APT32F104x 系列使用手册 V1.1

104x 系列通用型 32 位微处理控制器

简介

该使用手册面向应用开发人员，旨在给与 104x 列内核，存储及全部外围的完整信息。

相关文档

APT32F1041 数据手册

APT32F1042 数据手册

APT32F1043 数据手册

APT32F1044 数据手册

产品分类定义

Table 0-1 APT32F104x系列产品分类定义

	资源	1041	1042	1043	1044
异	PFlash+SRAM	48K PFlash +3K SRAM			48K PFlash +3K SRAM / 16K PFlash +2K SRAM
	Package	SOP28_2/ SOP20/ SOP16/	SOP28_3	SOP28_1	SOP28_4 SOP20 SOP16
	Pin	28/20/16 pin	28 pin		28/20/16 pin
	ADC	26ch/18ch/14ch	26ch		26ch/18ch/14ch
	LED	Max 9*11 or 4*16 Max 8*8 or 3*13 Max 7*6 or 2*11	Max 9*11 or 4*16		Max 9*11 or 4*16 Max 8*8 or 3*13 Max 7*6 or 2*11
	TOUCH	26ch/18ch/14ch	26ch		26ch/18ch/14ch
同	DFlash	2K			
	BT	4			
	TC3	1			
	GPTA	1			
	GPTB	1			
	WWDT	1			
	I2C	1			
	UART	2			
	SPI	1			

历史版本说明

版本	修改日期	修改概要
P0.0	2024-11-4	初版
V0.0	2024-11-14	1. 增加系列产品分类定义表 2. 更新最新的寄存器内容 3. 部分IP描述修改
V1.0	2025-2-27	1. 修改部分章节的格式 2. 部分IP描述修改 3. 部分IP更新最新的寄存器内容
V1.1	2025-6	1. 增加EXI滤波方式的描述 2. 更新客户信息区大小和地址 3. 更新紧急模式相关寄存器内容 4. 修改部分错误描述

1 系统存储空间

1.1 概述

本章节介绍了 APT32F104x 系列系统存储空间。

本章包含内容如下：

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

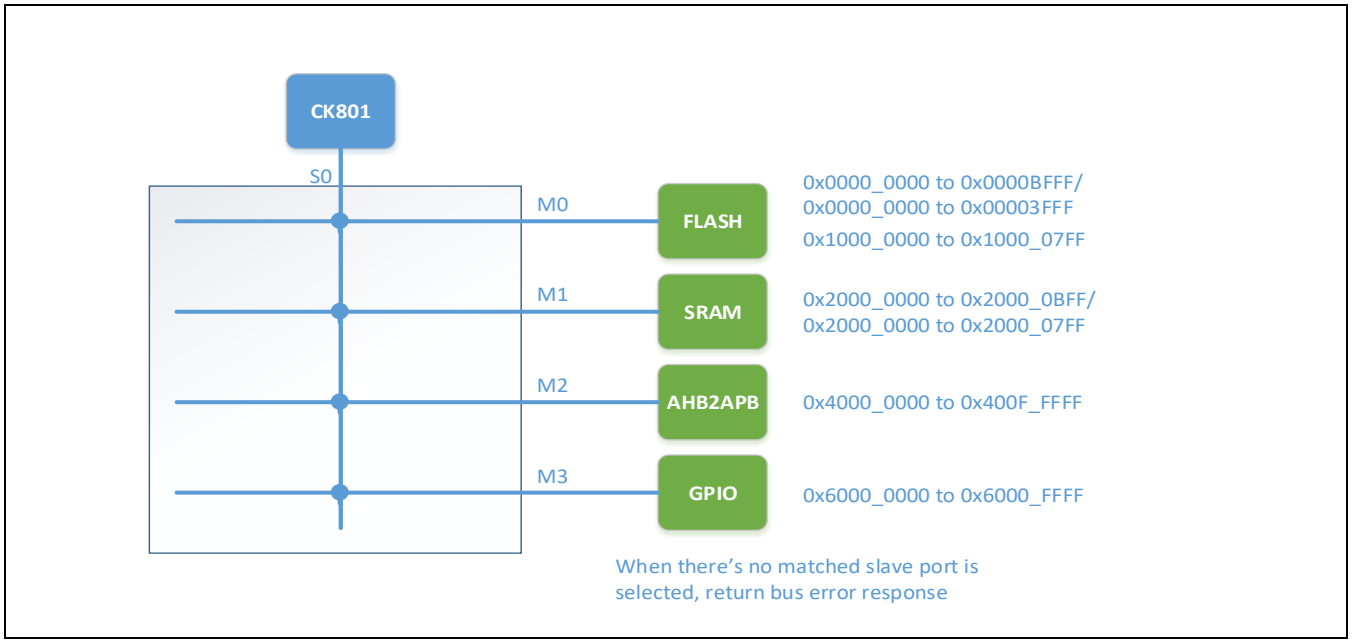


Figure 1-1 系统总线架构

1.2 默认存储地址表

Table 1-1 存储地址

Address	Memory
Reserved	Reserved
0xE000_0000 to 0xE00F_FFFF	CPU内部寄存器
Reserved	Reserved
Reserved	Reserved
Reserved	Reserved
0x6000_0000 to 0x6000_FFFF	GPIO控制器
Reserved	Reserved
Reserved	Reserved
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_0BFF/ 0x2000_0000 to 0x2000_07FF	SRAM
Reserved	Reserved
0x1000_0000 to 0x1000_07FF	数据闪存 (Data Flash)
Reserved	Reserved
0x0000_0000 to 0x0000_BFFF/ 0x0000_0000 to 0x0000_3FFF	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
GPIO	0x6000_F000	通用IO端口-GROUP
	0x6000_2000	通用IO端口-B (GPIO B)
	0x6000_0000	通用IO端口-A (GPIO A)
I2C	0x400A_0000	I2C串行接口 (I2C)
SPI	0x4009_0000	同步并行接口 (SPI)
UART	0x4008_1000	通用异步收发器1 (UART1)
	0x4008_0000	通用异步收发器0 (UART0)
LED	0x4007_0000	LED控制器 (LED)
WWDT	0x4006_2000	窗口型看门口定时器（WWDT）
TC	0x4005_9000	保留 (Reserved)
	0x4005_8000	保留 (Reserved)
	0x4005_7000	保留 (Reserved)
	0x4005_6000	可编程定时器/计数器B (GPTB)
	0x4005_5000	通用型可编程定时器/计数器A (GPTA)
	0x4005_4000	基本定时器 (BT3)
	0x4005_3000	基本定时器 (BT2)
	0x4005_2000	基本定时器 (BT1)
	0x4005_1000	基本定时器 (BT0)

	0x4005_0000	时钟定时器 (TC3)
ADC	0x4003_0000	模数转换器 (ADC)
TOUCH	0x4002_0000	电容式触摸按键传感器（TOUCH）
SYSTEM	0x4001_2000	事件触发控制器 (ETCB)
	0x4001_1000	系统控制器 (SYSCON)
	0x4001_0000	闪存控制器 (IFC)
	0x4000_0000	设备信息寄存器 (Device ID)

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

2 中断向量控制器(INTC)

2.1 概述

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

2.1.1 特性

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

2.2 中断向量表

Table 2-1 System Interrupt Vectors

Number	Address	Vector	Interrupt Sources
32/0	0x0000_0080	CORET	CK801 CPU Core Timer
33/1	0x0000_0084	SYSCON	System controller interrupt
34/2	0x0000_0088	IFC	Program flash controller interrupt
35/3	0x0000_008C	ADC	ADC Interrupt
36/4	0x0000_0090	-	Reserved
37/5	0x0000_0094	-	Reserved
38/6	0x0000_0098	-	Reserved
39/7	0x0000_009C	EXI0	External interrupt GROUP0, GROUP16
40/8	0x0000_00A0	EXI1	External interrupt GROUP1, GROUP17
41/9	0x0000_00A4	GPTA	GPTA Interrupt
42/10	0x0000_00A8	GPTB	GPTB Interrupt
43/11	0x0000_00AC	-	Reserved
44/12	0x0000_00B0	-	Reserved
45/13	0x0000_00B4	UART0	UART 0 interrupt
46/14	0x0000_00B8	UART1	UART 1 interrupt
47/15	0x0000_00BC	-	Reserved
48/16	0x0000_00C0	-	Reserved
49/17	0x0000_00C4	I2C	I2C interrupt
50/18	0x0000_00C8	-	Reserved
51/19	0x0000_00CC	SPI	SPI interrupt
52/20	0x0000_00D0	-	Reserved
53/21	0x0000_00D4	EXI2	External Interrupt GROUP2 ~ 3, GROUP18~19
54/22	0x0000_00D8	EXI3	External Interrupt GROUP4 ~ 9
55/23	0x0000_00DC	EXI4	External Interrupt GROUP10 ~ 15
56/24	0x0000_00E0	TC3	TC3 Interrupt
57/25	0x0000_00E4	TKEY	Touch Key interrupt
58/26	0x0000_00E8	WWDT	Window Watchdog Interrupt
59/27	0x0000_00EC	LED	LED Interrupt
60/28	0x0000_00F0	BT0	BT0 interrupt
61/29	0x0000_00F4	BT1	BT1 interrupt
62/30	0x0000_00F8	BT2	BT2 interrupt
63/31	0x0000_00FC	BT3	BT3 interrupt

中断向量号，是请求在异常表的位置编号。0~31用作处理器内部识别的向量；31号以后的向量号是留给软件的，用作指向系统描述符指针；从32号开始的向量是留给外设请求的。例如“32/0”这样的表示，说明CORET作为外设的第0号向量，实际对应处理器的32号向量。

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

2.3 工作原理

2.3.1 中断处理

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

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

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

中断的处理过程可以分以下几个步骤进行：

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

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

2.3.2 中断优先级和中断抢占

中断的优先级可以通过VIC_IPR0 ~7这8个寄存器来设置。每个VIC_IPR寄存器对应四个中断源的优先级设置。

中断的优先级共分为4级设置，通过VIC_IPR寄存器的PRI_x中的最高两位来设置。数值越小，代表的优先级越

高，所以设置为‘0’时代表最高优先级。如果优先级号相同，则根据中断源号来决定优先的顺序，号码越小，优先级越高。例如，IRQ0 和 IRQ1的优先级号设置为相同，当IRQ0和IRQ1同时提交中断，由于IRQ0的中断源号小于IRQ1，因此IRQ0先得到CPU的响应。

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

中断抢占的优先级条件可分为两种：

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

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

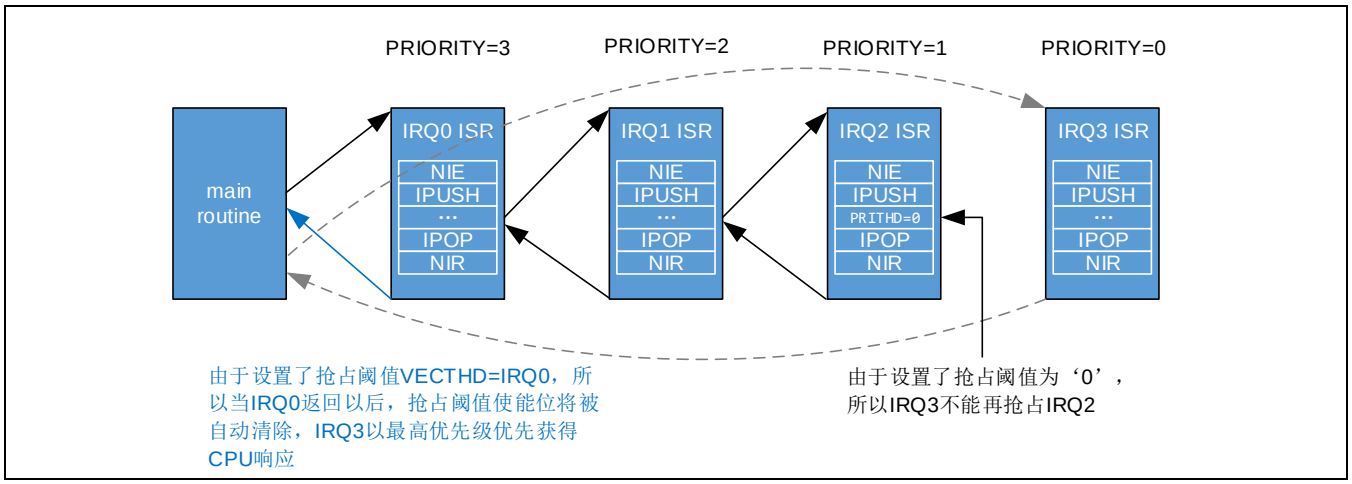


Figure 2-1 中断嵌套优先级示意图

2.3.3 中断响应时间和中断嵌套条件

一般中断在指令的边界上被确认，如下图所示。

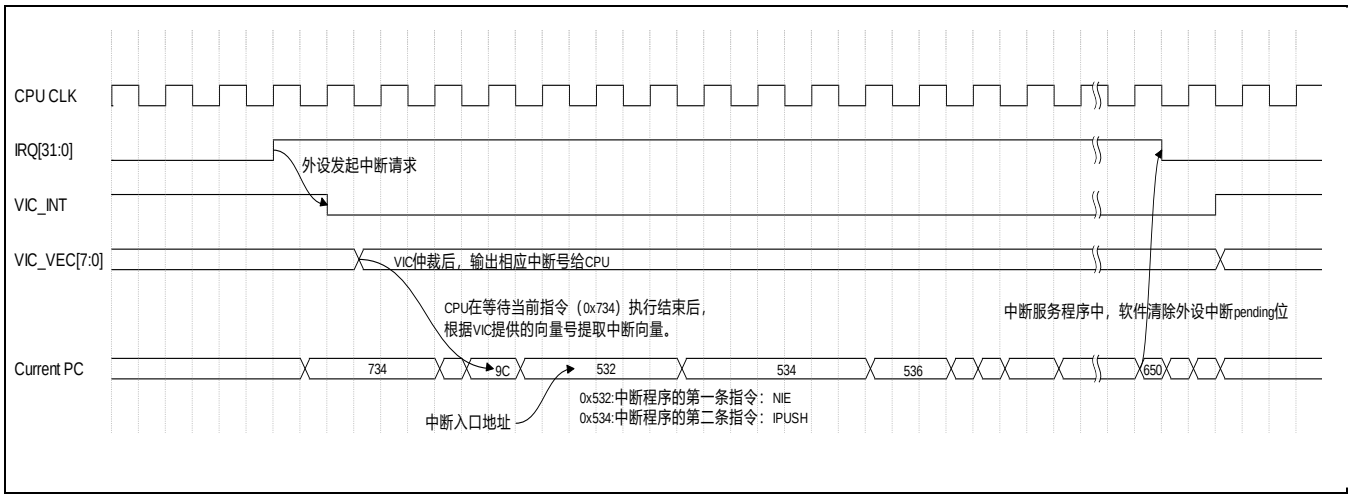


Figure 2-2 中断响应过程

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

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

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

为保证中断嵌套现场的保护和恢复，CPU在以下阶段不能被中断打断：

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

CPU在以下阶段可以安全的响应新的中断：

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

下图给出了IRQ0/IRQ1/IRQ2/IRQ3中断的嵌套过程。

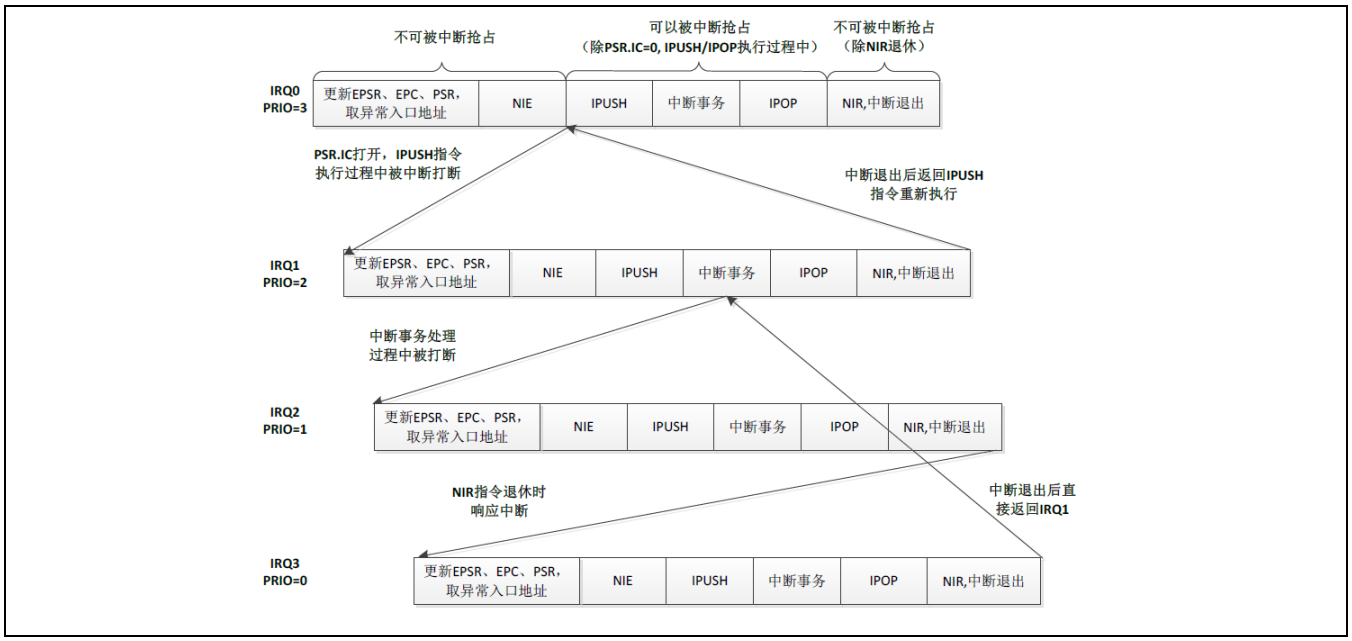


Figure 2-3 中断嵌套条件示例

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

2.3.4 中断唤醒

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

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

.

2.3.5 中断操作步骤

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

中断清除等待寄存器（VIC_ICPR）强制清除还没有被处理器响应的中断请求。

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

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

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

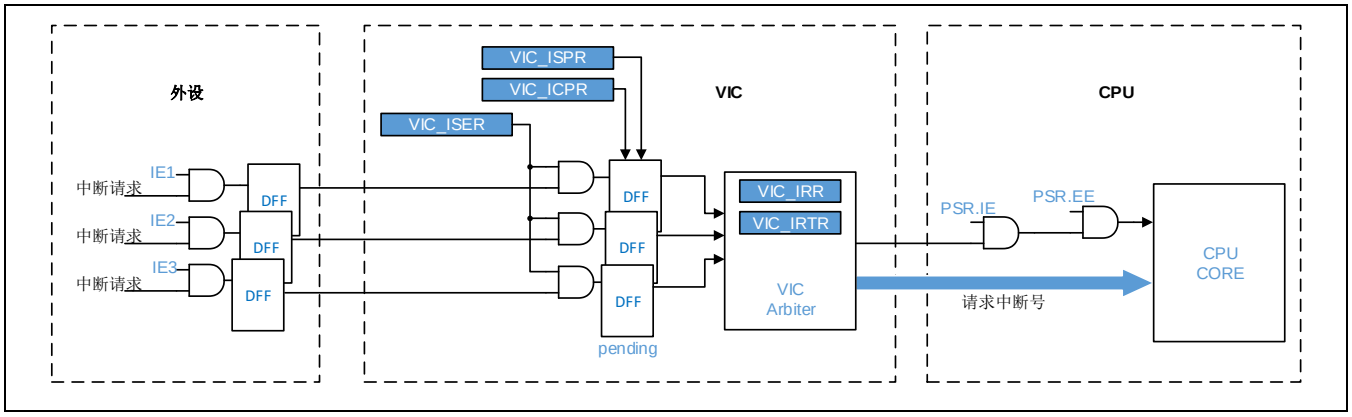


Figure 2-4 中断配置结构示意图

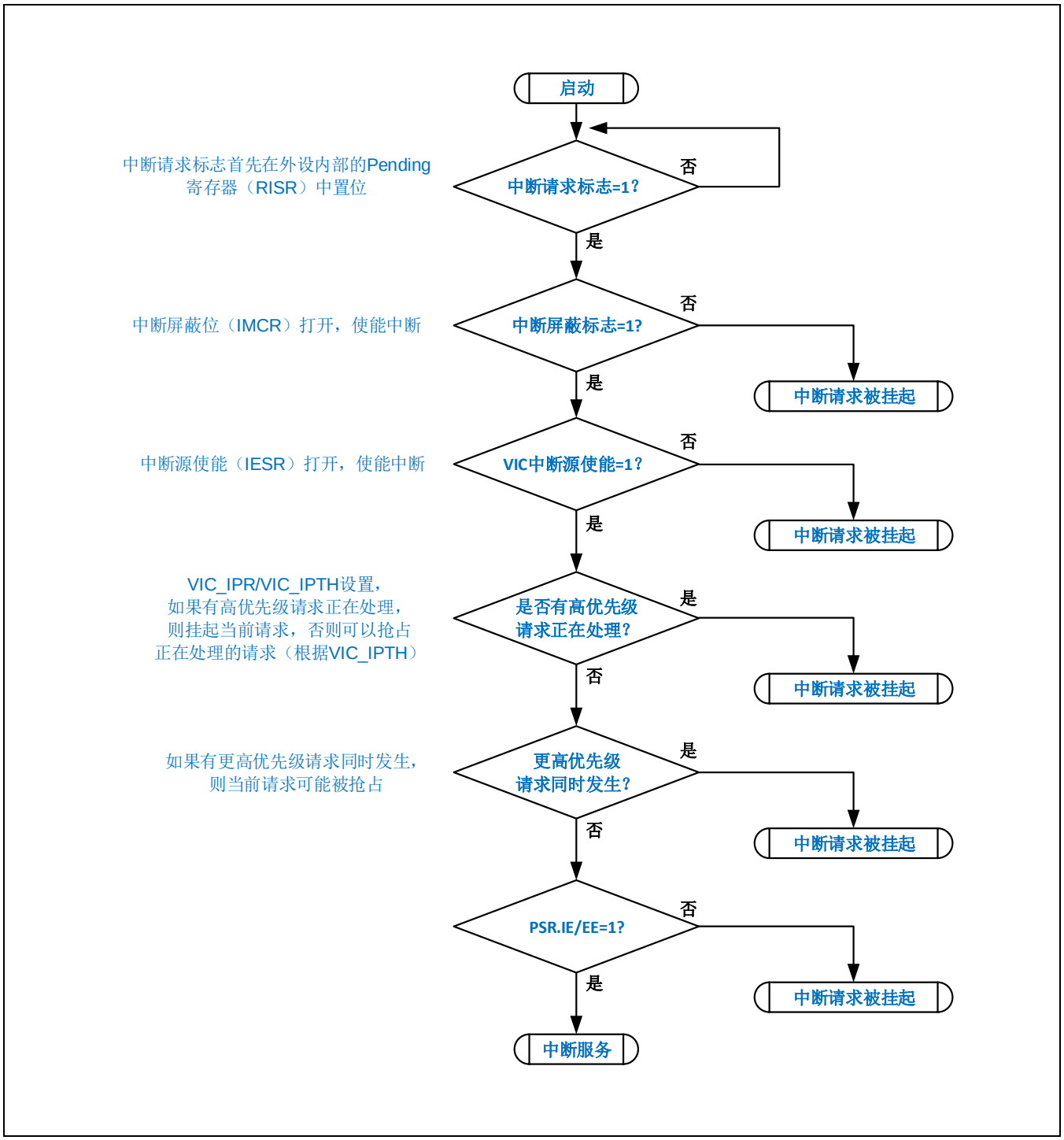


Figure 2-5 中断请求处理流程

2.4 寄存器说明

2.4.1 寄存器表

Base Address of INTERRUPT: 0xE000E000

Register	Offset	Description	Reset Value
VIC_ISER	0x100	Interrupt Set Enable Register	0x00000000
VIC_IWER	0x140	Interrupt Wakeup Enable Register	0x00000000
VIC_ICER	0x180	Interrupt Clear Enable Register	0x00000000
VIC_IWDR	0x1C0	Interrupt Wakeup Disable Register	0x00000000
VIC_ISPR	0x200	Interrupt Set Pending Register	0x00000000
VIC_ICPR	0x280	Interrupt Clear Pending Register	0x00000000
VIC_IABR	0x300	Interrupt Active Status Register	0x00000000
VIC_IPR0	0x400	Interrupt Priority Register 0	0x00000000
VIC_IPR1	0x404	Interrupt Priority Register 1	0x00000000
VIC_IPR2	0x408	Interrupt Priority Register 2	0x00000000
VIC_IPR3	0x40C	Interrupt Priority Register 3	0x00000000
VIC_IPR4	0x410	Interrupt Priority Register 4	0x00000000
VIC_IPR5	0x414	Interrupt Priority Register 5	0x00000000
VIC_IPR6	0x418	Interrupt Priority Register 6	0x00000000
VIC_IPR7	0x41C	Interrupt Priority Register 7	0x00000000
VIC_ISR	0xC00	Interrupt Status Register	0x00000000
VIC_IPTR	0xC04	Interrupt Priority Threshold Register	0x00000000

2.4.2 VIC_ISER(Interrupt Set Enable Register)

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
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	Type	Description
SETENA31~SETENA0	[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

2.4.3 VIC_IWER(Interrupt Wakeup Enable Register)

Address = Base Address+ 0x140, 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
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	Type	Description
SETENA31~SETENA0	[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

2.4.4 VIC_ICER(Interrupt Clear Enable Register)

Address = Base Address+ 0x180, 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
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	Type	Description
CLRENA31~CLRENA0	[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

2.4.5 VIC_IWDR(Interrupt Wakeup Disable Register)

Address = Base Address+ 0x1C0, 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
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	Type	Description
CLRENA31~CLRENA0	[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

2.4.6 VIC_ISPR(Interrupt Set Pending Register)

Address = Base Address+ 0x200, 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
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	Type	Description
SETPEND31~SETPEND0	[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

2.4.7 VIC_ICPR(Interrupt Clear Pending Register)

Address = Base Address+ 0x280, 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
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	Type	Description
CLRPEND31~CLRPEND0	[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

2.4.8 VIC_IABR(Interrupt Active Status Register)

Address = Base Address+ 0x300, 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
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	Type	Description
ACTIVE31~ACTIVE0	[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）

2.4.9 VIC_IPR0(Interrupt Priority Register 0)

Address = Base Address+ 0x400, 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
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	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R

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

2.4.10 VIC_IPR1(Interrupt Priority Register 1)

Address = Base Address+ 0x404, 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
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	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R

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

2.4.11 VIC_IPR2(Interrupt Priority Register 2)

Address = Base Address+ 0x408, 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
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	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R

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

2.4.12 VIC_IPR3(Interrupt Priority Register 3)

Address = Base Address+ 0x40C, 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
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	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R

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

2.4.13 VIC_IPR4(Interrupt Priority Register 4)

Address = Base Address+ 0x410, 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
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	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R

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

2.4.14 VIC_IPR5(Interrupt Priority Register 5)

Address = Base Address+ 0x414, 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
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	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R

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

2.4.15 VIC_IPR6(Interrupt Priority Register 6)

Address = Base Address+ 0x418, 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
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	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R

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

2.4.16 VIC_IPR7(Interrupt Priority Register 7)

Address = Base Address+ 0x41C, 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
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	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R

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

2.4.17 VIC_ISR(Interrupt Status Register)

Address = Base Address+ 0xC00, 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											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	Type	Description
VECPENDING	[20:12]	RW	指示当前等待的最高优先级中断向量号
VECACTIVE	[8:0]	RW	指示当前CPU正在处理的中断向量号

2.4.18 VIC_IPTR(Interrupt Priority Threshold Register)

Address = Base Address+ 0xC04, 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
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	Type	Description
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
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）
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。

3 系统定时器 (CORET)

3.1 概述

系统定时器是 CK801 CPU一个内部模块，它主要用于计时。系统定时器提供了一个简单易用的 24 位循环递减的计数器，当系统计时器使能时，计数器开始工作。当计数器递减到 0 时，会向中断控制器发起中断请求。

3.2 功能描述

3.2.1 模块框图

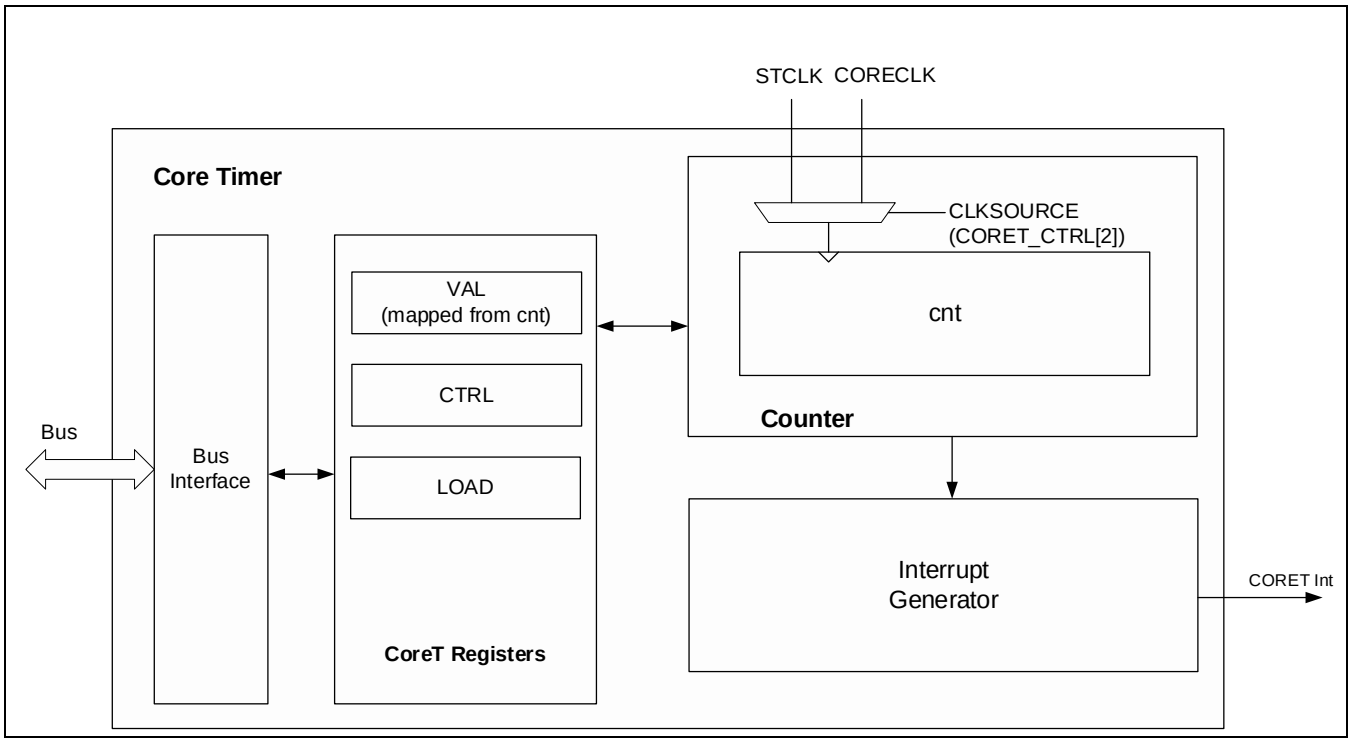


Figure 3-1 CORET模块框图

3.2.2 功能说明

3.2.2.1 定时器的时钟源

CORET 定时器有两个可选时钟源：

- CPU 时钟 (HCLK)
- 系统时钟的 8 分频 STCLK

时钟源的选择通过 CTRL 寄存器的第 2 位 CLKSOURCE 来实现。
时钟的使能/禁止和各种配置请参考 SYSCON 章节。

3.2.2.2 定时器的工作原理

CORET定时器是CK801 CPU提供的一个简单易用的24位循环递减的计数器，包含在CPU Core内部，产生的中断具有最高的优先级。CORET定时器可以用作任何简单的计时，或者可以作为操作系统的SYSTICK定时器。

当系统定时器使能 (CTRL[0]=1) 时，计数器开始工作。计数器从预设的值 (LOAD寄存器) 开始递减，当计数器递减到0时，如果使能了CORET中断 (CTRL[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_CTRL	0x010	控制寄存器	0x00000000
CORET_LOAD	0x014	回填值寄存器	0x00XXXXXX
CORET_VAL	0x018	当前值寄存器	0x00XXXXXX

3.3.2 CORET_CTRL(控制寄存器)

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															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VAL寄存器以及任何写VAL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STCLK，STCLK = CORECLK/8 且芯片SLEEP模式下仍然有时钟 1：时钟源为CORECLK，芯片SLEEP模式下无时钟 STOP模式下，两种时钟源都没有时钟。
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_LOAD(回填值寄存器)

Address = Base Address+ 0x014, Reset Value = 0x00XXXXXX

31	30	29	28	27	26	25	24	23	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	X	X	X	X	X	X	X	X	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	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_VAL寄存器。 向CORET_LOAD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_VAL 变为非0值时)，才可以将 CORET_LOAD 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_VAL(当前值寄存器)

Address = Base Address+ 0x018, Reset Value = 0x00XXXXXX

31	30	29	28	27	26	25	24	23	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	X	X	X	X	X	X	X	X	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	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	读 CORET_VAL 会返回访问寄存器时计数器的值。 写CORET_VAL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_LOAD里的值并赋给CORET_VAL。 注意写CORET_VAL不会导致系统计时器的中断状态位发生改变。

4 闪存控制器(IFC)

4.1 概述

本章节描述用来控制内部程序存储的闪存控制器。APT32F104x 系列片上带有 48K Bytes or 16K Bytes 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。APT32F104x 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。

4.1.1 主要特性

- 程序闪存(PROM)大小: 48K Bytes or 16K 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: 128 Bytes(PROM), 128 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

4.2 功能描述

4.2.1 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：

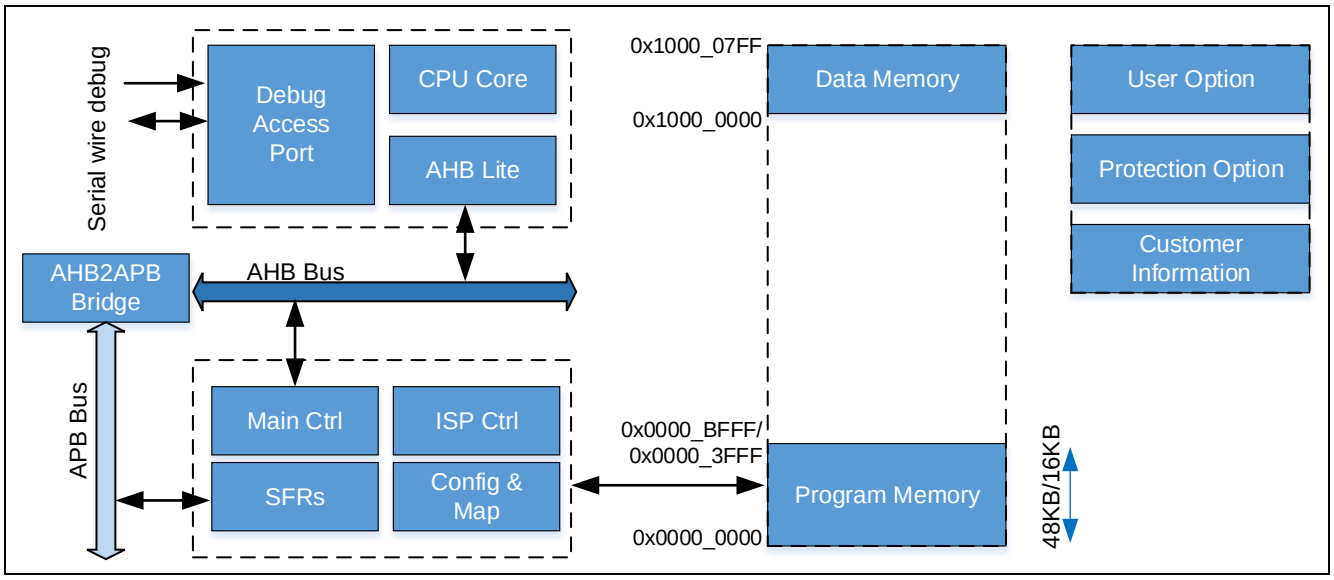


Figure 4-1 IFC模块框图

4.2.2 模块结构

APT32F104x 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 384/128 个页空间，每页有 128 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(Word)。

区域	页名称	大小	起始地址	结束地址
PROM Within 48KB	Page 0	128B	0x0000_0000	0x0000_007F
	Page 1	128B	0x0000_0080	0x0000_00FF
	Page 2	128B	0x0000_0100	0x0000_017F
	Page 3	128B	0x0000_0180	0x0000_01FF
	Page 4	128B	0x0000_0200	0x0000_027F
	Page 5	128B	0x0000_0280	0x0000_02FF
	Page 6	128B	0x0000_0300	0x0000_037F
	:	:	:	:
	Page 254	128B	0x0000_BF00	0x0000_BF7F
	Page 255	128B	0x0000_BF80	0x0000_BFFF
DROM	Page 0	128B	0x1000_0000	0x1000_007F
	Page 1	128B	0x1000_0080	0x1000_00FF
	Page 2	128B	0x1000_0100	0x1000_017F
	Page 3	128B	0x1000_0180	0x1000_01FF
	:	:	:	:
	Page 22	128B	0x1000_0700	0x1000_077F
	Page 23	128B	0x1000_0780	0x1000_07FF

Table 4-1 闪存地址映射

闪存结构如下图所示：

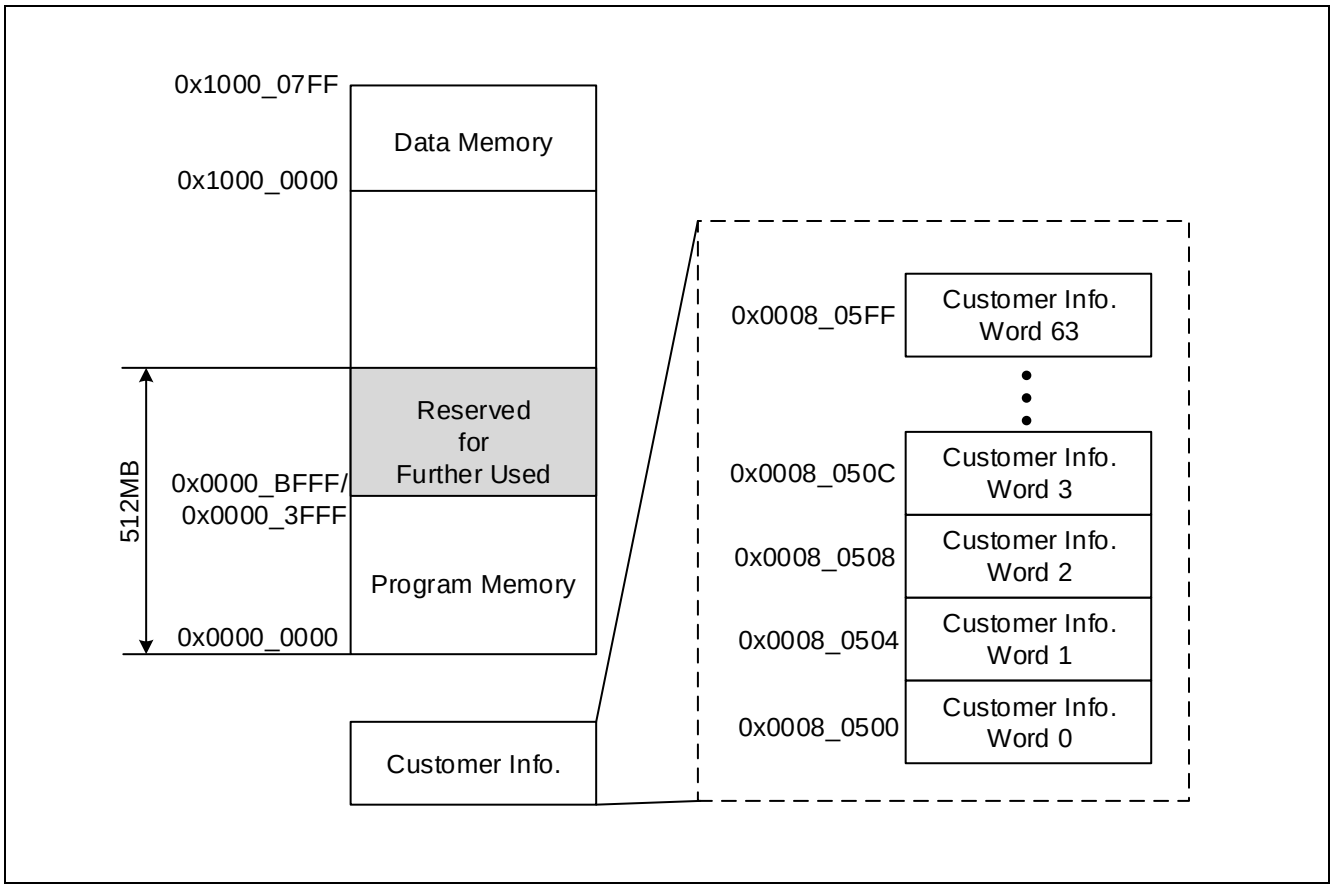


Figure 4-2 闪存地址空间结构

4.2.3 数据闪存

APT32F104x 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 128 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。

4.2.4 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。

4.2.4.1 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。

31	30	29	28	27	26	25	24	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 PA0.13为外部复位管脚，IO功能被禁用 其它值 PA0.13禁用外部复位功能，当作IO使用
IWDT	[31:16]	0x55AA: 禁用系统看门狗 IWDT 其它值: 使能系统看门狗 IWDT

Table 4-2 用户选项功能说明

4.2.4.2 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。

- 三种保护功能的小结：

保护功能	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

4.2.4.3 客户信息区 (Customer Information)

客户信息区域由 256 个字(1024 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。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_0800 ~ 0x0008_0BFF)直接读取。

4.2.4.4 UID (Unique ID)

UID 区域由 3 个字(12 字节)组成，制造工厂在生产时写入。工厂在写入后，该区域存储的内容不会被 ISP 或者烧录工具擦除。该区域只能通过 SYSCON 模块中相应的镜像寄存器 UID0~UID2 进行读取。UID 为该芯片的标识，每个芯片有单独唯一的 UID。

4.2.5 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。

自定义选项	烧片机	ISP 功能(IFC_CMR)
User Option	√	√
保护选项	√	√
客户信息区	√	X
UID	√	X

Table 4-3 自定义选项的设置方法

4.2.6 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。

4.2.7 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 (AHB接口)
- SWD接口

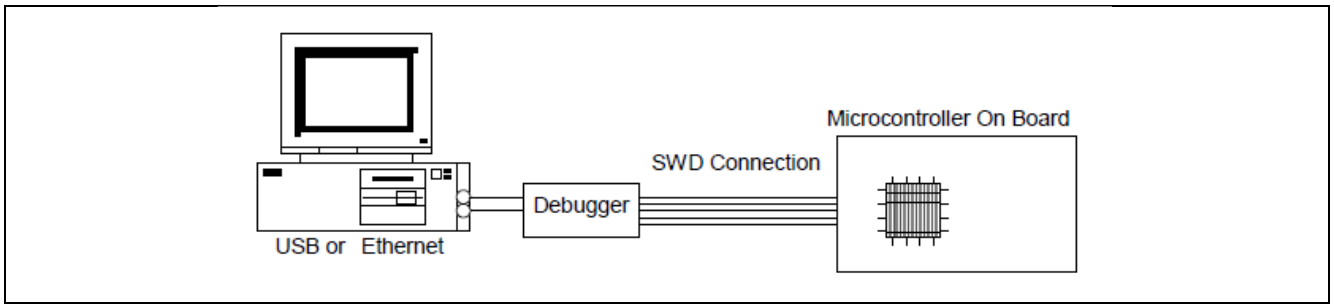


Figure 4-3 通过调试接口的烧写

- 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。

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

Table 4-4 闪存烧写信号

4.2.8 ISP功能

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

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

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。

4.2.8.1 页擦除操作

每页闪存中有 128 字节。页擦除操作会擦除 IFC_FM_ADDR 中地址所在的那一页闪存。在 ISP 操作前，用户必须将秘钥 0x5A5A_5A5A 写入 IFC_KEY 寄存器以禁止闪存模块的擦除/烧写保护。之后，用户需要将烧写的地址写入 IFC_FM_ADDR 寄存器，并将 IFC_CMR 里的指令 CMD[3:0]写为 0x2(页擦除操作)，最后将 IFC_CR 的 START 位置 1 启动该操作的执行。在 ISP 操作完成后，IFC_RISR 里的 END 位会置 1。用户同时也可以查询 IFC_CR 里的 START 位来判断 ISP 操作是否完成。

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

4.2.8.2 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 128 (PROM/DROM) 字节的页进行操作。本闪存包含一个页缓存空间，在写操作中，需要先将整个页的数据先写入到页缓存空间中，再执行写操作的命令，将整个页缓存空间中的数据一起写入闪存中。

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

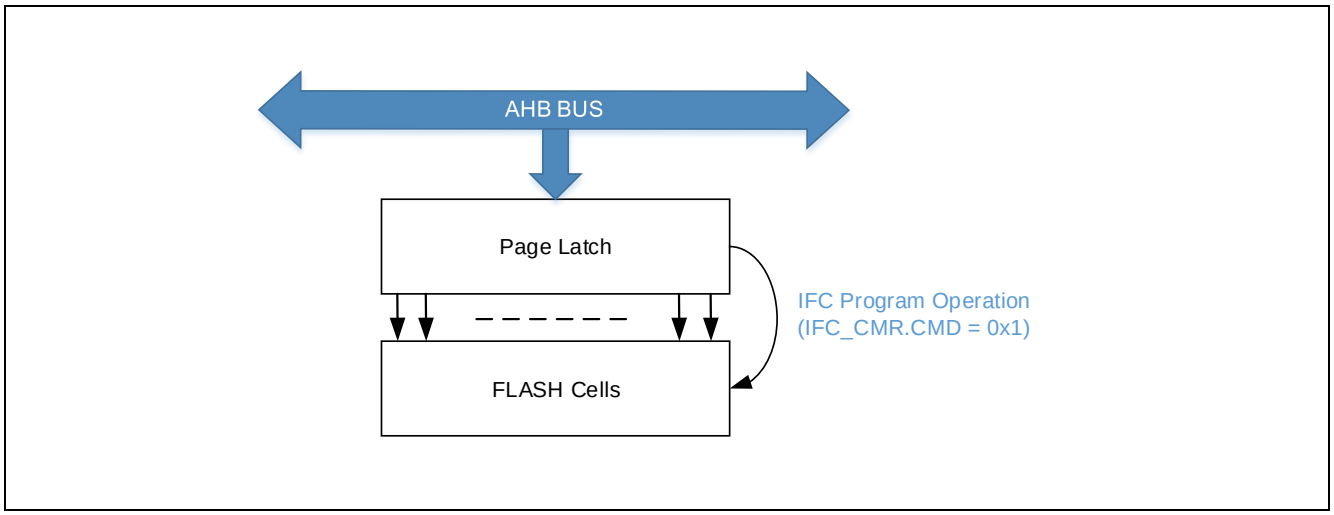


Figure 4-4 将页缓存数据写入闪存中

具体步骤如下：

1. 清除页缓存空间(IFC_CMR 寄存器中的 CMD = 0x7)。
2. 将需要写入的数据填入页缓存 (页缓存可通过总线直接写入，类似于 SRAM 内存空间)。
3. 预编程设定(IFC_CMR 寄存器中的 CMD = 0x6)，设置下一步的编程为预编程。

4. 执行写操作(IFC_CMR 寄存器中的 CMD = 0x1)，进行预编程。

5. 执行页擦除操作(IFC_CMR 寄存器中的 CMD = 0x2)，擦除整个页数据。

6. 执行写操作(IFC_CMR 寄存器中的 CMD = 0x1)，将页缓存的数据写入闪存中。

在每次执行 IFC 操作的命令前，用户必须将秘钥 0x5A5A_5A5A 写入 IFC_KEY 寄存器，之后将 IFC_CR 的 START 位置 1 启动该操作的执行。在 ISP 操作完成后，IFC_RISR 里的 END 位会置 1。用户同时也可以查询 IFC_CR 里的 START 位来判断 ISP 操作是否完成。

