APT32F173x 系列使用手册 V2.0

173x 系列通用型 32 位微处理控制器

1.1 简介

该使用手册面向应用开发人员，旨在给与 173x 系列内核、存储及全部外围的完整信息。

1.2 相关文档

APT32F1730 数据手册

APT32F1731 数据手册

APT32F1731B 数据手册

APT32F1732 数据手册

版权所有©深圳市爱普特微电子有限公司

本资料内容为深圳市爱普特微电子有限公司在现有数据资料基础上慎重且力求准确无误编制而成，本资料中所记载的实例以正确的使用方法和标准操作为前提，使用方在应用该等实例时请充分考虑外部诸条件，深圳市爱普特微电子有限公司不担保或确认该等实例在使用方的适用性、适当性或完整性，深圳市爱普特微电子有限公司亦不对使用方因使用本资料所有内容而可能或已经带来的风险或后果承担任何法律责任。基于使本资料的内容更加完善等原因，公司保留未经预告的修改权。

1.3 产品分类定义

Table 0-1 APT32F173x全系列产品分类定义

资源		1730	1731	1731B	1732
异	HSIO[1]	14(max)	14(max)	14(max)	14(max)
	ADC	1	1	1	2
	ADCIN	47/33/22	47/33/22	47/33/22	47/33
	OPA	0	4	2	4
	GPIO	54/40/29	54/40/29	54/40/29	54/40
	FREQ	105M(Max)	72M(Max)	105M(Max)	105M(Max)
	PFlash	64K	64K	64K	128K
	SRAM	16K	16K	16K	32K[3]
	Pin	64/48/32	64/48/32	64/48/32	64/48
	UART	3/3/2	3/3/2	3/3/2	3
同	DFlash	8K			
	DMA	2*6ch			
	LED	10com*8seg			
	DAC	1			
	CMP	3			
	USART	3			
	IWDT	1			
	WWDT	1			
	BT	4			
	CNTA	1			
	RTC	1			
	LPT	1			
	GPTA	4			
	GPTB	6			
	IIC	1			
	SIO[2]	2			
	CAN	1			
	SPI	2			
	Package	LQFP64 LQFP48 QFN32	LQFP48 LQFP32 QFN32	LQFP48 QFN32	LQFP64 LQFP48

[1]：HSIO(High Sink IO)，指具有大驱动能力（灌电流）的 IO。

[2]：SIO（Serial IO）

[3]：32K 的 SRAM 可以配置成：24K-DSRAM + 8K-ISRAM 或 16K-DSRAM + 16K-ISRAM。其中 DSRAM 为数据存储，ISRAM 为指令存储。

历史版本说明

版本	修改日期	修改概要
P0.0	2022-10-27	初版
V0.0	2023-5-8	正式版本初版
V1.0	2023-7-21	1、 根据寄存器覆盖率测试，更新了SYSCON、GPTB、LED、I2C、SPI、CA、BT 、RTC、CAN、SIO、ADC、DAC、CMP等模块寄存器，包含功能及读写属性； 2、 修改GPTB手册描述不正确的内容； 3、 更新了LPT、USART、UART、SPI页码格式 4、 去掉MEMORY章节中关于hwdiv的内容； 5、 更新了SYSCON中iomap图表中的描述，分别对应spi0和uart2功能。
V1.1	2023-10-27	1、 更新ADC外部正向参考VREF+功能，并修改温度传感器说明 2、 更新GPTA捕获框图和功能描述 3、 增加IRDA寄存器，并增加相关功能描述 4、 更新LPT寄存器 5、 增加DAC作为触发事件源的说明 6、 增加对PMP功能和寄存器的描述 7、 IFC章节删除IFP，并修改相关描述
V1.2	2024-10-28	1、 ADC章节修正转换时间计算等 2、 SYSCON章节增加时钟质量检测功能的说明
V2.0	2025-5-26	1、增加1730/1731B特性对比 2、RTC CR[OSEL]内容修改 3、部分章节格式修改

1 系统存储空间

1.1 概述

本章节介绍了芯片系统存储空间管理。

本章包含内容如下：

- 存储地址表
- 物理内存保护功能
- 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

1.2 系统总线架构

芯片集成

- E906fp内核，核载8KIcache
- 两个DMA挂接不同的外设： DMA1的操作对象可以是FLASH、SRAM、GPIO、所有APB上外设；DMA0操作对象只能是SRAM和所有APB上外设。
- 一个可配置的SRAM：总共32KB的SRAM空间由SRAM0和SRAM1组成。SRAM0和SRAM1的空间大小可以通过SYSCON_OPT1[SRAMBLKCTRL]配置为SRAM0 24KB + SRAM1 8KB或SRAM0 16KB+ SRAM1 16KB。其中SRAM0只能用作数据存储（DSRAM），SRAM1可通过SYSCON_OPT1[SRAN1FUNCCTRL]配置为指令存储（ISRAM）。一旦SRAM1配置为ISRAM，其地址会自动映射到0x40000开始的地址空间且其中的内容将不能被修改，除非重新设置SYSCON_OPT1[SRAN1FUNCCTRL]，或复位。

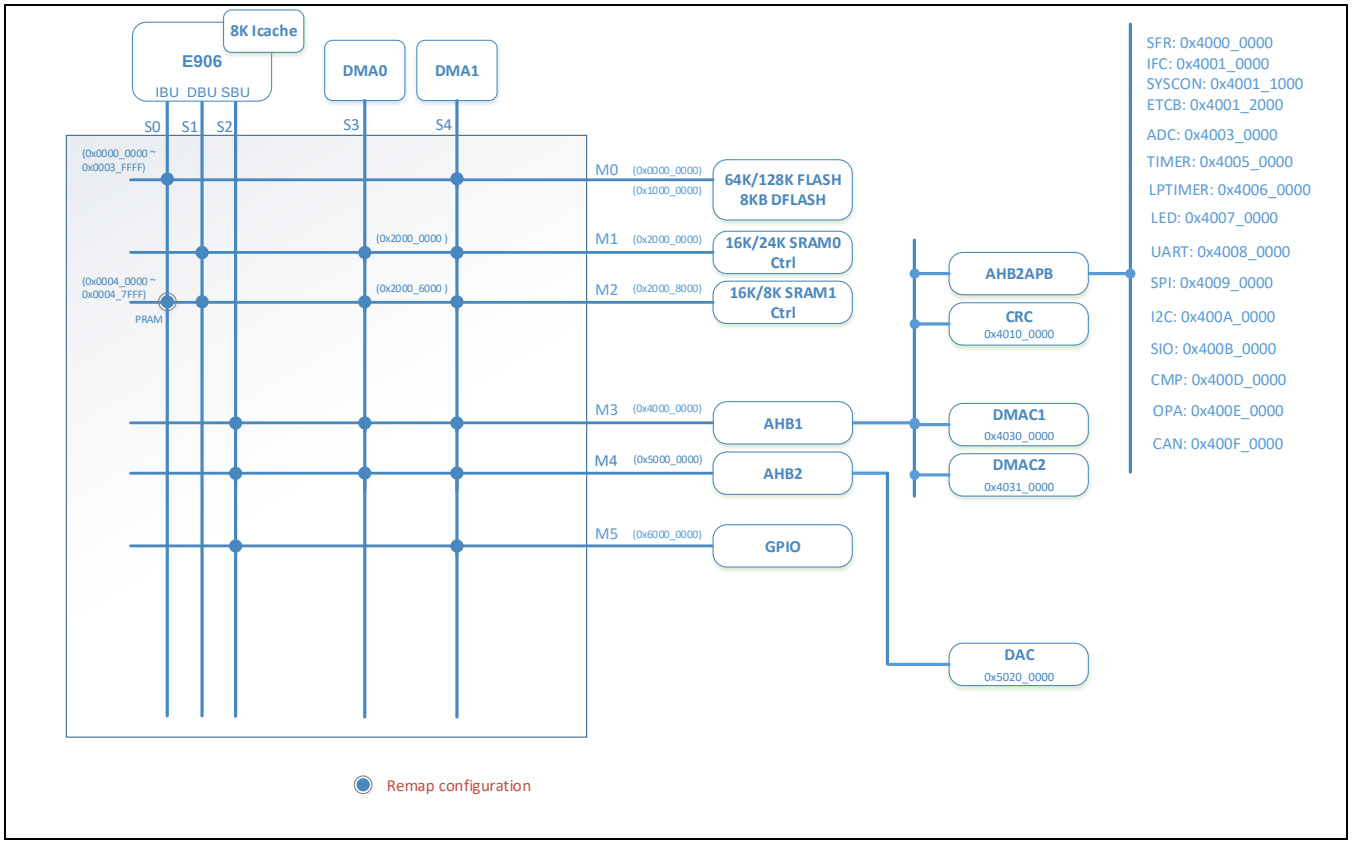


Figure 1-1 系统总线架构

1.3 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
0x6000_0000 to 0x6000_FFFF	GPIO控制器
Reserved	Reserved
0x4000_0000 to 0x400F_FFFF	特殊功能寄存器 (SFR)
Reserved	Reserved
0x2000_0000 to 0x2000_FFFF	SRAM
Reserved	Reserved
0x1000_0000 to 0x1000_1FFF	数据闪存 (Data Flash)
Reserved	Reserved
0x0000_0000 to 0x0003_FFFF	程序闪存 (Program Flash)

1.4 物理内存保护

物理内存保护单元（Physical Memory Protection, PMP）是E906内核中遵从RISC-V标准的模块。PMP 主要负责对存储器和外围设备访问的合法性进行检查，判定当前工作模式下CPU 是否具备对内存地址的读/写/执行权限。

APT32F173系列的PMP单元

- 支持8个可配置表项，当同一地址落入两个表项控制范围内时，实际受控于受表项号小的表项
- 支持OFF（关闭），TOR（Top of Range），NAPOT两种地址匹配模式
- 支持可读、可写、可执行三种权限的配置
- 支持表项lock功能

PMP控制通过操作PMPCFG0~1，PMPADDR0~7实现最多8个控制区间的内存保护。这些寄存器都需要通过特殊的汇编指令去控制。

1.4.1 地址匹配模式

1.4.1.1 OFF

在OFF模式下，该表项无效，即不使能，所有访问不会命中该表项。

1.4.1.2 TOR

在TOR模式中，PMPADDR[4:0] 不参与地址匹配逻辑运算。

表项0的地址空间下边界固定为0，上边界由PMPADDR0[29:5]决定，受控地址范围为(0, {PMPADDR0[29:5], 7’ b0})；表项1的地址空间下边界即表项0的上边界，上边界由PMPADDR1[29:5]决定，即表项1的受控地址范围为({PMPADDR0[29:5], 7’ b0}, {PMPADDR1[29:5], 7’ b0})；以此类推。如果一个表项的下边界≥上边界，则该表项视为OFF。

假设希望表项0的地址空间为(0,0x1100000)，

反推PMPADDR0 = (0x1100000 >>7) <<5 = 0x1100000>>2 = 0x440000。

1.4.1.3 NAPOT

在NAPOT模式中，PMPADDR高18位有效，即PMPADDR[3:0] 不参与地址匹配逻辑运算。

各表项的受控下边界和上边界均由PMPADDR决定。

假设PMPADDRi 中从bit[0] 到bit[31] 第一次出现比特位值为0 的是第n 位， 即PMPADDRi[n]， 则该表项基址为{PMPADDRi[29:n+1],(n+3){1’b0}}， 空间大小为$2^{(n+3)}$ 字节。

以PMPADDRi = 32’b aaaa_aaaa_aaaa_aaaa_aaaa_aaaa_aaa0_1111 为例，n = 4，

其基址为 32’ b aaaa_aaaa_aaaa_aaaa_aaaa_aaaa_a000_0000，空间大小为 $2^{(4+3)}$= 128B。

NAPOT模式中，各表项受控地址可以是非连续的，且最小划分颗粒为128B。

1.4.2 PMP寄存器

1.4.2.1 PMPCFGx （x=0,1）

物理内存保护设置寄存器，一个寄存器控制4个表项。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PMP(4x+3)CFG_R	PMP(4x+3)CFG_W	PMP(4x+3)CFG_X	PMP(4x+3)CFG_A		RSVD		PMP(4x+3)CFG_L	PMP(4x+2)CFG_R	PMP(4x+2)CFG_W	PMP(4x+2)CFG_X	PMP(4x+2)CFG_A		RSVD		PMP(4x+2)CFG_L	PMP(4x+1)CFG_R	PMP(4x+1)CFG_W	PMP(4x+1)CFG_X	PMP(4x+1)CFG_A		RSVD		PMP(4x+1)CFG_L	PMP(4x)CFG_R	PMP(4x)CFG_W	PMP(4x)CFG_X	PMP(4x)CFG_A		RSVD		PMP(4x)CFG_L
0	0	0	0	0	0	0	0	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

Name	Bit	Type	Description
PMP(4x+3)CFG_R	[31]	RW	可读属性： 0：该区域不可读 1：该区域可读
PMP(4x+3)CFG_W	[30]	RW	可写属性： 0：该区域不可写 1：该区域可写
PMP(4x+3)CFG_X	[29]	RW	可执行属性： 0：该区域不可执行 1：该区域可执行
PMP(4x+3)CFG_A	[28:27]	RW	地址匹配模式： 00：表项无效，即不使能，所有访问不会命中该表项 01：TOP（Top of Range）模式，使用相邻表项的地址作为匹配区间 10：不支持 11：NAPOT（Naturally Aligned Power Of Two），使用2 的幂次方大小作为匹配空间
PMP(4x+3)CFG_L	[24]	RW	锁控制： 0：该表项的物理内存保护设置寄存器和地址寄存器都可以修改，且机器模式下的访问权限不受R/W/X位的限制 1：该表项所有内容，包括L 位，在CPU 复位前都不可以修改。且机器模式下的访问和用户模式下的访问都将受到R/W/X 位的限制；当该表项的物理内存保护设置寄存器配置为TOR 地址匹配模式时，其前一表项的物理内存保护地址寄存器也无法修改。
PMP(4x+2)CFG_R	[23]	RW	同上
PMP(4x+2)CFG_W	[22]	RW	
PMP(4x+2)CFG_X	[21]	RW	
PMP(4x+2)CFG_A	[20:19]	RW	
PMP(4x+2)CFG_L	[16]	RW	
PMP(4x+1)CFG_R	[15]	RW	同上

PMP(4x+1)CFG_W	[14]	RW	
PMP(4x+1)CFG_X	[13]	RW	
PMP(4x+1)CFG_A	[12:11]	RW	
PMP(4x+1)CFG_L	[8]	RW	
PMP(4x+0)CFG_R	[7]	RW	同上
PMP(4x+0)CFG_W	[6]	RW	
PMP(4x+0)CFG_X	[5]	RW	
PMP(4x+0)CFG_A	[4:3]	RW	
PMP(4x+0)CFG_L	[0]	RW	

1.4.2.2 PMPADDRx（x=0~7）

物理内存保护地址设置寄存器，一个寄存器控制一个表项。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addr																															
0	0	0	0	0	0	0	0	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
addr	[31:0]	RW	每个表项中地址的意义由对应表项的地址匹配模式决定。
每个PMPADDR对应一个表项。			

1.5 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

1.5.1 CPU特殊功能寄存器表

Table 1-2 CPU SFR 表

Address	Function Description
0xE081_0000 to 0xEFFF_FFFF	Reserved
0xE080_0000 to 0xE080_FFFF	CLIC控制器
0xE010_0000 to 0xE07F_FFFF	Reserved

1.5.2 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_6000	通用IO端口-D (GPIO D)
	0x6000_4000	通用IO端口-C (GPIO C)
	0x6000_2000	通用IO端口-B (GPIO B)
	0x6000_0000	通用IO端口-A (GPIO A)
DAC	0x5020_0000	DAC0控制器
DMA	0x4031_0000	DMA1控制器
	0x4030_0000	DMA0控制器
CRC	0x4010_0000	CRC控制器
CAN	0x400F_1000	STAMP定时器 (STAMP)
	0x400F_0000	CAN控制器 (CAN)
OPA	0x400E_3000	运算放大控制器 (OPA3)
	0x400E_2000	运算放大控制器 (OPA2)
	0x400E_1000	运算放大控制器 (OPA1)
	0x400E_0000	运算放大控制器 (OPA0)
CMP	0x400D_2000	比较器控制器 (CMP2)
	0x400D_1000	比较器控制器 (CMP1)
	0x400D_0000	比较器控制器 (CMP0)
SIO	0x400B_1000	通用串行接口 (SIO1)
	0x400B_0000	通用串行接口 (SIO0)
I2C	0x400A_0000	I2C串行接口 (I2C)
SPI	0x4009_1000	同步并行接口 (SPI1)
	0x4009_0000	同步并行接口 (SPI0)
UART	0x4008_4000	通用异步收发器2 (UART2)
	0x4008_3000	通用异步收发器1 (UART1)
	0x4008_2000	通用异步收发器0 (UART0)
USART	0x4008_1000	通用同步异步收发器1 (USART1)
	0x4008_0000	通用同步异步收发器0 (USART0)
LED	0x4007_0000	LED驱动控制器 (LED)
LP_TC	0x4006_2000	窗口型看门狗定时器 (WWDT)
	0x4006_1000	低功耗定时器 (LPT)
	0x4006_0000	实时时钟定时器 (RTC)
	0x4005_E000	通用型可编程定时器/计数器 (GPTB5)

	0x4005_D000	通用型可编程定时器/计数器 (GPTB4)
	0x4005_C000	通用型可编程定时器/计数器 (GPTB3)
	0x4005_B000	通用型可编程定时器/计数器 (GPTB2)
	0x4005_A000	通用型可编程定时器/计数器 (GPTB1)
	0x4005_9000	通用型可编程定时器/计数器 (GPTB0)
	0x4005_8000	通用型可编程定时器/计数器 (GPTA3)
	0x4005_7000	通用型可编程定时器/计数器 (GPTA2)
	0x4005_6000	通用型可编程定时器/计数器 (GPTA1)
	0x4005_5000	通用型可编程定时器/计数器 (GPTA0)
	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	计数器A (COUNTER A)
FVR	0x4003_2000	FVR控制器 (FVR)
ADC	0x4003_1000	模数转换器 (ADC1)
	0x4003_0000	模数转换器 (ADC0)
SYSTEM	0x4001_2000	事件触发控制器 (ETCB)
	0x4001_1000	系统控制器 (SYSCON)
	0x4001_0000	闪存控制器 (IFC)
	0x4000_0000	设备信息寄存器 (Device ID)

2 异常和中断

2.1 概述

异常处理的关键是在异常发生时，保存CPU 当前指令运行的状态，在退出异常处理时恢复异常处理前的状态。异常能够在指令流水线的各个阶段被识别，并使后面的指令不会改变进入前CPU的状态。RISC-V的异常处理包括指令异常和外部中断。指令异常可以是非法指令、非对齐访问错误等，外部中断可以是外部设备的中断请求等。为了方便描述，下文用异常代替指令异常，用中断代替外部中断。

异常和中断在指令的边界上被处理，即CPU 在指令退休时才会响应异常或中断，并保存退出时下一条被执行的指令的地址。即使异常或中断指令在某一条指令退休前被识别，也要在相应的指令退休时才会被处理。

E906内核兼容RISV-V标准的异常和中断向量号。向量表如下图所示：

2.1.1 异常向量表

Table 2-1 System Exception Vectors

Vector Number		Exception Sources
0	RSVD	Reserved
1		取指令访问错误异常
2		非法指令异常
3		调试断点异常
4		加载指令非对齐访问异常
5		加载指令访问错误异常
6		存储指令非对齐访问异常
7		存储指令访问错误异常
8		用户模式环境调用异常
9	NA	未实现
10	RSVD	保留
11		机器模式环境调用异常
12~23	NA	未实现/保留
24		NMI
>=25	NA	未实现/保留

2.1.2 中断向量表

Table 2-2 System Interrupt Vectors

Vector Number		Interrupt Sources
0	RSVD	Reserved
1	RSVD	Reserved
2	RSVD	Reserved
3	SWI	CPU software interrupt
4	RSVD	Reserved
5	RSVD	Reserved
6	RSVD	Reserved
7	SYSTICK	CPU Systick interrupt
8	RSVD	Reserved
9	RSVD	Reserved
10	RSVD	Reserved

11	RSVD	Reserved
12	RSVD	Reserved
13	RSVD	Reserved
14	RSVD	Reserved
15	RSVD	Reserved
16	SYSCON	System controller interrupt
17	IFC	Internal flash controller interrupt
18	ADC0	ADC0 controller interrupt
19	ADC1	ADC1 controller interrupt
20	DMA0	DMA0 interrupt
21	DMA1	DMA1 interrupt
22	EXI0	External interrupt GROUP0, GROUP16
23	EXI1	External interrupt GROUP1, GROUP17
24	GPTA0	GPTA0 Interrupt
25	GPTA1	GPTA1 Interrupt
26	GPTA2	GPTA2 Interrupt
27	GPTA3	GPTA3 Interrupt
28	GPTB0	GPTB0 Interrupt
29	GPTB1	GPTB1 Interrupt
30	GPTB2	GPTB2 Interrupt
31	GPTB3	GPTB3 Interrupt
32	GPTB4	GPTB4 Interrupt
33	GPTB5	GPTB5 Interrupt
34	DAC0	DAC0 interrupt
35	RSVD	Reserved
36	USART0	USART 0 interrupt
37	USART1	USART 1 interrupt
38	UART0	UART0 interrupt
39	UART1	UART1 interrupt
40	UART2	UART2 interrupt
41	RSVD	Reserved
42	SIO0	SIO0 Interrupt
43	SIO1	SIO1 Interrupt
44	I2C	I2C interrupt
45	SPI0	SPI0 interrupt
46	SPI1	SPI1 interrupt

47	CAN	CAN interrupt
48	EXI2	External Interrupt GROUP2 ~ 3, GROUP18~19
49	EXI3	External Interrupt GROUP4 ~ 9
50	EXI4	External Interrupt GROUP10 ~ 15
51	CNTA	COUNTER A interrupt
52	LPT	Low power timer interrupt
53	WWDT	Window watchdog timer interrupt
54	RTC	Real timer clock interrupt
55	CMP0	CMP0 interrupt
56	CMP1	CMP1 interrupt
57	CMP2	CMP2 interrupt
58	LED	LED controller interrupt
59	RSVD	Reserved
60	BT0	BT0 interrupt
61	BT1	BT1 interrupt
62	BT2	BT2 interrupt
63	BT3	BT3 interrupt

中断向量号，是请求在异常表的位置编号。

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。

2.2 异常

异常的种类很多，比如访问非法的地址空间时，会进入加载指令访问错误异常。RISC-V 编程模型中没有异常使能寄存器，因此一旦触发异常即会被响应。所有的异常入口地址都是统一的，异常响应时的跳转执行入口均由MTVEC寄存器指定，因此在CPU 跳转到该入口执行程序时，软件可依据MCAUSE寄存器中的异常向量号来决定是否在异常服务程序中实现再次跳转到各自对应的服务程序进行处理。

对于非对齐访问异常，根据RISC-V标准定义，当取指或者内存数据读写操作地址不对齐时会上报相应的非对齐访问异常。因此当访存的 数据位宽和地址不对齐时按照 RISC-V 标准需产生加载指令或者存储指令非对齐访问异常。例如对于双精度浮点来说，双精度内存加载或者存储地址需要是双字对齐。为了降低软件编程复杂度，E906 扩展实现了非对其访问异常掩码开关，由扩展状态寄存器 MXSTATUS中的 MM 位来控制非对齐内存访问是按照标准定义触发非对齐访问异常抑或不上报异常，而是硬件自动做拆分访问处理。

2.2.1 NMI

NMI（不可屏蔽的外部中断）是一个特殊的异常。因为它处于异常区，所以它不受全局中断使能位MSTATUS.MIE的控制。NMI作为外部事件，E906 将其异常向量号定义为24，优先级在所有的中断和异常中最高。

为了使得任何时候响应NMI 并返回之后，CPU 都可以恢复正常运行程序流，NMI有自己的状态寄存器和异常程序

计数器，NMI服务程序中也使用了独立的的栈指针。

在NMI 的服务程序中如果触发了异常（非ECALL或EBREAK主动触发的异常），CPU 会进入锁定状态，此时会触发CPU核进行复位。SYSCON_RSR中相应复位源标志位会置起。

在APT32F173中，可以升级为NMI的外部中断源有：

- 外部时钟检测错误
- EXI0
- 内存（Flash，SRAM）校验错误
- 掉电检测

上述中断源的NMI使能后，不会影响原有的中断通路，比如EMCM的中断通路也是有效的，当然NMI的优先级是最高的，所以会以NMI的形式请求中断。

2.2.2 优先级

按照RISC-V 标准定义，异常优先级低于中断。异常内部优先级定义如表4.2 所示。E906 将NMI 的异常向量号定义为24，并将其设置为所有中断和异常中优先级最高的异常类型。即：NMI > 中断 > 异常

Vector Number	Priority	Exception Sources
3	高 ↓ 低	调试断点异常
1		取指令访问错误异常
2		非法指令异常
8		用户模式环境调用异常
11		机器模式环境调用异常
6		加载指令非对齐访问异常
4		存储指令非对齐访问错误异常
7		存储指令访问异常
5		加载指令访问异常

2.3 中断

E906内核设计实现了

- RISC-V标准的CLINT中断，包括软件中断、计时器中断、外部中断。支持中断咬尾、中断晚到和中断嵌套。
- RISC-V标准的CLIC中断控制器，仅用于对中断源进行采样、优先级仲裁和分发。

E906中的CLIC中断控制器 兼容CLIC SPEC-0.8 版本，每个中断都有1个专用32位寄存器，内容包括：

- IP：中断等待状态
- IE：中断使能控制
- ATTR：中断触发方式（电平/上升沿/下降沿）和是否为硬件矢量中断。
- CTL：参与仲裁的优先级

在CLIC 中只有符合条件的中断源才会参与仲裁。需满足的条件如下：

- 中断源处于等待状态（IP =1）；
- 中断优先级大于MINTTHRESH 域值寄存器设定的域值（中断优先级非零才可正常参与仲裁）；
- CLIC 中该中断使能位为1（IE=1）。

2.3.1 特性

- 支持64个中断源（IRQ[63:0]）
- 每个中断源具有独立的中断向量号
- 每个中断源具有独立可编程的中断优先级设置、中断类型、中断触发方式、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

2.3.2 硬件矢量中断和非硬件矢量中断

每个中断都可以通过其CLICINT[SHV]独立设置为硬件矢量中断和非硬件矢量中断。两者的区别在于

- 对于非矢量中断而言，中断服务程序入口都是统一由 MTVEC寄存器指定，从该地址取回的数据即为中断服务程序的第一条指令。而硬件矢量中断则由 MTVT 加中断号偏移量指定。
- 非矢量中断响应时，在任何中断触发模式下，CPU不会主动清除CLIC内对应的中断pending位，需要软件在中断服务程序中使用读MNXTI寄存器的方式触发pending位的清除；而矢量中断响应时，CPU会主动清除CLIC内中断pending位。

2.3.3 中断处理

内核中断能控制器（CLIC）协同其外围逻辑，用于中断的高效处理。每个中断源拥有独立的软件可编程优先级。只有当优先级高于中断阈值寄存器 MINTTHRESH[MTH]的值时，才能向处理器发起中断。

CLIC 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。实际操作时，如果需要嵌套响应中断，需要对当前的 MSTATUS，MCAUSE，MINTSTATUS 等寄存器进行保存并重新打开 MSTATUS 中的中断使能位。这些操作均属于真正的中断处理操作之前的必须操作。

CLIC 支持独立的软件中断使能设置。为了保证中断被正常响应，必须保证全局中断使能位 MSTATUS[MIE]为 1 以及该中断对应的中断使能寄存器 CLINCINTx 中的使能位为 1。

CLIC 中断控制器支持软件中断，软件可以通过设置软件中断配置寄存器（CLIC_MSIP）置高相应的中断等待状态位，向 CPU 发送中断请求。

中断整个响应、处理、返回的流程如下图所示。

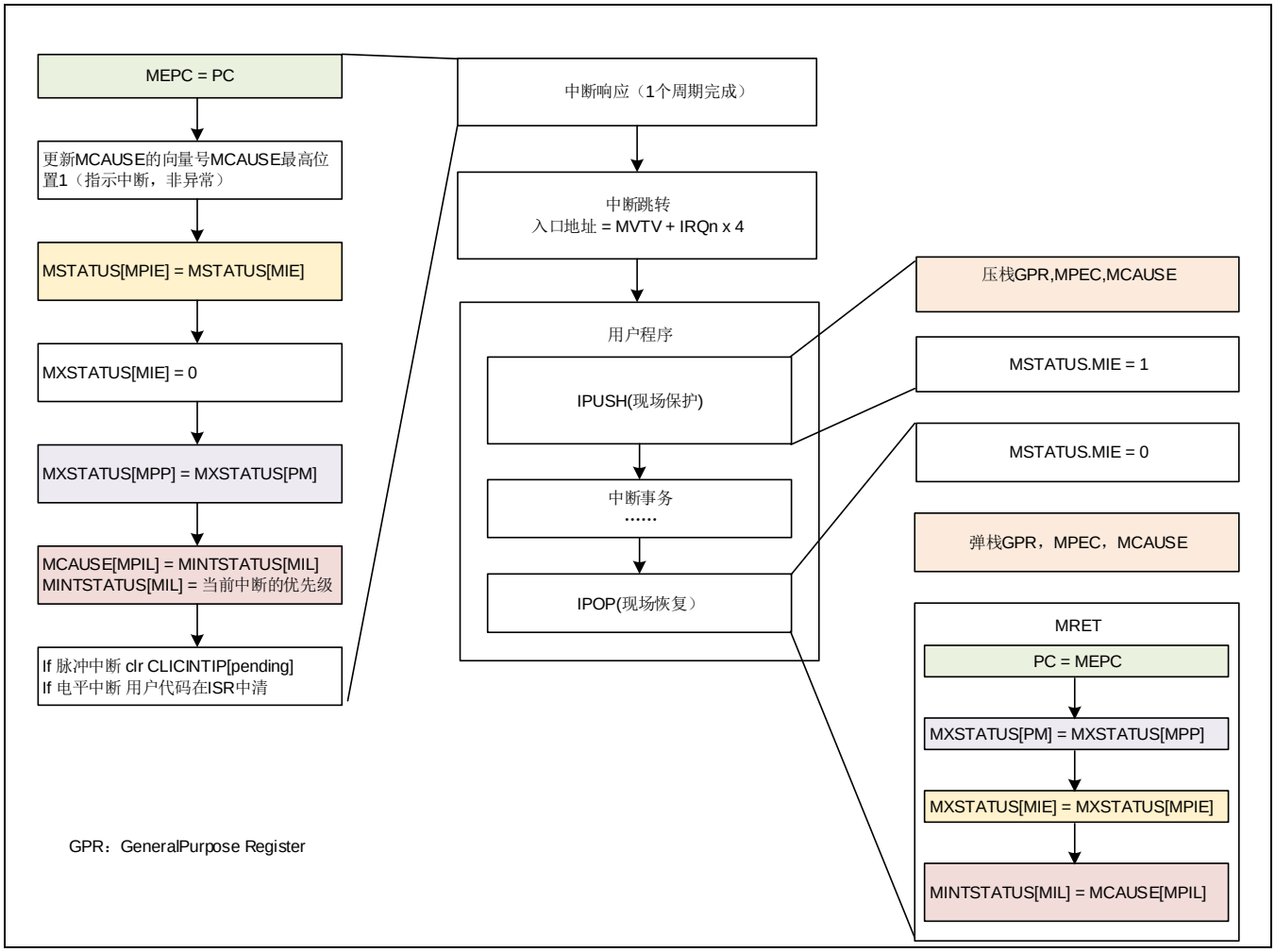


Figure 2-1 中断响应、处理、返回流程

2.3.4 中断优先级

每个中断源都可以通过 CLICINTx[INTCTL]（x 代表不同的中断源）独立设置中断优先级。APT32F173 实现支持 16 个优先级设置，值越大优先级越高。只有当优先级高于中断阈值寄存器 MINTTHRESH[MTH]的值时，才能向处理器发起中断。优先级为 0 的中断不参与仲裁，即为无效中断。

当CLIC 中有多个中断处于等待状态时，CLIC 仲裁出优先级最高的中断。优先级为0 的中断不参与仲裁；如多个中断拥有相同的优先级，则中断ID 较大的优先处理。CLIC 支持中断优先级的动态调整。

2.3.5 中断嵌套

CLIC 支持中断嵌套。IPUSH 指令的最后会自动开启全局中断能使能。如果不使用中断嵌套，可以将所有中断的优先级设置为相同，或在中断服务程序中禁止全局中断。

2.3.6 中断咬尾

CLIC支持中断咬尾。当前中断事务处理完成后，处理器在现场弹栈之前，会对CLIC 中等待处理的较低优先级中断（如果存在的话）进行查询，如果该优先级比进入中断服务程序前的中断优先级高，会跳过刚完成的中断服务程序的弹栈和下一个中断服务程序的压栈过程。所以中断咬尾功能可以加速中断的执行效率。

2.3.7 中断晚到

CLIC支持中断晚到。如果对某一中断的响应还处于早期入栈阶段，如果此时收到了更高优先级中断的请求，则入栈操作后即跳转到晚到高优先级中断服务程序中。

2.3.8 中断唤醒

当CPU处于低功耗模式模式时，外设产生的中断可以将CPU从低功耗模式唤醒。.

2.3.9 中断操作步骤

中断的配置基本分为两个级别，一个是外设的配置，另外一个CLIC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CLIC内部的中断控制，设置CLICINT，设置中断使能、优先级等，
- 开启CPU全局中断使能控制，MSTATUS[MIE] = 1

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CLIC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CLIC中的pending位在处理器响应中断请求后，会自动清除。

CLIC为每个中断源提供状态查询，可以通过查询CLICINT[IP]，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。

2.4 寄存器说明

2.4.1 寄存器表

Base Address of INTERRUPT: 0xE0800000

Register	Offset	Description	Reset Value
MINTTHRESH	0x0008	中断阈值设置寄存器	0x00000000
CLICINTx	0x1000~0x1100	中断x控制寄存器(x=0~63)	0x0FC00000

2.4.2 MINTTHRESH(中断阈值设置寄存器)

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
MTH								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	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
MTH	[31:24]	RW	中断阈值。CPU只响应高于此阈值的中断。

2.4.3 CLICINTx(中断x控制寄存器(x=0~63))

Address = Base Address+ 0x1000~0x1100, Reset Value = 0x0FC00000

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

Name	Bit	Type	Description
INT_CTL	[31:24]	RW	参与仲裁的优先级
MODE	[23:22]	RW	中断特权态： E906该位段为 3h。
TRIG	[18:17]	RW	中断触发方式： x0b：电平中断 01b：上升沿中断 11b：下降沿中断
SHV	[16]	RW	硬件矢量中断设置： 0h：非硬件矢量中断 1h：硬件矢量中断
IE	[8]	RW	该中断使能状态 0h：该中断没有使能 1h：该中断使能
IP	[0]	RW	该中断等待状态 0h：该中断没有处于等待状态 1h：该中断处于等待状态

注：INT_CTL参与仲裁的优先级:有对应中断，仲裁的优先级为0x0f,未实现的为0。

MODE中断特权态：E906该位段有对应的中断为3h,未实现的为0。

3 CRC

3.1 概述

本章节描述的是用来纠错的CRC引擎。该模块支持常用的CRC标准。

3.1.1 特性

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

3.2 功能描述

3.2.1 模块框图

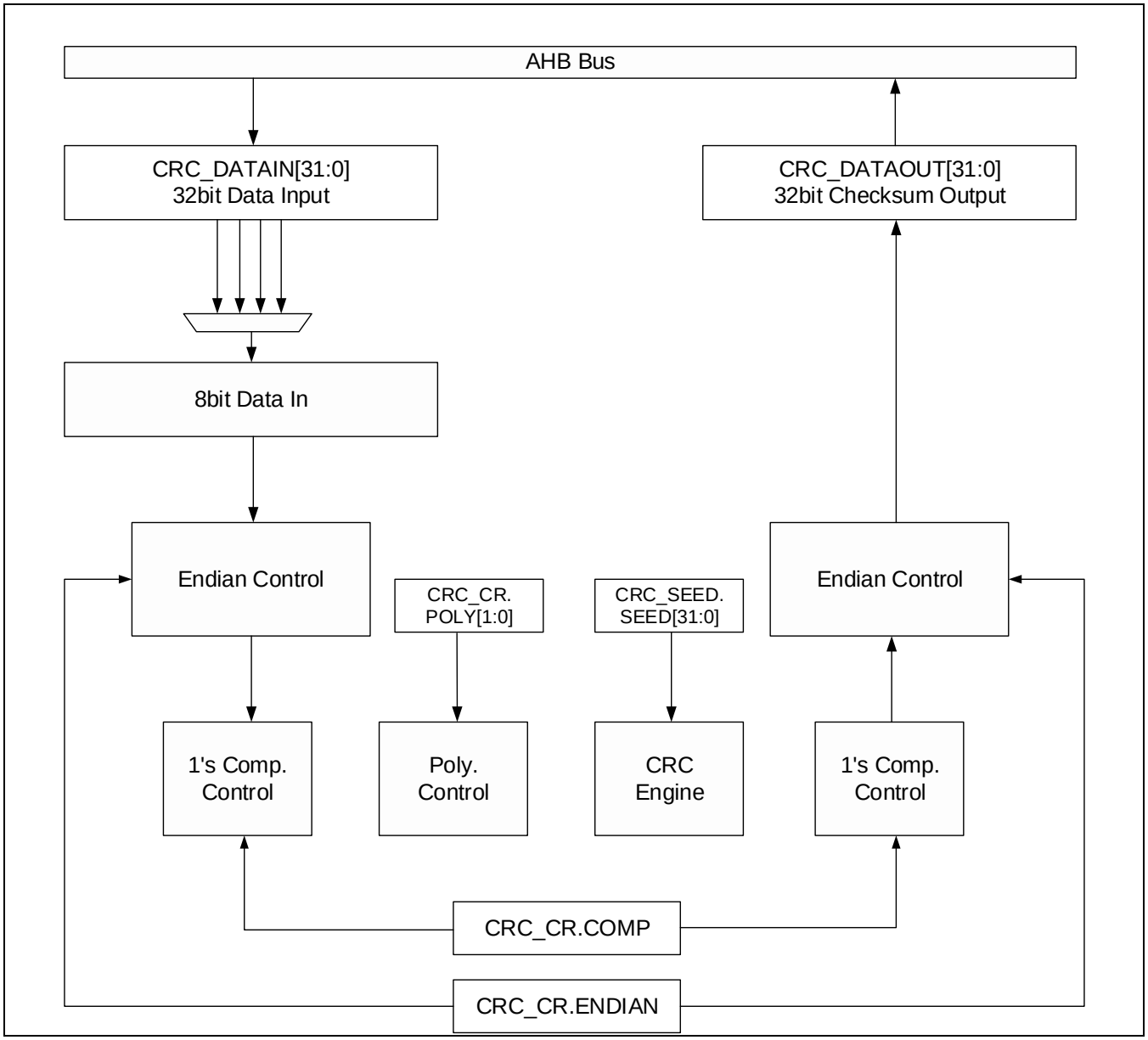


Figure 3-1 CRC模块框图

3.2.2 功能说明

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

3.3 寄存器说明

3.3.1 寄存器表

Base Address of CRC: 0x40100000

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

3.3.2 CRC_CEDR(时钟使能/禁止寄存器)

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

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

Name	Bit	Type	Description
CKEN	[0]	RW	CRC 引擎使能：0：禁止1：使能

3.3.3 CRC_SRR(软件复位寄存器)

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

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

Name	Bit	Type	Description
SWRST	[31]	W	软件复位写1会复位该模块

3.3.4 CRC_CR(控制寄存器)

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

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

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

3.3.5 CRC_SEED(种子值寄存器)

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

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

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

3.3.6 CRC_DATAIN(输入数据寄存器)

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

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

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

3.3.7 CRC_DATAOUT(输出数据寄存器)

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

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

Name	Bit	Type	Description
DATAOUT	[31:0]	R	该寄存器保存CRC计算结果。

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。APT32F173 系列片上带有 128K 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。APT32F173 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: 128K Bytes
- 数据闪存(DROM)大小: 8K Bytes
- 编程支持ISP模式和专用的工具模式
- 页大小: 256 Bytes(PROM), 64 Bytes(DROM)
- 可擦除单元: 页
- 可靠性: PROM和DROM都为100,000次?
- 可自定义的选项(称为User Option)支持iWDT使能和禁止，配置复位管脚
- 支持各种保护：调试接口保护，硬件保护和读保护

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：

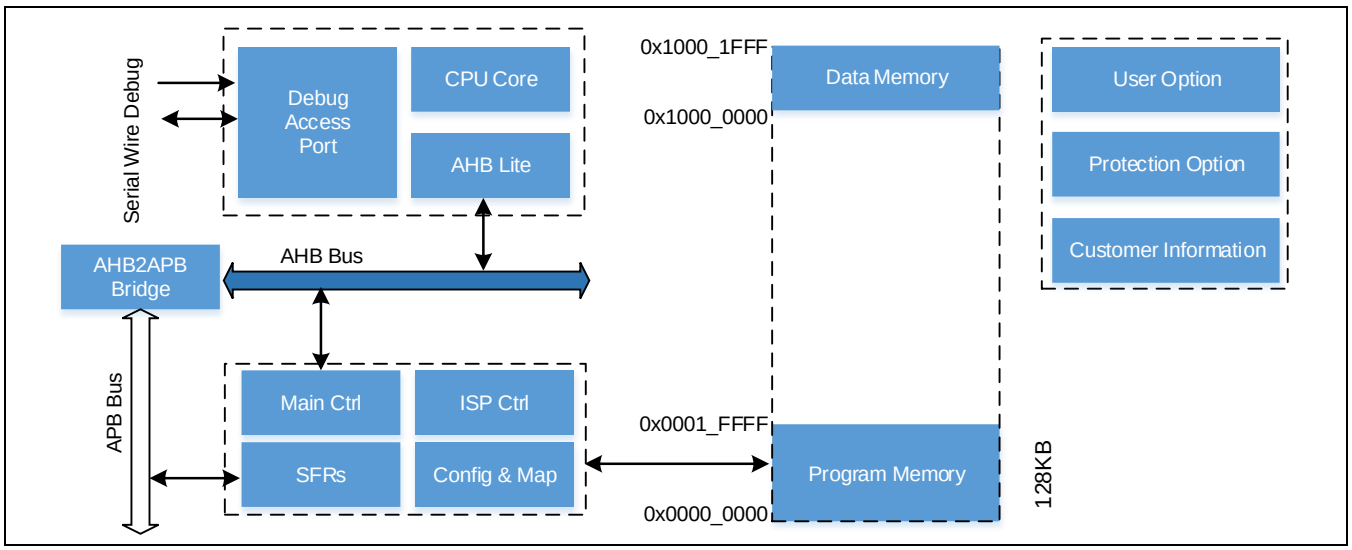


Figure 4-1 IFC模块框图

4.2.2 模块结构

APT32F173 系列闪存由程序存储单元(PROM)，数据存储单元(DROM)，用户配置单元(User Option)，保护选项和客户信息区域构成。PROM 有 256 个页空间，每页有 256 字节。最小的擦除和烧写单元为页空间，用户只可以对整个页空间进行擦除或者烧写，不能擦除或者烧写某个指定的字(Word)。

区域	页名称	大小	起始地址	结束地址
PROM 128KB	Page 0	256B	0x0000_0000	0x0000_00FF
	Page 1	256B	0x0000_0100	0x0000_01FF
	Page 2	256B	0x0000_0200	0x0000_02FF
	Page 3	256B	0x0000_0300	0x0000_03FF
	:	:	:	:
	Page 510	256B	0x0001_FC00	0x0001_FDFF
	Page 511	256B	0x0001_FE00	0x0001_FFFF
DROM	Page 0	64B	0x1000_0000	0x1000_007F

8KB	Page 1	64B	0x1000_0040	0x1000_00FF
	Page 2	64B	0x1000_0080	0x1000_017F
	Page 3	64B	0x1000_0100	0x1000_01FF
	:	:	:	:
	Page 126	64B	0x1000_1F00	0x1000_1F7F
	Page 127	64B	0x1000_1F80	0x1000_1FFF

Table 4-1 闪存地址映射

闪存结构如下图所示：

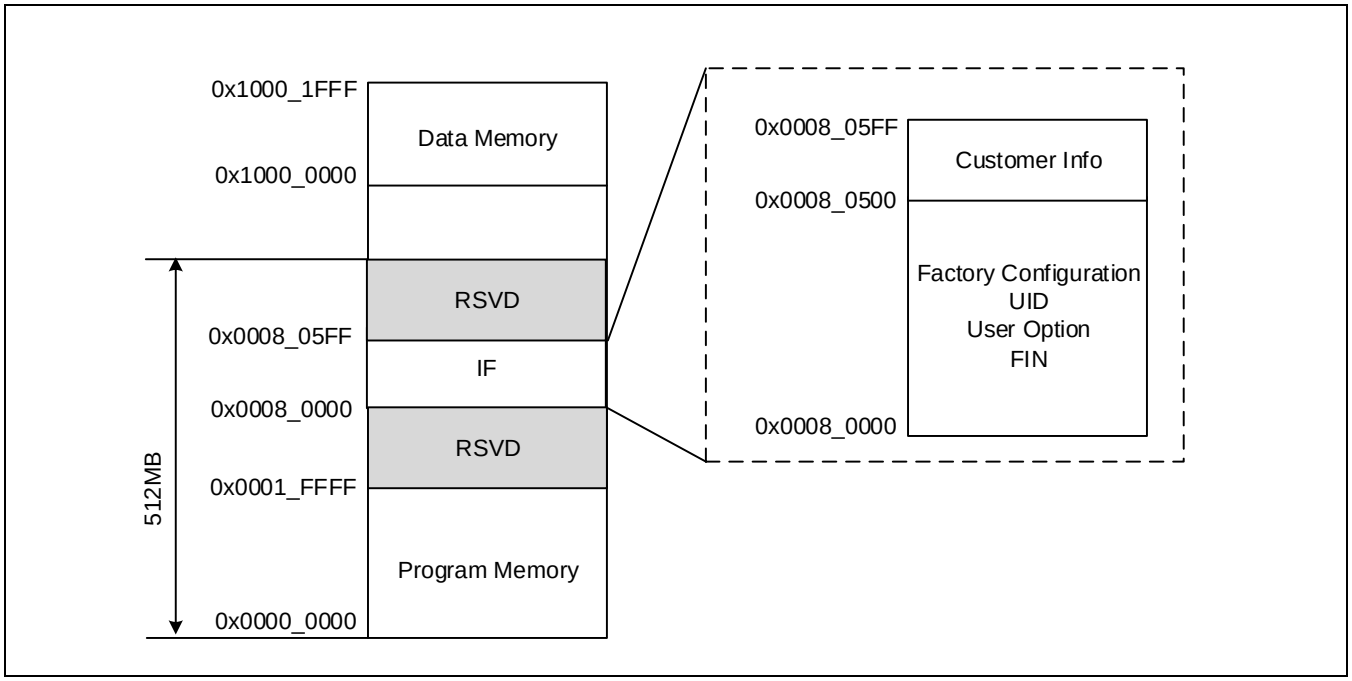


Figure 4-2 闪存地址空间结构

4.2.3 数据闪存

APT32F173 系列支持数据闪存，给用户存储普通数据。数据闪存可以通过 ISP 编程进行读写。擦除的最小单位为页。当需要改变某个字节时，该页中所有 64 个字节都需要提前复制到另一页里或者 SRAM 里暂存，或者使用特殊的软件算法在多页中轮循，模拟 EEPROM 的操作。尽管 PROM 也支持 ISP 功能，但为了数据的安全性和程序代码的完成性，我们强烈建议使用数据闪存空间来存放应用所需要存储的信息，而不是使用程序闪存。在进行全芯片擦除操作时，只有程序闪存的内容会被擦除掉。如果需要擦除所有数据闪存，需要借助烧录器。

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)

闪存中有一个信息区，里面包含 User Option，各种保护功能配置信息和给一些用户可以自定义的区间。

User Option、保护功能配置、UID 信息等内容在芯片上电后会被自动读取到相应的寄存器中。

即使芯片已经被焊在 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在上电时是否使能。其中除 EXTRST 和 EFLINIT 外，其他选择都可以通过软件更改配置。

该选项执行片擦除后为 0。

31	30	29	28	27	26	25	24	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				SRMRMP				EFLINIT				EFLECC				CPUFT				EXTRST			

名称	位	描述
EXTRST	[3:0]	外部复位管脚功能 **EXTRST[3:0]** / **功能** 0x5 / PD5为外部复位管脚，IO功能被禁用 其它值 / PD5禁用外部复位功能，当作IO使用
IWDT	[31:24]	0xAA: 禁用系统看门狗 IWDT 其它值: 使能系统看门狗 IWDT
SRMRMP	[19:16]	RSVD
EFLINIT	[15:12]	Flash Trim 值下载超时复位 0x5：超时复位 其他值：超时不复位
EFLCHK	[11:8]	Flash Check 默认状态 0x5：Enable 其他值：Disable
CPUFT	[7:4]	CPU Fault 复位 0xA：复位 其他值：不复位

Table 4-2 User Option功能

4.2.4.2 保护选项 (Protection)

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

- 硬件保护（HDP）

  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。

  HDP_FULL: 保护闪存全部PROM区域的内容

  HDP_32K: 保护从0x0地址开始的32K字节

  HDP_16K: 保护从0x0地址开始的16K字节

  IFC_4K: 保护从0x0地址开始的4K字节

  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)放在受保护区域内，然后使能该区域的硬件保护功能。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（RDP）

  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).

- 调试接口保护(SWDP)

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

- 代码加密保护（ENCRYPT）

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

虽然这个保护选项在程序 0x0000_0104 地址可以有映射，但建议该地址保持 0 值，量产时通过烧录器配置，以避免使能 SWDP 从而关闭调试接口。

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

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

Table 4-3 保护选项

注：HDP 硬件保护功能在该地址中无映射。

4.2.4.3 客户信息区 (Customer Information)

客户信息区域由 64 个字(256 字节)组成，可以根据客户所需存储应用 ID 或者序列号等等。这个区域不支持通过 ISP 编程，而且跟其它自定义选项一样在 CHIP ERASE 时会被擦除掉。该区域必须通过外部烧录工具进行烧写，读取可以直接通过访问该区域的地址(0x0008_0500 ~ 0x0008_05FF)直接读取。

4.2.4.4 工厂信息区 (Factory Information)

工厂信息区域由 3 个字(12 字节)组成，这个区域客户不能自己通过 ISP 功能或者烧录工具进行烧写，只能委托工厂在芯片出厂的时候一次性写入。工厂在写入后，该区域存储的内容不会被 ISP 或者烧录工具擦除。该区域只能通过 SYSCON 模块中相应的镜像寄存器 FINF0~FINF2 进行读取。

4.2.4.5 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。

	烧录器	程序代码里特殊地址	ISP 功能(IFC_CMR)
User Option	√	X	√
SWD 改口	√	X	√
保护选项	√	不建议 0x104 地址保持 0	√
客户信息区	√	X	X
工厂信息区	√	X	X
UID	√	X	X

Table 4-4 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。不同 CPU 频率下，WAIT 和 SPEED 的值参考值如下表。

SYSCLK	WAIT	SPEED
80MHz < SYSCLK ≤ 105MHz	4	1
64MHz < SYSCLK ≤ 80MHz	3	1
48MHz < SYSCLK ≤ 64MHz	2	1
16MHz < SYSCLK ≤ 48MHz	1	1
8MHz < SYSCLK ≤ 16MHz	0	1
SYSCLK ≤ 8MHz	0	0

Table 4-5 闪存读取的等待设置

从上表可以看出，

- 随着 CPU 工作频率的升高，需要更多的闪存等待周期，这将影响 CPU 执行指令的效率。为了提高效率，该产品的 CPU 支持 Icache 缓存，可以通过使能 Icache 缓存提高 CPU 指令的执行速度。

- 相同配置对应一个频率范围，在这个范围内，频率越高效率越高。

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
- 烧录工具 (专用串行接口)

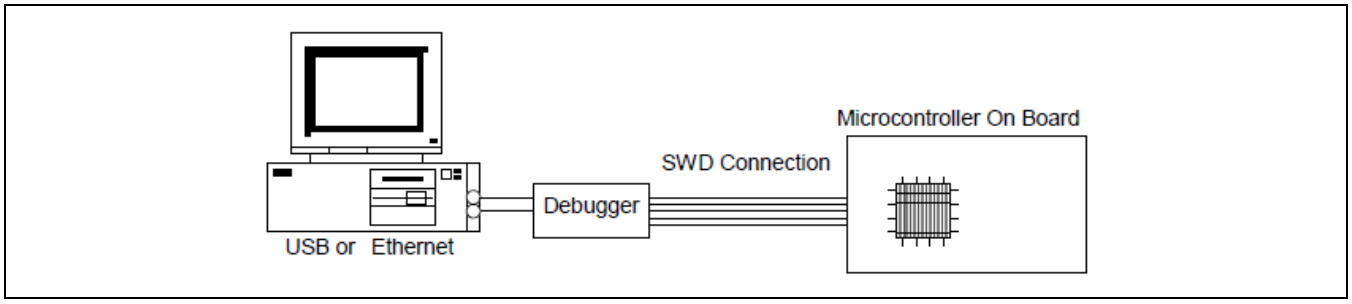


Figure 4-3 通过调试接口的烧写

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

APT 硬件烧录模式需要使用 5 根线作为烧写信号。烧写所需的信号如下表所示。

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

Table 4-6 闪存烧写信号

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

4.2.8 ISP功能

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

4.2.8.1 页擦除操作

每页闪存中有 256 (PROM) / 64 (DROM) 字节。页擦除操作会擦除 IFC_FM_ADDR 中地址所在的那一页闪存。在 ISP 操作前，用户必须将秘钥 0x5A5A_5A5A 写入 IFC_KEY 寄存器以禁止闪存模块的擦除/烧写保护。之后，用户需要将烧写的地址写入 IFC_FM_ADDR 寄存器，并将 IFC_CMR 里的指令 CMD[3:0]写为 0x2(页擦除操作)，最后将 IFC_CR 的 START 位置 1 启动该操作的执行。在 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 256 (PROM) / 64 (DROM) 字节的页进行操作。本闪存包含一个页缓存空间，在写操作中，需要先将整个页的数据先写入到页缓存空间中，再执行写操作的命令，将整个页缓存空间中的数据一起写入闪存中。

