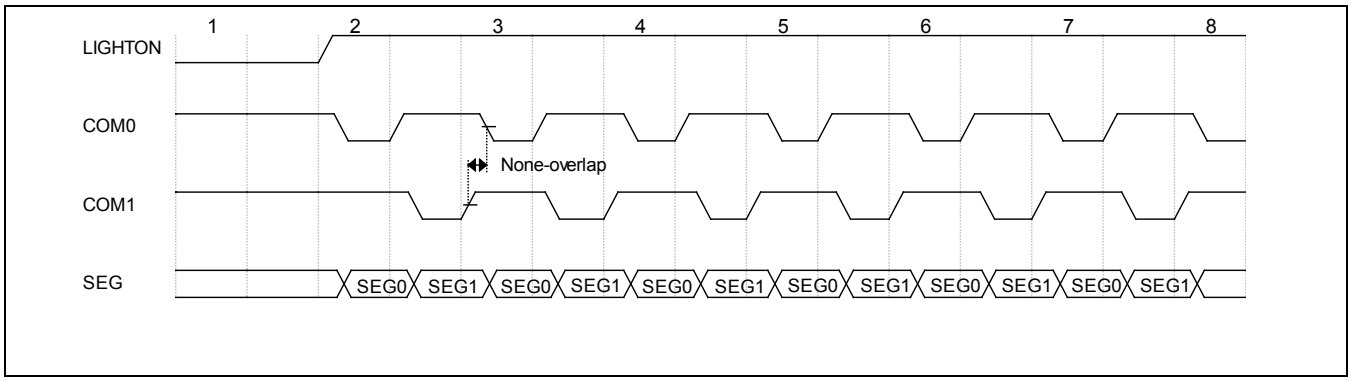


Figure 11-2 LED 扫描显示 （COM_EN = 8'b0000_0011模式）

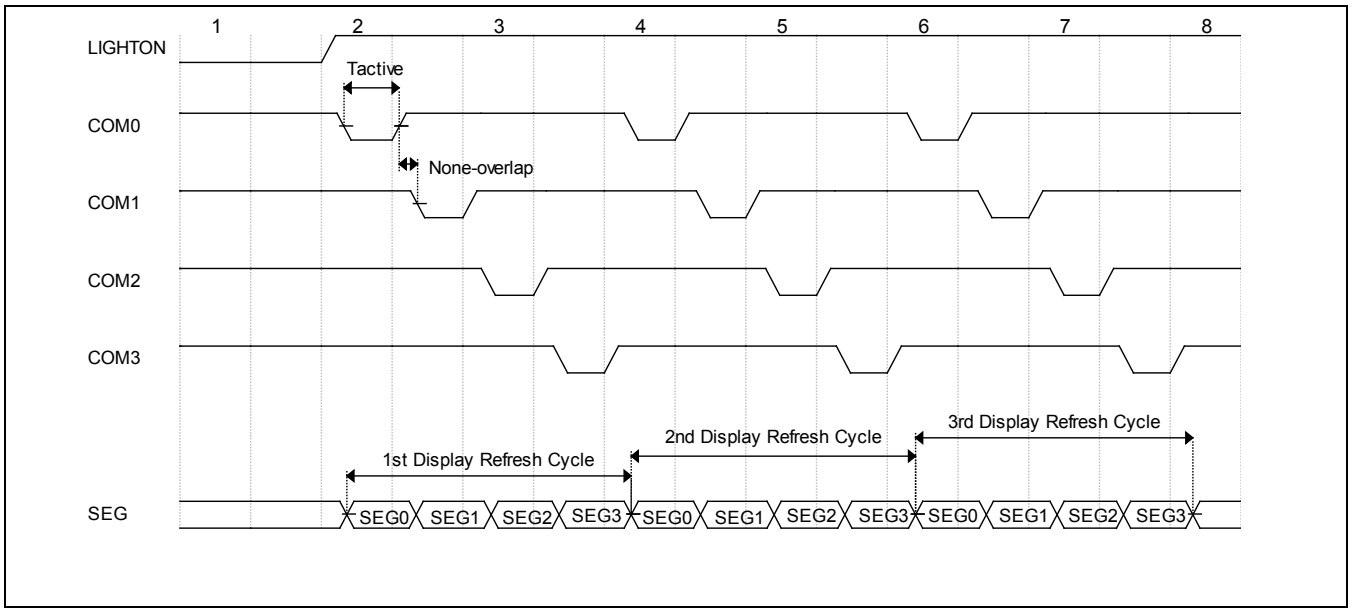


Figure 11-3 LED 扫描显示 （COM_EN = 8'b0000_1111模式）

上图中，T_{active} 是每个 COM 的使能有效时间，它的宽度可以通过下面的公式进行计算：

$$T_{active} = T_{lcd} / 8 \times COMCNT - T_{nov}$$

Where, T_{lcd} is LCD block clock cycle, T_{nov} is none-overlap time, which is decided by NOVCNT register. $T_{nov} = T_{lcd} \times NOVCNT$

LED 的显示亮度，可以通过 LED_BRIGHT 寄存器进行设置。寄存器的值表示不同的 COM 有效时间下的百分比宽度。百分比越低，COM 持续的时间就越短，显示的亮度就越低。

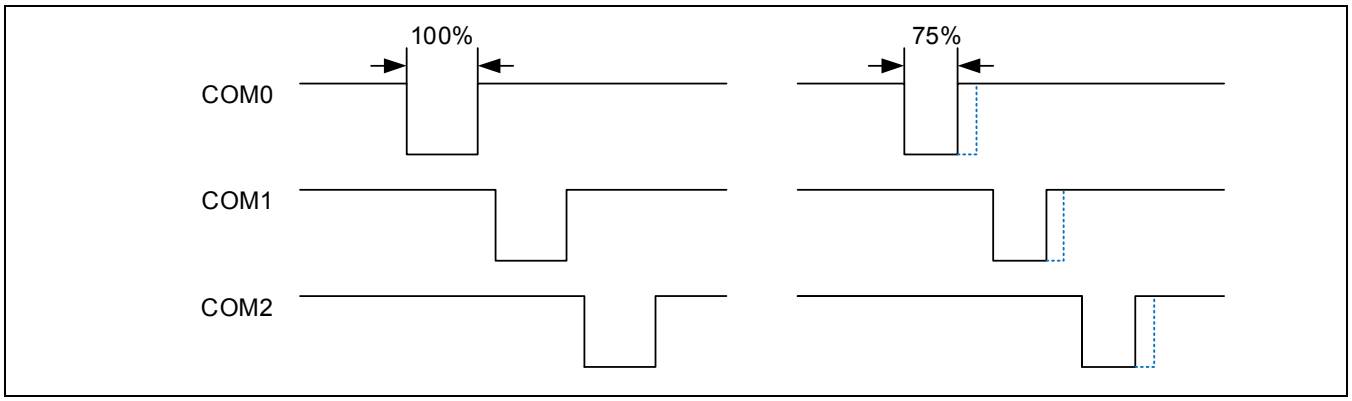


Figure 11-4 LED 显示亮度调节

17.2.3 LED 和 TOUCH 的复用

LED 的 SEGMENT 输出端口，可以通过不同的时间片段，来执行数码管的 SEG 输出，以及 TOUCH 的扫描工作。在 SEG 输出阶段，SEGMENT 端口输出相应 COM 的 SEGDATA 值，而在 TOUCH 扫描阶段，SEGMENT 端口的输出将关闭，切换为电容检查扫描端口。

TOUCH 分组的扫描时间和相应的 TOUCH 通道设置有关，比如灵敏度设置较大的情况下，扫描时间会比较长，扫描通道数比较多的时候，也会增加分组的整体扫描时间。TOUCH 分组的扫描时间增加，会增加 LED 的 COM 扫描的间隔时间，从而降低 LED 显示的刷新率。在 TOUCH 通道比较少的情况下，在不影响 LED 显示效果的前提下，可以适当增加 TOUCH 周期的时间，即增加 TOUCH 扫描周期内的按键通道扫描次数。这样可以有效改善 TOUCH 按键的响应时间。TOUCH 周期的按键通道扫描次数可以通过 LED_CR 寄存器中的 TKEYSCCNT 来进行设置。

在复用模式下工作时，SEGMENT 和 TOUCH 复用的 GPIO 应选择为 LED SEG 功能，同时应将 TOUCH 控制寄存器中的相应 TOUCH 通道使能（详细参考 TOUCH 的章节）。在启动扫描前通过设置 LED_CR 寄存器中的 TKEY_SHARE 控制位置高来选择复用模式。复用模式下，启动扫描的顺序为：先使能 LED 扫描，然后使能 TKEY 扫描。

LED 的 COM 扫描端口不能作为 TOUCH 的复用端口，在复用扫描的 TOUCH 扫描周期内，COM 信号一直输出高电平。

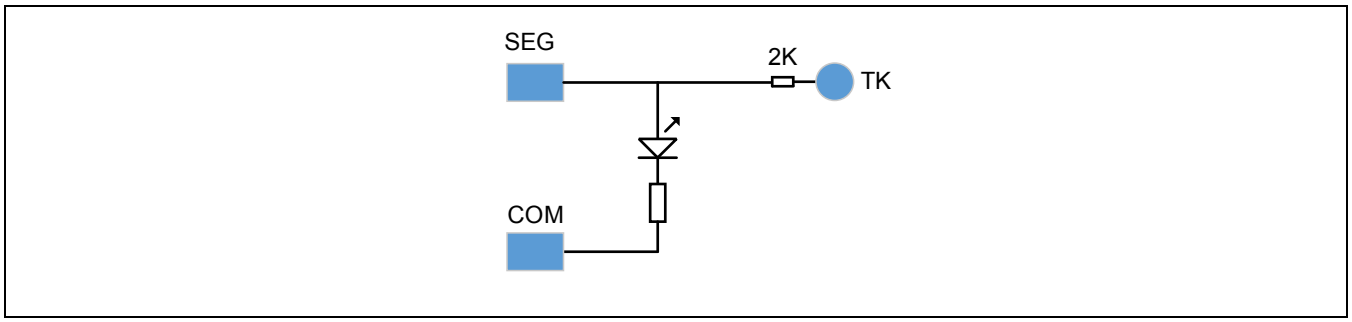


Figure 11-5 复用模式下的连接方案

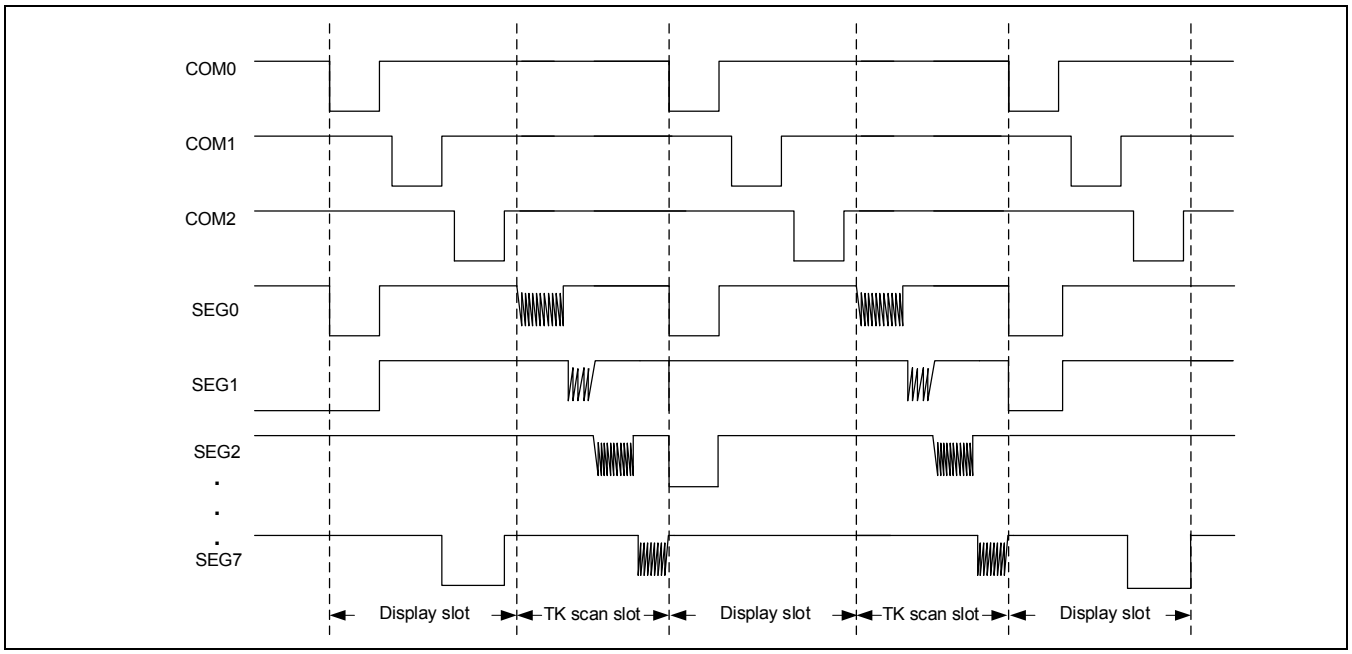


Figure 11-6 LED和TOUCH复用的工作模式

17.2.4 LED 闪烁控制

LED 的每个扫描 COM 端口，可以独立设置该周期内是否禁止输出，从而实现在不影响扫描显示效果的情况下（扫描顺序和周期不变化），实现特定 COM 的刷新停止，从而实现特定 COM 控制端的闪烁效果。COM 的输出禁止可以通过设置 BLKER 来设置，通过 BLKDR 来清除禁止标志。

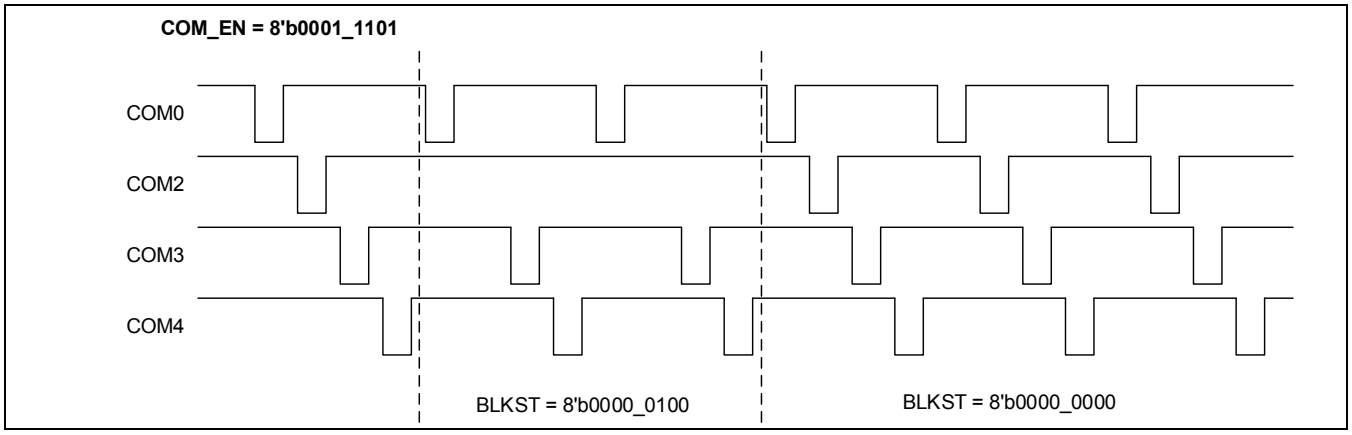


Figure 11-7 LED闪烁的工作模式

17.2.5 中断

LED 模式模块有三个中断。一个为 ICEND 中断，此中断将在每个 COM 扫描结束时触发。一个为 IPEND 中断，此中断将在所有的 COM 扫描结束后触发。另外一个为 IKEYDET 中断，此中断将在外部机械按键扫描时发现有按键状态发生变化时触发。当前按键的状态可以通过 KEYSR 寄存器获得。

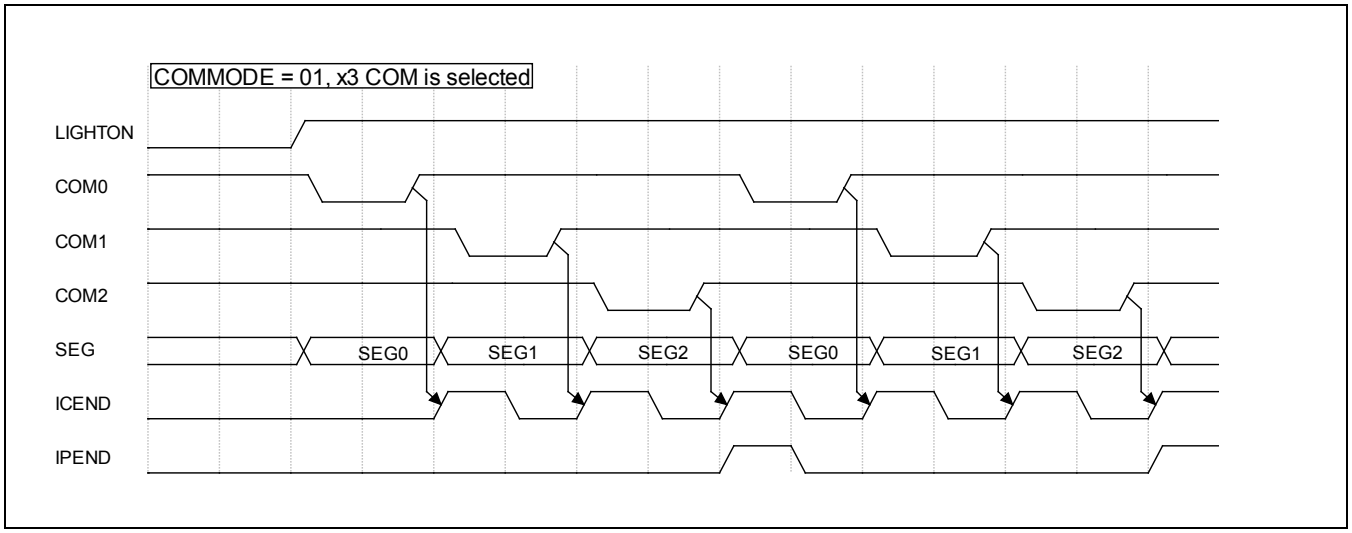


Figure 11-8 中断产生示意图

17.3 寄存器说明

17.3.1 寄存器表

- Base Address: 0x4006_0000

Table 11-1 寄存器表

Register	Offset	Description	Reset Value
LED_CR	0x00	通用控制寄存器	0x0000_0000
LED_BRIGHT	0x04	显示亮度控制寄存器	0x0000_0000
LED_RISR	0x08	原始中断状态标志寄存器	0x0000_0000
LED_IMCR	0x0C	中断使能控制寄存器	0x0000_0000
LED_MISR	0x10	中断标志寄存器	0x0000_0000
LED_ICR	0x14	中断标志清除寄存器	0x0000_0000
RSVD	0x18	保留	0x0000_0000
LED_TIMCR	0x1C	扫描时序控制寄存器	0x0000_0000
LED_BLKER	0x20	闪烁位设置控制寄存器	0x0000_0000
LED_BLKDR	0x24	闪烁位清除控制寄存器	0x0000_0000
LED_BLKST	0x28	闪烁位状态寄存器	0x0000_0000
LED_SEGDAT0	0x2C	段码输出数据寄存器0	0x0000_0000
LED_SEGDAT1	0x30	段码输出数据寄存器1	0x0000_0000
LED_SEGDAT2	0x34	段码输出数据寄存器2	0x0000_0000
LED_SEGDAT3	0x38	段码输出数据寄存器3	0x0000_0000
LED_SEGDAT4	0x3C	段码输出数据寄存器4	0x0000_0000
LED_SEGDAT5	0x40	段码输出数据寄存器5	0x0000_0000
LED_SEGDAT6	0x44	段码输出数据寄存器6	0x0000_0000
LED_SEGDAT7	0x48	段码输出数据寄存器7	0x0000_0000

NOTE:

17.3.2 LED_CR (通用控制寄存器)

- Address = Base Address + 0x0000, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								COM_EN								RSVD					TKEYSCCNT			SHAREMD	RSVD		COMOM	LEDCLK			LIGHTON
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
LIGHTON	[0]	R/W	LED 扫描启动。 1：启动 LED 自动扫描 0：停止 LED 自动扫描
LEDCLK	[3:1]	R/W	LED 模块工作时钟频率设置（基于 PCLK 的分频）。 000：F_{led} = PCLK/4 001：F_{led} = PCLK/8 010：F_{led} = PCLK/16 011：F_{led} = PCLK/32 100：F_{led} = PCLK/64 101：F_{led} = PCLK/128 110：F_{led} = PCLK/256 111：F_{led} = PCLK/32
COMOM	[4]	R/W	LED 和 TOUCH 复用时，COM 端口在 TOUCH 扫描时的状态。 0：高阻输出 1：高电平输出
SHAREMD	[7]	R/W	LED 和 TOUCH 复用扫描使能控制。 0：复用禁止，LED 单独工作 1：复用使能
TKEYSCCNT	[10:8]	R/W	在 LED 和 TOUCH 复用时，设置 TOUCH 周期内，TOUCH 扫描的循环次数。
COM_EN	[23:16]	R/W	COM 扫描的使能选择。COM_EN 的每一位对应一个 COM 扫描信号，当该位为‘1’时，所对应的 COM 扫描即被打开。 COM_EN[0]：COM0扫描使能 COM_EN[1]：COM1扫描使能 …… COM_EN[7]：COM7扫描使能

17.3.3 LED_BRIGHT (显示亮度控制寄存器)

- Address = Base Address + 0x0004, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
BRT	[2:0]	R/W	显示亮度的设置。 000：COM 的宽度为100% 001：COM 的宽度为87.5% 010：COM 的宽度为75% 011：COM 的宽度为62.5% 100：COM 的宽度为50% 101：COM 的宽度为37.5% 110：COM 的宽度为25% 111：COM 的宽度为12.5%

17.3.4 LED_RISR (原始中断状态标志寄存器)

- Address = Base Address + 0x0008, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
ICEND	[0]	R	一个 COM 扫描结束中断。 0：中断未发生 1：中断发生
IPEND	[1]	R	所有 COM 扫描结束中断。 0：中断未发生 1：中断发生

17.3.5 LED_IMCR (中断使能控制寄存器)

- Address = Base Address + 0x000C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
ICEND	[0]	R/W	一个 COM 扫描结束中断使能控制。 0：禁止中断 1：使能中断
IPEND	[1]	R/W	所有 COM 扫描结束中断使能控制。 0：禁止中断 1：使能中断

17.3.6 LED_MISR (中断标志寄存器)

- Address = Base Address + 0x0010, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
ICEND	[0]	R	一个 COM 扫描结束中断。 0：中断未发生 1：中断发生
IPEND	[1]	R	所有 COM 扫描结束中断。 0：中断未发生 1：中断发生

17.3.7 LED_ICR (中断标志清除寄存器)

- Address = Base Address + 0x0014, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
ICEND	[0]	W	清除一个 COM 扫描结束中断。
IPEND	[1]	W	清除所有 COM 扫描结束中断。

NOTE:

1） 寄存器只有在写入‘1’时有效，写入‘0’时无效。

2） ICR 在对应位写入‘1’时，对应中断状态标志被清除。

17.3.8 LED_TIMCR (扫描时序控制寄存器)

- Address = Base Address + 0x001C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
DCOMCNT	[7:0]	R/W	在显示期间，计数器值决定了一个 COM 显示周期的长度。该计数器的计数时钟为 LEDCLK/8。计数器的计数值为 DCOMCNT+7
NOVCNT	[15:8]	R/W	计数器值决定了相邻两个 COM 之间的互不交叠区间长度。计数器的计数值为 NOVCNT+1

17.3.9 LED_BLKER (闪烁位设置寄存器)

- Address = Base Address + 0x0020, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																								COM7_DIS	COM6_DIS	COM5_DIS	COM4_DIS	COM3_DIS	COM2_DIS	COM1_DIS	COM0_DIS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
COMx_DIS	[7:0]	W	COM 扫描周期内输出禁止（只写寄存器） 0：无效果 1：禁止该 COM 在扫描周期内输出

17.3.10 LED_BLKDR (闪烁位清除控制寄存器)

- Address = Base Address + 0x0024, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																								COM7_CLR	COM6_CLR	COM5_CLR	COM4_CLR	COM3_CLR	COM2_CLR	COM1_CLR	COM0_CLR
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
COMx_CLR	[7:0]	W	COM 扫描周期内输出使能（只写寄存器） 0：无效果 1：使能该 COM 在扫描周期内输出

17.3.11 LED_BLKST (闪烁位状态寄存器)

- Address = Base Address + 0x0028, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																								COM7_ST	COM6_ST	COM5_ST	COM4_ST	COM3_ST	COM2_ST	COM1_ST	COM0_ST
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
COMx_ST	[7:0]	R	COM 扫描周期内输出禁止状态 0：扫描输出使能 1：扫描输出禁止

17.3.12 LED_SEGDAT0~7 (段码输出数据寄存器0~7)

- LED_SEGDATA0: Address = Base Address + 0x002C, Reset Value = 0x0000_0000
- LED_SEGDATA1: Address = Base Address + 0x0030, Reset Value = 0x0000_0000
- LED_SEGDATA2: Address = Base Address + 0x0034, Reset Value = 0x0000_0000
- LED_SEGDATA3: Address = Base Address + 0x0038, Reset Value = 0x0000_0000
- LED_SEGDATA4: Address = Base Address + 0x003C, Reset Value = 0x0000_0000
- LED_SEGDATA5: Address = Base Address + 0x0040, Reset Value = 0x0000_0000
- LED_SEGDATA6: Address = Base Address + 0x0044, Reset Value = 0x0000_0000
- LED_SEGDATA7: Address = Base Address + 0x0048, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																								SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
SEGx	[7:0]	R/W	当 COMx 使能时，SEG 端口输出的数据

18 通用同步异步收发器 (USART)

18.1 概述

通用同步异步收发器(USART)用来在不同单片机之间进行通讯。USART串行发送比特位数据(低位优先)，在接收端，另外一个USART则将这些数据组合成完成数据字节。

串行数据传输通常使用在电脑之间的非网络通讯，以及终端和其它设备间的通讯。

18.1.1 主要特性

- 可编程波特率发生器
- 校验位，帧检测和数据溢出错误检测
- J1587协议的Idle标志
- 支持产生传输线打断(Break)和检测
- 支持自动应答，本地回环模式，和远程回环模式
- Multi-drop模式: 地址检测和产生
- 中断产生
- 5 到 9 位的字符长度
- 可控制的数据传输起始位
- 支持智能卡协议：产生错误信号和重新发送
- 异步模式最大波特率: PCLK/16

18.1.2 管脚描述

Table 18-1 USART管脚描述

管脚名称	功能	I/O 类型	有效电平	说明
USART_TX	USART发送数据线	O	–	–
USART_RX	USART接收数据线	I	–	–

18.2 功能描述

18.2.1 模块框图

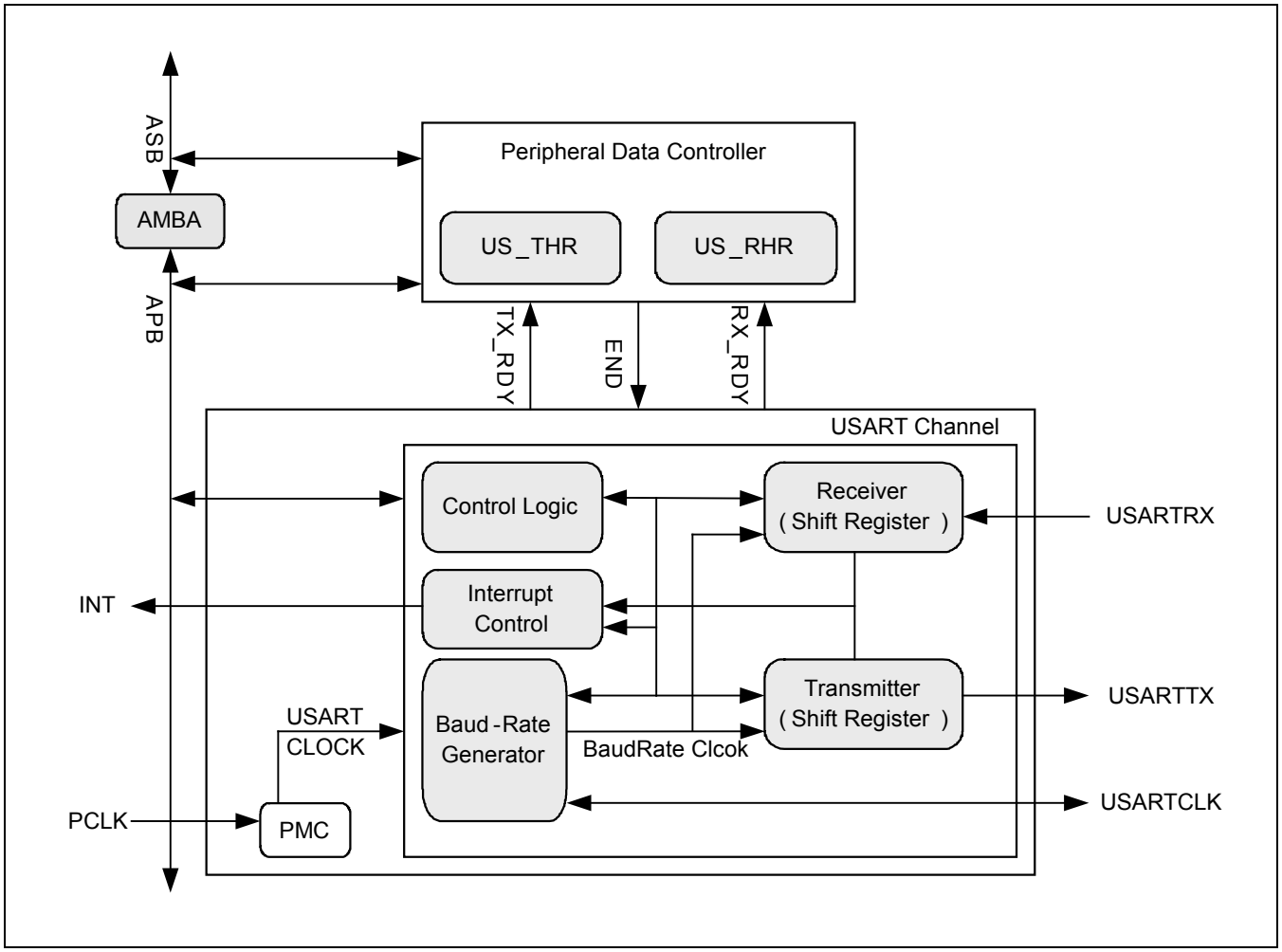


Figure 18-1 USART模块框图

18.2.2 波特率发生器

18.2.2.1 功能描述

波特率发生器用来给发送端和接收端提供时钟，其内部时钟源可以是PCLK，或者PCLK的8分频(PCLK/8)。USART用来传送1比特所需要的时间为位周期，位周期的倒数即是波特率。

18.2.2.2 异步模式

当UART工作在异步模式时(模式寄存器US_MR的SYNC=0)，波特率为选择的时钟除以16，再除以US_BRGR(波特率配置寄存器)中CD的值。如果US_BRGR寄存器为0，那么波特率发生器的时钟被禁止。

- 波特率 = 选择的时钟/(16 × CD)，选择的时钟可以是 PCLK 或者 PCLK/ 8.

18.2.2.3 同步模式

当UART工作在同步模式(模式寄存器US_MR的SYNC=1)，并且选择的时钟为内部时钟(模式寄存器US_MR中CLKS[1]=0)时，波特率为内部选择的时钟除以US_BRGR寄存器中的值。如果US_BRGR寄存器为0，那么波特率发生器的时钟被禁止。

- 波特率 = 选择的时钟/CD，选择的时钟可以是 PCLK 或者 PCLK/ 8.

18.2.2.4 模块框图

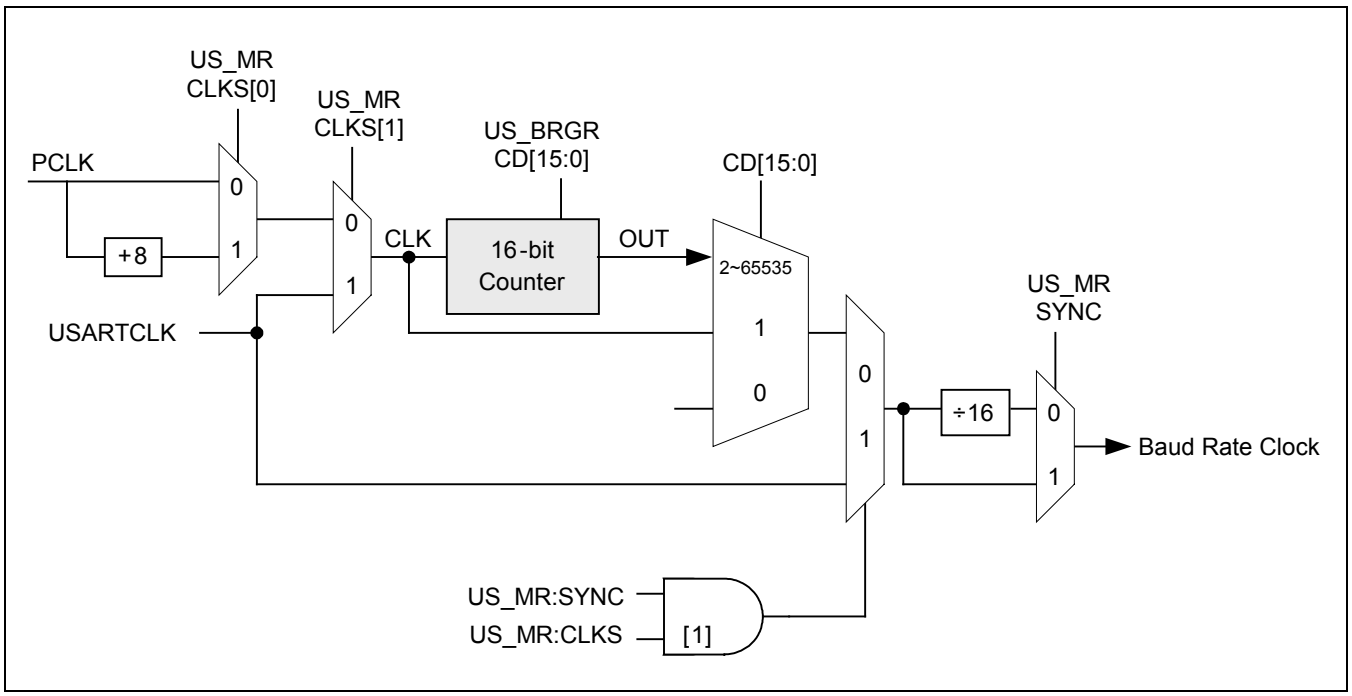


Figure 18-2 UART波特率发生器模块框图

18.2.2.5 波特率配置示例

下面表格为不同系统时钟下，US_BRGR不同配置值对应的波特率。误差栏表示实际波特率和期望波特率的差异。

下表为 CLKS[1:0] = 00 (UART时钟选择PCLK) 和 US_MR寄存器中SYNC = 0 (异步模式)的情况。

Table 18-2 异步模式 (SYNC = 0)

PCLK (MHz)	US_BRGR	波特率	% 误差
40	2083	1200	–0.02%
	1042	2400	0.03%
	521	4800	0.03%
	260	9600	–0.16%
	174	14400	0.22%
	130	19200	–0.16%
	65	38400	–0.16%
37.5	1953	1200	–0.01%
	977	2400	0.04%
	488	4800	–0.06%
	244	9600	–0.06%
	163	14400	0.15%
	122	19200	–0.06%
	61	38400	–0.06%
36	1875	1200	0.00%
	938	2400	0.05%
	469	4800	0.05%
	234	9600	–0.16%
	156	14400	–0.16%
	117	19200	–0.16%
	39	57600	–0.16%
30	1563	1200	0.03%
	781	2400	–0.03%
	391	4800	0.10%
	195	9600	–0.16%
	130	14400	–0.16%
	98	19200	0.35%
	49	38400	0.35%
20	1042	1200	0.03%
	521	2400	0.03%
	260	4800	–0.16%

PCLK (MHz)	US_BRGR	波特率	% 误差
	130	9600	–0.16%
	87	14400	0.22%
	65	19200	–0.16%
18.75	977	1200	0.04%
	488	2400	–0.06%
	244	4800	–0.06%
	122	9600	–0.06%
	81	14400	–0.47%
	61	19200	–0.06%
18	938	1200	0.05%
	469	2400	0.05%
	234	4800	–0.16%
	117	9600	–0.16%
	78	14400	–0.16%
16	833	1200	–0.04%
	417	2400	0.08%
	208	4800	–0.16%
	104	9600	–0.16%
	52	19200	–0.16%
	26	38400	–0.16%
15	781	1200	–0.03%
	391	2400	0.10%
	195	4800	–0.16%
	98	9600	0.3%
	65	14400	–0.16%
	49	19200	0.35%
10	521	1200	0.03%
	260	2400	–0.16%
	130	4800	–0.16%
	65	9600	–0.16%
9.375	488	1200	–0.06%
	244	2400	–0.06%
	122	4800	–0.06%
	61	9600	–0.06%
8	417	1200	0.08%
	208	2400	–0.16%

PCLK (MHz)	US_BRGR	波特率	% 误差
	104	4800	–0.16%
	52	9600	–0.16%
	26	19200	–0.16%
	13	38400	–0.16%
5	260	1200	–0.16%
	130	2400	–0.16%
	65	4800	–0.16%
4.6875	244	1200	–0.06%
	122	2400	–0.06%
	61	4800	–0.06%
4	208	1200	–0.16%
	104	2400	–0.16%
	52	4800	–0.16%
	26	9600	–0.16%
	13	19200	–0.16%
2.5	130	1200	–0.16%
	65	2400	–0.16%
2	104	1200	–0.16%
	52	2400	–0.16%
	26	4800	–0.16%
	13	9600	–0.16%
1.25	65	1200	–0.16%
1	52	1200	–0.16%
	26	2400	–0.16%
	13	4800	–0.16%
0.5	26	1200	–0.16%
	13	2400	–0.16%
0.25	13	1200	–0.16%