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

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

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

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

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

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

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

4.2.8.3 片擦除操作

片擦除操作会擦除整个闪存的程序存储，但不会擦除数据存储区域和自定义选项的区域。片擦除操作只能在用户特权模式下才能执行。在片擦除中，不需要指令 ISP 操作相关的地址和数据寄存器。在 ISP 操作前，用户必须将秘钥 0x5A5A_5A5A 写入 IFC_KEY 寄存器以禁止闪存模块的擦除/烧写保护。之后，将 IFC_CMR 里的指令 CMD[3:0]写为 0x4(片擦除操作)，HMODE[1:0]写为 0x1(用户特权模式)，最后将 IFC_CR 的 START 位置 1 启动该操作的执行。在 ISP 操作完成后，IFC_RISR 里的 END 位会置 1。用户同时也可以查询 IFC_CR 里的 START 位来判断 ISP 操作是否完成。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
```

4.2.8.4 擦除自定义选项区域

这个自定义选项擦除操作会擦除所有的 User Option，保护选项和客户信息区域。在 ISP 操作前，用户必须将秘钥 0x5A5A_5A5A 写入 IFC_KEY 寄存器以禁止闪存模块的擦除/烧写保护。之后，将 IFC_CMR 里的指令 CMD[3:0]写为 0x5(自定义选项擦除操作)，HMODE[1:0]写为 0x1(用户特权模式)，最后将 IFC_CR 的 START 位置 1 启动该操作的执行。在 ISP 操作完成后，IFC_RISR 里的 END 位会置 1,。用户同时也可以查询 IFC_CR 里的 START 位来判断 ISP 操作是否完成。

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

4.2.8.5 烧写自定义选项操作

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

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

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

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

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

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

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

4.2.9 闪存控制器的中断

闪存操作有 7 个中断源，如下所示。

中断	描述
ERS_END	擦除指令执行完成中断
PGM_END	写操作指令执行完成中断
PEP_END	预编程指令执行完成中断 (该中断在设置了预编程选项后的写闪存操作完成时产生)
PROT_ERR	保护错误；当硬件保护锁使能，仍然进行写操作或擦除操作
UDEF_ERR	未定义指令错误；CMD中定义的操作指令非法或者不允许在当前模式中执行
ADDR_ERR	地址错误；FM_ADDR中定义的地址超出了最大地址范围 (注意)
OVW_ERR	非法操作错误；当ISP操作正在进行时，尝试修改CMD，FM_ADDR，FM_DR，START寄存器

Table 4-5 中断源描述

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，而且该中断发生了(RISR 相应位置 1)，那么该中断会被送至 CPU 处理，进入中断子程序。用户可以在中断子程序中用 ICR 寄存器清除相应的中断状态位。

4.2.10 闪存控制流程图

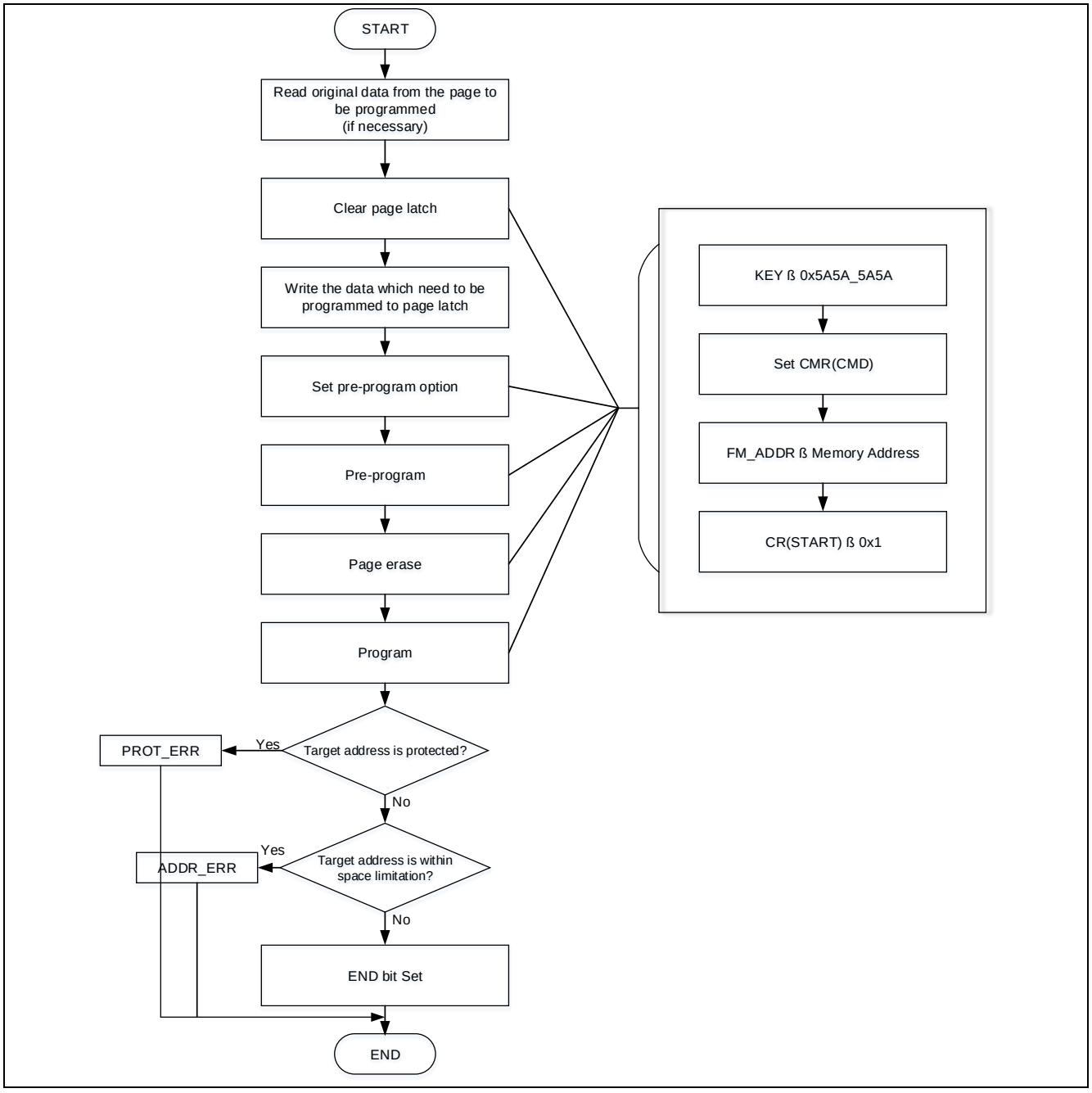


Figure 4-5 Flash Write – Normal Mode

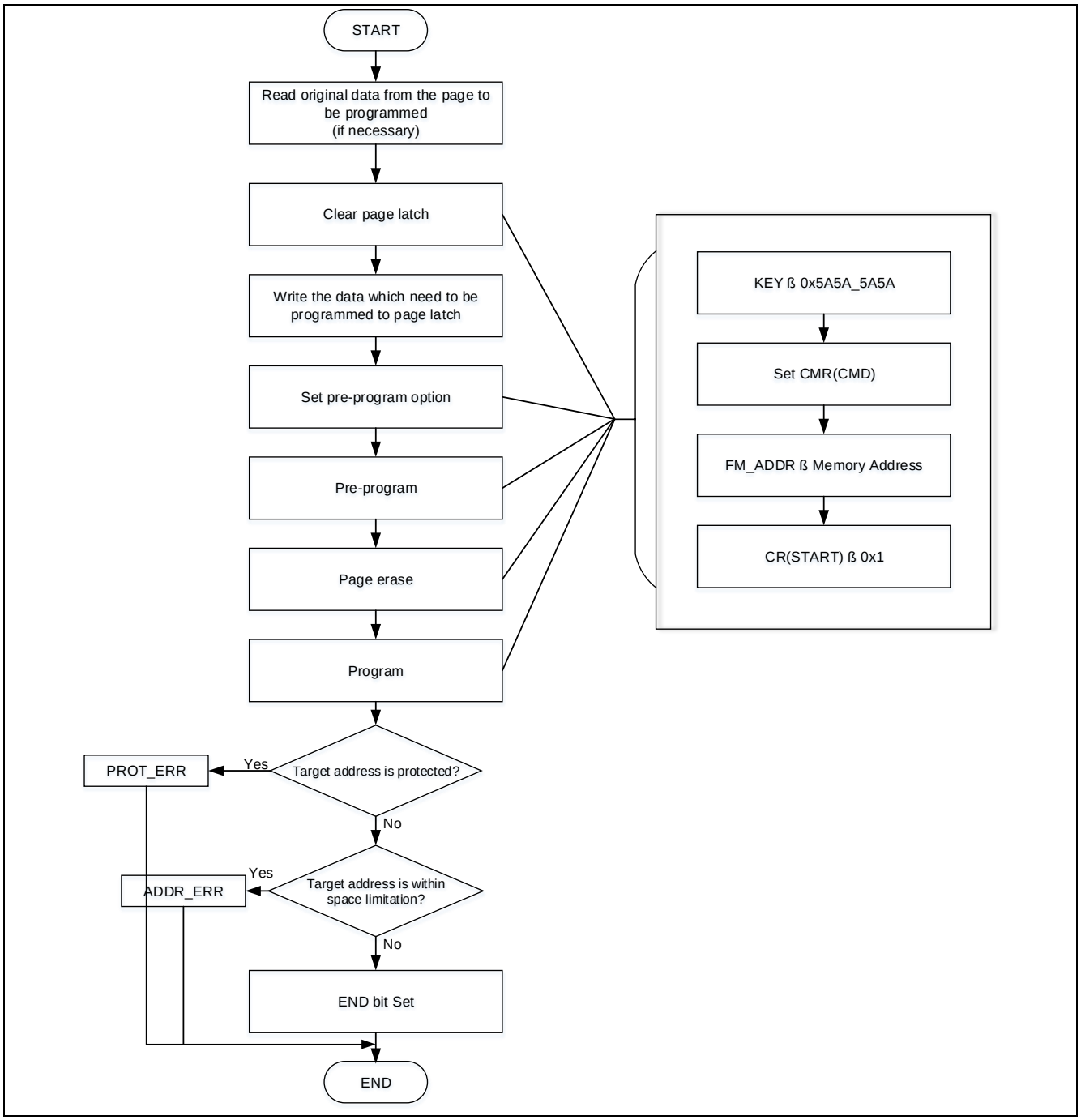


Figure 4-6 Flash Write – Parallel Mode

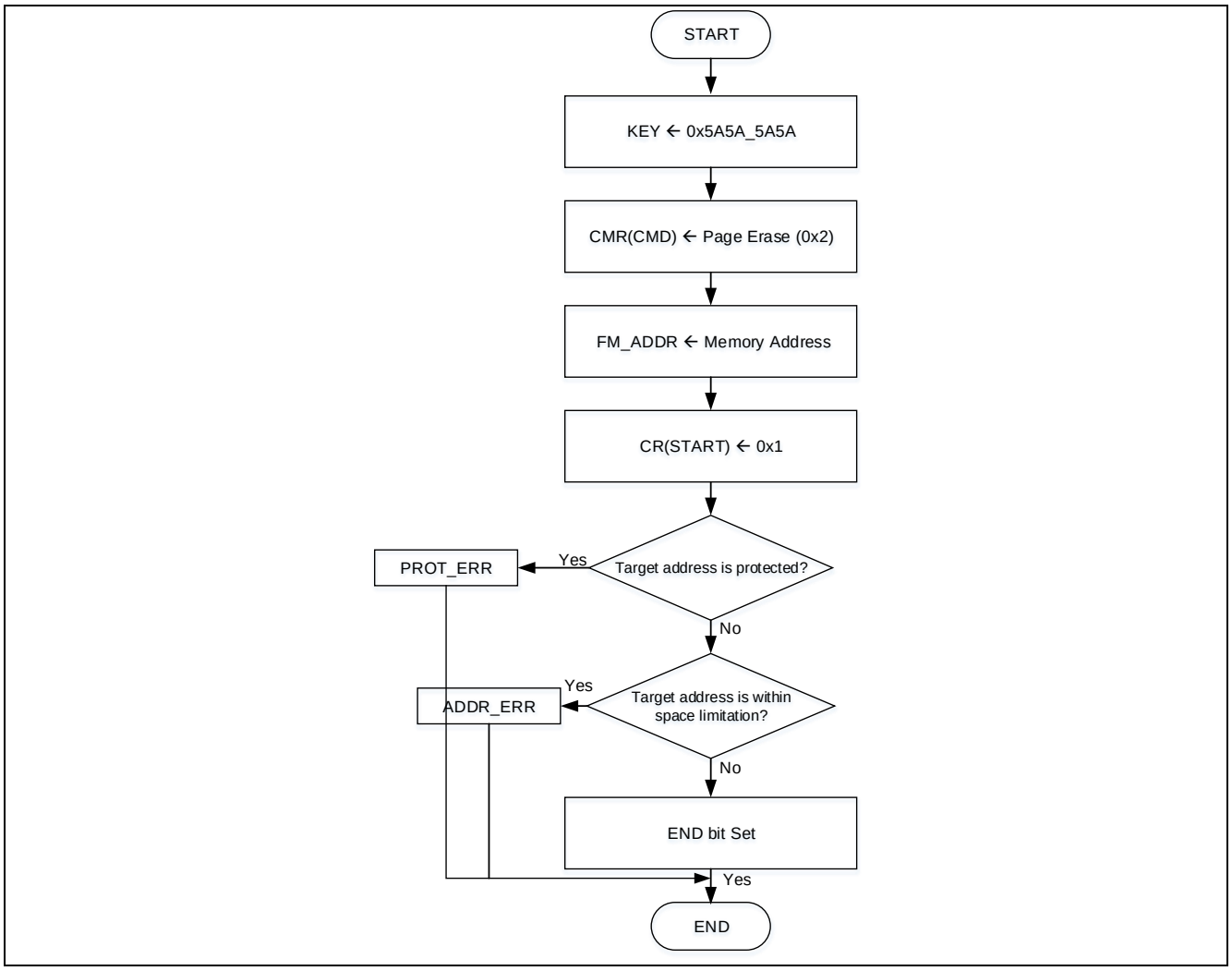


Figure 4-7 Page Erase

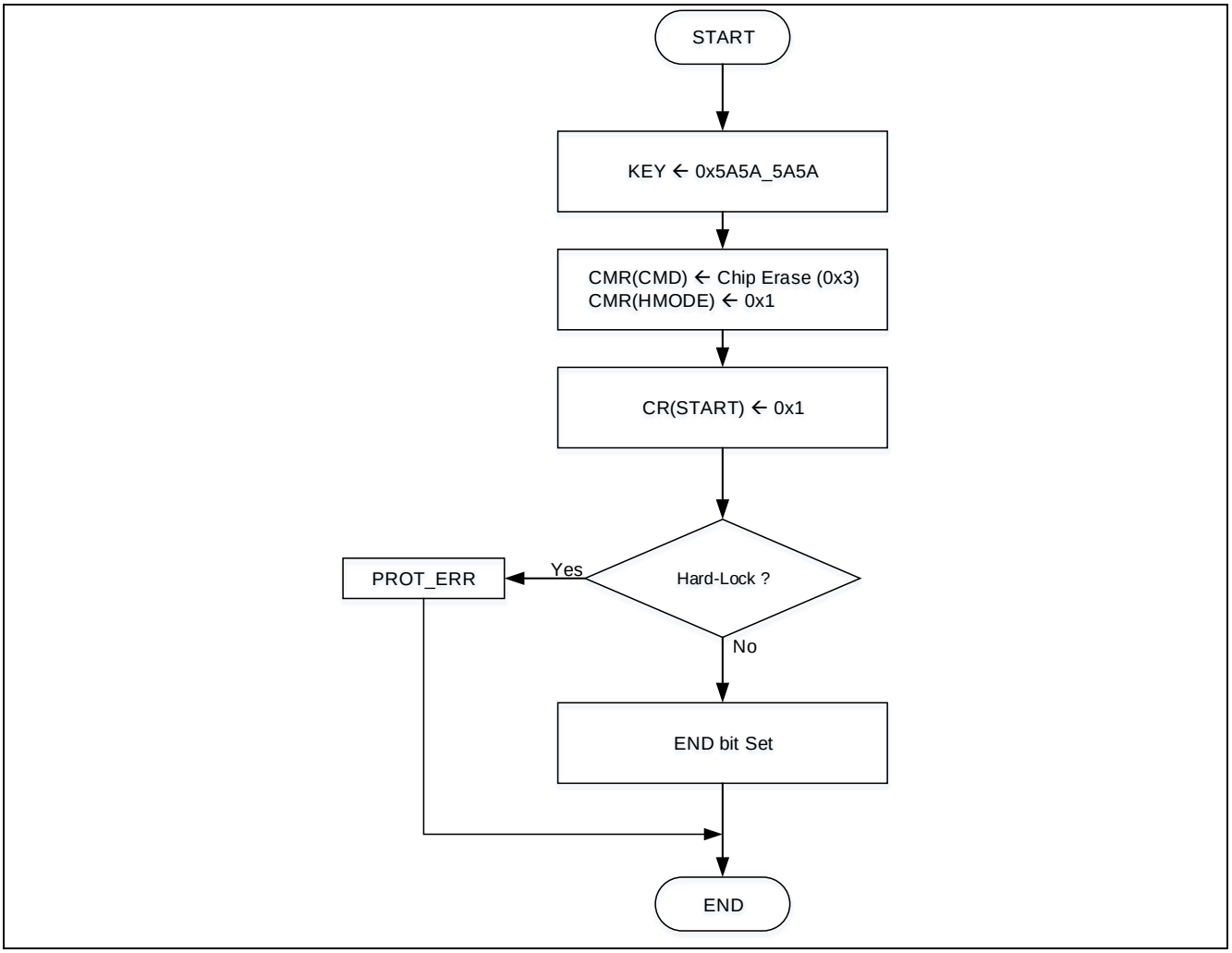


Figure 4-8 Chip Erase

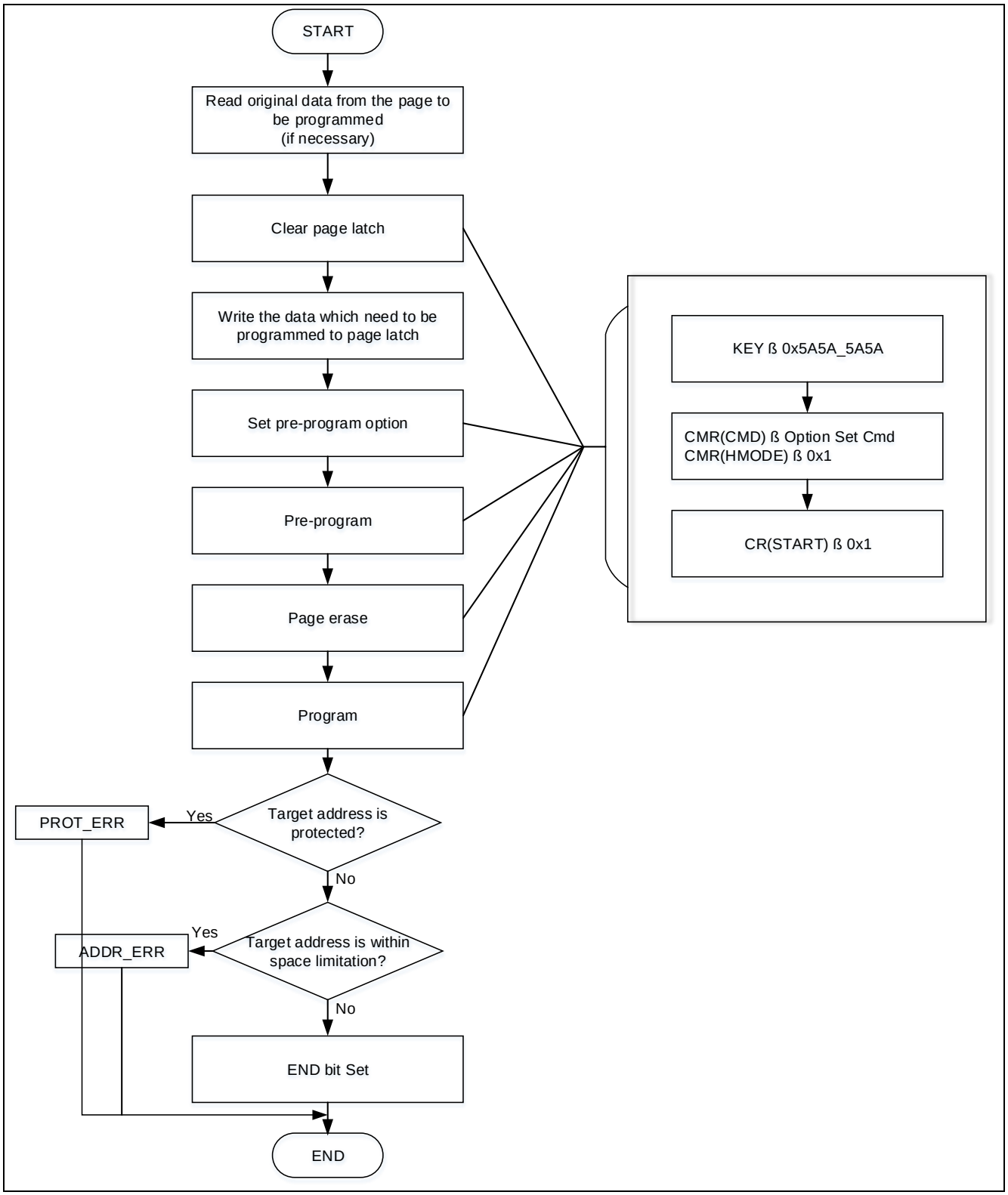


Figure 4-9 Option Cells Write

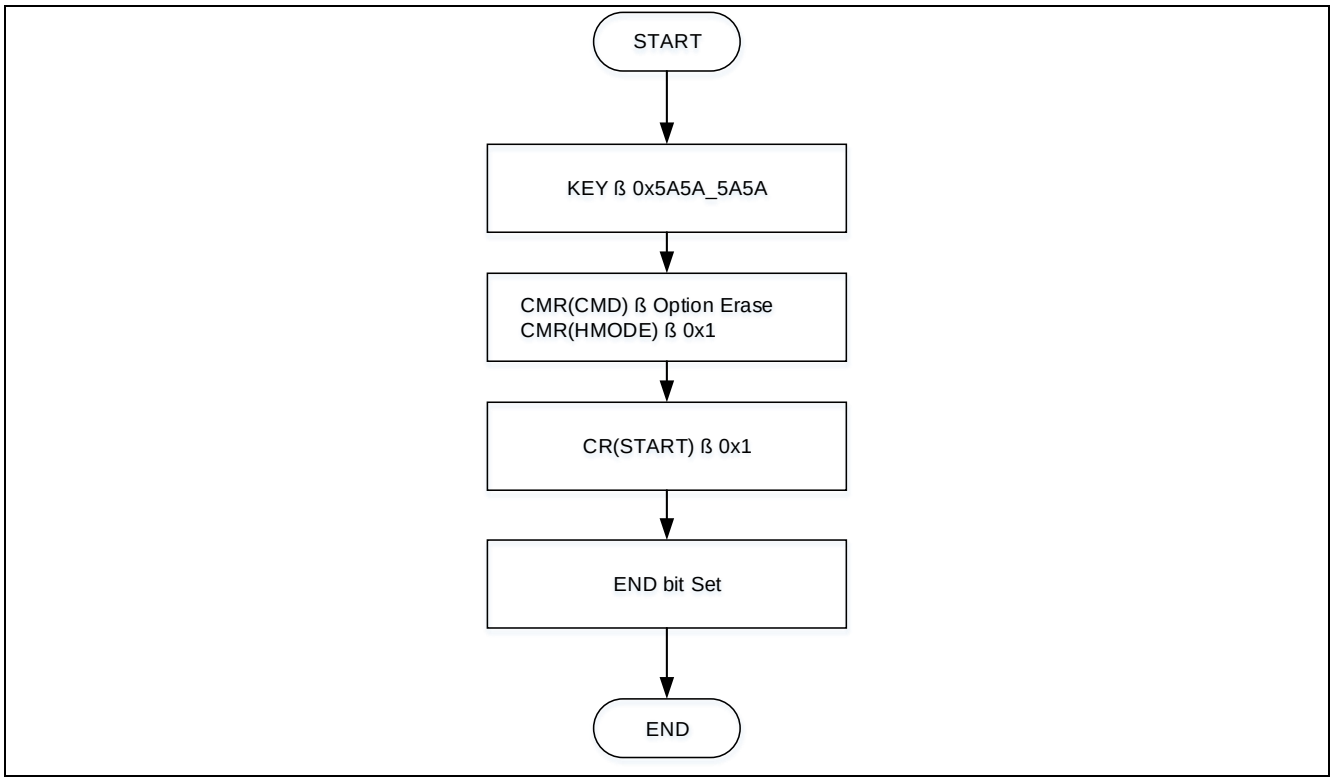


Figure 4-10 Option Cells Erase

4.3 寄存器说明

4.3.1 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	0x000001B0
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

4.3.2 IFC_IDR(闪存控制器ID寄存器)

Address = Base Address+ 0x00, Reset Value = 0x000001B0

31	30	29	28	27	26	25	24	23	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	1	0	1	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
IDCODE	[15:0]	RW	ID代码 (0x01B0)

4.3.3 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

4.3.4 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 除CEDR外的所有寄存器都会恢复初始值

4.3.5 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 10: 保留 11: 保留 在普通模式下，只有页擦除和写操作有效。其它指令都必须在用户特权模式下执行。
CMD	[3:0]	RW	写/擦除指令寄存器 CMD[3:0] 指令 0x1 写操作 0x2 页擦除 (Page Erase) 0x3 保留，禁止使用 0x4 片擦除 (Chip Erase) 0x5 自定义选项擦除 0x6 预编程设定 0x7 页缓存清除 0x8 – 0xC 保留，禁止使用 0xD 禁用调试口重映射 (SWD Remap) 0xE 使能调试口重映射 (SWD Remap) 0xF 写User Option操作 注意： 1. 当执行ISP操作时，禁止读取闪存内容 2. 当操作完成后，IFC_CMR寄存器会自动清零

			3. 如果IFC_KR的秘钥值不对，那么指令不会被执行

4.3.6 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

4.3.7 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													WAIT		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW

Name	Bit	Type	Description
SPEED	[16]	RW	FLASH IP速度模式选择 0: 低速模式 1: 高速模式
WAIT	[2:0]	RW	闪存读等待周期 0: 闪存读取中等待0个周期 n: 闪存读取中等待n个周期

注意：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
8MHz < SYSCLK ≤ 16MHz	0	1
SYSCLK ≤ 8MHz	0	0

4.3.8 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

4.3.9 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。

4.3.10 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	地址错误中断使能/禁止 0: 禁止中断 1: 使能中断
UDEF_ERR	[13]	RW	未定义指令错误中断使能/禁止 CMD中定义的操作指令非法或者不允许在当前模式中执行 0: 禁止中断 1: 使能中断
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

4.3.11 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

4.3.12 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	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状态 0: 该中断没有发生 1: 该中断发生
ADDR_ERR	[14]	R	地址错误中断的状态 0: 该中断没有发生 1: 该中断发生
UDEF_ERR	[13]	R	未定义指令错误中断的状态 0: 该中断没有发生 1: 该中断发生
PROT_ERR	[12]	R	保护错误中断的状态 0: 该中断没有发生 1: 该中断发生
PEP_END	[2]	R	预编程指令执行完成中断的原始状态 0: 该中断没有发生 1: 该中断发生
PGM_END	[1]	R	编程指令执行完成中断的原始状态 0: 该中断没有发生 1: 该中断发生
ERS_END	[0]	R	擦除指令执行完成中断的原始状态 0: 该中断没有发生 1: 该中断发生

4.3.13 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

5 系统控制器（SYSCON）

5.1 概述

系统控制器用于管理和配置系統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某一外围，那它就不具备该外围的所有资源。具体参考芯片的数据手册。

系统控制器的基本特性：

- 系统时钟源选择和 HCLK/PCLK 频率管理
  - 支持多种时钟源作为系统时钟运行：
    - 内部低速振荡器（IMOSC）为缺省时钟源：24MHz
    - 外部晶振（EMOSC）：0.4MHz ~ 24MHz/32.768KHz
    - PLL（源可选 IMOSC/EMOSC）: 48MHz max
    - 内部超低功耗振荡器（ISOSC）16 分频后：31.25KHz
  - 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。

5.2 功能描述

5.2.1 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。

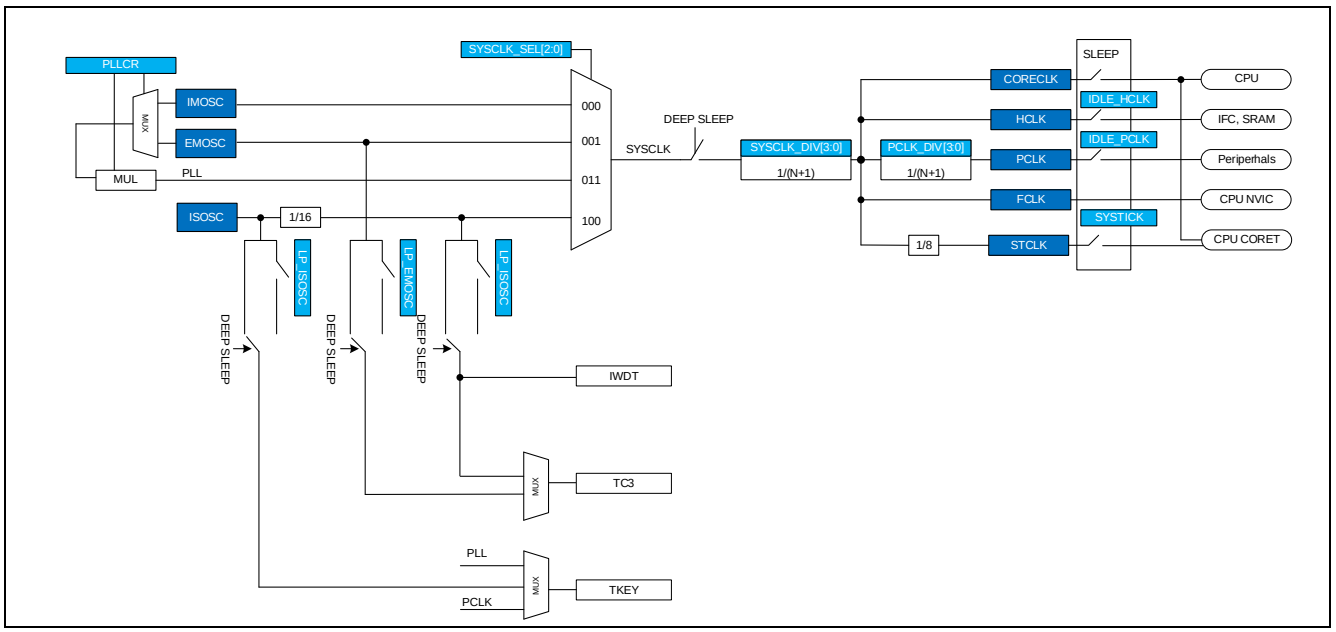


Figure 5-1 时钟结构示意图

NOTE:

1. 在 POR 完成以后，IMOSC 为系统的缺省时钟源。

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

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

每个外设都有独立的时钟使能控制开关，在操作该外设前，必须使能该模块的 PCLK 时钟控制。PCLK 的时钟控制可以通过 PCER、PCDR 这组寄存器进行操作。对 PCER 寄存器的对应控制位写入 1 时，可以使能指定模块

的 PCLK，对 PCDR 寄存器的对应控制位写入 1 时，可以关闭指定模块的 PCLK。通过读取 PCSR 寄存器可以获得开关的当前状态。关闭不使用的模块的 PCLK，可以有效降低系统的动态功耗。

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

5.2.2 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, 24MHz）：

内部高精度主振荡器，缺省选择 IMOSC。

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

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

ISOSC（Internal Sub OSC, 31.25KHz）：

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。

PLL（48MHz max）：源可选 EMOSC(≥16MHz)或 IMOSC

在芯片上电初始化时，系统将自动选择 IMOSC/4 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。

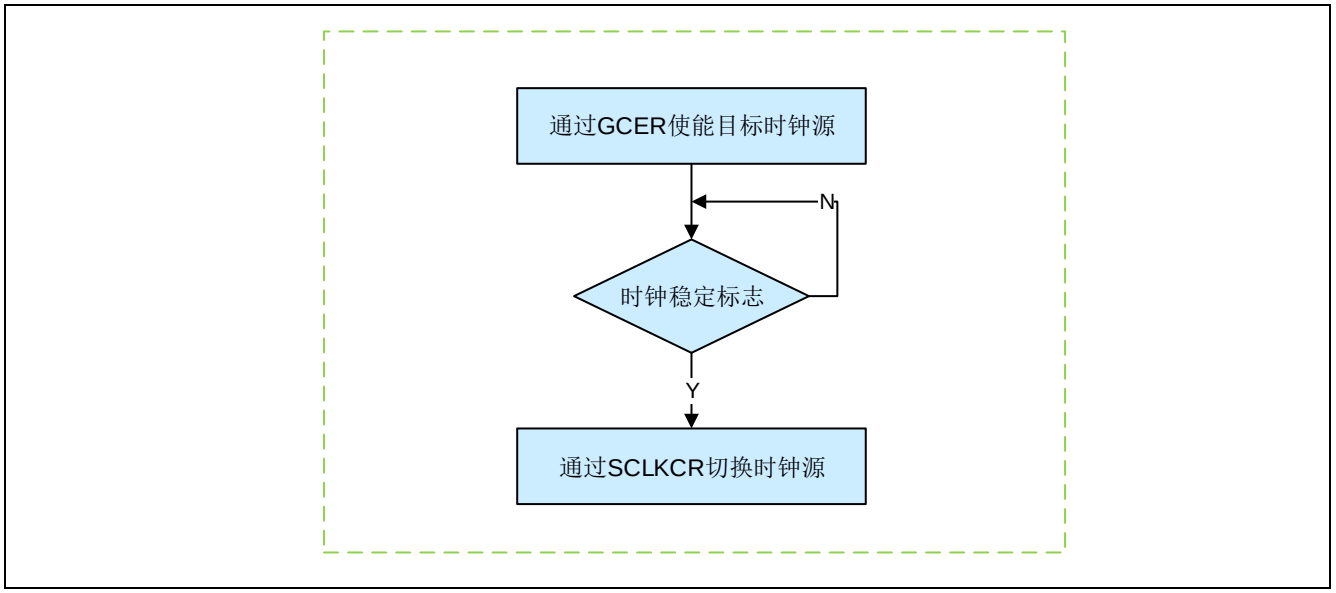


Figure 5-2 时钟源切换示意图

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。

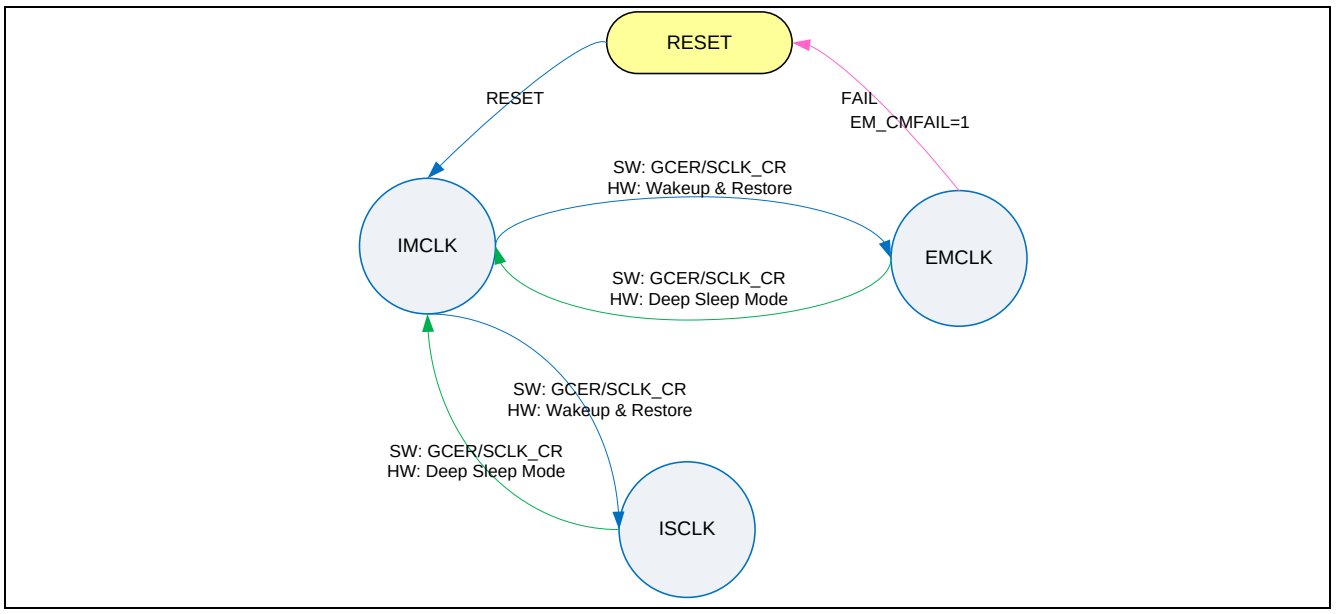


Figure 5-3 时钟切换状态机

5.2.2.1 选择内部时钟源进行工作

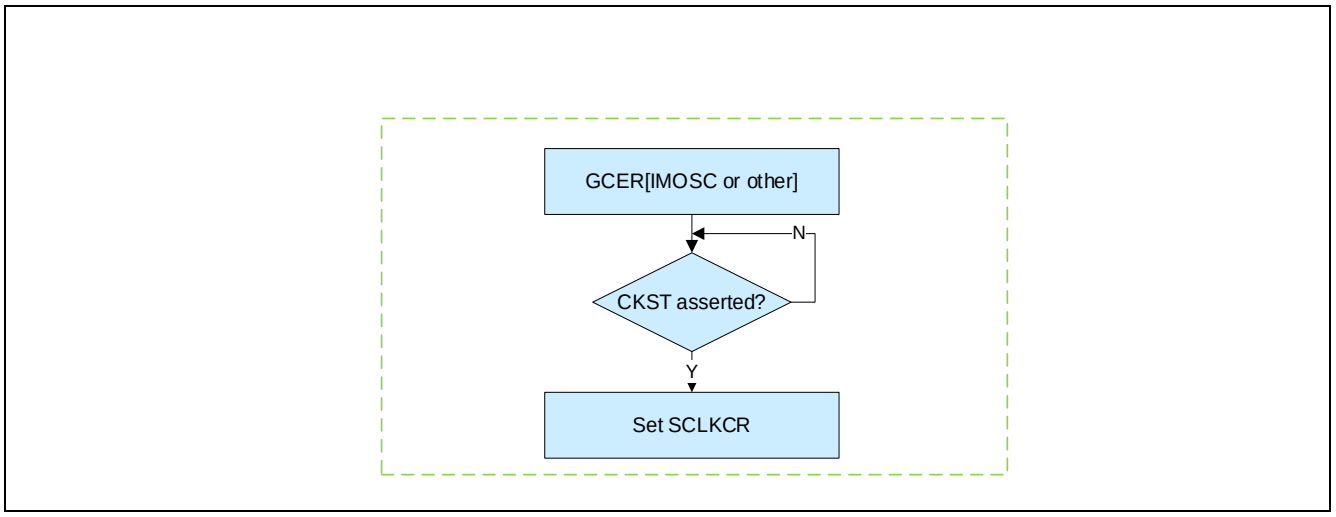


Figure 5-4 配置内部时钟源

在系统硬件初始化完成后，用户可以根据实际应用选择更低（ISOSC）或更高（PLL）频率的时钟作为系统时钟。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。

5.2.2.2 内部时钟源频率微调

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

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

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

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 5-1 系数 K 值计算方式

OSC	K 值
IMOSC	367-TRM
ISOSC	339-TRM

5.2.2.3 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：

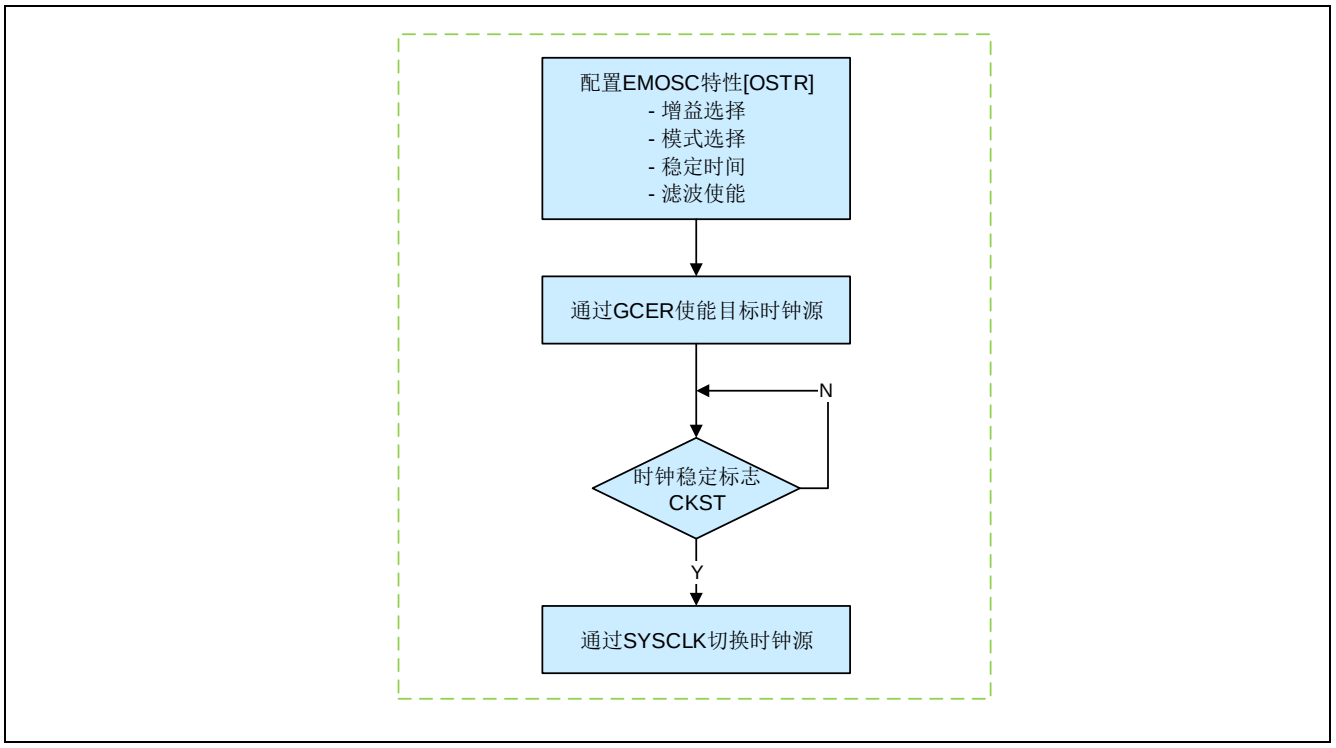


Figure 5-5 配置外部时钟源

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 5-2 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。

5.2.2.4 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）

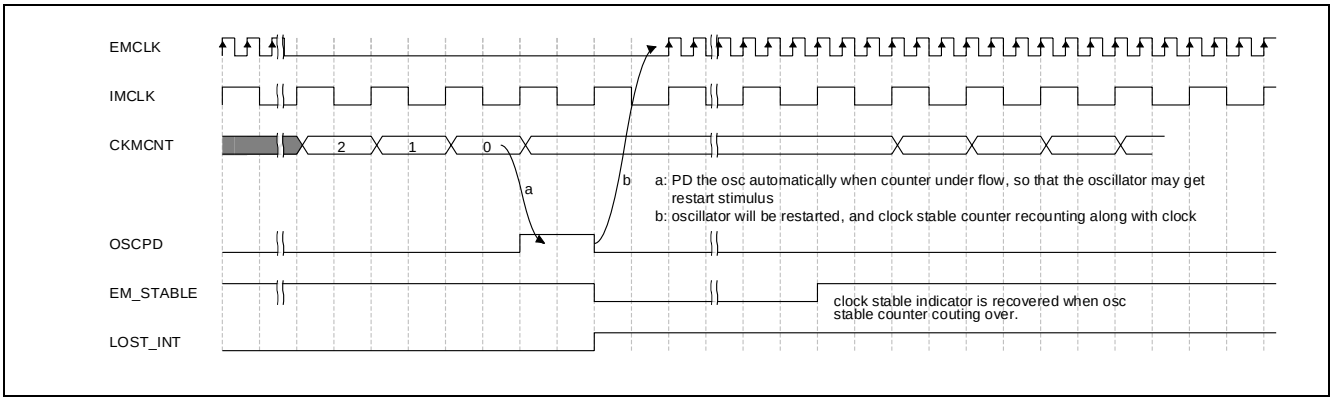


Figure 5-6 EMOSC 失效监测

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。

5.2.2.5 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。

5.2.3 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略低(SLEEP 模式不受影响)。

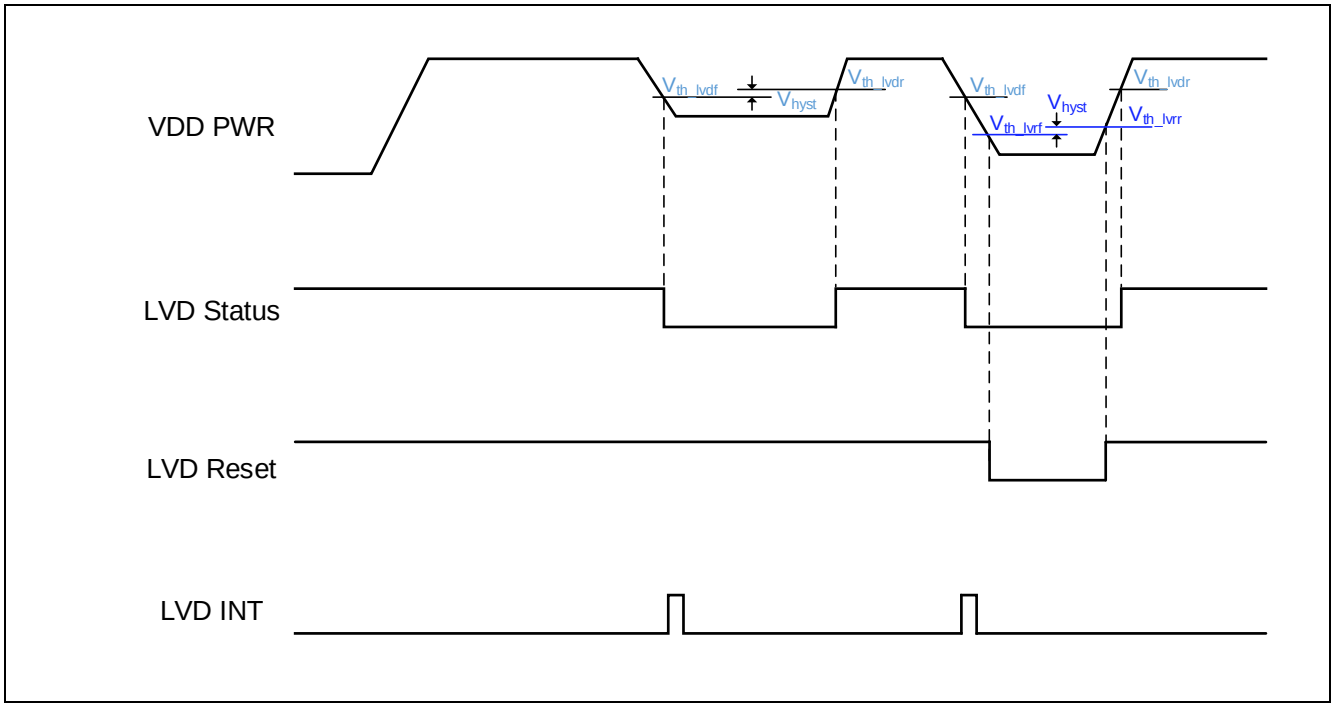


Figure 5-7 LVD 工作时序图

5.2.4 低功耗模式及唤醒

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

系统支持的低功耗模式有三种：

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

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

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

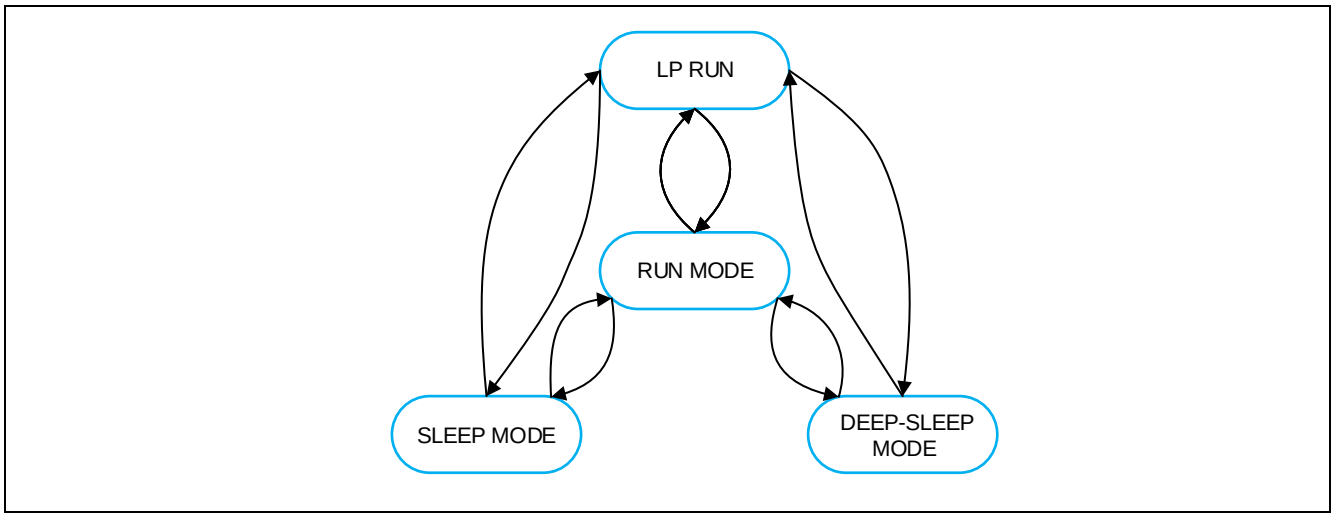


Figure 5-8 模式切换示意图

在不同模式间的切换和唤醒条件可以参考如下表格。

Table 5-3 低功耗模式总结

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

			No effect on other clock
DEEP-SLEEP	STOP Instruction	LVD, EXI, IWDT TKE 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 5-4 在不同工作模式下的功能可用性

PERIPHERAL	RUN	LP RUN	SLEEP	DEEP SLEEP	
				—	WAKEUP
CPU	Y	Y	—	—	—
FLASH	Y	O(2)	—	—	—
SRAM	Y	Y	Y	Y	—
IMOSC	O	O	O	O(3)	—
ISOSC	O	O	O	O(3)	—
EMOSC	O	O	O	O(3)	—
Clock Monitor	O	O	O	O	—
UART	O	O	O	—	—
I2C	O	O	O	—	—
SPI	O	O	O	—	—
ADC	O	O	O	—	—
TIMER	O	O	O	—	—
TC3	O	O	O	O	O
IWDT	O	O	O	O	O
CORET	O	O	O	—	—
EXI	O	O	O	O	O
TKEY	O	O	O	O	O

2. 图例：Y = Yes（Enable 有效），O = Optional（可配置），— = Not available（不可用）。

3. 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）。

4. 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。

5.2.4.1 调试模式

在调试模式下，当系统进入低功耗模式时，在功能上可以调试模式的切换，但是此时系统并没有真正进入低功耗模式，系统时钟在此时仍旧保持工作，以确保调试模式不会因为进入低功耗模式而退出。在调试模式下进入低功耗模式后（执行 DOZE 或者 STOP 指令后），系统将挂起直到被唤醒，唤醒后系统将继续运行直到断点被检测到。若程序调试断点设置在 DOZE 或者 STOP 指令上，则程序在唤醒后，停止在 DOZE 或者 STOP 指令后一条指令处。

在低功耗模式下，若调试接口没有关闭（GPIO 的 AF 功能设置为调试功能时），则任何来自调试器的连接尝试

都会将系统从低功耗模式唤醒。

5.2.4.2 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。

5.2.4.3 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，找到不影响应用的最低功耗模式。不同的低功耗模式有不同的驱动能力，功耗越低，驱动能力越弱。

- **退出 LP RUN 模式的方式：**

退出 LP RUN 模式的方式可以参考如下步骤进行：

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

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

3）切换系统频率到目标频率。

5.2.4.4 睡眠模式（SLEEP MODE）

在 SLEEP 模式下，系统控制器将挂起 CORE 模块的时钟（CPU 将不会工作）。缺省模式下，所有的逻辑外设控制时钟（PCLK）不会被停止，进入模式前被使能的外设将继续工作，且 IO 工作不受影响，例如 GPIO 上的 TIMER 输出在进入 SLEEP 前被打开，则进入 SLEEP 后，该 IO 仍旧可以正常输出。但是通过 GCDR[IDLE_PCLK] 控制位可以设置在 SLEEP 模式下也停止 PCLK。如果 SLEEP 模式下的 PCLK 被禁止，则需要注意外设在没有 PCLK 时可能不能工作，从而将不能正常产生唤醒事件。

在 SLEEP 模式下，所有振荡器的工作状态不会做任何改变。SLEEP 模式相比于 DEEP-SLEEP 模式，由于不存在时钟源的使能切换，有更快的唤醒响应时间， 可以在系统应用需要快速唤醒， 并对功耗有一定要求的应用中采用。

任何外设事件或者中断都可以触发系统从该模式退出。

Table 5-5 SLEEP 模式总结

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

5.2.4.5 深度睡眠模式（DEEP-SLEEP MODE）

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

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

需要注意：

1） 由于 DEEP-SLEEP 模式下，为获得最大限度的节能效果，内部参考电压源将切换到低功耗模式，若在进入模式前，使能 LVD 功能，可能对 LVD 在 DEEP-SLEEP 模式下的精度有一定影响。

2） IWDT 使能后，在 DEEP-SLEEP 模式下不会被自动关闭。此时如果系统处于 DEEP-SLEEP 的时间超过 IWDT 的溢出时间，IWDT 仍然会触发复位。可以根据应用需要进模式前关闭 IWDT，调整 IWDT 溢出时间或考虑

使用 WWDT。

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 5-6 可以唤醒 DEEP-SLEEP 的中断源

Peripheral	Event
GPIO	EXI interrupt of each GPIO
IWDT	Alert interrupt
TC3	Enabled Interrupt
LVD	Enabled Interrupt
TKEY	Enabled Interrupt

Table 5-7 DEEP-SLEEP 模式总结

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

5.2.4.6 低功耗唤醒和中断服务

当系统进入低功耗模式后，只有通过中断进行唤醒。系统唤醒经过两个阶段，

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

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

CPU 退出低功耗模式后，根据 NVIC 是否使能了该中断的中断服务跳转，系统可以立即进入该中断的中断服务程序或者不响应中断而继续 DOZE 或 STOP 指令后的代码。必须注意，当中断发生后，即使在 NVIC 中没有使能该

中断的中断服务跳转，也会导致该中断在 NVIC 中的 pending 标志被置位。所以必须通过软件清除该标志，否则接下来的 DOZE 或 STOP 指令将不会进入低功耗模式。具体可以参考 CPU 相关文档或者 INTERRUPT 章节。

初始化的时间为可以用下面的公式进行估计：

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

其中

- T_{imosc_stable} 为 IMOSC 的稳定时间。具体来说，IMOSC 的稳定时间大约为 64 个 IMCLK 周期（5.56MHz 时，一个 IMCLK 周期为 180ns）。如果在进入 DEEP-SLEEP 前，IMOSC 已经使能，则该时间可以忽略；
- T_{clk_sw} 为系统自动切换控制时钟的时间，为 2 个 IMCLK 周期。如果在进入 DEEP-SLEEP 前，系统时钟即为 IMOSC，则该时间可以忽略；
- T_{emo_dis} 为 EMOSC 的关闭时间，当 EMOSC 频率大于 IMOSC 时，为 2 个 IMCLK 周期，反之为 2 个 EMCLK 周期。如果在进入 DEEP-SLEEP 前，EMOSC 没有使能，则该时间可以忽略；
- T_{hfo_dis} 为 HFOSC 的关闭时间，为 2 个 IMCLK 周期。如果在进入 DEEP-SLEEP 前，HFOSC 没有使能，则该时间可以忽略；
- T_{iso_dis} 为 ISOSC 的关闭时间，为 2 个 ISCLK 周期。如果在进入 DEEP-SLEEP 前，ISOSC 没有使能，则该时间可以忽略；
- T_{imo_dis} 为 IMOSC 的关闭时间，为 2 个 IMCLK 周期。

5.2.5 功耗管理和优化

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

VOS 使能时，可以通过软件调整 LDO 的输出特性，包括电压和响应速度，以及参考电压源的精度。在不同工作模式和工作条件（如较低的 RUN 模式工作频率）下，可以通过选择合适的 VOS 配置以实现更低的系统整体功耗。

VOS 设置过程如下：

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

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

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

5.2.6 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 5-8 处理器复位信号源表

Reset Soure	Description
WWDT	WWDT产生的系统复位
PLLUNLOCK	PLLUNLOK导致的系统复位
CPUFAULT	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）
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（对CPU中MEXSTATUS寄存器的SOFT_RST位写入2b’01或者2b’10）。
EMCKM	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。
IWDT	由内部独立看门狗电路产生的复位信号。
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）。
LVR	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。
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。

5.2.7 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。

5.2.7.1 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）。

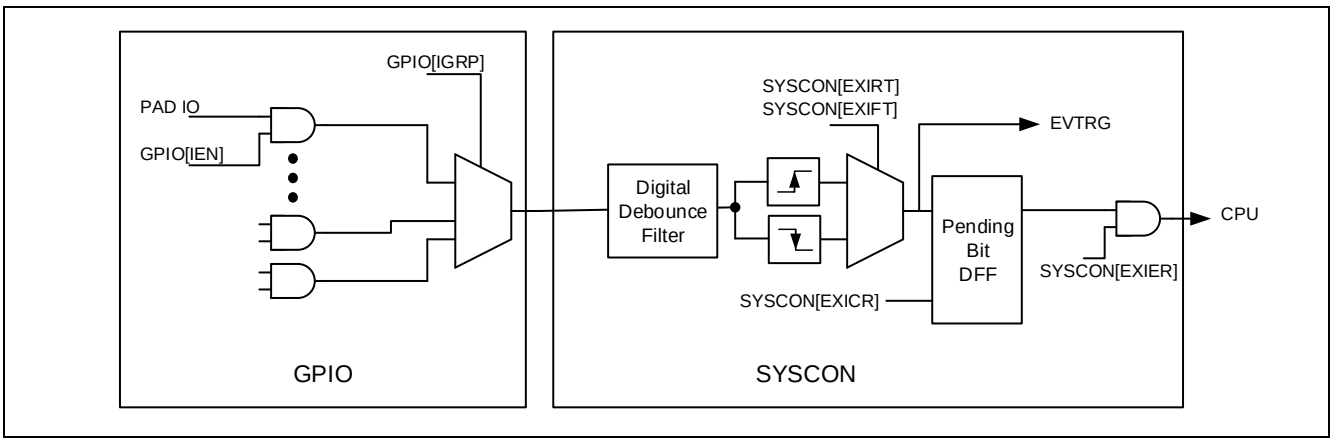


Figure 5-9 外部中断结构示意图

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

- 关闭 EXI 的中断
- 配置 EXI
- 对 Pending Bit 进行一次软件清除
- 使能 EXI

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

如何配置和使能外部中断？

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

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

第二栏中的 EXIx 表示 SYSCON 中的中断源。它们和 EXI GROUP 是一一对应的关系。外部中断的中断线控制在 SYSCON 中通过一组寄存器实现。EXIER/EXIDR 寄存器可以使能或者关闭指定的外部中断信号线，当前的中断线设置状态可以通过 EXIMR 寄存器获得。外部中断线的 pending 状态可以通过 EXICR 寄存器查询，对 EXICR 寄存器相应位写入‘1’，可以清除该中断线的 pending 状态。中断的原始 pending 状态可以通过 EXIRS 寄存器查询。外部中断可以支持软件触发，通过设置 EXIAR 可以在没有外部硬件触发的情况下，直接由软件触发外部中断。

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

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

Table 5-9 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。

5.2.7.2 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。去抖深度为 3 个采样周期。只有在所有采样时刻采样到相同的电平，才会认为该电平有效，否则会被视为噪声滤除。

5.2.8 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。

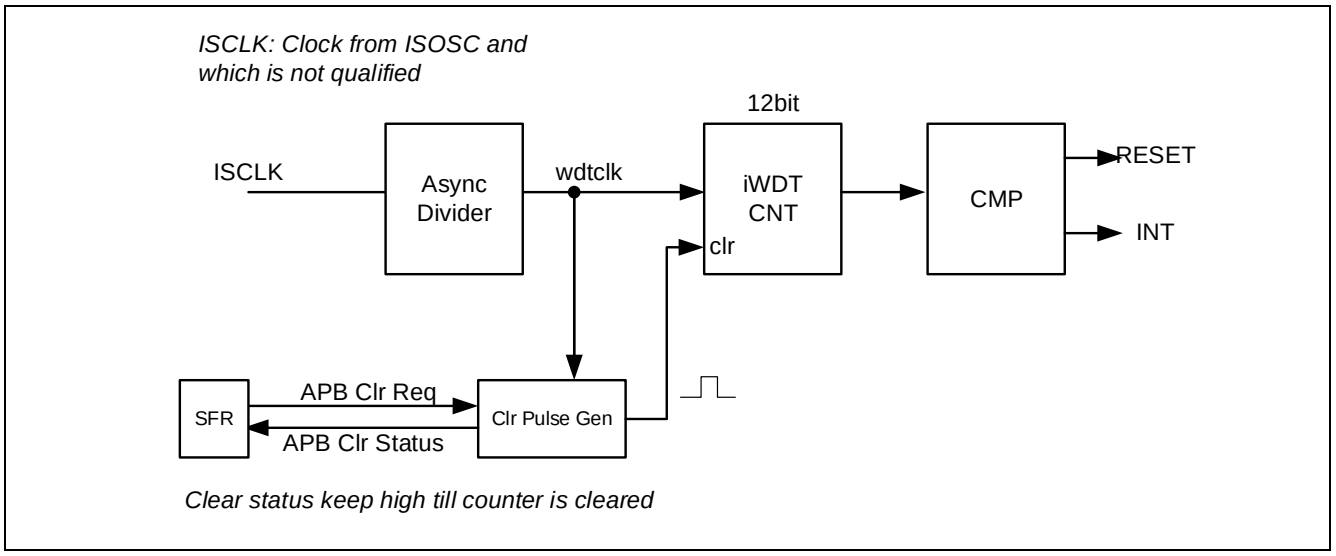


Figure 5-10 IWDT 结构框图

IWDT 的缺省状态可以通过 Flash 内部的 User Option 设置。在软件中，可以通过设置 IWDEDR 寄存器中的 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[CLR] 写入 0x5A 实现。只有当清除 CNT 操作发生时，计数器值才会被更新到最新的设置值，所以

在更新了 IWDCNT 寄存器后，需要软件清除一次 CNT，使得内部计数器及时更新到最新的配置。软件清除需要在 IWDT 启动以后执行（通过查询 IWDCR[BUSY]控制位确认 IWDT 是否已经启动）。

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

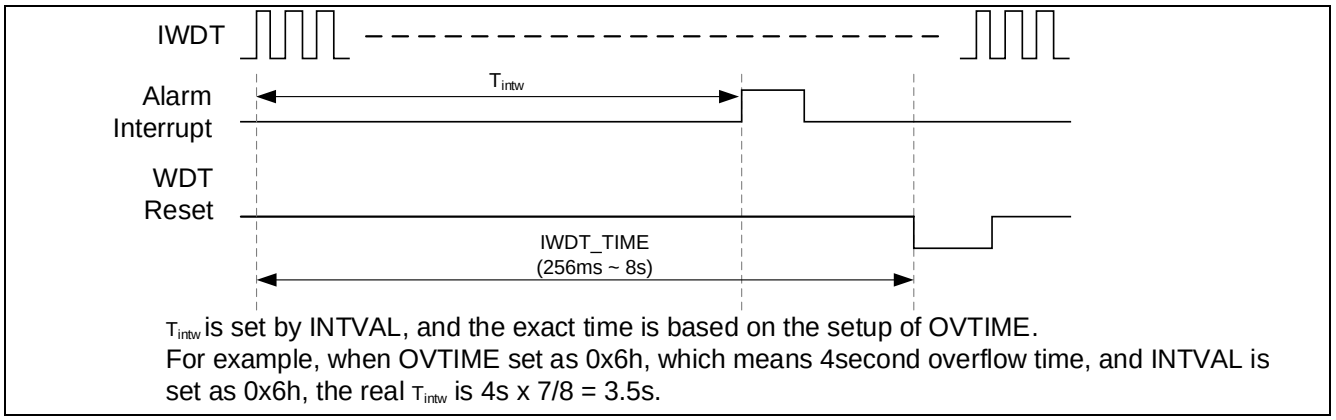


Figure 5-11 IWDT 计数器溢出和报警中断

IWDT 还支持报警功能。在计数器计数过程中，当计数值达到 IWDCR[INTVAL] 设置值时，会产生一个中断信号。IWDCR[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 IWDCR[OVTIME] 设置看门狗定时溢出时间为 4 秒，如果 IWDCR[INTVAL]设置为 3’b001，则表示在计数器计时到 4 x 2/8 = 1 秒时，系统会产生一个报警中断。通过此报警中断可以在低功耗模式下及时唤醒系统并进行计数器清除，以防止看门狗复位。

5.2.9 IO重定义

芯片最多提供两个预设的 GPIO GROUP， 分别为 GROUP0 和 GROUP1。所有支持 IOMAP 的管脚在数据手册管脚功能分配表（Table 2-1）中有相应的标注，如 G0.x，G1.x。这些管脚，除了表格中列举的 AF 功能外，还可以映射为另外 7 种预设的 AF 功能，具体可参考各型号数据手册 Table 2-2。

IO 重定义功能可通过 SYSCON_IOMAP0 和 SYSCON_IOMAP1 进行配置。

5.2.10 系统信息及自定义寄存器

系统中存在几类信息存储寄存器，作为软件获取系统信息的一种途径：

UID：唯一序列码

每个芯片在出厂测试时，都会设置唯一的序列号，该序列号由 96bit 组成。

UREG：自由寄存器

用于保存客户临时信息的寄存器，该寄存器内保存的数据只有在上电复位后才会丢失。这意味着，只要供电在，无论任何其他的复位都不会改变其中的数据。该寄存器无特殊功能，仅为用户提供在应用程序中可以方便地存储和监测不受其他复位影响的临时状态。

5.2.11 SYSCON 中断管理

系统控制器的中断由 6 中断源组成，每个中断源下可选择一个或者多个触发事件作为该中断源的触发信号。一个中断源是由 SYSCON 通用事件产生的通用中断请求源，其控制通过 RISR，MISR，IMCR，IMDR, IMER, ICR 这组控制寄存器进行控制。触发 SYSCON 中断的事件使能选择可以通过 IMCR 或者 IMER / IMDR 寄存器进行设置。IAR 寄存器则提供了通过软件触发中断事件的方法，可用于程序代码中对事件的 debug。

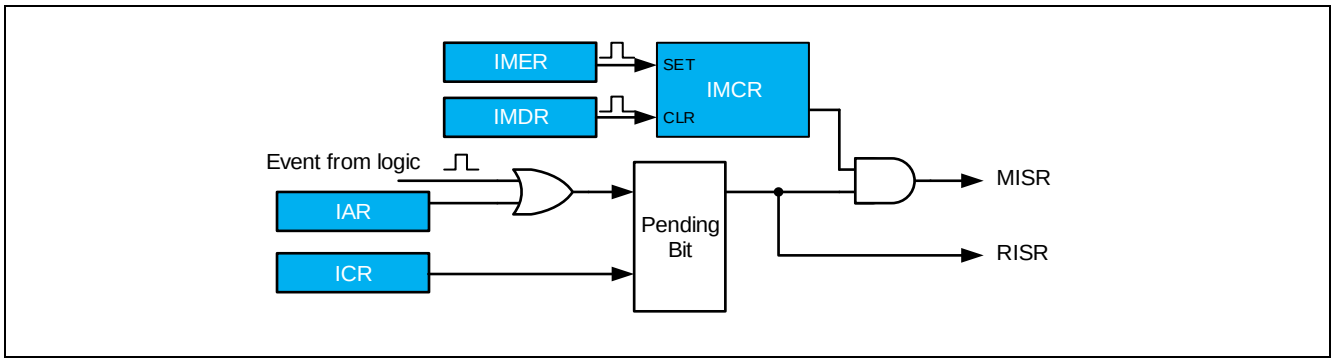


Figure 5-12 SYSCON 通用中断的控制逻辑

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

Table 5-10 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
PLL_ST	Asserted when PLL Clock is stabled
SYSCLK_ST	Asserted when SYSCLK is stabled
IWDT_INT	Asserted when IWDT counter reaches preset interval
LVD_INT	Asserted when power supply falls or rises to preset LVD level
OPL_ERR	Asserted when User Option fails to load at initialization
PLLUL	Asserted when PLL is unlocked
EM_CMFAIL	Asserted when external oscillator monitoring detects failure
EV0TRG	Asserted when event0 happens
EV1TRG	Asserted when event1 happens

EV2TRG	Asserted when event2 happens
EV3TRG	Asserted when event3 happens
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_V4。

5.2.12 触发事件控制

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

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

信号在模块间的流动如下图所示:

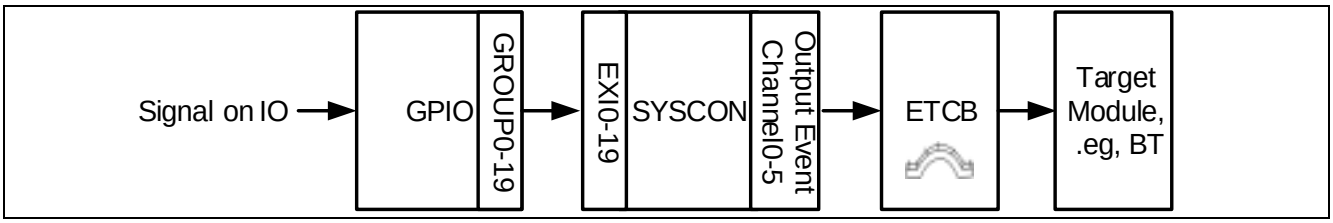


Figure 5-13 Signal Flow

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

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

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

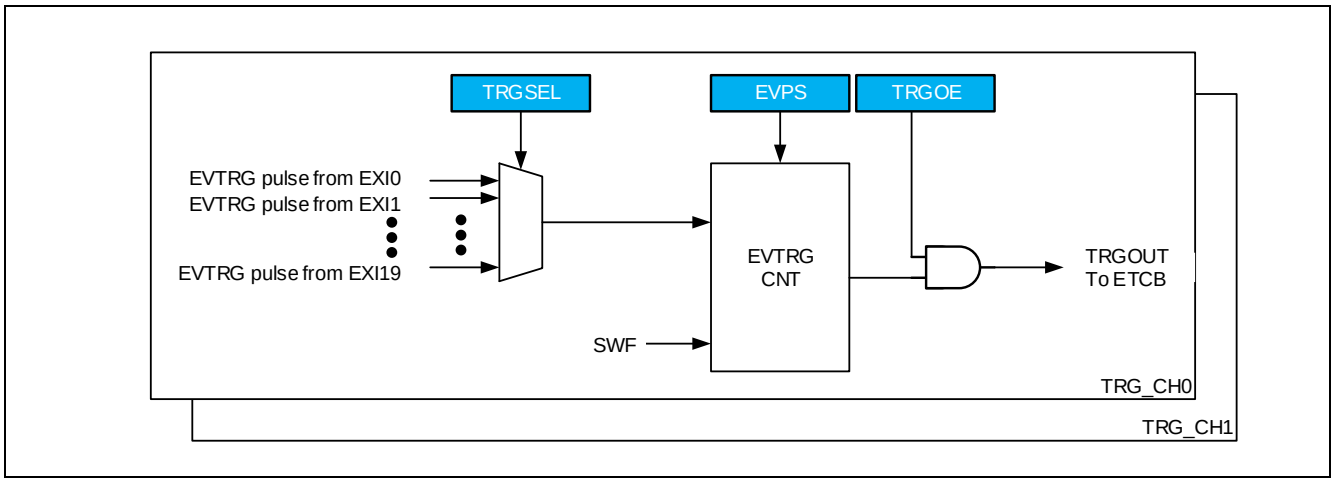


Figure 5-14 SYSCON 外部事件触发控制逻辑

注意：当使用通道 4 或通道 5 时，PCLK 必须保持 1 分频。

5.2.13 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，需要设置调试控制寄存器 DBGCR[DBG_UNLOCK] 为 0x5a。

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。

5.3 寄存器说明

5.3.1 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_SCLKCR	0x01C	系统时钟控制寄存器	0x00000400
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	0x00000001
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_PLLCR	0x044	PLL控制寄存器	0x00000000
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	0x00000000
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]	0x00000000
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	0x0C0	GPIO分组0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_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	0x00000000

5.3.2 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	RSVD		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	R	R	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]	W	SYSCON产生软件复位。效果和CPU通过软复位指令产生软件复位一致。 0h：没有效果。 1h：执行软件复位。
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。

5.3.3 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		PLL	RSVD	EMOSC	RSVD	IMOSC	ISOSC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	W	W	R	R	W	R	W	W	W	R	W	W	R	R	R	R	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。
PLL	[5]	R	使能或禁止PLL。 0h：无效。 1h：使能或禁止PLL。
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。

5.3.4 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		PLL	RSVD	EMOSC	RSVD	IMOSC	ISOSC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	W	W	R	R	W	R	W	W	W	R	W	W	R	R	W	R	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。
PLL	[5]	W	使能或禁止PLL。 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。

5.3.5 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		PLL	RSVD	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。
PLL	[5]	R	PLL工作状态。 0h：PLL被关闭。 1h：PLL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。

5.3.6 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		PLL	RSVD	EMOSC	RSVD	IMOSC	ISOSC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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。
PLL	[5]	R	PLL稳定状态。 0h：PLL未稳定。 1h：PLL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。			

5.3.7 SYSCON_SCLKCR(系统时钟控制寄存器)

Address = Base Address+ 0x01C, Reset Value = 0x00000400

31	30	29	28	27	26	25	24	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	1	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	R	R	R	R	RW	RW	RW	RW	R	R	R	R	R	RW	RW	RW

Name	Bit	Type	Description
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：8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：IMCLK作为系统工作时钟源。 1h：EMCLK作为系统工作时钟源。 2h：保留 3h：PLL作为系统时钟源。 4h：ISCLK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。			

5.3.8 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。			

5.3.9 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									I2C	RSVD					SPI	RSVD			LED	RSVD		UART1	UART0	ETCB	TKEY	RSVD	ADC	RSVD			IFC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	W	R	R	R	R	R	W	R	R	R	W	R	R	W	W	W	W	R	W	R	R	R	W

Name	Bit	Type	Description
I2C	[22]	W	
SPI	[16]	W	
LED	[12]	W	
UART1~UART0	[9:8]	W	
ETCB	[7]	W	
TKEY	[6]	W	
ADC	[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。

5.3.10 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									I2C	RSVD					SPI	RSVD						UART1	UART0	ETCB	RSVD		ADC	RSVD			IFC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	W	R	R	R	R	R	W	R	R	R	R	R	R	W	W	W	R	R	W	R	R	R	W

Name	Bit	Type	Description
I2C	[22]	W	
SPI	[16]	W	
UART1~UART0	[9:8]	W	
ETCB	[7]	W	
ADC	[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。

5.3.11 SYSCON_PCSR0(外设时钟状态寄存器0)

Address = Base Address+ 0x030, 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									I2C	RSVD					SPI	RSVD						UART1	UART0	ETCB	RSVD		ADC	RSVD			IFC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
I2C	[22]	R	
SPI	[16]	R	
UART1~UART0	[9:8]	R	
ETCB	[7]	R	
ADC	[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。			

5.3.12 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															GPTB	GPTA	BT3	BT2	BT1	BT0	TC3	RSVD		WWDT	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	R	R	W	R	R	R	R	R	R	R

Name	Bit	Type	Description
GPTB	[16]	W	
GPTA	[15]	W	
BT3~BT0	[14:11]	W	
TC3	[10]	W	
WWDT	[7]	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。

5.3.13 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															GPTB	GPTA	BT3	BT2	BT1	BT0	TC3	RSVD		WWDT	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	R	R	W	R	R	R	R	R	R	R

Name	Bit	Type	Description
GPTB	[16]	W	
GPTA	[15]	W	
BT3~BT0	[14:11]	W	
TC3	[10]	W	
WWDT	[7]	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。

5.3.14 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															GPTB	GPTA	BT3	BT2	BT1	BT0	TC3	RSVD		WWDT	RSVD						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
GPTB	[16]	R	
GPTA	[15]	R	
BT3~BT0	[14:11]	R	
TC3	[10]	R	
WWDT	[7]	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。			

5.3.15 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

5.3.16 SYSCON_PLLCR(PLL控制寄存器)

Address = Base Address+ 0x044, 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																UNLCKRST	RSVD													CKSEL	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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	R	R	R	R	R	R	R	R	R	RW	RW

Name	Bit	Type	Description
UNLCKRST	[15]	RW	PLL 失锁复位控制位。 0h：在PLL失锁时，禁止产生系统复位。 1h：在PLL失锁时，产生系统复位。
CKSEL	[1:0]	RW	PLL时钟输入源选择控制位 0h：Disabled 1h：IMOSC 选择作为时钟源 2h：Disabled 3h：EMOSC 选择作为时钟源

5.3.17 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 7h：LVDIN_1V 注1：除LVDIN_1V外比较电压为VDD电压和选择的电压进行比较。 注2：当选择LVDIN_1V时，应输入引脚选择LVDIN复用功能，该引脚电压和1V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。			

5.3.18 SYSCON_CLCR(内部振荡器调整寄存器)

Address = Base Address+ 0x050, 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
ISO_TRM								IMO_TRM								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	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

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。

5.3.19 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 SLEEP控制 0h: disable 1h: enable: Main regulator工作SLEEP(低速)模式，低功耗。 bit3: 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 SLEEP控制 0h: disable 1h: enable: Main regulator工作SLEEP(低速)模式，低功耗。 bit3: 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 SLEEP控制 0h: disable 1h: enable: Main regulator工作SLEEP(低速)模式，低功耗。 bit3: 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。
VOSEN	[3:0]	RW	VOS模式使能控制。（1） 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）在低功耗模式下，不允许切换模式配置。改变模式配置，必须在RUN模式下进行。 2）在各种模式下，BGR的使能可独立配置。 3）不同的运行模式中，LP0~2的低功耗策略各不相同。从LP0到LP2功耗递增，驱动能力也递增。调试的时候可以从最低功耗模式开始尝试，直到找到系统可以正常运行的模式。			

5.3.20 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。			

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

Address = Base Address+ 0x064, 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								EMCKM_DUR			RSVD		EXIFLTCKS		EXIFLTEN	EFL_LPMD	CLODIV			CLOMX				RSVD						IMO_FSEL	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	RW	RW	RW	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	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：PLL输出。 7h：PCLK输出。 8h：HCLK输出。 9h：IWDTCLK输出。 Dh：SYSCLK输出。 其他：保留。
IMO_FSEL	[1:0]	RW	IMOSC频率选择。 0h：24MHz 1h：保留 2h：保留 3h：保留

5.3.22 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				CPUFTRST	EXIRSTEN	IWDTEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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。
CPUFTRST	[2]	R	CPU Fault时产生硬件复位User Option设置状态查询。 0h：禁止自动复位。 1h：使能自动复位。
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User Option设置状态查询。 0h：缺省关闭看门狗。 1h：缺省打开看门狗。

5.3.23 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																		TC3_WKEN	TCH_WKEN	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	RW	RW	RW	R	R	RW	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
TC3_WKEN	[13]	RW	TC3中断唤醒DEEP-SLEEP使能控制位。 0h：禁止TC3中断唤醒DEEP-SLEEP。 1h：使能TC3中断唤醒DEEP-SLEEP。
TCH_WKEN	[12]	RW	TOUCH中断唤醒DEEP-SLEEP使能控制位。 0h：禁止TOUCH中断唤醒DEEP-SLEEP。 1h：使能TOUCH中断唤醒DEEP-SLEEP。
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。			

5.3.24 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	RSVD			PLLUL	OPL_ERR	RSVD		LVD_INT	RSVD		IWDT_INT	SYSCLK_ST	RSVD	PLL_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	R	R	R	W	W	R	R	W	R	R	W	W	R	W	R	R	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
PLLUL	[15]	W	PLL UNLOCK中断
OPL_ERR	[14]	W	Option初始化配置加载失败中断。
LVD_INT	[11]	W	LVD中断。
IWDT_INT	[8]	W	IWDT中断。
SYSCLK_ST	[7]	W	SYSCLK时钟稳定中断。
PLL_ST	[5]	W	PLL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。

5.3.25 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		PLLUL	OPL_ERR	RSVD		LVD_INT	RSVD		IWDT_INT	SYSCLK_ST	RSVD	SYPLL	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	W	W	R	R	W	R	R	W	W	R	W	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
PLLUL	[15]	W	PLL UNLOCK
OPL_ERR	[14]	W	Option初始化配置加载失败中断。
LVD_INT	[11]	W	LVD中断。
IWDT_INT	[8]	W	IWDT中断。
SYSCLK_ST	[7]	W	SYSCLK时钟稳定中断。
SYPLL	[5]	W	PLL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。

5.3.26 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		PLLUL	OPL_ERR	RSVD		LVD_INT	RSVD		IWDT_INT	SYSCLK_ST	RSVD	SYPLL	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	RW	RW	R	R	RW	R	R	RW	RW	R	RW	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。
PLLUL	[15]	RW	PLLUL校验失败中断。 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。
LVD_INT	[11]	RW	
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。
SYPLL	[5]	RW	PLL时钟稳定中断。

			0h：禁止中断。 1h：使能中断。
EMOSC_ST	[3]	RW	
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。

5.3.27 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	RSVD		LVD_INT	RSVD		IWDT_INT	SYSCLK_ST	RSVD			EMOSC_ST	RSVD	IMOSC_ST	ISOSC_ST
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	W	R	R	R	R	R	R	W	W	W	W	W	R	R	R	W	R	R	W	R	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。
LVD_INT	[11]	W	LVD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。

5.3.28 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		PLLUL	OPL_ERR	RSVD		LVD_INT	RSVD		IWDT_INT	SYSCLK_ST	RSVD	PLL_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	W	W	R	R	W	R	R	W	W	R	W	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
PLLUL	[15]	W	PLL UNLOCK中断
OPL_ERR	[14]	W	Option初始化配置加载失败中断。
LVD_INT	[11]	W	LVD中断。
IWDT_INT	[8]	W	IWDT中断。
SYSCLK_ST	[7]	W	SYSCLK时钟稳定中断。
