APT32S003系列使用手册 V1.5

S003系列通用型32 位微处理控制器

简介

该使用手册面向应用开发人员，旨在给与 APT32S003 系列内核，存储及全部外围的完整信息。

相关文档

APT32S003S7P6 数据手册
APT32S003F8PT 数据手册

产品分类定义

Table 0-1 APT32S003系列产品分类定义

资源		003xxxx	003xxxT
异	TOUCH	-	1
同	UART	2	2
	SPI	1	
	HWD	1	
	ADC	16	
	PFlash	64K/32K	
	Package	TSSOP20	
	Pin	20	
	DFlash	2K	
	SRAM	4K	
	CRC	1	
	IWDT	1	
	WWDT	1	
	BT	2	
	CNTA	1	
	LPT	1	
	EPT	1	
	GPT	1	
	IIC	1	

历史版本说明

版本	修改日期	修改概要
V0.0	2020-11-04	初版
V1.0	2020-12-01	完善初版
V1.1	2020-12-07	修改串口资源
V1.2	2021-01-08	更新logo
V1.3	2021-02-01	修改TTL、FIN、SWD、GPTA、EPT部分错误
V1.4	2021-03-30	更正LVD错误
V1.5	2022-3-3	修改GPTA、EPT、GPIO_DSCR部分错误 英文初版后纠错。

1 系统存储空间

1.1 概述

本章节介绍了 APT32S003 系统存储空间。

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

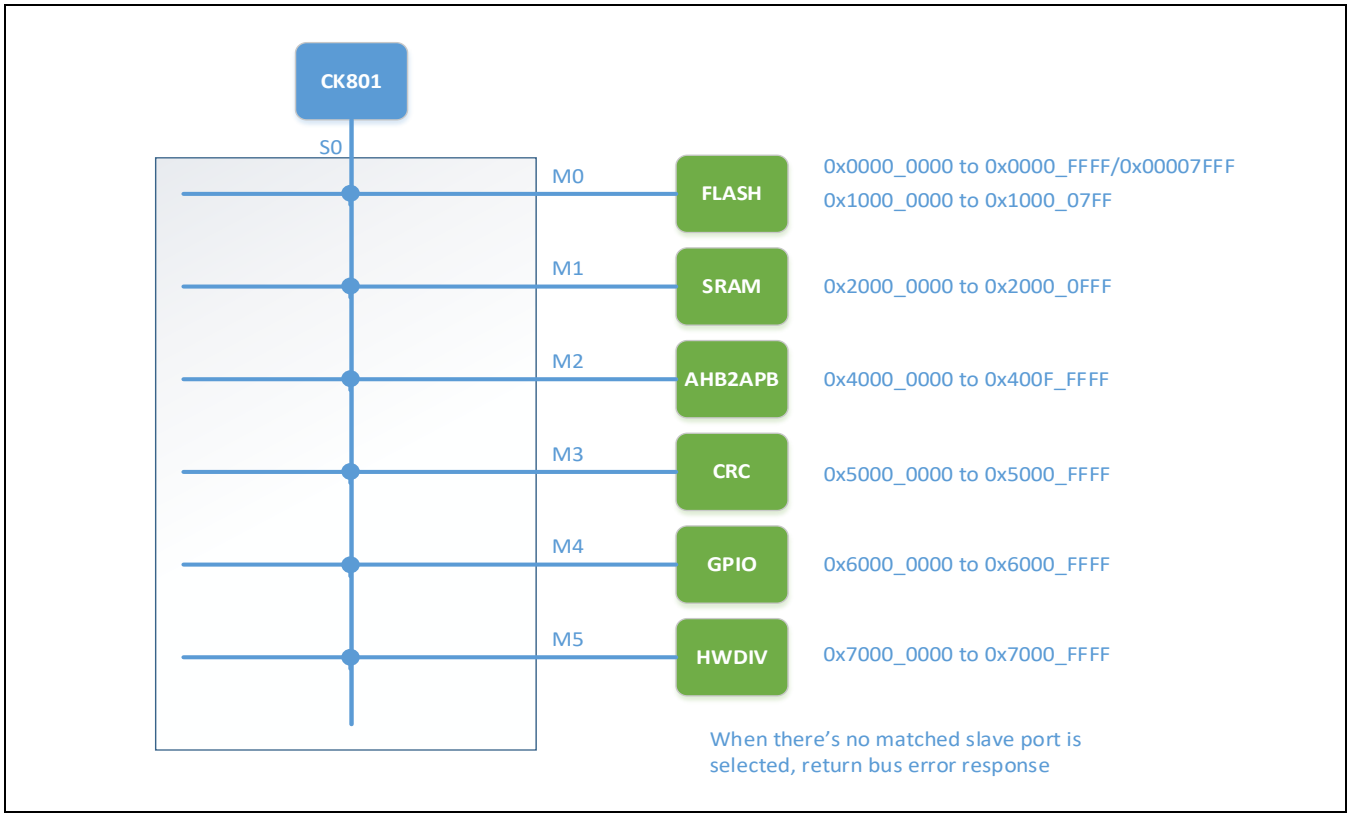


Figure 1-1 系统总线架构

1.2 默认存储地址表

Table 1-1 存储地址

Address	Memory
Reserved	Reserved
0xE000_0000 to 0xE00F_FFFF	CPU内部寄存器
Reserved	Reserved
0x7000_0000 to 0x7000_FFFF	硬件除法器
Reserved	Reserved
0x6000_0000 to 0x6000_FFFF	GPIO控制器
Reserved	Reserved
0x5000_0000 to 0x5000_FFFF	CRC控制器
Reserved Address Space	保留地址空间
0x4000_0000 to 0x400F_FFFF	特殊功能寄存器 (SFR)
Reserved	Reserved
0x2000_0000 to 0x2000_0FFF	SRAM
Reserved	Reserved
0x1000_0000 to 0x1000_07FF	数据闪存 (Data Flash)
Reserved	Reserved
0x0000_0000 to 0x0000_FFFF	程序闪存 (Program Flash)

1.3 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

1.3.1 CPU特殊功能寄存器表

Table 1-2 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

1.3.2 外围设备特殊功能寄存器表

Table 1-3 外围设备SFR表

Peripheral	Base Address	Function Description
-	0x7000_0000	保留 (Reserved)
GPIO	0x6000_2000	通用IO端口-B (GPIO B)
	0x6000_0000	通用IO端口-A (GPIO A)
CRC	0x5000_0000	CRC控制器
-	0x400B_0000	保留 (Reserved)
I2C	0x400A_0000	I2C串行接口 (I2C)
SPI	0x4009_0000	同步并行接口 (SPI)
UART	0x4008_2000	通用异步收发器2 (UART2)
	0x4008_1000	保留 (Reserved)
	0x4008_0000	通用异步收发器0 (UART0)
-	0x4007_0000	保留 (Reserved)
LP_TC	0x4006_2000	窗口型看门狗定时器 (WWDT)
	0x4006_1000	低功耗定时器 (LPT)
	0x4006_0000	保留 (Reserved)
TC	0x4005_9000	增强型可编程定时器/计数器 (EPT)
	0x4005_8000	保留 (Reserved)
	0x4005_7000	保留 (Reserved)

	0x4005_6000	保留 (Reserved)
	0x4005_5000	通用型可编程定时器/计数器 (GPT)
	0x4005_4000	保留 (Reserved)
	0x4005_3000	保留 (Reserved)
	0x4005_2000	基本定时器 (BT1)
	0x4005_1000	基本定时器 (BT0)
	0x4005_0000	计数器A (COUNTER A)
ADC	0x4003_0000	模数转换器 (ADC)
-	0x4002_0000	保留 (Reserved)
SYSTEM	0x4001_2000	事件触发控制器 (ETCB)
	0x4001_1000	系统控制器 (SYSCON)
	0x4001_0000	闪存控制器 (IFC)
	0x4000_0000	设备信息寄存器 (Device ID)

NOTE：如果芯片本身不具有某一外围器件，那它就不具备该外围的所有资源。具体请参看芯片的数据手册。

2 中断向量控制器(INTC)

2.1 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。

2.1.1 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

2.2 中断向量表

Table 2-1 System Interrupt Vectors

Number	Address	Vector	Interrupt Sources
32/0	0x0000_0080	CORET	CK801 CPU Core Timer
33/1	0x0000_0084	SYSCON	System controller interrupt
34/2	0x0000_0088	IFC	Program flash controller interrupt
35/3	0x0000_008C	ADC	ADC Interrupt
36/4	0x0000_0090	EPT	EPT Interrupt
37/5	0x0000_0094	-	Reserved
38/6	0x0000_0098	WWDT	Window Watchdog Interrupt
39/7	0x0000_009C	EXI0	External interrupt GROUP0, GROUP16
40/8	0x0000_00A0	EXI1	External interrupt GROUP1, GROUP17
41/9	0x0000_00A4	GPTA	GPTA Interrupt
42/10	0x0000_00A8	-	Reserved
43/11	0x0000_00AC	-	Reserved
44/12	0x0000_00B0	RTC	RTC interrupt
45/13	0x0000_00B4	UART0	UART 0 interrupt
46/14	0x0000_00B8	UART1	UART 1 interrupt
47/15	0x0000_00BC	UART2	UART 2 interrupt
48/16	0x0000_00C0	-	Reserved
49/17	0x0000_00C4	I2C	I2C interrupt
50/18	0x0000_00C8	-	Reserved
51/19	0x0000_00CC	SPI	SPI interrupt
52/20	0x0000_00D0	SIO	SIO Interrupt
53/21	0x0000_00D4	EXI2	External Interrupt GROUP2 ~ 3, GROUP18~19
54/22	0x0000_00D8	EXI3	External Interrupt GROUP4 ~ 9
55/23	0x0000_00DC	EXI4	External Interrupt GROUP10 ~ 15
56/24	0x0000_00E0	CNTA	COUNTER A interrupt
57/25	0x0000_00E4	TKEY	Touch Key interrupt
58/26	0x0000_00E8	LPT	LPT interrupt
59/27	0x0000_00EC	-	Reserved
60/28	0x0000_00F0	BT0	BT0 interrupt
61/29	0x0000_00F4	BT1	BT1 interrupt
62/30	0x0000_00F8	-	Reserved
63/31	0x0000_00FC	-	Reserved

中断向量号，是请求在异常表的位置编号。0~31用作处理器内部识别的向量；31号以后的向量号是留给软件的，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。例如“32/0”这样的表示，说明CORET作为外设的第0号向量，实际对应处理器的32号向量。

注：如果芯片本身不具有某一外围器件，那它就不具备该外围的所有资源。具体请参看芯片的数据手册。

2.3 工作原理

2.3.1 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。

2.3.2 中断优先级和中断抢占

中断的优先级可以通过VIC_IPR0 ~7这8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。

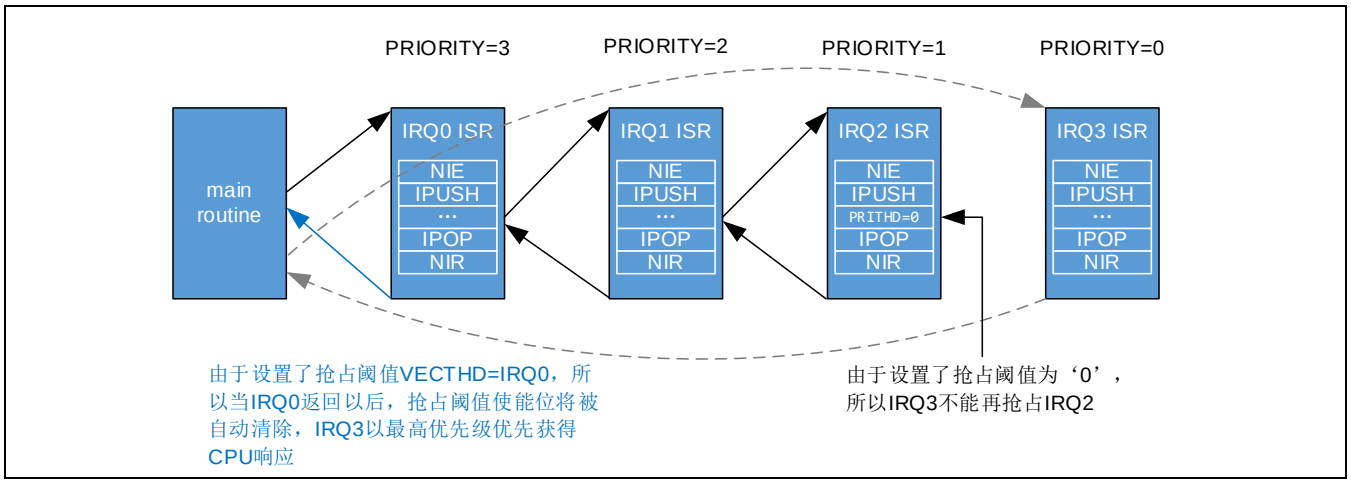


Figure 2-1 中断嵌套优先级示意图

2.3.3 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。

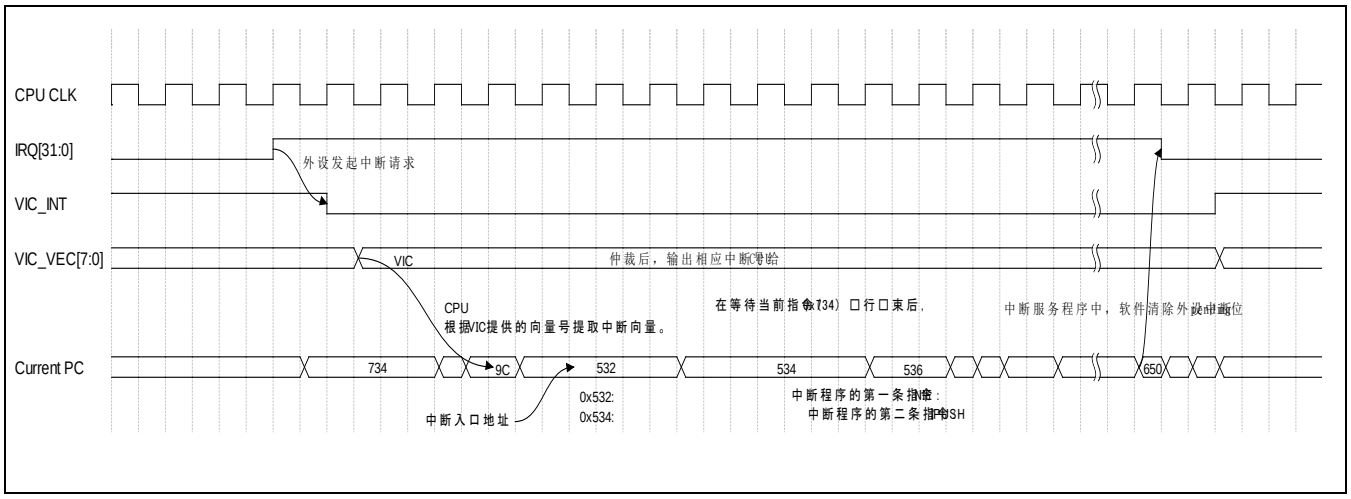


Figure 2-2 中断响应过程

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。

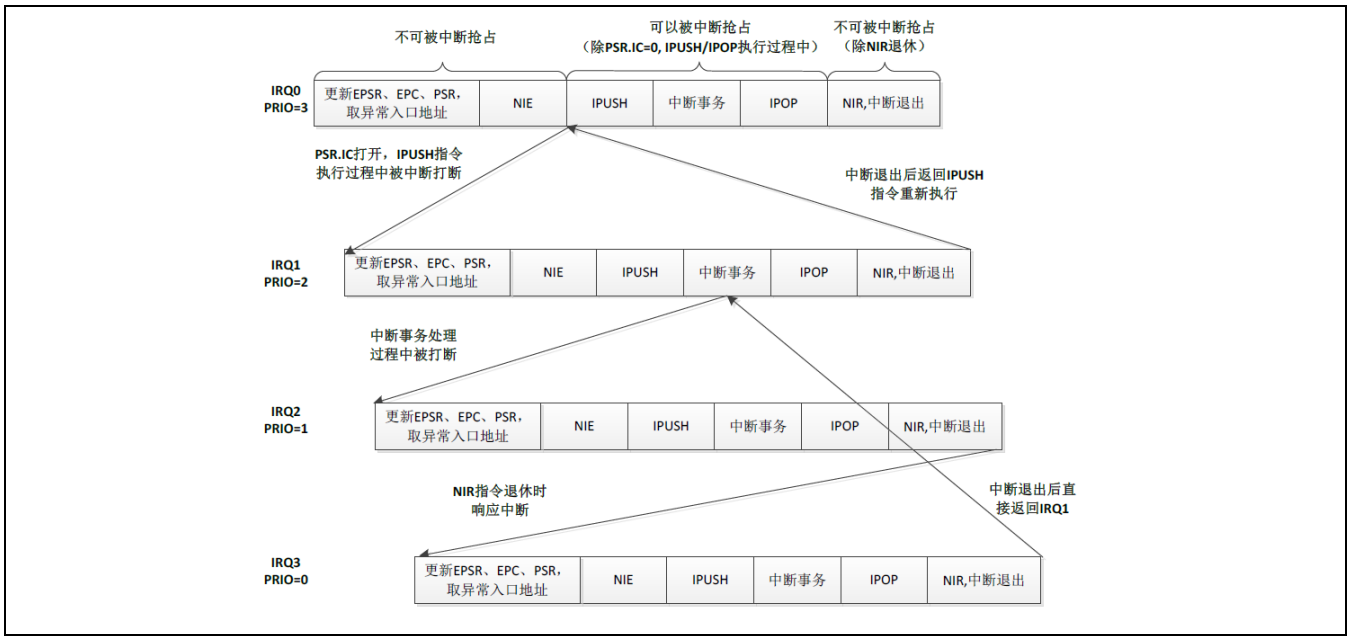


Figure 2-3 中断嵌套条件示例

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。

2.3.4 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。

.

2.3.5 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。

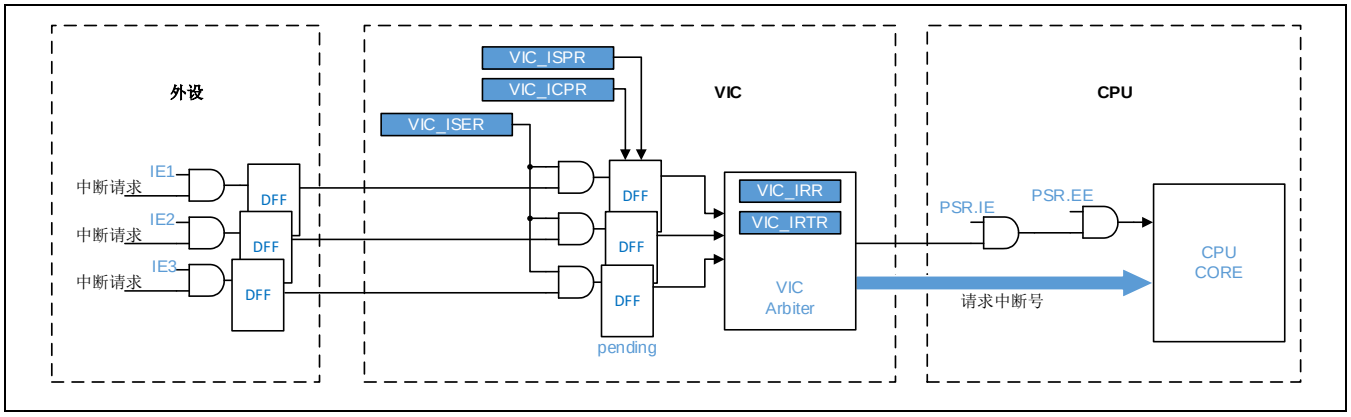


Figure 2-4 中断配置结构示意图

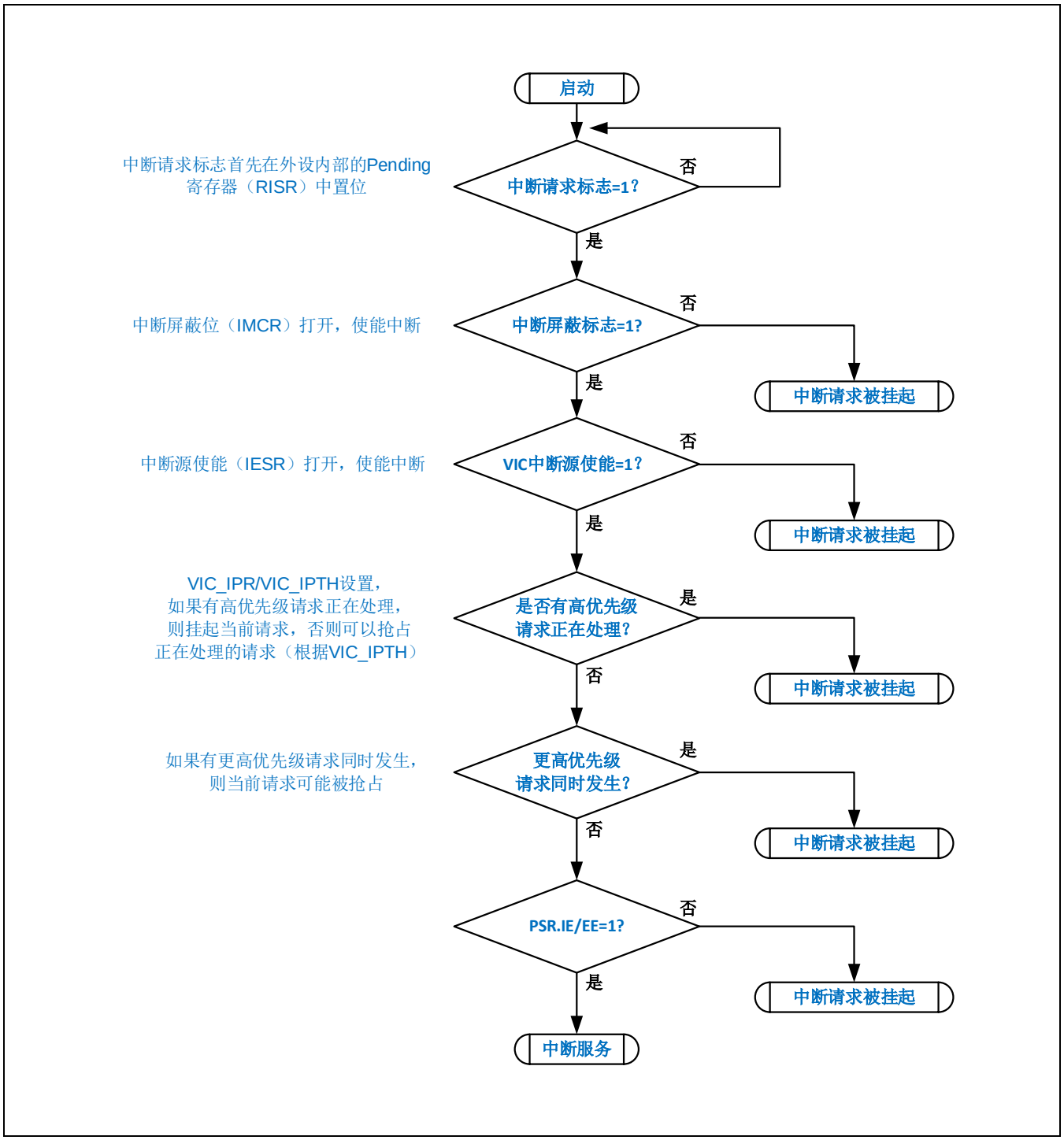


Figure 2-5 中断请求处理流程

寄存器说明

2.3.6 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:

2.3.6.1 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

2.3.6.2 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

2.3.6.3 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

2.3.6.4 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

2.3.6.5 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

2.3.6.6 VIC_ICPR (中断等待设置寄存器)

- 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

2.3.6.7 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

2.3.6.8 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(x=0..3)	[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_x： 中断号 x的优先级设置	0x0

2.3.6.9 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(x=4..7)	[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_x： 中断号x的优先级设置	0x0

2.3.6.10 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(x=8..11)	[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_x: 中断号x的优先级设置	0x0

2.3.6.11 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(x=12…15)	[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_x: 中断号x的优先级设置	0x0

2.3.6.12 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
PR_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(x=16..19)	[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_x: 中断号x的优先级设置	0x0

2.3.6.13 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(x=20..23)	[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_x: 中断号x的优先级设置	0x0

2.3.6.14 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(x=24..27)	[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_x: 中断号x的优先级设置	0x0

2.3.6.15 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(x=28..31)	[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_x: 中断号x的优先级设置	0x0

2.3.6.16 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	[20:12]	RW	指示当前等待的最高优先级中断向量号	0x0

2.3.6.17 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	RW	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

3 系统定时器 (CORET)

3.1 概述

系统定时器是 CK801 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。

3.2 功能描述

3.2.1 模块框图

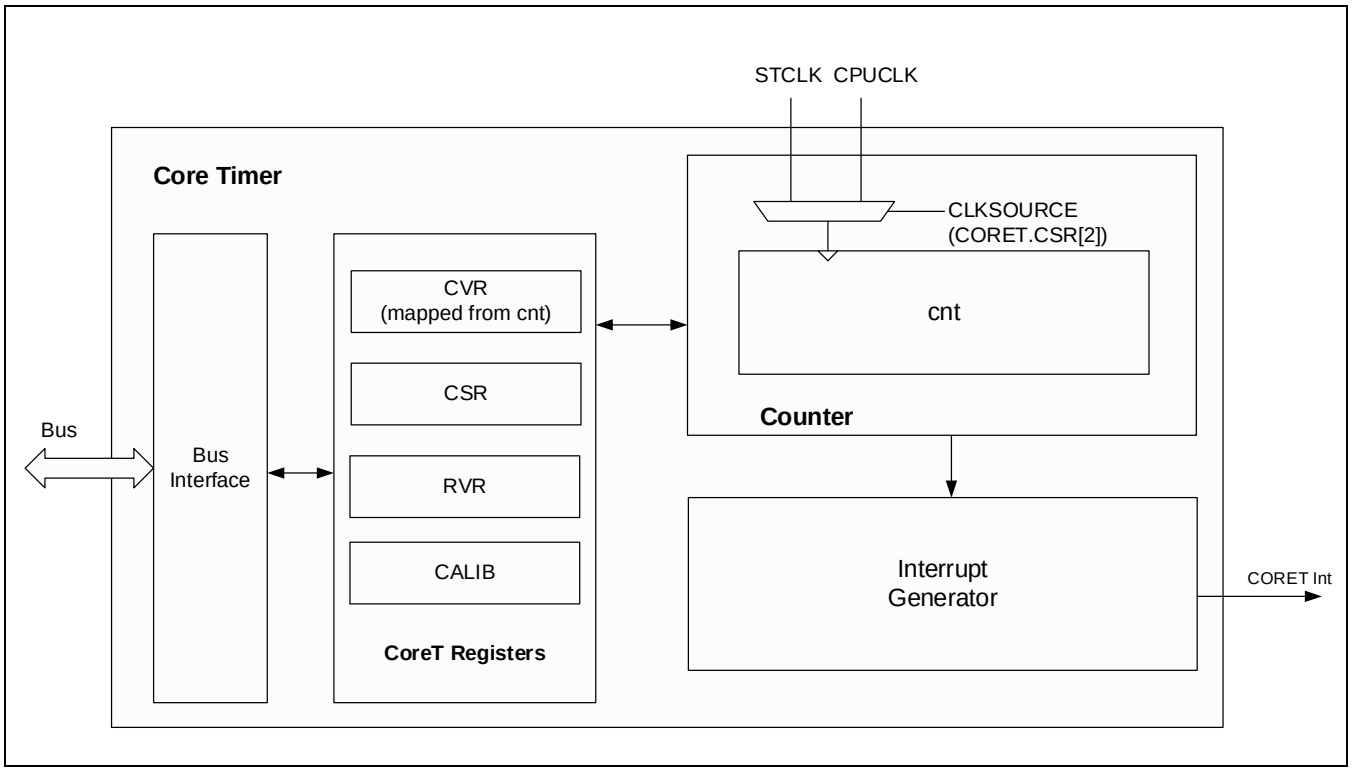


Figure 3-1 CORET模块框图

3.2.2 功能说明

3.2.2.1 定时器的时钟源

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 寄存器的第 2 位 CLKSOURCE 来实现。
时钟的使能/禁止和各种配置请参考 SYSCON 章节。

3.2.2.2 定时器的工作原理

CORET定时器是CK801 CPU提供的一个简单易用的24位循环递减的计数器，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。

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。

CORET的计数器不具有复位清零功能。在每次复位后，需要通过软件进行初始化。

3.3 寄存器说明

3.3.1 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

3.3.1.1 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	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	系统定时器时钟(STCLK)的时钟源选择： 0：时钟源为(CORECLK/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

NOTE: CORECLK 是 SYSTEM CLOCK 经过分频后，供 CPU 工作使用的 clock，和 HCLK 同频。

3.3.1.2 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

3.3.1.3 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

4 CRC

4.1 概述

本章节描述的是用来纠错的CRC引擎。该模块支持常用的CRC标准。

4.1.1 特性

- 支持字节，半字节，全字节操作
  - 字节操作单时钟周期
  - 半字操作需要2个时钟周期
  - 全字操作需要4个时钟周期
- 可编程的多项式
  - CRC-CCITT: x16 + x12 + x5 + 1
  - CRC-16: x16 + x15 + x2 + 1
  - CRC-32: x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1
- 可编程的CRC种子值 (预置值，初值)
- 可编程的数据反转 (首先处理LSB或者MSB)，输入值和输出值可计算补码
- 工作时钟为HCLK

4.2 功能描述

4.2.1 模块框图

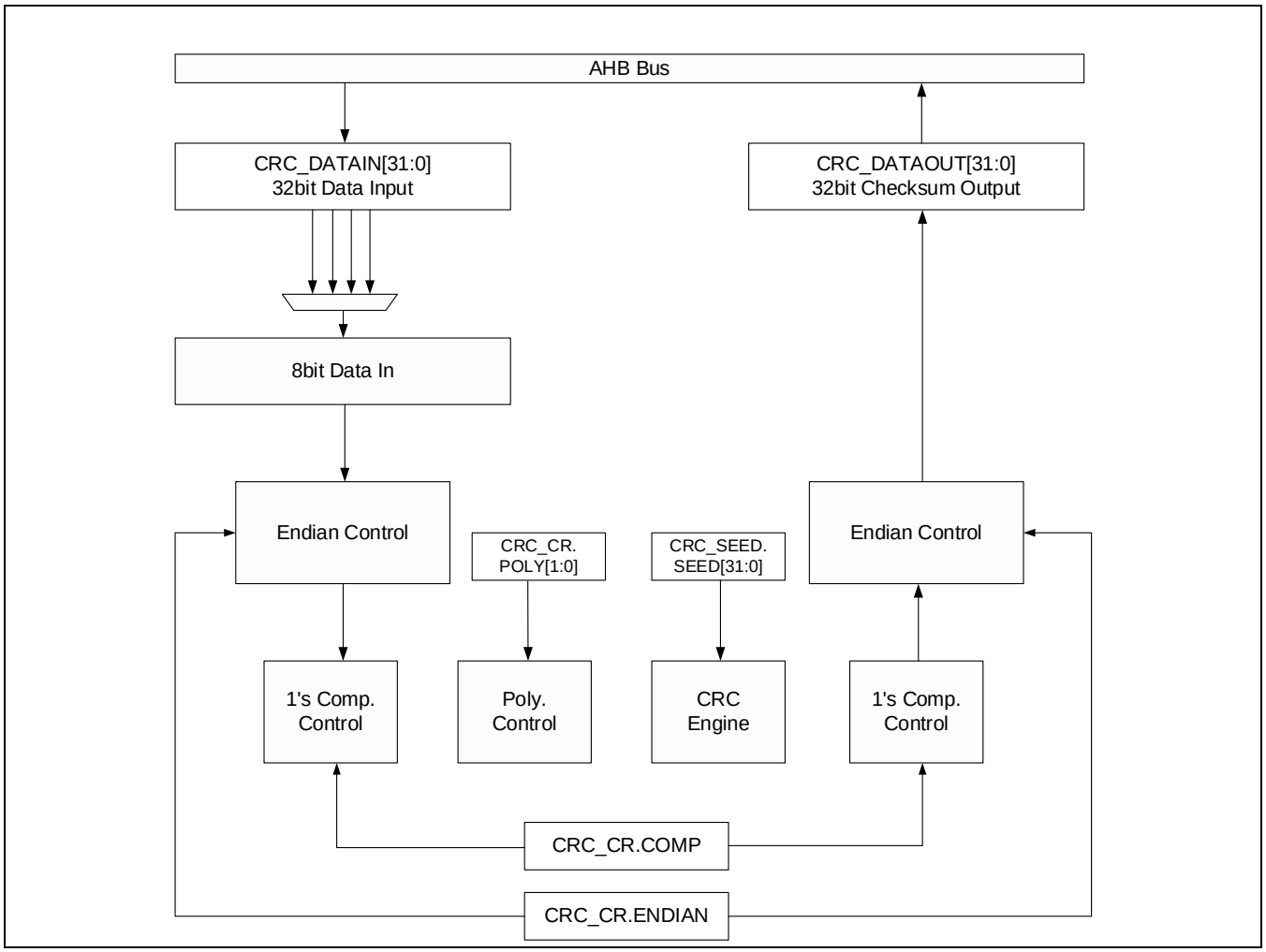


Figure 4-1 CRC模块框图

4.2.2 功能说明

CRC 模块的使用非常简单，只需要将数据写入 CRC_SEED 和 CRC_DATAIN 寄存器中，并且在下一个指令读取 CRC_DATAOUT 寄存器即可。CRC_CR 寄存器用来配置 CRC 引擎。在 CRC 模块工作前，必须使能 CRC_CEDR 寄存器中的时钟使能 CKEN 位。

寄存器说明

4.2.3 寄存器表

Base Address: 0x5000_0000

Name	Offset	Description	R/W	Reset State
RSVD	0x0000	保留	R/W	-
CRC_CEDR	0x0004	时钟使能/禁止寄存器	R/W	0x00000000
CRC_SRR	0x0008	软件复位寄存器	R/W	0x00000000
CRC_CR	0x000C	控制寄存器	R/W	0x00000000
CRC_SEED	0x0010	种子值寄存器	R/W	0x00000000
CRC_DATAIN	0x0014	输入数据寄存器	R/W	0x00000000
CRC_DATAOUT	0x0018	输出数据寄存器	R/W	0x00000000

4.2.3.1 CRC_CEDR (时钟使能/禁止寄存器)

- 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																															CKEN
0	0	0	0	0	0	0	0	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	Reset Value
CKEN	[0]	RW	CRC 引擎使能： 0：禁止 1：使能	0

4.2.3.2 CRC_SRR (软件复位寄存器)

- 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
SWRST	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	R	R

Name	Bit	Type	Description	Reset Value
SWRST	[31]	W	软件复位 写1会复位该模块	0x0

4.2.3.3 CRC_CR (控制寄存器)

- Address = Base Address + 0x000C

31	30	29	28	27	26	25	24	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																										POLY		REFOUT	REFIN	XOROUT	XORIN
0	0	0	0	0	0	0	0	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	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
XORIN	[0]	R/W	CRC输入数据的异或控制 0: 无异或 1: 使能异或	0x0
XOROUT	[1]	R/W	CRC输出数据的异或控制 0: 无异或 1: 使能异或	0x0
REFIN	[2]	R/W	CRC输入数据的按位反转控制 0: 无反转 1: 使能反转	0x0
REFOUT	[3]	R/W	CRC输出数据的按位反转控制 0: 无反转 1: 使能反转	0x0
POLY	[5:4]	R/W	多项式控制 0x: CRC-CCITT 10: CRC-16 11: CRC-32	0x0

4.2.3.4 CRC_SEED (种子值寄存器)

- 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
SEED																															
0	0	0	0	0	0	0	0	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
SEED	[31:0]	R/W	对该寄存器的写操作会将该种子值载入CRC_DATAOUT寄存器中作为预置值(初值)。 SEED值必须在每次CRC计算前都操作一次。	0x0

4.2.3.5 CRC_DATAIN (输入数据寄存器)

- 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
DATAIN																															
0	0	0	0	0	0	0	0	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
DATAIN	[31:0]	W	用来计算CRC的输入数据。 支持字节，半字，全字三种格式。	0x0

4.2.3.6 CRC_DATAOUT (输出数据寄存器)

- Address = Base Address + 0x0018

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAOUT																															
0	0	0	0	0	0	0	0	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
DATAOUT	[31:0]	R	该寄存器保存CRC计算结果。	0x0

5 闪存控制器(IFC)

5.1 概述

本章节描述用来控制内部程序存储的闪存控制器。APT32S003 系列片上带有 64K/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。APT32S003 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。

5.1.1 主要特性

- 程序闪存(PROM)大小: 64K/32K Bytes
- 数据闪存(DROM)大小: 2K Bytes
- 编程支持ISP模式和专用的工具模式
- 页大小: 256 Bytes(PROM), 64 Bytes(DROM)
- 可擦除单元: 页
- 可靠性: PROM和DROM都为100,000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

5.2 功能描述

5.2.1 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：

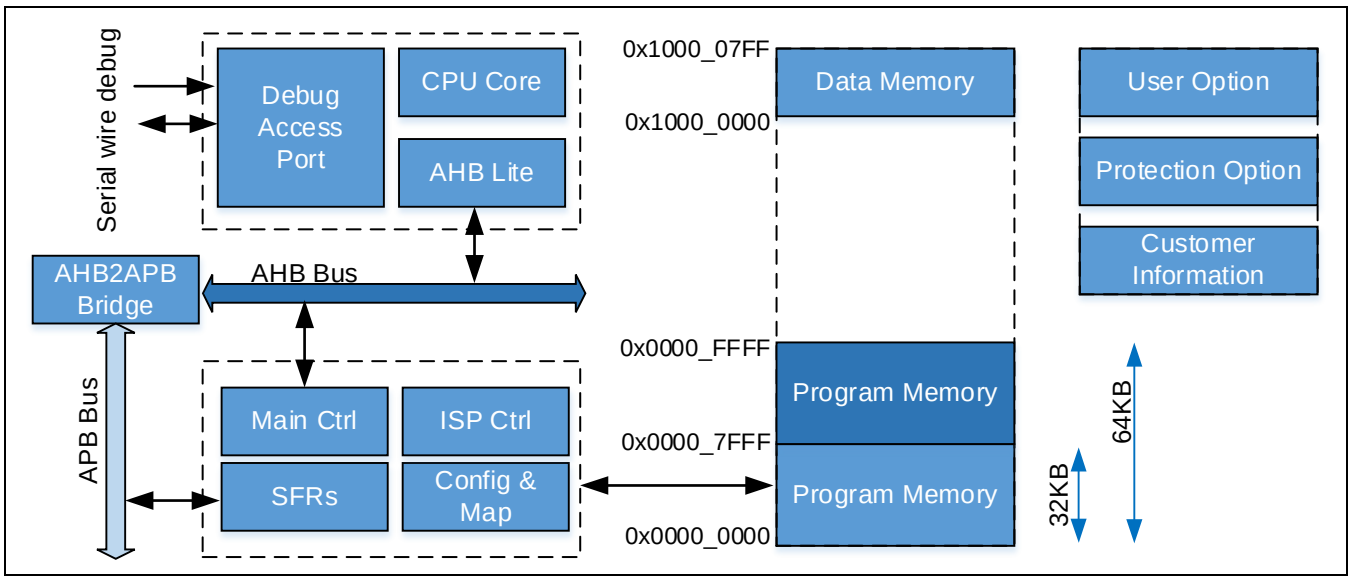


Figure 6-1 IFC模块框图

5.2.2 模块结构

APT32S003 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 256/128 个页空间，每页有 256 字节。最小的擦除和烧写单元为页空间，用户只可以对整个页空间进行擦除或者烧写，不能擦除或者烧写某个指定的字节(Word)。

Table 6-1 闪存地址映射

区域	页名称	大小	起始地址	结束地址
PROM Within 32KB	Page 0	256B	0x0000_0000	0x0000_00FF
	Page 1	256B	0x0000_0100	0x0000_01FF
	Page 2	256B	0x0000_0200	0x0000_02FF
	Page 3	256B	0x0000_0300	0x0000_03FF
	Page 4	256B	0x0000_0400	0x0000_04FF
	Page 5	256B	0x0000_0500	0x0000_05FF
	Page 6	256B	0x0000_0600	0x0000_06FF
	Page 7	256B	0x0000_0700	0x0000_07FF
	:	:	:	:
	Page 126	256B	0x0000_7E00	0x0000_7EFF

	Page 127	256B	0x0000_7F00	0x0000_7FFF
PROM Within 64KB	Page 128	256B	0x0000_8000	0x0000_80FF
	Page 129	256B	0x0000_8100	0x0000_81FF
	Page 130	256B	0x0000_8200	0x0000_82FF
	Page 131	256B	0x0000_8300	0x0000_83FF
	:	:	:	:
	Page 254	256B	0x0000_FE00	0x0000_FEFF
	Page 255	256B	0x0000_FF00	0x0000_FFFF
DROM	Page 0	64B	0x1000_0000	0x1000_003F
	Page 1	64B	0x1000_0040	0x1000_007F
	Page 2	64B	0x1000_0080	0x1000_00BF
	Page 3	64B	0x1000_00C0	0x1000_00FF
	:	:	:	:
	Page 30	64B	0x1000_0780	0x1000_07BF
	Page 31	64B	0x1000_07C0	0x1000_07FF

闪存结构如下图所示：

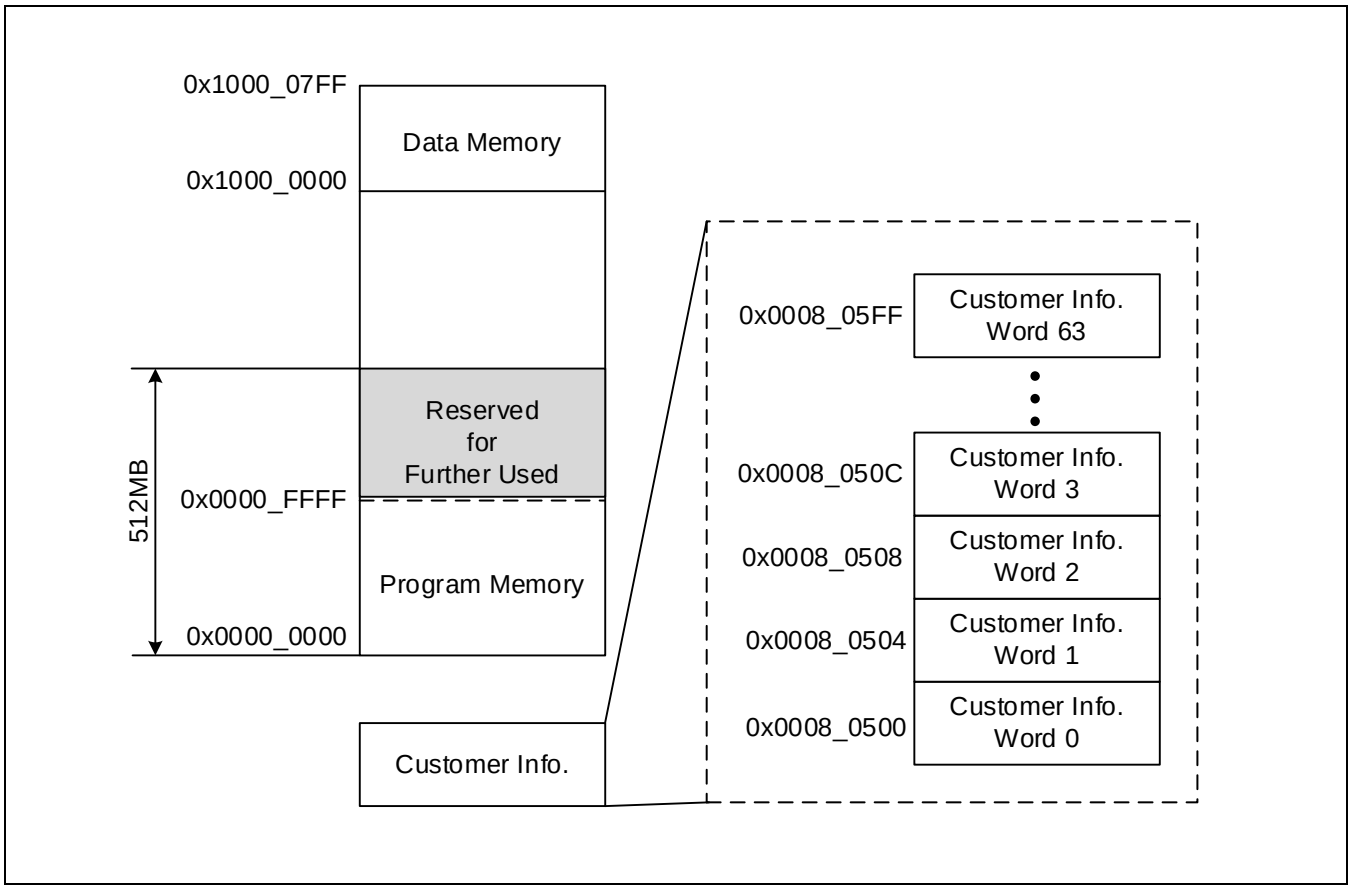


Figure 6-2 闪存地址空间结构

5.2.3 数据闪存

APT32S003 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 64 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。

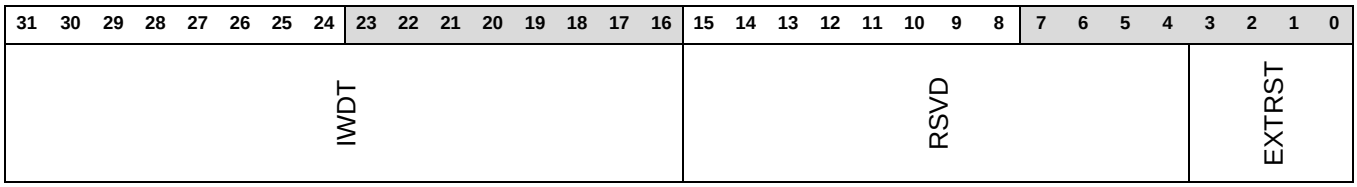
5.2.4 自定义选项 (User Option, 保护选项，SWD调试接口重映射，客户信息区域，UID区域)

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，重新映射 SWD 调试接口，和给客户存储一些自定义的信息。

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。

5.2.4.1 User Option

User Option 用来配置各种不同应用所需的功能，该功能需要专用的烧录器来实现。



名称	位	描述
EXTRST	[3:0]	外部复位管脚功能 **EXTRST[3:0]** / **功能** 0x5 / PA0.2为外部复位管脚，IO功能被禁用 其它值 / PA0.2禁用外部复位功能，当作IO使用
IWDT	[31:16]	0x55AA: 禁用系统看门狗 IWDT 其它值: 使能系统看门狗 IWDT

Table 6-2 程序代码 0x0000_0100 User Option功能映射

5.2.4.2 保护选项 (Protection)

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)

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 ~ 0xFFF区域内，然后选择IFC_4K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0xEE (意为Error).

- 调试接口(SWD)保护

  这个保护功能用来使能或禁止调试接口(SWD)的访问。在系统开发阶段，SWD 可以让开发者方便的查询系统状态并且调试芯片的工作。但是当代码开发完成后，如果不禁用 SWD 的话，那么程序代码仍旧就可以通过 SWD 读取出来。

- 代码加密保护

  这个保护功能用来加密 FLASH 中的程序代码。一旦使能该功能，在将程序代码写入 FLASH 时，闪存控制器会使用特定的算法和秘钥进行加密，而只有在 CPU 读取时，闪存控制器才会进行解密，并且允许 CPU 读取；在其它非 CPU 读取时，比如烧片机读取，或者非正常渠道的破解读取时，读到的代码都为加密代码。

保护选项建议通过烧片机进行使能。同时，保护功能在程序代码的 0x0000_0104 地址也有一个映射，在程序代码的 0x0000_0104 地址空间写上相应的值，会使能该保护功能。

注意：在一般的程序调试过程中，建议将 0x0000_0104 地址的值保持为 0，在最后量产阶段再将需要的配置值写入该地址中，否则一旦使能了 SWDP 功能，SWD 仿真器将无法连接。

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

名称	位	描述
RDP	[31:24]	读保护功能控制 0x1A: 使能 其它值: 禁止
SWDP	[15:8]	调试接口保护功能控制 0x5A: 使能 其它值: 禁止
ENCRYPT	[7:0]	代码加密保护功能控制 0x9A: 使能 其它值: 禁止

Table 6-3 程序代码 0x0000_0104 保护选项映射

注：HDP 硬件保护功能在该地址中无映射。

5.2.4.3 客户信息区 (Customer Information)

客户信息区域由 64 个字(256 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。该区域必须通过外部烧录工具进行烧写，读取可以直接通过访问该区域的地址(0x0008_0500 ~ 0x0008_05FF)直接读取。

5.2.4.4 UID (Unique ID)

UID 区域由 3 个字(12 字节)组成，制造工厂在生产时写入。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。该区域只能通过 SYSCON 模块中相应的镜像寄存器 UID0~UID2 进行读取。UID 为该芯片的标识，每个芯片有单独唯一的 UID。

5.2.4.5 SWD调试接口重映射

调试接口重映射功能用来重映射 IO 上的 SWD 功能，可以将 SWD 功能改到其它 IO 上。建议使用专用烧片机进行重映射。同时 IFC_CMR 寄存器的 SWD Remap 功能，也可以用来重映射，详细流程请参考程序库或者应用文档。

地址	配置	SWD 重映射端口
0x0008_01C0	0x0000_0055	PA0_06/PA0_07
	0x0000_00AA	PA0_00/PA0_01
	Others	PA0_05/PA0_12

注意：在 SWD 对应 GPIO 的 AF 功能设置(CONHR, CONLR 寄存器)里，请勿将 SWD 的 AF 功能修改成其它功能，否则将会造成芯片无法连接调试器。

5.2.4.6 自定义选项的设置方法

自定义选项的设置方法有多种，各种选项对应的设置方法不同，具体方法可参见下表。

	烧片机	程序代码里特殊地址	ISP 功能(IFC_CMR)
User Option	√	√ (0x0000_0100)	√
保护选项	√	√ (0x0000_0104)	√
客户信息区	√	X	X
UID	√	X	X
SWD 接口重映射	√	X	√

Table 6-4 自定义选项的设置方法

5.2.5 读操作

闪存控制器支持最大 16MHz 系统频率下的 0-wait 读取。当频率超过 16MHz 时，CPU 读取闪存时需要增加额外的等待周期，请参考 IFC_MR 寄存器。不同 CPU 频率下，WAIT 和 SPEED 的值参考值如下表。

	WAIT	SPEED
24MHz < SYSCLK ≤ 48MHz	2	1
16MHz < SYSCLK ≤ 24MHz	1	1
SYSCLK ≤ 16MHz	0	0

5.2.6 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)

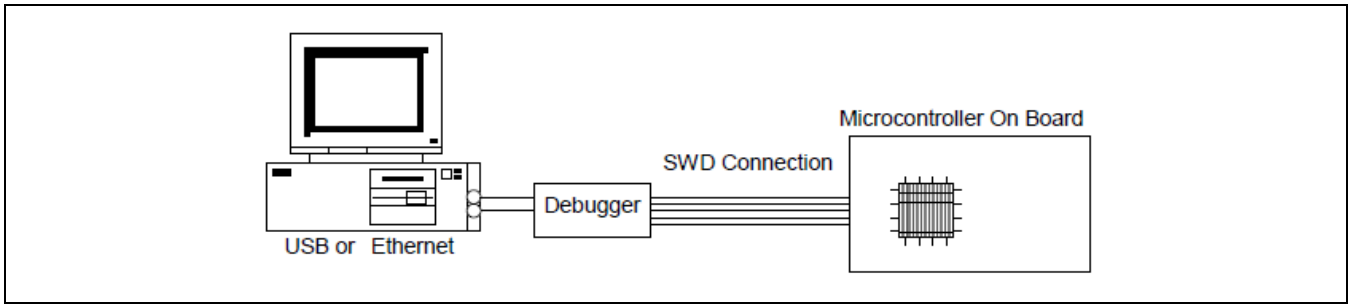


Figure 6-3 通过调试接口的烧写

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 硬件烧录模式需要使用 5 根线作为烧写信号。烧写所需的信号如下表所示。

信号	管脚名称	I/O	描述
VDD	VDD	P	芯片电源 (建议在VDD和VSS之间接入0.1uF的去耦电容)
VSS	VSS	G	芯片地
RESET	F_RSTB	I	芯片复位管脚
SDAT	F_SDAT	I/O	串行双向数据管脚
SCLK	F_SCL	I	串行时钟输入管脚

Table 6-5 闪存烧写信号

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。

5.2.7 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。

5.2.7.1 页擦除操作

每页闪存中有 256 (PROM) / 64 (DROM) 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
```

5.2.7.2 写闪存操作

由于本产品闪存的操作对象为页，而不是字节或者字，所以写闪存跟擦除闪存一样，都要对整个 256 (PROM) / 64 (DROM) 字节的页进行操作。本闪存包含一个页缓存空间，在写操作中，需要先将整个页的数据先写入到页缓存空间中，再执行写操作的命令，将整个页缓存空间中的数据一起写入闪存中。

另外，基于该产品的闪存特性，在擦除闪存之前，需要有一个预编程操作，以防止闪存单元的“过擦除”，影响闪存寿命。

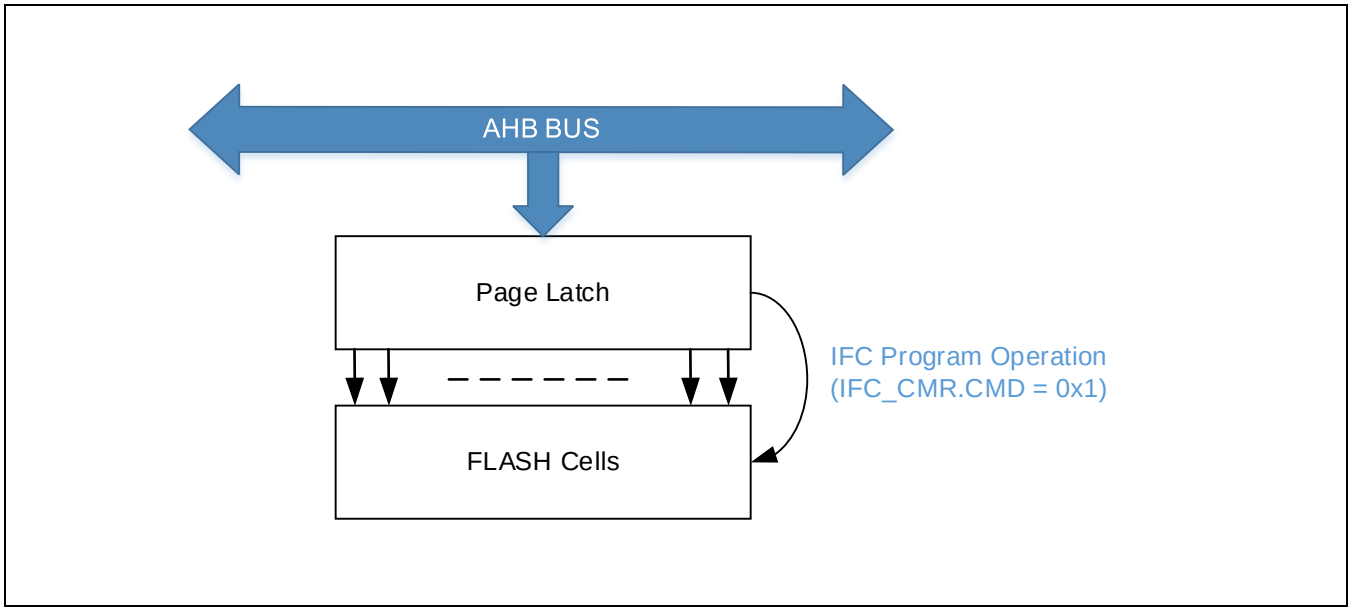


Figure 6-4 将页缓存数据写入闪存中

具体步骤如下：

1. 清除页缓存空间(IFC_CMR 寄存器中的 CMD = 0x7)。
2. 将需要写入的数据填入页缓存 (页缓存可通过总线直接写入，类似于 SRAM 内存空间)。
3. 预编程设定(IFC_CMR 寄存器中的 CMD = 0x6)，设置下一步的编程为预编程。
4. 执行写操作(IFC_CMR 寄存器中的 CMD = 0x1)，进行预编程。
5. 执行页擦除操作(IFC_CMR 寄存器中的 CMD = 0x2)，擦除整个页数据。
6. 执行写操作(IFC_CMR 寄存器中的 CMD = 0x1)，将页缓存的数据写入闪存中。

在每次执行 IFC 操作的命令前，用户必须将秘钥 0x5A5A_5A5A 写入 IFC_KEY 寄存器，之后将 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：

```
// Step1. Clear Page Latch
    CSP_IFC_SET_KR(IFC, USER_KEY);          // Write Key
    CSP_IFC_SET_CMR(IFC, PAGE_LAT_CLR);     // Clear Page Latch
    CSP_IFC_SET_AR(IFC, PAGE_ADDR);         // Program address
    CSP_IFC_SET_CR(IFC, START);             // Start Program
    while ( CSP_IFC_GET_CR(IFC) != 0x0 );   // Wait for operation done
```

```
// Step2. Write Page Latch
    for(i=0;i<page_size;i++){
      *(volatile unsigned int *)(PAGE_ADDR+4*i) = buffer[i];
    }

// Step3. Set Pre-Program Option
    CSP_IFC_SET_KR(IFC, USER_KEY);          // Write Key
    CSP_IFC_SET_CMR(IFC, PRE_PGM);          // Pre-Program
    CSP_IFC_SET_AR(IFC, PAGE_ADDR);         // Program address
    CSP_IFC_SET_CR(IFC, START);             // Start Program
    while ( CSP_IFC_GET_CR(IFC) != 0x0 );   // Wait for operation done

// Step4. Execute Pre-Program Operation
    CSP_IFC_SET_KR(IFC, USER_KEY);          // Write Key
    CSP_IFC_SET_CMR(IFC, PROGRAM);          // Program
    CSP_IFC_SET_AR(IFC, PAGE_ADDR);         // Program address
    CSP_IFC_SET_CR(IFC, START);             // Start Program
    while ( CSP_IFC_GET_CR(IFC) != 0x0 );   // Wait for operation done

// Step5. Page Erase
    CSP_IFC_SET_KR(IFC, USER_KEY);          // Write Key
    CSP_IFC_SET_CMR(IFC, PAGE_ERASE);       // Page Erase
    CSP_IFC_SET_AR(IFC, PAGE_ADDR);         // Erase address
    CSP_IFC_SET_CR(IFC, START);             // Start Erase
    while ( CSP_IFC_GET_CR(IFC) != 0x0 );   // Wait for operation done

// Step6. Execute Program Operation
    CSP_IFC_SET_KR(IFC, USER_KEY);          // Write Key
    CSP_IFC_SET_CMR(IFC, PROGRAM);          // Program
    CSP_IFC_SET_AR(IFC, PAGE_ADDR);         // Program address
    CSP_IFC_SET_CR(IFC, START);             // Start Program
    while ( CSP_IFC_GET_CR(IFC) != 0x0 );   // Wait for operation done
```

需要注意的是，由于本产品只能对整页数据进行操作，如果只需要写 1 个字(Word)，那么程序必须将该页中所有数据读出来并且写回页缓存区域中，并且替换该页中需要操作的那个字(Word)的数据，最后再将含有新数据的页缓存全部写入闪存中。

5.2.7.3 片擦除操作

片擦除操作会擦除整个闪存的程序存储，但不会擦除数据存储区域和自定义选项的区域。片擦除操作只能在用户特权模式下才能执行。在片擦除中，不需要指令 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 0x4(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
```

5.2.7.4 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 0x5(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
```

5.2.7.5 烧写自定义选项操作

共有 6 种自定义的选项支持通过 ISP 操作(HDP, RDP, SWDP, SWD_REMAP, ENCRYPT, USER_OPTION)。写操作的步骤跟普通写闪存操作一样，不同的是，在最后第 6 步写入闪存的时候，IFC_CMR 里的指令 CMD[20:16] / CMD[3:0]写为对应指令，以及在每一步配置 IFC_CMR 时，HMODE[1:0]都需要写为 0x1(用户特权模式)。烧写自定义选项时，不需要在每个步骤中设置地址寄存器 IFC_AR，只需要在第一部中将地址寄存器设置为 0 即可，系统会自动控制烧写的地址。由于在 SYSCON 中有独立看门狗电路的控制位，所以这里没有控制 iWDT 的 ISP 操作，只有外部烧录工具支持单独修改 iWDT 设置。

示例 (烧写 USER_OPTION)：

```
    unsigned int buffer[0] = USER_OPTION_VALUE;     // Load USER_OPTION value to the
                                                    // lowest address of page latch
// Step1. Clear Page Latch
    CSP_IFC_SET_KR(IFC, USER_KEY);                  // Write Key
    CSP_IFC_SET_CMR(IFC, PAGE_LAT_CLR|HIDM1);       // Clear Page Latch
    CSP_IFC_SET_AR(IFC, 0x0);                       // Program address
    CSP_IFC_SET_CR(IFC, START);                     // Start Program
    while ( CSP_IFC_GET_CR(IFC) != 0x0 );           // Wait for operation done

// Step2. Write Page Latch
    for(i=0;i<page_size;i++){
```

```
      *(volatile unsigned int *)(PAGE_ADDR+4*i) = buffer[i];
    }

// Step3. Set Pre-Program Option
    CSP_IFC_SET_KR(IFC, USER_KEY);                  // Write Key
    CSP_IFC_SET_CMR(IFC, PRE_PGM|HIDM1);            // Pre-Program
    CSP_IFC_SET_CR(IFC, START);                     // Start Program
    while ( CSP_IFC_GET_CR(IFC) != 0x0 );           // Wait for operation done

// Step4. Execute Pre-Program Operation
    CSP_IFC_SET_KR(IFC, USER_KEY);                  // Write Key
    CSP_IFC_SET_CMR(IFC, PROGRAM|HIDM1);            // Program
    CSP_IFC_SET_CR(IFC, START);                     // Start Program
    while ( CSP_IFC_GET_CR(IFC) != 0x0 );           // Wait for operation done

// Step5. Page Erase
    CSP_IFC_SET_KR(IFC, USER_KEY);                  // Write Key
    CSP_IFC_SET_CMR(IFC, IF0_ERASE|HIDM1);          // Page Erase
    CSP_IFC_SET_CR(IFC, START);                     // Start Erase
    while ( CSP_IFC_GET_CR(IFC) != 0x0 );           // Wait for operation done