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

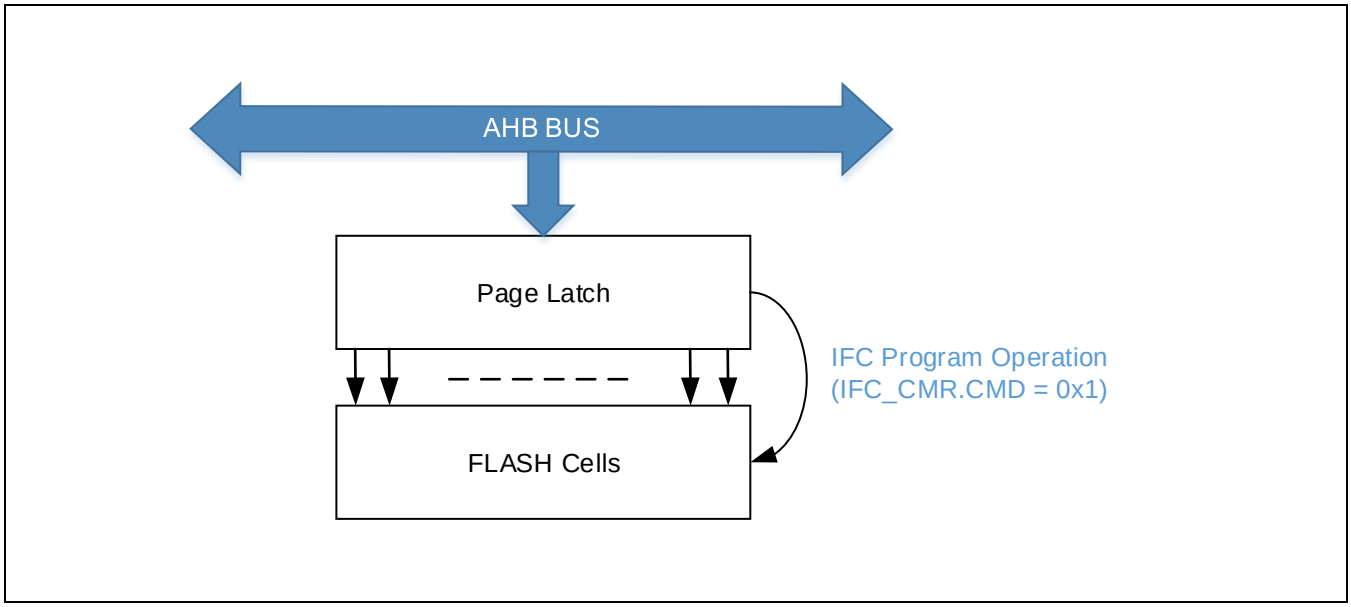


Figure 4-4 将页缓存数据写入闪存中

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 写闪存操作 - 并行模式

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

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

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

4.2.8.4 片擦除操作

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

。在 ISP 操作完成后，IFC_RISR 里的 END 位会置 1。用户同时也可以查询 IFC_CR 里的 START 位来判断 ISP 操作是否完成。

示例：

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

4.2.8.5 擦除自定义选项区域

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

示例：

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

4.2.8.6 烧写自定义选项操作

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

示例 (烧写 USER_OPTION)：

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

```
    CSP_IFC_SET_CR(IFC, START);                 // Start Program
    while ( CSP_IFC_GET_CR(IFC) != 0x0 );       // Wait for operation done

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

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

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

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

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

4.2.9 ECC校验

为了数据的可靠性，闪存控制器支持 ECC 校验功能。

4.2.10 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
ECC_ERR	ECC单bit错误；当读取数据的ECC校验发现有单bit错误(但被纠正)时，该位会置1

Table 4-7 中断源描述

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

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

4.2.11 闪存控制流程图

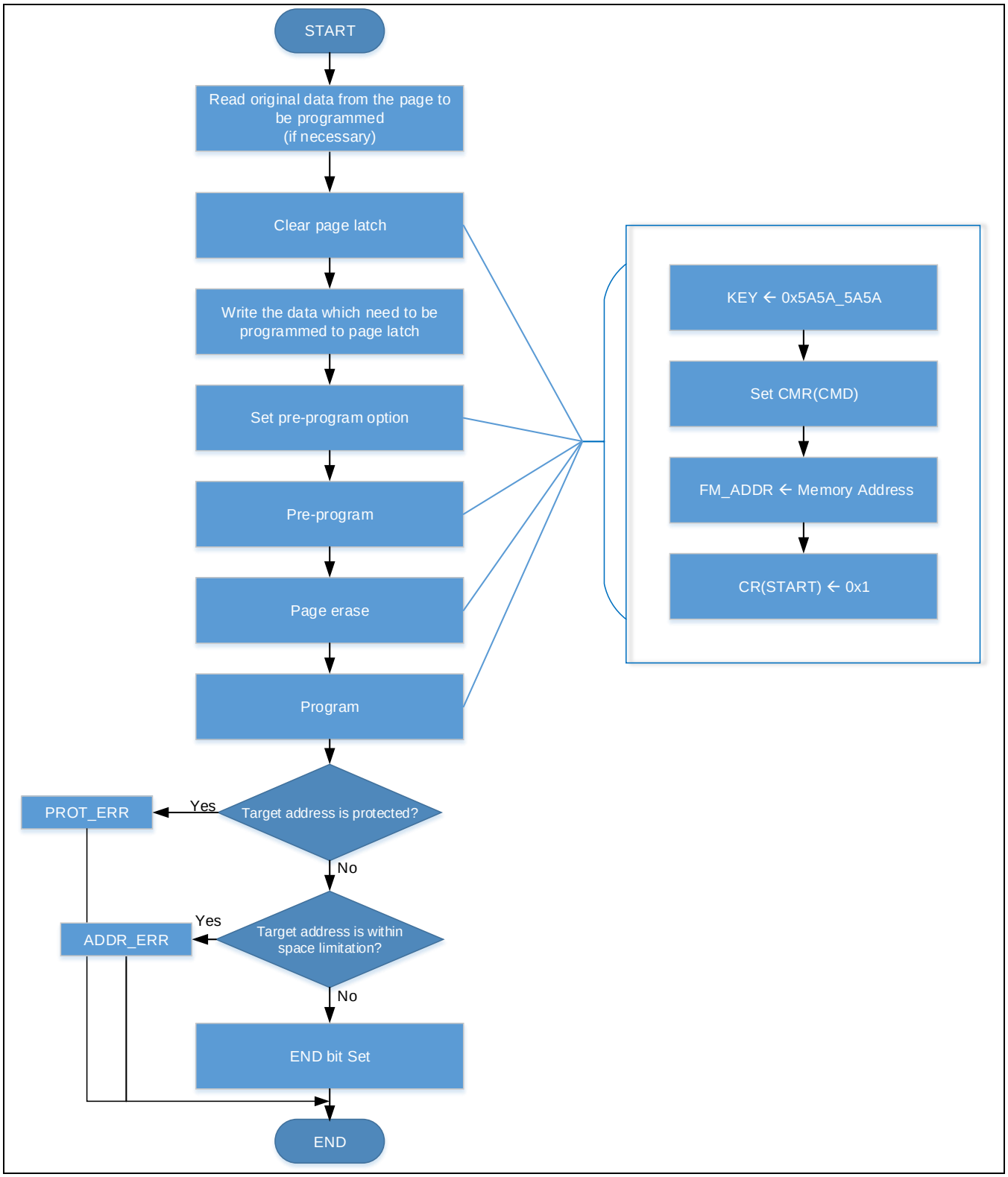


Figure 4-5 写操作流程图 – 普通模式

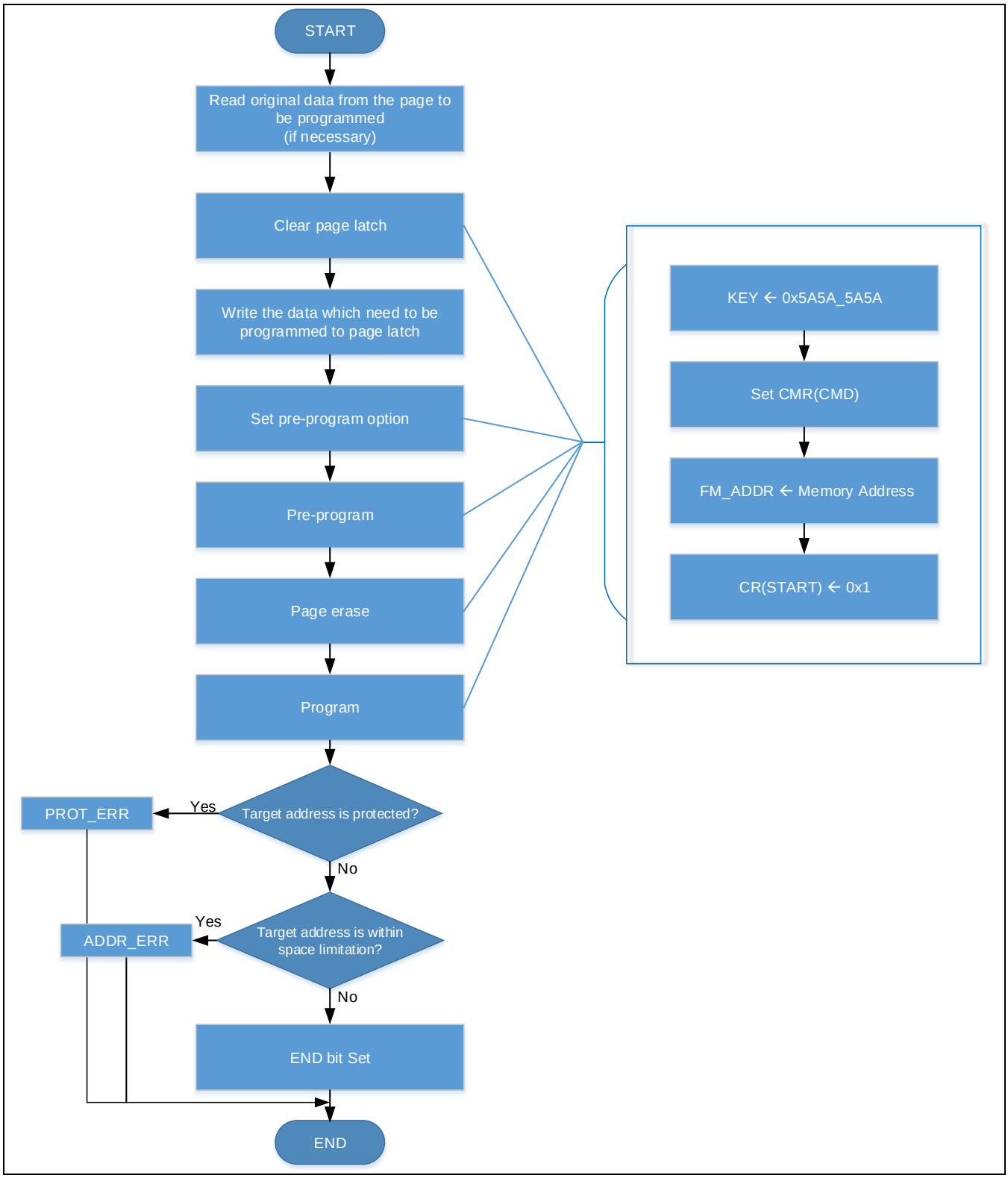


Figure 4-6 写操作流程图 – 并行模式

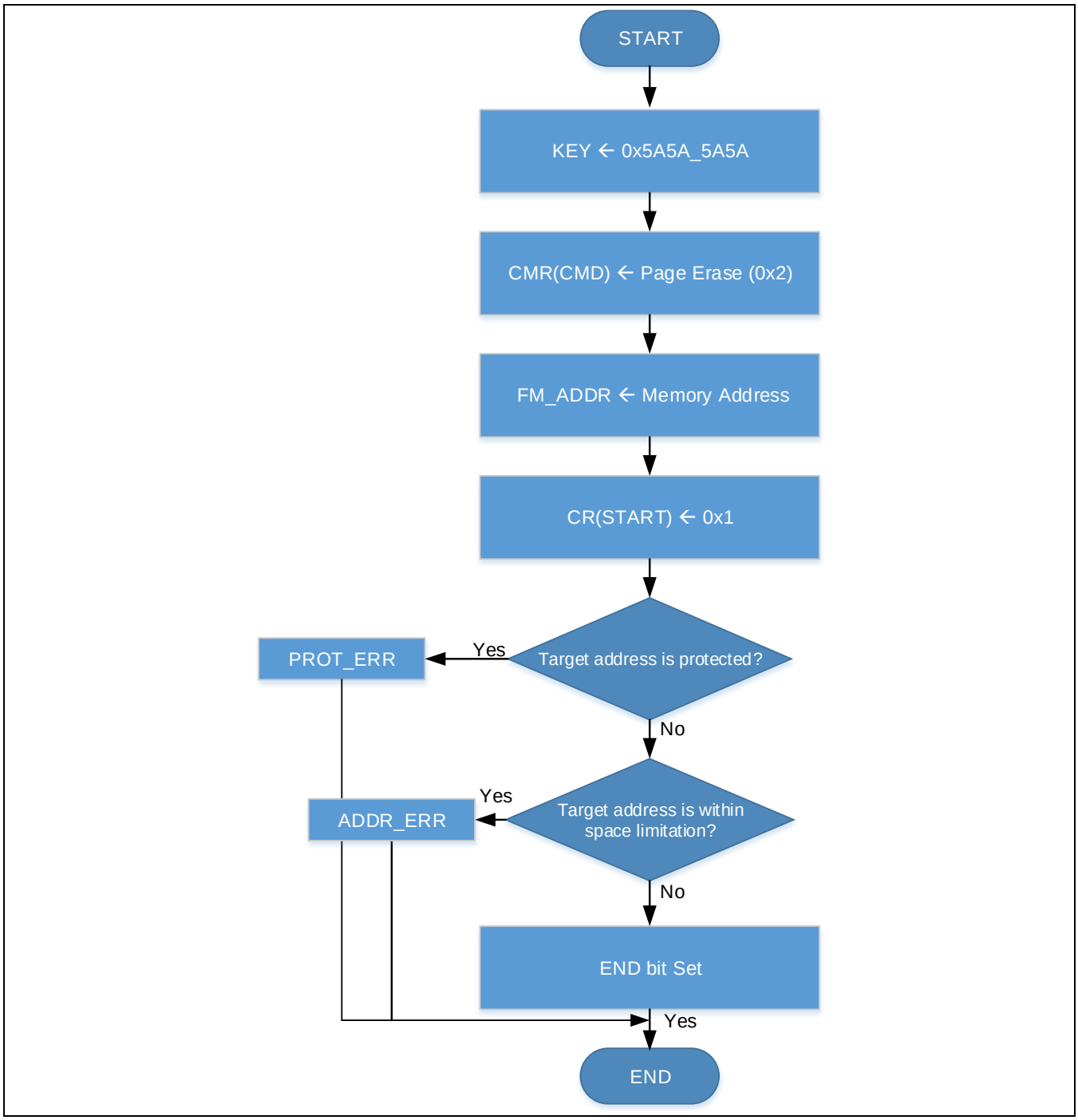


Figure 4-7 页擦除操作流程图

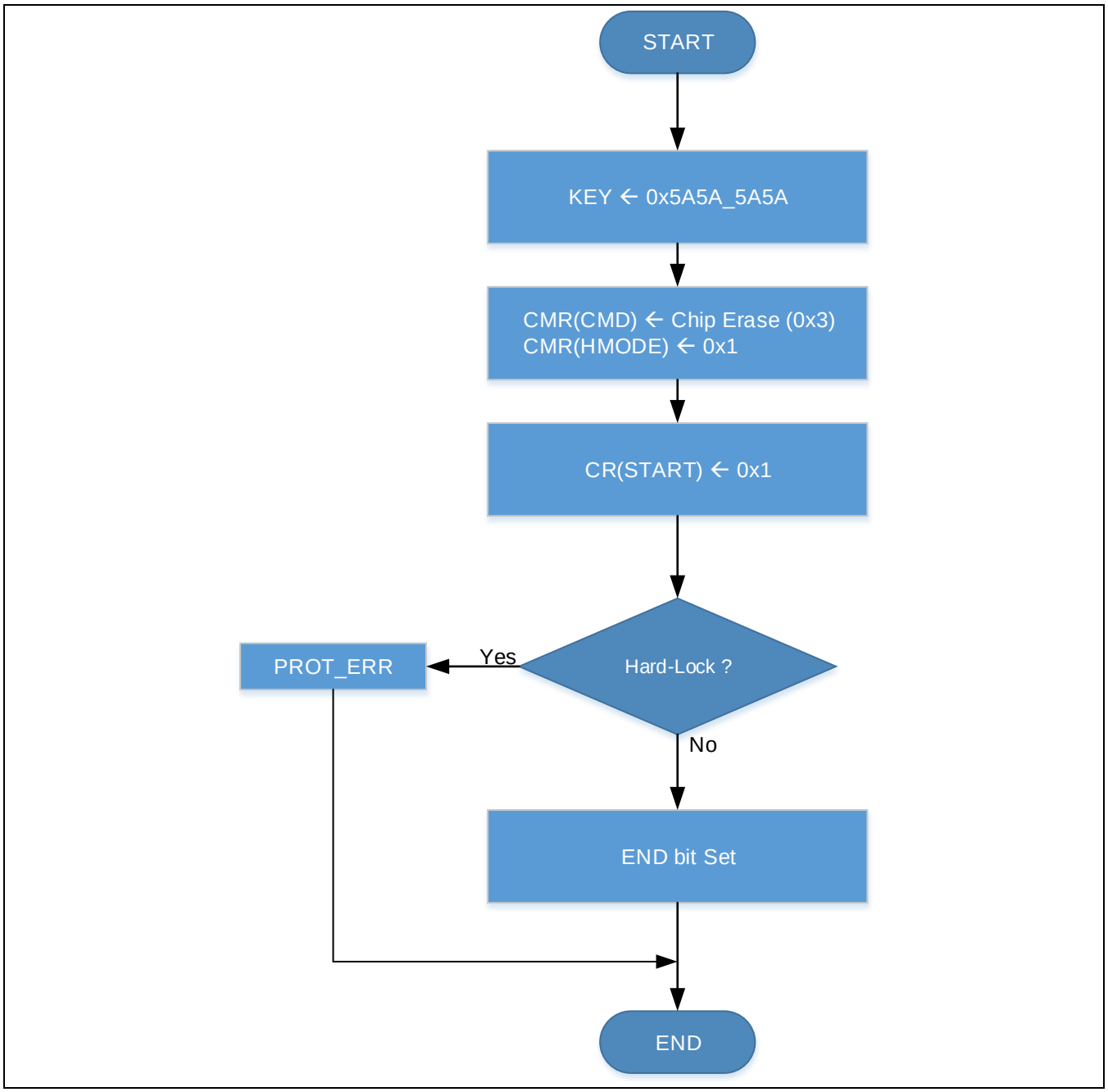


Figure 4-8 片擦除操作流程图

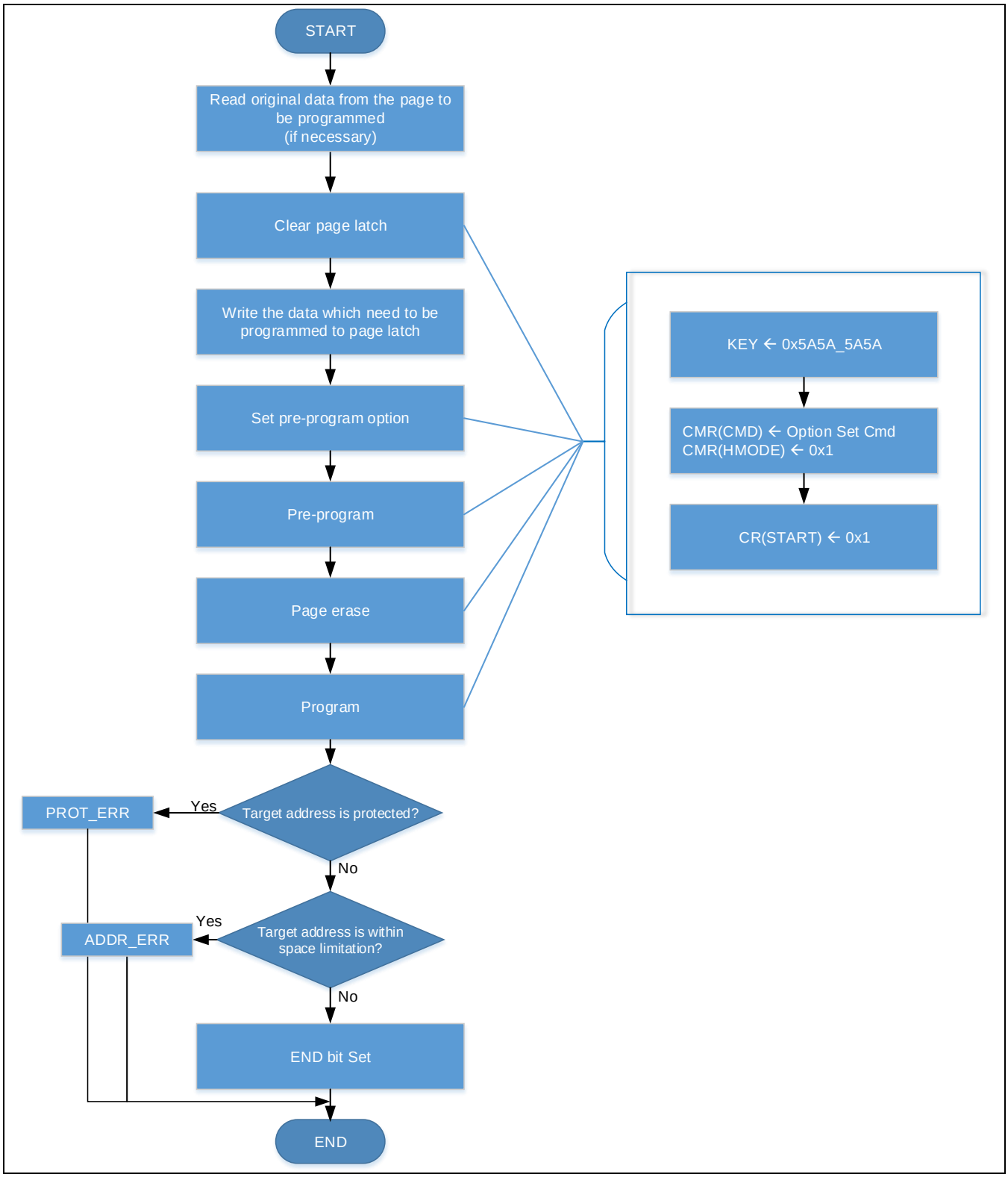


Figure 4-9 自定义选项写操作流程图

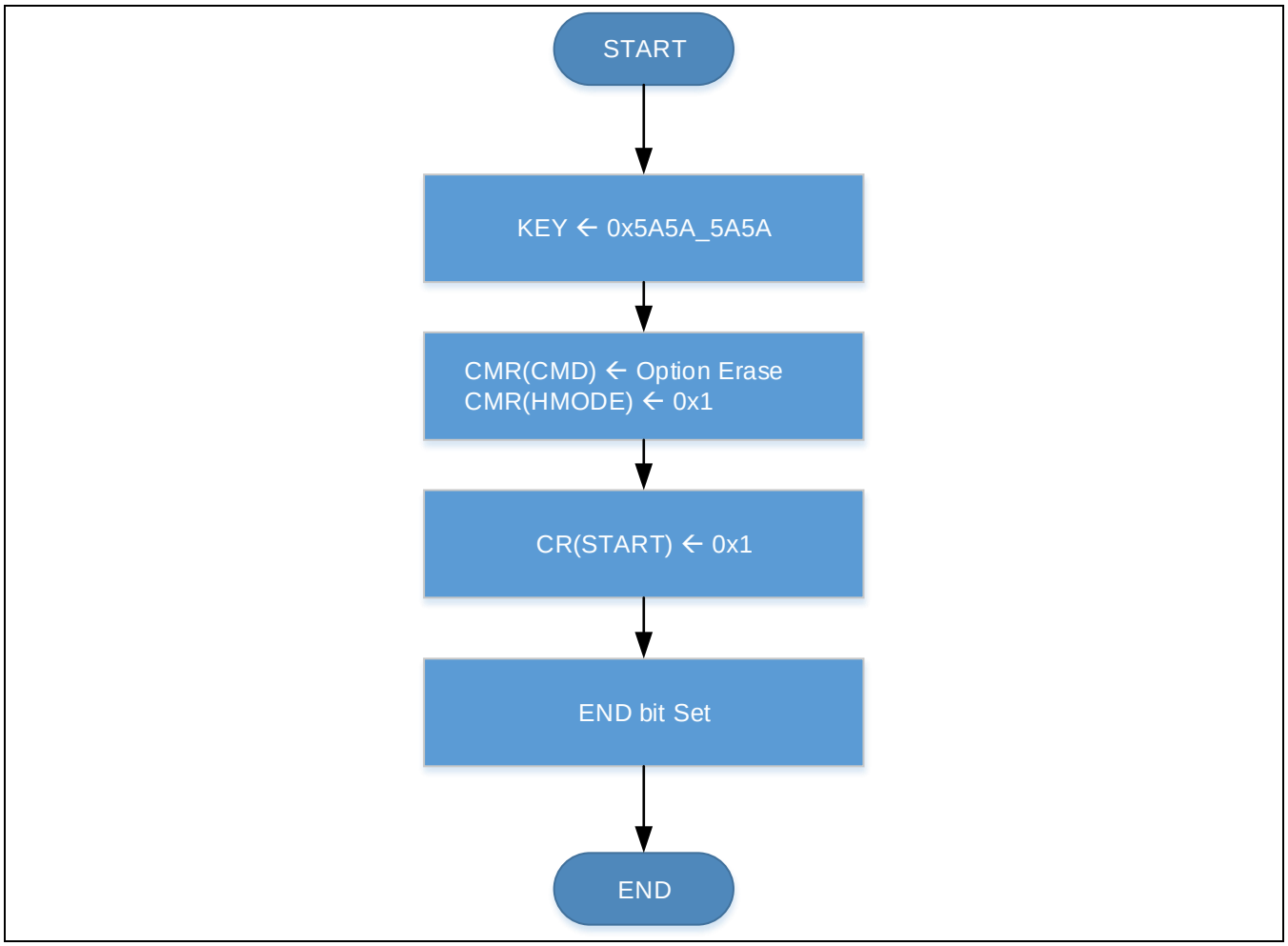


Figure 4-10 自定义选项擦除操作流程图

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	0x000	闪存控制器ID寄存器	0x00000170
IFC_CEDR	0x004	时钟使能/禁止寄存器	0x00000000
IFC_SRR	0x008	软件复位寄存器	0x00000000
IFC_CMR	0x00C	指令寄存器	0x00000000
IFC_CR	0x010	控制寄存器	0x00000000
IFC_MR	0x014	工作模式寄存器	0x00000C00
IFC_FM_ADDR	0x018	ISP地址寄存器	0x00000000
IFC_KR	0x020	ISP秘钥寄存器	0x00000000
IFC_IMCR	0x024	中断控制寄存器	0x00000000
IFC_RISR	0x028	中断原始状态寄存器	0x00000000
IFC_MISR	0x02C	中断状态寄存器	0x00000000
IFC_ICR	0x030	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+ 0x000, Reset Value = 0x00000170

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

Name	Bit	Type	Description
IDCODE	[16:0]	RW	ID代码 (IFC版本号)

4.3.3 IFC_CEDR(时钟使能/禁止寄存器)

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																															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+ 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																															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+ 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								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	RW	RW	RW	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	[23:16]	RW	保护功能选择寄存器 [23] : RSVD [22] : HDP_32K [21] : HDP_16K [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+ 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																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+ 0x014, Reset Value = 0x00000C00

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

Name	Bit	Type	Description
SPEED	[16]	RW	FLASH IP速度模式选择 0: 低速模式 1: 高速模式
PMODE	[8]	RW	并行模式选择（只有DFLASH支持并行模式） 0：等待 1：并行模式（DFLASH擦写的同时，可以进行PFLASH读取）
WAIT	[3:0]	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
80MHz < SYSCLK ≤ 105MHz	4	1
64MHz < SYSCLK ≤ 80MHz	3	1
48MHz < SYSCLK ≤ 64MHz	2	1
16MHz < SYSCLK ≤ 48MHz	1	1
SYSCLK ≤ 16MHz	0	1
SYSCLK ≤ 8MHz	0	0

4.3.8 IFC_FM_ADDR(ISP地址寄存器)

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
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+ 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
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+ 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																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 地址寄存器里写入非法地址 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+ 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																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+ 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																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+ 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																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统的时钟以及和系统工作相关的功能模块，包括不同工作模式下的具体时钟配置，功耗优化控制，系统运行可靠性监测和异常处理（RESET 源历史记录，外部晶振失效监测，低电压报警和复位，看门狗设置，以及外部中断），以及系统安全信息和工程信息等。

通过系统控制器还可以对系统的缺省硬件配置状态（看门狗使能状态，调试口使能状态，Flash 硬件写保护状态，Flash 读保护，用户信息数据）进行查询。系统中若存在支持特性微调的模拟外设，其调整功能一般也通过系统控制器进行微调。

系统控制器的基本特性：

- 系统时钟源选择和 HCLK/PCLK 频率管理
  - 支持多种时钟源作为系统时钟运行：
    - 内部低速振荡器（IMOSC）为缺省时钟源：5.556MHz/4.194MHz/2.097MHz/131.072KHz
    - 内部高速振荡器（HFOSC）：24MHz
    - 外部主晶振（EMOSC）：32.768KHz/0.4MHz ~ 24MHz
    - PLL（源可选 HFOSC/EMOSC）：105MHz max
    - 外部 RTC 晶振（ESOSC）：32.768KHz
    - 内部超低功耗振荡器（ISOSC）：27KHz
  - 可编程 CPU 时钟（HCLK）和外设时钟（PCLK）
  - 外部时钟失效监测（Clock Fail Monitor），支持时钟去抖选项
  - 可选择的系统内部时钟源输出（CLO）
- 各个外设独立的时钟门控，提供功耗优化选择
- 独立看门狗模块，支持上电自动运行并可通过 USEROPTION 定义使能
- 支持复位源记录功能。可用于诊断系统复位，为错误恢复提供支持
- 支持低功耗优化控制。对于不同 CPU 运行模式和负载情况，用户可以自定义电源策略，从而有效降低系统动态功耗。
- 外部中断管理支持从 GPIO 输入的触发信号作为系统事件触发 CPU 中断。
- 低压检测模块（Low voltage Detector）支持外部供电电压监测，在电压低于预设值时可以产生系统中断或

者触发系统复位。

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

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。

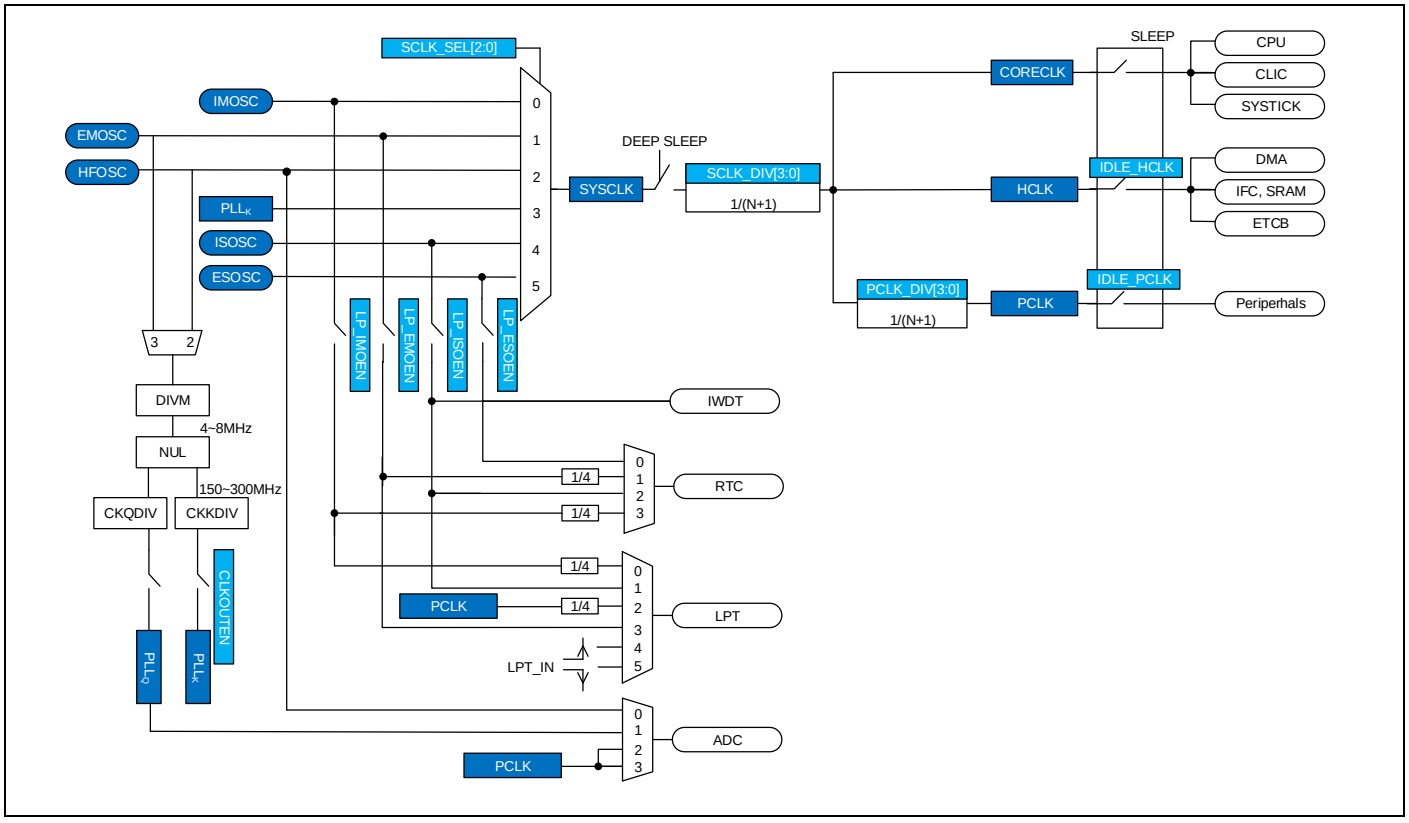


Figure 5-1时钟结构示意图

NOTE:

1) 在POR完成以后，IMOSC为系统的缺省时钟源。
2) 外部时钟（EMOSC/ESOSC）可以由软件使能、关闭。

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

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

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

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

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, 5.556MHz/4.194MHz/2.097MHz/131.072KHz）：内部高精度主振荡器，提供 4 种可选的频率，以满足不同功耗要求。系统上电时，缺省选择 IMOSC 的 5.556MHz 作为工作时钟。同时作为外部主晶振的失效监测管理时钟。

HFOSC（High Frequency OSC, 24MHz）：内部高速振荡器，提供高效的 CPU 运行时钟。在高速时钟下工作时，必须注意 Flash 的读取速度匹配问题。在切换到高速时钟前，必须配置合适的 FlashWait 节拍以匹配 CPU 的速度，匹配关系和设置方法具体参考 Flash 控制器章节。

EMOSC（External Main OSC, 32.768KHz / 0.4MHz ~ 24MHz）：外部主晶振振荡器，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。

PLL（105MHz max）：源可选 EMOSC 或 HFOSC。

ESOSC（External Sub OSC, 32.768KHz）：外部副晶振振荡器。

ISOSC（Internal Sub OSC, 27KHz）：内部超低速振荡器，主要提供 IWDT 的计数时钟。同时作为外部副晶振的失效监测管理时钟。

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

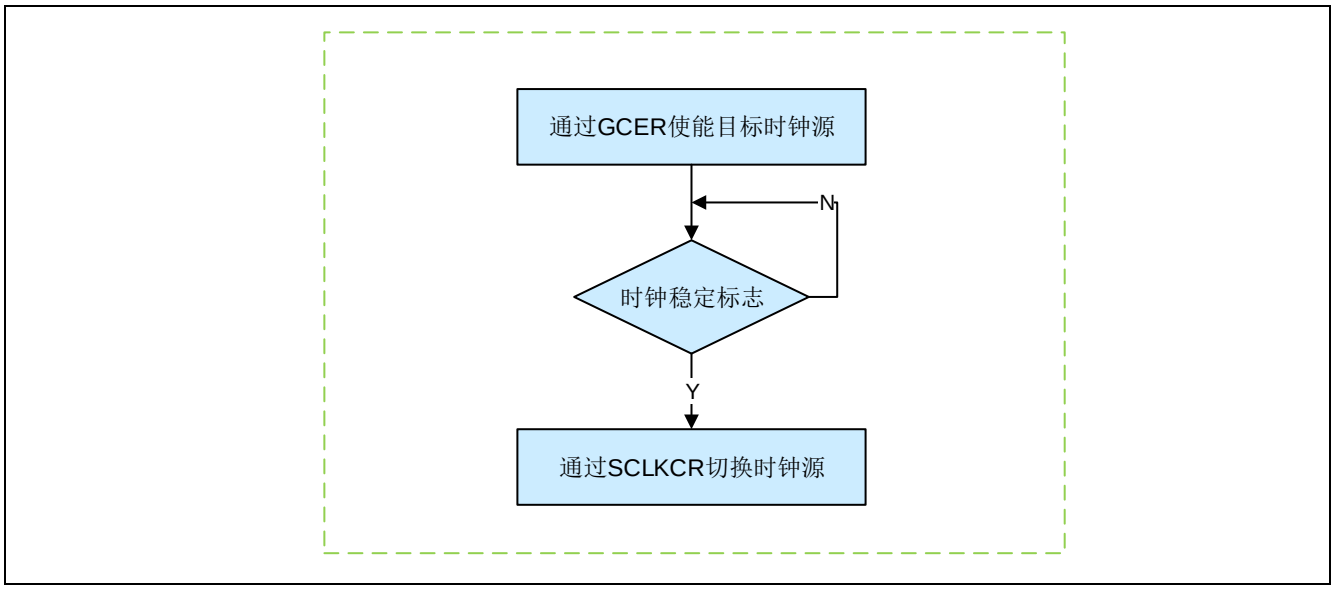


Figure 5-2时钟源切换示意图

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。

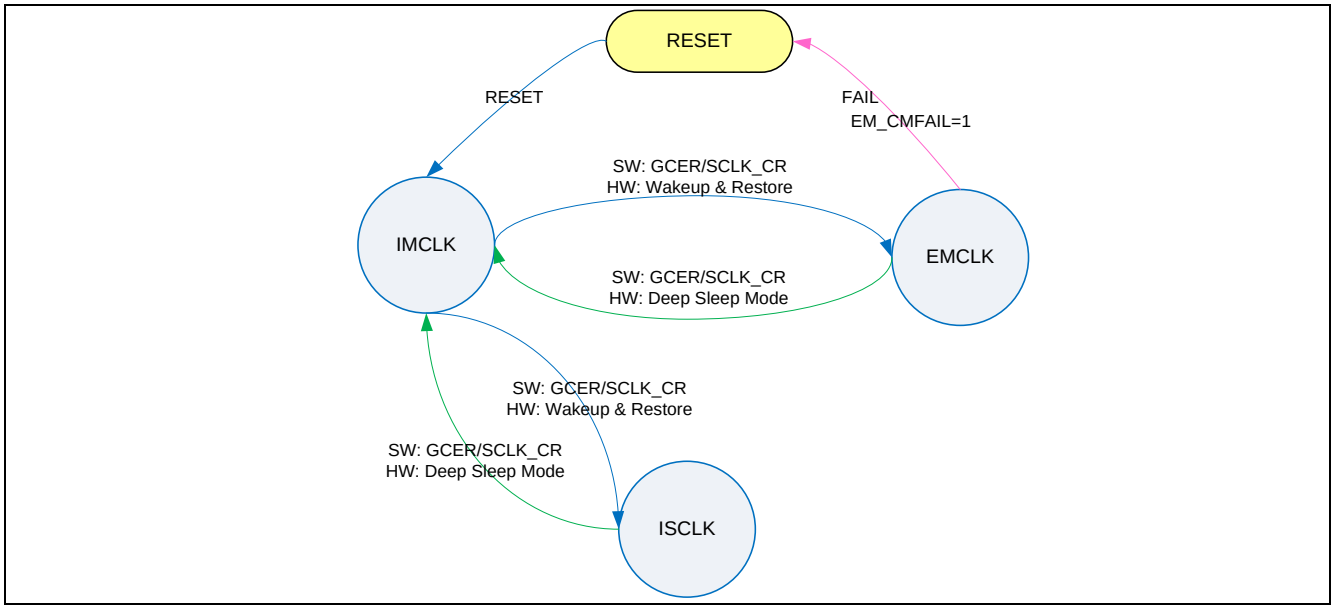


Figure 5-3时钟切换状态机

5.2.2.1 选择内部时钟源进行工作

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

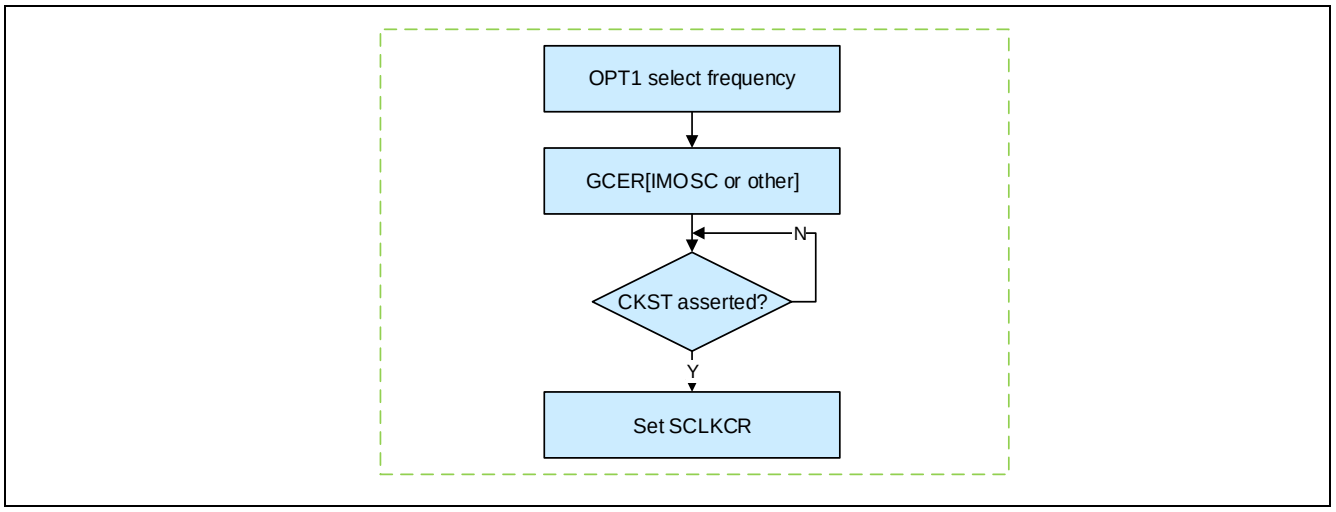


Figure 5-4配置内部时钟源

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

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

5.2.2.2 内部时钟源频率微调

内部时钟源（包括 IMOSC，HFOSC，ISOSC）在出厂前已经经过精度校准，以保证振荡器的频率在 Spec 定义范围之内。系统也为客户提供了在程序中再次调整频率的途径，以满足客户自定义频率的需求。其中 HFOSC 还提供精细调节的功能。HFOSC 的细调通过 CLCR[HFO_TUNE]控制进行调节，其缺省值为 0h100，当增大 HFO_TUNE 时，HFOSC 的频率变快，减少 HFO_TUNE 时，HFOSC 的频率变慢。HFO_TUNE 的每一步调节都

是等量的且分辨率为 0.65‰，即 HFO_TUNE 的值每变化 1，频率变化

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[HFO_TRM]、[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
HFOSC	183-TRM
ISOSC	339-TRM

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

5.2.2.3 选择PLL进行工作

芯片内集成了 PLL 模块，允许更高更灵活的频率设置。PLL 模块框图如下：

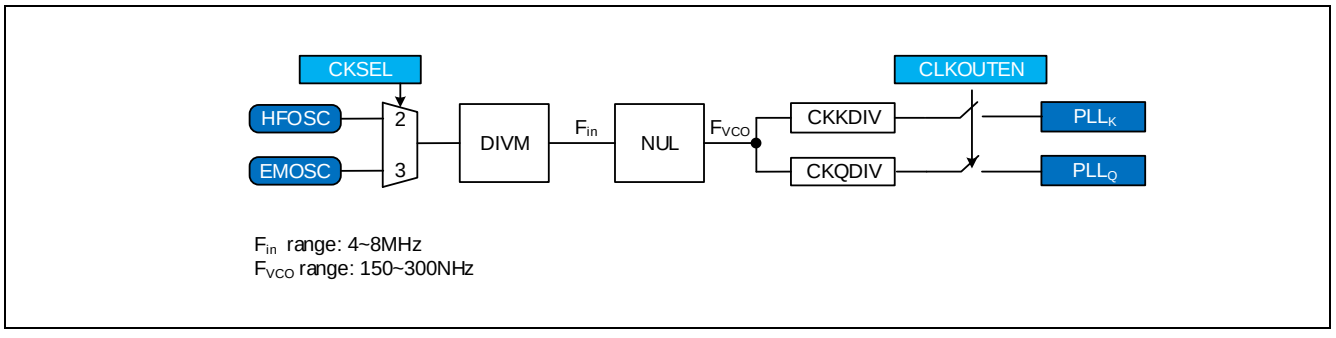


Figure 5-5 PLL模块框图

时钟输入可以通过 PLLCR[CKSEL]，从 HFOSC 和 EMOSC 中选择。DIVM 的设置必须保证 Fin 在 4~8MHz 内，NUL 的设置必须保证 FVCO 在 150~300MHz 内，否则系统会进入异常，同时 PLL 失锁标志位（RISR[PLLUL]）置起。如果此时 PLL 失锁中断使能（IMCR[PLLUL] = 1）,会触发失锁中断。

PLL 模块可以同时有两个不同的频率输出，分别为 PLL_K 和 PLL_Q，同时受 PLLCR[CLKOUTEN]开关控制。PLL_K 可以被选为系统时钟，PLL_Q 可以被选为 ADC 模块的时钟源。

5.2.2.4 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：

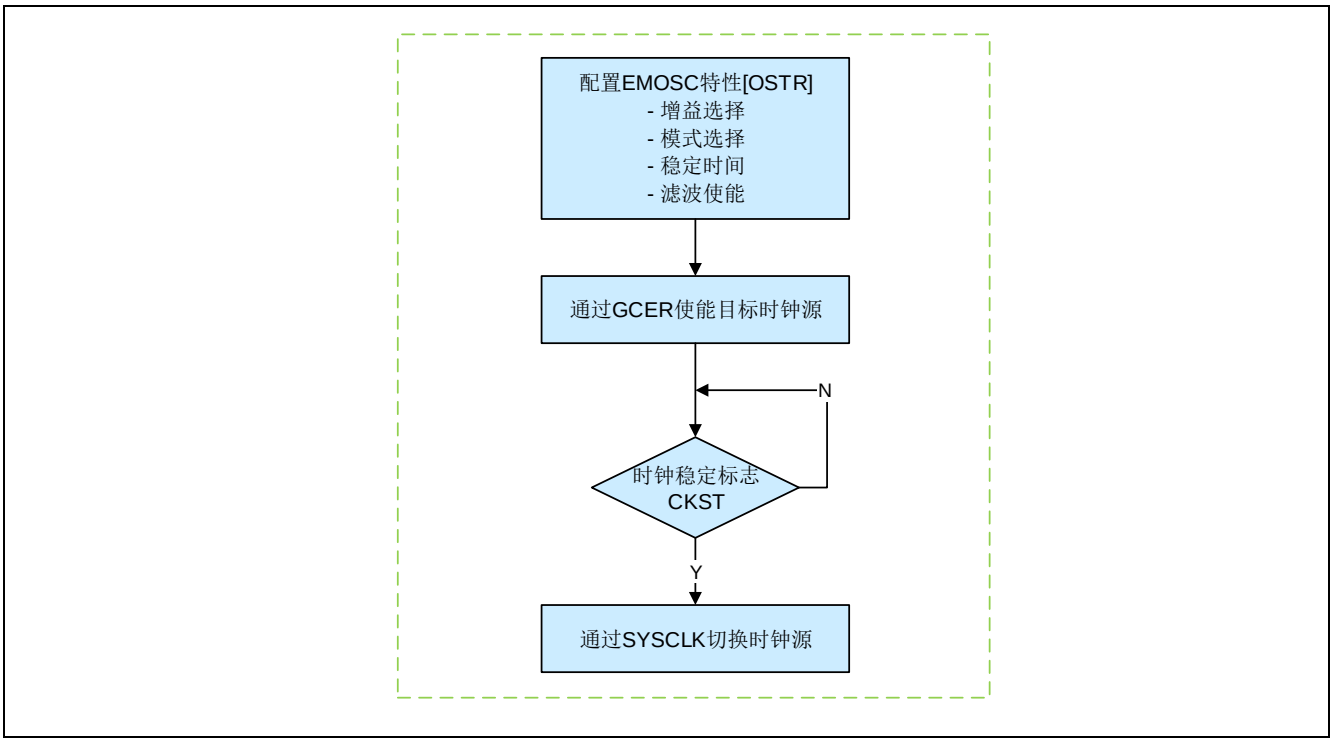


Figure 5-6配置外部时钟源

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 5-2CYOSC_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。

5.2.2.5 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）

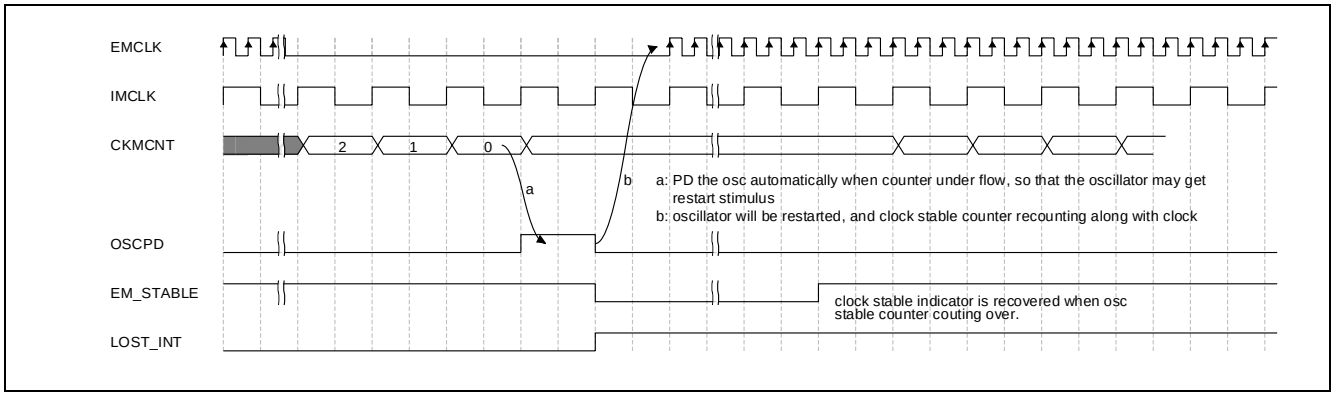


Figure 5-7 EMOSC 失效监测

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。

5.2.2.6 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 RSTDET_LVL 的设置值时，产生硬件复位信号。当外部供电电压 VDD 低于 INTDET_LVL 的设置值或高于 INTDET_LVL 设置值时，系统将会产生 LVD 中断请求（IER、IDR 寄存器中的 LVD_INT 位可以设置或者清除中断标志）。通过 INTDET_POL 控制位可以选择中断触发的条件，选择 VDD 下降沿触发或上升沿触发，或者两者均可触发。当前外部供电电压的状态，可以通过 LVDCR 的 LVDFLAG 位检测到。当外部供电电压低于检测 level 时，该标志位为‘1’，当高于检测 level 时，该标志位为‘0’。

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

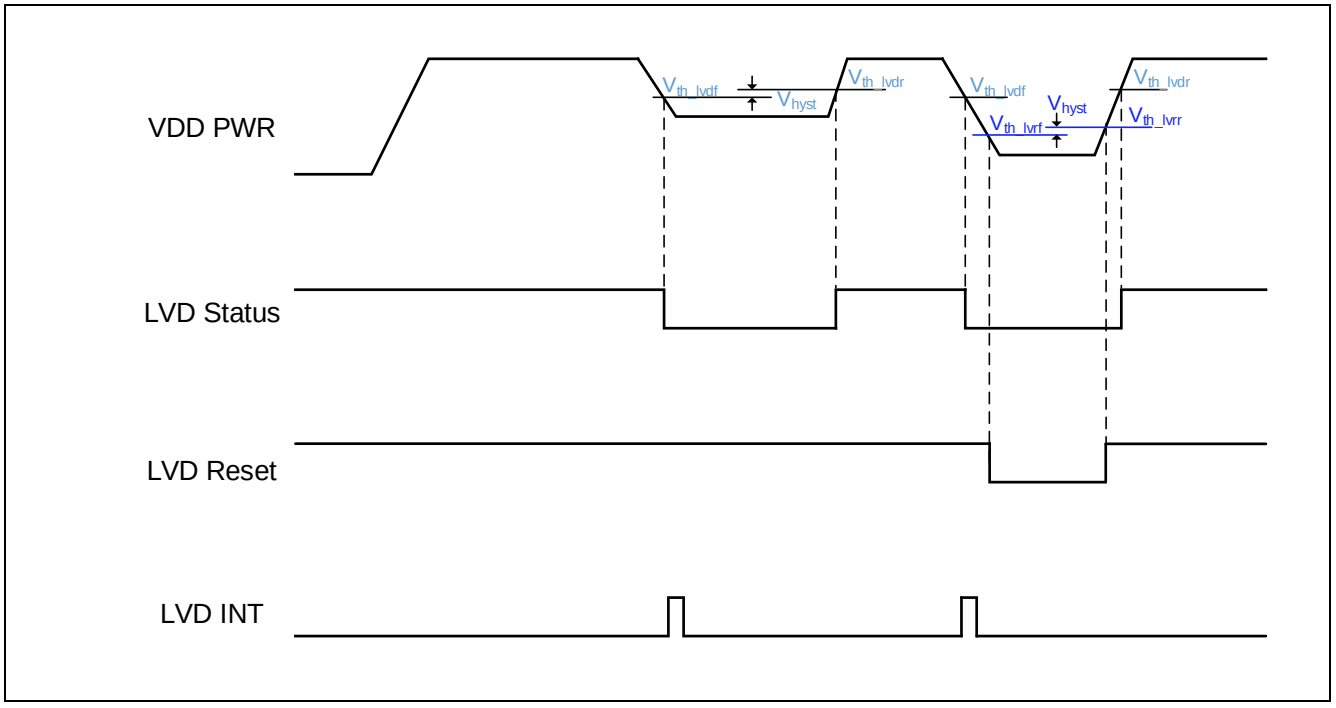


Figure 5-8LVD 工作时序图

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

系统支持的低功耗模式有三种：

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

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

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

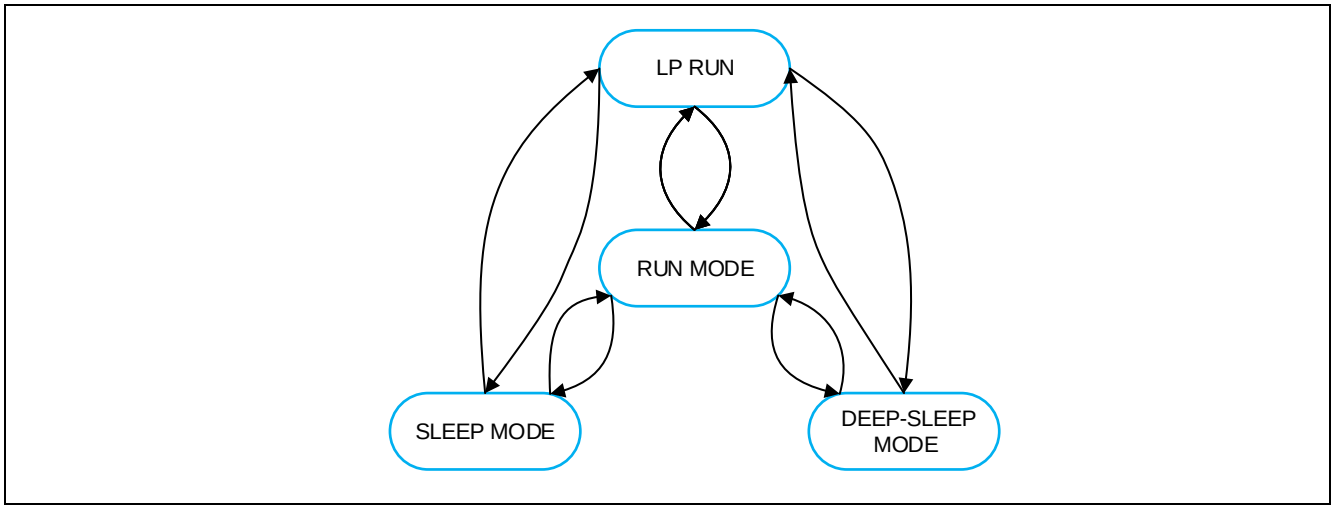


Figure 5-9模式切换示意图

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, WDT, RTC, LPT interrupt. Corresponding WKEN bit should be set	All Clock is Off, including CPU and peripheral clock in default Clock source can be selected to keep working by STPEN bit

Table 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	—
HFOSC	O	—	O	—	—
IMOSC	O	O	O	O(3)	—
ISOSC	O	O	O	O(3)	—
EMOSC	O	O	O	O(3)	—
Clock Monitor	O	O	O	O	—
RTC	O	O	O	O	O
UART	O	O	O	—	—
I2C	O	O	O	—	—
SPI	O	O	O	—	—
ADC	O	O	O	—	—
TIMER	O	O	O	—	—
LPT	O	O	O	O	O
IWDT	O	O	O	O	O
WWDT	O	O	O	—	—
CORET	O	O	O	—	—
EXI	O	O	O	O	O

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

2. Flash 可以通过软件配置为停止工作，缺省状态为使能（即不停止工作）。

3. 在 DEEP-SLEEP 模式下，缺省将关闭所有的时钟源，但为保证某些外设可以继续工作，可以设置在该低功耗模式下不关闭特定的时钟源。

调试模式

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

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

5.2.4.1 运行模式（RUNMODE）

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。

5.2.4.2 低功耗运行模式（LOWPOWERRUNMODE）

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 FlashMemory 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 FlashMemory 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 WAITCYCLE 以保证 Flash 的时序正确。

进入 LP RUN 模式的方式：

进入 LPRUN 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。通过 PWROPT[FLASH_PD]寄存器可以禁止 FlashMemory，并停止 FlashMemory 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[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）设置 VOSLCK 寄存器并使能 VOS 功能；设置 PWRCR[VOSEN]使能 RUN 模式的 VOS 选项并同时设置相应的 RUN_CFG 控制位。

退出 LPRUN 模式的方式：

退出 LPRUN 模式的方式可以参考如下步骤进行：

1）通过设置 VOSLCK 取消 VOS，或者清除 PWRCR[VOSEN]中的 RUN 模式控制位。

2）清除 OPT1[LPMD]控制位，恢复 Flash 的访问时间限制。

3）切换系统频率到目标频率。

5.2.4.3 睡眠模式（SLEEPMODE）

在 SLEEP 模式下，系统控制器将挂起 CORE 模块的时钟（CPU 将不会工作）。缺省模式下，所有的逻辑外设控制时钟（PCLK）不会被停止，但是通过 GCDR[IDLE_PCLK]控制位可以设置在 SLEEP 模式下也停止 PCLK。如果 SLEEP 模式下的 PCLK 被禁止，则需要注意外设在没有 PCLK 时可能不能工作，从而将不能正常产生唤醒事件。在 SLEEP 模式下，所有振荡器的工作状态不会做任何改变。任何外设事件或者中断都可以触发系统从该模式退出。

在 SLEEP 模式下，若 PCLK 没有被禁止，则在进入模式前被使能的外设将继续工作，且 IO 工作不受影响，例如 GPIO 上的 TIMER 输出在进入 SLEEP 前被打开，则进入 SLEEP 后，该 IO 仍旧可以正常输出。

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

Table 5-5SLEEP 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	

5.2.4.4 深度睡眠模式（DEEP-SLEEPMODE）

在 DEEP-SLEEP 模式下，系统控制器将挂起所有的时钟源，但内部逻辑电源仍将保持不变。在 IWDT 使能前提下，ISOSC 将不受模式切换的影响，一直保持工作。在进入 DEEP-SLEEP 时，系统将首先挂起总线和外设时钟，然后切换系统时钟到内部 IMOSC，并依次关闭所有时钟源。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
RTC	
LPT	
LVD	

可以通过设置 GCER[STP_xxx]控制位来设置时钟源关闭的例外情况。当相应时钟源的 STP 控制位被使能，则该时钟源在 DEEP-SLEEP 模式下将不会被自动关闭，其状态将一直保持在进入 DEEP-SLEEP 模式前的状态。该设置用于保证某些使用特殊时钟的外设可以在 DEEP-SLEEP 模式下继续工作，例如 LPTIMER 或 RTC 等。

在 DEEP-SLEEP 模式下，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。

Table 5-7DEEP-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	

5.2.4.5 低功耗唤醒和中断服务

当系统进入低功耗模式后，只有通过中断进行唤醒。系统唤醒经过两个阶段，第一阶段，当 SYSCON 检测到有中断发生，会自动切换工作环境为后续系统运行恢复环境，这包括切换 LDO 到相应的工作状态，切换参考电压源以保证精度，唤醒 Flash memory 并初始化，和系统时钟的恢复等。第二阶段，当系统工作环境恢复以后，SYSCON 会提供所有的总线时钟包括 CPU 时钟，同时给予 CPU 相应的中断信号，CPU 在检测到中断请求信号后，会根据设置退出低功耗模式（DOZE 和 STOP 指令），并继续正常的取指和执行工作。

CPU 退出低功耗模式由 NVIC 中的唤醒寄存器控制，当 Active 的中断没有在 NVIC 中设置为唤醒中断源，则 CPU 不会响应该中断（即使该中断已经置位），并继续保持低功耗模式。CPU 退出低功耗模式后，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。

如果 IMOSC 在 DEEP-SLEEP 进入前没有使能，则初始化的时间需要额外增加 IMOSC 的稳定时间。初始化的时间为可以用下面的公式进行估计：

$$T_{init_stop} = T_{imosc_stable} + T_{clk_sw} + T_{emo_dis} + T_{hfo_dis} + T_{iso_dis} + T_{imo_dis}$$

其中 T_{imosc_stable} 为 IMOSC 的稳定时间，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 功耗优化和管理

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

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

VOS 设置过程如下：

设置 VOSLCK 寄存器，使能 VOS 功能。

配置 PWRCR 寄存器，使能相应 VOSEN 控制位和配置相应 VOSCFG

在没有特殊功耗需求的前提下，可以不使能 VOS 功能，若使能 VOS 则请特别注意当设置的 LDO 特性不能满足当前工作需求时，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
CPU1RST	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）。
WWDT	WWDT产生的系统复位。
EFL_ERR	FLASH硬件校验错误，并且重试次数溢出后复位。

RAM_ERR	SRAM硬件校验错误，并且重试次数溢出后复位。
PLLUNLOCK	PLL失锁产生的系统复位
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’10）。
EMCKM	EMOSC CKM Fail产生的系统复位。
ESCKM	ESOSC CKM Fail产生的系统复位。
IWDTRST	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）。
LVDRST	由低电压监测模块（LVD）产生的系统复位信号。可以通过软件势能或者关闭该功能。
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（EXI0~EXI4）。

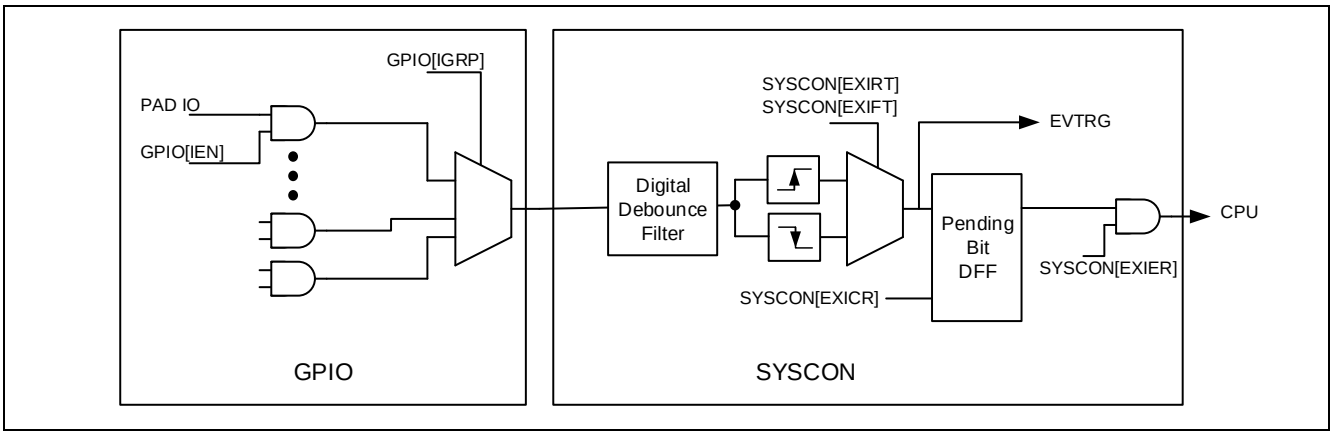


Figure 5-10外部中断结构示意图

外部中断在配置时，由于每个 PADIO 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。因此，在配置过程中，应将 EXI 的中断关闭，且在配置完成后，对 PendingBit 进行一次软件清除，以保证在使能 EXI 后，不会立即触发 CPU 中断。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。

Table 5-9EXI 中断配置信息

GPIO GROUP IN GPIO	EXILINE 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

NOTE: 上表中的 GROUP 对应 GPIO 的分组，EXI LINE 中的 EXIx 表示在 SYSCON 中的中断源。CPU VECTOR 列表示该中断源标志位被置位后，将触发的 CPU 中断。

配置和使能外部中断的方法：

1）确认 EXI 的开关被关闭（通过 EXIDR 寄存器进行设置）

1）设置 CPU 中 NVIC 的 EXI 中断为使能状态，并设置 GPIO 内的 EXIEN 控制位，使能相应 GPIO 的 EXI 功能。配置 GPIO 的 IGRP，选择 EXI 触发事件的信号源。

2）设置 SYSCON 内的 EXIRT 或 EXIFT 选择触发信号的边沿类型。

3）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，然后设置 EXIER 打开相应的 EXI 中断。

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

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

5.2.7.2 外部中断的滤波控制

所有的管脚都可以通过 GPIO_FLTEN[Px]使能或旁路一个模拟滤波器，用来滤除 30ns 以下的脉冲毛刺信号。随后在 SYSCON 的 EXI 模块中还有一种由 HFOSC 时钟同步的数字滤波器。当数字滤波器被使能，外部输入的 EXI 触发信号，在数字滤波器内通过 HFOSC 时钟对信号采样并进行数字去抖处理。只有 Group0-3 和 Group16-19 可以通过 EXIFLTCR[GRPx]选择使用该滤波器。每个 group 可以设置不同的滤波参数。

如果在应用中设置通过 EXI 唤醒低功耗模式，用户需要确认 HFOSC 在该低功耗模式下的状态，如果不工作，则必须在进入低功耗模式前，将数字滤波器禁止，以保证 EXI 通路上没有时钟同步滤波逻辑。

5.2.8 内存可靠性监测

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

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

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

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

内存校验功能分为 FLASH 校验和 SRAM 校验，分别通过 EFLCHK 和 RAMCHK 寄存器进行设置。缺省条件下，校验功能为禁止状态。Flash 的校验功能可以通过 USER Option 设置缺省使能或者禁止状态。

5.2.9 独立看门狗

看门狗定时器的作用是在系统运行中，由于程序干扰或者错误运行，导致处理器运行到一个未知的状态中时，产生一个系统复位请求，把处理器重新置位到初始化状态。他可以保证系统不会因为程序运行错误导致永久挂起。除外，看门狗定时器中断还可以作为处理器在 DEEP-SLEEP 模式下定时唤醒的中断源，间隔唤醒系统工作，大幅降低系统的整体功耗。IWDT 是一个独立工作的看门狗模块，和系统的工作状态无关。它内部通过一个 12 位的 Free Running 递减计数器控制定时时间。独立看门狗和运行模式无关，一旦使能则必须在计数器溢出前进行清除，否则会产生系统复位。

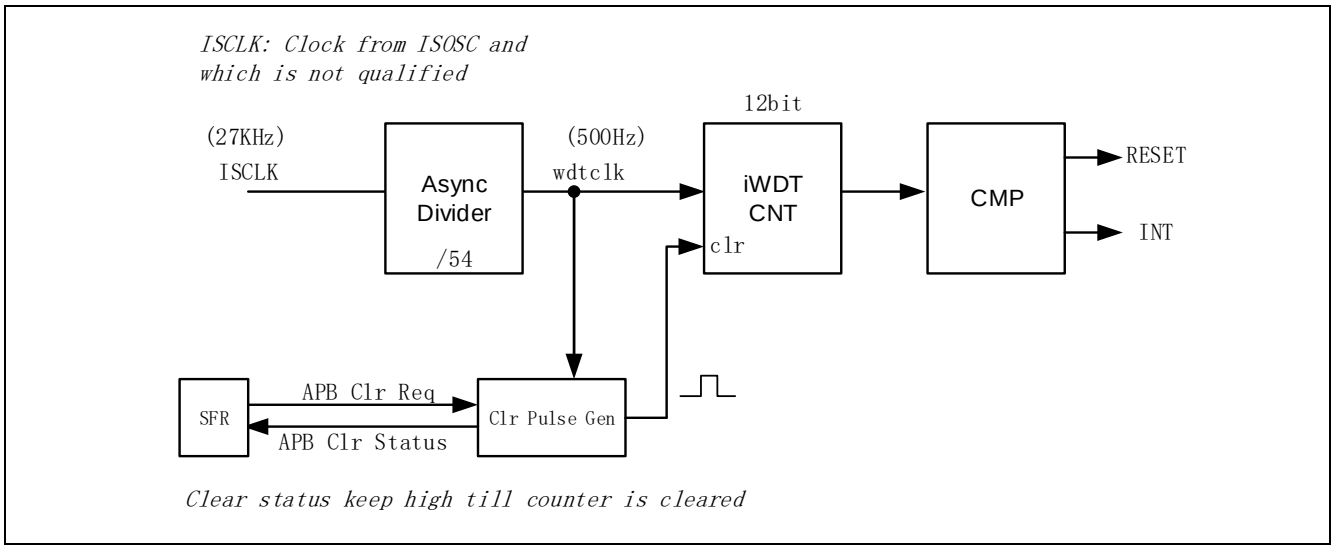


Figure 5-11IWDT 结构框图

IWDT 的缺省状态可以通过 Flash 内部的 User Option 设置。在软件中，可以通过设置 IWDEDR 寄存器中的 IWDT_EDC 位来打开或者关闭 IWDT。当清除 IWDT 操作时，IWDT 中计数器的预置数会被重新置位。清除 IWDT 通过对 IWDCNT 寄存器的 IWDT_CLR 位写入 0x5A 实现。只有当清除 CNT 操作发生时，计数器值才会被更新到最新的设置值，所以在更新了 IWDCNT 寄存器后，需要软件清除一次 CNT，使得内部计数器及时更新到最新的配置。软件清除需要在 IWDT 启动以后执行（通过查询 IWDCR[BUSY]控制位确认 IWDT 是否已经启动）。

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

间为 128ms。

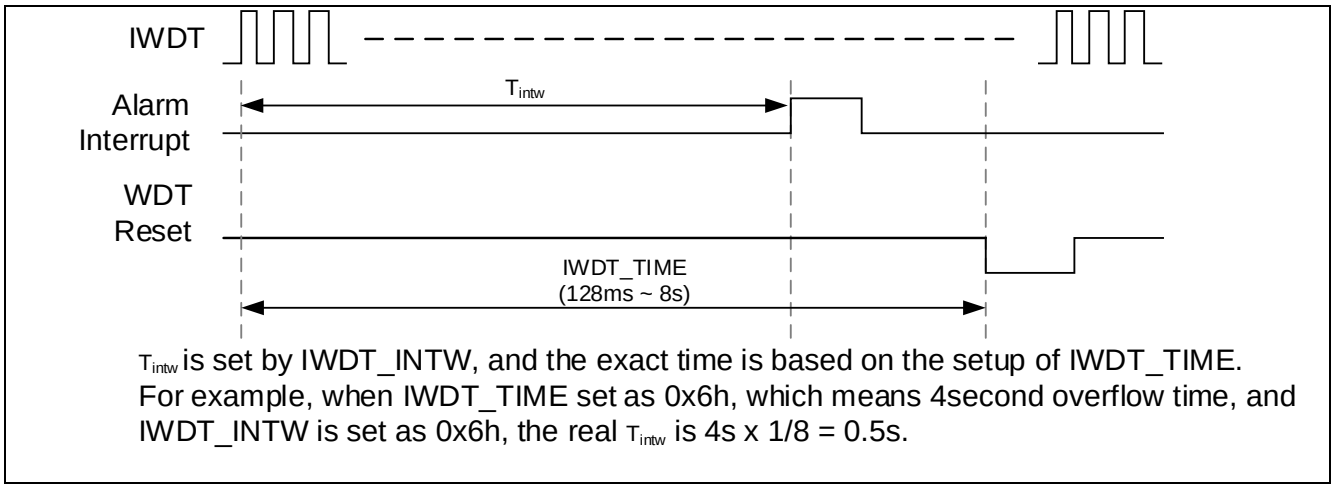


Figure 5-12IWDT 计数器溢出和报警中断

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

5.2.10 IO重定义

为提供更灵活的 IO 功能配置，系统提供了自定义 GPIO 复用的功能。芯片提供两个预设的 GPIOGROUP，分别为 GROUP0 和 GROUP1，两个 GROUP 分别对应 8 个预设的可选择的复用功能。在每个 GROUP 内，每个 GPIO 可以被指定为这 8 个预设功能中的任意一个作为该 GPIO 的 AF7 或 AF8 功能，见数据手册 2.3 管脚功能分配。

Table 5-10IOGROUP0和 GPIO 对应关系

GPIO	G0	CFGVAL in IOMAP0
PB10	AF7 G0.0	CFGVAL0
PB11	AF7 G0.1	CFGVAL1
PB12	AF7 G0.2	CFGVAL2
PA15	AF7 G0.3	CFGVAL3
PB5	AF7 G0.4	CFGVAL4
PD3	AF7 G0.5	CFGVAL5
PB8	AF8 G0.6	CFGVAL6
PB9	AF8 G0.7	CFGVAL7

Table 5-11IOGROUP1和 GPIO 对应关系

GPIO	G1	CFGVAL in IOMAP1
PD4	AF8 G1.0	CFGVAL0
PD0	AF8 G1.1	CFGVAL1
PA4	AF8 G1.2	CFGVAL2
PA5	AF8 G1.3	CFGVAL3
PA6	AF8 G1.4	CFGVAL4
PA7	AF8 G1.5	CFGVAL5
PB0	AF8 G1.6	CFGVAL6
PB1	AF8 G1.7	CFGVAL7

IOMAP0/1[CFGVALx] 的值决定了 IO 的最终功能。详见下表。比如，当 IOMAP0[CFGVAL0] = 0, PB10 的 AF7 功能即为 I2C_SCL，当 IOMAP1[CGFVAL3] = 4, PA5 的 AF8 即为 UART_RX。

Table 5-12 IOGROUP 配置信息

AFFunction	CFGVAL in IOMAP	GROUP
I2C_SCL / USART0_TX	0x00	GROUP0 / GROUP1
I2C_SDA / USART0_RX	0x01	GROUP0 / GROUP1
USART0_TX / USART0_SCK	0x02	GROUP0 / GROUP1
USART0_RX / UART2_TX	0x03	GROUP0 / GROUP1
SPI0_NSS/ UART2_RX	0x04	GROUP0 / GROUP1
SPI0_SCK/ GPTA0_CHA	0x05	GROUP0 / GROUP1
SPI0_MISO/ GPTA1_CHA	0x06	GROUP0 / GROUP1

SPI0_MOSI / CMP0_OUT	0x07	GROUP0 / GROUP1

注：当使用 IOMAP 功能时，PB12/PD4/PA5/PB1 不能做为设备的输入管脚。比如，当 IOMAP0[CFGVAL2] = 3, PB12 的 AF7 功能配置为 USART0_RX,因为 USART0_RX 为串口输入管脚，此时配置的 IO 复用功能无效。

5.2.11 系统信息及自定义寄存器

系统中存在几类信息存储寄存器，作为软件获取系统信息的一种途径。

ERRINF：错误信息寄存器

在对 SYSCON 的一些寄存器进行配置时，由于配置条件错误，或者配置顺序错误，可能导致系统失效。为避免系统失效，错误的寄存器配置操作会被忽略，此时可以通过 ERRINF 寄存器调试并获得错误相关信息。

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。

FINF：工厂信息区

保留作为工厂相关信息的存储

UREG：自由寄存器

用于保存客户临时信息的寄存器，该寄存器内保存的数据只有在上电复位后才会丢失。该寄存器无特殊功能，仅为用户提供在应用程序中可以方便地存储和监测不受其他复位影响的临时状态。

5.2.12 中断管理

系统控制器的中断由 22 个中断源组成，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，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。

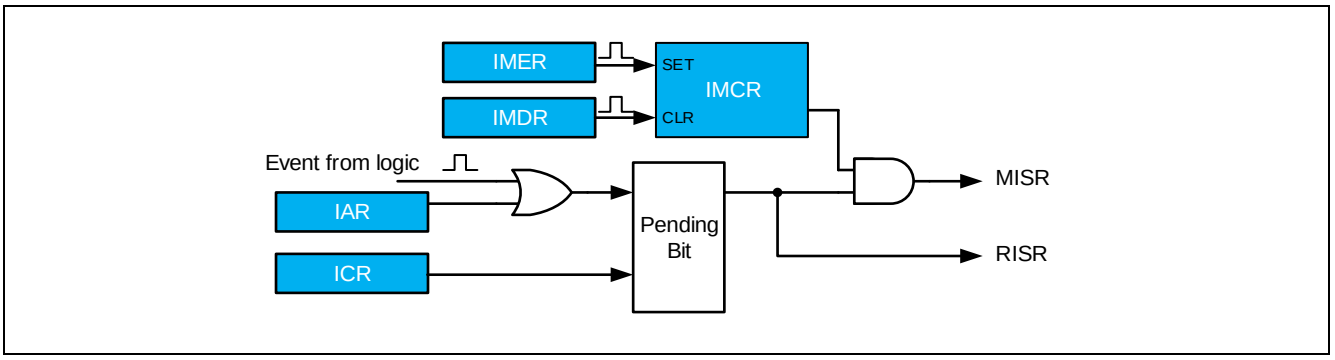


Figure 5-13SYSCON 中断控制逻辑

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 5-13 SYSCON General Event to Trigger Interrupt

Trigger Event	Description
CMD_ERR	命令错误中断，在对某些寄存器错误操作时会
EV3TRG ~EV0TRG	事件触发输出Event3~0 触发的中断
EM_CMFAIL	EMOSC时钟失效中断
CQCDN	Clock Qualification完成中断
ES_CMFALL	ESOSC时钟失效中断
PLLUL	PLL UNLOCK中断
OPL_ERR	Option初始化配置加载失败中断
EFL_ERR	EFLASH校验失败中断
LVD_INT	LVD中断
RAM_ERR	SRAM校验失败中断
IWDT_INT	IWDT中断
SYSCLK_ST	SYSCLK时钟稳定中断
PLL_ST	PLL时钟稳定中断
HFOSC_ST	HFOSC时钟稳定中断
EMOSC_ST	EMOSC时钟稳定中断
ESOSC_ST	ESOSC时钟稳定中断
IMOSC_ST	IMOSC时钟稳定中断
ISOSC_ST	ISOSC时钟稳定中断

另外 5 个中断源为外部中断，包括 EXI_V0~EXI_V4。

5.2.13 事件触发控制

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:

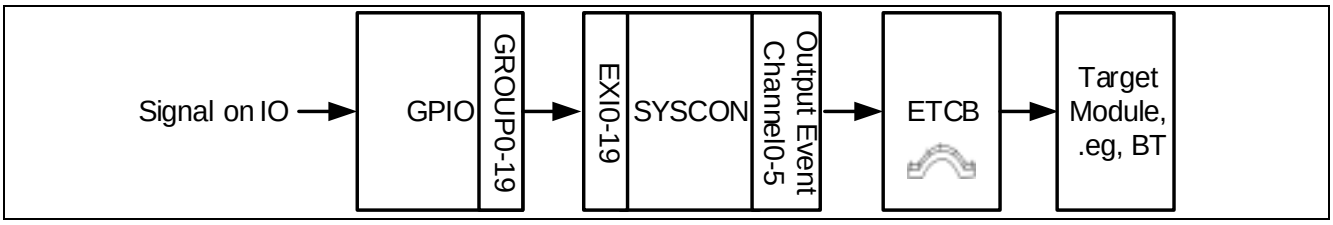


Figure 5-14 Signal Flow

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 的触发源可以从 EXI0~EXI15 中选择，当对应通道触发中断使能（IMCR[EVxTRG] = 1）时，可以触发中断； 通道 4 和通道 5 可以从 EXI16~EXI19 中选择，这两个通道不支持中断。

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。

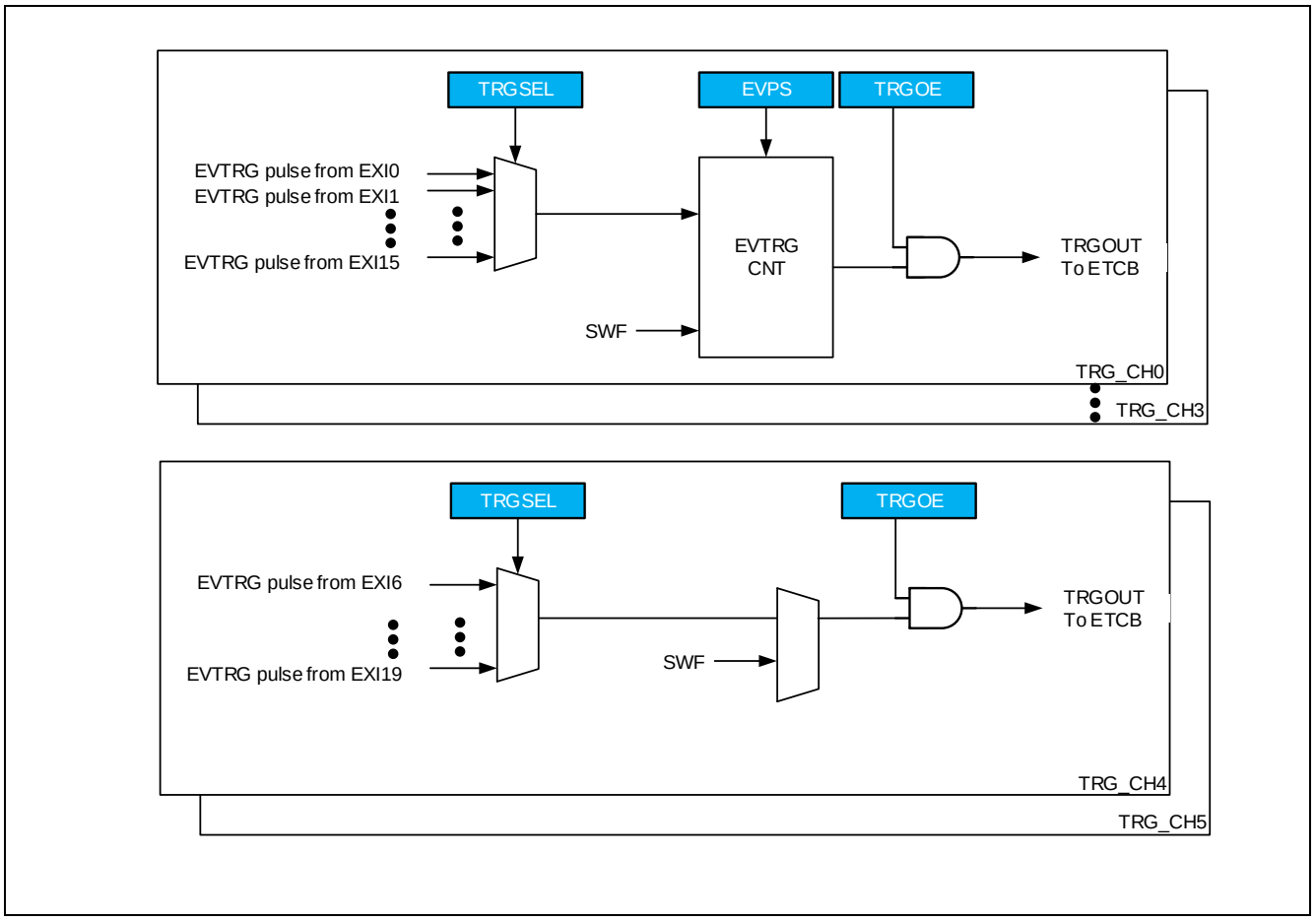


Figure 5-15SYSCON 外部事件触发控制逻辑

5.2.14 调试管脚的复用

在芯片上电时，系统控制器会锁定调试脚的复用功能，如果需要使用调试脚的复用功能，请通过设置调试控制寄存器 SYSCON_DBGCR 的 SWD_AFLOCK 位为 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。

5.2.15 时钟质量检测

时钟质量检测功能可用于比较两个时钟的计数频率。参考时钟可以通过 CQCR[REFSEL]选择 EMOSC 和 ESOSC 中的一个。被检测时钟，即源时钟可以通过 CQCR[SRCSEL]选择 IMOSC、ESOSC、ISOSC、HFOSC 中的一个。启动检测前，需要先设置源时钟计数停止值 CQCR[CQSVAL]。开启检测后，源时钟和参考时钟同时启动计数，当源时钟计数到 CQCR[CQSVAL]时，两个时钟同时停止计数，参考时钟的停止计数值存入 CQRR 中。

当源时钟和参考时钟都精准时，CQRR 的值满足：$\frac{CQRR}{Fref} = \frac{CQSVAL}{Fsrc}$。

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	0x00064801
SYSCON_GCER	0x004	通用使能控制寄存器	0x00000000
SYSCON_GCDR	0x008	通用禁止控制寄存器	0x00000000
SYSCON_GCSR	0x00C	通用状态寄存器	0x00081103
SYSCON_CKST	0x010	时钟状态寄存器	0x00000103
SYSCON_RAMCHK	0x014	SRAM 校验控制寄存器	0x0000FFFF
SYSCON_EFLCHK	0x018	Embedded Flash 校验控制寄存器	0x00FFFFFF
SYSCON_SCLKCR	0x01C	系统时钟控制寄存器	0x00000100
SYSCON_PCLKCR	0x020	外设时钟控制寄存器	0x00000100
SYSCON_EXIFLTCR	0x024	外设时钟控制寄存器	0x00000000
SYSCON_PCER0	0x028	外设时钟使能寄存器0	0x00000000
SYSCON_PCDR0	0x02C	外设时钟禁止寄存器0	0x00000000
SYSCON_PCSR0	0x030	外设时钟状态寄存器0	0x00100001
SYSCON_PCER1	0x034	外设时钟使能寄存器1	0x00000000
SYSCON_PCDR1	0x038	外设时钟禁止寄存器1	0x00000000
SYSCON_PCSR1	0x03C	外设时钟状态寄存器1	0x01000000
SYSCON_OSTR	0x040	外部振荡器稳定时间配置寄存器	0x70FF3BFF
SYSCON_PLLCR	0x044	PLL控制器	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	0xXXXX00XX
SYSCON_PWRCR	0x054	功耗控制寄存器	0x0f070040
SYSCON_PWRKEY	0x058	功耗调整使能寄存器	0x00000000
SYSCON_OPT1	0x064	系统配置寄存器1	0x00000000
SYSCON_OPT0	0x068	系统配置寄存器0	0x00000001
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	0x00000083
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	0x00000080
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	0x80000000
SYSCON_IWDCR	0x0B4	独立看门狗控制寄存器	0x0000170C
SYSCON_IWDCNT	0x0B8	独立看门狗控制计数器值	0x00000ffd
SYSCON_IWDEDR	0x0BC	独立看门狗使能寄存器	0x000000ff
SYSCON_IOMAP0	0x0C0	GPIO分组0的功能映射配置寄存器	0x00000000
SYSCON_IOMAP1	0x0C4	GPIO分组1的功能映射配置寄存器	0x00000000
SYSCON_FINF0	0x0D0	工厂信息寄存器0	0x00000000
SYSCON_FINF1	0x0D4	工厂信息寄存器1	0x00000000
SYSCON_FINF2	0x0D8	工厂信息寄存器2	0x00000000
SYSCON_ERRINF	0x0E0	错误信息寄存器	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	0x00000000
SYSCON_EVTRG	0x0F4	事件触发选择寄存器	0x00000000
SYSCON_EVPS	0x0F8	事件触发计数寄存器	0x00000000
SYSCON_EVSWF	0x0FC	事件计数器软件触发控制寄存器	0x00000000
SYSCON_UREG0	0x100	32位用户寄存器	0x00000000
SYSCON_UREG1	0x104	32位用户寄存器	0x00000000
SYSCON_UREG2	0x108	16位用户寄存器	0x00000000
SYSCON_CQCR	0x118	时钟源频率验证控制寄存器	0x00000000
SYSCON_CQRR	0x11C	时钟源频率验证计数器值	0x00000000
SYSCON_DBGCR	0x128	调试控制寄存器	0x00000000
SYSCON_BRPKEY	0x12C	复位地址配置保护寄存器	0x00000000
SYSCON_SRBREG	0x130	系统软件复位引导地址配置寄存器	0x00000000
SYSCON_ERBREG	0x134	系统异常复位引导地址配置寄存器	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 = 0x00064801

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

Name	Bit	Type	Description
IDCODE/ID_KEY	[31:16]	RW	对当前寄存器进行写入时，必须同时写入正确KEY值。只有在ID_KEY等于0xE11E时，写入才有效。
IDCODE	[15:8]	R	读取时，返回ID CODE（IP版本信息）
SWRST	[7:5]	W	SYSCON产生软件复位。效果和CPU通过软复位指令产生软件复位一致。 4h：产生系统复位。 其他：无效。
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。
写5h的CPU复位功能有bug，会是CPU一直复位，SPEC中删去此功能			

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	ESO_CMRST	ESO_CKM	LP_EMOEN	LP_ESOEN	LP_IMOEN	LP_ISOEN	SYSTICK	RSVD	IDLE_HCLK	IDLE_PCLK	RSVD		PLL	HFOSC	EMOSC	ESOSC	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	W	W	W	W	W	W	W	R	W	W	R	R	W	W	W	W	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。
ESO_CMRST	[17]	W	在ESO_CKM功能使能后，使能或禁止外部ESOSC失效复位。（缺省使能） 0h：无效。 1h：使能或禁止ESOSC失效产生系统复位。当禁止产生系统复位时，ESOSC失效后，会自动切换系统时钟到IMOSC上。
ESO_CKM	[16]	W	使能或禁止外部ESOSC监测功能（缺省禁止）。 0h：无效。 1h：使能或禁止ESOSC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_ESOEN	[14]	W	使能或禁止DEEP-SLEEP模式下ESOSC工作状态（缺省禁止）。 0h：无效。 1h：使能或禁止ESOSC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（缺省禁止）。 0h：无效。 1h：使能或禁止PLL。
HFOSC	[4]	W	使能或禁止HFOSC 内部振荡器（缺省禁止）。 0h：无效。 1h：使能或禁止指定振荡器。
EMOSC	[3]	W	使能或禁止EMOSC 外部振荡器（XIN和XOUT管脚功能必须已经在GPIO中进行预先配置）。 0h：无效。 1h：使能或禁止指定振荡器。
ESOSC	[2]	W	使能ESOSC 内部振荡器（缺省使能）。 0h：无效。 1h：使能指定振荡器。
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	ESO_CMRST	ESO_CKM	LP_EMOEN	LP_ESOEN	LP_IMOEN	LP_ISOEN	SYSTICK	RSVD	IDLE_HCLK	IDLE_PCLK	RSVD		PLL	HFOSC	EMOSC	ESOSC	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	W	W	W	W	W	W	W	R	W	W	R	R	W	W	W	W	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。
ESO_CMRST	[17]	W	在ESO_CKM功能使能后，使能或禁止外部ESOSC失效复位。（缺省使能） 0h：无效。 1h：使能或禁止ESOSC失效产生系统复位。当禁止产生系统复位时，ESOSC失效后，会自动切换系统时钟到IMOSC上。
ESO_CKM	[16]	W	使能或禁止外部ESOSC监测功能（缺省禁止）。 0h：无效。 1h：使能或禁止ESOSC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_ESOEN	[14]	W	使能或禁止DEEP-SLEEP模式下ESOSC工作状态（缺省禁止）。 0h：无效。 1h：使能或禁止ESOSC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（缺省禁止）。 0h：无效。 1h：使能或禁止PLL。
HFOSC	[4]	W	使能或禁止HFOSC 内部振荡器（缺省禁止）。 0h：无效。 1h：使能或禁止指定振荡器。
EMOSC	[3]	W	使能或禁止EMOSC 外部振荡器（XIN和XOUT管脚功能必须已经在GPIO中进行预先配置）。 0h：无效。 1h：使能或禁止指定振荡器。
ESOSC	[2]	W	使能或禁止ESOSC 内部振荡器（缺省使能）。 0h：无效。 1h：使能或禁止指定振荡器。
IMOSC	[1]	W	使能或禁止IMOSC 内部振荡器（缺省使能）。 0h：无效。 1h：使能或禁止指定振荡器。
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	ESO_CMRST	ESO_CKM	LP_EMOEN	LP_ESOEN	LP_IMOEN	LP_ISOEN	SYSTICK	RSVD	IDLE_HCLK	IDLE_PCLK	RSVD		PLL	HFOSC	EMOSC	ESOSC	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。
ESO_CMRST	[17]	R	ESOSC 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。
ESO_CKM	[16]	R	ESOSC Clock Monitor功能状态。 0h：ESO CKM被关闭。 1h：ESO 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_ESOEN	[14]	R	DEEP-SLEEP模式下ESOSC工作状态。 0h：ESOSC被关闭。 1h：ESOSC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。
HFOSC	[4]	R	HFOSC 内部振荡器工作状态。 0h：HFOSC被关闭。 1h：HFOSC被打开。
EMOSC	[3]	R	EMOSC 振荡器工作状态。 0h：EMOSC被关闭。 1h：EMOSC被打开。
ESOSC	[2]	R	ESOSC 振荡器工作状态。 0h：ESOSC被关闭。 1h：ESOSC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	HFOSC	EMOSC	ESOSC	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 荡器稳定状态。 0h：PLL时钟未稳定。 1h：PLL时钟已稳定。
HFOSC	[4]	R	HFOSC 内部振荡器稳定状态。 0h：HFOSC时钟未稳定。 1h：HFOSC时钟已稳定。
EMOSC	[3]	R	EMOSC 振荡器稳定状态。 0h：EMOSC时钟未稳定。 1h：EMOSC时钟已稳定。
ESOSC	[2]	R	ESOSC 振荡器稳定状态。 0h：ESOSC时钟未稳定。 1h：ESOSC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：IMOSC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_RAMCHK(SRAM 校验控制寄存器)

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

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

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

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

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

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

Name	Bit	Type	Description
CHKEN	[31:24]	RW	Flash校验使能控制位。 5Ah：使能Flash的crc校验功能。 其他：关闭Flash的校验功能。
CHKTIMES	[23:0]	RW	校验重试次数设置。 设置Flash校验错后的重试次数。如果Flash校验发生错误，Flash控制器会再次重读当前地址，进行校验，重读的次数由该寄存器设置。重读次数满足该寄存器设置的次数后，如果仍然失败，系统会产生相应系统复位(由CHKEN位控制)。
NOTE: 1）该配置寄存器缺省复位值可以由User Option进行设置。			

5.3.9 SYSCON_SCLKCR(系统时钟控制寄存器)

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

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

Name	Bit	Type	Description
SCLK_KEY	[31:16]	W	
SCLK_DIV	[11:8]	RW	系统时钟分频设置（SYSCLK分频后，即为HCLK频率）。 0h：不分频。 1h：不分频。 2h：2分频。
SCLK_SEL	[2:0]	RW	系统时钟源控制，选择当前系统时钟SYSCLK工作的时钟。 0h：IMOSC作为系统工作时钟源。 1h：EMOSC作为系统工作时钟源。 2h：HFOSC作为系统工作时钟源。 3h：PLL作为系统时钟源 4h：ISOSC作为系统工作时钟源。 5h：ESOSC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。			

5.3.10 SYSCON_PCLKCR(外设时钟控制寄存器)

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

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

Name	Bit	Type	Description
PCLK_KEY	[31:16]	W	
PCLK_DIV	[11:8]	W	PCLK的时钟分频设置，PCLK分频基于HCLK的频率。 0000B：不分频。 0001B：2分频。 001xB：4分频。 01xxB：8分频。 1xxxB：16分频。
NOTE: 1）对该配置寄存器写入时，需要同时高位写入相应PCLK_KEY，KEY值不为0xC33C时，写入无效。			

5.3.11 SYSCON_EXIFLTCR(外设时钟控制寄存器)

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
EXIFLT_GRP19				EXIFLT_GRP18				EXIFLT_GRP17				EXIFLT_GRP16				EXIFLT_GRP3				EXIFLT_GRP2				EXIFLT_GPR1				EXIFLT_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
RW	RW	RW	RW	RW	RW	RW	RW	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
EXIFLT_GRP19~EXIFLT_GRP2	[31:8]	RW	外部中断滤波器参数选择 0000：无滤波 0001: 3/HFOSC 0010: 6/HFOSC 0011: 9/HFOSC 0100: 12/HFOSC 0101: 18/HFOSC 0110: 24/HFOSC 0111: 36/HFOSC 1000: 48/HFOSC 1001: 60/HFOSC 1010: 72/HFOSC 1011: 84/HFOSC 1100: 96/HFOSC 1101: 120/HFOSC 1110: 168/HFOSC 1111: 216/HFOSC
EXIFLT_GPR1	[7:4]	RW	
EXIFLT_GRP0	[3:0]	RW	

5.3.12 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							CAN	RSVD	I2C	SIO1	SIO0	RSVD		SPI1	SPI0	RSVD	USART1	USART0	RSVD		UART2	UART1	UART0	ETCB	RSVD	ADC1	ADC0	FVR	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	W	R	W	W	W	R	R	W	W	R	W	W	R	R	W	W	W	W	R	W	W	W	R	R	W

Name	Bit	Type	Description
CAN	[24]	W	
I2C	[22]	W	
SIO1~SIO0	[21:20]	W	
SPI1~SPI0	[17:16]	W	
USART1~USART0	[14:13]	W	
UART2~UART0	[10:8]	W	
ETCB	[7]	W	
ADC1~ADC0	[5:4]	W	
FVR	[3]	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。

5.3.13 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							CAN	RSVD	I2C	SIO1	SIO0	RSVD		SPI1	SPI0	RSVD	USART1	USART0	RSVD		UART2	UART1	UART0	ETCB	RSVD	ADC1	ADC0	FVR	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	W	R	W	W	W	R	R	W	W	R	W	W	R	R	W	W	W	W	R	W	W	W	R	R	W

Name	Bit	Type	Description
CAN	[24]	W	
I2C	[22]	W	
SIO1~SIO0	[21:20]	W	
SPI1~SPI0	[17:16]	W	
USART1~USART0	[14:13]	W	
UART2~UART0	[10:8]	W	
ETCB	[7]	W	
ADC1~ADC0	[5:4]	W	
FVR	[3]	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。

5.3.14 SYSCON_PCSR0(外设时钟状态寄存器0)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD							CAN	RSVD	I2C	SIO1	SIO0	RSVD		SPI1	SPI0	RSVD	USART1	USART0	RSVD		UART2	UART1	UART0	ETCB	RSVD	ADC1	ADC0	FVR	RSVD		IFC
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
CAN	[24]	R	
I2C	[22]	R	
SIO1~SIO0	[21:20]	R	
SPI1~SPI0	[17:16]	R	
USART1~USART0	[14:13]	R	
UART2~UART0	[10:8]	R	
ETCB	[7]	R	
ADC1~ADC0	[5:4]	R	
FVR	[3]	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。			

5.3.15 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
BT3	BT2	RSVD			OPA3	OPA2	OPA1	OPA0	RSVD		CMP2	CMP1	CMP0	LED	BT1	BT0	RTC	CNTA	LPT	WDT	GPTB5	GPTB4	GPTB3	GPTB2	GPTB1	GPTB0	GPTA3	GPTA2	GPTA1	GPTA0	RSVD
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	R	R	R	W	W	W	W	R	R	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	R

Name	Bit	Type	Description
BT3~BT2	[31:30]	W	
OPA3~OPA0	[26:23]	W	
CMP2~CMP0	[20:18]	W	
LED	[17]	W	
BT1~BT0	[16:15]	W	
RTC	[14]	W	
CNTA	[13]	W	
LPT	[12]	W	
WDT	[11]	W	
GPTB5~GPTB0	[10:5]	W	
GPTA3~GPTA0	[4:1]	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。			

5.3.16 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
BT3	BT2	RSVD			OPA3	OPA2	OPA1	OPA0	RSVD		CMP2	CMP1	CMP0	LED	BT1	BT0	RTC	CNTA	LPT	WDT	GPTB5	GPTB4	GPTB3	GPTB2	GPTB1	GPTB0	GPTA3	GPTA2	GPTA1	GPTA0	RSVD
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	R	R	R	W	W	W	W	R	R	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	R