PLL_ST	[5]	W	PLL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。

5.3.29 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		PLLUL	OPL_ERR	RSVD		LVD_INT	RSVD		IWDT_INT	SYSCLK_ST	RSVD	PLL_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
PLLUL	[15]	R	PLL UNLOCK中断
OPL_ERR	[14]	R	Option初始化配置加载失败中断。
LVD_INT	[11]	R	LVD中断。
IWDT_INT	[8]	R	IWDT中断。
SYSCLK_ST	[7]	R	SYSCLK时钟稳定中断。
PLL_ST	[5]	R	PLL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。

5.3.30 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		PLLUL	OPL_ERR	RSVD		LVD_INT	RSVD		IWDT_INT	SYSCLK_ST	RSVD	PLL_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
PLLUL	[15]	R	PLL UNLOCK中断
OPL_ERR	[14]	R	Option初始化配置加载失败中断。
LVD_INT	[11]	R	LVD中断。
IWDT_INT	[8]	R	IWDT中断。
SYSCLK_ST	[7]	R	SYSCLK时钟稳定中断。
PLL_ST	[5]	R	PLL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。

5.3.31 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	RSVD		PLLUNLOCK	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	R	R	RW	RW	RW	RW	RW	R	RW	R	RW	RW	RW

Name	Bit	Type	Description
WWDT	[13]	RW	WWDT复位
PLLUNLOCK	[10]	RW	PLL失锁产生的系统复位
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	LVD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。			

5.3.32 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]		

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

5.3.33 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]		
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			

5.3.34 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]		

5.3.35 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]		

5.3.36 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]		

5.3.37 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]		

5.3.38 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]		

5.3.39 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]		

5.3.40 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	SHORT
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	1	0	0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	R	R	R	R	RW	RW	RW	RW	R	R	R	RW	RW	RW	R	RW

Name	Bit	Type	Description
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：调试禁止 1h：调试使能
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：3.096s。 6h：4.12s。 7h：8.196s。
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。
SHORT	[0]	RW	SHORT工作模式，该模式用于缩短IWDT的溢出时间。正常使用时，保持该位为零。 0h：禁止SHORT模式。 1h：使能SHORT模式。

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。

5.3.41 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。

5.3.42 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。

5.3.43 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~CFGVAL0	[31:0]	RW	IO GROUP0中对应GPIO的功能选择。 具体对应管脚参考产品数据手册管脚配置章节 Table 2-1。 具体配置数值参考产品数据手册管脚配置章节 Table 2-2。

5.3.44 SYSCON_IOMAP1(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~CFGVAL0	[31:0]	RW	IO GROUP1中对应GPIO的功能选择。 具体对应管脚参考产品数据手册管脚配置章节 Table 2-1。 具体配置数值参考产品数据手册管脚配置章节 Table 2-2。
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。			

5.3.45 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。

5.3.46 SYSCON_UID1(UID寄存器1)

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

5.3.47 SYSCON_UID2(UID寄存器2)

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

5.3.48 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
RSVD										EFLR_CTL		EFLR_PD		EFL_PD		TPWRCV_SLP								TPWRCV_DSL							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
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。
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。
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。			

5.3.49 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
CNTxCLR						TRGxOE						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

Name	Bit	Type	Description
CNTxCLR	[31:28]		TRGEVxCNT软件清除 0h：无效 1h：EVxCNT重置为零
	[27:26]		
TRGxOE	[25:20]		外部触发端口TRGOUTx使能。 0h：禁止触发输出。 1h：允许触发输出。
TRGSEL5	[19:18]		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]		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]		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]		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]		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]		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。
注意：当使用通道4或通道5时，PCLK必须保持1分频。			

5.3.50 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

5.3.51 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为16位寄存器。			

5.3.52 SYSCON_UREG1(32位用户寄存器)

Address = Base Address+ 0x104, 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。

5.3.53 SYSCON_UREG2(16位用户寄存器)

Address = Base Address+ 0x108, 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																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
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
UREG	[15:0]	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。

5.3.54 SYSCON_DBGCR(调试控制寄存器)

Address = Base Address+ 0x128, 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																								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	0	0	0	0	0	0	0
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： 5ah：调试管脚可设置为其他任意功能 others：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。

6 GPIO

6.1 概述

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

- Port A0: 16 位输入/输出端口, PA0.0 ~ PA0.15
- Port B0: 6 位输入/输出端口, PB0.0 ~ PB0.5

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

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

6.1.1 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（仅APT32F1043支持）

6.1.2 管脚描述

Table 6-1 管脚描述

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

注意：

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

6.2 功能描述

6.2.1 电路图

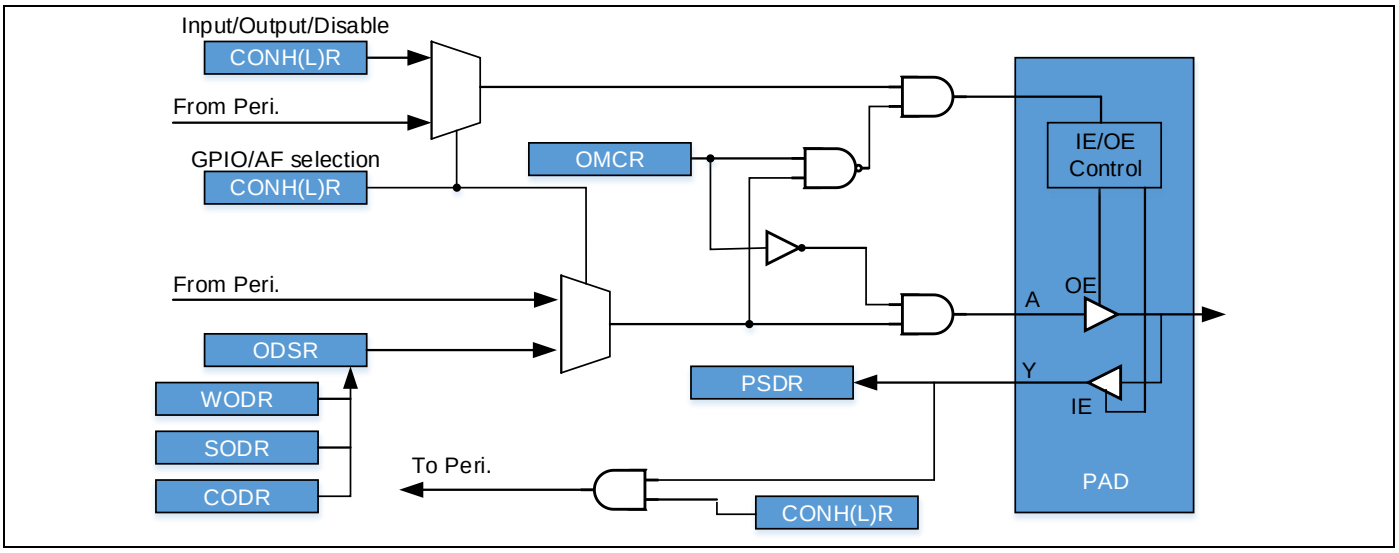


Figure 6-1 GPIO原理图

6.2.2 工作原理

6.2.2.1 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。

6.2.2.2 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。

6.2.3 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。

6.2.4 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。

6.2.5 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。

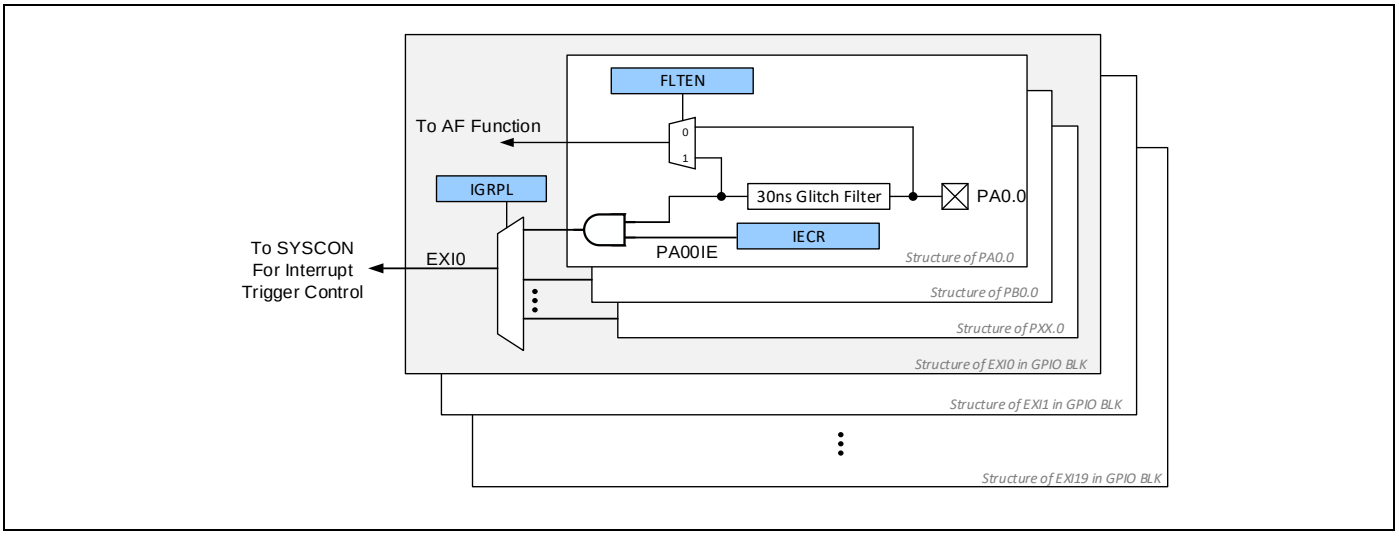


Figure 6-2 GPIO外部中断原理图

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。

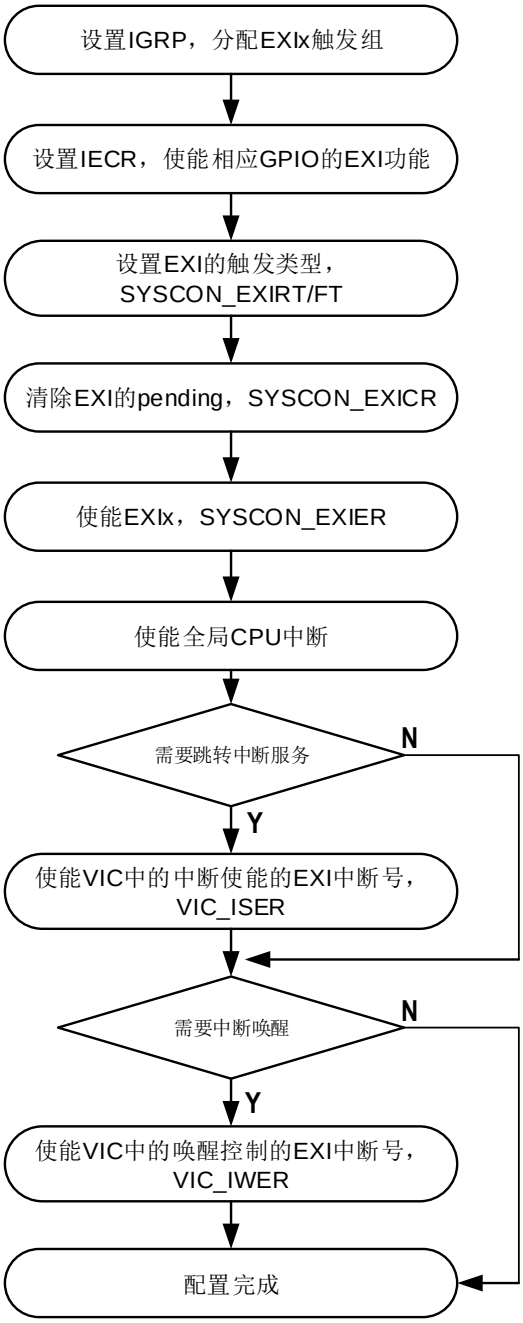


Figure 6-3 GPIO外部中断配置流程

6.3 寄存器说明

6.3.1 寄存器表

Base Address of GPIOA: 0x60000000

Base Address of GPIOB: 0x60002000

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。			

6.3.2 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

6.3.3 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）。

6.3.4 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~P0	[15:0]	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。

6.3.5 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~P0	[15:0]	W	
端口x 输出置1 0h：无效果 1h：相应GPIO管脚的输出数据被置1，高电平 只有当功能模式在寄存器CONLR 或 CONHR 中被设置成GPIO ，输出的数据才是有效的。			

6.3.6 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~P0	[15:0]	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。			

6.3.7 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~P0	[15:0]	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。

6.3.8 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~P0	[15:0]	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。			

6.3.9 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~P0	[15:0]	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。			

6.3.10 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。

6.3.11 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~P0	[31:0]	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。

6.3.12 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~CCM0	[31:16]	RW	
ODP15~ODP0	[15:0]	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。

6.3.13 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~IEN0	[15:0]	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			

6.3.14 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~IEE0	[15:0]	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

6.3.15 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~IED0	[15:0]	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

6.3.16 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 0010: GPIOB0.x 被选中 Other: 保留 ‘x’ 表示组数			

6.3.17 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 0010: GPIOB0.x 被选中 Other: 保留 ‘x’ 表示组数			

6.3.18 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 0h：PD00 被选中 1h：PD01 被选中 …… fh： PD015 被选中 其他：PD00 被选中
GRP18	[11:8]	RW	选择外部中断组18 0h：PC00 被选中 1h：PC01 被选中 …… fh： PC015 被选中 其他：PC00 被选中
GRP17	[7:4]	RW	选择外部中断组17 0h：PB00 被选中 1h：PB01 被选中 …… fh： PB015 被选中 其他：PB00 被选中
GRP16	[3:0]	RW	选择外部中断组16 0h：PA00 被选中 1h：PA01 被选中 …… fh： PA015 被选中 其他：PA00 被选中
说明：如果芯片管脚资源不包含选项，选择无效。			

6.3.19 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_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_B0	[2]	RW	
CLK_A1~CLK_A0	[1:0]	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			

7 事件触发控制器（ETCB）

7.1 概述

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

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

7.1.1 特性

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

7.2 功能描述

7.2.1 模块框图

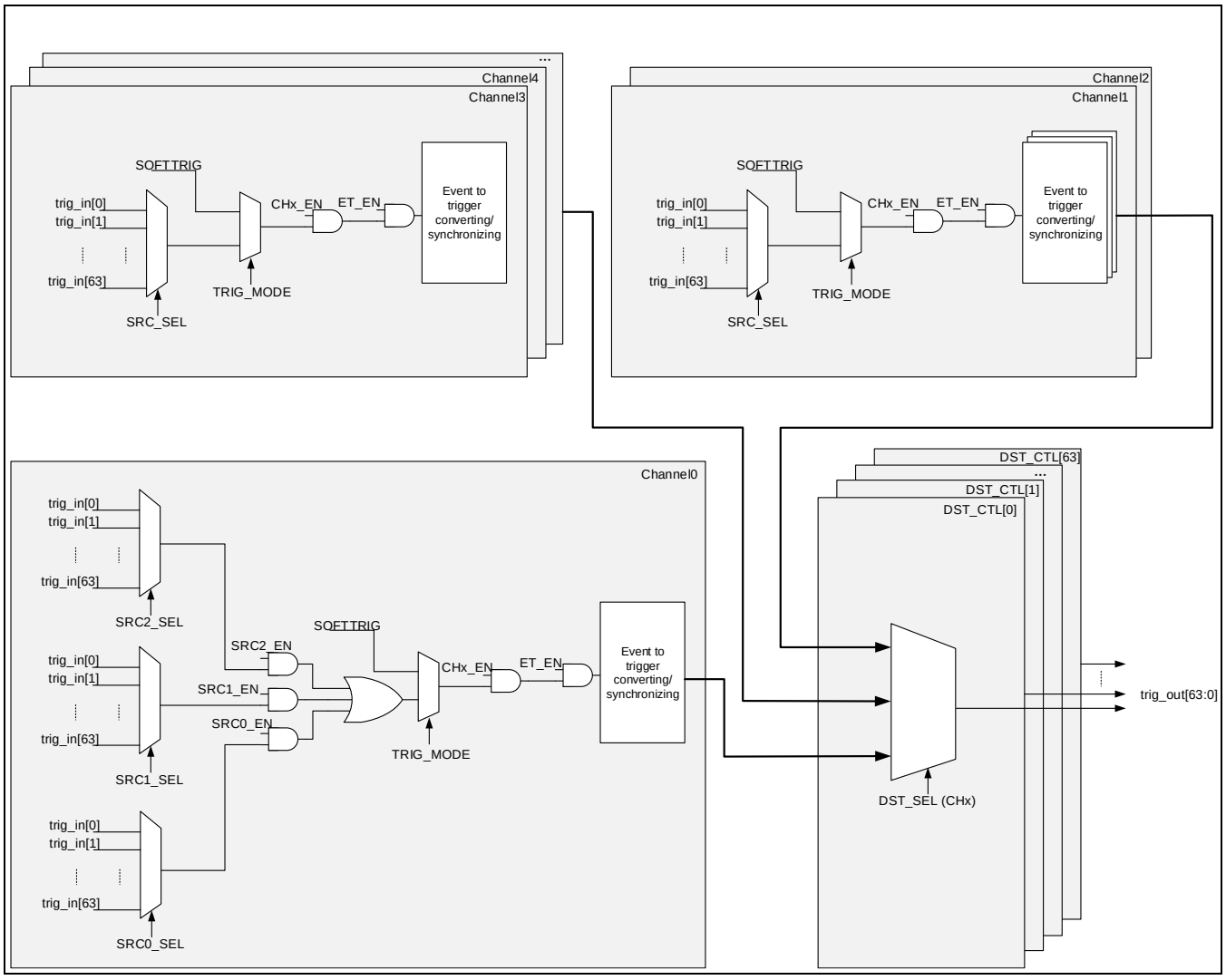


Figure 7-1 ETCB模块框图

7.2.2 主要功能

7.2.2.1 功能描述

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

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

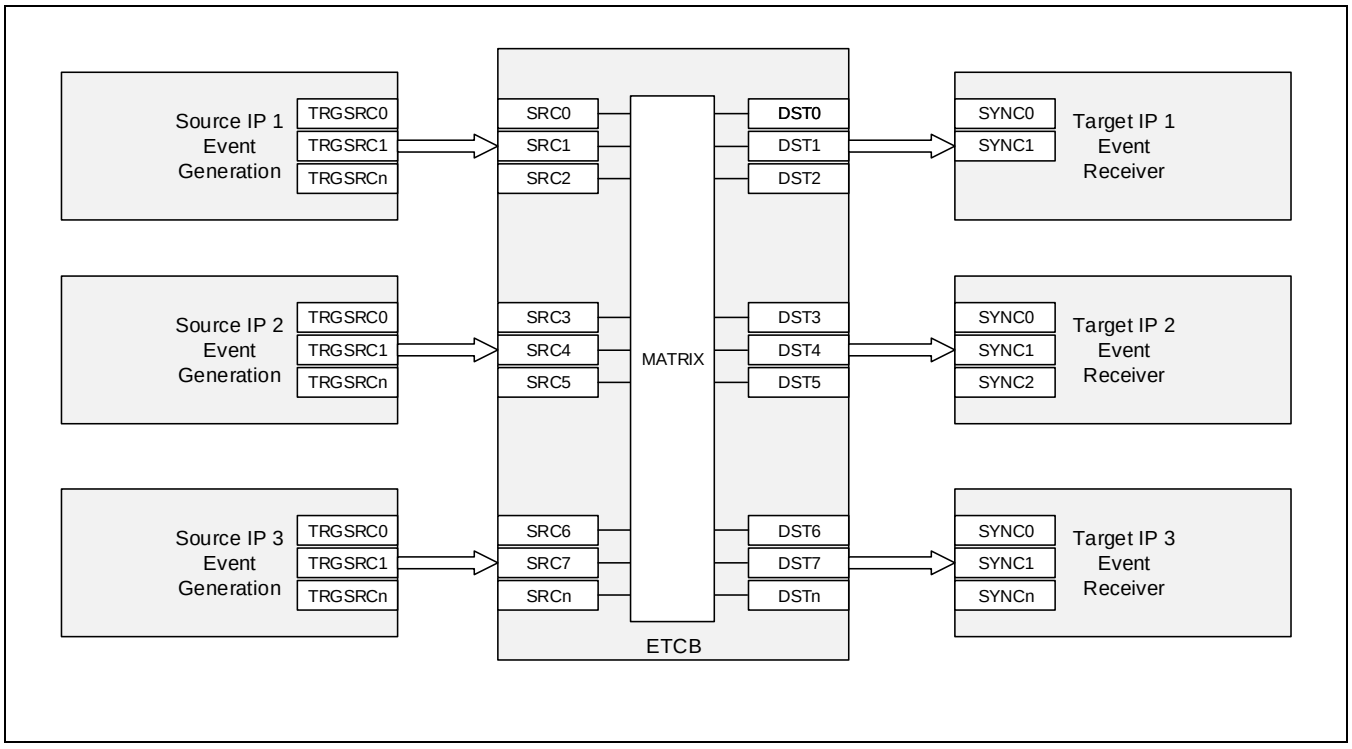


Figure 7-2 IP间通过ETCB事件同步

注意：

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

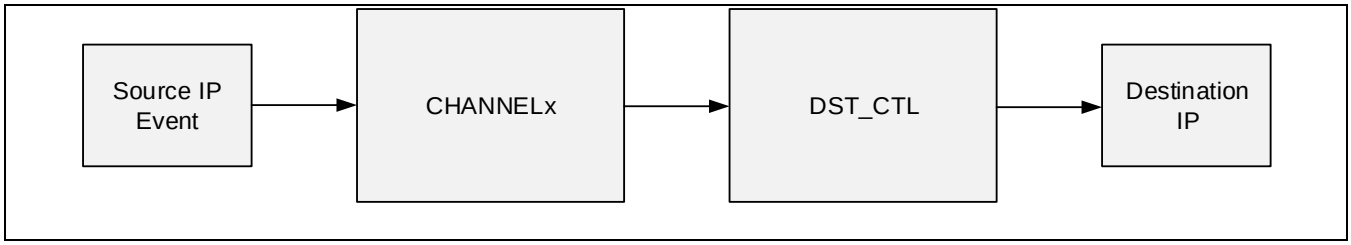


Figure 7-3 单个源触发单个目标

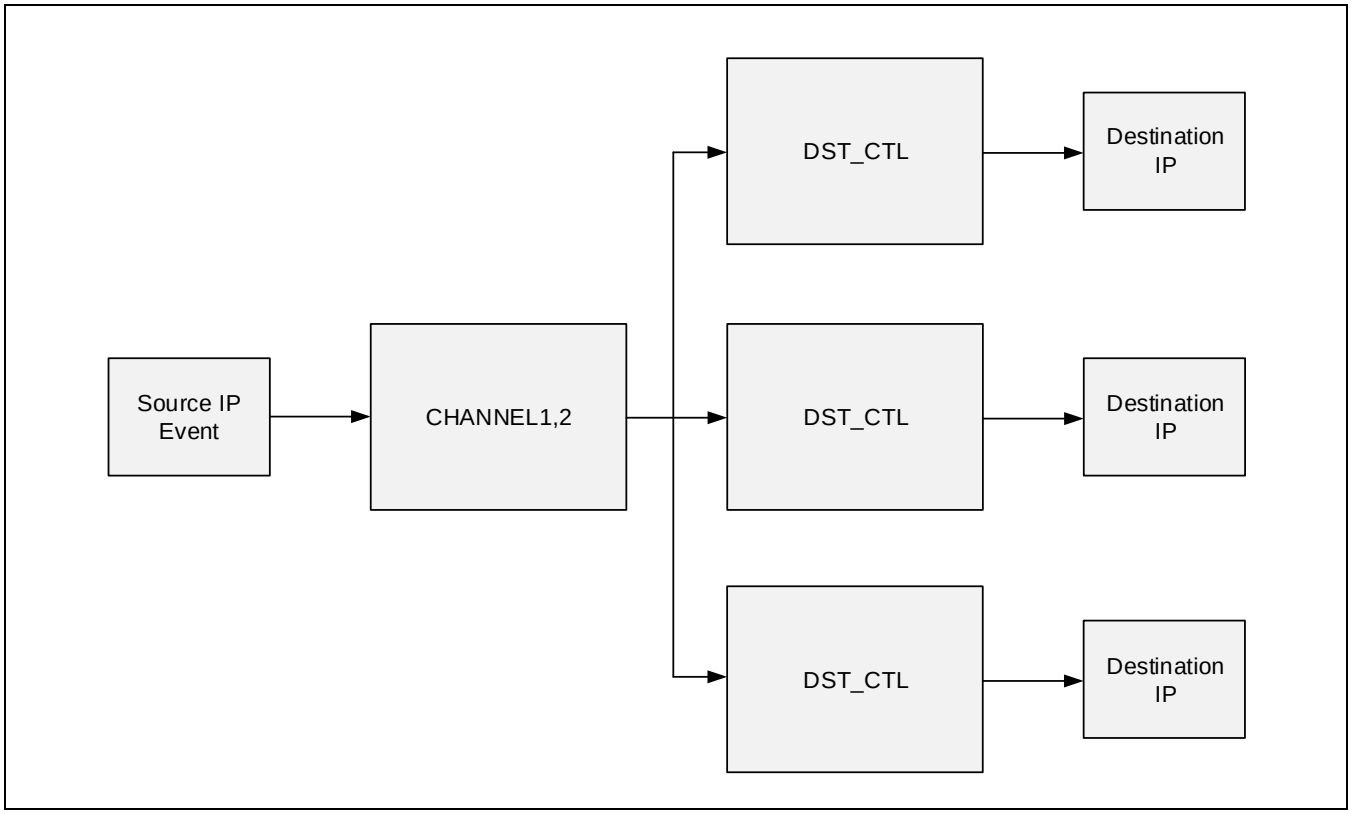


Figure 7-4 单个源触发多个目标

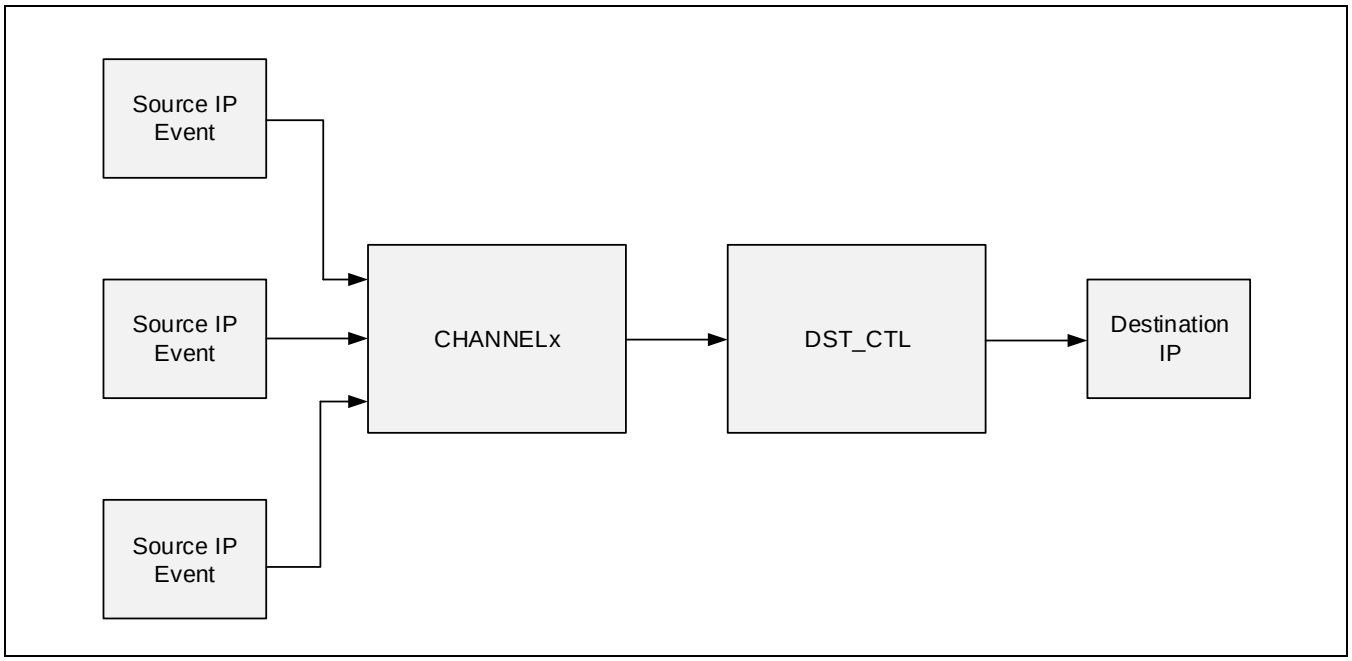


Figure 7-5　3个源触发单目标

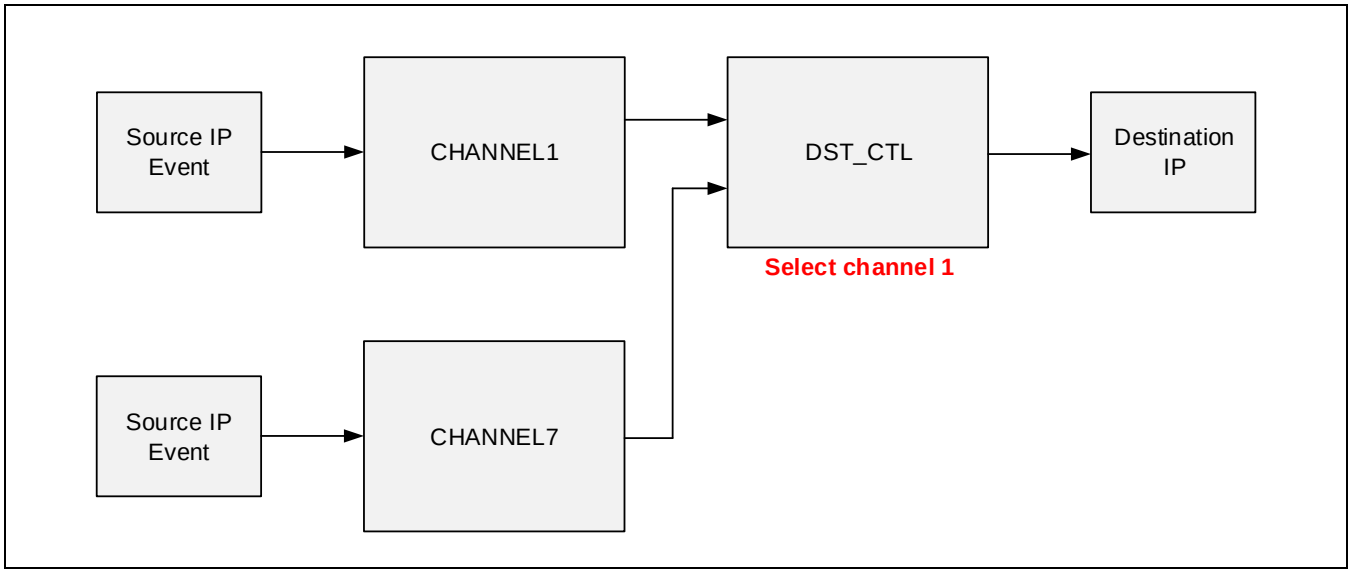


Figure 7-6　2个通道选择了相同的目标

7.2.2.2 事件对应表

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

Table 7-1 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）	ADC_SYNCIN0
7 （7H）	EXI_TRGOUT3	7 （7H）	ADC_SYNCIN1
8 （8H）	EXI_TRGOUT4	8 （8H）	ADC_SYNCIN2
9 （9H）	EXI_TRGOUT5	9 （9H）	ADC_SYNCIN3
10 （AH）	RSVD	10 （AH）	ADC_SYNCIN4
11 （BH）	RSVD	11 （BH）	ADC_SYNCIN5
12 （CH）	BT0_TRGOUT0	12 （CH）	BT1_SYNCIN1
13 （DH）	BT1_TRGOUT0	13 （DH）	BT1_SYNCIN2
14 （EH）	BT0_TRGOUT1	14 （EH）	RSVD
15 （FH）	BT1_TRGOUT1	15 （FH）	RSVD
16 （10H）	RSVD	16 （10H）	RSVD
17 （11H）	RSVD	17 （11H）	RSVD
18 （12H）	RSVD	18 （12H）	RSVD
19 （13H）	RSVD	19 （13H）	RSVD
20 （14H）	RSVD	20 （14H）	RSVD
21 （15H）	RSVD	21 （15H）	RSVD
22 （16H）	RSVD	22 （16H）	RSVD
23 （17H）	RSVD	23 （17H）	RSVD
24 （18H）	BT2_TRGOUT0	24 （18H）	BT2_SYNCIN0
25 （19H）	BT3_TRGOUT0	25 （19H）	BT2_SYNCIN1
26 （1AH）	BT2_TRGOUT1	26 （1AH）	BT2_SYNCIN2
27 （1BH）	BT3_TRGOUT1	27 （1BH）	BT3_SYNCIN0

28 （1CH）	RSVD	28 （1CH）	BT3_SYNCIN1
29 （1DH）	RSVD	29 （1DH）	BT3_SYNCIN2
30 （1EH）	RSVD	30 （1EH）	RSVD
31 （1FH）	RSVD	31 （1FH）	RSVD
32 （20H）	GPTA0_TRGOUT0	32 （20H）	RSVD
33 （21H）	GPTA0_TRGOUT1	33 （21H）	RSVD
34 （22H）	RSVD	34 （22H）	RSVD
35 （23H）	RSVD	35 （23H）	RSVD
36 （24H）	GPTB0_TRGOUT0	36 （24H）	GPTA0_SYNCIN0
37 （25H）	GPTB0_TRGOUT1	37 （25H）	GPTA0_SYNCIN1
38 （26H）	RSVD	38 （26H）	GPTA0_SYNCIN2
39 （27H）	RSVD	39 （27H）	GPTA0_SYNCIN3
40 （28H）	RSVD	40 （28H）	GPTA0_SYNCIN4
41 （29H）	RSVD	41 （29H）	GPTA0_SYNCIN5
42 （2AH）	RSVD	42 （2AH）	GPTA0_SYNCIN6
43 （2BH）	RSVD	43 （2BH）	RSVD
44 （2CH）	RSVD	44 （2CH）	RSVD
45 （2DH）	RSVD	45 （2DH）	RSVD
46 （2EH）	RSVD	46 （2EH）	RSVD
47 （2FH）	RSVD	47 （2FH）	RSVD
48 （30H）	ADC_TRGOUT0	48 （30H）	RSVD
49 （31H）	ADC_TRGOUT1	49 （31H）	RSVD
50 （32H）	RSVD	50 （32H）	GPTB0_SYNCIN0
51 （33H）	RSVD	51 （33H）	GPTB0_SYNCIN1
52 （34H）	RSVD	52 （34H）	GPTB0_SYNCIN2
53 （35H）	RSVD	53 （35H）	GPTB0_SYNCIN3
54 （36H）	RSVD	54 （36H）	GPTB0_SYNCIN4
55 （37H）	RSVD	55 （37H）	GPTB0_SYNCIN5
56 （38H）	RSVD	56 （38H）	GPTB0_SYNCIN6
57 （39H）	RSVD	57 （39H）	RSVD
58 （3AH）	RSVD	58 （3AH）	RSVD
59 （3BH）	RSVD	59 （3BH）	RSVD
60 （3CH）	TOUCH_TRGOUT	60 （3CH）	RSVD

61 （3DH）	RSVD	61 （3DH）	RSVD
62 （3EH）	RSVD	62 （3EH）	RSVD
63 （3FH）	RSVD	63 （3FH）	RSVD

7.3 寄存器说明

7.3.1 寄存器表

Base Address 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_CH3CON	0x0030	ETCB通道3控制寄存器	0x00000000
ETCB_CH4CON	0x0034	ETCB通道4控制寄存器	0x00000000
ETCB_CH5CON	0x0038	ETCB通道5控制寄存器	0x00000000

7.3.2 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
SRR								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
WO	WO	WO	WO	WO	WO	WO	WO	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
SRR	[31:24]	WO	软件复位控制位。 当对当前控制位写入‘0xA5’时，所有控制寄存器保持不变 （即不复位寄存器），恢复内部控制逻辑为初始状态。
ENABLE	[0]	RW	ETCB模块使能控制 0：禁止 1：使能

7.3.3 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
RSVD																										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
R	R	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
SWTRG_CHx	[5:0]	RW	软件触发控制 0：无效 1：触发该通道的事件

7.3.4 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					SRC2_SEL						SRC2_EN	RSVD			SRC1_SEL						SRC1_EN	RSVD			SRC0_SEL						SRC0_EN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW	RW	RW

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

7.3.5 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
DST_SEL						RSVD																								TRIG_MODE	CH0_EN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW

Name	Bit	Type	Description
DST_SEL	[31:26]	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

7.3.6 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

7.3.7 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	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
SRC_SEL	[31:26]	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

7.3.8 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

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

7.3.9 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	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
SRC_SEL	[31:26]	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

7.3.10 ETCB_CH3CON(ETCB通道3控制寄存器)

Address = Base Address+ 0x0030, 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	R	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	[17:12]	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

7.3.11 ETCB_CH4CON(ETCB通道4控制寄存器)

Address = Base Address+ 0x0034, 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	CH5_EN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	RW	RW

Name	Bit	Type	Description
DST_SEL	[31:26]	RW	触发目标选择 参考事件对应表
SRC_SEL	[17:12]	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)
CH5_EN	[0]	RW	通道5使能控制 0: 禁止 1: 使能

7.3.12 ETCB_CH5CON(ETCB通道5控制寄存器)

Address = Base Address+ 0x0038, 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	CH4_EN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	RW	RW

Name	Bit	Type	Description
DST_SEL	[31:26]	RW	触发目标选择 参考事件对应表
SRC_SEL	[17:12]	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)
CH4_EN	[0]	RW	通道4使能控制 0: 禁止 1: 使能

8 基本计数器 (Basic Timer)

8.1 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，那它就不具备本章描述的相关资源。具体参考芯片的数据手册。

8.1.1 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。

8.1.2 管脚描述

Table 8-1 BT 相关功能管脚描述

管脚名称	功能描述
BT_OUT	PWM 波形输出

8.2 功能描述

8.2.1 模块框图

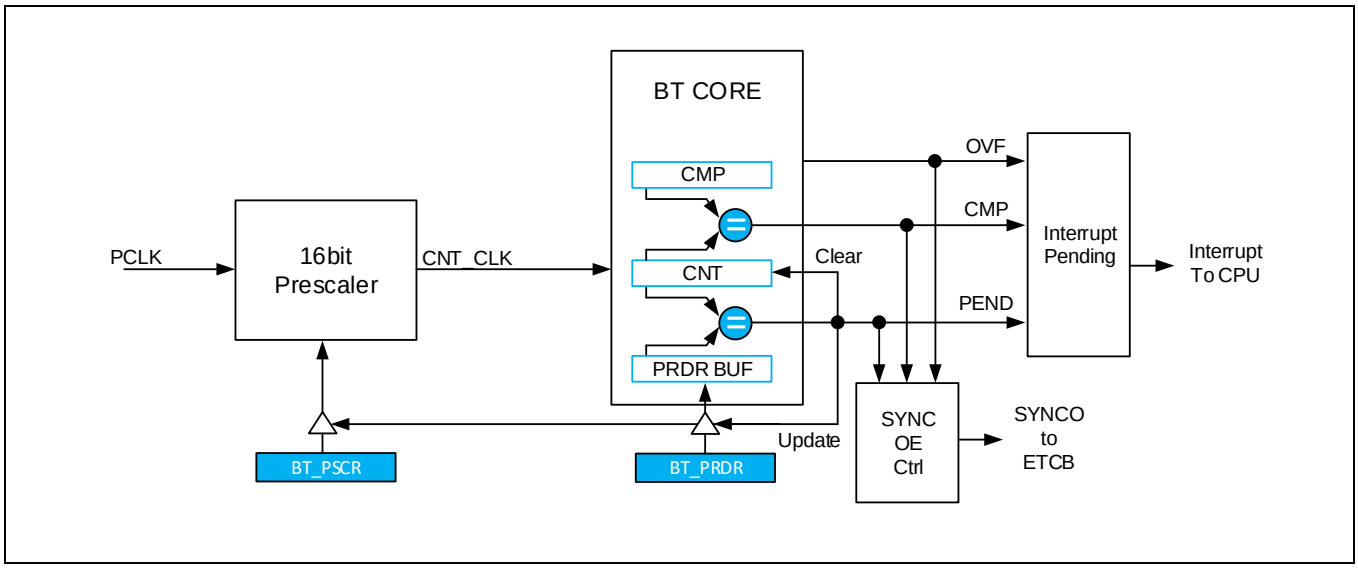


Figure 8-1 模块结构示意图

8.2.2 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。

当清除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）。

8.2.3 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。

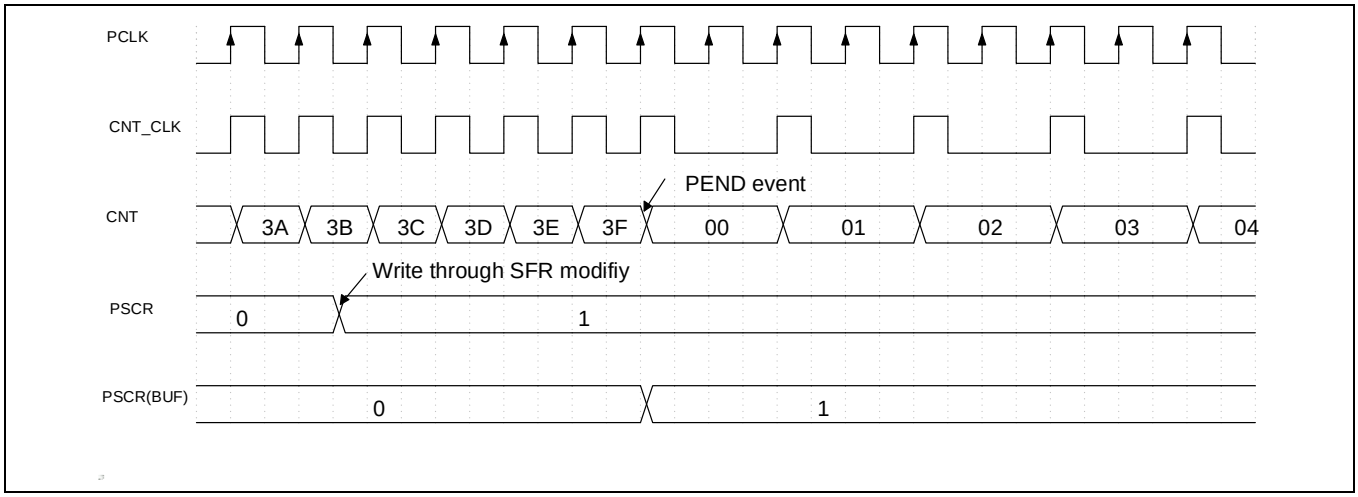


Figure 8-2 BT 周期计数模式

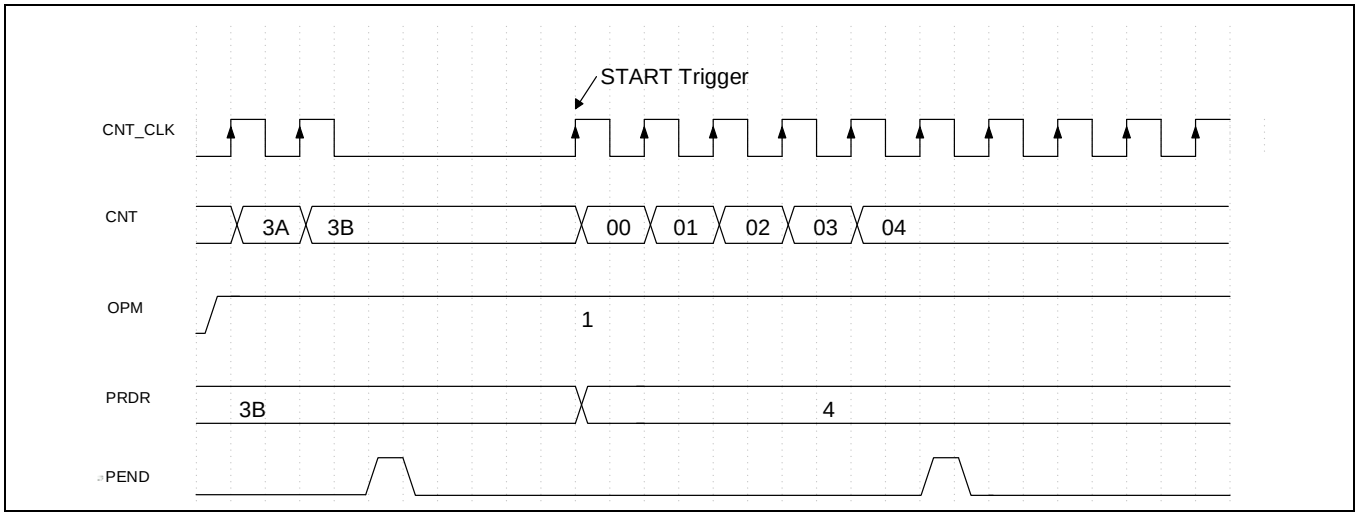


Figure 8-3 BT One Pulse 计数模式

8.2.4 连续计数

通过RCR寄存器，可以使能连续周期事件触发计数功能。当在连续周期计数被激活时重新启动计数器，若允许跨越窗口对齐，则当前周期计数值将被清零；若禁止跨越窗口对齐，则当前周期计数值不受影响。当周期事件计数值等于RCR[RPDR]+1时，停止连续BT计数器。

8.2.5 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。

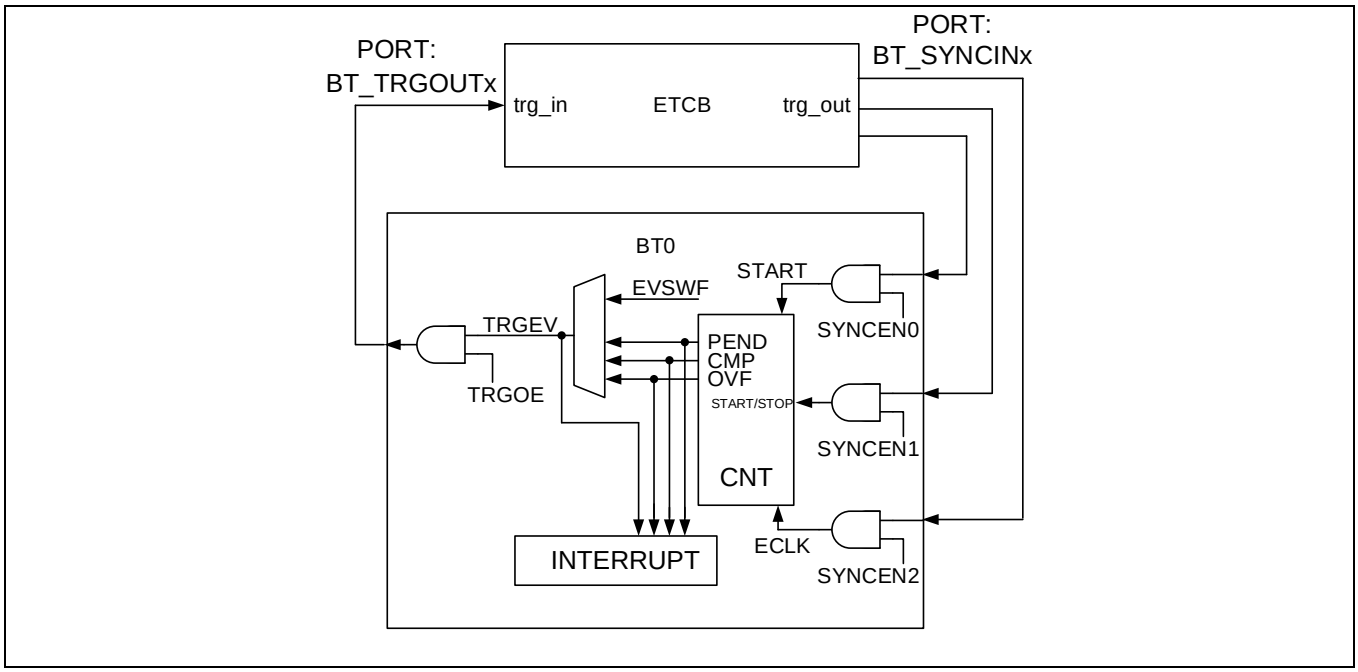


Figure 8-4 同步触发原理

8.2.5.1 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1可以触发Basic Timer的启停控制。只有当Basic Timer处于运行状态时，SYNCEN1才有效。第一次触发关闭Basic Timer，再次触发开启Basic Timer，依次轮循。

SYNC PORT2：同步输入端口2可以触发Basic Timer的计数值增加一拍。

8.2.5.2 事件触发（输出）

Basic Timer有两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，或者硬件直连到STOP。

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。

8.2.6 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。

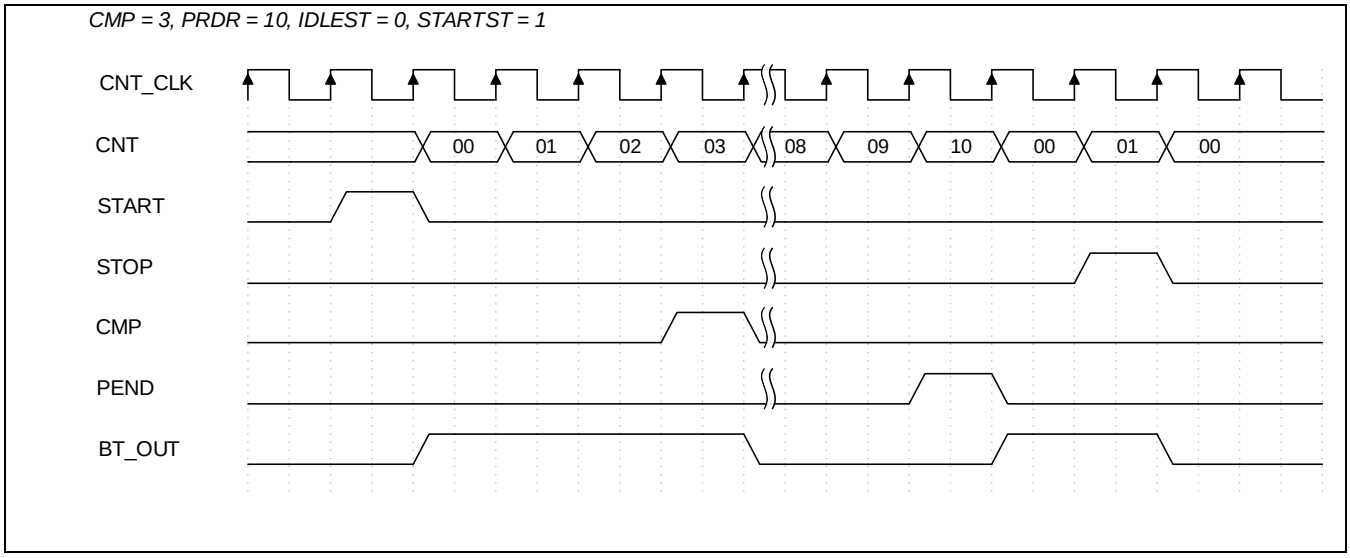


Figure 8-5 BT 输出波形时序图

8.2.7 中断控制

Basic Timer支持5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。

RPEND事件：连续PEND周期事件计数结束时发生。

8.3 寄存器说明

8.3.1 寄存器表

Base Address of BT0: 0x40051000
Base Address of BT1: 0x40052000
Base Address of BT2: 0x40053000
Base Address of BT3: 0x40054000

Register	Offset	Description	Reset Value
BT_RSSR	0x000	软件复位/启动控制寄存器	0x00000000
BT_CR	0x004	通用控制寄存器	0x00000000
BT_PSCR	0x008	计数器时钟预分频寄存器	0x00000000
BT_PRDR	0x00C	周期设置寄存器	0x00000000
BT_CMP	0x010	比较值寄存器	0x00000000
BT_CNT	0x014	时基计数器寄存器	0x00000000
BT_EVTRG	0x018	事件触发控制寄存器	0x00000000
BT_RCR	0x01C	连续计数控制寄存器	0x00000000
BT_EVSWF	0x024	事件软件触发控制寄存器	0x00000000
BT_RISR	0x028	原始中断状态寄存器	0x00000000
BT_IMCR	0x02C	中断使能控制寄存器	0x00000000
BT_MISR	0x030	中断状态寄存器	0x00000000
BT_ICR	0x034	中断清除寄存器	0x00000000

8.3.2 BT_RSSR(软件复位/启动控制寄存器)

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																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	WO	WO	WO	WO	R	R	R	R	R	R	R	R	R	R	R	RW

Name	Bit	Type	Description
SRR	[15:12]	WO	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。

8.3.3 BT_CR(通用控制寄存器)

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	REARM2	REARM1	REARM0	RSVD	OSTMD2	OSTMD1	OSTMD0	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
REARM2~REARM0	[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
OSTMD2~OSTMD0	[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。 x1：计数器周期结束时，自动REARM 1x：SYNCIN[0]时，自动REARM
AREARM0	[19:18]	RW	硬件自动REARM控制位。 x1：计数器周期结束时，自动REARM 1x：SYNCIN[1]时，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	外部同步触发输入使能控制,可以触发Basic Timer的计数值增加一拍。 0h：禁止外部触发 1h：使能外部触发
SYNCEN1	[9]	RW	外部同步触发输入使能控制，Basic Timer的启停控制。只有当Basic Timer处于运行状态时，SYNCEN1才有效。第一次触发关闭Basic

			Timer，再次触发开启Basic Timer，依次轮循。 0h：禁止外部触发 1h：使能外部触发
SYNCEN0	[8]	RW	外部同步触发输入使能控制,可以触发Basic Timer的START控制。 0h：禁止外部触发 1h：使能外部触发
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。

8.3.4 BT_PSCR(计数器时钟预分频寄存器)

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																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)

8.3.5 BT_PRDR(周期设置寄存器)

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
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。

8.3.6 BT_CMP(比较值寄存器)

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																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，产生CMP事件

8.3.7 BT_CNT(时基计数器寄存器)

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。

8.3.8 BT_EVTRG(事件触发控制寄存器)

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										TRG1OE	TRG0OE	RSVD												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	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
TRG1OE	[21]	RW	外部触发端口TRGOUT1输出使能。 0h：禁止触发输出到ETCB。 1h：允许触发输出到ETCB。
TRG0OE	[20]	RW	外部触发端口TRGOUT0输出使能。 0h：禁止触发输出到ETCB。 1h：允许触发输出到ETCB。
TRG1SEL	[7:4]	RW	TRGEV1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TRGEV1事件。 2h：指定CMP事件用于产生TRGEV1事件。 3h：指定OVF事件用于产生TRGEV1事件。 其他：保留
TRG0SEL	[3:0]	RW	TRGEV0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TRGEV0事件。 2h：指定CMP事件用于产生TRGEV0事件。 3h：指定OVF事件用于产生TRGEV0事件。 其他：保留

8.3.9 BT_RCR(连续计数控制寄存器)

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													RCNT			RSVD					RPRD			RSVD						CROSSMD	RMODE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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	R	R	R	R	R	R	RW	RW

Name	Bit	Type	Description
RCNT	[18:16]	R	当前连续周期事件计数值
RPRD	[10:8]	RW	连续周期事件计数值 当周期事件计数值等于（RPDR+1）时，停止连续BT计数器
CROSSMD	[1]	RW	连续周期计数跨越窗口对齐模式 当在连续周期计数被激活时重新启动计数器，若允许跨越窗口对齐，则当前周期计数值将被清零；若禁止跨越窗口对齐，则当前周期计数值不受影响 0：禁止跨越窗口对齐 1：允许跨越窗口对齐
RMODE	[0]	RW	连续周期计数使能位 0：禁止 1：使能

8.3.10 BT_EVSWF(事件软件触发控制寄存器)

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																														EVSWF1	EVSWF0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W

Name	Bit	Type	Description
EVSWF1	[1]	W	软件产生一次TRGEV1事件。 0h：写入’0’无效。 1h：软件产生一次TRGEV1事件。
EVSWF0	[0]	W	软件产生一次TRGEV0事件。 0h：写入’0’无效。 1h：软件产生一次TRGEV0事件。

8.3.11 BT_RISR(原始中断状态寄存器)

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																											RPEND	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
RPEND	[4]	R	连续PEND周期事件计数结束原始标志状态。
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

8.3.12 BT_IMCR(中断使能控制寄存器)

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																											RPEND	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	RW	RW	RW	RW	RW

Name	Bit	Type	Description
RPEND	[4]	RW	连续PEND计数中断使能控制位
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			

8.3.13 BT_MISR(中断状态寄存器)

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																											RPEND	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
RPEND	[4]	R	连续PEND周期事件计数结束标志状态。
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			

8.3.14 BT_ICR(中断清除寄存器)

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																											RPEND	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	W	W	W	W	W

Name	Bit	Type	Description
RPEND	[4]	W	清除PEND计数原始中断状态位。
EVTRG	[3]	W	清除事件触发中断原始中断状态位。
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’			

9 时钟定时器（TC3）

9.1 概述

时钟定时器是一个 16 位计数器。时钟定时器的时钟源可以设置为 EMOSC，ISOSC.

要启动时钟定时器，只需将控制寄存器(TC3_CR)中的 WTEN 位置 1，之后时钟定时器开始工作，一段时间后，时钟定时的中断会自动产生，中断的间隔 PRDR 或者 TIMDR 控制。

时钟定时器还可以产生一个稳定的信号驱动蜂鸣器输出管脚 BUZ，控制寄存器(TC3_CR)中的 BCSDIV 位可以设置蜂鸣器的输出频率。

9.1.1 特性

- 16位内部计数器。
- 可选工作时钟源。
  - EMOSC，支持32.768KHz的晶振。
  - ISOSC。
- 定时器支持两种工作模式：周期计数模式，连续计数模式。
- 频率可配置的蜂鸣器载波输出，输出频率可配。

9.1.2 管脚描述

Table 9-1 管脚描述

Pin Name	Function	I/O Type	Active Level	Comments
TC3_BUZZ	蜂鸣器频率输出	O	-	-

9.2 功能描述

9.2.1 模块框图

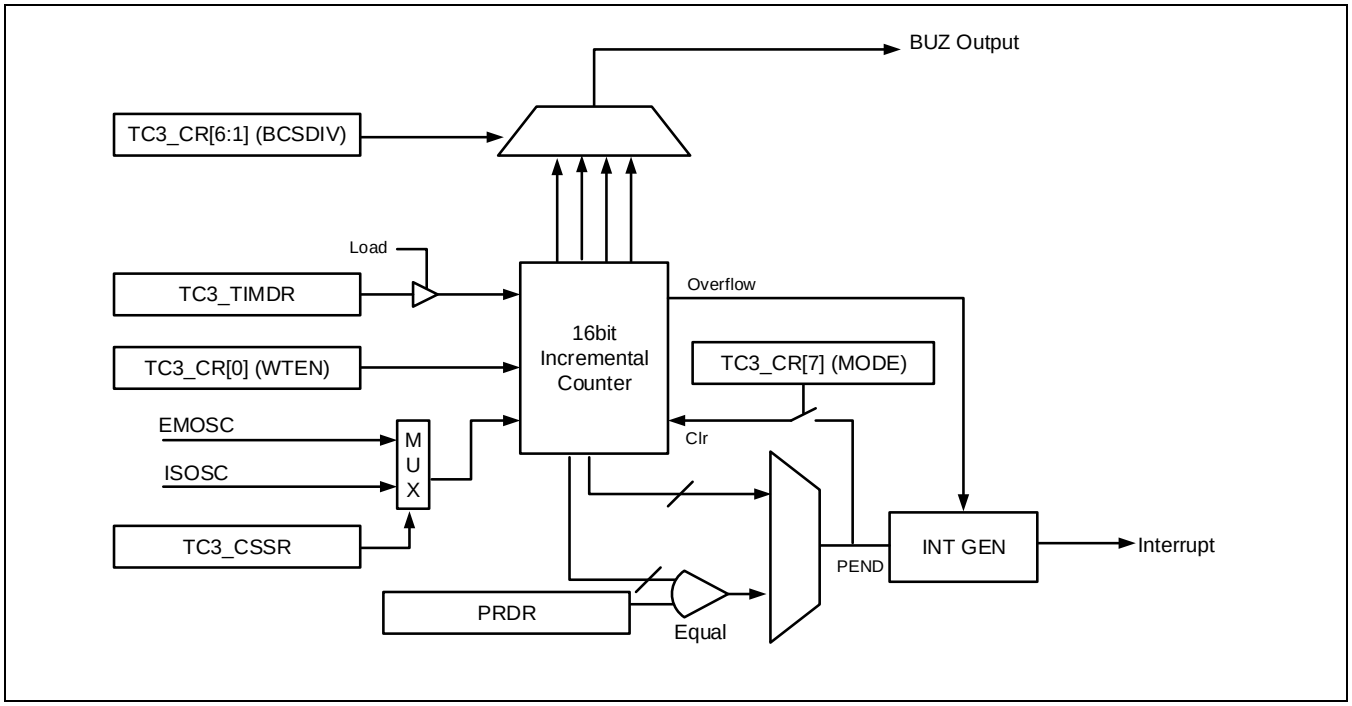


Figure 9-1 时钟定时器模块框图

9.2.2 工作原理

TC3 是一个 16 位长度的递增型计数器。工作时钟可以在 EMOSC 和 ISOSC 中进行选择。

TC3 可以支持两种工作模式，一种为普通计数模式（NORMAL MODE）。在此模式下，计数器一直递增，直到计数器溢出。溢出以后自动归零重新开始计数。在此模式下，当计数器值等于 PRDR 设置值时，会产生 PEND 中断。当计数器溢出时，会产生 OVF 中断。另外一种工作模式为周期计数模式（PERIOD MODE）。在此模式下，计数器一直递增，当发生 PEND 中断时，计数器会自动归零重新开始计数。

TC3 的计数在 WTEN 设置为高以后，开始计数。当 WTEN 被清除以后，计数器停止计数。在下一次 WTEN 置高时，计数器会被清零，重新开始计数。当对 TC3_TIMDR 进行写操作时，写入值将会被直接载入计数器，计数器从下一个周期开始，会从更新后的计数值开始计数。

在周期计数模式时，通过设置 PRDR 寄存器设置周期长度。计数周期 T_{Period} 与 PRDR 的关系如下：

$$T_{Period} = PRDR * BCSDIV * (1 / f_{tclk})$$

其中，f_{tclk} 为 TC3 的时钟频率，由 CSSR 寄存器配置决定。

TC3 自带蜂鸣器载波发生功能，可以通过配置 BCSDIV 设置蜂鸣器载波的频率。蜂鸣器的载波频率 f_{BUZ} 与 BSCDIV 的关系如下所示：

$$f_{BUZ} = f_{tclk} / (2 * BCSDIV)$$

其中，f_{tclk} 为 TC3 的时钟频率，由 CSSR 寄存器配置决定。

9.3 寄存器说明

9.3.1 寄存器表

Base Address of TC3: 0x40050000

Register	Offset	Description	Reset Value
TC3_IDR	0x000	ID控制寄存器	0x0001002D
TC3_CSSR	0x004	时钟源选择寄存器	0x00000001
TC3_CEDR	0x008	时钟使能/禁止控制寄存器	0x00000000
TC3_SRR	0x00C	软件复位寄存器	0x00000000
TC3_CR	0x010	控制寄存器	0x00000000
TC3_PRDR	0x014	周期设置寄存器	0x00000001
TC3_TIMDR	0x018	计数器数据寄存器	0x00000000
TC3_IMCR	0x01C	中断使能控制寄存器	0x00000000
TC3_RISR	0x020	中断原始状态寄存器	0x00000000
TC3_MISR	0x024	中断状态寄存器	0x00000000
TC3_ICR	0x028	中断状态清除寄存器	0x00000000

9.3.2 TC3_IDR(ID控制寄存器)

Address = Base Address+ 0x000, Reset Value = 0x0001002D

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

Name	Bit	Type	Description
IDR	[23:0]	R	ID Code寄存器 相应IP的ID Code，无法更改。

9.3.3 TC3_CSSR(时钟源选择寄存器)

Address = Base Address+ 0x004, 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																CSSR
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW

Name	Bit	Type	Description
CSSR	[0]	RW	计数器时钟源选择： 0： 外部晶振 1： 内部低速振荡器 当外部晶振作为32.768KHz时钟的输入源，定时器可以做精准的时间计时。

9.3.4 TC3_CEDR(时钟使能/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																														DBGEN	CLKEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW

Name	Bit	Type	Description
DBGEN	[1]	RW	Debug模式使能/禁止控制位。 0： 禁止Debug模式。 1： 使能Debug模式。 如果DBGEN置1，当CPU在debug模式被暂停后，计数器也同时被挂起。否则，计数器则继续工作
CLKEN	[0]	RW	计数器时钟使能/禁止控制位。 0：禁止计数器时钟。 1：使能计数器时钟。 SWRST不影响CLKEN位的状态。

9.3.5 TC3_SRR(软件复位寄存器)

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																															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

9.3.6 TC3_CR(控制寄存器)

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																								MODE	BCSDIV						WTEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
MODE	[7]	W	计数器工作模式选择 (只可写)。 0：NORMAL模式。 1：PERIOD模式。
BCSDIV	[6:1]	RW	蜂鸣器分频系数。 BCSDIV*2分频
WTEN	[0]	RW	计数器使能控制位 0： 停止计数 1： 启动计数

9.3.7 TC3_PRDR(周期设置寄存器)

Address = Base Address+ 0x014, 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																PRDR															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
PRDR	[15:0]	RW	周期设置寄存器。 周期等于PRDR*BCSDIV个时钟周期 即：Period = PRDR*BCSDIV * (1/ftclk), ftclk由CSSR设置决定

9.3.8 TC3_TIMDR(计数器数据寄存器)

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																TIMDR															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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
TIMDR	[15:0]	W	计数器数据寄存器 当对该寄存器进行写操作时，计数值被TIMDR重置。

9.3.9 TC3_IMCR(中断使能控制寄存器)

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																														OVF	PEND
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW

Name	Bit	Type	Description
OVF	[1]	RW	计数器溢出中断标志位
PEND	[0]	RW	周期结束中断标志位
该寄存器用于使能中断。 0： 关闭中断 1： 打开中断			

9.3.10 TC3_RISR(中断原始状态寄存器)

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																														OVF	PEND
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
OVF	[1]	R	计数器溢出中断原始标志位
PEND	[0]	R	周期结束中断原始标志位

9.3.11 TC3_MISR(中断状态寄存器)

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																														OVF	PEND
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
OVF	[1]	R	计数器溢出中断标志位
PEND	[0]	R	周期结束中断标志位

9.3.12 TC3_ICR(中断状态清除寄存器)

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																														OVF	PEND
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W

Name	Bit	Type	Description
OVF	[1]	W	清除计数器溢出中断标志位
PEND	[0]	W	清除周期结束中断标志位
该寄存器用于清除原始中断源标志位，该寄存器为只写寄存器 0： 无效 1： 清除中断标志			

10 增强型通用定时器 (GPTA)

10.1 概述

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

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

10.1.1 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）
- 波形产生控制单元，支持单路 PWM 输出
- 通过事件驱动 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
- 支持捕获模式，最多支持 4 个捕获值存储。

10.1.2 管脚描述

下表列出了不同模式下的管脚定义。

Table 10-1 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
GPTA_CHA	时钟控制使能	输出波形

10.2 功能描述

10.2.1 模块框图

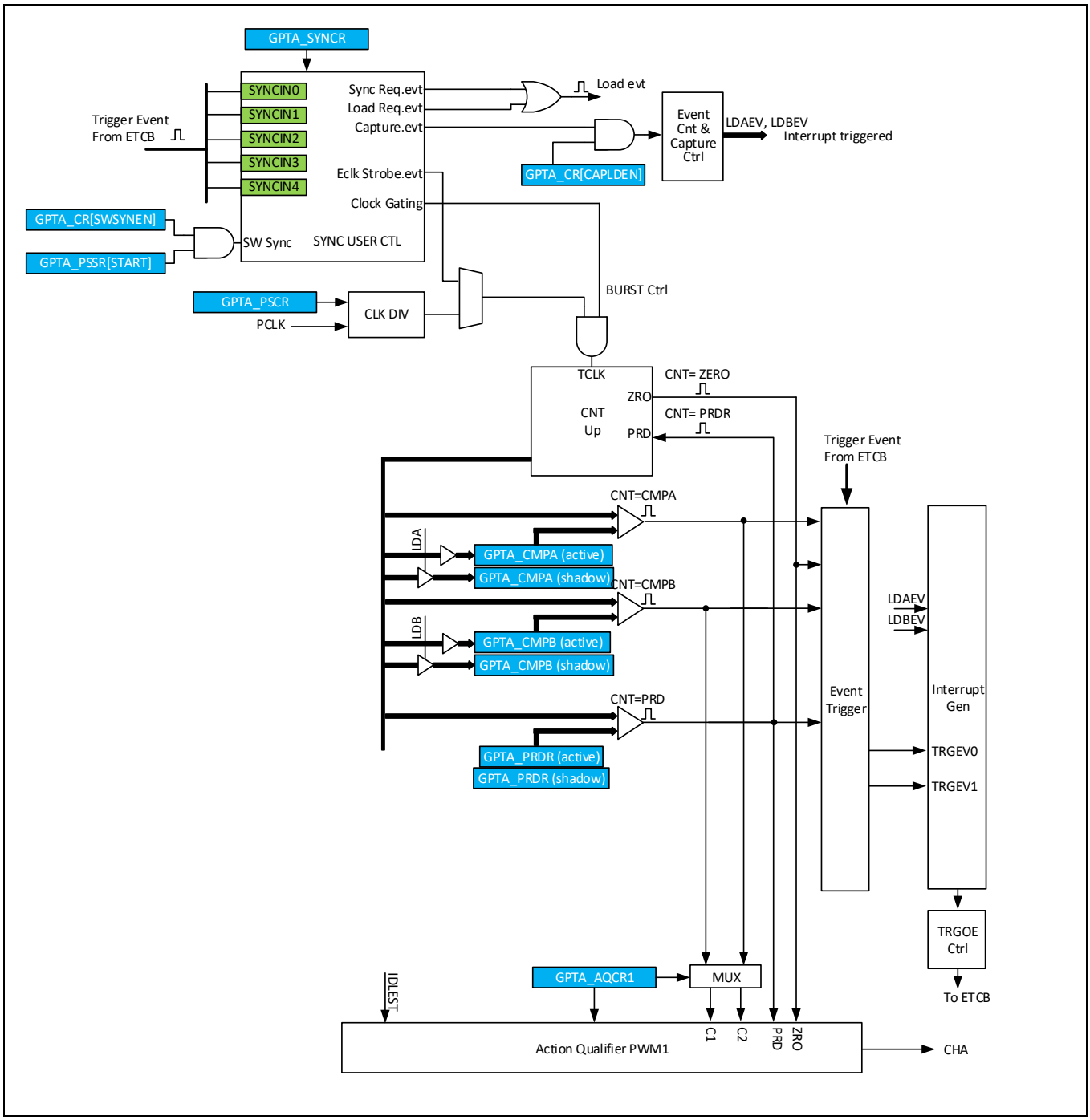


Figure 10-1 模块结构示意图

10.3 基本功能描述

一个完整的GPTA模块由一个输入/输出通道组成。多个GPTA或多个EPT可以通过SYNC链连接，通过SYNC链，实现多个GPTA和EPT的同步工作。在包含多个GPTA的器件中，以数字后缀区分不同的实例，例如：GPTA0代表第一个GPTA模块，GPTA1代表第二个GPTA模块。每个GPTA中根据功能划分，可以分为几个模块，包括时钟控制模块、时基模块（计数器）、计数器比较模块、捕捉控制模块、事件触发模块和同步触发控制模块。

GPT_CHA是GPTA在GPIO上映射的双向输入输出端口。在波形输出模式下，PWM信号通过GPT_CHA输出；在群脉冲模式下（GPTA_CR[BURST]使能），GPT_CHA可以作为门控时钟的时钟控制输入信号。

10.3.1 时钟源

10.3.1.1 概述

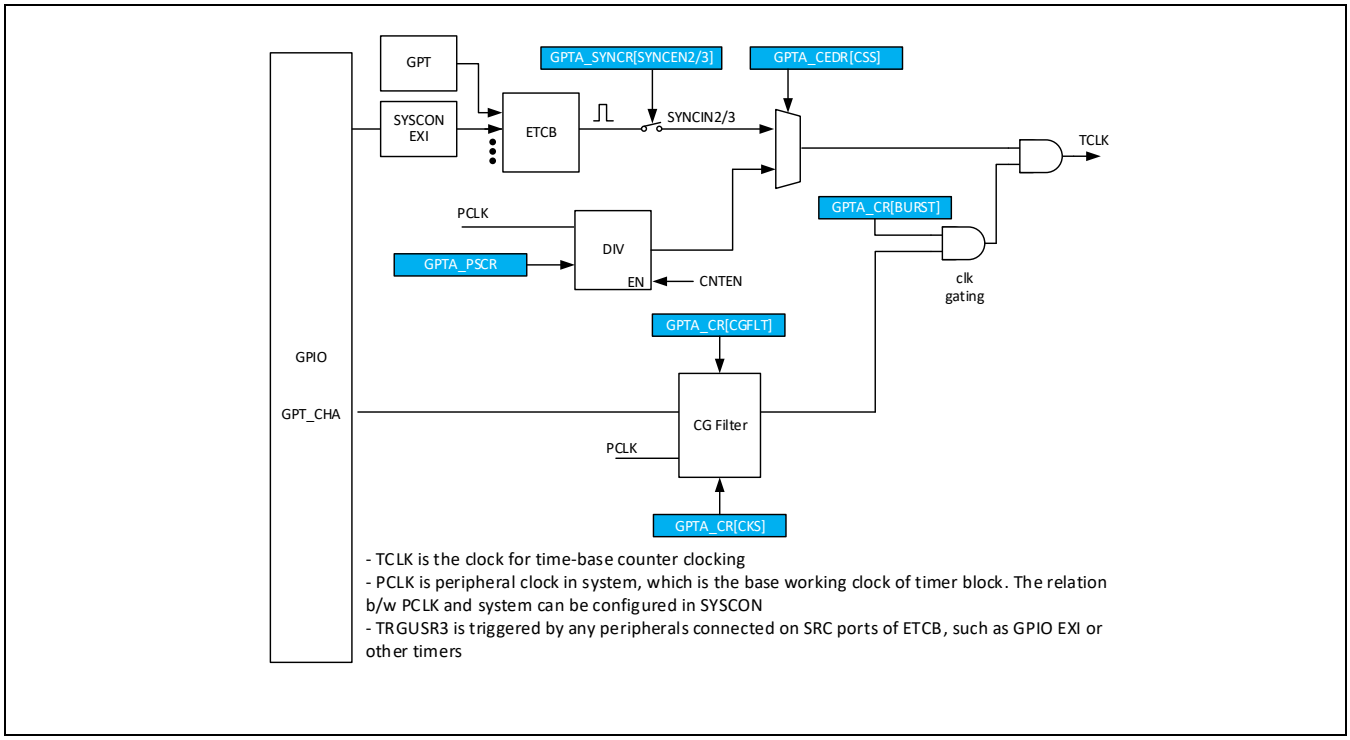


