APT32F171 使用手册 V0.0

历史版本说明

版本	修改日期	修改概要
V0.0	2022-10-26	初版

1 系统存储空间

1.1 概述

本章节介绍了 APT32F171 系统存储空间管理。

本章包含内容如下：

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

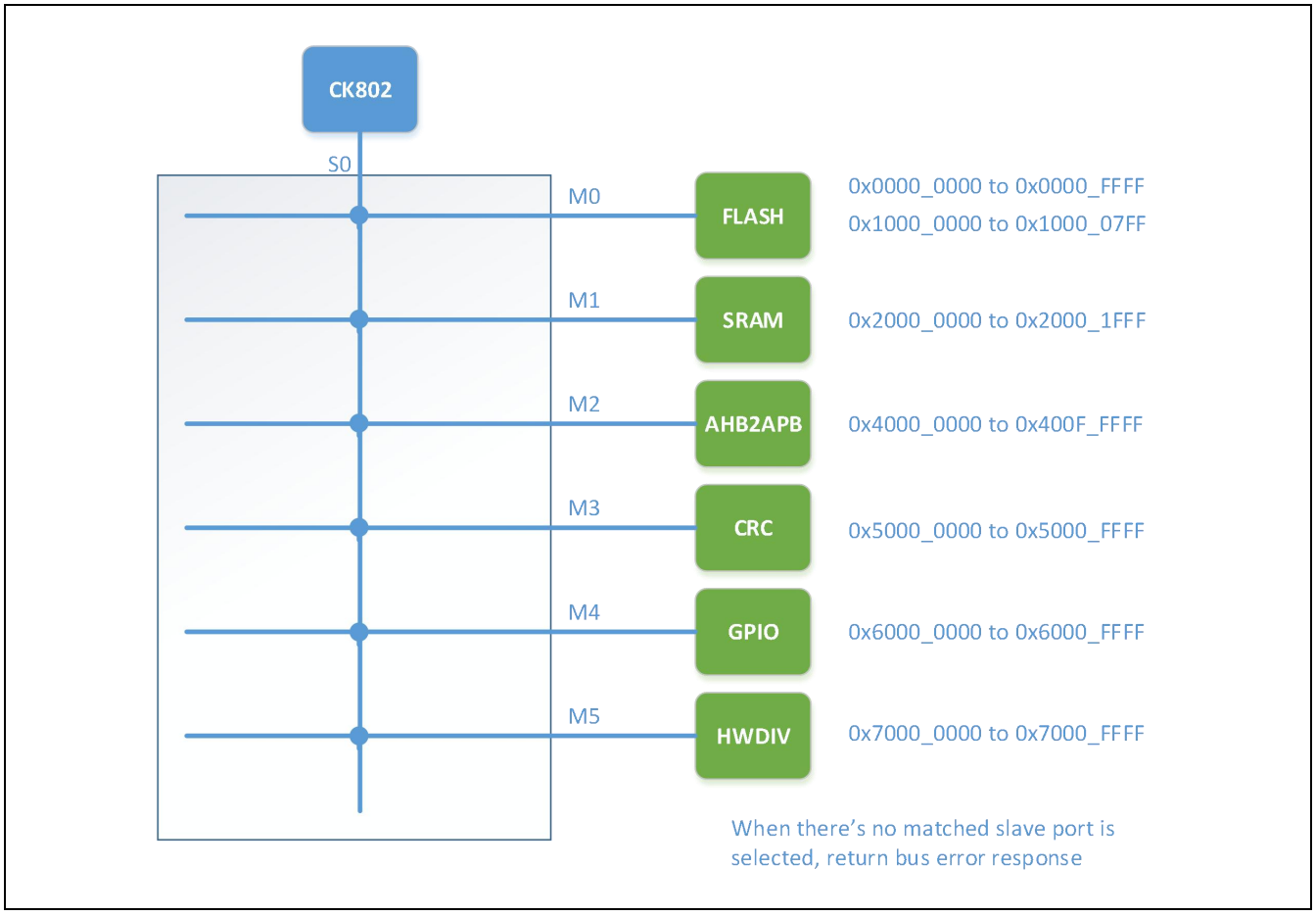


Figure 1-1 系统总线架构

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_1FFF	SRAM (8K)
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
HWDIV	0x7000_0000	硬件除法器 (HWDIV)
GPIO	0x6000_4000	通用IO端口-C (GPIO C)
	0x6000_2000	通用IO端口-B (GPIO B)
	0x6000_1000	通用IO端口-A1 (GPIO A1)
	0x6000_0000	通用IO端口-A0 (GPIO A0)
CRC	0x5000_0000	CRC控制器
OPA	0x400C_0000	运算放大器控制器 (OPA)
CMP	0x400B_0000	比较器控制器 (CMP)
-	0x400A_0000	保留 (Reserved)
	0x4009_0000	保留 (Reserved)
UART	0x4008_1000	通用异步收发器 (UART)
	0x4008_0000	通用异步收发器 (USART)
LP_TC	0x4007_2000	窗口型看门狗定时器 (WWDT)
TC	0x4006_9000	增强型可编程定时器/计数器 (EPT)
	0x4006_8000	保留 (Reserved)
	0x4006_7000	保留 (Reserved)
	0x4006_6000	保留 (Reserved)

	0x4006_5000	通用型可编程定时器/计数器 (GPT)
	0x4006_4000	基本定时器 (BT3)
	0x4006_3000	基本定时器 (BT2)
	0x4006_2000	基本定时器 (BT1)
	0x4006_1000	基本定时器 (BT0)
	0x4006_0000	保留 (Reserved)
	0x4005_4000	保留 (Reserved)
	0x4005_3000	保留 (Reserved)
	0x4005_2000	保留 (Reserved)
	0x4005_1000	保留 (Reserved)
	0x4005_0000	保留 (Reserved)
-	0x4004_0000	保留 (Reserved)
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)

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	CK802 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	-	Reserved
37/5	0x0000_0094	-	Reserved
38/6	0x0000_0098	-	Reserved
39/7	0x0000_009C	EXI_V0	External interrupt GROUP0, GROUP16
40/8	0x0000_00A0	EXI_V1	External interrupt GROUP1, GROUP17
41/9	0x0000_00A4	-	Reserved
42/10	0x0000_00A8	TC1	TC1 interrupt
43/11	0x0000_00AC	TC2	TC2 interrupt
44/12	0x0000_00B0	WWDT	Window Watchdog Interrupt
45/13	0x0000_00B4	USART	USART interrupt
46/14	0x0000_00B8	EPT	EPT Interrupt
47/15	0x0000_00BC	GPT	GPT Interrupt
48/16	0x0000_00C0	BT0	BT0 interrupt
49/17	0x0000_00C4	BT1	BT1 interrupt
50/18	0x0000_00C8	BT2	BT2 interrupt
51/19	0x0000_00CC	BT3	BT3 interrupt
52/20	0x0000_00D0	CMP5	CMP5 interrupt
53/21	0x0000_00D4	EXI_V2	External Interrupt GROUP2 ~ 3, GROUP18~19
54/22	0x0000_00D8	EXI_V3	External Interrupt GROUP4 ~ 9
55/23	0x0000_00DC	EXI_V4	External Interrupt GROUP10 ~ 15
56/24	0x0000_00E0	UART	UART interrupt
57/25	0x0000_00E4	-	Reserved
58/26	0x0000_00E8	-	Reserved
59/27	0x0000_00EC	CMP4	CMP4 interrupt
60/28	0x0000_00F0	CMP0	CMP0 interrupt
61/29	0x0000_00F4	CMP1	CMP1 interrupt
62/30	0x0000_00F8	CMP2	CMP2 interrupt
63/31	0x0000_00FC	CMP3	CMP3 interrupt

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

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

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

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

中断抢占的优先级条件可分为两种：

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

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

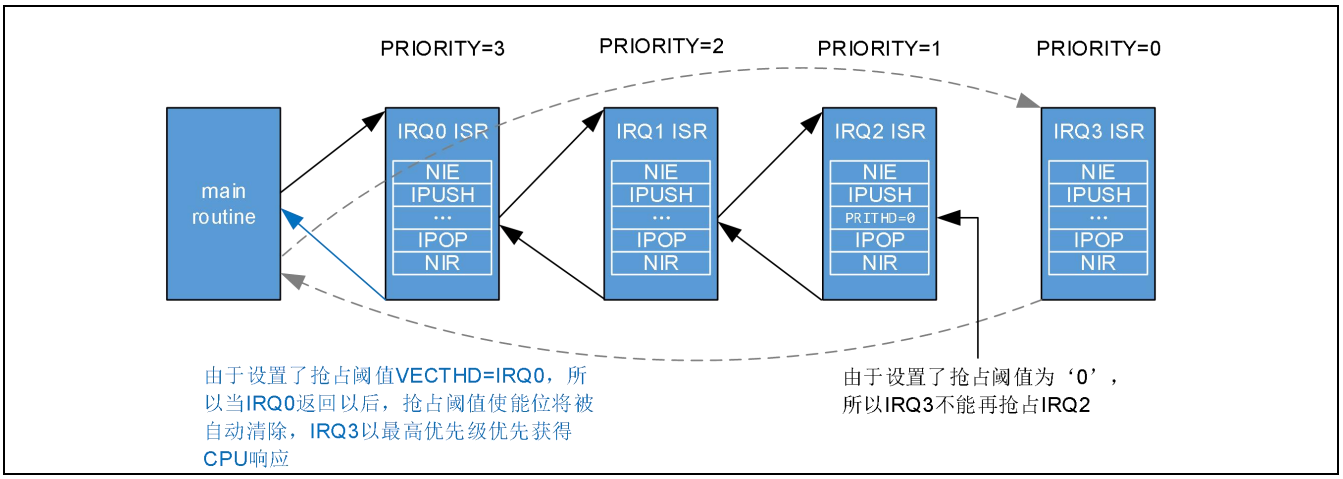


Figure 2-1 中断嵌套优先级示意图

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。

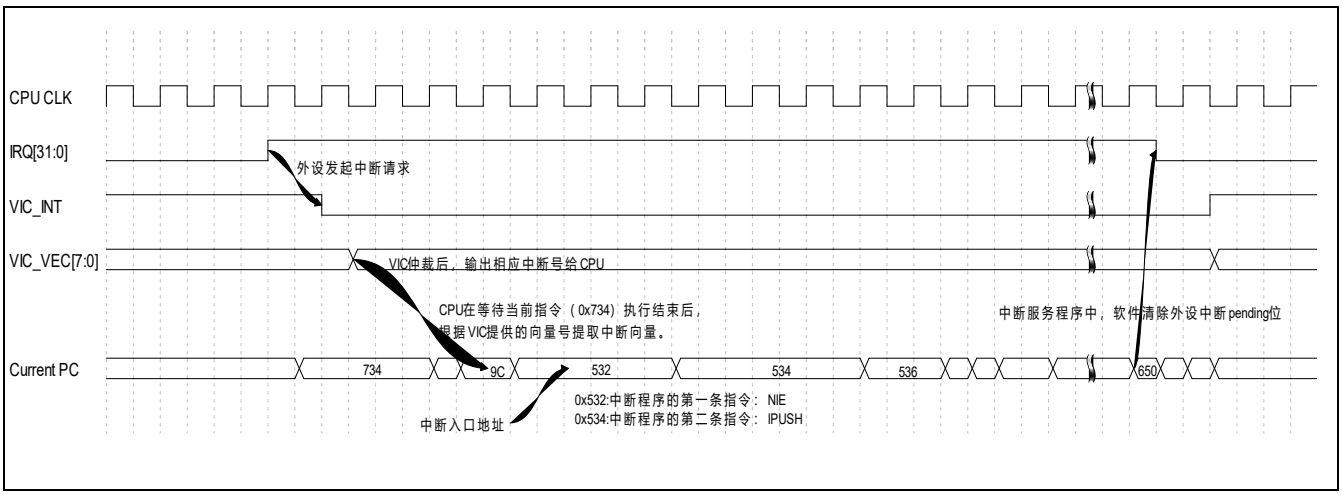


Figure 2-2 中断响应过程

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。

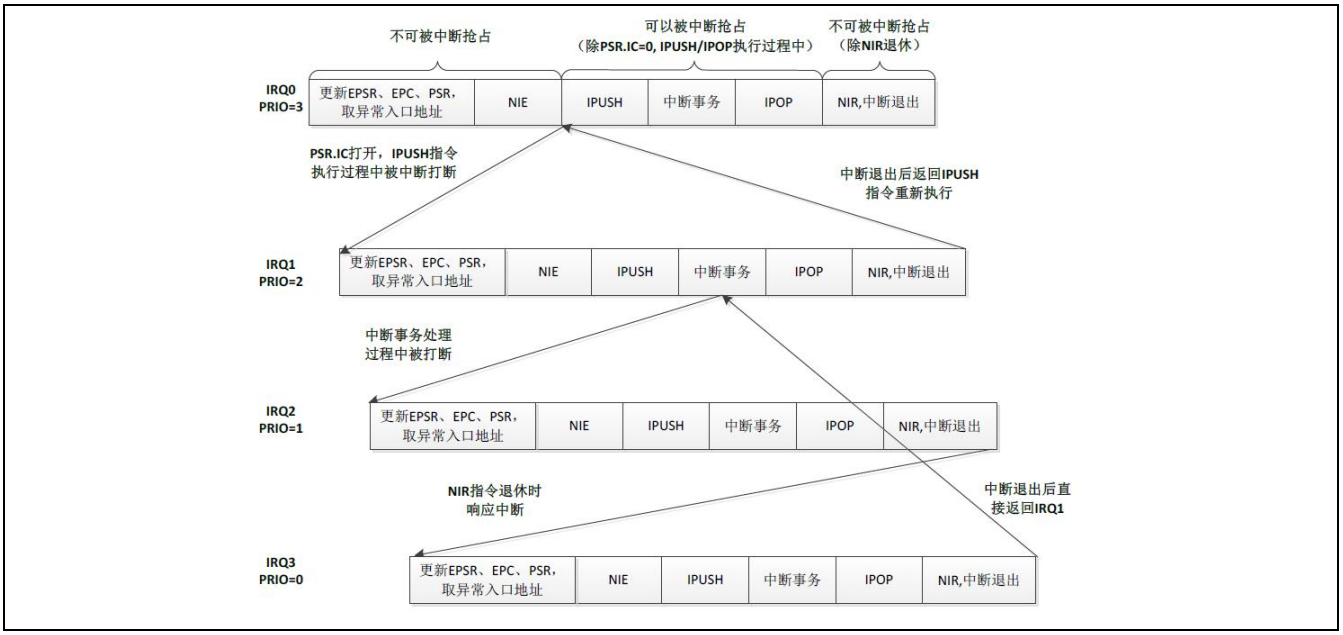


Figure 2-3 中断嵌套条件示例

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。

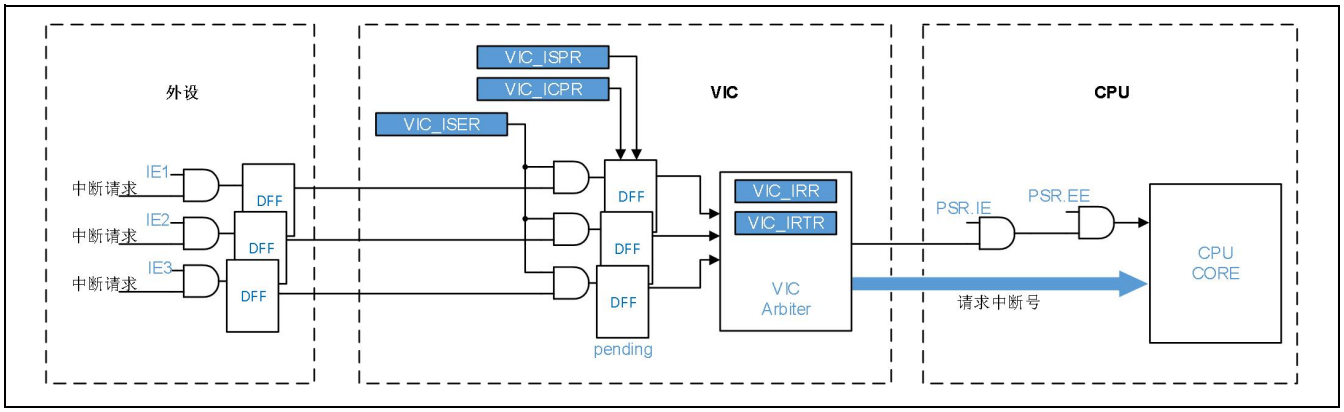


Figure 2-4 中断配置结构示意图

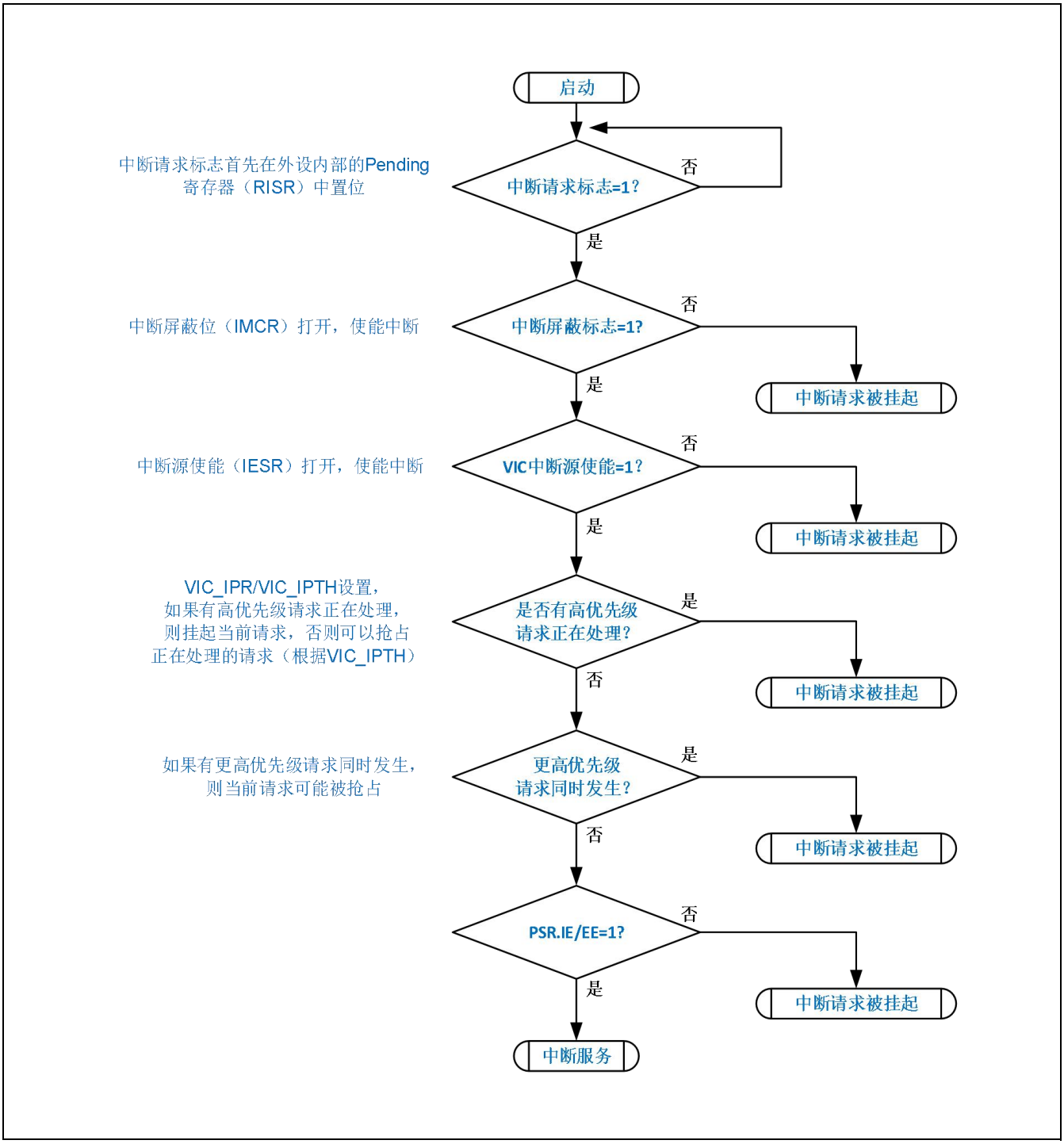


Figure 2- 5 中断请求处理流程

2.4 寄存器说明

2.4.1 寄存器表

- Base Address: 0xE000_E000

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

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

NOTE:

2.4.2 VIC_ISER (中断设置使能寄存器)

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

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

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

2.4.3 VIC_IWER (中断低功耗唤醒使能寄存器)

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

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

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

2.4.4 VIC_ICER (中断使能清除寄存器)

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

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

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

2.4.5 VIC_IWDR (中断低功耗唤醒清除寄存器)

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

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

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

2.4.6 VIC_ISPR (中断等待设置寄存器)

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

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

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

2.4.7 VIC_ICPR (清除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

2.4.8 VIC_IABR (中断响应状态寄存器)

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

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

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

2.4.9 VIC_IPR0 (中断优先级设置寄存器0)

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

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

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

2.4.10 VIC_IPR1 (中断优先级设置寄存器1)

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

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

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

2.4.11 VIC_IPR2 (中断优先级设置寄存器2)

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

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

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

2.4.12 VIC_IPR3 (中断优先级设置寄存器3)

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

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

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

2.4.13 VIC_IPR4 (中断优先级设置寄存器4)

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

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

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

2.4.14 VIC_IPR5 (中断优先级设置寄存器5)

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

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

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

2.4.15 VIC_IPR6 (中断优先级设置寄存器6)

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

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

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

2.4.16 VIC_IPR7 (中断优先级设置寄存器7)

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

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

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

2.4.17 VIC_ISR (中断状态寄存器)

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

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

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

2.4.18 VIC_IPTR (中断优先级阈值寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
THDEN	RSVD														VECTHD									PRITHD							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	R	R	R	R	R	R	R	R	R	R	R	R	R	R	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 CK802 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

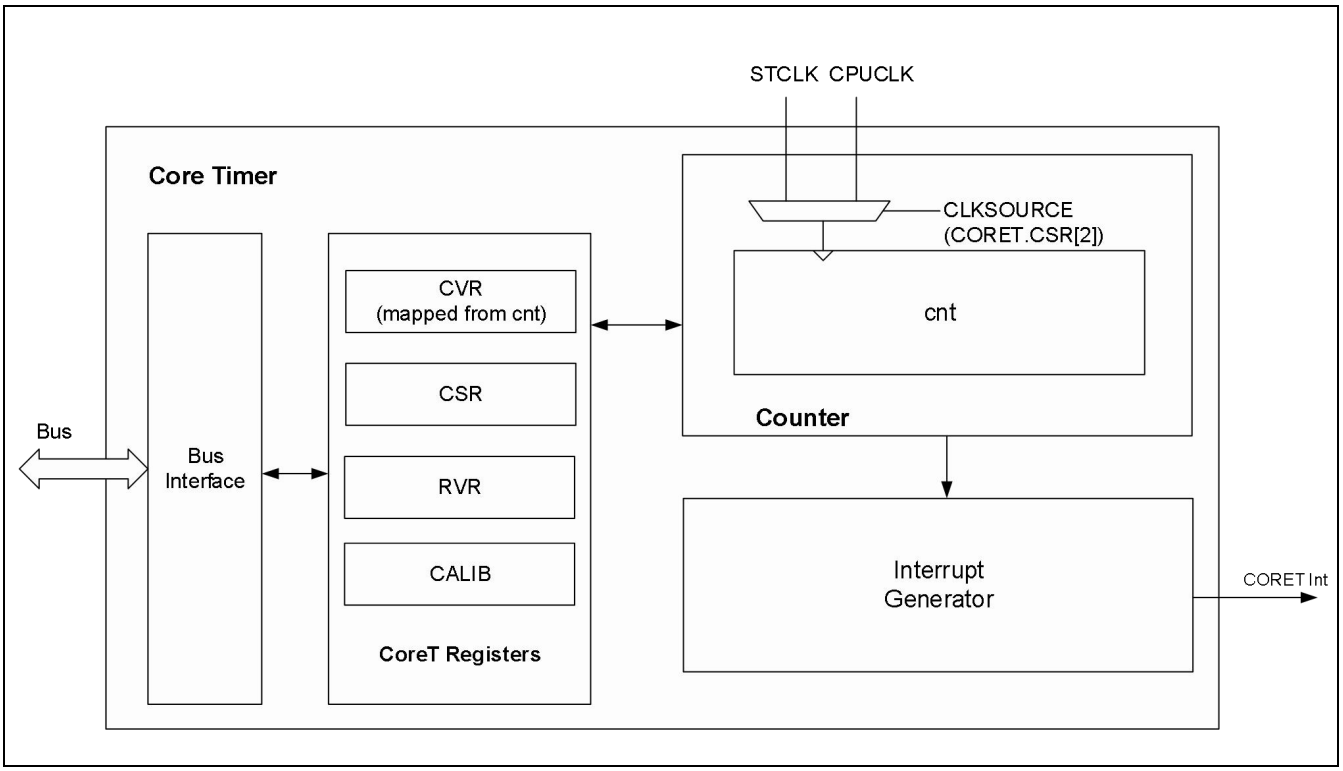


Figure 3- 1 CORET模块框图

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CK802 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 of CORET: 0xE000E000

Register	Offset	Description	Reset Value
CORET_CSR	0x10	控制寄存器	0x00000000
CORET_RVR	0x14	回填值寄存器	0x00000000
CORET_CVR	0x18	当前值寄存器	0x00000000

3.3.2 CORET_CSR(控制寄存器)

Address = Base Address+ 0x10, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	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
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。
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
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。
ENABLE	[0]	RW	定时器的使能控制： 0：禁止定时器 1：使能定时器
NOTE: CORECLK 是SYSTEM CLOCK经过分频后，供CPU工作使用的clock，和HCLK同频。			

3.3.3 CORET_RVR(回填值寄存器)

Address = Base Address+ 0x14, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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。

3.3.4 CORET_CVR(当前值寄存器)

Address = Base Address+ 0x18, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
CURRENT	[23:0]	RW	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。

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。

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。

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

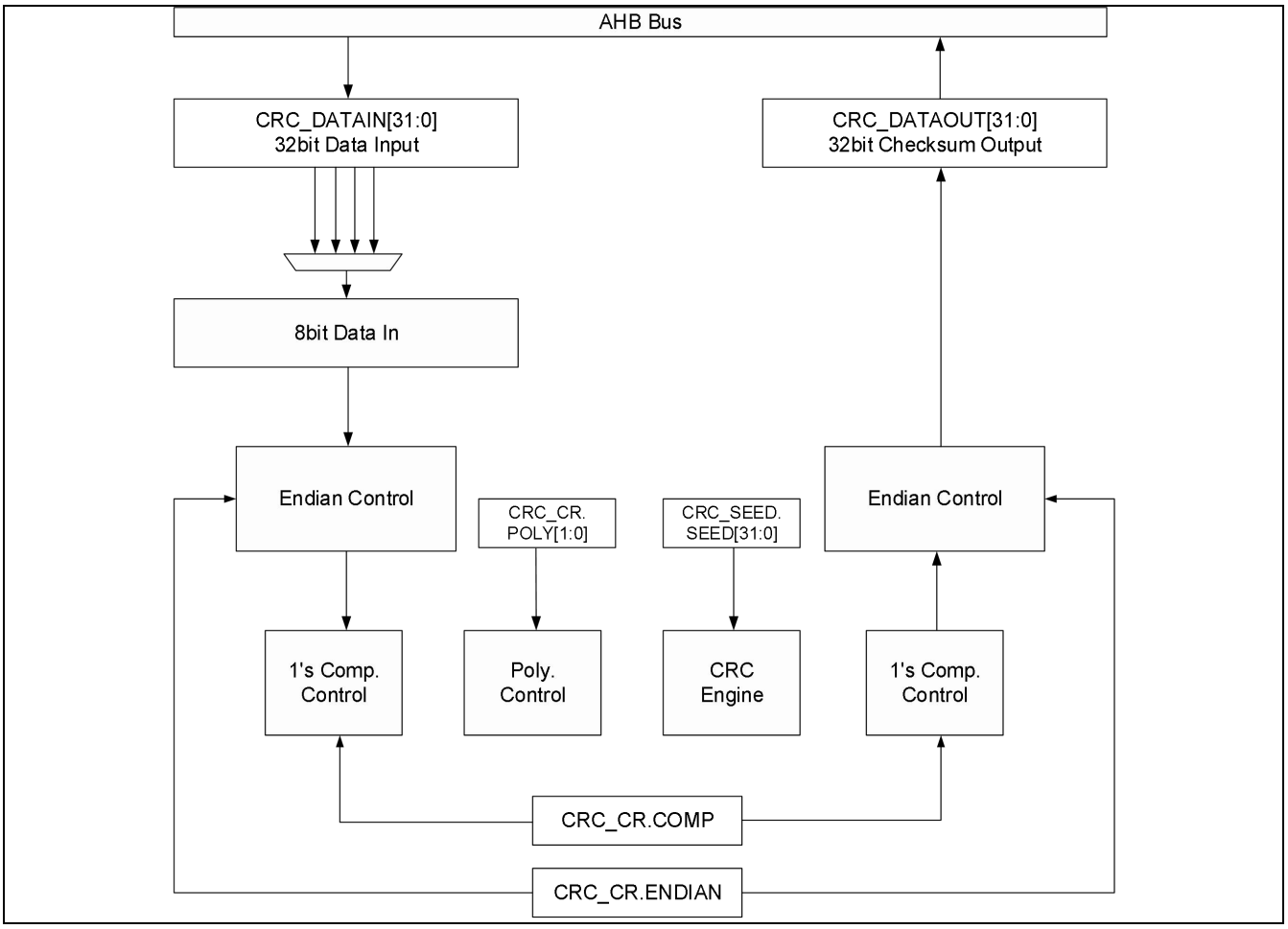


Figure 4-1 CRC模块框图

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。

4.3 寄存器说明

4.3.1 寄存器表

Base Address of CRC: 0x50000000

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

4.3.2 CRC_CEDR(时钟使能/禁止寄存器)

Address = Base Address+ 0x0004, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
CKEN	[0]	RW	CRC 引擎使能：0：禁止1：使能

4.3.3 CRC_SRR(软件复位寄存器)

Address = Base Address+ 0x0008, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
SWRST	[31]	W	软件复位写1会复位该模块

4.3.4 CRC_CR(控制寄存器)

Address = Base Address+ 0x000C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
POLY	[5:4]	RW	多项式控制 0x: CRC-CCITT 10: CRC-16 11: CRC-32
REFOUT	[3]	RW	CRC输出数据的按位反转控制0: 无反转1: 使能反转
REFIN	[2]	RW	CRC输入数据的按位反转控制0: 无反转1: 使能反转
XOROUT	[1]	RW	CRC输出数据的异或控制0: 无异或1: 使能异或
XORIN	[0]	RW	CRC输入数据的异或控制0: 无异或1: 使能异或

4.3.5 CRC_SEED(种子值寄存器)

Address = Base Address+ 0x0010, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
SEED	[31:0]	RW	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。

4.3.6 CRC_DATAIN(输入数据寄存器)

Address = Base Address+ 0x0014, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
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。

4.3.7 CRC_DATAOUT(输出数据寄存器)

Address = Base Address+ 0x0018, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
DATAOUT	[31:0]	R	该寄存器保存CRC计算结果。

5 硬件除法器 (HWDIV)

5.1 概述

硬件除法器可以快速执行除法操作，支持有符号和无符号的 32 位整形除法运算。

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。

5.1.1 主要特性

- 支持有符号和无符号运算
- 32 位除法运算，支持 32 位宽度的被除数和除数输入，输出 32 位商和余数
  - 无符号数范围：0 ~ 4294967295
  - 有符号数范围：-2147483648 ~ 2147483647
- 流水线结构除法器，5 个 HCLK 周期完成一次除法运算
- 支持除数为零错误中断 (该中断在 SYSCON 模块中，参见 SYSCON 中断相关寄存器， HWD_ERR 中断)

5.1.2 基本功能描述

硬件除法器可以自动运算当前结果，并在结果完成后自动输出。对除法器的被除数[DIVIDEND]或除数[DIVISOR]寄存器进行写入后，硬件除法器会自动触发硬件运算，并将结果输出到商[QUOTIENT]和余数[REMAINDER]寄存器中。不需要软件进行额外的启动或者查询操作。当软件发起读取商或余数寄存器操作时，若运算正在进行中，则系统会自动挂起当前的读取操作，直到结果返回。

硬件除法器支持有符号和无符号除法，在进行运算前，通过CR寄存器的UNSIGN控制位进行配置。当除数为零时，会产生系统中断（该中断标志位在SYSCON的RISR寄存器中）。在当前运算未结束前，更新被除数或除数寄存器，则当前运算被自动终止并开始新的运算，并在5个HCLK后输出结果。

5.2 寄存器说明

5.2.1 寄存器表

Base Address of HWDIV: 0x70000000

Register	Offset	Description	Reset Value
HWD_DIVIDEND	0x0000	被除数寄存器	0x00000000
HWD_DIVISOR	0x0004	除数寄存器	0x00000001
HWD_QUOTIENT	0x0008	商寄存器	0x00000000
HWD_REMAINDER	0x000C	余数寄存器	0x00000000
HWD_CR	0x0010	控制寄存器	0x00000000

5.2.2 HWD_DIVIDEND(被除数寄存器)

Address = Base Address+ 0x0000, Reset Value = 0x00000000

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

5.2.3 HWD_DIVISOR(除数寄存器)

Address = Base Address+ 0x0004, Reset Value = 0x00000001

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

Name	Bit	Type	Description
DIVISOR	[31:0]	RW	除数寄存器

5.2.4 HWD_QUOTIENT(商寄存器)

Address = Base Address+ 0x0008, Reset Value = 0x00000000

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

5.2.5 HWD_REMAINDER(余数寄存器)

Address = Base Address+ 0x000C, Reset Value = 0x00000000

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

5.2.6 HWD_CR(控制寄存器)

Address = Base Address+ 0x0010, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																															UNSIGN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
UNSIGN	[0]	RW	运算符号位控制寄存器 0h：有符号除法运算 1h：无符号除法运算

6 闪存控制器(IFC)

6.1 概述

本章节描述用来控制内部程序存储的闪存控制器。APT32F171 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。APT32F171 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。

6.1.1 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

6.2 功能描述

6.2.1 模块框图

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

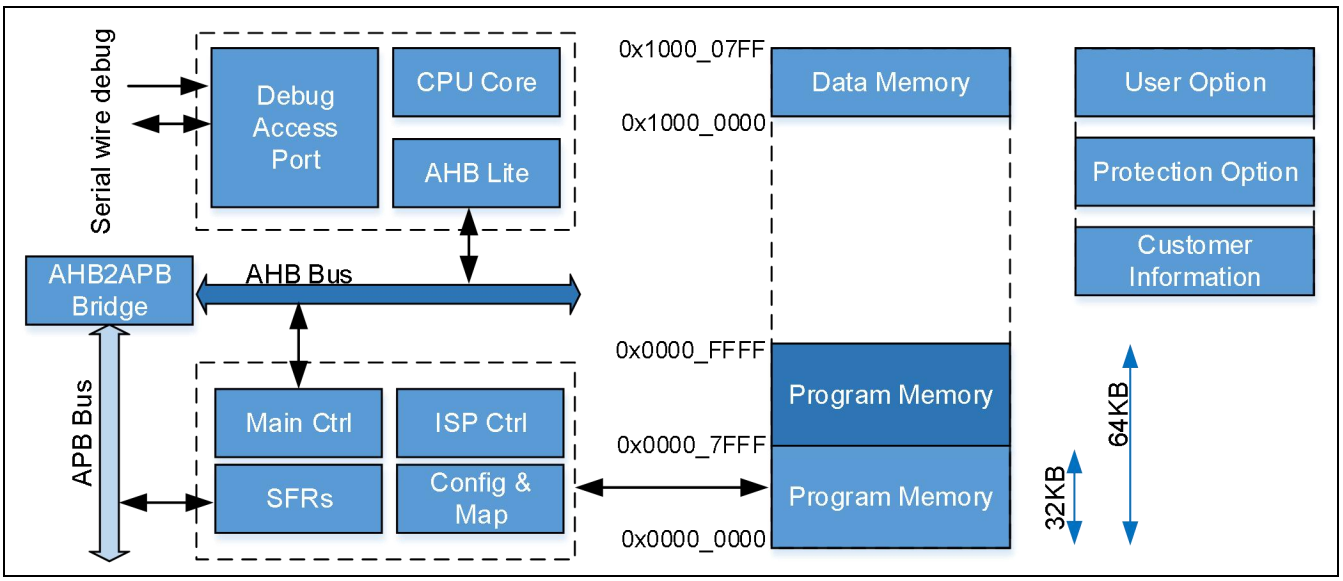


Figure 6-1 IFC模块框图

6.2.2 模块结构

APT32F171 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：

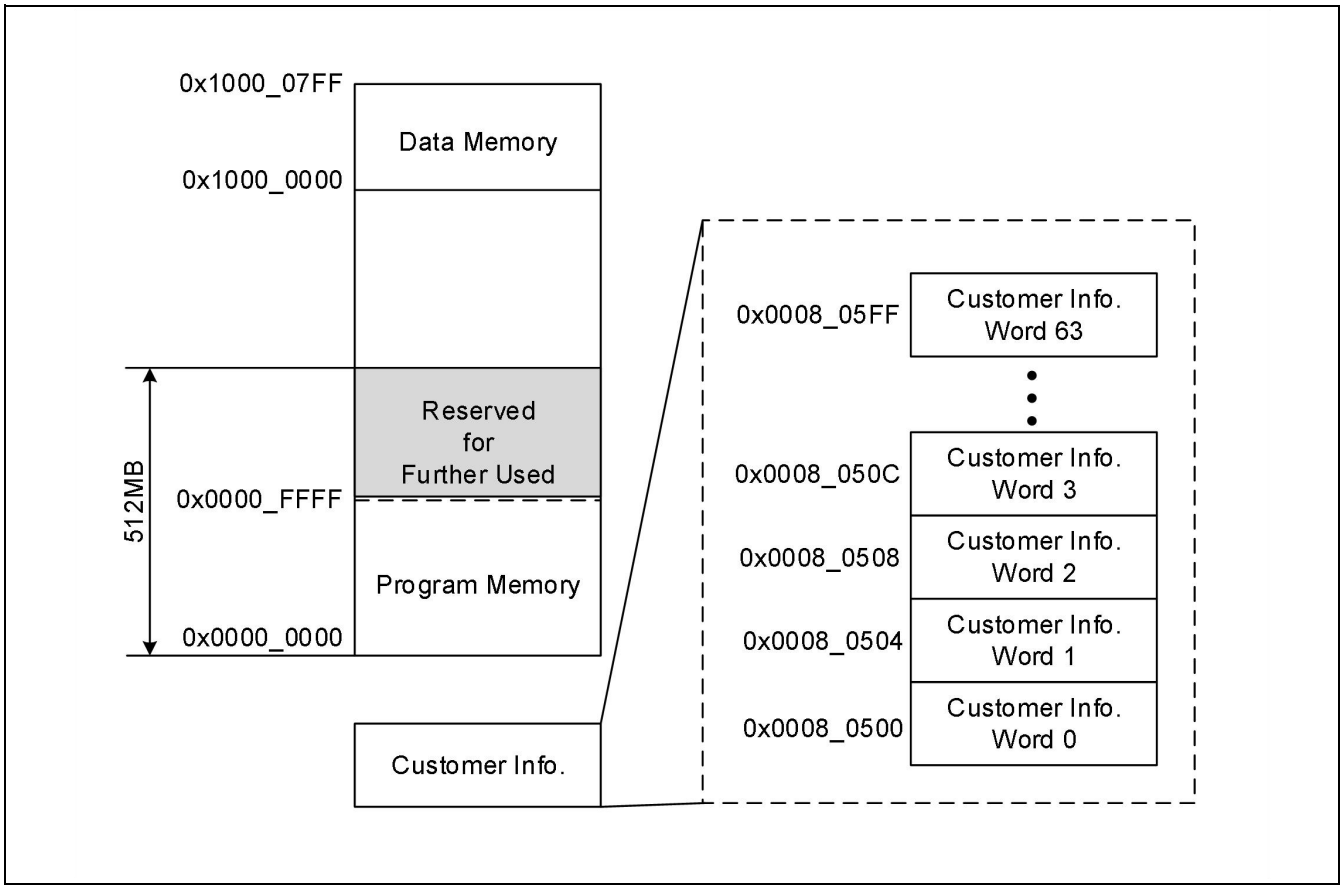


Figure 6-2 闪存地址空间结构

6.2.3 数据闪存

APT32F171 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。

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

6.2.4.1 User Option

User Option 用来配置各种不同应用所需的功能，该功能需要配合专用的烧录器使用。

Table 6-2 用户选项功能说明

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

名称	位	描述
EXTRST	[3:0]	外部复位管脚功能 **EXTRST[3:0]** / **功能** 0x5 / PB0.3/PA0.2(1)为外部复位管脚，IO功能被禁用 其它值 / PB0.3/PA0.2(1)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

NOTE: 不同的封装对应不同的外部复位管脚， 参看数据手册的管脚配置章节。

6.2.4.2 保护选项 (Protection)

保护选项是为了保护代码的安全。闪存控制器支持四种不同的保护机制，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，那么程序代码仍然可以通过 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，非正常渠道的破解读取时，读到的代码都为加密代码。

  注意：如果只是开启代码加密保护，程序仍然可以通过 SWD 口读出。如果希望彻底实现加密功能，需要同时使能 SWD 保护。另外，在 APT32F171 系列的实现中，SYSCON_CLCR 寄存器参与到加密中，所以如果开启代码加密保护，时钟微调功能将不能使用。

- 四种保护功能的小结：

	使能方式	SWD 读取	烧片机读取	CPU 读取/CPU 写入
HDP	ISP/PGM tool	可以	烧片机不支持读取操作	可以/被保护区不可以
RDP	ISP/PGM tool	可以		可以/可以
SWDP	ISP/PGM tool	不可以		可以/可以
ENCRYPT	PGM tool	烧入代码的区域可以读取； 未烧入代码的区域读回某个固定值（每个芯片不一样）(1)		可以/可以

NOTE(1) 这可以用来检查加密功能是否起效。

6.2.4.3 客户信息区 (Customer Information)

参考 Figure 6-3，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。

6.2.4.4 UID (Unique ID)

UID 为该芯片的标识，每个芯片有单独唯一的 UID。

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。该区域能通过 SYSCON 模块中相应的镜像寄存器 UID0~UID2 进行读取。

6.2.5 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。

Table 6-3 自定义选项的设置方法

	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	√	X	√
客户信息区	√	X	X
UID	√	X	X

6.2.6 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 寄存器中的描述。

6.2.7 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 (通过AHB接口，即ISP模式)
- SWD接口

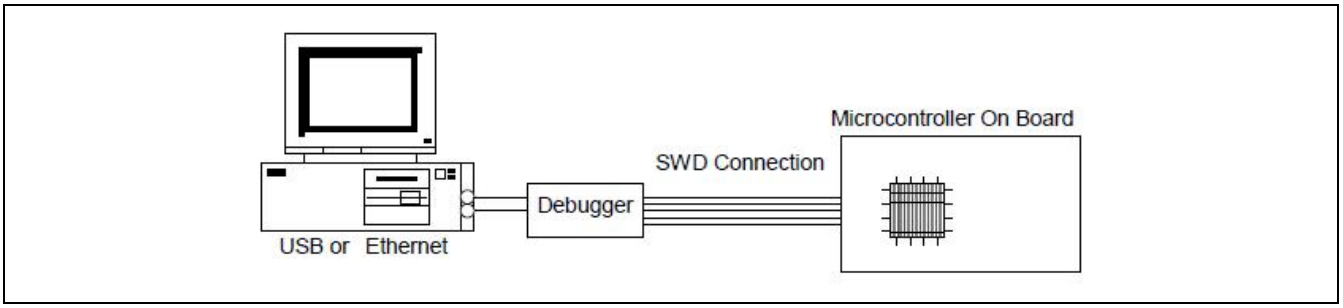


Figure 6-4 通过调试接口的烧写

- 烧录工具 (专用串行接口)

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。

Table 6-4 闪存烧写信号

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

6.2.8 ISP

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。闪存 ISP 功能通过 IFC 中的一些控制寄存器来实现。

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。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。

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。

6.2.8.1 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。在页擦除操作完成后，IFC_RISR 里的 ERS_END 位会置 1。用户同时也可以查询 IFC_CR 里的 START 位来判断页擦除操作是否完成。

示例：

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

6.2.8.2 写闪存操作 - 普通模式

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（Page Latch），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。

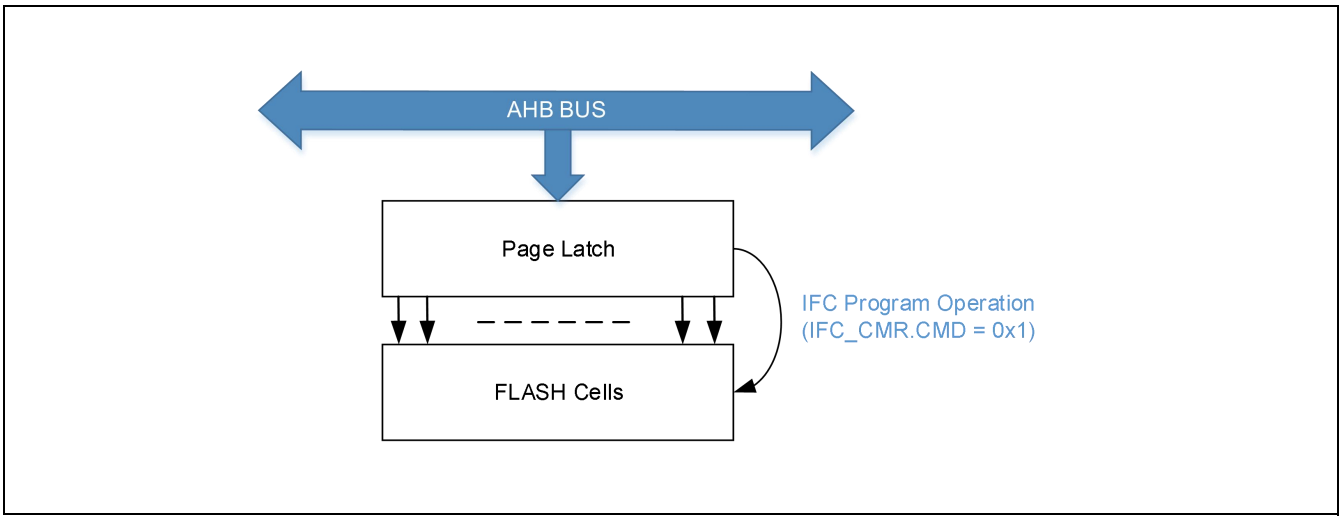


Figure 6-5 将页缓存数据写入闪存中

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。

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。

6.2.8.3 写闪存操作 - 并行模式

在普通模式中，开始写闪存操作后，CPU会一直处于等待状态，不能执行下一条指令，这时候的等待时间会影响CPU的执行效率。本产品中，预编程的时间约为200uS，擦除时间约为2.5mS，写操作时间约为1.5mS。由于每写一次闪存，都要经过预编程 - 擦除 - 写的过程，所以如果频繁对闪存进行写操作的话，每次都要等待至少4.2mS(200uS+1.5mS+2.5mS)，对CPU的执行效率产生较大影响。

并行模式则可以避免这个问题。并行模式使能(IFC_MR寄存器的PMODE位，置1)后，CPU在写操作的同时，仍然能够读取并且执行指令，而不需要等待4.2mS的闪存操作时间。但是，并行模式只对数据闪存(DROM)的擦除或者写操作有效，也就是只有在写数据闪存时，才可以使能并行模式，并且让CPU读取和执行程序闪存(PROM)内的代码。

并行模式的操作过程跟普通模式一样，只是在执行第4/5/6步的启动(START)时，CPU不会停止在该步骤，而是会继续执行下面的指令，用户需要判断IFC_RISR寄存器中的标志位来判断预编程/擦除/写操作是否完成，或者通过中断来得知操作是否完成。建议用户通过中断程序来完成整个写操作的过程，将第5和第6步放在中断子程序中。具体可以参考该产品的SDK。

6.2.8.4 片擦除操作

片擦除操作会擦除整个闪存的程序存储，自定义选项区，客户信息区，但不会擦除数据存储区域。片擦除操作只能在用户特权模式下才能执行。在片擦除中，不需要指定 ISP 操作相关的地址和数据寄存器。在 ISP 操作前，用户必须将秘钥 0x5A5A_5A5A 写入 IFC_KEY 寄存器以禁止闪存模块的擦除/烧写保护。之后，将 IFC_CMR 里的指令 CMD[3:0]写为 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。当然因为片擦除后 PROM 将没有内容，只有 SRAM 中运行的代码可以查询 Flash 状态。

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

6.2.8.5 擦除自定义选项区域

这个自定义选项擦除操作会擦除所有的 User Option，保护选项和客户信息区域。在 ISP 操作前，用户必须将秘钥 0x5A5A_5A5A 写入 IFC_KEY 寄存器以禁止闪存模块的擦除/烧写保护。之后，将 IFC_CMR 里的指令 CMD[3:0]写为 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
```

6.2.8.6 烧写自定义选项操作

共有 5 种自定义的选项支持通过 ISP 操作，分贝是 HDP, RDP, SWDP, SWD_REMAP 和 USER_OPTION。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。

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]写入对应指令，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。

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
```

6.2.9 闪存控制器的中断

闪存操作有 7 个中断源，如下所示。

Table 6- 5 中断源描述

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

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

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，而且该中断事件发生了(RISR 相应位置 1)，那么该中断会被送至 CPU 处理，进入中断处理程序。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。

6.2.10 闪存控制流程图

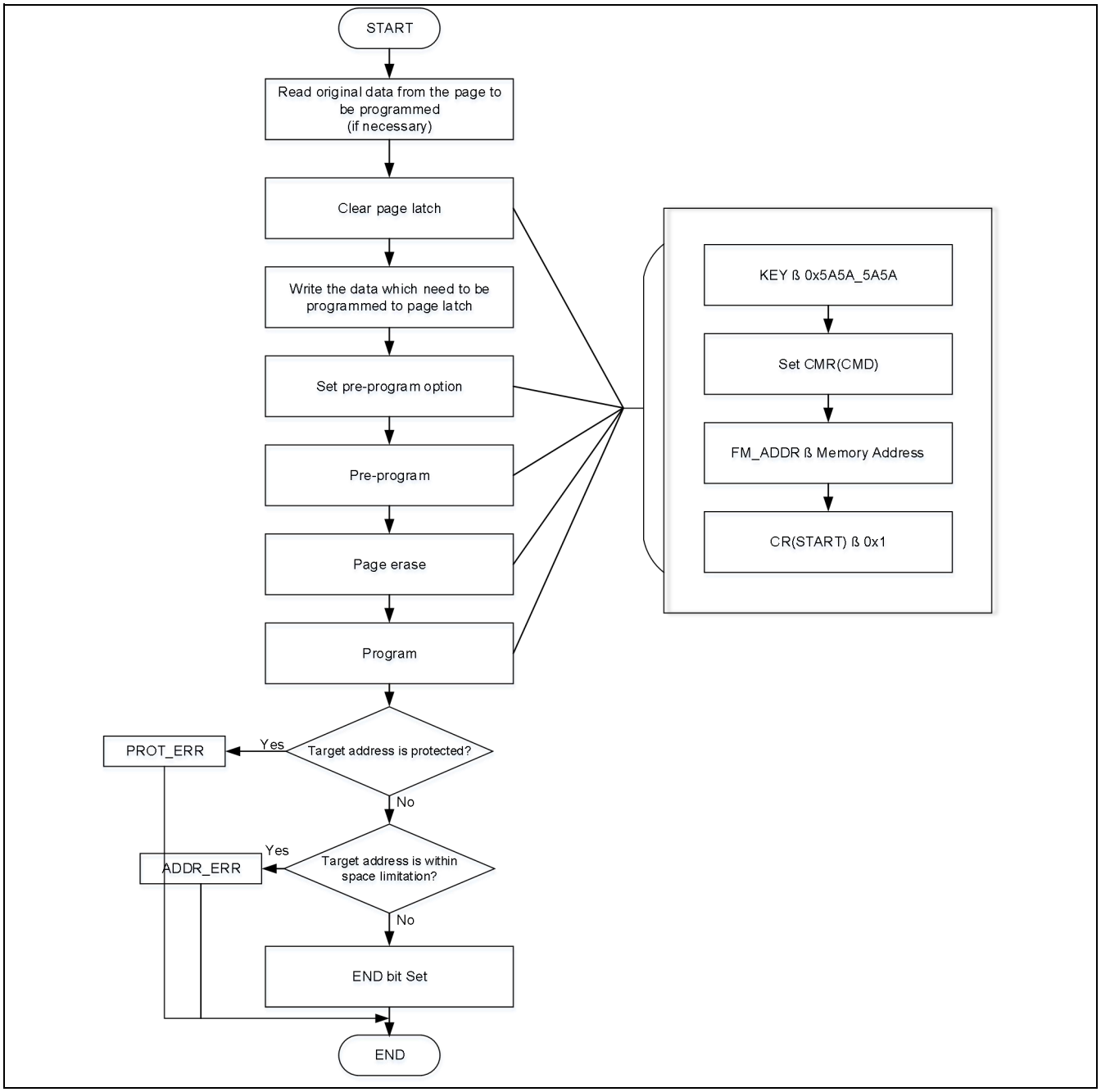


Figure 6-7 Flash Write – Normal Mode

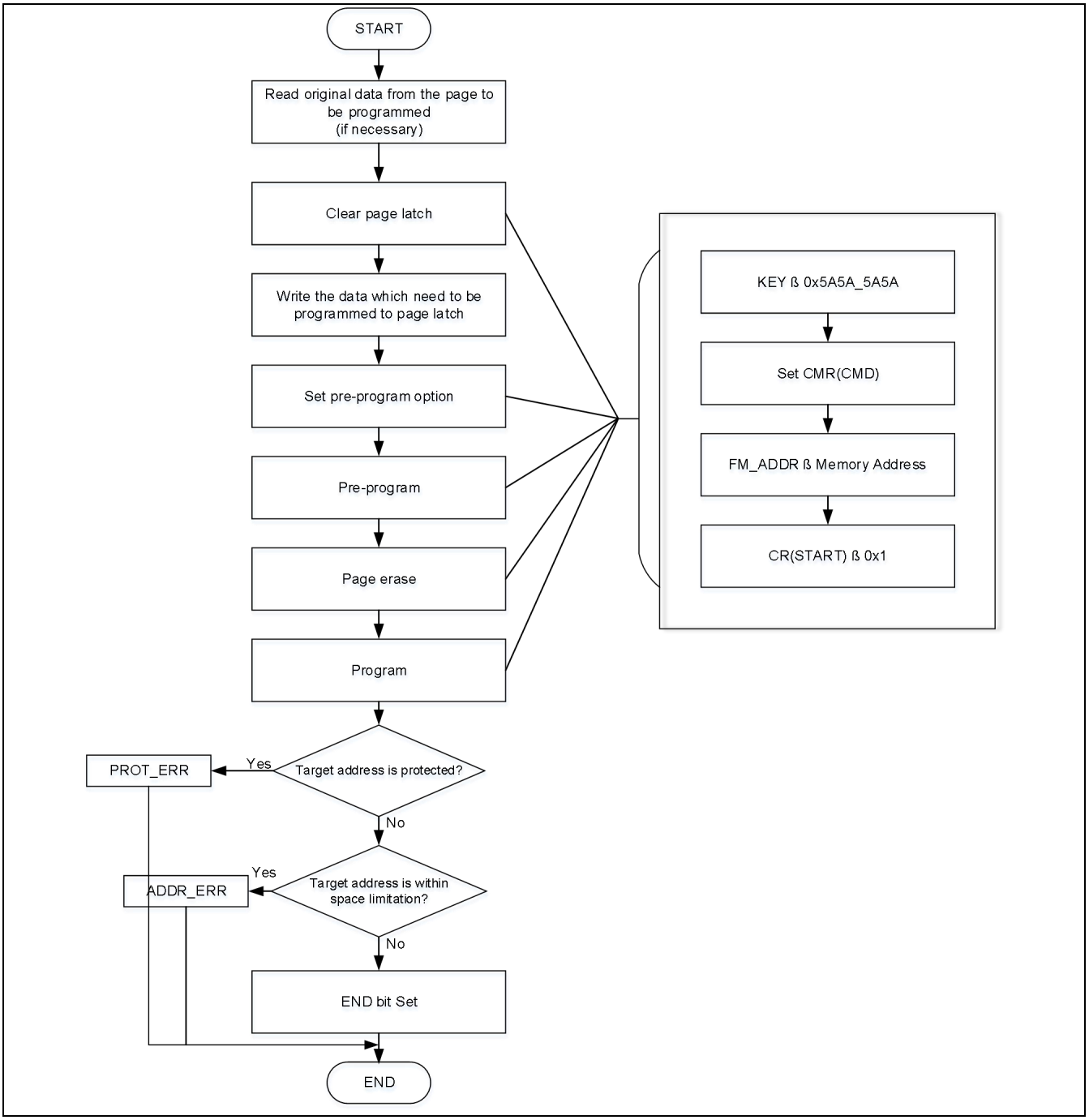


Figure 6- 8 Flash Write – Parallel Mode

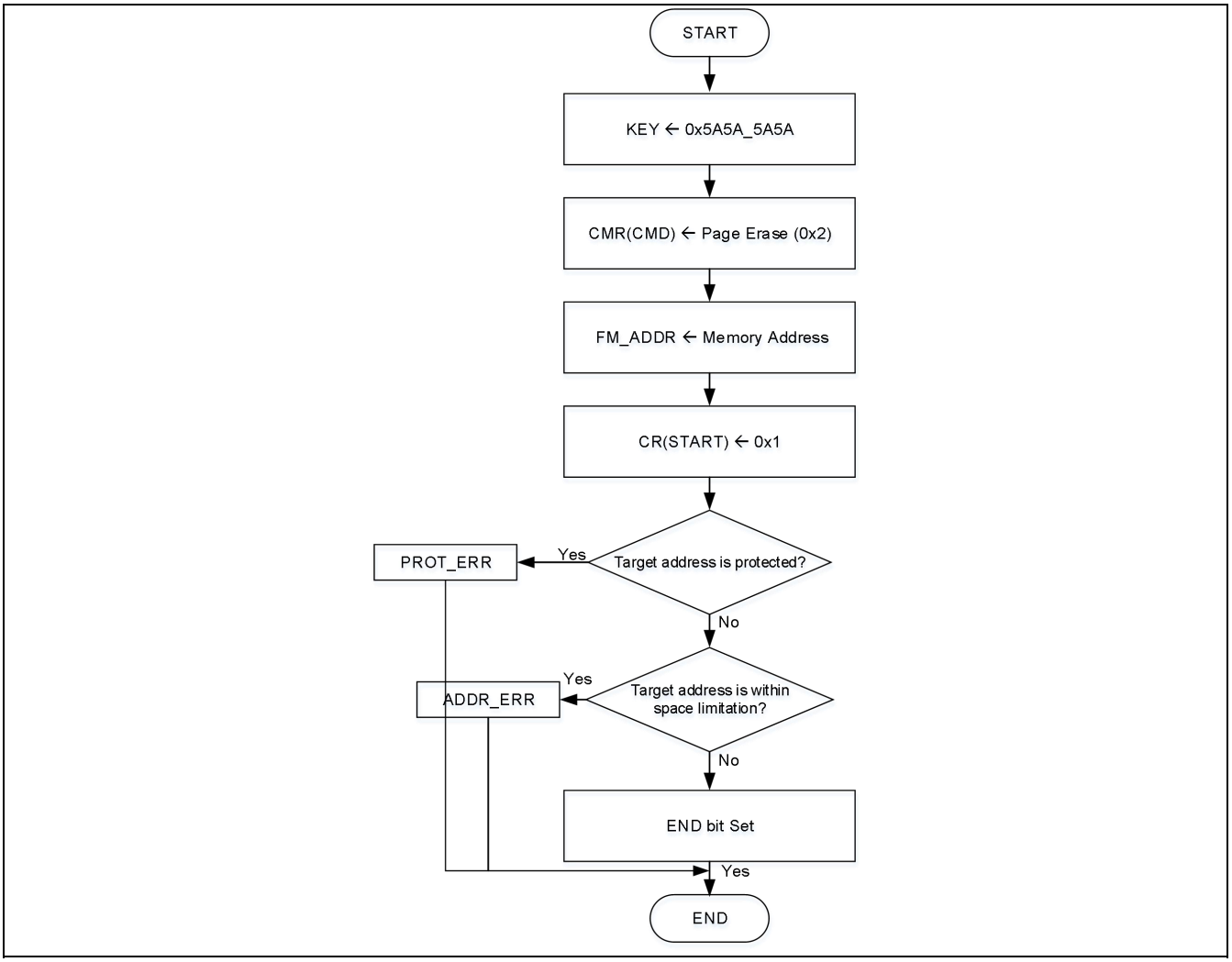


Figure 6-9 Page Erase

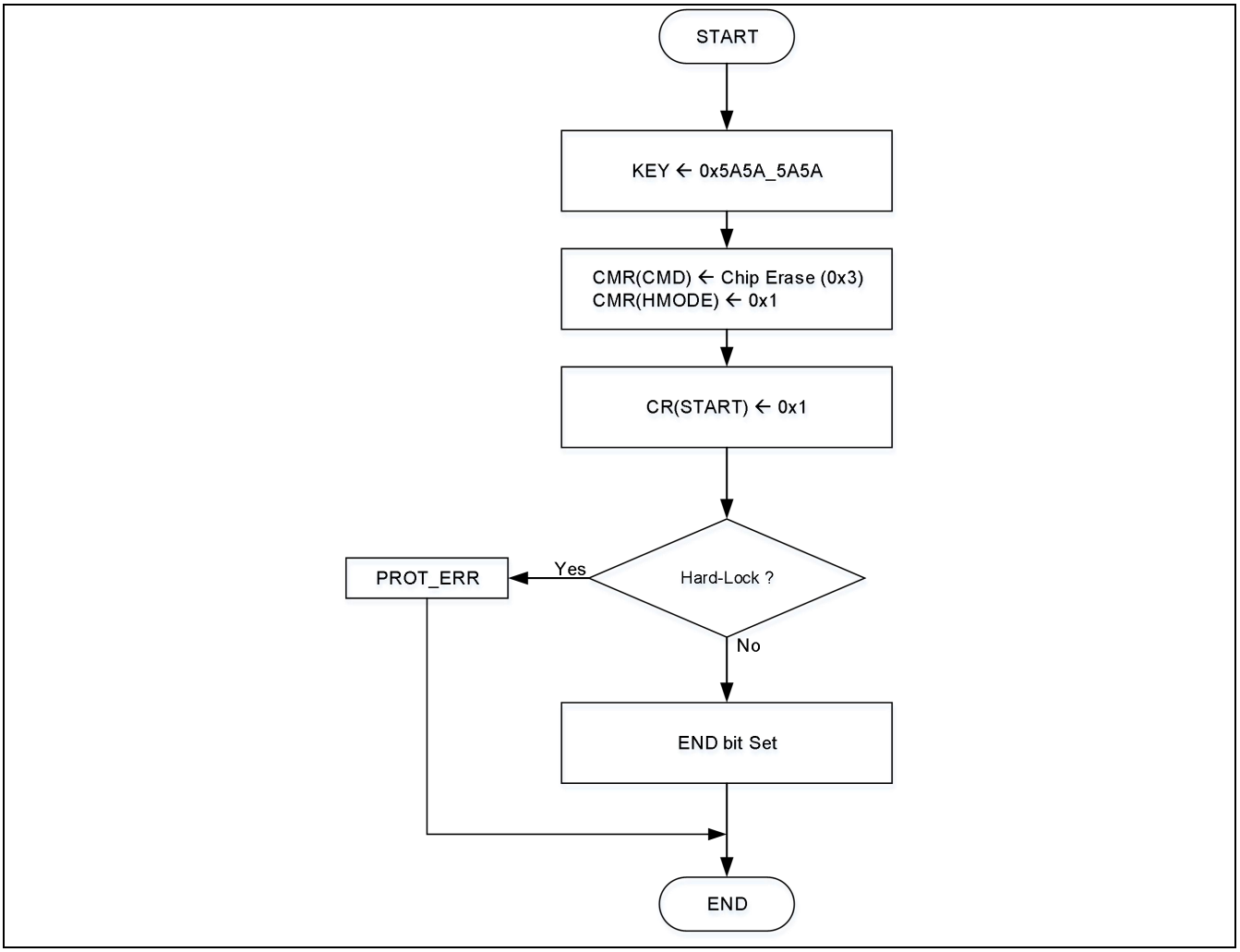


Figure 6- 10 Chip Erase

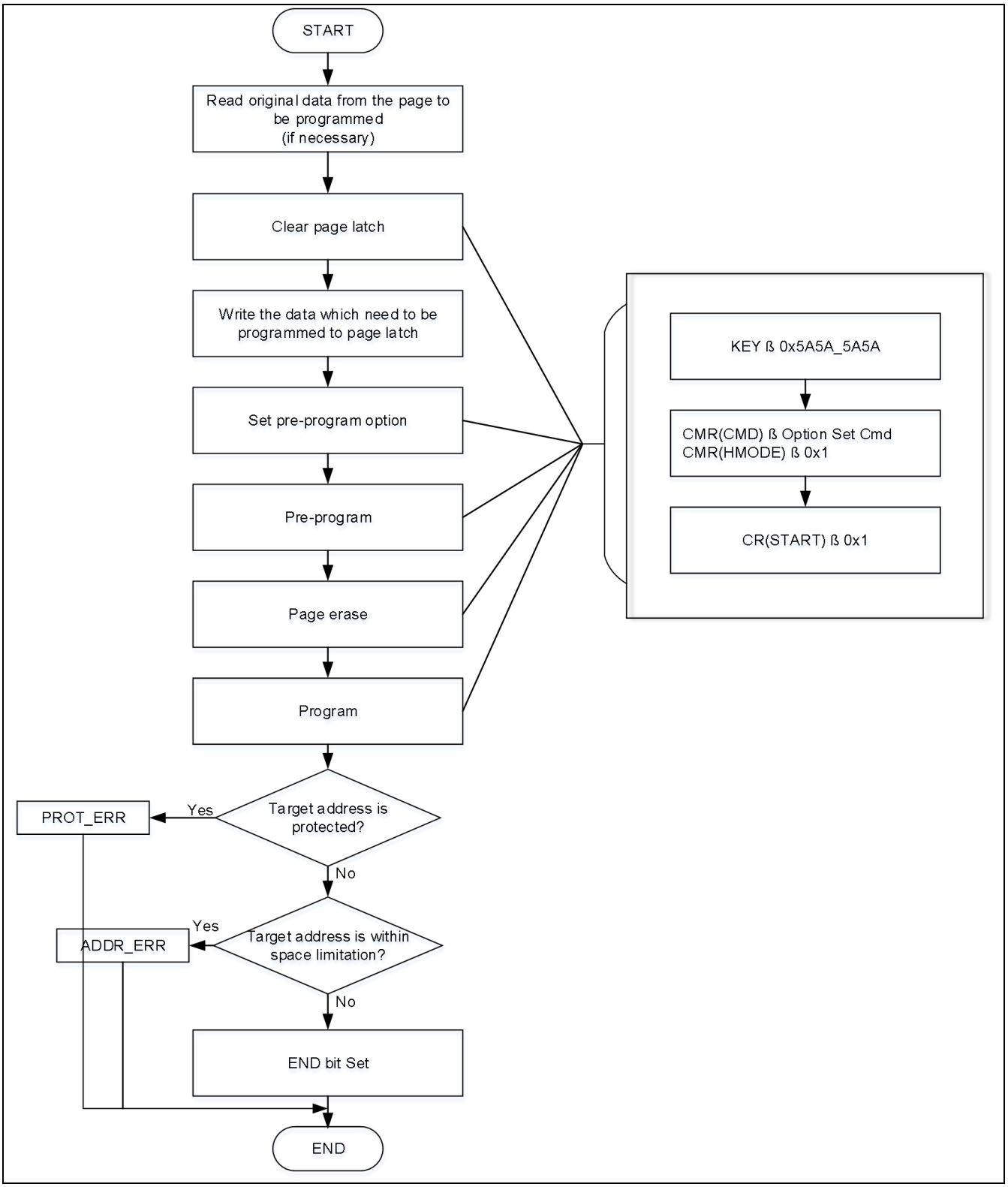


Figure 6- 11 Option Cells Write

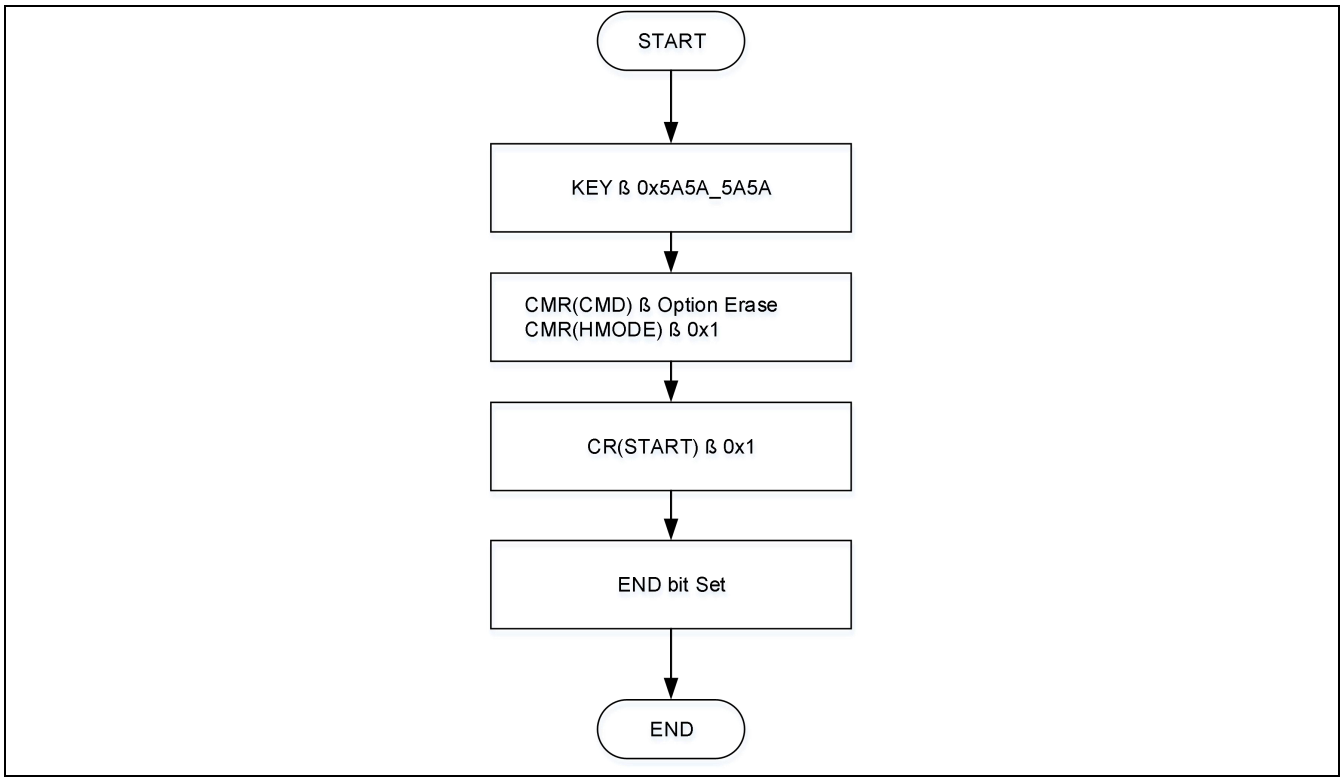


Figure 6- 12 Option Cells Erase

6.3 寄存器说明

6.3.1 寄存器表

Base Address of IFC: 0x40010000

Register	Offset	Description	Reset Value
IFC_IDR	0x00	闪存控制器ID寄存器	0x00000170
IFC_CEDR	0x04	时钟使能/禁止寄存器	0x00000000
IFC_SRR	0x08	软件复位寄存器	0x00000000
IFC_CMR	0x0C	指令寄存器	0x00000000
IFC_CR	0x10	控制寄存器	0x00000000
IFC_MR	0x14	工作模式寄存器	0x00000000
IFC_FM_ADDR	0x18	ISP地址寄存器	0x00000000
IFC_KR	0x20	ISP秘钥寄存器	0x00000000
IFC_IMCR	0x24	中断控制寄存器	0x00000000
IFC_RISR	0x28	中断原始状态寄存器	0x00000000
IFC_MISR	0x2C	中断状态寄存器	0x00000000
IFC_ICR	0x30	中断状态清除寄存器	0x00000000

6.3.2 IFC_IDR(闪存控制器 ID 寄存器)

Address = Base Address+ 0x00, Reset Value = 0x00000170

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								IDCODE																							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	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
IDCODE	[23:0]	RW	ID代码 (0x170)

6.3.3 IFC_CEDR(时钟使能/禁止寄存器)

Address = Base Address+ 0x04, Reset Value = 0x00000000

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

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

6.3.4 IFC_SRR (软件复位寄存器)

Address = Base Address+ 0x08, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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 复位除CEDR外的所有寄存器。

6.3.5 IFC_CMR(指令寄存器)

Address = Base Address+ 0x0C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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。
HMODE	[9:8]	RW	操作模式寄存器 00: 普通模式 01: 用户特权模式 others: 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。
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 （仅当HMODE=1时有效） 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（仅当HMODE=1时有效） 0xE 使能调试口重映射（仅当HMODE=1时有效） 0xF 写User Option操作（仅当HMODE=1时有效）

			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，指令不会被执行

6.3.6 IFC_CR(控制寄存器)

Address = Base Address+ 0x10, Reset Value = 0x00000000

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

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

6.3.7 IFC_MR(工作模式寄存器)

Address = Base Address+ 0x14, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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							PMODE	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	RW	R	R	R	R	R	RW	RW	RW

Name	Bit	Type	Description
SPEED	[16]	RW	FLASH IP速度模式选择 0: 低速模式 1: 高速模式
PMODE	[8]	RW	**并行模式选择** 0: 等待 1: 并行操作 如果该位为1， 那么程序闪存(PROM)的读取可以和数据闪存(DROM)的烧写同时进行。
WAIT	[2:0]	RW	闪存读等待周期 n: 闪存读取中等待n个周期
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
6MHz ≤ SYSCLK ≤ 16MHz	0	1
SYSCLK<6MHz	0	0

6.3.8 IFC_FM_ADDR(ISP 地址寄存器)

Address = Base Address+ 0x18, Reset Value = 0x00000000

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

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

6.3.9 IFC_KR(ISP 秘钥寄存器)

Address = Base Address+ 0x20, Reset Value = 0x00000000

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

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

6.3.10 IFC_IMCR(中断控制寄存器)

Address = Base Address+ 0x24, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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
ADDR_ERR	[14]	RW	
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
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
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
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
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

6.3.11 IFC_RISR(中断原始状态寄存器)

Address = Base Address+ 0x28, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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
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
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
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
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
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
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

6.3.12 IFC_MISR(中断状态寄存器)

Address = Base Address+ 0x2C, Reset Value = 0x00000000

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

Name	Bit	Type	Description
OVW_ERR	[15]	RW	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
UDEF_ERR	[13]	RW	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
PROT_ERR	[12]	RW	保护错误中断的状态 0: 该中断没有发生 1: 该中断发生
PEP_END	[2]	RW	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
PGM_END	[1]	RW	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
ERS_END	[0]	RW	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

6.3.13 IFC_ICR(中断状态清除寄存器)

Address = Base Address+ 0x30, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
OVW_ERR	[15]	W	非法操作错误中断状态清除 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
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
PROT_ERR	[12]	W	保护错误中断状态清除 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
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
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

7 系统控制器（SYSCON）

7.1 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。

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。

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（普通模式）；32.768k（低速模式）
    - 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

7.2 功能描述

7.2.1 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。

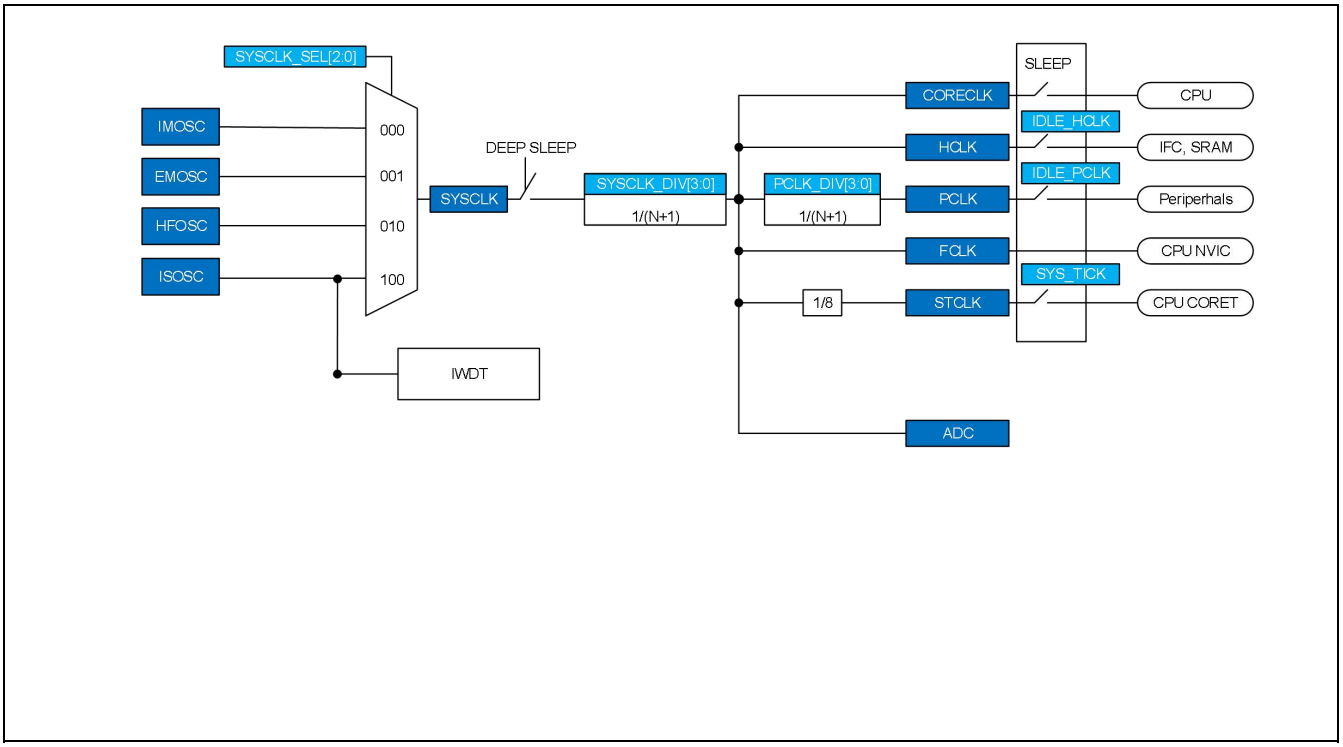


Figure 7-1 时钟结构示意图

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。

HCLK 由 SYSCLK 分频而来。给系统高速模块提供时钟，包括 CPU、内存控制单元、GPIO 控制器等。PCLK 基于 HCLK 继续分频得到，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。。

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，在使用 CORET 时，必须先通过 GCER 的控制位使能该模块的时钟。

7.2.2 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, 0.4MHz ~ 24MHz，32.768k）：

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。

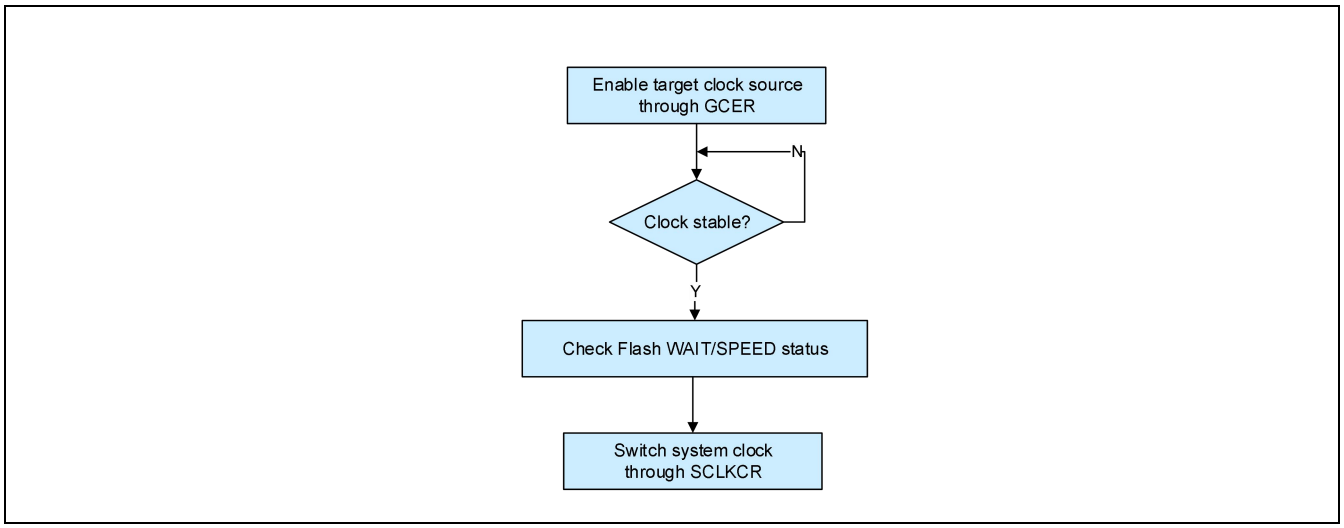


Figure 7-2 时钟源切换示意图

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。

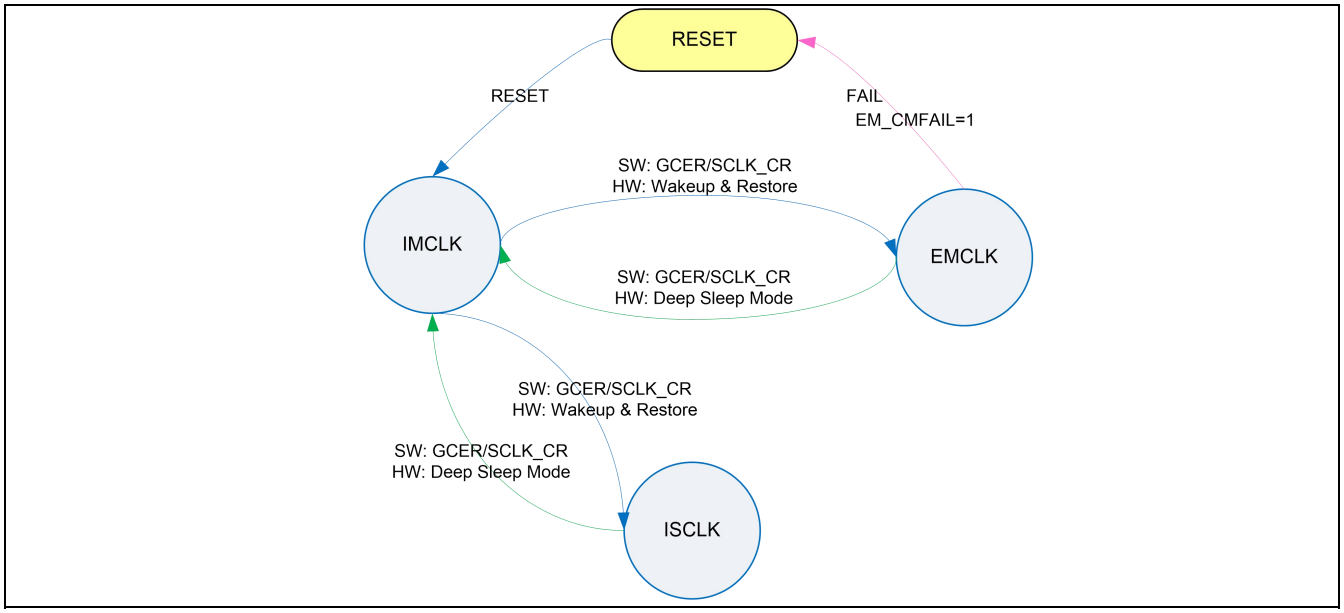


Figure 7-3 时钟切换状态机

7.2.2.1 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。

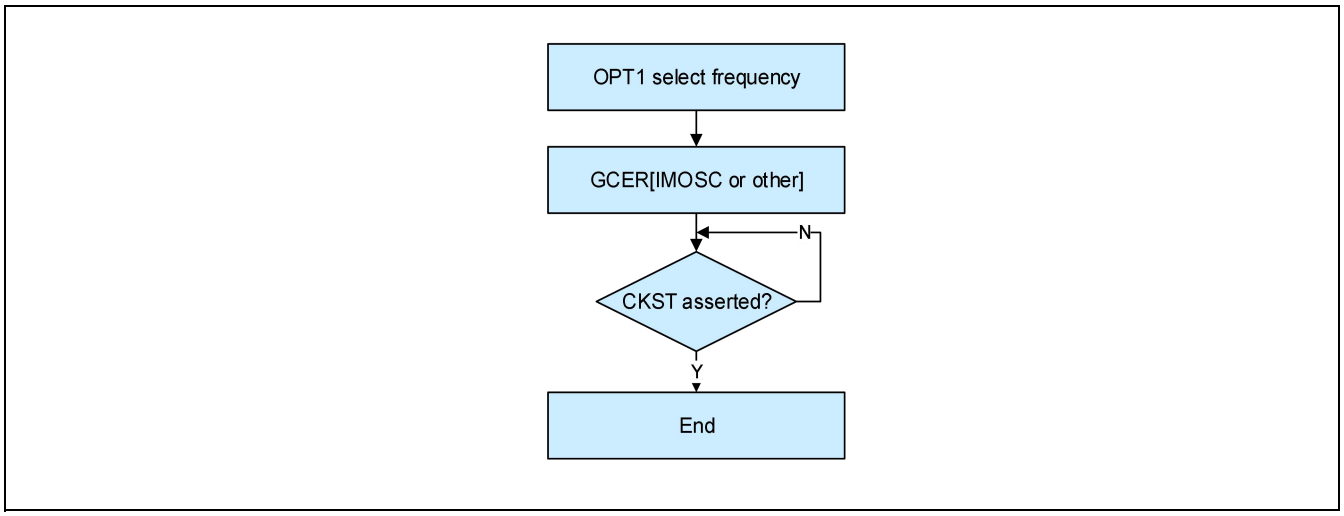


Figure 7-4 配置内部时钟源

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。

7.2.2.2 内部时钟源频率粗调

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，但方式上 HFOSC 有别于 ISOSC,IMOSC。HFOSC:步进值按所选 HFOSC 频率的 0.1458%递增或者递减。IMOSC 和 ISOSC 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。

$$Ftar = Ftrim\frac{K}{K - \Delta FSEL}$$

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 [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
ISOSC	339-TRM