Table 18-3 同步模式 (SYNC = 1)

PCLK (MHz)	US_BRGR CD	波特率	% 误差
40	174 × 16	14400	0.22%
	130 × 16	19200	–0.16%
	65 × 16	38400	–0.16%
37.5	244 × 16	9600	–0.06%
	163 × 16	14400	0.15%
	122 × 16	19200	–0.06%
	61 × 16	38400	–0.06%
36	234 × 16	9600	–0.16%
	156 × 16	14400	–0.16%
	117 × 16	19200	–0.16%
	39 × 16	57600	–0.16%
30	195 × 16	9600	–0.16%
	130 × 16	14400	–0.16%
	98 × 16	19200	0.35%
	49 × 16	38400	0.35%
20	130 × 16	9600	–0.16%
	87 × 16	14400	0.22%
	65 × 16	19200	–0.16%
18.75	244 × 16	4800	–0.06%
	122 × 16	9600	–0.06%
	81 × 16	14400	–0.47%
	61 × 16	19200	–0.06%
18	234 × 16	4800	–0.16%
	117 × 16	9600	–0.16%
	78 × 16	14400	–0.16%
16	208 × 16	4800	–0.16%
	104 × 16	9600	–0.16%
	52 × 16	19200	–0.16%
	26 × 16	38400	–0.16%
15	195 × 16	4800	–0.16%
	98 × 16	9600	0.35%
	65 × 16	14400	–0.16%
	49 × 16	19200	0.35%
10	130 × 16	4800	–0.16%
	65 × 16	9600	–0.16%

PCLK (MHz)	US_BRGR CD	波特率	% 误差
9.375	244 × 16	2400	–0.06%
	122 × 16	4800	–0.06%
	61 × 16	9600	–0.06%
8	208 × 16	2400	–0.16%
	104 × 16	4800	–0.16%
	52 × 16	9600	–0.16%
	26 × 16	19200	–0.16%
	13 × 16	38400	–0.16%
5	130 × 16	2400	–0.16%
	65 × 16	4800	–0.16%
4.6875	244 × 16	1200	–0.06%
	122 × 16	2400	–0.06%
	61 × 16	4800	–0.06%
4	208 × 16	1200	–0.16%
	104 × 16	2400	–0.16%
	52 × 16	4800	–0.16%
	26 × 16	9600	–0.16%
	13 × 16	19200	–0.16%
2.5	130 × 16	1200	–0.16%
	65 × 16	2400	–0.16%
2	104 × 16	1200	–0.16%
	52 × 16	2400	–0.16%
	26 × 16	4800	–0.16%
	13 × 16	9600	–0.16%
1.25	65 × 16	1200	–0.16%
1	52 × 16	1200	–0.16%
	26 × 16	2400	–0.16%
	13 × 16	4800	–0.16%
0.5	26 × 16	1200	–0.16%
	13 × 16	2400	–0.16%
0.25	13 × 16	1200	–0.16%

18.2.3 接收端功能

18.2.3.1 异步接收

当SYNC = 0(US_MR中的第8位)时，UART被设置为异步工作模式。在异步模式下，UART会通过一直采样UARTRX信号来检测起始位，直到检测到一个有效的起始位为止。如果UARTRX上的一个低电平(SPACE)时长超过了7个采样时钟的周期，那么这个长低电平则会被判断为一个有效的起始位。采样时钟的频率为波特率的16倍。所以一个长于7/16位周期的低电平为有效起始位，而短于7/16位周期的低电平会接收端被忽略，然后接收端会继续等待有效的起始位。

当一个有效起始位被检测到，接收端会继续在每个位周期的理论中心点对UARTRX信号进行采样。假设每个比特的长度为采样时钟周期的16倍(1比特位)，那么采样点为该比特位开始后的第8个采样时钟周期(0.5比特位)。所以第一个采样点为起始位下降沿后的第24个采样时钟周期(1.5比特位)，接下来的每个采样点都跟前一个采样点间隔16个采样时钟周期(1比特位)。

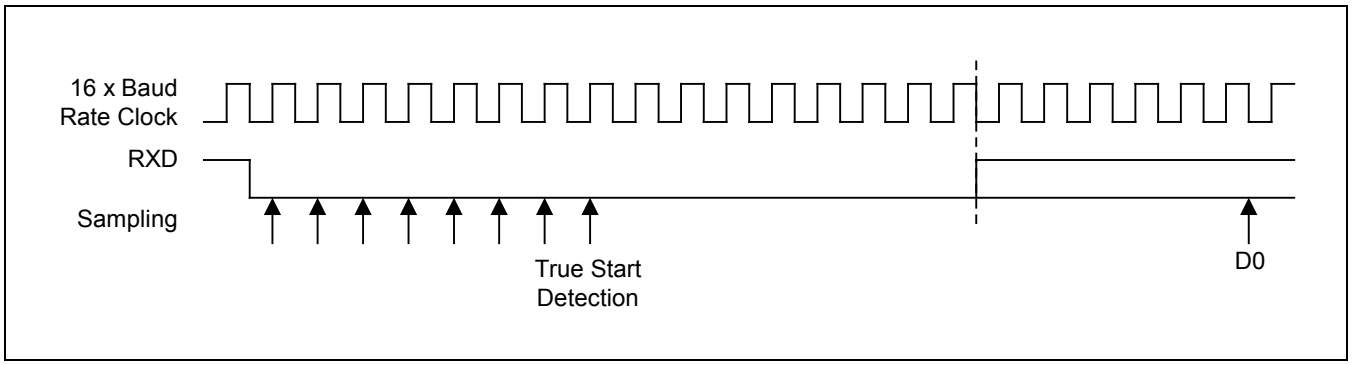


Figure 18-3 异步模式，起始位检测

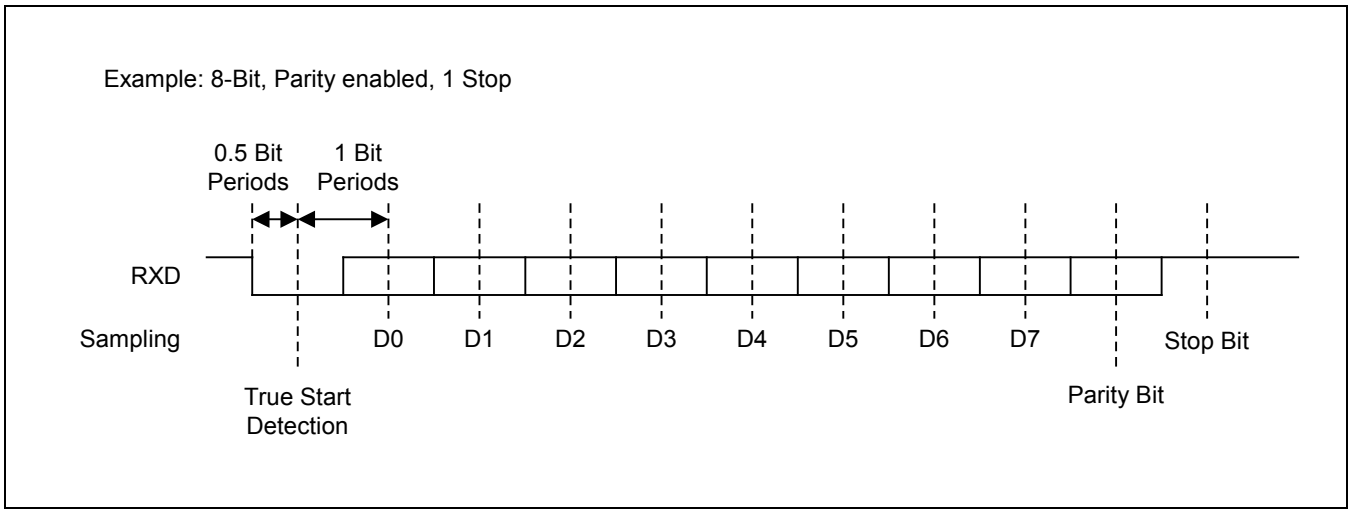


Figure 18-4 异步模式，字节接收

18.2.3.2 同步接收

当配置成同步模式(SYNC = 1)，接收端在每个USARTCLK的上升沿对RXD信号进行采样。如果检测到一个低电平，那么这个低电平就被认为是起始位。收到起始位后，接收端继续采样数据位，校验位和停止位，然后继续等待下一个起始位。参考下面的示例图。

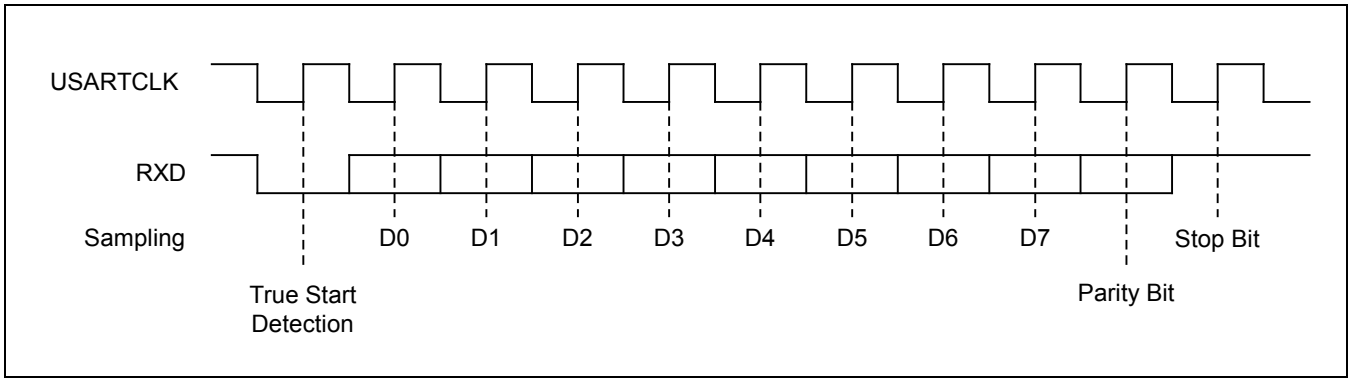


Figure 18-5 同步模式，字节接收

18.2.3.3 接收标志位

收到一个完整的字节后，该字节会被存到US_RHR中，同时US_SR寄存器中的RXRDY标志位会被置1。RXRDY在最后的停止位结束后被置1。

18.2.3.4 溢出错误

如果US_RHR寄存器在被读取之前，又被存入了新接收的字节，那么US_SR寄存器中的OVER状态位会被置1。

18.2.3.5 校验错误

每次接收到一个字节，接收端都会根据US_MR(UART模式寄存器)中PAR[2:0]的值计算接收到数据的校验值，然后跟接收到的校验位进行比较，如果不相同，那么US_SR寄存器中的PARE校验错误位会被置1。

18.2.3.6 帧错误

如果接收到的停止位为低电平并且接收到的数据位至少有一个高电平，那么接收端会产生一个帧错误标志，将US_SR寄存器中的FRAME位置1。

18.2.3.7 空闲标志

空闲标志在UART收到一个起始位后变低，在J1587协议帧结束后(10个停止位后)变高。空闲标志位在置起时可以产生中断。

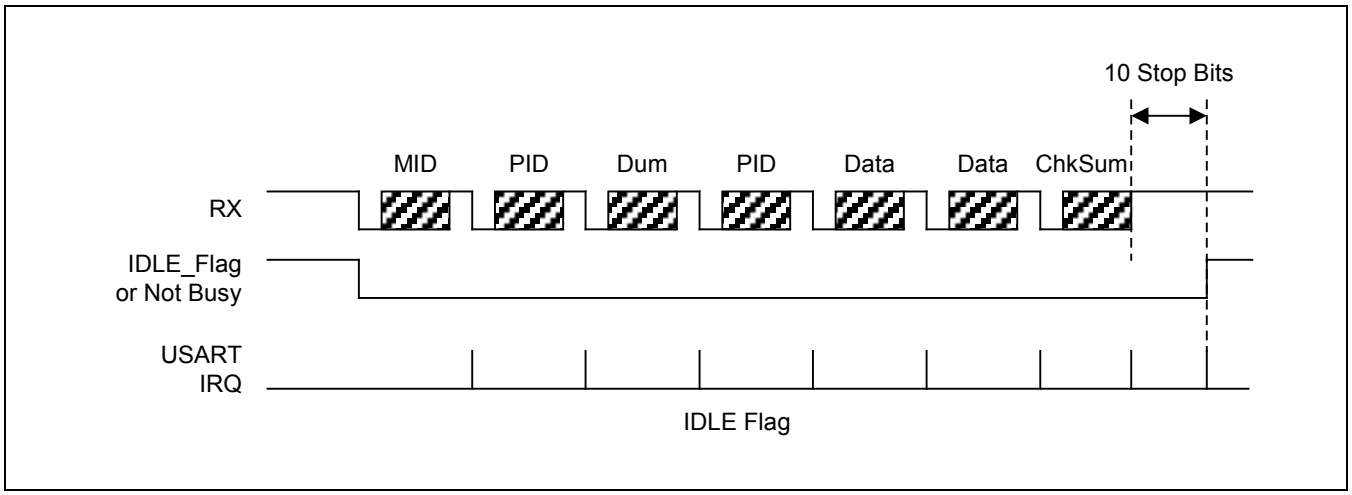


Figure 18-6 空闲标志

18.2.3.8 超时

这个功能可以检测RXD的空闲状态。USART等待接收一个新字节的最大时间可以在US_RTOR (Receiver Time-out)寄存器的TO[15:0]里设置。当这个寄存器设置为0时，超时功能关闭。

在超时功能打开的情况下，接收端在接收到第一个字节后，会打开一个计数器，这个计数器在每个位周期自动减1，并且会在接收到字节后重新载入计数值(即TO[15:0]设置的值)。当计数器计到0时，US_SR中的TIMEOUT被置1。用户通过对US_CR寄存器中STTTO (Start Time-Out)位写1来启动(或者重新启动)这个功能。

也就是说，启动超时功能，必须满足下面条件：

- US_RTOR不为0
- US_CR寄存器中STTTO (Start Time-Out)位写1
- 收到一个字节

超时的时长计算：

时长 = 寄存器值(TO[15:0]) × 位周期 (异步模式)
时长 = 寄存器值(TO[15:0]) × 16 × 采样周期 (同步模式)

18.2.4 发送端

18.2.4.1 功能描述

发送功能在同步模式和异步模式下的行为是完全一样的。发送端将开始位，数据位，校验位和停止位串行移位出去，低位(LSB)先发，高位(MSB)后发。

数据位的个数由US_MR寄存器中的CHRL[1:0]选择。

校验位由US_MR寄存器中的PAR[2:0]位设置。如果校验为偶校验，那么校验位是所有数据位的和(单比特相加的和)。如果校验为奇校验，那么校验位是所有数据位的和取反。

停止位的个数由US_MR寄存器中的NBSTOP[1:0]选择。

当需要传送的字节被写入到US_THR (Transmit Holding)中时，只要移位寄存器是空的，那么该字节会被马上复制到移位寄存器中。

当发送操作发生时，US_SR中的TXRDY位会被置1，直到US_THR寄存器被写入了新的字节。如果移位寄存器和US_THR寄存器都是空的，那么US_SR中的TXEMPTY会被置1 (在最后的停止位之后)。

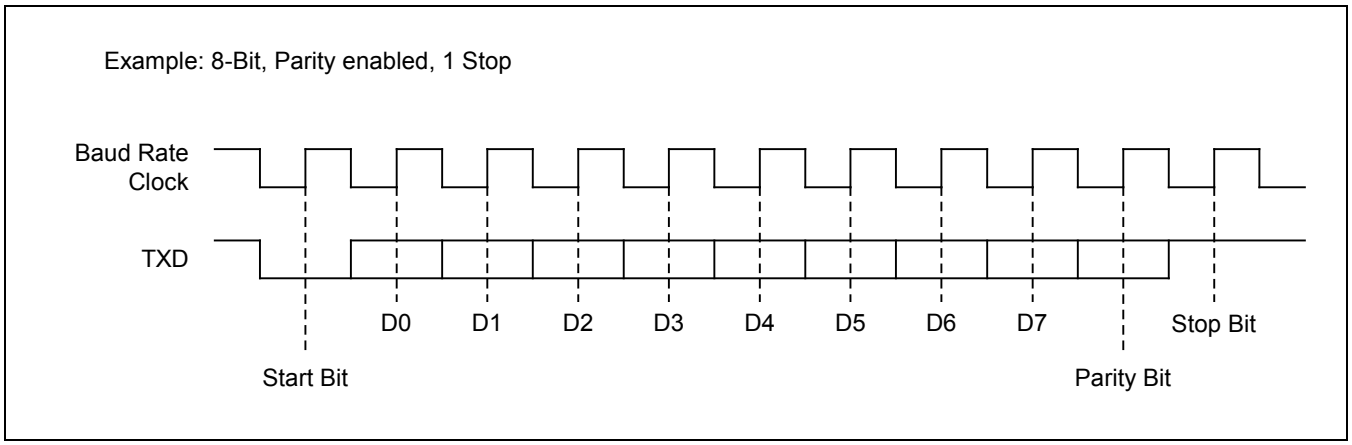


Figure 18-7 同步和异步模式，字节发送

18.2.4.2 Time-Guard

Time-guard功能可以在USARTTX上发送的2个字节中间插入一段空闲时间。这个空闲状态的时长在US_TTGR (Transmitter Time-Guard)寄存器中设置。当这个寄存器为0时，不产生time-guard。否则，发送端会在每次发送完一个字节后，将USARTTX拉高保持一段时间，时长为US_TTGR中设置的值乘以位周期。

18.2.4.3 Multi-Drop模式

当US_MR中PAR位等于11Xb时，USART被配置为multi-drop模式，用来自动检测地址和数据，这时候PARE (US_SR寄存器中的校验错误位)被用来区分数据字节(校验位为0)和地址字节(校验位为1)。所以在这个模式中，如果数据为一个地址字节，那么校验错误位(US_SR中的PARE)被置1。PARE状态可以由US_CSR(状态清除寄存器)中的PARE位来清除。如果校验位为0，表示该字节为数据字节，PARE不会被置1。

当发送地址命令(SENDA)被写入到US_CR中时，发送端发送的是地址字节(校验位置1)。这种情况下，写入到US_THR中的下个字节会被当作地址发送出去(校验位为1)，而在这个字节后的任何字节的校验位都为0。

18.2.5 Break

18.2.5.1 发送Break

当US_CR寄存器中的STTBRK (Start break)命令被置1时，发送端会在USARTTX上发送Break。在USARTTX被拉低之前，发送移位寄存器中的字节会被发送完。

如果要移除Break，那么必须将US_CR中的STOPBRK (Stop break)命令置1。USART最少发送一个字节长的Break。

之后USARTTX会恢复到高电平(空闲状态)并且持续12个位周期，保证Break被正确的检测到，然后发送端继续正常的操作。

18.2.5.2 接收Break

当所有的数据，校验和停止位都是0时，接收端认为检测到了Break。在检测到低电平地址位的时刻，接收端将US_SR中的RXBRK (Break received)位置1。

18.2.6 中断

US_SR中的大部分状态都在US_IMSCR (中断使能/禁止寄存器), US_RISR (原始中断状态寄存器), US_MISR (中断状态寄存器), 和US_ICR (中断状态清除寄存器) 中有对应的位，可以控制中断的产生。

18.2.7 测试模式

USART可以用US_MR中的CHMODE[1:0]配置成3种不同的测试模式。

自动回应模式：自动重新发送收到的数据，对发送端的任何配置都无效。

本地回环模式：接收自己发送的数据，不使用USARTTX和USARTRX管脚，而是内部将发送端的输出连到接收端的输入，USARTRX管脚的电平高低没有任何用途，并且USARTTX管脚会被一直拉高，就像是在空闲状态。

远程回环模式：直接将USARTTX管脚接到USARTRX管脚上，USART模块的发送和接收功能都禁止，只是负责将收到的数据直接转出去而不经过USART模块。

18.2.8 Smart-Card协议

USART兼容ISO7816-3协议，允许字节重送和校验错检测。

下面的描述只有在US_MR寄存器中的SMCARDPT位为1的时候才有效。

USART的Smart-Card协议需要发送端和接收端都支持。

如果GPIO模块允许，USARTTX可以配置成开漏输出模式，并且直接接到USARTRX管脚上，同时连接一个外部的上拉电阻，形成Smart-Card的数据信号线。

18.2.8.1 发送字节到Smart Card

USART可以通过检测Smart Card产生的校验错误信号来判断Smart Card是否正确的收到了上一个发送的字节。

当Smart Card产生了校验错的信号，上一个发送的字节会被USART重新发送，US_MR寄存器中的SENDTIME[2:0]用来控制重新发送的次数，直到Smart Card不再产生错误信号。

当USART检测到错误信号时，US_SR寄存器中的FRAME错误标志位会被置1。

USART检测错误信号的时间点是 t0 + t11 比特位，t0为开始位的下降沿 (也就是在2个停止位中间)。

下面的例子中，Smart Card检测到了校验错，在数据线(COMMS)上产生了一个错误信号。这个错误信号被USART检测到并且重新发送了上一个字节。

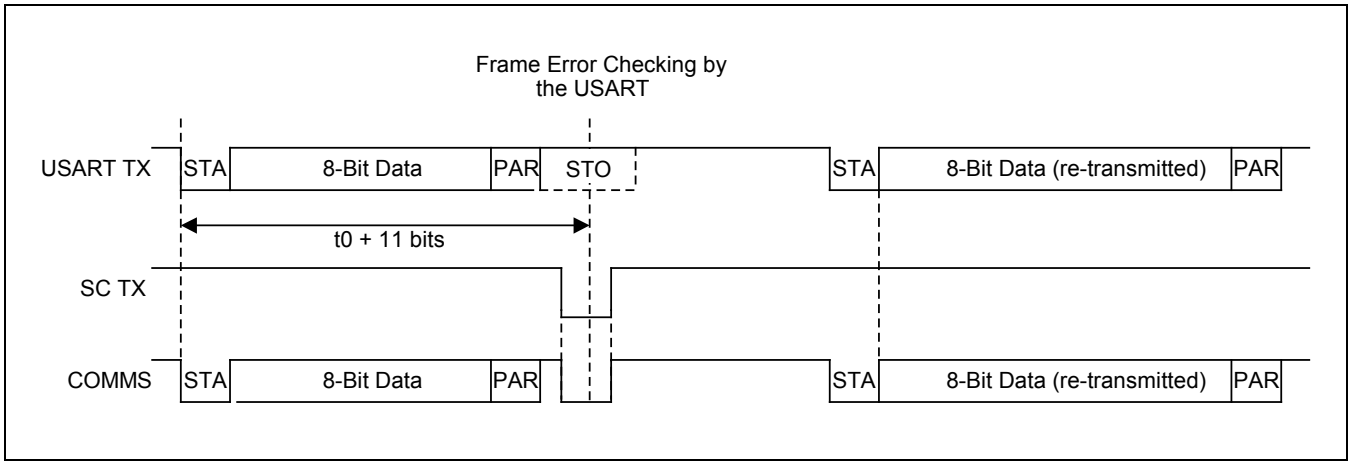


Figure 18-8 Smart-Card 发送错误

18.2.8.2 从Smart Card接收字节

当接收的字节有校验错时，USART可以产生错误信号 (参考ISO7816-3协议)。

当USART检测到校验错时，USART会在 t0 + 10.625 + [0:0.0625] 时间点(2个停止位中间)将传输线拉低1.0625个位周期，通知Smart Card上一个数据接收错误。使用T=0协议类型的Smart Card，必须重新发送该字节。

在这个情况下，USART会将US_SR寄存器中的PARE位置1，表示校验有错误。

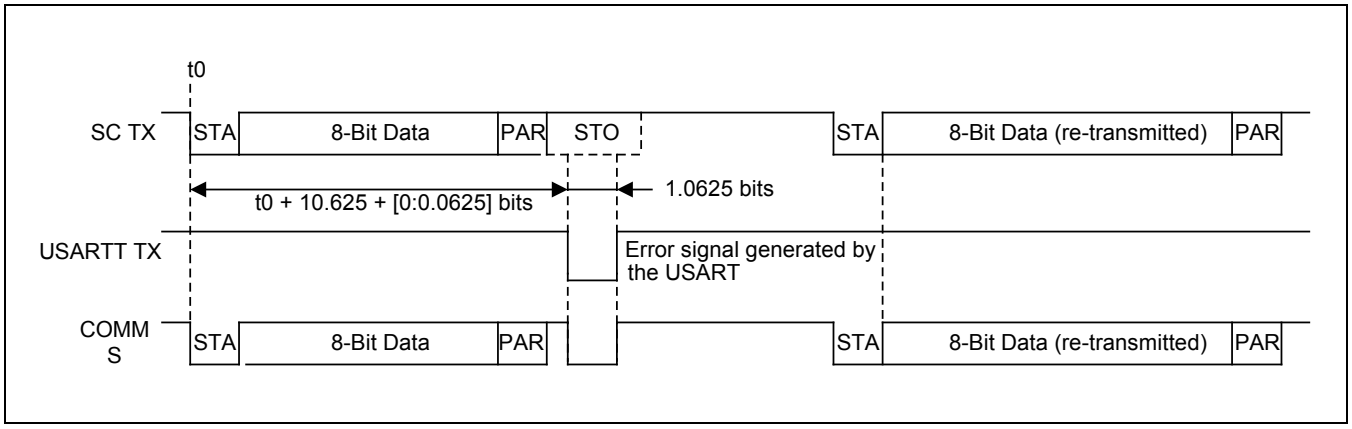


Figure 18-9 接收端的错误检测

18.2.8.3 Smart Card模式下的USART配置

要工作在Smart Card模式下，USART必须设置成普通模式，并且停止位的个数必须设置为2 (参考US_MR模式寄存器)。

18.3 寄存器说明

18.3.1 寄存器表 (Base Address: 0x4008_0000)

Register	Offset	Description	Reset Value
US_IDR	0x0000	ID寄存器	0x0011_001B
US_CEDR	0x0004	时钟使能/禁止寄存器	0x0000_0000
US_SRR	0x0008	软件复位寄存器	0x0000_0000
US_CR	0x000C	控制寄存器	0x0000_0000
US_MR	0x0010	模式寄存器	0x0000_0000
US_IMSCR	0x0014	中断使能/禁止寄存器	0x0000_0000
US_RISR	0x0018	原始中断状态寄存器	0x0000_0000
US_MISR	0x001C	中断状态寄存器	0x0000_0000
US_ICR	0x0020	中断状态清除寄存器	0x0000_0000
US_SR	0x0024	状态寄存器	0x0000_0800
US_RHR	0x0028	接收数据寄存器	0x0000_0000
US_THR	0x002C	发送数据寄存器	0x0000_0000
US_BRGR	0x0030	波特率配置寄存器	0x0000_0000
US_RTOR	0x0034	接收超时配置寄存器	0x0000_0000
US_TTGR	0x0038	发送端Time-Guard寄存器	0x0000_0000

18.3.1.1 US_IDR (USART ID寄存器)

- Address = Base Address + 0x0000, Reset Value = 0x0011_001B

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

Name	Bit	Type	Description	Reset Value
IDCODE	[25:0]	R	ID寄存器 IP的ID代码	0x0011_001B

18.3.1.2 US_CEDR (USART时钟使能/禁止寄存器)

- Address = Base Address + 0x0004, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
CLKEN	[0]	RW	时钟使能/禁止控制位 0 = 时钟禁止 1 = 时钟使能	0
DBGEN	[31]	RW	调试模式使能/禁止控制位 0 = 禁止调试模式 1 = 使能调试模式 说明： 0 = 进入调试模式后不影响USART功能 1 = 进入调试模式后冻结USART的功能，但USART内部寄存器的读写不受影响	0

18.3.1.3 US_SRR (USART软件复位寄存器)

- Address = Base Address + 0x0008, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
SWRST	[0]	W	软件复位 0 = 无效 1 = 软件复位	0

18.3.1.4 US_CR (USART控制寄存器)

- Address = Base Address + 0x000C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																			SENDA	STTTO	STPBRK	STTBRK	RSVD	TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	RSVD	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W	W	W	W	R	R

Name	Bit	Type	Description	Reset Value
RSTRX	[2]	W	复位接收端 0 = 无效 1 = 复位接收端逻辑	0
RSTTX	[3]	W	复位发送端 0 = 无效 1 = 复位发送端逻辑	0
RXEN	[4]	W	接收使能 0 = 无效 1 = 如果RXDIS是0，写1使能接收	0
RXDIS	[5]	W	接收禁止 0 = 无效 1 = 接收禁止	0
TXEN	[6]	W	发送使能 0 = 无效 1 = 如果TXDIS是0，写1使能发送	0
TXDIS	[7]	W	发送禁止 0 = 无效 1 = 发送禁止	0
STTBRK	[9]	W	开始Break. 0 = 无效 1 = 如果Break没有发送，那么写1会在当前移位寄存器中的数据发送完之后，开始发送Break状态	0
STPBRK	[10]	W	停止Break. 0 = 无效 1 = 如果一个Break状态正在发送，那么写1会在最少一个字节长度的Break状态后停止Break，并且发送一个12位周期的高电平	0
STTTO	[11]	W	开启超时接收	0

Name	Bit	Type	Description	Reset Value
			0 = 无效 1 = 必须在超时计数器计数完成之前，接收到字节数据，否则报错	
SENDA	[12]	W	发送地址 0 = 无效 1 = 只在Multi-drop模式，下一个写入US_THR的字节会被当作地址字节发送	0

18.3.1.5 US_MR (USART模式寄存器)

- Address = Base Address + 0x0010, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD											DSB	RSVD	CLKO	MODE9	SMCARDPT	CHMODE		NBSTOP		PAR			SYNC	CHRL		CLKS		SENDTIME			RSVD
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R

Name	Bit	Type	Description	Reset Value
SENDTIME	[3:1]	RW	表示USART被配置成Smart Card协议时，重复发送最多的次数 • SENDTIME配置位 SENDTIME[2:0] / 发送次数: 0 0 0 → 0; 0 0 1 → 1; –; 1 1 1 → 7	000'b
CLKS	[5:4]	RW	时钟选择 (波特率发生器的输入时钟). • CLKS 时钟选择位 CLKS[1:0] / 选择的时钟: 0 0 → PCLK; 0 1 → PCLK/8; 1 x → 保留，请勿使用	00'b
CHRL	[7:6]	RW	字节长度 (除开始位，停止位和校验位外的字节长度) • 字节长度位 CHRL[1:0] / 字节长度: 0 0 → 5位; 0 1 → 6位; 1 0 → 7位; 1 1 → 8位	00'b

<table>
<tr><th>Name</th><th>Bit</th><th>Type</th><th>Description</th><th>Reset Value</th></tr>
<tr><td>SYNC</td><td>[8]</td><td>RW</td><td>同步模式选择
0 = USART工作在异步模式
1 = USART工作在同步模式</td><td>0</td></tr>
<tr><td>PAR</td><td>[11:9]</td><td>RW</td><td>校验类型
• 校验类型位
<table><tr><th colspan="3">PAR[2:0]</th><th>校验类型</th></tr><tr><td>0</td><td>0</td><td>0</td><td>偶校验</td></tr><tr><td>0</td><td>0</td><td>1</td><td>奇校验</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0校验 (Space)</td></tr><tr><td>0</td><td>1</td><td>1</td><td>1校验 (Mark)</td></tr><tr><td>1</td><td>0</td><td>X</td><td>无校验</td></tr><tr><td>1</td><td>1</td><td>X</td><td>Multi-drop模式</td></tr></table>注意：如果使用LIN，PAR[2:0]必须设置为‘10X’.</td><td>000'b</td></tr>
<tr><td>NBSTOP</td><td>[13:12]</td><td>RW</td><td>停止位的个数
停止位个数跟SYNC设置的模式有关
• NBSTOP配置位
<table><tr><th colspan="2">NBSTOP [1:0]</th><th>异步模式 (SYNC = 0)</th><th>同步模式 (SYNC = 1)</th></tr><tr><td>0</td><td>0</td><td>1个停止位</td><td>1个停止位</td></tr><tr><td>0</td><td>1</td><td>1.5个停止位</td><td>保留</td></tr><tr><td>1</td><td>0</td><td>2个停止位</td><td>2个停止位</td></tr><tr><td>1</td><td>1</td><td>保留</td><td>保留</td></tr></table></td><td>00'b</td></tr>
<tr><td>CHMODE</td><td>[15:14]</td><td>RW</td><td>通道模式
• 通道模式位
<table><tr><th colspan="2">CHMODE [1:0]</th><th>模式描述</th></tr><tr><td>0</td><td>0</td><td>普通模式
USART通道工作为正常的Rx/Tx功能</td></tr><tr><td>0</td><td>1</td><td>自动回应
收到的数据自动通过USARTTX发送</td></tr><tr><td>1</td><td>0</td><td>本地回环
发送端的输出信号短接到接收端的输入信号</td></tr><tr><td>1</td><td>1</td><td>远程回环
USARTRX管脚内部直接短接到USARTTX管脚</td></tr></table></td><td>00'b</td></tr>
</table>

Name	Bit	Type	Description	Reset Value
SMCARDPT	[16]	RW	Smart Card协议 0 = 禁止smart card协议 1 = 使能smart card协议	0'b
MODE9	[17]	RW	9位字节长度 0 = CHRL位定义字节长度 1 = 9位字节长度	0'b
CLKO	[18]	RW	时钟输出选择 0 = USART不输出USARTCLK 1 = 如果CLKS[1]是0，USART输出USARTCLK	0'b
DSB	[20]	RW	数据开始位选择 0 = 数据发送从低位LSB开始，到高位MSB结束 1 = 数据发送从高位MSB开始，到低位LSB结束	0'b

18.3.1.6 US_IMSCR (USART中断使能/禁止寄存器)

- Address = Base Address + 0x0014, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																					IDLE	TXEMPTY	TIMEOUT	PARE	FRAME	OVRE	RSVD		RXBRK	TXRDY	RXRDY
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW

Name	Bit	Type	Description	Reset Value
RXRDY	[0]	RW	接收端待机中断 0 = 禁止中断 1 = 使能中断	0
TXRDY	[1]	RW	发送端待机中断 0 = 禁止中断 1 = 使能中断	0
RXBRK	[2]	RW	接收端Break中断 0 = 禁止中断 1 = 使能中断	0
OVRE	[5]	RW	溢出错误中断 0 = 禁止中断 1 = 使能中断	0
FRAME	[6]	RW	帧错误中断 0 = 禁止中断 1 = 使能中断	0
PARE	[7]	RW	校验错中断 0 = 禁止中断 1 = 使能中断	0
TIMEOUT	[8]	RW	超时中断 0 = 禁止中断 1 = 使能中断	0
TXEMPTY	[9]	RW	发送缓冲空闲中断 0 = 禁止中断 1 = 使能中断	0
IDLE	[10]	RW	空间中断 0 = 禁止中断 1 = 使能中断	0

18.3.1.7 US_RISR (USART原始中断寄存器)

- Address = Base Address + 0x0018, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																					IDLE	TXEMPTY	TIMEOUT	PARE	FRAME	OVRE	RSVD		RXBRK	TXRDY	RXRDY
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description	Reset Value
RXRDY	[0]	R	接收端待机中断原始状态 RXRDY的原始中断状态，不管该中断是使能还是禁止	0
TXRDY	[1]	R	发送端待机中断原始状态 TXRDY的原始中断状态，不管该中断是使能还是禁止	0
RXBRK	[2]	R	接收端Break中断原始状态 RXBRK的原始中断状态，不管该中断是使能还是禁止	0
OVRE	[5]	R	溢出错误中断原始状态 OVRE的原始中断状态，不管该中断是使能还是禁止	0
FRAME	[6]	R	帧错误中断原始状态 FRAME的原始中断状态，不管该中断是使能还是禁止	0
PARE	[7]	R	帧错误中断原始状态 PARE的原始中断状态，不管该中断是使能还是禁止	0
TIMEOUT	[8]	R	超时中断原始状态 TIMEOUT的原始中断状态，不管该中断是使能还是禁止	0
TXEMPTY	[9]	R	发送缓冲空闲中断原始状态 TXEMPTY的原始中断状态，不管该中断是使能还是禁止	0
IDLE	[10]	R	空闲中断原始状态 IDLE的原始中断状态，不管该中断是使能还是禁止	0

读操作返回相应中断的原始状态，不管该中断是使能状态还是禁止状态。写操作无任何效果。

18.3.1.8 US_MISR (USART中断状态寄存器)

- Address = Base Address + 0x001C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																					IDLE	TXEMPTY	TIMEOUT	PARE	FRAME	OVRE	RSVD		RXBRK	TXRDY	RXRDY
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description	Reset Value
RXRDY	[0]	R	接收端待机中断状态 RXRDY中断使能后的状态	0
TXRDY	[1]	R	发送端待机中断状态 TXRDY中断使能后的状态	0
RXBRK	[2]	R	接收端Break中断状态 RXBRK中断使能后的状态	0
OVRE	[5]	R	溢出错误中断状态 OVRE中断使能后的状态	0
FRAME	[6]	R	帧错误中断状态 FRAME中断使能后的状态	0
PARE	[7]	R	帧错误中断状态 PARE中断使能后的状态	0
TIMEOUT	[8]	R	超时中断状态 TIMEOUT中断使能后的状态	0
TXEMPTY	[9]	R	发送缓冲空闲中断状态 TXEMPTY中断使能后的状态	0
IDLE	[10]	R	空闲中断状态 IDLE中断使能后的状态	0