Figure 10-2 时钟控制模块

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。

10.3.1.2 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。

10.3.1.3 内部时钟

当PCLK作为计数器的计数时钟时，可以通过一个16位的预分频器对PCLK进行分频而产生计数用的TCLK。预

分频可以通过GPTA_PSCR进行设置。在对GPTA_PSCR进行读写时，操作的对象为PSCR的影子寄存器（Shadow Register）；当时基计数器的值等于零时，影子寄存器的值将被更新到内部的活动寄存器中（Active Register）。当对GPTA_PSCR更新后，新的分频将在下一个计数周期开始时有效。

10.3.1.4 群脉冲时钟

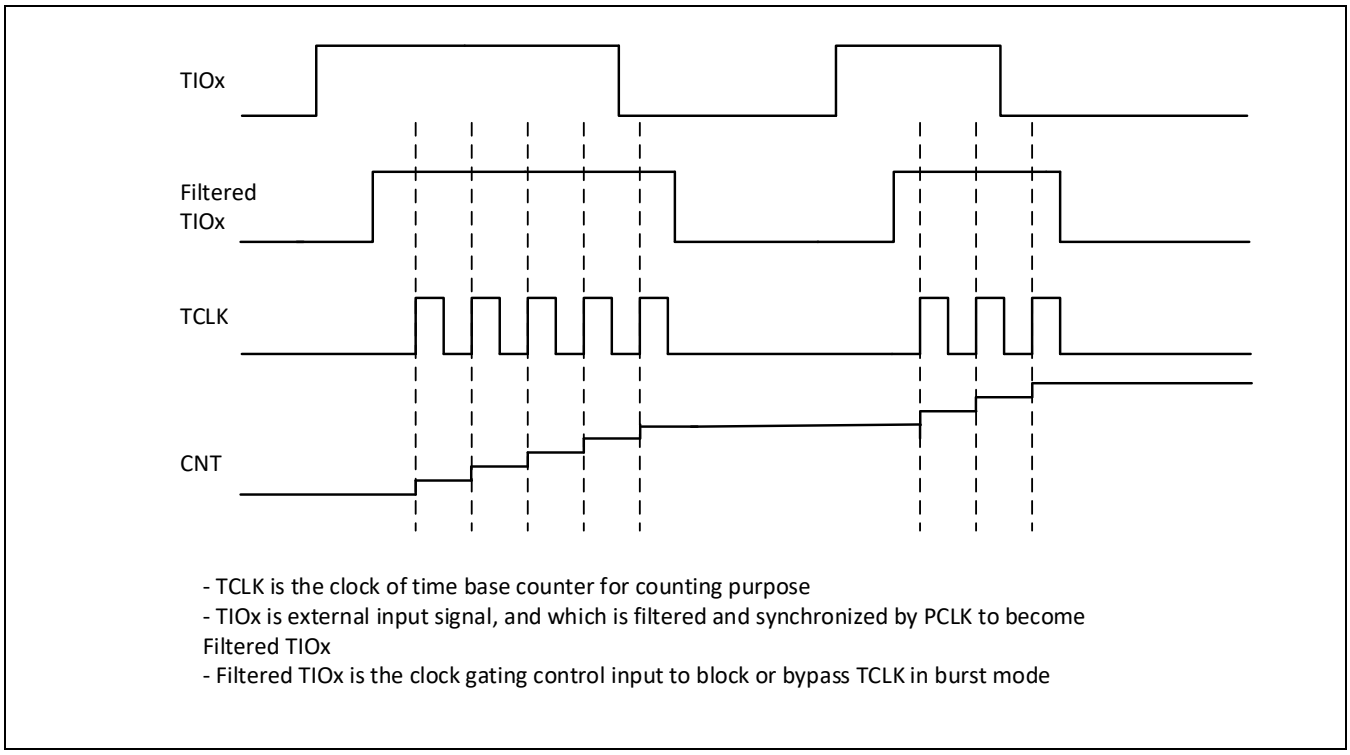


Figure 10-3 群脉冲时钟模式时序

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，支持GPT_CHA的外部输入作为门控信号。当GPT_CHA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。

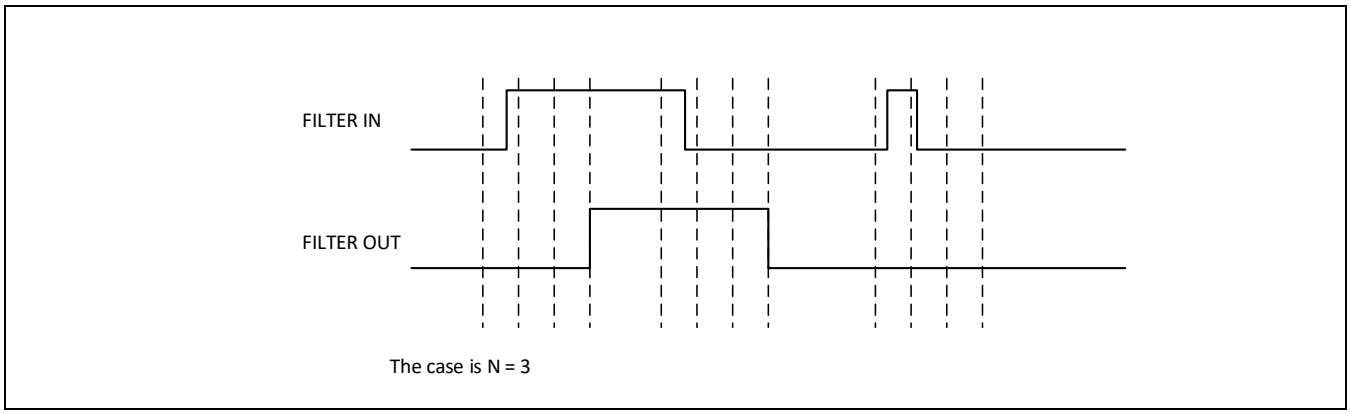


Figure 10-4 CG Filter 数字滤波器的原理

10.3.2 时基控制

10.3.2.1 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
- 周期寄存器（GPTA_PRDR）：计数器周期控制寄存器。

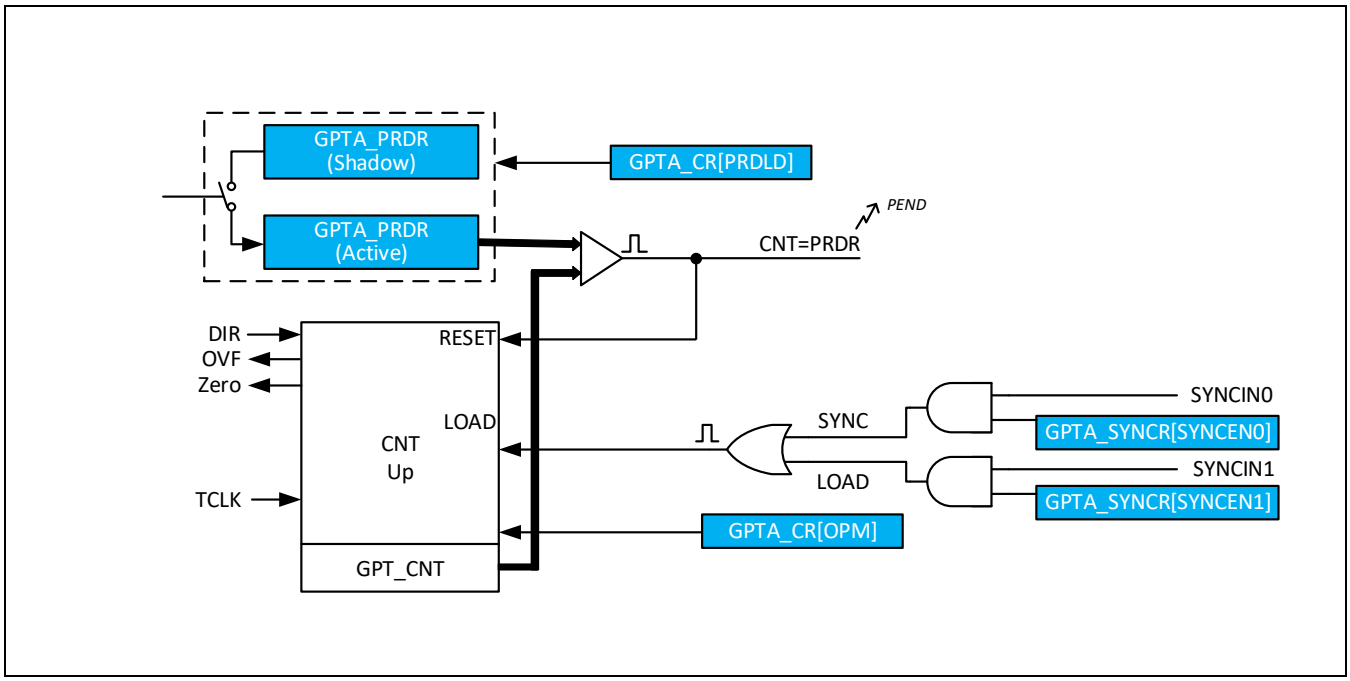


Figure 10-5 计数器时基模块

计数器的计数周期由周期寄存器（GPTA_PRDR）的设置值决定。计数器支持递增计数模式：

- 递增模式（Up-Counting Mode）：

  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。

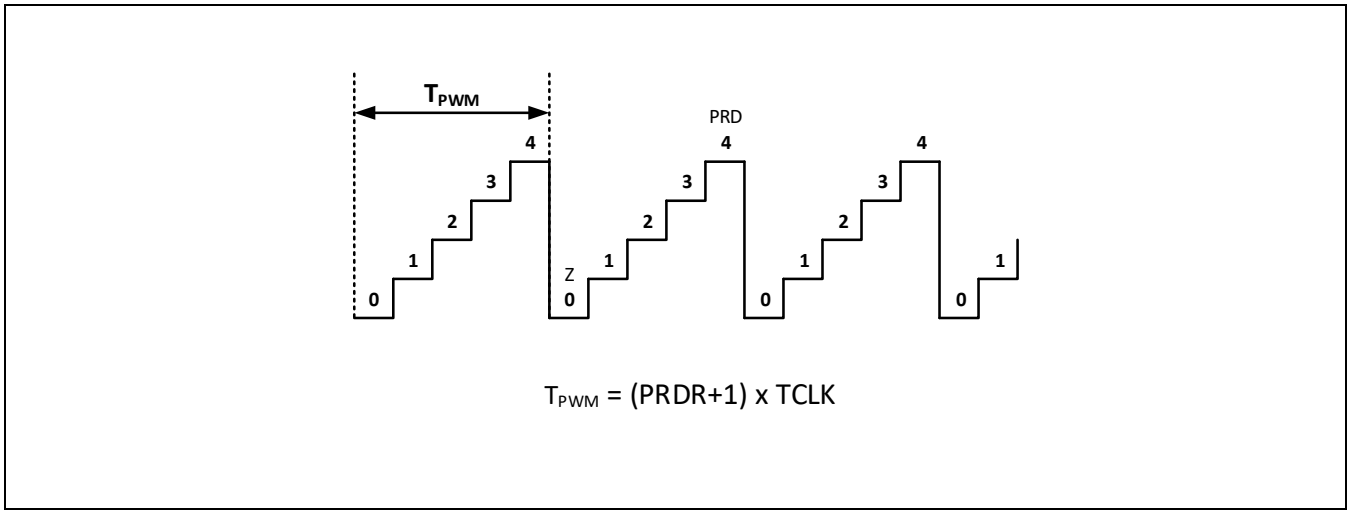


Figure 10-6 计数器工作模式

当计数达到最大计数值(0xFFFF)时，计数器回滚到0x0000，并且将溢出标志位(GPTA_RISR[IOVF]置高，同时可以产生一个中断(通过中断使能控制寄存器GPTA_IER[IOVF]设置)，然后继续开始计数。

10.3.2.2 计数器重置和周期设置

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

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

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

- **PRDR寄存器的Shadow模式**

  PRDR的缓冲（Shadow Register）在GPTA_CR[PRDLD]控制位不等于'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。当时基计数器值等于零时，或者SYNC触发时，影子寄存器的值被硬件自动载入到活动寄存器（Active Register）中。在缺省配置下，只有时基计数器值等于零时，自动载入才会发生，用户可以通过配置GPTA_CR[PRDLD]控制位进行修改。

- **PRDR寄存器的立即加载模式**

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

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

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

10.3.2.3 计数模式和时序

时基计数器工作模式为递增计数模式（非对称）。

在下面的图示中说明了递增工作模式下，时基计数器根据触发条件如何工作和产生相应事件。

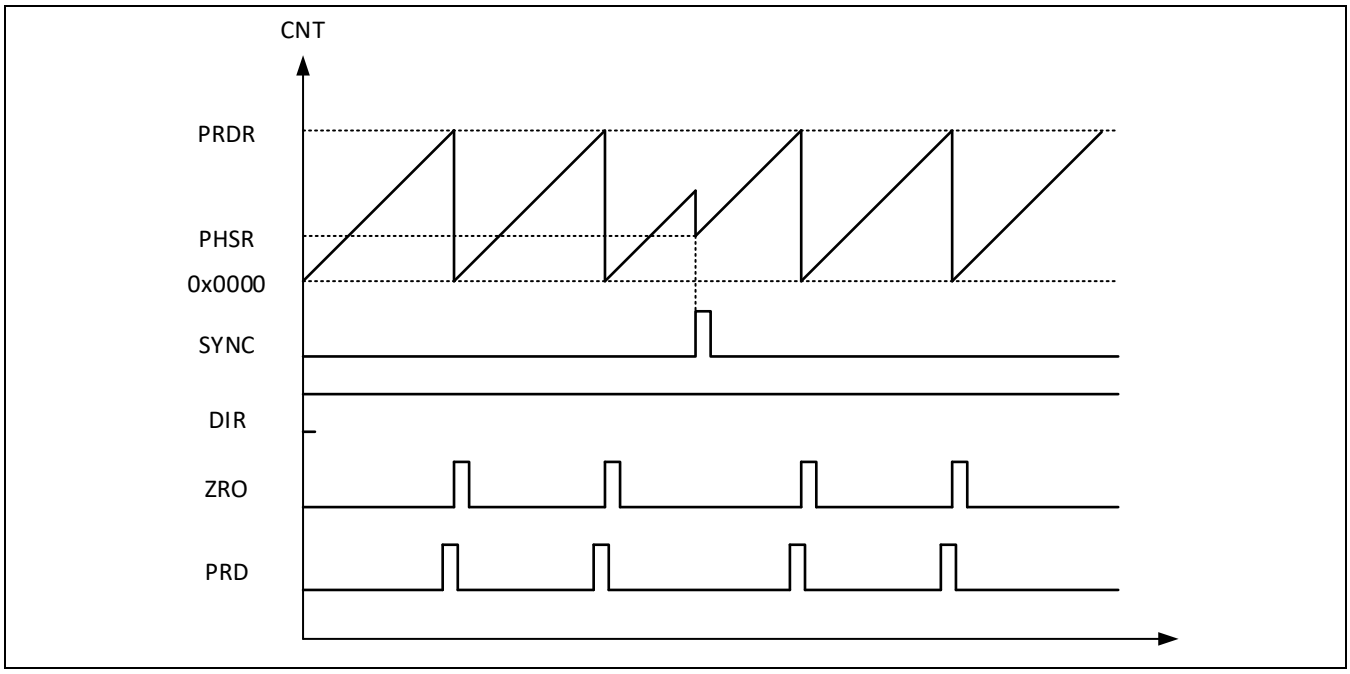


Figure 10-7 递增工作模式

10.3.3 计数器数值比较控制

10.3.3.1 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）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

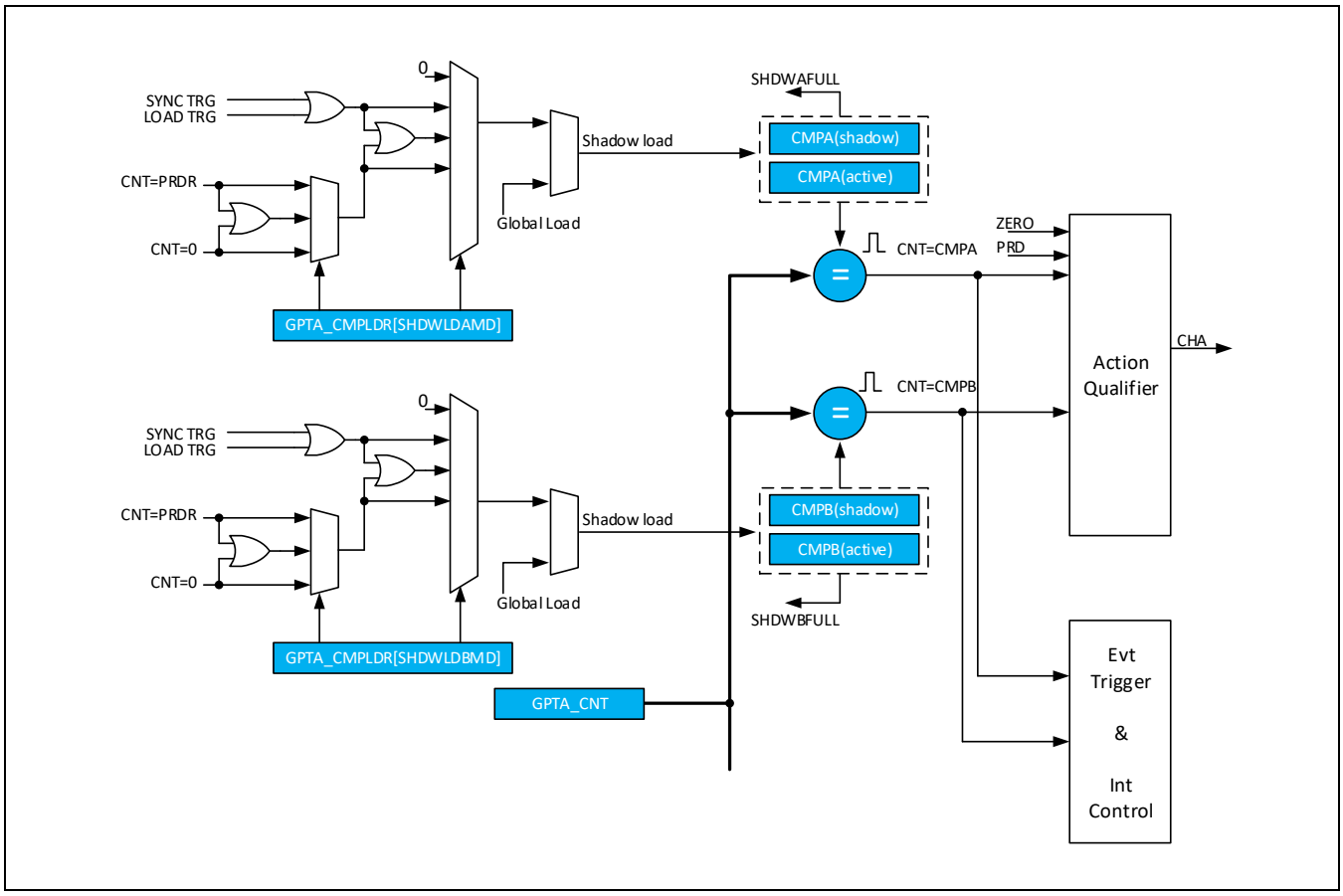


Figure 10-8 计数器值比较控制

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

计数器递增模式，每个比较事件在一个计数周期内只会发生一次。CMPA和CMPB这两个触发事件，在波形发生模块中决定了输出波形的跳转时间点。

10.3.3.2 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。

递增模式的时序

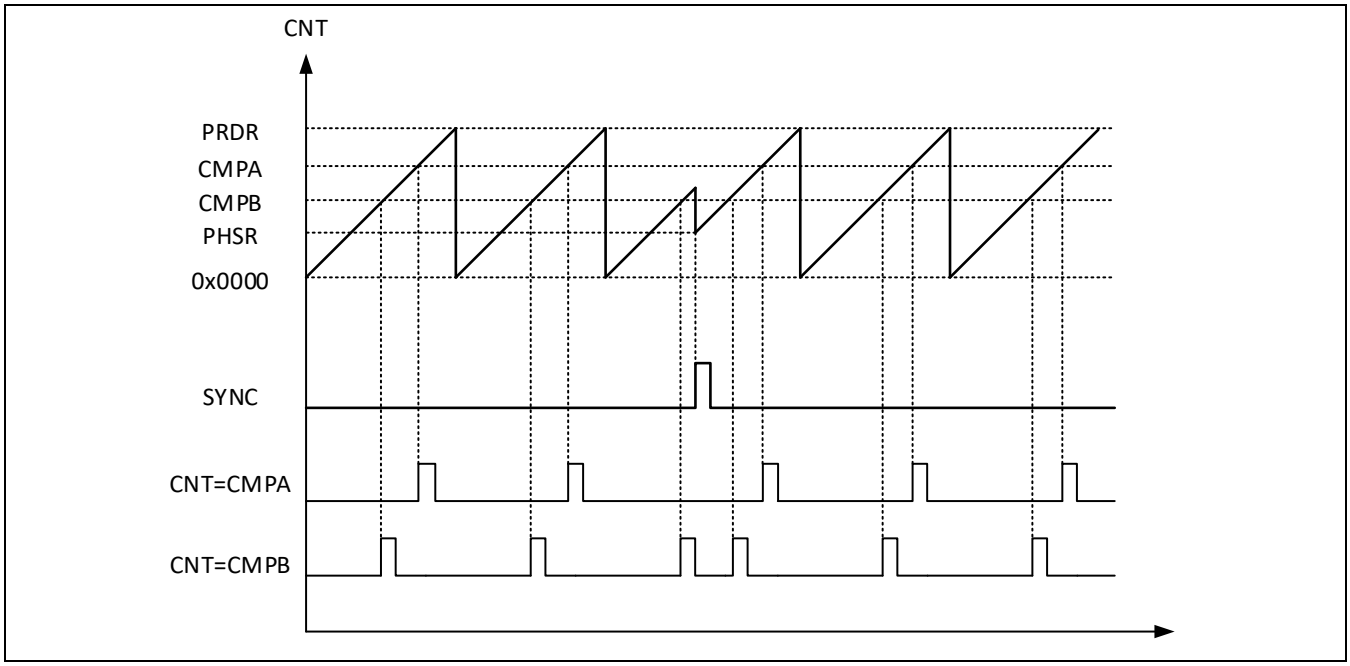


Figure 10-9 递增模式下比较事件产生时序

10.3.4 工作模式选择

GPTA通过配置，工作模式可以仅支持捕获模式，仅支持波形发生，或者同时支持捕获和波形发生。GPTA工作模式具体配置如下：

CR[CAP_WAVE]= 0时，CMPB用于捕获，CMPA用于捕获。

CR[CAP_WAVE]= 1时，CMPB用于波形，CMPA用于捕获。

CR[CAP_WAVE]= 2时，CMPB用于捕获，CMPA用于波形。

CR[CAP_WAVE]= 3时，CMPB用于波形，CMPA用于波形。

10.3.5 波形发生控制

10.3.5.1 事件驱动的波形输出

GPTA中有一路独立的波形输出通路（PWM1），注意，这里的PWM1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的设置，可以独立映射事件触发到PWM1的输出状态。GPTA_AQCR1对应控制PWM1通道上的波形输出。GPTA_AQCR1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AQCRx[C1SEL]设置）
- CNT = C2 （计数器值等于C2设置值，C2参考值由AQCRx[C2SEL]设置 ）
- 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_AQCR1[C1SEL]和GPTA_AQCR1[C2SEL]选择不同的数据源。

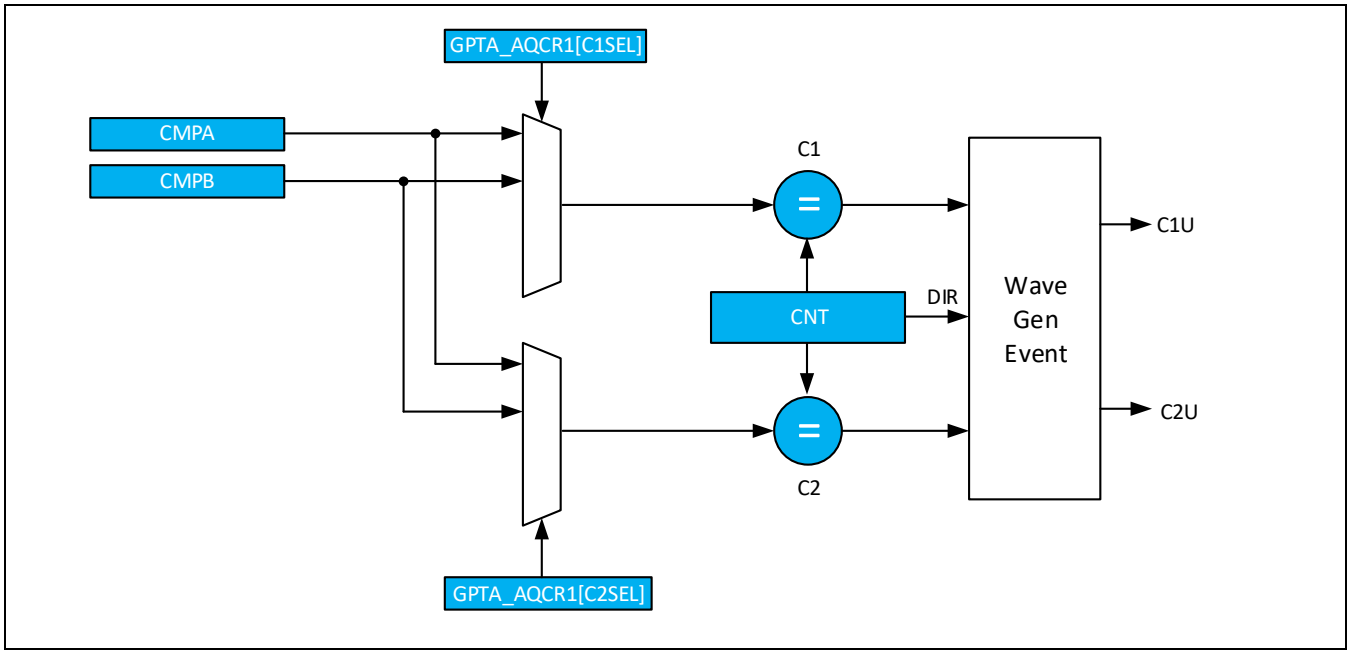


Figure 10-10 C1 and C2 selection control

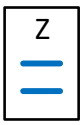
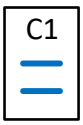
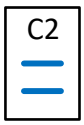
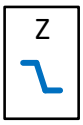
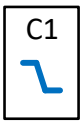
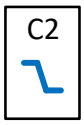
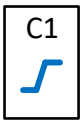
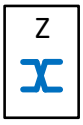
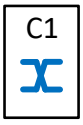
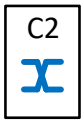
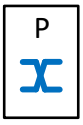
波形发生模块根据发生的事件，决定PWM1通道上的动作。所支持的的输出动作包括：

- 设置高电平 （在PWM1通道上设置高电平输出）
- 设置低电平 （在PWM1通道上设置低电平输出）
- 翻转 （在PWM1通道上对输出进行翻转）
- 保持 （保持当前PWM1通道上的电平）

波形发生器可以独立定义PWM1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 10-2 各种事件在PWM1上可能触发的动作

软件 Force	CNT 值等于				动作
	Zero	C1SEL	C2SEL	PRD	
					没有动作
					低电平输出
					高电平输出
					翻转输出

10.3.5.2 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，列出递增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 10-3 递增模式下的事件优先级

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

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，由于计数器设置为递增模式，C1U/C2U事件始终不会被触发。

10.3.5.3 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Shadow寄存器的Load方式。

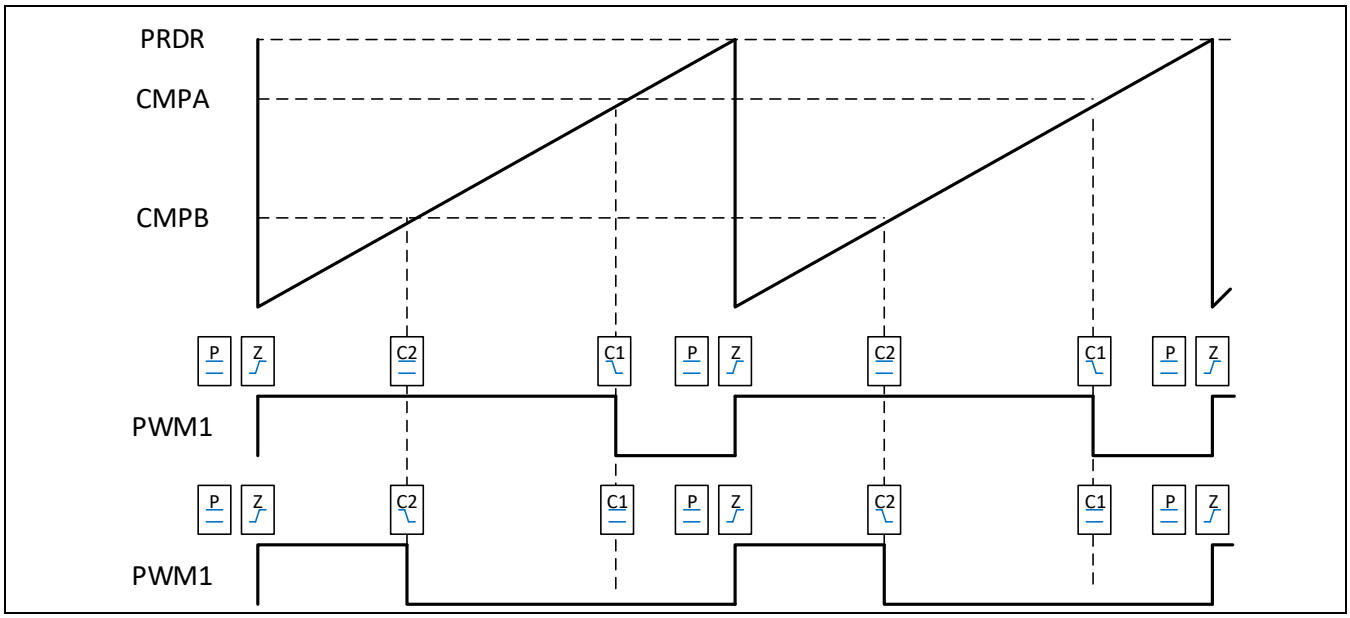


Figure 10-11 递增单沿，非对称波形输出

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。

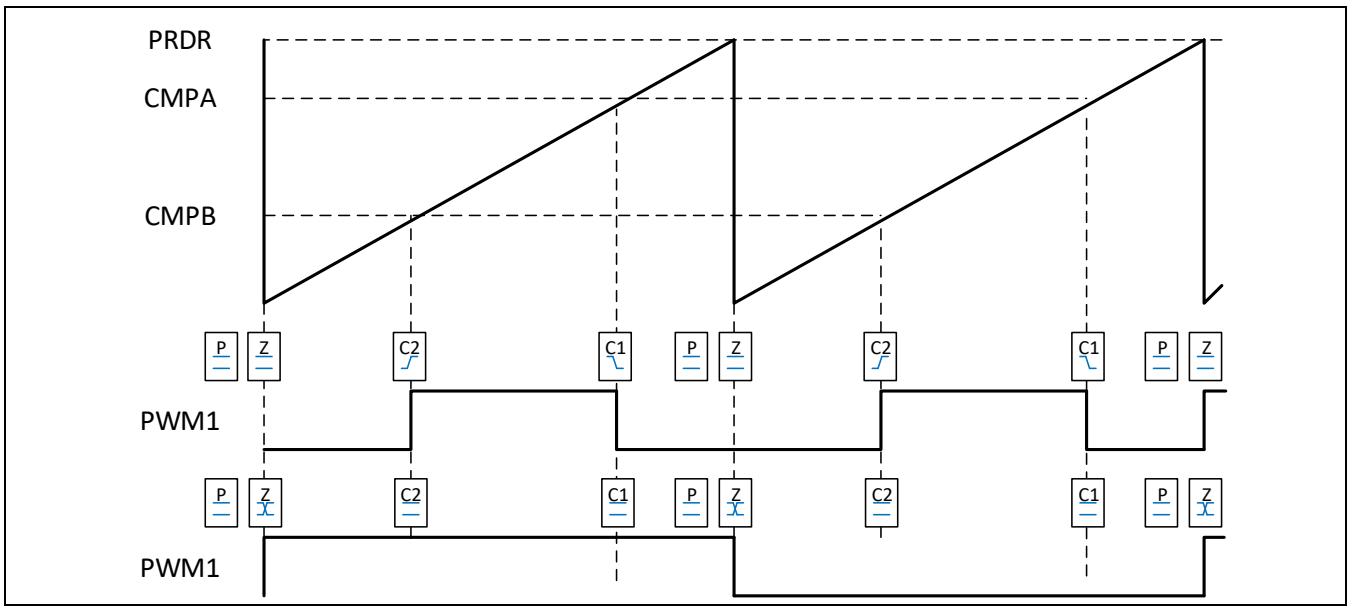


Figure 10-20 递增，脉冲定位非对称波形输出

10.3.6 捕获模式

10.3.6.1 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

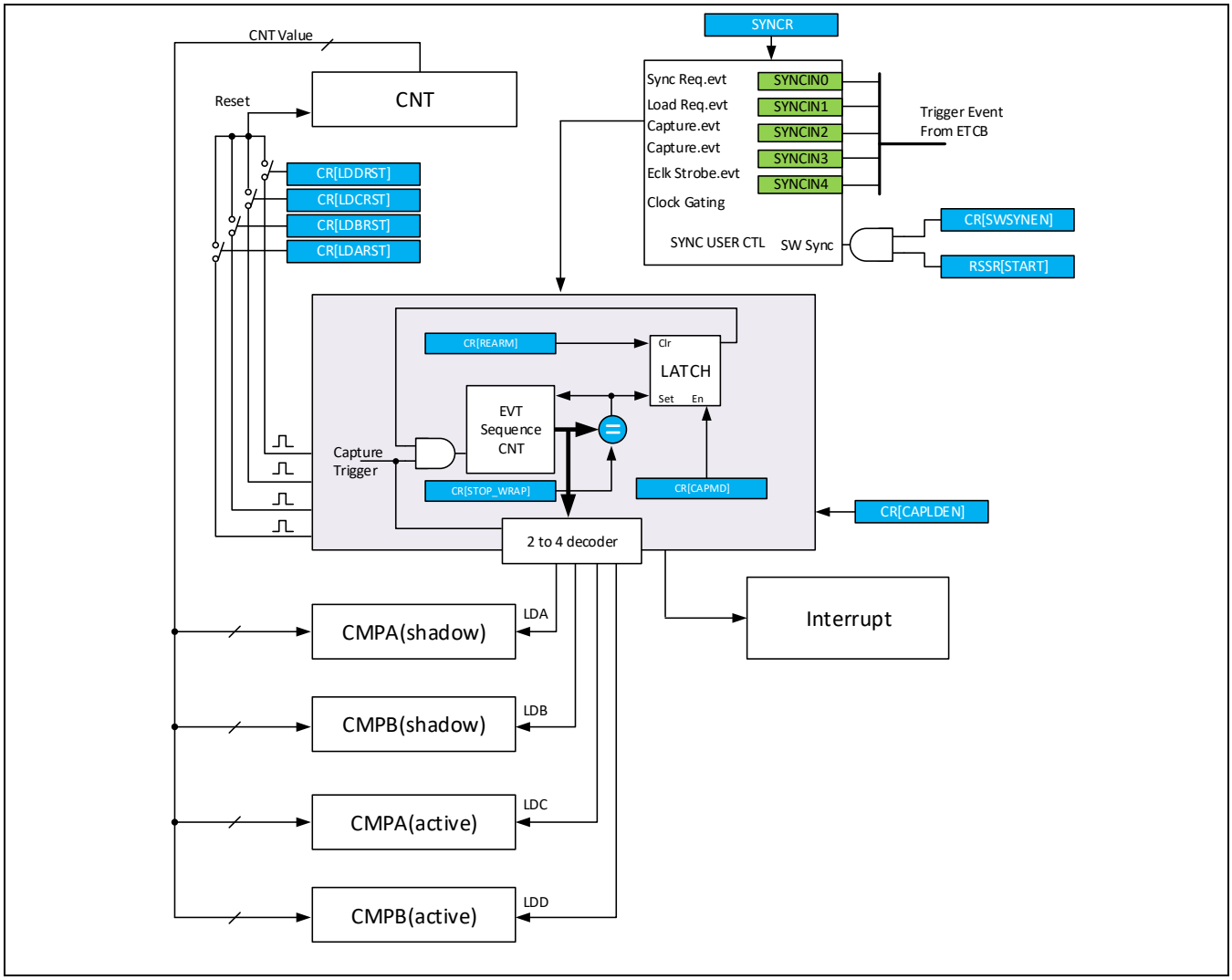


Figure 10-12 捕捉模式结构框图

捕获模式支持合并捕获和分开捕获两种模式，可以通过CR[CAPMD_SEL]配置。

当CR[CAPMD_SEL]=0时，为合并捕捉模式，该模式不区分SYNCIN2/SYNCIN3的触发，支持四次捕获，捕获值分别存入CMPA,CMPB,CMPAA, CMPBA。

当CR[CAPMD_SEL]=1时，为分开捕捉模式，该模式区分SYNCIN2/SYNCIN3的触发。根据捕获与波形的选择

位CR[CAP_WAVE]的配置不同，捕获值的存入寄存器不同，具体如下：

- CR[CAP_WAVE]= 0时，SYNCIN2的触发，捕捉值存入CMPA；SYNCIN3的触发，捕捉值存入CMPB。
- CR[CAP_WAVE] = 1时，SYNCIN2的触发，捕捉值存入CMPA；SYNCIN3的触发，捕捉值存入CMPAA。
- CR[CAP_WAVE]= 2时，SYNCIN2的触发，捕捉值存入CMPB；SYNCIN3的触发，捕捉值存入CMPBA

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，SYNCIN3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。

10.3.6.2 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[REAMR]控制位置高，进行重新初始化）。在连续模式下，当捕获触发条件满足时，序列计数器超出STOP_WRAP后，会从零开始重新计数，若在新计数器值被捕获时，当前通道捕获标志已经置位，则捕获值覆盖标志位将被置位。捕获标志可以通过软件清除，或者在读取相应CMP寄存器后，硬件自动清除；捕获值覆盖标志必须通过软件清除。

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

10.3.6.3 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/SYNCIN3。需要通过设置ETCB，连接GPTA SYNCIN2/ SYNCIN3的输入事件。计数器在每次捕获事件触发后，可以自动清零，通过设置GPTA_CR寄存器中[LDxRST]控制位进行设置。在触发事件发生时，相应的中断标志位被置位，可以通过使能相应的中断开关，控制进入CPU中断。

当同一个外部信号被同时配置为SYNCIN0和SYNCIN2/ SYNCIN3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。

10.3.6.4 应用举例

下面有一些例子，说明如何使用捕捉模式。

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

One-shot模式，STOP_WRAP = 2，LDA/BRST = 1。设置TIOB上升沿为SYNCIN0输入，TIOA上升沿和下降沿都设为SYNCIN2输入。TIOB上升沿复位计数器，TIOA的上升沿触发第一次load，计数值存入CMPA中。TIOA下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。

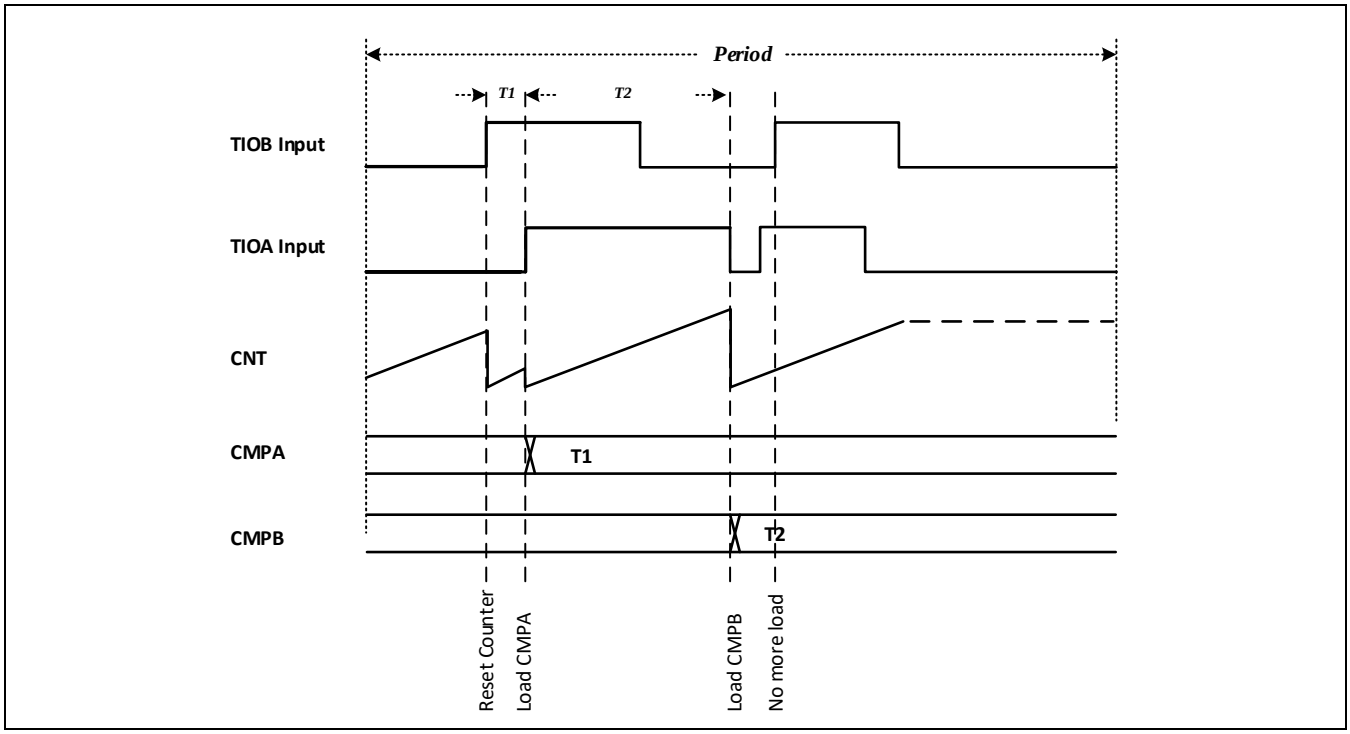


Figure 10-13 测量TIOA和TIOB相位差

- **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。

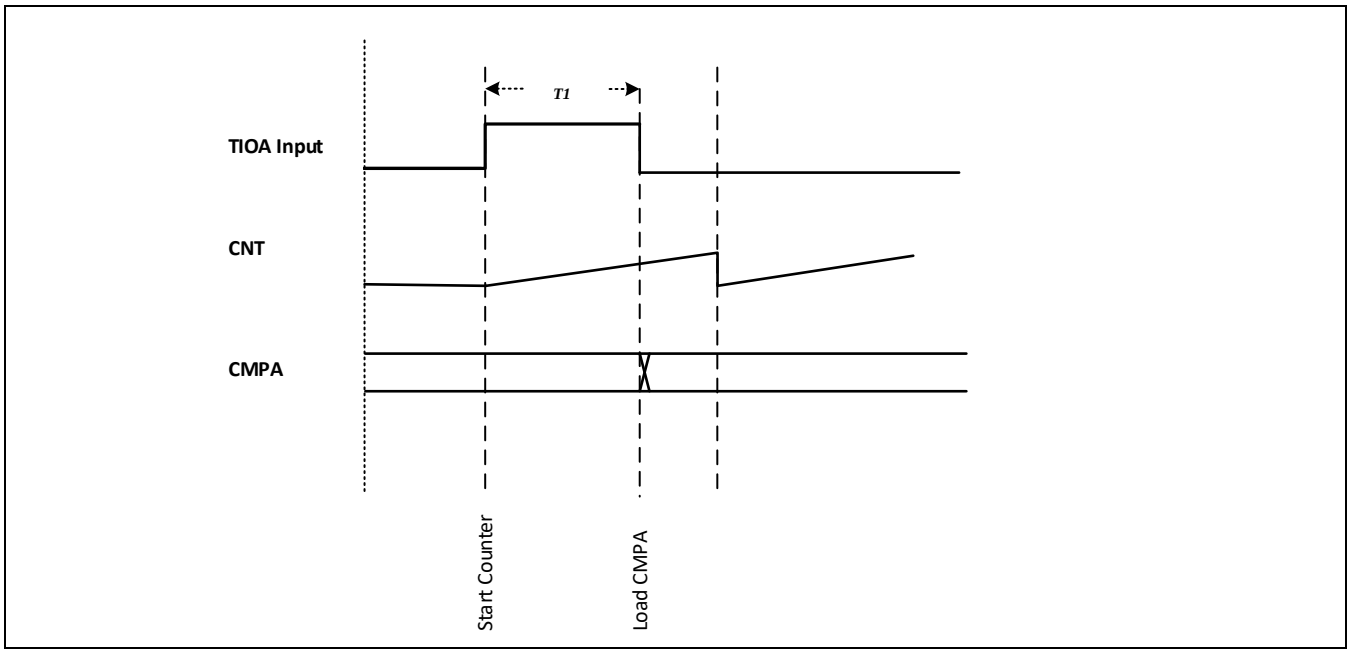


Figure 10-14 测量TIOA的脉冲宽度

10.3.7 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。

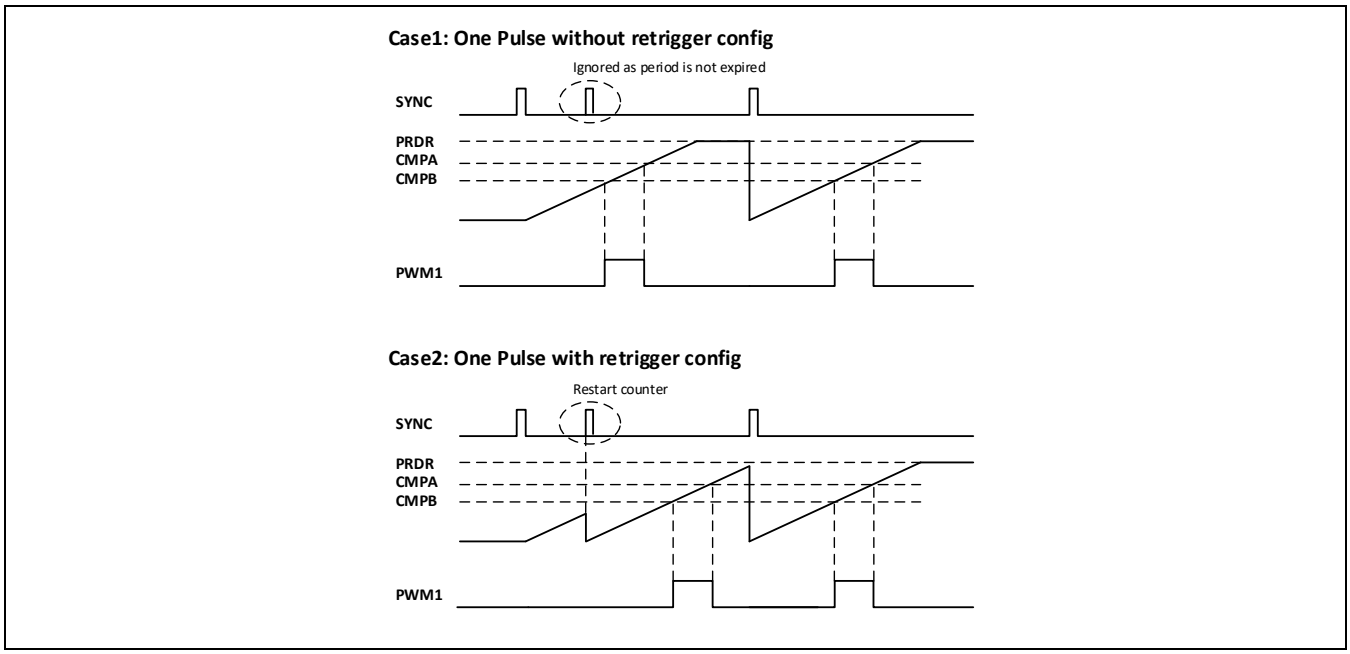


Figure 10-15 单次触发

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。

10.3.8 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。

10.3.8.1 同步触发输入接口

GPTA支持模块间的同步触发功能，可以支持的触发功能包括如下几种：

- 重置和启动计数器
- 寄存器的更新（从Shadow寄存器更新到Active寄存器）
- 当前计数器值捕捉
- 计数器值递增一个计数值
- 触发改变PWM的输出状态

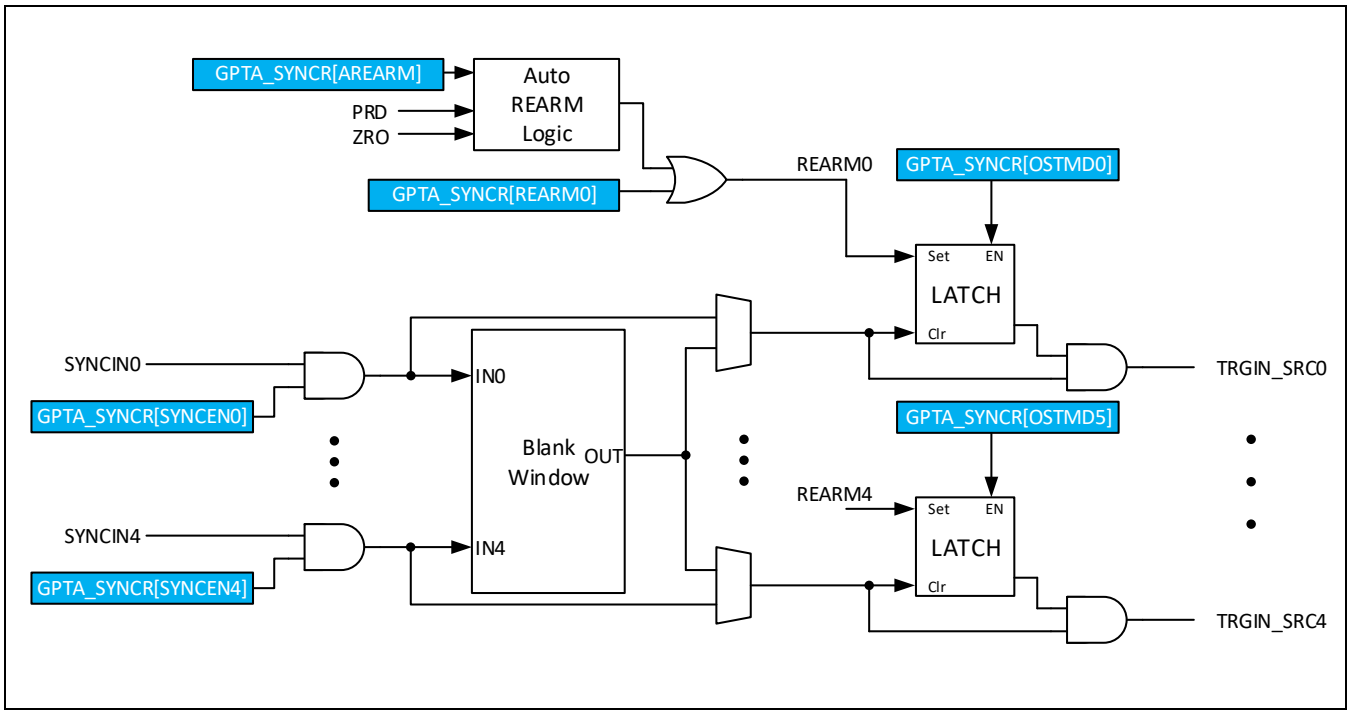


Figure 10-16 同步触发输入

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。

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

重置和启动计数器（SYNCIN0）

当该端口被触发，下列动作将被同时执行：

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

寄存器的更新（SYNCIN1）

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

计数器值捕捉（SYNCIN2）

当该端口被触发，将触发捕获事件。只有在GPTA_CR[CAP_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。在分开捕获模式下，该端口代表CAP_LDA触发事件

计数器值捕捉（SYNCIN3）

当该端口被触发，将触发捕获事件。只有在GPTA_CR[CAP_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。在分开捕获模式下，该端口代表CAP_LDB触发事件

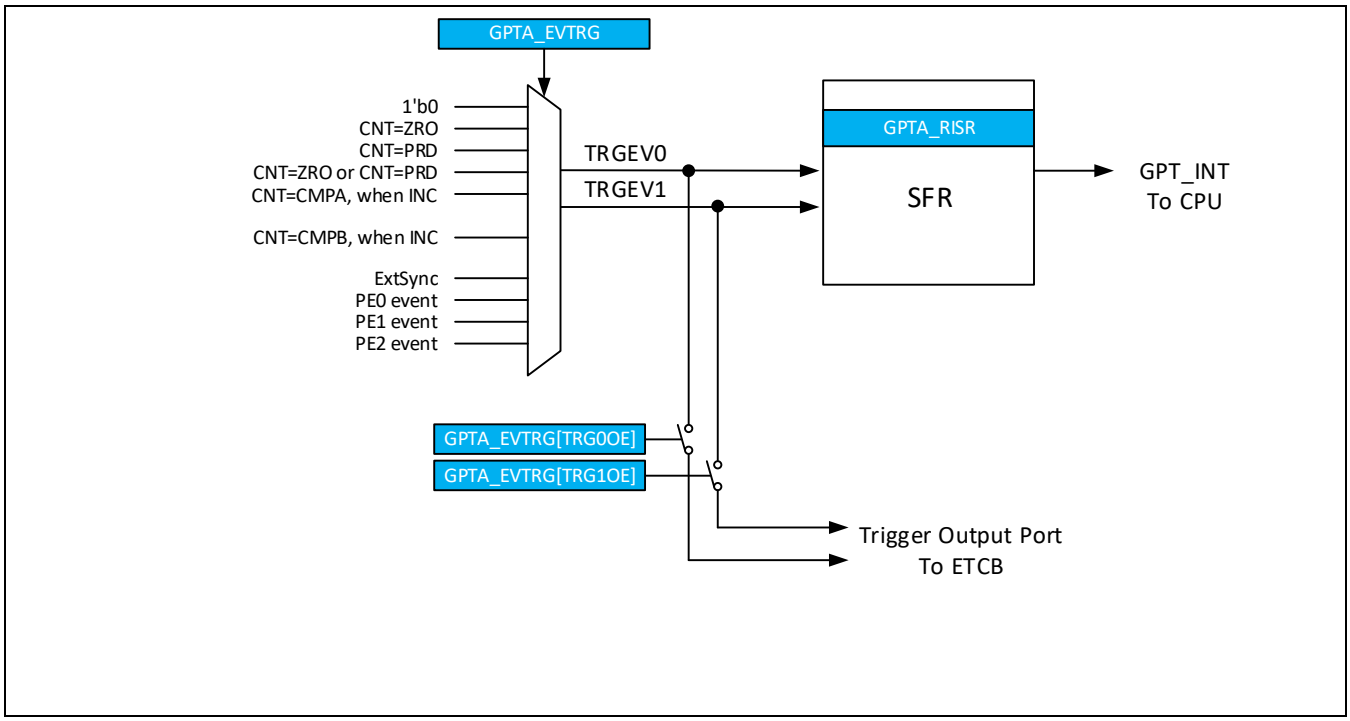
计数器值递增一个计数值（SYNCIN4）

当该端口被触发，计数器自动增加一个计数值。只有在GPTA_CEDR[CSS]控制位选择SYNCIN4时，该端口的触发才会被计数器检测到。

10.3.9 事件触发（输出）

10.3.9.1 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。



10.3.10 寄存器链接

增强型通用定时器A (GPTA)和增强型通用定时器B (GPTB)的PRDR、CMPA和CMPB等寄存器可以相互链接(具体请参考各章节的REGLK，REGLK2寄存器)。链接目标对应表如下表所示。

Table 10-4 寄存器链接对应表

链接值	链接目标
0x0	不链接
0x1	GPTA0
0x2	GPTB0

说明：如果芯片中不包含上述模块，则配置无效。

如果一个或多个定时器(以下称目标定时器)在REGLK或REGLK2中设置了PRDR或CMPA等寄存器(以下称为目标寄存器)的链接值，且该链接值指向同一个定时器(以下称源定时器)， 当程序更新源定时器的PRDR或CMPA等寄存器(以下称为源寄存器)时,目标寄存器的PRDR或CMPA等寄存器也同时更新。例如当GPTA0.REGLK.PRDR = 0x2时，当程序更新GPTB0.PRDR时，GPTA0 PRDR会同时更新为相同值。这样多个定时器的寄存器相互链接，并通过相同事件触发，可实现多个定时器的级联。

10.4 寄存器说明

10.4.1 寄存器表

Base Address of GPT: 0x40055000

Register	Offset	Description	Reset Value
GPTA_CEDR	0x0000	ID和时钟控制寄存器	0xBE980000
GPTA_RSSRn	0x0004	启停控制寄存器	0x00000000
GPTA_PSCR	0x0008	时钟分频控制寄存器	0x00000000
GPTA_CRn	0x000C	控制寄存器，捕捉模式WAVE=0	0x00010010
GPTA_CRn	0x000C	控制寄存器，波形输出模式WAVE=1	0x00010010
GPTA_SYNCRn	0x0010	同步控制寄存器	0x00000000
GPTA_PRDR	0x0024	周期设置寄存器	0x00000000
GPTA_CMPA	0x002C	比较值A寄存器	0x00000000
GPTA_CMPB	0x0030	比较值B寄存器	0x00000000
GPTA_CMPLDR	0x003C	比较值载入控制寄存器	0x00002490
GPTA_CNT	0x0040	时基计数器寄存器	0x00000000
GPTA_AQLDR	0x0044	波形输出载入控制寄存器	0x00000024
GPTA_AQCR1	0x0048	PWM1波形输出控制寄存器	0x00000000
GPTA_EVTRG	0x00C0	事件触发选择寄存器	0x00000000
GPTA_EVSWF	0x00CC	事件计数器软件触发控制寄存器	0x00000000
GPTA_RISR	0x00D0	原始中断状态寄存器	0x00000000
GPTA_MISR	0x00D4	中断状态寄存器	0x00000000
GPTA_IMCR	0x00D8	中断使能控制寄存器	0x00000000
GPTA_ICR	0x00DC	中断清除寄存器	0x00000000
GPTA_REGLK	0x00E0	寄存器链接控制器	0x00000000
GPTA_PROT	0x00E8	寄存器写保护控制器	0x00000000
GPTA_CMPAA	0x082C	比较值A active寄存器	0x00000000
GPTA_CMPBA	0x0830	比较值B active寄存器	0x00000000

10.4.2 GPTA_CEDR(ID和时钟控制寄存器)

Address = Base Address+ 0x0000, Reset Value = 0xBE980000

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

Name	Bit	Type	Description
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SYNCIN4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。

10.4.3 GPTA_RSSR(PORT)(启停控制寄存器)

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																SRR				RSVD											START
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	R	R	R	R	R	R	R	R	R	R	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。
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，需要先在PROT中写入KEY值解锁，才能写入。			

10.4.4 GPTA_PSCR(时钟分频控制寄存器)

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
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。

10.4.5 GPTA_CR(WAVE=0)(控制寄存器，捕捉模式WAVE=0)

Address = Base Address+ 0x000C, Reset Value = 0x00010010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAP_WAVE		RSVD		LDBARST	LDAARST	LDBRST	LDARST	STOP_WRAP		CAPMD	REARM	CAPMD_SEL	RSVD	PSCLD		CGFLT			RSVD		FLTIPSCLD	BURST	CAPLDEN	RSVD		PRDLD		RSVD	SWSYNEN	RSVD	
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	1	0	0	0	0
RW	RW	R	R	RW	RW	RW	RW	RW	RW	RW	W	RW	R	RW	RW	RW	RW	RW	R	R	RW	RW	RW	R	R	RW	RW	R	RW	R	R

Name	Bit	Type	Description
CAP_WAVE	[31:30]	RW	捕获与波形选择位 00b：CMPB用于捕获，CMPA用于捕获。 01b：CMPB用于波形，CMPA用于捕获。 10b：CMPB用于捕获，CMPA用于波形。 11b：CMPB用于波形，CMPA用于波形。
LDBARST	[27]	RW	CMPB(Active)捕捉载入后，计数器值计数状态控制位。 0h：当前捕获触发后，计数器值不进行重置 1h：当前捕获触发后，计数器值进行重置
LDAARST	[26]	RW	CMPA(Active)捕捉载入后，计数器值计数状态控制位。 0h：当前捕获触发后，计数器值不进行重置 1h：当前捕获触发后，计数器值进行重置
LDBRST	[25]	RW	CMPB(Shadow)捕捉载入后，计数器值计数状态控制位。 0h：当前捕获触发后，计数器值不进行重置 1h：当前捕获触发后，计数器值进行重置
LDARST	[24]	RW	CMPA(Shadow)捕捉载入后，计数器值计数状态控制位。 0h：当前捕获触发后，计数器值不进行重置 1h：当前捕获触发后，计数器值进行重置
STOP_WRAP	[23:22]	RW	Capture模式下，捕获事件计数器周期设置值。
CAPMD	[21]	RW	捕捉模式设置。 0h：连续捕捉模式 1h：一次性捕捉模式
REARM	[20]	W	重置CAPTURE控制。 0h：无效 1h：重置捕捉 重置时，捕捉事件计数器被清零，自动打开CAPLDEN
CAPMD_SEL	[19]	RW	捕捉模式选择。 0h：合并捕捉模式 注：不区分SYNCIN2/3的触发，支持4次捕获，捕获值分别存入CMPA、CMPB、CMPAA、CMPBA。 1h：分开捕捉模式 注：区分SYNCIN2/3的触发，

			CAP_WAVE = 0时，SYNCIN2的触发，捕捉值存入CMPA； SYNCIN3的触发，捕捉值存入CMPB CAP_WAVE = 1时，SYNCIN2的触发，捕捉值存入CMPA； SYNCIN3的触发，捕捉值存入CMPAA CAP_WAVE = 2时，SYNCIN2的触发，捕捉值存入CMPB； SYNCIN3的触发，捕捉值存入CMPBA
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
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/B在捕捉事件触发时，载入使能控制。 0h：禁止对CMPA/B寄存器的捕获载入 1h：使能对CMPA/B寄存器的捕获载入 此控制位在禁止对CMPA/B寄存器载入时，并不影响捕捉事件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计数器值等于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]
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。

10.4.6 GPTA_CR(WAVE=1)(控制寄存器，波形输出模式WAVE=1)

Address = Base Address+ 0x000C, Reset Value = 0x00010010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAP_WAVE		RSVD												PSCLD		CGFLT			RSVD		CKS	BURST	RSVD		OPM	PRDLD		IDLEST	SWSYNEN	RSVD	
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	1	0	0	0	0
RW	RW	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	R	R	RW	RW	R	R	RW	RW	RW	RW	RW	R	R

Name	Bit	Type	Description
CAP_WAVE	[31:30]	RW	捕获与波形选择位 00b：CMPB用于捕获，CMPA用于捕获。 01b：CMPB用于波形，CMPA用于捕获。 10b：CMPB用于捕获，CMPA用于波形。 11b：CMPB用于波形，CMPA用于波形
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
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
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在计数器值等于零或SYNCIN1触发时 11b：立即更新，所有对PRDR操作直接作用于活动寄存器 [1]
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。
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。			

10.4.7 GPTA_SYNCR(PORT)(同步控制寄存器)

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
AREARM		TRGO1SEL			TRGO0SEL			RSVD				REARM4	REARM3	REARM2	REARM1	REARM0	RSVD		OSTMD4	OSTMD3	OSTMD2	OSTMD1	OSTMD0	RSVD			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	R	R	R	R	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	R	R	R	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TRGSEL1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 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TRGSEL0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 其他：保留
REARM4~REARM0	[19:15]	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
OSTMD4~OSTMD0	[12:8]	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。
SYNCEN4~SYNCEN0	[4:0]	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，分开捕捉模式下代表CAP_LDA触发事件 SYNCIN3：Capture触发事件，分开捕捉模式下代表CAP_LDB触发事件 SYNCIN4：CNT增一拍触发事件
NOTE: 该寄存器受REGPROT保护，需要先在PROT中写入KEY值解锁，才能写入。			

10.4.8 GPTA_PRDR(周期设置寄存器)

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
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。

10.4.9 GPTA_CMPA(比较值A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
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CMPLDR[SHDWCMPA]进行设置。在Shadow模式下，可以通过CMPLDR[LDAMD]选择Shadow到Active载入的触发条件。在写入前，可以通过SHDWAFULL控制位检测当前寄存器状态。 当工作于Capture模式下，此寄存器对应CAPLD0事件触发的捕获值。

10.4.10 GPTA_CMPB(比较值B寄存器)

Address = Base Address+ 0x0030, 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CMPLDR[SHDWCMPB]进行设置。在Shadow模式下，可以通过CMPLDR[LDBMD]选择Shadow到Active载入的触发条件。在写入前，可以通过SHDWBFULL控制位检测当前寄存器状态。 当工作于Capture模式下，此寄存器对应CAPLD1事件触发的捕获值。

10.4.11 GPTA_CMPLDR(比较值载入控制寄存器)

Address = Base Address+ 0x003C, 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										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	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	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。			

10.4.12 GPTA_CNT(时基计数器寄存器)

Address = Base Address+ 0x0040, 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。

10.4.13 GPTA_AQLDR(波形输出载入控制寄存器)

Address = Base Address+ 0x0044, 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																											SHDWLD1MD			RSVD	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	R	R	R	RW	RW	RW	R	RW

Name	Bit	Type	Description
SHDWLD1MD	[4:2]	RW	Shadow模式下，Active AQCRA从Shadow载入控制。 xx1b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 x1xb：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 1xxb：外部LOAD触发或SYNC触发时，Shadow寄存器载入到Active寄存器中 000b：不进行载入 每个控制位分别对应一个触发条件，可以同时使能多个触发条件。例如，当设置011b时，CNT=ZRO或CNT=PRD时，都会触发寄存器载入。
LDAQ1MD	[0]	RW	AQCRA寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式

10.4.14 GPTA_AQCR1(PWM1波形输出控制寄存器)

Address = Base Address+ 0x0048, 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		RSVD										C2U		RSVD		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	R	R	R	R	R	R	R	R	R	R	RW	RW	R	R	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。
C2U	[9:8]	RW	当CNT值等于C2，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，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。 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。 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）

10.4.15 GPTA_EVTRG(事件触发选择寄存器)

Address = Base Address+ 0x00C0, 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												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	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
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
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 0110：当 CNT = CMPB 且计数方向为递增时，产生TRGEV事件 1100：ExtSync通道 1101：PE0 event 1110：PE1 event 1111：PE2 event others：RSVD
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 0110：当 CNT = CMPB 且计数方向为递增时，产生TRGEV事件 1100：ExtSync通道 1101：PE0 event 1110：PE1 event 1111：PE2 event others：RSVD

10.4.16 GPTA_EVSWF(事件计数器软件触发控制寄存器)

Address = Base Address+ 0x00CC, 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

10.4.17 GPTA_RISR(原始中断状态寄存器)

Address = Base Address+ 0x00D0, 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					CBU	RSVD	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
CBU	[10]	R	CNT = CMPB中断请求原始标志状态
CAU	[8]	R	CNT = CMPA中断请求原始标志状态
CAP_LD3	[7]	R	捕获值到CMPB Active中断请求原始标志状态
CAP_LD2	[6]	R	捕获值到CMPA Active中断请求原始标志状态
CAP_LD1	[5]	R	捕获值到CMPB Shadow中断请求原始标志状态
CAP_LD0	[4]	R	捕获值到CMPA Shadow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			

10.4.18 GPTA_MISR(中断状态寄存器)

Address = Base Address+ 0x00D4, 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					CBU	RSVD	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
CBU	[10]	R	CNT = CMPB中断请求标志状态
CAU	[8]	R	CNT = CMPA中断请求标志状态
CAP_LD3	[7]	R	捕获值到CMPA Active中断请求标志状态
CAP_LD2	[6]	R	捕获值到CMPA Active中断请求标志状态
CAP_LD1	[5]	R	捕获值到CMPB Shadow中断请求标志状态
CAP_LD0	[4]	R	捕获值到CMPA Shadow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			

10.4.19 GPTA_IMCR(中断使能控制寄存器)

Address = Base Address+ 0x00D8, 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					CBU	RSVD	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	R	RW	R	RW	RW	RW	RW	RW	R	R	RW	RW

Name	Bit	Type	Description
PEND	[16]	RW	周期结束中断使能控制位
CBU	[10]	RW	CNT = CMPB中断使能控制位
CAU	[8]	RW	CNT = CMPA中断使能控制位
CAP_LD3	[7]	RW	捕获值到CMPB Active中断使能控制位
CAP_LD2	[6]	RW	捕获值到CMPA Active中断使能控制位
CAP_LD1	[5]	RW	捕获值到CMPB Shadow中断使能控制位
CAP_LD0	[4]	RW	捕获值到CMPA Shadow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
CPU中断请求使能控制。当该控制位使能时，允许触发CPU中断。 0h：禁止该中断 1h：允许该中断			

10.4.20 GPTA_ICR(中断清除寄存器)

Address = Base Address+ 0x00DC, 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					CBU	RSVD	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	R	W	R	W	W	W	W	R	R	W	W

Name	Bit	Type	Description
PEND	[16]	W	周期结束中断清除控制位
CBU	[10]	R	CNT = CMPB中断清除控制位
CAU	[8]	R	CNT = CMPA中断清除控制位
CAP_LD3	[7]	W	捕获值到CMPB Active中断清除控制位
CAP_LD2	[6]	W	捕获值到CMPA Active中断清除控制位
CAP_LD1	[5]	W	捕获值到CMPB Shadow中断清除控制位
CAP_LD0	[4]	W	捕获值到CMPA Shadow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’			

10.4.21 GPTA_REGLK(寄存器链接控制器)

Address = Base Address+ 0x00E0, 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				RSSR				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	RW	RW	RW	RW	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
RSSR	[27:24]	RW	RSSR寄存器链接目标。
CMPB	[11:8]	RW	CMPB寄存器链接目标。
CMPA	[7:4]	RW	CMPA寄存器链接目标。
PRDR	[3:0]	RW	PRDR寄存器链接目标。
连接到相应的定时器。 0h:不链接 1h:GPTA0 2h:GPTB0			

10.4.22 GPTA_PROT(寄存器写保护控制器)

Address = Base Address+ 0x00E8, 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。

10.4.23 GPTA_CMPAA(比较值A active寄存器)

Address = Base Address+ 0x082C, 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	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

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	[14:0]	R	比较值A Active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。

10.4.24 GPTA_CMPBA(比较值B active寄存器)

Address = Base Address+ 0x0830, 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	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

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。

11 增强型通用定时器B (GPTB)

11.1 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。GPTB 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。

11.1.1 主要特性

- 16 位可复位计数器
- 支持递增计数(Up-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
  - 1 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
- 支持捕获模式，最多支持 4 个捕获值存储。

11.1.2 管脚描述

下表列出了不同模式下的管脚定义。

Table 11-1 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
GPT_CHAX	时钟控制使能	输出波形	输出波形
GPT_CHAY	NA	输出波形	输出波形
GPT_CHB	时钟控制使能	输出波形	输出波形

11.2 功能描述

11.2.1 模块框图

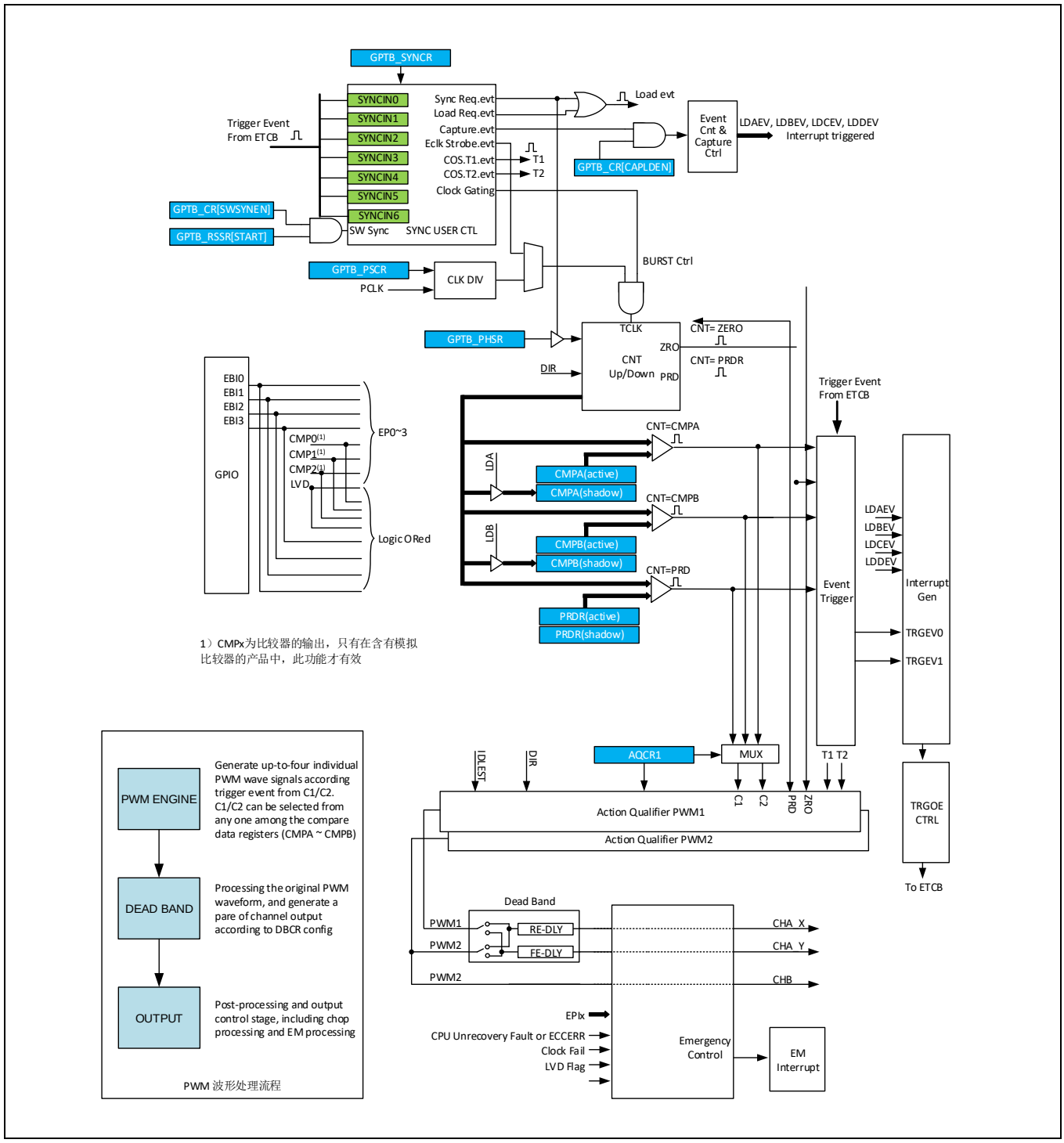


Figure 11-1 模块结构示意图

11.3 基本功能描述

一个完整的GPT模块由两个TIMER输入/输出通道组成。多个GPTB可以通过ETCB链连接，通过ETCB链，实现多个GPTB的同步工作。在包含多个GPTB的器件中，以数字后缀区分不同的实例，例如：GPTB0代表第一个GPTB模块，GPTB1代表第二个GPTB模块。每个GPTB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、捕获控制模块、事件触发模块、紧急处理模块和同步触发控制模块。

CHAX和CHB是GPTB在GPIO上映射的双向输入输出端口。在波形输出模式下，CHAX和CHB作为PWM信号的输出端口，在群脉冲模式下(GPT_CR[BURST]使能)，CHAX或CHB可以作为门控时钟的时钟控制输入信号。EBIx为外部GPIO上映射的输入功能，可以作为紧急状态处理的触发信号。

11.3.1 时钟源

11.3.1.1 概述

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

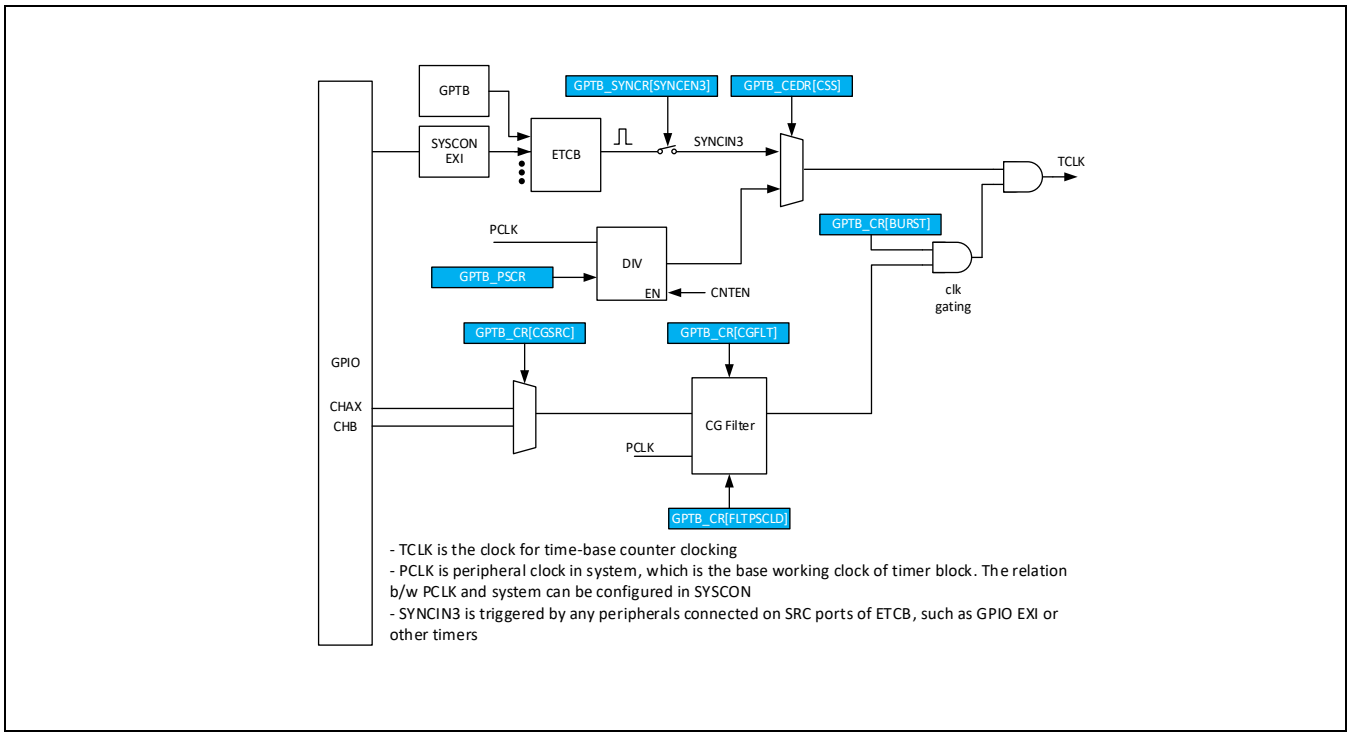


Figure 11-2 时钟控制模块

11.3.1.2 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。

11.3.1.3 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GPTB_PSCR进行设置。在对GPTB_PSCR进行读写时，操作的对象为PSCR的影子寄存器(Shadow Register),当时基计数器的值等于零时，影子寄存器的值将被更新到内部的活动寄存器中(Active Register)。当对GPTB_PSCR更新后，新的分频将在下一个计数周期开始时有效。

11.3.1.4 群脉冲时钟

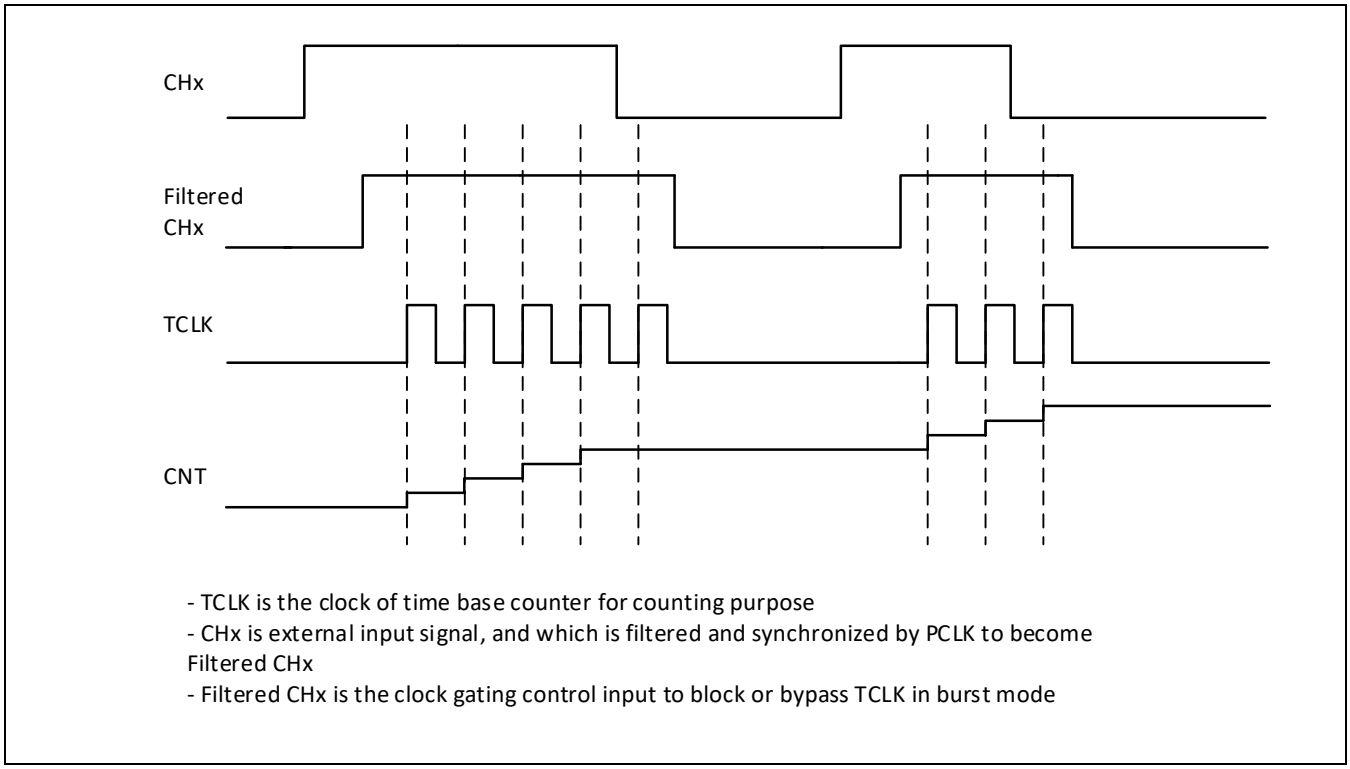


Figure 11-3 群脉冲时钟模式时序

在群脉冲时钟模式下GPTB_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GPTB_CR[CGSRC]控制位进行选择，支持CHAX通道或者CHB的外部输入作为门控信号。当CHAX或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GPTB_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]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：Tdly = Tfltclk x CR[CGFLT] = ((CEDR[FLTCKPRS]+1)/PCLK) x CR[CGFLT]，其中Tfltclk为滤波器工作时钟的周期。

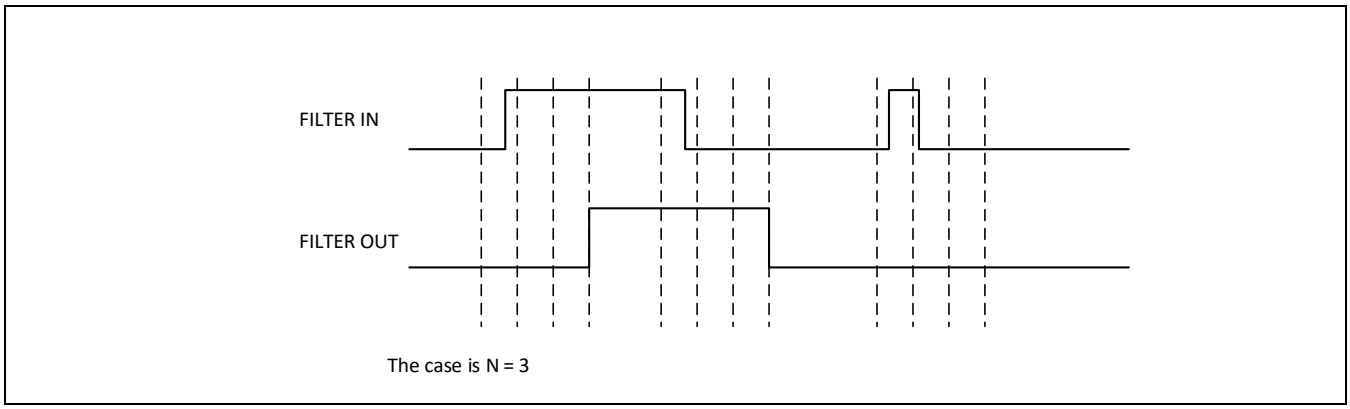


Figure 11-4 CG Filter 数字滤波器的原理

11.3.2 时基控制

11.3.2.1 概述

作为GPTB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(GPTB_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GPTB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(GPTB_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(GPTB_PRDR)：计数器周期控制寄存器。

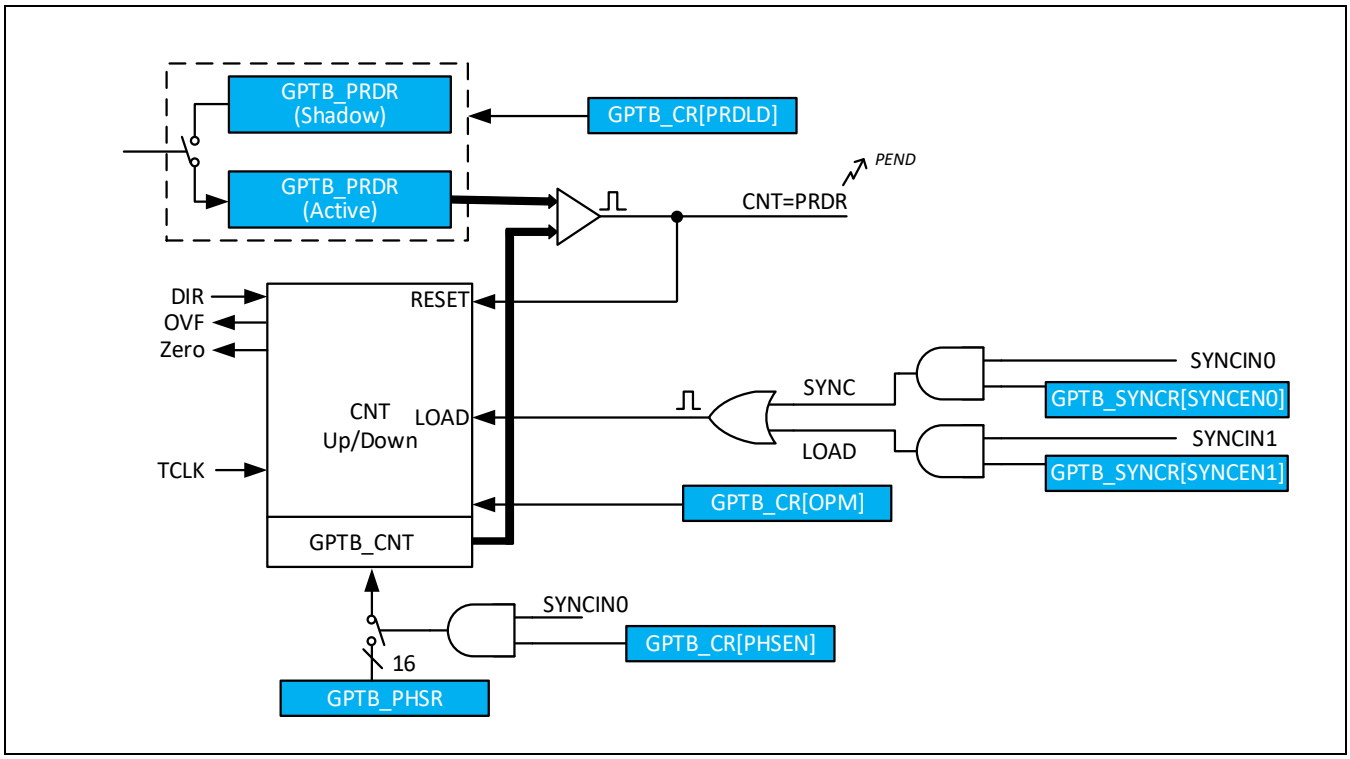


Figure 11-5 计数器时基模块

计数器的计数周期由周期寄存器(GPTB_PRDR)的设置值决定，计数器支持一种计数模式：

- 递增模式(Up-Counting Mode)：

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

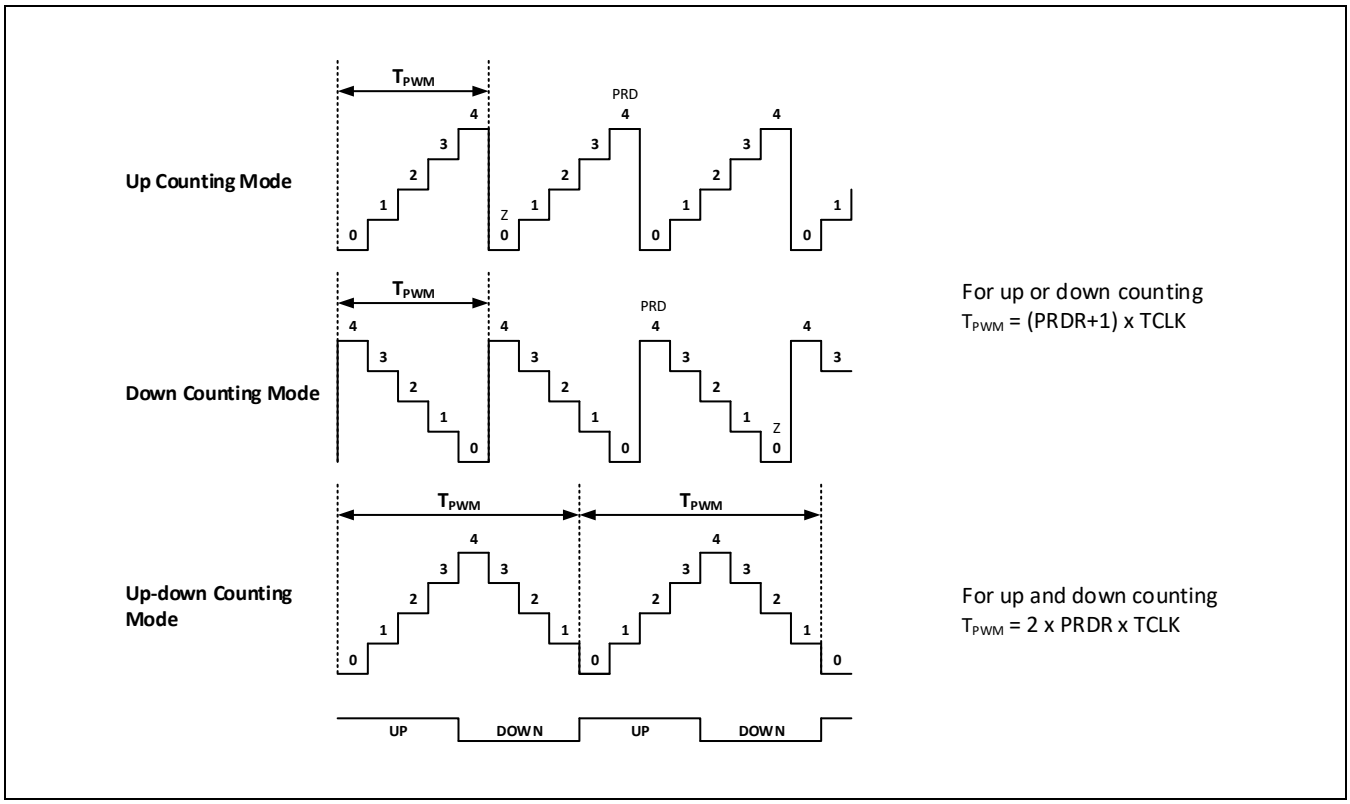


Figure 11-6 计数器工作模式

11.3.2.2 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，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。
- 计数值等于0x0000:当周期开始计数且当前计数值等于零，计数器的值将被重置到PRDR所设置的数值。
- 软件直接更新:通过软件直接写入计数器活动寄存器进行更新。

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

11.3.2.3 计数模式和时序

时基计数器可以分为2种工作模式：

- 递增计数模式(非对称)
- 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。

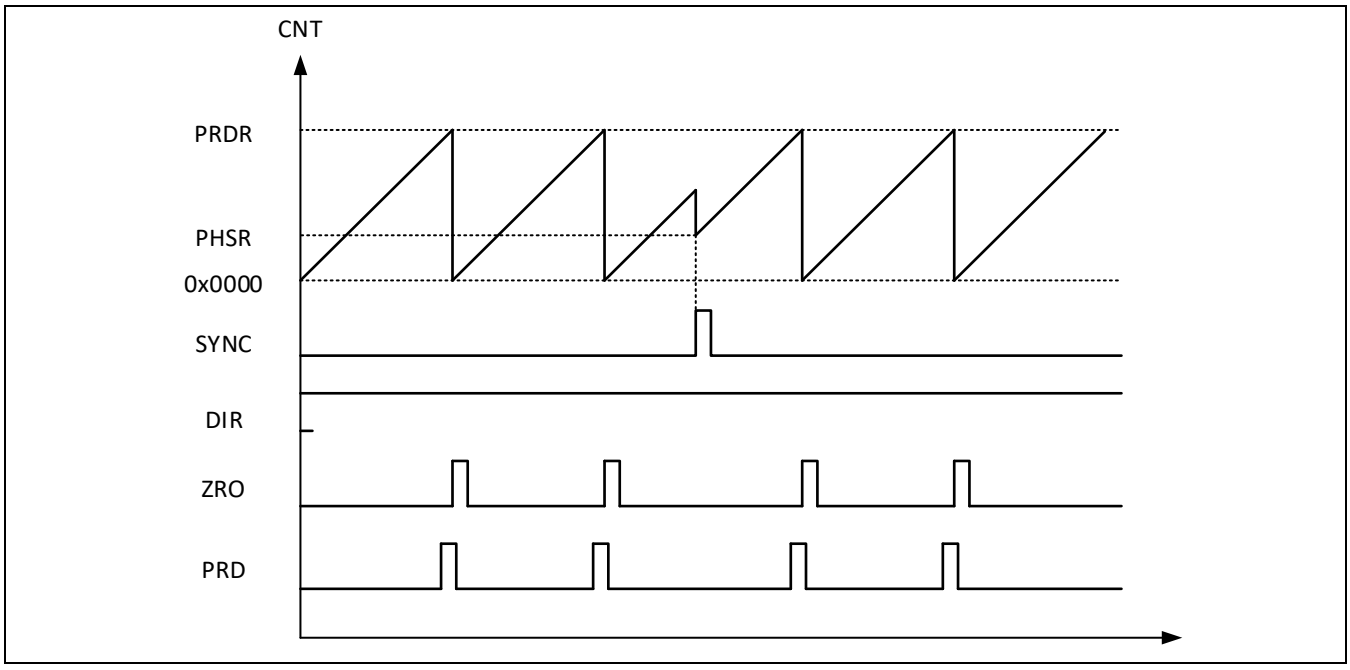


Figure 11-7 递增工作模式

11.3.2.4 全局载入控制

GPTB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[OSMD]=1)，只在全局条件满足时，触发一次全局载入，后续的载入将被屏蔽。必须通过软件重新初始化后，才能进行新的触发。通过设置GLDCR2[GFRCLD[控制位，软件可以强制触发全局载入。

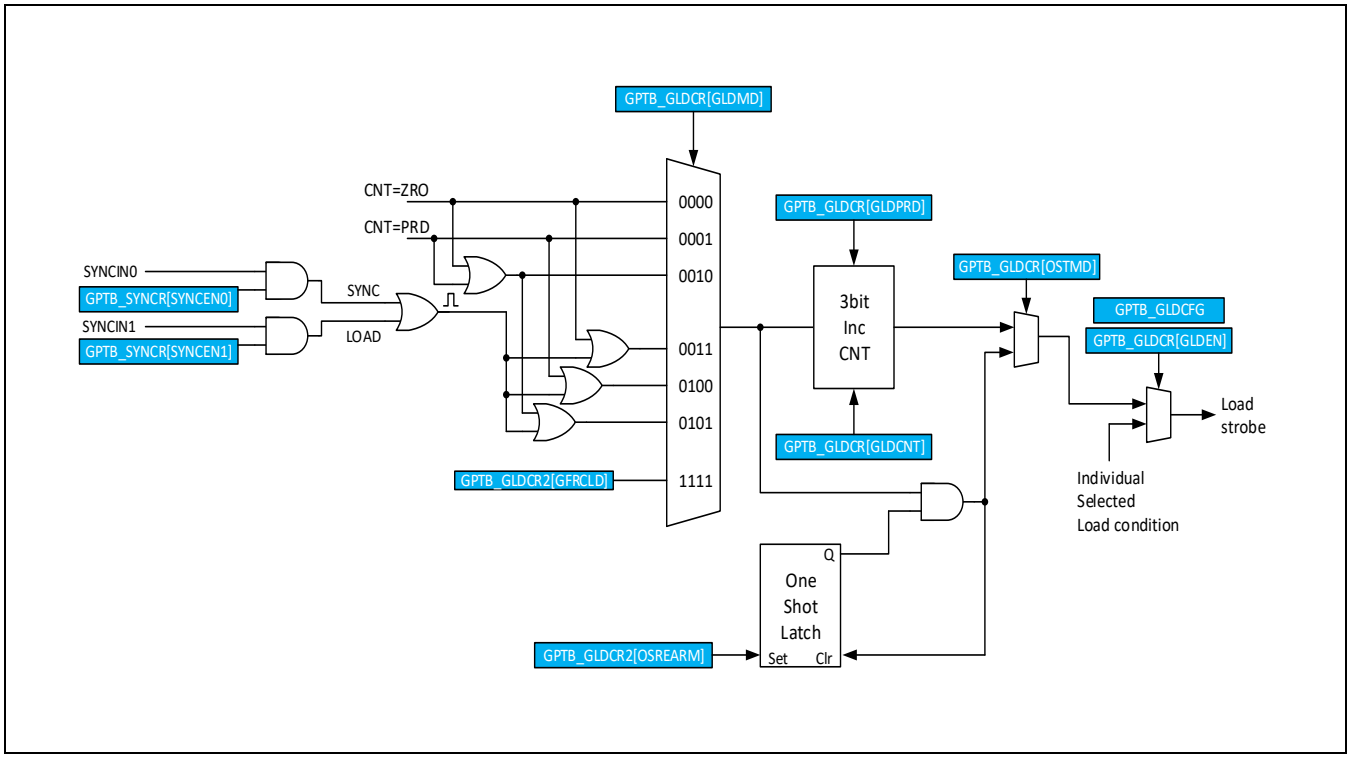


Figure 11-8 全局载入控制

11.3.3 计数器数值比较控制

11.3.3.1 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)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
- PWM的波形控制基于CMPA和CMPB
- 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

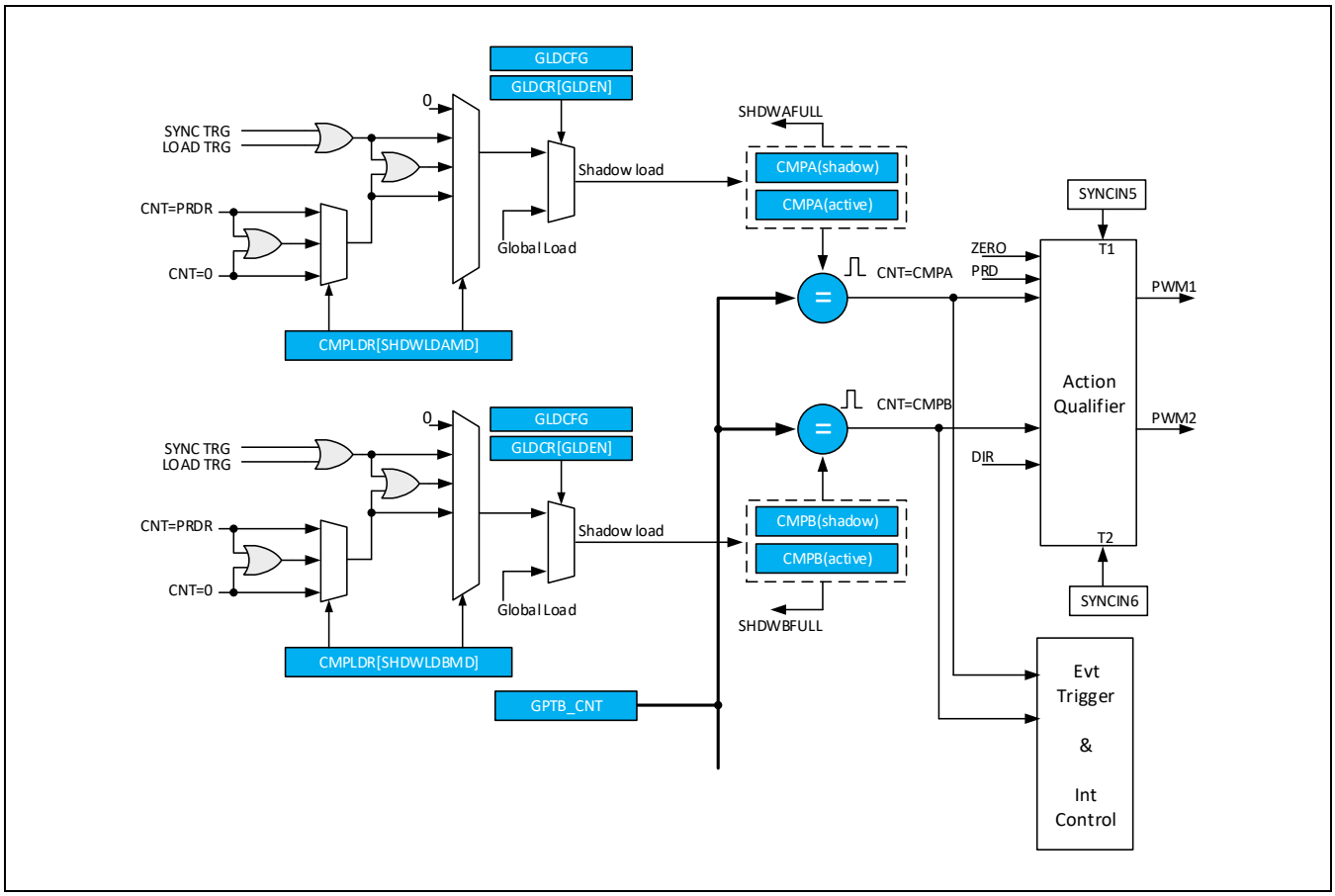


Figure 11-9 计数器值比较控制

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

在递增模式模式下，每个比较事件在一个计数周期内只会发生一次。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。

11.3.3.2 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(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。

11.3.3.3 不同计数模式的时序

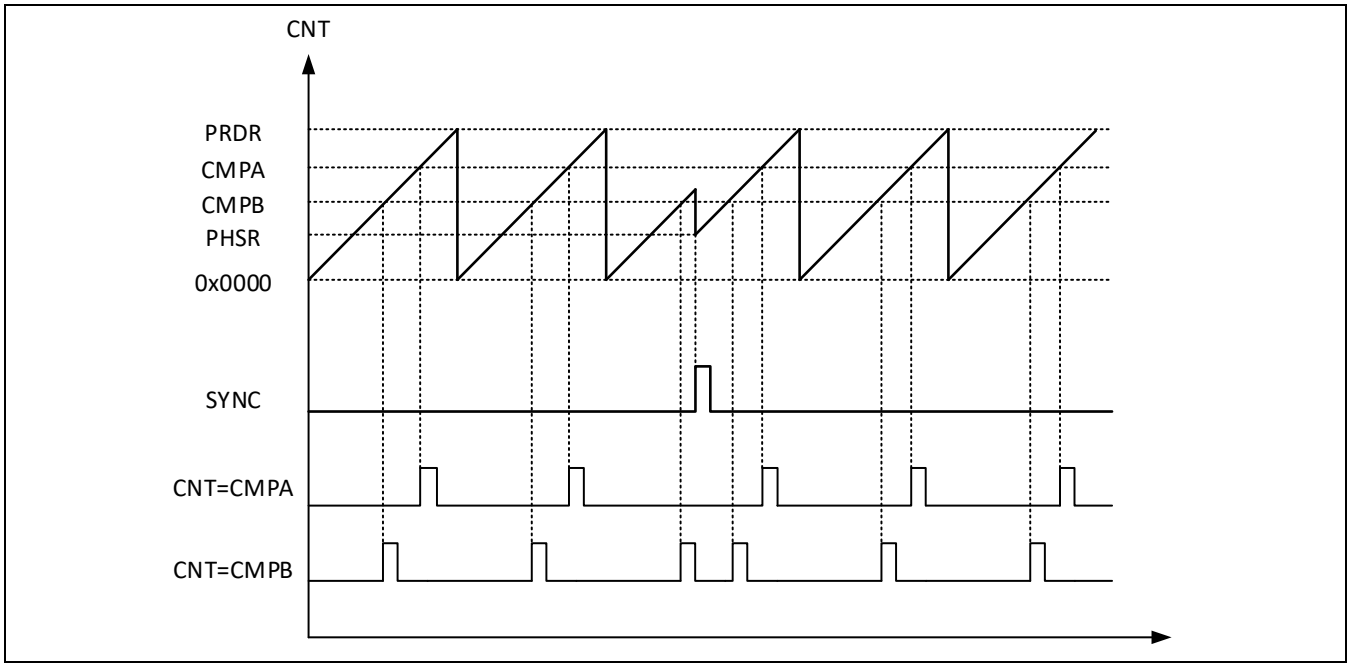


Figure 11-10 递增模式下比较事件产生时序

11.3.4 波形发生控制

11.3.4.1 事件驱动的波形输出

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

- CNT = PRD (计数器值等于周期设置值)
- CNT = ZERO (计数器值等于零)
- CNT = CMPA (计数器值等于CMPA设置值)
- CNT = CMPB (计数器值等于CMPB设置值)
- 软件Force事件 (通过软件触发的异步强制置位)

波形发生模块根据计数器的当前计数方向和发生的事件，决定PWMA和PWMB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)

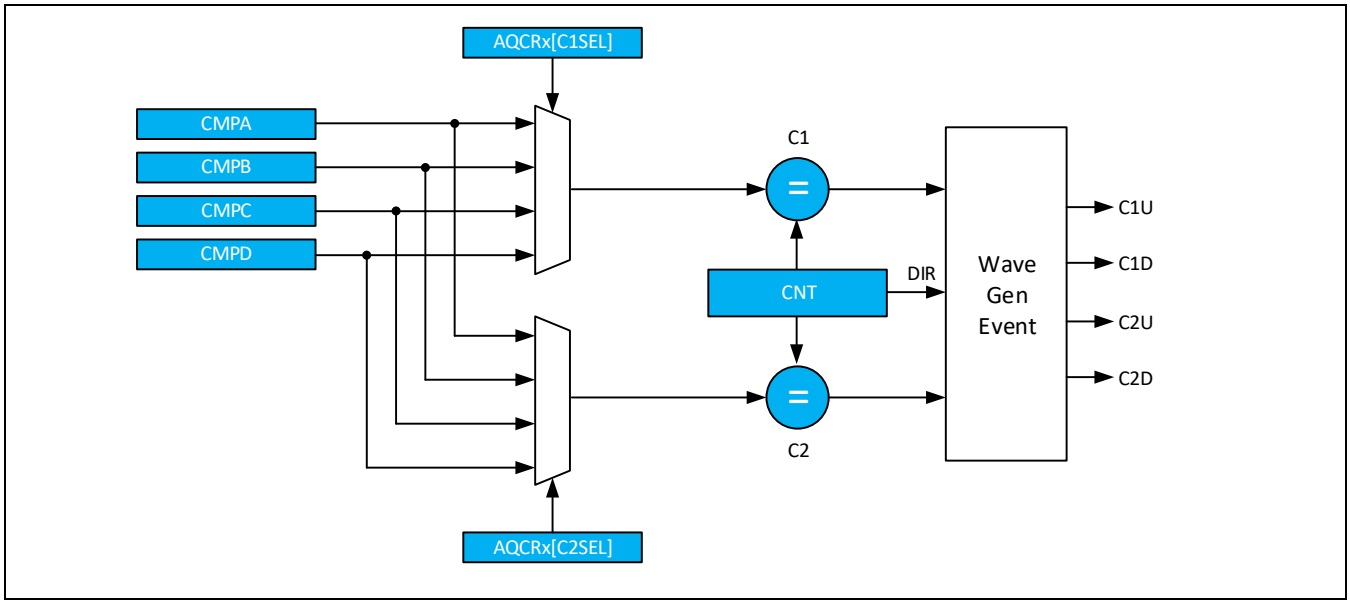


Figure 11-11 C1和C2选择控制

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 11-2 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	CMPA	CMPB	PRD	T1	T2	
SW —	Z —	C1 —	C2 —	P —	T1 —	T2 —	没有动作 (保持原来状态)
SW ⌊	Z ⌊	C1 ⌊	C2 ⌊	P ⌊	T1 ⌊	T2 ⌊	低电平输出
SW ⌈	Z ⌈	C1 ⌈	C2 ⌈	P ⌈	T1 ⌈	T2 ⌈	高电平输出
SW X	Z X	C1 X	C2 X	P X	T1 X	T2 X	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。

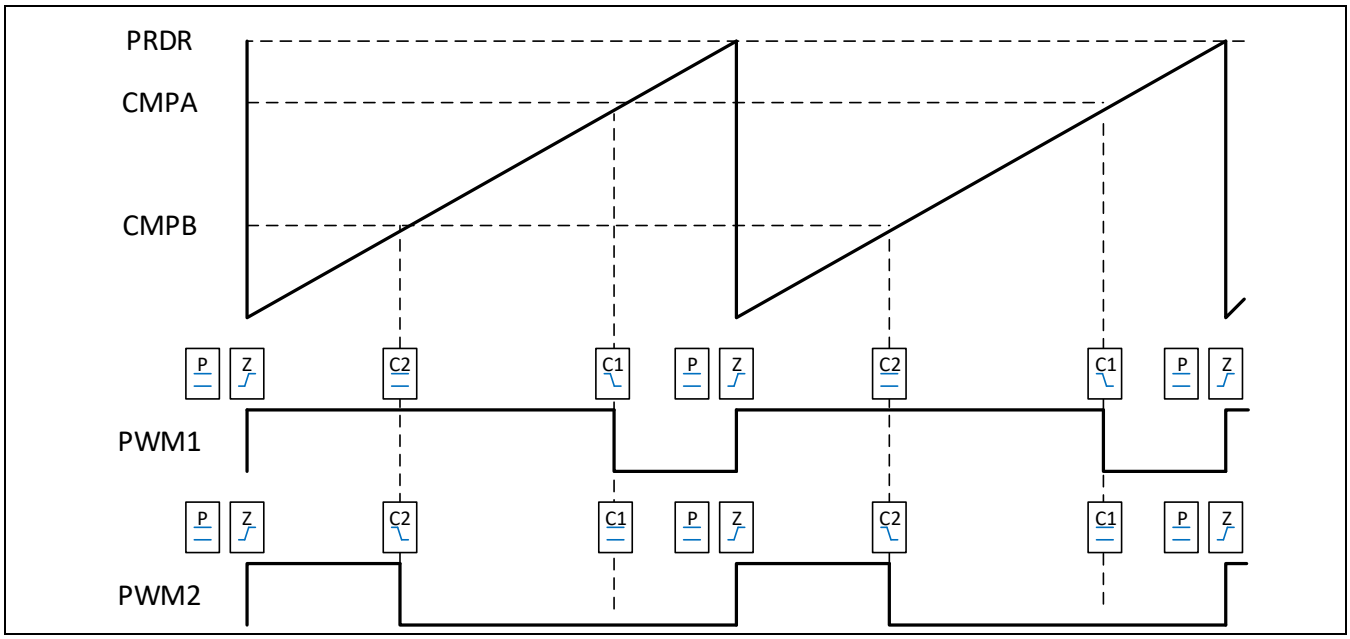


Figure 11-12 递增单沿，非对称波形输出

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。

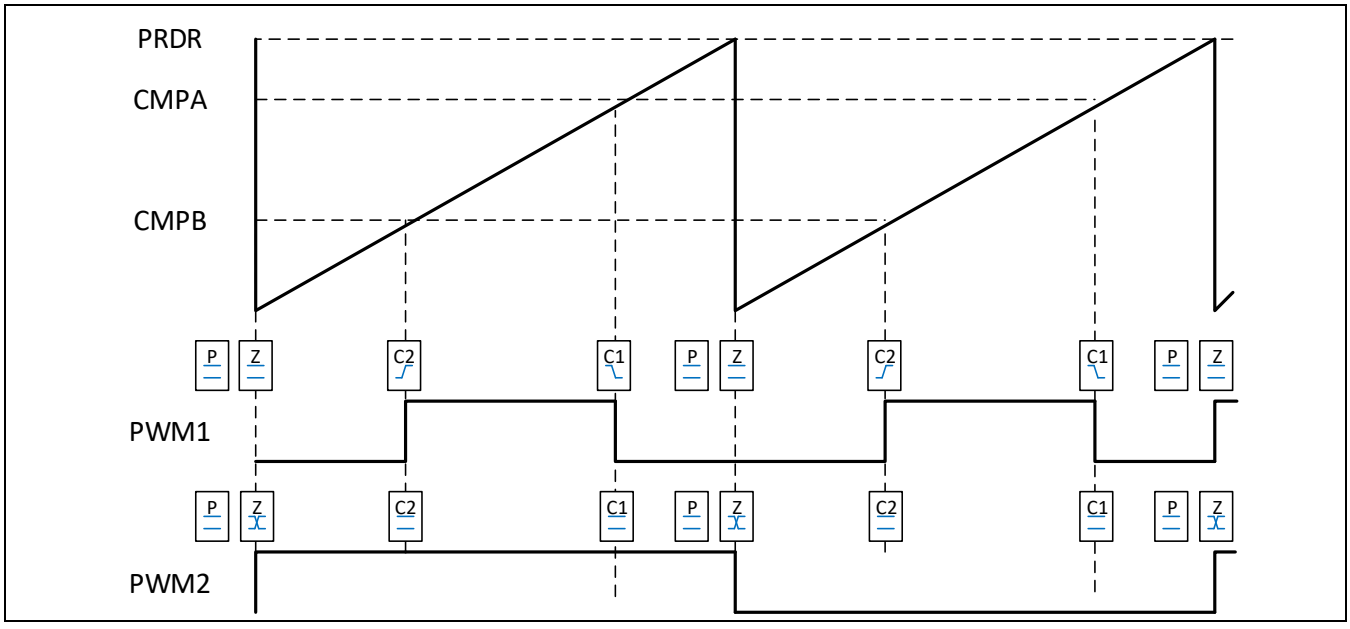


Figure 11-13 递增，脉冲定位非对称波形输出

11.3.4.2 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 11-3 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 equals period
3	RSVD
4	RSVD
5	CNT equals CMPB on up-count(C2U)
6	CNT equals CMPA on up-count(C1U)
7(Lowest)	CNT equals zero

用户可以随意设置CMPA和CMPB的值，当设置的CMP值大于Period的设置值时，将会按照下述方式进行操作。计数器设置为递增模式时, C1D和C1U/C2U事件始终不会被触发。

11.3.4.3 通过软件强制设置波形

软件强制输出可以将输出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。

PWM的波形输出支持两种通过软件进行控制的方式。一种方式控制通道输出，一种方式直接控制到管脚的输出。

11.3.4.3.1 控制通道输出

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通道PWM1/2(注意不是最终管脚上的输出波形，位置见模块框图)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GPTB_AQOSF[ACT1/2]控制位，设置在一次性强制输出触发时，PWM1/2的输出电平状态。通过对GPTB_AQOSF[OSTSF1/2]控制位写入1，触发一次性强制输出。

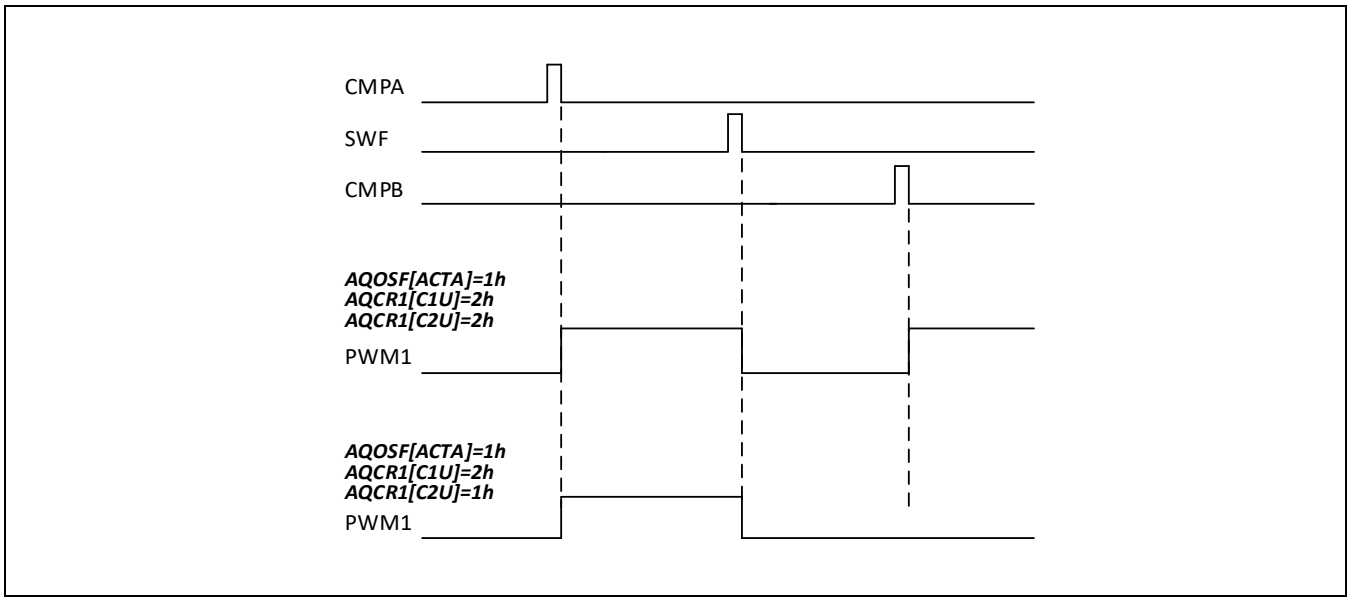


Figure 11-14 一次性软件强制输出

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]控制位，进行强制输出设置或者清除。

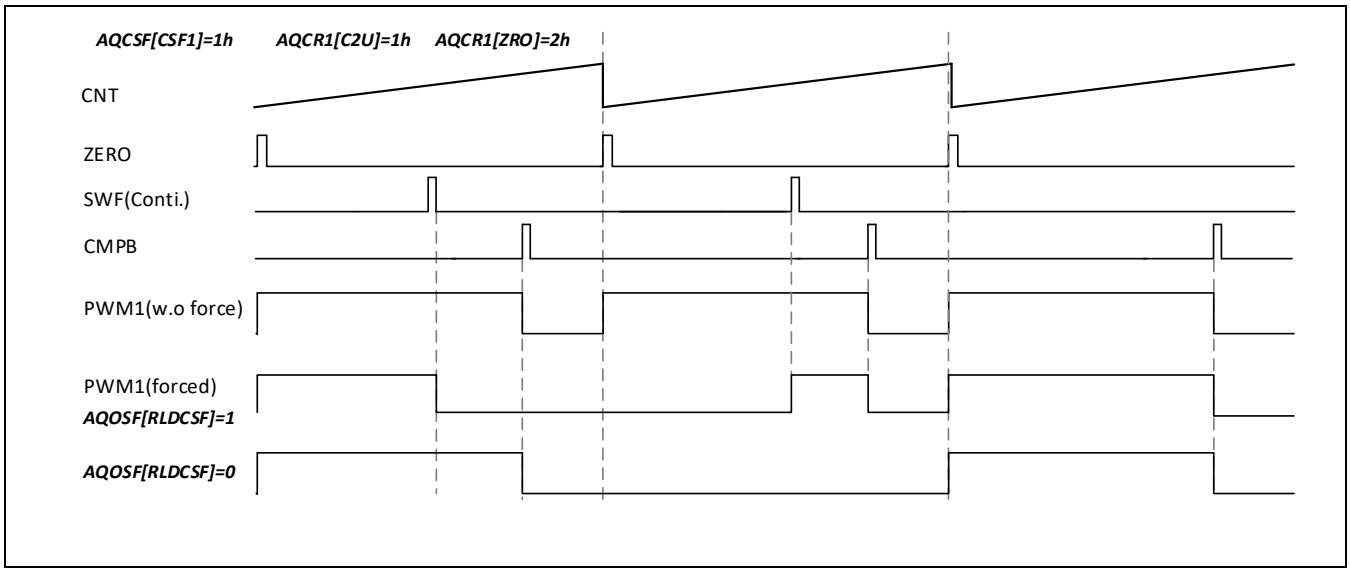


Figure 11-15 持续性软件强制输出

11.3.4.3.2 控制到管脚的输出

软件强制控制到管脚的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GPTB_CHAX、GPTB_CHAY和GPTB_CHB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GPTB_CXOSF[ACTAX/ACTAY/ACTB]控制位，设置在一次性强制输出触发时，GPTB_CHAX、GPTB_CHAY和GPTB_CHB呈现的电平状态。通过对GPTB_CXOSF [OSTSFAX/ OSTSFAY/OSTSFB]控制位写入1，触发一次性强制输出。

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GPTB_CHAX、GPTB_CHAY和GPTB_CHB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。只要往寄存器GPTB_CXCSF[CSFAX/CSFAY/CSFB]控制位写入0或1，即实现强制输出。

为了增强安全性，软件控制到管脚的强制输出还支持MASK功能，预先在GPTB_CXMSK中相应管脚位置写入屏蔽值（不允许输出的状态），无论是一次性或连续强制输出时，当软件设置的电平和MASK内容一致时，软件强制输出功能无效，且管脚将自动呈现高阻状态。

11.3.4.4 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）。

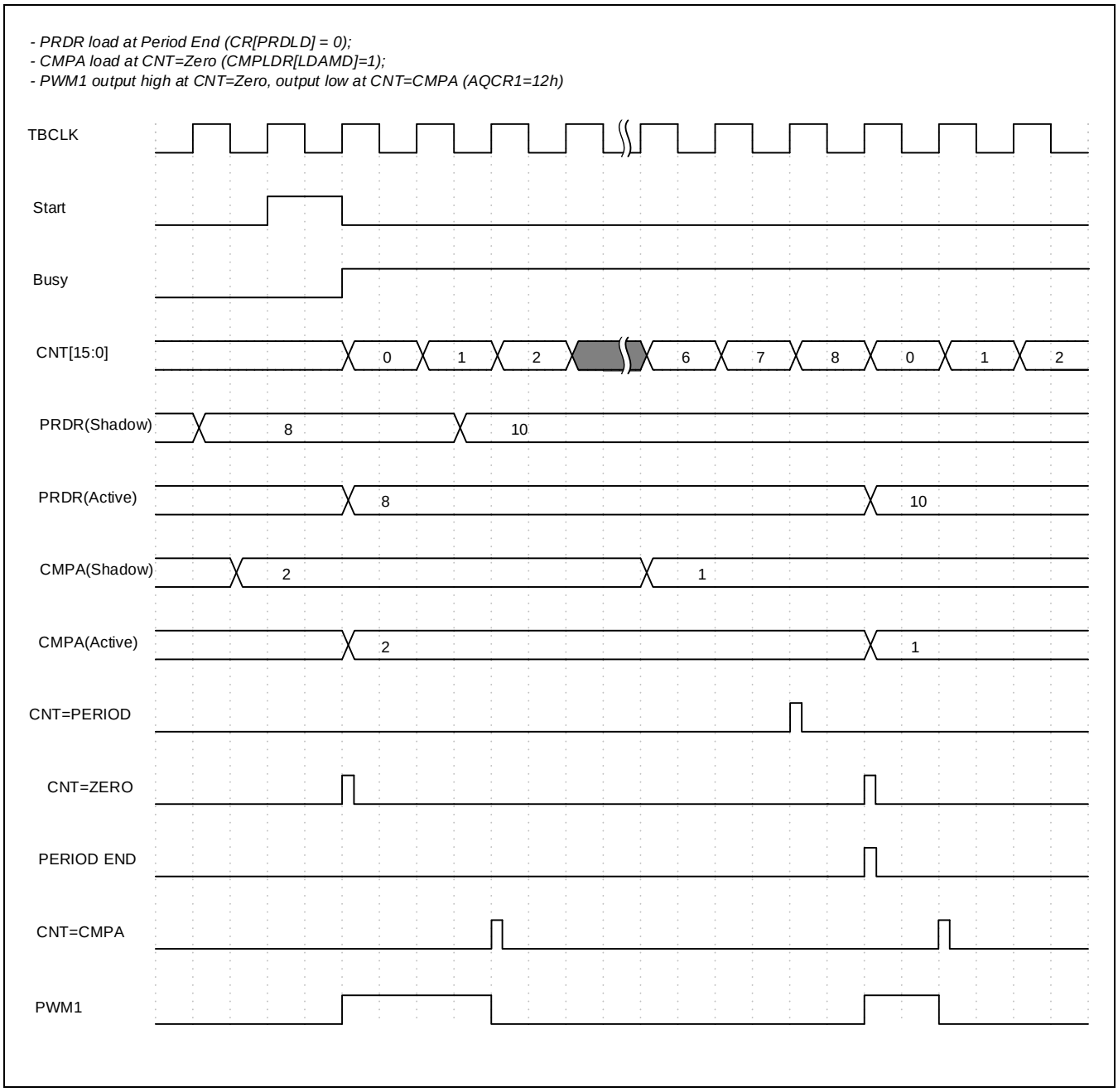


Figure 11-16 递增单比较值时，非对称波形输出

11.3.5 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。

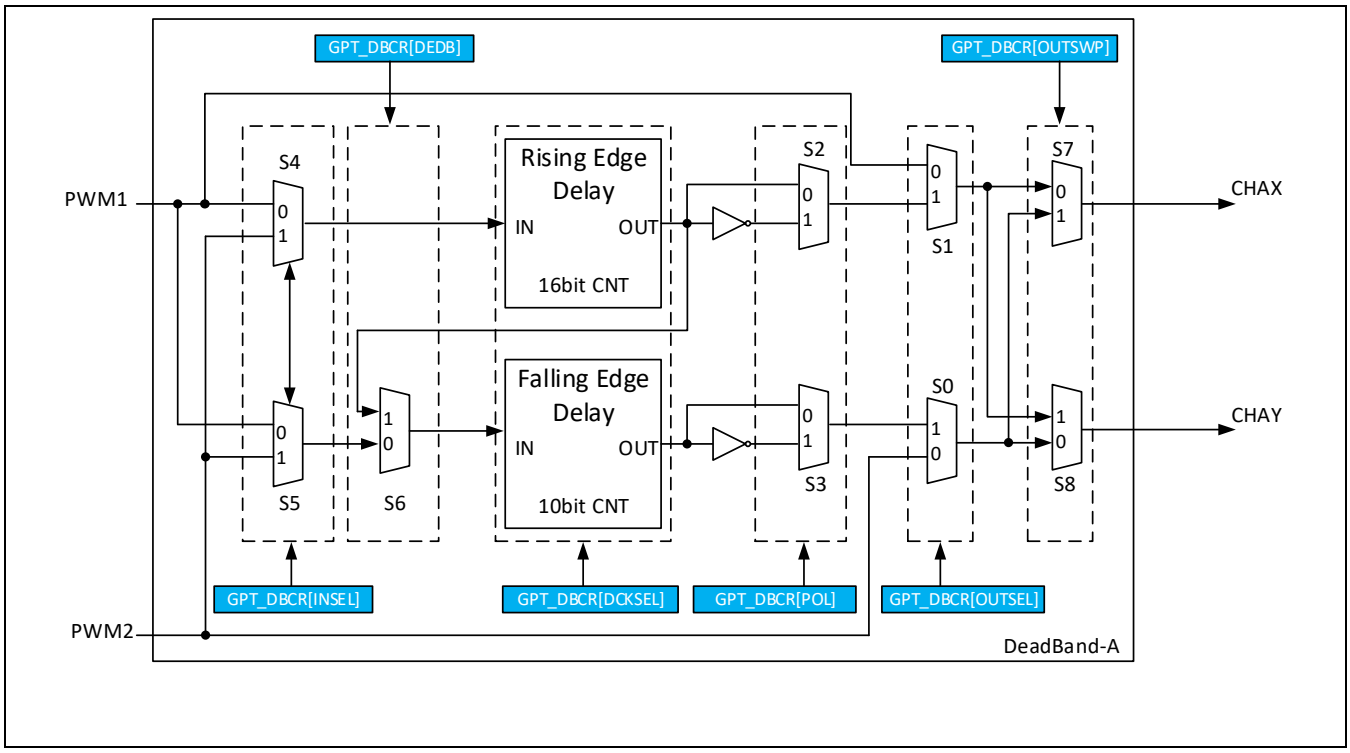


Figure 11-17 死区控制模块

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。	DBCR[CH1_INSEL]
S2，S3	延时模块输出端的极性选择。	DBCR[CH1_ POLARITY]
S0，S1	控制是否旁路延时控制模块	DBCR[CH1_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[CH1_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DBDTR[DTR]和DBDTF[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PCLK作为时钟源。当选择PCLK时钟作为时钟源时，T_{DBCLK}的时钟频率为 PCLK/(DPSC+1)。选择PCLK作为死区控制时钟，可以更加精确的控制死区宽度。

DBCR、DPSCR、DBDTR和DBDTF都具有对应的Shadow寄存器，可以通过DBLDR设置活动寄存器的加载方式，全局载入控制可以覆盖DBCR、DBDTR和DBDTF这三个寄存器的加载方式。

Table 11-4 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

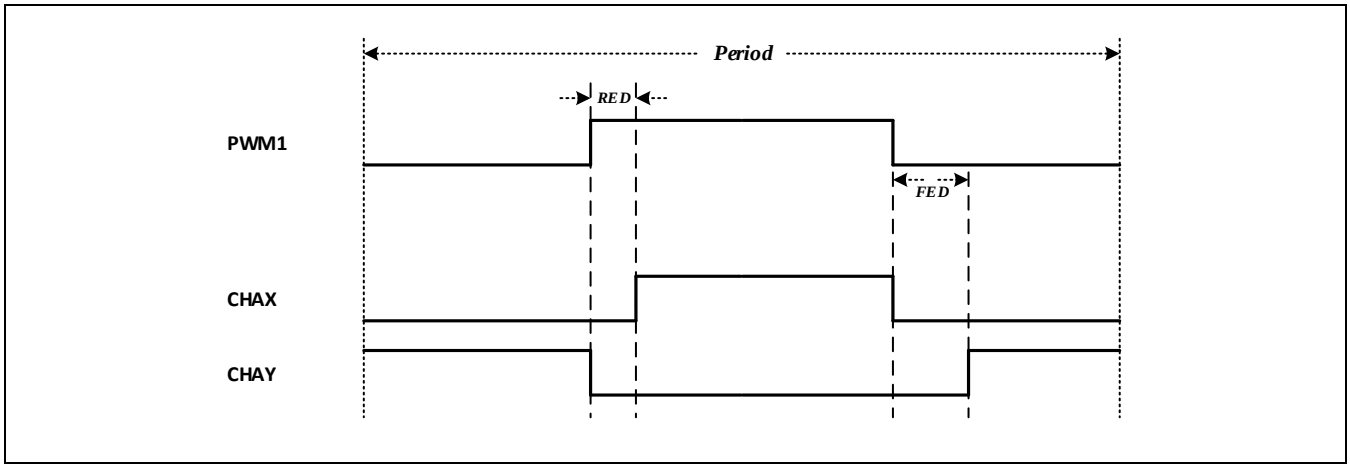


Figure 11-18 模式2：高电平有效的死区互补输出举例

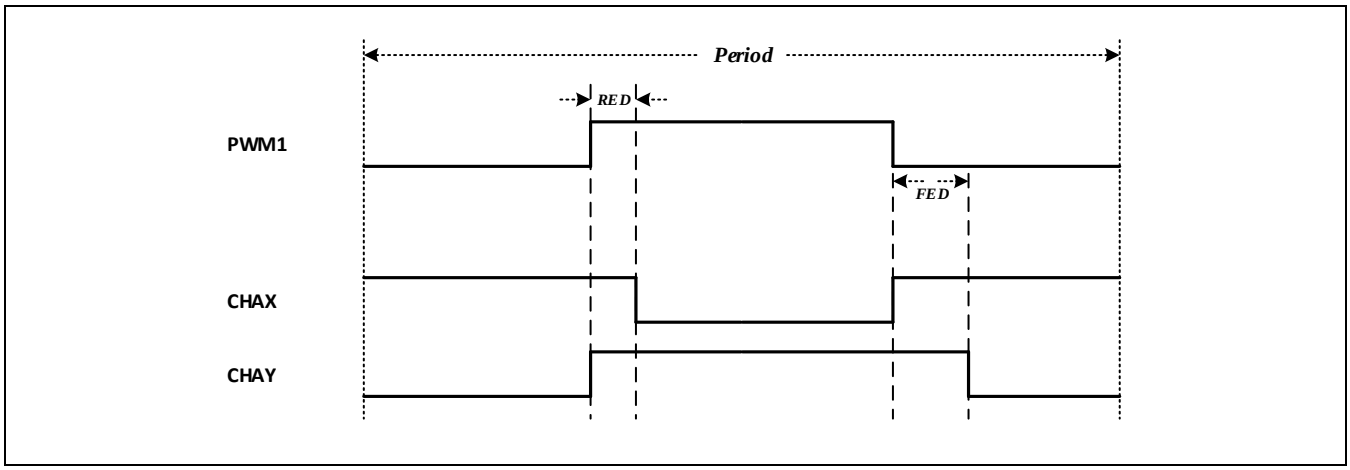


Figure 11-19 模式3：低电平有效的死区互补输出举例