例如：当前芯片 IMOSC 缺省频率为 4.108MHz，缺省 HFO_TRM 为 0xAF；若调整 IMOSC 到 4.5MHz，则先根据表格算得 K 为 367-175 = 192。根据公式计算ΔFSEL = 192 – (4.108 x 192 / 4.5)，结果为 16.7。所以 IMO_TRM 应增加 17 可以获得 Target 近似频率。

7.2.2.3 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：

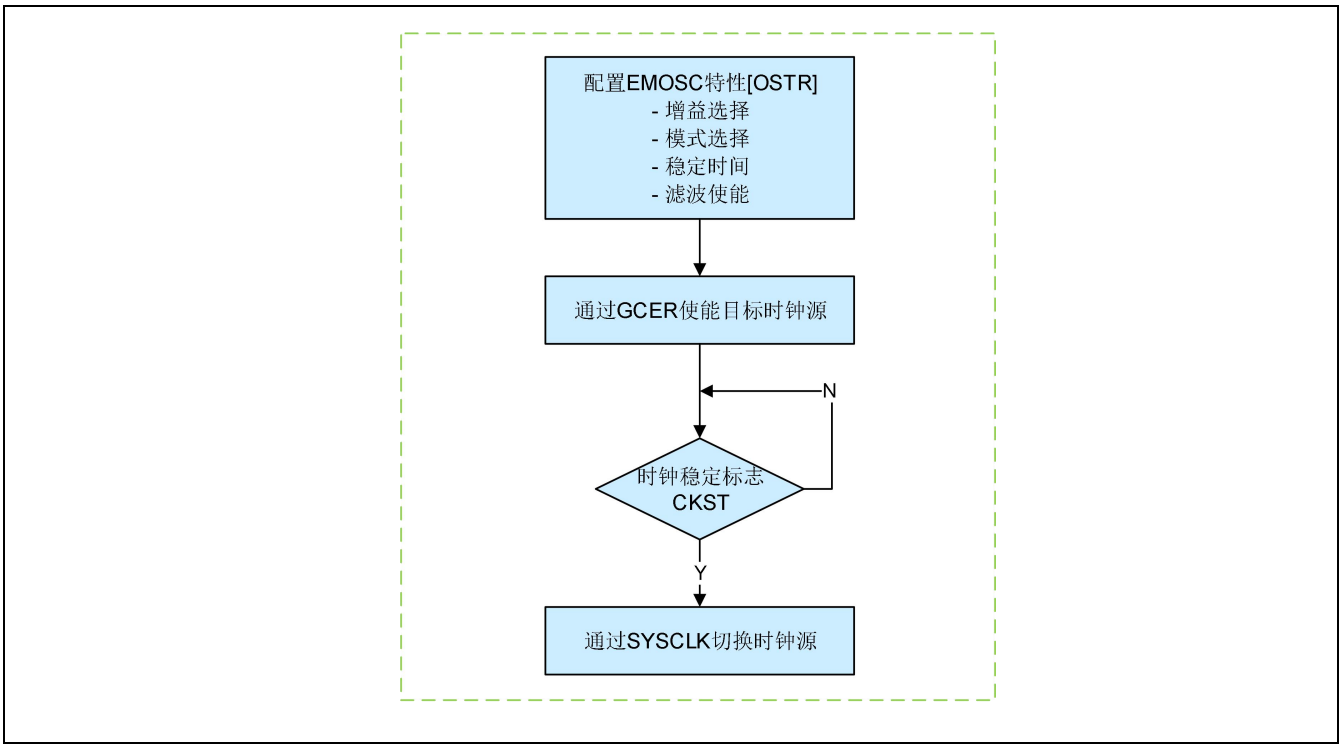


Figure 7-5 配置外部时钟源

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。

7.2.2.4 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（IMOSC 此时必须是使能状态）

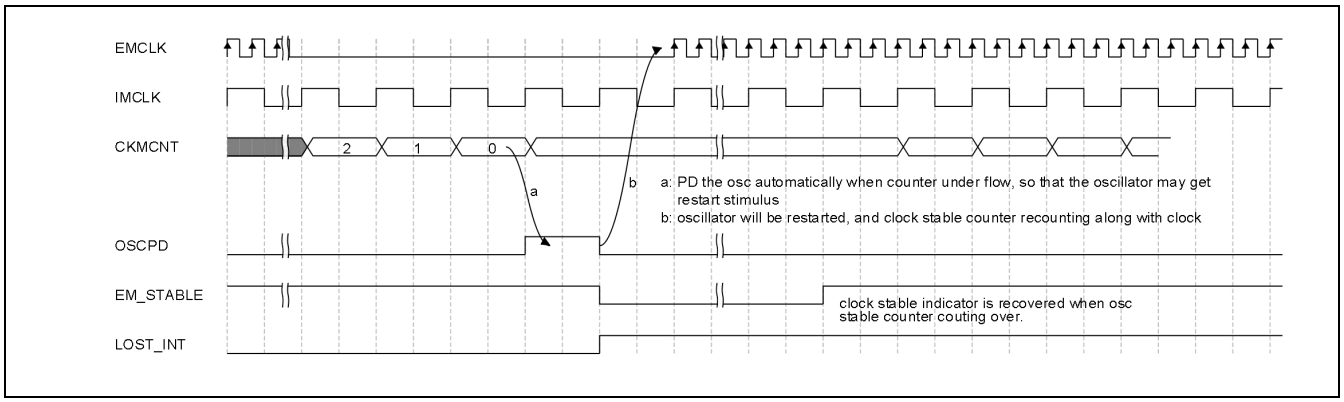


Figure 7-6 EMOSC 失效监测

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。

7.2.2.5 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。

7.2.3 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 RSTLVL 的设置值时，产生硬件复位信号。当外部供电电压 VDD 低于 INTLVL 的设置值或高于 INTLVL 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 INTPOL 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低(SLEEP 模式不受影响)，如果要精度不受影响需要开启 BGR。

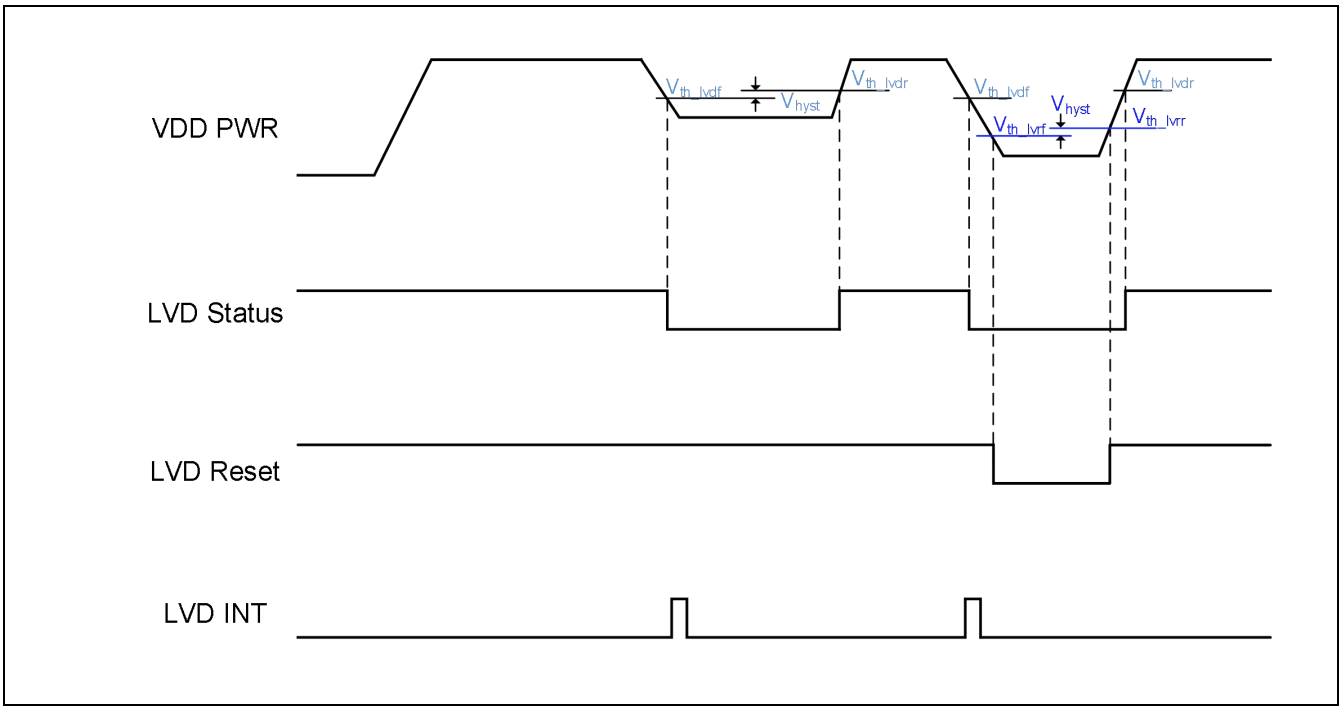


Figure 7-7 LVD 工作时序图

7.2.4 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：低功耗运行模式、睡眠模式、深睡眠模式。

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，为确保不会因为进入低功耗模式而退出调试模式，系统时钟仍旧保持工作。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。

7.2.4.1 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。

7.2.4.2 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[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。

7.2.4.3 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，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。任何外设事件或者中断都可以触发系统从该模式退出。

在 SLEEP 模式下，若 PCLK 没有被禁止，则在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。

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。

Table 7-3 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	

7.2.4.4 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。

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 7-4 可以唤醒 DEEP-SLEEP 的中断源

Peripheral	Event
GPIO	EXI interrupt of each GPIO
IWDT	Alert interrupt
LVD	Any

Table 7-5 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	

7.2.4.5 低功耗唤醒和中断服务

CPU 退出低功耗模式由 NVIC 中的唤醒寄存器（IWCR）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。系统唤醒经过两个阶段，

- 第一阶段，当 SYSCON 检测到有中断事件发生，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：

$$T_{init\_stop} = T_{imosc\_stable} + T_{clk\_sw} + T_{emo\_dis} + T_{hfo\_dis} + T_{iso\_dis} + T_{imo\_dis}$$

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。

7.2.4.6 低功耗模式总结

在不同模式间的切换和唤醒条件可以参考如下表格。

Table 7-6 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, IWDT, 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 7-7 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	—
OPA	O	O	O	—	—
UART	O	O	O	—	—
CMP	O	O	O	—	—
ADC	O	O	O	—	—
TIMER	O	O	O	—	—
IWDT	O	O	O	O	O
WWDT	O	O	O	—	—
CORET	O	O	O	—	—
EXI	O	O	O	O	O
SWD	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。

7.2.5 功耗优化和管理

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。

7.2.6 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。

7.2.7 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。

7.2.7.1 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）。

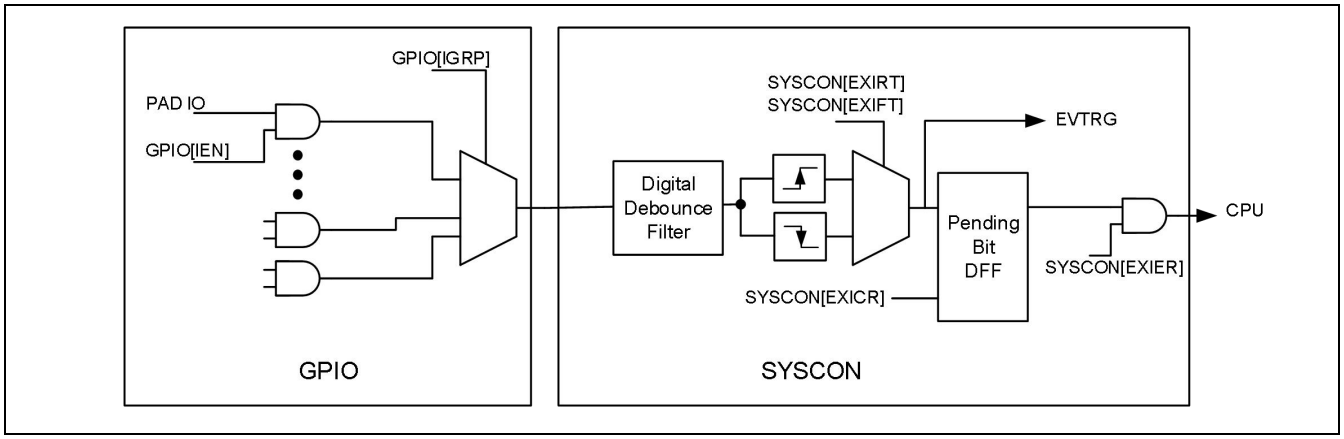


Figure 7- 8 外部中断结构示意图

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, PC0.0 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~PA0.7,PB0.0~PB0.3,PC0.0~PC0.3 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_V2
GROUP3	EXI3	EXI_V2
GROUP4	EXI4	EXI_V3
GROUP5	EXI5	EXI_V3
GROUP6	EXI6	EXI_V3
GROUP7	EXI7	EXI_V3
GROUP8	EXI8	EXI_V3
GROUP9	EXI9	EXI_V3
GROUP10	EXI10	EXI_V4
GROUP11	EXI11	EXI_V4
GROUP12	EXI12	EXI_V4
GROUP13	EXI13	EXI_V4
GROUP14	EXI14	EXI_V4

GROUP15	EXI15	EXI_V4
GROUP16	EXI16	EXI_V0
GROUP17	EXI17	EXI_V1
GROUP18	EXI18	EXI_V2
GROUP19	EXI19	EXI_V2

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。

7.2.7.2 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。

7.2.8 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，系统将会强制复位。

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。

内存校验功能分为 FLASH 校验和 SRAM 校验，分别通过 EFLCHK 和 RAMCHK 寄存器进行设置。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。

7.2.9 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。

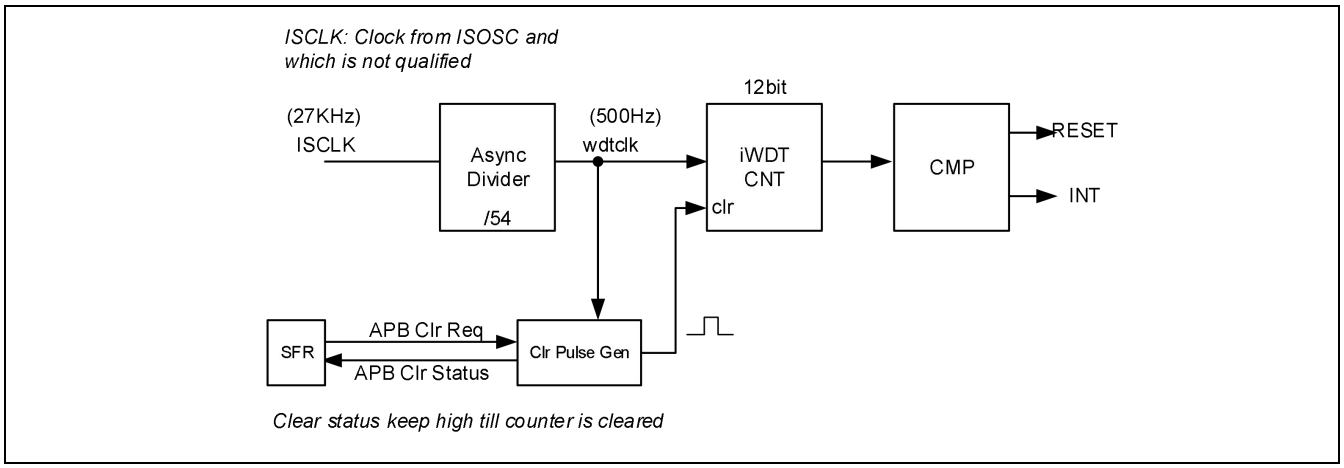


Figure 7-9 IWDT 结构框图

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 ENDIS 位来打开或者关闭 IWDT。当清除 IWDT 操作时，IWDT 中计数器的预置数会被重新置位。清除 IWDT 通过对 IWDCNT 寄存器的 CLR 位写入 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。

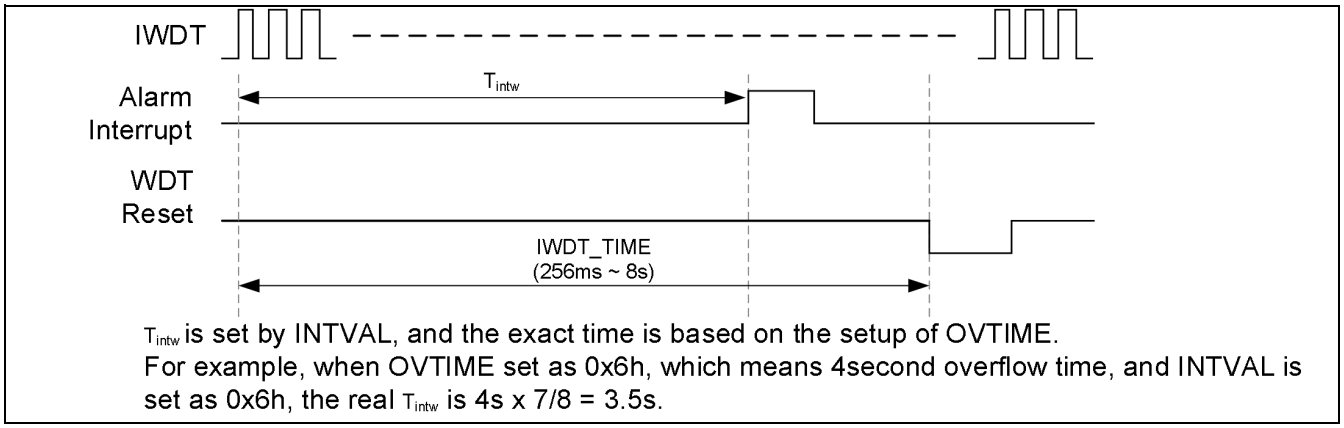


Figure 7-10 IWDT 计数器溢出和报警中断

IWDT 还支持报警功能。在计数器计数过程中，当计数值达到 INTVAL 设置值时，会产生一个中断信号。INTVAL 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 OVTIME 设置看门狗定时溢出时间为 4 秒，如果 INTVAL 设置为 3’b011，则表示在计数器计时到 4 x 4/8 = 2 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。

7.2.10 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。

Table 7-10 GPIO in IOGROUP0

GPIO	G1	CFGVAL in IOMAP0
PA0.0	G1.0	CFGVAL0
PA0.1	G1.1	CFGVAL1
PA1.0	G1.2	CFGVAL2
PA1.1	G1.3	CFGVAL3
PA1.2	G1.4	CFGVAL4
PA1.3	G1.5	CFGVAL5
PA1.4	G1.6	CFGVAL6
PA1.5	G1.7	CFGVAL7

Table 7-11 GPIO in IOGROUP1

GPIO	G2	CFGVAL in IOMAP1
PA0.2	G2.0	CFGVAL0
PA0.5	G2.1	CFGVAL1
PA0.6	G2.2	CFGVAL2
PA0.7	G2.3	CFGVAL3
PA0.8	G2.4	CFGVAL4
PA0.9	G2.5	CFGVAL5
PA0.10	G2.6	CFGVAL6
PA0.11	G2.7	CFGVAL7

IOMAP0/1[CFGVALx] 的值决定了 IO 的最终功能。详见下表。比如，当 IOMAP0[CFGVAL0] = 2, PA0.0 的 AF6 功能即为 USART_TX, 当 IOMAP1[CGFVAL3] = 0, PA0.7 的 AF6 即为 USART_TX。

Table 7-12 IO GROUP 配置信息

AF Function	CFGVAL in IOMAP	GROUP
RSVD / USART_TX	0x00	GROUP0 / GROUP1
RSVD / USART_RX	0x01	GROUP0 / GROUP1
USART_TX / USART_CK	0x02	GROUP0 / GROUP1

USART_RX / RSVD	0x03	GROUP0 / GROUP1
RSVD / RSVD	0x04	GROUP0 / GROUP1
RSVD / RSVD	0x05	GROUP0 / GROUP1
GPT_CHA / EPT_CHAX	0x06	GROUP0 / GROUP1
GPT_CHB / EPT_CHAY	0x07	GROUP0 / GROUP1

7.2.11 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。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。

7.2.12 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。

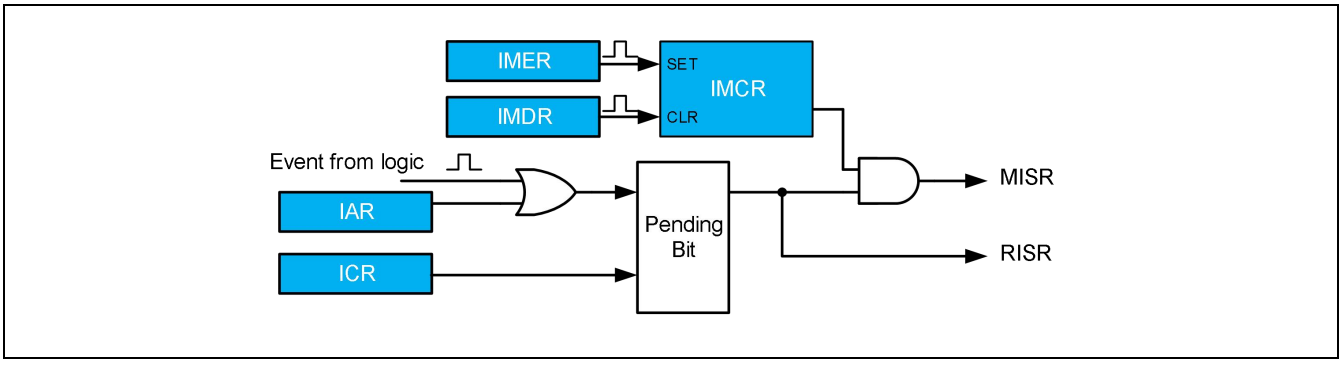


Figure 7-11 SYSCON 通用中断的控制逻辑

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 EXI_V0, EXI_V1, EXI_V2 and EXI_V3。

7.2.13 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。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 ADC 进行数据采集。
- 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 PWM 的 One Pulse 输出。

外部中断信号的流向如下图：

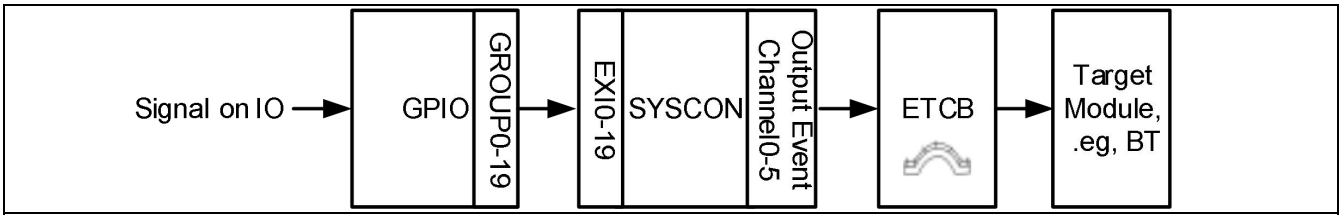


Figure 7- 12 Signal Flow

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。外部触发端口还支持事件计数功能。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。

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。

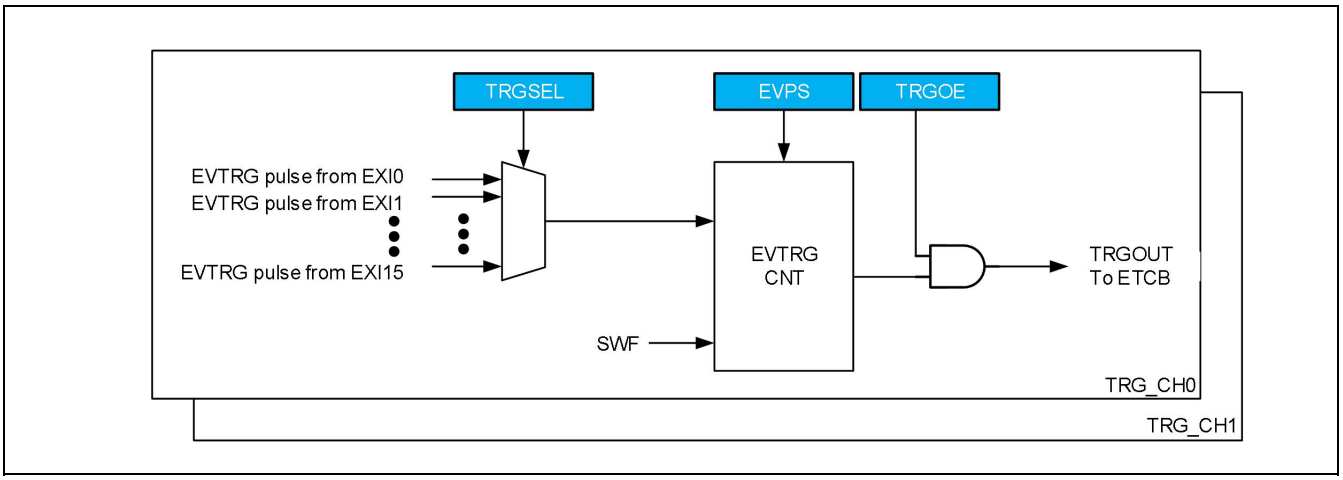


Figure 7-13 SYSCON 外部事件触发控制逻辑

7.2.14 调试管脚的复用

在芯片上电时，系统控制器会锁定调试脚的复用功能。如果需要使用调试脚的复用功能，请通过设置调试控制寄存器 DBGCR[DBG_UNLOCK] 为除 0x5a 外的任意值。

如果用户程序修改调试脚的复位功能为非调试功能，调试器和芯片的连接将会断开。

7.3 寄存器说明

7.3.1 寄存器表

Base Address of SYSCON: 0x40011000

Register	Offset	Description	Reset Value
SYSCON_IDCCR	0x000	ID和控制器模块时钟控制寄存器	0xFFFFFF01
SYSCON_GCER	0x004	通用使能控制寄存器	0x00000000
SYSCON_GCDR	0x008	通用禁止控制寄存器	0x00000000
SYSCON_GCSR	0x00C	通用状态寄存器	0x00081103
SYSCON_CKST	0x010	时钟状态寄存器	0x00000103
SYSCON_RAMCHK	0x014	SRAM 校验控制寄存器	0x0000FFFF
SYSCON_EFLCHK	0x018	Embedded Flash 校验控制寄存器	0x00FFFFFF
SYSCON_SCLKCR	0x01C	系统时钟控制寄存器	0x00000100
SYSCON_PCLKCR	0x020	外设时钟控制寄存器	0x00000100
SYSCON_PCER0	0x028	外设时钟使能寄存器0	0x00000000
SYSCON_PCDR0	0x02C	外设时钟禁止寄存器0	0x00000000
SYSCON_PCSR0	0x030	外设时钟状态寄存器0	0x00100001
SYSCON_PCER1	0x034	外设时钟使能寄存器1	0x00000000
SYSCON_PCDR1	0x038	外设时钟禁止寄存器1	0x00000000
SYSCON_PCSR1	0x03C	外设时钟状态寄存器1	0x00000000
SYSCON_OSTR	0x040	外部振荡器稳定时间配置寄存器	0x70FF3BFF
SYSCON_LVDCR	0x04C	低电压检测控制寄存器	0x0000000A
SYSCON_CLCR	0x050	内部振荡器调整寄存器	0xXXXXX100
SYSCON_PWRCR	0x054	功耗控制寄存器	0x141F1F00
SYSCON_PWRKEY	0x058	功耗调整使能寄存器	0x00000000
SYSCON_OPT1	0x064	系统配置寄存器1 (TRIM value for OSC) [1]	0x0000XXXX
SYSCON_OPT0	0x068	系统配置寄存器0 [2]	0x00000000
SYSCON_WKCR	0x06C	低功耗唤醒使能控制寄存器	0x00000000
SYSCON_IMER	0x074	中断使能控制寄存器	0x00000000
SYSCON_IMDR	0x078	中断禁止控制寄存器	0x00000000
SYSCON_IMCR	0x07C	中断使能/禁止状态寄存器	0x00000000
SYSCON_IAR	0x080	中断软件触发寄存器	0x00000000
SYSCON_ICR	0x084	中断清除寄存器	0x00000000
SYSCON_RISR	0x088	原始中断标志状态寄存器	0x00000000

SYSCON_MISR	0x08C	中断标志状态寄存器	0x00000000
SYSCON_RSR	0x090	复位源记录状态寄存器	0x00000000
SYSCON_EXIRT	0x094	外部中断上升沿选择寄存器	0x00000000
SYSCON_EXIFT	0x098	外部中断下降沿选择寄存器	0x00000000
SYSCON_EXIER	0x09C	外部中断使能寄存器	0x00000000
SYSCON_EXIDR	0x0A0	外部中断禁止寄存器	0x00000000
SYSCON_EXIMR	0x0A4	外部中断使能/禁止状态寄存器	0x00000000
SYSCON_EXIAR	0x0A8	外部中断软件触发寄存器	0x00000000
SYSCON_EXICR	0x0AC	外部中断清除寄存器	0x00000000
SYSCON_EXIRS	0x0B0	外部中断原始标志状态寄存器	0x00000000
SYSCON_IWDCR	0x0B4	独立看门狗控制寄存器	0x0000070C
SYSCON_IWDCNT	0x0B8	独立看门狗控制计数器值	0x0003FFFF
SYSCON_IWDEDR	0x0BC	独立看门狗使能寄存器	0x0000XXXX
SYSCON_IOMAP0	0x0C0	GPIO分组0的功能映射配置寄存器	0x00000000
SYSCON_IOMAP1	0x0C4	GPIO分组1的功能映射配置寄存器	0x00000000
SYSCON_UID0	0x0E4	UID寄存器0	0x00000000
SYSCON_UID1	0x0E8	UID寄存器1	0x00000000
SYSCON_UID2	0x0EC	UID寄存器2	0x00000000
SYSCON_PWROPT	0x0F0	供电恢复时间调整寄存器	0x00004040
SYSCON_EVTRG	0x0F4	事件触发选择寄存器	0x00000000
SYSCON_EVPS	0x0F8	事件触发计数寄存器	0x00000000
SYSCON_EVSWF	0x0FC	事件计数器软件触发控制寄存器	0x00000000
SYSCON_UREG0	0x100	32位用户寄存器	0x00000000
SYSCON_UREG1	0x104	32位用户寄存器	0x00000000
SYSCON_UREG2	0x108	16位用户寄存器	0x00000000
SYSCON_DBGCR	0x128	调试控制寄存器	0x0000005A

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。

7.3.2 SYSCON_IDCCR(ID和控制器模块时钟控制寄存器)

Address = Base Address+ 0x000, Reset Value = 0xFFFFFF01

31	30	29	28	27	26	25	24	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																IDCODE								SWRST			CPUFTRST				CLKEN
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	W	W	W	RW	RW	RW	RW	RW

Name	Bit	Type	Description
IDCODE/ID_KEY	[31:16]	RW	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。
IDCODE	[15:8]	R	读取时，返回ID CODE（IP版本信息）
SWRST	[7:5]	W	SYSCON产生软件复位。 4h: System reset。效果和CPU通过软复位指令产生软件复位一致。 5h: CPU Reset (只复位CPU)
CPUFTRST	[4:1]	RW	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。
CLKEN	[0]	RW	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。

7.3.3 SYSCON_GCER(通用使能控制寄存器)

Address = Base Address+ 0x004, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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		LP_EMOEN	RSVD	LP_IMOEN	LP_ISOEN	SYSTICK	RSVD	IDLE_HCLK	IDLE_PCLK	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	R	W	W

Name	Bit	Type	Description
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。
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。
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。
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_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。
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。
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。
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。
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。
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。
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。
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。

7.3.4 SYSCON_GCDR(通用禁止控制寄存器)

Address = Base Address+ 0x008, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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		LP_EMOEN	RSVD	LP_IMOEN	LP_ISOEN	SYSTICK	RSVD	IDLE_HCLK	IDLE_PCLK	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	R	W	W

Name	Bit	Type	Description
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。
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。
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。
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_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。
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。
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。
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。
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。
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。
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。
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。

7.3.5 SYSCON_GCSR(通用状态寄存器)

Address = Base Address+ 0x00C, Reset Value = 0x00081103

31	30	29	28	27	26	25	24	23	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		LP_EMOEN	RSVD	LP_IMOEN	LP_ISOEN	SYSTICK	RSVD	IDLE_HCLK	IDLE_PCLK	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	1	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
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。
EMO_CKM	[18]	R	EMOSC Clock Monitor功能状态。 0h：EMO CKM被关闭。 1h：EMO CKM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。
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_ISOEN	[12]	R	DEEP-SLEEP模式下ISOSC工作状态。 0h：ISOSC被关闭。 1h：ISOSC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。
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。
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。
HFOSC	[4]	R	HFOSC 内部振荡器工作状态。

			0h：HFOSC被关闭。 1h：HFOSC被打开。
EMOSC	[3]	R	EMOSC 振荡器工作状态。 0h：EMOSC被关闭。 1h：EMOSC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。
ISOSC	[0]	R	ISOSC 内部振荡器工作状态。 0h：ISOSC被关闭。 1h：ISOSC被打开。

7.3.6 SYSCON_CKST(时钟状态寄存器)

Address = Base Address+ 0x010, Reset Value = 0x00000103

31	30	29	28	27	26	25	24	23	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
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。
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。
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。
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。
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。
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。			

7.3.7 SYSCON_RAMCHK(SRAM 校验控制寄存器)

Address = Base Address+ 0x014, Reset Value = 0x0000FFFF

31	30	29	28	27	26	25	24	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
CHKEN	[31:24]	RW	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。
RSTEN	[23:16]	RW	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。
CHKTIMES	[15:0]	RW	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)。

7.3.8 SYSCON_EFLCHK(Embedded Flash 校验控制寄存器)

Address = Base Address+ 0x018, Reset Value = 0x00FFFFFF

31	30	29	28	27	26	25	24	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
CHKEN	[31:24]	RW	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。
CHKTIMES	[23:0]	RW	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)。
NOTE: 1）该配置寄存器缺省复位值可以由User Option进行设置。			

7.3.9 SYSCON_SCLKCR(系统时钟控制寄存器)

Address = Base Address+ 0x01C, Reset Value = 0x00000100

31	30	29	28	27	26	25	24	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
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	R	R	R	R	RW	RW	RW	RW	R	R	R	R	R	RW	RW	RW

Name	Bit	Type	Description
SCLK_KEY	[31:16]	W	
SCLK_DIV	[11:8]	RW	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。
SCLK_SEL	[2:0]	RW	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
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。			

7.3.10 SYSCON_PCLKCR(外设时钟控制寄存器)

Address = Base Address+ 0x020, Reset Value = 0x00000100

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

Name	Bit	Type	Description
PCLK_KEY	[31:16]	W	
PCLK_DIV	[11:8]	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。			

7.3.11 SYSCON_PCER0(外设时钟使能寄存器0)

Address = Base Address+ 0x028, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																						UART0	USART0	ETCB	RSVD		ADC0	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	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	R	R	W	R	R	R	W

Name	Bit	Type	Description
UART0	[9]	W	
USART0	[8]	W	
ETCB	[7]	W	
ADC0	[4]	W	
IFC	[0]	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。			

7.3.12 SYSCON_PCDR0(外设时钟禁止寄存器0)

Address = Base Address+ 0x02C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																						UART0	USART0	ETCB	RSVD		ADC0	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	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	R	R	W	R	R	R	W

Name	Bit	Type	Description
UART0	[9]	W	
USART0	[8]	W	
ETCB	[7]	W	
ADC0	[4]	W	
IFC	[0]	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。			

7.3.13 SYSCON_PCSR0(外设时钟状态寄存器0)

Address = Base Address+ 0x030, Reset Value = 0x00100001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																						UART0	USART0	ETCB	RSVD		ADC0	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
UART0	[9]	R	
USART0	[8]	R	
ETCB	[7]	R	
ADC0	[4]	R	
IFC	[0]	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。			

7.3.14 SYSCON_PCER1(外设时钟使能寄存器1)

Address = Base Address+ 0x034, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									OPA	RSVD				CMP		RSVD			RSVD	RSVD	RSVD	RSVD	RSVD	EPT0	GPT0	BT3	BT2	BT1	BT0	RSVD	WWDT
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	W	R	R	R	R	W	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W	R	W

Name	Bit	Type	Description
OPA	[22]	W	
CMP	[17]	W	
EPT0	[7]	W	
GPT0	[6]	W	
BT3	[5]	W	
BT2	[4]	W	
BT1	[3]	W	
BT0	[2]	W	
WWDT	[0]	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。			

7.3.15 SYSCON_PCDR1(外设时钟禁止寄存器1)

Address = Base Address+ 0x038, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									OPA	RSVD				CMP		RSVD			RSVD	RSVD	RSVD	RSVD	RSVD	EPT0	GPT0	BT3	BT2	BT1	BT0	RSVD	WWDT
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	W	R	R	R	R	W	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W	R	W

Name	Bit	Type	Description
OPA	[22]	W	
CMP	[17]	W	
EPT0	[7]	W	
GPT0	[6]	W	
BT3	[5]	W	
BT2	[4]	W	
BT1	[3]	W	
BT0	[2]	W	
WWDT	[0]	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。			

7.3.16 SYSCON_PCSR1(外设时钟状态寄存器1)

Address = Base Address+ 0x03C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									OPA	RSVD				CMP		RSVD			RSVD	RSVD	RSVD	RSVD	RSVD	EPT0	GPT0	BT3	BT2	BT1	BT0	RSVD	WWDT
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
OPA	[22]	R	
CMP	[17]	R	
EPT0	[7]	R	
GPT0	[6]	R	
BT3	[5]	R	
BT2	[4]	R	
BT1	[3]	R	
BT0	[2]	R	
WWDT	[0]	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。			

7.3.17 SYSCON_OSTR(外部振荡器稳定时间配置寄存器)

Address = Base Address+ 0x040, Reset Value = 0x70FF3BFF

31	30	29	28	27	26	25	24	23	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
EM_FLTSEL	[27:26]	RW	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。
EM_FLTEN	[25]	RW	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_GMCTL	[15:11]	RW	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_LFSEL	[10]	RW	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。
EMCNT	[9:0]	RW	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

7.3.18 SYSCON_LVDCR(低电压检测控制寄存器)

Address = Base Address+ 0x04C, Reset Value = 0x0000000A

31	30	29	28	27	26	25	24	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	R	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
LVD_KEY	[31:16]	W	
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。
RSTLVL	[14:12]	RW	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
INTLVL	[10:8]	RW	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。
INTPOL	[7:6]	RW	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。

LVDEN	[3:0]	RW	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。
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。 2）如果需要在deepsleep下保证LVD精度，需要打开BGR。			

7.3.19 SYSCON_CLCR(内部振荡器调整寄存器)

Address = Base Address+ 0x050, Reset Value = 0xXXXXX100

31	30	29	28	27	26	25	24	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								RSVD								HFO_TRM							
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
ISO_TRM	[31:24]	RW	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。
IMO_TRM	[23:16]	RW	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。
HFO_TRM	[8:0]	RW	HFOSC 频率的粗调设置。 通过 HFO_TRM控制位，可以调整 HFOSC 的输出频率。调整的步进大约为HFOSC的0.1458%， 例如，当HFOSC为48000000，步进值就大约为70KHz。 0x1FF：标准输出频率（工程校准后的输出值）。 0x1FE：标准输出频率-70KHz。 0x1FD：标准输出频率-140KHz。 0x1FC：标准输出频率-210KHz ……..

7.3.20 SYSCON_PWRCR(功耗控制寄存器)

Address = Base Address+ 0x054, Reset Value = 0x141F1F00

31	30	29	28	27	26	25	24	23	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
DSL_CFG	[28:24]	RW	DEEP-SLEEP模式下下功耗策略控制，只有当VOSEN[2]置位时有效。 bit0: VCref(VDDCORE 参考)选择 0h: CMOS(0.9V) 1h: BGR(1.0V) bit1: VDDCORE 控制 0h: VDDCORE = VCref x 1.0 1h: VDDCORE = VCref x 1.2 bit2: Main regulator POWERDOWN控制 0h: disable 1h: power-down: 固定VDDCORE = VCref x 1.0的同时消耗更低的功耗，驱动能力弱。
SLP_CFG	[20:16]	RW	SLEEP模式下下功耗策略控制，只有当VOSEN[1]置位时有效。 bit0: VCref(VDDCORE 参考)选择 0h: CMOS(0.9V) 1h: BGR(1.0V) bit1: VDDCORE 控制 0h: VDDCORE = VCref x 1.0 1h: VDDCORE = VCref x 1.2 bit2: Main regulator POWERDOWN控制 0h: disable 1h: power-down: 固定VDDCORE = VCref x 1.0的同时消耗更低的功耗，驱动能力弱。
RUN_CFG	[12:8]	RW	RUN模式下功耗策略控制，只有当VOSEN[0]置位时有效。 bit0: VCref(VDDCORE 参考)选择 0h: CMOS(0.9V) 1h: BGR(1.0V) bit1: VDDCORE 控制 0h: VDDCORE = VCref x 1.0 1h: VDDCORE = VCref x 1.2

			bit2: Main regulator POWERDOWN控制 0h: disable 1h: power-down: 固定VDDCORE = VCref x 1.0的同时消耗更低的功耗，驱动能力弱。 bit3: Main regulator SLEEP控制 0h: disable 1h: enable: Main regulator工作SLEEP(低速)模式，低功耗。
VOSEN	[3:0]	RW	VOS模式使能控制。 xxx1b：使能RUN模式下的功耗策略控制 xx1xb：使能SLEEP模式下的功耗策略控制 x1xxb：使能DEEP-SLEEP模式下的功耗策略控制
NOTE: 1）该寄存器的配置需要先在PWRKEY中写入KEY值： 0xA66A。 NOTE: 2）改变配置后，只有当下次切换到该模式时，新配置才生效。 NOTE: 3）在各种模式下，VCref的选择可独立配置。 NOTE: 4）改变VDDCORE的电压，对芯片功耗、最高工作频率、内部振荡器精度、内部参考电压、LVD、LVR都会产生影响。一般而言，电压越低，功耗越低，芯片最高工作频率越低、内部振荡器精度越差、内部参考电压越低。 NOTE: 5）驱动能力随着功耗的降低而越小，所以某些配置下，系统可能无法正常工作。			

7.3.21 SYSCON_PWRKEY(功耗调整使能寄存器)

Address = Base Address+ 0x058, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
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
PWR_KEY	[31:16]	W	
VOSLCK	[15:0]	RW	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。			

7.3.22 SYSCON_OPT1(系统配置寄存器1 (TRIM value for OSC) [1])

Address = Base Address+ 0x064, Reset Value = 0x0000XXXX

31	30	29	28	27	26	25	24	23	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_FSEL		RSVD		IMO_FSEL	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
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
EMCKM_DUR	[23:21]	RW	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
EXIFLTCKS	[18:17]	RW	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
EXIFLTEN	[16]	RW	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
EFL_LPMD	[15]	RW	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。
CLODIV	[14:12]	RW	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。
CLOMX	[11:8]	RW	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。 7h：PCLK输出。 8h：HCLK输出。 9h：IWDTCLK输出。 Dh：SYSCLK输出。 其他：保留。
HFO_FSEL	[5:4]	RW	HFOSC频率选择。 0h：48MHz。 1h：24MHz。 2h：12MHz。 3h：6MHz。
IMO_FSEL	[1:0]	RW	IMOSC频率选择。 0h：5.556MHz。 1h：4.194MHz。 2h：2.097MHz。 3h：131.072KHz。

7.3.23 SYSCON_OPT0(系统配置寄存器0 [2])

Address = Base Address+ 0x068, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				RDP	RSVD									HDP_4K	HDP_ALL	RSVD							DBP	CIPVALID	RSVD					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	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
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。
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。
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。
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。
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。
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。
IWDTEN	[0]	R	独立看门狗设置状态查询。 0h：缺省关闭看门狗。 1h：缺省打开看门狗。
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。			

7.3.24 SYSCON_WKCR(低功耗唤醒使能控制寄存器)

Address = Base Address+ 0x06C, Reset Value = 0x00000000

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

Name	Bit	Type	Description
LVD_WKEN	[11]	RW	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。
IWDT_WKEN	[8]	RW	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。
NOTE: EXI 始终可以唤醒低功耗模式，所以不存在EXI的WKEN控制。			

7.3.25 SYSCON_IMER(中断使能控制寄存器)

Address = Base Address+ 0x074, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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			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	R	W	R	W	W

Name	Bit	Type	Description
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。
EV3TRG	[22]	W	同步触发输出Event3 触发的中断
EV2TRG	[21]	W	同步触发输出Event2 触发的中断
EV1TRG	[20]	W	同步触发输出Event1 触发的中断
EV0TRG	[19]	W	同步触发输出Event0 触发的中断
EM_CMFAIL	[18]	W	EMOSC时钟失效中断
OPL_ERR	[14]	W	Option初始化配置加载失败中断。
EFL_ERR	[13]	W	EFLASH校验失败中断。
HWD_ERR	[12]	W	硬件除法器除零中断。
LVD_INT	[11]	W	LVD中断。
RAM_ERR	[10]	W	SRAM校验失败中断。
IWDT_INT	[8]	W	IWDT中断。
SYSCLK_ST	[7]	W	SYSCLK时钟稳定中断。
EMOSC_ST	[3]	W	EMOSC时钟稳定中断。
IMOSC_ST	[1]	W	IMOSC时钟稳定中断。
ISOSC_ST	[0]	W	ISOSC时钟稳定中断。

7.3.26 SYSCON_IMDR(中断禁止控制寄存器)

Address = Base Address+ 0x078, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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			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	R	W	R	W	W

Name	Bit	Type	Description
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。
EV3TRG	[22]	W	同步触发输出Event3 触发的中断
EV2TRG	[21]	W	同步触发输出Event2 触发的中断
EV1TRG	[20]	W	同步触发输出Event1 触发的中断
EV0TRG	[19]	W	同步触发输出Event0 触发的中断
EM_CMFAIL	[18]	W	EMOSC时钟失效中断
OPL_ERR	[14]	W	Option初始化配置加载失败中断。
EFL_ERR	[13]	W	EFLASH校验失败中断。
HWD_ERR	[12]	W	硬件除法器除零中断。
LVD_INT	[11]	W	LVD中断。
RAM_ERR	[10]	W	SRAM校验失败中断。
IWDT_INT	[8]	W	IWDT中断。
SYSCLK_ST	[7]	W	SYSCLK时钟稳定中断。
EMOSC_ST	[3]	W	EMOSC时钟稳定中断。
IMOSC_ST	[1]	W	IMOSC时钟稳定中断。
ISOSC_ST	[0]	W	ISOSC时钟稳定中断。

7.3.27 SYSCON_IMCR(中断使能/禁止状态寄存器)

Address = Base Address+ 0x07C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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			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	R	RW	R	RW	RW

Name	Bit	Type	Description
CMD_ERR	[29]	RW	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。
EV3TRG	[22]	RW	同步触发输出Event3 触发的中断。 0h：禁止中断。 1h：使能中断。
EV2TRG	[21]	RW	同步触发输出Event2 触发的中断。 0h：禁止中断。 1h：使能中断。
EV1TRG	[20]	RW	同步触发输出Event1 触发的中断。 0h：禁止中断。 1h：使能中断。
EV0TRG	[19]	RW	同步触发输出Event0 触发的中断。 0h：禁止中断。 1h：使能中断。
EM_CMFAIL	[18]	RW	EMOSC时钟失效中断。 0h：禁止中断。 1h：使能中断。
OPL_ERR	[14]	RW	Option初始化配置加载失败中断。 0h：禁止中断。 1h：使能中断。
EFL_ERR	[13]	RW	EFLASH校验失败中断。 0h：禁止中断。 1h：使能中断。
HWD_ERR	[12]	RW	硬件除法器除零中断。 0h：禁止中断。

			1h：使能中断。
LVD_INT	[11]	RW	LVD中断。 0h：禁止中断。 1h：使能中断。
RAM_ERR	[10]	RW	SRAM校验失败中断。 0h：禁止中断。 1h：使能中断。
IWDT_INT	[8]	RW	IWDT中断。 0h：禁止中断。 1h：使能中断。
SYSCLK_ST	[7]	RW	SYSCLK时钟稳定中断。 0h：禁止中断。 1h：使能中断。
EMOSC_ST	[3]	RW	EMOSC时钟稳定中断。 0h：禁止中断。 1h：使能中断。
IMOSC_ST	[1]	RW	IMOSC时钟稳定中断。 0h：禁止中断。 1h：使能中断。
ISOSC_ST	[0]	RW	ISOSC时钟稳定中断。 0h：禁止中断。 1h：使能中断。

7.3.28 SYSCON_IAR(中断软件触发寄存器)

Address = Base Address+ 0x080, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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			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	R	W	R	W	W

Name	Bit	Type	Description
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。
EV3TRG	[22]	W	同步触发输出Event3 触发的中断
EV2TRG	[21]	W	同步触发输出Event2 触发的中断
EV1TRG	[20]	W	同步触发输出Event1 触发的中断
EV0TRG	[19]	W	同步触发输出Event0 触发的中断
EM_CMFAIL	[18]	W	EMOSC时钟失效中断
OPL_ERR	[14]	W	Option初始化配置加载失败中断。
EFL_ERR	[13]	W	EFLASH校验失败中断。
HWD_ERR	[12]	W	硬件除法器除零中断。
LVD_INT	[11]	W	LVD中断。
RAM_ERR	[10]	W	SRAM校验失败中断。
IWDT_INT	[8]	W	IWDT中断。
SYSCLK_ST	[7]	W	SYSCLK时钟稳定中断。
EMOSC_ST	[3]	W	EMOSC时钟稳定中断。
IMOSC_ST	[1]	W	IMOSC时钟稳定中断。
ISOSC_ST	[0]	W	ISOSC时钟稳定中断。

7.3.29 SYSCON_ICR(中断清除寄存器)

Address = Base Address+ 0x084, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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			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	R	W	R	W	W

Name	Bit	Type	Description
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。
EV3TRG	[22]	W	同步触发输出Event3 触发的中断
EV2TRG	[21]	W	同步触发输出Event2 触发的中断
EV1TRG	[20]	W	同步触发输出Event1 触发的中断
EV0TRG	[19]	W	同步触发输出Event0 触发的中断
EM_CMFAIL	[18]	W	EMOSC时钟失效中断
OPL_ERR	[14]	W	Option初始化配置加载失败中断。
EFL_ERR	[13]	W	EFLASH校验失败中断。
HWD_ERR	[12]	W	硬件除法器除零中断。
LVD_INT	[11]	W	LVD中断。
RAM_ERR	[10]	W	SRAM校验失败中断。
IWDT_INT	[8]	W	IWDT中断。
SYSCLK_ST	[7]	W	SYSCLK时钟稳定中断。
EMOSC_ST	[3]	W	EMOSC时钟稳定中断。
IMOSC_ST	[1]	W	IMOSC时钟稳定中断。
ISOSC_ST	[0]	W	ISOSC时钟稳定中断。

7.3.30 SYSCON_RISR(原始中断标志状态寄存器)

Address = Base Address+ 0x088, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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			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
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。
EV3TRG	[22]	R	事件触发输出Event3 触发的中断
EV2TRG	[21]	R	事件触发输出Event2 触发的中断
EV1TRG	[20]	R	事件触发输出Event1 触发的中断
EV0TRG	[19]	R	事件触发输出Event0 触发的中断
EM_CMFAIL	[18]	R	EMOSC时钟失效中断
OPL_ERR	[14]	R	Option初始化配置加载失败中断。
EFL_ERR	[13]	R	EFLASH校验失败中断。
HWD_ERR	[12]	R	硬件除法器除零中断。
LVD_INT	[11]	R	LVD中断。
RAM_ERR	[10]	R	SRAM校验失败中断。
IWDT_INT	[8]	R	IWDT中断。
SYSCLK_ST	[7]	R	SYSCLK时钟稳定中断。
EMOSC_ST	[3]	R	EMOSC时钟稳定中断。
IMOSC_ST	[1]	R	IMOSC时钟稳定中断。
ISOSC_ST	[0]	R	ISOSC时钟稳定中断。

7.3.31 SYSCON_MISR(中断标志状态寄存器)

Address = Base Address+ 0x08C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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			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
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。
EV3TRG	[22]	R	事件触发输出Event3 触发的中断
EV2TRG	[21]	R	事件触发输出Event2 触发的中断
EV1TRG	[20]	R	事件触发输出Event1 触发的中断
EV0TRG	[19]	R	事件触发输出Event0 触发的中断
EM_CMFAIL	[18]	R	EMOSC时钟失效中断
OPL_ERR	[14]	R	Option初始化配置加载失败中断。
EFL_ERR	[13]	R	EFLASH校验失败中断。
HWD_ERR	[12]	R	硬件除法器除零中断。
LVD_INT	[11]	R	LVD中断。
RAM_ERR	[10]	R	SRAM校验失败中断。
IWDT_INT	[8]	R	IWDT中断。
SYSCLK_ST	[7]	R	SYSCLK时钟稳定中断。
EMOSC_ST	[3]	R	EMOSC时钟稳定中断。
IMOSC_ST	[1]	R	IMOSC时钟稳定中断。
ISOSC_ST	[0]	R	ISOSC时钟稳定中断。

7.3.32 SYSCON_RSR(复位源记录状态寄存器)

Address = Base Address+ 0x090, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	0
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	RW	R	RW	R	RW	RW	RW

Name	Bit	Type	Description
WWDT	[13]	RW	WWDT复位。
EFL_ERR	[12]	RW	EFLASH校验错误复位。
RAM_ERR	[11]	RW	SRAM校验错误复位。
CPUFAULT	[9]	RW	CPU异常自动复位。
SWRST	[8]	RW	SYSCON产生软件复位。
CPURST	[7]	RW	CPU软件复位。
EMCKM	[6]	RW	EMOSC CKM Fail复位。
IWDT	[4]	RW	IWDT复位。
EXTRST	[2]	RW	外部复位管脚复位。
LVR	[1]	RW	LVR复位。
POR	[0]	RW	POR上电复位。
NOTE: 对相应位写入‘1’可以清除当前标志位。			

7.3.33 SYSCON_EXIRT(外部中断上升沿选择寄存器)

Address = Base Address+ 0x094, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												EXI0 ~ EXI19																			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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]	RW	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

7.3.34 SYSCON_EXIFT(外部中断下降沿选择寄存器)

Address = Base Address+ 0x098, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												EXI0 ~ EXI19																			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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]	RW	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			

7.3.35 SYSCON_EXIER(外部中断使能寄存器)

Address = Base Address+ 0x09C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												EXI0 ~ EXI19																			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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]	RW	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。

7.3.36 SYSCON_EXIDR(外部中断禁止寄存器)

Address = Base Address+ 0x0A0, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												EXI0 ~ EXI19																			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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]	RW	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。

7.3.37 SYSCON_EXIMR(外部中断使能/禁止状态寄存器)

Address = Base Address+ 0x0A4, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												EXI0 ~ EXI19																			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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]	RW	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。

7.3.38 SYSCON_EXIAR(外部中断软件触发寄存器)

Address = Base Address+ 0x0A8, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												EXI0 ~ EXI19																			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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]	RW	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。

7.3.39 SYSCON_EXICR(外部中断清除寄存器)

Address = Base Address+ 0x0AC, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												EXI0 ~ EXI19																			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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]	RW	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。

7.3.40 SYSCON_EXIRS(外部中断原始标志状态寄存器)

Address = Base Address+ 0x0B0, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												EXI0 ~ EXI19																			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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]	RW	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。

7.3.41 SYSCON_IWDCR(独立看门狗控制寄存器)

Address = Base Address+ 0x0B4, Reset Value = 0x0000070C

31	30	29	28	27	26	25	24	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	
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	R

Name	Bit	Type	Description
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
BUSY	[12]	R	看门狗工作状态。 0h：看门狗未使能。 1h：看门狗已使能。
DBGEN	[11]	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：调试使能（调试暂停时，IWDT时钟输入也暂停） 1h：调试禁止（调试暂停时间超过IWDT溢出时间，恢复运行时立刻触发复位）
OVTIME	[10:8]	RW	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：3.072s。 6h：4.096s。 7h：8.192s。
INTVAL	[4:2]	RW	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。
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。			

7.3.42 SYSCON_IWDCNT(独立看门狗控制计数器值)

Address = Base Address+ 0x0B8, Reset Value = 0x0003FFFF

31	30	29	28	27	26	25	24	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	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R	W	W	W	W	W	W	W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
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。
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。
CNT	[11:0]	R	返回IWDT当前计数值。

7.3.43 SYSCON_IWDEDR(独立看门狗使能寄存器)

Address = Base Address+ 0x0BC, Reset Value = 0x0000XXXX

31	30	29	28	27	26	25	24	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	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
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
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。
ENDIS	[15:0]	RW	IWDT使能控制。 写入0x55AA时，关闭IWDT。 写入其他值时，使能IWDT。

7.3.44 SYSCON_IOMAP0(GPIO分组0的功能映射配置寄存器)

Address = Base Address+ 0x0C0, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
CFGVAL7	[31:28]	RW	
CFGVAL6	[27:24]	RW	
CFGVAL5	[23:20]	RW	
CFGVAL4	[19:16]	RW	
CFGVAL3	[15:12]	RW	
CFGVAL2	[11:8]	RW	
CFGVAL1	[7:4]	RW	
CFGVAL0	[3:0]	RW	
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。			

7.3.45 SYSCON_IOMAP1(GPIO分组1的功能映射配置寄存器)

Address = Base Address+ 0x0C4, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
CFGVAL7	[31:28]	RW	
CFGVAL6	[27:24]	RW	
CFGVAL5	[23:20]	RW	
CFGVAL4	[19:16]	RW	
CFGVAL3	[15:12]	RW	
CFGVAL2	[11:8]	RW	
CFGVAL1	[7:4]	RW	
CFGVAL0	[3:0]	RW	
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。			

7.3.46 SYSCON_UID0(UID寄存器0)

Address = Base Address+ 0x0E4, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
UID	[31:0]	R	唯一ID寄存器。 在出厂时，由工厂写入的唯一ID码。

7.3.47 SYSCON_UID1(UID寄存器1)

Address = Base Address+ 0x0E8, Reset Value = 0x00000000

7.3.48 SYSCON_UID2(UID寄存器2)

Address = Base Address+ 0x0EC, Reset Value = 0x00000000

7.3.49 SYSCON_PWROPT(供电恢复时间调整寄存器)

Address = Base Address+ 0x0F0, Reset Value = 0x00004040

31	30	29	28	27	26	25	24	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
W	W	W	W	W	W	W	W	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
PWR_KEY	[31:24]	W	KEY: 0xB6,其他值无效
EFLR_CTL	[21:20]	RW	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。
EFLR_PD	[19:18]	RW	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。
EFL_PD	[17:16]	RW	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。
TPWRCV_SLP	[15:8]	RW	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。（计数值设0的时候实际为256）
TPWRCV_DSL	[7:0]	RW	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。（计数值设0的时候实际为256）
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。			

7.3.50 SYSCON_EVTRG(事件触发选择寄存器)

Address = Base Address+ 0x0F4, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
W	W	W	W	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
CNT3CLR	[31]	W	TRGEV3CNT软件清除 0h：无效 1h：EV3CNT重置为零
CNT2CLR	[30]	W	TRGEV2CNT软件清除 0h：无效 1h：EV2CNT重置为零
CNT1CLR	[29]	W	TRGEV1CNT软件清除 0h：无效 1h：EV1CNT重置为零
CNT0CLR	[28]	W	TRGEV0CNT软件清除 0h：无效 1h：EV0CNT重置为零
TRG5OE	[25]	RW	外部触发端口TRGOUT5使能。 0h：禁止触发输出。 1h：允许触发输出。
TRG4OE	[24]	RW	外部触发端口TRGOUT4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。
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。
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。
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。
TRGSEL5	[19:18]	RW	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。
TRGSEL4	[17:16]	RW	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。
TRGSEL3	[15:12]	RW	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。
TRGSEL2	[11:8]	RW	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。
TRGSEL1	[7:4]	RW	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。
TRGSEL0	[3:0]	RW	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。

7.3.51 SYSCON_EVPS(事件触发计数寄存器)

Address = Base Address+ 0x0F8, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
TRGEV3CNT	[31:28]	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。
TRGEV2CNT	[27:24]	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。
TRGEV1CNT	[23:20]	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。
TRGEV0CNT	[19:16]	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。
TRGEV3PRD	[15:12]	RW	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。
TRGEV2PRD	[11:8]	RW	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。
TRGEV1PRD	[7:4]	RW	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。
TRGEV0PRD	[3:0]	RW	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。

7.3.52 SYSCON_EVSWF(事件计数器软件触发控制寄存器)

Address = Base Address+ 0x0FC, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
EV5SWF	[5]	W	软件产生一次EV5的触发 0h：写入‘0’无效 1h：软件产生一次触发
EV4SWF	[4]	W	软件产生一次EV4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
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
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
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

7.3.53 SYSCON_UREG0(32位用户寄存器)

Address = Base Address+ 0x100, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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]	RW	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。			

7.3.54 SYSCON_UREG1(32位用户寄存器)

Address = Base Address+ 0x104, Reset Value = 0x00000000

7.3.55 SYSCON_UREG2(16位用户寄存器)

Address = Base Address+ 0x108, Reset Value = 0x00000000

7.3.56 SYSCON_DBGCR(调试控制寄存器)

Address = Base Address+ 0x128, Reset Value = 0x0000005A

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

Name	Bit	Type	Description
DBG_UNLOCK	[7:0]	RW	调试管脚复用功能解锁控制： others：调试管脚可设置为其他复用功能 5ah：调试管脚锁定为调试功能。软件设置GPIO的控制寄存器，虽然值会变化，但实际仍然为调试功能。 该控制位段，只有在POR时会恢复复位值。

8 事件触发控制器

8.1 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。

8.1.1 特性

- 32个可配置的事件通道
  - 通道0-2支持单个源触发多个目标
  - 通道3-31只支持单个源触发单个目标
- 支持软件触发

8.2 功能描述

8.2.1 模块框图

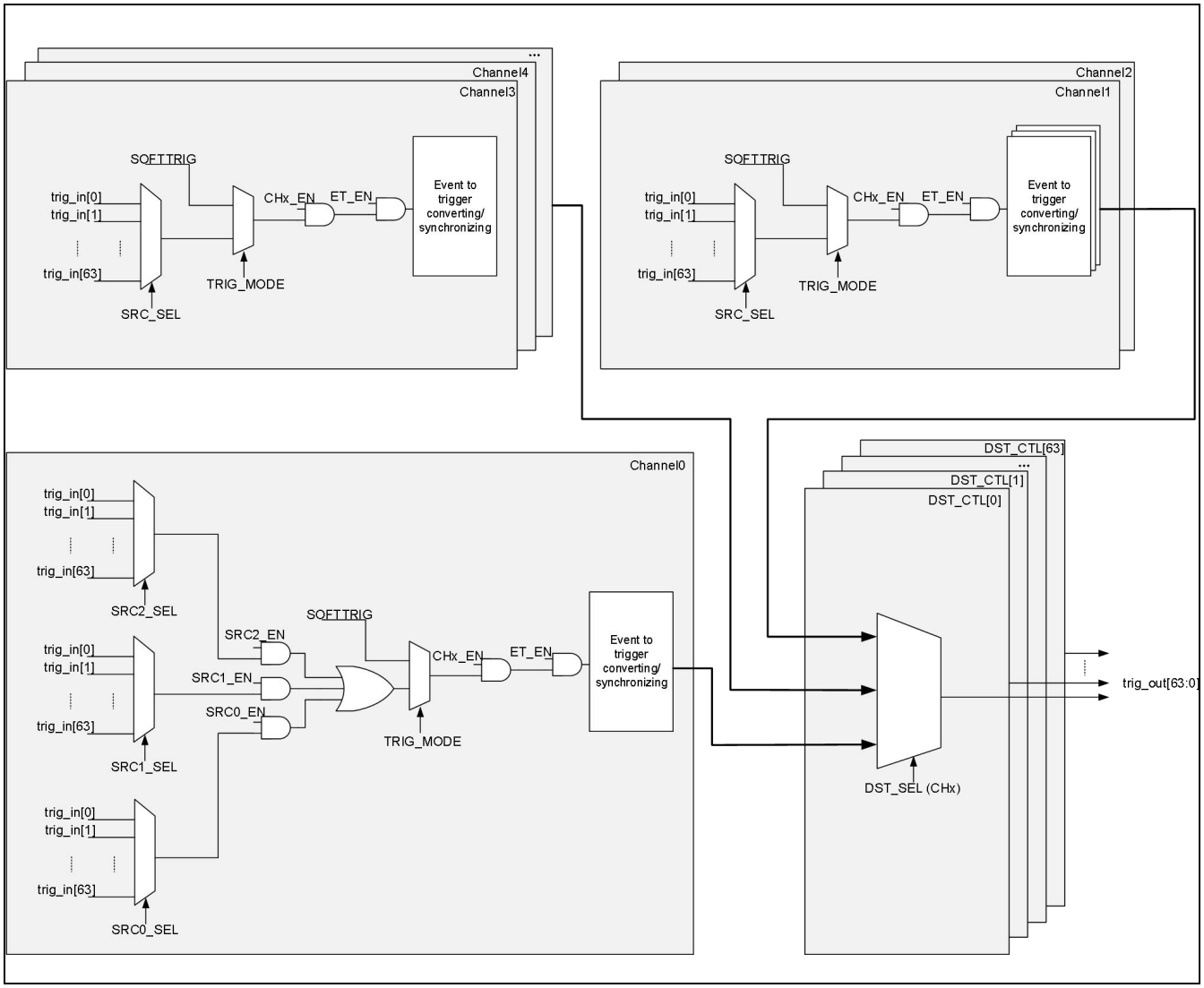


Figure 8-1 ET模块框图

8.2.2 主要功能

8.2.2.1 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 31 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、通道 1、通道 2 则可以用一个源来触发多个目标。

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 ET 模块有可能会丢失一部分触发信号。
- 事件源输出触发后，ET 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选择， 如果 2 个或者多个通道都选择了同一个目标，那么序号小的目标有更高的优先级。

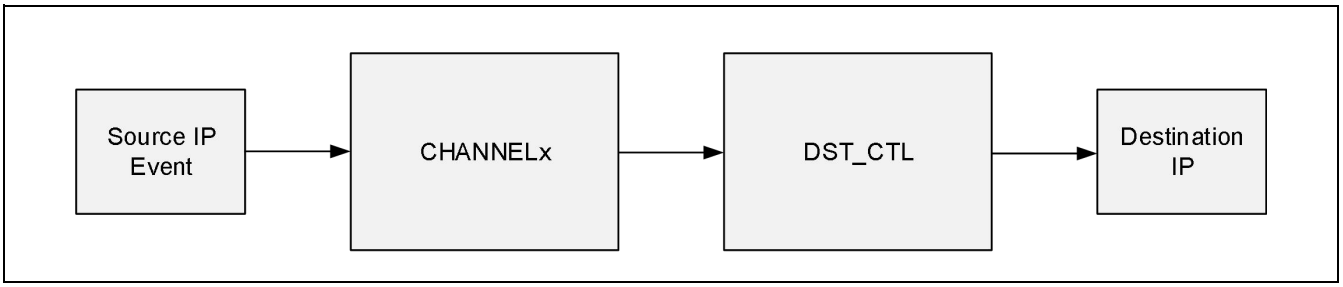


Figure 8-2 单个源触发单个目标

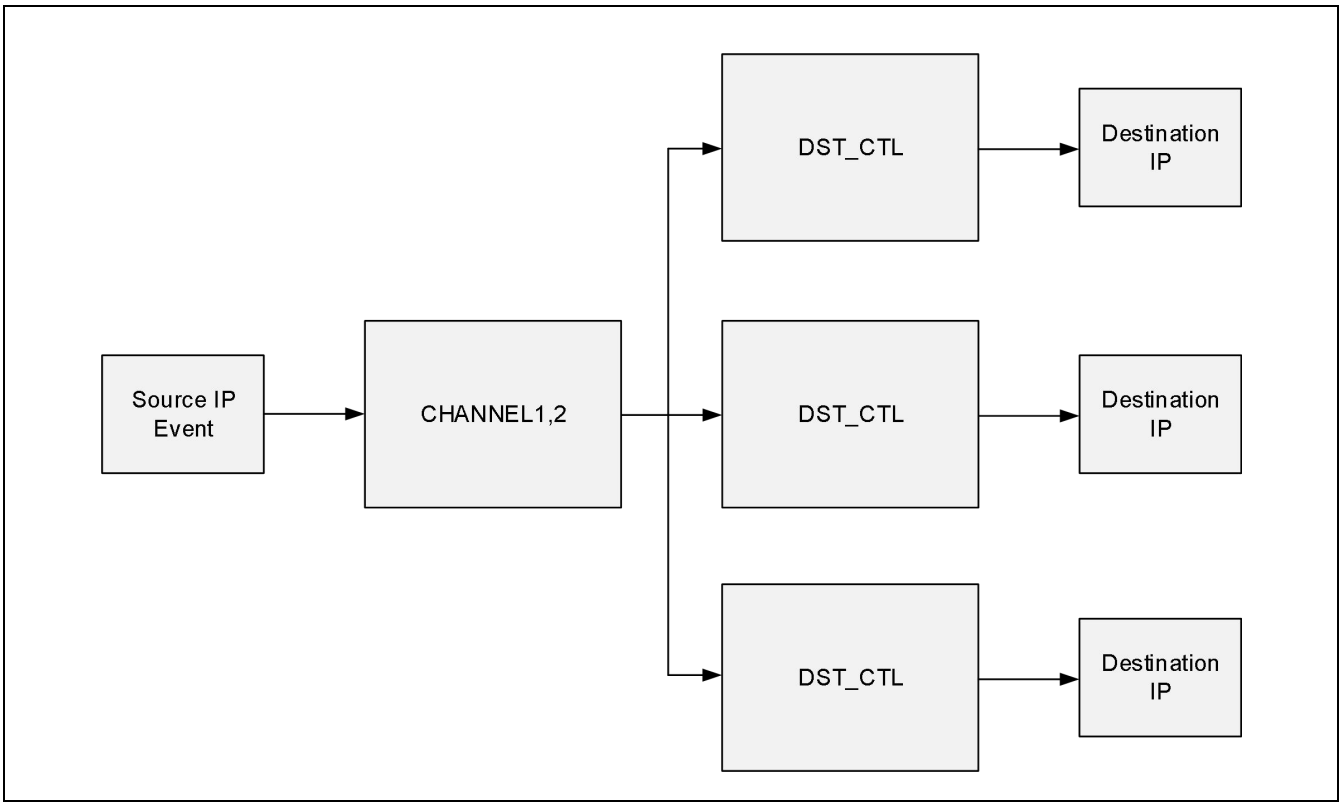


Figure 8-3 单个源触发多个目标

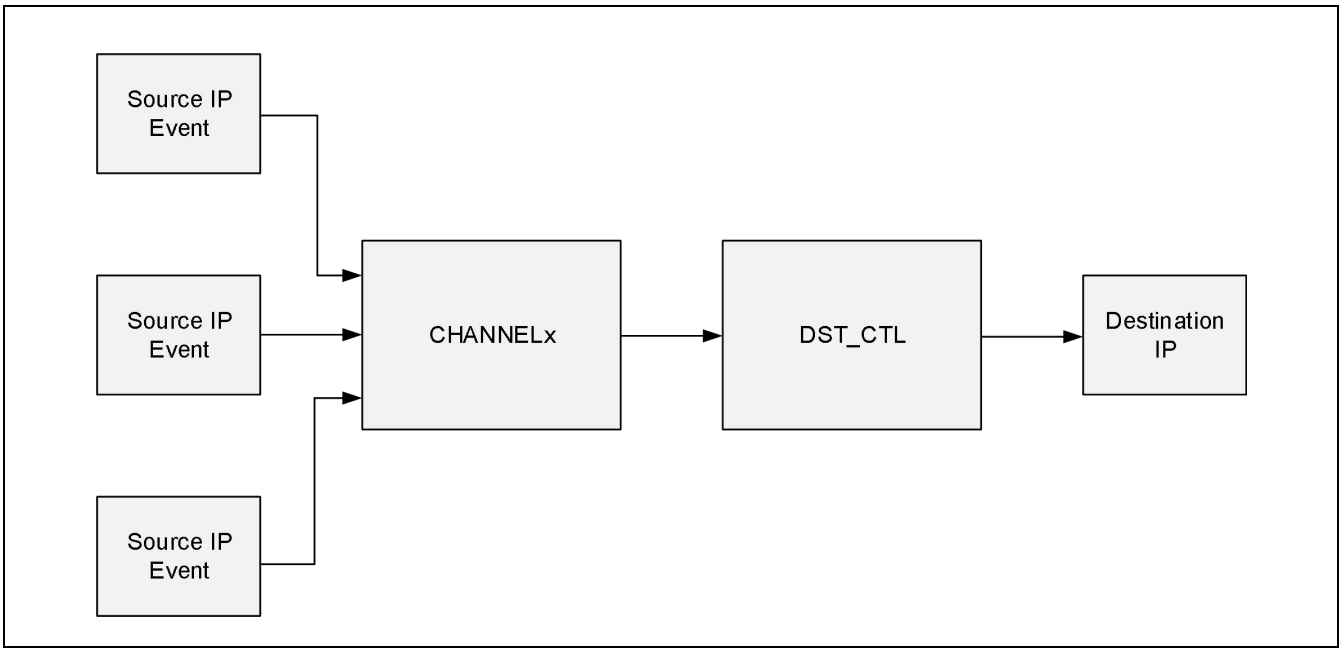


Figure 8-4 3个源触发单目标

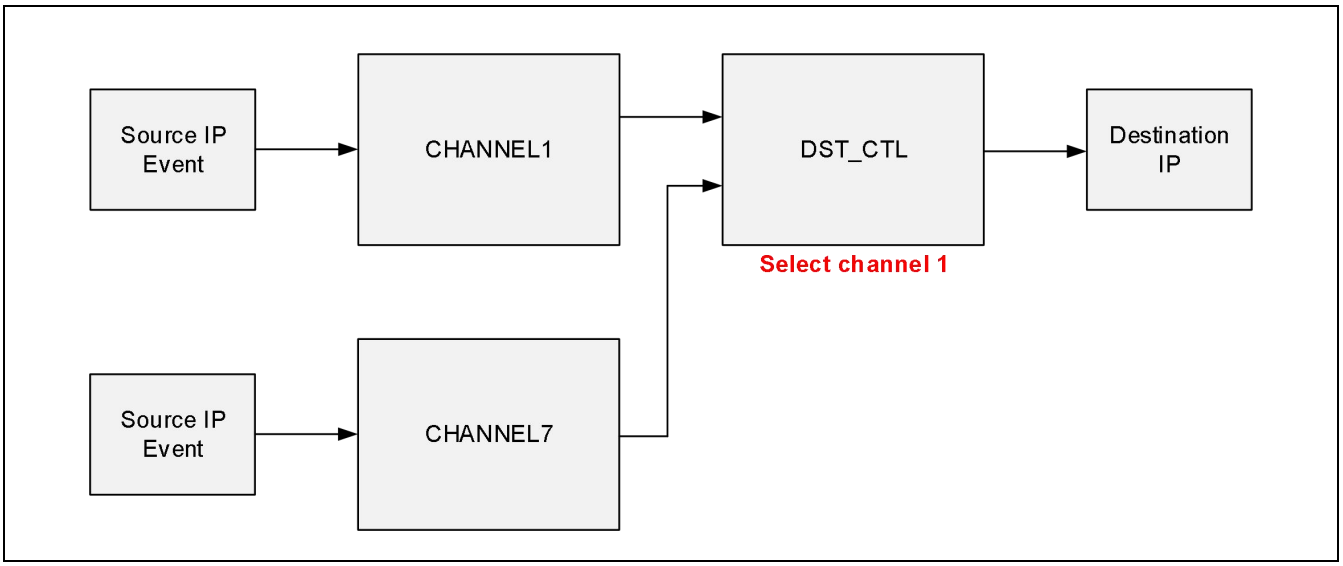


Figure 8-5 2个通道选择了相同的目标

8.2.2.2 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）	RSVD	0 （0H）	RSVD
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）	BT0_SYNCIN2
5 （5H）	EXI_TRGOUT1	5 （5H）	BT1_SYNCIN0
6 （6H）	EXI_TRGOUT2	6 （6H）	BT1_SYNCIN1
7 （7H）	EXI_TRGOUT3	7 （7H）	BT1_SYNCIN2
8 （8H）	EXI_TRGOUT4	8 （8H）	BT2_SYNCIN0
9 （9H）	EXI_TRGOUT5	9 （9H）	BT2_SYNCIN1
10 （AH）	BT0_TRGOUT	10 （AH）	BT2_SYNCIN2
11 （BH）	BT1_TRGOUT	11 （BH）	BT3_SYNCIN0
12 （CH）	BT2_TRGOUT	12 （CH）	BT3_SYNCIN1

13 （DH）	BT3_TRGOUT	13 （DH）	BT3_SYNCIN2
14 （EH）	RSVD	14 （EH）	RSVD
15 （FH）	RSVD	15 （FH）	RSVD
16 （10H）	RSVD	16 （10H）	RSVD
17 （11H）	RSVD	17 （11H）	RSVD
18 （12H）	EPT_TRGOUT0	18 （12H）	ADC_ SYNCIN0
19 （13H）	EPT_TRGOUT1	19 （13H）	ADC_ SYNCIN1
20 （14H）	EPT_TRGOUT2	20 （14H）	ADC_ SYNCIN2
21 （15H）	EPT_TRGOUT3	21 （15H）	ADC_ SYNCIN3
22 （16H）	RSVD	22 （16H）	ADC_ SYNCIN4
23 （17H）	RSVD	23 （17H）	ADC_ SYNCIN5
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）	EPT_SYNCIN0
31 （1FH）	RSVD	31 （1FH）	EPT_SYNCIN1
32 （20H）	GPT_TRGOUT0	32 （20H）	EPT_SYNCIN2
33 （21H）	GPT_TRGOUT1	33 （21H）	EPT_SYNCIN3
34 （22H）	CMP0_TRG_ADC	34 （22H）	EPT_SYNCIN4
35 （23H）	CMP1_TRG_ ADC	35 （23H）	EPT_SYNCIN5
36 （24H）	CMP2_TRG_ ADC	36 （24H）	GPT_SYNCIN0
37 （25H）	CMP3_TRG_ ADC	37 （25H）	GPT_SYNCIN1
38 （26H）	CMP4_TRG_ ADC	38 （26H）	GPT_SYNCIN2
39 （27H）	CMP5_TRG_ ADC	39 （27H）	GPT_SYNCIN3
40 （28H）	CMP0_TRG_TC1	40 （28H）	GPT_SYNCIN4

41 （29H）	CMP1_TRG_ TC1	41 （29H）	GPT_SYNCIN5
42 （2AH）	CMP2_TRG_ TC1	42 （2AH）	RSVD
43 （2BH）	CMP3_TRG_ TC1	43 （2BH）	RSVD
44 （2CH）	CMP4_TRG_ TC1	44 （2CH）	RSVD
45 （2DH）	CMP5_TRG_ TC1	45 （2DH）	RSVD
46 （2EH）	CMP_TRG_TC1_START	46 （2EH）	RSVD
47 （2FH）	CMP_TRG_ADC	47 （2FH）	RSVD
48 （30H）	ADC_TRGOUT0	48 （30H）	CMP1_SYNCIN
49 （31H）	ADC_TRGOUT1	49 （31H）	CMP2_SYNCIN
50 （32H）	RSVD	50 （32H）	CMP3_SYNCIN
51 （33H）	RSVD	51 （33H）	CMP4_SYNCIN
52 （34H）	RSVD	52 （34H）	RSVD
53 （35H）	RSVD	53 （35H）	RSVD
54 （36H）	RSVD	54 （36H）	RSVD
55 （37H）	RSVD	55 （37H）	EPWM_SYNCIN0
56 （38H）	RSVD	56 （38H）	EPWM_SYNCIN1
57 （39H）	RSVD	57 （39H）	EPWM_SYNCIN2
58 （3AH）	RSVD	58 （3AH）	EPWM_SYNCIN3
59 （3BH）	RSVD	59 （3BH）	RSVD
60 （3CH）	RSVD	60 （3CH）	RSVD
61 （3DH）	RSVD	61 （3DH）	RSVD
62 （3EH）	RSVD	62 （3EH）	RSVD
63 （3FH）	RSVD	63 （3FH）	RSVD

8.3 寄存器说明

8.3.1 寄存器表

Base Address of ETCB: 0x40012000

Register	Offset	Description	Reset Value
ETCB_ENABLE	0x0000	ETCB使能寄存器	0x00000000
ETCB_SWTRG	0x0004	ETCB软件触发寄存器	0x00000000
ETCB_CH0CON0	0x0008	ETCB通道0控制寄存器0	0x00000000
ETCB_CH0CON1	0x000C	ETCB通道0控制寄存器1	0x00000000
ETCB_CH1CON0	0x0010	ETCB通道1控制寄存器0	0x00000000
ETCB_CH1CON1	0x0014	ETCB通道1控制寄存器1	0x00000000
ETCB_CH2CON0	0x0018	ETCB通道2控制寄存器0	0x00000000
ETCB_CH2CON1	0x001C	ETCB通道2控制寄存器1	0x00000000
ETCB_CH(3)CON	0x0030 ~ 0x00A0	ETCB通道3-31控制寄存器	0x00000000

8.3.2 ETCB_ENABLE(ETCB 使能寄存器)

Address = Base Address+ 0x0000, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
ENABLE	[0]	RW	ETCB模块使能控制 0： 禁止 1： 使能

8.3.3 ETCB_SWTRG(ETCB 软件触发寄存器)

Address = Base Address+ 0x0004, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWTRG_CHx																															
0	0	0	0	0	0	0	0	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
SWTRG_CHx	[31:0]	RW	软件触发控制 0：无效 1：触发该通道的事件

8.3.4 ETCB_CH0CON0(ETCB 通道 0 控制寄存器 0)

Address = Base Address+ 0x0008, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
DST2_SEL	[26:21]	RW	触发目标2的事件选择位 参考事件对应表
DST2_EN	[20]	RW	触发目标2使能控制 0: 禁止 1: 使能
DST1_SEL	[16:11]	RW	触发目标1的事件选择位 参考事件对应表
DST1_EN	[10]	RW	触发目标1使能控制 0: 禁止 1: 使能
DST0_SEL	[6:1]	RW	触发目标0的事件选择位 参考事件对应表
DST0_EN	[0]	RW	触发目标0使能控制 0: 禁止 1: 使能

8.3.5 ETCB_CH0CON1(ETCB 通道 0 控制寄存器 1)

Address = Base Address+ 0x000C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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	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	RW	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
SRC_SEL	[31:25]	RW	触发目标选择 参考事件对应表
TRIG_MODE	[1]	RW	触发模式选择 0: 硬件触发 1: 软件触发 (由ETCB_SWTRG寄存器触发)
CH0_EN	[0]	RW	通道0使能控制 0: 禁止 1: 使能

8.3.6 ETCB_CH1CON0(ETCB 通道 1 控制寄存器 0)

Address = Base Address+ 0x0010, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
DST2_SEL	[26:21]	RW	触发目标2的事件选择位 参考事件对应表
DST2_EN	[20]	RW	触发目标2使能控制 0: 禁止 1: 使能
DST1_SEL	[16:11]	RW	触发目标1的事件选择位 参考事件对应表
DST1_EN	[10]	RW	触发目标1使能控制 0: 禁止 1: 使能
DST0_SEL	[6:1]	RW	触发目标0的事件选择位 参考事件对应表
DST0_EN	[0]	RW	触发目标0使能控制 0: 禁止 1: 使能