读操作返回相应中断使能后的状态，如果该中断被禁止，那么读操作返回的值永远是0。写操作无任何效果。

18.3.1.9 US_ICR (USART中断状态清除寄存器)

- Address = Base Address + 0x0020, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																					IDLE	RSVD	TIMEOUT	PARE	FRAME	OVRE	RSVD		RXBRK	RSVD	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	R	W	W	W	W	R	R	W	R	R

Name	Bit	Type	Description	Reset Value
RXBRK	[2]	W	接收端Break中断状态 0 = 无效 1 = 清除该中断状态	0
OVRE	[5]	W	溢出错误中断状态 0 = 无效 1 = 清除该中断状态	0
FRAME	[6]	W	帧错误中断状态 0 = 无效 1 = 清除该中断状态	0
PARE	[7]	W	帧错误中断状态 0 = 无效 1 = 清除该中断状态	0
TIMEOUT	[8]	W	超时中断状态 0 = 无效 1 = 清除该中断状态	0
IDLE	[10]	W	空闲中断状态 0 = 无效 1 = 清除该中断状态	0

写1清除相应中断状态，写0无效。

18.3.1.10 US_SR (USART状态寄存器)

- Address = Base Address + 0x0024, Reset Value = 0x0000_0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																				IDLEFLAG	IDLE	TXEMPTY	TIMEOUT	PARE	FRAME	OVRE	RSVD			RXBRK	TXRDY	RXRDY
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description	Reset Value
RXRDY	[0]	R	接收端待机 0 = 自从上次读取US_RHR后没有收到任何完成的字节，或者接收端被禁止 1 = 自从上次读取US_RHR后没有收到了至少一个完成的字节	0'b
TXRDY	[1]	R	发送端待机 0 = US_THR中有一个字节正在等待发送到移位寄存器中，或者发送端被禁止 1 = US_THR中没有任何字节 等于0表示USART被禁止了，或者处于复位状态。US_CR中的发送使能命令会将这位置1。	0'b
RXBRK	[2]	R	接收端Break. 0 = 在上一次状态复位后，还没有检测到Break 1 = 在上一次状态复位后，检测到Break	0'b
OVRE	[5]	R	溢出错误 0 = 当RXRDY有效后，没有字节从接收移位寄存器传到US_RHR寄存器 1 = 当RXRDY有效后，至少有一个字节从接收移位寄存器传到了US_RHR寄存器	0'b
FRAME	[6]	R	帧错误 0 = 在上一次状态复位后，没有停止位被检测到低电平 1 = 在上一次状态复位后，至少有一个停止位被检测到低电平	0'b
PARE	[7]	R	校验错误 0 = 在上一次状态复位后，没有检测到校验位错(或者multi-drop模式下的数据字节) 1 = 在上一次状态复位后，检测到至少1个校验位错(或者multi-drop模式下的地址字节)	0'b
TIMEOUT	[8]	R	超时	0'b

Name	Bit	Type	Description	Reset Value
			0 = 开始超时接收后，没有检测到超时，或者超时寄存器被设置为0 1 = 开始超时接收后，检测到了超时	
TXEMPTY	[9]	R	发送缓冲空闲 0 = US_THR寄存器或者发送缓冲寄存器中有字节待发送 1 = US_THR寄存器或者发送缓冲寄存器中没有字节待发送 当USART被禁止或者复位后，该位为0，US_CR中的发送使能功能会将该位置1。	0'b
IDLE	[10]	R	空闲状态 0 = 没有检测到J1587的结束帧 1 = 检测到J1587的结束帧	0'b
IDLEFLAG	[11]	R	0 = USART正在接收一个帧 1 = USART没有在接收任何帧 该位表示J1587协议的帧传送状态，当接收开始时变低，在接收完成+10个停止位(10个周期的高电平)后变高。	1'b

18.3.1.11 US_RHR (USART接收数据寄存器)

- Address = Base Address + 0x0028, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
RXCHR	[8:0]	R	接收到的字节 当RXRDY有效时，储存接收到的字节。当位数小于9时，数据为右对齐。	0x000

注意：
读取此寄存器后，RXRDY位会被自动清除。在调试模式，用户可以使用镜像寄存器来避免RXRDY被清除。

18.3.1.12 US_THR (USART发送数据寄存器)

- Address = Base Address + 0x002C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
TXCHR	[8:0]	W	需要发送的字节 当TXRDY有效时，存储下一个要发送的字节。如果TXRDY是0，那么当前US_THR寄存器的值会被覆盖。当位数小于9时，数据为右对齐。	0x000

18.3.1.13 US_BRGR (USART波特率配置寄存器)

- Address = Base Address + 0x0030, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
FRACTION	[3:0]	RW	小数修正值 同步模式下如果选择了外部时钟，此寄存器无效。 *异步模式)* *FRACTION = ROUND[((Fclk / (Baud Rate x 16)) - CD) x 16, 0]* *同步模式)* *FRACTION = ROUND[((Fclk / Baud Rate) - CD) x 16, 0]* **FRACTION[3:0]** / **Action** 0 to 15 / 波特率(异步模式) = Fclk / (16 x (CD + FRACTION/16)) 波特率(同步模式) = Fclk / (CD + FRACTION/16)	0x0000
CD	[15:4]	RW	分频 同步模式下如果选择了外部时钟，此寄存器无效。 *异步模式) CD = FLOOR[Fclk / (Baud Rate x 16), 1]* *同步模式) CD = FLOOR[Fclk / Baud Rate, 1]* **CD[15:4]** / **Action** 0 / 关闭时钟 1 / 无分频 (1分频) 2 to 65535 / 波特率(异步模式) = Fclk /(16 × (CD+FRACTION/16)) 波特率(同步模式) = Fclk /(CD+FRACTION/16)	0x0000

注意：*Fclk 是 USART 模块的输入时钟，波特率是通讯速度。*

注意：
在同步模式，为了保证50:50的1/0占空比，这个寄存器的值必须为偶数，并且用户在US_MR寄存器中配置波特率时钟后必须使能时钟，才能产生波特率时钟。
当使用内部时钟(PCLK)的时候，不可以使用CD = 1。

18.3.1.14 US_RTOR (USART接收超时配置寄存器)

- Address = Base Address + 0x0034, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
TO	[15:0]	RW	超时配置 给这个寄存器写值后，会自动开启超时接收的指令 • 超时配置位 TO[15:0] / Action 0 / 禁止接收端的超时功能 1–65565 / 当开启超时接收时，或者每当收到一个数据字节时，超时计数器会被载入TO[15:0]的值 异步模式：超时时长 = TO[15:0] × 位周期 同步模式：超时时长 = TO[15:0] × 16 × 位周期	0x0000

注意：
当设置US_CR寄存器的RXDIS位禁止接收端后，超时功能被停止，这时如果又通过US_CR的RXEN位重新使能了接收端，那么超时计数器会从刚才停止的地方继续开始(不会被复位)。

18.3.1.15 US_TTGR (USART发送端Time-Guard配置寄存器)

- Address = Base Address + 0x0038, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
TG	[7:0]	RW	Time-Guard配置 • Time-Guard配置位 TO[15:0] / Action: 0 – 禁止发送端的time-guard功能 1–255 – USARTTX在每发送完一个字节后，会变高一段时间，这个时间段为time-guard时长 Time-guard时长 = TG[7:0] × 位周期	0x00

4MHz–40MHz Asynchronous Mode (SYNC = 0)

Table 1-4 Asynchronous Mode (SYNC = 0)

SYSCLK	PDIV	PCLK	US_BRGR CD[15:0]	Baud Rate	Result	% Error
4MHz	1	4	208	1200	1201.92	–0.16%
			104	2400	2403.85	–0.16%
			52	4800	4807.69	–0.16%
			26	9600	9615.38	–0.16%
			13	19200	19230.77	–0.16%
	2	2	104	1200	1201.92	–0.16%
			52	2400	2403.85	–0.16%
			26	4800	4807.69	–0.16%
			13	9600	9615.38	–0.16%
	4	1	52	1200	1201.92	–0.16%
			26	2400	2403.85	–0.16%
			13	4800	4807.69	–0.16%
	8	0.5	26	1200	1201.92	–0.16%
			13	2400	2403.85	–0.16%
	16	0.25	13	1200	1201.92	–0.16%
8MHz	1	8	417	1200	1199.04	0.08%
			208	2400	2403.85	–0.16%
			104	4800	4807.69	–0.16%
			52	9600	9615.38	–0.16%
			26	19200	19230.77	–0.16%
			13	38400	38461.54	–0.16%
	2	4	208	1200	1201.92	–0.16%
			104	2400	2403.85	–0.16%
			52	4800	4807.69	–0.16%
			26	9600	9615.38	–0.16%
			13	19200	19230.77	–0.16%
	4	2	104	1200	1201.92	–0.16%
			52	2400	2403.85	–0.16%
			26	4800	4807.69	–0.16%
			13	9600	9615.38	–0.16%
	8	1	52	1200	1201.92	–0.16%
			26	2400	2403.85	–0.16%
			13	4800	4807.69	–0.16%

SYSCLK	PDIV	PCLK	US_BRGR CD[15:0]	Baud Rate	Result	% Error
16MHz	16	0.5	26	1200	1201.92	–0.16%
			13	2400	2403.85	–0.16%
	1	16	833	1200	1200.48	–0.04%
			417	2400	2398.08	0.08%
			208	4800	4807.69	–0.16%
			104	9600	9615.38	–0.16%
			52	19200	19230.77	–0.16%
			26	38400	38461.54	–0.16%
	2	8	417	1200	1199.04	0.08%
			208	2400	2403.85	–0.16%
			104	4800	4807.69	–0.16%
			52	9600	9615.38	–0.16%
			26	19200	19230.77	–0.16%
			13	38400	38461.54	–0.16%
	4	4	208	1200	1201.92	–0.16%
			104	2400	2403.85	–0.16%
			52	4800	4807.69	–0.16%
			26	9600	9615.38	–0.16%
			13	19200	19230.77	–0.16%
	8	2	104	1200	1201.92	–0.16%
			52	2400	2403.85	–0.16%
			26	4800	4807.69	–0.16%
			13	9600	9615.38	–0.16%
	16	1	52	1200	1201.92	–0.16%
			26	2400	2403.85	–0.16%
			13	4800	4807.69	–0.16%
20MHz	1	20	1042	1200	1199.62	0.03%
			521	2400	2399.23	0.03%
			260	4800	4807.69	–0.16%
			130	9600	9615.38	–0.16%
			87	14400	14367.82	0.22%
			65	19200	19230.77	–0.16%
	2	10	521	1200	1199.62	0.03%
			260	2400	2403.85	–0.16%
			130	4800	4807.69	–0.16%
			65	9600	9615.38	–0.16%

SYSCLK	PDIV	PCLK	US_BRGR CD[15:0]	Baud Rate	Result	% Error
40MHz	4	5	260	1200	1201.92	–0.16%
			130	2400	2403.85	–0.16%
			65	4800	4807.69	–0.16%
	8	2.5	130	1200	1201.92	–0.16%
			65	2400	2403.85	–0.16%
	16	1.25	65	1200	1201.92	–0.16%
	1	40	2083	1200	1200.19	–0.02%
			1042	2400	2399.23	0.03%
			521	4800	4798.46	0.03%
			260	9600	9615.38	–0.16%
			174	14400	14367.82	0.22%
			130	19200	19230.77	–0.16%
			65	38400	38461.54	–0.16%
	2	20	1042	1200	1199.62	0.03%
			521	2400	2399.23	0.03%
			260	4800	4807.69	–0.16%
			130	9600	9615.38	–0.16%
			87	14400	14367.82	0.22%
			65	19200	19230.77	–0.16%
	4	10	521	1200	1199.62	0.03%
			260	2400	2403.85	–0.16%
			130	4800	4807.69	–0.16%
			65	9600	9615.38	–0.16%
	8	5	260	1200	1201.92	–0.16%
			130	2400	2403.85	–0.16%
			65	4800	4807.69	–0.16%
	16	2.5	130	1200	1201.92	–0.16%
			65	2400	2403.85	–0.16%

19 通用异步收发器（UART）

19.1 概述

UART是一个简单通用的异步串行接收和发送接口，支持8位的数据通信，不支持校验位，每次发送都以一个停止位结束。

19.1.1 主要特性

- 可配置的波特率
- 固定的8位发送长度
- 发送接收溢出检测
- 发送接收完成中断和溢出中断
- 支持4种校验位，奇偶校验和0/1校验

19.1.2 管脚描述

Table 13-1 UART 管脚描述

管脚名称	功能	I/O类型	有效电平	说明
UART_RX	UART发送数据线	O	–	–
UART_TX	UART接收数据线	I	–	–

19.2 功能描述

19.2.1 模块框图

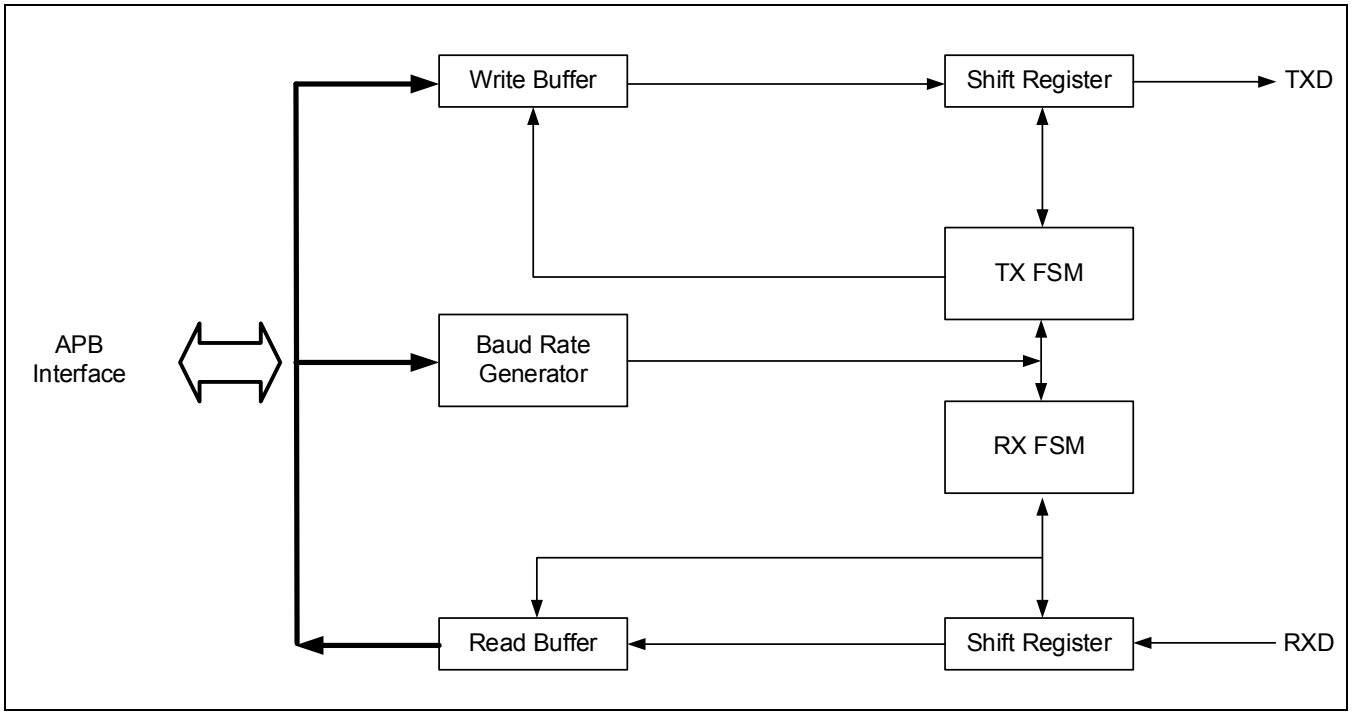


Figure 13-1 UART模块框图

19.2.2 功能说明

19.2.2.1 波特率的产生

波特率产生电路可以给发送和接收电路产生波特率时钟。在使用 UART 前必须设置波特率分频寄存器(UART_BRDIV 的 DIV 位)，计算公式如下：

波特率 = PCLK / DIV

例如，如果 PCLK 是 12MHz，需要的波特率为 9600，那么用户必须将 UART_BRDIV 寄存器设为：
12,000,000/9600 = 1250

Table 13-2 波特率设置示例

PCLK	DIV	Baud Rate	% Error
20	2083	9600	0.02%
	1042	19200	-0.03%
	521	38400	-0.03%
	174	115200	-0.22%
16	1667	9600	-0.02%
	833	19200	0.04%
	417	38400	-0.08%
	139	115200	-0.08%
12	1250	9600	0.00%
	625	19200	0.00%
	313	38400	-0.16%
	104	115200	0.16%
8	833	9600	0.04%
	417	19200	-0.08%
	208	38400	0.16%
	69	115200	0.64%

19.2.2.2 接收

UART 通过检测 RXD 信号来判断接收字节的起始位。如果 RXD 上的低电平超过 7 个采样时钟的周期，那么这个低电平则被认为是有效的起始位。采样时钟的频率为波特率的 16 倍。所以长于 7/16 采样周期的低电平为有效，而比 7/16 个采样周期短的低电平则会被忽略，忽略后 UART 会继续等待有效的起始位。

当检测到一个有效的起始位，接收端开始在理论上每位的中心点读取 RXD 信号。假设每个数据位有 16 个采样周期的宽度，那么采样点则在起始位后的第 8 个采样周期(0.5 个数据位)处。所以第一个采样点是在 RXD 下降沿后的第 24 个采样周期(1.5 个数据位)时，之后每个采样点则每隔 16 个采样周期(1 个数据位)。

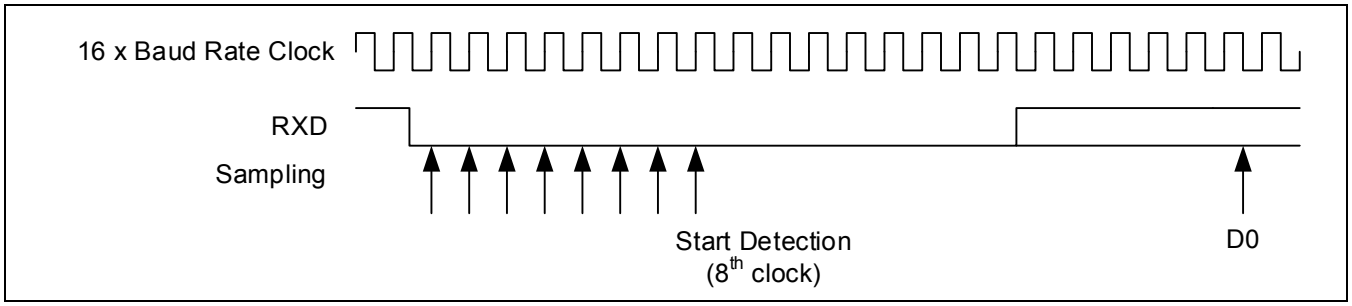


Figure 13-2 起始位检测

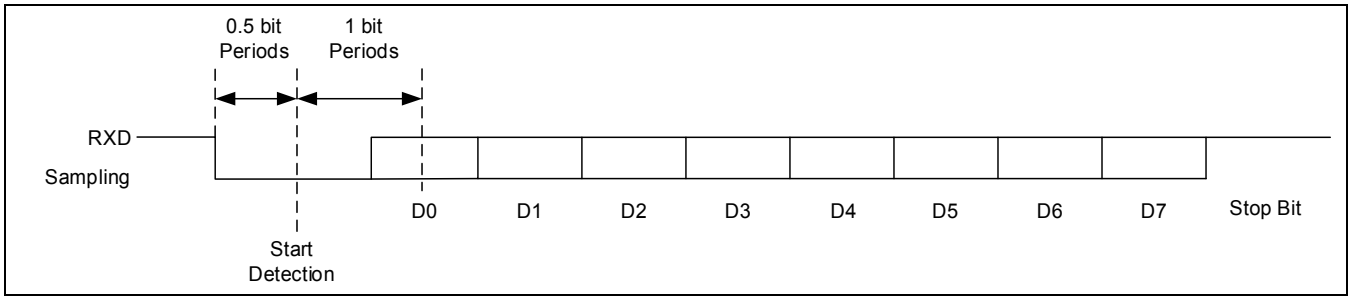


Figure 13-3 接收数据

19.2.2.3 发送

发送过程中，起始位，数据位和停止位按顺序被移出，最低位(LSB)优先。需要发送数据时，先将 UART 模块使能(UART_CTRL 中的 TX 使能位)，再将数据写入数据寄存器(UART_DATA)即可。当写完数据寄存器 UART_DATA 后，数据会被立即发送出去。

Figure 13-4 数据发送

19.2.2.4 校验位

UART_CTRL寄存器中的PARITY位用来设置校验方式。PARITY的第2位PARITY[2]如果为0，那么校验位被禁止，发送和接收都没有校验位。如果PARITY[2]为1，则校验位使能，根据PARITY[1:0]的设置，校验的模式不同：

PARITY[1:0]为00： 偶校验，数据位(8位)与校验位中1的个数为偶数。

PARITY[1:0]为01： 奇校验，数据位(8位)与校验位中1的个数为奇数。

PARITY[1:0]为10 : 0校验，校验位一直为0。

PARITY[1:0]为11 : 1校验，校验位一直为1。

19.2.2.5 中断

当接收到一个数据或者发送完一个数据后，状态寄存器 UART_SR 中的相应位会被置 1。如果收到的数据没有来得及被 CPU 读取而又再收到另一个数据时，或者如果当前数据还没发送完 CPU 就又往 UART_DATA 里写数据，那么 UART_SR 中的溢出位将会被置 1。

如果相应的中断被使能，那么 UART_ISR 里的寄存器也会被置位，同时 CPU 将会收到中断请求。

19.3 寄存器说明

19.3.1 寄存器表

Base Address: 0x4008_1000

Offset Address	Name	Description	R/W	Reset State
0x000	UART_DATA	数据寄存器	R/W	0x00000000
0x004	UART_SR	状态寄存器	R/W	0x00000000
0x008	UART_CTRL	控制寄存器	R/W	0x00000000
0x00C	UART_ISR	中断状态寄存器	R/W	0x00000000
0x010	UART_BRDIV	波特率分频寄存器	R/W	0x00000000

19.3.1.1 UART_DATA (数据寄存器)

- Address = Base Address + 0x0000

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

Name	Bit	Type	Description	Reset Value
DATA	[7:0]	R/W	发送或接收到的数据 读 = 接收到的数据 写 = 发送的数据	-

19.3.1.2 UART_SR (状态寄存器)

- Address = Base Address + 0x0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																											PARITY_ERR	RX_OVER	TX_OVER	RX_FULL	TX_FULL
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	R	R

Name	Bit	Type	Description	Reset Value
TX_FULL	[0]	R	TX缓冲区状态 0 = TX缓冲区没有满(可以发送数据) 1 = TX缓冲区已满(正在发送数据)	0
RX_FULL	[1]	R	RX缓冲区状态 0 = RX缓冲区没有满(未收到数据或数据已被读取) 1 = RX缓冲区已满(收到数据，并且未被读取)	0
TX_OVER	[2]	R/W	TX缓冲区溢出状态 0 = TX缓冲区没有溢出 1 = TX缓冲区溢出(读取) 1 = 清除TX缓冲区溢出标志(写)	0
RX_OVER	[3]	R/W	RX缓冲区溢出状态 0 = RX缓冲区没有溢出 1 = RX缓冲区溢出(读取) 1 =清除RX缓冲区溢出标志(写)	0
PARITY_ERR	[4]	R/W	校验状态 0 = 没有校验错误 1 = 校验错误(读取) 1 = 清除校验错误 (写)	0

19.3.1.3 UART_CTRL (控制寄存器)

- Address = Base Address + 0x0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																					PARITY			INT_PARITY	TEST	INT_OVER_RX	INT_OVER_TX	INT_RX	INT_TX	RX	TX
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
TX	[0]	R/W	TX 使能/禁止 0 = 禁止TX 1 = 使能TX	0
RX	[1]	R/W	RX 使能/禁止 0 = 禁止RX 1 = 使能RX	0
INT_TX	[2]	R/W	TX 中断使能/禁止 0 = 禁止TX中断 1 = 使能TX中断	0
INT_RX	[3]	R/W	RX 中断使能/禁止 0 = 禁止RX中断 1 = 使能RX中断	0
INT_OVER_TX	[4]	R/W	TX溢出中断使能/禁止 0 = 禁止TX溢出中断 1 = 使能TX溢出中断	0
INT_OVER_RX	[5]	R/W	RX溢出中断使能/禁止 0 = 禁止RX溢出中断 1 = 使能RX溢出中断	0
TEST	[6]	R/W	测试模式 此为请保持为0	0
INT_PARITY	[7]	R/W	校验错误中断使能/禁止 0 = 禁止校验错误中断 1 = 使能校验错误中断	0
PARITY	[10:8]	R/W	校验位类型 0XX：无校验位 100：偶校验	0

			101： 奇校验 110 : 0校验，校验位一直为0 (Space) 111 : 1校验，校验位一直为1 (Mark)	

19.3.1.4 UART_ISR (中断状态寄存器)

- Address = Base Address + 0x000C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																											PARITY_ERR	RX_OVER_INT	TX_OVER_INT	RX_INT	TX_INT
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
TX_INT	[0]	RW	TX中断 0 = TX中断没发生 1 = TX中断发生(读取) 1 = 清除TX中断(写)	0
RX_INT	[1]	RW	RX中断 0 = RX中断没发生 1 = RX中断发生(读取) 1 = 清除RX中断(写)	0
TX_OVER_INT	[2]	RW	TX溢出中断 0 = TX溢出中断没发生 1 = TX溢出中断发生(读取) 1 = 清除TX溢出中断(写)	0
RX_OVER_INT	[3]	RW	RX溢出中断 0 = RX溢出中断没发生 1 = RX溢出中断发生(读取) 1 = 清除RX溢出中断(写)	0
PARITY_ERR	[4]	R/W	校验错误中断 0 = 校验错误中断没发生 1 = 校验错误中断发生(读取) 1 = 清除校验错误中断(写)	0

19.3.1.5 UART_BRDIV (波特率分频寄存器)

- Address = Base Address + 0x0010, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
DIV	[19:0]	RW	波特率分频 最小值为16	0x00000

20 I2C总线

20.1 概述

I2C总线是一个由数据(SDA)和时钟(SCL)组成的两线同步串行接口。每个接在总线上器件都可以被一个唯一的地址寻址。SDA和SCL为双向接口，通过一个上拉电阻接到正向电源。接到总线上的器件输出必须设置成开漏模式以实现”线与”的功能。

I2C总线是一个真正的多主机总线，因为它包含了冲突检测和仲裁，在多主机同时启动数据传输时可以避免数据丢失。时钟的同步通过I2C接口和SCL之间线与的方式实现。

I2C接口可以工作在快速模式和标准模式。快速模式支持的波特率范围为0到400Kbit/s，标准模式支持的波特率范围为0到100Kbit/s。该模块支持4种模式：主机发送，主机接收，从机发送，从机接收；支持7位寻址和10位寻址，并且支持检测本机地址功能和General Call寻址功能(从机模式)。

20.1.1 主要特性

- 多主机总线
- 串行，8位的双向数据传输
- 支持标准模式的100Kbit/s，快速模式最高支持400Kbit/s

20.1.2 管脚描述

Table 20-1 I2C 管脚描述

Pin Name	Function	I/O Type	Active Level	Comments
SDA	串行数据线	I/O	高有效	–
SCL	串行时钟线	I/O	高有效	–

20.2 功能描述

20.2.1 模块框图

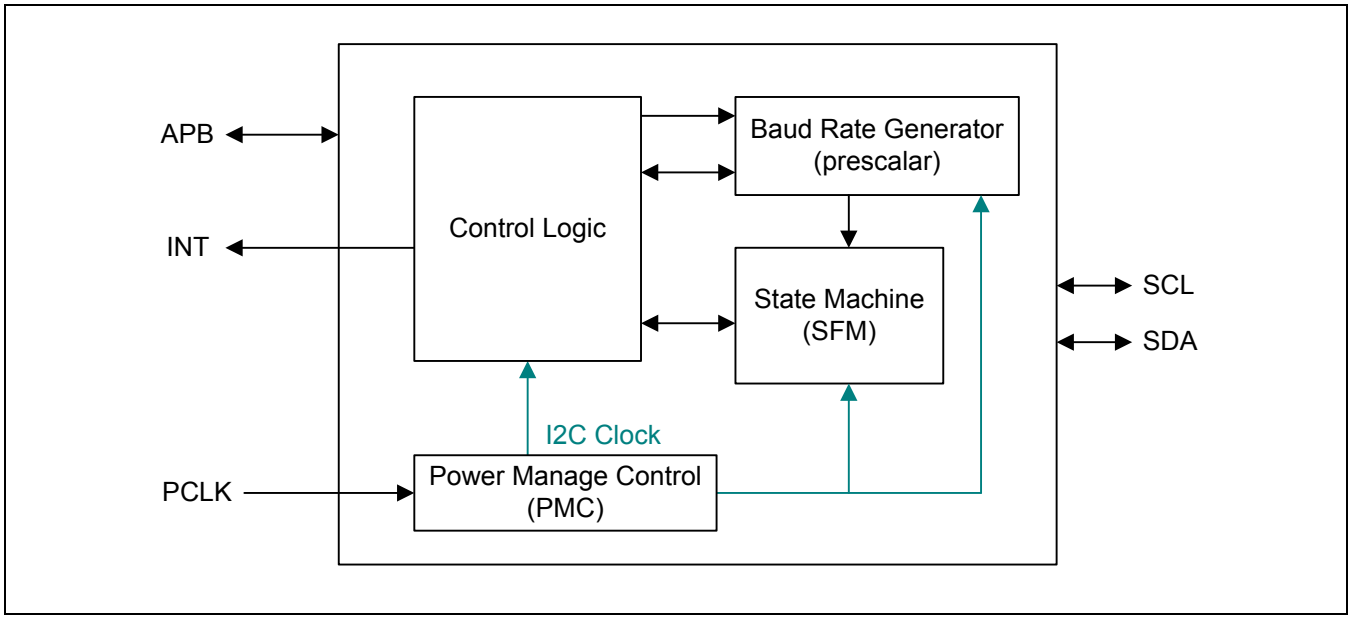


Figure 20-1 I2C模块框图

20.2.2 功能介绍

20.2.2.1 I2C总线概念

20.2.2.1.1 协议概念

串行数据 SDA 和串行时钟 SCL 这两根线能够在连接到它们的器件之间传输数据。每个器件都有一个唯一的地址，不管该器件是单片机，LCD 驱动芯片，存储芯片还是键盘接口，根据所需功能的不同，都可以作为一个发送端或者一个接收端。显然 LCD 驱动只能作为接收端，而存储芯片既能接收也能发送数据。除了作为发送端和接收端，在进行数据传输时，I2C 的器件也可以被称作主机或者从机。主机是一个可以在总线上发起数据传输，并且产生时钟信号来完成该传输的器件，在这个时候，任何被寻址到的器件都被当作是一个从机。

I2C总线是一个支持多主机的总线，意思是可以连接很多具有控制总线功能的器件。由于主机通常都是单片机，让我们以I2C总线上的两个单片机为例。要注意这些关系并不是永久性的，因为这个关系跟数据传输的方向有关。数据的传输过程如下：

1: 假设单片机A希望给单片机B发送信息

- 单片机A(主机)寻址单片机B(从机)
- 单片机A(主机-发送端)把数据发给单片机B(从机)
- 单片机A结束该传输

2: 如果单片机A希望从单片机B接收数据

- 单片机A(主机)寻址单片机B(从机)
- 单片机A(主机-接收端)从单片机B(从机-发送端)接收数据
- 单片机A结束该传输

即使在这种情况下(上面情况2)，数据的传输也是由主机(单片机A)来产生时钟并且结束。

能够把多个单片机接到I2C总线上的意思就是总线支持多个主机同时发起数据传输。为了避免混乱，I2C支持总线仲裁机制，这个机制依赖于I2C总线上所有I2C接口的线与连接。

如果2个或者多个主机尝试发起数据传输，那么第一个成功产生“1”的主机将获得发送权而其它为成功产生“1“的主机则失去发送权。仲裁过程中的时钟信号是由线与连接到SCL的主机时钟信号经过同步逻辑产生的。

时钟信号总是由主机来负责产生和发送；每个主机在传输数据的时候，都是主机自己来产生时钟信号。只有当慢速从机拉低时钟线的时候，或者当仲裁发生时其它主机拉低了时钟线的时候，主机产生的时钟信号才会被改变。

20.2.2.1.2 一般特性

SDA和SCL都是双向传输线，通过一个上拉电阻接到正向的电源电压。当总线空闲时，两个信号都是高电平状态。连接到总线上器件的输出都必须设置成开漏输出以支持线与的功能。I2C总线的数据传输在标准模式下可以到100Kbit/s，而在快速模式下则高达400Kbit/s。接在总线上每个接口的寄生电容不能超过400pF。

下表列出了一些寄存器设置对应的波特率。波特率的快慢跟I2C时钟，快速模式和I2C_MR寄存器里的PRV位有关。

Table 20-2 波特率设置示例

I2C Clock	PRV	Baud Rate	FAST	% Error
20	204	96000	0	–0.16%
	156	125000	1	0.00%
	100	192000	1	–0.16%
	48	384000	1	–0.16%
18	184	96000	0	0.27%
	140	125000	1	0.00%
	90	192000	1	0.27%
	43	384000	1	0.27%
37.5	387	96000	0	0.10%
	296	125000	1	0.00%
	191	192000	1	–0.16%
	94	384000	1	0.35%
18.75	191	96000	0	–0.16%
	146	125000	1	0.00%
	94	192000	1	0.35%
	45	384000	1	0.35%
10	100	96000	0	–0.16%
	76	125000	1	0.00%
	48	192000	1	–0.16%
	22	384000	1	–0.16%
9.375	94	96000	0	0.35%
	71	125000	1	0.00%
	45	192000	1	0.35%
4.6875	45	96000	0	0.35%

20.2.2.2 位传输

由于各种不同工艺的器件(CMOS, NMOS, bipolar)都能连接在I2C总线上，所以逻辑0和1的电平是不确定的，跟VDD的电平有关。每个时钟脉冲传输1位数据。

20.2.2.2.1 数据有效性

SDA传输线的数据必须要在时钟信号为高的期间保持不变。数据线的高低状态转换必须发生在SCL为低电平的期间。

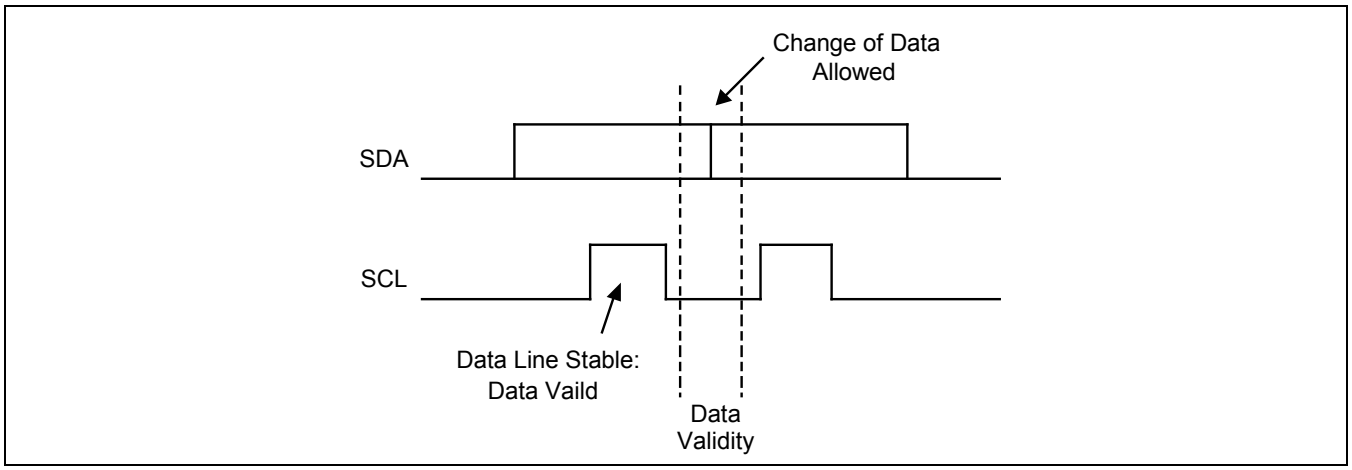


Figure 20-2 数据有效性

20.2.2.2.2 起始位和停止位

在I2C总线的传输过程中，一些特殊的情况被定义成起始位和停止位。

当SCL是高的时候，SDA从高变低，被定义为起始位。

当SCL是高的时候，SDA从低变高，被定义为停止位。

起始位和停止位都是由主机产生的。在起始位产生以后，总线被认为是处于工作状态(BUSY)，直到停止位产生后总线则被认为是处于空闲状态。

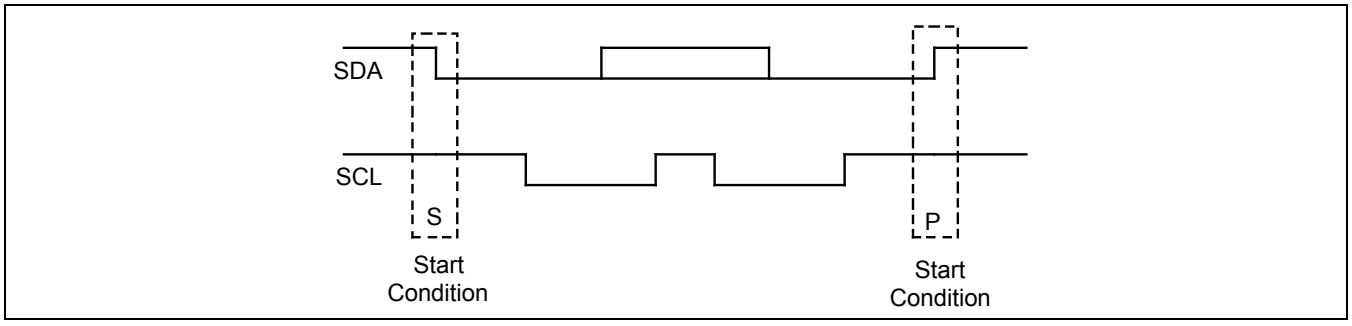


Figure 20-3 起始位和停止位

20.2.2.3 数据传输

20.2.2.3.1 传输字节格式

SDA上传输的每个字节的长度为8位。每次传输字节的总个数没有限定，也就是说理论上可以传输无限个字节的数据。每个字节传输完后，紧接着会有一个应答位。数据的最高位先发送(MSB优先)。如果接收端在它完成某个其它任务前无法接收时，例如在处理中断服务程序时，接收端可以拉低SCL信号线强制让发送端进入等待状态。当接收端准备好后则释放SCL信号线，之后数据传输继续。