Name	Bit	Type	Description
BT3~BT2	[31:30]	W	
OPA3~OPA0	[26:23]	W	
CMP2~CMP0	[20:18]	W	
LED	[17]	W	
BT1~BT0	[16:15]	W	
RTC	[14]	W	
CNTA	[13]	W	
LPT	[12]	W	
WDT	[11]	W	
GPTB5~GPTB0	[10:5]	W	
GPTA3~GPTA0	[4:1]	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。			

5.3.17 SYSCON_PCSR1(外设时钟状态寄存器1)

Address = Base Address+ 0x03C, Reset Value = 0x01000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BT3	BT2	RSVD			OPA3	OPA2	OPA1	OPA0	RSVD		CMP2	CMP1	CMP0	LED	BT1	BT0	RTC	CNTA	LPT	WDT	GPTB5	GPTB4	GPTB3	GPTB2	GPTB1	GPTB0	GPTA3	GPTA2	GPTA1	GPTA0	RSVD
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
BT3~BT2	[31:30]	R	
OPA3~OPA0	[26:23]	R	
CMP2~CMP0	[20:18]	R	
LED	[17]	R	
BT1~BT0	[16:15]	R	
RTC	[14]	R	
CNTA	[13]	R	
LPT	[12]	R	
WDT	[11]	R	
GPTB5~GPTB0	[10:5]	R	
GPTA3~GPTA0	[4:1]	R	
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。			

5.3.18 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
ES_GMCTL				EM_FLTSEL		EM_FLTEN	ES_SMTDIS	ESCNT								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
RW	RW	RW	RW	RW	RW	RW	RW	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
ES_GMCTL	[31:28]	RW	外部副晶振的增益控制位。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_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
ES_SMTDIS	[24]	RW	外部副晶振的斯密特滤波使能控制位。斯密特滤波可以提升振荡器的抗干扰性能，但具有额外功耗。 0h：使能斯密特滤波。 1h：关闭斯密特滤波。
ESCNT	[23:16]	RW	外部副晶振的时钟稳定计数器
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

5.3.19 SYSCON_PLLCR(PLL控制器)

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					CKQDIV		RSVD		CKPDIV						CLKEN	UNLCKRST	NUL							RSVD		DIVM				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	RW	RW	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CKQDIV	[26:25]	RW	CKQ 端口的VCO分频输出设置 0h：div2 1h：div4 2h：div6 3h：div8
CKPDIV	[22:17]	RW	CKP 端口的VCO分频输出设置 MSB控制位决定采用TYPEA或者TYPEB分频机制中的一种。 0h：TYPEA 分频。 1h：TYPEB 分频。 TYPEA 配置：Fckp = Fvco / (CKPDIV[4:0] + 1) TYPEB 配置： 0h：和 TYPEA 相同的配置公式。 1h：div2 2h：div6 3h：div8
CLKEN	[16]	RW	PLL 端口输出使能控制。 0h：禁止PLLCLK的时钟输出。 1h：打开PLLCLK的时钟输出。
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。
NUL	[14:8]	RW	VCO倍频设置。Fvco在150MHz~300MHz Fvco = Fin x N(NUL)
DIVM	[5:2]	RW	PLL输入时钟的预分频控制。VCO的最大输入频率范围：4MHz ~ 8MHz Fin = Fcksrc / (M + 1)
CKSEL	[1:0]	RW	PLL时钟输入源选择控制位 0h：Disabled 1h：Disabled 2h：HFOSC 选择作为时钟源

			3h：EMOSC 选择作为时钟源

5.3.20 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：2.1V 1h：2.4V 2h：2.7V 3h：3.0V 4h：3.3V 5h：3.6V 6h：3.9V 7h：EXTIN_1.0V 注1：比较电压为VDD电压和选择的电压进行比较。 注2：EXTIN_1.0V，外部输入1.0V电压；使用时对应输入引脚选择LVDIN复用功能
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。			

5.3.21 SYSCON_CLCR(内部振荡器调整寄存器)

Address = Base Address+ 0x050, Reset Value = 0xXXXX00XX

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

Name	Bit	Type	Description
ISO_TRM	[31:24]	RW	ISOSC 的 TRIM 调整位 ISOSC 的频率输出随着调整值递增，值越大，频率越高。
IMO_TRM	[23:16]	RW	IMOSC 的 TRIM 调整位 IMOSC 的频率输出随着调整值递增，值越大，频率越高。
HFO_TRM	[6:0]	RW	HFOSC 的 TRIM 调整位 HFOSC 的频率输出随着调整值递增，值越大，频率越高。

5.3.22 SYSCON_PWRCR(功耗控制寄存器)

Address = Base Address+ 0x054, Reset Value = 0x0f070040

31	30	29	28	27	26	25	24	23	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	0	1	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
R	R	R	R	RW	RW	RW	RW	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
DSL_CFG	[27:24]	RW	Power srtategy under deep-sleep mode bit0: VCref(VDDCORE reference) selection 0h: BGR(1.0V) 1h: CMOS(0.9V) bit1: VDDCORE control 0h: VDDCORE = VCref x 1.2 1h: VDDCORE = VCref x 1.0 bit2: SLP ctl of main reg. 0h: disable 1h: enable: main reg is working in lowspeed mode whcih consumes less power bit3: pd ctl of main reg. 0h: enable 1h: power-down: fix VDDCORE = VCref x 1.0 with extra low power consumption but weak driving capability
SLP_CFG	[19:16]	RW	Power srtategy under sleep mode bit0: VCref(VDDCORE reference) selection 0h: BGR(1.0V) 1h: CMOS(0.9V) bit1: VDDCORE control 0h: VDDCORE = VCref x 1.2 1h: VDDCORE = VCref x 1.0 bit2: SLP ctl of main reg. 0h: disable 1h: enable: main reg is working in lowspeed mode whcih consumes less power bit3: pd ctl of main reg. 0h: enable 1h: power-down: fix VDDCORE = VCref x 1.0 with extra low power consumption but weak driving capability
RUN_CFG	[11:8]	RW	Power srtategy under normal run mode bit0: VCref(VDDCORE reference) selection 0h: BGR(1.0V) 1h: CMOS(0.9V) bit1: VDDCORE control 0h: VDDCORE = VCref x 1.2

			1h: VDDCORE = VCref x 1.0 bit2: SLP ctl of main reg. 0h: disable 1h: enable: main reg is working in lowspeed mode whcih consumes less power bit3: pd ctl of main reg. 0h: enable 1h: power-down: fix VDDCORE = VCref x 1.0 with extra low power consumption but weak driving capability
VOSEN	[3:0]	RW	VOS模式使能控制。（1） xxx1b：使能RUN模式下的VOS xx1xb：使能SLEEP模式下的VOS x1xxb：使能DEEP-SLEEP模式下的VOS
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。			

5.3.23 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。			

5.3.24 SYSCON_OPT1(系统配置寄存器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	NMI_SEL				SRAMBLKCTRL	RSVD	SRAM1FUNCCTRL	EMCKM_DUR			ESCKM_DUR		RSVD			EFL_LPMD	CLODIV			CLOMX				RSVD		HFO_FSEL		RSVD		IMO_FSEL	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW	R	R	RW	RW

Name	Bit	Type	Description
NMI_SEL	[30:27]	RW	NMI触发源选择控制位。通过相应控制位独立选择触发源的使能。 0h：禁止该触发源触发NMI 1h：使能该触发源触发NMI bit0：LVD中断事件 bit1：内存错误事件（SRAM或者FLASH校验异常） bit2：EXI0中断事件 bit3：外部时钟异常事件
SRAMBLKCTRL	[26]	RW	SRAM 分区控制。SRAM0始终为DSRAM，SRAM1的功能受OPT1[24]位控制 0h： SRAM0 24K。 地址范围0x20000000~0x20005FFF SRAM1 8K。 地址范围0x20006000~0x20007FFF 1h: SRAM0 16K。 地址范围0x20000000~0x20003FFF SRAM1 16K。 地址范围0x20004000~0x20007FFF
SRAM1FUNCCTRL	[24]	RW	SRAM1指令存取功能控制 0h：SRAM1作为DSRAM用（作为普通SRAM使用） 1h：SRAM1作为ISRAM用，不可写（作为指令空间使用）
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
ESCKM_DUR	[20:19]	RW	ESCKM的检测时间间隔设置，检测周期基于当前IMOSC的频率设置。 0h：40个周期

			1h：30个周期 2h：20个周期 3h：10个周期
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。 2h：ESCLK输出。 3h：EMCLK输出。 4h：HFCLK输出。 6h：RTCCLK输出。 7h：PCLK输出。 8h：HCLK输出。 9h：IWDTCLK输出。 Ah：PLL_PCLK输出。 Bh：PLL_QCLK输出。 Dh：SYSCLK输出。 其他：保留。
HFO_FSEL	[5:4]	RW	HFOSC频率选择。 0h：24MHz。 1h：12MHz。 2h：6MHz。 3h：3MHz。
IMO_FSEL	[1:0]	RW	IMOSC频率选择。 0h：5.556MHz。 1h：4.194MHz。 2h：2.097MHz。 3h：131.072KHz。

5.3.25 SYSCON_OPT0(系统配置寄存器0)

Address = Base Address+ 0x068, 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				RDP	RSVD						HDP_IFE	HDP_32K	HDP_16K	HDP_8K	HDP_ALL	RSVD							DBP	CIPVALID	RSVD			CKHEN	CPUFTRST	EXIRSTEN	IWDTEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
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_IFE	[20]	R	取址保护使能状态查询
HDP_32K	[19]	R	32K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_16K	[18]	R	16K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_8K	[17]	R	8K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。
CKHEN	[3]	R	上电后，Flash的crc校验功能使能状态
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。

5.3.26 SYSCON_WKCR(低功耗唤醒使能控制寄存器)

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

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

Name	Bit	Type	Description
LVD_WKEN	[11]	RW	LVD中断唤醒DEEP-SLEEP使能控制位。 0h：禁止LVD中断唤醒DEEP-SLEEP。 1h：使能LVD中断唤醒DEEP-SLEEP。
LPT_WKEN	[10]	RW	LPT中断唤醒DEEP-SLEEP使能控制位。 0h：禁止LPT中断唤醒DEEP-SLEEP。 1h：使能LPT中断唤醒DEEP-SLEEP。
RTC_WKEN	[9]	RW	RTC中断唤醒DEEP-SLEEP使能控制位。 0h：禁止RTC中断唤醒DEEP-SLEEP。 1h：使能RTC中断唤醒DEEP-SLEEP。
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。			

5.3.27 SYSCON_IMER(中断使能控制寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		CMD_ERR	RSVD						EV3TRG	EV2TRG	EV1TRG	EV0TRG	EM_CMFAIL	CQCDN	ES_CMFALL	PLLUL	OPL_ERR	EFL_ERR	RSVD	LVD_INT	RAM_ERR	RSVD	IWDT_INT	SYSCLK_ST	RSVD	PLL_ST	HFOSC_ST	EMOSC_ST	ESOSC	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	W	W	W	W	W	R	W	W	R	W	W	R	W	W	W	W	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	事件触发输出Event3 触发的中断
EV2TRG	[21]	W	事件触发输出Event2 触发的中断
EV1TRG	[20]	W	事件触发输出Event1 触发的中断
EV0TRG	[19]	W	事件触发输出Event0 触发的中断
EM_CMFAIL	[18]	W	EMOSC时钟失效中断
CQCDN	[17]	W	Clock Qualification完成中断
ES_CMFALL	[16]	W	ESOSC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
EFL_ERR	[13]	W	EFLASH校验失败中断
LVD_INT	[11]	W	LVD中断
RAM_ERR	[10]	W	SRAM校验失败中断
IWDT_INT	[8]	W	IWDT中断
SYSCLK_ST	[7]	W	SYSCLK时钟稳定中断
PLL_ST	[5]	W	PLL时钟稳定中断
HFOSC_ST	[4]	W	HFOSC时钟稳定中断
EMOSC_ST	[3]	W	EMOSC时钟稳定中断
ESOSC	[2]	W	ESOSC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_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	CQCDN	ES_CMFALL	PLLUL	OPL_ERR	EFL_ERR	RSVD	LVD_INT	RAM_ERR	RSVD	IWDT_INT	SYSCLK_ST	RSVD	SYPLL	HFOSC_ST	EMOSC_ST	ESOSC_ST	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	W	W	W	W	W	R	W	W	R	W	W	R	W	W	W	W	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
CQCDN	[17]	W	Clock Qualification完成中断
ES_CMFALL	[16]	W	ESOSC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。
EFL_ERR	[13]	W	EFLASH校验失败中断。
LVD_INT	[11]	W	LVD中断。
RAM_ERR	[10]	W	SRAM校验失败中断。
IWDT_INT	[8]	W	IWDT中断。
SYSCLK_ST	[7]	W	SYSCLK时钟稳定中断。
SYPLL	[5]	W	PLL时钟稳定中断
HFOSC_ST	[4]	W	HFOSC时钟稳定中断。
EMOSC_ST	[3]	W	EMOSC时钟稳定中断。
ESOSC_ST	[2]	W	ESOSC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_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	CQCDN	ES_CMFALL	PLLUL	OPL_ERR	EFL_ERR	RSVD	LVD_INT	RAM_ERR	RSVD	IWDT_INT	SYSCLK_ST	RSVD	SYPLL	HFOSC_ST	EMOSC_ST	ESOSC_ST	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	RW	RW	RW	RW	RW	R	RW	RW	R	RW	RW	R	RW	RW	RW	RW	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。
CQCDN	[17]	RW	CQCDN校验失败中断。 0h：禁止中断。 1h：使能中断。
ES_CMFALL	[16]	RW	ES_CMFALL校验失败中断。 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。
EFL_ERR	[13]	RW	EFLASH校验失败中断。

			0h：禁止中断。 1h：使能中断。
LVD_INT	[11]	RW	LVD中断。 0h：禁止中断。 1h：使能中断。
RAM_ERR	[10]	RW	SRAM校验失败中断。 0h：禁止中断。 1h：使能中断。
IWDT_INT	[8]	RW	IWDT中断。 0h：禁止中断。 1h：使能中断。
SYSCLK_ST	[7]	RW	SYSCLK时钟稳定中断。 0h：禁止中断。 1h：使能中断。
SYPLL	[5]	RW	PLL时钟稳定中断。 0h：禁止中断。 1h：使能中断。
HFOSC_ST	[4]	RW	HFOSC时钟稳定中断。 0h：禁止中断。 1h：使能中断。
EMOSC_ST	[3]	RW	EMOSC时钟稳定中断。 0h：禁止中断。 1h：使能中断。
ESOSC_ST	[2]	RW	ESOSC时钟稳定中断。 0h：禁止中断。 1h：使能中断。
IMOSC_ST	[1]	RW	IMOSC时钟稳定中断。 0h：禁止中断。 1h：使能中断。
ISOSC_ST	[0]	RW	ISOSC时钟稳定中断。 0h：禁止中断。 1h：使能中断。

5.3.30 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	CQCDN	ES_CMFALL	PLLUL	OPL_ERR	EFL_ERR	RSVD	LVD_INT	RAM_ERR	RSVD	IWDT_INT	SYSCLK_ST	RSVD	SYPLL	HFOSC_ST	EMOSC_ST	ESOSC_ST	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	W	W	W	W	W	R	W	W	R	W	W	R	W	W	W	W	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
CQCDN	[17]	W	Clock Qualification完成中断
ES_CMFALL	[16]	W	ESOSC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。
EFL_ERR	[13]	W	EFLASH校验失败中断。
LVD_INT	[11]	W	LVD中断。
RAM_ERR	[10]	W	SRAM校验失败中断。
IWDT_INT	[8]	W	IWDT中断。
SYSCLK_ST	[7]	W	SYSCLK时钟稳定中断。
SYPLL	[5]	W	PLL时钟稳定中断
HFOSC_ST	[4]	W	HFOSC时钟稳定中断。
EMOSC_ST	[3]	W	EMOSC时钟稳定中断。
ESOSC_ST	[2]	W	ESOSC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。

5.3.31 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	CQCDN	ES_CMFALL	PLLUL	OPL_ERR	EFL_ERR	RSVD	LVD_INT	RAM_ERR	RSVD	IWDT_INT	SYSCLK_ST	RSVD	PLL_ST	HFOSC_ST	EMOSC_ST	ESOSC_ST	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	W	W	W	W	W	R	W	W	R	W	W	R	W	W	W	W	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	事件触发输出Event3 触发的中断
EV2TRG	[21]	W	事件触发输出Event2 触发的中断
EV1TRG	[20]	W	事件触发输出Event1 触发的中断
EV0TRG	[19]	W	事件触发输出Event0 触发的中断
EM_CMFAIL	[18]	W	EMOSC时钟失效中断
CQCDN	[17]	W	Clock Qualification完成中断
ES_CMFALL	[16]	W	ESOSC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
EFL_ERR	[13]	W	EFLASH校验失败中断
LVD_INT	[11]	W	LVD中断
RAM_ERR	[10]	W	SRAM校验失败中断
IWDT_INT	[8]	W	IWDT中断
SYSCLK_ST	[7]	W	SYSCLK时钟稳定中断
PLL_ST	[5]	W	PLL时钟稳定中断
HFOSC_ST	[4]	W	HFOSC时钟稳定中断
EMOSC_ST	[3]	W	EMOSC时钟稳定中断
ESOSC_ST	[2]	W	ESOSC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

5.3.32 SYSCON_RISR(原始中断标志状态寄存器)

Address = Base Address+ 0x088, Reset Value = 0x00000083

31	30	29	28	27	26	25	24	23	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	CQCDN	ES_CMFALL	PLLUL	OPL_ERR	EFL_ERR	RSVD	LVD_INT	RAM_ERR	RSVD	IWDT_INT	SYSCLK_ST	RSVD	PLL_ST	HFOSC_ST	EMOSC_ST	ESOSC_ST	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	1	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
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
CQCDN	[17]	R	Clock Qualification完成中断
ES_CMFALL	[16]	R	ESOSC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
EFL_ERR	[13]	R	EFLASH校验失败中断
LVD_INT	[11]	R	LVD中断
RAM_ERR	[10]	R	SRAM校验失败中断
IWDT_INT	[8]	R	IWDT中断
SYSCLK_ST	[7]	R	SYSCLK时钟稳定中断
PLL_ST	[5]	R	PLL时钟稳定中断
HFOSC_ST	[4]	R	HFOSC时钟稳定中断
EMOSC_ST	[3]	R	EMOSC时钟稳定中断
ESOSC_ST	[2]	R	ESOSC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

5.3.33 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	CQCDN	ES_CMFALL	PLLUL	OPL_ERR	EFL_ERR	RSVD	LVD_INT	RAM_ERR	RSVD	IWDT_INT	SYSCLK_ST	RSVD	PLL_ST	HFOSC_ST	EMOSC_ST	ESOSC_ST	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
CQCDN	[17]	R	Clock Qualification完成中断
ES_CMFALL	[16]	R	ESOSC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
EFL_ERR	[13]	R	EFLASH校验失败中断
LVD_INT	[11]	R	LVD中断
RAM_ERR	[10]	R	SRAM校验失败中断
IWDT_INT	[8]	R	IWDT中断
SYSCLK_ST	[7]	R	SYSCLK时钟稳定中断
PLL_ST	[5]	R	PLL时钟稳定中断
HFOSC_ST	[4]	R	HFOSC时钟稳定中断
EMOSC_ST	[3]	R	EMOSC时钟稳定中断
ESOSC_ST	[2]	R	ESOSC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

5.3.34 SYSCON_RSR(复位源记录状态寄存器)

Address = Base Address+ 0x090, Reset Value = 0x00000080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																CPU1RST	RSVD	WWDT	EFL_ERR	RAM_ERR	PLLUNLOCK	CPUFAULT	SWRST	CPURST	EMCKM	ESCKM	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	1	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	R	RW	RW	RW

Name	Bit	Type	Description
CPU1RST	[15]	RW	CPU产生的软件复位（CPU CORE复位）。
WWDT	[13]	RW	WWDT复位。
EFL_ERR	[12]	RW	EFLASH校验错误产生的系统复位。
RAM_ERR	[11]	RW	SRAM校验错误产生的系统复位。
PLLUNLOCK	[10]	R	PLL失锁产生的系统复位
CPUFAULT	[9]	RW	CPU锁定产生的系统复位。
SWRST	[8]	RW	SYSCON产生软件复位（系统级复位）。
CPURST	[7]	RW	CPU产生的软件复位（系统级复位）。
EMCKM	[6]	RW	EMOSC CKM Fail产生的系统复位。
ESCKM	[5]	RW	ESOSC CKM Fail产生的系统复位。
IWDT	[4]	RW	IWDT产生的系统复位。
EXTRST	[2]	RW	外部复位管脚复位。
LVR	[1]	RW	LVD产生的系统复位。
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。			

5.3.35 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

5.3.36 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

5.3.37 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	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
EXI0 ~ EXI19	[19:0]		

5.3.38 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	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
EXI0 ~ EXI19	[19:0]		

5.3.39 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	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
EXI0 ~ EXI19	[19:0]		

5.3.40 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]		

5.3.41 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]		

5.3.42 SYSCON_EXIRS(外部中断原始标志状态寄存器)

Address = Base Address+ 0x0B0, Reset Value = 0x80000000

31	30	29	28	27	26	25	24	23	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																			
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	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]		

5.3.43 SYSCON_IWDCR(独立看门狗控制寄存器)

Address = Base Address+ 0x0B4, Reset Value = 0x0000170C

31	30	29	28	27	26	25	24	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	1	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	RW	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]	RW	看门狗工作状态。 0h：看门狗未使能。 1h：看门狗已使能。
DBGEN	[11]	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。
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。			

5.3.44 SYSCON_IWDCNT(独立看门狗控制计数器值)

Address = Base Address+ 0x0B8, Reset Value = 0x00000ffd

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

5.3.45 SYSCON_IWDEDR(独立看门狗使能寄存器)

Address = Base Address+ 0x0BC, 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
IWDTE_KEY																ENDIS															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
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。

5.3.46 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 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。

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

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

5.3.48 SYSCON_FINF0(工厂信息寄存器0)

Address = Base Address+ 0x0D0, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FINF																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
FINF	[31:0]	R	工厂信息寄存器,显示工厂信息

5.3.49 SYSCON_FINF1(工厂信息寄存器1)

Address = Base Address+ 0x0D4, Reset Value = 0x00000000

5.3.50 SYSCON_FINF2(工厂信息寄存器2)

Address = Base Address+ 0x0D8, Reset Value = 0x00000000

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

5.3.51 SYSCON_ERRINF(错误信息寄存器)

Address = Base Address+ 0x0E0, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																ERR_WDTREF	ERR_EMCKMEN	ERR_PLLEN	ERR_PLLDIS	ERR_ESCKMEN	ERR_HFOSCDIS	ERR_EMOSCEN	ERR_EMOSCDIS	ERR_ESOSCDIS	ERR_IMOSCDIS	ERR_ISOSCDIS	ERR_SYSCKSEL	ERR_CQS	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
ERR_WDTREF	[15]	R	尝试重载iWDT时发生错误，可能ISOSC未稳定。
ERR_EMCKMEN	[14]	R	尝试打开EMCKM时发生错误，可能监测时钟未正确配置。
ERR_PLLEN	[13]	R	尝试打开PLL时发生错误，可能监测时钟未正确配置。
ERR_PLLDIS	[12]	R	尝试关闭PLL时发生错误，可能PLL正被其他模块占用。
ERR_ESCKMEN	[11]	R	尝试打开ESCKM时发生错误，可能监测时钟未正确配置。
ERR_HFOSCDIS	[10]	R	尝试关闭HFOSC时发生错误，可能HFOSC正被其他模块占用。
ERR_EMOSCEN	[9]	R	尝试打开EMOSC时发生错误，可能EMOSC管脚未正确配置。
ERR_EMOSCDIS	[8]	R	尝试关闭EMOSC时发生错误，可能EMOSC正被其他模块占用。
ERR_ESOSCDIS	[7]	R	尝试关闭ESOSC时发生错误，可能ESOSC正被其他模块占用。
ERR_IMOSCDIS	[6]	R	尝试关闭IMOSC时发生错误，可能IMOSC正被其他模块占用。
ERR_ISOSCDIS	[5]	R	尝试关闭ISOSC时发生错误，可能ISOSC正被其他模块占用。
ERR_SYSCKSEL	[4]	R	尝试将SYSCLK选择为未使能的时钟源。
ERR_CQS	[3]	R	尝试启动CQC功能时发生错误，可能是所选时钟未稳定。
当前寄存器的值将在被读取后，自动清楚。			

5.3.52 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。

5.3.53 SYSCON_UID1(UID寄存器1)

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

5.3.54 SYSCON_UID2(UID寄存器2)

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

5.3.55 SYSCON_PWROPT(供电恢复时间调整寄存器)

Address = Base Address+ 0x0F0, Reset Value = 0x00000000

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

Name	Bit	Type	Description
KEY	[31:24]	W	
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。
SLPRCVTIM	[3:2]	RW	SLEEP低功耗模式唤醒稳定时间，Tslp为固定值 0h: Tslp+30us 1h: Tslp+20us 2h: Tslp+10us 3h: Tslp+5us
DSPRCVTIM	[1:0]	RW	DEEPSLEEP低功耗模式唤醒稳定时间，Tdsp为固定值 0h: Tdsp+30us 1h: Tdsp+20us 2h: Tdsp+10us 3h: Tdsp+5us
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。			

5.3.56 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。

5.3.57 SYSCON_EVPS(事件触发计数寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRGEV3CNT				TRGEV2CNT				TRGEV1CNT				TRGEV0CNT				TRGEV3PRD				TRGEV2PRD				TRGEV1PRD				TRGEV0PRD			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
TRGEV3CNT	[31:28]	R	当前TRGEVx事件计数器值。 读取时，返回当前事件计数器值。此计数器不能设置初始值，只能通过EVTRG的CNTCLR控制位进行清零。
TRGEV2CNT	[27:24]	R	当前TRGEVx事件计数器值。 读取时，返回当前事件计数器值。此计数器不能设置初始值，只能通过EVTRG的CNTCLR控制位进行清零。
TRGEV1CNT	[23:20]	R	当前TRGEVx事件计数器值。 读取时，返回当前事件计数器值。此计数器不能设置初始值，只能通过EVTRG的CNTCLR控制位进行清零。
TRGEV0CNT	[19:16]	R	当前TRGEVx事件计数器值。 读取时，返回当前事件计数器值。此计数器不能设置初始值，只能通过EVTRG的CNTCLR控制位进行清零。
TRGEV3PRD	[15:12]	RW	TRGEVx 的事件计数器周期设置。 当TRGEVx事件发生时，TRGEVxCNT递增一次，当TRGEVxCNT的计数值等于TRGEVxPRD设置周期时，产生TRGEVx触发事件。
TRGEV2PRD	[11:8]	RW	TRGEVx 的事件计数器周期设置。 当TRGEVx事件发生时，TRGEVxCNT递增一次，当TRGEVxCNT的计数值等于TRGEVxPRD设置周期时，产生TRGEVx触发事件。
TRGEV1PRD	[7:4]	RW	TRGEVx 的事件计数器周期设置。 当TRGEVx事件发生时，TRGEVxCNT递增一次，当TRGEVxCNT的计数值等于TRGEVxPRD设置周期时，产生TRGEVx触发事件。
TRGEV0PRD	[3:0]	RW	TRGEVx 的事件计数器周期设置。 当TRGEVx事件发生时，TRGEVxCNT递增一次，当TRGEVxCNT的计数值等于TRGEVxPRD设置周期时，产生TRGEVx触发事件。

5.3.58 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

5.3.59 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。			

5.3.60 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。
NOTE: 1）UREG0 和 UREG1为32位寄存器，UREG2为16位寄存器。			

5.3.61 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。
NOTE: 1）UREG0 和 UREG1为32位寄存器，UREG2为16位寄存器。			

5.3.62 SYSCON_CQCR(时钟源频率验证控制寄存器)

Address = Base Address+ 0x118, Reset Value = 0x00000000

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

Name	Bit	Type	Description
CQSVAL	[25:8]	RW	设置被测试（源）时钟计数停止值。
SRCSEL	[7:6]	RW	CQC 源时钟选择控制位（源时钟为需要进行频率检查的时钟） 0h：IMOSC 1h：ESOSC 2h：ISOSC 3h：HFOSC
REFSEL	[5:4]	RW	CQC 参考时钟选择控制位。 0h：EMOSC 1h：ESOSC 其他：保留
CQCREQ	[0]	RW	CQC启动控制位。 写入'1'时，启动CQC 读取时，返回当前CQC状态： 0h：CQC未启动 1h：CQC已启动

5.3.63 SYSCON_CQRR(时钟源频率验证计数器值)

Address = Base Address+ 0x11C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												CQRR																			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
CQRR	[19:0]	R	参考时钟计数器的目标计数值设置。 当CQC启动后，参考时钟计数器和源时钟计数器同时被触发。当源时钟计数器的计数值等于CQSVAL的设置值时，参考计数器的当前计数值将被载入到CQRR寄存器中。 当源时钟准确时，CQRR的值满足： CQRR/Fref = CQSVAL/Fsrc

5.3.64 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																								SWD_AFLOCK							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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
SWD_AFLOCK	[7:0]	RW	SWD AF功能LOCK控制寄存器。 缺省SWD功能的GPIO，在LOCK状态下，不用通过GPIO的CONR进行AF功能修改，必须解除LOCK后，才能进行修改。 0x5A：解除LOCK。 其他：LOCK状态。

5.3.65 SYSCON_BRPKEY(复位地址配置保护寄存器)

Address = Base Address+ 0x12C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																RBP_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
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
RBP_KEY	[15:0]	RW	SRBREG, ERBREG 更新保护寄存器。当对该寄存器写入0x9559后，才能更新SRB和ERB寄存器。每次更新前，必须对该KEY值进行写入。任何错误值的写入，或者对SRB或ERB的更新，都会自动清除RBP_KEY。

5.3.66 SYSCON_SRBREG(系统软件复位引导地址配置寄存器)

Address = Base Address+ 0x130, Reset Value = 0x00000000

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

Name	Bit	Type	Description
SRB	[11:0]	RW	系统软件复位地址配置寄存器。由SYSCON产生的系统复位发生后，CPU将从0x(SRB<<12)地址开始引导（即SRB只支持4K地址对齐方式进行配置） 该复位寄存器的设置值只有在SYSCON软件复位后，才会起作用。其他正常复位，例如POR或外部复位等，CPU都使用默认0x00000000作为引导初始地址。 对SRB的修改，必须先对BRPKEY进行写入。

5.3.67 SYSCON_ERBREG(系统异常复位引导地址配置寄存器)

Address = Base Address+ 0x134, Reset Value = 0x00000000

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

Name	Bit	Type	Description
ERB	[11:0]	RW	系统异常复位地址配置寄存器。异常复位产生后，CPU将从0x(ERB<<12)地址开始引导（即ERB只支持4K地址对齐方式进行配置） 该复位寄存器的设置值只有在系统异常复位后，才会起作用。其他正常复位，例如POR或外部复位等，CPU都使用默认0x00000000作为引导初始地址。异常复位源包括：WWDT, EFLERR, SRAMERR, PLLUNLOCK, CPUFAULT, EMCKM, IWDT, LVD。 对ERB的修改，必须先对BRPKEY进行写入。

6 事件触发控制器

6.1 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。

6.1.1 特性

- 32个可配置的事件通道
  - 通道0-2支持单个源触发多个目标
  - 通道3-31只支持单个源触发单个目标
  - 通道20-31支持DMA触发通道
- 支持软件触发

6.2 功能描述

6.2.1 模块框图

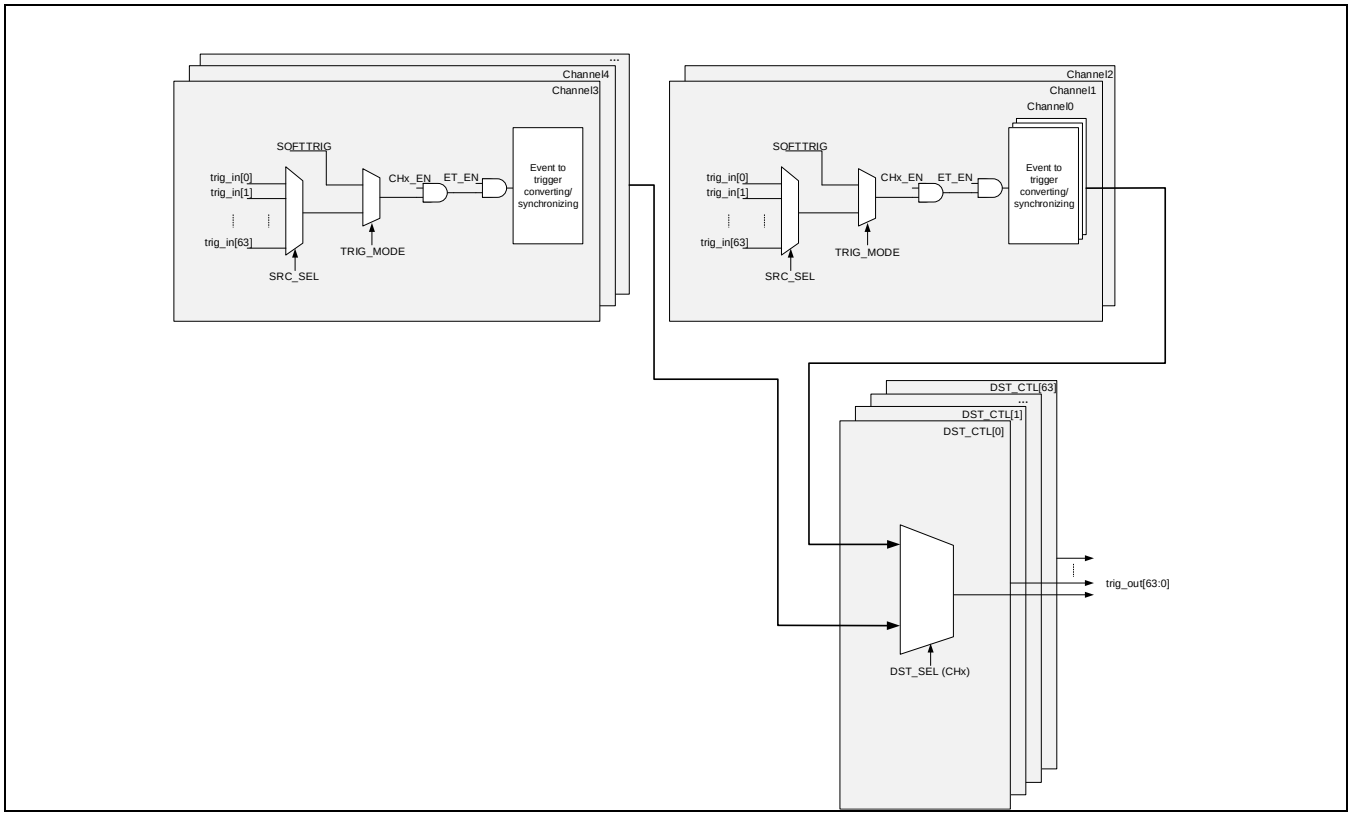


Figure 6-1 ET模块框图

6.2.2 主要功能

6.2.2.1 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 32 个通道。每个通道都可以由一个源来触发另一个目标。通道 0、通道 1 和通道 2 则可以用一个源来触发多个目标。如果需要使用 DMA 的触发功能，只能使用通道 20~31，并且在通道的配置寄存器中使能 DMA 功能。

注意：

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

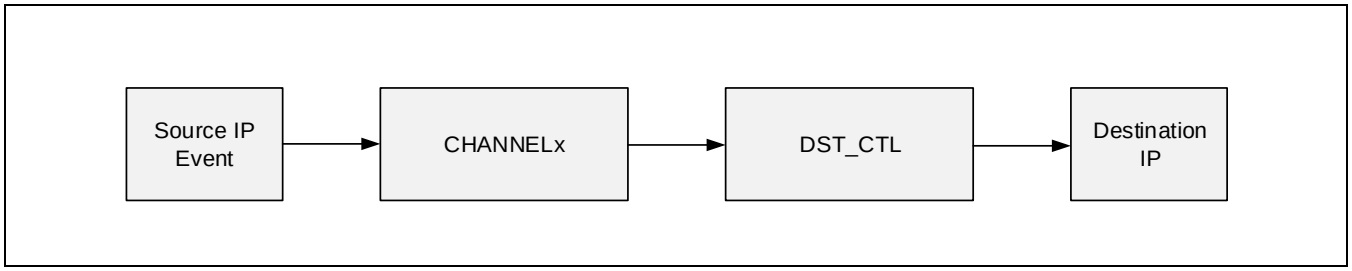


Figure 6-2 单个源触发单个目标

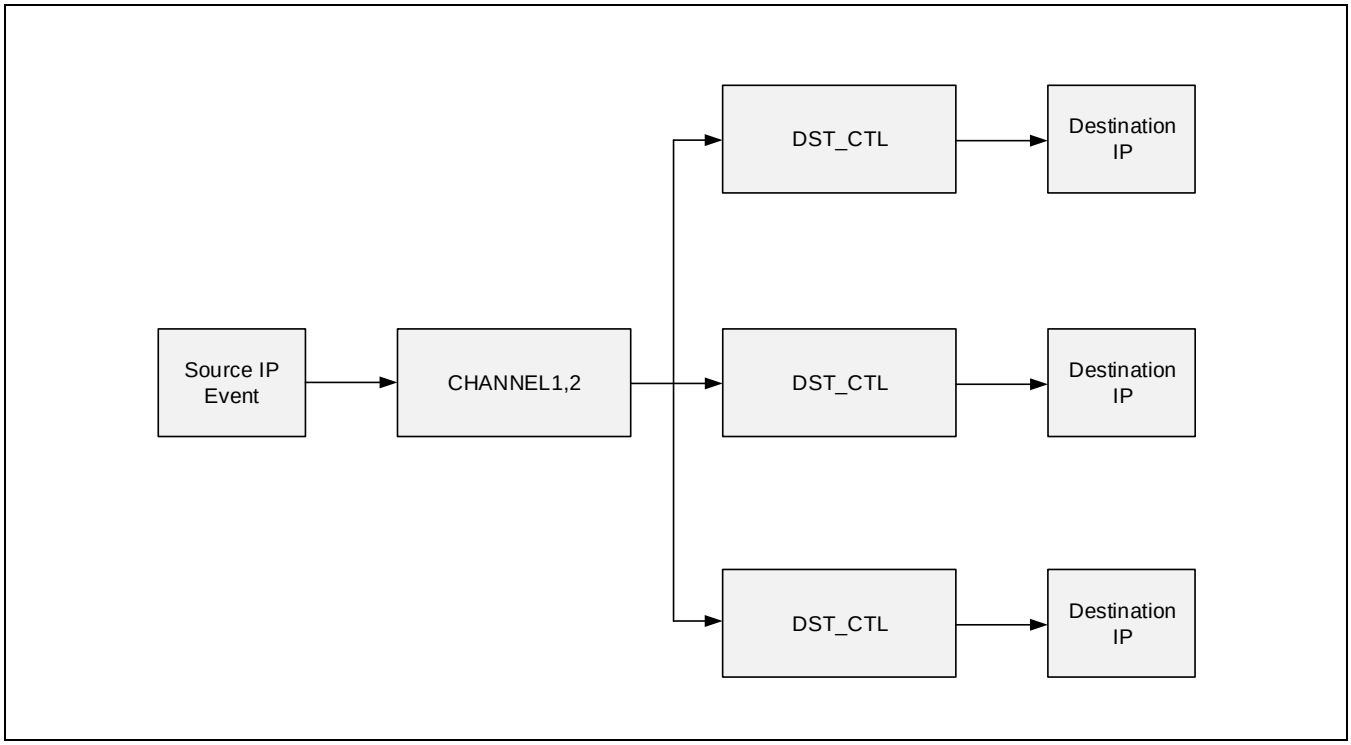


Figure 6-3 单个源触发多个目标

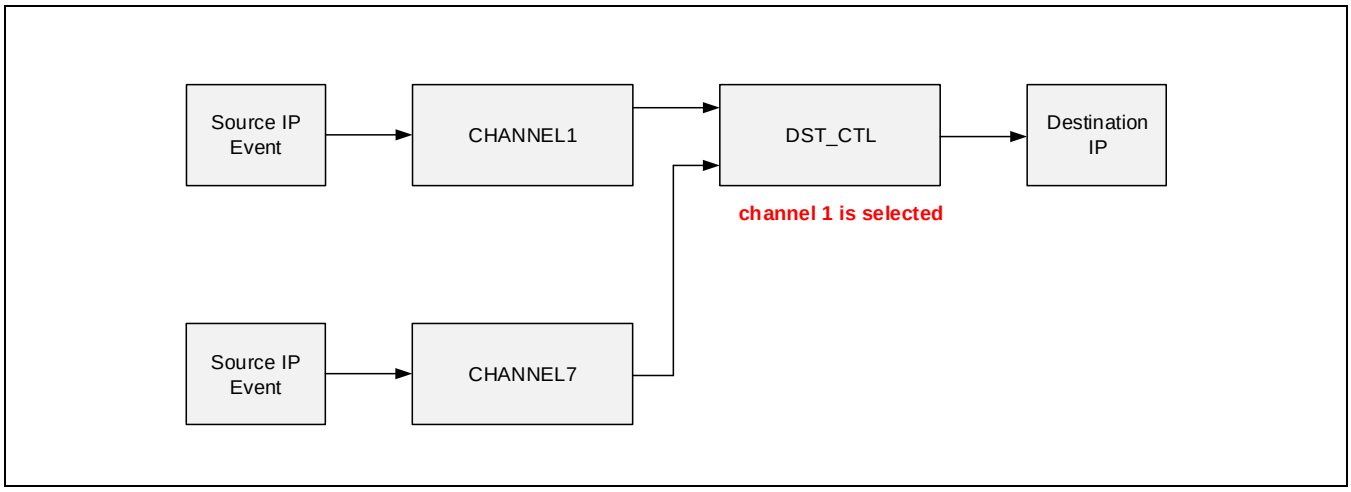


Figure 6-4 2个通道选择了相同的目标

6.2.2.2 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 6-1 事件对应表

源序号	事件源	目标序号	目标事件
0 （0H）	LPT_TRGOUT	0 （0H）	LPT_SYNCIN
1 （1H）	RSVD	1 （1H）	BT0_SYNCIN0
2 （2H）	RSVD	2 （2H）	BT0_SYNCIN1
3 （3H）	RSVD	3 （3H）	BT0_SYNCIN2
4 （4H）	EXI_TRGOUT0	4 （4H）	BT1_SYNCIN0
5 （5H）	EXI_TRGOUT1	5 （5H）	BT1_SYNCIN1
6 （6H）	EXI_TRGOUT2	6 （6H）	BT1_SYNCIN3
7 （7H）	EXI_TRGOUT3	7 （7H）	BT2_SYNCIN0
8 （8H）	RTC_TRGOUT0	8 （8H）	BT2_SYNCIN1
9 （9H）	RTC_TRGOUT1	9 （9H）	BT2_SYNCIN2
10 （AH）	BT0_TRGOUT	10 （AH）	BT3_SYNCIN0
11 （BH）	BT1_TRGOUT	11 （BH）	BT3_SYNCIN1
12 （CH）	BT2_TRGOUT	12 （CH）	BT3_SYNCIN2
13 （DH）	BT3_TRGOUT	13 （DH）	ADC0_ SYNCIN0
14 （EH）	RSVD	14 （EH）	ADC0_ SYNCIN1
15 （FH）	RSVD	15 （FH）	ADC0_ SYNCIN2
16 （10H）	RSVD	16 （10H）	ADC0_ SYNCIN3
17 （11H）	RSVD	17 （11H）	ADC0_ SYNCIN4
18 （12H）	RSVD	18 （12H）	ADC0_ SYNCIN5
19 （13H）	RSVD	19 （13H）	ADC0_ SYNCIN6
20 （14H）	RSVD	20 （14H）	ADC0_ SYNCIN7
21 （15H）	RSVD	21 （15H）	ADC0_ SYNCIN8
22 （16H）	RSVD	22 （16H）	ADC0_ SYNCIN9
23 （17H）	RSVD	23 （17H）	ADC0_ SYNCIN10
24 （18H）	RSVD	24 （18H）	ADC0_ SYNCIN11
25 （19H）	RSVD	25 （19H）	ADC0_ SYNCIN12
26 （1AH）	RSVD	26 （1AH）	ADC0_ SYNCIN13
27 （1BH）	RSVD	27 （1BH）	ADC0_ SYNCIN14
28 （1CH）	RSVD	28 （1CH）	ADC1_ SYNCIN0
29 （1DH）	RSVD	29 （1DH）	ADC1_ SYNCIN1
30 （1EH）	ADC0_TRGOUT0	30 （1EH）	ADC1_ SYNCIN2
31 （1FH）	ADC0_TRGOUT1	31 （1FH）	ADC1_ SYNCIN3
32 （20H）	ADC1_TRGOUT0	32 （20H）	ADC1_ SYNCIN4
33 （21H）	ADC1_TRGOUT1	33 （21H）	ADC1_ SYNCIN5
34 （22H）	RSVD	34 （22H）	ADC1_ SYNCIN6

35 （23H）	RSVD	35 （23H）	ADC1_ SYNCIN7
36 （24H）	RSVD	36 （24H）	ADC1_ SYNCIN8
37 （25H）	RSVD	37 （25H）	ADC1_ SYNCIN9
38 （26H）	CMP0_TRGOUT	38 （26H）	ADC1_ SYNCIN10
39 （27H）	CMP1_TRGOUT	39 （27H）	ADC1_ SYNCIN11
40 （28H）	CMP2_TRGOUT	40 （28H）	ADC1_ SYNCIN12
41 （29H）	RSVD	41 （29H）	ADC1_ SYNCIN13
42 （2AH）	RSVD	42 （2AH）	ADC1_ SYNCIN14
43 （2BH）	RSVD	43 （2BH）	DAC0_ SYNCIN0
44 （2CH）	RSVD	44 （2CH）	DAC0_ SYNCIN1
45 （2DH）	RSVD	45 （2DH）	DAC0_ SYNCIN2
46 （2EH）	RSVD	46 （2EH）	CMP0_SYNCIN
47 （2FH）	GPTA0_TRGOUT0	47 （2FH）	CMP1_SYNCIN
48 （30H）	GPTA0_TRGOUT1	48 （30H）	CMP2_SYNCIN
49 （31H）	GPTA1_TRGOUT0	49 （31H）	GPTA0_SYNCIN0
50 （32H）	GPTA1_TRGOUT1	50 （32H）	GPTA0_SYNCIN1
51 （33H）	GPTA2_TRGOUT0	51 （33H）	GPTA0_SYNCIN2
52 （34H）	GPTA2_TRGOUT1	52 （34H）	GPTA0_SYNCIN3
53 （35H）	GPTA3_TRGOUT0	53 （35H）	GPTA0_SYNCIN4
54 （36H）	GPTA3_TRGOUT1	54 （36H）	GPTA0_SYNCIN5
55 （37H）		55 （37H）	GPTA0_SYNCIN6
56 （38H）		56 （38H）	GPTA1_SYNCIN0
57 （39H）		57 （39H）	GPTA1_SYNCIN1
58 （3AH）		58 （3AH）	GPTA1_SYNCIN2
59 （3BH）	RSVD	59 （3BH）	GPTA1_SYNCIN3
60 （3CH）	RSVD	60 （3CH）	GPTA1_SYNCIN4
61 （3DH）	RSVD	61 （3DH）	GPTA1_SYNCIN5
62 （3EH）	RSVD	62 （3EH）	GPTA1_SYNCIN6
63 （3FH）	GPTB0_TRGOUT0	63 （3FH）	GPTA2_SYNCIN0
64 （40H）	GPTB0_TRGOUT1	64 （40H）	GPTA2_SYNCIN1
65 （41H）	GPTB1_TRGOUT0	65 （41H）	GPTA2_SYNCIN2
66 （42H）	GPTB1_TRGOUT1	66 （42H）	GPTA2_SYNCIN3
67 （43H）	GPTB2_TRGOUT0	67 （43H）	GPTA2_SYNCIN4
68 （44H）	GPTB2_TRGOUT1	68 （44H）	GPTA2_SYNCIN5
69 （45H）	GPTB3_TRGOUT0	69 （45H）	GPTA2_SYNCIN6
70 （46H）	GPTB3_TRGOUT1	70 （46H）	GPTA3_SYNCIN0
71 （47H）	GPTB4_TRGOUT0	71 （47H）	GPTA3_SYNCIN1

72 （48H）	GPTB4_TRGOUT1	72 （48H）	GPTA3_SYNCIN2
73 （49H）	GPTB5_TRGOUT0	73 （49H）	GPTA3_SYNCIN3
74 （4AH）	GPTB5_TRGOUT1	74 （4AH）	GPTA3_SYNCIN4
75 （4BH）	I2C0_TXSRC	75 （4BH）	GPTA3_SYNCIN5
76 （4CH）	I2C0_RXSRC	76 （4CH）	GPTA3_SYNCIN6
77 （4DH）	RSVD	77 （4DH）	GPTB0_SYNCIN0
78 （4EH）	RSVD	78 （4EH）	GPTB0_SYNCIN1
79 （4FH）	RSVD	79 （4FH）	GPTB0_SYNCIN2
80 （50H）	RSVD	80 （50H）	GPTB0_SYNCIN3
81 （51H）	SPI0_TXSRC	81 （51H）	GPTB0_SYNCIN4
82 （52H）	SPI0_RXSRC	82 （52H）	GPTB0_SYNCIN5
83 （53H）	SPI1_TXSRC	83 （53H）	GPTB0_SYNCIN6
84 （54H）	SPI1_RXSRC	84 （54H）	GPTB1_SYNCIN0
85 （55H）	RSVD	85 （55H）	GPTB1_SYNCIN1
86 （56H）	RSVD	86 （56H）	GPTB1_SYNCIN2
87 （57H）	SIO0_TXSRC	87 （57H）	GPTB1_SYNCIN3
88 （58H）	SIO0_RXSRC	88 （58H）	GPTB1_SYNCIN4
89 （59H）	SIO1_TXSRC	89 （59H）	GPTB1_SYNCIN5
90 （5AH）	SIO1_RXSRC	90 （5AH）	GPTB1_SYNCIN6
91 （5BH）	RSVD	91 （5BH）	GPTB2_SYNCIN0
92 （5CH）	RSVD	92 （5CH）	GPTB2_SYNCIN1
93 （5DH）	USART0_TXSRC	93 （5DH）	GPTB2_SYNCIN2
94 （5EH）	USART0_RXSRC	94 （5EH）	GPTB2_SYNCIN3
95 （5FH）	USART1_TXSRC	95 （5FH）	GPTB2_SYNCIN4
96 （60H）	USART1_RXSRC	96 （60H）	GPTB2_SYNCIN5
97 （61H）	UART0_TXSRC	97 （61H）	GPTB2_SYNCIN6
98 （62H）	UART0_RXSRC	98 （62H）	GPTB3_SYNCIN0
99 （63H）	UART1_TXSRC	99 （63H）	GPTB3_SYNCIN1
100 （64H）	UART1_RXSRC	100 （64H）	GPTB3_SYNCIN2
101 （65H）	UART2_TXSRC	101 （65H）	GPTB3_SYNCIN4
102 （66H）	UART2_RXSRC	102 （66H）	GPTB3_SYNCIN5
103 （67H）	RSVD	103 （67H）	GPTB3_SYNCIN6
104 （68H）	RSVD	104 （68H）	GPTB4_SYNCIN0
105 （69H）	RSVD	105 （69H）	GPTB4_SYNCIN1
106 （6AH）	RSVD	106 （6AH）	GPTB4_SYNCIN2
107 （6BH）	CAN_TXSRC	107 （6BH）	GPTB4_SYNCIN4
108 （6CH）	CAN_RXSRC	108 （6CH）	GPTB4_SYNCIN5

109 （6DH）	RSVD	109 （6DH）	GPTB4_SYNCIN6
110 （6EH）	RSVD	110 （6EH）	GPTB5_SYNCIN0
111 （6FH）	DAC0_TRGOUT	111 （6FH）	GPTB5_SYNCIN1
112 （70H）	RSVD	112 （70H）	GPTB5_SYNCIN2
113 （71H）	RSVD	113 （71H）	GPTB5_SYNCIN4
114 （72H）	RSVD	114 （72H）	GPTB5_SYNCIN5
115 （73H）	RSVD	115 （73H）	GPTB5_SYNCIN6
116 （74H）	RSVD	116 （74H）	DMA0_CH0
117 （75H）	RSVD	117 （75H）	DMA0_CH1
118 （76H）	RSVD	118 （76H）	DMA0_CH2
119 （77H）	RSVD	119 （77H）	DMA0_CH3
120 （78H）	RSVD	120 （78H）	DMA0_CH4
121 （79H）	RSVD	121 （79H）	DMA0_CH5
122 （7AH）	RSVD	122 （7AH）	DMA1_CH0
123 （7BH）	RSVD	123 （7BH）	DMA1_CH1
124 （7CH）	RSVD	124 （7CH）	DMA1_CH2
125 （7DH）	RSVD	125 （7DH）	DMA1_CH3
126 （7EH）	RSVD	126 （7EH）	DMA1_CH4
127 （7FH）	RSVD	127 （7FH）	DMA1_CH5

6.3 寄存器说明

6.3.1 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_CHxCONA	0x0030-0x0070	ETCB通道3-19控制寄存器	0x00000000
ETCB_CHxCONB	0x0074-0x00A0	ETCB通道20-31控制寄存器	0x00000000

6.3.2 ETCB_ENABLE(ETCB使能寄存器)

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

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

Name	Bit	Type	Description
ENABLE	[0]	RW	ETCB模块使能控制 0：禁止 1：使能

6.3.3 ETCB_SWTRG(ETCB软件触发寄存器)

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

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

Name	Bit	Type	Description
SWTRG_CHx	[31:0]	RW	软件触发控制 0：无效 1：触发该通道的事件

6.3.4 ETCB_CH0CON0(ETCB通道0控制寄存器0)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD			DST2_SEL								DST2_EN	RSVD	DST1_SEL								DST1_EN	RSVD	DST0_SEL								DST0_EN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
DST2_SEL	[28:21]	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	[18:11]	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	[8:1]	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

6.3.5 ETCB_CH0CON1(ETCB通道0控制寄存器1)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC_SEL							RSVD																							TRIG_MODE	CH0_EN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW

Name	Bit	Type	Description
SRC_SEL	[31:25]	RW	触发源选择 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)
CH0_EN	[0]	RW	通道0使能控制 0: 禁止 1: 使能

6.3.6 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	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
DST2_SEL	[28:21]	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	[18:11]	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	[8:1]	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