8.3.7 ETCB_CH1CON1(ETCB 通道 1 控制寄存器 1)

Address = Base Address+ 0x0014, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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	RW	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
SRC_SEL	[31:25]	RW	触发源选择 参考事件对应表
TRIG_MODE	[1]	RW	触发模式选择 0: 硬件触发 1: 软件触发 (由ETCB_SWTRG寄存器触发)
CH1_EN	[0]	RW	通1使能控制 0: 禁止 1: 使能

8.3.8 ETCB_CH2CON0(ETCB 通道 2 控制寄存器 0)

Address = Base Address+ 0x0018, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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_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	RW

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

8.3.9 ETCB_CH2CON1(ETCB 通道 2 控制寄存器 1)

Address = Base Address+ 0x001C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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	CH2_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	RW	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
SRC_SEL	[31:25]	RW	触发源选择 参考事件对应表
TRIG_MODE	[1]	RW	触发模式选择 0: 硬件触发 1: 软件触发 (由ETCB_SWTRG寄存器触发)
CH2_EN	[0]	RW	通道2使能控制 0: 禁止 1: 使能

8.3.10 ETCB_CH(3)CON(ETCB 通道 3-31 控制寄存器)

Address = Base Address+ 0x0030 ~ 0x00A0, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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	CH3_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	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	RW	RW

Name	Bit	Type	Description
DST_SEL	[31:26]	RW	触发目标选择 参考事件对应表
SRC_SEL	[18:12]	RW	触发源选择 参考事件对应表
TRIG_MODE	[1]	RW	触发模式选择 0: 硬件触发 1: 软件触发 (由ETCB_SWTRG寄存器触发)
CH3_EN	[0]	RW	通道3使能控制 0: 禁止 1: 使能

9 模数转换器 (ADC)

9.1 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。

9.1.1 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[14:0]，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

9.1.2 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 AIN14	模拟信号输入	模拟	-	-

9.1.3 模块框图

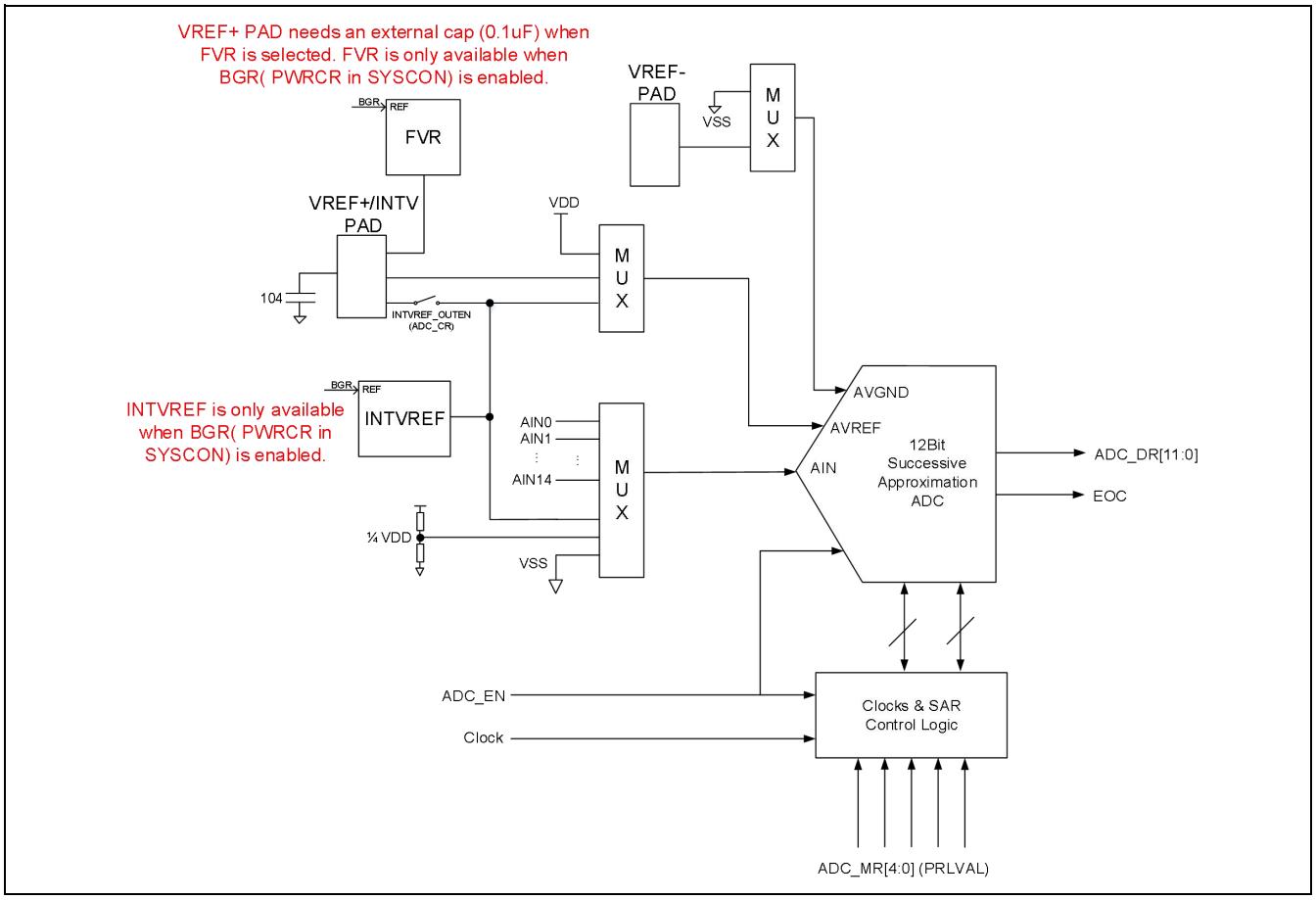


Figure 9-1 ADC模块框图

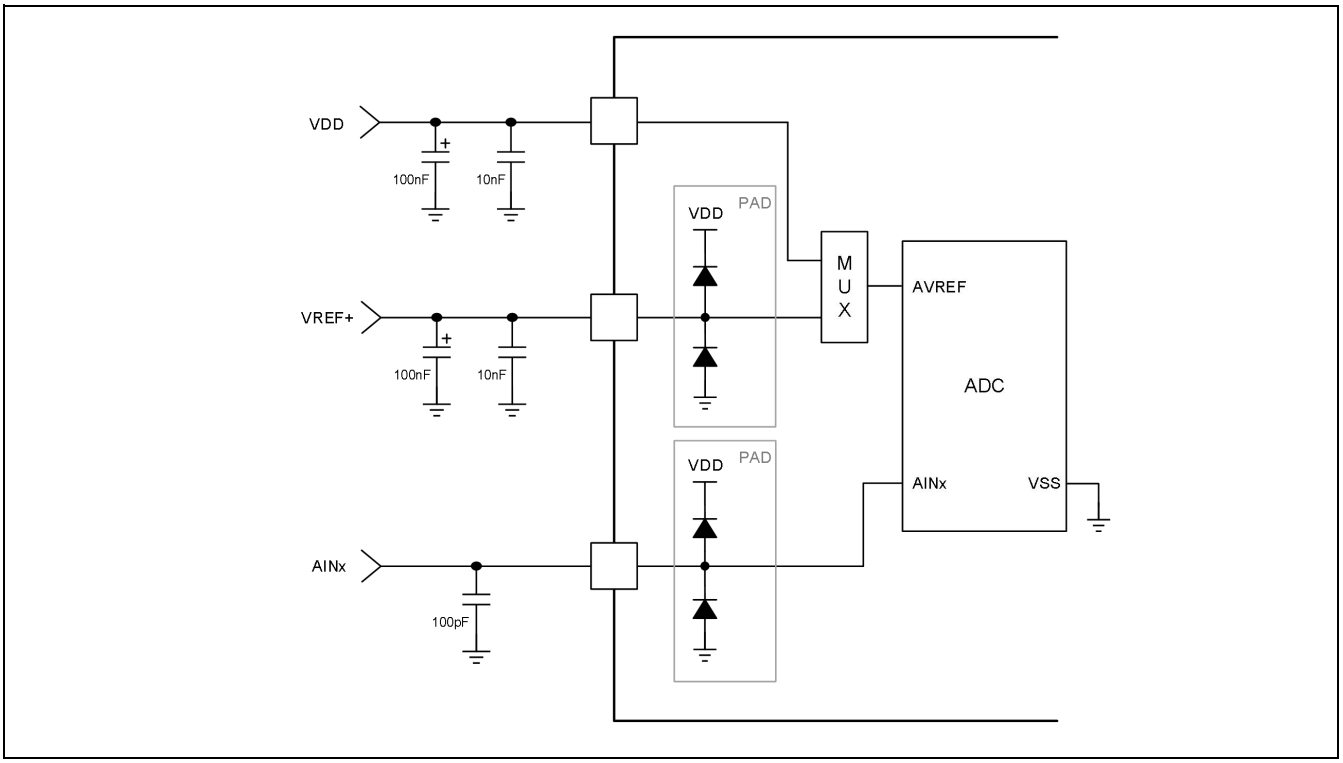


Figure 9-2 参考电路

9.1.4 输入和输出

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

Input Voltage Range: 0.0 V ~ 5.0 V
Reference Bottom Voltage: 0.0 V
Reference Top Voltage: 5.0 V

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

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

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

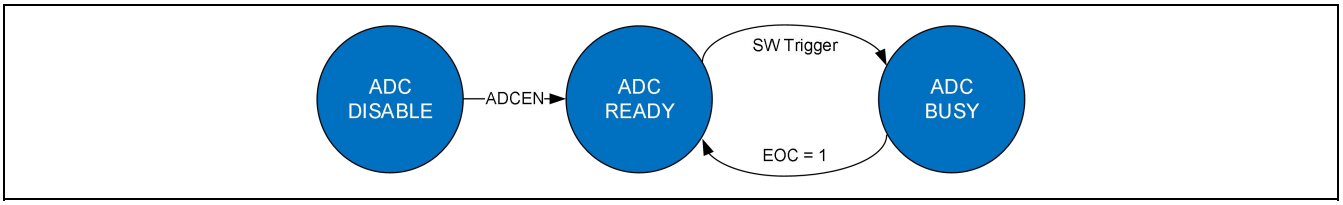


Figure 9-3 ADC状态机

9.1.5 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 9-1 ADC模块框图。

9.1.6 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+13)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

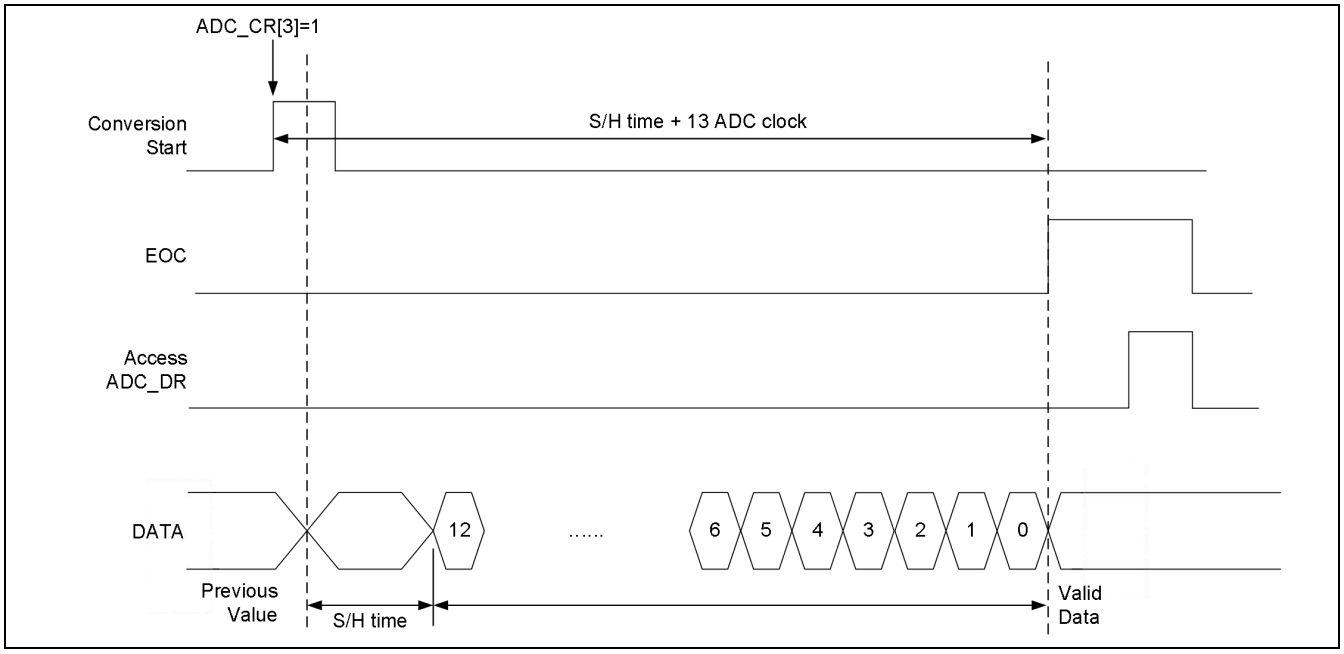


Figure 9-4 ADC工作时序图

9.1.7 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
…	No Input (input floating)	N/A
1_1100	Input 28	INTVREF
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。

9.1.8 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 15 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。

9.1.9 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。

9.1.10 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。

9.1.11 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。

9.1.12 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) 或TC1脉冲匹配中断
101	ADC_SYNCIN3触发源 (参考ETCB章节)或EPWM触发
110	ADC_SYNCIN4触发源 (参考ETCB章节)或CMP触发
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。

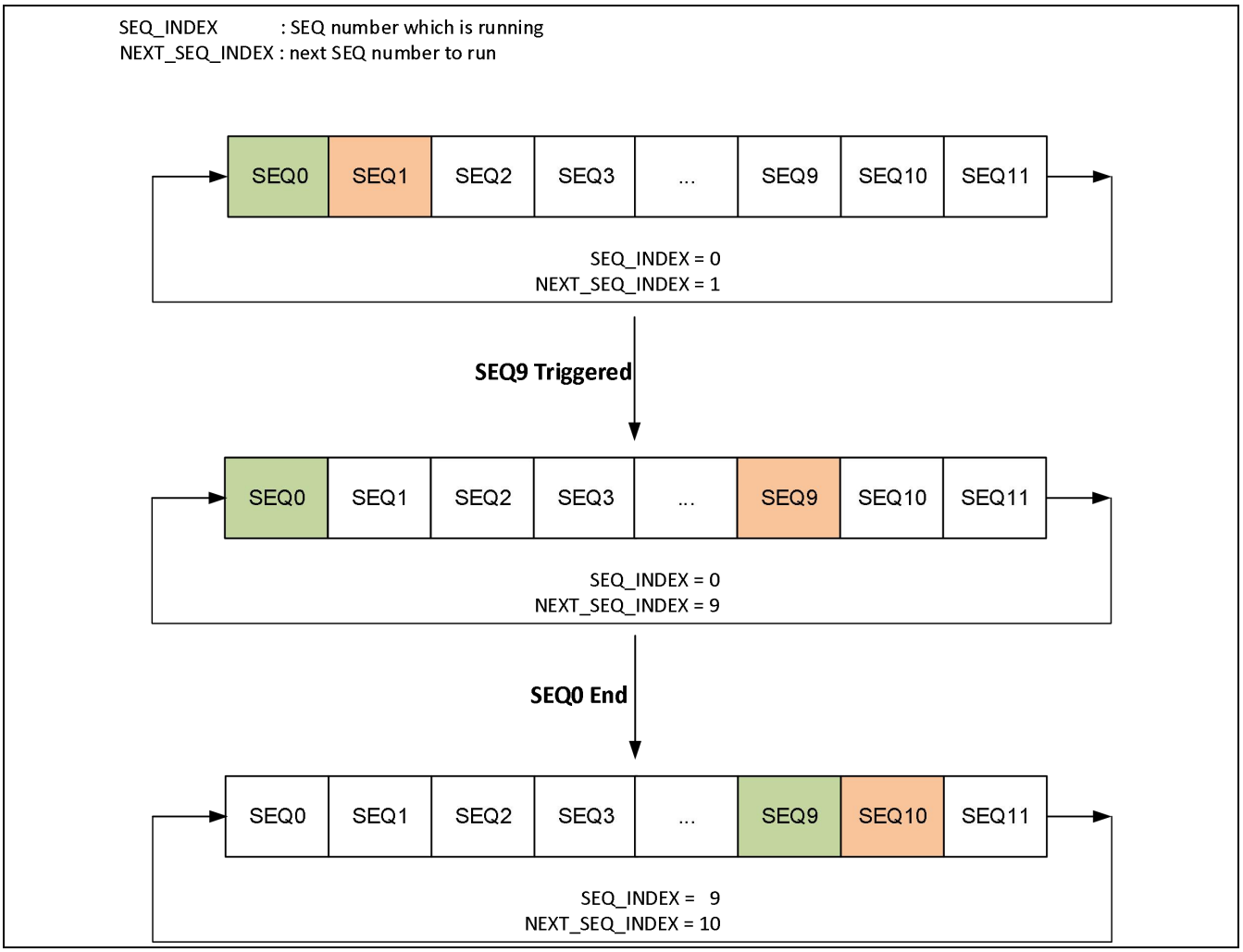


Figure 9-5 触发原理示意图

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。

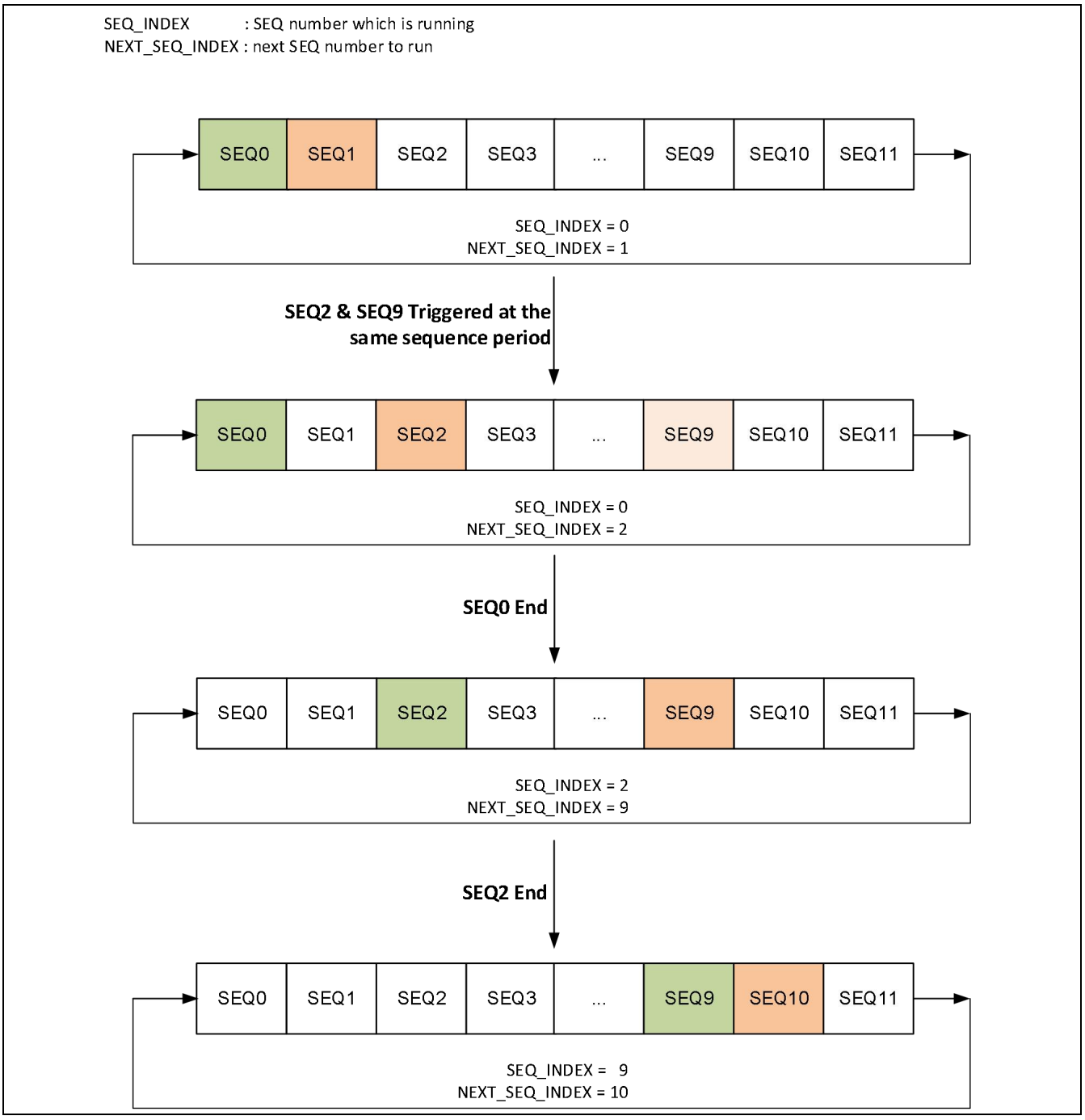


Figure 9-6 同时触发示意图

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。

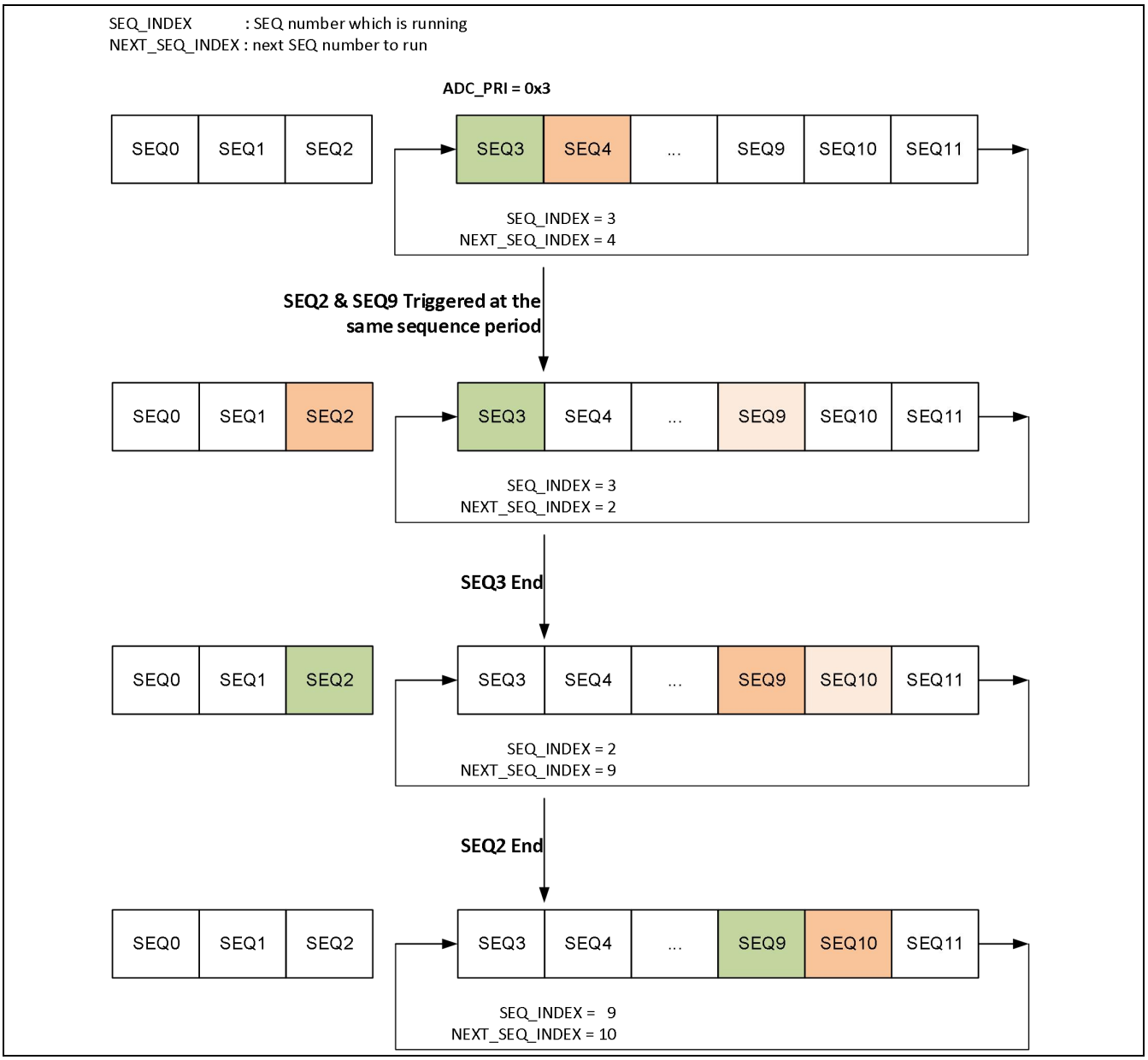


Figure 9-7 触发优先级示意图

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。

9.1.13 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

9.1.14 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。

9.1.15 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。

9.1.16 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)。

9.1.17 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)。

9.1.18 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)。

9.1.19 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TRGxSEL位用来选择作为输出的ADC事件，TRGxOE用来使能该事件的输出。

9.1.20 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。

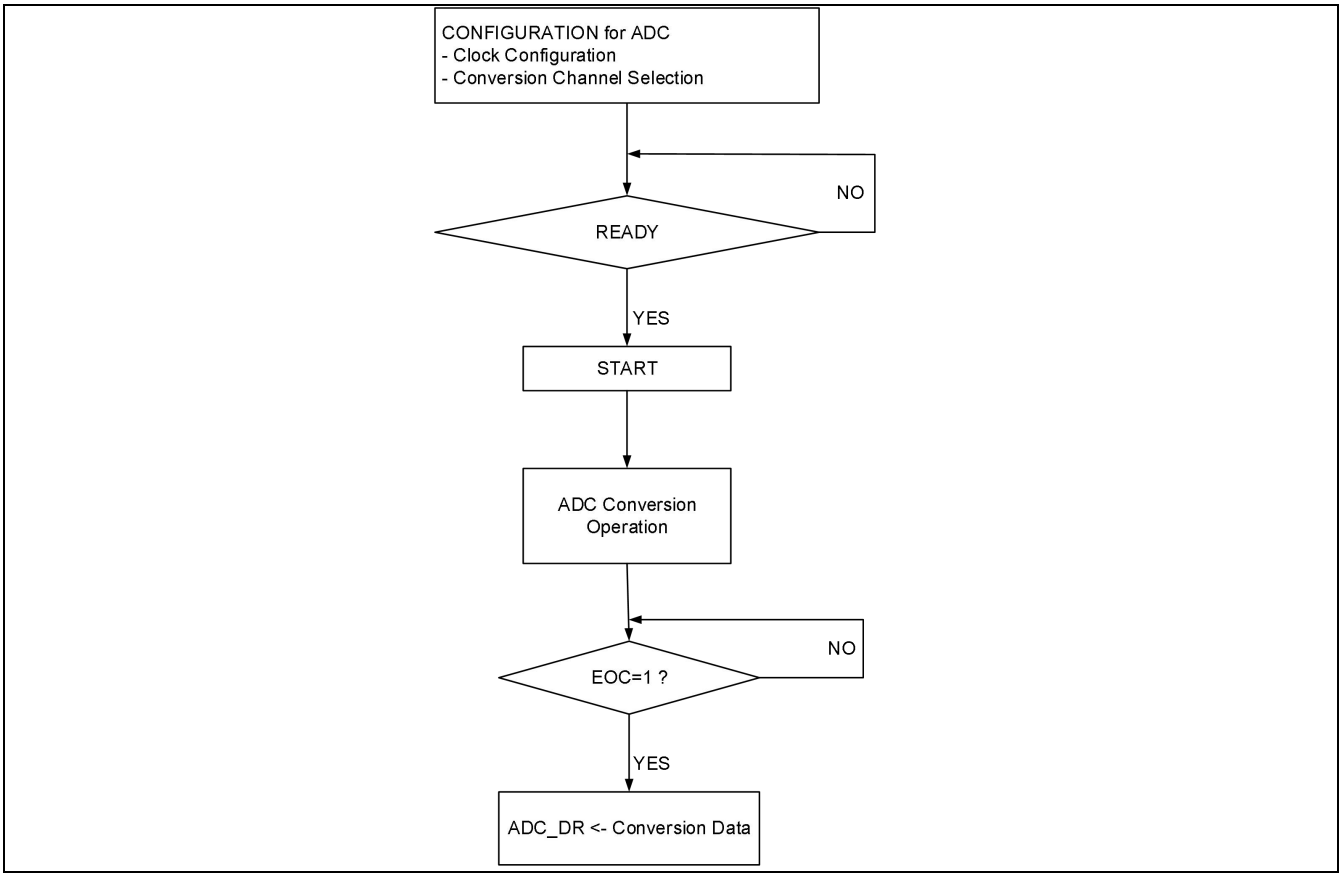


Figure 9-8 ADC工作流程图

9.1.21 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。

9.2 寄存器说明

9.2.1 寄存器表

Base Address of ADC0: 0x40030000

Register	Offset	Description	Reset Value
ADC_ECR	0x0000	时钟使能寄存器	0x00000000
ADC_DCR	0x0004	时钟禁止寄存器	0x00000000
ADC_PMSR	0x0008	功耗管理状态寄存器	0x2AAAAAA0
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	0x00000000
ADC_SR	0x0020	状态寄存器	0x00000000
ADC_IER	0x0024	中断使能寄存器	0x00000000
ADC_IDR	0x0028	中断禁止寄存器	0x00000000
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
ADC_EVTRG	0x0088	事件触发选择寄存器	0x00000000
ADC_DRx	0x0100 ~ 0x013C	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

9.2.2 ADC_ECR(时钟使能寄存器)

Address = Base Address+ 0x0000, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
DBGEN	[31]	W	DBGEN: ADC调试模式使能 0: 无效 1: 使能ADC调试模式
ADCCLKEN	[1]	W	ADC: ADC时钟使能 0: 无效 1: 使能ADC时钟

9.2.3 ADC_DCR(时钟禁止寄存器)

Address = Base Address+ 0x0004, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
DBGEN	[31]	W	DBGEN: ADC调试模式禁止 0: 无效 1: 禁止ADC调试模式
ADCCLKEN	[1]	W	ADC: ADC时钟禁止 0: 无效 1: 禁止ADC时钟

9.2.4 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	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
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
IPICODE	[29:4]	R	IPICODE[25:0] : IP识别码 模块的版本号，共26位
ADCCLKEN	[1]	R	ADC : ADC时钟状态 0: ADC时钟被禁止 1: ADC时钟被使能

9.2.5 ADC_CR(控制寄存器)

Address = Base Address+ 0x0010, Reset Value = 0x80040800

31	30	29	28	27	26	25	24	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_OUTEN	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	R	R	R	R	RW	RW	R	R	R	R	R	RW	RW	RW	R	R	R	R	R	R	RW	RW	RW	RW	W	W	W	W	W	W

Name	Bit	Type	Description
ACCURACY	[31]	RW	ACCURACY: ADC转换精度选择位 0: 12位 1: 10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。
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。
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。
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
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。
SWTRG	[5]	W	SWTRG : 软件触发 0: 无效 1: 触发转换序列
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
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)
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。
ADCEN	[1]	W	ADCEN : ADC模拟模块使能 0: 无效 1: 使能ADC模块
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。

9.2.6 ADC_MR(模式寄存器)

Address = Base Address+ 0x0014, Reset Value = 0x00000001

31	30	29	28	27	26	25	24	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	Bit	Type	Description
CONTCV	[31]	RW	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。
CMP_OS	[30]	RW	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。
NBRCMP1	[25:22]	RW	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
NBRCMP0	[19:16]	RW	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
NBRCH	[13:10]	RW	NBRCH[3:0]: 转换序列个数 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。
PRLVAL	[4:0]	RW	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

9.2.7 ADC_SHR(采样保持周期寄存器)

Address = Base Address+ 0x0018, Reset Value = 0x00000003

31	30	29	28	27	26	25	24	23	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	Bit	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。

9.2.8 ADC_CSR(状态清除寄存器)

Address = Base Address+ 0x001C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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[x](x=0~15)																RSVD								CMP1L	CMP1H	CMP0L	CMP0H	RSVD	OVR	READY	RSVD
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	R	R	R	R	R	R	R	R	W	W	W	W	R	W	W	R

Name	Bit	Type	Description
SEQ_END[x](x=0~15)	[31:16]	W	SEQ_END[x] : SEQx序列转换完成中断 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
CMP1H	[6]	W	CMP1H : 转换结果大于ADC_CMP1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
CMP0H	[4]	W	CMP0H : 转换结果大于ADC_CMP0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
READY	[1]	W	READY : ADC已准备好可以转换中断 0: 无效 1: 清除该中断

9.2.9 ADC_SR(状态寄存器)

Address = Base Address+ 0x0020, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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[x](x=0~15)																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	Bit	Type	Description
SEQ_END[x](x=0~15)	[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
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
CTCVS	[9]	R	CTCVS : 连续转换模式状态 0: 单次模式 1: 连续模式
ADCENS	[8]	R	ADCENS : ADC使能状态 0: ADC被禁止 1: ADC 被使能
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
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
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
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

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
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
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

注意

1．序列转换时，一旦有一个通道完成转换，EOC位即会置起

2．读取任何一个DR，都会清除EOC位及所有的SEQ_END位

3．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。

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

9.2.10 ADC_IER(中断使能寄存器)

Address = Base Address+ 0x0024, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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[x](x=0~15)																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	R	W	W	W

Name	Bit	Type	Description
SEQ_END[x](x=0~15)	[31:16]	W	SEQ_END[x] : SEQx序列转换完成中断 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
CMP1H	[6]	W	CMP1H : 转换结果高于ADC_CMP1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
CMP0H	[4]	W	CMP0H : 转换结果高于ADC_CMP0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
READY	[1]	W	READY : ADC READY中断 0: 无效 1: 使能该中断
EOC	[0]	W	EOC : 转换结束中断 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。			

9.2.11 ADC_IDR(中断禁止寄存器)

Address = Base Address+ 0x0028, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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[x](x=0~15)																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	R	W	W	W

Name	Bit	Type	Description
SEQ_END[x](x=0~15)	[31:16]	W	SEQ_END[x] : SEQx序列转换完成中断 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
CMP1H	[6]	W	CMP1H : 转换结果高于ADC_CMP1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
CMP0H	[4]	W	CMP0H : 转换结果高于ADC_CMP0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
READY	[1]	W	READY : ADC READY中断 0: 无效 1: 禁止该中断
EOC	[0]	W	EOC : 转换结束中断 0: 无效

			1: 禁止该中断

9.2.12 ADC_IMR(中断使能状态寄存器)

Address = Base Address+ 0x002C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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[x](x=0~15)																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	Bit	Type	Description
SEQ_END[x](x=0~15)	[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
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
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
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
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
OVR	[2]	R	OVR : 转换溢出中断 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
EOC	[0]	R	EOC : 转换结束中断 0: 该中断没有使能

			1: 该中断使能

9.2.13 ADC_SEQx(转换序列寄存器 x (x=0~15))

Address = Base Address+ 0x0030 ~ 0x006C, Reset Value = 0x0000009F

31	30	29	28	27	26	25	24	23	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	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
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) 或CMP触发 111 : ADC_SYNCIN5 (ETCB)
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
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。
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
AIN_SEL	[4:0]	RW	模拟输入通道选择 AIN_SEL值 输入选择 BIN DEC 00000 0 AIN0 00001 1 AIN1 00010 2 AIN2 …… …… 01110 14 AIN14 …… …… N/A 11100 28 INTVREF 11101 29 1/4VDD 11110 30 VSS 11111 31 N/A
• 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			

9.2.14 ADC_PRI(转换序列优先级寄存器)

Address = Base Address+ 0x0070, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	Bit	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

9.2.15 ADC_TDL0(触发延时寄存器 0)

Address = Base Address+ 0x0074, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	Bit	Type	Description
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
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
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
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。			

9.2.16 ADC_TDL1(触发延时寄存器 1)

Address = Base Address+ 0x0078, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								TRGIN5_TDL								TRGIN4_TDL								TRGIN3_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	Bit	Type	Description
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
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
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
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。			

9.2.17 ADC_SYNCR(触发同步控制寄存器)

Address = Base Address+ 0x007C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD										REARMx(x=0~5)						RSVD		OSTMDx(x=0~5)						RSVD		SYNCENx(x=0~5)					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
REARMx(x=0~5)	[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
OSTMDx(x=0~5)	[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。
SYNCENx(x=0~5)	[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 [2]：TC1脉冲匹配中断 [3]：EPWM触发 [4]：CMP触发

9.2.18 ADC_EVTRG(事件触发选择寄存器)

Address = Base Address+ 0x0088, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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
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
TRG1SEL	[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
TRG0SEL	[4:0]	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

9.2.19 ADC_DRx(转换结果寄存器 x (x=0~15))

Address = Base Address+ 0x0100 ~ 0x013C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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。

- 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

9.2.20 ADC_CMP0(比较数据 0 寄存器)

Address = Base Address+ 0x0140, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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。

9.2.21 ADC_CMP1(比较数据 1 寄存器)

Address = Base Address+ 0x0144, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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。

9.2.22 ADC_DRMASK(禁止转换结果更新寄存器)

Address = Base Address+ 0x0148, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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。

10 GPIO

10.1 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: 输入/输出端口, PA0.0 ~ PA0.11
- Port A1: 输入/输出端口, PA1.0 ~ PA1.5
- Port B0: 输入/输出端口, PB0.0 ~ PB0.7
- Port C0: 输入/输出端口, PC0.0 ~ PC0.3

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。

10.1.1 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
- 通讯口支持TTL电平输入配置

10.1.2 管脚描述

Table 10-1 管脚描述

Pin Name	Function	I/O Type	Active Level	Comments
PA0[11:0]	通用 I/O 口 A0	I/O	-	-
PA1[5:0]	通用 I/O 口 A1	I/O	-	-
PB[7:0]	通用 I/O 口 B0	I/O	-	-
PB[3:0]	通用 I/O 口 C0	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。

10.2 功能描述

10.2.1 电路图

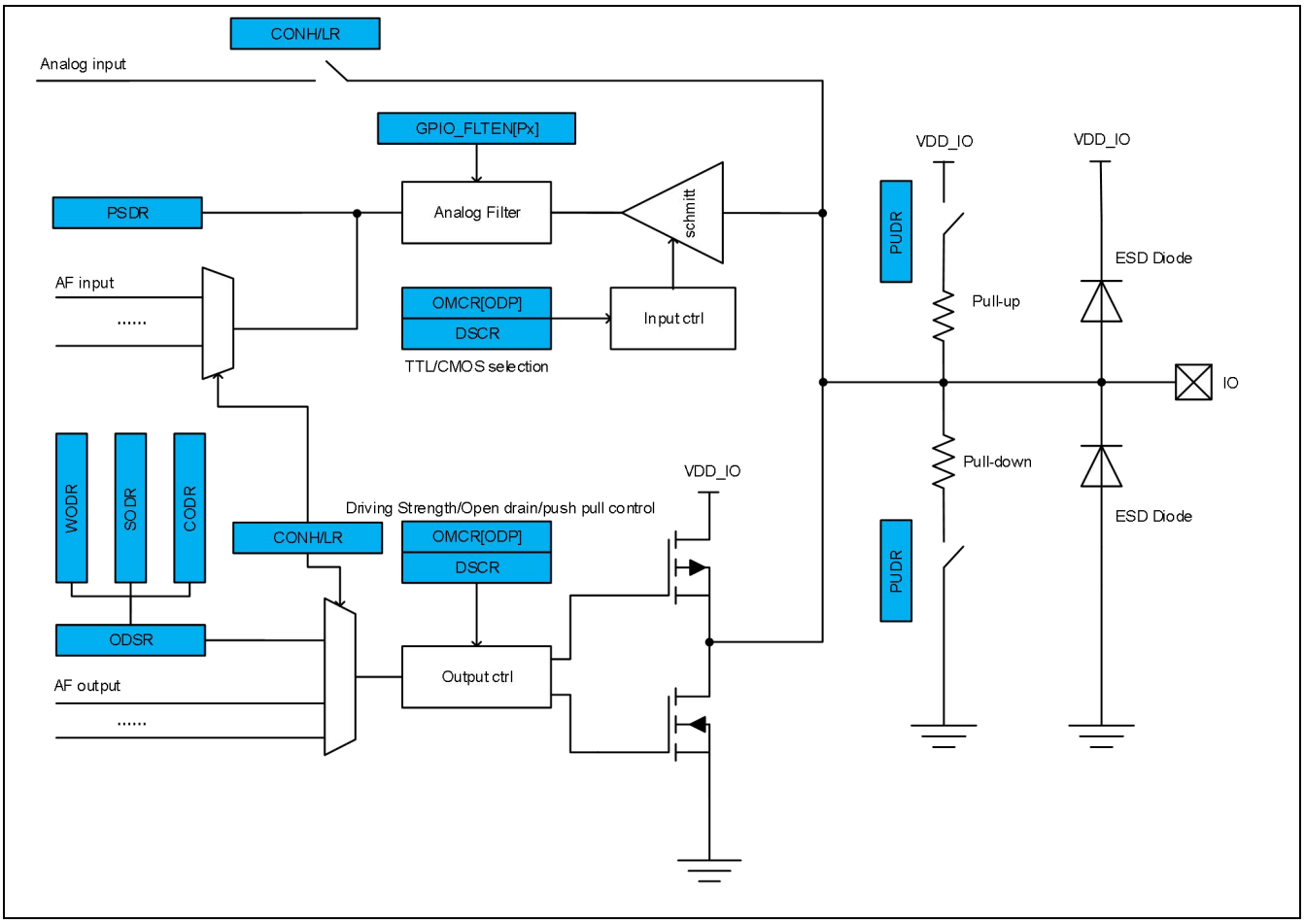


Figure 10-1 GPIO原理图

10.2.2 工作原理

10.2.2.1 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。

10.2.2.2 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。

10.2.3 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。

10.2.4 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。

10.2.5 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。

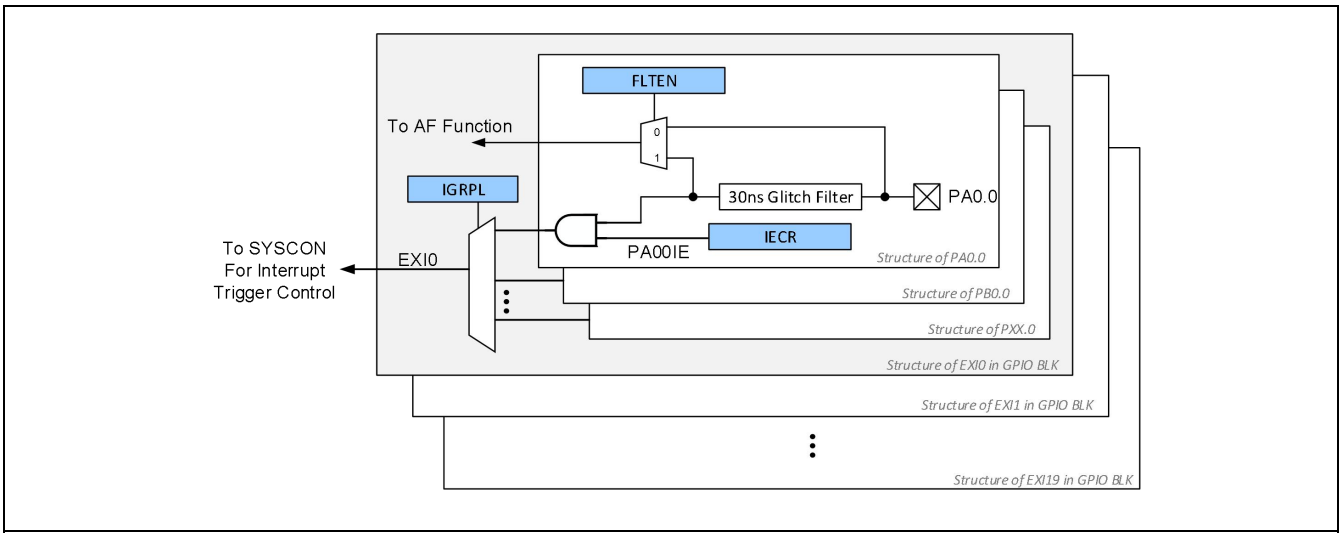


Figure 10- 2 GPIO外部中断原理图

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。

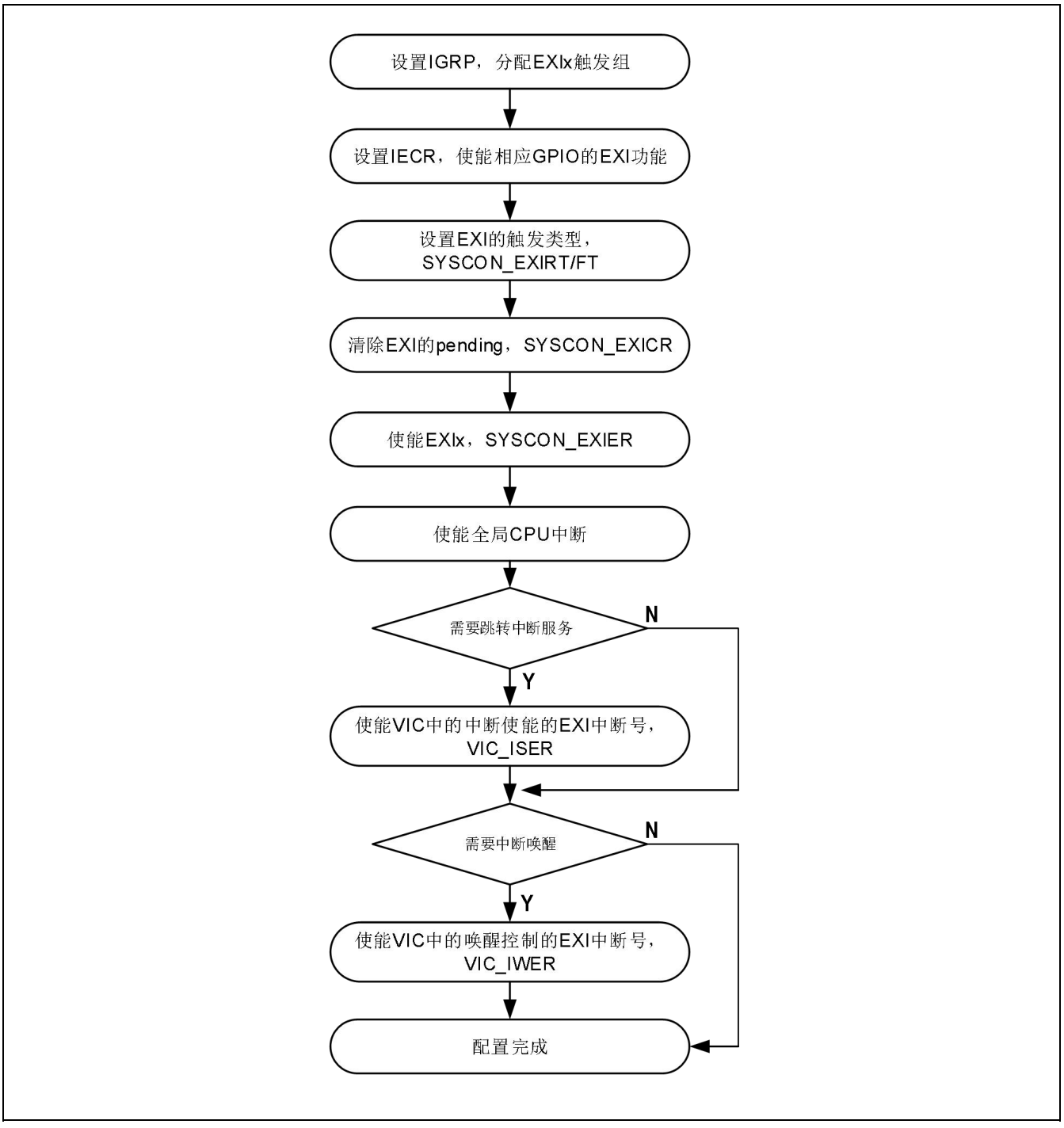


Figure 10- 3 GPIO外部中断配置流程

10.3 寄存器说明

10.3.1 寄存器表

Base Address of GPIOA0: 0x60000000

Base Address of GPIOA1: 0x60001000

Base Address of GPIOB0: 0x60002000

Base Address of GPIOC0: 0x60004000

Register	Offset	Description	Reset Value
GPIO_CONLR	0x00	低位控制寄存器	0x00000005
GPIO_CONHR	0x04	高位控制寄存器	0x00000000
GPIO_WODR	0x08	输出数据寄存器	0x00000000
GPIO_SODR	0x0C	输出置位寄存器	0x00000000
GPIO_CODR	0x10	输出清除寄存器	0x00000000
GPIO_ODSR	0x14	输出状态寄存器	0x00000000
GPIO_PSDR	0x18	管脚状态寄存器	0x00000000
GPIO_FLTEN	0x1C	输入信号滤波器使能控制寄存器	0x00000000
GPIO_PUDR	0x20	上拉/下拉配置寄存器	0x00000000
GPIO_DSCR	0x24	驱动强度配置寄存器	0x00000000
GPIO_OMCR	0x28	输出模式配置寄存器	0x00000000
GPIO_IECR	0x2C	外部中断使能寄存器	0x00000000
GPIO_IEER	0x30	外部中断使能设置寄存器	0x00000000
GPIO_IEDR	0x34	外部中断使能清除寄存器	0x00000000
Base Address of GPIO_IGRP: 0x6000F000			
GPIO_IGRPL	0x00	外部中断组配置寄存器	0x00000000
GPIO_IGRPH	0x04	外部中断组配置寄存器	0x00000000
GPIO_IGREX	0x08	外部中断组扩展配置寄存器	0x00000000
GPIO_CLKEN	0x0C	GPIO组时钟使能控制寄存器	0x00000000

（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。

10.3.2 GPIO_CONLR(低位控制寄存器)

Address = Base Address+ 0x00, Reset Value = 0x00000005

31	30	29	28	27	26	25	24	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	1	0	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
P7	[31:28]	RW	IO管脚7的模式配置
P6	[27:24]	RW	IO管脚6的模式配置
P5	[23:20]	RW	IO管脚5的模式配置
P4	[19:16]	RW	IO管脚4的模式配置
P3	[15:12]	RW	IO管脚3的模式配置
P2	[11:8]	RW	IO管脚2的模式配置
P1	[7:4]	RW	IO管脚1的模式配置
P0	[3:0]	RW	IO管脚0的模式配置

10.3.3 GPIO_CONHR(高位控制寄存器)

Address = Base Address+ 0x04, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
P15	[31:28]	RW	IO管脚15的模式配置
P14	[27:24]	RW	IO管脚14的模式配置
P13	[23:20]	RW	IO管脚13的模式配置
P12	[19:16]	RW	IO管脚12的模式配置
P11	[15:12]	RW	IO管脚11的模式配置
P10	[11:8]	RW	IO管脚10的模式配置
P9	[7:4]	RW	IO管脚9的模式配置
P8	[3:0]	RW	IO管脚8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（参见管脚配置）。

10.3.4 GPIO_WODR(输出数据寄存器)

Address = Base Address+ 0x08, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
P15	[15]	W	
P14	[14]	W	
P13	[13]	W	
P12	[12]	W	
P11	[11]	W	
P10	[10]	W	
P9	[9]	W	
P8	[8]	W	
P7	[7]	W	
P6	[6]	W	
P5	[5]	W	
P4	[4]	W	
P3	[3]	W	
P2	[2]	W	
P1	[1]	W	
P0	[0]	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。

10.3.5 GPIO_SODR(输出置位寄存器)

Address = Base Address+ 0x0C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	

Name	Bit	Type	Description
P15	[15]	W	
P14	[14]	W	
P13	[13]	W	
P12	[12]	W	
P11	[11]	W	
P10	[10]	W	
P9	[9]	W	
P8	[8]	W	
P7	[7]	W	
P6	[6]	W	
P5	[5]	W	
P4	[4]	W	
P3	[3]	W	
P2	[2]	W	
P1	[1]	W	
P0	[0]		

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。

10.3.6 GPIO_CODR(输出清除寄存器)

Address = Base Address+ 0x10, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
P15	[15]	W	
P14	[14]	W	
P13	[13]	W	
P12	[12]	W	
P11	[11]	W	
P10	[10]	W	
P9	[9]	W	
P8	[8]	W	
P7	[7]	W	
P6	[6]	W	
P5	[5]	W	
P4	[4]	W	
P3	[3]	W	
P2	[2]	W	
P1	[1]	W	
P0	[0]	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。

10.3.7 GPIO_ODSR(输出状态寄存器)

Address = Base Address+ 0x14, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
P15	[15]	R	
P14	[14]	R	
P13	[13]	R	
P12	[12]	R	
P11	[11]	R	
P10	[10]	R	
P9	[9]	R	
P8	[8]	R	
P7	[7]	R	
P6	[6]	R	
P5	[5]	R	
P4	[4]	R	
P3	[3]	R	
P2	[2]	R	
P1	[1]	R	
P0	[0]	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。

10.3.8 GPIO_PSDR(管脚状态寄存器)

Address = Base Address+ 0x18, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
P15	[15]	R	
P14	[14]	R	
P13	[13]	R	
P12	[12]	R	
P11	[11]	R	
P10	[10]	R	
P9	[9]	R	
P8	[8]	R	
P7	[7]	R	
P6	[6]	R	
P5	[5]	R	
P4	[4]	R	
P3	[3]	R	
P2	[2]	R	
P1	[1]	R	
P0	[0]	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。			

10.3.9 GPIO_FLTEN(输入信号滤波器使能控制寄存器)

Address = Base Address+ 0x1C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
P15	[15]	RW	
P14	[14]	RW	
P13	[13]	RW	
P12	[12]	RW	
P11	[11]	RW	
P10	[10]	RW	
P9	[9]	RW	
P8	[8]	RW	
P7	[7]	RW	
P6	[6]	RW	
P5	[5]	RW	
P4	[4]	RW	
P3	[3]	RW	
P2	[2]	RW	
P1	[1]	RW	
P0	[0]	RW	
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。			

10.3.10 GPIO_PUDR(上拉/下拉配置寄存器)

Address = Base Address+ 0x20, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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	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 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。			

10.3.11 GPIO_DSCR(驱动强度配置寄存器)

Address = Base Address+ 0x24, Reset Value = 0x00000000

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

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：仅在做输出时有效，用于控制驱动能力。（0-弱驱，1-强驱）

BIT1：做输出时用于控制驱动斜率。（0-慢速，1-快速）

做输入时用于控制逻辑电平判断阈值。（0-CMOS输入，1-TTL输入）OMCR里可以选择TTL电平。。

10.3.12 GPIO_OMCR(输出模式配置寄存器)

Address = Base Address+ 0x28, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
CCM15	[31]	RW	
CCM14	[30]	RW	
CCM13	[29]	RW	
CCM12	[28]	RW	
CCM11	[27]	RW	
CCM10	[26]	RW	
CCM9	[25]	RW	
CCM8	[24]	RW	
CCM7	[23]	RW	
CCM6	[22]	RW	
CCM5	[21]	RW	
CCM4	[20]	RW	
CCM3	[19]	RW	
CCM2	[18]	RW	
CCM1	[17]	RW	
CCM0	[16]	RW	
ODP15	[15]	RW	
ODP14	[14]	RW	
ODP13	[13]	RW	
ODP12	[12]	RW	
ODP11	[11]	RW	
ODP10	[10]	RW	
ODP9	[9]	RW	
ODP8	[8]	RW	
ODP7	[7]	RW	
ODP6	[6]	RW	
ODP5	[5]	RW	

ODP4	[4]	RW	
ODP3	[3]	RW	
ODP2	[2]	RW	
ODP1	[1]	RW	
ODP0	[0]	RW	

ODPx 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 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。

10.3.13 GPIO_IECR(外部中断使能寄存器)

Address = Base Address+ 0x2C, Reset Value = 0x00000000

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

10.3.14 GPIO_IEER(外部中断使能设置寄存器)

Address = Base Address+ 0x30, Reset Value = 0x00000000

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

10.3.15 GPIO_IEDR(外部中断使能清除寄存器)

Address = Base Address+ 0x34, Reset Value = 0x00000000

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

10.3.16 GPIO_IGRPL(外部中断组配置寄存器)

Address = Base Address+ x00, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	选择外部中断组7
GRP6	[26:24]	RW	选择外部中断组6
GRP5	[22:20]	RW	选择外部中断组5
GRP4	[18:16]	RW	选择外部中断组4
GRP3	[14:12]	RW	选择外部中断组3
GRP2	[10:8]	RW	选择外部中断组2
GRP1	[6:4]	RW	选择外部中断组1
GRP0	[2:0]	RW	选择外部中断组0
0000: GPIOA0.x 被选中 0001: GPIOA1.x 被选中 0010: GPIOB0.x 被选中 0011: 保留 0100: GPIOC0.x 被选中 Other: 保留 ‘x’ 表示组数			

10.3.17 GPIO_IGRPH(外部中断组配置寄存器)

Address = Base Address+ x04, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	选择外部中断组15
GRP14	[26:24]	RW	选择外部中断组14
GRP13	[22:20]	RW	选择外部中断组13
GRP12	[18:16]	RW	选择外部中断组12
GRP11	[14:12]	RW	选择外部中断组11
GRP10	[10:8]	RW	选择外部中断组10
GRP9	[6:4]	RW	选择外部中断组9
GRP8	[2:0]	RW	选择外部中断组8
0000: GPIOA0.x 被选中 0001: GPIOA1.x 被选中 0010: GPIOB0.x 被选中 0011: 保留 0100: GPIOC0.x 被选中 Other: 保留 ‘x’ 表示组数			

10.3.18 GPIO_IGREX(外部中断组扩展配置寄存器)

Address = Base Address+ x08, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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				GRP17				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
GRP19	[15:12]	RW	选择外部中断组19 配置同GPR18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 4h：GPIOC0.0 被选中 5h：GPIOC0.1 被选中 6h：GPIOC0.2 被选中 7h：GPIOC0.3 被选中 其它：保留
GRP17	[7:4]	RW	选择外部中断组17 配置同GPR16控制位
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 8h：GPIOB0.0 被选中 9h：GPIOB0.1 被选中 Ah：GPIOB0.2 被选中 Bh：GPIOB0.3 被选中 Ch：GPIOC0.0 被选中

			Dh：GPIOC0.1 被选中 Eh：GPIOC0.2 被选中 Fh：GPIOC0.3 被选中

10.3.19 GPIO_CLKEN(GPIO组时钟使能控制寄存器)

Address = Base Address+ x0C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																											CLK_C0	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	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
CLK_C0	[4]	RW	
CLK_B0	[2]	RW	
CLK_A1	[1]	RW	
CLK_A0	[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			

11 基本计数器 (Basic Timer)

11.1 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。

11.1.1 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。

11.1.2 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

11.2 功能描述

11.2.1 模块框图

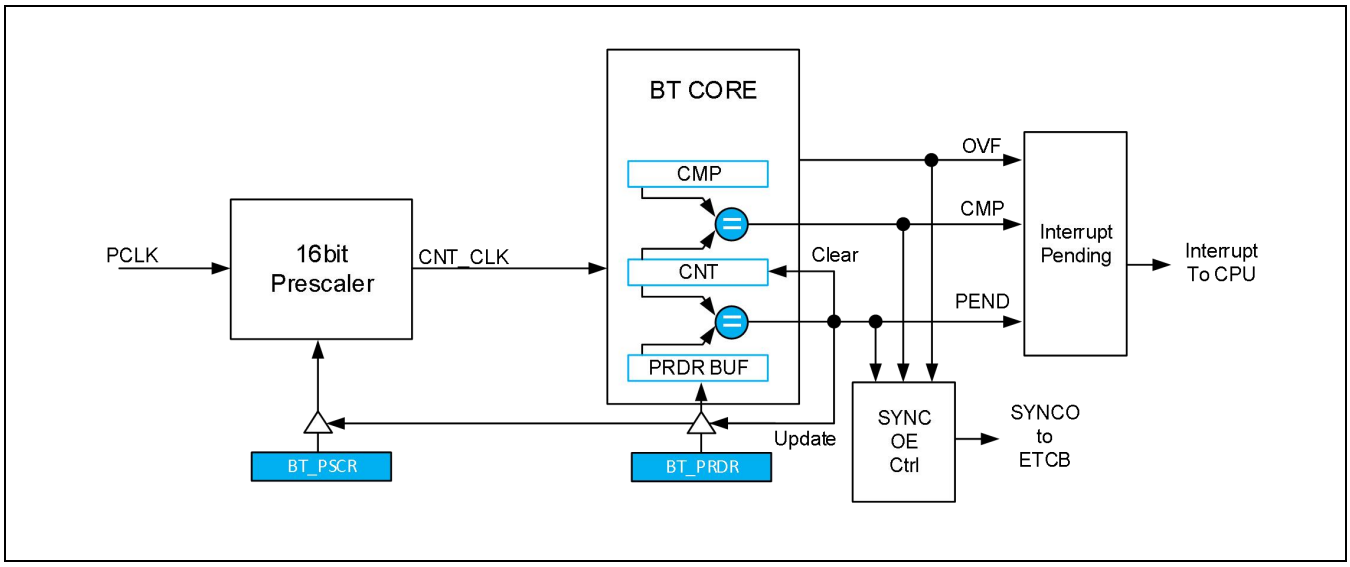


Figure 11-1 模块结构示意图

11.2.2 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）。

11.2.3 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。

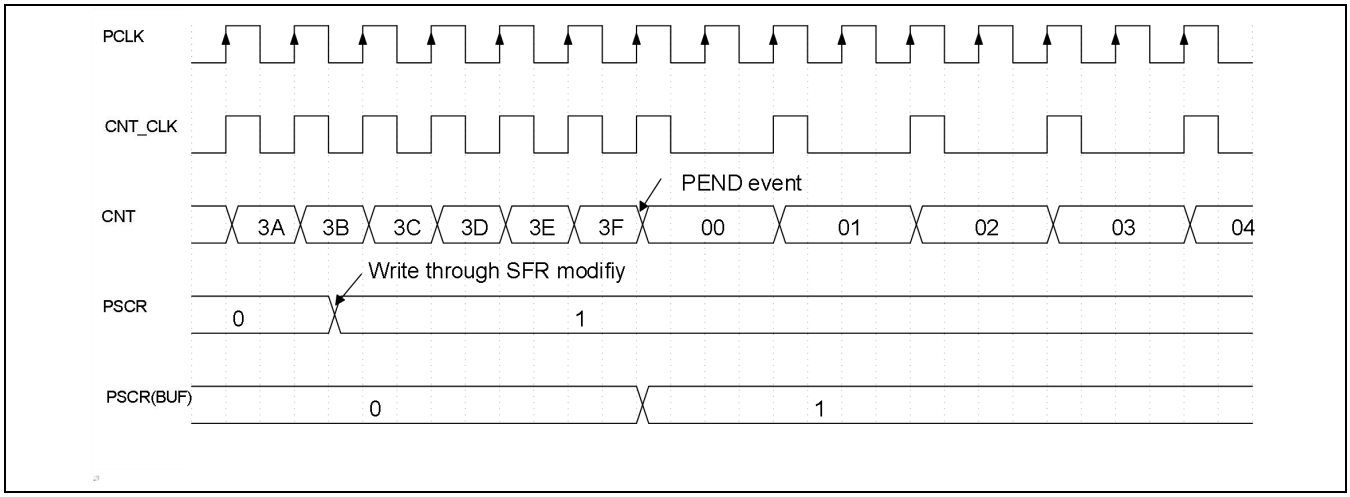


Figure 11-2 BT 周期计数模式

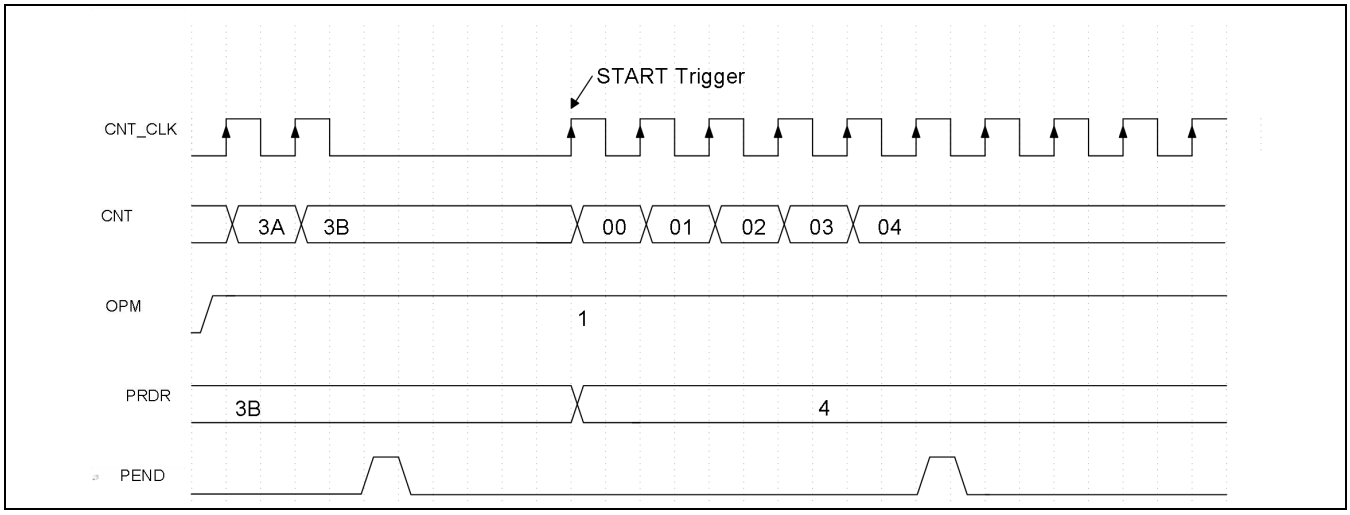


Figure 11-3 BT One Pulse 计数模式

11.2.4 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。

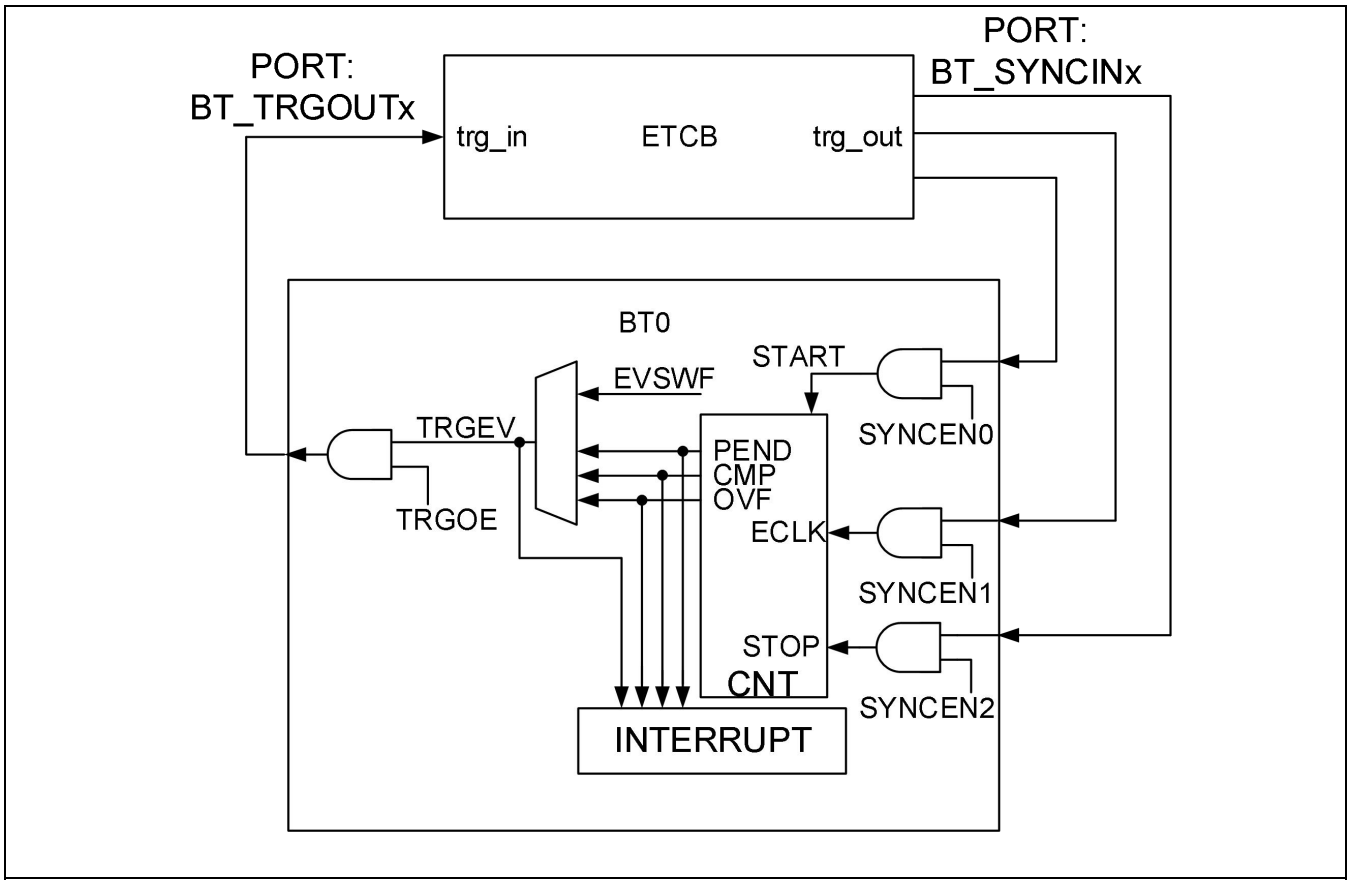


Figure 11-4 同步触发原理

11.2.4.1 同步触发(输入)

Basic Timer有三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2可以触发Basic Timer的STOP控制。

SYNC PORT2：同步输入端口1可以触发Basic Timer的计数值增加一拍。

11.2.4.2 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。

11.2.5 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。

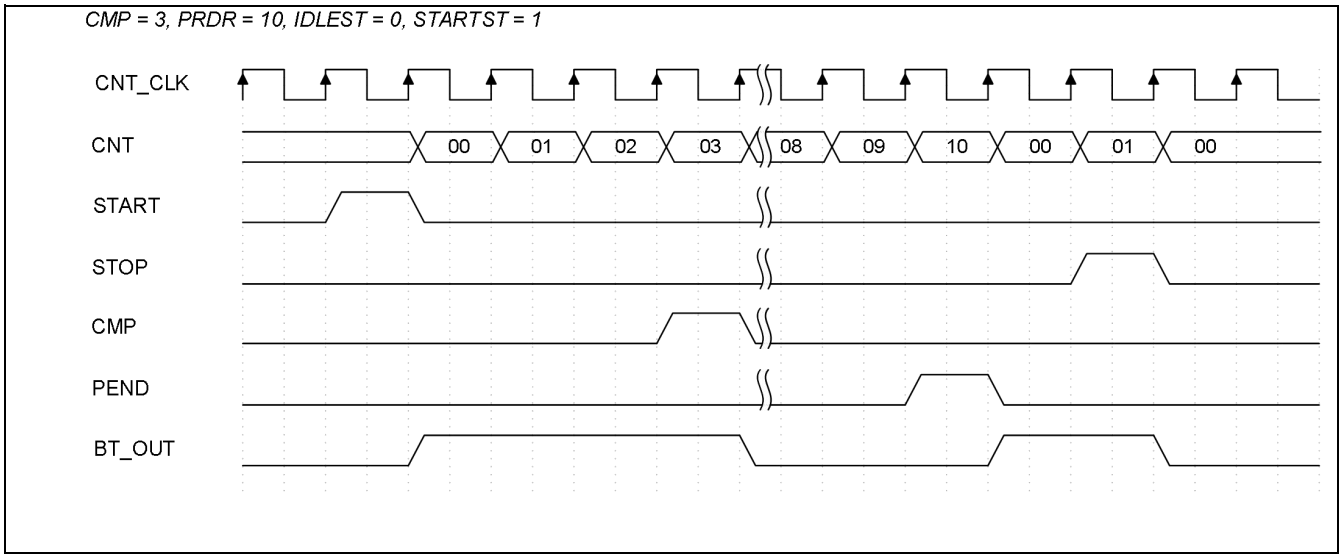


Figure 11-5 BT 输出波形时序图

11.2.6 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。

11.3 寄存器说明

11.3.1 寄存器表

Base Address of BT0: 0x40061000

Base Address of BT1: 0x40062000

Base Address of BT2: 0x40063000

Base Address of BT3: 0x40064000

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

11.3.2 BT_RSSR(Reset/Start Control Register)

Address = Base Address+ 0x000, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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。
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。

11.3.3 BT_CR(General Control Register)

Address = Base Address+ 0x004, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	REARMx			RSVD	OSTMDx			RSVD		AREARM1		AREARM0		RSVD	CNTRLD	SYNCMD	RSVD				SYNCEN2	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	RW	RW	RW	R	RW	RW	RW	R	R	RW	RW	RW	RW	R	RW	RW	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
REARMx	[30:28]	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
OSTMDx	[26:24]	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。
AREARM1	[21:20]	RW	硬件自动REARM控制位。 0：禁止硬件自动REARM 1：使能硬件自动REARM BIT0：使能时，计数器周期结束时，自动REARM BIT1：使能时，SYNC0事件，自动REARM
AREARM0	[19:18]	RW	硬件自动REARM控制位。 0：禁止硬件自动REARM 1：使能硬件自动REARM BIT0：使能时，计数器周期结束时，自动REARM BIT1：使能时，SYNC1 事件，自动REARM
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。

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。
SYNCEN2	[10]	RW	外部同步触发输入PORT2使能控制。 0h：禁止外部触发 1h：使能外部触发，触发BT的计数值增加一拍。
SYNCEN1	[9]	RW	外部同步触发输入PORT1使能控制。 0h：禁止外部触发 1h：使能外部触发，触发BT的STOP
SYNCEN0	[8]	RW	外部同步触发输入PORT0使能控制。 0h：禁止外部触发 1h：使能外部触发，触发BT的START
STARTST	[7]	RW	BT开始计数时，BT_OUT状态设置。 0：低电平 1：高电平
IDLEST	[6]	RW	BT停止计数时，BT_OUT状态设置。 0：低电平 1：高电平
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
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
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
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
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
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。

11.3.4 BT_PSCR(Counter Clock Prescaler Register)

Address = Base Address+ 0x008, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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)

11.3.5 BT_PRDR(Period Register)

Address = Base Address+ 0x00C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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	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
W	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
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。
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。

11.3.6 BT_CMP(Compare Data)

Address = Base Address+ 0x010, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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。

11.3.7 BT_CNT(Counter Register)

Address = Base Address+ 0x014, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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。

11.3.8 BT_EVTRG(Event Generation Control Register)

Address = Base Address+ 0x018, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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。
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

11.3.9 BT_EVSWF(Event Counter Software Trigger Register)

Address = Base Address+ 0x024, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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。

11.3.10 BT_RISR(Raw Interrupt Status Register)

Address = Base Address+ 0x028, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
TRGEV	[3]	R	事件触发中断请求原始标志状态。
OVF	[2]	R	OVF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。
PEND	[0]	R	PEND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			

11.3.11 BT_IMCR(Interrupt Masking Control Register)

Address = Base Address+ 0x02C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																												EVTRG	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
EVTRG	[3]	RW	事件触发中断中断使能控制位。
OVF	[2]	RW	OVF中断使能控制位。
CMP	[1]	RW	CMP Match中断使能控制位
PEND	[0]	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			

11.3.12 BT_MISR(Masked Interrupt Status Register)

Address = Base Address+ 0x030, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																												EVTRG	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
EVTRG	[3]	R	事件触发中断请求标志状态。
OVF	[2]	R	OVF中断请求标志状态。
CMP	[1]	R	CMP Match中断请求标志状态。
PEND	[0]	R	PEND周期结束中断请求标志状态。
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			

11.3.13 BT_ICR(Interrupt Clear Register)

Address = Base Address+ 0x034, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																												EVTRG	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
EVTRG	[3]	W	清除事件触发中断原始中断状态位。
OVF	[2]	W	清除OVF原始中断状态位。
CMP	[1]	W	清除CMP Match原始中断状态位。
PEND	[0]	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’			

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(捕获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 4 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 12-1 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

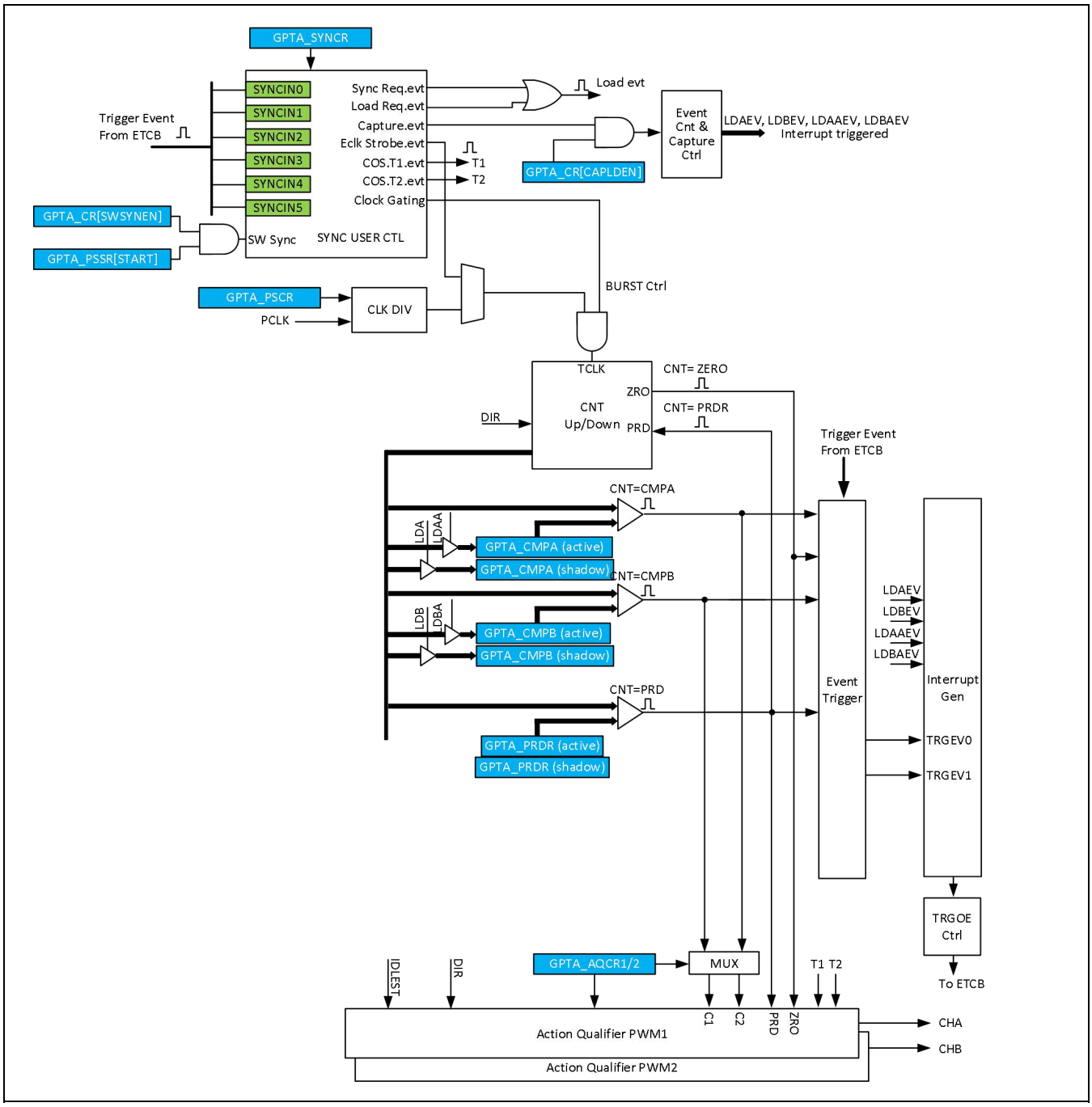


Figure 12-1 模块结构示意图

12.3 基本功能描述

一个完整的GPTA模块包含两个输入/输出通道。多个GPTA或多个EPT可以通过ETCB来实现同步工作。如果芯片内包含多个GPTA，我们以数字后缀来作不同的区分，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、捕获控制模块、事件触发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。在在波形输出模式下，PWM信号通过GPTA_CHA和GPTA_CHB输出；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。

12.3.1 时钟源

12.3.1.1 概述

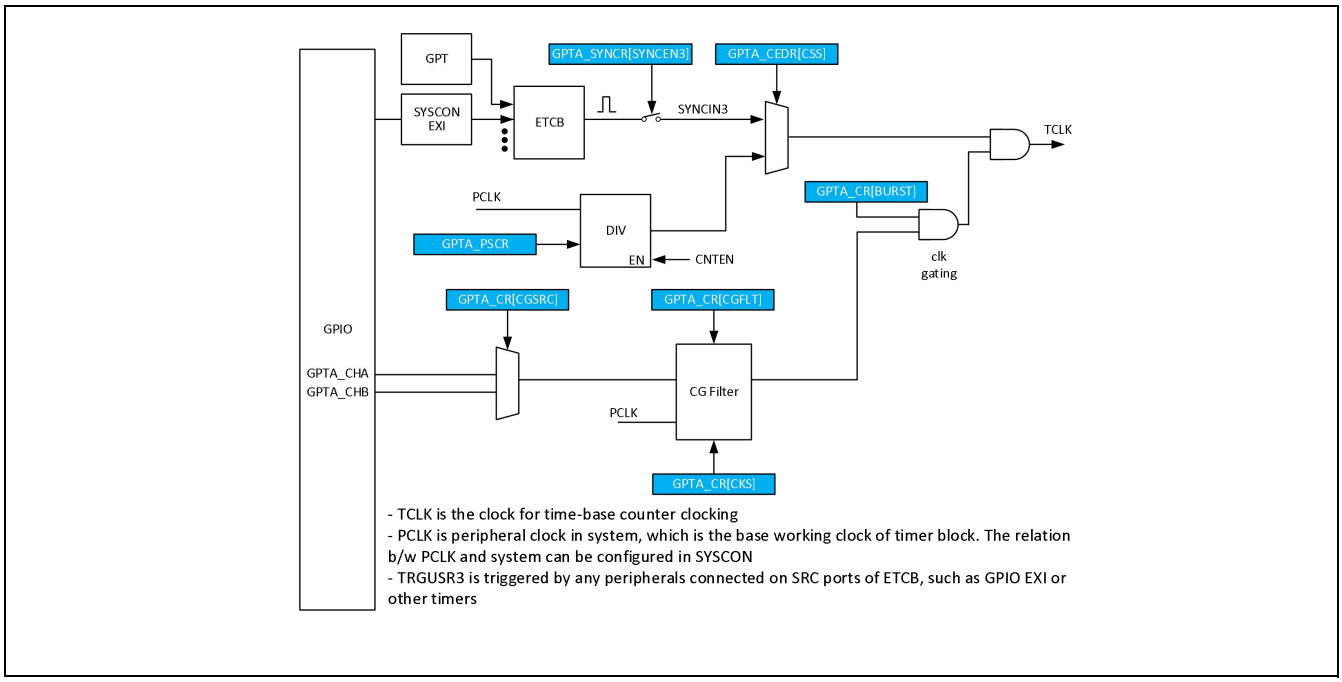


Figure 12-2 时钟控制模块

增强型通用定时器GPTA可以工作在PCLK下，通过GPTA_PSCR[PSC]设置分频系数；也可以工作在外部时钟信号下。需要注意的是，外部时钟信号的频率必须低于1/2PCLK，以保证被PCLK同步。

12.3.1.2 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。

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0或PRDR时(可通过GPTA_CR[PSCLD]设置载入的条件)，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