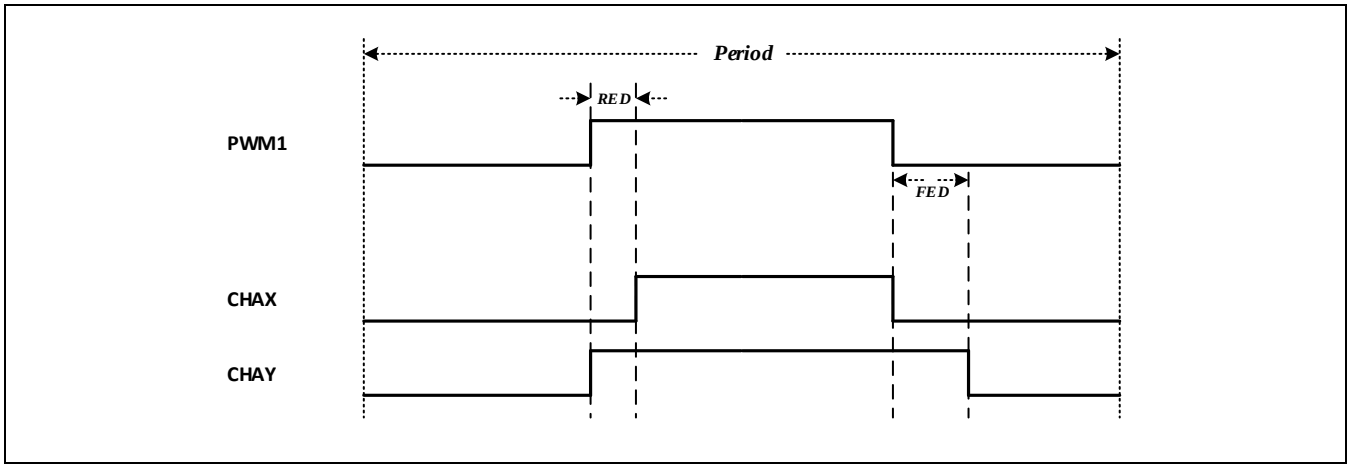


Figure 11-20 模式4：高低电平有效死区输出举例

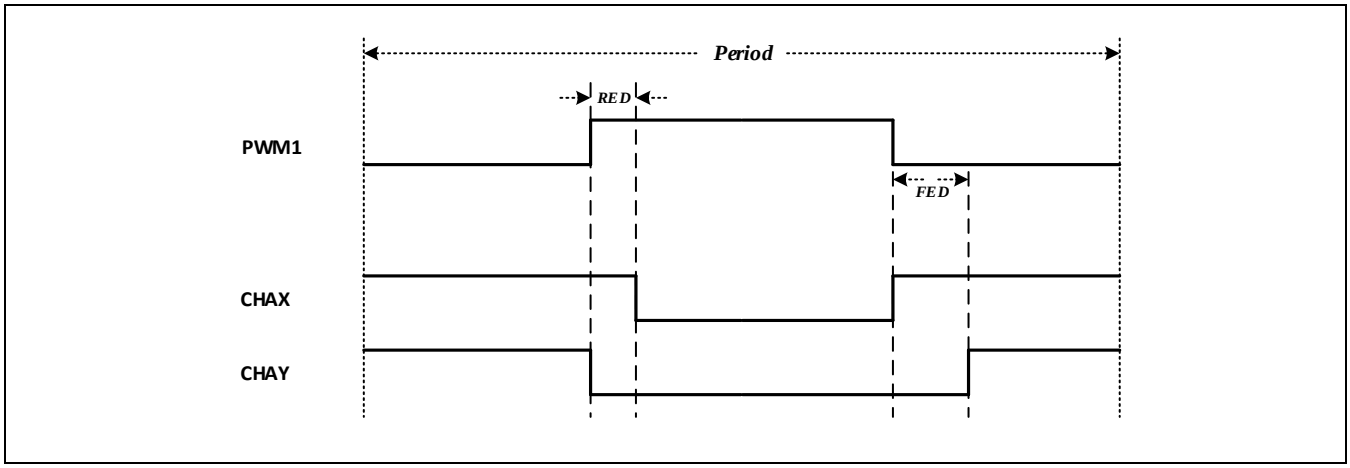


Figure 11-21 模式5：低高电平有效死区输出举例

11.3.6 紧急模式控制

11.3.6.1 紧急模式工作机制

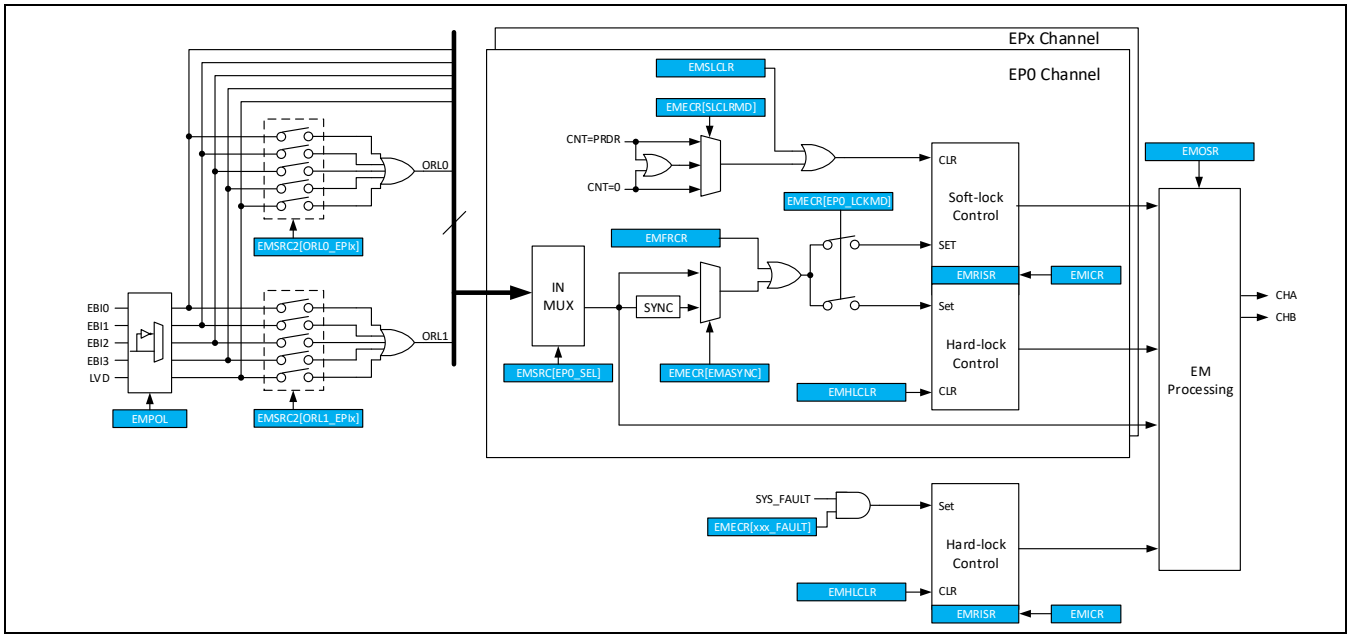


Figure 11-22紧急模式控制模块

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4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4路触发输入，每个触发输入可以独立选择触发信号源，实现PWM输出的保护联动。

-独立的系统故障触发源。

-独立的紧急模式触发中断源。

-支持软件强制触发紧急状态。

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

EBI的触发极性可以通过EMPOL寄存器进行设置，缺省为高电平有效。当EBI满足触发条件时，PWM的输出会立即做出相应变化，但对于EM的FLAG，需要进过PCLK同步后才能置位。当输入的EBI宽度小于PCLK的同步周期时(同步需要1到2个PCLK周期)，PWM的输出端口，在EBI条件不满足时，不会继续保持EM输出状态，同时EM的

FLAG也不会发生变化。通过设置EP通道的同步，可以确保只有在EM的FLAG被置位后，才会有PWM的状态改变，但这样会额外增加EM输出的响应时间。

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

- **软锁止模式 Soft-lock(SL)：**

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

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

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

- **硬锁止模式 Hard-lock(HL)：**

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

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

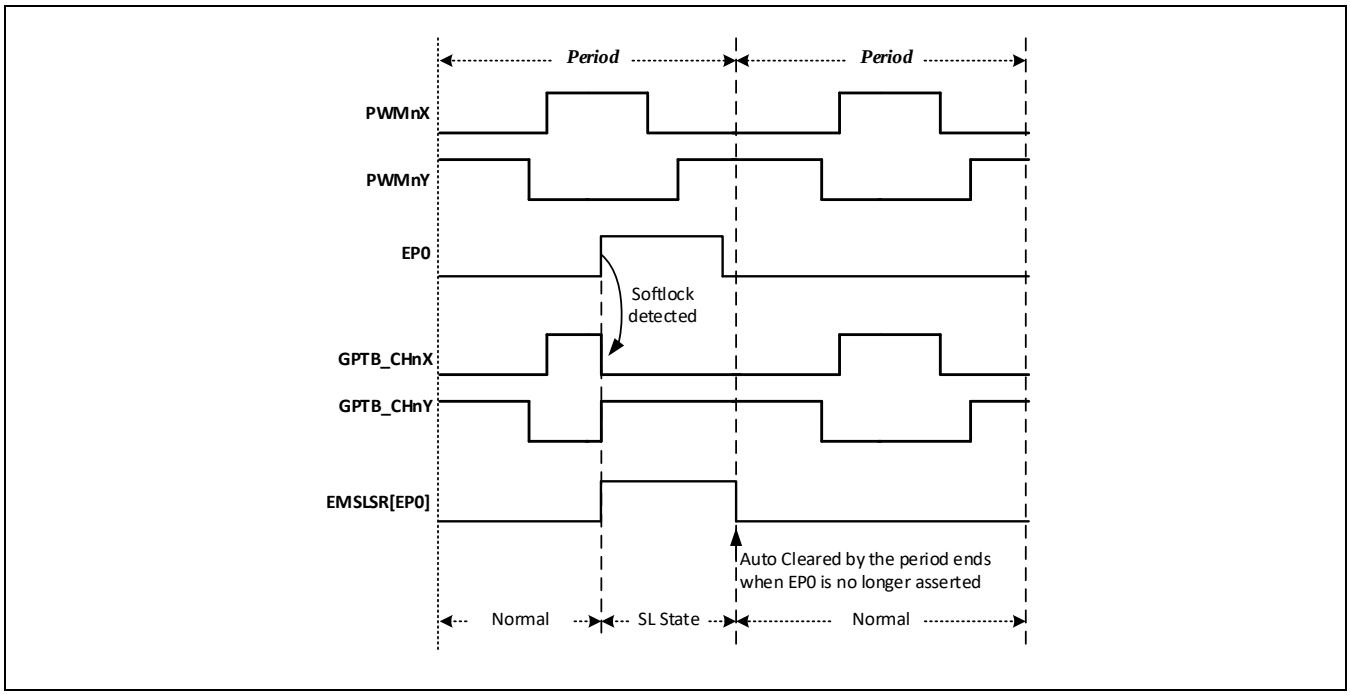


Figure 11-23 软锁止模式

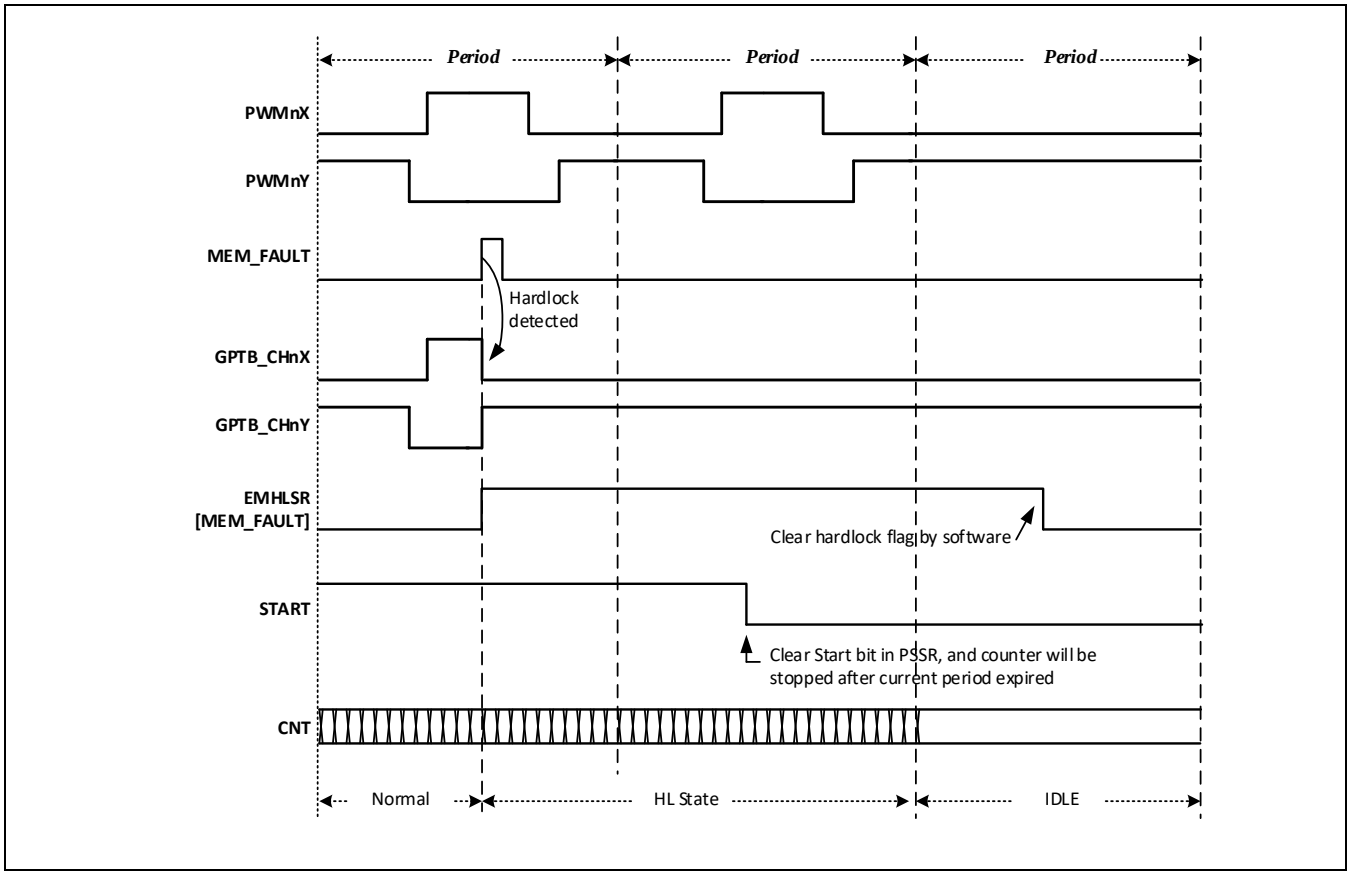


Figure 11-24 硬锁止模式

11.3.6.2 紧急模式中断

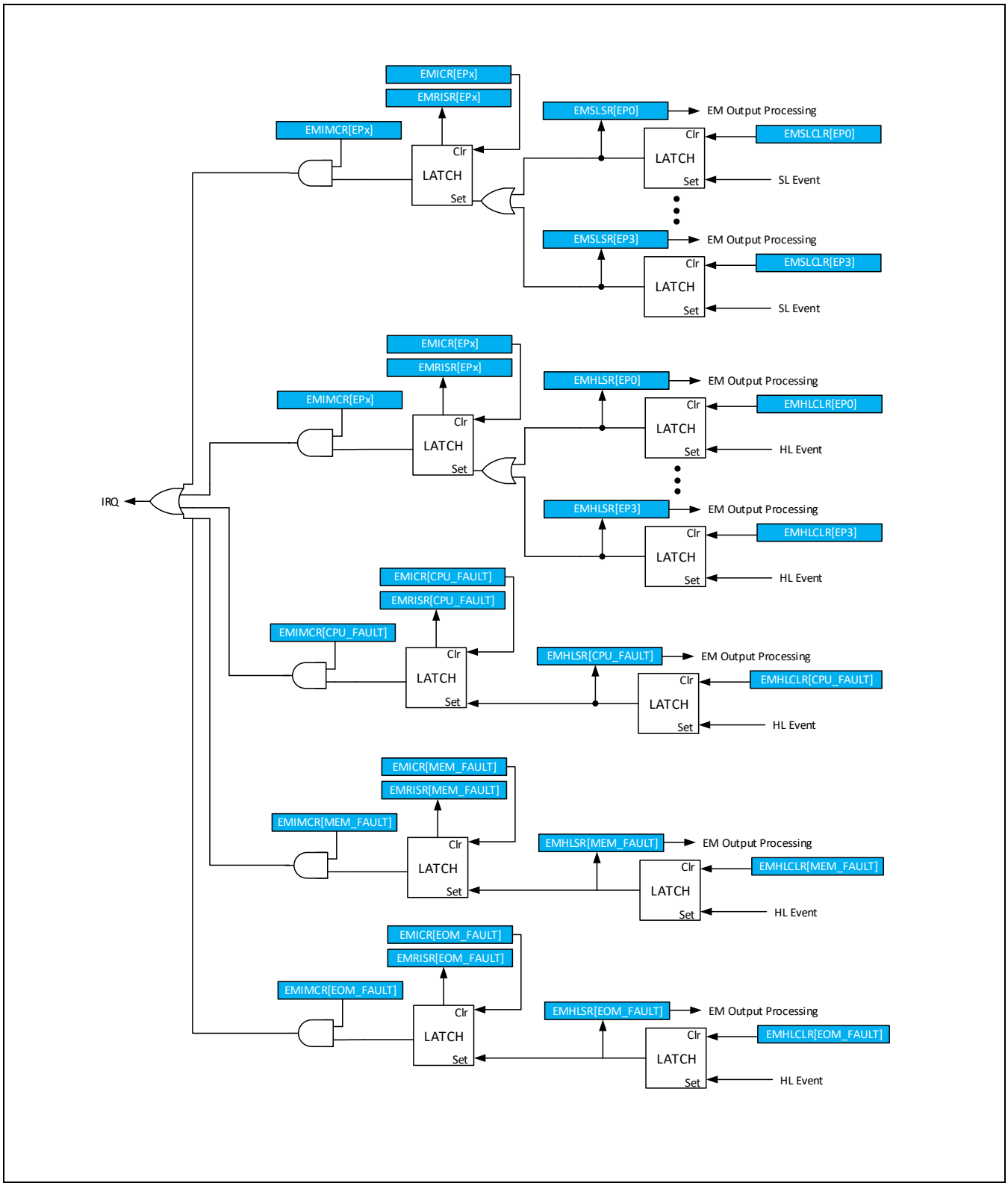


Figure 11-25 紧急模式中断

11.3.7 捕获模式

11.3.7.1 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

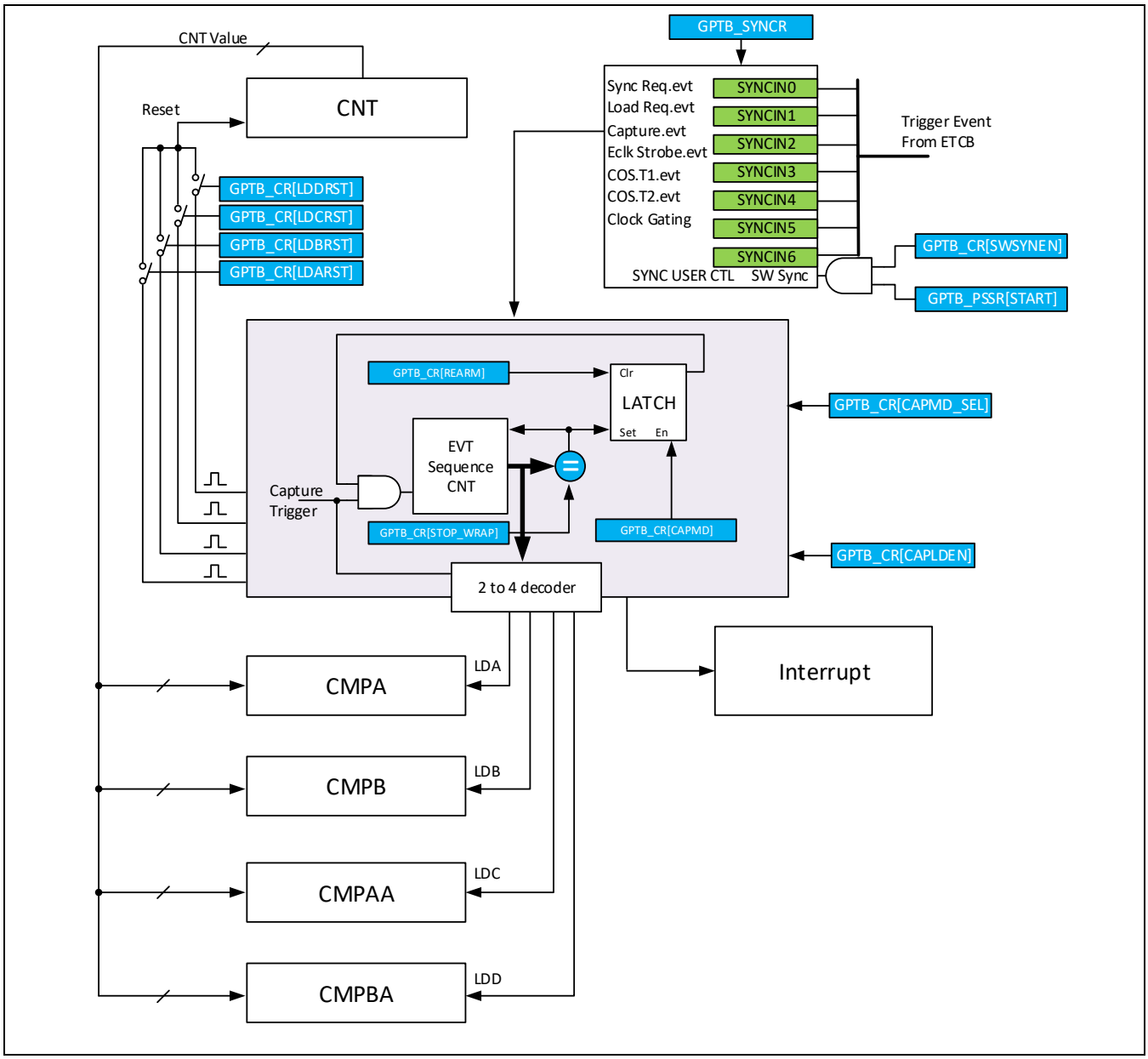


Figure 11-26 捕获模式结构框图

当GPTB_CR[WAVE]控制位设置为0时，GPTB工作在捕获模式。在捕获模式下，捕获的触发信号通过

SYNCIN2和SYNCIN3端口输入，通过GPTB_CR[CAPMD_SEL]控制位来决定是否合并SYNCIN2和SYNCIN3端口的捕获事件，捕获模块的主要功能特性如下：

- 支持4个捕获事件(GPTB_CR[CAPMD_SEL]=0)，捕获事件触发时，计数器值分别存入CMPA、CMPB、CMPAA、CMPBA寄存器中。
- 支持2个捕获事件(GPTB_CR[CAPMD_SEL]=1)，捕获事件触发时，计数器值分别存入CMPA(对应SYNCIN2触发事件)、CMPB寄存器中(对应SYNCIN3触发事件)。
- 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

11.3.7.2 捕获事件计数器

在捕获模式下，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，此时比较值寄存器将作为捕获值存储功能使用。捕获事件计数器在检测到一次捕获触发事件(SYNCIN2或SYNCIN3上的输入)时，将自动递增一次。当序列计数器值计数值超出GPTB_CR[STOP_WRAP]的设置时，计数器自动清零，并重新开始计数。

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 11-5 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 CMPA active, CAP_LD2 is triggered
3	CMPBA	CAP_LD3	Current counter value is loaded into CMPB active, CAP_LD3 is triggered

11.3.7.3 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GPTB_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GPTB_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。

11.3.7.4 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/SYNCIN3。需要通过设置ETCB，连接EPT SYNCIN2/ SYNCIN3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SYNCIN0输入和SYNCIN2输入时，此时先处理SYNCIN2事件，即存储捕获值；再处理SYNC0事件，将计数器(CNT)重置。

11.3.7.5 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。(Figure12-36)

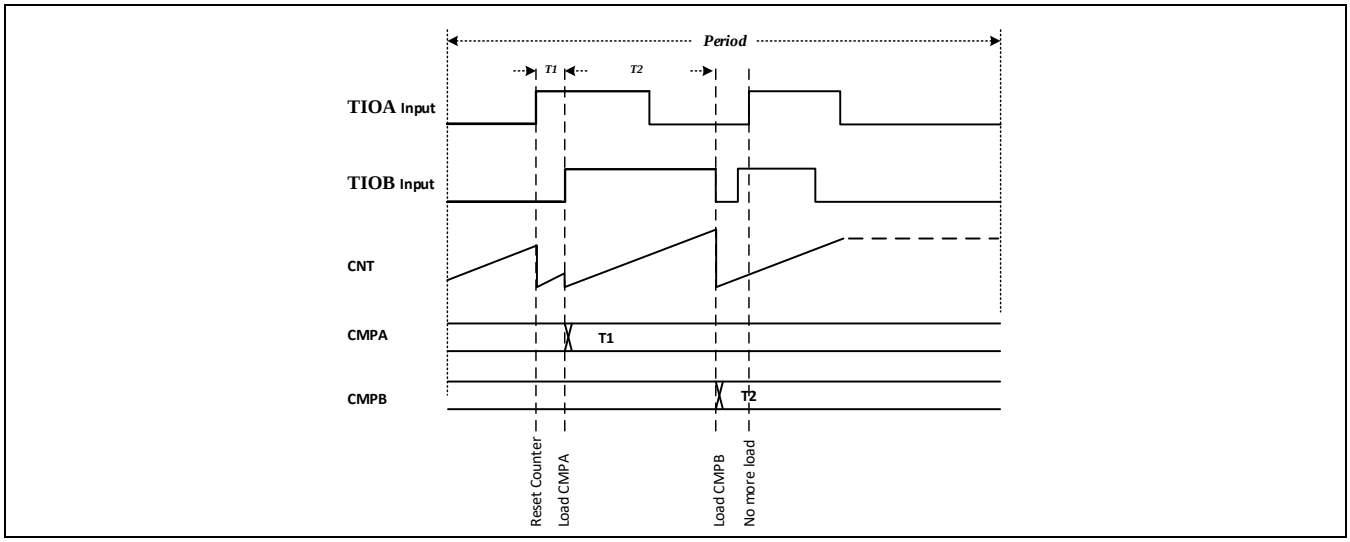


Figure 11-27 CHA的脉冲宽度检测

- **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。(Figure12-37)

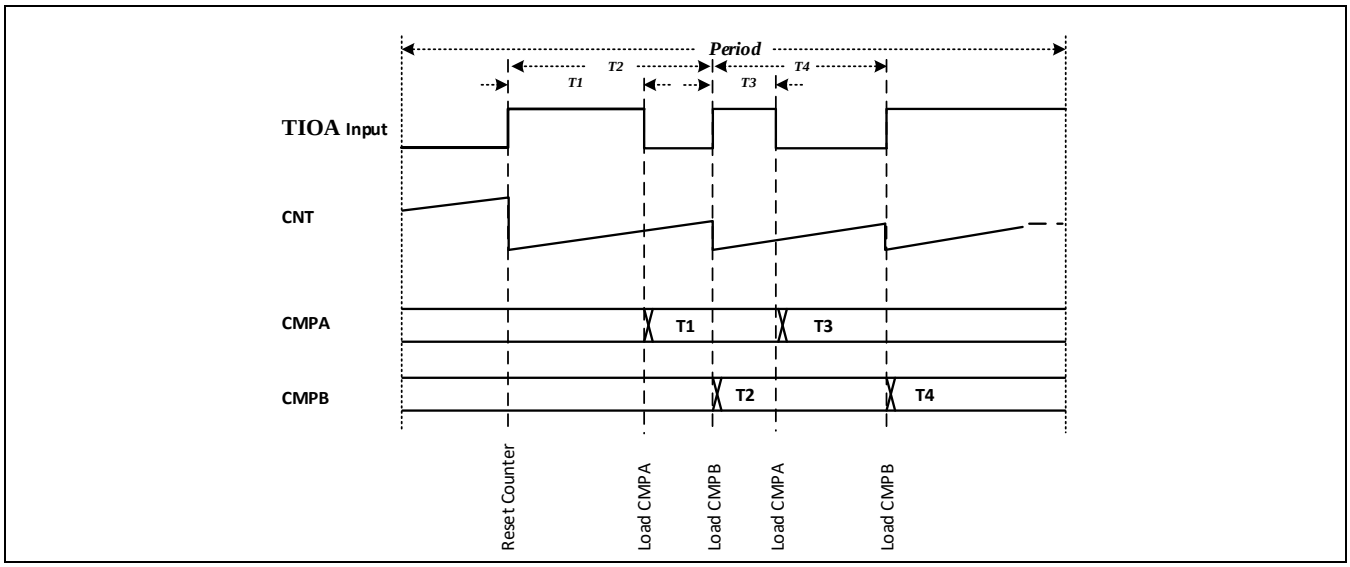


Figure 11-28 CHA的脉冲宽度和周期

11.3.8 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GPTB_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。

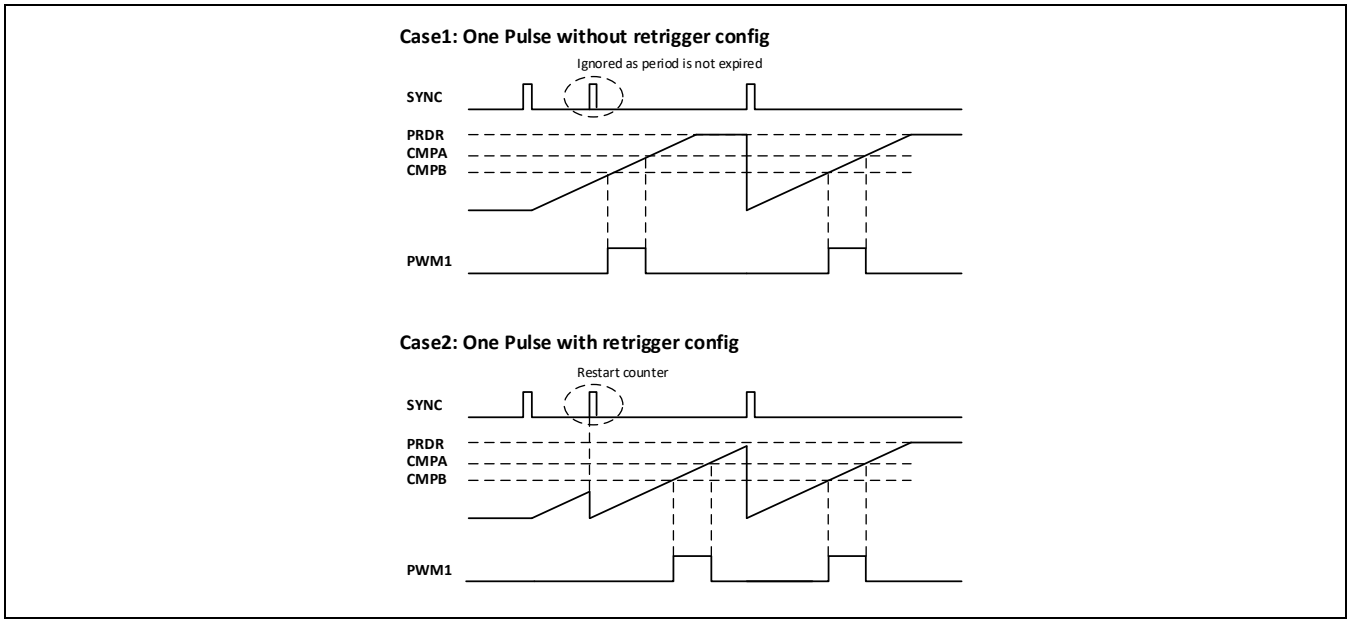


Figure 11-29 单次触发

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GPTB_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。

11.3.9 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。GPTB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。

11.3.9.1 同步触发输入接口

GPTB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
- 计数器值递增一个计数值
- 触发改变PWM的输出状态

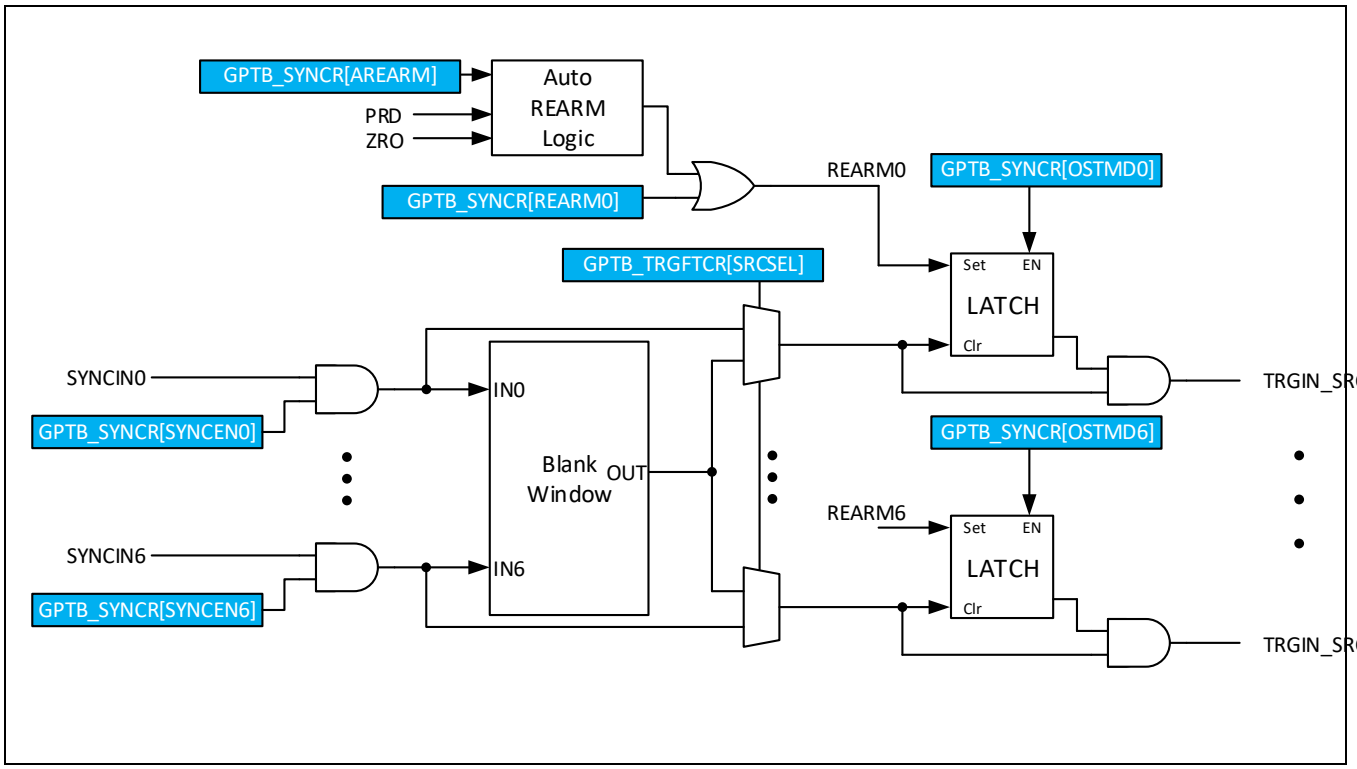


Figure 11-30 同步触发输入

每一个独立触发源被定义为SYNCIN端口，通过GPTB_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TRGIN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GPTB_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。

11.3.9.2 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。
- 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：计数器值捕获(SYNCIN2/SYNCIN3)

当该端口被触发，将触发捕获事件。只有在GPTB_CR[WAVE]设置为捕获模式时，且GPTB_CR[CAPLDEN]控制位使能时，该触发事件才能被捕获模块检测到。根据捕获模式设置为分开捕获或合并捕获，SYNCIN2和SYNCIN3的同步触发效果不一样。在合并捕获模式（GPTB_CR[CAPMD_SEL] = 0）下，来自SYNCIN2和SYNCIN3的事件都将触发一次捕获加载，按序存入四个捕获地址；在分开捕获模式下（GPTB_CR[CAPMD_SEL] = 1）下，来自SYNCIN2的事件将当前计数值加载到CMPA，来自SYNCIN3的事件将当前计数值加载到CMPB。

CNT增触发：计数器值递增一个计数值(SYNCIN4)

当该端口被触发，计数器将根据当前计数方向，自动增加一个计数值。只有在GPTB_CEDR[CSS]控制位选择TRGIN3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(SYNCIN5/ SYNCIN6)

SYNCIN5和SYNCIN6用于产生内部T1和T2触发事件。

11.3.9.3 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TRGIN端口作为事件滤波器的输入。

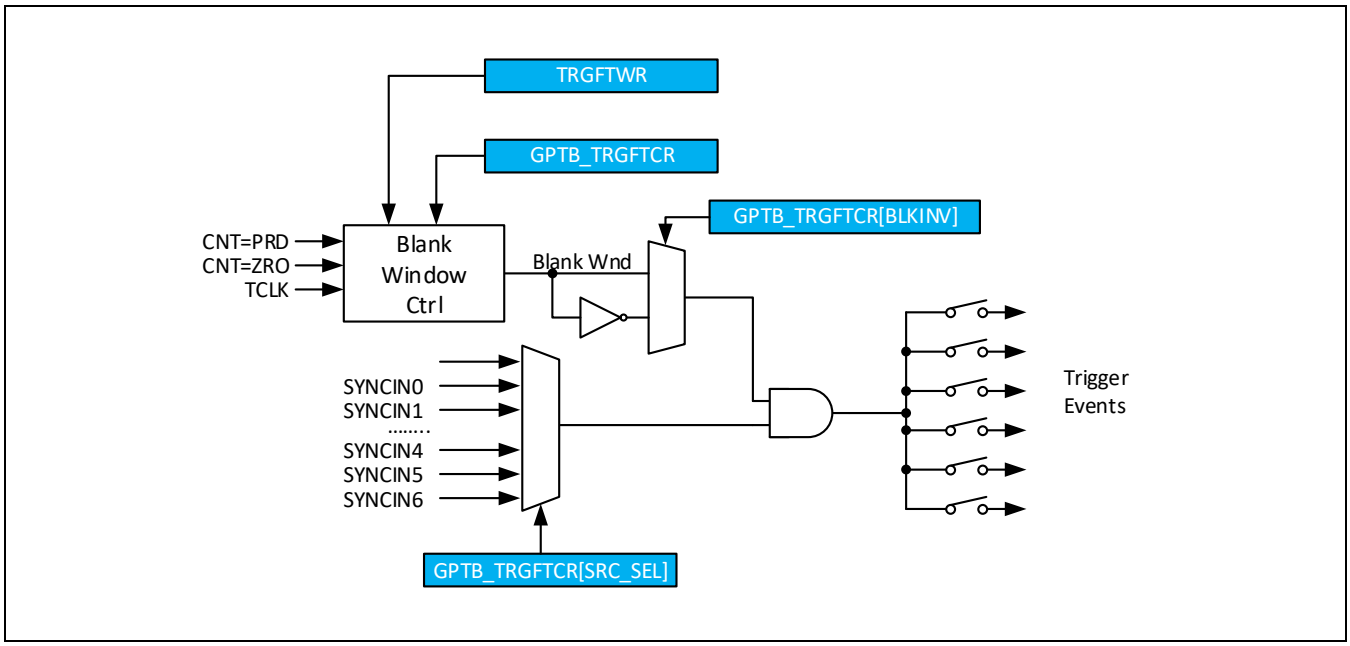


Figure 11-31 滤波器框图

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。

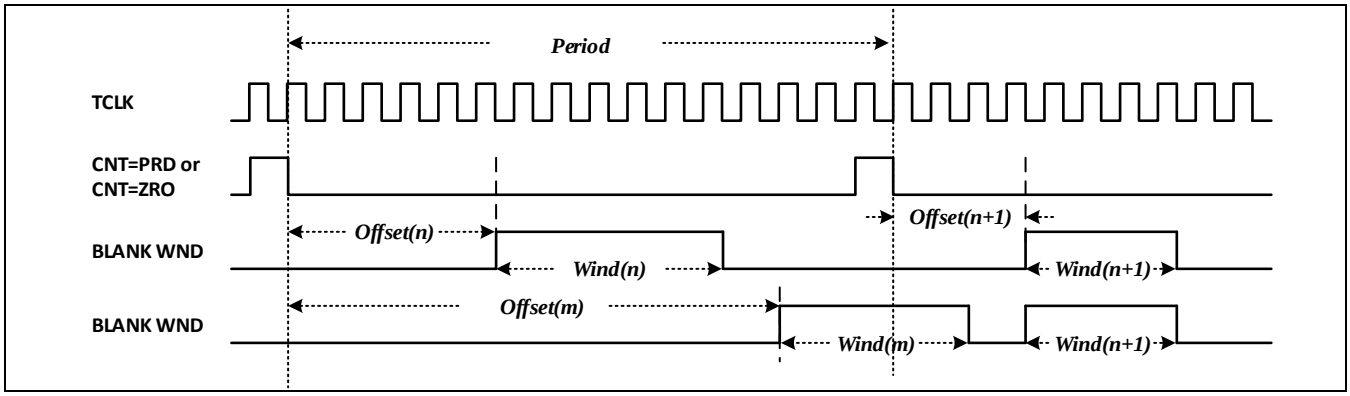


Figure 11-32 滤波器时序

11.3.10 事件触发(输出)

11.3.10.1 同步触发输出接口

GPTB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2路事件触发输出，每个事件输出对应一个GPTB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。

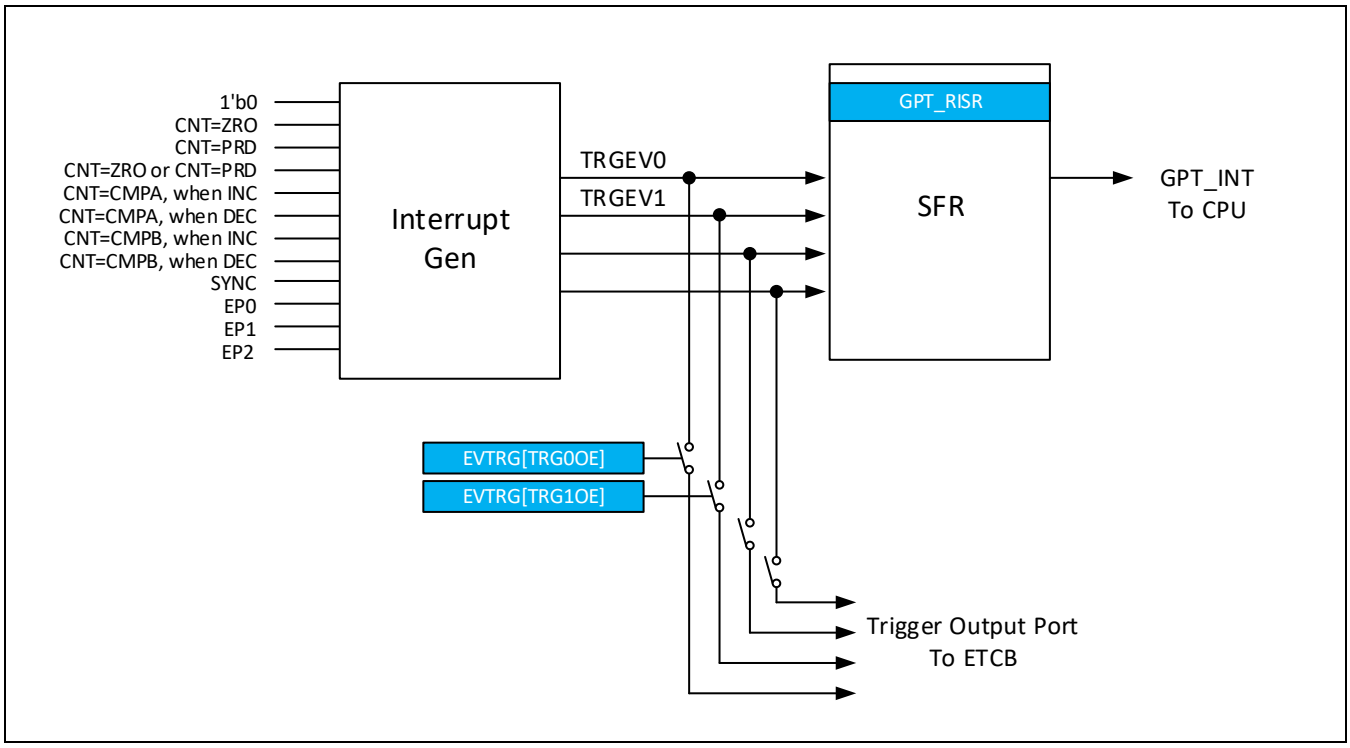


Figure 11-33 同步触发输出

11.3.10.2 事件计数和中断

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

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

中断模块一共支持2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GPTB_EVTRG寄存器进行选择。通过配置GPTB_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GPTB_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GPTB_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。

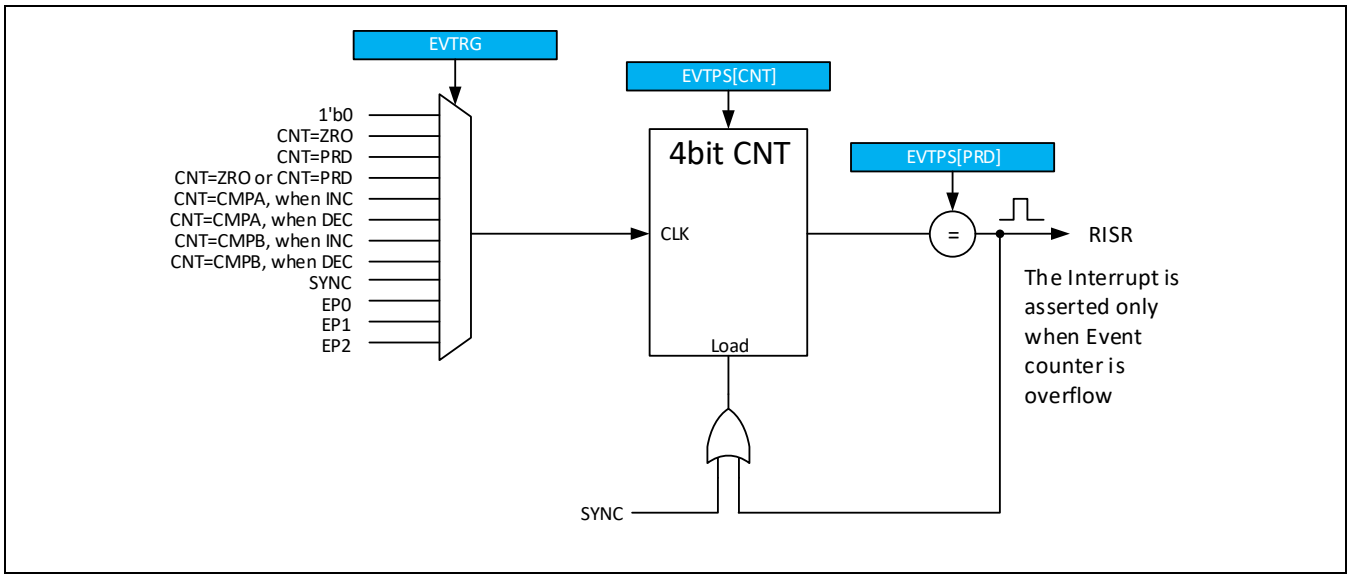


Figure 11-34 事件计数器

11.3.11 寄存器链接

增强型通用定时器A (GPTA)和增强型通用定时器B (GPTB)的PRDR、CMPA和CMPB等寄存器可以相互链接(具体请参考各章节的REGLK，REGLK2寄存器)。链接目标对应表如下表所示。

Table 11-6 寄存器链接对应表

链接值	链接目标
0x0	不链接
0x1	GPTA0
0x2	GPTB0

如果一个或多个定时器(以下称目标定时器)在REGLK或REGLK2中设置了PRDR或CMPA等寄存器(以下称为目标寄存器)的链接值，且该链接值指向同一个定时器(以下称源定时器)，当程序更新源定时器的PRDR或CMPA等寄存器(以下称为源寄存器)时,目标寄存器的PRDR或CMPA等寄存器也同时更新。例如当GPTB0.REGLK.PRDR = 0x1时，当程序更新GPTA0.PRDR时，GPTB0 PDRD会同时更新为相同值。这样多个定时器的寄存器相互链接，并通过相同事件触发，可实现多个定时器的级联。

11.4 寄存器说明

11.4.1 寄存器表

Base Address of GPTB: 0x40056000

Register	Offset	Description	Reset Value
GPTB_CEDR	0x0000	ID和时钟控制寄存器	0x6A980000
GPTB_RSSRn	0x0004	启停控制寄存器	0x00000000
GPTB_PSCR	0x0008	时钟分频控制寄存器	0x00000000
GPTB_CRn	0x000C	控制寄存器，捕捉模式 WAVE=0	0x00000000
GPTB_CRn	0x000C	控制寄存器，波形输出模式：WAVE=1	0x00000000
GPTB_SYNCRn	0x0010	同步控制寄存器	0x00000000
GPTB_GLDCR	0x0014	全局载入控制寄存器	0x00000000
GPTB_GLDCFGn	0x0018	全局载入配置	0x00000000
GPTB_GLDCR2	0x001C	全局载入控制寄存器2	0x00000001
GPTB_PRDR	0x0024	周期设置寄存器	0x00000000
GPTB_PHSR	0x0028	相位设置寄存器	0x00000000
GPTB_CMPA	0x002C	比较值A寄存器	0x00000000
GPTB_CMPB	0x0030	比较值B寄存器	0x00000000
GPTB_CMPLDR	0x003C	比较值载入控制寄存器	0x00000090
GPTB_CNT	0x0040	时基计数器寄存器	0x00000000
GPTB_AQLDR	0x0044	波形输出载入控制寄存器	0x00000024
GPTB_AQCR1	0x0048	PWM1波形输出控制寄存器	0x00000000
GPTB_AQCR2	0x004C	PWM2波形输出控制寄存器	0x00000000
GPTB_AQOSF	0x005C	一次性软件强制输出控制	0x00010000
GPTB_AQCSF	0x0060	连续软件强制输出控制	0x00000000
GPTB_DBLDR	0x0064	死区配置载入控制寄存器	0x00000492
GPTB_DBCR	0x0068	死区配置控制寄存器	0x00000000
GPTB_DPSCR	0x006C	死区延迟时钟分频控制寄存器	0x00000000
GPTB_DBDTR	0x0070	死区控制上升沿延时寄存器	0x00000000
GPTB_DBDTF	0x0074	死区控制下降沿延时寄存器	0x00000000
GPTB_EMSRCn	0x007C	紧急状态输入控制寄存器	0x00000000
GPTB_EMSRC2n	0x0080	紧急状态输入控制寄存器2	0x00000000
GPTB_EMPOLn	0x0084	紧急状态输入极性控制寄存器	0x00000000
GPTB_EMECRn	0x0088	紧急状态使能控制寄存器	0x00400000
GPTB_EMOSRn	0x008C	紧急状态输出控制寄存器	0x00000000
GPTB_EMSLSR	0x0094	紧急软锁止状态寄存器	0x00000000
GPTB_EMSLCLR	0x0098	紧急软锁止清除寄存器	0x00000000
GPTB_EMHLSR	0x009C	紧急硬锁止状态寄存器	0x00000000
GPTB_EMHLCLR	0x00A0	紧急硬锁止清除寄存器	0x00000000
GPTB_EMFRCRn	0x00A4	紧急状态软件触发寄存器	0x00000000
GPTB_EMRISR	0x00A8	紧急中断原始状态寄存器	0x00000000
GPTB_EMMISR	0x00AC	紧急中断标志寄存器	0x00000000

GPTB_EMIMCR	0x00B0	紧急中断使能控制寄存器	0x00000000
GPTB_EMICR	0x00B4	紧急中断清除寄存器	0x00000000
GPTB_EVTRG	0x00C0	事件触发选择寄存器	0x00000000
GPTB_EVSWF	0x00CC	事件计数器软件触发控制寄存器	0x00000000
GPTB_RISR	0x00D0	原始中断状态寄存器	0x00000000
GPTB_MISR	0x00D4	中断状态寄存器	0x00000000
GPTB_IMCR	0x00D8	中断使能控制寄存器	0x00000000
GPTB_ICR	0x00DC	中断清除寄存器	0x00000000
GPTB_REGLK	0x00E0	寄存器链接控制器	0x00000000
GPTB_REGLK2	0x00E4	寄存器链接控制器2	0x00000000
GPTB_PROT	0x00E8	寄存器写保护控制器	0x00000000
GPTB_CXOSF	0x00F0	一次性软件直接波形控制寄存器	0x40000000
GPTB_CXCSF	0x00F4	持续性软件直接波形控制寄存器	0x00000000
GPTB_CXMSKn	0x00F8	软件直接波形控制屏蔽寄存器	0x00000002
GPTB_CMPAA	0x82C	比较值A active寄存器	0x00000000
GPTB_CMPBA	0x830	比较值B active寄存器	0x00000000

11.4.2 GPTB_CEDR(ID和时钟控制寄存器)

Address = Base Address+ 0x0000, Reset Value = 0x6A980000

31	30	29	28	27	26	25	24	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	1	1	0	1	0	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	R	RW	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
IDCODE	[31:16]	R	当前GPTB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
CSS	[3]	RW	计数器时钟源选择位。 0h：PCLK 1h：由SYNCIN4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。

11.4.3 GPTB_RSSR(PROT)(启停控制寄存器)

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																SRR				RSVD											START
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	R	R	R	R	R	R	R	R	R	R	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。
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GPTB的启动，当GPTB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。			

11.4.4 GPTB_PSCR(时钟分频控制寄存器)

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
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)

11.4.5 GPTB_CR(WAVE=0)(控制寄存器，捕捉模式 WAVE=0)

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				LDBARST	LDAARST	LDBRST	LDARST	STOP_WRAP		CAPMD	REARM	CAPMD_SEL	WAVE	PSCLD		CGFLT			CGSRC		FLTIPSCLD	BURST	CAPLDEN	RSVD		PRDLD		RSVD	SWSYNEN	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	RW	RW	RW	RW	RW	RW	RW	W	RW	RW	RW	RW	RW	RW	RW	RW	RW	W	RW	RW	R	R	RW	RW	R	RW	R	R

Name	Bit	Type	Description
LDBARST	[27]	RW	CMPBA捕捉载入后，计数器值计数状态控制位。 0h：CMPBA触发后，计数器值不进行重置 1h：CMPBA触发后，计数器值进行重置
LDAARST	[26]	RW	CMPAA捕捉载入后，计数器值计数状态控制位。 0h：CMPAA触发后，计数器值不进行重置 1h：CMPAA触发后，计数器值进行重置
LDBRST	[25]	RW	CMPB捕捉载入后，计数器值计数状态控制位。 0h：CMPB触发后，计数器值不进行重置 1h：CMPB触发后，计数器值进行重置
LDARST	[24]	RW	CMPA捕捉载入后，计数器值计数状态控制位。 0h：CMPA触发后，计数器值不进行重置 1h：CMPA触发后，计数器值进行重置
STOP_WRAP	[23:22]	RW	Capture模式下，捕获事件计数器周期设置值。
CAPMD	[21]	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	[20]	W	重置CAPTURE 捕捉事件计数器控制。 0h：无效 1h：重置捕捉计数器 重置时，捕捉事件计数器被清零，自动打开CAPLDEN。捕捉事件计数器周期通过STOP_WRAP进行设置。
CAPMD_SEL	[19]	RW	捕捉模式选择。 0h：合并捕捉模式 注：不区分SYNCIN2/3的触发，支持4次捕获，捕获值分别存入CMPA、CMPB、CMPAA、CMPBA。 1h：分开捕捉模式 注：区分SYNCIN2/3的触发，SYNCIN2的触发，捕捉值存入CMPA，SYNCIN3的触发，捕捉值存入CMPB

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 = 4 011b：N = 6 100b：N = 8 101b：N = 16 110b：N = 32 111b：N = 64
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：CHB作为CG的输入源
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周期结束(PEND)或外部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]
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。
NOTE: 该寄存器受REGPROT保护，需要先解锁，才能写入。			

11.4.6 GPTB_CR(WAVE=1)(控制寄存器，波形输出模式：WAVE=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
RSVD													WAVE	PSCLD		CGFLT			CGSRC		FLTIPSCLD	BURST	RSVD	PHSEN	OPM	PRDLD		IDLEST	SWSYNEN	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	RW	R	R	RW	RW	RW	RW	RW	RW	RW	R	R	R	RW	RW	R	RW	R	R

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 = 4 011b：N = 6 100b：N = 8 101b：N = 16 110b：N = 32 111b：N = 64
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：CHB作为CG的输入源
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周期结束(PEND)或外部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。

11.4.7 GPTB_SYNCR(PROT)(同步控制寄存器)

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
AREARM		TRGO1SEL			TRGO0SEL			RSVD		REARM6	REARM5	REARM4	REARM3	REARM2	REARM1	REARM0	OSTMDx	OSTMD5	OSTMD4	OSTMD3	OSTMD2	OSTMD1	OSTMD0	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	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	RW	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TRGSEL1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 6h：选择SYNCIN6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TRGSEL0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 6h：选择SYNCIN6作为TRGSRC0的ExtSync触发 其他：保留
REARM6~REARM0	[21:15]	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	[14]	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。
OSTMD5~OSTMD0	[13:8]	RW	
SYNCENx	[6:0]	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，分开捕捉模式下代表CAP_LDA触发事件 SYNCIN3：Capture触发事件，分开捕捉模式下代表CAP_LDB触发事件 SYNCIN4：CNT增加一拍触发事件 + 软件直接force管脚启动事件 (1) SYNCIN5：软件直接force管脚停止事件 注意：SYNCIN4会同时触发两个功能，即CNT增加一拍的同时管脚会被软件直接force为CXOSF的设置。建议只用作定时或capture功能。

11.4.8 GPTB_GLDCR(全局载入控制寄存器)

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
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	R	R	R	RW	RW	RW	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
GLDCNT	[12:10]	R	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。

11.4.9 GPTB_GLDCFG(PROT)(全局载入配置)

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																		EMOSR	AQCSF	RSVD		AQCR2	AQCR1	DBCR	DBDTF	DBDTR	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	RW	RW	R	R	RW	RW	RW	RW	RW	R	R	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
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
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

11.4.10 GPTB_GLDCR2(全局载入控制寄存器2)

Address = Base Address+ 0x001C, 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 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。			

11.4.11 GPTB_PRDR(周期设置寄存器)

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
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GPT_CR[PRDLD]可以选择Shadow到Active载入的触发条件。
NOTE: 该寄存器受REGPROT保护，需要先解锁，才能写入。			

11.4.12 GPTB_PHSR(相位设置寄存器)

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
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			

11.4.13 GPTB_CMPA(比较值A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
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。

11.4.14 GPTB_CMPB(比较值B寄存器)

Address = Base Address+ 0x0030, 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。

11.4.15 GPTB_CMPLDR(比较值载入控制寄存器)

Address = Base Address+ 0x003C, 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	RW	RW	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]

11.4.16 GPTB_CNT(时基计数器寄存器)

Address = Base Address+ 0x0040, 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。

11.4.17 GPTB_AQLDR(波形输出载入控制寄存器)

Address = Base Address+ 0x0044, 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

11.4.18 GPTB_AQCR1(PWM1波形输出控制寄存器)

Address = Base Address+ 0x0048, 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		RSVD										C2U		RSVD		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	R	R	R	R	R	R	R	R	R	R	RW	RW	R	R	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
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
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）
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。 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。 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）

11.4.19 GPTB_AQCR2(PWM2波形输出控制寄存器)

Address = Base Address+ 0x004C, 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		RSVD										C2U		RSVD		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	R	R	R	R	R	R	R	R	R	R	RW	RW	R	R	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
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
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）
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）

11.4.20 GPTB_AQOSF(一次性软件强制输出控制)

Address = Base Address+ 0x005C, 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，通道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	在通道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，直到有其他改变通道2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	在通道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，直到有其他改变通道1输出状态的触发事件发生。

11.4.21 GPTB_AQCSF(连续软件强制输出控制)

Address = Base Address+ 0x0060, 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	通过软件对通道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	通过软件对通道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

11.4.22 GPTB_DBLDR(死区配置载入控制寄存器)

Address = Base Address+ 0x0064, 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																				SHDWLDPSCMD		LDPSCMD		SHDWLDDTFMD		LDDTFMD		SHDWLDDTRMD		LDDTRMD		SHDWLDCRMD		LDCRMD	
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
SHDWLDPSCMD	[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
LDPSCMD	[9]	RW	DCKPSC寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
SHDWLDDTFMD	[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
LDDTFMD	[6]	RW	DBDTF寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
SHDWLDDTRMD	[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
LDDTRMD	[3]	RW	DBDTR寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
SHDWLDCRMD	[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

LDCRMD	[0]	RW	DBCR寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式

11.4.23 GPTB_DBCR(死区配置控制寄存器)

Address = Base Address+ 0x0068, 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						CH1_EDEB	DCKSEL	RSVD																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	R	R	RW	RW	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
CH1_EDEB	[25]	RW	在PWM1 DBCOUTY 上选择死区双沿模式（S6开关） 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
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

11.4.24 GPTB_DPSCR(死区延迟时钟分频控制寄存器)

Address = Base Address+ 0x006C, 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)

11.4.25 GPTB_DBDTR(死区控制上升沿延时寄存器)

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																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

11.4.26 GPTB_DBDTF(死区控制下降沿延时寄存器)

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																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

11.4.27 GPTB_EMSRC(PROT)(紧急状态输入控制寄存器)

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																								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
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
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。

11.4.28 GPTB_EMSRC2(PROT)(紧急状态输入控制寄存器2)

Address = Base Address+ 0x0080, 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_EBI3	ORL1_EBI2	ORL1_EBI1	ORL1_EBI0	RSVD				FLT_PACE0				RSVD				ORL1_EBI3	ORL1_EBI2	ORL0_EBI1	ORL0_EBI0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	R	R	R	R	RW	RW	RW	RW	R	R	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
ORL1_EBI3~ORL1_EBI0	[19:16]	RW	多路通道的逻辑OR作为EPx中的输入信号(ORL1)。 0h：屏蔽当前通道作为OR输入 1h：使能当前通道作为OR输入
FLT_PACE0	[11:8]	RW	EP0、EP1的数字去抖滤波检查周期数。 0000：禁止滤波 0001：2个周期 0010：4个周期 0011：6个周期 0100：8个周期 0101：16个周期 0110：32个周期 0111：64个周期
ORL0_EBI3~ORL0_EBI0	[3:0]	RW	多路通道的逻辑OR作为EPx中的输入信号(ORL0)。 0h：屏蔽当前通道作为OR输入 1h：使能当前通道作为OR输入

11.4.29 GPTB_EMPOL(PROT)(紧急状态输入极性控制寄存器)

Address = Base Address+ 0x0084, 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																												EBI3_POL	EBI2_POL	EBI1_POL	EBI0_POL
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
EBI3_POL	[3]	RW	
EBI2_POL	[2]	RW	
EBI1_POL	[1]	RW	
EBI0_POL	[0]	RW	

紧急状态的输入有效极性选择控制。

0h：高电平有效

1h：低电平有效

当EBIx作为异常处理输入时，以电平方式工作。当EBIx作为事件触发源时，设置为高电压有效时，即上升沿触发；设置为低电平有效时，即下降沿触发。

NOTE: 该寄存器受REGPROT保护，需要先解锁，才能写入。

11.4.30 GPTB_EMECR(PROT)(紧急状态使能控制寄存器)

Address = Base Address+ 0x0088, 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		SHDWLDOSRMD		LDOSRMD	RSVD																	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	R	R	R	R	R	R	R	R	R	R	R	R	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
SHDWLDOSRMD	[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
LDOSRMD	[21]	RW	EMOSR寄存器的Shadow功能使能控制。 0h：Immediate模式 1h：Shadow模式
EP1_LCKMD	[3:2]	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
EP0_LCKMD	[1:0]	RW	

11.4.31 GPTB_EMOSR(PROT)(紧急状态输出控制寄存器)

Address = Base Address+ 0x008C, 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_COAY		RSVD				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_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_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。			

11.4.32 GPTB_EMSLSR(紧急软锁止状态寄存器)

Address = Base Address+ 0x0094, 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																														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
EP1~EP0	[1:0]	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

11.4.33 GPTB_EMSLCLR(紧急软锁止清除寄存器)

Address = Base Address+ 0x0098, 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																														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	W	W

Name	Bit	Type	Description
EP1~EP0	[1:0]	W	软件清除EPx触发的软锁止状态标志。 0h：对当前控制位写‘0’无效，读取时总返回‘0’ 1h：清除当前标志位

11.4.34 GPTB_EMHLSR(紧急硬锁止状态寄存器)

Address = Base Address+ 0x009C, 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																											MEM_FAULT	EOM_FAULT	CPU_FAULT	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
MEM_FAULT	[4]	R	MEM FAULT事件触发的硬锁止状态位。 0h：硬锁止未触发 1h：硬锁止已触发
EOM_FAULT	[3]	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
CPU_FAULT	[2]	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
EP1~EP0	[1:0]	R	EPx触发的硬锁止状态标志。 0h：硬锁止未触发 1h：硬锁止已触发

11.4.35 GPTB_EMHLCLR(紧急硬锁止清除寄存器)

Address = Base Address+ 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
RSVD																											MEM_FAULT	EOM_FAULT	CPU_FAULT	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	W	W	W	W	W

Name	Bit	Type	Description
MEM_FAULT	[4]	W	软件清除MEM FAULT事件触发的硬锁止状态位。 0h：对当前控制位写‘0’无效，读取时总返回‘0’ 1h：清除当前标志位
EOM_FAULT	[3]	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
CPU_FAULT	[2]	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
EP1~EP0	[1:0]	W	软件清除EPx触发的硬锁止状态标志。 0h：对当前控制位写‘0’无效，读取时总返回‘0’ 1h：清除当前标志位

11.4.36 GPTB_EMFRCR(PROT)(紧急状态软件触发寄存器)

Address = Base Address+ 0x00A4, 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_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	R	R	R	R	R	R	W	W

Name	Bit	Type	Description
FRC_EP1~FRC_EP0	[1:0]	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。			

11.4.37 GPTB_EMRISR(紧急中断原始状态寄存器)

Address = Base Address+ 0x00A8, 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																					MEM_FAULT	EOM_FAULT	CPU_FAULT	RSVD						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
MEM_FAULT	[10]	R	MEM_FAULT事件触发的异常事件中断原始标志位。 0h：无中断请求发生 1h：中断请求发生
EOM_FAULT	[9]	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
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
EP1~EP0	[1:0]	R	EPx事件触发的异常事件中断原始标志位。 0h：无中断请求发生 1h：中断请求发生

11.4.38 GPTB_EMMISR(紧急中断标志寄存器)

Address = Base Address+ 0x00AC, 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																					MEM_FAULT	EOM_FAULT	CPU_FAULT	RSVD						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
MEM_FAULT	[10]	R	MEM_FAULT事件触发的异常事件中断标志位。 0h：无中断请求发生 1h：中断请求发生
EOM_FAULT	[9]	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
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
EP1~EP0	[1:0]	R	EPx事件触发的异常事件中断标志位。 0h：无中断请求发生 1h：中断请求发生

11.4.39 GPTB_EMIMCR(紧急中断使能控制寄存器)

Address = Base Address+ 0x00B0, 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																					MEM_FAULT	EOM_FAULT	CPU_FAULT	RSVD						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	R	R	R	R	R	RW	R	RW

Name	Bit	Type	Description
MEM_FAULT	[10]	RW	MEM FAULT事件触发的异常中断使能控制。 0h：禁止对CPU发起中断请求 1h：允许对CPU发起中断请求
EOM_FAULT	[9]	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
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
EP1~EP0	[1:0]	R	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

11.4.40 GPTB_EMICR(紧急中断清除寄存器)

Address = Base Address+ 0x00B4, 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																					MEM_FAULT	EOM_FAULT	CPU_FAULT	RSVD						EP1	EP0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	R	R	R	R	R	R	W	W

Name	Bit	Type	Description
MEM_FAULT	[10]	W	软件清除MEM FAULT事件触发的中断标志。 0h：对当前控制位写‘0’无效，读取时总返回‘0’ 1h：清除当前标志位
EOM_FAULT	[9]	W	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
CPU_FAULT	[8]	W	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
EP1~EP0	[1:0]	W	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

11.4.41 GPTB_EVTRG(事件触发选择寄存器)

Address = Base Address+ 0x00C0, 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				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	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
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
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 1100：ExtSync通道 1101：EP0 event 1110：EP1 event 1111：EP2 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 1100：ExtSync通道 1101：EP0 event 1110：EP1 event

			1111：EP2 event

11.4.42 GPTB_EVSWF(事件计数器软件触发控制寄存器)

Address = Base Address+ 0x00CC, 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	W	W

Name	Bit	Type	Description
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

11.4.43 GPTB_RISR(原始中断状态寄存器)

Address = Base Address+ 0x00D0, 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													ZERO	PRD	PEND	RSVD					CBU	RSVD	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
ZERO	[18]	R	ZRO事件中断请求原始标志状态
PRD	[17]	R	PRD事件中断请求原始标志状态
PEND	[16]	R	周期结束中断请求原始标志状态 up-counting:CNT=PRDR发生PEND事件 down-counting:CNT=ZRO发生PEND事件 up-down-counting:CNT=1发生PEND事件(递减到1)
CBU	[10]	R	递增阶段CNT = CMPB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			

11.4.44 GPTB_MISR(中断状态寄存器)

Address = Base Address+ 0x00D4, 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													ZERO	PRD	PEND	RSVD					CBU	RSVD	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
ZERO	[18]	R	ZRO事件中断请求标志状态
PRD	[17]	R	PRD事件中断请求标志状态
PEND	[16]	R	周期结束中断请求标志状态 up-counting:CNT=PRDR发生PEND事件 down-counting:CNT=ZRO发生PEND事件 up-down-counting:CNT=1发生PEND事件(递减到1)
CBU	[10]	R	递增阶段CNT = CMPB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
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			

11.4.45 GPTB_IMCR(中断使能控制寄存器)

Address = Base Address+ 0x00D8, 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													ZERO	PRD	PEND	RSVD					CBU	RSVD	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	RW	RW	RW	R	R	R	R	R	RW	R	RW	RW	RW	RW	RW	R	R	RW	RW

Name	Bit	Type	Description
ZERO	[18]	RW	ZRO事件中断使能控制位
PRD	[17]	RW	PRD事件中断使能控制位
PEND	[16]	RW	周期结束中断中断使能控制位 up-counting:CNT=PRDR发生PEND事件 down-counting:CNT=ZRO发生PEND事件 up-down-counting:CNT=1发生PEND事件(递减到1)
CBU	[10]	RW	递增阶段CNT = CMPB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 CMPBA中断使能控制位。
CAP_LD2	[6]	RW	Capture Load to CMPAA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。
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。			

11.4.46 GPTB_ICR(中断清除寄存器)

Address = Base Address+ 0x00DC, 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													ZERO	PRD	PEND	RSVD					CBU	RSVD	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	W	W	W	R	R	R	R	R	W	R	W	W	W	W	W	R	R	W	W

Name	Bit	Type	Description
ZERO	[18]	W	ZRO事件中断清除
PRD	[17]	W	PRD事件中断清除
PEND	[16]	W	周期结束中断清除 up-counting:CNT=PRDR发生PEND事件 down-counting:CNT=ZRO发生PEND事件 up-down-counting:CNT=1发生PEND事件(递减到1)