某些特殊情况下，允许使用与I2C总线不同的数据格式(例如兼容CBUS的器件)。这种特殊情况下的数据传输即使在一个字节的传输当中，也可以由停止位来终止，不需要应答位。

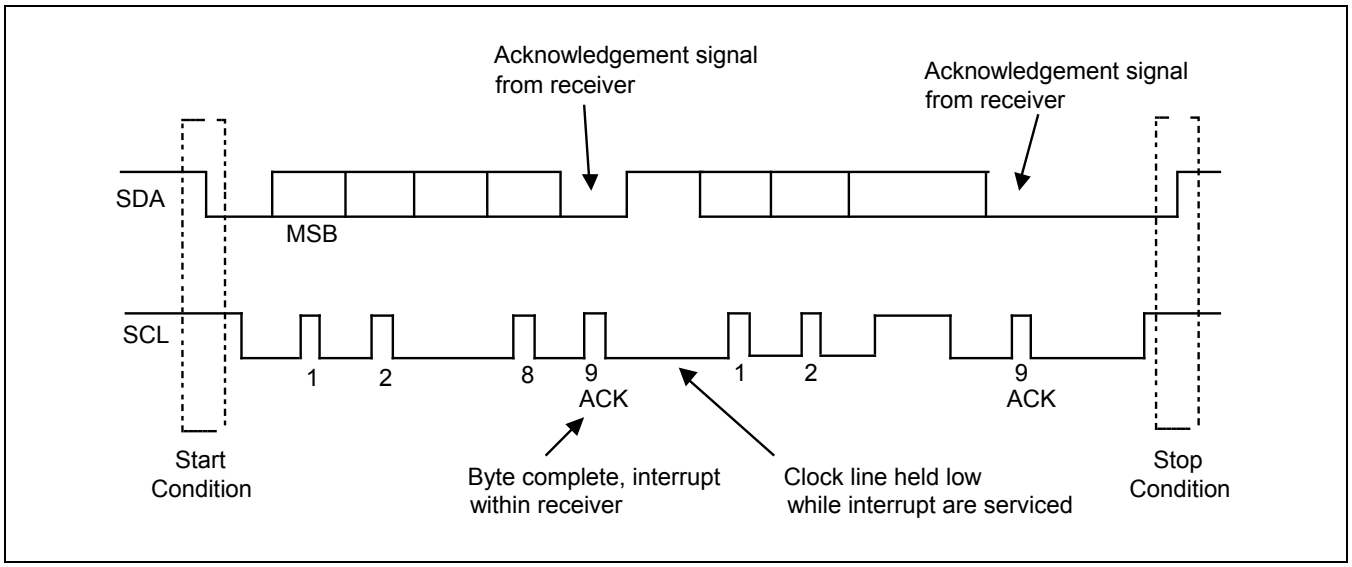


Figure 20-4 I2C总线的数据传输

20.2.2.3.2 应答

带应答机制的数据传输在I2C协议中是必须的。应答信号需要的时钟脉冲是由主机来产生的。发送端在应答时钟脉冲宽度内，释放SDA信号线(高电平)的控制权，也就是不输出。

接收端必须在应答时钟脉冲期间内拉低SDA信号线，并且在这个时钟为高的期间一直保持低。当然，注意setup和hold时间也必须计算入内。

通常接收端在收到每个字节后都必须发送一个应答信号，除非该传输是CBUS的地址。

当从机-接收端无法应答从机地址时(比如正在处理一些实时任务)，从机必须将数据线拉高。这时主机可以产生一个停止位，终止该传输。

如果从机-接收端应答了从机地址，但是在一段时间后的传输中无法再接收更多的数据了，这时主机必须再次终止传输。也就是说，从机在第一个字节传输后的应答位上发送一个“非应答”，在应答时钟脉冲周期内让数据线保持高电平，这样主机就会产生一个停止位。

主机-接收端在数据传输时，通过不发送最后一个字节的应答信号，告诉从机-发送端该传输已经结束。从机-发送端则必须释放数据线，让主机来产生停止位或者重复开始位。

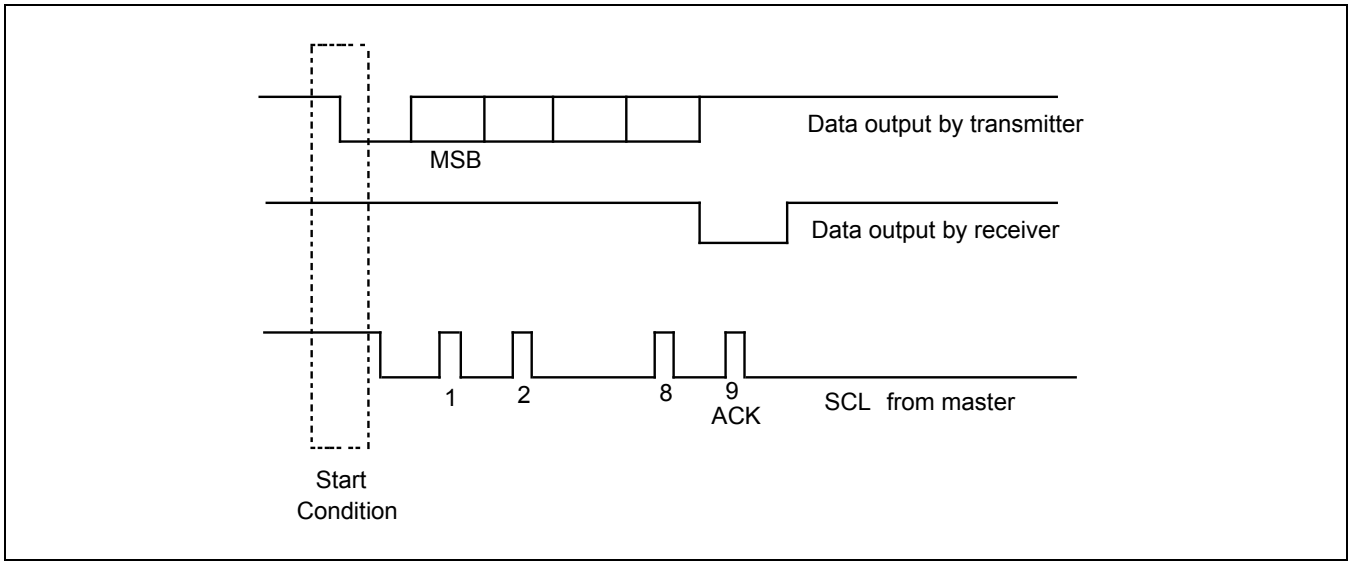


Figure 20-5 应答

20.2.2.4 时钟仲裁

20.2.2.4.1 同步

所有主机在I2C总线上传输数据的时候，都会产生它们自己的时钟。数据只有在时钟电平为高的时候有效。所以需要一个统一的时钟，以完成仲裁。

时钟同步利用连接到I2C接口的SCL线与功能实现。SCL上一个高到低的下降沿会让所有器件的低电平计数器开始计数，并且一旦有一个器件输出低电平，那么它就会拉低SCL直到时钟变高电平。然而，如果有其它时钟仍然处于输出低的状态，那么这个时钟的低到高的跳变并不会影响SCL的低输出。也就是说，SCL的低电平会保持低电平时间最长的那个时钟所输出的低，这时候更短低电平的时钟则进入一个等待高电平的状态。当所有器件的低电平都输出完以后，时钟信号变高。这时所有器件的时钟和SCL信号线之间就没有任何不同了，并且所有器件都开始输出高电平。第一个把高电平输出完的器件，会再次将SCL信号拉低。

在这个同步方法下，同步时钟的低电平由最长低电平周期的那个时钟产生，而高电平则由最短高电平的那个时钟产生。

20.2.2.4.2 仲裁

只有当总线空闲的时候，主机才可以发起一个传输。两个或多个主机有可能同时产生起始位，这时就需要仲裁。

仲裁发生在SCL是高的SDA传输线上，当某个主机A发送高电平的时候，其它主机正在发送低电平，那么主机A会检测到SDA上并不是它发送的电平，于是主机A中断它的数据输出，也就是丢失了仲裁。

仲裁可以在多个传输阶段上发生。第一个仲裁阶段是地址位的比较。如果多个主机都在同时寻址同一个器件，那么仲裁会继续在数据传输阶段发生。由于地址和数据都会被用来仲裁，所以传输过程中不会有信息丢失。

失去仲裁的主机会在丢失仲裁的那个字节传输中一直产生时钟脉冲。

如果一个主机还有从机功能并且在寻址阶段失去了仲裁，那么有可能赢得仲裁的主机正在寻址它。所以这个失去仲裁的主机应该马上转换成从机-接收模式。

由于I2C总线的控制权是单独由竞争主机发生的地址和数据决定的，所以总线没有中央主机，也没有任何优先权的机制。

特别要注意的一点，如果在一个串行传输中，仲裁发生在重复起始位或者停止位发送到I2C总线的瞬间，那么参与仲裁的主机需要在相同位置发送重复起始位或者停止位。也就是说，仲裁不允许发生在下面两个情况中间：

- 重复起始位和数据位
- 停止位和数据位
- 重复起始位和停止位

20.2.2.4.3 使用时钟同步机制作为握手

时钟的同步机制，除了可以在仲裁过程中使用，还可以用来让慢速的接收端与快速的发送端协同工作，支持字节协同和位协同。

对于字节协同工作的情况，慢速的器件可以用快的速度来接收传输的数据，但是需要时间来存储接收的字节或者准备另一个需要发送的字节。这种情况下，从机在收到和应答该字节后，拉低SCL，强制让主机进入等待状态，直到从机准备好下个字节的传输为止。

对于位协同的情况，比如一个单片机没有硬件I2C或者只有一个功能不全的I2C，那么它可以使用扩展时钟低电平时长的办法来降低传输速度，这样主机的速度就会自动适应为该单片机内部的速度。

20.2.2.4.4 7位寻址格式

起始位(S)后，发送的是从机地址。从机地址的长度为7位，第8位为数据方向位(读/写)——0表示发送(写)，1表示读请求(读)。数据传输总是由主机产生的停止位(P)来终止。但是，如果主机希望继续通信，那么它可以产生一个重复起始位(Sr)并且寻址其它从机，而不需要先产生一个停止位。各种读写格式的组合可以在这个传输中发生。

可以传输的格式为：

- 主机-发送端给从机-接收端发送数据。传输方向没有改变。
- 主机在第一个字节后，向从机读取数据

在第一个应答时刻，主机-发送端变成一个主机-接收端，而从机-接收端则变成一个从机-发送端。这个应答仍然由从机产生。

停止位由主机产生。

- 组合格式。在一个有方向变化的传输中，起始位和从机地址都会被重发，但是保留读写位。如果主机发送了重复起始位，那么之前它肯定发送了非应答位。

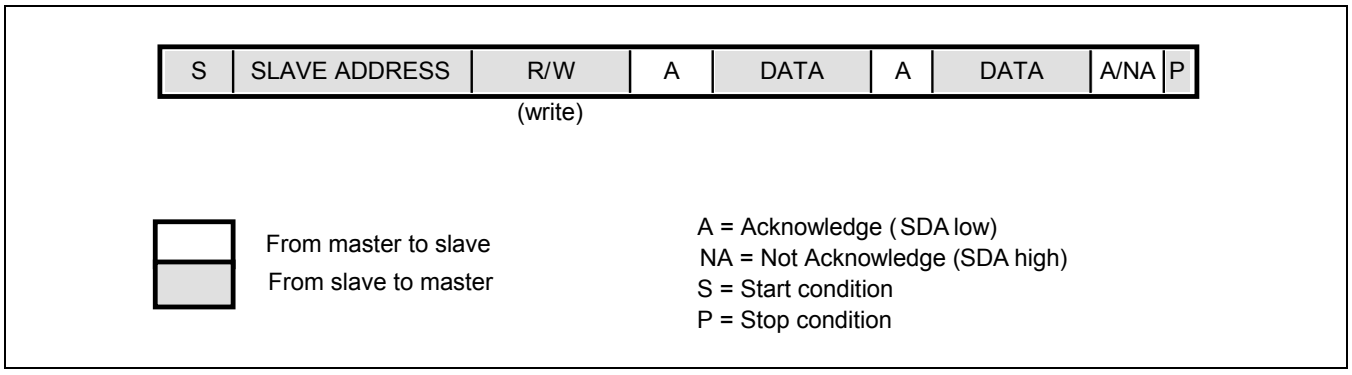


Figure 20-6 主机-发送端寻址从机

20.2.2.5 7位寻址

I2C总线的寻址过程是起始位后的第一个字节通常决定了主机要选择哪个从机。例外是“general call”寻址，可以寻址所有总线上的器件。当使用了这个地址时，总线上所有器件理论上都应该响应。然而，器件也可以设置成忽略该地址。”General call“的第二个字节则定义了接下来需要进行的操作。

20.2.2.5.1 定义第一个字节的各个位

第一个字节的前7位构成了从机地址。第8位是最低位LSB(least significant bit)，定义数据传输的方向。第8位LSB为0表明主机要向某个从机发送数据，而LSB为1则表明主机要从某个从机读数据。

当地址被发送后，系统里的每个器件在起始位后，都会将自己的地址和发送的地址进行比较，如果地址匹配，该器件就认为自己被主机选中为从机-接收端或者从机-发送端。是接收端还是发送端依赖于第8位读写位。

从机地址可以由一个固定部分和一个可编程部分组成。由于系统中很有可能存在一些相同地址的器件，所以从机地址中的可编程部分可以让这些器件尽可能的多。器件可编程地址的位数由器件中可用管脚的数量决定。例如，如果一个器件有4个固定地址位和3个可编程地址位，那么总共8个相同固定地址位的器件可以接到同一个I2C总线上。

I2C总线协议委员会负责协调I2C地址的分配。

两组共8个地址(0000XXX和1111XXX)保留为特殊用途，如下*Table 20-3*. 11110XX 的组合保留给10位寻址使用。

Table 20-3 第一个字节定义

从机地址	读写位	描述
0000 000	0	General call地址
0000 000	1	起始位(1)
0000 001	X	CBUS地址(2)
0000 010	X	保留给不同的总线格式(3)
0000 011	X	保留给将来使用
0000 1XX	X	HS模式主机代码
1111 1XX	X	保留给将来使用
1111 0XX	X	10位寻址

注意：

1. 所有器件都不允许在收到起始位后就应答。
2. CBUS 地址保留给 CBUS 兼容的器件和 I2C 总线兼容的器件混合使用。I2C 总线的器件收到该地址后不允许响应。
3. 该地址保留给其它不同总线。只有可以工作在这种总线和协议下的器件允许响应该地址。

General call地址用来寻址I2C总线上的每一个器件，但是如果一个器件不需要任何General call的数据，那么它可以通过不发送应答位来忽略该地址。如果一个器件确实需要从general call地址获取数据，那么它可以应答该地址并且以从机-接收端工作。

第二个字节和后面的字节都会被能处理该数据的从机-接收端应答。

如果不能处理这些字节中的某个字节，从机必须通过不发送应答位来忽略它。General call地址的功能，由第二个字节来指定。

需要考虑两种情况：

- LSB最低位B是0
- LSB最低位B是1

当最低位B是0，那么第二个字节有以下定义：

- 00000110 (H’06’). 复位并且由硬件写入从机地址的可编程部分。收到这个2-字节序列后，所有设计好能响应general call地址的器件将会复位，并且接收它们地址中的可编程部分。注意在上电后一定要保证器件不会拉低SDA和SCL，因为低电平会阻塞总线。
- 00000100 (H’04’). 由硬件写入从机地址的可编程部分。收到这个2-字节序列后，所有设计好能响应general call地址的器件将会接收它们地址中的可编程部分，但不会复位。
- 00000000 (H’00’). 不允许使用。

剩下所有的代码组合都还没有定义，并且所有器件都必须忽略它们。

当最低位B是1时，2-字节序列是一个硬件general call，意思是序列由一个硬件主机发送，比如键盘扫描器，它不能发送一个需要的从机地址。由于硬件主机不能事先知道数据需要发给哪个器件，所以它只能产生硬件general call和它自己的地址——把自己的信息发送给系统。

第二个字节中剩下的7位包含了该硬件主机的地址，这个地址可以由连接到总线上的智能设备(比如单片机)获取并且根据硬件主机的信息作出响应的动作。硬件主机还可以作为从机，从机地址跟主机地址一样。

在某些系统中，一种可能的情况是，硬件主机发送端在系统复位后被设为从机-接收端。

在这种情况下，系统设定好的主机可以告诉硬件主机-发送端(现在工作在从机-接收端模式)它需要发送的地址。在这个编程周期后，硬件主机仍然工作在主机-发送端模式。

20.2.2.5.2 起始字节

单片机可以用两种方法连接到I2C总线。带有I2C总线接口模块的单片机可以使用中断的方式处理总线的请求，但是当单片机没有接口模块，就必须用软件来实时查询监控总线。显然查询监控的次数越多，它能处理其它功能的时间就越少。所以带有硬件接口模块的单片机和依赖软件查询的单片机，有速度上的差异。

在这种情况下，数据传输可以由一个比通常时间要长的起始过程来进行。

这个起始过程由下面几个步骤组成：

- 一个起始位(S)
- 一个起始字节(00000001)
- 一个应答时钟脉冲(ACK)
- 一个重复起始位(Sr)

在主机发送一个起始位S请求占用总线后，再发送起始字节(00000001)。另一个单片机于是可以用较慢的查询速度来采样SDA传输线，直到检测到起始字节中任意一个低电平。在检测到这个SDA上的低电平后，单片机就可以切换到一个高速的采样频率来检测重复起始位Sr。

硬件接收端在收到重复起始位Sr后会复位，所以会忽略起始字节。

起始字节后会有一个应答位相关的时钟脉冲，这个脉冲只是为了让总线协议保持统一，起始字节不允许器件应答。

20.2.3 I2C总线规范的扩展

以100kbit/s速度传输数据和7位寻址的I2C总线协议已经存在三十多年没有变化了。I2C的总线概念已经成为世界范围内的标准，市面上有成千上万种兼容I2C总线的芯片。现在I2C总线规范可以扩展下面两种特性：

- 支持高达400kbit/s传输速度的快速模式
- 10位寻址模式，支持1024个地址空间

扩展I2C总线规范有两个原因：

- 新兴应用会需要传输更多的串行数据，从而需要比100kbit/s更快的速度。IC制造技术的进步可以在不增加成本的前提下支持4倍甚至更高的速度。
- 7位寻址所支持的112个地址已经被授权多次。为了避免地址重复的问题，地址需要更多的组合。使用新的10位寻址可以获得约10倍的可用地址空间。

所有新的I2C总线接口器件都支持快速模式，他们更希望以400kbit/s的速度接收或者发送数据。最低的需求是它们能同步一个400kbit/s的传输；它们也能延长SCL信号的低电平以降低传输速度。快速模式的器件必须向下兼容，也就是能够跟100kbit/s的器件进行通信。

显然0到100kbit/s的器件不能在快速I2C总线的系统里工作，因为它们无法跟上更高的传输速度，有可能发生无法预测的问题。

支持快速I2C总线接口的从机可以使用7位或者10位寻址，但是推荐使用7位寻址方式，因为7位寻址成本更低而且传输的数据相对更也少。7位寻址和10位寻址的器件可以混合使用在同一个I2C总线系统中，不管系统是工作在0到100kbit/s的标准模式还是0到400kbit/s的快速模式。当前存在的主机和将来的主机都可以产生7位或者10位地址。

20.2.4 1 快速模式

在快速模式中，之前I2C总线规范定义的协议，格式，逻辑电平和SDA/SCL传输线上的最大负载电容都保持不变。跟之前规范不同的是：

- 最大比特率增加到400kbit/s
- 串行数据SDA和串行时钟SCL信号的时序不同。不需要兼容其它总线系统比如CBUS，因为它们不能工作在这个速度。
- 工作在快速模式的器件必须在输入上抑制毛刺信号，并且输入端需要施密特触发器。
- 工作在快速模式的器件必须在输出端设计SDA和SCL信号的下降沿斜率控制。
- 如果工作在快速模式的器件掉电了，那么SDA和SCL的IO管脚必须处于悬空状态，避免干扰总线。
- 接到总线上的外部上拉器件必须适配快速模式的I2C总线所允许的信号上升时间。对于总线负载电容小于200pF的情况，上拉器件可以是一个电阻；对于总线负载电容在200pF到400pF之间的情况，上拉器件可以是一个电流源(最大3mA)或者一个开关电阻。

20.2.4.1 10位寻址

使用10位寻址不改变I2C总线规范的协议。10位地址开发利用了起始位(S)和重复起始位(Sr)后第一个字节的前7位中保留的1111XXX组合。

10位地址也不影响现有的7位寻址方式。7位寻址和10位寻址的器件可以接在同一个I2C总线上，并且7位寻址和10位寻址的器件都可以用在标准模式(100kbit/s)的系统中或者快速模式(400kbit/s)的系统中。

尽管保留地址1111XXX有8种可能的组合，但是只有4种组合11110XX是10位寻址可用的。剩下的11111XX组合保留给将来使用。

A – 头两个字节的位定义

10位地址由起始位(S)或者重复起始位(Sr)后的头两个字节组成。

第一个字节的前7位是11110XX，其中的最后两位XX是10位地址的最高两位(MSB)；第一个字节的第8位是读写位，用来定义传输方向，0表示主机写从机，1表示主机读从机。

如果读写位是0，那么第二个字节为剩下的8位地址(XXXXXXXX)。如果读写位是1，那么下个字节为从机发给主机的数据。

B – 10位寻址方式

10位寻址的传输中可能包含各种读写的组合。可能涉及到的数据传输格式有：

- 主机-发送端给从机-接收端发送一个10位的从机地址，数据传输方向不变化。在起始位后，各个从机将自己的地址跟第一个字节的前7位(11110XX)进行比较，并且判断第8位读写位是否为0。很有可能多个器件都能匹配上，并且发送一个应答位(A1)。所有匹配上的从机将继续比较第二个字节的8位从机地址(XXXXXXXX)，这时候应该只有1个从机匹配，并且发送应答位(A2)。匹配上的从机将一直保留这个被选中的状态，直到它收到停止位(P)或者后面跟着不同从机地址的重复起始位(Sr)。
- 主机-接收端使用10位地址向从机-发送端读取数据，在第二个读写位后传输方向发生了变化。直到应答位A2，读取的过程都跟上面发送的过程一样。在重复起始位(Sr)后，匹配的从机会记住自己是被选中过的。然后这个从机比较重复起始位Sr后第一个字节的前7位是否跟起始位后的7位相同，并且判断第8位是否为1，如果是的话，从机认为自己被寻址到，并且选中为发送端，于是该从机发送应答位A3。

从机-发送端会一直保留被选中的状态，直到它收到一个停止位(P)或者一个跟着不同从机地址的重复起始位(Sr)。在重复起始位(Sr)后，所有其它从机也会都开始比较第一个字节的前7位(11110XX)，并且判断读写位。但是，由于读写位为1(对10位地址的器件)，或者从机地址位11110XX(对7位地址的器件不匹配)，所以它们中没有任何一个会被寻址选中。

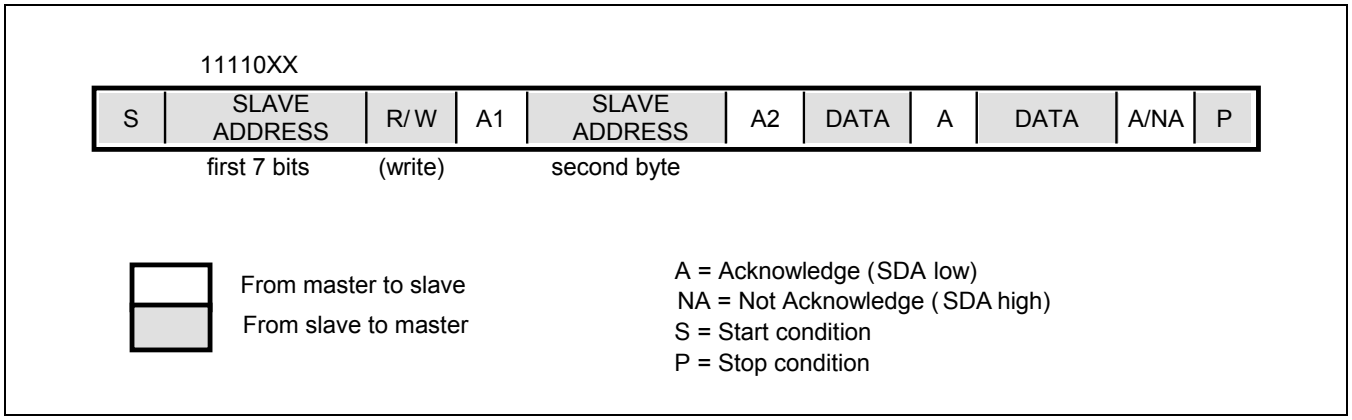


Figure 20-7 主机-发送端用10位地址寻址从机-接收端

20.2.4.2 General Call寻址和起始字节

I2C总线10位地址的寻址过程是起始位后的头两个地址决定哪个从机被主机选中。其中的例外就是“general call”地址00000000 (H’00’)。

10位寻址方式的从机跟7位寻址的从机一样，会响应”general call”寻址。

硬件主机可以在”general call”后发送它们的10位地址。这种情况下，”general call”地址后，紧接着两个连续的字节，这两个字节包含的是主机-发送端的10位地址。

10位寻址中起始字节00000001 (H’01’)的产生跟7位寻址一样。

20.3 I2C时序

Table 20-4 时序要求

Parameter	Symbol	标准模式的I2C总线		快速模式的I2C总线		Unit
		Min	Max	Min	Max	
SCL时钟周期	FSCL	0	100	0	400	kHz
停止位和起始位之间的总线空闲时间	TBUF	4.7	–	1.3	–	us
(重复)起始位Hold time 这个时间后，产生第一个时钟脉冲	THD;STA	4.0	–	0.6	–	us
SCL时钟的低电平时长	TLOW	4.7	–	1.3	–	us
SCL时钟的高电平时长	THIGH	4.0	–	0.6	–	us
重复起始位的Set-up time	TSU;STA	4.7	–	0.6	–	us
数据位 hold time	THD;DAT	0	–	0	0.9	us
数据位 set-up time	TSU;DAT	250	–	100	–	ns
SDL和SCL信号的上升时间	Tr	–	1000	20+01Cb	300	ns
SDL和SCL信号的下降时间	Tf	–	300	20+01Cb	300	ns
停止位的Set-up time	TSU;STO	4.0	–	0.6	–	us
每个信号线的负载电容	Cb	–	400	–	400	pF

20.4 寄存器说明

20.4.1 寄存器表 (Base Address: 0x400A_0000)

Offset Address	Name	Description	R/W	Reset State
0x000 ~ 0x04C	–	Reserved	–	–
0x050	I2C_ECR	时钟使能寄存器	W	–
0x054	I2C_DCR	时钟禁止寄存器	W	–
0x058	I2C_PMSR	电源管理状态寄存器	R	–
0x05C	–	Reserved	–	–
0x060	I2C_CR	控制寄存器	R/W	0x00000000
0x064	I2C_MR	模式寄存器	R/W	0x000001F4
0x068	–	Reserved	–	–
0x06C	–	Reserved	–	–
0x070	I2C_SR	状态寄存器	R	0x000000F8
0x074	I2C_IER	中断使能寄存器	W	–
0x078	I2C_IDR	中断禁止寄存器	W	–
0x07C	I2C_IMR	中断状态寄存器	R	0x00000000
0x080	I2C_DAT	数据寄存器	R/W	0x00000000
0x084	I2C_ADR	从机地址寄存器	R/W	0x00000000
0x088	I2C_THOLD	Hold/Setup 延时控制寄存器	R/W	0x00000001

20.4.1.1 I2C_ECR (时钟使能寄存器)

- Address = Base Address + 0x0050

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

Name	Bit	Type	Description	Reset Value
CLKEN	[1]	W	时钟使能控制位. 0 = 无效 1 = 使能I2C时钟	-
DBGEN	[1]	W	调试使能控制位 0 = 无效 1 = 使能I2C模块的调试功能	-

20.4.1.2 I2C_DCR (时钟禁止寄存器)

- Address = Base Address + 0x0054

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

Name	Bit	Type	Description	Reset Value
CLKEN	[1]	W	时钟禁止控制位. 0 = 无效 1 = 禁止I2C时钟	-
DBGEN	[1]	W	调试禁止控制位 0 = 无效 1 = 禁止I2C模块的调试功能	-

20.4.1.3 I2C_PMSR (电源管理状态寄存器)

- Address = Base Address + 0x0058

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

Name	Bit	Type	Description	Reset Value
CLKEN	[1]	R	CLKEN：时钟使能/禁止状态 0 = I2C时钟被禁止 1 = I2C时钟被使能	0
IPIDCODE	[29:4]	R	模块版本信息，共26位	-
DBGEN	[31]	R	DBGEN：调试模式 0 = dbgack_sclk输入对I2C功能无影响 1 = dbgack_sclk被使能。 当这位是低，I2C功能保持不变。当这位是高，I2C的功能被冻结无法使用，但是寄存器的读写功能不受影响，以方便调试。	0

20.4.1.4 I2C_CR (控制寄存器)

- Address = Base Address + 0x0060, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
SWRST	[0]	RW	SWRST : I2C软件复位 0 = 无效 1 = 产生一个软件复位(I2C_PMSR寄存器不会被复位)	0
AA	[1]	RW	I2C应答. 0 = 不发送应答位 (应答SCL时钟脉冲期间内SDA保持高) 1 = 当收到从机地址(或者当I2C_ADR的GC为为1时收到general call地址)，或者在接收模式收到了数据，发送应答位	0
STO	[2]	RW	I2C停止 0 = 不会在总线上发送停止位 1 = 产生一个停止位。当检测到总线上有停止位时，STO位会被自动清除。在从机模式，这个位用来从总线错误中恢复。在这种情况下，不发送停止位，但是I2C接口会认为已经收到停止位并且切换到“未被寻址到”的从机模式(STO位会被I2C接口硬件清除)	0
STA	[3]	RW	I2C启动 0 = 工作在从机模式 1 = 当设置为1，I2C接口工作在主机模式并且检测I2C总线的状态，如果总线处于空闲，那么产生一个起始位。如果总线不空闲，那么I2C接口将等到停止位后再产生一个起始位(在一个最小时间后)。当起始位成功产生后，该位会被自动清零。	0
SI	[4]	RW	SI : I2C中断 0 = 清除SI 1 = 有中断需要处理。当SI为1,时，i2c_int信号为高，并且SCL传输线会被拉低。传输被暂停，直到SI被清除。	0
ENA	[8]	RW	I2C使能 0 = 禁用I2C接口(当I2C接口被禁用，SCL和SDA也被禁	0

			用，没有任何输出和输入) 1 = 使能I2C接口	

20.4.1.5 I2C_MR (模式寄存器)

- Address = Base Address + 0x0064, Reset Value = 0x0000_01F4

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

Name	Bit	Type	Description	Reset Value
PRV	[11:0]	RW	预分频值 这个值用来设置总线的速度(FSCL). PRV (pre-scaler Value)的值用下面的公式来产生FSCL： FSCL = PCLK/(PRV+4)	0x1F4
FAST	[12]	RW	快速模式 0 = 禁用快速模式，使能标准模式。在这个模式下，高电平和低电平的比为1:1并且最大波特率为100kHz。 1 = 使能快速模式。在这个模式下，高电平和低电平的比为2:3并且最大波特率为400kHz。	0

Table 20-5 基于 PRV, FAST 和 PCLK 的 FSCL 值，单位 kHz

PRV (Decimal)	FAST	PCLK (MHz)	
		10	20
500	0	19.8	39.7
400	0	24.7	49.5
250	0	39.3	78.7
200	0	49	98
125	0	77.5	–
125	1	77.5	155
100	1	96	192
62.5	1	150	300
50	1	185	370
25	1	344	–

注意：

1. PCLK 的频率至少是 FSCL 频率的 6 倍(用于 SCL 同步+状态机)。
2. PRV 值复位后是’500’ (十进制)。
3. PRV 不可以写’000’ (十六进制)。

20.4.1.6 I2C_SR (状态寄存器)

- Address = Base Address + 0x0070, Reset Value = 0x0000_00F8

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

Name	Bit	Type	Description	Reset Value
SR	[7:3]	R	I2C状态代码 接口状态代码，共有27种可能的状态 I2C_SR复位值是0x000000F8. 当I2C_SR的值是这个复位值时，表明当前没有任何相关信息。所有其它状态值都跟某个工作状态有关。当I2C接口的状态机执行到某个状态时，这个寄存器的值也会更新到该状态代码，并且SI中断位会被置1。 所有这些状态代码的意义，软件执行的下一个动作以及I2C接口执行的下一个动作，都在下一页中描述。	0x1F

Table 20-6 主机-发送模式的状态码

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
0x0000_0008	起始位已经发送	x	0	x	0	将从机地址写入I2C_DAT并且写入读写位，将I2C_CR中的SI清零	将发送从机地址和读写位，并且等待ACK
0x0000_0010	重复起始位已经发送	x	0	x	0	将从机地址写入I2C_DAT并且写入读写位，将I2C_CR中的SI清零	将发送从机地址和读写位，并且等待ACK。如果读写位是读，那么将切换到接收模式。
0x0000_0018	从机地址和读写位已经被发送，并且收到ACK	0	0	x	0	将数据写入I2C_DAT，将I2C_CR中的SI清零	数据字节将被发送，并且等待ACK.
		1	0	x	0	将I2C_CR中的STA置1，并将I2C_CR中的SI清零	将发送重复起始位
		0	1	x	0	将I2C_CR中的STO置1，并将I2C_CR中的SI清零	将发送停止位
		1	1	x	0	将I2C_CR中的STA和STO都置1，并将I2C_CR中的SI清零	将发送停止位，然后再发送起始位
0x0000_0020	从机地址和读写位已经被发送，但没有收到ACK	同上				同上	同上
0x0000_0028	数据已经被发送，并且收到ACK	同上				同上	同上
0x0000_0030	数据已经被发送，但没有收到ACK	同上				同上	同上
0x0000_0038	在发送从机地址和读写位，或者发送数据时，丢失了仲裁	0	0	x	0	将I2C_CR中的SI清零	释放I2C总线，切换到从机模式
		1	0	x	0	将I2C_CR中的STA置1，并将I2C_CR中的SI清零	等待直到I2C总线空闲，然后发送一个起始位

Table 20-7 主机-接收模式状态码

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
0x0000_0008	起始位已经发送	x	0	x	0	将从机地址写入I2C_DAT并且写入读写位，将I2C_CR中的SI清零	将发送从机地址和读写位，并且等待ACK
0x0000_0010	重复起始位已经发送	x	0	x	0	将从机地址写入I2C_DAT并且写入读写位，将I2C_CR中的SI清零	将发送从机地址和读写位，并且等待ACK。如果读写位是读，那么将切换到接收模式。
0x0000_0038	在发送从机地址和读写位时，丢失了仲裁	0	0	x	0	将I2C_CR中的SI清零	释放I2C总线，切换到从机模式
		1	0	x	0	将I2C_CR中的STA置1，并将I2C_CR中的SI清零	等待直到I2C总线空闲，然后发送一个起始位
0x0000_0040	从机地址和读请求已经被发送，并且收到ACK	0	0	1	0	将I2C_CR中的SI清零 将I2C_CR中的AA置1	将收到数据，然后返回ACK
		0	0	0	0	将I2C_CR中的SI清零 将I2C_CR中的AA清零	将收到数据，然后不返回ACK
0x0000_0048	从机地址和读请求已经被发送，但没有收到ACK	1	0	x	0	将I2C_CR中的STA置1，并将I2C_CR中的SI清零	将发送重复起始位
		0	1	x	0	将I2C_CR中的STO置1，并将I2C_CR中的SI清零	将发送停止位
		1	1	x	0	将I2C_CR中的STA和STO都置1，并将I2C_CR中的SI清零	将发送停止位，然后再发送起始位
0x0000_0050	收到数据位，并返回了ACK	0	0	1	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1	将收到下一个数据，然后返回ACK
		0	0	0	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零	将收到下一个数据，然后不返回ACK
0x0000_0058	收到数据位，但没有返回ACK	1	0	x	0	读数据，将I2C_CR中的STA置1，并将I2C_CR中的SI清零	将发送重复起始位

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
		0	1	x	0	读数据，将I2C_CR中的STO置1，并将I2C_CR中的SI清零	将发送停止位
		1	1	x	0	读数据，将I2C_CR中的STA和STO都置1，并将I2C_CR中的SI清零	将发送停止位，然后再发送起始位