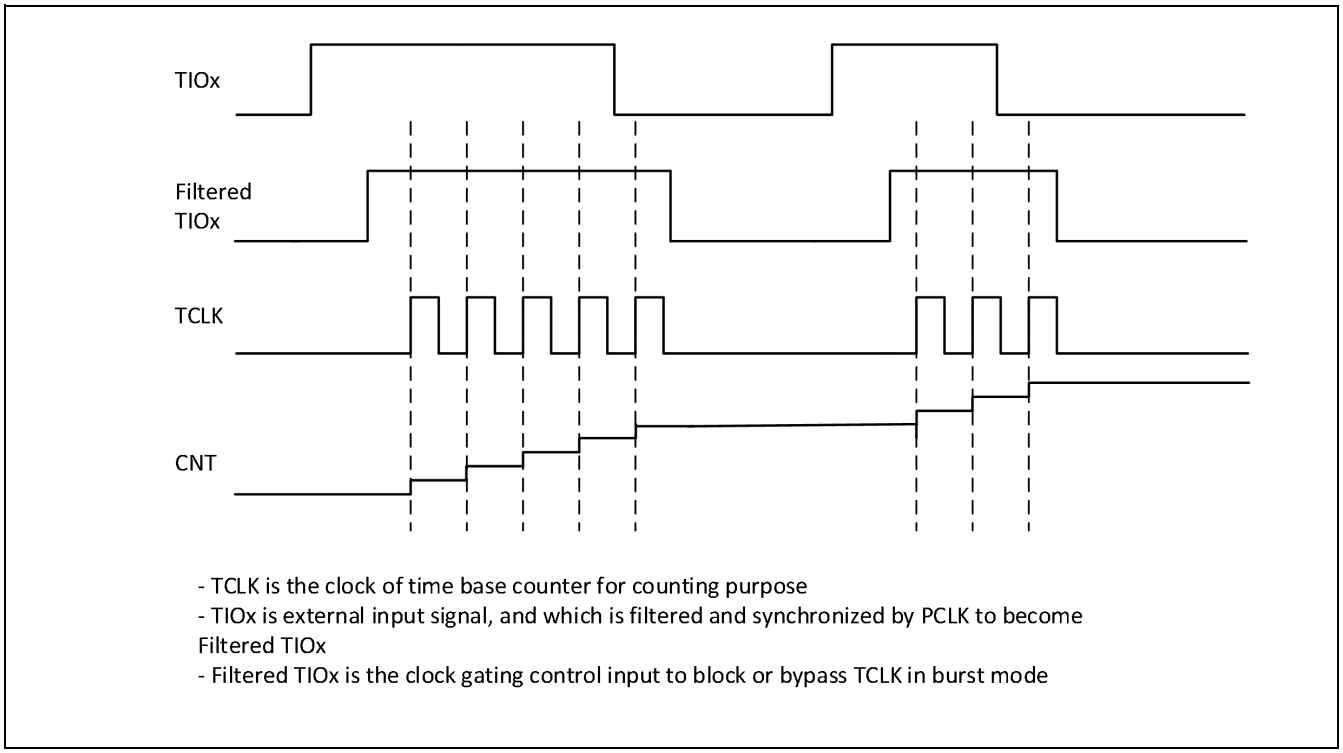


Figure 12- 3 群脉冲时钟模式时序

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。

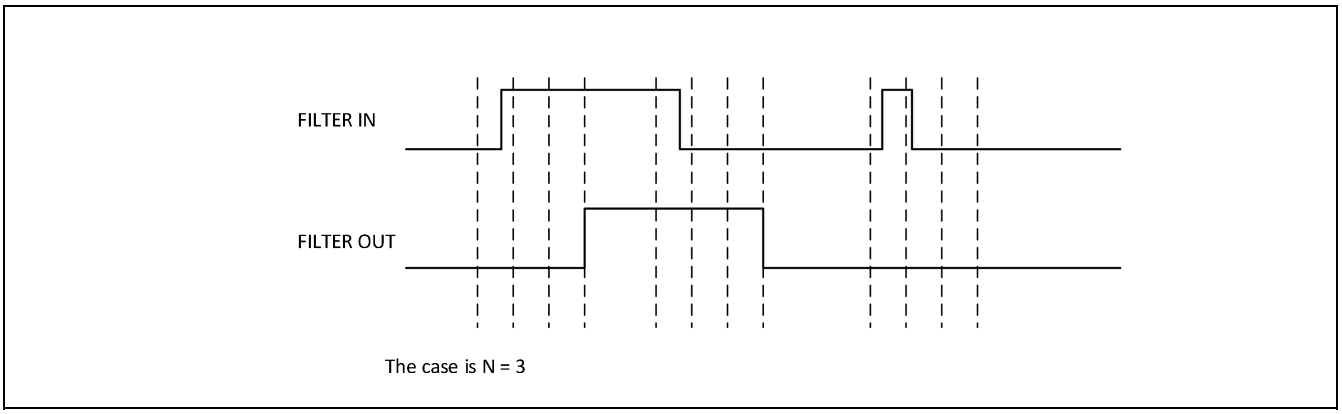


Figure 12- 4 CG Filter 数字滤波器的原理

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。

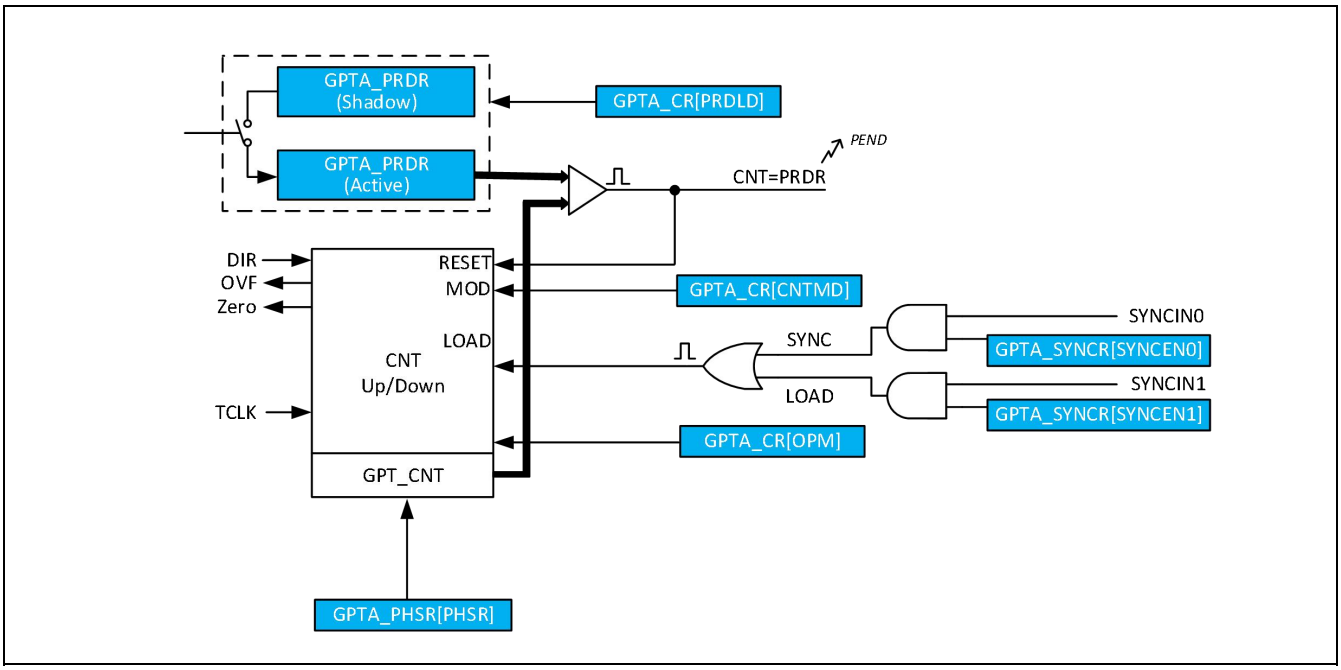


Figure 12-5 计数器时基模块

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。

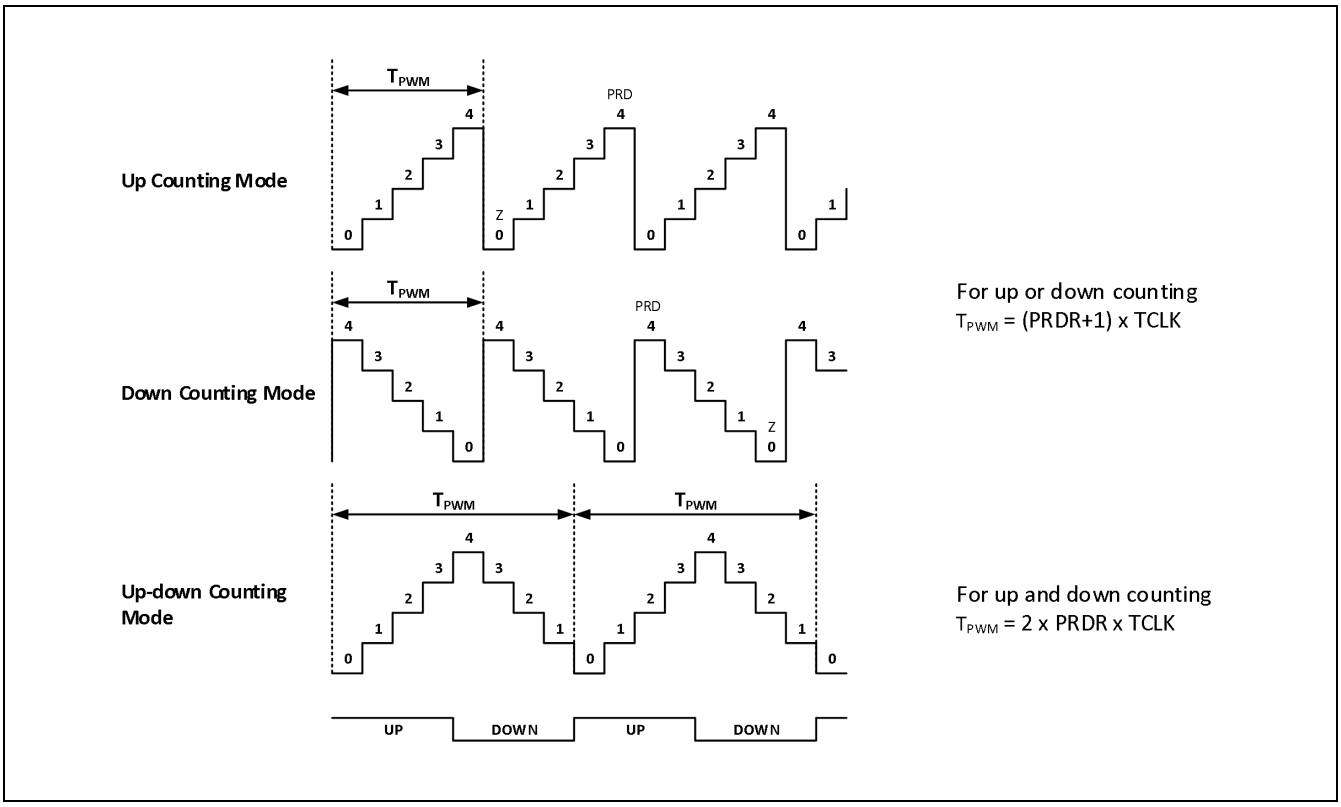


Figure 12-6 计数器工作模式

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[PEND]置高，同时可以产生一个中断(通过中断使能控制寄存器GPTA_IMCR[PEND]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。
- 计数周期结束：当计数周期结束时，根据当前计数模式，计数器的值将被重置到0x0000或者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(GPTA_CR[PRDLD]=11b)，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'11'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。当计数周期结束时(产生PEND事件)，或者SYNC1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。用户可以通过GPTA_CR[PRDLD]控制位进行配置。

- **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。

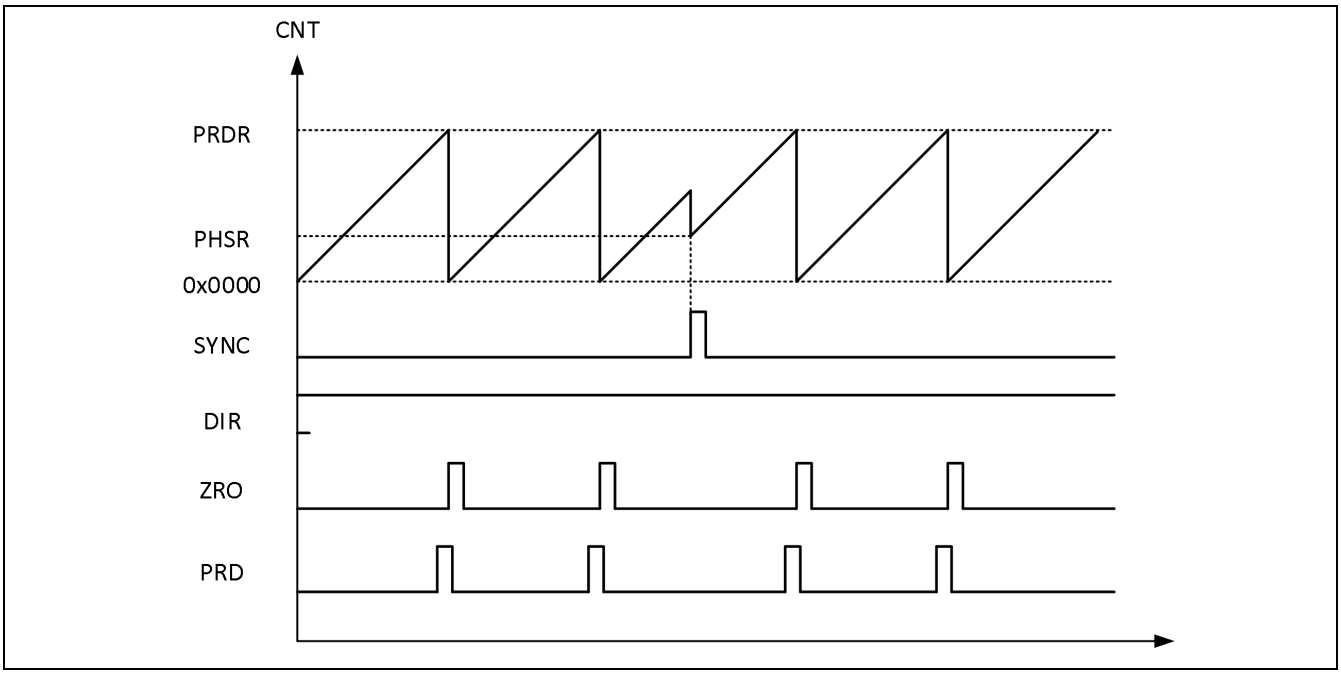


Figure 12-7 递增工作模式

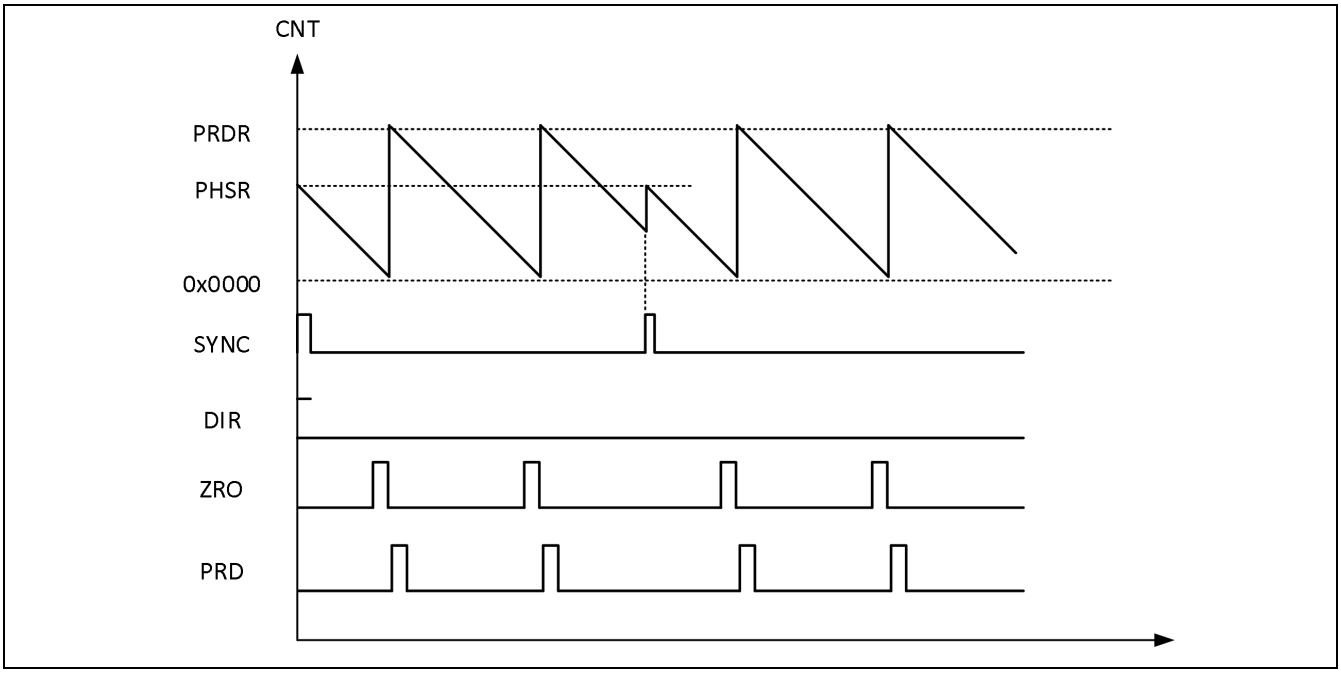


Figure 12-8 递减工作模式

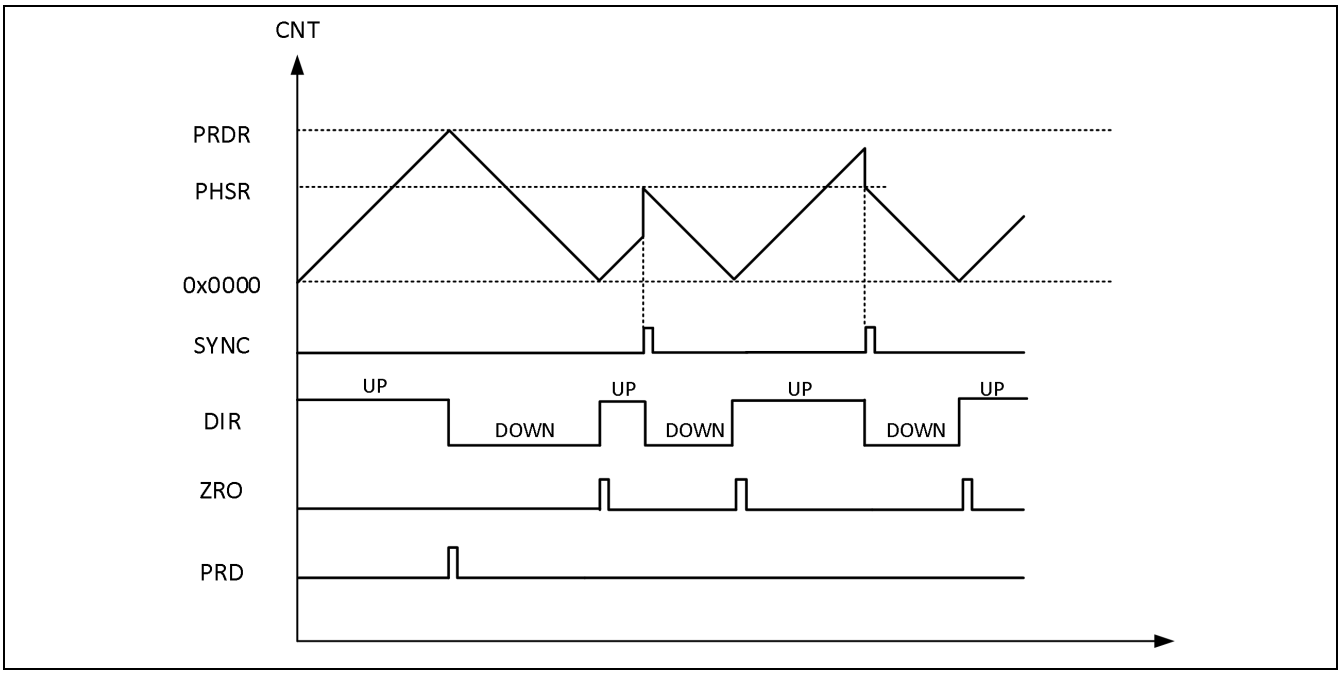


Figure 12- 9 递增递减工作模式

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[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[OSTMD]=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。

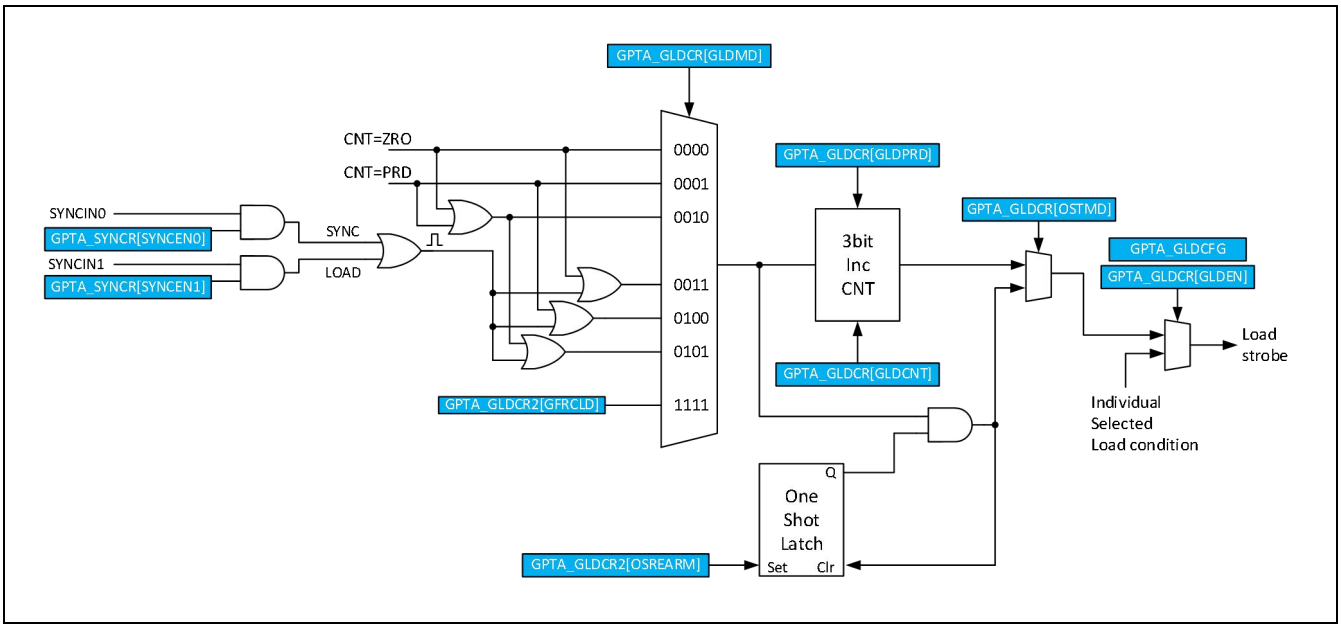


Figure 12-10 全局载入控制

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模块实时比较当前计数器的计数值和比较值寄存器（CMPA、CMPB）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

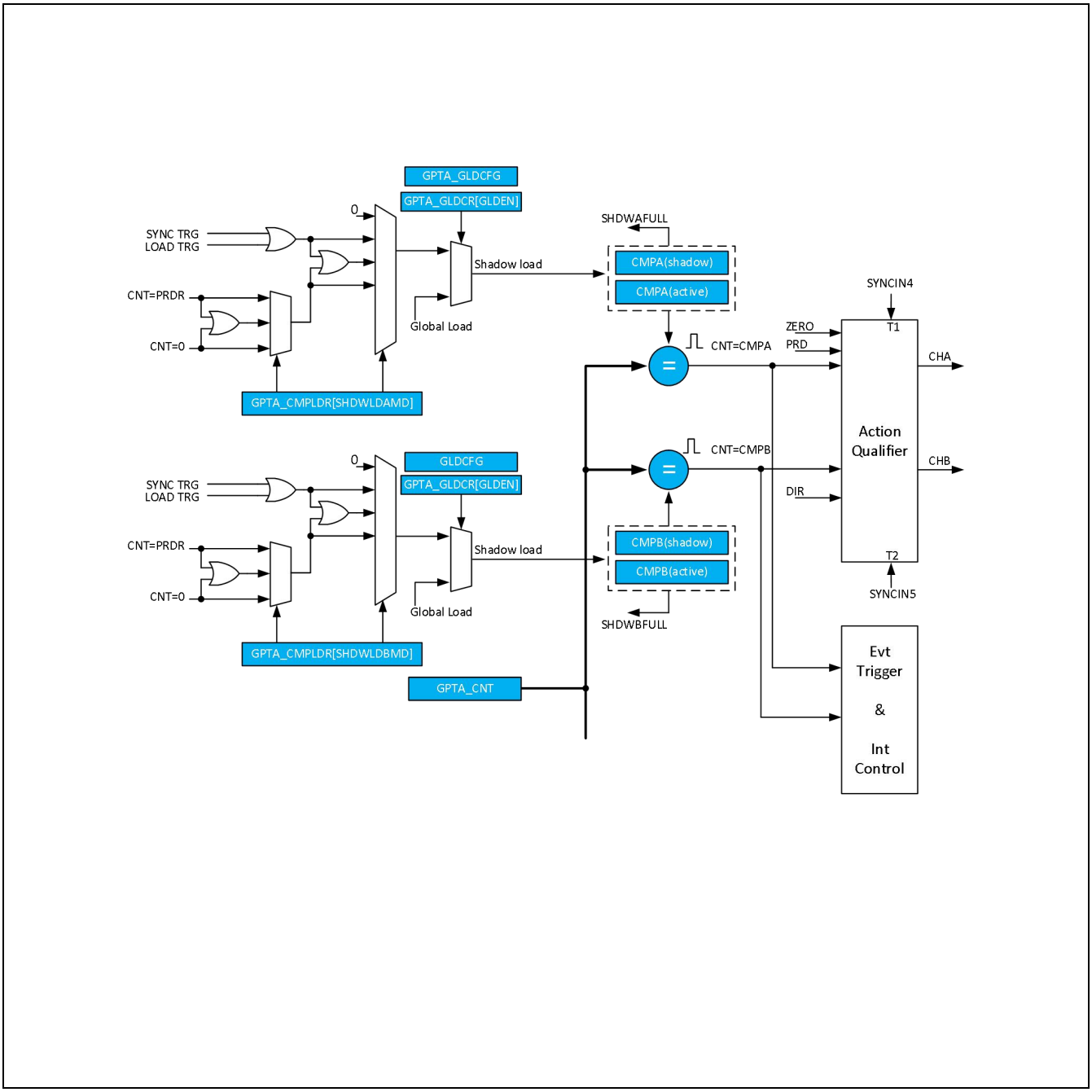


Figure 12- 11 计数器值比较控制

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个比较事件可以用于触发中断或者同步，也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

CMPA、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GPTA_CMPLDR[SHDWLDxMD]控制位进行设置。影子寄存器的使能可以通过GPTA_CMPLDR[LDCMPxMD]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GPTA_CMPLDR[SHDWLDxMD]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

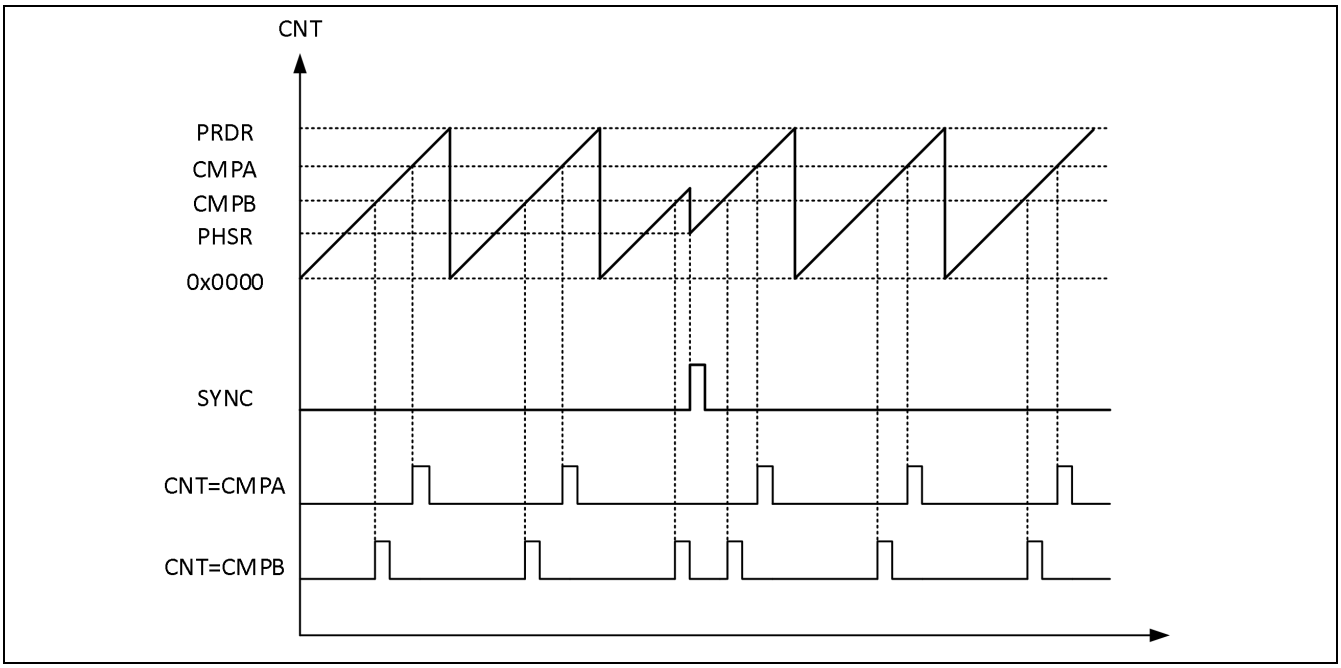


Figure 12-12 递增模式下比较事件产生时序

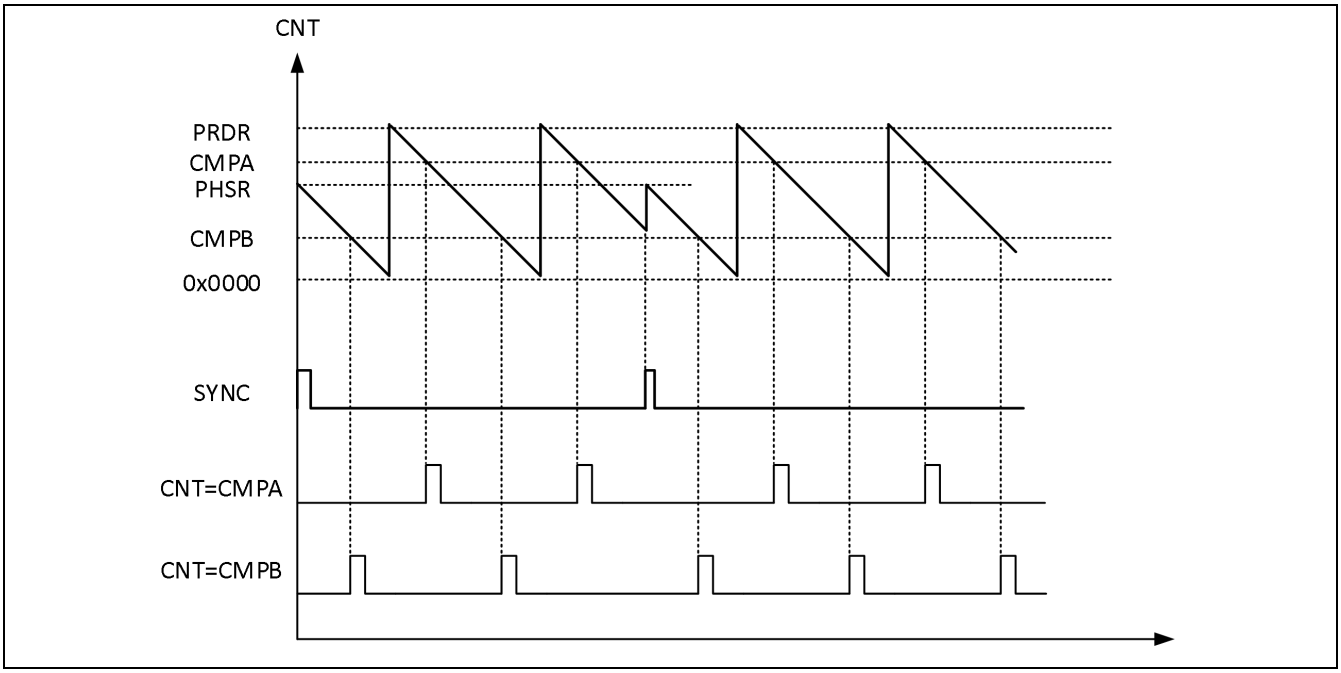


Figure 12-13 递减模式下比较事件产生时序

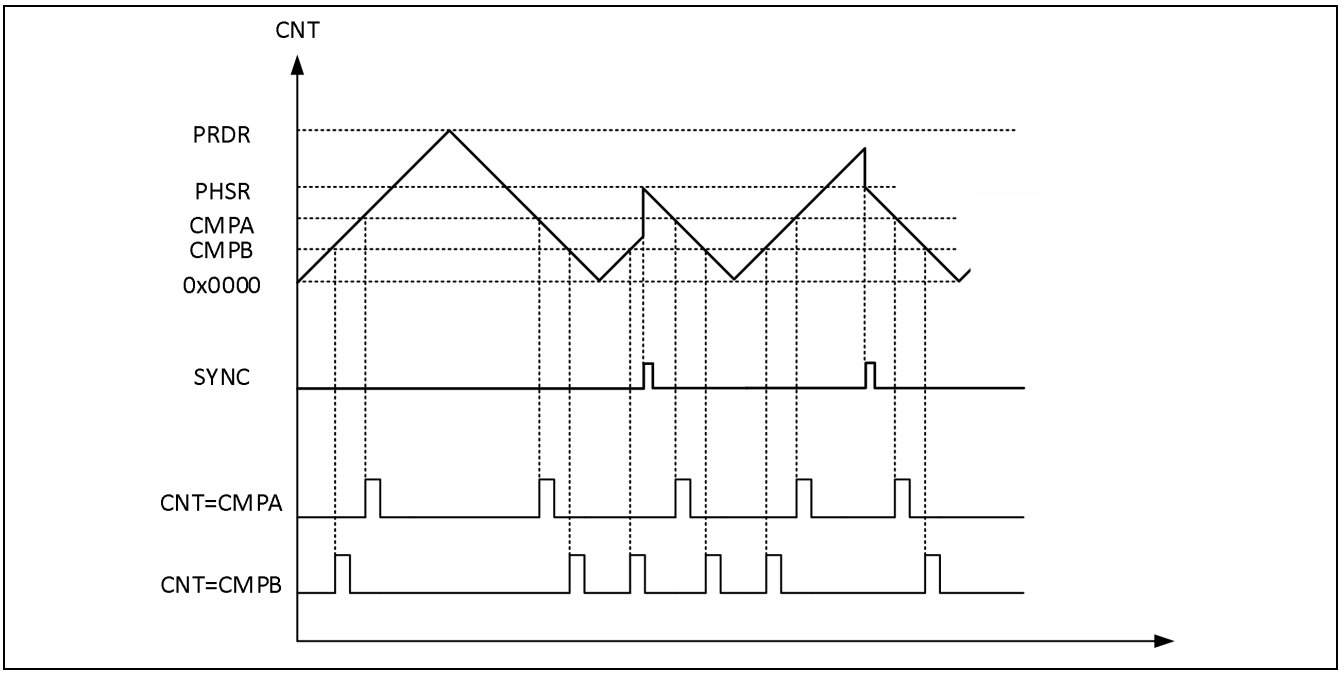


Figure 12-14 递增递减模式下比较事件产生时序

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，注意，这里的PWM1和PWM2是内部信号，不是外部引脚输出信号。PWM的波形产生基于不同事件的驱动，通过控制寄存器GPTA_AQCR1和GPTA_AQCR2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。GPTA_AQCR1对应控制PWM1通道上的波形输出，GPTA_AQCR2对应PWM2通道上的波形输出。GPTA_AQCR1和GPTA_AQCR2都具有影子寄存器功能，可以通过GPTA_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 （计数器值等于C1设置值，C1参考值由GPTA_AQCRx[C1SEL]设置）
- CNT = C2 （计数器值等于C2设置值，C2参考值由GPTA_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）

C1和C2是两个数字比较单元，可通过GPTA_AQCRx[C1SEL]和GPTA_AQCRx[C2SEL]选择不同的数据源。

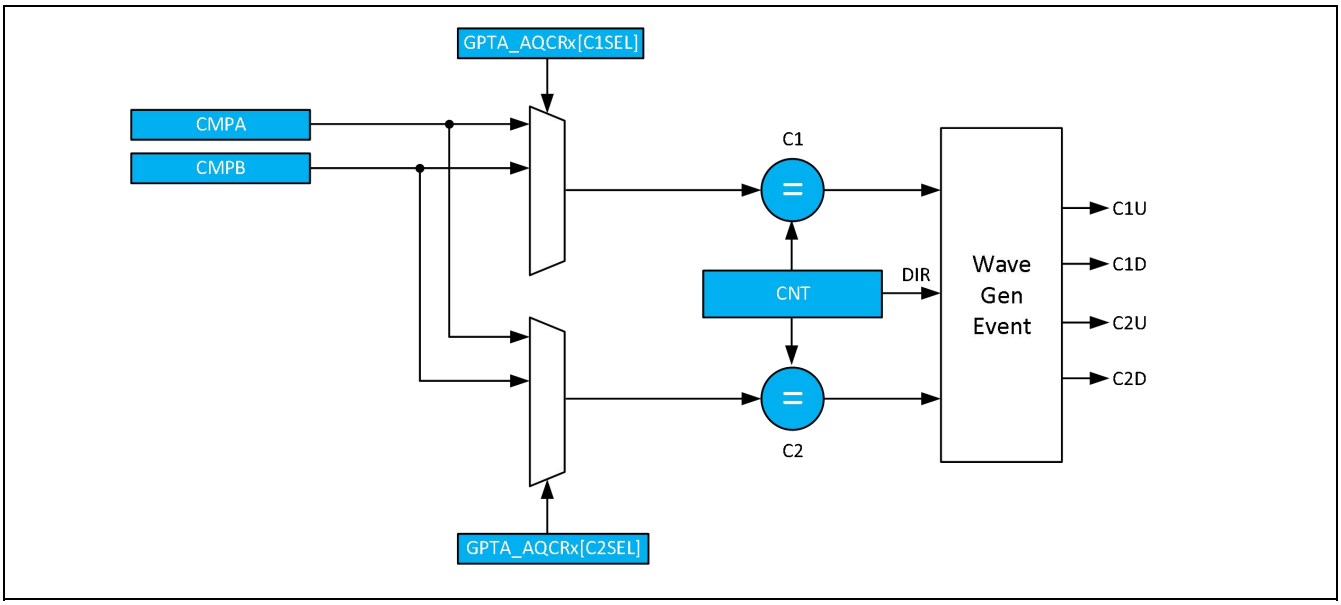


Figure 12-15 C1 and C2 selection control

T1和T2事件是来自于SYNCIN4/5的两个独立触发事件，可通过ETCB设置触发源，具体参考ETCB章节部分内容。T1和T2事件对计数器的计数值不会有任何影响，只是用于波形发生控制单元。

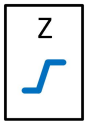
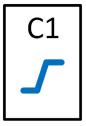
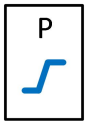
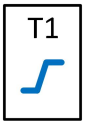
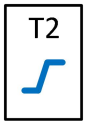
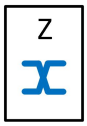
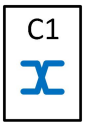
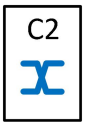
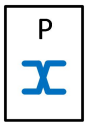
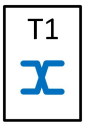
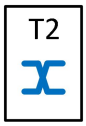
波形发生模块根据计数器的当前计数方向和发生的事件，决定PWM1和PWM2通道上的动作。所支持的的输出动作包括：

- 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 12-2 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	
SW	Z	C1	C2	P	T1	T2	没有动作
SW	Z	C1	C2	P	T1	T2	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 12-3 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 12-4 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 12-5 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。

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PWM1/2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。

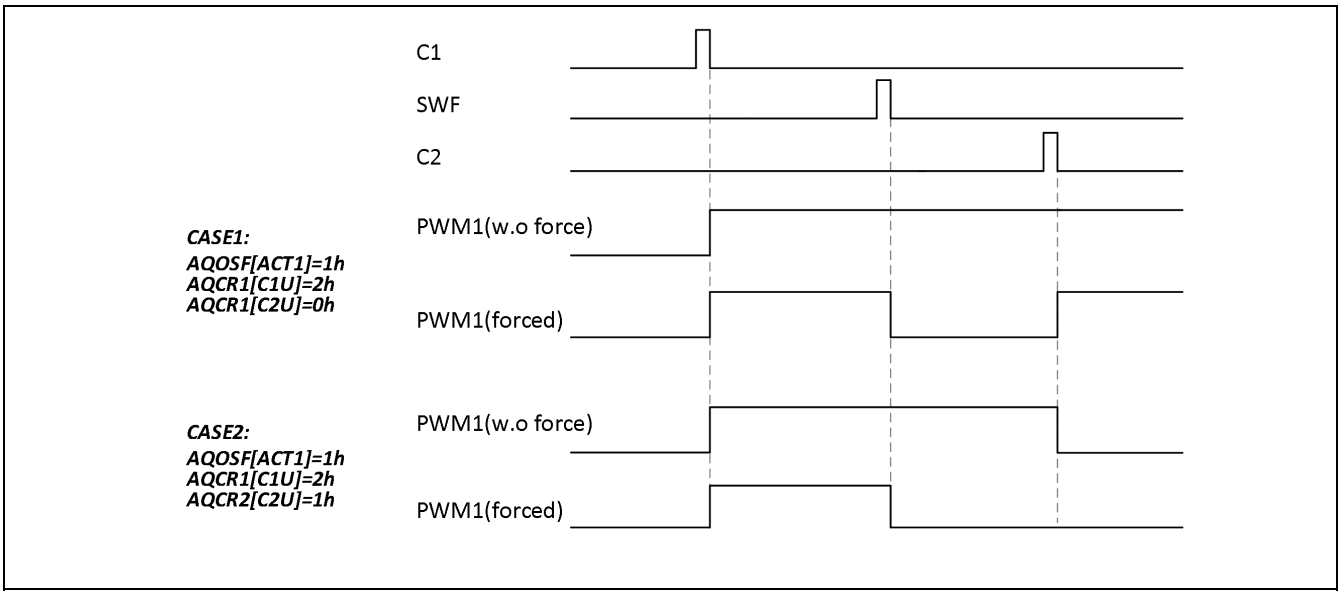


Figure 12-15 一次性软件强制输出

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。

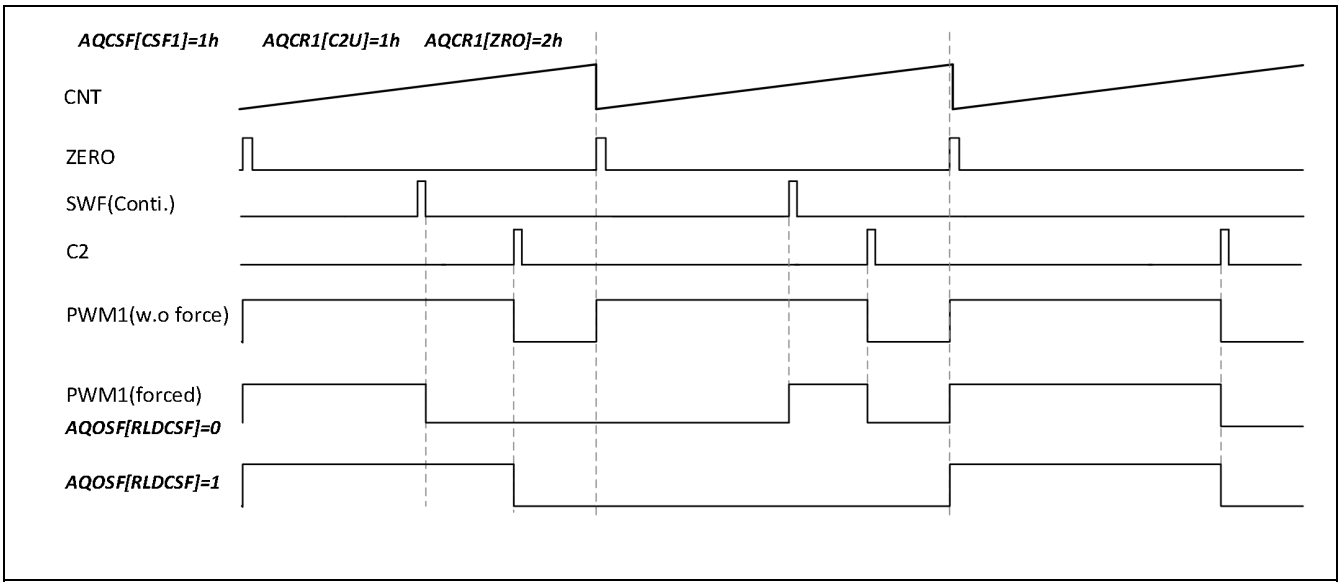


Figure 12-16 持续性软件强制输出

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。

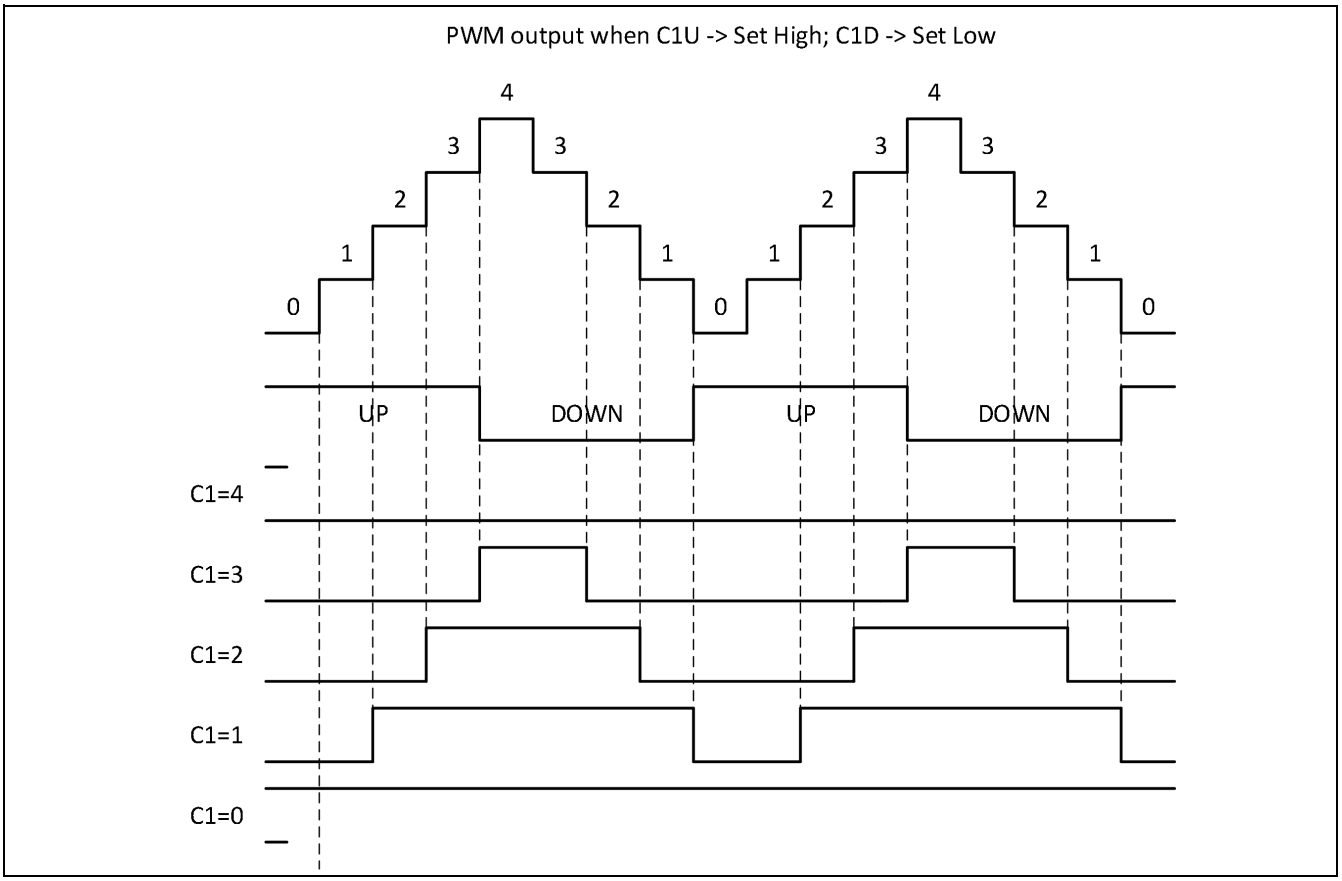


Figure 12-17 递增递减单比较值时，对称波形输出

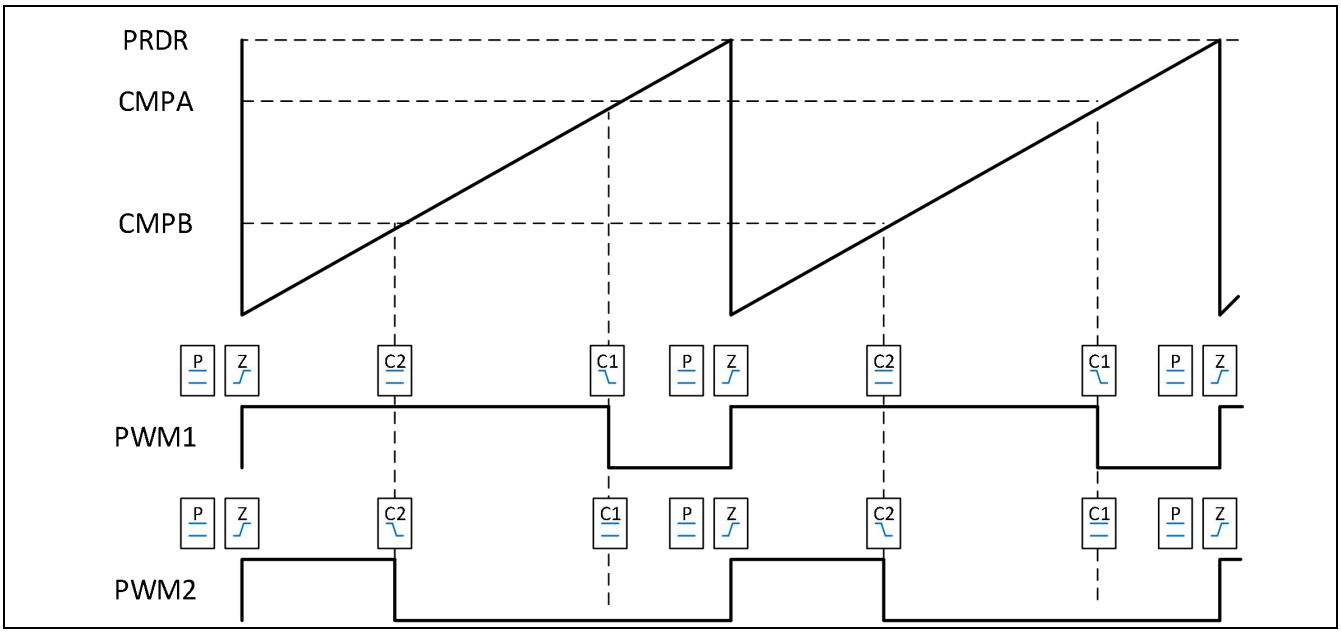


Figure 12-18 递增单沿，非对称波形输出

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。

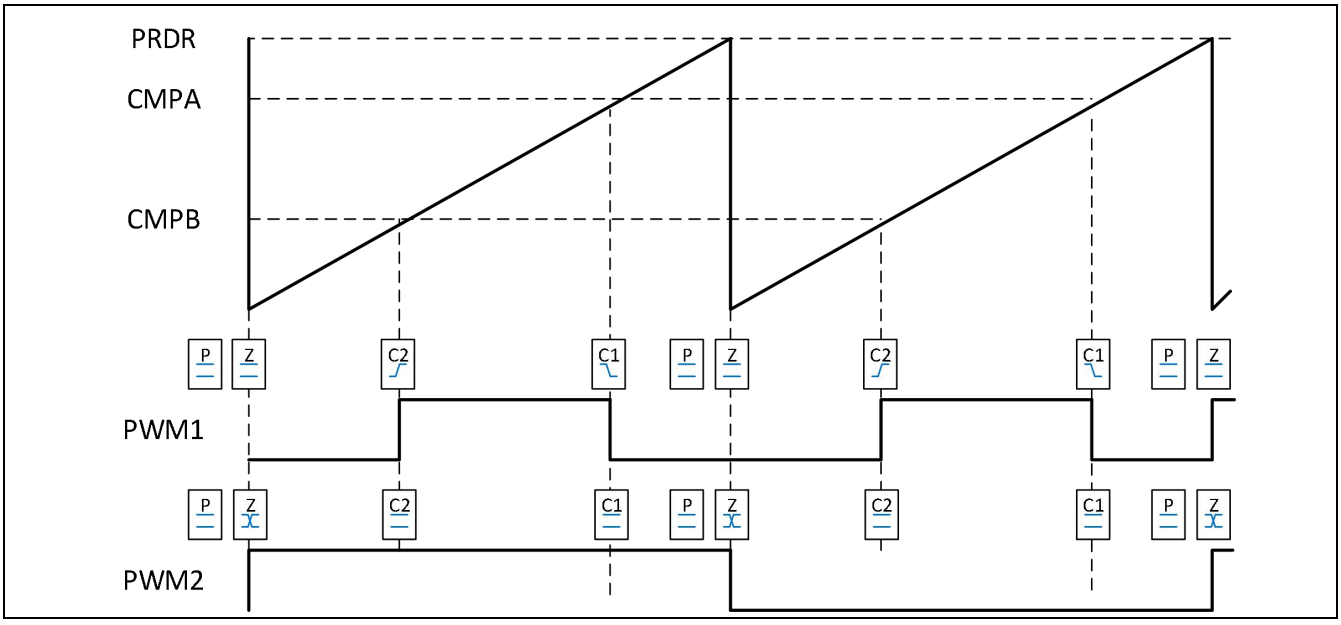


Figure 12-19 递增，脉冲定位非对称波形输出

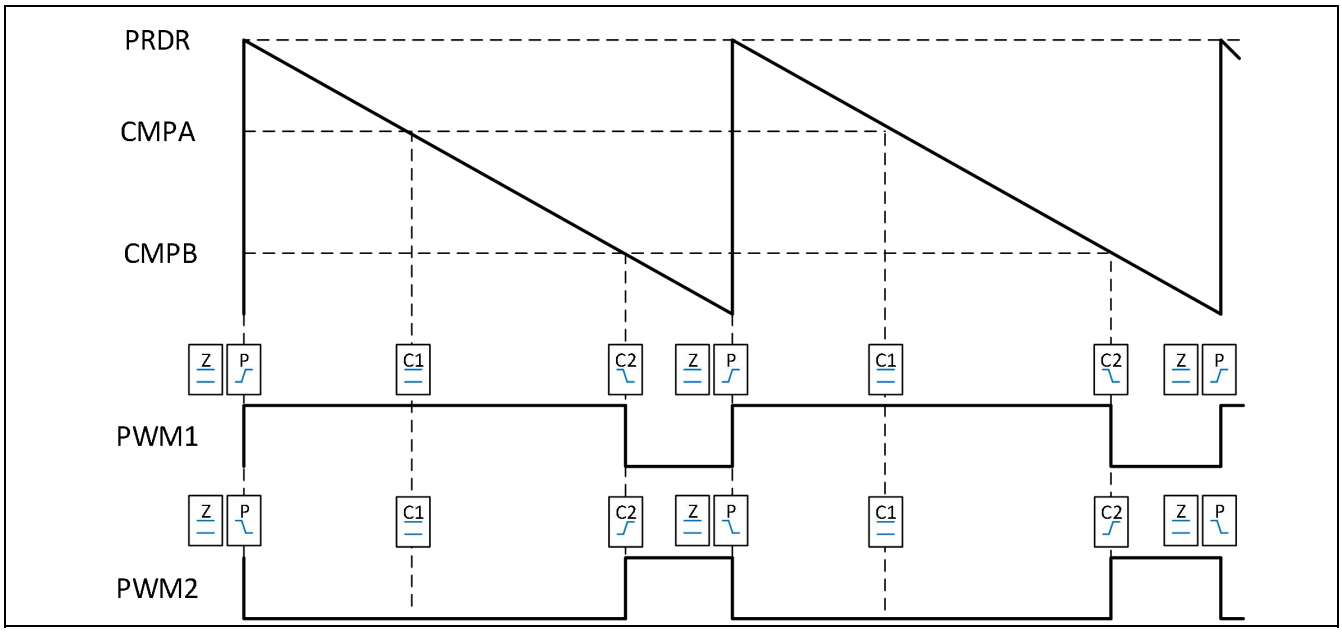


Figure 12-20 递减单沿，非对称波形输出

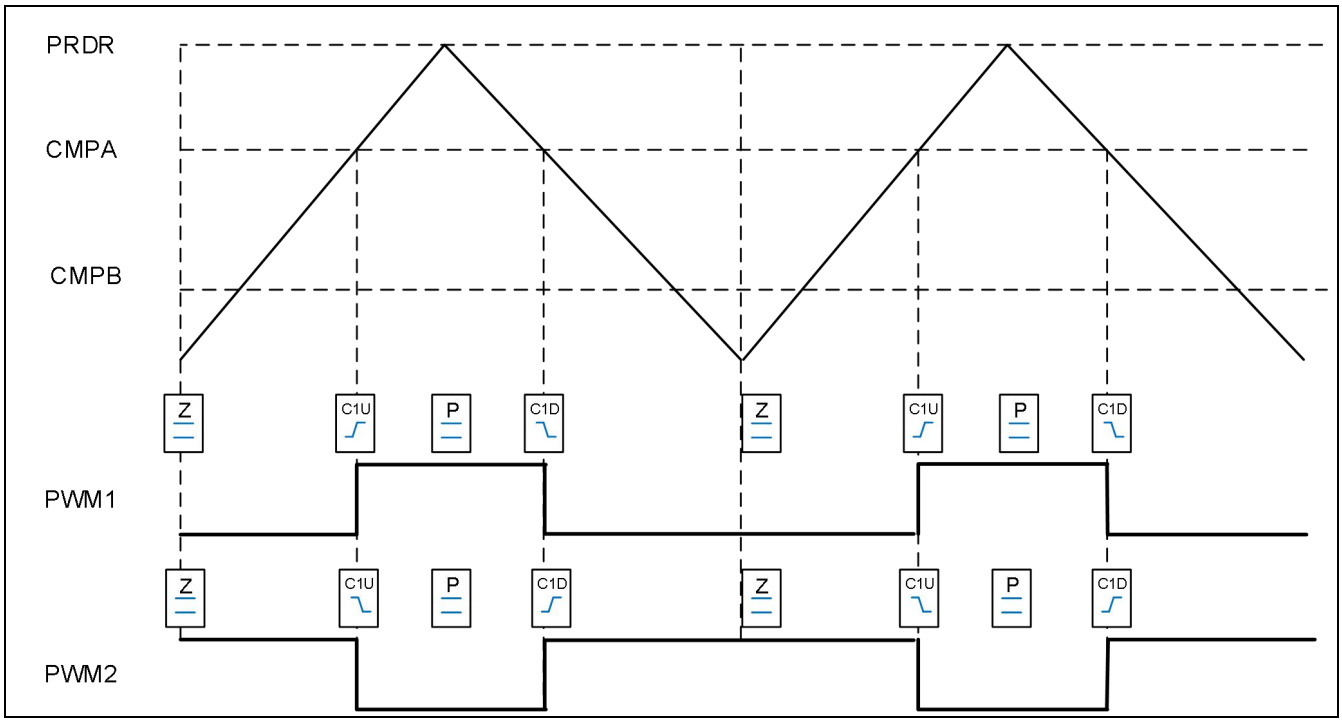


Figure 12-21 递增递减，双沿对称波形输出

12.3.5 捕获模式

12.3.5.1 概述

捕获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

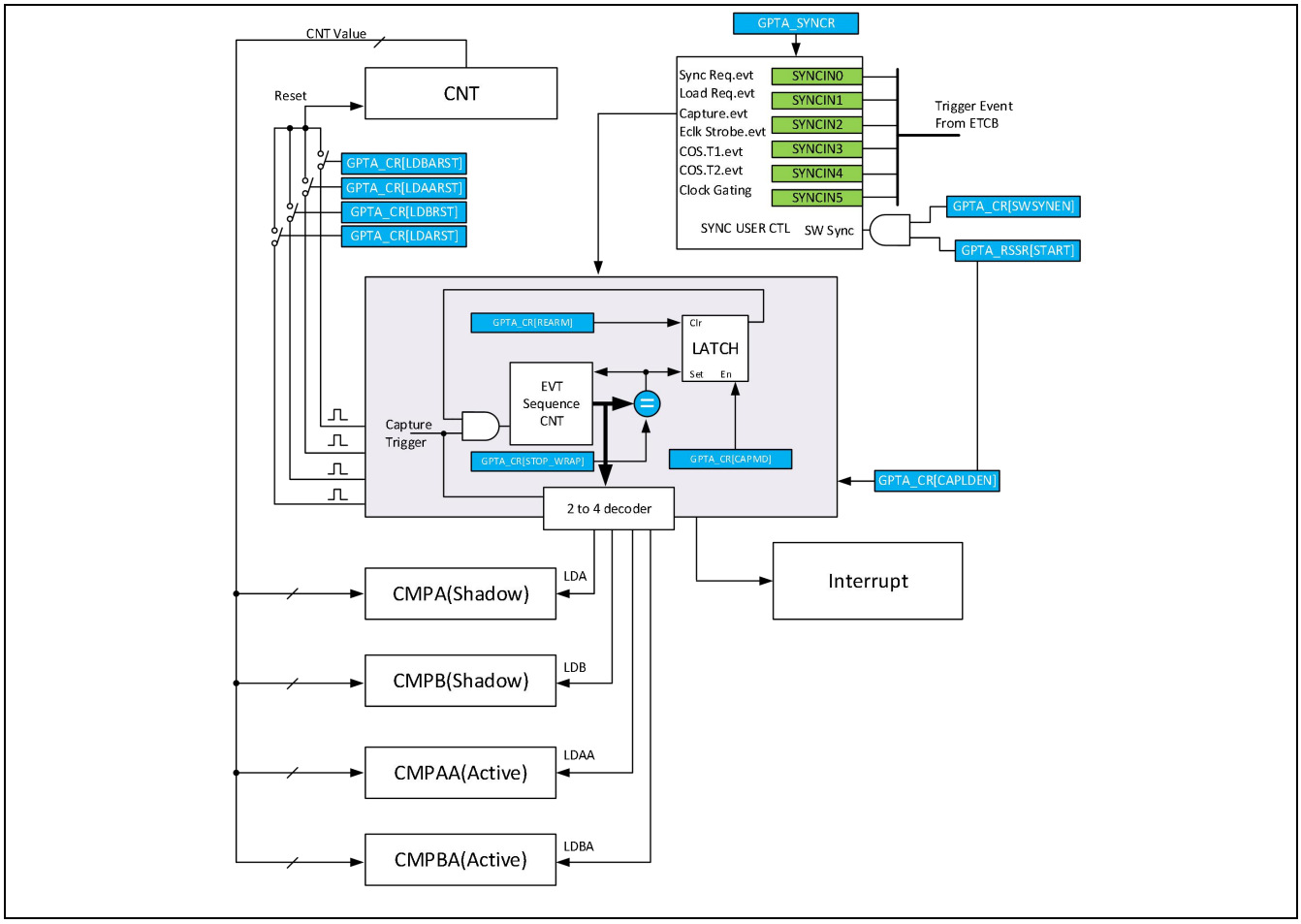


Figure 12- 22 捕获模式结构框图

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

- 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、CMPAA、CMPBA寄存器中。
- 捕获序列控制，可支持最大连续4个计数值捕获
- 捕获后计数器重置或继续计数

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 12-6 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 CMPB shadow, CAP_LD1 is triggered
2	CMPAA	CAP_LD2	Current counter value is loaded into CMPAA active, CAP_LD2 is triggered
3	CMPBA	CAP_LD3	Current counter value is loaded into CMPBA active, CAP_LD3 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（Continuous）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[REARM]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GPTA 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GPTA_CR[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，说明如何使用捕获模式。

- **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的第一个上升沿触发第一次load，计数值存入CMPA中，TIOA的第一个下降沿触发第二次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。（Figure12-23）

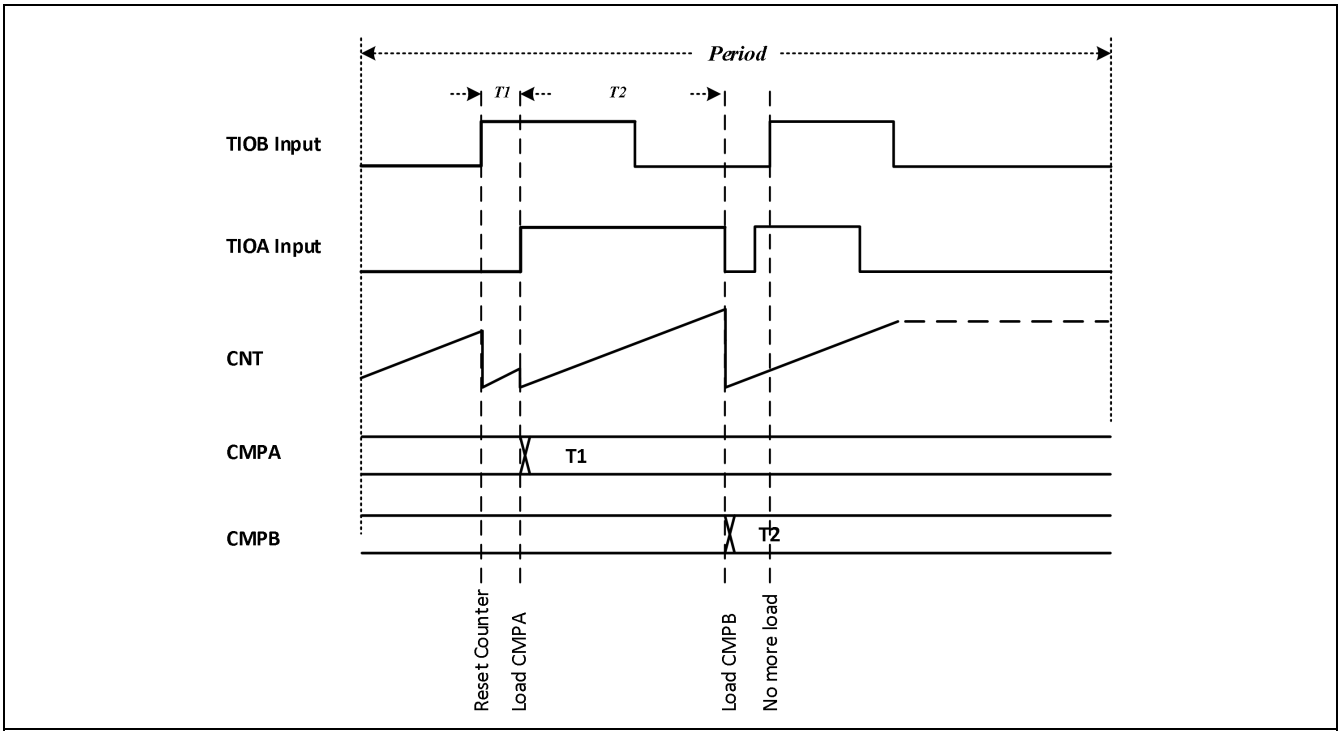


Figure 12-23 测量TIOA和TIOB相位差

- **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。（Figure12-24）

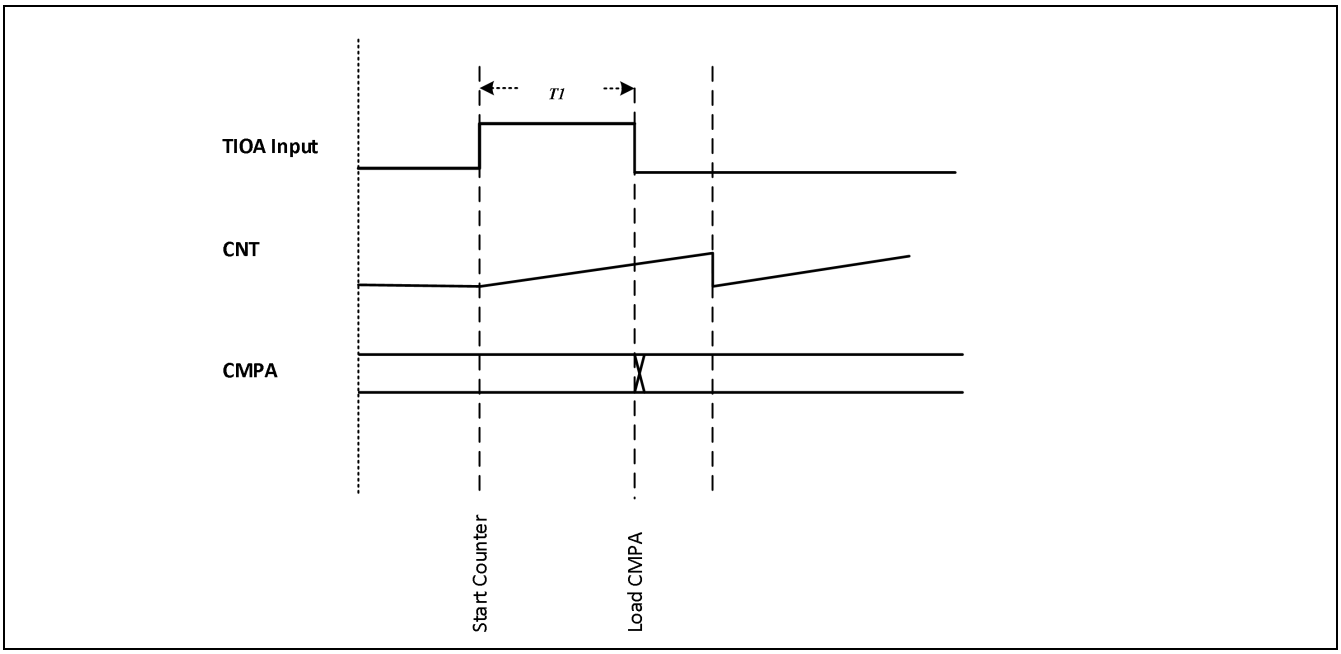


Figure 12-24 测量TIOA的脉冲宽度

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。

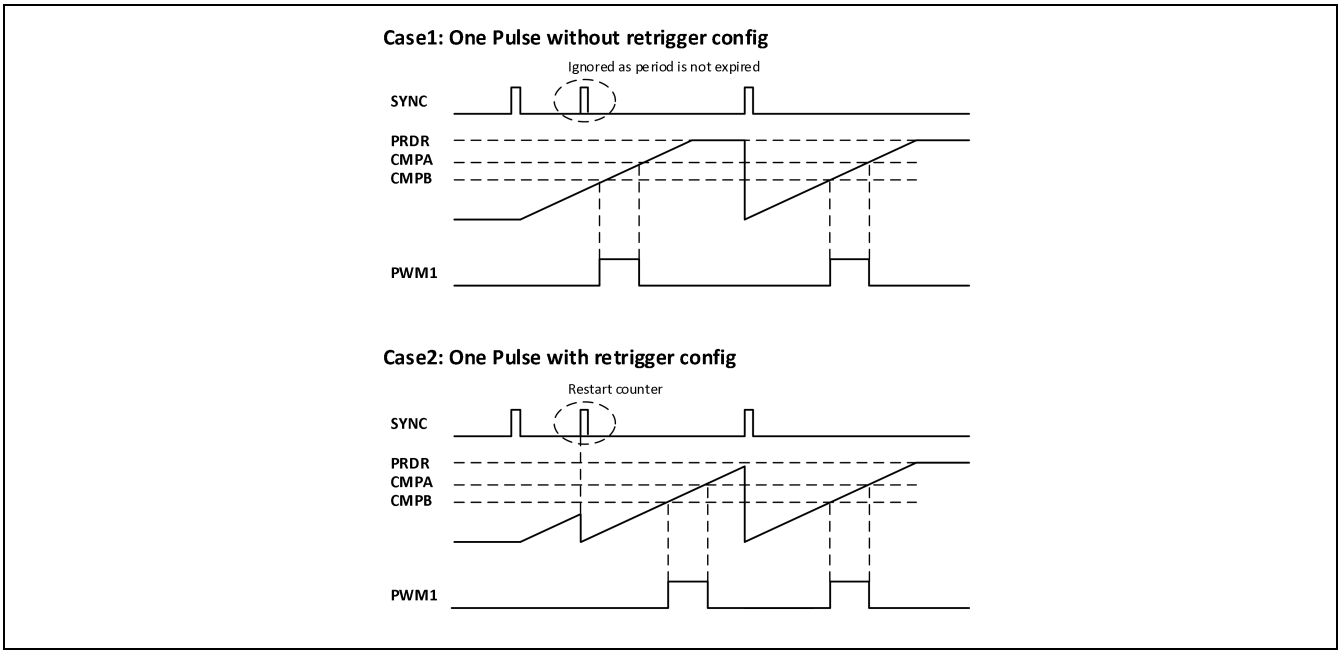


Figure 12-25 单次触发

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）
- 当前计数器值捕获
- 计数器值递增或递减一个计数值
- 触发改变PWM的输出状态

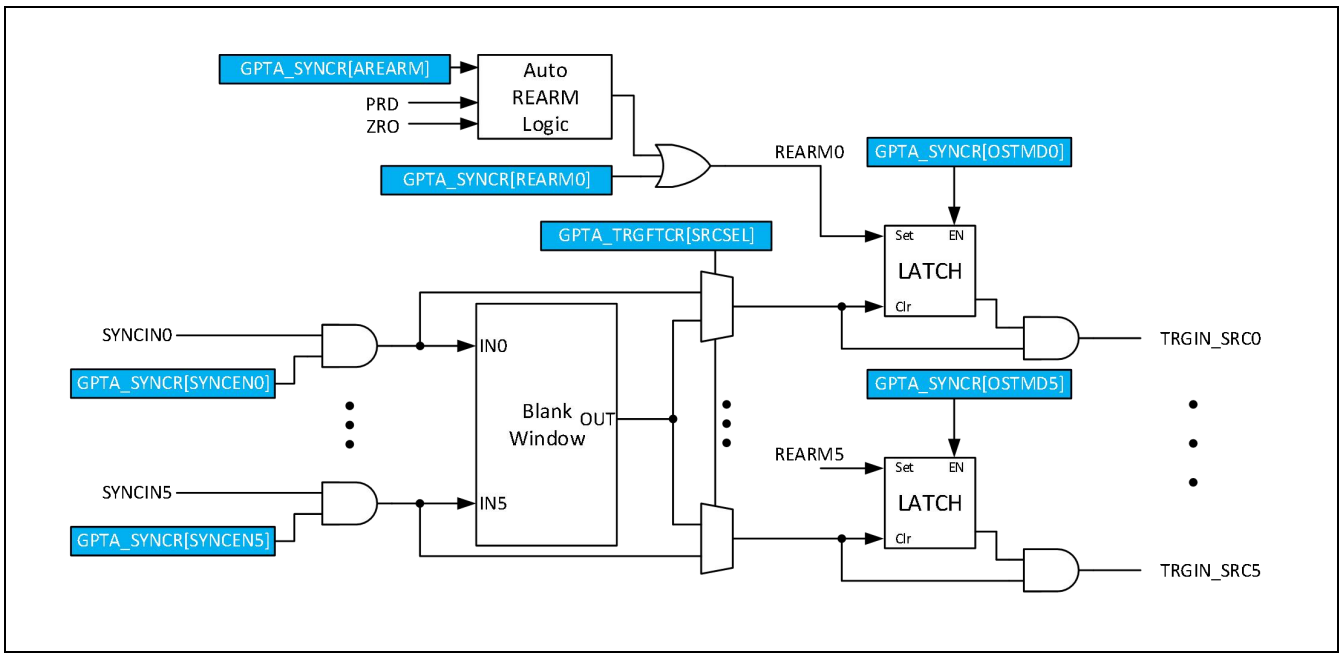


Figure 12-26 同步触发输入

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

计数器值捕获（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。

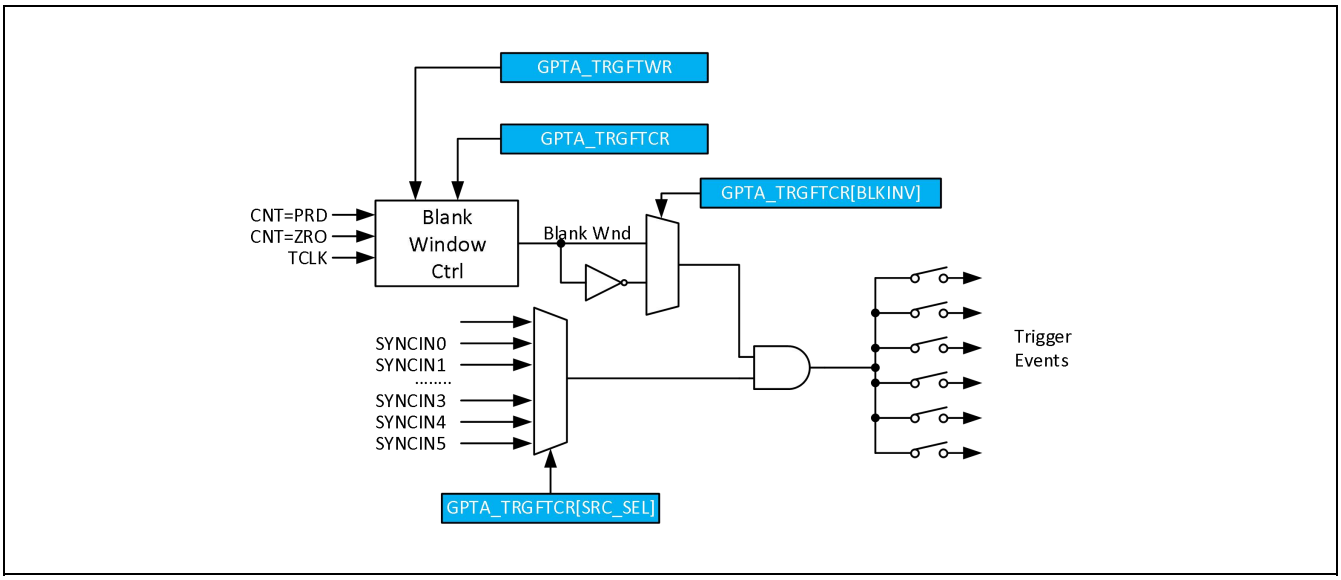


Figure 12-27 滤波器框图

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GPTA_TRGFTCR[ALIGNMD]）。窗口的延时和宽度可以通过GPTA_TRGFWR进行设置。

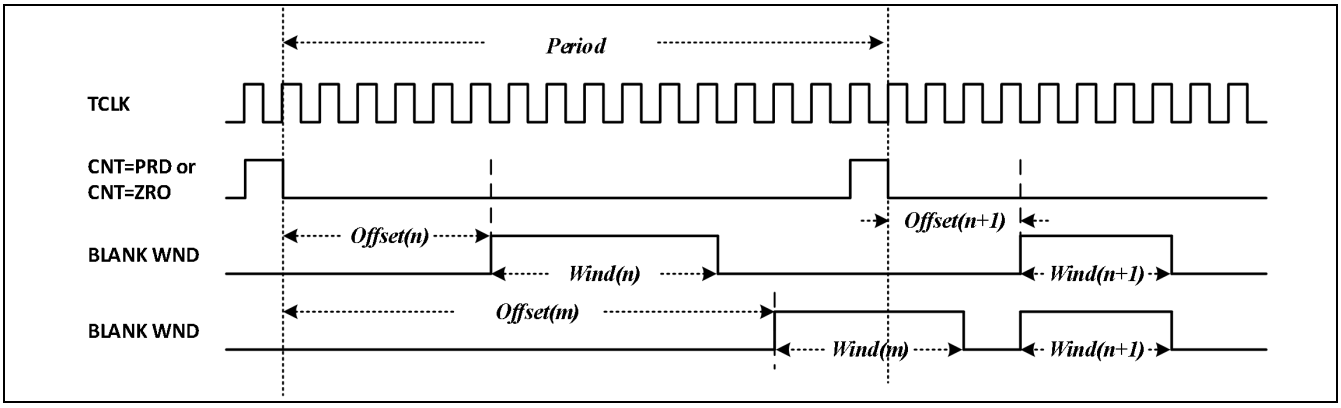


Figure 12-28 滤波器时序

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2路事件触发输出。每个事件输出对应一个GPTA中断信号，事件触发信号通过GPTA_EVTRG[TRGSEL]选择触发源，GPTA_EVTRG[TRGxOE]控制位用于使能触发信号输出到其他外设。

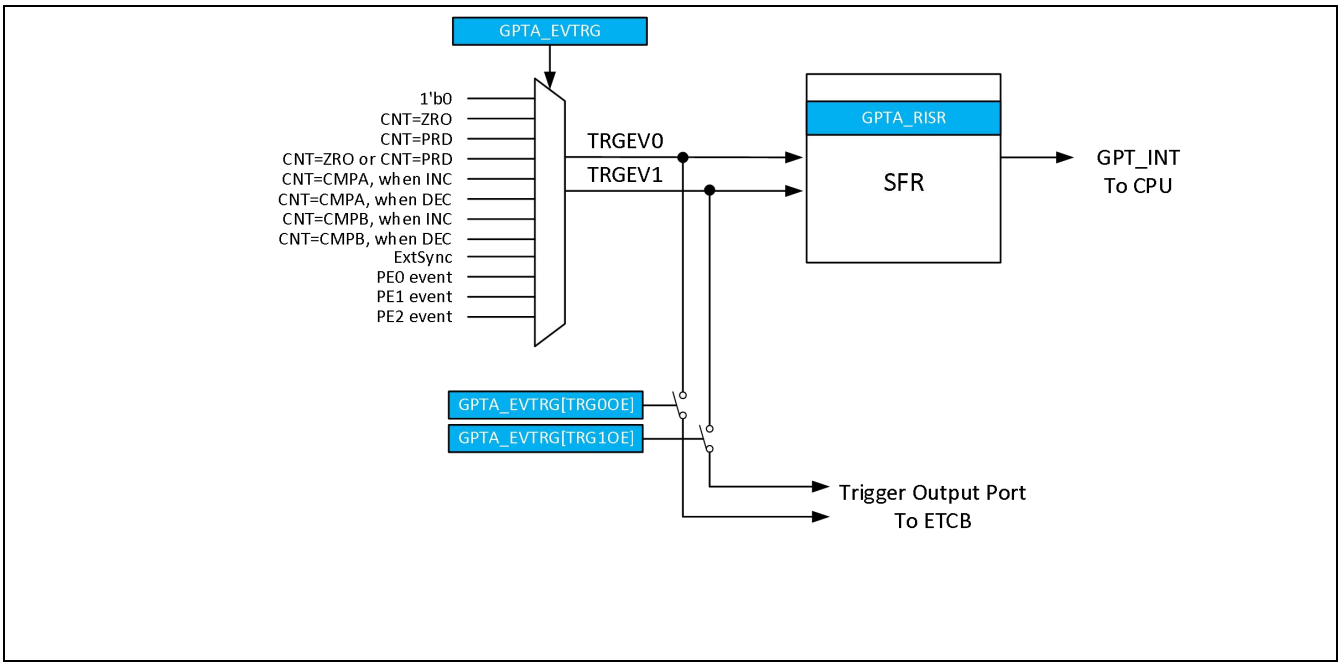


Figure 12-29 事件触发输出

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。

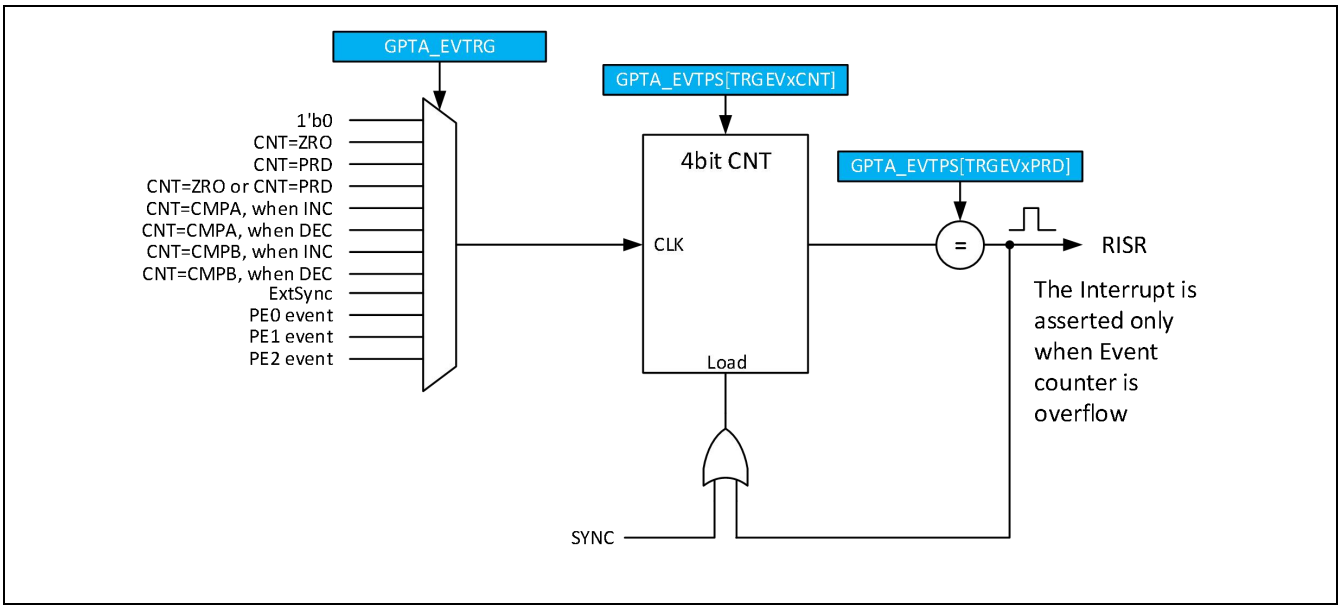


Figure 12-30 事件计数器

12.4 寄存器说明

12.4.1 寄存器表

Base Address of GPTA0: 0x40065000

Register	Offset	Description	Reset Value
GPTA_CEDR	0x00	ID和时钟控制寄存器	0x28980000
GPTA_RSSR	0x04	启停控制寄存器	0x00000000
GPTA_PSCR	0x08	时钟分频控制寄存器	0x00000000
GPTA_CR(WAVE=0)	0x0C	控制寄存器，捕获模式WAVE=0	0x00000000
GPTA_CR(WAVE=1)	0x0C	控制寄存器，波形输出模式WAVE=1	0x00000000
GPTA_SYNCR	0x10	同步控制寄存器	0x00000000
GPTA_GLDCR	0x14	全局载入控制寄存器	0x00000000
GPTA_GLDCFG	0x18	全局载入配置	0x00000000
GPTA_GLDCR2	0x1C	全局载入控制寄存器2	0x00000001
GPTA_PRDR	0x24	周期设置寄存器	0x00000000
GPTA_PHSR	0x28	相位设置寄存器	0x00000000
GPTA_CMPA	0x2C	比较值A寄存器	0x00000000
GPTA_CMPB	0x30	比较值B寄存器	0x00000000
GPTA_CMPLDR	0x3C	比较值载入控制寄存器	0x00000090
GPTA_CNT	0x40	时基计数器寄存器	0x00000000
GPTA_AQLDR	0x44	波形输出载入控制寄存器	0x00000024
GPTA_AQCR1	0x48	PWM1波形输出控制寄存器	0x00000000
GPTA_AQCR2	0x4C	PWM2波形输出控制寄存器	0x00000000
GPTA_AQOSF	0x5C	一次性软件强制输出控制	0x00010000
GPTA_AQCSF	0x60	连续软件强制输出控制	0x00000000
GPTA_TRGFTCR	0xB8	数字比较器滤波窗控制寄存器	0x00000000
GPTA_TRGFTWR	0xBC	数字比较器滤波窗时序寄存器	0x00000000
GPTA_EVTRG	0xC0	事件触发选择寄存器	0x00000000
GPTA_EVPS	0xC4	事件触发计数寄存器	0x00000000
GPTA_EVCNTINIT	0xC8	事件触发计数器初始化值寄存器	0x00000000
GPTA_EVSWF	0xCC	事件计数器载入控制寄存器	0x00000000
GPTA_RISR	0xD0	原始中断状态寄存器	0x00000000
GPTA_MISR	0xD4	中断状态寄存器	0x00000000
GPTA_IMCR	0xD8	中断使能控制寄存器	0x00000000