CBU	[10]	W	递增阶段CNT = CMPB请中断清除
CAU	[8]	W	递增阶段CNT = CMPA中断清除
CAP_LD3	[7]	W	Capture Load to CMPBA中断清除
CAP_LD2	[6]	W	Capture Load to CMPAA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
TRGEV1	[1]	W	TRGEV1中断清除
TRGEV0	[0]	W	TRGEV0中断清除

11.4.47 GPTB_REGLK(寄存器链接控制器)

Address = Base Address+ 0x00E0, 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				RSSR				GLD2				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	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
RSSR	[27:24]	RW	RSSR寄存器链接目标。 请参考寄存器链接描述章节
GLD2	[23:20]	RW	GLDCR2寄存器链接目标。 请参考寄存器链接描述章节
CMPB	[11:8]	RW	CMPB寄存器链接目标。 请参考寄存器链接描述章节
CMPA	[7:4]	RW	CMPA寄存器链接目标。 请参考寄存器链接描述章节
PRDR	[3:0]	RW	PRDR寄存器链接目标。 请参考寄存器链接描述章节
连接到相应的定时器。 0h:不链接 1h:GPTA0 2h:GPTB0			

11.4.48 GPTB_REGLK2(寄存器链接控制器2)

Address = Base Address+ 0x00E4, 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				AQOSF				EMFRCR				EMICR				EMHLCLR				EMSLCLR			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
AQCSF	[23:20]	RW	AQCSF寄存器链接目标。 请参考寄存器链接描述章节
AQOSF	[19:16]	RW	AQOSF寄存器链接目标。 请参考寄存器链接描述章节
EMFRCR	[15:12]	RW	EMFRCR寄存器链接目标。 请参考寄存器链接描述章节
EMICR	[11:8]	RW	EMICR寄存器链接目标。 请参考寄存器链接描述章节
EMHLCLR	[7:4]	RW	EMHLCLR寄存器链接目标。 请参考寄存器链接描述章节
EMSLCLR	[3:0]	RW	EMSLCLR寄存器链接目标。 请参考寄存器链接描述章节
连接到相应的定时器。 0h:不链接 1h:GPTA0 2h:GPTB0			

11.4.49 GPTB_PROT(寄存器写保护控制器)

Address = Base Address+ 0x00E8, 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。

11.4.50 GPTB_CXOSF(一次性软件直接波形控制寄存器)

Address = Base Address+ 0x00F0, Reset Value = 0x40000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RLDCSF		RSVD											OSTSFB	OSTSFAY	OSTSFAX	RSVD													ACTB	ACTAY	ACTAX
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	R	R	R	R	R	R	R	R	R	R	R	WO	WO	WO	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW

Name	Bit	Type	Description
RLDCSF	[31:30]	RW	CXCSF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
OSTSFB	[18]	WO	在通道B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AX输出状态的触发事件发生。
OSTSFAY	[17]	WO	在通道AY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AX输出状态的触发事件发生。
OSTSFAX	[16]	WO	在通道AX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AX输出状态的触发事件发生。
ACTB	[2]	RW	通过软件对BX做直接强制赋值。 0h：强制输出低 1h：强制输出高
ACTAY	[1]	RW	通过软件对AY做直接强制赋值。 0h：强制输出低 1h：强制输出高
ACTAX	[0]	RW	通过软件对AX做直接强制赋值。 0h：强制输出低 1h：强制输出高

11.4.51 GPTB_CXCSF(持续性软件直接波形控制寄存器)

Address = Base Address+ 0x00F4, 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																													CSFB	CSFAY	CSFAX
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	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
CSFB	[2]	RW	通过软件对B做连续直接强制赋值。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CXOSF寄存器中的RLDCSF控制位进行配置。 0h：禁止强制赋值 1h：使能强制赋值
CSFAY	[1]	RW	通过软件对AY做连续直接强制赋值。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CXOSF寄存器中的RLDCSF控制位进行配置。 0h：禁止强制赋值 1h：使能强制赋值
CSFAX	[0]	RW	通过软件对AX做连续直接强制赋值。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CXOSF寄存器中的RLDCSF控制位进行配置。 0h：禁止强制赋值 1h：使能强制赋值

11.4.52 GPTB_CXMSK(PROT)(软件直接波形控制屏蔽寄存器)

Address = Base Address+ 0x00F8, Reset Value = 0x00000002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																													MSKB	MSKA	
0	0	0	0	0	0	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
R	R	R	R	R	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
MSKB	[2]	RW	若软件直接强制赋值的B值和MSKB相同，则软件直接强制赋值无效。
MSKA	[1:0]	RW	若软件直接强制赋值的AY,AX值和MSKA相同，则软件直接强制赋值无效。

11.4.53 GPTB_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	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

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。

11.4.54 GPTB_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	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

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。

12 窗口型看门狗 (WWDT)

12.1 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。

12.1.1 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

12.2 功能描述

12.2.1 模块框图

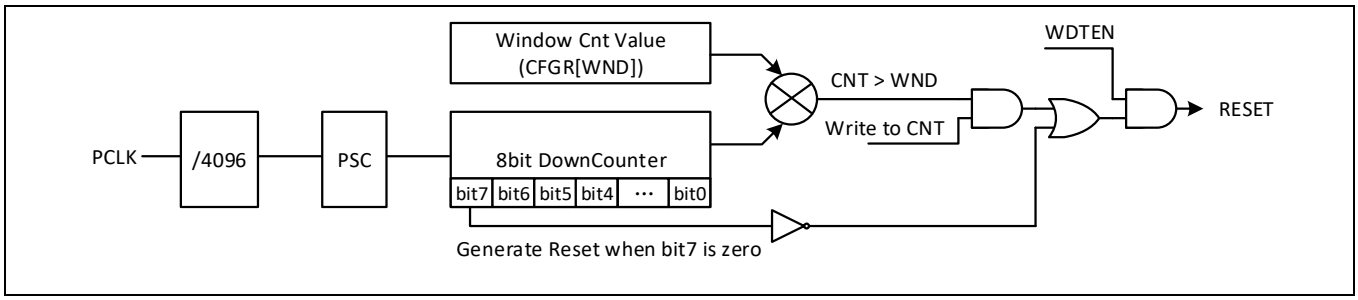


Figure 12-1 模块结构示意图

12.2.2 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

12.2.3 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：

$$T_{WWDT} = T_{PCLK} \times 4096 \times 2^{PSC} \times (CNT[6:0]+1)$$

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 12-1 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。

12.2.4 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。

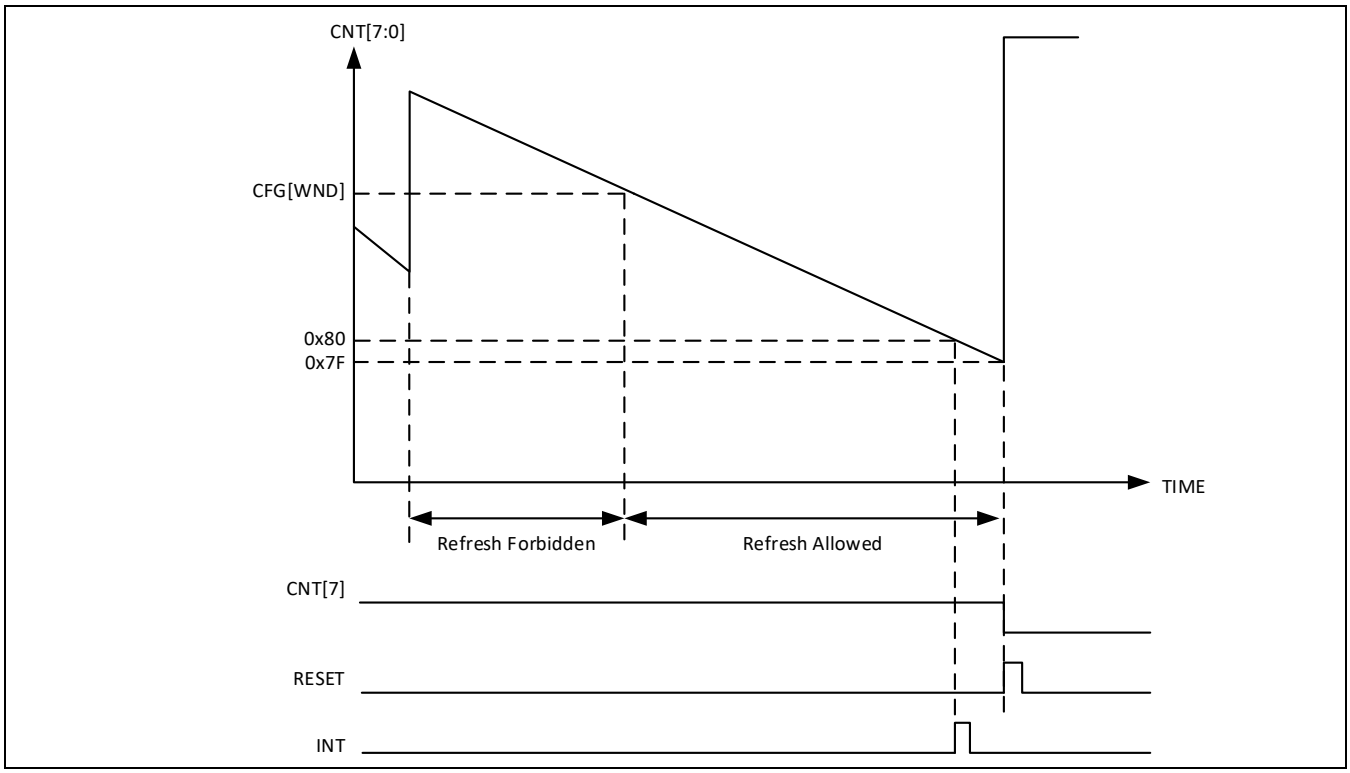


Figure 12-2 计数器工作示例

12.2.5 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

12.3 寄存器说明

12.3.1 寄存器表

Base Address of WWDT: 0x40062000

Register	Offset	Description	Reset Value
WWDT_CR	0x0000	控制寄存器	0x000000FF
WWDT_CFGR	0x0004	配置寄存器	0x000000FF
WWDT_RISR	0x0008	原始中断状态寄存器	0x00000000
WWDT_MISR	0x000C	中断状态寄存器	0x00000000
WWDT_IMCR	0x0010	中断使能控制寄存器	0x00000000
WWDT_ICR	0x0014	中断清除寄存器	0x00000000

12.3.2 WWDT_CR(控制寄存器)

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。 读： 0h：看门狗禁止状态 1h：看门狗使能状态 写： 0h：无效 1h：使能看门狗 注意：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。
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’			

12.3.3 WWDT_CFGR(配置寄存器)

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。

12.3.4 WWDT_RISR(原始中断状态寄存器)

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
															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

12.3.5 WWDT_MISR(中断状态寄存器)

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																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

12.3.6 WWDT_IMCR(中断使能控制寄存器)

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																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			

12.3.7 WWDT_ICR(中断清除寄存器)

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
															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’			

13 I2C总线

13.1 概述

I2C总线是一个由数据(SDA)和时钟(SCL)组成的两线同步串行接口。每个接在总线上器件都可以被一个唯一的地址寻址。SDA和SCL为双向接口，通过一个上拉电阻接到正向电源。接到总线上的器件输出必须设置成开漏模式以实现”线与”的功能。

I2C总线是一个真正的多主机总线，因为它包含了冲突检测和仲裁，在多主机同时启动数据传输时可以避免数据丢失。时钟的同步通过I2C接口和SCL之间线与的方式实现。

I2C接口可以工作在快速模式和标准模式。快速模式支持的波特率范围为0到400Kbit/s，标准模式支持的波特率范围为0到100Kbit/s。该模块支持4种模式：主机发送，主机接收，从机发送，从机接收；支持7位寻址和10位寻址，并且支持检测本机地址功能和General Call寻址功能(从机模式)。

13.1.1 主要特性

- 多主机总线
- 串行，8位的双向数据传输
- 支持标准模式的100Kbit/s，快速模式最高支持400Kbit/s

13.1.2 管脚描述

Table 13-1 I2C 管脚描述

Pin Name	Function	I/O Type	Active Level	Comments
SDA	串行数据线	I/O	高有效	–
SCL	串行时钟线	I/O	高有效	–

13.2 功能描述

13.2.1 模块框图

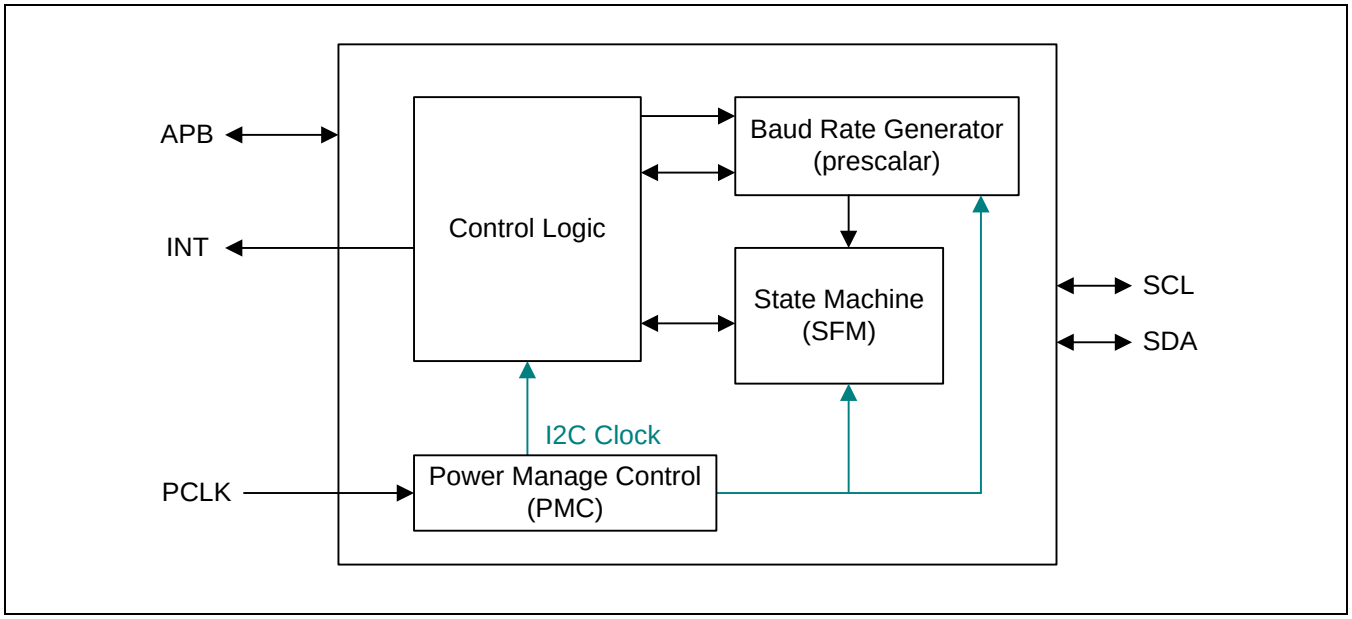


Figure 13-1 I2C模块框图

13.2.2 功能介绍

13.2.2.1 I2C总线概念

13.2.2.1.1 协议概念

串行数据 SDA 和串行时钟 SCL 这两根线能够在连接到它们的器件之间传输数据。每个器件都有一个唯一的地址，不管该器件是单片机，LCD 驱动芯片，存储芯片还是键盘接口，根据所需功能的不同，都可以作为一个发送端或者一个接收端。显然 LCD 驱动只能作为接收端，而存储芯片既能接收也能发送数据。除了作为发送端和接收端，在进行数据传输时，I2C 的器件也可以被称作主机或者从机。主机是一个可以在总线上发起数据传输，并且产生时钟信号来完成该传输的器件，在这个时候，任何被寻址到的器件都被当作是一个从机。

I2C总线是一个支持多主机的总线，意思是可以连接很多具有控制总线功能的器件。由于主机通常都是单片机，让我们以I2C总线上的两个单片机为例。要注意这些关系并不是永久性的，因为这个关系跟数据传输的方向有关。数据的传输过程如下：

1: 假设单片机A希望给单片机B发送信息

- 单片机A(主机)寻址单片机B(从机)
- 单片机A(主机-发送端)把数据发给单片机B(从机)
- 单片机A结束该传输

2: 如果单片机A希望从单片机B接收数据

- 单片机A(主机)寻址单片机B(从机)
- 单片机A(主机-接收端)从单片机B(从机-发送端)接收数据
- 单片机A结束该传输

即使在这种情况下(上面情况2)，数据的传输也是由主机(单片机A)来产生时钟并且结束。

能够把多个单片机接到I2C总线上的意思就是总线支持多个主机同时发起数据传输。为了避免混乱，I2C支持总线仲裁机制，这个机制依赖于I2C总线上所有I2C接口的线与连接。

如果2个或者多个主机尝试发起数据传输，那么第一个成功产生“1”的主机将获得发送权而其它为成功产生“1“的主机则失去发送权。仲裁过程中的时钟信号是由线与连接到SCL的主机时钟信号经过同步逻辑产生的。

时钟信号总是由主机来负责产生和发送；每个主机在传输数据的时候，都是主机自己来产生时钟信号。只有当慢速从机拉低时钟线的时候，或者当仲裁发生时其它主机拉低了时钟线的时候，主机产生的时钟信号才会被改变。

13.2.2.1.2 一般特性

SDA和SCL都是双向传输线，通过一个上拉电阻接到正向的电源电压。当总线空闲时，两个信号都是高电平状态。连接到总线上器件的输出都必须设置成开漏输出以支持线与的功能。I2C总线的数据传输在标准模式下可以到100Kbit/s，而在快速模式下则高达400Kbit/s。接在总线上每个接口的寄生电容不能超过400pF。

下表列出了一些寄存器设置对应的波特率。波特率的快慢跟I2C时钟，快速模式和I2C_MR寄存器里的PRV位有关。

Table 13-2 波特率设置示例

I2C Clock	PRV	Baud Rate	FAST	% Error
20	204	96000	0	–0.16%
	156	125000	1	0.00%
	100	192000	1	–0.16%
	48	384000	1	–0.16%
18	184	96000	0	0.27%
	140	125000	1	0.00%
	90	192000	1	0.27%
	43	384000	1	0.27%
37.5	387	96000	0	0.10%
	296	125000	1	0.00%
	191	192000	1	–0.16%
	94	384000	1	0.35%
18.75	191	96000	0	–0.16%
	146	125000	1	0.00%
	94	192000	1	0.35%
	45	384000	1	0.35%
10	100	96000	0	–0.16%
	76	125000	1	0.00%
	48	192000	1	–0.16%
	22	384000	1	–0.16%
9.375	94	96000	0	0.35%
	71	125000	1	0.00%
	45	192000	1	0.35%
4.6875	45	96000	0	0.35%

13.2.2.2 位传输

由于各种不同工艺的器件(CMOS, NMOS, bipolar)都能连接在I2C总线上，所以逻辑0和1的电平是不确定的，跟VDD的电平有关。每个时钟脉冲传输1位数据。

13.2.2.2.1 数据有效性

SDA传输线的数据必须要在时钟信号为高的期间保持不变。数据线的高低状态转换必须发生在SCL为低电平的期间。

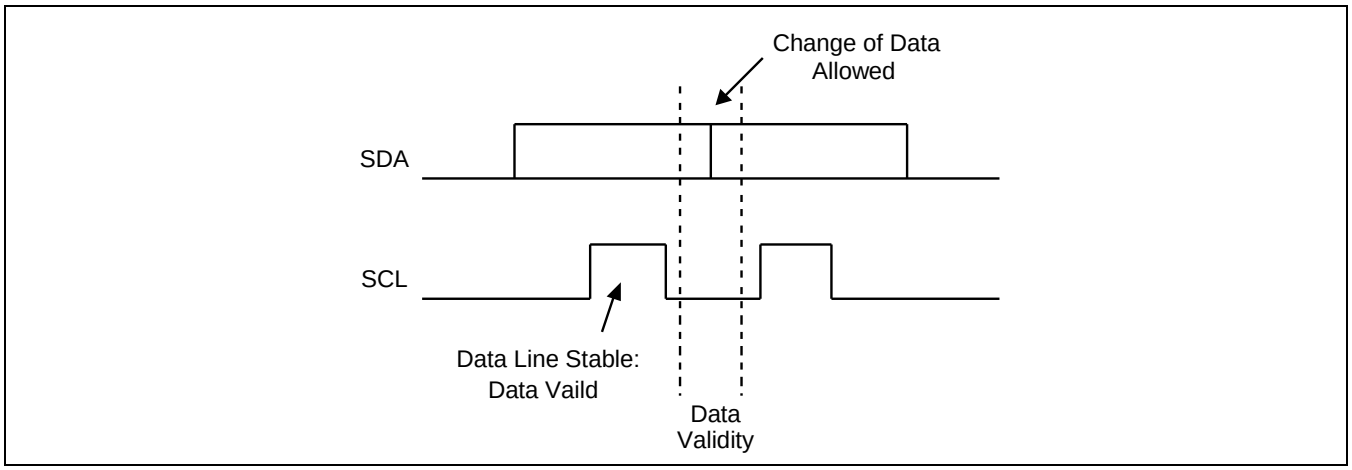


Figure 13-2 数据有效性

13.2.2.2.2 起始位和停止位

在I2C总线的传输过程中，一些特殊的情况被定义成起始位和停止位。

当SCL是高的时候，SDA从高变低，被定义为起始位。

当SCL是高的时候，SDA从低变高，被定义为停止位。

起始位和停止位都是由主机产生的。在起始位产生以后，总线被认为是处于工作状态(BUSY)，直到停止位产生后总线则被认为是处于空闲状态。

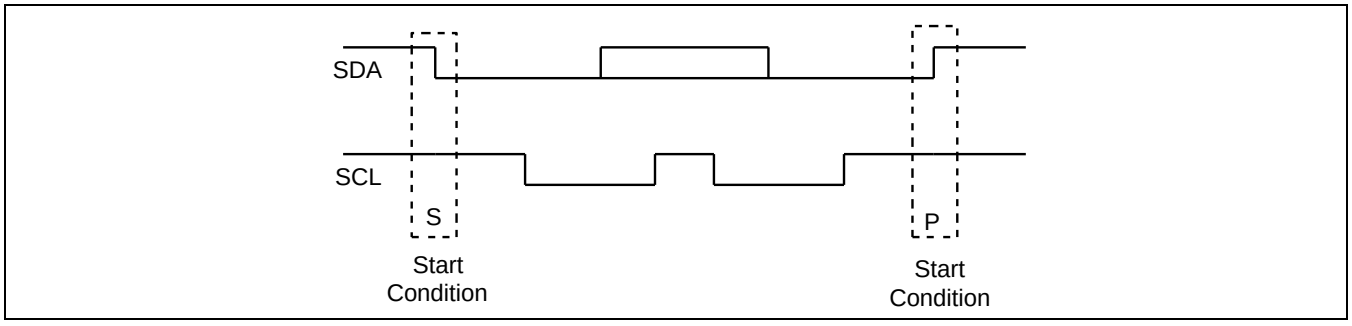


Figure 13-3 起始位和停止位

13.2.2.3 数据传输

13.2.2.3.1 传输字节格式

SDA上传输的每个字节的长度为8位。每次传输字节的总个数没有限定，也就是说理论上可以传输无限个字节的数据。每个字节传输完后，紧接着会有一个应答位。数据的最高位先发送(MSB优先)。如果接收端在它完成某个其它任务前无法接收时，例如在处理中断服务程序时，接收端可以拉低SCL信号线强制让发送端进入等待状态。当接收端准备好后则释放SCL信号线，之后数据传输继续。

某些特殊情况下，允许使用与I2C总线不同的数据格式(例如兼容CBUS的器件)。这种特殊情况下的数据传输即使在一个字节的传输当中，也可以由停止位来终止，不需要应答位。

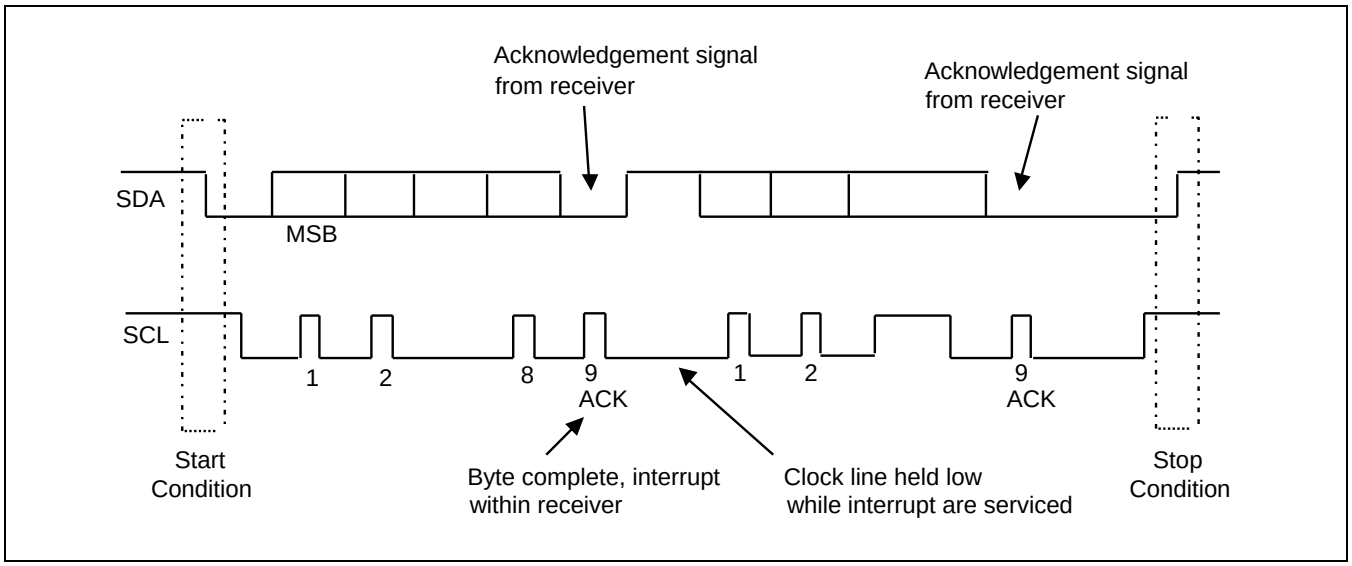


Figure 13-4 I2C总线的数据传输

13.2.2.3.2 应答

带应答机制的数据传输在I2C协议中是必须的。应答信号需要的时钟脉冲是由主机来产生的。发送端在应答时钟脉冲宽度内，释放SDA信号线(高电平)的控制权，也就是不输出。

接收端必须在应答时钟脉冲期间内拉低SDA信号线，并且在这个时钟为高的期间一直保持低。当然，注意setup和hold时间也必须计算入内。

通常接收端在收到每个字节后都必须发送一个应答信号，除非该传输是CBUS的地址。

当从机-接收端无法应答从机地址时(比如正在处理一些实时任务)，从机必须将数据线拉高。这时主机可以产生一个停止位，终止该传输。

如果从机-接收端应答了从机地址，但是在一段时间后的传输中无法再接收更多的数据了，这时主机必须再次终止传输。也就是说，从机在第一个字节传输后的应答位上发送一个“非应答”，在应答时钟脉冲周期内让数据线保持高电平，这样主机就会产生一个停止位。

主机-接收端在数据传输时，通过不发送最后一个字节的应答信号，告诉从机-发送端该传输已经结束。从机-发送端则必须释放数据线，让主机来产生停止位或者重复开始位。

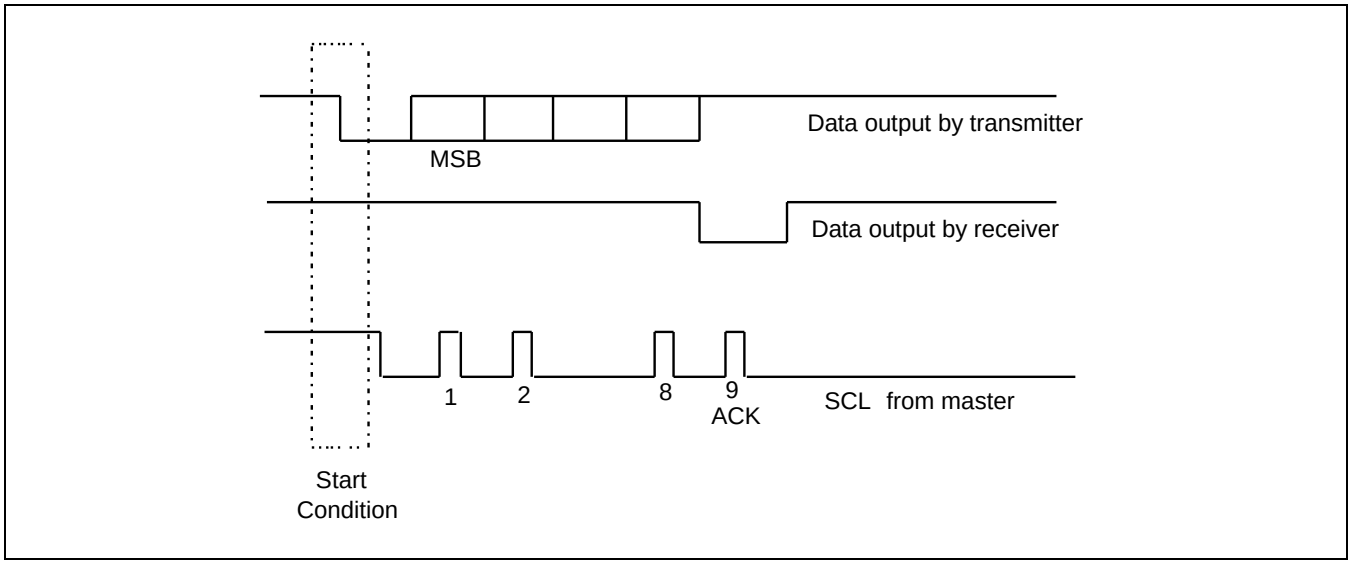


Figure 13-5 应答

13.2.2.4 时钟仲裁

13.2.2.4.1 同步

所有主机在I2C总线上传输数据的时候，都会产生它们自己的时钟。数据只有在时钟电平为高的时候有效。所以需要一个统一的时钟，以完成仲裁。

时钟同步利用连接到I2C接口的SCL线与功能实现。SCL上一个高到低的下降沿会让所有器件的低电平计数器开始计数，并且一旦有一个器件输出低电平，那么它就会拉低SCL直到时钟变高电平。然而，如果有其它时钟仍然处于输出低的状态，那么这个时钟的低到高的跳变并不会影响SCL的低输出。也就是说，SCL的低电平会保持低电平时间最长的那个时钟所输出的低，这时候更短低电平的时钟则进入一个等待高电平的状态。当所有器件的低电平都输出完以后，时钟信号变高。这时所有器件的时钟和SCL信号线之间就没有任何不同了，并且所有器件都开始输出高电平。第一个把高电平输出完的器件，会再次将SCL信号拉低。

在这个同步方法下，同步时钟的低电平由最长低电平周期的那个时钟产生，而高电平则由最短高电平的那个时钟产生。

13.2.2.4.2 仲裁

只有当总线空闲的时候，主机才可以发起一个传输。两个或多个主机有可能同时产生起始位，这时就需要仲裁。

仲裁发生在SCL是高的SDA传输线上，当某个主机A发送高电平的时候，其它主机正在发送低电平，那么主机A会检测到SDA上并不是它发送的电平，于是主机A中断它的数据输出，也就是丢失了仲裁。

仲裁可以在多个传输阶段上发生。第一个仲裁阶段是地址位的比较。如果多个主机都在同时寻址同一个器件，那么仲裁会继续在数据传输阶段发生。由于地址和数据都会被用来仲裁，所以传输过程中不会有信息丢失。

失去仲裁的主机会在丢失仲裁的那个字节传输中一直产生时钟脉冲。

如果一个主机还有从机功能并且在寻址阶段失去了仲裁，那么有可能赢得仲裁的主机正在寻址它。所以这个失去仲裁的主机应该马上转换成从机-接收模式。

由于I2C总线的控制权是单独由竞争主机发生的地址和数据决定的，所以总线没有中央主机，也没有任何优先权的机制。

特别要注意的一点，如果在一个串行传输中，仲裁发生在重复起始位或者停止位发送到I2C总线的瞬间，那么参与仲裁的主机需要在相同位置发送重复起始位或者停止位。也就是说，仲裁不允许发生在下面两个情况中间：

- 重复起始位和数据位
- 停止位和数据位
- 重复起始位和停止位

13.2.2.4.3 使用时钟同步机制作为握手

时钟的同步机制，除了可以在仲裁过程中使用，还可以用来让慢速的接收端与快速的发送端协同工作，支持字节协同和位协同。

对于字节协同工作的情况，慢速的器件可以用快的速度来接收传输的数据，但是需要时间来存储接收的字节或者准备另一个需要发送的字节。这种情况下，从机在收到和应答该字节后，拉低SCL，强制让主机进入等待状态，直到从机准备好下个字节的传输为止。

对于位协同的情况，比如一个单片机没有硬件I2C或者只有一个功能不全的I2C，那么它可以使用扩展时钟低电平时长的办法来降低传输速度，这样主机的速度就会自动适应为该单片机内部的速度。

13.2.2.4.4 7位寻址格式

起始位(S)后，发送的是从机地址。从机地址的长度为7位，第8位为数据方向位(读/写)——0表示发送(写)，1表示读请求(读)。数据传输总是由主机产生的停止位(P)来终止。但是，如果主机希望继续通信，那么它可以产生一个重复起始位(Sr)并且寻址其它从机，而不需要先产生一个停止位。各种读写格式的组合可以在这个传输中发生。

可以传输的格式为：

- 主机-发送端给从机-接收端发送数据。传输方向没有改变。
- 主机在第一个字节后，向从机读取数据

在第一个应答时刻，主机-发送端变成一个主机-接收端，而从机-接收端则变成一个从机-发送端。这个应答仍然由从机产生。

停止位由主机产生。

- 组合格式。在一个有方向变化的传输中，起始位和从机地址都会被重发，但是保留读写位。如果主机发送了重复起始位，那么之前它肯定发送了非应答位。

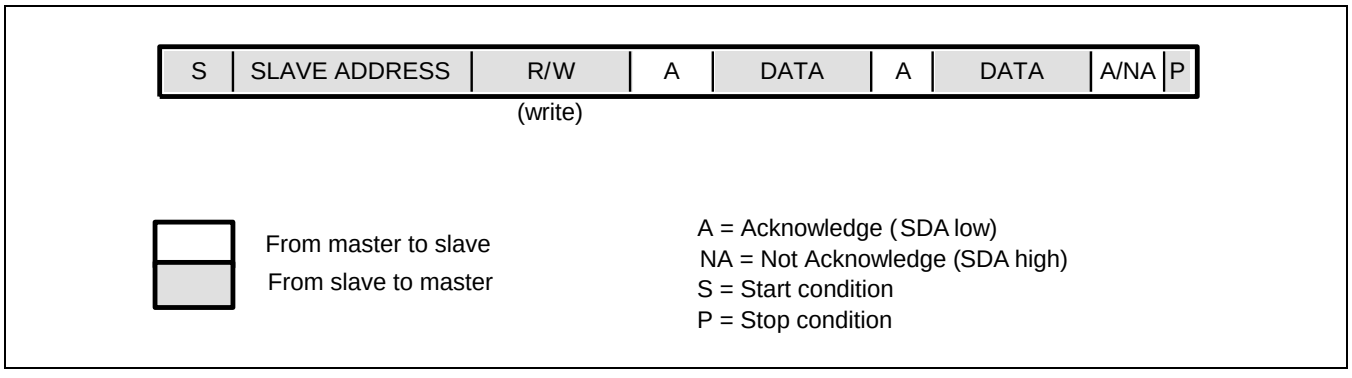


Figure 13-6 主机-发送端寻址从机

13.2.2.5 7位寻址

I2C总线的寻址过程是起始位后的第一个字节通常决定了主机要选择哪个从机。例外是“general call”寻址，可以寻址所有总线上的器件。当使用了这个地址时，总线上所有器件理论上都应该响应。然而，器件也可以设置成忽略该地址。”General call“的第二个字节则定义了接下来需要进行的操作。

13.2.2.5.1 定义第一个字节的各个位

第一个字节的前7位构成了从机地址。第8位是最低位LSB(least significant bit)，定义数据传输的方向。第8位LSB为0表明主机要向某个从机发送数据，而LSB为1则表明主机要从某个从机读数据。

当地址被发送后，系统里的每个器件在起始位后，都会将自己的地址和发送的地址进行比较，如果地址匹配，该器件就认为自己被主机选中为从机-接收端或者从机-发送端。是接收端还是发送端依赖于第8位读写位。

从机地址可以由一个固定部分和一个可编程部分组成。由于系统中很有可能存在一些相同地址的器件，所以从机地址中的可编程部分可以让这些器件尽可能的多。器件可编程地址的位数由器件中可用管脚的数量决定。例如，如果一个器件有4个固定地址位和3个可编程地址位，那么总共8个相同固定地址位的器件可以接到同一个I2C总线上。

I2C总线协议委员会负责协调I2C地址的分配。

两组共8个地址(0000XXX和1111XXX)保留为特殊用途，如下*Table 13-3*. 11110XX 的组合保留给10位寻址使用。

Table 13-3 第一个字节定义

从机地址	读写位	描述
0000 000	0	General call地址
0000 000	1	起始位(1)
0000 001	X	CBUS地址(2)
0000 010	X	保留给不同的总线格式(3)
0000 011	X	保留给将来使用
0000 1XX	X	HS模式主机代码
1111 1XX	X	保留给将来使用
1111 0XX	X	10位寻址

注意：

1. 所有器件都不允许在收到起始位后就应答。
2. CBUS 地址保留给 CBUS 兼容的器件和 I2C 总线兼容的器件混合使用。I2C 总线的器件收到该地址后不允许响应。
3. 该地址保留给其它不同总线。只有可以工作在这种总线和协议下的器件允许响应该地址。

General call地址用来寻址I2C总线上的每一个器件，但是如果一个器件不需要任何General call的数据，那么它可以通过不发送应答位来忽略该地址。如果一个器件确实需要从general call地址获取数据，那么它可以应答该地址并且以从机-接收端工作。

第二个字节和后面的字节都会被能处理该数据的从机-接收端应答。

如果不能处理这些字节中的某个字节，从机必须通过不发送应答位来忽略它。General call地址的功能，由第二个字节来指定。

需要考虑两种情况：

- LSB最低位B是0
- LSB最低位B是1

当最低位B是0，那么第二个字节有以下定义：

- 00000110 (H’06’). 复位并且由硬件写入从机地址的可编程部分。收到这个2-字节序列后，所有设计好能响应general call地址的器件将会复位，并且接收它们地址中的可编程部分。注意在上电后一定要保证器件不会拉低SDA和SCL，因为低电平会阻塞总线。
- 00000100 (H’04’). 由硬件写入从机地址的可编程部分。收到这个2-字节序列后，所有设计好能响应general call地址的器件将会接收它们地址中的可编程部分，但不会复位。
- 00000000 (H’00’). 不允许使用。

剩下所有的代码组合都还没有定义，并且所有器件都必须忽略它们。

当最低位B是1时，2-字节序列是一个硬件general call，意思是序列由一个硬件主机发送，比如键盘扫描器，它不能发送一个需要的从机地址。由于硬件主机不能事先知道数据需要发给哪个器件，所以它只能产生硬件general call和它自己的地址——把自己的信息发送给系统。

第二个字节中剩下的7位包含了该硬件主机的地址，这个地址可以由连接到总线上的智能设备(比如单片机)获取并且根据硬件主机的信息作出响应的动作。硬件主机还可以作为从机，从机地址跟主机地址一样。

在某些系统中，一种可能的情况是，硬件主机发送端在系统复位后被设为从机-接收端。

在这种情况下，系统设定好的主机可以告诉硬件主机-发送端(现在工作在从机-接收端模式)它需要发送的地址。在这个编程周期后，硬件主机仍然工作在主机-发送端模式。

13.2.2.5.2 起始字节

单片机可以用两种方法连接到I2C总线。带有I2C总线接口模块的单片机可以使用中断的方式处理总线的请求，但是当单片机没有接口模块，就必须用软件来实时查询监控总线。显然查询监控的次数越多，它能处理其它功能的时间就越少。所以带有硬件接口模块的单片机和依赖软件查询的单片机，有速度上的差异。

在这种情况下，数据传输可以由一个比通常时间要长的起始过程来进行。

这个起始过程由下面几个步骤组成：

- 一个起始位(S)
- 一个起始字节(00000001)
- 一个应答时钟脉冲(ACK)
- 一个重复起始位(Sr)

在主机发送一个起始位S请求占用总线后，再发送起始字节(00000001)。另一个单片机于是可以用较慢的查询速度来采样SDA传输线，直到检测到起始字节中任意一个低电平。在检测到这个SDA上的低电平后，单片机就可以切换到一个高速的采样频率来检测重复起始位Sr。

硬件接收端在收到重复起始位Sr后会复位，所以会忽略起始字节。

起始字节后会有一个应答位相关的时钟脉冲，这个脉冲只是为了让总线协议保持统一，起始字节不允许器件应答。

13.2.2.6 状态寄存器

按照工作模式，I2C状态代码可以分为四大类：主机发送模式状态码，主机接收模式状态码，从机接收模式状态码和从机发送模式状态码。所有状态代码的意义，软件执行的下一个动作以及I2C接口执行的下一个动作，见下面的表格：

Table 13-4 主机-发送模式的状态码

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
0x0000_0008	起始位已经发送	x	0	x	0	将从机地址写入I2C_DAT并且写入读写位，将I2C_CR中的SI清零	将发送从机地址和读写位，并且等待ACK
0x0000_0010	重复起始位已经发送	x	0	x	0	将从机地址写入I2C_DAT并且写入读写位，将I2C_CR中的SI清零	将发送从机地址和读写位，并且等待ACK。如果读写位是读，那么将切换到接收模式。
0x0000_0018	从机地址和读写位已经被发送，并且收到ACK	0	0	x	0	将数据写入I2C_DAT，将I2C_CR中的SI清零	数据字节将被发送，并且等待ACK.
		1	0	x	0	将I2C_CR中的STA置1，并将I2C_CR中的SI清零	将发送重复起始位
		0	1	x	0	将I2C_CR中的STO置1，并将I2C_CR中的SI清零	将发送停止位
		1	1	x	0	将I2C_CR中的STA和STO都置1，并将I2C_CR中的SI清零	将发送停止位，然后再发送起始位
0x0000_0020	从机地址和读写位已经被发送，但没有收到ACK	同上				同上	同上
0x0000_0028	数据已经被发送，并且收到ACK	同上				同上	同上
0x0000_0030	数据已经被发送，但没有收到ACK	同上				同上	同上
0x0000_0038	在发送从机地址和读写位，或者发送数据时，丢失了仲裁	0	0	x	0	将I2C_CR中的SI清零	释放I2C总线，切换到从机模式
		1	0	x	0	将I2C_CR中的STA置1，并将I2C_CR中的SI清零	等待直到I2C总线空闲，然后发送一个起始位

Table 13-5 主机-接收模式状态码

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
0x0000_0008	起始位已经发送	x	0	x	0	将从机地址写入I2C_DAT并且写入读写位，将I2C_CR中的SI清零	将发送从机地址和读写位，并且等待ACK
0x0000_0010	重复起始位已经发送	x	0	x	0	将从机地址写入I2C_DAT并且写入读写位，将I2C_CR中的SI清零	将发送从机地址和读写位，并且等待ACK。如果读写位是读，那么将切换到接收模式。
0x0000_0038	在发送从机地址和读写位时，丢失了仲裁	0	0	x	0	将I2C_CR中的SI清零	释放I2C总线，切换到从机模式
		1	0	x	0	将I2C_CR中的STA置1，并将I2C_CR中的SI清零	等待直到I2C总线空闲，然后发送一个起始位
0x0000_0040	从机地址和读请求已经被发送，并且收到ACK	0	0	1	0	将I2C_CR中的SI清零 将I2C_CR中的AA置1	将收到数据，然后返回ACK
		0	0	0	0	将I2C_CR中的SI清零 将I2C_CR中的AA清零	将收到数据，然后不返回ACK
0x0000_0048	从机地址和读请求已经被发送，但没有收到ACK	1	0	x	0	将I2C_CR中的STA置1，并将I2C_CR中的SI清零	将发送重复起始位
		0	1	x	0	将I2C_CR中的STO置1，并将I2C_CR中的SI清零	将发送停止位
		1	1	x	0	将I2C_CR中的STA和STO都置1，并将I2C_CR中的SI清零	将发送停止位，然后再发送起始位
0x0000_0050	收到数据位，并返回了ACK	0	0	1	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1	将收到下一个数据，然后返回ACK
		0	0	0	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零	将收到下一个数据，然后不返回ACK
0x0000_0058	收到数据位，但没有返回ACK	1	0	x	0	读数据，将I2C_CR中的STA置1，并将I2C_CR中的SI清零	将发送重复起始位

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
		0	1	x	0	读数据，将I2C_CR中的STO置1，并将I2C_CR中的SI清零	将发送停止位
		1	1	x	0	读数据，将I2C_CR中的STA和STO都置1，并将I2C_CR中的SI清零	将发送停止位，然后再发送起始位

Table 13-6 从机-接收模式状态码

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
0x0000_0060	收到本从机地址+写操作，返回了ACK	x	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	将会收到数据，并返回ACK
		x	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零	将会收到数据，但不返回ACK
0x0000_0068	发送从机地址+读写位时丢失仲裁(在主机模式下)，切换到从机模式，收到本主机地址，返回了ACK	x	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	将会收到数据，并返回ACK
		x	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零	将会收到数据，但不返回ACK
0x0000_0070	收到General Call 地址，返回了ACK	x	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	将会收到数据，并返回ACK
		x	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零	将会收到数据，但不返回ACK
0x0000_0078	发送从机地址+读写位时丢失仲裁(在主机模式下)，切换到从机模式，收到General Call 地址，返回了ACK	x	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	将会收到数据，并返回ACK
		x	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零	将会收到数据，但不返回ACK
0x0000_0080	被本机地址寻址到，写操作，收到数据，并返回了 ACK	x	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	将会收到数据，并返回ACK
		x	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零	将会收到数据，但不返回ACK
0x0000_0088	被本机地址寻址到，写操作，收到数据，但没有返回ACK	0	0	0	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零	切换到“未被选中”的从机模式，禁止本机地址识别和general call地址识别。
		0	0	1	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1	切换到“未被选中”的从机模式，应答本机地址寻址和general call (如果I2C_ADR的GC=1)
		1	0	0	0	读数据，将I2C_CR	切换到“未被选中”的从

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
						中的SI清零，将I2C_CR中的AA清零，将I2C_CR中的STA置1	机模式，禁止本机地址识别和general call地址识别。一旦总线空闲，将马上发送起始位。
		1	0	1	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1，将I2C_CR中的STA置1	切换到“未被选中”的从机模式，应答本机地址寻址和general call (如果I2C_ADR的GC=1)。一旦总线空闲，将马上发送起始位。
0x0000_0090	被general call寻址选中，收到数据，并返回了ACK	x	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	将会收到数据，并返回ACK
		x	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零	将会收到数据，但不返回ACK
0x0000_0098	被general call寻址选中，收到数据，但没有返回ACK	0	0	0	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零	切换到“未被选中”的从机模式，禁止本机地址识别和general call地址识别。
		0	0	1	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1	切换到“未被选中”的从机模式，应答本机地址寻址和general call (如果I2C_ADR的GC=1)
		1	0	0	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零，将I2C_CR中的STA置1	切换到“未被选中”的从机模式，禁止本机地址识别和general call地址识别。一旦总线空闲，将马上发送起始位。
		1	0	1	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1，将I2C_CR中的STA置1	切换到“未被选中”的从机模式，应答本机地址寻址和general call (如果I2C_ADR的GC=1)。一旦总线空闲，将马上发送起始位。
0x0000_00A0	仍然被当作从机被寻址到时，收到停止位或者重复起始位	0	0	0	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零	同上

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
		0	0	1	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1	
		1	0	0	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零，将I2C_CR中的STA置1	
		1	0	1	0	读数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1，将I2C_CR中的STA置1	

Table 13-7 从机-发送模式状态码

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
0x0000_00A8	收到本从机地址+读操作，返回了ACK	x	0	1	0	写数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1	数据将被发送
		x	0	0	0	写数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零	最后一个字节的数据将被发送，并且之后从机再也不响应
0x0000_00B0	在作为主机发送从机地址+读写位时丢失了仲裁；收到了本从机地址，并返回了ACK	x	0	1	0	写数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1	数据将被发送
		x	0	0	0	写数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零	最后一个字节的数据将被发送，并且之后从机再也不响应
0x0000_00B8	数据已经被成功发送，并且返回了ACK	x	0	1	0	写数据，将I2C_CR中的SI清零，将I2C_CR中的AA置1	数据将被发送
		x	0	0	0	写数据，将I2C_CR中的SI清零，将I2C_CR中的AA清零	最后一个字节的数据将被发送，并且之后从机再也不响应
0x0000_00C0	数据已经被成功发送，但没有收到ACK	0	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零	切换到“未被选中”的从机模式，禁止本机地址识别和general call地址识别。
		0	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	切换到“未被选中”的从机模式，应答本机地址寻址和general call (如果I2C_ADR的GC=1)
		1	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零，将I2C_CR中的STA置1	切换到“未被选中”的从机模式，禁止本机地址识别和general call地址识别。一旦总线空闲，将马上发送起始位。
		1	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1，将I2C_CR中的STA置1	切换到“未被选中”的从机模式，应答本机地址寻址和general call (如果I2C_ADR的GC=1)。一旦总线空闲，将马上发送起始位。
0x0000_00C8	最后一个字节的数据	0	0	0	0	将I2C_CR中的SI清	

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
	已经被成功发送，并且收到了ACK					零，将I2C_CR中的AA清零	同上
		0	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	
		1	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零，将I2C_CR中的STA置1	
		1	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1，将I2C_CR中的STA置1	

Table 13-8 其它状态码

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
0x0000_00F8	没有相关的状态信息；I2C_CR中的SI=0	–	–	–	–	无动作	等待或者进行当前的操作
0x0000_0000	由于非法的起始位或者停止位产生的总线错误	0	1	x	0	无动作	只有内部硬件会被影响。任何情况下，总线都会被释放，并且STO被清零。

13.2.3 I2C总线规范的扩展

以100kbit/s速度传输数据和7位寻址的I2C总线协议已经存在三十多年没有变化了。I2C的总线概念已经成为世界范围内的标准，市面上有成千上万种兼容I2C总线的芯片。现在I2C总线规范可以扩展下面两种特性：

- 支持高达400kbit/s传输速度的快速模式
- 10位寻址模式，支持1024个地址空间

扩展I2C总线规范有两个原因：

- 新兴应用会需要传输更多的串行数据，从而需要比100kbit/s更快的速度。IC制造技术的进步可以在不增加成本的前提下支持4倍甚至更高的速度。
- 7位寻址所支持的112个地址已经被授权多次。为了避免地址重复的问题，地址需要更多的组合。使用新的10位寻址可以获得约10倍的可用地址空间。

所有新的I2C总线接口器件都支持快速模式，他们更希望以400kbit/s的速度接收或者发送数据。最低的需求是它们能同步一个400kbit/s的传输；它们也能延长SCL信号的低电平以降低传输速度。快速模式的器件必须向下兼容，也就是能够跟100kbit/s的器件进行通信。

显然0到100kbit/s的器件不能在快速I2C总线的系统里工作，因为它们无法跟上更高的传输速度，有可能发生无法预测的问题。

支持快速I2C总线接口的从机可以使用7位或者10位寻址，但是推荐使用7位寻址方式，因为7位寻址成本更低而且传输的数据相对更也少。7位寻址和10位寻址的器件可以混合使用在同一个I2C总线系统中，不管系统是工作在0到100kbit/s的标准模式还是0到400kbit/s的快速模式。当前存在的主机和将来的主机都可以产生7位或者10位地址。

13.2.4 快速模式

在快速模式中，之前I2C总线规范定义的协议，格式，逻辑电平和SDA/SCL传输线上的最大负载电容都保持不变。跟之前规范不同的是：

- 最大比特率增加到400kbit/s
- 串行数据SDA和串行时钟SCL信号的时序不同。不需要兼容其它总线系统比如CBUS，因为它们不能工作在这个速度。
- 工作在快速模式的器件必须在输入上抑制毛刺信号，并且输入端需要施密特触发器。
- 工作在快速模式的器件必须在输出端设计SDA和SCL信号的下降沿斜率控制。
- 如果工作在快速模式的器件掉电了，那么SDA和SCL的IO管脚必须处于悬空状态，避免干扰总线。
- 接到总线上的外部上拉器件必须适配快速模式的I2C总线所允许的信号上升时间。对于总线负载电容小于200pF的情况，上拉器件可以是一个电阻；对于总线负载电容在200pF到400pF之间的情况，上拉器件可以是一个电流源(最大3mA)或者一个开关电阻。

13.2.4.1 10位寻址

使用10位寻址不改变I2C总线规范的协议。10位地址开发利用了起始位(S)和重复起始位(Sr)后第一个字节的前7位中保留的1111XXX组合。

10位地址也不影响现有的7位寻址方式。7位寻址和10位寻址的器件可以接在同一个I2C总线上，并且7位寻址和10位寻址的器件都可以用在标准模式(100kbit/s)的系统中或者快速模式(400kbit/s)的系统中。

尽管保留地址1111XXX有8种可能的组合，但是只有4种组合11110XX是10位寻址可用的。剩下的11111XX组合保留给将来使用。

A – 头两个字节的位定义

10位地址由起始位(S)或者重复起始位(Sr)后的头两个字节组成。

第一个字节的前7位是11110XX，其中的最后两位XX是10位地址的最高两位(MSB)；第一个字节的第8位是读写位，用来定义传输方向，0表示主机写从机，1表示主机读从机。

如果读写位是0，那么第二个字节为剩下的8位地址(XXXXXXXX)。如果读写位是1，那么下个字节为从机发给主机的数据。

B – 10位寻址方式

10位寻址的传输中可能包含各种读写的组合。可能涉及到的数据传输格式有：

- 主机-发送端给从机-接收端发送一个10位的从机地址，数据传输方向不变化。在起始位后，各个从机将自己的地址跟第一个字节的前7位(11110XX)进行比较，并且判断第8位读写位是否为0。很有可能多个器件都能匹配上，并且发送一个应答位(A1)。所有匹配上的从机将继续比较第二个字节的8位从机地址(XXXXXXXX)，这时候应该只有1个从机匹配，并且发送应答位(A2)。匹配上的从机将一直保留这个被选中的状态，直到它收到停止位(P)或者后面跟着不同从机地址的重复起始位(Sr)。
- 主机-接收端使用10位地址向从机-发送端读取数据，在第二个读写位后传输方向发生了变化。直到应答位A2，读取的过程都跟上面发送的过程一样。在重复起始位(Sr)后，匹配的从机会记住自己是被选中过的。然后这个从机比较重复起始位Sr后第一个字节的前7位是否跟起始位后的7位相同，并且判断第8位是否为1，如果是的话，从机认为自己被寻址到，并且选中为发送端，于是该从机发送应答位A3。

从机-发送端会一直保留被选中的状态，直到它收到一个停止位(P)或者一个跟着不同从机地址的重复起始位(Sr)。在重复起始位(Sr)后，所有其它从机也会都开始比较第一个字节的前7位(11110XX)，并且判断读写位。但是，由于读写位为1(对10位地址的器件)，或者从机地址位11110XX(对7位地址的器件不匹配)，所以它们中没有任何一个会被寻址选中。

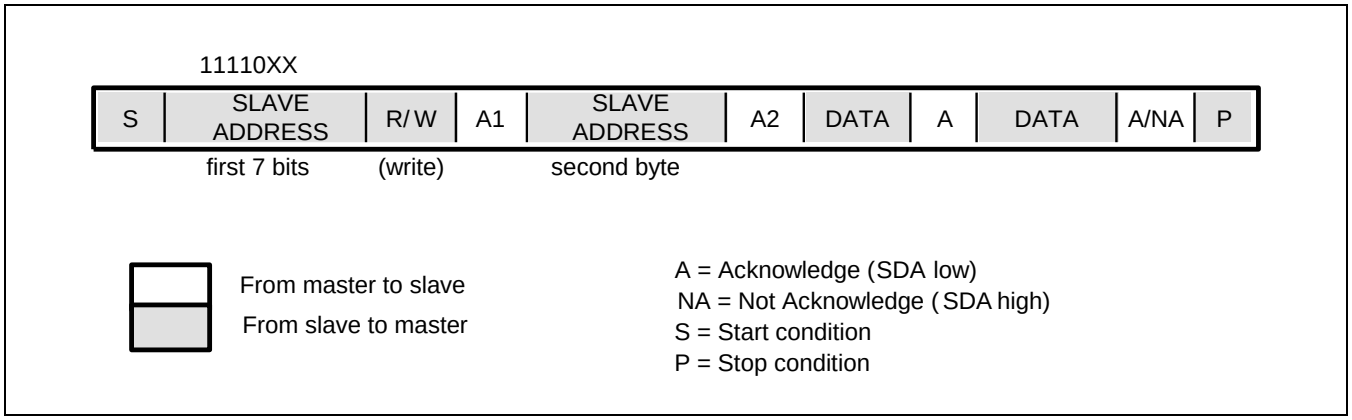


Figure 13-7 主机-发送端用10位地址寻址从机-接收端

13.2.4.2 General Call寻址和起始字节

I2C总线10位地址的寻址过程是起始位后的头两个地址决定哪个从机被主机选中。其中的例外就是“general call”地址00000000 (H’00’)。

10位寻址方式的从机跟7位寻址的从机一样，会响应”general call”寻址。

硬件主机可以在”general call”后发送它们的10位地址。这种情况下，”general call”地址后，紧接着两个连续的字节，这两个字节包含的是主机-发送端的10位地址。

10位寻址中起始字节00000001 (H’01’)的产生跟7位寻址一样。

13.3 I2C时序

Table 13-9 时序要求

Parameter	Symbol	标准模式的I2C总线		快速模式的I2C总线		Unit
		Min	Max	Min	Max	
SCL时钟周期	FSCL	0	100	0	400	kHz
停止位和起始位之间的总线空闲时间	TBUF	4.7	–	1.3	–	us
(重复)起始位Hold time 这个时间后，产生第一个时钟脉冲	THD;STA	4.0	–	0.6	–	us
SCL时钟的低电平时长	TLOW	4.7	–	1.3	–	us
SCL时钟的高电平时长	THIGH	4.0	–	0.6	–	us
重复起始位的Set-up time	TSU;STA	4.7	–	0.6	–	us
数据位 hold time	THD;DAT	0	–	0	0.9	us
数据位 set-up time	TSU;DAT	250	–	100	–	ns
SDL和SCL信号的上升时间	Tr	–	1000	20+01Cb	300	ns
SDL和SCL信号的下降时间	Tf	–	300	20+01Cb	300	ns
停止位的Set-up time	TSU;STO	4.0	–	0.6	–	us
每个信号线的负载电容	Cb	–	400	–	400	pF

13.4 寄存器说明

13.4.1 寄存器表

Base Address of I2C: 0x400A0000

Register	Offset	Description	Reset Value
I2C_ECR	0x050	时钟使能寄存器	0x00000000
I2C_DCR	0x054	时钟禁止寄存器	0x00000000
I2C_PMSR	0x058	电源管理状态寄存器	0x00000000
I2C_CR	0x060	控制寄存器	0x00000000
I2C_MR	0x064	模式寄存器	0x000001F4
I2C_SR	0x070	状态寄存器	0x000000F8
I2C_IER	0x074	中断使能寄存器	0x00000000
I2C_IDR	0x078	中断禁止寄存器	0x00000000
I2C_IMR	0x07C	中断状态寄存器	0x00000000
I2C_SDR	0x080	数据寄存器	0x00000000
I2C_ADR	0x084	从机地址寄存器	0x00000000
I2C_THOLD	0x088	Hold/Setup延时控制寄存器	0x00000001

13.4.2 I2C_ECR(时钟使能寄存器)

Address = Base Address+ 0x050, 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	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	调试使能控制位 0 = 无效 1 = 使能I2C模块的调试功能
CLKEN	[1]	W	时钟使能控制位. 0 = 无效 1 = 使能I2C时钟

13.4.3 I2C_DCR(时钟禁止寄存器)

Address = Base Address+ 0x054, 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	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	调试使能控制位 0 = 无效 1 = 使能I2C模块的调试功能
CLKEN	[1]	W	时钟禁止控制位. 0 = 无效 1 = 禁止I2C时钟

13.4.4 I2C_PMSR(电源管理状态寄存器)

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
DBGEN	RSVD	IPIDCODE																										RSVD		CLKEN	RSVD
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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 = dbgack_sclk输入对I2C功能无影响 1 = dbgack_sclk被使能。 当这位是低，I2C功能保持不变。当这位是高，I2C的功能被冻结无法使用，但是寄存器的读写功能不受影响，以方便调试。
IPIDCODE	[29:4]	R	模块版本信息，共26位
CLKEN	[1]	R	CLKEN : 时钟使能/禁止状态 0 = I2C时钟被禁止 1 = I2C时钟被使能

13.4.5 I2C_CR(控制寄存器)

Address = Base Address+ 0x060, 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																							ENA	RSVD			SI	STA	STO	AA	SWRST
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	RW	RW	RW	RW	RW

Name	Bit	Type	Description
ENA	[8]	RW	I2C使能 0 = 禁用I2C接口(当I2C接口被禁用，SCL和SDA也被禁用，没有任何输出和输入) 1 = 使能I2C接口
SI	[4]	RW	SI : I2C中断 0 = 清除SI 1 = 有中断需要处理。当SI为1,时，i2c_int信号为高，并且SCL传输线会被拉低。传输被暂停，直到SI被清除。
STA	[3]	RW	I2C启动 0 = 工作在从机模式 1 = 当设置为1，I2C接口工作在主机模式并且检测I2C总线的状态，如果总线处于空闲，那么产生一个起始位。如果总线不空闲，那么I2C接口将等到停止位后再产生一个起始位(在一个最小时间后)。当起始位成功产生后，该位会被自动清零。
STO	[2]	RW	I2C停止 0 = 不会在总线上发送停止位 1 = 产生一个停止位。当检测到总线上有停止位时，STO位会被自动清除。在从机模式，这个位用来从总线错误中恢复。在这种情况下，不发送停止位，但是I2C接口会认为已经收到停止位并且切换到“未被寻址到”的从机模式(STO位会被I2C接口硬件清除)
AA	[1]	RW	I2C应答. 0 = 不发送应答位 (应答SCL时钟脉冲期间内SDA保持高) 1 = 当收到从机地址(或者当I2C_ADR的GC为为1时收到general call地址)，或者在接收模式收到了数据，发送应答位
SWRST	[0]	RW	SWRST : I2C软件复位 0 = 无效 1 = 产生一个软件复位(I2C_PMSR寄存器不会被复位)

13.4.6 I2C_MR(模式寄存器)

Address = Base Address+ 0x064, Reset Value = 0x000001F4

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

Name	Bit	Type	Description
FAST	[12]	RW	快速模式 0 = 禁用快速模式，使能标准模式。在这个模式下，高电平和低电平的比为1:1并且最大波特率为100kHz。 1 = 使能快速模式。在这个模式下，高电平和低电平的比为2:3并且最大波特率为400kHz。 注意：配置为400K的时候，推荐外接2K上拉电阻。
PRV	[11:0]	RW	预分频值 这个值用来设置总线的速度(FSCL). PRV (pre-scaler Value)的值用下面的公式来产生FSCL： FSCL = PCLK/(PRV+4)

13.4.7 I2C_SR(状态寄存器)

Address = Base Address+ 0x070, Reset Value = 0x000000F8

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

Name	Bit	Type	Description
SR	[7:3]	R	I2C状态代码 接口状态代码，共有27种可能的状态 I2C_SR复位值是0x000000F8. 当I2C_SR的值是这个复位值时，表明当前没有任何相关信息。所有其它状态值都跟某个工作状态有关。当I2C接口的状态机执行到某个状态时，这个寄存器的值也会更新到该状态代码，并且SI中断位会被置1。 所有这些状态代码的意义，软件执行的下一个动作以及I2C接口执行的下一个动作，都在下一页中描述。

13.4.8 I2C_IER(中断使能寄存器)

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																											SI	RSVD				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	R	R	R	R