6.3.7 ETCB_CH1CON1(ETCB通道1控制寄存器1)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC_SEL							RSVD																							TRIG_MODE	CH1_EN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW

Name	Bit	Type	Description
SRC_SEL	[31:25]	RW	触发源选择 参考事件对应表
TRIG_MODE	[1]	RW	触发模式选择 0: 硬件触发 1: 软件触发 (由ETCB_SWTRG寄存器触发)
CH1_EN	[0]	RW	通1使能控制 0: 禁止 1: 使能

6.3.8 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_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	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
DST2_SEL	[28:21]	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	[18:11]	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	[8:1]	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

6.3.9 ETCB_CH2CON1(ETCB通道2控制寄存器1)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC_SEL							RSVD																							TRIG_MODE	CH2_EN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW

Name	Bit	Type	Description
SRC_SEL	[31:25]	RW	触发源选择 参考事件对应表
TRIG_MODE	[1]	RW	触发模式选择 0: 硬件触发 1: 软件触发 (由ETCB_SWTRG寄存器触发)
CH2_EN	[0]	RW	通道2使能控制 0: 禁止 1: 使能

6.3.10 ETCB_CHxCONA(ETCB通道3-19控制寄存器)

Address = Base Address+ 0x0030-0x0070, 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
DST_SEL								RSVD					SRC_SEL							RSVD										TRIG_MODE	CHx_EN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	RW	RW

Name	Bit	Type	Description
DST_SEL	[31:24]	RW	触发目标选择 参考事件对应表
SRC_SEL	[18:12]	RW	触发源选择 参考事件对应表
TRIG_MODE	[1]	RW	触发模式选择 0: 硬件触发 1: 软件触发 (由ETCB_SWTRG寄存器触发)
CHx_EN	[0]	RW	通道x使能控制 0: 禁止 1: 使能

6.3.11 ETCB_CHxCONB(ETCB通道20-31控制寄存器)

Address = Base Address+ 0x0074-0x00A0, Reset Value = 0x00000000

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

Name	Bit	Type	Description
DST_SEL	[31:24]	RW	触发目标选择 参考事件对应表
SRC_SEL	[18:12]	RW	触发源选择 参考事件对应表
DMA_EN	[2]	RW	DMA功能 0: 禁止 1: 使能
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)
CHx_EN	[0]	RW	通道x使能控制 0: 禁止 1: 使能

7 直接存储器访问控制器 (DMA)

7.1 概述

本章节描述DMA控制器的功能，从用户的角度详细说明如何操作DMA。

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。

7.1.1 主要特性

- 最多支持6个独立配置的通道
- 源和目标地址都可以自增或保持不变
- 原子传输数据大小可以是8-bit (byte), 16-bit (half-word) or 32-bit (word)或者是4×8-bit (byte)，4×16-bit (half-word) or 4×32-bit (word)
- 源和目标可以
    - 都是系统总线 (例如：存储器到存储器的传输。注意，Flash存储器只能为DMA1的源)
    - 源是系统总线，目标为外设总线 (例如：储存器到I/O 模块的传输)
    - 源时外设总线，目标是系统总线 (例如：I/O 模块到存储器的传输。注意，Flash存储器只能为DMA1的源)
    - 源和目标都是外设总线 (例如：I/O 模块到I/O 模块的传输)
- 可以以软件或硬件的方式触发传输

  注：只有DMA1可以访问Flash存储器，DMA0不能访问Flash存储器。

7.2 功能描述

每个DMA通道可以通过DMA_MTRx[CHEN] 独立使能。DMA_SRR[SWRST]可以实现对整个DMA模块的复位。

7.2.1 模块框图

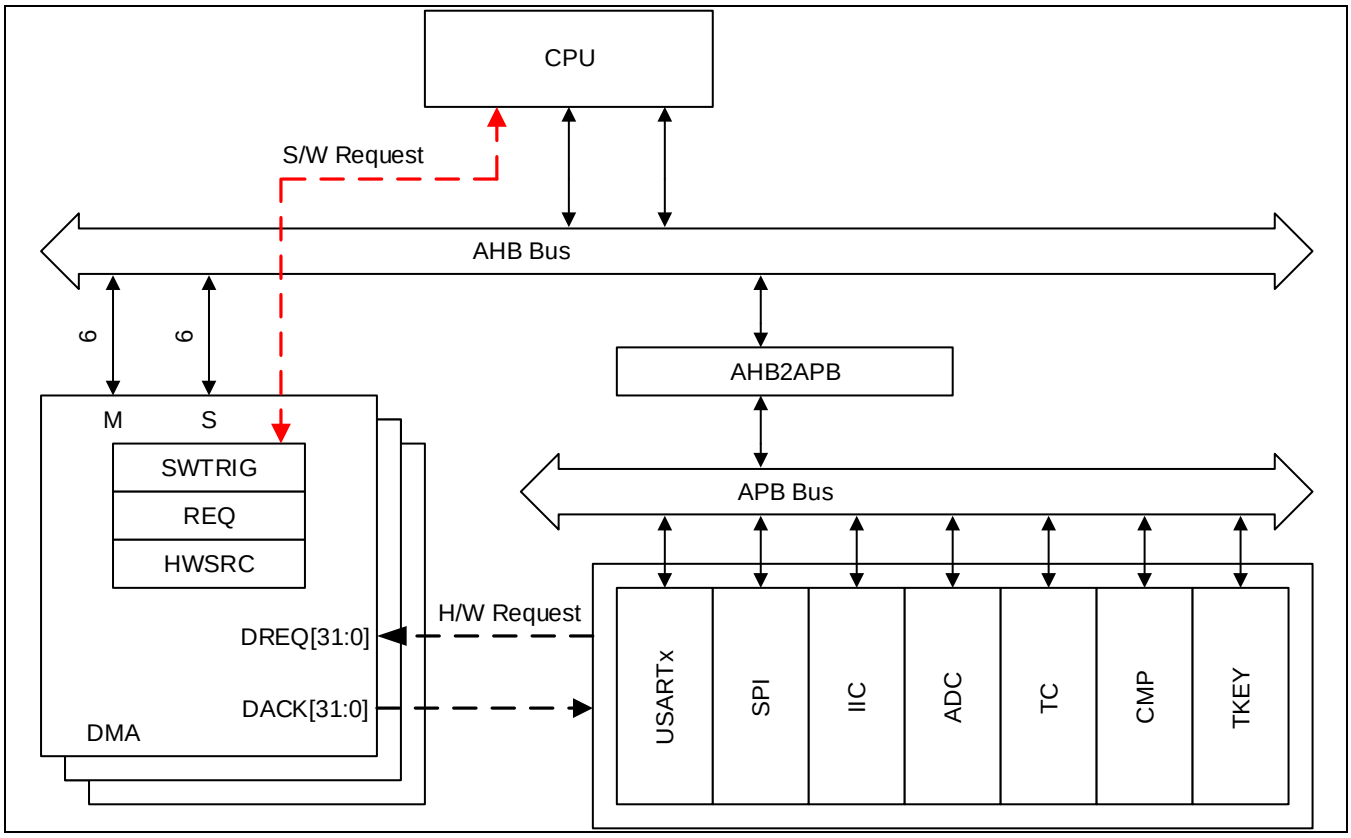


Figure 7-1 DMA 请求

7.2.2 功能描述

DMA（直接存储器访问）在源和目标间直接传输数据，源或目标可以是SRAM，U(S)ART, SPI, CMP, TKEY or IIC, Flash存储器和ADC等，其中Flash存储器和ADC只能为源。

通过写DMA控制寄存器以设置通道的控制信息，比如源地址，目标地址，传输数据大小等。

7.2.2.1 传输数据大小

7.2.2.1.1 原子传输

每个通道原子传输的大小可以通过DMA_CRx[DSIZE]独立配置成字节（byte）、半字（half-word）或字（word）。同时DMA内部有4字的FIFO类型缓存，可以将原子传输扩展为4个单元，即4个字节、4个半字或4个字。传输FIFO可以通过DMA_CRx[TSIZE]使能。此时DMA的一次原子传输将连续进行4个读周期和4个写周期，如Figure 7-4所示。

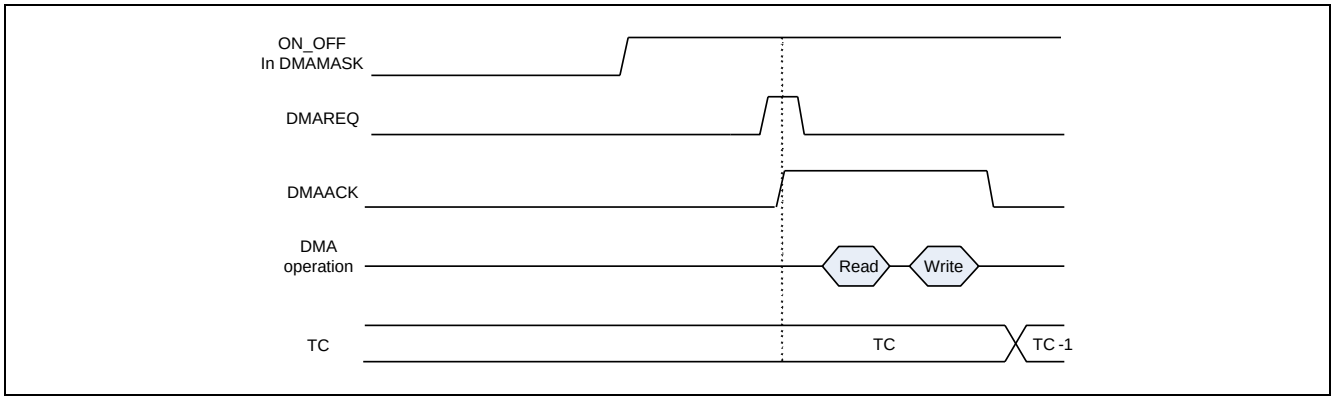


Figure 7-2 FIFO未使能时的原子传输，DMA_CRx[TSIZE] = 0

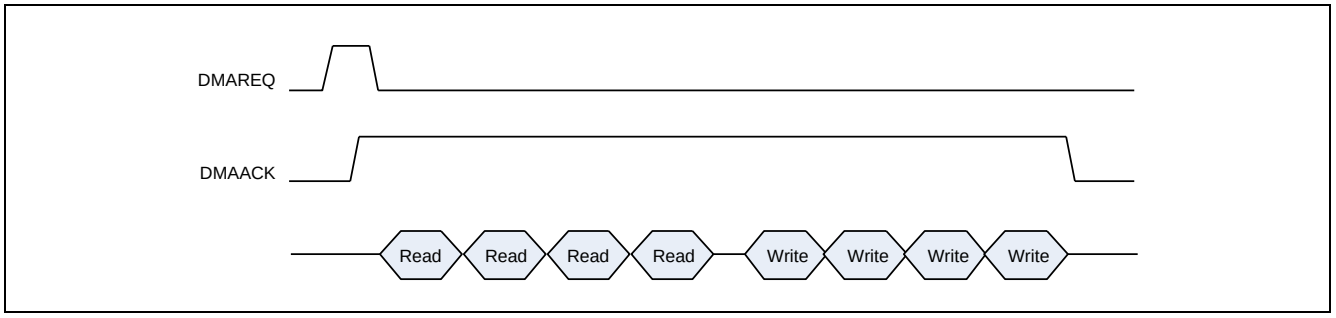


Figure 7-3 FIFO使能时的原子传输，DMA_CRx[TSIZE] = 1

7.2.2.1.2 传输计数

DMA内部有两个向下计数器，其初始值由DMA_CRx[HTC] 和 DMA_CRx[LTC]决定，HTC为高传输计数，LTC为低传输计数。低计数器在每次原子传输结束后减1，高计数器在每次低计数器溢出后减1。

配置DAM传输时，HTC 和 LTC必须为1到4095之间的值。如果HTC或LTC为0，则通道x掩码触发寄存器DMA_MTRx[CHEN]位将不能被置“1”，即无法开启传输。

7.2.2.1.3 地址变化方式

源和目标地址可以通过DMA_ISCRx和DMA_IDCRx设置为自增或固定。假设有如下设置：初始地址是“0x0”，数据大小是字（Word），HTC为“3”和LTC为“4”（地址可以为源或目标地址，在下图中，地址的表达前未加“0x”）。

1）DMA_CRx[TSIZE]配置为“0”时，DMA的一次原子传输，传输大小为1个字，地址控制的效果如下表所示：

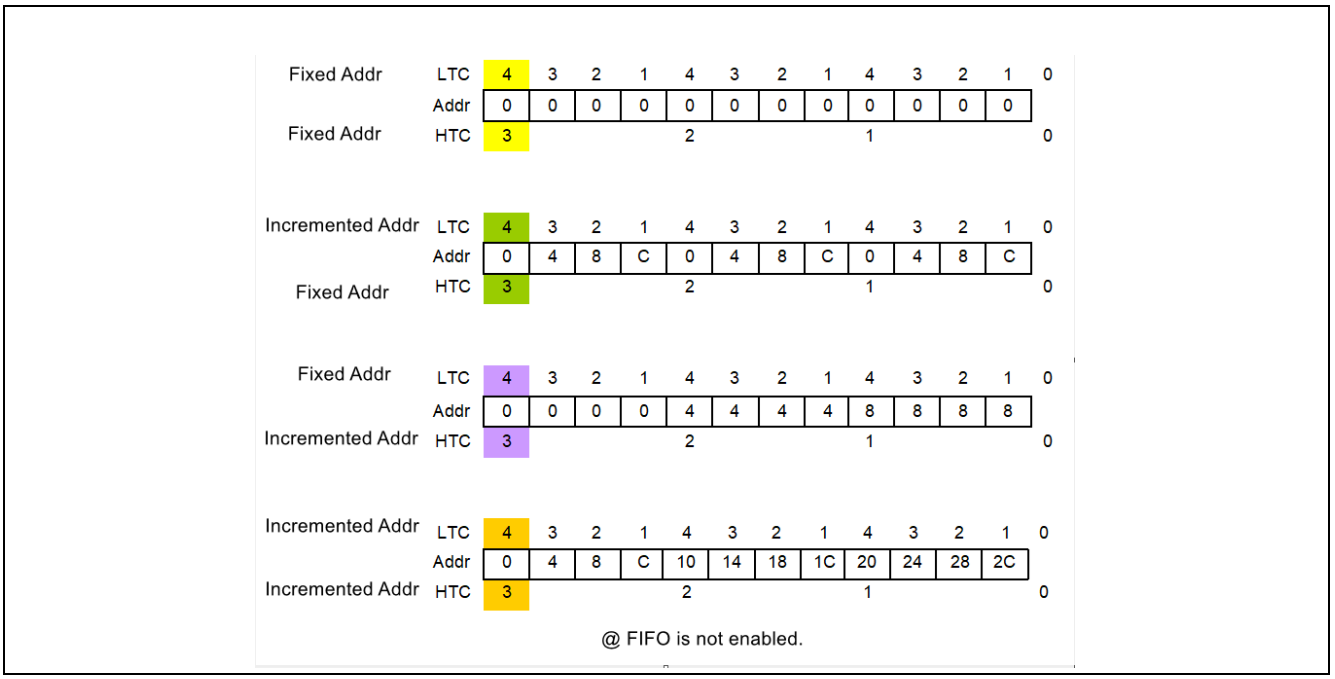


Figure 7-4 传输计数和地址的关系 @ FIFO未使能

DMA_CRx[TSIZE]配置为“0”时，下面以源地址变化为例，说明其与高位、低位传输计数的源地址自增控制位之间的关系（目标地址变化与自增控制位的关系与源地址的一样）：

- 当DMA_ISCRx= 0x3（即低位和高位地址自增控制位，都配置为Fixed Addr（不自增））时，不论是低位计数值LTC减少，还是高位计数值HTC减少时，源地址都保持不变。
- 当DMA_ISCRx= 0x2（即低位地址自增控制位，配置为Incremented Addr（自增），而高位地址自增控制位，配置为Fixed Addr（不自增））时：
  - 低位计数值LTC减少，高位计数值HTC不变，源地址自增；
  - 低位计数值LTC不变，高位计数值HTC减少，源地址保持不变。
- 当DMA_ISCRx= 0x1（即低位地址自增控制位，配置为Fixed Addr（不自增），而高位地址自增控制位，配置为Incremented Addr（自增））时：
  - 低位计数值LTC减少，高位计数值HTC不变，源地址保持不变；
  - 低位计数值LTC不变，高位计数值HTC减少，源地址自增。
- 当DMA_ISCRx= 0x0（即低位和高位地址自增控制位，都配置为Incremented Addr（自增））时：
  - 低位计数值LTC减少，高位计数值HTC不变，源地址自增；

■ 低位计数值LTC不变，高位计数值HTC减少，源地址自增。

2）DMA_CRx[TSIZE]配置为“1”时，DMA的一次原子传输，传输大小为4个字，地址控制的效果如下表所示：

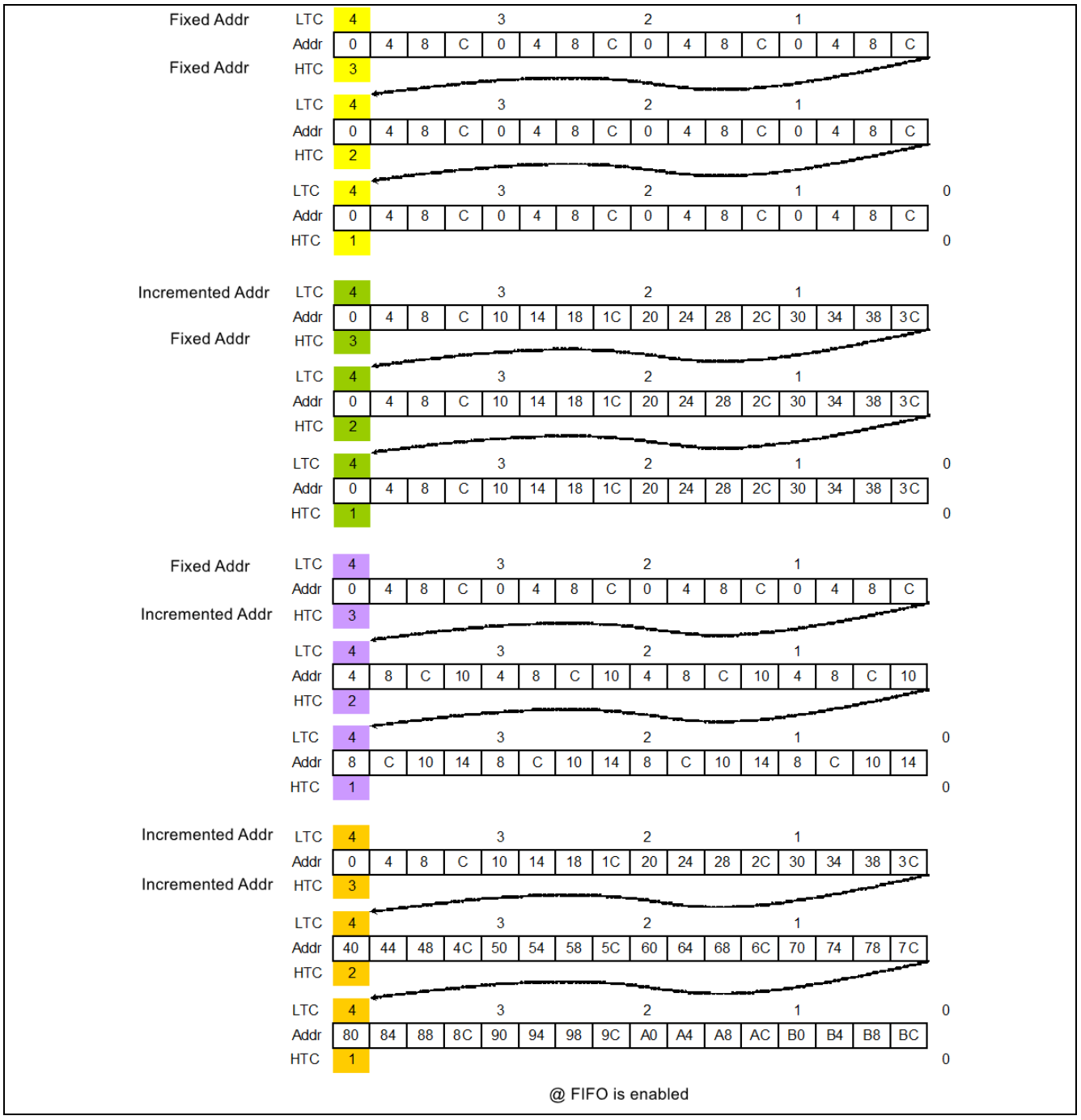


Figure 7-5 传输计数和地址的关系 @ FIFO使能

DMA_CRx[TSIZE]配置为“1”时，一次原子传输的数据大小为4个字，占16个地址空间。其源地址和目的地址自增的规律与DMA_CRx[TSIZE]配置为“0”时类似。

当前传输源地址和目标地址可以分别通过DMA_CSRx[CURR_SADDR]和DMA_CDRx[CURR_DADDR]查询。

根据每次请求实现的原子传输次数，可以将传输分为单次服务模式和连续服务模式。

7.2.2.1.4 服务模式

DMA传输可以工作在单次服务模式（DMA_CRx[SMODE] = 0）或连续服务模式（DMA_CRx[SMODE] = 1）。

- **单次服务模式**

单次服务模式下，DMA每次读/写的原子传输都需要请求和响应握手机制。此时每次DMA收到请求后都只进行一次原子传输，当DMA请求信号变高后，DMA控制器将输出响应信号拉高，随后开始读周期和写周期操作，写周期操作后DMA控制器将输出响应信号拉低，完成一次原子操作。DMA_SR[CURR _LTC]减1。

当DMA_SRx[CURR_LTC] 向下计数到0时将置起LTCINT请求；当DMA_SRx[CURR_LTC] 和 DMA_SRx[CURR_HTC] 都向下计数到0时，将置起TCINT请求，DMA传输结束。

注意：在CRx[LTC]个数据传输完成前，DMA将忽略其他通道的所有请求。所以如果不同通道间的DMA事件发生间隔比较短的话，要尽量减小CRx[LTC]的值。

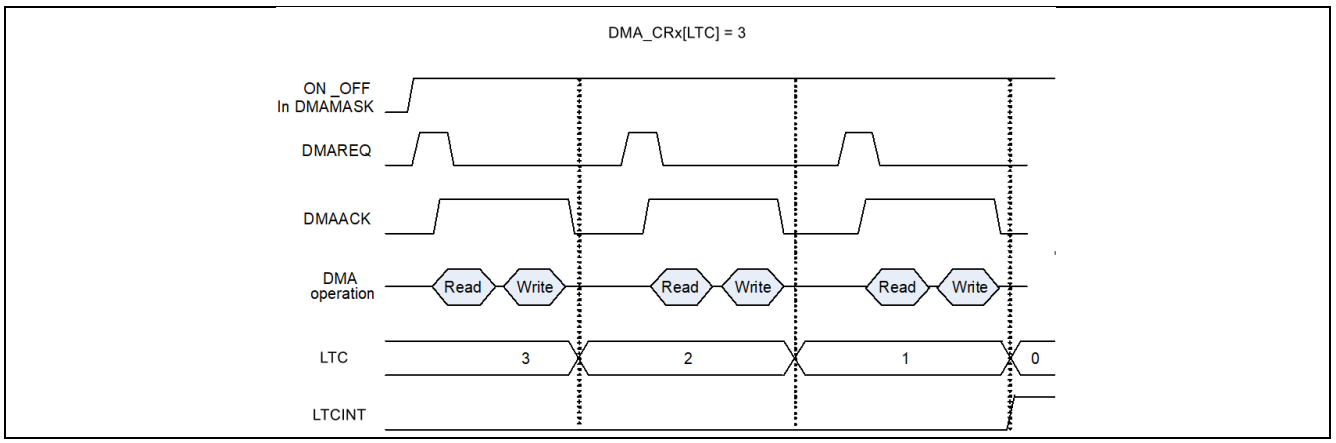


Figure 7-6 单次服务模式的传输序列 @ DMA_CRx[TSIZE] = 0

- **连续服务模式**

在连续模式下， DMA在收到一次操作请求后，将连续传输LTC[11:0]次原子传输，每次传输 DMA_SRx[CURR_LTC] 都减为0。LTCIT请求置起；当DMA_SRx[CURR_LTC] 和 DMA_SRx[CURR_HTC] 都向下计数到0时，将置起TCIT请求，DMA传输结束。

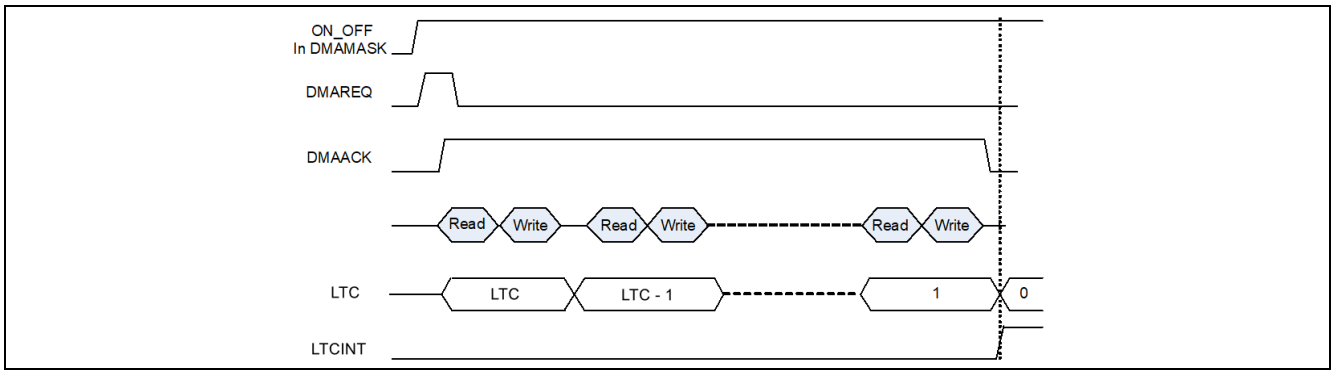


Figure 7-7 连续服务模式 @ DMA_CRx[TSIZE] = 0

7.2.2.2 传输开始

DMA 控制器可以通过两种方式启动。一种是软件方式，由 CPU 通过系统总线写 DMA 内部寄存器来实现；另一种是外设 I/O 模块（如 UART0/1/2，USART0，SPI，IIC，CMP0/1, TKEY, ADC 等）通过 ETCB 模块向 DMA 发送传输请求，DMA 响应该请求后将发出对应的应答信号。程序可以通过软件查询或中断处理的方式来停止或重新启动 DMA 传输。

7.2.2.2.1 软件请求

当DMA_RSRx[REQ] = 0时，DMA工作在只响应软件请求的模式下。此时只有置位DMA_MTRx[SWTRIG]，才能发起DMA传输。具体设置过程如下：

1. DMA操作条件设置

   - 设置源初始地址和地址模式 (DMA_ISRx, DMA_ISCRx)
   - 设置目标地址和地址模式 (DMA_IDRx, DMA_IDCRx)
   - 设置DMA 中断，服务模式，原子传输大小（包括FIFO状态），HTC和LTC)

2. DMA请求设置，DMA使能和开始

   - 设置DMA通道x请求选择寄存器（DMA_RSRx[REQ] = 0 (软件))
   - 设置DMA使能和开始 (DMA_MTRx[CHEN] = 1, DMA_MTRx[SWTRIG] = 1). 当SWTRIG为被置‘1’，DMA将启动基于步骤1中配置的操作，且SWTRIG将被自动清零。

   A. 操作模式(单次服务模式)

   A1. 当DMA完成一次原子传输（一个单元或连续四个单元读写周期）并将CURR_LTC减1后，DMA进入停止状态并等SWTRIG被置“1”。

   A2. 重复A1 直到CURR_LTC向下计数到0，当CURR_LTC为0，DMA将CURR_HTC减1。

   A3. 重复A2 直到CURR_HTC 和CURR_LTC都变成0。

   B. 操作模式(连续服务模式)

B1. 当DMA完成所有低计数DMA传输（LTC × 原子传输）并将CURR_HTC减1后，MA进入停止状态并等SWTRIG被置“1”。

B2. 重复B1直到CURR_HTC 和CURR_LTC都变成0。

7.2.2.2.2 硬件请求

当DMA_RSRx[REQ] = 1时，DMA工作在只响应硬件请求的模式下。此时除了发起请求的模块外，还需要配置ETCB。具体设置过程如下：

1. DMA操作条件设置
   - 设置源地址和控制寄存器 (DMA_ISRx, DMA_ISCRx)
   - 设置目标地址和控制寄存器 (DMA_IDRx, DMA_IDCRx)
   - 设置DMA 控制寄存器(中断，操作模式，传输大小，连续模式、传输个数)
2. DMA 请求设置，DMA 使能和开始
   - 设置DMA通道x请求选择寄存器（DMA_RSRx）(REQ = 1 (硬件))，并设置好ETCB模块中外设I/O模块到DMA的通道。
   - 设置DMA使能(DMA_MTRx: CHEN = 1)
3. DMA 开始
   - 当DMA接收到外设I/O模块的硬件请求后，DMA将启动基于步骤1中配置的操作

A. 操作模式(单次服务模式)

a. 当DMA完成一次原子传输（一个单元或连续四个单元读写周期）并将CURR_LTC[11:0]减1后，DMA进入停止状态并等外设I/O模块的硬件请求。

b. 重复原子传输直到CURR_LTC[11:0]向下计数到0，当CURR_LTC[11:0]为0，DMA将CURR_HTC[11:0]减1。

c. 重复a 和b操作直到CURR_HTC[11:0] 和CURR_LTC[11:0] 都变成0。

B. 操作模式(连续服务模式)

a. 当DMA完成所有低计数DMA传输（CURR_LTC[11:0] × 原子传输）并将CURR_HTC[11:0]减1后，MA进入停止状态并等外设I/O模块的硬件请求

b. 重复a操作直到CURR_HTC[11:0] 变成0。

7.2.2.3 传输结束

每次对 DMA 任何一个通道的配置，将会在该通道上完成 DMA_CRx[HTC] x DMA_CRx[LTC]次原子传输。结束后，即使重用上一次的传输配置，也需要再次使能通道。如果重用上一次的配置，且在完成一次传输后自动使能DMA，可以设置 DMA_CRx[RELOAD] = 1。

在一轮传输的过程中，可以通过DMA_MTRx[STOP] 停止传输。如果在停止时，没有原子传输在进行，则DMA立即停止工作，如果停止时，有原子传输在进行，则DMA会等待当前原子传输完成后停止工作。

7.2.2.4 通道优先级

DMA通道优先级和通道数字一致，即通道0的优先级最高，通道3的优先级最低。例如，程序想设置USART_TX的DMA请求为最高优先级，那么在ETCB中设置USART_TX到DMA的通道时，事件源选择USART_TX，目标事件选DMA_CH0。

7.2.3 中断控制

DMA模块一共支持2种类型的中断。

DMA_ISR[CHx_LTCIT]标志位置起表示对应通道的CURR_LTC已经计数到0，此时如果DMA_CRx[LTCIT]已使能，将会向CPU发出中断请求。DMA_ISR[CHx_TCIT]标志位置起表示对应通道的CURR_HTC已经计数到0，此时如果DMA_CRx[TCIT]已使能，将会向CPU发出中断请求。

中断标志位可以通过写DMA_ICR清除。

7.3 寄存器说明

7.3.1 寄存器表

Base Address of DMA0: 0x40300000

Base Address of DMA1: 0x40310000

Register	Offset	Description	Reset Value
DMA_ISRx	0x0000+0x080*x	通道x初始寄存器	0x00000000
DMA_ISCRx	0x0004+0x080*x	通道x初始源控制寄存器	0x00000000
DMA_IDRx	0x0008+0x080*x	通道x初始目标寄存器	0x00000000
DMA_IDCRx	0x000C+0x080*x	通道x初始目标控制寄存器	0x00000000
DMA_CRx	0x0010+0x080*x	通道x控制寄存器	0x00000000
DMA_SRx	0x0014+0x080*x	通道x状态寄存器	0x00000000
DMA_CSRx	0x0018+0x080*x	通道x当前源寄存器	0x00000000
DMA_CDRx	0x001c+0x080*x	通道x当前目标寄存器	0x00000000
DMA_MTRx	0x0020+0x080*x	通道x掩码触发寄存器	0x00000000
DMA_RSRx	0x0024+0x080*x	通道x请求选择寄存器	0x00000000
DMA_IDR	0x0500	ID寄存器	0x00000370
DMA_SRR	0x0504	软件复位寄存器	0x00000000
DMA_CESR	0x0508	通道状态寄存器	0x00000000
DMA_ISR	0x050c	中断状态寄存器	0x00000000
DMA_ICR	0x0510	中断清除寄存器	0x00000000

注意:

1. ‘x’代表DMA的通道，取指范围从0到5。
2. 通道寄存器起始地址为：Ch0: 0x000 Ch1: 0x080 Ch2: 0x100 Ch3: 0x180 ch4:0x200 ch5:0x280
3. 通道优先级和通道数字一致，即通道0的优先级最高，通道5的优先级最低。当同一时间有两个或两个以上DMA触发时DMA控制器将先传输通道数字最低的通道。

7.3.2 DMA_ISRx(通道x初始寄存器)

Address = Base Address+ 0x0000, Reset Value = 0x00000000
Address = Base Address+ 0x0080, Reset Value = 0x00000000
Address = Base Address+ 0x0100, Reset Value = 0x00000000
Address = Base Address+ 0x0180, Reset Value = 0x00000000
Address = Base Address+ 0x0200, Reset Value = 0x00000000
Address = Base Address+ 0x0280, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SADDR																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
SADDR	[31:0]	RW	DMA初始源地址. 该位为传输源的基地址（开始地址）

7.3.3 DMA_ISCRx(通道x初始源控制寄存器)

Address = Base Address+ 0x0004, Reset Value = 0x00000000
Address = Base Address+ 0x0084, Reset Value = 0x00000000
Address = Base Address+ 0x0104, Reset Value = 0x00000000
Address = Base Address+ 0x0184, Reset Value = 0x00000000
Address = Base Address+ 0x0204, Reset Value = 0x00000000
Address = Base Address+ 0x0284, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														RSVD																HINC	LINC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	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
HINC	[1]	RW	高传输计数的源地址自增控制位 0b：自增 1b：不自增 如果该位为0，每次传输完成后根据连续（burst）或单次（single）模式的高传输计数，源地址自增数据长度。 如果该位为1，每次传输完成源地址保持不变。
LINC	[0]	RW	低传输计数的源地址自增控制位 0b：自增 1b：不自增 如果该位为0，每次传输完成后根据连续（burst）或单次（single）模式的低传输计数，源地址自增数据长度。 如果该位为1，每次传输完成源地址保持不变。

7.3.4 DMA_IDRx(通道x初始目标寄存器)

Address = Base Address+ 0x0008, Reset Value = 0x00000000
Address = Base Address+ 0x0088, Reset Value = 0x00000000
Address = Base Address+ 0x0108, Reset Value = 0x00000000
Address = Base Address+ 0x0188, Reset Value = 0x00000000
Address = Base Address+ 0x0208, Reset Value = 0x00000000
Address = Base Address+ 0x0288, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DADDR																															
0	0	0	0	0	0	0	0	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
DADDR	[31:0]	RW	目标地址. 该位为传输目标的基地址（开始地址）

7.3.5 DMA_IDCRx(通道x初始目标控制寄存器)

Address = Base Address+ 0x000C, Reset Value = 0x00000000
Address = Base Address+ 0x008C, Reset Value = 0x00000000
Address = Base Address+ 0x010C, Reset Value = 0x00000000
Address = Base Address+ 0x018C, Reset Value = 0x00000000
Address = Base Address+ 0x020C, Reset Value = 0x00000000
Address = Base Address+ 0x028C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																														HINC	LINC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	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
HINC	[1]	RW	高传输计数的目标地址自增控制位 0b：自增 1b：不自增 如果该位为0，每次传输完成后根据连续（burst）或单次（single）模式的高传输计数，目标地址自增数据长度。 如果该位为1，每次传输完成目标地址保持不变。
LINC	[0]	RW	低传输计数的目标地址自增控制位 0b：自增 1b：不自增 如果该位为0，每次传输完成后根据连续（burst）或单次（single）模式的低传输计数，目标地址自增数据长度。 如果该位为1，每次传输完成目标地址保持不变。

7.3.6 DMA_CRx(通道x控制寄存器)

Address = Base Address+ 0x0010, Reset Value = 0x00000000
Address = Base Address+ 0x0090, Reset Value = 0x00000000
Address = Base Address+ 0x0110, Reset Value = 0x00000000
Address = Base Address+ 0x0190, Reset Value = 0x00000000
Address = Base Address+ 0x0210, Reset Value = 0x00000000
Address = Base Address+ 0x0290, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	TCIT	LTCIT	TSIZE	SMODE	RELOAD	DSIZE		HTC												LTC											
0	0	0	0	0	0	0	0	0	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	RW	RW	RW	RW	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
TCIT	[30]	RW	传输计数中断使能/禁止控制位 0b: 禁止传输计数向下计数到0时的中断，此时用户需查询状态寄存器的传输计数位来判断传输情况。 1b: 使能传输计数向下计数到0（CURR_LTC[11:0] 和 CURR_HTC[11:0]都为0）时的中断。
LTCIT	[29]	RW	低传输计数中断使能/禁止控制位 0b: 禁止低传输计数向下计数到0时的中断，此时用户需查询状态寄存器的传输计数位来判断传输情况。 1b: 使能低传输计数向下计数到0（CURR_LTC[11:0]为0）时的中断。
TSIZE	[28]	RW	传输大小 选择原子传输（每次DMA获取总线后，并在其释放总线前的传输） 0b: 单次传输，传输的大小为DSIZE定义的大小。 1b: 连续传输，传输的大小为DSIZE定义的大小乘以4
SMODE	[27]	RW	服务模式选择位 选择DMA操作模式为单次服务或全服务模式 0b: 单次服务模式，但每个原子传输（单次或连续）完成后停止DMA服务并等待新的DMA请求。 1b：连续服务模式，一次DMA请求后将重复原子传输直到低位传输计数LTC[11:0]为0 注：即使是全服务模式，DMA在每次原子传输后将释放总线，并再次发起总线请求以防止阻塞其他总线主机。
RELOAD	[26]	RW	自动重载使能/禁止控制位 0b: 当前传输计数为0（所有传输请求已完成）时启动自动重载。自动重载可用于重复的DMA操作（使用相同地址和传输数目）。 1b: 当传输计数为0时关闭DMA通道。建议设置通道开/关位（CHEN）为0以防止意外启动新的DMA传输
DSIZE	[25:24]	RW	传输数据大小选择位 00b = 字节（Byte）

			01b = 半字（Half word） 10b = 字 （Word） 11b = 无效
HTC	[23:12]	RW	高传输计数 初始化高传输个数 注：传输操作的个数由HTC[11:0] □ LTC[11:0]所决定，当CURR_HTC[11:0] 和 CURR_LTC[11:0]都为0时传输结束。
LTC	[11:0]	RW	低传输计数 初始化低传输个数。
注意： 1、HTC[11:0] 和 LTC[11:0]都不为0时，才能设置出一个有效的DMA操作。 2、在每次CRx[LTC]个数据传输完成前，DMA将忽略所有请求。所以如果DMA事件发生间隔比较短的话，要尽量减小CRx[LTC]的值。			

7.3.7 DMA_SRx(通道x状态寄存器)

Address = Base Address+ 0x0014, Reset Value = 0x00000000
Address = Base Address+ 0x0094, Reset Value = 0x00000000
Address = Base Address+ 0x0114, Reset Value = 0x00000000
Address = Base Address+ 0x0194, Reset Value = 0x00000000
Address = Base Address+ 0x0214, Reset Value = 0x00000000
Address = Base Address+ 0x0294, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTCST	RSVD							CURR_ HTC												CURR_LTC											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
LTCST	[31]	R	低传输计数状态位 0b: DAM控制器处于空闲状态。 1b: DAM控制器处于忙状态 当触发事件到来并启动DMA操作，LTCST位被置“1”（BUSY）
CURR_ HTC	[23:12]	R	当前高传输计数值 设置DMA_CR想寄存器的HTC[11:0]位时，初始化传输计数值为HTC[11:0]，当一次原子传输完成后若LTC[11:0]为0，则该传输计数值减一。 传输操作的计数值由HTC[11:0] x LTC[11:0]决定，当CUR_HTC[11:0] 和CUR_LTC[11:0] 计数值都为0时将停止传输并将DMA_MTR的STOP为置“1”
CURR_LTC	[11:0]	R	当前低传输计数值 设置DMA_CR寄存器的LTC[11:0]位时，初始化传输计数值为LTC[11:0]，每完成一次原子传输，传输计数值减一。 当CURR_HTC[11:0]不为“0”且CURR_LTC[11:0]为“1”时，在一次传输完成后，CURR_HTC[11:0]将减一，CURR_LTC[11:0]将初始化成DMA_CR寄存器的LTC[11:0]的值。

7.3.8 DMA_CSRx(通道x当前源寄存器)

Address = Base Address+ 0x0018, Reset Value = 0x00000000
Address = Base Address+ 0x0098, Reset Value = 0x00000000
Address = Base Address+ 0x0118, Reset Value = 0x00000000
Address = Base Address+ 0x0198, Reset Value = 0x00000000
Address = Base Address+ 0x0218, Reset Value = 0x00000000
Address = Base Address+ 0x0298, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURR_SADDR																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
CURR_SADDR	[31:0]	R	当前源地址 当前传输的源地址。

7.3.9 DMA_CDRx(通道x当前目标寄存器)

Address = Base Address+ 0x001c, Reset Value = 0x00000000
Address = Base Address+ 0x009c, Reset Value = 0x00000000
Address = Base Address+ 0x011c, Reset Value = 0x00000000
Address = Base Address+ 0x019c, Reset Value = 0x00000000
Address = Base Address+ 0x021c, Reset Value = 0x00000000
Address = Base Address+ 0x029c, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURR_DADDR																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
CURR_DADDR	[31:0]	R	当前目标地址 当前传输的目标地址

7.3.10 DMA_MTRx(通道x掩码触发寄存器)

Address = Base Address+ 0x0020, Reset Value = 0x00000000
Address = Base Address+ 0x00A0, Reset Value = 0x00000000
Address = Base Address+ 0x0120, Reset Value = 0x00000000
Address = Base Address+ 0x01A0, Reset Value = 0x00000000
Address = Base Address+ 0x0220, Reset Value = 0x00000000
Address = Base Address+ 0x02A0, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																													STOP	CHEN	SWTRIG
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	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
STOP	[2]	RW	DMA 操作停止位 0b：无效 1b: 如果有当前原子传输，则DMA在当前原子传输完成后立即停止。若当前没有原子传输，则DMA立即停止。 注：由于当前可能存在原子传输，所以“停止”操作可能需要一些时钟周期，当查到CHEN位为“0”时，该“停止”操作才是真正完成。
CHEN	[1]	RW	DMA 通道开/关位 0b: 关闭DMA通道（对该通道的DMA请求将被忽略） 1b: 开启DMA通道，对该通道的DMA请求将被处理。 对该位置“1”开始DMA操作，当DMA_MTRx寄存器的STOP位被置“1”该位会被自动清零（“0”）；当DMA_MTRx寄存器的STOP位为“0”且DMA_CRx[26]位为“0”（禁止自动重载），当CURR_LTC[11:0]和CURR_HTC[11:0]都为“0”时，该位会被自动清零（“0”）。 注：在DMA操作时，不建议手动修改该寄存器位（i.e., 建议只通过DMA_CRx[26]或STOP位进行操作）。
SWTRIG	[0]	RW	DMA 软件触发 以软件请求的方式触发DMA通道，对该位置“1”以开始DMA操作，当DMA操作开始后，该位自动清零。 0b：无效 1b: 请求DMA操作

7.3.11 DMA_RSRx(通道x请求选择寄存器)

Address = Base Address+ 0x0024, Reset Value = 0x00000000
Address = Base Address+ 0x00A4, Reset Value = 0x00000000
Address = Base Address+ 0x0124, Reset Value = 0x00000000
Address = Base Address+ 0x01A4, Reset Value = 0x00000000
Address = Base Address+ 0x0224, Reset Value = 0x00000000
Address = Base Address+ 0x02A4, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															RSVD																REQ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	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
REQ	[0]	RW	请求类型选择位 0: 软件请求模式，DMA将被DMA_MTRx寄存器的SWTRIG位触发。 1：硬件请求模式，DMA将被HWSRC[3:0]对应的请求所触发

7.3.12 DMA_IDR(ID寄存器)

Address = Base Address+ 0x0500, Reset Value = 0x00000370

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

Name	Bit	Type	Description
IDCODE	[31:0]	R	ID代码

7.3.13 DMA_SRR(软件复位寄存器)

Address = Base Address+ 0x0504, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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 = 全局软件复位

7.3.14 DMA_CESR(通道状态寄存器)

Address = Base Address+ 0x0508, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																										CH5EN	CH4EN	CH3EN	CH2EN	CH1EN	CH0EN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
CH5EN	[5]	R	通道5使能状态位 0 = 通道5未使能. 1 = 通道5已使能.
CH4EN	[4]	R	通道4使能状态位 0 = 通道4未使能. 1 = 通道4已使能.
CH3EN	[3]	R	通道3使能状态位 0 = 通道3未使能. 1 = 通道3已使能.
CH2EN	[2]	R	通道2使能状态位 0 = 通道2未使能. 1 = 通道2已使能.
CH1EN	[1]	R	通道1使能状态位 0 = 通道1未使能. 1 = 通道1已使能.
CH0EN	[0]	R	通道0使能状态位 0 = 通道0未使能. 1 = 通道0已使能.

7.3.15 DMA_ISR(中断状态寄存器)

Address = Base Address+ 0x050c, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD										CH5_TCIT	CH4_TCIT	CH3_TCIT	CH2_TCIT	CH1_TCIT	CH0_TCIT	RSVD										CH5_LTCIT	CH4_LTCIT	CH3_LTCIT	CH2_LTCIT	CH1_LTCIT	CH0_LTCIT
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
CH5_TCIT	[21]	R	通道5传输计数中断状态位 0: 传输计数完成中断未发生。 1：当CRx寄存器的TCIT位为“1”时，当所有传输完成（CURR_LTC[11:0]和CURR_HTC[11:0]都等于0）时，该位将被置“1”。
CH4_TCIT	[20]	R	通道4传输计数中断状态位 0: 传输计数完成中断未发生。 1：当CRx寄存器的TCIT位为“1”时，当所有传输完成（CURR_LTC[11:0]和CURR_HTC[11:0]都等于0）时，该位将被置“1”。
CH3_TCIT	[19]	R	通道3传输计数中断状态位 0: 传输计数完成中断未发生。 1：当CRx寄存器的TCIT位为“1”时，当所有传输完成（CURR_LTC[11:0]和CURR_HTC[11:0]都等于0）时，该位将被置“1”。
CH2_TCIT	[18]	R	通道2传输计数中断状态位 0: 传输计数完成中断未发生。 1：当CRx寄存器的TCIT位为“1”时，当所有传输完成（CURR_LTC[11:0]和CURR_HTC[11:0]都等于0）时，该位将被置“1”。
CH1_TCIT	[17]	R	通道1传输计数中断状态位 0: 传输计数完成中断未发生。 1：当CRx寄存器的TCIT位为“1”时，当所有传输完成（CURR_LTC[11:0]和CURR_HTC[11:0]都等于0）时，该位将被置“1”。
CH0_TCIT	[16]	R	通道0传输计数中断状态位 0: 传输计数完成中断未发生。 1：当CRx寄存器的TCIT位为“1”时，当所有传输完成（CURR_LTC[11:0]和CURR_HTC[11:0]都等于0）时，该位将被置“1”。
CH5_LTCIT	[5]	R	通道5低传输计数完成中断状态位 0: 低传输计数完成中断未发生。

			1：当CRx寄存器的LTCIT位为“1”时，当所有低传输完成（CURR_LTC[11:0]等于0）时，该位将被置“1”。
CH4_LTCIT	[4]	R	通道4低传输计数完成中断状态位 0: 低传输计数完成中断未发生。 1：当CRx寄存器的LTCIT位为“1”时，当所有低传输完成（CURR_LTC[11:0]等于0）时，该位将被置“1”。
CH3_LTCIT	[3]	R	通道3低传输计数完成中断状态位 0: 低传输计数完成中断未发生。 1：当CRx寄存器的LTCIT位为“1”时，当所有低传输完成（CURR_LTC[11:0]等于0）时，该位将被置“1”。
CH2_LTCIT	[2]	R	通道2低传输计数完成中断状态位 0: 低传输计数完成中断未发生。 1：当CRx寄存器的LTCIT位为“1”时，当所有低传输完成（CURR_LTC[11:0]等于0）时，该位将被置“1”。
CH1_LTCIT	[1]	R	通道1低传输计数完成中断状态位 0: 低传输计数完成中断未发生。 1：当CRx寄存器的LTCIT位为“1”时，当所有低传输完成（CURR_LTC[11:0]等于0）时，该位将被置“1”。
CH0_LTCIT	[0]	R	通道0低传输计数完成中断状态位 0: 低传输计数完成中断未发生。 1：当CRx寄存器的LTCIT位为“1”时，当所有低传输完成（CURR_LTC[11:0]等于0）时，该位将被置“1”。

7.3.16 DMA_ICR(中断清除寄存器)

Address = Base Address+ 0x0510, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																										CH5_IT	CH4_IT	CH3_IT	CH2_IT	CH1_IT	CH0_IT
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	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
CH5_IT	[5]	W	通道5中断清除位 0 = 无效No effect 1 = 清除通道 x 中断 (CHx_TCIT 和CHx_LTCIT)
CH4_IT	[4]	W	通道4中断清除位 0 = 无效No effect 1 = 清除通道 x 中断 (CHx_TCIT 和CHx_LTCIT)
CH3_IT	[3]	W	通道3中断清除位 0 = 无效No effect 1 = 清除通道 x 中断 (CHx_TCIT 和CHx_LTCIT)
CH2_IT	[2]	W	通道2中断清除位 0 = 无效No effect 1 = 清除通道 x 中断 (CHx_TCIT 和CHx_LTCIT)
CH1_IT	[1]	W	通道1中断清除位 0 = 无效No effect 1 = 清除通道 x 中断 (CHx_TCIT 和CHx_LTCIT)
CH0_IT	[0]	W	通道0中断清除位 0 = 无效No effect 1 = 清除通道 x 中断 (CHx_TCIT 和CHx_LTCIT)

8 GPIO

8.1 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 A: 16 位输入/输出端口, PA0 ~ PA15
- Port B: 16 位输入/输出端口, PB0 ~ PB15
- Port C: 16 位输入/输出端口, PC0 ~ PC15
- Port D: 6 位输入/输出端口, PD0 ~ PD5

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。

8.1.1 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

8.1.2 管脚描述

Table 8-1 管脚描述

Pin Name	Function	I/O Type	Active Level	Comments
PA[15:0]	通用 I/O 口	I/O	-	-
PB[15:0]	通用 I/O 口	I/O	-	-
PC[15:0]	通用 I/O 口	I/O	-	-
PD[5:0]	通用 I/O 口	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。

8.2 功能描述

8.2.1 电路图

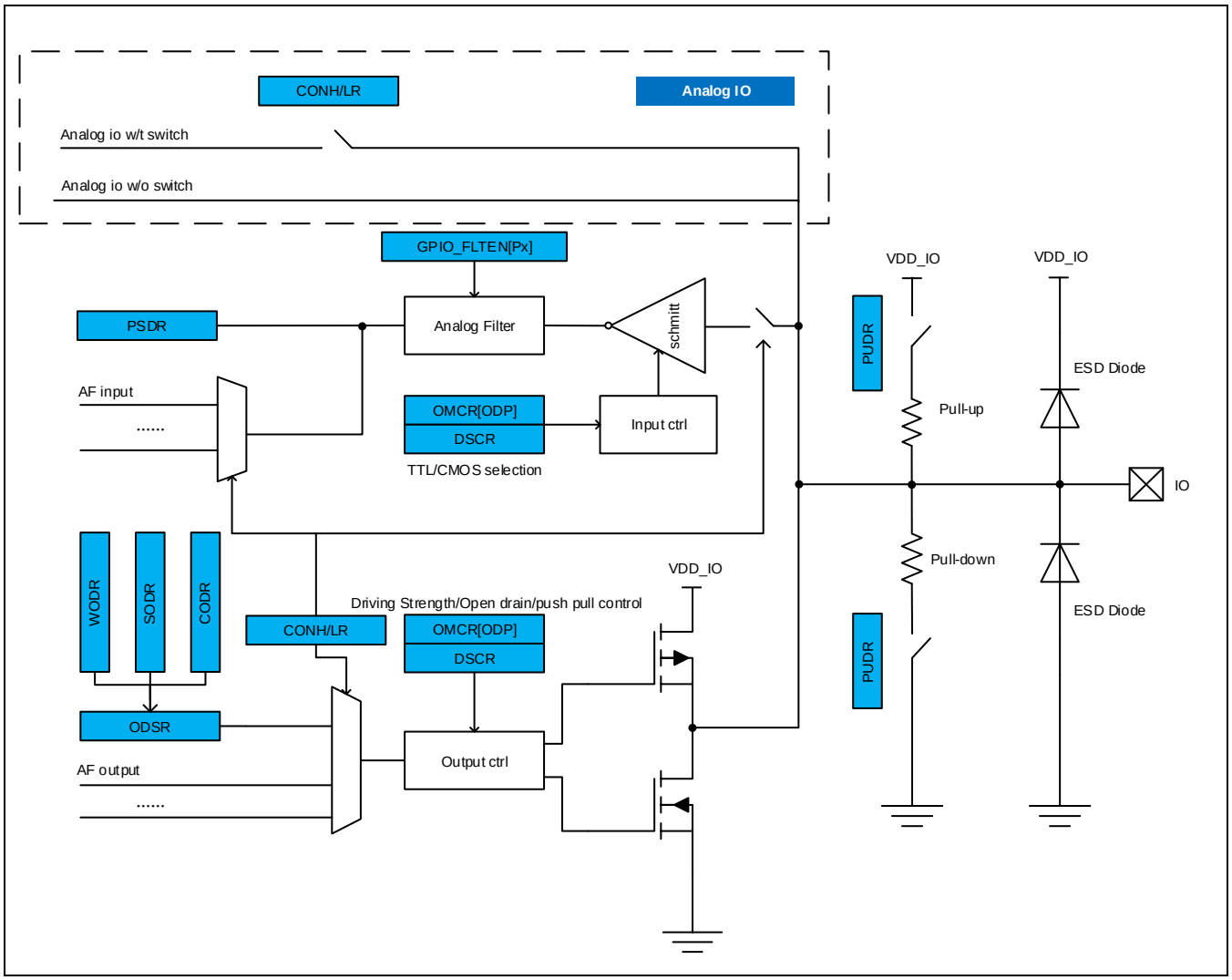


Figure 8-1 GPIO原理图

8.2.2 工作原理

8.2.2.1 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。

8.2.2.2 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。

8.2.3 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。

8.2.4 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 0.8VDD，低电平最大输入阈值为 0.2VDD。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。

8.2.5 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。

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。

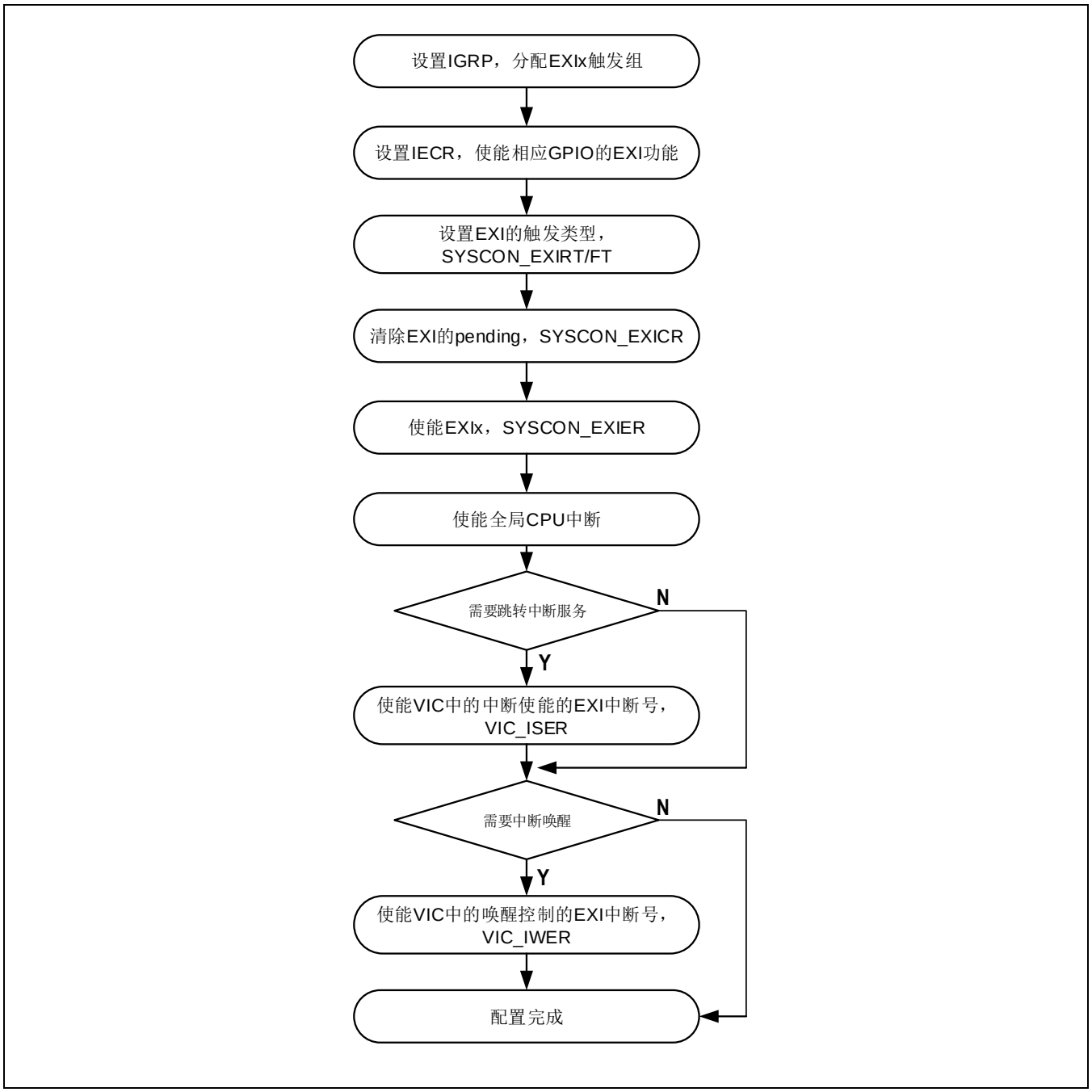


Figure 8-2 GPIO外部中断配置流程

8.3 寄存器说明

8.3.1 寄存器表

Base Address of GPIOA: 0x60000000
Base Address of GPIOB: 0x60002000
Base Address of GPIOC: 0x60004000
Base Address of GPIOD: 0x60006000

Register	Offset	Description	Reset Value
GPIO_CONLR	0x00	低位控制寄存器	0x00000000
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	0x000000FF

8.3.2 GPIO_CONLR(低位控制寄存器)

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
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
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

8.3.3 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）。

8.3.4 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。

8.3.5 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。			

8.3.6 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。			

8.3.7 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。			

8.3.8 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。			

8.3.9 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。			

8.3.10 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。

8.3.11 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）

对于LED SEG管脚: 0-强驱，1-使能恒流源。(LED SEG管脚没有弱驱功能)

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。

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。

8.3.12 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。

8.3.13 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			

8.3.14 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

8.3.15 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

8.3.16 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: GPIOA.x 被选中 0001: RSVD 0010: GPIOB.x 被选中 0011: RSVD 0100: GPIOC.x 被选中 0101: RSVD 0110: GPIOD.x 被选中 Other: RSVD ‘x’ 表示组数			

8.3.17 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: GPIOA.x 被选中 0001: RSVD 0010: GPIOB.x 被选中 0011: RSVD 0100: GPIOC.x 被选中 0101: RSVD 0110: GPIOD.x 被选中 Other: RSVD ‘x’ 表示组数			

8.3.18 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：PD0 被选中 1h：PD1 被选中 2h：PD2 被选中 3h：PD3 被选中 4h：PD4 被选中 5h：PD5 被选中 6h：RSVD 7h：RSVD 8h：RSVD 9h：RSVD ah：RSVD bh：RSVD ch：RSVD dh：RSVD eh：RSVD fh： RSVD
GRP18	[11:8]	RW	选择外部中断组18 0h：PC0 被选中 1h：PC1 被选中 2h：PC2 被选中 3h：PC3 被选中 4h：PC4 被选中 5h：PC5 被选中 6h：PC6 被选中 7h：PC7 被选中 8h：PC8 被选中 9h：PC9 被选中 ah：PC10 被选中 bh：PC11 被选中 ch：PC12 被选中 dh：PC13 被选中 eh：PC14 被选中 fh： PC15 被选中

GRP17	[7:4]	RW	选择外部中断组17 0h：PB0 被选中 1h：PB1 被选中 2h：PB2 被选中 3h：PB3 被选中 4h：PB4 被选中 5h：PB5 被选中 6h：PB6 被选中 7h：PB7 被选中 8h：PB8 被选中 9h：PB9 被选中 ah：PB10 被选中 bh：PB11 被选中 ch：PB12 被选中 dh：PB13 被选中 eh：PB14 被选中 fh： PB15 被选中
GRP16	[3:0]	RW	选择外部中断组16 0h：PA0 被选中 1h：PA1 被选中 2h：PA2 被选中 3h：PA3 被选中 4h：PA4 被选中 5h：PA5 被选中 6h：PA6 被选中 7h：PA7 被选中 8h：PA8 被选中 9h：PA9 被选中 ah：PA10 被选中 bh：PA11 被选中 ch：PA12 被选中 dh：PA13 被选中 eh：PA14 被选中 fh： PA15 被选中
0000: GPIOA.x 被选中 0001: RSVD 0010: GPIOB.x 被选中 0011: RSVD 0100: GPIOC.x 被选中 0101: RSVD 0110: GPIOD.x 被选中 Other: RSVD ‘x’ 表示组数			

8.3.19 GPIO_CLKEN(GPIO组时钟使能控制寄存器)

Address = Base Address+ x0C, 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																									CLK_D	RSVD	CLK_C	RSVD	CLK_B	RSVD	CLK_A
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	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CLK_D	[6]	RW	
CLK_C	[4]	RW	
CLK_B	[2]	RW	
CLK_A	[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			

9 基本计数器 (Basic Timer)

9.1 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。

9.1.1 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。

9.1.2 管脚描述

Table 9-1 BT 相关功能管脚描述

管脚名称	功能描述
BT_OUT	PWM 波形输出

9.2 功能描述

9.2.1 模块框图

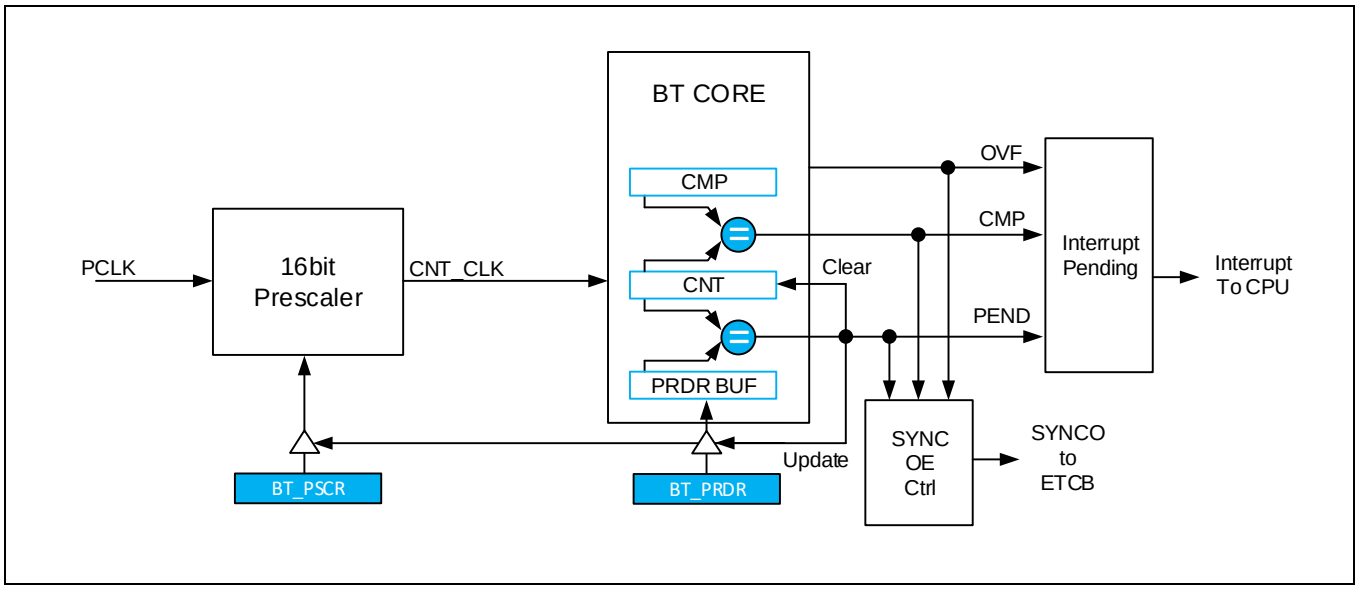


Figure 9-1 模块结构示意图

9.2.2 基本功能描述

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

BT的计数器的启动可以通过两种方式触发：

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

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

PRDR和PSCR寄存器同样具有SHADOW寄存器，对PRDR和PSCR的写入被保存在SHADOW寄存器中，当周期事件(PEND)发生时，BT会将SHADOW寄存器中的设置更新到ACTIVE寄存器中。除了PEND事件，还可以通过软件写CR[UPDATE]控制位，触发强制更新。或者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）。

BT支持One Pulse工作模式，通过CR[OPM]控制位选择。当OPM被使能后，BT在启动后只进行一次计数。当CNT=PRDR后，计数器停止计数。当CR[CNTRLD]被禁止时，OPM无效。

9.2.3 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。

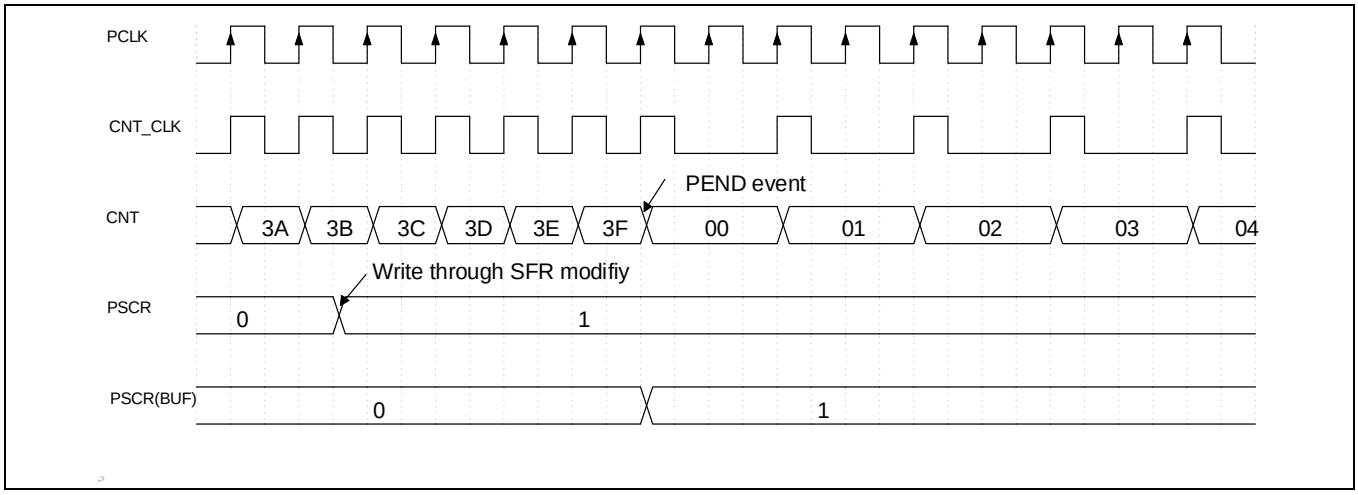


Figure 9-2 BT 周期计数模式

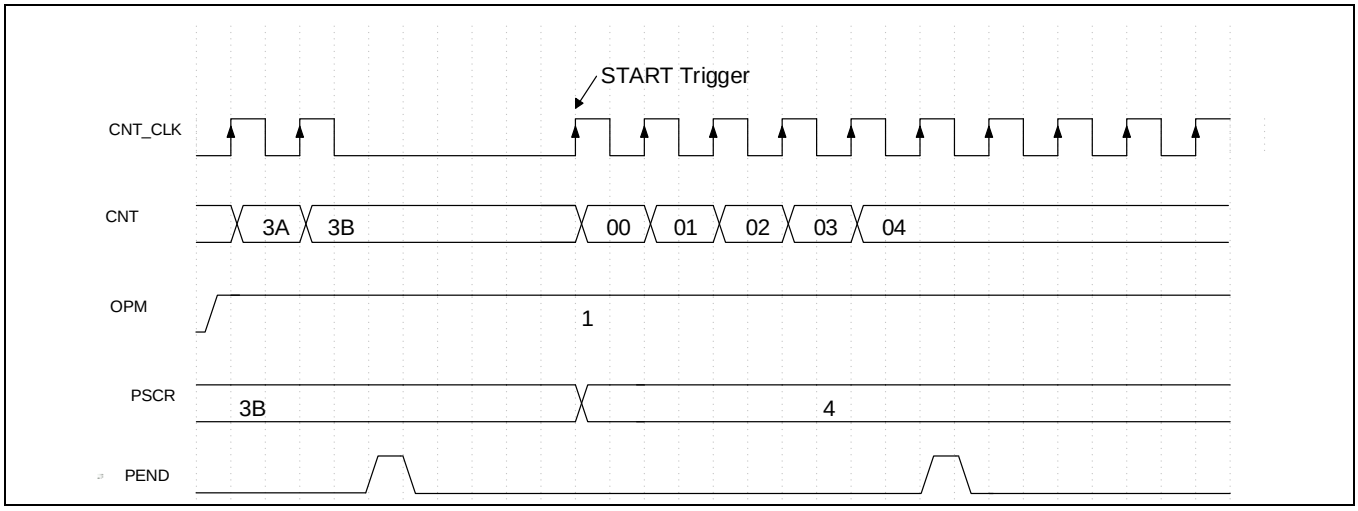


Figure 9-3 BT One Pulse 计数模式

9.2.4 同步触发

Basic Timer支持通过ETCB和芯片内其他模块的同步触发，其具有两个同步输入端口和一个触发输出端口。

- 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START/STOP控制。只有当Basic Timer处于运行状态时，SYNC PORT1事件才有效。第一次触发关闭Basic Timer，再次触发开启Basic Timer，依

此类推。

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

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。

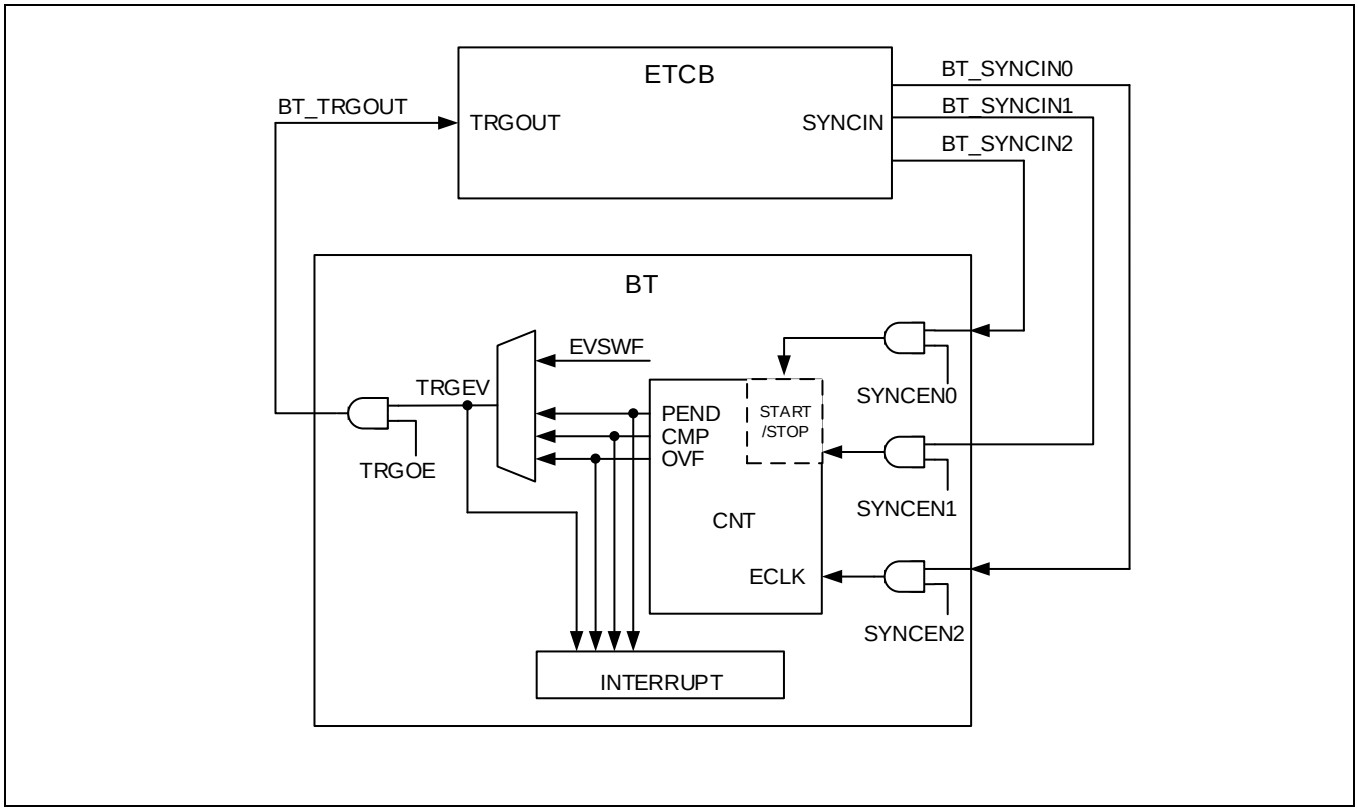


Figure 9-4 同步触发原理

9.2.5 波形发生

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

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

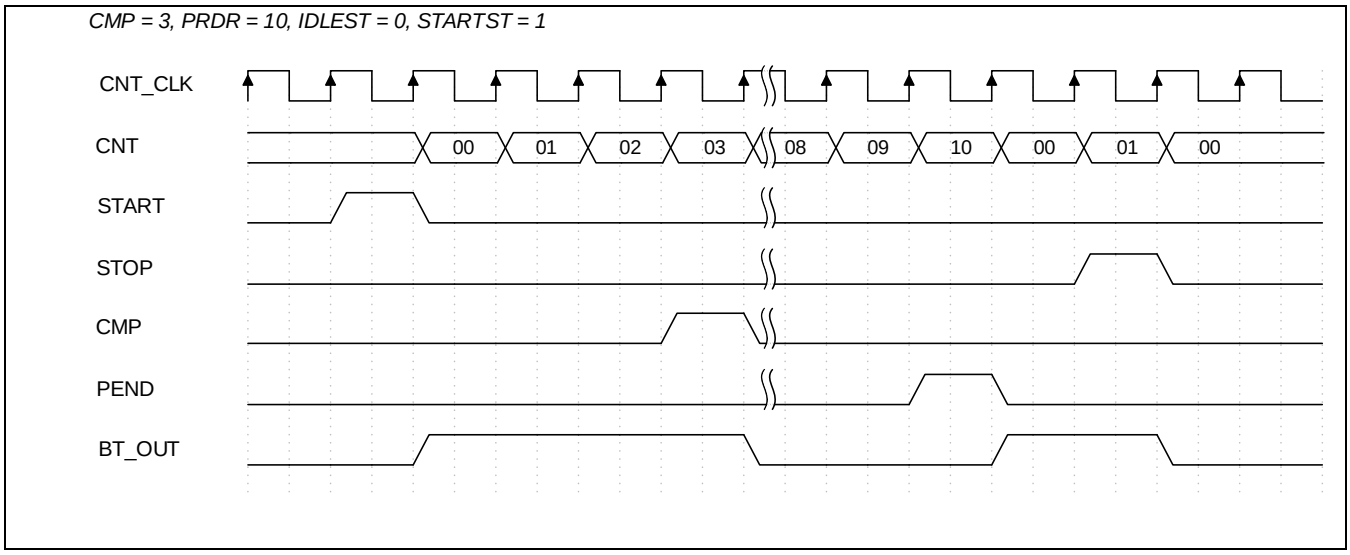


Figure 9-5 BT 输出波形时序图

9.2.6 中断控制

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

PEND事件：计数器周期结束时发生。

CMP事件：计数器计数值等于CMP寄存器设置时发生。

OVF事件：计数器计数溢出（0xFFFF）时发生。

TRGEV事件：同步触发输出事件有输出是发生。

9.3 寄存器说明

9.3.1 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	复位/启停控制寄存器	0x00000000
BT_CR	0x004	通用控制寄存器	0x00000000
BT_PSCR	0x008	计数器预分频寄存器	0x00000000
BT_PRDR	0x00C	周期值寄存器	0x00000000
BT_CMP	0x010	比较值寄存器	0x00000000
BT_CNT	0x014	计数值寄存器	0x00000000
BT_EVTRG	0x018	事件生成控制寄存器	0x00000000
BT_EVSWF	0x024	事件软件触发寄存器	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

9.3.2 BT_RSSR(复位/启停控制寄存器)

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	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，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。

9.3.3 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	REARMx			RSVD	OSTMDx			RSVD		AREARM1		AREARM0		RSVD	CNTRLD	SYNCMD	RSVD				SYNCEN2	SYNCEN1	SYNCEN0	STARTST	IDLEST	EXTCKM	OPM	SHDWSTP	UPDATE	DBGEN	CLKEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	W	W	W	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	W	RW	RW

Name	Bit	Type	Description
REARMx	[30:28]	W	在一次性同步触发模式下，软件重置当前通道状态控制位。 当读取时，返回当前通道状态 0h：允许触发 1h：已经检测到触发，不允许后续触发 当写入时， 0h：无效 1h：清除当前通道状态，并允许新的触发
OSTMDx	[26:24]	RW	一次性同步触发模式选择。 0h：连续触发模式 1h：一次性触发模式 当该输入通道被设置为一次性触发模式后，在一次触发事件被检测到后，该通道将不允许后续的触发事件通过，直到被软件重置（REARM）后才允许新的触发事件通过。
AREARM1	[21:20]	RW	硬件自动REARM控制位。 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]	W	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。

9.3.4 BT_PSCR(计数器预分频寄存器)

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)

9.3.5 BT_PRDR(周期值寄存器)

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。

9.3.6 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，下个计数周期开始BT输出将翻转。

9.3.7 BT_CNT(计数值寄存器)

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。

9.3.8 BT_EVTRG(事件生成控制寄存器)

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											TRGOE	RSVD																TRGSEL			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW

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

9.3.9 BT_EVSWF(事件软件触发寄存器)

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																															EVSWF
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW

Name	Bit	Type	Description
EVSWF	[0]	RW	软件产生一次TRGEV事件。 0h：写入’0’无效。 1h：软件产生一次TRGEV事件。

9.3.10 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																												TRGEV	OVF	CMP	PEND
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
TRGEV	[3]	R	事件触发中断请求原始标志状态。
OVF	[2]	R	OVF中断请求原始标志状态。
CMP	[1]	R	CMP Match中断请求原始标志状态。
PEND	[0]	R	PEND周期结束中断请求原始标志状态。

原始中断标志表示中断事件发生，通过设置IMCR中相应位，可以允许该中断请求CPU中断。原始中断标志位需要通过软件清除。

0h：该中断未置位

1h：该中断已置位

9.3.11 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																												EVTRG	OVF	CMP	PEND
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW

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

9.3.12 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																												EVTRG	OVF	CMP	PEND
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
EVTRG	[3]	R	事件触发中断请求标志状态。
OVF	[2]	R	OVF中断请求标志状态。
CMP	[1]	R	CMP Match中断请求标志状态。
PEND	[0]	R	PEND周期结束中断请求标志状态。

中断标志表示CPU中断请求的状态，通过写ICR寄存器可以清除该标志位。

0h：该中断未置位

1h：该中断已置位

9.3.13 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																												EVTRG	OVF	CMP	PEND
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W