GPTA_ICR	0xDC	中断清除寄存器	0x00000000
GPTA_REGPROT	0xE8	寄存器写保护控制器	0x00000000
GPTA_ CMPAA	0x82C	比较值A active寄存器	0x00000000
GPTA_ CMPBA	0x830	比较值B active寄存器	0x00000000

12.4.2 GPTA_CEDR(ID和时钟控制寄存器)

Address = Base Address+ 0x00, Reset Value = 0x28980000

31	30	29	28	27	26	25	24	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
0	0	1	0	1	0	0	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
IDCODE	[31:16]	R	当前GPTA模块的版本信息。
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)
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
CSS	[3]	RW	计数器时钟源选择位。 0h：PCLK 1h：由SYNCIN3控制 其他：保留
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
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。

12.4.3 GPTA_RSSR (启停控制寄存器)

Address = Base Address+ 0x04, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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。
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
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）。
NOTE: 该寄存器受REGPROT保护，需要先解锁，才能写入。			

12.4.4 GPTA_PSCR(时钟分频控制寄存器)

Address = Base Address+ 0x08, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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。TCLK的频率：FTCLK = FPCLK / (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(WAVE=0)(控制寄存器，捕获模式 WAVE=0)

Address = Base Address+ 0x0C, Reset Value = 0x00000000

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

Name	Bit	Type	Description
LDBARST	[26]	RW	CMPBA(Shadow)捕获载入后，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
LDAARST	[25]	RW	CMPAA(Shadow)捕获载入后，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	CMPB(Shadow)捕获载入后，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
LDARST	[23]	RW	CMPA(Shadow)捕获载入后，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
STOP_WRAP	[22:21]	RW	Capture模式下，捕获事件计数器周期设置值。
CAPMD	[20]	RW	捕获模式设置。 0h：连续捕获模式 1h：一次性捕获模式
REARM	[19]	RW	重置CAPTURE控制。 0h：无效 1h：重置捕获 重置时，捕获事件计数器被清零，自动打开CAPLDEN
WAVE	[18]	RW	GPTA工作模式选择。 0h：捕获模式 1h：波形发生模式
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
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GPTA_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
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
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
BURST	[9]	RW	群脉冲模式。 0h：禁止群脉冲模式 1h：使能群脉冲模式
CAPLDEN	[8]	RW	CMPA和CMPB在捕获事件触发时，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，并不影响捕获事件SYNCIN2的触发。
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周期结束(PEND)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]
IDLEST	[3]	RW	

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。
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
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(WAVE=1)(控制寄存器，波形输出模式 WAVE=1)

Address = Base Address+ 0x0C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	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
R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
WAVE	[18]	RW	GPTA工作模式选择。 0h：捕获模式 1h：波形发生模式
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
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
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
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

BURST	[9]	RW	群脉冲模式。 0h：禁止群脉冲模式 1h：使能群脉冲模式
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
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
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周期结束(PEND)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]
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
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。
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
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+ 0x10, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
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
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
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
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
REARM5	[21]	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
REARM4	[20]	RW	
REARM3	[19]	RW	
REARM2	[18]	RW	
REARM1	[17]	RW	
REARM0	[16]	RW	
OSTMD5	[13]	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。
OSTMD4	[12]	RW	
OSTMD3	[11]	RW	
OSTMD2	[10]	RW	
OSTMD1	[9]	RW	
OSTMD0	[8]	RW	
SYNCEN5	[5]	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）
SYNCEN4	[4]	RW	
SYNCEN3	[3]	RW	
SYNCEN2	[2]	RW	
SYNCEN1	[1]	RW	

SYNCEN0	[0]	RW	
NOTE: 该寄存器受REGPROT保护，需要先解锁，才能写入。			

12.4.8 GPTA_GLDCR(全局载入控制寄存器)

Address = Base Address+ 0x14, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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。
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
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。
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]
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
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+ 0x18, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	RSVD		AQCR2	AQCR1	RSVD					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
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
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
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
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
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
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

12.4.10 GPTA_GLDCR2 (全局载入控制寄存器2)

Address = Base Address+ 0x1C, Reset Value = 0x00000001

31	30	29	28	27	26	25	24	23	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
GFRCLD	[1]	W	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
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
NOTE: 该寄存器受REGPROT保护，需要先解锁，才能写入。			

12.4.11 GPTA_PRDR (周期设置寄存器)

Address = Base Address+ 0x24, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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GPTA_CR[PRDLD]可以选择Shadow到Active载入的触发条件。

12.4.12 GPTA_PHSR(相位设置寄存器)

Address = Base Address+ 0x28, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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
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
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GPTA_CR[PHSEN] = 0时，同步事件不会触发PHSR载入到CNT中，当GPTA_CR[PHSEN] = 1时，同步事件发生会触发PHSR载入到CNT中。
NOTE: PHSR寄存器只有在外部Sync触发时(SYNCIN0)才有效			

12.4.13 GPTA_CMPA(比较值A寄存器)

Address = Base Address+ 0x2C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
OVWRT	[31]	R	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。
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GPTA_CMPLDR[SHDWCMPA]进行设置。在Shadow模式下，可以通过GPTA_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。

12.4.14 GPTA_CMPB(比较值B寄存器)

Address = Base Address+ 0x30, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
OVWRT	[31]	R	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。
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GPTA_CMPLDR[SHDWCMPB]进行设置。在Shadow模式下，可以通过GPTA_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。

12.4.15 GPTA_CMPLDR(比较值载入控制寄存器)

Address = Base Address+ 0x3C, Reset Value = 0x00000090

31	30	29	28	27	26	25	24	23	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										SHDWLDBMD		SHDWLDAMD				RSVD		LDCMPBMD	LDCMPAMD
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
SHDWBFULL	[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
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
SHDWLDBMD	[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
SHDWLDAMD	[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。

LDCMPBMD	[1]	RW	CMPB的Shadow功能使能控制。 0h：Shadow模式 1h：Immediate模式 [1]
LDCMPAMD	[0]	RW	CMPA的Shadow功能使能控制。 0h：Shadow模式 1h：Immediate模式 [1]
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。			

12.4.16 GPTA_CNT(时基计数器寄存器)

Address = Base Address+ 0x40, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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。

12.4.17 GPTA_AQLDR(波形输出载入控制寄存器)

Address = Base Address+ 0x44, Reset Value = 0x00000024

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																								SHDWLD2MD			SHDWLD1MD			LDAQ2MD	LDAQ1MD
0	0	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
SHDWLD2MD	[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。
SHDWLD1MD	[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。
LDAQ2MD	[1]	RW	AQCR2寄存器的Shadow功能使能控制。 0h：Shadow模式 1h：Immediate模式
LDAQ1MD	[0]	RW	AQCR1寄存器的Shadow功能使能控制。 0h：Shadow模式 1h：Immediate模式

12.4.18 GPTA_AQCR1(PWM1波形输出控制寄存器)

Address = Base Address+ 0x48, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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。
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。
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）
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）
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）
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）
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）
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）
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）
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）
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）
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）

12.4.19 GPTA_AQCR2(PWM2波形输出控制寄存器)

Address = Base Address+ 0x4C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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		CBD		CBU		CAD		CAU		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
C2SEL	[23:22]	RW	CB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。
C1SEL	[21:20]	RW	CA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。
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）
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）
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）
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）
CBD	[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）
CBU	[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）
CAD	[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）
CAU	[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）
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）
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）

12.4.20 GPTA_AQOSF(一次性软件强制输出控制)

Address = Base Address+ 0x5C, Reset Value = 0x00010000

31	30	29	28	27	26	25	24	23	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								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
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
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）
OSTSF2	[4]	W	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PWM2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）
OSTSF1	[0]	W	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。

12.4.21 GPTA_AQCSF(连续软件强制输出控制)

Address = Base Address+ 0x60, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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GPTA_AQOSF[RLDCSF]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
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GPTA_AQOSF[RLDCSF]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

12.4.22 GPTA_TRGFTCR (数字比较器滤波窗控制寄存器)

Address = Base Address+ 0xB8, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																								CROSSMD	ALIGNMD		BLKINV	RSVD	SRC_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
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
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
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
BLKINV	[4]	RW	窗口使能反转控制。 0h：窗口不反转 1h：窗口反转
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

12.4.23 GPTA_TRGFTWR (数字比较器滤波窗时序寄存器)

Address = Base Address+ 0xBC, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
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。
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。

12.4.24 GPTA_EVTRG(事件触发选择寄存器)

Address = Base Address+ 0xC0, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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
CNT1INITFRC	[25]	R	TRGEV1CNT软件触发更新 0h：无效 1h：EVCNT1INIT内容更新到EVCNT1中
CNT0INITFRC	[24]	R	TRGEV0CNT软件触发更新 0h：无效 1h：EVCNT0INIT内容更新到EVCNT0中
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
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
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。
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。
TRGSEL1	[7:4]	RW	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
TRGSEL0	[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

12.4.25 GPTA_EVPS(事件触发计数寄存器)

Address = Base Address+ 0xC4, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								TRGEV1CNT				TRGEV0CNT				RSVD								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	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
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。
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。
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
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

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

Address = Base Address+ 0xC8, Reset Value = 0x00000000

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

Name	Bit	Type	Description
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。
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。

12.4.27 GPTA_EVSWF(事件计数器载入控制寄存器)

Address = Base Address+ 0xCC, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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
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

12.4.28 GPTA_RISR(原始中断状态寄存器)

Address = Base Address+ 0xD0, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	CAP_LD3	CAP_LD2	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
PEND	[16]	R	周期结束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
CBU	[10]	R	递增阶段CNT = CMPB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
CAU	[8]	R	递增阶段CNT = CMPA中断请求原始标志状态
CAP_LD3	[7]	R	Capture Load to CMPBA中断请求原始标志状态
CAP_LD2	[6]	R	Capture Load to CMPAA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_LD0	[4]	R	Capture Load to CMPA中断请求原始标志状态
TRGEV1	[1]	R	TRGEV1中断请求原始标志状态
TRGEV0	[0]	R	TRGEV0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

12.4.29 GPTA_MISR(中断状态寄存器)

Address = Base Address+ 0xD4, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	CAP_LD3	CAP_LD2	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
PEND	[16]	R	周期结束中断请求标志状态
CBD	[11]	R	递减阶段CNT = CMPB中断请求标志状态
CBU	[10]	R	递增阶段CNT = CMPB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
CAU	[8]	R	递增阶段CNT = CMPA中断请求标志状态
CAP_LD3	[7]	R	Capture Load to CMPBA中断请求标志状态
CAP_LD2	[6]	R	Capture Load to CMPAA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_LD0	[4]	R	Capture Load to CMPA中断请求标志状态
TRGEV1	[1]	R	TRGEV1中断请求标志状态
TRGEV0	[0]	R	TRGEV0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			

12.4.30 GPTA_IMCR(中断使能控制寄存器)

Address = Base Address+ 0xD8, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	CAP_LD3	CAP_LD2	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	R	R	RW	RW	R	R	RW	RW

Name	Bit	Type	Description
PEND	[16]	RW	周期结束中断请求使能控制位
CBD	[11]	RW	递减阶段CNT = CMPB中断请求使能控制位
CBU	[10]	RW	递增阶段CNT = CMPB中断请求使能控制位
CAD	[9]	RW	递减阶段CNT = CMPA中断请求使能控制位
CAU	[8]	RW	递增阶段CNT = CMPA中断请求使能控制位
CAP_LD3	[7]	RW	Capture Load to CMPBA中断请求使能控制位
CAP_LD2	[6]	RW	Capture Load to CMPAA中断请求使能控制位
CAP_LD1	[5]	RW	Capture Load to CMPB中断请求使能控制位
CAP_LD0	[4]	RW	Capture Load to CMPA中断请求使能控制位
TRGEV1	[1]	RW	TRGEV1中断使能控制位
TRGEV0	[0]	RW	TRGEV0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			

12.4.31 GPTA_ICR(中断清除寄存器)

Address = Base Address+ 0xDC, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	CAP_LD3	CAP_LD2	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	R	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
PEND	[16]	W	周期结束中断清除控制位
CBD	[11]	W	递减阶段CNT = CMPB中断清除控制位
CBU	[10]	W	递增阶段CNT = CMPB中断清除控制位
CAD	[9]	W	递减阶段CNT = CMPA中断清除控制位
CAU	[8]	W	递增阶段CNT = CMPA中断清除控制位
CAP_LD3	[7]	W	Capture Load to CMPBA中断清除控制位
CAP_LD2	[6]	W	Capture Load to CMPAA中断清除控制位
CAP_LD1	[5]	W	Capture Load to CMPB中断清除控制位
CAP_LD0	[4]	W	Capture Load to CMPA中断清除控制位
TRGEV1	[1]	W	TRGEV1中断清除控制位
TRGEV0	[0]	W	TRGEV0中断清除控制位
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’			

12.4.32 GPTA_REGPROT (寄存器写保护控制器)

Address = Base Address+ 0xE8, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
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
WRKEY	[31:16]	R	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
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

12.4.33 GPTA_CMPAA (比较值A active寄存器)

Address = Base Address+ 0x82C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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															CMPAA															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
OVWRT	[31]	R	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。
CMPAA	[15:0]	R	比较值A寄存器。 当工作于Capture模式下，此寄存器对应CAPLD2事件触发的捕获值。

12.4.34 GPTA_CMPBA (比较值B active寄存器)

Address = Base Address+ 0x830, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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															CMPBA															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
OVWRT	[31]	R	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。
CMPBA	[15:0]	R	比较值B active寄存器。 当工作于Capture模式下，此寄存器对应CAPLD3事件触发的捕获值。

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。

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。

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 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
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

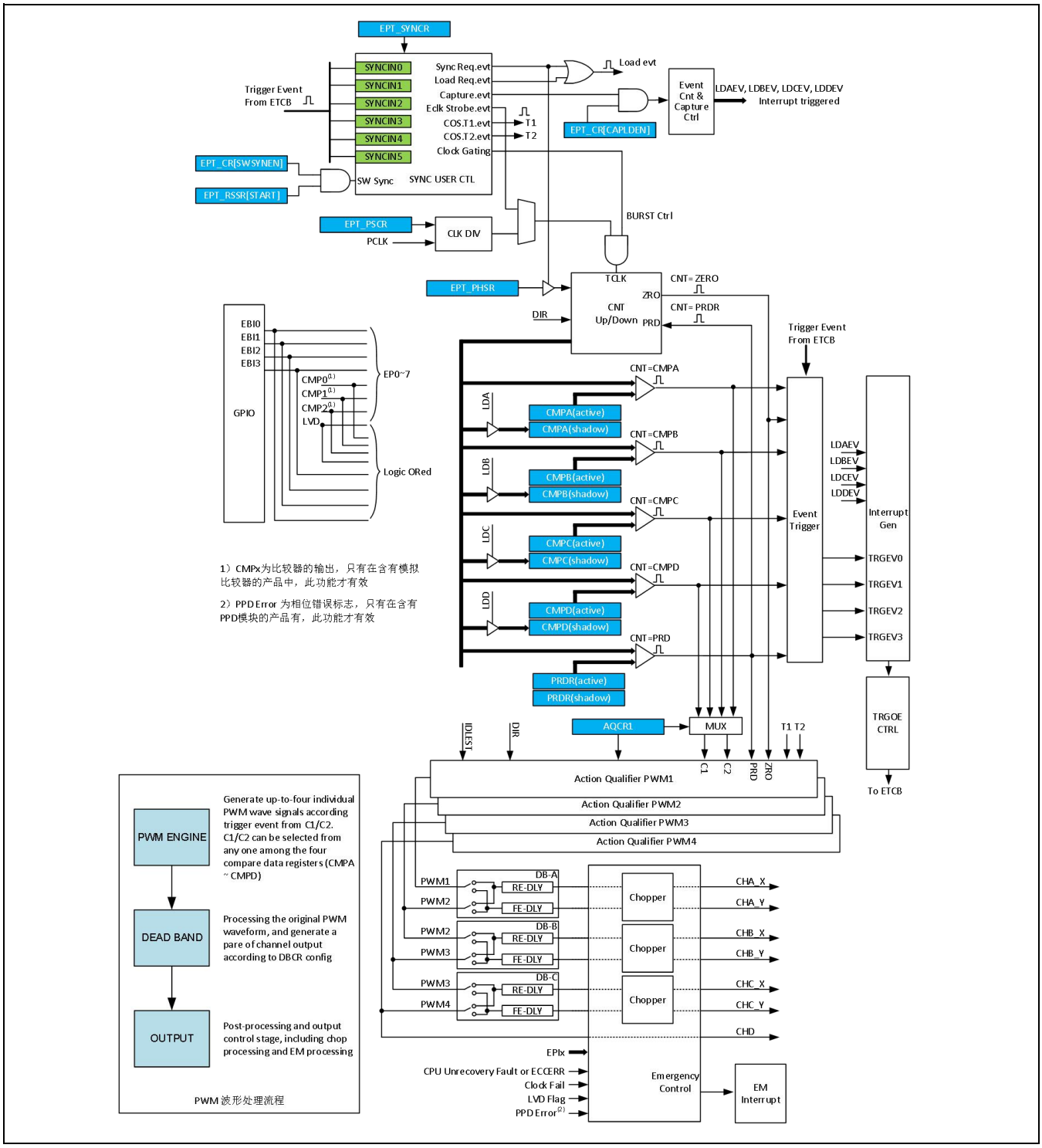


Figure 13-1 模块结构示意图

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。

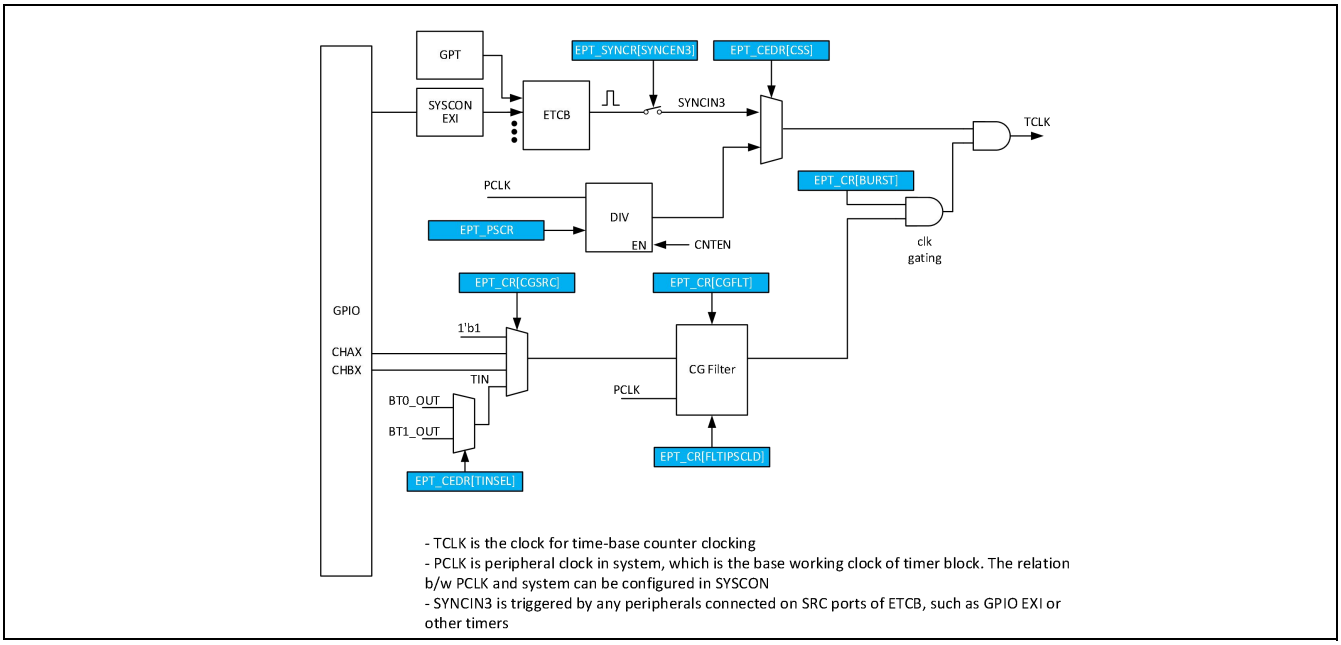


Figure 13-2 时钟控制模块

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0或PRDR时(可通过EPT_CR[PSCLD]设置载入的条件)，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

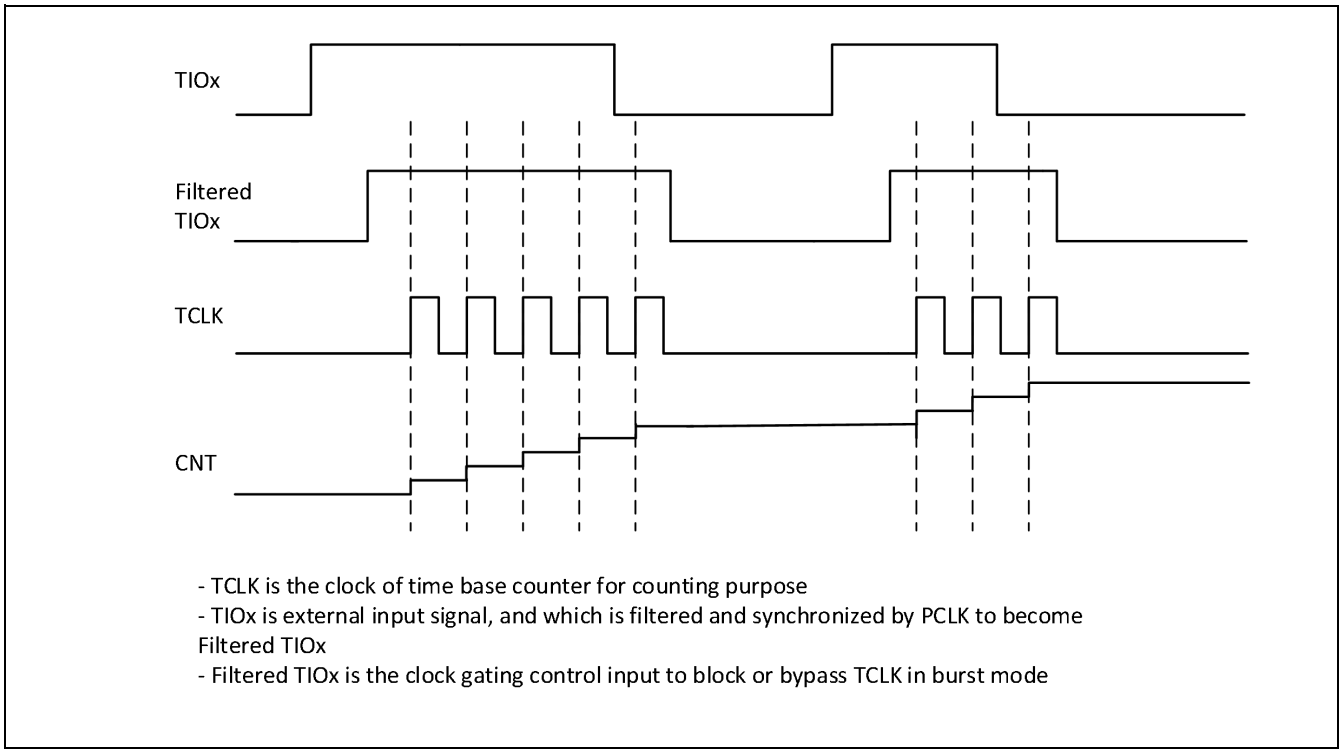


Figure 13- 3 群脉冲时钟模式时序

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。滤波时钟源通过CR[CGSRC], CEDR[TINSEL]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：$T_{dly} = T_{fltclk} \times CR[CGFLT] = ((CEDR[FLTCKPRS]+1)/PCLK) \times CR[CGFLT]$，其中$T_{fltclk}$为滤波器工作时钟的周期。

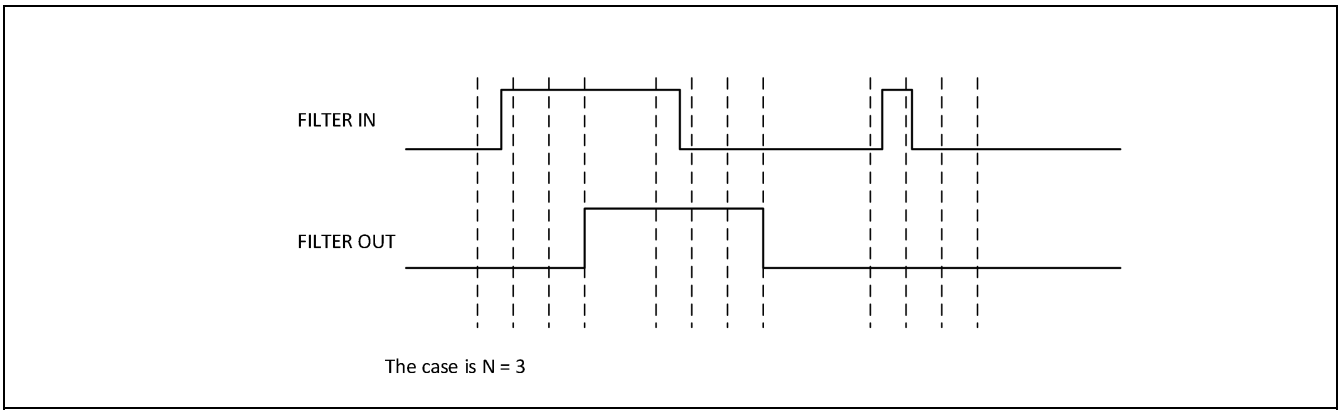


Figure 13-4 CG Filter 数字滤波器的原理

13.3.2 时基控制

13.3.2.1 概述

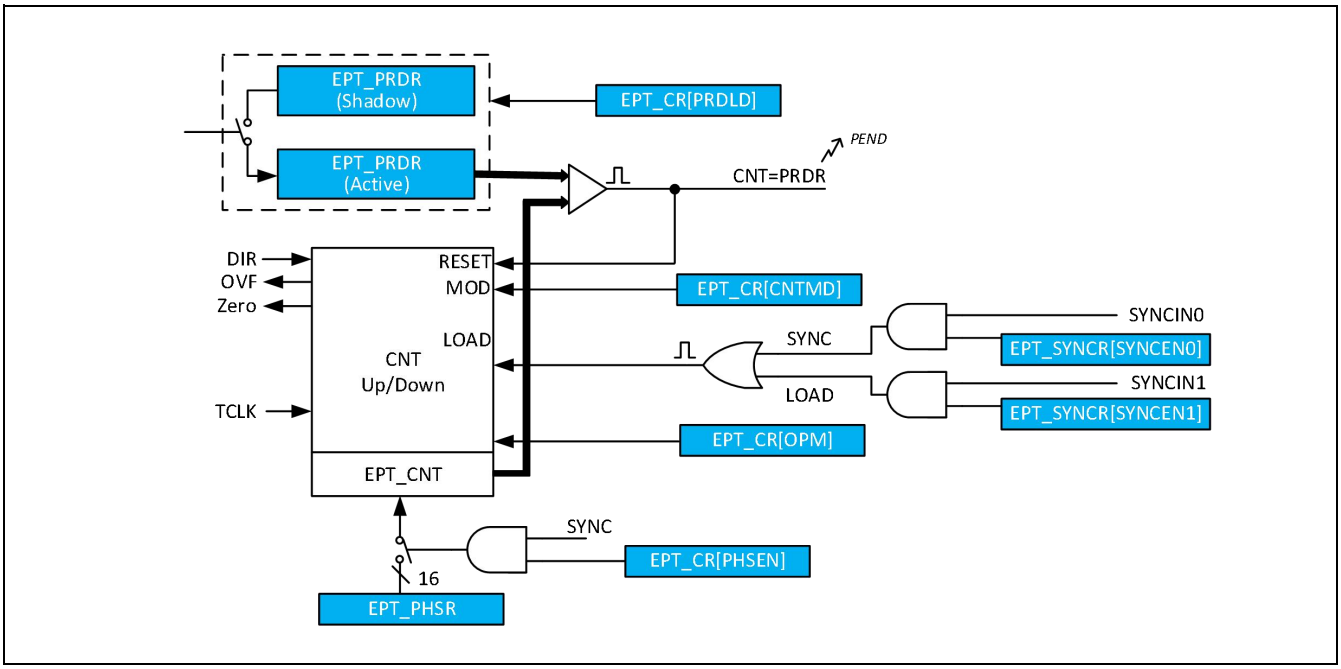


Figure 13-5 计数器时基模块

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SYNCIN0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。

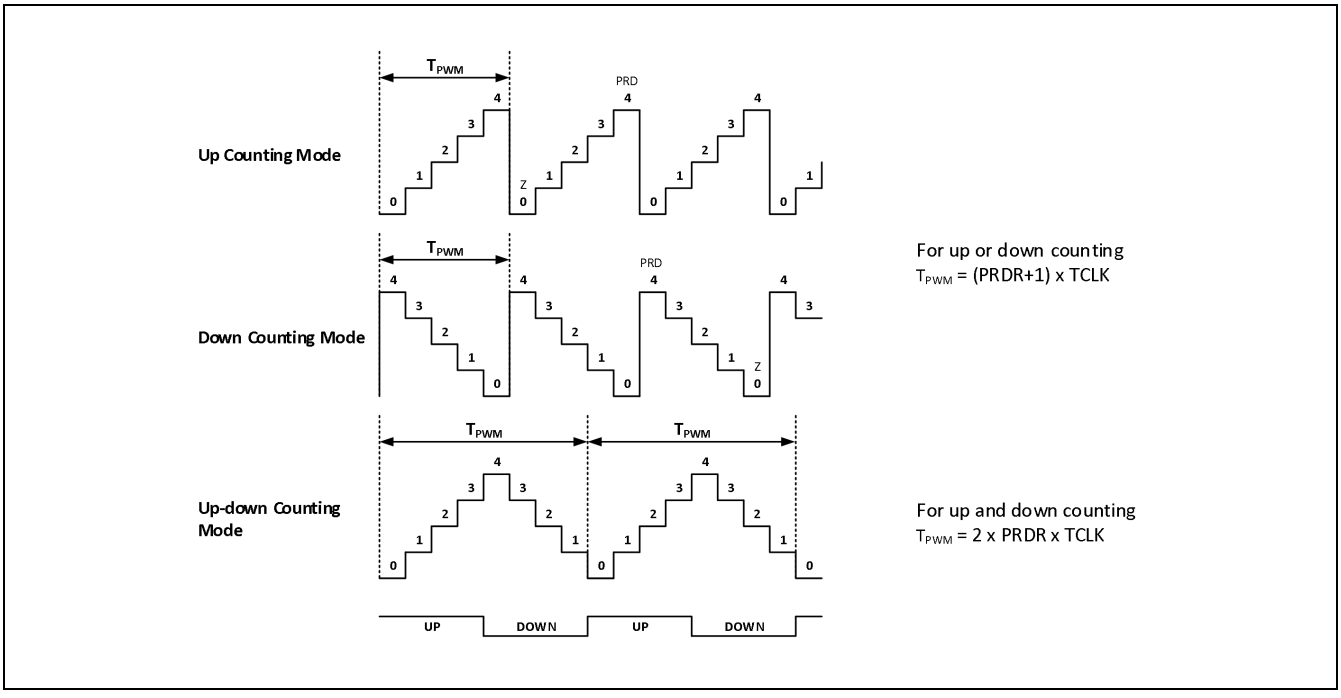


Figure 13-6 计数器工作模式

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（SYNCIN0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。

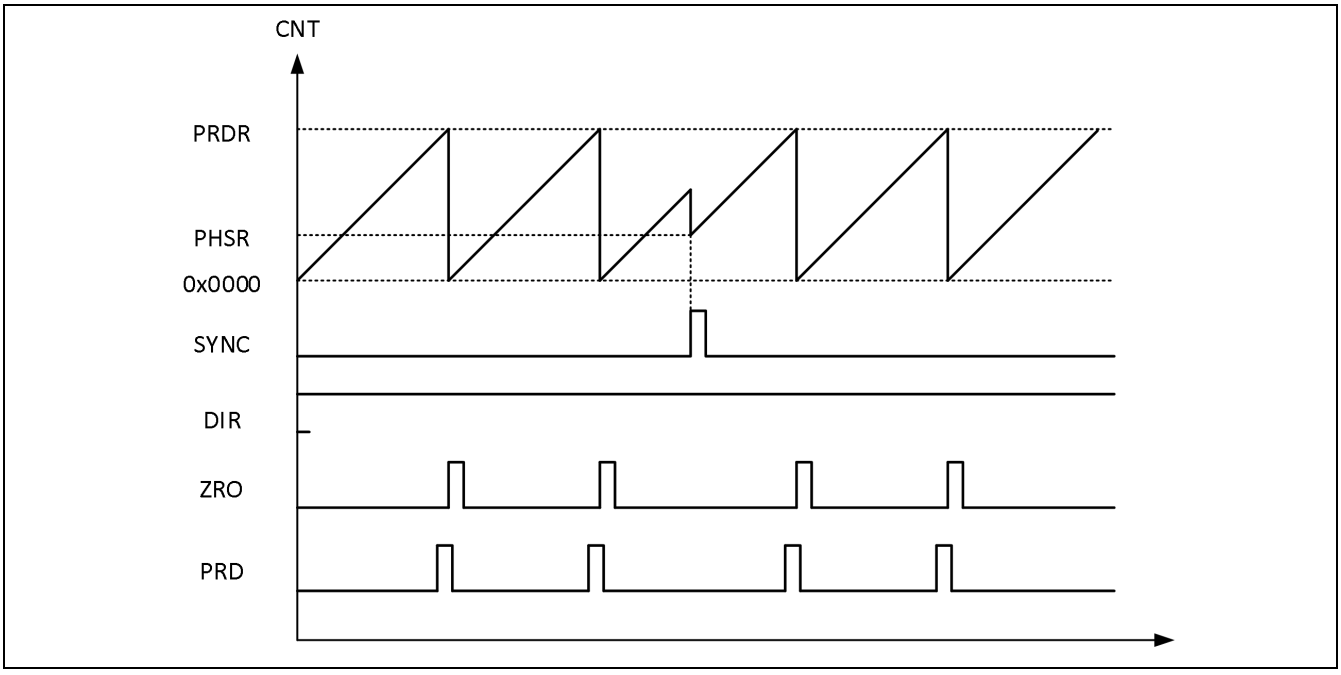


Figure 13-7 递增工作模式

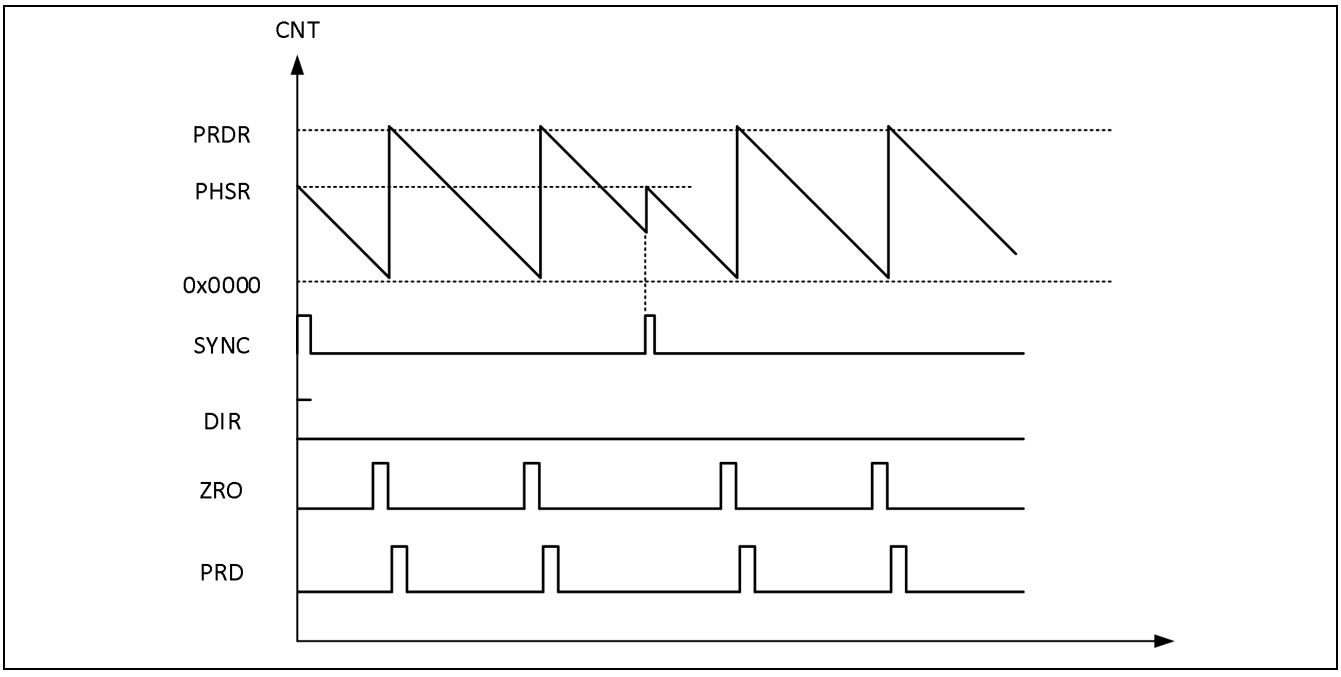


Figure 13-8 递减工作模式

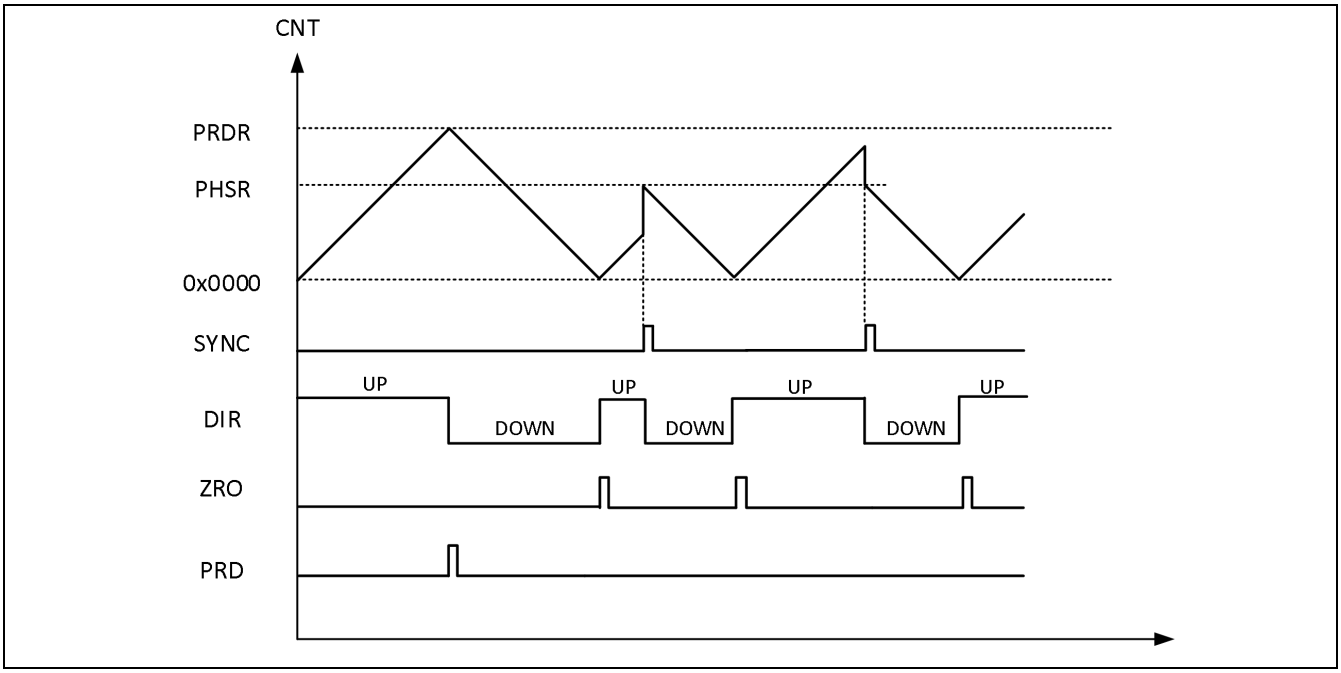


Figure 13-9 递增递减工作模式

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[OSTMD]=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。

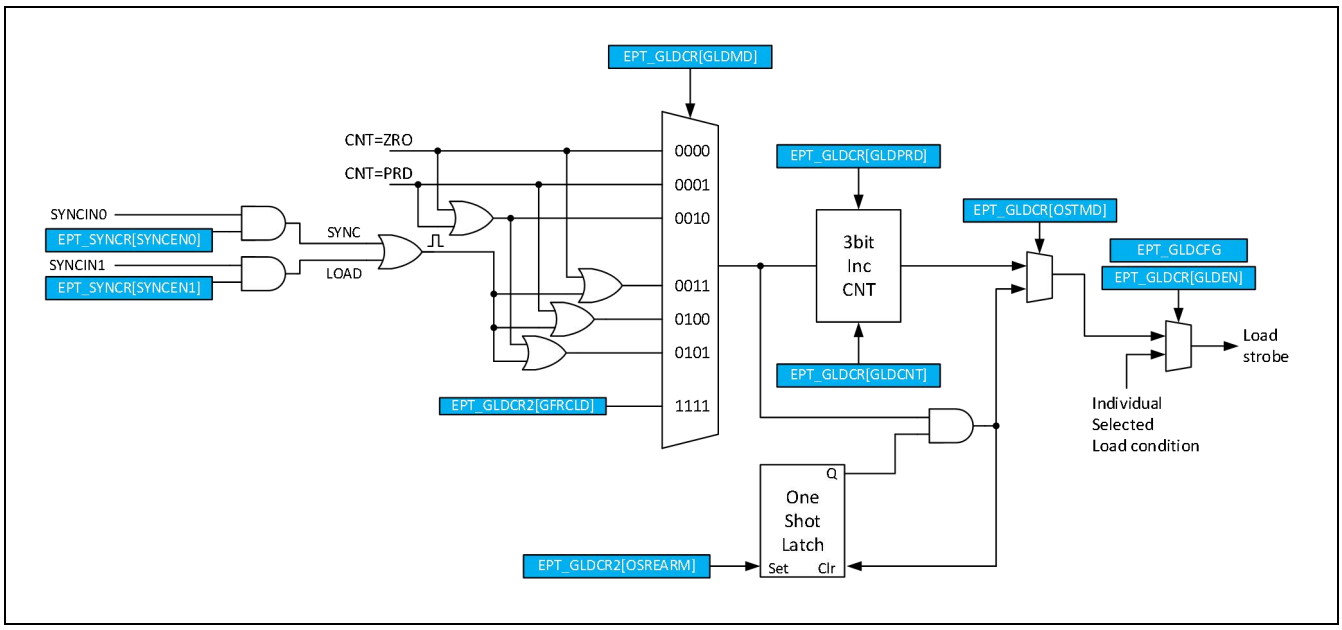


Figure 13- 10 全局载入控制

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。

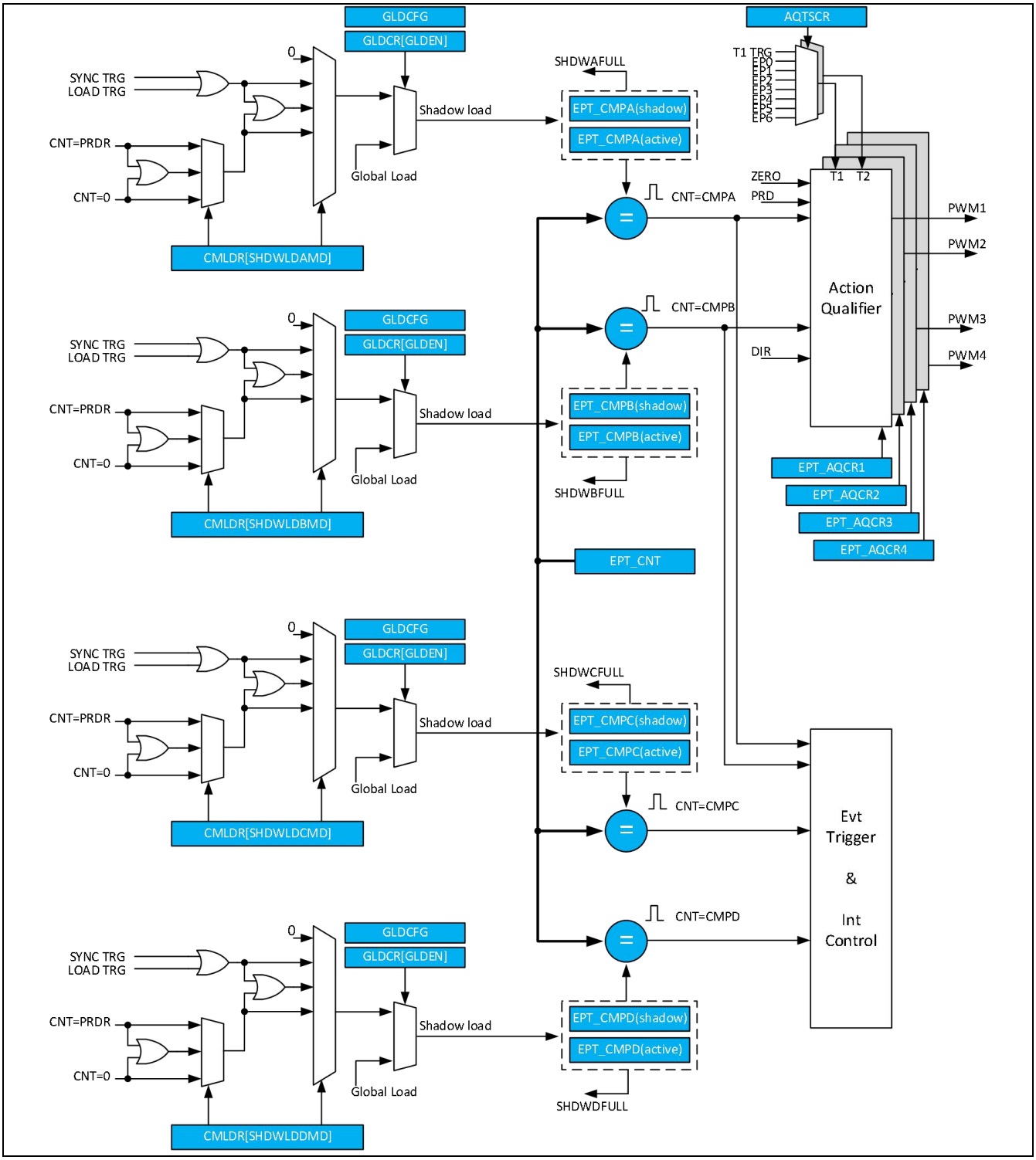


Figure 13- 11 计数器值比较控制

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[SHDWLDxMD]控制位进行设置。影子寄存器的使能可以通过CMPLDR[LDCMPxMD]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[SHDWLDxMD]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。

13.3.3.3 不同计数模式的时序

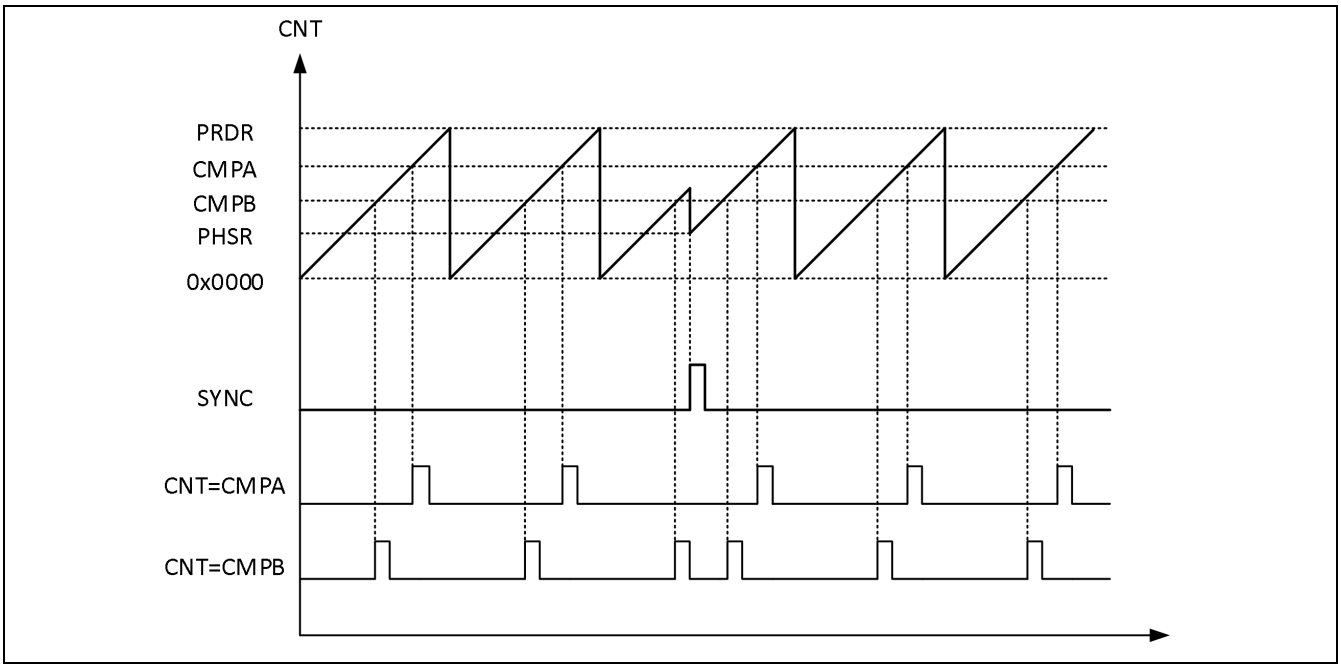


Figure 13-12 递增模式下比较事件产生时序

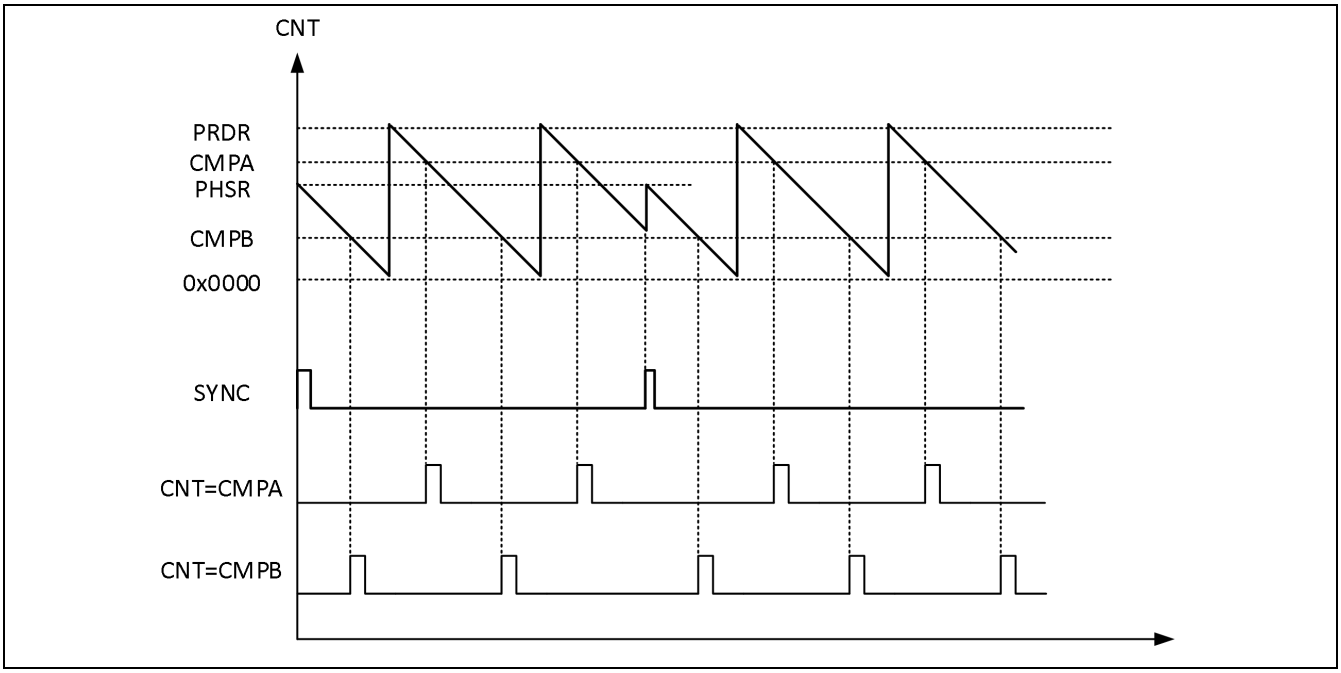


Figure 13-13 递减模式下比较事件产生时序

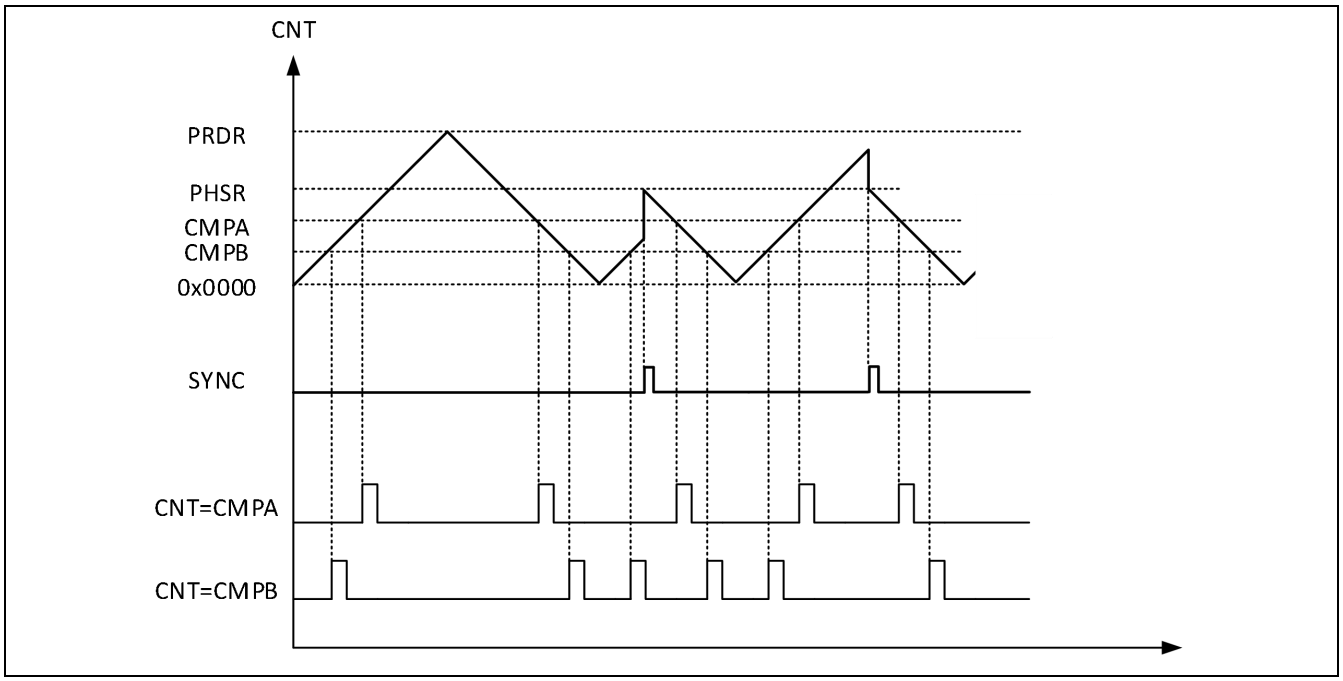


Figure 13-14 递增递减模式下比较事件产生时序

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-counting （计数器值等于C1的参考值，C1参考值由AQCRx[C1SEL]设置）
- CNT = C1 when down-counting （计数器值等于C1的参考值，C1参考值由AQCRx[C1SEL]设置）
- CNT = C2 when up-counting （计数器值等于C2的参考值，C2参考值由AQCRx[C2SEL]设置）
- CNT = C2 when down-counting （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-counting （通过AQTSCR选择T1事件触发源）

- T1 事件 when down-counting （通过AQTSCR选择T1事件触发源）
- T2 事件 when up-counting （通过AQTSCR选择T2事件触发源）
- T2 事件 when down-counting （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。

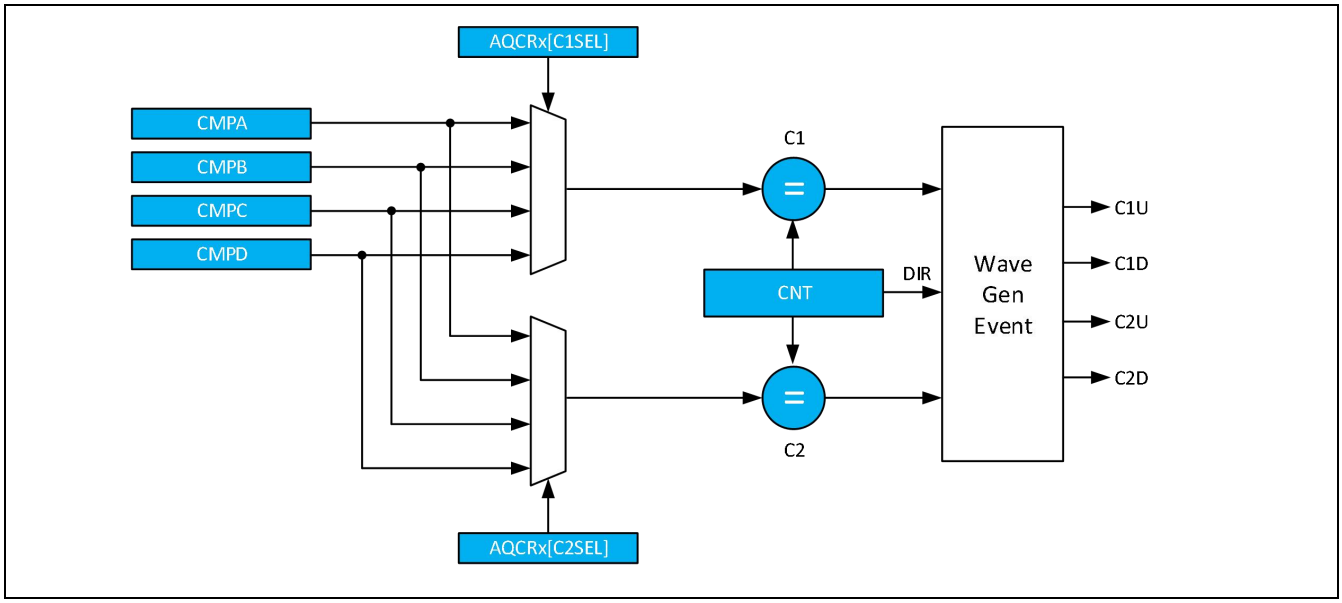


Figure 13-15 C1和C2选择控制

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 13-2 各种在PWMx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。

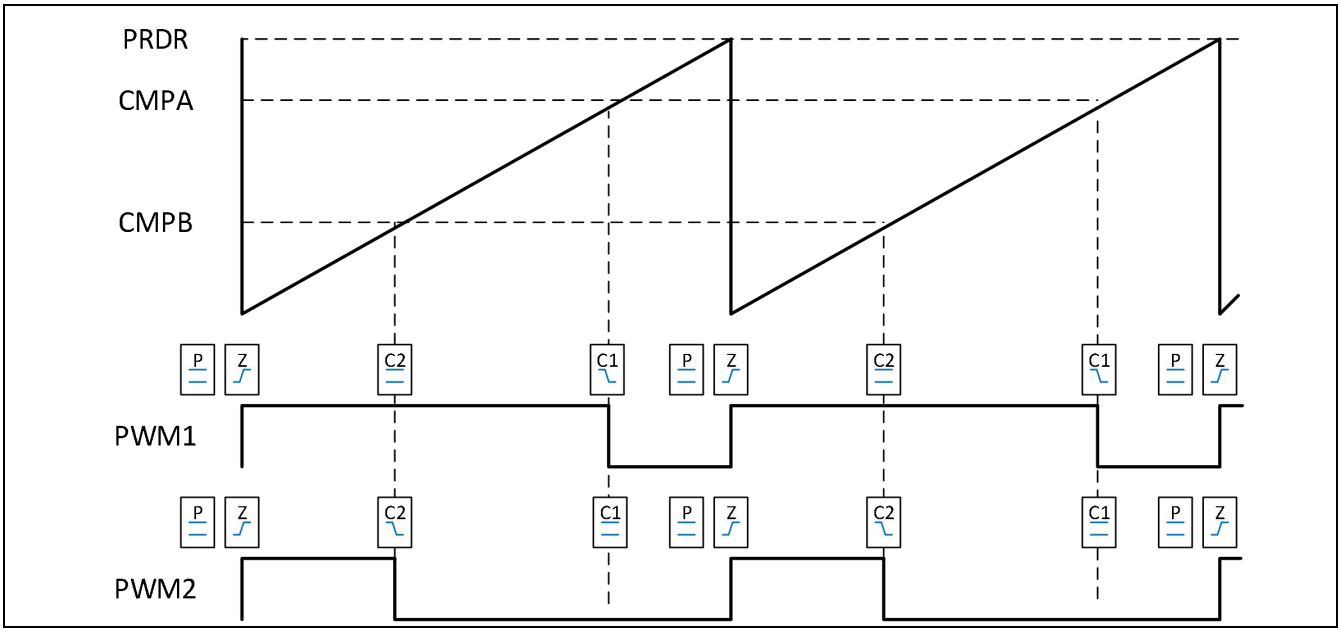


Figure 13-16 递增单沿，非对称波形输出

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。

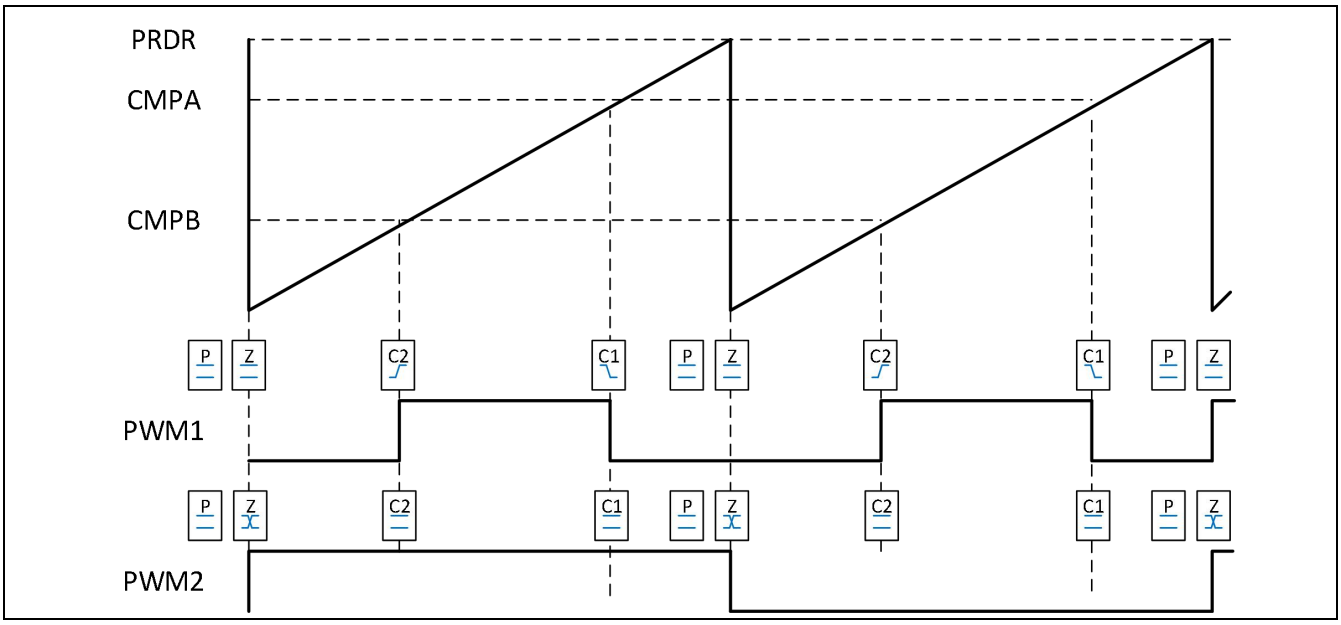


Figure 13-17 递增，脉冲定位非对称波形输出

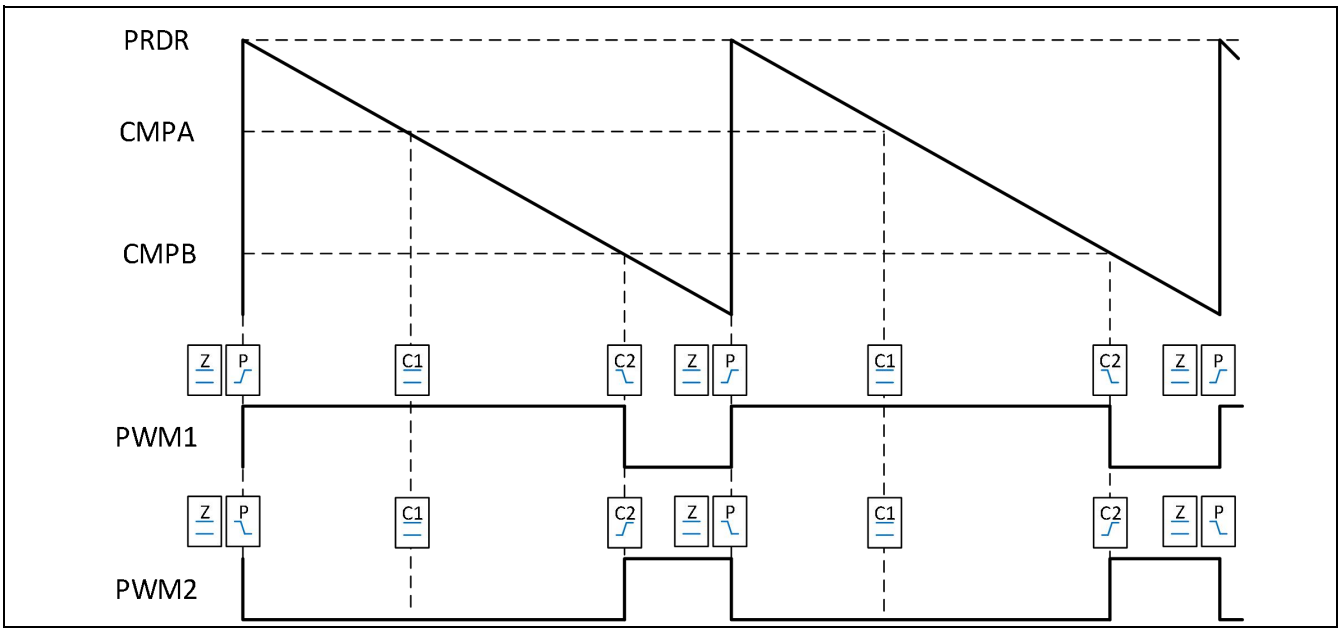


Figure 13-18 递减单沿，非对称波形输出

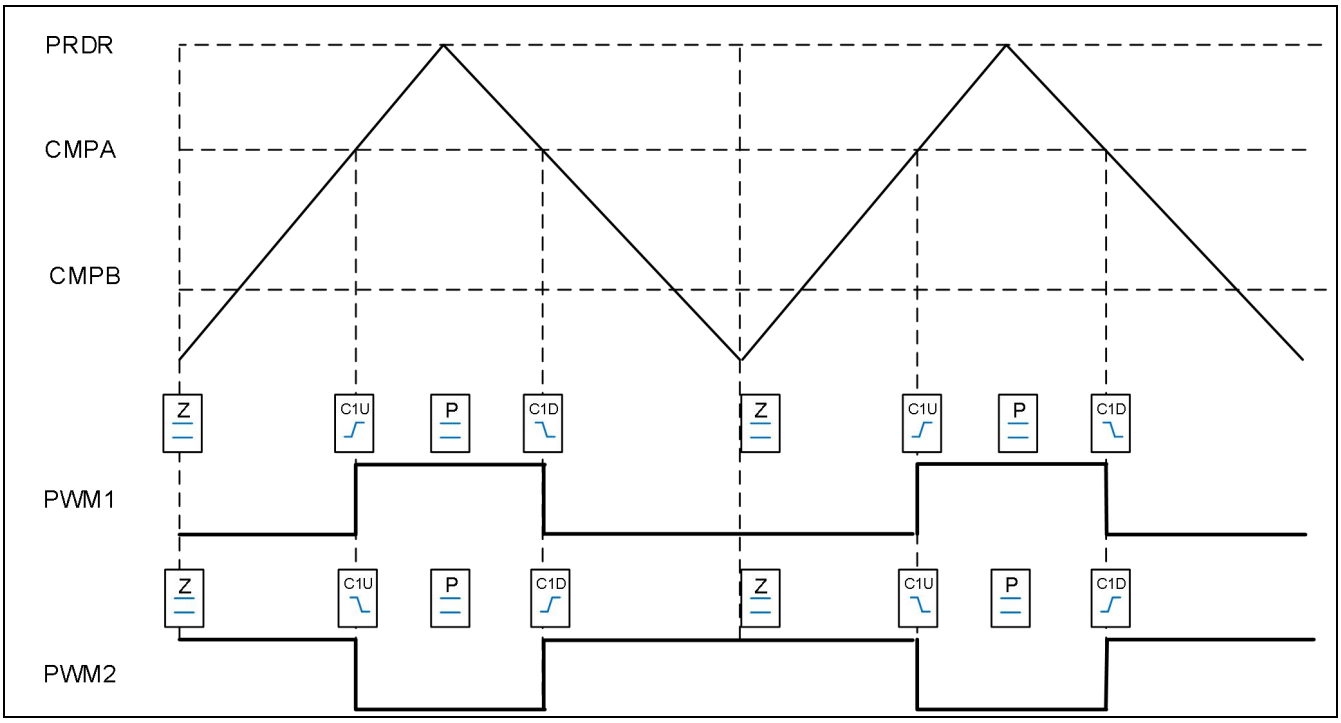


Figure 13-19 递增递减，双沿对称波形输出

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 13-3 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 13-4 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 13- 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[ACT1/2/3/4]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。

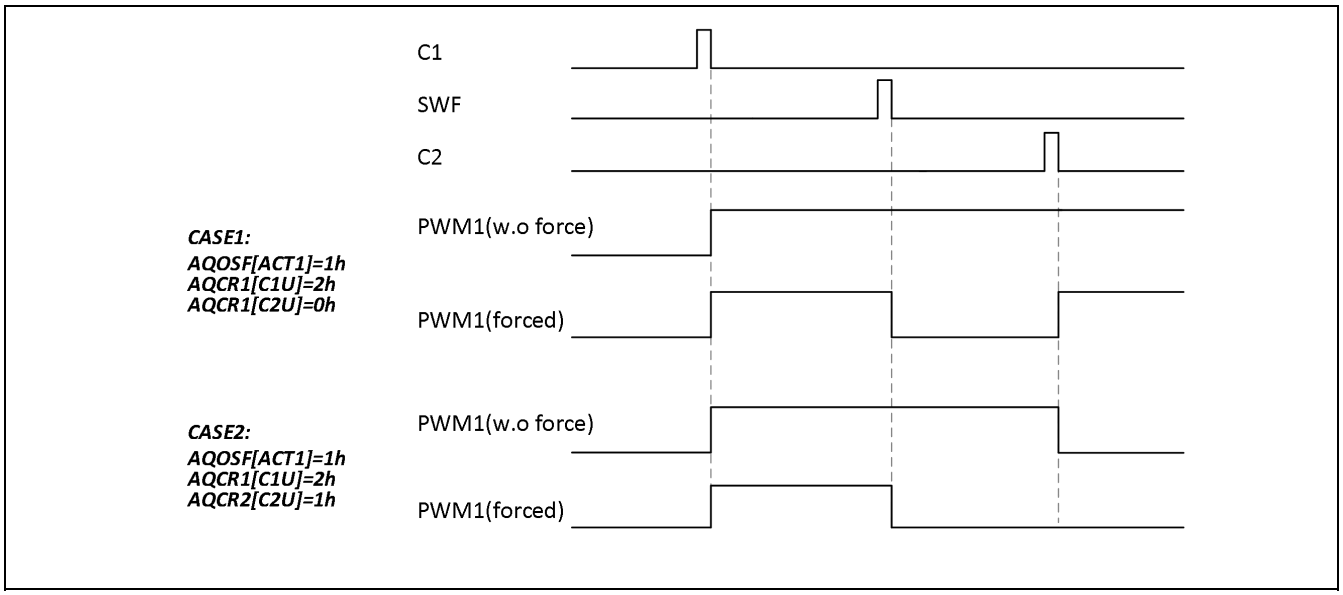


Figure 13- 20 一次性软件强制输出

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。

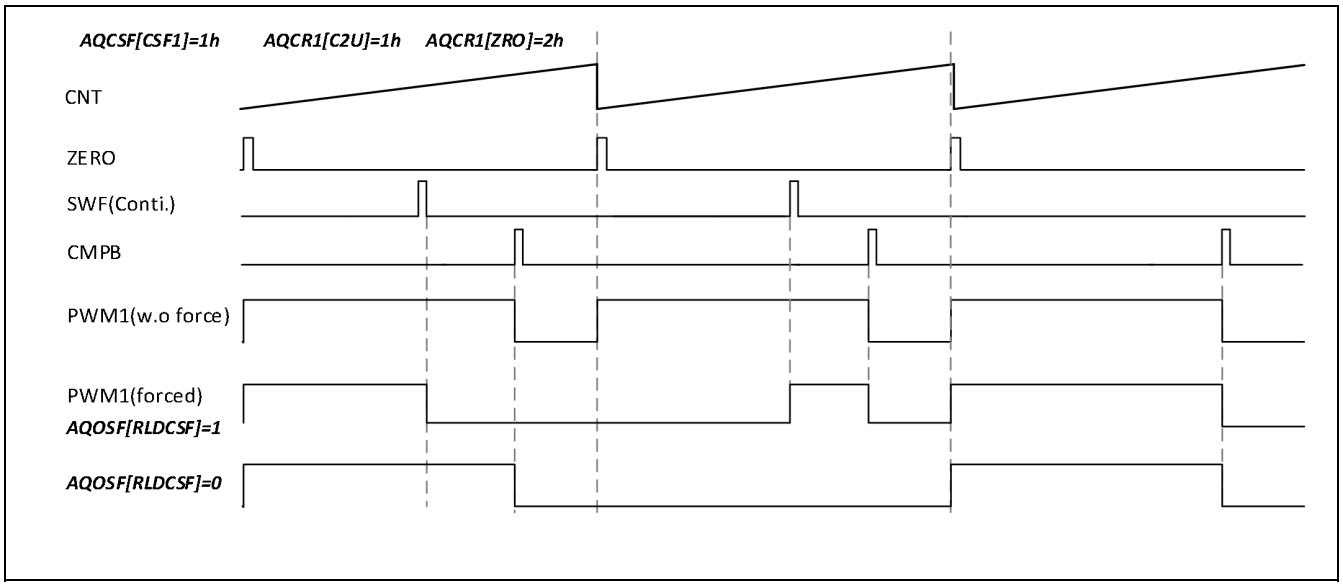


Figure 13- 21 持续性软件强制输出

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）。

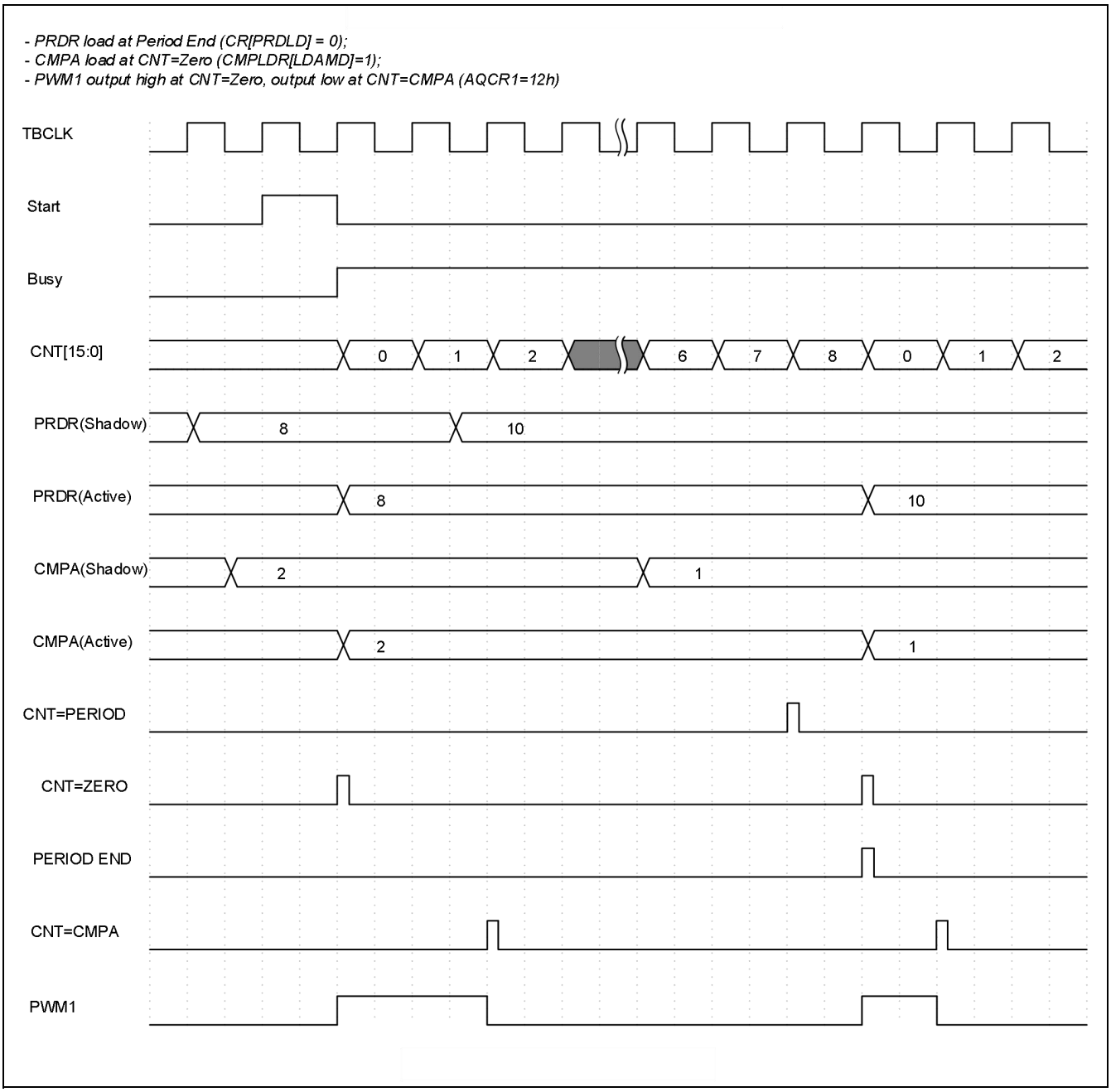


Figure 13- 22 递增单比较值时，非对称波形输出

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）。

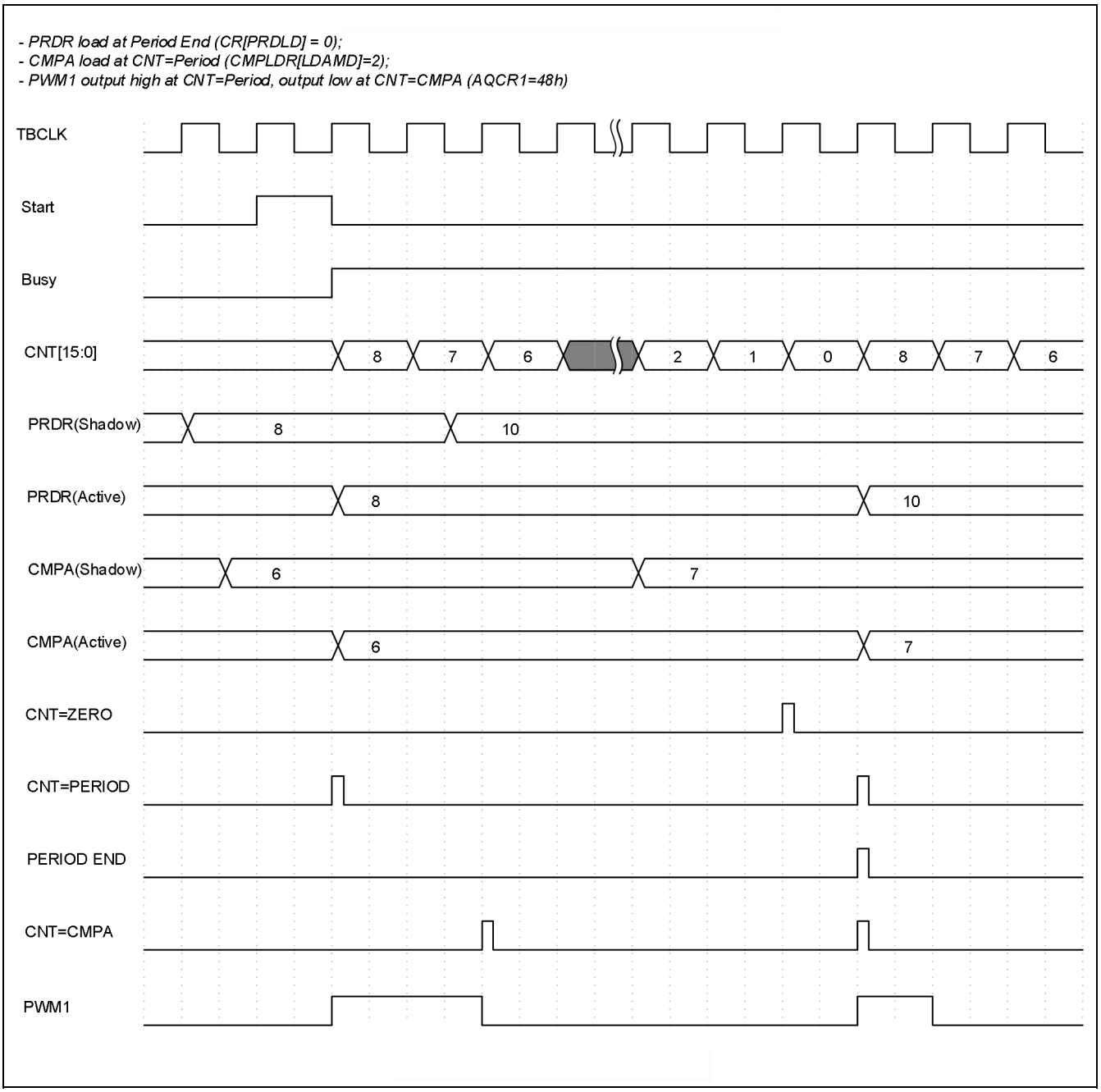


Figure 13- 23 递减单比较值时，非对称波形输出

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。

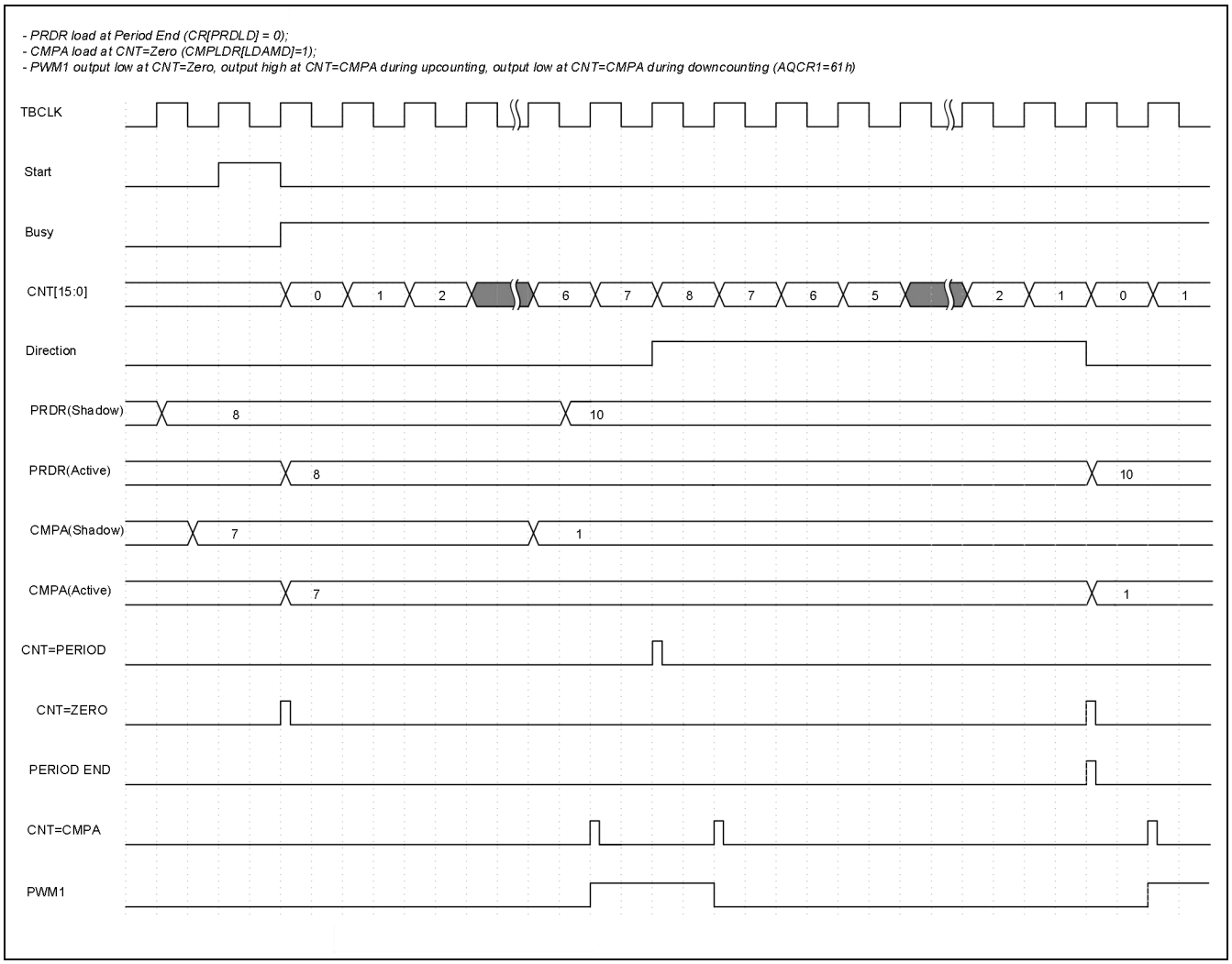


Figure 13-24 递增递减单比较值时，对称波形输出

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。

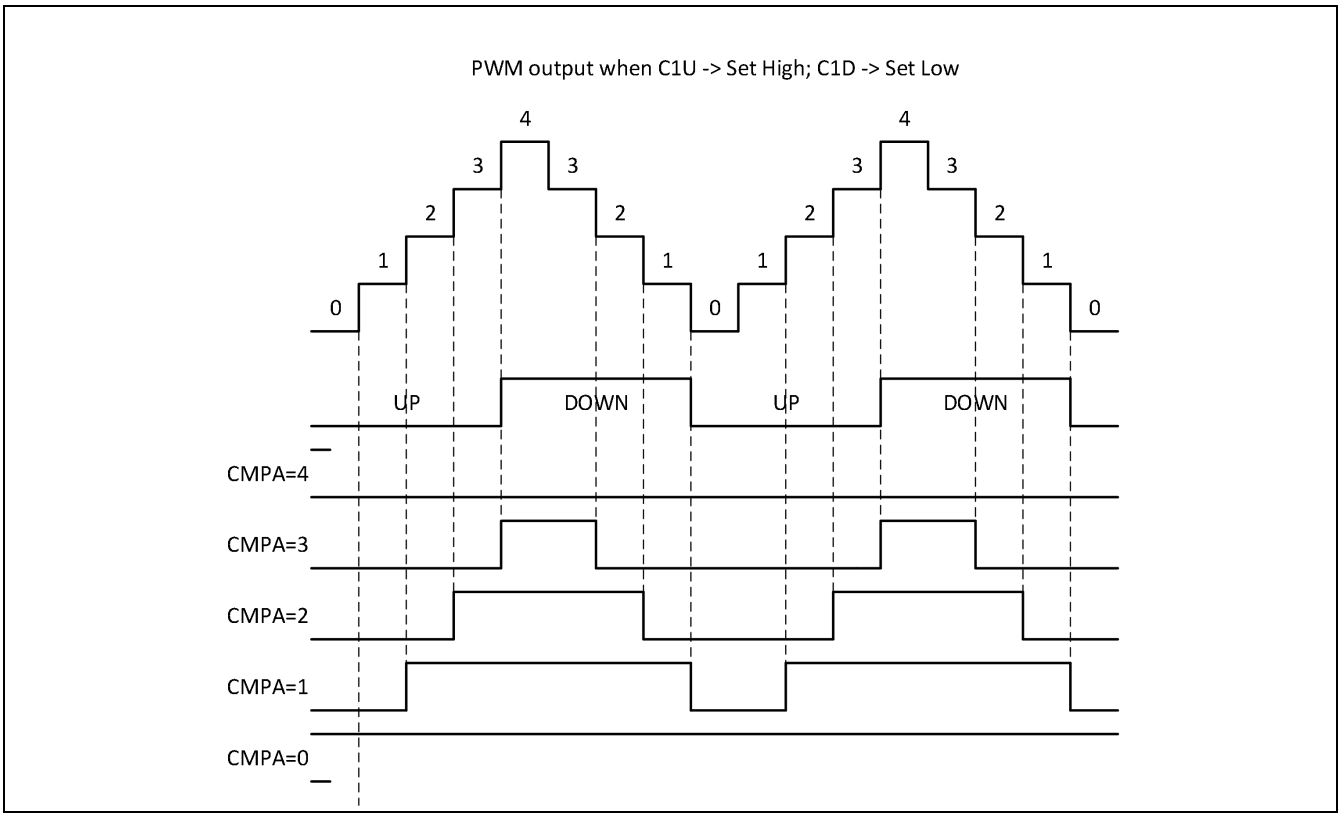


Figure 13-25 递增递减单比较值时，对称波形输出

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。

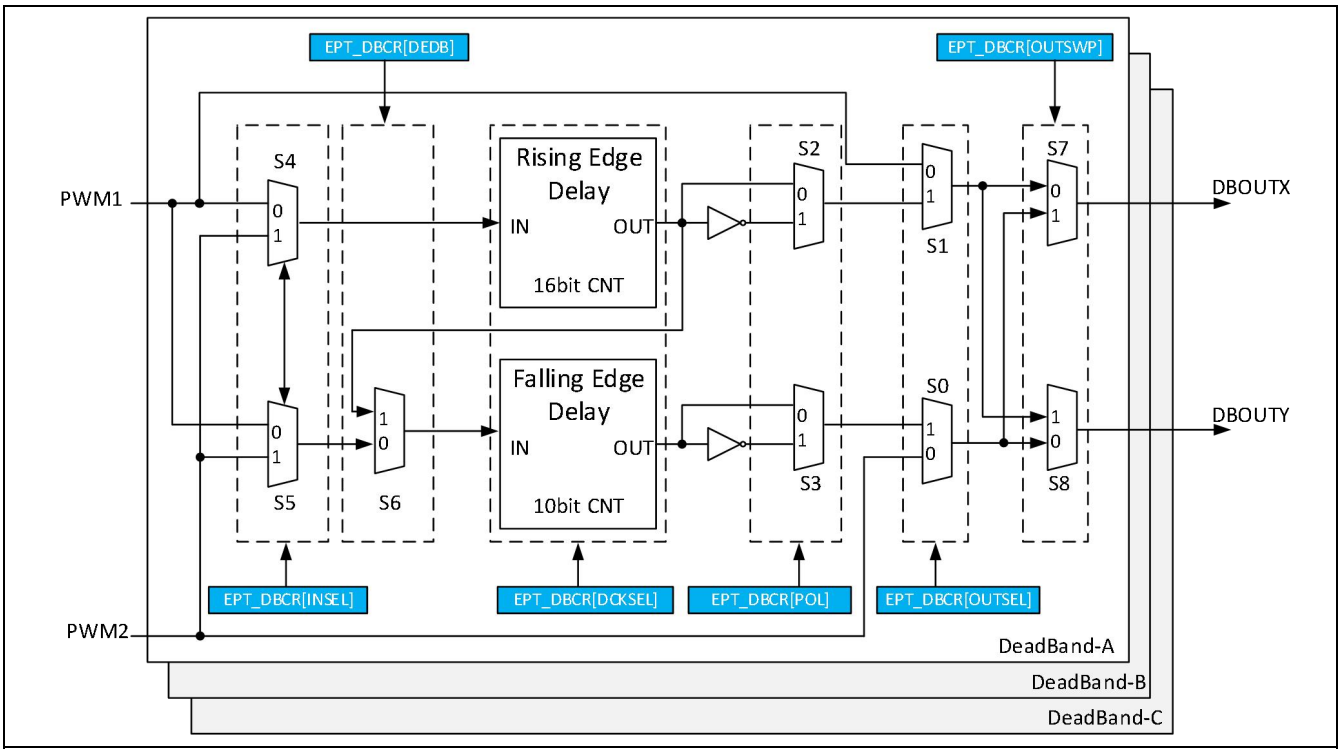


Figure 13-26 死区控制模块

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 13-6 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