Table 20-8 从机-接收模式状态码

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
0x0000_0060	收到本从机地址+写操作，返回了ACK	x	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	将会收到数据，并返回ACK
		x	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零	将会收到数据，但不返回ACK
0x0000_0068	发送从机地址+读写位时丢失仲裁(在主机模式下)，切换到从机模式，收到本主机地址，返回了ACK	x	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	将会收到数据，并返回ACK
		x	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零	将会收到数据，但不返回ACK
0x0000_0070	收到General Call 地址，返回了ACK	x	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	将会收到数据，并返回ACK
		x	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零	将会收到数据，但不返回ACK
0x0000_0078	发送从机地址+读写位时丢失仲裁(在主机模式下)，切换到从机模式，收到General Call 地址，返回了ACK	x	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	将会收到数据，并返回ACK
		x	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零	将会收到数据，但不返回ACK
0x0000_0080	被本机地址寻址到，写操作，收到数据，并返回了 ACK	x	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	将会收到数据，并返回ACK
		x	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零	将会收到数据，但不返回ACK
0x0000_0088	被本机地址寻址到，写操作，收到数据，但没有返回ACK	0	0	0	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零	切换到“未被选中”的从机模式，禁止本机地址识别和general call地址识别。
		0	0	1	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1	切换到“未被选中”的从机模式，应答本机地址寻址和general call(如果I2C_ADR的GC=1)
		1	0	0	0	读数据，将I2C_CR	切换到“未被选中”的从

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
						中的SI清零，将I2C_CR中的AA清零，将I2C_CR中的STA置1	机模式，禁止本机地址识别和general call地址识别。一旦总线空闲，将马上发送起始位。
		1	0	1	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1，将I2C_CR中的STA置1	切换到“未被选中”的从机模式，应答本机地址寻址和general call (如果I2C_ADR的GC=1)。一旦总线空闲，将马上发送起始位。
0x0000_0090	被general call寻址选中，收到数据，并返回了ACK	x	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	将会收到数据，并返回ACK
		x	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零	将会收到数据，但不返回ACK
0x0000_0098	被general call寻址选中，收到数据，但没有返回ACK	0	0	0	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零	切换到“未被选中”的从机模式，禁止本机地址识别和general call地址识别。
		0	0	1	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1	切换到“未被选中”的从机模式，应答本机地址寻址和general call (如果I2C_ADR的GC=1)
		1	0	0	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零，将I2C_CR中的STA置1	切换到“未被选中”的从机模式，禁止本机地址识别和general call地址识别。一旦总线空闲，将马上发送起始位。
		1	0	1	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1，将I2C_CR中的STA置1	切换到“未被选中”的从机模式，应答本机地址寻址和general call (如果I2C_ADR的GC=1)。一旦总线空闲，将马上发送起始位。
0x0000_00A0	仍然被当作从机被寻址到时，收到停止位或者重复起始位	0	0	0	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零	同上

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
		0	0	1	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1	
		1	0	0	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零，将I2C_CR中的STA置1	
		1	0	1	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1，将I2C_CR中的STA置1	

Table 20-9 从机-发送模式状态码

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
0x0000_00A8	收到本从机地址+读操作，返回了ACK	x	0	1	0	写数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1	数据将被发送
		x	0	0	0	写数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零	最后一个字节的数据将被发送，并且之后从机再也不响应
0x0000_00B0	在作为主机发送从机地址+读写位时丢失了仲裁；收到了本从机地址，并返回了ACK	x	0	1	0	写数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1	数据将被发送
		x	0	0	0	写数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零	最后一个字节的数据将被发送，并且之后从机再也不响应
0x0000_00B8	数据已经被成功发送，并且返回了ACK	x	0	1	0	写数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1	数据将被发送
		x	0	0	0	写数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零	最后一个字节的数据将被发送，并且之后从机再也不响应
0x0000_00C0	数据已经被成功发送，但没有收到ACK	0	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零	切换到“未被选中”的从机模式，禁止本机地址识别和general call地址识别。
		0	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	切换到“未被选中”的从机模式，应答本机地址寻址和general call (如果I2C_ADR的GC=1)
		1	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零，将I2C_CR中的STA置1	切换到“未被选中”的从机模式，禁止本机地址识别和general call地址识别。一旦总线空闲，将马上发送起始位。
		1	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1，将I2C_CR中的STA置1	切换到“未被选中”的从机模式，应答本机地址寻址和general call (如果I2C_ADR的GC=1)。一旦总线空闲，将马上发送起始位。

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
0x0000_00C8	最后一个字节的数据已经被成功发送，并且收到了ACK	0	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零	同上
		0	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	
		1	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零，将I2C_CR中的STA置1	
		1	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1，将I2C_CR中的STA置1	

Table 20-10 其它状态码

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
0x0000_00F8	没有相关的状态信息；I2C_CR中的SI=0	–	–	–	–	无动作	等待或者进行当前的操作
0x0000_0000	由于非法的起始位或者停止位产生的总线错误	0	1	x	0	无动作	只有内部硬件会被影响。任何情况下，总线都会被释放，并且STO被清零。

20.4.1.7 I2C_IER (中断使能寄存器)

- Address = Base Address + 0x0074

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

Name	Bit	Type	Description	Reset Value
SI	[4]	W	SI : SI中断使能 0 = 无效 1 = 使能SI中断	-

20.4.1.8 I2C_IDR (中断禁止寄存器)

- Address = Base Address + 0x0078

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

Name	Bit	Type	Description	Reset Value
SI	[4]	W	SI : SI中断禁止 0 = 无效 1 = 禁止SI中断	-

20.4.1.9 I2C_IMR (中断状态寄存器)

- Address = Base Address + 0x007C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
SI	[4]	R	SI使能后的中断状态 当这位是1时，表示有中断需要处理，这时i2c_int为高，SCL传输线则被拉低，传输被暂停，直到SI被处理完并清零。	0

读取这个寄存器返回的是中断被使能后的状态，写寄存器无效。

20.4.1.10 I2C_SDR (数据寄存器)

- Address = Base Address + 0x0080, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
DAT	[7:0]	RW	I2C数据. 在接收模式下，该字节是从I2C总线上收到的数据；在发送模式下，该字节是需要发送到I2C总线上的数据。 DAT总是从右向左移，也就是说，在发送模式下，总是先发送最高位MSB，而在接收模式下，也是先接收到最高位MSB。 在一个发送过程中(该模块往I2C总线发送数据)，当数据被移位出去的时候，SDA传输线上的数据同时也被移位进来。所以在丢失仲裁的情况下，DAT将会包含正确的数据字节(从总线上读到的值)。	0x00

20.4.1.11 I2C_ADR (从机地址寄存器)

- Address = Base Address + 0x0084, Reset Value = 0x0000_0000

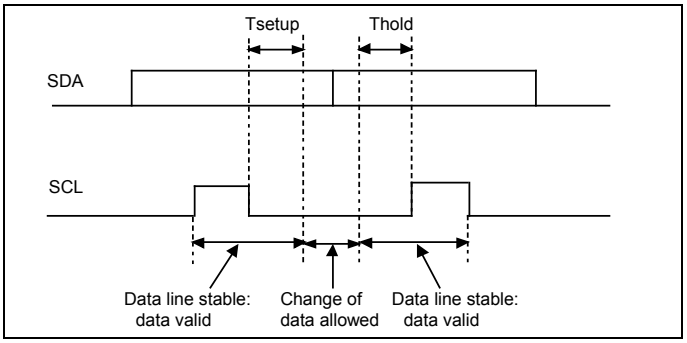
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																								ADR							GC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
GC	[0]	RW	General Call. 使能对general call地址的响应功能，当GC位置1时，如果识别到general call地址，I2C接口会产生中断。	0
ADR	[7:1]	RW	I2C地址 包含一个7位的I2C地址，如果I2C接口被设定为一个从机(发送端或者接收端)，那么将会响应这个从机地址。	0x00

20.4.1.12 I2C_THOLD (Hold/Setup延时控制寄存器)

- Address = Base Address + 0x0088, Reset Value = 0x0000_0001

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

Name	Bit	Type	Description	Reset Value
DL	[7:0]	RW	Hold/setup延时. Hold/Setup延时的值，由下面公式计算： THOLD = DL[7-0] × PCLK, TSETUP = DL[7-0] × PCLK  **Figure 1-8 Hold/Setup延时** 注意： 1. 复位后的DL值为'1' (十六进制). 2. Setup延时(TSETUP) 必须至少为250 ns (标准模式)，至少为100 ns (快速模式). 3. I2C器件必须在内部保证SDA信号上至少有300ns的hold时间。用户必须保证正确的hold值来满足慢速器件的时序。 4. DL的值不允许为0。	0x01

21 串行外设接口 (SPI)

21.1 概述

SPI，串行外设接口，又叫同步串行端口 (SSP)，用来连接串行外设。

21.1.1 主要功能

SSP是一个主机或者从机接口，可以用来跟其它外设进行同步串行通讯，只要外设有下面接口：

- 摩托罗拉(Motorola) SPI兼容接口

在主机和从机配置下，SSP都可以进行：

- 发送FIFO的并行数据转串行数据，内部FIFO有16位宽，8地址深
- 接收到的数据串行转并行，缓存到一个16位宽，8地址深的FIFO

产生的中断用来:

- 请求发送和接收FIFO
- 通知系统接收FIFO溢出了
- 通知系统在一段空闲时间后接收FIFO已经收到了数据

21.1.1.1 SSP的主要功能

SSP有如下功能特性:

- 主机或者从机选择
- 可编程的时钟比特率和分频
- 分开的发送和接收FIFO缓存，16位宽，8个地址深
- 4到16位可编程的数据帧大小
- 独立可控制的发送FIFO中断，接收FIFO中断和溢出中断
- 内部环回测试模式

21.1.1.2 可编程的参数

下列参数可编程：

- 主机或者从机模式
- 传送使能
- 帧格式
- 通信波特率
- 时钟相位和极性
- 数据宽度4到16位
- 中断控制

21.1.1.3 SPI主要特性

Motorola SPI兼容接口的主要特性:

- 全双工，4线同步传输
- 可编程的时钟极性和相位

21.1.2 管脚描述

Table 21-1 SSP 管脚描述

管脚名称	功能	I/O类型	说明
SSPCLK	SPI 串行时钟	I/O	–
SSPMOSI	主机输出从机输入	I/O	–
SSPMISO	主机输入从机输出	I/O	–
SSPFSS	帧，从机选择 (作为主机时) 帧输入 (作为从机时)	I/O	–

21.2 功能描述

21.2.1 模块框图

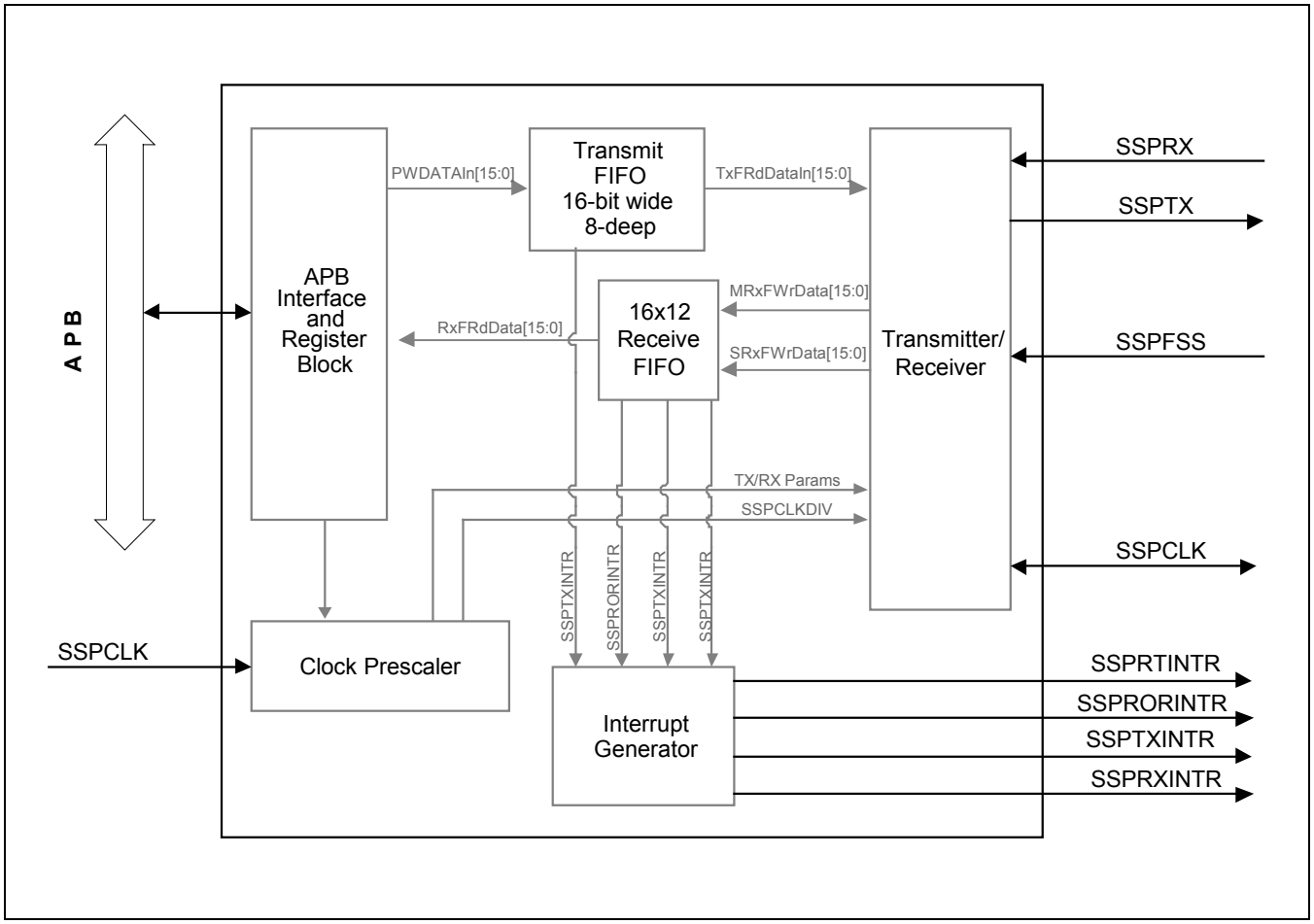


Figure 21-1 SSP 模块框图

21.2.2 功能描述

21.2.2.1 SSP概述

SSP是一个用来跟使用Motorola SPI协议的外围器件进行同步串行通讯的主机或者从机接口。

SSP可以将接收到的数据进行串-并转换。CPU通过AMBA APB接口读取SSP的数据，控制和状态信息。SSP接口的发送和接收使用了内部FIFO存储，支持8个16位数据的存储，发送模式和接收模式都支持。串行数据通过SSPTXD管脚发送，通过SSPRXD管脚接收。SSP模块包含一个可编程的比特率时钟分频器和预分频器，通过模块的输入时钟FSSPCLK产生串行输出时钟SSPCLK。比特率支持2MHz以及更高，跟配置的SSPCLK频率有关，最高频率跟外围器件和外围电路的时序有关。

SSP的工作模式，帧格式，帧大小，由控制寄存器SSPCR0和SSPCR1配置。

支持4种可配置的中断：

- SSPTXINTR中断：请求处理发送缓冲
- SSPRXINTR中断：请求处理接收缓冲
- SSPRORINTR：表示在接收FIFO中有溢出
- SSPRTINTR：表示数据在接收FIFO里的时间超时

依据选择的操作模式，SSPFSS可以输出为SPI的从机选择信号，低有效。

21.2.2.2 SSP功能描述

时钟分频

当配置成主机时,有2个串联在一起的分频计数器，用来给串行输出时钟SSPCLK提供时钟源。

用户可以通过SSPCPSR寄存器来配置预分频器，分两步给FSSPCLK进行2到254分频。SSPCPSR寄存器的最低位被被设计成无法使用，也就是无法使用奇数分频，保证了产生时钟的对称性(1/0占空比相等)。

预分频的输出接到了另一个1到256的分频器，通过SSPCR0可配置，这个分频器的输出最终输出到SSPCLK管脚。

发送FIFO

发送FIFO是一个16位宽，8地址深的先进先出缓冲区。CPU通过AMBA APB接口将数据写入该缓存，直到发送逻辑将数据读出。在通过SSPTXD管脚串行发送数据前，需要先将并行的数据写入发送FIFO。

接收FIFO

接收FIFO是一个16位宽，8地址深的先进先出缓冲区。从串行接口接收到的数据存在缓冲区里，直到被CPU通过AMBA APB总线读出。在通过SSPRXD管脚收到串行数据后，才能读取FIFO的并行数据。

发送和接收逻辑

当配置成主机时，连接到从机的时钟是由FSSPCLK分频得到的，在前面已经提到。主机发送逻辑连续的从发送FIFO中读取数据，并且将该并行数据转换成串行数据。然后串行数据和帧控制信号，通过SSPTXD管脚与SSPCLK管脚同步输出到从机。主机接收逻辑则在SSPRXD接收到的数据上进行串并转换，将数据提取并存储在接收FIFO中，供APB接口读取。

当配置成从机时，SSPCLK管脚的时钟由连接的主机提供，并且用这个时钟来进行发送和接收。从机的发送逻辑，在主机时钟的控制下，连续的从发送FIFO中读取数据，并且将该并行数据转换成串行数据，通过SSPTXD管脚将串行数据和帧控制信号输出到主机。从机接收逻辑在SSPRXD接收到的数据上进行串并转换，将数据提取并存储在接收FIFO中，供APB接口读取。

中断产生逻辑

SSP可产生并配置4种中断。4种中断通过或逻辑后，发送给CPU请求中断处理，在处理程序中，CPU需要读取SSP状态寄存器的值来判断发生的是哪种中断。

21.2.2.3 SSP操作方法

配置SSP

复位后，SSP逻辑是没有使能的，必须经过配置后才能使用。控制寄存器SSPCR0和SSPCR1需要将SSP配置为主机或者从机。从SSPCLK分频得到的比特率，必须通过时钟分频寄存器SSPCPSR寄存器配置。

使能SSP传送

要启动SSP的发送接收，可以往发送FIFO里写8个16位宽的数据，或者允许发送FIFO给CPU产生一个中断。一旦SSP被使能，数据的发送或者接收就会立即在SSPTXD和SSPRXD管脚上启动。

时钟比例

PCLK频率和 F_{SSPCLK} 频率的比例有一定的限制。F_{SSPCLK} 的频率必须小于或者等于PCLK的频率，保证SSPCLK时钟域上的控制信号在一个数据帧内能被PCLK及时同步：

- $F_{SSPCLK} \leftarrow F_{PCLK}$

在从机模式的工作中，为了对外部输入的SSPCLK进行同步和检测，以及同步，检测和处理SSPTXD信号，F_{PCLK} 必须是SSPCLK管脚信号频率的12倍。

在从机模式F_{PCLK}的最小至少是SSPCLK频率的12倍，而在主机模式，F_{PCLK}至少是SSPCLK的2倍。

在主机模式，要产生最大1MHz的比特率，F_{PCLK}需要至少为2MHz。如果PCLK频率是2MHz，那么SSPCPSR寄存器的值必须是2，SSPCR0寄存器中SCR[7:0]的值则必须为0。

在从机模式，如果需要工作在1Mbps的比特率，那么PCLK的频率必须至少为12MHz，SSPCPSR寄存器的值可以配置为12，SSPCR0寄存器中SCR[7:0]的值配置为0。

另外，PCLK的最大频率由SSPCPSR和SSPCR0的SCR[7:0]的最大值决定，为SSPCLK管脚信号频率的254 x 256倍。

PCLK的最小频率可以由下面的公式得到：

- F_{PCLK} (min) $\rightarrow 2 \times F_{SSPCLKOUT}$ (max) [主机模式]
- F_{PCLK} (min) $\rightarrow 12 \times F_{SSPCLKIN}$ (max) [从机模式]

PCLK的最大频率可以由下面的公式得到：

- F_{PCLK} (max) $\leftarrow 254 \times 256 \times F_{SSPCLKOUT}$ (min) [主机模式]
- F_{PCLK} (max) $\leftarrow 254 \times 256 \times F_{SSPCLKIN}$ (min) [从机模式]

对SSPCR0控制寄存器编程

SSPCR0寄存器可以用来：

- 配置串行时钟速率
- 选择三种协议中的一种
- 选择数据字的大小

串行时钟速率值(SCR)，跟SSPCPSR预分频值(CPSDVSR)一起，用来决定发送和接收的比特率。帧格式使用FRF位控制，数据大小则由DSS位控制。SPH和SPO位用来配置Motorola SPI格式中的相位和极性。

对SSPCR1控制寄存器编程

SSPCR1寄存器用来：

- 选择主机或者从机模式
- 使能SSP

SSPCR1寄存器的主机或者从机选择位(MS)如果写0，那么是主机模式，也是复位后默认的模式；如果写1，那么是从机模式。如果配置成从机，SSPCR1寄存器的SOD位可以用来使能或者禁止SSPTXD信号的输出，这个功能可以用在多从机环境中，在主机需要并行广播的时候。

将SSE位写1，可以启动SSP进行工作。

比特率的产生

串行比特率由输入时钟PCLK的分频得到。时钟PCLK先由一个偶数预分频器进行分频(CPSDVSR)，支持2到254中偶数的分频，然后再由一个1到256的分频器(SCR)分频，分频系数为SCR+1，SCR在SSPCR0寄存器中。

输出的比特率时钟SSPCLK频率由下面公式定义：

- $F_{SSPCLK} = F_{PCLK} / (CPSDVR \times (1 + SCR))$

例如，如果PCLK是 10MHz，并且CPSDVSR = 2，那么SSPCLK的频率范围为 19.5kHz 到 5MHz。

帧格式

每个数据帧为4到16位长，由编程配置的数据长度决定，从MSB高位开始发送。

对于所有三种格式，当SSP在空闲时，串行时钟(SSPCLK管脚)都会被置于非活动状态。SSPCLK的空闲状态被用来提供一个接收超时功能，表示在某个时长后，FIFO内接收到的数据仍未被读取。

对Motorola SPI, 串行帧管脚为低有效，并且在整个传送过程中都被拉低。

Motorola SPI 帧格式

Motorola SPI 接口是一个4线的接口，SSPFSS信号用作从机选择。Motorola SPI格式的主要特征是SSPCLK管脚信号的非活动状态和相位可以用SSPCR0寄存器中的SPO和SPH位选择。

- SPO，时钟极性

如果SPO时钟极性控制位是0，那么在数据没有被传送时，SSPCLK管脚的稳定状态为低电平；如果SPO时钟极性控制为是1，那么SSPCLK管脚的稳定状态为高电平。

- SPH，时钟相位

SPH控制位可以选择捕获数据的时钟沿，并且允许它改变状态。该位对传输数据的第一位影响最大，可以设置在第一个数据捕获沿前允许时钟变化或者不允许时钟变化。当SPH相位控制位是0，数据在第一个时钟沿捕获，而当SPH相位控制位是1，数据在第二个时钟沿捕获。

Motorola SPI 格式, SPO = 0, SPH = 0

Motorola SPI 格式中 SPO = 0, SPH = 0 的示意图如下：

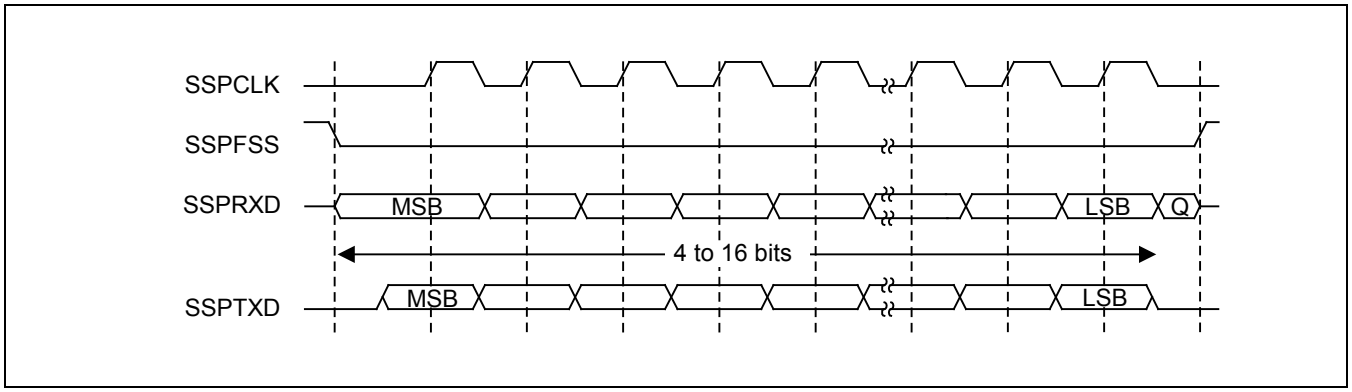


Figure 21-2 Motorola SPI 帧格式 (单次传输) SPO = 0, SPH = 0

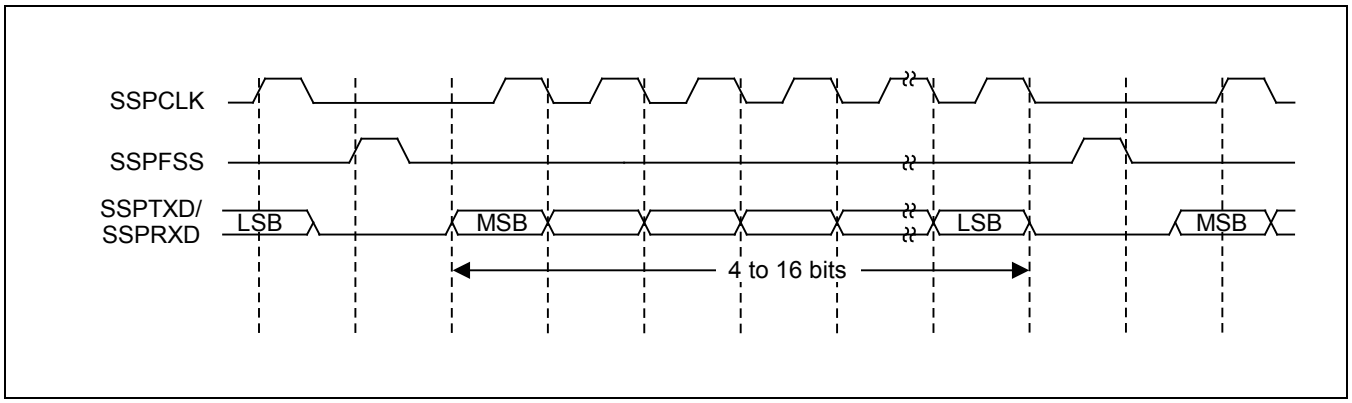


Figure 21-3 Motorola SPI 帧格式 (连续传输) SPO = 0, SPH = 0

这个配置中，在空闲时间：

- SSPCLK管脚信号被拉低
- SSPFSS被拉高
- 传输数据线SSPTXD被强制拉低
- 当SSP为主机时，SSPCLK管脚使能
- 当SSP为从机时，SSPCLK管脚禁止

当SSP被使能并且发送FIFO中数据有效时，传输的开始由SSPFSS主机信号拉低表示，此时，从机的数据也会发送到主机的SSPRXD信号线上。

半个SSPCLK周期后，有效的主机数据发送到SSPTXD管脚。于是主机和从机双方的数据都被发送出去，再过半个SSPCLK周期， SSPCLK主机时钟管脚变高。这时数据在SSPCLK管脚信号的上升沿被捕捉，并且会一直保持到时钟的下降沿。在单字传输的情况下，当所有数据位都传输完后，SSPFSS信号线在最后一位被捕捉后的一个SSPCLK周期后回到它的空闲状态 - 高电平。但是在连续的传输中，SSPFSS信号必须在数据传输中间一直保持高电平。这是因为从机选择管脚会冻结从机串行寄存器，并且如果SPH位是0就不允许改变。所以主机必须将SSPFSS管脚拉高，让从机在数据字的传输中间使能串行数据的写操作。在连续传输结束时，SSPFSS管脚在最后一位被捕捉后的一个SSPCLK周期后又会回到它的空闲状态。

Motorola SPI 格式，SPO = 0, SPH = 1

Motorola SPI格式中SPO = 0, SPH = 1 的信号传输示意图如下：

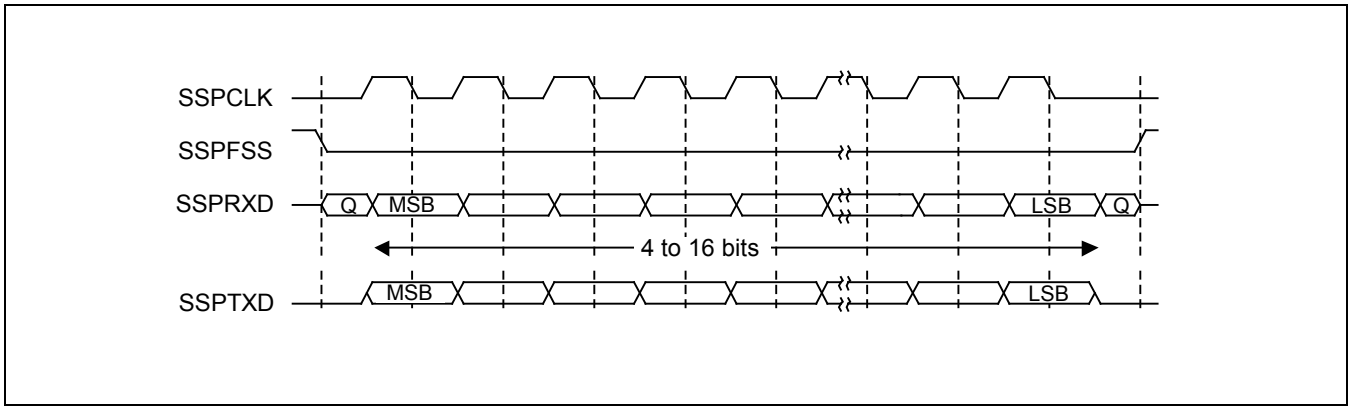


Figure 21-4 Motorola SPI 帧格式， SPO = 0 和 SPH = 1

这个配置中，在空闲时间：

- SSPCLK信号被拉低
- SSPFSS被拉高
- 传输数据线SSPTXD被强制拉低
- 当SSP为主机时，SSPCLK管脚使能
- 当SSP为从机时，SSPCLK管脚禁止

当SSP被使能并且发送FIFO中数据有效时，传输的开始由SSPFSS主机信号拉低表示。主机的SSPTXD输出管脚被使能。半个SSPCLK周期后，主机和从机的有效数据都会出现在相应的传输线上，同时SSPCLK也被使能，出现上升沿。然后数据会在SSPCLK的下降沿被捕捉，并且一直保持到SSPCLK的上升沿。

在单次传输的情况下，当所有数据位都传输完后，SSPFSS信号线在最后一位被捕捉后的一个SSPCLK周期后回到它的空闲状态 - 高电平。

在连续的传输中，SSPFSS信号在连续的数据传输中间一直保持低电平，最终结束的时候跟单次传输的情况一样。

Motorola SPI格式，SPO = 1, SPH = 0

Motorola SPI格式中的 SPO = 1, SPH = 0 信号传输示意图如下：

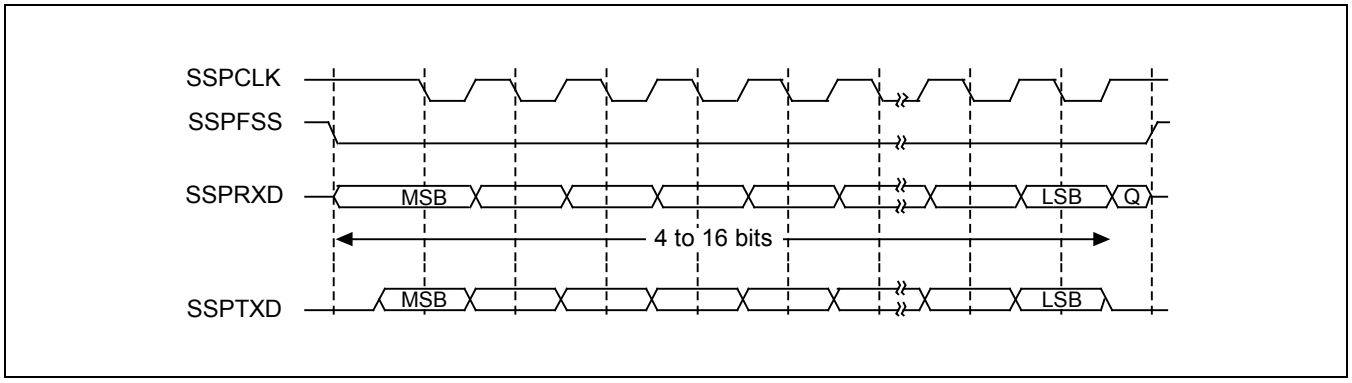


Figure 21-5 Motorola SPI 帧格式 (单次传输)，SPO = 1 和 SPH = 0

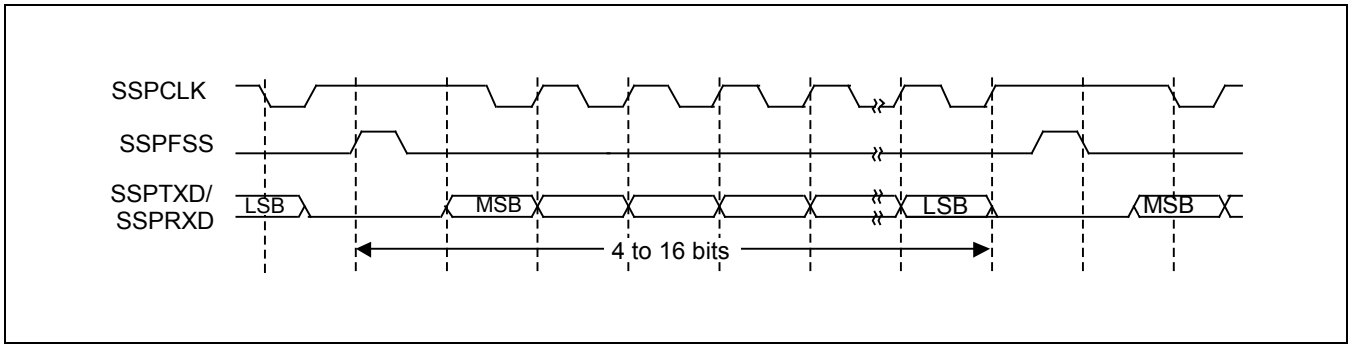


Figure 21-6 Motorola SPI 帧格式 (连续传输)， SPO = 1 和 SPH = 0

这种情况下，在空闲时间：

- SSPCLK信号被拉高
- SSPFSS被拉高
- 传输数据线SSPTXD被强制拉低
- 当SSP为主机时，SSPCLK管脚使能
- 当SSP为从机时，SSPCLK管脚禁止

当SSP被使能并且发送FIFO中数据有效时，传输的开始由SSPFSS主机信号拉低表示，此时，从机的数据也会发送到主机的SSPRXD信号线上。主机的SSPTXD输出管脚被使能。

半个周期后，主机的有效数据被发送到SSPTXD上。此时主机和从机双方都已发送数据，再经过半个SSPCLK周期后，SSPCLK主机时钟管脚变低。这意味着数据在时钟的下降沿被捕捉，并且一直保持到SSPCLK的下个上升沿。

在单次传输的情况下，当所有数据位都传输完后，SSPFSS信号线在最后一位被捕捉后的一个SSPCLK周期后回到它的空闲状态 - 高电平。但是在连续的传输中，SSPFSS信号必须在数据传输中间一直保持高电平。这是因为从机选择管脚会冻结从机串行寄存器，并且如果SPH位是0就不允许改变。所以主机必须将SSPFSS管脚拉高，让从机在数据字的传输中间使能串行数据的写操作。在连续传输结束时，SSPFSS管脚在最后一位被捕捉后的一个SSPCLK周期后又会回到它的空闲状态。

Motorola SPI 格式， SPO = 1, SPH = 1

Motorola SPI 格式中SPO = 1, SPH = 1的传输示意图如下：

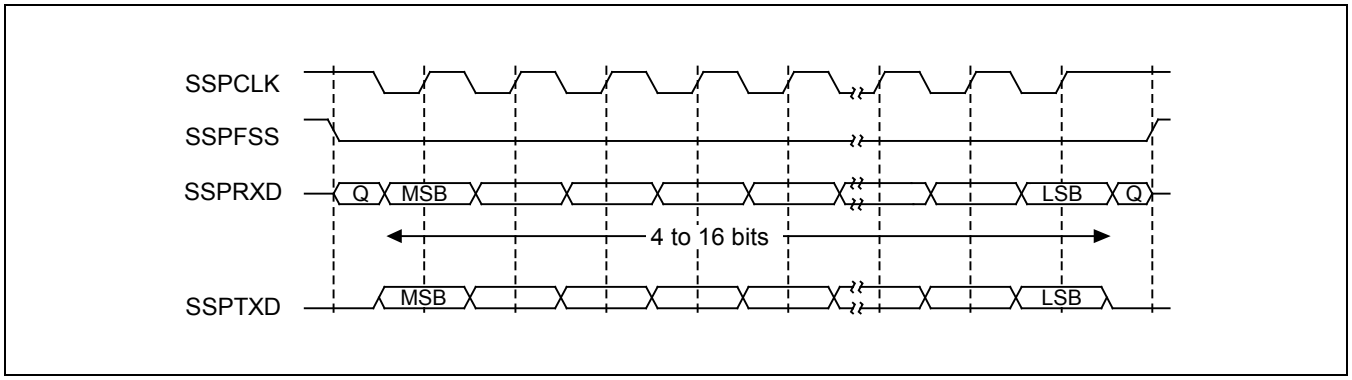


Figure 21-7 Motorola SPI 帧格式， SPO = 1 和 SPH = 1

这种情况下，在空闲时间：

- SSPCLK信号被拉高
- SSPFSS被拉高
- 传输数据线SSPTXD被强制拉低
- 当SSP为主机时，SSPCLK管脚使能
- 当SSP为从机时，SSPCLK管脚禁止