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

10 计数器A

10.1 概述

本章节介绍计数器 A，一个 16 位的计数器，用来产生载波频率。

10.1.1 主要特性

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

注意：
CPU 时钟必须比计数器 A 的时钟快。

10.2 功能描述

10.2.1 模块框图

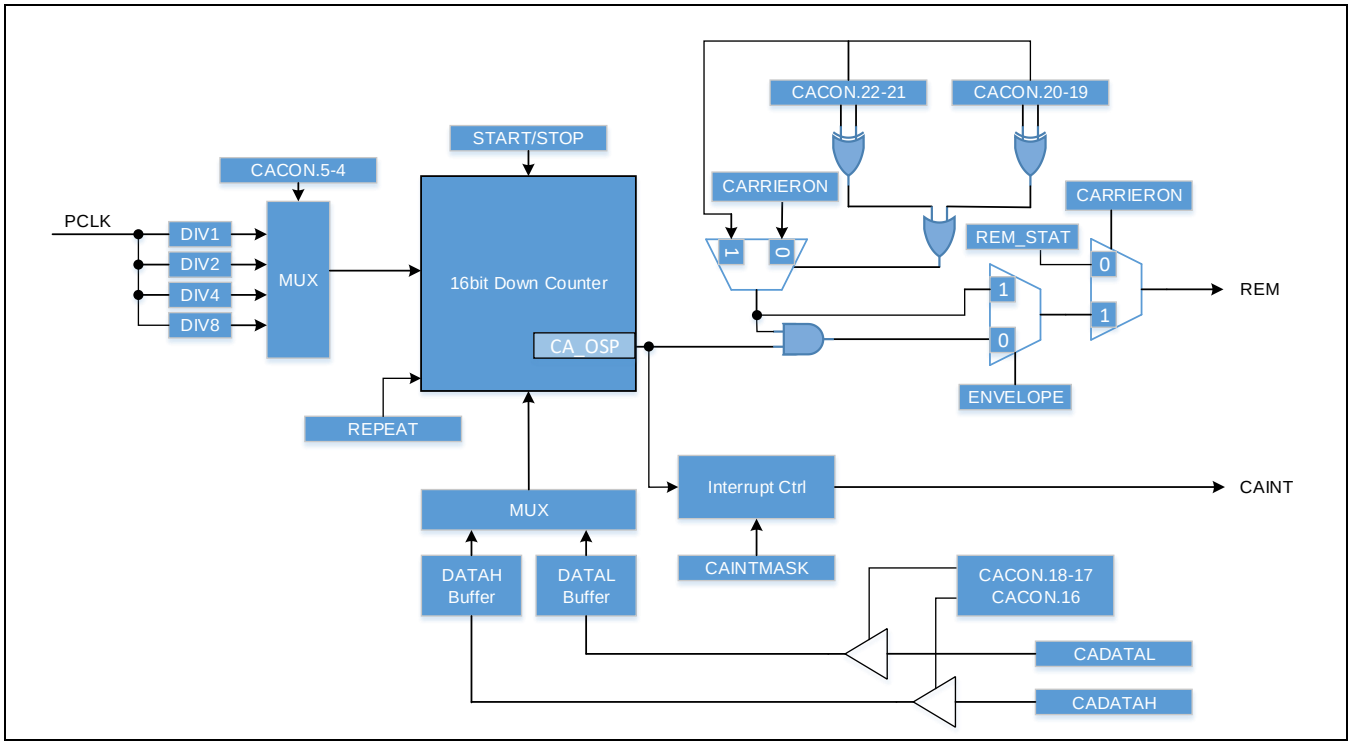


Figure 10-1 计数器A模块框图

10.2.2 工作原理

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

Table 10-1 REM管脚输出状态

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

注意：

1) 如果 BTMATCH_REM_ONOFF 为 1，那么在 BT0 匹配中断发生时，REM 输出状态将被改变。

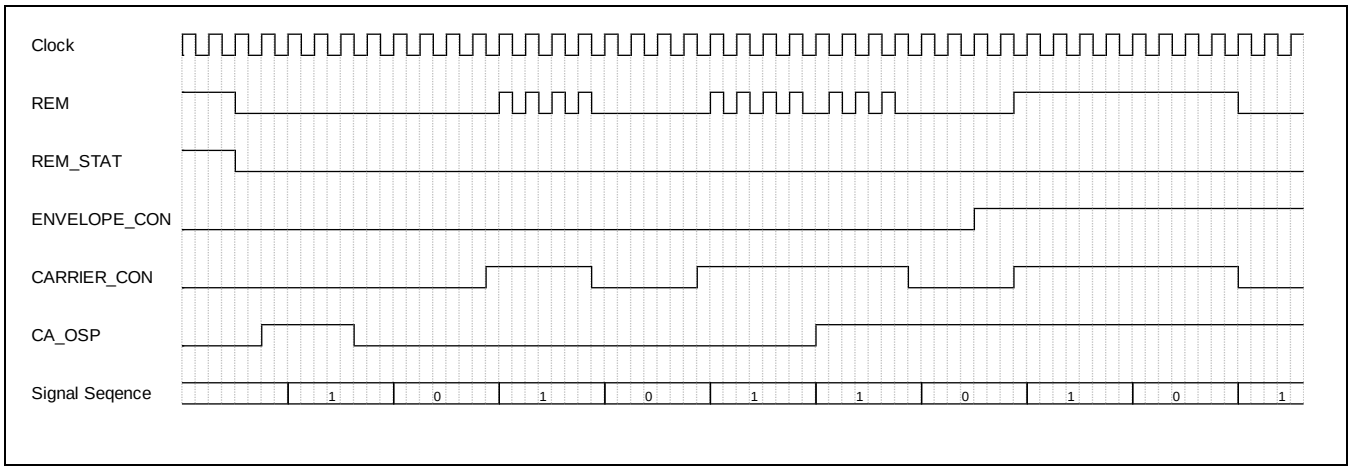


Figure 10-2 计数器A输出信号波形

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

在设置 HW_STROBE_DATA 后，计数值的更新也可以由硬件自动更新，这个 HW_STROBE_DATA 位不会覆盖 SW_STROBE_DATA。当 REM 变为高的时候，计数值将会被更新到计数器中。如果选择了 BT0 脉冲匹配中断，那么当 BT0 脉冲匹配中断发生时，计数值会被更新到计数器中。如果选择了 BT0 周期结束中断，那么当 BT0 周期结束中断发生时，计数值会被更新到计数器中。当两种 BT0 中断都被选择时，两个中断发生的时候都会更新计数值。当 SW_STROBE_DATA 被置 1 时，计数值也会更新到计数器中，而且更新状态(是否更新完成)可以读回 SW_STROBE_DATA 位来进行查询。另外，每次写 START 位的时候，计数值会被自动更新到计数器中。

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

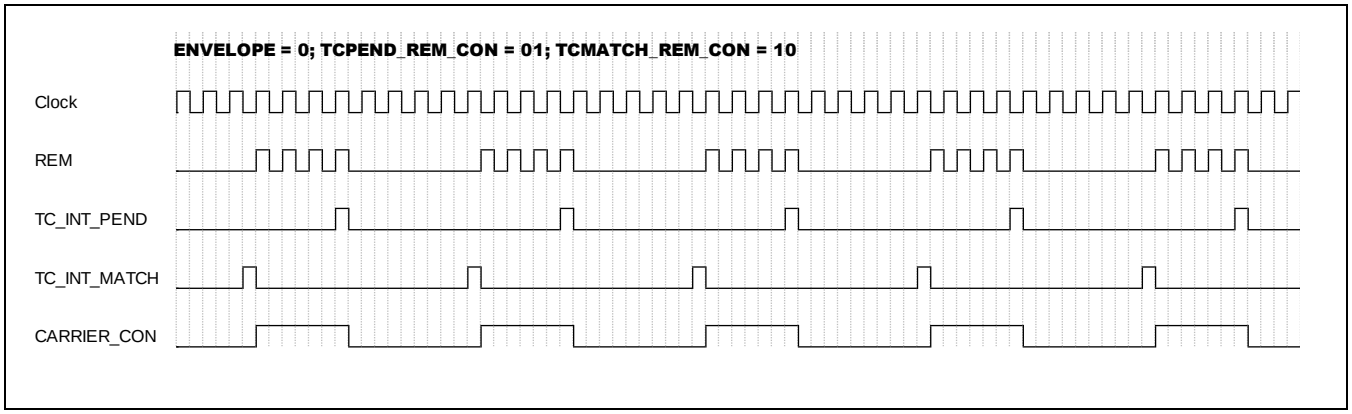


Figure 10-3 硬件触发的波形

10.2.3 脉冲宽度计算

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

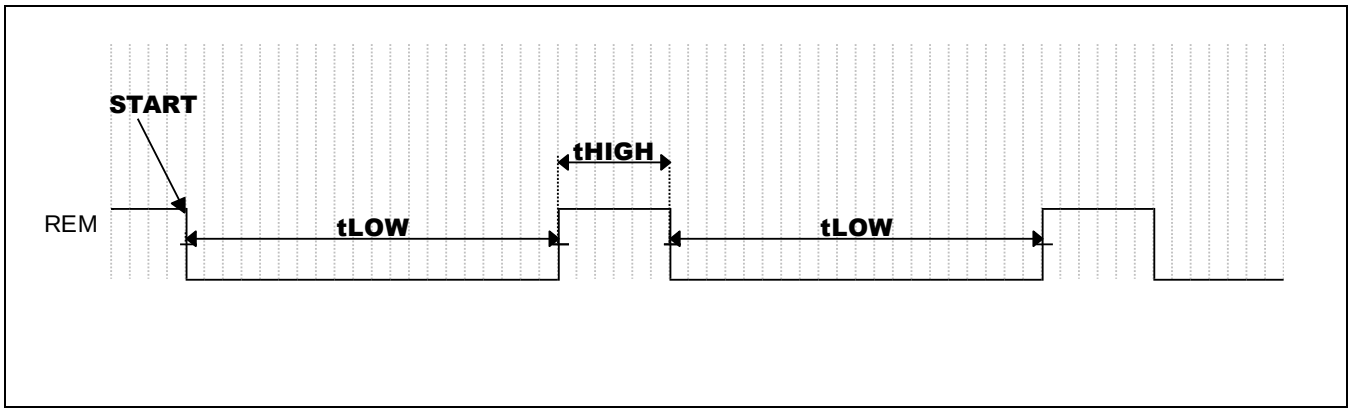


Figure 10-4 重复模式波形

当 CA_OSP = 0,

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

当 CA_OSP = 1,

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

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

[方法 1] 当 CA_OSP = 0,

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

[方法 2] 当 CA_OSP = 1,

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

10.2.4 中断

计数器 A 产生 2 个中断。

- 低电平结束中断
- 高电平结束中断

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

10.2.5 编程提示

下面的例子演示如何产生一个 38KHz, 1/3 占空比的信号。

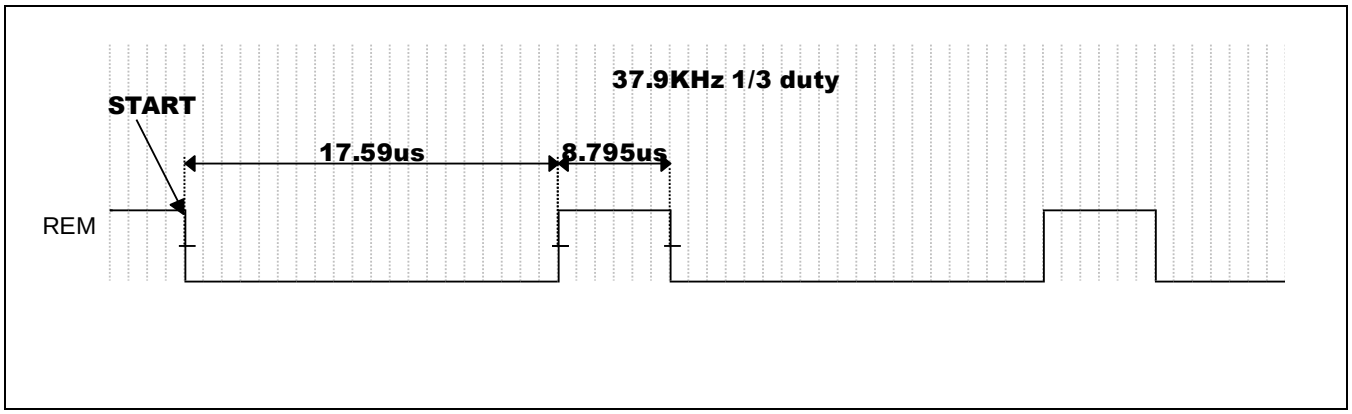


Figure 10-5 38KHz, 1/3占空比的REM输出波形

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

10.3 寄存器说明

10.3.1 寄存器表

Base Address of COUNTERA: 0x40050000

Register	Offset	Description	Reset Value
CADATAH	0x00	计数器A DATAH寄存器	0x00000000
CADATAL	0x04	计数器A DATAL寄存器	0x00000000
CACON	0x08	计数器A控制寄存器	0x00000030
CAINTMASK	0x0C	计数器A中断控制寄存器	0x00000000

10.3.2 CADATAH(计数器A DATAH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																CADATAH															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CADATAH	[15:0]	RW	DATAH控制计数器A输出的高电平宽度

10.3.3 CADATAL(计数器A DATAL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																CADATAL															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CADATAL	[15:0]	RW	DATAL控制计数器A输出的低电平宽度

10.3.4 CACON(计数器A控制寄存器)

Address = Base Address+ 0x08, 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						CARRIERON	ENVELOPE	REM_STAT	BTPEND_REM_CON		BTMATCH_REM_CON		HW_STROBE_DATA		SW_STROBE_DATA	RSVD										CA_CLK		STOP	START	MODE	OSP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	W	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CARRIERON	[25]	RW	载波信号控制位 0: 关闭载波 1: 打开载波
ENVELOPE	[24]	RW	REM输出信号选择位 0: 选择载波信号为输出 1: 选择包络信号为输出
REM_STAT	[23]	RW	当关闭载波时REM的输出状态 0: 低 1: 高
BTPEND_REM_CON	[22:21]	RW	当BT周期结束中断发生时，硬件触发控制载波的打开和关闭 00/11: 禁止CARRIERON的硬件自动触发 01: BT周期结束中断发生时，CARRIERON位会被硬件自动清零 10: BT周期结束中断发生时，CARRIERON位会被硬件自动置位
BTMATCH_REM_CON	[20:19]	RW	当BT脉冲匹配中断发生时，硬件触发控制载波的打开和关闭 00/11: 禁止CARRIERON的硬件自动触发 01: BT脉冲匹配中断发生时，CARRIERON位会被硬件自动清零 10: BT脉冲匹配中断发生时，CARRIERON位会被硬件自动置位
HW_STROBE_DATA	[18:17]	RW	使能计数值寄存器的硬件自动更新功能 X1: 当BT脉冲匹配中断发生时，计数值会自动载入计数器 1X: 当BT周期结束中断发生时，计数值会自动载入计数器
SW_STROBE_DATA	[16]	W	使能计数值寄存器软件更新 当该位置位时，CADATAH和CADATAL的值会更新到计数器中

CA_CLK	[5:4]	RW	输入时钟频率选择位 00: PCLK 01: PCLK/2 10: PCLK/4 11: PCLK/8
STOP	[3]	RW	停止位 计数器A在工作时，该位为0。需要停止计数器A，则需将该位写1。
START	[2]	RW	启动位。 写1会启动计数器A。 计数器A开始计数后，该位会被自动清零。
MODE	[1]	RW	模式选择位 0: 单次(One Shot)模式 1: 重复模式
OSP	[0]	RW	载波输出波形的初始极性选择 0: 低 1: 高

10.3.5 CAINTMASK(计数器A中断控制寄存器)

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																														PLEND_INT	PHEND_INT
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW

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

11 增强型通用定时器 (GPTA)

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。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。

11.1.1 主要特性

- 16 位可复位计数器
- 可编程计数器计数方式
  - 递增计数（Up-counting）
  - 递减计数（Down-counting）
  - 递增递减计数（Up-down-counting）
- 两路波形产生控制单元，支持双路独立输出：
  - 两路独立的 PWM 输出，单边沿工作
  - 两路独立的 PWM 输出，双边沿对称工作
- 通过软件异步重置 PWM 的波形输出
- 支持片间多设备同步
  - 支持多个 TIMER 间的同步触发
  - 触发源包括 GPIO 输入，其他外设触发，软件设置和事件触发
  - 支持单次触发和连续触发模式
- 支持单脉冲输出模式
- 支持突发计数模式
- 支持通过外部时钟计数
- 支持事件计数器，可通过配置事件计数器（最大 15）触发相应中断
- 支持捕获模式，最多支持 4 个捕获值存储。

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
GPTA_CHA	时钟控制使能	输出波形	输出波形
GPTA_CHB	时钟控制使能	输出波形	输出波形

11.2 功能描述

11.2.1 模块框图

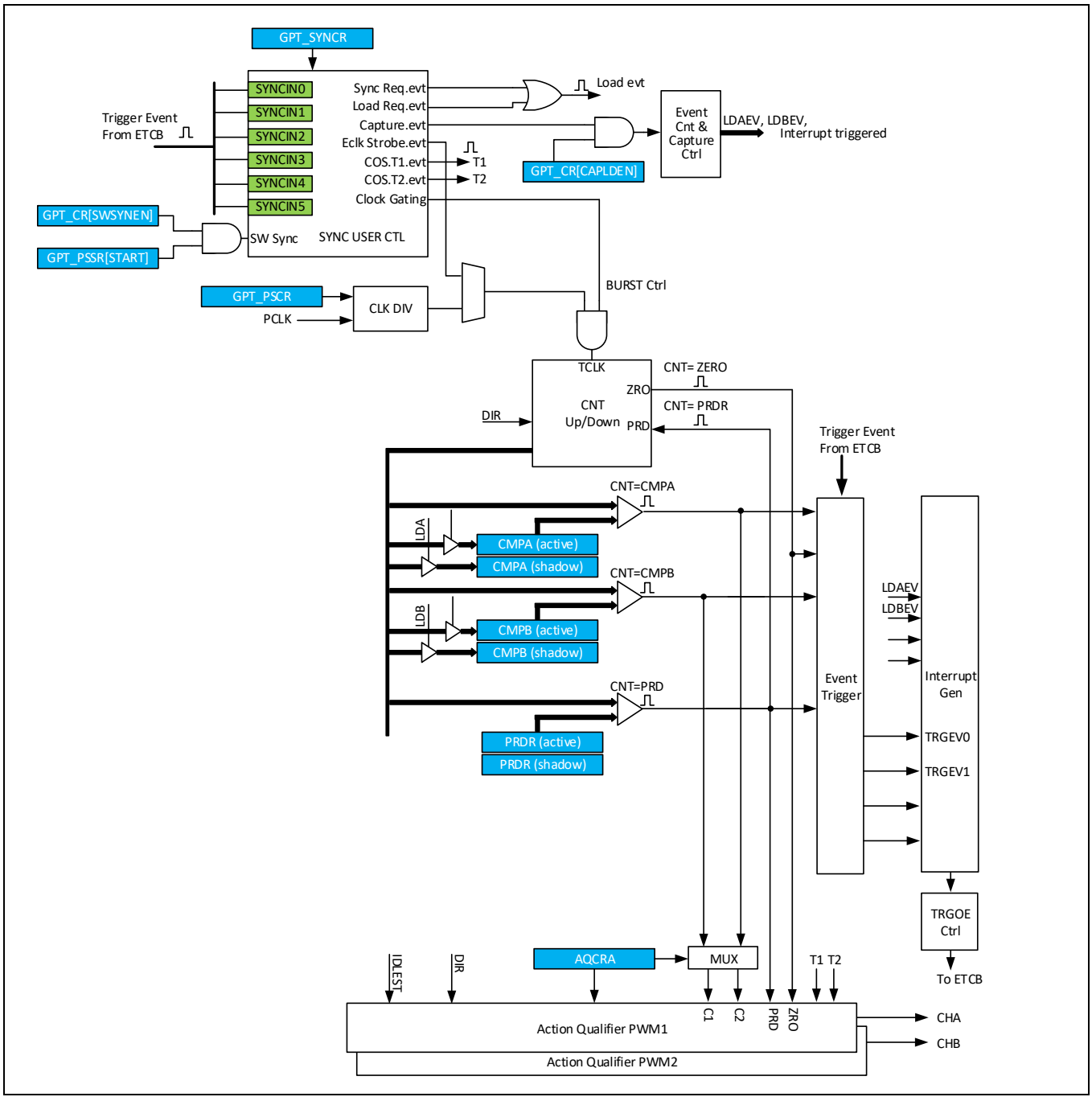


Figure 11-1 模块结构示意图

11.3 基本功能描述

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

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

11.3.1 时钟源

11.3.1.1 概述

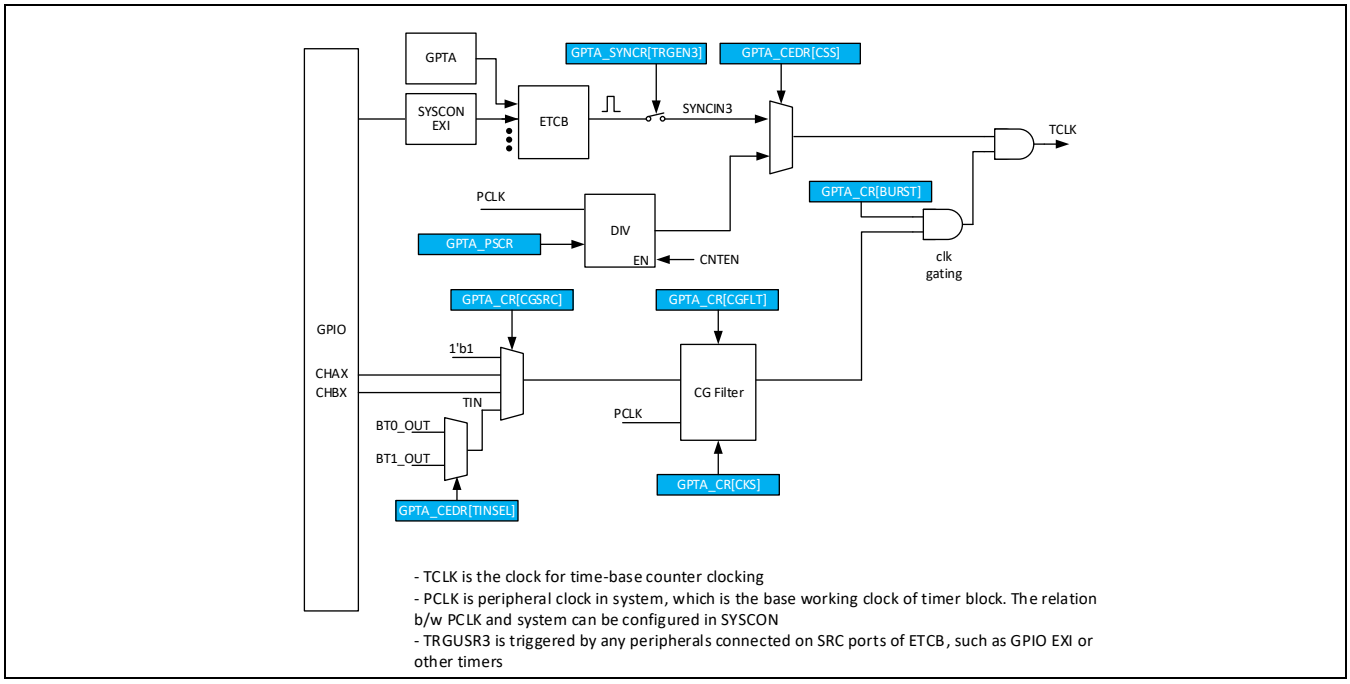


Figure 11-2 时钟控制模块

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。

11.3.1.4 群脉冲时钟

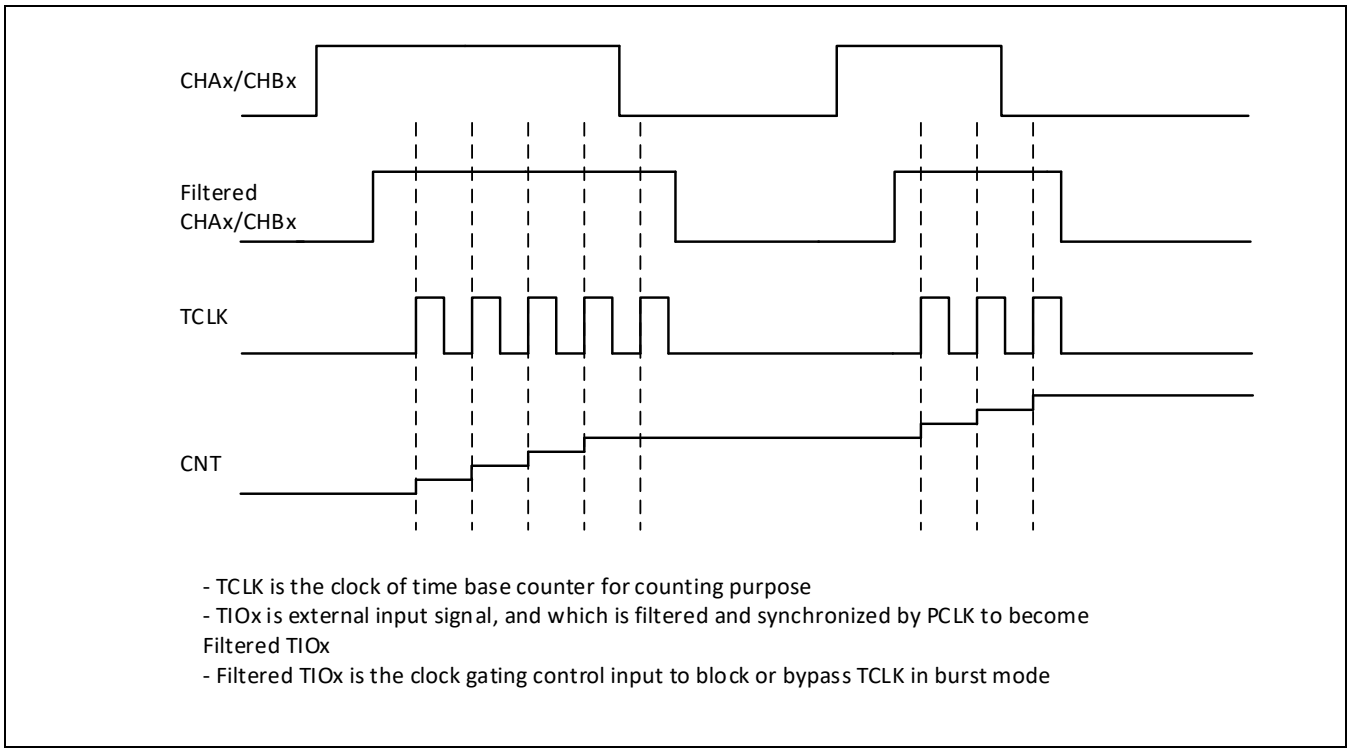


Figure 11-3 群脉冲时钟模式时序

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

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

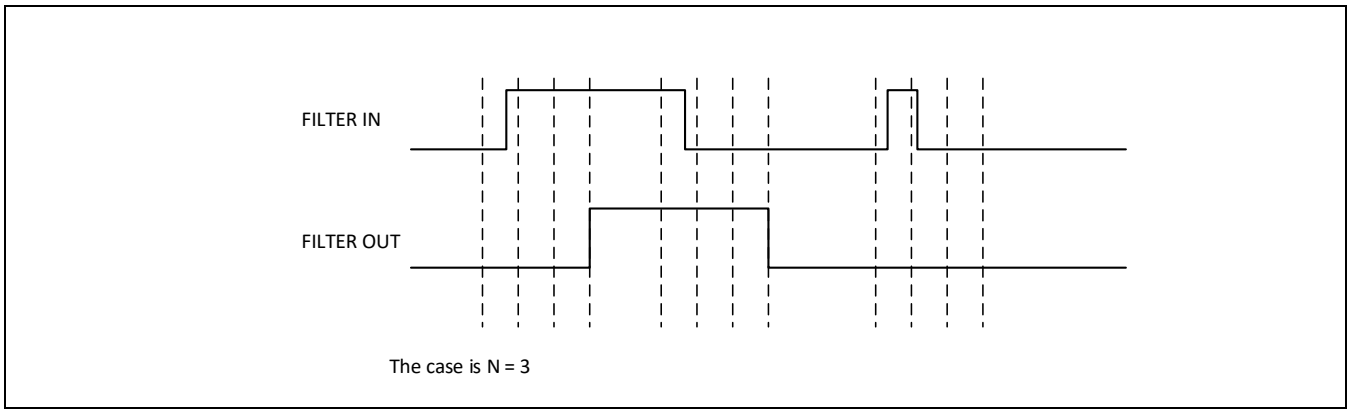


Figure 11-4 CG Filter 数字滤波器的原理

11.3.2 时基控制

11.3.2.1 概述

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

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

时基模块的寄存器包括：

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

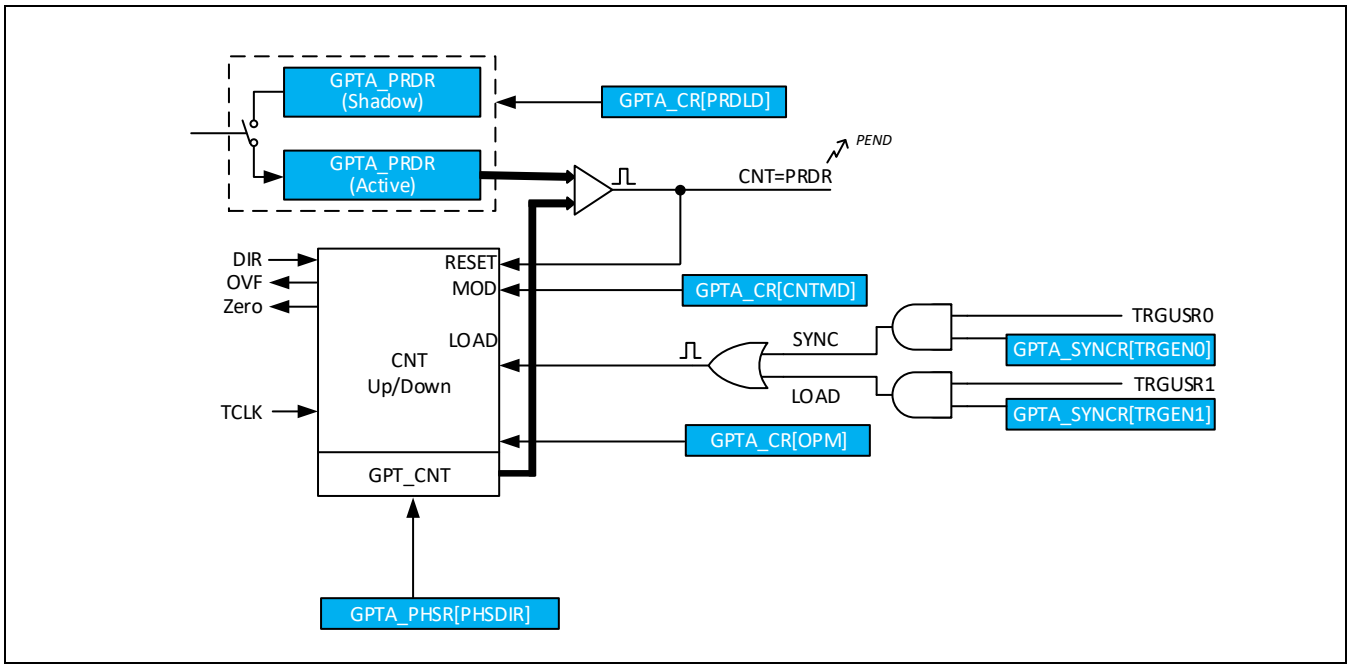


Figure 11-5 计数器时基模块

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

- 递增模式（Up-Counting Mode）：

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

- 递减模式：（Down-Counting Mode）

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

- 递增递减模式：（Up-Down-Counting Mode）

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

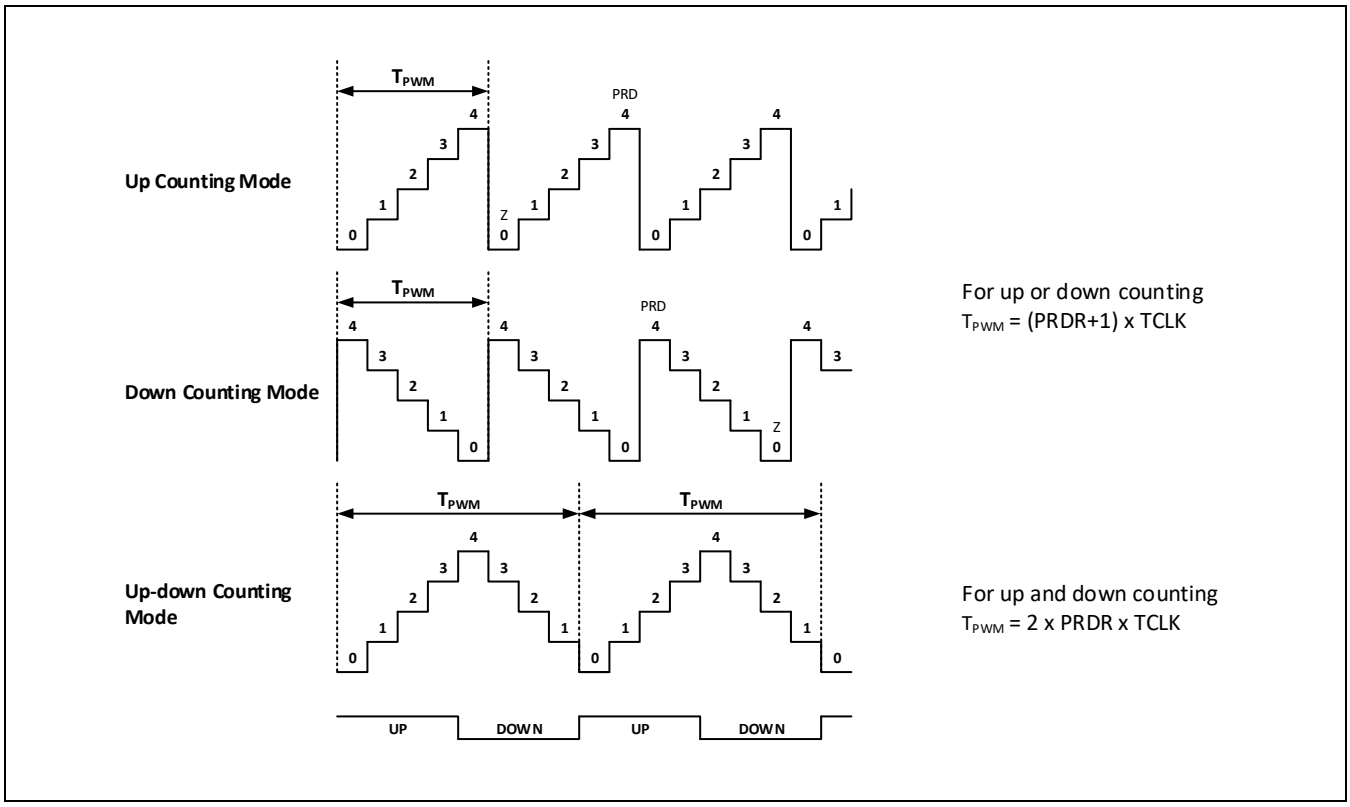


Figure 11-6 计数器工作模式

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。

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，或者是PRDR的设置值。

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

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

- **PRDR寄存器的Shadow模式**

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

- **PRDR寄存器的立即加载模式**

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

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

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

11.3.2.3 计数模式和时序

时基计数器可以分为四种工作模式：

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

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

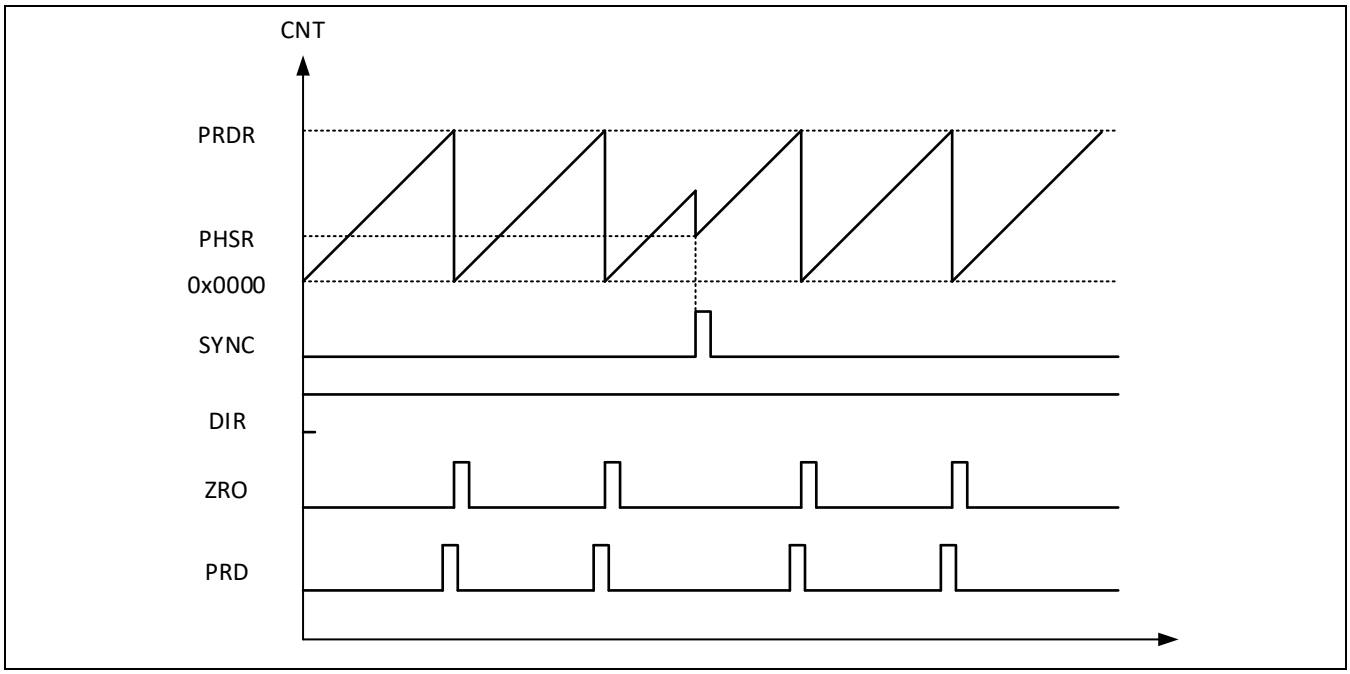


Figure 11-7 递增工作模式

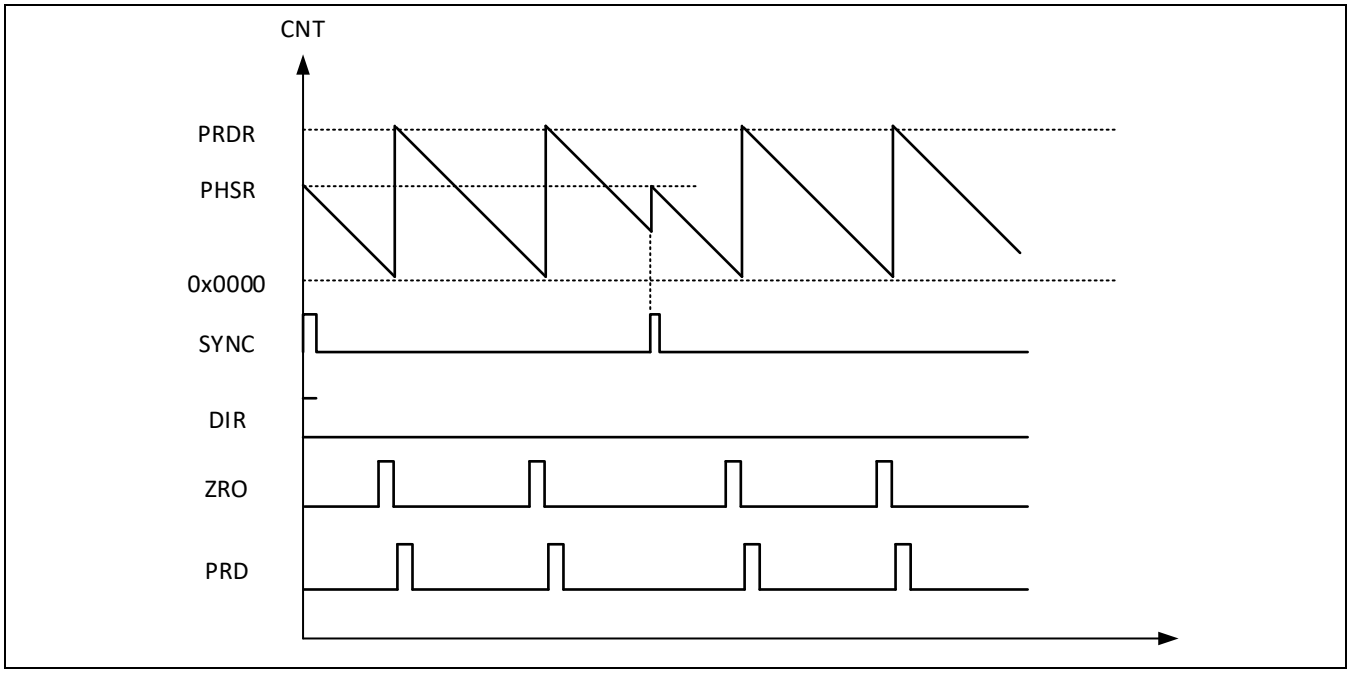


Figure 11-8 递减工作模式

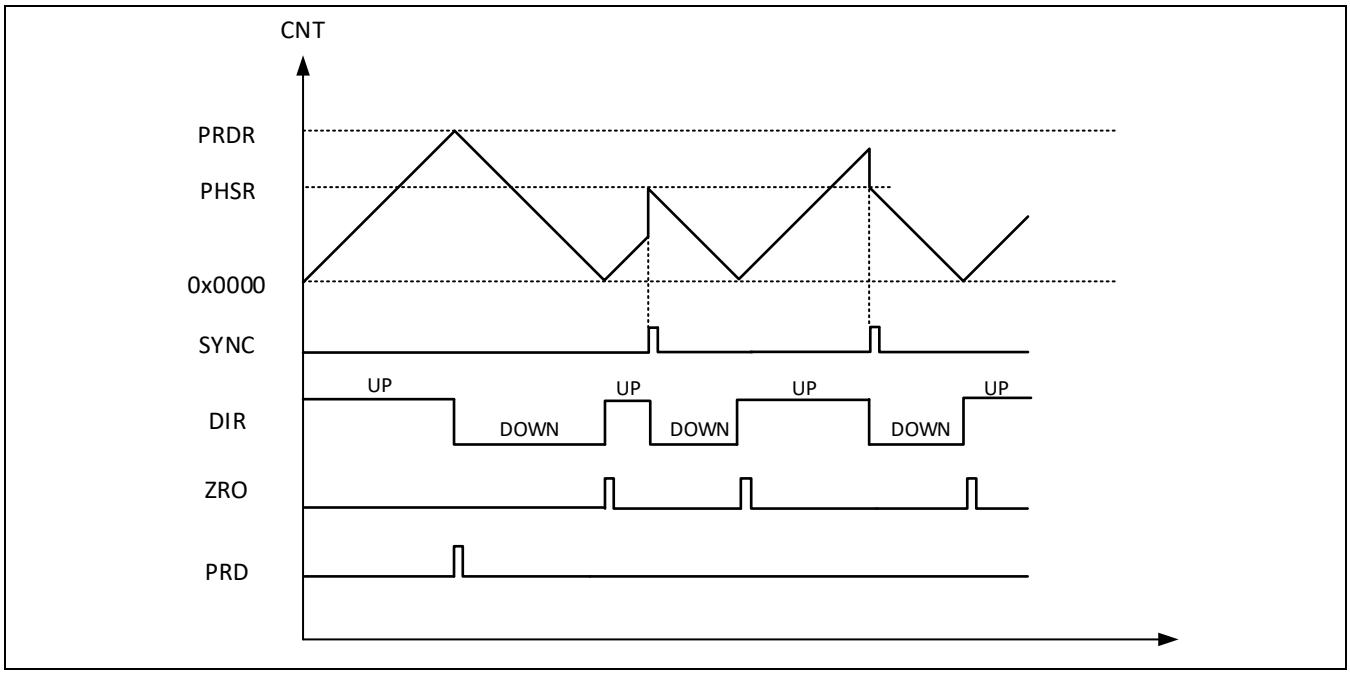


Figure 11-9 递增递减工作模式

11.3.2.4 全局载入控制

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

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

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

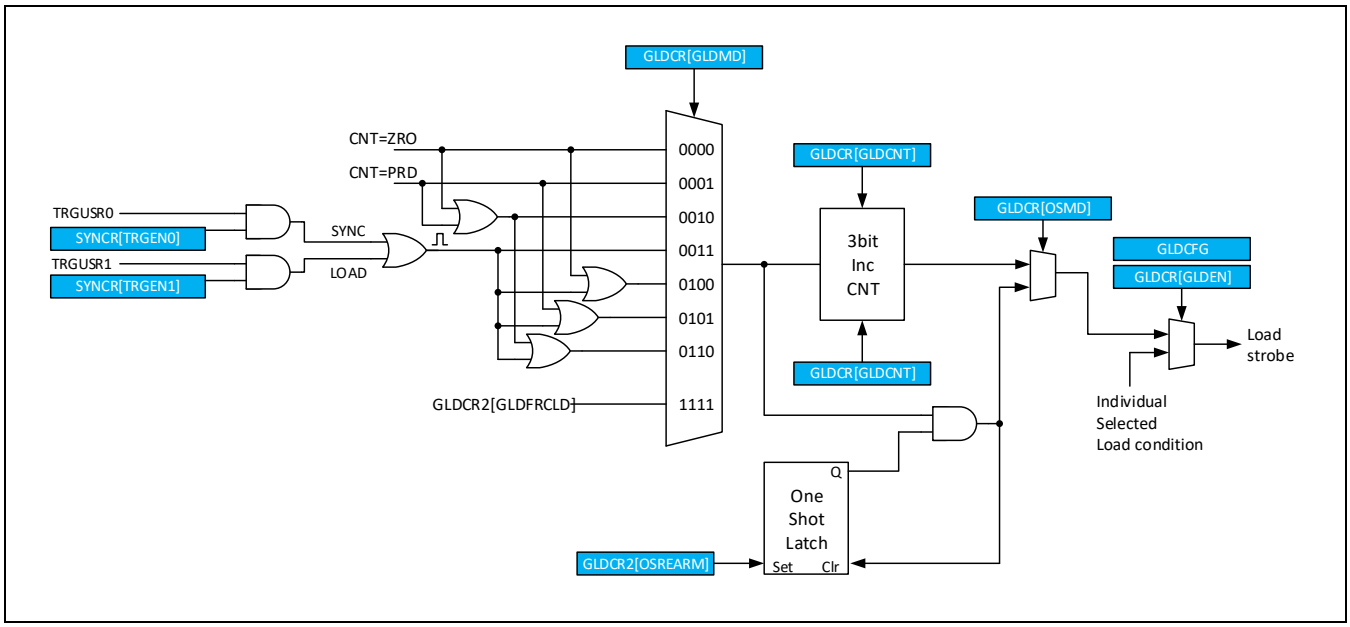


Figure 11-10 全局载入控制

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、CM）的值，当计数值等于其中任意一个比较值时，比较控制模块将产生一个相应的事件触发。主要特性如下：

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

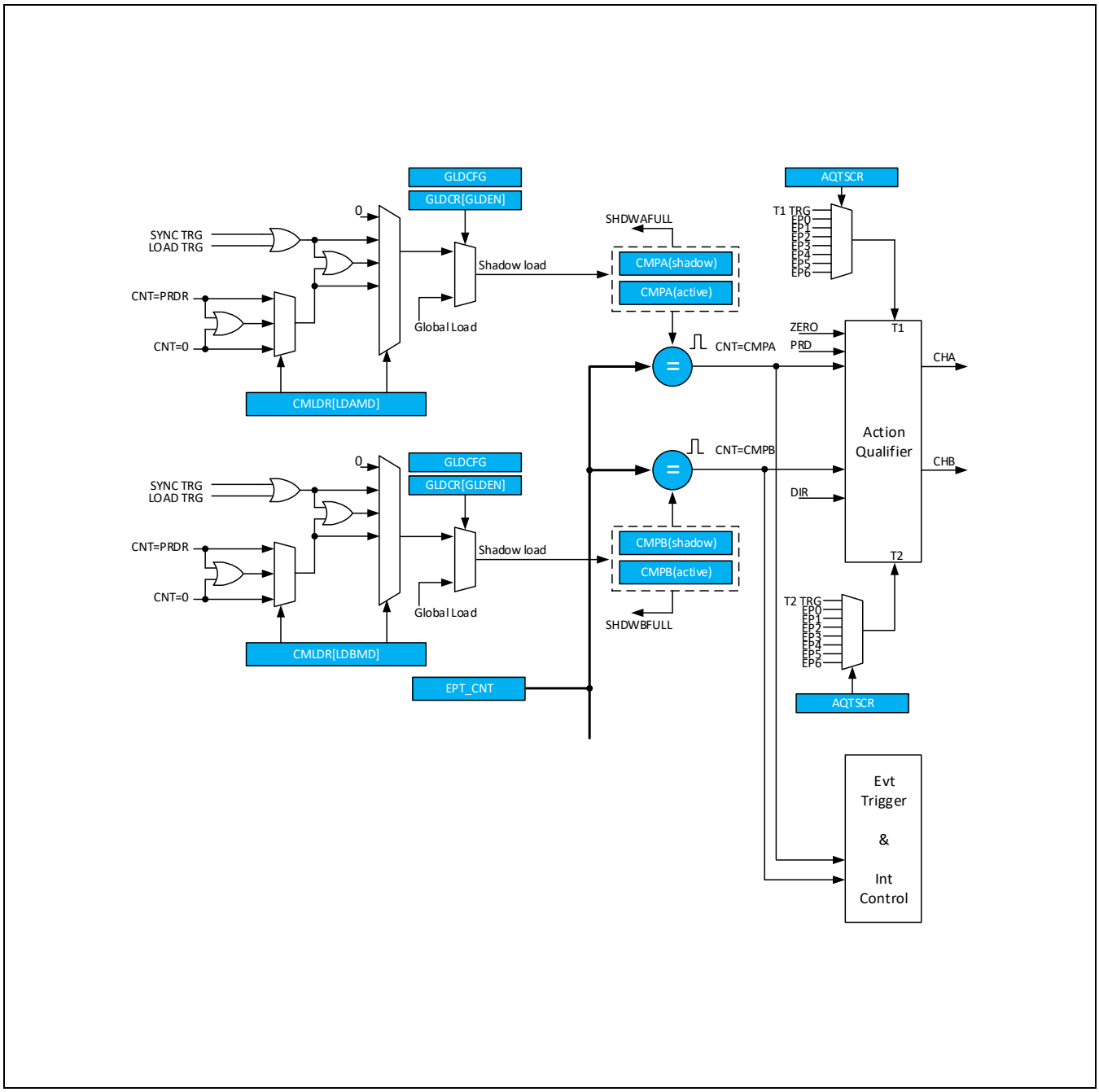


Figure 11-11 计数器值比较控制

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

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

11.3.3.2 比较值寄存器载入方式

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

- **CMPx寄存器的Shadow模式**

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

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

- **CMPx寄存器的立即加载模式**

  在立即加载模式下，对CMPx的操作直接影响活动寄存器。

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

11.3.3.3 不同计数模式的时序

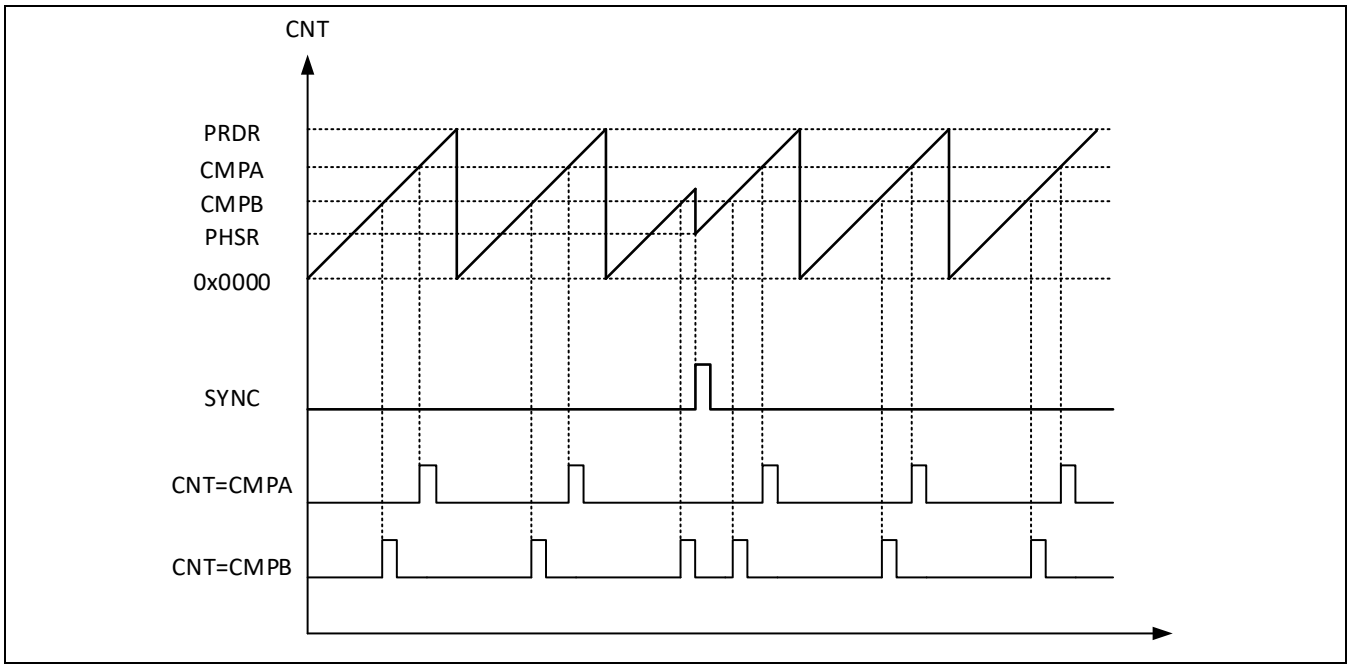


Figure 11-12 递增模式下比较事件产生时序

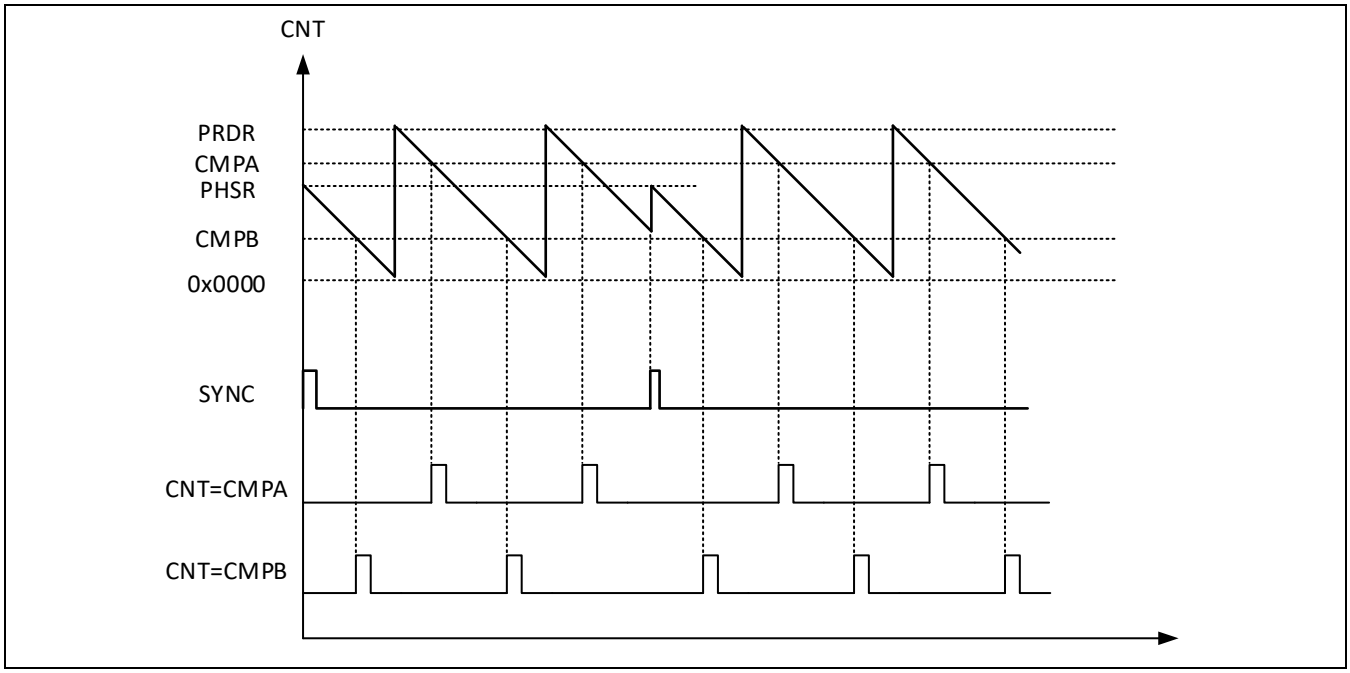


Figure 11-13 递减模式下比较事件产生时序

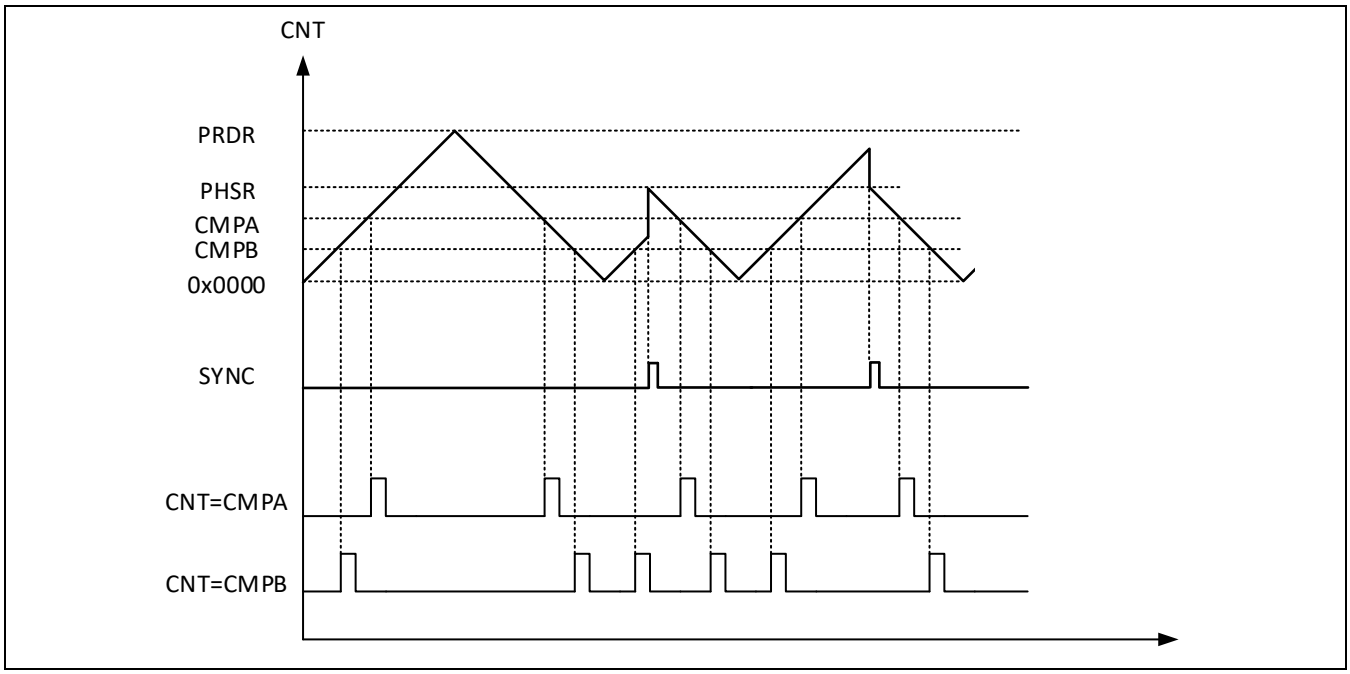


Figure 11-14 递增递减模式下比较事件产生时序

11.3.4 波形发生控制

11.3.4.1 事件驱动的波形输出

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

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

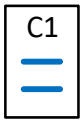
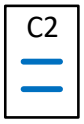
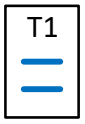
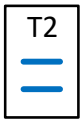
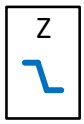
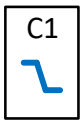
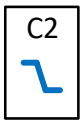
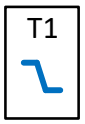
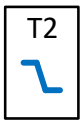
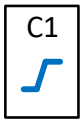
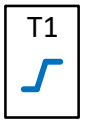
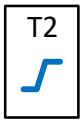
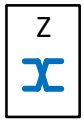
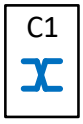
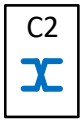
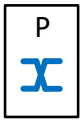
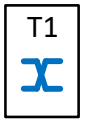
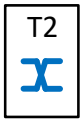
波形发生模块根据计数器的当前计数方向和发生的事件，决定PWM1和PWM2通道上的动作。所支持的的输出动作包括：

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

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

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

Table 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	C1SEL	C2SEL	PRD	T1	T2	
							没有动作
							低电平输出
							高电平输出
							翻转输出

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 递增递减模式（Up-Down-Count）下的事件优先级

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

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

Table 11-4 递增模式下的事件优先级

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

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

Table 11-5 递减模式下的事件优先级

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

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

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

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

- 计数器设置为递减模式时：
  1）如果CMPA/C2 < PRD时，C1D/CBD事件在CNT等于CMPA/CMPB时被触发；
  2）如果CMPA/C2 ≥ PRD时，C1U/CBU事件在CNT等于Period时触发；

3）C1U/CBU事件始终不会被触发。

- 计数器设置为递增递减模式时：
  1）如果CMPA/C2 < PRD且计数器计数方向为递增时，C1U/CBU事件在CNT等于C1/C2时被触发；
  2）如果CMPA/C2 < PRD且计数器计数方向为递减时，C1D/CBD事件在CNT等于C1C2时被触发；
  3）如果CMPA/C2 ≥ PRD时，C1U/C2U/C1D/C2D事件在CNT等于Period时触发。

11.3.4.3 软件强制输出

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

软件强制输出可以分为两种模式：一次性Force和连续Force。

一次性软件强制输出（One-Shot Software Forcing）

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

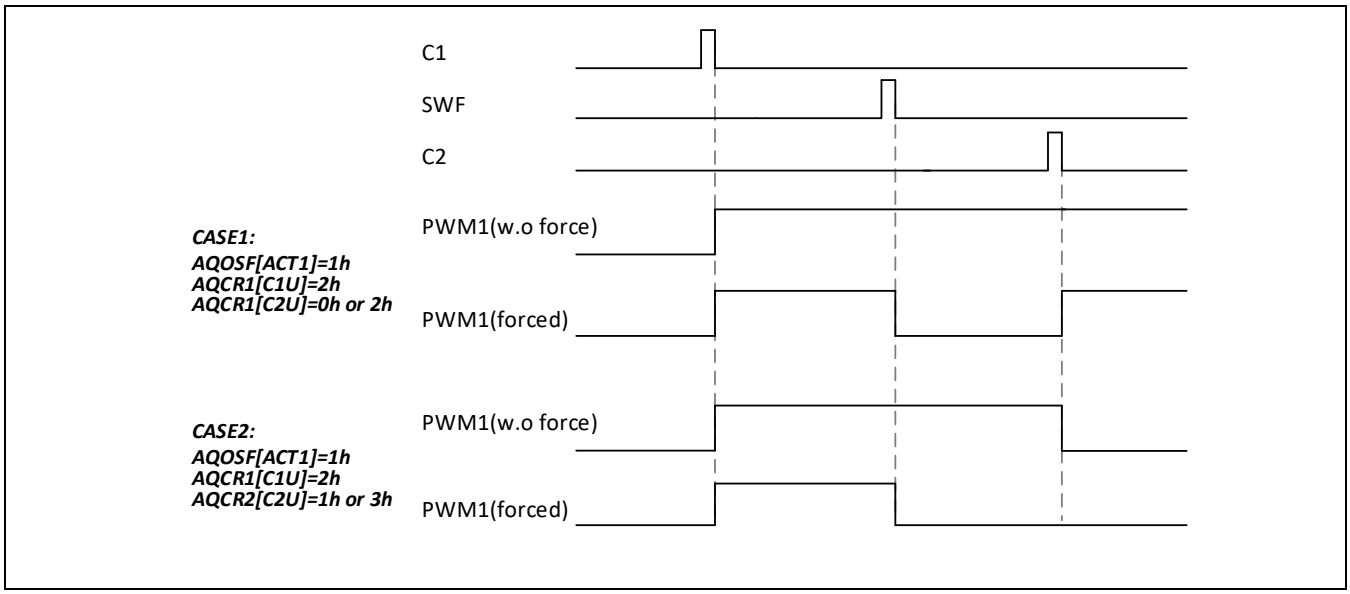


Figure 11-15 一次性软件强制输出

连续软件强制输出（Continuous Software Forcing）

在此模式下，通过寄存器的设置可以将PWM1或者PWM2的输出强制修改成软件设置电平，且该电平一直维持到软件清除强制输出。当软件清除强制输出后，通道电平将恢复到强制输出前的状态。可以通过设置寄存器GPTA_AQCSF[CSF1] 和GPTB_AQCSF[CSF2]控制位，进行强制输出设置或者清除。

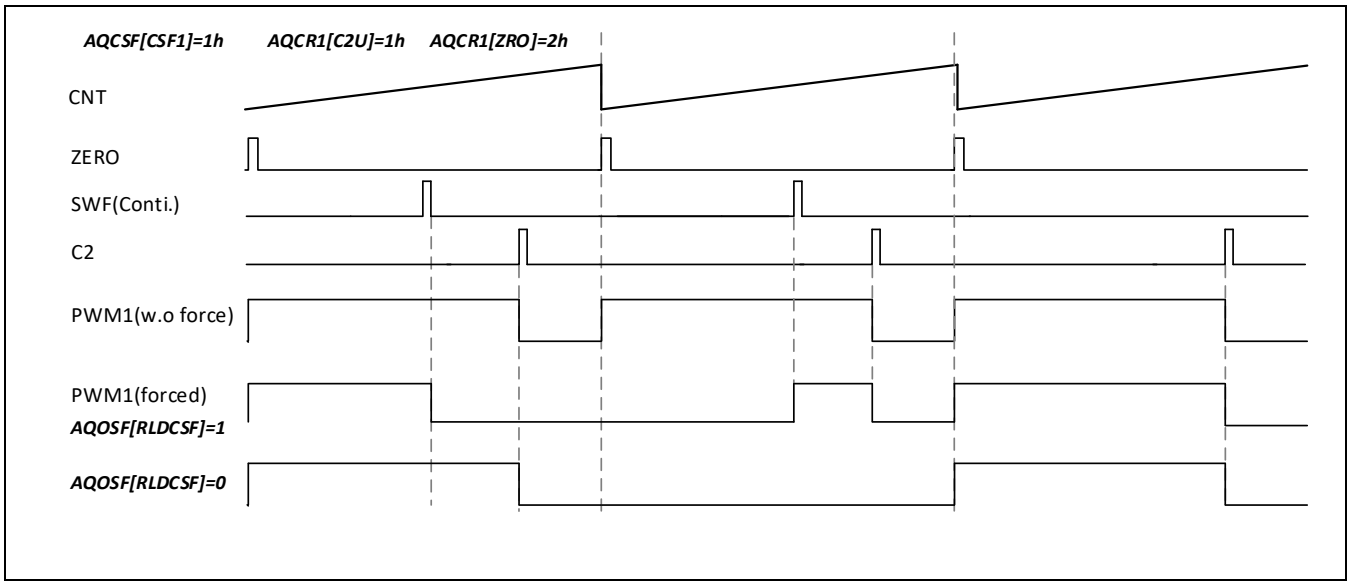


Figure 11-16 持续性软件强制输出

11.3.4.4 常见配置下的波形输出

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

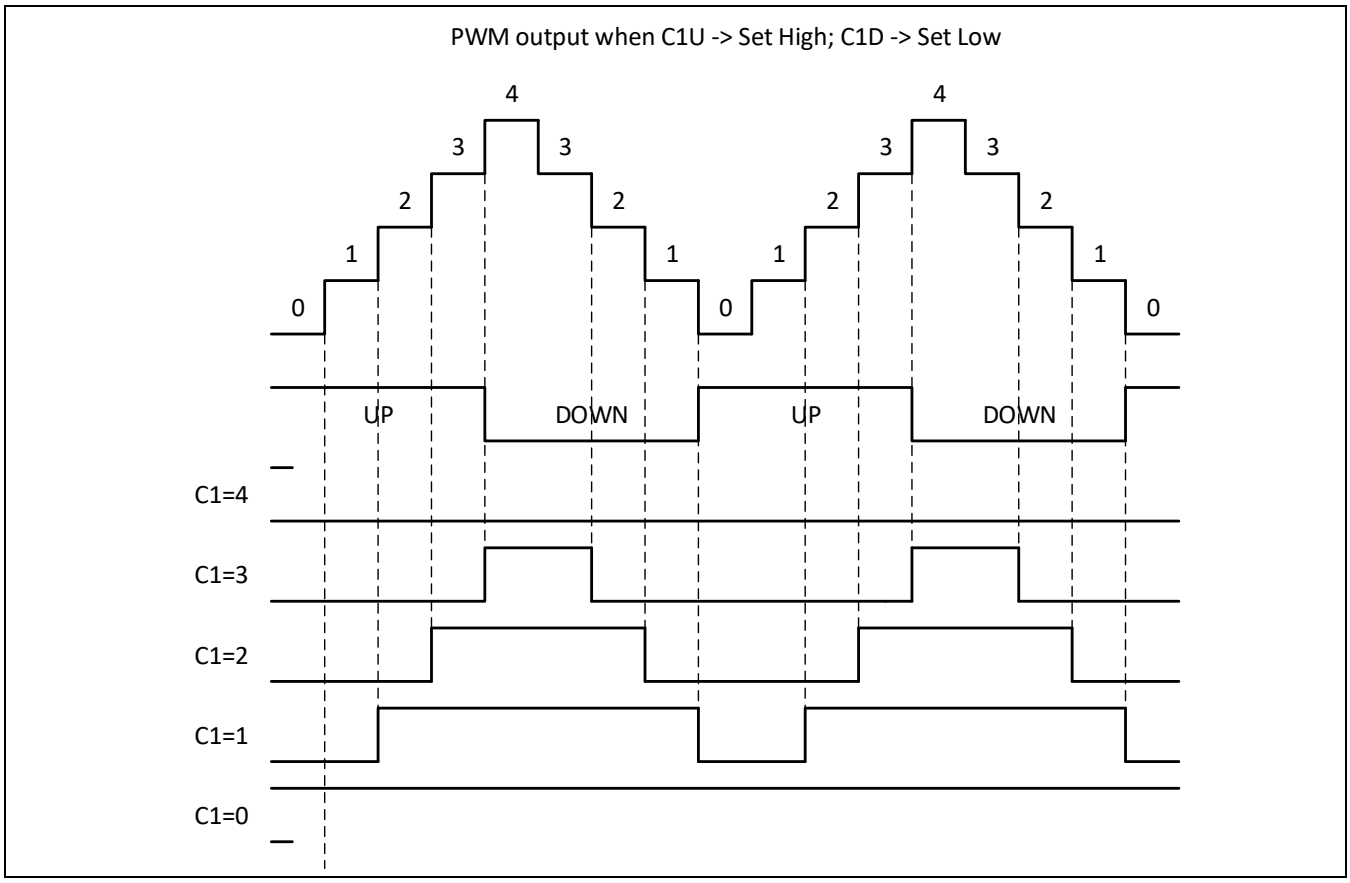


Figure 11-17 递增递减单比较值时，对称波形输出

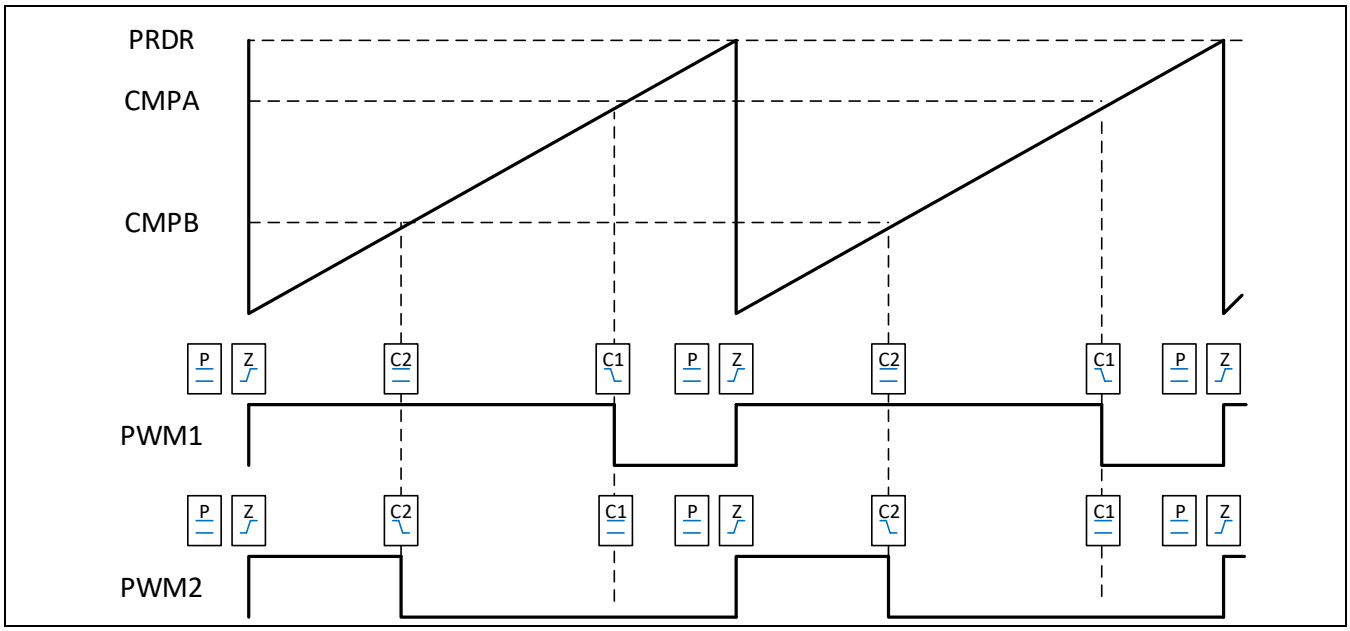


Figure 11-18 递增单沿，非对称波形输出

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

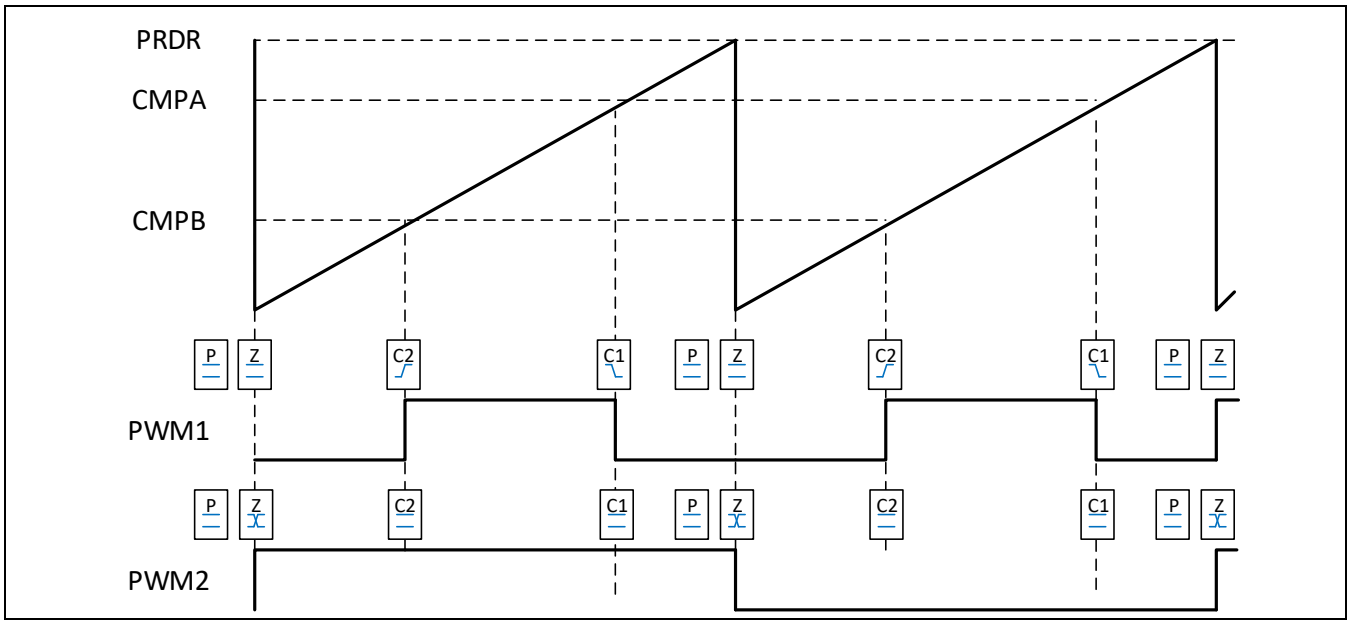


Figure 11-19 递增，脉冲定位非对称波形输出

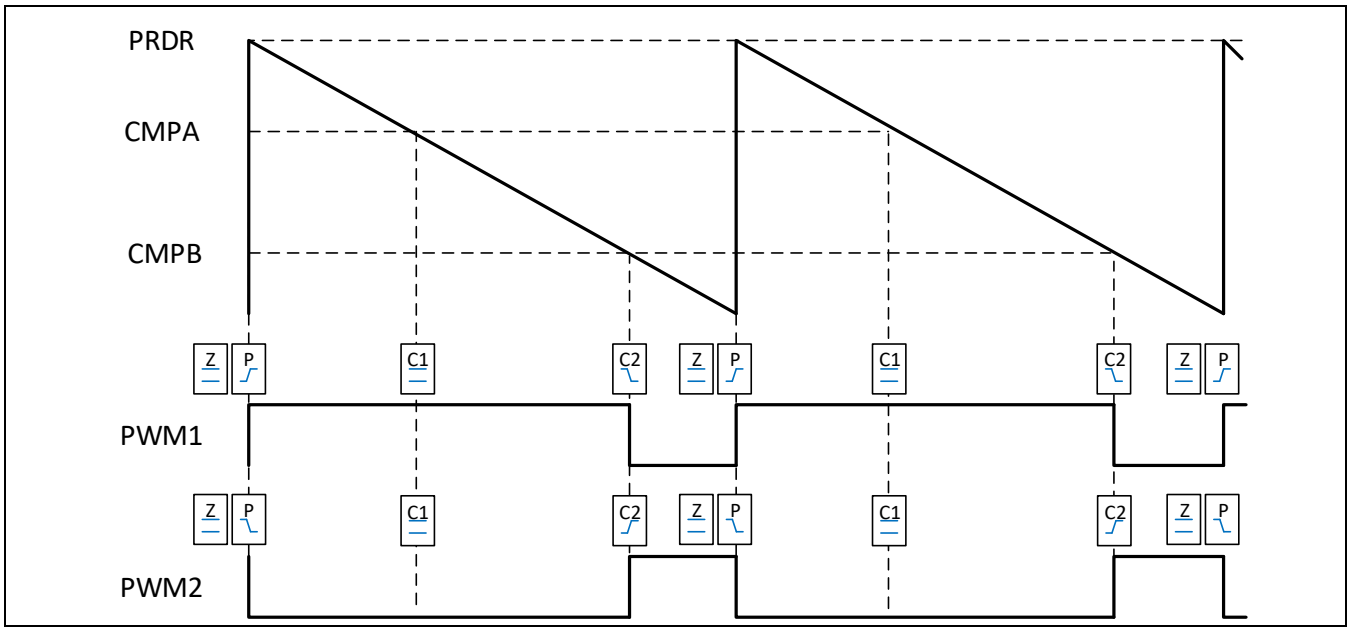


Figure 11-20 递减单沿，非对称波形输出

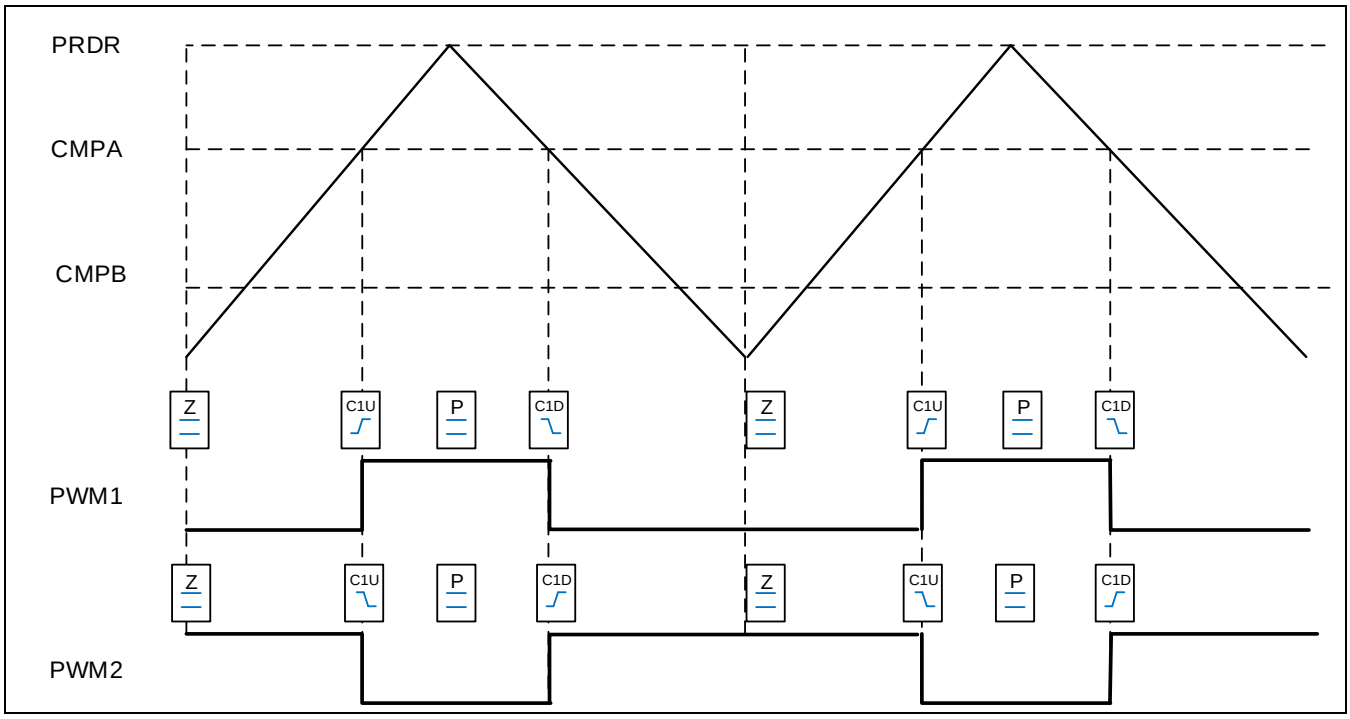


Figure 11-21 递增递减，双沿对称波形输出

11.3.5 捕捉模式

11.3.5.1 概述

捕捉模式一般用于如下几个常用的应用：

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

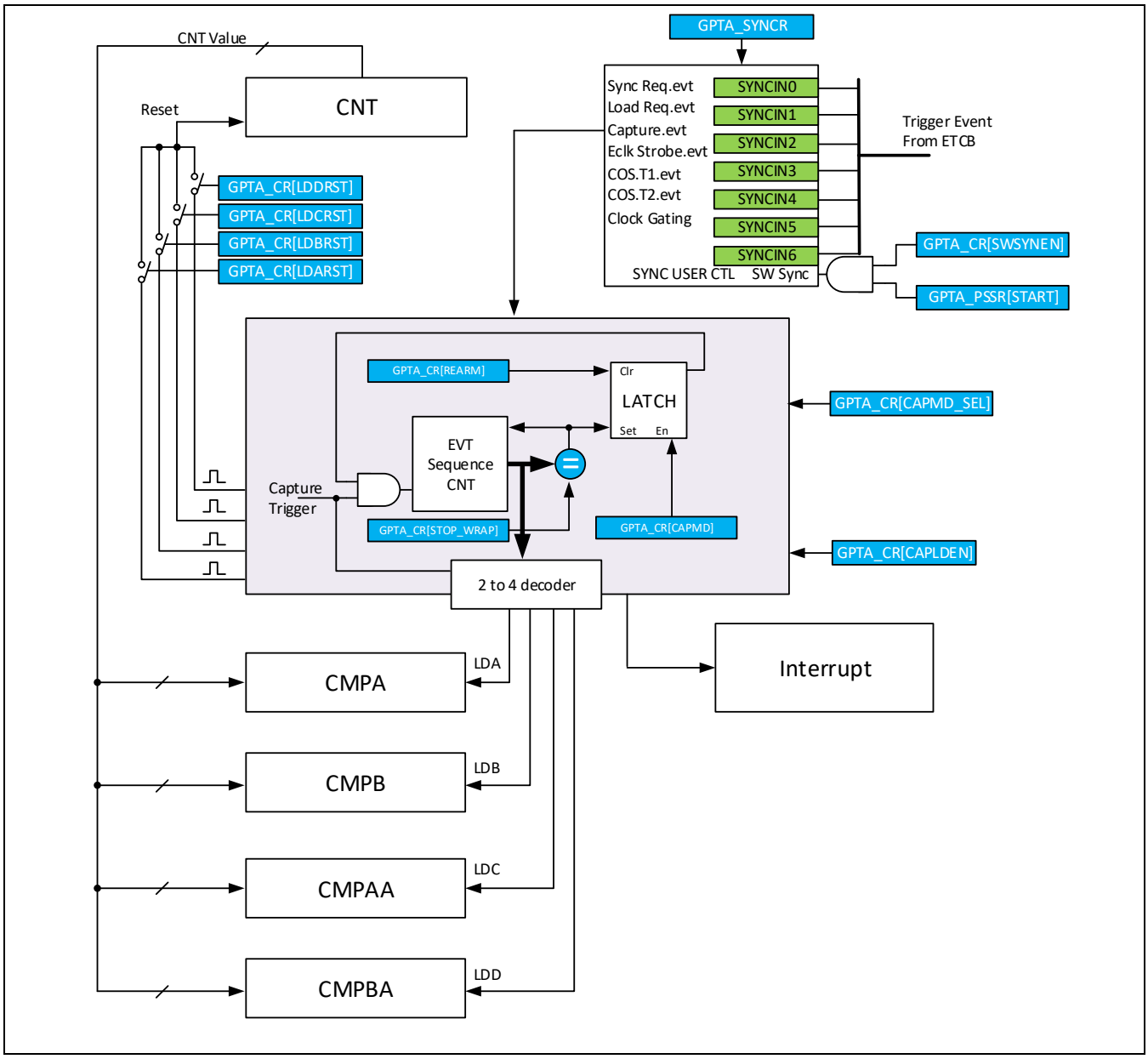


Figure 11-22 捕捉模式结构框图

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

- 支持4个捕获事件(GPTA_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(GPTA_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

当GPTA_CR[WAVE]控制位为零时，GPTA工作于捕获模式。在该模式下，捕获值将根据当前捕获事件序列计

数器值被存储到相对应的寄存器中。在捕获模式下，比较值寄存器将作为捕获值存储功能使用。捕获事件序列计数器在检测到一次捕获触发事件（SYNCIN2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GPTA_CR[STOP_WRAP]的设置时，计数器自动清零，并重新开始计数。

11.3.5.2 捕获事件计数器

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

Table 11-6 捕获存储寄存器列表

EVT CNT	Load Target	Trigger Event	Description
0	CMPA(SHD)	CAP_LD0	Current counter value is loaded into CMPA shadow, CAP_LD0 is triggered
1	CMPB(SHD)	CAP_LD1	Current counter value is loaded into CMPB shadow, CAP_LD1 is triggered
2	CMPAA	CAP_LD2	Current counter value is loaded into CMPA active, CAP_LD2 is triggered
3	CMPBA	CAP_LD3	Current counter value is loaded into CMPB active, CAP_LD3 is triggered

11.3.5.3 两种捕获模式

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

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

11.3.5.4 捕获模式下的事件

捕获模式的启动事件：

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

捕获模式的捕获事件：

捕获事件一旦发生，就会触发计数器load操作。捕获事件即SYNCIN2/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。

11.3.5.5 应用举例

下面有一些例子，说明如何使用捕捉模式。

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

One-shot模式，STOP_WRAP = 2，LDA/BRST = 1。设置TIOB上升沿为SYNCIN0输入，TIOA上升沿和下降沿都设为SYNCIN2输入。TIOB上升沿复位计数器，TIOA的下降沿触发第一次load，计数值存入CMPA中。TIOA下一个上升沿触发第二次load，计数值存入CMPB。计数器随即停止计数。此时CMPA的结果为相位差，CMPB的结果为TIOA的高电平宽度。（Figure18-22）

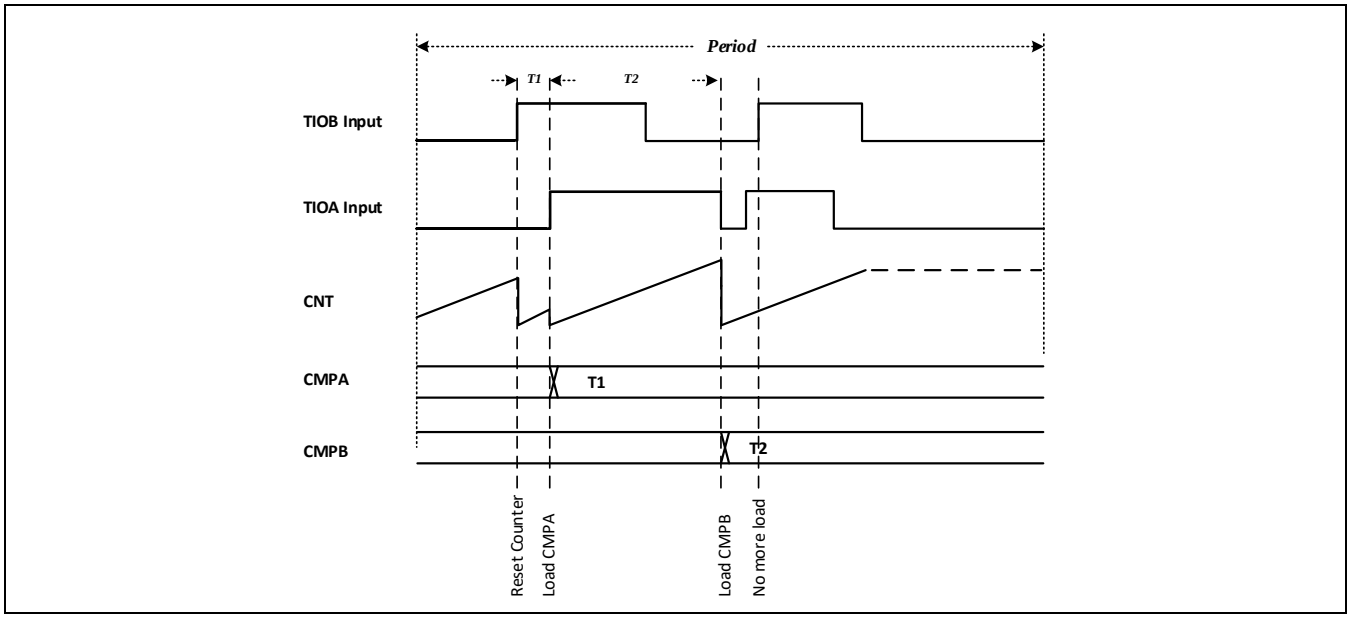


Figure 11-23 测量TIOA和TIOB相位差

- **检测TIOA上高电平脉冲宽度**

Continous 模式，STOP_WRAP = 1，LDA/BRST = 0。将TIOA设为EXIn（n<16），配置EXIn上升沿为SYNCIN0输入，同时将TIOA设为EXIm（m>16，扩展EXI），配置EXIm的下降沿为CMPA的SYNCIN2。第一个TIOA上升沿发生时，SYNC事件发生，计数器被复位。TIOA下降沿触发第一次load，计数值存入CMPA。所以，CMPA的结果为高电平宽度。（Figure18-23）

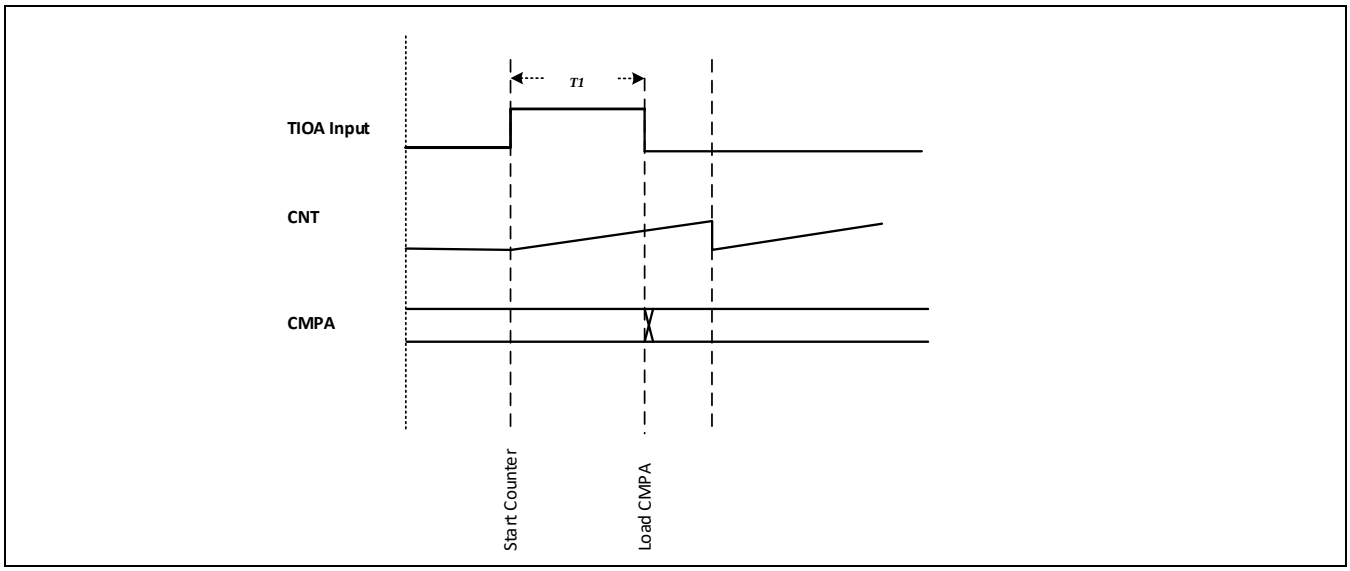


Figure 11-24 测量TIOA的脉冲宽度

11.3.6 单次触发模式

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

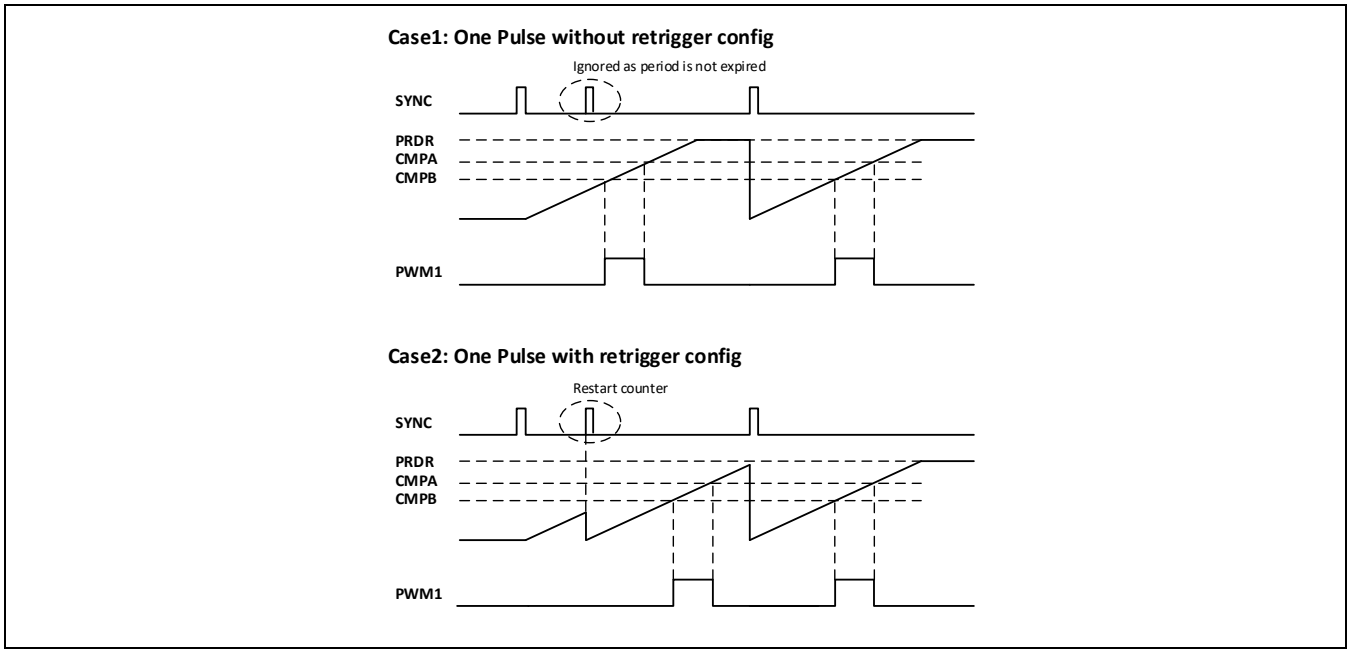


Figure 11-25 单次触发

单次触发模式工作时，缺省情况下，无论当前计数周期是否结束，当有新的触发被检测到时，计数器将开始重

新计数。在某些应用中，需要在被触发后的一个计数周期内禁止新的触发，以保证得到一个完整的周期波形输出。在这种条件下，可以通过设置触发控制寄存器GPTA_SYNCR中的OSTMD控制位，将触发模式设置为一次性触发；或者将外部触发信号通过窗口滤波器滤波，屏蔽指定时间内的触发输入。

11.3.7 同步触发（输入）

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

11.3.7.1 同步触发输入接口

GPTA支持模块间的同步触发功能，可以支持的触发功能包括如下几种：

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

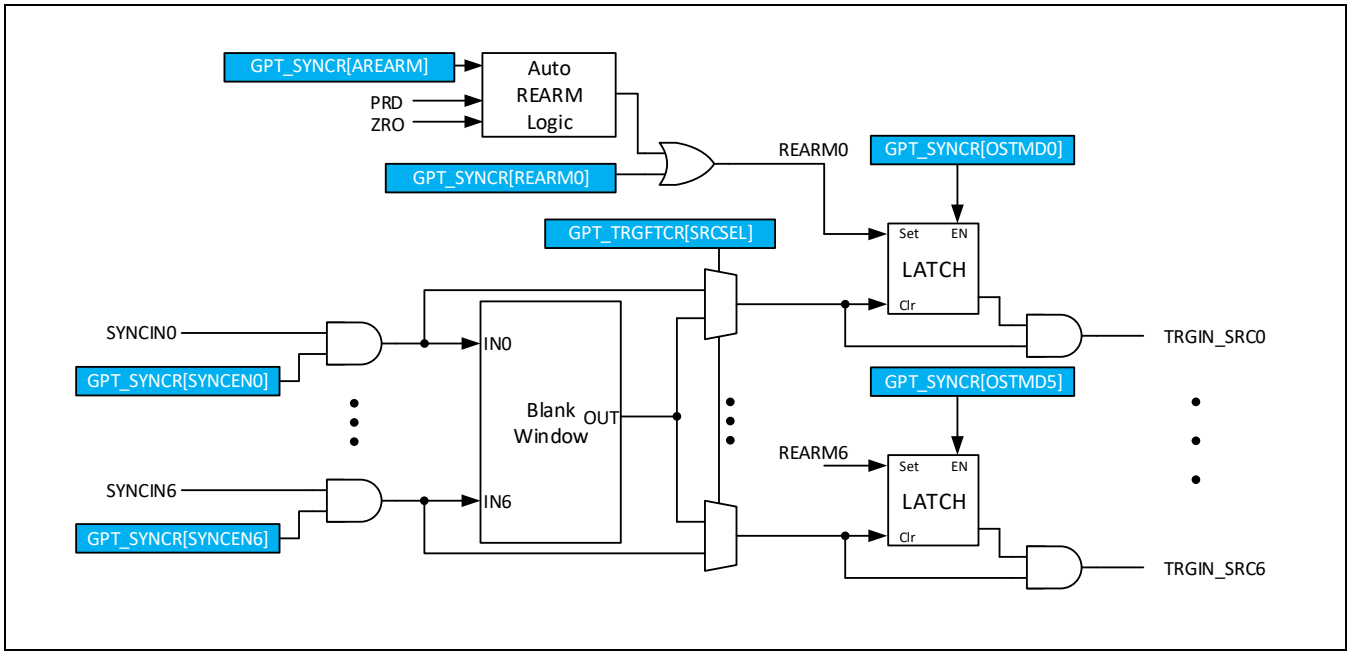


Figure 11-26 同步触发输入

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

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

重置和启动计数器（SYNCIN0）

当该端口被触发，下列动作将被同时执行：

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

寄存器的更新（SYNCIN1）

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

计数器值捕捉（SYNCIN2/SYNCIN3）

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

计数器值递增或递减一个计数值（SYNCIN4）

当该端口被触发，计数器将根据当前计数方向，自动增加或减少一个计数值。只有在GPTA_CEDR[CSS]控制位选择SYNCIN4时，该端口的触发才会被计数器检测到。

PWM输出状态改变（SYNCIN5/6）

SYNCIN 5用于产生内部T1触发事件，SYNCIN 6用于产生内部T2触发事件。

11.3.7.2 事件滤波器

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

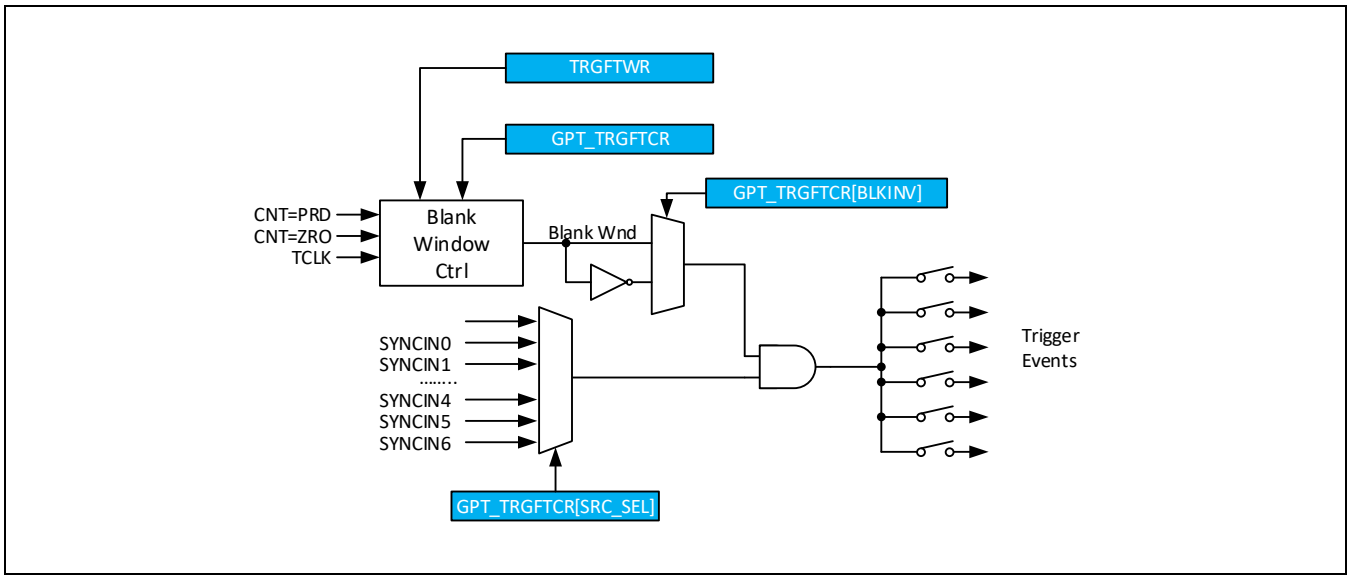


Figure 11-27 滤波器框图

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

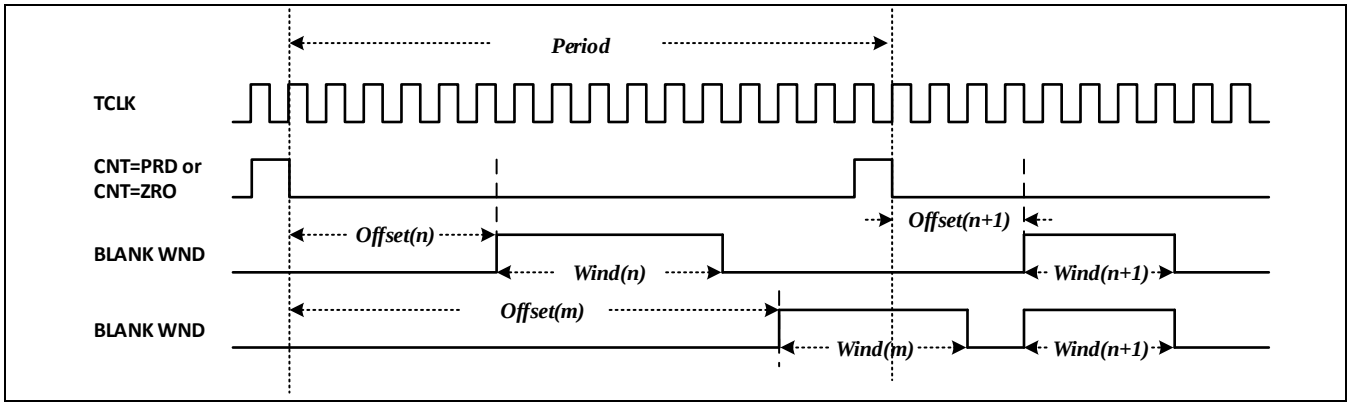


Figure 11-28 滤波器时序

11.3.8 事件触发（输出）

11.3.8.1 同步触发输出接口

事件触发输出接口支持2路事件触发输出，每个GPTA中断对应一个事件输出端口。可以通过GPTA_EVTRG控制寄存器选择GPTA中的任意一个事件作为每个中断的触发信号，事件中断触发信号可以通过TRGxOE控制位使能输出到其他外设，作为触发信号。

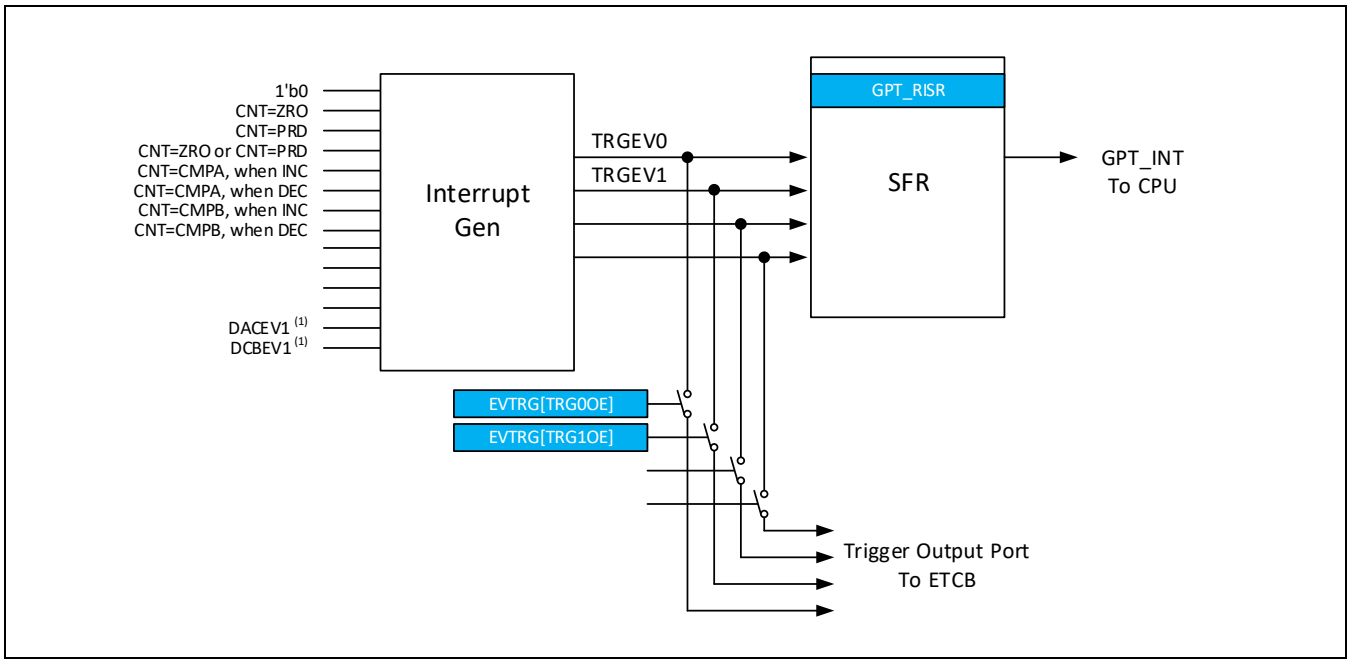


Figure 11-29 事件触发输出

11.3.8.2 事件计数和中断

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

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

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

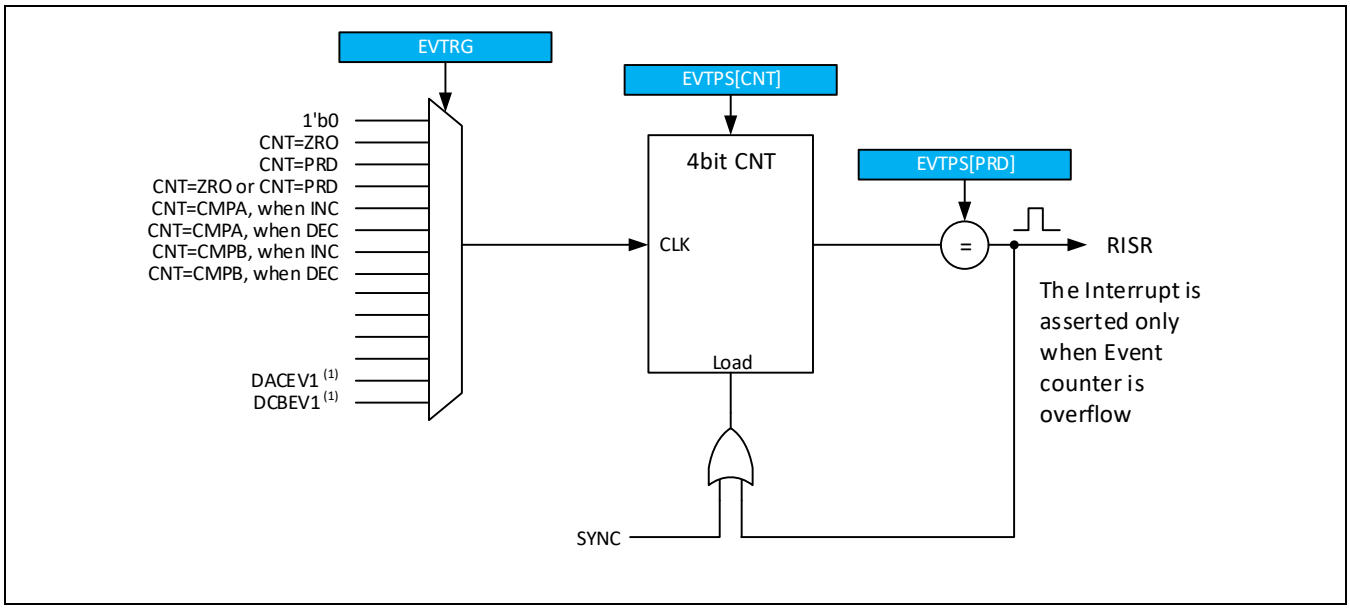


Figure 11-30 事件计数器

11.3.9 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 11-7 寄存器链接对应表

链接值	链接目标
0x0	不链接
0x1	GPTA0
0x2	GPTA1
0x3	GPTA2
0x4	GPTA3
0x5	GPTB0
0x6	GPTB1
0x7	GPTB2
0x8	GPTB3
0x9	GPTB4

0xA	GPTB5
其他值	不链接

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 = 0x1, GPTB0.REGLK.PRDR = 0x1时，当程序更新GPTB 0.PRDR时，GPTB 0，GPTA0和GPTB0的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 GPTA0: 0x40055000
Base Address of GPTA1: 0x40056000
Base Address of GPTA2: 0x40057000
Base Address of GPTA3: 0x40058000

Register	Offset	Description	Reset Value
GPTA_CEDR	0x0000	ID和时钟控制寄存器	0x28980000
GPTA_RSSR	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	0x00000000
GPTA_CRn	0x000C	控制寄存器，波形输出模式：WAVE=1	0x00000000
GPTA_SYNCR	0x0010	同步控制寄存器	0x00000000
GPTA_GLDCR	0x0014	全局载入控制寄存器	0x00000000
GPTA_GLDCFG	0x0018	全局载入配置	0x00000000
GPTA_GLDCR2	0x001C	全局载入控制寄存器2	0x00000001
GPTA_PRDR	0x0024	周期设置寄存器	0x00000000
GPTA_PHSR	0x0028	相位设置寄存器	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	0x00000090
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_AQCR2	0x004C	PWM2波形输出控制寄存器	0x00000000
GPTA_AQOSF	0x005C	一次性软件强制输出控制	0x00010000
GPTA_AQCSF	0x0060	连续软件强制输出控制	0x00000000
GPTA_TRGFTCR	0x00B8	数字比较器滤波窗控制寄存器	0x00000000
GPTA_TRGFTWR	0x00BC	数字比较器滤波窗时序寄存器	0x00000000
GPTA_EVTRG	0x00C0	事件触发选择寄存器	0x00000000
GPTA_EVPS	0x00C4	事件触发计数寄存器	0x00000000
GPTA_EVCNTINIT	0x00C8	事件触发计数器初始化值寄存器	0x00000000
GPTA_EVSWF	0x00CC	事件计数器载入控制寄存器	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_REGLK2	0x00E4	寄存器链接控制器2	0x00000000
GPTA_REGPROT	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

11.4.2 GPTA_CEDR(ID和时钟控制寄存器)

Address = Base Address+ 0x0000, Reset Value = 0x28980000

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

Name	Bit	Type	Description
IDCODE	[31:16]	R	当前GPTA模块的版本信息。
FLTCKPRS	[15:8]	RW	数字滤波器的时钟分频控制。 数字滤波器的时钟频率为PCLK/(FLTCKPRS+1)
SHDWSTP	[6]	RW	START控制位的Shadow功能使能控制。START置位不受此位控制，清除时受此位控制。当选择Shadow模式时，START控制位在周期结束时清除。 0h：Shadow模式 1h：Immediate模式
CSS	[3]	RW	计数器时钟源选择位。 0h：PCLK 1h：由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 GPTA_RSSR(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								CNTDIR	RSVD		START
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	R	R	R	R	R	R	R	R	R	R	R	RW

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

11.4.4 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。

11.4.5 GPTA_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				LDDRST	LDCRST	LDBRST	LDARST	STOP_WRAP		CAPMD	REARM	CAPMD_SEL	WAVE	PSCLD		CGFLT			CGSRC		FLTIPSCLD	BURST	CAPLDEN	RSVD		PRDLD		RSVD	SWSYNEN	CNTMD	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	RW	RW	RW	RW	RW	RW	RW	W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW	R	RW	RW	RW

Name	Bit	Type	Description
LDDRST	[27]	RW	CMPBA(Shadow)捕捉载入后，计数器值计数状态控制位。 0h：当前捕获触发后，计数器值不进行重置 1h：当前捕获触发后，计数器值进行重置
LDCRST	[26]	RW	CMPAA(Shadow)捕捉载入后，计数器值计数状态控制位。 0h：当前捕获触发后，计数器值不进行重置 1h：当前捕获触发后，计数器值进行重置
LDBRST	[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。（GPTA最大可设为4）
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 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	GPTA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通过CEDR[FLTCKPRS] 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：GPTA_CHA作为CG的输入源 1h：GPTA_CHB作为CG的输入源 其他：保留
FLTIPSCLD	[10]	RW	数字滤波器初始化控制。对该控制写’1’可以初始化数字滤波器计数器. 0h：无效 1h：执行初始化
BURST	[9]	RW	群脉冲模式。 0h：禁止群脉冲模式 1h：使能群脉冲模式
CAPLDEN	[8]	RW	CMPA和CMPB在捕捉事件触发时，载入使能控制。 0h：禁止对CMP寄存器的捕获载入 1h：使能对CMP寄存器的捕获载入 此控制位在禁止对CMP寄存器载入时，并不影响捕捉事件SYNCIN2的触发。
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]
SWSYNEN	[2]	RW	软件使能同步触发使能控制（RSSR中START控制位）。 0h：设置SW START控制只用于启动。 1h：设置SW START控制用于启动和以产生一次SYNCIN0事件，以外部触发的方式重新启动。

CNTMD	[1:0]	RW	计数模式设置。 计数模式一般只设置一次，并且在计数过程中不做改变。如果计数模式被改变，变化将发生在下一个TCLK的边沿，并且基于上一个计数器值进行递增或者递减。 00b：递增模式 01b：递减模式 10b：递增递减模式 11b：保留
注意 [1]：如果更新的周期值比更新前小，且立即更新发生时计数器已经超过更新的周期值，计数器将继续计数直到溢出，然后更新才会生效。			

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

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

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

Name	Bit	Type	Description
WAVE	[18]	RW	GPTA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通过CEDR[FLTCKPRS] 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：GPTA_CHA作为CG的输入源 1h：GPTA_CHB作为CG的输入源 其他：保留
CKS	[10]	RW	采样时钟频率控制位。此控制位决定数字滤波器的采样时钟频率。 0h：PCLK 1h：PCLK/2
BURST	[9]	RW	群脉冲模式。 0h：禁止群脉冲模式 1h：使能群脉冲模式
PHSEN	[7]	RW	PHSR使能控制位，当控制位有效时，计数器将在启动时被初始化为

			PHSR中的设置值。 0h：禁止通过PHSR初始化 1h：使能通过PHSR初始化
OPM	[6]	RW	计数器单次触发工作模式选择。 0h：连续计数工作模式 1h：单次触发工作模式 其他：保留
PRDLD	[5:4]	RW	PRDR活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值。 00b：PRDR活动寄存器更新发生在周期结束(PEND) 01b：PRDR活动寄存器更新发生在SYNCIN1被触发时 10b：PRDR活动寄存器更新发生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]	R	波形输出被停止时，输出端口的缺省状态。 0h：高阻输出 1h：低电平输出
SWSYNEN	[2]	RW	软件使能同步触发使能控制（RSSR中START控制位）。 0h：设置SW START控制只用于启动。 1h：设置SW START控制用于启动和以产生一次SYNCIN0事件，以外部触发的方式重新启动。
CNTMD	[1:0]	RW	计数模式设置。 计数模式一般只设置一次，并且在计数过程中不做改变。如果计数模式被改变，变化将发生在下一个TCLK的边沿，并且基于上一个计数器值进行递增或者递减。 00b：递增模式 01b：递减模式 10b：递增递减模式 11b：保留
注意 [1]：如果更新的周期值比更新前小，且立即更新发生时计数器已经超过更新的周期值，计数器将继续计数直到溢出，然后更新才会生效。			

11.4.7 GPTA_SYNCR(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			TXREARM0		REARM6	REARM5	REARM4	REARM3	REARM2	REARM1	REARM0	OSTMD6	OSTMD5	OSTMD4	OSTMD3	OSTMD2	OSTMD1	OSTMD0	RSVD	SYNCEN6	SYNCEN5	SYNCEN4	SYNCEN3	SYNCEN2	SYNCEN1	SYNCEN0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	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TRGSRC1控制位选择为ExtSync条件时有效。 0h：选择SYNCIN0作为TRGSRC1的ExtSync触发 1h：选择SYNCIN1作为TRGSRC1的ExtSync触发 2h：选择SYNCIN2作为TRGSRC1的ExtSync触发 3h：选择SYNCIN3作为TRGSRC1的ExtSync触发 4h：选择SYNCIN4作为TRGSRC1的ExtSync触发 5h：选择SYNCIN5作为TRGSRC1的ExtSync触发 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TRGSRC0控制位选择为ExtSync条件时有效。 0h：选择SYNCIN0作为TRGSRC0的ExtSync触发 1h：选择SYNCIN1作为TRGSRC0的ExtSync触发 2h：选择SYNCIN2作为TRGSRC0的ExtSync触发 3h：选择SYNCIN3作为TRGSRC0的ExtSync触发 4h：选择SYNCIN4作为TRGSRC0的ExtSync触发 5h：选择SYNCIN5作为TRGSRC0的ExtSync触发 6h：选择SYNCIN6作为TRGSRC0的ExtSync触发 其他：保留
TXREARM0	[23:22]	RW	Tx信号触发SYNCIN0的REARM 0：禁止硬件自动REARM 1：T1发生触发，自动REARM SYNCIN0通道 2：T2发生触发，自动REARM SYNCIN0通道 3：T1或者T2发生触发，自动REARM SYNCIN0通道
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
OSTMD6~OSTMD0	[14:8]	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。
SYNCEN6~SYNCEN0	[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 SYNCIN3：Capture触发事件 SYNCIN4：CNT增减一拍触发事件 SYNCIN5：外部COS事件（用于PWM波形输出控制） SYNCIN6：外部COS事件（用于PWM波形输出控制）

11.4.8 GPTA_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	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
GLDCNT	[12:10]	RW	全局载入事件计数器。 计数器值表示当前已发生多少次事件触发。
GLDPRD	[9:7]	RW	全局载入触发周期选择。 可以选择N次触发条件满足后，才进行一次全局载入。 000b：Disable Counter（立即触发） 001b：第2次条件满足时触发 010b：第3次条件满足时触发 011b：第4次条件满足时触发 100b：第5次条件满足时触发 101b：第6次条件满足时触发 110b：第7次条件满足时触发 111b：第8次条件满足时触发
OSTMD	[5]	RW	One Shot 载入模式使能控制位 0h：禁止One Shot模式，只要条件满足，Active寄存器都会从Shadow寄存器载入 1h：使能One Shot模式，只有在GLDCR2[OSREARM]写入‘1’后，才会进行一次载入。一旦载入被触发，需要再次对GLDCR2[OSREARM]写入‘1’，才能允许下一次载入触发。
GLDMD	[4:1]	RW	全局载入触发事件选择。 0h：CNT = ZRO 1h：CNT = PRD 2h：CNT = ZRO or CNT = PRD 3h：CNT = ZRO or 外部LOAD触发或SYNC触发 4h：CNT = PRD or 外部LOAD触发或SYNC触发 5h：CNT = ZRO or CNT = PRD or 外部LOAD触发或SYNC触发 Others：Reserved Fh：在GLDCR2[GFRCLD]写入‘1’时 [1]
GLDEN	[0]	RW	全局的Shadow到Active寄存器载入控制。 0：使用独立的单个配置（在各个寄存器中LDMD控制位分别指派的载入控制） 1：使用GLDMD中的设置，其他设置被屏蔽
注意 [1]：如果更新的周期值比更新前小，且立即更新发生时计数器已经超过更新的周期值，计数器将继续计数直			

到溢出，然后更新才会生效。类似，如果更新的比较值比更新前小，且立即更新发生时计数器已经超过更新的比较值，本周期将不会发生match事件。

11.4.9 GPTA_GLDCFG(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																			AQCSF	RSVD		AQCR2	AQCR1	RSVD						CMPB	CMPA	PRDR
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	RW	RW	R	R	R	R	R	RW	RW	RW

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

11.4.10 GPTA_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，读取时总是返回‘0’ 1：重置ONE SHOT模式。ONE SHOT模式下，一次触发后，需要重置模式才允许再次触发

11.4.11 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。

11.4.12 GPTA_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GPT_CR[PHSEN] = 0时，同步事件不会触发PHSR载入到CNT中，当GPT_CR[PHSEN] = 1时，同步事件发生会触发PHSR载入到CNT中。

11.4.13 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。

11.4.14 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。

11.4.15 GPTA_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	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。			

11.4.16 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。

11.4.17 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																								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 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	CBSEL	C2SEL	CASEL	T2D		T2U		T1D		T1U		C2D	C2BD	C2U		C1D		C1U		PRD		ZRO	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
C2SEL	[23]	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。
CBSEL	[22]	RW	
C2SEL	[21]	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。
CASEL	[20]	RW	
T2D	[19:18]	RW	当T2事件发生，且此时计数方向为递减时，在通道PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T2U	[17:16]	RW	当T2事件发生，且此时计数方向为递增时，在通道PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1D	[15:14]	RW	当T1事件发生，且此时计数方向为递减时，在通道PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1U	[13:12]	RW	当T1事件发生，且此时计数方向为递增时，在通道PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平）

			3h：反向（翻转）
C2D	[11]	RW	当CNT值等于C2，且此时计数方向为递减时，在通道PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C2BD	[10]	RW	
C2U	[9:8]	RW	当CNT值等于C2，且此时计数方向为递增时，在通道PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1D	[7:6]	RW	当CNT值等于C1，且此时计数方向为递减时，在通道PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1U	[5:4]	RW	当CNT值等于C1，且此时计数方向为递增时，在通道PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
PRD	[3:2]	RW	当CNT值等于PRDR时，在通道PWM1上做出的波形输出动作定义。 在递增递减模式时，当计数器值等于PRDR时，计数方向为递减模式 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
ZRO	[1:0]	RW	当CNT值等于零时，在通道PWM1上做出的波形输出动作定义。 在递增递减模式时，当计数器值等于零时，计数方向为递增模式 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）

11.4.19 GPTA_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		T2D		T2U		T1D		T1U		CBD		C2U		C1D		C1U		PRD		ZRO	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
C2SEL	[23:22]	RW	C2比较值数据源选择。 0h：CMPA寄存器作为C2的数据源。 1h：CMPB寄存器作为C2的数据源。 其他：保留。
C1SEL	[21:20]	RW	C1比较值的数据源选择。 0h：CMPA寄存器作为C1的数据源。 1h：CMPB寄存器作为C1的数据源。 其他：保留。
T2D	[19:18]	RW	当T2事件发生，且此时计数方向为递减时，在通道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）
T2U	[17:16]	RW	当T2事件发生，且此时计数方向为递增时，在通道PWM2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1D	[15:14]	RW	当T1事件发生，且此时计数方向为递减时，在通道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）
T1U	[13:12]	RW	当T1事件发生，且此时计数方向为递增时，在通道PWM2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
CBD	[11:10]	RW	当CNT值等于C2，且此时计数方向为递减时，在通道PWM2上做出的

			波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C2U	[9:8]	RW	当CNT值等于C2，且此时计数方向为递增时，在通道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）
C1D	[7:6]	RW	当CNT值等于C1，且此时计数方向为递减时，在通道PWM2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1U	[5:4]	RW	当CNT值等于C1，且此时计数方向为递增时，在通道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）
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：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
ZRO	[1:0]	RW	当CNT值等于零时，在通道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：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）

11.4.20 GPTA_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2上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
OSTSF2	[4]	W	在通道2上产生一次性软件强制输出。 0h：对当前位写‘0’无效 1h：产生一次性软件强制输出，此输出状态保持，直到有其他改变通道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1上做出的波形输出动作定义。 0h：保持原来的输出（不动作） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
OSTSF1	[0]	W	在通道1上产生一次性软件强制输出。 0h：对当前位写‘0’无效 1h：产生一次性软件强制输出，此输出状态保持，直到有其他改变通道1输出状态的触发事件发生。

11.4.21 GPTA_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2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1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 GPTA_TRGFTCR(数字比较器滤波窗控制寄存器)

Address = Base Address+ 0x00B8, Reset Value = 0x00000000

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

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

11.4.23 GPTA_TRGFTWR(数字比较器滤波窗时序寄存器)

Address = Base Address+ 0x00BC, Reset Value = 0x00000000

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

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

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

Name	Bit	Type	Description
CNT1INITFRC	[25]	R	TRGEV1CNT软件触发更新 0h：无效 1h：EVCNT1INIT内容更新到EVCNT1中
CNT0INITFRC	[24]	R	TRGEV0CNT软件触发更新 0h：无效 1h：EVCNT0INIT内容更新到EVCNT0中
TRG1OE	[21]	RW	外部触发端口TRGOUT1使能 0h：禁止触发输出 1h：允许触发输出
TRG0OE	[20]	RW	外部触发端口TRGOUT0使能 0h：禁止触发输出 1h：允许触发输出
CNT1INITEN	[17]	RW	TRGEV1CNT寄存器更新模式控制 0h：无效 1h：TRGEV1CNT在发生LOAD事件触发时，或者EV1CNTINITFRC控制位软件写入‘1’时，EV1CNTINIT的内容更新到EV1CNT中。
CNT0INITEN	[16]	RW	TRGEV0CNT寄存器更新模式控制 0h：无效 1h：TRGEV0CNT在发生LOAD事件触发时，或者EV0CNTINITFRC控制位软件写入‘1’时，EV0CNTINIT的内容更新到EV0CNT中。
TRGSEL1	[7:4]	RW	TRGEV1事件的触发源选择。 0000：禁止TRGOUT触发输出 0001：当 CNT = ZRO 产生TRGEV事件 0010：当 CNT = PRD 产生TRGEV事件 0011：当 CNT = ZRO or CNT = PRD 产生TRGEV事件 0100：当 CNT = CMPA 且计数方向为递增时，产生TRGEV事件 0101：当 CNT = CMPA 且计数方向为递减时，产生TRGEV事件 0110：当 CNT = CMPB 且计数方向为递增时，产生TRGEV事件 0111：当 CNT = CMPB 且计数方向为递减时，产生TRGEV事件 1100：ExtSync通道

TRGSEL0	[3:0]	RW	TRGEV0事件的触发源选择。 0000：禁止TRGOUT触发输出 0001：当 CNT = ZRO 产生TRGEV事件 0010：当 CNT = PRD 产生TRGEV事件 0011：当 CNT = ZRO or CNT = PRD 产生TRGEV事件 0100：当 CNT = CMPA 且计数方向为递增时，产生TRGEV事件 0101：当 CNT = CMPA 且计数方向为递减时，产生TRGEV事件 0110：当 CNT = CMPB 且计数方向为递增时，产生TRGEV事件 0111：当 CNT = CMPB 且计数方向为递减时，产生TRGEV事件 1100：ExtSync通道

11.4.25 GPTA_EVPS(事件触发计数寄存器)

Address = Base Address+ 0x00C4, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								TRGEV1CNT				TRGEV0CNT				RSVD								TRGEV1PRD				TRGEV0PRD			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

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

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

Address = Base Address+ 0x00C8, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																								CNT1INIT				CNT0INIT			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CNT1INIT	[7:4]	RW	TRGEV1CNT计数器的初始化值设置。 当EVTRG[CNT1INITEN]控制位有效时，CNT1INIT的值将在触发条件满足时(LOAD事件)，或EVTRG[CNT1INITFRC]软件置位时，被载入到TRGEV1CNT寄存器中。
CNT0INIT	[3:0]	RW	TRGEV0CNT计数器的初始化值设置。 当EVTRG[CNT0INITEN]控制位有效时，CNT0INIT的值将在触发条件满足时(LOAD事件)，或EVTRG[CNT0INITFRC]软件置位时，被载入到TRGEV0CNT寄存器中。

11.4.27 GPTA_EVSWF(事件计数器载入控制寄存器)

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