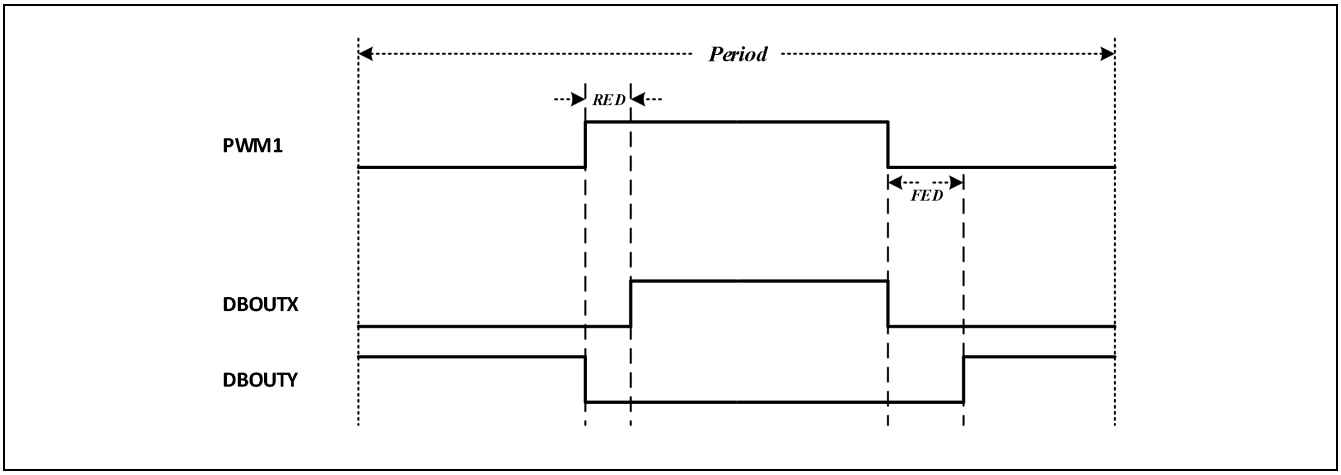


Figure 13-27 模式2：高电平有效的死区互补输出举例

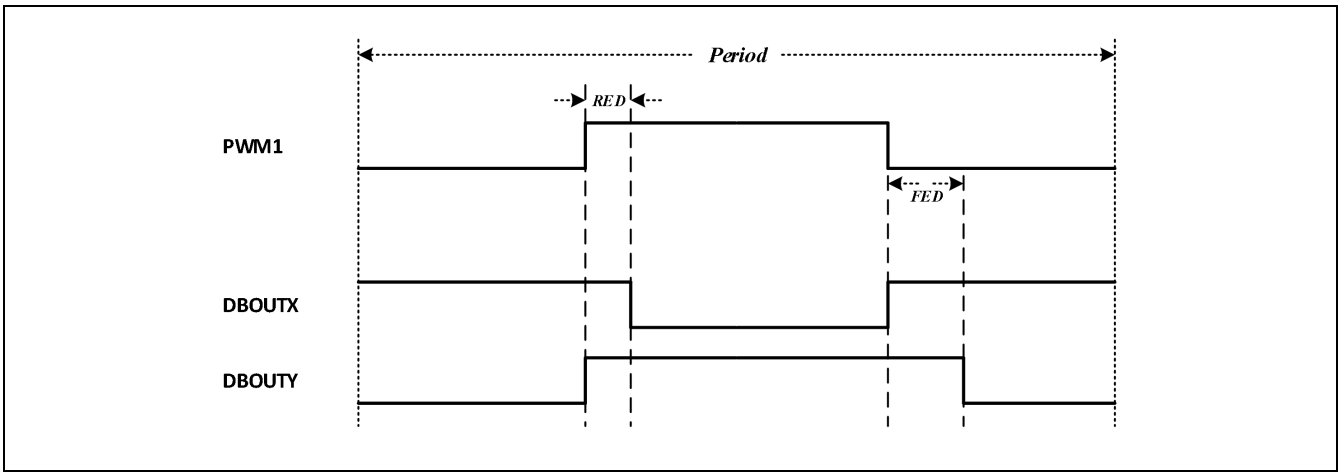


Figure 13-28 模式3：低电平有效的死区互补输出举例

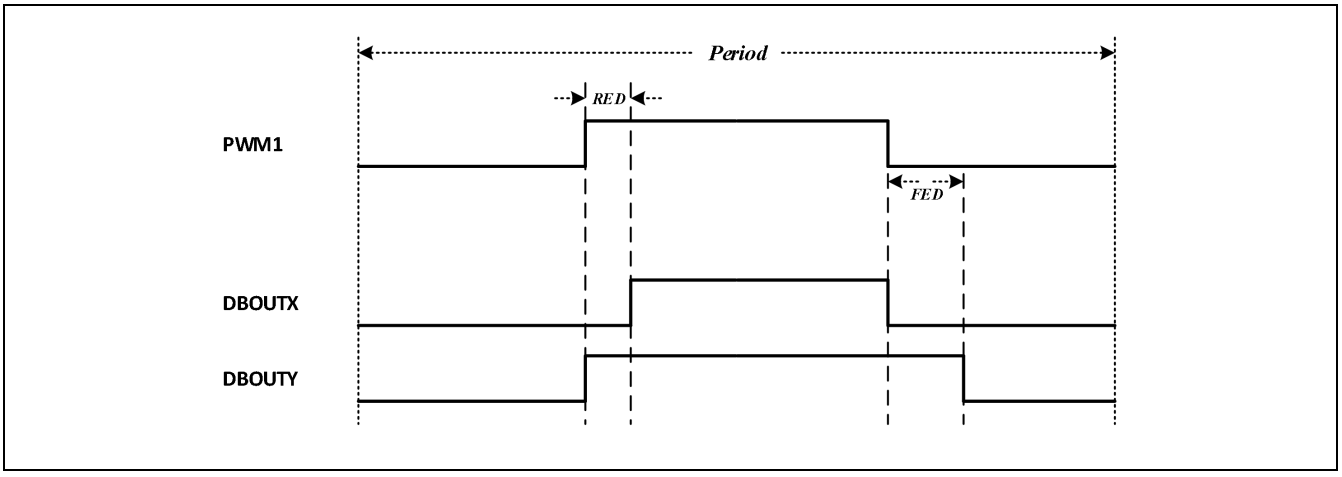


Figure 13-29 模式4：高低电平有效死区输出举例

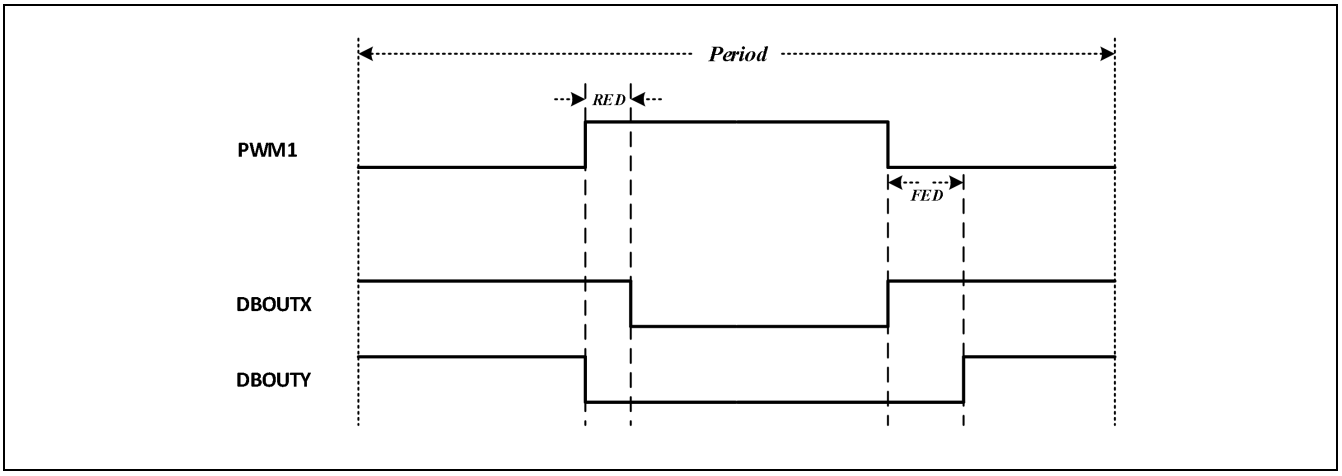


Figure 13-30 模式5：低高电平有效死区输出举例

13.3.6 斩波模块

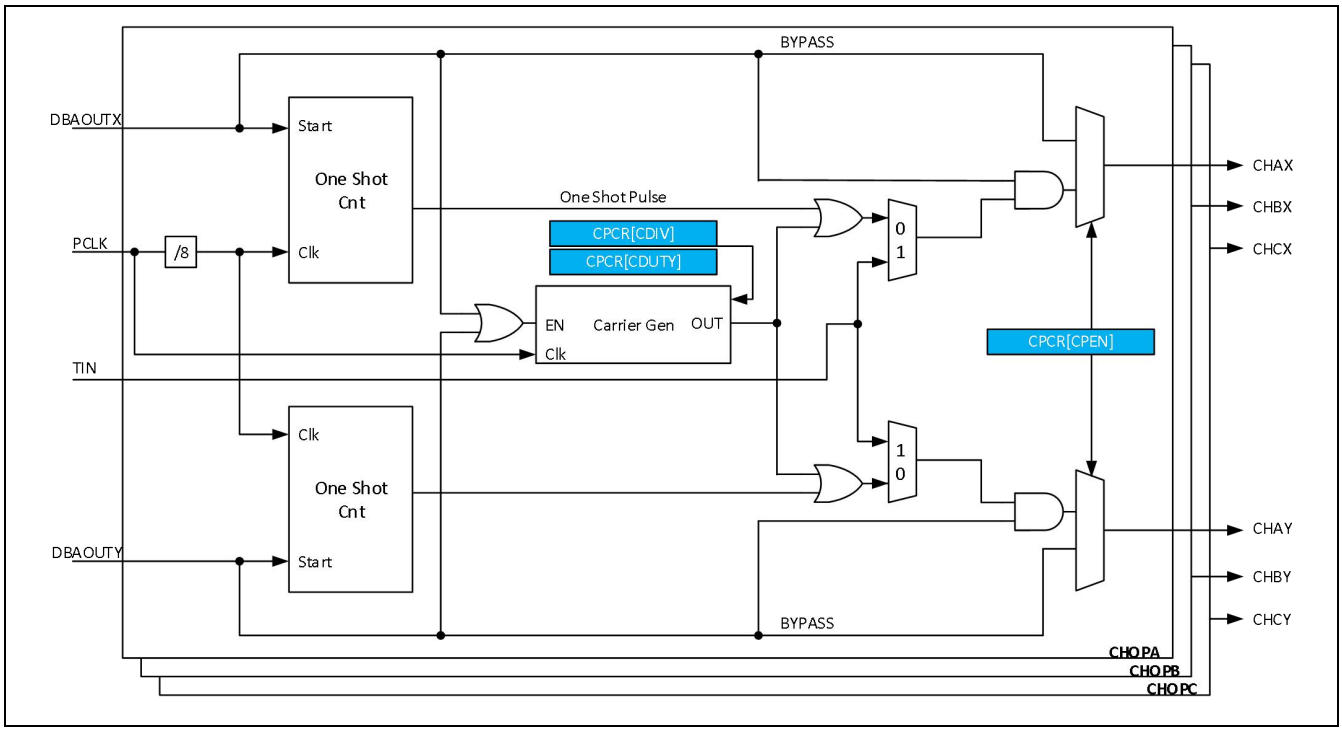


Figure 13-31 斩波使能模块

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}）。

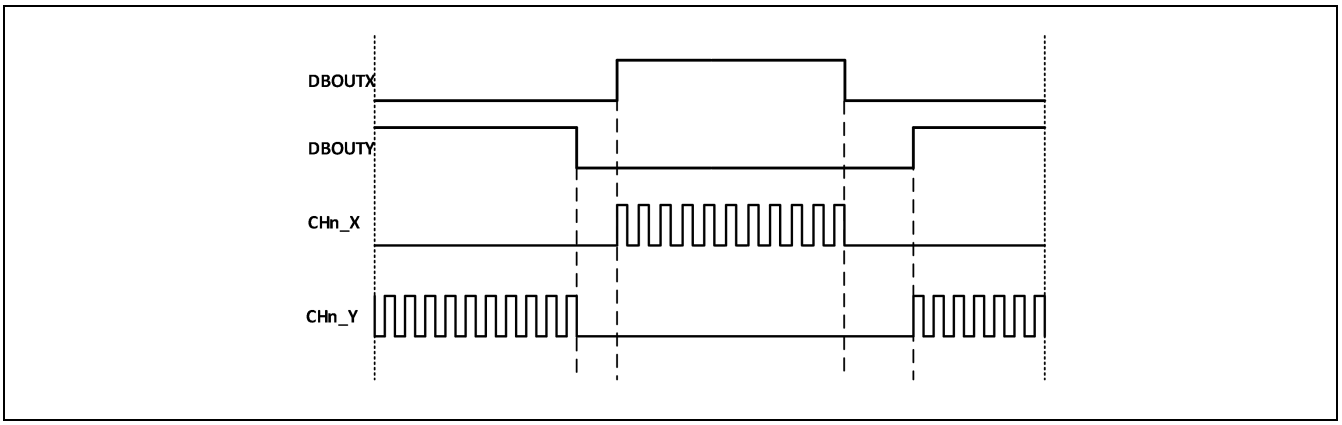


Figure 13-32 简单的斩波输出

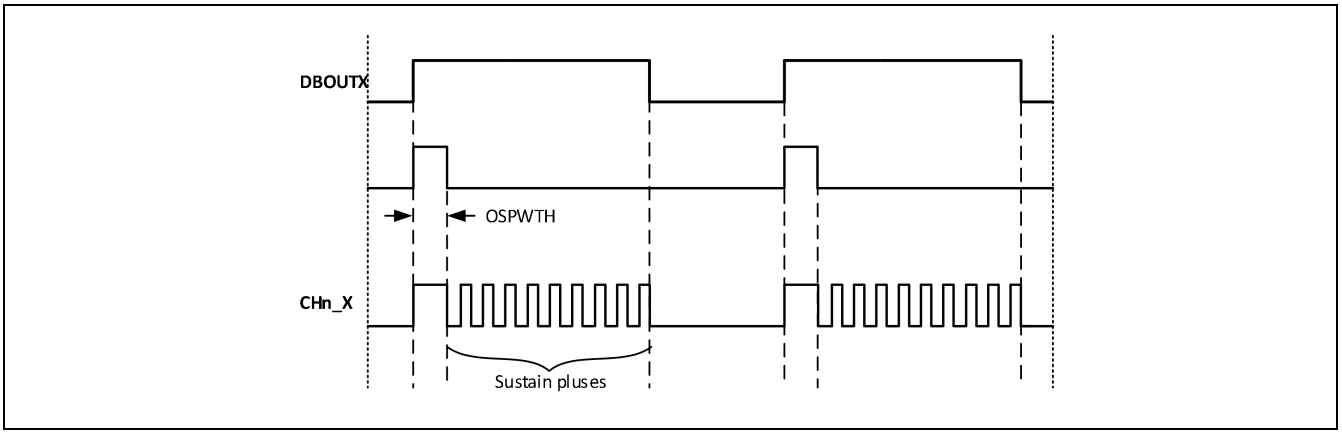


Figure 13-33 首脉冲扩展后的波形

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BT0或BT1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。

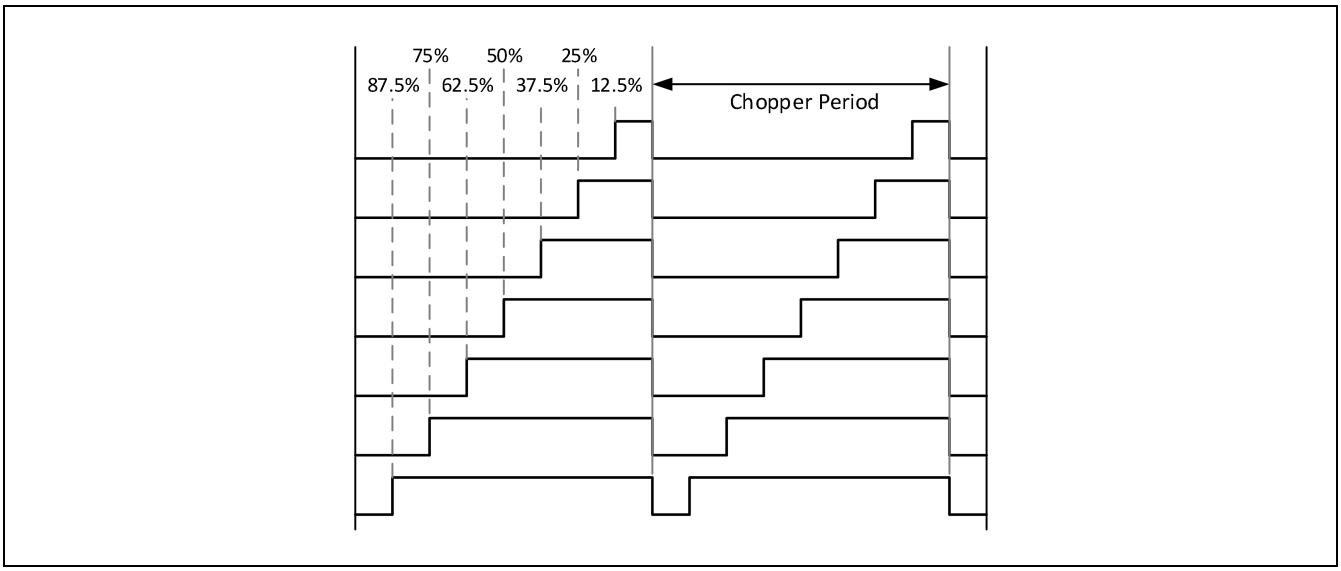


Figure 13-34 载波的占空比

13.3.7 紧急模式控制

13.3.7.1 紧急模式工作机制

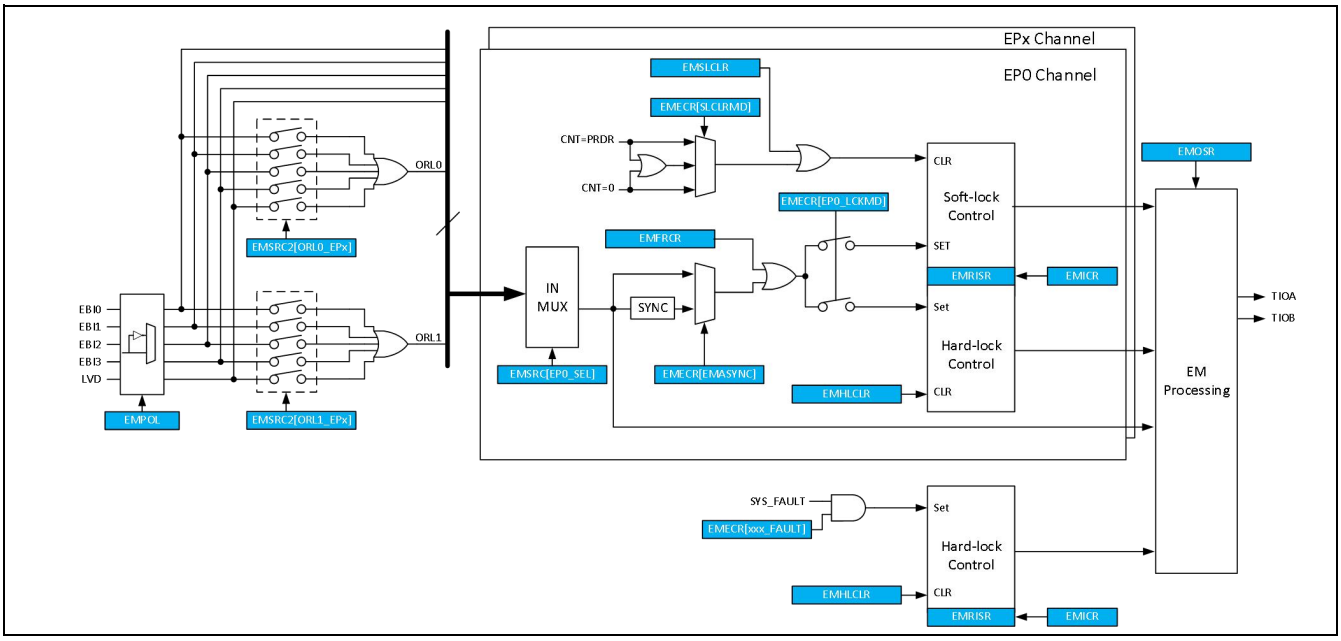


Figure 13-35紧急模式控制模块

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，需要经过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。

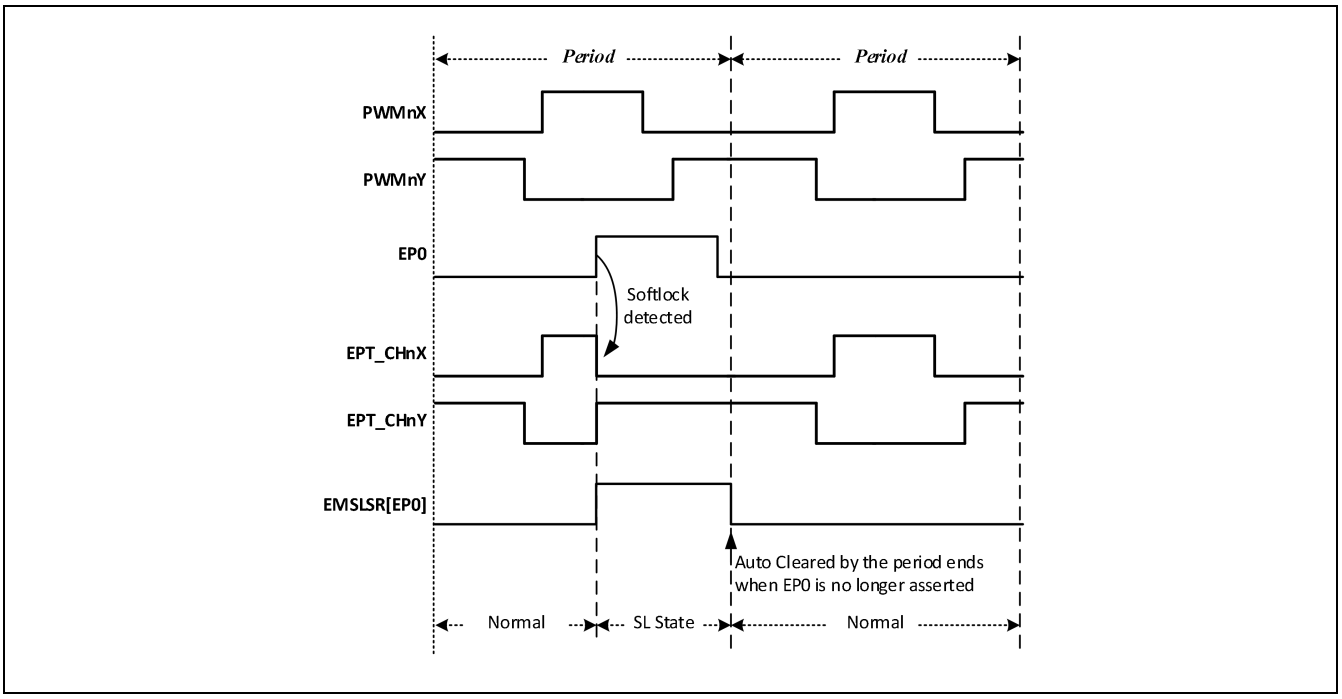


Figure 13-36 软锁止模式

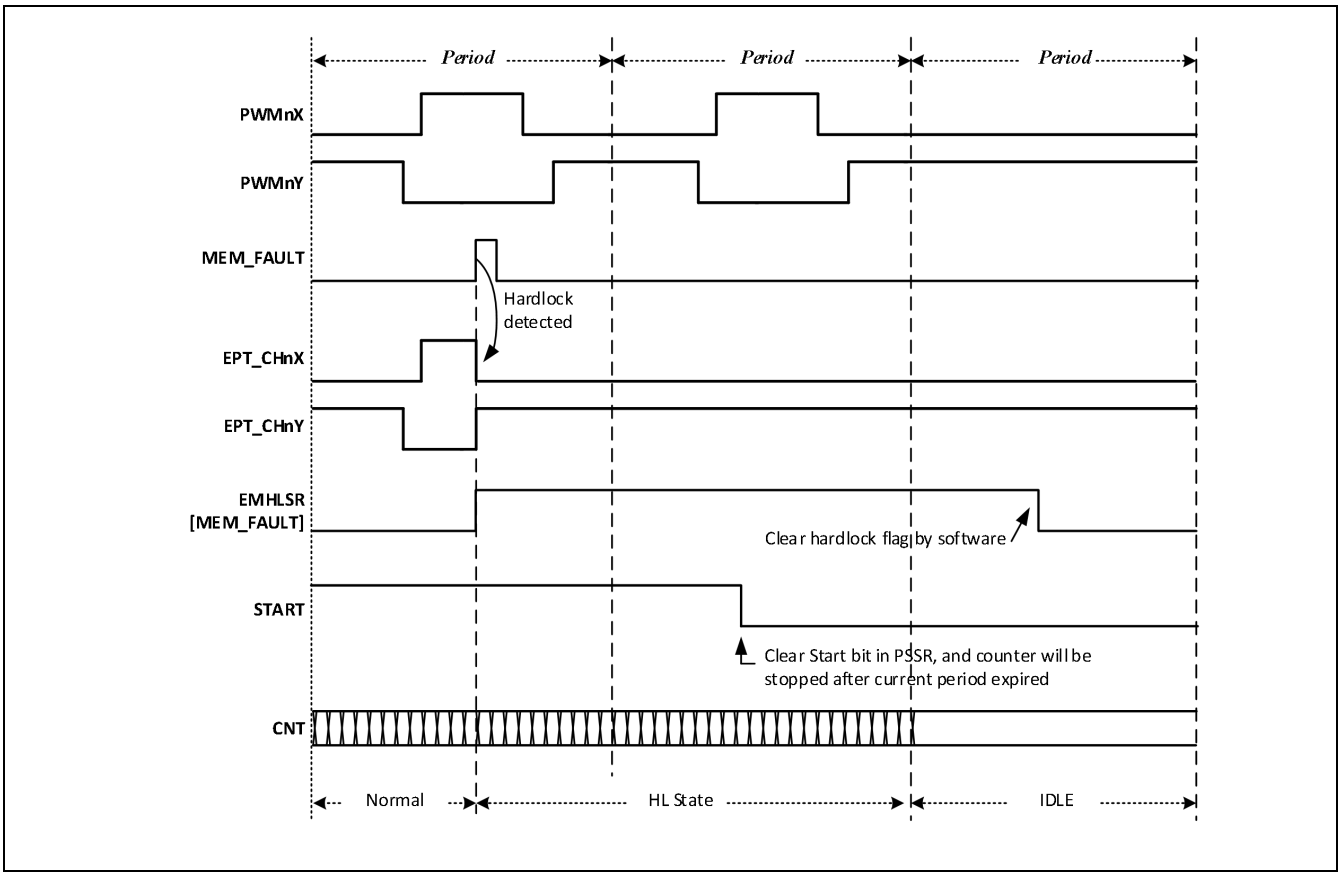


Figure 13- 37 硬锁止模式

13.3.7.2 紧急模式中断

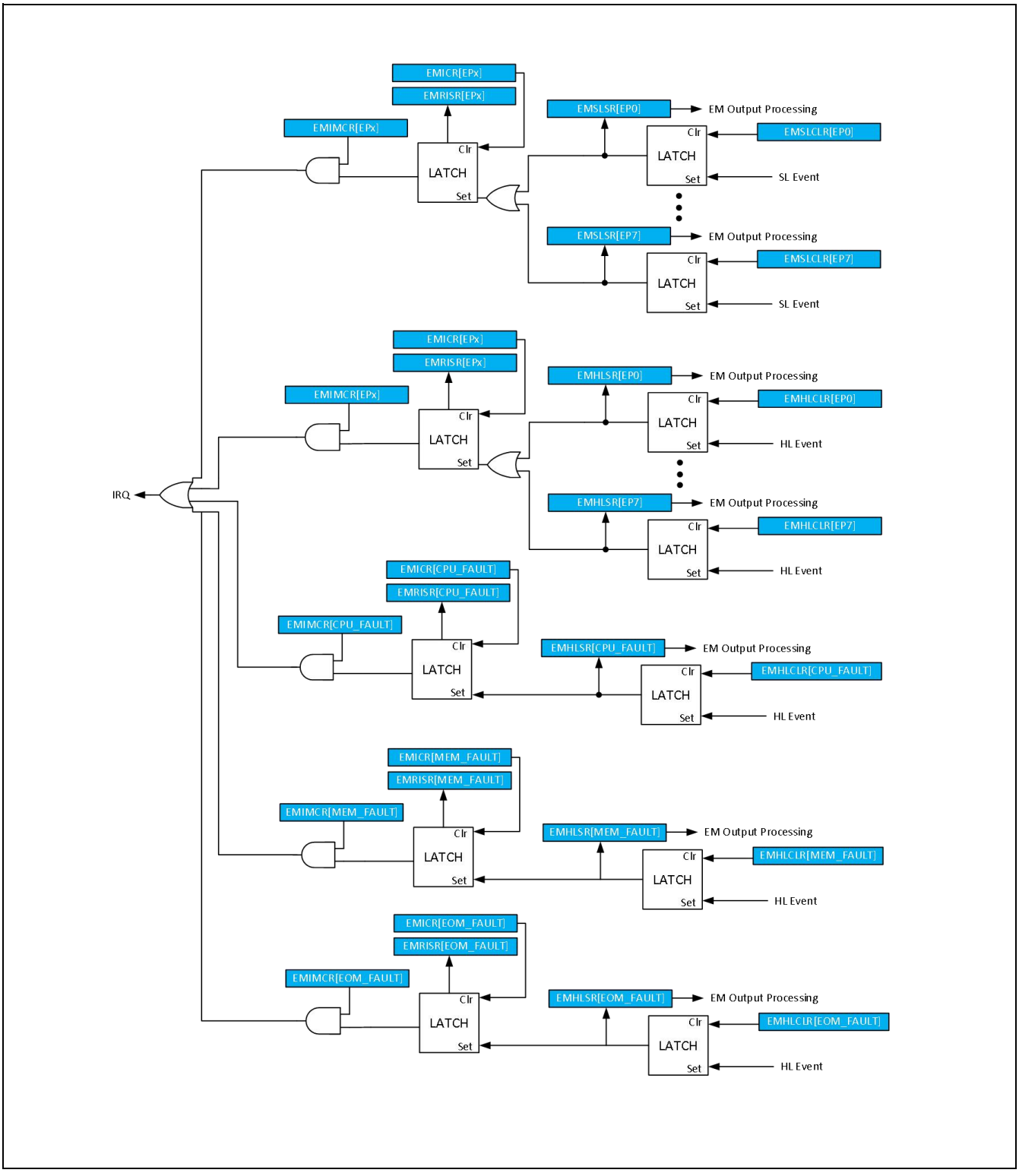


Figure 13-38 紧急模式中断

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

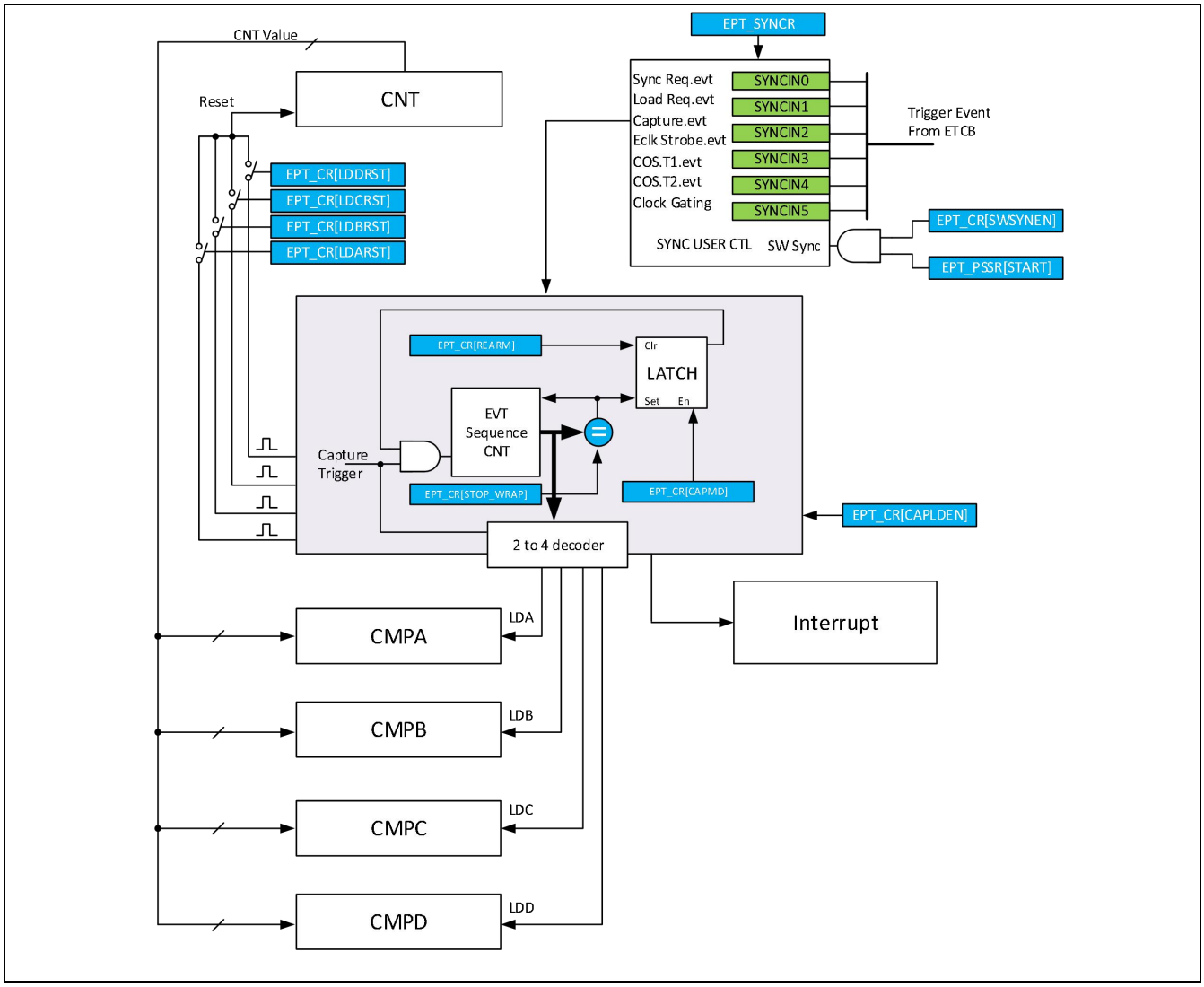


Figure 13- 39 捕捉模式结构框图

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在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 13- 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[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。（Figure13-40）

- **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。（Figure13-41）

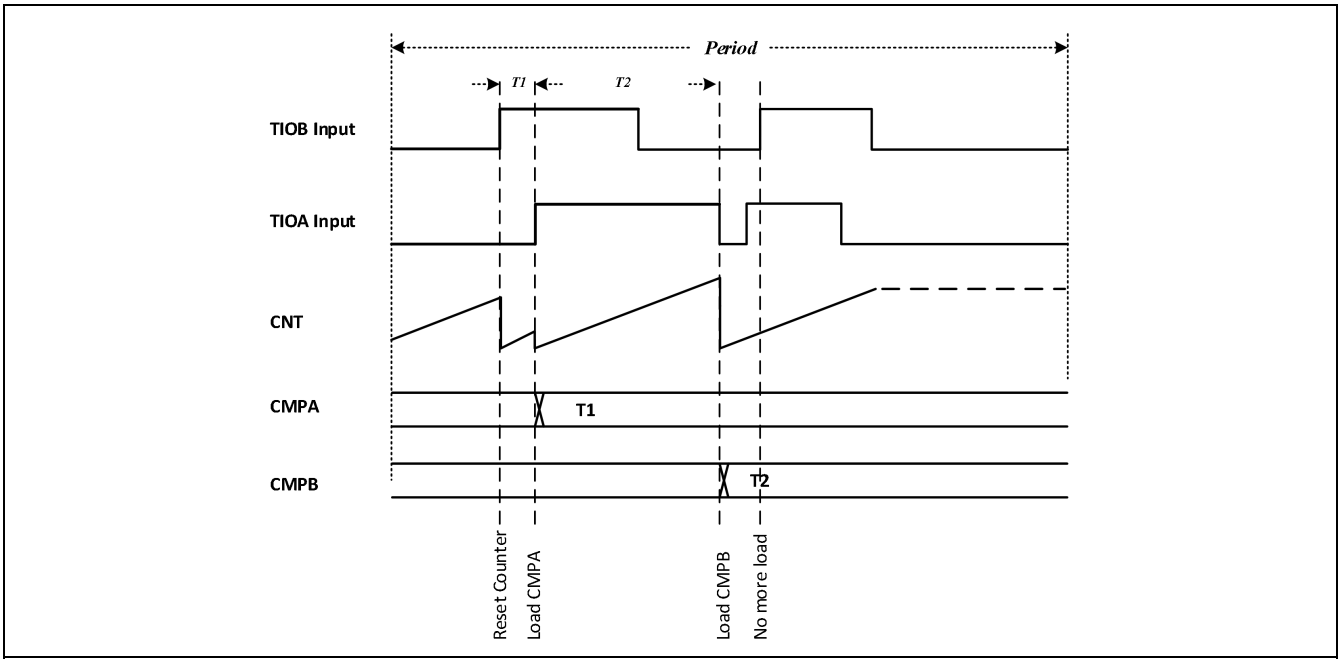


Figure 13-40 测量TIOA和TIOB相位差以及TIOA高电平脉宽

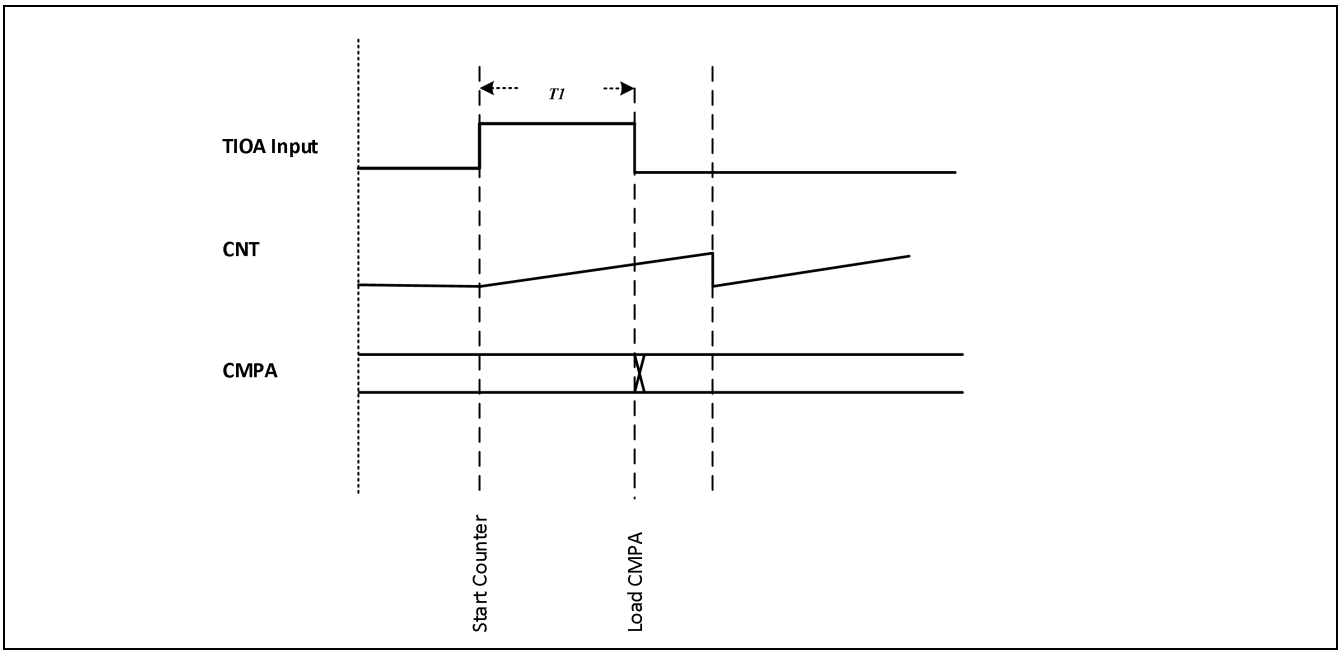


Figure 13-41 测量TIOA的脉冲宽度

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。

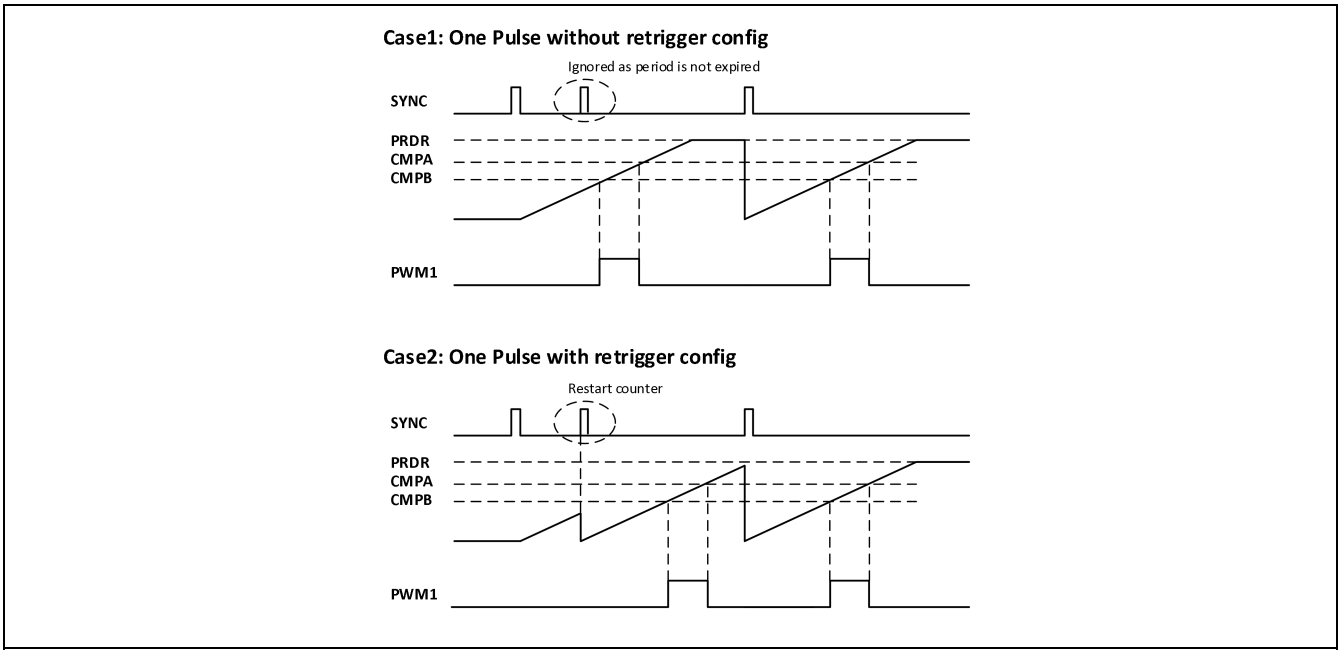


Figure 13-40 单次触发

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

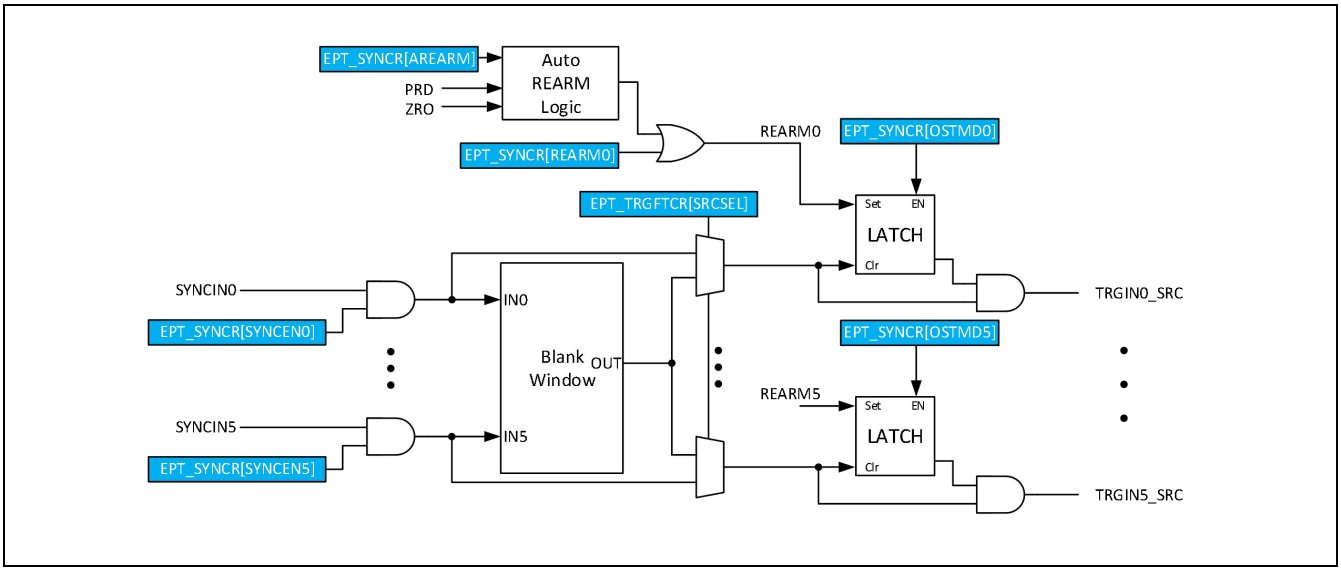


Figure 13-41 同步触发输入

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。

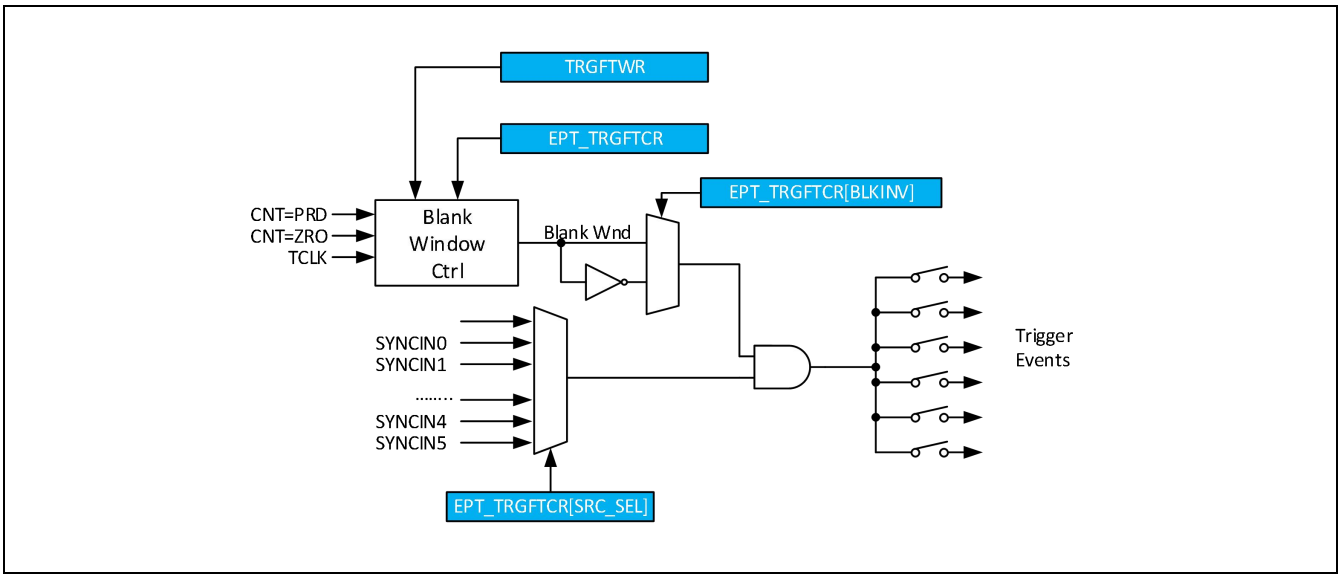


Figure 13-42 滤波器框图

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。

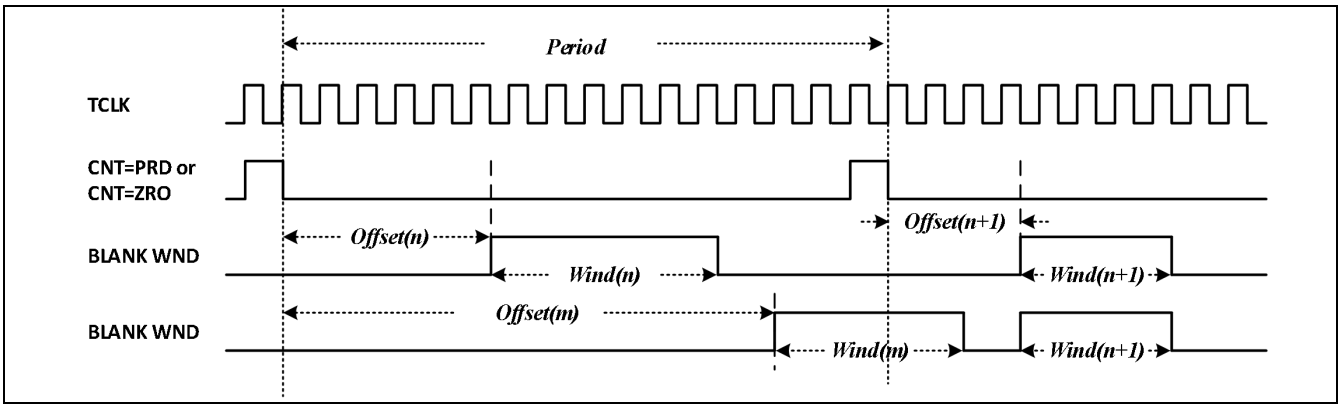


Figure 13-43 滤波器时序

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，每个事件输出对应一个EPT中断信号。事件触发信号通过EVTRG[TRGxSEL]选择触发源，EVTRG[TRGxOE]控制位用于使能触发信号输出到其他外设。

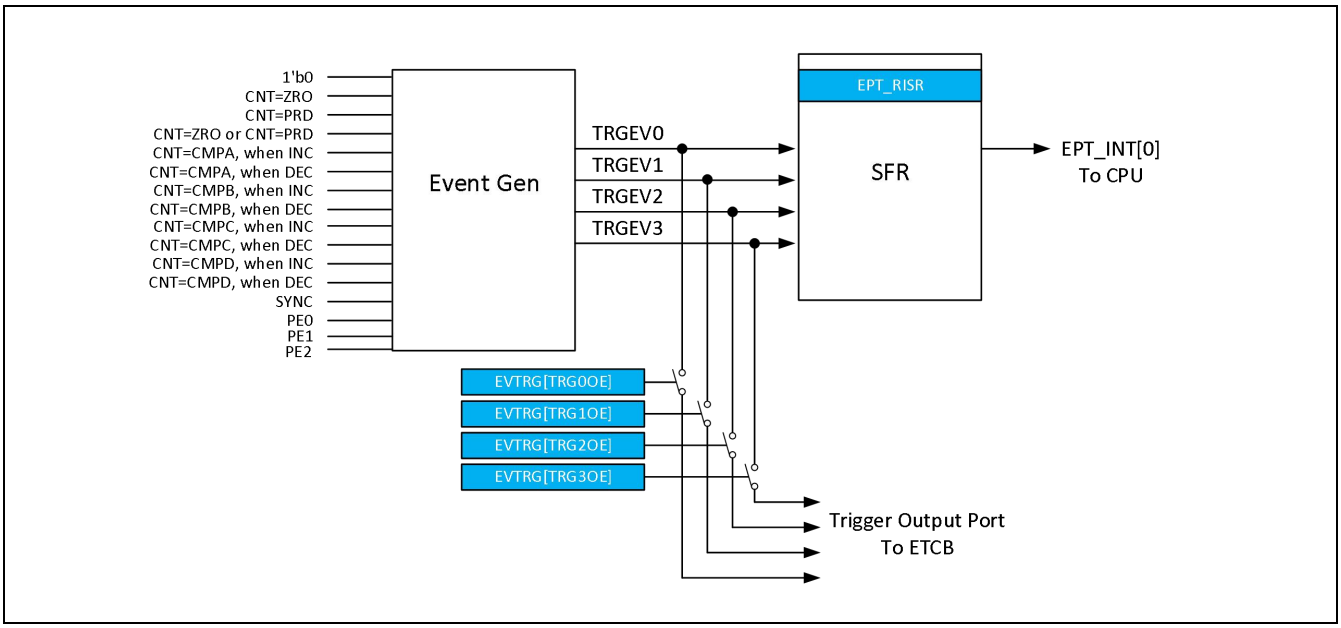


Figure 13-44 同步触发输出

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。

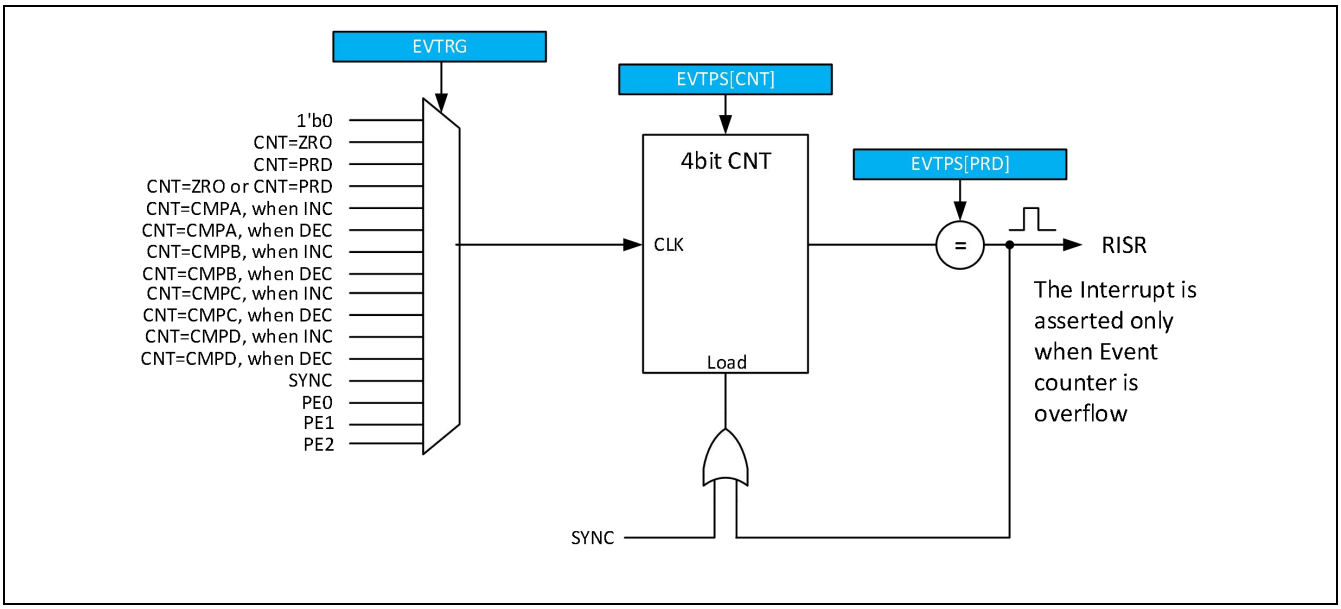


Figure 13-45 事件计数器

13.4 寄存器说明

13.4.1 寄存器表

Base Address of EPT: 0x40069000

Register	Offset	Description	Reset Value
CEDR	0x00	ID和时钟控制寄存器	0x7F980000
RSSR	0x04	启停控制寄存器	0x00000000
PSCR	0x08	时钟分频控制寄存器	0x00000000
CR(WAVE=0)	0x0C	控制寄存器，捕捉模式 WAVE=0	0x00000000
CR(WAVE=1)	0x0C	控制寄存器，波形输出模式：WAVE=1	0x00000000
SYNCR	0x10	同步控制寄存器	0x00000000
GLDCR	0x14	全局载入控制寄存器	0x00000000
GLDCFG	0x18	全局载入配置	0x00000000
GLDCR2	0x1C	全局载入控制寄存器2	0x00000001
PRDR	0x24	周期设置寄存器	0x00000000
PHSR	0x28	相位设置寄存器	0x00000000
CMPA	0x2C	比较值A寄存器	0x00000000
CMPB	0x30	比较值B寄存器	0x00000000
CMPC	0x34	比较值C寄存器	0x00000000
CMPD	0x38	比较值D寄存器	0x00000000
CMPLDR	0x3C	比较值载入控制寄存器	0x00002490
CNT	0x40	时基计数器寄存器	0x00000000
AQLDR	0x44	波形输出载入控制寄存器	0x00002424
AQCR1	0x48	PWM1波形输出控制寄存器	0x00000000
AQCR2	0x4C	PWM2波形输出控制寄存器	0x00000000
AQCR3	0x50	PWM3波形输出控制寄存器	0x00000000
AQCR4	0x54	PWM4波形输出控制寄存器	0x00000000
AQTSCR	0x58	T事件触发源选择寄存器	0x00000000
AQOSF	0x5C	一次性软件波形控制寄存器	0x00010000
AQCSF	0x60	持续性软件波形控制寄存器	0x00000000
DBLDR	0x64	死区配置载入控制寄存器	0x00000492
DBCR	0x68	死区配置控制寄存器	0x00000000
DPSCR	0x6C	死区延迟时钟分频控制寄存器	0x00000000
DBDTR	0x70	死区控制上升沿延时寄存器	0x00000000

DBDTF	0x74	死区控制下降沿延时寄存器	0x00000000
CPCR	0x78	斩波输出控制寄存器	0x00000000
EMSRC	0x7C	紧急状态输入控制寄存器	0x00000000
EMSRC2	0x80	紧急状态输入控制寄存器2	0x00000000
EMPOL	0x84	紧急状态输入极性控制寄存器	0x00000000
EMECR	0x88	紧急状态使能控制寄存器	0x00400000
EMOSR	0x8C	紧急状态输出控制寄存器	0x00000000
EMSLSR	0x94	紧急软锁止状态寄存器	0x00000000
EMSLCLR	0x98	紧急软锁止清除寄存器	0x00000000
EMHLSR	0x9C	紧急硬锁止状态寄存器	0x00000000
EMHLCLR	0xA0	紧急硬锁止清除寄存器	0x00000000
EMFRCR	0xA4	紧急状态软件触发寄存器	0x00000000
EMRISR	0xA8	紧急中断原始状态寄存器	0x00000000
EMMISR	0xAC	紧急中断标志寄存器	0x00000000
EMIMCR	0xB0	紧急中断使能控制寄存器	0x00000000
EMICR	0xB4	紧急中断清除寄存器	0x00000000
TRGFTCR	0xB8	数字比较器滤波窗控制寄存器	0x00000000
TRGFTWR	0xBC	数字比较器滤波窗时序寄存器	0x00000000
EVTRG	0xC0	事件触发选择寄存器	0x00000000
EVPS	0xC4	事件触发计数寄存器	0x00000000
EVCNTINIT	0xC8	事件触发计数器初始化值寄存器	0x00000000
EVSWF	0xCC	事件计数器软件触发控制寄存器	0x00000000
RISR	0xD0	原始中断状态寄存器	0x00000000
MISR	0xD4	中断状态寄存器	0x00000000
IMCR	0xD8	中断使能控制寄存器	0x00000000
ICR	0xDC	中断清除寄存器	0x00000000
REGPROT	0xE8	寄存器写保护控制器	0x00000000

13.4.2 CEDR(ID和时钟控制寄存器)

Address = Base Address+ 0x00, Reset Value = 0x7F980000

31	30	29	28	27	26	25	24	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
0	1	1	1	1	1	1	1	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
IDCODE	[31:16]	R	当前EPT模块的版本信息。
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)
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
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
CSS	[3]	RW	计数器时钟源选择位。 0h：PCLK 1h：由SYNCIN3控制 其他：保留
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
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。

13.4.3 RSSR(启停控制寄存器)

Address = Base Address+ 0x04, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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。
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
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）。
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+ 0x08, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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：FTCLK = FPCLK / (PSC+1)

13.4.5 CR(WAVE=0)(控制寄存器，捕捉模式 WAVE=0)

Address = Base Address+ 0x0C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
LDDRST	[26]	RW	CMPD捕捉载入后，计数器值计数状态控制位。 0h：CMPD触发后，计数器值进行重置 1h：CMPD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。 0h：CMPC触发后，计数器值进行重置 1h：CMPC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。 0h：CMPB触发后，计数器值进行重置 1h：CMPB触发后，计数器值不进行重置
LDARST	[23]	RW	CMPA捕捉载入后，计数器值计数状态控制位。 0h：CMPA触发后，计数器值进行重置 1h：CMPA触发后，计数器值不进行重置
STOP_WRAP	[22:21]	RW	Capture模式下，捕获事件计数器周期设置值。
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
REARM	[19]	RW	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。
WAVE	[18]	RW	EPT工作模式选择。 0h：捕捉模式 1h：波形发生模式

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
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
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
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
BURST	[9]	RW	群脉冲模式。 0h：禁止群脉冲模式 1h：使能群脉冲模式
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
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周期结束(PEND)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]
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）
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。
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
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(WAVE=1)(控制寄存器，波形输出模式：WAVE=1)

Address = Base Address+ 0x0C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	RW	RW	RW	RW	RW	RW	RW	R	R	R	RW	RW	R	RW	RW	RW

Name	Bit	Type	Description
WAVE	[18]	RW	EPT工作模式选择。 0h：捕捉模式 1h：波形发生模式
PSCLD	[17:16]	R	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
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
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
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
BURST	[9]	RW	群脉冲模式。 0h：禁止群脉冲模式 1h：使能群脉冲模式
PHSEN	[7]	R	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
OPM	[6]	R	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
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
IDLEST	[3]	R	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）
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。
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

13.4.7 SYNCR(同步控制寄存器)

Address = Base Address+ 0x10, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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		REARMx						RSVD		OSTMDx						RSVD		SYNCENx					
0	0	0	0	0	0	0	0	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
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
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
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
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
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
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。
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）
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+ 0x14, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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。
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
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。
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]

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
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+ 0x18, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	AQCRD	AQCRC	AQCRB	AQCRA	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
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
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
AQCR4	[11]	RW	AQCR4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	AQCR3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
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
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
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
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
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
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
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
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
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

13.4.10 GLDCR2(全局载入控制寄存器2)

Address = Base Address+ 0x1C, Reset Value = 0x00000001

31	30	29	28	27	26	25	24	23	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	RW	RW

Name	Bit	Type	Description
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
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
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+ 0x24, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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+ 0x28, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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
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
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。
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+ 0x2C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
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
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。
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[LDCMPAMD]进行设置。在Shadow模式下，可以通过CMPLDR[SHDWLDAMD]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。

13.4.14 CMPB(比较值 B 寄存器)

Address = Base Address+ 0x30, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
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
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。
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[LDCMPBMD]进行设置。在Shadow模式下，可以通过CMPLDR[SHDWLDBMD]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。

13.4.15 CMPC(比较值 C 寄存器)

Address = Base Address+ 0x34, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
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
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。
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[LDCMPCMD]进行设置。在Shadow模式下，可以通过CMPLDR[SHDWLDCMD]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。

13.4.16 CMPD(比较值 D 寄存器)

Address = Base Address+ 0x38, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
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
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。
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[LDCMPDMD]进行设置。在Shadow模式下，可以通过CMPLDR[SHDWLDDMD]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。

13.4.17 CMPLDR(比较值载入控制寄存器)

Address = Base Address+ 0x3C, Reset Value = 0x00002490

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								SHDWDFULL	SHDWCFULL	SHDWBFULL	SHDWAFULL	RSVD				SHDWLDDMD			SHDWLDCMD			SHDWLDBMD			SHDWLDAMD			LDCMPDMD	LDCMPCMD	LDCMPBMD	LDCMPAMD
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	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
SHDWDFULL	[23]	R	CMPD的Shadow寄存器非空标志位。 当对CMPD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
SHDWCFULL	[22]	R	CMPC的Shadow寄存器非空标志位。 当对CMPC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
SHDWBFULL	[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
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
SHDWLDDMD	[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

SHDWLDCMD	[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
SHDWLDBMD	[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
SHDWLDAMD	[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。
LDCMPDMD	[3]	RW	CMPD的Shadow功能使能控制。 0h：Shadow模式 1h：Immediate模式 [1]
LDCMPCMD	[2]	RW	CMPC的Shadow功能使能控制。 0h：Shadow模式 1h：Immediate模式 [1]
LDCMPBMD	[1]	RW	CMPB的Shadow功能使能控制。 0h：Shadow模式 1h：Immediate模式 [1]
LDCMPAMD	[0]	RW	CMPA的Shadow功能使能控制。 0h：Shadow模式 1h：Immediate模式 [1]
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+ 0x40, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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+ 0x44, Reset Value = 0x00002424

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																SHDWLD4MD			SHDWLD3MD			LDAQ4MD	LDAQ3MD	SHDWLD2MD			SHDWLD1MD			LDAQ2MD	LDAQ1MD
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
SHDWLD4MD	[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。
SHDWLD3MD	[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。
LDAQ4MD	[9]	RW	AQCR4寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
LDAQ3MD	[8]	RW	AQCR3寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
SHDWLD2MD	[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。
SHDWLD1MD	[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。
LDAQ2MD	[1]	RW	AQCR2寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
LDAQ1MD	[0]	RW	AQCR1寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式

13.4.20 AQCR1(PWM1 波形输出控制寄存器)

Address = Base Address+ 0x48, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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
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
T2D	[19:18]	RW	当T2事件发生，且此时计数方向为递减时，在通道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，在通道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，在通道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，在通道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，在通道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，在通道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，在通道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）
C1U	[5:4]	RW	当CNT值等于C1，且此时计数方向为递增时，在通道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）
PRD	[3:2]	RW	当CNT值等于PRDR时，在通道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）
ZRO	[1:0]	RW	当CNT值等于零时，在通道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）

13.4.21 AQCR2(PWM2 波形输出控制寄存器)

Address = Base Address+ 0x4C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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
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
T2D	[19:18]	RW	当T2事件发生，且此时计数方向为递减时，在通道pwm2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，在通道pwm2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，在通道pwm2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，在通道pwm2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，在通道pwm2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，在通道pwm2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，在通道pwm2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）
C1U	[5:4]	RW	当CNT值等于C1，且此时计数方向为递增时，在通道pwm2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）
PRD	[3:2]	RW	当CNT值等于PRDR时，在通道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：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）
ZRO	[1:0]	RW	当CNT值等于零时，在通道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：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）

13.4.22 AQCR3(PWM3 波形输出控制寄存器)

Address = Base Address+ 0x50, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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
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
T2D	[19:18]	RW	当T2事件发生，且此时计数方向为递减时，在通道pwm3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，在通道pwm3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，在通道pwm3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，在通道pwm3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，在通道pwm3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，在通道pwm3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，在通道pwm3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）
C1U	[5:4]	RW	当CNT值等于C1，且此时计数方向为递增时，在通道pwm3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）
PRD	[3:2]	RW	当CNT值等于PRDR时，在通道pwm3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）
ZRO	[1:0]	RW	当CNT值等于零时，在通道pwm3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）

13.4.23 AQCR4(PWM4 波形输出控制寄存器)

Address = Base Address+ 0x54, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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
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
T2D	[19:18]	RW	当T2事件发生，且此时计数方向为递减时，在通道pwm4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，在通道pwm4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，在通道pwm4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，在通道pwm4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，在通道pwm4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，在通道pwm4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，在通道pwm4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）
C1U	[5:4]	RW	当CNT值等于C1，且此时计数方向为递增时，在通道pwm4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）
PRD	[3:2]	RW	当CNT值等于PRDR时，在通道pwm4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）
ZRO	[1:0]	RW	当CNT值等于零时，在通道pwm4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）

13.4.24 AQTSCR(T 事件触发源选择寄存器)

Address = Base Address+ 0x58, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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
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

13.4.25 AQOSF(一次性软件波形控制寄存器)

Address = Base Address+ 0x5C, Reset Value = 0x00010000

31	30	29	28	27	26	25	24	23	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
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
ACT4	[14:13]	RW	当软件强制输出时，通道4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]	W	在通道4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，直到有其他改变通道A输出状态的触发事件发生。
ACT3	[10:9]	RW	当软件强制输出时，通道3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]	W	在通道3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，直到有其他改变通道A输出状态的触发事件发生。
ACT2	[6:5]	RW	当软件强制输出时，通道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）
OSTSF2	[4]	W	在通道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，直到有其他改变通道A输出状态的触发事件发生。
ACT1	[2:1]	RW	当软件强制输出时，通道1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）
OSTSF1	[0]	W	在通道1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，直到有其他改变通道A输出状态的触发事件发生。

13.4.26 AQCSF(持续性软件波形控制寄存器)

Address = Base Address+ 0x60, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
CSF4	[7:6]	RW	通过软件对通道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
CSF3	[5:4]	RW	通过软件对通道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
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
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

13.4.27 DBLDR(死区配置载入控制寄存器)

Address = Base Address+ 0x64, Reset Value = 0x00000492

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																				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
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
SHDWPSC	[9]	RW	DCKPSC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
SHDWDTF	[6]	RW	DBDTF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
SHDWDTR	[3]	RW	DBDTR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
CRSHDWEN	[0]	RW	DBCR寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式

13.4.28 DBCR(死区配置控制寄存器)

Address = Base Address+ 0x68, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				CH3_DEDB	CH2_DEDB	CH1_DEDB	DCKSEL	CH3_OUTSWAP		CH3_INSEL		CH3_POLARITY		CH3_OUTSEL		CH2_OUTSWAP		CH2_INSEL		CH2_POLARITY		CH2_OUTSEL		CH1_OUTSWAP		CH1_INSEL		CH1_POLARITY		CH1_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
CH3_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
CH2_DEDB	[26]	RW	在PWM2 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
CH1_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
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
CH3_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
CH3_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
CH3_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：Y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
CH3_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
CH2_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
CH2_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。
CH2_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：Y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
CH2_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
CH1_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
CH1_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
CH1_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：Y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
CH1_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

13.4.29 DPSCR(死区延迟时钟分频控制寄存器)

Address = Base Address+ 0x6C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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：FDBCLK = FHCLK / (DPSC+1)

13.4.30 DBDTR(死区控制上升沿延时寄存器)

Address = Base Address+ 0x70, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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 TRED = DTR x TDBCLK

13.4.31 DBDTF(死区控制下降沿延时寄存器)

Address = Base Address+ 0x74, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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 TRED = DTF x TDBCLK

13.4.32 CPCR(斩波输出控制寄存器)

Address = Base Address+ 0x78, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD										CH3Y_CPEN	CH3X_CPEN	CH2Y_CPEN	CH2X_CPEN	CH1Y_CPEN	CH1X_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	R	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
CH3Y_CPEN	[21]	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
CH3X_CPEN	[20]	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
CH2Y_CPEN	[19]	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
CH2X_CPEN	[18]	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
CH1Y_CPEN	[17]	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
CH1X_CPEN	[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
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
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
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。 Fchop = PCLK / ((CDIV+1) x 8)
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 = Tchop x OSPWTH （Tchop为一个载波的周期时间）

13.4.33 EMSRC(紧急状态输入控制寄存器)

Address = Base Address+ 0x7C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
EP7_SEL	[31:28]	RW	
EP6_SEL	[27:24]	RW	
EP5_SEL	[23:20]	RW	
EP4_SEL	[19:16]	RW	
EP3_SEL	[15:12]	RW	
EP2_SEL	[11:8]	RW	
EP1_SEL	[7:4]	RW	
EP0_SEL	[3:0]	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CMP0作为当前EP的输入

6h：选择CMP1作为当前EP的输入

7h：选择CMP2作为当前EP的输入

8h：选择CMP3作为当前EP的输入

9h：选择CMP4作为当前EP的输入

Ah：选择CMP5作为当前EP的输入

Bh：选择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 2)

Address = Base Address+ 0x80, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
ORL1_EP7	[23]	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
ORL1_EP6	[22]	RW	
ORL1_EP5	[21]	RW	
ORL1_EP4	[20]	RW	
ORL1_EP3	[19]	RW	
ORL1_EP2	[18]	RW	
ORL1_EP1	[17]	RW	
ORL1_EP0	[16]	RW	
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
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
ORL0_EP7	[7]	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
ORL0_EP6	[6]	RW	
ORL0_EP5	[5]	RW	
ORL0_EP4	[4]	RW	

ORL0_EP3	[3]	RW	
ORL0_EP2	[2]	RW	
ORL0_EP1	[1]	RW	
ORL0_EP0	[0]	RW	
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+ 0x84, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																						CMP5_POL	CMP4_POL	CMP3_POL	CMP2_POL	CMP1_POL	CMP0_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	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CMP5_POL	[9]	RW	
CMP4_POL	[8]	RW	
CMP3_POL	[7]	RW	
CMP2_POL	[6]	RW	
CMP1_POL	[5]	RW	
CMP0_POL	[4]	RW	
EBI3_POL	[3]	RW	
EBI2_POL	[2]	RW	
EBI1_POL	[1]	RW	
EBI0_POL	[0]	RW	

EBIx/CMPx_POL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/CMPx_POL作为异常处理输入时，以电平方式工作。当EBIx/CMPx_POL作为事件触发源时，设置为高电平有效时，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+ 0x88, Reset Value = 0x00400000

31	30	29	28	27	26	25	24	23	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_LCKMD		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
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
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
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
EMASYNC	[26]	RW	EP端口同步设置控制位。 0h：使能同步 1h：禁止同步
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
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
OSRSHDW	[21]	RW	EMOSR寄存器的Shadow功能使能控制。 0h：Immediate模式

			1h：Shadow模式
EP7_LCKMD	[15:14]	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
EP6_LCKMD	[13:12]	RW	
EP5_LCKMD	[11:10]	RW	
EP4_LCKMD	[9:8]	RW	
EP3_LCKMD	[7:6]	RW	
EP2_LCKMD	[5:4]	RW	
EP1_LCKMD	[3:2]	RW	
EP0_LCKMD	[1:0]	RW	
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)

Address = Base Address+ 0x8C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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_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_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
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_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_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_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_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_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
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+ 0x94, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
EP7	[7]	R	
EP6	[6]	R	
EP5	[5]	R	
EP4	[4]	R	
EP3	[3]	R	
EP2	[2]	R	
EP1	[1]	R	
EP0	[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+ 0x98, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
EP7	[7]	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
EP6	[6]	W	
EP5	[5]	W	
EP4	[4]	W	
EP3	[3]	W	
EP2	[2]	W	
EP1	[1]	W	
EP0	[0]	W	