当SSP被使能并且发送FIFO中数据有效时，传输的开始由SSPFSS主机信号拉低表示。主机的SSPTXD输出管脚被使能。半个SSPCLK周期后，主机和从机的有效数据都会出现在相应的传输线上，同时SSPCLK也被使能，出现下降沿。然后数据会在SSPCLK的上升沿被捕捉，并且一直保持到SSPCLK的下降沿。

在单次传输的情况下，当所有数据位都传输完后，SSPFSS信号线在最后一位被捕捉后的一个SSPCLK周期后回到它的空闲状态 - 高电平。

在连续的传输中，SSPFSS信号在连续的数据传输中间一直保持低电平，最终结束的时候跟单次传输的情况一样。

主从配置示例

下图演示了SSP如何连接到其它同步串行接口的例子。

注意：SSP 不支持在同一个系统中动态切换主机和从机配置，只能是配置成单一主机或者单一从机。

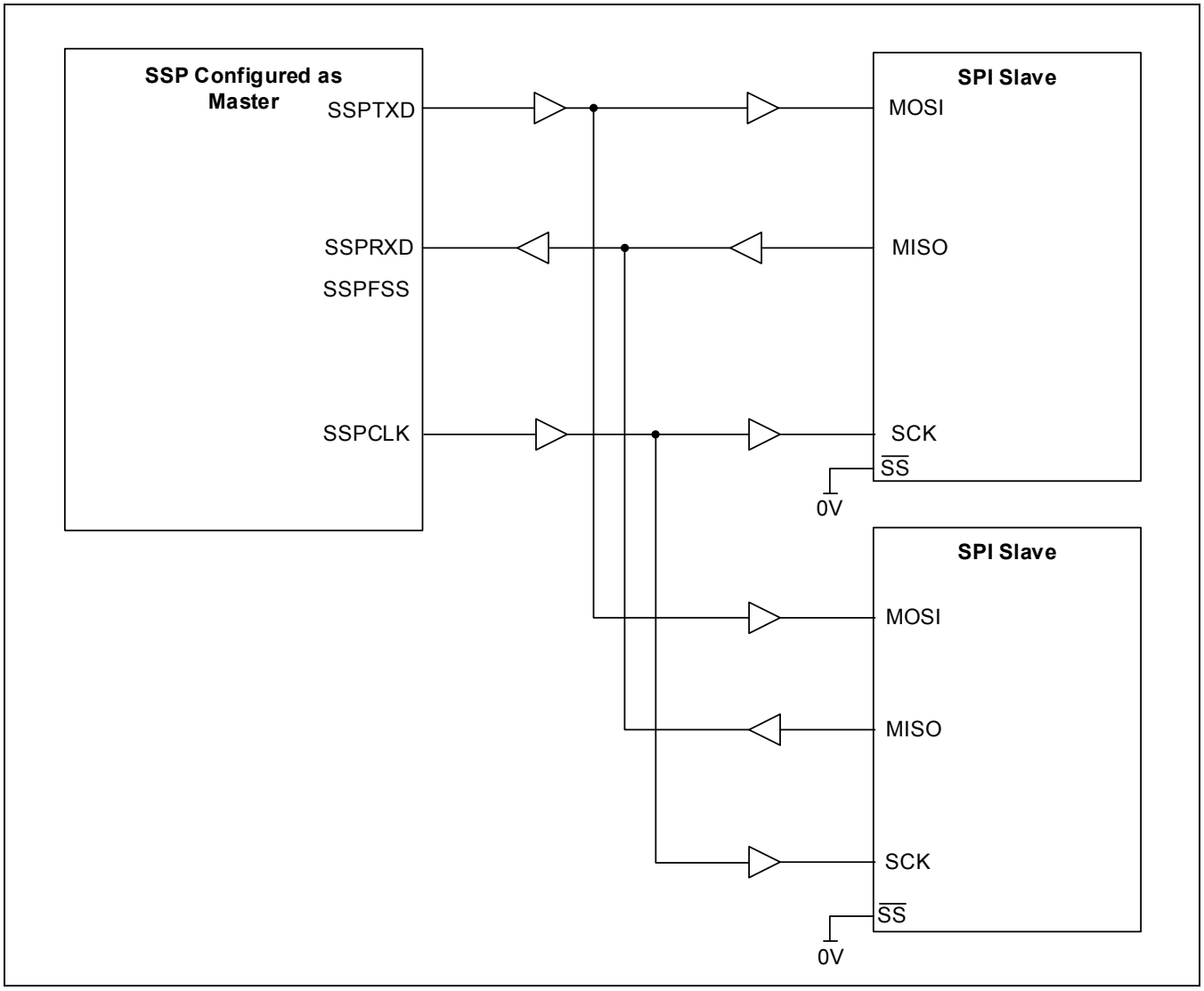


Figure 21-8 SSP主机连接两个SPI从机

上图为SSP配置成主机，连接了两个Motorola SPI从机。跟上述操作类似，主机可以通过SSPTXD给两个从机广播，只有一个从机可以驱动SPI MISO端口，给主机的SSPRXD发送数据。

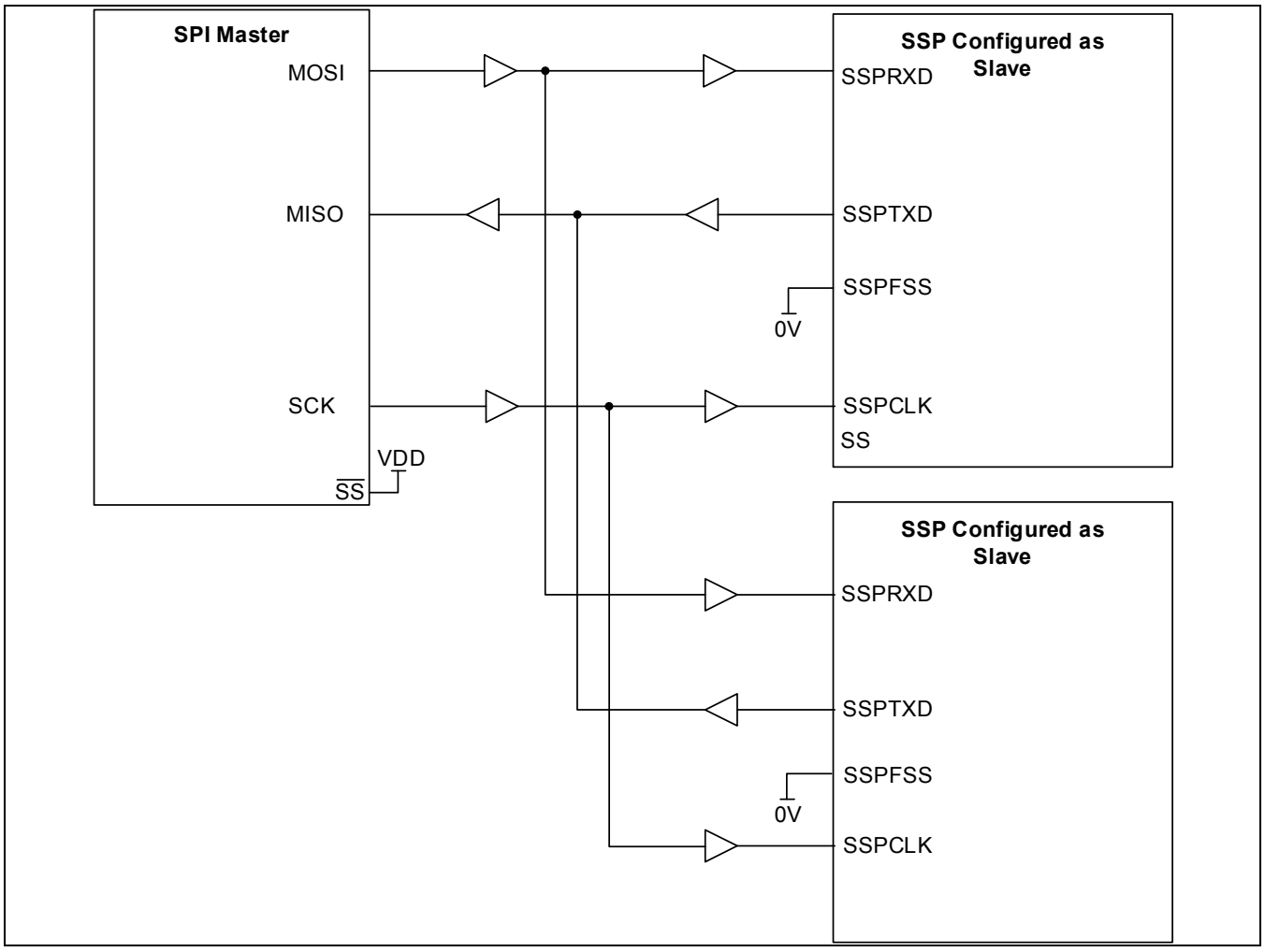


Figure 21-9 SPI 主机连接到两个SSP从机

上图为Motorola SPI作为主机，连接到两个SSP从机。在这种情况下，从机选择信号(SS)短接到高电平上并且配置成主机。主机通过SPI MOSI给两个从机广播，只有一个从机可以通过SSPTXD给主机的MISO发送数据。

21.2.2.4 中断

SSP可以产生5个中断：

- SSPRXINTR: SSP接收FIFO中断服务请求
- SSPTXINTR: SSP发送FIFO中断服务请求
- SSPRORINTR: SSP接收溢出中断请求
- SSPRTINTR: SSP超时中断请求

用户可以通过SSP_IMSCR寄存器中的相应位使能或者禁止这些中断。

- SSPRXINTR
  - 当接收FIFO的占用到一定数量后，会触发该中断。这个数量由SSP_CR1中的RXIFLSEL位设置。
- SSPTXINTR
  - 当发送FIFO的占用为4或者更少时，会触发该中断。这个发送中断SSPTXINTR不需要SSP使能，所以数据的发送可以用两种方法操作。一是数据可以在使能SSP和中断前就写入发送FIFO中，二是中断使能后在发送FIFO中断服务子程序中写入数据。
- SSPRORINTR
  - 当接收FIFO满后还收到了数据帧，会触发该中断。这个中断发生说明FIFO溢出了，此时新接收到的数据会覆盖接收移位寄存器，而不会写入FIFO中。
- SSPRTINTR
  - 当接收FIFO中有数据未被读取，并且SSP在32个位周期时长内一直处于空闲状态，会触发该中断。这个机制可以让用户知道接收FIFO中有数据需要处理。在接收FIFO被读取变空后，或者在SSPRXD上接收到新数据后，该中断会被清除。SSPICR寄存器中的RTIC位也可以清除该中断。

21.3 寄存器描述

21.3.1 寄存器表 (Base Address: 0x4009_0000)

Register	Offset	Description	Reset Value
SSP_CR0	0x0000	SSP控制寄存器0	0x0000_0000
SSP_CR1	0x0004	SSP控制寄存器1	0x0000_0010
SSP_DR	0x0008	SSP接收FIFO数据寄存器 (读该寄存器时) SSP发送FIFO数据寄存器 (写该寄存器时)	0x0000_0000
SSP_SR	0x000C	SSP状态寄存器	0x0000_0003
SSP_CPSR	0x0010	SSP时钟分频寄存器	0x0000_0000
SSP_IMSCR	0x0014	SSP中断使能/禁止寄存器	0x0000_0000
SSP_RISR	0x0018	SSP中断原始状态寄存器	0x0000_0008
SSP_MISR	0x001C	SSP中断状态寄存器	0x0000_0000
SSP_ICR	0x0020	SSP中断清除寄存器	0x0000_0000

21.3.1.1 SSP_CR0 (SSP控制寄存器0)

- Address = Base Address + 0x0000, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																SCR								SPH	SPO	FRF		DSS			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
DSS	[3:0]	RW	数据大小选择位 0000–0010 = 保留 0011 = 4位数据 0100 = 5位数据 0101 = 6位数据 0110 = 7位数据 0111 = 8位数据 1000 = 9位数据 1001 = 10位数据 1010 = 11位数据 1011 = 12位数据 1100 = 13位数据 1101 = 14位数据 1110 = 15位数据 1111 = 16位数据	0000'b
FRF	[5:4]	RW	帧格式选择位 Motorola SPI格式必须设置为00	00'b
SPO	[6]	RW	SSPCLK极性选择 0 = 没有数据传送时，SSPCLK管脚的稳定状态为低电平 1 = 没有数据传送时，SSPCLK管脚的稳定状态为高电平	0
SPH	[7]	RW	SSPCLKOUT相位 0 = 数据在第一个时钟沿捕捉 1 = 数据在第二个时钟沿捕捉	0
SCR	[15:8]	RW	串行时钟分频位 SCR用来产生发送和接收的比特率 比特率 = FPCLK/ (CPSDVR × (1 + SCR)) CPSDVSR为2到254之间的偶数，在SSPCPSR寄存器设置，SCR为0到255之间任意值。	0x00

21.3.1.2 SSP_CR1 (SSP控制寄存器1)

- Address = Base Address + 0x0004, Reset Value = 0x0000_0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																									RXIFLSEL			SOD	MS	SSE	LBM
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
LBM	[0]	RW	回送模式位 0 = 正常串行输出操作 1 = 发送移位寄存器的输出内部短接到接收串行移位寄存器	0
SSE	[1]	RW	SSP使能位 0 = SSP禁止 1 = SSP使能	0
MS	[2]	RW	主机或者从机模式 0 = 配置为主机 1 = 配置为从机	0
SOD	[3]	RW	从机模式输出禁止位 该位只有在从机模式(MS=1)下有效。在多从机系统里，SSP主机可以向系统里的所有从机广播，但是必须保证只有一个从机能够输出数据。在这样的系统里，多从机的RXD必须短接在一起，所以当SSP从机不应该输出数据驱动SSPTXD的时候，SOD位必须置1。 0 = SSP在从机模式可以驱动SSPTXD输出 1 = SSP在从机模式不驱动SSPTXD输出	0
RXIFLSEL	[6:4]	RW	接收FIFO中断触发点选择位 001 接收FIFO占用 > = 1/8 010 接收FIFO占用 > = 1/4 100 接收FIFO占用 > = 1/2 Others = 保留	0001'b

21.3.1.3 SSP_DR (SSP数据寄存器)

- Address = Base Address + 0x0008, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
DATA	[15:0]	RW	发送/接收FIFO 读：接收FIFO 写：发送FIFO 当数据位小于16的时候，必须右对齐，不用的高位无效。 接收逻辑为自动右对齐。	0x0000

读SSPDR时，接收FIFO的数据(当前FIFO读指针所指的值)被读取。当SSP接收逻辑把接收到的数据移除时，该数据会被存到接收FIFO中当前读指针所指的空间。

写SSPDR时，数据被写入发送FIFO中当前指针所指的空间。发送逻辑每发送一次就将发送FIFO中的数据移除一个，移除的数据载入到发送串行移位寄存器，以配置好的比特率串行发送到SSPTXD管脚。

当数据位小于16时，用户在写入发送FIFO的时候必须右对齐，发送逻辑会忽略没用到的高位。 接收到的数据如果小于16位，在接收缓冲区内也是右对齐的。

21.3.1.4 SSP_SR (SSP状态寄存器)

- Address = Base Address + 0x000C, Reset Value = 0x0000_0003

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

Name	Bit	Type	Description	Reset Value
TFE	[0]	R	发送FIFO是否为空状态位 0 = 发送FIFO非空 1 = 发送FIFO空	1
TNF	[1]	R	发送FIFO是否已满状态位 0 = 发送FIFO已满 1 = 发送FIFO未满	1
RNE	[2]	R	接收FIFO是否为空状态位 0 = 接收FIFO为空 1 = 接收FIFO非空	0
RFF	[3]	R	接收FIFO是否已满状态位 0 = 接收FIFO未满 1 = 接收FIFO已满	0
BSY	[4]	R	SSP工作状态标志位 0 = SSP空闲 1 = SSP正在发送并且/或者正在接收，或者发送FIFO非空	0

21.3.1.5 SSP_CPSR (SSP时钟分频寄存器)

- Address = Base Address + 0x0010, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
CPSDVSR	[7:0]	RW	时钟分频位 必须是2到254之间的偶数，根据PCLK的频率来决定。在读取的时候最低位总是返回0。	0x00

SSPCPSR位时钟预分频寄存器，用来设置F_{PCLK} 的分频系数，供下一分频器使用。寄存器的值必须是2到254之间的偶数。寄存器的最低位被硬件强制为0，即使将一个奇数写入该寄存器，读出来值的最低位也为0。

21.3.1.6 SSP_IMSCR (SSP中断使能/禁止寄存器)

- Address = Base Address + 0x0014, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
RORIM	[0]	RW	接收溢出中断 0 = 禁止RxFIFO溢出中断 1 = 使能RxFIFO溢出中断	0
RTIM	[1]	RW	接收超时中断 0 = 禁止RxFIFO超时中断 1 = 使能RxFIFO超时中断	0
RXIM	[2]	RW	接收FIFO中断 0 = 禁止接收FIFO中断 (1/2, 1/4, or 1/8可选) 1 = 使能接收FIFO中断 (1/2, 1/4, or 1/8可选)	0
TXIM	[3]	RW	发送FIFO中断 0 = 禁止发送FIFO中断 1 = 使能发送FIFO中断	0

读此寄存器返回相关中断的使能状态。写1使能相应中断，写0则禁止相应中断。

21.3.1.7 SSP_RISR (SSP中断原始状态寄存器)

- Address = Base Address + 0x0018, Reset Value = 0x0000_0008

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

Name	Bit	Type	Description	Reset Value
RORRIS	[0]	R	接收中断原始状态 SSPRORINTR中断的原始状态，不管该中断使能与否	0
RTRIS	[1]	R	接收超时中断原始状态 SSPRTINTR中断的原始状态，不管该中断使能与否	0
RXRIS	[2]	R	接收FIFO中断原始状态 SSPRXINTR中断的原始状态，不管该中断使能与否	0
TXRIS	[3]	R	发送FIFO中断原始状态 SSPTXINTR中断的原始状态，不管该中断使能与否	1

读该寄存器返回相应中断的原始状态，不管该中断是否被使能。写寄存器无效。

21.3.1.8 SSP_MISR (SSP中断状态寄存器)

- Address = Base Address + 0x001C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
ROMIS	[0]	R	接收中断原始状态 SSPRORINTR中断的原始状态，该中断使能后才能读到，否则一直为0	0
RTMIS	[1]	R	接收超时中断原始状态 SSPRTINTR中断的原始状态，该中断使能后才能读到，否则一直为0	0
RXMIS	[2]	R	接收FIFO中断原始状态 SSPRXINTR中断的原始状态，该中断使能后才能读到，否则一直为0	0
TXMIS	[3]	R	发送FIFO中断原始状态 SSPTXINTR中断的原始状态，该中断使能后才能读到，否则一直为0	0

读该寄存器返回相应中断的状态，该中断使能后才能读到，否则一直为0。写寄存器无效。

21.3.1.9 SSP_ICR (SSP中断清除寄存器)

- Address = Base Address + 0x0020, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
RORIC	[0]	W	接收溢出中断清除 0 = 无效 1 = 清除SSPRORINTR中断	0
RTIC	[1]	W	接收超时中断清除 0 = 无效 1 = 清除SSPRTINTR中断	0

写1清除该中断，写0无效。

22 电容式触摸按键传感器

22.1 概述

此MCU内嵌了一个最大支持20个扫描通道的电容式触摸按键检测模块。该模块支持基于电荷转移的检测技术，以满足不同应用条件下电容触摸检测。

22.1.1 特性

- 最大支持20通道按键检测
- 内嵌独立的硬件算法模块，支持硬件自动按键检测。
  - 灵活的硬件算法配置模块
  - 很强的抗干扰能力
- 自适应的扫描速度调节，优化功耗。
- 多种扫描触发模式。
  - 软件触发
  - 硬件自动间隔触发
  - iWDT中断触发
  - LED 扫描中断触发
- 多种按键中断模式和异常处理中断。

22.1.2 管脚描述

Table 22-1 TOUCH KEY 管脚描述

Pin Name	Function	I/O Type	Active Level	Comments
TCHx	触摸检测通道	A	-	-
ECP0	电荷转移模式下的外部电容接口	A	-	-

22.2 功能描述

22.2.1 模块框图

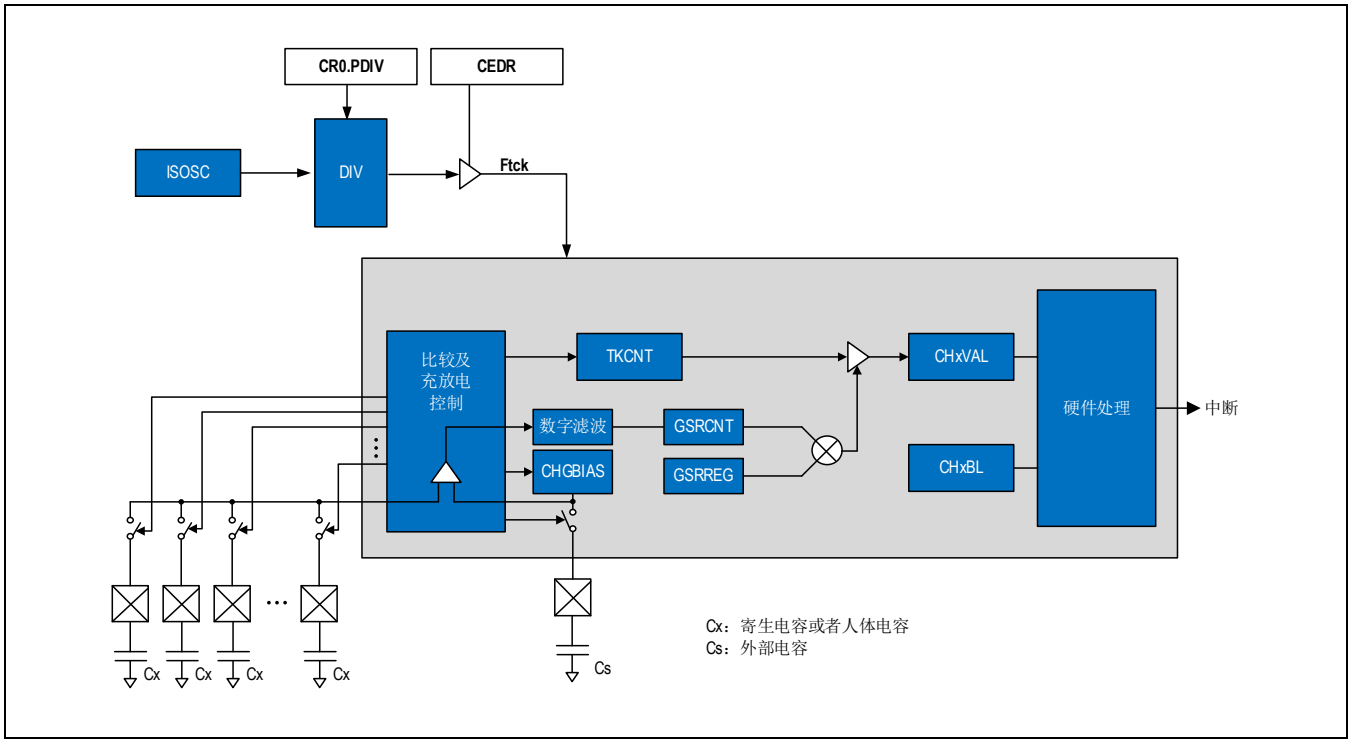


Figure 22-1 Touch Sensor模块框图

22.2.2 工作原理

电容式按键传感器是一种基于自电容检测技术，在人体或带电物体靠近传感极点时，导致自电容的变化，根据这种变化从而实现按键或者触摸滑条等应用的实现。

本芯片采用基于电荷转移的自电容检测。基于电荷转移的电容检测在抗干扰上有很强的优势，但是需要外接调整电容，只支持单路分时扫描模式。

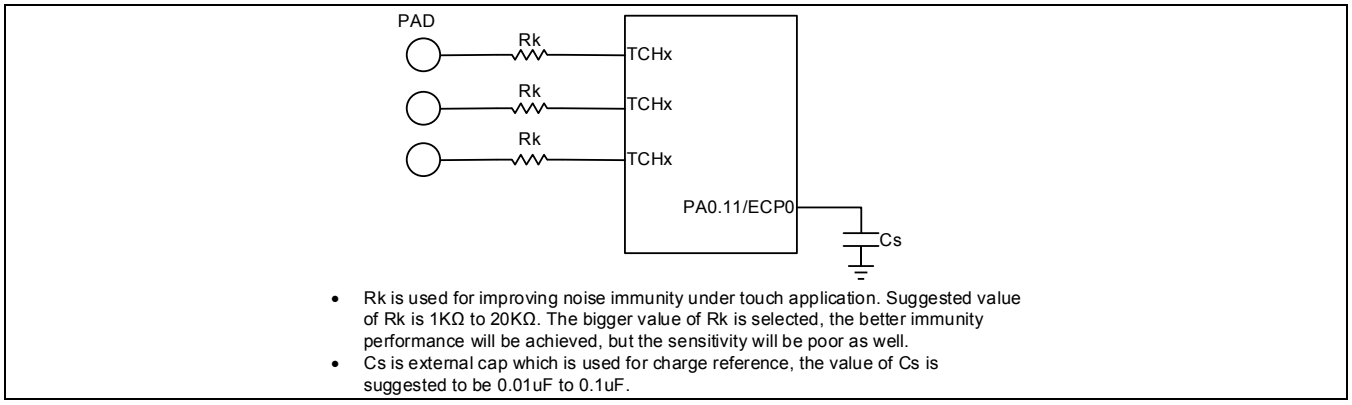
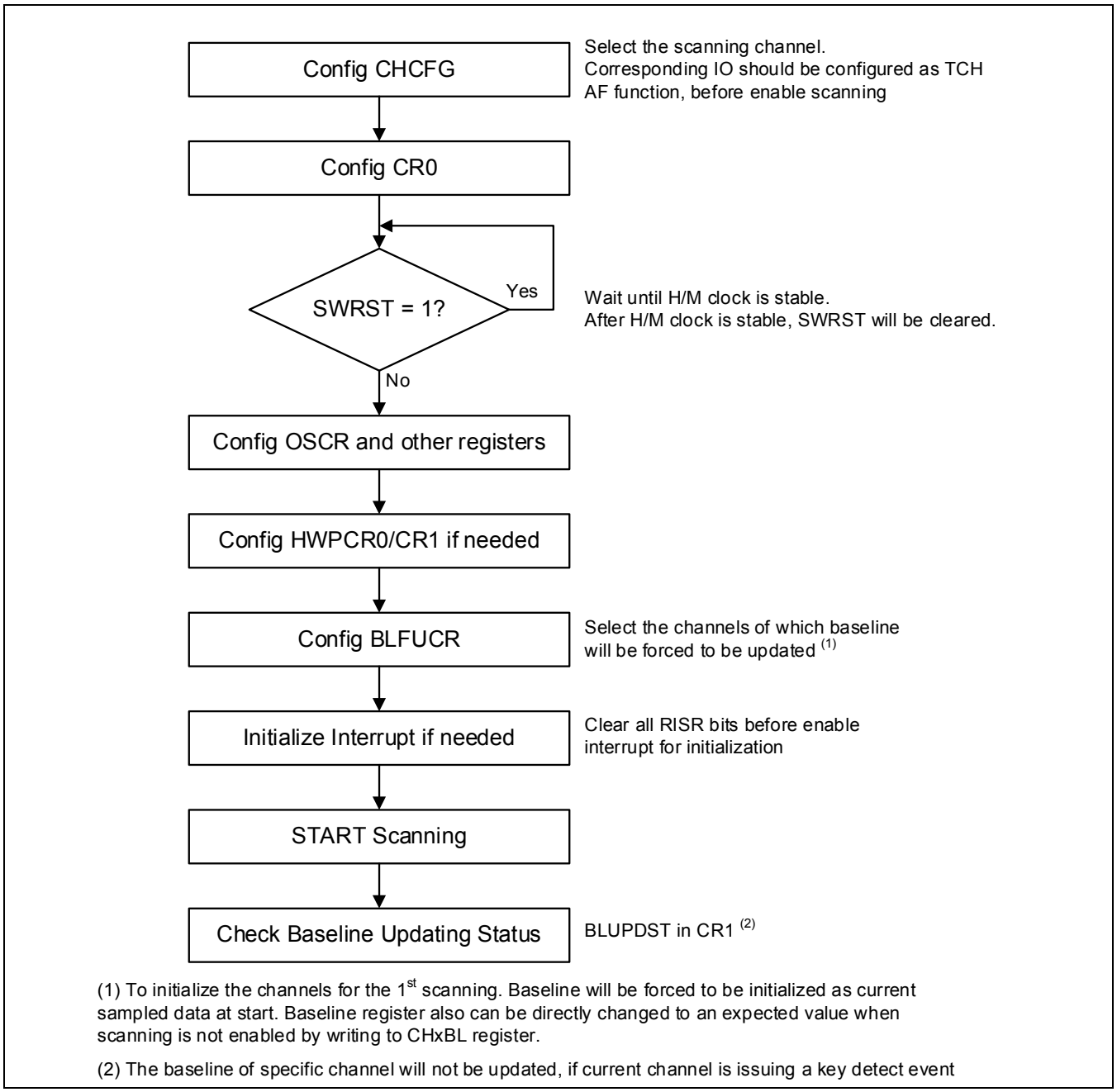


Figure 22-2 Touch Sensor应用连接图

电容检测的所有通道都通过内部开关连接到一个比较器上。开始扫描时，内部充电电路对外部参考电容（Cs）进行充电直到 Vth，然后被扫描通道的开关被打开，外部参考电容上的电荷将和扫描通道上的寄生电容(Cx)进行电荷平衡，平衡后，扫描通道电容(Cx)上的电荷将被放电，然后再继续平衡，如此往复直到外部参考电容上的电荷被放电到 Vtl。充放电的次数将被一个内部采样计数器记录（CHxCNT）。寄生电容变大时，CHxCNT 值会变小；寄生电容变小时，CHxCNT 值会随之变大。

外部参考电容的充电支持两种模式，分别为直接充电和通过恒流源充电，可以根据不同的功耗需求和抗干扰需求进行设置。



(1) To initialize the channels for the 1st scanning. Baseline will be forced to be initialized as current sampled data at start. Baseline register also can be directly changed to an expected value when scanning is not enabled by writing to CHxBL register.

(2) The baseline of specific channel will not be updated, if current channel is issuing a key detect event

Figure 22-3 软件配置流程

22.2.3 硬件数字处理引擎

处理器内嵌硬件数字处理引擎，可以通过硬件比较内部计数器（CHxCNT）和基准的差值来确认按键是否被按下。当检测到按键时，可以通过配置寄存器产生相应CPU的中断。由于集成硬件数字处理，按键扫描可以自动完成，不需要额外的CPU运算进行干预，即使在CPU停止工作时，按键也可以继续自动扫描。当检测到有按键时，可以通过按键中断唤醒CPU。

硬件处理引擎可以对采集数据直接进行滤波和运算，自动控制基准值更新，并根据预设的OFFSET产生硬件中断。每个通道都有一组独立的采样计数器（CHxCNT）和基准值寄存器（BLxVAL），当采样计数器和基准值寄存器之间的差值大于阈值寄存器（TCH_OSCR）所设置的值时，外部按键中断将被触发。由于基准值的自动更新模式存在，所以硬件处理引擎具有自校准能力，会自动随着环境变化，更新基准值以适应不同的应用。基准值的更新方式和优化可以通过寄存器进行设置。硬件处理引擎的使能通过TCH_CR0的HWPROC控制位设置。当硬件处理引擎被禁止时，采样仍旧可以进行，但是基准值处理和按键中断将无效。

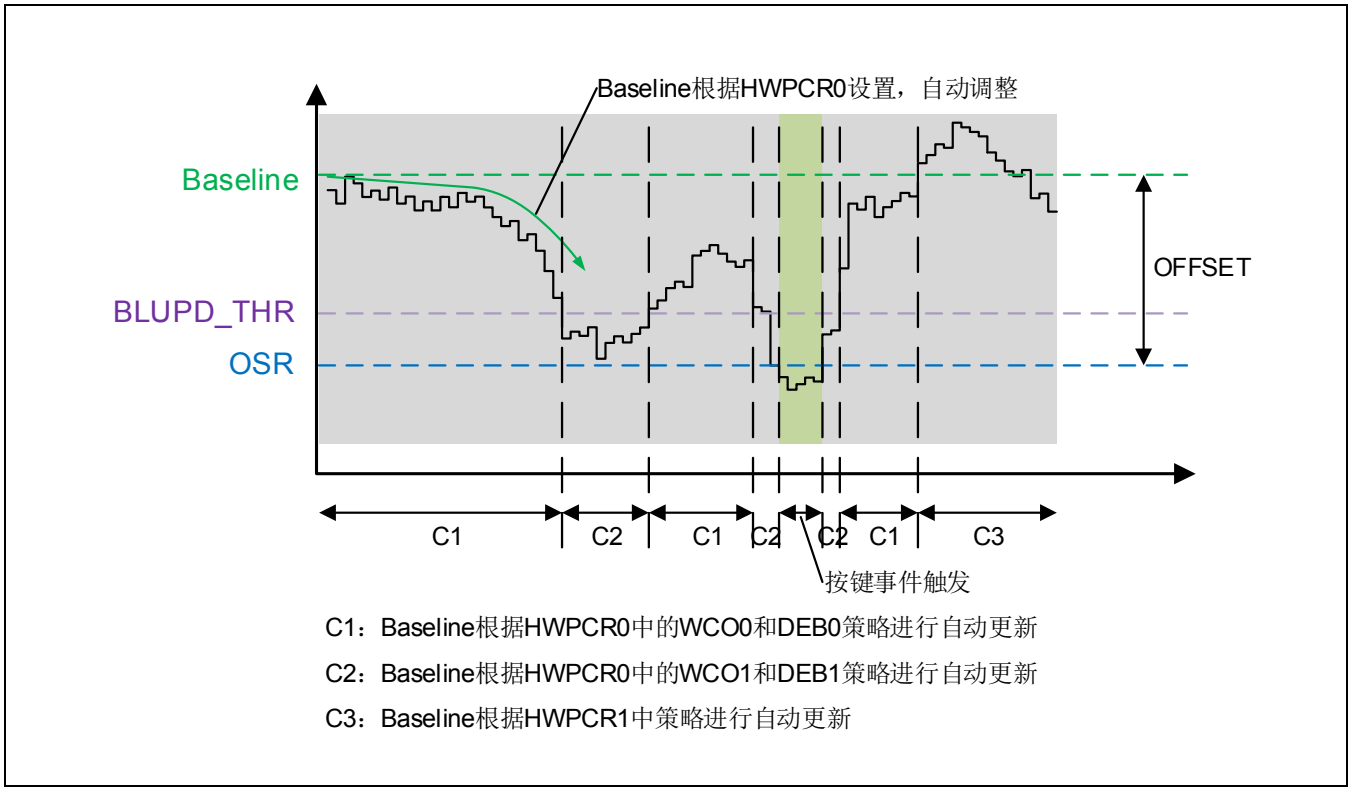


Figure 22-4 按键触发示意图

硬件处理引擎的配置通过TCH_CR1，TCH_BLFUCR，TCH_HWPCR0，TCH_HWPCR1，TCH_TSR，TCH_GSR，TCH_OSRx，TCH_TKCR 这几个寄存器来实现。

22.2.3.1 基准值（BASELINE）的初始化

每个通道的采样基准值（BASELINE）在上电复位后为零，需要通过设置寄存器进行初始化。在某些异常情况，例如基准值在自动更新过程中被错误更新导致按键检测异常等情况下，也需要对基准值进行初始化。

基准值初始化由软件配置寄存器触发，支持两种方式：

- 直接初始化方式（direct access）

- 在线初始化方式（on-the-fly）。

直接初始化：该方式可以直接将希望的基准值写入指定通道的BASELINE寄存器中。更新通过软件对TCH_CHxBL寄存器进行直接写操作，来更新相应通道的BASELINE值。直接更新只有在按键扫描停止时才可以进行，所以必须在按键扫描启动前进行，一旦按键扫描已经启动，必须先停止或者暂停按键的扫描（停止扫描可以通过TCH_ START控制；暂停扫描可以通过TCH_CEDR来进行控制）再对TCH_CHxBL进行写操作，否则写操作将被忽略。

在线初始化：该方式不受当前扫描状态的限制，可以在扫描已经开始后，在不暂停扫描操作的情况下进行。若在启动扫描前，使能在线初始化控制，则在扫描初次启动时自动根据在线初始化的预设配置对按键BASELINE进行更新。若扫描已经启动，则硬件会在下一轮扫描启动后自动执行基准值更新。在线初始化时，可以将当前通道采样计数值作为最新的基准值直接更新到当前通道BASELINE寄存器中，或者将TCH_BLFUDR寄存器中的指定值更新到BASELINE寄存器中。

在线初始化通过写TCH_BLFUCR寄存器来使能。当请求在线初始化时，如果当前扫描轮未结束，则该请求会被挂起，在新一轮扫描开始后，自动执行更新请求，并在所有使能的通道扫描完成后结束。更新状态可以通过读取TCH_CR1的BLUPDST位获得。

在线初始化请求执行时，如果有按键事件发生，检测到按键事件的通道的基准值将不会被更新。而直接初始化方式，不受按键检测事件的影响，无论是否有按键事件发生，通道基准值都会被更新。

22.2.3.2 基准值（BASELINE）的自动更新

在硬件处理引擎使能时，每个通道的基准值可以通过当前采样值通过硬件自动计算，并在满足特定的更新条件时，自动对基准值寄存器进行更新。基准值自动更新的机制，可以实现按键的自适应过程，减少软件处理的负荷，特别是在 CPU 不工作时，由于硬件处理按键，所以可以通过按键中断唤醒 CPU。