// Step6. Execute Program Operation
    CSP_IFC_SET_KR(IFC, USER_KEY);                  // Write Key
    CSP_IFC_SET_CMR(IFC, USER_OPTION|HIDM1);        // Program
    CSP_IFC_SET_CR(IFC, START);                     // Start Program
    while ( CSP_IFC_GET_CR(IFC) != 0x0 );           // Wait for operation done
```

5.2.8 闪存控制器的中断

闪存操作有 7 个中断源，如下所示。

中断	描述
ERS_END	擦除指令执行完成中断
PGM_END	写操作指令执行完成中断
PEP_END	预编程指令执行完成中断 (该中断在设置了预编程选项后的写闪存操作完成时产生)
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 6-6 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 IMCR 设置的影响。如果 IMCR 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 ICR 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。

5.2.9 闪存控制流程图

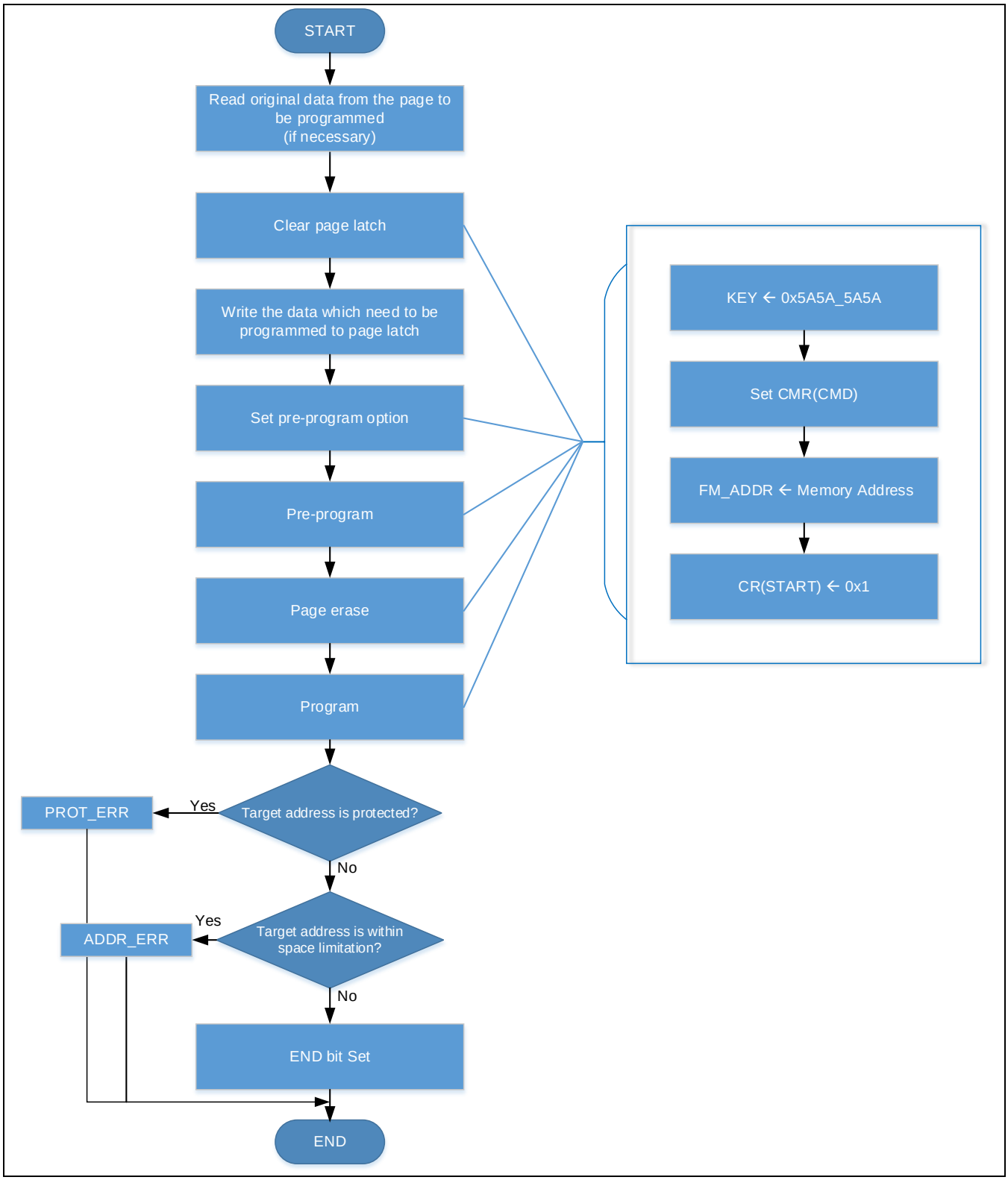


Figure 6-5 写操作流程图

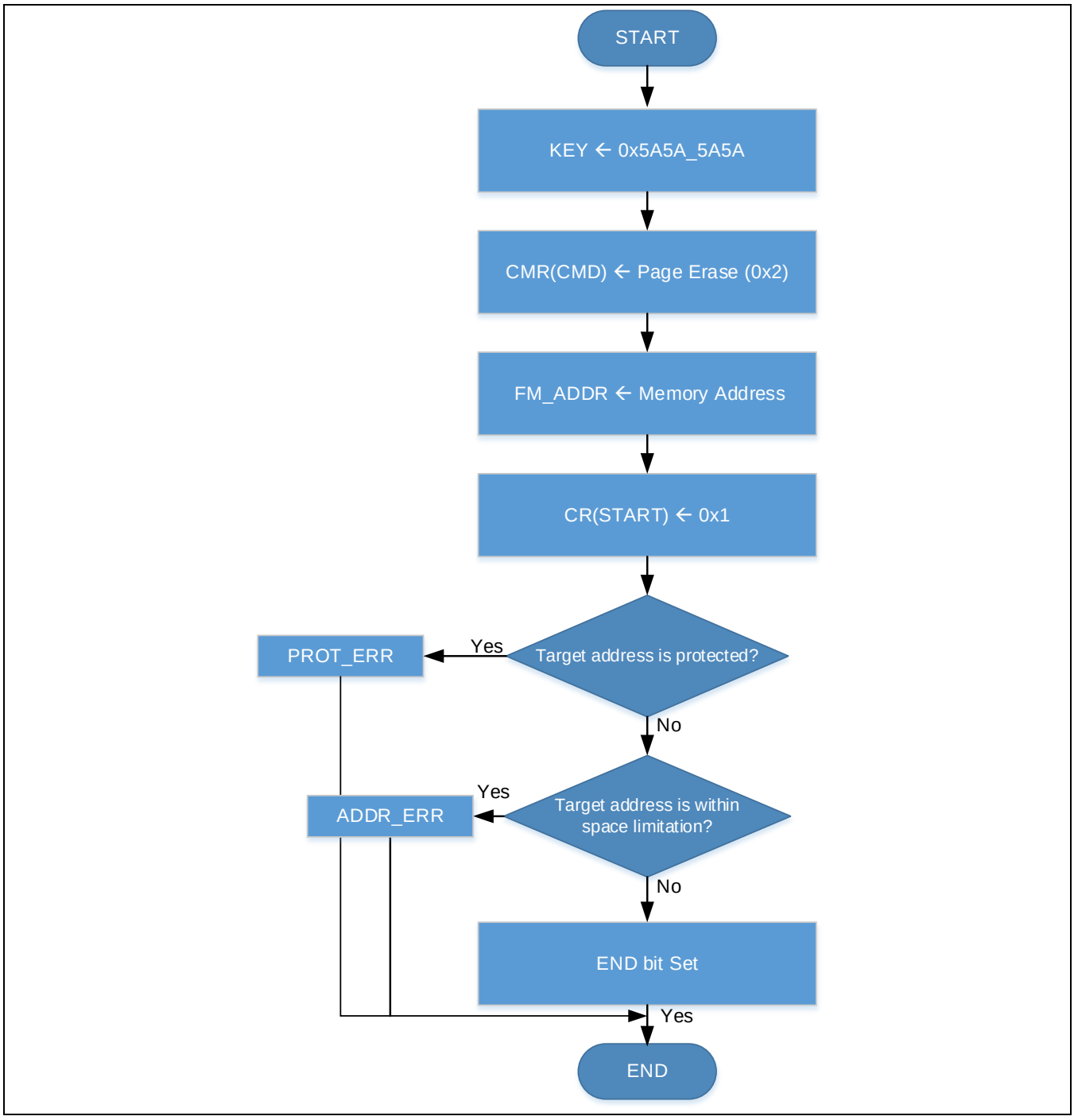


Figure 6-6 页擦除操作流程图

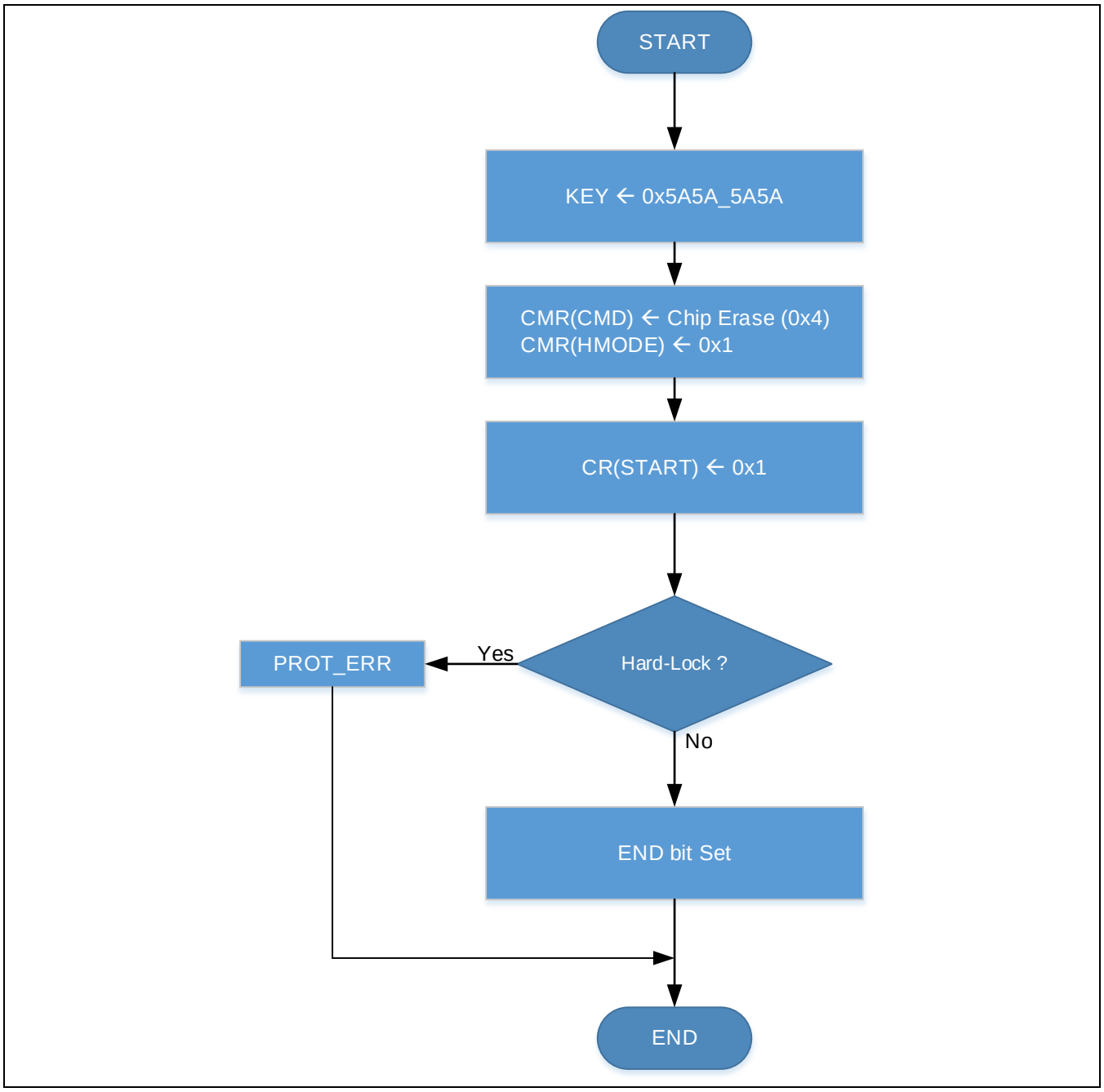


Figure 6-7 片擦除操作流程图

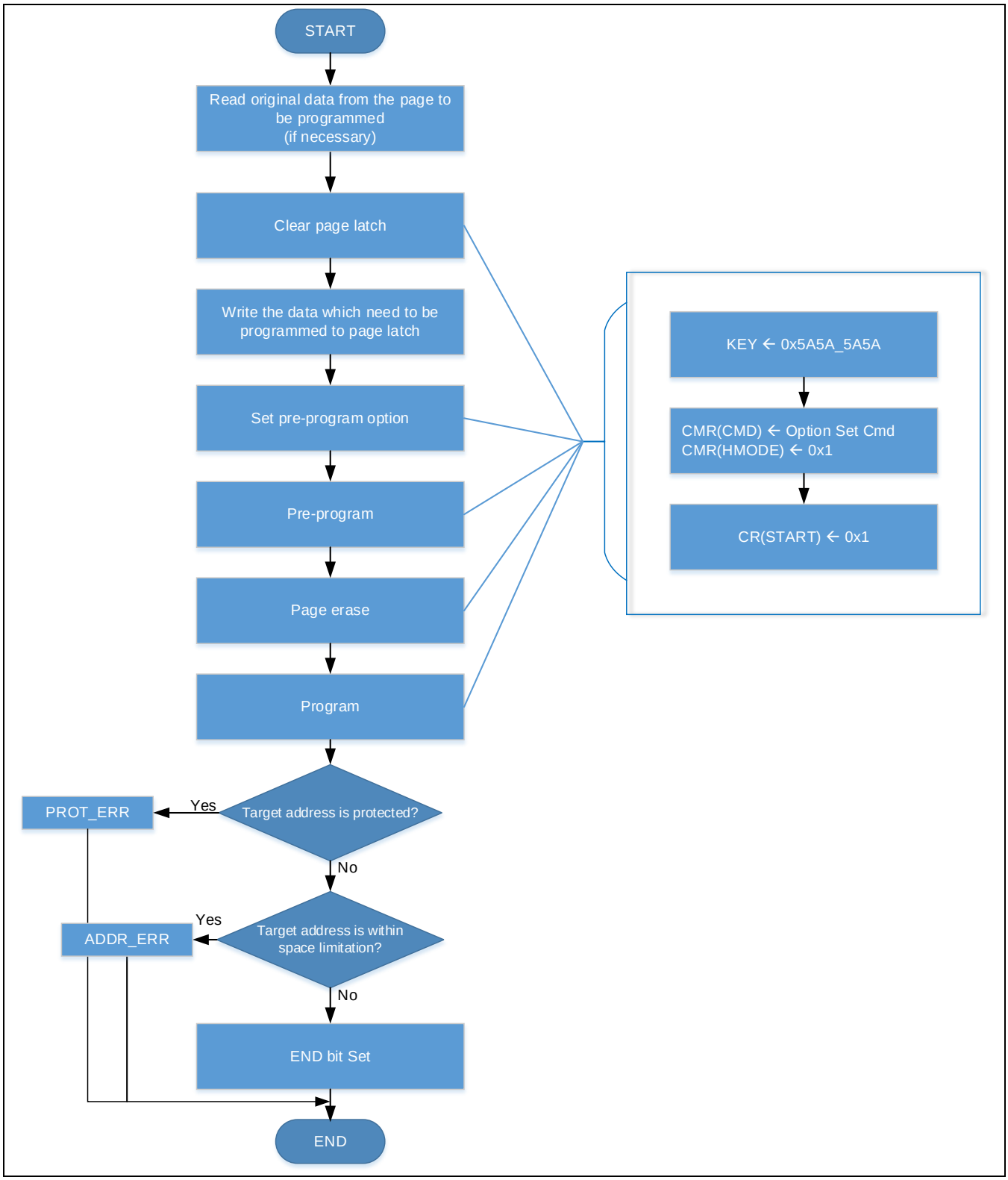


Figure 6-8 自定义选项写操作流程图

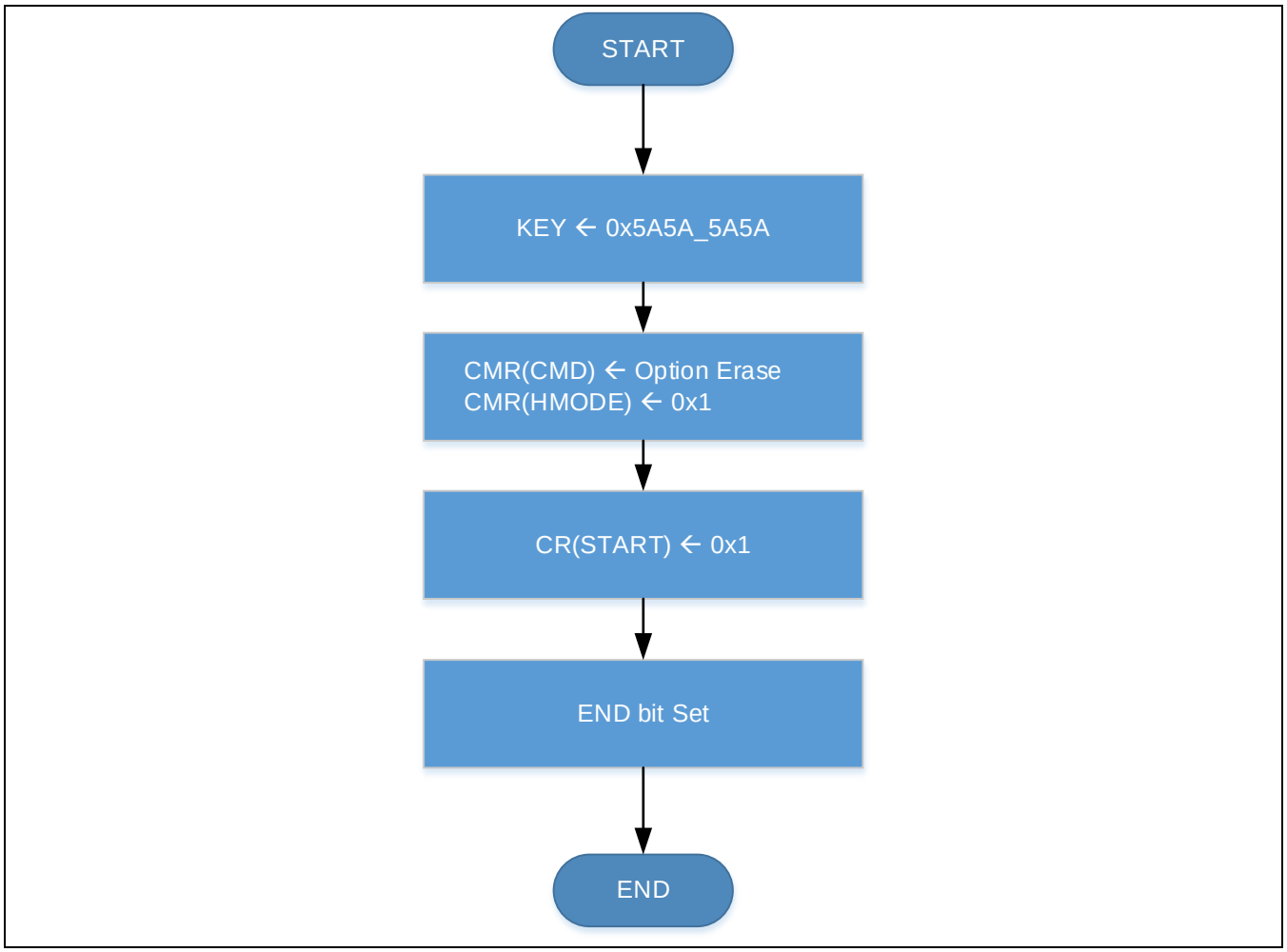


Figure 6-9 自定义选项擦除操作流程图

5.3 寄存器说明

5.3.1 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	0x0032_f102
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
RSVD	0x1C	保留	0x0000_0000
IFC_KR	0x20	ISP 秘钥寄存器	0x0000_0000
IFC_IMCR	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_ICR	0x30	中断状态清除寄存器	0x0000_0000

5.3.2 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
RSVD								IDCODE																							
0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	1	1	0	0	0	1	0	0	0	0	0	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	[23:0]	R	ID 代码 (0x32f102)

5.3.3 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

5.3.4 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

5.3.5 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											PROT					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	RW	RW	RW	RW	RW	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 (Page Erase) 0x3 保留，禁止使用 0x4 片擦除 (Chip Erase) 0x5 自定义选项擦除 0x6 预编程设定 0x7 页缓存清除 0x8 – 0xC 保留，禁止使用 0xD 禁用调试口重映射 (SWD Remap) 0xE 使能调试口重映射 (SWD Remap) 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。
PROT	[20:16]	RW	**保护功能选择寄存器** [20] : ENCRYPT [19] : SWDP [18] : RDP [17] : HDP_FULL [16] : HDP_4K 在 CMD 的写 User Option 命令中，使用 PROT 位来选择保护功能的使能，将相应位置 1 表示使能该项保护功能。

5.3.6 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

5.3.7 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															SPEED	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	RW	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
SPEED	[16]	RW	**FLASH IP 速度模式选择** 0: 低速模式 1: 高速模式

注意：不同的系统时钟频率下，WAIT 和 SPEED 的参考值如下表。

	WAIT	SPEED
24MHz < SYSCLK ≤ 48MHz	2	1
16MHz < SYSCLK ≤ 24MHz	1	1
SYSCLK ≤ 16MHz	0	0

5.3.8 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

5.3.9 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。

5.3.10 IFC_IMCR (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	ADDR_ERR	UDEF_ERR	PROT_ERR	RSVD									PEP_END	PGM_END	ERS_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	RW	RW	RW	R	R	R	R	R	R	R	R	R	RW	RW	RW

Name	Bit	Type	Description
ERS_END	[0]	RW	**擦除指令执行完成中断的原始状态** 0: 禁止中断 1: 使能中断
PGM_END	[1]	RW	**编程指令执行完成中断的原始状态** 0: 禁止中断 1: 使能中断
PEP_END	[2]	RW	**预编程指令执行完成中断的原始状态** 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

5.3.11 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									PEP_END	PGM_END	ERS_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
ERS_END	[0]	R	**擦除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
PGM_END	[1]	R	**编程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
PEP_END	[2]	R	**预编程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

5.3.12 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									PEP_END	PGM_END	ERS_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
ERS_END	[0]	R	**擦除指令执行完成中断的原始状态** 0: 该中断没有发生 1: 该中断发生
PGM_END	[1]	R	**编程指令执行完成中断的原始状态** 0: 该中断没有发生 1: 该中断发生
PEP_END	[2]	R	**预编程指令执行完成中断的原始状态** 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

5.3.13 IFC_ICR (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									PEP_END	PGM_END	ERS_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	W	W	W

Name	Bit	Type	Description
ERS_END	[0]	W	**擦除指令执行完成中断的原始状态** 0: 无效 1: 清除中断
PGM_END	[1]	W	**编程指令执行完成中断的原始状态** 0: 无效 1: 清除中断
PEP_END	[2]	W	**预编程指令执行完成中断的原始状态** 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

6 系统控制器（SYSCON）

6.1 概述

系统控制器用于管理和配置系统的时钟以及和系统工作相关的功能模块，包括不同工作模式下的具体时钟配置，功耗优化控制，系统运行可靠性监测和异常处理（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），以及系统安全信息和工程信息等。

通过系统控制器还可以对系统的缺省硬件配置状态（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。系统中若存在支持特性微调的模拟外设，其调整功能一般也通过系统控制器进行微调。

如果系列内芯片不具有某一外围，那它就不具备该外围的所有资源。具体参考芯片的数据手册。

系统控制器的基本特性：

- 系统时钟源选择和 HCLK/PCLK 频率管理
  - 支持多种时钟源作为系统时钟运行：
    - 内部低速振荡器（IMOSC）为缺省时钟源：5.556MHz/4.194MHz/2.097MHz/131.072KHz
    - 内部高速振荡器（HFOSC）：48MHz
    - 外部晶振（EMOSC）：0.4MHz ~ 24MHz
    - 内部超低功耗振荡器（ISOSC）：27KHz
  - 可编程 CPU 时钟（HCLK）和外设时钟（PCLK）
  - 外部时钟失效监测（Clock Fail Monitor），支持时钟去抖选项
  - 可选择的系统内部时钟源输出（CLO）
- 各个外设独立的时钟门控，提供功耗优化选择
- 独立看门狗模块，支持上电自动运行并可通过 USER OPTION 定义使能
- 支持复位源记录功能。可用于诊断系统复位，为错误恢复提供支持
- 支持低功耗优化控制。对于不同 CPU 运行模式和负载情况，用户可以自定义电源策略，从而有效降低系统动态功耗。
- 外部中断管理支持从 GPIO 输入的触发信号作为系统事件触发 CPU 中断。
- 低压检测模块（Low voltage Detector）支持外部供电电压监测，在电压低于预设值时可以产生系统

中断或者触发系统复位。

- 存储可靠性管理功能支持使能 FLASH 或者 SRAM 的硬件校验功能。
- UID、FINFO 等工程信息只读寄存器

6.2 功能描述

6.2.1 时钟管理和控制

系统控制器的最主要的功能是管理和分配系统时钟。整个系统的工作时钟结构如下图所示。

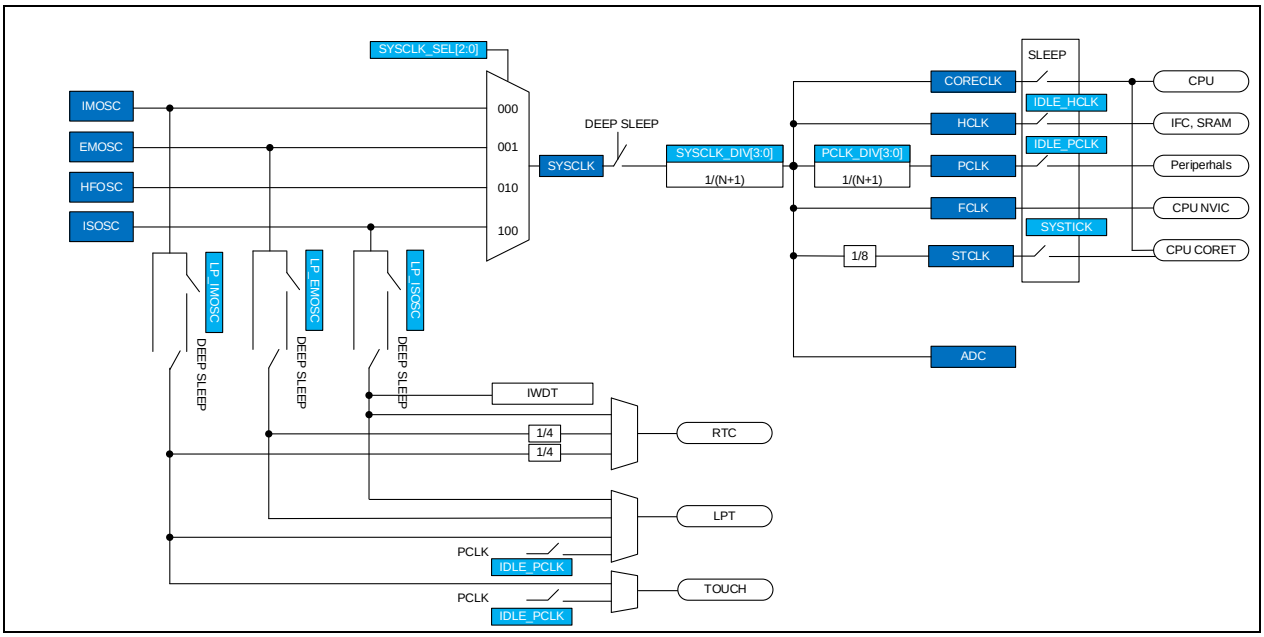


Figure 7-1 时钟结构示意图

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。

SYSCLK 作为系统时钟，提供整个系统的基础工作时钟。各个模块包括 CPU，MEMORY 和外设的时钟均由系统时钟产生。系统时钟通过时钟选择电路，支持多个时钟源中的任意一个作为系统时钟的输入，可以通过 SCLKCR 寄存器进行设置。当选择了特定的时钟源后，时钟源的当前频率决定了系统最高的运行频率。当系统时钟从一个时钟源切换到另一个时钟源时，系统时钟会出现短暂的停顿，以保证不同系统频率切换间不会产生时钟毛刺，当系统时钟再次稳定后，系统时钟会自动恢复。

系统时钟分别通过两个预分配器产生 HCLK 和 PCLK 时钟。由 HCLK 时钟控制的模块主要是系统高速模块，包括 CPU、内存控制单元、GPIO 控制器等。由 PCLK 时钟控制的模块主要是外设模块，包括 TIMER、PWM、通讯接口等。PCLK 的分频是基于 HCLK 进行分频的，所以 PCLK 的时钟频率不会超过 HCLK 的频率。

每个外设都有独立的时钟使能控制开关，在操作该外设前，必须使能该模块的 PCLK 时钟控制。PCLK 的时钟控制可以通过 PCER、PCDR 这组寄存器进行操作。对 PCER 寄存器的对应控制位写入 1 时，可以使能指定模块的 PCLK，对 PCDR 寄存器的对应控制位写入 1 时，可以关闭指定模块的 PCLK。通过读取 PCSR 寄存器可以获得开关的当前状态。关闭不使用的模块的 PCLK，可以有效降低系统的动态功耗。

CPU 的时钟在 NORMAL 模式下一直处于使能状态。在低功耗模式下，PCLK 和 HCLK 的使能或者禁止控制可以通过 GCER/GCDR 寄存器设置。具体配置可以参考本章节的低功耗模式部分。

CPU 内部的 CORET 计数器时钟基于系统时钟的 8 分频或 CPU 时钟，在使用 CORET 时，必须先通过 GCER 的控制位使能该模块的时钟。

6.2.2 时钟源的选择和切换

系统时钟可以根据不用应用要求，支持在多个时钟源间进行切换。系统支持多种时钟源作为系统的工作时钟，具体如下：

IMOSC（Internal Main OSC, 5.556MHz/4.194MHz/2.097MHz/131.072KHz）：

内部高精度主振荡器，提供 4 种可选的频率，以满足不同功耗要求。系统上电时，缺省选择 IMOSC 的 5.556MHz 作为工作时钟。

HFOSC（High Frequency OSC, 48MHz）：

内部高速振荡器，提供高效的 CPU 运行时钟。在高速时钟下工作时，必须注意 Flash 的读取速度匹配问题。在切换到高速时钟前，必须配置合适的 Flash Wait 节拍以匹配 CPU 的速度，匹配关系和设置方法具体参考 Flash 控制器章节。

EMOSC（External Main OSC, 32.768KHz / 0.4MHz ~ 24MHz）：

外部晶振振荡器，可支持两种工作模式，针对低速 32.768KHz 的低功耗模式以及普通模式。

ISOSC（Internal Sub OSC, 27KHz）：

内部超低速振荡器，主要提供 IWDT 的计数时钟。同时作为外部晶振的失效监测管理时钟。

在芯片上电初始化时，系统将自动选择 IMOSC 最高频率作为缺省工作时钟。在系统完成上电复位和硬件初始化后，系统软件可以通过设置相应的寄存器将系统工作时钟切换到希望的时钟源，并设置相应的 HCLK 和 PCLK 分频系数。

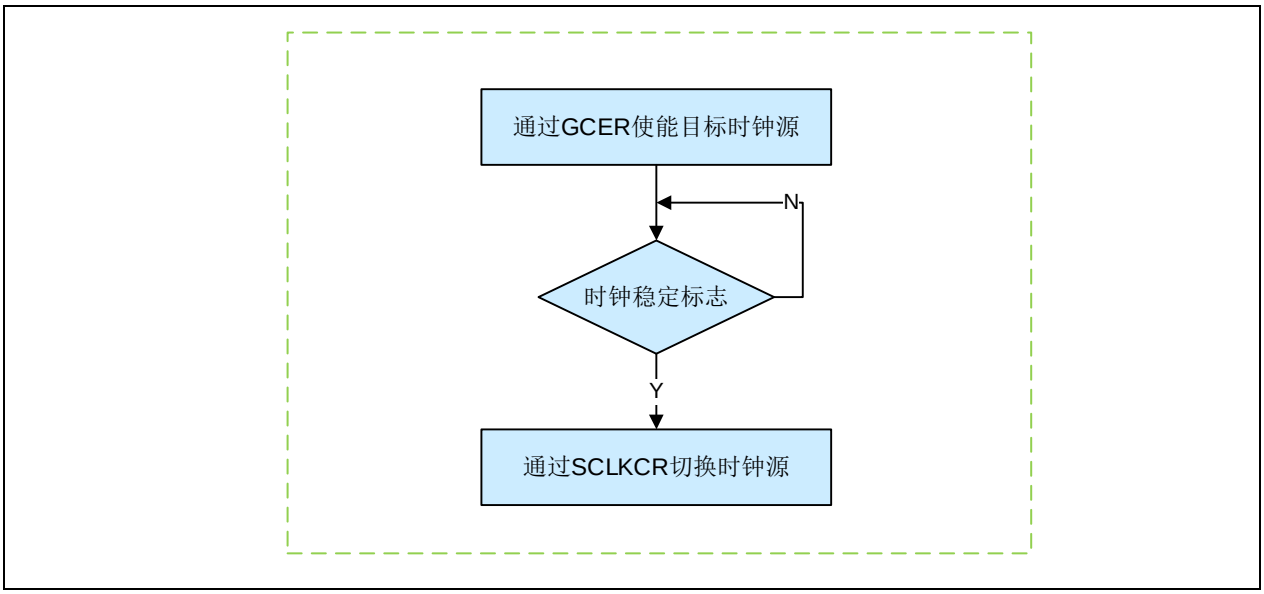


Figure 7-2 时钟源切换示意图

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，由 IMCLK 作为系统时钟，控制 DEEP-SLEEP 的初始化过程（包括系统时钟设置的备份，EMOSC，ISOSC 的停止，功耗模式的切换），在完成所有低功耗初始化后，IMOSC 会自动停止。芯片自此进入 DEEP-SLEEP 模式，直到被事件触发唤醒。DEEP-SLEEP 的初始化过程根据系统当前时钟设置的不同会有所差异。当系统退出 DEEP-SLEEP 模式时，系统工作时钟将被自动恢复到 STOP 指令执行前的情况。所有由于工作模式切换造成的时钟切换对于用户程序都是透明的。

除了软件触发的系统时钟切换和系统工作模式更改触发的系统时钟切换，还有一种系统时钟切换会发生在外部时钟（EMOSC）失效时。具体介绍可以参考本章的外部时钟可靠性监测部分。

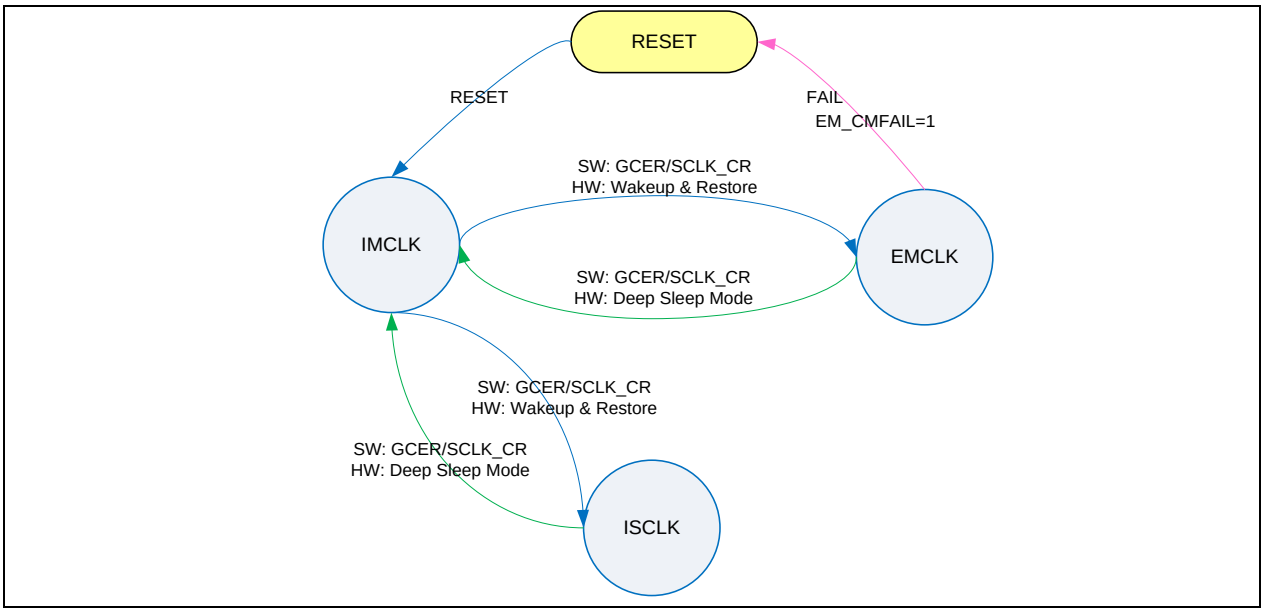


Figure 7-3 时钟切换状态机

6.2.2.1 选择内部时钟源进行工作

芯片内部包含 3 个振荡器，用户可以根据不同实际应用，选择合适的振荡器作为系统时钟进行工作。系统上电复位后，缺省采用 IMOSC 的 5.556MHz 频率进行工作，这样既能保证一定的执行速度，也进一步的降低了上电初始化时的动态功耗，保证在系统上电时不会出现大负荷的电流需求。

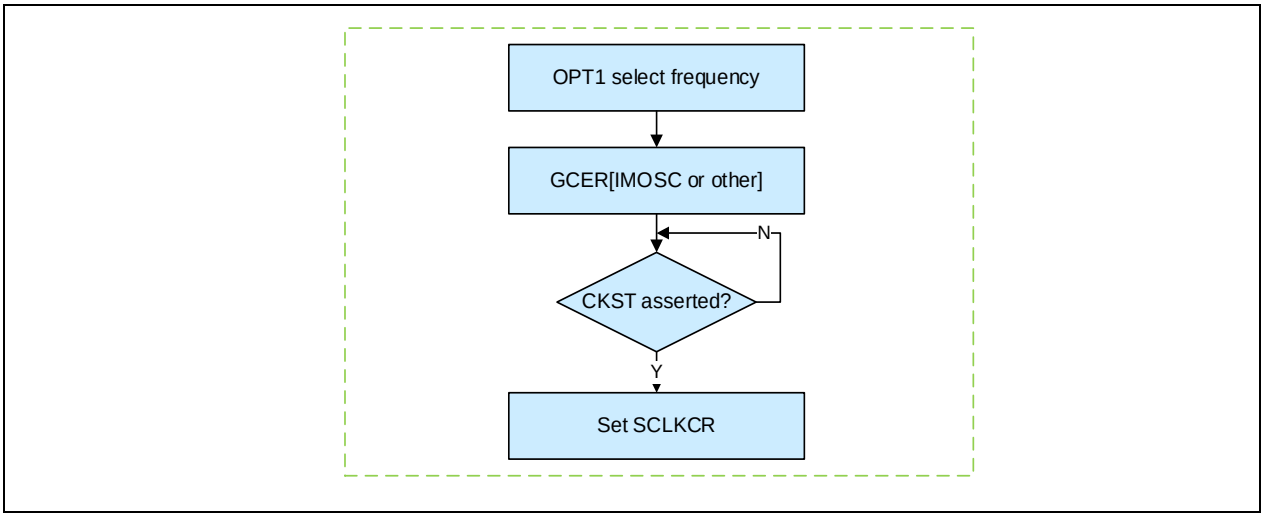


Figure 7-4 配置内部时钟源

在系统硬件初始化完成后，用户可以根据实际应用选择更低频率的时钟作为系统时钟，或者切换到高速内部振荡器进行工作。对 IMOSC 的频率设置，可以通过 OPT1[IMO_FSEL]控制位进行选择；对 HFOSC 的频率设置，可以通过 OPT1[HFO_FSEL]控制位进行选择。当对已经选择为系统时钟的时钟源进行频率设置时，频率切换间的同步 delay，会引起系统时钟短暂失效，此失效会延迟指令执行时间或者造成波形输出的当前周期变长，应用时需要注意。通过写入‘1’到 GCER 寄存器的相应控制位，可以使能对应的时钟源；通过写入‘1’到 GCDR 寄存器的相应控制位，可以关闭对应的时钟源；时钟源的当前状态可以通过 GCSR 寄存器获取。当设置使能某个时钟源时，必须在使能控制以后（设置 GCER 后），对相应时钟源的时钟稳定状态进行查询。只有当目标时钟源稳定后，才能将系统时钟切换到目标时钟源，否则切换无效。当切换错误时，ERRINF 寄存器的 ER_AHBCLK 位将被置位，同时 CMD_ERR 事件会被触发。

芯片还内置一个超低速度的低功耗振荡器（ISOSC）。ISOSC 作为独立看门狗的工作时钟，随着 IWDT 一同工作，以保证 IWDT 不会被系统其他时钟干扰。ISOSC 也支持作为系统工作时钟，供系统在超低速度下工作，以达到超低功耗的要求。

6.2.2.2 内部时钟源频率微调

内部时钟源（包括 IMOSC，HFOSC，ISOSC）在出厂前已经经过精度校准，以保证振荡器的频率在 Spec 定义范围之内。系统也为客户提供了在程序中再次调整频率的途径，以满足客户自定义频率的需求。

所有的内部时钟源都支持频率粗调，其调节方式可以参考下面的公式进行计算。但由于芯片内部寄生效应，该公式可用于估算 Target 频率，实际结果需要在应用中确认。

$$\mathrm{F}tar = \mathrm{F}trim\frac{\mathrm{K}}{\mathrm{K}-\Delta\mathrm{FSEL}}$$

其中：Ftar 为 Target 频率，即需要最终调节到的频率；

Ftrim 为当前 OSC 的缺省频率；K 为运算系数；ΔFSEL 为 TRIM 的调整值。

K 值按照下面的表格进行计算（其中 TRM 为 CLCR 中[HFO_TRM]、[IMO_TRM]和[ISO_TRM]缺省值）：

Table 7-1 系数 K 值计算方式

OSC	K 值
IMOSC	367-TRM
HFOSC	183-TRM
ISOSC	339-TRM

例如：当前芯片 HFOSC 缺省频率为 48.2MHz，缺省 HFO_TRM 为 0x47；若调整 HFOSC 到 47.5MHz，则先根据表格算得 K 为 183-71 = 112。根据公式计算ΔFSEL = 112 – (48.2 x 112 / 47.5)，结果为-1.6。所以 HFO_TRM 应减少 2 可以获得 Target 近似频率。

其中 HFOSC 还提供精细调节的功能。HFOSC 的细调通过 CLCR[HFO_TUNE]控制进行调节，其缺省值为 0h100，当增大 HFO_TUNE 时，HFOSC 的频率变快，减少 HFO_TUNE 时，HFOSC 的频率变慢。HFO_TUNE 的每一步调节都是等量的且分辨率为 0.65‰，即 HFO_TUNE 的值每变化 1，频率变化

6.2.2.3 选择外部时钟源进行工作

在对时钟精度有更高要求时，建议用户采用外部晶振作为系统的工作时钟。外部振荡器可以工作在两个模式：普通模式和低功耗模式。在低功耗模式下，振荡器专门针对 32.768KHz 进行功耗优化，以获得更低的工作电流。外部振荡器的切换过程如下图所示：

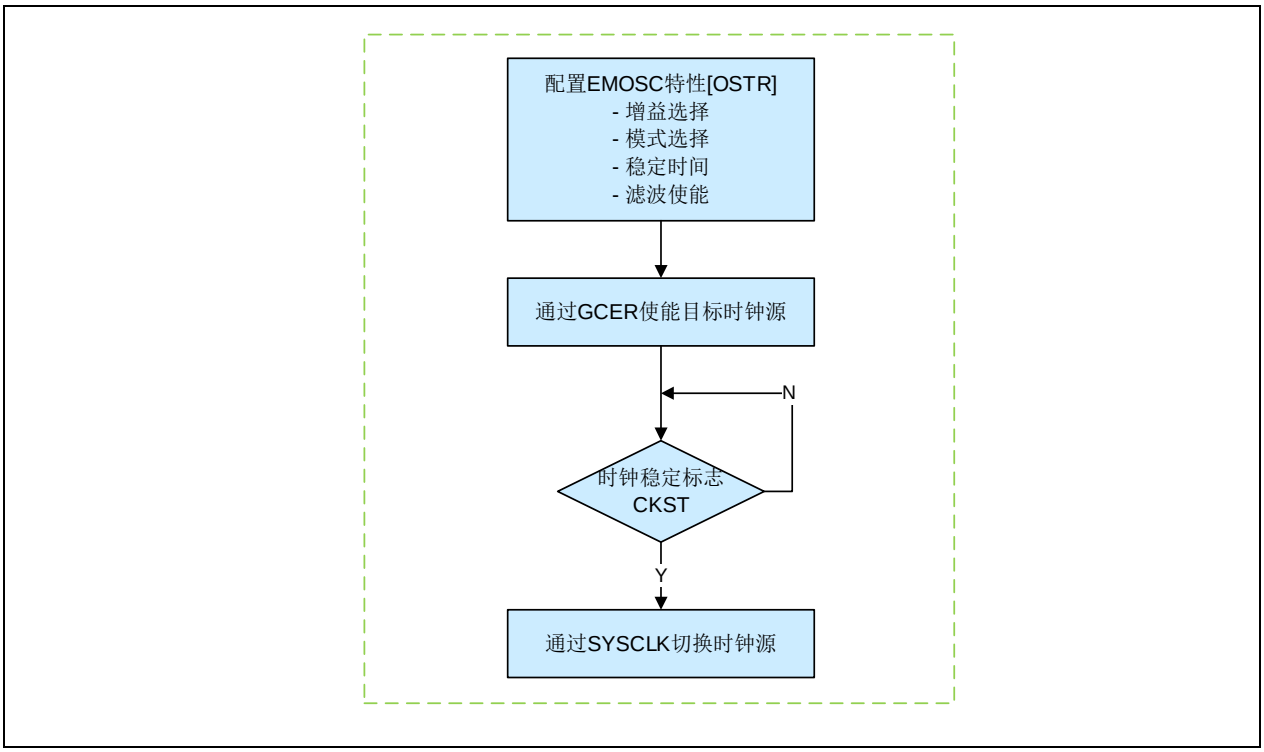


Figure 7-5 配置外部时钟源

在使能外部晶振前，系统通过 OSTR 寄存器对外部晶振特性进行设置。首先需要选择振荡器的工作模式，缺省模式下晶振工作于普通模式，即外接 1MHz~24MHz 的晶振，当外接 32.768KHz 时，需要将 OSTR[LFSEL] 控制位设置到低功耗模式。

外部晶振的增益控制调节可以针对不同晶振和外部负载电容做调节，以保证振荡器起振条件获得满足。在起振后，可以适当调低增益控制，用来减少振荡器功耗。一般推荐的 GM 设置可以参考下面的表格。

Table 7-2 CYOSC_GM 设置说明

EMOSC 频率	CYO_GM[4:0]
16MHz	11111
10MHz	11111
8MHz	11111
4MHz	11111
1MHz	11111
500KHz	11111
32.768KHz	00111

稳定时间设置用于控制晶振从使能到时钟输出稳定的计数周期。由于晶振启动后一段时间内输出的时钟具有很大的 jitter 漂移，这段时间内不能为系统提供稳定的时钟。通过设置 OSTR 的 EM_CNT 控制位可以设置在振荡器输出多少个时钟后将振荡器的稳定标志位置位。稳定计数器是一个 18 位的计数器，计数器的计数值高 10 位和 EM_CNT 中的设置进行比较，当比较值满足条件时，稳定标志被置起。EM_CNT 控制位不允许设置为 0。缺省的 EM_CNT 值为 0x3FF，在 8MHz 晶振工作时，稳定计数器的计数时间为 32.7ms。

外部晶振支持 glitch 滤除选项。在某些恶劣工作环境中，由于外部强干扰可能导致晶振的时钟信号引入glitch。当 Glitch 的发生点和时钟的上升沿非常接近时，可能引起内部逻辑电路的时序异常。而时序异常可能导致芯片工作失效。为保证时钟的可靠，此时用户可以通过使能晶振的 glitch 滤除功能避免 glitch 向内部时钟电路的传输。该功能通过 OSTR[EM_FLTEN]和[EM_FLTSEL]控制位进行配置。时钟的滤波功能配置必须在EMOSC 禁止时进行配置，一旦 EMOSC 使能，则任何对滤波器的设置操作均被禁止。

6.2.2.4 外部时钟的可靠性监测

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（IMOSC）作为参考时钟源，必须同时使能。一个内部的 6 位递减计数器在 EMOSC 的时钟控制下进行计数，每一次 EMOSC 的时钟从低变化到高都会被内部计数器检测到，从而重置计数器。当递减计数器未被及时重置，而计数到零时，则判定外部时钟失效。

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，可以通过设置 CMRST 位来使能自动产生系统的复位，或者切换到内部时钟。外部时钟失效监测的结果可以表现为以下两种情况：

- 芯片复位 （当 CMRST 位置位时）
- 切换到内部时钟

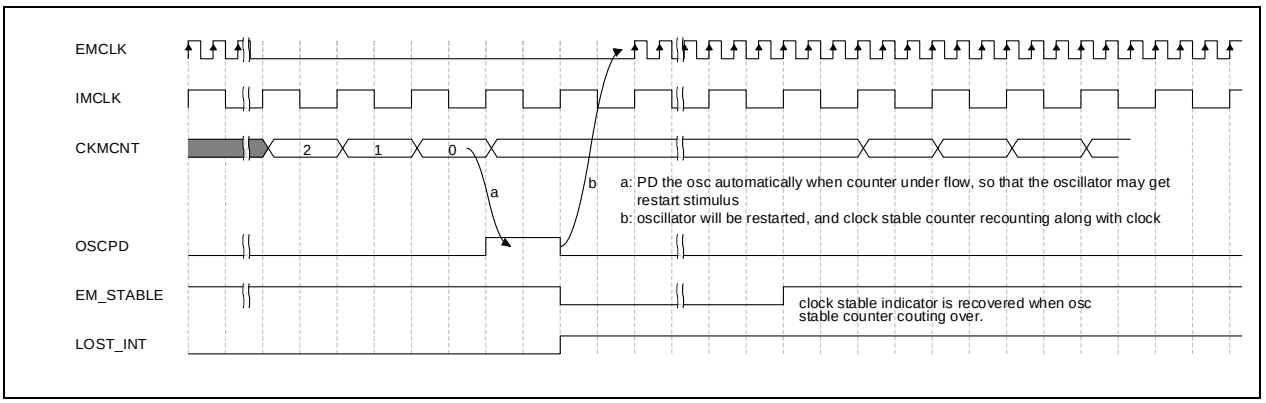


Figure 7-6 EMOSC 失效监测

当 EMOSC 失效时，可以产生 EM_CMLST 中断。系统在处理此中断时，需先通过 GCDR 寄存器禁止EMOSC，然后再次尝试通过 GCER 来使能 EMOSC。当 EMOSC 再次正常工作后，可以切换系统时钟到EMOSC 进行工作。

内部递减计数器的重载值通过 OPT1[EMCKM_DUR]进行设置。重载值设置越大，则允许的时钟偏差就越大，但是检错的时间就越长。用户需要根据实际的外部时钟频率配置合适的检查周期。在没有特别严苛的响应时间要求时，建议选择较大的重载值。

6.2.2.5 时钟输出配置（CLO）

系统支持将内部时钟通过外部管脚（CLO）输出，输出的时钟可以通过 OPT1[CLOMX]控制位进行选择。由于封装的寄生电容存在，所以在输出高频信号时会产生很大的驱动电流和 EMI 干扰，建议当 CLO 选择输出高频时，通过 OPT1[CLODIV]对时钟先进行分频，然后再输出。

6.2.3 低压监测和复位

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 支持掉电监测和电压恢复监测两种。

通过置高 LVDCR 的 LVDEN 控制位，使能 LVD 控制模块。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。当外部供电电压 VDD 低于 INTDET_LVL 的设置值或高于 INTDET_LVL 设置值时，系统将会产生 LVD 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）。通过 INTDET_POL 控制位可以选择中断触发的条件，选择 VDD 下降沿触发或上升沿触发，或者两者均可触发。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’。

系统复位后，缺省的 LVD 状态为关闭状态。由 LVD 产生的系统复位信号，不会清除 LVD 的使能状态。在低功耗模式下，LVD 也可以被使能，但由于受到低功耗模式下功耗控制限制，其精度会比普通模式下略低(SLEEP 模式不受影响)。

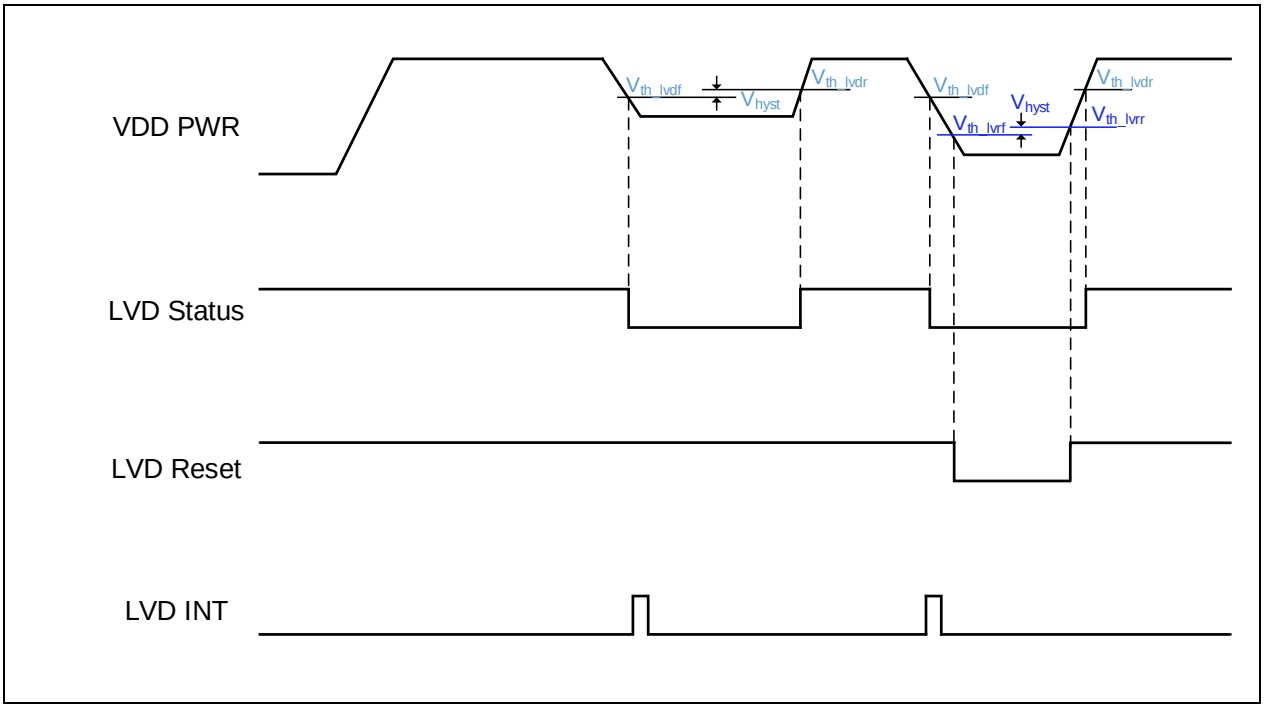


Figure 7-7 LVD 工作时序图

6.2.4 低功耗模式及唤醒

在缺省上电复位后，系统工作于运行模式（RUN MODE）。在某些特殊应用下，CPU 不需要再继续工作，出于节省功耗考虑，用户可以选择将系统切换到低功耗模式。在需要 CPU 再次处理任务时，通过预先设置的触发条件对系统进行唤醒。

系统支持的低功耗模式有三种：

- 低速运行模式 （LOW POWER RUN）：CPU 运行频率低于 1MHz，代码在 SRAM 或者 Flash 中运行。低速模式下需要通过软件配置内部逻辑供电切换到低功耗 LDO，并切换 FLASH 到低速模式，以实现有效降低工作电流的目的。当代码在 SRAM 中运行条件下，通过关闭 Flash 和 Flash 参考电压源，可以进一步降低功耗。通过使能 OPT1[LPMD]控制位切换到低速运行模式。当前系统时钟为 HFOSC 时，该控制位不起作用。

- 睡眠模式 （SLEEP MODE）：CPU 时钟被关闭，所有外设的时钟可以通过 PCER/PCDR 寄存器进行预先设置为关闭或者使能。CORT 的时钟不会被关闭，除非设置 GCDR[SYSTICK]控制位进行关闭。AHB 和 APB 的 CLOCK 是否使能，可以通过 GCDR[IDLE_HCLK]和[IDLE_PCLK]控制位进行设置。当有任何外设中断发生时，都可以唤醒 CPU，并退出 SLEEP 模式。

- 深睡眠模式（DEEP SLEEP MODE）：CPU 时钟被关闭，所有外设时钟被关闭。由于某些外设可以独立于 PCLK 工作（例如 RTC，LPT 和 TOUCH），可以通过配置 GCER[STP_xxx]控制，选择时钟源是否关闭。

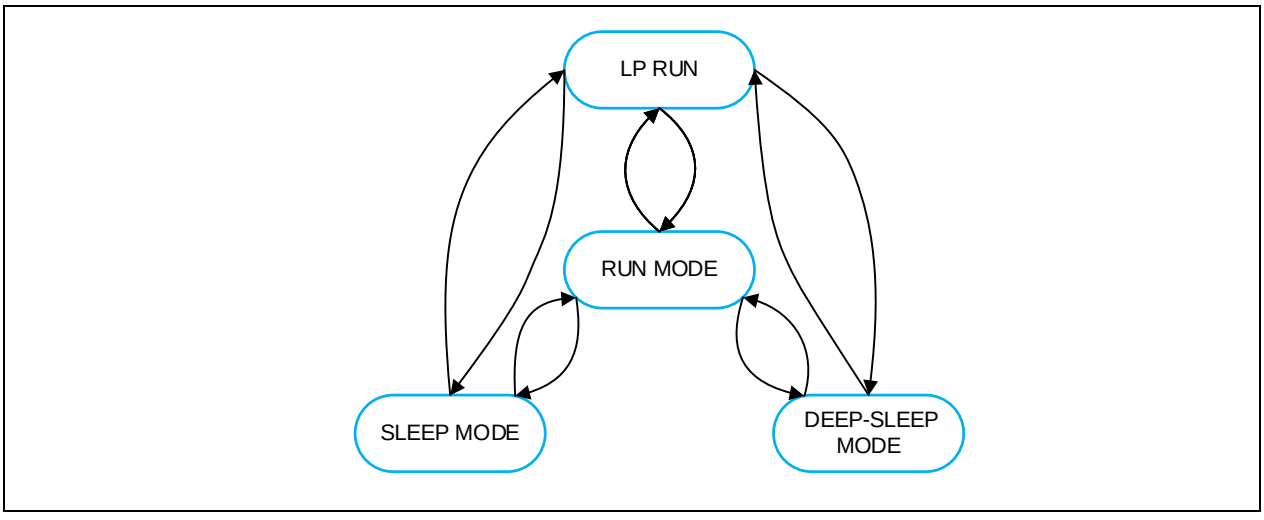


Figure 7-8 模式切换示意图

在不同模式间的切换和唤醒条件可以参考如下表格。

Table 7-3 低功耗模式总结

MODE	ENTRY	WAKEUP	CLOCK STATUS
LP RUN	Set OPT1[LPMD] bit	Clear OPT1[LPMD] bit	The same as normal
SLEEP	DOZE Instruction	Any Interrupt	CPU Clock Off

			No effect on other clock
DEEP-SLEEP	STOP Instruction	LVD, EXI, WDT, RTC, TOUCH, LPT interrupt. Corresponding WKEN bit should be set	All Clock is Off, including CPU and peripheral clock in default Clock source can be selected to keep working by STPEN bit

Table 7-4 在不同工作模式下的功能可用性

PERIPHERAL	RUN	LP RUN	SLEEP	DEEP SLEEP	
				—	WAKEUP
CPU	Y	Y	—	—	—
FLASH	Y	O(2)	—	—	—
SRAM	Y	Y	Y	Y	—
HFOSC	O	—	O	—	—
IMOSC	O	O	O	O(3)	—
ISOSC	O	O	O	O(3)	—
EMOSC	O	O	O	O(3)	—
Clock Monitor	O	O	O	O	—
RTC	O	O	O	O	O
UART	O	O	O	—	—
I2C	O	O	O	—	—
SPI	O	O	O	—	—
ADC	O	O	O	—	—
TIMER	O	O	O	—	—
LPT	O	O	O	O	O
IWDT	O	O	O	O	O
WWDT	O	O	O	—	—
CORET	O	O	O	—	—
TOUCH	O	O	O	O	O
EXI	O	O	O	O	O

1. 图例：Y = Yes（Enable 有效），O = Optional（可配置），— = Not available（不可用）。
2. Flash 可以通过软件配置为停止工作，缺省状态为使能（即不停止工作）。
3. 在 DEEP-SLEEP 模式下，缺省将关闭所有的时钟源，但为保证某些外设可以继续工作，可以设置在该低功耗模式下不关闭特定的时钟源。

调试模式

在调试模式下，当系统进入低功耗模式时，在功能上可以调试模式的切换，但是此时系统并没有真正进入低功耗模式，系统时钟在此时仍旧保持工作，以确保调试模式不会因为进入低功耗模式而退出。在调试模式下

进入低功耗模式后（执行 DOZE 或者 STOP 指令后），系统将挂起直到被唤醒，唤醒后系统将继续运行直到断点被检测到。若程序调试断点设置在 DOZE 或者 STOP 指令上，则程序在唤醒后，停止在 DOZE 或者 STOP 指令后一条指令处。

在低功耗模式下，若调试接口没有关闭（GPIO 的 AF 功能设置为调试功能时），则任何来自调试器的连接尝试都会将系统从低功耗模式唤醒。

6.2.4.1 运行模式（RUN MODE）

运行模式是系统工作的普通模式。在此模式下所有功能均可运行并不受限制。此模式下，追求处理器的处理性能作为主要目标，所以 LDO 和基准电压等均处于高驱动能力或者高精度模式下。系统可以通过设置相应配置寄存器选择不同的时钟源作为系统时钟源，并根据应用要求对 HCLK 和 PCLK 设置不同的分频系数。

各个外设有独立的 PCLK 控制，缺省情况下，相应外设的 PCLK 时钟开关处于关闭状态。在对外设进行设置前，需要使能相应模块的 PCLK 开关。通过 SYSCON 的 PCER 和 PCDR 寄存器可以设置外设模块的 PCLK 开关。

6.2.4.2 低功耗运行模式（LOW POWER RUN MODE）

系统运行功耗主要取决于运行时的总线频率，以及 Regulator 和 Flash Memory 的工作电流。为进一步降低系统工作电流，可以将系统切换到低功耗运行模式下工作。在此模式下，Regulator 处于低功耗模式，且 Flash Memory 在完成读取操作后自动休眠。由于 Regulator 在低功耗模式下的响应速度限制以及 Flash Memory 存在休眠唤醒时间，所以在此模式下，最大的总线速度建议设置在 1MHz 以下，且 Flash 的读取时间间隔需要保证大于 8us。当 CPU 速度大于 Flash 的访问时间，需要设置适当的 WAIT CYCLE 以保证 Flash 的时序正确。

- **进入 LP RUN 模式的方式：**

进入 LP RUN 模式的方式可以参考如下步骤进行：

1）（可选）程序跳转到 SRAM 中执行，此后 Flash 将不再被 CPU 访问。可以通过 PWROPT[FLASH_PD]寄存器禁止 Flash Memory，PWROPT[EFLR_PD]关闭 Flash Memory 的参考电压源。

2）降低系统工作频率到合适频率(1MHz 以内)，若程序仍在 Flash 中执行，则通过设置 OPT1[EFL_LPMD]控制位使能 Flash 的低功耗访问模式，此时必须注意 Flash 的访问时间限制。

3）如果应用对功耗要求很高，那么通过设置 PWRKEY[VOSLCK]和 PWRCR[VOSEN] 使能 run 模式下的 VOS 功能；尝试 RUN_CFG 位段的各种低功耗模式（$I_{LP0} < I_{LP1} < I_{LP2}$），找到不影响应用的最低功耗模式。不同的低功耗模式有不同的驱动能力，功耗越低，驱动能力越弱。

- **退出 LP RUN 模式的方式：**

退出 LP RUN 模式的方式可以参考如下步骤进行：

1）如果使能了 VOS，通过设置 PWRKEY[VOSLCK]和 PWRCR[VOSEN]取消 run 模式下的 VOS。

2）清除 OPT1[EFL_LPMD]控制位，恢复 Flash 的访问时间限制。

3）切换系统频率到目标频率。

6.2.4.3 睡眠模式（SLEEP MODE）

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）不会被停止，进入模式前被使能的外设将继续工作，且 IO 工作不受影响，例如 GPIO 上的 TIMER 输出在进入 SLEEP 前被打开，则进入 SLEEP 后，该 IO 仍旧可以正常输出。但是通过 GCDR[IDLE_PCLK]控制位可以设置在 SLEEP 模式下也停止 PCLK。如果 SLEEP 模式下的 PCLK 被禁止，则需要注意外设在没有 PCLK 时可能不能工作，从而将不能正常产生唤醒事件。

在 SLEEP 模式下，所有振荡器的工作状态不会做任何改变。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。

任何外设事件或者中断都可以触发系统从该模式退出。

Table 7-5 SLEEP 模式总结

Characteristics	Description
Mode Entry	◆ All pending bits of interrupt is cleared. ISPR in NVIC is cleared ◆ Instruction ‘DOZE’ is executed
Mode Exit	◆ PSR[IE] bit is set ◆ Corresponding interrupt vector bit is enabled in NVIC IWER ◆ Corresponding interrupt event is triggered
Wakeup Latency	

6.2.4.4 深度睡眠模式（DEEP-SLEEP MODE）

在 DEEP-SLEEP 模式下，系统控制器将挂起所有的时钟源， 同时维持内部逻辑电源保持不变。但可以有一些例外：首先，IWDT 使能的前提下，ISOSC 将不受模式切换的影响，一直保持工作。其次，可以通过设置 GCER[LP_ISOEN/IMOEM/EMOEN] 控制位来设置时钟源关闭的例外情况。当相应时钟源的控制位被使能时，该时钟源在 DEEP-SLEEP 模式下将不会被自动关闭，其状态将一直保持在进入 DEEP-SLEEP 模式前的状态。该设置用于保证某些使用特殊时钟的外设可以在 DEEP-SLEEP 模式下继续工作，例如 LPTIMER，TOUCH 或 RTC 等。

在进入 DEEP-SLEEP 时，系统将首先挂起总线和外设时钟，然后切换系统时钟到内部 IMOSC，并依次关闭所有时钟源。GPIO 将保持在进入模式前的状态。DEEP-SLEEP 低功耗模式不影响 IO 的工作。

需要注意：1）由于 DEEP-SLEEP 模式下，为获得最大限度的节能效果，内部参考电压源将切换到低功耗模式，若在进入模式前，使能 LVD 功能，可能对 LVD 在 DEEP-SLEEP 模式下的精度有一定影响。2）当 IWDT 被使能后，在 DEEP-SLEEP 模式下不会被自动关闭，所以需要使能 IWDT 的 Alert 中断在看门狗溢出前唤醒系统并进行清除。若不希望 DEEP-SLEEP 下不断唤醒系统进行喂狗，可以选择 WWDT 作为看门狗使用。

当处理器从 DEEP-SLEEP 模式退出时，时钟源会被自动恢复到进入低功耗模式前的状态，并且系统工作时钟（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 7-6 可以唤醒 DEEP-SLEEP 的中断源

Peripheral	Event
GPIO	EXI interrupt of each GPIO
IWDT	Alert interrupt
RTC	Any
LPT	Any
LVD	Any
TOUCH	Any

Table 7-7 DEEP-SLEEP 模式总结

Characteristics	Description
Mode Entry	◆ All pending bits of interrupt is cleared. ISPR in NVIC is cleared ◆ Instruction ‘STOP’ is executed
Mode Exit	◆ PSR[IE] bit is set ◆ Corresponding interrupt vector bit is enabled in NVIC IWER ◆ Corresponding interrupt event is triggered
Wakeup Latency	

6.2.4.5 低功耗唤醒和中断服务

当系统进入低功耗模式后，只有通过中断进行唤醒。系统唤醒经过两个阶段，

- 第一阶段，当 SYSCON 检测到有中断发生，会自动切换工作环境为后续系统运行恢复环境，这包括切换 LDO 到相应的工作状态，切换参考电压源以保证精度，唤醒 Flash memory 并初始化，和系统时钟的恢复等。

- 第二阶段，当系统工作环境恢复以后，SYSCON 会提供所有的总线时钟包括 CPU 时钟，同时给予 CPU 相应的中断信号，CPU 在检测到中断请求信号后，会根据设置退出低功耗模式（DOZE 和 STOP 指令），并继续正常的取指和执行工作。CPU 退出低功耗模式由 NVIC 中的唤醒寄存器控制，当 Active 的中断没有在 NVIC 中设置为唤醒中断源，则 CPU 不会响应该中断（即使该中断已经置位），并继续保持低功耗模式。

CPU 退出低功耗模式后，根据 NVIC 是否使能了该中断的中断服务跳转，系统可以立即进入该中断的中断服务程序或者不响应中断而继续 DOZE 或 STOP 指令后的代码。必须注意，当中断发生后，即使在 NVIC 中没有使能该中断的中断服务跳转，也会导致该中断在 NVIC 中的 pending 标志被置位。所以必须通过软件清除该标志，否则接下来的 DOZE 或 STOP 指令将不会进入低功耗模式。具体可以参考 CPU 相关文档或者 INTERRUPT 章节。

初始化的时间为可以用下面的公式进行估计：

$$T_{init\_deep\text{-}sleep} = T_{imosc\_stable} + T_{clk\_sw} + T_{emo\_dis} + T_{hfo\_dis} + T_{iso\_dis} + T_{imo\_dis}$$

其中

- T_{imosc_stable} 为 IMOSC 的稳定时间。具体来说，IMOSC 的稳定时间大约为 64 个 IMCLK 周期（5.56MHz 时，一个 IMCLK 周期为 180ns）。如果在进入 DEEP-SLEEP 前，IMOSC 已经使能，则该时间可以忽略；

- T_{clk_sw} 为系统自动切换控制时钟的时间，为 2 个 IMCLK 周期。如果在进入 DEEP-SLEEP 前，系统时钟即为 IMOSC，则该时间可以忽略；

- T_{emo_dis} 为 EMOSC 的关闭时间，当 EMOSC 频率大于 IMOSC 时，为 2 个 IMCLK 周期，反之为 2 个 EMCLK 周期。如果在进入 DEEP-SLEEP 前，EMOSC 没有使能，则该时间可以忽略；

- T_{hfo_dis} 为 HFOSC 的关闭时间，为 2 个 IMCLK 周期。如果在进入 DEEP-SLEEP 前，HFOSC 没有使能，则该时间可以忽略；

- T_{iso_dis} 为 ISOSC 的关闭时间，为 2 个 ISCLK 周期。如果在进入 DEEP-SLEEP 前，ISOSC 没有使能，则该时间可以忽略；

- T_{imo_dis} 为 IMOSC 的关闭时间，为 2 个 IMCLK 周期。

6.2.5 功耗管理和优化

为优化系统在不同工作条件下的功耗，特别是针对一些特别需要提供较低运行功耗的应用下。系统提供通过调节供电电压的方式来进一步降低系统功耗，即 VOS（Voltage Output Shift）功能。

VOS 使能时，可以通过软件调整 LDO 的输出特性，包括电压和响应速度，以及参考电压源的精度。在

不同工作模式和工作条件（如较低的 RUN 模式工作频率）下，可以通过选择合适的 VOS 配置以实现更低的系统整体功耗。

VOS 设置过程如下：

- 设置 VOSLCK 寄存器，使能 VOS 功能。
- 配置 PWRCR 寄存器，使能相应 VOSEN 控制位
- 配置 PWRCR[xxx_CFG]，尝试相应工作模式下的低功耗程度

在没有特殊功耗需求的前提下，可以不使能 VOS 功能。不是所有的低功耗配置都能保证应用的正常执行。因为降低的功耗是以牺牲驱动能力和性能为代价的。所以如果低功耗设置不能满足当前工作需求，可能造成系统的异常。

当对 VOS 的设置进行了更新，新的配置不会立即生效，只有在下次模式切换时才会生效。例如用户更新了 RUN 模式下 VOS 设置，则新的设置会在下一次从低功耗模式切换到 RUN 模式时才会生效。详细应用可以参考相关应用笔记。

6.2.6 复位源和复位信息记录

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 7-8 处理器复位信号源表

Reset Soure	Description
EXTRST	外部复位管脚触发的硬件 RESET 信号（低电平有效）。此复位只有在外部复位脚有效时可用（通过User Option配置）。
CMRST	时钟监测模块产生的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	由内部独立看门狗电路产生的复位信号。
SWRST	系统控制器产生的软件复位信号。 (IDCCR 寄存器中的 SWRST 控制位)
CPURST	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）。
CPUFATRST	CPU硬件错误产生不可恢复中断时，硬件产生复位（该选项通过寄存器IDCCR[CPUFTRST]控制位使能，或者由User Option配置缺省值）
SRAM_ERR	SRAM硬件校验错误，并且重试次数溢出后复位。
EFL_ERR	FLASH硬件校验错误，并且重试次数溢出后复位。
WWDTRST	WWDT产生的系统复位。

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。

6.2.7 外部中断

外部中断模块作为系统控制器的子模块，控制所有由外部管脚触发的中断事件。只要 GPIO 的输入通道被使能，该 GPIO 就可以被选择作为外部中断进行配置。处理器的所有 GPIO 都可以设置为外部中断输入。即使当 GPIO 被设置为其他 AF 功能时，只要通过 GPIO 中的 IECR 设置使能中断，此 GPIO 依然可以根据 IO 电平产生有效中断。例如：当 GPIO 配置为 UART 的 RXD 功能时，由于该 GPIO 的输入通道在 RXD 时使能，所以该 GPIO 作为 RXD 使用的同时，还可以作为外部中断触发源使用。

6.2.7.1 外部中断配置

SYSCON 的外部中断控制器支持 20 个中断触发源，分为 EXI0～EXI19。每个 EXI 对应 GPIO 中相应的外部中断组。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 中有独立的中断号（EXI_V0~EXI_V4）。

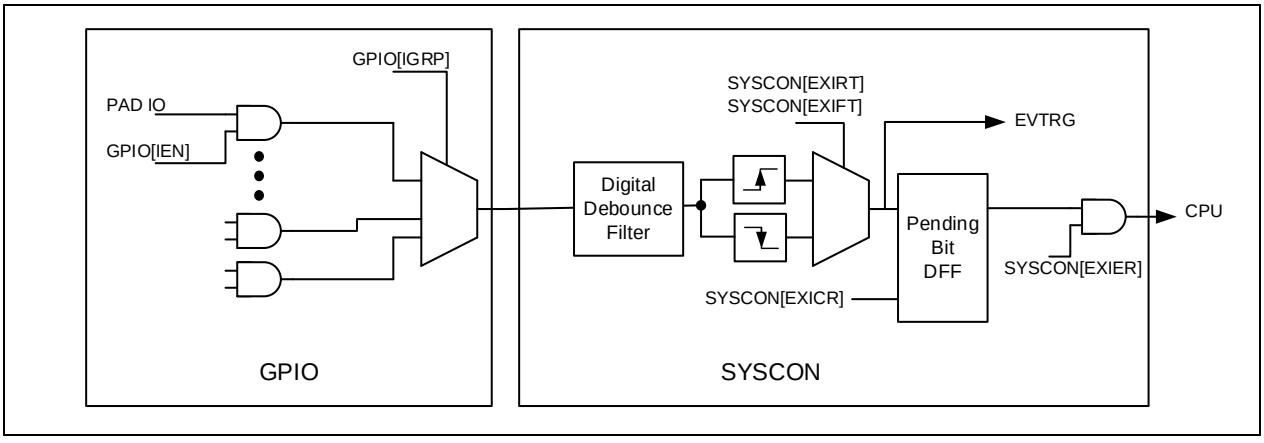


Figure 7-9 外部中断结构示意图

外部中断在配置时，由于每个 PAD IO 的初始状态以及 IEN 状态的不同，在切换配置过程中，可能产生错误的触发脉冲。为了避免此类情况，正确的操作过程为：

- 关闭 EXI 的中断
- 配置 EXI

- 对 Pending Bit 进行一次软件清除

- 使能 EXI

在对外部中断配置进行修改时，也建议遵循这个原则进行配置，以保证在修改 GPIO 的 GROUP 配置时，避免由于选择器的切换引入的毛刺而产生错误的中断触发。

如何配置和使能外部中断？

使能外部中断需要对三个模块的寄存器进行操作，分别为 GPIO，SYSCON 和 NVIC。

下表中，GROUPx 对应于 GPIO 中的 EXI group。EXI GROUP0~15 是以管脚名的后缀分组的。例如，GROUP0 只可能是 PA0.0, PA1.0, PB0.0, PB1.0 中的一个。而 EXI GROUP16~19 则依据管脚名的前缀来分组。例如，GROUP16 只可能是 PA0.0, PA0.1…PA0.7 中的一个。

第二栏中的 EXIx 表示 SYSCON 中的中断源。它们和 EXI GROUP 是一一对应的关系。外部中断的中断线控制在 SYSCON 中通过一组寄存器实现。EXIER/EXIDR 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。外部中断分组的配置详细参考 GPIO 的外部中断章节。

EXI_Vx 表示对应的外部中断发生时，向 CPU 发送的中断请求。具体分组信息请参考 NVIC 章节。

Table 7-9 EXI 中断配置信息

EXI GROUP IN GPIO	EXI LINE IN SYSCON	CPU INTERRUPT VECTOR
GROUP0	EXI0	EXI_V0
GROUP1	EXI1	EXI_V1
GROUP2	EXI2	EXI_V 2
GROUP3	EXI3	EXI_V 2
GROUP4	EXI4	EXI_V 3
GROUP5	EXI5	EXI_V 3
GROUP6	EXI6	EXI_V 3
GROUP7	EXI7	EXI_V 3
GROUP8	EXI8	EXI_V 3
GROUP9	EXI9	EXI_V 3
GROUP10	EXI10	EXI_V 4
GROUP11	EXI11	EXI_V 4

GROUP12	EXI12	EXI_V 4
GROUP13	EXI13	EXI_V 4
GROUP14	EXI14	EXI_V 4
GROUP15	EXI15	EXI_V 4
GROUP16	EXI16	EXI_V 0
GROUP17	EXI17	EXI_V 1
GROUP18	EXI18	EXI_V 2
GROUP19	EXI19	EXI_V 2

具体的操作流程如下：

1) 确认 EXI 的开关被关闭（通过 EXIDR 寄存器进行设置）
2) 设置 CPU 中 NVIC 的 EXI 中断为使能状态
3) 设置 GPIO 内的 EXIEN 控制位，使能相应 GPIO 的 EXI 功能
4) 配置 GPIO 的 IGRP，选择 EXI 触发事件的信号源。
5) 设置 SYSCON 内的 EXIRT 或 EXIFT 选择触发信号的边沿类型。
6) 首先通过 EXICR 清除一次由于配置不同的边沿过程中导致的中断 pending
7) 设置 EXIER 打开相应的 EXI 中断。

6.2.7.2 外部中断的滤波控制

在外部中断输入通路，具有两种滤波模块，一种为模拟型滤波器，可以滤除 30ns 以下的脉冲毛刺信号，还有一种为由 ISOSC 的时钟同步的数字滤波器。数字滤波可以通过 OPT1[EXIFLTEN]控制进行使能选择。在 DEEP-SLEEP 模式下，若 ISOSC 没有被使能（GCER[STP_ISO]，且在应用中设置为通过 EXI 唤醒低功耗模式，用户必须在进入低功耗模式前，将数字滤波器禁止。以保证在 EXI 通路上没有时钟同步滤波逻辑。

当数字滤波器被使能，外部输入的 EXI 触发信号，通过模拟滤波后，在数字滤波器内通过 ISOSC 的时钟对信号进行采样并进行数字去抖处理。去抖的时钟可以通过 OPT1[EXIFLTCKS]进行设置。去抖深度为 3 个采样周期。

6.2.8 内存可靠性监测

片上内存包括 Flash 和 SRAM 两种类型的存储单元。当数据从存储单元中被读取时，由于供电电压影响，或者信号干扰导致读取的目标数据和实际存储数据不一致时，将造成系统错误。例如由于从 Flash 读取到的代码错误，而导致 CPU 运行错误的指令，最终导致系统失效。

为提高系统的可靠性，在内存控制器单元中，增加了额外的硬件校验电路，并且在存储器中有额外的校验数据存取区间。当数据从系统写入到存储单元中时，通过硬件校验电路生成的校验码，同时会被写入到对应的

校验码区。当数据读取时，内存控制器会对读取的数据和相应的校验码做硬件校验，校验合格后，数据才会被送入系统总线，若校验失败，控制器会根据设置进行重试；当重试计数达到预设值时，系统将会根据 RAMCHK 和 EFLCHK 的设置采取相应的行动。

当内存校验错误发生后，但是通过重试，获得正确数据后继续工作时，系统不会产生复位，但 SYSCON 的 MEMERR 中断标志位将置起，当中断使能时，可以通过中断的方式通知系统。

缺省条件下，校验功能为禁止状态。Flash 的校验功能可以通过 USER Option 设置缺省使能或者禁止状态。

6.2.9 独立看门狗

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 12 位的 Free Running 递减计数器控制定时时间。独立看门狗和运行模式无关，一旦使能则必须在计数器溢出前进行清除，否则会产生系统复位。

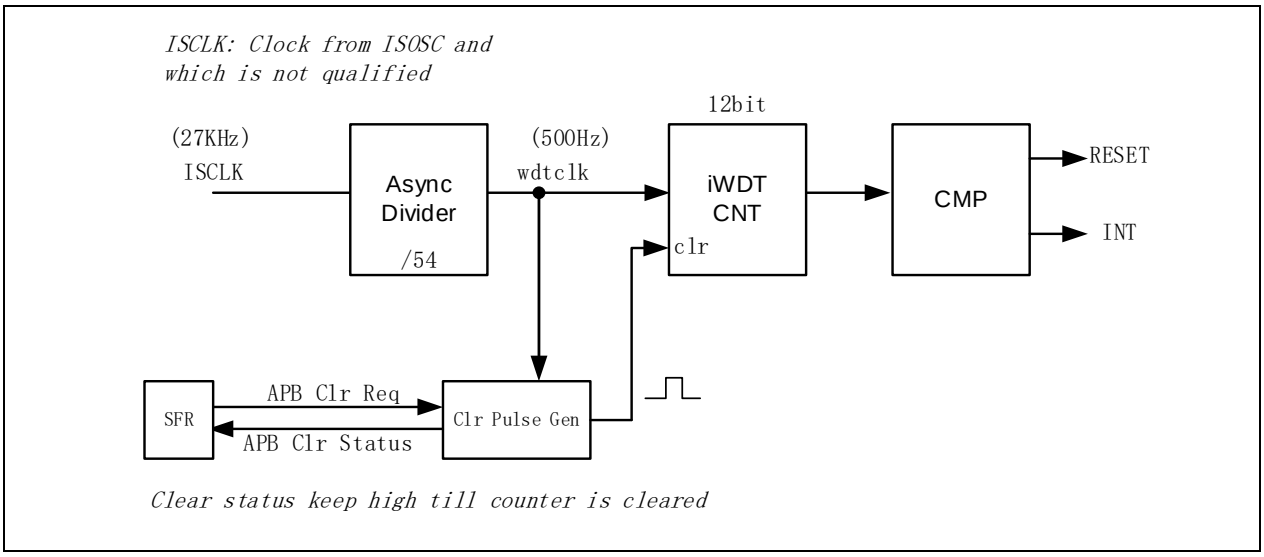


Figure 7-10 IWDT 结构框图

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[CLR] 写入 0x5A 实现。只有当清除 CNT 操作发生时，计数器值才会被更新到最新的设置值，所以在更新了 IWDCNT 寄存器后，需要软件清除一次 CNT，使得内部计数器及时更新到最新的配置。软件清除需要在 IWDT 启动以后执行（通过查询 IWDCR[BUSY]控制位确认 IWDT 是否已经启动）。

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 OVTIME 位设置。OVTIME 一共为 3 位，对应 8 种定时时间设置。最小定时时间为 128ms。

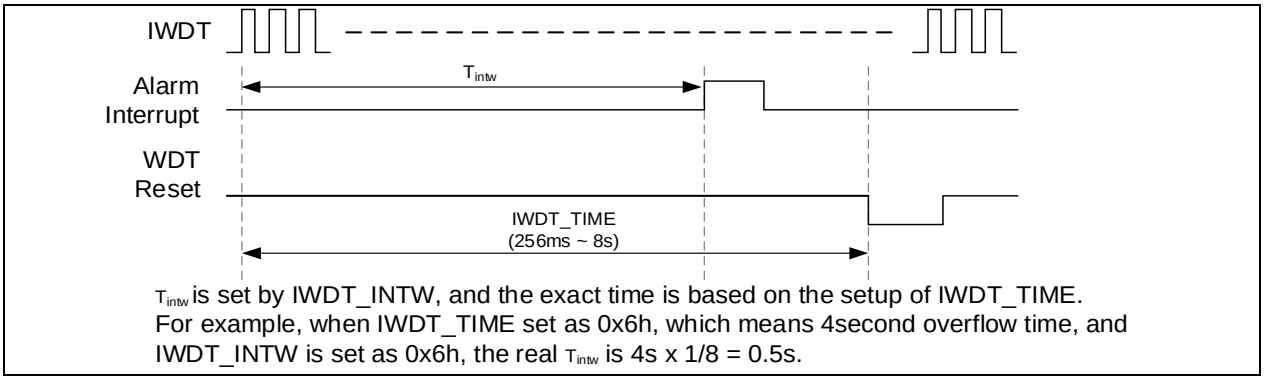


Figure 7-11 IWDT 计数器溢出和报警中断

IWDT 还支持报警功能。在计数器计数过程中，当计数值达到 IWDCR[INTERVAL]设置值时，会产生一个中断信号。IWDCR[INTERVAL]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 IWDCR[OVTIME]设置看门狗定时溢出时间为 4 秒，如果 IWDCR[INTERVAL]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。通过此报警中断可以在低功耗模式下及时唤醒系统并进行计数器清除，以防止看门狗复位。

6.2.10 IO重定义

为提供更灵活的 IO 功能配置，系统提供了自定义 GPIO 复用的功能。芯片提供两个预设的 GPIO GROUP，分别为 GROUP0 和 GROUP1，两个 GROUP 分别对应 8 个预设的可选择的复用功能。在每个 GROUP 内，每个 GPIO 可以被指定为这 8 个预设功能中的任意一个作为该 GPIO 的 AF7 功能。

如下表所示，IO GROUP0 中包含 8 个 GPIO（PA0.0~PA0.7）；IO GROUP1 中包含 8 个 GPIO（PB0.2，PB0.3，PA0.8~PA0.13）。这些 GPIO 的 AF7 功能是由 IOMAP0 或 IOMAP1 中相应的 CFGVAL 决定的。

Table 7-10 IOGROUP0中的 GPIO

GPIO	CFGVAL in IOMAP0
PA0.0	CFGVAL0
PA0.1	CFGVAL1
PA0.2	CFGVAL2
PA0.3	CFGVAL3
PA0.4	CFGVAL4
PA0.5	CFGVAL5
PA0.6	CFGVAL6
PA0.7	CFGVAL7

Table 7-11 IOGROUP1中的 GPIO

GPIO	CFGVAL in IOMAP1
PB0.2	CFGVAL0
PB0.3	CFGVAL1
PA0.8	CFGVAL2
PA0.9	CFGVAL3
PA0.10	CFGVAL4
PA0.11	CFGVAL5
PA0.12	CFGVAL6
PA0.13	CFGVAL7

IOMAP0/1 中 CFGVAL 提供了下表所示的配置选择。例如，当 IOMAP0 中的 CFGVAL0 = 0 时，PA0.0 （参考表 7-10）的 AF7 功能将是 I2C_SCL。当 IOMAP1 中的 CFGVAL3 = 4 时，PA0.9（参考表 7-12）的 AF7 功能将是 EPT_CHCX。

Table 7-12 IO REMAP

AF Function	CFGVAL in IOMAP	GROUP
I2C_SCL / UART0_RX	0x00	GROUP0 / GROUP1
I2C_SDA / UART0_TX	0x01	GROUP0 / GROUP1
GPT_CHA / EPT_CHAX	0x02	GROUP0 / GROUP1
GPT_CHB / EPT_CHBX	0x03	GROUP0 / GROUP1
SPI_MOSI / EPT_CHCX	0x04	GROUP0 / GROUP1
SPI_MISO / EPT_CHAY	0x05	GROUP0 / GROUP1
SPI_SCK / EPT_CHBY	0x06	GROUP0 / GROUP1
SPI_NSS / EPT_CHCY	0x07	GROUP0 / GROUP1

6.2.11 系统信息及自定义寄存器

系统中存在几类信息存储寄存器，作为软件获取系统信息的一种途径。

UID：唯一序列码

每个芯片在出厂测试时，都会设置唯一的序列号，该序列号由 96bit 组成。

UREG：自由寄存器

用于保存客户临时信息的寄存器，该寄存器内保存的数据只有在上电复位后才会丢失。这意味着，只要供

电在，无论任何其他的复位都不会改变其中的数据。该寄存器无特殊功能，仅为用户提供在应用程序中可以方便地存储和监测不受其他复位影响的临时状态。

6.2.12 SYSCON 中断管理

系统控制器的中断由 6 中断源组成，每个中断源下可选择一个或者多个触发事件作为该中断源的触发信号。

一个中断源是由 SYSCON 通用事件产生的通用中断请求源，其控制通过 RISR，MISR，IMCR，IMDR, IMER, ICR 这组控制寄存器进行控制。触发 SYSCON 中断的事件使能选择可以通过 IMCR 或者 IMER/IMDR 寄存器进行设置。IAR 寄存器则提供了通过软件触发中断事件的方法，可用于程序代码中对事件的 debug。

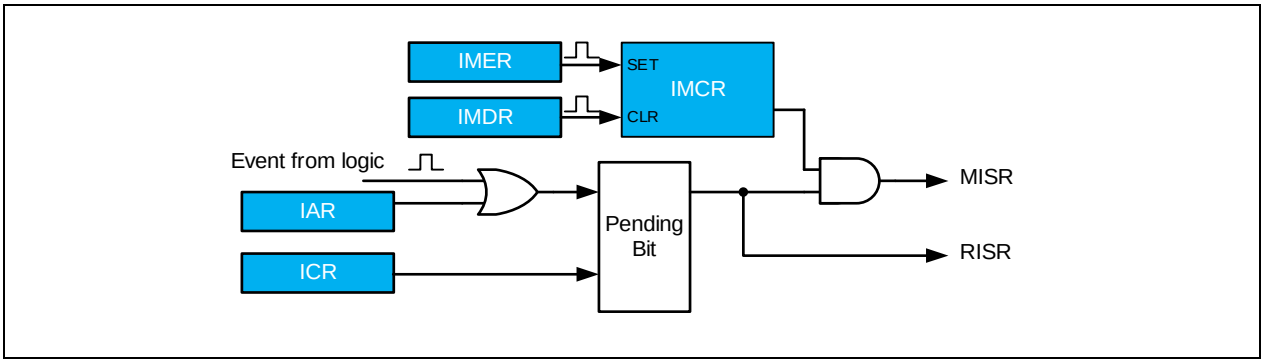


Figure 7-12 SYSCON 通用中断的控制逻辑

当系统控制器的中断产生时，应用程序将通过 CPU 的相应中断向量进行处理。对中断的标志位进行清除时，需要通过软件写 ICR 寄存器进行清除。支持的中断事件可以参考下表中的描述。

Table 7-13 SYSCON General Event to Trigger Interrupt

Trigger Event	Description
ISOSC_ST	Asserted when ISOSC is stabled
IMOSC_ST	Asserted when IMOSC is stabled
EMOSC_ST	Asserted when EMOSC is stabled
HFOSC_ST	Asserted when HFOSC is stabled
SYSCLK_ST	Asserted when SYSCLK is stabled
IWDT_INT	Asserted when IWDT counter reaches preset interval
RAM_ERR	Asserted when SRAM read attempt exceeds preset retry time
LVD_INT	Asserted when power supply falls or rises to preset LVD level
HWD_ERR	Asserted when divisor is zero
EFL_ERR	Asserted when flash read attempt exceeds preset retry time
OPL_ERR	Asserted when User Option fails to load at initialization

EM_CMFAIL	Asserted when external oscillator monitoring detects failure
CMD_ERR	Asserted when register operation is incorrect.

另外 5 个中断源为外部中断，包括 EXI0，EXI1，EXI2TO3，EXI4TO。

6.2.13 触发事件控制

外部中断事件可以作为片内模块间的触发信号，用户通过 ETCB 配置可以选择不同的触发事件对芯片内部其他模块进行动作触发。SYSCON 支持 6 个触发源输出端口，可以支持同时六个通道的同步事件触发。通过外部触发，可以实现一些典型的触发应用，例如通过配置特定 GPIO，使能 EXI 功能后：

- 可以通过特定 IO 的外部中断事件触发 TIMER 的捕获操作。
- 通过特定 IO 的外部中断事件触发 ADC 进行数据采集。
- 通过特定 IO 的外部中断事件触发 PWM 的 One Pulse 输出。

信号在模块间的流动如下图所示:

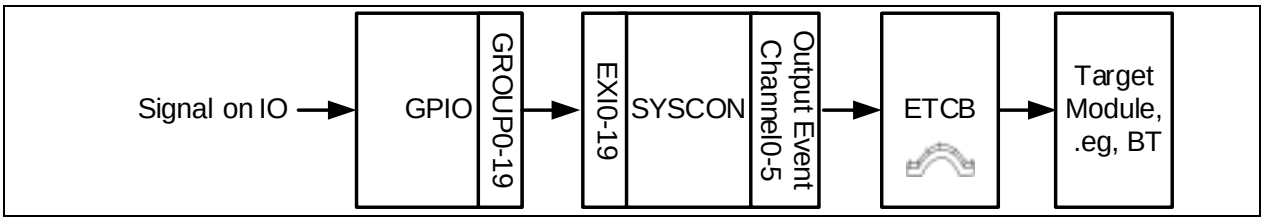


Figure 7-13 Signal Flow

外部触发通道的触发源选择通过 EVTRG 寄存器进行配置。只有 EVTRG[TRGxOE]的相应控制位被使能时，当前触发通道的触发信号才能被 ETCB 检查到。

通道 0~通道 3 支持事件计数功能，通道 4 和通道 5 不支持事件计数。每次检测到外部中断事件时，当前触发端口的事件计数器将自加一次。当计数器值等于 EVPS 的设置值时，下一次触发事件将使能一次触发信号到 ETCB，并清除当前的事件计数器值。例如，当 EVPS[TRGEV0PRD] = 3，则 GPIO 收到第四次触发事件时将产生一次触发事件到 ETCB，并自动清除计数器。即每间隔 3 次外部中断事件，产生一次触发事情到 ETCB。当计数器周期 EVPS 被设置为零时，计数器不使能，即每次事件都会流出事件通道到达 ETCB。

通过 EVSWF 寄存器可以通过软件强制产生一次触发事件到 ETCB。在有事件计数条件下，软件产生的触发事件对事件计数器进行递增操作。

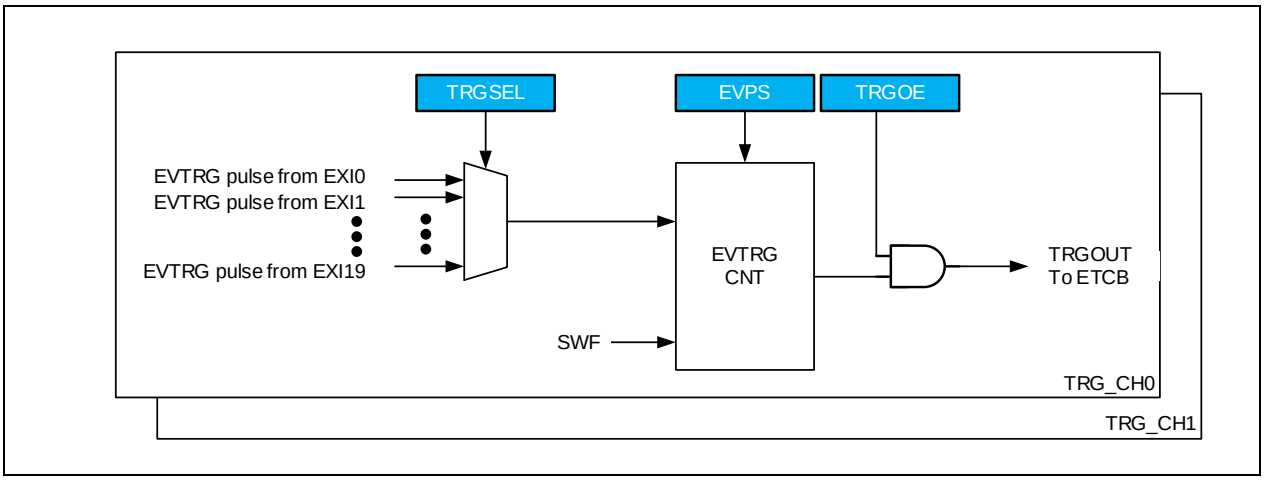


Figure 7-14 SYSCON 外部事件触发控制逻辑

6.3 寄存器说明

- Base Address: 0x4001_1000

6.3.1 寄存器列表

Register	Offset	Description	Reset Value
SYSCON_IDCCR	0x000	ID 和控制器模块时钟控制寄存器	0xFFFF_FF01
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	0x0008_1103
SYSCON_CKST	0x010	时钟状态寄存器	0x0000_0103
SYSCON_RAMCHK	0x014	SRAM 校验控制寄存器	0x0000_FFFF
SYSCON_EFLCHK	0x018	Embedded Flash 校验控制寄存器	0x00FF_FFFF
SYSCON_SCLKCR	0x01C	系统时钟控制寄存器	0x0000_0100
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	0x0010_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	0x70FF_3BFF
RSVD	-	0x044，0x048 保留	-
SYSCON_LVDCR	0x04C	低电压检测控制寄存器	0x0000_000A
SYSCON_CLCR	0x050	内部振荡器调整寄存器	0xXXXX_X100
SYSCON_PWRCR	0x054	功耗控制寄存器	0x141F_1F00
SYSCON_PWRKEY	0x058	功耗调整使能寄存器	0x0000_0000
RSVD	-	0x05C，0x060 保留	-
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]	0x0000_0000-
SYSCON_WKCR	0x06C	低功耗唤醒使能控制寄存器	0x0000_0000
RSVD	-	0x070 保留	0x0000_0000

Register	Offset	Description	Reset Value
SYSCON_IMER	0x074	中断使能控制寄存器	0x0000_0000
SYSCON_IMDR	0x078	中断禁止控制寄存器	0x0000_0000
SYSCON_IMCR	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_MISR	0x08C	中断标志状态寄存器	0x0000_0000
SYSCON_RSR	0x090	复位源记录状态寄存器	-
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	0x0000_0000
SYSCON_EXIRS	0x0B0	外部中断原始标志状态寄存器	0x0000_0000
SYSCON_IWDCR	0x0B4	独立看门狗控制寄存器	0x0000_070C
SYSCON_IWDCNT	0x0B8	独立看门狗控制计数器值	0x0003_FFFF
SYSCON_IWDEDR	0x0BC	独立看门狗使能寄存器	0x0000_XXXX
SYSCON_IOMAP0	0x0C0	GPIO 分组 0 的功能映射配置寄存器	0x0000_0000
SYSCON_IOMAP1	0x0C4	GPIO 分组 1 的功能映射配置寄存器	0x0000_0000
RSVD	-	0x0C8~0x0E0 保留	-
SYSCON_UID0	0x0E4	UID 寄存器 0	-
SYSCON_UID1	0x0E8	UID 寄存器 1	-
SYSCON_UID2	0x0EC	UID 寄存器 2	-
SYSCON_PWROPT	0x0F0	供电恢复时间调整寄存器	0x0000_4040
SYSCON_EVTRG	0x0F4	事件触发选择寄存器	0x0000_0000
SYSCON_EVPS	0x0F8	事件触发计数寄存器	0x0000_0000
SYSCON_EVSWF	0x0FC	事件计数器软件触发控制寄存器	0x0000_0000
SYSCON_UREG0	0x100	32 位用户寄存器	0x0000_0000
SYSCON_UREG1	0x104	32 位用户寄存器	0x0000_0000

Register	Offset	Description	Reset Value
SYSCON_UREG2	0x108	16 位用户寄存器	0x0000_0000
SYSCON_UREG3	0x10C	16 位用户寄存器	0x0000_0000

6.3.2 SYSCON_IDCCR（ID和时钟控制寄存器）

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
IDCODE/ID_KEY																								SWRST	RSVD		CPUFTRST				CLKEN
																											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	W	R	R	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CLKEN	[0]	R/W	使能或禁止 SYSCON 模块的 PCLK 时钟。 0h：禁止 SYSCON 模块的时钟。 1h：使能 SYSCON 模块的时钟。
CPUFTRST	[4:1]	R/W	CPU Fault 发生时硬件触发系统复位使能控制位。（可由 User Option 加载复位缺省值） Other：禁止 CPU Fault 时硬件触发复位。 Ah： 使能 CPU Fault 时硬件触发复位。
SWRST	[7]	W	SYSCON 产生软件复位。效果和 CPU 通过软复位指令产生软件复位一致。 0h：没有效果。 1h：执行软件复位。
IDCODE	[31:8]	R	读取时，返回 ID CODE（IP 版本信息）
ID_KEY	[31:16]	W	对当前寄存器进行写入时，必须同时写入正确 KEY 值。只有在 ID_KEY 等于 0xE11E 时，写入才有效。

6.3.3 SYSCON_GCER/GCDR（通用使能/禁止控制寄存器）

Address = Base Address + 0x0004/0x0008, 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												EMO_CMRST	EMO_CKM	RSVD	RSVD	LP_EMOEN	RSVD	LP_IMOEN	LP_ISOEN	SYSTICK	RSVD	IDLE_HCLK	IDLE_PCLK	RSVD	RSVD	RSVD	HFOSC	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	W	R	W	W	W	R	W	W	R	R	R	W	W	W	W	W

Name	Bit	Type	Description
ISOSC	[0]	W	使能或禁止 ISOSC 内部振荡器（缺省使能）。 0h：无效。 1h：使能或禁止指定振荡器。
IMOSC	[1]	W	使能或禁止 IMOSC 内部振荡器（缺省使能）。 0h：无效。 1h：使能或禁止指定振荡器。
EMOSC	[3]	W	使能或禁止 EMOSC 外部振荡器（XIN 和 XOUT 管脚功能必须已经在 GPIO 中进行预先配置）。 0h：无效。 1h：使能或禁止指定振荡器。
HFOSC	[4]	W	使能或禁止 HFOSC 内部振荡器（缺省禁止）。 0h：无效。 1h：使能或禁止指定振荡器。
IDLE_PCLK	[8]	W	使能或禁止 SLEEP 模式下的 PCLK（缺省使能）。 0h：无效。 1h：使能或禁止指定功能。
IDLE_HCLK	[9]	W	使能或禁止 SLEEP 模式下的 HCLK（缺省禁止）。 0h：无效。

			1h：使能或禁止指定功能。
SYSTICK	[11]	W	使能或禁止 CORET 的计数时钟（缺省禁止）。 0h：无效。 1h：使能或禁止指定功能。
LP_ISOEN	[12]	W	使能或禁止 DEEP-SLEEP 模式下 ISOSC 工作状态（缺省禁止，若 IWDT 使能，则该控制位设置无效）。 0h：无效。 1h：使能或禁止 ISOSC 在 DEEP-SLEEP 模式下工作。
LP_IMOEN	[13]	W	使能或禁止 DEEP-SLEEP 模式下 IMOSC 工作状态（缺省禁止）。 0h：无效。 1h：使能或禁止 IMOSC 在 DEEP-SLEEP 模式下工作。
LP_EMOEN	[15]	W	使能或禁止 DEEP-SLEEP 模式下 EMOSC 工作状态（缺省禁止）。 0h：无效。 1h：使能或禁止 EMOSC 在 DEEP-SLEEP 模式下工作。
EMO_CKM	[18]	W	使能或禁止外部 EMOSC 监测功能（缺省禁止）。 0h：无效。 1h：使能或禁止 EMOSC 监测。
EMO_CMRST	[19]	W	在 EMO_CKM 功能使能后，使能或禁止外部 EMOSC 失效复位。（缺省使能） 0h：无效。 1h：使能或禁止 EMOSC 失效产生系统复位。当禁止产生系统复位时，EMOSC 失效后，会自动切换系统时钟到 IMOSC 上。

6.3.4 SYSCON_GCSR（通用状态寄存器）

Address = Base Address + 0x000C, Reset Value = 0x0008_1103

31	30	29	28	27	26	25	24	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												EMO_CMRST	EMO_CKM	RSVD	RSVD	LP_EMOEN	RSVD	LP_IMOEN	LP_ISOEN	SYSTICK	RSVD	IDLE_HCLK	IDLE_PCLK	RSVD	RSVD	RSVD	HFOSC	EMOSC	RSVD	IMOSC	ISOSC
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	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 内部振荡器工作状态。 0h：ISOSC 被关闭。 1h：ISOSC 被打开。
IMOSC	[1]	R	IMOSC 内部振荡器工作状态。 0h：IMOSC 被关闭。 1h：IMOSC 被打开。
EMOSC	[3]	R	EMOSC 振荡器工作状态。 0h：EMOSC 被关闭。 1h：EMOSC 被打开。
HFOSC	[4]	R	HFOSC 内部振荡器工作状态。 0h：HFOSC 被关闭。 1h：HFOSC 被打开。
IDLE_PCLK	[8]	R	SLEEP 模式下 PCLK 状态。 0h：SLEEP 模式下 PCLK 被关闭。 1h：SLEEP 模式下 PCLK 被打开。
IDLE_HCLK	[9]	R	SLEEP 模式下 HCLK 状态。 0h：SLEEP 模式下 HCLK 被关闭。 1h：SLEEP 模式下 HCLK 被打开。

SYSTICK	[11]	R	CORET 的时钟工作状态。 0h：CORET 时钟被关闭。 1h：CORET 时钟被打开。
LP_ISOEN	[12]	R	DEEP-SLEEP 模式下 ISOSC 工作状态。 0h：ISOSC 被关闭。 1h：ISOSC 被打开。
LP_IMOEN	[13]	R	DEEP-SLEEP 模式下 IMOSC 工作状态。 0h：IMOSC 被关闭。 1h：IMOSC 被打开。
LP_EMOEN	[15]	R	DEEP-SLEEP 模式下 EMOSC 工作状态。 0h：EMOSC 被关闭。 1h：EMOSC 被打开。
EMO_CKM	[18]	R	EMOSC Clock Monitor 功能状态。 0h：EMO CKM 被关闭。 1h：EMO CKM 被打开。
EMO_CMRST	[19]	R	EMOSC Clock Fail 时，产生系统复位。 0h：禁止产生系统复位。 1h：使能产生系统复位。

6.3.5 SYSCON_CKST（时钟稳定状态寄存器）

Address = Base Address + 0x0010, Reset Value = 0x0000_0103

31	30	29	28	27	26	25	24	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								SYS_CLK			RSVD	HFOSC	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	1	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 内部振荡器稳定状态。 0h：ISOSC 时钟未稳定。 1h：ISOSC 时钟已稳定。
IMOSC	[1]	R	IMOSC 内部振荡器稳定状态。 0h：IMOSC 时钟未稳定。 1h：ISOSC 时钟已稳定。
EMOSC	[3]	R	EMOSC 振荡器稳定状态。 0h：EMOSC 时钟未稳定。 1h：EMOSC 时钟已稳定。
HFOSC	[4]	R	HFOSC 内部振荡器稳定状态。 0h：HFOSC 时钟未稳定。 1h：HFOSC 时钟已稳定。
SYS_CLK	[8]	R	SYS_CLK （系统时钟）稳定状态。 0h：SYS_CLK 时钟未稳定。 1h：SYS_CLK 时钟已稳定。

NOTE: 1）时钟源从关闭到打开的状态切换后，存在一段时钟稳定时间。在未稳定前，该时钟的稳定状态为未稳定状态。时钟源未稳定时，不能将系统时钟切换到该指定时钟源。

6.3.6 SYSCON_RAMCHK（SRAM校验控制寄存器）

Address = Base Address + 0x0014, Reset Value = 0x0000_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
CHKEN								RSTEN								CHKTIMES															
0	0	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
RW	RW	RW	RW	RW	RW	RW	RW	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
CHKTIMES	[15:0]	R/W	校验重试次数设置。 设置 SRAM 校验错后的重试次数。如果 SRAM 校验发生错误，SRAM 控制器会再次重读 SRAM 进行校验，重读的次数由该寄存器设置。连续重读次数满足该寄存器设置的次数后，如果仍然失败，那么会产生 SRAM 校验错的中断，或者产生系统复位(由 RSTEN 位控制)。
RSTEN	[23:16]	R/W	SRAM 校验失败复位控制位。在初次校验失败后，系统会自动重试（RE-READ），当重试次数达到 CHKCNT 设置值时，校验仍未通过，则系统认为校验失败。 5Ah：SRAM 校验失败后复位。 其他：SRAM 校验失败后不复位，只产生中断。
CHKEN	[31:24]	R/W	SRAM 校验使能控制位。 5Ah：使能 SRAM 的奇偶校验功能。 其他：关闭 SRAM 的奇偶校验功能。

6.3.7 SYSCON_EFLCHK（EFLASH校验控制寄存器）

Address = Base Address + 0x0018, Reset Value = 0x00FF_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
CHKEN								CHKTIMES																							
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	1	1	1	1	1	1
RW	RW	RW	RW	RW	RW	RW	RW	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
CHKTIMES	[23:0]	R/W	校验重试次数设置。 设置 Flash 校验错后的重试次数。如果 Flash 校验发生错误，Flash 控制器会再次重读当前地址，进行校验，重读的次数由该寄存器设置。重读次数满足该寄存器设置的次数后，如果仍然失败，系统会产生相应系统复位(由 CHKEN 位控制)。
CHKEN	[31:24]	R/W	Flash 校验使能控制位。 5Ah：使能 Flash 的校验功能。 其他：关闭 Flash 的校验功能。

NOTE: 1）该配置寄存器缺省复位值可以由 User Option 进行设置。

6.3.8 SYSCON_SCLKCR（系统时钟控制寄存器）

Address = Base Address + 0x001C, 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
SCLK_KEY																RSVD				SCLK_DIV				RSVD					SCLK_SEL		
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
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	R	R	R	R	R	RW	RW	RW

Name	Bit	Type	Description
SCLK_SEL	[2:0]	R/W	系统时钟源控制，选择当前系统时钟 SYSCLK 工作的时钟。 0h：IMOSC 作为系统工作时钟源。 1h：EMOSC 作为系统工作时钟源。 2h：HFOSC 作为系统工作时钟源。 4h：ISOSC 作为系统工作时钟源。 其他：保留
SCLK_DIV	[11:8]	R/W	系统时钟分频设置（SYSCLK 分频后，即为 HCLK 频率）。 0h：不分频。 1h：不分频。 2h：2 分频。 3h：3 分频。 4h：4 分频。 5h：5 分频。 6h：6 分频。 7h：9 分频。 8h：12 分频。 9h：16 分频。 Ah：24 分频。 Bh：32 分频。 Ch：36 分频。 Dh：64 分频。 Eh：128 分频。

			Fh：256 分频。

NOTE: 1）对该配置寄存器写入时，需要同时高位写入相应 SCLK_KEY，KEY 值不为 0xD22D 时，写入无效。

6.3.9 SYSCON_PCLKCR（外设时钟控制寄存器）

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
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
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	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 的时钟分频设置，PCLK 分频基于 HCLK 的频率。 0000B：不分频。 0001B：2 分频。 001xB：4 分频。 01xxB：8 分频。 1xxxB：16 分频。

NOTE: 1）对该配置寄存器写入时，需要同时高位写入相应 PCLK_KEY，KEY 值不为 0xC33C 时，写入无效。

6.3.10 SYSCON_PCER0/PCDR0（外设时钟使能控制寄存器0）

Address = Base Address + 0x0028/0x002C, 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	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	I2C	RSVD	SIO	RSVD	RSVD	RSVD	SPI	RSVD	RSVD	RSVD	RSVD	RSVD	UART2	UART1	UART0	ETCB	TOUCH	RSVD	ADC	RSVD	RSVD	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	W	R	R	R	W	R	R	R	R	R	W	W	W	W	W	R	W	R	R	R	W

Name	Bit	Type	Description
IFC ADC TOUCH UART0 UART1 UART2 SPI SIO I2C	[-]	W	使能或禁止相应模块的 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。 0h：无效。 1h：使能或禁止模块的 PCLK 时钟。

6.3.11 SYSCON_PCSR0（外设时钟状态控制寄存器0）

Address = Base Address + 0x0030, Reset Value = 0x0010_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	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	I2C	RSVD	SIO	RSVD	RSVD	RSVD	SPI	RSVD	RSVD	RSVD	RSVD	RSVD	UART2	UART1	UART0	ETCB	TOUCH	RSVD	ADC	RSVD	RSVD	RSVD	IFC
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	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 TOUCH UART0 UART1 UART2 SPI SIO I2C	[-]	R	相应外设模块的 PCLK 时钟的使能/禁止状态。 0h：对应模块的时钟处于关闭状态。 1h：对应模块的时钟处于打开状态。

6.3.12 SYSCON_PCER1/PCDR1（外设时钟使能控制寄存器1）

Address = Base Address + 0x0034/0x0038, 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	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	EPT	RSVD	RSVD	RSVD	GPTA	BT1	BT0	CNTA	LPT	RTC	WWDT	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	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	W	R	R	R	W	W	W	W	W	W	W	R	R	R	R	R	R	R

Name	Bit	Type	Description
WWDT RTC LPT CNTA BT0 BT1 GPTA EPT	[-]	W	使能或禁止相应模块的 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。 0h：无效。 1h：使能或禁止模块的 PCLK 时钟。

6.3.13 SYSCON_PCSR1（外设时钟状态控制寄存器）

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
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	EPT	RSVD	RSVD	RSVD	GPTA	BT1	BT0	CNTA	LPT	RTC	WWDT	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	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
WWDT RTC LPT CNTA BT0 BT1 GPTA EPT	[-]	R	相应外设模块的 PCLK 时钟的使能/禁止状态。 0h：对应模块的时钟处于关闭状态。 1h：对应模块的时钟处于打开状态。

6.3.14 SYSCON_OSTR（外部晶振稳定控制寄存器）

Address = Base Address + 0x0040, Reset Value = 0x70FF_3BFF

31	30	29	28	27	26	25	24	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_FLTSEL		EM_FLTEN	RSVD									EM_GMCTL					EM_LFSEL	EMCNT									
0	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1
R	R	R	R	RW	RW	RW	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
EMCNT	[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，同时 CKST 寄存器中的 EMOSC 状态位置位。 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
EM_LFSEL	[10]	R/W	外部晶振的低速模式设置。当外接 32.768KHz 晶振，需要将该位设置为使能。 0h：EMOSC 为普通模式。 1h：EMOSC 为低速模式。
EM_GMCTL	[15:11]	R/W	外部晶振的增益控制位。根据不同的震荡频率，选择适当的增益控制，频率越高，选择的 GM 值应越大。
EM_FLTEN	[25]	R/W	外部振荡器滤波使能控制位。在高噪声环境下，使能该滤波器可以防止时钟路径串入抖动信号造成系统工作错误。 0h：禁止滤波 1h：打开滤波
EM_FLTSEL	[27:26]	R/W	外部振荡器滤波范围选择。选择可滤除的信号间隔幅度。

			0h：小于 5ns 间隔脉冲滤波。 1h：小于 10ns 间隔脉冲滤波。 2h：小于 15ns 间隔脉冲滤波。 3h：小于 20ns 间隔脉冲滤波。

6.3.15 SYSCON_LVDCR（低电压检测控制寄存器）

Address = Base Address + 0x004C, Reset Value = 0x0000_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
LVD_KEY																LVDFLAG	RSTLVL			RSVD	INTLVL			INTPOL		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	1	0	1	0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	R	RW	RW	RW	W	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
LVDEN	[3:0]	R/W	使能/禁止 LVD 模块控制位。使能 LVD 模块后，当 VDD 电压低于 RSTLVL 设置值时，系统将产生低电压异常的复位。 0Ah：禁止 LVD 模块。 其他：使能 LVD 模块。
INTPOL	[7:6]	R/W	低电压检测中断触发的极性选择。 0h：无效。 1h：当电压下降到低于 LVDLVL(falling)设置时，触发中断。 2h：当电压升高到超过 LVDLVL(rising)设置时，触发中断。 3h：当电压下降到低于或者升高到超过 LVDLVL(both)时，触发中断。
INTLVL	[10:8]	R/W	低电压检测中断触发电压选择。 0h：2.4V 1h：2.1V 2h：2.7V 3h：3.0V 4h：3.3V 5h：3.6V 6h：3.9V 注：比较电压为 VDD 电压和选择的电压进行比较。

RSTLVL	[14:12]	R/W	低电压复位触发的电压选择。 0h：1.9V 1h：2.2V 2h：2.5V 3h：2.8V 4h：3.1V 5h：3.4V 6h：3.7V 7h：4.0V
LVDFLAG	[15]	R	LVD 的当前状态查询 0h：VDD 的当前电压高于 INTLVL 设置的检测阈值。 1h：VDD 的当前电压低于 INTLVL 设置的检测阈值。

NOTE: 1）对该配置寄存器写入时，需要同时高位写入相应 LVD_KEY，KEY 值不为 0xB44B 时，写入无效。

6.3.16 SYSCON_CLCR（内部振荡器调整控制寄存器）

Address = Base Address + 0x0050, Reset Value = 0xXXXX_X100

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

Name	Bit	Type	Description
HFO_TUNE	[8:0]	R/W	HFOSC 频率的微调设置。 通过 HFO_TUNE 控制位，可以精细调整 HFOSC 的输出频率。 0x1FF：标准输出频率（工程校准后的输出值）。 0x1FE：标准输出频率-大约 13KHz @ HFOSC=24MHz。 0x1FD：标准输出频率-大约 26KHz @ HFOSC=24MHz。 0x1FC：标准输出频率-大约 39KHz @ HFOSC=24MHz …….. 当 HFOSC 选择其他值时，步进频率等比例改变，即大约 26KHz@48MHz，大约 6.5KHz@12MHz。
HFO_TRM	[15:9]	R/W	HFOSC 的 TRIM 调整位 HFOSC 的频率输出随着调整值递增，值越大，频率越高。
IMO_TRM	[23:16]	R/W	IMOSC 的 TRIM 调整位 IMOSC 的频率输出随着调整值递增，值越大，频率越高。
ISO_TRM	[31:24]	R/W	ISOSC 的 TRIM 调整位 ISOSC 的频率输出随着调整值递增，值越大，频率越高。

6.3.17 SYSCON_PWRCR（功耗控制寄存器）

Address = Base Address + 0x0054, Reset Value = 0x141F_1F00

31	30	29	28	27	26	25	24	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			DSL_CFG					RSVD			SLP_CFG					RSVD			RUN_CFG					RSVD				VOSEN			
0	0	0	1	0	1	0	0	0	0	0	1	1	1	1	1	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0
R	R	R	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW	R	R	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
VOSEN	[3:0]	R/W	VOS 模式使能控制。(1) xxx1b：使能 RUN 模式下的 VOS xx1xb：使能 SLEEP 模式下的 VOS x1xxb：使能 DEEP-SLEEP 模式下的 VOS
RUN_CFG	[12:8]	R/W	在 RUN 模式下 VOS 控制，只有当 VOSEN[0]置位时有效。 Bit0：BGR 的配置位： 0h：BGR 被禁止 1h：BGR 被使能 Bit2-Bit1: 低功耗模式配置位： 0h：低功耗模式 0（LP0） 1h：低功耗模式 1（LP1） 2h：低功耗模式 2（LP2） 3h：RSVD Bit3：保留
SLP_CFG	[20:16]	R/W	在 SLEEP 模式下 VOS 控制，只有当 VOSEN[1]置位时有效。 Bit0：BGR 的配置位： 0h：BGR 被禁止 1h：BGR 被使能 Bit2-Bit1: 低功耗模式配置位：

			0h：低功耗模式 0（LP0） 1h：低功耗模式 1（LP1），亦为正常模式 2h：低功耗模式 2（LP2） 3h：RSVD Bit3: 保留
DSL_CFG	[28:24]	R/W	在 DEEP-SLEEP 模式下 VOS 控制，只有当 VOSEN[2]置位时有效。 Bit0：BGR 的配置位： 0h：BGR 被禁止 1h：BGR 被使能 Bit2-Bit1: 低功耗模式配置位： 0h：低功耗模式 0（LP0） 1h：低功耗模式 1（LP1） 其他：低功耗模式 2（LP2），亦为正常模式 Bit3: 保留

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）在各种模式下，BGR 的使能可独立配置。

3）不同的运行模式中，LP0~2 的低功耗策略各不相同。从 LP0 到 LP2 功耗递增，驱动能力也递增。调试的时候可以从最低功耗模式开始尝试，直到找到系统可以正常运行的模式。

6.3.18 SYSCON_PWRKEY（电压调整使能控制寄存器）

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
PWR_KEY																VOSLCK															
0	0	0	0	0	0	0	0	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
VOSLCK	[15:0]	R/W	VOS 全局使能控制。 只有在该控制器被配置为 0x6CC7 时，PWRCR 寄存器的配置才有效。

NOTE: 1）对该配置寄存器写入时，需要同时高位写入相应 PWR_KEY，KEY 值不为 0xA67A 时，写入无效。

6.3.19 SYSCON_OPT1（OPTION寄存器1）

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
RSVD								EMCKM_DUR			RSVD		EXIFLTCKS		EXIFLTEN	EFL_LPMD	CLODIV			CLOMX				RSVD		HFO_FEL		RSVD		IMO_FSEL	
0	0	0	0	0	0	0	0	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	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW	R	R	RW	RW

Name	Bit	Type	Description
IMO_FSEL	[1:0]	R/W	IMOSC 频率选择。 0h：5.556MHz。 1h：4.194MHz。 2h：2.097MHz。 3h：131.072KHz。
HFO_FSEL	[5:4]	R/W	HFOSC 频率选择。 0h：48MHz。 1h：24MHz。 2h：12MHz。 3h：6MHz。
CLOMX	[11:8]	R/W	CLO 输出选择。（CLO 管脚输出最高频率不超过 10MHz，若输出频率较高，可选择 CLODIV 分频后输出） 0h：ISCLK 输出。 1h：IMCLK 输出。 3h：EMCLK 输出。 4h：HFCLK 输出。 6h：RTCCLK 输出。 7h：PCLK 输出。 8h：HCLK 输出。 9h：IWDTCLK 输出。

			Dh：SYSCLK 输出。 其他：保留。
CLODIV	[14:12]	R/W	CLO 输出分频选择。 0h：4 分频。 1h：不分频。 2h：2 分频。 4h：8 分频。 5h：16 分频。 其他：4 分频。
EFL_LPMD	[15]	R/W	Flash 的 Low Power 模式选择。在此模式下，Flash 的读取周期保证大于 8us。当 HCLK 频率比较高时，需要配合选择合适的 WAIT CYCLE 来保证。 0h：禁止 Flash LP 模式。 1h：使能 Flash LP 模式。
EXIFLTEN	[16]	R/W	EXI 通道的数字滤波器使能控制。数字滤波的采用时钟为 ISCLK，当 ISCLK 没有使能时，该控制位无效。 0h：禁止数字滤波 1h：使能数字滤波
EXIFLTCKS	[18:17]	R/W	EXI 通道的数字滤波器检测频率设置。设置滤波器采用时钟的分频系数。 0h：不分频 1h：2 分频 2h：3 分频 3h：4 分频
EMCKM_DUR	[18:17]	R/W	EMCKM 的检测时间间隔设置，检测周期基于当前 IMOSC 的频率设置。 0h：18 个周期 1h：14 个周期 2h：10 个周期 3h：8 个周期 4h：6 个周期