11.4.28 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				CBD	CBU	CAD	CAU	CAP_LD3	CAP_LD2	CAP_LD1	CAP_LD0	RSVD		TRGEV1	TRGEV0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W

Name	Bit	Type	Description
PEND	[16]	R	周期结束中断请求原始标志状态
CBD	[11]	R	递减阶段CNT = CMPB中断请求原始标志状态
CBU	[10]	R	递增阶段CNT = CMPB中断请求原始标志状态
CAD	[9]	R	递减阶段CNT = CMPA中断请求原始标志状态
CAU	[8]	R	递增阶段CNT = CMPA中断请求原始标志状态
CAP_LD3	[7]	R	Capture Load to CMPBA Shadow中断请求原始标志状态
CAP_LD2	[6]	R	Capture Load to CMPAA Shadow中断请求原始标志状态
CAP_LD1	[5]	R	Capture Load to CMPB Shadow中断请求原始标志状态
CAP_LD0	[4]	R	Capture Load to CMPA Shadow中断请求原始标志状态
TRGEV1	[1]	W	TRGEV1中断请求原始标志状态
TRGEV0	[0]	W	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

11.4.29 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				CBD	CBU	CAD	CAU	CAP_LD3	CAP_LD2	CAP_LD1	CAP_LD0	RSVD		TRGEV1	TRGEV0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W

Name	Bit	Type	Description
PEND	[16]	R	周期结束中断请求原始标志状态
CBD	[11]	R	递减阶段CNT = CMPB中断请求原始标志状态
CBU	[10]	R	递增阶段CNT = CMPB中断请求原始标志状态
CAD	[9]	R	递减阶段CNT = CMPA中断请求原始标志状态
CAU	[8]	R	递增阶段CNT = CMPA中断请求原始标志状态
CAP_LD3	[7]	R	Capture Load to CMPBA Shadow中断请求原始标志状态
CAP_LD2	[6]	R	Capture Load to CMPAA Shadow中断请求原始标志状态
CAP_LD1	[5]	R	Capture Load to CMPB Shadow中断请求原始标志状态
CAP_LD0	[4]	R	Capture Load to CMPA Shadow中断请求原始标志状态
TRGEV1	[1]	W	TRGEV1中断请求标志状态
TRGEV0	[0]	W	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

11.4.30 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				CBD	CBU	CAD	CAU	CAP_LD3	CAP_LD2	CAP_LD1	CAP_LD0	RSVD		TRGEV1	TRGEV0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW

Name	Bit	Type	Description
PEND	[16]	RW	周期结束中断请求原始标志状态
CBD	[11]	RW	递减阶段CNT = CMPB中断请求原始标志状态
CBU	[10]	RW	递增阶段CNT = CMPB中断请求原始标志状态
CAD	[9]	RW	递减阶段CNT = CMPA中断请求原始标志状态
CAU	[8]	RW	递增阶段CNT = CMPA中断请求原始标志状态
CAP_LD3	[7]	RW	Capture Load to CMPBA Shadow中断请求原始标志状态
CAP_LD2	[6]	RW	Capture Load to CMPAA Shadow中断请求原始标志状态
CAP_LD1	[5]	RW	Capture Load to CMPB Shadow中断请求原始标志状态
CAP_LD0	[4]	RW	Capture Load to CMPA Shadow中断请求原始标志状态
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			

11.4.31 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				CBD	CBU	CAD	CAU	CAP_LD3	CAP_LD2	CAP_LD1	CAP_LD0	RSVD		TRGEV1	TRGEV0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	R	R	R	R	W	W	W	W	W	W	W	W	R	R	W	W

Name	Bit	Type	Description
PEND	[16]	W	周期结束中断请求原始标志状态
CBD	[11]	W	递减阶段CNT = CMPB中断请求原始标志状态
CBU	[10]	W	递增阶段CNT = CMPB中断请求原始标志状态
CAD	[9]	W	递减阶段CNT = CMPA中断请求原始标志状态
CAU	[8]	W	递增阶段CNT = CMPA中断请求原始标志状态
CAP_LD3	[7]	W	Capture Load to CMPBA Shadow中断请求原始标志状态
CAP_LD2	[6]	W	Capture Load to CMPAA Shadow中断请求原始标志状态
CAP_LD1	[5]	W	Capture Load to CMPB Shadow中断请求原始标志状态
CAP_LD0	[4]	W	Capture Load to CMPA Shadow中断请求原始标志状态
TRGEV1	[1]	W	清除TRGEV1原始中断状态位
TRGEV0	[0]	W	清除TRGEV0原始中断状态位
中断清除控制位。 对该寄存器写‘0’时，无效；对该寄存器写‘1’时，清除相应中断标志位 读取时，总是返回‘0’			

11.4.32 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				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:GPTA1
3H:GPTA2
4H:GPTA3
5H:GPTB0
6H:GPTB1
7H:GPTB2
8H:GPTB3
9H:GPTB4
AH:GPTB5

11.4.33 GPTA_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:GPTA1 3H:GPTA2 4H:GPTA3 5H:GPTB0 6H:GPTB1 7H:GPTB2 8H:GPTB3 9H:GPTB4 AH:GPTB5			

11.4.34 GPTA_REGPROT(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
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
WRKEY	[31:16]	R	写入保护KEY 当对PROTKEY进行写操作时，必须将KEY设置为A55Ah，否则写入无效
PROTKEY	[15:0]	RW	写保护使能控制。 当此寄存器的值不等于C73Ah时，具有写保护功能的寄存器将禁止写入操作。只有解锁后，具有写保护功能的寄存器才允许写操作。对于具有写保护寄存器的写操作完成后，写保护寄存器会自动清除（自动保护使能），所以每次对任意具有写保护功能的寄存器写入之前，都必须进行解锁操作

11.4.35 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	[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。

11.4.36 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 增强型通用定时器B (GPTB)

12.1 概述

通用定时器(General Purpose Timer)作为 MCU 的关键外设，可以提供多种时基计数和波形产生功能。通过灵活的 PWM 输出，可以适用于各种复杂多变的应用。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。

12.1.1 主要特性

- 16 位可复位计数器
- 可编程计数器计数方式
  - 递增计数(Up-counting)
  - 递减计数(Down-counting)
  - 递增递减计数(Up-down-counting)
- 两路波形产生控制单元，支持双路独立输出：
  - 两路独立的 PWM 输出，单边沿工作
  - 两路独立的 PWM 输出，双边沿对称工作
  - 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。

12.1.2 管脚描述

下表列出了不同模式下的管脚定义。

Table 12-1 不同模式下的管脚描述

管脚名称	突发计数模式	波形发生器： 单波形输出模式	波形发生器： 双波形输出模式
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

12.2 功能描述

12.2.1 模块框图

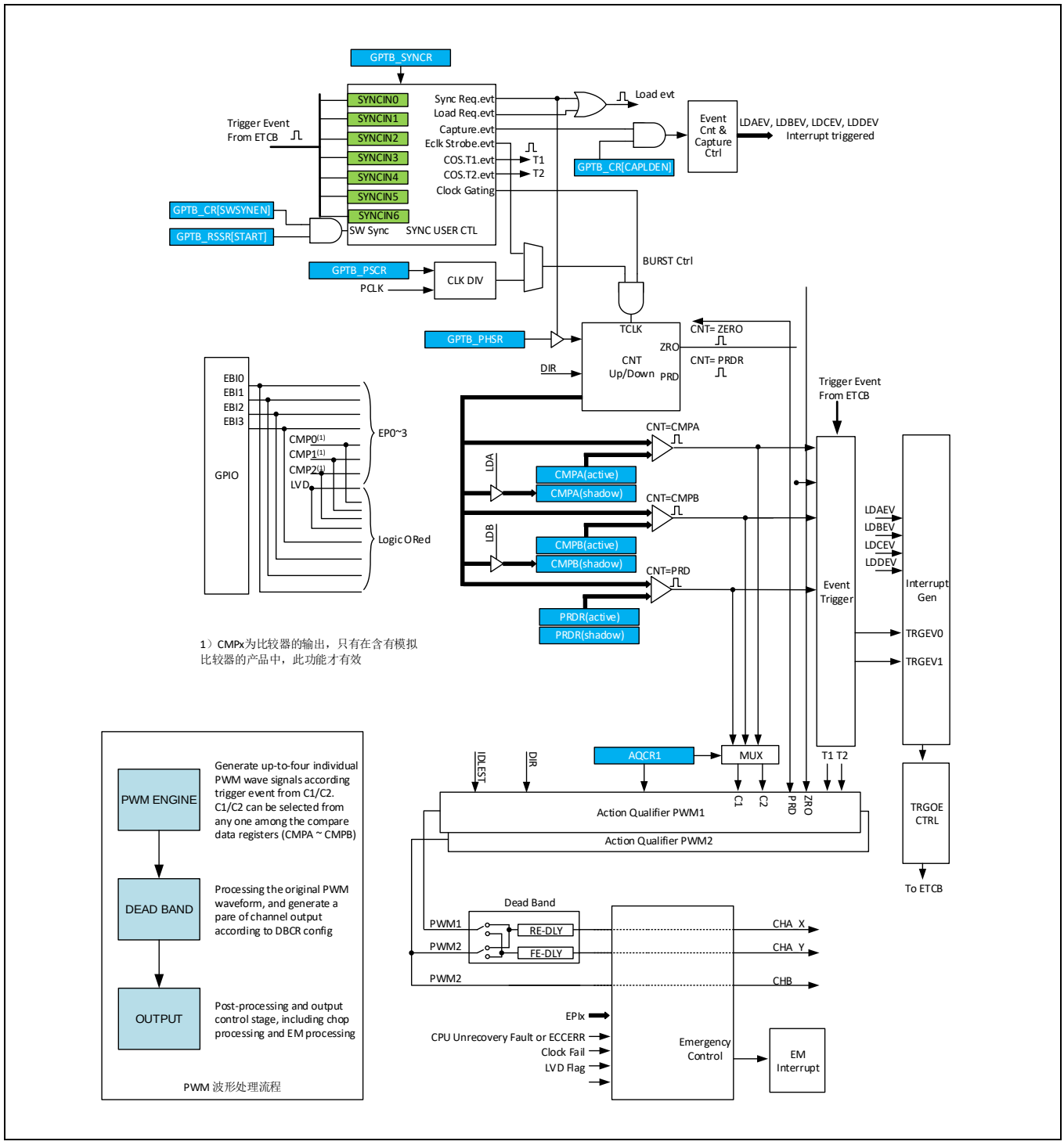


Figure 12-1 模块结构示意图

12.3 基本功能描述

一个完整的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。

12.3.1 时钟源

12.3.1.1 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，触发计数器递增或递减，外部计数时钟频率必须低于1/2倍PCLK，以保证被PCLK同步，例如：当PCLK频率为4MHz时，则最高外部输入时钟为2MHz。在外部时钟超过PCLK频率限制时，可以通过异步预分频对输入时钟进行预先分频，保证分频后的时钟频率满足被PCLK同步的条件。

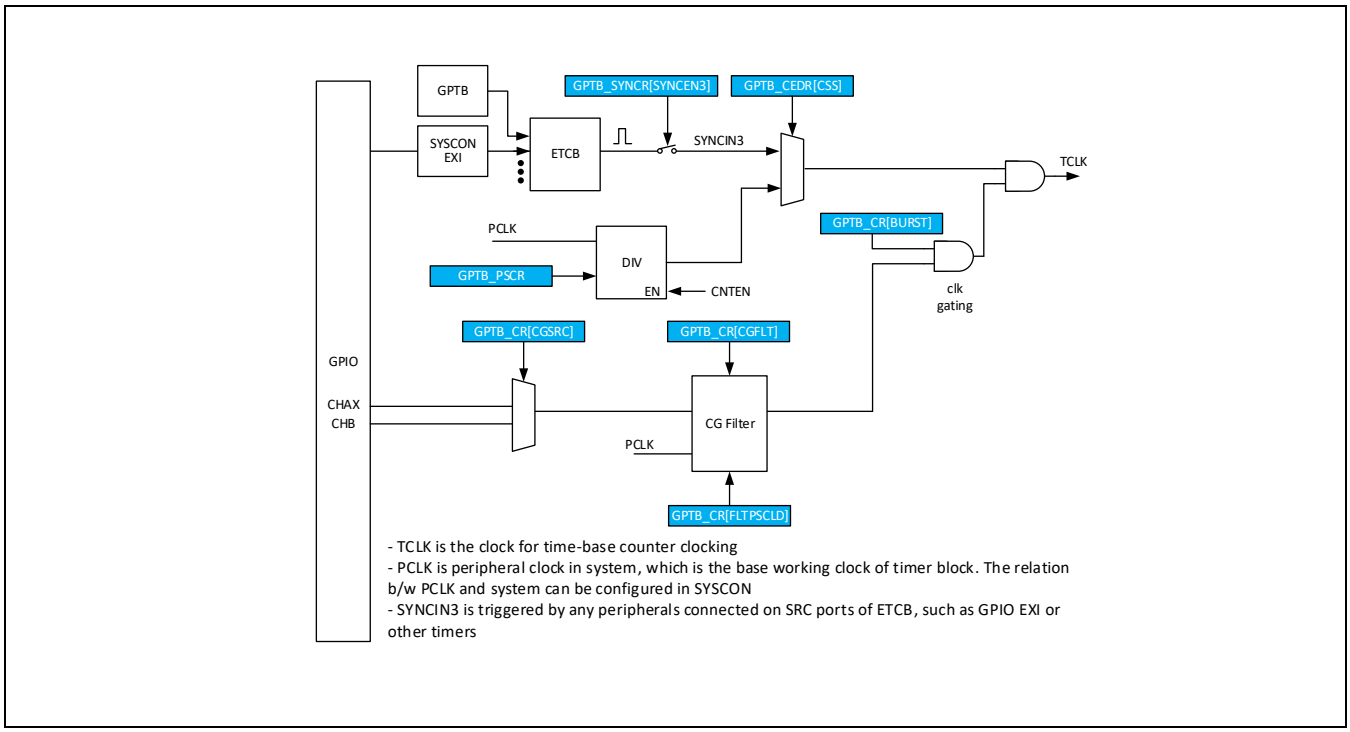


Figure 12-2 时钟控制模块

12.3.1.2 外部时钟

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

12.3.1.3 内部时钟

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

12.3.1.4 群脉冲时钟

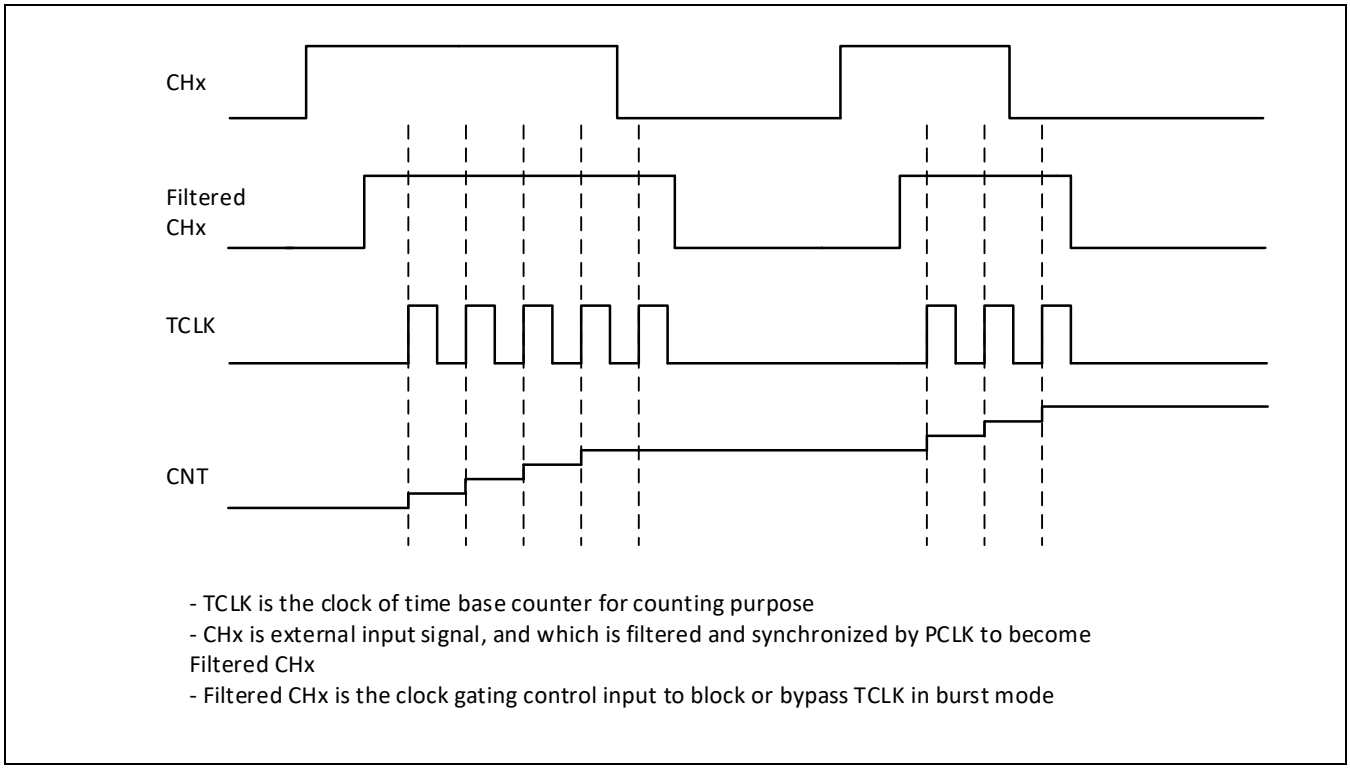


Figure 12-3 群脉冲时钟模式时序

在群脉冲时钟模式下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。

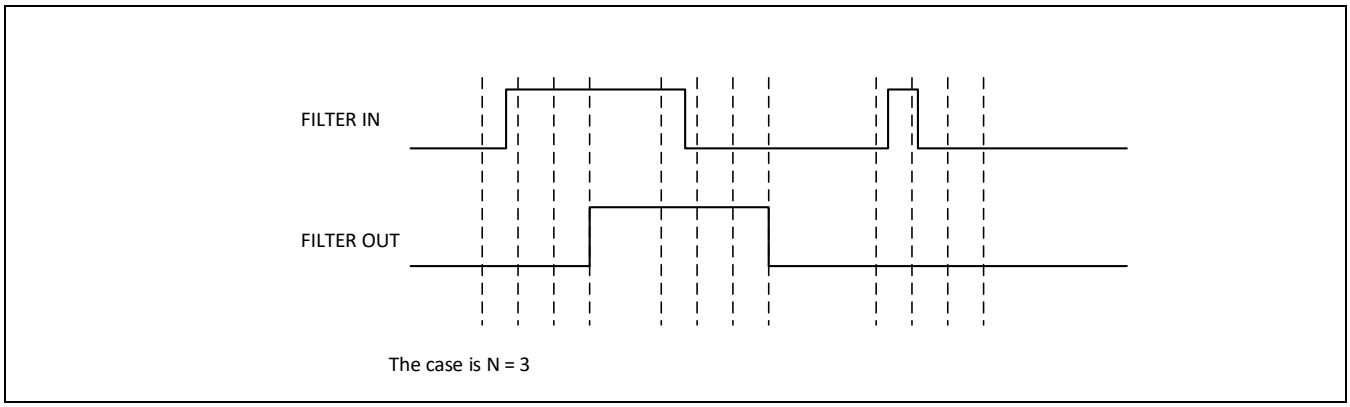


Figure 12-4 CG Filter 数字滤波器的原理

12.3.2 时基控制

12.3.2.1 概述

作为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。

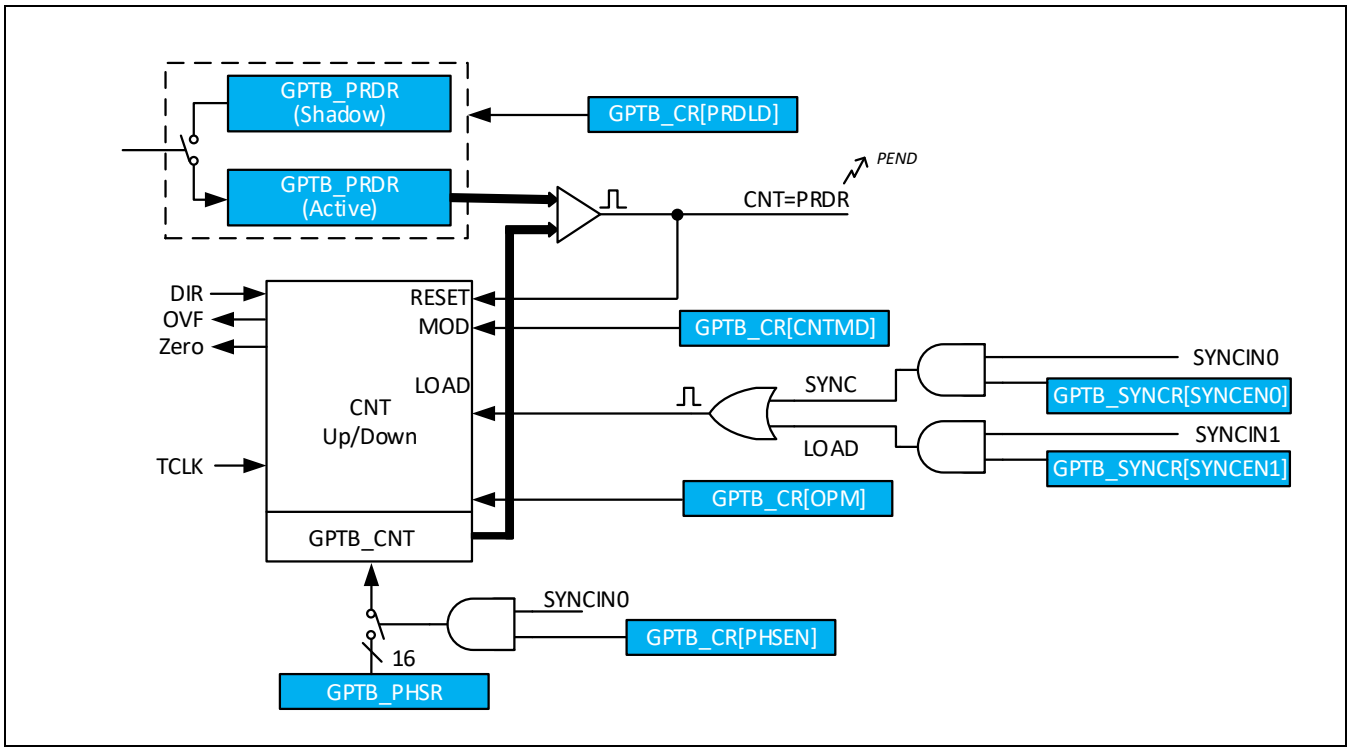


Figure 12-5 计数器时基模块

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

- 递增模式(Up-Counting Mode)：

  在递增模式下，时基计数器从0x0000开始递增计数，一直计数到周期设置值(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。

- 递减模式：(Down-Counting Mode)

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

- 递增递减模式：(Up-Down-Counting Mode)

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

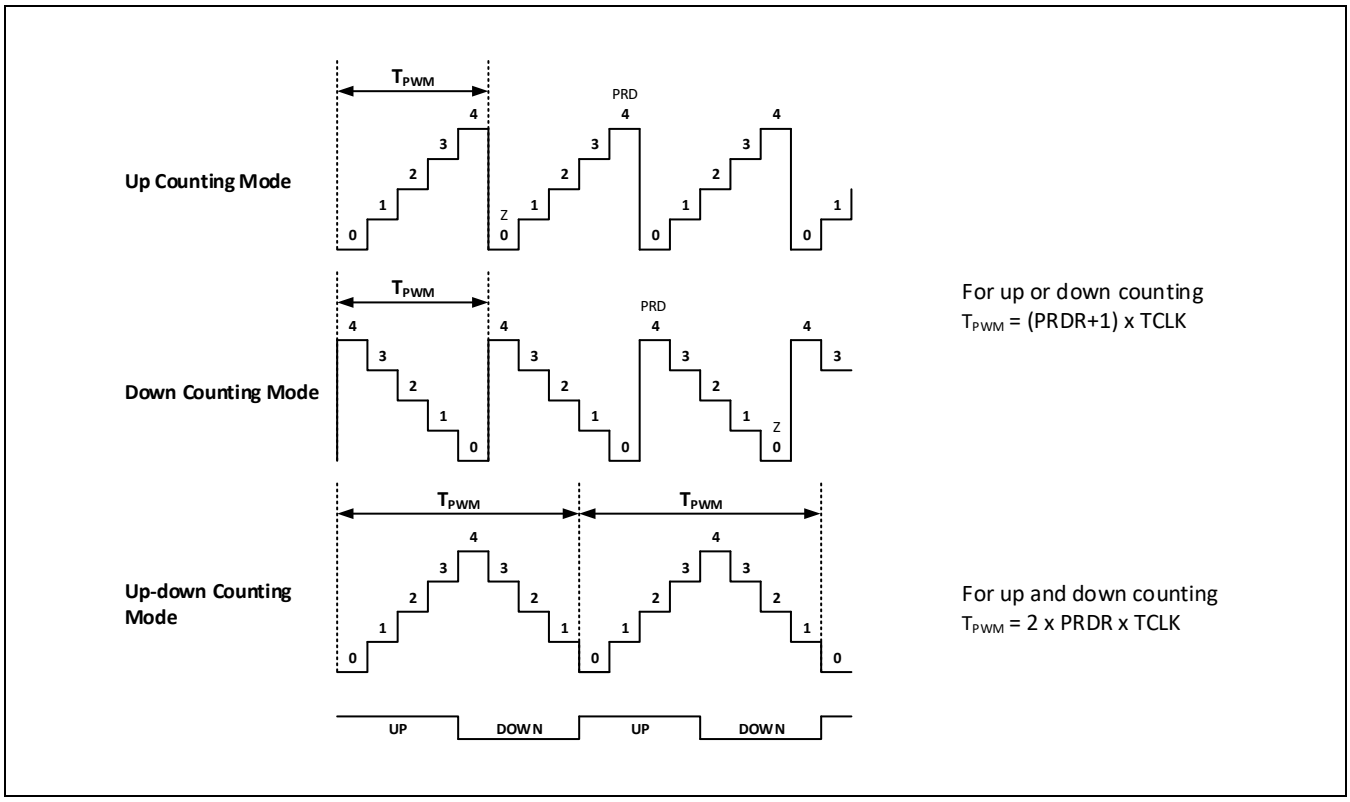


Figure 12-6 计数器工作模式

12.3.2.2 计数器重置和周期设置

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

12.3.2.3 计数模式和时序

时基计数器可以分为四种工作模式：

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

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

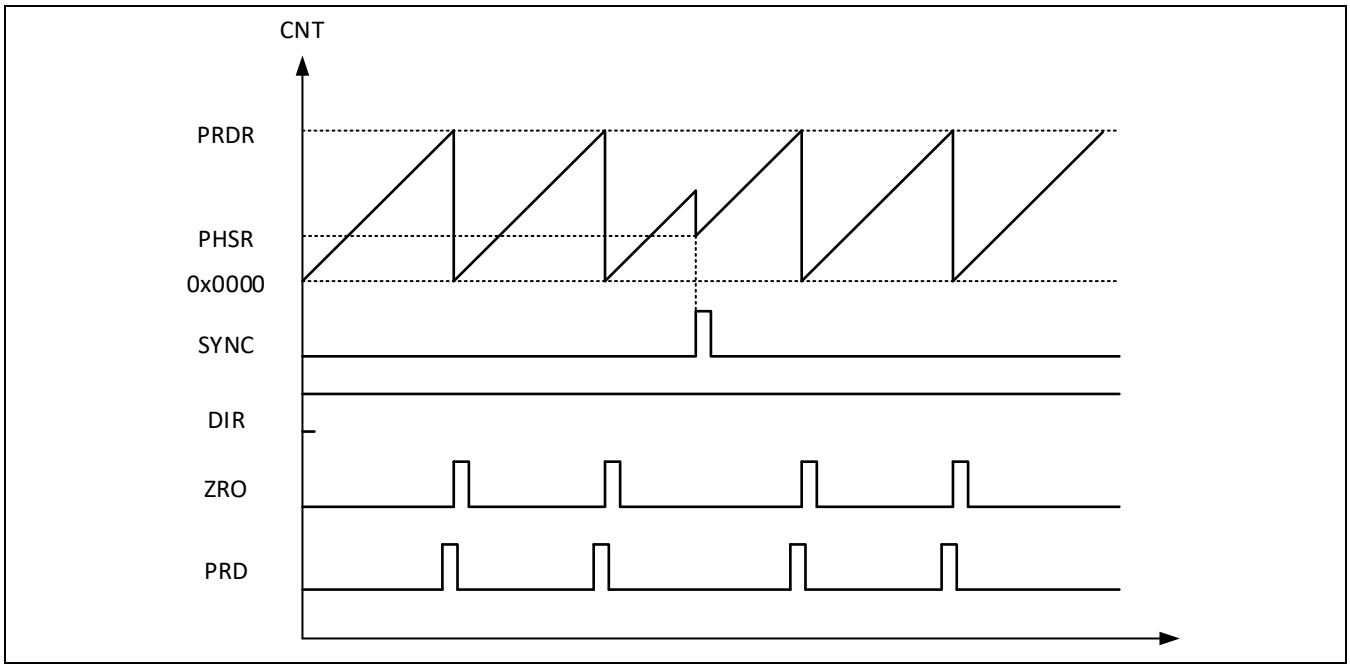


Figure 12-7 递增工作模式

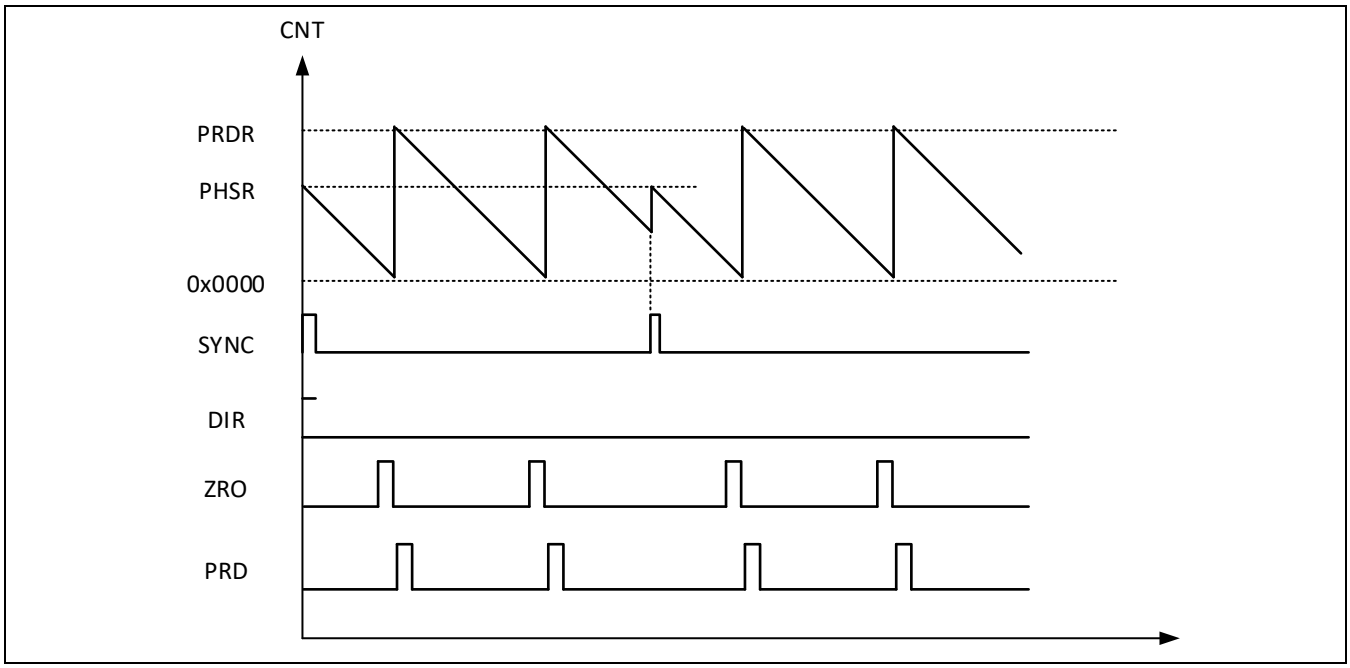


Figure 12-8 递减工作模式

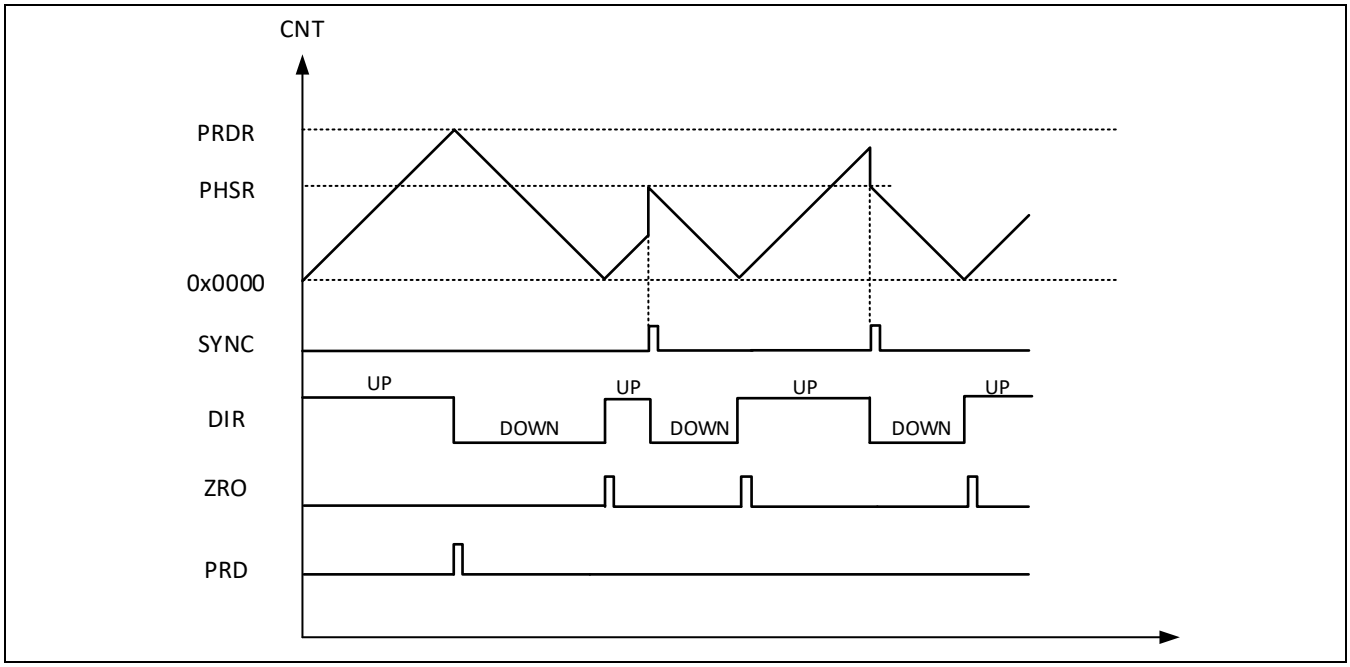


Figure 12-9 递增递减工作模式

12.3.2.4 全局载入控制

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。

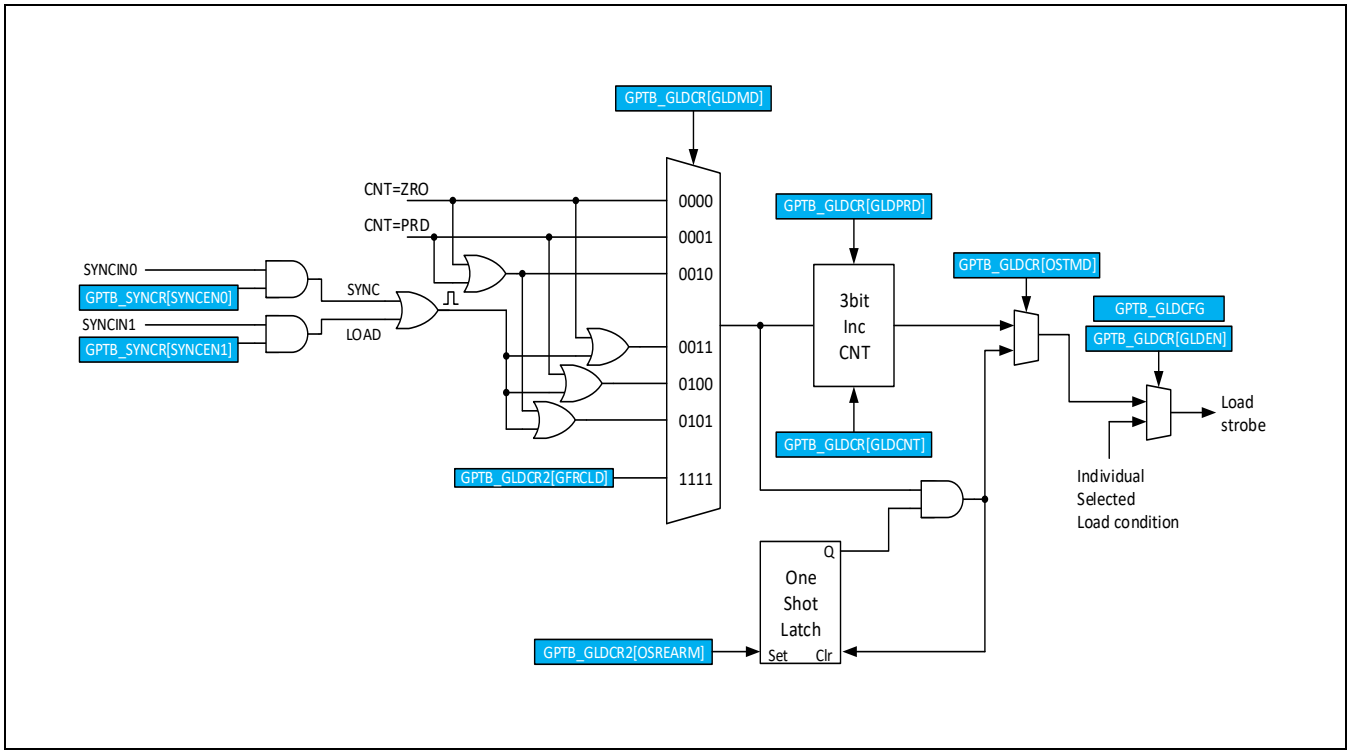


Figure 12-10 全局载入控制

12.3.3 计数器数值比较控制

12.3.3.1 概述

计数器数值比较控制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

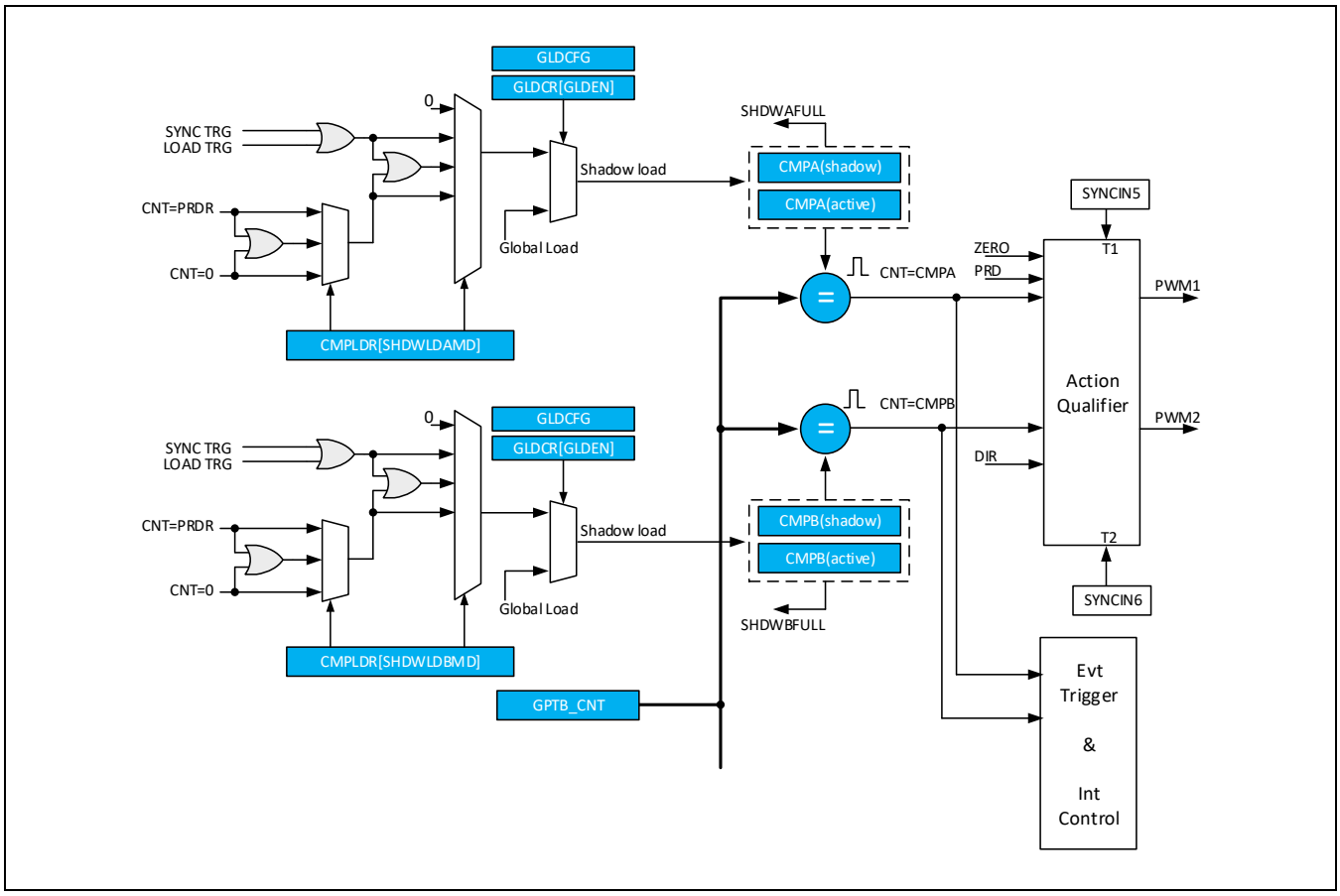


Figure 12-11 计数器值比较控制

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

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

12.3.3.2 比较值寄存器载入方式

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。

12.3.3.3 不同计数模式的时序

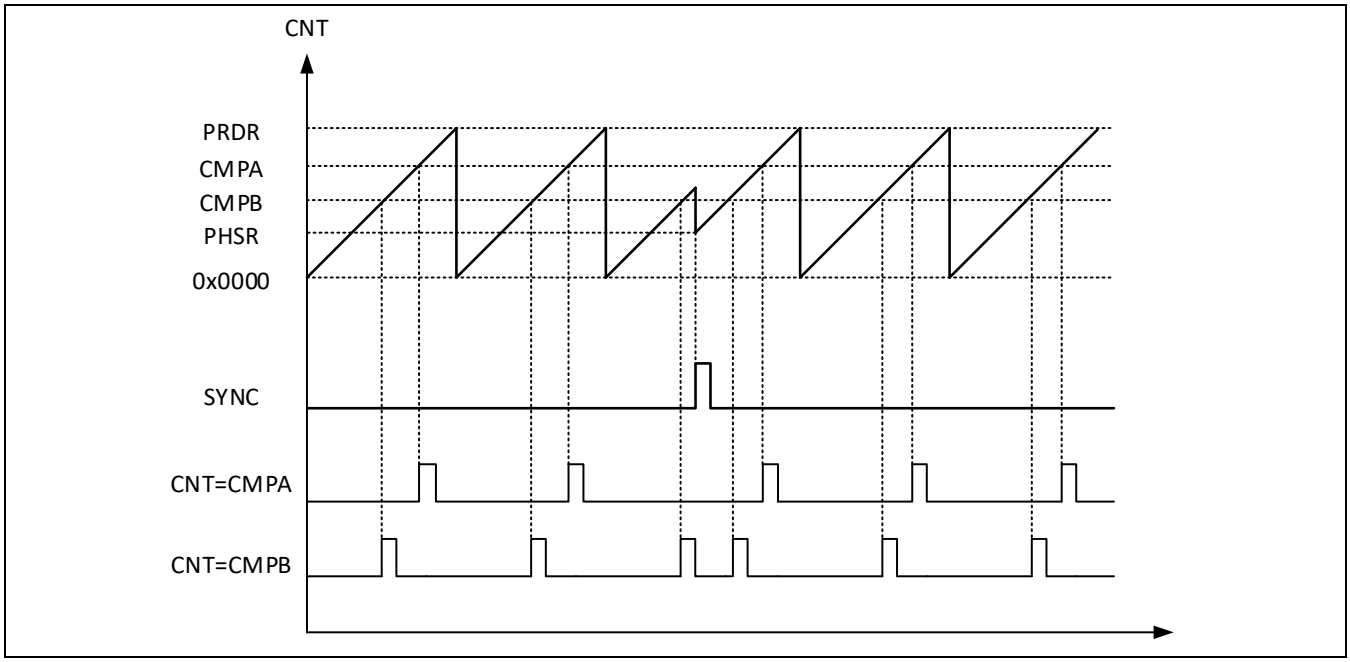


Figure 12-12 递增模式下比较事件产生时序

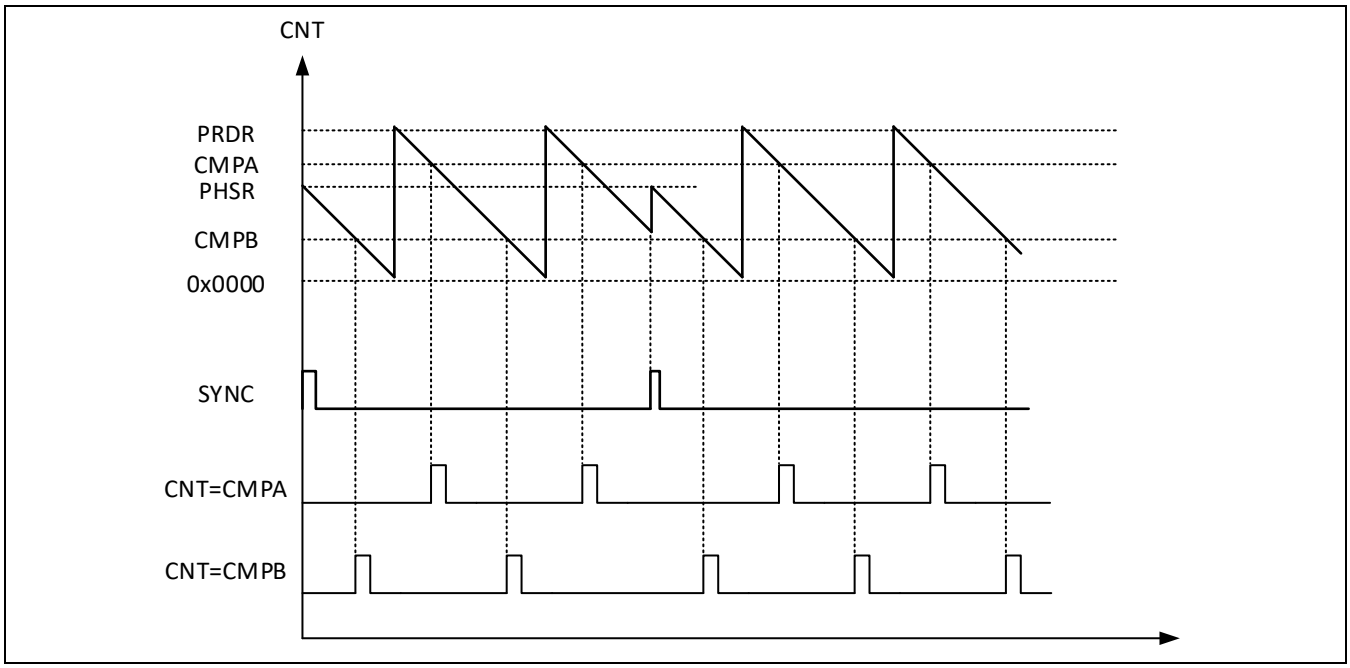


Figure 12-13 递减模式下比较事件产生时序

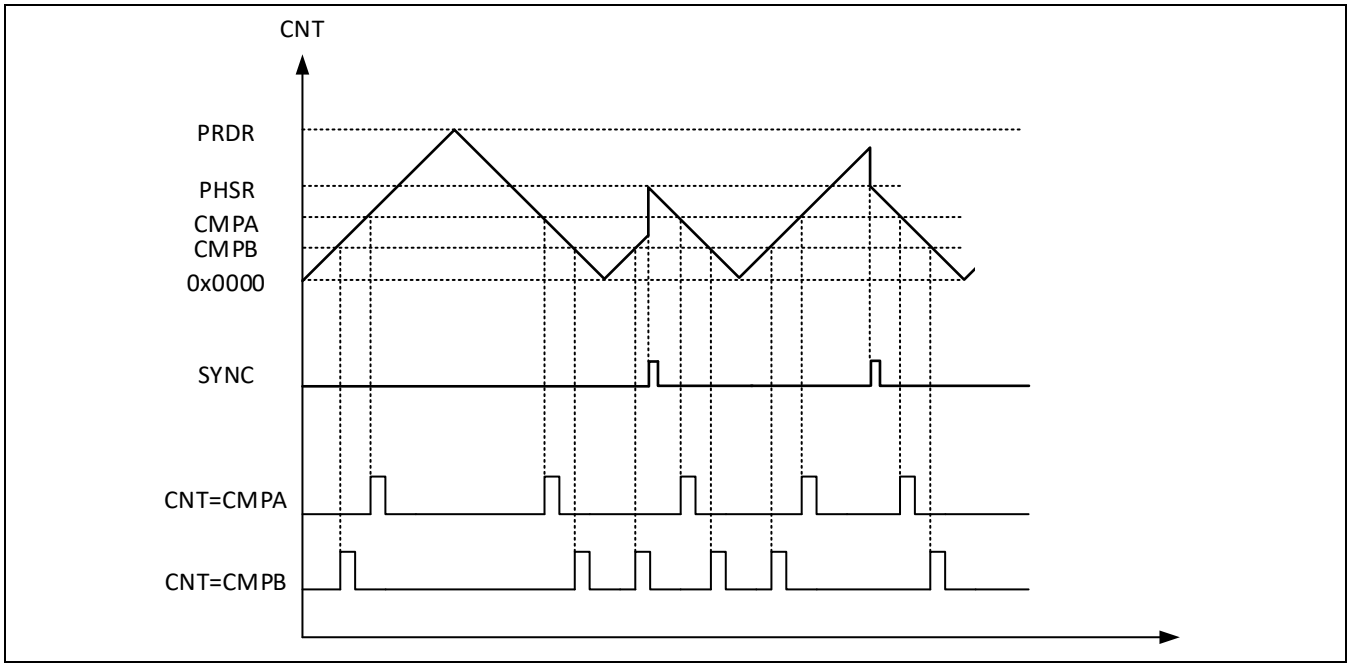


Figure 12-14 递增递减模式下比较事件产生时序

12.3.4 波形发生控制

12.3.4.1 事件驱动的波形输出

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)
- T1 事件 (SYNCIN5事件)
- T2 事件 (SYNCIN6事件)
- 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)

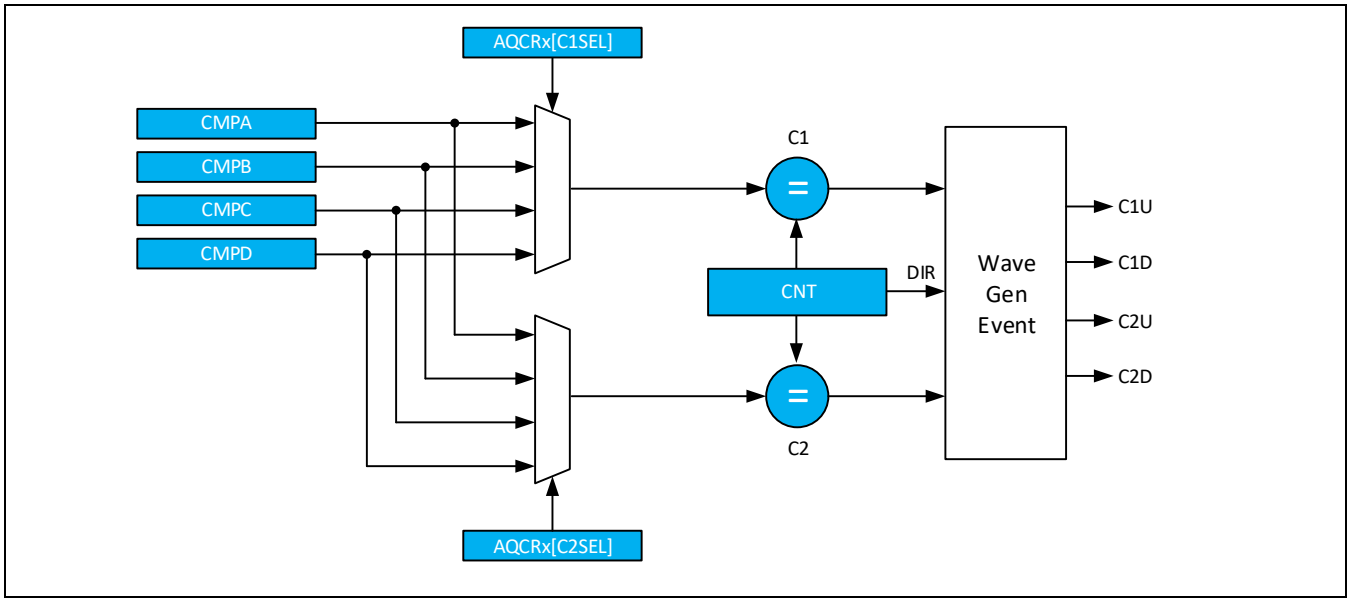


Figure 12-15 C1和C2选择控制

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

Table 12-2 各种在PWM1和PWM2上可能触发的动作

软件 Force	CNT 值等于				事件触发		动作
	Zero	CMPA	CMPB	PRD	T1	T2	

SW	Z	C1	C2	P	T1	T2	
							没有动作 (保持原来状态)
							低电平输出
							高电平输出
							翻转输出

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

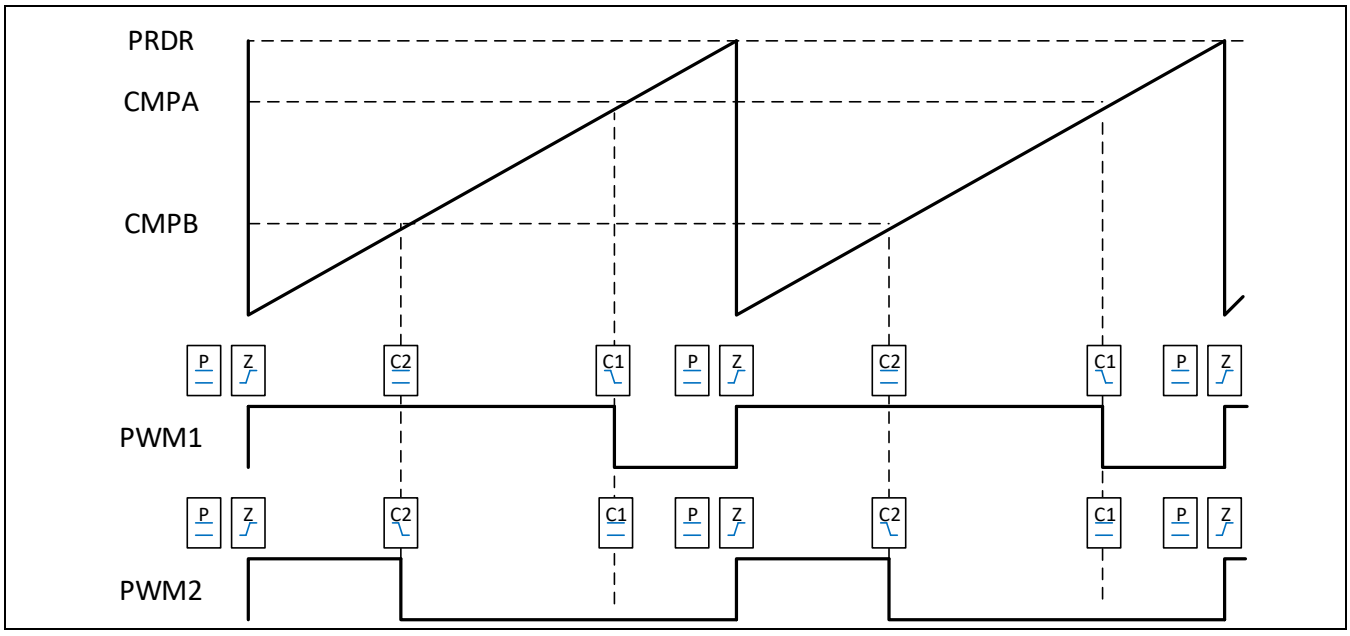


Figure 12-16 递增单沿，非对称波形输出

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

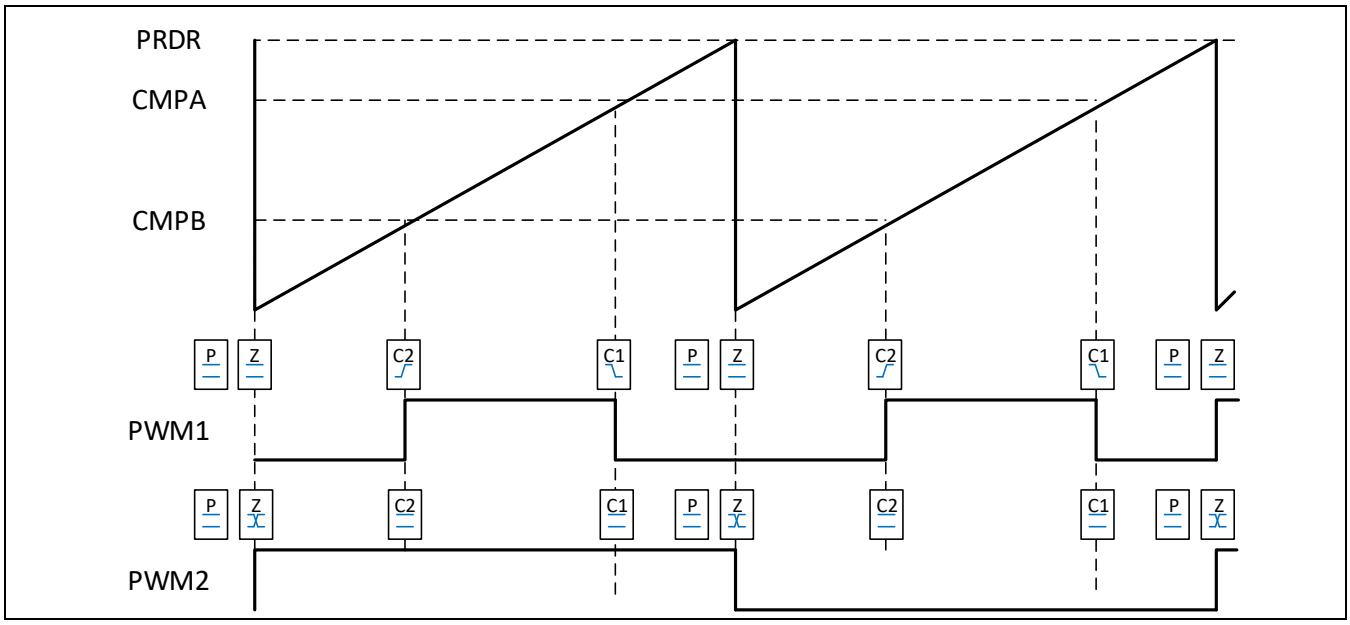


Figure 12-17 递增，脉冲定位非对称波形输出

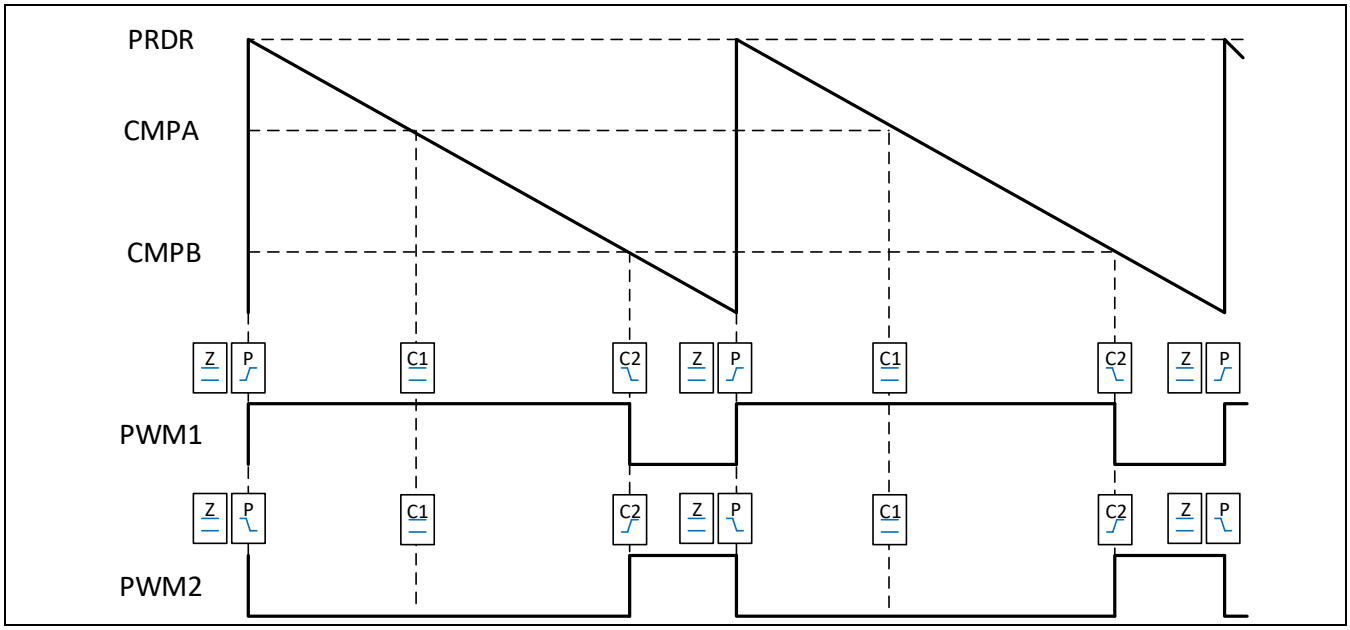


Figure 12-18 递减单沿，非对称波形输出

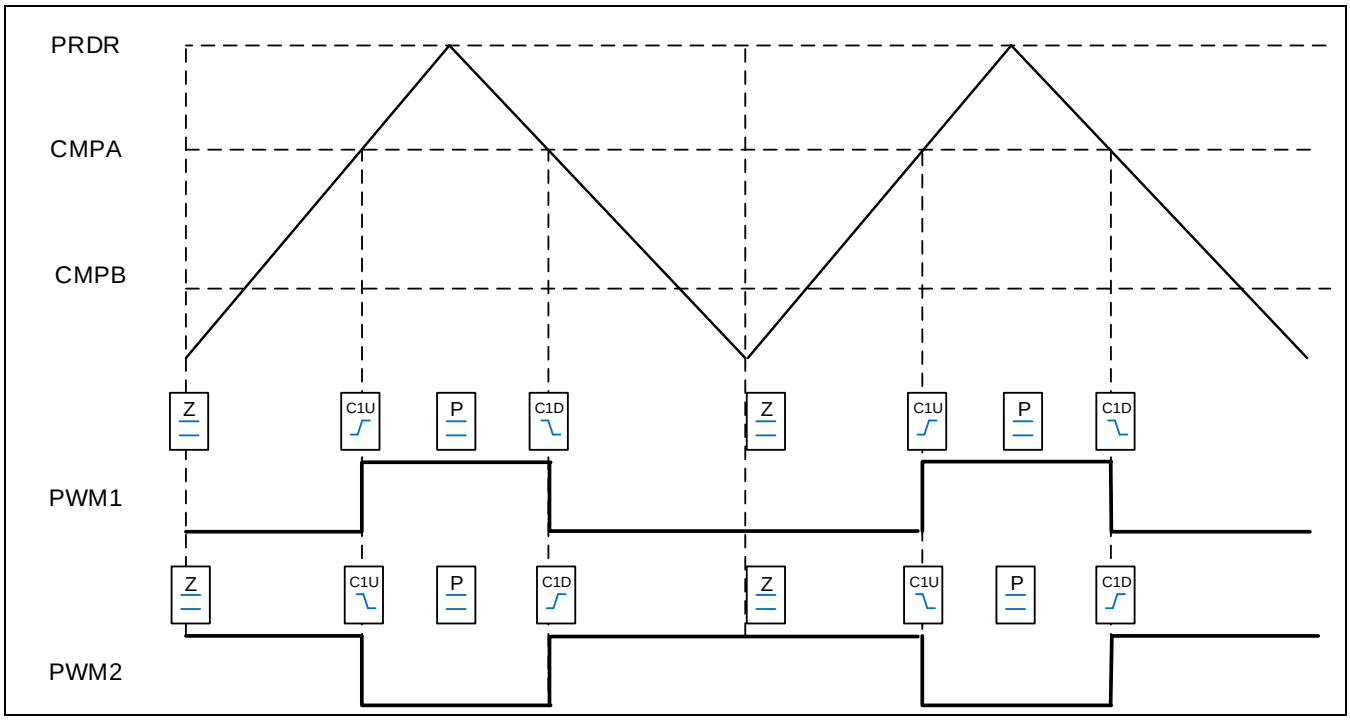


Figure 12-19 递增递减，双沿对称波形输出

12.3.4.2 触发事件的优先级

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

Table 12-3 递增递减模式(Up-Down-Count)下的事件优先级

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

Table 12-4 递增模式下的事件优先级

优先级	触发事件
1(Highest)	Software Forced event

2	CNT equals period
3	T1 on up-count(T1U)
4	T2 on up-count(T2U)
5	CNT equals CMPB on up-count(C2U)
6	CNT equals CMPA on up-count(C1U)
7(Lowest)	CNT equals zero

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

Table 12-5 递减模式下的事件优先级

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

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

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

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

12.3.4.3 通过软件强制设置波形

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

软件强制输出可以分为两种模式：一次性Force和连续Force。

一次性软件强制输出(One-Shot Software Forcing)

在此模式下，通过寄存器的设置可以将通道PWM1/2/3/4(注意不是最终管脚上的输出波形)的输出强制修改成软件设置电平，且该电平一直维持到有新的触发事件发生。可以通过设置寄存器GPTB_AQOSF[ACT1/2]控制位，设置在一次性强制输出触发时，PWM1/2/3/4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。

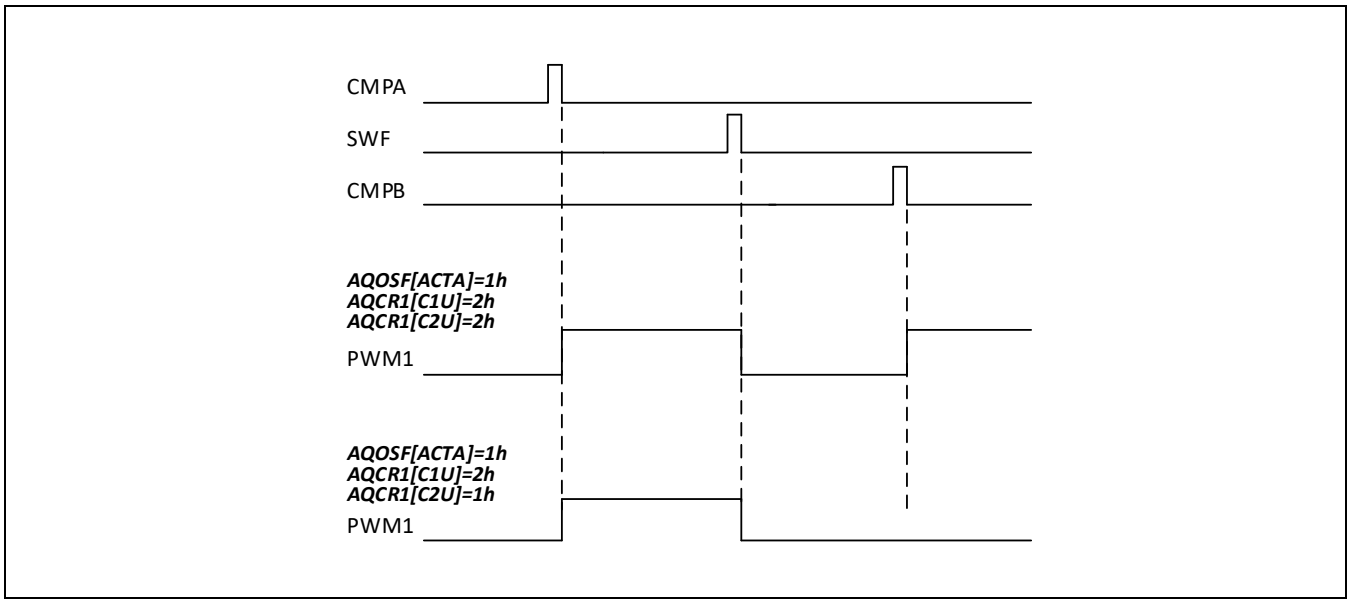


Figure 12-20 一次性软件强制输出

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。

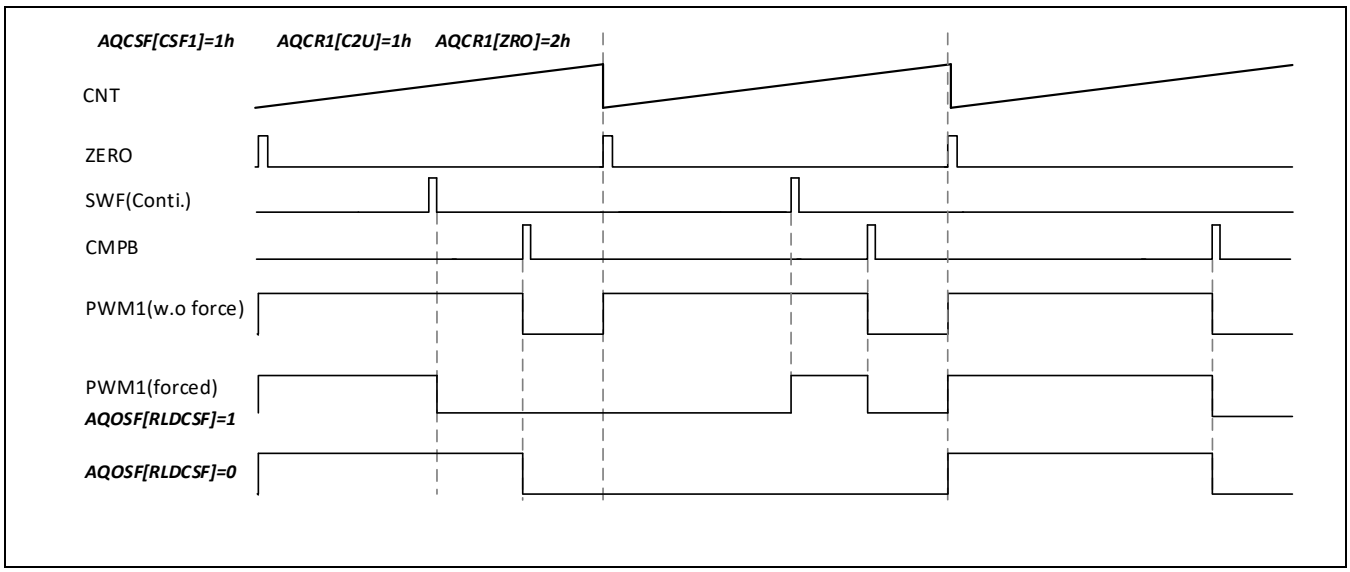


Figure 12-21 持续性软件强制输出

12.3.4.4 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）。

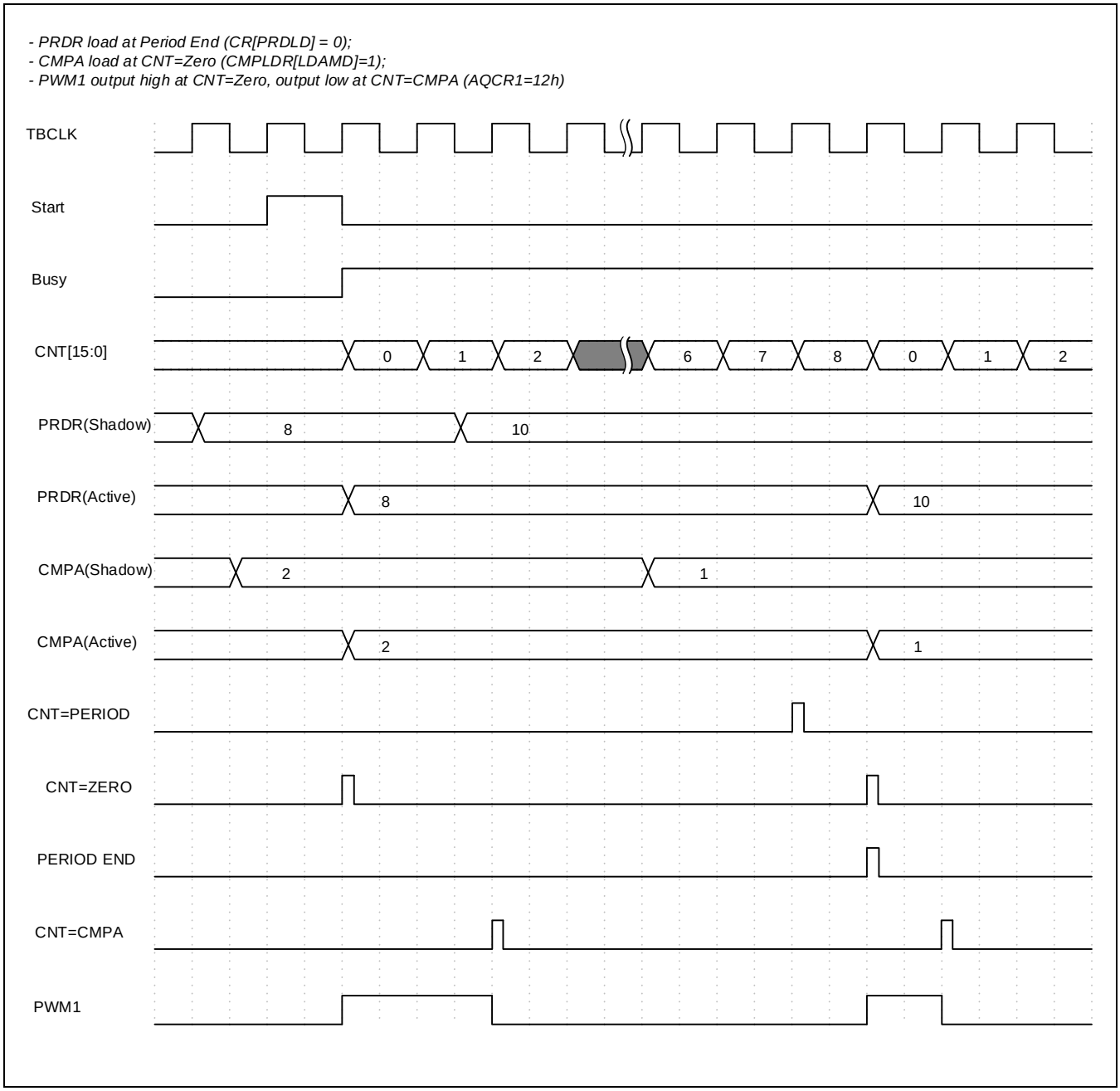


Figure 12-22 递增单比较值时，非对称波形输出

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

空比输出的情下，需要选择递增计数模式，或者递增递减计数模式）。

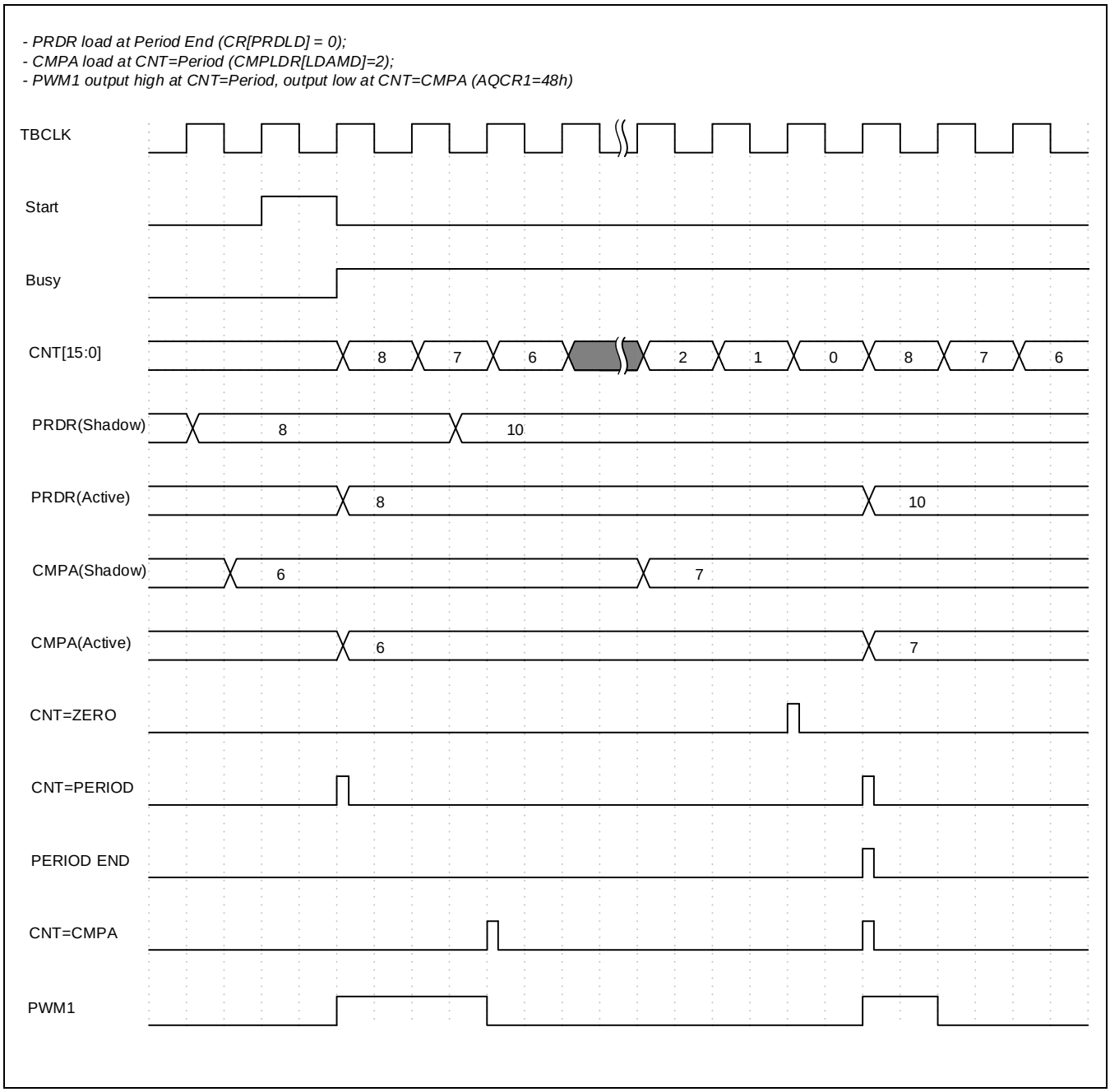


Figure 12-23 递减单比较值时，非对称波形输出

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

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

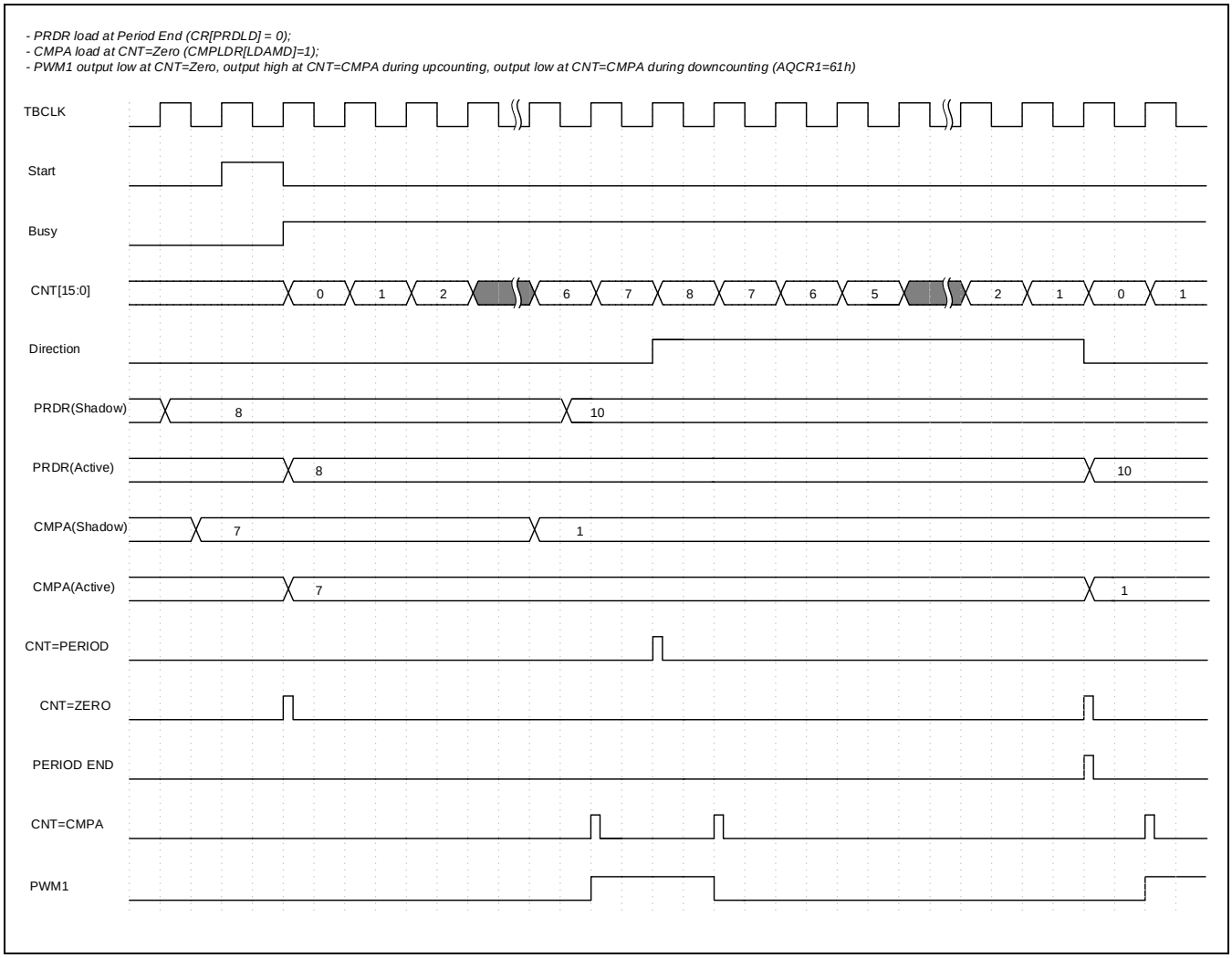


Figure 12-24 递增递减单比较值时，对称波形输出

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

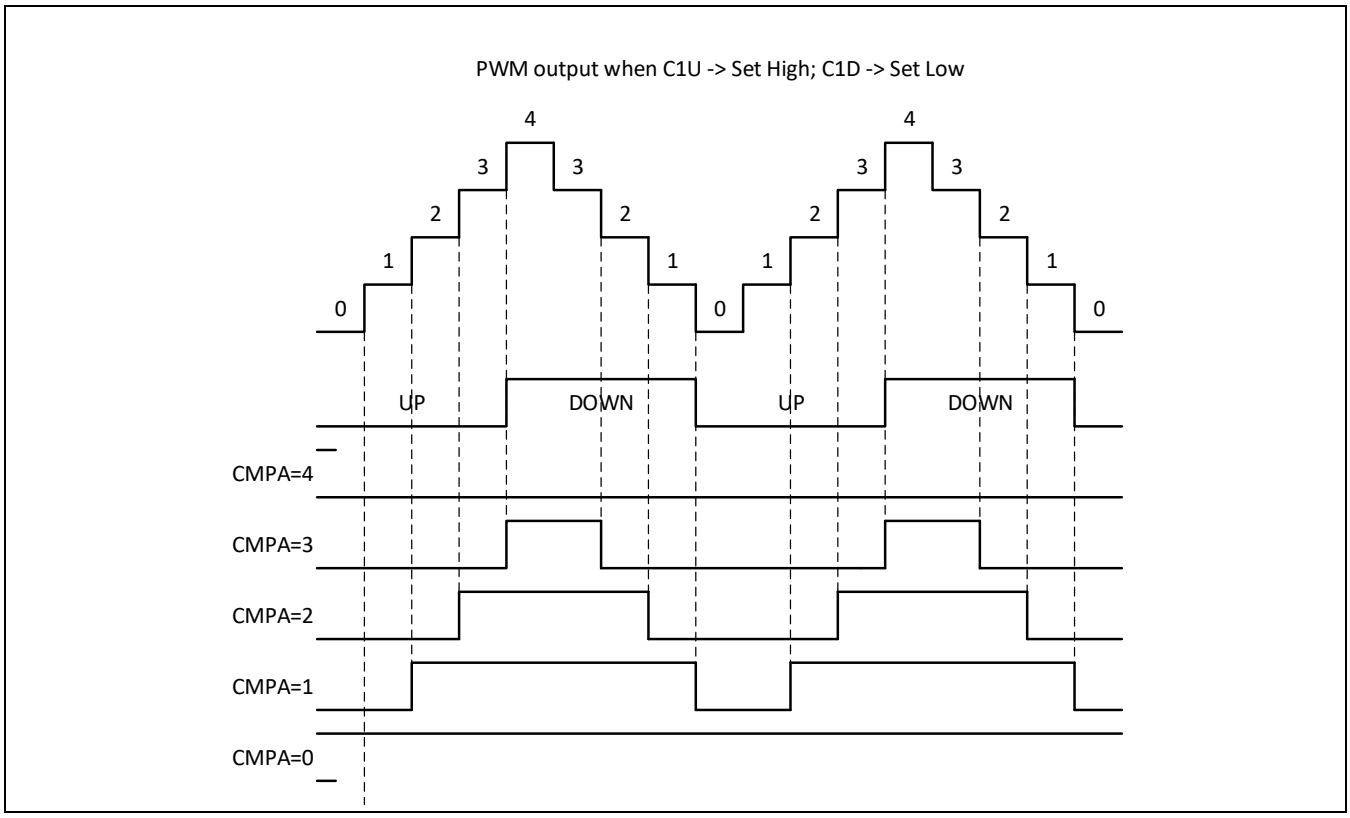


Figure 12-25 递增递减单比较值时，对称波形输出

12.3.5 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。

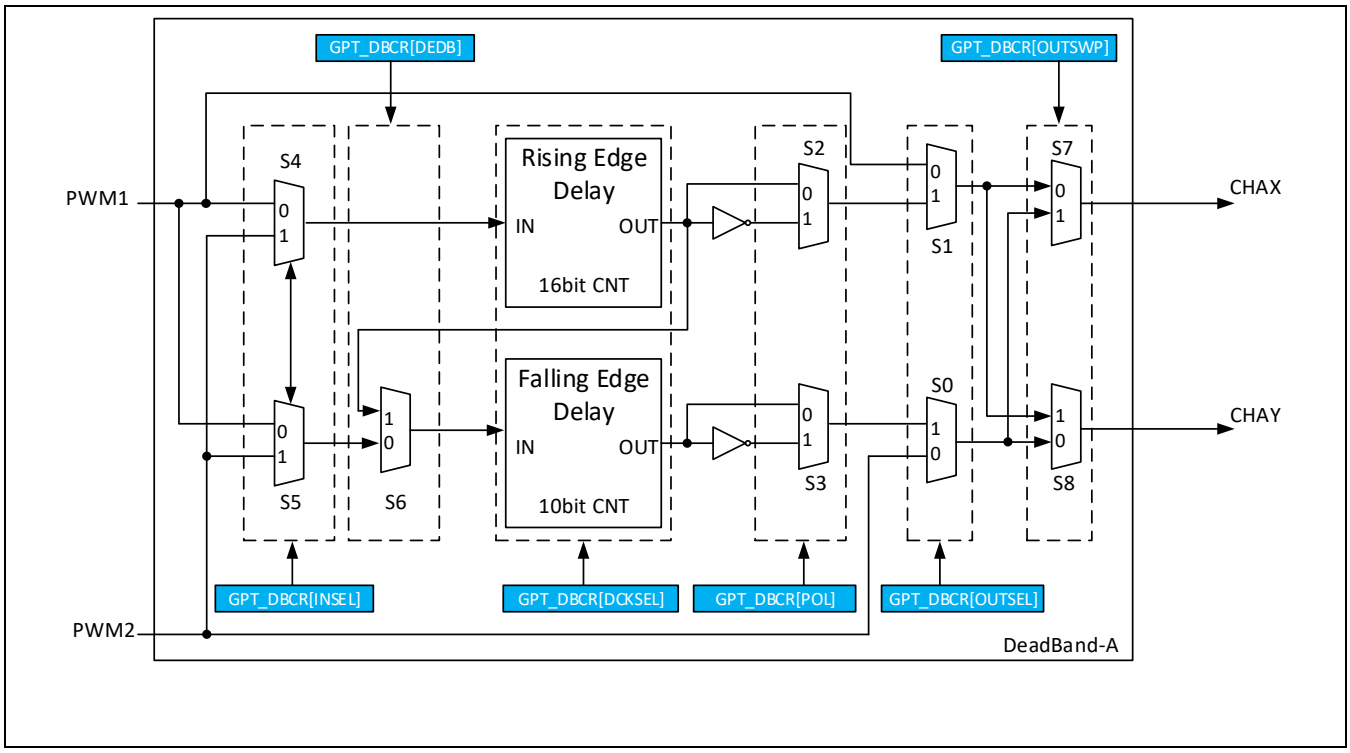


Figure 12-26 死区控制模块

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

开关	功能描述	寄存器控制位
S4，S5	对于X通道和Y通道的延时电路，独立选择两个来自于PWM数字引擎的PWM输出信号作为输入源。对于DeadBandA，PWM1/2可选。	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：

$T_{RED} = DTR \times T_{DBCLK}$

$T_{FED} = DTF \times T_{DBCLK}$

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 12-6 支持的死区控制模式

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

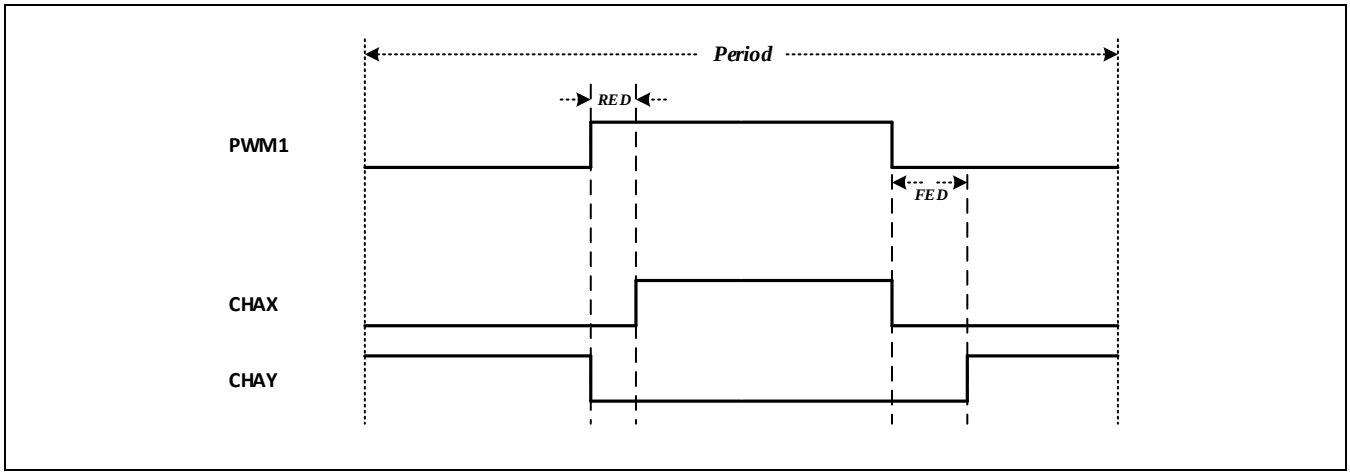


Figure 12-27 模式2：高电平有效的死区互补输出举例

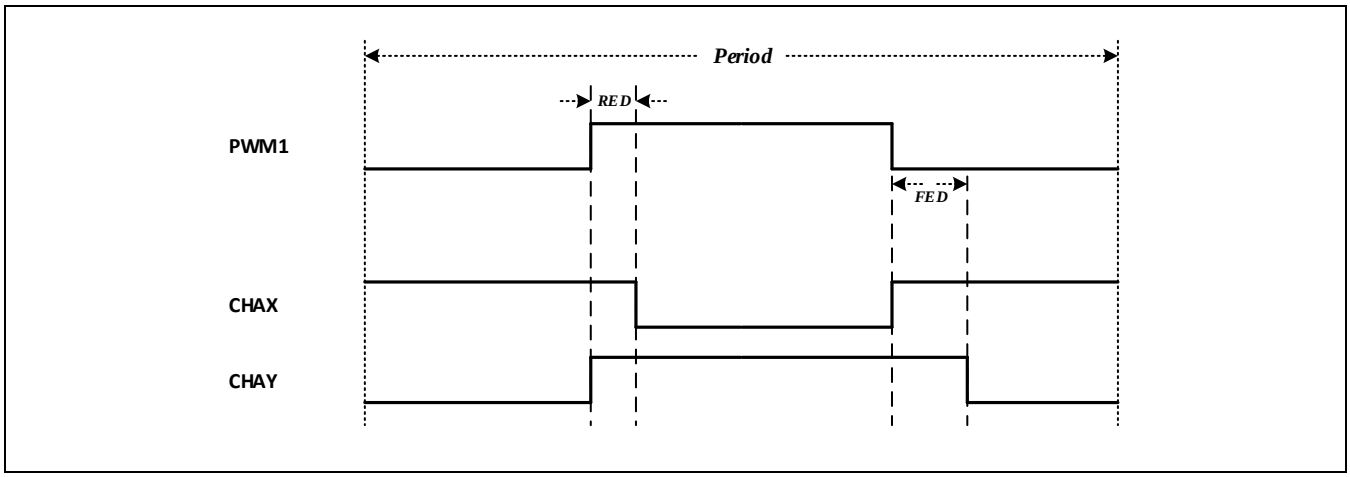


Figure 12-28 模式3：低电平有效的死区互补输出举例

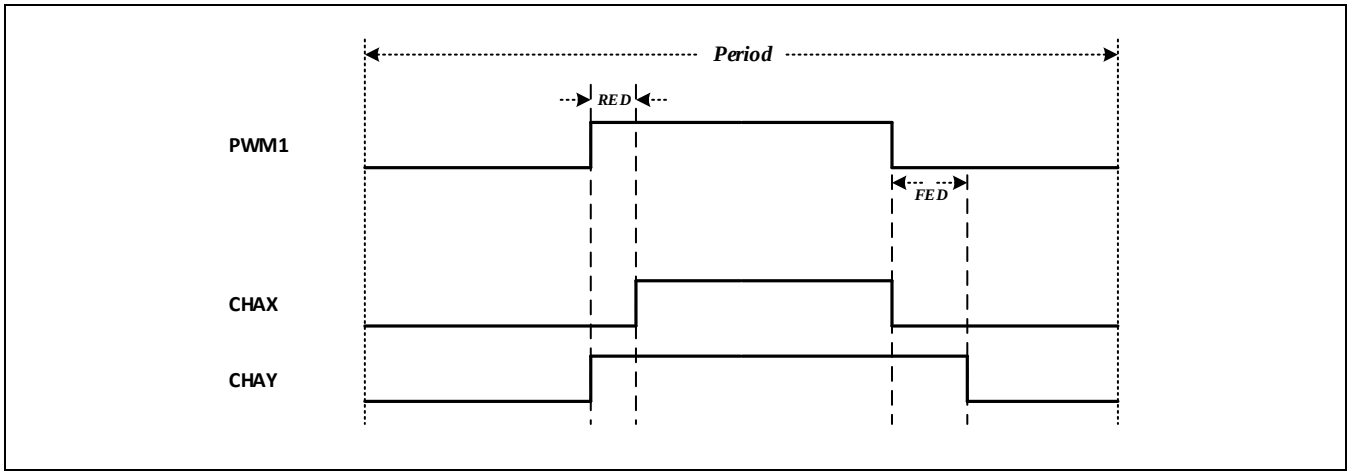


Figure 12-29 模式4：高低电平有效死区输出举例

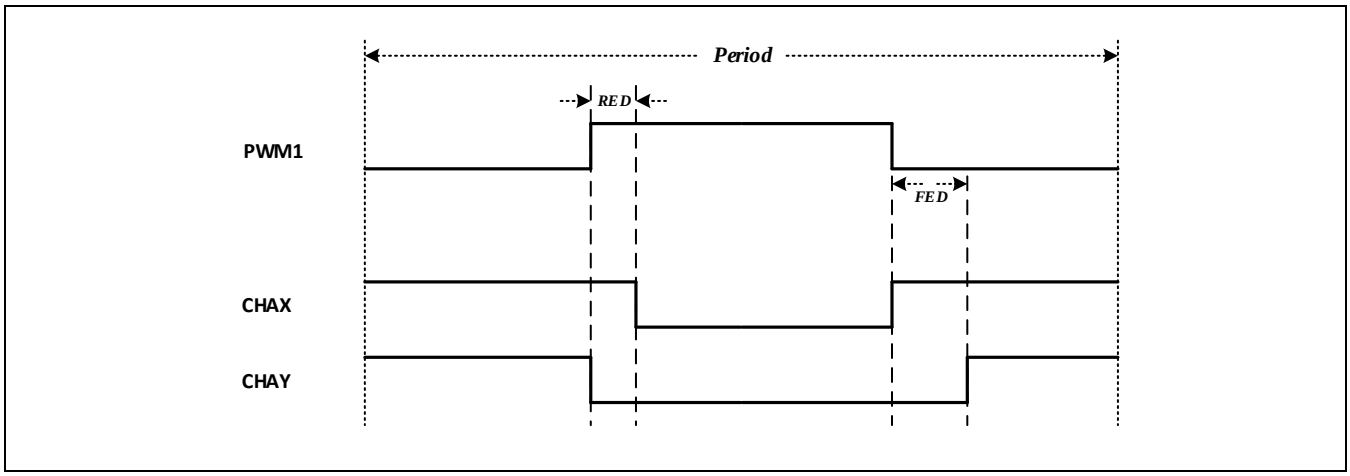


Figure 12-30 模式5：低高电平有效死区输出举例

12.3.6 紧急模式控制

12.3.6.1 紧急模式工作机制

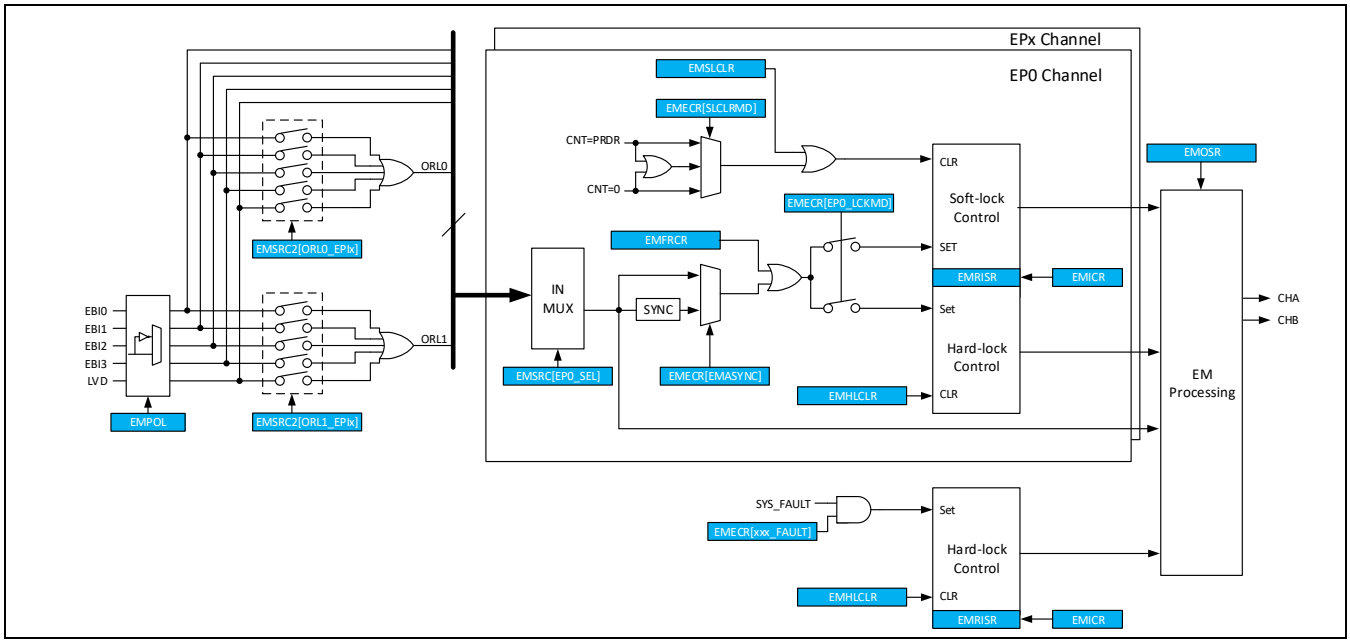


Figure 12-31紧急模式控制模块

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。

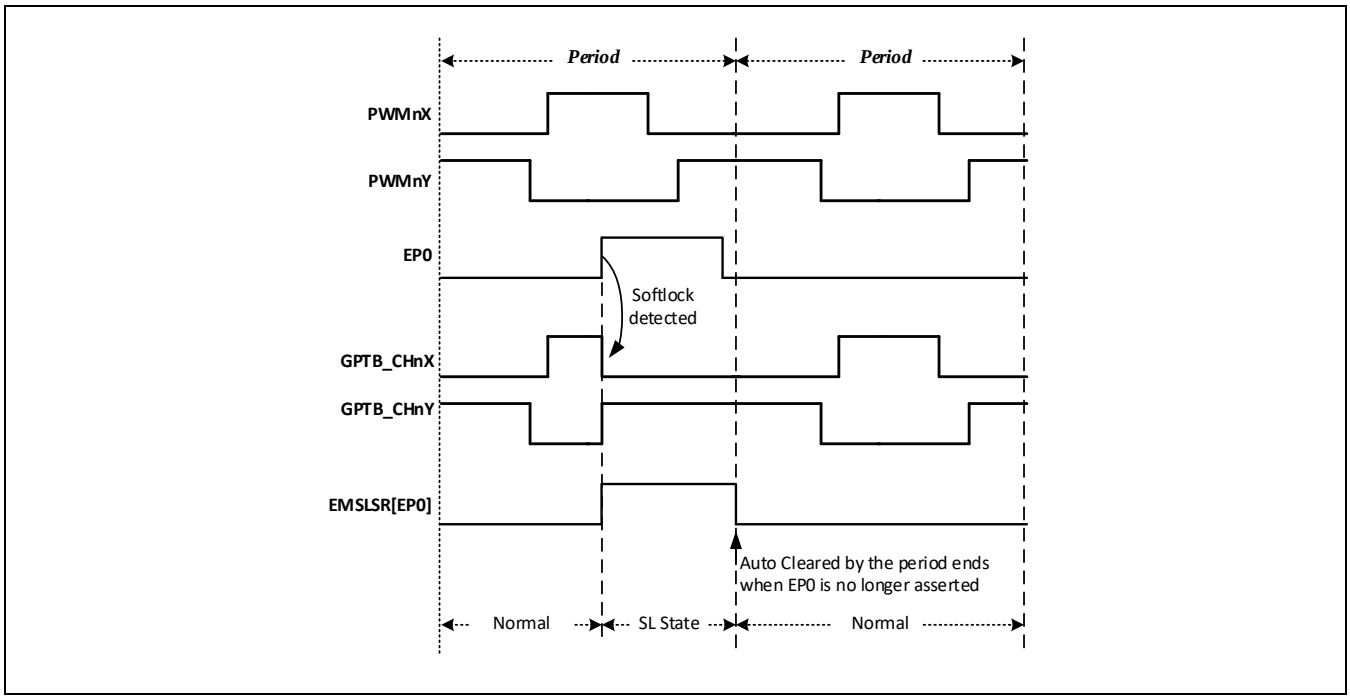


Figure 12-32 软锁止模式

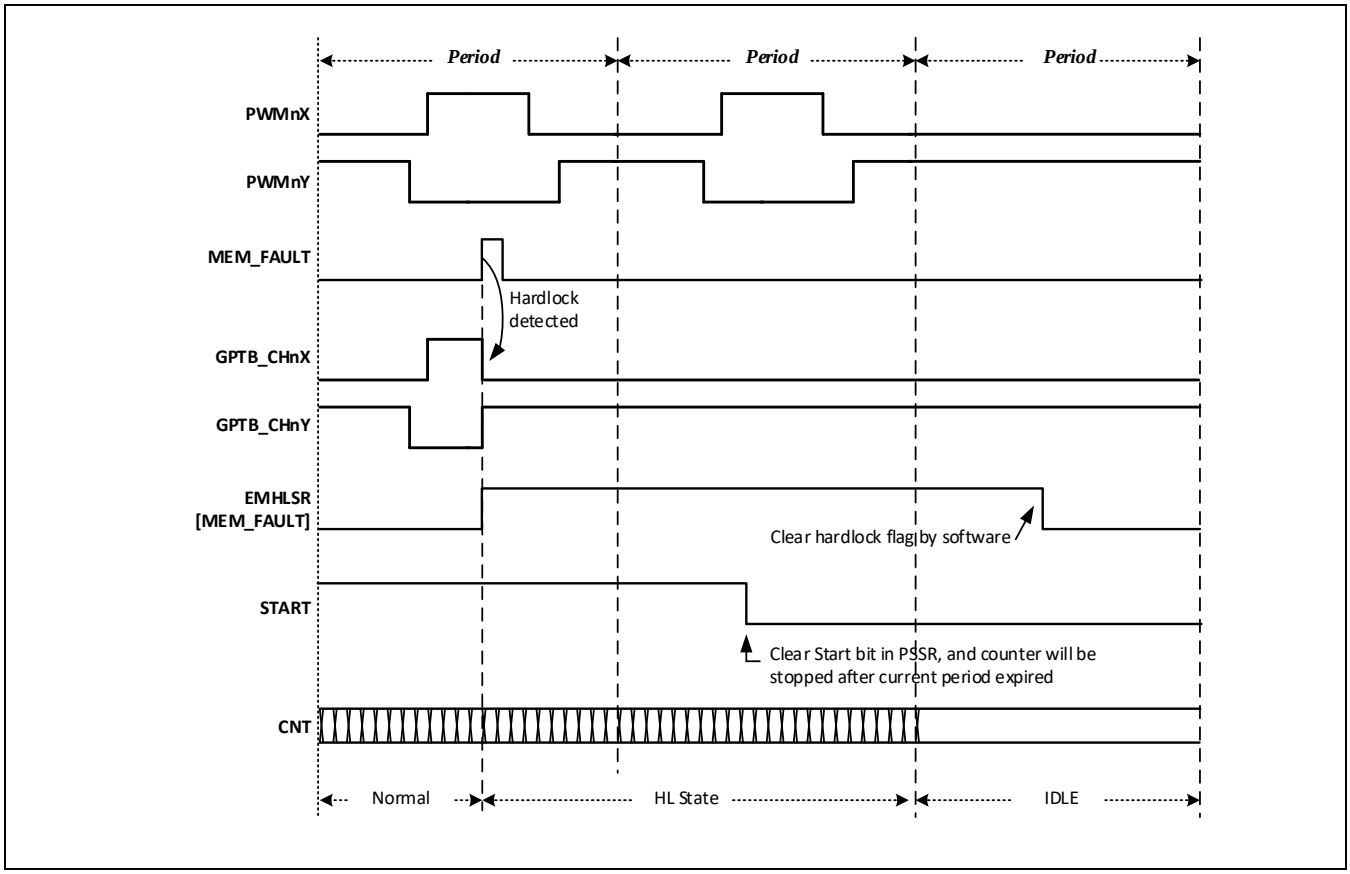


Figure 12-33 硬锁止模式

12.3.6.2 紧急模式中断

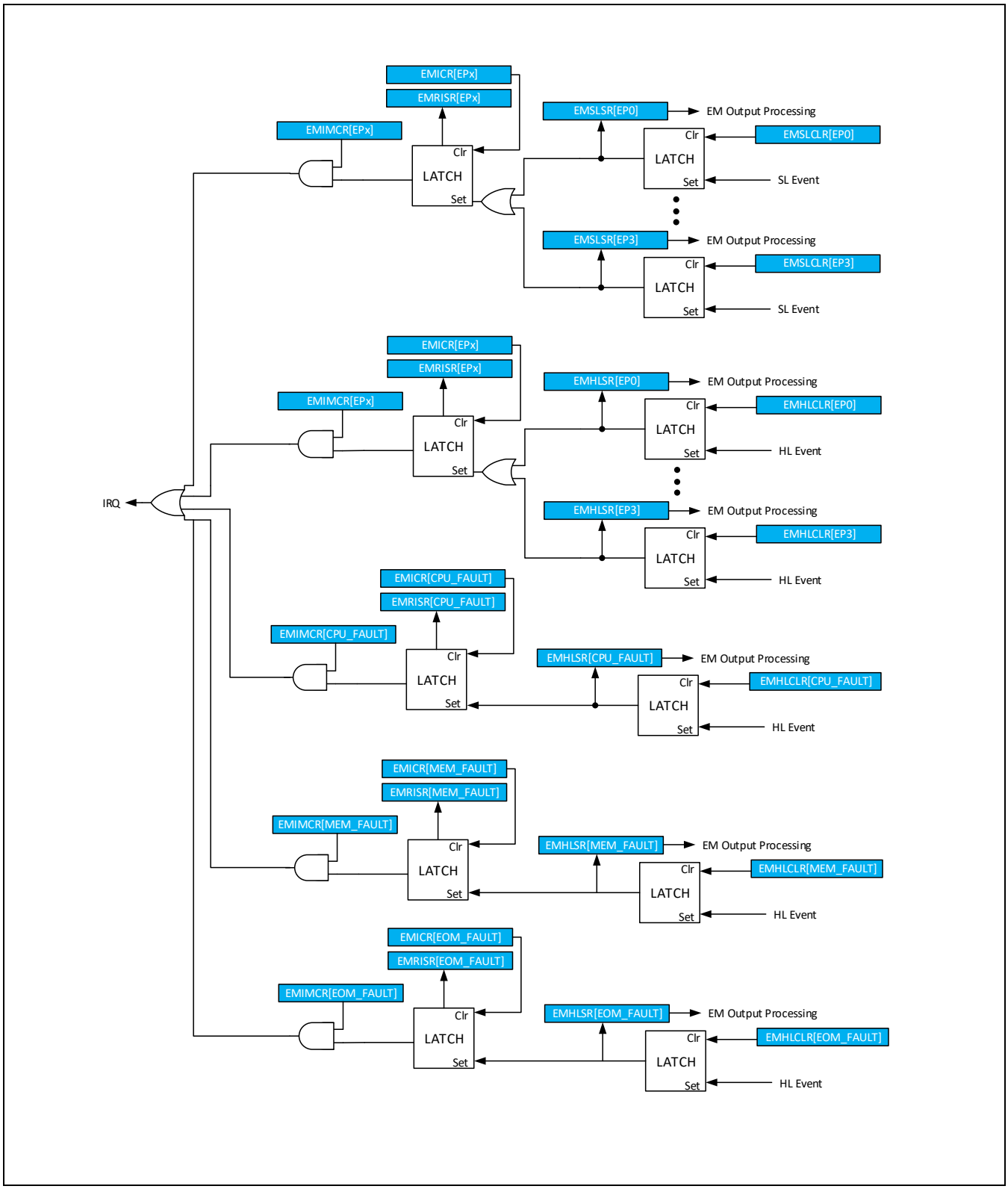


Figure 12-34 紧急模式中断

12.3.7 捕获模式

12.3.7.1 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

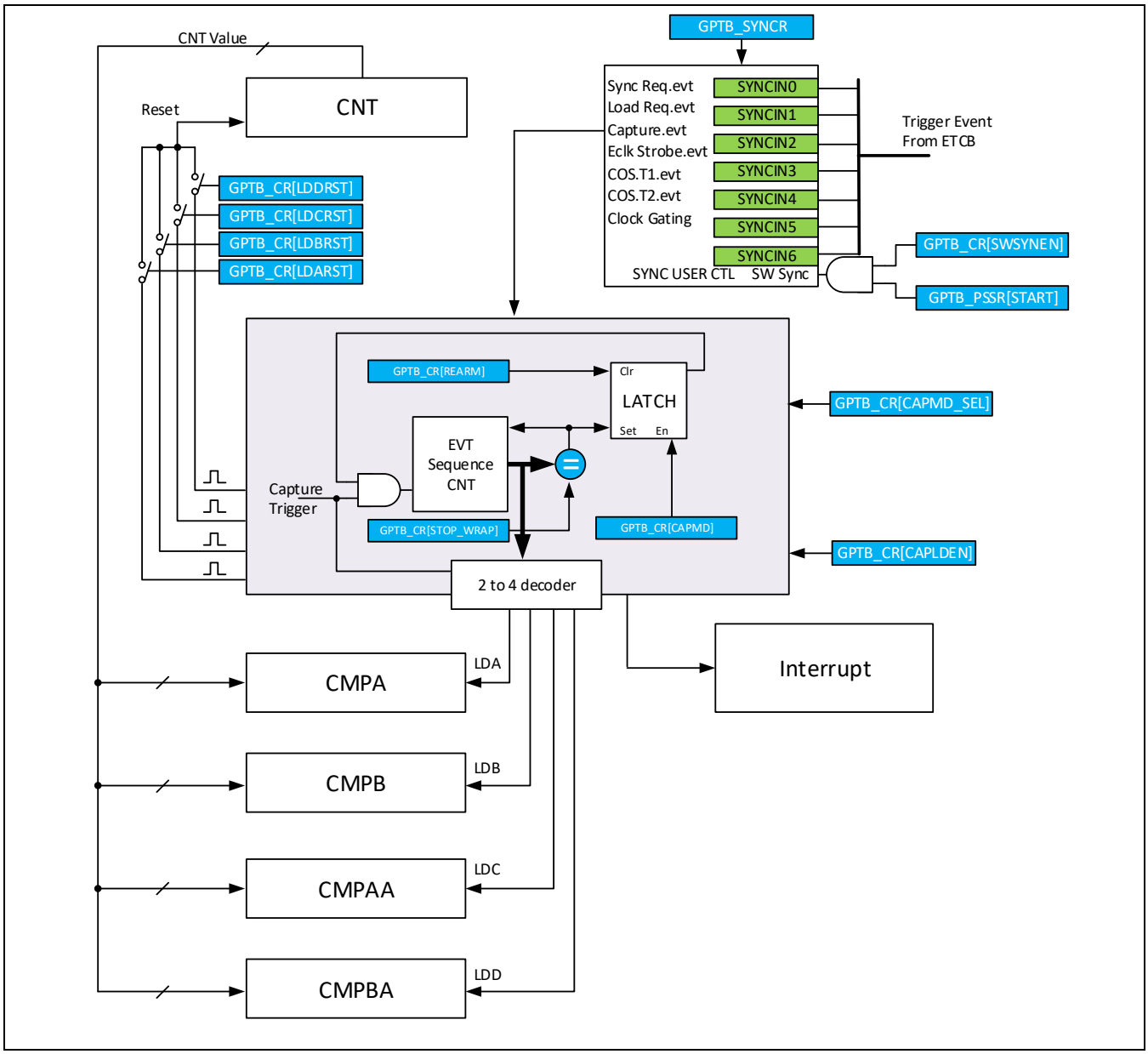


Figure 12-35 捕获模式结构框图

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

12.3.7.2 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 12-7 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

12.3.7.3 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。

12.3.7.4 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。

12.3.7.5 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)

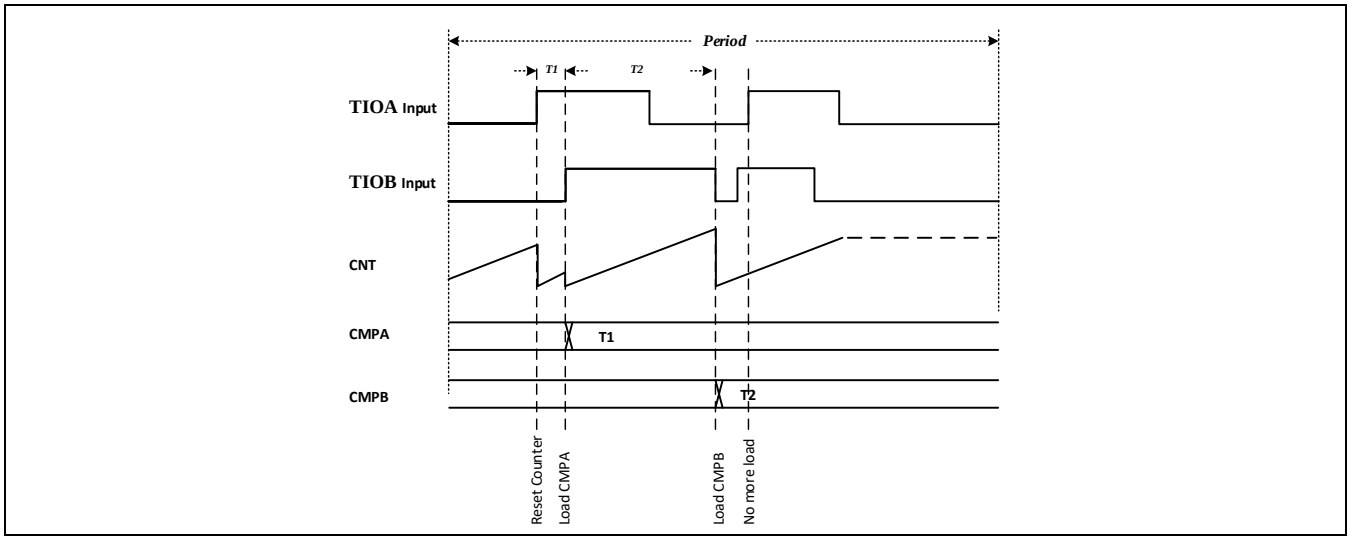


Figure 12-36 CHA的脉冲宽度检测

- **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)

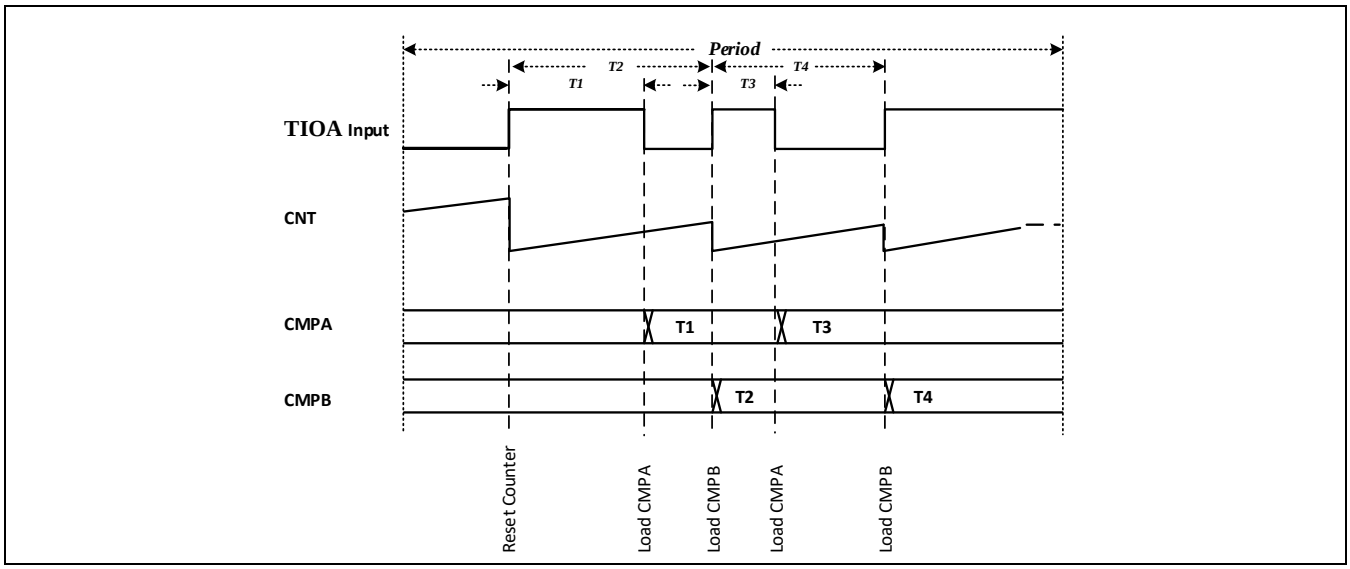


Figure 12-37 CHA的脉冲宽度和周期

12.3.8 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。

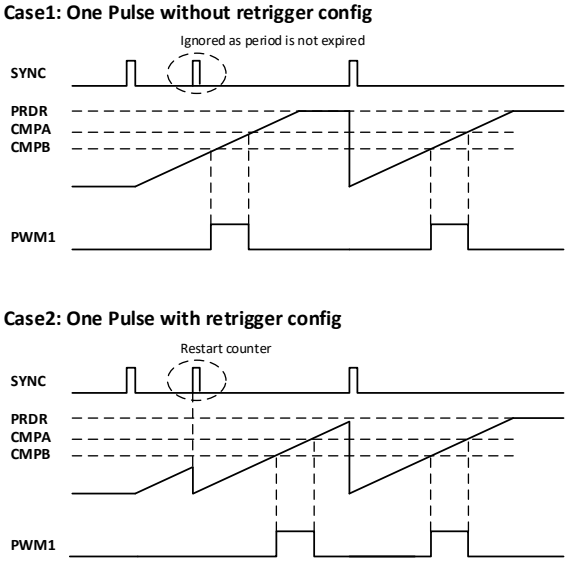


Figure 12-38 单次触发

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。

12.3.9 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。

12.3.9.1 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
- 计数器值递增或递减一个计数值
- 触发改变PWM的输出状态

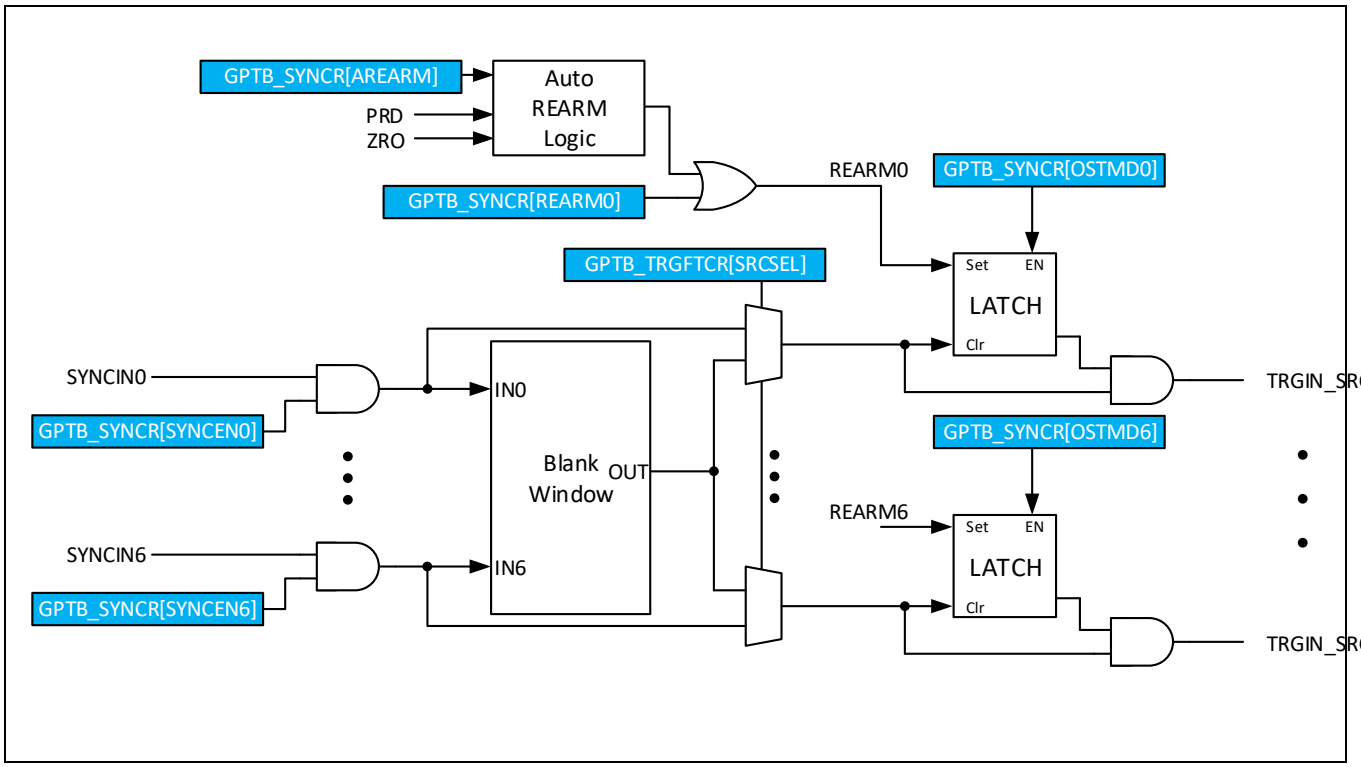


Figure 12-39 同步触发输入

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。

12.3.9.2 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。

CNT增减触发：计数器值递增或递减一个计数值(SYNCIN4)

当该端口被触发，计数器将根据当前计数方向，自动增加或减少一个计数值。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。

12.3.9.3 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。

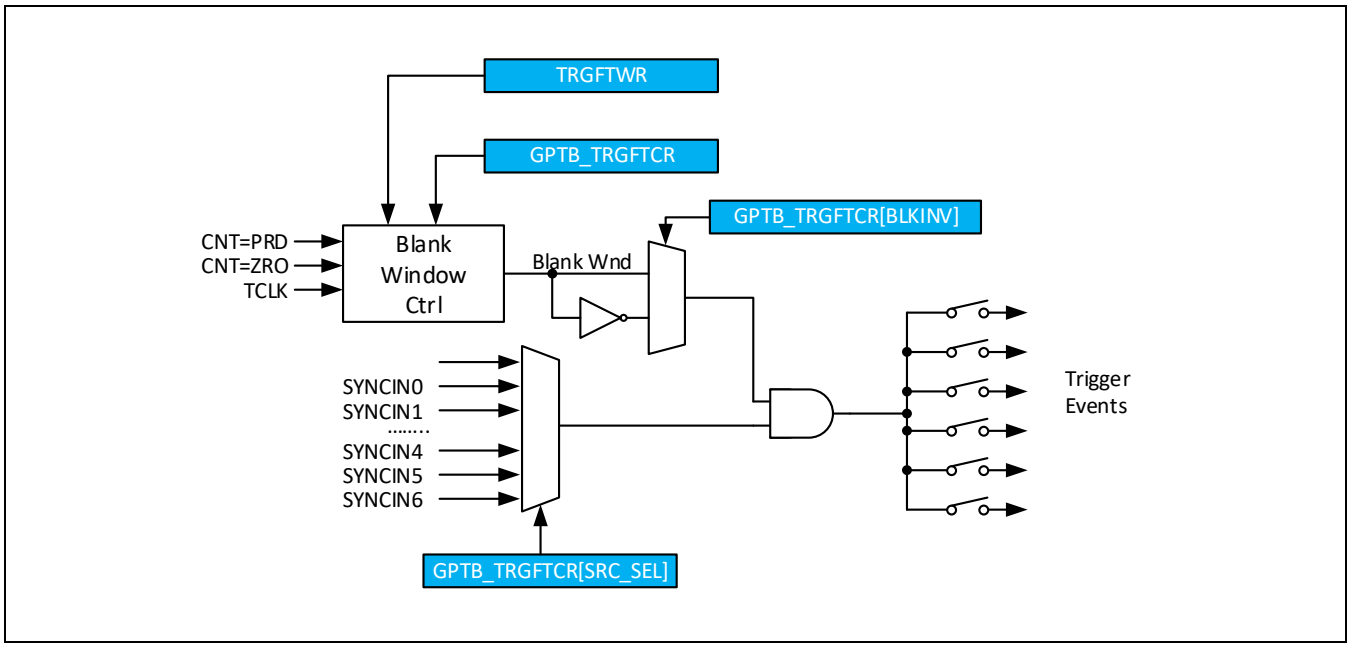


Figure 12-40 滤波器框图

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。

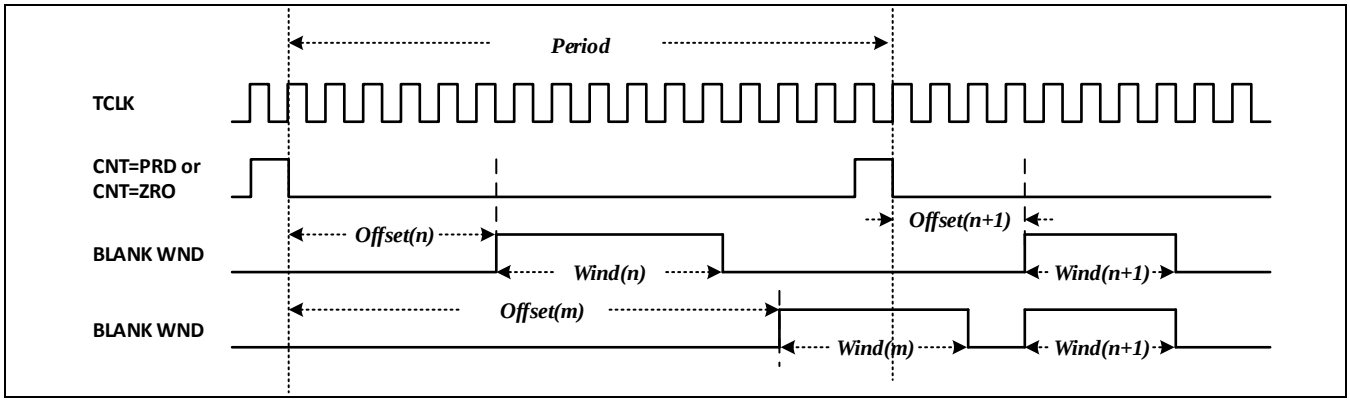


Figure 12-41 滤波器时序

12.3.10 事件触发(输出)

12.3.10.1 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。

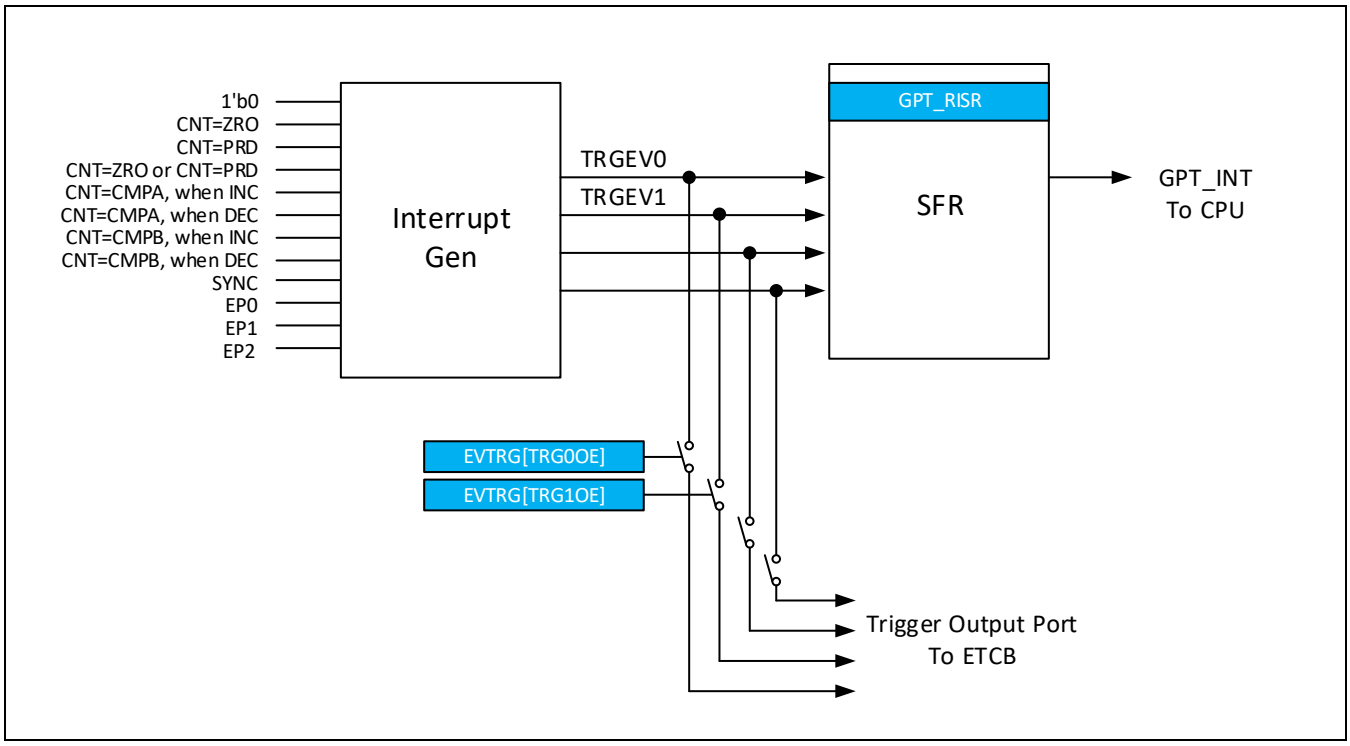


Figure 12-42 同步触发输出

12.3.10.2 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。

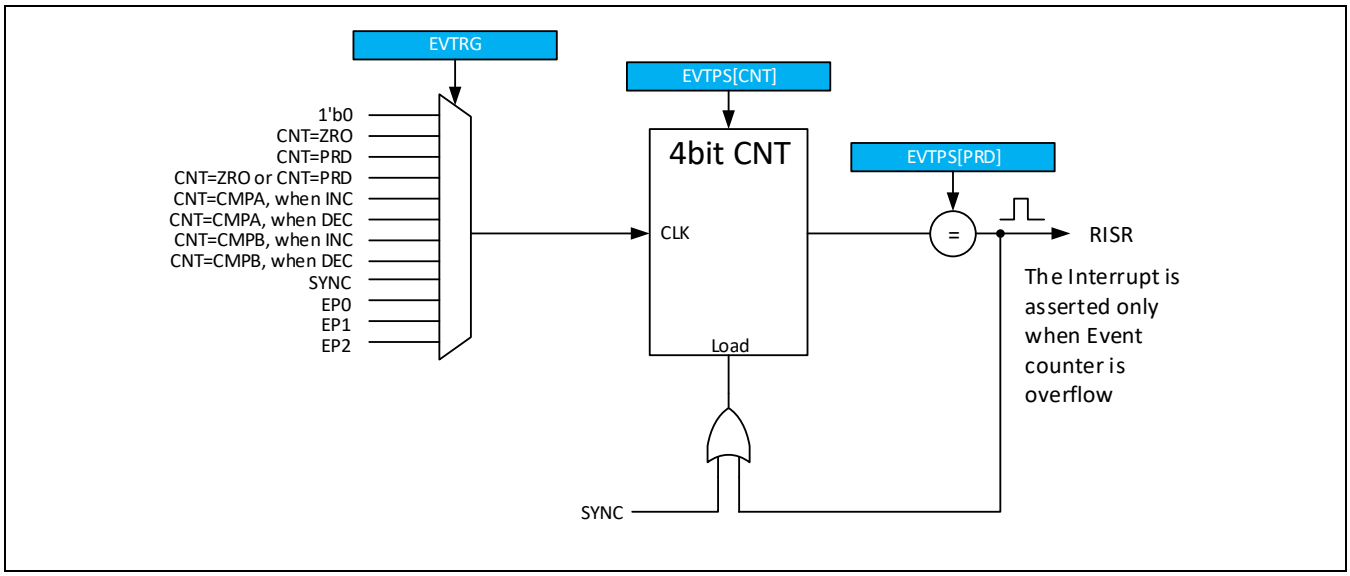


Figure 12-43 事件计数器

12.3.11 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 12-8 寄存器链接对应表

链接值	链接目标
0x0	不链接
0x1	GPTA0
0x2	GPTA1
0x3	GPTA2
0x4	GPTA3
0x5	GPTB0
0x6	GPTB1
0x7	GPTB2
0x8	GPTB3
0x9	GPTB4
0xA	GPTB5

其他值	不链接

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GPTB1.REGLK.PRDR = 0x5, GPTB2.REGLK.PRDR = 0x5时，当程序更新GPTB0.PRDR时，GPTB0、GPTB1和GPTB2的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。

12.4 寄存器说明

12.4.1 寄存器表

Base Address of GPTB0: 0x40059000
Base Address of GPTB1: 0x4005A000
Base Address of GPTB2: 0x4005B000
Base Address of GPTB3: 0x4005C000
Base Address of GPTB4: 0x4005D000
Base Address of GPTB5: 0x4005E000

Register	Offset	Description	Reset Value
GPTB_CEDR	0x0000	ID和时钟控制寄存器	0x6A980000
GPTB_RSSR	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_SYNCR	0x0010	同步控制寄存器	0x00000000
GPTB_GLDCR	0x0014	全局载入控制寄存器	0x00000000
GPTB_GLDCFG	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_EMSRC	0x007C	紧急状态输入控制寄存器	0x00000000
GPTB_EMSRC2	0x0080	紧急状态输入控制寄存器2	0x00000000
GPTB_EMPOL	0x0084	紧急状态输入极性控制寄存器	0x00000000
GPTB_EMECR	0x0088	紧急状态使能控制寄存器	0x00400000
GPTB_EMOSR	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_EMFRCR	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_TRGFTCR	0x00B8	数字比较器滤波窗控制寄存器	0x00000000
GPTB_TRGFTWR	0x00BC	数字比较器滤波窗时序寄存器	0x00000000
GPTB_EVTRG	0x00C0	事件触发选择寄存器	0x00000000
GPTB_EVPS	0x00C4	事件触发计数寄存器	0x00000000
GPTB_EVCNTINIT	0x00C8	事件触发计数器初始化值寄存器	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_CMPAA	0x82C	比较值A active寄存器	0x00000000
GPTB_CMPBA	0x830	比较值B active寄存器	0x00000000

12.4.2 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EPT模块的版本信息。
FLTCKPRS	[15:8]	RW	CGFLT数字滤波器的时钟分频控制。 数字滤波器的时钟频率为PCLK/(FLTCKPRS+1)
SHDWSTP	[6]	RW	START控制位的Shadow功能使能控制。START置位不受此位控制，清除时受此位控制。当选择Shadow模式时，START控制位在周期结束时清除。 0h：Shadow模式 1h：Immediate模式
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。

12.4.3 GPTB_RSSR(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								CNTDIR	RSVD		START
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	R	R	R	R	R	R	R	R	R	R	R	RW

Name	Bit	Type	Description
SRR	[15:12]	W	软件复位控制位。 当对当前控制位写入‘0x5’时，TIMER模块会被复位。复位后，所有寄存器都恢复为RESET状态。
CNTDIR	[3]	R	当前计数器计数方向状态。 0h：当前计数器方向为递增 1h：当前计数器方向为递减
START	[0]	RW	计数器启动控制位。 0h：当写‘0’时，停止计数器 1h：当写‘1’时，启动计数器 当对START位进行读取时，返回当前计数器工作状态 0h：计数器处于IDLE状态 1h：计数器正在工作 当CR[SWSYNEN]控制位为低时，START控制位用于控制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。			

12.4.4 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)

12.4.5 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				LDDRST	LDCRST	LDBRST	LDARST	STOP_WRAP		CAPMD	REARM	CAPMD_SEL	WAVE	PSCLD		CGFLT			CGSRC		FLTIPSCLD	BURST	CAPLDEN	RSVD		PRDLD		RSVD	SWSYNEN	CNTMD	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	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	RW	RW

Name	Bit	Type	Description
LDDRST	[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
LDCRST	[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：CHBX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。
CNTMD	[1:0]	RW	计数模式设置。 计数模式一般只设置一次，并且在计数过程中不做改变。如果计数模式被改变，变化将发生在下一个TCLK的边沿，并且基于上一个计数器值进行递增或者递减。 00b：递增模式 01b：递减模式 （该模式不支持输出满幅波形） 10b：递增递减模式 11b：保留
NOTE: 该寄存器受REGPROT保护，需要先解锁，才能写入。			

12.4.6 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	CNTMD	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	RW	RW	RW	RW	RW	RW	RW	R	R	R	RW	RW	R	RW	RW	RW

Name	Bit	Type	Description
WAVE	[18]	RW	EPT工作模式选择。 0h：捕捉模式 1h：波形发生模式
PSCLD	[17:16]	R	PSCR活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值。 00b：当CNT=ZRO时，Shadow寄存器载入到Active寄存器中 01b：当CNT=PRD时，Shadow寄存器载入到Active寄存器中 10b：当CNT=ZRO或者PRD时，Shadow寄存器载入到Active寄存器中 11b：不进行载入
CGFLT	[15:13]	RW	门控输入数字滤波控制。此控制定义了滤波器监测的步数，只有连续N次监测结果一致时，滤波器才输出有效的电平翻转。滤波器的采样时钟频率在通过CKS控制位定义。 000b：Bypass 001b：N = 2 010b：N = 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：CHBX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。
CNTMD	[1:0]	RW	计数模式设置。 计数模式一般只设置一次，并且在计数过程中不做改变。如果计数模式被改变，变化将发生在下一个TCLK的边沿，并且基于上一个计数器值进行递增或者递减。 00b：递增模式 01b：递减模式 10b：递增递减模式 11b：保留

12.4.7 GPTB_SYNCR(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			TXREARM0		REARM6	REARM5	REARM4	REARM3	REARM2	REARM1	REARM0	OSTMD6	OSTMD5	OSTMD4	OSTMD3	OSTMD2	OSTMD1	OSTMD0	RSVD	SYNCEN6	SYNCEN5	SYNCEN4	SYNCEN3	SYNCEN2	SYNCEN1	SYNCEN0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	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TRGSRC1控制位选择为ExtSync条件时有效。 0h：选择SYNCIN0作为TRGSRC1的ExtSync触发 1h：选择SYNCIN1作为TRGSRC1的ExtSync触发 2h：选择SYNCIN2作为TRGSRC1的ExtSync触发 3h：选择SYNCIN3作为TRGSRC1的ExtSync触发 4h：选择SYNCIN4作为TRGSRC1的ExtSync触发 5h：选择SYNCIN5作为TRGSRC1的ExtSync触发 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TRGSRC0控制位选择为ExtSync条件时有效。 0h：选择SYNCIN0作为TRGSRC0的ExtSync触发 1h：选择SYNCIN1作为TRGSRC0的ExtSync触发 2h：选择SYNCIN2作为TRGSRC0的ExtSync触发 3h：选择SYNCIN3作为TRGSRC0的ExtSync触发 4h：选择SYNCIN4作为TRGSRC0的ExtSync触发 5h：选择SYNCIN5作为TRGSRC0的ExtSync触发 6h：选择SYNCIN6作为TRGSRC0的ExtSync触发 其他：保留
TXREARM0	[23:22]	RW	Tx信号触发SYNCIN0的REARM 0：禁止硬件自动REARM 1：T1发生触发，自动REARM SYNCIN0通道 2：T2发生触发，自动REARM SYNCIN0通道 3：T1或者T2发生触发，自动REARM SYNCIN0通道
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
OSTMD6~OSTMD0	[14:8]	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。
SYNCEN6~SYNCEN0	[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增减一拍触发事件 SYNCIN5：外部COS事件（用于PWM波形输出控制） SYNCIN6：外部COS事件（用于PWM波形输出控制）

12.4.8 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。			

12.4.9 GPTB_GLDCFG(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		AQCRB	AQCRA	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
AQCRB	[9]	RW	AQCRB寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
AQCRA	[8]	RW	AQCRA寄存器Shadow到Active寄存器载入控制。 0：即使GLDEN=1，仍旧使用独立的载入配置 1：当GLDEN=1时，使用全局载入配置
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

12.4.10 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。			

12.4.11 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。			

12.4.12 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			

12.4.13 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。

12.4.14 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。

12.4.15 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]

12.4.16 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。

12.4.17 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

12.4.18 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		T2D		T2U		T1D		T1U		C2D		C2U		C1D		C1U		PRD		ZRO	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
C2SEL	[23:22]	RW	C2比较值数据源选择 0h：CMPA寄存器作为C2的数据源 1h：CMPB寄存器作为C2的数据源
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
T2D	[19:18]	RW	当T2事件发生，且此时计数方向为递减时，在通道PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T2U	[17:16]	RW	当T2事件发生，且此时计数方向为递增时，在通道PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1D	[15:14]	RW	当T1事件发生，且此时计数方向为递减时，在通道PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1U	[13:12]	RW	当T1事件发生，且此时计数方向为递增时，在通道PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C2D	[11:10]	RW	当CNT值等于C2，且此时计数方向为递减时，在通道PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平）

			2h：置位输出（高电平） 3h：反向（翻转）
C2U	[9:8]	RW	当CNT值等于C2，且此时计数方向为递增时，在通道PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1D	[7:6]	RW	当CNT值等于C1，且此时计数方向为递减时，在通道PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1U	[5:4]	RW	当CNT值等于C1，且此时计数方向为递增时，在通道PWM1上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
PRD	[3:2]	RW	当CNT值等于PRDR时，在通道PWM1上做出的波形输出动作定义。 在递增递减模式时，当计数器值等于PRDR时，计数方向为递减模式 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
ZRO	[1:0]	RW	当CNT值等于零时，在通道PWM1上做出的波形输出动作定义。 在递增递减模式时，当计数器值等于零时，计数方向为递增模式 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）

12.4.19 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		T2D		T2U		T1D		T1U		C2D		C2U		C1D		C1U		PRD		ZRO	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
C2SEL	[23:22]	RW	C2比较值数据源选择 0h：CMPA寄存器作为C2的数据源 1h：CMPB寄存器作为C2的数据源
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
T2D	[19:18]	RW	当T2事件发生，且此时计数方向为递减时，在通道PWM2上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T2U	[17:16]	RW	当T2事件发生，且此时计数方向为递增时，在通道PWM2上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1D	[15:14]	RW	当T1事件发生，且此时计数方向为递减时，在通道PWM2上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
T1U	[13:12]	RW	当T1事件发生，且此时计数方向为递增时，在通道PWM2上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C2D	[11:10]	RW	当CNT值等于C2，且此时计数方向为递减时，在通道PWM2上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平）

			2h：置位输出（高电平） 3h：反向（翻转）
C2U	[9:8]	RW	当CNT值等于C2，且此时计数方向为递增时，在通道PWM2上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1D	[7:6]	RW	当CNT值等于C1，且此时计数方向为递减时，在通道PWM2上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
C1U	[5:4]	RW	当CNT值等于C1，且此时计数方向为递增时，在通道PWM2上做出的波形输出动作定义。 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
PRD	[3:2]	RW	当CNT值等于PRDR时，在通道PWM2上做出的波形输出动作定义。 在递增递减模式时，当计数器值等于PRDR时，计数方向为递减模式 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）
ZRO	[1:0]	RW	当CNT值等于零时，在通道PWM2上做出的波形输出动作定义。 在递增递减模式时，当计数器值等于零时，计数方向为递增模式 0h：不动作（过滤该处理事件） 1h：清除输出（低电平） 2h：置位输出（高电平） 3h：反向（翻转）

12.4.20 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。

12.4.21 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

12.4.22 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

12.4.23 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

12.4.24 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)

12.4.25 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

12.4.26 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

12.4.27 GPTB_EMSRC(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																EP3_SEL				EP2_SEL				EP1_SEL				EP0_SEL			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
EP3_SEL	[15:12]	RW	
EP2_SEL	[11:8]	RW	
EP1_SEL	[7:4]	RW	
EP0_SEL	[3:0]	RW	

EPx的输入选择控制。

1h：选择EBI0（GPIO）作为当前EP的输入

2h：选择EBI1（GPIO）作为当前EP的输入

3h：选择EBI2（GPIO）作为当前EP的输入

4h：选择EBI3（GPIO）作为当前EP的输入

5h：选择CMP0作为当前EP的输入

6h：选择CMP1作为当前EP的输入

7h：选择CMP2作为当前EP的输入

8h：选择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。

12.4.28 GPTB_EMSRC2(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_CMP2	ORL1_CMP1	ORL1_CMP0	ORL1_EBI3	ORL1_EBI2	ORL1_EBI1	ORL1_EBI0	FLT_PACE1				FLT_PACE0				RSVD	ORL0_CMP2	ORL0_CMP1	ORL0_CMP0	ORL0_EBI3	ORL0_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	RW	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
ORL1_CMP2~ORL1_CMP0	[22:20]	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
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_PACE1	[15:12]	RW	EP2和EP3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
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_CMP2~ORL0_CMP0	[6:4]	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
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

12.4.29 GPTB_EMPOL(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																									CMP2_POL	CMP1_POL	CMP0_POL	EBI3_POL	EBI2_POL	EBI1_POL	EBI0_POL
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CMP2_POL	[6]	RW	
CMP1_POL	[5]	RW	
CMP0_POL	[4]	RW	
EBI3_POL	[3]	RW	
EBI2_POL	[2]	RW	
EBI1_POL	[1]	RW	
EBI0_POL	[0]	RW	

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。

12.4.30 GPTB_EMECR(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													EP3_LCKMD		EP2_LCKMD		EP1_LCKMD		EP0_LCKMD	
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
EOM_FAULT	[30]	RW	外部晶振错误触发硬锁止控制位。（需要同时使能外部晶振监测功能） 0h：禁止当前事件触发硬锁止 1h：使能当前事件触发硬锁止
MEM_FAULT	[29]	RW	MEM错误触发硬锁止控制位。（需要同时使能SRAM或者Flash校验功能） 0h：禁止当前事件触发硬锁止 1h：使能当前事件触发硬锁止
CPU_FAULT	[28]	RW	CPU错误触发硬锁止控制位。 0h：禁止当前事件触发硬锁止 1h：使能当前事件触发硬锁止
EMASYNC	[26]	RW	EP端口同步设置控制位。 0h：使能同步 1h：禁止同步
SLCLRMD	[25:24]	RW	软锁止清除条件设置。当CNT值等于设置值，且软锁止不再触发时，硬件自动清除软锁止状态和标志位。 00h：CNT = ZRO时，清除软锁止 01h：CNT = PRD时，清除软锁止 10h：CNT = ZRO或CNT = PRD时，清除软锁止 11h：不自动清除软锁止，必须通过软件清除
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
EP3_LCKMD	[7:6]	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
EP2_LCKMD	[5:4]	RW	
EP1_LCKMD	[3:2]	RW	
EP0_LCKMD	[1:0]	RW	

12.4.31 GPTB_EMOSR(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。			

12.4.32 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																												EP3	EP2	EP1	EP0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
EP3~EP0	[3:0]	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

12.4.33 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																												EP3	EP2	EP1	EP0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
EP3~EP0	[3:0]	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

12.4.34 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																									EOM_FAULT	MEM_FAULT	CPU_FAULT	EP3	EP2	EP1	EP0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
EOM_FAULT	[6]	R	EOM FAULT事件触发的硬锁止状态位。 0h：硬锁止未触发 1h：硬锁止已触发
MEM_FAULT	[5]	R	MEM FAULT事件触发的硬锁止状态位。 0h：硬锁止未触发 1h：硬锁止已触发
CPU_FAULT	[4]	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
EP3~EP0	[3:0]	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

12.4.35 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																									EOM_FAULT	MEM_FAULT	CPU_FAULT	EP3	EP2	EP1	EP0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W

Name	Bit	Type	Description
EOM_FAULT	[6]	W	软件清除EOM FAULT事件触发的硬锁止状态位。 0h：对当前控制位写‘0’无效，读取时总返回‘0’ 1h：清除当前标志位
MEM_FAULT	[5]	W	软件清除MEM FAULT事件触发的硬锁止状态位。 0h：对当前控制位写‘0’无效，读取时总返回‘0’ 1h：清除当前标志位
CPU_FAULT	[4]	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
EP3~EP0	[3:0]	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

12.4.36 GPTB_EMFRCR(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_EP3	FRC_EP2	FRC_EP1	FRC_EP0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W

Name	Bit	Type	Description
FRC_EP3~FRC_EP0	[3:0]	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。			

12.4.37 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																					EOM_FAULT	MEM_FAULT	CPU_FAULT	RSVD				EP3	EP2	EP1	EP0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
EOM_FAULT	[10]	R	EOM_FAULT事件触发的异常事件中断原始标志位。 0h：无中断请求发生 1h：中断请求发生
MEM_FAULT	[9]	R	MEM_FAULT事件触发的异常事件中断原始标志位。 0h：无中断请求发生 1h：中断请求发生
CPU_FAULT	[8]	R	CPU_FAULT事件触发的异常事件中断原始标志位。 0h：无中断请求发生 1h：中断请求发生
EP3~EP0	[3:0]	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

12.4.38 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																					EOM_FAULT	MEM_FAULT	CPU_FAULT	RSVD				EP3	EP2	EP1	EP0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
EOM_FAULT	[10]	R	EOM_FAULT事件触发的异常事件中断标志位。 0h：无中断请求发生 1h：中断请求发生
MEM_FAULT	[9]	R	MEM_FAULT事件触发的异常事件中断标志位。 0h：无中断请求发生 1h：中断请求发生
CPU_FAULT	[8]	R	CPU_FAULT事件触发的异常事件中断标志位。 0h：无中断请求发生 1h：中断请求发生
EP3~EP0	[3:0]	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

12.4.39 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																					EOM_FAULT	MEM_FAULT	CPU_FAULT	RSVD				EP3	EP2	EP1	EP0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	R	R	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
EOM_FAULT	[10]	RW	EOM FAULT事件触发的异常中断使能控制。 0h：禁止对CPU发起中断请求 1h：允许对CPU发起中断请求
MEM_FAULT	[9]	RW	MEM FAULT事件触发的异常中断使能控制。 0h：禁止对CPU发起中断请求 1h：允许对CPU发起中断请求
CPU_FAULT	[8]	RW	CPU FAULT事件触发的异常中断使能控制。 0h：禁止对CPU发起中断请求 1h：允许对CPU发起中断请求
EP3~EP0	[3:0]	RW	EPx事件触发的异常中断使能控制。 0h：禁止对CPU发起中断请求 1h：允许对CPU发起中断请求

12.4.40 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																					EOM_FAULT	MEM_FAULT	CPU_FAULT	RSVD				EP3	EP2	EP1	EP0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	R	R	R	R	W	W	W	W

Name	Bit	Type	Description
EOM_FAULT	[10]	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
MEM_FAULT	[9]	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
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
EP3~EP0	[3:0]	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

12.4.41 GPTB_TRGFTCR(数字比较器滤波窗控制寄存器)

Address = Base Address+ 0x00B8, Reset Value = 0x00000000

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

Name	Bit	Type	Description
CROSSMD	[7]	RW	允许滤波窗跨越窗口对齐点。 缺省条件下，当滤波窗在Align条件满足时若任然有效，将跨过窗口对齐点，一直持续到窗口计数器溢出。当禁止跨周期时，在Align条件满足时，窗口计数器将被停止。 0h：禁止对齐点跨窗口 1h：允许对齐点跨窗口
ALIGNMD	[6:5]	RW	窗口对齐模式选择。当对齐模式条件满足时，OFFSET将被重置；但窗口宽度将根据CORSSMD设置进行调整。 0h：CNT=ZRO 1h：CNT=PRD 2h：CNT=PRD or CNT=ZRO 3h：T1事件
BLKINV	[4]	RW	窗口使能反转控制。 0h：窗口不反转，窗口有效区间禁止滤波输入 1h：窗口反转，窗口有效区间使能滤波输入
SRC_SEL	[2:0]	RW	滤波模块的输入信号选择。 0h：禁止滤波 1h：使能SYNCIN0滤波 2h：使能SYNCIN1滤波 3h：使能SYNCIN2滤波 4h：使能SYNCIN3滤波 5h：使能SYNCIN4滤波 6h：使能SYNCIN5滤波 7h：使能SYNCIN6滤波

12.4.42 GPTB_TRGFTWR(数字比较器滤波窗时序寄存器)

Address = Base Address+ 0x00BC, Reset Value = 0x00000000

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

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

12.4.43 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						CNT1INITFRC	CNT0INITFRC	RSVD		TRG1OE	TRG0OE	RSVD		CNT1INITEN	CNT0INITEN	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	W	W	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CNT1INITFRC	[25]	W	TRGEV1CNT软件触发更新 0h：无效 1h：EVCNT1INIT内容更新到EVCNT1中
CNT0INITFRC	[24]	W	TRGEV0CNT软件触发更新 0h：无效 1h：EVCNT0INIT内容更新到EVCNT0中
TRG1OE	[21]	RW	外部触发端口TRGOUT1使能 0h：禁止触发输出 1h：允许触发输出
TRG0OE	[20]	RW	外部触发端口TRGOUT0使能 0h：禁止触发输出 1h：允许触发输出