基准值的自动更新在采样值满足如下条件时执行：

- 采样值小于基准值，但是采样值和基准值之差未达到OFFSET设定值，且满足去抖检测条件时
- 采样值大于基准值，且满足去抖检测条件时

基准值的计算是通过把当前采样值送入一个IIR型多点平均硬件滤波器自动完成的。IIR滤波器的抽样点数可以设置为2/4/8三种，抽样点越多，滤波结果越平滑，但对比于原始波形的失真度越大。在每一轮扫描结束后，硬件会自动检查基准值的更新条件是否满足去抖次数，当满足时，自动将最新的计算结果更新到基准值中。基准值更新的策略设置可以通过TCH_HWPCR0和TCH_HWPCR1来设置。当采样值小于基准值时，表示采样值处于正向区域内波动，滤波器采用TCH_HWPCR0寄存器中的设置对采样值进行计算，并对BASELINE进行更新。当采样值大于基准值时，表示采样值处于负向区域，滤波器采用TCH_HWPCR1寄存器中的设置对采样值进行计算，并对BASELINE进行更新。

在正向或者负向基准值更新区间内，更新区间通过BLUPD_THR被设置为上下两个半区。当采样值落入上半区时，基准值更新采用BLUPD_WCO0和BLUPD_DEB0的设置策略；当采样值落入下半区时，基准值更新采用BLUPD_WCO1和BLUPD_DEB1的设置策略。

设置TCH_CR1的BLUPDIS位以后，基准值自动更新将被关闭。

22.2.3.3 灵敏度调节

当触摸通道的寄生电容很大，或者需要感应的人体电容很弱时，需要适当的提高通道的灵敏度，以保证采样值在有手指触摸时，能够产生足够的偏差值。

通过设置TCH_TSR和TCH_GSR可以改变每个通道的灵敏度。TCH_TSR为全局灵敏度设置，一共支持4挡灵敏度设置，每个通道可以选择这4挡灵敏度中的任意一个作为本通道的灵敏度设置。当灵敏度越高，对单一通道的循环扫描次数就越多，最终的采样值是多次循环扫描结果的累加，所以采样值也会越大，而扫描的时间就越长。

22.2.3.4 按键检测

每个通道的按键检测都基于采样值和基准值直接的偏差，当偏差大于设置的阈值时，按键事件将会被触发。按键的比较阈值（OFFSET）通过TCH_OSRx设置。每个通道可以根据实际需要设置不同的比较阈值。当按键事件被检测到时，不会立即触发中断。只有当按键去抖条件满足时，才会触发中断。按键的去抖条件可以通过TCH_CR1寄存器中的KEYDETDEB位进行设置。

通过设置TCH_TKCRx选择，按键中断触发条件可以设置为按下，松开，或者按下和松开三种条件。当前所有通道的按键状态，可以通过TCH_TKEYST寄存器查询。

按键模式分为单按键模式和多按键模式，当选择单按键模式时，当已有按键被按下时，其他按键将被忽略。模式选择通过TCH_CR1中MKEYMOD位来设置。

为避免由于异常导致的按键长时间错误触发，硬件支持长时间按键锁死检测功能。当按键被检测到后，会自动启动一个内部计时器，直到按键释放。当该计时器溢出时，硬件会自动产生复位信号，对触摸逻辑进行复位，并产生中断报警。该功能可以通过TCH_CR1的KEYSTICKCHK位进行设置。

22.2.3.5 异常值滤波

处理引擎可以对输入的采样值进行范围限定，对于超出特定范围的按键值，处理引擎会自动忽略处理。每个通道采样值异常的判定基于当前OFFSET的设置，当OFFSET设置越小，限定的异常取值也越小。异常值范围的设定通过TCH_HWPCR中的ABNFLT位进行设置。

22.2.4 扫描启动和扫描触发方式设置

22.2.4.1 扫描工作时钟

整个通道扫描控制模块的基础工作时钟为ISOSC，ISOSC支持两种输出频率，可以根据功耗需求，通过USER OPTION设置ISCLK的频率。扫描工作时钟（Ftck）的频率可以通过TCH_CR0的PDIV来设置分频参数。

22.2.4.2 扫描启动

感应通道的扫描为顺序扫描。顺序扫描时，按照从低到高的通道号依次扫描。在扫描启动前，先应使能触控传感器前端模块（TCH_CR0中的TKEYEN），通过查询TCH_CR0中的SWRST位可以确认是否已经准备就绪。只有在扫描控制时钟就绪以后，才可以启动扫描。扫描通道的扫描时钟需要通过TCH_CEDR寄存器进行使能。

在前端模块和扫描时钟使能后，才可以启动按键扫描。扫描的启动通过设置TCH_START寄存器进行设置。在使能TCH_START以后，通过查询TCH_START状态获知当前的扫描状态。所有硬件处理引擎的控制寄存器在扫描进行时都不能进行改写操作，除了TCH_CR0的ICNT位，TCH_CR1的FORCE_BLUPD位，TCH_CR1的BLUPDIS位和TCH_FUCR寄存器。

22.2.4.3 扫描通道配置

在扫描开始前，作为扫描通道的管脚必须配置为TCHx功能，外部Cs管脚必须配置为ECP0功能。需要扫描的通道通过TCH_CHCFG寄存器设置。所有在TCH_CHCFG中使能的通道扫描完成后，该轮扫描结束。一轮扫描结束后，可以再次设置TCH_START开始下一轮的扫描，或者通过硬件自动触发下一轮扫描。

22.2.4.4 扫描启动触发方式

除软件启动扫描方式，芯片还支持自动硬件触发扫描方式工作。自动扫描模式下，当START控制位使能以后，硬件将会在上一轮扫描结束以后并满足触发条件时，自动启动下一轮扫描。扫描的触发条件可以通过TCH_CR0的STARTCFG位设置。

触发支持三种模式：自动间隔模式，iWDT触发，LED触发。

间隔触发模式：当设置为间隔触发模式时，一轮扫描结束后，系统将自动启动一个12位计数器，此计数器计数频率为扫描工作频率Ftck的128分频。当计数器值等于TCH_CR0的ICNT时，下一轮扫描被触发。间隔扫描还支持自动加速功能，在TCH_CR1的ICNTSWEN位使能后，一旦检测到按键（无论去抖是否满足），扫描间隔计数器比较值将由ICNT自动切换到SICNT；在按键没有被检测到时，自动切换回ICNT。自动扫描模式启动以后，可以通过清除START位来停止自动扫描。当START位被清除以后，内部扫描不会立即停止，而是要等到本轮扫描结束后才会停止。可以通过查询START位状态来判断当前的扫描状态。

iWDT触发方式：在iWDT使能的前提下，发生iWDT中断时自动触发一轮扫描。

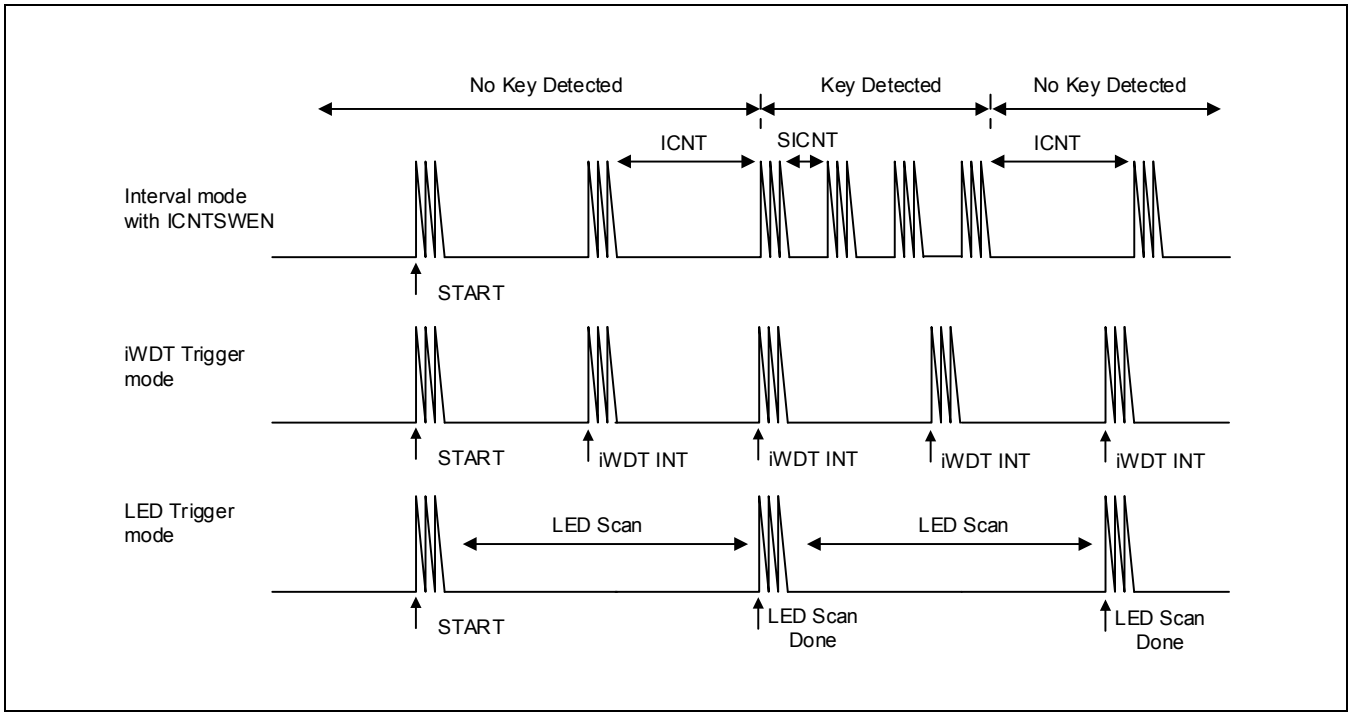


Figure 22-5 自动扫描模式

LED触发方式：该触发方式在LED和TOUCH复用时使用。当设置为LED触发模式时，一轮扫描结束后，系统将等待LED扫描结束信号。LED扫描结束后，触控按键扫描将自动启动。此模式配合LED同时工作，需要将LED控制寄存器中的SHARE模式使能。具体可参考LED章节。

当设置为LED触发模式时，TOUCH的扫描时间长度将会影响到LED的显示亮度。为避免LED闪烁，TOUCH的扫描频率应尽可能设置高，推荐PDIV设置为不分频，并且ISOSC的频率设置为3MHz。

22.2.5 中断产生

22.2.5.1 各个通道扫描完成中断

每个通道都有独立的扫描完成中断。通过设置CHx_DONE 位，可以设置该通道扫描完成后触发相应的通道扫描完成中断。

22.2.5.2 按键中断

在硬件处理引擎被开启以后，一旦有按键检测到以后，该中断会被触发。触发的条件可以通过TCH_TKCR来设置。同时需要考虑设置按键去抖（TCH_CR1的KEYDETDEB）增加抗干扰能力。

22.2.5.3 异常复位中断

在硬件处理引擎被开启以后，为防止某个按键由于错误的更新了基准值（一般为干扰引起），而导致的某个按键长时间处于按下状态，可以通过硬件自动监测功能来复位处理引擎。该监测功能通过设置TCH_CR1中TKEYSTICKCHK位来启动。该监测功能使能以后，一旦有按键按下，自动监测模块即开始计时，当计时超过KEYSTICKDUR的设定值时，该按键仍旧没有释放，将会强制复位处理引擎，并触发异常复位中断。

该异常复位效果等同于软件复位中的硬件处理引擎复位，但是软件复位不会产生该异常中断。该异常中断将自动清除所有的硬件算法控制电路的状态，包括采样值和基准值寄存器。但是配置寄存器的设置将会被保持，除了CR1中的FORCE_BLUPD控制位和START寄存器（扫描需要重新通过软件启动）。

22.2.5.4 基准值更新中断

在有基准值被更新后，该中断立即被触发。程序可以通过查询TCH_BLUPINF寄存器获得被更新的通道号。TCH_BLUPINF不会自动清除，在查询后，需要软件清除该寄存器。

22.2.5.5 全部通道扫描完成中断

所有扫描使能的通道在扫描处理完成后，触发该中断。该中断表示本轮扫描结束。

.

22.3 寄存器说明

22.3.1 寄存器表

- Base Address: 0x4002_0000

Register	Offset	Description	Reset Value
TCH_CR0	0x000	Touch Sensor General Control Register0	0xFFF0_0000
TCH_CR1	0x004	Touch Sensor General Control Register1	0x0850_0100
TCH_HWPCR0	0x008	Hardware Processing Control Register0	0xFF10_7318
TCH_HWPCR1	0x00C	Hardware Processing Control Register1	0xFF10_003C
TCH_BLFUCR	0x010	Baseline Force Updating Control Register	0x0000_0000
TCH_BLFUDR	0x014	Baseline Force Updating Value Set Register	0x0000_0000
TCH_START	0x018	Touch Sensor Start Control Register	0x0000_0000
TCH_CEDR	0x01C	Touch Scan Clock Enable/Disable Register	0x0000_0000
TCH_CHDSTL	0x020	Channel Disable Status Control Low Register	0x0000_0000
TCH_CHDSTH	0x024	Channel Disable Status Control High Register	
RSVD	0x028	Reserved	0x0000_0000
RSVD	0x02C	Reserved	0x0000_0000
TCH_CHCFG	0x030	Touch Sensor Channel Configuration Register	0x0000_0000
TCH_TSRL	0x034	Touch Sensor Sensitivity Select Low Register	0x0000_0000
TCH_TSRH	0x038	Touch Sensor Sensitivity Select High Register	
TCH_GSR	0x03C	Touch Sensor Global Sensitivity Register	0x0000_0000
TCH_OSR0	0x040	Offset Register for Channel 0 to 3	0x0000_0000
TCH_OSR1	0x044	Offset Register for Channel 4 to 7	0x0000_0000
TCH_OSR2	0x048	Offset Register for Channel 8 to 11	0x0000_0000
TCH_OSR3	0x04C	Offset Register for Channel 12 to 15	0x0000_0000
TCH_OSR4	0x050	Offset Register for Channel 16 to 19	0x0000_0000
TCH_TKCRL	0x054	Touch Key Interrupt Trigger Condition Low Register	0x0000_0000
TCH_TKCRH	0x058	Touch Key Interrupt Trigger Condition High Register	0x0000_0000
TCH_RISR	0x05C	Touch Senor Raw Interrupt Status Register	0x0000_0000
TCH_IMCR	0x060	Touch Senor Interrupt Masking Control Register	0x0000_0000
TCH_MISR	0x064	Touch Senor Masked Interrupt Status Register	0x0000_0000
TCH_ICR	0x068	Touch Senor Interrupt Clear Control Register	0x0000_0000
RSVD	0x06C	Reserved	
TCH_CHxCNT[20]	0x070 – 0x0BC	Touch Sensor Channel x Sampling Counter Value	0x0000_0000
RSVD	0x0C0 – 0x0CC	Reserved	

TCH_CHxBL[20]	0x0D0 – 0x11C	Touch Sensor Channel x Baseline Value	0x0000_0000
RSVD	0x120 – 0x12C	Reserved	
TCH_TKEYST	0x130	Touch Key Status Register	0x0000_0000
TCH_BLUPINF	0x134	Baseline Updating Information Register	0x0000_0000

NOTE:

22.3.2 TCH_CR0 (通用控制寄存器0)

- Address = Base Address + 0x0000, Reset Value = 0xFFF0_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICNT												STPENA	HWPROC	STARTCFG		HYST	PDIV			SICNT		BIASADJ	RSVD	SWRST					REFMOD	MODE	TKEYEN
1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
TKEYEN	[0]	RW	触摸按键Hard-macro使能 0： 关闭触控Hard-macro 模块 1： 开启触控Hard-macro 模块	0x0
MODE	[1]	RW	参考电容充电模式选择： 0： 恒流驱动 1： MOS管直接驱动	0x0
REFMOD	[2]	RW	参考电容充电参考点选择： 0： 低功耗模式 1： 高功耗模式	0x0
SWRST	[6:3]	RW	触控模块软件复位： 0001b： 硬件处理引擎复位 0101b： 全模块复位（所有寄存器恢复初始值） Bit3 可以读取到当前复位状态，当读取到‘1’时，表示当前复位没有释放，或者Hard-macro没有使能。	0x0
SICNT	[11:10]	RW	快速间隔扫描计数器	0x0
PDIV	[14:12]	RW	Ftck工作时钟分频控制： **PDIV** / **FREQ** 0/其他 / None-div 1 / 2 div 2 / 4 div 3 / 8 div 4 / 16 div 5 / 32 div	0x0

HYST	[15]	RW	按键检测迟滞设置： 0： 3/4 OFFSET 作为按键释放检测点 1： 1/2 OFFSET 作为按键释放检测点	0x0
STARTCFG	[17:16]	RW	扫描触发设置： 00b： 软件启动扫描 01b： 间隔扫描触发 10b： 硬件触发（iWDT中断） 11b： 硬件触发（LED）	0x0
HWPROC	[18]	RW	硬件处理引擎使能控制： 0： 禁止硬件处理 1： 启动硬件处理	0x0
STPENA	[19]	RW	在CPU停止时，扫描模块工作方式： 0： 禁止扫描模块工作 1： 保持扫描模块工作	0x0
ICNT	[31:20]	RW	间隔扫描时间控制计数器 该计数器以工作时钟ISCLK的128分频工作。 当设置为’0’时，间隔时间为两个周期。	0xFFF

NOTE: 如果需要在程序中多次修改扫描工作模式，请务必在修改模式前执行一次硬件处理引擎复位。

22.3.3 TCH_CR1 (通用控制寄存器1)

- Address = Base Address + 0x0004, Reset Value = 0x0850_0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLUPDIS_CYCLE								BLUPDIS	ICNTSWEN	KEYDETDEB		KEYSTICKDUR			KEYSTICKCHK	RSVD		SAMPLE_FLT		BLUPDINCT				RSVD				MKEYMODE	OFFSET_MUL		BLUPDST
0	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW	RW	R	R	R	R	RW	RW	RW	R

Name	Bit	RW	Description	Reset Value
BLUPDST	[0]	R	查询当前更新基准值操作状态： 0： 基准值更新已结束 1： 基准值更新正在进行	0x0
OFFSET_MUL	[2:1]	RW	OFFSET值自动放大处理。在某些应用上，由于干扰或系统稳定度比较低，造成采样值不稳定。这时需要比较大的OFFSET设置值。OFFSET寄存器最大设置为0xFF，当需要设置更大的OFFSET，可以通过此寄存器对OFFSET设置进行乘积放大。 0： 不进行放大 1： 放大到2倍 2： 放大到4倍 3： 放大到8倍	0x0
MKEYMODE	[3]	RW	按键自动检测模式设置 0： 单按键模式，最多只有一个按键可以被激活 1： 多按键模式，支持多个按键同时激活	0x0
BLUPDICNT	[11:8]	RW	自动基准值更新检测周期设置 0： 1轮扫描为一次检测周期 1： 2轮扫描为一次检测周期 2： 3轮扫描为一次检测周期 3： 4轮扫描为一次检测周期	0x1
SAMPLE_FLT	[13:12]	RW	原始采样值滤波设置 0： 禁止滤波 1： 开启1/2权重滤波	0x0

			2： 开启1/4权重滤波 3： 开启1/8权重滤波	
KEYSTICKCHK	[16]	RW	按键自动检测时，按键状态锁死监测设置 0： 禁止按键锁死监测 1： 开启按键锁死监测	0x0
KEYSTICKDUR	[19:17]	RW	按键锁死监测开启后，异常判读阈值设置。在按键锁死时间超过该阈值时，将自动触发数字处理引擎的复位。 时间阈值 = (KEYSTICKDUR+1) x 16秒	0x0
KEYDETDEB	[21:20]	RW	按键自动检测时，去抖设置 0： 无去抖 1： 1次去抖 2： 2次去抖 3： 3次去抖	0x1
ICNTSWEN	[22]	RW	自动间隔扫描时，扫描间隔周期自动切换设置。当该位被设置以后，在有按键时，自动间隔扫描时间由SICNT决定。在没有检测到按键时，自动间隔扫描时间由ICNT决定。 0： 禁止切换扫描间隔，扫描间隔由ICNT决定 1： 自动切换扫描间隔	0x1
BLUPDIS	[23]	RW	基准值自动更新停止寄存器 0： 总是启动基准值自动更新 1： 总是禁止基准值自动更新	0x0
BLUPDIS_CYCLE	[31:24]	RW	按键释放后，基准值自动更新禁止周期设置。当有按键被检测到时，可以设置接下来的几轮扫描为禁止基准值更新周期，以保证基准值的稳定。	0x8

22.3.4 TCH_HWPCR0 (硬件处理控制寄存器0)

- Address = Base Address + 0x0008, Reset Value = 0xFF10_7318

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLDUP_THVAL								RSVD		ABNFLT		BLUPD_DEB1								BLUPD_DEB0				BLUPD_WCO1				BLUPD_WCO0			
1	1	1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	1	1	1	0	0	1	1	0	0	0	1	1	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
BLUPD_THR	[1:0]	RW	基准值自动更新阈值设置。 00b： 设置更新阈值为1/4 OFFSET 01b： 设置更新阈值为1/2 OFFSET 10b： 设置更新阈值为3/4 OFFSET 11b： 设置更新阈值为BLUPD_THVAL的值	0x0
BLUPD_WCO0	[4:2]	RW	当采样值小于BLUPD_THR所设置的阈值时，基准值更新的权重设置。 0： 基准值不进行更新 1： 以1/8 权重进行更新 2： 以1/4 权重进行更新 3： 以1/2 权重进行更新 other： 用采样值直接进行更新	0x2
BLUPD_WCO1	[7:5]	RW	当采样值大于BLUPD_THR所设置的阈值时，基准值更新的权重设置。 0： 基准值不进行更新 1： 以1/8 权重进行更新 2： 以1/4 权重进行更新 3： 以1/2 权重进行更新 other： 用采样值直接进行更新	0x7
BLUPD_DEB0	[11:8]	RW	当采样值连续多次小于BLUPD_THR所设置的阈值时，处理引擎将根据WCO0设置的条件对基准值进行自动更新。通过该寄存器可以设置连续检测的次数。	0x3
BLUPD_DEB1	[19:12]	RW	当采样值连续多次大于BLUPD_THR所设置的阈值时，处理引擎将根据WCO1设置的条件对基准值进行自动更新。通过该寄存器可以设置连续检测的次数。	0x7

ABNFLT	[21:20]	RW	异常采样值滤波设置，当采样值大于设置阈值，将被处理引擎忽略 0： 大于4倍OFFSET 1： 大于8倍OFFSET 2： 大于16倍OFFSET 3： 禁止滤波功能	0x1
BLUPD_THVAL	[31:24]	RW	自动更新阈值设定值	0xFF

NOTE: 该寄存器在采样值不小于基准值时起作用。当采样值小于基准值时，处理引擎将使用 HWPCR1 中的设置对基准值进行处理。

22.3.5 TCH_HWPCR1 (硬件处理控制寄存器1)

- Address = Base Address + 0x000C, Reset Value = 0xFF10_003C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLDUP_THVAL								RSVD		ABNFLT		BLUPD_DEB1								BLUPD_DEB0				BLUPD_WCO1				BLUPD_WCO0			
1	1	1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
BLUPD_THR	[1:0]	RW	基准值自动更新阈值设置。 00b： 设置更新阈值为1/4 OFFSET 01b： 设置更新阈值为1/2 OFFSET 10b： 设置更新阈值为3/4 OFFSET 11b： 设置更新阈值为BLUPD_THVAL的值	0x0
BLUPD_WCO0	[4:2]	RW	当采样值小于BLUPD_THR所设置的阈值时，基准值更新的权重设置。 0： 基准值不进行更新 1： 以1/8 权重进行更新 2： 以1/4 权重进行更新 3： 以1/2 权重进行更新 other： 用采样值直接进行更新	0x2
BLUPD_WCO1	[7:5]	RW	当采样值大于BLUPD_THR所设置的阈值时，基准值更新的权重设置。 0： 基准值不进行更新 1： 以1/8 权重进行更新 2： 以1/4 权重进行更新 3： 以1/2 权重进行更新 other： 用采样值直接进行更新	0x7
BLUPD_DEB0	[11:8]	RW	当采样值连续多次小于BLUPD_THR所设置的阈值时，处理引擎将根据WCO0设置的条件对基准值进行自动更新。通过该寄存器可以设置连续检测的次数。	0x0
BLUPD_DEB1	[19:12]	RW	当采样值连续多次大于BLUPD_THR所设置的阈值时，处理引擎将根据WCO1设置的条件对基准值进行自动更新。通过该寄存器可以设置连续检测的次数。	0x0

ABNFLT	[21:20]	RW	异常采样值滤波设置，当采样值大于设置阈值，将被处理引擎忽略 0： 大于4倍OFFSET 1： 大于8倍OFFSET 2： 大于16倍OFFSET 3： 禁止滤波功能	0x1
BLUPD_THVAL	[31:24]	RW	自动更新阈值设定值	0xFF

22.3.6 TCH_BLFUCR (基准强制更新控制寄存器)

- Address = Base Address + 0x0010, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												CH19_BLUPD	CH18_BLUPD	CH17_BLUPD	CH16_BLUPD	CH15_BLUPD	CH14_BLUPD	CH13_BLUPD	CH12_BLUPD	CH11_BLUPD	CH10_BLUPD	CH9_BLUPD	CH8_BLUPD	CH7_BLUPD	CH6_BLUPD	CH5_BLUPD	CH4_BLUPD	CH3_BLUPD	CH2_BLUPD	CH1_BLUPD	CH0_BLUPD
R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
CHx_BLUPD	[19:0]	RW	指定需要强制更新基准值的通道号 0： 不启动在线初始化 1： 启动在线初始化	0x0

22.3.7 TCH_BLFUDR (基准强制更新数据寄存器)

- Address = Base Address + 0x0014, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
BLUPD_VAL	[15:0]	RW	强制更新值设定，当BLUPD_VAL设置为全零时，当前通道的采样值将会被强制更新到基准值寄存器中。	0x0

22.3.8 TCH_START (扫描启动控制寄存器)

- Address = Base Address + 0x0018, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
START	[0]	RW	扫描启动控制位 写操作时： 0： 停止扫描 1： 启动扫描 读操作时： 0： 扫描没有开始 1： 扫描正在进行	0x0

22.3.9 TCH_CEDR (扫描时钟控制寄存器)

- Address = Base Address + 0x001C, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
CLKEN	[0]	RW	扫描时钟使能控制： 0： 扫描时钟停止 1： 扫描时钟开启 在扫描过程中，停止时钟后，可以对锁定的控制寄存器进行修改。	0x0
DBGEN	[1]	RW	调试使能控制： 0： 程序调试状态中，程序挂起后，扫描继续 1： 程序调试状态中，程序挂起后，扫描暂停	0x0

22.3.10 TCH_CHDSTL (扫描禁止通道状态控制寄存器)

- Address = Base Address + 0x0020, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSR15		DSR14		DSR13		DSR12		DSR11		DSR10		DSR9		DSR8		DSR7		DSR6		DSR5		DSR4		DSR3		DSR2		DSR1		DSR0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
DSRx	[1:0] [3:2] …… [31:30]	RW	设置在扫描过程中，没有被激活为当前扫描通道的通道状态 00b： 高阻状态 01b： 低电平输出 10b： 高电平输出 11b： 高阻状态	0x0

22.3.11 TCH_CHDSTH (扫描禁止通道状态控制寄存器)

- Address = Base Address + 0x0024, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSR31		DSR30		DSR29		DSR28		DSR27		DSR26		DSR25		DSR24		DSR23		DSR22		DSR21		DSR20		DSR19		DSR18		DSR17		DSR16	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
DSRx	[1:0] [3:2] …… [31:30]	RW	设置在扫描过程中，没有被激活为当前扫描通道的通道状态 00b： 高阻状态 01b： 低电平输出 10b： 高电平输出 11b： 高阻状态	0x0

22.3.12 TCH_CHCFG (扫描通道配置寄存器)

- Address = Base Address + 0x0030, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH31EN	CH30EN	CH29EN	CH28EN	CH27EN	CH26EN	CH25EN	CH24EN	CH23EN	CH22EN	CH21EN	CH20EN	CH19EN	CH18EN	CH17EN	CH16EN	CH15EN	CH14EN	CH13EN	CH12EN	CH11EN	CH10EN	CH9EN	CH8EN	CH7EN	CH6EN	CH5EN	CH4EN	CH3EN	CH2EN	CH1EN	CH0EN
R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
CHxEN	[19..0]	RW	扫描通道使能控制 0： 禁止该通道扫描 1： 使能该通道扫描	0x0

NOTE: 即使在 TCH_CHCFG 中设置了通道使能，如果该通道所对应的 GPIO 功能没有被设置为 TKEY 功能，该通道扫描会被禁止。

22.3.13 TCH_TSRL (通道灵敏度选择寄存器)

- Address = Base Address + 0x0034, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSR15		TSR14		TSR13		TSR12		TSR11		TSR10		TSR9		TSR8		TSR7		TSR6		TSR5		TSR4		TSR3		TSR2		TSR1		TSR0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
TSRx	[1:0] [3:2] …… [31:30]	RW	选择每个通道所对应的灵敏度设置 00b： 选择GSR中的GSR0为当前通道灵敏度值 01b： 选择GSR中的GSR1为当前通道灵敏度值 10b： 选择GSR中的GSR2为当前通道灵敏度值 11b： 选择GSR中的GSR3为当前通道灵敏度值	0x0

22.3.14 TCH_TSRH (通道灵敏度选择寄存器)

- Address = Base Address + 0x0038, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSR31		TSR30		TSR29		TSR28		TSR27		TSR26		TSR25		TSR24		TSR23		TSR22		TSR21		TSR20		TSR19		TSR18		TSR17		TSR16	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
TSRx	[1:0] [3:2] …… [31:30]	RW	选择每个通道所对应的灵敏度设置 00b： 选择GSR中的GSR0为当前通道灵敏度值 01b： 选择GSR中的GSR1为当前通道灵敏度值 10b： 选择GSR中的GSR2为当前通道灵敏度值 11b： 选择GSR中的GSR3为当前通道灵敏度值	0x0

22.3.15 TCH_GSR (全局灵敏度值寄存器)

- Address = Base Address + 0x003C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GSR3								GSR2								GSR1								GSR0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
GSR0	[7:0]	RW	通道灵敏度值0	0x0
GSR1	[15:8]	RW	通道灵敏度值1	0x0
GSR2	[23:16]	RW	通道灵敏度值2	0x0
GSR3	[31:24]	RW	通道灵敏度值3	0x0

NOTE: GSR 的设置值表示对一个通道采样时，Cs 上充放电的循环次数。当设置值为 n 时，表示一次采样由 n+1 次 Cs 的充放电组成。

22.3.16 TCH_OSR0 (偏移量控制寄存器0)

- Address = Base Address + 0x0040, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OS_CH3								OS_CH2								OS_CH1								OS_CH0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
OS_CH0	[7:0]	RW	通道0的偏移（OFFSET）值	0x0
OS_CH1	[15:8]	RW	通道1的偏移（OFFSET）值	0x0
OS_CH2	[23:16]	RW	通道2的偏移（OFFSET）值	0x0
OS_CH3	[31:24]	RW	通道3的偏移（OFFSET）值	0x0

22.3.17 TCH_OSR1 (偏移量控制寄存器1)

- Address = Base Address + 0x0044, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OS_CH7								OS_CH6								OS_CH5								OS_CH4							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
OS_CH4	[7:0]	RW	通道4的偏移（OFFSET）值	0x0
OS_CH5	[15:8]	RW	通道5的偏移（OFFSET）值	0x0
OS_CH6	[23:16]	RW	通道6的偏移（OFFSET）值	0x0
OS_CH7	[31:24]	RW	通道7的偏移（OFFSET）值	0x0

22.3.18 TCH_OSR2 (偏移量控制寄存器2)

- Address = Base Address + 0x0048, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OS_CH11								OS_CH10								OS_CH9								OS_CH8							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
OS_CH8	[7:0]	RW	通道8的偏移（OFFSET）值	0x0
OS_CH9	[15:8]	RW	通道9的偏移（OFFSET）值	0x0
OS_CH10	[23:16]	RW	通道10的偏移（OFFSET）值	0x0
OS_CH11	[31:24]	RW	通道11的偏移（OFFSET）值	0x0

22.3.19 TCH_OSR3 (偏移量控制寄存器3)

- Address = Base Address + 0x004C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OS_CH15								OS_CH14								OS_CH13								OS_CH12							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
OS_CH12	[7:0]	RW	通道12的偏移（OFFSET）值	0x0
OS_CH13	[15:8]	RW	通道13的偏移（OFFSET）值	0x0
OS_CH14	[23:16]	RW	通道14的偏移（OFFSET）值	0x0
OS_CH15	[31:24]	RW	通道15的偏移（OFFSET）值	0x0

22.3.20 TCH_OSR4 (偏移量控制寄存器4)

- Address = Base Address + 0x0050, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OS_CH19								OS_CH18								OS_CH17								OS_CH16							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
OS_CH16	[7:0]	RW	通道16的偏移（OFFSET）值	0x0
OS_CH17	[15:8]	RW	通道17的偏移（OFFSET）值	0x0
OS_CH18	[23:16]	RW	通道18的偏移（OFFSET）值	0x0
OS_CH19	[31:24]	RW	通道19的偏移（OFFSET）值	0x0

22.3.21 TCH_TKCRL (按键中断触发条件控制寄存器)

- Address = Base Address + 0x0054, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TKCR15		TKCR14		TKCR13		TKCR12		TKCR11		TKCR10		TKCR9		TKCR8		TKCR7		TKCR6		TKCR5		TKCR4		TKCR3		TKCR2		TKCR1		TKCR0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
TKCRx	[1:0] [3:2] …… [31:30]	RW	选择每个通道按键中断触发方式 00b： 选择按键按下事件为中断触发方式 01b： 选择按键按下事件为中断触发方式 10b： 选择按键松开事件为中断触发方式值 11b： 选择按键按下/松开事件为中断触发方式	0x0

22.3.22 TCH_TKCRH (按键中断触发条件控制寄存器)

- Address = Base Address + 0x0058, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TKCR31		TKCR30		TKCR29		TKCR28		TKCR27		TKCR26		TKCR25		TKCR24		TKCR23		TKCR22		TKCR21		TKCR20		TKCR19		TKCR18		TKCR17		TKCR16	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
TKCRx	[1:0] [3:2] …… [31:30]	RW	选择每个通道按键中断触发方式 00b： 选择按键按下事件为中断触发方式 01b： 选择按键按下事件为中断触发方式 10b： 选择按键松开事件为中断触发方式值 11b： 选择按键按下/松开事件为中断触发方式	0x0

22.3.23 TCH_RISR (原始中断状态寄存器)

- Address = Base Address + 0x005C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRCDNE	BLUPD	RSVD	HWRST	KEYINT	RSVD							CH19_DNE	CH18_DNE	CH17_DNE	CH16_DNE	CH15_DNE	CH14_DNE	CH13_DNE	CH12_DNE	CH11_DNE	CH10_DNE	CH9_DNE	CH8_DNE	CH7_DNE	CH6_DNE	CH5_DNE	CH4_DNE	CH3_DNE	CH2_DNE	CH1_DNE	CH0_DNE
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	RW	Description	Reset Value
CHx_DNE	[19..0]	R	每个通道扫描完成中断	0x0
KEYINT	[27]	R	触摸按键中断	0x0
HWRST	[28]	R	异常复位中断	0x0
BLUPD	[30]	R	基准值更新中断	0x0
PRCDNE	[31]	R	全部通道扫描完成中断	0x0

该寄存器用于查询原始中断标志状态。一旦事件被触发，无论中断是否使能，该寄存器都会记录状态。

0： 原始中断未发生
1： 原始中断发生

22.3.24 TCH_IMCR (中断使能控制寄存器)

- Address = Base Address + 0x0060, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRCDNE	BLUPD	RSVD	HWRST	KEYINT	RSVD							CH19_DNE	CH18_DNE	CH17_DNE	CH16_DNE	CH15_DNE	CH14_DNE	CH13_DNE	CH12_DNE	CH11_DNE	CH10_DNE	CH9_DNE	CH8_DNE	CH7_DNE	CH6_DNE	CH5_DNE	CH4_DNE	CH3_DNE	CH2_DNE	CH1_DNE	CH0_DNE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
CHx_DNE	[19..0]	RW	每个通道扫描完成中断	0x0
KEYINT	[27]	RW	触摸按键中断	0x0
HWRST	[28]	RW	异常复位中断	0x0
BLUPD	[30]	RW	基准值更新中断	0x0
PRCDNE	[31]	RW	全部通道扫描完成中断	0x0

该寄存器用于使能中断。

0： 关闭中断
1： 打开中断

22.3.25 TCH_MISR (中断状态控制寄存器)

- Address = Base Address + 0x0064, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRCDNE	BLUPD	RSVD	HWRST	KEYINT	RSVD							CH19_DNE	CH18_DNE	CH17_DNE	CH16_DNE	CH15_DNE	CH14_DNE	CH13_DNE	CH12_DNE	CH11_DNE	CH10_DNE	CH9_DNE	CH8_DNE	CH7_DNE	CH6_DNE	CH5_DNE	CH4_DNE	CH3_DNE	CH2_DNE	CH1_DNE	CH0_DNE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	RW	Description	Reset Value
CHx_DNE	[19..0]	R	每个通道扫描完成中断	0x0
KEYINT	[27]	R	触摸按键中断	0x0
HWRST	[28]	R	异常复位中断	0x0
BLUPD	[30]	R	基准值更新中断	0x0
PRCDNE	[31]	R	全部通道扫描完成中断	0x0