			5h：5 个周期 6h：4 个周期 7h：3 个周期

6.3.20 SYSCON_OPT0（OPTION寄存器0）

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
				RDP										HDP_4K	HDP_ALL								DBP	CIPVALID					CPUFTRST	EXIRSTEN	IWDTEN
0	0	0	0	0	0	0	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
IWDTEN	[0]	R	独立看门狗 User Option 设置状态查询。 0h：缺省关闭看门狗。 1h：缺省打开看门狗。
EXIRSTEN	[1]	R	外部复位管脚功能 User Option 设置状态查询。 0h：外部复位管脚禁用。 1h：外部复位管脚使能。
CPUFTRST	[2]	R	CPU Fault 时产生硬件复位 User Option 设置状态查询。 0h：禁止自动复位。 1h：使能自动复位。
CIPVALID	[7]	R	烧录器加密通讯使能 User Option 状态查询。 0h：禁止通讯加密。 1h：使能通讯加密。
DBP	[8]	R	Debug Port 使能 User Option 状态查询。 0h：缺省打开 Debug Port。 1h：缺省关闭 Debug Port。
HDP_ALL	[16]	R	Hard Protection 使能 User Option 状态查询。 0h：缺省禁止 Hard Protection。 1h：缺省使能 Hard Protection。

HDP_4K	[17]	R	4K 空间 Hard Protection 使能 User Option 状态查询。 0h：缺省禁止 Hard Protection。 1h：缺省使能 Hard Protection。
RDP	[27]	R	Read Protection 使能 User Option 状态查询。 0h：缺省关闭 Read Protection。 1h：缺省打开 Read Protection。

6.3.21 SYSCON_WKCR（低功耗唤醒控制寄存器）

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
RSVD																			TCH_WKEN	LVD_WKEN	LPT_WKEN	RTC_WKEN	IWDT_WKEN	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	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
IWDT_WKEN	[8]	R/W	看门狗中断唤醒 DEEP-SLEEP 使能控制位。 0h：禁止 IWDT 中断唤醒 DEEP-SLEEP。 1h：使能 IWDT 中断唤醒 DEEP-SLEEP。
RTC_WKEN	[9]	R/W	RTC 中断唤醒 DEEP-SLEEP 使能控制位。 0h：禁止 RTC 中断唤醒 DEEP-SLEEP。 1h：使能 RTC 中断唤醒 DEEP-SLEEP。
LPT_WKEN	[10]	R/W	LPT 中断唤醒 DEEP-SLEEP 使能控制位。 0h：禁止 LPT 中断唤醒 DEEP-SLEEP。 1h：使能 LPT 中断唤醒 DEEP-SLEEP。
LVD_WKEN	[11]	R/W	LVD 中断唤醒 DEEP-SLEEP 使能控制位。 0h：禁止 LVD 中断唤醒 DEEP-SLEEP。 1h：使能 LVD 中断唤醒 DEEP-SLEEP。
TCH_WKEN	[12]	R/W	TOUCH 中断唤醒 DEEP-SLEEP 使能控制位。 0h：禁止 TCH 中断唤醒 DEEP-SLEEP。 1h：使能 TCH 中断唤醒 DEEP-SLEEP。

NOTE: EXI 始终可以唤醒低功耗模式，所以不存在 EXI 的 WKEN 控制。

6.3.22 SYSCON_IMER/IMDR/IAR/ICR（中断使能/禁止/软件触发/清除控制寄存器）

Address = Base Address + 0x0074/0x0078/0x0080/0x0084, 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						EV3TRG	EV2TRG	EV1TRG	EV0TRG	EM_CMFAIL	RSVD			OPL_ERR	EFL_ERR	HWD_ERR	LVD_INT	RAM_ERR	RSVD	IWDT_INT	SYSCLK_ST	RSVD		HFOSC_ST	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	W	W	W	W	W	R	R	R	W	W	W	W	W	R	W	W	R	R	W	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。
HFOSC_ST	[4]	W	HFOSC 时钟稳定中断。
SYSCLK_ST	[7]	W	SYSCLK 时钟稳定中断。
IWDT_INT	[8]	W	IWDT 中断。
RAM_ERR	[10]	W	SRAM 校验失败中断。
LVD_INT	[11]	W	LVD 中断。
HWD_ERR	[12]	W	硬件除法器除零中断。
EFL_ERR	[13]	W	EFLASH 校验失败中断。
OPL_ERR	[14]	W	Option 初始化配置加载失败中断。
EM_CMFAIL	[18]	W	EMOSC 时钟失效中断
EV0TRG	[19]	W	同步触发输出 Event0 触发的中断
EV1TRG	[20]	W	同步触发输出 Event1 触发的中断
EV2TRG	[21]	W	同步触发输出 Event2 触发的中断
EV3TRG	[22]	W	同步触发输出 Event3 触发的中断

CMD_ERR	[29]	W	命令错误中断，在对某些寄存器错误操作时会产生此中断。

6.3.23 SYSCON_IMCR（中断控制寄存器）

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		CMD_ERR	RSVD						EV3TRG	EV2TRG	EV1TRG	EV0TRG	EM_CMFAIL	RSVD			OPL_ERR	EFL_ERR	HWD_ERR	LVD_INT	RAM_ERR	RSVD	IWDT_INT	SYSCLK_ST	RSVD		HFOSC_ST	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	RW	R	R	R	R	R	R	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW	R	RW	RW	R	R	RW	RW	R	RW	RW

Name	Bit	Type	Description
ISOSC_ST	[0]	R/W	ISOSC 时钟稳定中断。 0h：禁止中断。 1h：使能中断。
IMOSC_ST	[1]	R/W	IMOSC 时钟稳定中断。 0h：禁止中断。 1h：使能中断。
EMOSC_ST	[3]	R/W	EMOSC 时钟稳定中断。 0h：禁止中断。 1h：使能中断。
HFOSC_ST	[4]	R/W	HFOSC 时钟稳定中断。 0h：禁止中断。 1h：使能中断。
SYSCLK_ST	[7]	R/W	SYSCLK 时钟稳定中断。 0h：禁止中断。 1h：使能中断。
IWDT_INT	[8]	R/W	IWDT 中断。 0h：禁止中断。 1h：使能中断。
RAM_ERR	[10]	R/W	SRAM 校验失败中断。 0h：禁止中断。 1h：使能中断。

LVD_INT	[11]	R/W	LVD 中断。 0h：禁止中断。 1h：使能中断。
HWD_ERR	[12]	R/W	硬件除法器除零中断。 0h：禁止中断。 1h：使能中断。
EFL_ERR	[13]	R/W	EFLASH 校验失败中断。 0h：禁止中断。 1h：使能中断。
OPL_ERR	[14]	R/W	Option 初始化配置加载失败中断。 0h：禁止中断。 1h：使能中断。
EM_CMFAIL	[18]	R/W	EMOSC 时钟失效中断。 0h：禁止中断。 1h：使能中断。
EV0TRG	[19]	R/W	同步触发输出 Event0 触发的中断。 0h：禁止中断。 1h：使能中断。
EV1TRG	[20]	R/W	同步触发输出 Event1 触发的中断。 0h：禁止中断。 1h：使能中断。
EV2TRG	[21]	R/W	同步触发输出 Event2 触发的中断。 0h：禁止中断。 1h：使能中断。
EV3TRG	[22]	R/W	同步触发输出 Event3 触发的中断。 0h：禁止中断。 1h：使能中断。
CMD_ERR	[29]	R/W	命令错误中断，在对某些寄存器错误操作时会产生此中断。 0h：禁止中断。 1h：使能中断。

6.3.24 SYSCON_RISR/MISR（中断原始/状态标志寄存器）

Address = Base Address + 0x0088/0x008C, 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						EV3TRG	EV2TRG	EV1TRG	EV0TRG	EM_CMFAIL	RSVD			OPL_ERR	EFL_ERR	HWD_ERR	LVD_INT	RAM_ERR	RSVD	IWDT_INT	SYSCLK_ST	RSVD		HFOSC_ST	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。
HFOSC_ST	[4]	R	HFOSC 时钟稳定中断。
SYSCLK_ST	[7]	R	SYSCLK 时钟稳定中断。
IWDT_INT	[8]	R	IWDT 中断。
RAM_ERR	[10]	R	SRAM 校验失败中断。
LVD_INT	[11]	R	LVD 中断。
HWD_ERR	[12]	R	硬件除法器除零中断。
EFL_ERR	[13]	R	EFLASH 校验失败中断。
OPL_ERR	[14]	R	Option 初始化配置加载失败中断。
EM_CMFAIL	[18]	R	EMOSC 时钟失效中断
EV0TRG	[19]	R	事件触发输出 Event0 触发的中断
EV1TRG	[20]	R	事件触发输出 Event1 触发的中断
EV2TRG	[21]	R	事件触发输出 Event2 触发的中断
EV3TRG	[22]	R	事件触发输出 Event3 触发的中断

CMD_ERR	[29]	R	命令错误中断，在对某些寄存器错误操作时会产生此中断。

6.3.25 SYSCON_RSR（复位源记录寄存器）

Address = Base Address + 0x0090, 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																		WWDT	EFL_ERR	RAM_ERR	RSVD	CPUFAULT	SWRST	CPURST	EMCKM	RSVD	IWDT	RSVD	EXTRST	LVR	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	1
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	R	RW	RW	RW	R	R	RW	R	RW	RW	RW

Name	Bit	Type	Description
POR	[0]	R/W	POR 上电复位。
LVR	[1]	R/W	LVD 复位。
EXTRST	[2]	R/W	外部复位管脚复位。
IWDT	[4]	R/W	IWDT 复位。
EMCKM	[6]	R/W	EMOSC CKM Fail 复位。
CPURST	[7]	R/W	CPU 软件复位。
SWRST	[8]	R/W	SYSCON 产生软件复位。
CPUFAULT	[9]	R/W	CPU 异常自动复位。
RAM_ERR	[11]	R/W	SRAM 校验错误复位。
EFL_ERR	[12]	R/W	EFLASH 校验错误复位。
WWDT	[13]	R/W	WWDT 复位。

NOTE: 对相应位写入‘1’可以清除当前标志位。

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

Address = Base Address + 0x0094/0x0098, 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												EXI19	EXI18	EXI17	EXI16	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	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
EXI0 ~ EXI19	[19:0]	R/W	外部中断上升沿/下降沿使能控制。 0h：该边沿触发关闭。 1h：该边沿触发打开。

NOTE: 1）EXIRT 是上升沿选择寄存器，EXIFT 是下降沿选择寄存器。

2）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

6.3.27 SYSCON_EXIER/EXIDR/EXIMR（外部中断使能/禁止/状态寄存器）

Address = Base Address + 0x009C/0x00A0/0x00A4, 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												EXI19	EXI18	EXI17	EXI16	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	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
EXI0 ~ EXI19	[19:0]	R/W	外部中断使能控制。 EXIER 和 EXIDR 为只写寄存器。通过 EXIER 使能中断，EXIDR 禁止中断。只有对寄存器写入'1'时才有效，读取时返回总为'0'。 寄存器写入时： 0h：无效。 1h：打开/关闭该 EXI 中断源。 EXIMR 为只读寄存器，读取时返回当前中断使能状态。 读取时： 0h：中断处于关闭状态。 1h：中断处于打开状态。

6.3.28 SYSCON_EXIAR/EXICR/EXIRS（外部中断软件触发/清除/原始状态寄存器）

Address = Base Address + 0x00A8/0x00AC/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												EXI19	EXI18	EXI17	EXI16	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	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
EXI0 ~ EXI19	[19:0]	R/W	EXIAR 为只写寄存器，只有对寄存器写入'1'时才有效，读取时返回总为'0'。 寄存器写入时： 0h：无效。 1h：软件触发该 EXI 中断源。 EXICR 为读写寄存器。读取时返回当前中断 pending 标志。写入‘1’时，清除当前中断标志；写入‘0’时无效。 读取时： 0h：中断标志处于未 Pending 状态。 1h：中断标志处于 Pending 状态。 写入时： 0h：无效。 1h：清除该 EXI Pending 状态。 EXIRS 为只读寄存器，读取时返回当前中断原始标志状态。 读取时： 0h：中断处于未 Pending 状态。 1h：中断处于 Pending 状态。

6.3.29 SYSCON_IWDCR（独立看门狗控制寄存器）

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			BUSY	DBGEN	OVTIME			RSVD			INTVAL			RSVD	SHORT
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	RW	RW	RW	RW	R	R	R	RW	RW	RW	R	RW

Name	Bit	Type	Description
SHORT	[0]	R/W	SHORT 工作模式，该模式用于缩短 IWDT 的溢出时间。正常使用时，保持该位为零。 0h：禁止 SHORT 模式。 1h：使能 SHORT 模式。
INTVAL	[4:2]	R/W	看门狗报警中断的时间设置。当看门狗计数器计数到总溢出时间的一定比例时，产生报警中断。 0h：1/8 总溢出时间。 1h：2/8 总溢出时间。 2h：3/8 总溢出时间。 3h：4/8 总溢出时间。 4h：5/8 总溢出时间。 5h：6/8 总溢出时间。 6h：7/8 总溢出时间。 7h：7/8 总溢出时间。
OVTIME	[10:8]	R/W	总溢出时间设置。当看门狗计数器计数溢出时，产生复位。 0h：128ms。 1h：256ms。 2h：512ms。 3h：1.024s。 4h：2.048s。

			5h：3.096s。 6h：4.12s。 7h：8.196s。
DBGEN	[11]	R/W	调试使能控制。调试使能时，在 CPU 被调试器挂起时，时基计数器的计数时钟同时也被挂起。 0h：调试禁止 1h：调试使能
BUSY	[12]	R	看门狗工作状态。 0h：看门狗未使能。 1h：看门狗已使能。
IWDT_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 IWDT_KEY 等于 0x8778 时，对本寄存器的写入才有效

NOTE: 当 IWDT 工作时，ISOSC 缺省使能。任何尝试关闭 ISOSC 的操作都会触发命令错误中断。

6.3.30 SYSCON_IWDCNT（独立看门狗控制计数器值）

Address = Base Address + 0x00B8, Reset Value = 0x0000_3FFF

31	30	29	28	27	26	25	24	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	CLR							RSVD												CNT											
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	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
CNT	[11:0]	R/W	返回 IWDT 当前计数值。
CLR	[30:24]	W	看门狗计数器清除请求。 只写控制位，只有写入‘0x5A’时有效。
CLR_BUSY	[31]	R	看门狗清除请求执行状态。 0h：没有挂起的清除操作。 1h：当前清除正在执行。

6.3.31 SYSCON_IWDEDR（独立看门狗使能控制寄存器）

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
IWDTE_KEY																ENDIS															
0	0	0	0	0	0	0	0	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
ENDIS	[15:0]	R/W	IWDT 使能控制。 写入 0x55AA 时，关闭 IWDT。 写入其他值时，使能 IWDT。
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。

6.3.32 SYSCON_IOMAP0/IOMAP1（GROUP0/1 IOMAP寄存器）

Address = Base Address + 0x00C0/0x00C4, 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
CFGVAL7				CFGVAL6				CFGVAL5				CFGVAL4				CFGVAL3				CFGVAL2				CFGVAL1				CFGVAL0			
0	0	0	0	0	0	0	0	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
CFGVALx	[31:0]	R/W	IO GROUP 中对应 GPIO 的功能选择。 具体配置数值和对应 GPIO，参照 IO 重定义章节的表格。

6.3.33 SYSCON_UID0（唯一ID寄存器）

Address = Base Address + 0x00E4/0x00E8/0x00EC, 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
UID																															
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R	R	R	R	R	R	R	R	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
UID	[31:0]	R	唯一 ID 寄存器。 在出厂时，由工厂写入的唯一 ID 码。

6.3.34 SYSCON_PWROPT（供电恢复时间调整寄存器）

Address = Base Address + 0x00F0, Reset Value = 0x0000_4040

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

Name	Bit	Type	Description
TPWRCV_DSL	[7:0]	R/W	从 DEEPSLEEP 模式唤醒后的电源稳定时间调整。稳定时间计数器的工作时钟为 2MHz。
TPWRCV_SLP	[15:8]	R/W	从 SLEEP 模式唤醒后的电源稳定时间调整。稳定时间计数器的工作时钟为 2MHz。
EFL_PD	[17:16]	R/W	EFLASH 的电源控制。 当程序在 SRAM 中运行时，为降低功耗，可以临时将 EFLASH 的供电关闭。当写入‘11’时，关闭 EFLASH 的供电；当写入其他值时，打开 EFLASH 的供电。
EFLR_PD	[19:18]	R/W	EFLASH 的内部参考软件使能控制。 当 EFLASH 断电时，可以关闭 EFLASH 的参考源，以降低功耗。当写入‘11’时，关闭参考源的供电；当写入其他值时，打开参考源的供电。EFLASH 参考源必须在 EFLASH 断电后，才能关闭；同样在恢复时，需要先恢复参考源，然后再打开 EFLASH 供电。
EFLR_CTL	[21:20]	R/W	EFLASH 的内部参考源使能硬件控制策略设置。 0h：参考源不随模式改变切换供电开关。 1h：参考源在 SLEEP 模式下自动关闭。 3h：参考源在 SLEEP 模式、EFL_PD 或 EFLASH LP 模式下自动关闭。 2h：保留。

			EFLASH LP 模式通过 OPT1[EFL_LPMD]控制位设置。

NOTE: 1）对该配置寄存器写入时，需要同时高位写入相应 PWR_KEY，KEY 值不为 0xB6 时，写入无效。

6.3.35 SYSCON_EVTRG（事件触发选择寄存器）

Address = Base Address + 0x00F4, 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
CNT3CLR	CNT2CLR	CNT1CLR	CNT0CLR	RSVD		TRG5OE	TRG4OE	TRG3OE	TRG2OE	TRG1OE	TRG0OE	TRGSEL5		TRGSEL4		TRGSEL3				TRGSEL2				TRGSEL1				TRGSEL0			
0	0	0	0	0	0	0	0	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	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
TRGSEL0 TRGSEL1 TRGSEL2 TRGSEL3	[15:0]	R/W	TRGEVx 事件的触发源选择。 0h：选择 EXI0事件作为当前触发通道事件。 1h：选择 EXI1事件作为当前触发通道事件。 2h：选择 EXI2事件作为当前触发通道事件。 3h：选择 EXI3事件作为当前触发通道事件。 …… Fh：选择 EXI15事件作为当前触发通道事件。
TRGSEL4 TRGSEL5	[19:16]	R/W	TRGEVx 事件的触发源选择。 0h：选择 EXI16事件作为当前触发通道事件。 1h：选择 EXI17事件作为当前触发通道事件。 2h：选择 EXI18事件作为当前触发通道事件。 3h：选择 EXI19事件作为当前触发通道事件。
TRGxOE	[25:20]	R/W	外部触发端口 TRGOUTx 使能。 0h：禁止触发输出。 1h：允许触发输出。
CNTxCLR	[31:28]	R/W	TRGEVxCNT 软件清除 0h：无效 1h：EVxCNT 重置为零

6.3.36 SYSCON_EVPS（事件触发计数寄存器）

Address = Base Address + 0x00F8, 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
TRGEV3CNT				TRGEV2CNT				TRGEV1CNT				TRGEV0CNT				TRGEV3PRD				TRGEV2PRD				TRGEV1PRD				TRGEV0PRD			
0	0	0	0	0	0	0	0	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
TRGEVxPRD	[15:0]	R/W	TRGEVx 的事件计数器周期设置。 当 TRGEVx 事件发生时，TRGEVxCNT 递增一次，当 TRGEVxCNT 的计数值等于 TRGEVxPRD 设置周期时，产生 TRGEVx 触发事件。
TRGEVxCNT	[31:16]	R	当前 TRGEVx 事件计数器值。 读取时，返回当前事件计数器值。此计数器不能设置初始值，只能通过 EVTRG 的 CNTCLR 控制位进行清零。

6.3.37 SYSCON_EVSWF（事件计数器软件触发控制寄存器）

Address = Base Address + 0x00FC, 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																										EV5SWF	EV4SWF	EV3SWF	EV2SWF	EV1SWF	EV0SWF
0	0	0	0	0	0	0	0	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	W	W	W	W	W	W

Name	Bit	Type	Description
EVxSWF	[5:0]	W	软件产生一次 EVx 的触发 0h：写入‘0’无效 1h：软件产生一次触发

6.3.38 SYSCON_UREG0/1/2/3（用户寄存器）

Address = Base Address + 0x0100/0x104/0x108/0x10C, 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
UREG																															
0	0	0	0	0	0	0	0	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
UREG	[31:0]	R/W	用户寄存器。 可用于用户自定义信息临时保存的寄存器。此寄存器中的内容只有在 POR 复位时才会清除。

NOTE: 1）UREG0 和 UREG1 为 32 位寄存器，UREG2 和 UREG3 为 16 位寄存器。

7 事件触发控制器（ETCB）

7.1 概述

本章节描述事件触发控制器，该模块用来将一个IP的信息传递到另一个IP，可以有效的减少对CPU的中断请求，从而降低CPU的负载。

注：如果系列内芯片不具有某一外围，那它就不具备该外围的所有资源。具体参考芯片的数据手册。

7.1.1 特性

- 8个可配置的事件通道
  - 通道0支持多个源触发单个目标
  - 通道1-2支持单个源触发多个目标
  - 通道3-7只支持单个源触发单个目标
- 支持软件触发

7.2 功能描述

7.2.1 模块框图

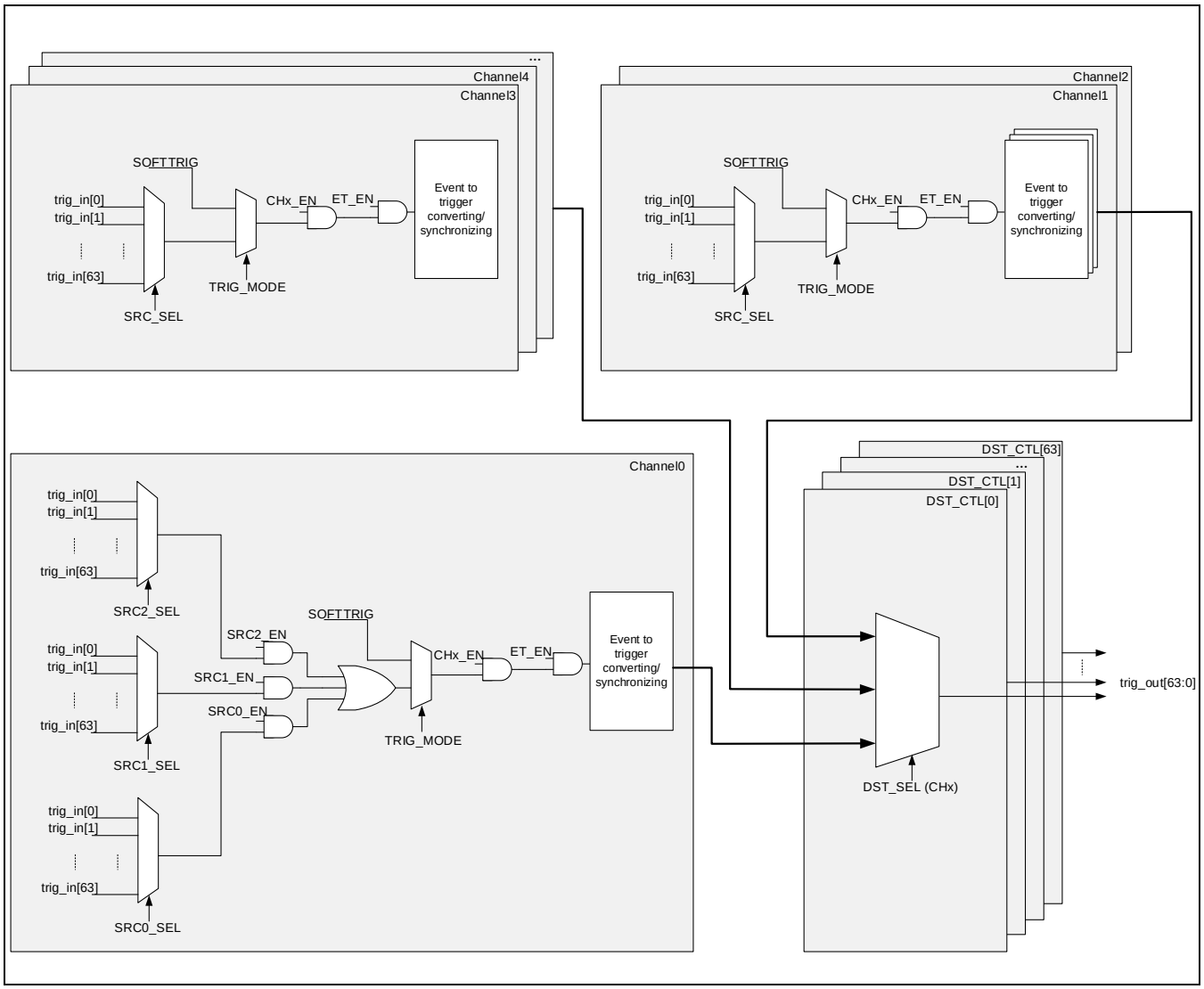


Figure 8-1 ETCB模块框图

7.2.2 主要功能

7.2.2.1 功能描述

事件触发控制器在收到一个 IP 的某个事件后，会触发另一个 IP 的相应动作。该模块能有效的减少 CPU 处理中断请求的时间，从而节省 CPU 的资源占用。例如，计时器的匹配事件可以配置成触发 ADC 的启动转换，这样每当计时器计数到特定数值时，ADC 会自动启动转换，不需要 CPU 的干涉。

该模块总共有 8 个通道。每个通道都可以由一个源来触发另一个目标。通道 0 可以由多个源触发一个目标，通道 1 和通道 2 则可以用一个源来触发多个目标。

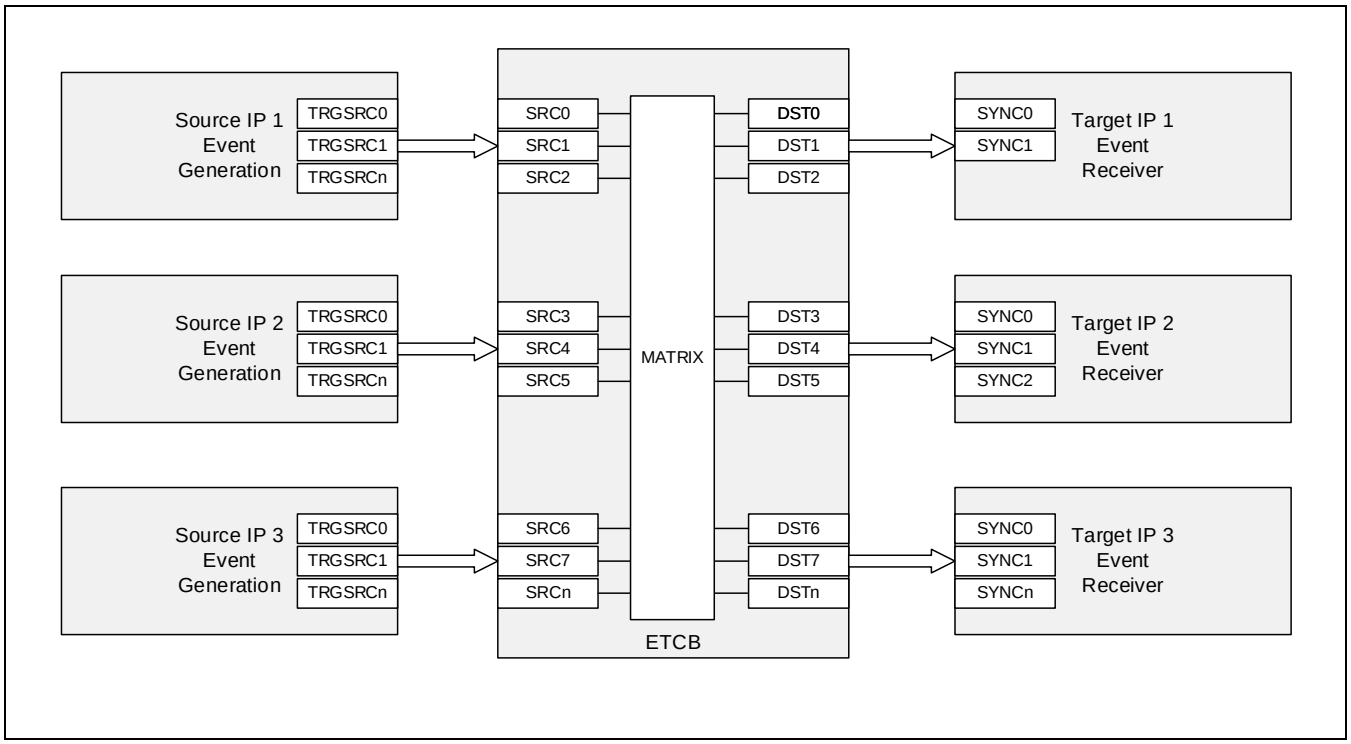


Figure 8-2 IP间通过ETCB事件同步

注意：

- 如果某个 IP 的事件源一直不停的以一个非常高的频率触发，那么 ETCB 模块有可能会丢失一部分触发信号。
- 事件源输出触发后，ETCB 模块需要一个时钟处理事件转发，即一个时钟后事件到目标模块。
- 一个目标只能被一个通道触发。如果 2 个或者多个通道都选择了同一个目标，那么序号小的通道有更高的优先级，因为占用目标。

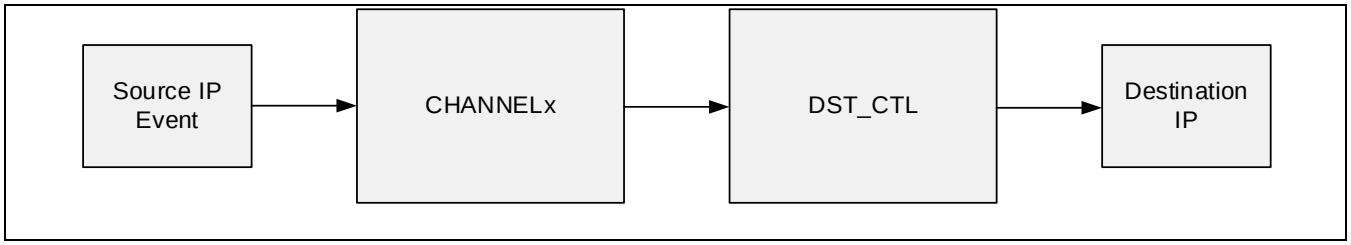


Figure 8-3 单个源触发单个目标

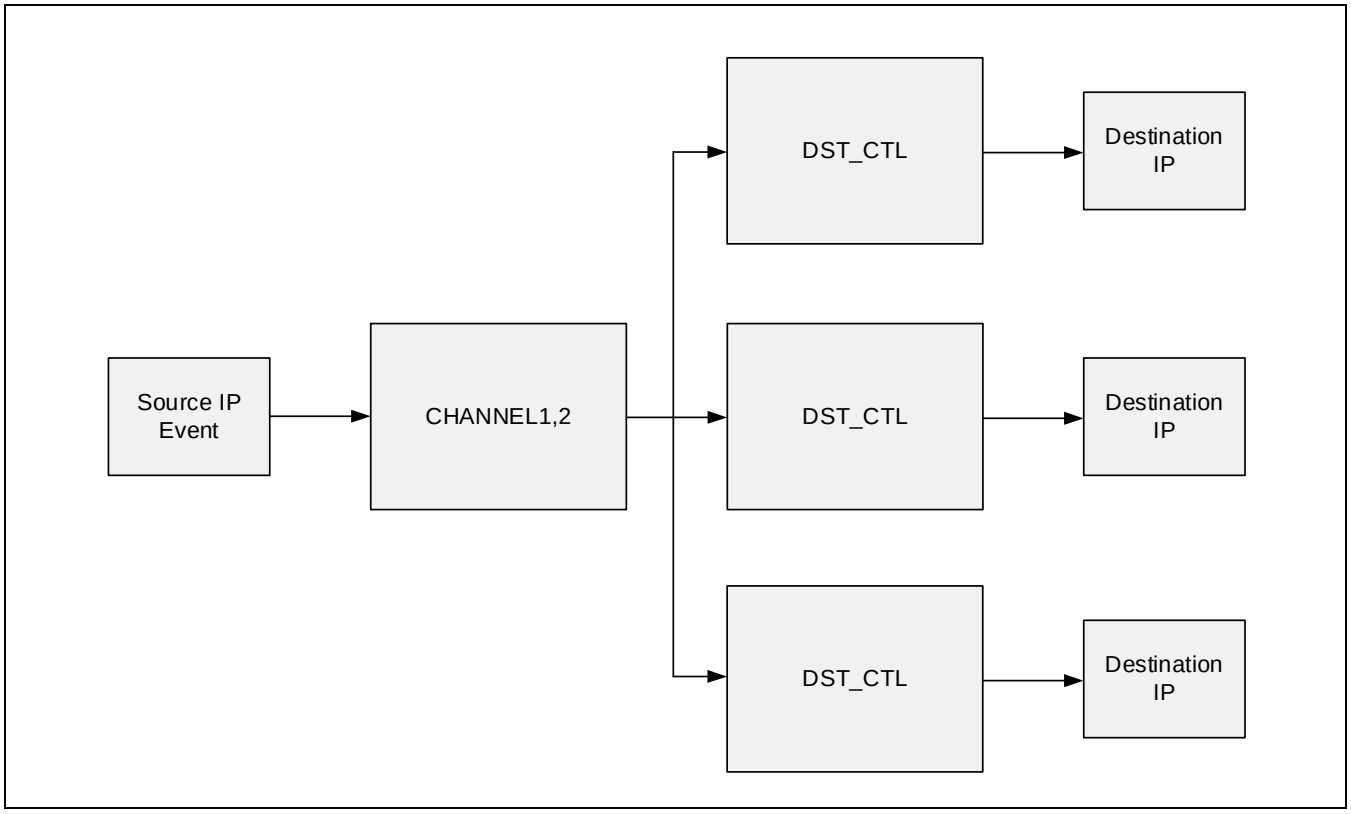


Figure 8-4 单个源触发多个目标

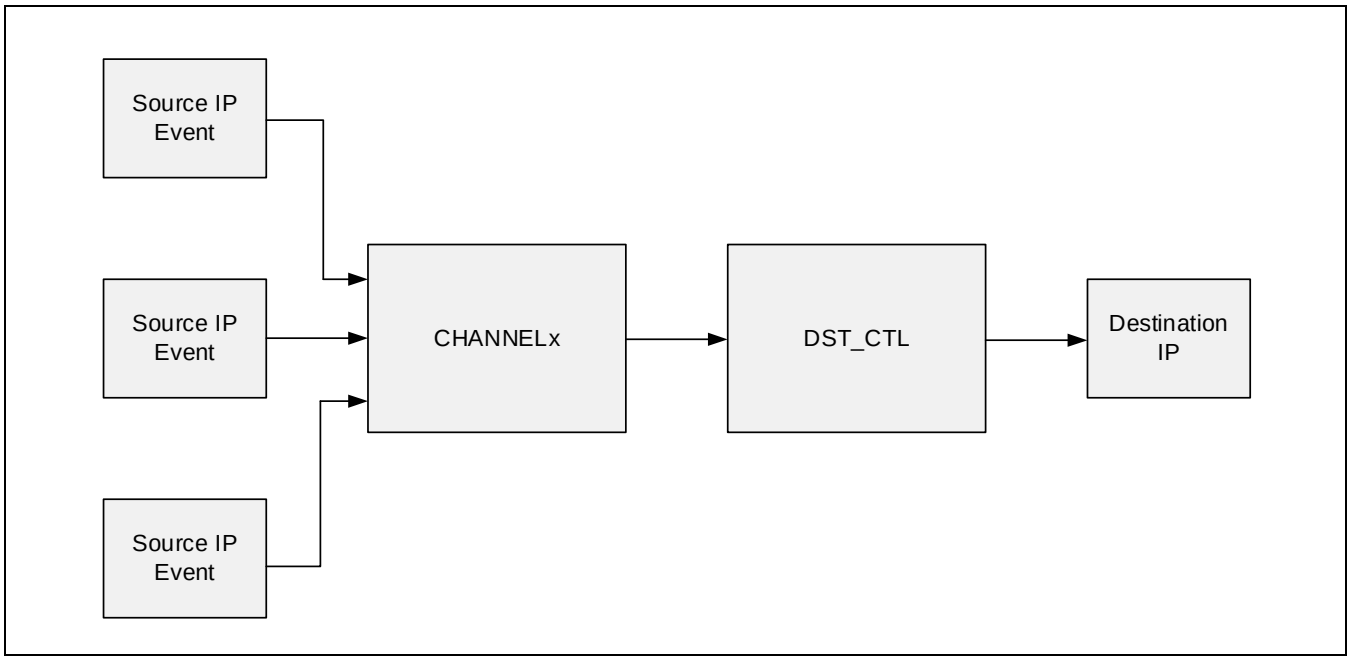


Figure 8-5　3个源触发单目标

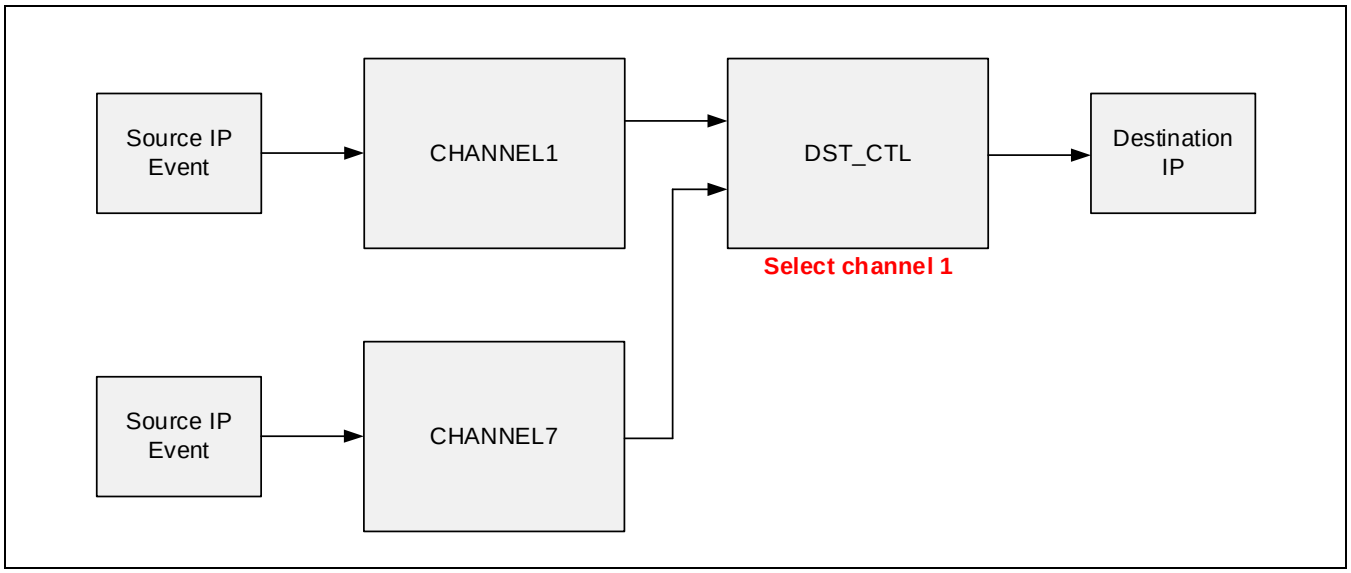


Figure 8-6　2个通道选择了相同的目标

7.2.2.2 事件对应表

事件源都是来自片上各IP模块。当IP在工作时，这些事件就会产生，而并不需要相应的中断使能。事件序号与IP的对应关系如下表格。

Table 8-1 事件对应表

源序号	事件源	目标序号	目标事件
0 （0H）	LPT_TRGOUT0	0 （0H）	LPT_SYNCIN0
1 （1H）	RSVD	1 （1H）	RSVD
2 （2H）	RSVD	2 （2H）	BT0_SYNCIN0
3 （3H）	RSVD	3 （3H）	BT0_SYNCIN1
4 （4H）	EXI_ TRGOUT0	4 （4H）	BT1_SYNCIN0
5 （5H）	EXI_ TRGOUT1	5 （5H）	BT1_SYNCIN1
6 （6H）	EXI_ TRGOUT2	6 （6H）	ADC_SYNCIN0
7 （7H）	EXI_TRGOUT3	7 （7H）	ADC_SYNCIN1
8 （8H）	EXI_TRGOUT4	8 （8H）	ADC_SYNCIN2
9 （9H）	EXI_TRGOUT5	9 （9H）	ADC_SYNCIN3
10 （AH）	RTC_TRGOUT0	10 （AH）	ADC_SYNCIN4
11 （BH）	RTC_TRGOUT1	11 （BH）	ADC_SYNCIN5
12 （CH）	BT_TRGOUT0	12 （CH）	RSVD
13 （DH）	BT_TRGOUT1	13 （DH）	RSVD
14 （EH）	RSVD	14 （EH）	RSVD
15 （FH）	RSVD	15 （FH）	RSVD
16 （10H）	EPT0_TRGOUT0	16 （10H）	EPT0_SYNCIN0
17 （11H）	EPT0_TRGOUT1	17 （11H）	EPT0_SYNCIN1
18 （12H）	EPT0_TRGOUT2	18 （12H）	EPT0_SYNCIN2
19 （13H）	EPT0_TRGOUT3	19 （13H）	EPT0_SYNCIN3
20 （14H）	RSVD	20 （14H）	EPT0_SYNCIN4
21 （15H）	RSVD	21 （15H）	EPT0_SYNCIN5
22 （16H）	RSVD	22 （16H）	RSVD
23 （17H）	RSVD	23 （17H）	RSVD
24 （18H）	RSVD	24 （18H）	RSVD
25 （19H）	RSVD	25 （19H）	RSVD
26 （1AH）	RSVD	26 （1AH）	RSVD

27 （1BH）	RSVD	27 （1BH）	RSVD
28 （1CH）	RSVD	28 （1CH）	RSVD
29 （1DH）	RSVD	29 （1DH）	RSVD
30 （1EH）	RSVD	30 （1EH）	RSVD
31 （1FH）	RSVD	31 （1FH）	RSVD
32 （20H）	GPT0_TRGOUT0	32 （20H）	RSVD
33 （21H）	GPT0_TRGOUT1	33 （21H）	RSVD
34 （22H）	RSVD	34 （22H）	RSVD
35 （23H）	RSVD	35 （23H）	RSVD
36 （24H）	RSVD	36 （24H）	GPT0_SYNCIN0
37 （25H）	RSVD	37 （25H）	GPT0_SYNCIN1
38 （26H）	RSVD	38 （26H）	GPT0_SYNCIN2
39 （27H）	RSVD	39 （27H）	GPT0_SYNCIN3
40 （28H）	RSVD	40 （28H）	GPT0_SYNCIN4
41 （29H）	RSVD	41 （29H）	GPT0_SYNCIN5
42 （2AH）	RSVD	42 （2AH）	RSVD
43 （2BH）	RSVD	43 （2BH）	RSVD
44 （2CH）	RSVD	44 （2CH）	RSVD
45 （2DH）	RSVD	45 （2DH）	RSVD
46 （2EH）	RSVD	46 （2EH）	RSVD
47 （2FH）	RSVD	47 （2FH）	RSVD
48 （30H）	ADC_TRGOUT0	48 （30H）	RSVD
49 （31H）	ADC_TRGOUT1	49 （31H）	RSVD
50 （32H）	RSVD	50 （32H）	RSVD
51 （33H）	RSVD	51 （33H）	RSVD
52 （34H）	RSVD	52 （34H）	RSVD
53 （35H）	RSVD	53 （35H）	RSVD
54 （36H）	RSVD	54 （36H）	RSVD
55 （37H）	RSVD	55 （37H）	RSVD
56 （38H）	RSVD	56 （38H）	RSVD
57 （39H）	RSVD	57 （39H）	RSVD
58 （3AH）	RSVD	58 （3AH）	RSVD

59 （3BH）	RSVD	59 （3BH）	RSVD
60 （3CH）	TOUCH_TRGOUT	60 （3CH）	TOUCH_SYNCIN
61 （3DH）	RSVD	61 （3DH）	RSVD
62 （3EH）	RSVD	62 （3EH）	RSVD
63 （3FH）	RSVD	63 （3FH）	RSVD

7.3 寄存器说明

7.3.1 寄存器表

Base Address: 0x4001_2000

Name	Offset	Description	R/W	Reset State
ETCB_ENABLE	0x0000	ETCB 使能寄存器	R/W	0x00000000
ETCB_SWTRG	0x0004	ETCB 软件触发寄存器	R/W	0x00000000
ETCB_CH0CON	0x0008	ETCB 通道 0 控制寄存器 0	R/W	0x00000000
ETCB_CH0CON	0x000C	ETCB 通道 0 控制寄存器 1	R/W	0x00000000
ETCB_CH1CON	0x0010	ETCB 通道 1 控制寄存器 0	R/W	0x00000000
ETCB_CH1CON	0x0014	ETCB 通道 1 控制寄存器 1	R/W	0x00000000
ETCB_CH2CON	0x0018	ETCB 通道 2 控制寄存器 0	R/W	0x00000000
ETCB_CH2CON	0x001C	ETCB 通道 2 控制寄存器 1	R/W	0x00000000
RSVD	0x0020 ~ 0x002C	保留	R	0x00000000
ETCB_CH3CON	0x0030	ETCB 通道 3 控制寄存器	R/W	0x00000000
ETCB_CH4CON	0x0034	ETCB 通道 4 控制寄存器	R/W	0x00000000
ETCB_CH5CON	0x0038	ETCB 通道 5 控制寄存器	R/W	0x00000000
ETCB_CH6CON	0x003C	ETCB 通道 6 控制寄存器	R/W	0x00000000
ETCB_CH7CON	0x0040	ETCB 通道 7 控制寄存器	R/W	0x00000000

7.3.1.1 ETCB_ENABLE (ETCB使能寄存器)

- 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																															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	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	Reset Value
ENABLE	[0]	RW	ETCB模块使能控制 0：禁止 1：使能	0

7.3.1.2 ETCB_SWTRG (ETCB软件触发寄存器)

- 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																								SWTRG_CH7	SWTRG_CH6	SWTRG_CH5	SWTRG_CH4	SWTRG_CH3	SWTRG_CH2	SWTRG_CH1	SWTRG_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
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	Reset Value
SWTRG_CHx	[7:0]	RW	软件触发控制 0：无效 1：触发该通道的事件	0

7.3.1.3 ETCB_CH0CON0 (通道0控制寄存器0)

- 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					SRC2_SEL						SRC2_EN	RSVD				SRC1_SEL					SRC1_EN	RSVD			SRC0_SEL						SRC0_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	RW	RW	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
SRC0_EN	[0]	R/W	触发源0使能控制 0: 禁止 1: 使能	0x0
SRC0_SEL	[6:1]	R/W	触发源0的事件选择位 参考事件对应表	0x0
SRC1_EN	[10]	R/W	触发源1使能控制 0: 禁止 1: 使能	0x0
SRC1_SEL	[16:11]	R/W	触发源1的事件选择位 参考事件对应表	0x0
SRC2_EN	[20]	R/W	触发源2使能控制 0: 禁止 1: 使能	0x0
SRC2_SEL	[26:21]	R/W	触发源2的事件选择位 参考事件对应表	0x0

7.3.1.4 ETCB_CH0CON1 (通道0控制寄存器1)

- Address = Base Address + 0x000C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DST_SEL						RSVD																								TRIG_MODE	CH0_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
RW	RW	RW	RW	RW	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	RW	RW

Name	Bit	Type	Description	Reset Value
CH0_EN	[0]	R/W	通道0使能控制 0: 禁止 1: 使能	0x0
TRIG_MODE	[1]	R/W	触发模式选择 0: 硬件触发 1: 软件触发 (由ETCB_SWTRG寄存器触发)	0x0
DST_SEL	[31:26]	R/W	触发目标选择 参考事件对应表	0x0

7.3.1.5 ETCB_CH1CON0 (通道1控制寄存器0)

- 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					DST2_SEL						DST2_EN	RSVD				DST1_SEL					DST1_EN	RSVD		DST0_SEL							DST0_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	RW	RW	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
DST0_EN	[0]	R/W	触发目标0使能控制 0: 禁止 1: 使能	0x0
DST0_SEL	[6:1]	R/W	触发目标0的事件选择位 参考事件对应表	0x0
DST1_EN	[10]	R/W	触发目标1使能控制 0: 禁止 1: 使能	0x0
DST1_SEL	[16:11]	R/W	触发目标1的事件选择位 参考事件对应表	0x0
DST2_EN	[20]	R/W	触发目标2使能控制 0: 禁止 1: 使能	0x0
DST2_SEL	[26:21]	R/W	触发目标2的事件选择位 参考事件对应表	0x0

7.3.1.6 ETCB_CH1CON1 (通道1控制寄存器1)

- 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
SRC_SEL								RSVD																						TRIG_MODE	CH1_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
RW	RW	RW	RW	RW	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	RW	RW

Name	Bit	Type	Description	Reset Value
CH1_EN	[0]	R/W	通1使能控制 0: 禁止 1: 使能	0x0
TRIG_MODE	[1]	R/W	触发模式选择 0: 硬件触发 1: 软件触发 (由ETCB_SWTRG寄存器触发)	0x0
SRC_SEL	[31:26]	R/W	触发源选择 参考事件对应表	0x0

7.3.1.7 ETCB_CH2CON0 (通道2控制寄存器0)

- Address = Base Address + 0x0018

31	30	29	28	27	26	25	24	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					DST2_SEL						DST2_EN	RSVD				DST1_SEL					DST1_EN	RSVD			DST0_SEL						DST0_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	RW	RW	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
DST0_EN	[0]	R/W	触发目标0使能控制 0: 禁止 1: 使能	0x0
DST0_SEL	[6:1]	R/W	触发目标0的事件选择位 参考事件对应表	0x0
DST1_EN	[10]	R/W	触发目标1使能控制 0: 禁止 1: 使能	0x0
DST1_SEL	[16:11]	R/W	触发目标1的事件选择位 参考事件对应表	0x0
DST2_EN	[20]	R/W	触发目标2使能控制 0: 禁止 1: 使能	0x0
DST2_SEL	[26:21]	R/W	触发目标2的事件选择位 参考事件对应表	0x0

7.3.1.8 ETCB_CH2CON1 (通道2控制寄存器1)

- Address = Base Address + 0x001C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC_SEL								RSVD																						TRIG_MODE	CH1_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
RW	RW	RW	RW	RW	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	RW	RW

Name	Bit	Type	Description	Reset Value
CH2_EN	[0]	R/W	通道2使能控制 0: 禁止 1: 使能	0x0
TRIG_MODE	[1]	R/W	触发模式选择 0: 硬件触发 1: 软件触发 (由ETCB_SWTRG寄存器触发)	0x0
SRC_SEL	[31:26]	R/W	触发源选择 参考事件对应表	0x0

7.3.1.9 ETCB_CHxCON (通道x控制寄存器) (x=3~7)

- CH3CON: Address = Base Address + 0x0030
- CH4CON: Address = Base Address + 0x0034
- CH5CON: Address = Base Address + 0x0038
- CH6CON: Address = Base Address + 0x003C
- CH7CON: Address = Base Address + 0x0040

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

Name	Bit	Type	Description	Reset Value
CHx_EN	[0]	R/W	通道x使能控制 0: 禁止 1: 使能	0x0
TRIG_MODE	[1]	R/W	触发模式选择 0: 硬件触发 1: 软件触发 (由ETCB_SWTRG寄存器触发)	0x0
SRC_SEL	[18:12]	R/W	触发源选择 参考事件对应表	0x0
DST_SEL	[31:26]	R/W	触发目标选择 参考事件对应表	0x0

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。

注：如果系列内芯片不具有某一外围，那它就不具备该外围的所有资源。具体参考芯片的数据手册。

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(INTVREF)
- 支持多路外部模拟输入AIN[15:0]，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20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

8.1.2 管脚描述

Table 9-1 ADC管脚描述

管脚名称	功能	I/O类型	有效电平	说明
VREF+/INTVREF	模拟参考电压	模拟	-	-
AIN0 to AIN15	模拟信号输入	模拟	-	-

8.1.3 模块框图

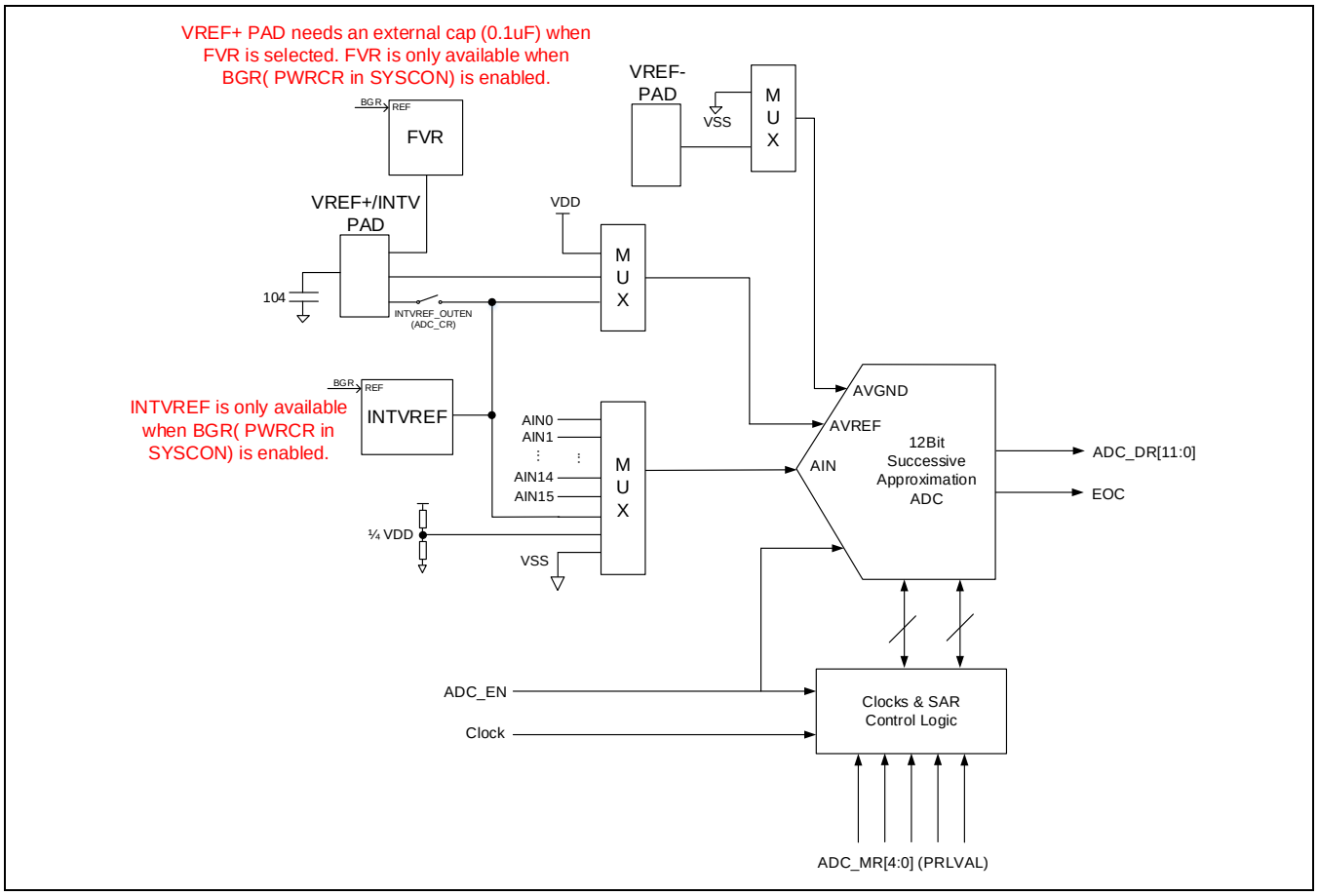


Figure 9-1 ADC模块框图

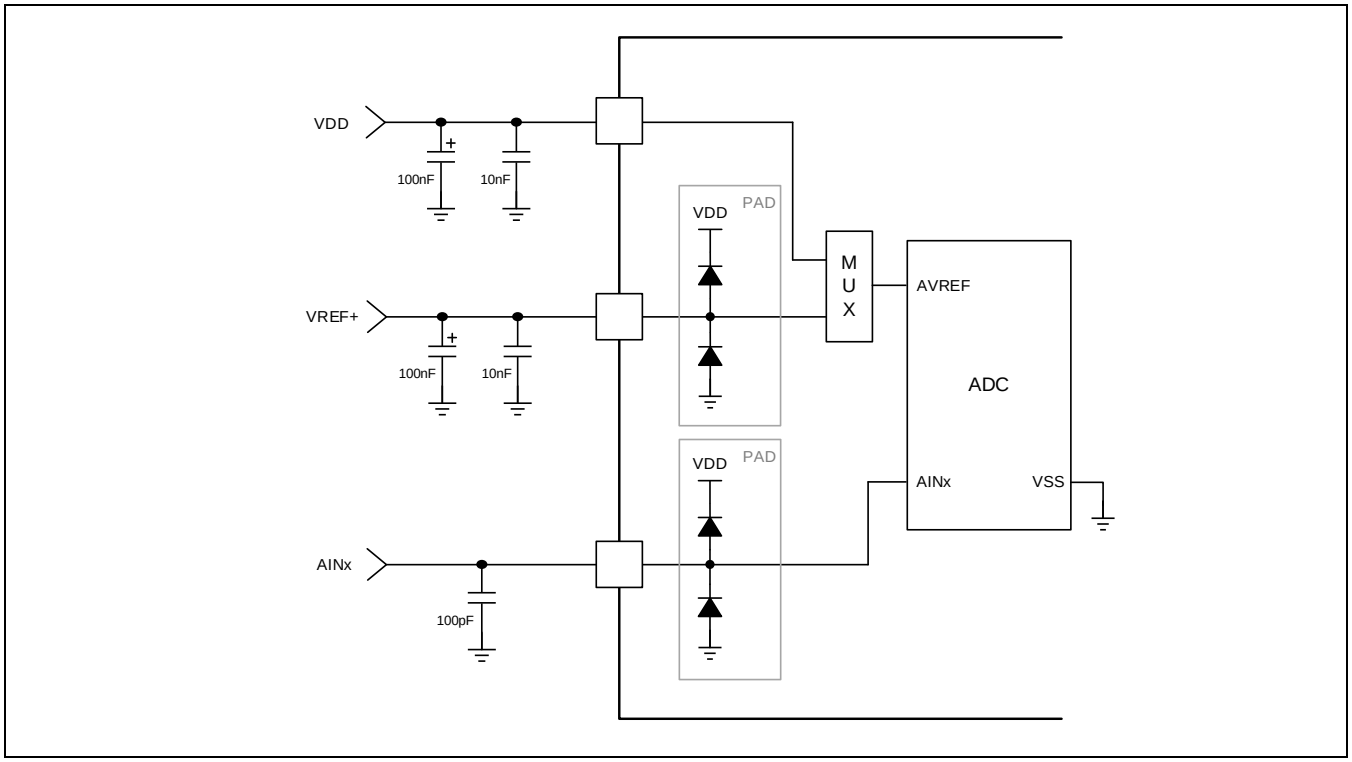


Figure 9-2 参考电路

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

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

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

Table 9-2 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

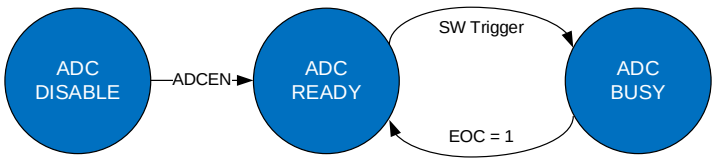


Figure 9-3 ADC状态机

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+)，同时负向参考电压源也可以由外部提供，由ADC_CR寄存器中的VREF_SEL位控制。正向电压参考源提供芯片VDD，固定电压源FVR，内部参考电压INTVREF输出，外部VREF+管脚的4种选择，负向电压参考源提供芯片VSS和外部VREF-管脚的2种选择，各种参考源的组合参见ADC_CR寄存器的VREF_SEL控制位。

如果希望使用固定电压源FVR提供参考电压，有两个配置需要满足：

1. 在GPIO的配置中使能对应的AF功能 (VREF+)

2. ADC_CR中VREF_SEL选择FVR作为正向参考电压，或者ADC_CR中FVR_EN设置为1.

使用固定电压源作为ADC的参考电压，实际是将FVR电压输出到VREF+管脚，再通过VREF+管脚连到ADC的参考电压上。如果在某些特殊应用中，不需要将FVR用作ADC参考，而是有其它用途，也可以通过使能ADC_CR中的FVR_EN位，将FVR输出到管脚上，这时候ADC的VREF_SEL可以选择其它的参考源，比如VDD, INTVREF等。

固定电压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。

如果希望使用内部参考电压INTVREF作为参考，只需要使用ADC_CR中的VREF_SEL选择INTVREF作为相应参考即可。如果需要将INTVREF输出到IO管脚上，则需要将ADC_CR中的INTVREF_OUTEN置1。如果不需要将INTVREF输出到外部，而只是用作ADC的输入或者ADC的参考电压，则不需要使能INTVREF_OUTEN。ADC_CR中的INTVREF_SEL位用来选择INTVREF电压。

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 11-1 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(S/H+12)个时钟周期。S/H(Sample&Hold 采样保持)时间可以通过 ADC_SHR 寄存器设置。默认的采样保持时间 (6 个周期) 可以满足大部分应用场景的需求，在某些特殊应用场景中，如果需要更长的采样和保持时间，可以通过特殊的 ADC_SHR 寄存器来实现。

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 S/H 时间为默认值 6 个周期)：

(6 个 S/H 时钟周期) + (每位 1 个时钟转换周期 x 13 位) + (3 个同步和结果处理时钟周期) = 22 个周期

22 x 100ns = 2.2us (476ksps)

采样和保持时间的长度可以由下面公式计算：

S/H 时间 = (6 + (ADC_SHR – 3)) * (1/F_ANA)

例如：F_ANA = 10MHz, ADC_SHR 设置 0x5 即 6+(5-3)=8 个周期，那么 S/H 时间为：8*100ns = 0.8us

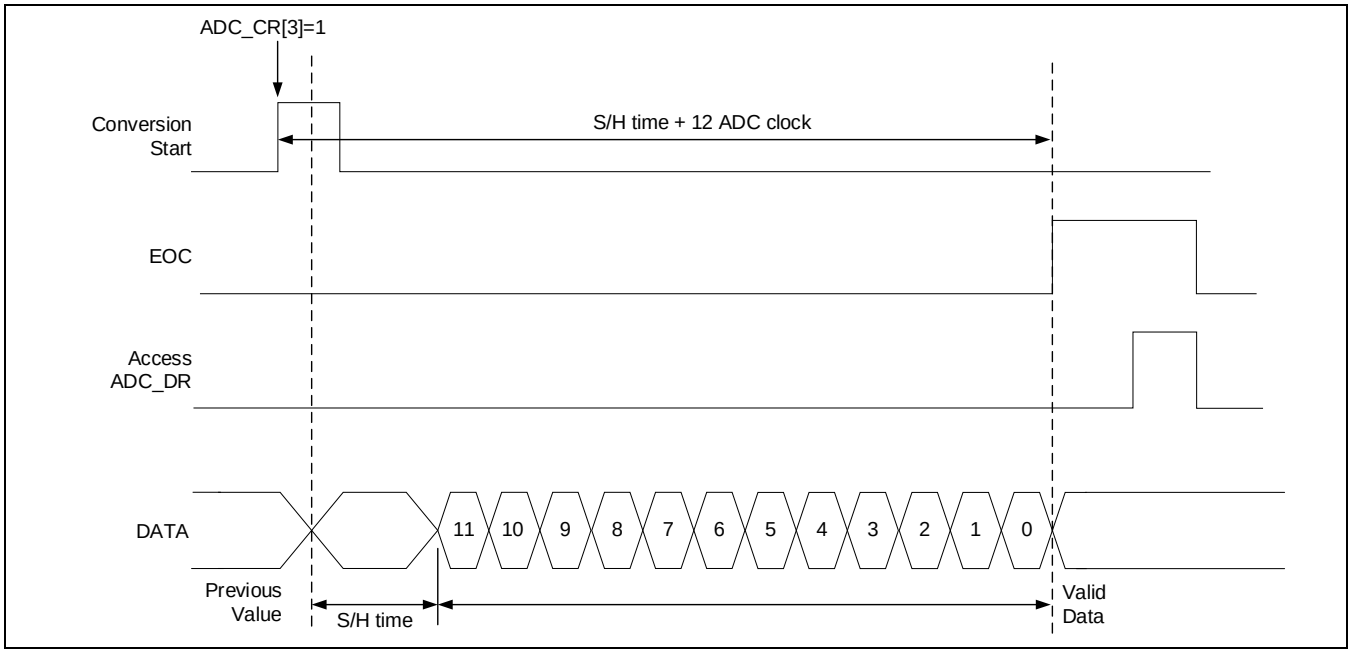


Figure 9-4 ADC工作时序图

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。用户可以设置转换序列的个数，最多16个，ADC_SEQ0~ADC_SEQ15这16个寄存器用来配置每个转换序列的输入通道，采样周期，是否做平均计算等参数。如果设置转换序列个数为16，那么ADC在启动后，会先转换ADC_SEQ0设置的通道，然后再转换ADC_SEQ1, ADC_SEQ2, ...，最后转换ADC_SEQ15设置的通道，并将转换结果存在ADC_DR0, ADC_DR1, ..., ADC_DR15这16个转换结果寄存器中。如果设置转换序列个数为1，那么ADC只会转换ADC_SEQ0设置的通道，并且将结果存入ADC_DR0。同理如果转换序列个数为5，那么ADC会依次转换ADC_SEQ0 ~ ADC_SEQ4设置的通道，将结果依次存入ADC_DR0 ~ ADC_DR4中。下表列出了ADC_MR寄存器中NBRCH和转换序列个数的关系：

Table 9-3 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
…	…
1110	15
1111	16

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。16 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 9-4 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
…	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 9-5 CV_CNT的值和重复采样次数

CV_CNT[3:0]	重复采样次数
0000	1
0001	2
0010	4
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512

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。平均值的计算方法可以选择，由 AVG_SEL 位决定。如果 AVG_SEL 选择 0，即为不平均，那么 ADC_DRx 结果寄存器的值为多次转换后的所有结果之和；如果 AVG_SEL 选择 1，那么 ADC_DRx 的值则为多次转换之和除以 2，也即多次转换之和做一次右移操作(前端补 0)。

Table 9-6 AVG_SEL的值和ADC_DRx结果

	AVG_SEL	平均值计算方法	ADC_DRx的值
CV_CNT寄存器选择的若干次采样结果之和为ADC_SUM	0000	ADC_SUM/1	ADC_SUM
	0001	ADC_SUM/2	ADC_SUM>>1
	0010	ADC_SUM/4	ADC_SUM>>2
	0011	ADC_SUM/8	ADC_SUM>>3
	0100	ADC_SUM/16	ADC_SUM>>4
	0101	ADC_SUM/32	ADC_SUM>>5
	0110	ADC_SUM/64	ADC_SUM>>6
	0111	ADC_SUM/128	ADC_SUM>>7
	1000	ADC_SUM/256	ADC_SUM>>8
	1001	ADC_SUM/512	ADC_SUM>>9

例如，在 ADC_SEQ0 中设置 CV_CNT=0x3，AVG_CAL=1，AVG_SEL=0，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)/1；如果 AVG_SEL=1，那么 ADC_DR0 的值为(DATA0+DATA1+…+DATA7)/2；如果 AVG_SEL=3，那么 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 转换结果处理

从转换序列定义章节可以知道，每个转换序列都有一个对应的转换结果寄存器 ADC_DRx，在每个转换序列结束后，该寄存器的结果都会被更新为当次 ADC 的转换结果。在一些应用场景中，某些序列的转换结果可能不需要被更新，而是需要保持上次转换的值，那么 ADC_DRMASK 寄存器可以用来实现该场景的需求。ADC_DRMASK 有 16 位，对应于 16 个 ADC_DR。如果 ADC_DRMASK 的相应位为 1，那么该位对应的 ADC_DR 寄存器值不会被更新，直到 MASK 值被改为 0 为止。

8.1.11 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，并且触发相应的中断；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，并且触发相应的中断。在 ADC_MR 第 30 位 CMP_OS 位为 0 的持续触发模式下，只要 ADC 转换结果比设定的阈值高或者低，就会持续触发相应的 H 或者 L 中断；而在 CMP_OS 位为 1 的单次触发模式下，只有当 ADC 转换结果从比阈值低变成比阈值高(或者从比阈值高变成比阈值低)的当次转换后会触发中断，后续如果一直保持在比阈值高或者低的状态，那么相应的 H 或者 L 中断不会被触发。

ADC_MR 寄存器中的 NBRCMPx 位用来指定要比较的转换序列号。

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

Table 9-7 NBRCMPx[3:0]的值和需要比较的转换序列号

NBRCMPx[3:0]	需要比较的转换序列号
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。

8.1.12 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 9-8 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	ADC_SYNCIN0触发源 (参考ETCB章节)

011	ADC_SYNCIN1触发源 (参考ETCB章节)
100	ADC_SYNCIN2触发源 (参考ETCB章节)
101	ADC_SYNCIN3触发源 (参考ETCB章节)
110	ADC_SYNCIN4触发源 (参考ETCB章节)
111	ADC_SYNCIN5触发源 (参考ETCB章节)

ADC在使能触发(ADC_SEQx中TRG_SRC不为0，ADC_SYNCR中相应的SYNCEN使能位为1)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触发功能支持一次性触发和连续触发模式。当触发输入通道被设置为一次性触发模式时，在一次触发事件被检测到后，该通道将不允许后续的触发事件通过，直到被软件重置（ADC_SYNCR中的REARM位）后才允许新的触发事件通过。

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。

如下图所示，绿色SEQ0为正在转换的序列0，按照正常转换顺序，SEQ1为下一个需要转换的序列。在SEQ0转换的过程中，SEQ9所设置的触发源被触发了，那么下一个需要转换的序列马上变为SEQ9。当SEQ0转换完成后，SEQ9将继续开始转换，并且SEQ10将变为SEQ9的下一个转换序列。可以看到，因为SEQ9被触发转换，所以SEQ1以及后续的SEQ2 ~ SEQ8都被跳过了。

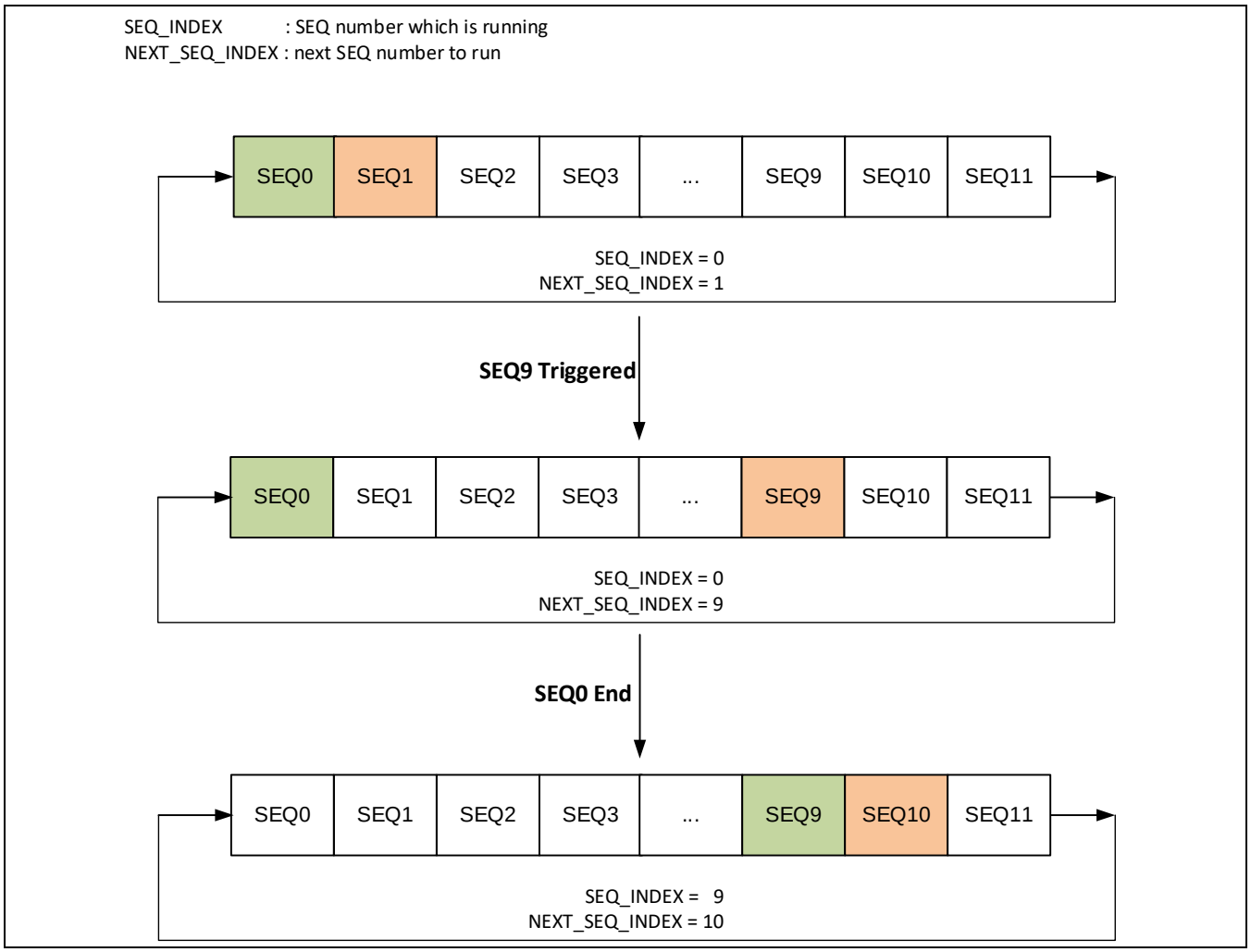


Figure 9-5 触发原理示意图

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。如下图，当SEQ0在转换时，SEQ2和SEQ9同时被触发了，那么这两个转换都会被执行，但是序号小的SEQ2会被先转换，然后再继续转换SEQ9，接着再按照顺序继续执行下去。

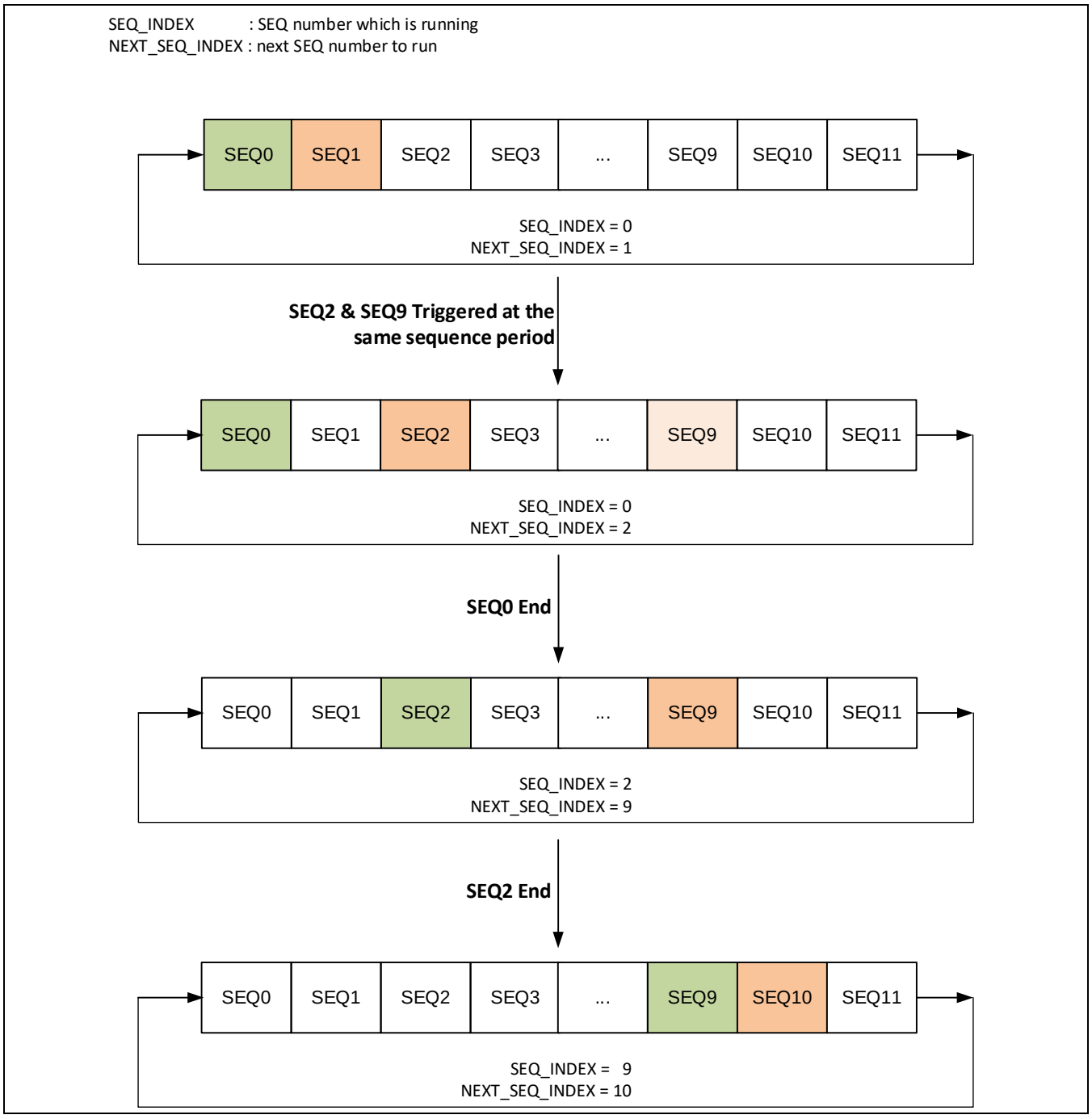


Figure 9-6 同时触发示意图

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，ADC_PRI寄存器设置为0x3，那么SEQ0 ~ SEQ2将不会在转换序列当中，ADC启动时，会直接转换SEQ3。如果在SEQ3转换时，SEQ2和SEQ9同时发生，那么SEQ2会先被转换，之后再转换SEQ9，接着再按照顺序继续转换下去。也就是说SEQ0 ~ SEQ2只有在被触发的时候(需要的时候)才会转换，而不会在转换序列中一直不停的被循环执行。

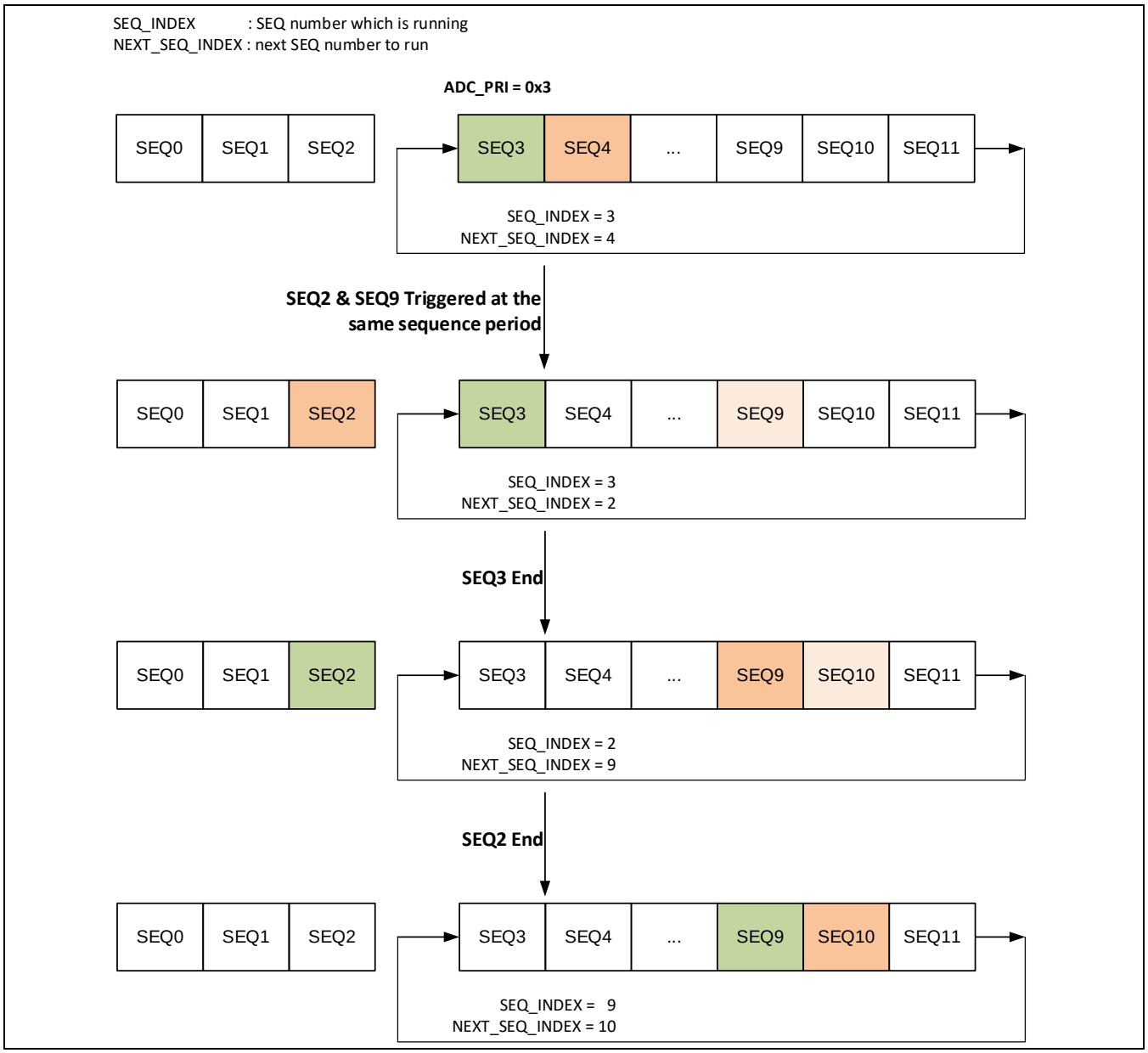


Figure 9-7 触发优先级示意图

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。

8.1.13 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 9-9 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ADCCLKEN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

8.1.14 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：通常在单次转换模式下检查 EOC 标志位，每次读任何一个转换结果寄存器(ADC_DR)都会将 EOC 标志位清零。

8.1.15 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。

8.1.16 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)。

8.1.17 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)。

8.1.18 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)。

8.1.19 ADC事件输出接口 (ETCB接口)

ADC的各种事件可以作为输出，给ETCB模块作为其它功能模块的触发输入。ADC_EVTRG中的SYNCINx位用来选择作为输出的ADC事件，TRGxOE用来使能该事件的输出。

8.1.20 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。

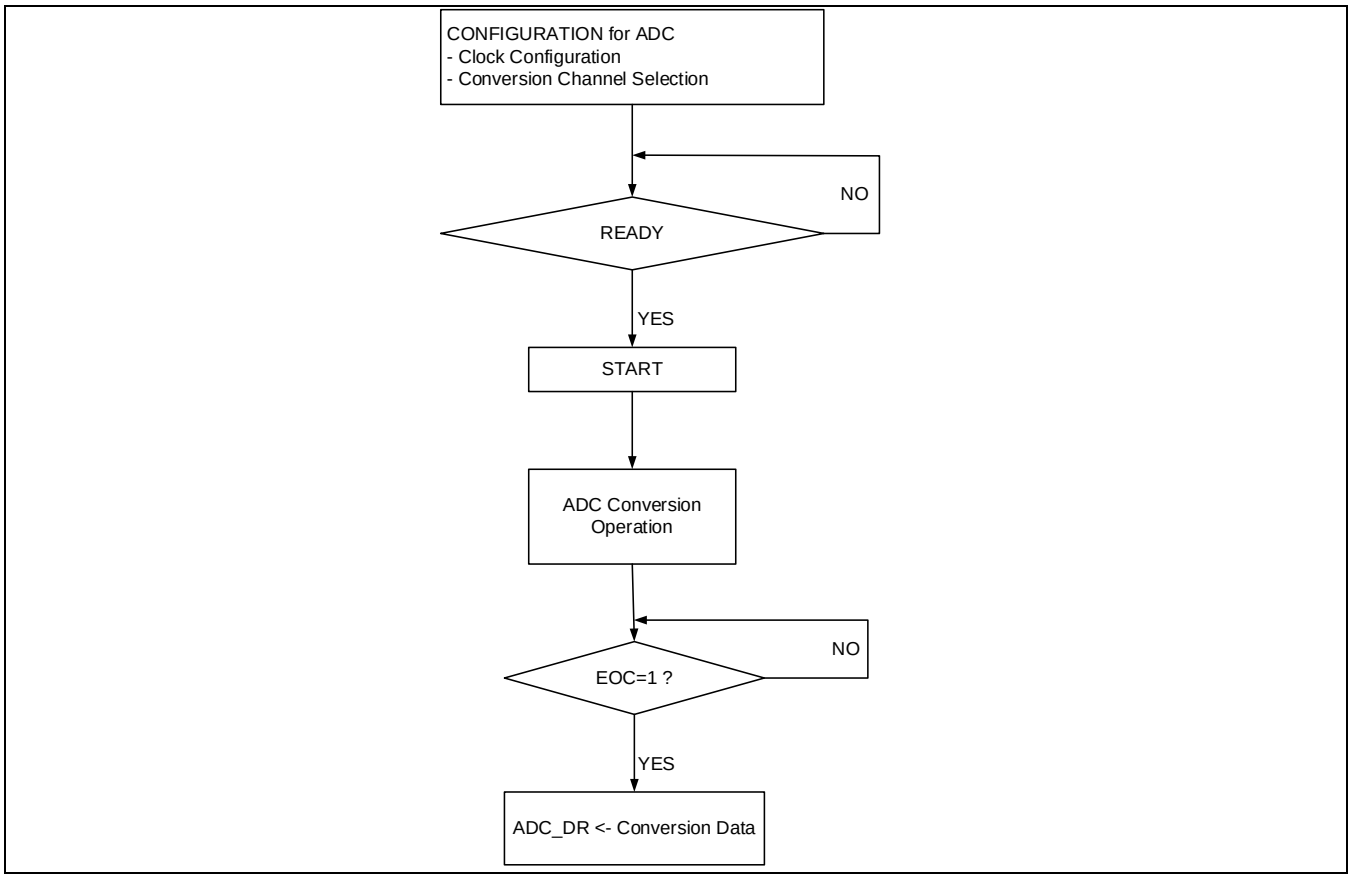


Figure 9-8 ADC工作流程图

8.1.21 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 22 个时钟周期后，转换完成。12 位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	0x2AAAAAA0
–	0x000C	保留	–
ADC_CR	0x0010	控制寄存器	0x80040800
ADC_MR	0x0014	模式寄存器	0x00000001
ADC_SHR	0x0018	采样保持周期寄存器	0x00000003
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)	0x0000009F
ADC_PRI	0x0070	转换序列优先级寄存器	0x00000000
ADC_TDL0	0x0074	触发延时寄存器0	0x00000000
ADC_TDL1	0x0078	触发延时寄存器1	0x00000000
ADC_SYNCR	0x007C	触发同步控制寄存器	0x00000000
–	0x0080	保留	0x00000000
–	0x0084	保留	0x00000000
ADC_EVTRG	0x0088	事件触发选择寄存器	0x00000000
–	0x008C ~ 0x00FC	Reserved	0x00000000

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
ADC_DRMASK	0x0148	禁止转换结果更新寄存器	0x00000000

8.2.2 ADC_ECR (时钟使能寄存器)

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

31	30	29	28	27	26	25	24	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																													ADCCLKEN	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
ADCCLKEN	[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 = —

31	30	29	28	27	26	25	24	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																													ADCCLKEN	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
ADCCLKEN	[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 = 0x2aaaaaa0

31	30	29	28	27	26	25	24	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		ADCCLKEN	RSVD
0	0	0	0	0	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
ADCCLKEN	[1]	R	**ADC：ADC 时钟状态** 0: ADC 时钟被禁止 1: ADC 时钟被使能
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

8.2.5 ADC_CR (控制寄存器)

- Address = Base Address + 0x0010, Reset Value = 0x8004_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
ACCURACY	RSVD					FVR_LVL	FVR_EN	RSVD					INTVREF_SEL		INTVREF_EN	RSVD						VREF_SEL				SWTRG	STOP	START	ADCDIS	ADCEN	SWRST
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
RW	R	W	W	W	W	RW	RW	W	W	W	W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	W	W	W	W	W	W

Name	B	Type	Description
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 模拟模块使能** 0: 无效 1: 使能 ADC 模块
ADCDIS	[2]	W	**ADCDIS : ADC 模拟模块禁止** 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_SEL	[9:6]	RW	**VREF: ADC 电压参考电源选择** 0000: 正向为内部 VDD，负向为 VSS 0001: 正向为外部 VREF+管脚，负向为 VSS 0010: 正向为 FVR 2.048V 输出，负向为 VSS 0011: 正向为 FVR 4.096V 输出，负向为 VSS 0100: 正向为内部 INTVREF 输出，负向为 VSS 1000: 正向为内部 VDD，负向为 VREF- 1001: 正向为外部 VREF+管脚，负向为 VREF- 1010: 正向为 FVR 2.048V 输出，负向为 VREF- 1011: 正向为 FVR 4.096V 输出，负向为 VREF- 1100: 正向为内部 INTVREF 输出，负向为 VREF- 其它: 保留 注意：使用 FVR 做参考时，外部需要接 100nF 的电容。
INTVREF_OUTEN	[16]	RW	**INTVREF_OUTEN: 使能内部参考电压输出到管脚** 0: 输出到管脚(INTV)禁止 1: 输出到管脚(INTV)使能 注：该位只影响 INTVREF 是否输出到 IO 管脚，并不影响 AVREF 以及 ADC 输入通道的 INTVREF 使用
INTVREF_SEL	[18:17]	RW	**INTVREF_SEL: 内部参考电压输入源选择** 00: 保留 01: 保留 10: 内部 1.0V 电压 11: 保留 注意：目前只提供 1.0V 作为参考电压。
FVR_EN	[24]	RW	**FVR_EN: 使能固定电压参考源** 0: 禁止 1: 使能 注：AVREF 选择 FVR 时无需操作这位。
FVR_LVL	[25]	RW	**FVR_LVL: 固定电压参考源的电压值选择** 0: 2.048V 1: 4.096V 注：AVREF 选择 FVR 时无需操作这位。
ACCURACY	[31]	R/W	**ACCURACY: ADC 转换精度选择位** 0: 保留 1: 12 位 注意：该位请保持为 1.

8.2.6 ADC_MR (模式寄存器)

- Address = Base Address + 0x0014, 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
CONTCV	CMP_OS	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	RW	R	R	R	R	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
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 - 当选择 INTVREF 作为 ADC 参考电压时，ADC 模拟模块的时钟频率不能超过 2MHz。FVR 做参考时，没有限制。 - 如果系统时钟为 40MHz，那么 PRLVAL 至少为 1
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。
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
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

CMP_OS	[30]	R/W	**CMP_OS: 一次性比较** 0: 每次得到比较结果时，都会产生 ADC_CMPx 中断 1: 只有转换结果从比 ADC_CMPxH 小的值变成比它大的值，或者从比 ADC_CMPxL 大的值变成比它小的值时，才会产生 ADC_CMPx 中断。
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。 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_SHR (采样保持周期寄存器)

Address = Base Address + 0x018, 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																								SHR							
0	0	0	0	0	0	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	RW	RW	RW	RW	RW	RW	RW	RW

Name	B	Type	Description
SHR	[7:0]	RW	**SHR：采样保持 (Sample & Hold) 周期数** 设置 ADC 转换中采样保持的周期数，该周期数基于 ADC_MR 寄存器中 PRLVAL 分频后的 ADC 工作时钟频率 FADC。采样保持周期数至少为 3 个周期，小于 3 的值无法写入该寄存器。

8.2.8 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	RVSD
0	0	0	0	0	0	0	0	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	R

Name	B	Type	Description
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

8.2.9 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
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
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
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
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
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
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
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

ADCENS	[8]	R	**ADCENS : ADC 使能状态** 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
SEQ_INDEX	[13:10]	R	**SEQ_INDEX : 当前转换序列** 该寄存器的值为当前转换的序列号
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

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 9-10 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

8.2.10 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
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。

8.2.11 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
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

8.2.12 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
EOC	[0]	R	**EOC： 转换结束中断** 0: 该中断没有使能 1: 该中断使能
READY	[1]	R	**READY：ADC READY 中断** 0: 该中断没有使能 1: 该中断使能
OVR	[2]	R	**OVR： 转换溢出中断** 0: 该中断没有使能 1: 该中断使能
CMP0H	[4]	R	**CMP0H： 转换结果高于 ADC_CMP0 中断** 0: 该中断没有使能 1: 该中断使能
CMP0L	[5]	R	**CMP0L： 转换结果低于 ADC_CMP0 中断** 0: 该中断没有使能 1: 该中断使能
CMP1H	[6]	R	**CMP1H： 转换结果高于 ADC_CMP1 中断** 0: 该中断没有使能 1: 该中断使能
CMP1L	[7]	R	**CMP1L： 转换结果低于 ADC_CMP1 中断** 0: 该中断没有使能 1: 该中断使能
SEQ_END[x]	[31:16]	R	**SEQ_END[x]：SEQx 序列转换完成中断** 0: 该中断没有使能

			1: 该中断使能

8.2.13 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_SEL				AVG_CAL	CV_CNT			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	1	1	1	1	1
R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW

Name	B	Type	Description
AIN_SEL	[4:0]	RW	模拟输入通道选择 AIN_SEL 值 (BIN / DEC) — 输入选择 00000 / 0 — AIN0 00001 / 1 — AIN1 00010 / 2 — AIN2 …… / …… — …… 10001 / 17 — AIN17 …… / …… — N/A 11100 / 28 — INTVREF 11101 / 29 — 1/4VDD 11110 / 30 — VSS 11111 / 31 — N/A
RSVD	[7:5]	RW	保持默认值: **100**
CV_CNT	[11:8]	RW	**CV_CNT:** 连续重复采样次数 0000 : 1 0001 : 2 0010 : 4 0011 : 8

			0100 : 16 0101 : 32 0110 : 64 0111 : 128 1000 : 256 1001 : 512 Other : 保留
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，由 CV_CNT 和 AVG_SEL 决定。否则，ADC_DRx 将保存最后一次转换的值。
AVG_SEL	[16:13]	RW	**AVG_SEL: 平均系数选择** 0000 : 1 (不平均) 0001 : 2 0010 : 4 0011 : 8 0100 : 16 0101 : 32 0110 : 64 0111 : 128 1000 : 256 1001 : 512 Other : 保留
TRG_SRC	[19:17]	RW	**TRG_SRC : 触发源选择** 000 : 无触发 001 : 软件触发(ADC_CR 中的 SWTRG 位) 010 : ADC_SYNCIN0 (ETCB) 011 : ADC_SYNCIN1 (ETCB) 100 : ADC_SYNCIN2 (ETCB) 101 : ADC_SYNCIN3 (ETCB)

			110 : ADC_SYNCIN4 (ETCB) 111 : ADC_SYNCIN5 (ETCB)

8.2.14 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
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

8.2.15 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								TRGIN2_TDL								TRGIN1_TDL								TRGIN0_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	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
TRGIN0_TDL	[7:0]	RW	**TRGIN0_TDL：ADC_TRGIN0 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 = (TRGIN0_TDL+1) x 4 x PCLK 周期
TRGIN1_TDL	[15:8]	RW	**TRGIN1_TDL：ADC_TRGIN1 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 = (TRGIN1_TDL+1) x 4 x PCLK 周期
TRGIN2_TDL	[23:16]	RW	**TRGIN2_TDL：ADC_TRGIN2 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 = (TRGIN2_TDL+1) x 4 x PCLK 周期

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_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								SOC5_TDL								SOC4_TDL								SOC3_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	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
TRGIN3_TDL	[7:0]	RW	**TRGIN3_TDL：ADC_TRGIN3 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 = (TRGIN3_TDL+1) x 4 x PCLK 周期
TRGIN4_TDL	[15:8]	RW	**TRGIN4_TDL：ADC_TRGIN4 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 = (TRGIN4_TDL+1) x 4 x PCLK 周期
TRGIN5_TDL	[23:16]	RW	**TRGIN5_TDL：ADC_TRGIN5 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 = (TRGIN5_TDL+1) x 4 x PCLK 周期

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。

8.2.17 ADC_SYNCR (同步控制寄存器)

- 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										REARM5	REARM4	REARM3	REARM2	REARM1	REARM0	RSVD		OSTMD5	OSTMD4	OSTMD3	OSTMD2	OSTMD1	OSTMD0	RSVD		SYNCEN5	SYNCEN4	SYNCEN3	SYNCEN2	SYNCEN1	SYNCEN0
0	0	0	0	0	0	0	0	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	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
SYNCENx	[5:0]	RW	外部同步触发使能控制。 0：禁止当前触发输入通道 1：使能当前触发输入通道 SYNCINx： ETCB模块中配置的触发源
OSTMDx	[13:8]	RW	一次性同步触发模式选择。 0h：连续触发模式 1h：一次性触发模式 当该输入通道被设置为一次性触发模式后，在一次触发事件被检测到后，该通道将不允许后续的触发事件通过，直到被软件重置（REARM）后才允许新的触发事件通过。
REARMx	[21:16]	RW	在一次性同步触发模式下，软件重置当前通道状态控制位。 当读取时，返回当前通道状态 0h：允许触发 1h：已经检测到触发，不允许后续触发 当写入时， 0h：无效 1h：清除当前通道状态，并允许新的触发

8.2.18 ADC_EVTRG (事件触发选择寄存器)

- 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										TRG1OE	TRG0OE	RSVD							TRG1SEL					RSVD			TRG0SEL				
0	0	0	0	0	0	0	0	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	RW	RW	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW

Name	Bit	Type	Description
TRG0SEL TRG1SEL	[4:0] [12:8]	RW	TRGEV0，TRGEV1事件的触发源选择。 00000：无触发输出 00001：EOC事件 00010：READY事件 00011：OVR事件 00100：CMP0H事件 00101：CMP0L事件 00110：CMP1H事件 00111：CMP1L事件 01000：SEQ_END[0]事件 01001：SEQ_END[1]事件 01010：SEQ_END[2]事件 01011：SEQ_END[3]事件 01100：SEQ_END[4]事件 01101：SEQ_END[5]事件 01110：SEQ_END[6]事件 01111：SEQ_END[7]事件 10000：SEQ_END[8]事件 10001：SEQ_END[9]事件 10010：SEQ_END[10]事件 10011：SEQ_END[11]事件 10100：SEQ_END[12]事件 10101：SEQ_END[13]事件 10110：SEQ_END[14]事件

			10111：SEQ_END[15]事件
TRG0OE	[20]	RW	触发输出端口ADC_TRGOUT0使能 0h：禁止触发输出 1h：允许触发输出
TRG1OE	[21]	RW	触发输出端口ADC_ TRGOUT1使能 0h：禁止触发输出 1h：允许触发输出

8.2.19 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_DR15 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	Bit	Type	Description
DATA	[20:0]	R	**DATA[20:0]：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。 **注意：** 该寄存器的有效位数跟 ADC_SEQx 的 AVG_SEL 位有关，选择的平均次数小于转换次数时，该寄存器的位数会多于 12 位。

8.2.20 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	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
CMP0	[20:0]	RW	**CMP0[20:0]：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。

8.2.21 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	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
CMP1	[20:0]	RW	**CMP1[20:0]：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。

8.2.22 ADC_DRMASK (禁止转换结果更新寄存器)

Address = Base Address + 0x0148, 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																DRMASK															
0	0	0	0	0	0	0	0	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
DRMASK	[15:0]	RW	**DRMASK：禁止转换结果更新** 如果该位为 1 (MASK)，那么对应的 ADC_DRx 寄存器的转换结果则不会被更新，而是保持 MASK 之前的值。

9 GPIO

9.1 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 B0: 6 位输入/输出端口, PB0.0 ~ PB0.5

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。

注：如果系列内芯片不具有某一外围，那它就不具备该外围的所有资源。具体参考芯片的数据手册。

9.1.1 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
- 管脚可以独立设置驱动能力和斜率控制

9.1.2 管脚描述

Table 10-1 管脚描述

Pin Name	Function	I/O Type	Active Level	Comments
PA0[15:0]	通用 I/O 口 A0	I/O	-	-
PB0[5:0]	通用 I/O 口 B0	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。

9.2 功能描述

9.2.1 电路图

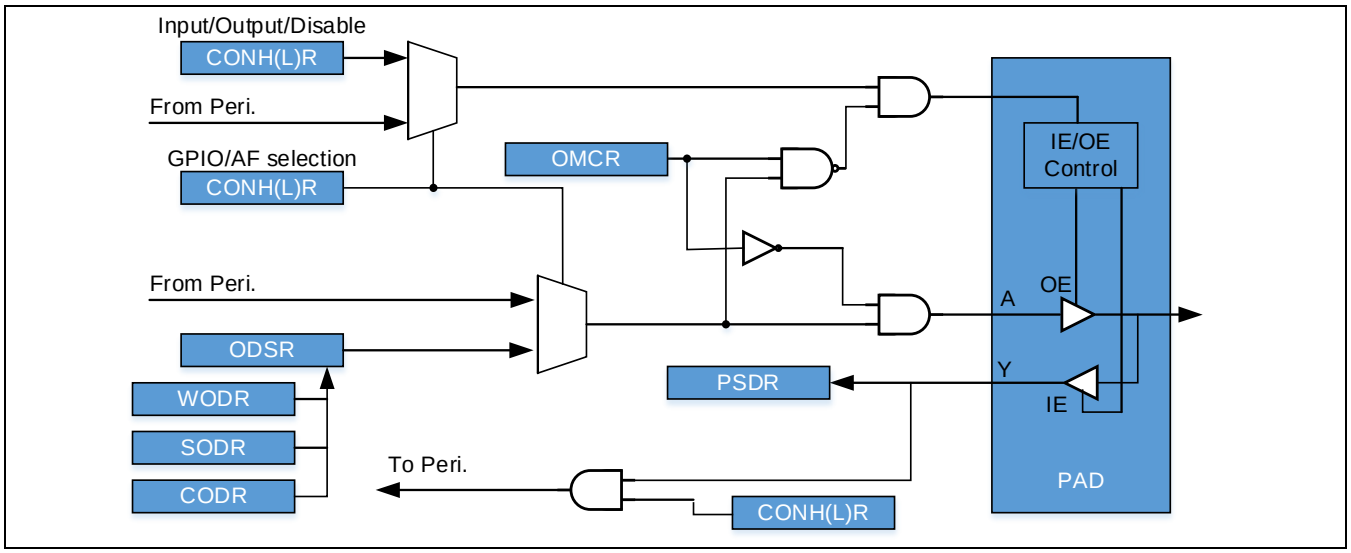


Figure 10-1 GPIO原理图

9.2.2 工作原理

9.2.2.1 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。

9.2.2.2 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，驱动强度和斜率控制功能可通过寄存器 GPIO_DSCR 设置，在缺省模式下，GPIO 设置为较低的驱动能力和较慢的跳变斜率，这样的设置可以提供芯片较好的 EMI 特性。在需要特定 GPIO 提供大电流或者高速通讯能力时，可以对特定 GPIO 的驱动能力和斜率做出调整。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。

9.2.3 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 HCLK 驱动。可以通过断开 GPIO 的时钟来减少功耗。GPIO 的工作时钟通过 CLKEN 寄存器进行配置。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。

9.2.4 输入特性配置

GPIO 的输入缓冲器具有斯密特迟滞特性，缺省条件下兼容 CMOS 电平标准，即高电平的最小输入阈值为 0.7VDD，低电平最大输入阈值为 0.3VDD。为支持 5V 供电条件下，兼容更低输入电平标准，某些 GPIO 具有 TTL 输入选项。具有该选项的 GPIO，可以通过配置 DSCR 寄存器的 SR 控制位选择更低的输入电平阈值，以兼容 TTL 输入标准。需要注意，当 SR 使能的同时，该 GPIO 的输出电压摆率(Slew Rate)也同时被设定为快速模式。

在 TTL 输入特性使能时，某些 GPIO 可以还支持选择两个 TTL 电平的 Option，该 Option 可以将输入的阈值调整到更低的水平。TTL 电平 Option 通过 OMCR 的 CCM 控制位进行选择。查询具有该特性的 GPIO 具体参考 PIN ASSIGNMENT SPEC。

9.2.5 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。

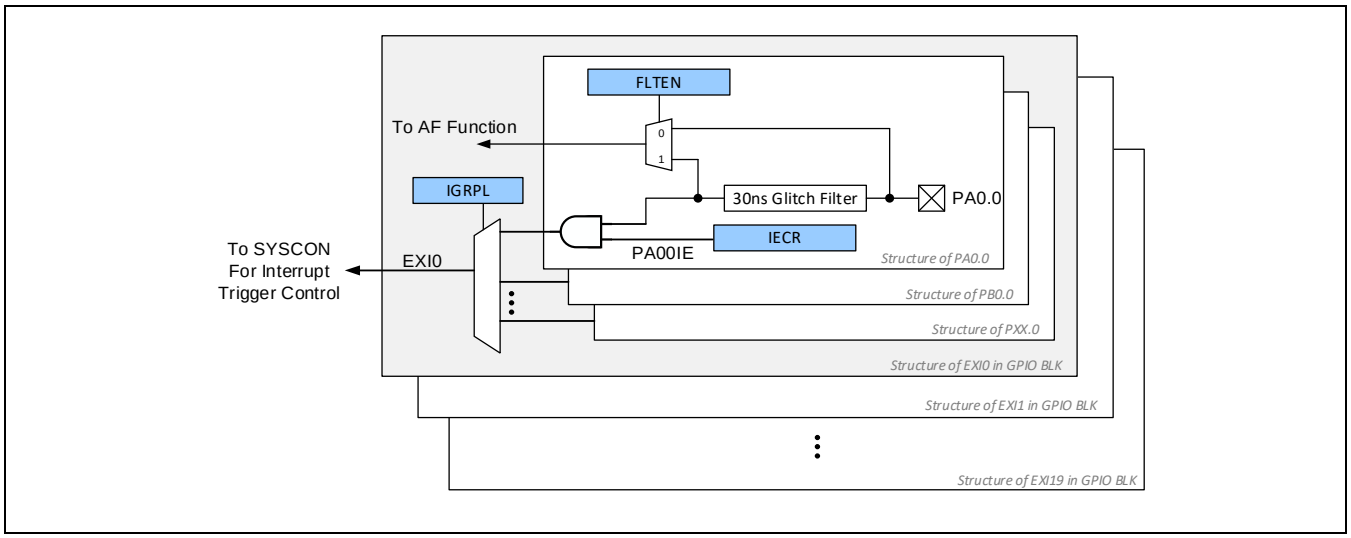


Figure 10-2 GPIO外部中断原理图

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。

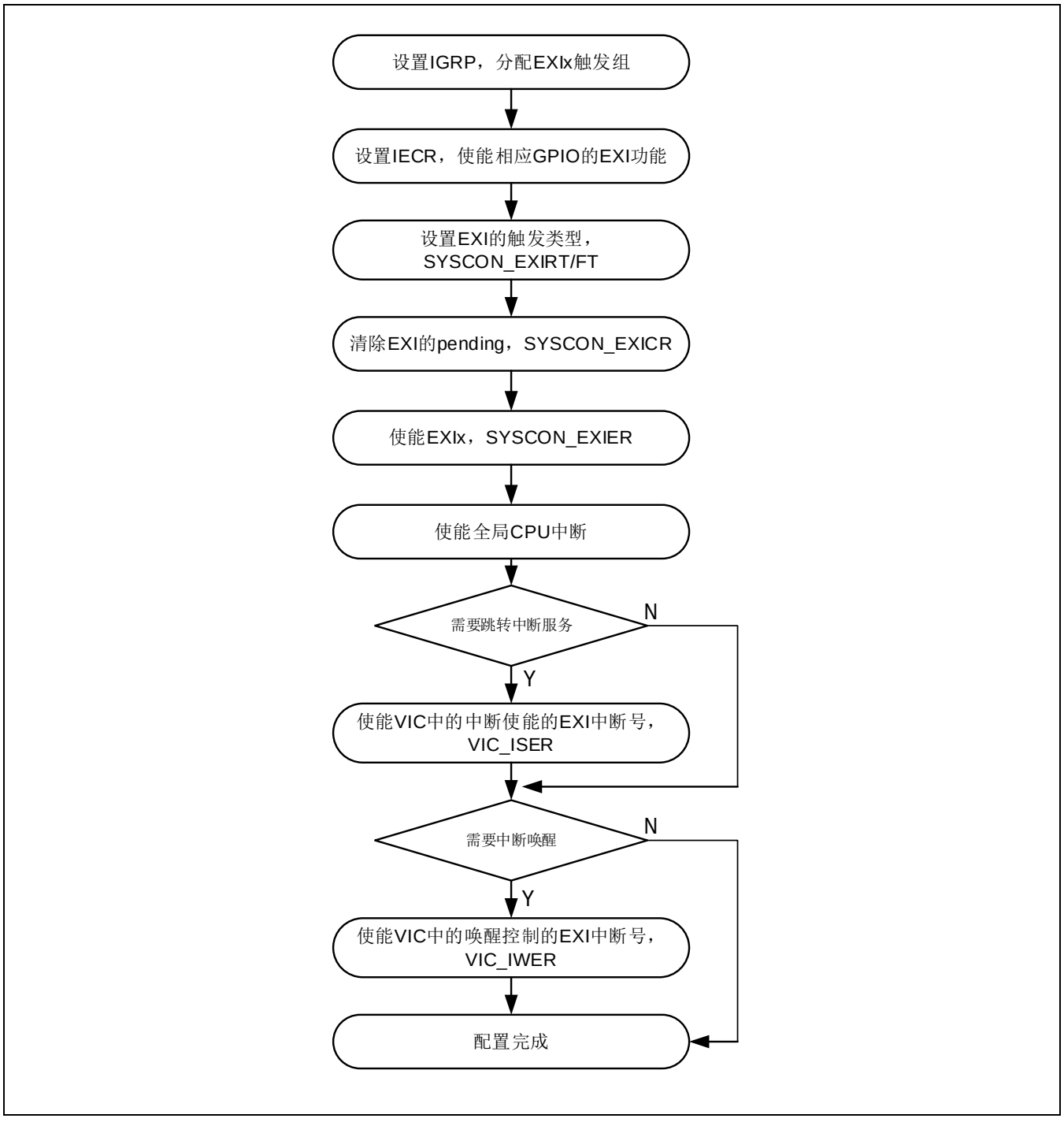


Figure 10-3 GPIO外部中断配置流程

9.3 寄存器说明

9.3.1 寄存器表

- Base Address of A0: 0x6000_0000
- Base Address of B0: 0x6000_2000

Register	Offset	Description	Reset Value
GPIO_CONLR	0x00	低位控制寄存器	0x0000_0000(1)
GPIO_CONHR	0x04	高位控制寄存器	0x0000_0000(1)
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
GPIO_FLTEN	0x1C	输入信号滤波器使能控制寄存器	0x0000_0000
GPIO_PUDR	0x20	上拉/下拉配置寄存器	0x0000_0000(1)
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：

（1）SWD 接口和外部复位管脚的缺省复位值随 SWD 接口的上电配置和外部复位的使能状态有所区别。

（2）GPIO 通过 AHB 总线进行控制，可以通过设置 GPIO_CLKEN 寄存器关闭指定 GPIO 组的控制时钟，时钟关闭后，该 GPIO 组不能进行配置更改。

- Base Address of GPIO_IGRP: 0x6000_F000

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
GPIO_IGREX	0x08	外部中断组扩展配置寄存器	0x0000_0000
GPIO_CLKEN	0x0C	GPIO组时钟使能控制寄存器	0x0000_0000

9.3.2 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
P0	[3:0]	R/W	IO管脚0的模式配置。
P1	[7:4]	R/W	IO管脚1的模式配置。
P2	[11:8]	R/W	IO管脚2的模式配置。
P3	[15:12]	R/W	IO管脚3的模式配置。
P4	[19:16]	R/W	IO管脚4的模式配置。
P5	[23:20]	R/W	IO管脚5的模式配置。
P6	[27:24]	R/W	IO管脚6的模式配置。
P7	[31:28]	R/W	IO管脚7的模式配置。

GPIO 模式控制位

0h：GPD (GPIO Disabled)，当前 GPIO 输入输出禁止模式，即高阻态 (默认模式)。

1h：GPI (GPIO Input)，当前 GPIO 设置为输入模式。

2h：GPO (GPIO Output)，当前 GPIO 设置为输出模式，输入禁止。

3h：GPO (GPIO Output)，当前 GPIO 设置为输出模式，输出监测使能（输入 Buffer 使能）。

4h ~15h：AFx (x从‘1’开始)，功能复用模式（参见对应型号数据手册中《管脚配置》章节）。

9.3.3 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
P8	[3:0]	R/W	IO管脚8的模式配置。
P9	[7:4]	R/W	IO管脚9的模式配置。
P10	[11:8]	R/W	IO管脚10的模式配置。
P11	[15:12]	R/W	IO管脚11的模式配置。
P12	[19:16]	R/W	IO管脚12的模式配置。
P13	[23:20]	R/W	IO管脚13的模式配置。
P14	[27:24]	R/W	IO管脚14的模式配置。
P15	[31:28]	R/W	IO管脚15的模式配置。

GPIO 模式控制位

0h：GPD (GPIO Disabled)，当前 GPIO 输入输出禁止模式，即高阻态 (默认模式)。

1h：GPI (GPIO Input)，当前 GPIO 设置为输入模式。

2h：GPO (GPIO Output)，当前 GPIO 设置为输出模式，输入禁止。

3h：GPO (GPIO Output)，当前 GPIO 设置为输出模式，输出监测使能（输入 Buffer 使能）。

4h ~15h：AFx (x从‘1’开始)，功能复用模式（参见对应型号数据手册中《管脚配置》章节）。

9.3.4 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
Px	[x]	W	端口x 输出数据控制位 0h：对应管脚置‘0’，低电平。 1h：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。

9.3.5 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
Px	[x]	W	端口x 输出置1 0h：无效果 1h：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。