CNT1INITEN	[17]	RW	TRGEV1CNT寄存器更新模式控制 0h：无效 1h：TRGEV1CNT在发生LOAD事件触发时，或者EV1CNTINITFRC控制位软件写入‘1’时，EV1CNTINIT的内容更新到EV1CNT中。
CNT0INITEN	[16]	RW	TRGEV0CNT寄存器更新模式控制 0h：无效 1h：TRGEV0CNT在发生LOAD事件触发时，或者EV0CNTINITFRC控制位软件写入‘1’时，EV0CNTINIT的内容更新到EV0CNT中。
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

12.4.44 GPTB_EVPS(事件触发计数寄存器)

Address = Base Address+ 0x00C4, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								TRGEV1CNT				TRGEV0CNT				RSVD								TRGEV1PRD				TRGEV0PRD			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
TRGEV1CNT	[23:20]	R	TRGEV1事件计数器值。 读取时，返回当前事件计数器值。
TRGEV0CNT	[19:16]	R	TRGEV0事件计数器值。 读取时，返回当前事件计数器值。
TRGEV1PRD	[7:4]	RW	TRGEV1事件计数的周期设置。 当TRGEV1事件发生次数满足周期时，才产生TRGEV1触发事件
TRGEV0PRD	[3:0]	RW	TRGEV0事件计数的周期设置。 当TRGEV0事件发生次数满足周期时，才产生TRGEV0触发事件

12.4.45 GPTB_EVCNTINIT(事件触发计数器初始化值寄存器)

Address = Base Address+ 0x00C8, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																								CNT1INIT				CNT0INIT			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CNT1INIT	[7:4]	RW	TRGEV1CNT计数器的初始化值设置。 当EVTRG[CNT1INITEN]控制位有效时，CNT1INIT的值将在触发条件满足时(LOAD事件)，或EVTRG[CNT1INITFRC]软件置位时，被载入到TRGEV1CNT寄存器中。
CNT0INIT	[3:0]	RW	TRGEV0CNT计数器的初始化值设置。 当EVTRG[CNT0INITEN]控制位有效时，CNT0INIT的值将在触发条件满足时(LOAD事件)，或EVTRG[CNT0INITFRC]软件置位时，被载入到TRGEV0CNT寄存器中。

12.4.46 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

12.4.47 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													ZRMA	PRMA	PEND	RSVD				CBD	CBU	CAD	CAU	CAP_LD3	CAP_LD2	CAP_LD1	CAP_LD0	RSVD		TRGEV1	TRGEV0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
ZRMA	[18]	R	ZRO事件中断请求原始标志状态
PRMA	[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)
CBD	[11]	R	递减阶段CNT = CMPB中断请求原始标志状态
CBU	[10]	R	递增阶段CNT = CMPB中断请求原始标志状态
CAD	[9]	R	递减阶段CNT = CMPA中断请求原始标志状态
CAU	[8]	R	递增阶段CNT = CMPA中断请求原始标志状态
CAP_LD3	[7]	R	Capture Load to CMPBA中断请求原始标志状态
CAP_LD2	[6]	R	Capture Load to CMPAA中断请求原始标志状态
CAP_LD1	[5]	R	Capture Load to CMPB中断请求原始标志状态
CAP_LD0	[4]	R	Capture Load to CMPA中断请求原始标志状态
TRGEV1	[1]	R	TRGEV1中断请求原始标志状态
TRGEV0	[0]	R	TRGEV0中断请求原始标志状态
原始中断标志表示中断事件发生，通过设置IMCR中相应位，可以允许该中断请求CPU中断。原始中断标志位需 要通过软件清除。 0h：该中断未置位 1h：该中断已置位			

12.4.48 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													ZRMA	PRMA	PEND	RSVD				CBD	CBU	CAD	CAU	CAP_LD3	CAP_LD2	CAP_LD1	CAP_LD0	RSVD		TRGEV1	TRGEV0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
ZRMA	[18]	R	ZRO事件中断请求标志状态
PRMA	[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)
CBD	[11]	R	递减阶段CNT = CMPB中断请求标志状态
CBU	[10]	R	递增阶段CNT = CMPB中断请求标志状态
CAD	[9]	R	递减阶段CNT = CMPA中断请求标志状态
CAU	[8]	R	递增阶段CNT = CMPA中断请求标志状态
CAP_LD3	[7]	R	Capture Load to CMPBA中断请求标志状态
CAP_LD2	[6]	R	Capture Load to CMPAA中断请求标志状态
CAP_LD1	[5]	R	Capture Load to CMPB中断请求标志状态
CAP_LD0	[4]	R	Capture Load to CMPA中断请求标志状态
TRGEV1	[1]	R	TRGEV1中断请求标志状态
TRGEV0	[0]	R	TRGEV0中断请求标志状态
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			

12.4.49 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													ZRMA	PRMA	PEND	RSVD				CBD	CBU	CAD	CAU	CAP_LD3	CAP_LD2	CAP_LD1	CAP_LD0	RSVD		TRGEV1	TRGEV0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW

Name	Bit	Type	Description
ZRMA	[18]	RW	ZRO事件中断使能控制位
PRMA	[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)
CBD	[11]	RW	递减阶段CNT = CMPB中断使能控制位。
CBU	[10]	RW	递增阶段CNT = CMPB中断使能控制位。
CAD	[9]	RW	递减阶段CNT = CMPA中断使能控制位。
CAU	[8]	RW	递增阶段CNT = CMPA中断使能控制位。
CAP_LD3	[7]	RW	Capture Load to 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。			

12.4.50 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													ZRMA	PRMA	PEND	RSVD				CBD	CBU	CAD	CAU	CAP_LD3	CAP_LD2	CAP_LD1	CAP_LD0	RSVD		TRGEV1	TRGEV0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	R	R	R	R	W	W	W	W	W	W	W	W	R	R	W	W

Name	Bit	Type	Description
ZRMA	[18]	W	ZRO事件中断清除
PRMA	[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)
CBD	[11]	W	递减阶段CNT = CMPB中断清除
CBU	[10]	W	递增阶段CNT = CMPB请中断清除
CAD	[9]	W	递减阶段CNT = CMPA中断清除
CAU	[8]	W	递增阶段CNT = CMPA中断清除
CAP_LD3	[7]	W	Capture Load to 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

12.4.51 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:GPTA1
3h:GPTA2
4h:GPTA3
5h:GPTB0
6h:GPTB1
7h:GPTB2
8h:GPTB3
9h:GPTB4
Ah:GPTB5

12.4.52 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:GPTA1
3h:GPTA2
4h:GPTA3
5h:GPTB0
6h:GPTB1
7h:GPTB2
8h:GPTB3
9h:GPTB4
Ah:GPTB5

12.4.53 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。

12.4.54 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。

12.4.55 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。

13 实时时钟计数器（RTC）

13.1 概述

RTC 在所有低功耗模式下均可独立运行，并支持系统唤醒。

实时时钟计数器（RTC）为一个独立的 BCD 编码计数器，提供实时的日历和时间信息，包括星期，年、月、日和小时、分钟、秒。支持可编程的闹钟中断。RTC 具有产生周期性的中断事件的能力。RTC 一旦初始化成功并开始工作，任何复位信号均不能影响其工作，除非重新上电。

RTC 的基本特性：

- 支持万年历功能，自动闰年判定。计时器包括小时、分钟、秒和微秒，BCD 格式计数。
- 二十四小时或者十二小时制可选，支持星期判断。
- 支持多个时钟源，包括外部晶振、内部低速振荡器和内部主振荡器。
- 支持低功耗唤醒功能
- 两个可编程闹钟。
- 支持周期事件触发。

13.2 功能描述

13.2.1 功能框图

RTC 的系统框图如下图所示。

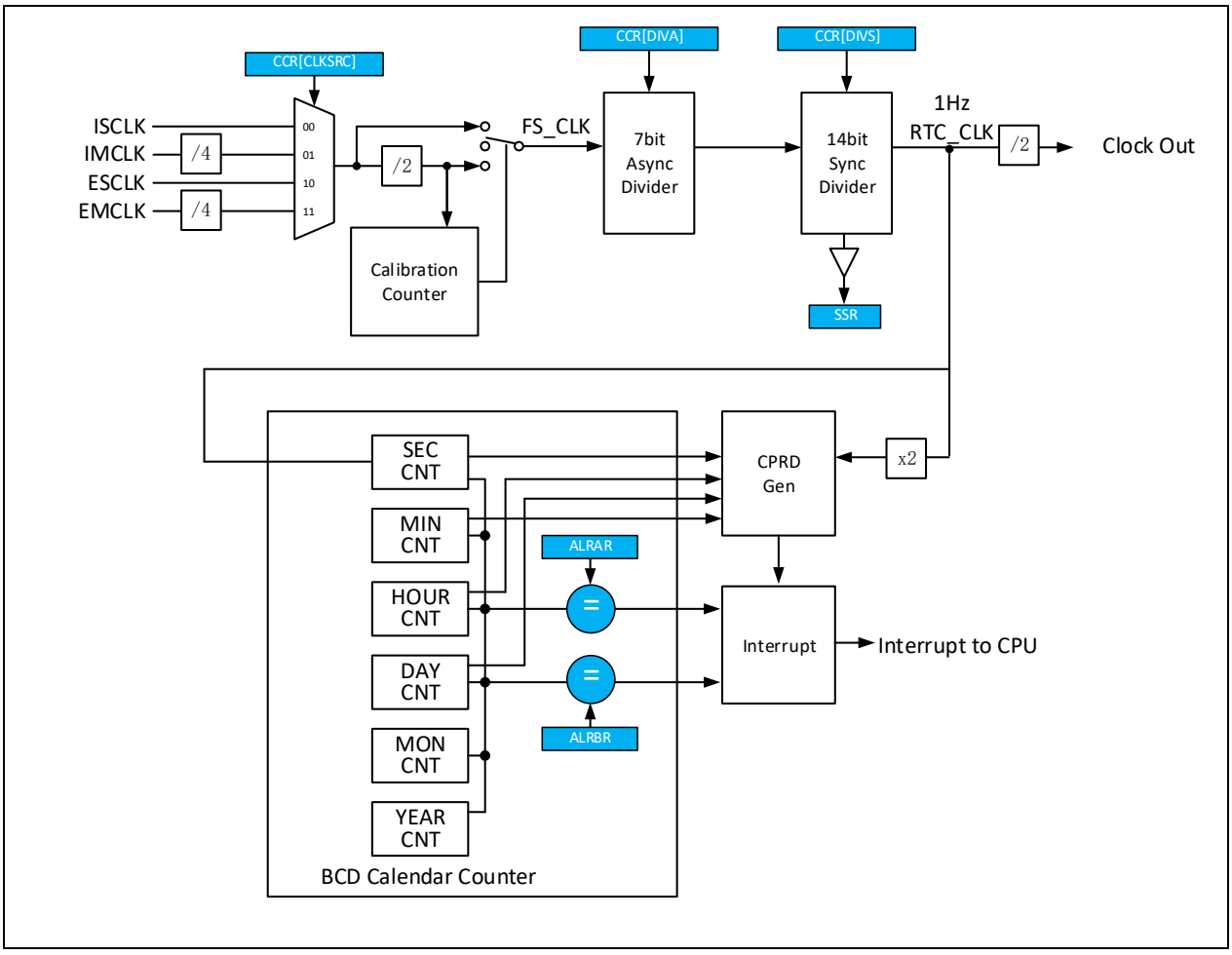


Figure 13-1 功能框图

13.2.2 时钟控制

RTC 的工作时钟（RTC_CLK）可以从多个时钟源中选择，包括低速内部振荡器(ISOSC)、内部主振荡器(IMOSC)的 4 分频、外部晶振（EMOSC）和外部副晶振（ESOSC）。

在分频控制电路中，对输入的时钟源进行预分频处理后再由后续 DIVA 和 DIVS 两级分频器进行分频处理。预分频由 Calibration 控制电路进行时钟使能控制，在 Calibration 禁止时，缺省为输入时钟 DIV2 处理。

经过分频处理后的时钟用于 BCD 计数器的计数控制。为降低 RTC 工作时的功耗，分频器分成两级可配置分频，包括 DIVA 和 DIVS。

- 7 位异步分频器：通过 CCR[DIVA]配置分频系数，分频公式：$F_{async} = F_{FS_CLK} / (DIVA+1)$
- 16 位同步分频器：通过 CCR[DIVS]配置分频系数，分频公式：$F_{RTC_CLK} = F_{async} / (DIVS+1)$

当两级分频器都被配置时，建议尽量将 DIVA 配置为高分频系数，从而最大限度的降低功耗。当 RTC 用

于精确计时，应根据不同的时钟源，通过设置分频系数将 RTC_CLK 的频率设置为 2Hz。参考下表给出的一些例子可以了解具体的时钟分配规则。

Table 13-1 F_{RTC_CLK} = 2Hz (0.5Sec)的时钟配置

CLKSRC	DIVA	Fasync (Hz)	DIVS
ISOSC：27KHz	49	270	134
IMOSC：5.556MHz	124	5556	2777
IMOSC：4.194MHz	124	4194	2096
IMOSC：2.097MHz	124	2097	1047
IMOSC：131.072KHz	63	256	127
EMOSC：(eg:24MHz/4)	99	60000	29999
EMOSC：32.768KHz	63	64	31

RTC 的时钟选择控制通过 RTC_CCR 寄存器进行配置，时钟配置在 RTC 初始化成功后，将不能被改写。需要修改时钟控制，必须在 RTC 停止后进行。RTC_CCR[DBGEN]控制位没有此类限制，通过配置该位决定在调试器挂起系统时，RTC 是否继续工作。

在 RTC 的时钟进行切换时（即修改 CLKSRC），在写入新的 CLKSRC 配置后，需要等待 RTC 工作时钟稳定。该操作可以通过查询 RTC_CCR[RTCCK_ST]控制位进行。在新的 RTC 时钟源未稳定前，对 RTC 的任何操作均无效。

为保证寄存器回读的正确性，必须保证 PCLK 的频率为所选 RTC 时钟源频率的 2 倍以上，例如当 RTC 选择 IMOSC 2.094MHz 为时钟源，则 PCLK 频率不应低于 2.904÷4 x 2 = 1.047MHz。

为保证 RTC 的工作时钟在系统低功耗模式时（DEEP-SLEEP 模式）持续有效，应通过 SYSCON 的 GCER 寄存器配置相应时钟源在低功耗模式下持续工作。在缺省条件下，系统进入 DEEP-SLEEP 模式后，会自动关闭所有的时钟模块。若应用中还需支持 RTC 唤醒 DEEP-SLEEP 模式，还应将 SYSCON 中 WKCR 寄存器中 RTC 唤醒使能控制位打开。

13.2.3 RTC的初始化

RTC 的初始化通过 RTC_CR 的 INIT 控制位进行。在上电复位后，INIT 状态位为高，表示 RTC 没有初始化。为避免程序异常运行对 RTC 模块进行误操作，当 RTC_KEY 寄存器不等于特定 KEY 值时，阻止任何对受保护寄存器对象的更改操作。除中断控制和事件同步触发控制相关寄存器，所有 RTC 的控制寄存器都受 KEY 寄存器保护。

由于 RTC 工作时钟与 APB 总线时钟（PCLK）为异步时钟，对 RTC 的读写操作均需要通过两个异步时钟同步后进行。任何对 RTC 控制器的操作不会立即产生作用，可以通过对 CR[UPD_BSY]控制位进行查询，以获得更新状态。UPD_BSY 控制位在写请求发生时，会立即置位，直到更新结束后才会自动清除。在 RTC 未初始化前，对寄存器的更新值都保存在缓冲中，所以只需要在初始化时，查询 UPD_BSY 即可。当 RTC 已经初始化完成后（RTC 已经开始工作），任何对寄存器的修改，例如 CR 和 ALRAR，都需要查询 UPD_BSY，以保证更新成功。

对 RTC 的初始化，包括如下操作。

1） 设置保护寄存器关闭

2） 设置 TIMR、DATR、ALRAR、ALRBR

3） 设置 CCR，包括时钟选择和分频选择

4） 使能 RTC 时钟（设置 CCR[CLKEN]），等待 RTC 工作时钟稳定（通过 CCR[RTCCK_ST]控制位查询）。

5） 设置 CR 寄存器，以设置时间格式或闹钟使能等配置。同时清除 INIT 位，启动初始化。

6） 回读 INIT 状态位，确认 RTC 工作。

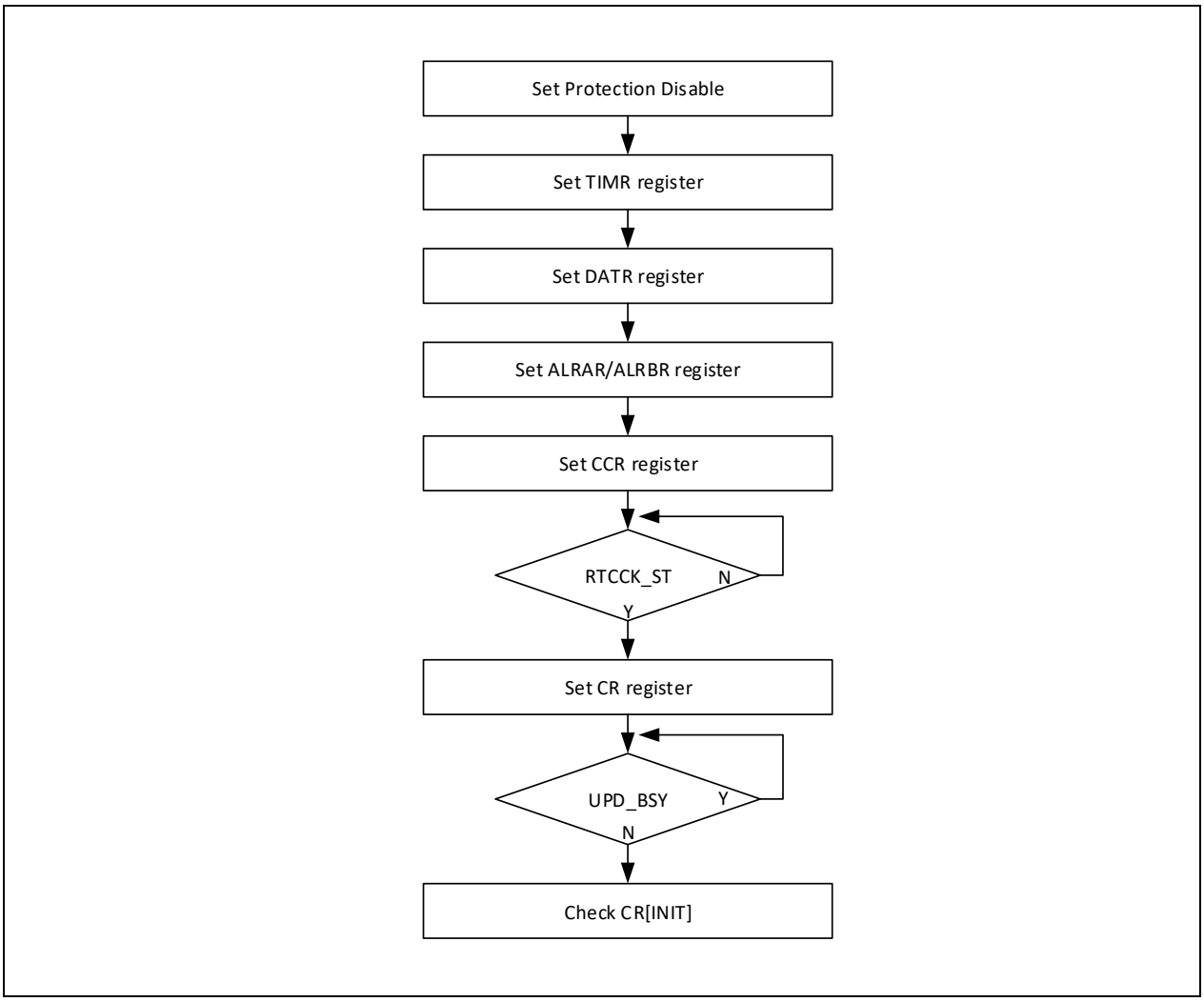


Figure 13-2 初始化流程

TIMR 和 DATR 寄存器在回读关闭时，作为数据更新缓冲使用。当回读使能时，作为回读缓冲使用。回读使能时，读取 TIMR 和 DATR 将返回当前 BCD 计数器的当前值。在 RTC 运行时，不支持从缓冲更新数据到 BCD 计数器。当需要更新 BCD 计数器，必须将 RTC 停止后，更新缓冲后，再次初始化 RTC。停止 RTC 通过置高 INIT 控制位进行操作。

13.2.4 闹钟设置

RTC 支持两路闹钟设置，当 RTC 中的 BCD 计数器的当前计数值满足 ALRAR 或者 ALRBR 的设置值，且 CR 寄存器中的 ALRAE 或者 ALRBE 同时有效，则 RTC 会产生一个闹钟事件。该闹钟事件可以通过系统中断将系统从低功耗模式唤醒，并执行相应中断处理。

闹钟设置寄存器中的 MASK 控制位可以分别控制相应数据段的比较控制使能。当相应 MASK 位有效时，闹钟数据比较时，将自动屏蔽该字段的数据。例如，SMSK 控制位有效时，则屏蔽秒单位的比较，只比较其他的数据段。

闹钟设置的更改支持 RTC 运行中进行，设置步骤如下（通过设置闹钟 A 为例）：

1） 设置保护寄存器关闭。

2） 对 ALRAR 写入新的闹钟配置数据。

3） 清除 CR[ALRAE]控制位，并查询 UPD_BSY 等待更新完成。

4） 使能 CR[ALRAE]控制位，并查询 UPD_BSY 等待更新完成

13.2.5 周期事件触发

RTC 支持产生周期性触发事件，事件的选择通过 CR 寄存器的 CPRD 控制位进行选择。周期事件在 RTC_CLK 为 2Hz 的条件下，支持如下周期事件：每 0.5 秒、每 1 秒、每分钟、每小时、每天、每个月。

13.2.6 时间回读和readback功能

模块内有一个 readback 控制信号，关闭 readback 功能可以减少模块功耗，但是时间回读的时候必须使能该功能。readback 可以通过 RTC_CR[RBEN]开启或关闭。

在 readback 开启时，RTC 计数值可以通过 RTC_TIMR 和 RTC_DATR 读回。但模块工作在 RTCCLK 时钟域，而这些寄存器则连在 APB 总线上，因为这两个时钟通常不是同步的，所以通过这些寄存器读回的值和当前 RTC 实际计数值可能是不同的。RTC 实际计数值只有在下列两种情况下会更新到 RTC_TIMR 和 RTC_DATR：

1、RTCCLK 周期到了

2、RTC_CR[RBEN]从 0 变成 1，且生效后（如果在 free running 模式下需要等 RTC_CR[UPD_BSY]）

13.2.7 中断控制和同步事件

RTC 支持同步事件触发输出到 ETCB 模块，通过 EVTRG 寄存器可以配置最大两个同步触发事件的输出。任意 RTC 的中断均支持同步输出。

同步输出接口支持事件计数和软件触发。当 EVPS[TRGEVxPRD]控制位不为零时，事件计数器会对触发事件进行计数，只有计数值满足 PRD 条件时，才会向 ETCB 输出同步信号。

RTC 的中断原值标志通过 RISR 寄存器进行查询。通过 IMCR 可以使能或者禁止相应中断。对 ICR 进行写入操作可以软件清除 RISR 中的标志位。

13.3 寄存器说明

13.3.1 寄存器表

Base Address of RTC: 0x40060000

Register	Offset	Description	Reset Value
RTC_TIMR	0x000	时间控制寄存器	0x00000000
RTC_DATR	0x004	日期控制寄存器	0x00000000
RTC_CR	0x008	通用控制寄存器	0x00000001
RTC_CCR	0x00C	时钟控制寄存器	0x00000000
RTC_ALRA	0x010	闹钟A设置寄存器	0x00000000
RTC_ALRB	0x014	闹钟B设置寄存器	0x00000000
RTC_SSR	0x018	毫秒计数器	0x00000000
RTC_RISR	0x020	中断原始状态标志寄存器	0x00000000
RTC_IMCR	0x024	中断使能控制寄存器	0x00000000
RTC_MISR	0x028	中断状态标志寄存器	0x00000000
RTC_ICR	0x02C	中断状态清除寄存器	0x00000000
RTC_KEY	0x030	保护寄存器	0x00000000
RTC_EVTRG	0x034	同步事件触发控制寄存器	0x00000000
RTC_EVPS	0x038	同步事件计数控制器	0x00000000
RTC_EVSWF	0x03C	同步事件软件触发寄存器	0x00000000

13.3.2 RTC_TIMR(时间控制寄存器)

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									PM	HORT		HORU				RSVD	MINT			MINU				RSVD	SECT			SECU			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
PM	[22]	RW	AM或PM标志。 0h：AM 或 24小时格式。 1h：PM。
HORT	[21:20]	RW	BCD格式中，小时单位的十位数配置。
HORU	[19:16]	RW	BCD格式中，小时单位的个位数配置。
MINT	[14:12]	RW	BCD格式中，分钟单位的十位数配置。
MINU	[11:8]	RW	BCD格式中，分钟单位的个位数配置。
SECT	[6:4]	RW	BCD格式中，秒单位的十位数配置。
SECU	[3:0]	RW	BCD格式中，秒单位的个位数配置。
该寄存器值只有在下列两种情况下更新 1、RTCCLK周期到了 2、CR[RBEN]从0变成1，且生效后（如果在free running模式下需要等CR[UPD_BSY]）			

13.3.3 RTC_DATR(日期控制寄存器)

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					WKD			YEAT				YEAU				RSVD			MONT	MONU				RSVD		DAYT		DAYU			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
WKD	[26:24]	RW	一周中星期的配置。 0h：RSVD。 1h：星期一。 2h：星期二。 3h：星期三。 4h：星期四。 5h：星期五。 6h：星期六。 7h：星期日。
YEAT	[23:20]	RW	BCD格式中，年单位的十位数配置。
YEAU	[19:16]	RW	BCD格式中，年单位的个位数配置。
MONT	[12]	RW	BCD格式中，月单位的十位数配置。
MONU	[11:8]	RW	BCD格式中，月单位的个位数配置。
DAYT	[5:4]	RW	BCD格式中，日单位的十位数配置。
DAYU	[3:0]	RW	BCD格式中，日单位的个位数配置。
该寄存器值只有在下列两种情况下更新 1、RTCCLK周期到了 2、CR[RBEN]从0变成1，且生效后（如果在free running模式下需要等CR[UPD_BSY]）			

13.3.4 RTC_CR(通用控制寄存器)

Address = Base Address+ 0x008, 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								SWRST				RSVD			RBEN	CPRD			OSEL			RSVD				FMT	ALRBE	ALRAE	RSVD	UPD_BSY	INIT
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R	R	R	R	R	R	R	R	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	RW	RW	RW	R	R	RW

Name	Bit	Type	Description
SWRST	[23:20]	RW	软件复位控制位。 当对当前控制位写入‘0x5’时，RTC模块会被复位。复位后，所有寄存器都恢复为RESET状态。
RBEN	[16]	RW	TIMR和DATR回读控制。只有在回读使能时，RTC的状态可以被更新到TIMR和DATR中。关闭回读可以降低功耗。 0h：回读禁止。 1h：回读使能。
CPRD	[15:13]	RW	周期事件选择。该配置适用RTC_CLK设置为2Hz情况，当RTC_CLK不为2Hz时，需要用户自己根据比例进行换算。 0h：禁止周期事件。 1h：每0.5秒发生一次。 2h：每秒发生一次。 3h：每分钟发生一次（在秒为0的时候）。 4h：每小时发生一次（在秒和分钟为0的时候）。 5h：每天发生一次（在秒、分钟和小时为0的时候）。 6h：每月发生一次（每个月的第一天）。 其他：保留。
OSEL	[12:10]	RW	RTC_ALM管脚输出信号选择。 0h：闹钟A的脉冲输出。 1h：闹钟B的脉冲输出。 2h：CPRD的脉冲输出。 其他：保留。
FMT	[5]	RW	计时模式设置。 0h：24小时制。 1h：12小时制(AM/PM)。
ALRBE	[4]	RW	闹钟B使能控制。 0h：关闭闹钟B。 1h：打开闹钟B。
ALRAE	[3]	RW	闹钟A使能控制。 0h：关闭闹钟A。 1h：打开闹钟A。
UPD_BSY	[1]	R	更新状态查询寄存器。当RTC处于Free Running模式下，对于CR寄存器的修改，需要查询UPD_BSY以确认更新完成。 0h：更新完成。

			1h：更新正在进行。
INIT	[0]	RW	初始化控制位。 0h：设置RTC到Free Running模式。在此模式下，TIMR和DATR不能修改。 1h：设置RTC到初始化模式。该模式下，RTC停止工作，TIMR和DATR可以通过软件修改。

13.3.5 RTC_CCR(时钟控制寄存器)

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				CLKEN	RTCCK_ST	CLKSRC		DBGEN	DIVA							RSVD	DIVS														
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CLKEN	[27]	RW	时钟使能控制。 0h：RTC_CLK关闭。 1h：RTC_CLK打开。
RTCCK_ST	[26]	RW	时钟稳定状态查询。 0h：RTC_CLK时钟未稳定。 1h：RTC_CLK时钟已稳定。
CLKSRC	[25:24]	RW	RTC时钟源选择。 0h：ISOSC。 1h：IMOSC/4。 2h：ESOSC。 3h：EMOSC/4。
DBGEN	[23]	RW	RTC在调试挂起时是否继续工作。 0h：保持RTC一直工作。 1h：调试挂起时，RTC自动暂停。
DIVA	[22:16]	RW	异步分频器分频设置。
DIVS	[14:0]	RW	同步分频器分频设置。

13.3.6 RTC_ALRA(闹钟A设置寄存器)

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
DMSK	WDSEL	DAYT		DAYU				HMSK	PM	HORT		HORU				MMSK	MINT			MINU				SMSK	SECT			SECU			
0	0	0	0	0	0	0	0	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
DMSK	[31]	RW	屏蔽日的闹钟控制。 0h：打开日单位的比较。 1h：关闭日单位的比较。
WDSEL	[30]	RW	星期/日闹钟选择控制。 0h：根据DAYT/DAYU的设置日期比较闹钟。 1h：根据DAYU设置的星期比较闹钟。
DAYT	[29:28]	RW	BCD格式中，天的十位数配置。
DAYU	[27:24]	RW	BCD格式中，天的个位数配置。
HMSK	[23]	RW	屏蔽小时闹钟控制。 0h：打开小时单位的比较。 1h：关闭小时单位的比较。
PM	[22]	RW	AM/PM控制。24小时制下，该位控制无效。 0h：AM。 1h：PM。
HORT	[21:20]	RW	BCD格式中，小时单位的十位数配置。
HORU	[19:16]	RW	BCD格式中，小时单位的个位数配置。
MMSK	[15]	RW	屏蔽分钟闹钟控制。 0h：打开分钟单位的比较。 1h：关闭分钟单位的比较。
MINT	[14:12]	RW	BCD格式中，分钟单位的十位数配置。
MINU	[11:8]	RW	BCD格式中，分钟单位的个位数配置。
SMSK	[7]	RW	屏蔽秒闹钟控制。 0h：打开秒单位的比较。 1h：关闭秒单位的比较。
SECT	[6:4]	RW	BCD格式中，秒单位的十位数配置。
SECU	[3:0]	RW	BCD格式中，秒单位的个位数配置。

13.3.7 RTC_ALRB(闹钟B设置寄存器)

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
DMSK	WDSEL	DAYT		DAYU				HMSK	PM	HORT		HORU				MMSK	MINT			MINU				SMSK	SECT			SECU			
0	0	0	0	0	0	0	0	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
DMSK	[31]	RW	屏蔽日的闹钟控制。 0h：打开日单位的比较。 1h：关闭日单位的比较。
WDSEL	[30]	RW	星期/日闹钟选择控制。 0h：根据DAYT/DAYU的设置日期比较闹钟。 1h：根据DAYU设置的星期比较闹钟。
DAYT	[29:28]	RW	BCD格式中，天的十位数配置。
DAYU	[27:24]	RW	BCD格式中，天的个位数配置。
HMSK	[23]	RW	屏蔽小时闹钟控制。 0h：打开小时单位的比较。 1h：关闭小时单位的比较。
PM	[22]	RW	AM/PM控制。24小时制下，该位控制无效。 0h：AM。 1h：PM。
HORT	[21:20]	RW	BCD格式中，小时单位的十位数配置。
HORU	[19:16]	RW	BCD格式中，小时单位的个位数配置。
MMSK	[15]	RW	屏蔽分钟闹钟控制。 0h：打开分钟单位的比较。 1h：关闭分钟单位的比较。
MINT	[14:12]	RW	BCD格式中，分钟单位的十位数配置。
MINU	[11:8]	RW	BCD格式中，分钟单位的个位数配置。
SMSK	[7]	RW	屏蔽秒闹钟控制。 0h：打开秒单位的比较。 1h：关闭秒单位的比较。
SECT	[6:4]	RW	BCD格式中，秒单位的十位数配置。
SECU	[3:0]	RW	BCD格式中，秒单位的个位数配置。

13.3.8 RTC_SSR(毫秒计数器)

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															SSR																
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
SSR	[16:0]	RW	毫秒计数器值。返回当前RTC的毫秒单位计数值。 Bit[16:7]为毫秒值。bit[6:0]为微秒值，每个Step为8uS。 以上计时均在RTC_CLK为2Hz条件下获得。

13.3.9 RTC_RISR(中断原始状态标志寄存器)

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																											TRGEV1	TRGEV0	CPRD	ALRB	ALRA
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
TRGEV1	[4]	R	同步触发事件1的原始中断标志位。
TRGEV0	[3]	R	同步触发事件0的原始中断标志位。
CPRD	[2]	R	CPRD的原始中断标志位。
ALRB	[1]	R	闹钟B的原始中断标志位。
ALRA	[0]	R	闹钟A的原始中断标志位。

13.3.10 RTC_IMCR(中断使能控制寄存器)

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																											TRGEV1	TRGEV0	CPRD	ALRB	ALRA
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	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
TRGEV1	[4]	RW	同步触发事件1的中断使能控制。 0h：关闭中断。 1h：打开中断。
TRGEV0	[3]	RW	同步触发事件0的中断使能控制。 0h：关闭中断。 1h：打开中断。
CPRD	[2]	RW	CPRD的中断使能控制。 0h：关闭中断。 1h：打开中断。
ALRB	[1]	RW	闹钟B的中断使能控制。 0h：关闭中断。 1h：打开中断。
ALRA	[0]	RW	闹钟A的中断使能控制。 0h：关闭中断。 1h：打开中断。

13.3.11 RTC_MISR(中断状态标志寄存器)

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																											TRGEV1	TRGEV0	CPRD	ALRB	ALRA
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
TRGEV1	[4]	R	同步触发事件1的中断标志位。
TRGEV0	[3]	R	同步触发事件0的中断标志位。
CPRD	[2]	R	CPRD的中断标志位。
ALRB	[1]	R	闹钟B的中断标志位。
ALRA	[0]	R	闹钟A的中断标志位。

13.3.12 RTC_ICR(中断状态清除寄存器)

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																											TRGEV1	TRGEV0	CPRD	ALRB	ALRA
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	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
TRGEV1	[4]	W	清除同步触发事件1的原始中断标志位。
TRGEV0	[3]	W	清除同步触发事件0的原始中断标志位。
CPRD	[2]	W	清除CPRD的原始中断标志位。
ALRB	[1]	W	清除闹钟B的原始中断标志位。
ALRA	[0]	W	清除闹钟A的原始中断标志位。

13.3.13 RTC_KEY(保护寄存器)

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																RTCKEY															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
RTCKEY	[15:0]	RW	RTC的保护寄存器。 只有对当前控制位写入0xCA53后，才能对具有保护功能的寄存器进行修改。

13.3.14 RTC_EVTRG(同步事件触发控制寄存器)

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										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。 0h：禁止TRGSRC触发输出 1h：AlarmA事件产生ESYNC事件 2h：AlarmB事件产生ESYNC事件 3h：AlarmA或AlarmB事件产生ESYNC事件 4h：CPRD事件产生ESYNC事件 其他：保留 注意：Sleep模式下如果PCLK关闭，trigger事件不能用作唤醒源。DeepSleep模式，Snoose和ShutDowm模式Trigger事件也不能用作唤醒源。
TRGSEL0	[3:0]	RW	TRGEV0事件的触发源选择。 0h：禁止TRGSRC触发输出 1h：AlarmA事件产生ESYNC事件 2h：AlarmB事件产生ESYNC事件 3h：AlarmA或AlarmB事件产生ESYNC事件 4h：CPRD事件产生ESYNC事件 其他：保留 注意：Sleep模式下如果PCLK关闭，trigger事件不能用作唤醒源。DeepSleep模式，Snoose和ShutDowm模式Trigger事件也不能用作唤醒源。

13.3.15 RTC_EVPS(同步事件计数控制器)

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								TRGEV1CNT				TRGEV0CNT				RSVD								TRGEV1PRD				TRGEV0PRD			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

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

13.3.16 RTC_EVSWF(同步事件软件触发寄存器)

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																														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

14 低功耗定时器 (LPT)

14.1 概述

低功耗定时器（Low Power Timer）作为 MCU 的低功耗外设，可以支持异步时钟计数操作，从而实现超低功耗下计数。由于可以基于外部 CLOCK 或独立的时钟源计数，LPT 可以支持在低功耗模式下将系统唤醒。当使用外部时钟计数时，LPT 可以作为外部脉冲计数使用。LPT 内部包含一个 16 位的定时/计数模块，支持 PWM 输出。

14.1.1 主要特性

- 16 位递增计数器
- 4 Bit 预分频控制，支持（1，2，4，8，16，32，64，128，256，512，1024，2048，4096 分频）
- 支持多种计数时钟：
  - 内部时钟：ISCLK, IMCLK, EMCLK, ESCLK 或 PCLK
  - 外部时钟：来自 GPIO 触发模块的 TBSTI（当没有内部时钟时，可以作为脉冲计数）
  - 一路独立的 PWM 输出
- 一个比较值寄存器
- 支持连续或单次计数模式
- 支持通过 ET 触发
- 支持脉冲和 PWM 输出模式

14.1.2 管脚描述

GPIO 上相关 AF 功能映射如下。

Table 14-1 LPT 功能管脚描述

管脚名称	IO 方向	功能描述
LPT_OUT	输出	LPT 的波形输出
LPT_IN	输入	LPT 的时钟输入

14.2 功能描述

14.2.1 模块框图

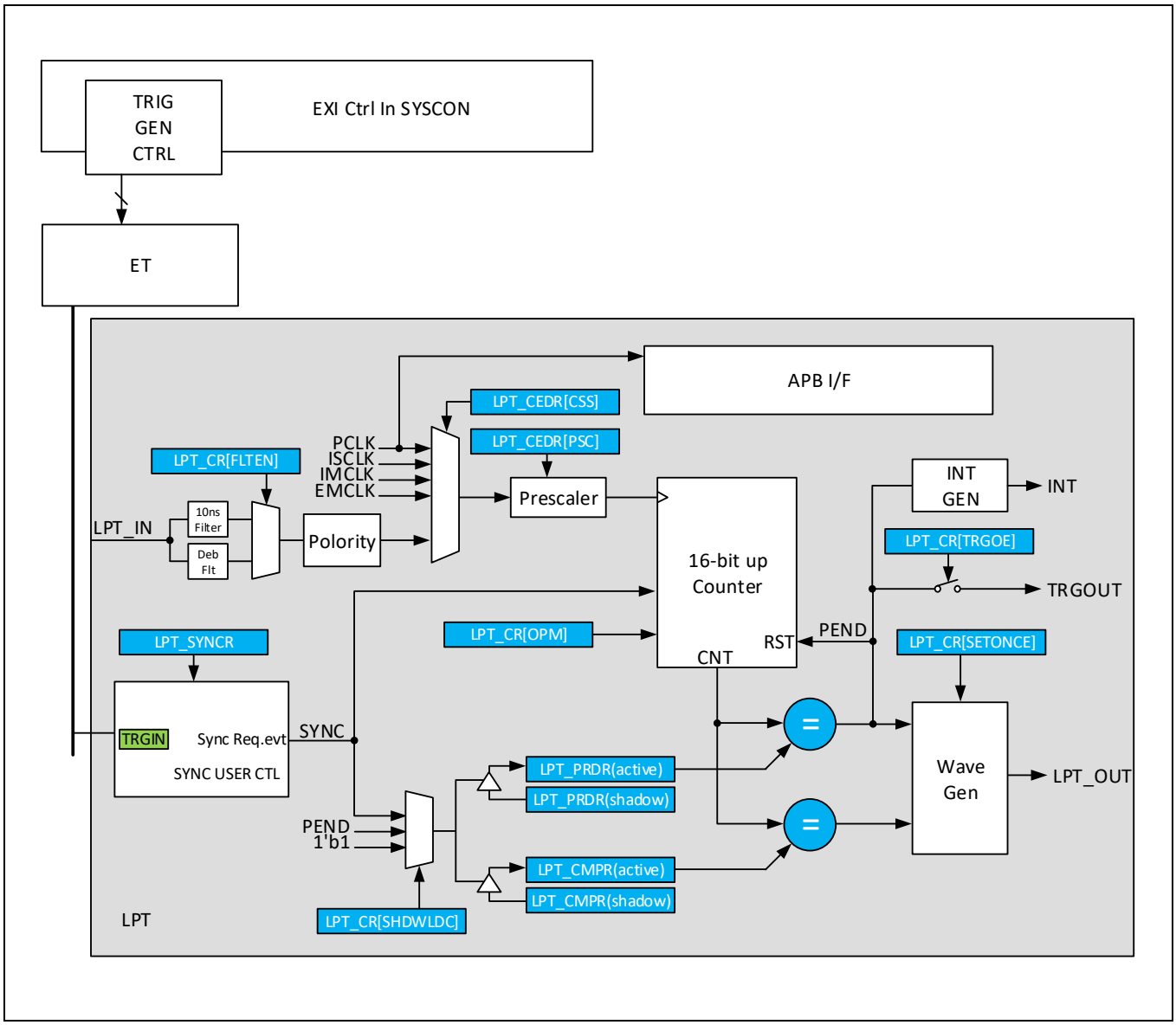


Figure 14-1 模块结构示意图

14.3 基本功能描述

LPT模块是一个简单的计数器，主要目的为低功耗计时。LPT模块的时钟源可以选择ISCLK, IMCLK, EMCLK, ESCLK或PCLK/4，同时还可以选择外部管脚LPT_IN作为时钟输入，实现对外部事件的计数功能。LPT模块还有一个输出LPT_OUT，可以输出简单的PWM波形。LPT_OUT输出在计数值等于比较值(LPT_CMP)和周期值(LPT_PRDR)时会翻转。LPT_OUT输出的初始值可以通过LPT_CR寄存器的输出极性位POL控制。

14.3.1 时钟源

14.3.1.1 概述

LPT计数器计数时钟可以选择PCLK的4分频，系统ISCLK，IMCLK，EMCLK，ESCLK或者由外部管脚提供，计数器在外部计数器时钟控制下计数时，外部计数时钟经过LPT模块内的控制和滤波，产生相应的触发脉冲，该脉冲作为LPT的时基计数器外部触发源，触发计数器递增计数。

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

14.3.1.2 外部时钟

外部时钟的选择和极性控制，由LPT_CEDR的CSS位控制。在外部时钟模式下，外部时钟通路上集成有数字滤波器。数字滤波器可以在外部时钟有较大干扰时打开。

在外部时钟输入通道上，可以通过设置LPT_CR的FLTEN位使能数字滤波。数字滤波在连续检测到N个一致结果时，才会确认输出翻转，否则将保持当前输出状态。如下图所示。

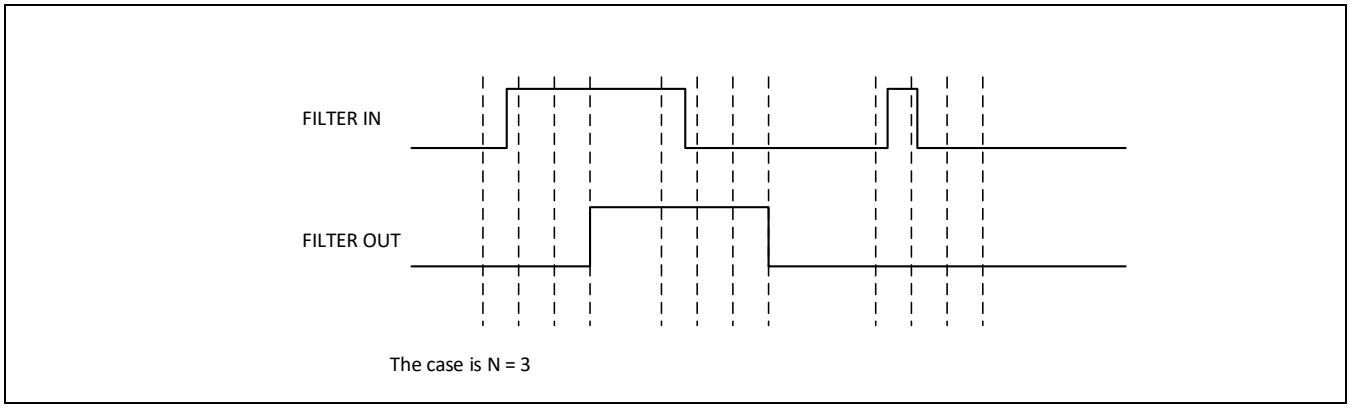


Figure 14-2 外部时钟输入通道的数字滤波器原理

14.3.1.3 内部时钟

LPT的内部时钟可以通过LPT_CEDR的CSS位选择PCLK的4分频，或者系统ISCLK，IMCLK，EMCLK, ESCLK。当选择低频时钟时，可以实现超低功耗的计时，用来对系统进行定时唤醒。

14.3.2 时基控制

14.3.2.1 概述

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

- 确定时基计数器（LPT_CNT）的频率，或者控制事件触发的周期。
- 根据计数器值产生事件触发PWM输出

时基模块的寄存器包括：

- 计数器寄存器（LPT_CNT）：在每个计数时钟周期增加
- 周期寄存器（LPT_PRDR）：计数器周期控制寄存器。

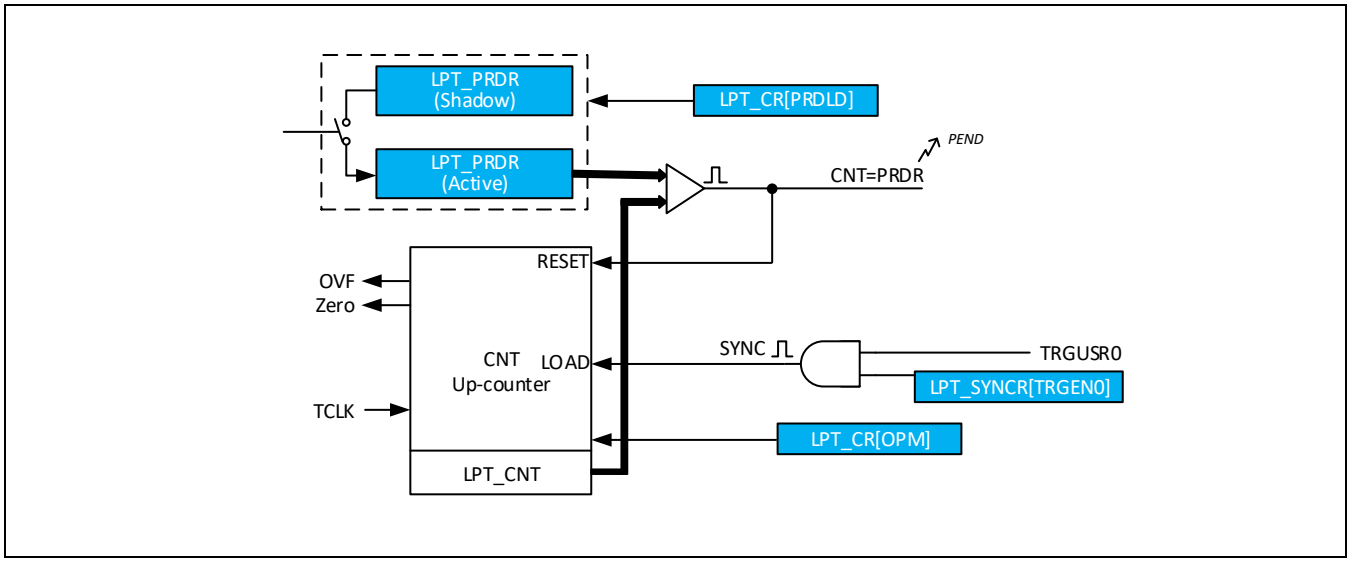


Figure 14-3 计数器时基模块

计数器的计数周期由周期寄存器（LPT_PRDR）的设置值决定。计数器只支持一种计数模式：

- 递增模式（Up-Counting Mode）：

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

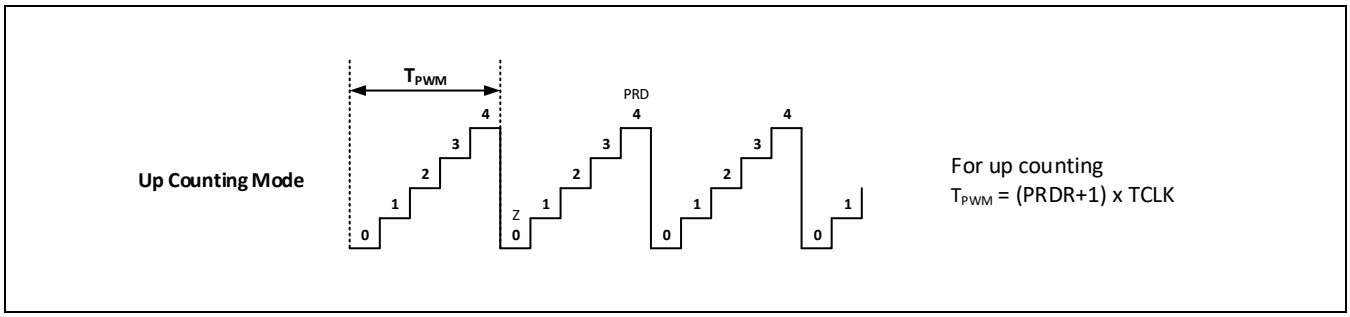


Figure 14-4 计数器工作模式

14.3.2.2 计数器重置和周期设置

在下列条件满足时，计数器值将会被重置。重置发生时，计数器将被重置为PRDR的设置值或者用户指定值。

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

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

- **PRDR寄存器的Shadow模式**

  PRDR的缓冲（Shadow Register）在LPT_CR[PRDLD]控制位等于1的时候有效。在此配置下，CPU对PRDR的读写操作对象为PRDR的影子寄存器。当时基计数器值等于PRDR时，或者SYNC触发时，影子寄存器的值被硬件自动载入到活动寄存器（Active Register）中。在缺省配置下，只有时基计数器值等于PRDR时，自动载入才会发生，用户可以通过配置LPT_CR[PRDLD]控制位进行修改。

- **PRDR寄存器的立即加载模式**

  在立即加载模式下（LPT_CR[PRDLD]=0），CPU对PRDR的读写操作对象是PRDR的活动寄存器。任何对PRDR的更新操作将被直接反应到活动寄存器中。

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

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

14.3.3 计数器数值比较控制

14.3.3.1 概述

计数器数值比较控制模块实时比较当前计数器的计数值和比较值寄存器（CMP）或者周期寄存器（PRDR）的值，当计数值相等时，比较控制模块将产生一个相应的事件触发。主要特性如下：

- 支持的触发事件和触发条件：
  - CNT = CMP： 时基计数器当前值等于计数器比较值寄存器的值
  - CNT = PRDR：时基计数器当前值等于周期寄存器PRDR寄存器的值
- PWM的波形控制基于CMP和PRDR，数值相等时PWM输出翻转
- 比较值寄存器和周期值寄存器都具有影子寄存器功能，以防止PWM输出产生毛刺

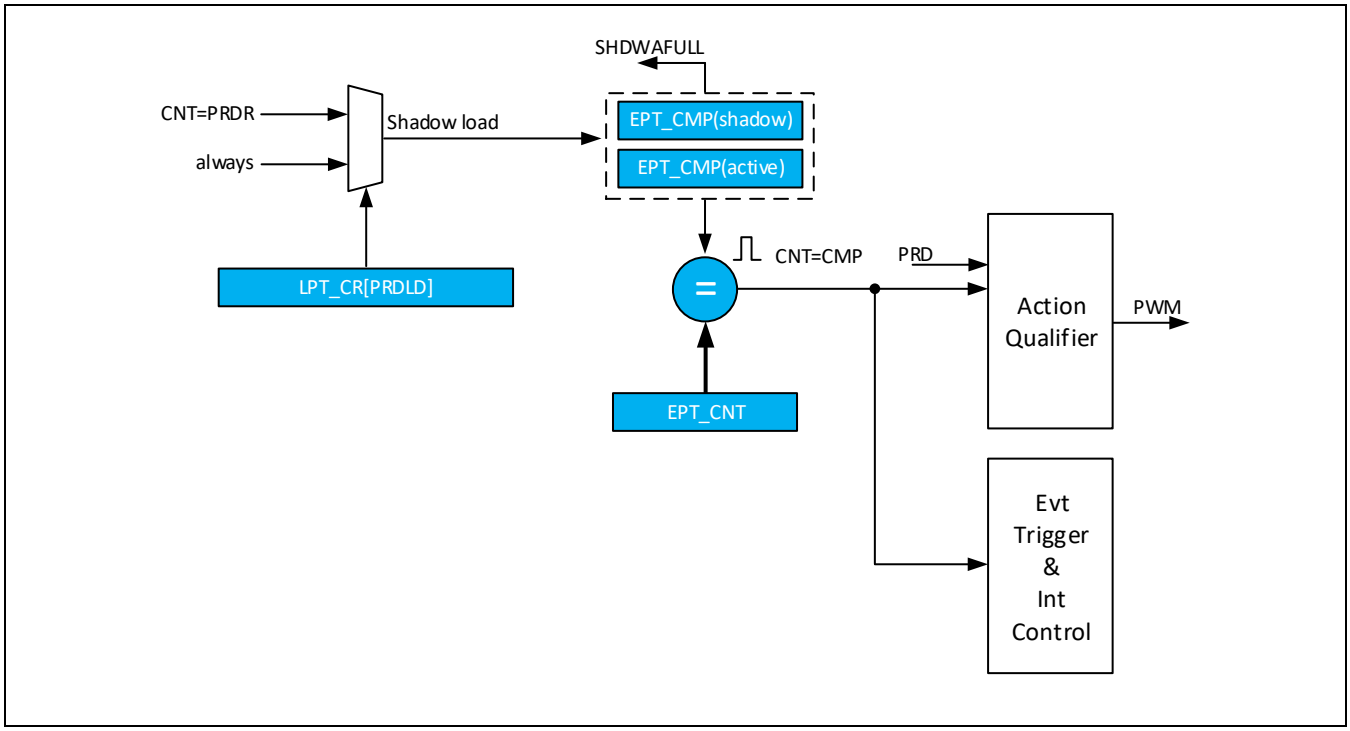


Figure 14-5 计数器值比较控制

14.3.3.2 比较值寄存器载入方式

CMP有相应的Shadow寄存器，在缺省设置下，所有对CMP寄存器的读写对象都是影子寄存器。影子寄存器的使能可以通过LPT_CR[CMPLD]控制位进行设置。当Shadow模式被禁止时，所有对CMP寄存器的操作将直接作用到内部活动寄存器上。

- **CMP寄存器的Shadow模式**

  当Shadow模式使能时，Shadow寄存器中的内容将在CNT=PRDR周期寄存器的值时，被自动传送到活动寄存中。

- **CMPx寄存器的立即加载模式**

  在立即加载模式下，对CMP的操作直接影响活动寄存器。

14.3.4 单次触发模式

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

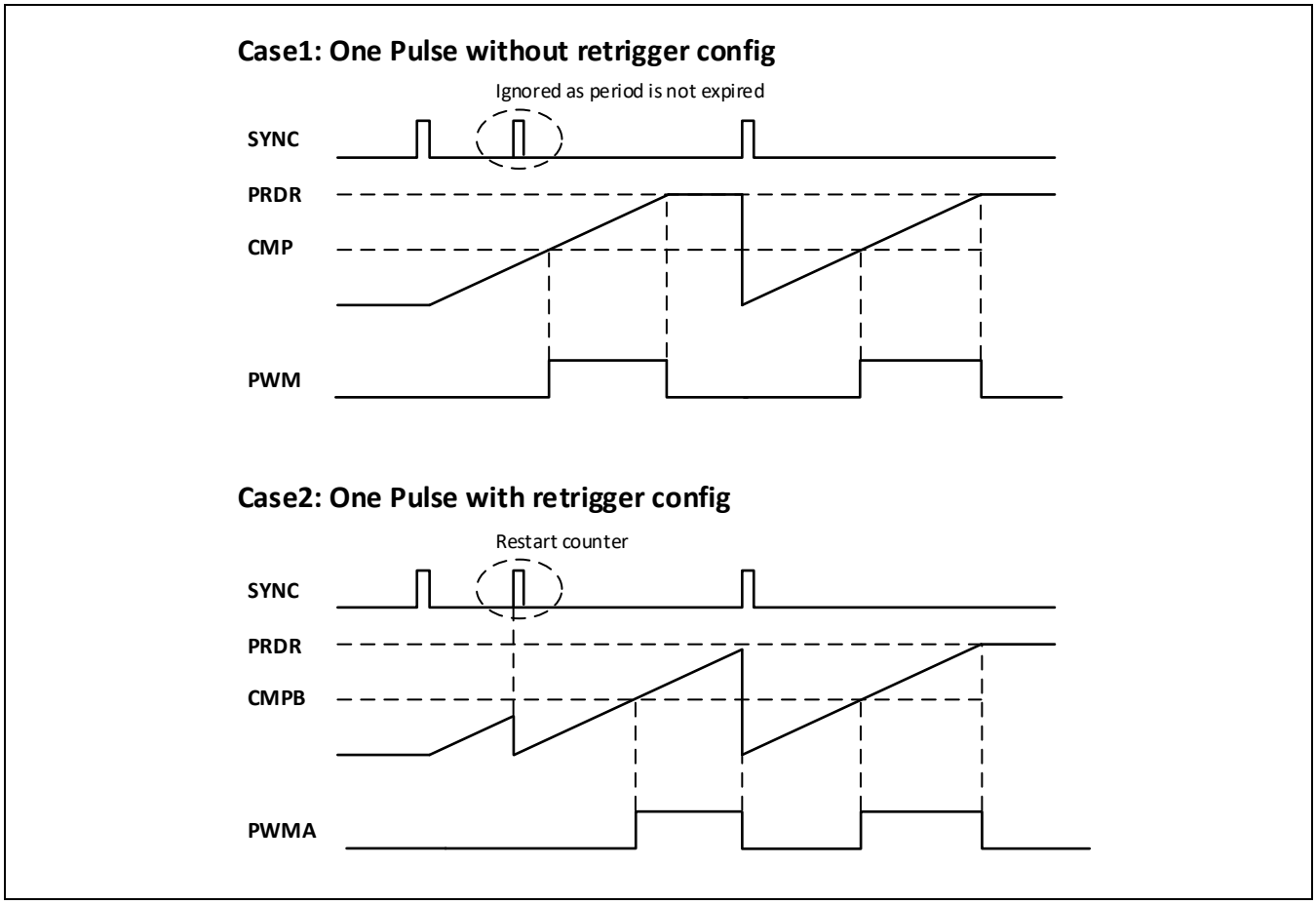


Figure 14-6 单次触发

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

14.3.5 同步触发

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

14.3.5.1 同步触发输入接口

LPT支持模块间的同步触发功能，可以支持的触发功能只有：

- 重置和启动计数器

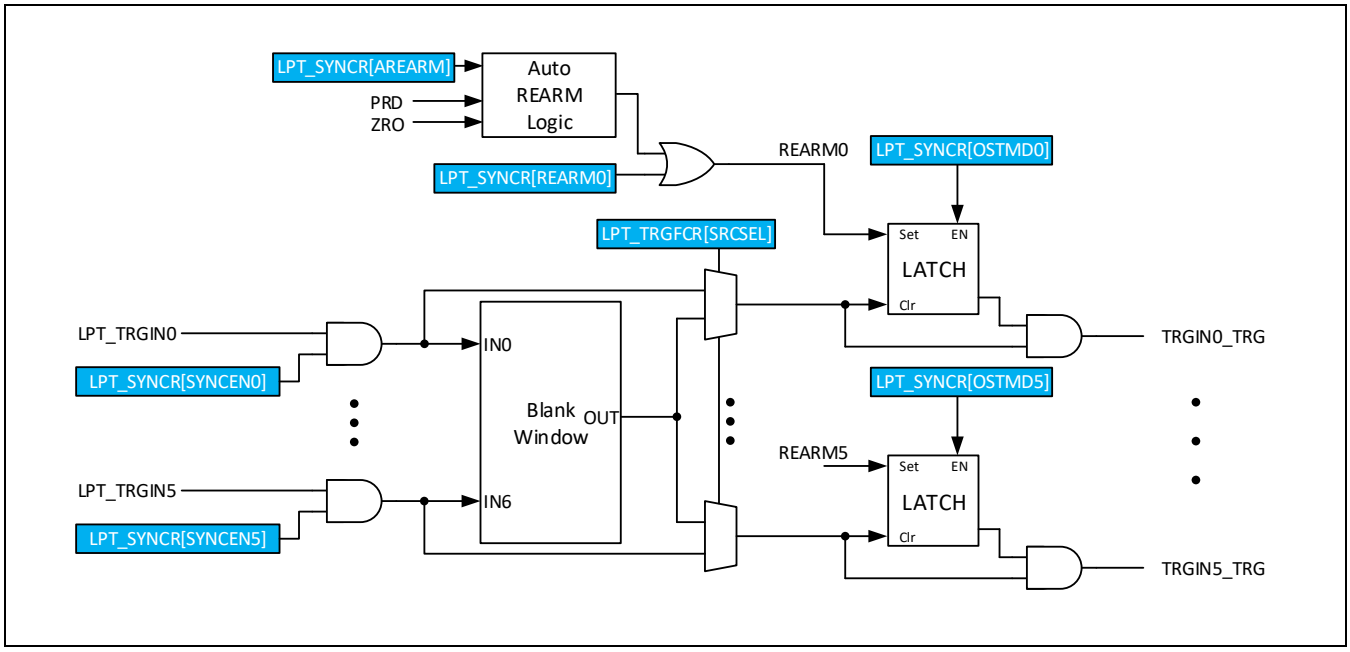


Figure 14-7 同步触发输入 (当前LPT只支持一个源LPT_SYNCIN0)

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

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

重置和启动计数器（SYNCIN0）

当该端口被触发，下列动作将被同时执行：

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

14.3.5.2 事件滤波器

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

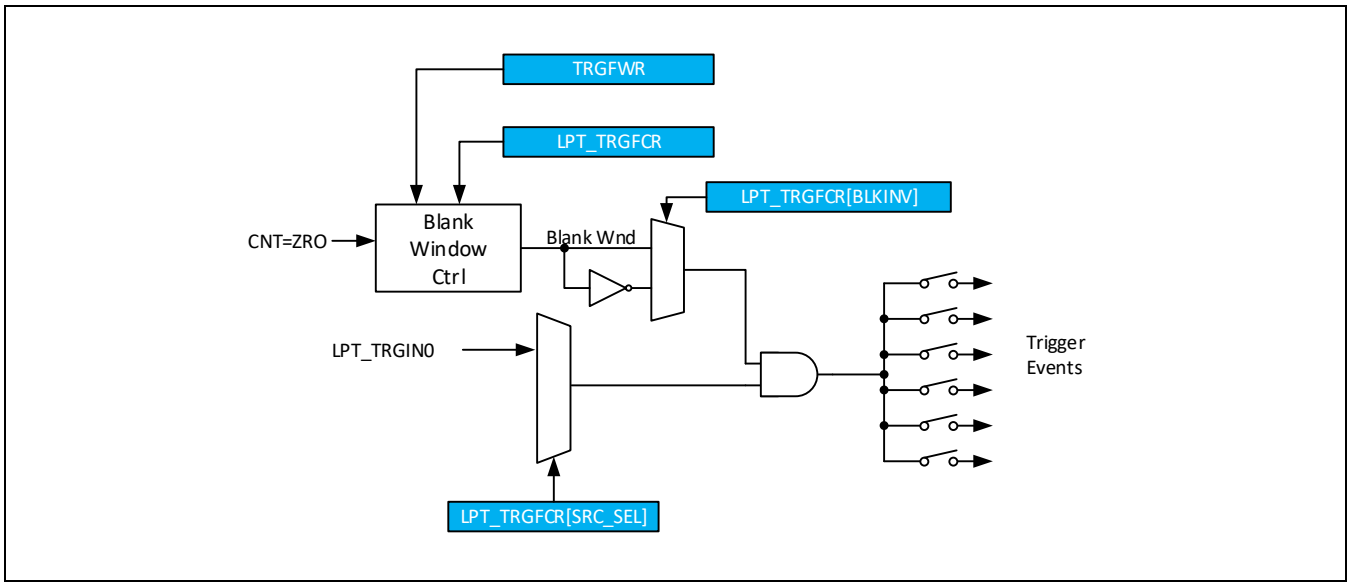


Figure 14-8 滤波器框图

如果使能窗逻辑被打开，在使能窗口内，或者窗口外，将禁止输入事件通过。窗口的激活时间可以为CNT=ZRO。窗口的延时和宽度可以通过TRGFWR进行设置。

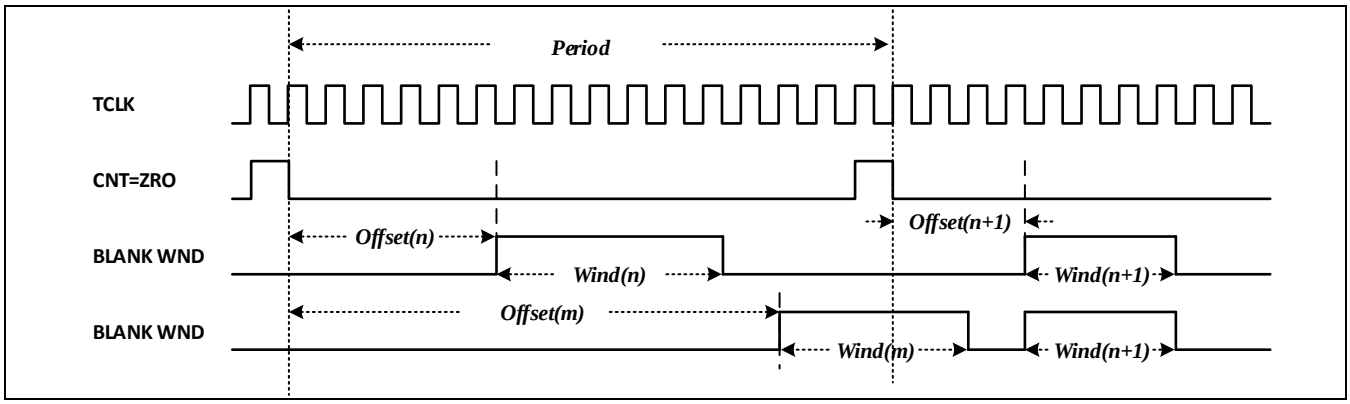


Figure 14-9 滤波器时序

14.3.5.3 同步触发输出接口

同步触发输出接口支持4路事件触发输出，每个LPT中断对应一个事件输出端口。可以通过LPT_EVTRG控制寄存器选择LPT中的任意一个事件作为每个中断的触发信号，同时中断触发信号可以通过ESYNxOE控制位使能输出到ET (LPT_SYNC源)，并且发送给其他外设作为触发信号。

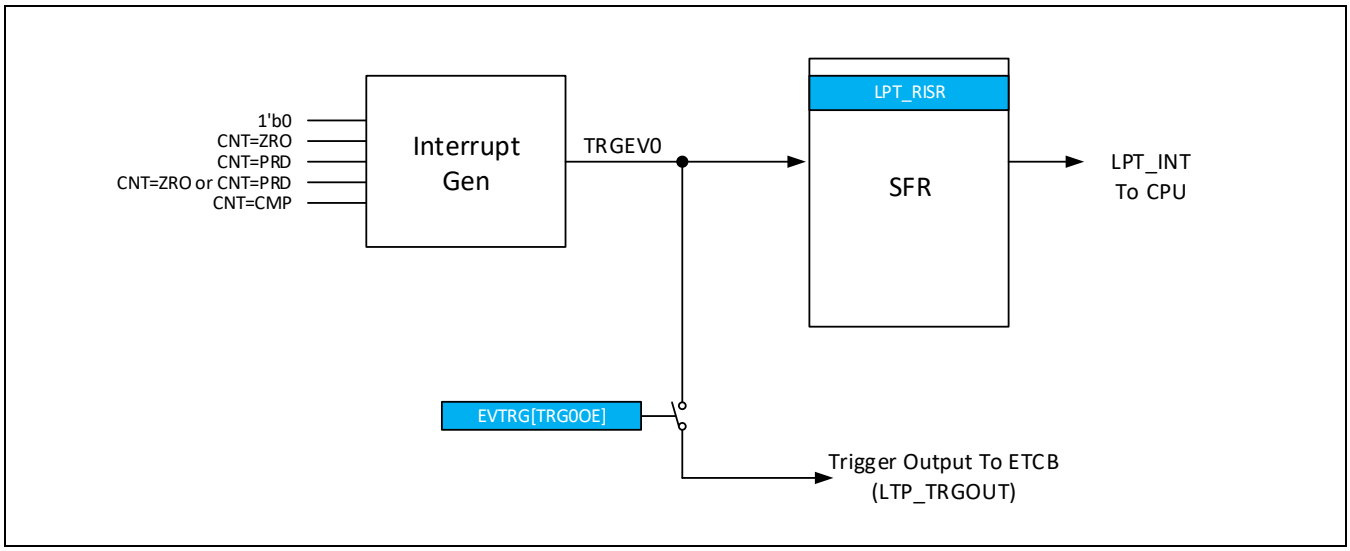


Figure 14-10 同步触发输出

14.3.6 中断触发

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

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

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

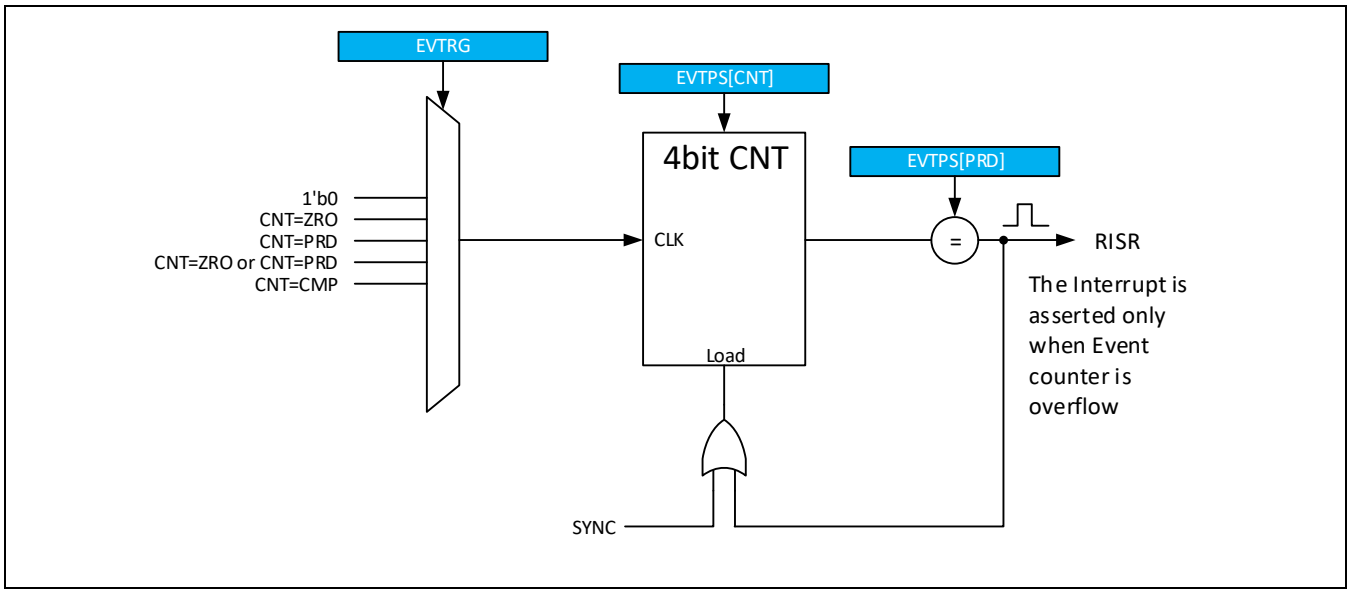


Figure 14-11 事件计数器

14.4 寄存器说明

14.4.1 寄存器表

Base Address of LPT: 0x40061000

Register	Offset	Description	Reset Value
LPT_CEDR	0x000	ID和时钟控制寄存器	0x20180000
LPT_RSSR	0x004	启停控制寄存器	0x00000000
LPT_PSCR	0x008	时钟分频控制寄存器	0x00000000
LPT_CR	0x00C	控制寄存器	0x00030010
LPT_SYNCR	0x010	同步控制寄存器	0x00000000
LPT_PRDR	0x014	周期设置寄存器	0x00000000
LPT_CMP	0x018	比较值寄存器	0x00000000
LPT_CNT	0x01C	时基计数器寄存器	0x00000000
LPT_TRGFCR	0x020	触发事件滤波窗控制寄存器	0x00000000
LPT_TRGFWR	0x024	触发事件滤波窗时序寄存器	0x00000000
LPT_EVTRG	0x028	事件触发选择寄存器	0x00000000
LPT_EVPS	0x02C	事件触发计数寄存器	0x00000000
LPT_EVSWF	0x030	软件强制触发事件控制寄存器	0x00000000
LPT_RISR	0x034	原始中断状态寄存器	0x00000000
LPT_MISR	0x038	中断状态寄存器	0x00000000
LPT_IMCR	0x03C	中断使能控制寄存器	0x00000000
LPT_ICR	0x040	中断清除寄存器	0x00000000
LPT_SR	0x044	同步状态寄存器	0x00000000

14.4.2 LPT_CEDR(ID和时钟控制寄存器)

Address = Base Address+ 0x000, Reset Value = 0x20180000

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

Name	Bit	Type	Description
IDCODE	[31:16]	R	当前LPT模块的版本信息。
FLTCKPRS	[15:8]	R	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。 0h：Shadow模式 1h：Immediate模式
CSS	[4:2]	RW	计数器时钟源选择位。 0h：PCLK/4 1h：ISCLK 2h：IMCLK/4 3h：EMCLK 4h：ESCLK 5h：LPT_IN的上升沿 6h：LPT_IN的下降沿 其他：保留
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。

14.4.3 LPT_RSSR(启停控制寄存器)

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																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LPT_CR[SWSYNEN]控制位为低时，START控制位用于控制LPT的启动，当LPT启动后，再次写入START将被忽略；当LPT_CR[SWSYNEN]控制位为高时，START控制位用于软件触发同步事件，每次对START的写入，会产生一次外部Sync事件（等同于SYNCR中的TRGUSR0触发）。

14.4.4 LPT_PSCR(时钟分频控制寄存器)

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																												PSC			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
PSC	[3:0]	RW	时钟分频控制。 TCLK作为时基模块的计时时钟和工作时钟。 0x0：不分频 0x1：2分频 0x2：4分频 0x3：8分频 0x4：16分频 0x5：32分频 0x6：64分频 0x7：128分频 0x8：256分频 0x9：512分频 0xA：1024分频 0xB：2048分频 0xC：4096分频 其他：保留

14.4.5 LPT_CR(控制寄存器)

Address = Base Address+ 0x00C, Reset Value = 0x00030010

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

Name	Bit	Type	Description
LPTOUTEN	[31]	RW	LPT输出使能控制 0h：禁止LPT输出 1h：使能LPT输出
CMPLD	[17]	RW	CMP寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值。 0h：立即更新，所有对CMP操作直接作用于活动寄存器 [2] 1h：CMP活动寄存器更新发生在周期结束时
PSCLD	[16]	RW	PSCR寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值。 0h：PSCR活动寄存器更新发生在启动或者外部触发时 1h：PSCR活动寄存器更新发生在周期结束，启动或者外部触发时
FLTDEB	[15:13]	RW	数字滤波去抖控制。此控制定义了滤波器监测的步数，只有连续N次监测结果一致时，滤波器才输出有效的电平翻转。滤波器的采样时钟频率在通过CKS控制位定义。 000b：Bypass 001b：N = 2 010b：N = 3 011b：N = 4 100b：N = 6 101b：N = 8 110b：N = 16 111b：N = 32
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
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
POL	[5]	RW	波形输出极性控制。

			0h：当CNT的值小于CMP时，输出高电平(当CMP=0时，输出低电平) 1h：当CNT的值小于CMP时，输出低电平(当CMP=0时，输出高电平)
PRDLD	[4]	RW	PRDR活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值。 0h：立即更新，所有对PRDR操作直接作用于活动寄存器 [1] 1h：PRDR活动寄存器更新发生在周期结束时
IDLEST	[3]	R	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。 注意 [2]：如果更新的比较值比更新前小，且立即更新发生时计数器已经超过更新的比较值，本周期将不会发生match事件。			

14.4.6 LPT_SYNCR(同步控制寄存器)

Address = Base Address+ 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	AREARM	RSVD													REARMx	RSVD							OSTMDx	RSVD							SYNCEN0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	RW	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	R	R	R	R	RW	R	R	R	R	R	R	R	RW

Name	Bit	Type	Description
AREARM	[30]	RW	硬件自动REARM控制位。 0：禁止硬件自动REARM 1：周期结束时，自动REARM
REARMx	[16]	RW	在一次性同步触发模式下，软件重置当前通道状态控制位。 当读取时，返回当前通道状态 0h：允许触发 1h：已经检测到触发，不允许后续触发 当写入时， 0h：无效 1h：清除当前通道状态，并允许新的触发
OSTMDx	[8]	RW	一次性同步触发模式选择。 0h：连续触发模式 1h：一次性触发模式 当该输入通道被设置为一次性触发模式后，在一次触发事件被检测到后，该通道将不允许后续的触发事件通过，直到被软件重置（REARM）后才允许新的触发事件通过。
SYNCEN0	[0]	RW	外部同步触发使能控制。 0：禁止当前触发输入通道 1：使能当前触发输入通道 LPT_SYNCIN0：外部Sync事件 (来自ETCB模块)

14.4.7 LPT_PRDR(周期设置寄存器)

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																PRDR															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

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

14.4.8 LPT_CMP(比较值寄存器)

Address = Base Address+ 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																CMP															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CMP	[15:0]	RW	比较值寄存器。 此寄存器有Shadow寄存器，只有在周期结束时才会对活动寄存器进行更新。

14.4.9 LPT_CNT(时基计数器寄存器)

Address = Base Address+ 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																CNT															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

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

14.4.10 LPT_TRGFCR(触发事件滤波窗控制寄存器)

Address = Base Address+ 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																								CROSSMD	RSVD		BLKINV	RSVD			SRC_SEL
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	RW	R	R	R	RW

Name	Bit	Type	Description
CROSSMD	[7]	RW	允许滤波窗跨越多个TB的周期。 缺省条件下，当滤波窗在周期结束时若任然有效，将跨过周期点，一直持续到窗口计数器溢出。当禁止跨周期时，在周期结束时，窗口计数器将被停止。 0h：禁止跨周期 1h：允许跨周期
BLKINV	[4]	RW	窗口使能反转控制。 0h：窗口不反转 1h：窗口反转
SRC_SEL	[0]	RW	滤波模块的输入信号选择。 0h：禁止滤波 1h：使能LPT_SYNCIN滤波

14.4.11 LPT_TRGFWR(触发事件滤波窗时序寄存器)

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
WINDOW																OFFSET															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
WINDOW	[31:16]	RW	滤波窗的宽度设置。 此16bit控制位定义了滤波窗的宽度，窗口宽度是基于TCLK的计数值。当OFFSET计数器溢出时，WINDOW计数器被重置，并开始计数直到溢出。当OFFSET计数器溢出，但WINDOW状态已经激活时，WINDOW计数器不会重置。在应用时必须注意此条件的设置。
OFFSET	[15:0]	RW	滤波窗的OFFSET设置。 此16bit控制位定义了从周期起始位置开始计数多少个TCLK后，开始有效的滤波窗口。OFFSET的Shadow寄存器在周期起始时，载入到Active寄存器中，并重新开始计数。

14.4.12 LPT_EVTRG(事件触发选择寄存器)

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											TRG0OE	RSVD																TRG0SEL			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
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
TRG0SEL	[3:0]	RW	TRGEV0事件的触发源选择。 0000：禁止TRGOUT触发输出 0001：当 CNT = ZRO 产生TRGEV0事件 0010：当 CNT = PRD 产生TRGEV0事件 0011：当 CNT = ZRO or CNT = PRD 产生TRGEV0事件 0100：当 CNT = CMP时，产生TRGEV0事件 其他：保留

14.4.13 LPT_EVPS(事件触发计数寄存器)

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												TRGEV0CNT				RSVD												TRGEV0PRD			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW

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

14.4.14 LPT_EVSWF(软件强制触发事件控制寄存器)

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																															EV0SWF
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW

Name	Bit	Type	Description
EV0SWF	[0]	RW	软件产生一次EV0的触发 0h：写入‘0’无效 1h：软件产生一次触发

14.4.15 LPT_RISR(原始中断状态寄存器)

Address = Base Address+ 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																													PEND	MATCH	TRGEV0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
PEND	[2]	R	PEND中断请求原始标志状态
MATCH	[1]	R	MATCH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

14.4.16 LPT_MISR(中断状态寄存器)

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																													PEND	MATCH	TRGEV0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
PEND	[2]	R	PEND中断请求标志状态
MATCH	[1]	R	MATCH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

14.4.17 LPT_IMCR(中断使能控制寄存器)

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																													PEND	MATCH	TRGEV0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW

Name	Bit	Type	Description
PEND	[2]	RW	PEND中断使能控制位
MATCH	[1]	RW	MATCH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			

14.4.18 LPT_ICR(中断清除寄存器)

Address = Base Address+ 0x040, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																													PEND	MATCH	TRGEV0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W

Name	Bit	Type	Description
PEND	[2]	W	清除PEND原始中断状态位
MATCH	[1]	W	清除MATCH原始中断状态位
TRGEV0	[0]	W	清除TRGEV0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’

14.4.19 LPT_SR(同步状态寄存器)

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																										CNT_BUSY	CMP_BUSY	PRDR_BUSY	CR_BUSY	PSCR_BUSY	CEDR_BUSY
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
CNT_BUSY	[5]	R	寄存器值同步状态 0：同步已经完成 1：该寄存器正在同步至LPT时钟域内，同步完成后该配置才生效
CMP_BUSY	[4]	R	寄存器值同步状态 0：同步已经完成 1：该寄存器正在同步至LPT时钟域内，同步完成后该配置才生效
PRDR_BUSY	[3]	R	寄存器值同步状态 0：同步已经完成 1：该寄存器正在同步至LPT时钟域内，同步完成后该配置才生效
CR_BUSY	[2]	R	寄存器值同步状态 0：同步已经完成 1：该寄存器正在同步至LPT时钟域内，同步完成后该配置才生效
PSCR_BUSY	[1]	R	寄存器值同步状态 0：同步已经完成 1：该寄存器正在同步至LPT时钟域内，同步完成后该配置才生效
CEDR_BUSY	[0]	R	寄存器值同步状态 0：同步已经完成 1：该寄存器正在同步至LPT时钟域内，同步完成后该配置才生效