Name	Bit	Type	Description
SI	[4]	W	SI : SI中断使能 0 = 无效 1 = 使能SI中断

13.4.9 I2C_IDR(中断禁止寄存器)

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																											SI	RSVD				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	R	R	R	R

Name	Bit	Type	Description
SI	[4]	W	SI : SI中断禁止 0 = 无效 1 = 禁止SI中断

13.4.10 I2C_IMR(中断状态寄存器)

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																											SI	RSVD				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
SI	[4]	R	SI使能后的中断状态 当这位是1时，表示有中断需要处理，这时i2c_int为高，SCL传输线则被拉低，传输被暂停，直到SI被处理完并清零。

13.4.11 I2C_SDR(数据寄存器)

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																								DAT							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
DAT	[7:0]	RW	I2C数据. 在接收模式下，该字节是从I2C总线上收到的数据；在发送模式下，该字节是需要发送到I2C总线上的数据。 DAT总是从右向左移，也就是说，在发送模式下，总是先发送最高位MSB，而在接收模式下，也是先接收到最高位MSB。 在一个发送过程中(该模块往I2C总线发送数据)，当数据被移位出去的时候，SDA传输线上的数据同时也被移位进来。所以在丢失仲裁的情况下，DAT将会包含正确的数据字节(从总线上读到的值)。

13.4.12 I2C_ADR(从机地址寄存器)

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																								ADR							GC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
ADR	[7:1]	RW	I2C地址 包含一个7位的I2C地址，如果I2C接口被设定为一个从机(发送端或者接收端)，那么将会响应这个从机地址。
GC	[0]	RW	General Call. 使能对general call地址的响应功能，当GC位置1时，如果识别到general call地址，I2C接口会产生中断。

13.4.13 I2C_THOLD(Hold/Setup延时控制寄存器)

Address = Base Address+ 0x088, 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																								DL							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
DL	[7:0]	RW	Hold/setup延时. Hold/Setup延时的值，由下面公式计算： THOLD = DL[7-0] x PCLK, TSETUP = DL[7-0] x PCLK 注意： 1. 复位后的DL值为’1’ (十六进制). 2. Setup延时(TSETUP) 必须至少为250 ns (标准模式)， 至少为100 ns (快速模式). 3. I2C器件必须在内部保证SDA信号上至少有300ns的hold时间。用户必须保证正确的hold值来满足慢速器件的时序。 4. DL的值不允许为0。

14 串行外设接口 (SPI)

14.1 概述

SPI，串行外设接口，又叫同步串行端口 (SPI)，用来连接串行外设。
注：如果系列内芯片不具有本外围，那它就不具备本章描述的相关资源。具体参考芯片的数据手册。

14.2 主要功能

- 主机或者从机选择
- 可编程的时钟比特率和分频。比特率支持2MHz以及更高，跟配置的SPICLK频率有关。同时，最高频率受限于外围器件和外围电路。
- 可编程的时钟相位和极性
- 分开的发送和接收FIFO缓存，32位宽，1个地址深
- 可编程帧格式，支持4-16位数据宽度
- 独立可控制的发送FIFO中断，接收FIFO中断和溢出中断
- 内部环回测试模式
- 中断控制

SPI兼容摩托罗拉(Motorola) SPI，在主机和从机配置下，可以进行：

- 发送FIFO的并行数据转串行数据，内部FIFO有32位宽，1地址深
- 接收到的数据串行转并行，缓存到一个32位宽，1地址深的FIFO

14.2.1 管脚描述

Table 14-1 SPI 管脚描述

管脚名称	功能	I/O类型	说明
SPI_SCK	SPI 串行时钟	I/O	–
SPI_MOSI	主机输出从机输入	I/O	–
SPI_MISO	主机输入从机输出	I/O	–
SPI_NSS	帧，从机选择 (作为主机时) 帧输入 (作为从机时)	I/O	–

14.3 功能描述

14.3.1 模块框图

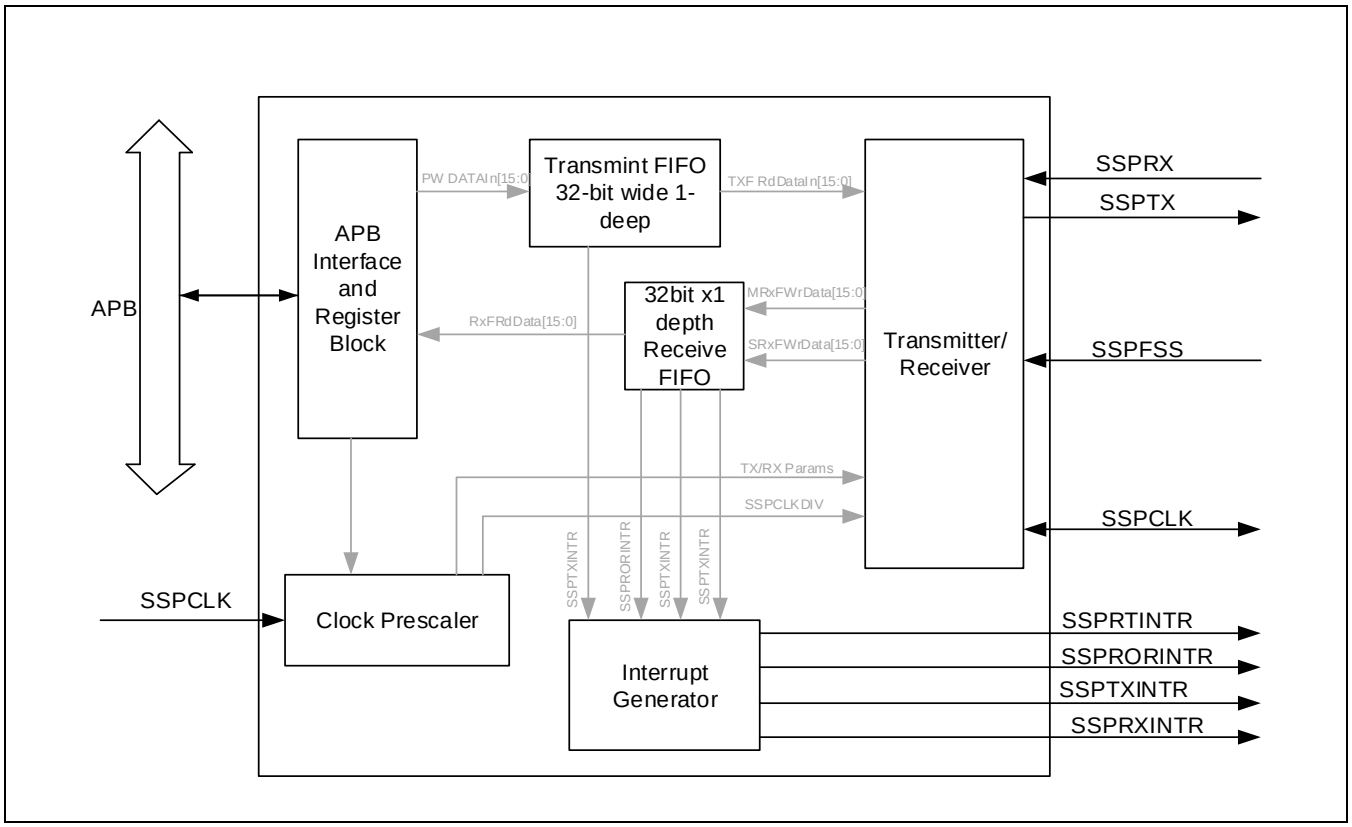


Figure 14-1 SPI 模块框图

14.3.2 功能描述

SPI模块包含时钟分频模块、一个发送FIFO，一个接收FIFO、一个收发器和一个中断控制器。

14.3.2.1 时钟分频

当配置成主机时,有2个串联在一起的分频计数器，用来给串行输出时钟SPICLK提供时钟源。

用户可以通过SPICPSR寄存器来配置预分频器，分两步给FSPICLK进行2到254分频。SPICPSR寄存器的最低位被被设计成无法使用，也就是无法使用奇数分频，保证了产生时钟的对称性(1/0占空比相等)。

预分频的输出接到了另一个1到256的分频器，通过SPICR0可配置，这个分频器的输出最终输出到SPICLK管脚。

14.3.2.2 发送FIFO

发送FIFO是一个32位宽，1地址深的先进先出缓冲区。实际发送时当数据位小于或者等于8位时，发送FIFO相当于有8bits宽的4级深FIFO。当数据位数大于8位时，发送FIFO相当于16bits宽，2级深FIFO。CPU通过AMBA APB接口将数据写入发送FIFO中当前指针所指的空间，直到发送逻辑将数据读出。在通过SPITXD管脚串行发送数据前，需要先将并行的数据写入发送FIFO。

发送的数据如果小于16位，在发送缓冲区内也是右对齐的。

14.3.2.3 接收FIFO

接收FIFO是一个32位宽，1地址深的先进先出缓冲区。实际接收时当数据位小于或者等于8位时，接收FIFO相当于有8bits宽的4级深FIFO。当数据位大于8位时，接收FIFO相当于16bits宽，2级深FIFO。从串行接口接收到的数据会存在接收FIFO中当前读指针所指的空间，直到被CPU通过AMBA APB总线读出。在通过SPIRXD管脚收到串行数据后，才能读取FIFO的并行数据。

接收到的数据如果小于16位，在接收缓冲区内也是右对齐的。

14.3.2.4 发送和接收单线模式

SPI支持单线收发模式，在某些特殊应用中，SPI可以只使用一个端口进行通讯。SPI_CR1寄存器中的LPMD位用来使能单线模式，只有SPI的主机模式支持该单线功能。

在单线模式中，SPI将只使用SPI_MOSI端口进行通讯，SPIRX和SPITX都将接在SPI_MOSI管脚上，SPIRX将会一直接收SPI_MOSI上的信号，SPITX则通过SPI_CR1中的LPTXOE位来控制是否连接到SPI_MOSI管脚进行发送。在需要发送时，将LPTXOE置1，SPITX信号将通过SPI_MOSI管脚将数据发出；在需要接收时，将LPTXOE置0，这样SPITX发送出去的信号就不会与从机发来的信号在通讯线上产生冲突。

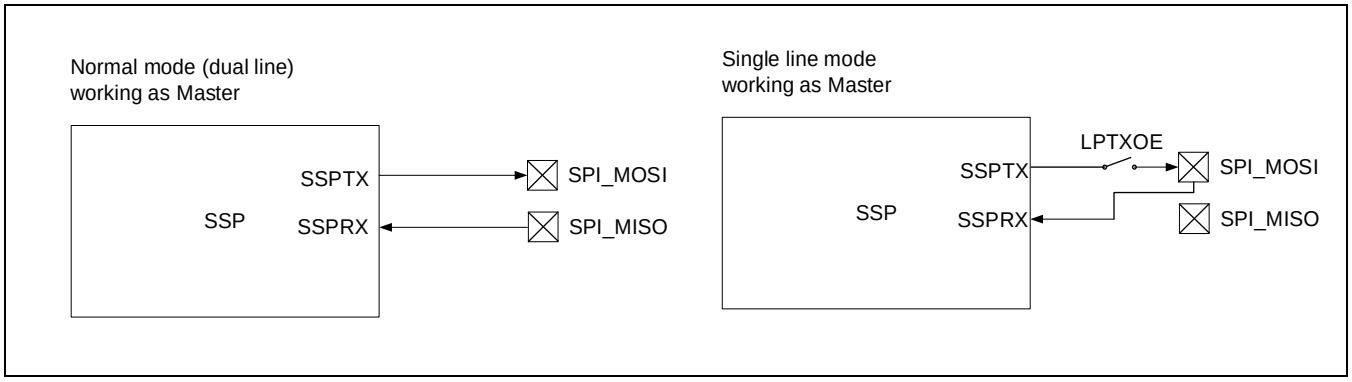


Figure 14-2 SPI 单线通讯模式

14.3.2.5 中断产生逻辑

SPI可产生并配置4种中断:

- SPIRXINTR : SPI接收FIFO中断服务请求
  - 当接收FIFO的占用到一定数量后，会触发该中断。这个数量由SPI_CR1中的RXIFLSEL位设置。
    - RXIFLSEL=1时，接收FIFO接收至少为1个数据时，会触发该中断。
    - RXIFLSEL=2时，如果数据位数（通过CR0[DSS]配置）不超过8bit，接收FIFO接收两个数据，会触发该中断；如果数据位数（通过CR0[DSS]配置）大于8bit，接收FIFO接收1个数据，会触发该中断。

- SPITXINTR : SPI发送FIFO中断服务请求
  - 当发送FIFO中数据位数少于16bit（数据位数8bit以内，少于两个数；8bit以上，少于1个数）时，会触发该中断。这个发送中断SPITXINTR不需要SPI使能，所以数据的发送可以用两种方法操作。一是数据可以在使能SPI和中断前就写入发送FIFO中，二是中断使能后在发送FIFO中断服务子程序中写入数据。

- SPIRORINTR : SPI接收溢出中断请求
  - 当接收FIFO满后还收到了数据帧，会触发该中断。这个中断发生说明FIFO溢出了，此时新接收到的数据会覆盖接收移位寄存器，而不会写入FIFO中。

- SPIRTINTR : SPI超时中断请求
  - 当接收FIFO中有数据未被读取，并且SPI在32个位周期时长内一直处于空闲状态，会触发该中断。这个机制可以让用户知道接收FIFO中有数据需要处理。在接收FIFO被读取变空后，或者在SPIRXD上接收到新数据后，该中断会被清除。SPI_ICR寄存器中的RTIC位也可以清除该中断。

4种中断通过或逻辑后，发送给CPU请求中断处理，在处理程序中，CPU需要读取SPI状态寄存器的值来判断发生的是哪种中断。用户可以通过SPI_IMCR寄存器中的相应位使能或者禁止这些中断。

14.3.3 SPI操作方法

14.3.3.1 复位SPI

除了系统复位外，SPI 模块内部的复位可以在两个时钟域内进行。一个是 SPICLK 时钟域，用来复位 SPI 状态机。一个是 PCLK 时钟域，用来复位寄存器设置。用户可以根据不同的应用需求做不同的复位动作。

14.3.3.2 主从配置

通过控制寄存器SPICR1[MS]可以将SPI配置为主机或者从机。在主机模式下，模块控制整个传输过程，在从机模式下，模块在传输中处于被动状态。时钟由网络中的主机提供。此时仅当模块被选中时，才可能有TX管脚的输出。下图演示了SPI如何连接到其它同步串行接口的例子。

注意：SPI 不支持在同一个系统中动态切换主机和从机配置，只能是配置成单一主机或者单一从机。

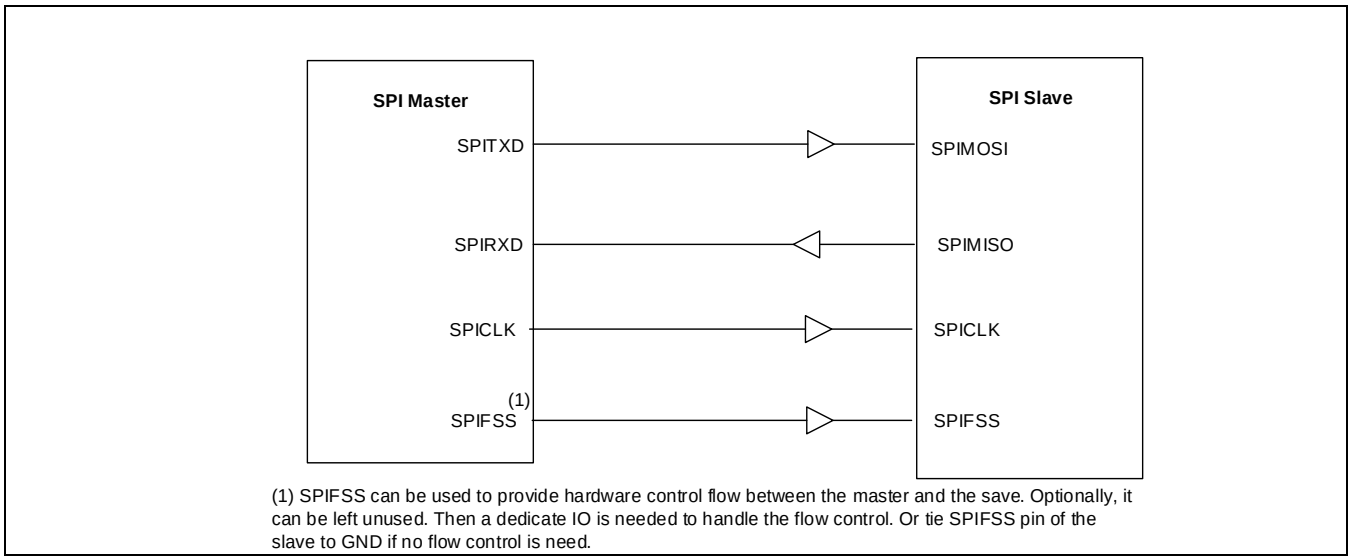


Figure 14-3 SPI主机连接1个SPI从机

上图为SPI配置成主机，连接了一个Motorola SPI从机。

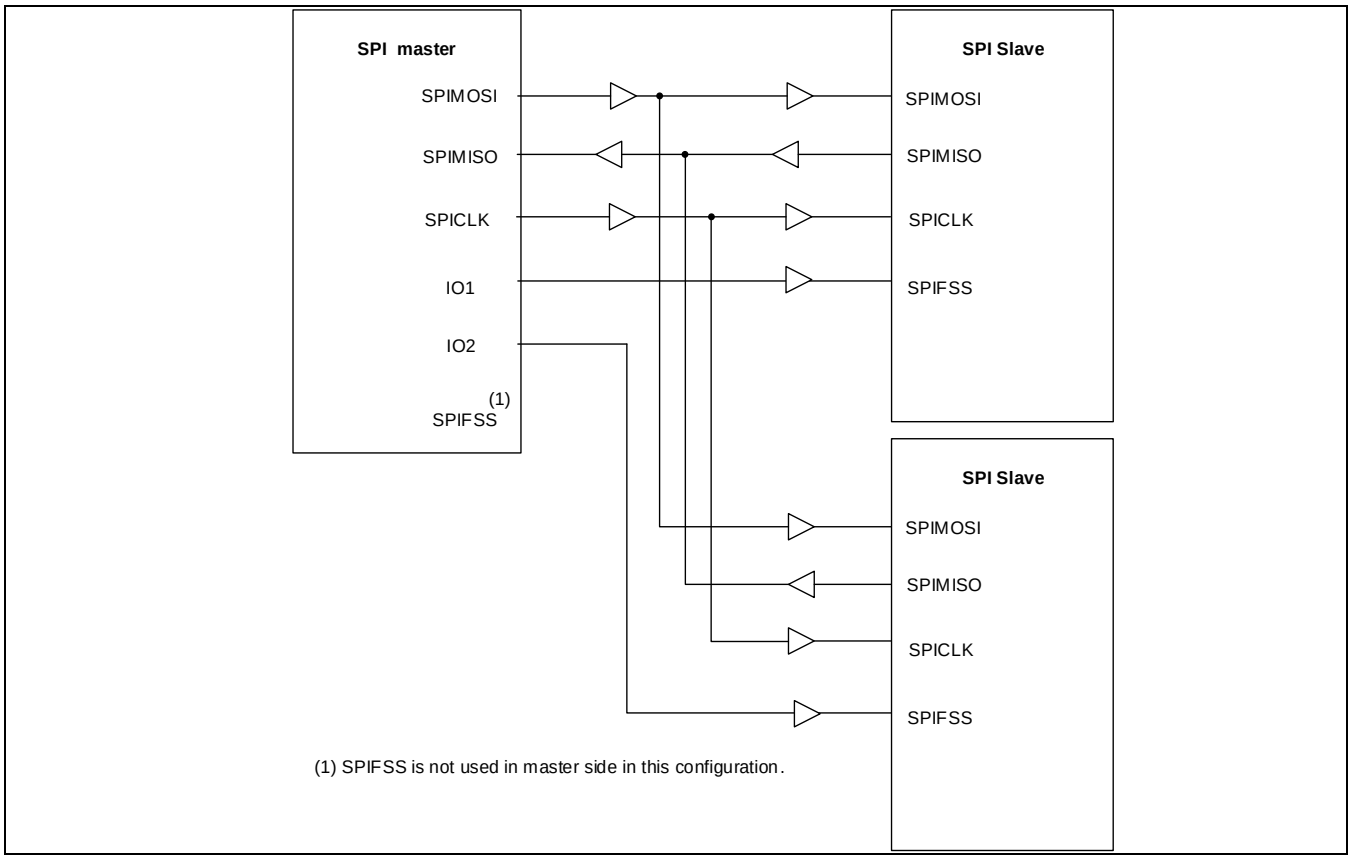


Figure 14-4 SPI 主机连接到两个SPI从机

上图为Motorola SPI作为主机，连接到两个SPI从机。在这种情况下，IO1和IO2用来控制从机片选。因为所有MISO管脚连到了一起，所以所有从机的MISO管脚必须使能开漏输出。

14.3.3.3 时钟配置

在从机模式下，为了保证时钟线和数据线被正确采样和同步，PCLK ≥ 6 x SPICLK 。即使是在主机模式下，为了保证输出50%占空比的时钟，PCLK ≥ 2 x SPICLK。如下表所示：

Table 14-2 PCLK和SPICLK的关系限制

	主机模式	从机模式
SPICLK和PCLK的关系	254×256×SPICLK (out)≥PCLK ≥ 2 x SPICLK	254×256×SPICLK(IN)≥PCLK ≥ 6 x SPICLK

其中，254是SPI_CPSR[CPSDVSR]的最大值。256是SPI_CR0[SCR]的最大值。

所以在主机模式，要产生1MHz的比特率（SPICLK = 1MHz），PCLK至少为2MHz。此时如果PCLK=2MHz，因为SPI_CPSR[CPSDVSR] 最小只能设置为2，所以SPI_CR0[SCR]的值必须为0。

在从机模式，如果同样要工作在1MHz的比特率，那么PCLK的频率必须至少为6MHz。此时

SPI_CPSR[CPSDVSR]和SPI_CR0[SCR]的设置不是唯一的。一种可行的配置是：SPI_CPSR[CPSDVSR] = 6，SPI_CR0[SCR] = 0。

14.3.3.4 帧格式配置

每个数据帧的长度可以配置为4到16位，由SPI_CR0[DSS]决定，发送时从MSB高位开始发送。

SPI支持Motorola SPI帧格式，是一个4线的接口。其中SPIFSS信号用作从机选择。对于所有格式，当SPI在空闲时，串行时钟(SPICLK管脚)都会被置于非活动状态（由SPI_CR0[SPO]决定实际电平）。这个非活动状态可以被用来提供一个接收超时功能，表示在某个时长后，FIFO内接收到的数据仍未被读取。SPICLK管脚信号相位可以用SPI_CR0[SPH]位选择。

- SPO，时钟极性

如果SPO=0，那么在数据没有被传送时，SPICLK管脚的稳定状态为低电平；如果SPO=1，那么SPICLK管脚的稳定状态为高电平。

- SPH，时钟相位

SPH控制位可以选择捕获数据的时钟沿。当SPH=0，数据在第一个时钟沿捕获，而当SPH=1，数据在第二个时钟沿捕获。发送的数据也需要在对应的时钟沿到来前准备完毕。

为了正确的数据传输，传输网络上的所有 SPI 设备中这两个参数都需要保证同样的设置。

14.3.3.4.1 Motorola SPI 格式, SPO = 0, SPH = 0

SPO = 0，无数据传输时，SPICLK 处于低电平（开始阶段）。

SPH = 0，第一个时钟沿是上升沿，所以接收数据捕获即发生于上升沿，发送数据则在上升沿时处于稳定状态。

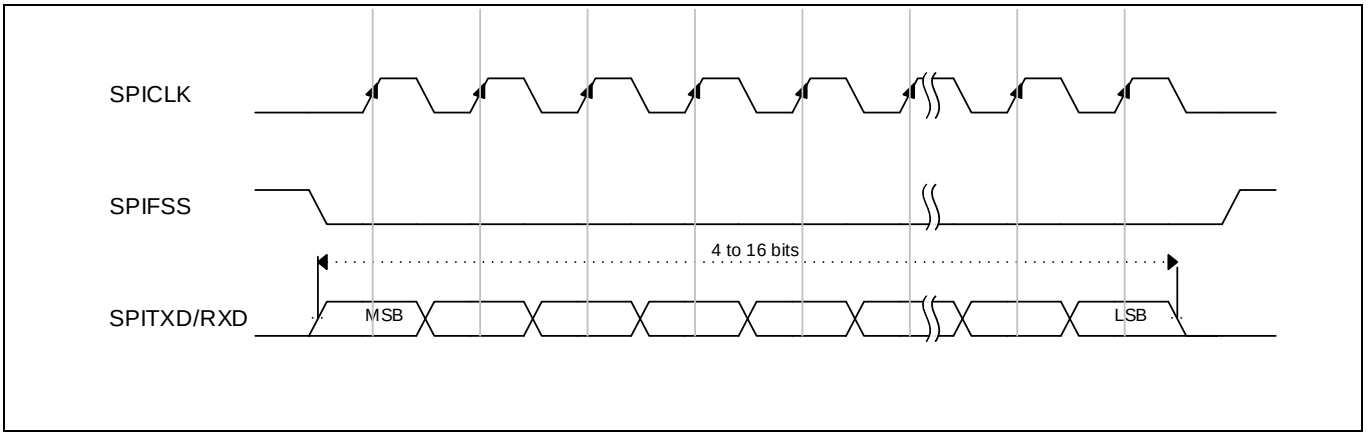


Figure 14-5 Motorola SPI 帧格式 (单帧传输) SPO = 0, SPH = 0

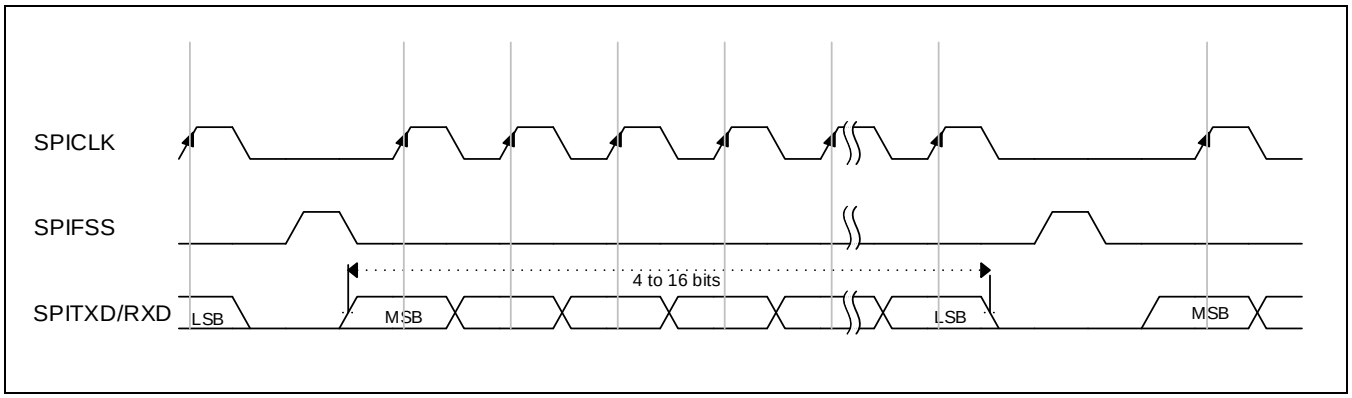


Figure 14-6 Motorola SPI 帧格式 (连续传输) SPO = 0, SPH = 0

这个配置中，在空闲时间：

- SPICLK管脚信号被拉低
- SPIFSS被拉高
- 传输数据线SPITXD被强制拉低
- 当SPI为主机时，SPICLK管脚使能
- 当SPI为从机时，SPICLK管脚禁止

在单字传输的情况下，当所有数据位都传输完后，SPIFSS信号线在最后一位被捕捉后的一个SPICLK周期后回到它的空闲状态 - 高电平。但是在连续的传输中，SPIFSS信号必须在数据传输中间一直保持高电平。这是因为从机选择管脚会冻结从机串行寄存器，并且如果SPH位是0就不允许改变。所以主机必须将SPIFSS管脚拉高，让从机在数据字的传输中间使能串行数据的写操作。在连续传输结束时，SPIFSS管脚在最后一位被捕捉后的一个SPICLK周期后又会回到它的空闲状态。

14.3.3.4.2 Motorola SPI 格式，SPO = 0, SPH = 1

SPO = 0， 无数据传输时，SPICLK 处于低电平（开始阶段）。

SPH = 1，第二个时钟沿是下降沿，所以接收数据捕获即发生于下降沿，发送数据则在下降沿时处于稳定状态。

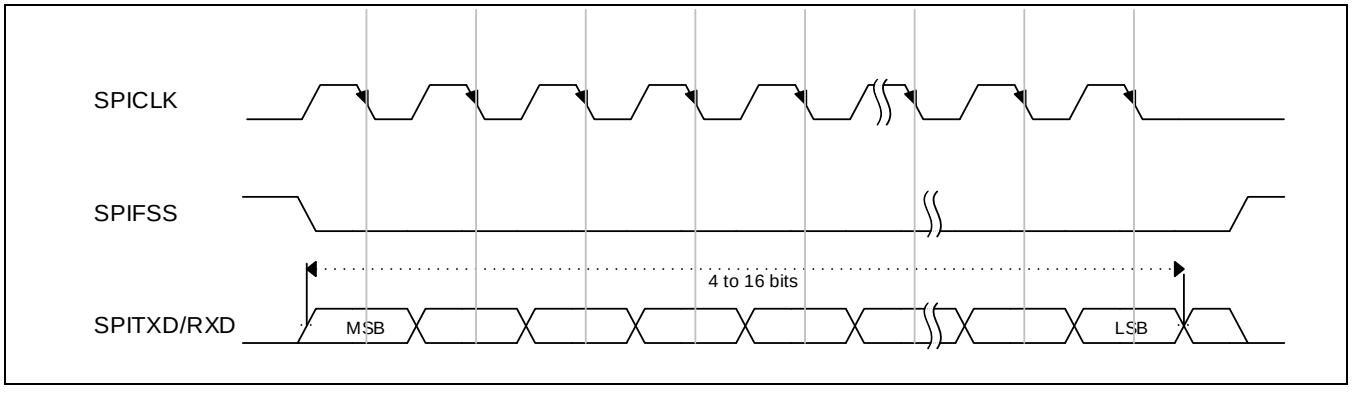


Figure 14-7 Motorola SPI 帧格式（单帧传输）， SPO = 0 和 SPH = 1

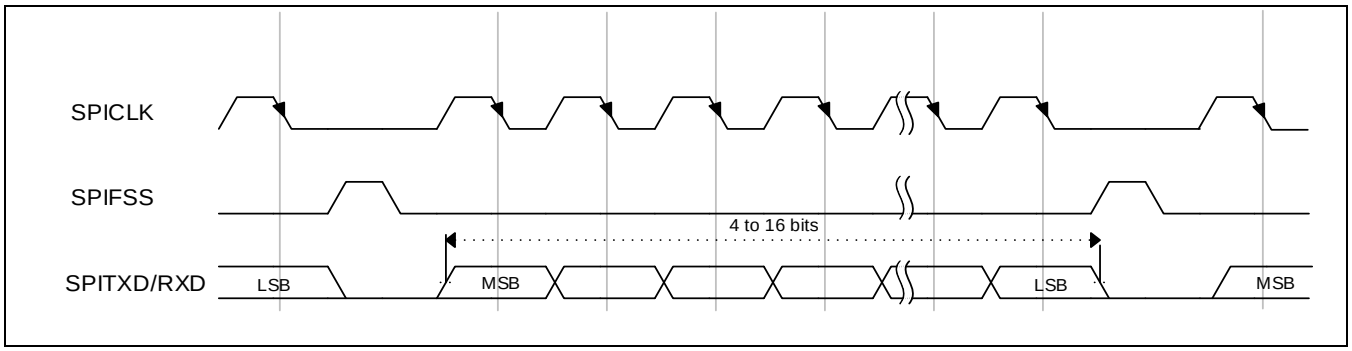


Figure 14-8 Motorola SPI 帧格式（连续传输）， SPO = 0 和 SPH = 1

这个配置中，在空闲时间：

- SPICLK信号被拉低
- SPIFSS被拉高
- 传输数据线SPITXD被强制拉低
- 当SPI为主机时，SPICLK管脚使能
- 当SPI为从机时，SPICLK管脚禁止

在单次传输的情况下，当所有数据位都传输完后，SPIFSS信号线在最后一位被捕捉后的一个SPICLK周期后回到它的空闲状态 - 高电平。在连续的传输中，SPIFSS信号在连续的数据传输中间一直保持低电平，最终结束的时候跟单次传输的情况一样。

14.3.3.4.3 Motorola SPI格式，SPO = 1, SPH = 0

Motorola SPI格式中的 SPO = 1, SPH = 0 信号传输示意图如下：

SPO = 1， 无数据传输时，SPICLK 处于高电平（开始阶段）。

SPH = 0，第一个时钟沿是上升沿，所以接收数据捕获即发生于上升沿，发送数据则在上升沿时处于稳定状态。

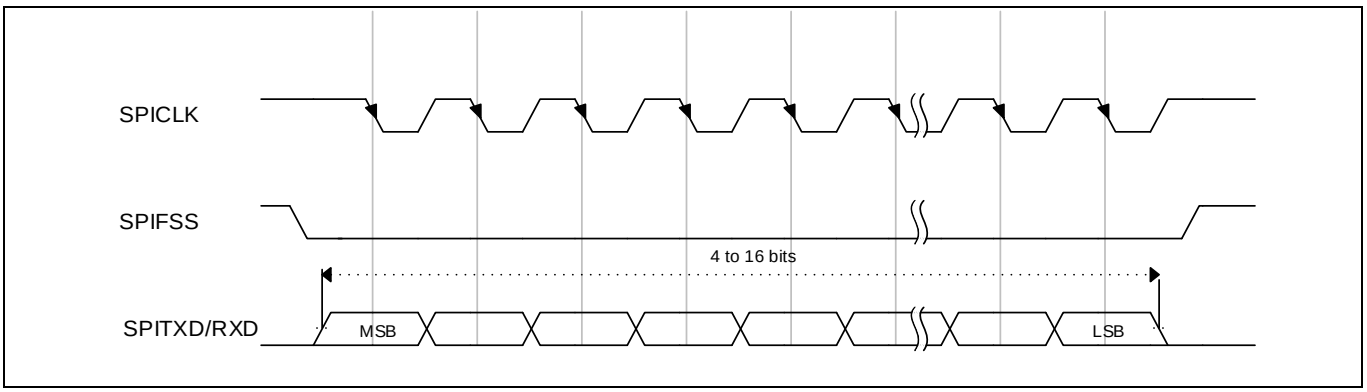


Figure 14-9 Motorola SPI 帧格式 (单帧传输)，SPO = 1 和 SPH = 0

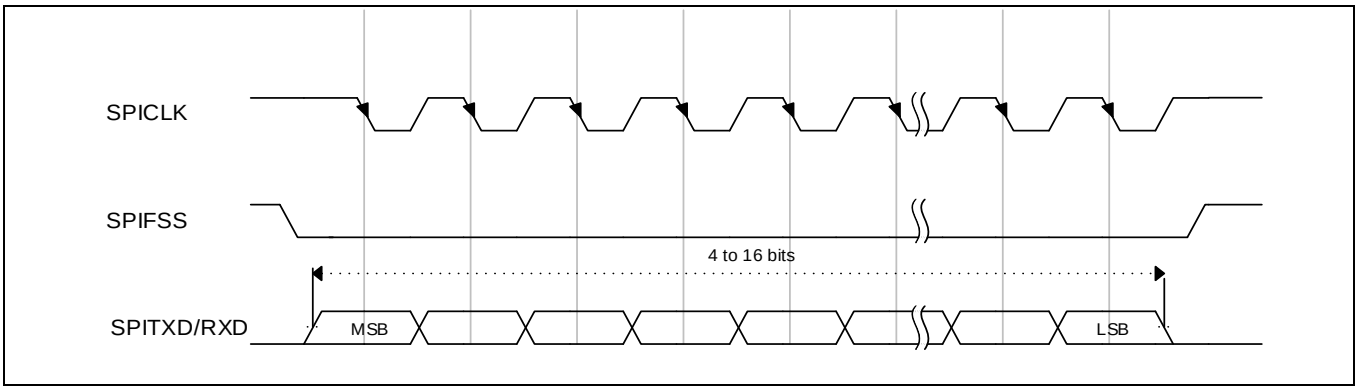


Figure 14-10 Motorola SPI 帧格式 (连续传输)， SPO = 1 和 SPH = 0

这种情况下，在空闲时间：

- SPICLK信号被拉高
- SPIFSS被拉高
- 传输数据线SPITXD被强制拉低
- 当SPI为主机时，SPICLK管脚使能
- 当SPI为从机时，SPICLK管脚禁止

在单次传输的情况下，当所有数据位都传输完后，SPIFSS信号线在最后一位被捕捉后的一个SPICLK周期后回到它的空闲状态 - 高电平。但是在连续的传输中，SPIFSS信号必须在数据传输中间一直保持高电平。这是因为从机选择管脚会冻结从机串行寄存器，并且如果SPH位是0就不允许改变。所以主机必须将SPIFSS管脚拉高，让从机在数据字的传输中间使能串行数据的写操作。在连续传输结束时，SPIFSS管脚在最后一位被捕捉后的一个SPICLK周期后又会回到它的空闲状态。

14.3.3.4.4 Motorola SPI 格式， SPO = 1, SPH = 1

Motorola SPI 格式中SPO = 1, SPH = 1的传输示意图如下：

SPO = 1， 无数据传输时，SPICLK 处于高电平（开始阶段）。

SPH = 1，第二个时钟沿是下降沿，所以接收数据捕获即发生于下降沿，发送数据则在下降沿时处于稳定状态。

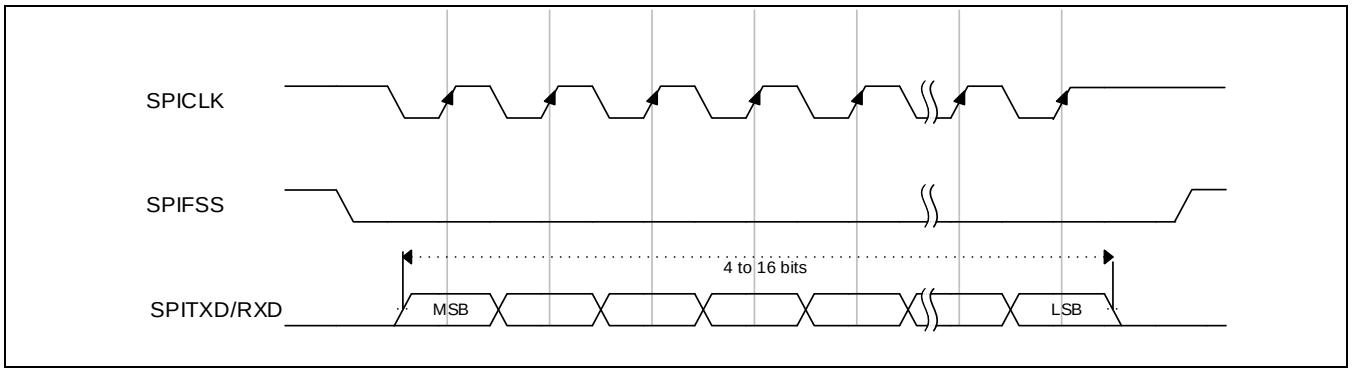


Figure 14-11 Motorola SPI 帧格式(单帧传输)， SPO = 1 和 SPH = 1

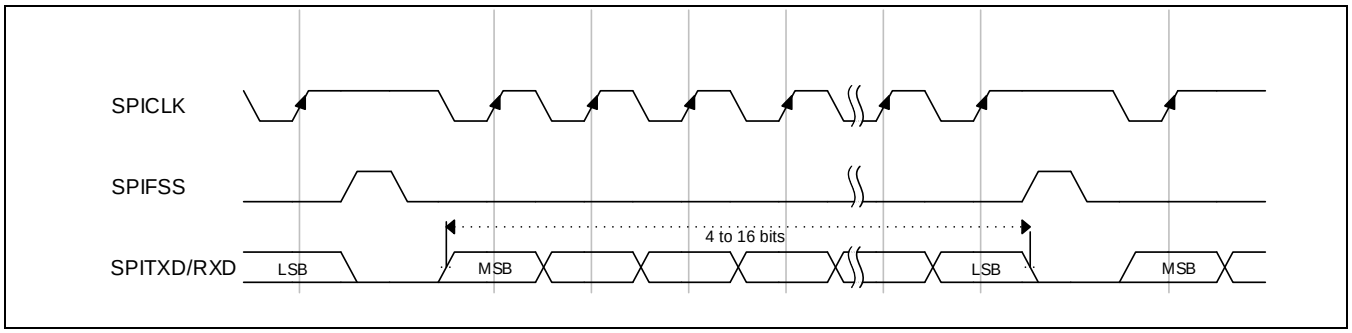


Figure 14-12 Motorola SPI 帧格式(连续传输)， SPO = 1 和 SPH = 1

这种情况下，在空闲时间：

- SPICLK信号被拉高
- SPIFSS被拉高
- 传输数据线SPITXD被强制拉低
- 当SPI为主机时，SPICLK管脚使能
- 当SPI为从机时，SPICLK管脚禁止

在单次传输的情况下，当所有数据位都传输完后，SPIFSS信号线在最后一位被捕捉后的一个SPICLK周期后回到它的空闲状态 - 高电平。在连续的传输中，SPIFSS信号在连续的数据传输中间一直保持低电平，最终结束的时候跟单次传输的情况一样。

14.3.3.5 使能SPI传送

要启动SPI的发送接收，可以往发送FIFO里写1个数据，或者允许发送FIFO给CPU产生一个中断。一旦SPI被使能，数据的发送或者接收就会立即在SPITXD和SPIRXD管脚上启动

14.4 寄存器说明

14.4.1 寄存器表

Base Address of SPI: 0x40090000

Register	Offset	Description	Reset Value
SPI_CR0	0x0000	SPI控制寄存器0	0x00000000
SPI_CR1	0x0004	SPI控制寄存器1	0x00000000
SPI_DR	0x0008	SPI接收FIFO数据寄存器 (读该寄存器时) SPI发送FIFO数据寄存器 (写该寄存器时)	0x00000000
SPI_SR	0x000C	SPI状态寄存器	0x00000003
SPI_PSCR	0x0010	SPI时钟分频寄存器	0x00000000
SPI_IMCR	0x0014	SPI中断使能/禁止寄存器	0x00000000
SPI_RISR	0x0018	SPI中断原始状态寄存器	0x00000008
SPI_MISR	0x001C	SPI中断状态寄存器	0x00000000
SPI_ICR	0x0020	SPI中断清除寄存器	0x00000000
SPI_SRR	0x0028	SSP软件复位寄存器	0x00000000

14.4.2 SPI_CR0(SPI控制寄存器0)

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																SDIV								SPH	SPO	FRF		DATALEN			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
SDIV	[15:8]	RW	串行时钟分频位 SCR用来产生发送和接收的比特率 比特率 = PCLK/ (CDIV * (1 + SDIV)) CDIV为2到254之间的偶数，在SPI_CPSR寄存器设置，SDIV为0到255之间任意值。
SPH	[7]	RW	SPICLKOUT相位 0 = 数据在第一个时钟沿捕捉 1 = 数据在第二个时钟沿捕捉
SPO	[6]	RW	SPICLK极性选择 0 = 没有数据传送时，SPICLK管脚的稳定状态为低电平 1 = 没有数据传送时，SPICLK管脚的稳定状态为高电平
FRF	[5:4]	RW	帧格式选择位 Motorola SPI格式必须设置为00
DATALEN	[3:0]	RW	数据大小选择位 0000‐0010 = 保留 0011 = 4位数据 0100 = 5位数据 0101 = 6位数据 0110 = 7位数据 0111 = 8位数据 1000 = 9位数据 1001 = 10位数据 1010 = 11位数据 1011 = 12位数据 1100 = 13位数据 1101 = 14位数据 1110 = 15位数据 1111 = 16位数据

14.4.3 SPI_CR1(SPI控制寄存器1)

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																							LPTXOE	LPMD	RXIFL			SOD	MODE	SPIEN	LBM
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
LPTXOE	[8]	RW	主机单线模式下，数据发送使能控制 0 = 禁止数据发送 1 = 使能数据发送
LPMD	[7]	RW	0 = SPI普通模式 1 = 主机单线模式
RXIFL	[6:4]	RW	接收FIFO中断触发点选择位 001：接收FIFO非空 010：接收FIFO超过一半 Data Len 为 4 - 8bit时，SPI FIFO 宽度为8bit，深度为4。此时接收到2个数据即触发中断。 Data Len 为 9 - 16bit时，SPI FIFO宽度为16bit，深度为2；此时，接收到1个数据即触发中断。 Others = 保留
SOD	[3]	RW	从机模式输出禁止位 该位只有在从机模式(MS=1)下有效。在多从机系统里，SPI主机可以向系统里的所有从机广播，但是必须保证只有一个从机能够输出数据。在这样的系统里，多从机的RXD必须短接在一起，所以当SPI从机不应该输出数据驱动SPITXD的时候，SOD位必须置1。 0 = SPI在从机模式可以驱动SPITXD输出 1 = SPI在从机模式不驱动SPITXD输出
MODE	[2]	RW	主机或者从机模式 0 = 配置为主机 1 = 配置为从机
SPIEN	[1]	RW	SPI使能位 0 = SPI禁止 1 = SPI使能
LBM	[0]	RW	回送模式位 0 = 正常串行输出操作 1 = 发送移位寄存器的输出内部短接到接收串行移位寄存器

14.4.4 SPI_DR(SPI接收FIFO数据寄存器 (读该寄存器时)S发送FIFO数据寄存器 (写该寄存器时))

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																DATA															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
DATA	[15:0]	RW	

发送/接收FIFO

读：接收FIFO

写：发送FIFO

当数据位小于16的时候，必须右对齐，不用的高位无效。接收逻辑为自动右对齐。

读SPI_DR时，接收FIFO的数据(当前FIFO读指针所指的值)被读取。当SPI接收逻辑把接收到的数据移除时，该数据会被存到接收FIFO中当前读指针所指的空间。

写SPI_DR时，数据被写入发送FIFO中当前指针所指的空间。发送逻辑每发送一次就将发送FIFO中的数据移除一个，移除的数据载入到发送串行移位寄存器，以配置好的比特率串行发送到SPITXD管脚。

当数据位小于或者等于8位时，相当于有8bits宽的四级深FIFO。大于8位时，相当于16bits宽，2级深FIFO。 接收到的数据如果小于16位，在接收缓冲区内也是右对齐的。

14.4.5 SPI_SR(SPI状态寄存器)

Address = Base Address+ 0x000C, 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																											BSY	RFF	RNF	TNF	TFE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
BSY	[4]	R	SPI工作状态标志位 0 = SPI空闲 1 = SPI正在发送并且/或者正在接收，或者发送FIFO非空
RFF	[3]	R	接收FIFO是否已满状态位 0 = 接收FIFO未满 1 = 接收FIFO已满
RNF	[2]	R	接收FIFO是否为空状态位 0 = 接收FIFO为空 1 = 接收FIFO非空
TNF	[1]	R	发送FIFO是否已满状态位 0 = 发送FIFO已满 1 = 发送FIFO未满
TFE	[0]	R	发送FIFO是否为空状态位 0 = 发送FIFO非空 1 = 发送FIFO空

14.4.6 SPI_PSCR(SPI时钟分频寄存器)

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																								CDIV							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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
CDIV	[7:0]	RW	时钟分频位 用来设置FPCLK 的分频系数，供下一分频器使用。寄存器的值必须是2到254之间的偶数。寄存器的最低位被硬件强制为0，即使将一个奇数写入该寄存器，读出来值的最低位也为0。

14.4.7 SPI_IMCR(SPI中断使能/禁止寄存器)

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
RSVD																												TX	RX	RTO	ROVF
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	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
TX	[3]	RW	发送FIFO中断 0 = 禁止发送FIFO中断 1 = 使能发送FIFO中断
RX	[2]	RW	接收FIFO中断 0 = 禁止接收FIFO中断 1 = 使能接收FIFO中断
RTO	[1]	RW	接收超时中断 0 = 禁止RxFIFO超时中断 1 = 使能RxFIFO超时中断
ROVF	[0]	RW	接收溢出中断 0 = 禁止RxFIFO溢出中断 1 = 使能RxFIFO溢出中断
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。			

14.4.8 SPI_RISR(SPI中断原始状态寄存器)

Address = Base Address+ 0x0018, Reset Value = 0x00000008

31	30	29	28	27	26	25	24	23	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	RX	RTO	ROVF
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
TX	[3]	R	发送FIFO中断请求原始标志状态
RX	[2]	R	接收FIFO中断请求原始标志状态 说明：读取SPI[DR]时，该位自动清除
RTO	[1]	R	接收超时中断请求原始标志状态
ROVF	[0]	R	接收中断中断请求原始标志状态
读该寄存器返回相应中断的原始状态，不管该中断是否被使能。写寄存器无效。			

14.4.9 SPI_MISR(SPI中断状态寄存器)

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
RSVD																												TX	RX	RTO	ROVF
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
TX	[3]	R	发送FIFO中断请求标志状态
RX	[2]	R	接收FIFO中断请求标志状态 说明：读取SPI[DR]时，该位自动清除
RTO	[1]	R	接收超时中断请求标志状态
ROVF	[0]	R	接收溢出中断请求标志状态
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			

14.4.10 SPI_ICR(SPI中断清除寄存器)

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
RSVD																														RTO	ROVF
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W

Name	Bit	Type	Description
RTO	[1]	W	接收超时中断清除 0 = 无效 1 = 清除中断
ROVF	[0]	W	接收溢出中断清除 0 = 无效 1 = 清除中断
写1清除该中断，写0无效。			

14.4.11 SPI_SRR(SSP软件复位寄存器)

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
RSVD																						TXFIFO_RST	RXFIFO_RST	RSVD						SCLK_ SWRST	PCLK_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	W	W	R	R	R	R	R	R	W	W

Name	Bit	Type	Description
TXFIFO_RST	[9]	W	发送FIFO复位 0 = 无效 1 = 复位
RXFIFO_RST	[8]	W	接收FIFO复位 0 = 无效 1 = 复位
SCLK_ SWRST	[1]	W	SPICLK时钟域软件复位 0 = 无效 1 = 软件复位
PCLK_SWRST	[0]	W	PCLK时钟域软件复位 0 = 无效 1 = 软件复位

15 通用异步收发器 (UART)

15.1 概述

UART是一个简单通用的异步串行接收和发送接口，支持7/8位的数据通信。

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

- 可配置的波特率
- 支持2种数据位数，7/8位
- 发送接收溢出检测
- 发送接收完成中断和溢出中断
- 支持自动波特率

15.1.2 管脚描述

Table 15-1 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	–	–

15.2 功能描述

15.2.1 模块框图

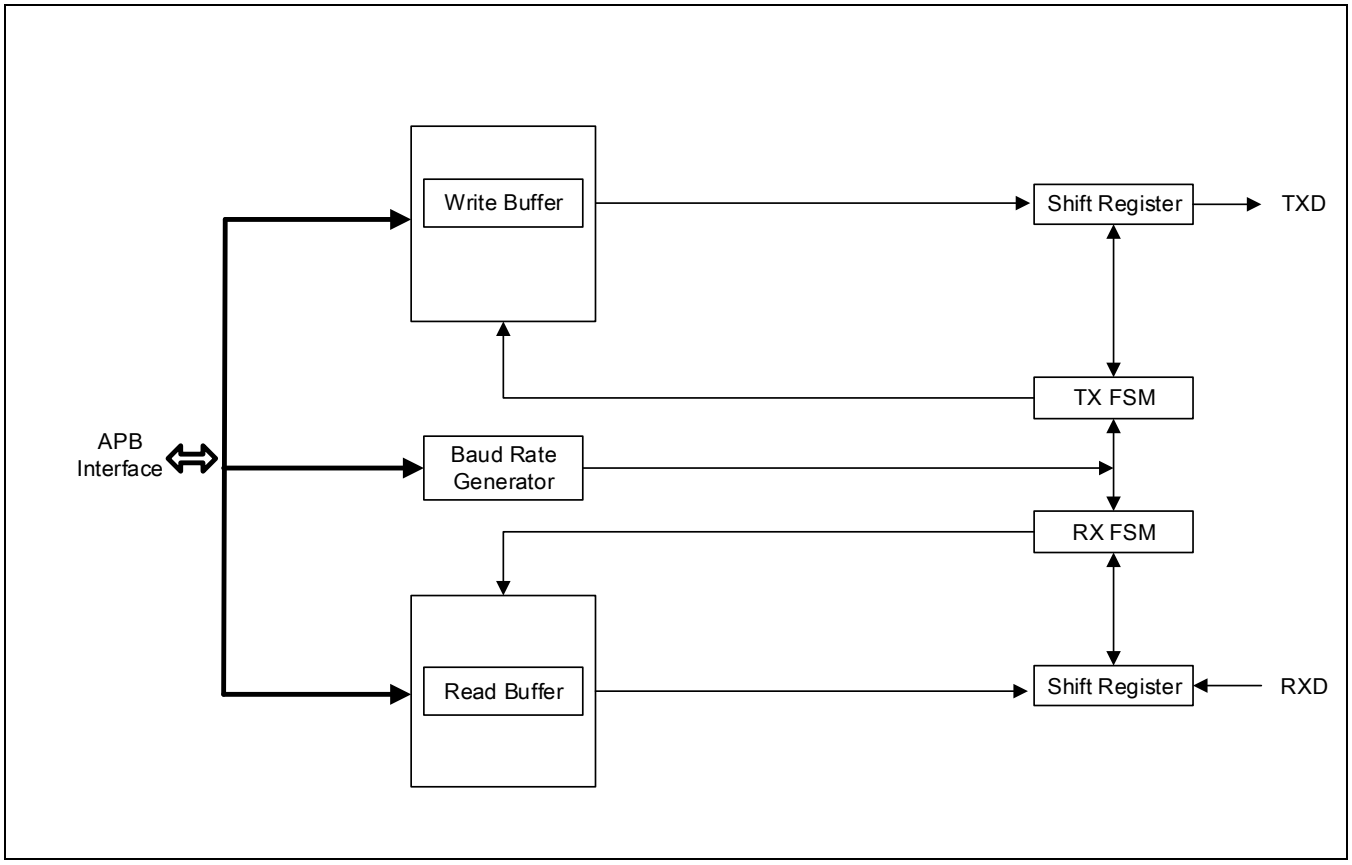


Figure 15-1 UART模块框图

15.2.2 功能说明

15.2.2.1 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 15-2 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%

15.2.2.2 自动波特率

UART支持自动波特率功能，可以根据发送方的波特率，修改UART的波特率。

要使用自动波特率，UART配置如下：通信数据长度配置为8位，无校验。

寄存器UART_CTRL[28] 写入1，使能自动波特率，每次自动波特完成之后，硬件会自动清除该使能位。故每次使用自动波特率之前，都需要使能自动波特率。

自动波特率的实现，要求在正常UART通信前，发送方发送特定字符“0x7f”，字符长度8bit。波特率检测模块计算起始位的上升沿和停止位的上升沿时间差，即可计算出发送端的波特率,并将波特率的分频写入BRDIV寄存器中。

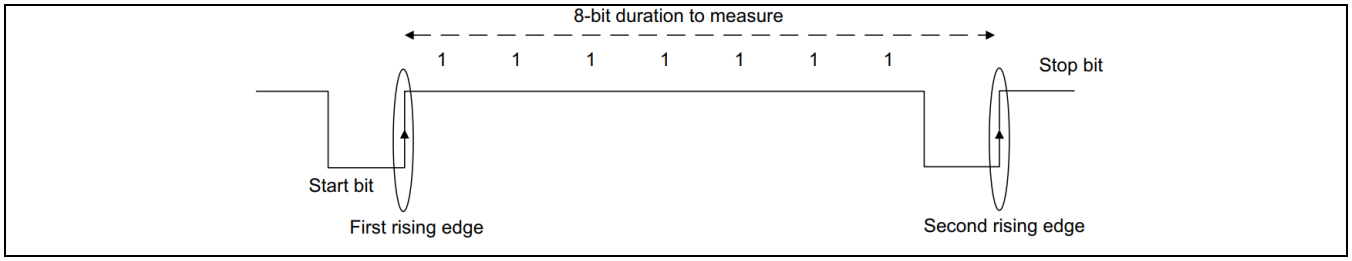


Figure 15-2 自动波特率原理

15.2.2.3 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)。

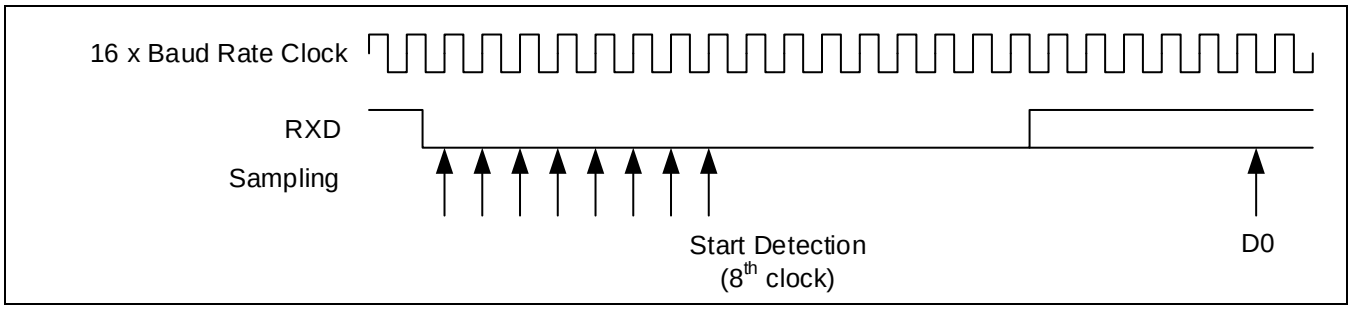


Figure 15-3 起始位检测

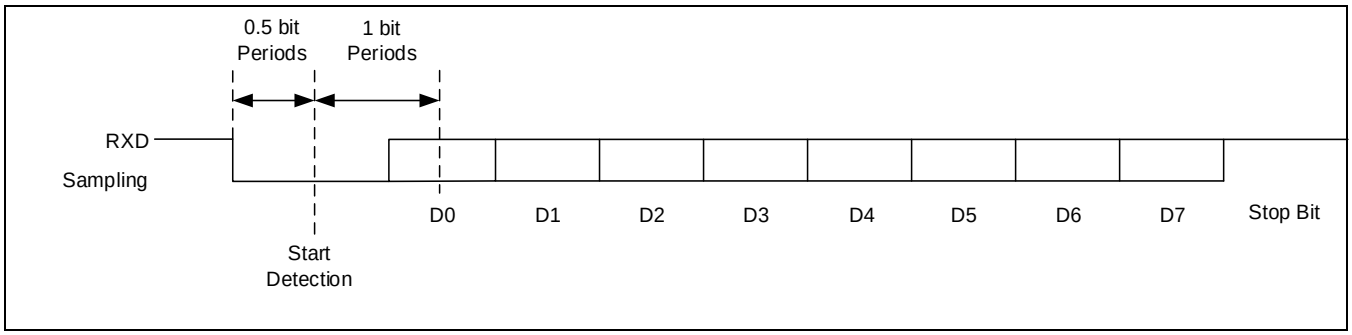


Figure 15-4 接收数据

15.2.2.4 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。

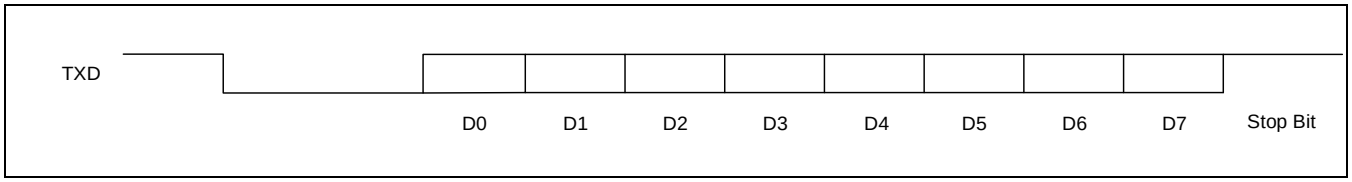


Figure 15-5 数据发送

15.2.2.5 Break 信号

Break信号是一个逻辑低电平 ，持续时间至少为一帧的长度（开始位+数据位+校验位+停止位）。任何一个有效数据帧都不会呈现这样的低电平特性，所以是一个特殊的信号，可以当做断开信号。

15.2.2.5.1 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。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。

15.2.2.5.2 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。在检测到低电平停止位的时刻，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.

15.2.2.6 数据位

UART_CTRL寄存器中的DATA位用来选择数据长度。

- DATA[1:0]为00：数据长度为7位

- DATA[1:0]为01：数据长度为8位

15.2.2.7 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。

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。

15.3 寄存器说明

15.3.1 寄存器表

Base Address of UART0: 0x40080000
Base Address of UART1: 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

15.3.2 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

15.3.3 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																				TX_BSY	RXBRK	TIMEOUT	RSVD					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	R	RW	RW	R	R

Name	Bit	Type	Description
TX_BSY	[11]	R	发送数据端忙 0 = 没有数据发送 1 = 正在进行数据发送
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Break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）
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)

15.3.4 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		ARB_EN	CHRL		RSVD	STPBRK	STTBRK	INT_RXBRK	INT_RXTO	STTT0	INT_TX_DONE_EN	RSVD												TEST	INT_OVER_RX	INT_OVER_TX	INT_RX	INT_TX	RX	TX
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	R	R	W	RW	RW	R	W	W	RW	RW	W	W	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
DBGEN	[31]	W	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
ARB_EN	[28]	W	自动波特率使能
CHRL	[27:26]	RW	数据位长度 00 = 7位 01 = 8位 其他保留
STPBRK	[24]	W	停止Break 0= 无效 1= 如果一个Break状态位正在发送，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完数据之后，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
STTT0	[20]	W	开启超时接收 0= 不开启 1= 开启
INT_TX_DONE_EN	[19]	W	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
TEST	[6]	RW	测试模式

			此为请保持为0
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

15.3.5 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						ABR_ERROR	ABR_DONE	UART_RXBRK_INT_S	UART_RXTO_INT_S	RSVD					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	RW	R	R	R	R	R	R	RW	RW	RW	R	R	R	R	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
TX_DONE_INT	[19]	RW	发送完成中断 0= 发送完成中断没发生 1= 发送完成中断发生(读取) 1= 清除发送完成中断(写)
ABR_ERROR	[12]	RW	ABR失败中断 0= ABR失败中断没发生 1= ABR失败中断发生(读取) 1= 清除ABR失败中断(写)
ABR_DONE	[11]	RW	ABR完成中断 0= ABR中断没发生 1= ABR中断发生(读取) 1= 清除ABR中断(写)
UART_RXBRK_INT_S	[10]	RW	接收端Break 0= 没有产生Break中断 1= 产生了Break中断 1= 清除Break中断(写)
UART_RXTO_INT_S	[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)
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)

15.3.6 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

15.3.7 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

15.3.8 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"。

15.3.9 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

16 模数转换器 (ADC)

16.1 概述

本章节描述ADC控制器的功能，从用户的角度详细说明如何操作ADC。

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

16.1.1 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[25:0]，内部固定电压参考源输入，以及1/5VDD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

16.1.2 管脚描述

Table 16-1 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 AIN25	模拟信号输入	模拟	-	-

16.1.3 模块框图

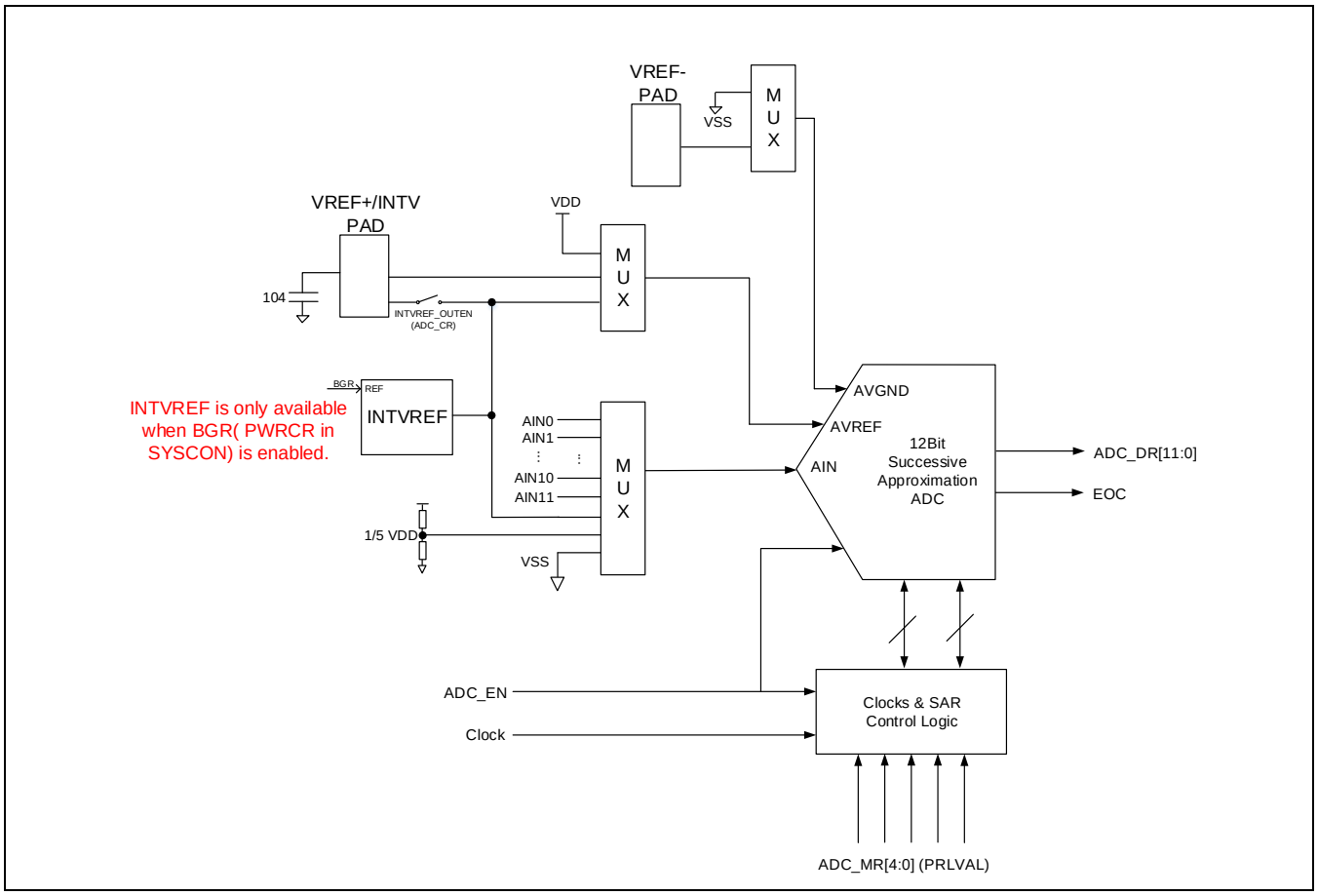


Figure 16-1 ADC模块框图

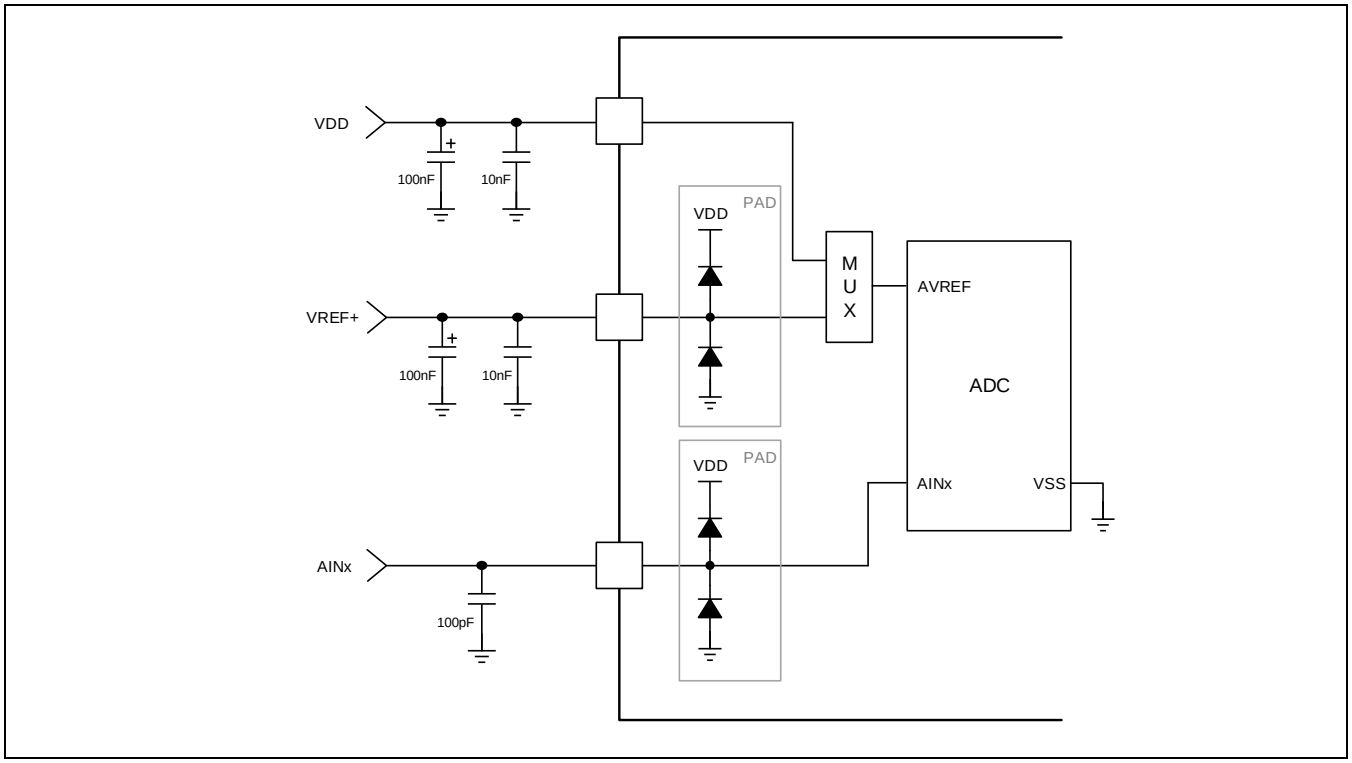


Figure 16-2 参考电路

16.1.4 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 16-2 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

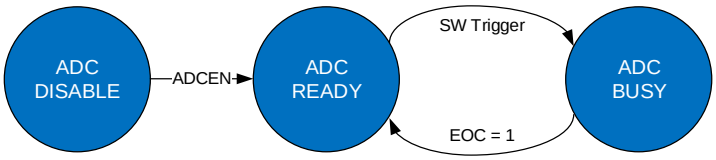


Figure 16-3 ADC状态机

16.1.5 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，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。

16.1.6 时钟频率和转换时间

ADC 工作的时钟是从 PCLK 获得的。AD 转换的过程需要总共(S/H+12)个时钟周期。S/H(Sample&Hold 采样保持)时间可以通过 ADC_SHR 寄存器设置。默认的采样保持时间 (6 个周期) 可以满足大部分应用场景的需求，在某些特殊应用场景中，如果需要更长的采样和保持时间，可以通过特殊的 ADC_SHR 寄存器来实现。

ADC 模块提供一个时钟分频器，该分频器是一个 6 位计数器，由模式寄存器里的 PRLVAL 控制。 下面的表达式给出了系统频率和 ADC 模拟模块时钟频率之间的关系。

如果 PRLVAL 是 0，那么 F_ANA = PCLK

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

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

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

22 x 100ns = 2.2us (454ksps)

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

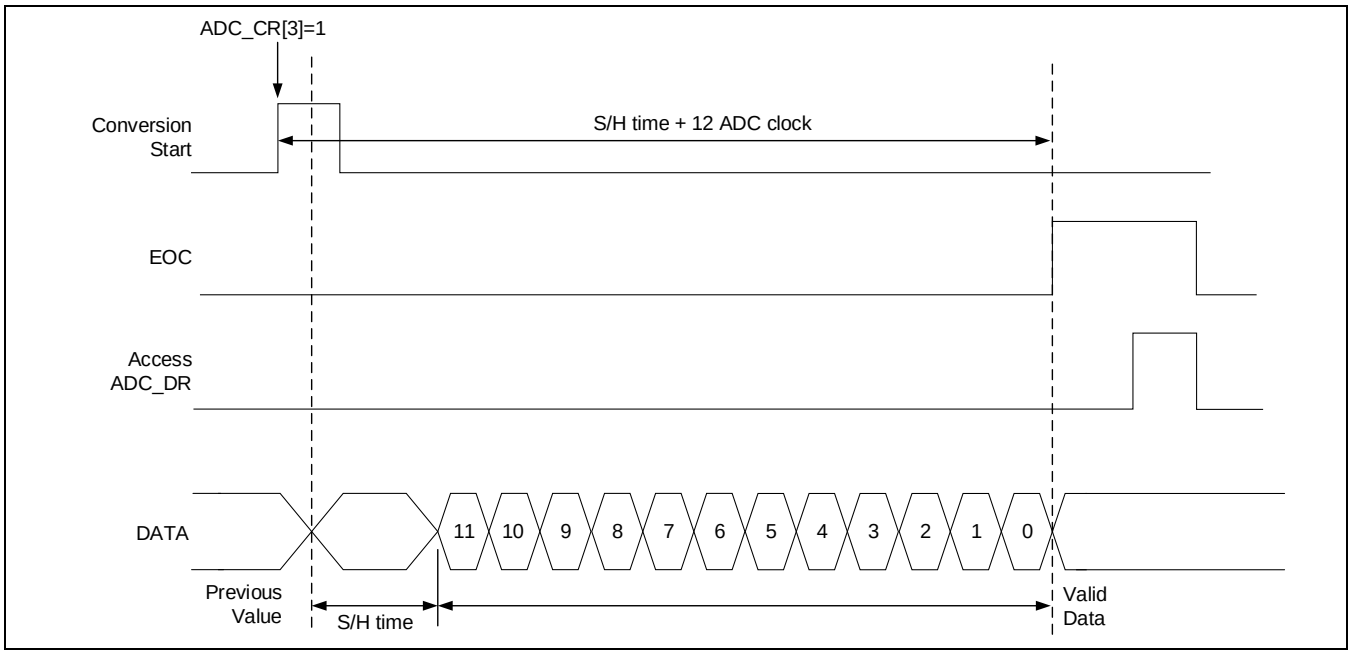


Figure 16-4 ADC工作时序图

16.1.7 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，包括连续转换序列和高优先级转换序列。连续转换序列个数是由最大转换序列位MR[NBRCH]和转换序列优先选择位PRI[PRI]共同决定的，优先选择位PRI[PRI]（具体功能可参见Figure 16-5触发优先级示意图）定义连续转换序列的最小序列，MR[NBRCH]定义连续转换序列的最大序列。

ADC_SEQ0~ADC_SEQ7这8个寄存器用来配置每个转换序列的输入通道，触发源等参数。PRI保持默认值0的情况下，如果设置最大转换序列号为7，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_SEQ7设置的通道，并将转换结果存在ADC_DR0, ADC_DR1, …, ADC_DR7这8个转换结果寄存器中。如果设置最大转换序列号为0，那么ADC只会转换ADC_SEQ0设置的通道，并且将结果存入ADC_DR0。同理如果最大转换序列号为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和最大转换序列的关系：

Table 16-3 NBRCH[3:0]的值和最大转换序列

NBRCH[3:0]	最大转换序列
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6

0111	7

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。8 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 16-4 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
…	…	…
1_1001	Input 25	AIN25
…	No Input (input floating)	N/A
1_1100	Input 28	INTVREF
1_1101	Input 29	1/5 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。

16.1.8 转换模式

ADC 可以配置成三种模式：单次转换模式、连续转换模式或者等待转换模式。

16.1.8.1 单次转换模式

将模式寄存器中的 MODE 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。

16.1.8.2 连续转换模式

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

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

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

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

16.1.8.3 等待转换模式

将 ADC_MR[MODE]位设 2 或 3 为等待转换模式。在等待转换模式下，ADC 转换和触发源紧密相关。当某个触发 事件到来时，只有将触发源设置为该事件的通道会按照通道号顺序进行转换，通道号越小优先级越高，转换的结果存放在相应的转换结果寄存器（ADC_DRx）中。最后一个通道转换完成后，进入等待，直到有新的触发源启动与其相应的新的 ADC_SEQx 转换。

在等待转换模式下，ADC_SEQx[TRG_SRC] 为零的通道将永远不会参与转换。

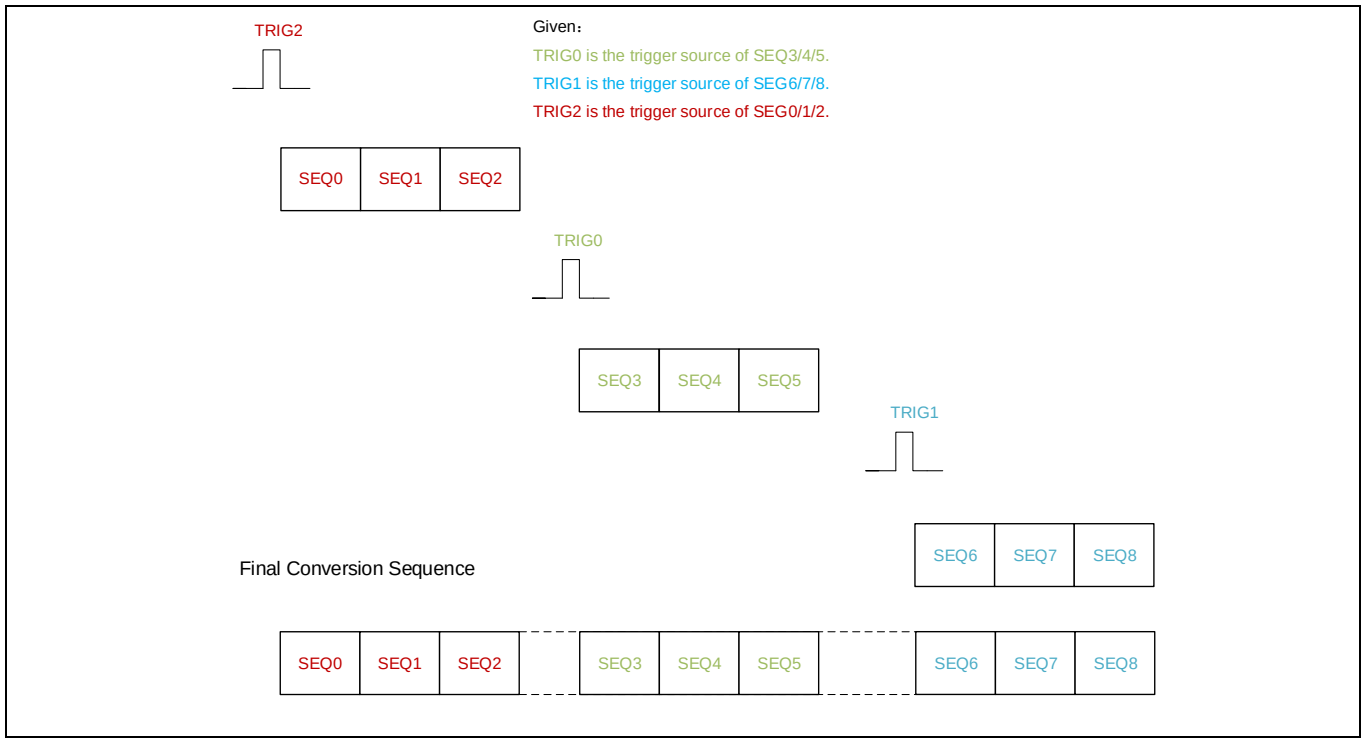


Figure 16-5 等待模式 ADC 工作时序图

如果当前触发源触发的多个 ADC_SEQx 还未转换完时，又有新的触发源信号到来，那在当前通道转换结束后，根据通道优先级，重新对未转换通道和新通道的优先级进行排列并进行转换。

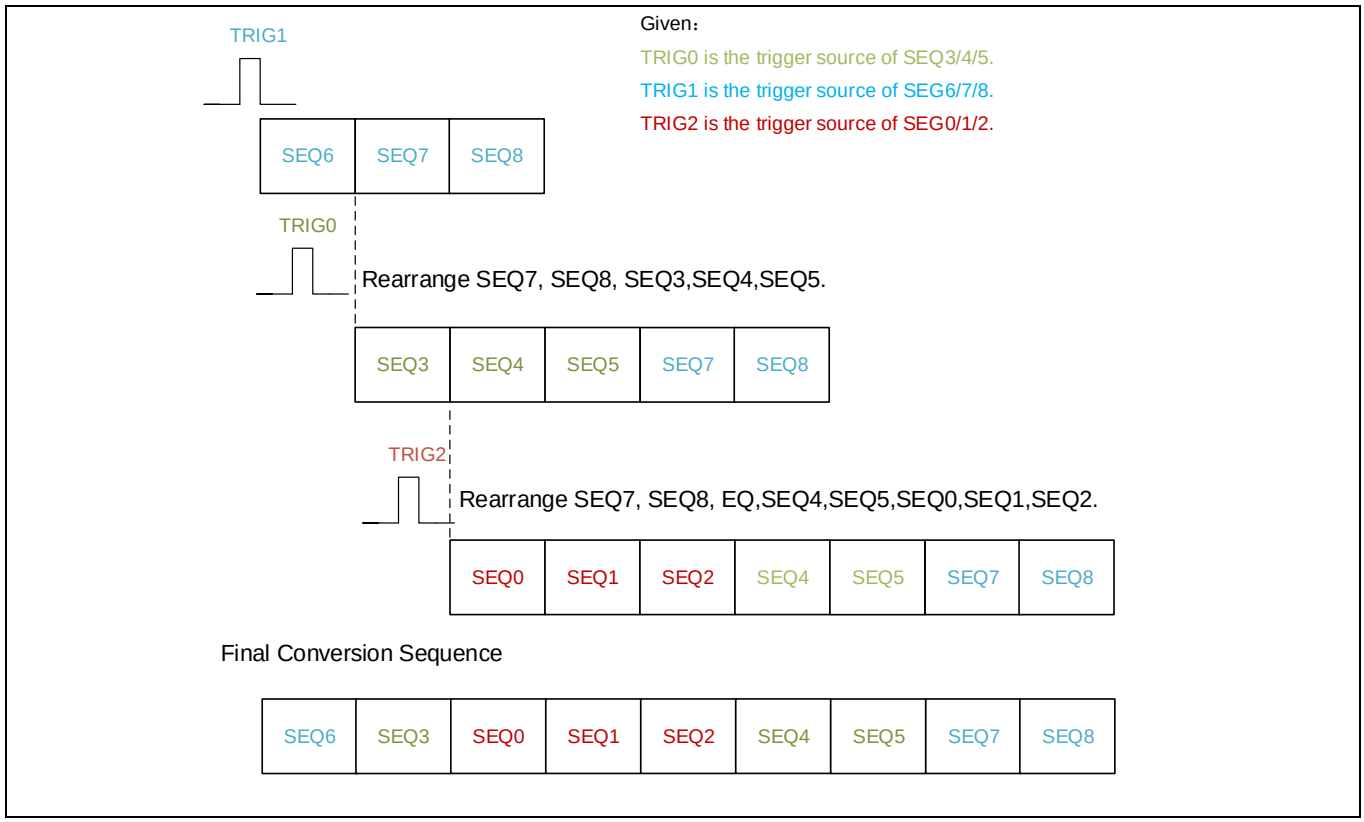


Figure 16-6 等待模式 ADC 工作时序图

16.1.9 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 8 位，对应于 8 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。

16.1.10 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 16-5 TRG_SRC[2:0]的值和选择的触发源

TRG_SRC[2:0]	触发源
000	无触发
001	软件触发(ADC_CR中的SWTRG位)
010	ADC_SYNCIN0触发源 (参考ETCB章节)
011	ADC_SYNCIN1触发源 (参考ETCB章节)

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

ADC在使能触发(ADC_SEQx中TRG_SRC不为0，ADC_SYNCR中相应的SYNCEN使能位为1)并且接收到该触发源后，会立即按预设的配置开始进行转换，也就是触发源和ADC_CR寄存器的开始转换位(START)功能一致。

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

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

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

如下图所示，绿色SEQ0为正在转换的序列0，按照正常转换顺序，SEQ1为下一个需要转换的序列。在SEQ0转换的过程中，SEQ5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SEQ5。当SEQ0转换完成后，SEQ5将继续开始转换，并且SEQ6将变为SEQ5的下一个转换序列。可以看到，因为SEQ5被触发转换，所以SEQ1以及后续的SEQ2 ~ SEQ4都被跳过了。

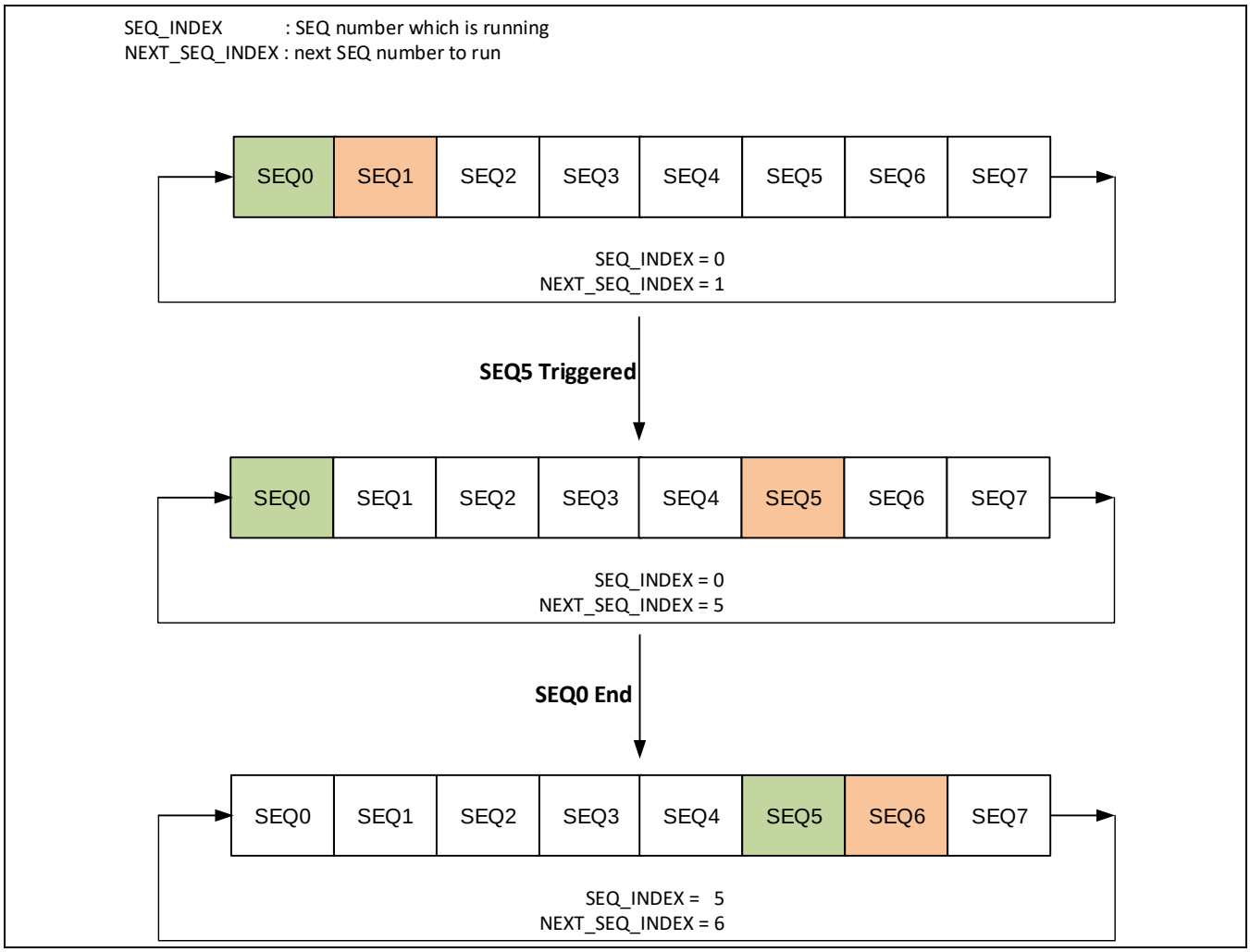


Figure 16-6 触发原理示意图

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SEQ5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SEQ5，接着再按照顺序继续执行下去。

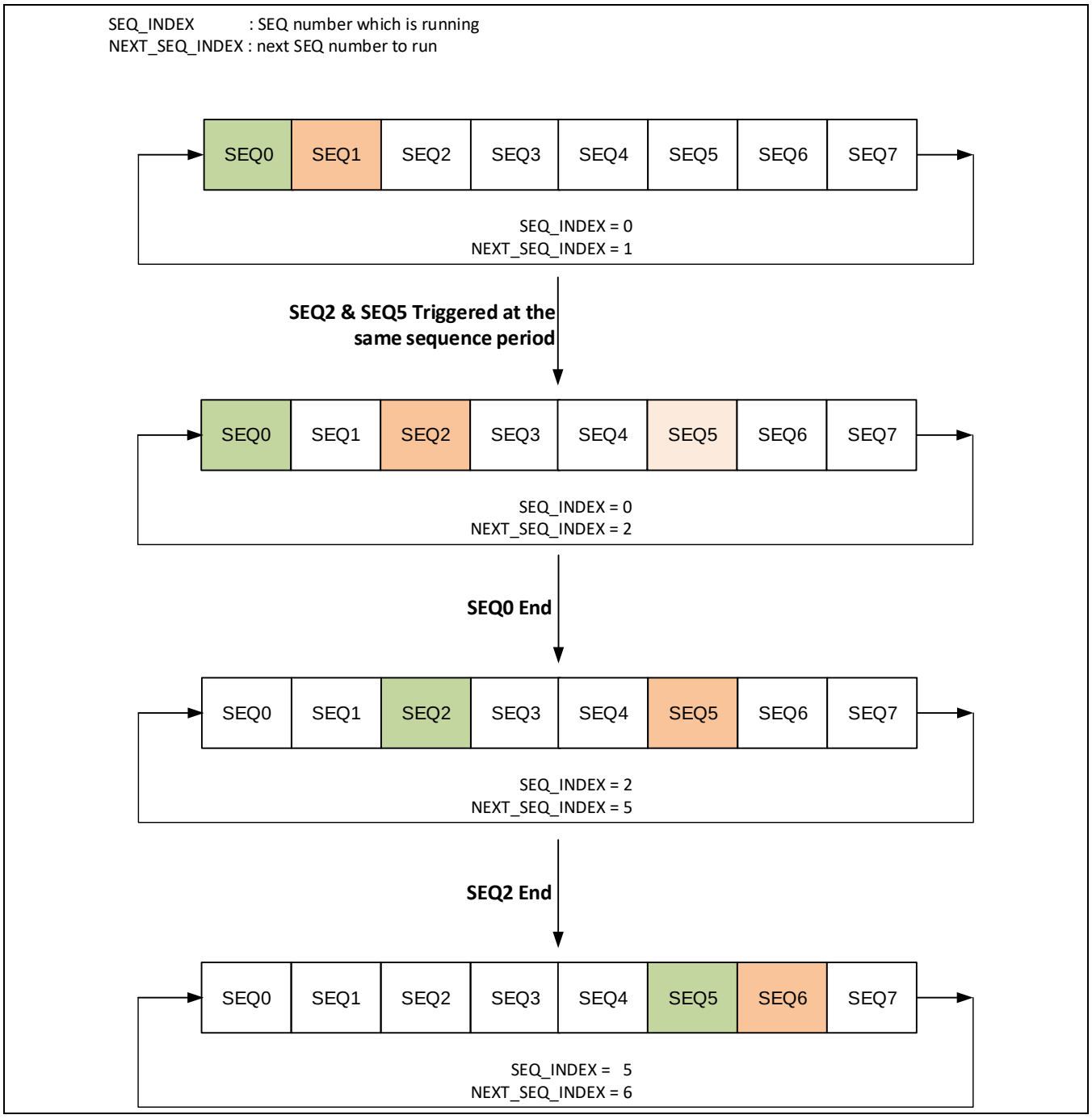


Figure 16-7 同时触发示意图

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SEQ6同时发生，那么SEQ2会先被转换，之后再转换SEQ6，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。

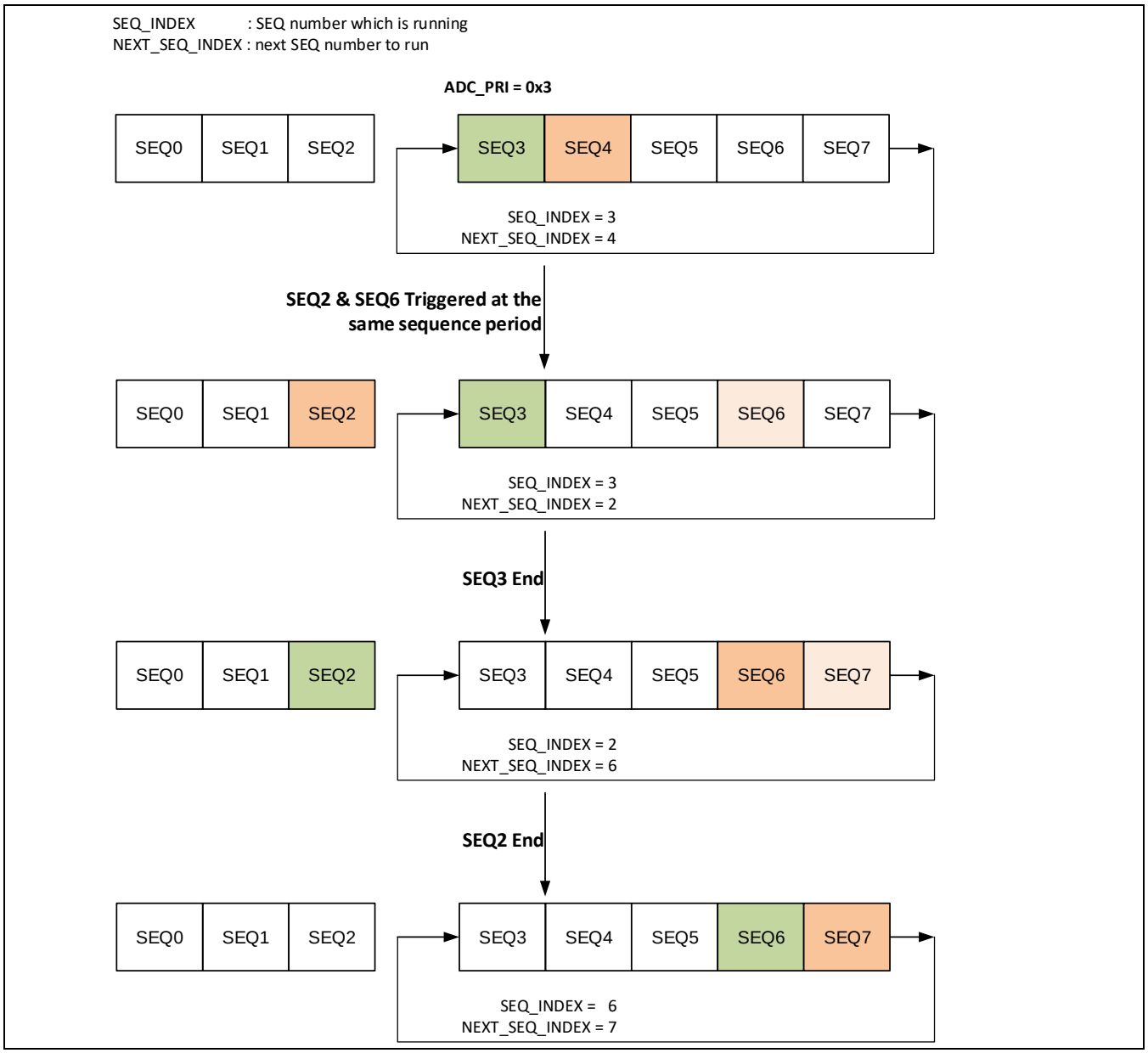


Figure 16-8 触发优先级示意图

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, SEQ6, SEQ7，BT触发SEQ5，那么当PWM和BT触发同时发生时，BT的SEQ5会抢在SEQ6和SEQ7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，BT触

发SEQ7，这样当PWM和BT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。

16.1.11 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 16-6 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

16.1.12 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。

16.1.13 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。

16.1.14 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)

16.1.15 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)。

16.1.16 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)。

16.1.17 ADC事件输出接口 (ETCB接口)

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

16.1.18 工作流程

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

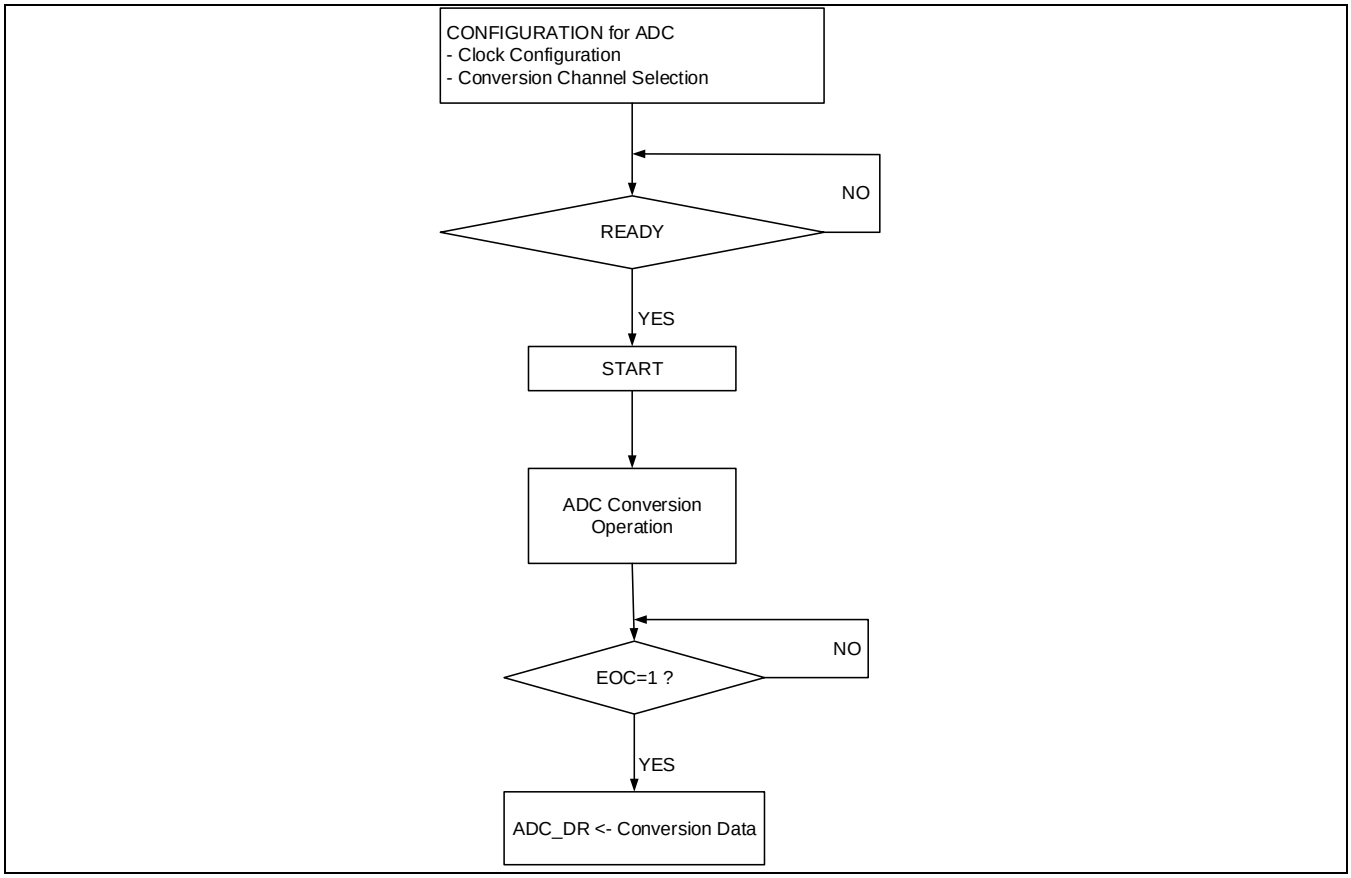


Figure 16-9 ADC工作流程图

16.1.19 转换的软件操作流程

下面描述了在复位后使用 ADC 模块的基本操作流程：

1. 在 ADC_ECR 中使能时钟
2. 在 ADC_MR 中设置 ADC 工作模式。PRLVAL 的值不能让模拟模块的工作时钟频率超过 10MHz。设置单次转换还是连续转换模式。定义转换序列：转换序列个数(NBRCH)和哪些输入通道需要被转换(ADC_SEQx 中的 AIN_SEL)
3. 使能 ADC 模块(ADC_CR 中的 ADC_EN)
4. 等待 ADC_SR 中的 READY 位。只有当这个标志位被置 1 后，ADC 才能正常的开始转换。如果 ADC_IMR 中的相应中断被使能，那么当 READY 标志置起的时候，会产生一个中断
5. 通过写 ADC_CR 中的 START 位，开始转换
6. ADC 选择转换序列中的第一个模拟输入通道
7. 模拟输入电压被采样并且在 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 MODE 是 1，那么 ADC 会从第 5 步重新开始另一个转换序列。

16.1.20 温度传感器

温度传感器是依赖ADC模块实现的。

16.1.20.1 温度传感器的采集输入

如果希望使用内部温度传感器，需要通过SEQx[AIN_SEL]把ADC的任意输入通道AINx配置成TS，选择参考电压为内部参考电压，并且ADC_CR[INTVREF_LVL] 配置成0，即1.2V。

16.1.20.2 温度传感器的采集值到温度的转换原理

该温度传感器的电压值和芯片温度成接近负温度系数的线性关系。

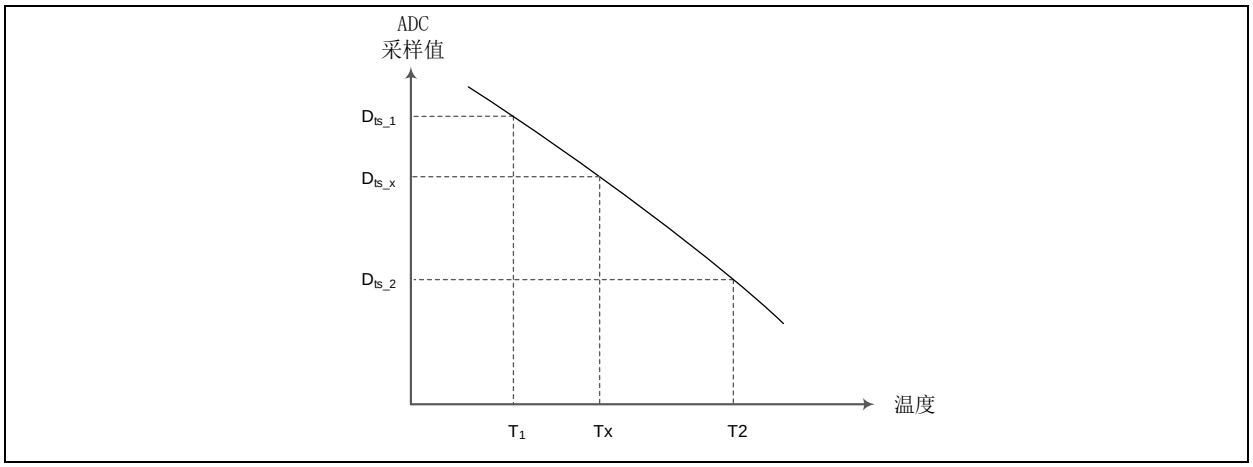


Figure 16-10 ADC采样内部温度传感器

为获得当前温度Tx，需要知道：

1） 传感器温度（℃）与传感器电压（V）的对应线性公式。根据两点确定一条直线的规则，可以通过出厂前在两个温度（T1，T2）下所做的标定值获取D_{ts_1}, D_{ts_2}。

2） TS当前温度的ADC采样值 （D_{ts_x}）。

16.1.20.3 温度的计算方法

出厂标定数据 TS1、TS2均是32-bit 的数据且以相同的格式进行存储。以 TS1为例，TS1[FLAG]是数据有效位，TS1[TEMP]是当前标定值的温度信息（非实际温度℃，若要得到实际温度℃，需要进行相应的换算），TS1[RSVD]是保留位，TS1[DATA]是当前温度传感器的电压采样值。

当前的标定值的温度信息与实际温度（℃）的对应关系如下表所示，规定实际温度℃非负时，TEMP 值以原码存储，实际温度℃为负时，TEMP 值以补码存储。

Table 16-7 温度信息与实际温度℃的对应关系

Temperature（℃）	TEMP（binary）	TEMP（hex）
85	0101 0101 0000	550
80	0101 0000 0000	500
75	0100 1011 0000	4B0
25	0001 1001 0000	190
0.25	0000 0000 0100	004
0	0000 0000 0000	000
–0.25	1111 1111 1100	FFC
–25	1110 0111 0000	E70

-40	1101 1000 0000	D80

Note: 正数的反码和补码都与原码相同。

负数的绝对值转换成二进制位然后在高位补1就是这个负数的原码

负数的反码为对该数的原码除符号位外各位取反。

负数的补码为对该数的原码除符号位外各位取反，然后在最后一位加1

16.1.20.4 实际温度（℃）的计算方法

为了计算时方便区别各标定值的各段信息，做如下说明：

$TS_{X\_TEMPERATURE}$ 表示当前实际温度，单位℃

$TS_{1\_TEMPERATURE}$ 和 $TS_{2\_TEMPERATURE}$ 表示标定值TS1和TS2的实际温度，单位℃

TS_{1_TEMP} 和 TS_{2_TEMP} 表示标定值TS1和TS2的TEMP段，即bit16 ~ bit27

TS_{1_DATA} 和 TS_{2_DATA} 表示标定值TS1和TS2的DATA段，即bit0 ~ bit11

TS_{X_DATA} 表示当前温度传感器的电压采样值

计算步骤步骤如下：

STEP 1：确定标定值是否为有效数据。

判断 TS_{1_FLAG}、TS_{2_FLAG} 是否等于0x5。若等于0x5，进入STEP 2。否则，不进行计算。

STEP 2：确定标定值标定的实际温度（℃）是否为负温度。

判断TS1和TS2的bit27位是否等于0。若等于0，表示标定温度大于等于0℃。若等于1，表示标定温度小于0℃；

STEP 3：计算标定值所对应的实际温度（℃）。

如果当前标定值的实际温度大于等于0℃，则

$$TS_{1\_TEMPERATURE} = \frac{1}{16} \times TS_{1\_TEMP}（℃）$$

$$TS_{2\_TEMPERATURE} = \frac{1}{16} \times TS_{2\_TEMP}（℃）$$

如果当前标定值的实际温度小于0℃，则

$$TS_{1\_TEMPERATURE} = \frac{1}{16} \times (0 - (((TS_{1\_TEMP} - 1)\text{^}0x7FF) \& 0x7FF))(℃)$$

$$TS_{2\_TEMPERATURE} = \frac{1}{16} \times (0 - (((TS_{2\_TEMP} - 1)\text{^}0x7FF) \& 0x7FF))(℃)$$

STEP 4：代入公式计算当前的实际温度$TS_{X_TEMPERATURE}$

$$TS_{X\_TEMPERATURE} = TS_{2\_TEMPERATURE} - \frac{TS_{2\_TEMPERATURE} - TS_{1\_TEMPERATURE}}{TS_{2\_DATA} - TS_{1\_DATA}} \times (TS_{2\_DATA} - TS_{X\_DATA})$$

举例：

若芯片内部存储的标定值为TS1 = 0x51B00517， TS2 = 0x5D800673，当前ADC采样值TS_{X_DATA} = 0x4AD。