9.3.6 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
Px	[x]	W	端口x 的输出清零 0h：无效果 1h：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。

9.3.7 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
Px	[x]	R	端口x 输出状态 0h：对应管脚当前输出缓冲区为‘0’，低电平。 1h：对应管脚当前输出缓冲区为‘1’，高电平。

9.3.8 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
Px	[x]	R	端口x 输入状态 0h：对应管脚当前输入缓冲区为‘0’，低电平。 1h：对应管脚当前输入缓冲区为‘1’，高电平。

9.3.9 GPIO_FLTEN (输入信号滤波器使能控制寄存器)

- 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																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	1	1	1	1	1	1	1	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	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
Px	[x]	R	端口 x 输入信号滤波器使能控制位，该滤波器为 30ns 模拟滤波器 0h：旁路对应管脚输入滤波器。 1h：使能对应管脚输入滤波器。

9.3.10 GPIO_PUDR (上拉/下拉配置寄存器)

- Address = Base Address + 0x0020, Reset Value = 0x0X00_XX0X（GPIOA0）；0x0000_0000（GPIOB0）

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

P0	[1:0]	RW	上拉/下拉IO 管脚0

'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
注意：
1、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。

2、复位值标注“x”，是因为被选中的芯片 SWD 口上电默认上拉使能。

9.3.11 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
P15	[31:30]	RW	IO 管脚15驱动强度配置
P14	[29:28]	RW	IO 管脚14驱动强度配置
P13	[27:26]	RW	IO 管脚13驱动强度配置
P12	[25:24]	RW	IO 管脚12驱动强度配置
P11	[23:22]	RW	IO 管脚11驱动强度配置
P10	[21:20]	RW	IO 管脚10驱动强度配置
P9	[19:18]	RW	IO 管脚9驱动强度配置
P8	[17:16]	RW	IO 管脚8驱动强度配置
P7	[15:14]	RW	IO 管脚7驱动强度配置
P6	[13:12]	RW	IO 管脚6驱动强度配置
P5	[11:10]	RW	IO 管脚5驱动强度配置
P4	[9:8]	RW	IO 管脚4驱动强度配置
P3	[7:6]	RW	IO 管脚3驱动强度配置
P2	[5:4]	RW	IO 管脚2驱动强度配置
P1	[3:2]	RW	IO 管脚1驱动强度配置
P0	[1:0]	RW	IO 管脚0驱动强度配置

每个 IO 通过两个 bit 分别设置驱动能力和斜率控制。

注意：DSCR 的设置是独立于 CONLR 和 CONHR 的。

BIT0: 仅当 IO 做输出时有效，用于控制驱动能力。0 – 弱驱动能力；1 – 强驱动能力

BIT1： IO 做输入或输入时有不同的控制意义。

IO 做输入时，用于控制输入逻辑电平判断值。0 – CMOS 输入；1 – TTL 输入 （OMCR 里可以选择 TTL 电平）

IO 做输出时，用于控制驱动斜率。0 – 慢速；1- 快速。

NOTE: CMOS 或者 TTL 输入特性选择只有在支持 TTL 输入的 I/O 有效。

9.3.12 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
CCM15	CCM14	CCM13	CCM12	CCM11	CCM10	CCM9	CCM8	CCM7	CCM6	CCM5	CCM4	CCM3	CCM2	CCM1	CCM0	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
RW	RW	RW	RW	RW	RW	RW	RW	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
ODPx	[x]	RW	端口 x 开漏使能/禁止。 0h：GPIO管脚x不处于开漏输出模式 (推挽输出模式)。 1h：GPIO管脚x处于开漏输出模式。
CCMx	[x]	RW	端口 x TTL 输入电平选择。 0h：选择TTL1输入特性。 1h：选择TTL2输入特性。

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。

9.3.13 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
IENx	[x]	RW	端口 x 外部中断使能/禁止 0h：外部中断禁止 1h：外部中断使能

NOTE:

9.3.14 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	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
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

NOTE:

该寄存器为只写寄存器

9.3.15 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	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
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

NOTE:

该寄存器为只写寄存器

9.3.16 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
GRP7	[30:28]	RW	外部中断组7源选择
GRP6	[26:24]	RW	外部中断组6源选择
GRP5	[22:20]	RW	外部中断组5源选择
GRP4	[18:16]	RW	外部中断组4源选择
GRP3	[14:12]	RW	外部中断组3源选择
GRP2	[10:8]	RW	外部中断组2源选择
GRP1	[7:4]	RW	外部中断组1源选择
GRP0	[3:0]	RW	外部中断组0源选择
0000: GPIOA0.x 被选中 0001: GPIOA1.x 被选中 0010: GPIOB0.x 被选中 0011: GPIOB1.x 被选中 Other: 保留 ‘x’ 表示组数			

9.3.17 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
GRP15	[30:28]	RW	外部中断组15源选择
GRP14	[26:24]	RW	外部中断组14源选择
GRP13	[22:20]	RW	外部中断组13源选择
GRP12	[18:16]	RW	外部中断组12源选择
GRP11	[14:12]	RW	外部中断组11源选择
GRP10	[10:8]	RW	外部中断组10源选择
GRP9	[6:4]	RW	外部中断组9源选择
GRP8	[2:0]	RW	外部中断组8源选择
0000: GPIOA0.x 被选中 0001: GPIOA1.x 被选中 0010: GPIOB0.x 被选中 0011: GPIOB1.x 被选中 Other: 保留 ‘x’ 表示组数			

9.3.18 GPIO_IGREX (外部中断组扩展配置寄存器)

- 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																GRP19				GRP18				BGR17				GRP16			
0	0	0	0	0	0	0	0	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
GRP16	[3:0]	RW	选择外部中断组 16 0h：GPIOA0.0 被选中 1h：GPIOA0.1 被选中 2h：GPIOA0.2 被选中 3h：GPIOA0.3 被选中 4h：GPIOA0.4 被选中 5h：GPIOA0.5 被选中 6h：GPIOA0.6 被选中 7h：GPIOA0.7 被选中 其他：GPIOA0.0 被选中
GRP17	[7:4]	RW	选择外部中断组 17 配置同GPR16控制位
GRP18	[11:8]	RW	选择外部中断组 18 0h：GPIOB0.0 被选中 1h：GPIOB0.1 被选中 2h：GPIOB0.2 被选中 3h：GPIOB0.3 被选中 其他：保留
GRP19	[15:12]	RW	选择外部中断组 19 配置同GPR18控制位

9.3.19 GPIO_CLKEN (GPIO组时钟使能控制寄存器)

- 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																										RSVD	RSVD	RSVD	CLK_B0	CLK_A1	CLK_A0
0	0	0	0	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
R	R	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

Name	Bit	Type	Description
CLK_A0 CLK_A1 CLK_B0	[3:0]	RW	GPIO 组控制时钟使能/禁止 0h：禁止控制时钟 1h：使能控制时钟

10 基本计数器 (Basic Timer)

10.1 概述

基本型计数器（Basic Timer）是一个 16 位计数器。Timer 工作在递增模式下，并支持自动重载功能。Basic Timer 可提供基础定时/计数功能和简单的 PWM 波形输出。

注：如果系列内芯片不具有本外围，那它就不具备本章说述的相关资源。具体参考芯片的数据手册。

10.1.1 主要特性

- 16 位可编程递增计数器。
- 16 位预设计数器时钟分频器 (支持 On-the-fly 修改配置)。
- 一个比较值寄存器，支持 PWM 波形输出。
- 支持通过 ETCB 进行硬件自动同步触发和外部计数。

10.1.2 管脚描述

Table 11-1 BT 相关功能管脚描述

管脚名称	功能描述
BT_OUT	PWM 波形输出

10.2 功能描述

10.2.1 模块框图

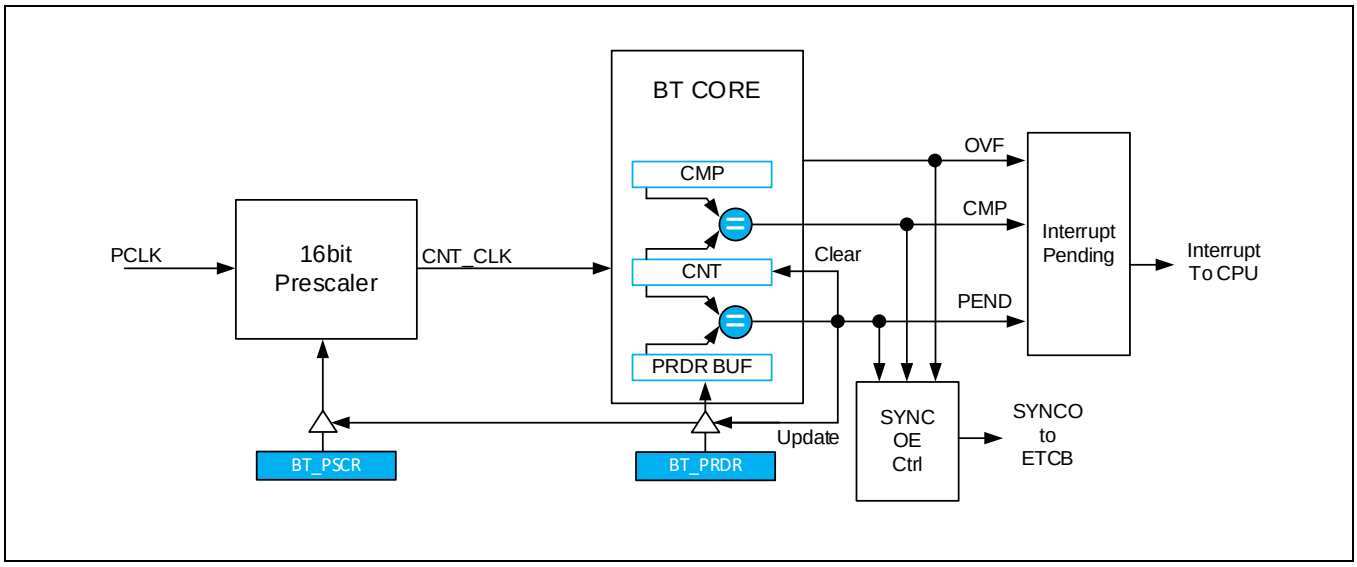


Figure 11-1 模块结构示意图

10.2.2 基本功能描述

Basic Timer是一个自动重载的16位递增计数器。计数器从零开始计数，当计数值等于PRDR的设定值时，会自动在下一个时钟重置为零，重新开始计数。自动重载功能可以通过CR[CNTRLD]关闭，当禁止自动重载时，计数器在计数过程中不会被重置，直到计数溢出后重新从零开始计数。

BT的计数器的启动可以通过两种方式触发：

- 软件触发：通过对RSSR[START]控制位写'1'
- 硬件触发：通过ETCB触发。在使能ETCB相应通道的同时，需要将CR[SYNCEN]使能。

当清楚START控制位时，可以停止BT的工作。START控制位具有SHADOW功能，当START的SHADOW使能时，START被清除后，BT不会立即停止工作，而是等计数器完成当前周期计数后（CNT=PRDR后的下一个周期）才停止工作。当START的SHADOW功能被禁止，则START一旦被清除BT就立即停止工作。START的SHADOW功能通过CR[SHDWSTP]控制位进行设置。PRDR和PSCR寄存器同样具有SHADOW寄存器，对PRDR和PSCR的写入被保存在SHADOW寄存器中。

当下列事件发生时，SHADOW寄存器的内容将会被加载到其对应的ACTIVE寄存器中。

- 周期事件(PEND)发生时
- 软件强制更新，写CR[UPDATE]控制位
- 通过外部触发事情进行更新（CR[SYNCMD]控制位可以选择是否在外部触发时对寄存器进行更新）。

10.2.3 工作模式

Basic Timer支持两种工作模式：连续计数模式和一次性计数模式。

在连续计数模式下，计数器从零开始计数，直到计数周期结束或者计数器溢出。当计数器值等于周期设置值或者溢出时，计数器会在下一个计数周期自动清零后，重新开始计数。软件通过CR[OPM]控制位进行模式选择。当CR[CNTRLD]被禁止时，OPM无效。

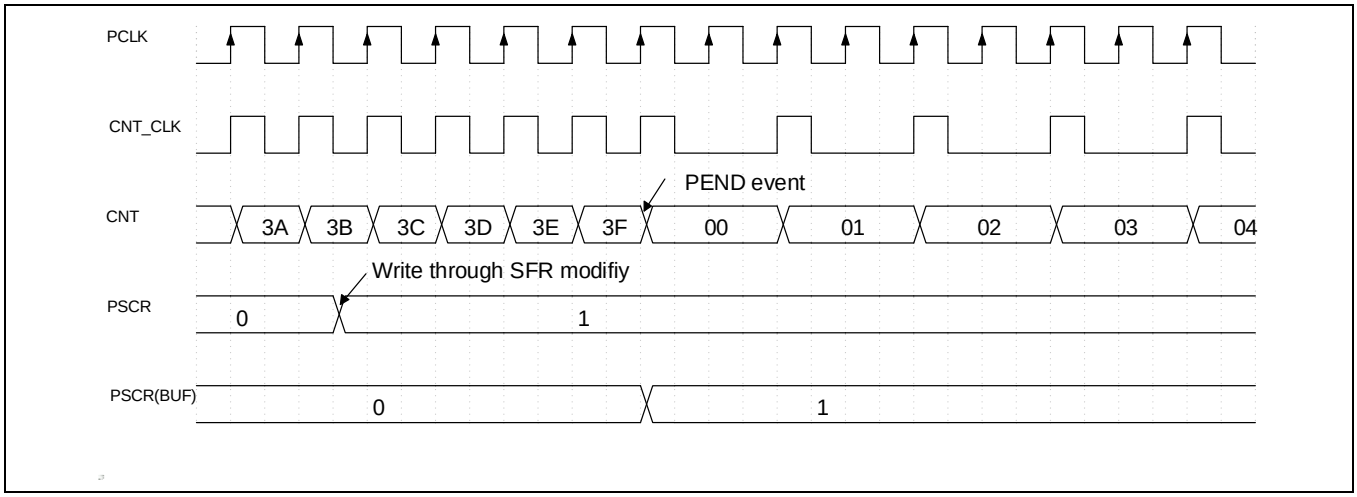


Figure 11-2 BT 周期计数模式

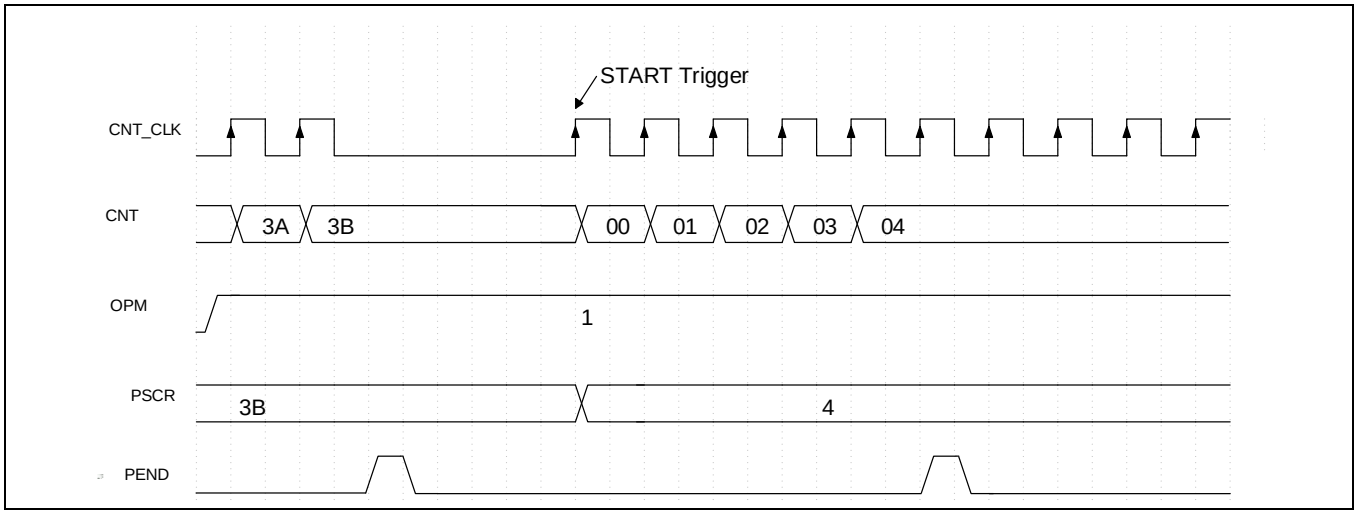


Figure 11-3 BT One Pulse 计数模式

10.2.4 同步触发和事件触发

Basic Timer可以通过ETCB和其他片上模块进行通信。

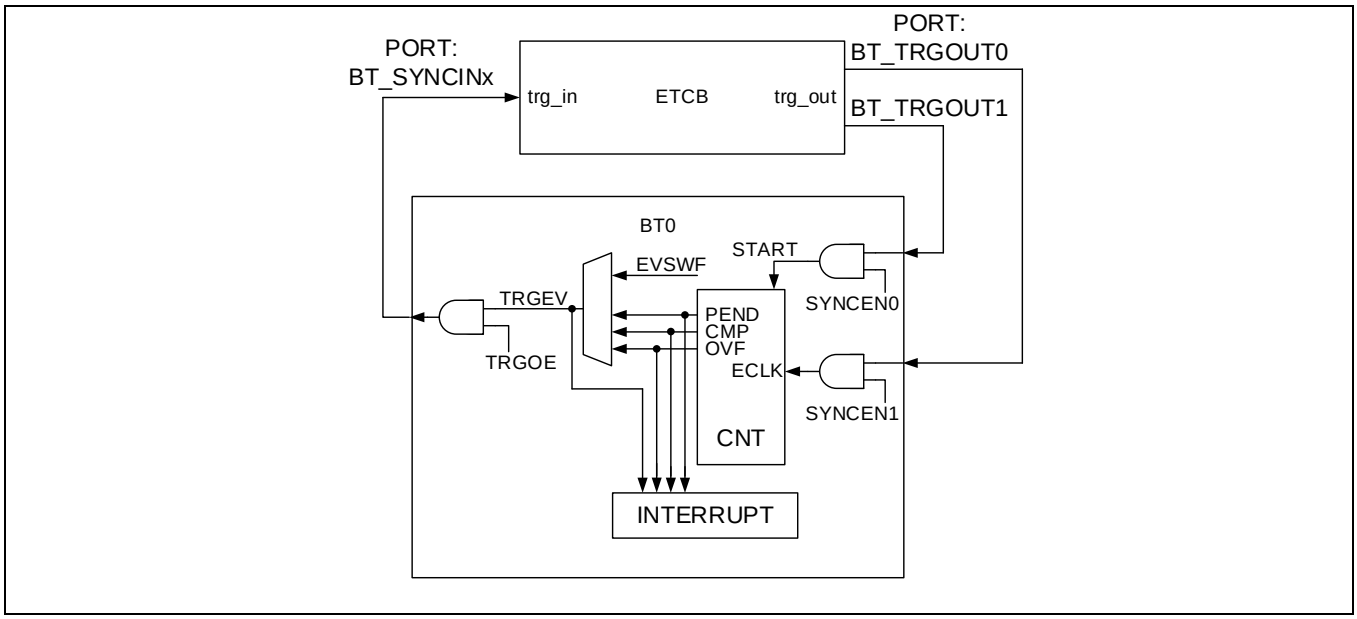


Figure 11-4 同步触发原理

10.2.4.1 同步触发(输入)

Basic Timer有两个同步输入端口可以通过ETCB的桥接接收来自其他模块的事件触发信号。

SYNC PORT0：同步输入端口0可以触发Basic Timer的START控制。

SYNC PORT1：同步输入端口1可以触发Basic Timer的计数值增加一拍。

10.2.4.2 事件触发（输出）

Basic Timer有一个事件触发输出端口，可以通过ETCB的桥接向其他模块输出触发事件。

触发输出通过EVTRG控制寄存器可以选择BT中断触发信号中的任意一个作为触发输出。

通过软件写EVSWF寄存器，可以强制产生一个TRGEV触发输出信号。该功能可以用于调试触发通路或者在需要软件控制触发输出的应用中采用。

10.2.5 波形发生

Basic Timer支持PWM波形输出功能，通过CMP寄存器、PRDR寄存器、CR[IDLEST]和CR[STARTST]控制位可以设置不同的输出波形。

当BT启动时，BT_OUT的输出状态由CR[STARTST]的控制位决定；当PEND事件或者CMP事件发生时，BT_OUT的输出状态将被反转。当BT处于IDLE时（未启动或者被停止），BT_OUT的状态由CR[IDLEST]控制位决定。在OPM模式下，当计数器被挂起时，BT_OUT保持PEND事件后的输出状态，此时计数器仍旧处于工作状态，只有清除RSSR[START]控制位后，输出才由CR[IDLEST]的控制位决定。

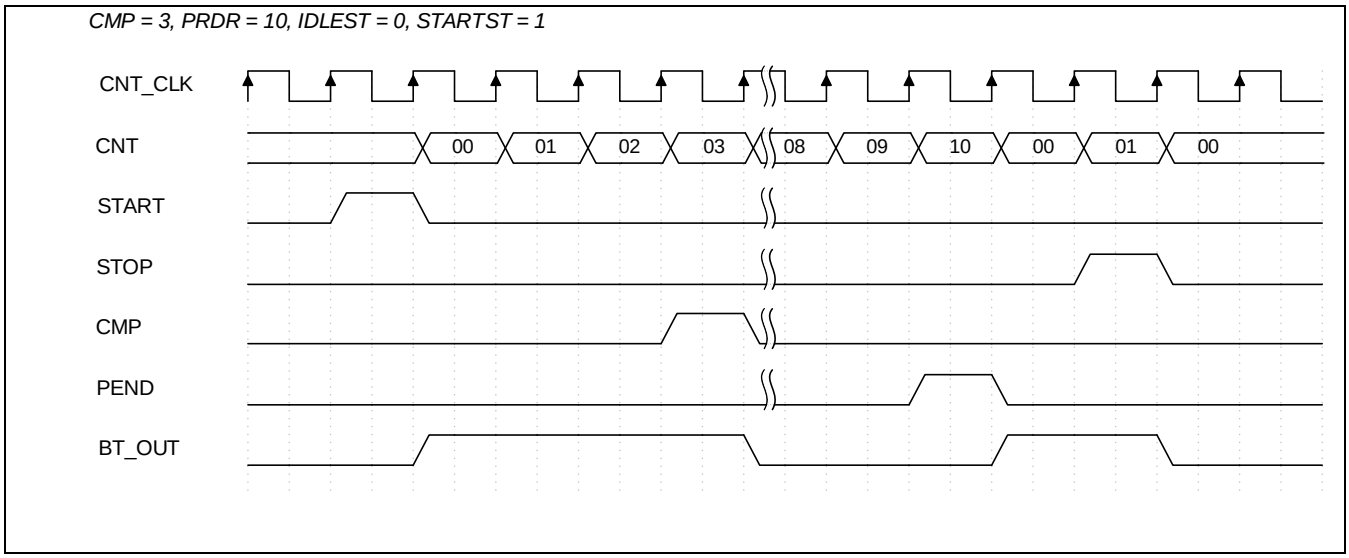


Figure 11-5 BT 输出波形时序图

10.2.6 中断控制

Basic Timer支持4种中断，中断触发信号可以作为同步脉冲输出给ETCB，或者用于产生CPU中断请求。中断事件一旦发生，无论中断是否使能，其相对应的RISR标志位都会被置位。当IMCR控制器中的相应位被使能时，该标志位可以产生CPU中断请求。

PEND事件：计数器周期结束时发生。

CMP事件：计数器计数值等于CMP寄存器设置时发生。

OVF事件：计数器计数溢出（0xFFFF）时发生。

TRGEV事件：同步触发输出事件有输出时发生。

10.3 寄存器说明

10.3.1 寄存器列表

- Base Address：BT0：0x4005_1000

  BT1：0x4005_2000

Register	Offset	Description	Reset Value
BT_RSSR	0x000	Reset/Start Control Register	0x00000000
BT_CR	0x004	General Control Register	0x00000000
BT_PSCR	0x008	Counter Clock Prescaler Register	0x00000000
BT_PRDR	0x00C	Period Register	0x00000000
BT_CMP	0x010	Compare Data	0x00000000
BT_CNT	0x014	Counter Register	0x00000000
BT_EVTRG	0x018	Event Generation Control Register	0x00000000
BT_EVSWF	0x024	Event Counter Software Trigger Register	0x00000000
BT_RISR	0x028	Raw Interrupt Status Register	0x00000000
BT_IMCR	0x02C	Interrupt Masking Control Register	0x00000000
BT_MISR	0x030	Masked Interrupt Status Register	0x00000000
BT_ICR	0x034	Interrupt Clear Register	0x00000000

10.3.2 BT_RSSR (启动停止控制寄存器)

- Address = Base Address +0x000, 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																SRR				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	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	R	RW

Name	Bit	Type	Description
START	[0]	RW	计数器启动控制。 0h：写入'0'时，停止计数器。 1h：写入'1'时，启动计数器。 读取时，返回当前计数器的工作状态。 0h：计数器处于IDLE状态。 1h：计数器处于工作状态。
SRR	[15:12]	RW	软件复位控制位。 当对当前控制位写入‘0x5’时，BT模块会被复位。复位后，所有寄存器都恢复为RESET状态。

10.3.3 BT_CR (控制寄存器)

- Address = Base Address +0x004, 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															CNTRLD	SYNCMD	AREARM	REARM1	REARM0	OSTMD1	OSTMD0	SYNCEN1	SYNCEN0	STARTST	IDLEST	EXTCKM	OPM	SHDWSTP	UPDATE	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	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
CLKEN	[0]	RW	时基计数器的时钟使能控制。 0h：计数器计数时钟禁止。 1h：计数器计数时钟使能。
DBGEN	[1]	RW	调试使能控制。调试使能时，在CPU被调试器挂起时，时基计数器的计数时钟同时也被挂起。 0h：调试禁止 1h：调试使能
UPDATE	[2]	RW	PRDR和PSCR软件强制更新。当对UPDATE写入‘1’时，PRDR和PSCR的Shadow寄存器内容将被载入到活动寄存器中。 0h: 无效. 1h: 触发更新
SHDWSTP	[3]	RW	START控制位的Shadow功能使能控制。START置位不受此位控制，清除时受此位控制。当选择Shadow模式时，START控制位在周期结束时清除。 0h：Shadow模式 1h：Immediate模式
OPM	[4]	RW	计数器单次触发工作模式选择。 0h：连续计数工作模式 1h：单次触发工作模式
EXTCKM	[5]	RW	计数器时钟源选择。 0h：计数器基于PCLK的分频计数 1h：由同步触发端口触发计数

IDLEST	[6]	RW	BT停止计数时，BT_OUT状态设置。 0：低电平 1：高电平
STARTST	[7]	RW	BT开始计数时，BT_OUT状态设置。 0：低电平 1：高电平
SYNCENx	[9:8]	RW	外部同步触发输入使能控制。 0h：禁止外部触发 1h：使能外部触发
OSTMDx	[11:10]	RW	一次性同步触发模式选择。 0h：连续触发模式 1h：一次性触发模式 当该输入通道被设置为一次性触发模式后，在一次触发事件被检测到后，该通道将不允许后续的触发事件通过，直到被软件重置（REARM）后才允许新的触发事件通过。
REARMx	[13:12]	RW	在一次性同步触发模式下，软件重置当前通道状态控制位。 当读取时，返回当前通道状态 0h：允许触发 1h：已经检测到触发，不允许后续触发 当写入时， 0h：无效 1h：清除当前通道状态，并允许新的触发
AREARM	[14]	RW	硬件自动REARM控制位。 0：禁止硬件自动REARM 1：计数器周期结束时，自动REARM
SYNCMD	[15]	RW	同步触发结果控制位。 0h：同步触发发生时，计数器重置并触发所有具有缓存（Shadow）的寄存器将缓存内容更新到活动寄存器中。 1h：同步触发发生时，只是计数器重置。
CNTRLD	[16]	RW	硬件自动重载CNT值控制位。 0：当CNT计数值等于PRDR时，CNT自动清零，重新开始计数。 1：不进行CNT重载，计数器一直计数直到溢出后重新开始计数。

10.3.4 BT_PSCR (时钟分频寄存器)

- 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																PSCR															
0	0	0	0	0	0	0	0	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
PSCR	[15:0]	RW	时基控制周期寄存器。 BT的计数器时钟频率为PCLK/(PSCR+1)

10.3.5 BT_PRDR (周期寄存器)

- 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
CMPLINK																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	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
PRDR	[15:0]	RW	时基控制周期寄存器。 当计数器计数值等于PRDR的设置值时，下一个计数周期计数器将从零开始计数。
CMPLINK	[31]	W	CMP寄存器同步写入控制。 当对PRDR进行更新时，若该控制位同时写入'1'时，则CMP寄存器同时被更新成和PRDR一样的值。

10.3.6 BT_CMP (比较值寄存器)

- 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																CMP															
0	0	0	0	0	0	0	0	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
CMP	[15:0]	RW	比较值寄存器。 当CNT值等于CMP时，下个计数周期开始BT输出将翻转。

10.3.7 BT_CNT (计数器当前计数值寄存器)

- 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																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
CNT	[15:0]	RW	当前计数器计数值寄存器。 对CNT读取时，返回当前计数器值。对CNT写入时，将直接更新CNT的计数值。 CNT计数器没有Shadow寄存器，CPU的写入将直接影响当前计数器值。

10.3.8 BT_EVTRG (事件触发控制寄存器)

- Address = Base Address +0x00418, 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											TRGOE	RSVD																TRGSEL			
0	0	0	0	0	0	0	0	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	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
TRGSEL	[3:0]	RW	TRGEV事件的触发源选择控制位。 0h：禁止TRGOUT的触发。 1h：指定PEND事件用于产生TRGEV事件。 2h：指定CMP事件用于产生TRGEV事件。 3h：指定OVF事件用于产生TRGEV事件。 其他：保留
TRGOE	[20]	RW	外部触发端口TRGOUT输出使能。 0h：禁止触发输出到ETCB。 1h：允许触发输出到ETCB。

10.3.9 BT_EVSWF (事件触发软件触发寄存器)

- Address = Base Address +0x00424, 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																															EVSWF
0	0	0	0	0	0	0	0	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
EVSWF	[0]	RW	软件产生一次TRGEV事件。 0h：写入'0'无效。 1h：软件产生一次TRGEV事件。

10.3.10 BT_RISR (原始中断状态寄存器)

- Address = Base Address +0x00828, 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																												TRGEV	OVF	CMP	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	Type	Description
PEND	[0]	R	PEND周期结束中断请求原始标志状态。
CMP	[1]	R	CMP Match中断请求原始标志状态。
OVF	[2]	R	OVF中断请求原始标志状态。
TRGEV	[3]	R	事件触发中断请求原始标志状态。
原始中断标志表示中断事件发生，通过设置IMCR中相应位，可以允许该中断请求CPU中断。原始中断标志位需要通过软件清除。 0h：该中断未置位 1h：该中断已置位			

10.3.11 BT_IMCR (中断使能寄存器)

- Address = Base Address +0x0082C, 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																												TRGEV	OVF	CMP	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	RW	RW	RW	RW

Name	Bit	Type	Description
PEND	[0]	RW	PEND中断使能控制位。
CMP	[1]	RW	CMP Match中断使能控制位
OVF	[2]	RW	OVF中断使能控制位。
EVTRG	[3]	RW	事件触发中断中断使能控制位。
CPU中断请求使能控制。当该控制位使能时，允许触发CPU中断。 0h：禁止该中断 1h：允许该中断			

10.3.12 BT_MISR (中断使能寄存器)

- Address = Base Address +0x00830, 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																												TRGEV	OVF	CMP	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	Type	Description
PEND	[0]	R	PEND周期结束中断请求标志状态。
CMP	[1]	R	CMP Match中断请求标志状态。
OVF	[2]	R	OVF中断请求标志状态。
EVTRG	[3]	R	事件触发中断请求标志状态。
中断标志表示CPU中断请求的状态，通过写ICR寄存器可以清除该标志位。 0h：该中断未置位 1h：该中断已置位			

10.3.13 BT_ICR (中断使能寄存器)

- Address = Base Address +0x00834, 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																												TRGEV	OVF	CMP	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	W	W	W	W

Name	Bit	Type	Description
PEND	[0]	W	清除PEND原始中断状态位。
CMP	[1]	W	清除CMP Match原始中断状态位。
OVF	[2]	W	清除OVF原始中断状态位。
EVTRG	[3]	W	清除事件触发中断原始中断状态位。
中断清除控制位。 对该寄存器写‘0’时，无效；对该寄存器写‘1’时，清除相应中断标志位 读取时，总是返回‘0’			

11 计数器A

11.1 概述

本章节介绍计数器 A，一个 16 位的计数器，用来产生载波频率。

注：如果系列内芯片不具有本外围，那它就不具备本章说述的相关资源。具体参考芯片的数据手册。

11.1.1 主要特性

- 一个普通定时器，在特定时间产生一个计数器A中断
- 16位递减计数器，支持自动重载功能
- 软件/硬件可配置的输出使能/禁止控制
- 输出波形单周期内，高低电平的脉冲宽度可配置

注意：CPU 时钟必须比计数器 A 的时钟快。

11.2 功能描述

11.2.1 模块框图

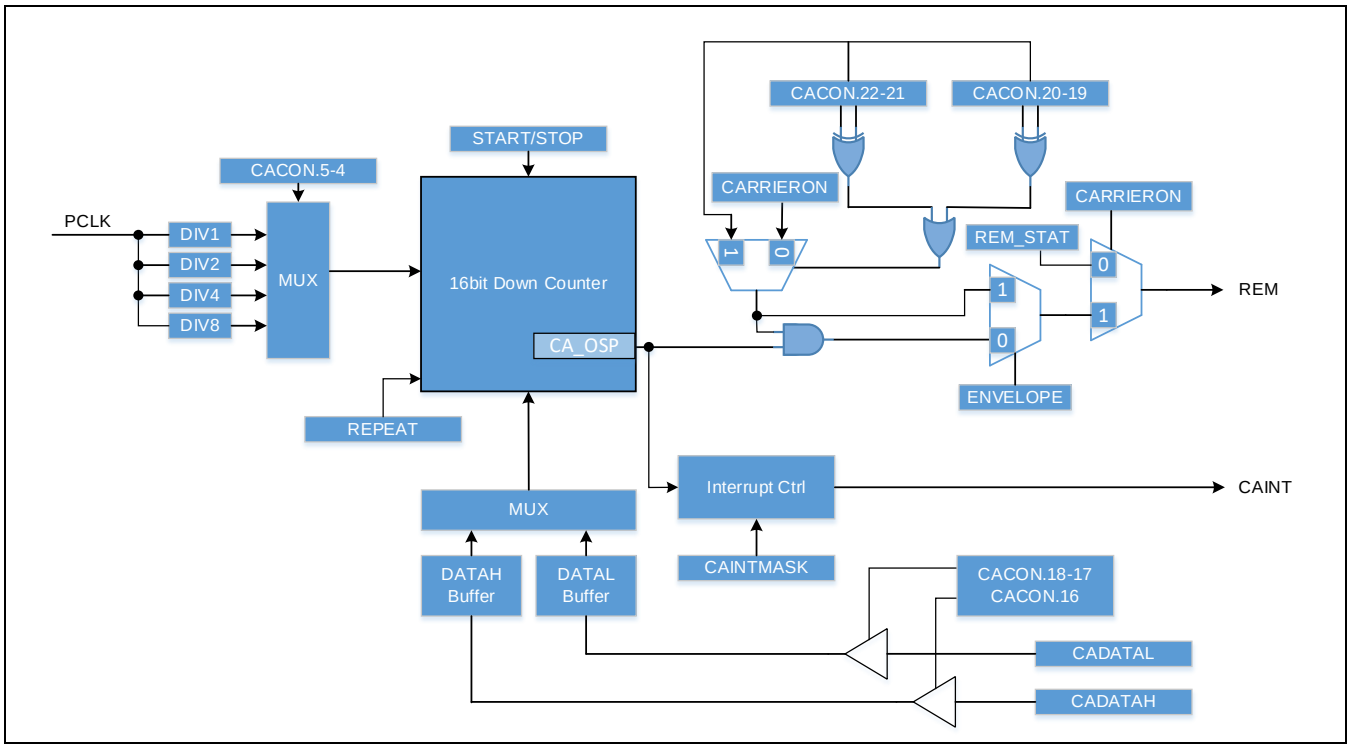


Figure 12-1 计数器A模块框图

11.2.2 工作原理

计数器 A 可以用来产生一个载波频率，支持两种输出模式，一种是包络输出，另一种是载波输出，由 CACON 寄存器的 ENVELOP 位控制。REM 管脚的输出由包络模式设置和载波打开/关闭的设置两者共同决定。当选择载波输出模式，但载波输出又被关闭时，REM 的输出状态由 CACON 寄存器里的 REM_STAT 位决定。

Table 12-1 REM管脚输出状态

ENVELOP	CARRIERON	REM_STAT	REM
OFF	ON	X	CAOUT
OFF	OFF	H	H
OFF	OFF	L	L
ON	ON	X	H
ON	OFF	X	L

注意：

1) 如果 TCMATCH_REM_ONOFF 为 1，那么在 TC 匹配中断发生时，REM 输出状态将被改变。

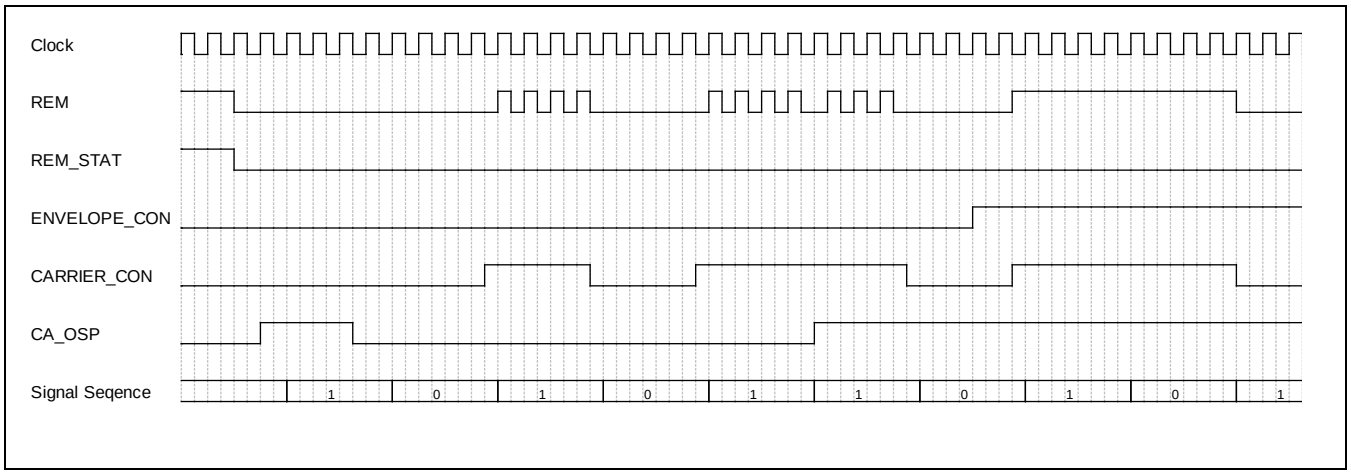


Figure 12-2 计数器A输出信号波形

计数器 A 可以工作在硬件自动触发模式，让 REM 的输出跟 BT0 的中断事件同步，只要有 BT0 的脉冲匹配中断事件和周期结束中断事件即可，不需要使能中断。当通过 BT0 在工作，并且 CACON 寄存器中的 BTMATCH(BTPEND)_REM_CON 被置位后，BTMATCH_REM_CON 和 BTPEND_REM_CON 的配置会覆盖 CACON 中 CARRIER 位的配置。如果 BTMATCH_REM_CON 置 1，当 BT0 脉冲匹配中断发生后，载波会被使能或被禁止。如果 BTPEND_REM_CON 置 1，当 BT0 周期结束中断发生后，载波会被使能或者禁止。如果两个硬件触发源都没有使能，那么载波输出由软件(CARRIER 位)控制。

在设置 HW_STROBE_DATA 后，计数值的更新也可以由硬件自动更新，这个 HW_STROBE_DATA 位不会覆盖 SW_STROBE_DATA。当 REM 变为高的时候，计数值将会被更新到计数器中。如果选择了 TC 脉冲匹配中断，那么当 TC 脉冲匹配中断发生时，计数值会被更新到计数器中。如果选择了 TC 周期结束中断，那么当 TC 周期结束中断发生时，计数值会被更新到计数器中。当两种 TC 中断都被选择时，两个中断发生的时候都会更新计数值。当 SW_STROBE_DATA 被置 1 时，计数值也会更新到计数器中，而且更新状态(是否更新完成)可以读回 SW_STROBE_DATA 位来进行查询。另外，每次写 START 位的时候，计数值会被自动更新到计数器中。

REM 输出载波波形的极性可以由 CACON 寄存器的 OSP 位控制。OSP 只有当 ENVELOP 是 0 并且 CARRIER 是 1 的时候才能改变波形的极性。

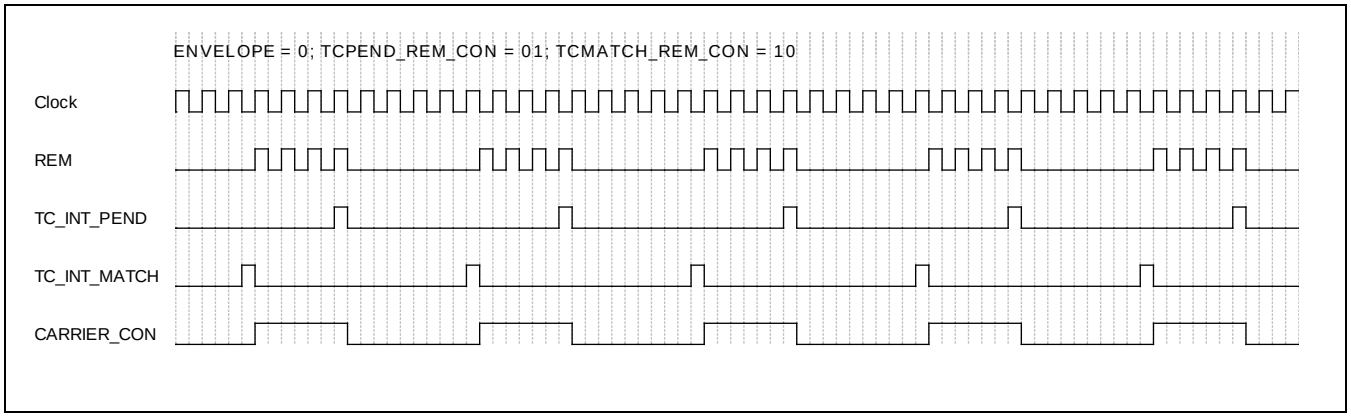


Figure 12-3 硬件触发的波形

11.2.3 脉冲宽度计算

如下图，一个重复的 REM 输出波形，由低电平时长 t_{LOW} 和高电平时长 t_{HIGH} 组成。

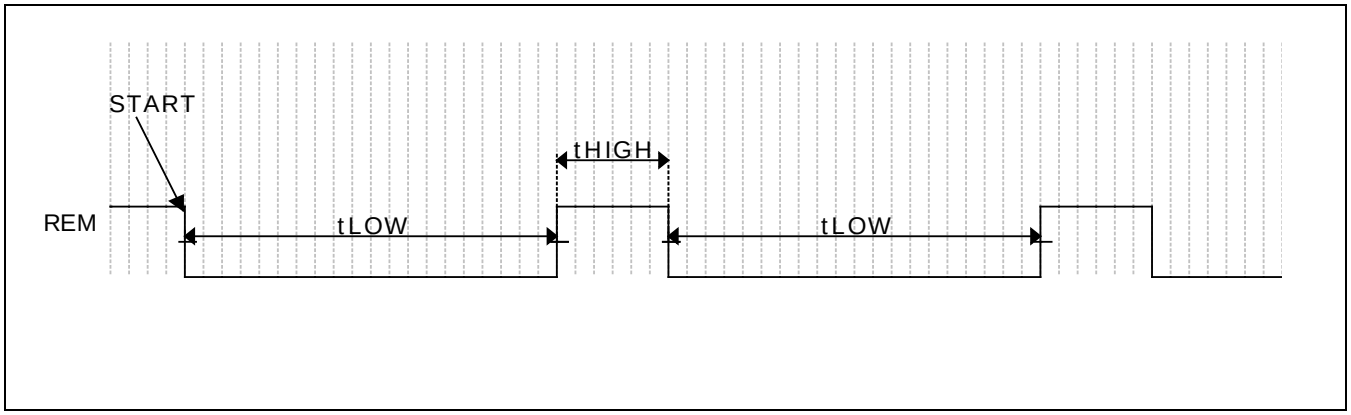


Figure 12-4 重复模式波形

当 CA_OSP = 0,

t_{LOW} = (CADATAL + 3) x 1/CA_CLK. 0H < CADATAL < 10000H.
t_{HIGH} = (CADATAH + 3) x 1/CA_CLK. 0H < CADATAH < 10000H.

当 CA_OSP = 1,

t_{LOW} = (CADATAH + 3) x 1/CA_CLK. 0H < CADATAH < 10000H.
t_{HIGH} = (CADATAL + 3) x 1/CA_CLK. 0H < CADATAL < 10000H.

为了让 t_{LOW} = 24us, 并且 t_{HIGH} = 15us, PCLK = 4MHz, CA_CLK = 4MHz/4 = 1MHz

[方法 1] 当 CA_OSP = 0,

t_{LOW} = 24 us = (CADATAL + 3) / CA_CLK = (CADATAL + 2) x 1us, CADATAL = 21.
t_{HIGH} = 15 us = (CADATAH + 3) / CA_CLK = (CADATAH + 2) x 1us, CADATAH = 12.

[方法 2] 当 CA_OSP = 1,

t_{HIGH} = 15 us = (CADATAL + 3) / CA_CLK = (CADATAL + 2) x 1us, CADATAL = 12.
t_{LOW} = 24 us = (CADATAH + 3) / CA_CLK = (CADATAH + 2) x 1us, CADATAH = 21.

11.2.4 中断

计数器 A 产生 2 个中断。

- 低电平结束中断
- 高电平结束中断

当输出的低电平结束时，产生 PLENDI 中断；当输出的高电平结束时，产生 PHENDI 中断。两个中断都只有一个周期的宽度，在重复模式下会被自动清除。

11.2.5 编程提示

下面的例子演示如何产生一个 38KHz, 1/3 占空比的信号。

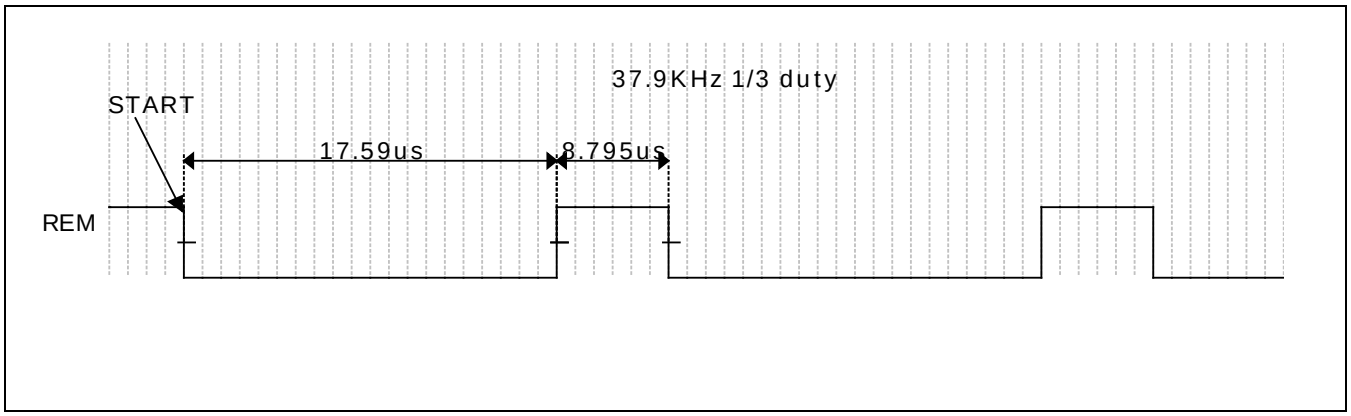


Figure 12-5 38KHz, 1/3占空比的REM输出波形

- 设置计数器A为重复模式
- 设置计数器A的时钟为4MHz (0.25us).
- 高电平持续: 8.795us/0.25us = 35.18, CADATAH = 32 (检测0计数值和转换计数模式需要3个周期)
- 低电平持续: 17.59us/0.25us = 70.36, CADATAL = 67

11.3 寄存器说明

11.3.1 寄存器表

- Base Address: 0x4005_0000

Register	Offset	Description	Reset Value
CADATAH	0x00	计数器 A DATAH 寄存器	0x0000_0000
CADATAL	0x04	计数器 A DATAL 寄存器	0x0000_0000
CACON	0x08	计数器 A 控制寄存器	0x0000_0000
CAINTMASK	0x0C	计数器 A 中断控制寄存器	0x0000_0000

注意：

- 计数器 A 的中断会在一个 PCLK 周期后被自动清除掉，没有中断状态寄存器，所以如果两个中断都使能的情况下，无法查询是哪一个中断。

11.3.2 CADATAH (DATAH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																CADATAH															
0	0	0	0	0	0	0	0	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
CADATAH	[15:0]	RW	DATAH控制计数器A输出的高电平宽度

11.3.3 CADATAL (DATAL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																CADATAHL															
0	0	0	0	0	0	0	0	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
CADATAL	[15:0]	RW	DATAL控制计数器A输出的低电平宽度

11.3.4 CACON (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						CARRIERON	ENVELOPE	REM_STAT	BTPEND_REM_CON		BTMATCH_REM_CON		HW_STROBE_DATA		SW_STROBE_DATA	RSVD										CA_CLK		STOP	START	MODE	OSP
0	0	0	0	0	0	0	0	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	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CARRIERON	[25]	RW	载波信号控制位 0: 关闭载波 1: 打开载波
ENVELOPE	[24]	RW	REM输出信号选择位 0: 选择载波信号为输出 1: 选择包络信号为输出
REM_STAT	[23]	RW	当关闭载波时REM的输出状态 0: 低 1: 高
BTPEND_REM_CON	[22:21]	RW	当BT周期结束中断发生时，硬件触发控制载波的打开和关闭 00/11: 禁止CARRIERON的硬件自动触发 01: BT周期结束中断发生时，CARRIERON位会被硬件自动清零 10: BT周期结束中断发生时，CARRIERON位会被硬件自动置位
BTMATCH_REM_CON	[20:19]	RW	当BT脉冲匹配中断发生时，硬件触发控制载波的打开和关闭 00/11: 禁止CARRIERON的硬件自动触发 01: BT脉冲匹配中断发生时，CARRIERON位会被硬件自动清零 10: BT脉冲匹配中断发生时，CARRIERON位会被硬件自动置位
HW_STROBE_DATA	[18:17]	RW	使能计数值寄存器的硬件自动更新功能

			X1: 当BT脉冲匹配中断发生时，计数值会自动载入计数器 1X: 当BT周期结束中断发生时，计数值会自动载入计数器
SW_STROBE_DATA	[16]	RW	使能计数值寄存器软件更新 当该位置位时，CADATAH和CADATAL的值会更新到计数器中
CA_CLK	[5:4]	RW	输入时钟频率选择位 00: PCLK 01: PCLK/2 10: PCLK/4 11: PCLK/8
STOP	[3]	RW	停止位 计数器A在工作时，该位为0。需要停止计数器A，则需将该位写1。
START	[2]	RW	启动位。 写1会启动计数器A。 计数器A开始计数后，该位会被自动清零。
MODE	[1]	RW	模式选择位 0: 单次(One Shot)模式 1: 重复模式
OSP	[0]	RW	载波输出波形的初始极性选择 0: 低 1: 高

11.3.5 CAINTMASK (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																														PLEND_INT	PHENT_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	R	R	R	RW	RW

Name	Bit	Type	Description
PLEND_INT	[1]	RW	低电平结束中断的使能/禁止
PHEND_INT	[0]	RW	高电平结束中断的使能/禁止

12 增强型通用定时器 (GPTA)

12.1 概述

通用定时器（General Purpose Timer）作为 MCU 的关键外设，可以提供多种时基计数和波形产生功能。通过灵活的 PWM 输出，可以适用于各种复杂多变的应用。GPTA 内部包含一个 16 位的定时/计数模块，支持 2 种工作模式(捕捉模式和波形发生器模式)。

注：如果系列内芯片不具有本外围，那它就不具备本章描述的相关资源。具体参考芯片的数据手册。

12.1.1 主要特性

- 16 位可复位计数器
- 可编程计数器计数方式
  - 递增计数（Up-counting）
  - 递减计数（Down-counting）
  - 递增递减计数（Up-down-counting）
- 两路波形产生控制单元，支持双路独立输出：
  - 两路独立的 PWM 输出，单边沿工作
  - 两路独立的 PWM 输出，双边沿对称工作
- 通过软件异步重置 PWM 的波形输出
- 支持片间多设备同步
  - 支持多个 TIMER 间的同步触发
  - 触发源包括 GPIO 输入，其他外设触发，软件设置和事件触发
  - 支持单次触发和连续触发模式
- 支持单脉冲输出模式
- 支持突发计数模式
- 支持通过外部时钟计数
- 支持事件计数器，可通过配置事件计数器（最大 15）触发相应中断
- 支持捕获模式，最多支持 2 个捕获值存储。

12.1.2 管脚描述

下表列出了不同模式下的管脚定义。

Table 13-1 不同模式下的管脚描述

管脚名称	突发计数模式	波形发生器： 单波形输出模式	波形发生器： 双波形输出模式
GPTA_CHA	时钟控制使能	输出波形	输出波形
GPTA_CHB	时钟控制使能	输出波形	输出波形

12.2 功能描述

12.2.1 模块框图

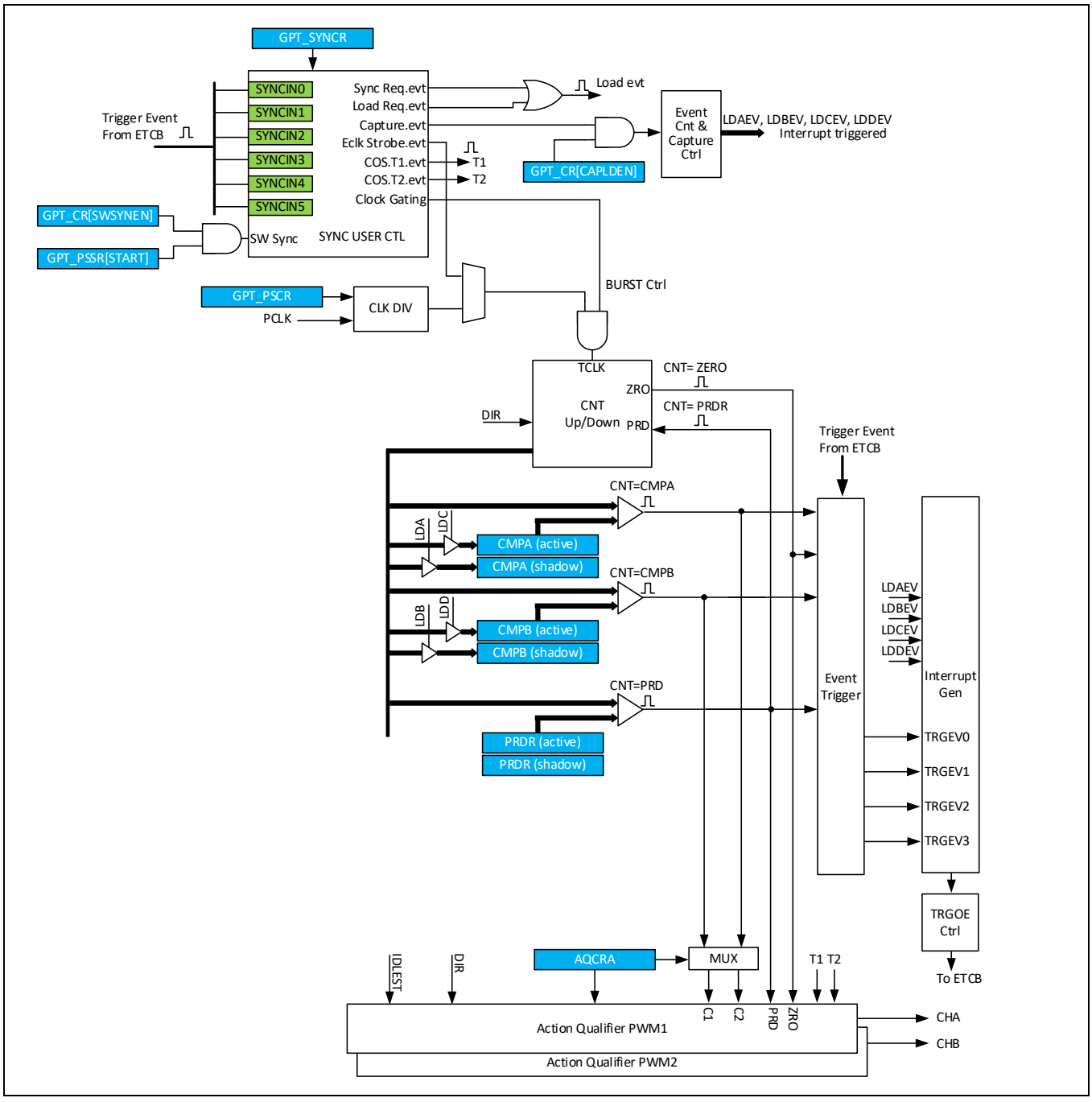


Figure 13-1 模块结构示意图

12.3 基本功能描述

一个完整的GPTA模块由两个TIMER输入/输出通道组成。多个GPTA或多个EPT可以通过SYNC链连接，通过SYNC链，实现多个GPTA和EPT的同步工作。在包含多个GPTA的器件中，以数字后缀区分不同的实例，例如：GPTA0代表第一个GPTA模块，GPTA1代表第二个GPTA模块。每个GPTA中根据功能划分，可以分为几个模块，包括时钟控制模块、时基模块（计数器）、计数器比较模块、动作限定模块、死区控制模块、斩波模块、捕捉控制模块、事件触发模块、紧急处理模块和同步触发控制模块。

GPTA_CHA和GPTA_CHB是GPTA在GPIO上映射的双向输入输出端口。在波形输出模式下，任何一个EXI都可以通过ETCB模块配置为PWM信号的输出端口，在群脉冲模式下（GPTA_CR[BURST]使能），GPTA_CHA和GPTA_CHB可以作为门控时钟的时钟控制输入信号。EPIx为外部GPIO上映射的输入功能，可以作为紧急状态处理的触发信号。

12.3.1 时钟源

12.3.1.1 概述

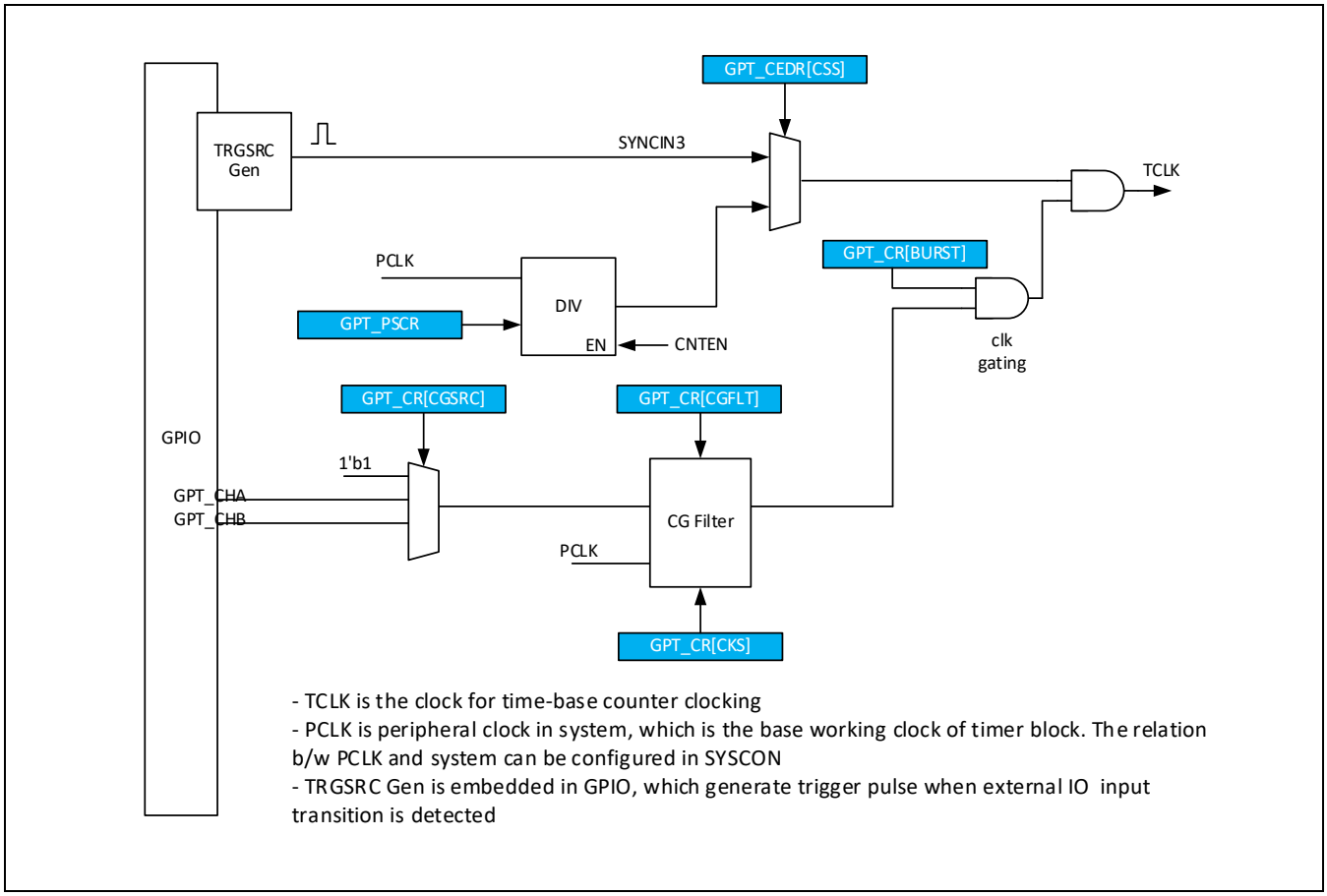


Figure 13-2 时钟控制模块

增强型通用定时器GPTA工作在PCLK下。计数器计数时钟TCLK可以通过选择PCLK的分频后输出，或者由外部提供，计数器在外部计数器时钟控制下计数时，外部计数时钟在GPIO模块内根据设置，产生相应的触发脉冲，该脉冲作为GPTA的时基计数器外部触发源，触发计数器递增或递减，外部计数时钟频率必须低于1/2倍PCLK，以保证被PCLK同步，例如：当PCLK频率为4MHz时，则最高外部输入时钟为2MHz。在外部时钟超过PCLK频率限制时，可以通过异步预分频对输入时钟进行预先分频，保证分频后的时钟频率满足被PCLK同步的条件。

12.3.1.2 外部时钟

外部时钟的选择和极性控制，通过GPIO内的TRGSRC GEN进行控制。在外部时钟模式下，外部时钟通路上集成有异步时钟分频器和数字滤波器。异步分频器可以在外部时钟超过2倍PCLK时，用以降低输入频率。数字滤波器可以在外部时钟有较大干扰时打开。

12.3.1.3 内部时钟

当PCLK作为计数器的计数时钟时，可以通过一个16位的预分频器对PCLK进行分频而产生计数用的TCLK。预分频可以通过GPTA_PSCR进行设置。在对GPTA_PSCR进行读写时，操作的对象为PSCR的影子寄存器（Shadow Register）；当时基计数器的值等于零时，影子寄存器的值将被更新到内部的活动寄存器中（Active Register）。当对GPTA_PSCR更新后，新的分频将在下一个计数周期开始时有效。

12.3.1.4 群脉冲时钟

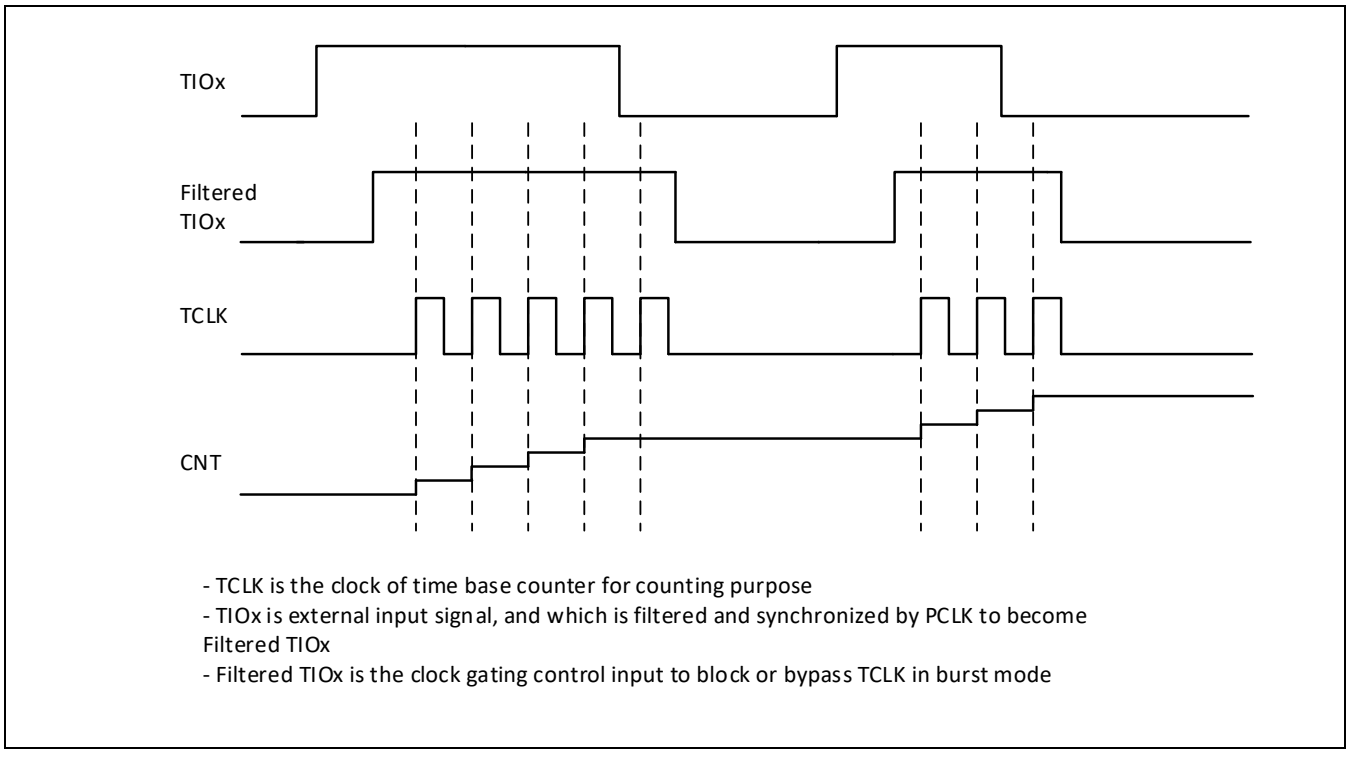


Figure 13-3 群脉冲时钟模式时序

在群脉冲时钟模式下GPTA_CR[BURST]，计数器的计数时钟将会和相应的控制信号进行与操作。计数器只有在被选择的使能信号有效时，才进行计数。门控时钟使能信号可以通过GPTA_CR[CGSRC]控制位进行选择，支持GPTA_CHA或者CHB的外部输入作为门控信号。当GPTA_CHA或GPTA_CHB选择为门控时钟时，该通道自动设置为输入状态，并禁止该通道的波形输出。

在CG输入通道上，可以通过设置GPTA_CR[CGFLT]使能数字滤波。数字滤波在连续检测到N个一致结果时，才会确认输出翻转，否则将保证当前输出状态。如下图所示。数字滤波器的滤波时钟可以通过GPTA_CR[CKS]控制位进行设置。

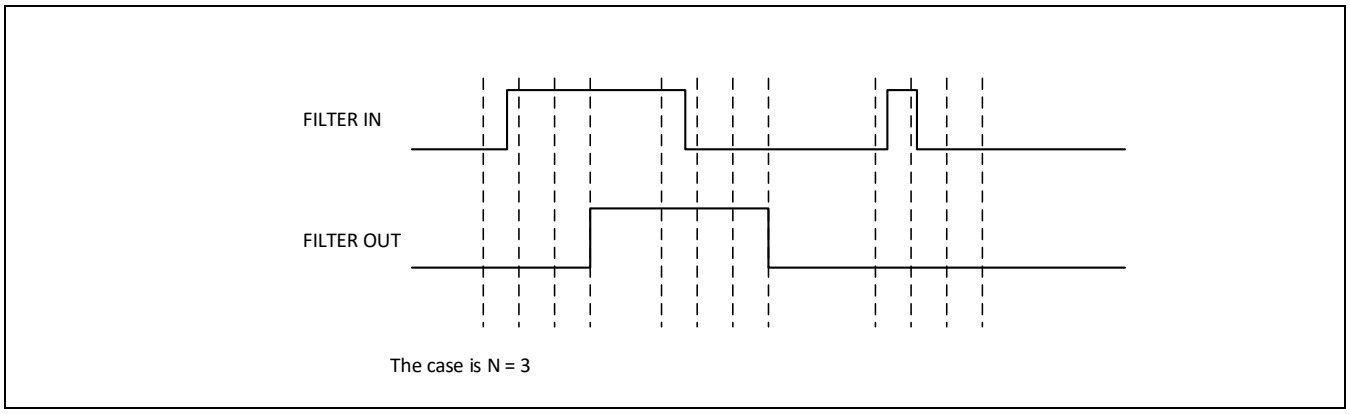


Figure 13-4 CG Filter 数字滤波器的原理

12.3.2 时基控制

12.3.2.1 概述

作为GPTA主要的控制模块，时基控制模块由一个16位的计数器和相应的自动重载寄存器组成。模块的主要功能有：

- 确定时基计数器（GPTA_CNT）的频率，或者控制事件触发的周期。
- 管理和其他模块间的同步
- 控制和其他GPTA模块间的相位关系
- 设置计数器工作模式
- 根据计数器值产生不同的触发事件

时基模块的寄存器包括：

- 计数器寄存器（GPTA_CNT）：在每个计数时钟周期根据计数模式增加或者减少
- 周期寄存器（GPTA_PRDR）：计数器周期控制寄存器。

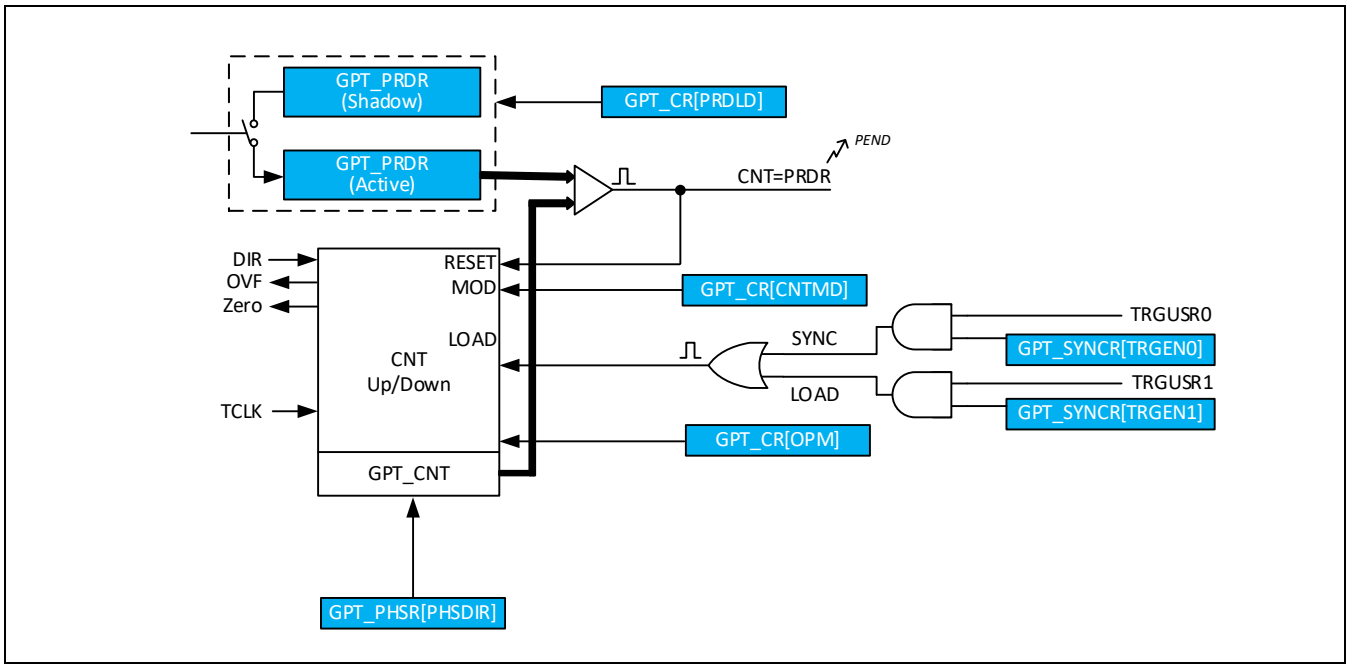


Figure 13-5 计数器时基模块

计数器的计数周期由周期寄存器（GPTA_PRDR）的设置值以及计数器的计数模式（GPTA_CR[CNTMD]）共同决定。计数器支持三种计数模式：

- 递增模式（Up-Counting Mode）：

  在递增模式下，时基计数器从0x0000开始递增计数，一直计数到周期设置值（GPTA_PRDR）。当计数值等于周期设置值时，时基计数器被复位，重新开始从0x0000进行新一轮计数。

- 递减模式：（Down-Counting Mode）

  在递减模式下，时基计数器从周期设置值开始递减计数，一直计数到0x0000。当计数器值等于0x0000时，时基计数器被重置为周期设置值并开始新一轮计数。

- 递增递减模式：（Up-Down-Counting Mode）

  在递增递减模式下，时基计数器从0x0000开始递增，递增到周期设置值开始递减计数，一直计数到0x0000。当计数器值等于0x0000时，重新开始新一轮计数。

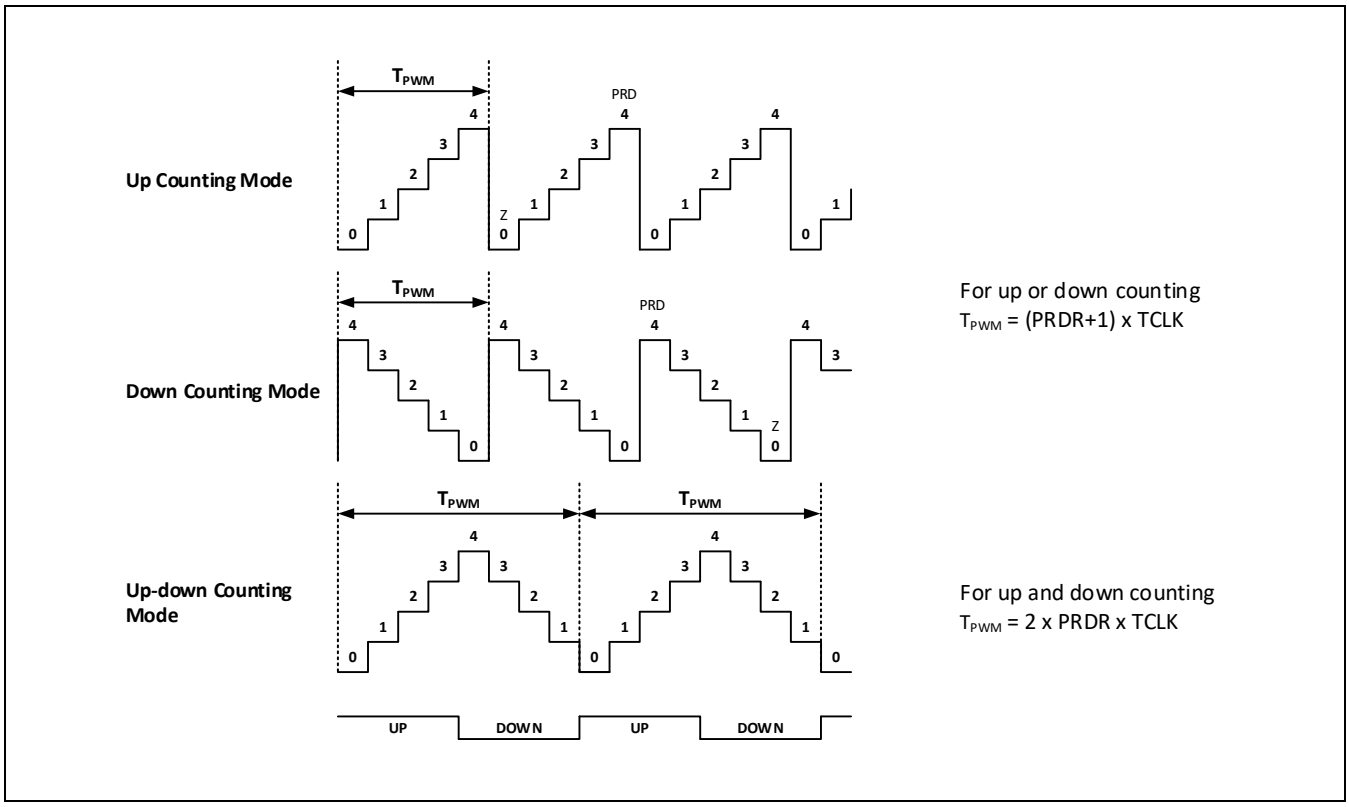


Figure 13-6 计数器工作模式

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(GPTA_RISR[IOVF]置高，同时可以产生一个中断(通过中断使能控制寄存器GPTA_IER[IOVF]设置)，然后继续开始计数。

12.3.2.2 计数器重置和周期设置

在下列条件满足时，计数器值将会被重置。重置发生时，根据当前计数模式，计数器将被重置为0x0000，或者是PRDR的设置值。

- 同步事件触发：当同步事件发生时，可以配置计数器重置。
- 计数值等于0x0000：当周期开始计数且当前计数值等于零，计数器的值将被重置到PRDR所设置的数值。
- 软件直接更新：通过软件直接写入计数器活动寄存器进行更新。

GPTA_PRDR周期寄存器由两个物理寄存器组成：活动寄存器（Active）和影子寄存器（Shadow）。影子寄存器的值通过硬件同步到活动寄存器中，保证对内部活动寄存器更新操作和计数周期同步。活动寄存器直接参与计数器控制事件的产生；影子寄存器作为数据缓冲，为活动寄存器提供临时的数据保存。影子寄存器值不会直接影响硬件控制动作，而是根据预设策略，在特定时间将缓冲的内容传送到活动寄存器中。这样的机制，避免了由于软件非同步地对寄存器操作而引起的硬件输出错误。活动寄存器和影子寄存器共享同一个物理访问地址，当前读写操作的对象是活动寄存器还是影子寄存器，可以通过GPTA_CR[PRDLD]控制位进行选择。当影子寄存器被屏蔽时，对PRDR的写入值，会直接改变活动寄存器的值，而对PRDR读取时，将直接返回活动寄存器的值。

- **PRDR寄存器的Shadow模式**

PRDR的缓冲（Shadow Register）在GPTA_CR[PRDLD]控制位不等于’00’的时候有效。在此配置下，CPU对PRDR的读写操作对象为PRDR的影子寄存器。当时基计数器值等于零时，或者SYNC触发时，影子寄存器的值被硬件自动载入到活动寄存器（Active Register）中。在缺省配置下，只有时基计数器值等于零时，自动载入才会发生，用户可以通过配置GPTA_CR[PRDLD]控制位进行修改。

- **PRDR寄存器的立即加载模式**

在立即加载模式下（GPTA_CR[PRDLD]=3），CPU对PRDR的读写取操作对象是PRDR的活动寄存器。任何对PRDR的更新操作将被直接反应到活动寄存器中。

在对PRDR进行立即更新时，需要注意考虑当前的计数器值。若将PRDR更新到一个比当前计数器值小的值，将导致计数器在后续计数过程中都不会发生PERIOD事件，计数器将一直计数到整个计数器溢出后重新开始计数。

计数器在停止计数后，不会自动清除计数值，而是保留当前的计数值，需要通过软件进行清零。

12.3.2.3 计数模式和时序

时基计数器可以分为四种工作模式：

- 递增计数模式（非对称）
- 递减计数模式（非对称）
- 递增递减模式（对称）
- 冻结模式，在此模式下计数器保持当前计数值

在下面的图示中说明了上述前三种工作模式下，时基计数器根据触发条件如何工作和产生相应事件。

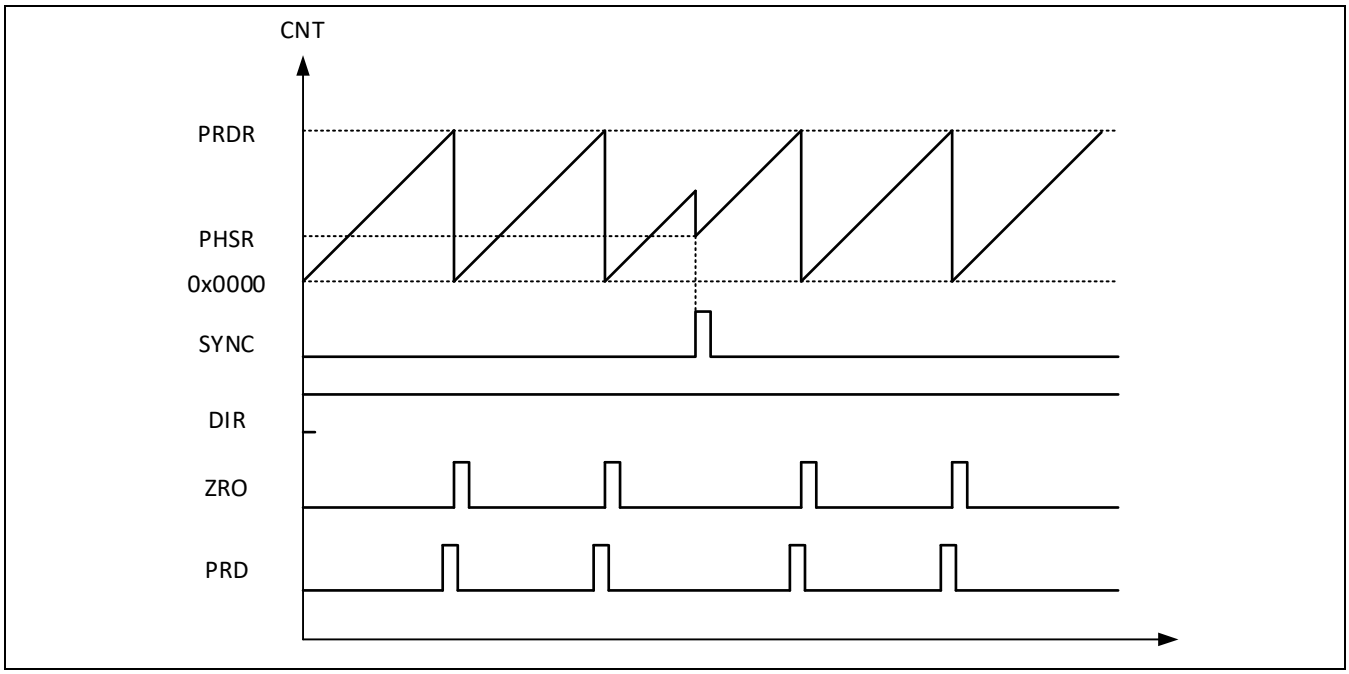


Figure 13-7 递增工作模式

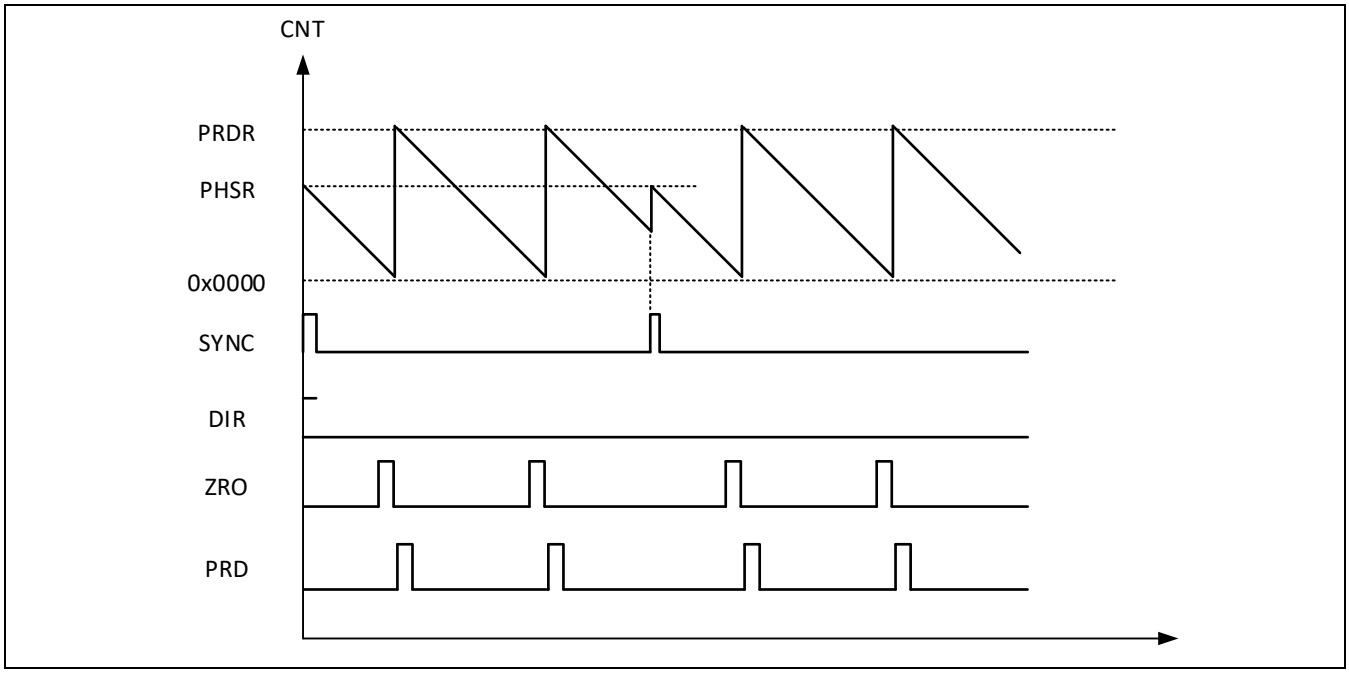


Figure 13-8 递减工作模式

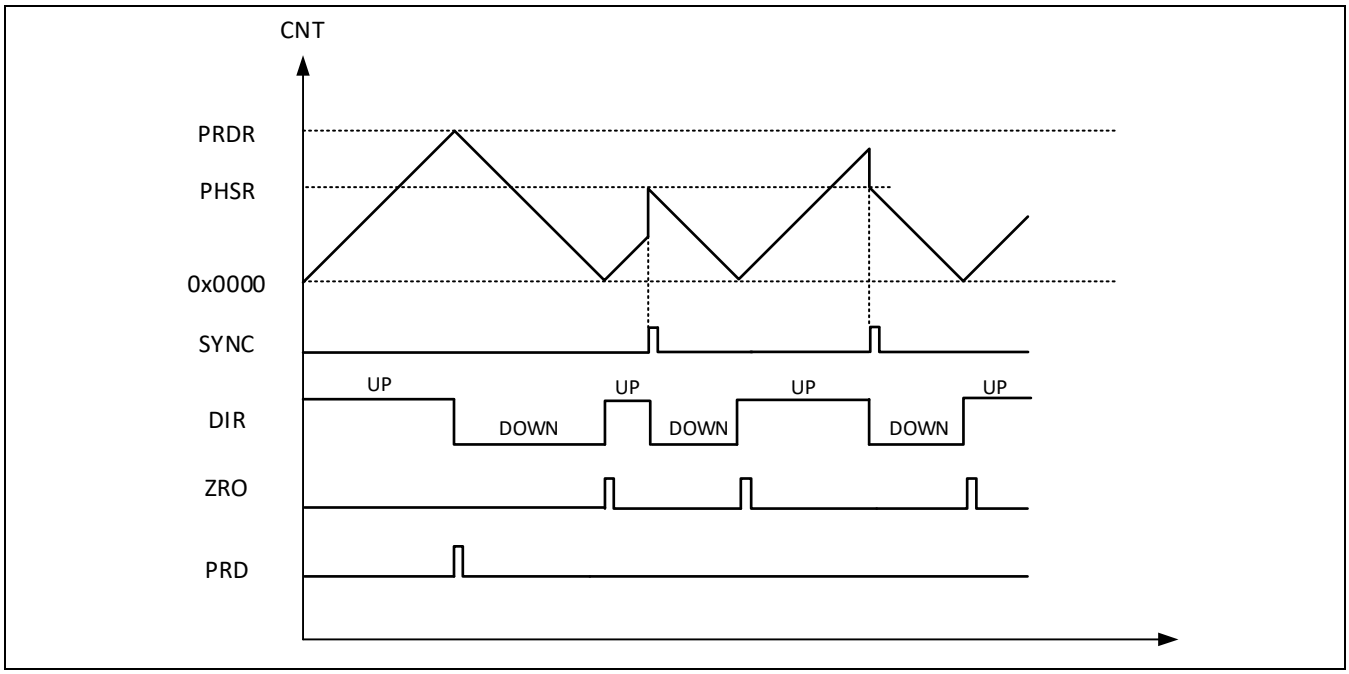


Figure 13-9 递增递减工作模式

12.3.2.4 全局载入控制

GPTA中，很多寄存器具有影子寄存器功能。每个影子寄存器只有在特定条件满足时，才会更新到活动寄存器中。每个影子寄存器对活动寄存器的更新条件均可以独立设置。全局载入模式可以重载这些配置，当全局载入使能时，所有影子寄存器对活动寄存器的更新都将受全局载入条件控制，并在全局载入条件满足时，将全部影子寄存器的当前值更新到对应的活动寄存器中。全局载入使能通过GLDCR[GLDEN]设置，当全局载入使能时，可以通过GLDCFG寄存器配置每个影子寄存器是否受全局载入控制，用户可以通过配置GLDCFG，选择不需要受全局载入控制的影子寄存器。设置为不受全局载入控制的寄存器，将使用自己独立的载入控制配置。例如：当GLDEN=1, 且GLDCFG[C1]=1，GLDCFG[C2]=0时，则CMPA的影子寄存器将在全局载入条件满足时，更新到活动寄存器中；CMPB的影子寄存器的更新条件不受全局载入条件控制，仍旧按照CMPLDR[LDBMD]的设置进行更新。

全局载入控制支持事件计数，只有当选中的触发事件在第N次发生时，才会进行全局载入，N为事件计数器的设置值。可以通过寄存器GLDCR[GLDPRD]控制位设置事件计数器值，当前已经发生的触发次数可以通过GLDCR[GLDCNT]控制位进行查询。

全局载入可以被连续触发，或者只允许发生一次。当One-shot 载入模式使能时（GLDCR[OSMD]=1），只在全局条件满足时，触发一次全局载入，后续的载入将被屏蔽。必须通过软件重新初始化后，才能进行新的触发。通过设置GLDCR2[GFRCLD[控制位，软件可以强制触发全局载入。

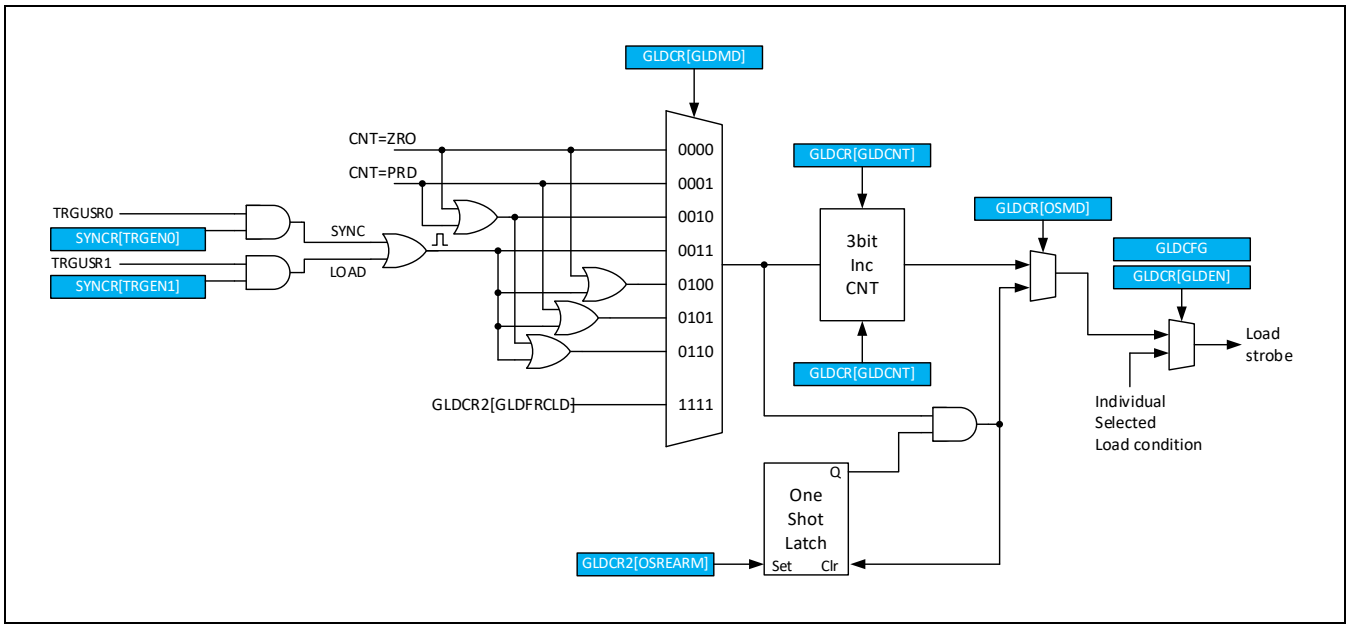


Figure 13-10 全局载入控制

12.3.3 计数器数值比较控制

12.3.3.1 概述

计数器数值比较控制比较控制模块实时比较当前计数器的计数值和比较值寄存器（CMPA、CM）的值，当计数值等于其中任意一个比较值时，比较控制模块将产生一个相应的事件触发。主要特性如下：

- 支持的触发事件和触发条件如下：
  - CNT = CMPA：时基计数器当前值等于计数器比较值A寄存器的值
  - CNT = CMPB：时基计数器当前值等于计数器比较值B寄存器的值
- PWM1和PWM2的波形控制是基于CMPA和CMPB的
- 比较值寄存器具有影子寄存器功能，以防止PWM输出产生毛刺

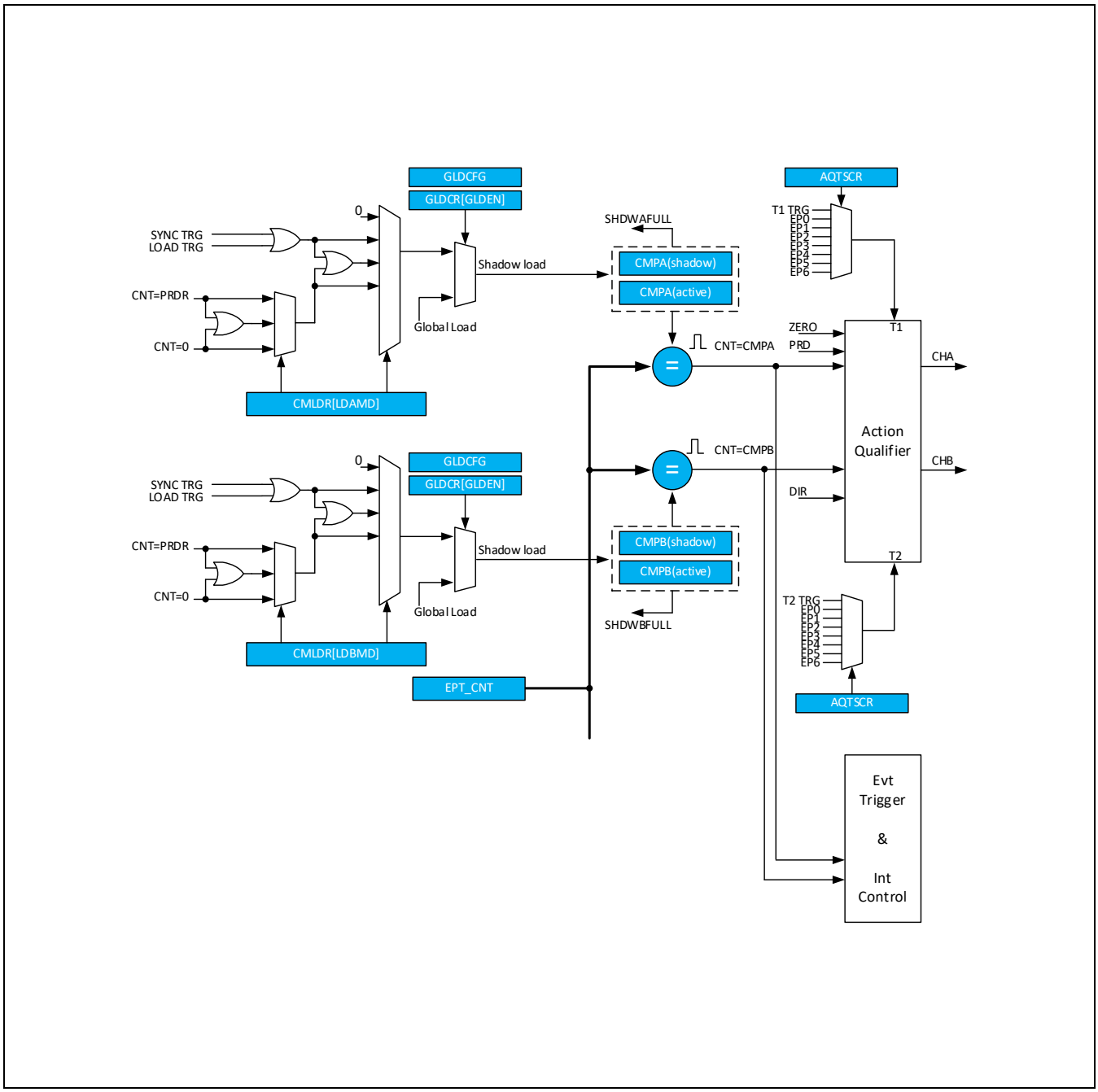


Figure 13-11 计数器值比较控制

计数值比较模块不断监测当前时基计数器的计数值，当计数值等于两个比较值中的任意一个时，都会触发独立的比较事件。2个比较事件都可以用于触发中断或者同步，但只有CMPA和CMPB比较事件可以用于波形发生控制。

在递增模式或者递减模式下，每个比较事件在一个计数周期内只会发生一次。在递增递减模式下，如果比较值设置为0到PRD之间时，每个事件在一个计数周期内会发生两次；而如果比较值设置为0或者PRD时，每个事件在一个计数周期内只发生一次。CMPA和CMPB这两个触发事件以及和来自于时基模块的当前计数方向信号，在波形发生模块中共同决定了输出波形的跳转时间点。

12.3.3.2 比较值寄存器载入方式

C1、CMPB都有相应的Shadow寄存器，在缺省设置下，所有对CMPx寄存器的读写对象都是影子寄存器。Shadow load的时间可以通过相应CMPLDR[LDxMD]控制位进行设置。影子寄存器的使能可以通过CMPLDR[SHDWCMPx]控制位进行设置。当Shadow模式被禁止时，所有对CMPx寄存器的操作将直接作用到内部活动寄存器上。

- **CMPx寄存器的Shadow模式**

  当Shadow模式使能时，Shadow寄存器中的内容将在下列事件触发时，被自动传送到活动寄存中。可以通过CMPLDR[LDxMD]控制位选择触发CMPx活动寄存器进行更新的事件。下列任意一种组合都可以被作为更新寄存器的触发条件。

  - CNT = ZRO时，触发更新
  - CNT = PRD时，触发更新
  - CNT = PRD或者CNT = ZRO时，触发更新
  - 外部事件（外部LOAD触发或SYNC触发）触发更新
  - 外部事件（外部LOAD触发或SYNC触发）或上述任意CNT MATCH事件触发更新

- **CMPx寄存器的立即加载模式**

  在立即加载模式下，对CMPx的操作直接影响活动寄存器。

当全局载入使能，且相应的CMPx在全局载入控制中被选择，全局载入模式的设置将覆盖CMPLDR中的载入方式设置。全局载入的设置具体参照**全局载入控制**章节。

12.3.3.3 不同计数模式的时序

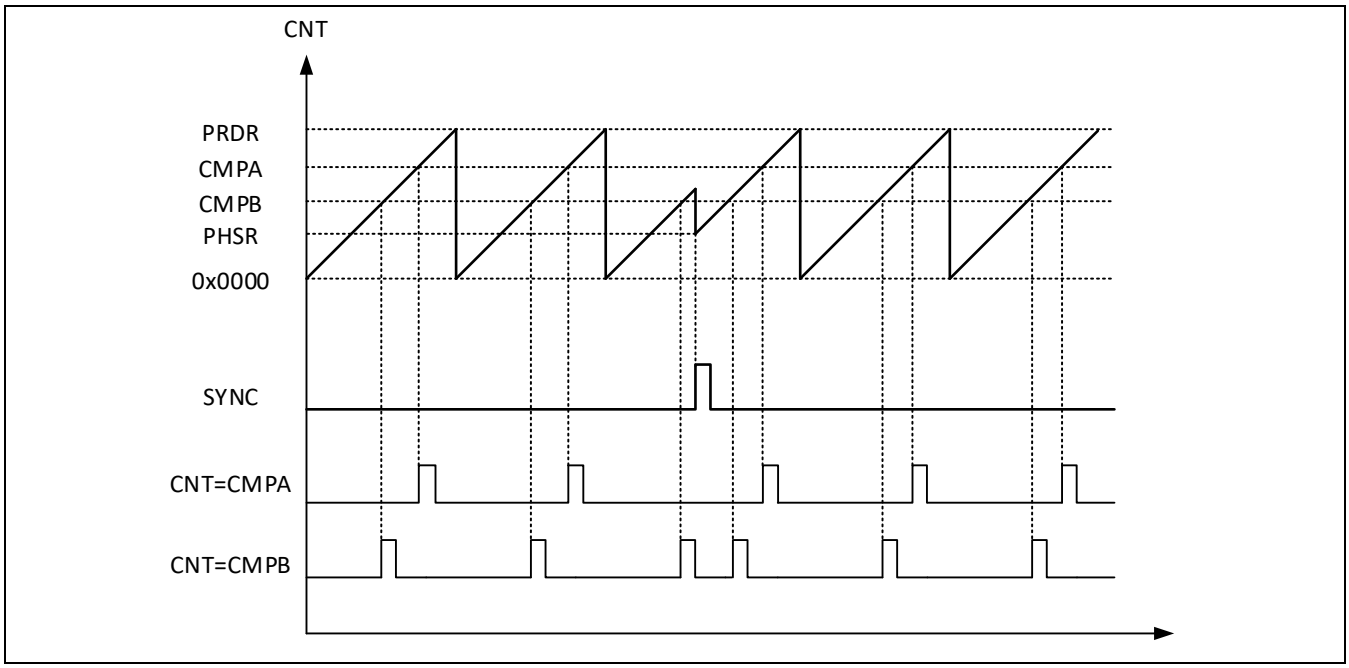


Figure 13-12 递增模式下比较事件产生时序

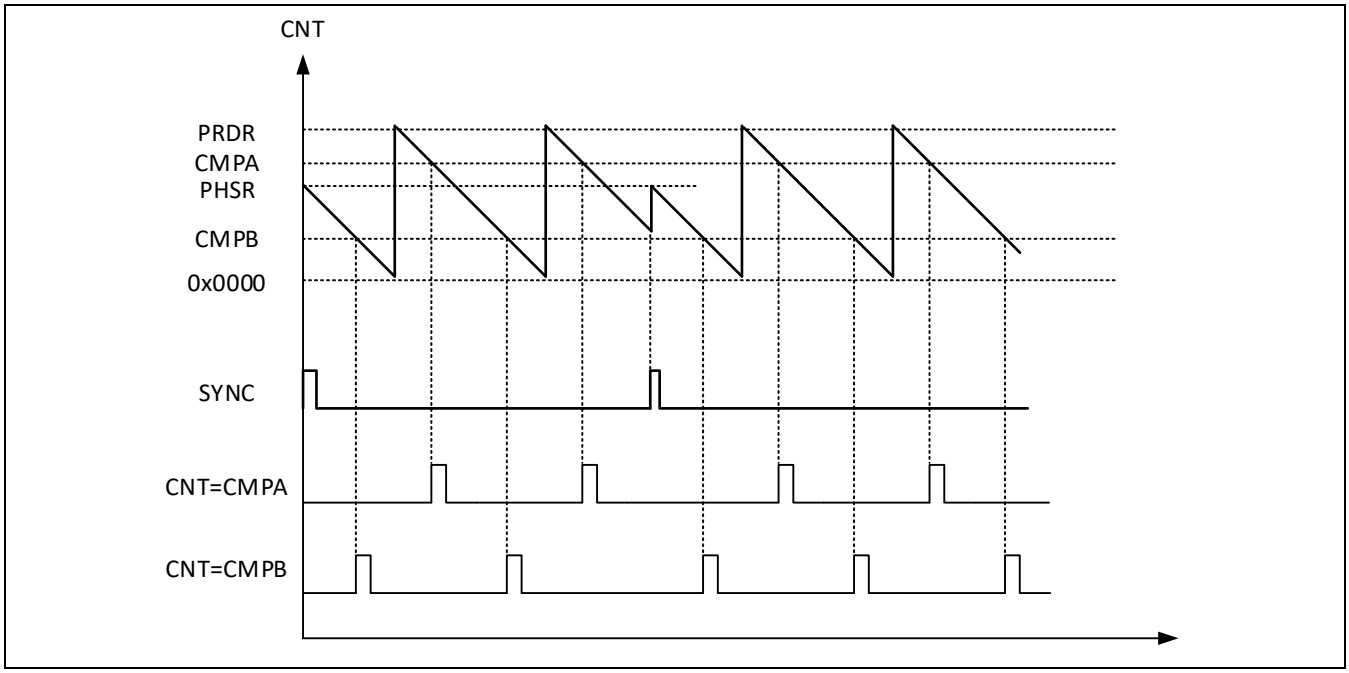


Figure 13-13 递减模式下比较事件产生时序

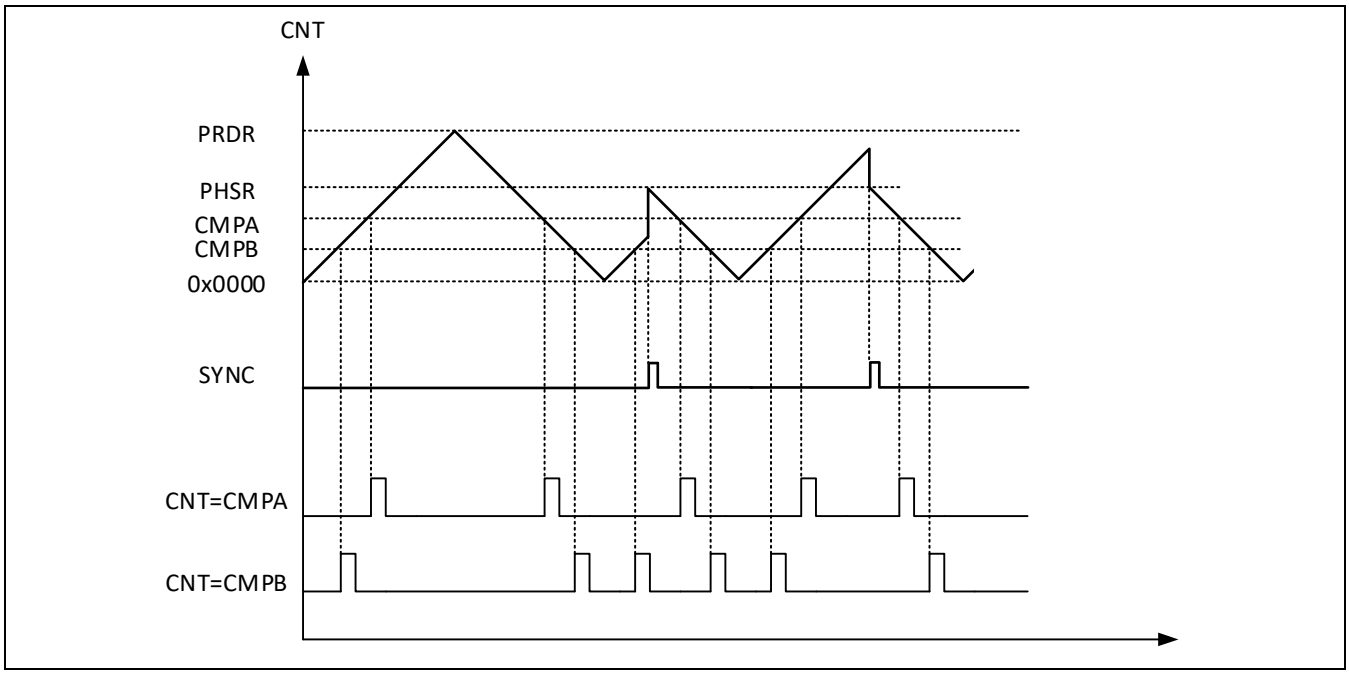


Figure 13-14 递增递减模式下比较事件产生时序

12.3.4 波形发生控制

12.3.4.1 事件驱动的波形输出

GPTA中有两路独立的波形输出通路（PWM1和PWM2），PWM的波形产生基于不同事件的驱动，通过控制寄存器AQCRA和AQCRB的设置，可以独立映射各种事件触发到PWM1或者PWM2的输出状态。AQCR1对应控制PWM1通道上的波形输出，AQCR2对应PWM2通道上的波形输出。AQCR1和AQCR2都具有影子寄存器功能，可以通过AQCR寄存器对影子寄存器载入到活动寄存器的触发条件进行配置，原理和CMPx的影子寄存器相同，可以参考比较值寄存器载入方式章节。PWM波形控制所支持的触发事件包括：

- CNT = PRD （计数器值等于周期设置值）
- CNT = ZERO （计数器值等于零）
- CNT = C1 （计数器值等于C1设置值，C1参考值由AQCRx[C1SEL]设置）
- CNT = C2 （计数器值等于C2设置值，C2参考值由AQCRx[C2SEL]设置 ）
- T1事件 （来自SYNCIN4）
- T2事件 （来自SYNCIN5）
- 软件Force事件 （通过软件触发的异步强制置位）

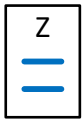
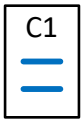
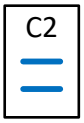
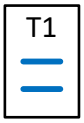
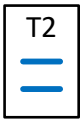
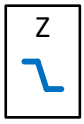
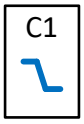
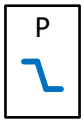
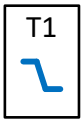
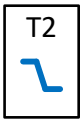
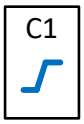
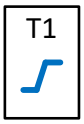
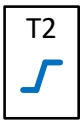
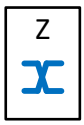
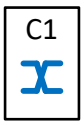
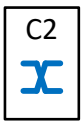
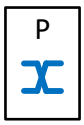
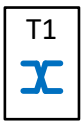
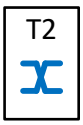
波形发生模块根据计数器的当前计数方向和发生的事件，决定PWM1和PWM2通道上的动作。所支持的的输出动作包括：

- 设置高电平 （在PWM1或者PWM2通道上设置高电平输出）
- 设置低电平 （在PWM1或者PWM2通道上设置低电平输出）

- 翻转 （在PWM1或者PWM2通道上对输出进行翻转）
- 保持 （保持当前PWM1或者PWM2通道上的电平）

波形发生器可以独立定义PWM1或者PWM2通道上的输出动作。任何触发事件中的一个或者全部都可以用于产生输出动作。在对不需要触发任何波形输出变化的事件配置中，可以将该事件的触发动作设置为保持（相当于禁止该事件的处理）。比如CNT=CMPA和CNT=CMPB同时可以作为PWM1的输出控制。在下面的例图中，给出了在波形描述图示中会引用到的图标。

Table 13-2 各种在PWM1和PWM2上可能触发的动作

软件 Force	CNT 值等于				事件触发		动作
	Zero	C1SEL	C2SEL	PRD	T1	T2	
							没有动作
							低电平输出
							高电平输出
							翻转输出

12.3.4.2 触发事件的优先级

在同一个时间可能有多个事件同时触发，在这种情况下，具有高优先级的事件将决定输出的状态。通常，后发生的事件具有更高的优先级，而软件强制输出具有最高的优先级。优先级通过硬件决定，不能由寄存器进行更改。在下表中，列出各种计数模式下的优先级设置，优先级数字越小代表优先级更高。

Table 13-3 递增递减模式（Up-Down-Count）下的事件优先级

priority	Trigger event (Up-Counting phase)	Trigger event (decreasing phase)
1(Highest)	Software Forced event	Software Forced event
2	T1 on up-count(T1U)	T1 on down-count(T1D)
3	T2 on up-count(T2U)	T2 on down-count(T2D)
4	CNT equals C2 on up-count(C2U)	CNT equals C2 on down-count(C2D)
5	CNT equals C1 on up-count(C1U)	CNT equals CMBA on down-count(C1D)
6	CNT equals zero	CNT equals period

7	T1 on down-count(T1D)	T1 on up-count(T1U)
8	T2 on down-count(T2D)	T2 on up-count(T2U)
9	CNT equals C2 on down-count(C2D)	CNT equals C2 on up-count(C2U)
10(Lowest)	CNT equals CMBA on down-count(C1D)	CNT equals C1 on up-count(C1U)

Table 13-4 递增模式下的事件优先级

priority	Trigger Event
1(Highest)	Software Forced event
2	CNT equals period
3	T1 on up-count(T1U)
4	T2 on up-count(T2U)
5	CNT equals C2 on up-count(C2U)
6	CNT equals C1 on up-count(C1U)
7(Lowest)	CNT equals zero

在递增模式下，由于计数器方向一直保持递增，所以和递减相关的事件将永远不会发生。

Table 13-5 递减模式下的事件优先级

Priority	Trigger Event
1(Highest)	Software Forced event
2	CNT equals zero
3	T1 on down-count(T1D)
4	T2 on down-count(T2D)
5	CNT equals C2 on down-count(C2D)
6	CNT equals C1 on down-count(C1D)
7(Lowest)	CNT equals period

在递减模式下，由于计数器方向一直保持递减，所以和递增相关的事件将永远不会发生。

用户可以随意设置CMPA和CMPB的值，当设置的CMP值大于Period的设置值时，将会按照下述方式进行操作。

- 计数器设置为递增模式时：
  1）如果CMPA/C2 ≤ PRD时，C1U/CBU事件在CNT等于CMPA/CMPB时被触发；
  2）如果CMPA/C2 > PRD时，C1U/CBU事件不会被触发；
  3）C1D/CBD事件始终不会被触发。

- 计数器设置为递减模式时：
  1）如果CMPA/C2 < PRD时，C1D/CBD事件在CNT等于CMPA/CMPB时被触发；

2）如果CMPA/C2 ≥ PRD时，C1U/CBU事件在CNT等于Period时触发；

3）C1U/CBU事件始终不会被触发。

- 计数器设置为递增递减模式时：

1）如果CMPA/C2 < PRD且计数器计数方向为递增时，C1U/CBU事件在CNT等于C1/C2时被触发；

2）如果CMPA/C2 < PRD且计数器计数方向为递减时，C1D/CBD事件在CNT等于C1C2时被触发；

3）如果CMPA/C2 ≥ PRD时，C1U/C2U/C1D/C2D事件在CNT等于Period时触发。

12.3.4.3 软件强制输出

PWM的波形输出支持通过软件进行控制。软件强制输出可以将输出通道通过软件强制设置为预设电平，此功能类似紧急模式下的波形输出，但是紧急模式下的波形输出具有更高的优先级，且具有异常标志和中断报警特性。

软件强制输出可以分为两种模式：一次性Force和连续Force。

一次性软件强制输出（One-Shot Software Forcing）

在此模式下，通过寄存器的设置可以将PWM1或者B的输出强制修改成软件设置电平，且该电平一直维持到有新的触发事件发生。可以通过设置寄存器GPTA_AQOSF[ACT1]和GPTA_AQOSF[ACT2]控制位，设置在一次性强制输出触发时，PWM1和PWM2上的输出电平状态。通过对GPTA_AQOSF[OSTSF1]和GPTA_AQOSF[OSTSF2]控制位写入1，触发一次性强制输出。

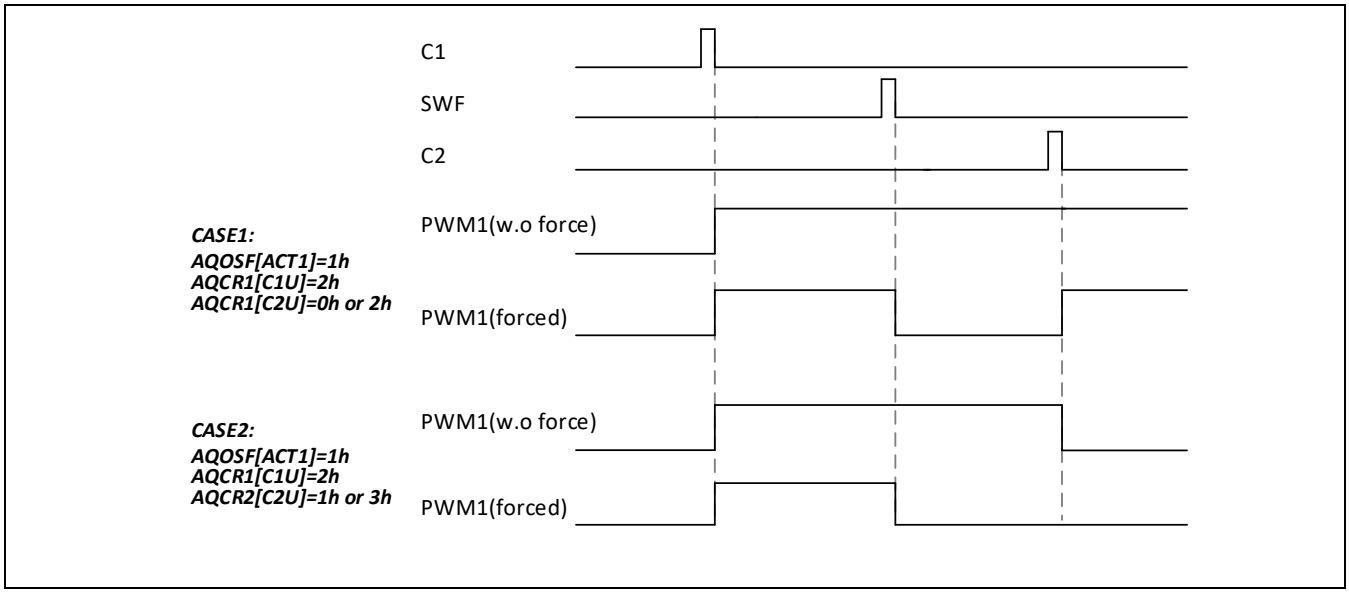


Figure 13-15 一次性软件强制输出

连续软件强制输出（Continuous Software Forcing）

在此模式下，通过寄存器的设置可以将PWM1或者2的输出强制修改成软件设置电平，且该电平一直维持到软件清除强制输出。当软件清除强制输出后，通道电平将恢复到强制输出前的状态。可以通过设置寄存器GPTA_AQCSF[CSF1]控制位，进行强制输出设置或者清除。

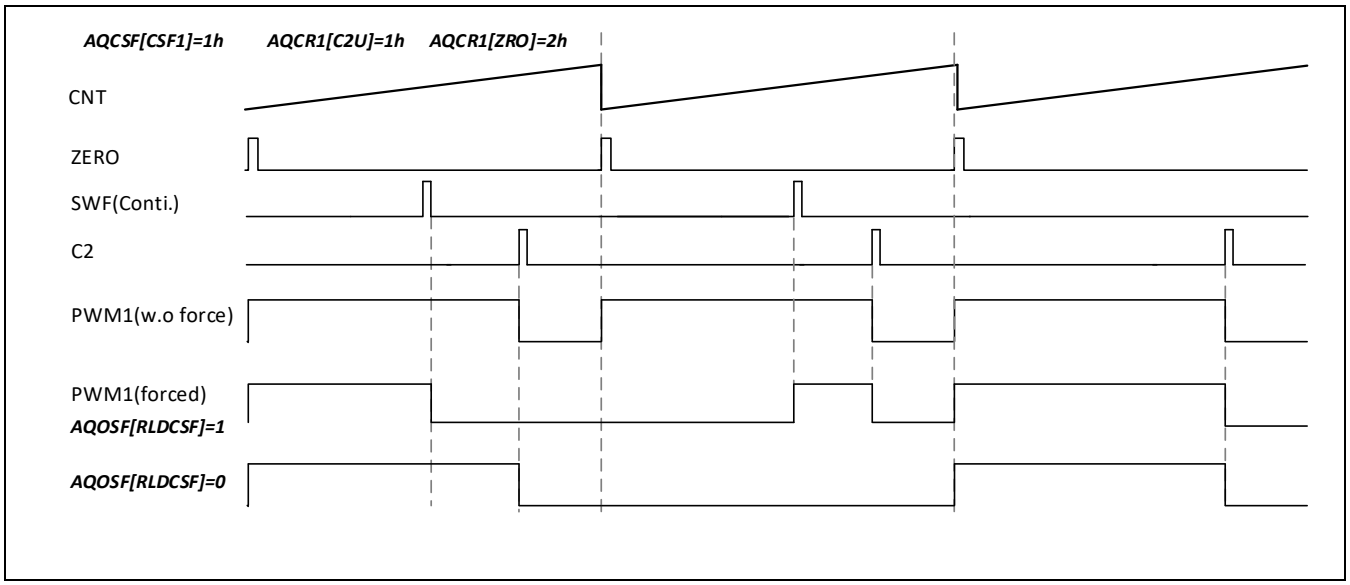


Figure 13-16 持续性软件强制输出

12.3.4.4 常见配置下的波形输出

下面的示例中，所有的条件都基于CMP值不变的情况。在实际系统中，用户可以在每个周期动态调整CMP的设置值。由于Shadow寄存器的作用，实际产生的波形可能晚于设置一个工作周期，或者在下一个计数周期开始时才改变，这都基于用户采用何种计数方式，以及Shadow寄存器的Load方式。

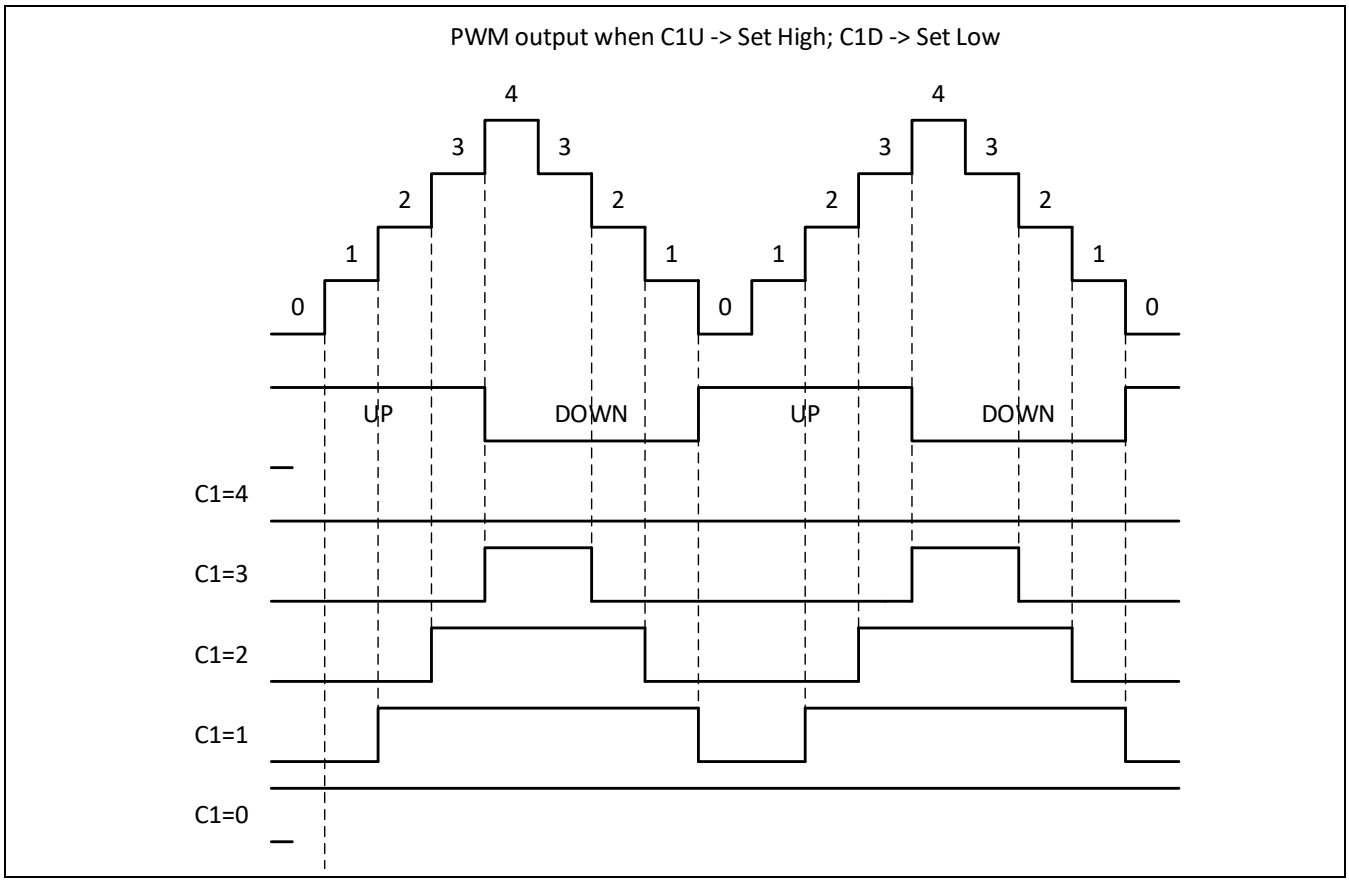


Figure 13-17 递增递减单比较值时，对称波形输出

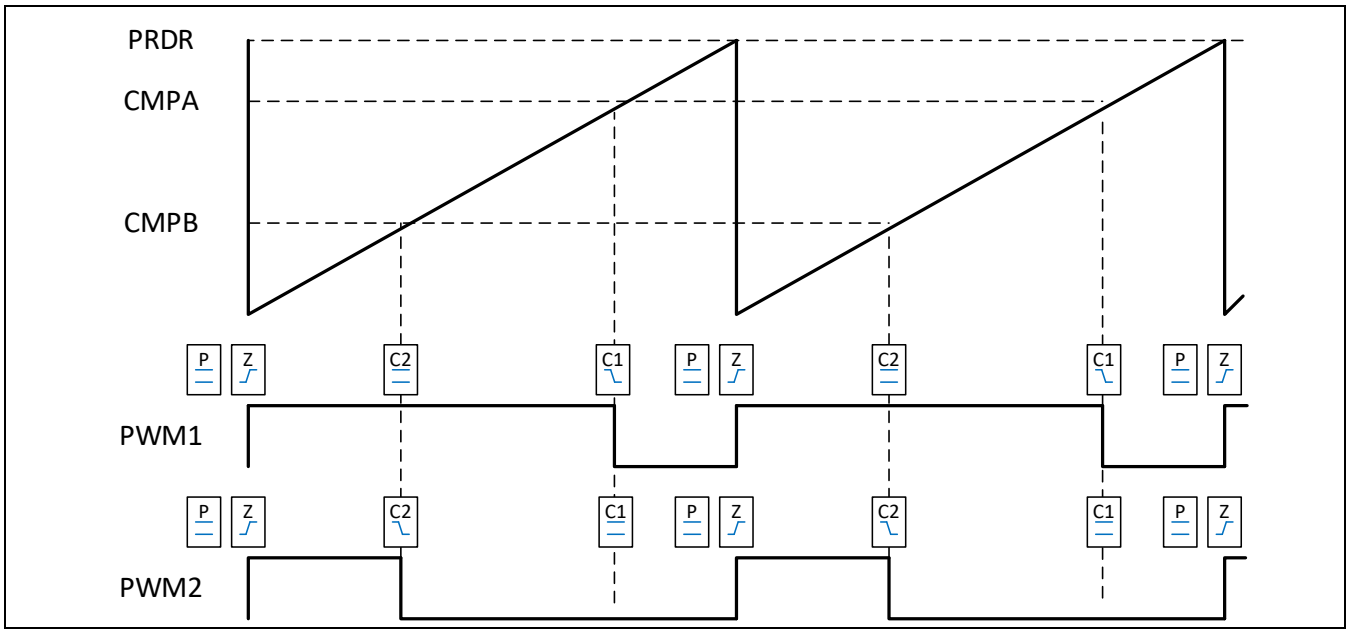


Figure 13-18 递增单沿，非对称波形输出

在上图中，不能很清楚的分辨出Zero和Period事件触发的区别，实际上，两个事件相差一个计数时钟。

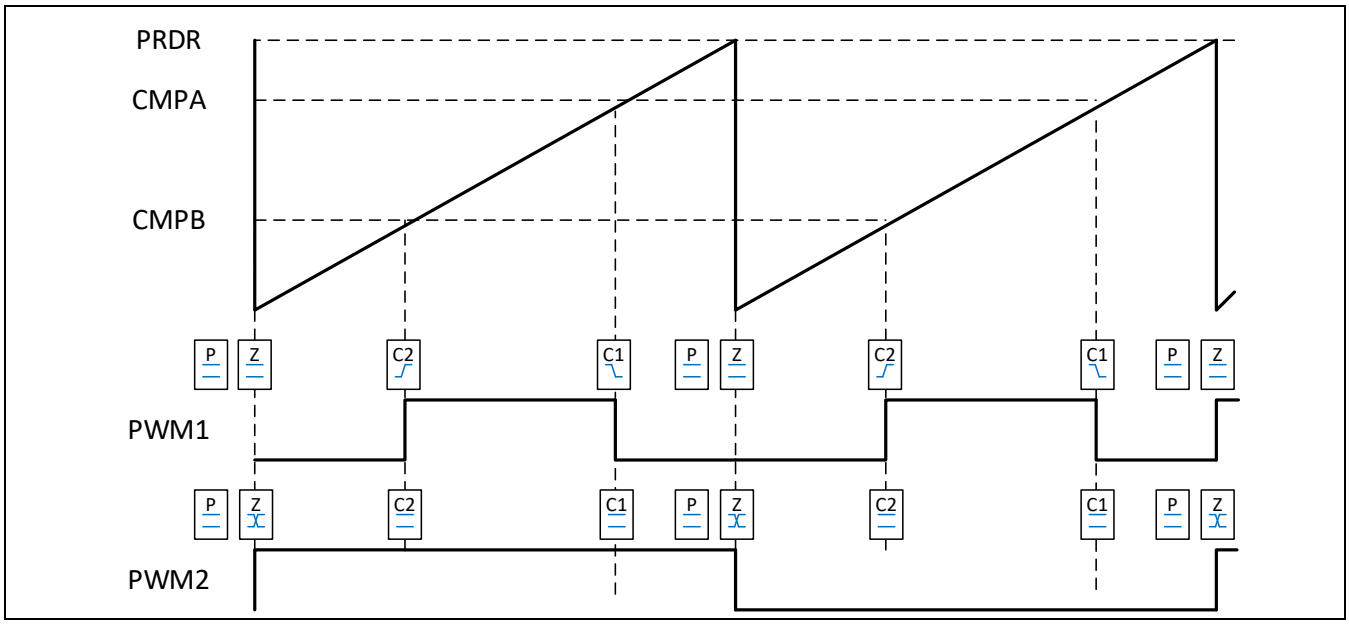


Figure 13-19 递增，脉冲定位非对称波形输出

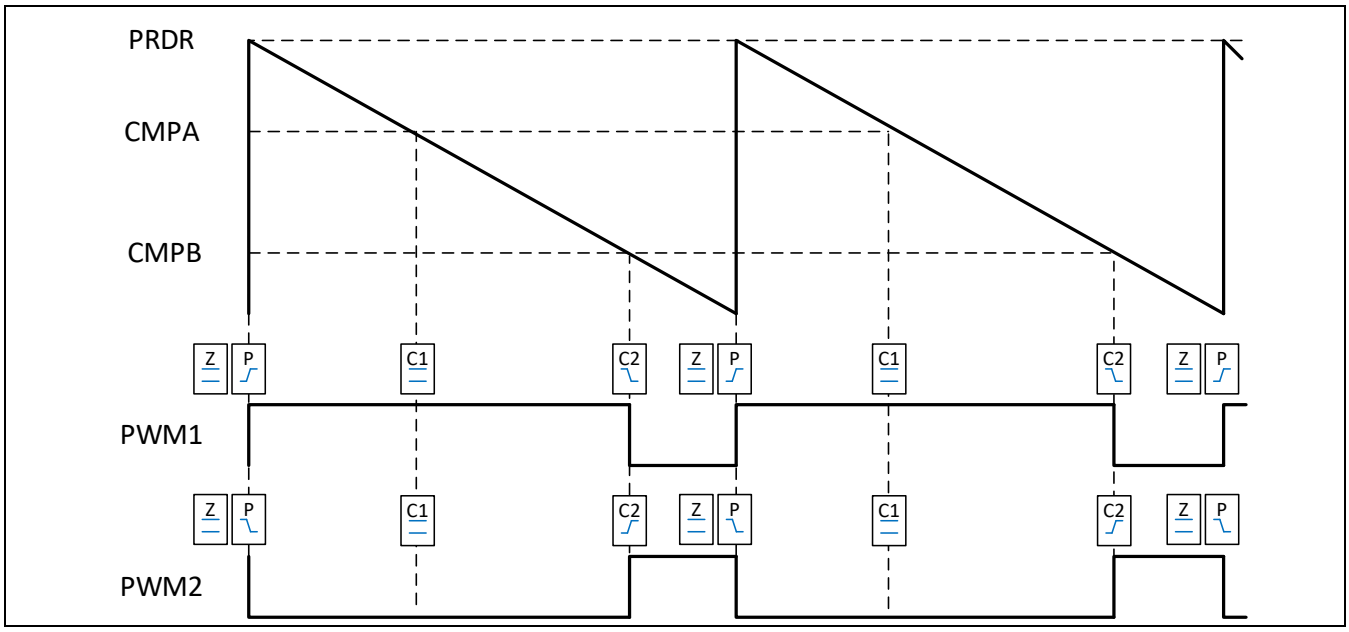


Figure 13-20 递减单沿，非对称波形输出

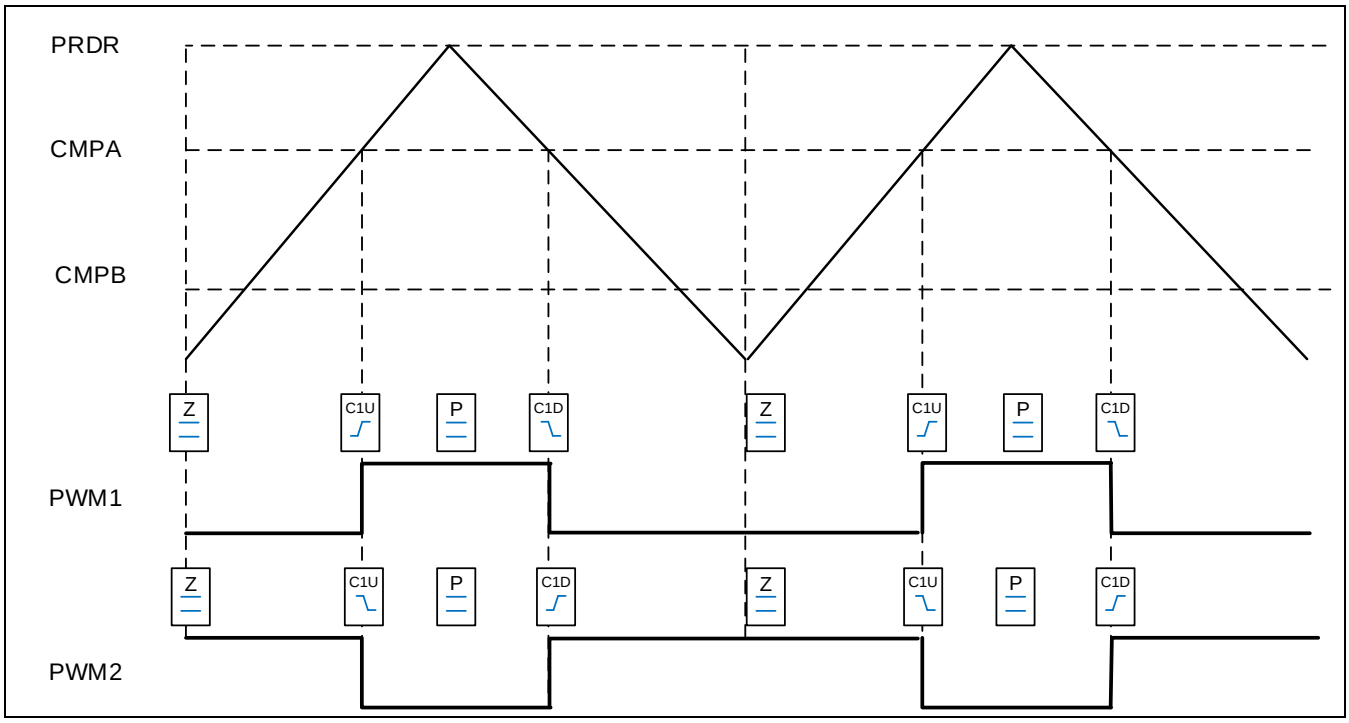


Figure 13-21 递增递减，双沿对称波形输出

12.3.5 捕捉模式

12.3.5.1 概述

捕捉模式一般用于如下几个常用的应用：

- 旋转机构的速度测量（比如霍尔传感器）
- 位置传感器的脉冲间隔时间测量
- 脉冲群的周期和占空比测量

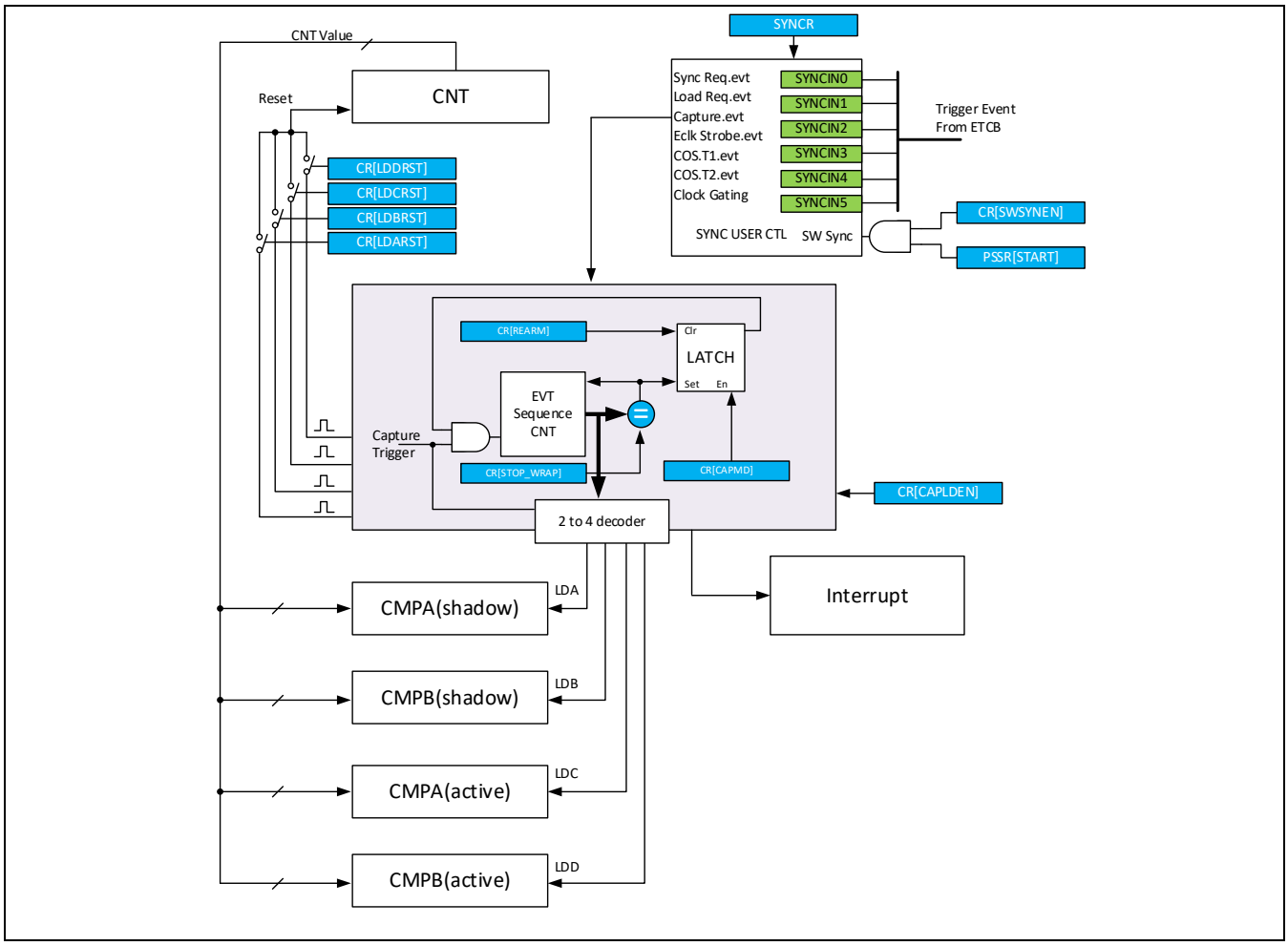


Figure 13-22 捕捉模式结构框图

当GPTA_CR[WAVE]控制位设置为0时，GPTA工作在捕捉模式。在捕捉模式下，捕捉的触发信号通过SYNCIN2端口输入。捕捉模块的主要功能特性如下：

- 支持2个捕获事件，捕获事件触发时，计数器值分别存入CMPA、CMPB的shadow寄存器中。
- 捕获序列控制，可支持最大连续2个计数值捕获
- 捕获后计数器重置或继续计数

当GPTA_CR[WAVE]控制位为零时，GPTA工作于捕获模式。在该模式下，捕获值将根据当前捕获事件序列计数器值被存储到相对应的寄存器中。在捕获模式下，比较值寄存器将作为捕获值存储功能使用。捕获事件序列计数器在检测到一次捕获触发事件（SYNCIN2上的脉冲输入）时，将自动递增一次。当序列计数器值计数值超出GPTA_CR[STOP_WRAP]的设置时，计数器自动清零，并重新开始计数。

12.3.5.2 捕获事件计数器

捕获的计数器值存入目标寄存器和触发相应捕获中断事件，与触发事件发生时，当前的序列计数器值相关。其对应关系如下表所示。

Table 13-6 捕获存储寄存器列表

EVT CNT	Load Target	Trigger Event	Description
0	CMPA(SHD)	CAP_LD0	Current counter value is loaded into CMPA shadow, CAP_LD0 is triggered
1	CMPB(SHD)	CAP_LD1	Current counter value is loaded into CMP shadow, CAP_LD1 is triggered

12.3.5.3 两种捕获模式

捕获支持两种工作方式，一次性捕获（One-shot）模式和连续捕获（Continouse）模式。模式设置可以通过GPTA_CR[CAPMD]控制位进行设置。在一次性捕获模式下，当序列计数器计数到STOP_WRAP后，计数器即停止工作，并禁止对CMPx的再次载入。只有通过软件再次使能后才能恢复（通过对GPTA_CR[REAMR]控制位置高，进行重新初始化）。在连续模式下，当捕获触发条件满足时，序列计数器超出STOP_WRAP后，会从零开始重新计数，若在新计数器值被捕获时，当前通道捕获标志已经置位，则捕获值覆盖标志位将被置位。捕获标志可以通过软件清除，或者在读取相应CMP寄存器后，硬件自动清除；捕获值覆盖标志必须通过软件清除。

可以通过设置GPTA_CR[LDxRST]位，决定相应捕获事件发生时是否需要清除计数器值。

12.3.5.4 捕获模式下的事件

捕获模式的启动事件：

捕获前，需要首先软件启动计数器，或者用SYNCIN0事件清除和启动计数器，这需要通过设置ETCB，连接GPTA SYNCIN0的输入事件。

捕获模式的捕获事件：

捕获事件一旦发生，就会触发计数器load操作。捕获事件即SYNCIN2。需要通过设置ETCB，连接GPTASYNCIN2的输入事件。计数器在每次捕获事件触发后，可以自动清零，通过设置GPTA_CR寄存器中[LDxRST]控制位进行设置。在触发事件发生时，相应的中断标志位被置位，可以通过使能相应的中断开关，控制进入CPU中断。

当同一个外部信号被同时配置为SYNCIN0和SYNCIN2时：

- 如果此时计数器在计数，该信号会被视作捕获事件。
- 如果此时计数器没有计数，该信号会被视作计数器启动事件。

12.3.5.5 应用举例

下面有一些例子，说明如何使用捕捉模式。

- **检测TIOA的高电平脉冲宽度，以及TIOB和TIOA的相位 (TIOA和TIAB为任意被预先设置为EXI的GPIO)**

One-shot模式，STOP_WRAP = 2，LDA/BRST = 1。设置TIOB上升沿为SYNCIN0输入，TIOA上升沿和下降沿都设为SYNCIN2输入。TIOB上升沿复位计数器，TIOA的下降沿触发第一次load，计数值存入CMPA中。TIOA下一个上升沿触发第二次load，计数值存入CMPB。计数器随即停止计数。此时CMPA的结果为相位差，CMPB的

结果为TIOA的高电平宽度。（Figure13-22）

- **检测TIOA上高电平脉冲宽度**

Continous 模式，STOP_WRAP = 1，LDA/BRST = 0。将TIOA设为EXIn（n<16），配置EXIn上升沿为SYNCIN0输入，同时将TIOA设为EXIm（m>16，扩展EXI），配置EXIm的下降沿为CMPA的SYNCIN2。第一个TIOA上升沿发生时，SYNC事件发生，计数器被复位。TIOA下降沿触发第一次load，计数值存入CMPA。所以，CMPA的结果为高电平宽度。（Figure13-23）

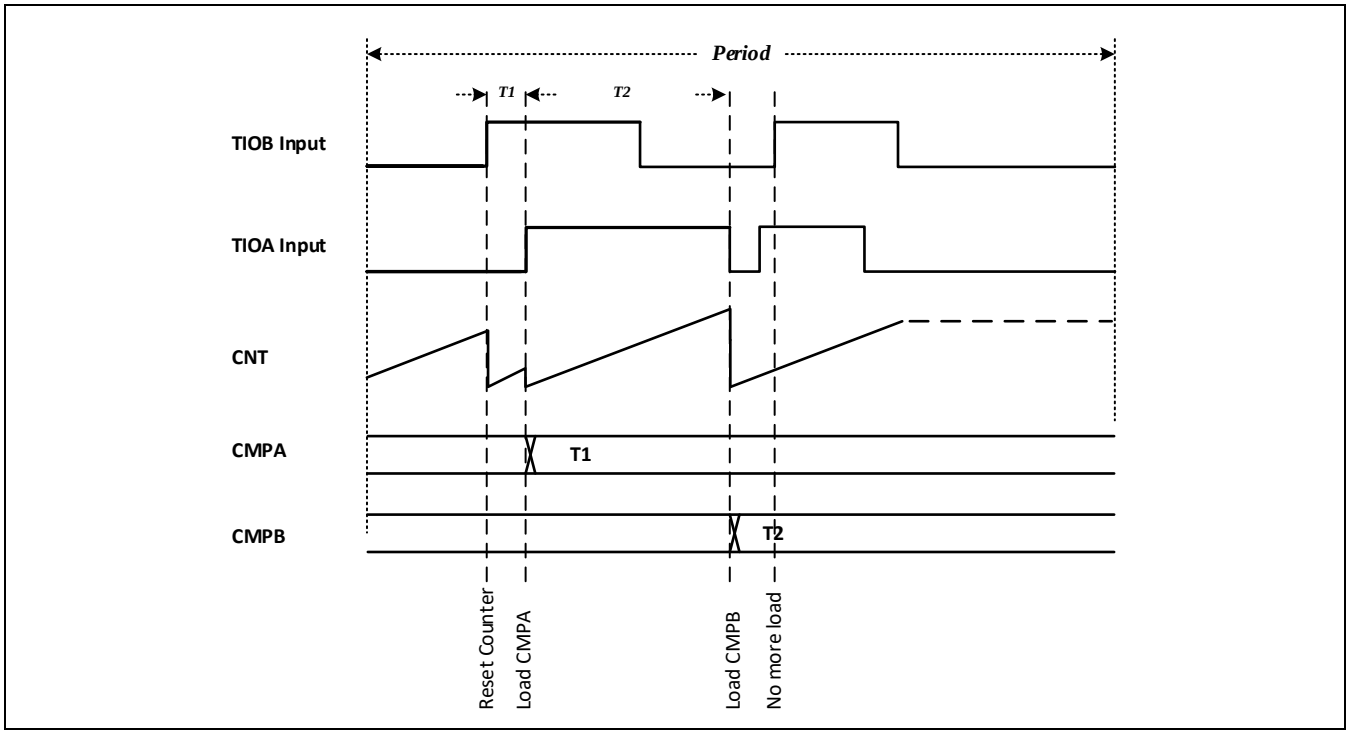


Figure 13-23 测量TIOA和TIOB相位差

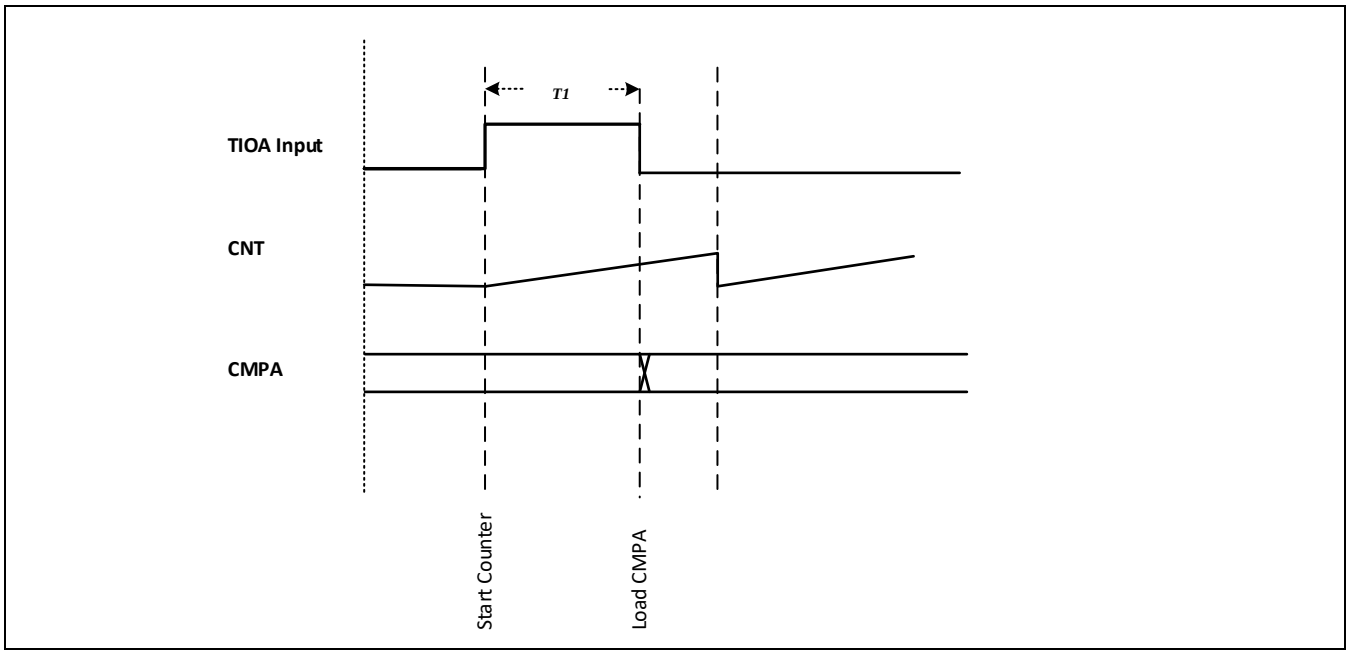


Figure 13-24 测量TIOA的脉冲宽度

12.3.6 单次触发模式

单次触发模式是一种特殊的工作模式，在此模式下，计数器在外部触发事件发生时，只产生一个延时和脉宽可编程的脉冲信号。计数器在启动后只进行一个周期的计数，在周期结束后，计数器Freeze。设置单次触发模式，可以通过寄存器GPTA_CR[OPM]控制位进行设置。在计数器Freeze状态下，计数器保持当前的计数值，直到有新的触发条件被检测到。

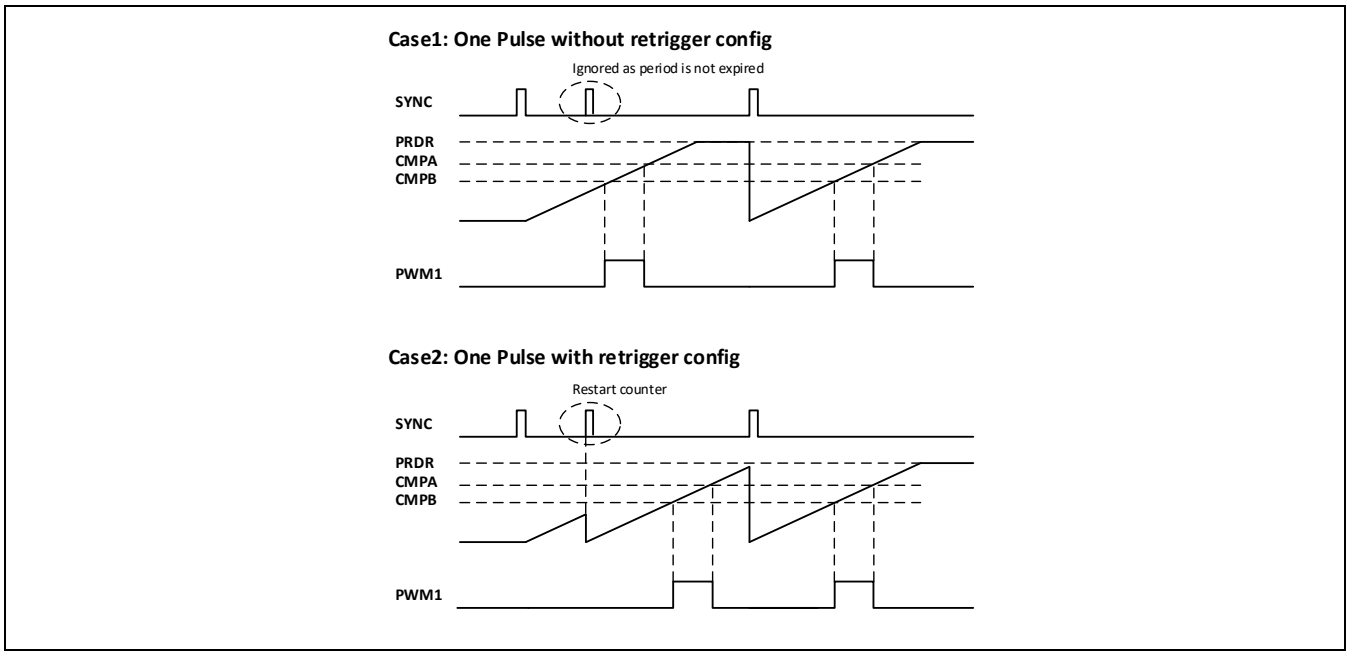


Figure 13-25 单次触发

单次触发模式工作时，缺省情况下，无论当前计数周期是否结束，当有新的触发被检测到时，计数器将开始重新计数。在某些应用中，需要在被触发后的一个计数周期内禁止新的触发，以保证得到一个完整的周期波形输出。在这种条件下，可以通过设置触发控制寄存器GPTA_SYNCR中的OSTMD控制位，将触发模式设置为一次性触发；或者将外部触发信号通过窗口滤波器滤波，屏蔽指定时间内的触发输入。

12.3.7 同步触发（输入）

同步触发功能用于在多个外设间通过硬件自动耦合同步不同外设的工作。GPTA通过同步输入接口接收来自于其他外设的触发信号，不同的触发端口对应独立的同步任务。当相应输入接口被触发，相对应的同步任务即被激活。同样，GPTA的事件输出接口，可用于产生对其他外设的任务的触发信号。

12.3.7.1 同步触发输入接口

GPTA支持模块间的同步触发功能，可以支持的触发功能包括如下几种：

- 重置和启动计数器
- 寄存器的更新（从Shadow寄存器更新到Active寄存器）
- 当前计数器值捕捉
- 计数器值递增或递减一个计数值
- 触发改变PWM的输出状态

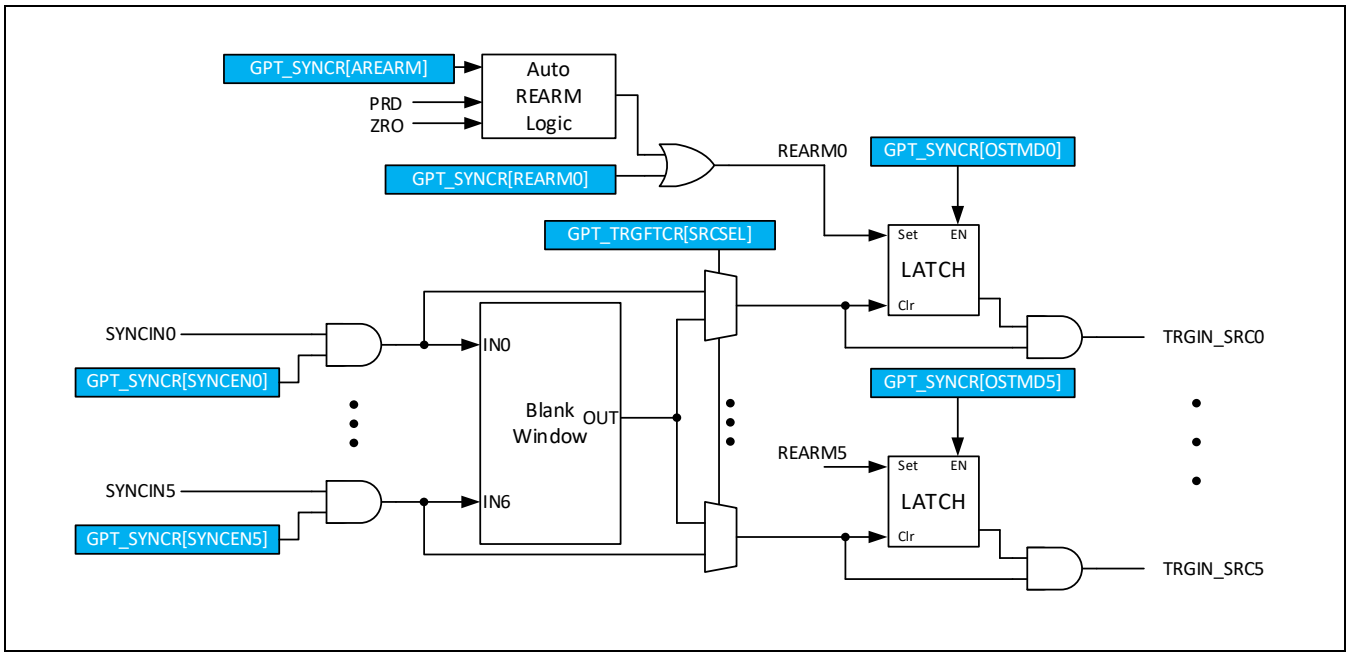


Figure 13-26 同步触发输入

每一个独立触发源被定义为SYNCIN端口，通过GPTA_SYNCR寄存器可以独立控制每个触发源的使能。触发源的输入为ETCB模块的输出，通过ETCB可以定义某个外设作为当前SYNCIN端口的触发信号源。具体配置参考ETCB章节。在触发输入接口中，包含一个事件滤波器，可以选择一个事件输入端口作为滤波器输入，对该事件进

行滤波处理。

每一个触发端口可以工作在两种工作模式：连续触发或者单次触发模式。在单次触发模式下，只运行一次触发发生，在检测到发生一次触发后，该端口将被禁止，直到软件重置该端口（REARM）后，才允许新的触发发生。重置端口也可以通过硬件自动完成，在设置GPTA_SYNCR[AREARM]后，在周期结束或者开始时，硬件会自动重置REARM，以保证在一个周期内只发生一次触发。

重置和启动计数器（SYNCIN0）

当该端口被触发，下列动作将被同时执行：

- 时基计数器（CNT）被重置。
- 时钟分频器重置。当该触发条件发生时，时钟分频器将重新开始计数
- 所有具有Shadow寄存器的控制寄存器将自动从Shadow更新Active寄存器

寄存器的更新（SYNCIN1）

当该端口被触发，所有具有Shadow寄存器的控制寄存器将自动从Shadow更新Active寄存器

计数器值捕捉（SYNCIN2）

当该端口被触发，将触发捕获事件。只有在GPTA_CR[WAVE]设置为捕获模式时，且GPTA_CR[CAPLDEN]控制位使能时，该触发事件才能被捕获模块检测到。

计数器值递增或递减一个计数值（SYNCIN3）

当该端口被触发，计数器将根据当前计数方向，自动增加或减少一个计数值。只有在GPTA_CEDR[CSS]控制位选择SYNCIN3时，该端口的触发才会被计数器检测到。

PWM输出状态改变（SYNCIN4/5）

SYNCIN 4用于产生内部T1触发事件，SYNCIN 5用于产生内部T2触发事件。

12.3.7.2 事件滤波器

在同步触发输入接口中，有一个事件滤波器，事件滤波器是一个时间窗口滤波器，它可以在一个特定的窗口周期内阻止输入信号的通过，或者只有在窗口内允许信号通过，从而实现去除干扰事件的目的。例如当模拟比较器的输出作为外部触发源时，可以滤除由于模拟比较器的抖动而产生的错误触发。可以选择任意一个SYNCIN端口作为事件滤波器的输入。

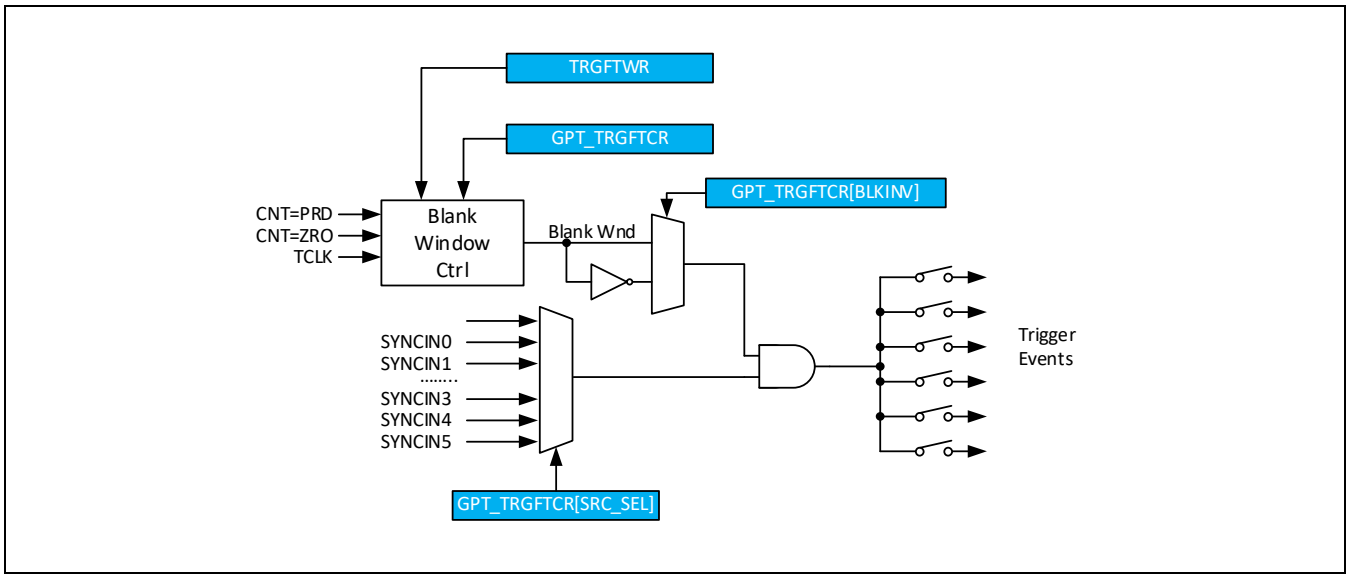


Figure 13-27 滤波器框图

如果使能窗逻辑被打开，在使能窗口内，或者窗口外，将禁止输入事件通过。窗口的激活时间可以设置为CNT=PRD，CNT=ZRO或者两个条件都可以（通过配置TRGFTCR[ALIGNMD]）。窗口的延时和宽度可以通过TRGFWR进行设置。

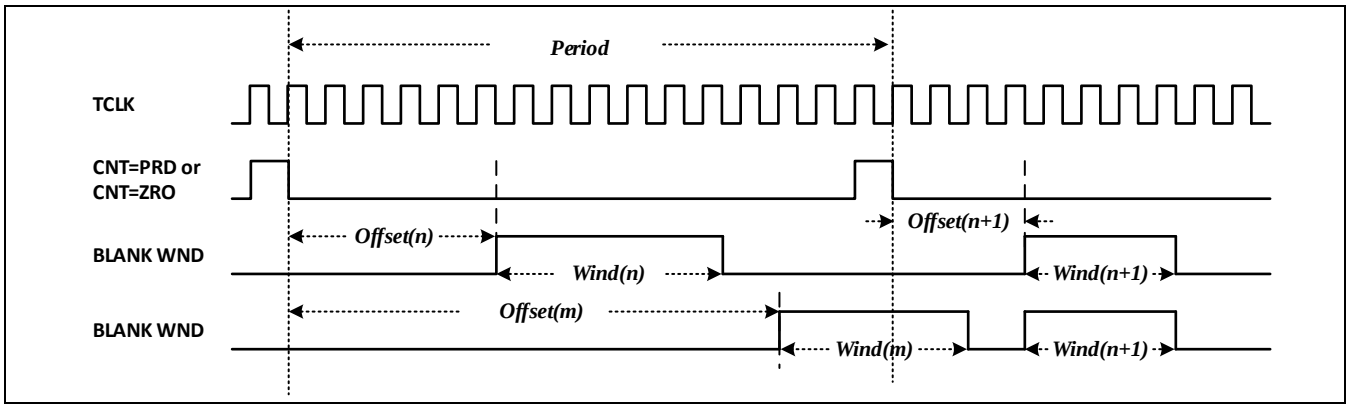


Figure 13-28 滤波器时序

12.3.8 事件触发（输出）

12.3.8.1 同步触发输出接口

事件触发输出接口支持4路事件触发输出，每个GPTA中断对应一个事件输出端口。可以通过GPTA_EVTRG控制寄存器选择GPTA中的任意一个事件作为每个中断的触发信号，事件中断触发信号可以通过TRGxOE控制位使能输出到其他外设，作为触发信号。

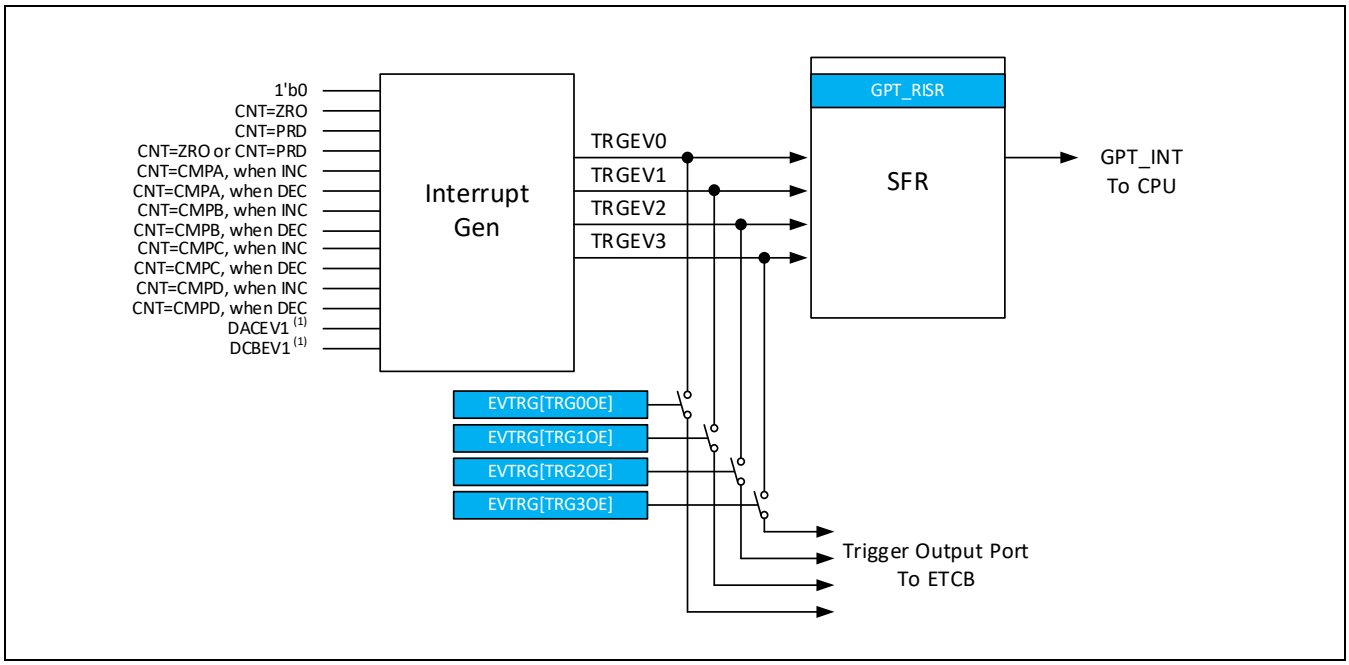


Figure 13-29 事件触发输出

12.3.8.2 事件计数和中断

触发中断基于各种触发事件，可以支持对触发事件计数的基础上决定是否产生中断请求。触发中断控制支持如下几种配置：

- 每次发生触发事件产生中断
- 每N次触发事情产生一次中断，N最大支持到15

中断模块一共支持4个中断源，每一个中断触发都支持事件计数器。中断的具体触发条件，可以通过GPTA_EVTRG寄存器进行选择。通过配置GPTA_EVPS可以配置每个中断事件计数器在计数到多少时产生一个中断请求。当中断发生次数等于GPTA_EVPS[TRGEVxPRD]控制位设置值时，将产生一次中断触发。计数器最大支持15个事件的计数，在中断请求发生后计数器将自动清除。当前已经计数的事件个数可以通过GPTA_EVPS[TRGEVxCNT]进行读取。CNT计数器具有Shadow功能，在缺省设置下，对CNT的读写时，操作对象是Shadow寄存器，Shadow寄存器的值会在同步事件发生时，或者计数器值等于PRD设置值时自动载入到活动寄存器中。当Shadow模式被禁止时，对CNT的操作直接影响活动寄存器的值。

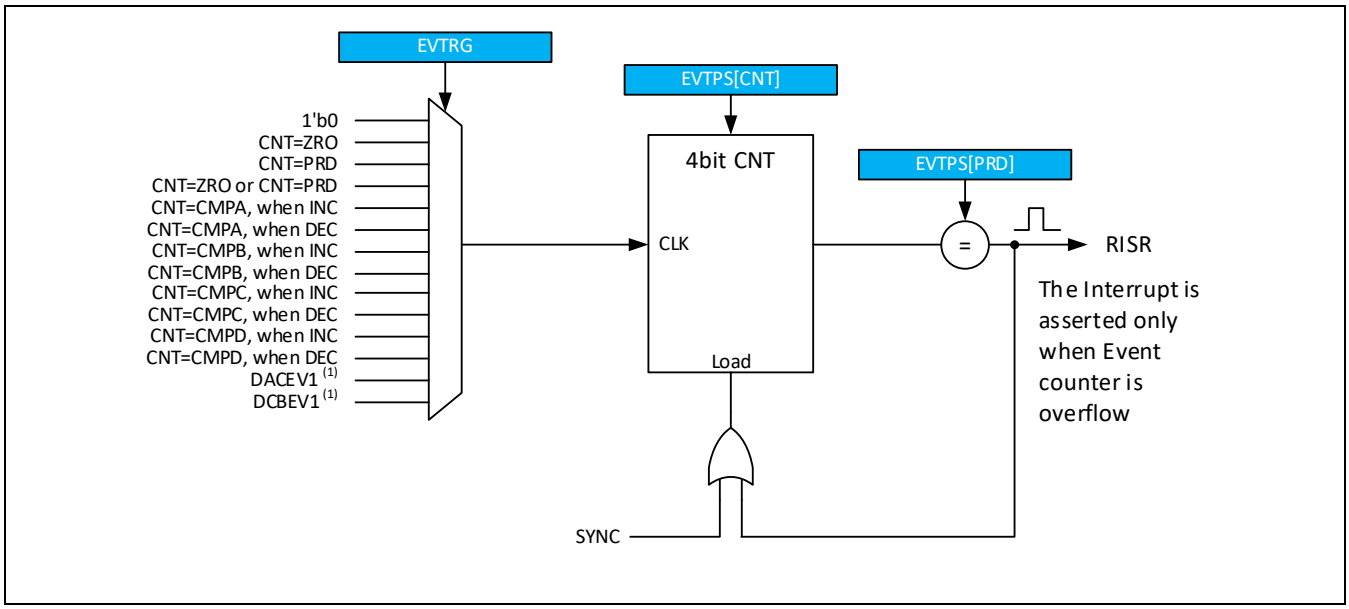


Figure 13-30 事件计数器

12.4 寄存器说明

12.4.1 寄存器表

- Base Address: GPTA0 : 0x4005_5000

Register	Offset	Description	Reset Value
GPTA_CEDR	0x00	ID & Clock Enable/Disable Register	0xBE980000
GPTA_RSSR	0x04	RESET/START/STOP Register	0x00000000
GPTA_PSCR	0x08	Counter Clock Prescaler Register	0x00000000
GPTA_CR	0x0C	General Control Register	0x00010010
GPTA_SYNCR	0x10	Synchronization Control Register	0x00000000
GPTA_GLDCR	0x14	Global Load Control Register	0x00000000
GPTA_GLDCFG	0x18	Global Load Configuration Register	0x00000000
GPTA_GLDCR2	0x1C	Global Load Control Register2	0x00000000
GPTA_PRDR	0x24	Period Register	0x00000000
GPTA_C1	0x2C	Compare A Register	0x00000000
GPTA_C2	0x30	Compare B Register	0x00000000
GPTA_CMPLDR	0x3C	Compare Data Load Control Register	0x00002490
GPTA_CNT	0x40	Counter Register	0x00000000
GPTA_AQLDR	0x44	Action Qualifier Load Control Register	0x00000024
GPTA_AQCR1	0x48	Action Qualifier Control Register 1	0x00000000
GPTA_AQCR2	0x4C	Action Qualifier Control Register 2	0x00000000
GPTA_AQOSF	0x5C	Action Qualifier One Shot Force Register	0x00000100
GPTA_AQCSF	0x60	Action Qualifier Continuous Force Register	0x00000000
GPTA_TRGFTCR	0xB8	Digital Compare Filter Control Register	0x00000000
GPTA_TRGFTWR	0xBC	Digital Compare Filter Window Register	0x00000000
GPTA_EVTRG	0xC0	Event Generation Control Register	0x00000000
GPTA_EVPS	0xC4	Event Counter Prescaler	0x00000000
GPTA_EVCNTINIT	0xC8	Event Counter Initial Value	0x00000000
GPTA_EVSWF	0xCC	Event Counter Load Control Register	0x00000000
GPTA_RISR	0xD0	Raw Interrupt Status Register	0x00000000
GPTA_MISR	0xD4	Masked Interrupt Status Register	0x00000000
GPTA_IMCR	0xD8	Interrupt Masking Control Register	0x00000000
GPTA_ICR	0xDC	Interrupt Clear Register	0x00000000

12.4.2 GPTA_CEDR (ID和时钟控制寄存器)

- Address = Base Address + 0x0000, Reset Value = 0xBE98_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																FLTCKPRS								RSVD	SHDWSTP	RSVD		CSS	DBGEN		CLKEN
1	0	1	1	1	1	1	0	1	0	0	1	1	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	R	RW	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
CLKEN	[0]	RW	时基计数器的时钟使能控制。 0h：计数器计数时钟禁止。 1h：计数器计数时钟使能。
DBGEN	[2:1]	RW	调试使能控制。调试使能时，在CPU被调试器挂起时，时基计数器的计数时钟同时也被挂起。 0h：调试禁止 1h：调试使能，PWM输出高阻 其他：调试使能，PWM输出保持
CSS	[3]	RW	计数器时钟源选择位。 0h：PCLK 1h：由SYNCIN3控制 其他：保留
SHDWSTP	[6]	RW	START控制位的Shadow功能使能控制。START置位不受此位控制，清除时受此位控制。当选择Shadow模式时，START控制位在周期结束时清除。 0h：Shadow模式 1h：Immediate模式
FLTCKPRS	[15:8]	RW	数字滤波器的时钟分频控制。 数字滤波器的时钟频率为PCLK/(FLTCKPRS+1)
IDCODE	[31:16]	RW	当前GPTA模块的版本信息。

12.4.3 GPTA_RSSR (启停控制寄存器)

- 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																SRR				RSVD								CNTDIR	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	W	W	W	W	R	R	R	R	R	R	R	R	R	R	R	R W

Name	Bit	Type	Description
START	[0]	RW	计数器启动控制位。 0h：当写‘0’时，停止计数器 1h：当写‘1’时，启动计数器 当对START位进行读取时，返回当前计数器工作状态 0h：计数器处于IDLE状态 1h：计数器正在工作 当GPTA_CR[SWSYNEN]控制位为低时，START控制位用于控制GPTA的启动，当GPTA启动后，再次写入START将被忽略； 当GPTA_CR[SWSYNEN]控制位为高时，START控制位用于软件触发同步事件，每次对START的写入，会产生一次外部Sync事件（等同于SYNCR中的SYNCIN0触发）。
CNTDIR	[3]	R	当前计数器计数方向状态。 0h：当前计数器方向为递减 1h：当前计数器方向为递增
SRR	[15:12]	RW	软件复位控制位。 当对当前控制位写入‘0x5’时，TIMER模块会被复位。复位后，所有寄存器都恢复为RESET状态。

12.4.4 GPTA_PSCR (时钟分频控制寄存器)

- 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																PSC															
0	0	0	0	0	0	0	0	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
PSC	[15:0]	RW	时钟分频控制。 TCLK作为时基模块的计时时钟和工作时钟。TCLK的时钟从PCLK分频得到。 TCLK的频率：$F_{TCLK} = F_{PCLK} / (PSC+1)$ 此寄存器具有Shadow寄存器，可通过GPTA_CR[PSCLD]设置载入的条件。

12.4.5 GPTA_CR (控制寄存器，捕捉模式 WAVE=0)

- Address = Base Address + 0x000C, Reset Value = 0001_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					LDDRST	LDCRST	LDBRST	LDARST	STOP_WRAP		CAPMD	REARM	WAVE	PSCLD		CGFLT			CGSRC		FLTIPSCLD	BURST	CAPLDEN	RSVD		PRDLD		IDLEST	SWSYNEN	CNTMD	
0	0	0	0	0	0	0	0	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	RW	RW	RW	RW	RW	RW	RW	W	RW	RW	RW	RW	RW	RW	RW	RW	W	RW	RW	R	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CNTMD	[1:0]	RW	计数模式设置。 计数模式一般只设置一次，并且在计数过程中不做改变。如果计数模式被改变，变化将发生在下一个TCLK的边沿，并且基于上一个计数器值进行递增或者递减。 00b：递增模式 01b：递减模式 10b：递增递减模式 11b：保留
SWSYNEN	[2]	RW	软件使能同步触发使能控制（RSSR 中 START 控制位）。 0h：设置SW START控制只用于启动。 1h：设置SW START控制用于启动和以产生一次SYNCIN0事件，以外部触发的方式重新启动。
PRDLD	[5:4]	RW	PRDR活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值。 00b：PRDR活动寄存器更新发生在周期结束(PEND) 01b：PRDR活动寄存器更新发生在SYNCIN1被触发时 10b：PRDR活动寄存器更新发生在计数器值等于零或SYNCIN1触发时 11b：立即更新，所有对PRDR操作直接作用于活动寄存器 [1]
CAPLDEN	[8]	RW	CMPA和CMPB在捕捉事件触发时，载入使能控制。 0h：禁止对CMP寄存器的捕获载入 1h：使能对CMP寄存器的捕获载入 此控制位在禁止对CMP寄存器载入时，并不影响捕捉事件SYNCIN2的触发。

BURST	[9]	RW	群脉冲模式。 0h：禁止群脉冲模式 1h：使能群脉冲模式
FLTIPSCLD	[10]	RW	数字滤波器初始化控制。对该控制写'1'可以初始化数字滤波器计数器. 0h：无效 1h：执行初始化
CGSRC	[12:11]	RW	群脉冲模式下，时钟门控的输入源选择。 0h：GPTA_CHA作为CG的输入源 1h：GPTA_CHB作为CG的输入源 其他：保留
CGFLT	[15:13]	RW	门控输入数字滤波控制。此控制定义了滤波器监测的步数，只有连续N次监测结果一致时，滤波器才输出有效的电平翻转。滤波器的采样时钟频率通过CEDR[FLTCKPRS] 控制位定义。 000b：Bypass 001b：N = 2 010b：N = 3 011b：N = 4 100b：N = 6 101b：N = 8 110b：N = 16 111b：N = 32
PSCLD	[17:16]	RW	PSCR活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值。 00b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 01b：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 10b：当CNT=ZRO或者PRD时，Shadow寄存器载入到Active寄存器中 11b：不进行载入
WAVE	[18]	RW	GPTA工作模式选择。 0h：捕捉模式 1h：波形发生模式
REARM	[19]	RW	重置CAPTURE控制。 0h：无效 1h：重置捕捉 重置时，捕捉事件计数器被清零，自动打开CAPLDEN

CAPMD	[20]	RW	捕捉模式设置。 0h：连续捕捉模式 1h：一次性捕捉模式
STOP_WRAP	[22:21]	RW	Capture模式下，捕获事件计数器周期设置值。（GPTA最大可设为2）
LDARST	[23]	RW	C1(Shadow)捕捉载入后，计数器值计数状态控制位。 0h：当前捕获触发后，计数器值不进行重置 1h：当前捕获触发后，计数器值进行重置
LDBRST	[24]	RW	C2(Shadow)捕捉载入后，计数器值计数状态控制位。 0h：当前捕获触发后，计数器值不进行重置 1h：当前捕获触发后，计数器值进行重置
LDCRST	[25]	RW	C1(Active)捕捉载入后，计数器值计数状态控制位。 0h：当前捕获触发后，计数器值不进行重置 1h：当前捕获触发后，计数器值进行重置
LDDRST	[26]	RW	C2(Active)捕捉载入后，计数器值计数状态控制位。 0h：当前捕获触发后，计数器值不进行重置 1h：当前捕获触发后，计数器值进行重置

注意 [1]：如果更新的周期值比更新前小，且立即更新发生时计数器已经超过更新的周期值，计数器将继续计数直到溢出，然后更新才会生效。

12.4.6 GPTA_CR (控制寄存器，波形输出模式：WAVE=1)

- Address = Base Address + 0x0010, Reset Value = 0001_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													WAVE	PSCLD		CGFLT			CGSRC		CKS	BURST	RSVD	RSVD	OPM	PRDLD		IDLEST	SWSYNEN	CNTMD	
0	0	0	0	0	0	0	0	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	R	R	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW	R	RW	RW	RW

Name	Bit	Type	Description
CNTMD	[1:0]	RW	计数模式设置。 计数模式一般只设置一次，并且在计数过程中不做改变。如果计数模式被改变，变化将发生在下一个TCLK的边沿，并且基于上一个计数器值进行递增或者递减。 00b：递增模式 01b：递减模式 10b：递增递减模式 11b：保留
SWSYNEN	[2]	RW	软件使能同步触发使能控制（RSSR 中 START 控制位）。 0h：设置SW START控制只用于启动。 1h：设置SW START控制用于启动和以产生一次外部触发的方式重新启动。
IDLEST	[3]	RW	波形输出被停止时，输出端口的缺省状态。 0h：高阻输出 1h：低电平输出
PRDLD	[5:4]	RW	PRDR活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值。 0h：PRDR活动寄存器更新发生在计数器值等于零（CNT=ZRO） 1h：PRDR活动寄存器更新发生在计数器值等于零和同步（CNT=ZRO & SYNC） 2h：PRDR活动寄存器更新发生在同步事件触发（SYNC） 3h：立即更新，所有对PRDR操作直接作用于活动寄存器 [1]
OPM	[6]	RW	计数器单次触发工作模式选择。 0h：连续计数工作模式 1h：单次触发工作模式 其他：保留

BURST	[9]	RW	群脉冲模式。 0h：禁止群脉冲模式 1h：使能群脉冲模式
CKS	[10]	RW	采样时钟频率控制位。此控制位决定数字滤波器的采样时钟频率。 0h：PCLK 1h：PCLK/2
CGSRC	[12:11]	RW	群脉冲模式下，时钟门控的输入源选择。 0h：GPTA_CHA作为CG的输入源 1h：GPTA_CHB作为CG的输入源 其他：保留
CGFLT	[15:13]	RW	门控输入数字滤波控制。此控制定义了滤波器监测的步数，只有连续N次监测结果一致时，滤波器才输出有效的电平翻转。滤波器的采样时钟频率在通过CKS控制位定义。 000b：Bypass 001b：N = 2 010b：N = 3 011b：N = 4 100b：N = 6 101b：N = 8 110b：N = 16 111b：N = 32