该寄存器用于查询触发中断的中断源。

0： 中断未触发
1： 中断触发

22.3.26 TCH_ICR (中断清除控制寄存器)

- Address = Base Address + 0x0068, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRCDNE	BLUPD	RSVD	HWRST	KEYINT	RSVD							CH19_DNE	CH18_DNE	CH17_DNE	CH16_DNE	CH15_DNE	CH14_DNE	CH13_DNE	CH12_DNE	CH11_DNE	CH10_DNE	CH9_DNE	CH8_DNE	CH7_DNE	CH6_DNE	CH5_DNE	CH4_DNE	CH3_DNE	CH2_DNE	CH1_DNE	CH0_DNE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	RW	Description	Reset Value
CHx_DNE	[19..0]	W	每个通道扫描完成中断	0x0
KEYINT	[27]	W	触摸按键中断	0x0
HWRST	[28]	W	异常复位中断	0x0
BLUPD	[30]	W	基准值更新中断	0x0
PRCDNE	[31]	W	全部通道扫描完成中断	0x0

该寄存器用于清除原始中断源标志位，该寄存器为只写寄存器

0： 无效果
1： 清除中断标志

22.3.27 TCH_CHxCNT (采样计数器值寄存器)

- Address = Base Address + 0x006C ~ Base Address + 0x00B8, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
TCH_CHxCNT[0]	[15..0]	R	通道0原值采样值寄存器	0x0
TCH_CHxCNT[1]	[15..0]	R	通道1原值采样值寄存器	0x0
TCH_CHxCNT[2]	[15..0]	R	通道2原值采样值寄存器	0x0
TCH_CHxCNT[3]	[15..0]	R	通道3原值采样值寄存器	0x0
TCH_CHxCNT[4]	[15..0]	R	通道4原值采样值寄存器	0x0
TCH_CHxCNT[5]	[15..0]	R	通道5原值采样值寄存器	0x0
TCH_CHxCNT[6]	[15..0]	R	通道6原值采样值寄存器	0x0
TCH_CHxCNT[7]	[15..0]	R	通道7原值采样值寄存器	0x0
TCH_CHxCNT[8]	[15..0]	R	通道8原值采样值寄存器	0x0
TCH_CHxCNT[9]	[15..0]	R	通道9原值采样值寄存器	0x0
TCH_CHxCNT[10]	[15..0]	R	通道10原值采样值寄存器	0x0
TCH_CHxCNT[11]	[15..0]	R	通道11原值采样值寄存器	0x0
TCH_CHxCNT[12]	[15..0]	R	通道12原值采样值寄存器	0x0
TCH_CHxCNT[13]	[15..0]	R	通道13原值采样值寄存器	0x0
TCH_CHxCNT[14]	[15..0]	R	通道14原值采样值寄存器	0x0
TCH_CHxCNT[15]	[15..0]	R	通道15原值采样值寄存器	0x0
TCH_CHxCNT[16]	[15..0]	R	通道16原值采样值寄存器	0x0
TCH_CHxCNT[17]	[15..0]	R	通道17原值采样值寄存器	0x0
TCH_CHxCNT[18]	[15..0]	R	通道18原值采样值寄存器	0x0
TCH_CHxCNT[19]	[15..0]	R	通道19原值采样值寄存器	0x0

22.3.28 TCH_CHxBL (基准值寄存器)

- Address = Base Address + 0x00D0 ~ Base Address + 0x011C, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
TCH_CHxBL[0]	[15..0]	RW	通道0基准值寄存器	0x0
TCH_CHxBL[1]	[15..0]	RW	通道1基准值寄存器	0x0
TCH_CHxBL[2]	[15..0]	RW	通道2基准值寄存器	0x0
TCH_CHxBL[3]	[15..0]	RW	通道3基准值寄存器	0x0
TCH_CHxBL[4]	[15..0]	RW	通道4基准值寄存器	0x0
TCH_CHxBL[5]	[15..0]	RW	通道5基准值寄存器	0x0
TCH_CHxBL[6]	[15..0]	RW	通道6基准值寄存器	0x0
TCH_CHxBL[7]	[15..0]	RW	通道7基准值寄存器	0x0
TCH_CHxBL[8]	[15..0]	RW	通道8基准值寄存器	0x0
TCH_CHxBL[9]	[15..0]	RW	通道9基准值寄存器	0x0
TCH_CHxBL[10]	[15..0]	RW	通道10基准值寄存器	0x0
TCH_CHxBL[11]	[15..0]	RW	通道11基准值寄存器	0x0
TCH_CHxBL[12]	[15..0]	RW	通道12基准值寄存器	0x0
TCH_CHxBL[13]	[15..0]	RW	通道13基准值寄存器	0x0
TCH_CHxBL[14]	[15..0]	RW	通道14基准值寄存器	0x0
TCH_CHxBL[15]	[15..0]	RW	通道15基准值寄存器	0x0
TCH_CHxBL[16]	[15..0]	RW	通道16基准值寄存器	0x0
TCH_CHxBL[17]	[15..0]	RW	通道17基准值寄存器	0x0
TCH_CHxBL[18]	[15..0]	RW	通道18基准值寄存器	0x0
TCH_CHxBL[19]	[15..0]	RW	通道19基准值寄存器	0x0

22.3.29 TCH_TKEYST (触控按键状态寄存器)

- Address = Base Address + 0x0130, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												TKEYST19	TKEYST18	TKEYST17	TKEYST16	TKEYST15	TKEYST14	TKEYST13	TKEYST12	TKEYST11	TKEYST10	TKEYST9	TKEYST8	TKEYST7	TKEYST6	TKEYST5	TKEYST4	TKEYST3	TKEYST2	TKEYST1	TKEYST0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	RW	Description	Reset Value
TKEYSTx	[19..0]	R	当前通道按键状态查询值	0x0

22.3.30 TCH_BLUPINF (基准值更新状态寄存器)

- Address = Base Address + 0x013C, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												BLUPD_CH19	BLUPD_CH18	BLUPD_CH17	BLUPD_CH16	BLUPD_CH15	BLUPD_CH14	BLUPD_CH13	BLUPD_CH12	BLUPD_CH11	BLUPD_CH10	BLUPD_CH9	BLUPD_CH8	BLUPD_CH7	BLUPD_CH6	BLUPD_CH5	BLUPD_CH4	BLUPD_CH3	BLUPD_CH2	BLUPD_CH1	BLUPD_CH0
R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
BLUPD_CHx	[19..0]	RW	基准值自动更新通道信息查询 被自动更新过的通道对应位将被置位成'1'。该寄存器不会自动清除，只有通过软件清除。	0x0

NOTE: 在每次需要获取基准值更新信息前，必须通过软件对该寄存器进行清零处理，从而获得从清零时间点开始后的更新记录。

23 电气特性

23.1 极限参数

器件在超过下述“极限参数”条件下工作可能会造成永久损坏。 器件只有在说明书所规定的条件范围内才能确保正常工作，在“极限参数”条件下工作会影响器件的可靠性。

Table 23-1 极限参数

参数	符号	条件	数值	单位
工作电压	V_{DD}	–	–0.1 to 6.5	V
输入电压	V_{IN}	–	–0.1 to V_{DD} + 0.3	V
输出电压	V_O	所有端口	–0.1 to V_{DD} + 0.3	V
		普通端口	15	mA
		强下拉驱动端口	120	mA
上拉电流	I_{OH}	所有端口	15	mA
工作环境温度	T_A	–	–40 to 85	°C
储存温度	T_{STG}	–	–65 to 150	°C

23.2 推荐工作条件

器件需要在推荐的工作条件下才能正常工作。 本章所列电气特性参数需要在推荐条件下才能得到确保。器件在超出推荐条件以外的工作条件下工作可能会降低其可靠性，甚至造成器件损坏。

Table 23-2 推荐工作条件

参数	符号	条件	数值	单位
工作电压	V_{DD}	–	2.4 to 5.5	V
工作环境温度	T_A	–	–40 to 85	°C

23.3 I/O 端口特性

Table 23-3 I/O 端口特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
输入高电压	V_{IH}	所有端口	0.7 V_{DD}	–	V_{DD}	V
输入低电压	V_{IL}	所有端口	–	–	0.3 V_{DD}	V
输出高电压	V_{OH}	I_{OH} = -15mA, V_{DD} = 5V	V_{DD} – 1.0	–	–	V
输出低电压	V_{OL1}	I_{OL1} = 15mA , V_{DD} = 5V （所有端口）	–	–	1	V
	V_{OL2}	I_{OL2} = 120mA , V_{DD} = 5V (PA0.0, PB0.0, PB0.1, PC0.0, PC0.1, PA1.3 ~ PA1.5强下拉驱动模式)	–	–	1	V
高输入漏电流	I_{LIH}	所有端口, V_{IN} = V_{DD}	–	–	1	uA
低输入漏电流	I_{LIL}	所有端口, V_{IN} = 0	–	–	–1	uA
上拉电阻	R_{PU}	V_{DD} = 5V, V_{IN} = 0V	25	50	75	kΩ
下拉电阻	R_{PD}	V_{DD} = 5V, V_{IN} = 5V	25	50	75	kΩ

23.4 I/O 端口交流特性

Table 23-4 I/O 端口交流特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
输入最大频率	IOF_{IN}	所有端口		10		MHz
输出最大频率	IOF_{OUT}	所有端口		10		MHZ

23.5 输入复位特性

Table 23-5 输入复位特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
最小低压脉宽	T_{NRST}	-	100	300	500	nS
nRESET 迟滞电压	V_{hyst}	上升/下降		1		V

NOTE: 输入复位信号的滤波器宽度为 100ns 至 500 ns。
如果输入复位信号宽度低于 100ns 将被认为无效信号（不复位）。
如果输入复位信号宽度高于 500ns 将被认为有效信号（复位）。

Figure 23-1 nRESET 输入时序

23.6 上电和掉电复位特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
上电电源变化速率	SR_{VDD}	-	0.1	-	-	V/mS
掉电 (Brown-out) 复位电压	V_{BO}	-	-	0.3	-	V
掉电保持时间	T_{BO}	-	10	-	-	mS

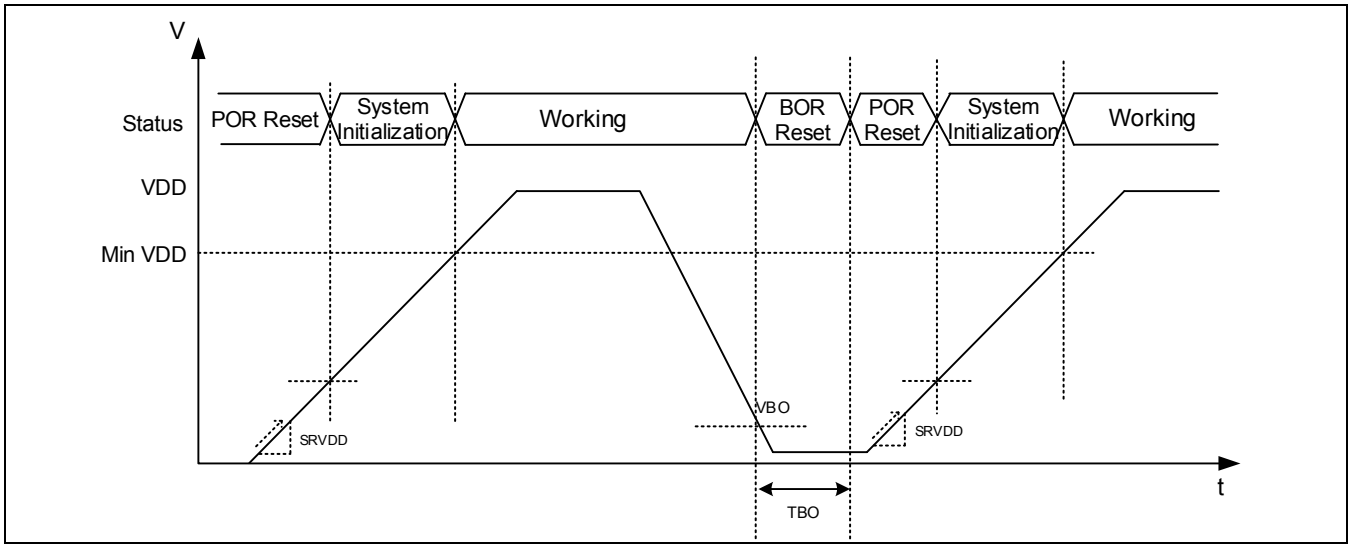


Figure 23-2 上电和掉电示意图

23.7 外部中断输入特性

Table 23-6 外部中断输入特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
中断输入高脉宽	t_{INTH}	V_{DD} = 5.0V	15	30	45	nS
中断输入低脉宽	t_{INTL}	V_{DD} = 5.0V	15	30	45	nS

NOTE: 输入复位信号的滤波器宽度为 15ns 至 45 ns。
如果输入复位信号宽度低于 15ns 将被认为无效信号。
如果输入复位信号宽度高于 45ns 将被认为有效信号。

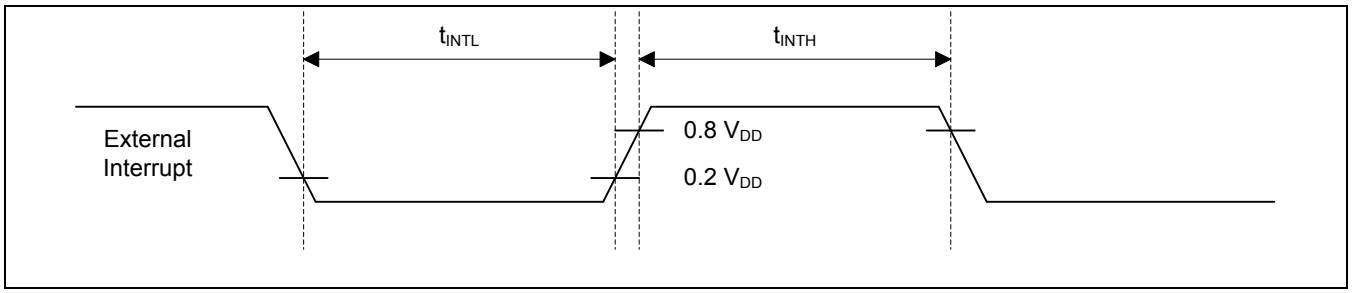


Figure 23-3 外部中断输入时序

23.8 振荡器特性

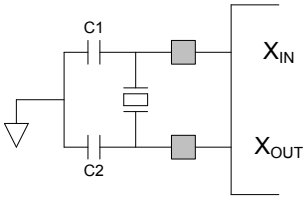
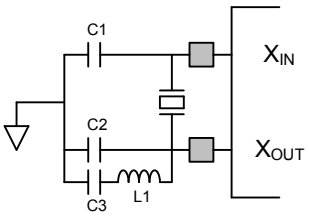
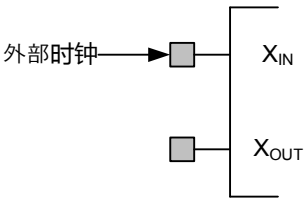
系统中包括三种振荡器：

- 外部主振荡器
- 内部主振荡器
- 内部副振荡器

23.8.1 外部主振荡器

Table 23-7 外部主振荡器特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
振荡器频率	F_{EMOSC}	-	0.4	-	24	Mhz
内部反馈电阻	R_{FD}	XIN 端口	2	4	10	MΩ
稳定时间	T_{STA}	–	–	20	-	ms
外接晶振	–		0.4	–	24	MHz
外接晶振(仅V_{DD} = 5V)	–			40		MHz
外部时钟	–		0.4	–	24	MHz

23.8.2 内部主振荡器特性

Table 23-8 内部主振荡器特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
振荡器频率	F_{IMOSC}	模式1	-	20	-	Mhz
		模式2		40		Mhz
占空比	T_{OD}	-	40	-	60	%
校准后精度	T_{ACC}	T_A = 27°C	-	±1	-	%
		T_A = –40 to 85°C	-	±2	-	%
稳定时间	T_{STA}	电源电压达到最低工作值后	–	–	10	Clk

23.8.3 内部副振荡器特性

Table 23-9 内部副振荡器特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
振荡器频率	F_{IMOSC}	模式1	-	0.5	-	Mhz
		模式2	-	3.0	-	
占空比	T_{OD}	-	40	-	60	%
精度	T_{ACC}		-50	-	+50	%
稳定时间	T_{STA}	电源电压达到最低工作值后	–	–	10	Clk

23.9 工作电流

Table 23-10 工作电流

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	说明	条件	状态名称	最小值	典型值	最大值	单位
工作电流	I_{DD1}	正常工作	-	RUN	–	10	–	mA
	I_{DD2}	CPU 时钟关闭	-	SLEEP	–	1	–	mA
	I_{DD3}	所有时钟及模拟模块关闭	V_{DD} = 5.0V , T_A = 25°C	DEEP SLEEP	–	0.7	5	uA
			V_{DD} = 2.4V to 5.5V , T_A = –40 to 85°C		–	0.7	10	

NOTE: 工作电流不包括 I/O 端口的上拉、下拉电流.

23.10 低压复位监测特性

Table 23-11 低压复位检测特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
低压复位电压 (V_{DD} 下降沿)	Vthrf	–	2.05	2.15	2.25	V
		–	2.65	2.75	2.85	
		–	3.25	3.35	3.45	
		–	3.55	3.65	3.75	
低压监测电压 (V_{DD} 下降沿)	Vthdf	–	2.45	2.55	2.65	
		–	2.90	3.00	3.10	
		–	3.80	3.90	4.00	
		–	4.00	4.10	4.20	
迟滞电压	ΔV_{LVD}	–	–	200	–	mV
工作电流	I_{CC}	–	–	9	–	uA
关断电流	I_{PD}	–	–	0.1	–	uA

23.11 12位模/数转换器特性（ADC 12位模式）

Table 23-12 12位模/数转换器特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
精度	–	–	–	12	–	Bit
工作电压	V_{ADC}	–	3.0	5	5.5	V
基准参考电压	V_{REF}	V_{REF} <V_{ADC}	2	5	5.5	V
输入电压范围	V_{AIN}	–	0	–	V_{REF}	V
转换速率	F_S	–	–	–	1	MHz
微分非线性	DNL	F_S = 0.5MHz V_{ADC} = 5V	–	-	±2.0	LSB
积分非线性	INL		–	-	±4.0	
偏移误差	TOPOFF		–	-	±10.0	
	BOTOFF		–	-	±10.0	
工作电流	I_{OP}	–	–	1	–	mA
关断电流	I_{PD}	–	–	1	–	μA

NOTE: 以上数据为应用评估结果，非量产测试结果。

23.12 10位模/数转换器特性（ADC 10位模式）

Table 23-13 10位模/数转换器特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
精度	–	–	–	10	–	Bit
工作电压	V_{ADC}	–	3.0	5	5.5	V
基准参考电压	V_{REF}	V_{REF} <V_{ADC}	2	5	5.5	V
输入电压范围	V_{AIN}	–	0	–	V_{REF}	V
转换速率	F_S	–	–	–	1	MHz
微分非线性	DNL	F_S = 1MHz V_{ADC} = 5V	–	-	±1.0	LSB
积分非线性	INL		–	-	±2.0	
偏移误差	TOPOFF		–	-	±4.0	
	BOTOFF		–	-	±4.0	
工作电流	I_{OP}	–	–	1	–	mA
关断电流	I_{PD}	–	–	1	–	μA

NOTE: 以上数据为应用评估结果，非量产测试结果。

23.13 运算放大器特性

Table 23-14 运算放大器特性

(T_A = –40 to 85°C, V_{DD} = 3V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
工作电压	V_{AMP}	–	3.0	5	5.5	V
输入失调电压	V_{OFF}	–	–	1	3	mV
输入共模电压	V_{ICM}	–	0	–	V_{DD} – 1.5	V
转换速率	SR	–	5	10	–	V/us
输出电压范围	V_{OUT}	–	0.2	–	V_{DD} – 0.2	V
输出电流	I_{OUT}	V_{DD} = 5V 1V<V_{OUT} <4V	10	–	–	mA
增益带宽积	GBW	–	2	5		MHz
开环增益	GA_{OPEN}	–		80		dB

23.14 比较器特性

Table 23-15 比较器特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
输入失调电压	V_{OFF}	–	–	1	3	mV
输入共模电压	V_{ICM}	–	0	–	V_{DD} – 1.5	V
响应时间(1)	T_{RESP}	差分输入1mV	–	100	–	nS
		差分输入10mV		50		
		差分输入100mV		40		
迟滞电压	V_{HYST}	模式0	–	0	–	mV
		模式1	–	80	–	
		模式2	–	100	–	
		模式3	–	150	–	

NOTE: 1）该响应时间为比较器本体响应时间，如开启后续数字滤波器，需要增加数字滤波器延迟，参考 9.比较器。

23.15 内部参考电压特性

Table 23-16 内部参考电压源特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
低参考电压	FVR_L	V_{DD} > 2.7V	2.007 (-2%)	2.048	2.089 (+2%)	V
高参考电压	FVR_H	V_{DD} = 5V	4.014 (-2%)	4.096	4.178 (+2%)	V

23.16 存储器特性

Table 23-17 RAM和寄存器的特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
数据保持电压(1)	V_{DDDR}	深睡眠模式	1.2	–	V_{DD}	V

NOTE: 1）保证 RAM 中的数据不丢失的最低电压值（深睡眠模式下），或者是保持寄存器的状态的最低电压值（深睡眠模式下）。由设计保证，不在量产中测试。

Table 23-18 FLASH内存的特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
编程大小	F_{WSIZE}	–	–	4	–	Byte
页面大小	F_{PSIZE}	–	–	1024	–	Byte
编程时间（1Word）	F_{tprog}	–	20	–	–	us
页擦除时间	F_{tpera}	–	2	–	–	ms
全芯片擦除时间	F_{tmera}	–	10	–	–	ms
编程次数	F_{nwe}	–	200,000	–	–	Times
数据保持时间	F_{tdr}	–	20	–	–	Years
功耗（编程或擦除时）	F_{idd}	–	–	–	5	mA

23.17 静电防护（ESD）特性

Table 23-19 静电防护特性

参数	符号	模型	最小值	典型值	最大值	单位
静电防护耐压	V_{ESD}	HBM	4000	–	–	V
		MM	200	–	–	V
		CDM	500	–	–	V

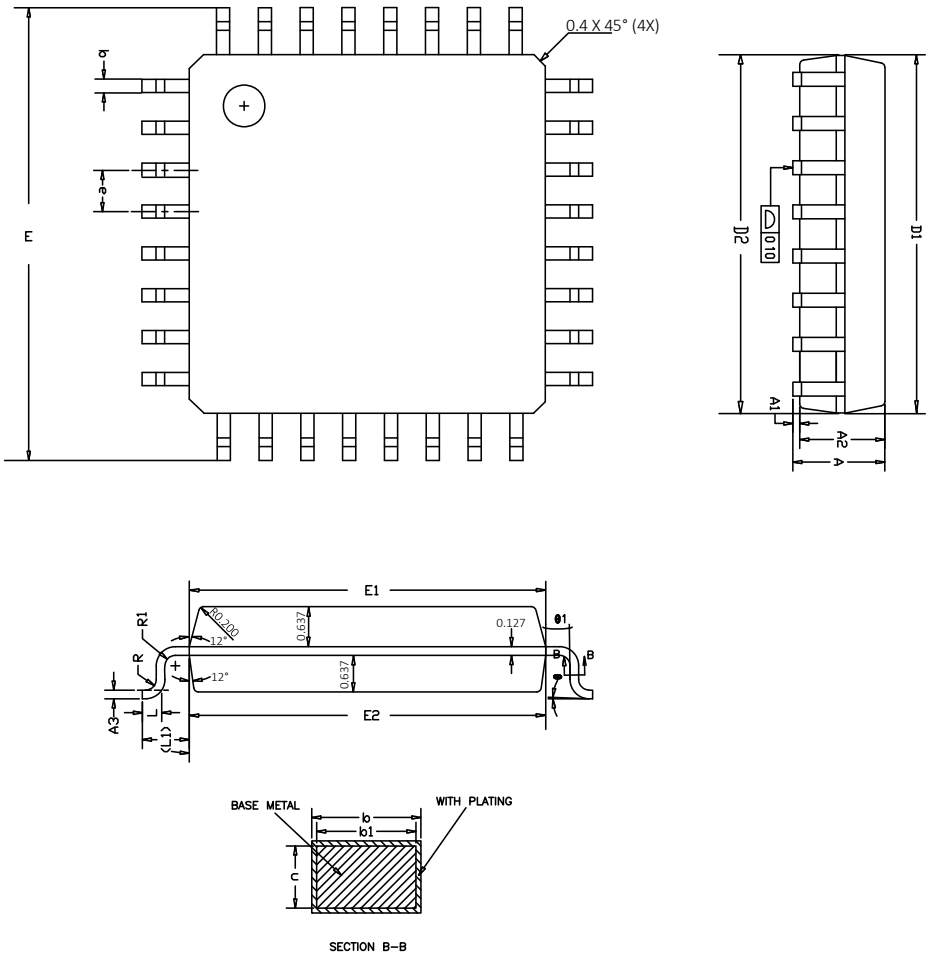
封装尺寸

Package Dimension

LQFP32 / QFN32 / SOP28 / SSOP24

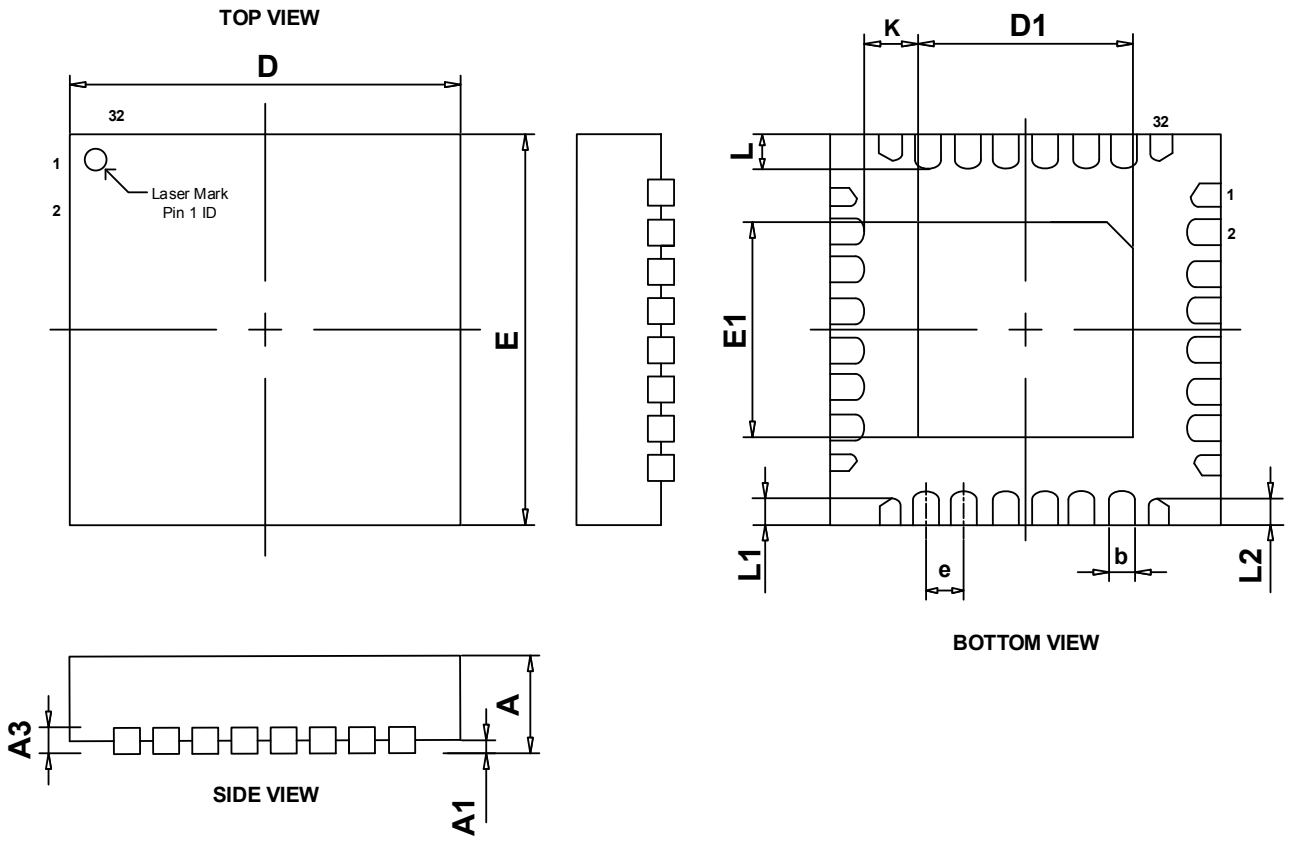
Revision 1.2

May 2018



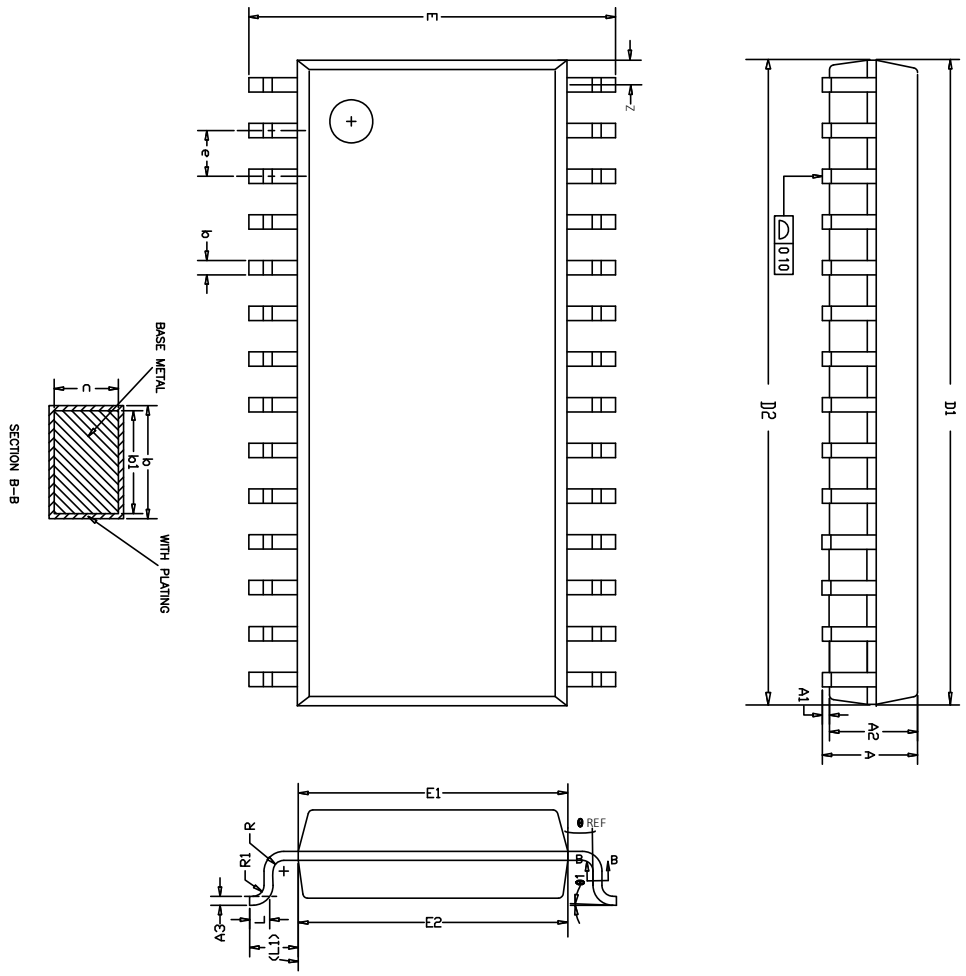
尺寸 / 标注	最小(mm)	正常(mm)	最大(mm)	尺寸 / 标注	最小(mm)	正常(mm)	最大(mm)
A	1.450	1.550	1.650	E1	6.850	6.950	7.050
A1	0.010	-	0.210	E2	6.900	7.000	7.100
A2	1.300	1.400	1.500	e	-	0.800	-
A3	-	0.254	-	L	0.430	-	0.710
b	0.300	0.350	0.400	L1	0.900	1.000	1.100
b1	0.310	0.370	0.430	R	0.100	-	0.250
c	-	0.127	-	R1	0.100	-	-
D1	6.850	6.950	7.050	θ	0.000	-	10°
D2	6.900	7.000	7.100	θ1	0.000	-	-
E	8.800	9.000	9.200				

LQFP32 (0.8mm)封装尺寸



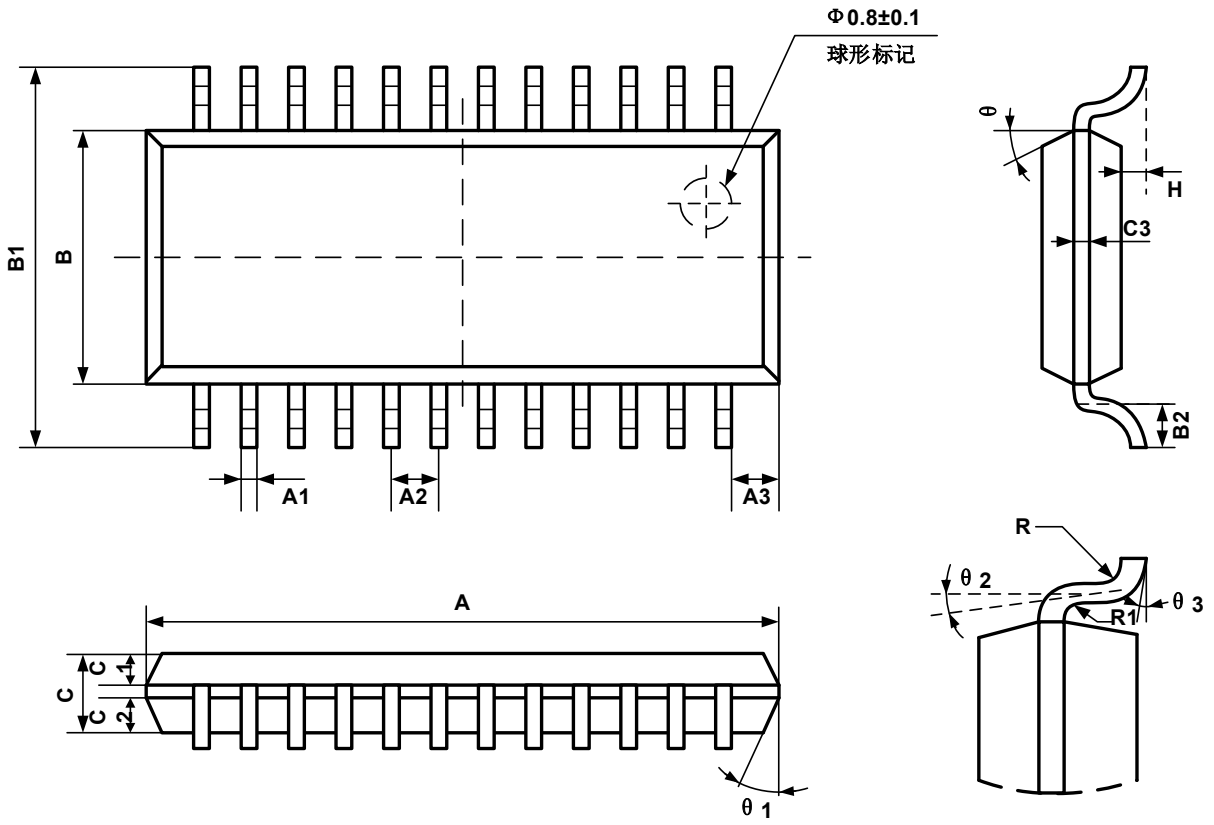
尺寸 / 标注	最小(mm)	正常(mm)	最大(mm)	尺寸 / 标注	最小(mm)	正常(mm)	最大(mm)
A	0.70	0.75	0.80	L	0.30	0.40	0.50
A1	0.00	-	0.05	L1	0.31	0.36	0.41
A3	0.203REF			L2	0.13	0.18	0.23
b	0.15	0.20	0.25				
D	3.90	4.00	4.10				
E	3.90	4.00	4.10				
D1	2.55	2.65	2.75				
E1	2.55	2.65	2.75				
e	0.40TYP						
K	0.20	-	-				

QFN32 (4x4, 0.4mm)封装尺寸



尺寸 / 标注	最小(mm)	正常(mm)	最大(mm)	尺寸 / 标注	最小(mm)	正常(mm)	最大(mm)
A	2.465	2.515	2.565	E1	7.374	7.450	7.574
A1	0.100	0.150	0.200	E2	7.424	7.500	7.624
A2	2.100	2.300	2.500	e	-	1.270	-
A3	-	0.274	-	L	0.764	0.864	0.964
b	0.356	0.406	0.456	L1	1.303	1.403	1.503
b1	0.366	0.426	0.486	R	-	0.200	-
c	-	0.254	-	R1	-	0.300	-
D1	17.750	17.950	18.150	θ	0.000	-	-
D2	17.800	18.000	18.200	θ1	0.000	-	10°
E	10.100	10.300	10.500	Z	-	0.745	-

SOP28 (1.27mm)封装尺寸



尺寸 / 标注	最小（mm）	最大（mm）	尺寸 / 标注	最小（mm）	最大（mm）
A	8.60	8.70	C3	0.203TYP	
A1	0.254TYP		H	0.10	0.25
A2	0.635TYP		θ	8°TYP4	
A3	0.705TYP		θ1	7°TYP4	
B	3.85	3.95	θ2	4°～ 12°	
B1	5.80	6.20	θ3	0°～ 8°	
B2	0.40	0.70	R	0.20TYP	
C	1.40	1.50	R1	0.20TYP	
C1	0.40	0.70			
C2	0.55	0.65			

SSOP24 （0.635mm）封装尺寸

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for aptchip ***manufacturer:***

Other Similar products are found below :

APT32F101H6M6 APT32F003F6P6 APT8L08SE APT32F101H6S6