STEP 1：TS_{1_FLAG}、TS_{2_FLAG} 都是0x5，表明数据都有效

STEP 2：TS1的bit27为0说明标定值1的温度大于等于0℃，TS2的bit27为1说明标定值2的温度小于0℃

STEP 3：计算标定值的实际温度

$$TS_{1\_TEMPERATURE} = \frac{1}{16} \times 0x1B0 = 27(℃)$$

$$TS_{2\_TEMPERATURE} = \frac{1}{16} \times (0 - (((0xD80 - 1)\text{^}0x7FF) \& 0x7FF)) = -40(℃)$$

STEP 4：带入公式计算

$$TS_{X\_TEMPERATURE} = -40 - \frac{-40 - 27}{0x673 - 0x517} \times (0x673 - 0x4AD) = 47.41(℃)$$

16.1.20.5 温度传感器采集温度基本步骤

STEP 1：ADC初始化

STEP 2：ADC参考电压选择INTVREF_1.2V，且模拟输入选择TS时，采样到的值为TS_{X_DATA}

STEP 3：读取芯片内部标定值

STEP 4：代入公式计算

16.2 寄存器说明

16.2.1 寄存器表

Base Address of ADC: 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 - 0x004C	转换序列寄存器x (x=0~7)	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 - 0x011C	转换结果寄存器x (x=0~7)	0x00000000
ADC_DRMASK	0x0148	禁止转换结果更新寄存器	0x00000000

16.2.2 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

16.2.3 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

16.2.4 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

16.2.5 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													INTVREF_LVL	INTVREF_OUTEN	RSVD						VREF_SEL				SWTRG	STOP	START	ADCDIS	ADCEN	SWRST
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
RW	R	R	R	R	R	R	R	R	R	R	R	R	R	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: 保留 1: 12位 注意：该位请保持为1.
INTVREF_LVL	[17]	RW	INTVREF_LVL: 内部参考电压输入源选择 0: 内部1.2V电压 1: 内部2.5V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
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: 正向为内部INTVREF输出，负向为VSS 0011: 正向为内部VDD，负向为VREF- 0100: 正向为外部VREF+管脚，负向为VREF- 1000: 正向为内部INTVREF输出，负向为VREF- 其它: 保留
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。

16.2.6 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
MODE		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	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	R	R	R	R	R	RW	RW	RW	RW	RW

Name	Bit	Type	Description
MODE	[31:30]	RW	MODE：转换模式 00: 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 01: 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 10/11: 等待转换模式。待触发来时，按照ADC_SEQx所配置的对应通道转换。
NBRCH	[12:10]	RW	NBRCH[2:0]: 转换序列最大数数 000b:0 001b:1 … 111b:7 注意： 即使在单次转换模式，如果NBRCH[2:0]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转换速率不能超过500KHz。 - 如果系统时钟为40MHz，那么PRLVAL至少为1

16.2.7 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。

16.2.8 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
RSVD								SEQ_END7	SEQ_END6	SEQ_END5	SEQ_END4	SEQ_END3	SEQ_END2	SEQ_END1	SEQ_END0	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
R	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	R

Name	Bit	Type	Description
SEQ_END7~SEQ_END0	[23:16]	W	SEQ_END[x] : SEQx序列转换完成中断 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

16.2.9 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
RSVD								SEQ_END7	SEQ_END6	SEQ_END5	SEQ_END4	SEQ_END3	SEQ_END2	SEQ_END1	SEQ_END0	RSVD		SEQ_INDEX				CTCVS	ADCENS	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_END7~SEQ_END0	[23:16]	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
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

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

16.2.10 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
RSVD								SEQ_END7	SEQ_END6	SEQ_END5	SEQ_END4	SEQ_END3	SEQ_END2	SEQ_END1	SEQ_END0	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	W	W	W	W	W	W	W	W	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W

Name	Bit	Type	Description
SEQ_END7~SEQ_END0	[23:16]	W	SEQ_END[x] : SEQx序列转换完成中断 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

16.2.11 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
RSVD								SEQ_END7	SEQ_END6	SEQ_END5	SEQ_END4	SEQ_END3	SEQ_END2	SEQ_END1	SEQ_END0	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	W	W	W	W	W	W	W	W	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W

Name	Bit	Type	Description
SEQ_END7~SEQ_END0	[23:16]	W	SEQ_END[x] : SEQx序列转换完成中断 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

16.2.12 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
RSVD								SEQ_END7	SEQ_END6	SEQ_END5	SEQ_END4	SEQ_END3	SEQ_END2	SEQ_END1	SEQ_END0	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_END7~SEQ_END0	[23:16]	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
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

16.2.13 ADC_SEQx(转换序列寄存器x (x=0~7))

Address = Base Address+ 0x0030 - 0x004C, 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			RSVD									RESERVED			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	R	R	R	R	R	R	R	R	R	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) 111 : ADC_SYNCIN5 (ETCB)
RESERVED	[7:5]	RW	初始化时必须将这几位改为：000
AIN_SEL	[4:0]	RW	模拟输入通道选择 AIN_SEL值 输入选择 0 AIN0 1 AIN1 2 AIN2 …… …… 25 AIN25 …… N/A 28 TS 29 1/5VDD 30 VSS 31 N/A

16.2.14 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

16.2.15 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。			

16.2.16 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。			

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

Name	Bit	Type	Description
REARM5~REARM0	[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
OSTMD5~OSTMD0	[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。
SYNCEN5~SYNCEN0	[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

16.2.18 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	R	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	[3:0]	RW	TRGEV0，TRGEV1事件的触发源选择。 0000：无触发输出

			0001：EOC事件 0010：READY事件 0011：OVR事件 0100：SEQ_END[0]事件 0101：SEQ_END[1]事件 0110：SEQ_END[2]事件 0111：SEQ_END[3]事件 1000：SEQ_END[4]事件 1001：SEQ_END[5]事件 1010：SEQ_END[6]事件 1011：SEQ_END[7]事件

16.2.19 ADC_DRx(转换结果寄存器x (x=0~7))

Address = Base Address+ 0x0100 - 0x011C, 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。

- ADC_DR0 Address = Base Address + 0x0100, Reset Value = 0x0000_0000
- ADC_DR1 Address = Base Address + 0x0104, Reset Value = 0x0000_0000
- ……
- ADC_DR7 Address = Base Address + 0x011C, Reset Value = 0x0000_0000

16.2.20 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	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
DRMASK	[7:0]	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。

16.3 寄存器说明

16.3.1 寄存器表

Base Address of TS: 0x00080140

Register	Offset	Description	Reset Value
TS_TS1	0x0000	常温下温度传感器信息1	0x00000000
TS_IR1	0x0004	常温下内部1.2V电压参考信息1	0x00000000
TS_TS2	0x0040	高温下温度传感器信息2	0x00000000
TS_IR2	0x0044	高温下内部1.2V电压参考信息2	0x00000000

16.3.2 TS_TS1(常温下温度传感器信息1)

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
FLAG				TEMP												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
FLAG	[31:28]	R	标志位，表示数据是否有效 5h: TEMP，DATA是有效数据 others：TEMP，DATA不是有效数据
TEMP	[27:16]	R	· TEMP的最高位，即bit27为符号位。假设换算的实际温度为Temperature (℃) · 当bit27为0时，则Temperature >= 0，TEMP以原码的形式存储 Temperature = TEMP * 0.0625(℃) · 当bit27为1时，Temperature < 0，TEMP以补码形式存储 Temperature = (0 - (((TEMP - 1) ^ 0x7FF) & 0x7FF)) * 0.0625(℃) · 举例 TEMP Temperature 550h 85(℃) 000h 0(℃) D80h -40(℃)
DATA	[11:0]	R	常温下温度传感器采样值1

16.3.4 TS_TS2(高温下温度传感器信息2)

Address = Base Address+ 0x0040, 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
FLAG				TEMP												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
FLAG	[31:28]	R	标志位，表示数据是否有效 5h: TEMP，DATA是有效数据 others：TEMP，DATA不是有效数据
TEMP	[27:16]	R	· TEMP的最高位，即bit27为符号位。假设换算的实际温度为Temperature (℃) · 当bit27为0时，则Temperature >= 0，TEMP以原码的形式存储 Temperature = TEMP * 0.0625(℃) · 当bit27为1时，Temperature < 0，TEMP以补码形式存储 Temperature = (0 - (((TEMP - 1) ^ 0x7FF) & 0x7FF)) * 0.0625(℃) · 举例 TEMP Temperature 550h 85(℃) 000h 0(℃) D80h -40(℃)
DATA	[11:0]	R	高温下温度传感器采样值2

17 LED控制器（LED DRIVER）

17.1 概述

此 MCU 内嵌一个 LED 自动扫描控制器模块。支持最大 11SEG x 9COM 矩阵驱动或 8 x 9 串行点阵驱动

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

17.1.1 特性

- 最大支持9个大电流灌入口
- 矩阵驱动模式
  - 仅用于控制共阴 LED 数码管
  - 可配置的 COM 通道数（支持从1COM 到9个 COM 扫描模式）
  - 通过寄存器可配置 LED 的刷新率
  - 可选择的亮度调节功能
  - 支持 LED 闪烁扫描
- 点阵驱动模式
  - 最大支持72双灯同时导通模式（共阴）
  - 单个灯导通时间可独立配置
  - 支持正序或逆序扫描，扫描起始口可配置

17.2 功能描述

17.2.1 模块框图

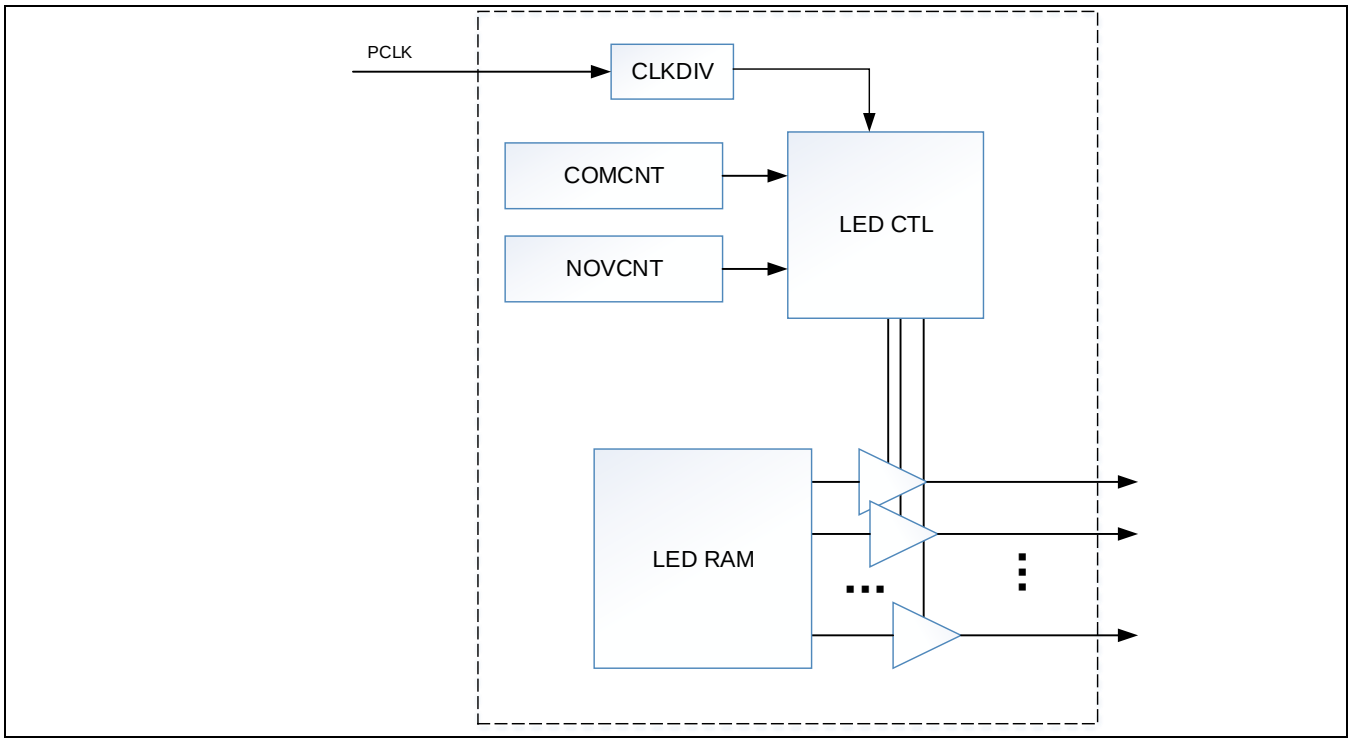


Figure 17-1 LED 扫描模块功能图

17.2.2 工作模式

17.2.2.1 矩阵驱动模式

LED矩阵驱动模式，用11（SEG）+9（COM）根线，最多可以实现99个LED灯的共阴驱动。

17.2.2.2 点阵驱动模式

LED点阵模式一次点两个共阴极LED，对应LED0～LED8口，最多可配置驱动72个灯。支持3x3、4x4、4x5、5x5、6x6、6x7、7x7、8x8、8x9点阵。每个灯在不同点阵配置下，在LED RAM中地址是唯一的。

17.2.3 LED RAM

LED工作在不同模式下时，LED RAM的功能不同。

17.2.3.1 矩阵驱动模式

此模式（LED_CR[MODE] = 0）下，每个COM周期仅RAM中的一列（1bit）有效。即PB0x的COM周期时，仅RAM的bit x有效。

Table 17-1 矩阵驱动模式下的LED RAM

Register	bit8 (PB08)	bit7 (PB07)	bit6 (PB06)	bit 5 (PB05)	bit 4 (PB04)	bit 3 (PB03)	bit 2 (PB02)	bit 1 (PB01)	bit 0 (PB00)
SEGDAT0	SEG0	SEG0	SEG0	SEG0	SEG0	SEG0	SEG0	SEG0	SEG0
…									
SEGDAT15	SEG15	SEG15	SEG15	SEG15	SEG15	SEG15	SEG15	SEG15	SEG15

17.2.3.2 点阵驱动模式

点阵驱动模式下，LED RAM被分为两部分，前8个word（LEDDAT0~7）控制灯阵的ON和OFF，后8个word（LEDWDTH0~7）控制灯阵中每个灯的导通时间。

- LEDDAT0~7控制灯阵的ON和OFF

Table 17-2 点阵驱动模式下的LED RAM第一部分

Register	bit8	bit7	bit6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LEDDAT0	D8	D7	D6	D5	D4	D3	D2	D1	D0
LEDDAT1	D17	D16	D15	D14	D13	D12	D11	D10	D9
…									
LEDDAT8	D71	D70	D69	D68	D67	D66	D65	D64	D63

- LEDWDTH0~7控制灯阵中每个灯的导通时间，每个灯的导通时间都可以从两个预设值（LED_RIMCR）中选择

Table 17-3 点阵驱动模式下的LED RAM第一部分

Register	bit8	bit7	bit6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LEDWDTH0	D8	D7	D6	D5	D4	D3	D2	D1	D0
LEDWDTH1	D17	D16	D15	D14	D13	D12	D11	D10	D9
…									
LEDWDTH8	D71	D70	D69	D68	D67	D66	D65	D64	D63

17.2.4 扫描的工作模式

17.2.4.1 矩阵驱动模式

LED 自动扫描驱动器模块可以自动产生不互相叠加的扫描信号，用以驱动多个8段数码管。多个8段数码管的 SEG 端复用同一个驱动源。本驱动模块支持最大10个共阴极扫描的8段数码管进行并联扫描。在不同的时间段，通过使能 COM 端来点亮不同的数码管。每个 COM 管脚，均支持大电流驱动模式，可以在不外加驱动电路的前提下，直接驱动数码管（外部限流电阻仍然需要）。扫描的 COM 数，可以通过 LED_CR 寄存器中的 COM_EN 来设置。

当需要点亮 LED 时，LED 的扫描时钟频率需要预先设置，而且每个 COM 的扫描有效时间长度也同样需要配置好。扫描的时序控制可以通过 LED_TIMCR 来设置。在每个 COM 使能的时间内，相对应的存储在 SEGDATA 中的数据将会从 SEG7~SEG0口输出。

启动 LED 的自动扫描，通过设置 LED_CR 寄存器中的 LIGHTON 来设置。

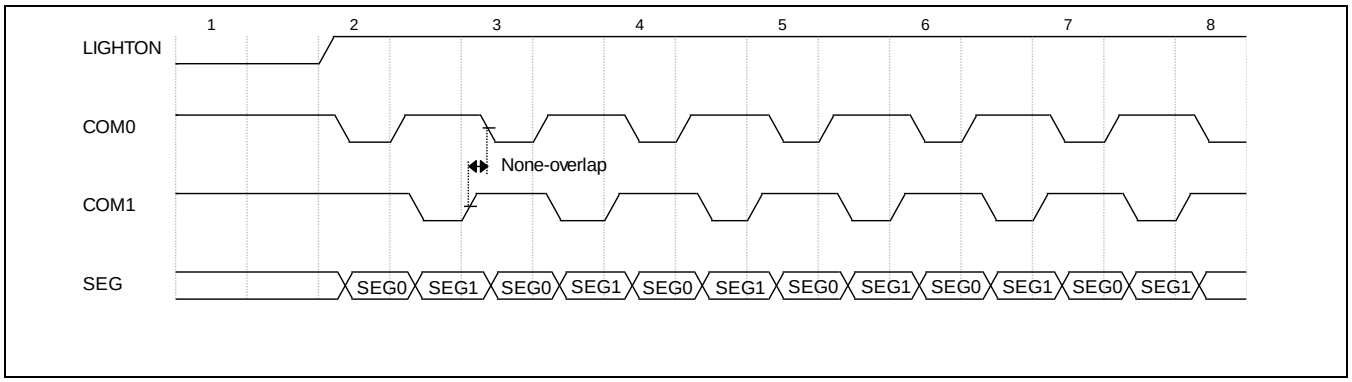


Figure 17-2 LED 扫描显示 （COM_EN = 8'b0000_0011模式）

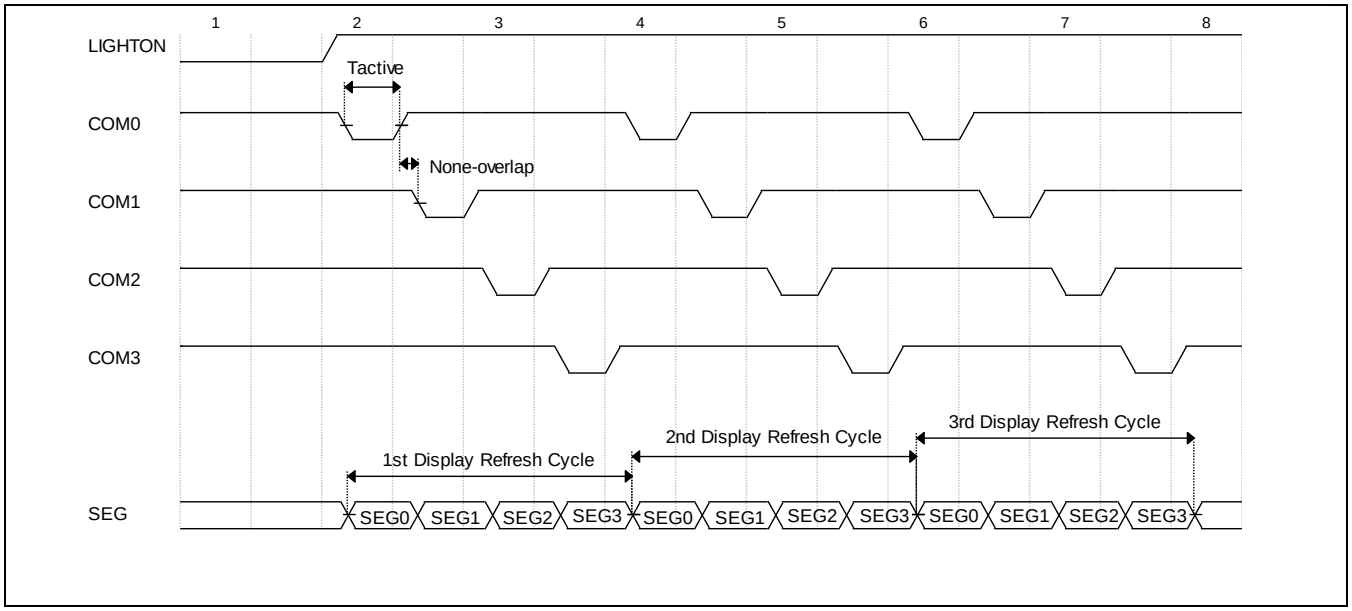


Figure 17-3 LED 扫描显示 （COM_EN = 8'b0000_1111模式）

上图中，T_{active} 是每个 COM 的使能有效时间，它的宽度可以通过下面的公式进行计算：

$$T_{active} = T_{lcd} / 8 \times COMCNT - T_{nov}$$

Where, T_{lcd} is LCD block clock cycle, T_{nov} is none-overlap time, which is decided by NOVCNT register. $T_{nov} = T_{lcd} \times NOVCNT$

LED 的显示亮度，可以通过 LED_BRIGHT 寄存器进行设置。寄存器的值表示不同的 COM 有效时间下的百分比宽度。百分比越低，COM 持续的时间就越短，显示的亮度就越低。

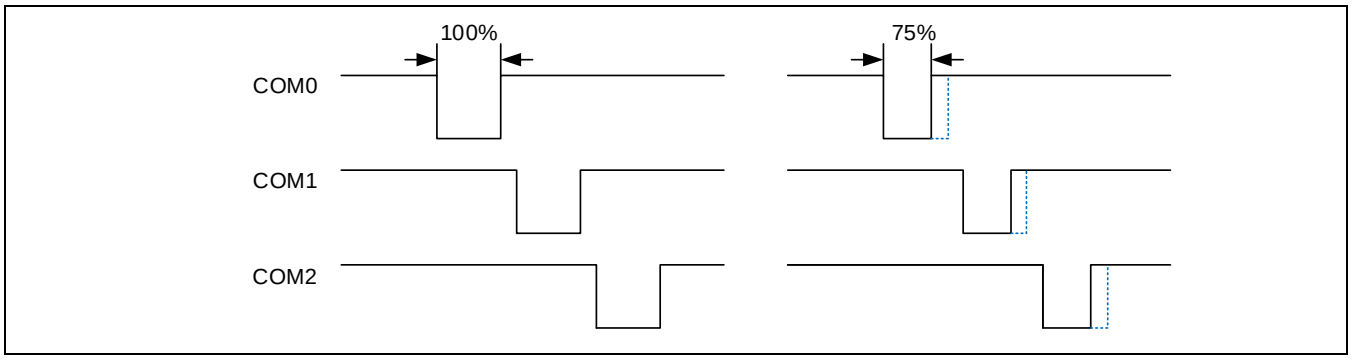


Figure 17-4 LED 显示亮度调节

17.2.4.2 点阵驱动模式

点阵驱动模式用9个IO（LED0~8）实现最大8*9点阵双灯驱动。扫描顺序支持正序扫描或逆序扫描，可通过LED_CR[SCAN_ORDER]设置，且扫描起始LED口可设置为LED0~8中的任意一个，可通过LED_CR[SCAN_IO_ST]设置。

点阵驱动图如下：

- 8*9点阵

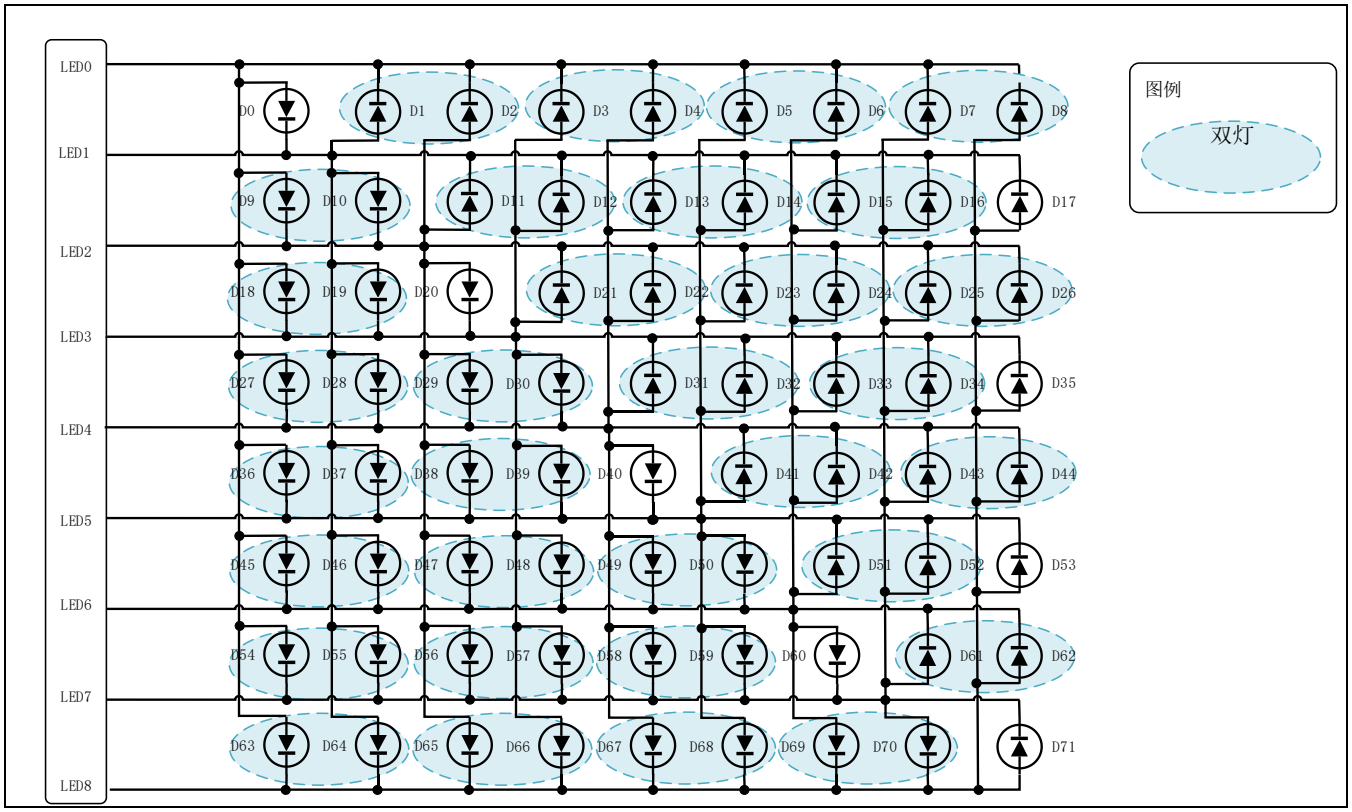


Figure 17-5 8*9点阵驱动

- 8*8点阵

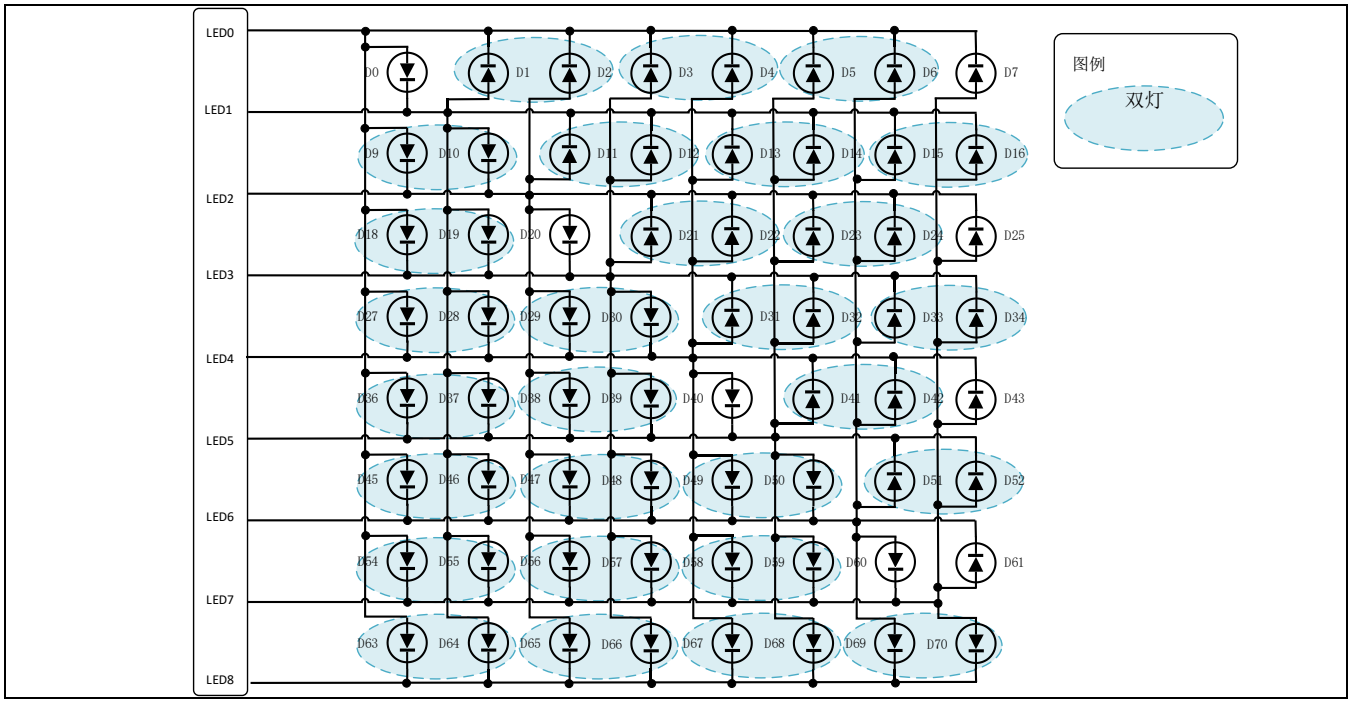


Figure 17-6 8*8点阵驱动

- 7*7点阵

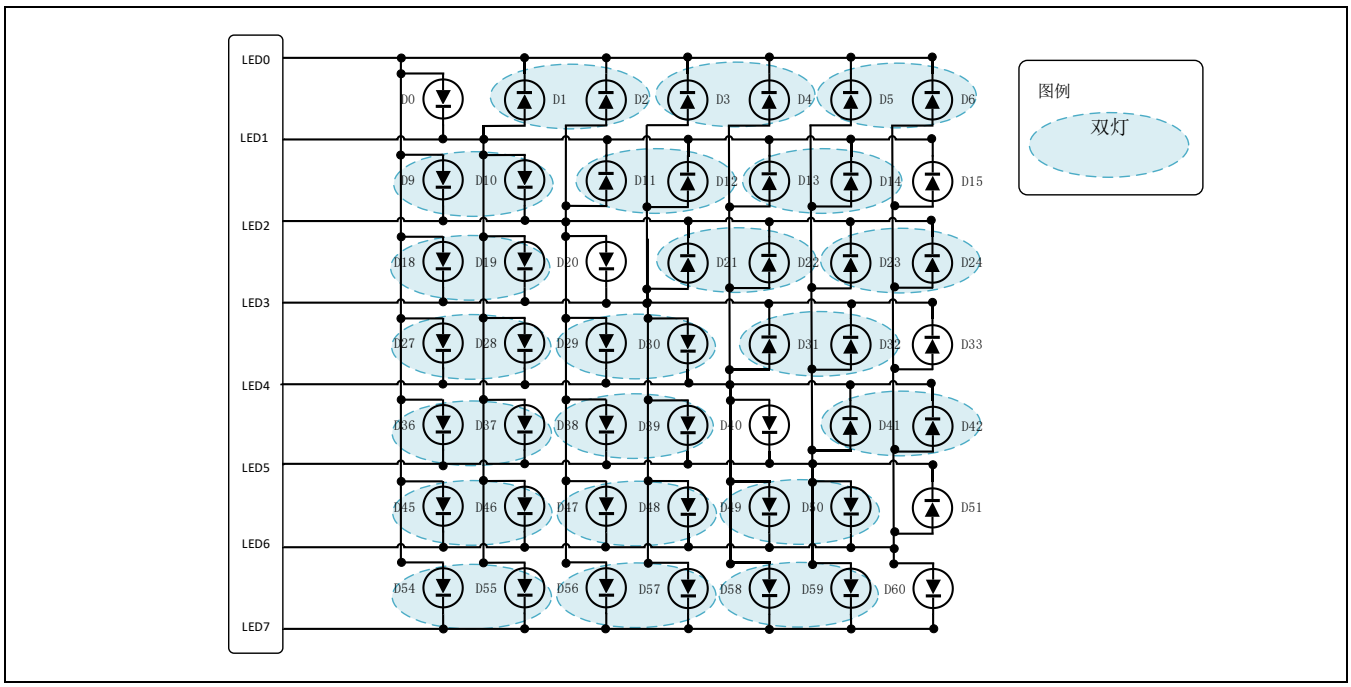


Figure 17-7 7*7点阵驱动

- 6*7点阵

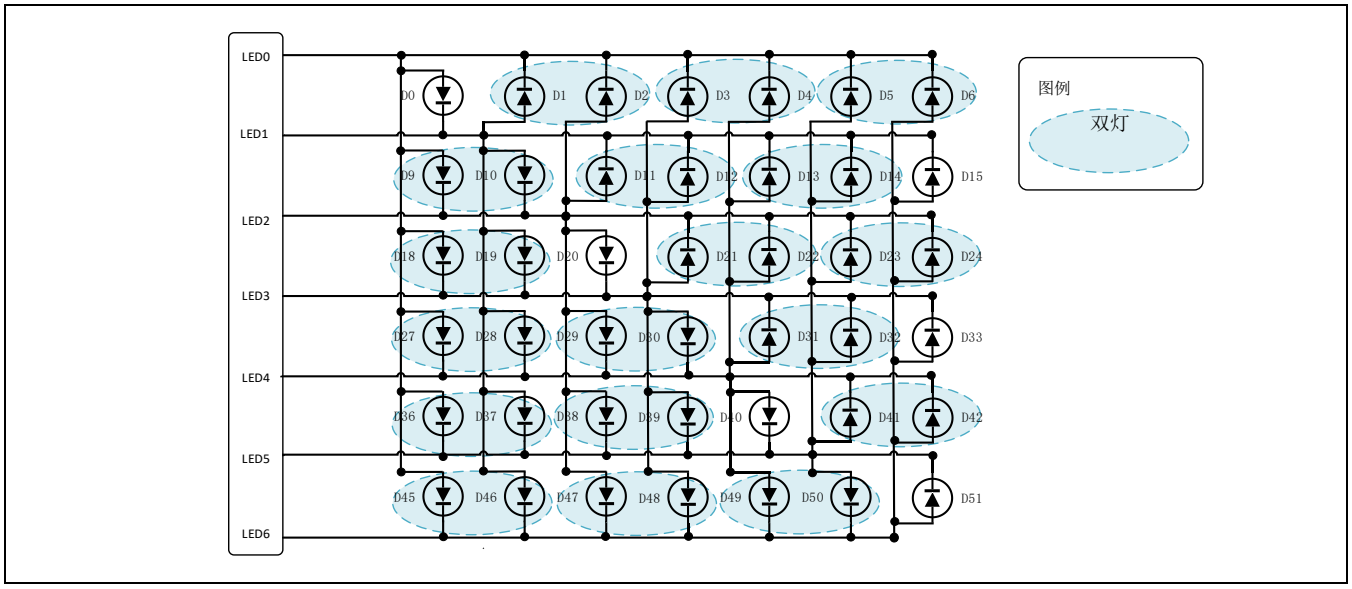


Figure 17-8 6*7点阵驱动

- 6*6点阵

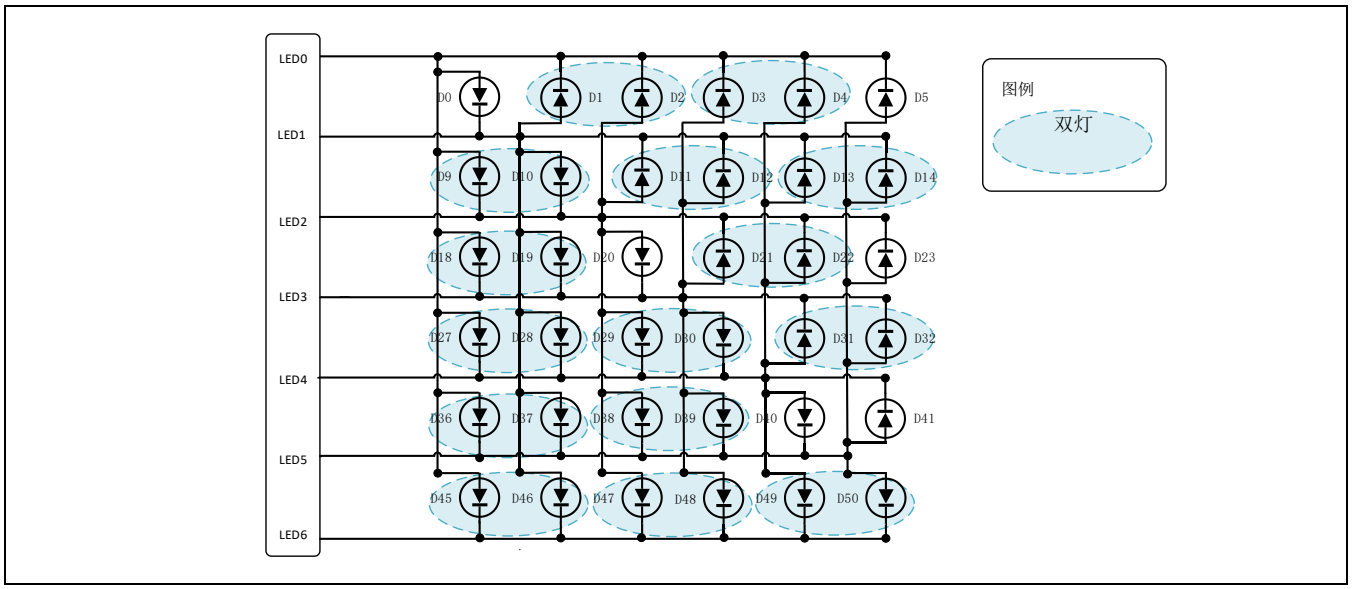


Figure 17-9 6*6点阵驱动

- 5*5点阵

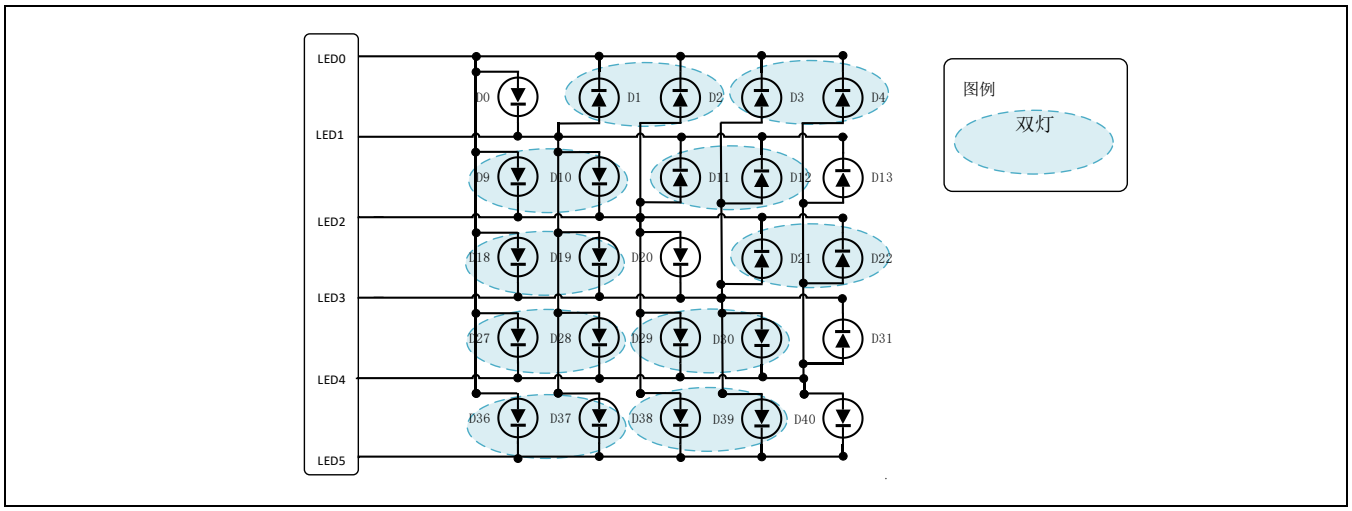


Figure 17-10 5*5点阵驱动

- 4*5点阵

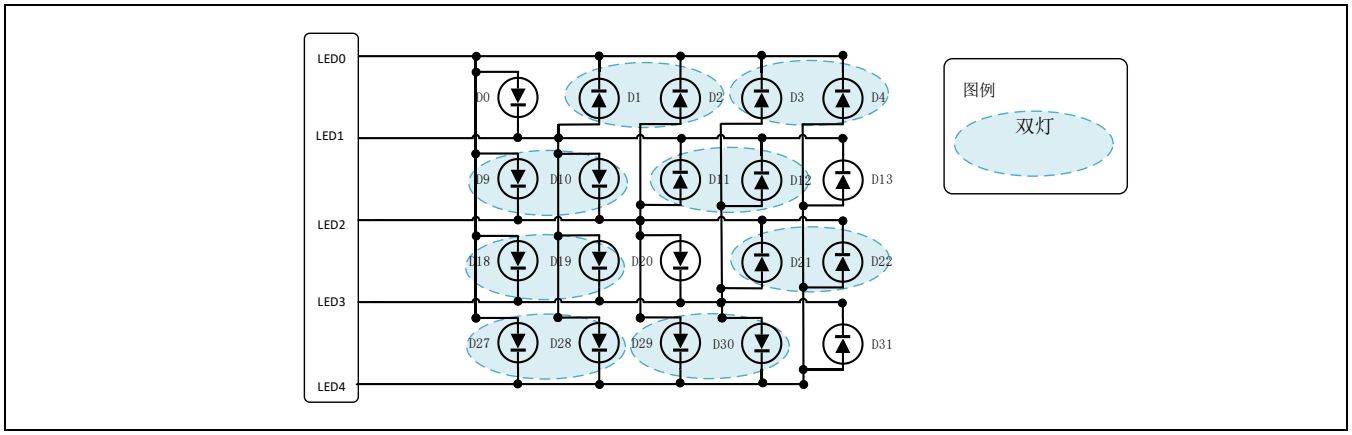


Figure 17-11 4*5点阵驱动

- 4*4点阵

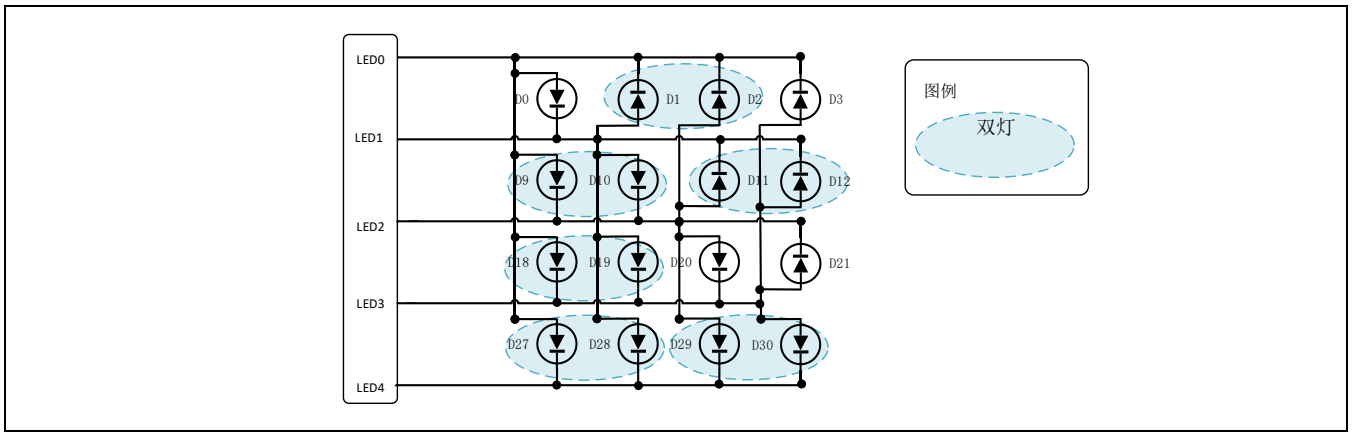


Figure 17-12 4*4点阵驱动

- 3*3点阵

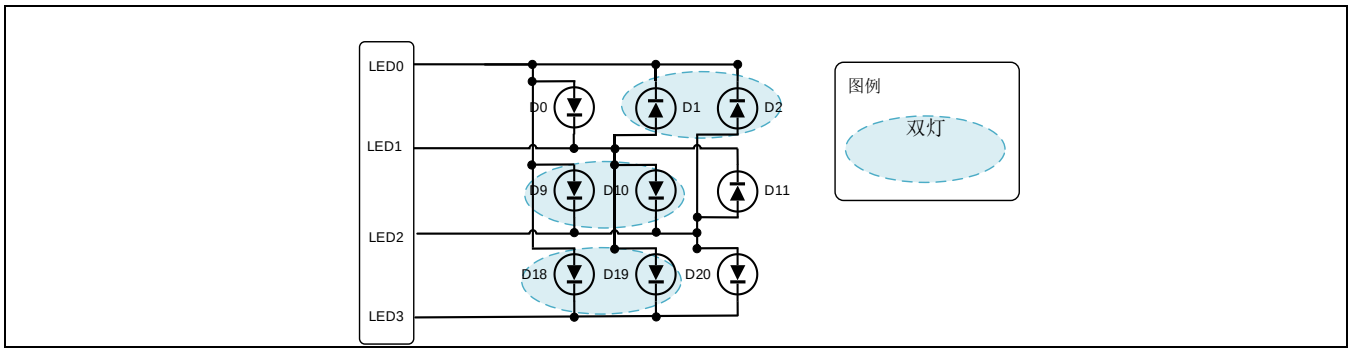


Figure 17-13 3*3点阵驱动

以点亮D0、D1、D2为例，点阵驱动模式的数字控制信号时序如下。

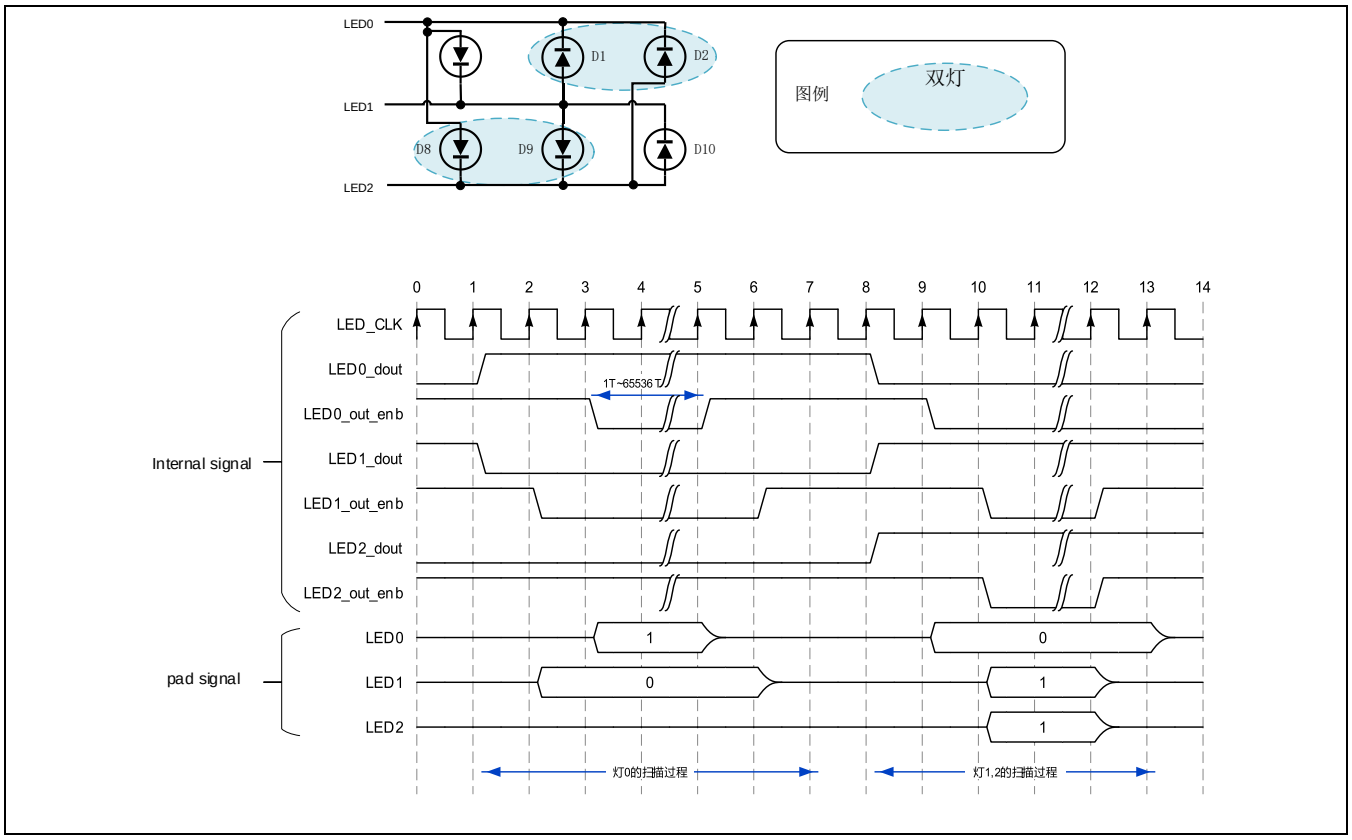


Figure 17-14 点阵驱动数字信号时序图

LED点阵扫描时间

$$T = \left(t_{单灯0扫描时间} + t_{单灯1扫描时间} + \cdots + t_{单灯n扫描时间}\right) + \left(t_{双灯组0扫描时间} + t_{双灯组1扫描时间} + \cdots + t_{双灯组n扫描时间}\right) + \left(n_{单灯总数} + n_{双灯总组数}\right) \times 5 \times t_{LED}$$

其中，$t_{单灯n扫描时间}$由LED_MD1_LEDWDTHx选择LED_TIMR[WIDTH0]或[WIDTH1]中的一个预设值。

$t_{双灯组0扫描时间}$由双灯组中扫描时长较长的一个灯决定。

t_{LED}为LED控制器的时钟周期，由LED_PSCR[PSC]的设置决定。$t_{LED} = \frac{PSC+1}{PCLK}$

17.2.5 闪烁控制

LED 闪烁控制仅在矩阵驱动模式下有效。

LED 的每个扫描 COM 端口，可以独立设置该周期内是否禁止输出，从而实现在不影响扫描显示效果的情况下（扫描顺序和周期不变化），实现特定 COM 的刷新停止，从而实现特定 COM 控制端的闪烁效果。COM 的输出禁止可以通过设置 BLKER 来设置，通过 BLKDR 来清除禁止标志。

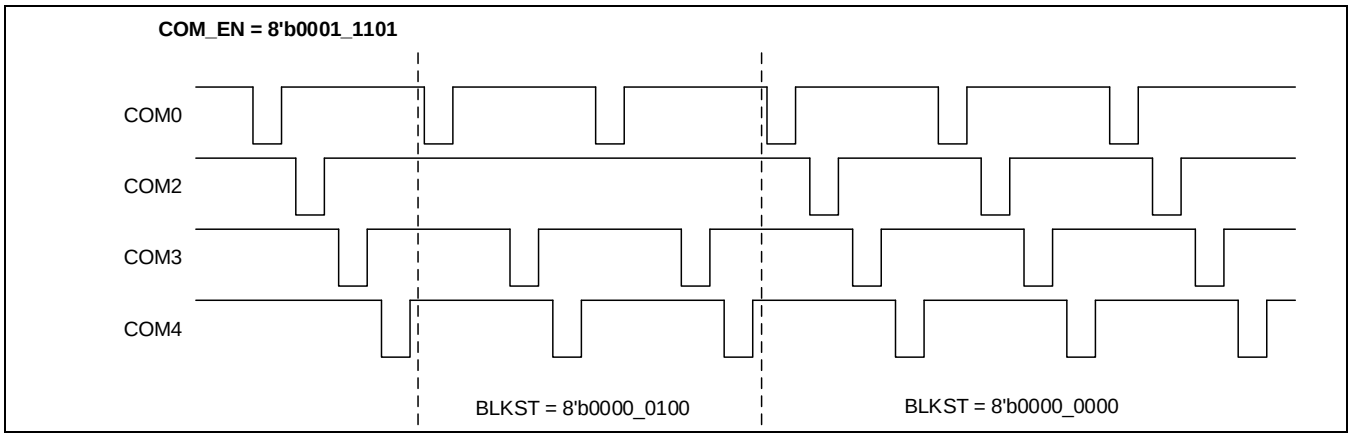


Figure 17-15 LED 闪烁的工作模式

17.2.6 管脚驱动能力

无论点阵驱动模式还是矩阵驱动模式，LED控制输出管脚的驱动能力都可以配置为由GPIO模块控制或LED模块控制。通过LED_DRVSEL寄存器控制。

如果选择LED模块控制，LED管脚驱动能力可以实现按组分别控制。每组5个LED口。驱动能力详见寄存器说明章节LED_DRV寄存器的描述。

Table 17-4 LED模块控制管脚驱动能力时的分组方式

CRU	LED口
0	LED8/COM8，LED7/COM7，LED6/COM6，LED5/COM5
1	LED4/COM4/SEG0，LED3/COM3/SEG1，LED2/COM2/SEG2，LED1/COM1/SEG3
2	LED0/COM0/SEG4，SEG5，SEG6，SEG7
3	SEG8，SEG9，SEG10，SEG11
4	SEG12，SEG13，SEG14，SEG15

如果选择GPIO模块控制，每个LED管脚的驱动能力可以实现独立控制。驱动能力详见数据手册电气参数章节。

17.2.7 中断

LED 模式模块有3个中断。

矩阵驱动模式使用两个：一个为 ICEND 中断，此中断将在每个 COM 扫描结束时触发。一个为 IPEND 中断，此中断将在所有的 COM 扫描结束后触发。

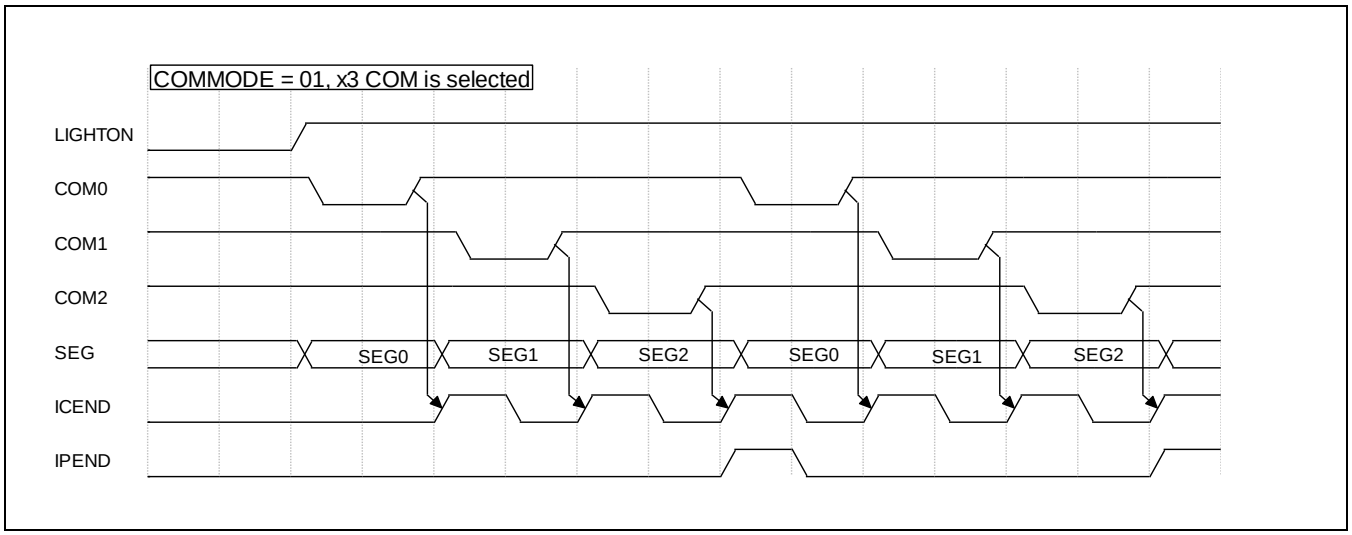


Figure 17-16 中断产生示意图

点阵驱动模式使用一个：FEND 中断，此中断在一帧扫描结束后触发。

17.3 寄存器说明

17.3.1 寄存器表

Base Address of LED: 0x40070000

Register	Offset	Description	Reset Value
LED_CR	0x00	通用控制寄存器	0x00000000
LED_MCR	0x04	矩阵扫描控制寄存器	0x00000000
LED_RISR	0x08	原始中断状态标志寄存器	0x00000000
LED_IMCR	0x0C	中断使能控制寄存器	0x00000000
LED_MISR	0x10	中断标志寄存器	0x00000000
LED_ICR	0x14	中断标志清除寄存器	0x00000000
LED_PSCR	0x18	时钟分频控制寄存器	0x00000030
LED_TIMCR	0x1C	扫描时序控制寄存器	0x00000000
LED_BLKER	0x20	闪烁位设置控制寄存器	0x00000000
LED_BLKDR	0x24	闪烁位清除控制寄存器	0x00000000
LED_BLKST	0x28	闪烁位状态寄存器	0x00000000
LED_DRVSEL	0x30	驱动选择控制器	0x00000000
LED_DRV	0x34	LED SEG驱动电流选择寄存器	0x00000000
LED_DRVEN	0x38	LED SEG驱动使能使能寄存器	0x00000000
LED_MD1_LEDDATx	0x3C~0x58	点阵驱动模式输出ON/OFF寄存器x（x = 0～7）	0x00000000
LED_MD0_SEGDATx	0x3C~0x78	矩阵驱动模式段码输出数据寄存器x（x= 0～15）	0x00000000
LED_MD1_LEDWDTHx	0x5C~0x78	点阵驱动模式输出宽度寄存器x（x = 0～7）	0x00000000

17.3.2 LED_CR(通用控制寄存器)

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		SCAN_ODER	SCAN_IO_ST				ONE_SHOT	RSVD				MATRIX				KEY								MODE	RSVD						LIGHTON
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	RW	RW	RW	RW	RW	RW	RO	RO	RO	RO	RW	RW	RW	RW	WO	WO	WO	WO	WO	WO	WO	WO	RW	R	R	R	R	R	R	RW

Name	Bit	Type	Description
SCAN_ODER	[29]	RW	点阵模式下，扫描顺序控制位 0h: 正序，按PB00->PB08的顺序扫描 1h: 逆序，按PB08->PB00的顺序扫描
SCAN_IO_ST	[28:25]	RW	点阵模式下，扫描起始口控制位 0h: 正序时，起始扫描IO为PB00；逆序时，起始扫描IO为PB08 1h: 正序时，起始扫描IO为PB01；逆序时，起始扫描IO为PB07 2h: 正序时，起始扫描IO为PB02；逆序时，起始扫描IO为PB06 3h: 正序时，起始扫描IO为PB03；逆序时，起始扫描IO为PB05 4h: 正序时，起始扫描IO为PB04；逆序时，起始扫描IO为PB04 5h: 正序时，起始扫描IO为PB05；逆序时，起始扫描IO为PB03 6h: 正序时，起始扫描IO为PB06；逆序时，起始扫描IO为PB02 7h: 正序时，起始扫描IO为PB07；逆序时，起始扫描IO为PB01 8h: 正序时，起始扫描IO为PB08；逆序时，起始扫描IO为PB00
ONE_SHOT	[24]	RW	点阵模式，扫描模式选择 0h: 循环扫描模式 1h: 中断扫描模式 （当选择中断扫描模式，完成一次扫描，硬件清除LIGHTON位，中断标志位置1,软件重写LIGHTON为1后，重新进行扫描
MATRIX	[19:16]	RW	点阵选择 0h: 无效 1h: 3x3点阵 2h: 无效 3h: 4x4点阵 4h: 4x5点阵 5h: 5x5点阵 6h: 无效 7h: 6x6点阵 8h: 6x7点阵 9h: 7x7点阵 Ah: 无效 Bh: 8x8点阵

			Ch: 8x9点阵 其他: 无效
KEY	[15:8]	WO	0x5a: 使能LEDMODE设置，完成设置后LEDKEY自动清零 其他值: 无效设置
MODE	[7]	RW	0h: LED矩阵驱动模式 1h: LED点阵驱动模式 注: 往LEDKEY写0x5A时,才能设置LEDMODE
LIGHTON	[0]	RW	LED扫描启动。 1：启动LED自动扫描 0：停止LED自动扫描

17.3.3 LED_MCR(矩阵扫描控制寄存器)

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																			COM_EN									RSVD	BRT		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW

Name	Bit	Type	Description
COM_EN	[12:4]	RW	矩阵驱动模式下COM扫描的使能选择。COM_EN的每一位对应一个COM扫描信号，当该位为‘1’时，所对应的COM扫描即被打开。 COM_EN[0]：PB00的COM扫描使能 COM_EN[1]：PB01的COM扫描使能 …… COM_EN[8]：PB08的COM扫描使能
BRT	[2:0]	RW	矩阵模式下，显示亮度的设置。 000：COM的宽度为100% 001：COM的宽度为87.5% 010：COM的宽度为75% 011：COM的宽度为62.5% 100：COM的宽度为50% 101：COM的宽度为37.5% 110：COM的宽度为25% 111：COM的宽度为12.5%

17.3.4 LED_RISR(原始中断状态标志寄存器)

Address = Base Address+ 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																													FEND	IPEND	ICEND
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RO	RO	RO

Name	Bit	Type	Description
FEND	[2]	RO	点阵模式下，一帧扫描结束中断。 0：中断事件未发生 1：中断事件已发生
IPEND	[1]	RO	矩阵模式下，所有COM扫描结束中断。 0：中断事件未发生 1：中断事件已发生
ICEND	[0]	RO	矩阵模式下，一个COM扫描结束中断。 0：中断事件未发生 1：中断事件已发生

17.3.5 LED_IMCR(中断使能控制寄存器)

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																													FEND	IPEND	ICEND
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW

Name	Bit	Type	Description
FEND	[2]	RW	点阵模式下，一帧扫描结束中断使能控制。 0：禁止中断 1：使能中断
IPEND	[1]	RW	矩阵模式下，所有COM扫描结束中断使能控制。 0：禁止中断 1：使能中断
ICEND	[0]	RW	矩阵模式下，一个COM扫描结束中断使能控制。 0：禁止中断 1：使能中断

17.3.6 LED_MISR(中断标志寄存器)

Address = Base Address+ 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																													FEND	IPEND	ICEND
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
FEND	[2]	R	点阵模式下，一帧扫描结束中断。 0：中断未发生 1：中断发生
IPEND	[1]	R	矩阵模式下，所有COM扫描结束中断。 0：中断未发生 1：中断发生
ICEND	[0]	R	矩阵模式下，，一个COM扫描结束中断。 0：中断未发生 1：中断发生

17.3.7 LED_ICR(中断标志清除寄存器)

Address = Base Address+ 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																													FEND	IPEND	ICEND
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W

Name	Bit	Type	Description
FEND	[2]	W	清除点阵模式下，一帧扫描结束中断。
IPEND	[1]	W	清除矩阵模式下，所有COM扫描结束中断。
ICEND	[0]	W	清除矩阵模式下，一个COM扫描结束中断。

17.3.8 LED_PSCR(时钟分频控制寄存器)

Address = Base Address+ 0x18, Reset Value = 0x00000030

31	30	29	28	27	26	25	24	23	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	1	1	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
PSC	[9:0]	RW	时钟分频控制 LED扫描时钟从PCLK分频得到。LED扫描频率：FLED = PCLK / (PSC+1)

17.3.9 LED_TIMCR(扫描时序控制寄存器)

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
WIDTH1																WIDTH0															
0	0	0	0	0	0	0	0	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
WIDTH1	[31:16]	RW	LED矩阵驱动模时: WIDTH1[7:0]有效。计数器值决定了相邻两个COM间隔（non-overlap）的长度。该计数器的计数时钟为LEDCLK。计数器的计数值为2(WIDTH1+7) LED点阵驱动模式时: WIDTH1[15:0]有效。LED扫描导通时间值1, 扫描宽度为(WIDTH1 + 1) * tLED
WIDTH0	[15:0]	RW	LED矩阵驱动模时: WIDTH0[7:0]有效。在显示期间，计数器值决定了一个COM显示周期的长度。该计数器的计数时钟为LEDCLK/8。计数器的计数值为WIDTH0+7 LED点阵驱动模式时: WIDTH0[15:0]有效。LED扫描导通时间值0, 导通时间为(WIDTH0 + 1) * tLED

17.3.10 LED_BLKER(闪烁位设置控制寄存器)

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																							COMx_DIS								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
COMx_DIS	[8:0]	W	矩阵模式下，PB0x的COM口扫描周期内输出禁止（只写寄存器） 0：无效果 1：禁止PB0x 的COM口在扫描周期内输出

17.3.11 LED_BLKDR(闪烁位清除控制寄存器)

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																							COMx_CLR								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
COMx_CLR	[8:0]	W	矩阵模式下，PB0x的COM口扫描周期内输出使能（只写寄存器） 0：无效果 1：使能PB0x的COM口在扫描周期内输出

17.3.12 LED_BLKST(闪烁位状态寄存器)

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																								COMx_ST							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
COMx_ST	[8:0]	R	矩阵模式下，PB0x的COM口扫描周期内输出禁止状态 0：扫描输出使能 1：扫描输出禁止

17.3.13 LED_DRVSEL(驱动选择控制器)

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												SEL19	SEL18	SEL17	SEL16	SEL15	SEL14	SEL13	SEL12	SEL11	SEL10	SEL9	SEL8	SEL7	SEL6	SEL5	SEL4	SEL3	SEL2	SEL1	SEL0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
SEL19	[19]	RW	PA0.12脚位驱动能力选择位
SEL18	[18]	RW	PA0.11脚位驱动能力选择位
SEL17	[17]	RW	PA0.10脚位驱动能力选择位
SEL16	[16]	RW	PA0.9脚位驱动能力选择位
SEL15	[15]	RW	PA0.8脚位驱动能力选择位
SEL14	[14]	RW	PA0.7脚位驱动能力选择位
SEL13	[13]	RW	PA0.6脚位驱动能力选择位
SEL12	[12]	RW	PA0.5脚位驱动能力选择位
SEL11	[11]	RW	PA0.4脚位驱动能力选择位
SEL10	[10]	RW	PA0.3脚位驱动能力选择位
SEL9	[9]	RW	PA0.2脚位驱动能力选择位
SEL8	[8]	RW	PB0.8脚位驱动能力选择位
SEL7	[7]	RW	PB0.7脚位驱动能力选择位
SEL6	[6]	RW	PB0.6脚位驱动能力选择位
SEL5	[5]	RW	PB0.5脚位驱动能力选择位
SEL4	[4]	RW	PB0.4脚位驱动能力选择位
SEL3	[3]	RW	PB0.3脚位驱动能力选择位
SEL2	[2]	RW	PB0.2脚位驱动能力选择位
SEL1	[1]	RW	PB0.1脚位驱动能力选择位
SEL0	[0]	RW	PB0.0脚位驱动能力选择位
0h：由GPIO模块控制 1h：由LED模块控制（LED_DRV） 注意，非LED管脚必须配置为"0"，由GPIO模块控制，否则同组LED管脚驱动能力将可能显著降低。			

17.3.14 LED_DRV(LED SEG驱动电流选择寄存器)

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												CRU4				CRU3				CRU2				CRU1				CRU0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
CRU4	[19:16]	RW	LED脚位组4驱动能力选择 I = 2.7mA * (CRU4)
CRU3	[15:12]	RW	LED脚位组3驱动能力选择 I = 2.7mA * (CRU3)
CRU2	[11:8]	RW	LED脚位组2驱动能力选择 I = 2.7mA * (CRU2)
CRU1	[7:4]	RW	LED脚位组1驱动能力选择 I = 2.7mA * (CRU1)
CRU0	[3:0]	RW	LED脚位组0驱动能力选择 I = 2.7mA * (CRU0)

说明：每个配置控制4个LED管脚。

CRU0：LED8/COM8，LED7/COM7，LED6/COM6，LED5/COM5
CUR1：LED4/COM4/SEG0，LED3/COM3/SEG1，LED2/COM2/SEG2，LED1/COM1/SEG3
CUR2：LED0/COM0/SEG4，SEG5，SEG6，SEG7
CUR3：SEG8，SEG9，SEG10，SEG11
CUR4：SEG12，SEG13，SEG14，SEG15

17.3.15 LED_DRVEN(LED SEG驱动使能使能寄存器)

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																											EN4	EN3	EN2	EN1	EN0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
EN4	[4]	RW	LED脚位组4驱动能力使能位
EN3	[3]	RW	LED脚位组3驱动能力使能位
EN2	[2]	RW	LED脚位组2驱动能力使能位
EN1	[1]	RW	LED脚位组1驱动能力使能位
EN0	[0]	RW	LED脚位组0驱动能力使能位

17.3.16 LED_MD1_LEDDATx(点阵驱动模式输出ON/OFF寄存器x（x = 0～7）)

Address = Base Address+ 0x3C~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																							D(8 + 9*x)	D(7 + 9*x)	D(6 + 9*x)	D(5 + 9*x)	D(4 + 9*x)	D(3 + 9*x)	D(2 + 9*x)	D(1 + 8*x)	D(0 + 9*x)
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
D(8 + 9*x)	[8]	RW	LED工作于点阵驱动模式时，对应D(8 + 9*x) ON/OFF控制，即D8、D17... D71 0h：OFF 1h：ON
D(7 + 9*x)	[7]	RW	LED工作于点阵驱动模式时，对应D(7 + 9*x) ON/OFF控制，即D7、D16... D70 0h：OFF 1h：ON
D(6 + 9*x)	[6]	RW	LED工作于点阵驱动模式时，对应D(6 + 9*x) ON/OFF控制，即D6、D15... D69 0h：OFF 1h：ON
D(5 + 9*x)	[5]	RW	LED工作于点阵驱动模式时，对应D(5 + 9*x) ON/OFF控制，即D5、D14... D68 0h：OFF 1h：ON
D(4 + 9*x)	[4]	RW	LED工作于点阵驱动模式时，对应D(4 + 9*x) ON/OFF控制，即D4、D13... D67 0h：OFF 1h：ON
D(3 + 9*x)	[3]	RW	LED工作于点阵驱动模式时，对应D(3 + 9*x) ON/OFF控制，即D3、D12... D66 0h：OFF 1h：ON
D(2 + 9*x)	[2]	RW	LED工作于点阵驱动模式时，对应D(2 +9*x) ON/OFF控制，即D2、D11... D65 0h：OFF 1h：ON
D(1 + 8*x)	[1]	RW	LED工作于点阵驱动模式时，对应D(1 + 9*x) ON/OFF控制，即D1、D10... D64 0h：OFF 1h：ON
D(0 + 9*x)	[0]	RW	LED工作于点阵驱动模式时，对应D(0 + 9*x) ON/OFF控制，即D0、D9... D63

			0h：OFF 1h：ON
说明：点阵驱动控制时，8*9灯阵中每个LED灯都有唯一的地址，其ON/OFF控制对应于LED_MD1_LEDDATx（x=0～7）寄存器构成的矩阵。			

17.3.17 LED_MD0_SEGDATx(矩阵驱动模式段码输出数据寄存器x（x= 0～15）)

Address = Base Address+ 0x3C~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																							SEGx								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
SEGx	[8:0]	RW	LED工作于矩阵驱动模式时，PB0x的COM周期，仅SEGx有效。 如 PB00的COM周期，MD0_SEGDAT0~15的bit0分别对应SEG0～15的显示内容。 PB01的COM周期，MD0_SEGDAT0~15的bit1分别对应SEG0～15的显示内容。 ...

17.3.18 LED_MD1_LEDWDTHx(点阵驱动模式输出宽度寄存器x（x = 0～7）)

Address = Base Address+ 0x5C~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																							D(8 + 9*x)	D(7 + 9*x)	D(6 + 9*x)	D(5 + 9*x)	D(4 + 9*x)	D(3 + 9*x)	D(2 + 9*x)	D(1 + 9*x)	D(0 + 9*x)
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
D(8 + 9*x)	[8]	RW	LED工作于点阵驱动模式时，对应D(8 + 9*x) 导通时间控制，即 D8、D17... D72 0h：WIDTH0 1h：WIDTH1
D(7 + 9*x)	[7]	RW	LED工作于点阵驱动模式时，对应D(7 + 9*x) 导通时间控制，即 D7、D16... D71 0h：WIDTH0 1h：WIDTH1
D(6 + 9*x)	[6]	RW	LED工作于点阵驱动模式时，对应D(6 + 9*x) 导通时间控制，即 D6、D15... D70 0h：WIDTH0 1h：WIDTH1
D(5 + 9*x)	[5]	RW	LED工作于点阵驱动模式时，对应D(5 + 9*x) 导通时间控制，即 D5、D14... D69 0h：WIDTH0 1h：WIDTH1
D(4 + 9*x)	[4]	RW	LED工作于点阵驱动模式时，对应D(4 + 9*x) 导通时间控制，即 D4、D13... D68 0h：WIDTH0 1h：WIDTH1
D(3 + 9*x)	[3]	RW	LED工作于点阵驱动模式时，对应D(3 + 9*x) 导通时间控制，即 D3、D12... D67 0h：WIDTH0 1h：WIDTH1
D(2 + 9*x)	[2]	RW	LED工作于点阵驱动模式时，对应D(2 + 9*x) 导通时间控制，即 D2、D11... D66 0h：WIDTH0 1h：WIDTH1
D(1 + 9*x)	[1]	RW	LED工作于点阵驱动模式时，对应D(1 + 9*x) 导通时间控制，即 D1、D10... D65 0h：WIDTH0 1h：WIDTH1
D(0 + 9*x)	[0]	RW	LED工作于点阵驱动模式时，对应D(0 + 9*x) 导通时间控制，即

			D0、D9... D64 0h：WIDTH0 1h：WIDTH1
说明： 1）点阵驱动控制时，8*9灯阵中每个LED灯都有唯一的地址，其驱动时间控制对应于LED_MD1_WDTHx（x=0～7）寄存器构成的矩阵。 2）WIDTH0和WIDTH1对应的时间在LED_TIMCR中配置。			

18 电容式触摸按键传感器（TOUCH）

18.1 概述

此MCU内嵌了一个最大支持26个扫描通道的电容式触摸按键检测模块。该模块支持基于电荷转移的检测技术，以满足不同应用条件下电容触摸检测。

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

18.1.1 特性

- 最大支持26通道按键检测
- 可设置通道扫描时间，扫描超时时间，序列间隔时间
- 支持硬件debounce滤波
- 支持baseline自动跟随
- 支持低功耗模式，在低功耗模式下，debounce和baseline跟随仍然有效
- 多种扫描触发模式：
  - 软件触发
  - 硬件触发（通过ETCB）

18.1.2 管脚描述

Table 18-1 TOUCH KEY 管脚描述

Pin Name	Function	I/O Type	Active Level	Comments
TCHx	触摸检测通道	A	-	-

18.2 功能描述

18.2.1 模块框图

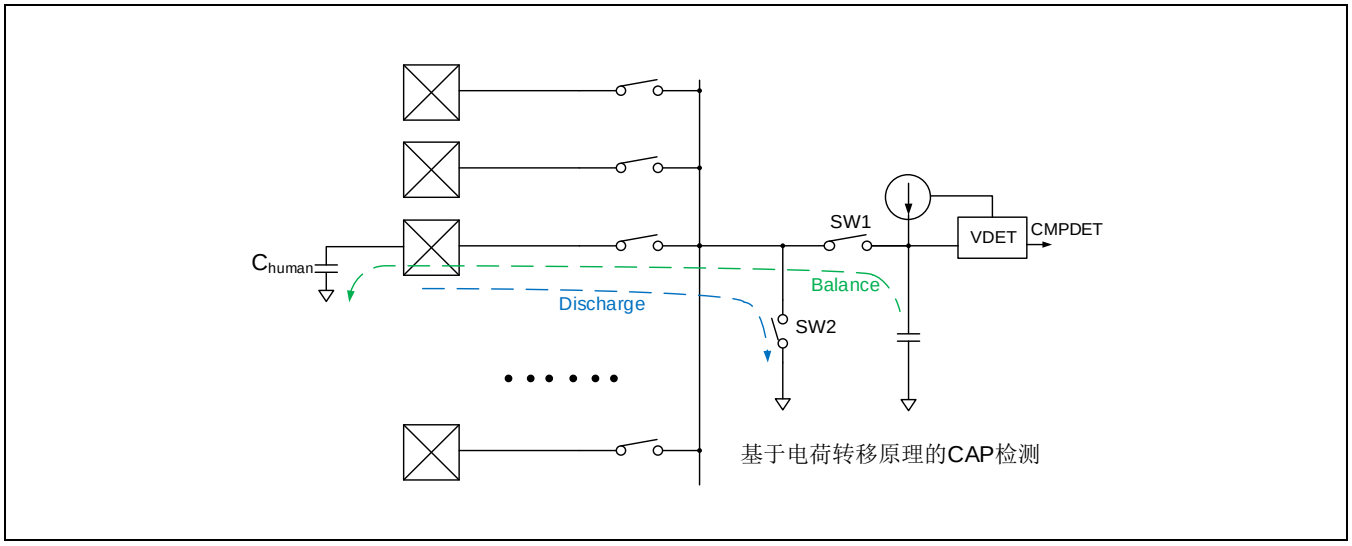


Figure 18-1 Touch Sensor模拟框图

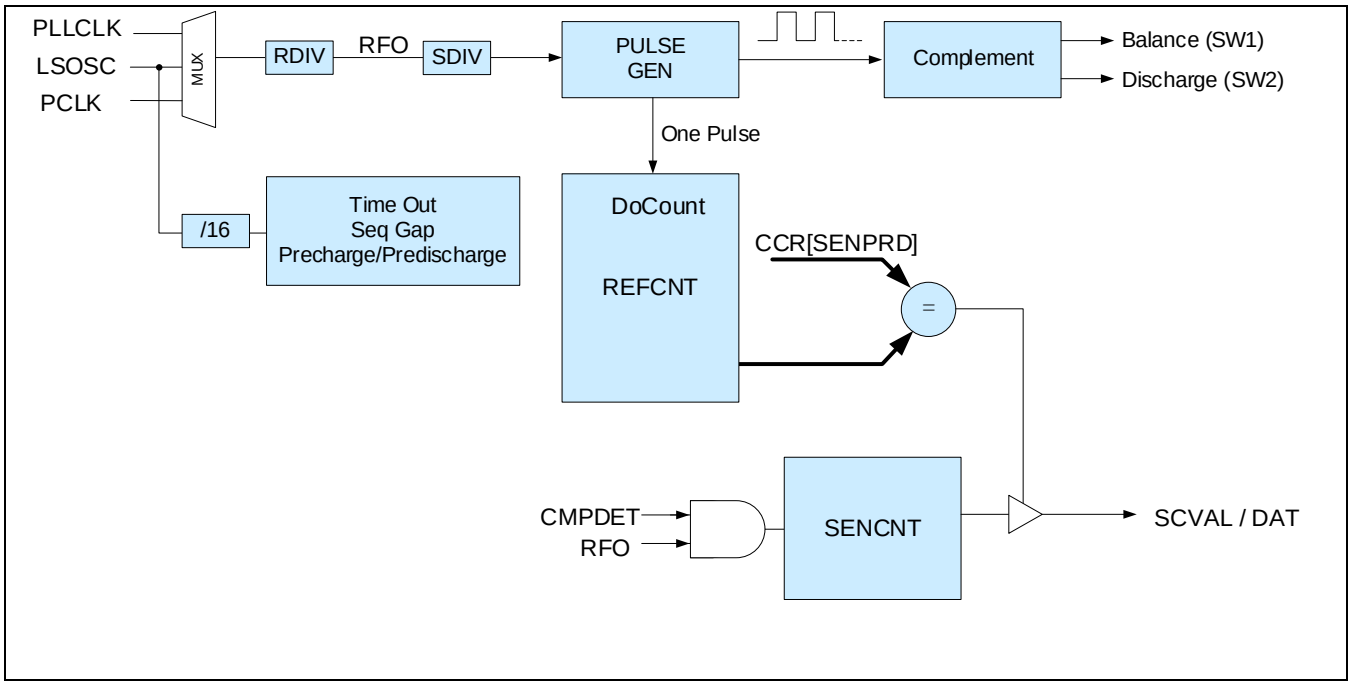


Figure 18-2 Touch Sensor数字框图

18.2.2 工作原理

电容式按键传感器是一种基于自电容检测技术，在人体或带电物体靠近传感极点时，导致自电容的变化，根据这种变化从而实现按键或者触摸滑条等应用的实现。

由随机时钟调制后控制 TOUCH IO 对触摸电容充放电（频率固定，随机相位）。充电电流由内部 LDO 提供，LDO 的输出电流镜像给感应振荡器 S-OSC，控制 S-OSC 输出频率。因为充电频率固定，S-OSC 输出频率正比于 TOUCH IO 负载电容，在 R-OSC 经过 N 个周期所确定的固定时间内，SFO 的周期数将被一个内部采样计数器记录（CHxCNT）。寄生电容变大时，CHxCNT 值会变大；寄生电容变小时，CHxCNT 值会随之变小。

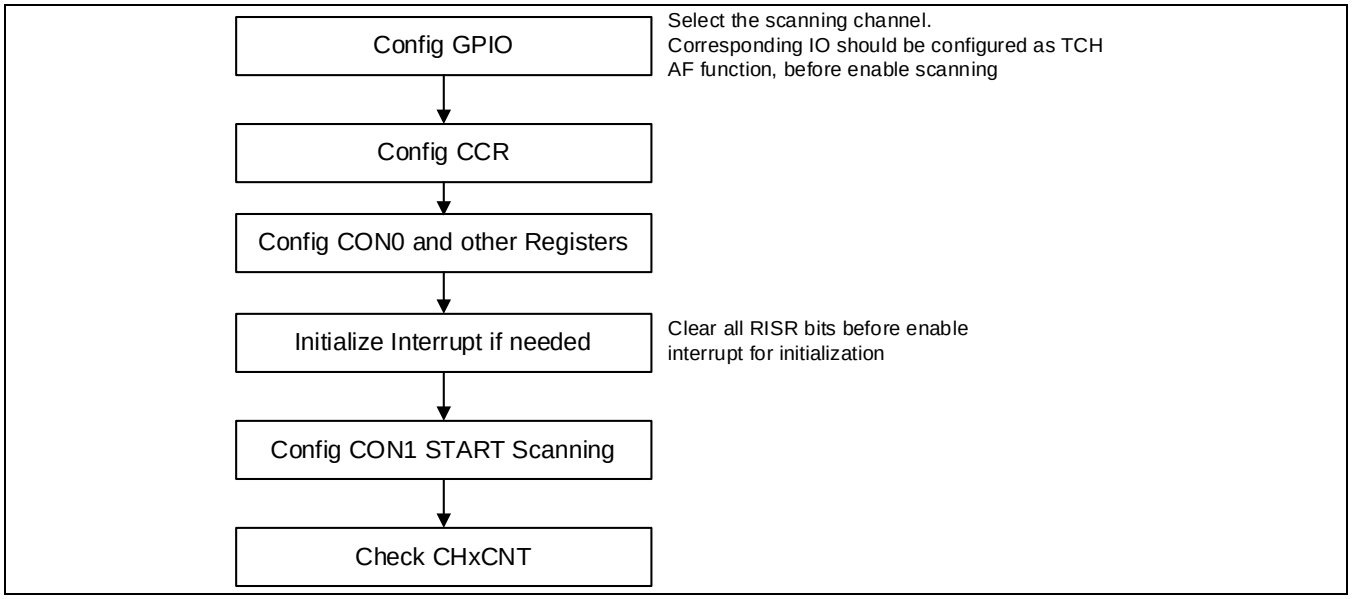


Figure 18-3 软件配置流程

18.2.3 模块内工作时钟

18.2.3.1 TKEY模块内存在四个时钟

- 一个触摸模块时钟，RFO，可以从 HFOSC、LSOSC 和 PLL 中选择，用于模块内部工作时钟。正常工作时使用 HFOSC 或 PCLK，低功耗模式使用 LSOSC
- 一个控制扫描，RFO 经 SDIV 分频后控制扫描时序
- 一个用于定时，来自 ISOSC，控制扫描超时、序列间隔时间
- 一个寄存器读写时钟，PCLK

18.2.3.2 扫描相关时间设置

- 固定扫描时间

所有通道采用相同的扫描时间，由TKEY_CCR[SENPRD]决定，时基为所选时钟源经RDIV和SDIV分频后的时钟。

- 扫描超时时间

扫描目标值需要在配置的扫描时间内能够完成，如果在扫描时间到达时，REFCNT计数值未达到配置的扫描目标值，则视为扫描超时异常，可通过TCH_RISR[TO]查看通道超时状态，本次扫描结果无效，需要调整该通道扫描时间和扫描目标值以保证扫描能够在此时间内达到扫描配置目标值。

- 序列间隔时间

可以通过TCH_CON2[GAPCNT]设置两个序列间的扫描间隔时间，时基为ISOSC/16 = 31.25KHz。

- 预充/预放电时间

每个通道扫描前可选择预放电和预充电，使regulator为固定值。预放电和预充电的使能和时长可通过CON0寄存器进行设置。时基为ISOSC = 500KHz。

18.2.4 扫描模式配置

一般扫描模块工作于高性能模式（TCH_CON0[HM_HPEN] = 1），如果在低功耗模式运行，可以选择低功耗模式（TCH_CON0[HM_HPEN] = 0）。

扫描方式可以选择单次模式，每次只扫描一个序列（TCH_CON0[HMEN] = 0）；也可以选择连续模式，实现循环扫描（TCH_CON0[HMEN] = 1）。

18.2.5 扫描通道配置

在扫描开始前，作为扫描通道的管脚必须配置为TCHx功能，并且通过TCH_SEQCONx配置各通道参数，包括：

- 通道对应的管脚TCHx。如果对应的管脚没有配置为TCH的AF功能，RISR[CHERR]会置起。
- 该通道的ICON。建议在确定扫描周期后，通过调整ICON，使采样值自适应到满量程值的一定比例附近（如70%）。
- 该通道的Baseline OFFSET。这是一个负向的OFFSET，会在采样到数值的基础上减去该OFFSET，适用于采样值很大的情况。

一次序列扫描包含的扫描通道数可以通过TCH_CON0[SEQLEN]来设置。当一个序列内所有通道扫描完成后，RISR[SEQDNE]会置位。

18.2.6 扫描启动

置位TCH_CON1[START]位可以软件启动扫描。

除软件启动扫描方式，芯片还支持自动硬件触发扫描方式工作。自动扫描模式下，当START控制位使能以后，硬件将会在上一轮扫描结束以后并满足触发条件时，自动启动下一轮扫描。

18.2.7 Baseline跟随功能

开启硬件处理器（TCH_CON1[HW_PROC_EN] = 1），即开启了Baseline跟随功能。Baseline跟随功能主要是用于实现对环境因素变化的自适应，特别是当触摸模块在低功耗下运行，软件无法实时调整的情况下。

18.2.8 Debounce功能

Debounce功能用于自动按键检测。基本原理是，当某个通道若干次采样的结果都超过预设值时，判断为有键按下，RISR[PRESSED]置起。

Debounce功能可以用于高性能模式和低功耗模式。

18.2.9 采样数据的读取

触摸模块工作于一般模式时，可以从TCH_DATx[SCANDAT]中读取当次采样值。

当Baseline跟随功能开启时，可以从TCH_HW_DATx[BASELINE]中读取当次采样值。该域只有16位，如果baseline数值比较大，超过16位时，可以通过TCH_OPTCR[DATA_SEL]选择TCH_HW_DATx[BASELINE]对应的原始数据格式。

18.2.10 中断产生

每个通道都有独立的扫描完成中断。通过设置TCH_IMCR位，可以设置该通道扫描完成后触发相应的通道扫描完成中断。

18.2.11 事件触发（输出）

TOUCH可以产生触发信号，通过ETCB实现对其他外设的任务。TOUCH触发输出源由TCH_EVTRG，TCH_EVPS寄存器进行配置。

18.3 寄存器说明

18.3.1 寄存器表

Base Address of TOUCH: 0x40020000

Register	Offset	Description	Reset Value
TCH_IDR	0x000	ID寄存器	0x00000652
TCH_CCR	0x004	时钟控制寄存器	0x10000000
TCH_CON0	0x008	通用控制寄存器0	0x00000000
TCH_CON1	0x00C	通用控制寄存器1	0x00000000
TCH_RSSR	0x010	复位/启动寄存器	0x00000000
TCH_THR	0x014	偏移阈值寄存器	0x00000000
TCH_SCVAL	0x018	当前扫描通道采样值寄存器	0x00000000
TCH_TKST	0x01C	当前扫描通道状态寄存器	0x00002001
TCH_CHINF	0x020	当前扫描通道信息寄存器	0x00000000
TCH_RISR	0x024	原始中断状态寄存器	0x00000000
TCH_MISR	0x028	中断状态寄存器	0x00000000
TCH_IMCR	0x02C	中断使能寄存器	0x00000000
TCH_ICR	0x030	中断清除寄存器	0x00000000
EVTRG	0x034	触发事件控制寄存器	0x00000000
EVPS	0x038	触发周期控制寄存器	0x00000000
EVSWF	0x03C	软件触发寄存器	0x00000000
TCH_HWCR0	0x040	硬件控制寄存器0	0x040A040A
TCH_HWCR1	0x044	硬件控制寄存器1	0x04200828
TCH_HWCR2	0x048	硬件控制寄存器2	0x00000100
TCH_CON2	0x04C	通用控制寄存器2	0x01000100

TCH_DATx	0x1000 - 0x107C	采样值寄存器	0xXXXXXXXX
TCH_HW_DATx	0x1000 - 0x107C	硬件处理器使能时baseline值寄存器	0xXXXXXXXX
TCH_SEQCONx	0x2000 - 0x207C	通道控制寄存器	0xXXXXXXXX
TCH_HW_SEQCONx	0x2000 - 0x207C	硬件处理器使能时通道控制寄存器	0xXXXXXXXX