PSCLD	[17:16]	RW	PSCR活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值。 0h：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 1h：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 2h：当CNT=ZRO或者PRD时，Shadow寄存器载入到Active寄存器中 3h：不进行载入
WAVE	[18]	RW	GPTA工作模式选择。 0h：捕捉模式 1h：波形发生模式

注意 [1]：如果更新的周期值比更新前小，且立即更新发生时计数器已经超过更新的周期值，计数器将继续计数直到溢出，然后更新才会生效。

12.4.7 GPTA_SYNCR (同步控制寄存器)

- 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
AREARM		TRGO1SEL			TRGO0SEL			TXREARM0		REARM5	REARM4	REARM3	REARM2	REARM1	REARM0	RSVD		OSTMD5	OSTMD4	OSTMD3	OSTMD2	OSTMD1	OSTMD0	RSVD		SYNCEN5	SYNCEN4	SYNCEN3	SYNCEN2	SYNCEN1	SYNCEN0
0	0	0	0	0	0	0	0	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	R	R	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
SYNCENx	[5:0]	RW	外部同步触发使能控制。 0：禁止当前触发输入通道 1：使能当前触发输入通道 SYNCIN0：外部Sync事件 SYNCIN1：Load触发 SYNCIN2：Capture触发事件 SYNCIN3：CNT增减一拍触发事件 SYNCIN4：外部COS事件（用于PWM波形输出控制） SYNCIN5：外部COS事件（用于PWM波形输出控制）
OSTMDx	[13:8]	RW	一次性同步触发模式选择。 0h：连续触发模式 1h：一次性触发模式 当该输入通道被设置为一次性触发模式后，在一次触发事件被检测到后，该通道将不允许后续的触发事件通过，直到被软件重置（REARM）后才允许新的触发事件通过。
REARMx	[21:16]	RW	在一次性同步触发模式下，软件重置当前通道状态控制位。 当读取时，返回当前通道状态 0h：允许触发 1h：已经检测到触发，不允许后续触发 当写入时， 0h：无效 1h：清除当前通道状态，并允许新的触发
TXREARM0	[23:22]	RW	Tx 信号触发 SYNCIN0 的 REARM

			0：禁止硬件自动REARM 1：T1发生触发，自动REARM SYNCIN0通道 2：T2发生触发，自动REARM SYNCIN0通道 3：T1 或者 T2 发生触发，自动 REARM SYNCIN0 通道
TRGO0SEL	[26:24]	RW	输入触发通道直通作为TRGSRC0的ExtSync条件的选择。只有当EVTRG寄存器中TRGSRC0控制位选择为ExtSync条件时有效。 0h：选择SYNCIN0作为TRGSRC0的ExtSync触发 1h：选择SYNCIN1作为TRGSRC0的ExtSync触发 2h：选择SYNCIN2作为TRGSRC0的ExtSync触发 3h：选择SYNCIN3作为TRGSRC0的ExtSync触发 4h：选择SYNCIN4作为TRGSRC0的ExtSync触发 5h：选择SYNCIN5作为TRGSRC0的ExtSync触发 其他：保留
TRGO1SEL	[29:27]	RW	输入触发通道直通作为TRGSRC1的ExtSync条件的选择。只有当EVTRG寄存器中TRGSRC1控制位选择为ExtSync条件时有效。 0h：选择SYNCIN0作为TRGSRC1的ExtSync触发 1h：选择SYNCIN1作为TRGSRC1的ExtSync触发 2h：选择SYNCIN2作为TRGSRC1的ExtSync触发 3h：选择SYNCIN3作为TRGSRC1的ExtSync触发 4h：选择SYNCIN4作为TRGSRC1的ExtSync触发 5h：选择SYNCIN5作为TRGSRC1的ExtSync触发 其他：保留
AREARM	[31:30]	RW	硬件自动REARM控制位。 0：禁止硬件自动REARM 1：CNT = ZRO时，自动REARM 2：CNT = PRD时，自动REARM 3：CNT = ZRO or CNT = PRD时，自动REARM

12.4.8 GPTA_GLDCR (全局载入控制寄存器)

- 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																			GLDCNT			GLDPRD			RSVD	OSTMD	GLDMD				GLDEN
0	0	0	0	0	0	0	0	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	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
GLDEN	[0]	RW	全局的Shadow到Active寄存器载入控制。 0：使用独立的单个配置（在各个寄存器中LDMD控制位分别指派的载入控制） 1：使用GLDMD中的设置，其他设置被屏蔽
GLDMD	[4:1]	RW	全局载入触发事件选择。 0h：CNT = ZRO 1h：CNT = PRD 2h：CNT = ZRO or CNT = PRD 3h：CNT = ZRO or 外部LOAD触发或SYNC触发 4h：CNT = PRD or 外部LOAD触发或SYNC触发 5h：CNT = ZRO or CNT = PRD or 外部LOAD触发或SYNC触发 Others：Reserved Fh：在GLDCR2[GFRCLD]写入‘1’时 [1]
OSTMD	[5]	RW	One Shot 载入模式使能控制位 0h：禁止One Shot模式，只要条件满足，Active寄存器都会从Shadow寄存器载入 1h：使能One Shot模式，只有在GLDCR2[OSREARM]写入‘1’后，才会进行一次载入。一旦载入被触发，需要再次对GLDCR2[OSREARM]写入‘1’，才能允许下一次载入触发。
GLDPRD	[9:7]	RW	全局载入触发周期选择。 可以选择N次触发条件满足后，才进行一次全局载入。 000b：Disable Counter（立即触发） 001b：第2次条件满足时触发 010b：第3次条件满足时触发 011b：第4次条件满足时触发

			100b：第5次条件满足时触发 101b：第6次条件满足时触发 110b：第7次条件满足时触发 111b：第8次条件满足时触发
GLDCNT	[12:10]	RW	全局载入事件计数器。 计数器值表示当前已发生多少次事件触发。

注意 [1]：如果更新的周期值比更新前小，且立即更新发生时计数器已经超过更新的周期值，计数器将继续计数直到溢出，然后更新才会生效。类似，如果更新的比较值比更新前小，且立即更新发生时计数器已经超过更新的比较值，本周期将不会发生match事件。

12.4.9 GPTA_GLDCFG (全局载入配置)

- 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																			AQCSF			AQCR2	AQCR1						CMPB	CMPA	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	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	RW	RW	R	R	R	R	R	RW	RW	RW

Name	Bit	Type	Description
PRDR	[0]	RW	PRDR寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
CMPA	[1]	RW	CMPA寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
CMPB	[2]	RW	CMPB寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
AQCR1	[8]	RW	AQCRA寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
AQCR2	[9]	RW	AQCRB寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
AQCSF	[12]	RW	AQCSF寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置

12.4.10 GPTA_GLDCR2 (全局载入控制寄存器2)

- 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																														GFRCLD	OSREARM
0	0	0	0	0	0	0	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	W	RW

Name	Bit	Type	Description
OSREARM	[0]	RW	重置ONE SHOT模式 0：写入‘0’无效，读取时总是返回‘0’ 1：重置ONE SHOT模式。ONE SHOT模式下，一次触发后，需要重置模式才允许再次触发
GFRCLD	[1]	RW	软件产生一次GLD触发。 0：写入‘0’无效，读取时总是返回‘0’ 1：软件产生一次GLD触发事件

12.4.11 GPTA_PRDR (周期设置寄存器)

- 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
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	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
PRDR	[15:0]	RW	时基控制周期寄存器。 此控制位决定了PWM输出波形的周期值。通过设置GPTA_CR[PRDLD]可以选择Shadow到Active载入的触发条件。

12.4.12 GPTA_CMPA (比较值A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
OVWRT	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
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
CMPA	[15:0]	RW	比较值A寄存器。 此寄存器有Shadow寄存器，Shadow模式可以通过CMPLDR[SHDWCMPA]进行设置。在Shadow模式下，可以通过CMPLDR[LDAMD]选择Shadow到Active载入的触发条件。在写入前，可以通过SHDWAFULL控制位检测当前寄存器状态。 当工作于Capture模式下，此寄存器对应CAPLD0事件触发的捕获值。
OVWRT	[31]	RW	Over Write Flag 标志位。 表示当前Capture值是否被Over Written。当连续两次针对于当前寄存器的捕获事件发生，而两次捕获之间未对当前寄存器进行过读取操作，则Over Written标志被置位。读取当前寄存器会自动清除Over Written标志。 此标志位只有在Capture模式下有效。

12.4.13 GPTA_CMPB (比较值B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
OVWRT	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
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
CMPB	[15:0]	RW	比较值B寄存器。 此寄存器有Shadow寄存器，Shadow模式可以通过CMPLDR[SHDWCMPB]进行设置。在Shadow模式下，可以通过CMPLDR[LDBMD]选择Shadow到Active载入的触发条件。在写入前，可以通过SHDWBFULL控制位检测当前寄存器状态。 当工作于Capture模式下，此寄存器对应CAPLD1事件触发的捕获值。
OVWRT	[31]	RW	Over Write Flag 标志位。 表示当前Capture值是否被Over Written。当连续两次针对于当前寄存器的捕获事件发生，而两次捕获之间未对当前寄存器进行过读取操作，则Over Written标志被置位。读取当前寄存器会自动清除Over Written标志。 此标志位只有在Capture模式下有效。

12.4.14 GPTA_CMPLDR (比较值载入控制寄存器)

- Address = Base Address + 0x003C, Reset Value = 0x0000_2490

31	30	29	28	27	26	25	24	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										SHDWBFULL	SHDWAFULL	RSVD										LDBMD		LDAMD				RSVD		SHDWCMPA	SHDWCMPA
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	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	RW	RW	RW	RW	RW	RW	R	R	RW	RW

Name	Bit	Type	Description
SHDWCMPA	[0]	RW	CMPA的Shadow功能使能控制。 0h：Shadow模式 1h：Immediate模式 [1]
SHDWCMPB	[1]	RW	CMPB的Shadow功能使能控制。 0h：Shadow模式 1h：Immediate模式 [1]
LDAMD	[6:4]	RW	Shadow模式下，Active CMPA从Shadow CMPA载入控制。 xx1b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 x1xb：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 1xxb：外部LOAD触发或SYNC触发时，Shadow寄存器载入到Active寄存器中 000b：不进行载入 每个控制位分别对应一个触发条件，可以同时使能多个触发条件。例如，当设置011b时，CNT=ZRO或CNT=PRD时，都会触发寄存器载入。
LDBMD	[9:7]	RW	Shadow模式下，Active CMPB从Shadow CMPB载入控制。 xx1b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 x1xb：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 1xxb：外部LOAD触发或SYNC触发时，Shadow寄存器载入到Active寄存器中 000b：不进行载入
SHDWAFULL	[20]	R	CMPA的Shadow寄存器非空标志位。 当对CMPA进行写操作时，该标志位置位。该标志位在Shadow被载入到Active后，会自动清除。 0h：Shadow空

			1h：Shadow非空，对当前CMP寄存器写入会覆盖Shadow中未被载入的值
SHDWAFULL	[21]	R	CMPB的Shadow寄存器非空标志位。 当对CMPB进行写操作时，该标志位置位。该标志位在Shadow被载入到Active后，会自动清除。 0h：Shadow空 1h：Shadow非空，对当前CMP寄存器写入会覆盖Shadow中未被载入的值

注意 [1]：如果更新的比较值比更新前小，且立即更新发生时计数器已经超过更新的比较值，本周期将不会发生match事件。

12.4.15 GPTA_CNT (时基计数器寄存器)

- 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
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
CNT	[15:0]	RW	时基计数器寄存器。 对CNT读取时，返回当前计数器值。对CNT写入时，将直接更新CNT的计数值。 CNT计数器没有Shadow寄存器，CPU的写入将直接影响当前计数器值。

12.4.16 GPTA_AQLDR (波形输出载入控制寄存器)

- Address = Base Address + 0x0044, Reset Value = 0x0000_0024

31	30	29	28	27	26	25	24	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																								LDBMD			LDAMD			SHDWAQ2	SHDWAQ1
0	0	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	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	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
SHDWAQ1	[0]	RW	AQCRA寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
SHDWAQ2	[1]	RW	AQCRB寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
LDAMD	[4:2]	RW	Shadow模式下，Active AQCRA从Shadow载入控制。 xx1b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 x1xb：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 1xxb：外部LOAD触发或SYNC触发时，Shadow寄存器载入到Active寄存器中 000b：不进行载入 每个控制位分别对应一个触发条件，可以同时使能多个触发条件。例如，当设置011b时，CNT=ZRO或CNT=PRD时，都会触发寄存器载入。
LDBMD	[7:5]	RW	Shadow模式下，Active AQCRB从Shadow载入控制。 xx1b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 x1xb：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 1xxb：外部LOAD触发或SYNC触发时，Shadow寄存器载入到Active寄存器中 000b：不进行载入 每个控制位分别对应一个触发条件，可以同时使能多个触发条件。例如，当设置011b时，CNT=ZRO或CNT=PRD时，都会触发寄存器载入。

12.4.17 GPTA_AQCR1 (PWM1波形输出控制寄存器)

- 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								C2SEL		C1SEL		T2D		T2U		T1D		T1U		C2D		C2U		C1D		C1U		PRD		ZRO	
0	0	0	0	0	0	0	0	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
ZRO	[1:0]	RW	当CNT值等于零时，在PWM1上做出的波形输出动作定义。 在递增递减模式时，当计数器值等于零时，计数方向为递增模式 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
PRD	[3:2]	RW	当CNT值等于PRDR时，在PWM1上做出的波形输出动作定义。 在递增递减模式时，当计数器值等于PRDR时，计数方向为递减模式 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1U	[5:4]	RW	当CNT值等于C1，且此时计数方向为递增时，在PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1D	[7:6]	RW	当CNT值等于C1，且此时计数方向为递减时，在PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）

C2U	[9:8]	RW	当CNT值等于C2，且此时计数方向为递增时，在PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C2D	[11:10]	RW	当CNT值等于C2，且此时计数方向为递减时，在PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1U	[13:12]	RW	当T1事件发生，且此时计数方向为递增时，在PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1D	[15:14]	RW	当T1事件发生，且此时计数方向为递减时，在PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T2U	[17:16]	RW	当T2事件发生，且此时计数方向为递增时，在PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T2D	[19:18]	RW	当T2事件发生，且此时计数方向为递减时，在PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）

C1SEL	[21:20]	RW	C1比较值的数据源选择。 0h：CMPA寄存器作为C1的数据源。 1h：CMPB寄存器作为C1的数据源。 其他：保留。
C2SEL	[23:22]	RW	C2比较值数据源选择。 0h：CMPA寄存器作为C2的数据源。 1h：CMPB寄存器作为C2的数据源。 其他：保留。

12.4.18 GPTA_AQCR2 (PWM2波形输出控制寄存器)

- 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								C2SEL		C1SEL		T2D		T2U		T1D		T1U		C2D		C2U		C1D		C1U		PRD		ZRO	
0	0	0	0	0	0	0	0	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
ZRO	[1:0]	RW	当CNT值等于零时，在PWM2上做出的波形输出动作定义。 在递增递减模式时，当计数器值等于零时，计数方向为递增模式 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
PRD	[3:2]	RW	当CNT值等于PRDR时，在PWM2上做出的波形输出动作定义。 在递增递减模式时，当计数器值等于PRDR时，计数方向为递减模式 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1U	[5:4]	RW	当CNT值等于CMPA，且此时计数方向为递增时，在PWM2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1D	[7:6]	RW	当CNT值等于CMPA，且此时计数方向为递减时，在PWM2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）

C2U	[9:8]	RW	当CNT值等于CMPB，且此时计数方向为递增时，在PWM2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C2D	[11:10]	RW	当CNT值等于CMPB，且此时计数方向为递减时，在PWM2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1U	[13:12]	RW	当T1事件发生，且此时计数方向为递增时，在PWM2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1D	[15:14]	RW	当T1事件发生，且此时计数方向为递减时，在PWM2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T2U	[17:16]	RW	当T2事件发生，且此时计数方向为递增时，在PWM2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T2D	[19:18]	RW	当T2事件发生，且此时计数方向为递减时，在PWM2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）

C1SEL	[21:20]	RW	C1比较值的数据源选择。 0h：CMPA寄存器作为C1的数据源。 1h：CMPB寄存器作为C1的数据源。 其他：保留。
C2SEL	[23:22]	RW	C2比较值数据源选择。 0h：CMPA寄存器作为C2的数据源。 1h：CMPB寄存器作为C2的数据源。 其他：保留。

12.4.19 GPTA_AQOSF (一次性软件强制输出控制)

- Address = Base Address + 0x005C, 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
RSVD														RLDCSF		RSVD								RSVD	ACT2		OSTSF2	RSVD	ACT1		OSTSF1
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	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	R	R	R	RW	RW	W	R	RW	RW	W

Name	Bit	Type	Description
OSTSF1	[0]	RW	在PWM1上产生一次性软件强制输出。 0h：对当前位写‘0’无效 1h：产生一次性软件强制输出，此输出状态保持，直到有其他改变PWM1输出状态的触发事件发生。
ACT1	[2:1]	RW	当软件强制输出时，PWM1上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
OSTSF2	[4]	RW	在PWM2上产生一次性软件强制输出。 0h：对当前位写‘0’无效 1h：产生一次性软件强制输出，此输出状态保持，直到有其他改变PWM1输出状态的触发事件发生。
ACT2	[6:5]	RW	当软件强制输出时，PWM2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
RLDCSF	[17:16]	RW	AQCSF寄存器从Shadow载入到Active的控制。 01b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 10b：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 11b：当CNT=ZRO或者PRD时，Shadow寄存器载入到Active寄存器中 00b：立即载入

12.4.20 GPTA_AQCSF (连续软件强制输出控制)

- 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																												CSF2		CSF1	
0	0	0	0	0	0	0	0	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
CSF1	[1:0]	RW	通过软件对PWM1做连续强制赋值。在立即更新模式下，赋值将在配置后下一个TCLK输出。在Shadow模式下，在Shadow更新到Active后的下一个TCLK后输出。对Shadow更新到Active的控制，可以通过AQOSF寄存器中的RLDCSF控制位进行配置。 0h：禁止强制赋值 1h：强制输出低 2h：强制输出高 3h：禁止强制赋值
CSF2	[3:2]	RW	通过软件对PWM2做连续强制赋值。在立即更新模式下，赋值将在配置后下一个TCLK输出。在Shadow模式下，在Shadow更新到Active后的下一个TCLK后输出。对Shadow更新到Active的控制，可以通过AQOSF寄存器中的RLDCSF控制位进行配置。 0h：禁止强制赋值 1h：强制输出低 2h：强制输出高 3h：禁止强制赋值

12.4.21 GPTA_TRGFTCR (数字比较器滤波窗控制寄存器)

- 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
																								CROSSMD	ALIGNMD		BLKINV	RSVd	SRCSEL		
0	0	0	0	0	0	0	0	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	R	RW	RW	RW

Name	Bit	Type	Description
SRC_SEL	[2:0]	RW	滤波模块的输入信号选择。 0h：禁止滤波 1h：使能SYNCIN0滤波 2h：使能SYNCIN1滤波 3h：使能SYNCIN2滤波 4h：使能SYNCIN3滤波 5h：使能SYNCIN4滤波 6h：使能SYNCIN5滤波 7h：保留
BLKINV	[4]	RW	窗口使能反转控制。 0h：窗口不反转 1h：窗口反转
ALIGNMD	[6:5]	RW	窗口对齐模式选择。 0h：CNT=PRD 1h：CNT=ZRO 2h：CNT=PRD or CNT=ZRO 3h：T1事件
CROSSMD	[7]	RW	允许滤波窗跨越多个TB的周期。 缺省条件下，当滤波窗在周期结束时若任然有效，将跨过周期点，一直持续到窗口计数器溢出。当禁止跨周期时，在周期结束时，窗口计数器将被停止。 0h：禁止跨周期 1h：允许跨周期

12.4.22 GPTA_TRGFTWR (数字比较器滤波窗时序寄存器)

- 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
WINDOW																OFFSET															
0	0	0	0	0	0	0	0	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
OFFSET	[15:0]	RW	滤波窗的OFFSET设置。 此16bit控制位定义了从窗口参考起始位置开始计数多少个TCLK后，开始有效的滤波窗口。参考位置的定义，在TRGFTCR[ALIGNMD]控制位中进行选择。OFFSET的Shadow寄存器在ALIGNMD指定的条件满足时，载入到Active寄存器中，并重新开始计数。
WINDOW	[31:16]	RW	滤波窗的宽度设置。 此16bit控制位定义了滤波窗的宽度，窗口宽度是基于TCLK的计数值。当OFFSET计数器溢出时，WINDOW计数器被重置，并开始计数直到溢出。当OFFSET计数器溢出，但WINDOW状态已经激活时，WINDOW计数器不会重置。在应用时必须注意此条件的设置。

12.4.23 GPTA_EVTRG (事件触发选择寄存器)

- 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				RSVD		CNT1INITFRC	CNT0INITFRC	RSVD		TRG1OE	TRG0OE	RSVD		CNT1INITEN	CNT0INITEN	RSVD								TRGSEL1				TRGSEL0			
0	0	0	0	0	0	0	0	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	RW	RW	R	R	RW	RW	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
TRGSEL0 TRGSEL1	[3:0] [7:4]	RW	TRGEV0，TRGEV1事件的触发源选择。 0000：禁止TRGOUT触发输出 0001：当 CNT = ZRO 产生TRGEV事件 0010：当 CNT = PRD 产生TRGEV事件 0011：当 CNT = ZRO or CNT = PRD 产生TRGEV事件 0100：当 CNT = CMPA 且计数方向为递增时，产生TRGEV事件 0101：当 CNT = CMPA 且计数方向为递减时，产生TRGEV事件 0110：当 CNT = CMPB 且计数方向为递增时，产生TRGEV事件 0111：当 CNT = CMPB 且计数方向为递减时，产生TRGEV事件 1100：ExtSync通道 1101：PE0 event 1110：PE1 event 1111：PE2 event
CNT0INITEN	[16]	RW	TRGEV0CNT寄存器更新模式控制 0h：无效 1h：TRGEV0CNT在发生LOAD事件触发时，或者EV0CNTINITFRC控制位软件写入‘1’时，EV0CNTINIT的内容更新到EV0CNT中。
CNT1INITEN	[17]	RW	TRGEV1CNT寄存器更新模式控制 0h：无效 1h：TRGEV1CNT在发生LOAD事件触发时，或者EV1CNTINITFRC控制位软件写入‘1’时，EV1CNTINIT的内容更新到EV1CNT中。
TRG0OE	[20]	RW	外部触发端口TRGOUT0使能 0h：禁止触发输出

			1h：允许触发输出
TRG1OE	[21]	RW	外部触发端口TRGOUT1使能 0h：禁止触发输出 1h：允许触发输出
CNT0INITFRC	[24]	RW	TRGEV0CNT软件触发更新 0h：无效 1h：EVCNT0INIT内容更新到EVCNT0中
CNT1INITFRC	[25]	RW	TRGEV1CNT软件触发更新 0h：无效 1h：EVCNT1INIT内容更新到EVCNT1中

12.4.24 GPTA_EVPS (事件触发计数寄存器)

- 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
								TRGEV1CNT				TRGEV0CNT												TRGEV1PRD				TRGEV0PRD			
0	0	0	0	0	0	0	0	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
TRGEV0PRD	[3:0]	RW	TRGEV0事件计数的周期设置。 当TRGEV0事件发生次数满足周期时，才产生TRGEV0触发事件
TRGEV1PRD	[7:4]	RW	TRGEV1事件计数的周期设置。 当TRGEV1事件发生次数满足周期时，才产生TRGEV1触发事件
TRGEV0CNT	[19:16]	RW	TRGEV0事件计数器设置。 读取时，返回当前事件计数器值； 写入时，直接更新事件计数器值。
TRGEV1CNT	[23:20]	RW	TRGEV1事件计数器设置。 读取时，返回当前事件计数器值； 写入时，直接更新事件计数器值。

12.4.25 GPTA_EVCNTINIT (事件触发计数器初始化值寄存器)

- Address = Base Address +错误!未定义书签。, 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
																								CNT1INIT				CNT0INIT			
0	0	0	0	0	0	0	0	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
CNT0INIT	[3:0]	RW	TRGEV0CNT计数器的初始化值设置。 当EVTRG[CNT0INITEN]控制位有效时，CNT0INIT的值将在触发条件满足时(LOAD事件)，或EVTRG[CNT0INITFRC]软件置位时，被载入到TRGEV0CNT寄存器中。
CNT1INIT	[7:4]	RW	TRGEV1CNT计数器的初始化值设置。 当EVTRG[CNT1INITEN]控制位有效时，CNT1INIT的值将在触发条件满足时(LOAD事件)，或EVTRG[CNT1INITFRC]软件置位时，被载入到TRGEV1CNT寄存器中。

12.4.26 GPTA_EVSWF (事件计数器载入控制寄存器)

- 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																														EV1SWF	EV0SWF
0	0	0	0	0	0	0	0	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
EV0SWF	[0]	RW	软件产生一次EV0的触发 0h：写入‘0’无效 1h：软件产生一次触发
EV1SWF	[1]	RW	软件产生一次EV1的触发 0h：写入‘0’无效 1h：软件产生一次触发

12.4.27 GPTA_RISR (原始中断状态寄存器)

- Address = Base Address + 0x00D0, 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															PEND	RSVD				CBD	CBU	CAD	CAU	RSVD		CAP_LD1	CAP_LD0	RSVD		TRGEV1	TRGEV0
0	0	0	0	0	0	0	0	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
TRGEV0	[0]	W	TRGEV0中断请求原始标志状态
TRGEV1	[1]	W	TRGEV1中断请求原始标志状态
CAP_LD0	[4]	R	Capture Load to C1 Shadow中断请求原始标志状态
CAP_LD1	[5]	R	Capture Load to C2 Shadow中断请求原始标志状态
CAU	[8]	R	递增阶段CNT = CMPA中断请求原始标志状态
CAD	[9]	R	递减阶段CNT = CMPA中断请求原始标志状态
CBU	[10]	R	递增阶段CNT = CMPB中断请求原始标志状态
CBD	[11]	R	递减阶段CNT = CMPB中断请求原始标志状态
PEND	[16]	R	周期结束中断请求原始标志状态
原始中断标志表示中断事件发生，通过设置IMCR中相应位，可以允许该中断请求CPU中断。原始中断标志位需要通过软件清除。 0h：该中断未置位 1h：该中断已置位			

12.4.28 GPTA_MISR (中断状态寄存器)

- Address = Base Address + 0x00D4, 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															PEND	RSVD				CBD	CBU	CAD	CAU	RSVD		CAP_LD1	CAP_LD0	RSVD		TRGEV1	TRGEV0
0	0	0	0	0	0	0	0	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
TRGEV0	[0]	W	TRGEV0中断请求标志状态
TRGEV1	[1]	W	TRGEV1中断请求标志状态
CAP_LD0	[4]	R	Capture Load to C1 Shadow中断请求原始标志状态
CAP_LD1	[5]	R	Capture Load to C2 Shadow中断请求原始标志状态
CAU	[8]	R	递增阶段CNT = CMPA中断请求原始标志状态
CAD	[9]	R	递减阶段CNT = CMPA中断请求原始标志状态
CBU	[10]	R	递增阶段CNT = CMPB中断请求原始标志状态
CBD	[11]	R	递减阶段CNT = CMPB中断请求原始标志状态
PEND	[16]	R	周期结束中断请求原始标志状态
中断标志表示CPU中断请求的状态，通过写ICR寄存器可以清除该标志位。 0h：该中断未置位 1h：该中断已置位			

12.4.29 GPTA_IMCR (中断使能控制寄存器)

- Address = Base Address + 0x00D8, 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															PEND	RSVD				CBD	CBU	CAD	CAU	RSVD		CAP_LD1	CAP_LD0	RSVD		TRGEV1	TRGEV0
0	0	0	0	0	0	0	0	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	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW

Name	Bit	Type	Description
TRGEV0	[0]	W	TRGEV0中断使能控制位
TRGEV1	[1]	W	TRGEV1中断使能控制位
CAP_LD0	[4]	R	Capture Load to C1 Shadow中断请求原始标志状态
CAP_LD1	[5]	R	Capture Load to C2 Shadow中断请求原始标志状态
CAU	[8]	R	递增阶段CNT = CMPA中断请求原始标志状态
CAD	[9]	R	递减阶段CNT = CMPA中断请求原始标志状态
CBU	[10]	R	递增阶段CNT = CMPB中断请求原始标志状态
CBD	[11]	R	递减阶段CNT = CMPB中断请求原始标志状态
PEND	[16]	R	周期结束中断请求原始标志状态
CPU中断请求使能控制。当该控制位使能时，允许触发CPU中断。 0h：禁止该中断 1h：允许该中断			

12.4.30 GPTA_ICR (中断清除寄存器)

- Address = Base Address + 0x00DC, 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															PEND	RSVD				CBD	CBU	CAD	CAU	RSVD		CAP_LD1	CAP_LD0	RSVD		TRGEV1	TRGEV0
0	0	0	0	0	0	0	0	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	W	R	R	R	W	W	W	W	W	W	W	W	W	R	R	W	W

Name	Bit	Type	Description
TRGEV0	[0]	W	清除TRGEV0原始中断状态位
TRGEV1	[1]	W	清除TRGEV1原始中断状态位
CAP_LD0	[4]	R	Capture Load to C1 Shadow中断请求原始标志状态
CAP_LD1	[5]	R	Capture Load to C2 Shadow中断请求原始标志状态
CAU	[8]	R	递增阶段CNT = CMPA中断请求原始标志状态
CAD	[9]	R	递减阶段CNT = CMPA中断请求原始标志状态
CBU	[10]	R	递增阶段CNT = CMPB中断请求原始标志状态
CBD	[11]	R	递减阶段CNT = CMPB中断请求原始标志状态
PEND	[16]	R	周期结束中断请求原始标志状态
中断清除控制位。 对该寄存器写‘0’时，无效；对该寄存器写‘1’时，清除相应中断标志位 读取时，总是返回‘0’			

13 增强型通用定时器 (EPT)

13.1 概述

增强型通用定时器（Enhanced Purpose Timer）作为 MCU 的关键外设，可以在各种功率控制应用中发挥关键控制作用。通过灵活的 PWM 输出，可以适用于各种复杂多变的应用，这些应用包括数字马达控制、开关电源控制、不间断电源控制、变频功率转换控制等。EPT 内部包含一个 16 位的定时/计数模块，支持 2 种工作模式(捕捉模式和波形发生器模式)。此 EPT 为 TYPE-A 类型。

13.1.1 主要特性

- 16 位可复位计数器
- 可编程计数器计数方式
  - 递增计数（Up-counting）
  - 递减计数（Down-counting）
  - 递增递减计数（Up-down-counting）
- 7 路 PWM 输出，包括 4 路波形产生控制单元，支持 4 路独立输出或者 3 组互补输出：
  - 4 路独立的 PWM 输出，单边沿工作
  - 4 路独立的 PWM 输出，双边沿对称工作
  - 3 组独立的 PWM 互补输出 + 1 路独立的 PWM 输出
- 可编程的死区控制单元
- 通过软件异步重置 PWM 的波形输出
- 支持可编程的相位控制
- 异常情况处理控制单元
  - 异常事件发生时，自动触发预设波形输出
  - 多种触发方式，包括外部管脚和模拟比较器（如含有）
- 支持片间多设备同步
  - 支持多个 TIMER 间的同步触发
  - 触发源包括 GPIO 输入，其他外设触发，软件设置和事件触发
  - 支持单次触发和连续触发模式
- 支持单脉冲输出模式
- 支持突发计数模式

- 支持通过外部时钟计数
- 支持事件计数器，可通过配置事件计数器（最大 15）触发相应中断
- 支持 PWM 对更高载波频率进行斩波输出
- 支持捕获模式，最多支持 4 个捕获值存储。

13.1.2 管脚描述

下表列出了不同模式下的管脚定义。

Table 14-1 不同模式下的管脚描述

管脚名称	突发计数模式	波形发生器： 单波形输出模式	波形发生器： 双波形输出模式
CHAX	时钟控制使能	输出波形	输出波形
CHAY	NA	输出波形	输出波形
CHBX	时钟控制使能	输出波形	输出波形
CHBY	NA	输出波形	输出波形
CHCX	NA	输出波形	输出波形
CHCY	NA	输出波形	输出波形
CHD	NA	输出波形	输出波形

13.2 功能描述

13.2.1 模块框图

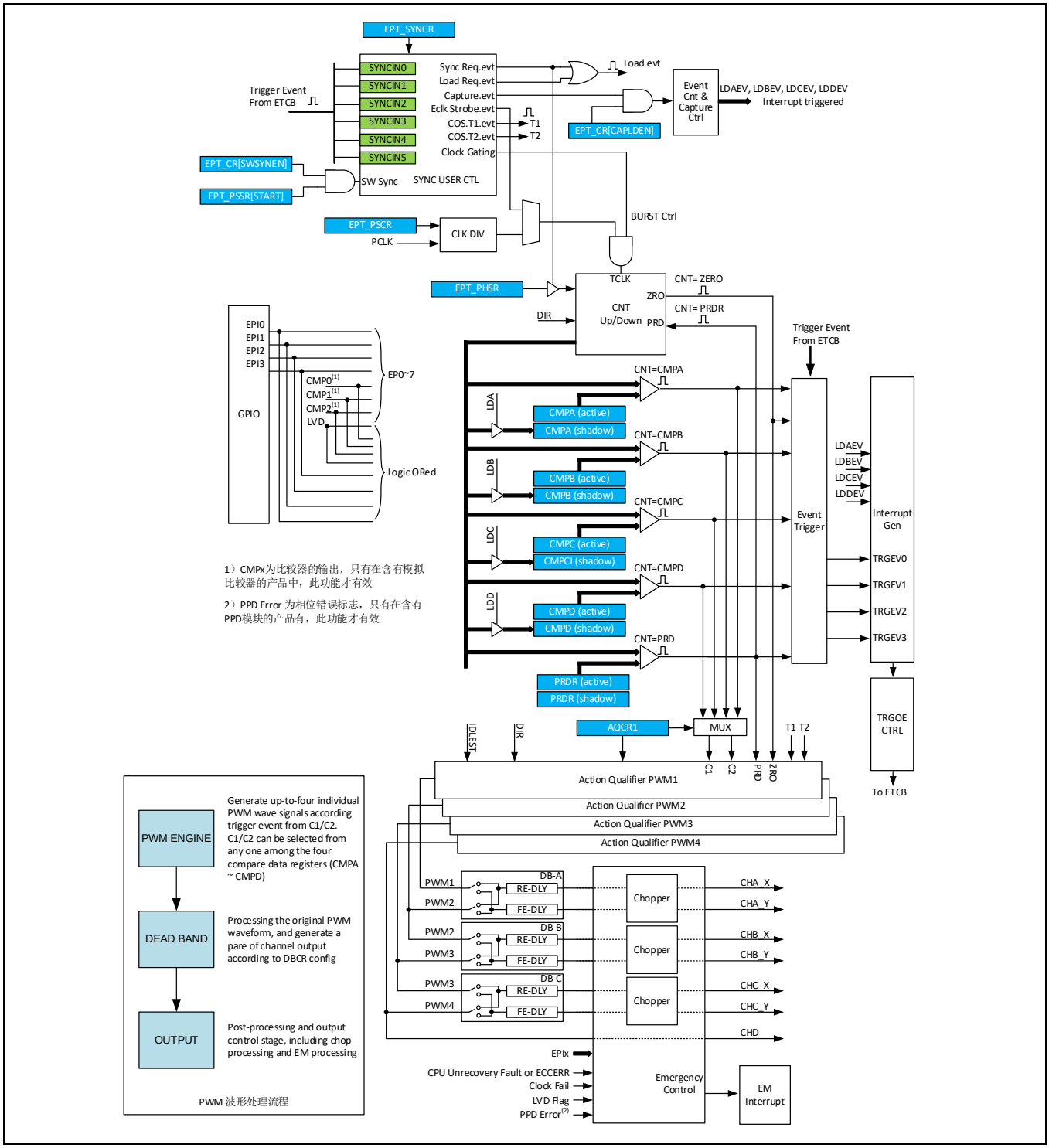


Figure 14-1 模块结构示意图

13.3 基本功能描述

一个完整的EPT模块包含7个TIMER输出通道。多个EPT或GPT和其他外设间可以通过ETCB连接，通过ETCB链，实现多个EPT和其他外设的同步工作。在包含多个EPT的器件中，以数字后缀区分不同的实例，例如：EPT0代表第一个EPT模块，EPT1代表第二个EPT模块。每个EPT中根据功能划分，可以分为几个模块，包括时钟控制模块、时基模块（计数器）、计数器比较模块、动作限定模块、死区控制模块、斩波模块、捕捉控制模块、事件触发模块、紧急处理模块和同步触发控制模块。

CHAX/CHAY、CHBX/CHBY、CHCX/CHCY、CHD是EPT在GPIO上映射的输出端口，其中CHAX和CHBX支持输入功能（外部时钟使能控制）。在波形输出模式下，这7个端口作为PWM信号的输出端口，在群脉冲模式下（CR[BURST]使能），CHAX或者CHBX可以作为门控时钟的时钟控制输入信号。EBIx为外部GPIO上映射的异常输入功能，可以作为紧急状态处理的触发信号。

EPT内部PWM引擎具有4个独立驱动模块，每个模块中有C1和C2两个数字比较器。通过C1和C2，配合时基计数器，PWM引擎可以产生4路独立的同周期PWM信号。4路PWM信号通过信号选择器接入后续的三个独立死区处理模块，在每个死区处理模块中，可以对输入信号进行极性反转，上升沿和下降沿的延时处理。每个死区控制模块以信号对的方式将处理后的两路波形信号送入最后的输出控制级。在输出控制级，可以控制最终输出到PAD上的波形信号，包括斩波处理和关断方式处理。

13.3.1 时钟源

13.3.1.1 概述

增强型通用定时器EPT工作在PCLK下。计数器计数时钟TCLK可以通过选择PCLK的分频后输出，或者由外部提供。当计数器选择为外部计数器时钟控制下计数时，必须使能EPT的同步触发端口SYNCIN3。计数器在外部计数模式下，只有在SYNCIN3被触发时，才会对计数器值进行一次增减操作（SYNCR[SYNCEN3]控制位）。SYNCIN3的触发源可以通过ETCB配置为多种外设，包括GPIO或者其他TIMER。

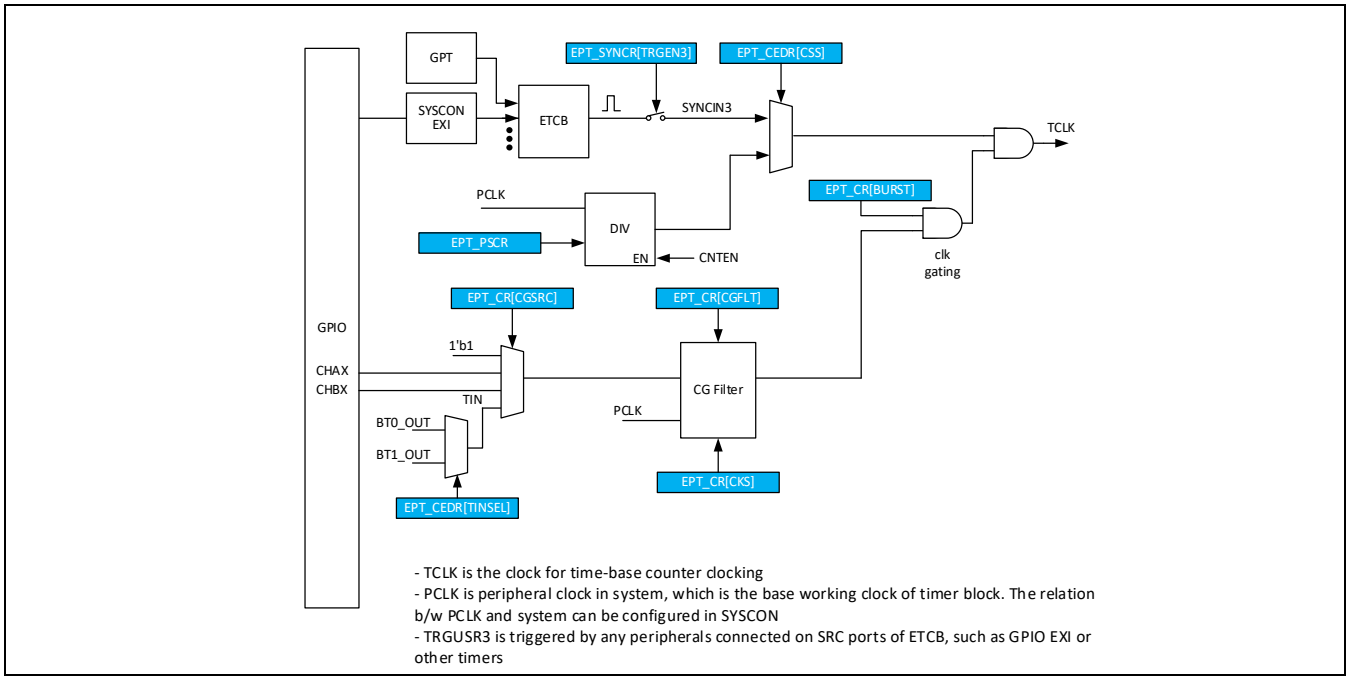


Figure 14-2 时钟控制模块

13.3.1.2 外部时钟

当使用外部GPIO作为外部时钟的输入时，通道选择和极性控制，通过SYSCON内的触发控制进行选择。具体参考SYSCON章节。

13.3.1.3 内部时钟

当PCLK作为计数器的计数时钟时，可以通过一个16位的预分频器对PCLK进行分频而产生计数用的TCLK。预分频可以通过PSCR进行设置。在对PSCR进行读写时，操作的对象为PSCR的影子寄存器（Shadow Register）；当时基计数器的值等于零时，影子寄存器的值将被更新到内部的活动寄存器中（Active Register）。当对PSCR更新后，新的分频将在下一个计数周期开始时有效。

13.3.1.4 群脉冲时钟

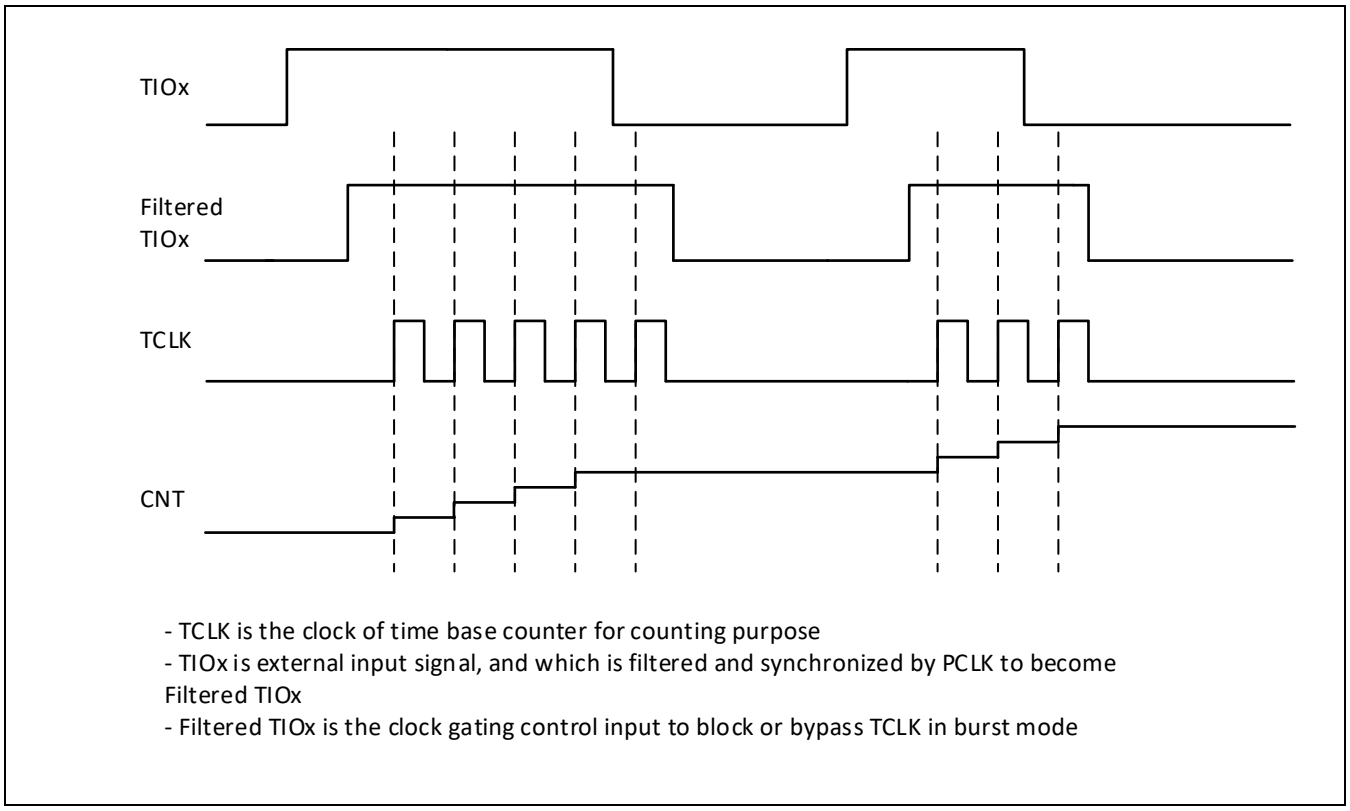


Figure 14-3 群脉冲时钟模式时序

在群脉冲时钟模式下(CR[BURST]=1)，计数器的计数时钟将会和相应的控制信号进行与操作。计数器只有在被选择的使能信号有效时，才进行计数。门控时钟使能信号可以通过CR[CGSRC]控制位进行选择，支持CHAX通道或者CHBX的外部输入作为门控信号，或者由TIN的输入信号来控制。当CHAX或CHBX选择为门控时钟时，该通道自动设置为输入状态，并禁止该通道的波形输出。TIN由BT0的输出或者BT1的输出决定，TIN的信号源可以通过CEDR[TINSEL]控制位进行选择。

在CG(clock gating) 输入通道上，可以通过设置CR[CGFLT]使能数字滤波。数字滤波在连续检测到N个一致结果时，才会确认输出翻转，否则将保证当前输出状态，如下图所示。数字滤波器的设置包括滤波器的时钟源选择，滤波时钟频率以及滤波深度。滤波时钟可以通过CEDR[FLTCKPRS]控制位进行设置。滤波工作时钟频率通过CEDR[FLTCKPRS]控制位进行设置。滤波深度通过CR[CGFLT]进行设置。滤波器的延时通过如下公式进行计算：$T_{dly} = T_{fltclk}$ x CR[CGFLT] x CEDR[FLTCKPRS]，其中T_{fltclk}为滤波器工作时钟的周期。

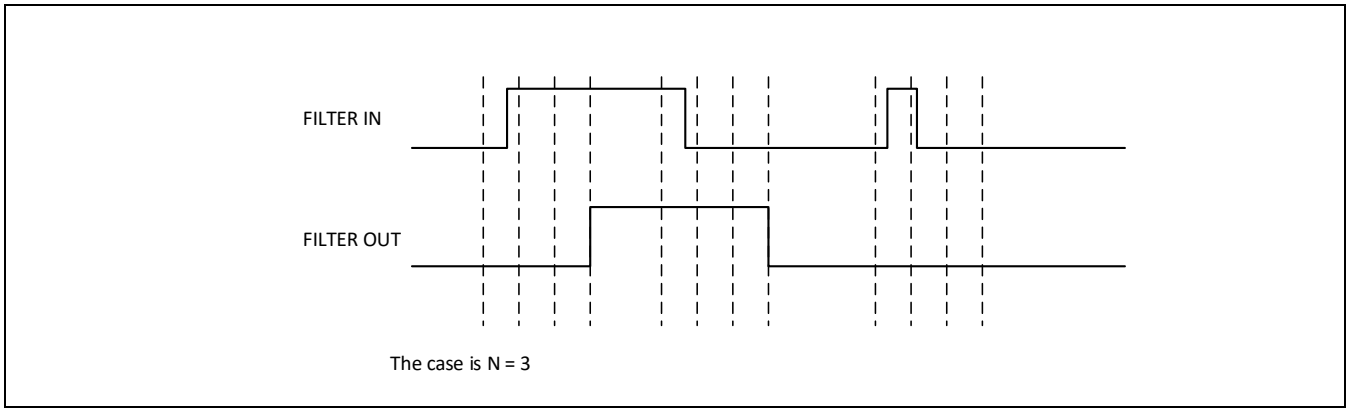


Figure 14-4 CG Filter 数字滤波器的原理

13.3.2 时基控制

13.3.2.1 概述

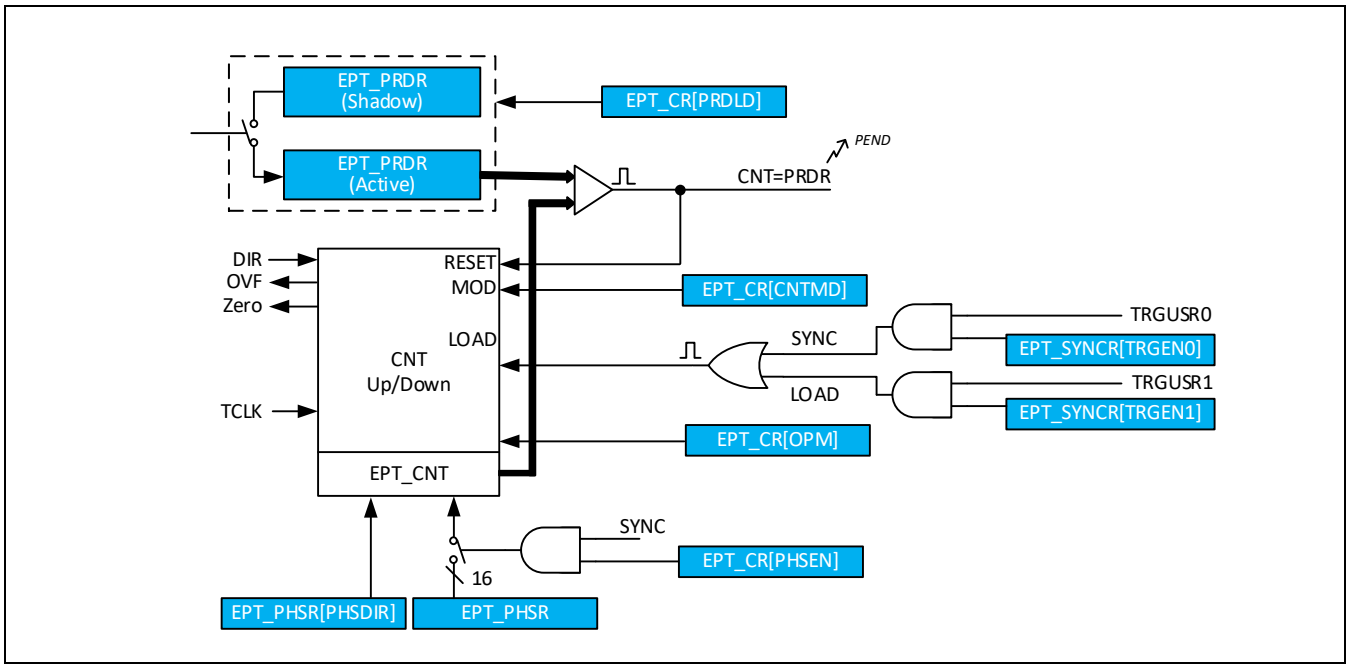


Figure 14-5 计数器时基模块

作为EPT主要的控制模块，时基控制模块由一个16位的计数器和相应的自动重载寄存器组成。模块的主要功能有：

- 确定时基计数器（CNT）的频率，或者控制事件触发的周期。
- 管理和其他模块间的同步
- 控制和其他EPT模块间的相位关系
- 设置计数器工作模式
- 根据计数器值产生不同的触发事件

时基模块的寄存器包括：

- 计数器寄存器（CNT）：在每个计数时钟周期根据计数模式增加或者减少
- 相位寄存器（PHSR）： CR[PHSEN]使能时，计数器将在SYNC触发时被自动重置为相位寄存器的设置值。
- 周期寄存器（PRDR）：计数器周期控制寄存器。

计数器的计数周期由周期寄存器（PRDR）的设置值以及计数器的计数模式（CR[CNTMD]）共同决定。计数器支持三种计数模式：

- 递增模式（Up-Counting Mode）：

  在递增模式下，时基计数器从0x0000开始递增计数，一直计数到周期设置值（PRDR）。当计数值等于周期设置值时，时基计数器被复位，重新开始从0x0000进行新一轮计数。

- 递减模式：（Down-Counting Mode）

  在递减模式下，时基计数器从周期设置值开始递减计数，一直计数到0x0000。当计数器值等于0x0000时，时基计数器被重置为周期设置值并开始新一轮计数。

- 递增递减模式：（Up-Down-Counting Mode）

  在递增递减模式下，时基计数器从0x0000开始递增计数，一直计数到周期设置值（PRDR），然后开始递减计数，一直计数到0x0000。当计数器值等于0x0000时，重新开始新一轮计数。

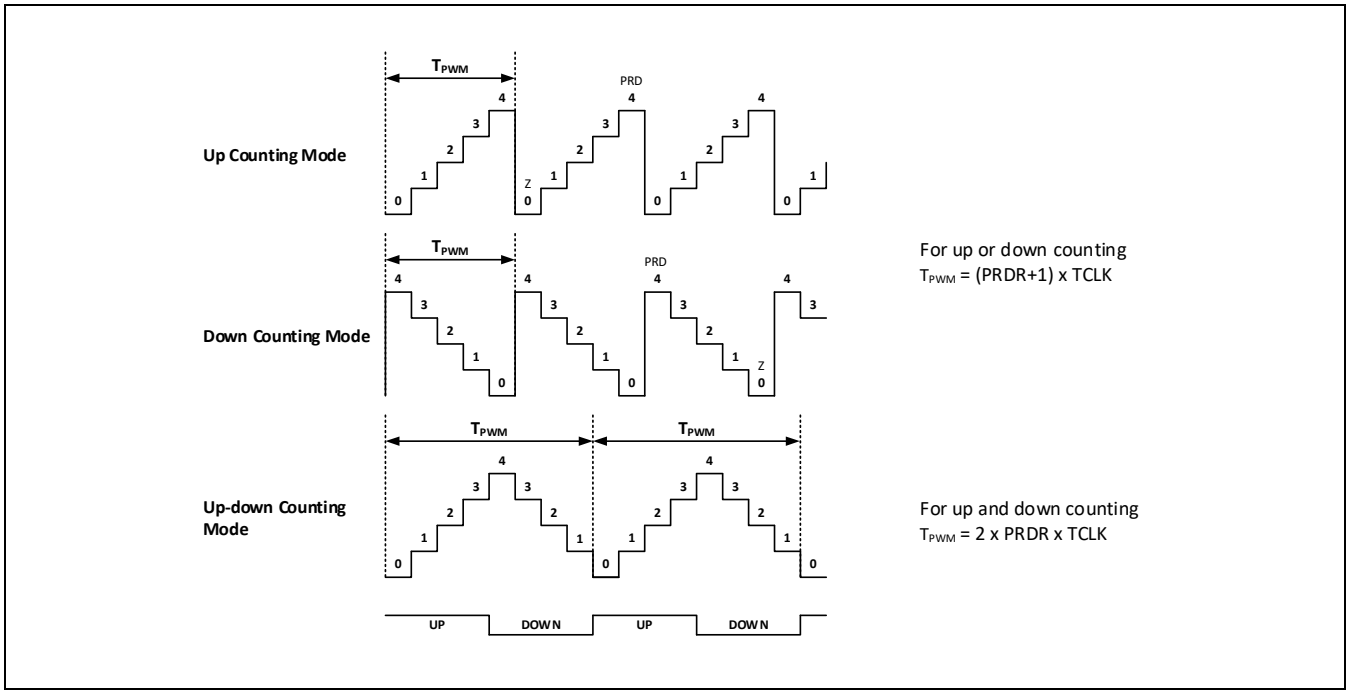


Figure 14-6 计数器工作模式

13.3.2.2 计数器重置和周期设置

PRDR寄存器控制计数的周期，周期时间为（PRDR+1）x TCLK。在下列条件满足时，计数器值将会被重置。重置发生时，根据当前计数模式，计数器将被重置为三种预设值中的一种：0x0000，PHSR的设置值或者是PRDR的设置值。

- 同步事件触发（SYNC触发）：当同步事件发生时，可以配置计数器重置。当CR[PHSEN]控制位使能时，计数器将被重置到PHSR所设置的值，否则计时器将根据当前设置的计数模式被初始化为0或PRDR的设置值。
- 通过RSSR[START]控制位启动计数器计数时：当计数模式设置为递增或递增递减模式时，计数器被初始化为0，当计数模式设置为递减模式时，计数器被初始化为PRDR所设置的数值。
- 软件直接更新（对CNT直接写入）：通过软件直接写入计数器活动寄存器进行更新。在计数器开始计数前通过软件直接更新的值，会被计数器启动初始化时新的配置值所替代，而在计数器已经启动后，进行的更新操作，更新值将直接被载入到当前计数器中。

PRDR周期寄存器由两个物理寄存器组成：活动寄存器（Active）和影子寄存器（Shadow）。影子寄存器的值通过硬件在特定条件满足时自动同步到活动寄存器中，以保证对活动寄存器更新操作和计数器计数周期同步。活动寄存器直接参与计数器控制事件的产生；影子寄存器作为数据缓冲，为活动寄存器提供临时的数据保存。影子寄存器值不会直接影响硬件控制动作，而是根据预设策略，在特定时间将缓冲的内容传送到活动寄存器中。这样的机制，避免了由于软件触发的寄存器更新操作和计数器当前工作状态非同步，而引起的硬件输出错误。活动寄存器和影子寄存器共享同一个物理访问地址，当前读写操作的对象是活动寄存器还是影子寄存器，可以通过CR[PRDLD]控制位进行选择。当影子寄存器被屏蔽时，对PRDR的写入值，会直接改变活动寄存器的值，而对PRDR读取时，将直接返回活动寄存器的值。

- **PRDR寄存器的Shadow模式**

  PRDR的缓冲（Shadow Register）在CR[PRDLD]控制位不等于'11b'的时候有效。在此配置下，CPU对PRDR的读写操作对象为PRDR的影子寄存器。当时基计数器值等于零时，或者SYNC触发时，影子寄存器的值被硬件自动载入到活动寄存器（Active Register）中。在缺省配置下，只有时基计数器值等于零时，自动载入才会发生，用户可以通过配置CR[PRDLD]控制位进行修改。

- **PRDR寄存器的立即加载模式**

  在立即加载模式下（CR[PRDLD]=3），CPU对PRDR的读写取操作对象是PRDR的活动寄存器。任何对PRDR的更新操作将被直接反应到活动寄存器中。

  在对PRDR进行立即更新时，需要注意考虑当前的计数器值。若将PRDR更新到一个比当前计数器值小的值，将导致计数器在后续计数过程中都不会发生PERIOD事件，计数器将一直计数到整个计数器溢出后重新开始计数。

13.3.2.3 计数模式和时序

时基计数器可以分为四种工作模式：

- 递增计数模式（非对称）

- 递减计数模式（非对称）
- 递增递减模式（对称）
- 冻结模式，在此模式下计数器保持当前计数值

在下面的图示中说明了上述前三种工作模式下，时基计数器根据触发条件如何工作和产生相应事件。

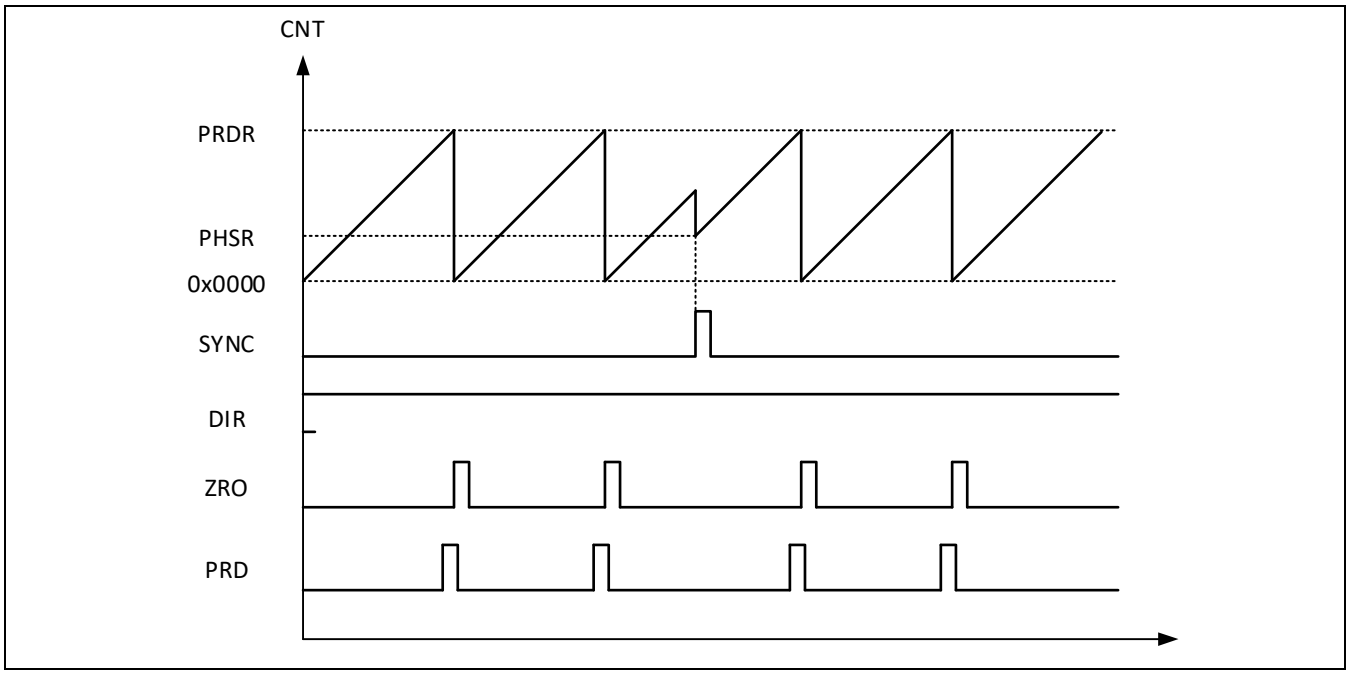


Figure 14-7 递增工作模式

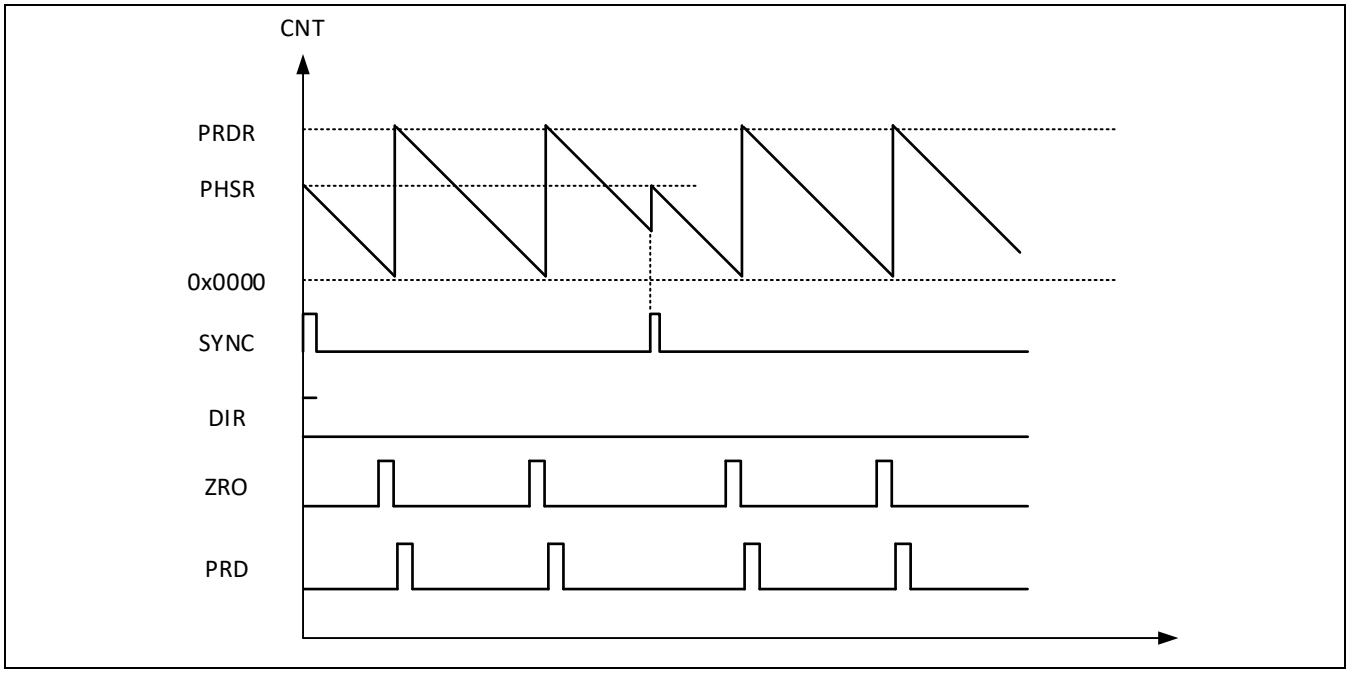


Figure 14-8 递减工作模式

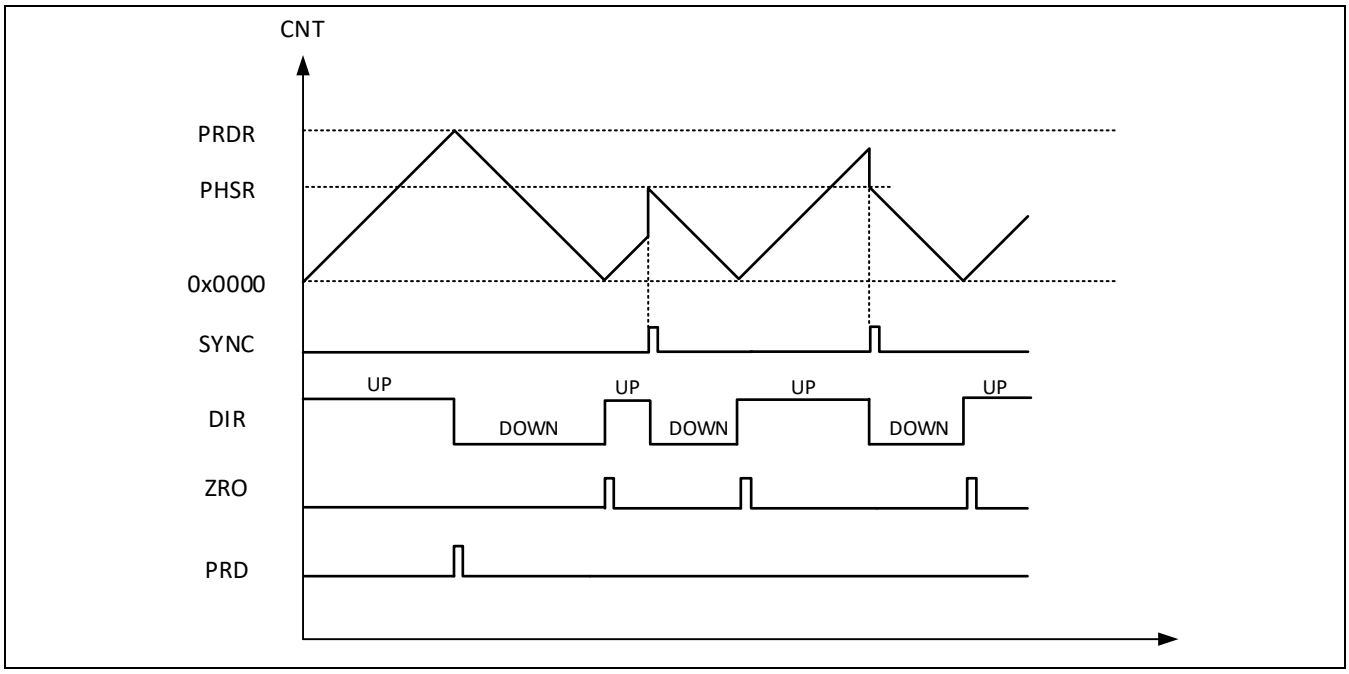


Figure 14-9 递增递减工作模式

13.3.2.4 全局载入控制

EPT中，很多寄存器具有影子寄存器功能。每个影子寄存器只有在特定条件满足时，才会更新到活动寄存器中。每个影子寄存器对活动寄存器的更新条件均可以独立设置。

如果EPT内大多数寄存器都可以共用相同的载入条件，可以使用全局载入控制。当全局载入使能时，所有影子寄存器对活动寄存器的更新都将受全局载入条件控制，并在全局载入条件满足时，将全部影子寄存器的当前值更新到对应的活动寄存器中。全局载入使能通过GLDCR[GLDEN]设置。

即便全局载入使能时，也可以通过GLDCFG寄存器配置每个影子寄存器是否受全局载入控制，用户可以通过配置GLDCFG，选择不需要受全局载入控制的影子寄存器。设置为不受全局载入控制的寄存器，仍将使用自己独立的载入控制配置。例如：当GLDEN=1, 且GLDCFG[CMPA]=1，GLDCFG[CMPB]=0时，则CMPA的影子寄存器将在全局载入条件满足时，更新到活动寄存器中；CMPB的影子寄存器的更新条件不受全局载入条件控制，仍旧按照CMPLDR[LDBMD]的设置进行更新。

全局载入控制支持事件计数，只有当选中的触发事件在第N次发生时，才会进行全局载入，N为事件计数器的设置值。可以通过寄存器GLDCR[GLDPRD]控制位设置事件计数器值，当前已经发生的触发次数可以通过GLDCR[GLDCNT]控制位进行查询。

全局载入可以被连续触发，或者只允许发生一次。当One-shot 载入模式使能时（GLDCR[OSMD]=1），全局条件满足时，触发一次全局载入，后续的载入将被屏蔽。必须通过软件重新初始化后，才能进行新的触发。

通过设置GLDCR2[GFRCLD[控制位，软件可以强制触发全局载入。

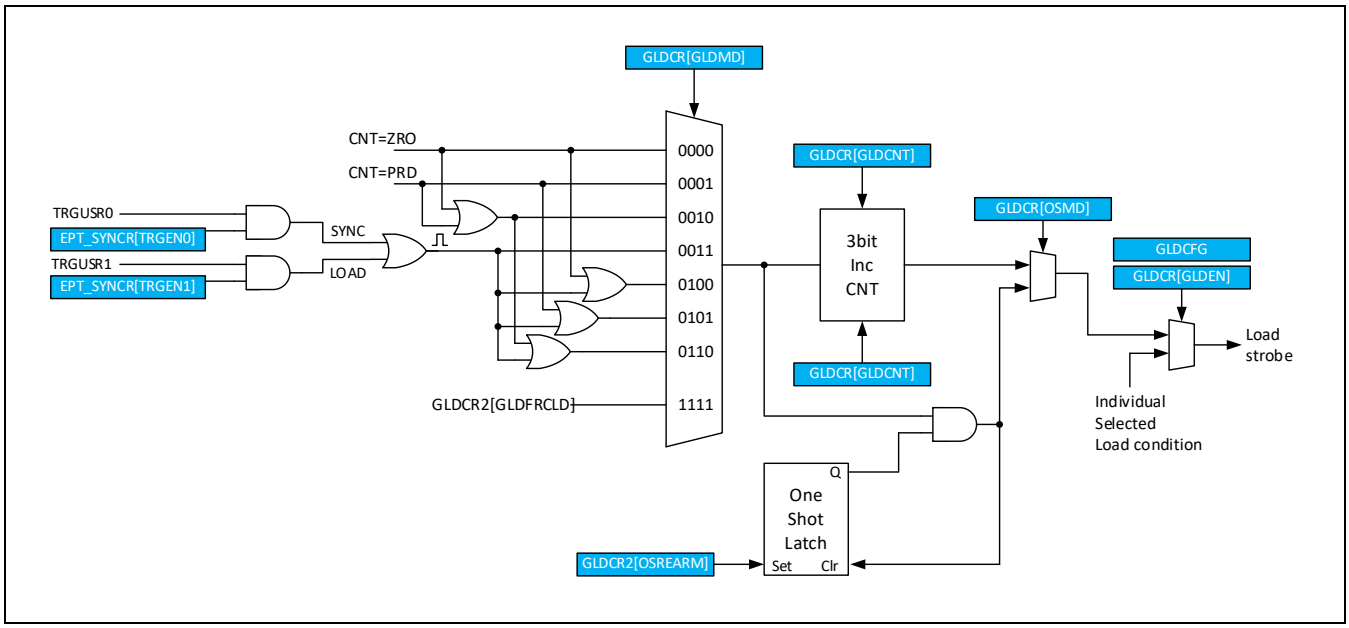


Figure 14-10 全局载入控制

13.3.3 计数器数值比较控制

13.3.3.1 概述

计数器数值比较控制比较控制模块实时比较当前计数器的计数值和比较值寄存器（CMPA、CMPB、CMPC和CMPD）的值，当计数值等于其中任意一个比较值时，比较控制模块将产生一个相应的事件触发。主要特性如下：

- 支持的触发事件和触发条件如下：
  - CNT = CMPA：时基计数器当前值等于计数器比较值A寄存器的值
  - CNT = CMPB：时基计数器当前值等于计数器比较值B寄存器的值
  - CNT = CMPC：时基计数器当前值等于计数器比较值C寄存器的值
  - CNT = CMPD：时基计数器当前值等于计数器比较值D寄存器的值
- PWM的波形控制基于CMPA、CMPB、CMPC和CMPD
- 比较值寄存器具有影子寄存器功能，以防止PWM输出产生毛刺

计数值比较模块不断监测当前时基计数器的计数值，当计数值等于四个比较值中的任意一个时，都会触发独立的比较事件。4个比较事件都可以用于触发中断或者同步。比较值寄存器还用于PWM引擎，控制PWM波形产生。每个PWM引擎通道具有两个数字比较器(C1 and C2)用于控制波形输出，这两个数字比较器的参考值可以通过AQCRx[C1SEL]和AQCRx [C2SEL]控制位选择CMPA到CMPD中的任意一个作为输入。PWM数字引擎中C1和C2产生的触发信号只用于波形控制，不能直接产生中断或者同步触发。

在递增模式或者递减模式下，每个比较事件在一个计数周期内只会发生一次。在递增递减模式下，如果比较值设置为0到PRD之间时，每个事件在一个计数周期内会发生两次；而如果比较值设置为0或者PRD时，每个事件在一

个计数周期内只发生一次。C1和C2这两个触发事件以及和来自于时基模块的当前计数方向信号，在波形发生模块中共同决定了输出波形的跳转时间点。

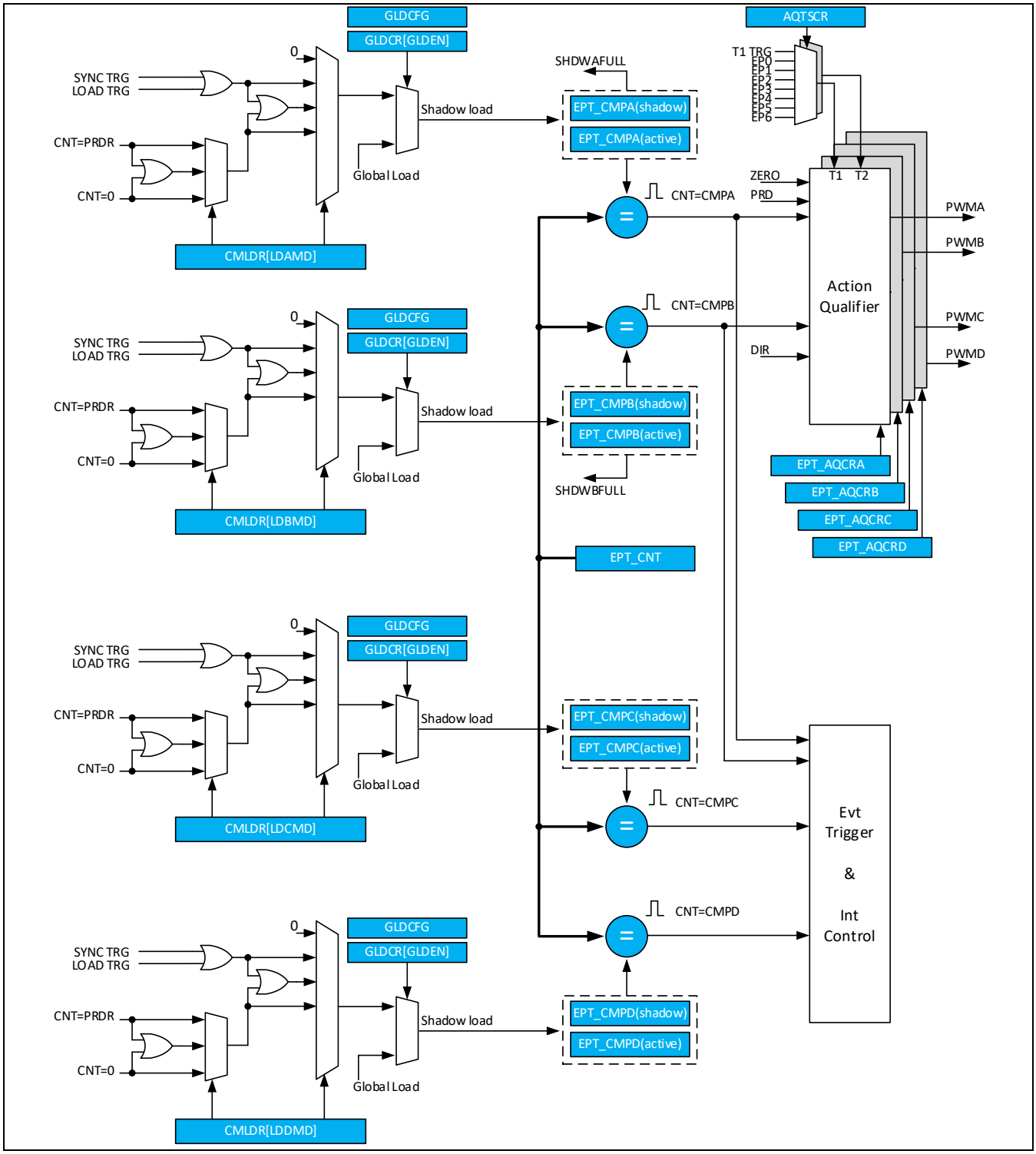


Figure 14-11 计数器值比较控制

13.3.3.2 比较值寄存器载入方式

CMPA、CMPB、CMPC和CMPD都有相应的Shadow寄存器，在缺省设置下，所有对CMPx寄存器的读写对象都是影子寄存器。Shadow load的时间可以通过相应CMPLDR[LDxMD]控制位进行设置。影子寄存器的使能可以通过CMPLDR[SHDWCMPx]控制位进行设置。当Shadow模式被禁止时，所有对CMPx寄存器的操作将直接作用到内部活动寄存器上。

- **CMPx寄存器的Shadow模式**

  当Shadow模式使能时，Shadow寄存器中的内容将在下列事件触发时，被自动传送到活动寄存中。可以通过CMPLDR[LDxMD]控制位选择触发CMPx活动寄存器进行更新的事件。下列任意一种组合都可以被作为更新寄存器的触发条件。

  - CNT = ZRO时，触发更新
  - CNT = PRD时，触发更新
  - CNT = PRD或者CNT = ZRO时，触发更新
  - 外部事件（LOAD触发或SYNC触发）触发更新
  - 外部事件（LOAD触发或SYNC触发）或上述任意CNT MATCH事件触发更新

- **CMPx寄存器的立即加载模式**

  在立即加载模式下，对CMPx的操作直接影响活动寄存器。

当全局载入使能，且相应的CMPx在全局载入控制中被选择，全局载入模式的设置将覆盖CMPLDR中的载入方式设置。全局载入的设置具体参照**全局载入控制**章节。

在不同计数模式下，应选择合适的寄存器载入方式以保证波形在下一个周期可以正确输出。选择活动寄存器载入的方式和计数模式的相应关系设置如下：

- 递增模式下，活动寄存器的载入应设置为CNT=ZRO时触发
- CNT = ZRO时，触发更新
- CNT = ZRO时，触发更新

13.3.3.3 不同计数模式的时序

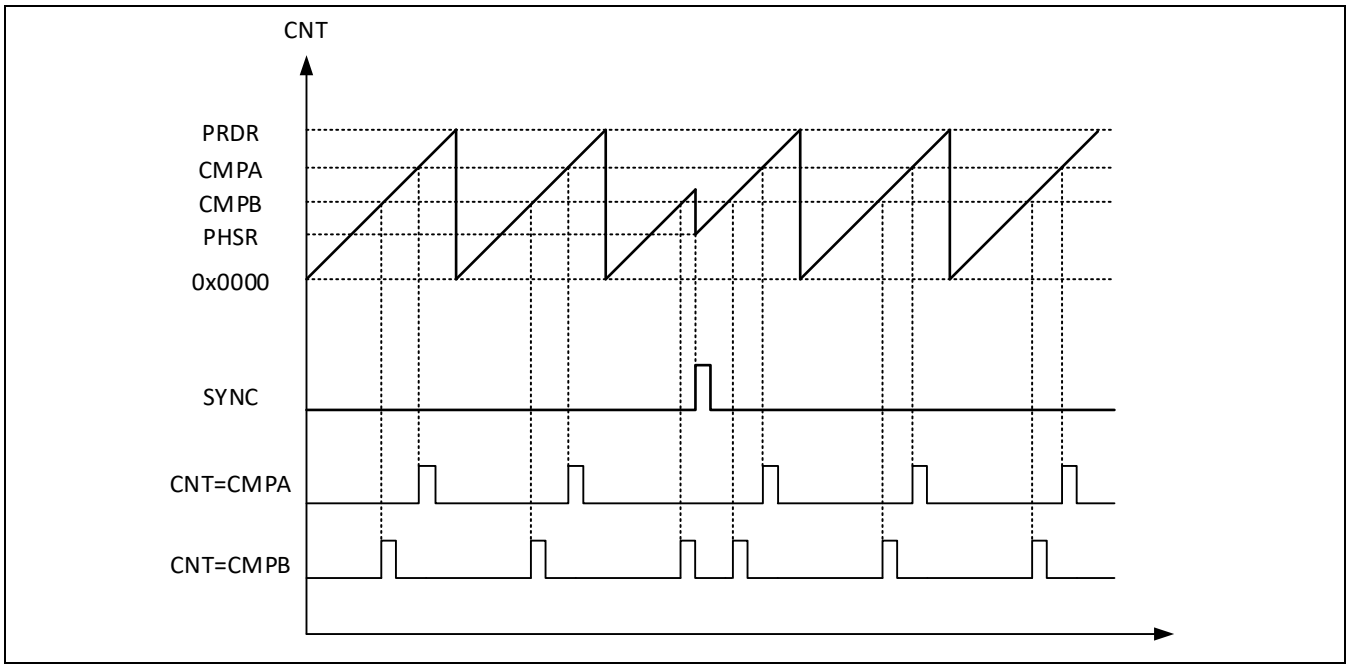


Figure 14-12 递增模式下比较事件产生时序

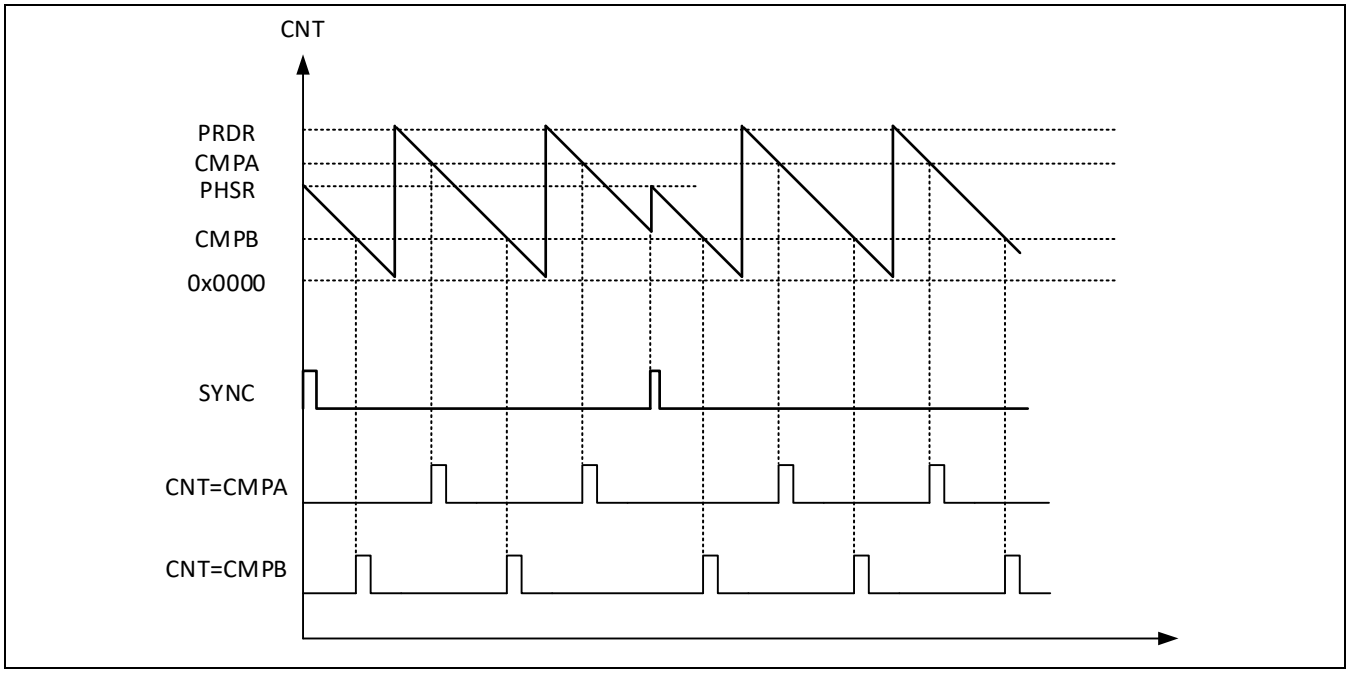


Figure 14-13 递减模式下比较事件产生时序

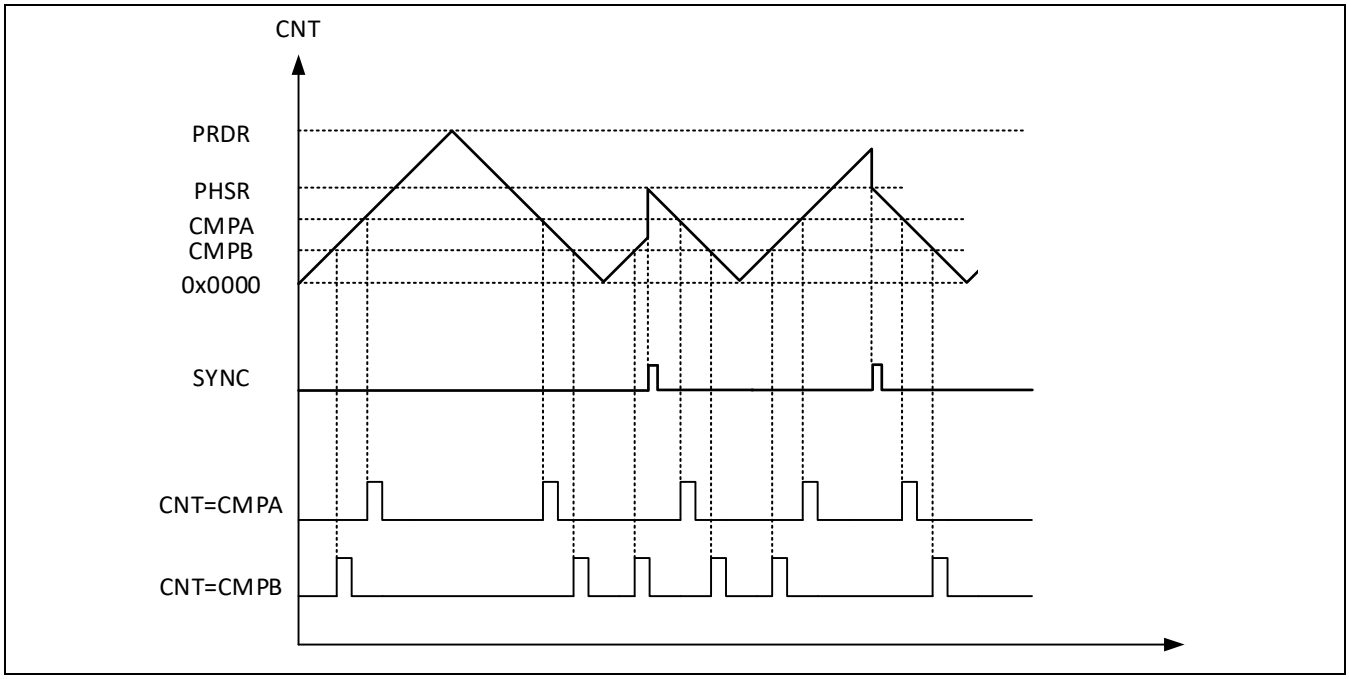


Figure 14-14 递增递减模式下比较事件产生时序

13.3.4 波形发生控制 （Action Qualifier）

13.3.4.1 事件驱动的波形输出

EPT中的PWM信号由PWM硬件引擎产生。硬件引擎支持4路独立的波形输出通路（PWM1、PWM2、PWM3和PWM4，需要注意，此处的PWMx信号为内部信号，并非最终PAD上的驱动信号），在每个通道包含两个数字比较器（C1和C2），C1和C2的比较结果结合当前计数方向可以产生四种触发事件，除此之外，外部触发事件T1、T2以及计数器当前状态等于零或者等于周期值时也会产生触发事件。PWM的波形产生基于不同事件的驱动，通过控制寄存器AQCR1、AQCR2、AQCR3和AQCR4的设置，可以独立映射各种事件触发每个PWM输出通道上的状态。

AQCR1对应控制PWM1通道上的波形输出，AQCR2对应PWM2通道上的波形输出，AQCR3对应PWM3通道上的波形输出，AQCR4对应PWM4通道上的波形输出。AQCRx都具有影子寄存器功能，可以通过AQLDR寄存器对影子寄存器载入到活动寄存器的触发条件进行配置，原理和CMPx的影子寄存器相同，可以参考比较值寄存器载入方式章节。PWM波形控制所支持的触发事件包括：

- CNT = PRD （计数器值等于周期设置值）
- CNT = ZERO （计数器值等于零）
- CNT = C1 when up-couting （计数器值等于C1的参考值，C1参考值由AQCRx[C1SEL]设置）
- CNT = C1 when down-couting （计数器值等于C1的参考值，C1参考值由AQCRx[C1SEL]设置）
- CNT = C2 when up-couting （计数器值等于C2的参考值，C2参考值由AQCRx[C2SEL]设置）
- CNT = C2 when down-couting （计数器值等于C2的参考值，C2参考值由AQCRx[C2SEL]设置）
- T1 事件 when up-couting （通过AQTSCR选择T1事件触发源）

- T2 事件 when down-couting （通过AQTSCR选择T2事件触发源）
- 软件Force事件 （通过软件触发的异步强制置位）

T1和T2是两个独立的触发事件，通过AQTSCR控制寄存器可以从触发事件（SYNCIN4/5）和紧急事件（EPx）中选择一个作为当前事件的触发源。T1和T2事件是基于外部触发的事件，与当前计数器值没有关系，且只用于波形输出控制。

波形发生模块根据计数器的当前计数方向和发生的事件，决定PWM通道上的动作。所支持的的输出动作包括：

- 设置高电平 （在相应PWM通道上设置高电平输出）
- 设置低电平 （在相应PWM通道上设置低电平输出）
- 翻转 （在相应PWM通道上对输出进行翻转）
- 不动作 （不对相应PWM通道进行处理）

C1和C2是波形发生单元中的两个数字比较器，比较器的参考比较值可以通过寄存器AQCRx[C1SEL]和AQCRx[C2SEL]控制位选择CMPx寄存器中的任意一个作为当前比较参考值。

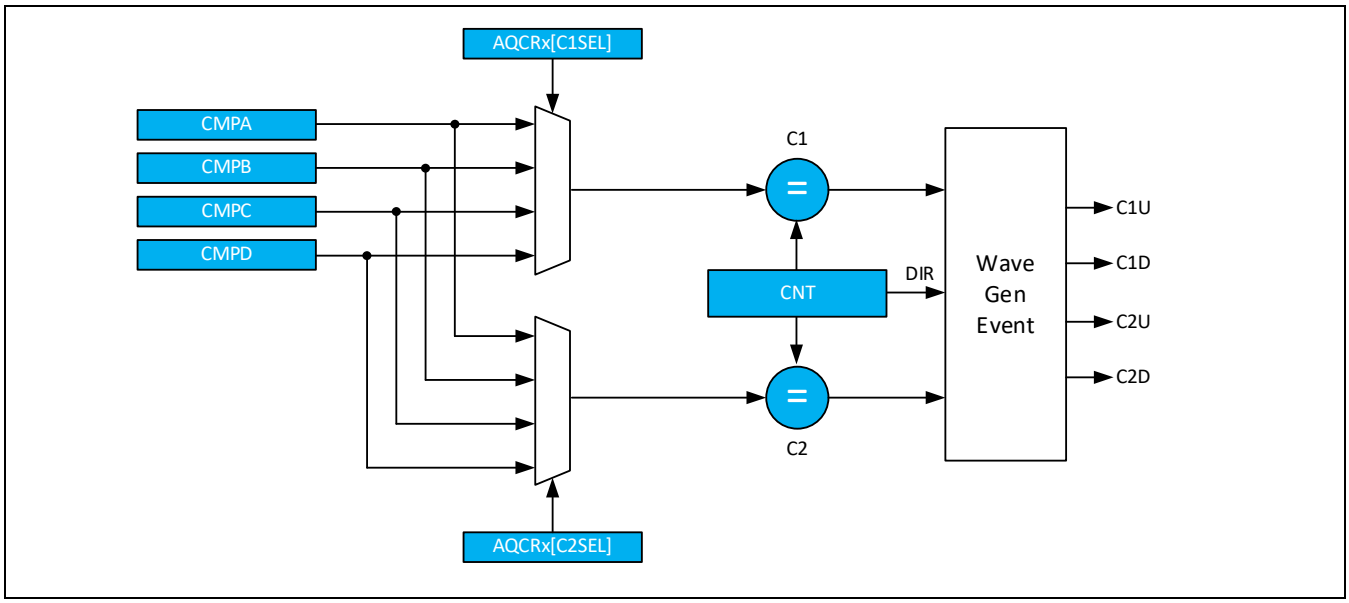


Figure 14-15 C1和C2选择控制

波形发生器可以独立定义每个PWM通道上的输出动作。任何触发事件中的一个或者全部都可以用于产生输出动作。在对不需要触发任何波形输出变化的事件配置中，可以将该事件的触发动作设置为不动作（相当于忽略该事）。在下面的例图中，给出了在波形描述图示中会引用到的图标。

Table 14-2 各种在PWMxX和PWMxY上可能触发的动作

软件 Force	CNT 值等于				事件触发		动作
	Zero	C1SEL	C2SEL	PRD	T1	T2	

SW	Z	C1	C2	P	T1	T2	
—	—	—	—	—	—	—	没有动作
↓	↓	↓	↓	↓	↓	↓	低电平输出
↑	↑	↑	↑	↑	↑	↑	高电平输出
↕	↕	↕	↕	↕	↕	↕	翻转输出

下面的示例中，所有的条件都基于计数器工作时，CMP值不变的情况。在实际系统中，用户可以在每个周期动态调整CMP的设置值。由于Shadow寄存器的作用，实际产生的波形可能晚于设置一个工作周期，或者在下一个计数周期开始时才改变，这都基于用户采用何种计数方式，以及Shadow寄存器的Load方式。所有示例图中C1的比较值选择为CMPA，C2的比较值选择为CMPB。

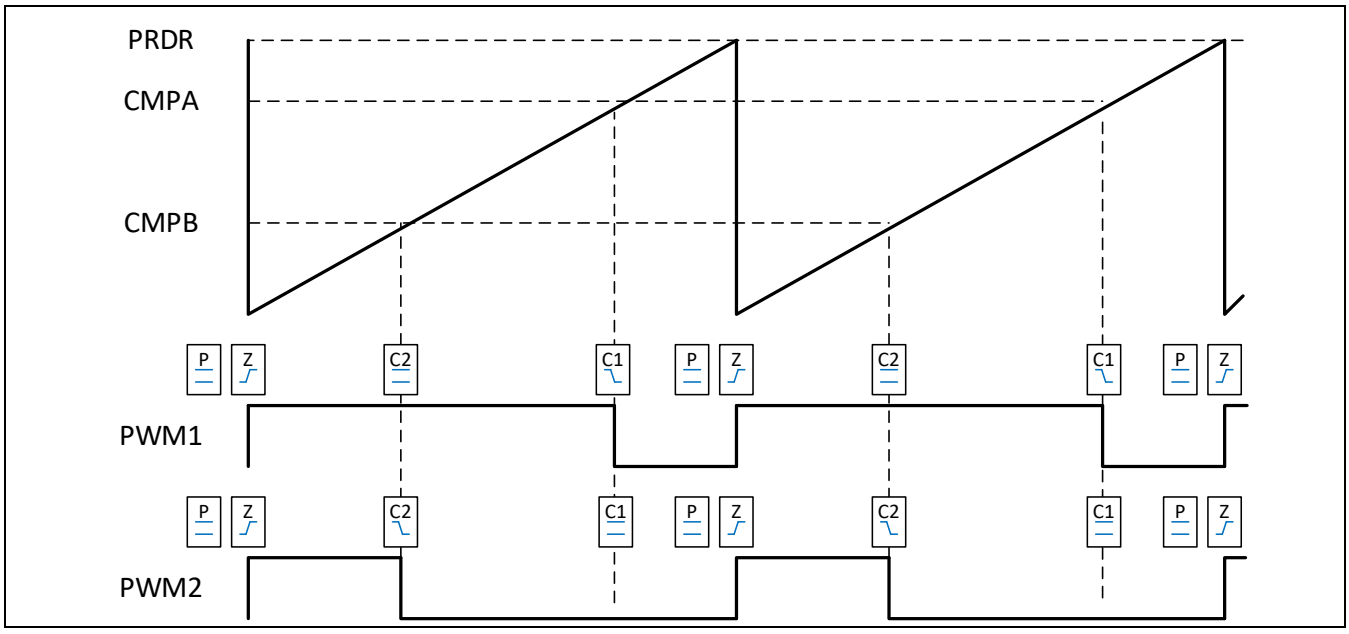


Figure 14-16 递增单沿，非对称波形输出

在上图中，不能很清楚的分辨出Zero和Period事件触发的区别，实际上，两个事件相差一个计数时钟。

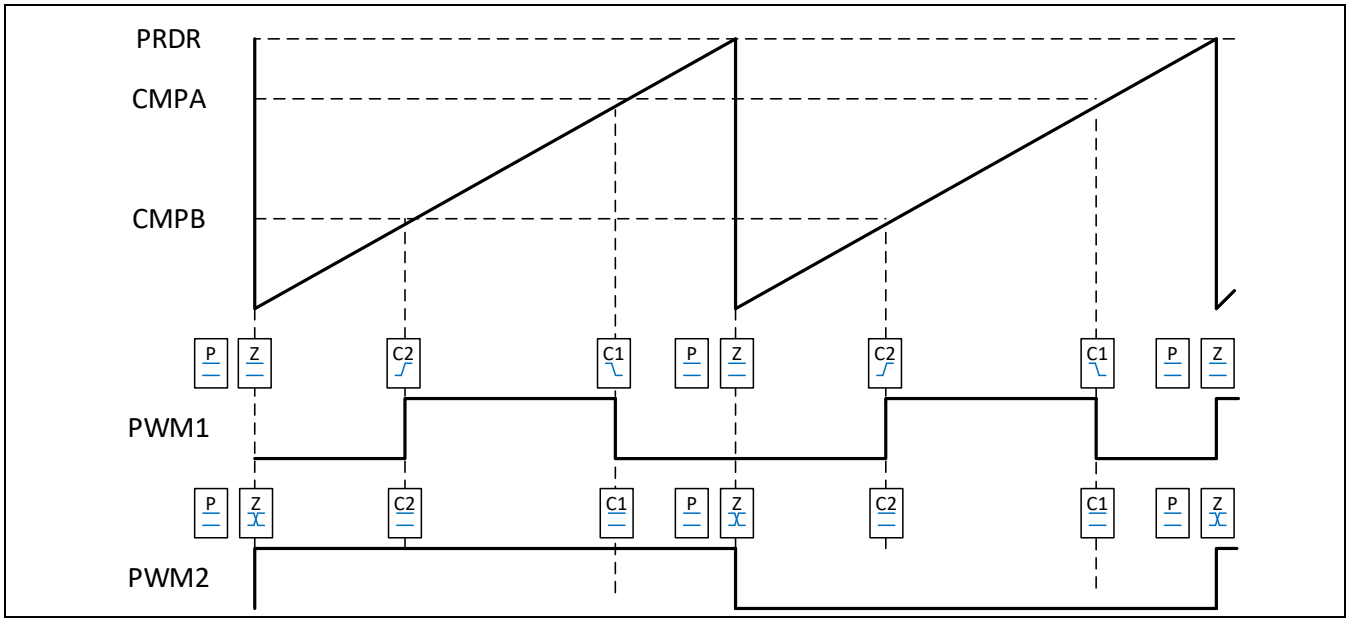


Figure 14-17 递增，脉冲定位非对称波形输出

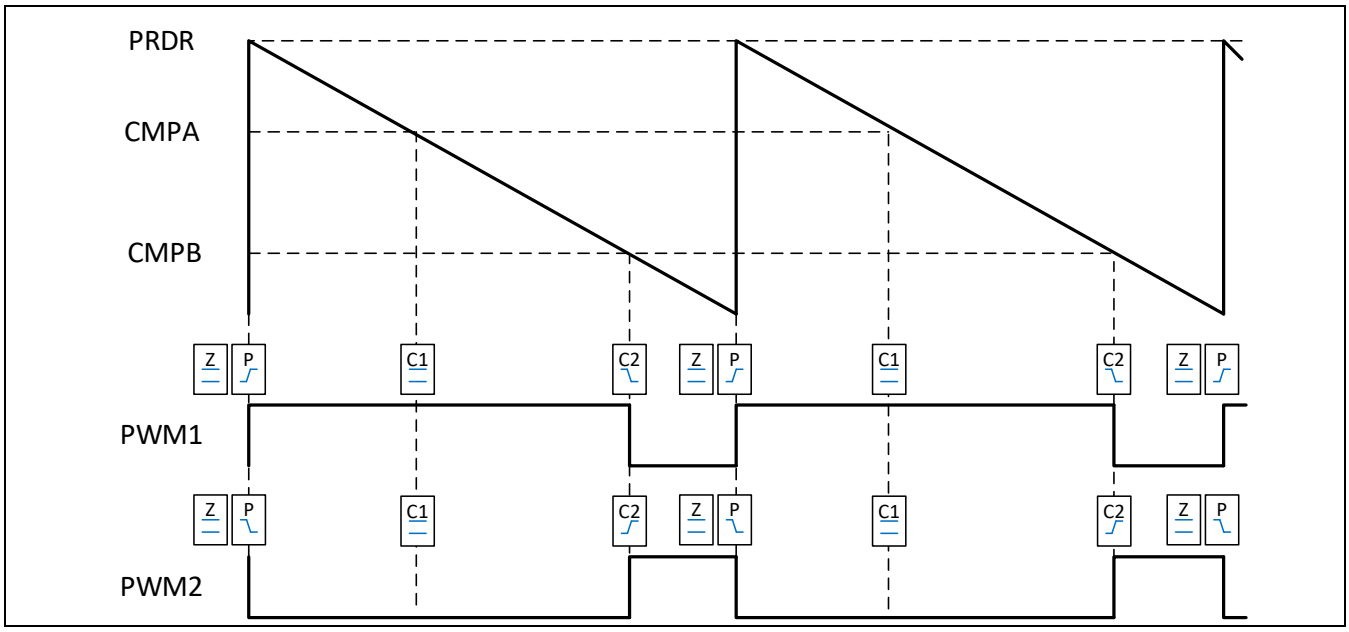


Figure 14-18 递减单沿，非对称波形输出

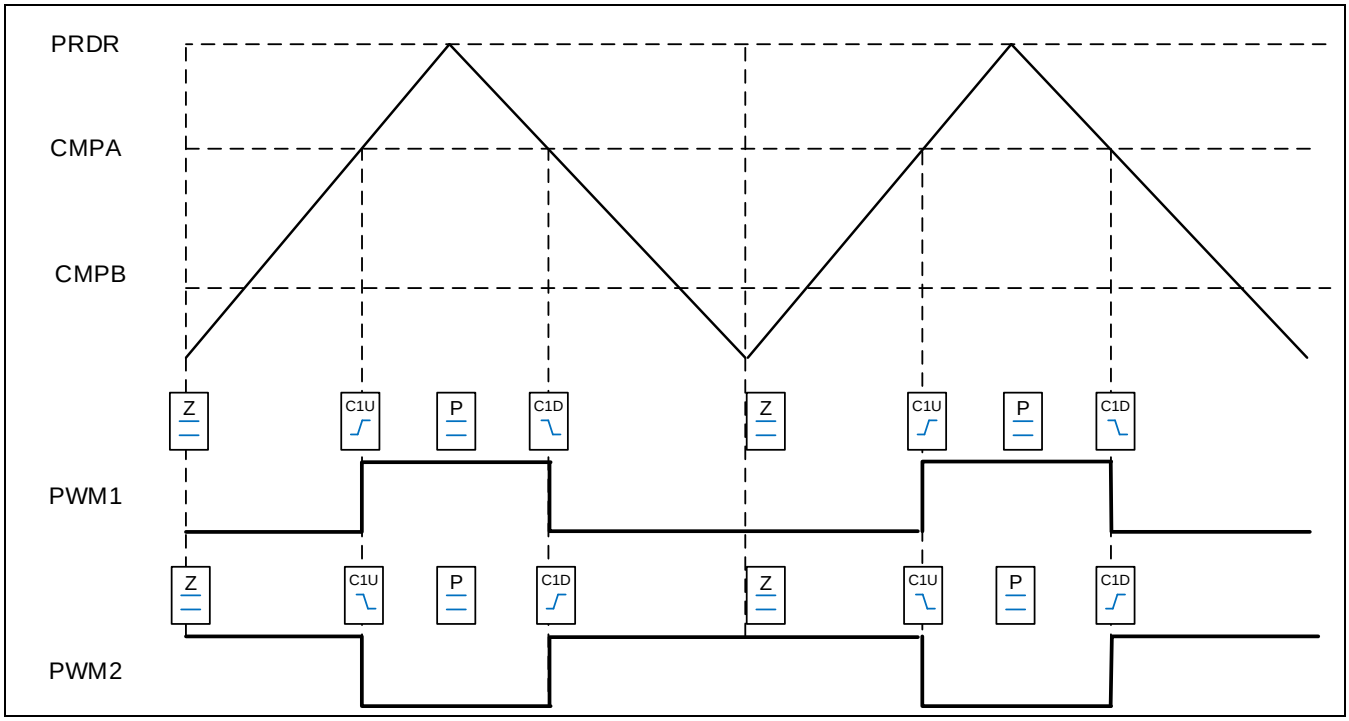


Figure 14-19 递增递减，双沿对称波形输出

13.3.4.2 触发事件的优先级

在同一个时间可能有多个事件同时触发，在这种情况下，具有高优先级的事件将决定输出的状态。通常，后发生的事件具有更高的优先级，而软件强制输出具有最高的优先级。优先级通过硬件决定，不能由寄存器进行更改。在下表中，列出各种计数模式下的优先级设置，优先级数字越小代表优先级更高。

Table 14-3 递增递减模式（Up-Down-Count）下的事件优先级

优先级	触发事件（递增阶段）	触发事件（递减阶段）
1(Highest)	Software Forced event	Software Forced event
2	T1 on up-count(T1U)	T1 on down-count(T1D)
3	T2 on up-count(T2U)	T2 on down-count(T2D)
4	CNT = C2 on up-count(C2U)	CNT = C2 on down-count(C2D)
5	CNT = C1 on up-count(C1U)	CNT = C1 on down-count(C1D)
6	CNT equals zero	CNT equals period
7	T1 on down-count(T1D)	T1 on up-count(T1U)
8	T2 on down-count(T2D)	T2 on up-count(T2U)
9	CNT = C2 on down-count(C2D)	CNT = C2 on up-count(C2U)
10(Lowest)	CNT = C1 on down-count(C1D)	CNT = C1 on up-count(C1U)

Table 14-4 递增模式下的事件优先级

优先级	触发事件
1(Highest)	Software Forced event
2	CNT = period
3	T1 on up-count(T1U)
4	T2 on up-count(T2U)
5	CNT = C2 on up-count(C2U)
6	CNT = C1 on up-count(C1U)
7(Lowest)	CNT = zero

在递增模式下，由于计数器方向一直保持递增，所以和递减相关的事件将永远不会发生。

Table 14-5 递减模式下的事件优先级

优先级	触发事件
1(Highest)	Software Forced event
2	CNT equals zero
3	T1 on down-count(T1D)
4	T2 on down-count(T2D)
5	CNT equals C2 on down-count(C2D)
6	CNT equals C1 on down-count(C1D)
7(Lowest)	CNT equals period

在递减模式下，由于计数器方向一直保持递减，所以和递增相关的事件将永远不会发生。

用户可以随意设置CMPx的值，当设置的CMP值大于Period的设置值时，将会按照下述方式进行操作。

- 计数器设置为递增或递减模式时, C1D/C2D和C1U/C2U事件始终不会被触发。
- 计数器设置为递增递减模式时：C1U/C2U不会被触发，C1D/C2D事件在CNT等于Period时触发。

13.3.4.3 通过软件强制设置波形

PWM引擎的波形输出支持通过软件进行控制。软件强制输出可以将输出信号通过软件强制设置为预设电平，此功能类似输出控制级中紧急模式下的波形输出控制，但是紧急模式下的波形输出具有更高的优先级，且具有异常标志和中断报警特性。

软件强制输出可以分为两种模式：一次性Force和持续性Force。

一次性软件强制输出（One-Shot Software Forcing）

在此模式下，通过寄存器的设置可以将PWM1/2/3/4（注意不是最终管脚上的输出波形）的输出强制修改成软件设置电平，且该电平一直维持到有新的触发事件发生。可以通过设置寄存器AQOSF[ACTA/2/3/4]控制位，设置在一次性强制输出触发时，PWM1/2/3/4上的输出电平状态。通过对AQOSF[OSTSF1/2/3/4]控制位写入1，触发一次性强制输出。

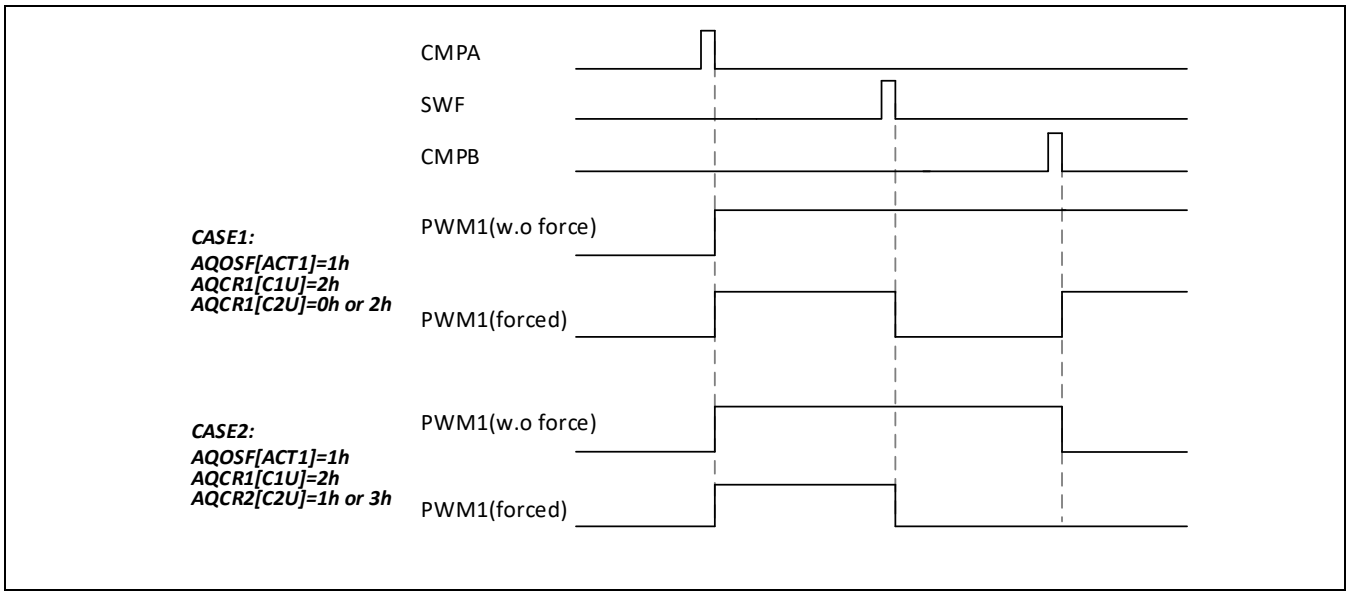


Figure 14-20 一次性软件强制输出

持续性软件强制输出（Continuous Software Forcing）

在此模式下，通过寄存器的设置可以将通道的输出强制修改成软件设置电平，且该电平一直维持到软件清除才结束。当软件清除持续性强制输出状态后，通道电平将恢复到强制输出前的状态。可以通过设置寄存器AQCSF[CSF1/2/3/4]控制位，进行强制输出设置或者清除。

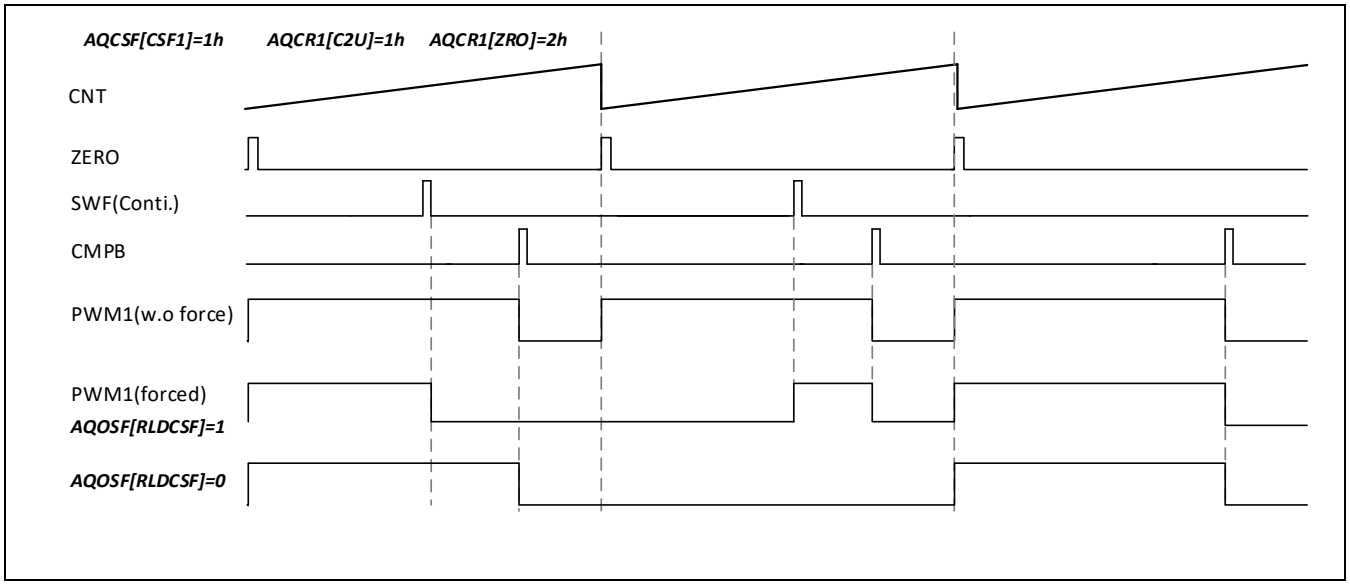


Figure 14-21 持续性软件强制输出

13.3.4.4 不同计数模式下的波形输出

在计数器递增（Up-counting）模式下，可以配置产生非对称的PWM波形。在递增模式下，通常设置活动寄存器的更新触发点为CNT=Zero，即周期开始前将Shadow寄存器载入到活动寄存器中。通过AQCR寄存器设置计数器在ZRO点、C1U点和C2U点时PWM的输出动作。

当CMP值由0到PRDR+1进行调整时，可以获得0到100%的PWM占空比输出（注意：需要获得100%占空比，需要设置CMP值>PRDR，在此设置下，将不会发生C1U或者C2U触发事件）。

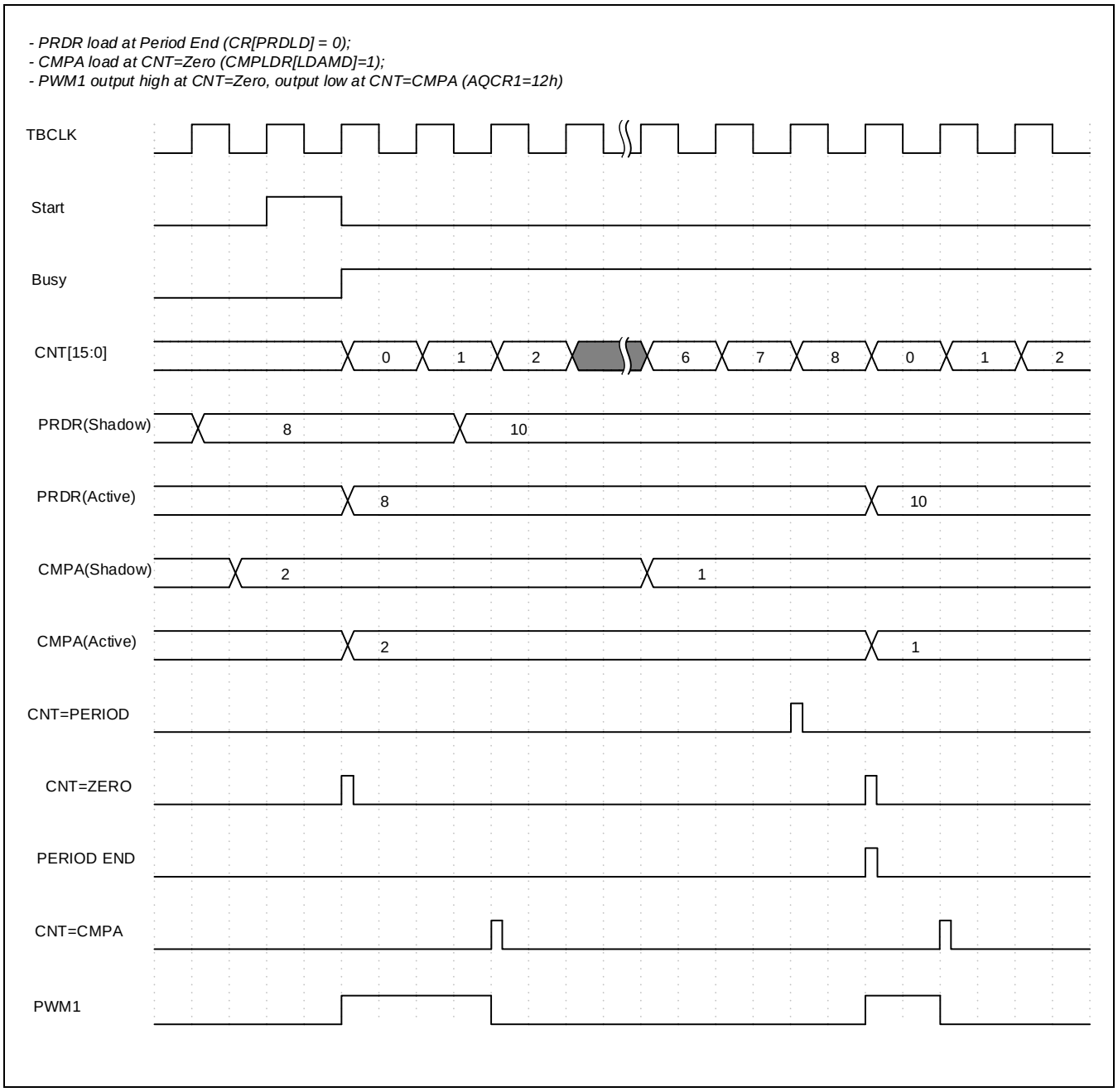


Figure 14-22 递增单比较值时，非对称波形输出

在计数器递减（Down-counting）模式下，可以配置产生非对称的PWM波形。在递减模式下，通常设置活动寄存器的更新触发点为CNT=Period，即周期开始前将Shadow寄存器载入到活动寄存器中。通过AQCR寄存器设置PRD点时PWM的有效电平输出，并在递减阶段比较值相等时清除PWM输出(C1D or C2D)。当CMP值由PRDR到0进行调整时，可以获得0到几乎100%的PWM占空比输出（注意：由于在递减模式下，CMP比较值不能设置为比0更小的数值。但是当比较值设置为0时，PWM在周期结束前，会输出一个CLK宽度的脉冲。在需要完全100%PWM占空比输出的情下，需要选择递增计数模式，或者递增递减计数模式）。

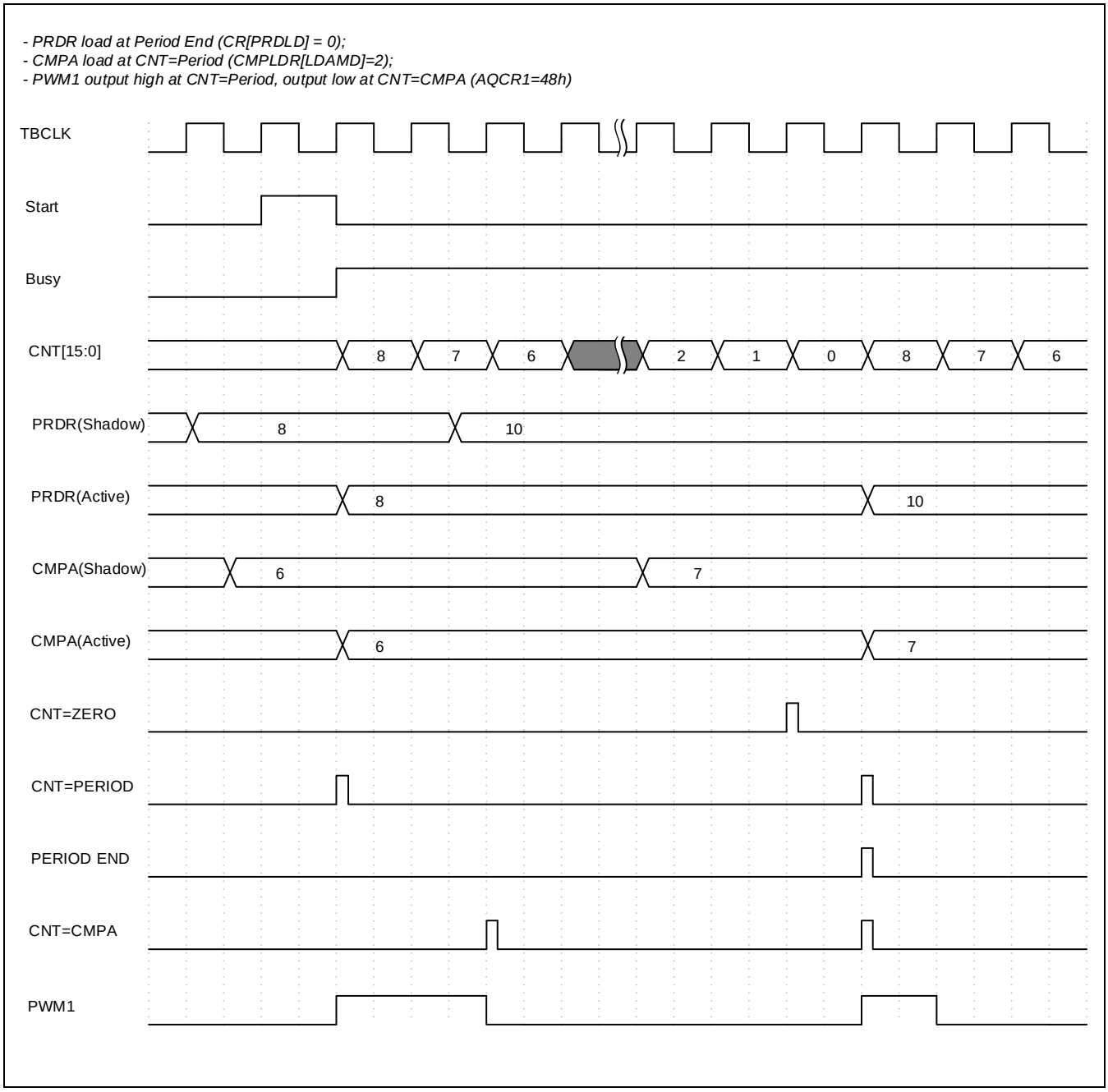


Figure 14-23 递减单比较值时，非对称波形输出

在计数器递增递减（Up-down-counting）模式下，可以配置产生非对称或者对称的PWM波形。通常情况下，在递增和递减阶段使用同一个比较值对输出进行处理，可以输出一个对称的PWM波形。在对称输出时，当比较值配置为零时，可以得到100%占空比输出的PWM对称波形。随着比较值的增大，输出波形的占空比逐渐缩小。当比较值等于PRDR-1时，可以获得最小非零占空比输出的波形。当比较值设为等于或者大于PRDR时，输出的PWM波形占空比为零。

在计数器递增递减模式下，配置为非对称PWM波形输出时，可以通过设置递增阶段的C1和递减阶段的C2两个比较点产生非对称的PWM波形输出。

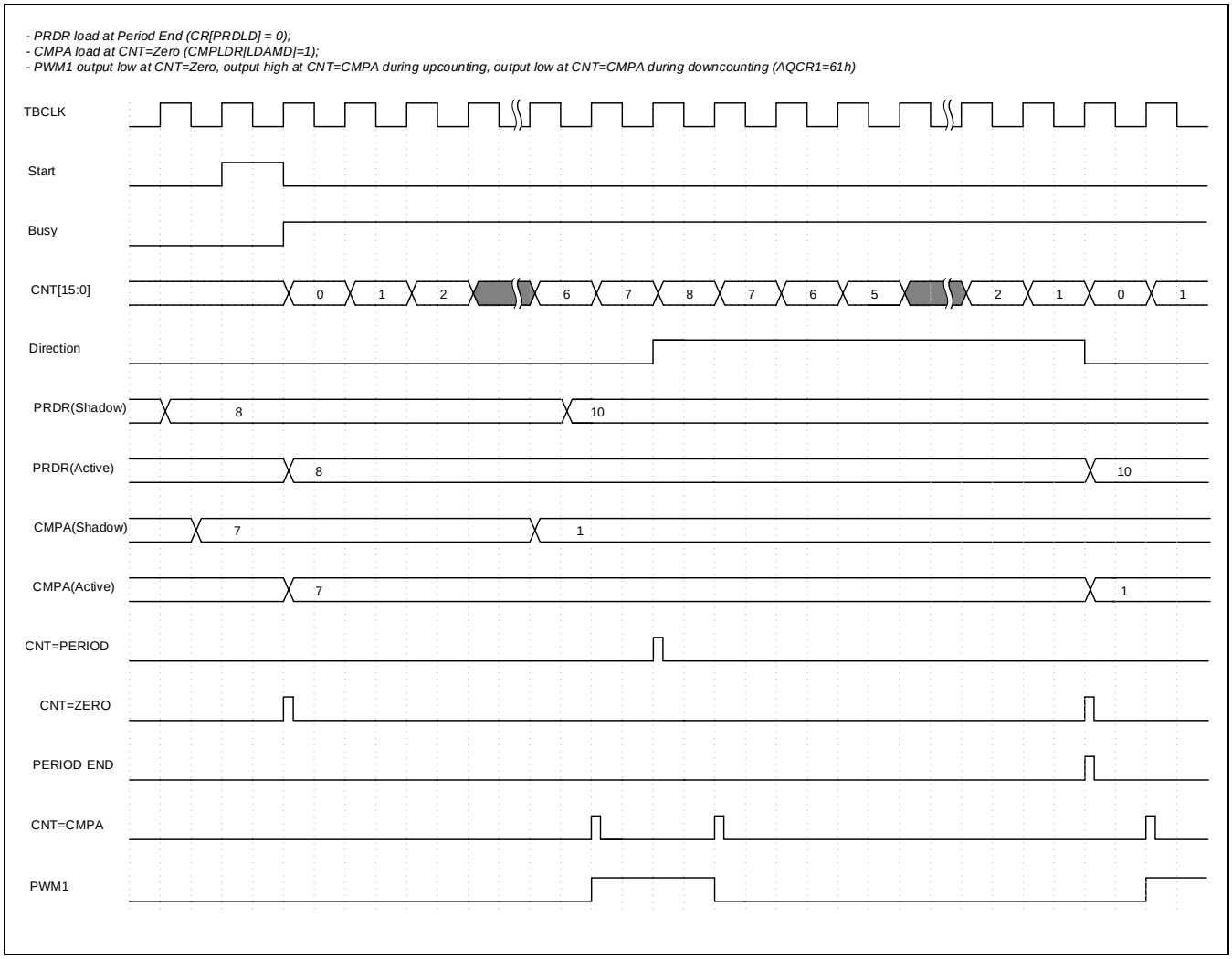


Figure 14-24 递增递减单比较值时，对称波形输出

在递增递减模式下，可以支持多种PWM波形输出方式，下面给出几种在递增递减模式下结合不同配置产生双边沿PWM波形的示意图。

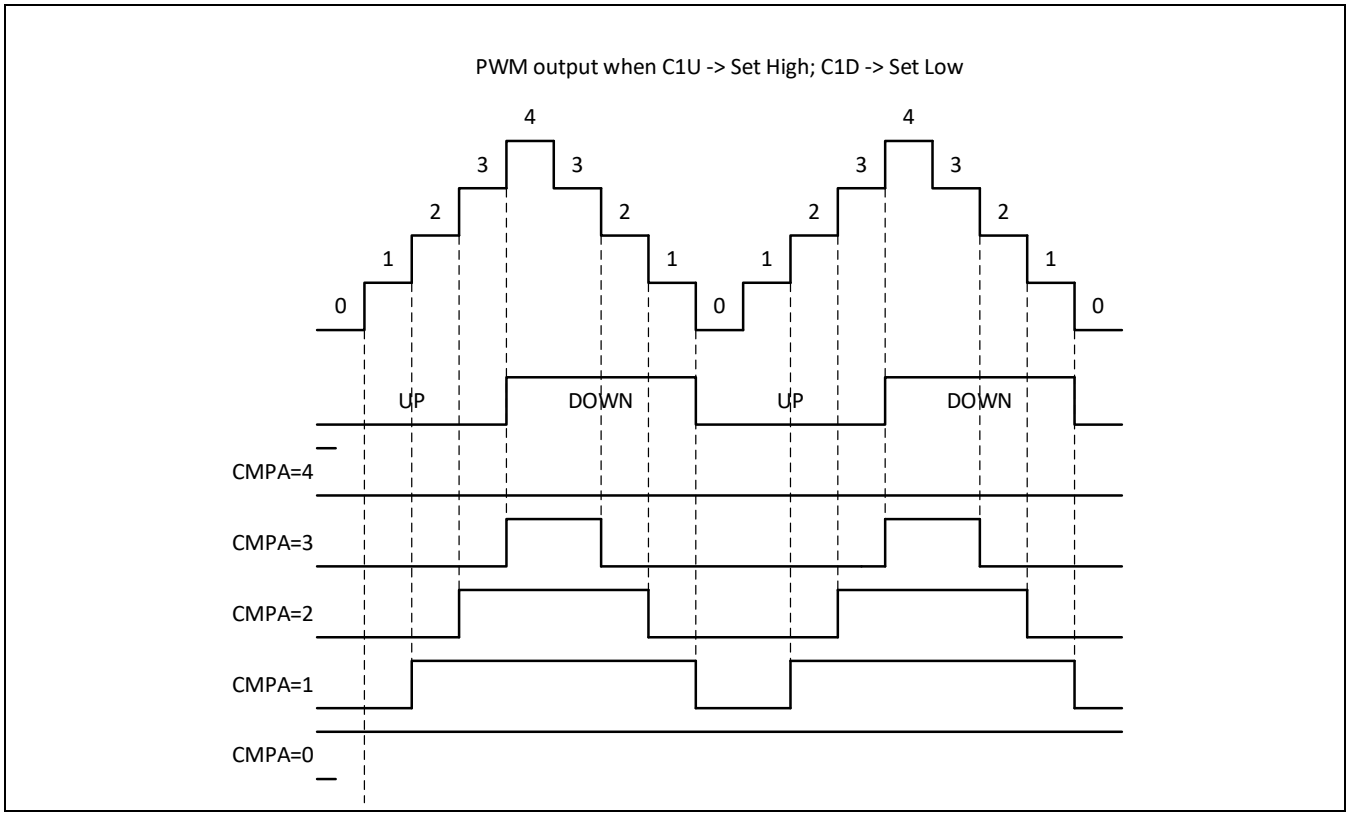


Figure 14-25 递增递减单比较值时，对称波形输出

13.3.5 死区控制

在前一章节中已经介绍过，通过脉冲定位的输出方式，可以由软件控制输出自定义的带死区的互补波形。然而，如果用户希望采用更加经典的，基于边沿延时来实现的极性可调的死区控制，此功能可以通过使能死区控制模块来实现。

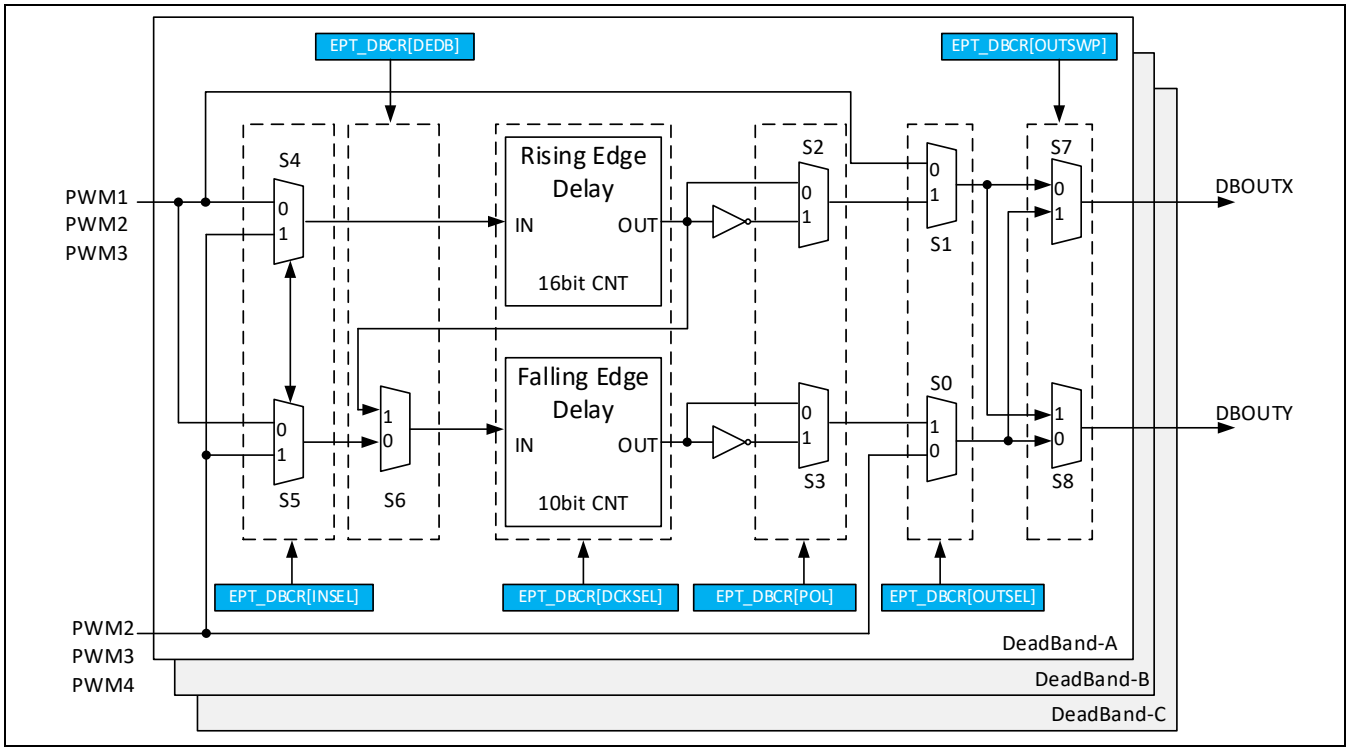


Figure 14-26 死区控制模块

死区控制模块主要由两个边沿延时模块以及相应的信号选择开关组成。现在对每个开关的功能描述如下：

开关	功能描述	寄存器控制位
S4，S5	对于X通道和Y通道的延时电路，独立选择两个来自于PWM数字引擎的PWM输出信号作为输入源。 对于DeadBandA，PWM1/2可选；对于DeadBandB，PWM2/3可选；对于DeadBandC，PWM3/4可选。	DBCR[CHA_INSEL] DBCR[CHB_INSEL] DBCR[CHC_INSEL]
S2，S3	延时模块输出端的极性选择。	DBCR[CHA_POL] DBCR[CHB_POL] DBCR[CHC_POL]
S0，S1	控制是否旁路延时控制模块	DBCR[CHA_OUSEL] DBCR[CHB_OUSEL] DBCR[CHC_OUSEL]
S7，S8	输出交换控制	DBCR[CHA_OUTSWAP]
S6	Y通道是否使用两级延时(dual-edge delay)	DBCR[CHA_DEDB] DBCR[CHB_DEDB] DBCR[CHC_DEDB]

延时控制电路是一个基于14位计数器的延时逻辑，分为上升沿延时和下降沿延时两种。上升沿延时模块，只对输入信号的上升沿作延时处理，下降沿则保持和输入信号一致；而下降沿延时模块，只对输入信号的下降沿作延时

处理，上升沿则保持和输入信号一致。延时的长度由DBCR[DTR]和DBCR[DTF]决定。延时的计算方式如下：

T_{RED} = DTR x T_{DBCLK}

T_{FED} = DTF x T_{DBCLK}

T_{DBCLK} 表示死区延时控制计数器的计数时钟，此时钟可以通过DBCR[DCKSEL]控制位选择TCLK或者HCLK作为时钟源。当选择HCLK时钟作为时钟源时，T_{DBCLK}的时钟频率为 HCLK/DPSCR。选择HCLK作为死区控制时钟，可以更加精确的控制死区宽度。

DBMD、DBDTR和DBDTF都具有对应的Shadow寄存器，可以通过DBCR设置DBMD活动寄存器的加载方式，DBDTR和DBDTF活动寄存器的加载方式设置，在DBMD中进行设置。全局载入控制可以覆盖这三个寄存器的加载方式。

Table 14-6 支持的死区控制模式

模式	模式描述	OUTSWP		DEDB	POL		OUTSEL	
		S8	S7	S6	S3	S2	S1	S0
1	Dead-band control is bypassed (No delay)	0	0	0	X	X	0	0
2	Active high with complementary	0	0	0	1	0	1	1
3	Active low with complementary	0	0	0	0	1	1	1
4	Both active high	0	0	0	0	0	1	1
5	Both active low	0	0	0	1	1	1	1
6	OUTX has no delay, OUTY is falling edge delayed	0	0	0	X	X	0	1
7	OUTX is rising edge delayed, OUTY has no delay	0	0	0	X	X	1	0
	OUTY is dual-edge delayed.	X	X	1	0/1	0/1	0/1	0

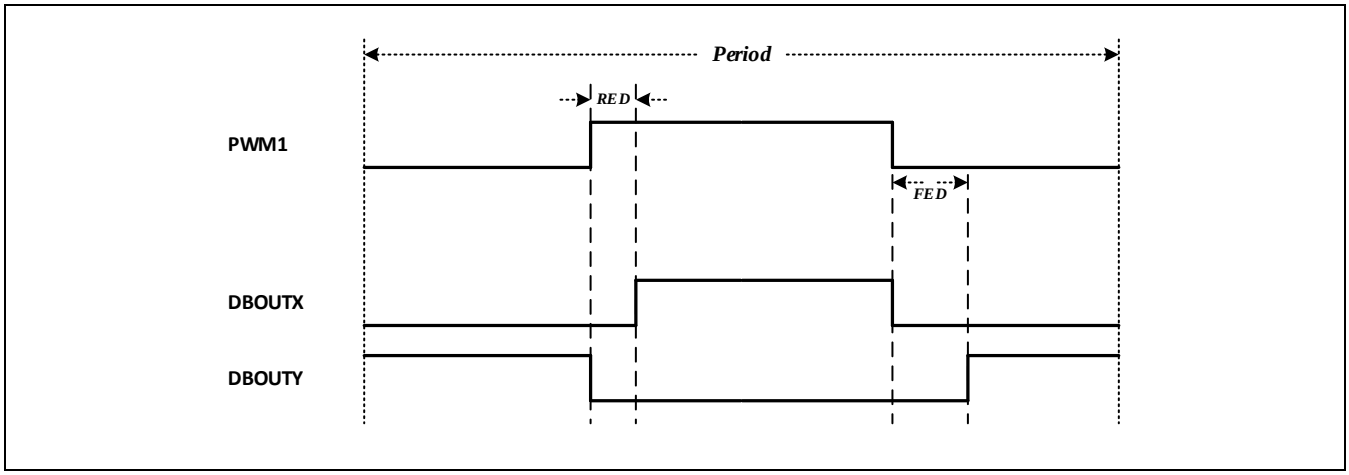


Figure 14-27 模式2：高电平有效的死区互补输出举例

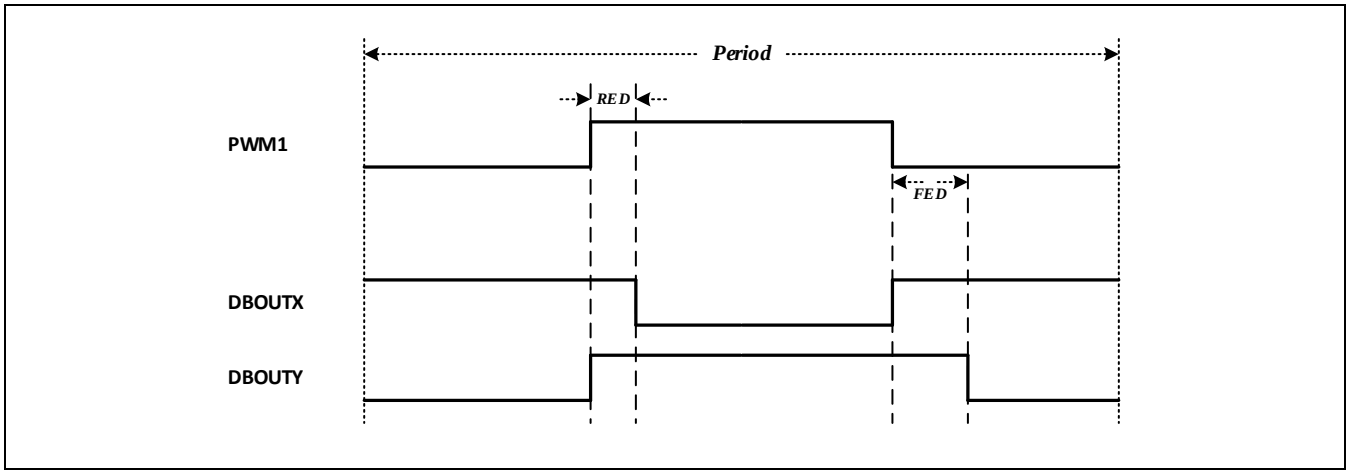


Figure 14-28 模式3：低电平有效的死区互补输出举例

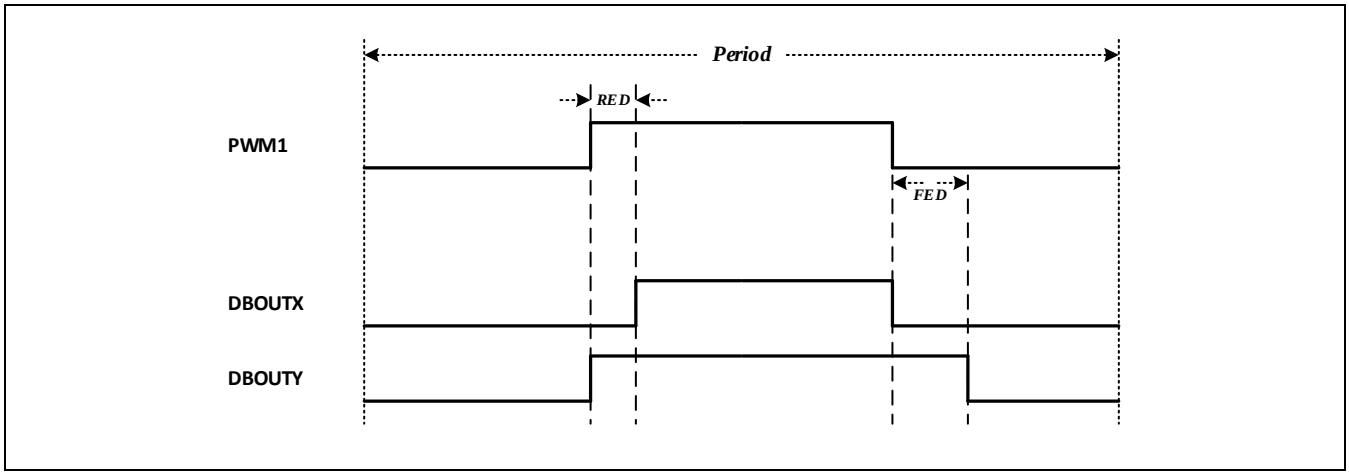


Figure 14-29 模式4：高低电平有效死区输出举例

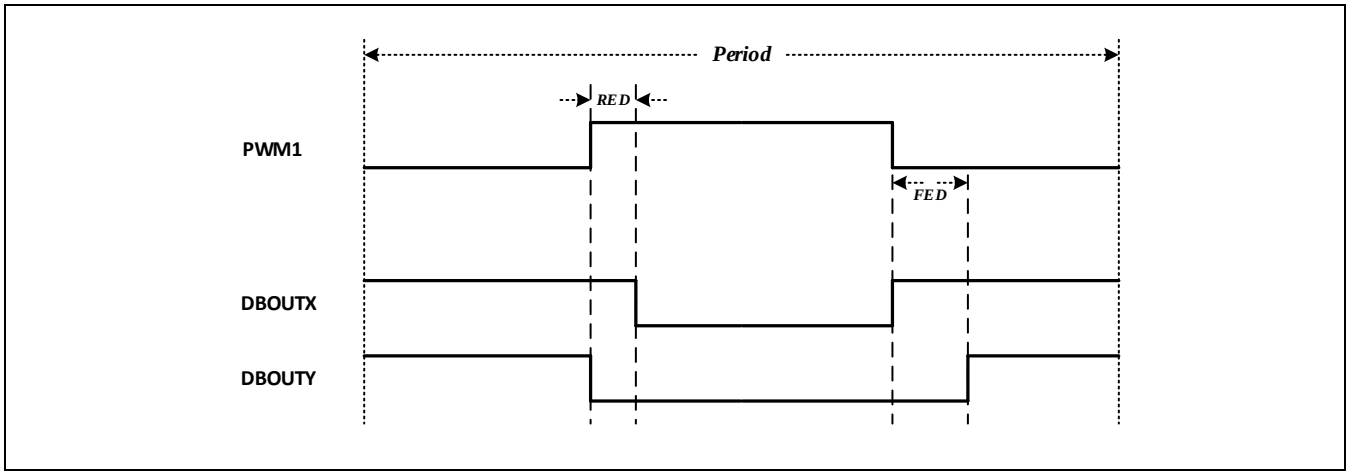


Figure 14-30 模式5：低高电平有效死区输出举例

13.3.6 斩波模块

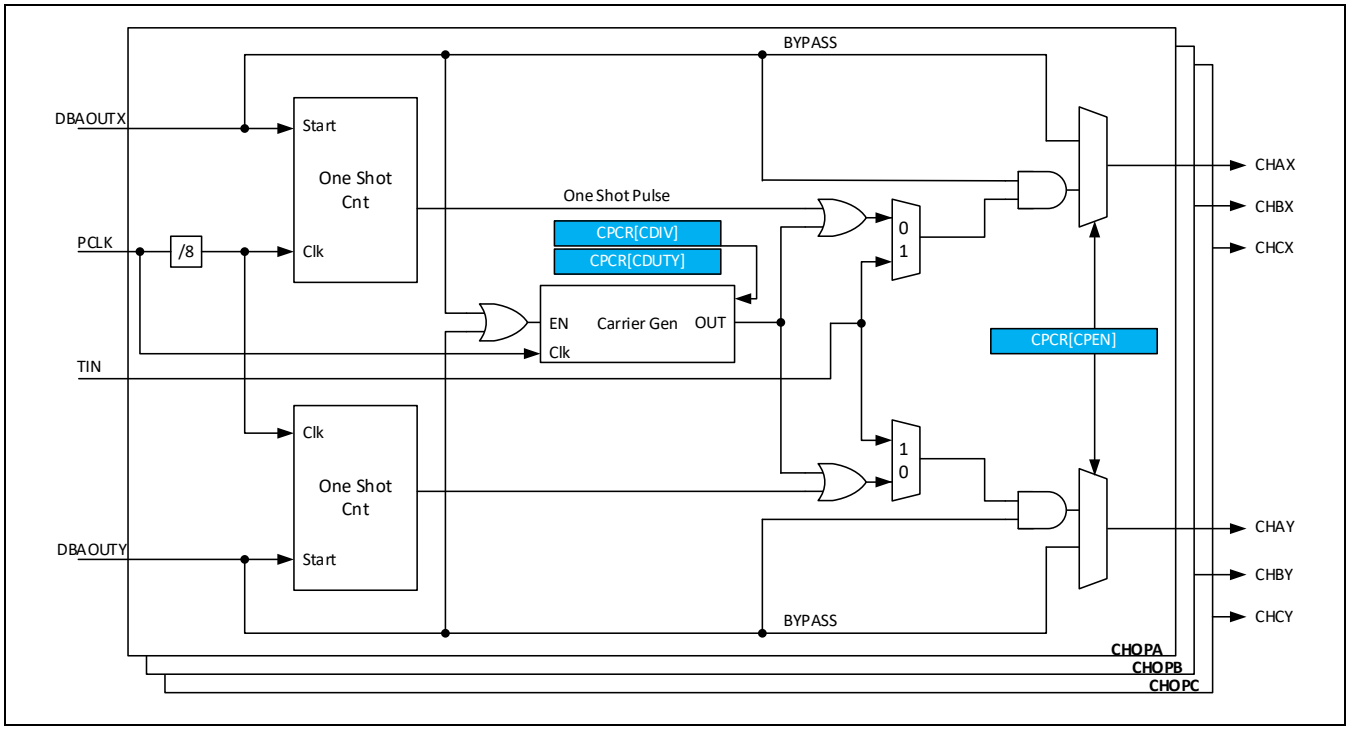


Figure 14-31 斩波使能模块

PWM的斩波模块是将一个频率更高的载波信号来调制波形发生模块产生的PWM信号。这个功能对于通过脉冲转换的开关功率驱动应用非常重要。斩波模块可以实现可编程的载波频率，可编程的载波首脉冲的脉冲宽度，可编程的第二个和后续脉冲的占空比。如果应用中不需要斩波功能，用户可以旁路该处理模块。

斩波模块的载波通过PCLK分频得到。载波频率基于PCLK的8分频倍率配置，计算方式如下：

F_{chop} = PCLK / (8 x (CDIV+1))

载波的频率和占空比可以通过CPCR[CDIV]和CPCR[CDUTY]控制位进行设置。占空比设置支持7种配置，以1/8为基础，逐次累加，最高到7/8占空比。斩波使能阶段的首个输出脉冲的宽度可以通过软件进行扩展，以保证功率开关的快速打开，后续的规律脉冲则保持功率开关管的状态维持。首脉冲宽度可以通过CPCR[OSPWTH]控制位设置。当CPCR[OSPWTH]为零时，首脉冲宽度和后续脉冲一致。当设置首脉冲的宽度时，设置宽度是载波周期的整数倍。

$T_{1stpulse}$ = T_{chop} x OSPWTH

其中，T_{chop}为CPCR[CDIV]设置的载波周期时间（1/ F_{chop}）。

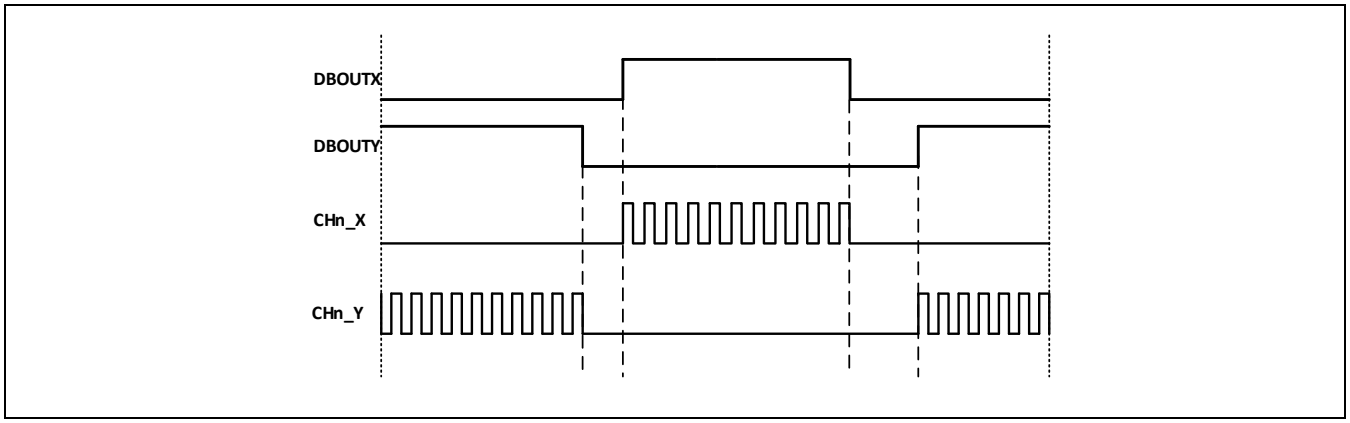


Figure 14-32 简单的斩波输出

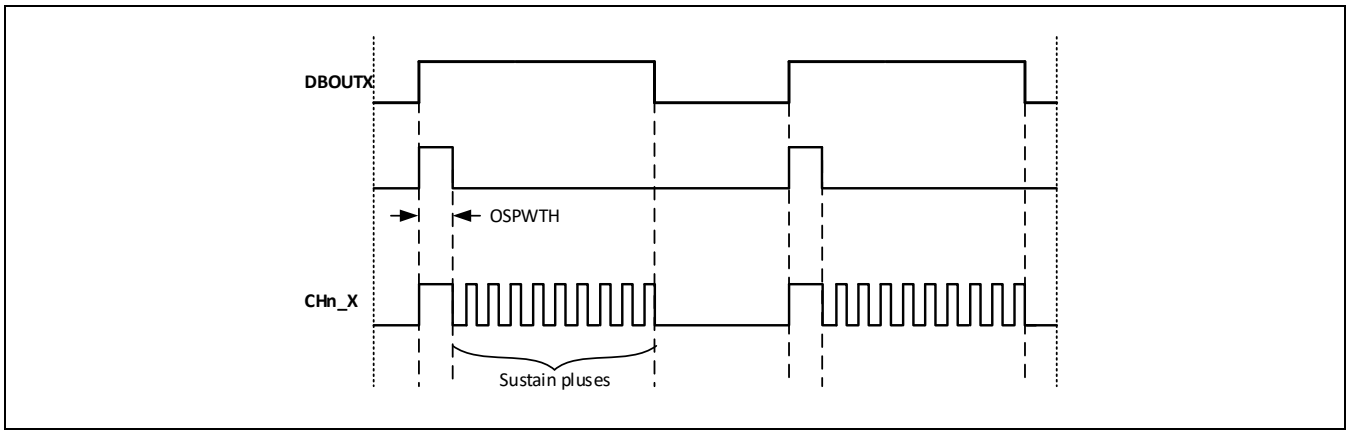


Figure 14-33 首脉冲扩展后的波形

载波信号也可以通过CPCR[C1SEL]控制位选择外部其他计时器作为载波信号的发生源，当选择外部载波时，TIN的输入可以通过CEDR[TINSEL]控制位指定LPT或者CNTA作为外部载波的发生器。在外部载波模式下，由于载波信号和调制波无周期设置相关性，所以不能保证载波和调制波在相位上完全对齐，可能造成起始的第一个载波周期，或者结束的最后一个周期小于载波信号的设置周期。

为保证相位一致，需要将载波周期和调制波周期设置为整数倍，且通过EPT的周期信号同步触发载波信号发生TIMER的计数器，或者通过多个TIMER同步计数功能，同时启动EPT和载波生成TIMER。

斩波功能只有在死区控制的输出通路上有作用，没有经过死区控制的PWM4通道不支持该功能。

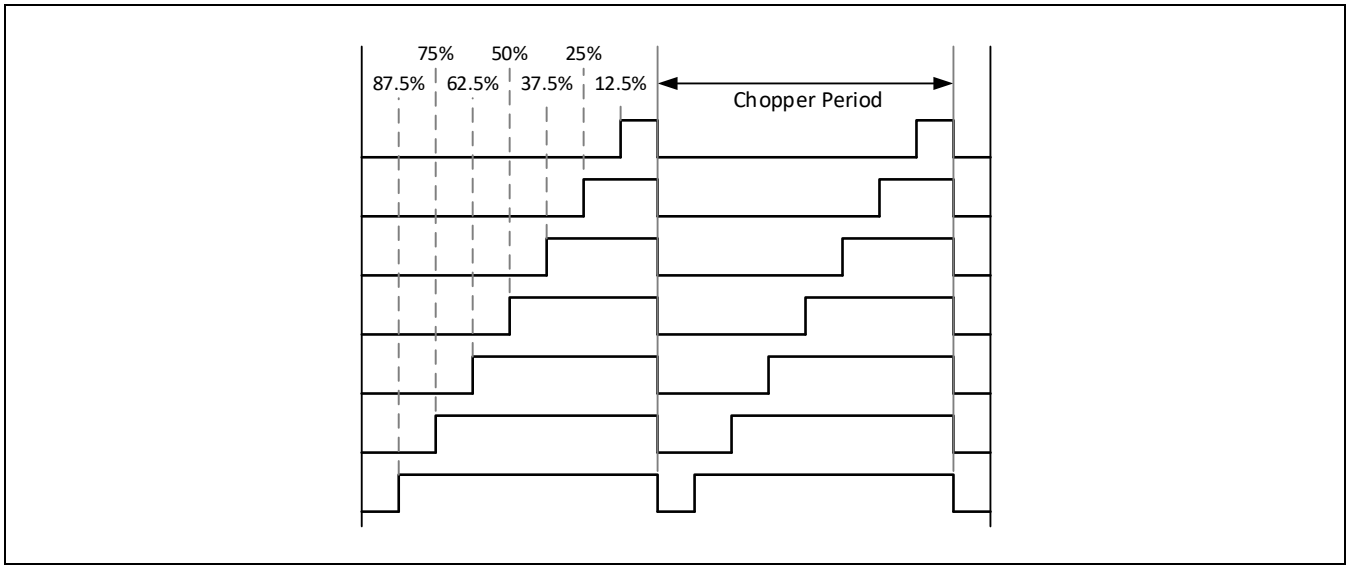


Figure 14-34 载波的占空比

13.3.7 紧急模式控制

13.3.7.1 紧急模式工作机制

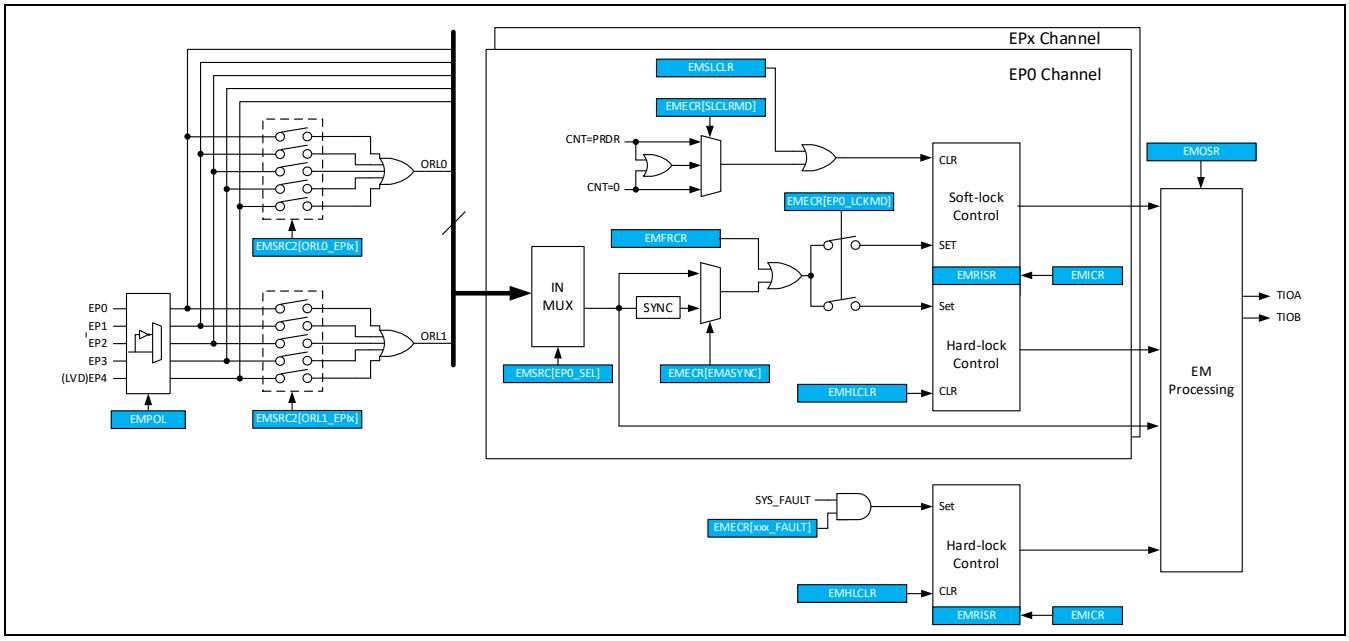


Figure 14-35紧急模式控制模块

在很多应用场合，PWM输出需要对紧急异常状态做出相应的输出控制，实现过载保护，或故障处理。紧急事件

处理模块可以对应外部触发，产生相应输出并产生相应中断。模块内支持8路紧急触发信号输入（EPx）通道，和3路系统故障触发通道(SYSFAIL)。

紧急模式控制模块主要支持特性有：

- 紧急模式下，PWM的输出可以被定义为：高电平输出，低电平输出，高阻，或者不进行动作。
- 对紧急模式的处置策略有两种：硬锁止（Hard-Lock），主要用于短路或者过流保护；软锁止（Soft-Lock），主要用于限流保护动作。
- 支持8路触发输入，每个触发输入可以独立选择触发信号源，实现PWM输出的保护联动。
- 独立的系统故障触发源。
- 独立的紧急模式触发中断源。
- 支持软件强制触发紧急状态。

每路EP触发通道，可以从外部GPIO，LVD标志，模拟比较器输出（或系统内嵌比较器）或者所有可能的触发源的逻辑或输出中选择一个作为当前EP通道的触发源。EP的输入源选择，通过EMSRC寄存器进行设置，逻辑或的输入选择通过EMSRC2进行设置。紧急模式还支持在系统发生错误，触发独立的硬锁止输出。系统错误包含：CPU错误（不可恢复异常），内存错误（Flash校验错误，或者SRAM校验错误）以及外部晶振失效错误。

EBI的触发极性可以通过EMPOL寄存器进行设置，缺省为高电平有效。当EBI满足触发条件时，PWM的输出会立即做出相应变化，但对于EM的FLAG，需要进过PCLK同步后才能置位。当输入的EBI宽度小于PCLK的同步周期时（同步需要1到2个PCLK周期），PWM的输出端口，在EBI条件不满足时，不会继续保持EM输出状态，同时EM的FLAG也不会发生变化。通过设置EP通道的同步，可以确保只有在EM的FLAG被置位后，才会有PWM的状态改变，但这样会额外增加EM输出的响应时间。

每一个EP可以被配置为Soft-lock或者Hard-lock处置策略。所有的系统错误触发只能作为Hard-lock处置策略的触发源，EP的对应处置策略可以通过EMECR控制寄存器设置。响应策略的紧急状态输出设置可以通过EMOSR寄存器配置。

- **软锁止模式 Soft-lock（SL）：**

  当SL事件被检测到，PWM1和PWM2端口的输出将根据EMOSR中控制位的设置立即作出动作。所有可能设置的紧急状态输出如下：

  - 高阻态。
  - 高电平输出。
  - 低电平输出。
  - 不做处理。

  当SL触发条件满足时，相应输出端口将输出预设值，EMSLSR寄存器中的相应位将被置位，EMRISR寄存器中的相应中断标志位置也将被置位，如果中断使能控制位(EMIMCR)有效，则会产生相应的中断请求。当锁止标志被清除后，PWM输出将恢复。软锁止标志的清除可以通过软件对EMSLCLR寄存器相应控制位写1进行，或者在计数器值等于EMECR [SLCLRMD]控制位选择的条件时，硬件自动清除。当清除标志位时，如果SL的触发条件仍然满足，则清除操作无效。当软锁止标志位被清除，PWM恢复输出时，中断标志位位仍EMRISR中的相应旧需要软件进行清除。

- **硬锁止模式 Hard-lock（HL）：**

当HL事件被检测到，PWM1和PWM2端口输出将根据EMOSR中控制位的设置作出动作。所有可能设置的紧急状态输出和软锁止相同。当HL触发条件满足时，相应输出端口将输出预设值，EMHLSR寄存器中的相应位被置位，EMRISR寄存器中的相应中断标志位也将被置位，如果中断使能控制位(EMIMCR)有效，则会产生相应的中断请求。HL条件触发的FLG不会自动清除，必须要通过软件写EMHLCLR寄存器进行清除。在FLG标志未清除前，相应的输出始终保持在紧急输出状态。

紧急状态也支持通过软件触发。当对EMFRCR寄存器相应控制位写入1时，相应EP通道将被触发。触发的效果和外部触发设置相同。所有的紧急状态输出，只有在相应的FLG状态位被清除后，才会恢复到正常输出。

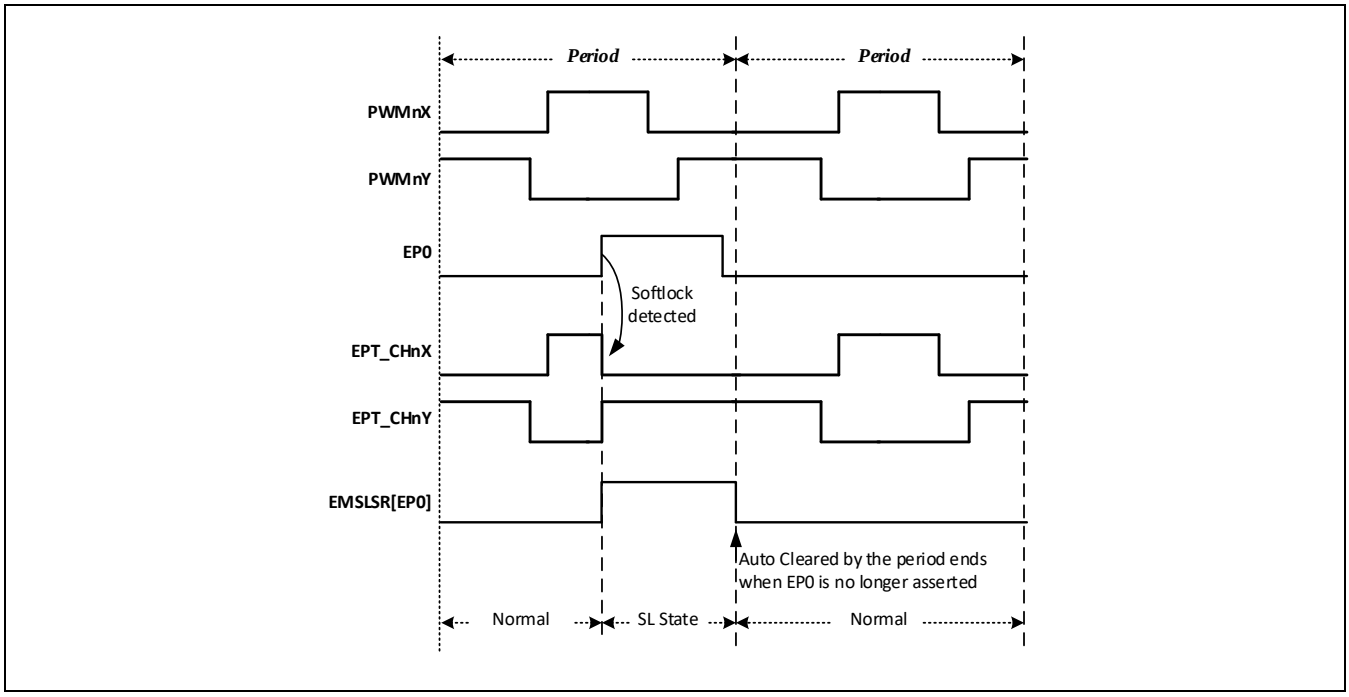


Figure 14-36 软锁止模式

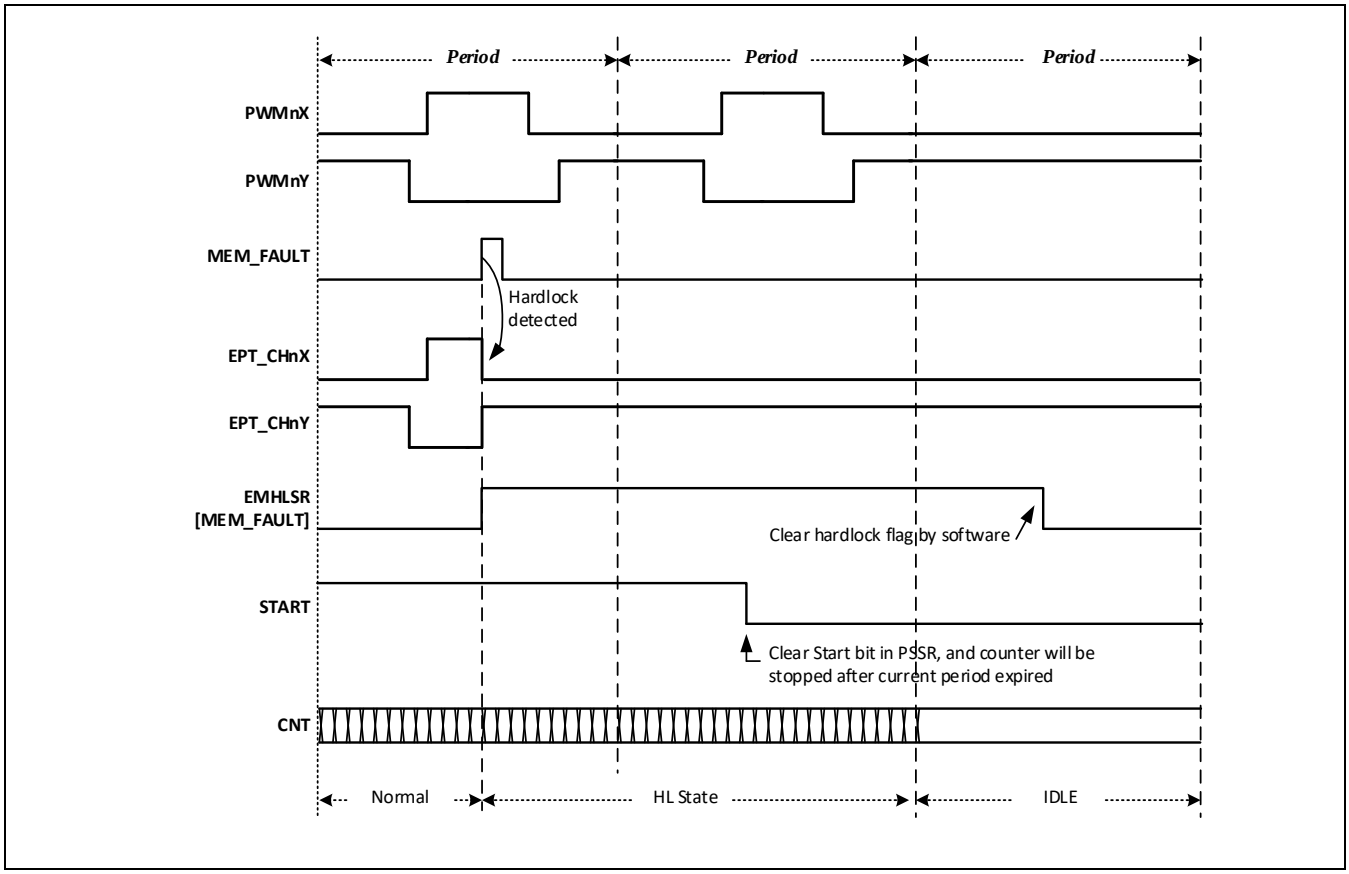


Figure 14-37 硬锁止模式

13.3.7.2 紧急模式中断

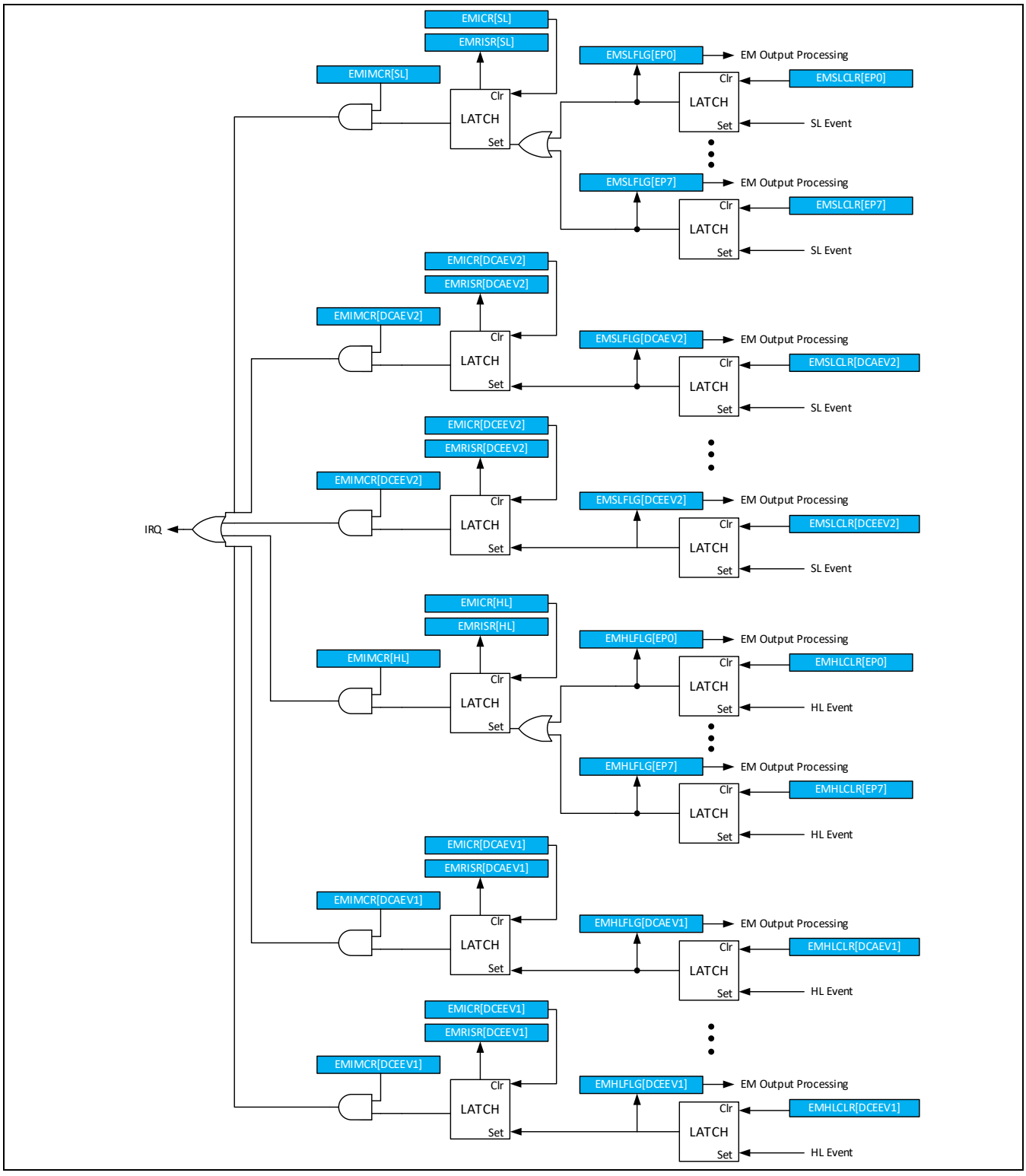


Figure 14-38 紧急模式中断

13.3.8 捕捉模式

13.3.8.1 概述

捕捉模式一般用于如下几个常用的应用：

- 旋转机构的速度测量（比如霍尔传感器）
- 位置传感器的脉冲间隔时间测量
- 脉冲群的周期和占空比测量

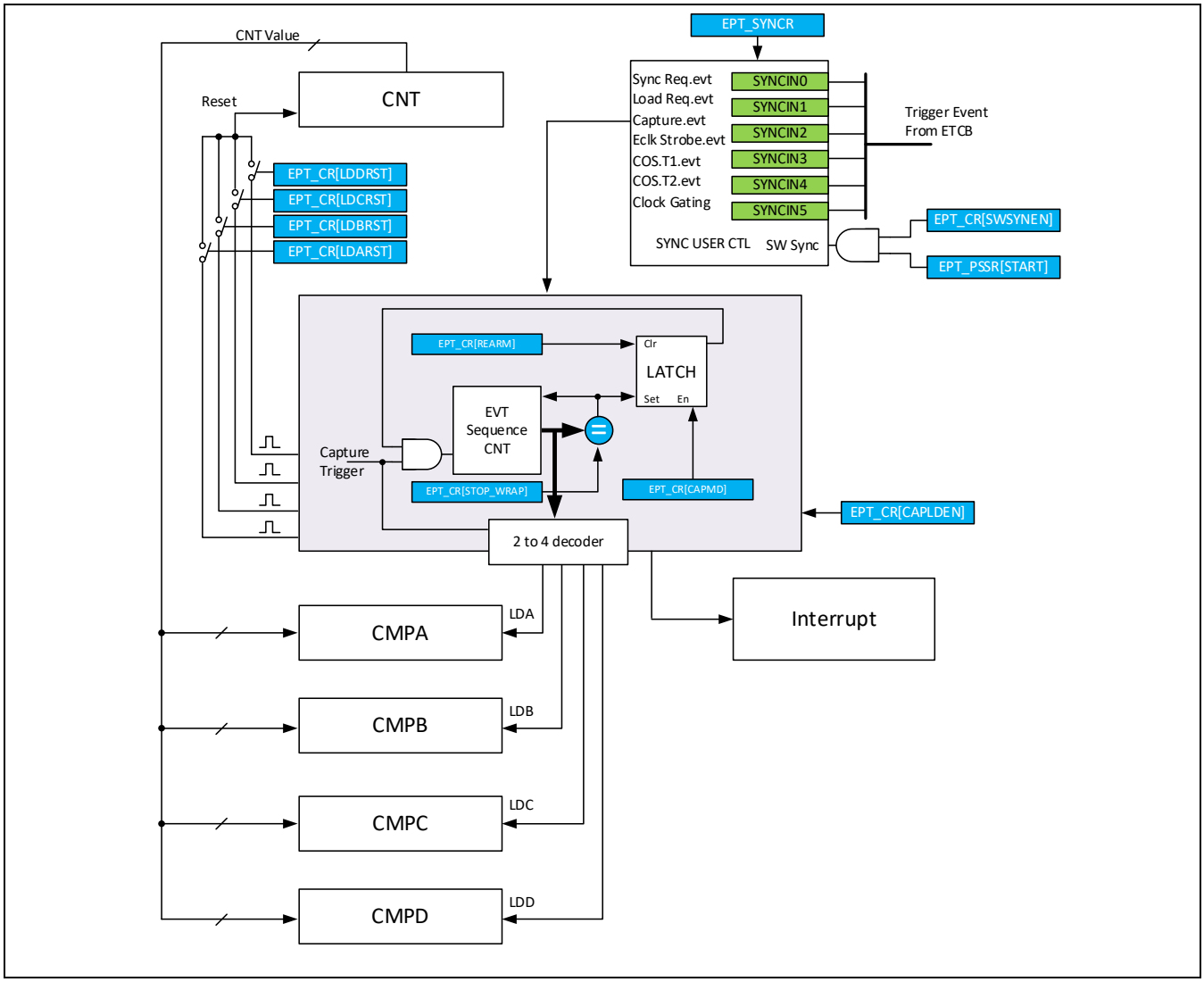


Figure 14-39 捕捉模式结构框图

当CR[WAVE]控制位设置为0时，EPT工作在捕捉模式。在捕捉模式下，捕捉的触发信号通过SYNCIN2端口输入。捕捉模块的主要功能特性如下：

- 最多支持4个捕获事件。捕获事件触发时，计数器值分别存入CMPA、CMPB、CMPC和CMPD（在捕获模

式下，比较值寄存器将作为捕获值存储功能使用）

- 捕获后计数器重置或继续计数

13.3.8.2 捕获事件计数器

在捕捉模式下，捕获值将根据当前捕获事件序列计数器值被存储到相对应的寄存器中。捕获事件计数器在检测到一次捕获触发事件（SYNCIN2上的脉冲输入）时，将自动递增一次。当序列计数器值计数值超出CR[STOP_WRAP]的设置时，计数器自动清零，并重新开始计数。例如：当STOP_WRAP设置为0时，EVT_CNT将一直保持零，所以每次捕获的数据都被保存再CMPA中；当STOP_WRAP设置为2时，EVT_CNT将按照0,1,2计数模式重复，每次捕获的数据分别存入CMPA,CMPB和CMPC。

捕获的计数器值存入目标寄存器和触发相应捕获中断事件，与触发事件发生时，当前的序列计数器值相关。其对应关系如下表所示。

Table 14-7 捕获存储寄存器列表

EVT CNT	Load Target	Trigger Event	Description
0	CMPA	CAP_LD0	Current counter value is loaded into CMPA, CAP_LD0 is triggered
1	CMPB	CAP_LD1	Current counter value is loaded into CMPA, CAP_LD1 is triggered
2	CMPC	CAP_LD2	Current counter value is loaded into CMPA, CAP_LD2 is triggered
3	CMPD	CAP_LD3	Current counter value is loaded into CMPA, CAP_LD3 is triggered

13.3.8.3 两种捕获模式

捕获支持两种工作方式，一次性捕获（One-shot）模式和连续捕获（Continouse）模式。模式设置可以通过CR[CAPMD]控制位进行设置。在一次性捕获模式下，当序列计数器计数到STOP_WRAP后，计数器即停止工作，并禁止对CMPx的再次载入。只有通过软件再次使能后才能恢复（通过对CR[REAMR]控制位置高，进行重新初始化）。在连续模式下，当捕获触发条件满足时，序列计数器超出STOP_WRAP后，会重零开始重新计数，若在新计数器值被捕获时，当前通道捕获标志已经置位，则捕获值覆盖标志位将被置位。捕获标志可以通过软件清楚，或者在读取相应CMP寄存器后，硬件自动清除；捕获值覆盖标志必须通过软件清除。

可以通过设置CR[LDxRST]位，决定相应捕获事件发生时是否需要清除计数器值。

13.3.8.4 捕获模式下的事件

捕获模式的启动事件：

捕获前，需要首先软件使能计数器，或者用SYNCIN0事件清除和启动计数器，这需要通过设置ETCB，连接EPT SYNCIN0的输入事件。

捕获模式的捕获事件：

捕获事件一旦发生，就会触发计数器load操作。捕获事件即SYNCIN2。需要通过设置ETCB，连接EPT SYNCIN2的输入事件。计数器在每次捕获事件触发后，可以自动清零，通过设置CR寄存器中[LDxRST]控制位进行设置。在触发事件发生时，相应的中断标志位被置位，可以通过使能相应的中断开关，控制进入CPU中断。

当同一个外部信号被同时配置为SYNC0 输入和SYNC2 输入时：

- 如果此时计数器在计数，该信号会被视作捕获事件。
- 如果此时计数器没有计数，该信号会被视作计数器启动事件。

13.3.8.5 应用举例

下面有一些例子，说明如何使用捕捉模式。

- **检测TIOA的高电平脉冲宽度，以及TIOB和TIOA的相位 (TIOA和TIAB为任意被预先设置为EXI的GPIO)**

One-shot模式，STOP_WRAP = 2，LDA/BRST = 1。设置TIOB上升沿为SYNC0输入，TIOA上升沿和下降沿都设为SYNC2输入。TIOB上升沿复位计数器，TIOA的下降沿触发第一次load，计数值存入CMPA中。TIOA下一个上升沿触发第二次load，计数值存入CMPB。计数器随即停止计数。此时CMPA的结果为相位差，CMPB的结果为TIOA的高电平宽度。（Figure14-40）

- **检测TIOA上高电平脉冲宽度**

Continous 模式，STOP_WRAP = 1，LDA/BRST = 0。将TIOA设为EXIn（n<16），配置EXIn上升沿为SYNC0输入，同时将TIOA设为EXIm（m>16，扩展EXI），配置EXIm的下降沿为CMPA的SYNC2。第一个TIOA上升沿发生时，SYNC事件发生，计数器被复位。TIOA下降沿触发第一次load，计数值存入CMPA。所以，CMPA的结果为高电平宽度。（Figure14-41）

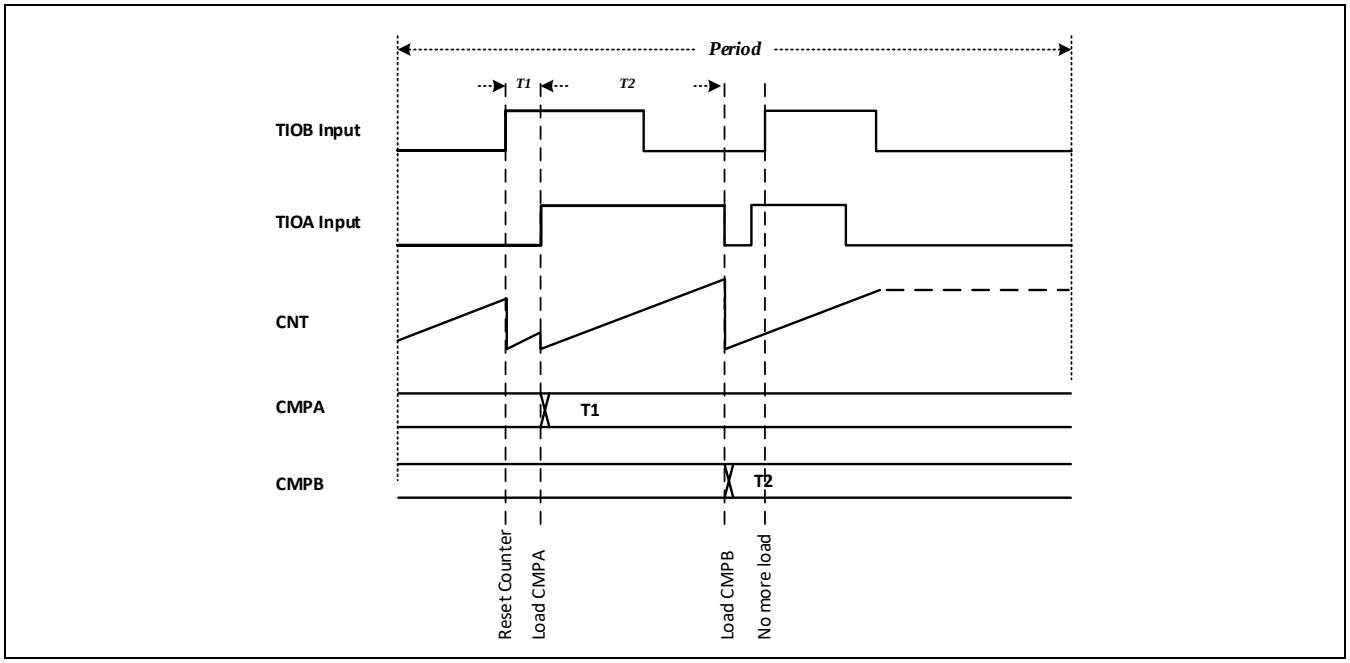


Figure 14-40 测量TIOA和TIOB相位差以及TIOA高电平脉宽

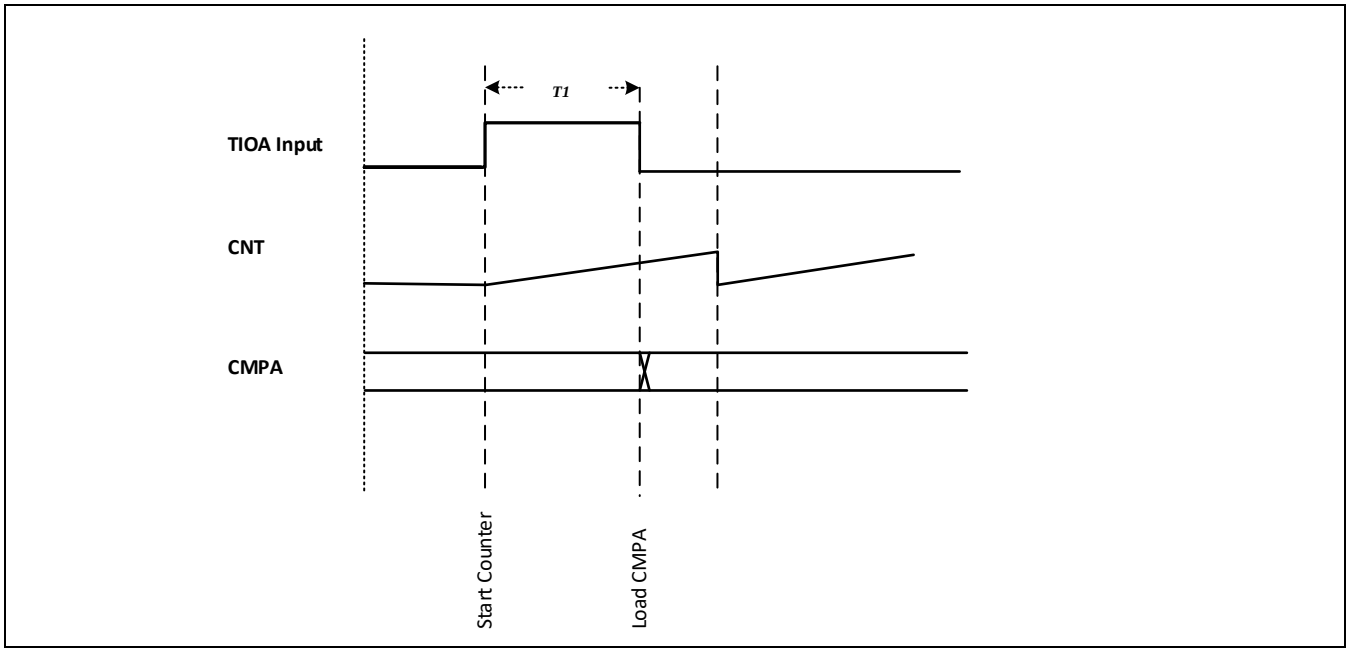


Figure 14-41 测量TIOA的脉冲宽度

13.3.9 单次触发模式

单次触发模式是一种特殊的工作模式，在此模式下，计数器在外部触发事件发生时，只产生一个延时和脉宽可编程的脉冲信号。计数器在启动后只进行一个周期的计数，在周期结束后，计数器Freeze。设置单次触发模式，可以通过寄存器CR[OPM]控制位进行设置。在计数器Freeze状态下，计数器保持当前的计数值，直到有新的触发条件被检测到。

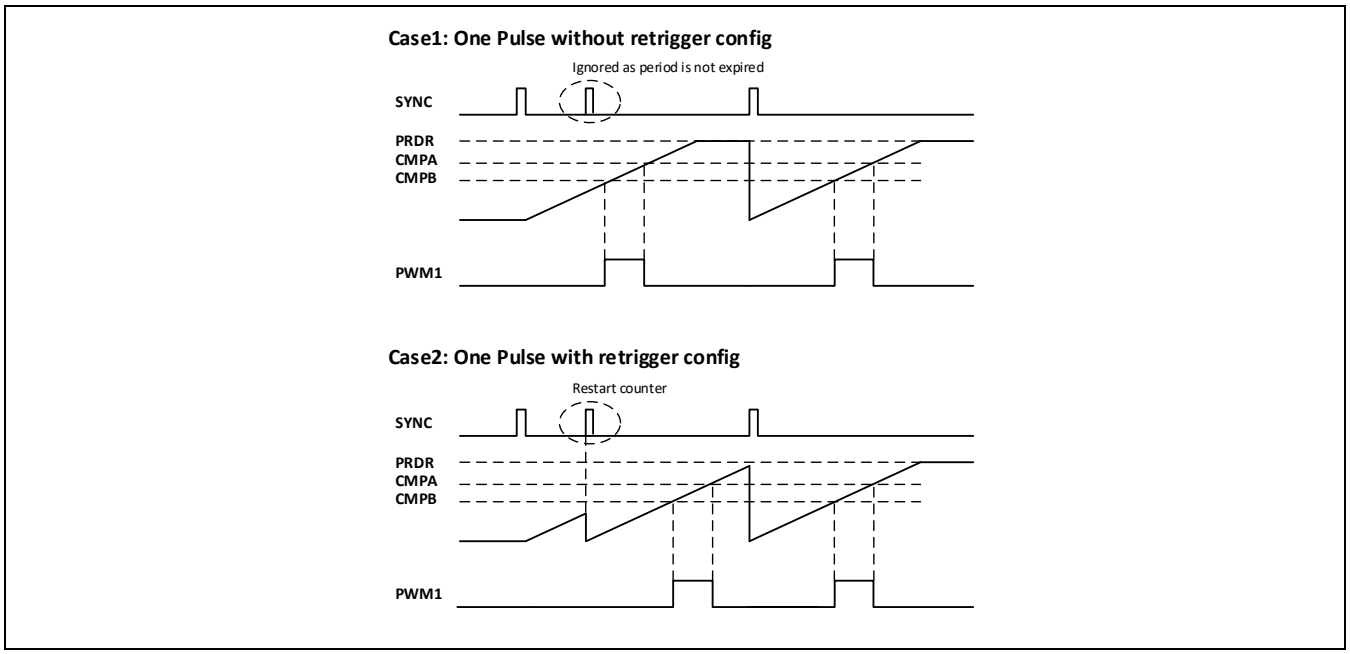


Figure 14-40 单次触发

单次触发模式工作时，缺省情况下，无论当前计数周期是否结束，当有新的触发被检测到时，计数器将开始重新计数。在某些应用中，需要在被触发后的一个计数周期内禁止新的触发，以保证得到一个完整的周期波形输出。在这种条件下，可以通过将外部触发信号通过窗口滤波器滤波，屏蔽指定时间内的触发输入。

13.3.10 同步触发（输入）

同步触发和事件触发功能用于在多个外设间通过硬件自动耦合同步不同外设的工作。EPT通过同步输入接口接收来自于其他外设的触发信号，不同的触发端口对应独立的同步任务。当相应输入接口被触发，相对应的同步任务即被激活。

13.3.10.1 同步触发输入接口

EPT支持模块间的同步触发功能，可以支持的触发功能包括如下几种：

- 重置和启动计数器
- 寄存器的更新（从Shadow寄存器更新到Active寄存器）
- 当前计数器值捕捉
- 计数器值递增或递减一个计数值
- 触发改变PWM的输出状态

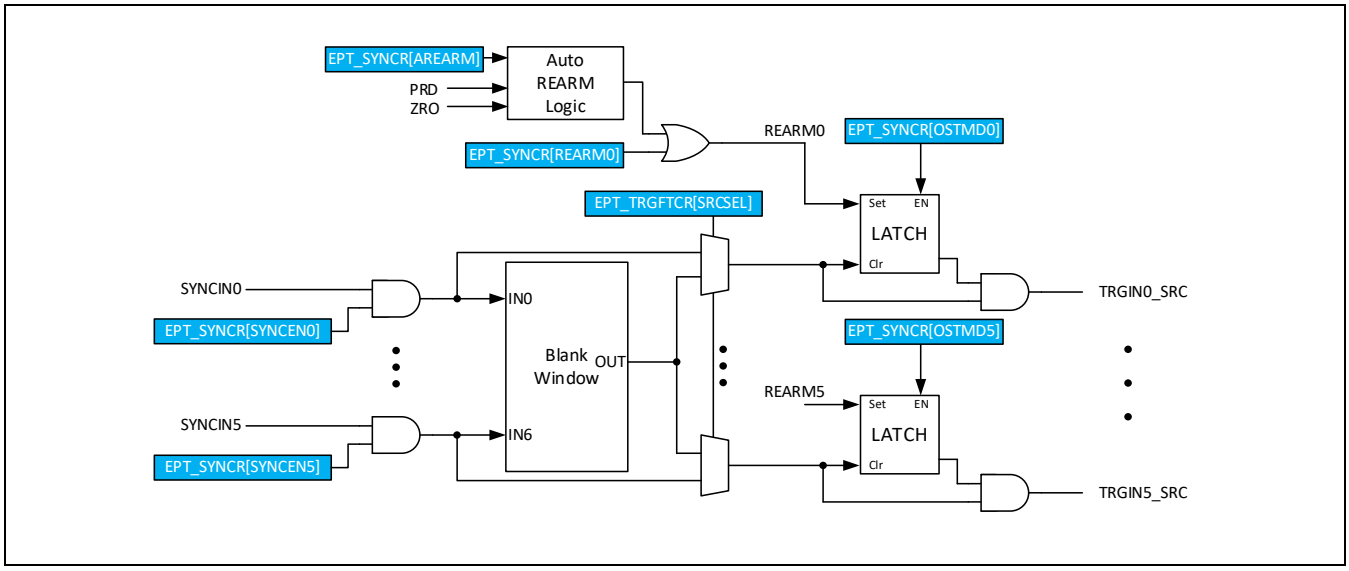


Figure 14-41 同步触发输入

每一个独立触发源被定义为SYNCIN端口，通过SYNCR寄存器可以独立控制每个触发源的使能。触发源的输入为ETCB模块的输出，通过ETCB可以定义某个外设作为当前SYNCIN端口的触发信号源。具体配置参考ETCB章节。在触发输入接口中，包含一个事件滤波器，可以选择一个事件输入端口作为滤波器输入，对该事件进行滤波处理。

每一个触发端口可以工作在两种工作模式：连续触发或者单次触发模式。在单次触发模式下，只运行一次触发

发生，在检测到发生一次触发后，该端口将被禁止，直到软件重置该端口（REARM）后，才允许新的触发发生。重置端口也可以通过硬件自动完成，在设置SYNCR[AREARM]后，在周期结束或者开始时，硬件会自动重置REARM，以保证在一个周期内只发生一次触发。

13.3.10.2 同步触发事件

SYNC触发：重置和启动计数器（SYNCIN0）

当该端口被触发，下列动作将被同时执行：

- 时基计数器（CNT）被重置。计数器的重置数据被存放在PHSR寄存器中，当该触发条件发生时，时基计数器在下一个TCLK将从PHSR的设置值开始计数。在递增递减模式下，计数器的计数方向同样将根据PHSR[PHSDIR]的设置作出变化。
- 时钟分频器重置。当该触发条件发生时，时钟分频器将重新开始计数
- 所有具有Shadow寄存器的控制寄存器将自动从Shadow更新Active寄存器

LOAD触发：寄存器的更新（SYNCIN1）

当该端口被触发，所有具有Shadow寄存器的控制寄存器将自动从Shadow更新Active寄存器

CAPTURE触发：计数器值捕捉（SYNCIN2）

当该端口被触发，将触发捕获事件。只有在CR[WAVE]设置为捕获模式时，且CR[CAPLDEN]控制位使能时，该触发事件才能被捕获模块检测到。

CNT增减触发：计数器值递增或递减一个计数值（SYNCIN3）

当该端口被触发，计数器将根据当前计数方向，自动增加或减少一个计数值。只有在CEDR[CSS]控制位选择SYNCIN3时，该端口的触发才会被计数器检测到。

COS(Change Output Status)触发：PWM输出状态改变（SYNCIN4/5）

SYNCIN4和SYNCIN5用于产生内部T1和T2触发事件。可以通过设置AQTSCR[T1SEL]控制位选择SYNCIN4作为T1事件，通过设置AQTSCR[T2SEL]控制位选择SYNCIN5作为T2事件。

13.3.10.3 事件滤波器

在同步触发输入接口中，有一个事件滤波器，事件滤波器是一个时间窗口滤波器，它可以在一个特定的窗口周期内阻止输入信号的通过，或者只有在窗口内允许信号通过，从而实现去除干扰事件的目的。例如当模拟比较器的输出作为外部触发源时，可以滤除由于模拟比较器的抖动而产生的错误触发。可以选择任意一个SYNCIN端口作为事件滤波器的输入。

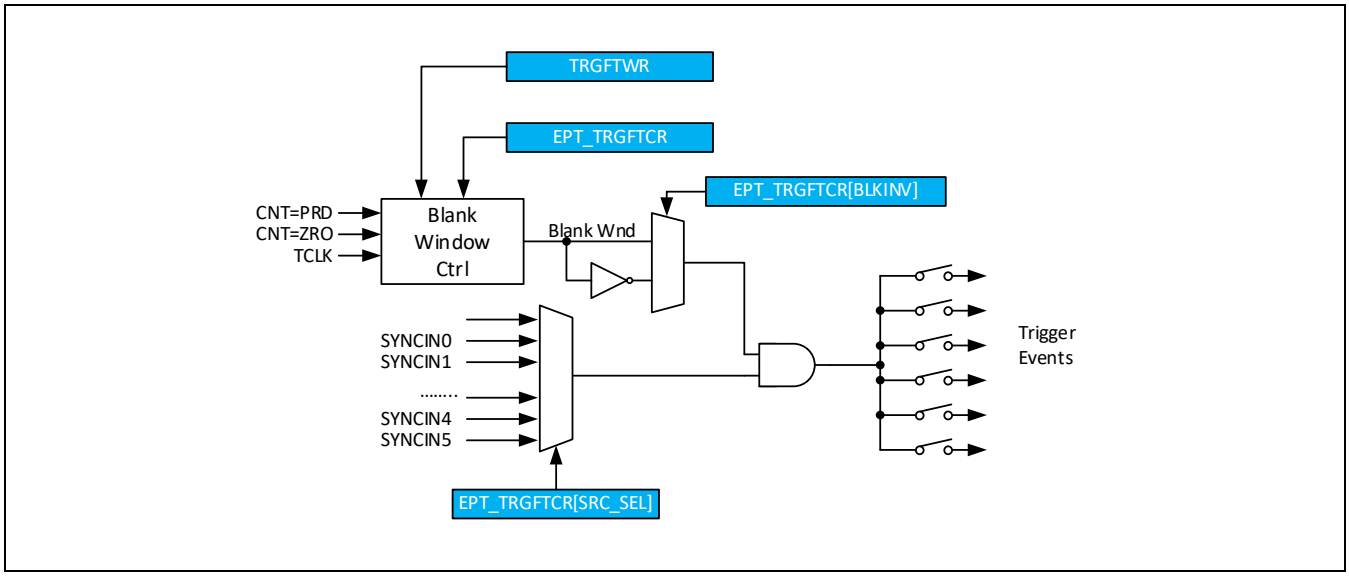