13.4.40 EMHLSR(紧急硬锁止状态寄存器)

Address = Base Address+ 0x9C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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
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
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
EP7	[7]	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
EP6	[6]	R	
EP5	[5]	R	
EP4	[4]	R	
EP3	[3]	R	
EP2	[2]	R	
EP1	[1]	R	
EP0	[0]	R	

13.4.41 EMHLCLR(紧急硬锁止清除寄存器)

Address = Base Address+ 0xA0, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
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
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
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
EP7	[7]	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
EP6	[6]	W	
EP5	[5]	W	
EP4	[4]	W	
EP3	[3]	W	
EP2	[2]	W	
EP1	[1]	W	
EP0	[0]	W	

13.4.42 EMFRCR(紧急状态软件触发寄存器)

Address = Base Address+ 0xA4, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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_EP7	[7]	W	
FRC_EP6	[6]	W	
FRC_EP5	[5]	W	
FRC_EP4	[4]	W	
FRC_EP3	[3]	W	
FRC_EP2	[2]	W	
FRC_EP1	[1]	W	
FRC_EP0	[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+ 0xA8, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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
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
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
EP7	[7]	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
EP6	[6]	R	
EP5	[5]	R	
EP4	[4]	R	
EP3	[3]	R	
EP2	[2]	R	
EP1	[1]	R	
EP0	[0]	R	

13.4.44 EMMISR(紧急中断标志寄存器)

Address = Base Address+ 0xAC, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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
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
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
EP7	[7]	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
EP6	[6]	R	
EP5	[5]	R	
EP4	[4]	R	
EP3	[3]	R	
EP2	[2]	R	
EP1	[1]	R	
EP0	[0]	R	

13.4.45 EMIMCR(紧急中断使能控制寄存器)

Address = Base Address+ 0xB0, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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
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
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
EP7	[7]	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
EP6	[6]	RW	
EP5	[5]	RW	
EP4	[4]	RW	
EP3	[3]	RW	
EP2	[2]	RW	
EP1	[1]	RW	
EP0	[0]	RW	

13.4.46 EMICR(紧急中断清除寄存器)

Address = Base Address+ 0xB4, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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
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
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
EP7	[7]	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
EP6	[6]	RW	
EP5	[5]	RW	
EP4	[4]	RW	
EP3	[3]	RW	
EP2	[2]	RW	
EP1	[1]	RW	
EP0	[0]	RW	

13.4.47 TRGFTCR(数字比较器滤波窗控制寄存器)

Address = Base Address+ 0xB8, Reset Value = 0x00000000

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

13.4.48 TRGFTWR(数字比较器滤波窗时序寄存器)

Address = Base Address+ 0xBC, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
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。
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。

13.4.49 EVTRG(事件触发选择寄存器)

Address = Base Address+ 0xC0, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
CNT3INITFRC	[27]	RW	TRGEV3CNT软件触发更新 0h：无效 1h：EVCNT3INIT内容更新到EVCNT3中
CNT2INITFRC	[26]	RW	TRGEV2CNT软件触发更新 0h：无效 1h：EVCNT2INIT内容更新到EVCNT2中
CNT1INITFRC	[25]	RW	TRGEV1CNT软件触发更新 0h：无效 1h：EVCNT1INIT内容更新到EVCNT1中
CNT0INITFRC	[24]	RW	TRGEV0CNT软件触发更新 0h：无效 1h：EVCNT0INIT内容更新到EVCNT0中
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
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
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
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
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。

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。
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。
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。
TRG3SEL	[15:12]	RW	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
TRG2SEL	[11:8]	RW	TRGEV2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
TRG1SEL	[7:4]	RW	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
TRG0SEL	[3:0]	RW	TRGEV0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

13.4.50 EVPS(事件触发计数寄存器)

Address = Base Address+ 0xC4, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
TRGEV3CNT	[31:28]	R	TRGEV3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。
TRGEV1CNT	[23:20]	R	TRGEV1事件计数器值。 读取时，返回当前事件计数器值。
TRGEV0CNT	[19:16]	R	TRGEV0事件计数器值。 读取时，返回当前事件计数器值。
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
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
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
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

13.4.51 EVCNTINIT(事件触发计数器初始化值寄存器)

Address = Base Address+ 0xC8, Reset Value = 0x00000000

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

Name	Bit	Type	Description
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。
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。
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。
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。

13.4.52 EVSWF(事件计数器软件触发控制寄存器)

Address = Base Address+ 0xCC, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
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
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
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
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

13.4.53 RISR(原始中断状态寄存器)

Address = Base Address+ 0xD0, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
PEND	[16]	R	周期结束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
CDU	[14]	R	递增阶段CNT = CMPD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
CCU	[12]	R	递增阶段CNT = CMPC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
CBU	[10]	R	递增阶段CNT = CMPB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
CAU	[8]	R	递增阶段CNT = CMPA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
CAP_LD2	[6]	R	Capture Load to CMPC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_LD0	[4]	R	Capture Load to CMPA中断请求原始标志状态
TRGEV3	[3]	R	TRGEV3中断请求原始标志状态
TRGEV2	[2]	R	TRGEV2中断请求原始标志状态
TRGEV1	[1]	R	TRGEV1中断请求原始标志状态
TRGEV0	[0]	R	TRGEV0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+ 0xD4, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
PEND	[16]	R	周期结束中断请求标志状态
CDD	[15]	R	递减阶段CNT = CMPD中断请求标志状态
CDU	[14]	R	递增阶段CNT = CMPD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
CCU	[12]	R	递增阶段CNT = CMPC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
CBU	[10]	R	递增阶段CNT = CMPB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
CAU	[8]	R	递增阶段CNT = CMPA中断请求标志状态
CAP_LD3	[7]	R	Capture Load to CMPD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_LD1	[5]	R	Capture Load to CMPB中断请求标志状态
CAP_LD0	[4]	R	Capture Load to CMPA中断请求标志状态
TRGEV3	[3]	R	TRGEV3中断请求标志状态
TRGEV2	[2]	R	TRGEV2中断请求标志状态
TRGEV1	[1]	R	TRGEV1中断请求标志状态
TRGEV0	[0]	R	TRGEV0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+ 0xD8, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
PEND	[16]	RW	周期结束中断中断使能控制位。
CDD	[15]	RW	递减阶段CNT = CMPD中断使能控制位。
CDU	[14]	RW	递增阶段CNT = CMPD中断使能控制位。
CCD	[13]	RW	递减阶段CNT = CMPC中断使能控制位。
CCU	[12]	RW	递增阶段CNT = CMPC中断使能控制位。
CBD	[11]	RW	递减阶段CNT = CMPB中断使能控制位。
CBU	[10]	RW	递增阶段CNT = CMPB中断使能控制位。
CAD	[9]	RW	递减阶段CNT = CMPA中断使能控制位。
CAU	[8]	RW	递增阶段CNT = CMPA中断使能控制位。
CAP_LD3	[7]	RW	Capture Load to CMPD中断使能控制位。
CAP_LD2	[6]	RW	Capture Load to CMPC中断使能控制位。
CAP_LD1	[5]	RW	Capture Load to CMPB中断使能控制位。
CAP_LD0	[4]	RW	Capture Load to CMPA中断使能控制位。
TRGEV3	[3]	RW	TRGEV3中断使能控制位。
TRGEV2	[2]	RW	TRGEV2中断使能控制位。
TRGEV1	[1]	RW	TRGEV1中断使能控制位。
TRGEV0	[0]	RW	TRGEV0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+ 0xDC, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
PEND	[16]	W	周期结束中断中断清除。
CDD	[15]	W	递减阶段CNT = CMPD中断清除。
CDU	[14]	W	递增阶段CNT = CMPD中断清除。
CCD	[13]	W	递减阶段CNT = CMPC中断清除。
CCU	[12]	W	递增阶段CNT = CMPC中断清除。
CBD	[11]	W	递减阶段CNT = CMPB中断清除。
CBU	[10]	W	递增阶段CNT = CMPB中断清除。
CAD	[9]	W	递减阶段CNT = CMPA中断清除。
CAU	[8]	W	递增阶段CNT = CMPA中断清除。
CAP_LD3	[7]	W	Capture Load to CMPD中断清除。
CAP_LD2	[6]	W	Capture Load to CMPC中断清除。
CAP_LD1	[5]	W	Capture Load to CMPB中断清除。
CAP_LD0	[4]	W	Capture Load to CMPA中断清除。
TRGEV3	[3]	W	TRGEV3中断清除。
TRGEV2	[2]	W	TRGEV2中断清除。
TRGEV1	[1]	W	TRGEV1中断清除。
TRGEV0	[0]	W	TRGEV0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+ 0xE8, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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
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。
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。

14 模拟运算放大器（OP-AMP）

14.1 概述

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

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 特性

- 支持最大2个独立运算放大器。
- 每个运算放大器可以支持外部独立工作，或者内部增益控制模式。
- 可配置的内部增益放大系数
  - X4 / X5 / X6 / X7 / X24 / X30 / X35 / X40。

14.1.2 管脚描述

Table 14-1 OPA 管脚描述

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

14.2 功能描述

14.2.1 运算放大器功能模块

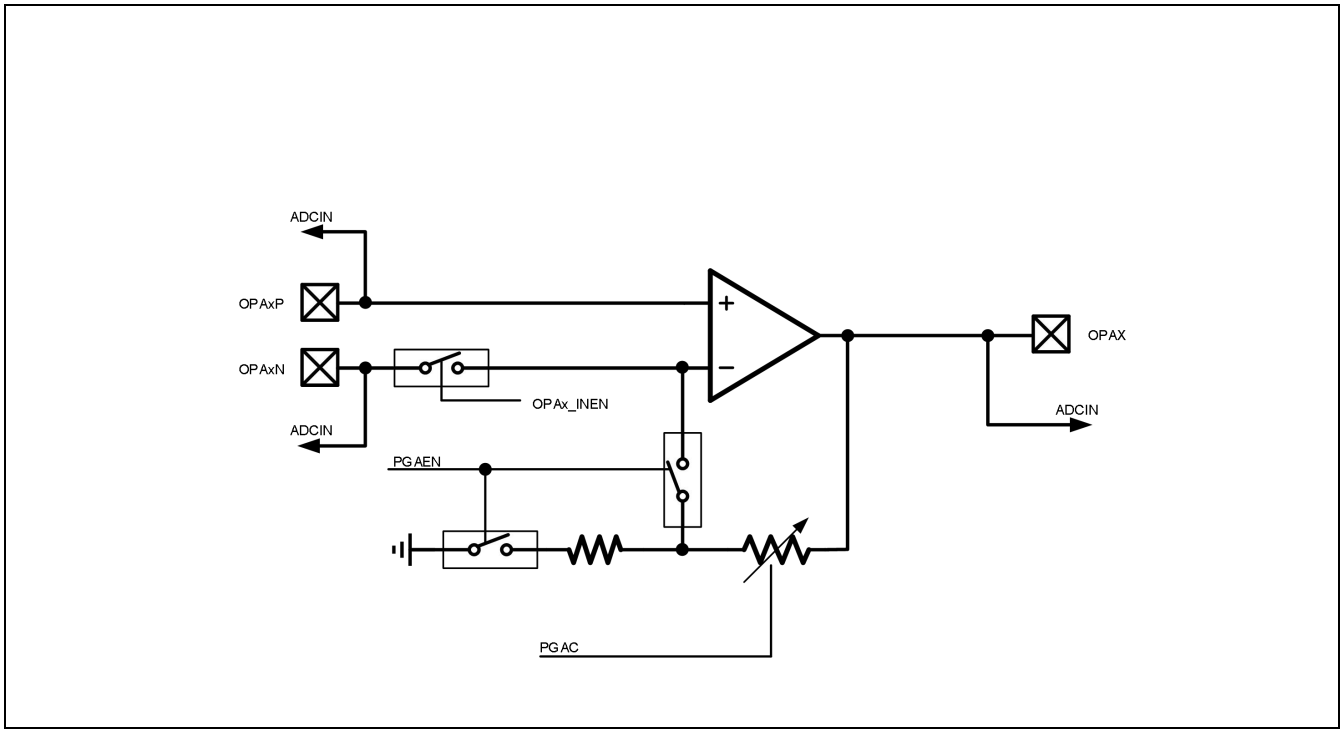


Figure 14-1 运算放大器框图

14.2.2 运算放大器工作模式

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

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

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

运放也支持外部部增益控制模式工作，当选择外部增益控制时，可以通过外部电阻网络来控制环路增益。

在低功耗模式下，运放被强制关闭。

14.2.3 运算放大器作为ADC输入的典型应用

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

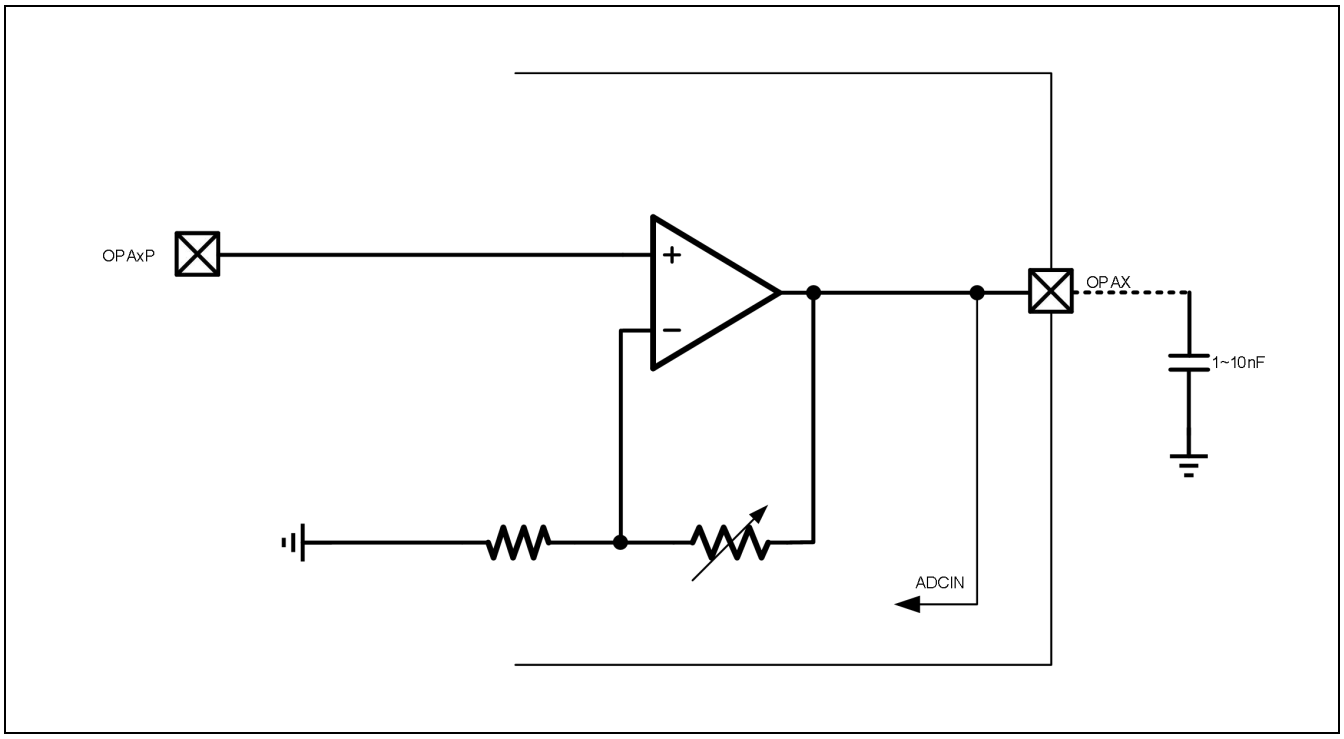


Figure 14-2 运算放大器作为 ADC 输入

需要进行的配置为：

1. 相应的GPIO设置为AF7模拟信号输入输出功能(运放输出以及ADC输入)。
2. 相应的运算放大器使能内部增益 (PGAEN=1)，选择需要的放大倍数。
3. ADC选择对应的ADCIN输入通道 (ADCIN3/5)，并且根据需求选择合适的转换时钟和转换模式。

14.3 寄存器说明

14.3.1 寄存器表

Base Address of OPA0: 0x400C0000

Base Address of OPA1: 0x400C0004

Register	Offset	Description	Reset Value
OPAx_CR	0x00	运算放大器0/1控制寄存器	0x00000000

14.3.2 OPAx_CR(运算放大器 0/1 控制寄存器)

Address = Base Address+ 0x00, Reset Value = 0x00000000

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

Name	Bit	Type	Description
BUFEN	[9]	RW	BUF模式 (只在内部增益模式下有效) 0：非BUF模式 1：内部增益使用BUF模式 (增益x1)
INEN	[8]	RW	INN输入端控制，内部增益模式(PGAEN=1)中，使能INN管脚外部输入 0：不连接外部端口 1：连接到外部输入端口OPAxN 注：外部增益模式下，INN始终连接在OPAxN上
PGAC	[4:2]	RW	内部增益控制设置。 0：X 4 1：X 5 2：X 6 3：X 7 4：X 24 5：X 30 6：X 35 7：X 40
PGAEN	[1]	RW	内部增益控制使能。（1） 0：禁止内部增益控制 1：使能内部增益控制
OPAEN	[0]	RW	OPA模块的使能控制。 0： 禁止OPA模块 1： 使能OPA模块
NOTE: （1） 内部增益控制只支持正向输入的情况			

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

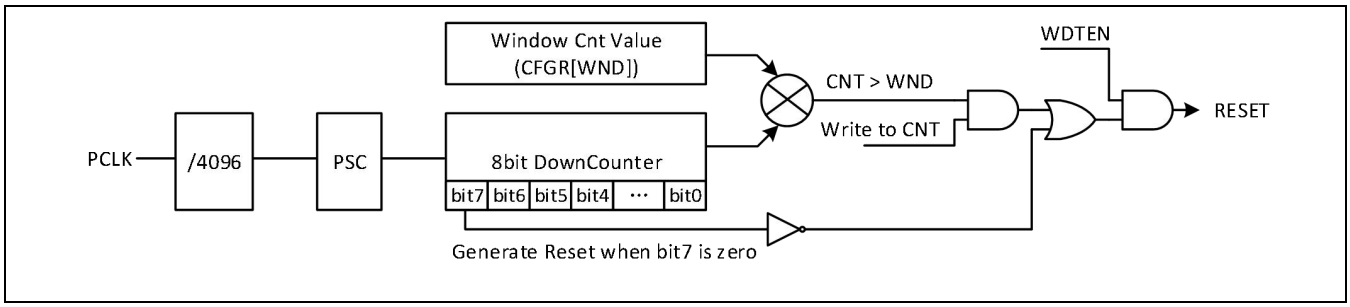


Figure 15-1 模块结构示意图

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 15-1 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。

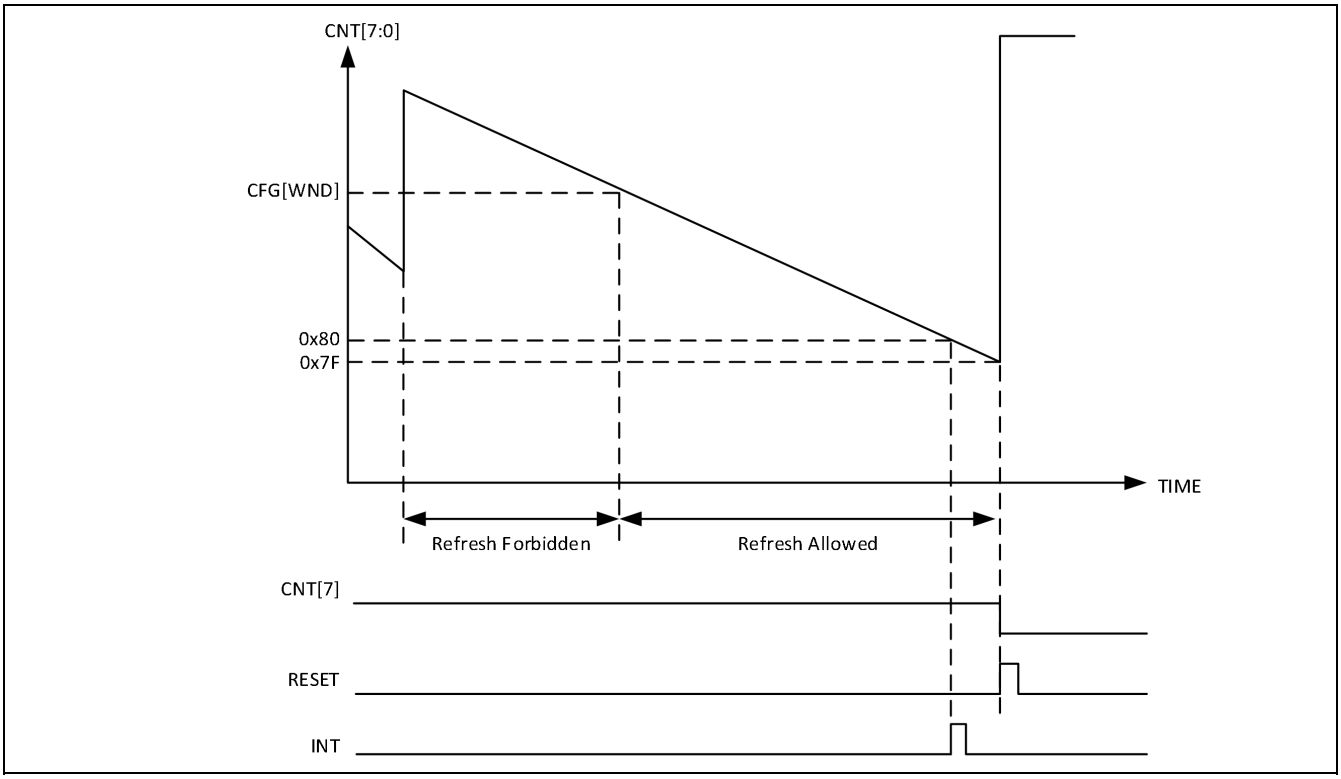


Figure 15- 2 计数器工作示例

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 寄存器说明

15.3.1 寄存器表

Base Address of WWDT: 0x40072000

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

15.3.2 WWDT_CR(Control Register)

Address = Base Address+ 0x0000, Reset Value = 0x000000FF

31	30	29	28	27	26	25	24	23	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
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。
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。

15.3.3 WWDT_CFGR(Configuration Register)

Address = Base Address+ 0x0004, Reset Value = 0x000000FF

31	30	29	28	27	26	25	24	23	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
DBGEN	[10]	RW	调试模式控制位。 0h：禁止调试模式 1h：使能调试模式
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
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。

15.3.4 WWDT_RISR(Raw Interrupt Status Register)

Address = Base Address+ 0x0008, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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]	R	EVI中断请求原始标志状态

15.3.5 WWDT_MISR(Masked Interrupt Status Register)

Address = Base Address+ 0x000C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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]	R	EVI中断请求标志状态

15.3.6 WWDT_IMCR(Interrupt Masking Control Register)

Address = Base Address+ 0x0010, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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			

15.3.7 WWDT_ICR(Interrupt Pending Clear Register)

Address = Base Address+ 0x0014, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	
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
- 固定一个(位)停止位
- 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 16-1 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

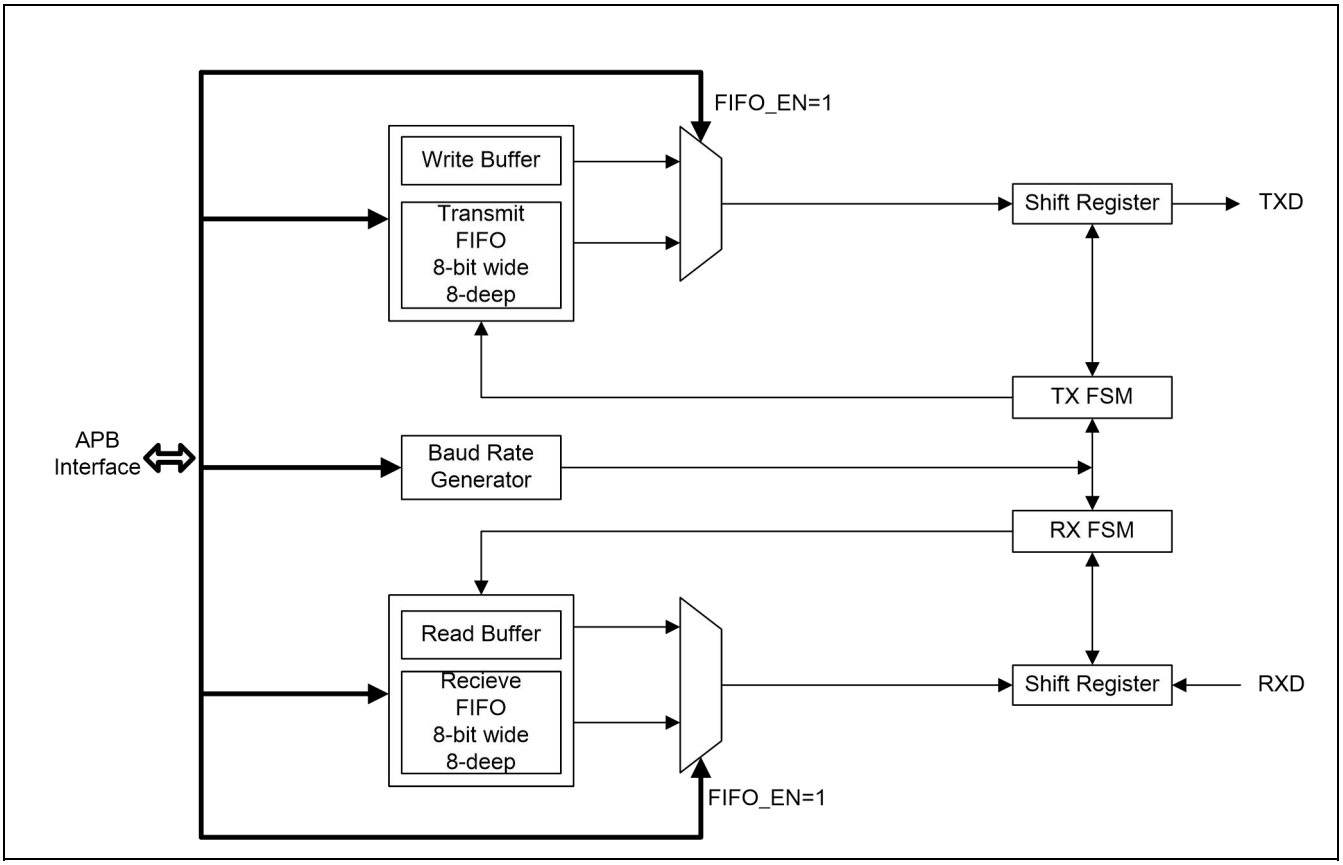


Figure 16-1 UART模块框图

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 16-2 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)。

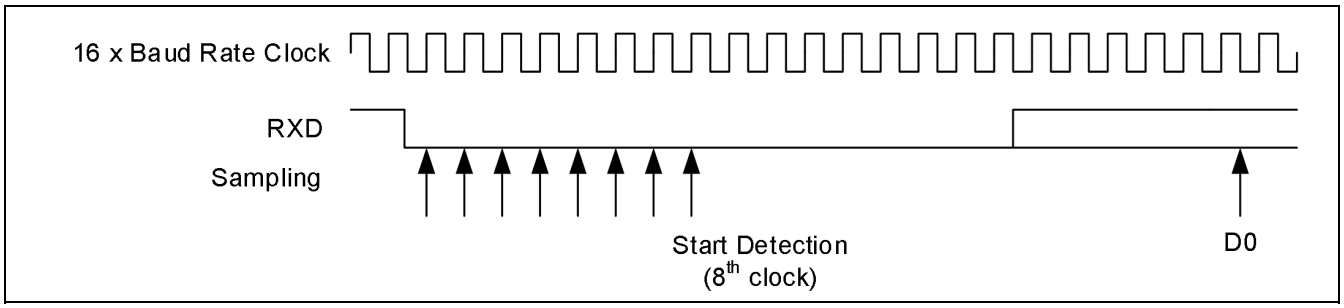


Figure 16-2 起始位检测

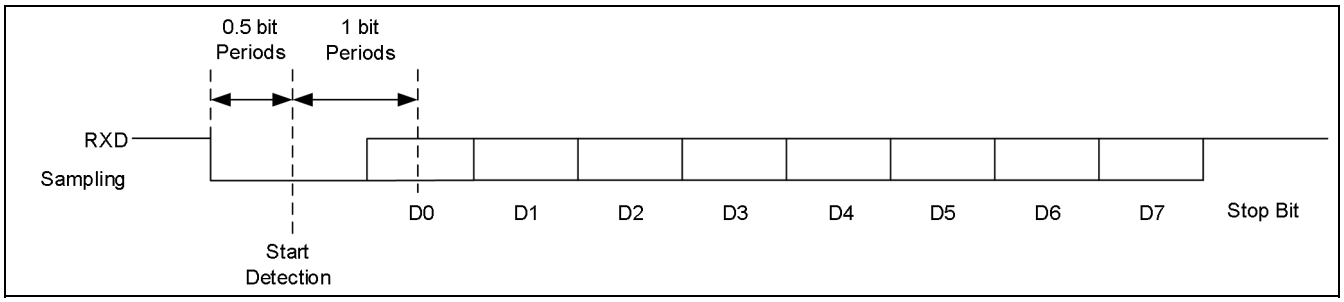


Figure 16-3 接收数据

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。

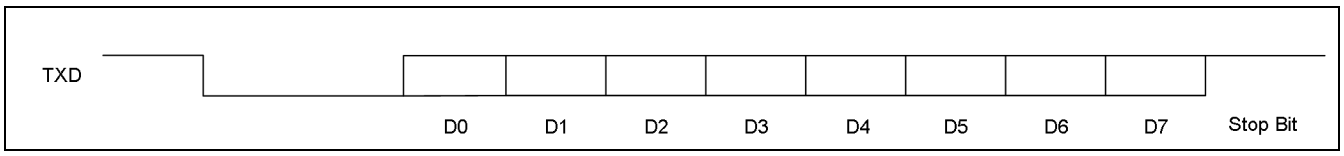


Figure 16-4 数据发送

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 Break

16.2.2.4.1 发送Break

当UART_CTRL寄存器中的STTBRK (Start break)命令被置1时，发送端会在UART TX上发送Break。在UART TX被拉低之前，发送移位寄存器中的字节会被发送完。

如果要移除Break，那么必须将UART_CTRL中的STPBRK (Stop break)命令置1。UART最少发送一个字节长的Break。

之后UART TX会恢复到高电平(空闲状态)并且持续12个位周期，保证Break被正确的检测到，然后发送端继续正常的操作。

16.2.2.4.2 接收Break

当所有的数据，校验和停止位都是0时，接收端认为检测到了Break。在检测到低电平地址位的时刻，接收端将UART_SR中的RXBRK (Break received)位置1。如果在UART_CTRL寄存器中使能了INT_RXBRK中断，将会产生对应中断，UART_ISR中RXBRK位置1.

16.2.2.5 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

16.2.2.6 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 of UART0: 0x40081000

Register	Offset	Description	Reset Value
UART_DATA	0x000	数据寄存器	0x00000000
UART_SR	0x004	状态寄存器	0x00000000
UART_CTRL	0x008	控制寄存器	0x00000000
UART_ISR	0x00C	中断状态寄存器	0x00000000
UART_BRDIV	0x010	波特率分频寄存器	0x00000000
UART_RTOR	0x018	接收超时配置寄存器	0x0000FFFF
UART_TTGR	0x01C	发送端Time-Guard配置寄存器	0x00000000
UART_SRR	0x020	软件复位寄存器	0x00000000

16.3.2 UART_DATA(数据寄存器)

Address = Base Address+ 0x000, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
DATA	[7:0]	RW	发送或接收到的数据 读 = 接收到的数据 写 = 发送的数据

16.3.3 UART_SR(状态寄存器)

Address = Base Address+ 0x004, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																					RXBRK	TIMEOUT	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
RXBRK	[10]	R	接收端Break. 0 = 在上一次状态复位后，还没有检测到Break 1 = 在上一次状态复位后，检测到Break（读） 1 = 清除RXBRK状态位(写)
TIMEOUT	[9]	R	超时 0 = 开始超时接收后，没有检测到超时，或者超时寄存器被设置为0 1 = 开始超时接收后，检测到了超时(读) 1 = 清除超时状态位(写)
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
RNE	[7]	R	接收FIFO是否为空状态位 0 = 接收FIFO为空 1 = 接收FIFO非空
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
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
PARITY_SR	[4]	RW	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)

RX_OVER	[3]	RW	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)
TX_OVER	[2]	RW	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)
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)
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)

16.3.4 UART_CTRL(控制寄存器)

Address = Base Address+ 0x008, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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						STPBRK	STTBRK	INT_RXBRK	INT_RXTO	STTTO	INT_TX_DONE_EN	INT_OVER	RSVD	RXIFLSEL			INT_FIFO_RX	INT_FIFO_TX	FIFO_EN	PARITY			INT_PARITY	RSVD	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	W	W	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	W	W	RW	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
DBGEN	[31]	RW	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
STPBRK	[24]	W	停止Break. 0 = 无效 1 = 如果一个Break状态正在发送，那么写1会在最少一个字节长度的Break状态后停止Break，并且发送一个12位周期的高电平
STTBRK	[23]	W	开始Break. 0 = 无效 1 = 如果Break没有发送，那么写1会在当前移位寄存器中的数据发送完之后，开始发送Break状态
INT_RXBRK	[22]	RW	接收到Break中断使能/禁止 0= 禁止接收到Break中断 1= 使能接收到Break中断
INT_RXTO	[21]	RW	接收到Timeout中断使能/禁止 0= 禁止接收到Timeout中断 1= 使能接收到Timeout中断
STTTO	[20]	RW	开启超时接收 0 = 不开启 1 = 必须在超时计数器计数完成之前，接收到字节数据，否则报错

INT_TX_DONE_EN	[19]	RW	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
INT_OVER	[18]	RW	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
RXIFLSEL	[16:14]	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
INT_FIFO_RX	[13]	RW	[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
INT_FIFO_TX	[12]	RW	[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
FIFO_EN	[11]	RW	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
PARITY	[10:8]	W	校验位类型 0XX：无校验位 100：偶校验 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)
INT_PARITY	[7]	RW	PARITY中断使能/禁止 0 = 禁止PARITY中断 1 = 使能PARITY中断
INT_OVER_RX	[5]	RW	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
INT_OVER_TX	[4]	RW	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

INT_RX	[3]	RW	RX 中断使能/禁止 0 = 禁止RX中断 1 = 使能RX中断
INT_TX	[2]	RW	TX 中断使能/禁止 0 = 禁止TX中断 1 = 使能TX中断
RX	[1]	RW	RX 使能/禁止 0 = 禁止RX 1 = 使能RX
TX	[0]	RW	TX 使能/禁止 0 = 禁止TX 1 = 使能TX

16.3.5 UART_ISR(中断状态寄存器)

Address = Base Address+ 0x00C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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								RXBRK	TIMEOUT	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
TX_DONE_INT	[19]	R	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)
RXBRK	[10]	R	接收端Break. 0 = 没有产生Break中断 1 = 产生了Break中断（读） 1 = 清除RXBRK中断(写)
TIMEOUT	[9]	R	超时 0 = 超时接收中断没发生 1 =超时接收中断发生(读取) 1= 清除超时接收中断(写)
RORMIS	[7]	R	接收FIFO溢出中断 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)
RXMIS	[6]	R	接收FIFO中断 0= RX中断没发生 1= RX中断发生(读取)
TXMIS	[5]	R	发送FIFO中断 0= TX中断没发生 1= TX中断发生(读取)

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)
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)
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)
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)
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)

16.3.6 UART_BRDIV(波特率分频寄存器)

Address = Base Address+ 0x010, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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
DIV	[19:0]	RW	波特率分频 最小值为16

16.3.7 UART_RTOR(接收超时配置寄存器)

Address = Base Address+ 0x018, Reset Value = 0x0000FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																TO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
TO	[15:0]	RW	超时配置 异步模式：超时时长 = TO[15:0] 位周期

16.3.8 UART_TTGR(发送端 Time-Guard 配置寄存器)

Address = Base Address+ 0x01C, Reset Value = 0x00000000

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

Name	Bit	Type	Description
TG	[7:0]	RW	ime-Guard配置 " Time-Guard配置位 TO[15:0] Action 0 禁止发送端的time-guard功能 1-255 UARTTX在每发送完一个字节后，会变高一段时间，这个时间段为time-guard时长 Time-guard时长 = TG[7:0] 位周期 注：如果想将此寄存器值由大变小要确保UART没有在发"发送完一个字节后的time-guard时长"。

16.3.9 UART_SRR(软件复位寄存器)

Address = Base Address+ 0x020, Reset Value = 0x00000000

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

Name	Bit	Type	Description
SWRST	[0]	W	软件复位 0 = 无效 1 = 软件复位

17 通用同步异步收发器 (USART)

17.1 概述

通用同步异步收发器(USART)用来在不同单片机之间进行通讯。USART串行发送比特位数据(低位优先)，在接收端，另外一个USART则将这些数据组合成完成数据字节。

串行数据传输通常使用在电脑之间的非网络通讯，以及终端和其它设备间的通讯。

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 主要特性

- 可编程波特率发生器
- 校验位，帧检测和数据溢出错误检测
- J1587协议的Idle标志
- 支持产生传输线打断(Break)和检测
- 支持自动应答，本地回环模式，和远程回环模式
- Multi-drop模式: 地址检测和产生
- 中断产生
- 5 到 9 位的字符长度
- 可控制的数据传输起始位
- 支持智能卡协议：产生错误信号和重新发送
- 异步模式最大波特率: PCLK/16

17.1.2 管脚描述

Table 17-1 USART管脚描述

管脚名称	功能	I/O 类型	有效电平	说明
USART_TX	USART发送数据线	O	–	–
USART_RX	USART接收数据线	I	–	–

17.2 功能描述

17.2.1 模块框图

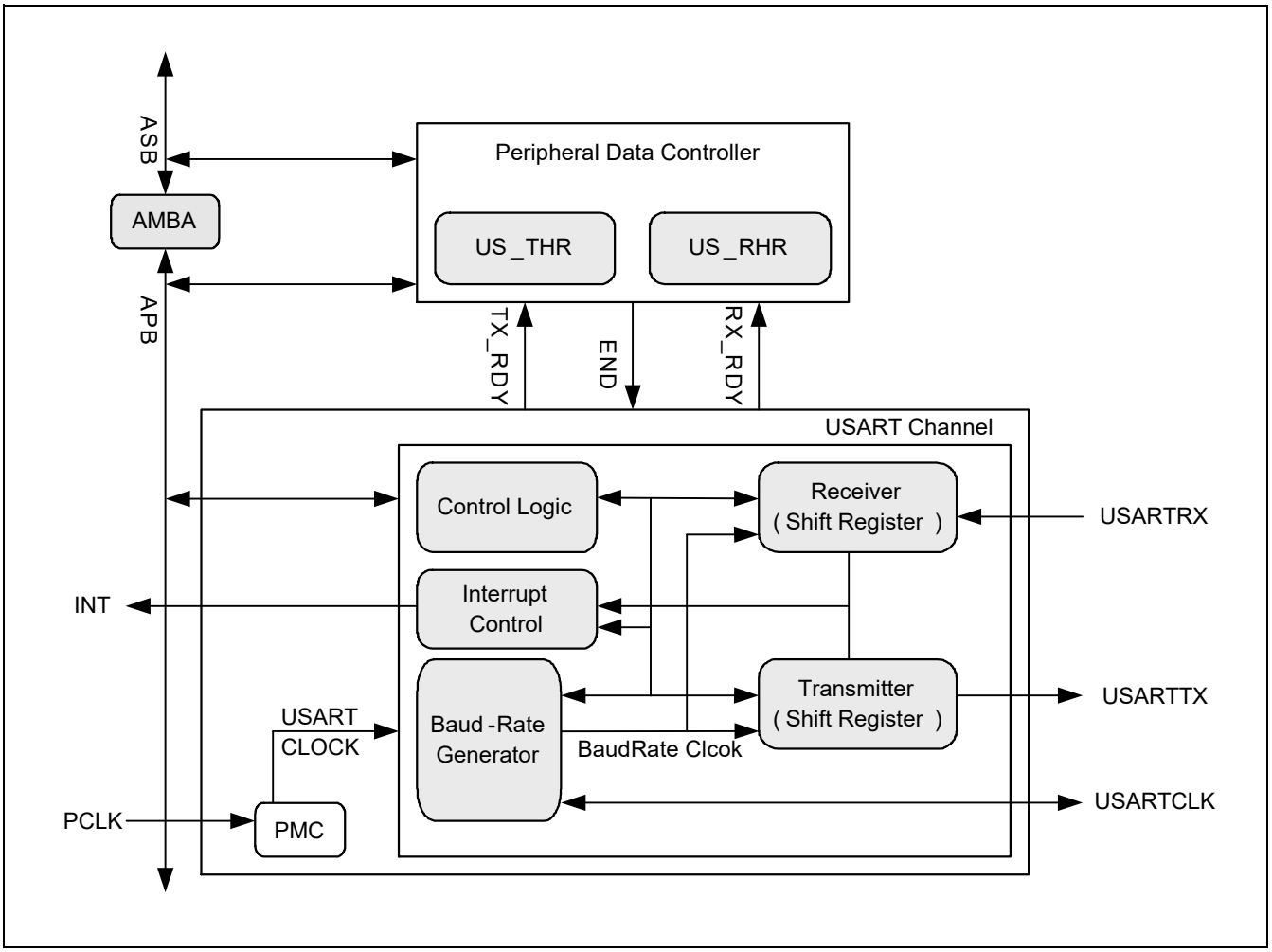


Figure 17- 1 USART模块框图

17.2.2 波特率发生器

17.2.2.1 功能描述

波特率发生器用来给发送端和接收端提供时钟，其内部时钟源可以是PCLK，或者PCLK的8分频(PCLK/8)。USART用来传送1比特所需要的时间为位周期，位周期的倒数即是波特率。

17.2.2.2 异步模式

当UART工作在异步模式时(模式寄存器US_MR的SYNC=0)，波特率为选择的时钟除以16，再除以US_BRGR(波特率配置寄存器)中CD的值。如果US_BRGR寄存器为0，那么波特率发生器的时钟被禁止。

- 波特率 = 选择的时钟/(16 × CD)，选择的时钟可以是 PCLK 或者 PCLK/ 8.

17.2.2.3 同步模式

当UART工作在同步模式(模式寄存器US_MR的SYNC=1)，并且选择的时钟为内部时钟(模式寄存器US_MR中CLKS[1]=0)时，波特率为内部选择的时钟除以US_BRGR寄存器中的值。如果US_BRGR寄存器为0，那么波特率发生器的时钟被禁止。

- 波特率 = 选择的时钟/CD，选择的时钟可以是 PCLK 或者 PCLK/ 8.

17.2.2.4 模块框图

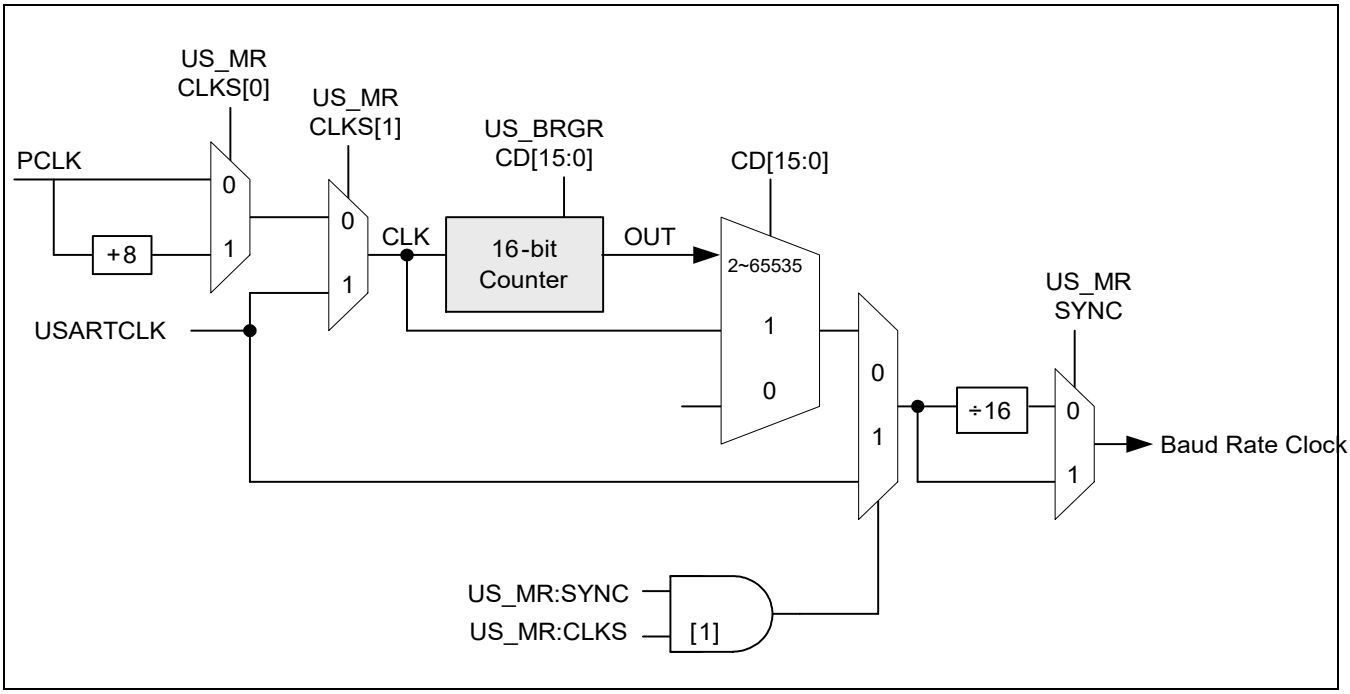


Figure 17-2 UART波特率发生器模块框图

17.2.2.5 波特率配置示例

下面表格为不同系统时钟下，US_BRGR不同配置值对应的波特率。误差栏表示实际波特率和期望波特率的差异。

下表为 CLKS[1:0] = 00 (UART时钟选择PCLK) 和 US_MR寄存器中SYNC = 0 (异步模式)的情况。

Table 17-2 异步模式 (SYNC = 0)

PCLK (MHz)	US_BRGR	波特率	% 误差
40	2083	1200	–0.02%
	1042	2400	0.03%
	521	4800	0.03%
	260	9600	–0.16%
	174	14400	0.22%
	130	19200	–0.16%
	65	38400	–0.16%
37.5	1953	1200	–0.01%
	977	2400	0.04%
	488	4800	–0.06%
	244	9600	–0.06%
	163	14400	0.15%
	122	19200	–0.06%
	61	38400	–0.06%
36	1875	1200	0.00%
	938	2400	0.05%
	469	4800	0.05%
	234	9600	–0.16%
	156	14400	–0.16%
	117	19200	–0.16%
	39	57600	–0.16%
30	1563	1200	0.03%
	781	2400	–0.03%
	391	4800	0.10%
	195	9600	–0.16%
	130	14400	–0.16%
	98	19200	0.35%
	49	38400	0.35%
20	1042	1200	0.03%
	521	2400	0.03%
	260	4800	–0.16%

PCLK (MHz)	US_BRGR	波特率	% 误差
	130	9600	–0.16%
	87	14400	0.22%
	65	19200	–0.16%
18.75	977	1200	0.04%
	488	2400	–0.06%
	244	4800	–0.06%
	122	9600	–0.06%
	81	14400	–0.47%
	61	19200	–0.06%
18	938	1200	0.05%
	469	2400	0.05%
	234	4800	–0.16%
	117	9600	–0.16%
	78	14400	–0.16%
16	833	1200	–0.04%
	417	2400	0.08%
	208	4800	–0.16%
	104	9600	–0.16%
	52	19200	–0.16%
	26	38400	–0.16%
15	781	1200	–0.03%
	391	2400	0.10%
	195	4800	–0.16%
	98	9600	0.3%
	65	14400	–0.16%
	49	19200	0.35%
10	521	1200	0.03%
	260	2400	–0.16%
	130	4800	–0.16%
	65	9600	–0.16%
9.375	488	1200	–0.06%
	244	2400	–0.06%
	122	4800	–0.06%
	61	9600	–0.06%
8	417	1200	0.08%
	208	2400	–0.16%

PCLK (MHz)	US_BRGR	波特率	% 误差
	104	4800	–0.16%
	52	9600	–0.16%
	26	19200	–0.16%
	13	38400	–0.16%
5	260	1200	–0.16%
	130	2400	–0.16%
	65	4800	–0.16%
4.6875	244	1200	–0.06%
	122	2400	–0.06%
	61	4800	–0.06%
4	208	1200	–0.16%
	104	2400	–0.16%
	52	4800	–0.16%
	26	9600	–0.16%
	13	19200	–0.16%
2.5	130	1200	–0.16%
	65	2400	–0.16%
2	104	1200	–0.16%
	52	2400	–0.16%
	26	4800	–0.16%
	13	9600	–0.16%
1.25	65	1200	–0.16%
1	52	1200	–0.16%
	26	2400	–0.16%
	13	4800	–0.16%
0.5	26	1200	–0.16%
	13	2400	–0.16%
0.25	13	1200	–0.16%

Table 17-3 同步模式 (SYNC = 1)

PCLK (MHz)	US_BRGR CD	波特率	% 误差
40	174 × 16	14400	0.22%
	130 × 16	19200	–0.16%
	65 × 16	38400	–0.16%
37.5	244 × 16	9600	–0.06%
	163 × 16	14400	0.15%
	122 × 16	19200	–0.06%
	61 × 16	38400	–0.06%
36	234 × 16	9600	–0.16%
	156 × 16	14400	–0.16%
	117 × 16	19200	–0.16%
	39 × 16	57600	–0.16%
30	195 × 16	9600	–0.16%
	130 × 16	14400	–0.16%
	98 × 16	19200	0.35%
	49 × 16	38400	0.35%
20	130 × 16	9600	–0.16%
	87 × 16	14400	0.22%
	65 × 16	19200	–0.16%
18.75	244 × 16	4800	–0.06%
	122 × 16	9600	–0.06%
	81 × 16	14400	–0.47%
	61 × 16	19200	–0.06%
18	234 × 16	4800	–0.16%
	117 × 16	9600	–0.16%
	78 × 16	14400	–0.16%
16	208 × 16	4800	–0.16%
	104 × 16	9600	–0.16%
	52 × 16	19200	–0.16%
	26 × 16	38400	–0.16%
15	195 × 16	4800	–0.16%
	98 × 16	9600	0.35%
	65 × 16	14400	–0.16%
	49 × 16	19200	0.35%
10	130 × 16	4800	–0.16%
	65 × 16	9600	–0.16%

PCLK (MHz)	US_BRGR CD	波特率	% 误差
9.375	244 × 16	2400	–0.06%
	122 × 16	4800	–0.06%
	61 × 16	9600	–0.06%
8	208 × 16	2400	–0.16%
	104 × 16	4800	–0.16%
	52 × 16	9600	–0.16%
	26 × 16	19200	–0.16%
	13 × 16	38400	–0.16%
5	130 × 16	2400	–0.16%
	65 × 16	4800	–0.16%
4.6875	244 × 16	1200	–0.06%
	122 × 16	2400	–0.06%
	61 × 16	4800	–0.06%
4	208 × 16	1200	–0.16%
	104 × 16	2400	–0.16%
	52 × 16	4800	–0.16%
	26 × 16	9600	–0.16%
	13 × 16	19200	–0.16%
2.5	130 × 16	1200	–0.16%
	65 × 16	2400	–0.16%
2	104 × 16	1200	–0.16%
	52 × 16	2400	–0.16%
	26 × 16	4800	–0.16%
	13 × 16	9600	–0.16%
1.25	65 × 16	1200	–0.16%
1	52 × 16	1200	–0.16%
	26 × 16	2400	–0.16%
	13 × 16	4800	–0.16%
0.5	26 × 16	1200	–0.16%
	13 × 16	2400	–0.16%
0.25	13 × 16	1200	–0.16%

17.2.3 接收端功能

17.2.3.1 异步接收

当SYNC = 0(US_MR中的第8位)时，UART被设置为异步工作模式。在异步模式下，UART会通过一直采样UARTRX信号来检测起始位，直到检测到一个有效的起始位为止。如果UARTRX上的一个低电平(SPACE)时长超过了7个采样时钟的周期，那么这个长低电平则会被判断为一个有效的起始位。采样时钟的频率为波特率的16倍。所以一个长于7/16位周期的低电平为有效起始位，而短于7/16位周期的低电平会接收端被忽略，然后接收端会继续等待有效的起始位。

当一个有效起始位被检测到，接收端会继续在每个位周期的理论中心点对UARTRX信号进行采样。假设每个比特的长度为采样时钟周期的16倍(1比特位)，那么采样点为该比特位开始后的第8个采样时钟周期(0.5比特位)。所以第一个采样点为起始位下降沿后的第24个采样时钟周期(1.5比特位)，接下来的每个采样点都跟前一个采样点间隔16个采样时钟周期(1比特位)。

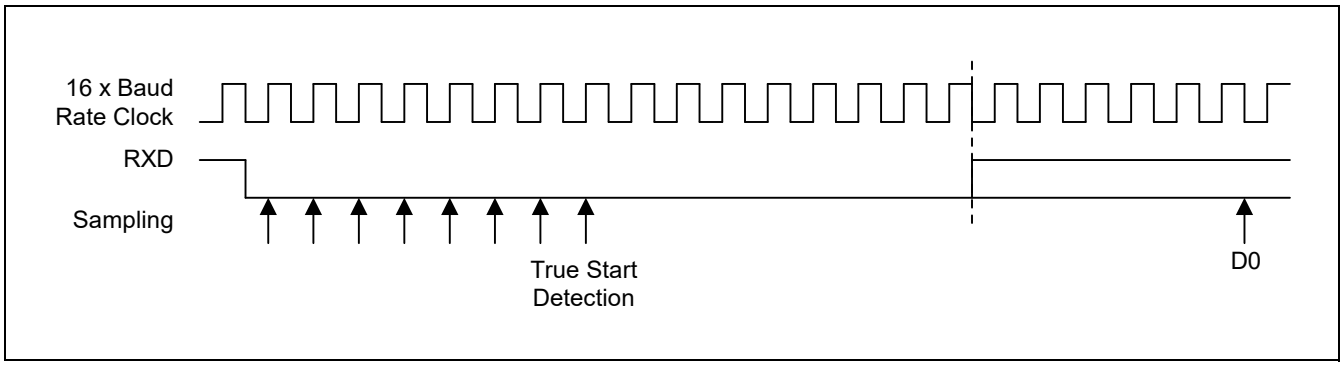


Figure 17-3 异步模式，起始位检测

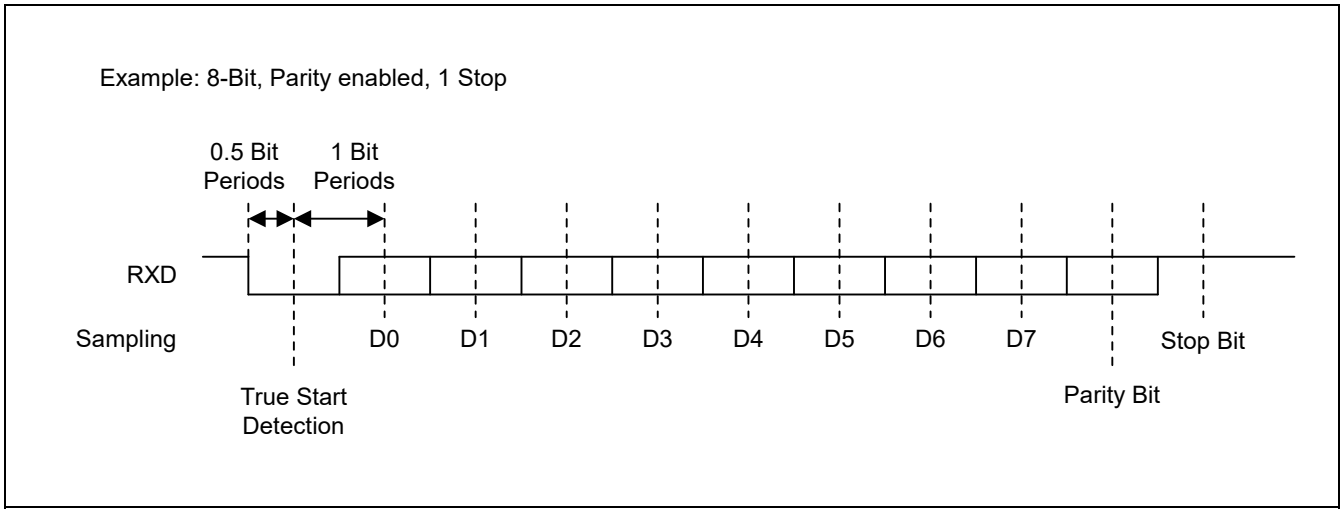


Figure 17-4 异步模式，字节接收

17.2.3.2 同步接收

当配置成同步模式(SYNC = 1)，接收端在每个USARTCLK的上升沿对RXD信号进行采样。如果检测到一个低电平，那么这个低电平就被认为是起始位。收到起始位后，接收端继续采样数据位，校验位和停止位，然后继续等待下一个起始位。参考下面的示例图。

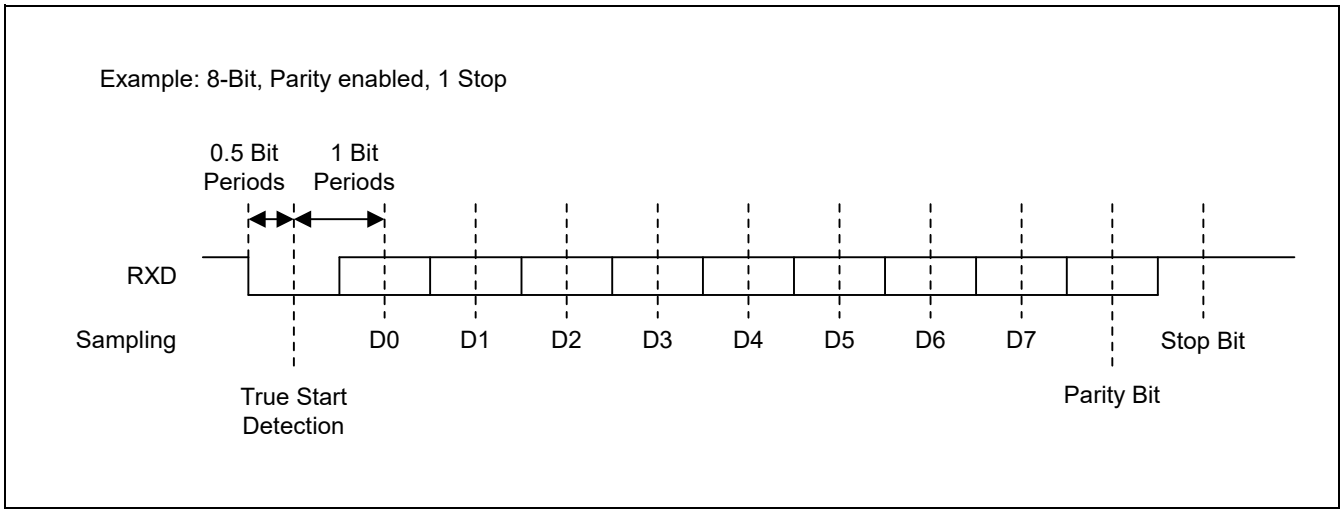


Figure 17- 5 同步模式，字节接收

17.2.3.3 接收标志位

收到一个完整的字节后，该字节会被存到US_RHR中，同时US_SR寄存器中的RXRDY标志位会被置1。RXRDY在最后的停止位结束后被置1。

17.2.3.4 溢出错误

如果US_RHR寄存器在被读取之前，又被存入了新接收的字节，那么US_SR寄存器中的OVER状态位会被置1。

17.2.3.5 校验错误

每次接收到一个字节，接收端都会根据US_MR(UART模式寄存器)中PAR[2:0]的值计算接收到数据的校验值，然后跟接收到的校验位进行比较，如果不相同，那么US_SR寄存器中的PARE校验错误位会被置1。

17.2.3.6 帧错误

如果接收到的停止位为低电平并且接收到的数据位至少有一个高电平，那么接收端会产生一个帧错误标志，将US_SR寄存器中的FRAME位置1。

17.2.3.7 空闲标志

空闲标志在UART收到一个起始位后变低，在J1587协议帧结束后(10个停止位后)变高。空闲标志位在置起时可以产生中断。

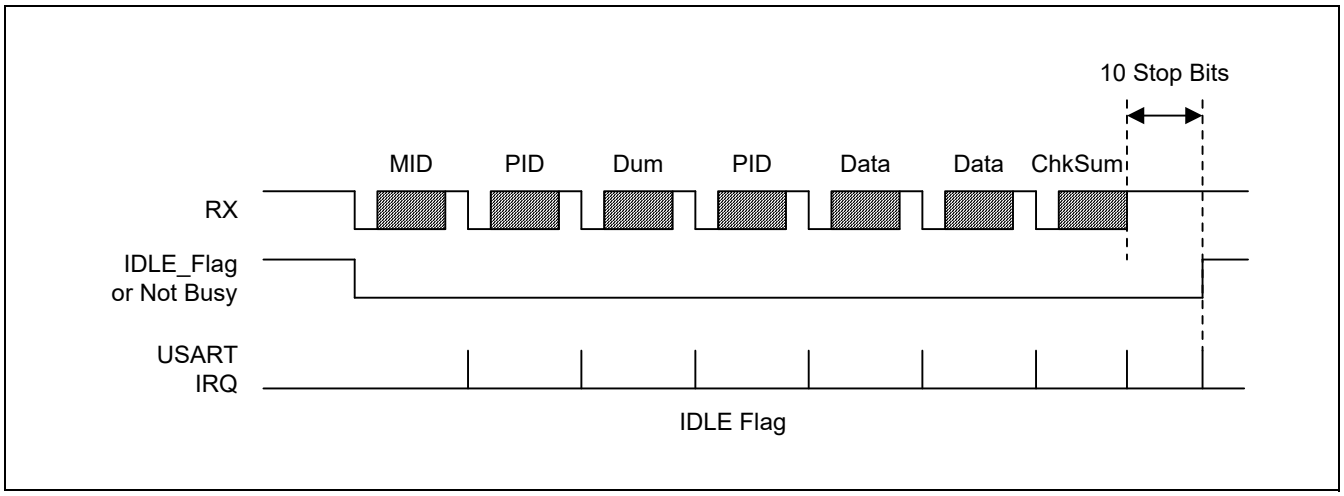


Figure 17- 6 空闲标志

17.2.3.8 超时

这个功能可以检测RXD的空闲状态。USART等待接收一个新字节的最大时间可以在US_RTOR (Receiver Time-out)寄存器的TO[15:0]里设置。当这个寄存器设置为0时，超时功能关闭。

在超时功能打开的情况下，接收端在接收到第一个字节后，会打开一个计数器，这个计数器在每个位周期自动减1，并且会在接收到字节后重新载入计数值(即TO[15:0]设置的值)。当计数器计到0时，US_SR中的TIMEOUT被置1。用户通过对US_CR寄存器中STTTO (Start Time-Out)位写1来启动(或者重新启动)这个功能。

也就是说，启动超时功能，必须满足下面条件：

- US_RTOR不为0
- US_CR寄存器中STTTO (Start Time-Out)位写1
- 收到一个字节

超时的时长计算：

时长 = 寄存器值(TO[15:0]) × 位周期 (异步模式)
时长 = 寄存器值(TO[15:0]) × 16 × 采样周期 (同步模式)

17.2.4 发送端

17.2.4.1 功能描述

发送功能在同步模式和异步模式下的行为是完全一样的。发送端将开始位，数据位，校验位和停止位串行移位出去，低位(LSB)先发，高位(MSB)后发。

数据位的个数由US_MR寄存器中的CHRL[1:0]选择。

校验位由US_MR寄存器中的PAR[2:0]位设置。如果校验为偶校验，那么校验位是所有数据位的和(单比特相加的和)。如果校验为奇校验，那么校验位是所有数据位的和取反。

停止位的个数由US_MR寄存器中的NBSTOP[1:0]选择。

当需要传送的字节被写入到US_THR (Transmit Holding)中时，只要移位寄存器是空的，那么该字节会被马上复制到移位寄存器中。

当发送操作发生时，US_SR中的TXRDY位会被置1，直到US_THR寄存器被写入了新的字节。如果移位寄存器和US_THR寄存器都是空的，那么US_SR中的TXEMPTY会被置1 (在最后的停止位之后)。

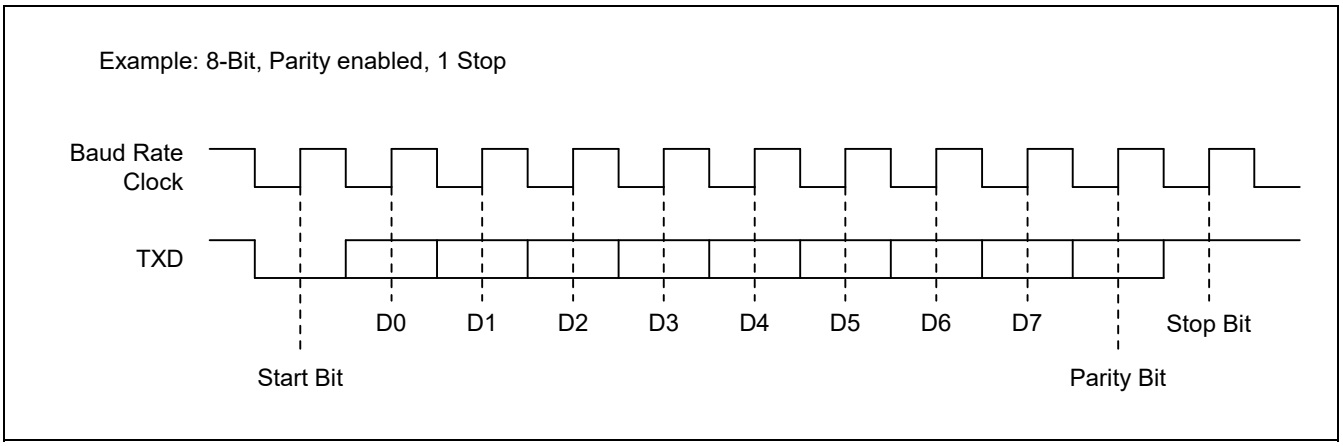


Figure 17-7 同步和异步模式，字节发送

17.2.4.2 Time-Guard

Time-guard功能可以在USARTTX上发送的2个字节中间插入一段空闲时间。这个空闲状态的时长在US_TTGR (Transmitter Time-Guard)寄存器中设置。当这个寄存器为0时，不产生time-guard。否则，发送端会在每次发送完一个字节后，将USARTTX拉高保持一段时间，时长为US_TTGR中设置的值乘以位周期。

17.2.4.3 Multi-Drop模式

当US_MR中PAR位等于11Xb时，USART被配置为multi-drop模式，用来自动检测地址和数据，这时候PARE (US_SR寄存器中的校验错误位)被用来区分数据字节(校验位为0)和地址字节(校验位为1)。所以在这个模式中，如果数据为一个地址字节，那么校验错误位(US_SR中的PARE)被置1。PARE状态可以由US_CSR(状态清除寄存器)中的PARE位来清除。如果校验位为0，表示该字节为数据字节，PARE不会被置1。

当发送地址命令(SENDA)被写入到US_CR中时，发送端发送的是地址字节(校验位置1)。这种情况下，写入到US_THR中的下个字节会被当作地址发送出去(校验位为1)，而在这个字节后的任何字节的校验位都为0。

17.2.5 Break

17.2.5.1 发送Break

当US_CR寄存器中的STTBRK (Start break)命令被置1时，发送端会在USARTTX上发送Break。在USARTTX被拉低之前，发送移位寄存器中的字节会被发送完。

如果要移除Break，那么必须将US_CR中的STOPBRK (Stop break)命令置1。USART最少发送一个字节长的Break。

之后USARTTX会恢复到高电平(空闲状态)并且持续12个位周期，保证Break被正确的检测到，然后发送端继续正常的操作。

17.2.5.2 接收Break

当所有的数据，校验和停止位都是0时，接收端认为检测到了Break。在检测到低电平地址位的时刻，接收端将US_SR中的RXBRK (Break received)位置1。

17.2.6 中断

US_SR中的大部分状态都在US_IMSCR (中断使能/禁止寄存器), US_RISR (原始中断状态寄存器), US_MISR (中断状态寄存器), 和US_ICR (中断状态清除寄存器) 中有对应的位，可以控制中断的产生。

17.2.7 测试模式

USART可以用US_MR中的CHMODE[1:0]配置成3种不同的测试模式。

自动回应模式：自动重新发送收到的数据，对发送端的任何配置都无效。

本地回环模式：接收自己发送的数据，不使用USARTTX和USARTRX管脚，而是内部将发送端的输出连到接收端的输入，USARTRX管脚的电平高低没有任何用途，并且USARTTX管脚会被一直拉高，就像是在空闲状态。

远程回环模式：直接将USARTTX管脚接到USARTRX管脚上，USART模块的发送和接收功能都禁止，只是负责将收到的数据直接转出去而不经过USART模块。

17.2.8 Smart-Card协议

USART兼容ISO7816-3协议，允许字节重送和校验错检测。

下面的描述只有在US_MR寄存器中的SMCARDPT位为1的时候才有效。

USART的Smart-Card协议需要发送端和接收端都支持。

如果GPIO模块允许，USARTTX可以配置成开漏输出模式，并且直接接到USARTRX管脚上，同时连接一个外部的上拉电阻，形成Smart-Card的数据信号线。

17.2.8.1 发送字节到Smart Card

USART可以通过检测Smart Card产生的校验错误信号来判断Smart Card是否正确的收到了上一个发送的字节。

当Smart Card产生了校验错的信号，上一个发送的字节会被USART重新发送，US_MR寄存器中的SENDTIME[2:0]用来控制重新发送的次数，直到Smart Card不再产生错误信号。

当USART检测到错误信号时，US_SR寄存器中的FRAME错误标志位会被置1。

USART检测错误信号的时间点是 t0 + t11 比特位，t0为开始位的下降沿 (也就是在2个停止位中间)。

下面的例子中，Smart Card检测到了校验错，在数据线(COMMS)上产生了一个错误信号。这个错误信号被USART检测到并且重新发送了上一个字节。

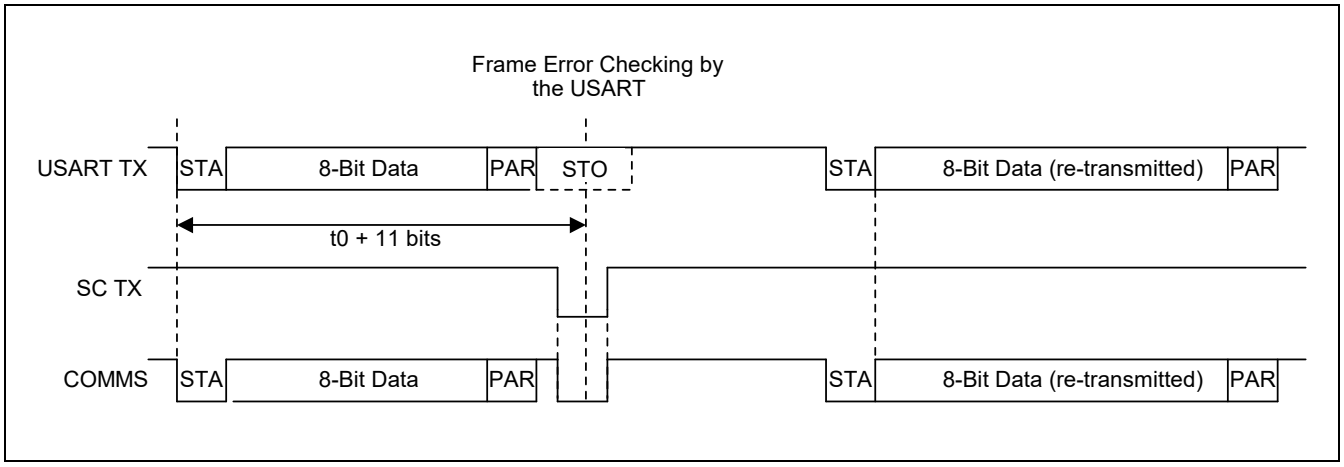


Figure 17-8 Smart-Card 发送错误

17.2.8.2 从Smart Card接收字节

当接收的字节有校验错时，USART可以产生错误信号 (参考ISO7816-3协议)。

当USART检测到校验错时，USART会在 t0 + 10.625 + [0:0.0625] 时间点(2个停止位中间)将传输线拉低1.0625个位周期，通知Smart Card上一个数据接收错误。使用T=0协议类型的Smart Card，必须重新发送该字节。

在这个情况下，USART会将US_SR寄存器中的PARE位置1，表示校验有错误。

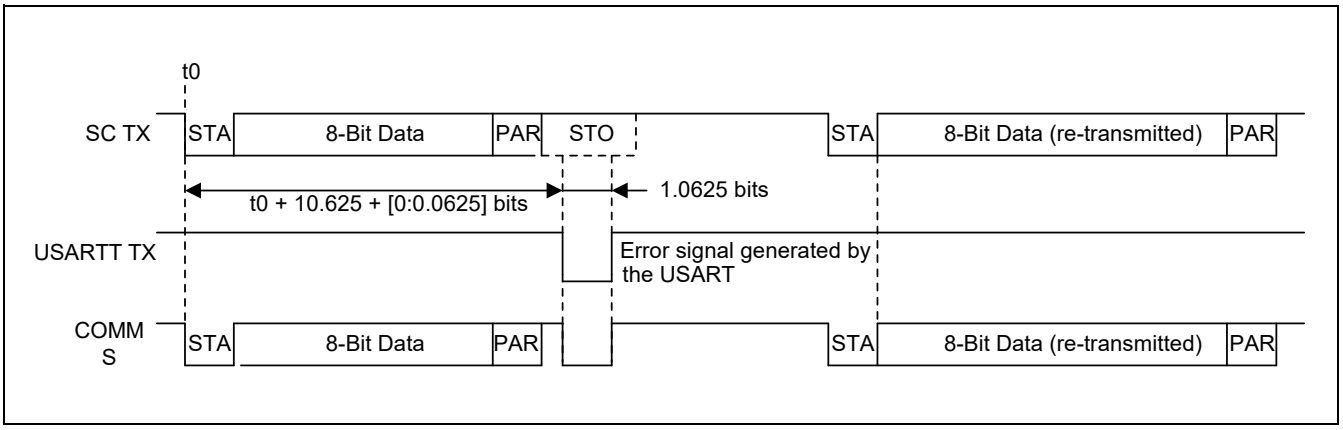


Figure 17- 9 接收端的错误检测

17.2.8.3 Smart Card模式下的USART配置

要工作在Smart Card模式下，USART必须设置成普通模式，并且停止位的个数必须设置为2 (参考US_MR模式寄存器)。

17.3 寄存器说明

17.3.1 寄存器表

Base Address of USART0: 0x40080000

Register	Offset	Description	Reset Value
US_IDR	0x00	ID寄存器	0x00003898
US_CEDR	0x04	时钟使能/禁止寄存器	0x00000000
US_SRR	0x08	软件复位寄存器	0x00000000
US_CR	0x0C	控制寄存器	0x00000000
US_MR	0x10	模式寄存器	0x00000000
US_IMSCR	0x14	中断使能/禁止寄存器	0x00000000
US_RISR	0x18	原始中断状态寄存器	0x00004202
US_MISR	0x1C	中断状态寄存器	0x00000000
US_ICR	0x20	中断状态清除寄存器	0x00000000
US_SR	0x24	状态寄存器	0x00064A02
US_RHR	0x28	接收数据寄存器	0x00000000
US_THR	0x2C	发送数据寄存器	0x00000000
US_BRGR	0x30	波特率配置寄存器	0x00000000
US_RTOR	0x34	接收超时配置寄存器	0x00000000
US_TTGR	0x38	发送端Time-Guard寄存器	0x00000000

17.3.2 US_IDR(ID 寄存器)

Address = Base Address+ 0x00, Reset Value = 0x00003898

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

Name	Bit	Type	Description
IDCODE	[25:0]	R	ID寄存器 IP的ID代码

17.3.3 US_CEDR(时钟使能/禁止寄存器)

Address = Base Address+ 0x04, Reset Value = 0x00000000

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

Name	Bit	Type	Description
DBGEN	[31]	RW	调试模式使能/禁止控制位 0 = 禁止调试模式1 = 使能调试模式 说明： 0 = 进入调试模式后不影响USART功能1 = 进入调试模式后冻结USART的功能，但USART内部寄存器的读写不受影响
CLKEN	[0]	RW	时钟使能/禁止控制位 0 = 时钟禁止1 = 时钟使能

17.3.4 US_SRR(软件复位寄存器)

Address = Base Address+ 0x08, Reset Value = 0x00000000

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

Name	Bit	Type	Description
SWRST	[0]	W	软件复位 0 = 无效 1 = 软件复位

17.3.5 US_CR(控制寄存器)

Address = Base Address+ 0x0C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																RXIFLSEL		FIFO_EN	SENDA	STTTO	STPBRK	STTBRK	RSVD	TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	RSVD	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W	R	W	W	W	W	W	W	R	R

Name	Bit	Type	Description
RXIFLSEL	[16:14]	W	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
FIFO_EN	[13]	W	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
SENDA	[12]	W	发送地址 0 = 无效 1 = 只在Multi-drop模式，下一个写入US_THR的字节会被当作地址字节发送
STTTO	[11]	W	开启超时接收 0 = 无效 1 = 必须在超时计数器计数完成之前，接收到字节数据，否则报错
STPBRK	[10]	W	停止Break. 0 = 无效 1 = 如果一个Break状态正在发送，那么写1会在最少一个字节长度的Break状态后停止Break，并且发送一个12位周期的高电平
STTBRK	[9]	W	开始Break. 0 = 无效 1 = 如果Break没有发送，那么写1会在当前移位寄存器中的数据发送完之后，开始发送Break状态
TXDIS	[7]	W	发送禁止 0 = 无效 1 = 发送禁止

TXEN	[6]	W	发送使能 0 = 无效 1 = 如果TXDIS是0，写1使能发送
RXDIS	[5]	W	接收禁止 0 = 无效 1 = 接收禁止
RXEN	[4]	W	接收使能 0 = 无效 1 = 如果RXDIS是0，写1使能接收
RSTTX	[3]	W	复位发送端 0 = 无效 1 = 复位发送端逻辑
RSTRX	[2]	W	复位接收端 0 = 无效 1 = 复位接收端逻辑

17.3.6 US_MR(模式寄存器)

Address = Base Address+ 0x10, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD											DSB	RSVD	CLKO	MODE9	SMCARDPT	CHMODE		NBSTOP		PAR			SYNC	CHRL		CLKS		SENDTIME			RSVD
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R

Name	Bit	Type	Description
DSB	[20]	RW	数据开始位选择 0 = 数据发送从低位LSB开始，到高位MSB结束 1 = 数据发送从高位MSB开始，到低位LSB结束
CLKO	[18]	RW	时钟输出选择 0 = USART不输出USARTCLK 1 = 如果CLKS[1]是0，USART输出USARTCLK
MODE9	[17]	RW	9位字节长度 0 = CHRL位定义字节长度 1 = 9位字节长度
SMCARDPT	[16]	RW	Smart Card协议 0 = 禁止smart card协议 1 = 使能smart card协议
CHMODE	[15:14]	RW	通道模式 • 通道模式位 CHMODE [1:0] 模式描述 00 普通模式USART通道工作为正常的Rx/Tx功能 01 自动回应收到的数据自动通过USARTTX发送 10 本地回环发送端的输出信号短接到接收端的输入信号 11 远程回环USARTRX管脚内部直接短接到USARTTX管脚
NBSTOP	[13:12]	RW	停止位的个数 停止位个数跟SYNC设置的模式有关

			• NBSTOP配置位 NBSTOP [1:0] 异步模式 (SYNC = 0) 同步模式 (SYNC = 1) 00 1个停止位 1个停止位 01 1.5个停止位 保留 10 2个停止位 2个停止位 11 保留 保留
PAR	[11:9]	RW	校验类型 • 校验类型位 PAR[2:0] 校验类型 000 偶校验 001 奇校验 010 0校验(Space) 011 1校验(Mark) 10X 无校验 11X Multi-drop模式 注意：如果使用LIN，PAR[2:0]必须设置为‘10X’.
SYNC	[8]	RW	同步模式选择 0 = USART工作在异步模式 1 = USART工作在同步模式
CHRL	[7:6]	RW	字节长度 (除开始位，停止位和校验位外的字节长度) • 字节长度位 CHRL[1:0] 字节长度 0 0 5位 0 1 6位 1 0 7位 1 1 8位
CLKS	[5:4]	RW	时钟选择 (波特率发生器的输入时钟). • CLKS 时钟选择位

			CLKS[1:0] 选择的时钟 0 0 PCLK 0 1 PCLK/8 1 x 保留，请勿使用
SENDTIME	[3:1]	RW	表示USART被配置成Smart Card协议时，重复发送最多的次数0-7

17.3.7 US_IMSCR(中断使能/禁止寄存器)

Address = Base Address+ 0x14, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	RORRIS	RXRIS	RSVD	IDLE	TXEMPTY	TIMEOUT	PARE	FRAME	OVRE	RSVD		RXBRK	TXRDY	RXRDY
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW

Name	Bit	Type	Description
TXRIS	[14]	RW	发送FIFO中断 0 = 禁止中断 1 = 使能中断
RORRIS	[13]	RW	接收FIFO溢出中断 0 = 禁止中断 1 = 使能中断
RXRIS	[12]	RW	接收FIFO中断 0 = 禁止中断 1 = 使能中断
IDLE	[10]	RW	空间中断 0 = 禁止中断 1 = 使能中断
TXEMPTY	[9]	RW	发送缓冲空闲中断 0 = 禁止中断 1 = 使能中断
TIMEOUT	[8]	RW	超时中断 0 = 禁止中断 1 = 使能中断
PARE	[7]	RW	校验错中断 0 = 禁止中断 1 = 使能中断
FRAME	[6]	RW	帧错误中断 0 = 禁止中断 1 = 使能中断
OVRE	[5]	RW	溢出错误中断

			0 = 禁止中断 1 = 使能中断
RXBRK	[2]	RW	接收端Break中断 0 = 禁止中断 1 = 使能中断
TXRDY	[1]	RW	发送端待机中断 0 = 禁止中断 1 = 使能中断
RXRDY	[0]	RW	接收端待机中断 0 = 禁止中断 1 = 使能中断

17.3.8 US_RISR(原始中断状态寄存器)

Address = Base Address+ 0x18, Reset Value = 0x00004202

31	30	29	28	27	26	25	24	23	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	RORRIS	RXRIS	RSVD	IDLE	TXEMPTY	TIMEOUT	PARE	FRAME	OVRE	RSVD		RXBRK	TXRDY	RXRDY
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	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
TXRIS	[14]	R	发送FIFO中断原始状态 TXRIS的原始中断状态，不管该中断是使能还是禁止
RORRIS	[13]	R	接收FIFO溢出中断原始状态 RORRIS的原始中断状态，不管该中断是使能还是禁止
RXRIS	[12]	R	接收FIFO中断原始状态 RXRIS的原始中断状态，不管该中断是使能还是禁止
IDLE	[10]	R	空闲中断原始状态 IDLE的原始中断状态，不管该中断是使能还是禁止
TXEMPTY	[9]	R	发送缓冲空闲中断原始状态 TXEMPTY的原始中断状态，不管该中断是使能还是禁止
TIMEOUT	[8]	R	超时中断原始状态 TIMEOUT的原始中断状态，不管该中断是使能还是禁止
PARE	[7]	R	帧错误中断原始状态 PARE的原始中断状态，不管该中断是使能还是禁止
FRAME	[6]	R	帧错误中断原始状态 FRAME的原始中断状态，不管该中断是使能还是禁止
OVRE	[5]	R	溢出错误中断原始状态 OVRE的原始中断状态，不管该中断是使能还是禁止
RXBRK	[2]	R	接收端Break中断原始状态 RXBRK的原始中断状态，不管该中断是使能还是禁止
TXRDY	[1]	R	发送端待机中断原始状态 TXRDY的原始中断状态，不管该中断是使能还是禁止
RXRDY	[0]	R	接收端待机中断原始状态 RXRDY的原始中断状态，不管该中断是使能还是禁止

17.3.9 US_MISR(中断状态寄存器)

Address = Base Address+ 0x1C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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	RORRIS	RXRIS	RSVD	IDLE	TXEMPTY	TIMEOUT	PARE	FRAME	OVRE	RSVD		RXBRK	TXRDY	RXRDY
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
TXRIS	[14]	R	发送FIFO中断状态 TXRIS中断使能后的状态
RORRIS	[13]	R	接收FIFO溢出中断状态 RORRIS中断使能后的状态
RXRIS	[12]	R	接收FIFO中断状态 RXRIS中断使能后的状态
IDLE	[10]	R	空闲中断状态 IDLE中断使能后的状态
TXEMPTY	[9]	R	发送缓冲空闲中断状态 TXEMPTY中断使能后的状态
TIMEOUT	[8]	R	超时中断状态 TIMEOUT中断使能后的状态
PARE	[7]	R	帧错误中断状态 PARE中断使能后的状态
FRAME	[6]	R	帧错误中断状态 FRAME中断使能后的状态
OVRE	[5]	R	溢出错误中断状态 OVRE中断使能后的状态
RXBRK	[2]	R	接收端Break中断状态 RXBRK中断使能后的状态
TXRDY	[1]	R	发送端待机中断状态 TXRDY中断使能后的状态
RXRDY	[0]	R	接收端待机中断状态 RXRDY中断使能后的状态