Figure 14-42 滤波器框图

如果使能窗逻辑被打开，在使能窗口内，或者窗口外，将禁止输入事件通过。窗口的激活时间可以设置为CNT=PRD，CNT=ZRO或者两个条件都可以（通过配置TRGFTCR[ALIGNMD]）。窗口的延时和宽度可以通过TRGFWR进行设置。

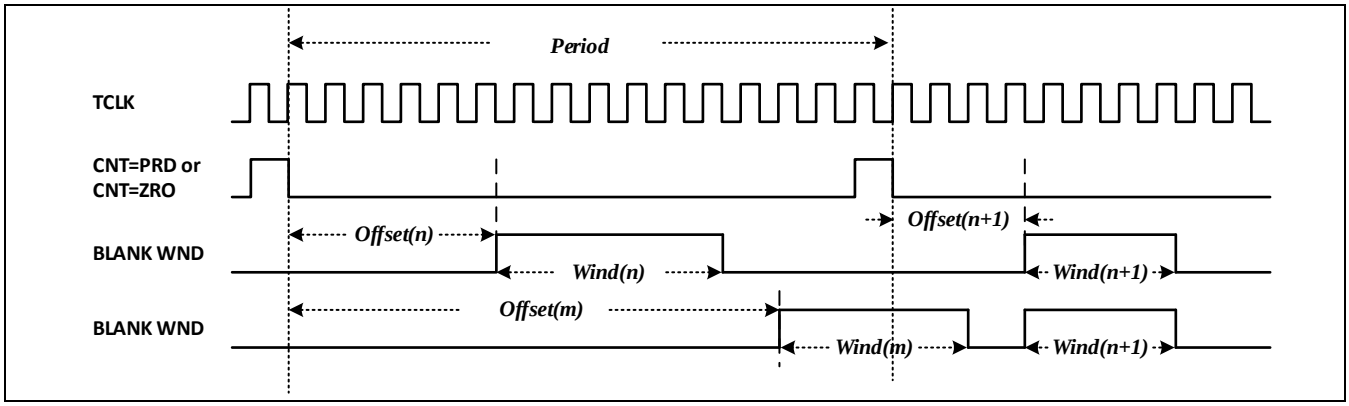


Figure 14-43 滤波器时序

13.3.11 事件触发（输出）

13.3.11.1 同步触发输出接口

EPT的事件输出接口，可用于产生对其他外设的任务的触发信号。事件触发输出接口支持4路事件触发输出，每个EPT中断对应一个事件输出端口。可以通过EVTRG控制寄存器选择EPT中的任意一个事件作为每个中断的触发信号，同时中断触发信号可以通过TRGxOE控制位使能输出到其他外设，作为触发信号。

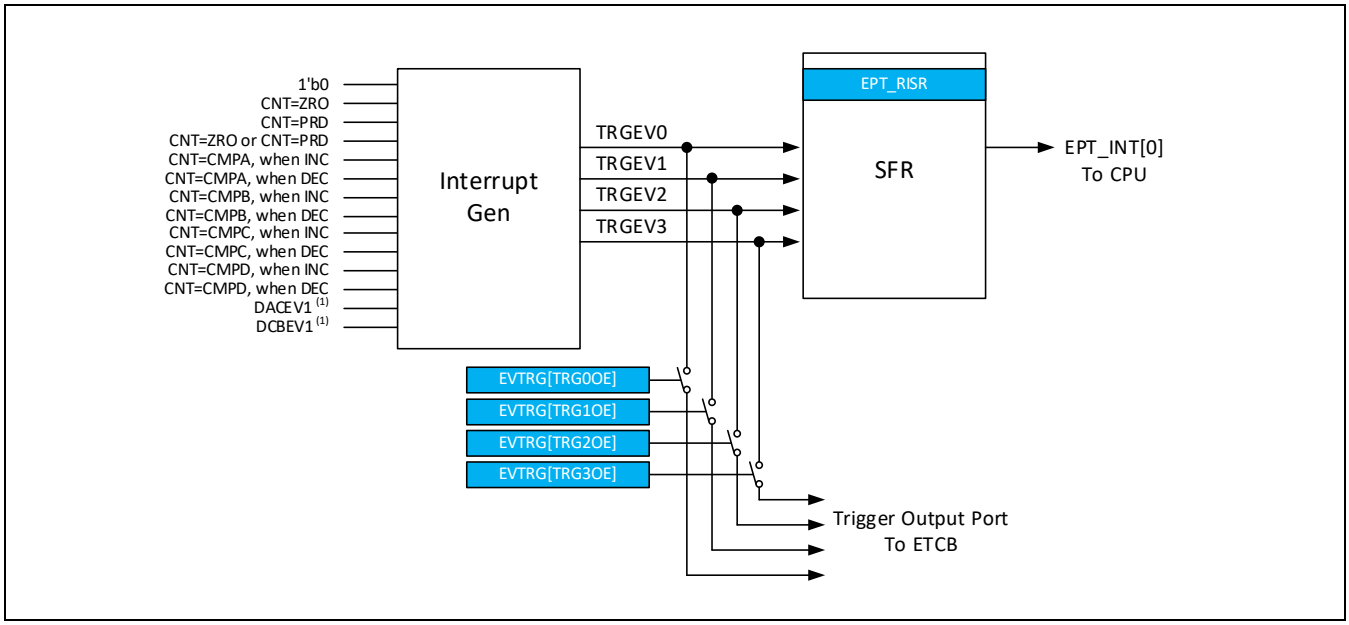


Figure 14-44 同步触发输出

13.3.11.2 事件计数和中断

触发中断基于各种触发事件，可以支持对触发事件计数的基础上决定是否产生中断请求。中断触发控制支持如下几种配置：

- 每次发生触发事件产生中断
- 每N次触发事情产生一次中断，N最大支持到15

中断模块一共支持4个中断源，每一个中断触发都支持事件计数器。中断的具体触发条件，可以通过EVTRG寄存器进行选择。通过配置EVPS可以配置每个中断事件计数器在计数到多少时产生一个中断请求。当中断发生次数等于EVPS[TRGEVxPRD]控制位设置值时，将产生一次中断触发。计数器最大支持15个事件的计数，在中断请求发生后计数器将自动清除。当前已经计数的事件个数可以通过EVPS[TRGEVxCNT]进行读取。CNT计数器具有Shadow功能，在缺省设置下，对CNT的读写时，操作对象是Shadow寄存器，Shadow寄存器的值会在同步事件发生时，或者计数器值等于PRD设置值时自动载入到活动寄存器中。当Shadow模式被禁止时，对CNT的操作直接影响活动寄存器的值。

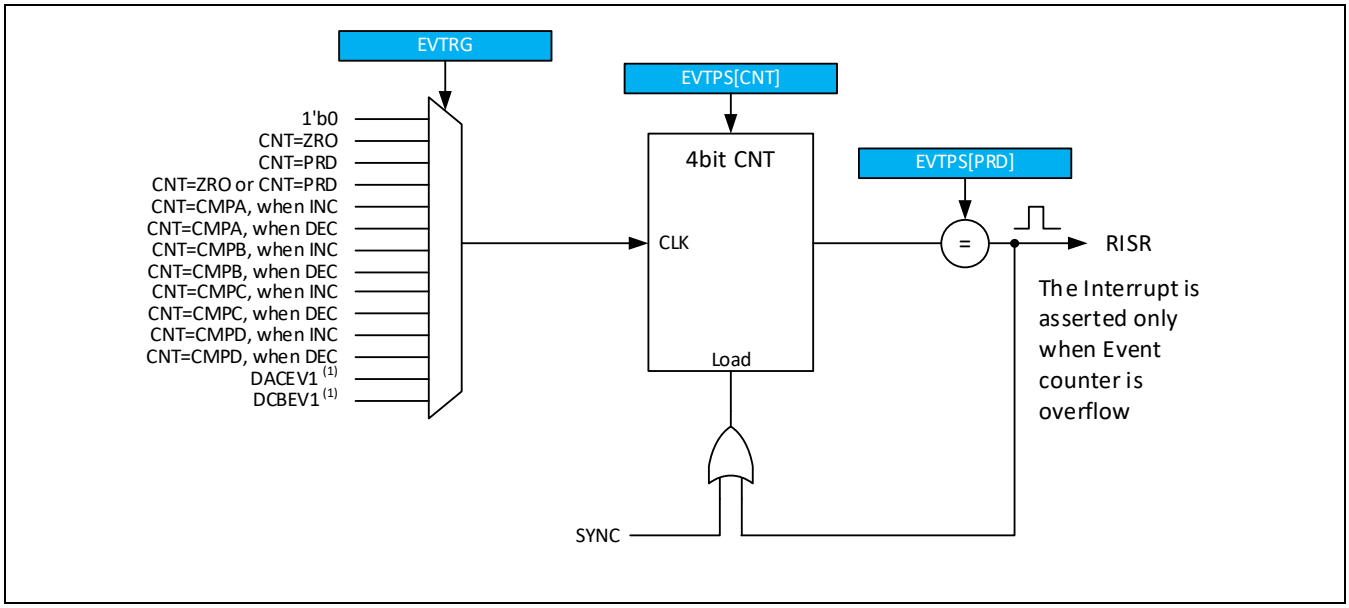


Figure 14-45 事件计数器

13.4 寄存器说明

13.4.1 寄存器表

- Base Address: EPT0 – 0x4005_9000

Register	Offset	Description	Reset Value	PROT
CEDR	0x0000	ID & Clock Enable/Disable Register	0xBE980000	
RSSR	0x0004	RESET/START/STOP Register	0x00000000	Y
PSCR	0x0008	Counter Clock Prescaler Register	0x00000000	
CR	0x000C	General Control Register	0x00010010	
SYNCR	0x0010	Synchronization Control Register	0x00000000	Y
GLDCR	0x0014	Global Load Control Register	0x00000000	
GLDCFG	0x0018	Global Load Configuration Register	0x00000000	
GLDCR2	0x001C	Global Load Control Register2	0x00000000	Y
RSVD	0x0020	Reserved	0x00000000	
PRDR	0x0024	Period Register	0x00000000	
PHSR	0x0028	Phase Control Register	0x00000000	
CMPA	0x002C	Compare A Register	0x00000000	
CMPB	0x0030	Compare B Register	0x00000000	
CMPC	0x0034	Compare C Register	0x00000000	
CMPD	0x0038	Compare D Register	0x00000000	
CMPLDR	0x003C	Compare Data Load Control Register	0x00002490	
CNT	0x0040	Counter Register	0x00000000	
AQLDR	0x0044	Action Qualifier Load Control Register	0x00000024	
AQCR1	0x0048	Action Qualifier Control Register 1	0x00000000	
AQCR2	0x004C	Action Qualifier Control Register 2	0x00000000	
AQCR3	0x0050	Action Qualifier Control Register 3	0x00000000	
AQCR4	0x0054	Action Qualifier Control Register 4	0x00000000	
AQTSCR	0x0058	Action Qualifier T Event Source Selection Register	0x00000000	
AQOSF	0x005C	Action Qualifier One Shot Force Register	0x00010000	
AQCSF	0x0060	Action Qualifier Continuous Force Register	0x00000000	
DBLDR	0x0064	Dead-Band Load Control Register	0x00000492	
DBCR	0x0068	Dead-Band Control Register	0x00000000	
DPSCR	0x006C	Dead-Band Delay Control Clock Prescaler	0x00000000	
DBDTR	0x0070	Dead-Band Rising Edge Delay Time Regiser	0x00000000	
DBDTF	0x0074	Dead-Band Falling Edge Delay Time Regiser	0x00000000	

Register	Offset	Description	Reset Value	PROT
CPCR	0x0078	Chop Control Register	0x00000000	
EMSRC	0x007C	Emergency Condition Input Config Register1	0x00000000	Y
EMSRC2	0x0080	Emergency Condition Input Config Register2	0x00000000	Y
EMPOL	0x0084	Emergency Condition Input Poloarity Register	0x00000000	Y
EMECR	0x0088	Emergency Condition Enable Register	0x00400000	Y
EMOSR	0x008C	Emergency Condition Output Status Set Register1	0x00000000	Y
RSVD	0x0090	Reserved	0x00000000	
EMSLSR	0x0094	Emergency Condition Soft-lock Status Register	0x00000000	
EMSLCLR	0x0098	Emergency Condition Soft-lock Clear Register	0x00000000	
EMHLSR	0x009C	Emergency Condition Hard-lock Status Register	0x00000000	
EMHLCLR	0x00A0	Emergency Condition Hard-lock Clear Register	0x00000000	
EMFRCR	0x00A4	Emergency Condition Software Force Register	0x00000000	Y
EMRISR	0x00A8	Emergency Condition Raw Interrupt Status Register	0x00000000	
EMMISR	0x00AC	Emergency Condition Masked Interrupt Status Register	0x00000000	
EMIMCR	0x00B0	Emergency Condition Interrupt Masking Control Register	0x00000000	
EMICR	0x00B4	Emergency Condition Interrupt Clear Register	0x00000000	
TRGFTCR	0x00B8	Digital Compare Filter Control Register	0x00000000	
TRGFTWR	0x00BC	Digital Compare Filter Window Register	0x00000000	
EVTRG	0x00C0	Event Generation Control Register	0x00000000	
EVPS	0x00C4	Event Counter Prescaler	0x00000000	
EVCNTINIT	0x00C8	Event Counter Initial Value	0x00000000	
EVSWF	0x00CC	Event Counter Software Trigger Register	0x00000000	
RISR	0x00D0	Raw Interrupt Status Register	0x00000000	
MISR	0x00D4	Masked Interrupt Status Register	0x00000000	
IMCR	0x00D8	Interrupt Masking Control Register	0x00000000	
ICR	0x00DC	Interrupt Clear Register	0x00000000	
RSVD	0x00E0	Reserved	0x00000000	
RSVD	0x00E4	Reserved	0x00000000	
PROT	0x00E8	Protection Register	0x00000000	

NOTE: 寄存器 PROT 列表示该寄存器受 PROT 保护，只有 PROT 保护解除后，才能对该寄存器进行更新操作。

13.4.2 CEDR (ID和时钟控制寄存器)

- Address = Base Address +0x0000, Reset Value = 0xBE98_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																FLTCKPRS								RSVD	SHDWSTP	TINSEL		CSS	DBGEN		CLKEN
1	0	1	1	1	1	1	0	1	0	0	1	1	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	R	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CLKEN	[0]	RW	时基计数器的时钟使能控制。 0h：计数器计数时钟禁止。 1h：计数器计数时钟使能。
DBGEN	[2:1]	RW	调试使能控制。调试使能时，在CPU被调试器挂起时，时基计数器的计数时钟同时也被挂起。 0h：调试禁止 1h：调试使能，PWM输出高阻 其他：调试使能，PWM输出保持
CSS	[3]	RW	计数器时钟源选择位。 0h：PCLK 1h：由SYNCIN3控制 其他：保留
TINSEL	[5:4]	RW	TIN输入源选择控制位。TIN可以作为计数器计数时钟的使能控制，或者作为载波输出模式下的载波。 0h：禁止TIN输入 1h：BT0_OUT作为TIN的输入 2h：BT1_OUT作为TIN的输入 3h：保留
SHDWSTP	[6]	RW	START控制位的Shadow功能使能控制。START置位不受此位控制，清除时受此位控制。当选择Shadow模式时，START控制位在周期结束时清除。 0h：Shadow模式 1h：Immediate模式

FLTCKPRS	[15:8]	RW	CGFLT数字滤波器的时钟分频控制。 数字滤波器的时钟频率为PCLK/(FLTCKPRS+1)
IDCODE	[31:16]	RW	当前EPT模块的版本信息。

13.4.3 RSSR (启停控制寄存器)

- 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																SRR				RSVD								CNTDIR	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	W	W	W	W	R	R	R	R	R	R	R	R	R	R	R	RW

Name	Bit	Type	Description
START	[0]	RW	计数器启动控制位。 0h：当写‘0’时，停止计数器 1h：当写‘1’时，启动计数器 当对START位进行读取时，返回当前计数器工作状态 0h：计数器处于IDLE状态 1h：计数器正在工作 当CR[SWSYNEN]控制位为低时，START控制位用于控制EPT的启动，当EPT启动后，再次写入START将被忽略；当CR[SWSYNEN]控制位为高时，START控制位用于软件触发同步事件，每次对START的写入，会产生一次外部Sync事件（等同于SYNCR中的SYNCIN0触发）。
CNTDIR	[3]	R	当前计数器计数方向状态。 0h：当前计数器方向为递增 1h：当前计数器方向为递减
SRR	[15:12]	W	软件复位控制位。 当对当前控制位写入‘0x5’时，TIMER模块会被复位。复位后，所有寄存器都恢复为RESET状态。

NOTE: 该寄存器受 REGPROT 保护，需要先解锁，才能写入。

13.4.4 PSCR (时钟分频控制寄存器)

- 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																PSC															
0	0	0	0	0	0	0	0	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
PSC	[15:0]	RW	时钟分频控制。 TCLK作为时基模块的计时时钟和工作时钟。TCLK的时钟从PCLK分频得到。此寄存器具有Shadow寄存器，可通过CR[PSCLD]设置载入的条件。 TCLK的频率：$F_{TCLK} = F_{PCLK} / (PSC+1)$

13.4.5 CR (控制寄存器，捕捉模式 WAVE=0)

- Address = Base Address +0x000C, Reset Value = 0000_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					LDDRST	LDCRST	LDBRST	LDARST	STOP_WRAP		CAPMD	REARM	WAVE	PSCLD		CGFLT			CGSRC		FLTIPSCLD	BURST	CAPLDEN	RSVD		PRDLD		IDLEST	SWSYNEN	CNTMD	
0	0	0	0	0	0	0	0	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	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	W	RW	RW	R	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CNTMD	[1:0]	RW	计数模式设置。 计数模式一般只设置一次，并且在计数过程中不做改变。如果计数模式被改变，变化将发生在下一个TCLK的边沿，并且基于上一个计数器值进行递增或者递减。 00b：递增模式 01b：递减模式 （该模式不支持输出满幅波形） 10b：递增递减模式 11b：保留
SWSYNEN	[2]	RW	软件使能同步触发使能控制（RSSR 中 START 控制位）。 0h：设置SW START控制只用于启动。 1h：设置SW START控制用于启动和以产生一次外部触发的方式重新启动。
IDLEST	[3]	RW	波形输出被停止时，GPIO 输出控制 0h：GPIO高阻输出 1h：PWM 信号低电平（此 PWM 信号为内部 PWMx 信号，GPIO 输出电平取决于是否使能死区控制，以及死区控制的配置）
PRDLD	[5:4]	RW	PRDR活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值。 00b：PRDR活动寄存器更新发生在周期结束(PEND) 01b：PRDR活动寄存器更新发生在外部LOAD触发或SYNC触发时 10b：PRDR活动寄存器更新发生在计数器值等于零和外部LOAD触发或SYNC触发时 11b：立即更新，所有对PRDR操作直接作用于活动寄存器 [1]

CAPLDEN	[8]	RW	CMPA和CMPB在捕捉事件触发时，载入使能控制。此控制位在禁止对CMP寄存器载入时，并不影响捕捉事件CEV的触发。 0h：禁止对CMP寄存器的捕获载入 1h：使能对CMP寄存器的捕获载入
BURST	[9]	RW	群脉冲模式。 0h：禁止群脉冲模式 1h：使能群脉冲模式
FLTIPSCLD	[10]	W	数字滤波器初始化控制。对该控制写'1'可以初始化数字滤波器计数器，计数器值被初始化为CEDR[FLTCKPRS]中的设置值。 0h：无效 1h：执行初始化
CGSRC	[12:11]	RW	群脉冲模式下，时钟门控的输入源选择。 0h：CHAX作为CG的输入源 1h：CHBX作为CG的输入源 2h：TIN作为CG的输入源 3h：保留
CGFLT	[15:13]	RW	门控输入数字滤波控制。此控制定义了滤波器监测的步数，只有连续N次监测结果一致时，滤波器才输出有效的电平翻转。滤波器的采样时钟频率在通过CKS控制位定义。 000b：Bypass 001b：N = 2 010b：N = 3 011b：N = 4 100b：N = 6 101b：N = 8 110b：N = 16 111b：N = 32
PSCLD	[17:16]	RW	PSCR活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值。 00b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 01b：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 10b：当CNT=ZRO或者PRD时，Shadow寄存器载入到Active寄存器中 11b：不进行载入
WAVE	[18]	RW	EPT工作模式选择。 0h：捕捉模式

			1h：波形发生模式
REARM	[19]	W	重置CAPTURE 捕捉事件计数器控制。 0h：无效 1h：重置捕捉计数器 重置时，捕捉事件计数器被清零，自动打开CAPLDEN。捕捉事件计数器周期通过STOP_WRAP进行设置。
CAPMD	[20]	RW	捕捉模式设置。在一次性捕捉模式下，当捕捉事件计数器等于STOP_WRAP设置值时，后续的捕捉事件将不能触发捕捉。必须通过软件REARM后才能继续触发捕捉。 0h：连续捕捉模式 1h：一次性捕捉模式
STOP_WRAP	[22:21]	RW	Capture模式下，捕获事件计数器周期设置值。
LDARST	[23]	RW	CMPA捕捉载入后，计数器值计数状态控制位。 1h：CMPA触发后，计数器值进行重置 0h：CMPA触发后，计数器值不进行重置
LDBRST	[24]	RW	CMPB捕捉载入后，计数器值计数状态控制位。 1h：CMPB触发后，计数器值进行重置 0h：CMPB触发后，计数器值不进行重置
LDCRST	[25]	RW	CMPC捕捉载入后，计数器值计数状态控制位。 1h：CMPC触发后，计数器值进行重置 0h：CMPC触发后，计数器值不进行重置
LDDRST	[26]	RW	CMPD捕捉载入后，计数器值计数状态控制位。 1h：CMPD触发后，计数器值进行重置 0h：CMPD触发后，计数器值不进行重置

注意 [1]：如果更新的周期值比更新前小，且立即更新发生时计数器已经超过更新的周期值，计数器将继续计数直到溢出，然后更新才会生效。

13.4.6 CR (控制寄存器，波形输出模式：WAVE=1)

- Address = Base Address +0x000C, Reset Value = 0001_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													WAVE	PSCLD		CGFLT			CGSRC		FLTIPSCLD	BURST	RSVD	PHSEN	OPM	PRDLD		IDLEST	SWSYNEN	CNTMD	
0	0	0	0	0	0	0	0	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	R	R	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW	R	RW	RW	RW

Name	Bit	Type	Description
CNTMD	[1:0]	RW	计数模式设置。 计数模式一般只设置一次，并且在计数过程中不做改变。如果计数模式被改变，变化将发生在下一个TCLK的边沿，并且基于上一个计数器值进行递增或者递减。 00b：递增模式 01b：递减模式 10b：递增递减模式 11b：保留
SWSYNEN	[2]	RW	软件使能同步触发使能控制（RSSR 中 START 控制位）。 0h：设置SW START控制只用于启动。 1h：设置SW START控制用于启动和以产生一次外部触发的方式重新启动。
IDLEST	[3]	RW	波形输出被停止时，GPIO 输出控制 0h：GPIO高阻输出 1h：PWM 信号低电平（此 PWM 信号为内部 PWMx 信号，GPIO 输出电平取决于是否使能死区控制，以及死区控制的配置）
PRDLD	[5:4]	RW	PRDR活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值。 00b：PRDR活动寄存器更新发生在周期结束(PEND) 01b：PRDR活动寄存器更新发生在外部LOAD触发或SYNC触发时 10b：PRDR活动寄存器更新发生在计数器值等于零和外部LOAD触发或SYNC触发时 11b：立即更新，所有对PRDR操作直接作用于活动寄存器
OPM	[6]	RW	计数器单次触发工作模式选择。

			0h：连续计数工作模式 1h：单次触发工作模式 其他：保留
PHSEN	[7]	RW	PHSR使能控制位，当控制位有效时，计数器将在启动时被初始化为PHSR中的设置值。 0h：禁止通过PHSR初始化 1h：使能通过PHSR初始化
BURST	[9]	RW	群脉冲模式。 0h：禁止群脉冲模式 1h：使能群脉冲模式
FLTIPSCLD	[10]	RW	数字滤波器初始化控制。对该控制写'1'可以初始化数字滤波器计数器，计数器值被初始化为CEDR[FLTCKPRS]中的设置值。 0h：无效 1h：执行初始化
CGSRC	[12:11]	RW	群脉冲模式下，时钟门控的输入源选择。 0h：CHAX作为CG的输入源 1h：CHBX作为CG的输入源 2h：TIN作为CG的输入源 3h：保留
CGFLT	[15:13]	RW	门控输入数字滤波控制。此控制定义了滤波器监测的步数，只有连续N次监测结果一致时，滤波器才输出有效的电平翻转。滤波器的采样时钟频率在通过CKS控制位定义。 000b：Bypass 001b：N = 2 010b：N = 3 011b：N = 4 100b：N = 6 101b：N = 8 110b：N = 16 111b：N = 32
PSCLD	[17:16]	RW	PSCR活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值。 00b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 01b：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 10b：当CNT=ZRO或者PRD时，Shadow寄存器载入到Active寄存器中

			11b：不进行载入
WAVE	[18]	RW	EPT工作模式选择。 0h：捕捉模式 1h：波形发生模式

13.4.7 SYNCR (同步控制寄存器)

- 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
AREARM		TRGO1SEL			TRGO0SEL			TXREARM0		REARM5	REARM4	REARM3	REARM2	REARM1	REARM0	RSVD		OSTMD5	OSTMD4	OSTMD3	OSTMD2	OSTMD1	OSTMD0	RSVD		SYNCEN5	SYNCEN4	SYNCEN3	SYNCEN2	SYNCEN1	SYNCEN0
0	0	0	0	0	0	0	0	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	R	R	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
SYNCENx	[5:0]	RW	外部同步触发使能控制。 0：禁止当前触发输入通道 1：使能当前触发输入通道 SYNCIN0：外部Sync事件 SYNCIN1：Load触发 SYNCIN2：Capture触发事件 SYNCIN3：CNT增减一拍触发事件 SYNCIN4：外部COS事件（用于PWM波形输出控制） SYNCIN5：外部COS事件（用于PWM波形输出控制）
OSTMDx	[13:8]	RW	同步触发模式选择。 0h：连续触发模式 1h：一次性同步触发模式 当该输入通道被设置为一次性同步触发模式后，在一次触发事件被检测到后，该通道将不允许后续的触发事件通过，直到被软件重置（REARM）后才允许新的触发事件通过。
REARMx	[21:16]	RW	在一次性同步触发模式下，软件重置当前通道状态控制位。 当读取时，返回当前通道状态 0h：允许触发 1h：已经检测到触发，不允许后续触发 当写入时， 0h：无效 1h：清除当前通道状态，并允许新的触发
TXREARM0	[23:22]	RW	Tx 信号触发 SYNCIN0 的 REARM

			0：禁止Tx硬件自动REARM 1：T1发生触发，自动REARM SYNCIN0通道 2：T2发生触发，自动REARM SYNCIN0通道 3：T1 或者 T2 发生触发，自动 REARM SYNCIN0 通道
TRGO0SEL	[26:24]	RW	输入触发通道直通作为TRGSRC0的ExtSync条件的选择。只有当EVTRG寄存器中TRGSRC0控制位选择为ExtSync条件时有效。 0h：选择SYNCIN0作为TRGSRC0的ExtSync触发 1h：选择SYNCIN1作为TRGSRC0的ExtSync触发 2h：选择SYNCIN2作为TRGSRC0的ExtSync触发 3h：选择SYNCIN3作为TRGSRC0的ExtSync触发 4h：选择SYNCIN4作为TRGSRC0的ExtSync触发 5h：选择SYNCIN5作为TRGSRC0的ExtSync触发 其他：保留
TRGO1SEL	[29:27]	RW	输入触发通道直通作为TRGSRC1的ExtSync条件的选择。只有当EVTRG寄存器中TRGSRC1控制位选择为ExtSync条件时有效。 0h：选择SYNCIN0作为TRGSRC1的ExtSync触发 1h：选择SYNCIN1作为TRGSRC1的ExtSync触发 2h：选择SYNCIN2作为TRGSRC1的ExtSync触发 3h：选择SYNCIN3作为TRGSRC1的ExtSync触发 4h：选择SYNCIN4作为TRGSRC1的ExtSync触发 5h：选择SYNCIN5作为TRGSRC1的ExtSync触发 其他：保留
AREARM	[31:30]	RW	硬件自动REARM控制位。 0：禁止硬件自动REARM 1：CNT = ZRO时，自动REARM 2：CNT = PRD时，自动REARM 3：CNT = ZRO or CNT = PRD时，自动REARM

NOTE: 该寄存器受 REGPROT 保护，需要先解锁，才能写入。

13.4.8 GLDCR (全局载入控制寄存器)

- 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																			GLDCNT			GLDPRD			RSVD	OSTMD	GLDMD				GLDEN
0	0	0	0	0	0	0	0	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	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
GLDEN	[0]	RW	全局的Shadow到Active寄存器载入控制。 0：使用独立的单个配置（在各个寄存器中LDMD控制位分别指派的载入控制） 1：使用GLDMD中的设置，其他设置被屏蔽
GLDMD	[4:1]	RW	全局载入触发事件选择。 0h：CNT = ZRO 1h：CNT = PRD 2h：CNT = ZRO or CNT = PRD 3h：CNT = ZRO or 外部LOAD触发或SYNC触发 4h：CNT = PRD or 外部LOAD触发或SYNC触发 5h：CNT = ZRO or CNT = PRD or 外部LOAD触发或SYNC触发 Others：Reserved Fh：在GLDCR2[GFRCLD]写入‘1’时 [1]
OSTMD	[5]	RW	One Shot 载入模式使能控制位 0h：禁止One Shot模式，只要条件满足，Active寄存器都会从Shadow寄存器载入 1h：使能One Shot模式，一旦载入被触发，需要再次对GLDCR2[OSREARM]写入‘1’，才能允许下一次载入触发。
GLDPRD	[9:7]	RW	全局载入触发周期选择。 可以选择N次触发条件满足后，才进行一次全局载入。 000b：Disable Counter（立即触发） 001b：第2次条件满足时触发 010b：第3次条件满足时触发 011b：第4次条件满足时触发

			100b：第5次条件满足时触发 101b：第6次条件满足时触发 110b：第7次条件满足时触发 111b：第8次条件满足时触发
GLDCNT	[12:10]	RW	全局载入事件计数器。 计数器值表示当前已发生多少次事件触发。

注意 [1]：如果更新的周期值比更新前小，且立即更新发生时计数器已经超过更新的周期值，计数器将继续计数直到溢出，然后更新才会生效。类似，如果更新的比较值比更新前小，且立即更新发生时计数器已经超过更新的比较值，本周期将不会发生match事件。

13.4.9 GLDCFG (全局载入配置)

- 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																		EMOSR	AQCSF	AQCR4	AQCR3	AQCR2	AQCR1	DBCR	DBDTF	DBDTR	CMPD	CMPC	CMPB	CMPA	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	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
PRDR	[0]	RW	PRDR寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
CMPA	[1]	RW	CMPA寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
CMPB	[2]	RW	CMPB寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
CMPC	[3]	RW	CMPC寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
CMPD	[4]	RW	CMPD寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
DBDTR	[5]	RW	DBDTR寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
DBDTF	[6]	RW	DBDTF寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置

DBCR	[7]	RW	DBCR寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
AQCR1	[8]	RW	AQCR1寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
AQCR2	[9]	RW	AQCR2寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
AQCR3	[10]	RW	AQCR1寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
AQCR4	[11]	RW	AQCR2寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
AQCSF	[12]	RW	AQCSF寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
EMOSR	[13]	RW	EMOSR寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置

13.4.10 GLDCR2 (全局载入控制寄存器2)

- 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																														GFRCLD	OSREARM
0	0	0	0	0	0	0	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	W	RW

Name	Bit	Type	Description
OSREARM	[0]	RW	重置ONE SHOT模式 0：写入‘0’无效，读取时总是返回‘0’ 1：重置ONE SHOT模式。ONE SHOT模式下，一次触发后，需要重置模式才允许再次触发
GFRCLD	[1]	RW	软件产生一次GLD触发。 0：写入‘0’无效，读取时总是返回‘0’ 1：软件产生一次GLD触发事件

NOTE: 该寄存器受 REGPROT 保护，需要先解锁，才能写入。

13.4.11 PRDR (周期设置寄存器)

- 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
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	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
PRDR	[15:0]	RW	时基控制周期寄存器。 此控制位决定了PWM输出波形的周期值。通过设置CR[PRDLD]可以选择Shadow到Active载入的触发条件。

13.4.12 PHSR (相位设置寄存器)

- 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
PHSDIR	RSVD															PHSR															
0	0	0	0	0	0	0	0	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	Type	Description
PHSR	[15:0]	RW	相位控制寄存器。 此控制位决定了PWM输出波形的相位。当CR[PHSEN] = 0时，同步事件不会触发PHSR载入到CNT中，当CR[PHSEN] = 1时，同步事件发生会触发PHSR载入到CNT中。
PHSDIR	[31]	RW	相位方向控制位。 此控制位只在计数模式为递增递减模式下有效。此控制位配置在同步事件发生时，计数器值从PHSR载入后，计数器的计数方向。新的配置方向与同步前计数器的计数方向无关。在递增模式或递减模式下，此控制位无效。 0h：同步后递减 1h：同步后递增

NOTE: PHSR 寄存器只有在外部 Sync 触发时(SYNCIN0)才有效

13.4.13 CMPA (比较值A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
OVWRT	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
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
CMPA	[15:0]	RW	比较值A寄存器。 此寄存器有Shadow寄存器，Shadow模式可以通过CMPLDR[SHDWCMPA]进行设置。在Shadow模式下，可以通过CMPLDR[LDAMD]选择Shadow到Active载入的触发条件。在写入前，可以通过SHDWAFULL控制位检测当前寄存器状态。 当工作于Capture模式下，此寄存器对应CAPLD0事件触发的捕获值。
OVWRT	[31]	RW	Over Write Flag 标志位。 表示当前Capture值是否被Over Written。当连续两次针对于当前寄存器的捕获事件发生，而两次捕获之间未对当前寄存器进行过读取操作，则Over Written标志被置位。读取当前寄存器会自动清除Over Written标志。 此标志位只有在Capture模式下有效。

13.4.14 CMPB (比较值B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
OVWRT	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
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
CMPB	[15:0]	RW	比较值B寄存器。 此寄存器有Shadow寄存器，Shadow模式可以通过CMPLDR[SHDWCMPB]进行设置。在Shadow模式下，可以通过CMPLDR[LDBMD]选择Shadow到Active载入的触发条件。在写入前，可以通过SHDWBFULL控制位检测当前寄存器状态。 当工作于Capture模式下，此寄存器对应CAPLD1事件触发的捕获值。
OVWRT	[31]	RW	Over Write Flag 标志位。 表示当前Capture值是否被Over Written。当连续两次针对于当前寄存器的捕获事件发生，而两次捕获之间未对当前寄存器进行过读取操作，则Over Written标志被置位。读取当前寄存器会自动清除Over Written标志。 此标志位只有在Capture模式下有效。

13.4.15 CMPC (比较值C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
OVWRT	RSVD															CMPC															
0	0	0	0	0	0	0	0	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
CMPC	[15:0]	RW	比较值C寄存器。 此寄存器有Shadow寄存器，Shadow模式可以通过CMPLDR[SHDWCMPC]进行设置。在Shadow模式下，可以通过CMPLDR[LDCMD]选择Shadow到Active载入的触发条件。 当工作于Capture模式下，此寄存器对应CAPLD2事件触发的捕获值。
OVWRT	[31]	RW	Over Write Flag 标志位。 表示当前Capture值是否被Over Written。当连续两次针对于当前寄存器的捕获事件发生，而两次捕获之间未对当前寄存器进行过读取操作，则Over Written标志被置位。读取当前寄存器会自动清除Over Written标志。 此标志位只有在Capture模式下有效。

13.4.16 CMPD (比较值D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
OVWRT	RSVD															CMPD															
0	0	0	0	0	0	0	0	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
CMPD	[15:0]	RW	比较值D寄存器。 此寄存器有Shadow寄存器，Shadow模式可以通过CMPLDR[SHDWCMPD]进行设置。在Shadow模式下，可以通过CMPLDR[LDDMD]选择Shadow到Active载入的触发条件。 当工作于Capture模式下，此寄存器对应CAPLD3事件触发的捕获值。
OVWRT	[31]	RW	Over Write Flag 标志位。 表示当前Capture值是否被Over Written。当连续两次针对于当前寄存器的捕获事件发生，而两次捕获之间未对当前寄存器进行过读取操作，则Over Written标志被置位。读取当前寄存器会自动清除Over Written标志。 此标志位只有在Capture模式下有效。

13.4.17 CMPLDR (比较值载入控制寄存器)

- Address = Base Address +0x003C, Reset Value = 0x0000_2490

31	30	29	28	27	26	25	24	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										SHDWBFULL	SHDWAFULL					LDDMD			LDCMD			LDBMD			LDAMD			SHDWCMPD	SHDWCMPC	SHDWCMPB	SHDWCMPA
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0
R	R	R	R	R	R	R	R	R	R	RW	RW	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
SHDWCMPA	[0]	RW	CMPA的Shadow功能使能控制。 0h：Shadow模式 1h：Immediate模式 [1]
SHDWCMPB	[1]	RW	CMPB的Shadow功能使能控制。 0h：Shadow模式 1h：Immediate模式 [1]
SHDWCMPC	[2]	RW	CMPC的Shadow功能使能控制。 0h：Shadow模式 1h：Immediate模式 [1]
SHDWCMPD	[3]	RW	CMPD的Shadow功能使能控制。 0h：Shadow模式 1h：Immediate模式 [1]
LDAMD	[6:4]	RW	Shadow模式下，Active CMPA从Shadow CMPA载入控制。 xx1b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 x1xb：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 1xxb：外部LOAD触发或SYNC触发时，Shadow寄存器载入到Active寄存器中 000b：不进行载入 每个控制位分别对应一个触发条件，可以同时使能多个触发条件。例如，当设置011b时，CNT=ZRO或CNT=PRD时，都会触发寄存器载入。
LDBMD	[9:7]	RW	Shadow模式下，Active CMPB从Shadow CMPB载入控制。 xx1b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 x1xb：当CNT=PRD时，Shadow寄存器载入到Active寄存器中

			1xxb：外部LOAD触发或SYNC触发时，Shadow寄存器载入到Active寄存器中 000b：不进行载入
LDCMD	[12:10]	RW	Shadow模式下，Active CMPC从Shadow CMPC载入控制。 xx1b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 x1xb：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 1xxb：外部LOAD触发或SYNC触发时，Shadow寄存器载入到Active寄存器中 000b：不进行载入
LDDMD	[15:13]	RW	Shadow模式下，Active CMPD从Shadow CMPD载入控制。 xx1b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 x1xb：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 1xxb：外部LOAD触发或SYNC触发时，Shadow寄存器载入到Active寄存器中 000b：不进行载入
SHDWAFULL	[20]	R	CMPA的Shadow寄存器非空标志位。 当对CMPA进行写操作时，该标志位置位。该标志位在Shadow被载入到Active后，会自动清除。 0h：Shadow空 1h：Shadow非空，对当前CMP寄存器写入会覆盖Shadow中未被载入的值
SHDWAFULL	[21]	R	CMPB的Shadow寄存器非空标志位。 当对CMPB进行写操作时，该标志位置位。该标志位在Shadow被载入到Active后，会自动清除。 0h：Shadow空 1h：Shadow非空，对当前CMP寄存器写入会覆盖Shadow中未被载入的值

注意 [1]：如果更新的比较值比更新前小，且立即更新发生时计数器已经超过更新的比较值，本周期将不会发生match事件。

13.4.18 CNT (时基计数器寄存器)

- 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
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
CNT	[15:0]	RW	时基计数器寄存器。 对CNT读取时，返回当前计数器值。对CNT写入时，将直接更新CNT的计数值。 CNT计数器没有Shadow寄存器，CPU的写入将直接影响当前计数器值。

13.4.19 AQLDR (波形输出载入控制寄存器)

- Address = Base Address +0x0044, Reset Value = 0x0000_0024

31	30	29	28	27	26	25	24	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																LDDMD			LDCMD			SHDWAQ4	SHDWAQ3	LDBMD			LDAMD			SHDWAQ2	SHDWAQ1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	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
SHDWAQ1	[0]	RW	AQCR1寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
SHDWAQ2	[1]	RW	AQCR2寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
LDAMD	[4:2]	RW	Shadow模式下，Active AQCR1从Shadow载入控制。 xx1b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 x1xb：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 1xxb：外部LOAD触发或SYNC触发时，Shadow寄存器载入到Active寄存器中 000b：不进行载入 每个控制位分别对应一个触发条件，可以同时使能多个触发条件。例如，当设置011b时，CNT=ZRO或CNT=PRD时，都会触发寄存器载入。
LDBMD	[7:5]	RW	Shadow模式下，Active AQCR2从Shadow载入控制。 xx1b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 x1xb：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 1xxb：外部LOAD触发或SYNC触发时，Shadow寄存器载入到Active寄存器中 000b：不进行载入 每个控制位分别对应一个触发条件，可以同时使能多个触发条件。例如，当设置011b时，CNT=ZRO或CNT=PRD时，都会触发寄存器载入。
SHDWAQ3	[8]	RW	AQCR3寄存器的Shadow功能使能控制。 0h：Immediate模式

			1h：Shadow模式
SHDWAQ4	[9]	RW	AQCR4寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
LDCMD	[12:10]	RW	Shadow模式下，Active AQCR3从Shadow载入控制。 xx1b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 x1xb：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 1xxb：外部LOAD触发或SYNC触发时，Shadow寄存器载入到Active寄存器中 000b：不进行载入 每个控制位分别对应一个触发条件，可以同时使能多个触发条件。例如，当设置011b时，CNT=ZRO或CNT=PRD时，都会触发寄存器载入。
LDDMD	[15:13]	RW	Shadow模式下，Active AQCR4从Shadow载入控制。 xx1b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 x1xb：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 1xxb：外部LOAD触发或SYNC触发时，Shadow寄存器载入到Active寄存器中 000b：不进行载入 每个控制位分别对应一个触发条件，可以同时使能多个触发条件。例如，当设置011b时，CNT=ZRO或CNT=PRD时，都会触发寄存器载入。

13.4.20 AQCR1 (PWM1 波形输出控制寄存器)

- 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								C2SEL		C1SEL		T2D		T2U		T1D		T1U		C2D		C2U		C1D		C1U		PRD		ZRO	
0	0	0	0	0	0	0	0	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
ZRO	[1:0]	RW	当CNT值等于零时，PWM1 上做出的波形输出动作定义。 在递增递减模式时，当计数器值等于零时，计数方向为递增模式 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
PRD	[3:2]	RW	当CNT值等于PRDR时，PWM 1上做出的波形输出动作定义。 在递增递减模式时，当计数器值等于PRDR时，计数方向为递减模式 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1U	[5:4]	RW	当CNT值等于C1，且此时计数方向为递增时，PWM 1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）

C1D	[7:6]	RW	当CNT值等于C1，且此时计数方向为递减时，PWM 1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C2U	[9:8]	RW	当CNT值等于C2，且此时计数方向为递增时，PWM 1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C2D	[11:10]	RW	当CNT值等于C2，且此时计数方向为递减时，PWM 1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1U	[13:12]	RW	当T1事件发生，且此时计数方向为递增时，PWM 1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1D	[15:14]	RW	当T1事件发生，且此时计数方向为递减时，PWM 1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T2U	[17:16]	RW	当T2事件发生，且此时计数方向为递增时，PWM 1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）

T2D	[19:18]	RW	当T2事件发生，且此时计数方向为递减时，PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1SEL	[21:20]	RW	C1比较值的数据源选择 0h：CMPA寄存器作为C1的数据源 1h：CMPB寄存器作为C1的数据源 2h：CMPC寄存器作为C1的数据源 3h：CMPD寄存器作为C1的数据源
C2SEL	[23:22]	RW	C2比较值数据源选择 0h：CMPA寄存器作为C2的数据源 1h：CMPB寄存器作为C2的数据源 2h：CMPC寄存器作为C2的数据源 3h：CMPD寄存器作为C2的数据源

13.4.21 AQCR2 (PWM2波形输出控制寄存器)

- 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								C2SEL		C1SEL		T2D		T2U		T1D		T1U		C2D		C2U		C1D		C1U		PRD		ZRO	
0	0	0	0	0	0	0	0	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
ZRO	[1:0]	RW	当CNT值等于零时，PWM 2上做出的波形输出动作定义。 在递增递减模式时，当计数器值等于零时，计数方向为递增模式 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
PRD	[3:2]	RW	当CNT值等于PRDR时，PWM 2上做出的波形输出动作定义。 在递增递减模式时，当计数器值等于PRDR时，计数方向为递减模式 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1U	[5:4]	RW	当CNT值等于C1，且此时计数方向为递增时，PWM 2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）

C1D	[7:6]	RW	当CNT值等于C1，且此时计数方向为递减时，PWM 2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C2U	[9:8]	RW	当CNT值等于C2，且此时计数方向为递增时，PWM 2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C2D	[11:10]	RW	当CNT值等于C2，且此时计数方向为递减时，PWM 2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1U	[13:12]	RW	当T1事件发生，且此时计数方向为递增时，PWM 2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1D	[15:14]	RW	当T1事件发生，且此时计数方向为递减时，PWM 2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T2U	[17:16]	RW	当T2事件发生，且此时计数方向为递增时，PWM 2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）

T2D	[19:18]	RW	当T2事件发生，且此时计数方向为递减时，PWM 2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1SEL	[21:20]	RW	C1比较值的数据源选择 0h：CMPA寄存器作为C1的数据源 1h：CMPB寄存器作为C1的数据源 2h：CMPC寄存器作为C1的数据源 3h：CMPD寄存器作为C1的数据源
C2SEL	[23:22]	RW	C2比较值数据源选择 0h：CMPA寄存器作为C2的数据源 1h：CMPB寄存器作为C2的数据源 2h：CMPC寄存器作为C2的数据源 3h：CMPD寄存器作为C2的数据源

13.4.22 AQCR3 (PWM3波形输出控制寄存器)

- 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								C2SEL		C1SEL		T2D		T2U		T1D		T1U		C2D		C2U		C1D		C1U		PRD		ZRO	
0	0	0	0	0	0	0	0	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
ZRO	[1:0]	RW	当CNT值等于零时，PWM3做出的波形输出动作定义。 在递增递减模式时，当计数器值等于零时，计数方向为递增模式 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
PRD	[3:2]	RW	当CNT值等于PRDR时，PWM3做出的波形输出动作定义。 在递增递减模式时，当计数器值等于PRDR时，计数方向为递减模式 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1U	[5:4]	RW	当CNT值等于C1，且此时计数方向为递增时，PWM3做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）

C1D	[7:6]	RW	当CNT值等于C1，且此时计数方向为递减时，PWM3做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C2U	[9:8]	RW	当CNT值等于C2，且此时计数方向为递增时，PWM3做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C2D	[11:10]	RW	当CNT值等于C2，且此时计数方向为递减时，PWM3做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1U	[13:12]	RW	当T1事件发生，且此时计数方向为递增时，PWM3做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1D	[15:14]	RW	当T1事件发生，且此时计数方向为递减时，PWM3做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T2U	[17:16]	RW	当T2事件发生，且此时计数方向为递增时，PWM3做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）

T2D	[19:18]	RW	当T2事件发生，且此时计数方向为递减时，PWM3做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1SEL	[21:20]	RW	C1比较值的数据源选择 0h：CMPA寄存器作为C1的数据源 1h：CMPB寄存器作为C1的数据源 2h：CMPC寄存器作为C1的数据源 3h：CMPD寄存器作为C1的数据源
C2SEL	[23:22]	RW	C2比较值数据源选择 0h：CMPA寄存器作为C2的数据源 1h：CMPB寄存器作为C2的数据源 2h：CMPC寄存器作为C2的数据源 3h：CMPD寄存器作为C2的数据源

13.4.23 AQCR4 (PWM4波形输出控制寄存器)

- 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
RSVD								C2SEL		C1SEL		T2D		T2U		T1D		T1U		C2D		C2U		C1D		C1U		PRD		ZRO	
0	0	0	0	0	0	0	0	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
ZRO	[1:0]	RW	当CNT值等于零时，PWM4做出的波形输出动作定义。 在递增递减模式时，当计数器值等于零时，计数方向为递增模式 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
PRD	[3:2]	RW	当CNT值等于PRDR时，PWM4做出的波形输出动作定义。 在递增递减模式时，当计数器值等于PRDR时，计数方向为递减模式 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1U	[5:4]	RW	当CNT值等于C1，且此时计数方向为递增时，PWM4做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）

C1D	[7:6]	RW	当CNT值等于C1，且此时计数方向为递减时，PWM4做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C2U	[9:8]	RW	当CNT值等于C2，且此时计数方向为递增时，PWM4做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C2D	[11:10]	RW	当CNT值等于C2，且此时计数方向为递减时，PWM4做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1U	[13:12]	RW	当T1事件发生，且此时计数方向为递增时，PWM4做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1D	[15:14]	RW	当T1事件发生，且此时计数方向为递减时，PWM4做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T2U	[17:16]	RW	当T2事件发生，且此时计数方向为递增时，PWM4做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）

T2D	[19:18]	RW	当T2事件发生，且此时计数方向为递减时，PWM4做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1SEL	[21:20]	RW	C1比较值的数据源选择 0h：CMPA寄存器作为C1的数据源 1h：CMPB寄存器作为C1的数据源 2h：CMPC寄存器作为C1的数据源 3h：CMPD寄存器作为C1的数据源
C2SEL	[23:22]	RW	C2比较值数据源选择 0h：CMPA寄存器作为C2的数据源 1h：CMPB寄存器作为C2的数据源 2h：CMPC寄存器作为C2的数据源 3h：CMPD寄存器作为C2的数据源

13.4.24 AQTSCR (T事件触发源选择寄存器)

- 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																								T2SEL				T1SEL			
0	0	0	0	0	0	0	0	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
T1SEL	[3:0]	RW	T1事件触发源选择。每个Bit独立控制单独输入源的使能，当多个Bit同时有效时，被使能的输入源通过逻辑或组成T1事件。 Bit0：SYNCIN4触发 Bit1：EP0 Bit2：EP1 Bit3：EP2 Bit4：EP3 Bit5：EP4 Bit6：EP5 Bit7：EP6
T2SEL	[7:4]	RW	T2事件触发源选择。每个Bit独立控制单独输入源的使能，当多个Bit同时有效时，被使能的输入源通过逻辑或组成T1事件。 0h：SYNCIN5触发 Bit1：EP0 Bit2：EP1 Bit3：EP2 Bit4：EP3 Bit5：EP4 Bit6：EP5 Bit7：EP6

13.4.25 AQOSF (一次性软件波形控制寄存器)

- Address = Base Address +0x005C, Reset Value = 0x0001_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														RLDCSF		RSVD	ACT4		OSTSF4	RSVD	ACT3		OSTSF3	RSVD	ACT2		OSTSF2	RSVD	ACT1		OSTSF1
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	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	R	RW	RW	W	R	RW	RW	W	R	RW	RW	W	R	RW	RW	W

Name	Bit	Type	Description
OSTSF1	[0]	RW	PWM1产生一次性软件强制输出。 0h：对当前位写‘0’无效 1h：产生一次性软件强制输出，此输出状态保持，直到有其他改变PWM1输出状态的触发事件发生。
ACT1	[2:1]	RW	当软件强制输出时，PWM1上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
OSTSF2	[4]	RW	PWM 2上产生一次性软件强制输出。 0h：对当前位写‘0’无效 1h：产生一次性软件强制输出，此输出状态保持，直到有其他改变PWM1输出状态的触发事件发生。
ACT2	[6:5]	RW	当软件强制输出时，PWM2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
OSTSF3	[8]	RW	PWM3产生一次性软件强制输出。 0h：对当前位写‘0’无效 1h：产生一次性软件强制输出，此输出状态保持，直到有其他改变PWM1输出状态的触发事件发生。

ACT3	[10:9]	RW	当软件强制输出时，PWM3上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
OSTSF4	[12]	RW	PWM4产生一次性软件强制输出。 0h：对当前位写‘0’无效 1h：产生一次性软件强制输出，此输出状态保持，直到有其他改变PWM1输出状态的触发事件发生。
ACT4	[14:13]	RW	当软件强制输出时，PWM4上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
RLDCSF	[17:16]	RW	AQCSF寄存器从Shadow载入到Active的控制。 01b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 10b：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 11b：当CNT=ZRO或者PRD时，Shadow寄存器载入到Active寄存器中 00b：立即载入

13.4.26 AQCSF (持续性软件波形控制寄存器)

- 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																								CSF4		CSF3		CSF2		CSF1	
0	0	0	0	0	0	0	0	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
CSF1	[1:0]	RW	通过软件对PWM1做连续强制赋值。在立即更新模式下，赋值将在配置后下一个TCLK输出。在Shadow模式下，在Shadow更新到Active后的下一个TCLK后输出。对Shadow更新到Active的控制，可以通过AQOSF寄存器中的RLDCSF控制位进行配置。 0h：禁止强制赋值 1h：强制输出低 2h：强制输出高 3h：禁止强制赋值
CSF2	[3:2]	RW	通过软件对PWM2做连续强制赋值。在立即更新模式下，赋值将在配置后下一个TCLK输出。在Shadow模式下，在Shadow更新到Active后的下一个TCLK后输出。对Shadow更新到Active的控制，可以通过AQOSF寄存器中的RLDCSF控制位进行配置。 0h：禁止强制赋值 1h：强制输出低 2h：强制输出高 3h：禁止强制赋值
CSF3	[5:4]	RW	通过软件对PWM3做连续强制赋值。在立即更新模式下，赋值将在配置后下一个TCLK输出。在Shadow模式下，在Shadow更新到Active后的下一个TCLK后输出。对Shadow更新到Active的控制，可以通过AQOSF寄存器中的RLDCSF控制位进行配置。 0h：禁止强制赋值 1h：强制输出低 2h：强制输出高 3h：禁止强制赋值

CSF4	[7:6]	RW	通过软件对PWM4做连续强制赋值。在立即更新模式下，赋值将在配置后下一个TCLK输出。在Shadow模式下，在Shadow更新到Active后的下一个TCLK后输出。对Shadow更新到Active的控制，可以通过AQOSF寄存器中的RLDCSF控制位进行配置。 0h：禁止强制赋值 1h：强制输出低 2h：强制输出高 3h：禁止强制赋值

13.4.27 DBLDR (死区配置载入控制寄存器)

- Address = Base Address +0x0064, Reset Value = 0x0000_0492

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																				LDPSCMD		SHDWPSC	LDDTFMD		SHDWDTF	LDDTRMD		SHDWDTR	CRLDMODE		CRSHDWEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	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
CRSHDWEN	[0]	RW	DBCR寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
CRLDMODE	[2:1]	RW	Shadow模式下，Active DBCR从Shadow载入控制。 00b：不进行载入 01b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 10b：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 11b：当CNT=ZRO或者PRD时，Shadow寄存器载入到Active寄存器中
SHDWDTR	[3]	RW	DBDTR寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
LDDTRMD	[5:4]	RW	Shadow模式下，Active DBDTR从Shadow载入控制。 00b：不进行载入 01b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 10b：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 11b：当CNT=ZRO或者PRD时，Shadow寄存器载入到Active寄存器中
SHDWDTF	[6]	RW	DBDTF寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
LDDTFMD	[8:7]	RW	Shadow模式下，Active DBDTF从Shadow载入控制。 00b：不进行载入 01b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中

			10b：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 11b：当CNT=ZRO或者PRD时，Shadow寄存器载入到Active寄存器中
SHDWPSC	[9]	RW	DCKPSC寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
LDPSCMD	[11:10]	RW	Shadow模式下，Active DCKPSC从Shadow载入控制。 00b：不进行载入 01b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 10b：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 11b：当CNT=ZRO或者PRD时，Shadow寄存器载入到Active寄存器中

13.4.28 DBCR (死区配置控制寄存器)

- 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
RSVD				CHC_EDEB	CHB_EDEB	CHA_EDEB	DCKSEL	CHC_OUTSEL		CHC_INSEL		CHC_POLARITY		CHC_OUTSEL		CHB_OUTSEL		CHB_INSEL		CHB_POLARITY		CHB_OUTSEL		CHA_OUTSWAP		CHA_INSEL		CHA_POLARITY		CHA_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	RW	RW	RW	RW	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
CHA_OUTSEL	[1:0]	RW	死区输出配置（S1、S0开关）。 0h：bypass死区控制，X通道输出PWM1，Y通道输出PWM2 1h：X通道输出PWM1，使能Y通道的下降沿延时 2h：使能X通道的上升沿延时，Y通道输出PWM2 3h：使能X通道的上升沿延时，使能Y通道的下降沿延时
CHA_POLARITY	[3:2]	RW	输出极性控制（S3、S2开关）。 0h：X通道和Y通道延时输出不反向 1h：X通道的延时输出反向 2h：X通道的延时输出反向 3h：X通道和Y通道延时输出全部反向
CHA_INSEL	[5:4]	RW	延时模块输入选择（S5、S4开关）。在经典死区控制模式下，上升沿延时和下降沿延时都选择同一个输入信号进行处理。 0h：PWM1作为上升沿和下降沿延时处理的输入信号 1h：PWM2作为上升沿延时输入，PWM1作为下降沿延时输入 2h：PWM1作为上升沿延时输入，PWM2作为下降沿延时输入 3h：PWM2作为上升沿和下降沿延时处理的输入信号
CHA_OUTSWAP	[7:6]	RW	死区输出交换控制（S8、S7开关）。 0h：OUTX=X通道输出，OUTY=Y通道输出 1h：OUTX=Y通道输出，OUTY=Y通道输出 2h：OUTX=X通道输出，OUTY=X通道输出 3h：OUTX=Y通道输出，OUTY=X通道输出

CHB_OUTSEL	[9:8]	RW	死区输出配置（S1、S0开关）。 0h：bypass死区控制，X通道输出PWM2，Y通道输出PWM3 1h：X通道输出PWM2，使能Y通道的下降沿延时 2h：使能X通道的上升沿延时，Y通道输出PWM3 3h：使能X通道的上升沿延时，使能Y通道的下降沿延时
CHB_POLARITY	[11:10]	RW	输出极性控制（S3、S2开关）。 0h：X通道和Y通道延时输出不反向 1h：X通道的延时输出反向 2h：X通道的延时输出反向 3h：X通道和Y通道延时输出全部反向
CHB_INSEL	[13:12]	RW	延时模块输入选择（S5、S4开关）。 0h：PWM2作为上升沿和下降沿延时处理的输入信号 1h：PWM3作为上升沿延时输入，PWM2作为下降沿延时输入 2h：PWM2作为上升沿延时输入，PWM3作为下降沿延时输入 3h：PWM3作为上升沿和下降沿延时处理的输入信号 在经典死区控制模式下，上升沿延时和下降沿延时始终选择同一个输入信号进行处理。
CHB_OUTSWAP	[15:14]	RW	死区输出交换控制（S8、S7开关）。 0h：OUTX=X通道输出，OUTY=Y通道输出 1h：OUTX=Y通道输出，OUTY=Y通道输出 2h：OUTX=X通道输出，OUTY=X通道输出 3h：OUTX=Y通道输出，OUTY=X通道输出
CHC_OUTSEL	[17:16]	RW	死区输出配置（S1、S0开关）。 0h：bypass死区控制，X通道输出PWM3，Y通道输出PWM4 1h：X通道输出PWM3，使能Y通道的下降沿延时 2h：使能X通道的上升沿延时，Y通道输出PWM4 3h：使能X通道的上升沿延时，使能Y通道的下降沿延时
CHC_POLARITY	[19:18]	RW	输出极性控制（S3、S2开关）。 0h：X通道和Y通道延时输出不反向 1h：X通道的延时输出反向 2h：X通道的延时输出反向 3h：X通道和Y通道延时输出全部反向

CHC_INSEL	[21:20]	RW	延时模块输入选择（S5、S4开关）。在经典死区控制模式下，上升沿延时和下降沿延时都选择同一个输入信号进行处理。 0h：PWM3作为上升沿和下降沿延时处理的输入信号 1h：PWM4作为上升沿延时输入，PWM3作为下降沿延时输入 2h：PWM3作为上升沿延时输入，PWM4作为下降沿延时输入 3h：PWM4作为上升沿和下降沿延时处理的输入信号
CHC_OUTSWAP	[23:22]	RW	死区输出交换控制（S8、S7开关）。 0h：OUTX=X通道输出，OUTY=Y通道输出 1h：OUTX=Y通道输出，OUTY=Y通道输出 2h：OUTX=X通道输出，OUTY=X通道输出 3h：OUTX=Y通道输出，OUTY=X通道输出
DCKSEL	[24]	RW	半周期时钟使能控制。 0：死区控制延时计数器以TCLK频率工作 1：死区控制延时计数器以HCLK/(DPSC+1)工作
CHA_DEDB	[25]	RW	在PWM1 DBCOUTY 上选择死区双沿模式（S6） 0h：不使用死区双沿 1h：使用死区双沿
CHB_DEDB	[26]	RW	PWM 2 DBCOUTY上选择死区双沿模式（S6） 0h：不使用死区双沿 1h：使用死区双沿
CHC_DEDB	[27]	RW	在PWM3 DBCOUTY上选择死区双沿模式（S6） 0h：不使用死区双沿 1h：使用死区双沿

13.4.29 DPSCR (死区延迟时钟分频控制寄存器)

- 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
RSVD																DPSC															
0	0	0	0	0	0	0	0	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
DPSC	[15:0]	RW	时钟分频控制。 DBCLK作为死区控制延时计数器的时钟，可以选择TCLK作为时钟源或者从HCLK分频得到。当DBCR[DCKSEL]选择HCLK的分频时，分频系数通过DPSC设置。 DBCLK的频率：$F_{DBCLK} = F_{HCLK} / (DPSC+1)$

13.4.30 DBDTR (死区控制上升沿延时寄存器)

- 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																DTR															
0	0	0	0	0	0	0	0	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
DTR	[15:0]	RW	上升沿延时数值 T_{RED} = DTR x T_{DBCLK}

13.4.31 DBDTF (死区控制下降沿延时寄存器)

- 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
RSVD																DTF															
0	0	0	0	0	0	0	0	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
DTF	[15:0]	RW	下降沿延时数值 $T_{RED} = DTF \times T_{DBCLK}$

13.4.32 CPCR (斩波输出控制寄存器)

- 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
RSVD										CHCY_CPEN	CHCX_CPEN	CHBY_CPEN	CHBX_CPEN	CHAY_CPEN	CHAX_CPEN	C1SEL		CDUTY			CDIV			OSPWTH						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	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R

Name	Bit	Type	Description
OSPWTH	[6:2]	RW	首脉冲宽度设置。首脉冲的宽度可以配置为载波周期的整数倍。当该控制位设为零时，所有脉冲宽度均由CDIV和CDUTY配置。 Twidth = T_{chop} x OSPWTH （T_{chop}为一个载波的周期时间）
CDIV	[10:7]	RW	载波频率设置。载波的频率设置基于PCLK的8倍分频进行设置。 F_{chop} = PCLK / ((CDIV+1) x 8)
CDUTY	[13:11]	RW	载波的占空比设置。 0h：禁止载波 1h：Duty = 7/8 2h：Duty = 6/8 …….. 6h：Duty = 2/8 7h：Duty = 1/8
C1SEL	[15:14]	RW	载波信号源选择控制位。 0h：EPT内部产生载波 1h：TIN的输入 其他：保留
CHx_CPEN	[21:16]	RW	斩波输出使能控制位。 0b：禁止当前通道斩波输出 1b：开启当前通道斩波输出

13.4.33 EMSRC (紧急状态输入控制寄存器)

- 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
EP7_SEL				EP6_SEL				EP5_SEL				EP4_SEL				EP3_SEL				EP2_SEL				EP1_SEL				EP0_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	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
EPx_SEL	[31:0]	RW	EPx的输入选择控制。 1h：选择EBI0（GPIO）作为当前EP的输入 2h：选择EBI1（GPIO）作为当前EP的输入 3h：选择EBI2（GPIO）作为当前EP的输入 4h：选择EBI3（GPIO）作为当前EP的输入 5h：选择EBI4（LVD）作为当前EP的输入 Eh：选择ORL0作为当前EP的输入 Fh：选择ORL1作为当前EP的输入 其他：保留

NOTE: 该寄存器受 REGPROT 保护，需要先解锁，才能写入。

13.4.34 EMSRC2 (紧急状态输入控制寄存器)

- 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								ORL1_EP7	ORL1_EP6	ORL1_EP5	ORL1_EP4	ORL1_EP3	ORL1_EP2	ORL1_EP1	ORL1_EP0	FLT_PACE1				FLT_PACE0				ORL0_EP7	ORL0_EP6	ORL0_EP5	ORL0_EP4	ORL0_EP3	ORL0_EP2	ORL0_EP1	ORL0_EP0
0	0	0	0	0	0	0	0	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
ORL0_EPx	[7:0]	RW	多路EP的逻辑OR输出作为EPx中的可选一路输入信号(ORL0)。 0h：屏蔽当前EP通道作为OR输入 1h：使能当前EP通道作为OR输入
FLT_PACE0	[11:8]	RW	EP0、EP1、EP2和EP3的数字去抖滤波检查周期数。 0h：禁止滤波 1h：2个周期 2h：3个周期 3h：4个周期
FLT_PACE1	[15:12]	RW	EP4、EP5、EP6和EP7的数字去抖滤波检查周期数。 0h：1个周期 1h：2个周期 2h：3个周期 3h：4个周期
ORL1_EPx	[23:16]	RW	多路EP的逻辑OR输出作为EPx中的可选一路输入信号(ORL1)。 0h：屏蔽当前EP通道作为OR输入 1h：使能当前EP通道作为OR输入

NOTE: 该寄存器受 REGPROT 保护，需要先解锁，才能写入。

13.4.35 EMPOL (紧急状态输入极性控制寄存器)

- 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																											EBI4_POL	EBI3_POL	EBI2_POL	EBI1_POL	EBI0_POL
0	0	0	0	0	0	0	0	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
EBIx_POL	[4:0]	RW	EBIx的输入有效极性选择控制。 0h：高电平有效 1h：低电平有效 当EBIx作为异常处理输入时，以电平方式工作。当EBIx作为事件触发源时，设置为高电压有效时，即上升沿触发；设置为低电平有效时，即下降沿触发。

NOTE: 该寄存器受 REGPROT 保护，需要先解锁，才能写入。

13.4.36 EMECR (紧急状态使能控制寄存器)

- Address = Base Address +0x0088, Reset Value = 0x0040_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	EOM_FAULT	MEM_FAULT	CPU_FAULT	RSVD	EMASYNC	SLCLRMD		OSRLDMD		OSRSHDW	RSVD					EP7_LCKMD		EP6_LCFKMD		EP5_LCKMD		EP4_LCKMD		EP3_LCKMD		EP2_LCKMD		EP1_LCKMD		EP0_LCKMD	
0	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
R	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	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
EPx_LCKMD	[15:0]	RW	EPx端触发锁止模式控制。 0h：禁止当前EPx触发锁止 1h：使能当前EPx触发软锁止 2h：使能当前EPx触发硬锁止 3h：禁止当前EPx触发锁止
OSRSHDW	[21]	RW	EMOSR寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
OSRLDMD	[23:22]	RW	Shadow模式下，Active EMOSR从Shadow载入控制。 00b：不进行载入 01b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 10b：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 11b：当CNT=ZRO或者PRD时，Shadow寄存器载入到Active寄存器中
SLCLRMD	[25:24]	RW	软锁止清除条件设置。当CNT值等于设置值，且软锁止不再触发时，硬件自动清除软锁止状态和标志位。 00h：CNT = ZRO时，清除软锁止 01h：CNT = PRD时，清除软锁止 10h：CNT = ZRO或CNT = PRD时，清除软锁止 11h：不自动清除软锁止，必须通过软件清除
EMASYNC	[26]	RW	EP端口同步设置控制位。 0h：使能同步 1h：禁止同步

CPU_FAULT	[28]	RW	CPU错误触发硬锁止控制位。 0h：禁止当前事件触发硬锁止 1h：使能当前事件触发硬锁止
MEM_FAULT	[29]	RW	MEM错误触发硬锁止控制位。（需要同时使能SRAM或者Flash校验功能） 0h：禁止当前事件触发硬锁止 1h：使能当前事件触发硬锁止
EOM_FAULT	[30]	RW	外部晶振错误触发硬锁止控制位。（需要同时使能外部晶振监测功能） 0h：禁止当前事件触发硬锁止 1h：使能当前事件触发硬锁止

NOTE: 该寄存器受 REGPROT 保护，需要先解锁，才能写入。

13.4.37 EMOSR (紧急状态输出控制寄存器1)

- 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
																		EM_COCY		EM_COBY		EM_COAY		EM_COD		EM_COCX		EM_COBX		EM_COAX	
0	0	0	0	0	0	0	0	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
EM_COAX	[1:0]	RW	当发生EP触发的软锁止或者硬锁止时，在CHAX通道上的输出状态设置。 0h：高阻态 1h：高电平 2h：低电平 3h：不做处理
EM_COBX	[3:2]	RW	当发生EP触发的软锁止或者硬锁止时，在CHBX通道上的输出状态设置。 0h：高阻态 1h：高电平 2h：低电平 3h：不做处理
EM_COCX	[5:4]	RW	当发生EP触发的软锁止或者硬锁止时，在CHCX通道上的输出状态设置。 0h：高阻态 1h：高电平 2h：低电平 3h：不做处理
EM_COD	[7:6]	RW	当发生EP触发的软锁止或者硬锁止时，在CHD通道上的输出状态设置。 0h：高阻态 1h：高电平 2h：低电平 3h：不做处理
EM_COAY	[9:8]	RW	当发生EP触发的软锁止或者硬锁止时，在CHAY通道上的输出状态设置。 0h：高阻态

			1h：高电平 2h：低电平 3h：不做处理
EM_COBY	[11:10]	RW	当发生EP触发的软锁止或者硬锁止时，在CHBY通道上的输出状态设置。 0h：高阻态 1h：高电平 2h：低电平 3h：不做处理
EM_COCY	[13:12]	RW	当发生EP触发的软锁止或者硬锁止时，在CHCY通道上的输出状态设置。 0h：高阻态 1h：高电平 2h：低电平 3h：不做处理

NOTE: 该寄存器受 REGPROT 保护，需要先解锁，才能写入。

13.4.38 EMSLSR (紧急软锁止状态寄存器)

- 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																								EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
0	0	0	0	0	0	0	0	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
EPx	[7:0]	R	EPx触发的软锁止状态标志。 0h：软锁止未触发 1h：软锁止已触发

13.4.39 EMSLCLR (紧急软锁止清除寄存器)

- 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																								EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
0	0	0	0	0	0	0	0	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
EPx	[7:0]	W	软件清除EPx触发的软锁止状态标志。 0h：对当前控制位写‘0’无效，读取时总返回‘0’ 1h：清除当前标志位

13.4.40 EMHLSR (紧急硬锁止状态寄存器)

- 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																					EOM_FAULT	MEM_FAULT	CPU_FAULT	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
0	0	0	0	0	0	0	0	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
EPx	[7:0]	R	EPx触发的硬锁止状态标志。 0h：硬锁止未触发 1h：硬锁止已触发
CPU_FAULT	[8]	R	CPU FAULT事件触发的硬锁止状态位。 0h：硬锁止未触发 1h：硬锁止已触发
MEM_FAULT	[9]	R	MEM FAULT事件触发的硬锁止状态位。 0h：硬锁止未触发 1h：硬锁止已触发
EOM_FAULT	[10]	R	EOM FAULT事件触发的硬锁止状态位。 0h：硬锁止未触发 1h：硬锁止已触发

13.4.41 EMHLCLR (紧急硬锁止清除寄存器)

- Address = Base Address +0x00A0, 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																					EOM_FAULT	MEM_FAULT	CPU_FAULT	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
0	0	0	0	0	0	0	0	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	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
EPx	[7:0]	W	软件清除EPx触发的硬锁止状态标志。 0h：对当前控制位写‘0’无效，读取时总返回‘0’ 1h：清除当前标志位
CPU_FAULT	[8]	W	软件清除CPU FAULT事件触发的硬锁止状态位。 0h：对当前控制位写‘0’无效，读取时总返回‘0’ 1h：清除当前标志位
MEM_FAULT	[9]	W	软件清除MEM FAULT事件触发的硬锁止状态位。 0h：对当前控制位写‘0’无效，读取时总返回‘0’ 1h：清除当前标志位
EOM_FAULT	[10]	W	软件清除EOM FAULT事件触发的硬锁止状态位。 0h：对当前控制位写‘0’无效，读取时总返回‘0’ 1h：清除当前标志位

13.4.42 EMFRCR (紧急状态软件触发寄存器)

- 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																								FRC_EP7	FRC_EP6	FRC_EP5	FRC_EP4	FRC_EP3	FRC_EP2	FRC_EP1	FRC_EP0
0	0	0	0	0	0	0	0	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
FRC_EPx	[7:0]	W	软件触发EPx事件。 0h：对当前控制位写‘0’无效，读取时总返回‘0’ 1h：触发EPx事件，置高标志位

NOTE: 该寄存器受 REGPROT 保护，需要先解锁，才能写入。

13.4.43 EMRISR (紧急中断原始状态寄存器)

- 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																					EOM_FAULT	MEM_FAULT	CPU_FAULT	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
0	0	0	0	0	0	0	0	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
EPx	[7:0]	R	EPx事件触发的异常事件中断原始标志位。 0h：无中断请求发生 1h：中断请求发生
CPU_FAULT	[8]	R	CPU_FAULT事件触发的异常事件中断原始标志位。 0h：无中断请求发生 1h：中断请求发生
MEM_FAULT	[9]	R	MEM_FAULT事件触发的异常事件中断原始标志位。 0h：无中断请求发生 1h：中断请求发生
EOM_FAULT	[10]	R	EOM_FAULT事件触发的异常事件中断原始标志位。 0h：无中断请求发生 1h：中断请求发生

13.4.44 EMMISR (紧急中断标志寄存器)

- 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																					EOM_FAULT	MEM_FAULT	CPU_FAULT	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
0	0	0	0	0	0	0	0	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
EPx	[7:0]	R	EPx事件触发的异常事件中断标志位。 0h：无中断请求发生 1h：中断请求发生
CPU_FAULT	[8]	R	CPU_FAULT事件触发的异常事件中断标志位。 0h：无中断请求发生 1h：中断请求发生
MEM_FAULT	[9]	R	MEM_FAULT事件触发的异常事件中断标志位。 0h：无中断请求发生 1h：中断请求发生
EOM_FAULT	[10]	R	EOM_FAULT事件触发的异常事件中断标志位。 0h：无中断请求发生 1h：中断请求发生

13.4.45 EMIMCR (紧急中断使能控制寄存器)

- 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																					EOM_FAULT	MEM_FAULT	CPU_FAULT	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
0	0	0	0	0	0	0	0	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
EPx	[7:0]	RW	EPx事件触发的异常中断使能控制。 0h：禁止对CPU发起中断请求 1h：允许对CPU发起中断请求
CPU_FAULT	[8]	RW	CPU FAULT事件触发的异常中断使能控制。 0h：禁止对CPU发起中断请求 1h：允许对CPU发起中断请求
MEM_FAULT	[9]	RW	MEM FAULT事件触发的异常中断使能控制。 0h：禁止对CPU发起中断请求 1h：允许对CPU发起中断请求
EOM_FAULT	[10]	RW	EOM FAULT事件触发的异常中断使能控制。 0h：禁止对CPU发起中断请求 1h：允许对CPU发起中断请求

13.4.46 EMICR (紧急中断清除寄存器)

- 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																					EOM_FAULT	MEM_FAULT	CPU_FAULT	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
0	0	0	0	0	0	0	0	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	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
EPx	[7:0]	RW	软件清除EPx事件触发的中断标志。 0h：对当前控制位写‘0’无效，读取时总返回‘0’ 1h：清除当前标志位
CPU_FAULT	[8]	RW	软件清除CPU FAULT事件触发的中断标志。 0h：对当前控制位写‘0’无效，读取时总返回‘0’ 1h：清除当前标志位
MEM_FAULT	[9]	RW	软件清除MEM FAULT事件触发的中断标志。 0h：对当前控制位写‘0’无效，读取时总返回‘0’ 1h：清除当前标志位
EOM_FAULT	[10]	RW	软件清除EOM FAULT事件触发的中断标志。 0h：对当前控制位写‘0’无效，读取时总返回‘0’ 1h：清除当前标志位

13.4.47 TRGFTCR (数字比较器滤波窗控制寄存器)

- 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
																								CROSSMD	ALIGNMD		BLKINV	RSVd	SRCSEL		
0	0	0	0	0	0	0	0	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	R	RW	RW	RW

Name	Bit	Type	Description
SRC_SEL	[2:0]	RW	滤波模块的输入信号选择。 0h：禁止滤波 1h：使能SYNCIN0滤波 2h：使能SYNCIN1滤波 3h：使能SYNCIN2滤波 4h：使能SYNCIN3滤波 5h：使能SYNCIN4滤波 6h：使能SYNCIN5滤波 7h：保留
BLKINV	[4]	RW	窗口使能反转控制。 0h：窗口不反转，窗口有效区间禁止滤波输入 1h：窗口反转，窗口有效区间使能滤波输入
ALIGNMD	[6:5]	RW	窗口对齐模式选择。当对齐模式条件满足时，OFFSET将被重置；但窗口宽度将根据CORSSMD设置进行调整。 0h：CNT=ZRO 1h：CNT=PRD 2h：CNT=PRD or CNT=ZRO 3h：T1事件
CROSSMD	[7]	RW	允许滤波窗跨越窗口对齐点。 缺省条件下，当滤波窗在Align条件满足时若任然有效，将跨过窗口对齐点，一直持续到窗口计数器溢出。当禁止跨周期时，在Align条件满足时，窗口计数器将被停止。 0h：禁止对齐点跨窗口

			1h：允许对齐点跨窗口

13.4.48 TRGFTWR (数字比较器滤波窗时序寄存器)

- 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
WINDOW																OFFSET															
0	0	0	0	0	0	0	0	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
OFFSET	[15:0]	RW	滤波窗的OFFSET设置。 此16bit控制位定义了从窗口参考起始位置开始计数多少个TCLK后，开始有效的滤波窗口。参考位置的定义，在TRGFTCR[ALIGNMD]控制位中进行选择。OFFSET的Shadow寄存器在ALIGNMD指定的条件满足时，载入到Active寄存器中，并重新开始计数。
WINDOW	[31:16]	RW	滤波窗的宽度设置。 此16bit控制位定义了滤波窗的宽度，窗口宽度是基于TCLK的计数值。当OFFSET计数器溢出时，WINDOW计数器被重置，并开始计数直到溢出。当OFFSET计数器溢出，但WINDOW状态已经激活时，WINDOW计数器不会重置。在应用时必须注意此条件的设置。

13.4.49 EVTRG (事件触发选择寄存器)

- 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				CNT3INITFRC	CNT2INITFRC	CNT1INITFRC	CNT0INITFRC	TRG3OE	TRG2OE	TRG1OE	TRG0OE	CNT3INITEN	CNT2INITEN	CNT1INITEN	CNT0INITEN	TRG3SEL				TRG2SEL				TRG1SEL				TRG0SEL			
0	0	0	0	0	0	0	0	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
TRG0SEL TRG1SEL	[3:0] [7:4]	RW	TRGEV0，TRGEV1事件的触发源选择。 0000：禁止TRGSRC触发输出 0001：当 CNT = ZRO 产生TRGx事件 0010：当 CNT = PRD 产生TRGx事件 0011：当 CNT = ZRO or CNT = PRD 产生TRGx事件 0100：当 CNT = CMPA 且计数方向为递增时，产生TRGx事件 0101：当 CNT = CMPA 且计数方向为递减时，产生TRGx事件 0110：当 CNT = CMPB 且计数方向为递增时，产生TRGx事件 0111：当 CNT = CMPB 且计数方向为递减时，产生TRGx事件 1000：当 CNT = CMPC 且计数方向为递增时，产生TRGx事件 1001：当 CNT = CMPC 且计数方向为递减时，产生TRGx事件 1010：当 CNT = CMPD 且计数方向为递增时，产生TRGx事件 1011：当 CNT = CMPD 且计数方向为递减时，产生TRGx事件 1100：ExtSync通道 1101：PE0 event 1110：PE1 event 1111：PE2 event

TRG2SEL TRG3SEL	[11:8] [15:12]	RW	TRGEV2，TRGEV3事件的触发源选择。 0000：禁止TRGSRC触发输出 0001：当 CNT = ZRO 产生TRGx事件 0010：当 CNT = PRD 产生TRGx事件 0011：当 CNT = ZRO or CNT = PRD 产生TRGx事件 0100：当 CNT = CMPA 且计数方向为递增时，产生TRGx事件 0101：当 CNT = CMPA 且计数方向为递减时，产生TRGx事件 0110：当 CNT = CMPB 且计数方向为递增时，产生TRGx事件 0111：当 CNT = CMPB 且计数方向为递减时，产生TRGx事件 1000：当 CNT = CMPC 且计数方向为递增时，产生TRGx事件 1001：当 CNT = CMPC 且计数方向为递减时，产生TRGx事件 1010：当 CNT = CMPD 且计数方向为递增时，产生TRGx事件 1011：当 CNT = CMPD 且计数方向为递减时，产生TRGx事件 1100：Period End 1101：PE0 event 1110：PE1 event 1111：PE2 event
CNT0INITEN	[16]	RW	TRGEV0CNT寄存器更新模式控制 0h：无效 1h：TRGEV0CNT在发生LOAD事件触发时，或者EV0CNTINITFRC控制位软件写入‘1’时，EV0CNTINIT的内容更新到EV0CNT中。
CNT1INITEN	[17]	RW	TRGEV1CNT寄存器更新模式控制 0h：无效 1h：TRGEV1CNT在发生LOAD事件触发时，或者EV1CNTINITFRC控制位软件写入‘1’时，EV1CNTINIT的内容更新到EV1CNT中。
CNT2INITEN	[18]	RW	TRGEV2CNT寄存器更新模式控制 0h：无效 1h：TRGEV2CNT在发生LOAD事件触发时，或者EV2CNTINITFRC控制位软件写入‘1’时，EV2CNTINIT的内容更新到EV2CNT中。
CNT3INITEN	[19]	RW	TRGEV3CNT寄存器更新模式控制 0h：无效 1h：TRGEV3CNT在发生LOAD事件触发时，或者EV3CNTINITFRC控制位软件写入‘1’时，EV3CNTINIT的内容更新到EV3CNT中。

TRG0OE	[20]	RW	外部触发端口TRGOUT0使能 0h：禁止触发输出 1h：允许触发输出
TRG1OE	[21]	RW	外部触发端口TRGOUT1使能 0h：禁止触发输出 1h：允许触发输出
TRG2OE	[22]	RW	外部触发端口TRGOUT2使能 0h：禁止触发输出 1h：允许触发输出
TRG3OE	[23]	RW	外部触发端口TRGOUT3使能 0h：禁止触发输出 1h：允许触发输出
CNT0INITFRC	[24]	R	TRGEV0CNT软件触发更新 0h：无效 1h：EVCNT0INIT内容更新到EVCNT0中
CNT1INITFRC	[25]	R	TRGEV1CNT软件触发更新 0h：无效 1h：EVCNT1INIT内容更新到EVCNT1中
CNT2INITFRC	[26]	R	TRGEV2CNT软件触发更新 0h：无效 1h：EVCNT2INIT内容更新到EVCNT2中
CNT3INITFRC	[27]	R	TRGEV3CNT软件触发更新 0h：无效 1h：EVCNT3INIT内容更新到EVCNT3中

13.4.50 EVPS (事件触发计数寄存器)

- 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
TRGEV3CNT				TRGEV2CNT				TRGEV1CNT				TRGEV0CNT				TRGEV3PRD				TRGEV2PRD				TRGEV1PRD				TRGEV0PRD			
0	0	0	0	0	0	0	0	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
TRGEV0PRD	[3:0]	RW	TRGEV0事件计数的周期设置。 当TRGEV0事件发生次数满足周期时，才产生TRGEV0触发事件
TRGEV1PRD	[7:4]	RW	TRGEV1事件计数的周期设置。 当TRGEV1事件发生次数满足周期时，才产生TRGEV1触发事件
TRGEV2PRD	[11:8]	RW	TRGEV2事件计数的周期设置。 当TRGEV2事件发生次数满足周期时，才产生TRGEV2触发事件
TRGEV3PRD	[15:12]	RW	TRGEV3事件计数的周期设置。 当TRGEV3事件发生次数满足周期时，才产生TRGEV3触发事件
TRGEV0CNT	[19:16]	R	TRGEV0事件计数器值。 读取时，返回当前事件计数器值。
TRGEV1CNT	[23:20]	R	TRGEV1事件计数器值。 读取时，返回当前事件计数器值。
TRGEV2CNT	[27:24]	R	TRGEV2事件计数器值。 读取时，返回当前事件计数器值。
TRGEV3CNT	[31:28]	R	TRGEV3事件计数器值。 读取时，返回当前事件计数器值。

13.4.51 EVCNTINIT (事件触发计数器初始化值寄存器)

- 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
																CNT3INIT				CNT2INIT				CNT1INIT				CNT0INIT			
0	0	0	0	0	0	0	0	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
CNT0INIT	[3:0]	RW	TRGEV0CNT计数器的初始化值设置。 当EVTRG[CNT0INITEN]控制位有效时，CNT0INIT的值将在触发条件满足时(LOAD事件)，或EVTRG[CNT0INITFRC]软件置位时，被载入到TRGEV0CNT寄存器中。
CNT1INIT	[7:4]	RW	TRGEV1CNT计数器的初始化值设置。 当EVTRG[CNT1INITEN]控制位有效时，CNT1INIT的值将在触发条件满足时(LOAD事件)，或EVTRG[CNT1INITFRC]软件置位时，被载入到TRGEV1CNT寄存器中。
CNT2INIT	[11:8]	RW	TRGEV2CNT计数器的初始化值设置。 当EVTRG[CNT2INITEN]控制位有效时，CNT2INIT的值将在触发条件满足时(LOAD事件)，或EVTRG[CNT2INITFRC]软件置位时，被载入到TRGEV2CNT寄存器中。
CNT3INIT	[15:12]	RW	TRGEV3CNT计数器的初始化值设置。 当EVTRG[CNT3INITEN]控制位有效时，CNT3INIT的值将在触发条件满足时(LOAD事件)，或EVTRG[CNT3INITFRC]软件置位时，被载入到TRGEV3CNT寄存器中。

13.4.52 EVSWF (事件计数器软件触发控制寄存器)

- 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																												EV3SWF	EV2SWF	EV1SWF	EV0SWF
0	0	0	0	0	0	0	0	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	W	W	W	W

Name	Bit	Type	Description
EV0SWF	[0]	W	软件产生一次EV0的触发 0h：写入‘0’无效 1h：软件产生一次触发
EV1SWF	[1]	W	软件产生一次EV1的触发 0h：写入‘0’无效 1h：软件产生一次触发
EV2SWF	[2]	W	软件产生一次EV2的触发 0h：写入‘0’无效 1h：软件产生一次触发
EV3SWF	[3]	W	软件产生一次EV3的触发 0h：写入‘0’无效 1h：软件产生一次触发

13.4.53 RISR (原始中断状态寄存器)

- Address = Base Address +0x00D0, 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															PEND	CDD	CDU	CCD	CCU	CBD	CBU	CAD	CAU	CAP_LD3	CAP_LD2	CAP_LD1	CAP_LD0	TRGEV3	TRGEV2	TRGEV1	TRGEV0
0	0	0	0	0	0	0	0	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
TRGEV0	[0]	R	TRGEV0中断请求原始标志状态
TRGEV1	[1]	R	TRGEV1中断请求原始标志状态
TRGEV2	[2]	R	TRGEV2中断请求原始标志状态
TRGEV3	[3]	R	TRGEV3中断请求原始标志状态
CAP_LD0	[4]	R	Capture Load to CMPA中断请求原始标志状态
CAP_LD1	[5]	R	Capture Load to CMPB中断请求原始标志状态
CAP_LD2	[6]	R	Capture Load to CMPC中断请求原始标志状态
CAP_LD3	[7]	R	Capture Load to CMPD中断请求原始标志状态
CAU	[8]	R	递增阶段CNT = CMPA中断请求原始标志状态
CAD	[9]	R	递减阶段CNT = CMPA中断请求原始标志状态
CBU	[10]	R	递增阶段CNT = CMPB中断请求原始标志状态
CBD	[11]	R	递减阶段CNT = CMPB中断请求原始标志状态
CCU	[12]	R	递增阶段CNT = CMPC中断请求原始标志状态
CCD	[13]	R	递减阶段CNT = CMPC中断请求原始标志状态
CDU	[14]	R	递增阶段CNT = CMPD中断请求原始标志状态
CDD	[15]	R	递减阶段CNT = CMPD中断请求原始标志状态
PEND	[16]	R	周期结束中断请求原始标志状态
原始中断标志表示中断事件发生，通过设置IMCR中相应位，可以允许该中断请求CPU中断。原始中断标志位需要通过软件清除。 0h：该中断未置位 1h：该中断已置位			

13.4.54 MISR (中断状态寄存器)

- Address = Base Address +0x00D4, 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															PEND	CDD	CDU	CCD	CCU	CBD	CBU	CAD	CAU	CAP_LD3	CAP_LD2	CAP_LD1	CAP_LD0	TRGEV3	TRGEV2	TRGEV1	TRGEV0
0	0	0	0	0	0	0	0	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
TRGEV0	[0]	R	TRGEV0中断请求标志状态
TRGEV1	[1]	R	TRGEV1中断请求标志状态
TRGEV2	[2]	R	TRGEV2中断请求标志状态
TRGEV3	[3]	R	TRGEV3中断请求标志状态
CAP_LD0	[4]	R	Capture Load to CMPA中断请求标志状态
CAP_LD1	[5]	R	Capture Load to CMPB中断请求标志状态
CAP_LD2	[6]	R	Capture Load to CMPC中断请求标志状态
CAP_LD3	[7]	R	Capture Load to CMPD中断请求标志状态
CAU	[8]	R	递增阶段CNT = CMPA中断请求标志状态
CAD	[9]	R	递减阶段CNT = CMPA中断请求标志状态
CBU	[10]	R	递增阶段CNT = CMPB中断请求标志状态
CBD	[11]	R	递减阶段CNT = CMPB中断请求标志状态
CCU	[12]	R	递增阶段CNT = CMPC中断请求标志状态
CCD	[13]	R	递减阶段CNT = CMPC中断请求标志状态
CDU	[14]	R	递增阶段CNT = CMPD中断请求标志状态
CDD	[15]	R	递减阶段CNT = CMPD中断请求标志状态
PEND	[16]	R	周期结束中断请求标志状态
由IMCR使能控制的中断标志。表示中断事件发生，并请求CPU中断。中断标志位随着RISR清除而清除。 0h：该中断未置位 1h：该中断已置位			

13.4.55 IMCR (中断使能控制寄存器)

- Address = Base Address +0x00D8, 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															PEND	CDD	CDU	CCD	CCU	CBD	CBU	CAD	CAU	CAP_LD3	CAP_LD2	CAP_LD1	CAP_LD0	TRGEV3	TRGEV2	TRGEV1	TRGEV0
0	0	0	0	0	0	0	0	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	RW

Name	Bit	Type	Description
TRGEV0	[0]	RW	TRGEV0中断使能控制位。
TRGEV1	[1]	RW	TRGEV1中断使能控制位。
TRGEV2	[2]	RW	TRGEV2中断使能控制位。
TRGEV3	[3]	RW	TRGEV3中断使能控制位。
CAP_LD0	[4]	RW	Capture Load to CMPA中断使能控制位。
CAP_LD1	[5]	RW	Capture Load to CMPB中断使能控制位。
CAP_LD2	[6]	RW	Capture Load to CMPC中断使能控制位。
CAP_LD3	[7]	RW	Capture Load to CMPD中断使能控制位。
CAU	[8]	RW	递增阶段CNT = CMPA中断使能控制位。
CAD	[9]	RW	递减阶段CNT = CMPA中断使能控制位。
CBU	[10]	RW	递增阶段CNT = CMPB中断使能控制位。
CBD	[11]	RW	递减阶段CNT = CMPB中断使能控制位。
CCU	[12]	RW	递增阶段CNT = CMPC中断使能控制位。
CCD	[13]	RW	递减阶段CNT = CMPC中断使能控制位。
CDU	[14]	RW	递增阶段CNT = CMPD中断使能控制位。
CDD	[15]	RW	递减阶段CNT = CMPD中断使能控制位。
PEND	[16]	RW	周期结束中断中断使能控制位。
中断使能控制。该控制位使能时，MISR的置位才允许发生。 0h：关闭中断。 1h：打开中断。			

13.4.56 ICR (中断清除寄存器)

- Address = Base Address +0x00DC, 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															PEND	CDD	CDU	CCD	CCU	CBD	CBU	CAD	CAU	CAP_LD3	CAP_LD2	CAP_LD1	CAP_LD0	TRGEV3	TRGEV2	TRGEV1	TRGEV0
0	0	0	0	0	0	0	0	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	RW

Name	Bit	Type	Description
TRGEV0	[0]	R	TRGEV0中断清除。
TRGEV1	[1]	R	TRGEV1中断清除。
TRGEV2	[2]	R	TRGEV2中断清除。
TRGEV3	[3]	R	TRGEV3中断清除。
CAP_LD0	[4]	R	Capture Load to CMPA中断清除。
CAP_LD1	[5]	R	Capture Load to CMPB中断清除。
CAP_LD2	[6]	R	Capture Load to CMPC中断清除。
CAP_LD3	[7]	R	Capture Load to CMPD中断清除。
CAU	[8]	R	递增阶段CNT = CMPA中断清除。
CAD	[9]	R	递减阶段CNT = CMPA中断清除。
CBU	[10]	R	递增阶段CNT = CMPB中断清除。
CBD	[11]	R	递减阶段CNT = CMPB中断清除。
CCU	[12]	R	递增阶段CNT = CMPC中断清除。
CCD	[13]	R	递减阶段CNT = CMPC中断清除。
CDU	[14]	R	递增阶段CNT = CMPD中断清除。
CDD	[15]	R	递减阶段CNT = CMPD中断清除。
PEND	[16]	R	周期结束中断中断清除。
对当前控制位软件写‘1’可以清除该中断，写入‘0’无效。			

13.4.57 REGPROT (寄存器写保护控制器)

- Address = Base Address +0x00E8, 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
WRKEY																PROTKEY															
0	0	0	0	0	0	0	0	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
PROTKEY	[15:0]	RW	写保护使能控制。 当此寄存器的值不等于C73Ah时，具有写保护功能的寄存器(参看寄存器表)将禁止写入操作。只有解锁后，具有写保护功能的寄存器才允许写操作。对于具有写保护寄存器的写操作完成后，写保护寄存器会自动清除（自动保护使能），所以每次对任意具有写保护功能的寄存器写入之前，都必须进行解锁操作。
WRKEY	[31:16]	W	写入保护KEY 当对PROTKEY进行写操作时，必须将KEY设置为A55Ah，否则写入无效。

14 低功耗定时器（LPT）

14.1 概述

低功耗定时器（Low Power Timer）作为 MCU 的低功耗外设，可以支持异步时钟计数操作，从而实现超低功耗下计数。由于可以基于外部 CLOCK 或独立的时钟源计数，LPT 可以支持在低功耗模式下将系统唤醒。当使用外部时钟计数时，LPT 可以作为外部脉冲计数使用。LPT 内部包含一个 16 位的定时/计数模块，支持 PWM 输出。

注：如果系列内芯片不具有本外围，那它就不具备本章描述的相关资源。具体参考芯片的数据手册。

14.1.1 主要特性

- 16 位递增计数器
- 4 Bit 预分频控制，支持（1，2，4，8，16，32，64，128，256，512，1024，2048，4096 分频）
- 支持多种计数时钟：
  - 内部时钟：ISCLK, IMCLK, EMCLK 或 PCLK
  - 外部时钟：LPT_IN（当没有内部时钟时，可以作为脉冲计数）
  - 一路独立的 PWM 输出
- 一个比较值寄存器
- 支持连续或单次计数模式
- 支持通过 ETCB 触发
- 支持脉冲和 PWM 输出模式

14.1.2 管脚描述

GPIO 上相关 AF 功能映射如下。

Table 16-1 LPT 功能管脚描述

管脚名称	IO 方向	功能描述
LPT_OUT	输出	LPT 的波形输出
LPT_IN	输入	LPT 的时钟输入

14.2 功能描述

14.2.1 模块框图

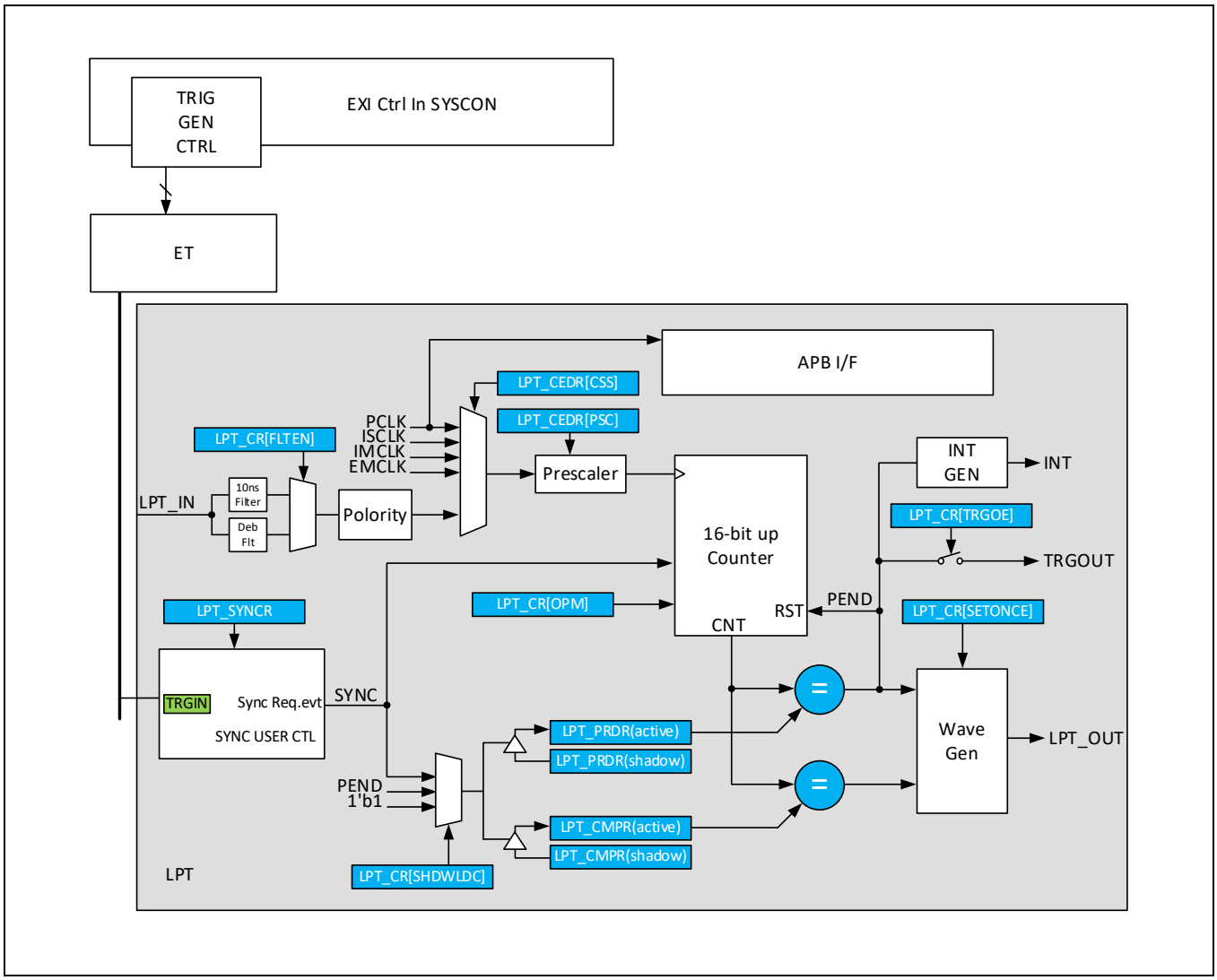


Figure 16-1 模块结构示意图

14.3 基本功能描述

LPT模块是一个简单的计数器，主要目的为低功耗计时。LPT模块的时钟源可以选择ISCLK, IMCLK, EMCLK或PCLK/4，同时还可以选择外部管脚LPT_IN作为时钟输入，实现对外部事件的计数功能。LPT模块还有一个输出LPT_OUT，可以输出简单的PWM波形。LPT_OUT输出在计数值等于比较值(LPT_CMP)和周期值(LPT_PRDR)时会翻转。LPT_OUT输出的初始值可以通过LPT_CR寄存器的输出极性位POL控制。

14.3.1 时钟源

14.3.1.1 概述

LPT计数器计数时钟可以选择PCLK的4分频，系统ISCLK，IMCLK，EMCLK或者由外部管脚提供，计数器在外部计数器时钟控制下计数时，外部计数时钟经过LPT模块内的控制和滤波，产生相应的触发脉冲，该脉冲作为LPT的时基计数器外部触发源，触发计数器递增计数。

LPT计数器的计数时钟选定后，可以通过LPT_PSCR寄存器进行分频。在对LPT_PSCR进行读写时，操作的对象为PSCR的影子寄存器（Shadow Register）；当时基计数器的值等于零时，影子寄存器的值将被更新到内部的活动寄存器中（Active Register）。当对LPT_PSCR更新后，新的分频将在下一个计数周期开始时有效。

14.3.1.2 外部时钟

外部时钟的选择和极性控制，由LPT_CEDR的CSS位控制。在外部时钟模式下，外部时钟通路上集成有数字滤波器。数字滤波器可以在外部时钟有较大干扰时打开。

在外部时钟输入通道上，可以通过设置LPT_CR的FLTEN位使能数字滤波。数字滤波在连续检测到N个一致结果时，才会确认输出翻转，否则将保持当前输出状态。如下图所示。

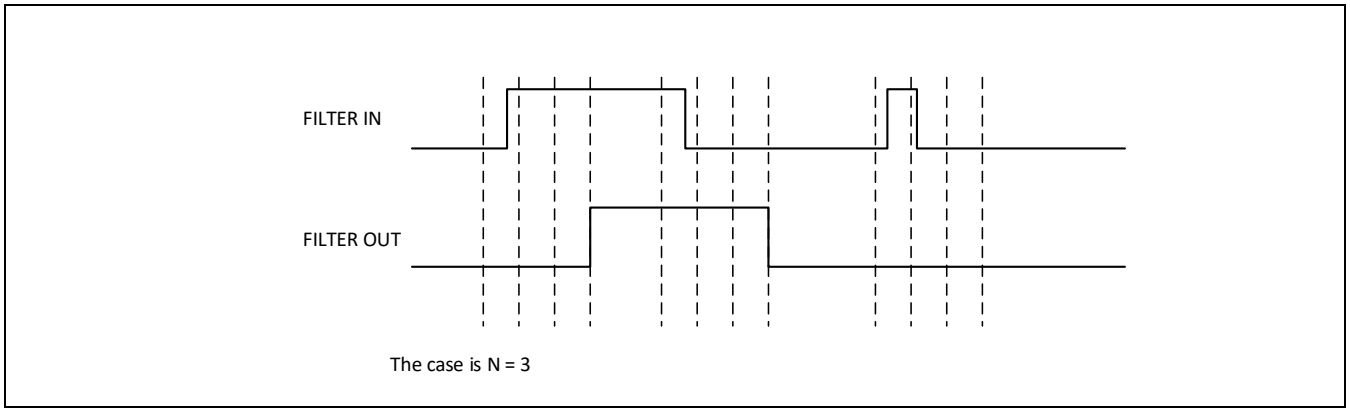


Figure 16-2 外部时钟输入通道的数字滤波器原理

14.3.1.3 内部时钟

LPT的内部时钟可以通过LPT_CEDR的CSS位选择PCLK的4分频，或者系统ISCLK，IMCLK，EMCLK。当选择低频时钟时，可以实现超低功耗的计时，用来对系统进行定时唤醒。

14.3.2 时基控制

14.3.2.1 概述

作为LPT主要的控制模块，时基控制模块由一个16位的计数器和相应的自动重载寄存器组成。模块的主要功能有：

- 确定时基计数器（LPT_CNT）的频率，或者控制事件触发的周期。
- 根据计数器值产生事件触发PWM输出

时基模块的寄存器包括：

- 计数器寄存器（LPT_CNT）：在每个计数时钟周期增加
- 周期寄存器（LPT_PRDR）：计数器周期控制寄存器。

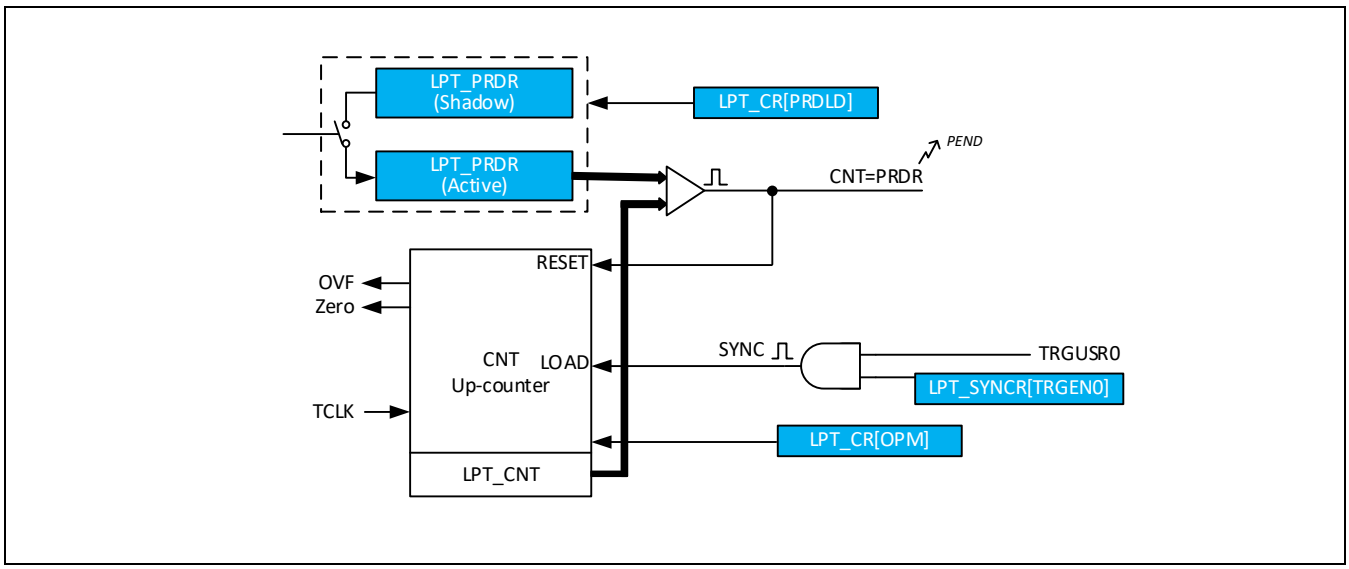


Figure 16-3 计数器时基模块

计数器的计数周期由周期寄存器（LPT_PRDR）的设置值决定。计数器只支持一种计数模式：

- 递增模式（Up-Counting Mode）：

  在递增模式下，时基计数器从0x0000开始递增计数，一直计数到周期设置值（LPT_PRDR）。当计数值等于周期设置值时，时基计数器被复位，重新开始从0x0000进行新一轮计数。

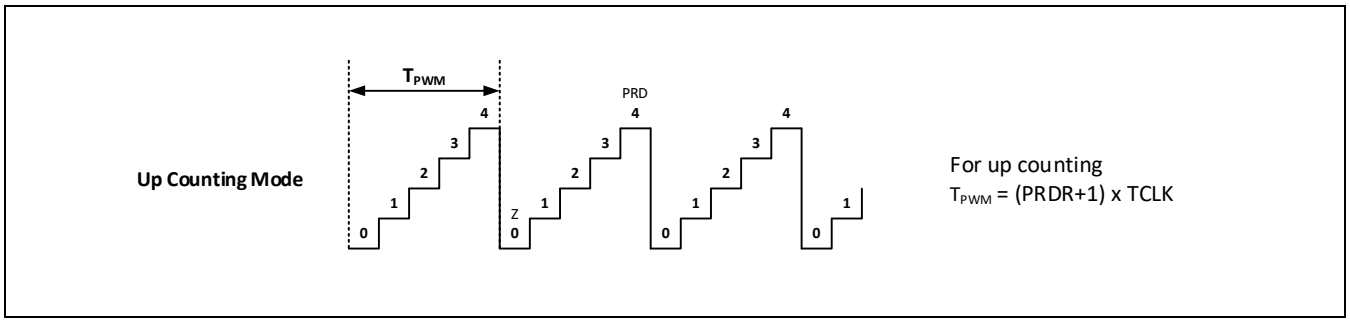


Figure 16-4 计数器工作模式

14.3.2.2 计数器重置和周期设置

在下列条件满足时，计数器值将会被重置。重置发生时，计数器将被重置为PRDR的设置值或者用户指定值。

- 计数值等于PRDR：当周期开始计数且当前计数值等于PRDR值，计数器的值将被重置到0x0000。
- 软件直接更新：通过软件直接写入计数器活动寄存器进行更新。

LPT_PRDR周期寄存器由两个物理寄存器组成：活动寄存器（Active）和影子寄存器（Shadow）。影子寄存器的值通过硬件同步到活动寄存器中，保证对内部活动寄存器更新操作和计数周期同步。活动寄存器直接参与计数器控制事件的产生；影子寄存器作为数据缓冲，为活动寄存器提供临时的数据保存。影子寄存器值不会直接影响硬件控制动作，而是根据预设策略，在特定时间将缓冲的内容传送到活动寄存器中。这样的机制，避免了由于软件非同步地对寄存器操作而引起的硬件输出错误。活动寄存器和影子寄存器共享同一个物理访问地址，当前读写操作的对象是活动寄存器还是影子寄存器，可以通过LPT_CR[PRDLD]控制位进行选择。当影子寄存器被屏蔽时，对PRDR的写入值，会直接改变活动寄存器的值，而对PRDR读取时，将直接返回活动寄存器的值。

- **PRDR寄存器的Shadow模式**

PRDR的缓冲（Shadow Register）在LPT_CR[PRDLD]控制位等于1的时候有效。在此配置下，CPU对PRDR的读写操作对象为PRDR的影子寄存器。当时基计数器值等于PRDR时，或者SYNC触发时，影子寄存器的值被硬件自动载入到活动寄存器（Active Register）中。在缺省配置下，只有时基计数器值等于PRDR时，自动载入才会发生，用户可以通过配置LPT_CR[PRDLD]控制位进行修改。

- **PRDR寄存器的立即加载模式**

在立即加载模式下（LPT_CR[PRDLD]=0），CPU对PRDR的读写操作对象是PRDR的活动寄存器。任何对PRDR的更新操作将被直接反应到活动寄存器中。

在对PRDR进行立即更新时，需要注意考虑当前的计数器值。若将PRDR更新到一个比当前计数器值小的值，将导致计数器在后续计数过程中都不会发生周期结束事件，计数器将一直计数到整个计数器溢出后重新开始计数。

计数器在停止计数后，不会自动清除计数值，而是保留当前的计数值，需要通过软件进行清零。

14.3.3 计数器数值比较控制

14.3.3.1 概述

计数器数值比较控制模块实时比较当前计数器的计数值和比较值寄存器（CMP）或者周期寄存器（PRDR）的值，当计数值相等时，比较控制模块将产生一个相应的事件触发。主要特性如下：

- 支持的触发事件和触发条件：
  - CNT = CMP： 时基计数器当前值等于计数器比较值寄存器的值
  - CNT = PRDR：时基计数器当前值等于周期寄存器PRDR寄存器的值
- PWM的波形控制基于CMP和PRDR，数值相等时PWM输出翻转
- 当CMP值由0到PRDR+1进行调整时，可以获得0到100%的PWM占空比输出（注意：需要获得100%的占空比，需要设置CMP值>PRDR值）
- 比较值寄存器和周期值寄存器都具有影子寄存器功能，以防止PWM输出产生毛刺

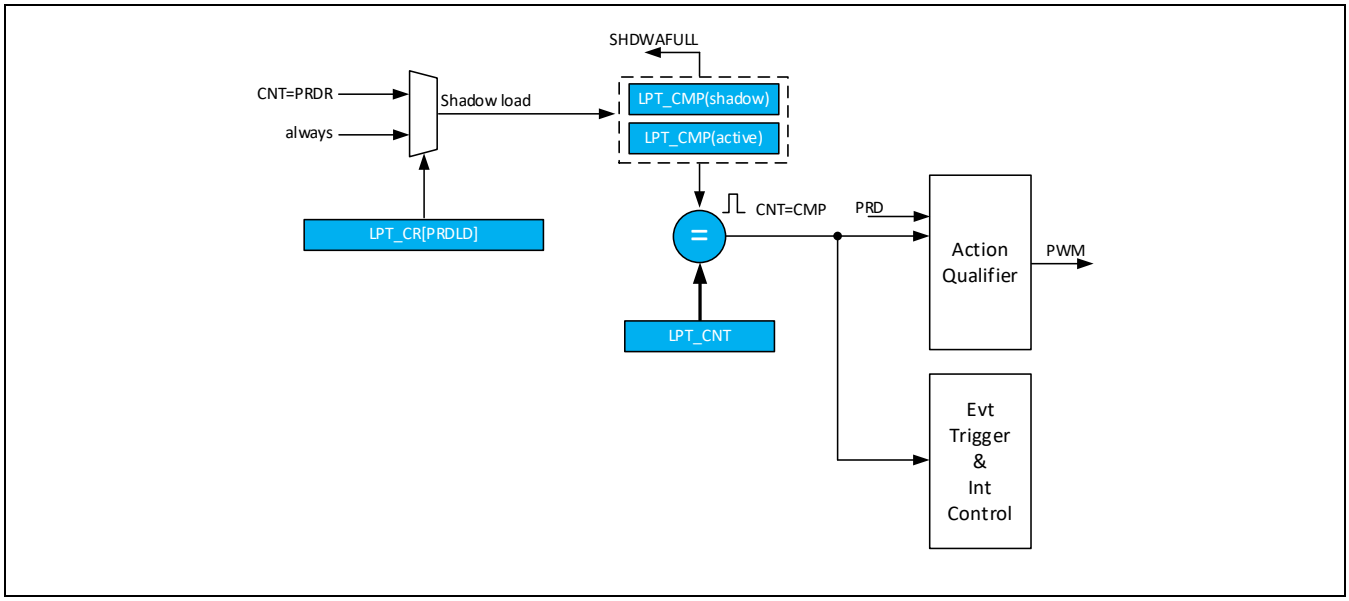


Figure 16-5 计数器值比较控制

14.3.3.2 比较值寄存器载入方式

CMP有相应的Shadow寄存器，在缺省设置下，所有对CMP寄存器的读写对象都是影子寄存器。影子寄存器的使能可以通过LPT_CR[CMPLD]控制位进行设置。当Shadow模式被禁止时，所有对CMP寄存器的操作将直接作用到内部活动寄存器上。

- **CMP寄存器的Shadow模式**

  当Shadow模式使能时，Shadow寄存器中的内容将在CNT=PRDR周期寄存器的值时，被自动传送到活动寄存中。

- **CMPx寄存器的立即加载模式**

  在立即加载模式下，对CMP的操作直接影响活动寄存器。

14.3.4 单次触发模式

单次触发模式是一种特殊的工作模式，在此模式下，计数器在外部触发事件发生时，只产生一个延时和脉宽可编程的脉冲信号。计数器在启动后只进行一个周期的计数，在周期结束后，计数器Freeze。设置单次触发模式，可以通过寄存器LPT_CR[OPM]控制位进行设置。在计数器Freeze状态下，计数器保持当前的计数值，直到有新的触发条件被检测到。

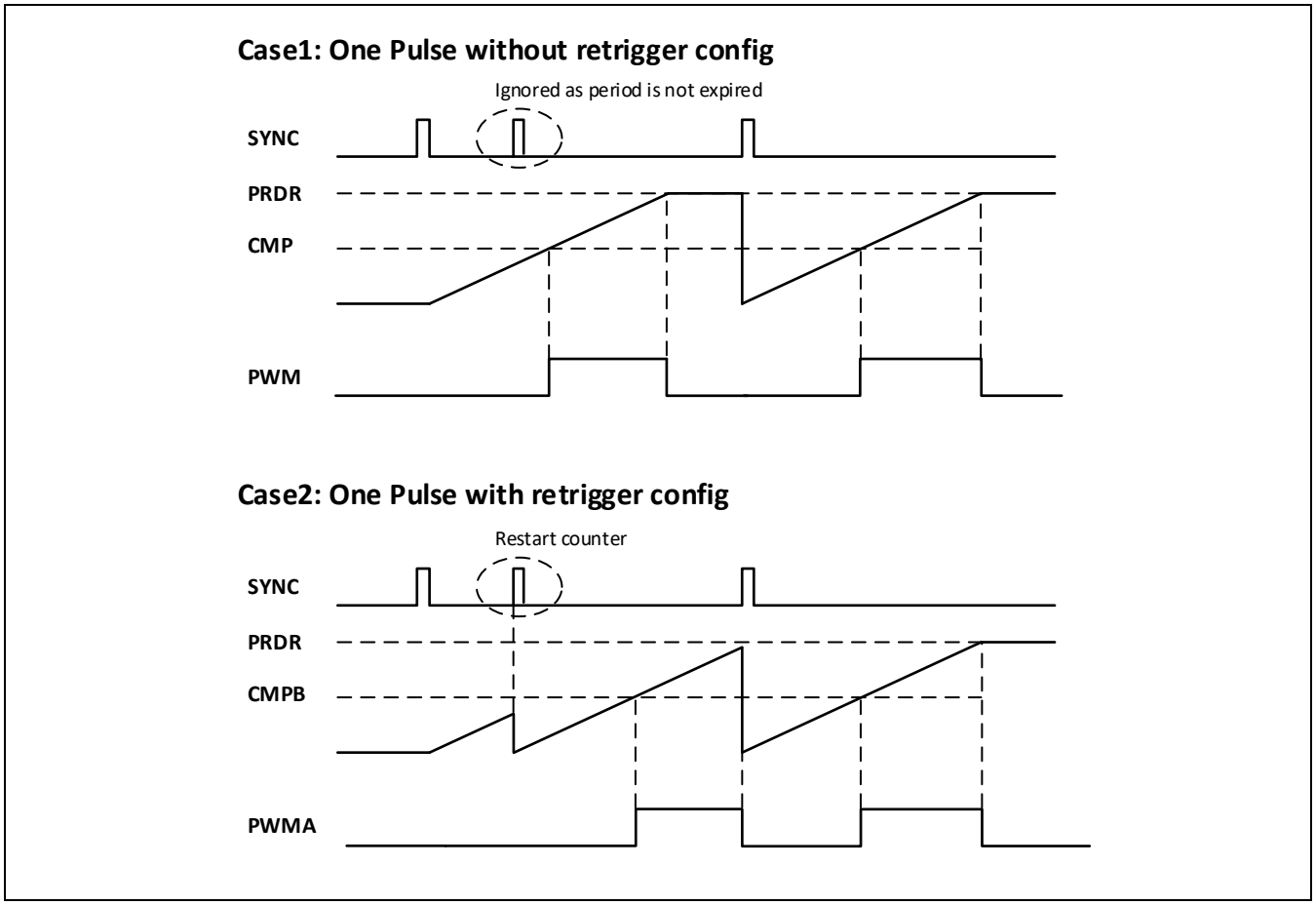


Figure 16-6 单次触发

单次触发模式工作时，缺省情况下，无论当前计数周期是否结束，当有新的触发被检测到时，计数器将开始重新计数。在某些应用中，需要在被触发后的一个计数周期内禁止新的触发，以保证得到一个完整的周期波形输出。在这种条件下，可以通过设置触发控制寄存器LPT_SYNCR中的OSTMD控制位，将触发模式设置为一次性触发；或者将外部触发信号通过窗口滤波器滤波，屏蔽指定时间内的触发输入。

14.3.5 同步触发

同步触发功能用于在多个外设间通过硬件自动耦合同步不同外设的工作。LPT通过同步输入接口接收来自于其他外设的触发信号，不同的触发端口对应独立的同步任务。当相应输入接口被触发，相对应的同步任务即被激活。同样，LPT的同步输出接口，可用于产生对其他外设的任务的触发信号。

14.3.5.1 同步触发输入接口

LPT支持模块间的同步触发功能，可以支持的触发功能只有：

- 重置和启动计数器

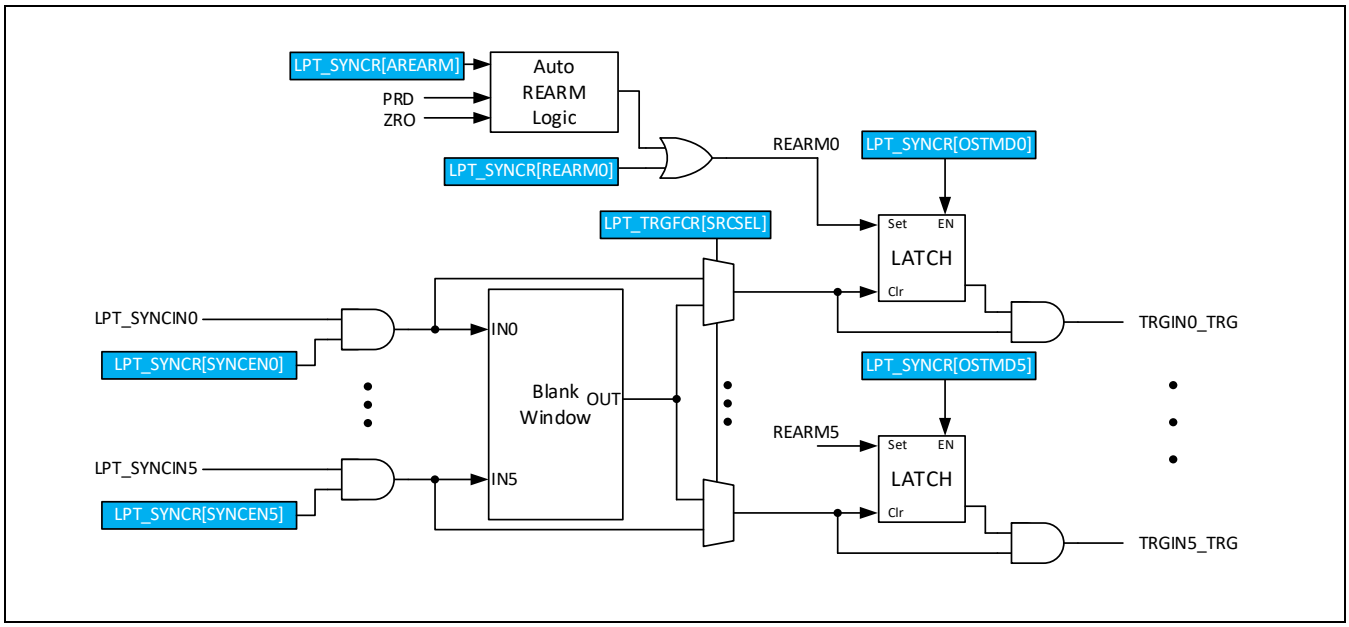


Figure 16-7 同步触发输入 (当前LPT只支持一个源LPT_SYNCIN0)

每一个独立触发源被定义为TRG端口，通过LPT_SYNCR寄存器可以独立控制触发源的使能。触发源的输入为ETCB模块的输出，通过ETCB可以定义某个外设作为当前TRG端口的触发信号源。具体配置参考ETCB章节。在触发输入接口中，包含一个事件滤波器，可以选择一个事件输入端口作为滤波器输入，对该事件进行滤波处理。

每一个触发端口可以工作在两种工作模式：连续触发或者单次触发模式。在单次触发模式下，只运行一次触发发生，在检测到发生一次触发后，该端口将被禁止，直到软件重置该端口（REARM）后，才允许新的触发发生。重置端口也可以通过硬件自动完成，在设置LPT_SYNCR[AREARM]后，在周期结束或者开始时，硬件会自动重置REARM，以保证在一个周期内只发生一次触发。

重置和启动计数器（SYNCIN0）

当该端口被触发，下列动作将被同时执行：

- 时基计数器（CNT）被重置。
- 时钟分频器重置。当该触发条件发生时，时钟分频器将重新开始计数
- 所有具有Shadow寄存器的控制寄存器将自动从Shadow更新Active寄存器

14.3.5.2 事件滤波器

在同步触发输入接口中，有一个事件滤波器，事件滤波器是一个时间窗口滤波器，它可以在一个特定的窗口周期内阻止输入信号的通过，或者只有在窗口内允许信号通过，从而实现去除干扰事件的目的。例如当模拟比较器的

输出作为外部触发源时，可以滤除由于模拟比较器的抖动而产生的错误触发。可以选择任意一个SYNCIN端口作为事件滤波器的输入。

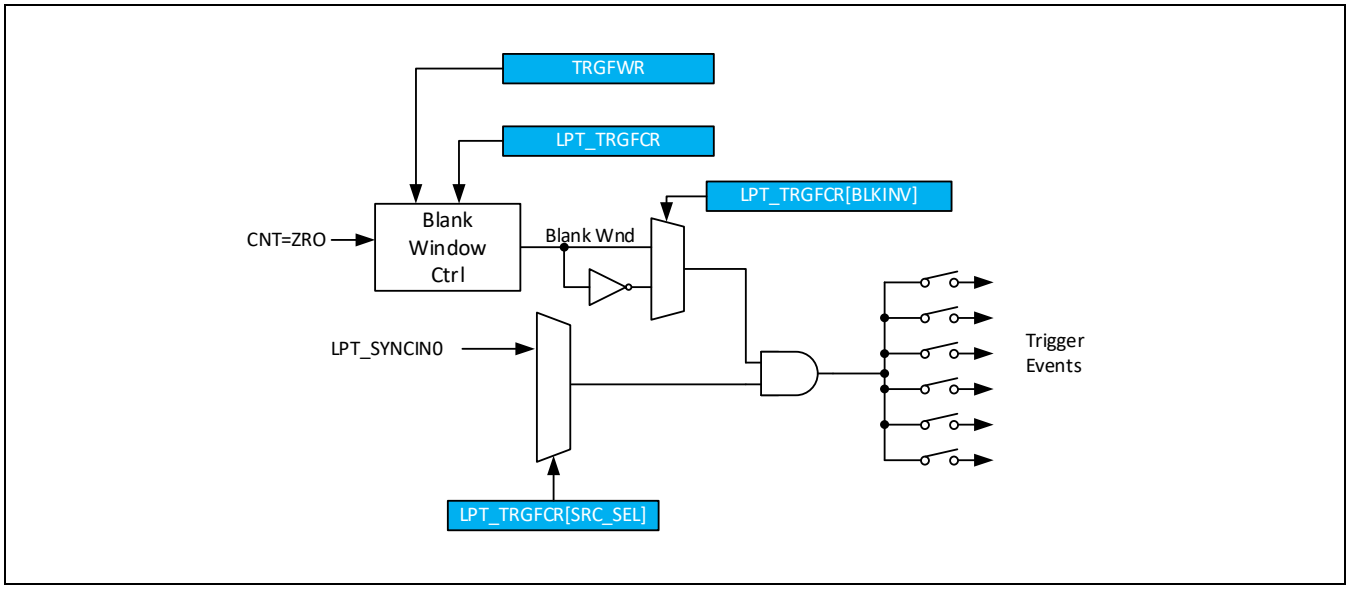


Figure 16-8 滤波器框图

如果使能窗逻辑被打开，在使能窗口内，或者窗口外，将禁止输入事件通过。窗口的激活时间可以为CNT=ZRO。窗口的延时和宽度可以通过TRGFWR进行设置。

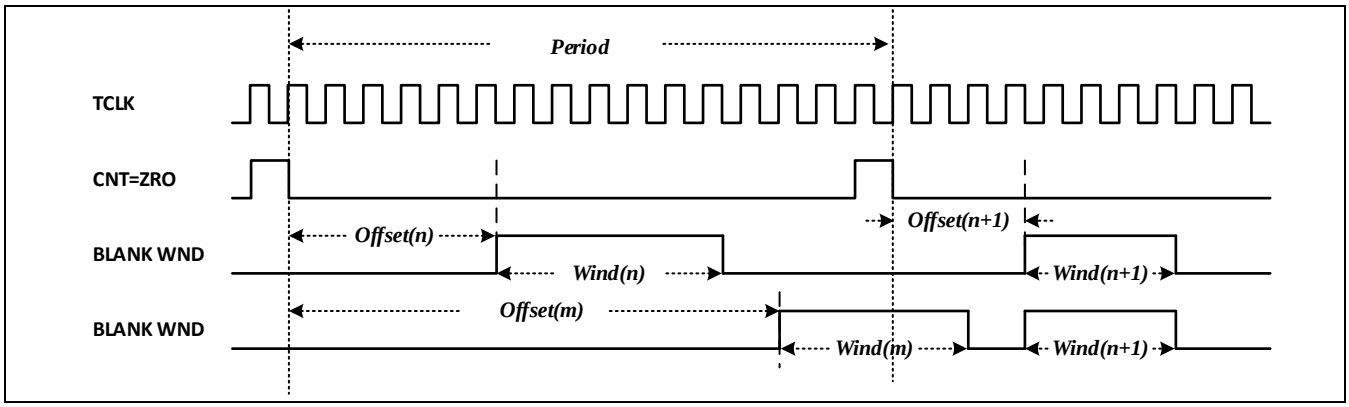


Figure 16-9 滤波器时序

14.3.5.3 同步触发输出接口

同步触发输出接口支持4路事件触发输出，每个LPT中断对应一个事件输出端口。可以通过LPT_EVTRG控制寄存器选择LPT中的任意一个事件作为每个中断的触发信号，同时中断触发信号可以通过ESYNxOE控制位使能输出到ETCB (LPT_TRGOUT源)，并且发送给其他外设作为触发信号。

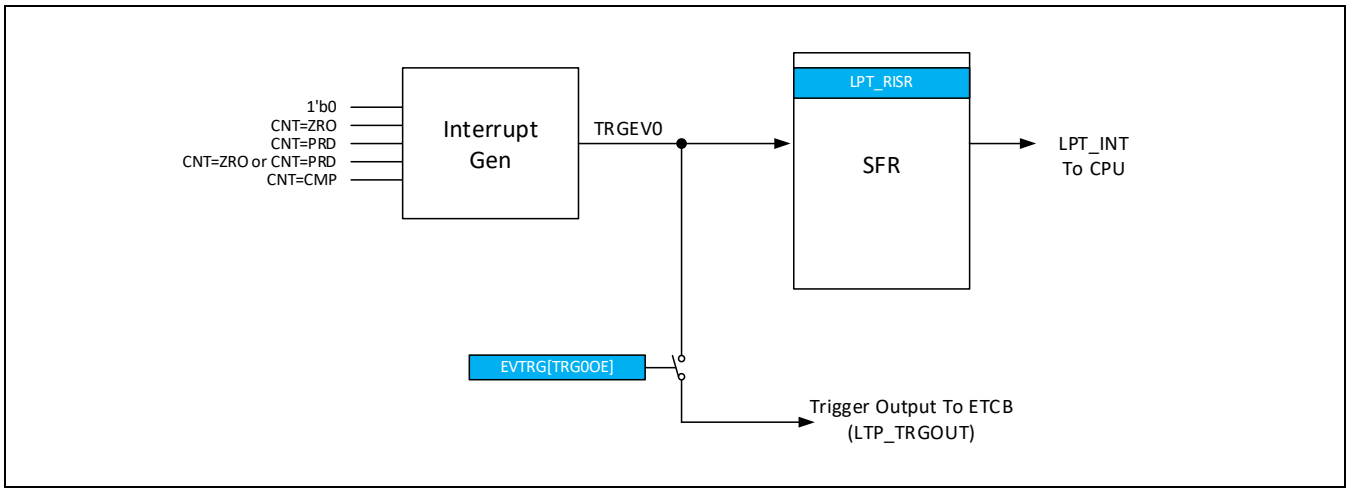


Figure 16-10 同步触发输出

14.3.6 中断触发

中断触发基于各种触发事件，可以支持对触发事件计数的基础上决定是否产生中断请求。中断触发控制支持如下几种配置：

- 每次发生触发事件产生中断
- 每N次触发事情产生一次中断，N最大支持到15

中断模块一共支持4个中断源，每一个中断触发都支持事件计数器。中断的具体触发条件，可以通过LPT_EVTRG寄存器进行选择。通过配置LPT_EVPS可以配置每个中断事件计数器在计数到多少时产生一个中断请求。当中断发生次数等于LPT_EVPS[TRGEVxPRD]控制位设置值时，将产生一次中断触发。计数器最大支持15个事件的计数，在中断请求发生后计数器将自动清除。当前已经计数的事件个数可以通过LPT_EVPS[TRGEVxCNT]进行读取。CNT计数器具有Shadow功能，在缺省设置下，对CNT的读写时，操作对象是Shadow寄存器，Shadow寄存器的值会在同步事件发生时，或者计数器值等于PRD设置值时自动载入到活动寄存器中。当Shadow模式被禁止时，对CNT的操作直接影响活动寄存器的值。

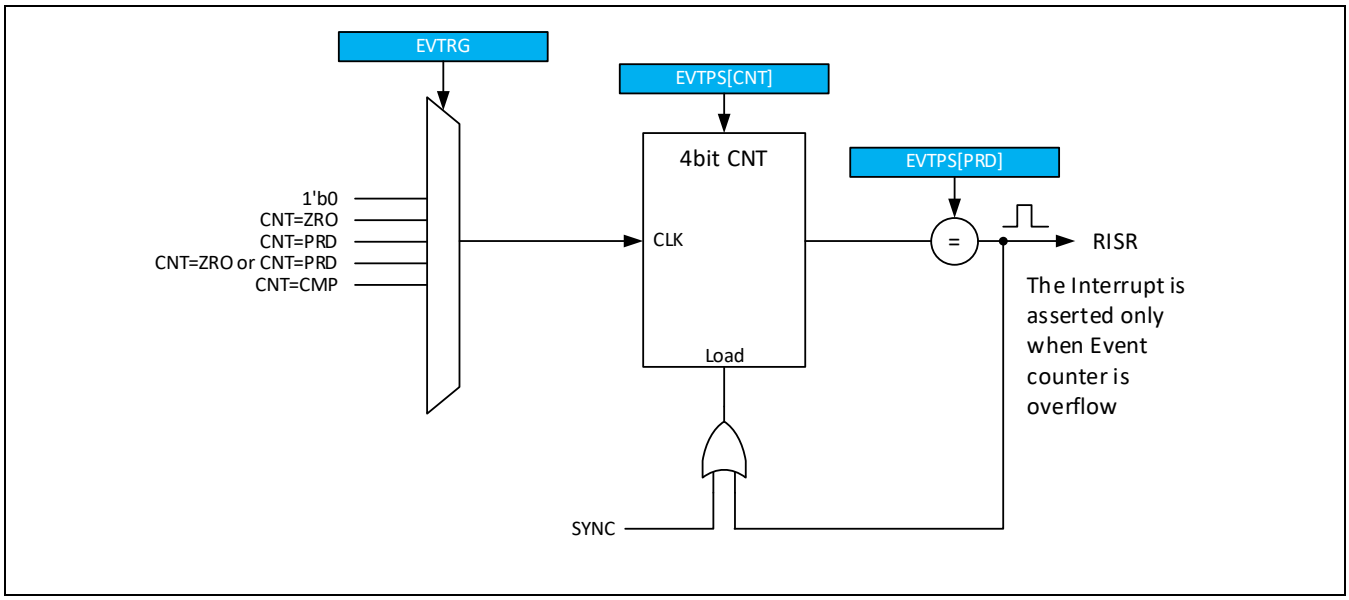


Figure 16-11 事件计数器

14.4 寄存器说明

14.4.1 寄存器表

- Base Address: 0x4006_1000

Register	Offset	Description	Reset Value
LPT_CEDR	0x000	ID和时钟控制寄存器	0xBE9C0000
LPT_RSSR	0x004	启停控制寄存器	0x00000000
LPT_PSCR	0x008	时钟分频控制寄存器	0x00000000
LPT_CR	0x00C	控制寄存器	0x00000000
LPT_SYNCR	0x010	同步控制寄存器	0x00000000
LPT_PRDR	0x014	周期设置寄存器	0x00000000
LPT_CMP	0x018	比较值寄存器	0x00000000
LPT_CNT	0x01C	时基计数器寄存器	0x00000000
LPT_TRGFCR	0x020	触发事件滤波窗控制寄存器	0x00000000
LPT_TRGFWR	0x024	触发事件滤波窗时序寄存器	0x00000000
LPT_EVTRG	0x028	事件触发选择寄存器	0x00000000
LPT_EVPS	0x02C	事件触发计数寄存器	0x00000000
LPT_EVSWF	0x030	软件强制触发事件控制寄存器	0x00000000
LPT_RISR	0x034	原始中断状态寄存器	0x00000000
LPT_MISR	0x038	中断状态寄存器	0x00000000
LPT_IMCR	0x03C	中断使能控制寄存器	0x00000000
LPT_ICR	0x040	中断清除寄存器	0x00000000

14.4.2 LPT_CEDR (ID和时钟控制寄存器)

- Address = Base Address + 0x0000, Reset Value = 0xBE98_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																FLTCKPRS								RSVD	SHDWSTP		CSS			DBGEN	CLKEN
1	0	1	1	1	1	1	0	1	0	0	1	1	1	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	RW	R	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CLKEN	[0]	RW	时基计数器的时钟使能控制。 0h：计数器计数时钟禁止。 1h：计数器计数时钟使能。
DBGEN	[1]	RW	调试使能控制。调试使能时，在CPU被调试器挂起时，时基计数器的计数时钟同时也被挂起。 0h：调试禁止 1h：调试使能
CSS	[4:2]	RW	计数器时钟源选择位。 0h：PCLK/4 1h：ISCLK 2h：IMCLK/4 3h：EMCLK 4h：LPT_IN的上升沿 5h：LPT_IN的下降沿 其他：保留
SHDWSTP	[6]	RW	START控制位的Shadow功能使能控制。 0h：Shadow模式 1h：Immediate模式
FLTCKPRS	[15:8]	RW	数字滤波器的时钟分频控制。 数字滤波器的时钟频率为PCLK/(FLTCKPRS+1)
IDCODE	[31:16]	RW	当前LPT模块的版本信息。

14.4.3 LPT_RSSR (启停控制寄存器)

- 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																SRR				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	W	W	W	W	R	R	R	R	R	R	R	R	R	R	W	R W

Name	Bit	Type	Description
START	[0]	RW	计数器启动控制位。 0h：当写‘0’时，停止计数器 1h：当写‘1’时，启动计数器 当对START位进行读取时，返回当前计数器工作状态 0h：计数器处于IDLE状态 1h：计数器正在工作 当LPT_CR[SWSYNEN]控制位为低时，START控制位用于控制LPT的启动，当LPT启动后，再次写入START将被忽略；当LPT_CR[SWSYNEN]控制位为高时，START控制位用于软件触发同步事件，每次对START的写入，会产生一次外部Sync事件（等同于SYNCR中的TRGUSR0触发）。
SRR	[15:12]	W	软件复位控制位。 当对当前控制位写入‘0x5’时，TIMER模块会被复位。复位后，所有寄存器都恢复为RESET状态。

14.4.4 LPT_PSCR (时钟分频控制寄存器)

- 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																												PSC			
0	0	0	0	0	0	0	0	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
PSC	[3:0]	RW	时钟分频控制。 TCLK作为时基模块的计时时钟和工作时钟。 0x0：不分频 0x1：2分频 0x2：4分频 0x3：8分频 0x4：16分频 0x5：32分频 0x6：64分频 0x7：128分频 0x8：256分频 0x9：512分频 0xA：1024分频 0xB：2048分频 0xC：4096分频 其他：保留

14.4.5 LPT_CR (控制寄存器)

- Address = Base Address + 0x000C, Reset Value = 0001_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														CMPLD	PSCLD	FLTDEB			RSVD	RSVD	FLTIPSCLD	RSVD		OPM	WAV_POL	PRDLD	IDLEST	SWSYNEN	RSVD		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	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	RW	RW	RW	RW	RW	R	RW	W	R	R	R	RW	RW	RW	RW	RW	R	R

Name	Bit	Type	Description
SWSYNEN	[2]	RW	软件使能同步触发使能控制（RSSR 中 START 控制位）。 0h：设置SW START控制只用于启动。 1h：设置SW START控制用于启动和以产生一次外部触发的方式重新启动。
IDLEST	[3]	RW	波形输出被停止时，输出端口的缺省状态。 0h：高阻输出 1h：低电平输出
PRDLD	[4]	RW	PRDR活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值。 0h：立即更新，所有对PRDR操作直接作用于活动寄存器 [1] 1h：PRDR活动寄存器更新发生在周期结束时
POL	[5]	RW	波形输出极性控制。 0h：当CNT的值小于CMP时，输出高电平 1h：当CNT的值小于CMP时，输出低电平
OPM	[6]	RW	计数器单次触发工作模式选择。 0h：连续计数工作模式 1h：单次触发工作模式 其他：保留
FLTIPSCLD	[10]	RW	数字滤波器初始化控制。对该控制写'1'可以初始化数字滤波器计数器，计数器值被初始化为CEDR[FLTCKPRS]中的设置值。 0h：无效 1h：执行初始化

FLTDEB	[15:13]	RW	数字滤波去抖控制。此控制定义了滤波器监测的步数，只有连续N次监测结果一致时，滤波器才输出有效的电平翻转。滤波器的采样时钟频率在通过CKS控制位定义。 000b：Bypass 001b：N = 2 010b：N = 3 011b：N = 4 100b：N = 6 101b：N = 8 110b：N = 16 111b：N = 32
PSCLD	[16]	RW	PSCR寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值。 0h：PSCR活动寄存器更新发生在启动或者外部触发时 1h：PSCR活动寄存器更新发生在周期结束，启动或者外部触发时
CMPLD	[17]	RW	CMP寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值。 0h：立即更新，所有对CMP操作直接作用于活动寄存器 [2] 1h：CMP活动寄存器更新发生在周期结束时

注意 [1]：如果更新的周期值比更新前小，且立即更新发生时计数器已经超过更新的周期值，计数器将继续计数直到溢出，然后更新才会生效。

注意 [2]：如果更新的比较值比更新前小，且立即更新发生时计数器已经超过更新的比较值，本周期将不会发生match事件。

14.4.6 LPT_SYNCR (同步控制寄存器)

- 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	AREARM	RSVD													REARM0	RSVD							OSTMD0	RSVD							SYNCEN0
0	0	0	0	0	0	0	0	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	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	R	R	R	R	RW	R	R	R	R	R	R	R	RW

Name	Bit	Type	Description
SYNCEN0	[0]	RW	外部同步触发使能控制。 0：禁止当前触发输入通道 1：使能当前触发输入通道 LPT_SYNCIN0：外部Sync事件 (来自ETCB模块)
OSTMDx	[8]	RW	一次性同步触发模式选择。 0h：连续触发模式 1h：一次性触发模式 当该输入通道被设置为一次性触发模式后，在一次触发事件被检测到后，该通道将不允许后续的触发事件通过，直到被软件重置（REARM）后才允许新的触发事件通过。
REARMx	[16]	RW	在一次性同步触发模式下，软件重置当前通道状态控制位。 当读取时，返回当前通道状态 0h：允许触发 1h：已经检测到触发，不允许后续触发 当写入时， 0h：无效 1h：清除当前通道状态，并允许新的触发
AREARM	[30]	RW	硬件自动REARM控制位。 0：禁止硬件自动REARM 1：周期结束时，自动REARM

14.4.7 LPT_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	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
PRDR	[15:0]	RW	时基控制周期寄存器。 此控制位决定了PWM输出波形的周期值。通过设置LPT_CR[PRDLD]可以选择Shadow到Active载入的触发条件。

14.4.8 LPT_CMP (比较值寄存器)

- 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																CMP															
0	0	0	0	0	0	0	0	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
CMP	[15:0]	RW	比较值寄存器。 此寄存器有Shadow寄存器，只有在周期结束时才会对活动寄存器进行更新。

14.4.9 LPT_CNT (时基计数器寄存器)

- 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																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
CNT	[15:0]	RW	时基计数器寄存器。 对CNT读取时，返回当前计数器值。对CNT写入时，将直接更新CNT的计数值。 CNT计数器没有Shadow寄存器，CPU的写入将直接影响当前计数器值。

14.4.10 LPT_TRGFCR (触发事件滤波窗控制寄存器)

- 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
																								CROSSMD	RSVD		BLKINV	RSVD			SRCSEL
0	0	0	0	0	0	0	0	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	R	R	RW	R	R	R	RW

Name	Bit	Type	Description
SRC_SEL	[0]	RW	滤波模块的输入信号选择。 0h：禁止滤波 1h：使能LPT_SYNCIN滤波
BLKINV	[4]	RW	窗口使能反转控制。 0h：窗口不反转 1h：窗口反转
CROSSMD	[7]	RW	允许滤波窗跨越多个TB的周期。 缺省条件下，当滤波窗在周期结束时若任然有效，将跨过周期点，一直持续到窗口计数器溢出。当禁止跨周期时，在周期结束时，窗口计数器将被停止。 0h：禁止跨周期 1h：允许跨周期

14.4.11 LPT_TRGFWR (触发事件滤波窗时序寄存器)

- 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
WINDOW																OFFSET															
0	0	0	0	0	0	0	0	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
OFFSET	[15:0]	RW	滤波窗的OFFSET设置。 此16bit控制位定义了从周期起始位置开始计数多少个TCLK后，开始有效的滤波窗口。OFFSET的Shadow寄存器在周期起始时，载入到Active寄存器中，并重新开始计数。
WINDOW	[31:16]	RW	滤波窗的宽度设置。 此16bit控制位定义了滤波窗的宽度，窗口宽度是基于TCLK的计数值。当OFFSET计数器溢出时，WINDOW计数器被重置，并开始计数直到溢出。当OFFSET计数器溢出，但WINDOW状态已经激活时，WINDOW计数器不会重置。在应用时必须注意此条件的设置。

14.4.12 LPT_EVTRG (事件触发选择寄存器)

- 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											TRG0OE	RSVD				RSVD												TRG0SEL			
0	0	0	0	0	0	0	0	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	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
TRG0SEL	[3:0]	RW	TRGEV0事件的触发源选择。 0000：禁止TRGOUT触发输出 0001：当 CNT = ZRO 产生TRGEV0事件 0010：当 CNT = PRD 产生TRGEV0事件 0011：当 CNT = ZRO or CNT = PRD 产生TRGEV0事件 0100：当 CNT = CMP时，产生TRGEV0事件 其他：保留
TRG0OE	[20]	RW	外部触发端口TRGOUT0使能 0h：禁止触发输出 1h：允许触发输出

14.4.13 LPT_EVPS (事件触发计数寄存器)

- 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												TRGEV0CNT				RSVD												TRGEV0PRD			
0	0	0	0	0	0	0	0	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	RW	RW	RW	RW

Name	Bit	Type	Description
TRGEV0PRD	[3:0]	RW	TRGEV0事件计数的周期设置。 当TRGEV0事件发生次数满足周期时，才产生TRGEV0触发事件
TRGEV0CNT	[19:16]	RW	TRGEV0事件计数器设置。 读取时，返回当前事件计数器值； 写入时，直接更新事件计数器值。

14.4.14 LPT_EVSWF (软件强制触发事件控制寄存器)

- 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																															EV0SWF
0	0	0	0	0	0	0	0	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
EV0SWF	[0]	RW	软件产生一次EV0的触发 0h：写入‘0’无效 1h：软件产生一次触发

14.4.15 LPT_RISR (原始中断状态寄存器)

- 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																													PEND	MATCH	TRGEV0
0	0	0	0	0	0	0	0	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
TRGEV0	[0]	R	TRGEV0中断请求原始标志状态
MATCH	[1]	R	MATCH中断请求原始标志状态
PEND	[2]	R	PEND中断请求原始标志状态
原始中断标志表示中断事件发生，通过设置IMCR中相应位，可以允许该中断请求CPU中断。原始中断标志位需要通过软件清除。 0h：该中断未置位 1h：该中断已置位			

14.4.16 LPT_MISR (中断状态寄存器)

- 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																													PEND	MATCH	TRGEV0
0	0	0	0	0	0	0	0	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
TRGEV0	[0]	R	TRGEV0中断请求标志状态
MATCH	[1]	R	MATCH中断请求标志状态
PEND	[2]	R	PEND中断请求标志状态
中断标志表示CPU中断请求的状态，通过写ICR寄存器可以清除该标志位。 0h：该中断未置位 1h：该中断已置位			

14.4.17 LPT_IMCR (中断使能控制寄存器)

- 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
RSVD																													PEND	MATCH	TRGEV0
0	0	0	0	0	0	0	0	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
TRGEV0	[0]	RW	TRGEV0中断使能控制位
MATCH	[1]	RW	MATCH中断使能控制位
PEND	[2]	RW	PEND中断使能控制位
CPU中断请求使能控制。当该控制位使能时，允许触发CPU中断。 0h：禁止该中断 1h：允许该中断			

14.4.18 LPT_ICR (中断清除寄存器)

- 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
RSVD																													PEND	MATCH	TRGEV0
0	0	0	0	0	0	0	0	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	W	W	W	W

Name	Bit	Type	Description
TRGEV0	[0]	W	清除TRGEV0原始中断状态位
MATCH	[1]	W	清除MATCH原始中断状态位
PEND	[2]	W	清除PEND原始中断状态位
中断清除控制位。 对该寄存器写‘0’时，无效；对该寄存器写‘1’时，清除相应中断标志位 读取时，总是返回‘0’			

15 窗口型看门狗 (WWDT)

15.1 概述

窗口型看门狗（Window Watchdog）作为可靠性保护逻辑，用于监测当前程序运行状况。当外部干扰或不可预见的逻辑错误发生时，造成当前程序运行错误，看门狗逻辑可以在预设时间周期结束时产生系统复位信号。看门狗计数器可以通过软件刷新以防止计数器溢出而产生复位，如果刷新事件发生在计数器值大于预设窗口计数值时，也将会触发复位信号。也就是刷新必须在预设的时间窗口内进行才有效。

注：如果系列内芯片不具有本外围，那它就不具备本章描述的相关资源。具体参考芯片的数据手册。

15.1.1 主要特性

- 8 位可编程递减计数器
- 预设计数器时钟分频器：Div (1/2/4/8 x 4096)
  - 计数器时钟基于 PCLK 工作
  - 分频器的基础分频为 PCLK/4096
  - 可选择基于 4096 分频后的二次分频：DIV1，DIV2、DIV4 和 DIV8
- 产生复位的条件：
  - 递减计数器计数器值小于 0x80
  - 软件刷新计数器发生在预设窗口外
  - 软件写入的刷新计数器的数值小于 0x80
- 报警中断：当计数器值等于 0x80 时，可产生中断

15.2 功能描述

15.2.1 模块框图

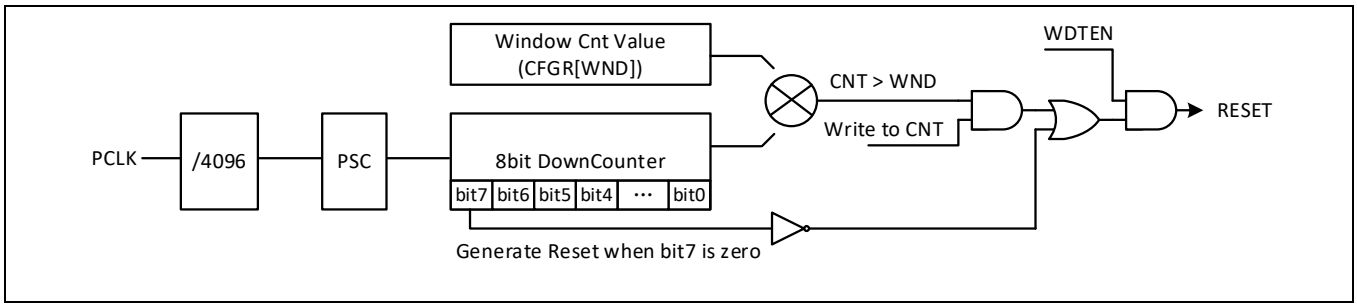


Figure 17-1 模块结构示意图

15.2.2 基本功能描述

看门狗缺省状态（系统复位后）为禁止状态，只有通过软件写入CR[WDTEN]后，才能使能。当看门狗被使能以后，不能通过WDTEN再关闭，只有系统复位发生后，看门狗逻辑被复位后，看门狗才会停止。

当看门狗被使能后（CR[WDTEN]被设置为高），看门狗计数器开始工作。当计数器值从0x80计数器到0x7F时，即计数器的最高位变成0时，将产生系统复位信号。当软件刷新计数器值时，当前计数器值大于窗口预设值（CFGR[WND]）时，也会触发系统复位。所以对看门狗计数器刷新时必须满足两个条件：

- 窗口条件：写CNT时，当前计数器值小于WND预设值
- 刷新数据：写入CNT的值必须在0xFF和0x80之间

15.2.3 时钟控制

看门狗工作时钟为系统PCLK时钟，当PCLK不工作时，看门狗计数器将暂停，直到PCLK恢复后才能继续工作。计算看门狗的溢出时间使用如下公式进行：

$$T_{WWDG} = T_{PCLK} \times 4096 \times 2^{PSC} \times (CNT[6:0]+1)$$

其中：T_{PCLK}为系统PCLK时钟周期，PSC为CFGR[PSC]的设置值，CNT为CR[CNT]寄存器设置值。

具体溢出时间可以参考下面表格中的数据

Table 17-1 最小和最大溢出时间(PCLK=24MHz)

PSC	最小溢出时间（CNT[6:0]=0x00）	最大溢出时间（CNT[6:0]=0x7F）
0	170.67 us	21.85 ms
1	341.33 us	43.69 ms
2	682.67 us	87.38 ms
3	1365.33 us	174.76 ms

在连接ICE Debugger时，通过设置使能调试模式CFGR[DBGEN]，将WWDT的工作时钟在调试暂停时挂起。调试使能后，通过CDK调试时，一旦CPU被挂起，WWDT的计数器也同时被暂停，以防止计数器溢出造成调试复位。

15.2.4 计数器操作

WWDT内置一个8位的Free-running递减计数器。当WWDT使能时，必须保证计数器的最高位bit7为'1'，以防止立即产生RESET事件。计数器计数范围被限制在 0xFF ~ 0x80之间，CNT[6:0]控制位定义了计数器的计数次数。通过设置CNT[6:0]可以定义看门狗的溢出时间。

WWDT具有窗口功能，可以限定对计数器进行刷新的时间窗口，以防止由于程序错误而连续刷新，导致看门狗监测功能被异常屏蔽。窗口的设置可以通过CFGR[WND]控制位进行设置。当刷新CNT计数器时，如果当前计数器值大于窗口设置值，将产生复位信号。所以为避免刷新时产生复位，必须保证刷新时CNT计数值小于窗口设置。

CNT的最高位bit7，可以用于产生软件复位。当CNT最高位被写入'0'时，会立即触发一个软件复位事件。

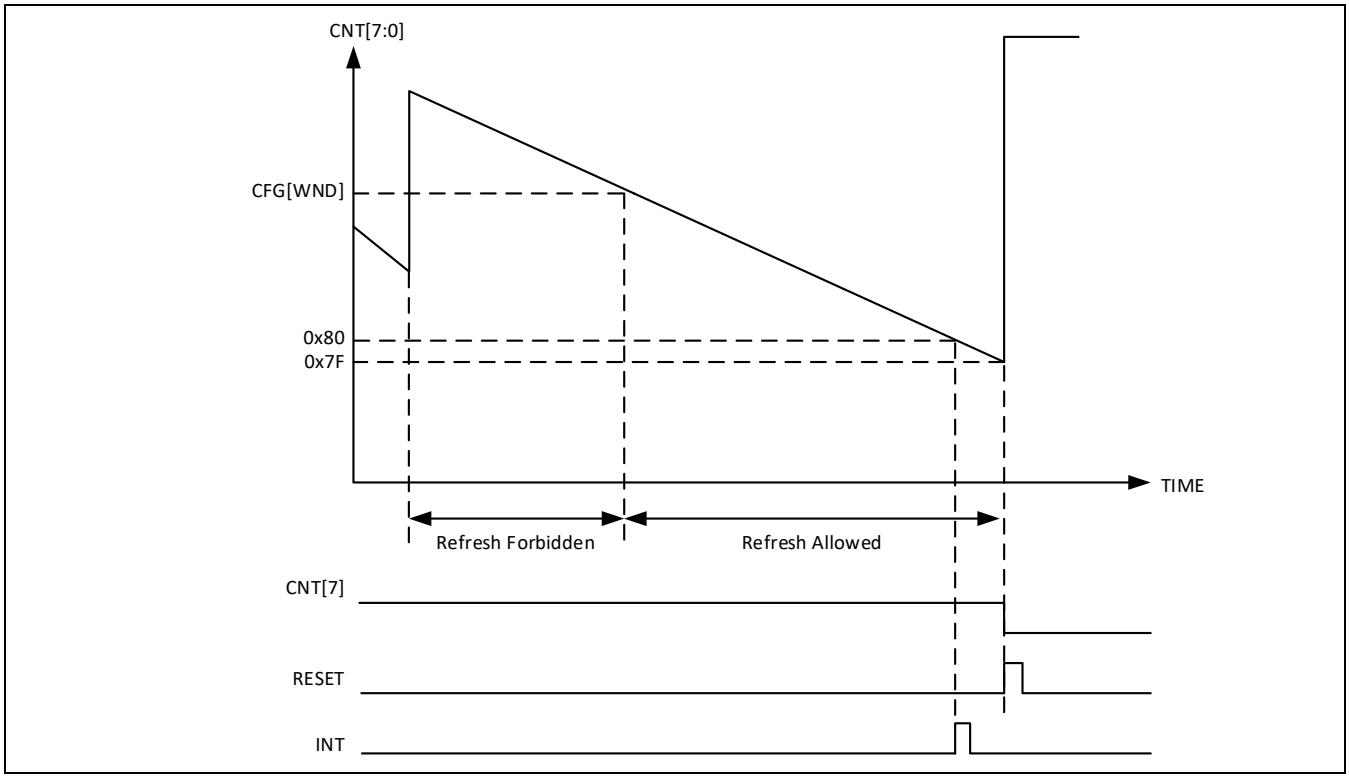


Figure 17-2 计数器工作示例

15.2.5 中断控制

WWDT的计数器计数到0x80时，可以产生一个警告中断。通过这个中断的服务程序，可以在将要发生的复位事件前作出一些处理，例如安全操作，现场保护和日志处理等。或者在中断服务程序中进行系统检查，然后确定是否刷新CNT以避免复位。

需要注意，在应用中当WWDT的中断优先级没有被置为最高，有可能导致WWDT中断服务程序被其他更高优先级的中断服务程序所阻挡，从而导致系统复位。

中断的使能通过IMCR寄存器进行控制。无论中断是否使能，中断的原始标志始终可以通过RISR寄存器进行查询。通过对ICR寄存器写入'1'，可有清除中断的标志位。寄存器说明

15.3 寄存器表

Base Address: 0x 4006_2000

Register	Offset	Description	Reset Value
WWDT_CR	0x0000	Control Register	0x000000FF
WWDT_CFGR	0x0004	Configuration Register	0x000000FF
WWDT_RISR	0x0008	Raw Interrupt Status Register	0x00000000
WWDT_MISR	0x000C	Masked Interrupt Status Register	0x00000000
WWDT_IMCR	0x0010	Interrupt Masking Control Register	0x00000000
WWDT_ICR	0x0014	Interrupt Pending Clear Register	0x00000000

15.3.1 WWDT_CR (控制寄存器)

- Address = Base Address +0x0000, Reset Value = 0x0000_00FF

31	30	29	28	27	26	25	24	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																							WDTEN	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	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	R	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CNT	[7:0]	RW	计数器刷新值。 写入时，将当前计数器设置为CNT的值。 读取时，返回当前计数器值。
WDTEN	[8]	RW	看门狗使能控制位。 0h：禁止看门狗 1h：使能看门狗 该使能控制位一旦使能后，不能通过软件关闭。需要复位后才能恢复初始禁止状态。

15.3.2 WWDT_CFGR (配置寄存器)

- Address = Base Address +0x0004, Reset Value = 0x0000_00FF

31	30	29	28	27	26	25	24	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	PSC		WND							
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	1	1	1
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
WND	[7:0]	RW	窗口预设值。 当CNT的当前计数值大于窗口设置时，任何对CNT的刷新操作都会触发复位事件，窗口预设值寄存器没有缓冲，设置后立即生效。
PSC	[9:8]	RW	计数器时钟分频控制位。分频控制是基于PCLK/4096后的分频。 0h：PCLK/4096 1h：PCLK/4096/2 2h：PCLK/4096/4 3h：PCLK/4096/8
DBGEN	[10]	RW	调试模式控制位。 0h：禁止调试模式 1h：使能调试模式

15.3.3 WWDT_RISR (原始中断状态寄存器)

- 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
RSVD																															EVI
0	0	0	0	0	0	0	0	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
EVI	[0]	RW	EVI中断请求原始标志状态

15.3.4 WWDT_MISR (中断状态寄存器)

- Address = Base Address +0x000C, Reset Value = 0000_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																															EVI
0	0	0	0	0	0	0	0	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
EVI	[0]	RW	EVI中断请求标志状态

15.3.5 WWDT_IMCR (中断使能控制寄存器)

- Address = Base Address +0x0010, Reset Value = 0000_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																															EVI
0	0	0	0	0	0	0	0	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
EVI	[0]	RW	EVI中断使能控制位
CPU中断请求使能控制。当该控制位使能时，允许触发CPU中断。 0h：禁止该中断 1h：允许该中断			

15.3.6 WWDT_ICR (中断清除寄存器)

- 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																															EVI
0	0	0	0	0	0	0	0	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
EVI	[0]	W	清除EVI原始中断状态位
中断清除控制位。 对该寄存器写‘0’时，无效；对该寄存器写‘1’时，清除相应中断标志位 读取时，总是返回‘0’			

16 通用异步收发器 (UART)

16.1 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，支持校验位，每次发送都以一个停止位结束。

注：如果系列内芯片不具有本外围，那它就不具备本章描述的相关资源。具体参考芯片的数据手册。

16.1.1 主要特性

- 可配置的波特率
- 固定的8位发送长度，支持8个单独的收发FIFO
- 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

16.1.2 管脚描述

Table 18-1 UART 管脚描述

管脚名称	功能	I/O类型	有效电平	说明
UART_TX	UART发送数据线	O	–	–
UART_RX	UART接收数据线	I	–	–

16.2 功能描述

16.2.1 模块框图

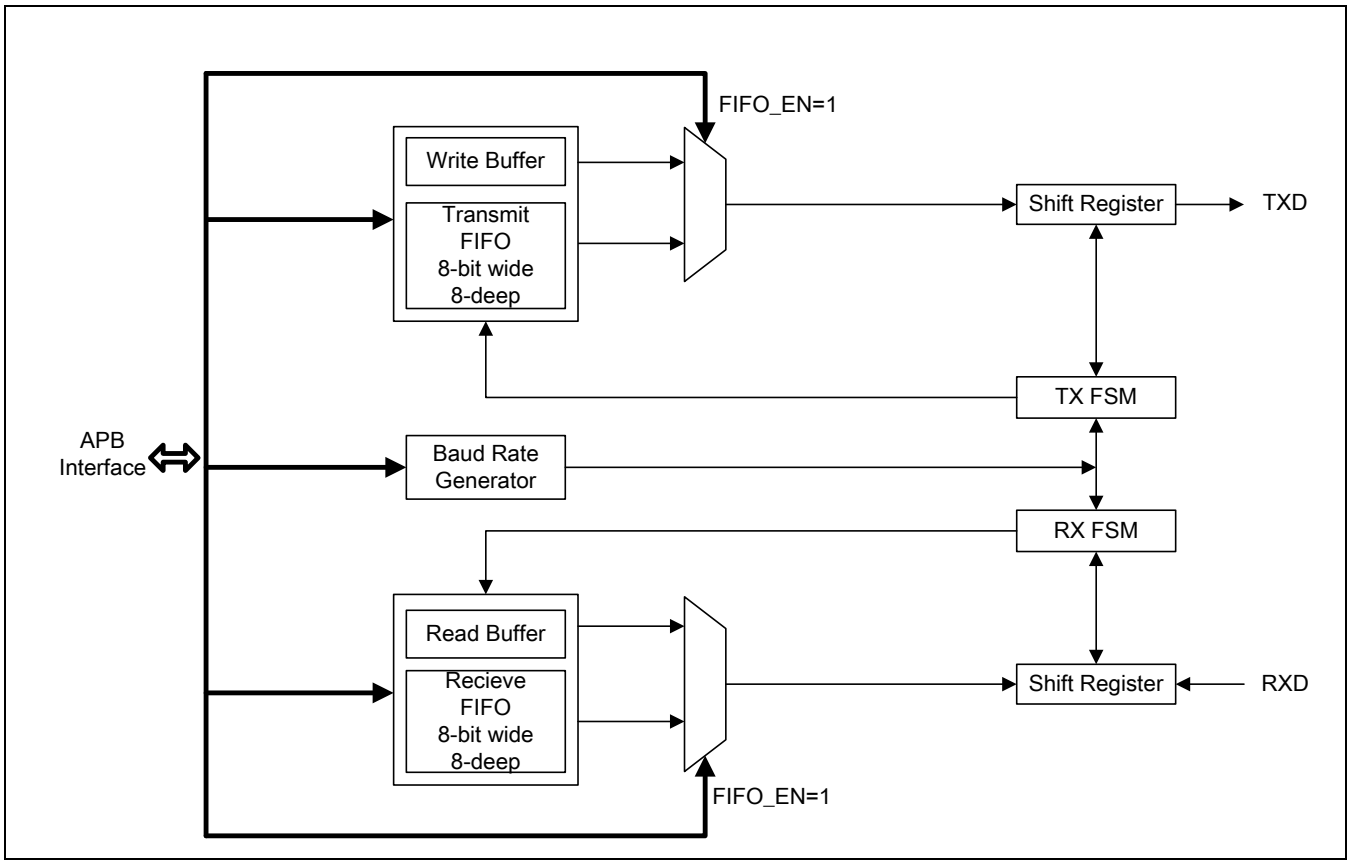


Figure 18-1 UART模块框图

16.2.2 功能说明

16.2.2.1 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 18-2 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%

16.2.2.2 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)。

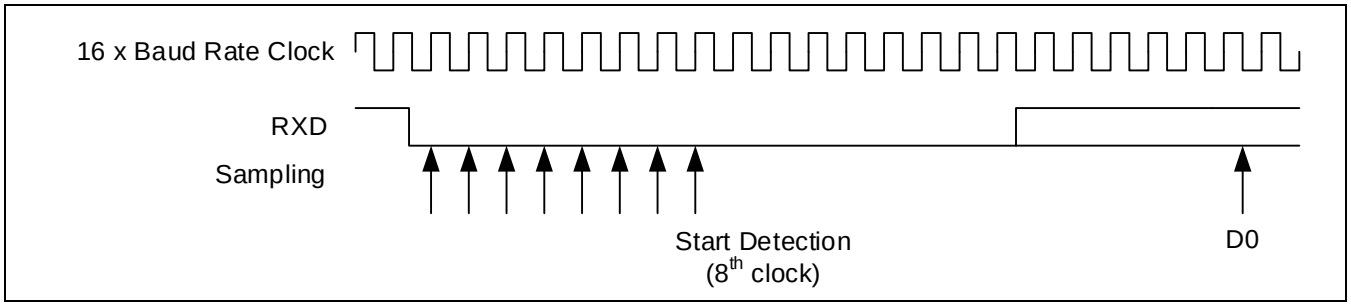


Figure 18-2 起始位检测

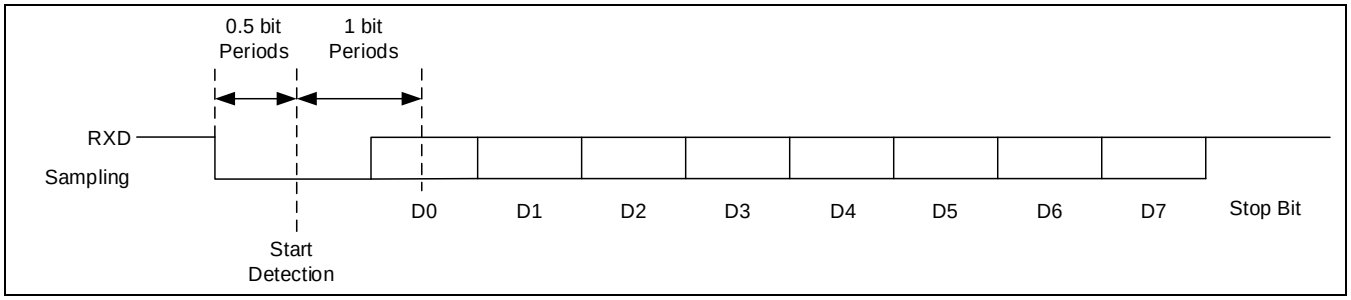


Figure 18-3 接收数据

当模块中的FIFO功能有效时，即当寄存器UART_CTRL中的FIFO_EN设置为1时，UART发送和接收都会分别通过发送FIFO和接收FIFO。接收FIFO是一个8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通过总线读出。

16.2.2.3 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 18-4 数据发送

当模块中的FIFO功能有效时，即当寄存器UART_CTRL中的FIFO_EN设置为1时，UART发送和接收数据都会分别通过发送FIFO和接收FIFO。发送FIFO是一个8位宽，8地址深的先进先出缓冲区。在需要通过UART_TX管脚串行发送数据时，先将数据写入发送FIFO，之后发送逻辑会自动将FIFO缓冲中的数据逐一发出。

16.2.2.4 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，数据为(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

16.2.2.5 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。

当模块中的 FIFO 使能时，即 UART_CTRL 中的 FIFO_EN 置为 1 时，UART 可以产生 3 个中断：

- UARTRXINTR_FIFO: UART 接收 FIFO 中断
- UARTTXINTR_FIFO: UART 发送 FIFO 中断
- UARTRORINTR_FIFO: UART 接收溢出中断

用户可以通过 UART_CTRL 寄存器中的行应为使能或禁止这些中断。

- UARTINTR_FIFO
  当接收 FIFO 占用到一定数量之后就会触发该中断。这个数量可以通过 UART_CTRL 中的 RXIFLSEL 位设置。
- UARTTXINTR_FIFO
  当发送 FIFO 的占用为 4 或者更少时，会触发该中断。数据的发送可以用两种方法操作。一是数据可以在中断前就写入发送 FIFO，二是中断使能后在发送 FIFO 中断服务子程序中写入数据。
- UARTRORINTR
  当接收 FIFO 满后还收到了数据，会触发该中断。这个中断发生说明 FIFO 溢出了，此时新接收的数据会覆盖接收移位寄存器，而不会写入 FIFO 中。

16.3 寄存器说明

16.3.1 寄存器表

Base Address: UART0 - 0x4008_0000

UART1 - 0x4008_1000

UART2 - 0x4008_2000

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

16.3.1.1 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	-

16.3.1.2 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																							RFF	RNE	TNF	TFE	PARITY_SR	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_SR	[4]	R/W	PARITY 错误状态位 0 = 校验没有错误 1 = 校验出错 1 = 清除校验错误状态位(写)	0
TFE	[5]	R	发送FIFO是否为空状态位 0 = 发送FIFO非空 1 = 发送FIFO为空	0
TNF	[6]	R	发送FIFO是否已满状态位 0 = 发送FIFO已满 1 = 发送FIFO未满	0
RNE	[7]	R	接收FIFO是否为空状态位 0 = 接收FIFO为空	0

			1 = 接收FIFO非空	
RFF	[8]	R	接收FIFO是否已满状态位 0 = 接收FIFO未满 1 = 接收FIFO已满	0

16.3.1.3 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
DBGEN	RSVD											INT_TX_DONE_EN	INT_OVER_FIFO	RSVD	RXIFSEL		INT_FIFO		FIFO_EN	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
RW	R	R	R	R	R	R	R	R	R	R	R	W	W	R	W	W	W	W	W	W	W	W	W	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	PARITY中断使能/禁止 0 = 禁止PARITY中断 1 = 使能PARITY中断	0
PARITY	[10:8]	W	校验位类型 0XX：无校验位 100：偶校验	0

<table>
<tr><td></td><td></td><td></td><td>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)</td><td></td></tr>
<tr><td>FIFO_EN</td><td>[11]</td><td>W</td><td>FIFO模块有效，接收和发送模式下都需要经过相应的FIFO模块
0= 禁用FIFO
1= 使能FIFO</td><td>0</td></tr>
<tr><td>INT_FIFO</td><td>[13:12]</td><td>W</td><td>[12]：FIFO使能下的TX中断使能/禁止
0= 禁止TX中断
1= 使能TX中断
[13]：FIFO使能下的RX中断使能/禁止
0= 禁止RX中断
1= 使能RX中断</td><td>0</td></tr>
<tr><td>RXIFLSEL</td><td>[16:14]</td><td>W</td><td>接收FIFO中断触发点选择位
<table><tr><td>001</td><td>接收FIFO占用>=1/8</td></tr><tr><td>010</td><td>接收FIFO占用>=1/4</td></tr><tr><td>100</td><td>接收FIFO占用>=1/2</td></tr><tr><td colspan="2">Others=保留</td></tr></table></td><td>001’b</td></tr>
<tr><td>INT_OVER_FIFO</td><td>[18]</td><td>W</td><td>FIFO使能下的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</td><td>0</td></tr>
<tr><td>INT_TX_DONE_EN</td><td>[19]</td><td>W</td><td>发送完成中断使能/禁止
0= 禁止发送完成中断
1= 使能发送完成中断</td><td>0</td></tr>
<tr><td>DBGEN</td><td>[31]</td><td>W</td><td>调试使能
0= 调试禁止
1= 调试使能，进入调试模式后，UART不工作</td><td>0</td></tr>
</table>

16.3.1.4 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												TX_DONE_INT	RSVD											RORMIS	RXMIS	TXMIS	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]	RW	PARITY错误中断 0= PARITY错误中断没发生 1= PARITY错误中断发生(读取) 1= 清除PARITY中断(写)	0
TXMIS	[5]	R	发送FIFO中断 0= TX中断没发生 1= TX中断发生(读取)	0
RXMIS	[6]	R	接收FIFO中断 0= RX中断没发生 1= RX中断发生(读取)	0
RORMIS	[7]	RW	接收FIFO溢出中断	0

			0= 溢出中断没发生 1= 溢出中断发生(读取) 1= 清除溢出中断(写)	
TX_DONE_INT	[19]	RW	发送完成中断 0= 发送完成中断没发生 1= 发送完成中断发生(读取) 1= 清除发送完成中断(写)	0

16.3.1.5 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

17 串行外设接口 (SPI)

17.1 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。
注：如果系列内芯片不具有本外围，那它就不具备本章描述的相关资源。具体参考芯片的数据手册。

17.1.1 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

17.1.1.1 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

17.1.1.2 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

17.1.1.3 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

17.1.2 管脚描述

Table 19-1 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	–

17.2 功能描述

17.2.1 模块框图

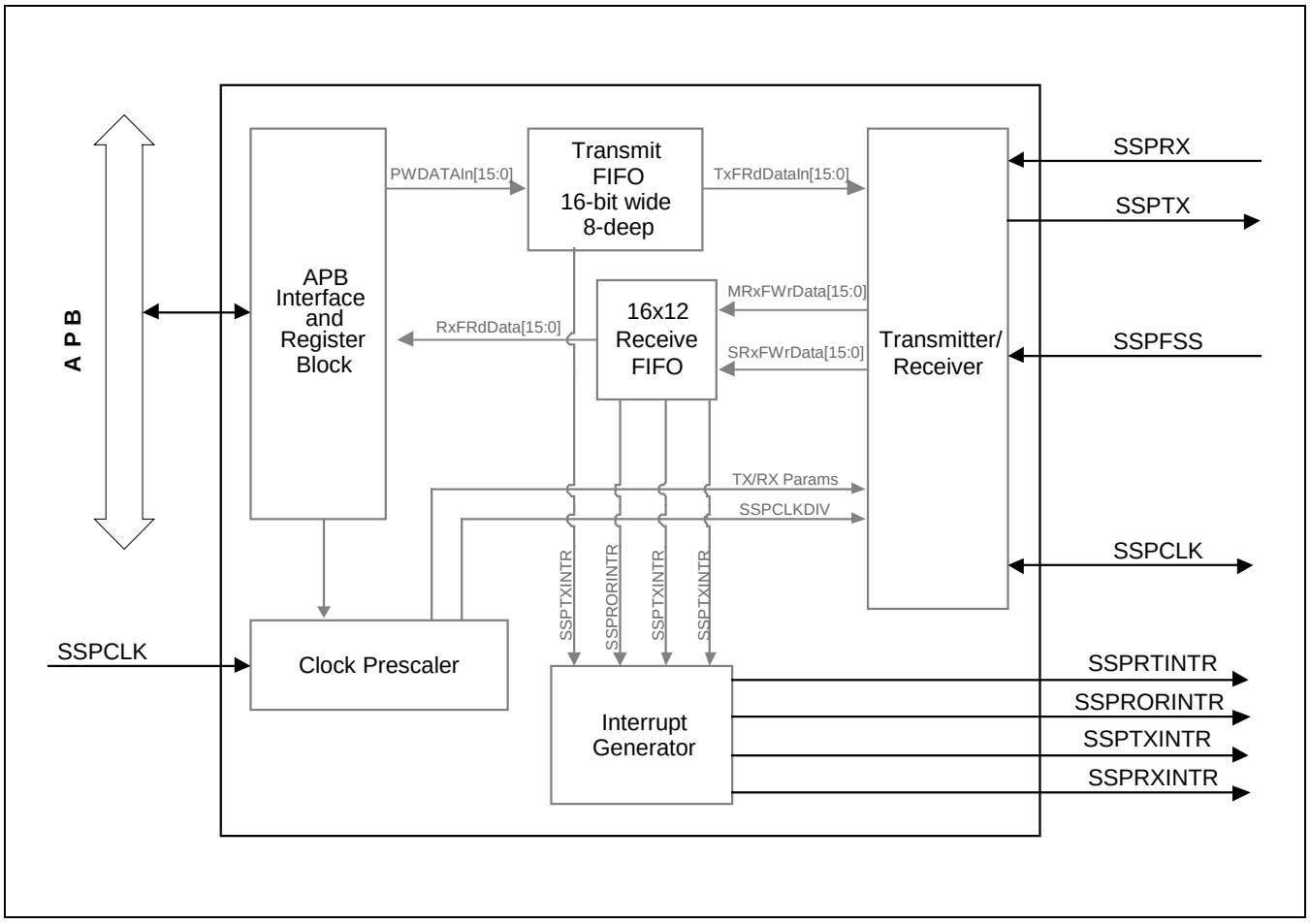


Figure 19-1 SSP 模块框图

17.2.2 功能描述

17.2.2.1 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。

17.2.2.2 SSP功能描述

17.2.2.2.1 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。

17.2.2.2.2 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。

17.2.2.2.3 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。

17.2.2.2.4 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。

17.2.2.2.5 发送和接收单线模式

SSP支持单线收发模式，在某些特殊应用中，SSP可以只使用一个端口进行通讯。SSP_CR1寄存器中的LPMD位用来使能单线模式，只有SSP的主机模式支持该单线功能。在单线模式中，SSP将只使用SPI_MOSI端口进行通讯，SSPRX和SSPTX都将接在SPI_MOSI管脚上，SSPRX将会一直接收SPI_MOSI上的信号，SSPTX则通过SSP_CR1中的LPTXOE位来控制是否连接到SPI_MOSI管脚进行发送。在需要发送时，将LPTXOE置1，SSPTX信号将通过SPI_MOSI管脚将数据发出；在需要接收时，将LPTXOE置0，这样SSPTX发送出去的信号就不会与从机发来的信号在通讯线上产生冲突。

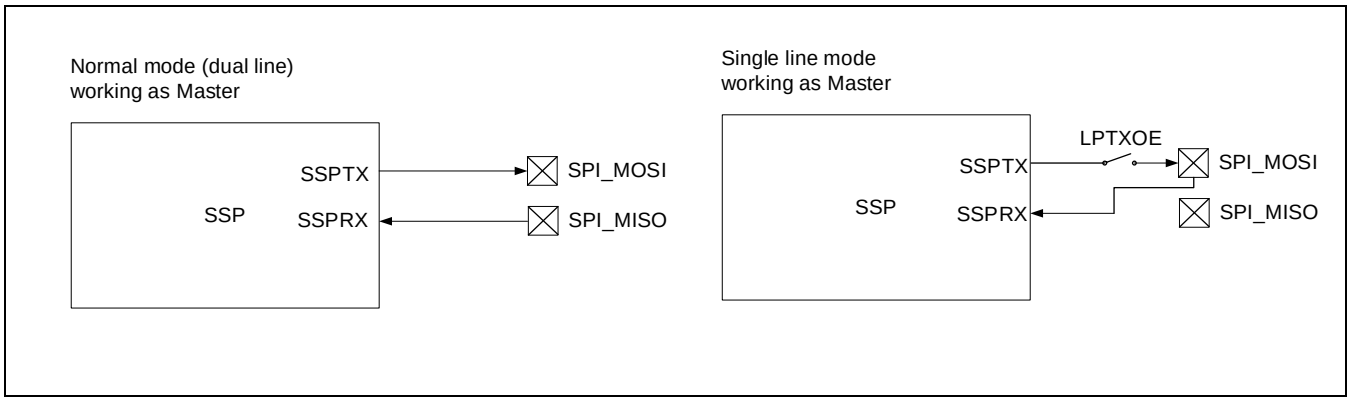


Figure 19-2 SSP 单线通讯模式

17.2.2.2.6 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。

17.2.2.3 SSP操作方法

17.2.2.3.1 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。

17.2.2.3.2 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。

17.2.2.3.3 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) → 2 × $F_{SSPCLKOUT}$ (max) [主机模式]
- F_{PCLK} (min) → 12 × $F_{SSPCLKIN}$ (max) [从机模式]

PCLK的最大频率可以由下面的公式得到：
- F_{PCLK} (max) ← 254 × 256 × $F_{SSPCLKOUT}$ (min) [主机模式]
- F_{PCLK} (max) ← 254 × 256 × $F_{SSPCLKIN}$ (min) [从机模式]

17.2.2.3.4 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。

17.2.2.3.5 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。

17.2.2.3.6 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。

17.2.2.3.7 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。

17.2.2.3.8 Motorola SPI 格式, SPO = 0, SPH = 0

Motorola SPI 格式中 SPO = 0, SPH = 0 的示意图如下：

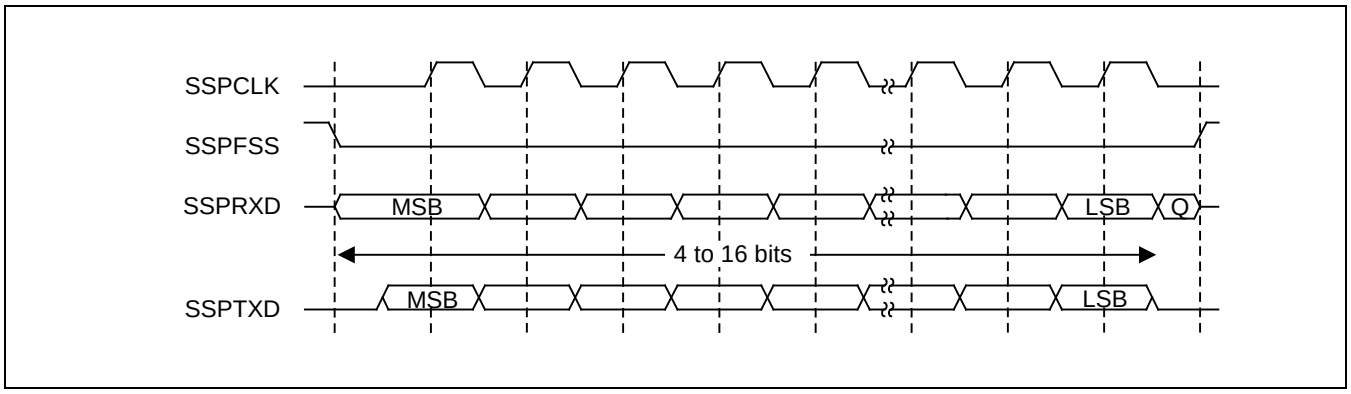


Figure 19-3 Motorola SPI 帧格式 (单次传输) SPO = 0, SPH = 0

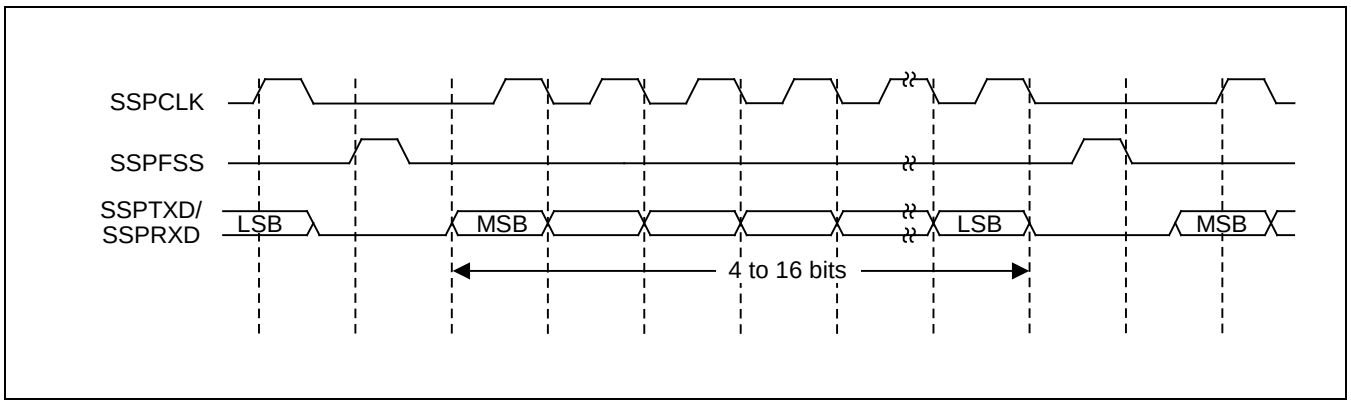


Figure 19-4 Motorola SPI 帧格式 (连续传输) SPO = 0, SPH = 0

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。

17.2.2.3.9 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：

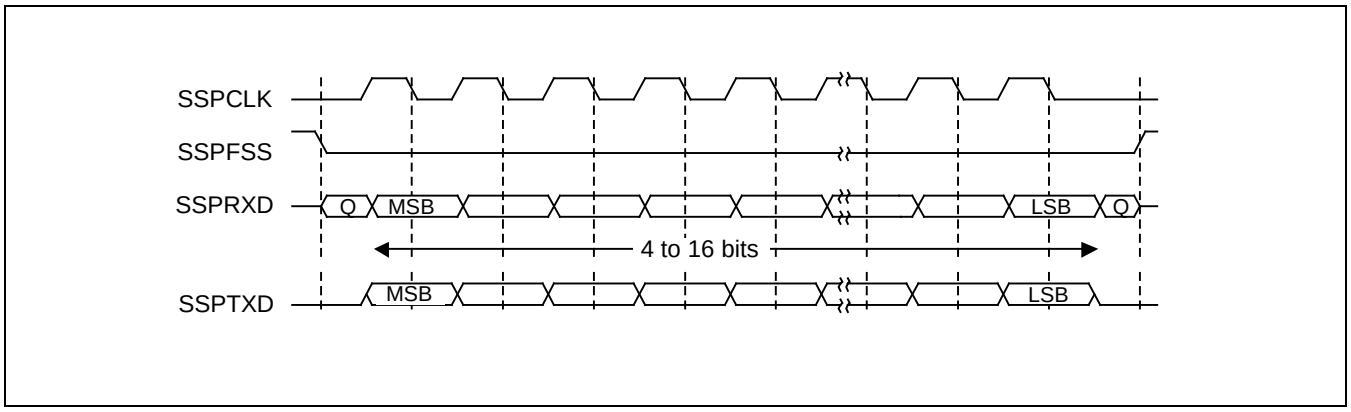


Figure 19-5 Motorola SPI 帧格式， SPO = 0 和 SPH = 1

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。

17.2.2.3.10 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：

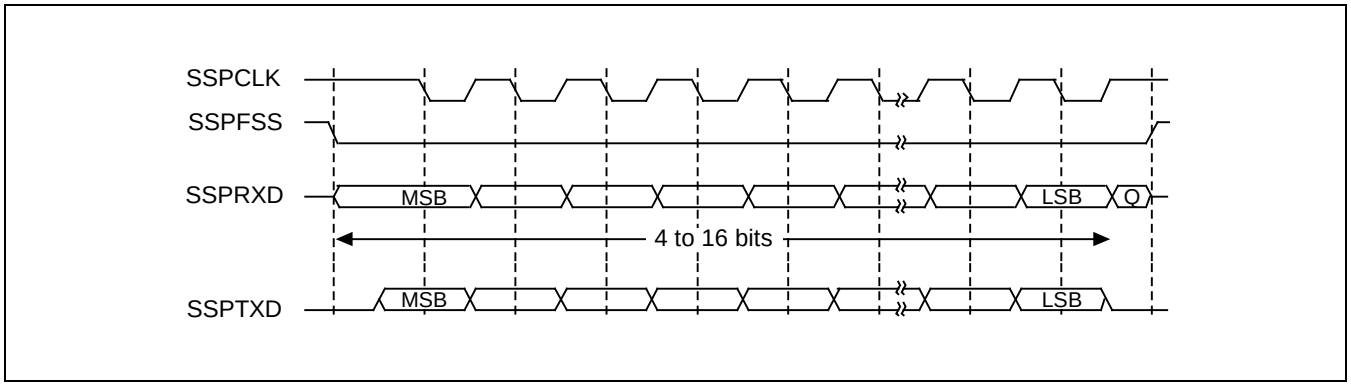


Figure 19-6 Motorola SPI 帧格式 (单次传输)，SPO = 1 和 SPH = 0

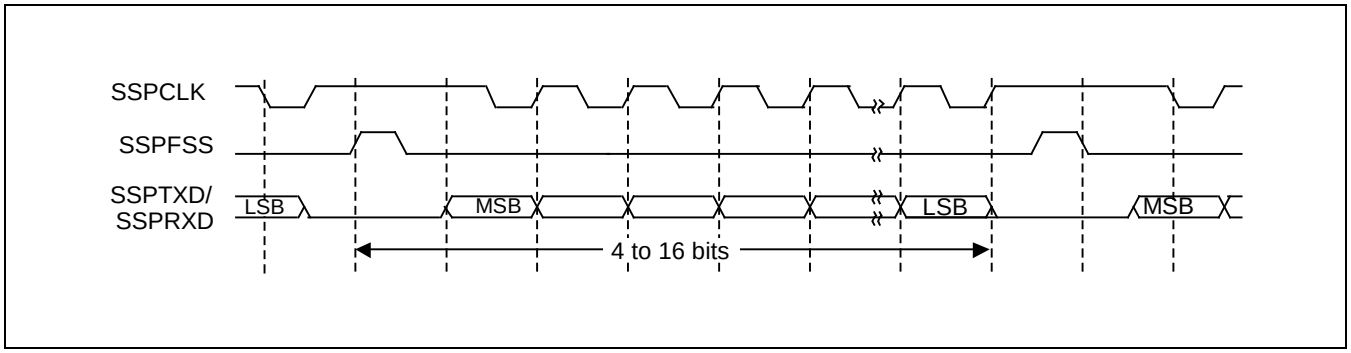


Figure 19-7 Motorola SPI 帧格式 (连续传输)， SPO = 1 和 SPH = 0

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。

17.2.2.3.11 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：

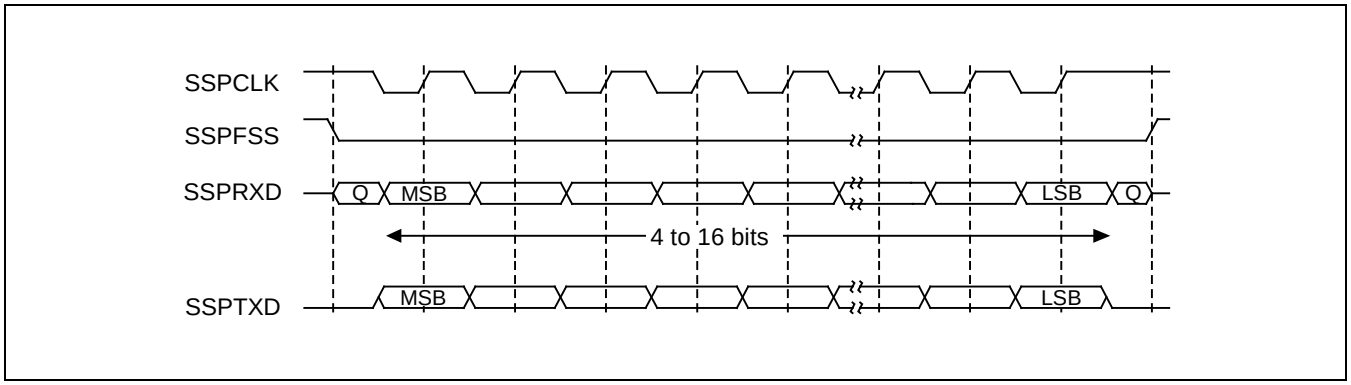


Figure 19-8 Motorola SPI 帧格式， SPO = 1 和 SPH = 1

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。

17.2.2.3.12 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。

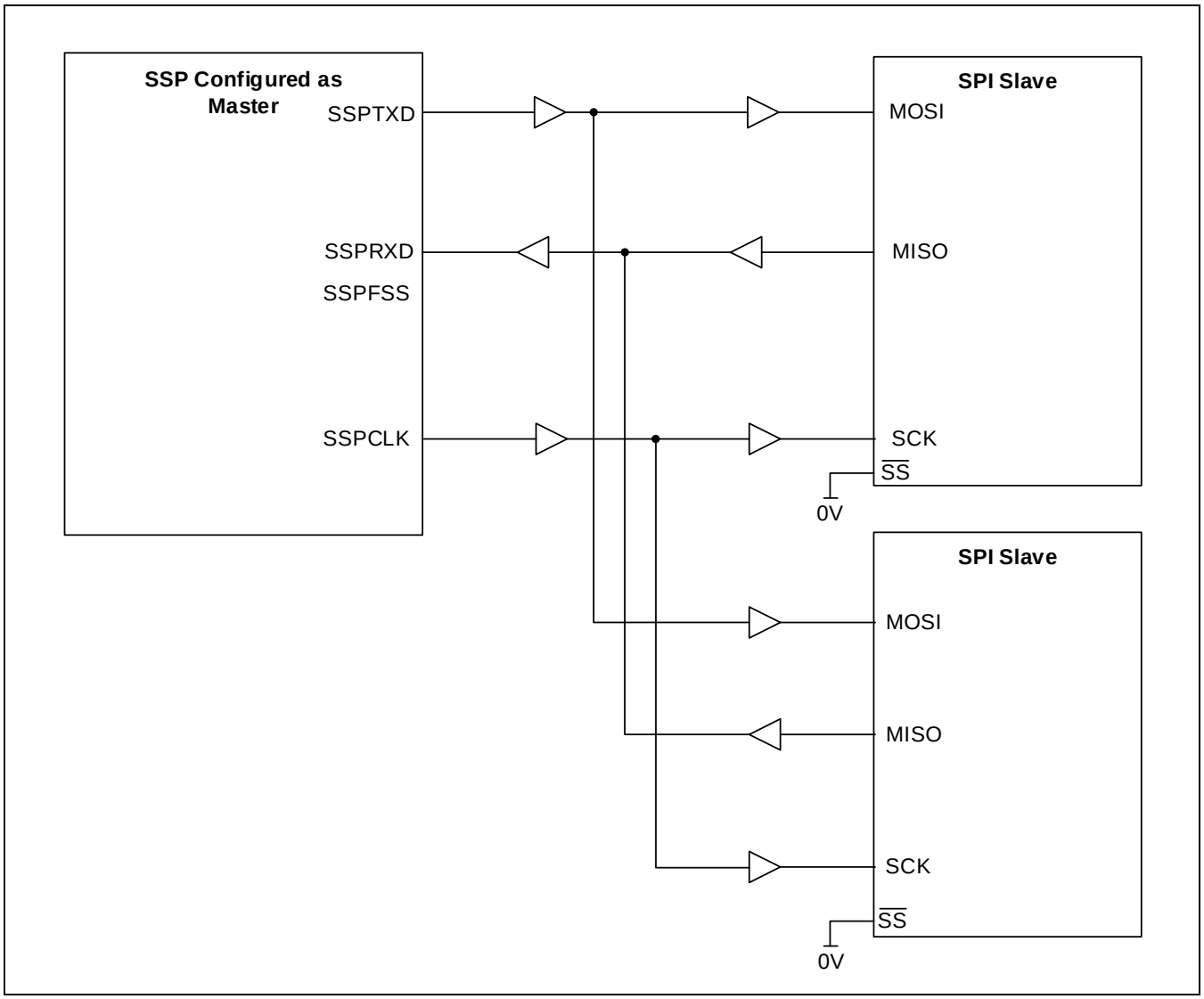


Figure 19-9 SSP主机连接两个SPI从机

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。

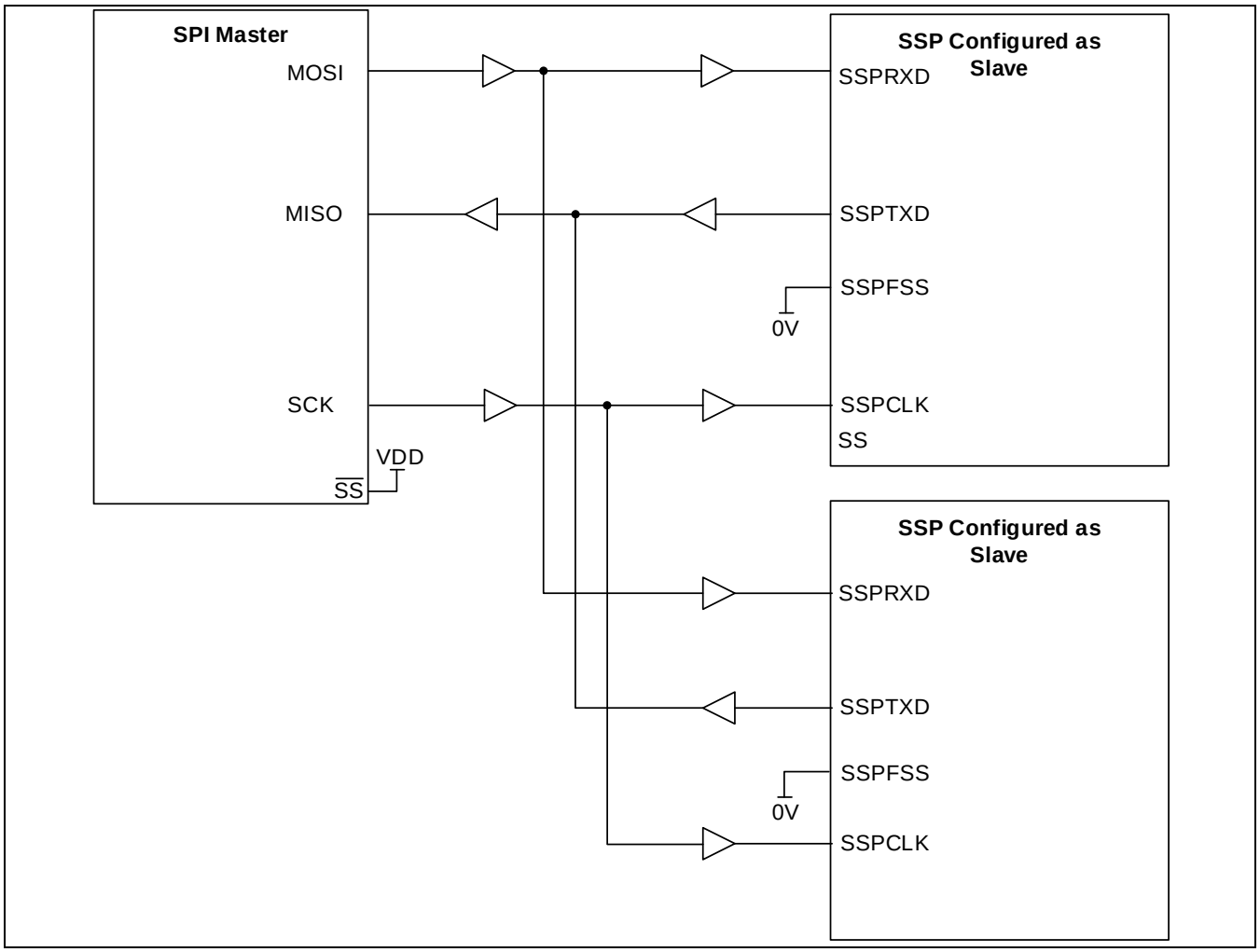


Figure 19-10 SPI 主机连接到两个SSP从机

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。

17.2.2.4 中断

SSP可以产生4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。

17.3 寄存器描述

17.3.1 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

17.3.1.1 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))	0x00

			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。	

17.3.1.2 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																							LPTXOE	LPMD	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	RW	RW	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

LPMD	[7]	RW	0 = SPI普通模式 1 = 主机单线模式	0x0
LPTXOE	[8]	RW	主机单线模式下，数据发送使能控制 0 = 禁止数据发送 1 = 使能数据发送	0x0

17.3.1.3 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。

17.3.1.4 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

17.3.1.5 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。

17.3.1.6 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。

17.3.1.7 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	RORRIS
0	0	0	0	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。

17.3.1.8 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。

17.3.1.9 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。

18 I2C总线

18.1 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接口可以工作在标准模式，快速模式和超快速模式(或称快速模式plus)。标准模式支持的波特率范围为0到100Kbit/s，快速模式支持的波特率范围为0到400Kbit/s，超快速模式支持的波特率范围为0到1000Kbit/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)。

注：如果系列内芯片不具有本外围，那它就不具备本章描述的相关资源。具体参考芯片的数据手册。

18.1.1 主要特性

- 主机或者从机工作模式，支持多主机总线
- 串行，8位的双向数据传输
- 两种速度：
  - 标准模式 (0 到 100Kbits/s)
  - 快速模式 (<=400Kbit/s) 或者超快速模式 (<=1000Kbit/s)
- 7位或者10位寻址方式
- 7位或者10位地址组合传输
- 从机发送模式下支持大量传输模式
- 支持发送和接收缓冲 (FIFO)
- 可编程的SDA保持时间
- 支持总线清除功能

18.1.2 管脚描述

Table 20-1 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	–

18.2 功能描述

18.2.1 模块框图

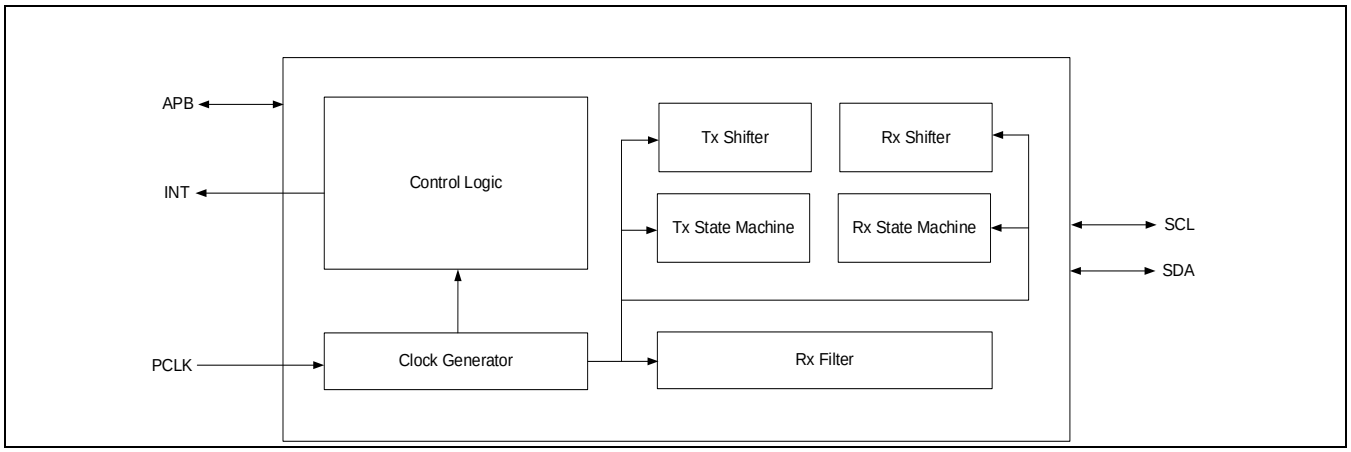


Figure 20-1 I2C模块框图

18.2.2 功能介绍

18.2.2.1 I2C总线概念

18.2.2.1.1 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。

18.2.2.2 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。

18.2.2.2.1 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。

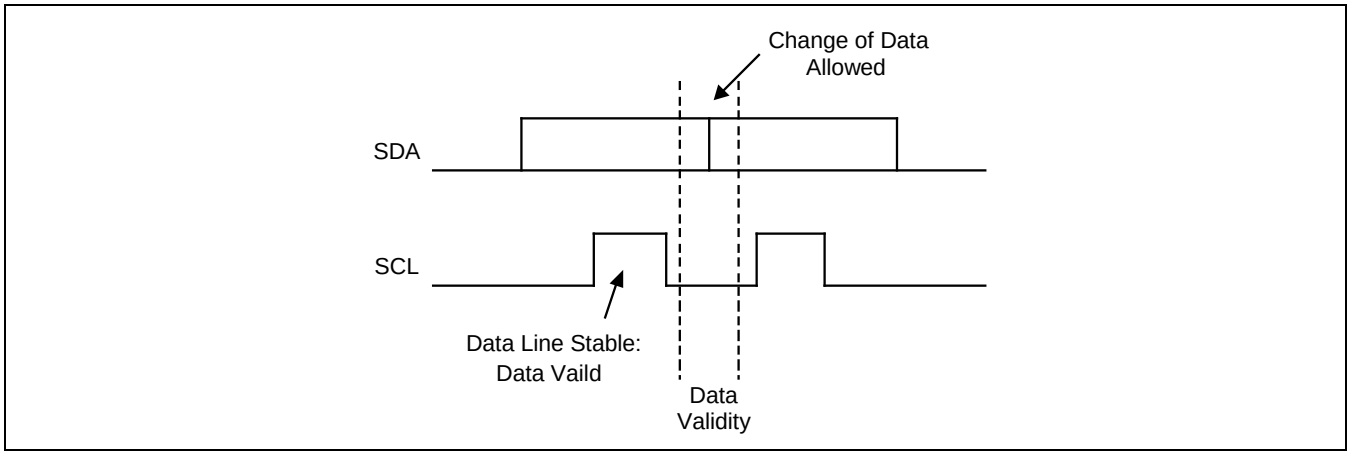


Figure 20-2 数据有效性

18.2.2.2.2 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。

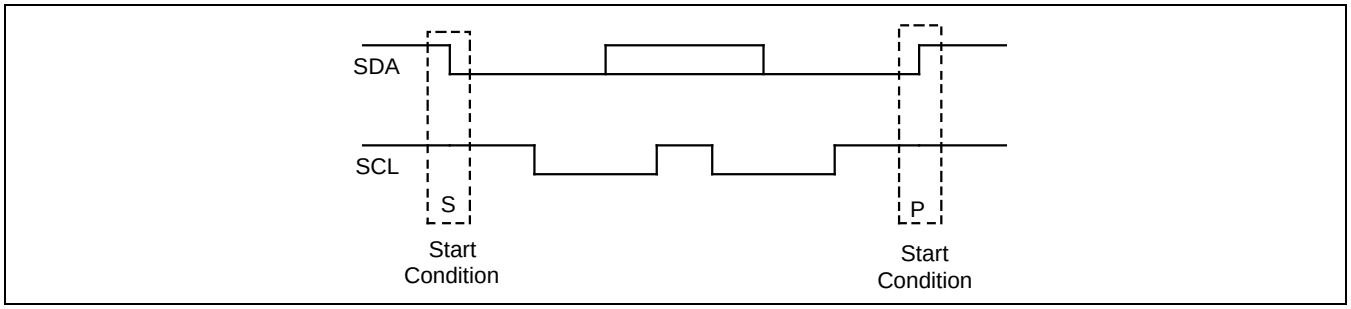


Figure 20-3 起始位和停止位

18.2.2.3 数据传输

18.2.2.3.1 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。

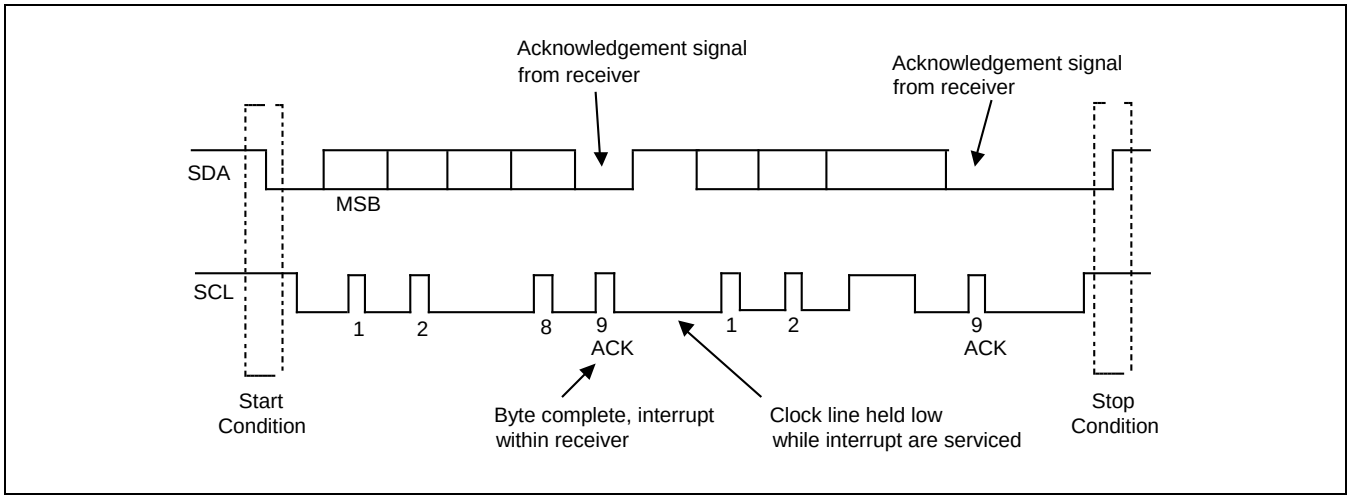


Figure 20-4 I2C总线的数据传输

18.2.2.3.2 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。

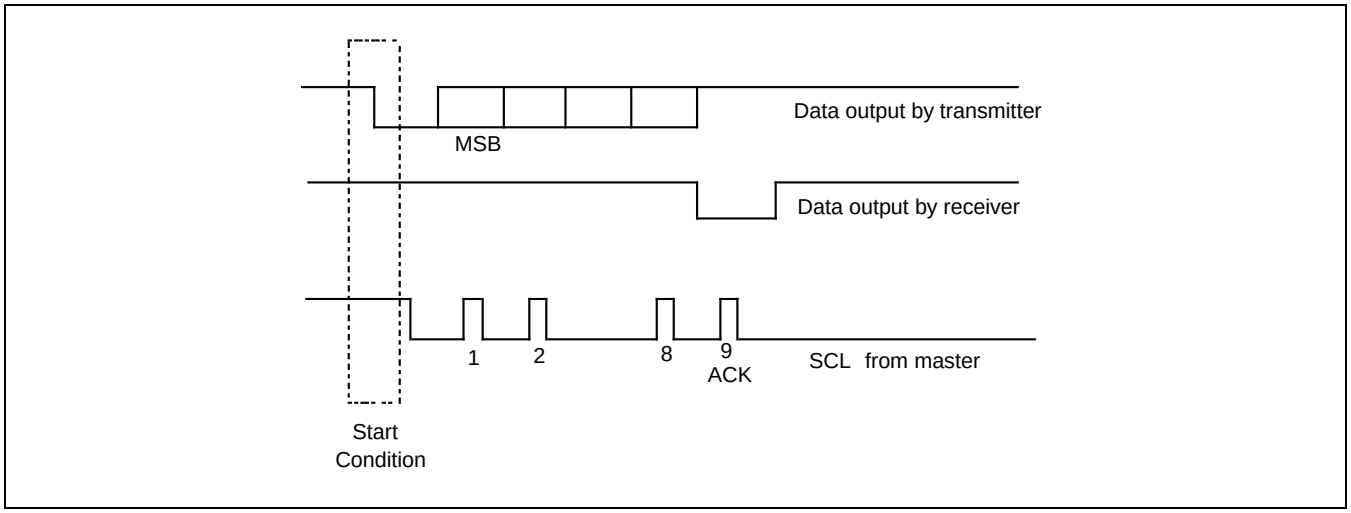


Figure 20-5 应答

18.2.2.4 时钟仲裁

18.2.2.4.1 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。

18.2.2.4.2 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

18.2.2.4.3 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。

18.2.2.4.4 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。

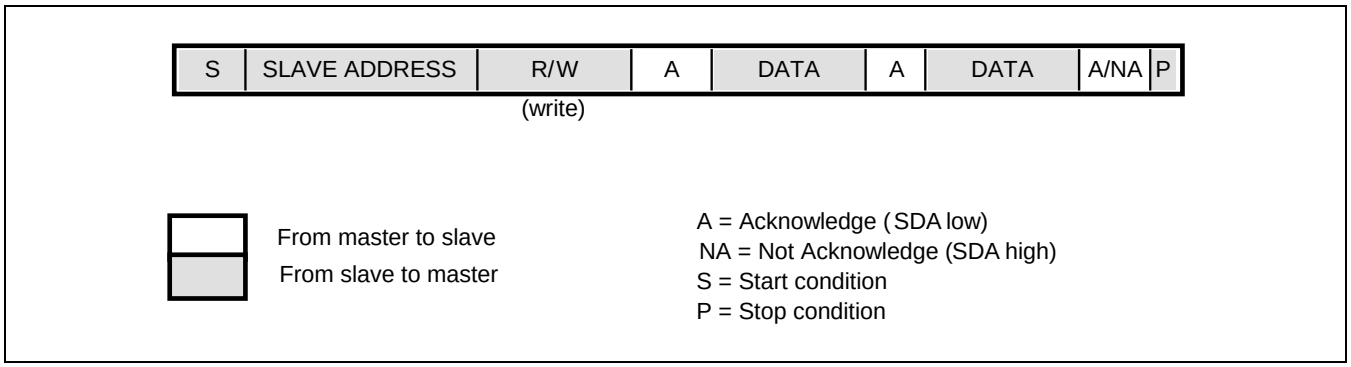


Figure 20-6 主机-发送端寻址从机

18.2.2.5 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。