17.3.10 US_ICR(中断状态清除寄存器)

Address = Base Address+ 0x20, Reset Value = 0x00000000

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

Name	Bit	Type	Description
IDLE	[10]	W	空闲中断状态 0 = 无效 1 = 清除该中断状态
TIMEOUT	[8]	W	超时中断状态 0 = 无效 1 = 清除该中断状态
PARE	[7]	W	帧错误中断状态 0 = 无效 1 = 清除该中断状态
FRAME	[6]	W	帧错误中断状态 0 = 无效 1 = 清除该中断状态
OVRE	[5]	W	溢出错误中断状态 0 = 无效 1 = 清除该中断状态
RXBRK	[2]	W	接收端Break中断状态 0 = 无效 1 = 清除该中断状态

17.3.11 US_SR(状态寄存器)

Address = Base Address+ 0x24, Reset Value = 0x00064A02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													TFE	TNF	RNE	RFF	TXRIS	RORRIS	RXRIS	IDLEFLAG	IDLE	TXEMPTY	TIMEOUT	PARE	FRAME	OVRE	RSVD		RXBRK	TXRDY	RXRDY
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	1	0	1	0	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
TFE	[18]	R	发送FIFO是否为空状态位 0 = 发送FIFO非空 1 = 发送FIFO为空
TNF	[17]	R	发送FIFO是否已满状态位 0 = 发送FIFO已满 1 = 发送FIFO未满
RNE	[16]	R	接收FIFO是否为空状态位 0 = 接收FIFO为空 1 = 接收FIFO非空
RFF	[15]	R	接收FIFO是否已满状态位 0 = 接收FIFO未满 1 = 接收FIFO已满
TXRIS	[14]	R	发生FIFO状态位 0 = 发生FIFO大于4 1 = 发生FIFO小于等于4
RORRIS	[13]	R	接收FIFO溢出状态位 0 = 接收FIFO未溢出 1 = 接收FIFO溢出
RXRIS	[12]	R	接收FIFO状态位 0 = 接收FIFO小于中断触发点 1 = 接收FIFO大于等于中断触发点 接收FIFO中断触发点选择请参考US_CR(USART控制寄存器)的RXIFLSEL位
IDLEFLAG	[11]	R	0 = USART正在接收一个帧 1 = USART没有在接收任何帧

			该位表示J1587协议的帧传送状态，当接收开始时变低，在接收完成+10个停止位(10个周期的高电平)后变高。
IDLE	[10]	R	空闲状态 0 = 没有检测到J1587的结束帧 1 = 检测到J1587的结束帧
TXEMPTY	[9]	R	发送缓冲空闲 0 = US_THR寄存器或者发送缓冲寄存器中有字节待发送 1 = US_THR寄存器或者发送缓冲寄存器中没有字节待发送 当USART被禁止或者复位后，该位为0，US_CR中的发送使能功能会将该位置1。
TIMEOUT	[8]	R	超时 0 = 开始超时接收后，没有检测到超时，或者超时寄存器被设置为0 1 = 开始超时接收后，检测到了超时
PARE	[7]	R	校验错误 0 = 在上一次状态复位后，没有检测到校验位错(或者multi-drop模式下的数据字节) 1 = 在上一次状态复位后，检测到至少1个校验位错(或者multi-drop模式下的地址字节)
FRAME	[6]	R	帧错误 0 = 在上一次状态复位后，没有停止位被检测到低电平 1 = 在上一次状态复位后，至少有一个停止位被检测到低电平
OVRE	[5]	R	溢出错误 0 = 当RXRDY有效后，没有字节从接收移位寄存器传到US_RHR寄存器 1 = 当RXRDY有效后，至少有一个字节从接收移位寄存器传到了US_RHR寄存器
RXBRK	[2]	R	接收端Break. 0 = 在上一次状态复位后，还没有检测到Break 1 = 在上一次状态复位后，检测到Break
TXRDY	[1]	R	发送端待机 0 = US_THR中有一个字节正在等待发送到移位寄存器中，或者发送端被禁止 1 = US_THR中没有任何字节,等于0表示USART被禁止了，或者处于复位状态。US_CR中的发送使能命令会将这位置1。
RXRDY	[0]	R	接收端待机 0 = 自从上次读取US_RHR后没有收到任何完成的字节，或者接收端被禁止

			1 = 自从上次读取US_RHR后没有收到了至少一个完成的字节

17.3.12 US_RHR(接收数据寄存器)

Address = Base Address+ 0x28, Reset Value = 0x00000000

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

Name	Bit	Type	Description
RXCHR	[8:0]	R	接收到的字节 当RXRDY有效时，储存接收到的字节。当位数小于9时，数据为右对齐。 注意： 读取此寄存器后，RXRDY位会被自动清除。在调试模式，用户可以使用镜像寄存器来避免RXRDY被清除。

17.3.13 US_THR(发送数据寄存器)

Address = Base Address+ 0x2C, Reset Value = 0x00000000

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

Name	Bit	Type	Description
TXCHR	[8:0]	R	需要发送的字节 当TXRDY有效时，存储下一个要发送的字节。如果TXRDY是0，那么当前US_THR寄存器的值会被覆盖。当位数小于9时，数据为右对齐。

17.3.14 US_BRGR(波特率配置寄存器)

Address = Base Address+ 0x30, Reset Value = 0x00000000

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

Name	Bit	Type	Description
CD	[15:4]	RW	分频 同步模式下如果选择了外部时钟，此寄存器无效。 异步模式):CD = FLOOR[Fclk / (Baud Rate x 16), 1] 同步模式):CD = FLOOR[Fclk / Baud Rate, 1] CD[15:4] Action 0 关闭时钟 1 无分频 (1分频) 2 to 65535 波特率(异步模式)= Fclk /(16 × (CD+FRACTION/16)) 波特率(同步模式)= Fclk /(CD+FRACTION/16)
FRACTION	[3:0]	RW	小数修正值(0-15) 同步模式下如果选择了外部时钟，此寄存器无效。 异步模式:FRACTION = ROUND[((Fclk / (Baud Rate x 16)) - CD) x 16, 0] 同步模式:FRACTION = ROUND[((Fclk / Baud Rate) - CD) x 16, 0] 波特率(异步模式)= Fclk / (16 x (CD + FRACTION/16)) 波特率(同步模式)= Fclk / (CD + FRACTION/16)

17.3.15 US_RTOR(接收超时配置寄存器)

Address = Base Address+ 0x34, Reset Value = 0x00000000

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

Name	Bit	Type	Description
TO	[15:0]	RW	超时配置 给这个寄存器写值后，会自动开启超时接收的指令 • 超时配置位 TO[15:0] Action 0 禁止接收端的超时功能 1－65565 当开启超时接收时，或者每当收到一个数据字节时，超时计数器会被载入TO[15:0]的值 异步模式：超时时长= TO[15:0] × 位周期 同步模式：超时时长= TO[15:0] × 16 × 位周期 注意： 当设置US_CR寄存器的RXDIS位禁止接收端后，超时功能被停止，这时如果又通过US_CR的RXEN位重新使能了接收端，那么超时计数器会从刚才停止的地方继续开始(不会被复位)。

17.3.16 US_TTGR(发送端 Time-Guard 寄存器)

Address = Base Address+ 0x38, Reset Value = 0x00000000

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

Name	Bit	Type	Description
TG	[7:0]	RW	Time-Guard配置 Time-Guard配置位 TG[7:0] Action 0 禁止发送端的time-guard功能 1–255 USARTTX在每发送完一个字节后，会变高一段时间，这个时间段为time-guard时长 Time-guard时长 = TG[7:0] x位周期

4MHz–40MHz Asynchronous Mode (SYNC = 0)

Table 17-4 Asynchronous Mode (SYNC = 0)

SYSCLK	PDIV	PCLK	US_BRGR CD[15:0]	Baud Rate	Result	% Error
4MHz	1	4	208	1200	1201.92	–0.16%
			104	2400	2403.85	–0.16%
			52	4800	4807.69	–0.16%
			26	9600	9615.38	–0.16%
			13	19200	19230.77	–0.16%
	2	2	104	1200	1201.92	–0.16%
			52	2400	2403.85	–0.16%
			26	4800	4807.69	–0.16%
			13	9600	9615.38	–0.16%
	4	1	52	1200	1201.92	–0.16%
			26	2400	2403.85	–0.16%
			13	4800	4807.69	–0.16%
	8	0.5	26	1200	1201.92	–0.16%
			13	2400	2403.85	–0.16%
	16	0.25	13	1200	1201.92	–0.16%
8MHz	1	8	417	1200	1199.04	0.08%
			208	2400	2403.85	–0.16%
			104	4800	4807.69	–0.16%
			52	9600	9615.38	–0.16%
			26	19200	19230.77	–0.16%
			13	38400	38461.54	–0.16%
	2	4	208	1200	1201.92	–0.16%
			104	2400	2403.85	–0.16%
			52	4800	4807.69	–0.16%
			26	9600	9615.38	–0.16%
			13	19200	19230.77	–0.16%
	4	2	104	1200	1201.92	–0.16%
			52	2400	2403.85	–0.16%
			26	4800	4807.69	–0.16%
			13	9600	9615.38	–0.16%
	8	1	52	1200	1201.92	–0.16%
			26	2400	2403.85	–0.16%
			13	4800	4807.69	–0.16%

SYSCLK	PDIV	PCLK	US_BRGR CD[15:0]	Baud Rate	Result	% Error
16MHz	16	0.5	26	1200	1201.92	–0.16%
			13	2400	2403.85	–0.16%
	1	16	833	1200	1200.48	–0.04%
			417	2400	2398.08	0.08%
			208	4800	4807.69	–0.16%
			104	9600	9615.38	–0.16%
			52	19200	19230.77	–0.16%
			26	38400	38461.54	–0.16%
	2	8	417	1200	1199.04	0.08%
			208	2400	2403.85	–0.16%
			104	4800	4807.69	–0.16%
			52	9600	9615.38	–0.16%
			26	19200	19230.77	–0.16%
			13	38400	38461.54	–0.16%
	4	4	208	1200	1201.92	–0.16%
			104	2400	2403.85	–0.16%
			52	4800	4807.69	–0.16%
			26	9600	9615.38	–0.16%
			13	19200	19230.77	–0.16%
	8	2	104	1200	1201.92	–0.16%
			52	2400	2403.85	–0.16%
			26	4800	4807.69	–0.16%
			13	9600	9615.38	–0.16%
	16	1	52	1200	1201.92	–0.16%
			26	2400	2403.85	–0.16%
			13	4800	4807.69	–0.16%
20MHz	1	20	1042	1200	1199.62	0.03%
			521	2400	2399.23	0.03%
			260	4800	4807.69	–0.16%
			130	9600	9615.38	–0.16%
			87	14400	14367.82	0.22%
			65	19200	19230.77	–0.16%
	2	10	521	1200	1199.62	0.03%
			260	2400	2403.85	–0.16%
			130	4800	4807.69	–0.16%

SYSCLK	PDIV	PCLK	US_BRGR CD[15:0]	Baud Rate	Result	% Error
			65	9600	9615.38	–0.16%
	4	5	260	1200	1201.92	–0.16%
			130	2400	2403.85	–0.16%
			65	4800	4807.69	–0.16%
	8	2.5	130	1200	1201.92	–0.16%
			65	2400	2403.85	–0.16%
	16	1.25	65	1200	1201.92	–0.16%
40MHz	1	40	2083	1200	1200.19	–0.02%
			1042	2400	2399.23	0.03%
			521	4800	4798.46	0.03%
			260	9600	9615.38	–0.16%
			174	14400	14367.82	0.22%
			130	19200	19230.77	–0.16%
			65	38400	38461.54	–0.16%
	2	20	1042	1200	1199.62	0.03%
			521	2400	2399.23	0.03%
			260	4800	4807.69	–0.16%
			130	9600	9615.38	–0.16%
			87	14400	14367.82	0.22%
			65	19200	19230.77	–0.16%
	4	10	521	1200	1199.62	0.03%
			260	2400	2403.85	–0.16%
			130	4800	4807.69	–0.16%
			65	9600	9615.38	–0.16%
	8	5	260	1200	1201.92	–0.16%
			130	2400	2403.85	–0.16%
			65	4800	4807.69	–0.16%
	16	2.5	130	1200	1201.92	–0.16%
			65	2400	2403.85	–0.16%

18 模拟比较器（CMP）

18.1 概述

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

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 特性

- 支持最大6个独立模拟比较器。
- 每个模拟比较器支持正负端外部输入，比较结果直接 IO 输出。
  - 比较器的正向端可以选择外部输入，或者运放输出作为输入。
  - 比较器的负向端可以选择外部输入，或者内部参考电压。
- 126个内部参考电压
  - 支持126个从 VDD 分压得到的内部参考电压（最小分辨率：VDD/126）。
  - 支持 FVR 作为比较器输入参考电压源
  - 每个模拟比较器可以独立设置不同的参考电压输入。
- 可配置的比较器输出极性。
- 可配置的输出迟滞特性。
- 可配置的比较器输出数字滤波选择。
- 可配置的捕获滤波器选择。
- 中断触发模式选择（上升沿、下降沿）。
- 支持中断硬件自动触发 ADC 转换。
- 支持比较器触发 EPWM
- 支持比较器和 TC1联动
  - 触发 TC1计数。
  - 计较器输出作为 TC1的时钟，对比较器翻转个数进行计数。
  - 比较器输出作为 TC1的 Capture 输入，对比较器的输出电平宽度进行测量。

18.1.2 管脚描述

Table 18-1 CMP 管脚描述

管脚名称	功能	I/O类型	有效状态	备注
CPINP[9:0]	比较器模拟输入正向端通道	A	-	-
CPINN[3:0]	比较器模拟输入负向端通道	A	-	-
CPx_OUT	比较器输出输出	O	-	-

18.2 功能描述

Table 18-2 CMP 功能列表

序号	功能				
	迟滞	数字滤波	窗口捕获	触发输入	触发输出
CMP0	O	O	-	-	O
CMP1	O	O	-		O
CMP2	O	O	O-	O(CMP2_SYNCIN)	O
CMP3	O	O	O-	O(CMP3_SYNCIN)	O
CMP4	O	O	O	O(CMP4_SYNCIN)	O
CMP5	O	O	-	-	O

注：O 表示有此功能；-表示没有。

18.2.1 比较器功能

处理器中内嵌 6 个一致的模拟比较器，其数字输出和模拟输入的关系如下图所示。当 VIN+的电压低于 VIN-的时候，比较器的数字输出为低电平。反之，则为高电平。比较器 0 和比较器 1 的数字处理部分一致，比较器 2、比较器 3、比较器 4 和比较器 5 的数字处理部分，增加了比较器输出可由特定事件触发捕获的功能。

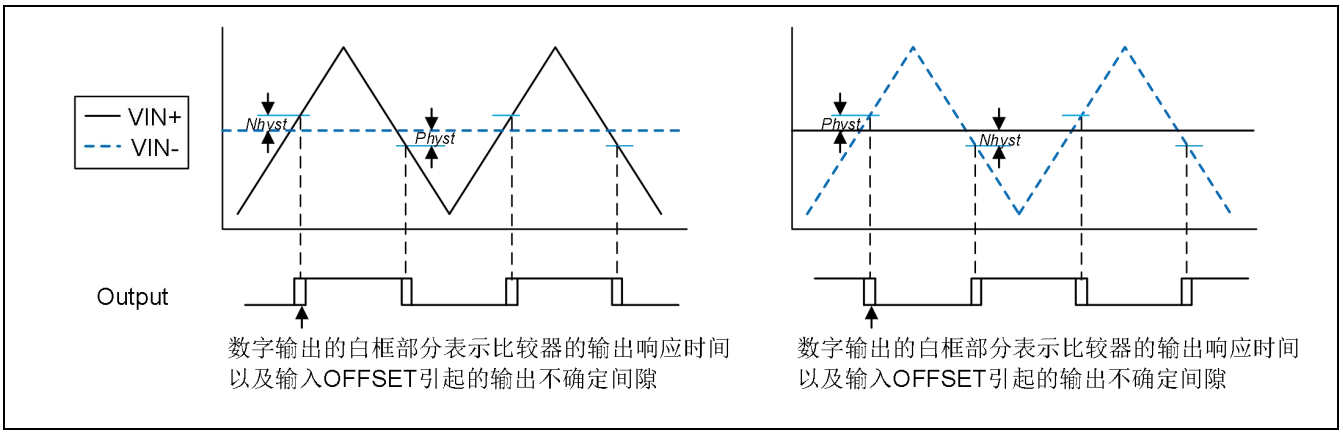


Figure 18-1 单一比较器示意图

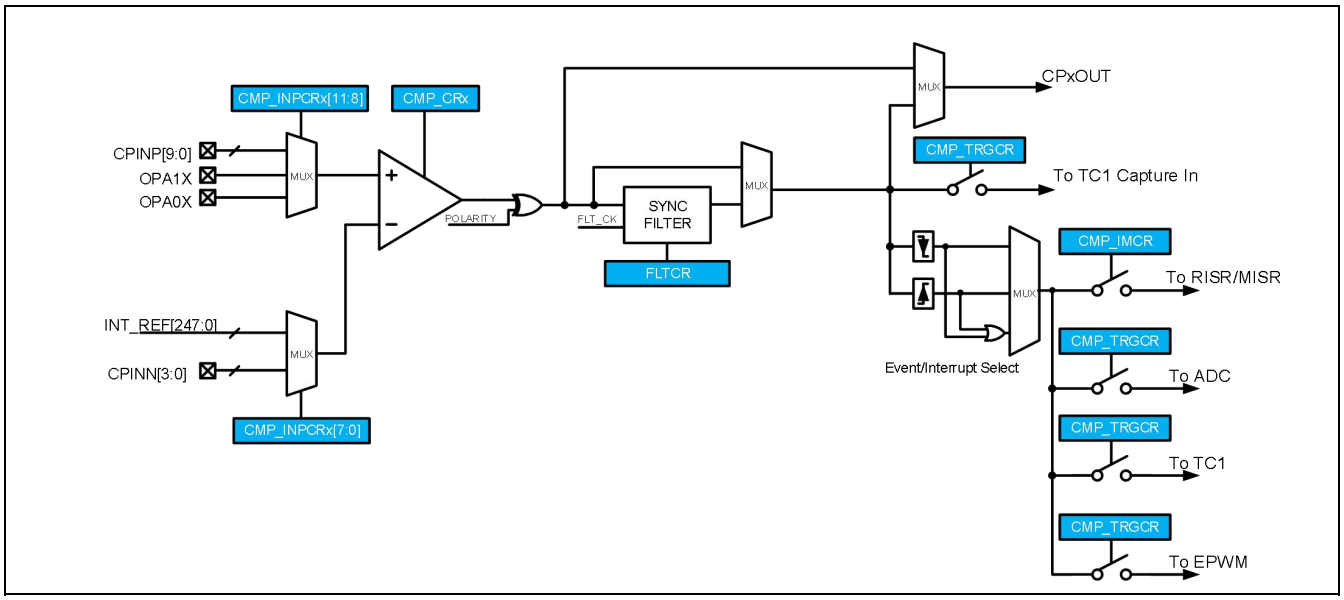


Figure 18-2 CMP0、CMP1比较器结构示意图

NOTE:

1) 所有比较器的模拟输入通道都是共享的，每个通道不是单一对应某一个比较器的模拟输入端。
2) 每个比较器的内部参考电压可以分别独立设置为126个内部参考点中的任意一个。
3) 比较器的控制时钟为比较器模块 PCLK 的时钟频率。

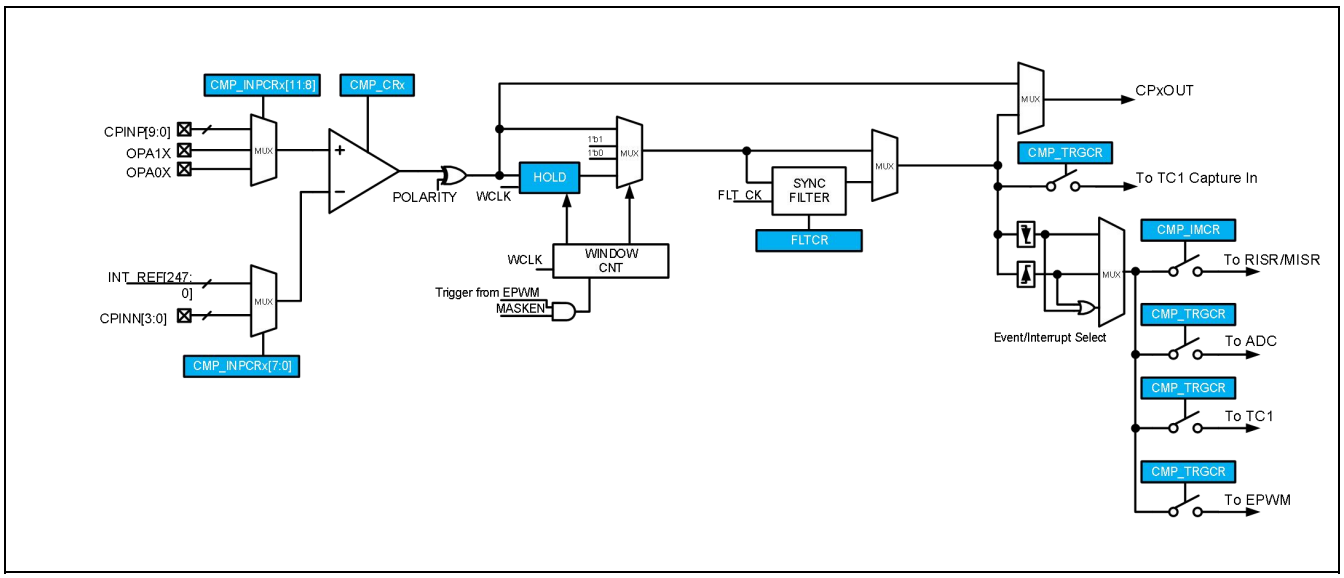


Figure 18-3 CMP2、CMP3、CMP4比较器结构示意图

NOTE:

1) 所有比较器的模拟输入通道都是共享的，每个通道不是单一对应某一个比较器的模拟输入端。
2) 每个比较器的内部参考电压可以分别独立设置为126个内部参考点中的任意一个。
3) 比较器的控制时钟为比较器模块 PCLK 的时钟频率。

18.2.2 比较器控制

每一个模拟比较器都有独立的控制寄存器来进行模式配置：CMP_CRx。控制寄存器中，可以设置比较器的使能，输出极性，响应速度以及相关的输入输出特性。对于比较器的输入模拟通道的选择，可以通过相对应的通道选择控制寄存器进行设置：CMP_INPCRx。比较器有两个模拟输入端，正向和负向输入端。正向模拟输入端，可以支持外部模拟信号的直接输入或者从模拟运算放大器的输出端得到模拟输入。负向输入端，可以支持外部模拟信号的直接输入或者从内部的126个电压参考源中选择任意一个作为输入。每个模拟比较器支持比较结果直接输出到外部管脚。

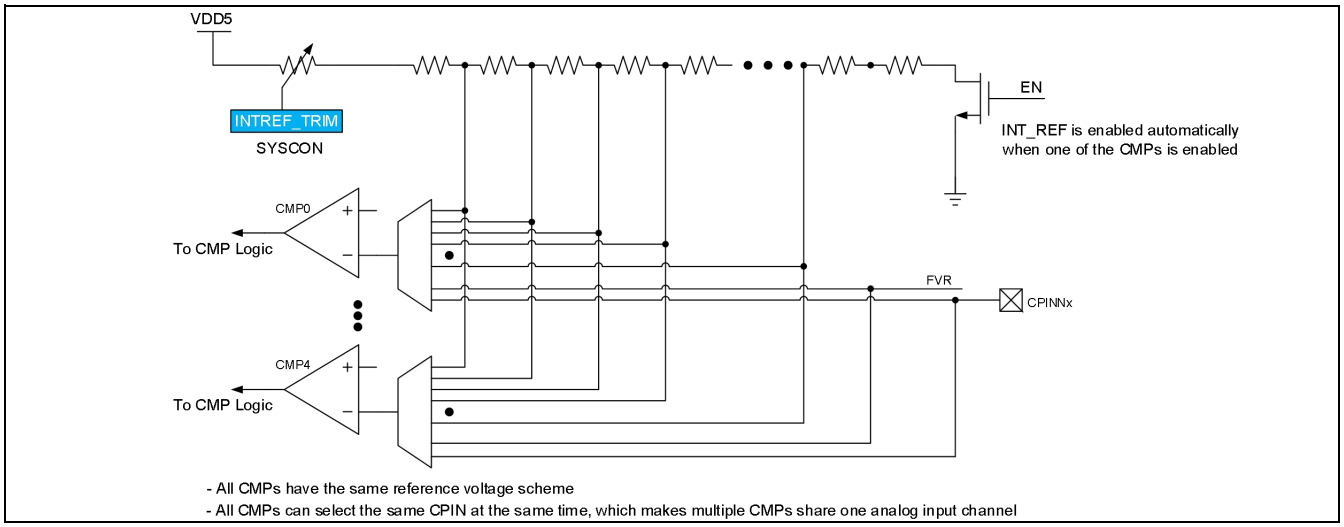


Figure 18-4 比较器内部参考电压

比较器负向输入支持从内部的电压参考源接入。内部参考电压源提供126个电压可供选择，每个模拟比较器可以通过通道设置控制寄存器独立选择其中一个作为输入。

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

Table 18-3 比较器输出状态和输入关系

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

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

18.2.3 比较器响应时间

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

18.2.4 比较器输出数字滤波

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

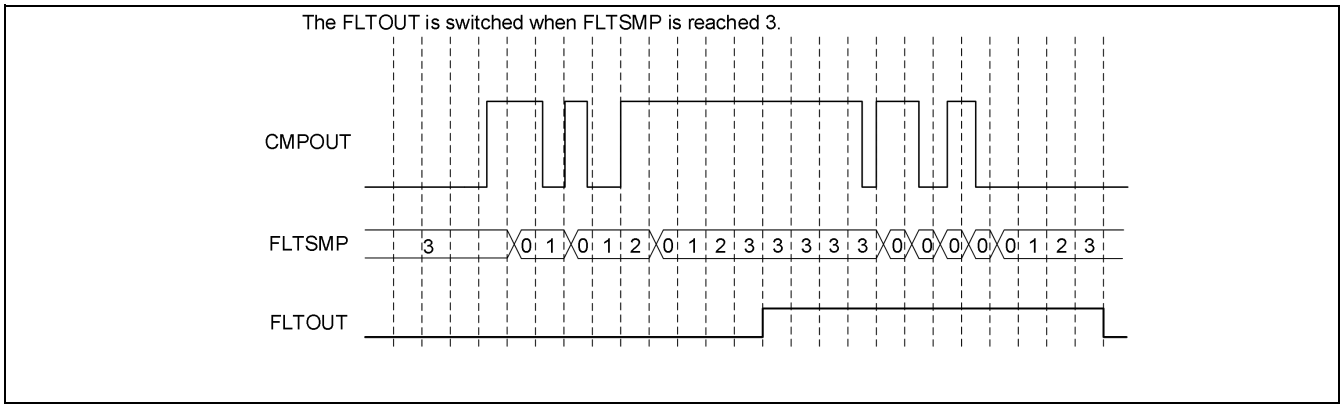


Figure 18-5 数字滤波机制

数字输出滤波采用数字去抖算法，滤波器在每个采样周期结束时对输入值进行采样。当输入电压从低电平到高电平切换时，在连续三次采样结果均为高电平的情况下，输出会确认从低电平切换到高电平。反之，当输入电压从高电平切换到低电平时，在连续三次采样结果均为低电平的情况下，滤波输出会确认从高电平切换到低电平。在三次连续采样中，如果出现一次采样结果和前一次采样不一致时，滤波计数将会被清除，并重新开始三次采样。滤波器的采样周期通过 FLTCR 中的分频系数进行设置。

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

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

18.2.5 比较器捕获滤波器

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

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

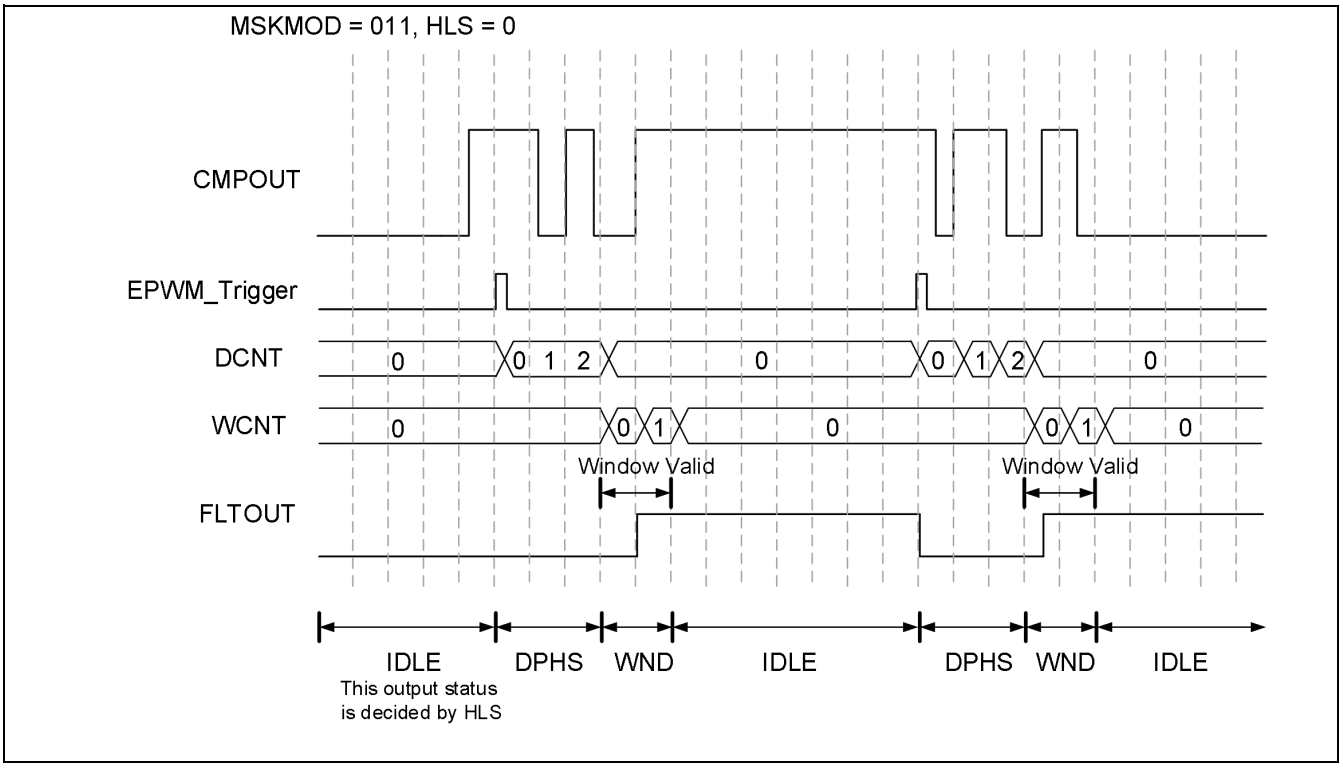


Figure 18-6 捕获滤波机制

18.2.6 比较器中断

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

比较器模块一共占用6个CPU的中断源，每个比较器都可触发对应独立的中断。如下图所示：

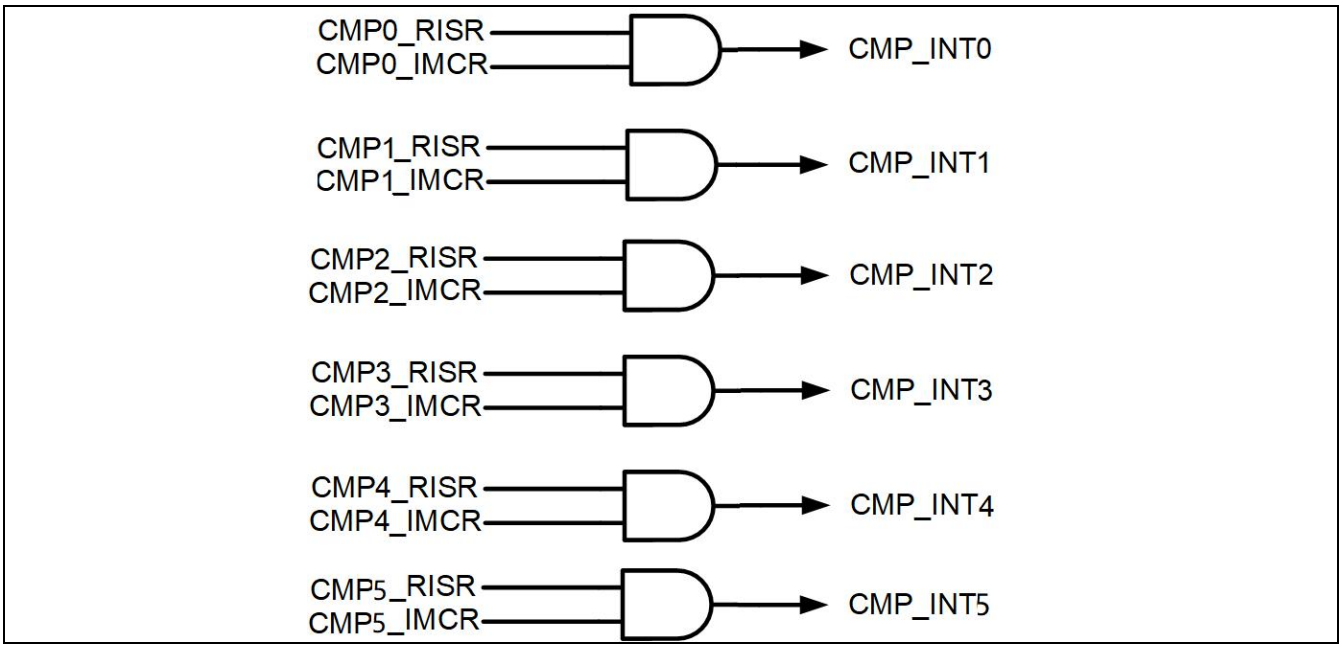


Figure 18-7 中断分配

18.2.7 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。CMP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

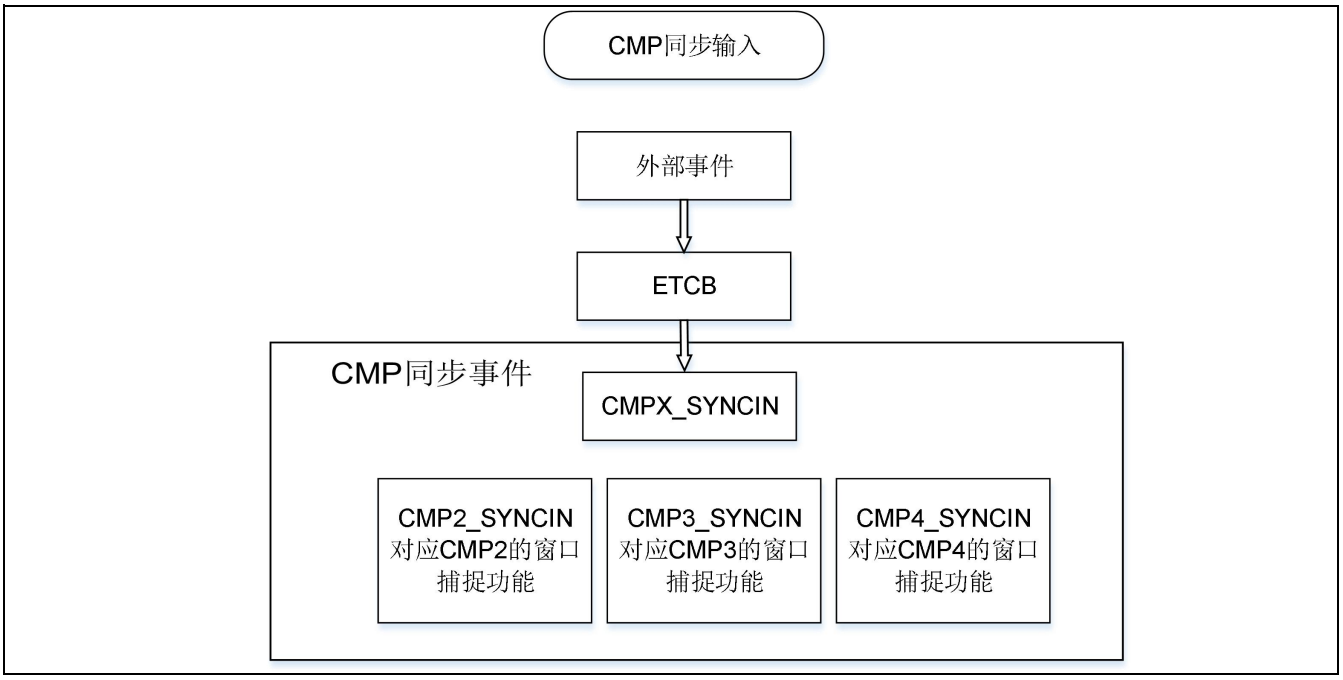


Figure 18-8 同步触发

CMP支持模块间的同步触发功能，可以触发启动比较器窗口捕获功能：

启动比较器窗口捕获功能(CMP2_SYNCIN ~ CMP4_SYNCIN)

- 捕获滤波器触发信号来自EPWM或者ETCB模块

18.2.8 事件及触发（输出）

CMP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6路事件触发输出，每个CMP中断对应一个事件输出端口。可以通过EVTRG控制寄存器选择CMP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。

CMP触发输出包含以下3种：

- CMP输出触发ADC采集。
- CMP输出作为TC1的Capture/CLK输入。
- CMP的输出作为TC1的启动触发输入。

同时CMP触发，在CMP模块的TRGCR中，选择具体的触发源，通过ETCB联动的方式，触发其他的目标事件。

详见下图:

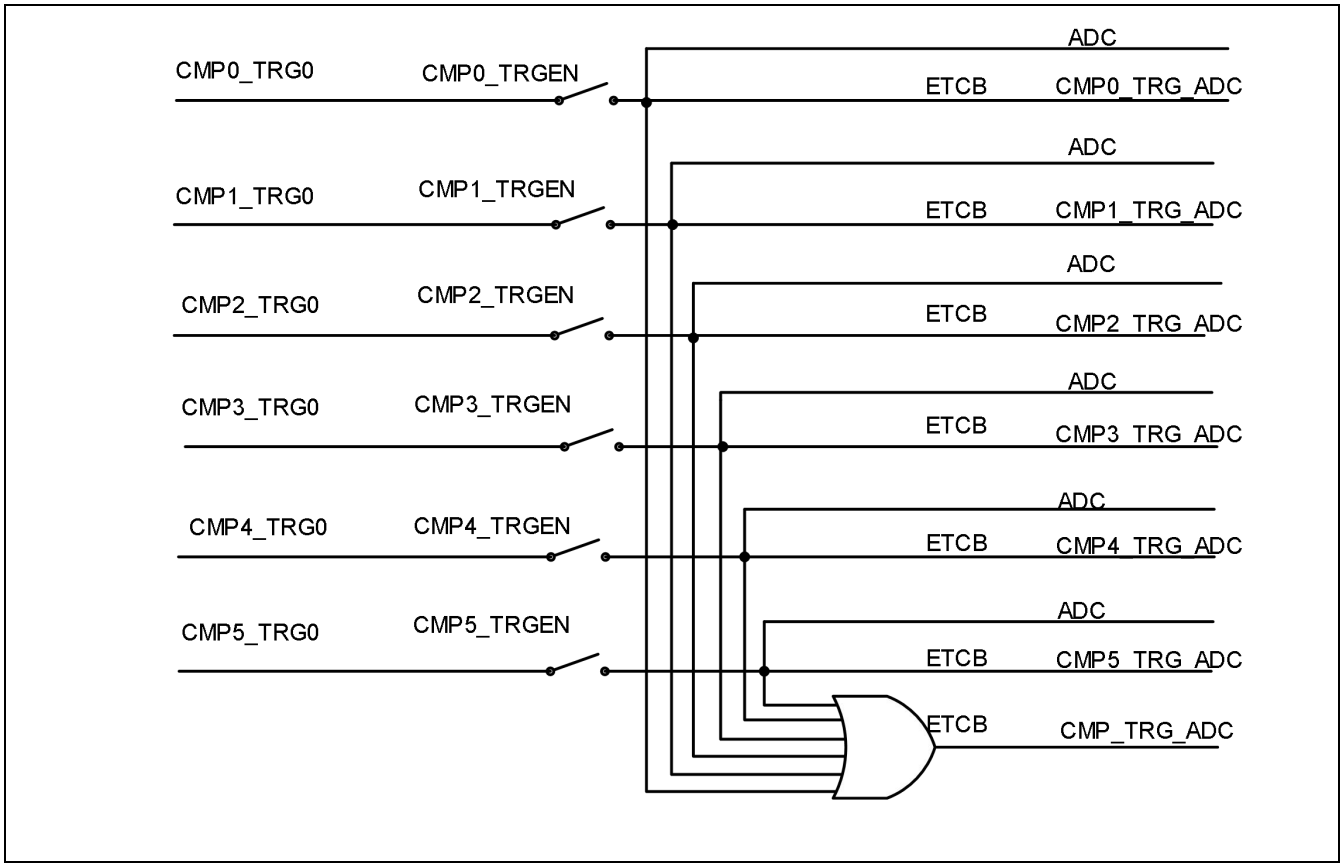


Figure 18-9 CMP 触发 ADC 及产生相关 ETCB 触发信号

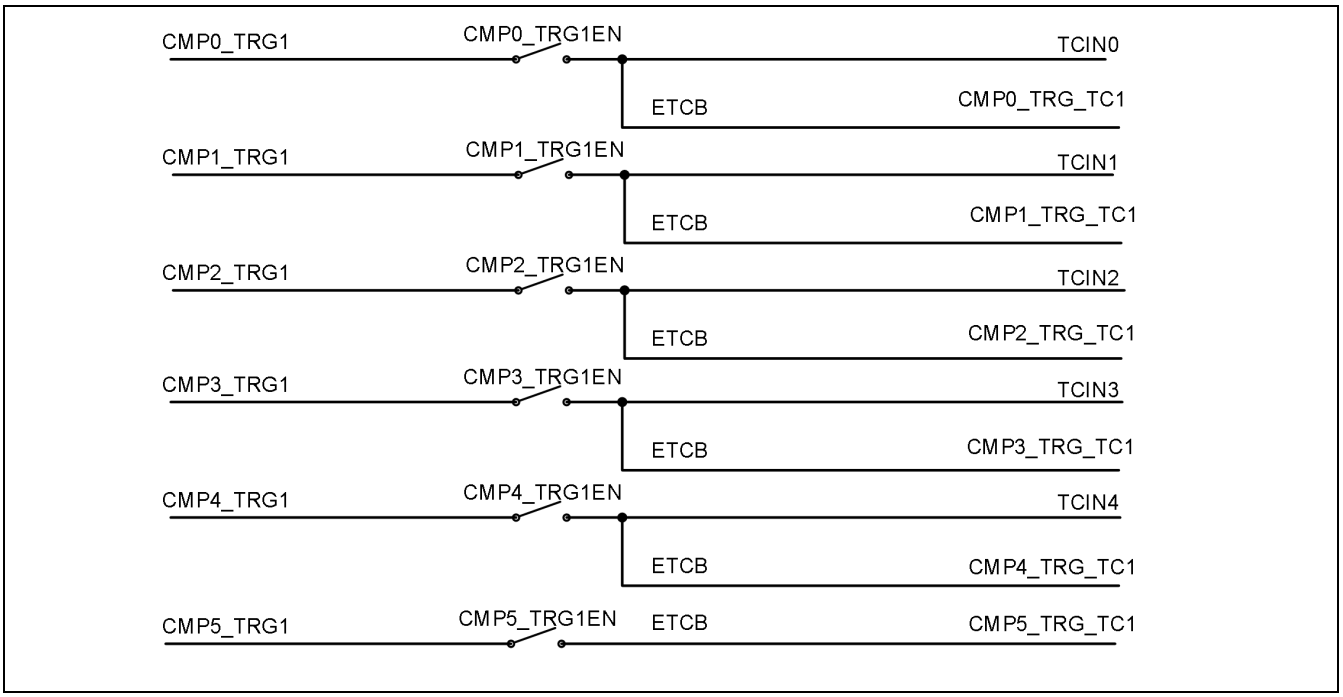


Figure 18-10 CMP 触发 TC1的 Capture/CLK 输入及产生相关 ETCB 触发信号

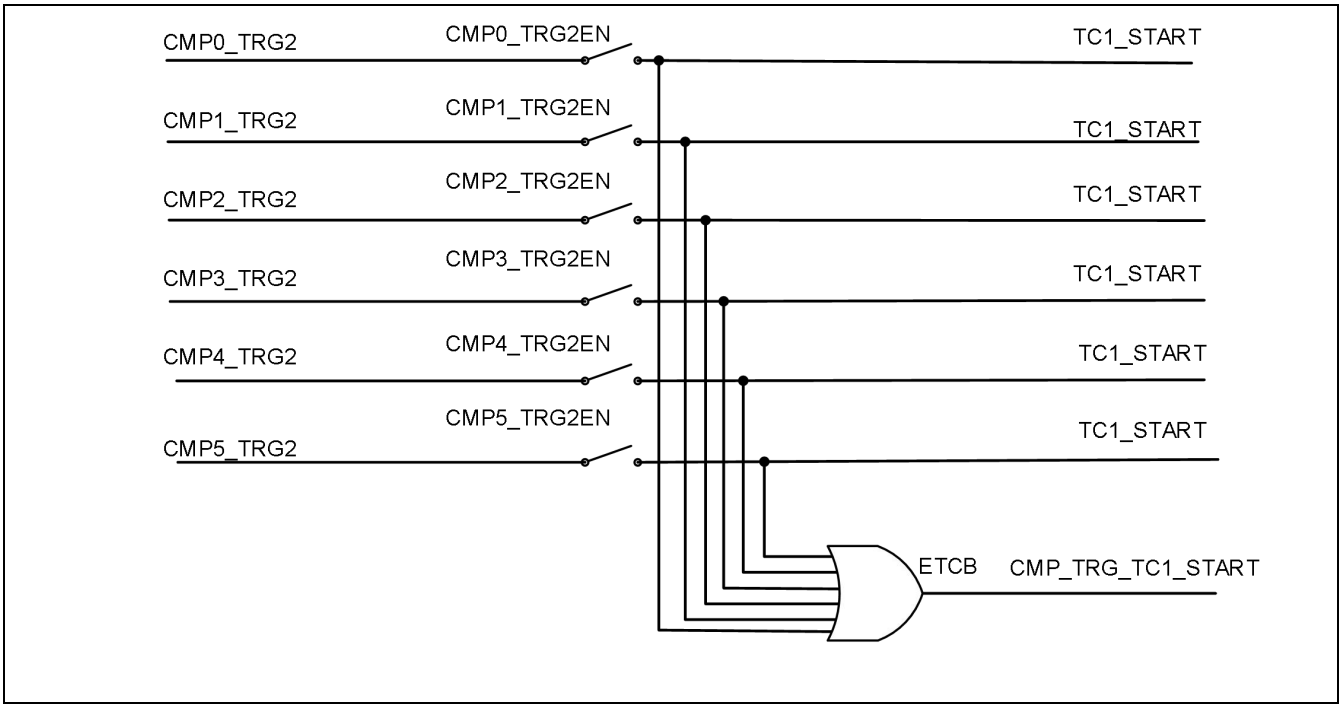


Figure 18-11 CMP 触发 TC1的启动触发及产生相关 ETCB 触发信号

18.3 寄存器说明

18.3.1 寄存器表

Base Address of CMP: 0x400B0000

Register	Offset	Description	Reset Value
CMP_CEDR	0x00	ID和时钟使能控制寄存器	0x00000000
CMP_CR0	0x04	比较器0的控制寄存器	0x00000000
CMP_CR1	0x08	比较器1的控制寄存器	0x00000000
CMP_CR2	0x0C	比较器2的控制寄存器	0x00000000
CMP_CR3	0x10	比较器3的控制寄存器	0x00000000
CMP_CR4	0x14	比较器4的控制寄存器	0x00000000
CMP_CR5	0x18	比较器5的控制寄存器	0x00000000
CMP_FLTCR0	0x1C	比较器0的数字滤波控制器	0x00000000
CMP_FLTCR1	0x20	比较器1的数字滤波控制器	0x00000000
CMP_FLTCR2	0x24	比较器2的数字滤波控制器	0x00000000
CMP_FLTCR3	0x28	比较器3的数字滤波控制器	0x00000000
CMP_FLTCR4	0x2C	比较器4的数字滤波控制器	0x00000000
CMP_FLTCR5	0x30	比较器5的数字滤波控制器	0x00000000
CMP_WCNT0	0x34	比较器2的捕捉窗口控制寄存器	0x00000000
CMP_WCNT1	0x38	比较器3的捕捉窗口控制寄存器	0x00000000
CMP_WCNT2	0x3C	比较器4的捕捉窗口控制寄存器	0x00000000
CMP_INPCR0	0x40	比较器0的输入控制寄存器	0x00000000
CMP_INPCR1	0x44	比较器1的输入控制寄存器	0x00000000
CMP_INPCR2	0x48	比较器2的输入控制寄存器	0x00000000
CMP_INPCR3	0x4C	比较器3的输入控制寄存器	0x00000000
CMP_INPCR4	0x50	比较器4的输入控制寄存器	0x00000000
CMP_INPCR5	0x54	比较器5的输入控制寄存器	0x00000000
CMP_TRGCR	0x58	比较器触发输出控制寄存器	0x00000000
CMP_IMCR	0x5C	中断使能禁止控制寄存器	0x00000000
CMP_RISR	0x60	原始中断状态寄存器	0x00000000
CMP_MISR	0x64	中断状态寄存器	0x00000000
CMP_ICR	0x68	中断标志清除寄存器	0x00000000
CMP_VOLSEL	0x6C	参考电压选择寄存器	0x00000000
NOTE:（1）工程模式预留			

18.3.2 CMP_CEDR(ID和时钟使能控制寄存器)

Address = Base Address+ 0x00, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	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																					RSVD			SWRST	RSVD	CLKENx					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
IDCODE/ID_KEY	[31:11]	R	ID Code寄存器。 这个区域保存了相应IP的IDCODE。
SWRST	[7]	W	软件复位。 0：没有效果 1：执行软件复位操作
CLKENx	[5:0]	RW	CMPx的时钟使能/禁止控制位。 0： 停止控制时钟 1： 使能控制时钟

18.3.3 CMP_CR0(比较器 0 的控制寄存器)

Address = Base Address+ 0x04, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPOUT5	CMPOUT4	CMPOUT3	CMPOUT2	CMPOUT1	CMPOUT0	RSVD										CPOS	RSVD				FLTEN	EVE_SEL		POLARITY	PHYST			NHYST			CMPEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	R	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CMPOUT5	[31]	R	比较器5输出状态
CMPOUT4	[30]	R	比较器4输出状态
CMPOUT3	[29]	R	比较器3输出状态
CMPOUT2	[28]	R	比较器2输出状态
CMPOUT1	[27]	R	比较器1输出状态
CMPOUT0	[26]	R	比较器0输出状态 当POLARITY=0时（非取反模式） 0：VIN+ < VIN- 1：VIN+ > VIN- 当POLARITY=1时（取反模式） 0：VIN+ > VIN- 1：VIN+ < VIN-
CPOS	[15]	RW	比较器输出管脚输出选择 0：比较器输出状态 1：比较器滤波后输出状态
FLTEN	[10]	RW	输出数字滤波计数器设置。 0：跳过滤波器 1：滤波器使能
EVE_SEL	[9:8]	RW	事件触发边沿选择。 0：下降沿 1：上升沿 2：下降沿和上升沿 3：下降沿和上升沿
POLARITY	[7]	RW	比较器输出极性选择. 0：输出不反向 1：输出反向

			注意： 只有在CMPEN为‘0’时，才可以改变比较器的极性。当比较器未使能时，对CMPEN的写1操作和极性配置可以同时进行。
PHYST	[6:4]	RW	比较器正向输入迟滞。 0：0mV 1：10mV 2：20mV 3：35mV 4：45mV 5：60mV 6：80mV 7：100mV
NHYST	[3:1]	RW	比较器负向输入迟滞。 0：0mV 1：10mV 2：20mV 3：35mV 4：45mV 5：60mV 6：80mV 7：100mV
CMPEN	[0]	RW	使能/禁止模拟比较器。 0：禁止模拟比较器 1：使能模拟比较器

18.3.4 CMP_CR1(比较器 1 的控制寄存器)

Address = Base Address+ 0x08, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPOUT5	CMPOUT4	CMPOUT3	CMPOUT2	CMPOUT1	CMPOUT0	RSVD										CPOS	RSVD				FLTEN	EVE_SEL		POLARITY	PHYST			NHYST			CMPEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	R	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CMPOUT5	[31]	R	比较器5输出状态
CMPOUT4	[30]	R	比较器4输出状态
CMPOUT3	[29]	R	比较器3输出状态
CMPOUT2	[28]	R	比较器2输出状态
CMPOUT1	[27]	R	比较器1输出状态
CMPOUT0	[26]	R	比较器0输出状态 当POLARITY=0时（非取反模式） 0：VIN+ < VIN- 1：VIN+ > VIN- 当POLARITY=1时（取反模式） 0：VIN+ > VIN- 1：VIN+ < VIN-
CPOS	[15]	RW	比较器输出管脚输出选择 0：比较器输出状态 1：比较器滤波后输出状态
FLTEN	[10]	RW	输出数字滤波计数器设置。 0：跳过滤波器 1：滤波器使能
EVE_SEL	[9:8]	RW	事件触发边沿选择。 0：下降沿 1：上升沿 2：下降沿和上升沿 3：下降沿和上升沿
POLARITY	[7]	RW	比较器输出极性选择. 0：输出不反向 1：输出反向

			注意： 只有在CMPEN为‘0’时，才可以改变比较器的极性。当比较器未使能时，对CMPEN的写1操作和极性配置可以同时进行。
PHYST	[6:4]	RW	比较器正向输入迟滞。 0：0mV 1：10mV 2：20mV 3：35mV 4：45mV 5：60mV 6：80mV 7：100mV
NHYST	[3:1]	RW	比较器负向输入迟滞。 0：0mV 1：10mV 2：20mV 3：35mV 4：45mV 5：60mV 6：80mV 7：100mV
CMPEN	[0]	RW	使能/禁止模拟比较器。 0：禁止模拟比较器 1：使能模拟比较器

18.3.5 CMP_CR2(比较器 2 的控制寄存器)

Address = Base Address+ 0x0C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPOUT5	CMPOUT4	CMPOUT3	CMPOUT2	CMPOUT1	CMPOUT0	RSVD										CPOS	RSVD				FLTEN	EVE_SEL		POLARITY	PHYST			NHYST			CMPEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	R	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CMPOUT5	[31]	R	比较器5输出状态
CMPOUT4	[30]	R	比较器4输出状态
CMPOUT3	[29]	R	比较器3输出状态
CMPOUT2	[28]	R	比较器2输出状态
CMPOUT1	[27]	R	比较器1输出状态
CMPOUT0	[26]	R	比较器0输出状态 当POLARITY=0时（非取反模式） 0：VIN+ < VIN- 1：VIN+ > VIN- 当POLARITY=1时（取反模式） 0：VIN+ > VIN- 1：VIN+ < VIN-
CPOS	[15]	RW	比较器输出管脚输出选择 0：比较器输出状态 1：比较器滤波后输出状态
FLTEN	[10]	RW	输出数字滤波计数器设置。 0：跳过滤波器 1：滤波器使能
EVE_SEL	[9:8]	RW	事件触发边沿选择。 0：下降沿 1：上升沿 2：下降沿和上升沿 3：下降沿和上升沿
POLARITY	[7]	RW	比较器输出极性选择. 0：输出不反向 1：输出反向

			注意： 只有在CMPEN为‘0’时，才可以改变比较器的极性。当比较器未使能时，对CMPEN的写1操作和极性配置可以同时进行。
PHYST	[6:4]	RW	比较器正向输入迟滞。 0：0mV 1：10mV 2：20mV 3：35mV 4：45mV 5：60mV 6：80mV 7：100mV
NHYST	[3:1]	RW	比较器负向输入迟滞。 0：0mV 1：10mV 2：20mV 3：35mV 4：45mV 5：60mV 6：80mV 7：100mV
CMPEN	[0]	RW	使能/禁止模拟比较器。 0：禁止模拟比较器 1：使能模拟比较器

18.3.6 CMP_CR3(比较器 3 的控制寄存器)

Address = Base Address+ 0x10, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPOUT5	CMPOUT4	CMPOUT3	CMPOUT2	CMPOUT1	CMPOUT0	RSVD										CPOS	RSVD				FLTEN	EVE_SEL		POLARITY	PHYST			NHYST			CMPEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	R	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CMPOUT5	[31]	R	比较器5输出状态
CMPOUT4	[30]	R	比较器4输出状态
CMPOUT3	[29]	R	比较器3输出状态
CMPOUT2	[28]	R	比较器2输出状态
CMPOUT1	[27]	R	比较器1输出状态
CMPOUT0	[26]	R	比较器0输出状态 当POLARITY=0时（非取反模式） 0：VIN+ < VIN- 1：VIN+ > VIN- 当POLARITY=1时（取反模式） 0：VIN+ > VIN- 1：VIN+ < VIN-
CPOS	[15]	RW	比较器输出管脚输出选择 0：比较器输出状态 1：比较器滤波后输出状态
FLTEN	[10]	RW	输出数字滤波计数器设置。 0：跳过滤波器 1：滤波器使能
EVE_SEL	[9:8]	RW	事件触发边沿选择。 0：下降沿 1：上升沿 2：下降沿和上升沿 3：下降沿和上升沿
POLARITY	[7]	RW	比较器输出极性选择. 0：输出不反向 1：输出反向

			注意： 只有在CMPEN为‘0’时，才可以改变比较器的极性。当比较器未使能时，对CMPEN的写1操作和极性配置可以同时进行。
PHYST	[6:4]	RW	比较器正向输入迟滞。 0：0mV 1：10mV 2：20mV 3：35mV 4：45mV 5：60mV 6：80mV 7：100mV
NHYST	[3:1]	RW	比较器负向输入迟滞。 0：0mV 1：10mV 2：20mV 3：35mV 4：45mV 5：60mV 6：80mV 7：100mV
CMPEN	[0]	RW	使能/禁止模拟比较器。 0：禁止模拟比较器 1：使能模拟比较器

18.3.7 CMP_CR4(比较器 4 的控制寄存器)

Address = Base Address+ 0x14, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPOUT5	CMPOUT4	CMPOUT3	CMPOUT2	CMPOUT1	CMPOUT0	RSVD										CPOS	RSVD				FLTEN	EVE_SEL		POLARITY	PHYST			NHYST			CMPEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	R	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CMPOUT5	[31]	R	比较器5输出状态
CMPOUT4	[30]	R	比较器4输出状态
CMPOUT3	[29]	R	比较器3输出状态
CMPOUT2	[28]	R	比较器2输出状态
CMPOUT1	[27]	R	比较器1输出状态
CMPOUT0	[26]	R	比较器0输出状态 当POLARITY=0时（非取反模式） 0：VIN+ < VIN- 1：VIN+ > VIN- 当POLARITY=1时（取反模式） 0：VIN+ > VIN- 1：VIN+ < VIN-
CPOS	[15]	RW	比较器输出管脚输出选择 0：比较器输出状态 1：比较器滤波后输出状态
FLTEN	[10]	RW	输出数字滤波计数器设置。 0：跳过滤波器 1：滤波器使能
EVE_SEL	[9:8]	RW	事件触发边沿选择。 0：下降沿 1：上升沿 2：下降沿和上升沿 3：下降沿和上升沿
POLARITY	[7]	RW	比较器输出极性选择. 0：输出不反向 1：输出反向

			注意： 只有在CMPEN为‘0’时，才可以改变比较器的极性。当比较器未使能时，对CMPEN的写1操作和极性配置可以同时进行。
PHYST	[6:4]	RW	比较器正向输入迟滞。 0：0mV 1：10mV 2：20mV 3：35mV 4：45mV 5：60mV 6：80mV 7：100mV
NHYST	[3:1]	RW	比较器负向输入迟滞。 0：0mV 1：10mV 2：20mV 3：35mV 4：45mV 5：60mV 6：80mV 7：100mV
CMPEN	[0]	RW	使能/禁止模拟比较器。 0：禁止模拟比较器 1：使能模拟比较器

18.3.8 CMP_CR5(比较器 5 的控制寄存器)

Address = Base Address+ 0x18, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPOUT5	CMPOUT4	CMPOUT3	CMPOUT2	CMPOUT1	CMPOUT0	RSVD										CPOS	RSVD				FLTEN	EVE_SEL		POLARITY	PHYST			NHYST			CMPEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	R	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CMPOUT5	[31]	R	比较器5输出状态
CMPOUT4	[30]	R	比较器4输出状态
CMPOUT3	[29]	R	比较器3输出状态
CMPOUT2	[28]	R	比较器2输出状态
CMPOUT1	[27]	R	比较器1输出状态
CMPOUT0	[26]	R	比较器0输出状态 当POLARITY=0时（非取反模式） 0：VIN+ < VIN- 1：VIN+ > VIN- 当POLARITY=1时（取反模式） 0：VIN+ > VIN- 1：VIN+ < VIN-
CPOS	[15]	RW	比较器输出管脚输出选择 0：比较器输出状态 1：比较器滤波后输出状态
FLTEN	[10]	RW	输出数字滤波计数器设置。 0：跳过滤波器 1：滤波器使能
EVE_SEL	[9:8]	RW	事件触发边沿选择。 0：下降沿 1：上升沿 2：下降沿和上升沿 3：下降沿和上升沿
POLARITY	[7]	RW	比较器输出极性选择. 0：输出不反向 1：输出反向

			注意： 只有在CMPEN为‘0’时，才可以改变比较器的极性。当比较器未使能时，对CMPEN的写1操作和极性配置可以同时进行。
PHYST	[6:4]	RW	比较器正向输入迟滞。 0：0mV 1：10mV 2：20mV 3：35mV 4：45mV 5：60mV 6：80mV 7：100mV
NHYST	[3:1]	RW	比较器负向输入迟滞。 0：0mV 1：10mV 2：20mV 3：35mV 4：45mV 5：60mV 6：80mV 7：100mV
CMPEN	[0]	RW	使能/禁止模拟比较器。 0：禁止模拟比较器 1：使能模拟比较器

18.3.9 CMP_FLTCR0(比较器 0 的数字滤波控制器)

Address = Base Address+ 0x1C, Reset Value = 0x00000000

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

Name	Bit	Type	Description
DIVM	[15:8]	RW	数字滤波器时钟分频系数M 注意： NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/(2的N次方)
DIVN	[7:3]	RW	数字滤波器时钟分频系数N 注意： NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/(2的N次方)
CKSRC	[2:0]	RW	数字滤波器时钟源选择。 0：PCLK 1：TC1 PEND触发 2：TC2 PEND触发 其他：无效
NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/2N,滤波固定深度为3.			

18.3.10 CMP_FLTCR1(比较器 1 的数字滤波控制器)

Address = Base Address+ 0x20, Reset Value = 0x00000000

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

Name	Bit	Type	Description
DIVM	[15:8]	RW	数字滤波器时钟分频系数M 注意： NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/(2的N次方)
DIVN	[7:3]	RW	数字滤波器时钟分频系数N 注意： NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/(2的N次方)
CKSRC	[2:0]	RW	数字滤波器时钟源选择。 0：PCLK 1：TC1 PEND触发 2：TC2 PEND触发 其他：无效
NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/2N,滤波固定深度为3.			

18.3.11 CMP_FLTCR2(比较器 2 的数字滤波控制器)

Address = Base Address+ 0x24, Reset Value = 0x00000000

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

Name	Bit	Type	Description
DIVM	[15:8]	RW	数字滤波器时钟分频系数M 注意： NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/(2的N次方)
DIVN	[7:3]	RW	数字滤波器时钟分频系数N 注意： NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/(2的N次方)
CKSRC	[2:0]	RW	数字滤波器时钟源选择。 0：PCLK 1：TC1 PEND触发 2：TC2 PEND触发 其他：无效
NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/2N,滤波固定深度为3.			

18.3.12 CMP_FLTCR3(比较器 3 的数字滤波控制器)

Address = Base Address+ 0x28, Reset Value = 0x00000000

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

Name	Bit	Type	Description
DIVM	[15:8]	RW	数字滤波器时钟分频系数M 注意： NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/(2的N次方)
DIVN	[7:3]	RW	数字滤波器时钟分频系数N 注意： NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/(2的N次方)
CKSRC	[2:0]	RW	数字滤波器时钟源选择。 0：PCLK 1：TC1 PEND触发 2：TC2 PEND触发 其他：无效
NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/2N,滤波固定深度为3.			

18.3.13 CMP_FLTCR4(比较器 4 的数字滤波控制器)

Address = Base Address+ 0x2C, Reset Value = 0x00000000

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

Name	Bit	Type	Description
DIVM	[15:8]	RW	数字滤波器时钟分频系数M 注意： NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/(2的N次方)
DIVN	[7:3]	RW	数字滤波器时钟分频系数N 注意： NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/(2的N次方)
CKSRC	[2:0]	RW	数字滤波器时钟源选择。 0：PCLK 1：TC1 PEND触发 2：TC2 PEND触发 其他：无效
NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/2N,滤波固定深度为3.			

18.3.14 CMP_FLTCR5(比较器 5 的数字滤波控制器)

Address = Base Address+ 0x30, Reset Value = 0x00000000

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

Name	Bit	Type	Description
DIVM	[15:8]	RW	数字滤波器时钟分频系数M 注意： NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/(2的N次方)
DIVN	[7:3]	RW	数字滤波器时钟分频系数N 注意： NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/(2的N次方)
CKSRC	[2:0]	RW	数字滤波器时钟源选择。 0：PCLK 1：TC1 PEND触发 2：TC2 PEND触发 其他：无效
NOTE: 数字滤波器的时钟FLT_CK = PCLK/(M+1)/2N,滤波固定深度为3.			

18.3.15 CMP_WCNT0(比较器 2 的捕捉窗口控制寄存器)

Address = Base Address+ 0x34, Reset Value = 0x00000000

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

Name	Bit	Type	Description
TRGSEL	[31:28]	RW	捕获滤波器触发信号选择（来自EPWM或者ETCB模块）。 0：PWM_START事件触发 1：PWM_STOP事件触发 2：PWM_PEND事件触发 3：PWM_CENTER事件触发 4：PWM0_CMPAUM事件触发 5：PWM0_CMPADM事件触发 6：PWM0_CMPBUM事件触发 7：PWM0_CMPBDM事件触发 8：PWM1_CMPAUM事件触发 9：PWM1_CMPADM事件触发 10：PWM1_CMPBUM事件触发 11：PWM1_CMPBDM事件触发 12：保留 13：保留 14：保留 15：ETCB SYNCIN触发
MSKMOD	[27:25]	RW	捕获滤波器控制 Value IDLE DPHS 000 Skip filter 001 LOW LOW 010 HIGH LOW 011 HOLD LOW 100 Skip filter 101 LOW HIGH 110 HIGH HIGH 111 HOLD HIGH IDLE：在窗口结束后，但未有被触发前的区间。该区间的初始极性电平

			可以通过WCNT中的HLS位进行设置。
HLS	[24]	RW	滤波器初次触发前的缺省输出相位。 0：低电平输出 1：高电平输出
DCNT	[23:16]	RW	设置捕捉窗口延时计数器。在触发后，内部8位递减计数器根据CLKDIV的设置频率计数。延时的时间为(DCNT+1) x Twcnt。 注意：当DCNT为零时，延时窗口被关闭，而不是1个Twcnt。
CLKDIV	[15:10]	RW	设置捕捉滤波器的频率Fwcnt。 Value DIV Value DIV 0 不分频 28 Div208 1 Div2 29 Div224 2 Div3 30 Div240 3 Div4 31 Div256 4 Div5 32 Div288 …… …… 33 Div320 15 Div16 34 Div352 16 Div24 35 Div384 17 Div32 36 Div416 18 Div40 37 Div448 19 Div48 38 Div480 20 Div56 39 Div512 21 Div64 40 Div640 22 Div72 41 Div720 23 Div128 42 Div1024 24 Div144 43 Div2048 25 Div160 Other 不分频 26 Div176 27 Div192 Fwcnt的分频是基于比较器模块PCLK时钟的分频。
WCNT	[9:0]	RW	设置捕捉窗口宽度计数器。在触发后，当延时计数器计数完成后，内部10位计数器根据CLKDIV的设置频率计数。 窗口的宽度为：(WCNT+1)x Twcnt

18.3.16 CMP_WCNT1(比较器 3 的捕捉窗口控制寄存器)

Address = Base Address+ 0x38, Reset Value = 0x00000000

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

Name	Bit	Type	Description
TRGSEL	[31:28]	RW	捕获滤波器触发信号选择（来自EPWM或者ETCB模块）。 0：PWM_START事件触发 1：PWM_STOP事件触发 2：PWM_PEND事件触发 3：PWM_CENTER事件触发 4：PWM0_CMPAUM事件触发 5：PWM0_CMPADM事件触发 6：PWM0_CMPBUM事件触发 7：PWM0_CMPBDM事件触发 8：PWM1_CMPAUM事件触发 9：PWM1_CMPADM事件触发 10：PWM1_CMPBUM事件触发 11：PWM1_CMPBDM事件触发 12：保留 13：保留 14：保留 15：ETCB SYNCIN触发
MSKMOD	[27:25]	RW	捕获滤波器控制 Value IDLE DPHS 000 Skip filter 001 LOW LOW 010 HIGH LOW 011 HOLD LOW 100 Skip filter 101 LOW HIGH 110 HIGH HIGH 111 HOLD HIGH IDLE：在窗口结束后，但未有被触发前的区间。该区间的初始极性电平

<table>
<tr><td></td><td></td><td></td><td>可以通过WCNT中的HLS位进行设置。</td></tr>
<tr><td>HLS</td><td>[24]</td><td>RW</td><td>滤波器初次触发前的缺省输出相位。
0：低电平输出
1：高电平输出</td></tr>
<tr><td>DCNT</td><td>[23:16]</td><td>RW</td><td>设置捕捉窗口延时计数器。在触发后，内部8位递减计数器根据CLKDIV的设置频率计数。延时的时间为(DCNT+1) x Twcnt。
注意：当DCNT为零时，延时窗口被关闭，而不是1个Twcnt。</td></tr>
<tr><td>CLKDIV</td><td>[15:10]</td><td>RW</td><td>设置捕捉滤波器的频率Fwcnt。
Value DIV Value DIV
0 不分频 28 Div208
1 Div2 29 Div224
2 Div3 30 Div240
3 Div4 31 Div256
4 Div5 32 Div288
…… …… 33 Div320
15 Div16 34 Div352
16 Div24 35 Div384
17 Div32 36 Div416
18 Div40 37 Div448
19 Div48 38 Div480
20 Div56 39 Div512
21 Div64 40 Div640
22 Div72 41 Div720
23 Div128 42 Div1024
24 Div144 43 Div2048
25 Div160 Other 不分频
26 Div176
27 Div192
Fwcnt的分频是基于比较器模块PCLK时钟的分频。</td></tr>
<tr><td>WCNT</td><td>[9:0]</td><td>RW</td><td>设置捕捉窗口宽度计数器。在触发后，当延时计数器计数完成后，内部10位计数器根据CLKDIV的设置频率计数。
窗口的宽度为：(WCNT+1)x Twcnt</td></tr>
</table>

18.3.17 CMP_WCNT2(比较器 4 的捕捉窗口控制寄存器)

Address = Base Address+ 0x3C, Reset Value = 0x00000000

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

Name	Bit	Type	Description
TRGSEL	[31:28]	RW	捕获滤波器触发信号选择（来自EPWM或者ETCB模块）。 0：PWM_START事件触发 1：PWM_STOP事件触发 2：PWM_PEND事件触发 3：PWM_CENTER事件触发 4：PWM0_CMPAUM事件触发 5：PWM0_CMPADM事件触发 6：PWM0_CMPBUM事件触发 7：PWM0_CMPBDM事件触发 8：PWM1_CMPAUM事件触发 9：PWM1_CMPADM事件触发 10：PWM1_CMPBUM事件触发 11：PWM1_CMPBDM事件触发 12：保留 13：保留 14：保留 15：ETCB SYNCIN触发
MSKMOD	[27:25]	RW	捕获滤波器控制 Value IDLE DPHS 000 Skip filter 001 LOW LOW 010 HIGH LOW 011 HOLD LOW 100 Skip filter 101 LOW HIGH 110 HIGH HIGH 111 HOLD HIGH IDLE：在窗口结束后，但未有被触发前的区间。该区间的初始极性电平

<table>
<tr><td></td><td></td><td></td><td>可以通过WCNT中的HLS位进行设置。</td></tr>
<tr><td>HLS</td><td>[24]</td><td>RW</td><td>滤波器初次触发前的缺省输出相位。
0：低电平输出
1：高电平输出</td></tr>
<tr><td>DCNT</td><td>[23:16]</td><td>RW</td><td>设置捕捉窗口延时计数器。在触发后，内部8位递减计数器根据CLKDIV的设置频率计数。延时的时间为(DCNT+1) x Twcnt。
注意：当DCNT为零时，延时窗口被关闭，而不是1个Twcnt。</td></tr>
<tr><td>CLKDIV</td><td>[15:10]</td><td>RW</td><td>设置捕捉滤波器的频率Fwcnt。
Value DIV Value DIV
0 不分频 28 Div208
1 Div2 29 Div224
2 Div3 30 Div240
3 Div4 31 Div256
4 Div5 32 Div288
…… …… 33 Div320
15 Div16 34 Div352
16 Div24 35 Div384
17 Div32 36 Div416
18 Div40 37 Div448
19 Div48 38 Div480
20 Div56 39 Div512
21 Div64 40 Div640
22 Div72 41 Div720
23 Div128 42 Div1024
24 Div144 43 Div2048
25 Div160 Other 不分频
26 Div176
27 Div192
Fwcnt的分频是基于比较器模块PCLK时钟的分频。</td></tr>
<tr><td>WCNT</td><td>[9:0]</td><td>RW</td><td>设置捕捉窗口宽度计数器。在触发后，当延时计数器计数完成后，内部10位计数器根据CLKDIV的设置频率计数。
窗口的宽度为：(WCNT+1)x Twcnt</td></tr>
</table>

18.3.18 CMP_INPCR0(比较器 0 的输入控制寄存器)

Address = Base Address+ 0x40, Reset Value = 0x00000000

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

Name	Bit	Type	Description
PSEL	[11:8]	RW	比较器正向输入选择。 0：CPINP0 1：CPINP1 …… 9：CPINP9 Others：Not used 13：OPA1X 14：OPA0X 15：RSVD（1）
NSEL	[3:0]	RW	比较器负向输入选择。 0：CPINN0 1：CPINN1 2：CPINN2 3：CPINN3 4：CPINN4 5：RSVD 6：GND 7：INT_REF = VOL_REF/126 * 1 8：INT_REF = VOL_REF/126 * 2 9：INT_REF = VOL_REF/126 * 3 10：INT_REF = VOL_REF/126 * 4 11：INT_REF = VOL_REF/126 * 5 12：INT_REF = VOL_REF/126 * 6 13：INT_REF = VOL_REF/126 * 7 14：FVR 15：RSVD
NOTE:（1）工程模式预留			

18.3.19 CMP_INPCR1(比较器1的输入控制寄存器)

Address = Base Address+ 0x44, Reset Value = 0x00000000

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

Name	Bit	Type	Description
PSEL	[11:8]	RW	比较器正向输入选择。 0：CPINP0 1：CPINP1 …… 9：CPINP9 Others：Not used 13：OPA1X 14：OPA0X 15：RSVD（1）
NSEL_MIN	[7:0]	RW	比较器负向输入选择。 0：GND 1：INT_REF = VOL_REF/126 * 1 2：INT_REF = VOL_REF/126 * 2 …… 126：INT_REF = VOL_REF/126 * 126 128：CPINN0 129：CPINN1 130：CPINN2 131：CPINN3 132：CPINN4 254：FVR Others：Not used
NOTE:（1）工程模式预留			

18.3.20 CMP_INPCR2(比较器 2 的输入控制寄存器)

Address = Base Address+ 0x48, Reset Value = 0x00000000

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

Name	Bit	Type	Description
PSEL	[11:8]	RW	比较器正向输入选择。 0：CPINP0 1：CPINP1 …… 9：CPINP9 Others：Not used 13：OPA1X 14：OPA0X 15：RSVD（1）
NSEL	[7:0]	RW	比较器负向输入选择。 0：GND 1：INT_REF = VOL_REF/126 * 1 2：INT_REF = VOL_REF/126 * 2 …… 126：INT_REF = VOL_REF/126 * 126 128：CPINN0 129：CPINN1 130：CPINN2 131：CPINN3 132：CPINN4 254：FVR Others：Not used
NOTE:（1）工程模式预留			

18.3.21 CMP_INPCR3(比较器 3 的输入控制寄存器)

Address = Base Address+ 0x4C, Reset Value = 0x00000000

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

Name	Bit	Type	Description
PSEL	[11:8]	RW	比较器正向输入选择。 0：CPINP0 1：CPINP1 …… 9：CPINP9 Others：Not used 13：OPA1X 14：OPA0X 15：RSVD（1）
NSEL	[7:0]	RW	比较器负向输入选择。 0：GND 1：INT_REF = VOL_REF/126 * 1 2：INT_REF = VOL_REF/126 * 2 …… 126：INT_REF = VOL_REF/126 * 126 128：CPINN0 129：CPINN1 130：CPINN2 131：CPINN3 132：CPINN4 254：FVR Others：Not used
NOTE:（1）工程模式预留			

18.3.22 CMP_INPCR4(比较器 4 的输入控制寄存器)

Address = Base Address+ 0x50, Reset Value = 0x00000000

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

Name	Bit	Type	Description
PSEL	[11:8]	RW	比较器正向输入选择。 0：CPINP0 1：CPINP1 …… 9：CPINP9 Others：Not used 13：OPA1X 14：OPA0X 15：RSVD（1）
NSEL	[7:0]	RW	比较器负向输入选择。 0：GND 1：INT_REF = VOL_REF/126 * 1 2：INT_REF = VOL_REF/126 * 2 …… 126：INT_REF = VOL_REF/126 * 126 128：CPINN0 129：CPINN1 130：CPINN2 131：CPINN3 132：CPINN4 254：FVR Others：Not used
NOTE:（1）工程模式预留			

18.3.23 CMP_INPCR5(比较器 5 的输入控制寄存器)

Address = Base Address+ 0x54, Reset Value = 0x00000000

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

Name	Bit	Type	Description
PSEL	[11:8]	RW	比较器正向输入选择。 0：CPINP0 1：CPINP1 …… 9：CPINP9 Others：Not used 13：OPA1X 14：OPA0X 15：RSVD（1）
NSEL	[7:0]	RW	比较器负向输入选择。 0：GND 1：INT_REF = VOL_REF/126 * 1 2：INT_REF = VOL_REF/126 * 2 …… 126：INT_REF = VOL_REF/126 * 126 128：CPINN0 129：CPINN1 130：CPINN2 131：CPINN3 132：CPINN4 254：FVR Others：Not used
NOTE:（1）工程模式预留			

18.3.24 CMP_TRGCR(比较器触发输出控制寄存器)

Address = Base Address+ 0x58, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																	CMPx_TRGEN							CMPx_TRG1EN						CMPx_TRG2EN		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
CMPx_TRG0EN	[14:9]	RW	CMPx的ADC转换硬件触发使能控制。 0：禁止CMP硬件触发ADC 1：使能CMP硬件触发ADC
CMPx_TRG1EN	[8:3]	RW	TCIN0: CMP0输出作为TC1的Capture/CLK输入。 TCIN1: CMP1输出作为TC1的Capture/CLK输入。 TCIN2: CMP2输出作为TC1的Capture/CLK输入。 TCIN3: CMP3输出作为TC1的Capture/CLK输入。 TCIN4: CMP4输出作为TC1的Capture/CLK输入。 0：不输出（保持低电平） 1：输出使能
CMPx_TRG2EN	[2:0]	RW	选择比较器的输出作为TC1的启动触发输入。 0：不输出 1：CMPOUT0作为TC1的启动触发 2：CMPOUT1作为TC1的启动触发 3：CMPOUT2作为TC1的启动触发 4：CMPOUT3作为TC1的启动触发 5：CMPOUT4作为TC1的启动触发 6：CMPOUT5作为TC1的启动触发 Others：不输出

18.3.25 CMP_IMCR(中断使能禁止控制寄存器)

Address = Base Address+ 0x5C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																										EDGEDET5	EDGEDET4	EDGEDET3	EDGEDET2	EDGEDET1	EDGEDET0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
EDGEDET5	[5]	RW	CMP5的输出边沿检测中断使能控制。 0：禁止中断发生 1：使能中断发生
EDGEDET4	[4]	RW	CMP4的输出边沿检测中断使能控制。 0：禁止中断发生 1：使能中断发生
EDGEDET3	[3]	RW	CMP3的输出边沿检测中断使能控制。 0：禁止中断发生 1：使能中断发生
EDGEDET2	[2]	RW	CMP2的输出边沿检测中断使能控制。 0：禁止中断发生 1：使能中断发生
EDGEDET1	[1]	RW	CMP1的输出边沿检测中断使能控制。 0：禁止中断发生 1：使能中断发生
EDGEDET0	[0]	RW	CMP0的输出边沿检测中断使能控制。 0：禁止中断发生 1：使能中断发生

18.3.26 CMP_RISR(原始中断状态寄存器)

Address = Base Address+ 0x60, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																										EDGEDET5	EDGEDET4	EDGEDET3	EDGEDET2	EDGEDET1	EDGEDET0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
EDGEDET5	[5]	R	CMP5的输出边沿检测中断原始状态位。 0：中断未发生 1：中断发生
EDGEDET4	[4]	R	CMP4的输出边沿检测中断原始状态位。 0：中断未发生 1：中断发生
EDGEDET3	[3]	R	CMP3的输出边沿检测中断原始状态位。 0：中断未发生 1：中断发生
EDGEDET2	[2]	R	CMP2的输出边沿检测中断原始状态位。 0：中断未发生 1：中断发生
EDGEDET1	[1]	R	CMP1的输出边沿检测中断原始状态位。 0：中断未发生 1：中断发生
EDGEDET0	[0]	R	CMP0的输出边沿检测中断原始状态位。 0：中断未发生 1：中断发生

18.3.27 CMP_MISR(中断状态寄存器)

Address = Base Address+ 0x64, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																										EDGEDET5	EDGEDET4	EDGEDET3	EDGEDET2	EDGEDET1	EDGEDET0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
EDGEDET5	[5]	R	CMP5的输出边沿检测中断状态位。 0：中断未发生 1：中断发生
EDGEDET4	[4]	R	CMP4的输出边沿检测中断状态位。 0：中断未发生 1：中断发生
EDGEDET3	[3]	R	CMP3的输出边沿检测中断状态位。 0：中断未发生 1：中断发生
EDGEDET2	[2]	R	CMP2的输出边沿检测中断状态位。 0：中断未发生 1：中断发生
EDGEDET1	[1]	R	CMP1的输出边沿检测中断状态位。 0：中断未发生 1：中断发生
EDGEDET0	[0]	R	CMP0的输出边沿检测中断状态位。 0：中断未发生 1：中断发生

18.3.28 CMP_ICR(中断标志清除寄存器)

Address = Base Address+ 0x68, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																										EDGEDET5	EDGEDET4	EDGEDET3	EDGEDET2	EDGEDET1	EDGEDET0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W

Name	Bit	Type	Description
EDGEDET5	[5]	W	CMP5的输出边沿检测中断状态清除位。（只有写入时有效） 0：无效果 1：清除CMP5中断标志位
EDGEDET4	[4]	W	CMP4的输出边沿检测中断状态清除位。（只有写入时有效） 0：无效果 1：清除CMP4中断标志位
EDGEDET3	[3]	W	CMP3的输出边沿检测中断状态清除位。（只有写入时有效） 0：无效果 1：清除CMP3中断标志位
EDGEDET2	[2]	W	CMP2的输出边沿检测中断状态清除位。（只有写入时有效） 0：无效果 1：清除CMP2中断标志位
EDGEDET1	[1]	W	CMP1的输出边沿检测中断状态清除位。（只有写入时有效） 0：无效果 1：清除CMP1中断标志位
EDGEDET0	[0]	W	CMP0的输出边沿检测中断状态清除位。（只有写入时有效） 0：无效果 1：清除CMP0中断标志位

18.3.29 CMP_VOLSEL(参考电压选择寄存器)

Address = Base Address+ 0x6C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																															VOLSEL
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
VOLSEL	[0]	RW	比较器参考电压源VOL_REF选择 0：0.7VDD 1：3.5V (芯片供电VDD必须大于4.0V)