18.2.2.5.1 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，如下表格. 11110XX 的组合保留给10位寻址使用。

Table 20-2 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。

18.2.2.5.2 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。

18.2.3 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和高达1000kbit/s的超快速模式(或称快速模式plus)
- 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。

18.2.4 快速模式和超快速模式

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 或 1000kbit/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。

18.2.4.1 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。

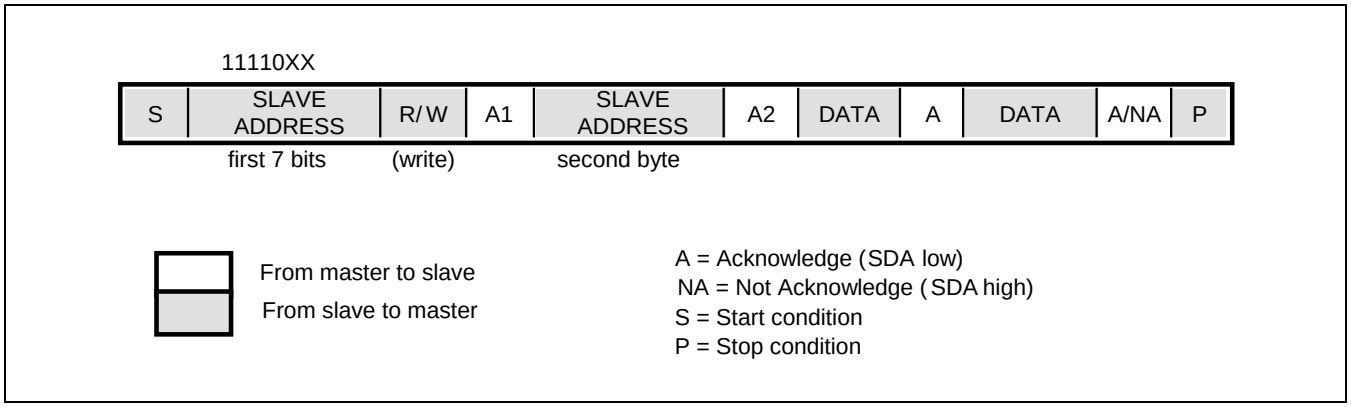


Figure 20-7 主机-发送端用10位地址寻址从机-接收端

18.2.4.2 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。

18.2.5 发送FIFO管理

该I2C模块在发送FIFO为空的时候，不会自动产生停止位，而是将SCL拉低并保持，直到发送FIFO有新数据为止。停止位只有在用户对I2C_DATA_CMD寄存器的第9位写1发送停止指令的时候才会产生。

RESTART	STOP	CMD	DATA
10	9	8	7 ... 0

DATA - 可读写；发送和接收的数据位。

CMD - 只可写；该位决定传输的操作为读操作(CMD=1)还是写操作(CMD=0)。

STOP - 只可写；该位决定在数据被发送或者接收后，是否产生停止位。

RESTART - 只可写；该位决定在数据被发送或者接收后，是否产生重复起始位(如果重复起始位功能没有使能，那么该位决定是否在起始位后产生停止位)

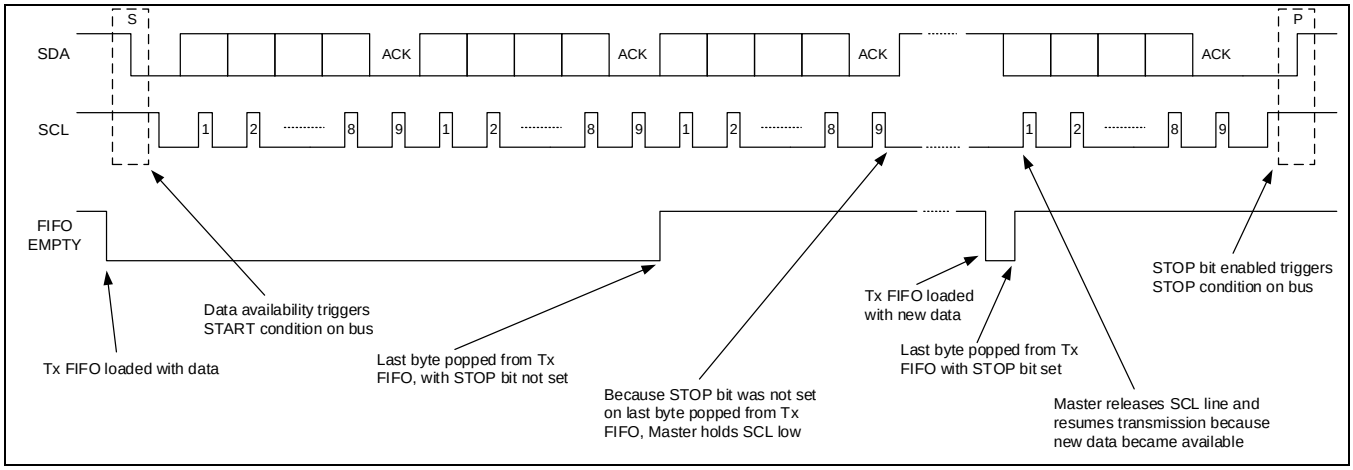


Figure 20-8 主机 – 发送 FIFO 为空 / 停止位的产生

上图说明了该I2C模块工作为主机发送端或者接收端，并且发送FIFO为空时的波形，以及停止位的产生。

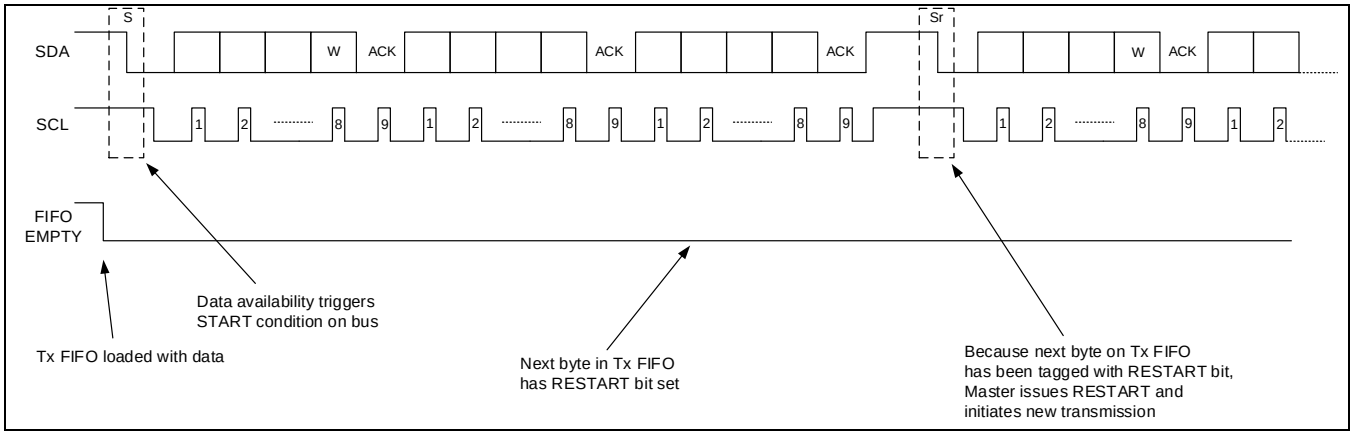


Figure 20-9 主机发送端 – I2C_DATA_CMD的RESTART位置1

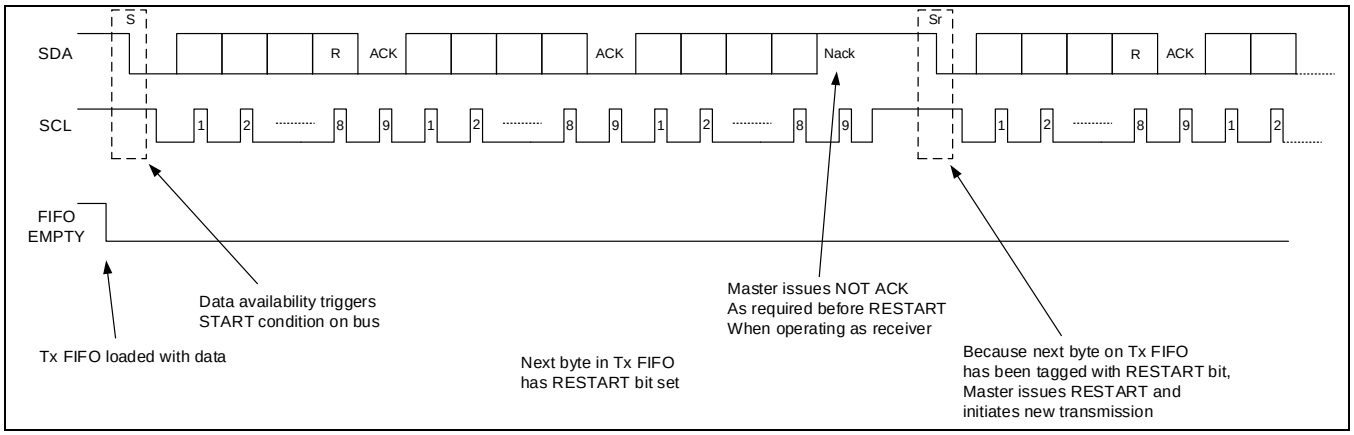


Figure 20-10 主机接收端 – I2C_DATA_CMD的RESTART位置1

上面两幅图演示了用户如何控制 I2C 总线上的重复起始位。如果 I2C_DATA_CMD 寄存器的第 10 位(RESTART)为 1 并且重复起始位功能使能(I2C_CR.RESTART_EN=1)，那么在发送数据或者读取数据前，该模块会产生一个重复起始位。如果重复起始位的功能没有被使能，那么替代重复起始位的是跟在起始位后的一个停止位。

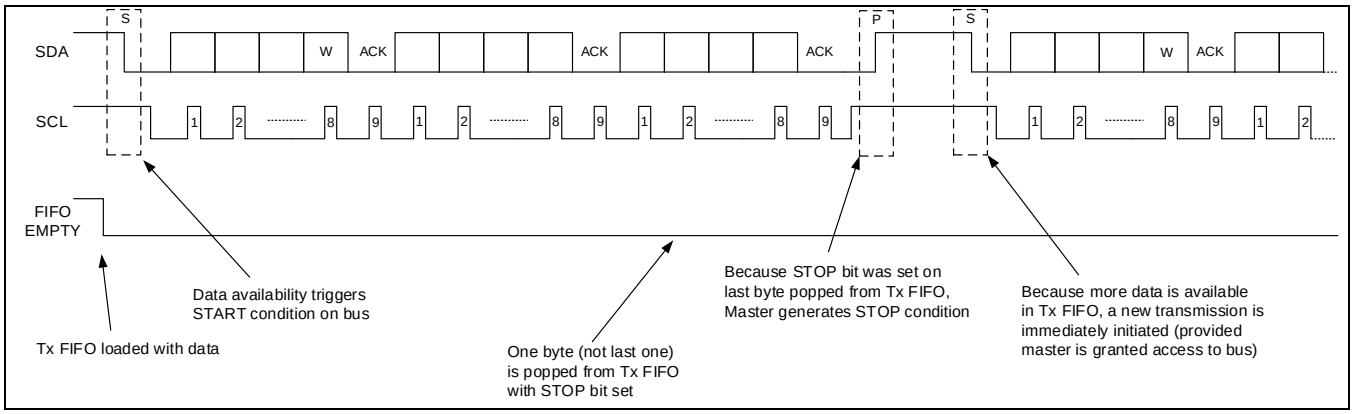


Figure 20-11 主机发送端 – I2C_DATA_CMD的停止位为1 / 发送 FIFO 非空

上图为I2C_DATA_CMD寄存器中STOP位置1并且发送FIFO非空的情况下，该I2C作为主机发送端的工作波形。

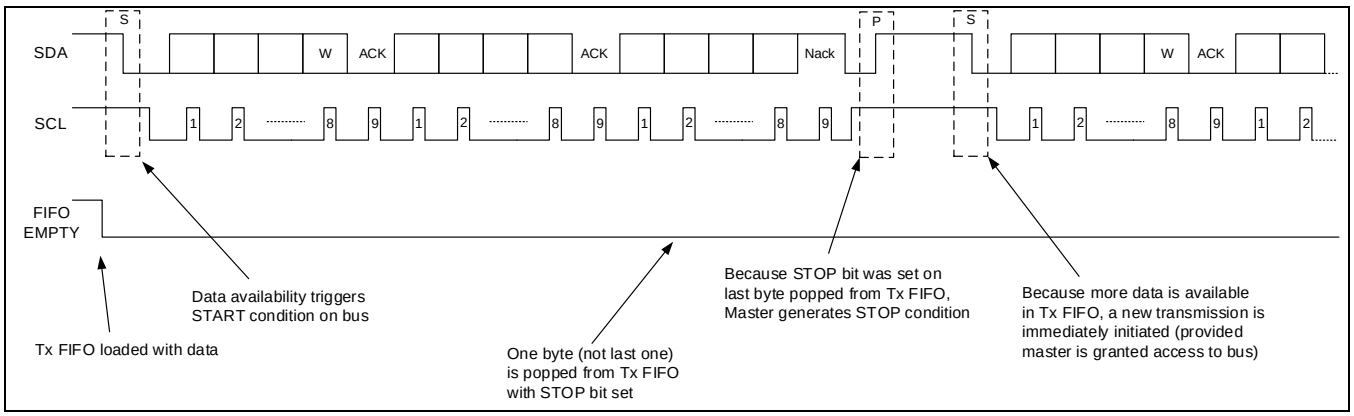


Figure 20-12 主机接收端 – I2C_DATA_CMD的停止位为1 / 发送 FIFO 非空

上图为I2C_DATA_CMD寄存器中STOP位置1并且发送FIFO非空的情况下，该I2C作为主机接收端的工作波形。

18.2.6 工作模式

18.2.6.1 主机模式

18.2.6.1.1 初始化配置

让I2C工作于主机模式，需要进行如下操作：

1. 将I2C_ENABLE寄存器的第0位写0，禁止I2C模块。
2. 配置I2C_CR寄存器，设置需要的工作速度(I2C_CR[2:1])以及寻址模式(7位或者10位)，并且确认I2C_CR的第6位(SLAVEn)为1(默认值)，第0位(MASTER)为1(默认值)。(该I2C模块的默认工作模式为主机模式。)
3. 将目标I2C器件的地址写入I2C_TADDR寄存器。这个寄存器也可以用来产生general call命令或者起始字节命令。
4. 将I2C_ENABLE寄存器的第0位写1，使能I2C模块。
5. 将传输方向和数据写入I2C_DATA_CMD寄存器。如果在I2C使能前写I2C_DATA_CMD寄存器，那么写入的数据和命令将会丢失，因为I2C在禁止状态下，所有缓存都是处于清除的状态。这个步骤会产生起始位并且发送数据字节。一旦I2C使能并且发送FIFO中有数据，I2C会开始发送数据。

注意：从机和主机不需要使用相同的寻址模式，例如作为从机时使用7位地址，而作为主机时使用10位地址。

18.2.6.1.2 主机发送和接收

该I2C模块支持在读和写之间动态切换。需要发送数据时，将数据写入I2C_DATA_CMD寄存器的低字节，并且将该寄存器的第8位写0。同时，如果需要读数据，那么只需要将I2C_DATA_CMD寄存器的第8位写1，低字节位可以为任意值。只要发送FIFO中有数据或者命令，I2C主机就会一直不停的将数据或者命令发送出去。如果数据发送完成，发送FIFO变空，那么主机根据I2C_DATA_CMD[9]这位的状态发送相应的动作：

- 如果为1，那么完成当前数据传输后，发送一个停止位。
- 如果为0，那么将SCL拉低，直到下一个数据或者命令被写入发送FIFO。

更多细节，请参考“发送FIFO管理”章节。

18.2.6.2 从机模式

18.2.6.2.1 初始化配置

让I2C工作于从机模式，需要进行如下操作：

1. 将I2C_ENABLE寄存器的第0位写0，禁止I2C模块。
2. 将从机地址写入I2C_SADDR寄存器。
3. 配置I2C_CR寄存器选择寻址模式 (7位或者10位)。将第6位(SLAVEn)写0，第0位(MASTER)写0，使能从机模式。
4. 将I2C_ENABLE寄存器的第0位写1，使能I2C模块。

注意：从机和主机不需要使用相同的寻址模式，例如作为从机时使用7位地址，而作为主机时使用10位地址。

[特别注意]

如果需要复位，那么建议在I2C总线处于空闲状态时，再对I2C从机进行复位操作。如果在总线有数据传输的时候进行复位，那么当复位结束时，内部用来同步SDA和SCL的锁存器会从复位值1变成总线上实际的值，这样就有可能导致在SCL为1的时候，SDA从1变成0，从而导致I2C从机会监测到一个错误的起始位。用户也可以先配置I2C_CR寄存器为主机模式(SLAVEn=1和MASTER=1)来避免这个问题，然后在复位结束后的大概6个PCLK同步时钟周期后再将这两位配置为0来使能从机模式。

18.2.6.2.2 从机发送端发送单个字节

当另一个总线上的I2C主机寻址到了该I2C从机并且要求数据传输，那么该I2C工作为从机发送端，需要按照以下步骤操作：

1. 另一个I2C主机以一个主机地址启动I2C传输，该地址匹配I2C_SADDR中设置的从机地址。
2. I2C从机应答该从机地址，并且识别传输的方向，知道自己是作为一个从机发送端。
3. I2C从机产生RD_REQ中断(I2C_RISR寄存器的第5位)并且将SCL拉低。I2C进入一个等待状态，直到软件响应。如果RD_REQ中断没有使能，那么I2C_MISR[5]寄存器(RD_REQ位)不会置1，只有I2C_RISR的第5位会置1，所以需要硬件或者软件的查询程序让CPU周期性的读取I2C_RISR寄存器的值。
    a. 读取I2C_RISR[5] (RD_REQ位)，判断是否为1，也就是判断RD_REQ事件是否发生。
    b. 软件必须响应并且满足I2C传输的需求。
    c. 软件查询的间隔必须是最快SCL时钟周期的10倍。例如，对于400kb/s的速度，间隔时间是25us。(推荐10这个值是因为这个值大概是I2C总线上传输单个字节所需要的时间。)
4. 如果在收到读请求(RD_REQ)之前，发送FIFO里仍然还有数据，那么I2C模块会产生一个TX_ABRT中断

(I2C_RISR的第6位)，并且把发送FIFO中已有的数据清除。

注意：因为只要TX_ABRT事件发生，I2C发送FIFO就会被强制进入清除/复位状态，所以在将数据写入发送FIFO之前，软件需要将TX_ABRT状态清除(I2C_ICR寄存器的相应位写1)。更多信息请参考I2C_RISR寄存器。
如果TX_ABRT中断没有使能，那么I2C_MISR[6]寄存器(TX_ABRT位)不会置1，只有I2C_RISR的第6位会置1，所以需要硬件或者软件的查询程序让CPU周期性的读取I2C_RISR寄存器的值(像上一步一样)。

a. 读取I2C_RISR[6] (TX_ABRT位)，判断是否为1，也就是判断TX_ABRT事件是否发生。

b. 软件查询的间隔跟上面查询I2C_RISR[5]寄存器一样。

5. 软件将需要发送的数据写入I2C_DATA_CMD寄存器(第8位写0)。
6. 软件必须清除I2C_RISR寄存器中RD_REQ和TX_ABRT的中断状态(第5和第6位)。
7. I2C释放SCL并且发送数据。
8. 主机可能使用一个重复起始位占据I2C总线或者使用一个停止位释放总线。

18.2.6.2.3 从机接收端接收单个字节

当另一个总线上的I2C主机寻址到了该I2C从机并且向它发送数据，那么该I2C工作为从机接收端，需要按照以下步骤操作：

1. 另一个I2C主机以一个主机地址启动I2C传输，该地址匹配I2C_SADDR中设置的从机地址。
2. I2C从机应答该从机地址，并且识别传输的方向，知道自己是作为一个从机接收端。
3. I2C从机接收到数据并且将数据存入接收缓冲(接收FIFO)中。
4. I2C产生RX_FULL中断(I2C_RISR[2]寄存器)。
   如果RX_FULL中断没有使能，建议使用一个软件查询程序周期性的查询I2C_STATUS寄存器的值(参考"从机发送端发送单个字节")。如果读到I2C_STATUS寄存器第3位RFNE的值为1，那么软件可以认为系统产生了RX_FULL中断。
5. 软件从I2C_DATA_CMD寄存器中读取接收到的数据。
6. 主机可能使用一个重复起始位占据I2C总线或者使用一个停止位释放总线。

18.2.6.2.4 从机的大量数据传输

在标准I2C协议中，所有传输都是单字节传输，远程主机端要求读取数据时，程序员将一个字节写入从机的发送FIFO中。当从机(发送端)每次从远程主机(接收端)收到一个读请求(RD_REQ)，都需要有至少有一个数据被写入从机发送端的发送FIFO中。本I2C模块设计成能处理发送FIFO中的大量数据，不需要在每次的数据请求时都产生一个中断，避免了每次中断都往发送FIFO中写数据的大量不必要的延时。

该模式只有I2C工作于从机发送端时有效。如果远程主机端应答了从机发送端发送的数据，并且从机发送FIFO中已经没有数据了，那么从机会拉低SCL并且产生一个读请求(RD_REQ)的中断，等待数据写入发送FIFO。

如果RD_REQ中断没有使能，建议使用一个软件查询程序周期性的查询I2C_STATUS寄存器的值(参考"从机发送端发送单个字节")。如果读到I2C_STATUS寄存器第5位RD_REQ的值为1，那么软件可以认为系统产生了RD_REQ中断。

RD_REQ中断是由读请求产生的，像其它中断一样，在退出中断服务子程序(ISR)的时候，该中断状态需要被清除。在ISR中允许将1个或者多个数据写入发送FIFO中。在这些数据发送到主机的过程中，如果主机应答了最后一个字节，那么从机必须再产生一个RD_REQ中断，因为主机在要求读更多的数据。

如果程序员提前知道远程主机端需要的数据包有n字节，那么当远程主机端寻址到该从机并且请求数据时，发送FIFO可以直接填入n字节的数据，让远程主机一次性连续读取出去。例如，只要远程主机一直在应答数据并且从机的发送FIFO中一直有数据时，从机会连续不停的将数据发送出去，而不是去拉低SCL等待数据或者产生RD_REQ中断。

如果远程主机需要从I2C从机读取n字节数据，但是程序员写入发送FIFO的数据超过了n字节，那么当从机完成n字节的发送后，它会清除发送FIFO，忽略多余的字节。这时，I2C会产生一个TX_ABRT事件，以表示发送FIFO被清除。在需要收到应答位时，如果收到一个非应答(NACK)位，那么说明远程主机已经完成了数据读取，这时候从机会产生一个标志位，告诉从机的工作状态机去清除发送FIFO中多余的数据。

18.2.6.3 关闭I2C

18.2.6.3.1 关闭流程

1. 定义一个 10 倍于系统中最高传输速度对应周期的时间间隔 (ti2c_poll)，例如，对于 400kb/s 的速度，这个 ti2c_poll 值为 25us.
2. 定义一个最大的超时参数, MAX_T_POLL_COUNT，如果重复查询的操作时长超过了这个最大值，那么报错。
3. 执行一个阻塞的线程/进程/函数，软件上不允许启动任何 I2C 主机传输，只允许未完成的传输结束。(如果 I2C 只配置为从机，那么此步骤可以忽略。)
4. 初始化 POLL_COUNT 变量为 0.
5. 将 I2C_ENABLE 寄存器的第 0 位置 0.
6. 读取 I2C_STATUS 寄存器并且查询第 0 位 ENABLE 位，POLL_COUNT 加 1，如果 POLL_COUNT >= MAX_T_POLL_COUNT，退出并且返回相关的错误代码。
7. 如果 I2C_STATUS[0] 为 1，那么睡眠等待 ti2c_poll 时长，然后继续上一个步骤，否则退出并且返回相关的成功代码。

18.2.6.4 中止I2C传输

I2C_ENABLE寄存器中的ABORT控制位允许软件在完成发送FIFO中的传输命令之前刷新I2C总线。控制器在收到ABORT请求后，会在I2C总线上发送一个停止位，并且随后清除发送FIFO。只有在主机模式下，允许中止传输。

18.2.6.4.1 中止流程

1. 停止对发送 FIFO (I2C_DATA_CMD)的写入。
2. 将 I2C_ENABLE 寄存器的第 1 位 (ABORT) 置 1.
3. 等待 TX_ABRT 中断。
4. 读取 I2C_TX_ABRT 寄存器确认中止源为 USER_ABRT.

18.2.6.5 干扰过滤

干扰过滤逻辑基于对输入信号(SCL 和 SDA)监控的一个计数器，检测它们在预设数量的 PCLK 周期内是否保持稳定的状态。每个信号(SCL 和 SDA)都有单独计数器。PCLK 周期的数量可以由用户设定，需要根据 PCLK 的频率和毛刺干扰信号的长度进行计算。

当计数器的输入变化时，计数器会重新开始计数。根据输入信号的行为，会有下面的情况。

– 输入信号保持不变，直到计数器计数到了预设的最大值。在这个情况下，内部信号(过滤后的输出信号)被更新成输入信号，计数器复位并且停止，直到输入信号再次变化的时候，计数器才会重新开始计数。

– 输入信号在计数器计数到预设最大值之前，发生了变化。在这个情况下，计数器复位并且停止计数，内部信号不会更新成输入信号。计数器保持停止，直到输入信号再次变化。

如下图所示：

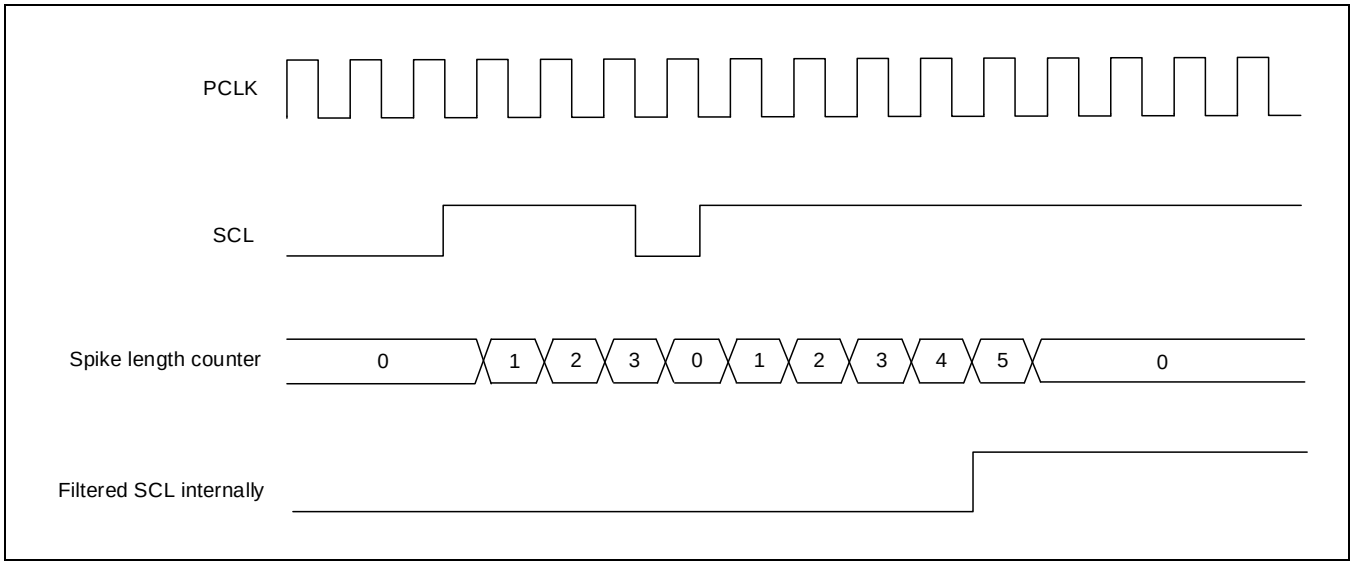


Figure 20-13 毛刺干扰信号的过滤

在图中的例子中，计数器的预设最大值为5，基于PCLK周期50ns，毛刺250ns计算得到。

注意：SCL输入上有一个两级的同步电路，但为了清楚的描述过滤电路，该同步电路的延时没有表示在上图中。

毛刺干扰过滤配置寄存器I2C_SPKLEN只有在I2C关闭的时候可写，该寄存器最小值为1（也是默认值），写0无效。

18.2.6.6 总线清除特性

该I2C模块支持总线清除的特性，当时钟(SCL)或者数据(SDA)传输线被意外拉低锁死时，可以进行恢复。

下面章节描述了SDA和SCL被拉低锁死时的恢复机制。

18.2.6.6.1 SDA拉低锁死的恢复

在SDA被锁死并且一直拉低的情况下，主机会使用下面的方法进行恢复。

1. 主机发送最大9个时钟脉冲，用来从这9个时钟造成的低电平中恢复。
   - 时钟脉冲的个数会有不同，跟从机需要发送的剩余位数有关。由于最大的位数是9，所以主机最多发9个，让从机进行恢复。
   - 主机在SDA上发送一个高电平，然后检测SDA是否恢复。如果SDA没有恢复(读到0)，那么主机会继续发送SCL，直到最大9个SCL时钟脉冲。
2. 如果SDA在9个时钟脉冲内恢复，那么主机发送一个停止位，释放总线的控制。
3. 如果在9个时钟脉冲内，SDA还没有恢复，那么系统需要一个硬件复位。

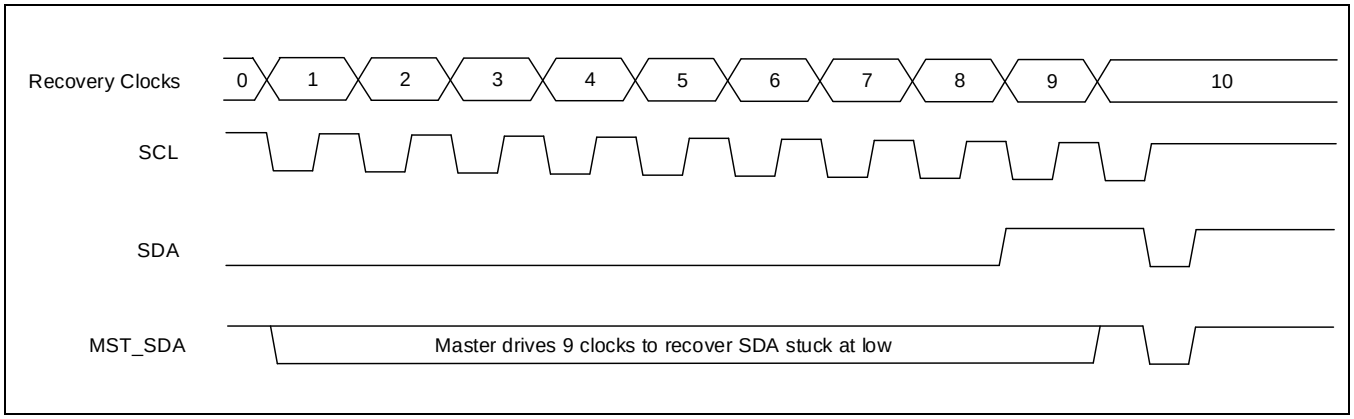


Figure 20-14 SDA在9个SCL时钟内恢复

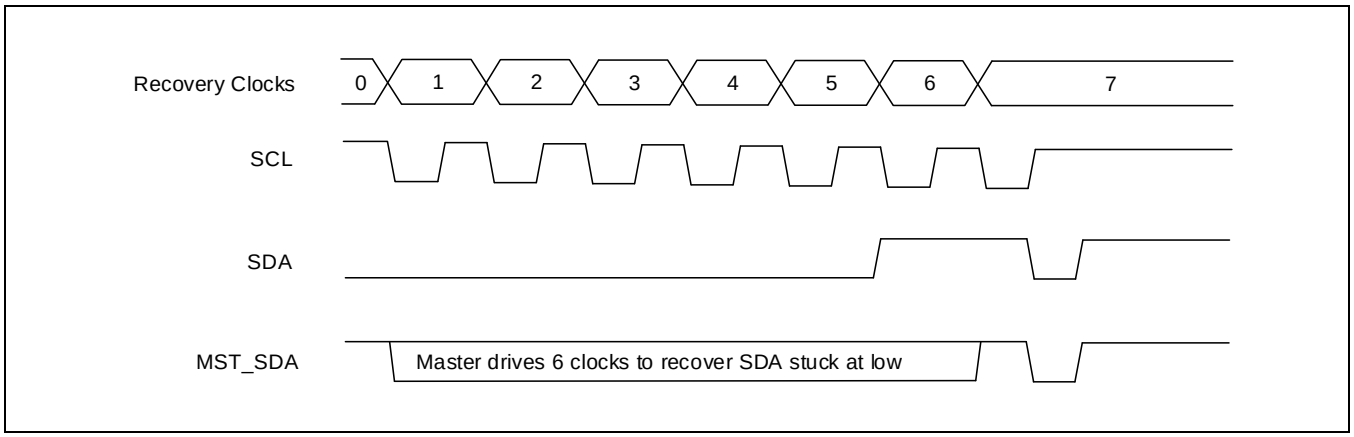


Figure 20-15 SDA在6个SCL时钟内恢复

18.2.6.6.2 SCL拉低锁死的恢复

如果时钟信号SCL被意外锁死拉低，那么除了使用硬件复位信号去复位总线意外，没有其它办法去解决。关于SCL拉低锁死恢复的详细说明，请参考编程示例。

18.2.6.7 SDA Hold时间

I2C协议标准要求标准模式和快速模式下，在SDA信号(tHD:DAT)上有300ns的hold时间，并且在快速模式或者超快速模式下hold时间足够长以覆盖SCL从1变到0下降沿的不稳定区间。

SCL和SDA信号在板上的延时会导致I2C主机上虽然满足hold时间的要求，但在I2C从机上不满足 (反之亦然)。由于每个应用的环境都不同，延时也都不同，所以该I2C模块含有一个软件可编程的寄存器(I2C_SDA_THOLD)，用来动态调节SDA的hold时间。

低16位[15:0]用来控制SDA在从机和主机模式下发送时的hold时间 (在SCL从高变低后)。

高8位[23:16]用来在主机和从机的接收状态下，扩展SDA的变化时间节点(如果有的话)。

如果不同速度模式下需要不同的SDA hold时间，那么I2C_SDA_THOLD寄存器必须在速度模式改变后重新写入。I2C_SDA_THOLD寄存器只有在I2C被禁止(I2C_ENABLE[0] = 0)的时候才可以写入。

18.2.6.7.1 SDA在接收端的hold时序

当I2C工作为接收端，根据I2C协议，设备必须在内部保持SDA的状态足够长时间，以覆盖SCL从1变道0时的不稳定区间。

SDA_RX_THOLD位用来改变内部对输入SDA信号的hold时间，寄存器的值表示以PCLK周期为单位的保持时长。SDA_RX_THOLD的最小值为0.这个hold时间只有在SCL是高电平的时候有效，接收端在SCL内部变低后就不会去扩展SDA信号了。

下图为I2C作为接收端，SDA_RX_THOLD大于等于3时的情况。

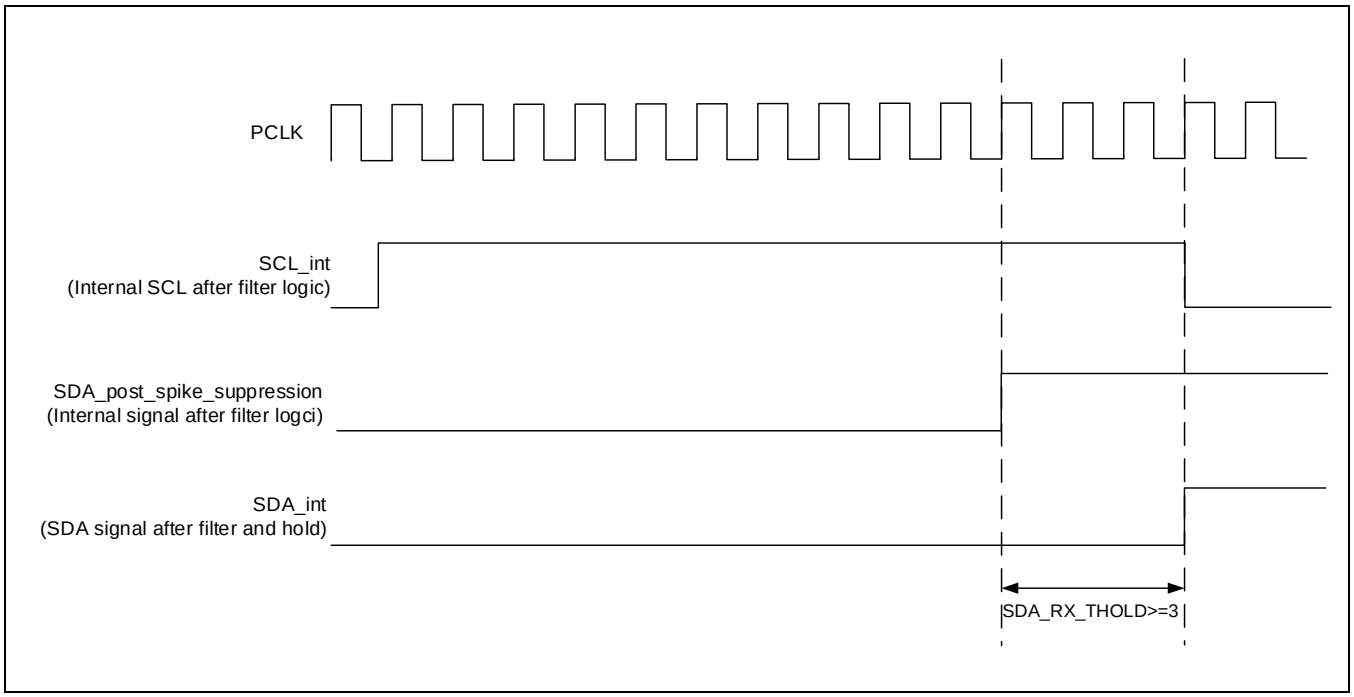


Figure 20-16 SDA Hold时序 SDA_RX_THOLD>=3

如果SDA_RX_THOLD大于3，I2C在3个PCLK周期后就不会保持SDA了，因为内部SCL已经变低。

下图为I2C作为接收端，SDA_RX_THOLD等于2时的情况。

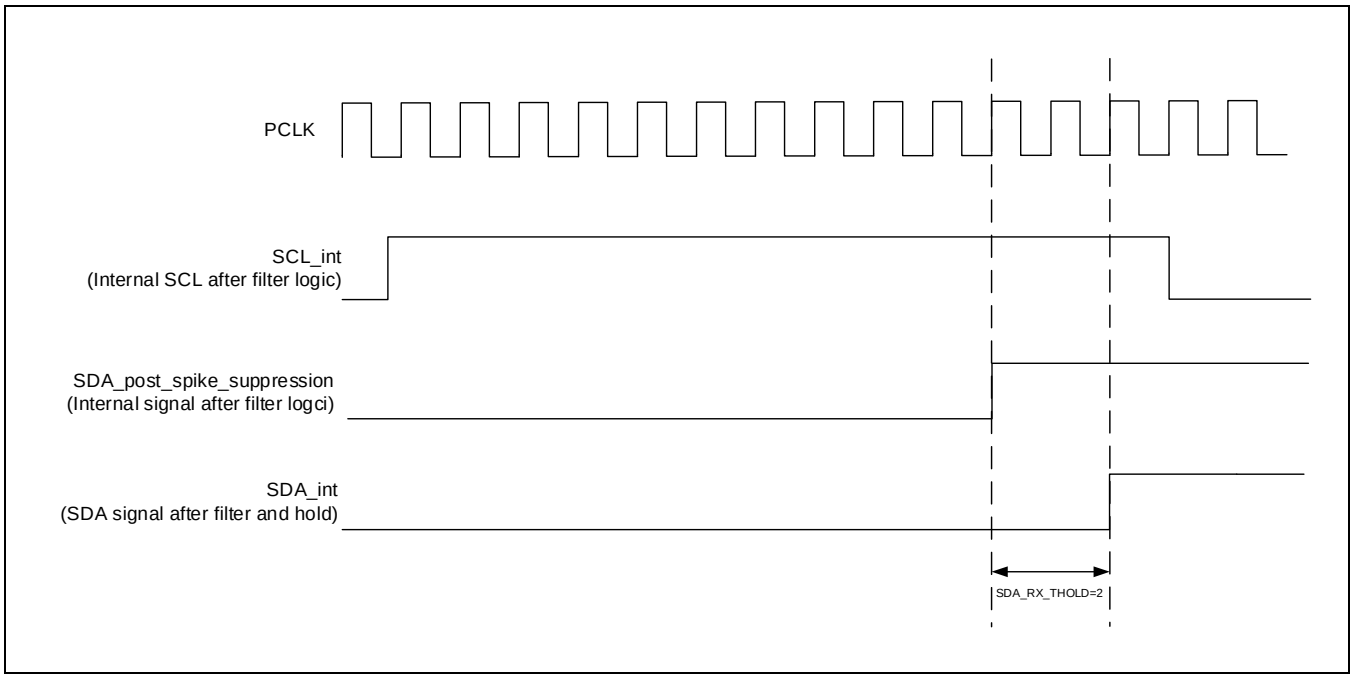


Figure 20-17 SDA Hold时序 SDA_RX_THOLD=2

SDA_RX_THOLD的最大值根据速度的不同，可以根据下面公式计算：

标准模式: I2C_SS_SCLH – I2C_SPKLEN – 3

快速模式或者超快速模式: I2C_FS_SCLH – I2C_SPKLEN – 3

注意： 最大值只在主机模式下有效。在从机模式下，必须保证 SDA_RX_THOLD 不超过SCL的下降沿。

18.2.6.7.2 SDA在发送端的Hold时序

SDA_TX_THOLD位可以用来改变I2C产生的SDA信号的时序，SDA_TX_THOLD寄存器值的单位为PCLK周期。

当I2C工作在主机模式，最小的tHD:DAT时间是1个PCLK周期。所以即使SDA_TX_THOLD的值为0，I2C也会在SCL变0后的下一个周期再驱动SDA。对于其它所有SDA_TX_THOLD的值：

– SDA的驱动输出发生在SCL变0后，再过SDA_TX_THOLD个PCLK周期

当I2C工作在从机模式，最小的tHD:DAT时间是I2C_SPKLEN + 7 PCLK周期。在这段时间内，在SCL信号上需要进行同步和毛刺噪声的过滤，所以即使SDA_TX_THOLD的值小于I2C_SPKLEN + 7，I2C也会在SCL变0的I2C_SPKLEN + 7 PCLK周期后再驱动输出SDA。对于其它所有SDA_TX_THOLD值：

– SDA的驱动输出发生在SCL变0后，再过SDA_TX_THOLD个PCLK周期

下图为I2C工作在主机模式，SDA_TX_THOLD等于3时的tHD:DAT时序。

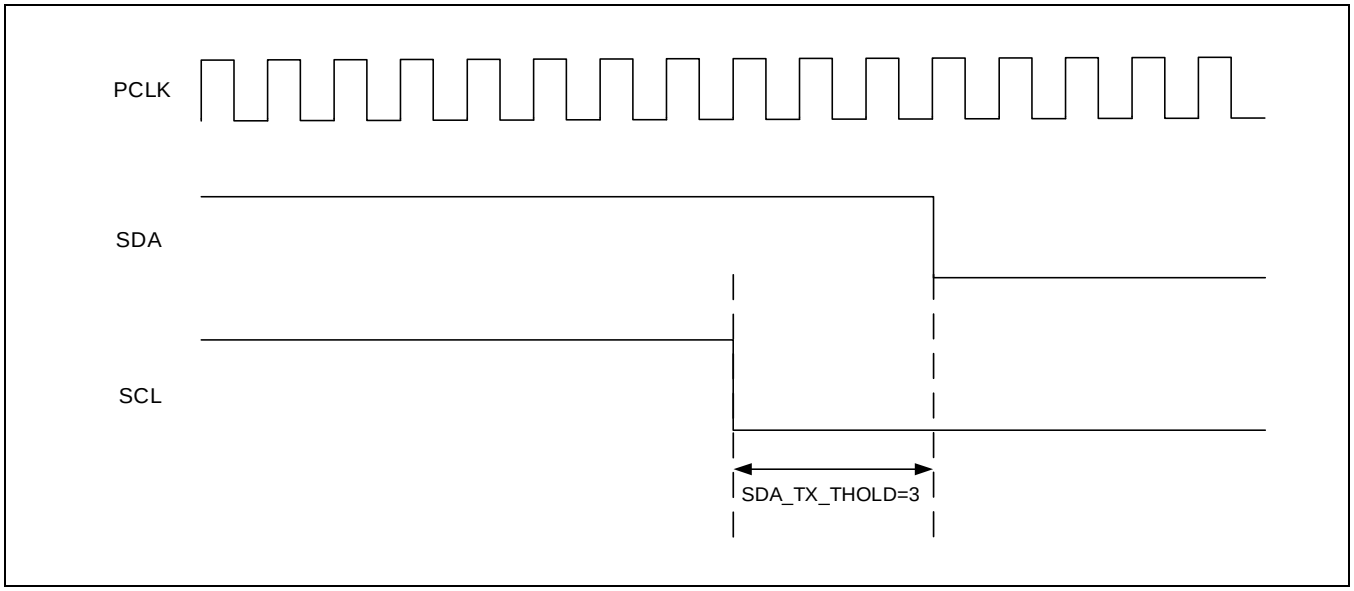


Figure 20-18 I2C主机模式SDA_TX_THOLD = 3下实现tHD;DAT

注意： 配置的SDA hold时间不能超过SCL低电平的时间长度，所以配置的值不能大于N_SCL_LOW-2，N_SCL_LOW是SCL低电平宽度，以PCLK周期为单位。

18.3 编程示例

下面的流程图为 I2C 主机模式的编程示例。

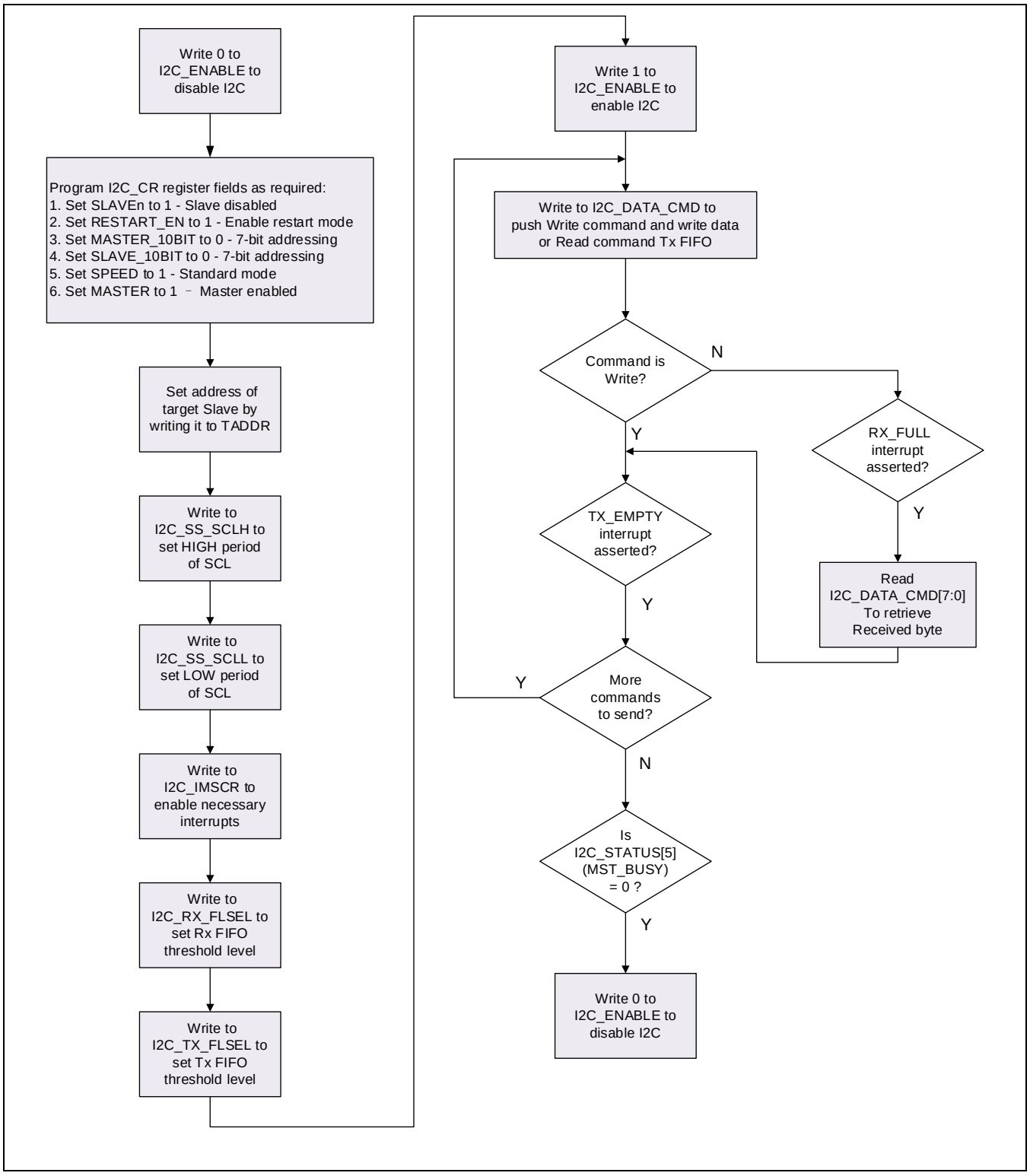


Figure 20-19 I2C主机的流程图

下面的流程图为I2C从机模式的编程示例。

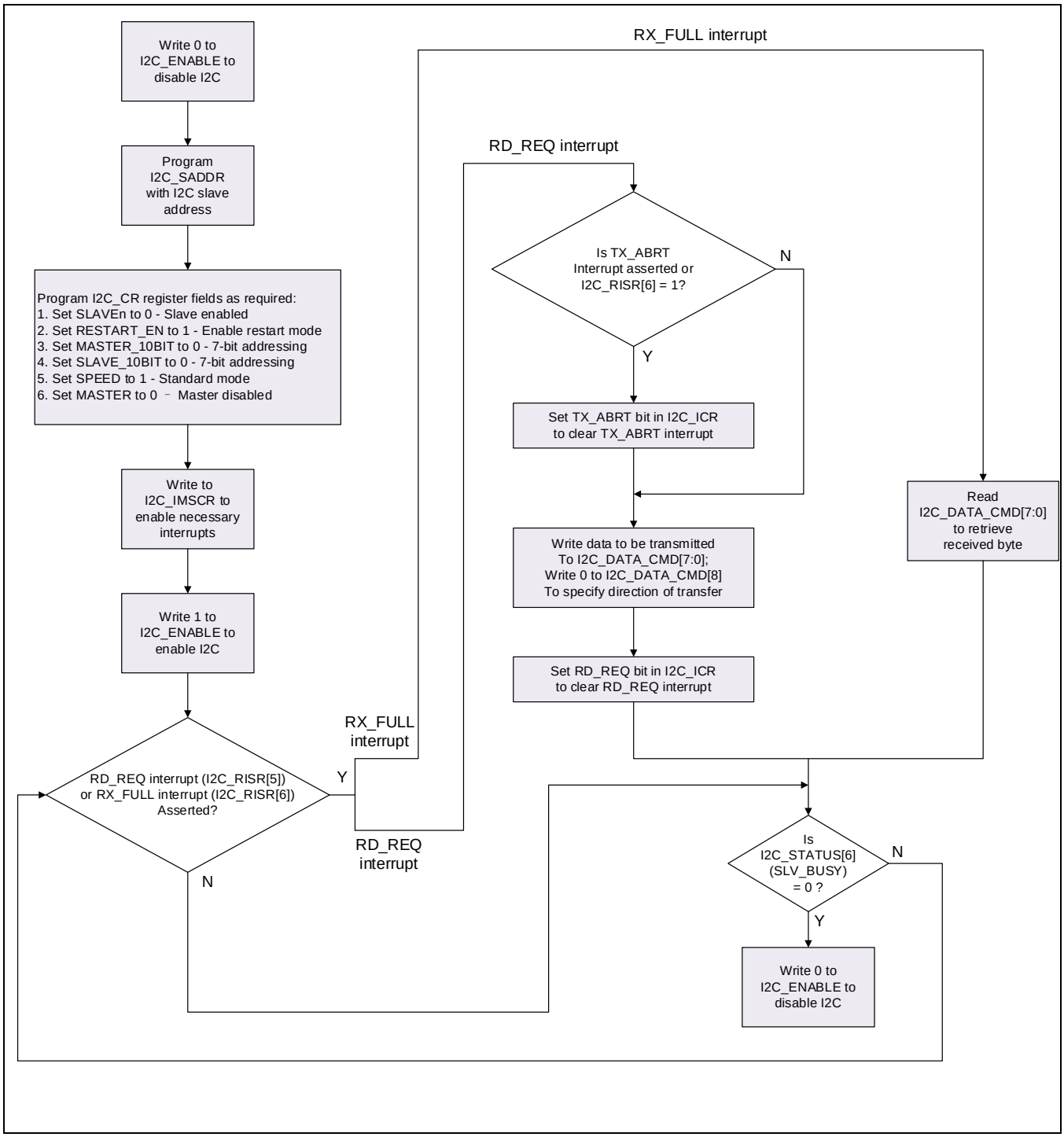


Figure 20-20 I2C从机的流程图

下面的流程图为SCL和SDA总线恢复的编程示例。

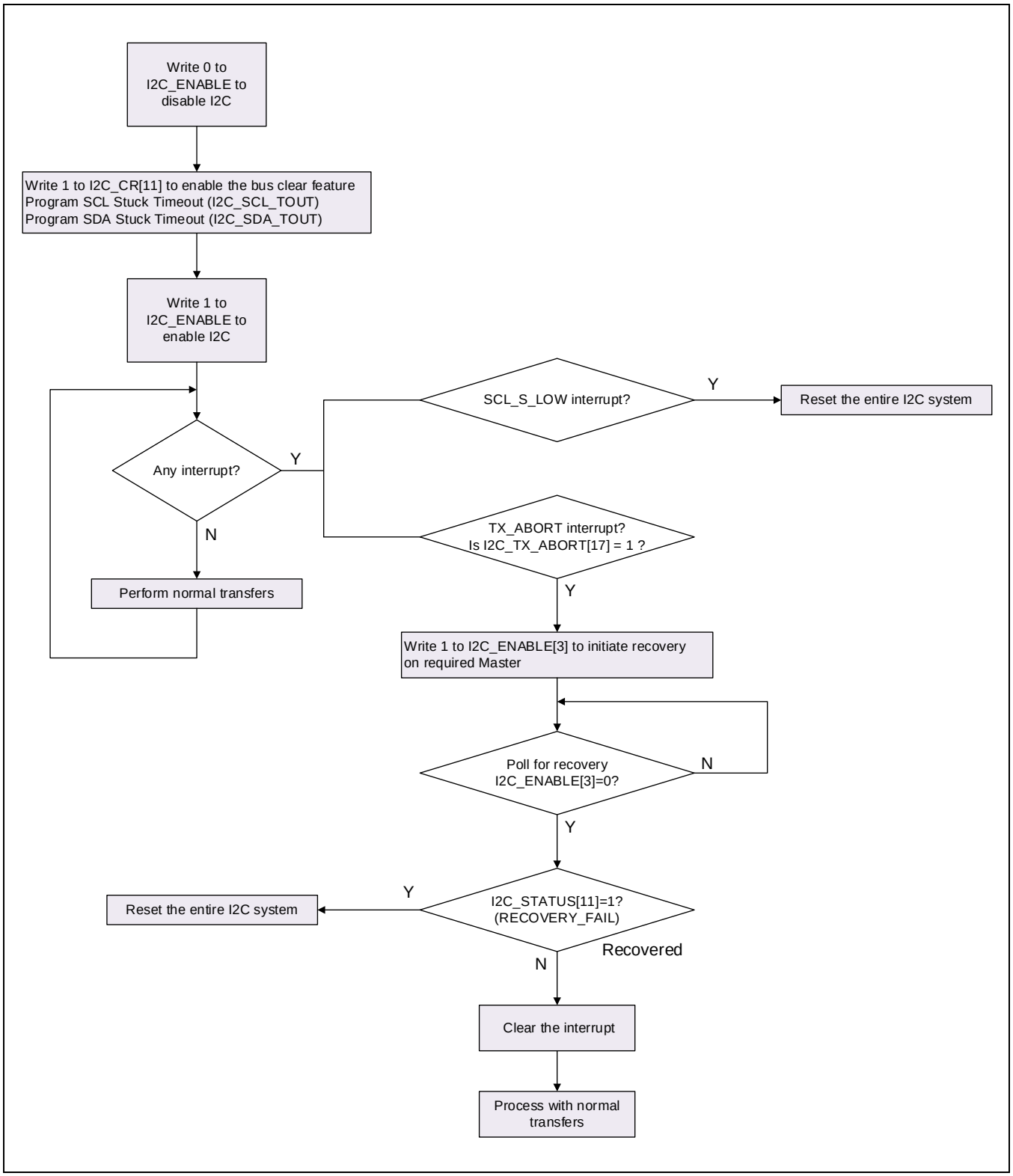


Figure 20-21 SCL和SDA总线恢复的流程图

18.4 寄存器说明

18.4.1 寄存器表 (Base Address: 0x400A_0000)

Offset Address	Name	Description	R/W	Reset State
0x000	I2C_CR	控制寄存器	R/W	0x0000007D
0x004	I2C_TADDR	目标地址寄存器	R/W	0x00000055
0x008	I2C_SADDR	从机地址寄存器	R/W	0x00000055
0x00C	Reserved	保留	R/W	0x00000000
0x010	I2C_DATA_CMD	数据和命令寄存器	R/W	0x00000000
0x014	I2C_SS_SCLH	标准模式 SCL 高电平长度计数寄存器	R/W	0x00000050
0x018	I2C_SS_SCLL	标准模式 SCL 低电平长度计数寄存器	R/W	0x0000005E
0x01C	I2C_FS_SCLH	高速模式 SCL 高电平长度计数寄存器	R/W	0x0000000C
0x020	I2C_FS_SCLL	高速模式 SCL 低电平长度计数寄存器	R/W	0x0000001A
0x024	Reserved	保留	R/W	0x00000000
0x028	Reserved	保留	R/W	0x00000000
0x02C	I2C_RX_FLSEL	接收 FIFO 阈值寄存器	R/W	0x00000000
0x030	I2C_TX_FLSEL	发送 FIFO 阈值寄存器	R/W	0x00000000
0x034	I2C_RX_FL	接收 FIFO 状态寄存器	R/W	0x00000000
0x038	I2C_TX_FL	发送 FIFO 状态寄存器	R/W	0x00000000
0x03C	I2C_ENABLE	使能寄存器	R/W	0x00000000
0x040	I2C_STATUS	状态寄存器	R/W	0x00000006
0x044	Reserved	保留	R/W	0x00000000
0x048	I2C_SDA_TSETUP	SDA Setup 时间寄存器	R/W	0x00000064
0x04C	I2C_SDA_THOLD	SDA Hold 时间寄存器	R/W	0x00000001
0x050	I2C_SPKLEN	毛刺干扰滤波控制寄存器	R/W	0x00000001
0x054	Reserved	保留	R/W	0x00000000
0x058	I2C_MISR	中断状态寄存器	R/W	0x00000000
0x05C	I2C_IMCR	中断使能寄存器	R/W	0x00000000
0x060	I2C_RISR	原始中断状态寄存器	R/W	0x00000000
0x064	I2C_ICR	中断清除寄存器	R/W	0x00000000
0x068	Reserved	保留	R/W	0x00000000
0x06C	I2C_SCL_TOUT	SCL 锁死超时控制寄存器	R/W	0xFFFFFFFF
0x070	I2C_SDA_TOUT	SDA 锁死超时控制寄存器	R/W	0xFFFFFFFF
0x074	I2C_TX_ABRT	发送中止状态寄存器	R/W	0x00000000
0x078	I2C_GCALL	General Call 控制寄存器	R/W	0x00000000
0x07C	I2C_NACK	从机 NACK 控制寄存器	R/W	0x00000000

18.4.2 I2C_CR (I2C控制寄存器)

- 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																				BUS_CLR_EN	RSVD	RXFULL_HLD	TX_EMPTY_CON	STOP_DET_CON	SLAVEn	RESTART_EN	MASTER_10BIT	SLAVE_10BIT	SPEED		MASTER
0	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	1
R	R	R	R	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

Name	Bit	Type	Description	Reset Value
MASTER	[0]	R/W	主机模式控制位 0: 禁用主机模式 1: 使能主机模式 注意如果此位写了1，那么第6位SLAVEn也必须写1.	0x1
SPEED	[2:1]	R/W	I2C工作速度 0x0: 无效 0x1: 标准模式 (0 到 100 Kb/s) 0x2: 高速模式 (<=400 Kb/s) 或超高速模式 (<=1000 Kb/s) 0x3: 无效	0x2
SLAVE_10BIT	[3]	R/W	从机模式寻址控制 0: 7位寻址 1: 10位寻址	0x1
MASTER_10BIT	[4]	R/W	主机模式寻址控制 0: 7位寻址 1: 10位寻址	0x1
RESTART_EN	[5]	R/W	重复起始位控制 0: 禁止 1: 使能 控制主机模式下是否发送重复起始位，有些老的从机设备不支持重复起始位。	0x1
SLAVEn	[6]	R/W	从机模式控制位 0 = 使能从机模式	0x1

			1 = 禁用从机模式 注意如果此位写了0，那么第0位MASTER也必须写0。	
STOP_DET_CON	[7]	R/W	STOP_DET中断控制 (只在从机模式下有效): 0: 不管被寻址成功与否，都产生STOP_DET中断 1: 只有被成功寻址到，才产生STOP_DET 注意：在general call寻址模式下，如果STOP_DET_CON=1，即使该从机设备产生了ACK去响应general call地址，该设备也不会产生STOP_DET中断。 只有当传输地址跟从机地址(SADDR)匹配的时候，才会产生STOP_DET中断。	0x0
TX_EMPTY_CON	[8]	R/W	TX_EMPTY (I2C_RISR)状态控制 0: 当发送缓冲的指针小于或则等于I2C_TX_FLSEL寄存器设置的阈值时，TX_EMPTY位置1 1: 当发送缓冲的指针小于或则等于I2C_TX_FLSEL寄存器设置的阈值，并且最近一个地址或者数据的传输完成时，TX_EMPTY位置1	0x0
RXFULL_HLD	[9]	R/W	当接收FIFO溢出时的总线控制 0: 不占用总线 1: 占用总线 (hold bus)	0x0
BUS_CLR_EN	[11]	R/W	总线清除功能控制(只有在主机模式下有效)： 0: 禁用 1: 使能 从机模式下该功能不工作。	0x0

18.4.3 I2C_TADDR (I2C目标地址寄存器)

- 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																				SPECIAL		TADDR									
0	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	1	0	1	0	1
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	Reset Value
TADDR	[9:0]	R/W	主机模式下传输的目标地址。 当发送general call的时候，该位无效。 产生起始字节时，CPU只需要对该位写一次。	0x055
SPECIAL	[11:10]	R/W	I2C地址的特殊命令控制，表示I2C的操作类型：普通地址，general call或者起始字节。 0x0: 普通I2C_TADDR地址 0x1: 普通I2C_TADDR地址 0x2: General Call地址 – 在发送General Call后，只有写操作有效，任何尝试读的操作都会导致RISR寄存器中的第6位(TX_ABRT)位变1。I2C会一直保持在General Call模式直到SPECIAL位被配置为普通地址。 0x3: 起始字节 (START BYTE)	0x0

18.4.4 I2C_SADDR (I2C从机地址寄存器)

- 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																						SADDR									
0	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	1	0	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	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
SADDR	[9:0]	R/W	当I2C工作为从机时， SADDR为从机地址。7位寻址模式下，只有I2C_SADDR[6:0]有效。 该寄存器只有在I2C接口被禁用(I2C_ENABLE[0]=0)的时候可写。 注意：从机地址不能配置成保留地址：0x00 to 0x07, 或者 0x78 到 0x7f. 如果I2C_SADDR或者I2C_TADDR被配置成以上保留地址，那么I2C有可能工作异常。	0x055

18.4.5 I2C_DATA_CMD (I2C数据和命令寄存器)

- 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																					RESTART	STOP	CMD	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	W	W	W	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
DATA	[7:0]	R/W	I2C总线发送或者收到的数据。 在I2C进行读操作的时候，如果写这个寄存器，那么写操作将无效，如果读这个寄存器，那么返回的是I2C接口从总线上收到的数据。	0x0
CMD	[8]	W	读写控制。 1: 读 0: 写 在I2C工作在从机模式下，该位不能控制传输的方向。在工作在主机模式时，该位用来控制传输方向。 当一个命令被写入发送FIFO时，该位被用来区分读和写。在从机接收模式下，该位为无效操作位，因为不需要对该位进行操作。在从机发送模式下，0表示I2C_DATA_CMD中的数据需要被发送。	-
STOP	[9]	W	停止位(STOP)控制。 0: 不管发送 FIFO 是否为空，都不发送停止位。如果发送 FIFO 非空，那么主机根据 CMD 位的值继续当前的发送或者接收传输。如果发送 FIFO 为空，那么主机将 SCL 拉低占用总线，直到发送 FIFO 中有新的命令。 1: 不管发送 FIFO 是否为空，都在当前字节传输完成后发送停止位。如果发送 FIFO 非空，那么主机会立即发送起始位申请总线仲裁来尝试重新开始一个新的传输。	-

RESTART	[10]	W	重复起始位(RESTART)控制。 0: 如果 I2C_CR.RESTART_EN 为 1，并且传输的方向发生改变，那么 I2C 会发送一个重复起始位；如果 I2C_CR.RESTART_EN 为 0，那么 I2C 会发送停止位并且跟着再重新发送一个起始位。 1: 如果 I2C_CR.RESTART_EN 为 1，不管传输方向是否发生改变，I2C 都会在数据发送或者接收前(根据 CMD 位的值)发送一个重复起始位；如果 I2C_CR.RESTART_EN 为 0，那么 I2C 会发送停止位并且跟着再重新发送一个起始位。	-

18.4.6 I2C_SS_SCLH (I2C标准模式SCL高电平长度计数寄存器)

- 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																SCL_HCNT															
0	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	1	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
SCL_HCNT	[15:0]	R/W	该寄存器设置标准速度模式下，SCL时钟高电平的周期长度计数值。为了保证IO时序的正确性，该寄存器必须在任何I2C总线传输开始之前设置。 该寄存器必须在I2C接口被禁用(I2C_ENABLE[0]=0)时进行配置。 可配置的最小值为6，硬件上禁止写任何比6小的数，如果尝试写比6小的数，结果就是寄存器的值为6。 注意：该寄存器的值不能超过65525，因为该I2C模块使用了一个16位的计数器来检测I2C总线的空闲状态，当这个计数器的值达到SCL_HCNT+10的时候，标记为空闲。	0x0050

18.4.7 I2C_SS_SCLL (I2C标准模式SCL低电平长度计数寄存器)

- Address = Base Address + 0x0018

31	30	29	28	27	26	25	24	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																SCL_LCNT															
0	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	1	1	1	1	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
SCL_LCNT	[15:0]	R/W	该寄存器设置标准速度模式下，SCL时钟低电平的周期长度计数值。为了保证IO时序的正确性，该寄存器必须在任何I2C总线传输开始之前设置。 该寄存器必须在I2C接口被禁用(I2C_ENABLE[0]=0)时进行配置。 可配置的最小值为8，硬件上禁止写任何比8小的数，如果尝试写比8小的数，结果就是寄存器的值为8。	0x005e

18.4.8 I2C_FS_SCLH (I2C高速模式SCL高电平长度计数寄存器)

- Address = Base Address + 0x001C

31	30	29	28	27	26	25	24	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																SCL_HCNT															
0	0	0	0	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
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
SCL_HCNT	[15:0]	R/W	该寄存器设置高速模式下，SCL时钟高电平的周期长度计数值。为了保证IO时序的正确性，该寄存器必须在任何I2C总线传输开始之前设置。 该寄存器必须在I2C接口被禁用(I2C_ENABLE[0]=0)时进行配置。 可配置的最小值为6，硬件上禁止写任何比6小的数，如果尝试写比6小的数，结果就是寄存器的值为6。	0x000c

18.4.9 I2C_FS_SCLL (I2C高速模式SCL低电平长度计数寄存器)

- Address = Base Address + 0x0020

31	30	29	28	27	26	25	24	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																SCL_LCNT															
0	0	0	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	1	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
SCL_LCNT	[15:0]	R/W	该寄存器设置高速或者超高速模式下，SCL时钟低电平的周期长度计数值。为了保证IO时序的正确性，该寄存器必须在任何I2C总线传输开始之前设置。 该寄存器必须在I2C接口被禁用(I2C_ENABLE[0]=0)时进行配置。 可配置的最小值为8，硬件上禁止写任何比8小的数，如果尝试写比8小的数，结果就是寄存器的值为8。	0x001a

18.4.10 I2C_RX_FLSEL (I2C接收FIFO阈值寄存器)

- Address = Base Address + 0x002C

31	30	29	28	27	26	25	24	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																								RX_FLSEL							
0	0	0	0	0	0	0	0	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
RX_FLSEL	[7:0]	R/W	接收FIFO阈值。 FIFO中数据的数量超过或者等于该阈值时，将触发RX_FULL中断(I2C_RISR的第2位)。该阈值不能超过FIFO深度(8)，如果设置的值超过了FIFO深度，那么实际写入的值为FIFO的深度值。 该寄存器的值0表示有1个数据，7表示阈值为8个数据。	0x0

18.4.11 I2C_TX_FLSEL (I2C发送FIFO阈值寄存器)

- Address = Base Address + 0x0030

31	30	29	28	27	26	25	24	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																								TX_FLSEL							
0	0	0	0	0	0	0	0	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
TX_FLSEL	[7:0]	R/W	发送FIFO阈值。 FIFO中数据的数量少于或者等于该阈值时，将触发TX_EMPTY中断(I2C_RISR的第4位)。该阈值不能超过FIFO深度(8)，如果设置的值超过了FIFO深度，那么实际写入的值为FIFO的深度值。 该寄存器的值0表示0个数据，7表示阈值为8个数据。	0x0

18.4.12 I2C_RX_FL (I2C接收FIFO状态寄存器)

- Address = Base Address + 0x0034

31	30	29	28	27	26	25	24	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																												RX_FL			
0	0	0	0	0	0	0	0	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
RX_FL	[3:0]	R	接收FIFO中有效数据的个数。	0x0

18.4.13 I2C_TX_FL (I2C发送FIFO状态寄存器)

- Address = Base Address + 0x0038

31	30	29	28	27	26	25	24	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																												TX_FL			
0	0	0	0	0	0	0	0	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
TX_FL	[3:0]	R	发送FIFO中有效数据的个数。	0x0

18.4.14 I2C_ENABLE (I2C使能寄存器)

- Address = Base Address + 0x003C

31	30	29	28	27	26	25	24	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																												RECOVER_EN	RSVD	ABORT	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	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	R	RW	RW

Name	Bit	Type	Description	Reset Value
ENABLE	[0]	R/W	使能I2C模块 0：禁用 1：使能 当I2C被禁用，会发生下面的事件： - 发送FIFO和接收FIFO被清空 - 状态标志位仍然保持有效状态直到I2C进入空闲状态 如果模块正在发送状态，那么它会在当前传输完成后停止并且删除发送缓冲中的内容。如果模块正在接收，那么I2C会在当前字节接收完成后停止，并且不发送应答位。	0x0
ABORT	[1]	R/W	中止I2C传输。 0: 中止操作没有发生或则中止操作已完成 1: 中止操作正在进行 在主机模式下，软件可以将该位置1，用来中止I2C传输。只有在I2C使能状态下(ENABLE=1)，才可以对该位置1，否则任何写操作都无效。发起中止操作后，I2C会在当前传输完成后产生一个停止位并且清空发送FIFO，然后在中止操作后产生TX_ABRT中断。ABORT位会在中止操作后自动清零。	0x0
RECOVER_EN	[3]	R/W	SDA被拉低锁死后的恢复功能控制 0: 禁用 1: 使能 如果TX_ABRT中断(I2C_ABRT[17])显示SDA被拉低锁死，	0x0

			那么该位可以控制使能SDA的恢复机制(发送最多9个SCL时钟和停止位尝试释放SDA)，然后该位自动清零。	

18.4.15 I2C_STATUS (I2C状态寄存器)

- Address = Base Address + 0x0040

31	30	29	28	27	26	25	24	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																ENABLE				RECOVER_FAIL	RSVD				SLV_BUSY	MST_BUSY	RFF	RFNE	TFE	TFNF	BUSY
0	0	0	0	0	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
R	R	R	R	R	R	R	R	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	[0]	R	I2C工作状态 0: 空闲 1: 工作中	0x0
TFNF	[1]	R	发送FIFO未满 0: 发送FIFO已满 1: 发送FIFO未满 当发送FIFO含有至少1个或者多个数据时置1，当FIFO为满时会被清0。	0x1
TFE	[2]	R	发送FIFO为空 0: 发送FIFO非空 1: 发送FIFO为空 当发送FIFO完全为空时置1，当FIFO含有至少1个或者多个数据时会被清0。该位不会产生中断。	0x1
RFNE	[3]	R	接收FIFO非空 0: 接收FIFO为空 1: 接收FIFO非空 当接收FIFO含有至少1个或者多个数据时置1，当FIFO为空时会被清0。	0x0
RFF	[4]	R	接收FIFO已满 0: 接收FIFO未满 1: 接收FIFO已满	0x0

			当接收FIFO已满时置1，当FIFO含有至少1个或者多个空位时会被清0。	
MST_BUSY	[5]	R	主机模式的状态机工作状态 0: 主机模式状态机处于空闲状态，I2C 的主机部分不在工作 1: 主机模式状态机处于工作状态，I2C 的主机部分正在工作 注意：I2C_STATUS[0] – 也就是工作状态位 – 是 MST_BUSY 和 SLV_BUSY 位的或逻辑结果。	0x0
SLV_BUSY	[6]	R	从机模式的状态机工作状态 0: 从机模式状态机处于空闲状态，I2C 的从机部分不在工作 1: 从机模式状态机处于工作状态，I2C的从机部分正在工作	0x0
RECOVER_FAIL	[11]	R	SDA拉低锁死的恢复状态 0: 恢复没有失败 1: 恢复失败 该位表示SDA拉低锁死后的恢复机制是否成功。	0x0
ENABLE	[15:12]	R	同I2C_ENABLE	0x0

18.4.16 I2C_SDA_TSETUP (I2C SDA Setup时间寄存器)

- Address = Base Address + 0x0048

31	30	29	28	27	26	25	24	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																								SDA_TSETUP							
0	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	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	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
SDA_TSETUP	[7:0]	R/W	Setup时间的长度使用下面公式计算： [(SDA_TSETUP - 1) * (PCLK周期)] 如果需要的延时是1000ns，PCLK频率是10MHz，那么建议设置SDA_TSETUP的值为11. SDA_TSETUP的最小值为2.	0x64

18.4.17 I2C_SDA_THOLD (I2C SDA Hold 时间寄存器)

- Address = Base Address + 0x004C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								SDA_RX_THOLD								SDA_TX_THOLD															
0	0	0	0	0	0	0	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	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
SDA_TX_THOLD	[15:0]	R/W	当I2C作为发送端时，设置SDA的hold时间，单位为PCLK的周期。 SDA_TX_THOLD的值必须大于最小的hold时间： - 主机模式下1个时钟周期 - 从机模式下7个时钟周期	0x1
SDA_RX_THOLD	[23:16]	R/W	当I2C作为接收端时，设置SDA的hold时间，单位为PCLK的周期。	0x0

18.4.18 I2C_SPKLEN (I2C毛刺干扰滤波控制寄存器)

- 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
RSVD																								SPKLEN							
0	0	0	0	0	0	0	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
SPKLEN	[7:0]	R/W	该寄存器设置毛刺干扰过滤逻辑可以过滤的SCL和SDA上最长的毛刺信号长度，以PCLK周期为单位。 该寄存器必须在I2C总线传输开始之前设置，以保证稳定的传输。 该寄存器只有在I2C接口被禁止(I2C_ENABLE[0]=0)的时候可以设置，否则写操作无效。可以设置的最小值为1，硬件禁止写比1小的值，写0仍然为1。	0x1

18.4.19 I2C_MISR (I2C中断状态寄存器)

- 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
RSVD																	SCL_S_LOW	RSVD	RESTART_DET	GEN_CALL	START_DET	STOP_DET	BUSY	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
0	0	0	0	0	0	0	0	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
SCL_S_LOW	14	R	中断状态位 (I2C_IMSCR寄存器中使能后才有效) 参考I2C_RISR寄存器中的具体说明。	0x0
RESTART_DET	12			
GEN_CALL	11			
START_DET	10			
STOP_DET	9			
BUSY	8			
RX_DONE	7			
TX_ABRT	6			
RD_REQ	5			
TX_EMPTY	4			
TX_OVER	3			
RX_FULL	2			
RX_OVER	1			
RX_UNDER	0			

18.4.20 I2C_IMCR (I2C中断使能寄存器)

- Address = Base Address + 0x005C

31	30	29	28	27	26	25	24	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																	SCL_S_LOW	RSVD	RESTART_DET	GEN_CALL	START_DET	STOP_DET	BUSY	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
0	0	0	0	0	0	0	0	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
SCL_S_LOW RESTART_DET GEN_CALL START_DET STOP_DET BUSY RX_DONE TX_ABRT RD_REQ TX_EMPTY TX_OVER RX_FULL RX_OVER RX_UNDER	14 12 11 10 9 8 7 6 5 4 3 2 1 0	R	相应的中断使能 0: 禁止 1: 使能 参考I2C_RISR寄存器中的具体说明。	0x0

18.4.21 I2C_RISR (I2C原始中断状态寄存器)

- Address = Base Address + 0x0060

31	30	29	28	27	26	25	24	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																	SCL_S_LOW	RSVD	RESTART_DET	GEN_CALL	START_DET	STOP_DET	BUSY	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
0	0	0	0	0	0	0	0	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
RX_UNDER	[0]	R	RX_UNDER 状态 0: RX under-flow 中断没有发生 1: RX under-flow 中断发生 当接收 FIFO 为空，处理器尝试通过 I2C_DATA_CMD 寄存器去读取接收数据时置 1。如果 I2C 模块被关闭 (I2C_ENABLE[0]=0)，那么这位的值会一直保持，直到主机或从机的状态机变为空闲状态。	0x0
RX_OVER	[1]		RX_OVER 状态 0: RX 溢出中断未发生 1: RX 溢出中断发生 如果接收缓冲被填满 8 个数据，而且处理器仍然尝试往 I2C_DATA_CMD 寄存器里写数据时置 1。如果 I2C 模块被关闭 (I2C_ENABLE[0]=0)，那么这位的值会一直保持，直到主机或从机的状态机变为空闲状态。	
RX_FULL	[2]	R	RX_FULL 状态 0: RX 缓冲区未满 (<RX_FLSEL) 1: RX 缓冲区已满 (>=RX_FLSEL) 当缓冲区数据个数等于或者大于 I2C_RX_FLSEL 寄存器中 RX_FLSEL 阈值时置 1，当数据个数小于阈值时会自动清零。当该模块被禁止(I2C_ENABLE[0]=0)时，接收 FIFO 会被清除并且处于复位状态，也就是接收 FIFO 是未满状态，所以一旦 I2C_ENABLE[0]=0，该位就会被清零。	0x0

TX_OVER	[3]	R	TX_OVER 状态 0: TX 溢出中断未发生 1: TX 溢出中断发生 如果发送缓冲被填满 8 个数据，而且处理器仍然尝试往 I2C_DATA_CMD 寄存器里写命令时置 1。如果 I2C 模块被关闭 (I2C_ENABLE[0]=0)，那么这位的值会一直保持，直到主机或从机的状态机变为空闲状态。	0x0
TX_EMPTY	[4]	R	TX_EMPTY 状态 0: TX 缓冲区非空 (>TX_FLSEL) 1: TX 缓冲区为空 (<=TX_FLSEL) TX_EMPTY 中断状态的工作行为跟 I2C_CR 寄存器中 TX_EMPTY_CON 的选择有关。 - 当 TX_EMPTY_CON = 0: 如果发送缓冲等于或者小于 I2C_TX_FLSEL 寄存器中设置的阈值，该位置 1。 - 当 TX_EMPTY_CON = 1: 如果发送缓冲等于或者小于 I2C_TX_FLSEL 寄存器中设置的阈值，并且当前传输的地址或者数据完成，该位置 1。 当缓冲区数据个数大于阈值时，该位会自动被清零。当 I2C_ENABLE[0]=0 时，发送 FIFO 会被清除并且处于复位状态，所以该位被置 1。	0x0
RD_REQ	[5]	R	RD_REQ 状态 (从机模式) 0: 没有读请求 1: 收到读请求 当 I2C 工作在从机模式并且另外一个 I2C 主机正在尝试从该从机设备读取数据时，该位置 1。I2C 会占用并且让总线处于等待状态(SCL=0)，直到该中断被响应并处理，意思是该从机被远端主机寻址到并且被要求传输数据。处理器必须响应该中断并且将要求传输的数据写入 I2C_DATA_CMD 寄存器。	0x0
TX_ABRT	[6]	R	TX_ABRT 状态 0: TX_ABRT 中断没有发生 1: TX_ABRT 中断发生 该位表示 I2C 作为发送端，不能够完成发送 FIFO 中要求	0x0

			完成的动作。这个情况在主机和从机模式下都有可能发生，意思是“发送被中止”。 当该位被置 1 时，I2C_TX_ABRT 寄存器会保存发送中止的原因。	
RX_DONE	[7]	R	RX_DONE 状态 0: RX_DONE 中断没有发生 1: RX_DONE 中断发生 当 I2C 作为接收发送端，如果主机不应答发送的字节，那么该位置 1。该状态一般发生在传输的最后一个字节，表示传输已经完成。	0x0
BUSY	[8]	R	I2C 工作状态 0: I2C 空闲 1: I2C 工作中 该位是 I2C 的工作状态，状态会一直保持直到被下面 3 种方式清除： - 禁用 I2C 模块 - 将 I2C_ICR 寄存器的 BUSY 位写 1 - 系统复位 一旦该位被置 1 后，只有通过上面 3 中方法清零。即使 I2C 模块处于空闲状态，该位仍然为 1，除非被上述 3 种方法清零。	0x0
STOP_DET	[9]	R	停止位检测状态 0: 没有检测到停止位 1: 检测到停止位 表示 I2C 总线上是否产生了停止位，不管是主机模式还是从机模式。 注意在从机模式： - 如果 I2C_CR[7]=1'b1 (STOP_DET_CON)，只有从机被成功寻址时，才会产生 STOP_DET 中断。 在 general call 寻址模式下，如果 STOP_DET_CON=1，即使该从机设备产生了 ACK 去响应 general call 地址，该设备也不会产生 STOP_DET 中断。只有当传输地址跟从机地址(SADDR)匹配的时候，才会产生 STOP_DET 中断。 - 如果 I2C_CR[7]=1'b0 (STOP_DET_CON)，不管是否被寻址，都会产生 STOP_DET 中断。	0x0

START_DET	[10]	R	起始位检测状态 0: 没有检测到起始位 1: 检测到起始位 表示 I2C 总线上是否产生了起始位，不管是主机模式还是从机模式。	0x0
GEN_CALL	[11]	R	General call 状态 0: 没有收到 general call 1: 收到 general call 当收到 general call 地址并且应答了该地址，该位置 1。该状态位会一直保持，直到关闭 I2C 或者对 I2C_ICR 寄存器中相应位写 1 来清零该位。I2C 将接收到的数据保存到接收缓冲区中。	0x0
RESTART_DET	[12]	R	重复起始位检测状态 (从机模式) 0: 没有检测到重复起始位 1: 检测到重复起始位 当 I2C 工作在从机模式时，表示 I2C 接口上是否产生了重复起始位，并且该从机是被寻址到的从机。	0x0
SCL_S_LOW	[14]	R	SCL 锁死状态 0: SCL 没有被拉低锁死 1: SCL 被拉低锁死 表示 SCL 是否被拉低锁死，锁死时间超过了 I2C_SCL_TOUT 个 PCLK 周期的时长。	0x0

18.4.22 I2C_ICR (I2C中断清除寄存器)

- Address = Base Address + 0x0064

31	30	29	28	27	26	25	24	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																SCL_S_LOW	RSVD	RESTART_DET	GEN_CALL	START_DET	STOP_DET	BUSY	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER	
0	0	0	0	0	0	0	0	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	W	R	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description	Reset Value
SCL_S_LOW	14	R	相应的中断清除 0: 无效 1: 清除中断状态 参考I2C_RISR寄存器中的具体说明。	0x0
RESTART_DET	12			
GEN_CALL	11			
START_DET	10			
STOP_DET	9			
BUSY	8			
RX_DONE	7			
TX_ABRT	6			
RD_REQ	5			
TX_EMPTY	4			
TX_OVER	3			
RX_FULL	2			
RX_OVER	1			
RX_UNDER	0			

18.4.23 I2C_SCL_TOUT (I2C SCL锁死超时控制寄存器)

- Address = Base Address + 0x006C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCL_TOUT																															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RW	RW	RW	RW	RW	RW	RW	RW	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
SCL_TOUT	[31:0]	R/W	如果I2C检测到SCL被锁死在低电平超过SCL_TOUT个PCLK周期时，会产生SCL拉低锁死中断。	0xFFFFFFFF

18.4.24 I2C_SDA_TOUT (I2C SDA锁死超时控制寄存器)

- Address = Base Address + 0x0070

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDA_TOUT																															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RW	RW	RW	RW	RW	RW	RW	RW	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
SDA_TOUT	[31:0]	R/W	如果 I2C 检测到 SDA 被锁死在低电平超过 SDA_TOUT 个 PCLK 周期，那么 I2C 可以通过使能 RECOVER_EN (I2C_ENABLE[3])寄存器来打开 SDA 的恢复功能。	0xFFFFFFFF

18.4.25 I2C_TX_ABRT (I2C发送中止状态寄存器)

- 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														SDA_S_LOW	USER_ABRT	SLVRD_INTX	SLV_ARBLOST	SLVFLUSH_TX	ARB_LOST	MASTER_DIS	10B_RD_NRSTRT	SBYTE_NRSTRT	RSVD	SBYTE_ACK	RSVD	GCALL_READ	GCALL_NACK	TXDATA_NACK	10ADDR2_NACK	10ADDR1_NACK	7ADDR_NACK
0	0	0	0	0	0	0	0	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
7ADDR_NACK	[0]	R	1: 主机工作在 7 位寻址模式并且没有任何从机应答主机发送的地址	0x0
10ADDR1_NACK	[1]	R	1: 主机工作在 10 位寻址模式并且没有任何从机应答主机发送的 10 位地址的第一个字节	0x0
10ADDR2_NACK	[2]	R	1: 主机工作在 10 位寻址模式并且没有任何从机应答主机发送的 10 位地址的第二个字节	0x0
TXDATA_NACK	[3]	R	1: 该位只在主机模式下有效。主机收到了地址的应答，但是当它发送数据到该地址时，却没有收到任何从机的应答。	0x0
GCALL_NACK	[4]	R	1: I2C 在主机模式发送了一个 general call 并且总线上没有从机应答该 general call	0x0
GCALL_READ	[5]	R	1: I2C 在主机模式发送了一个 general call 但是用户的程序在 general call 后从总线上读数据 (I2C_DATA_CMD[9]被设置为 1)	0x0
SBYTE_ACK	[7]	R	1: 主机发送了一个起始字节并且这个起始字节被应答了(错误行为)	0x0
SBYTE_NRSTRT	[9]	R	1: 重复起始位被禁止(RESTART_EN (I2C_CR[5]) = 0)并且用户正在尝试发送一个起始字节。 如果要清除第 9 位，SBYTE_NRSTRT 的触发源必须固定；重复起始位必须使能(I2C_CR[5]=1)，SPECIAL 位不能为 0x3 (I2C_TADDR[11:10])。一旦触发源固定，该位就可以跟清除其它位的方法一样进行清除，否则该位在清除后又会被置起。	0x0
10B_RD_NRSTRT	[10]	R	1: 重复起始位被禁止(RESTART_EN (I2C_CR[5]) = 0)并且主机在10位寻址模式发送了一个读命令。	0x0
MASTER_DIS	[11]	R	1: 用户在主机模式没有使能的情况下尝试主机操作	0x0

ARB_LOST	[12]	R	1: 主机丢失了仲裁，或者在I2C_TX_ABRT[14]置1的情况下从机发送端丢失了仲裁	0x0
SLVFLUSH_TX	[13]	R	1: 从机收到了一个读命令并且发送 FIFO 中有数据，所以从机产生了一个 TX_ABRT 中断去清除发送 FIFO 中的旧数据。	0x0
SLV_ARBLOST	[14]	R	1: 从机在发送数据到远端主机时丢失了总线。 I2C_TX_ABRT[12]会被同时置 1。	0x0
SLVRD_INTX	[15]	R	1: 当处理器(从机模式)响应了一个给远程主机发送数据的请求时，用户对 I2C_DATA_CMD 寄存器的第 8 位 CMD 写 1。	0x0
USER_ABRT	[16]	R	只在主机模式有效。主机检测到发送中止(IC_ENABLE[1]).	0x0
SDA_S_LOW	[17]	R	只在主机模式有效。主机检测到 SDA 被锁死在低电平超过 I2C_SDA_TOUT 个 PCLK 周期。	0x0

18.4.26 I2C_GCALL (I2C General Call控制寄存器)

- 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
																															ACK_GCALL
0	0	0	0	0	0	0	0	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
ACK_GCALL	[0]	R/W	应答General Call 1: I2C使用ACK应答General Call 0: I2C不产生General Call中断	0x1

18.4.27 I2C_NACK (I2C从机NACK控制寄存器)

- Address = Base Address + 0x007C

31	30	29	28	27	26	25	24	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																															NACK
0	0	0	0	0	0	0	0	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	Reset Value
NACK	[0]	R/W	NACK的产生控制 1: 在数据字节收到后产生NACK 0: 按正常情况产生NACK/ACK 该位只有在I2C工作在从机接收端时有效。如果该位为1，I2C在收到一个数据字节后只会产生一个NACK，此后的数据传输会被中止并且收到的数据不会被存入接收缓冲区内。如果该位为0，I2C按照正常的情况产生NACK/ACK。	0x0

19 电容式触摸按键传感器（TOUCH）

19.1 概述

此MCU内嵌了一个最大支持17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。

注：如果系列内芯片不具有本外围，那它就不具备本章描述的相关资源。具体参考芯片的数据手册。

19.1.1 特性

- 最大支持17通道按键检测
- 支持低功耗模式，并基于扫描值偏差自动唤醒CPU。
- 支持每个通道具有的确定的扫描时间
- 支持事件计数器，可通过配置事件计数器（最大15）触发相应中断
- 多种扫描触发模式。
  - 软件触发
  - 硬件触发（RTC，ETCB，LPT）

19.1.2 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	-	-

19.2 功能描述

19.2.1 模块框图

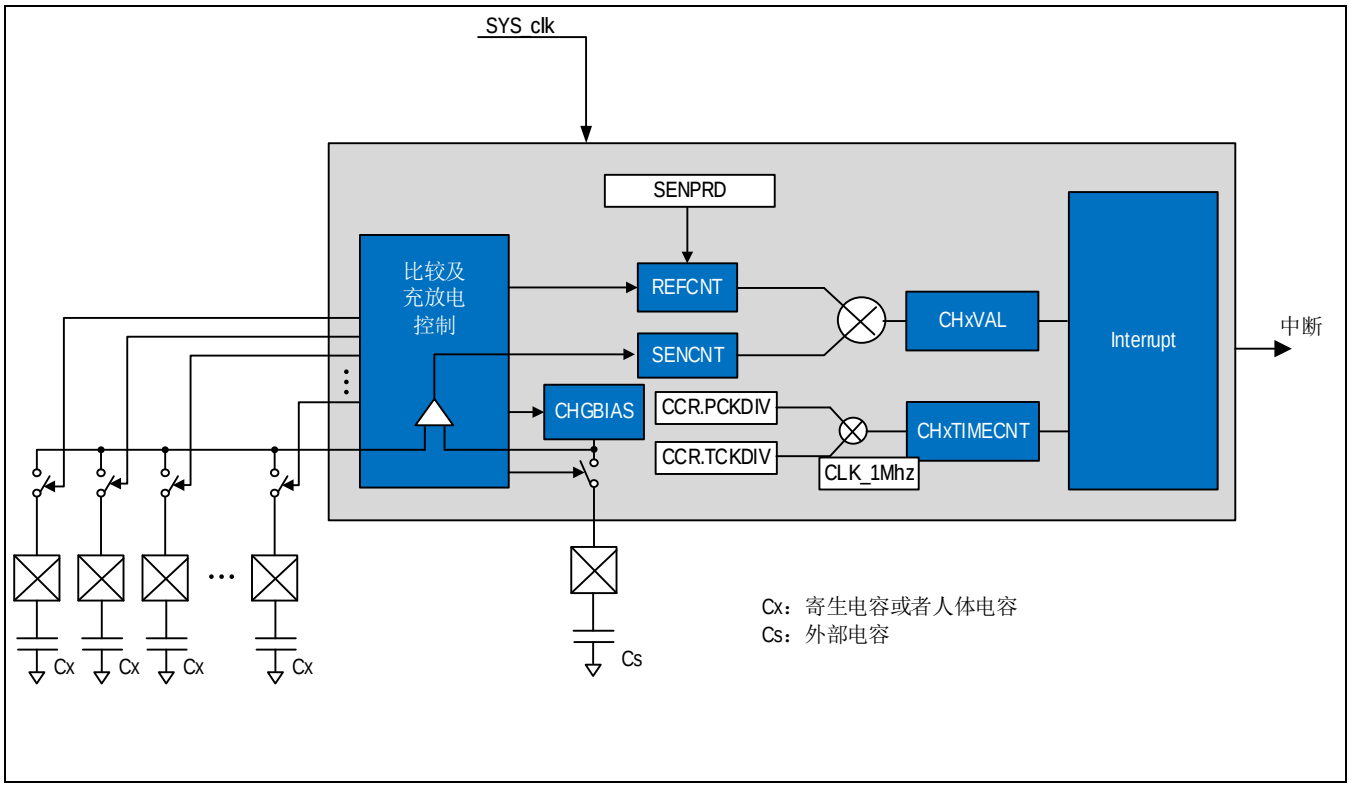


Figure 22-1 Touch Sensor模块框图

19.2.2 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。

系统时钟由随机时钟 MFO 调制后控制 TOUCH IO 对触摸电容充放电（固定频率，随机相位）。充电电流由内部 LDO 提供，LDO 的输出电流镜像给感应振荡器 S-OSC，控制 S-OSC 输出频率。因为充电频率固定，S-OSC 输出频率正比于 TOUCH IO 负载电容，在 R-OSC 经过 N 个周期所确定的固定时间内，SFO 的周期数将被一个内部采样计数器记录（CHxCNT）。寄生电容变大时，CHxCNT 值会变大；寄生电容变小时，CHxCNT 值会随之变小。

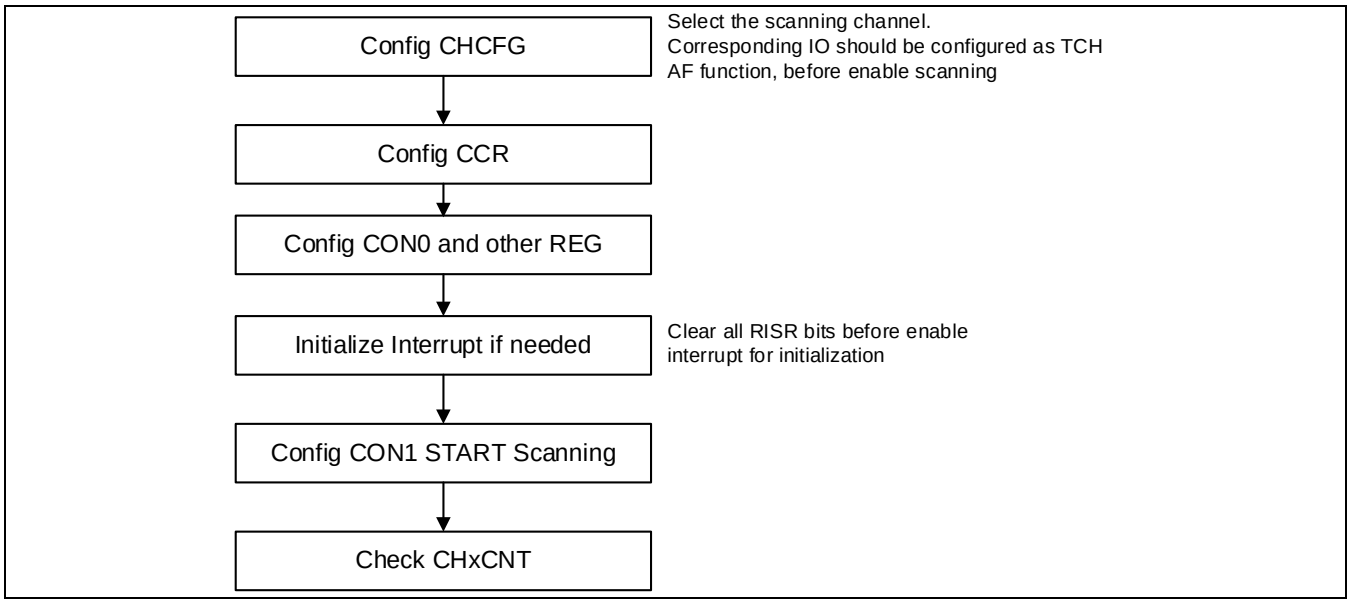


Figure 22-2 软件配置流程

19.2.3 扫描启动和扫描触发方式设置

19.2.3.1 扫描工作时钟

整个通道扫描控制模块的基础工作时钟为system clk or Tclk 选择。正常工作时使用system clk，低功耗模式使用Tclk。

19.2.3.2 扫描启动

在扫描启动前，需要使能TCH_CCR中CLKEN, 设置扫描通道数（Sequence模式每次扫描通道数为TCH_CON0[SEQLEN]+1），通过TCH_SEQxCON配置各通道参数，最后置位TCH_CON1[START]位启动扫描。

19.2.3.3 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C0功能。需要扫描的通道通过TCH_SEQxCON寄存器设置。一轮扫描结束后，可以再次设置TCH_CON1[START]开始下一轮的扫描，或者通过硬件自动触发下一轮扫描。

19.2.3.4 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_SYNCR中的SYNCxEN位设置。

支持三种触发源：RTC(TCH_SYNCR[SYNC0EN])，ETCB(TCH_SYNCR[SYNC1EN])，LPT(TCH_SYNCR[SYNC2EN])。RTC触发源信号选择由RTC_CR寄存器的TKEYTRG位控制，ETCB触发源则由ETCB控制，可以选择ETCB支持的任意触发输入，LPT触发源是LPT_PEND周期结束事件(不需要使能中断)。支持

单次触发模式和多次触发模式，通过TCH_SYNCR[OSTMD]配置，当该输入通道被设置为一次性触发模式后，在一次触发事件被检测到后，该通道将不允许后续的触发事件通过，直到被软件重置（REARM）后才允许新的触发事件通过。

同样的，TOUCH也可产生对其他外设的任务的触发信号，TOUCH触发输出源由TCH_EVTRG，TCH_EVPS寄存器进行配置。

19.2.3.5 固定扫描时间

TCH可配置每个通道的固定扫描时间，固定扫描时间的时钟基于system clk or Tclk分频出的1Mhz目标时钟。Sequence模式扫描时间配置TCH_SEQxCON[SCANTIME] ，扫描目标值配置(TCH_ SEQxCON [SENPRD], 4’b1111)。

扫描目标值需要在配置的扫描时间内能够完成，如果在扫描时间到达时，REFCNT计数值未达到配置的扫描目标值，则视为扫描超时异常，可通过TCH_OVT查看通道超时状态，本次扫描结果无效，需要调整该通道扫描时间和扫描目标值以保证扫描能够在此时间内达到扫描配置目标值。

19.2.4 中断产生

每个通道都有独立的扫描完成中断。通过设置TCH_IER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。

19.3 寄存器说明

19.3.1 寄存器表

- Base Address: 0x4002_0000

Register	Offset	Description	Reset Value
TCH_CCR	0x000	Clock Control Register	0x0000_0000
TCH_CON0	0x004	Touch Sensor General Control Register0	0x0000_0000
TCH_CON1	0x008	Touch Sensor General Control Register1	0x0000_0000
TCH_VALBUF	0x014	Touch Sample Value Buffer	0x0000_0000
TCH_SENCNT	0x018	Current Sensor Node Discharge Counter Value	0x0000_0000
TCH_TCHCNT	0x01C	Current Touch Sample Counter Value	0x0000_0000
TCH_THR	0x020	Touch OFFSET Threshold Register	0x0000_0000
TCH_RISR	0x028	Touch Senor Raw Interrupt Status Register	0x0000_0000
TCH_IER	0x02C	Touch Senor Interrupt Masking Control Register	0x0000_0000
TCH_ICR	0x030	Touch Senor Interrupt Clear Control Register	0x0000_0000
TCH_RWSR	0x034	Touch Sample Value R/W Status Register	0x0000_0000
TCH_OVW_THR	0x038	Touch OverThreshold Status Register	0x0000_0000
TCH_OVF	0x03C	Touch Overflow status Register	0x0000_0000
TCH_OVT	0x040	Touch OverTIME status Register	0x0000_0000
TCH_SYNCR	0x044	Touch Sync Control Register	0x0000_0000
TCH_EVTRG	0x048	Touch Trigger Event Generation Control Register	0x0000_0000
TCH_EVPS	0x04C	Touch Trigger Event Generation Prescaler	0x0000_0000
TCH_EVSWF	0x050	Touch Trigger Event Software Force Generation	0x0000_0000
TCH_CHxVAL[18]	0x1000 - 0x1044	CHx Touch Sample Value Register	0x0000_0000
TCH_SEQxCON[18]	0x1048 - 0x108C	SEQ Mode Control Register for each sub-sequence	0x0000_0000

NOTE:

19.3.2 TCH_CCR (时钟控制寄存器)

- 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																PCKDIV								TCKDIV						TEST_MODE	CLKEN
0	0	0	0	0	1	0	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
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
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	0x0
TEST_MODE	[1]	RW	TOUCH TEST MODE，用于调整TRIM值	0x0
TCKDIV	[7:2]	RW	时钟分频控制，基于IMOSC分频，目标时钟频率1Mhz，$F_{Target} = F_{TCLK} / (TCKDIV +1)$	0x00
PCKDIV	[15:8]	RW	时钟分频控制，基于SYSCLK分频，目标时钟频率1Mhz，$F_{Target} = F_{PCLK} / (PCKDIV +1)$	0x00
IDCODE	[31:16]	R	当前TOUCH硬件版本信息	0x0401

19.3.3 TCH_CON0 (通用控制寄存器0)

- 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				OTHRCNTCLR	OTHRCNT			RSVD		TSCANSTB		RSVD	DIO_EN	DSR		RSVD	IDLE_PDEN	PSEL	RSSEL	CKFEQSEL	CKRND	CKSPR	ECLVL	SEQLEN						MODE	HMEN
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	0	0
R	R	R	R	RW	RW	RW	RW	R	R	R	R	R	R	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
HMEN	[0]	RW	TOUCH使能控制位 0h： 关闭触控Hard-macro 模块 1h： 开启触控Hard-macro 模块	0x0
MODE	[1]	RW	扫描模式启动位： 0h：RSVD 1h：使能SEQUENCE模式	0x0
SEQLEN	[6:2]	RW	SEQUENCE模式下，扫描次数控制： 00000b： 1次扫描 00001b： 2次扫描 … 10000b： 17次扫描	0x00
ECLVL	[8:7]	RW	外部C0端口电压选择 0h：1V 1h：2V 2h：3V 3h：3.6V	0x0
CKSPR	[9]	RW	扫描时钟扩频控制 0h：关闭扫描时钟扩频 1h：打开扫描时钟扩频	0x0
CKRND	[10]	RW	扫描时钟频率随机化控制 0h：关闭扫描时钟随机化 1h：打开扫描时钟随机化	0x0
CKFEQSEL	[11]	RW	扫描频率选择 0h: 低频	0x0

			1h: 高频	
RSSEL	[12]	RW	TCH_OVW_THR寄存器状态选择 0h: OVW状态 1h: OverTHR状态	0x0
PSEL	[13]	RW	Power selection 0h: FVR 1h: AVDD	0x1
IDLE_PDEN	[14]	RW	扫描空闲时，触控Hard-macro 模块使能控制 0h：禁止 1h：使能	0x0
DSR	[17:16]	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 0h： 高阻状态 1h： 低电平输出 2h： 高电平输出 3h： 高阻状态	0x0
DIO_EN	[18]	RW	滤波二极管使能控制 0h：禁止 1h：使能	0x1
TSCANSTB	[21:20]	RW	发生LowPower-Normal模式切换，扫描稳定次数 0h：1次扫描 1h：2次扫描 2h：3次扫描 3h：4次扫描	0x0
OTHRCNT	[26:24]	RW	超出阈值判定 0h：超出阈值1次认为超出阈值 1h：连续2次超出阈值认为超出阈值 … 7h：连续8次超出阈值认为超出阈值	0x0
OTHRCNTCLR	[27]	W	软件清零当前OTHRCNT 0h：无效 1h：复位OTHRCNT	0x0

19.3.4 TCH_CON1 (通用控制寄存器1)

- 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																															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

19.3.5 TCH_VALBUF (扫描采样值缓存寄存器)

- 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																VALBUF															
0	0	0	0	0	0	0	0	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
VALBUF	[15:0]	R	当前扫描通道采样值	0x0000

19.3.6 TCH_SENCNT (Senser Node扫描计数器寄存器)

- 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																SENCNT															
0	0	0	0	0	0	0	0	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
SENCNT	[15:0]	R	Sensor Node 充放电次数计数器当前值	0x0000

19.3.7 TCH_TCHCNT (TCHCNT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																TCHCNT															
0	0	0	0	0	0	0	0	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
TCHCNT	[15:0]	R	扫描时间计数器当前值	0x0000

19.3.8 TCH_THR (扫描偏差阈值寄存器)

- 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																OFF_THR															
0	0	0	0	0	0	0	0	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
OFF_THR	[15:0]	RW	扫描偏差阈值	0x0000

19.3.9 TCH_RISR (原始中断状态寄存器)

- 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																										OVR_TIME	OVR_OVW	OVR_FLW	OVR_THR	SEQ_DNE	SIN_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
SIN_DNE	[0]	R	SEQ模式一个通道扫描完成中断	0x0
SEQ_DNE	[1]	R	SEQUENCE模式所有通道扫描完成中断	0x0
OVR_THR	[2]	R	两次扫描OFFSET值超出扫描OFFSET阈值中断	0x0
OVR_FLW	[3]	R	TCHCNT溢出中断	0x0
OVR_OVW	[4]	R	TCH_CHxVAL over write 中断	0x0
OVR_TIME	[5]	R	扫描超时中断	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				

19.3.10 TCH_IER (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
RSVD																										OVR_TIME	OVR_OVW	OVR_FLW	OVR_THR	SEQ_DNE	SIN_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	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
SIN_DNE	[0]	RW	SEQ模式一个通道扫描完成中断使能	0x0
SEQ_DNE	[1]	RW	SEQUENCE模式所有通道扫描完成中断使能	0x0
OVR_THR	[2]	RW	两次扫描OFFSET值超出扫描OFFSET阈值中断使能	0x0
OVR_FLW	[3]	RW	TCHCNT溢出中断使能	0x0
OVR_OVW	[4]	RW	TCH_CHxVAL over write 中断使能	0x0
OVR_TIME	[5]	RW	扫描超时中断使能	0x0
该寄存器用于使能中断。 0： 关闭中断 1： 打开中断				

19.3.11 TCH_ICR (中断清除控制寄存器)

- 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																										OVR_TIME	OVR_OVW	OVR_FLW	OVR_THR	SEQ_DNE	SIN_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	W	W	W	W	W	W

Name	Bit	RW	Description	Reset Value
SIN_DNE	[0]	W	清除SEQ模式一个通道扫描完成中断	0x0
SEQ_DNE	[1]	W	清除SEQUENCE模式所有通道扫描完成中断	0x0
OVR_THR	[2]	W	清除两次扫描OFFSET值超出扫描OFFSET阈值中断	0x0
OVR_FLW	[3]	W	清除TCHCNT溢出中断	0x0
OVR_OVW	[4]	W	清除TCH_CHxVAL over write 中断	0x0
OVR_TIME	[5]	W	清除扫描超时中断	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				

19.3.12 TCH_RWSR (通道采样值读写状态寄存器)

- 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															CH16_RWSR	CH15_RWSR	CH14_RWSR	CH13_RWSR	CH12_RWSR	CH11_RWSR	CH10_RWSR	CH9_RWSR	CH8_RWSR	CH7_RWSR	CH6_RWSR	CH5_RWSR	CH4_RWSR	CH3_RWSR	CH2_RWSR	CH1_RWSR	CH0_RWSR
0	0	0	0	0	0	0	0	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	RW

Name	Bit	RW	Description	Reset Value
CHx_RWSR	[16:0]	RW	通道采样值读写状态位，当采样值写入TCH_SEQxVA时，对应通道读写状态位置1，软件读取TCH_SEQxVA后读写状态位清0。软件也可以通过写入该状态寄存器手动清0 当读取时，返回当前通道采样值写入状态 0h：该通道没有新的采样值写入 1h：该通道有新的采样值写入，软件可以读取对应通道采样值寄存器，读取后自动清0 当写入时， 0h：无效 1h：清除当前通道写入状态	0x0

19.3.13 TCH_OVW_THR (通道采样值OVW/通道采样值偏差状态寄存器)

- 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															CH16_O_THR	CH15_O_THR	CH14_O_THR	CH13_O_THR	CH12_O_THR	CH11_O_THR	CH10_O_THR	CH9_O_THR	CH8_O_THR	CH7_O_THR	CH6_O_THR	CH5_O_THR	CH4_O_THR	CH3_O_THR	CH2_O_THR	CH1_O_THR	CH0__O_THR
0	0	0	0	0	0	0	0	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	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC

Name	Bit	RW	Description	Reset Value
CHx_OVW_THR	[16:0]	RC	当TCH_CON0[11]=0时，该状态寄存器表示通道采样值OVW状态。 0h：采样值寄存器的值是最新的 1h: 采样值寄存器当前值未被软件读取时，有新的采样值存入 当CH_CON0[11]=1时，该状态寄存器表示通道采样值偏差状态寄存器 0h：通道相邻两次采样值偏差没有超出阈值 1h：通道相邻两次采样值偏差超出阈值	0x0
RC： Read Clear				

19.3.14 TCH_OVF (TCHCNT溢出状态寄存器)

- 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
RSVD															CH16_OVF	CH15_OVF	CH14_OVF	CH13_OVF	CH12_OVF	CH11_OVF	CH10_OVF	CH9_OVF	CH8_OVF	CH7_OVF	CH6_OVF	CH5_OVF	CH4_OVF	CH3_OVF	CH2_OVF	CH1_OVF	CH0_OVF
0	0	0	0	0	0	0	0	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_OVF	[16:0]	RC	TCH CNT溢出状态标志	0x0

19.3.15 TCH_OVT (TCH扫描超时状态寄存器)

- 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
RSVD															CH16_OVT	CH15_OVT	CH14_OVT	CH13_OVT	CH12_OVT	CH11_OVT	CH10_OVT	CH9_OVT	CH8_OVT	CH7_OVT	CH6_OVT	CH5_OVT	CH4_OVT	CH3_OVT	CH2_OVT	CH1_OVT	CH0_OVT
0	0	0	0	0	0	0	0	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_OVT	[16:0]	RC	通道扫描时间内，未完成既定目标值扫描状态标志	0x0

19.3.16 TCH_SYNCR (同步控制寄存器)

- 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
RSVD																							REARM	OSTMD	RSVD					SYNC2EN	SYNC1EN	SYNC0EN
0	0	0	0	0	0	0	0	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	R	R	R	R	RW	RW	RW

Name	Bit	RW	Description	Reset Value
SYNC0EN	[0]	RW	外部触发START使能控制通道0。 0：禁止当前触发输入通道 1：使能RTC触发源通道	0x0
SYNC1EN	[1]	RW	外部触发START使能控制通道1。 0：禁止当前触发输入通道 1：使能ETCB TOUCH_SYNCIN触发源通道	0x0
SYNC2EN	[2]	RW	外部触发START使能控制通道2。 0：禁止当前触发输入通道 1：使能LPT_PEND触发源通道	0x0
OSTMD	[7]	RW	一次性同步触发模式选择。 0h：连续触发模式 1h：一次性触发模式 当该输入通道被设置为一次性触发模式后，在一次触发事件被检测到后，该通道将不允许后续的触发事件通过，直到被软件重置（REARM）后才允许新的触发事件通过。	0x0
REARM	[8]	RW	在一次性同步触发模式下，软件重置当前通道状态控制位。 当读取时，返回当前触发使能状态 0h：允许触发 1h：不允许后续触发，需要REARM 当写入时， 0h：无效 1h：清除当前通道状态，并允许新的触发	0x0

19.3.17 TCH_EVTRG (事件触发选择寄存器)

- 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																							TRGOE	RSVD					TRGSEL		
0	0	0	0	0	0	0	0	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	R	R	RW	RW	RW

Name	Bit	RW	Description	Reset Value
TRGSEL	[2:0]	RW	TRGEVT事件的触发源选择 000：禁止TRGOUT触发输出 001：SINGLE_DONE事件 010：SEQ_DONE事件 011：OVR_THR事件 100：OVR_OVF事件 101：OVR_OVW事件 110：OVR_OVT事件	0x0
TRGOE	[8]	RW	外部触发端口输出使能 0h：禁止触发输出 1h：允许触发输出	0x0

19.3.18 TCH_EVPS (事件触发计数寄存器)

- 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
RSVD																				TRGEVCNT				RSVD				TRGEVPRD			
0	0	0	0	0	0	0	0	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	R	R	R	R	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
TRGEVPRD	[3:0]	RW	TRGEVT事件计数的周期设置。 当TRGEVT事件发生次数满足周期时，才产生TRGEV触发事件	0x0
TRGEVCNT	[11:8]	RW	TRGEVT事件计数器设置。 读取时，返回当前事件计数器值； 写入时，直接更新事件计数器值。	0x0

19.3.19 TCH_EVSWF (事件计数器软件触发控制寄存器)

- 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																															EVSWF
0	0	0	0	0	0	0	0	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
EVSWF	[0]	W	软件产生一次EV0的触发 0h：写入‘0’无效 1h：软件产生一次触发	0x0

19.3.20 TCH_CHxVAL (通道采样值寄存器)

- Address = Base Address + 0x1000 ~ Base Address + 0x1044, 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																CHx_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
R	R	R	R	R	R	R	R	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
CHxCNT	[15:0]	RW	通道x原值采样值寄存器	0x0000

19.3.21 TCH_SEQxCON (通道扫描配置寄存器)

- Address = Base Address + 0x1048 ~ Base Address + 0x108C, Reset Value = 0x0140_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						DCKDIV		RSVD	ICON			CHSEL				SCANTIME				SENPRD											
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	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	RW	Description	Reset Value
SENPRD	[11:0]	RW	当前扫描通道目标值，实际值为[SENPRD,4'b1111]	0x0000
SCANTIME	[14:12]	RW	当前扫描通道扫描时间值 3'b000-禁止通道扫描时间设置 3'b001-通道扫描时间为1ms 3'b010-通道扫描时间为1.5ms 3'b011-通道扫描时间为2ms 3'b100-通道扫描时间为3ms 3'b101-通道扫描时间为5ms 3'b110-通道扫描时间为10ms 3'b111-通道扫描时间为100ms	0x0
CHSEL	[19:15]	RW	扫描通道配置	0x00
ICON	[22:20]	RW	补偿电流控制	0x4
DCKDIV	[25:24]	RW	扫描频率分频	0x1