15 窗口型看门狗 (WWDT)

15.1 概述

窗口型看门狗（Window Watchdog）作为可靠性保护逻辑，用于监测当前程序运行状况。当外部干扰或不可预见的逻辑错误发生时，造成当前程序运行错误，看门狗逻辑可以在预设时间周期结束时产生系统复位信号。看门狗计数器可以通过软件刷新以防止计数器溢出而产生复位，如果刷新事件发生在计数器值大于预设窗口计数值时，也将会触发复位信号。也就是刷新必须在预设的时间窗口内进行才有效。

15.1.1 主要特性

- 8 位可编程递减计数器
- 预设计数器时钟分频器：Div (1/2/4/8 x 4096)
  - 计数器时钟基于 PCLK 工作
  - 分频器的基础分频为 PCLK/4096
  - 可选择基于 4096 分频后的二次分频：DIV1，DIV2、DIV4 和 DIV8
- 产生复位的条件：
  - 递减计数器计数器值小于 0x80
  - 软件刷新计数器发生在预设窗口外
  - 软件写入的刷新计数器的数值小于 0x80
- 报警中断：当计数器值等于 0x80 时，可产生中断

15.2 功能描述

15.2.1 模块框图

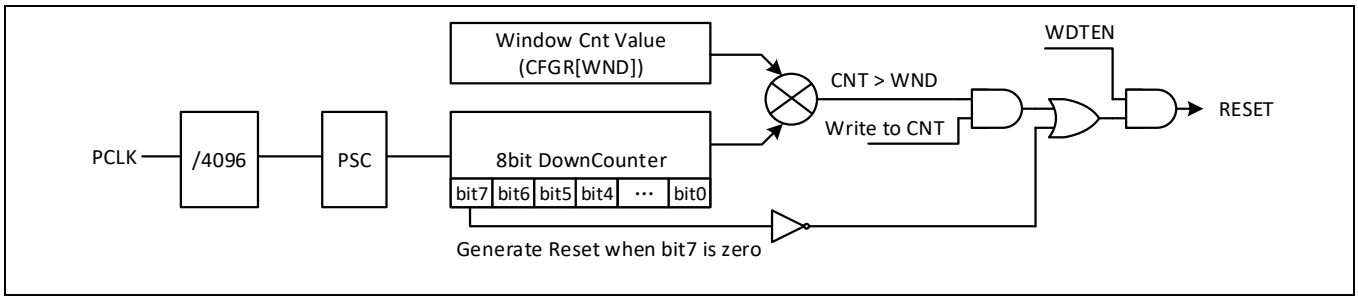


Figure 15-1 模块结构示意图

15.2.2 基本功能描述

看门狗缺省状态（系统复位后）为禁止状态，只有通过软件写入CR[WDTEN]后，才能使能。当看门狗被使能以后，不能通过WDTEN再关闭，只有系统复位发生后，看门狗逻辑被复位后，看门狗才会停止。

当看门狗被使能后（CR[WDTEN]被设置为高），看门狗计数器开始工作。当计数器值从0x80计数器到0x7F时，即计数器的最高位变成0时，将产生系统复位信号。当软件刷新计数器值时，当前计数器值大于窗口预设值（CFGR[WND]）时，也会触发系统复位。所以对看门狗计数器刷新时必须满足两个条件：

- 窗口条件：写CNT时，当前计数器值小于WND预设值
- 刷新数据：写入CNT的值必须在0xFF和0x80之间

15.2.3 时钟控制

看门狗工作时钟为系统PCLK时钟，当PCLK不工作时，看门狗计数器将暂停，直到PCLK恢复后才能继续工作。计算看门狗的溢出时间使用如下公式进行：

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

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 15-1 最小和最大溢出时间(PCLK=24MHz)

PSC	最小溢出时间（CNT[6:0]=0x00）	最大溢出时间（CNT[6:0]=0x7F）
0	170.67 us	21.85 ms
1	341.33 us	43.69 ms
2	682.67 us	87.38 ms
3	1365.33 us	174.76 ms

在连接ICE Debugger时，通过设置使能调试模式CFGR[DBGEN]，将WWDT的工作时钟在调试暂停时挂起。调试使能后，通过CDK调试时，一旦CPU被挂起，WWDT的计数器也同时被暂停，以防止计数器溢出造成调试复位。

15.2.4 计数器操作

WWDT内置一个8位的Free-running递减计数器。当WWDT使能时，必须保证计数器的最高位bit7为'1'，以防止立即产生RESET事件。计数器计数范围被限制在 0xFF ~ 0x80之间，CNT[6:0]控制位定义了计数器的计数次数。通过设置CNT[6:0]可以定义看门狗的溢出时间。

WWDT具有窗口功能，可以限定对计数器进行刷新的时间窗口，以防止由于程序错误而连续刷新，导致看门狗监测功能被异常屏蔽。窗口的设置可以通过CFGR[WND]控制位进行设置。当刷新CNT计数器时，如果当前计数器值大于窗口设置值，将产生复位信号。所以为避免刷新时产生复位，必须保证刷新时CNT计数值小于窗口设置。

CNT的最高位bit7，可以用于产生软件复位。当CNT最高位被写入'0'时，会立即触发一个软件复位事件。

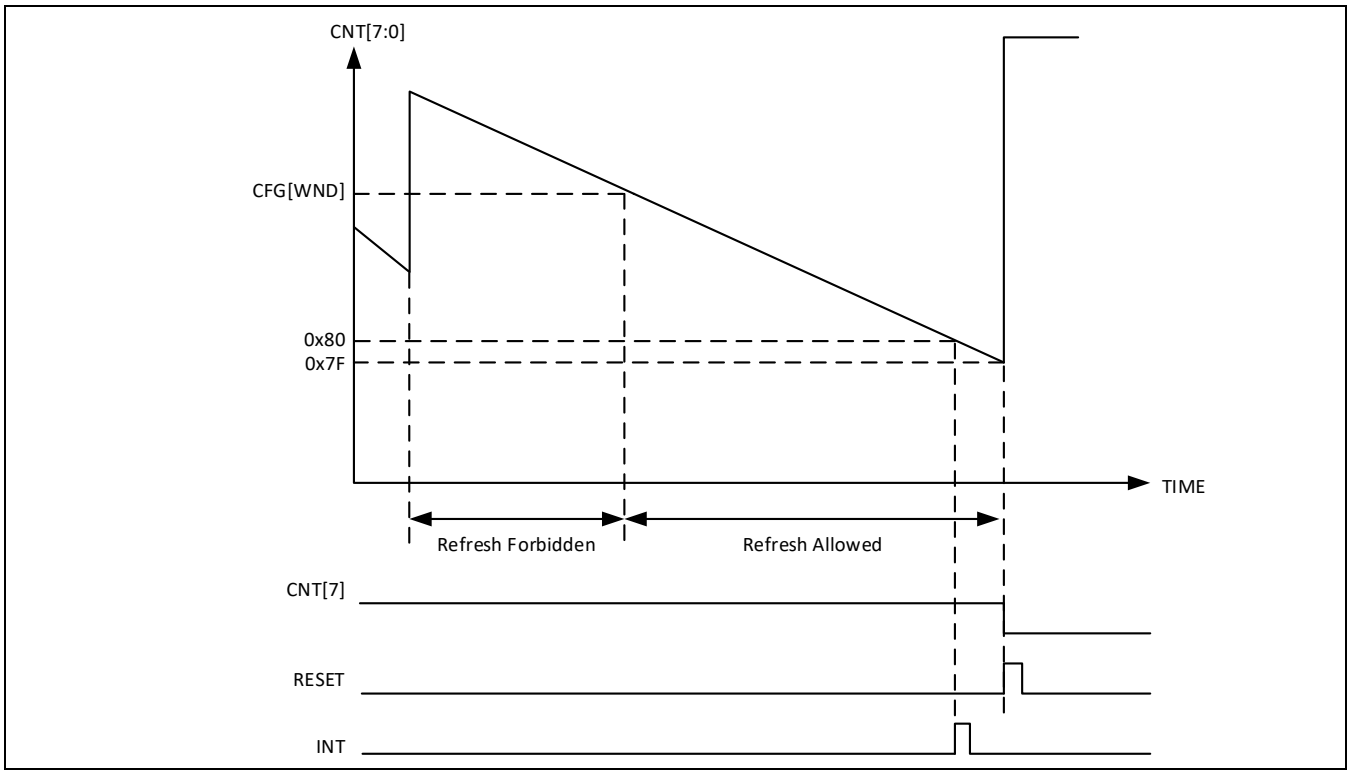


Figure 15-2 计数器工作示例

15.2.5 中断控制

WWDT的计数器计数到0x80时，可以产生一个警告中断。通过这个中断的服务程序，可以在将要发生的复位事件前作出一些处理，例如安全操作，现场保护和日志处理等。或者在中断服务程序中进行系统检查，然后确定是否刷新CNT以避免复位。

需要注意，在应用中当WWDT的中断优先级没有被置为最高，有可能导致WWDT中断服务程序被其他更高优先级的中断服务程序所阻挡，从而导致系统复位。

中断的使能通过IMCR寄存器进行控制。无论中断是否使能，中断的原始标志始终可以通过RISR寄存器进行查询。通过对ICR寄存器写入'1'，可有清除中断的标志位。寄存器说明

15.3 寄存器说明

15.3.1 寄存器表

Base Address of WWDT: 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

15.3.2 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。 0h：禁止看门狗 1h：使能看门狗 该使能控制位一旦使能后，不能通过软件关闭。需要复位后才能恢复初始禁止状态。
CNT	[7:0]	RW	计数器刷新值。 写入时，将当前计数器设置为CNT的值。 读取时，返回当前计数器值。

15.3.3 WWDT_CFGR(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。

15.3.4 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

15.3.5 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

15.3.6 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			

15.3.7 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’			

16 通用同步异步收发器 (USART)

16.1 概述

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

串行数据传输通常使用在电脑之间的非网络通讯，以及终端和其它设备间的通讯。

16.1.1 主要特性

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

16.1.2 管脚描述

Table 16-1 USART管脚描述

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

16.2 功能描述

16.2.1 波特率发生器

16.2.1.1 功能描述

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

16.2.1.2 异步模式

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

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

16.2.1.3 同步模式

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

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

16.2.1.4 波特率配置示例

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

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

Table 16-2 异步模式 (SYNC = 0)

PCLK (MHz)	US_BRGR CD[15:0]	波特率	% 误差
40	2083	1200	–0.02%
	1042	2400	0.03%
	521	4800	0.03%
	260	9600	–0.16%
	174	14400	0.22%
	130	19200	–0.16%
	65	38400	–0.16%
37.5	1953	1200	–0.01%
	977	2400	0.04%
	488	4800	–0.06%
	244	9600	–0.06%
	163	14400	0.15%
	122	19200	–0.06%
	61	38400	–0.06%
36	1875	1200	0.00%
	938	2400	0.05%
	469	4800	0.05%
	234	9600	–0.16%
	156	14400	–0.16%
	117	19200	–0.16%
	39	57600	–0.16%
30	1563	1200	0.03%
	781	2400	–0.03%
	391	4800	0.10%
	195	9600	–0.16%
	130	14400	–0.16%
	98	19200	0.35%
	49	38400	0.35%
20	1042	1200	0.03%
	521	2400	0.03%
	260	4800	–0.16%
	130	9600	–0.16%

PCLK (MHz)	US_BRGR CD[15:0]	波特率	% 误差
	87	14400	0.22%
	65	19200	–0.16%
18.75	977	1200	0.04%
	488	2400	–0.06%
	244	4800	–0.06%
	122	9600	–0.06%
	81	14400	–0.47%
	61	19200	–0.06%
18	938	1200	0.05%
	469	2400	0.05%
	234	4800	–0.16%
	117	9600	–0.16%
	78	14400	–0.16%
16	833	1200	–0.04%
	417	2400	0.08%
	208	4800	–0.16%
	104	9600	–0.16%
	52	19200	–0.16%
	26	38400	–0.16%
15	781	1200	–0.03%
	391	2400	0.10%
	195	4800	–0.16%
	98	9600	0.3%
	65	14400	–0.16%
	49	19200	0.35%
10	521	1200	0.03%
	260	2400	–0.16%
	130	4800	–0.16%
	65	9600	–0.16%
9.375	488	1200	–0.06%
	244	2400	–0.06%
	122	4800	–0.06%
	61	9600	–0.06%
8	417	1200	0.08%
	208	2400	–0.16%
	104	4800	–0.16%
	52	9600	–0.16%

PCLK (MHz)	US_BRGR CD[15:0]	波特率	% 误差
	26	19200	–0.16%
	13	38400	–0.16%
5	260	1200	–0.16%
	130	2400	–0.16%
	65	4800	–0.16%
4.6875	244	1200	–0.06%
	122	2400	–0.06%
	61	4800	–0.06%
4	208	1200	–0.16%
	104	2400	–0.16%
	52	4800	–0.16%
	26	9600	–0.16%
	13	19200	–0.16%
2.5	130	1200	–0.16%
	65	2400	–0.16%
2	104	1200	–0.16%
	52	2400	–0.16%
	26	4800	–0.16%
	13	9600	–0.16%
1.25	65	1200	–0.16%
1	52	1200	–0.16%
	26	2400	–0.16%
	13	4800	–0.16%
0.5	26	1200	–0.16%
	13	2400	–0.16%
0.25	13	1200	–0.16%

Table 16-3 同步模式 (SYNC = 1)

PCLK (MHz)	US_BRGR CD[15:0]	波特率	% 误差
40	2083 × 16	1200	–0.02%
	1042 × 16	2400	0.03%
	521 × 16	4800	0.03%
	260 × 16	9600	–0.16%
	174 × 16	14400	0.22%
	130 × 16	19200	–0.16%
	65 × 16	38400	–0.16%
37.5	1953 × 16	1200	–0.01%
	977 × 16	2400	0.04%
	488 × 16	4800	–0.06%
	244 × 16	9600	–0.06%
	163 × 16	14400	0.15%
	122 × 16	19200	–0.06%
	61 × 16	38400	–0.06%
36	1875 × 16	1200	0.00%
	938 × 16	2400	0.05%
	469 × 16	4800	0.05%
	234 × 16	9600	–0.16%
	156 × 16	14400	–0.16%
	117 × 16	19200	–0.16%
	39 × 16	57600	–0.16%
30	1563 × 16	1200	0.03%
	781 × 16	2400	–0.03%
	391 × 16	4800	0.10%
	195 × 16	9600	–0.16%
	130 × 16	14400	–0.16%
	98 × 16	19200	0.35%
	49 × 16	38400	0.35%
20	1042 × 16	1200	0.03%
	521 × 16	2400	0.03%
	260 × 16	4800	–0.16%
	130 × 16	9600	–0.16%
	87 × 16	14400	0.22%
	65 × 16	19200	–0.16%
18.75	977 × 16	1200	0.04%
	488 × 16	2400	–0.06%

PCLK (MHz)	US_BRGR CD[15:0]	波特率	% 误差
	244 × 16	4800	–0.06%
	122 × 16	9600	–0.06%
	81 × 16	14400	–0.47%
	61 × 16	19200	–0.06%
18	938 × 16	1200	0.05%
	469 × 16	2400	0.05%
	234 × 16	4800	–0.16%
	117 × 16	9600	–0.16%
	78 × 16	14400	–0.16%
16	833 × 16	1200	–0.04%
	417 × 16	2400	0.08%
	208 × 16	4800	–0.16%
	104 × 16	9600	–0.16%
	52 × 16	19200	–0.16%
	26 × 16	38400	–0.16%
15	781 × 16	1200	–0.03%
	391 × 16	2400	0.10%
	195 × 16	4800	–0.16%
	98 × 16	9600	0.35%
	65 × 16	14400	–0.16%
	49 × 16	19200	0.35%
10	521 × 16	1200	0.03%
	260 × 16	2400	–0.16%
	130 × 16	4800	–0.16%
	65 × 16	9600	–0.16%
9.375	488 × 16	1200	–0.06%
	244 × 16	2400	–0.06%
	122 × 16	4800	–0.06%
	61 × 16	9600	–0.06%
8	417 × 16	1200	0.08%
	208 × 16	2400	–0.16%
	104 × 16	4800	–0.16%
	52 × 16	9600	–0.16%
	26 × 16	19200	–0.16%
	13 × 16	38400	–0.16%
5	260 × 16	1200	–0.16%
	130 × 16	2400	–0.16%

PCLK (MHz)	US_BRGR CD[15:0]	波特率	% 误差
	65 × 16	4800	–0.16%
4.6875	244 × 16	1200	–0.06%
	122 × 16	2400	–0.06%
	61 × 16	4800	–0.06%
4	208 × 16	1200	–0.16%
	104 × 16	2400	–0.16%
	52 × 16	4800	–0.16%
	26 × 16	9600	–0.16%
	13 × 16	19200	–0.16%
2.5	130 × 16	1200	–0.16%
	65 × 16	2400	–0.16%
2	104 × 16	1200	–0.16%
	52 × 16	2400	–0.16%
	26 × 16	4800	–0.16%
	13 × 16	9600	–0.16%
1.25	65 × 16	1200	–0.16%
1	52 × 16	1200	–0.16%
	26 × 16	2400	–0.16%
	13 × 16	4800	–0.16%
0.5	26 × 16	1200	–0.16%
	13 × 16	2400	–0.16%
0.25	13 × 16	1200	–0.16%

16.2.2 接收端功能

16.2.2.1 异步接收

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

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

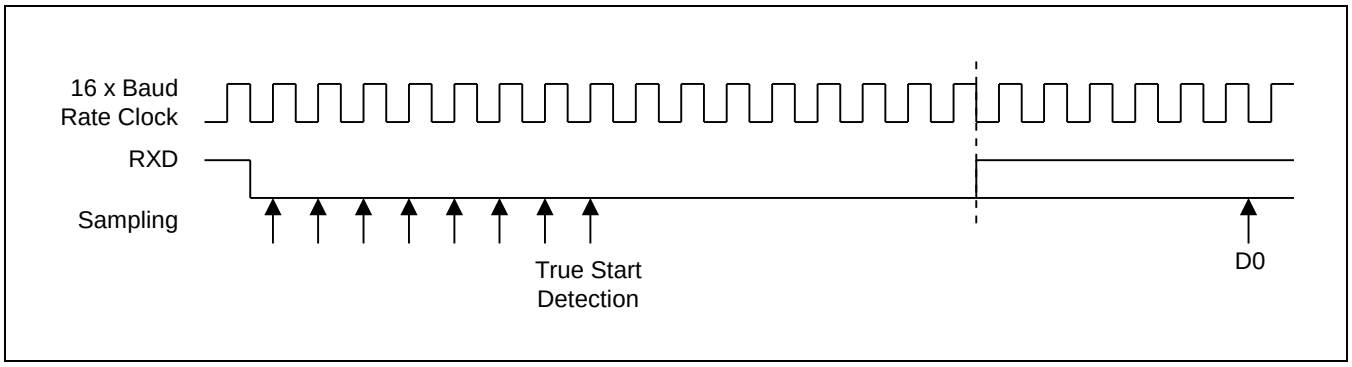


Figure 16-1 异步模式，起始位检测

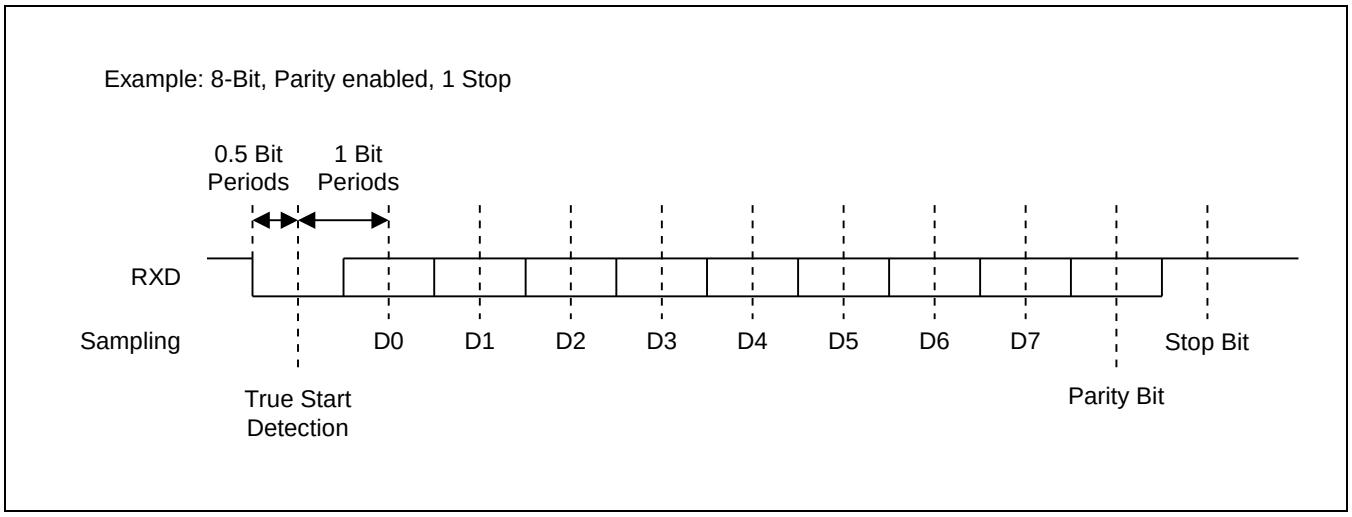


Figure 16-2 异步模式，字节接收

16.2.2.2 同步接收

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

16.2.2.3 接收标志位

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

16.2.2.4 溢出错误

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

16.2.2.5 校验错误

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

16.2.2.6 帧错误

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

16.2.2.7 空闲标志

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

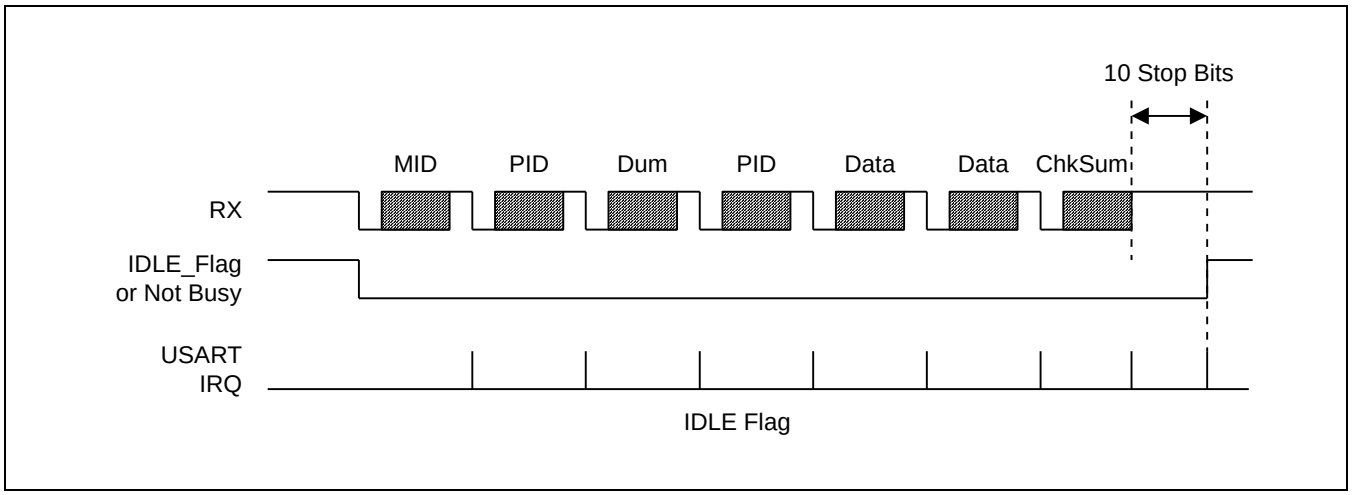


Figure 16-3 空闲标志

16.2.2.8 超时

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

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

也就是说，启动超时功能，必须满足下面条件：

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

超时的时长计算：

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

16.2.3 发送端

16.2.3.1 功能描述

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

数据位的个数由US_MR寄存器中的CHRL[1:0]选择。

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

停止位的个数由US_MR寄存器中的NBSTOP[1:0]选择。

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

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

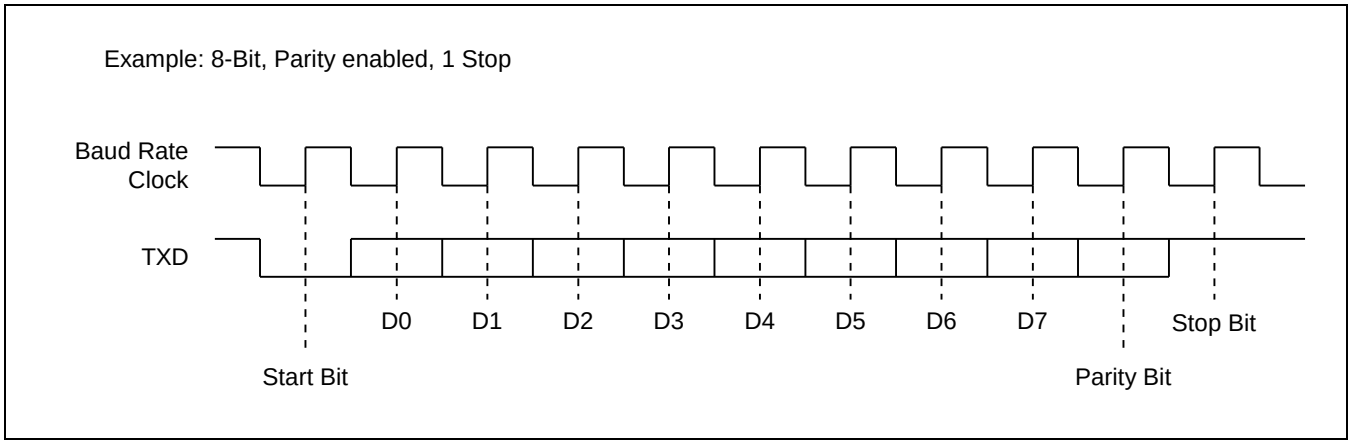


Figure 16-4 同步和异步模式，字节发送

16.2.3.2 Time-Guard

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

16.2.3.3 Multi-Drop模式

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

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

16.2.4 Break

16.2.4.1 发送Break

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

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

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

16.2.4.2 接收Break

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

16.2.5 中断

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

16.2.6 测试模式

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

自动回应模式：自动重新发送收到的数据，对发送端的任何配置都无效。

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

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

16.2.7 LIN协议（兼容LIN1.2和LIN2.0）

本节介绍USART支持的LIN协议。

当使能LIN协议（US_MR[LIN]=1b'1）时，USART充当主机（在LIN协议规范中定义的"Master Control Unit"）。

US_MR模式寄存器中的LIN2_0位选择LIN协议版本。默认情况下，USART支持LIN 1.2协议。软件用户在消息传输过程中无法修改此位。

如果在US_CR控制器寄存器中设置了STHEADER，则会自动生成"HEADER FRAME"，并且帧头的末尾通过状态寄存器上的 ENDHEADER位发出信号。帧头内容"IDENTIFIER"由标识符寄存器定义（读/写访问US_LIR）。标识符校验是自动计算的。在帧头部分中，SYNC_BREAK长度可通过同步中断长度寄存器（US_SBLR）进行配置（SYNC_BREAK可以从8Tbit到31Tbit）。

在US_CR寄存器中设置STRESP位具有发送"RESPONSE MESSAGE"（请参见LIN协议规范）的效果。对于LIN1.2，要传输的数据在US_DFWR[1：0]寄存器中定义。要传输的数据字节数取决于US_LIR寄存器中IDENTIFIER[5:4]的配置 。对于LIN2.0要传输的数据字节数取决于 寄存器US_LIR中NDATA[2：0]的配置。CHECKSUM字段将自动计算并发送（对于LIN1.2发布，校验和是经典的。对于LIN2.0发行版，校验和可以是增强的，也可以是经典的，这取决于US_LIR寄存器中CHK_SEL的配置）。

还可以填充对THR寄存器的消息响应。在这种情况下，用户必须先等待END HEADER中断，再开始填充THR寄存器（LDMAC可以使用数据字节。）

USART收到的"RESPONSE MESSAGE"可在US_DFRR[1：0]寄存器中使用，即使USART在US_DFWR[1：0]寄存器中输入了"RESPONSE MESSAGE"。

USART能够通过状态寄存器上的ENDMESS位检测并发出"LIN MESSAGE"的结束信号。此外，USART检测位错误，标识符奇偶校验错误，无响应错误，校验和错误和唤醒。

US_LIR和US_DFWR[1：0]寄存器，US_MR寄存器中的LIN2_0位，US_CR寄存器中的STHEADER，和US_SBLR寄存器中的SYNC_BRK[4：0]字段的写入，在消息传输期间都不起作用。

建议不要再帧头期间写US_THR寄存器。

字节间空间可通过US_TTGR寄存器（传输时间保护）进行配置，默认的字节间空间是一个Tbit。

16.2.7.1 LIN的USART配置

要在LIN模式下工作，USART必须设置为正常模式（US_MR[CHMODE] = 2b'00），并且其收发都使能。

16.2.7.2 消息特征

一个完整的LIN消息包括SYNCH Break，SYNCH Field，PID Field，Data Feild 和CheckSum Field，如下图所示。其中帧头是由SYNCH Break，SYNCH Field和PID Field组成。

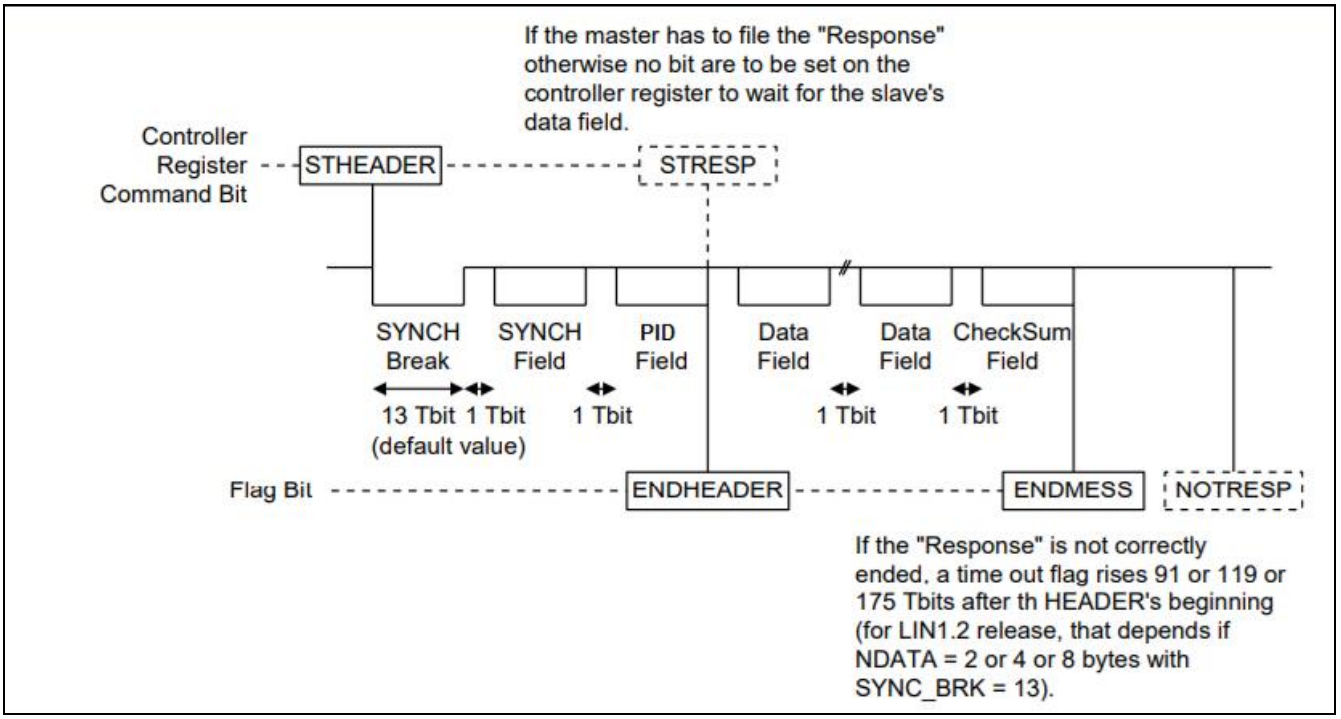


Figure 16-5 LIN消息特征

16.2.7.3 唤醒

USART可以通过发送唤醒信号从睡眠模式唤醒。对于LIN1.2协议，此信号由字符0xC0、0x80或0x00组成。这些字符将被主任务检测为有效数据。对于LIN2.0协议，唤醒请求通过强制总线处于显性状态至少250us实现，这将置位US_SR寄存器中的 WAKEUP位。

16.2.8 Smart-Card协议

USART兼容ISO7816-3协议，允许字节重送和校验错检测。

下面的描述只有在US_MR寄存器中的SMCARDPT位为1的时候才有效。

USART的Smart-Card协议需要发送端和接收端都支持。

如果GPIO模块允许，USARTTX可以配置成开漏输出模式，并且直接接到USARTRX管脚上，同时连接一个外

部的上拉电阻，形成Smart-Card的数据信号线。

16.2.8.1 发送字节到Smart Card

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

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

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

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

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

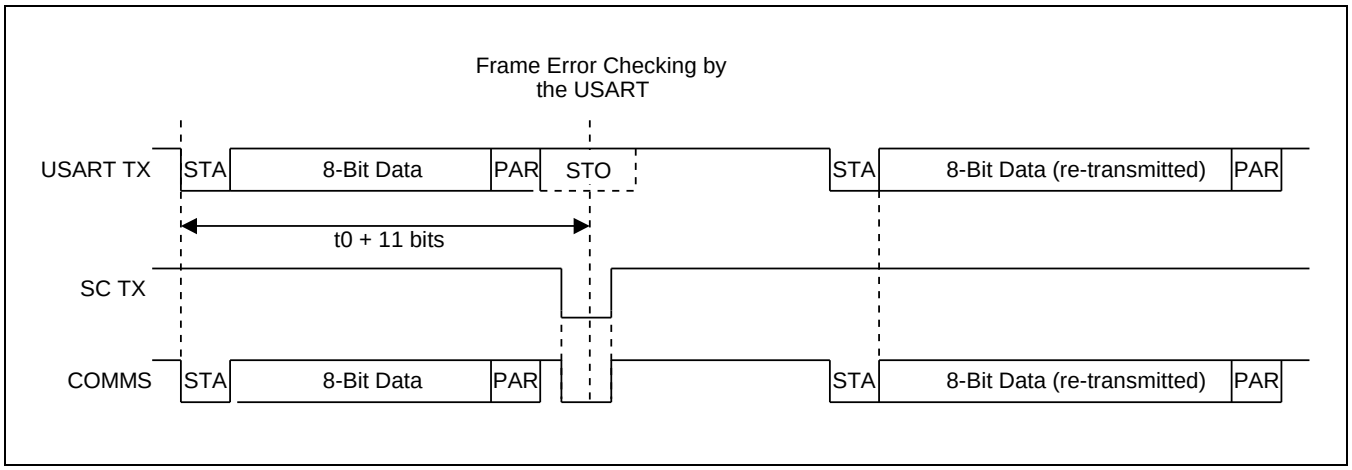


Figure 16-6 Smart-Card 发送错误

16.2.8.2 从Smart Card接收字节

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

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

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

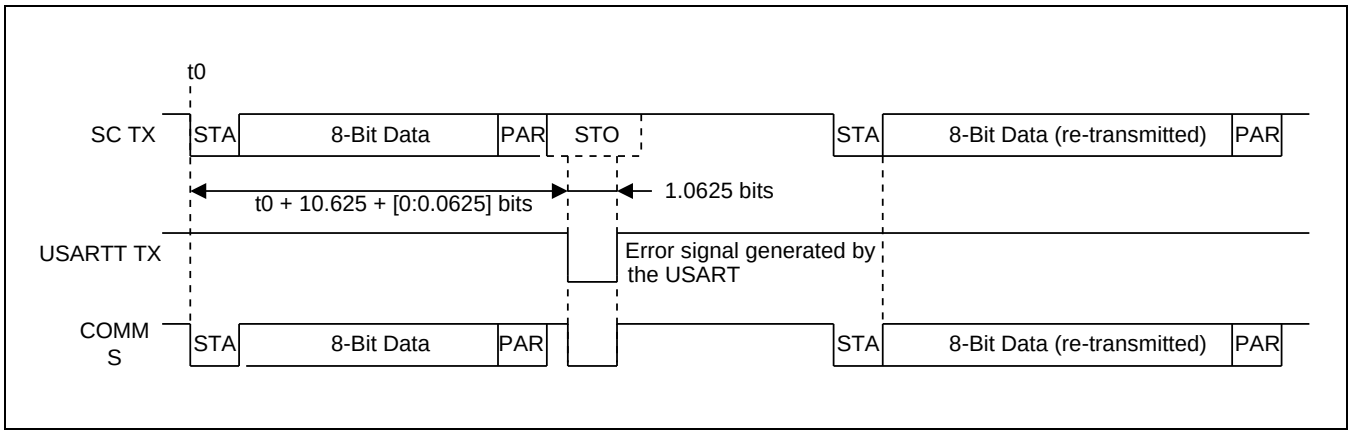


Figure 16-7 接收端的错误检测

16.2.8.3 Smart Card模式下的USART配置

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

16.2.9 IrDA SIR ENDEC模块

16.2.9.1 模块框图

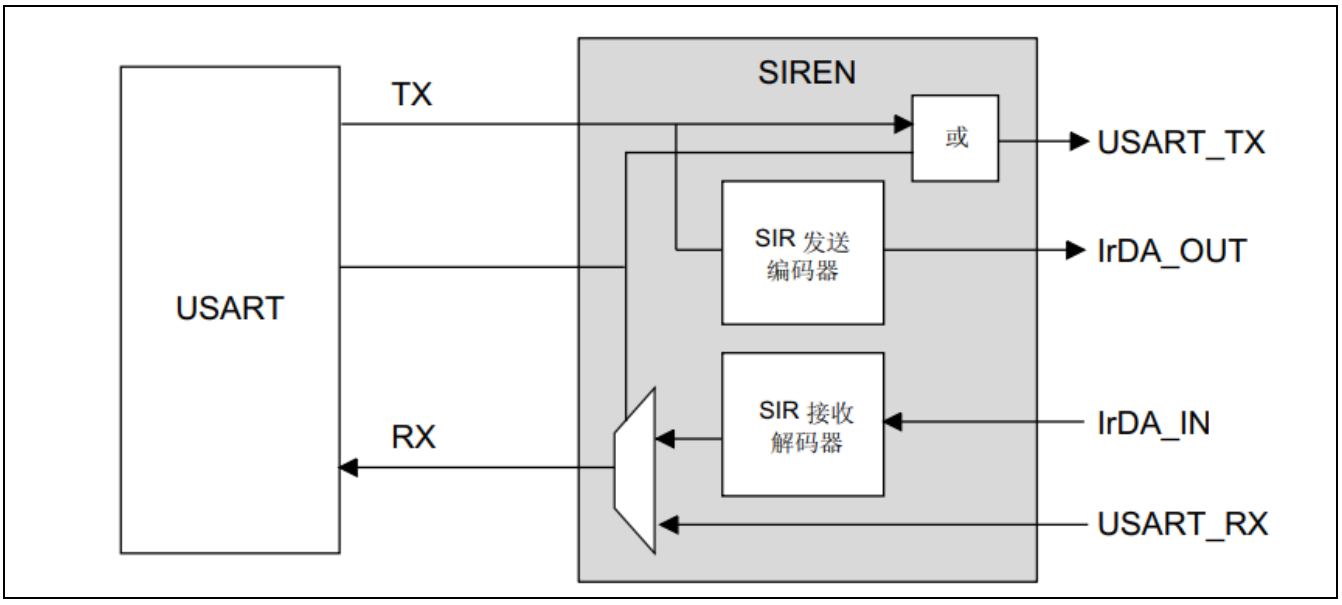


Figure 16-8 IrDA SIR ENDEC模块框图

16.2.9.2 IrDA数据调制

通过将US_IRDA寄存器中的IRDA_EN位置1来选择开启IrDA模式。在IrDA模式下，必须将LIN和Smart Card功能关闭。

IrDA SIR物理层规定使用反相归零（RZI）调试方案，它以红外脉冲表示逻辑0.

SIR发送编码器用于调制USART发出的非归零（NRZ）位流。输出脉冲流会发送到外部输出驱动器和红外LED。USART支持的SIR ENDEC比特率最高位115.2kbps。在默认情况下，所发送的脉冲宽度规定为一个位周期的3/16，其他的脉冲宽度可以通过寄存器US_IRDA的PLUSEWIDE位进行配置。

SIR 接收解码器用于解调由红外探测器发出的归零位流，并将接收到的NRZ串行位流输出到USART。在空闲状态下，解码器输入通常为高电平（标记状态）。发送编码器输出的极性与解码器输入相反。当解码器输入为低电平时，会检测到起始位。

- IrDA是一个半双工通信协议。如果发送器忙（USART正在向IrDA编码器发送数据时），则IrDA解码器会忽略IrDA接收线上的所有的数据；如果接收器忙（USART正在接收来自IrDA解码器的解码数据时），则IrDA不会对USART发送到IrDA的TX上的数据进行编码。接收数据时，应避免同时进行发送，因为这样可能会破坏要发送的数据。
- 0作为高电平脉冲发送，而1作为低电平脉冲发送。默认情况下，脉宽为位周期的3/16。
- SIR解码器用于将兼容IrDA的接收信号转换为USART位流。
- SIR接收逻辑将高电平状态视为逻辑1，将低电平脉冲视为逻辑0。
- 发送编码器输出的极性和解码器输入相反。SIR输出在空闲时处于低电平状态。

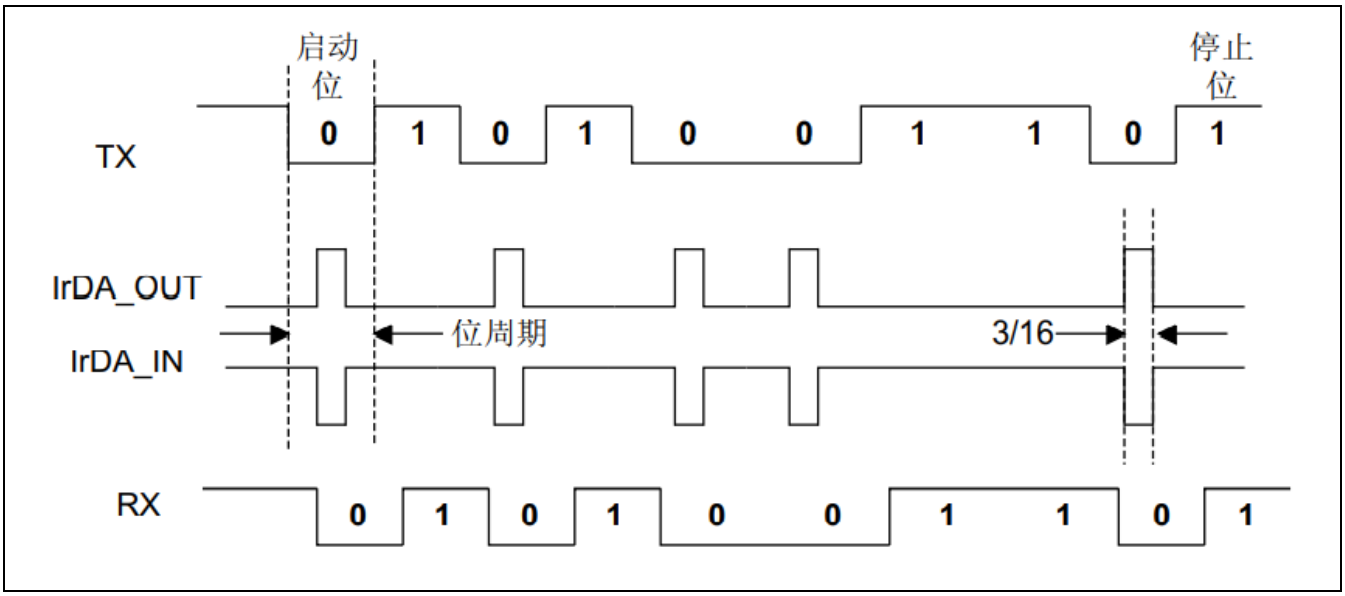


Figure 16-9 IrDA数据调制（3/16）

16.2.10 Smart-Card协议

USART兼容ISO7816-3协议，允许字节重送和校验错检测。

下面的描述只有在US_MR寄存器中的SMCARDPT位为1的时候才有效。

USART的Smart-Card协议需要发送端和接收端都支持。

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

16.2.10.1 发送字节到Smart Card

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

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

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

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

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

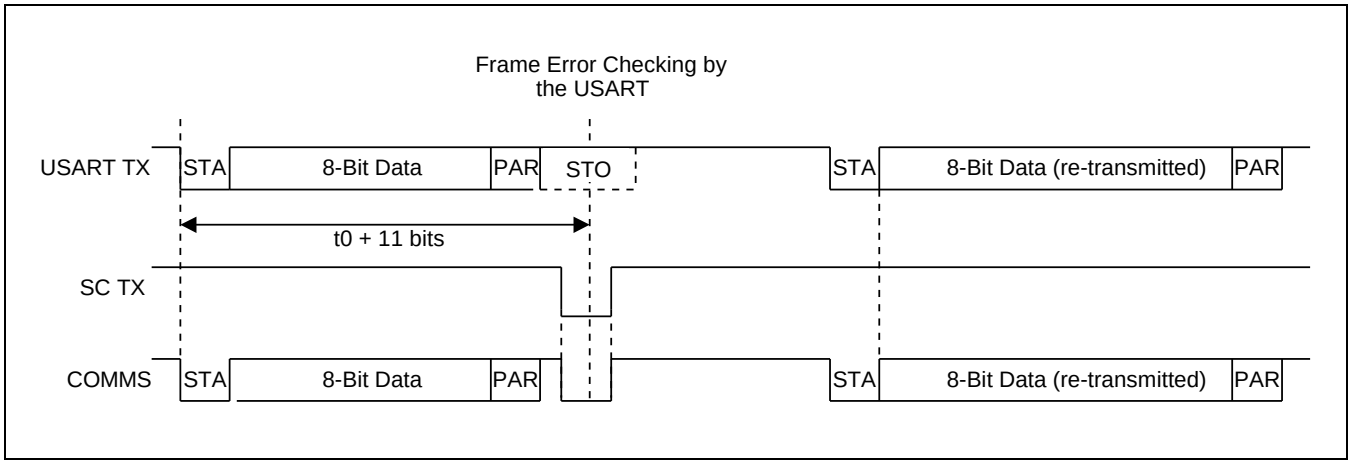


Figure 16-10 Smart-Card 发送错误

16.2.10.2 从Smart Card接收字节

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

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

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

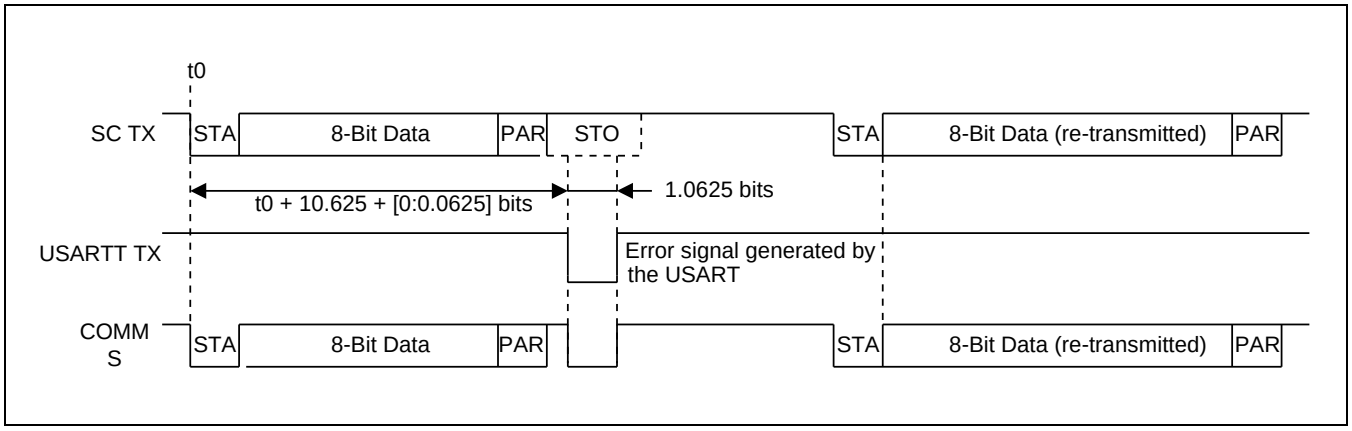


Figure 16-11 接收端的错误检测

16.2.10.3 Smart Card模式下的USART配置

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

16.3 寄存器说明

16.3.1 寄存器表

Base Address of USART0: 0x40080000

Base Address of USART1: 0x40081000

Register	Offset	Description	Reset Value
US_IDR	0x00	ID寄存器	0x00003839
US_CEDR	0x04	时钟使能/禁止寄存器	0x00000000
US_SRR	0x08	软件复位寄存器	0x00000000
US_CR	0x0C	控制寄存器	0x00000000
US_MR	0x10	模式寄存器	0x20000000
US_IMSCR	0x14	中断使能/禁止寄存器	0x00000000
US_RISR	0x18	原始中断状态寄存器	0x00004000
US_MISR	0x1C	中断状态寄存器	0x00000000
US_ICR	0x20	中断状态清除寄存器	0x00000000
US_SR	0x24	状态寄存器	0x00064800
US_RHR	0x28	接收数据寄存器	0x00000000
US_THR	0x2C	发送数据寄存器	0x00000000
US_BRGR	0x30	波特率配置寄存器	0x00000000
US_RTOR	0x34	接收超时配置寄存器	0x00000000
US_TTGR	0x38	发送端Time-Guard寄存器	0x00000000
US_LIR	0x3C	LIN标识寄存器	0x3AD40000
US_DFWR0	0x40	LIN写数据字段0寄存器	0x00000000
US_DFWR1	0x44	LIN写数据字段1寄存器	0x00000000
US_DFRR0	0x48	LIN读数据字段0寄存器	0x00000000
US_DFRR1	0x4C	LIN读数据字段1寄存器	0x00000000
US_SBLR	0x50	LIN同步间隔长度寄存器	0x0000000d
US_LCP1	0x54	LIN限制计数协议1寄存器	0x77685B4C
US_LCP2	0x58	LIN限制计数协议2寄存器	0xafa09284
US_DMACR	0x5C	DMA控制寄存器	0x00000000

16.3.2 US_IDR(ID寄存器)

Address = Base Address+ 0x00, Reset Value = 0x00003839

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

16.3.3 US_CEDR(时钟使能/禁止寄存器)

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

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

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

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

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

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

19.3.4 US_SRR(软件复位寄存器)

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

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

Name	Bit	Type	Description
TXFIFO_RST	[16]	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
SWRST	[0]	W	软件复位 0 = 无效 1 = 软件复位

16.3.5 US_CR(控制寄存器)

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

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

Name	Bit	Type	Description
RSTLIN	[19]	W	重启LIN 0= 无效 1= 重启LIN逻辑 此位重置 LIN 逻辑（APB 寄存器除外）配置。线路 TX/RX 处于空闲状态（即任何RX/TX 线路上的电流传输已中止）。
STMESSAGE	[18]	W	开始消息 0= 无效 1= 如果设置MR寄存器中的LIN位，连续发送LIN的帧头和响应
STRESP	[17]	W	开始响应 0= 无效 1= 发送一部分或者DFWR0和DFWR1寄存器内容
STHEADER	[16]	W	开始帧头 0= 无效 1= 如果设置MR寄存器中的LIN位，发送LIN的帧头
SENDA	[12]	W	发送地址 0 = 无效 1 = 只在Multi-drop模式，下一个写入US_THR的字节会被当作地址字节发送
STTTO	[11]	W	开启超时接收 0 = 无效 1 = 必须在超时计数器计数完成之前，接收到字节数据，否则报错
STPBRK	[10]	W	停止Break. 0 = 无效 1 = 如果一个Break状态正在发送，那么写1会在最少一个字节长度的Break状态后停止Break，并且发送一个12位周期的高电平
STTBRK	[9]	W	开始Break. 0 = 无效 1 = 如果Break没有发送，那么写1会在当前移位寄存器中的数据发送完之后，开始发送Break状态
TXDIS	[7]	W	发送禁止

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

16.3.6 US_MR(模式寄存器)

Address = Base Address+ 0x10, Reset Value = 0x20000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXIFLSEL			FIFO_EN	RSVD							DSB	LIN2_0	CLKO	MODE9	SMCARDPT	CHMODE		NBSTOP		PAR			SYNC	CHRL		CLKS		SENDTIME			LIN
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
RXIFLSEL	[31:29]	RW	接收FIFO中断触发点选择位 001 接收FIFO占用>=1/8 010 接收FIFO占用>=1/4 100 接收FIFO占用>=1/2 Others=保留
FIFO_EN	[28]	RW	FIFO模块有效，接收和发送模式下都需要经过相应的FIFO模块 0= 禁用FIFO 1= 使能FIFO
DSB	[20]	RW	数据开始位选择 0 = 数据发送从低位LSB开始，到高位MSB结束 1 = 数据发送从高位MSB开始，到低位LSB结束
LIN2_0	[19]	RW	选择LIN协议版本 设置LIN位时，此位很重要。软件用户无法在消息传输期间修改此值（即LIN总线忙）。 0 = USART支持LIN1.2协议 1 = USART支持LIN2.0协议
CLKO	[18]	RW	时钟输出选择 0 = USART不输出USARTCLK 1 = 如果CLKS[1]是0，且SYNC是1，USART输出USARTCLK
MODE9	[17]	RW	9位字节长度 0 = CHRL位定义字节长度 1 = 9位字节长度
SMCARDPT	[16]	RW	Smart Card协议 0 = 禁止smart card协议 1 = 使能smart card协议
CHMODE	[15:14]	RW	通道模式 • 通道模式位 CHMODE [1:0] 模式描述 00 普通模式USART通道工作为正常的Rx/Tx功能 01 自动回应收到的数据自动通过USARTTX发送

			10 本地回环发送端的输出信号短接到接收端的输入信号 11 远程回环USARTRX管脚内部直接短接到USARTTX管脚
NBSTOP	[13:12]	RW	停止位的个数 • NBSTOP配置位 NBSTOP [1:0] 停止位个数 00 1个停止位 01 保留 10 2个停止位 11 保留
PAR	[11:9]	RW	校验类型 • 校验类型位 PAR[2:0] 校验类型 000 偶校验 001 奇校验 010 0校验(Space) 011 1校验(Mark) 10X 无校验 11X Multi-drop模式 注意：如果使用LIN，PAR[2:0]必须设置为‘10X’.
SYNC	[8]	RW	同步模式选择 0 = USART工作在异步模式 1 = USART工作在同步模式
CHRL	[7:6]	RW	字节长度 (除开始位，停止位和校验位外的字节长度) • 字节长度位 CHRL[1:0] 字节长度 00 5位 01 6位 10 7位 11 8位
CLKS	[5:4]	RW	时钟选择 (波特率发生器的输入时钟). • CLKS 时钟选择位 CLKS[1:0] 选择的时钟 00 PCLK 01 PCLK/8 10 时钟为USART_CK0引脚输入时钟 11 保留
SENDTIME	[3:1]	RW	表示USART被配置成Smart Card协议时，重复发送最多的次数0-7 •SENDTIME配置位 SENDTIME[2:0]发送次数 000 0 001 1 – 111 7

LIN	[0]	RW	LIN模式 0 = USART上禁止LIN协议 1 = USART上使能LIN协议

16.3.7 US_IMSCR(中断使能/禁止寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	WAKEUP	CHECKSUM	IPERROR	BITERROR	NOTREPS	ENDMESS	ENDHEADER	RSVD									TXRIS	RORRIS	RXRIS	RSVD	IDLE	TXEMPTY	TIMEOUT	PARE	FRAME	OVRE	RSVD		RXBRK	TXRDY	RXRDY
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW

Name	Bit	Type	Description
WAKEUP	[30]	RW	唤醒中断 0 = 禁止中断 1 = 使能中断
CHECKSUM	[29]	RW	校验和错误中断 0 = 禁止中断 1 = 使能中断
IPERROR	[28]	RW	标识校验错误中断 0 = 禁止中断 1 = 使能中断
BITERROR	[27]	RW	LIN位错误中断 0 = 禁止中断 1 = 使能中断
NOTREPS	[26]	RW	LIN没有响应错误中断 0 = 禁止中断 1 = 使能中断
ENDMESS	[25]	RW	LIN消息结束中断 0 = 禁止中断 1 = 使能中断
ENDHEADER	[24]	RW	LIN帧头结束中断 0 = 禁止中断 1 = 使能中断
TXRIS	[14]	RW	发送FIFO中断 0 = 禁止中断 1 = 使能中断
RORRIS	[13]	RW	接收FIFO溢出中断 0 = 禁止中断 1 = 使能中断
RXRIS	[12]	RW	接收FIFO中断 0 = 禁止中断 1 = 使能中断

IDLE	[10]	RW	空间中断 0 = 禁止中断 1 = 使能中断
TXEMPTY	[9]	RW	发送缓冲空闲中断 0 = 禁止中断 1 = 使能中断
TIMEOUT	[8]	RW	超时中断 0 = 禁止中断 1 = 使能中断
PARE	[7]	RW	校验错中断 0 = 禁止中断 1 = 使能中断
FRAME	[6]	RW	帧错误中断 0 = 禁止中断 1 = 使能中断
OVRE	[5]	RW	溢出错误中断 0 = 禁止中断 1 = 使能中断
RXBRK	[2]	RW	接收端Break中断 0 = 禁止中断 1 = 使能中断
TXRDY	[1]	RW	发送端待机中断 0 = 禁止中断 1 = 使能中断
RXRDY	[0]	RW	接收端待机中断 0 = 禁止中断 1 = 使能中断

16.3.8 US_RISR(原始中断状态寄存器)

Address = Base Address+ 0x18, Reset Value = 0x00004000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	WAKEUP	CHECKSUM	IPERROR	BITERROR	NOTREPS	ENDMESS	ENDHEADER	RSVD									TXRIS	RORRIS	RXRIS	RSVD	IDLE	TXEMPTY	TIMEOUT	PARE	FRAME	OVRE	RSVD		RXBRK	TXRDY	RXRDY
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
WAKEUP	[30]	R	唤醒中断原始状态 WAKEUP的原始中断状态，不管该中断是使能还是禁止
CHECKSUM	[29]	R	校验和错误中断原始状态 CHECKSUM的原始中断状态，不管该中断是使能还是禁止
IPERROR	[28]	R	标识校验错误中断原始状态 IPERROR的原始中断状态，不管该中断是使能还是禁止
BITERROR	[27]	R	位错误中断原始状态 BITERROR的原始中断状态，不管该中断是使能还是禁止
NOTREPS	[26]	R	没有响应错误中断原始状态 NOTREPS的原始中断状态，不管该中断是使能还是禁止
ENDMESS	[25]	R	消息结束中断原始状态 ENDMESS的原始中断状态，不管该中断是使能还是禁止
ENDHEADER	[24]	R	帧头结束中断原始状态 ENDHEADER的原始中断状态，不管该中断是使能还是禁止
TXRIS	[14]	R	发送FIFO中断原始状态 TXRIS的原始中断状态，不管该中断是使能还是禁止
RORRIS	[13]	R	接收FIFO溢出中断原始状态 RORRIS的原始中断状态，不管该中断是使能还是禁止
RXRIS	[12]	R	接收FIFO中断原始状态 RXRIS的原始中断状态，不管该中断是使能还是禁止
IDLE	[10]	R	空闲中断原始状态 IDLE的原始中断状态，不管该中断是使能还是禁止
TXEMPTY	[9]	R	发送缓冲空闲中断原始状态 TXEMPTY的原始中断状态，不管该中断是使能还是禁止
TIMEOUT	[8]	R	超时中断原始状态 TIMEOUT的原始中断状态，不管该中断是使能还是禁止
PARE	[7]	R	帧错误中断原始状态 PARE的原始中断状态，不管该中断是使能还是禁止
FRAME	[6]	R	帧错误中断原始状态 FRAME的原始中断状态，不管该中断是使能还是禁止

OVRE	[5]	R	溢出错误中断原始状态 OVRE的原始中断状态，不管该中断是使能还是禁止
RXBRK	[2]	R	接收端Break中断原始状态 RXBRK的原始中断状态，不管该中断是使能还是禁止
TXRDY	[1]	R	发送端待机中断原始状态 TXRDY的原始中断状态，不管该中断是使能还是禁止
RXRDY	[0]	R	接收端待机中断原始状态 RXRDY的原始中断状态，不管该中断是使能还是禁止

16.3.9 US_MISR(中断状态寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	WAKEUP	CHECKSUM	IPERROR	BITERROR	NOTREPS	ENDMESS	ENDHEADER	RSVD									TXRIS	RORRIS	RXRIS	RSVD	IDLE	TXEMPTY	TIMEOUT	PARE	FRAME	OVRE	RSVD		RXBRK	TXRDY	RXRDY
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
WAKEUP	[30]	R	唤醒中断状态 WAKEUP中断使能后的状态
CHECKSUM	[29]	R	校验和错误中断状态 CHECKSUM中断使能后的状态
IPERROR	[28]	R	标识校验错误中断状态 IPERROR中断使能后的状态
BITERROR	[27]	R	位错误中断状态 BITERROR中断使能后的状态
NOTREPS	[26]	R	没有响应错误中断状态 NOTREPS中断使能后的状态
ENDMESS	[25]	R	消息结束中断状态 ENDMESS中断使能后的状态
ENDHEADER	[24]	R	帧头结束中断状态 ENDHEADER中断使能后的状态
TXRIS	[14]	R	发送FIFO中断状态 TXRIS中断使能后的状态
RORRIS	[13]	R	接收FIFO溢出中断状态 RORRIS中断使能后的状态
RXRIS	[12]	R	接收FIFO中断状态 RXRIS中断使能后的状态
IDLE	[10]	R	空闲中断状态 IDLE中断使能后的状态
TXEMPTY	[9]	R	发送缓冲空闲中断状态 TXEMPTY中断使能后的状态
TIMEOUT	[8]	R	超时中断状态 TIMEOUT中断使能后的状态
PARE	[7]	R	帧错误中断状态 PARE中断使能后的状态
FRAME	[6]	R	帧错误中断状态 FRAME中断使能后的状态

OVRE	[5]	R	溢出错误中断状态 OVRE中断使能后的状态
RXBRK	[2]	R	接收端Break中断状态 RXBRK中断使能后的状态
TXRDY	[1]	R	发送端待机中断状态 TXRDY中断使能后的状态
RXRDY	[0]	R	接收端待机中断状态 RXRDY中断使能后的状态

16.3.10 US_ICR(中断状态清除寄存器)

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

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

Name	Bit	Type	Description
WAKEUP	[30]	W	唤醒中断状态 0 = 无效 1 = 清除该中断状态
CHECKSUM	[29]	W	校验和错误中断状态 0 = 无效 1 = 清除该中断状态
IPERROR	[28]	W	标识校验错误中断状态 0 = 无效 1 = 清除该中断状态
BITERROR	[27]	W	位错误中断状态 0 = 无效 1 = 清除该中断状态
NOTREPS	[26]	W	没有响应错误中断状态 0 = 无效 1 = 清除该中断状态
ENDMESS	[25]	W	消息结束中断状态 0 = 无效 1 = 清除该中断状态
ENDHEADER	[24]	W	帧头结束中断状态 0 = 无效 1 = 清除该中断状态
IDLE	[10]	W	空闲中断状态 0 = 无效 1 = 清除该中断状态
TIMEOUT	[8]	W	超时中断状态 0 = 无效 1 = 清除该中断状态
PARE	[7]	W	帧错误中断状态 0 = 无效 1 = 清除该中断状态

FRAME	[6]	W	帧错误中断状态 0 = 无效 1 = 清除该中断状态
OVRE	[5]	W	溢出错误中断状态 0 = 无效 1 = 清除该中断状态
RXBRK	[2]	W	接收端Break中断状态 0 = 无效 1 = 清除该中断状态

16.3.11 US_SR(状态寄存器)

Address = Base Address+ 0x24, Reset Value = 0x00064800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINBUSY	WAKEUP	CHECKSUM	IPERROR	BITERROR	NOTREPS	ENDMESS	ENDHEADER	RSVD					TFE	TNF	RNE	RFF	TXRIS	RORRIS	RXRIS	IDLEFLAG	IDLE	TXEMPTY	TIMEOUT	PARE	FRAME	OVRE	RSVD		RXBRK	TXRDY	RXRDY
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
LINBUSY	[31]	R	LIN总线是否忙 0 = LIN总线空闲 1 = LIN总线忙
WAKEUP	[30]	R	是否发生唤醒 0 = 未检测到唤醒 1 = 检测到唤醒
CHECKSUM	[29]	R	是否发生校验和错误 0 = LIN上未检测到校验和错误 1 = LIN上检测到校验和错误
IPERROR	[28]	R	是否发生标识校验错误 0 = LIN上未检测到标识校验错误 1 = LIN上检测到标识校验错误
BITERROR	[27]	R	是否发送位错误 0 = LIN帧上未检测到位错误 1 = LIN帧上检测到位错误
NOTREPS	[26]	R	是否发生从机未响应错误 0 = LIN帧中未检测到从机未响应错误 1 = LIN帧中检测到从机未响应错误
ENDMESS	[25]	R	是否发送消息结束 0 = LIN帧消息未结束 1 = LIN帧消息结束
ENDHEADER	[24]	R	帧头是否发送结束 0 = LIN帧头发送未结束 1 = LIN帧头发送已结束
TFE	[18]	R	发送FIFO是否为空状态位 0 = 发送FIFO非空 1 = 发送FIFO为空
TNF	[17]	R	发送FIFO是否已满状态位 0 = 发送FIFO已满 1 = 发送FIFO未满

RNE	[16]	R	接收FIFO是否为空状态位 0 = 接收FIFO为空 1 = 接收FIFO非空
RFF	[15]	R	接收FIFO是否已满状态位 0 = 接收FIFO未满 1 = 接收FIFO已满
TXRIS	[14]	R	发送FIFO状态位 0 = 发送FIFO大于4 1 = 发送FIFO小于等于4
RORRIS	[13]	R	接收FIFO溢出状态位 0 = 接收FIFO未溢出 1 = 接收FIFO溢出
RXRIS	[12]	R	接收FIFO状态位 0 = 接收FIFO小于中断触发点 1 = 接收FIFO大于等于中断触发点 接收FIFO中断触发点选择请参考US_CR(USART控制寄存器)的RXIFLSEL位
IDLEFLAG	[11]	R	0 = USART正在接收一个帧 1 = USART没有在接收任何帧 该位表示J1587协议的帧传送状态，当接收开始时变低，在接收完成+10个停止位(10个周期的高电平)后变高。
IDLE	[10]	R	空闲状态 0 = 没有检测到J1587的结束帧 1 = 检测到J1587的结束帧
TXEMPTY	[9]	R	发送缓冲空闲 0 = US_THR寄存器或者发送缓冲寄存器中有字节待发送 1 = US_THR寄存器或者发送缓冲寄存器中没有字节待发送 当USART被禁止或者复位后，该位为0，US_CR中的发送使能功能会将该位置1。
TIMEOUT	[8]	R	超时 0 = 开始超时接收后，没有检测到超时，或者超时寄存器被设置为0 1 = 开始超时接收后，检测到了超时
PARE	[7]	R	校验错误 0 = 在上一次状态复位后，没有检测到校验位错(或者multi-drop模式下的数据字节) 1 = 在上一次状态复位后，检测到至少1个校验位错(或者multi-drop模式下的地址字节)
FRAME	[6]	R	帧错误 0 = 在上一次状态复位后，没有停止位被检测到低电平 1 = 在上一次状态复位后，至少有一个停止位被检测到低电平
OVRE	[5]	R	溢出错误 0 = 当RXRDY有效后，没有字节从接收移位寄存器传到US_RHR寄存

			器 1 = 当RXRDY有效后，至少有一个字节从接收移位寄存器传到了US_RHR寄存器
RXBRK	[2]	R	接收端Break. 0 = 在上一次状态复位后，还没有检测到Break 1 = 在上一次状态复位后，检测到Break
TXRDY	[1]	R	发送端待机 0 = US_THR中有一个字节正在等待发送到移位寄存器中，或者发送端被禁止 1 = US_THR中没有任何字节,等于0表示USART被禁止了，或者处于复位状态。US_CR中的发送使能命令会将这位置1。
RXRDY	[0]	R	接收端待机 0 = 自从上次读取US_RHR后没有收到任何完成的字节，或者接收端被禁止 1 = 自从上次读取US_RHR后没有收到了至少一个完成的字节

16.3.12 US_RHR(接收数据寄存器)

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

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

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

16.3.13 US_THR(发送数据寄存器)

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

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

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

16.3.14 US_BRGR(波特率配置寄存器)

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

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

Name	Bit	Type	Description
CD	[15:4]	RW	分频 同步模式下如果选择了外部时钟，此寄存器无效。 异步模式):CD = FLOOR[Fclk / (Baud Rate x 16), 1] 同步模式):CD = FLOOR[Fclk / Baud Rate, 1] CD[15:4] Action 0 关闭时钟 1 无分频 (1分频) 2 to 4095 波特率(异步模式)= Fclk /(16 × (CD+FRACTION/16)) 波特率(同步模式)= Fclk /(CD+FRACTION/16)
FRACTION	[3:0]	RW	小数修正值 同步模式下如果选择了外部时钟，此寄存器无效。 异步模式) FRACTION = ROUND[((Fclk / (Baud Rate x 16)) - CD) x 16, 0] 同步模式) FRACTION = ROUND[((Fclk / Baud Rate) - CD) x 16, 0] FRACTION[3:0] Action 0 to 15 波特率(异步模式)= Fclk / (16 x (CD + FRACTION/16)) 波特率(同步模式) = Fclk / (CD + FRACTION/16)
注意： Fclk是USART模块的输入时钟，波特率是通讯速度。 同步模式下，要想使用较低的波特率（如9600），可以使用IMOSC做主频，或者通过MR寄存器的CLKS位对主频进行分频			

16.3.15 US_RTOR(接收超时配置寄存器)

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

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

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

16.3.16 US_TTGR(发送端Time-Guard寄存器)

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

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

Name	Bit	Type	Description
TG	[7:0]	RW	Time-Guard配置 • Time-Guard配置位 TO[15:0] Action 0 禁止发送端的time-guard功能 1－255 USARTTX在每发送完一个字节后，会变高一段时间，这个时间段为time-guard时长 Time-guard时长= TG[7:0] × 位周期

16.3.17 US_LIR(LIN标识寄存器)

Address = Base Address+ 0x3C, Reset Value = 0x3AD40000

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

Name	Bit	Type	Description
WAKE_UP_TIME	[31:16]	RW	LIN2.0版本的唤醒时间 设置唤醒时间计数器。重启后计数器加载0x3AD4，该值对应于PCLK = 20 MHz 时至少为 753us。
CHK_SEL	[9]	RW	校验和选择 0 = 经典校验和 1 = 增强型校验和（兼容LIN2.0版本）
NDATA	[8:6]	RW	LIN2.0 版本的数据字段数 指定要发送或接收的数据字段的数量。范围是从0到7，对应于1到8数据字段。
IDENTIFIER	[5:0]	RW	LIN标识符 指示将被发送的下一个"HEADER MESSAGE"的LIN标识符内容。对于LIN1.2版，标识符[5：4]字段指定数据字段的数量。 标识符[5:4] 数据字段的数量 0 2 1 2 2 4 3 8

16.3.18 US_DFWR0(LIN写数据字段0寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA3								DATA2								DATA1								DATA0							
0	0	0	0	0	0	0	0	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
DATA3	[31:24]	RW	待发送的LIN字节字段 当CR寄存器上的STRESP进行配置时，"LIN响应消息"上待发送的数据消息。
DATA2	[23:16]	RW	待发送的LIN字节字段 当CR寄存器上的STRESP进行配置时，"LIN响应消息"上待发送的数据消息。
DATA1	[15:8]	RW	待发送的LIN字节字段 当CR寄存器上的STRESP进行配置时，"LIN响应消息"上待发送的数据消息。
DATA0	[7:0]	RW	待发送的LIN字节字段 当CR寄存器上的STRESP进行配置时，"LIN响应消息"上待发送的数据消息。
注意：信息传输期间不能写此寄存器			

16.3.19 US_DFWR1(LIN写数据字段1寄存器)

Address = Base Address+ 0x44, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA7								DATA6								DATA5								DATA4							
0	0	0	0	0	0	0	0	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
DATA7	[31:24]	RW	待发送的LIN字节字段7 当CR寄存器上的STRESP进行配置时，"LIN响应消息"上待发送的数据消息。
DATA6	[23:16]	RW	待发送的LIN字节字段6 当CR寄存器上的STRESP进行配置时，"LIN响应消息"上待发送的数据消息。
DATA5	[15:8]	RW	待发送的LIN字节字段5 当CR寄存器上的STRESP进行配置时，"LIN响应消息"上待发送的数据消息。
DATA4	[7:0]	RW	待发送的LIN字节字段4 当CR寄存器上的STRESP进行配置时，"LIN响应消息"上待发送的数据消息。

16.3.20 US_DFRR0(LIN读数据字段0寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA3								DATA2								DATA1								DATA0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
DATA3	[31:23]	R	接收的LIN字节字段3 当SR寄存器上的ENDMESS置位时，"LIN响应消息"上接收的数据消息。
DATA2	[22:16]	R	接收的LIN字节字段2 当SR寄存器上的ENDMESS置位时，"LIN响应消息"上接收的数据消息。
DATA1	[15:8]	R	接收的LIN字节字段1 当SR寄存器上的ENDMESS置位时，"LIN响应消息"上接收的数据消息。
DATA0	[7:0]	R	接收的LIN字节字段0 当SR寄存器上的ENDMESS置位时，"LIN响应消息"上接收的数据消息。

16.3.21 US_DFRR1(LIN读数据字段1寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA7								DATA6								DATA5								DATA4							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
DATA7	[31:24]	R	接收的LIN字节字段7 当SR寄存器上的ENDMESS置位时，"LIN响应消息"上接收的数据消息。
DATA6	[23:16]	R	接收的LIN字节字段6 当SR寄存器上的ENDMESS置位时，"LIN响应消息"上接收的数据消息。
DATA5	[15:8]	R	接收的LIN字节字段5 当SR寄存器上的ENDMESS置位时，"LIN响应消息"上接收的数据消息。
DATA4	[7:0]	R	接收的LIN字节字段4 当SR寄存器上的ENDMESS置位时，"LIN响应消息"上接收的数据消息。

16.3.22 US_SBLR(LIN同步间隔长度寄存器)

Address = Base Address+ 0x50, Reset Value = 0x0000000d

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																											SYNC_BRK				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
R	R	R	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
SYNC_BRK	[4:0]	RW	LIN同步间隔长度 此寄存器不能写入0~7的值（寄存器保持之前的值）。
注意：信息传输期间不能写此寄存器。			

16.3.23 US_LCP1(LIN限制计数协议1寄存器)

Address = Base Address+ 0x54, Reset Value = 0x77685B4C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCP3								LCP2								LCP1								LCP0							
0	1	1	1	0	1	1	1	0	1	1	0	1	0	0	0	0	1	0	1	1	0	1	1	0	1	0	0	1	1	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
LCP3~LCP0	[31:0]	RW	LIN限制计数协议 x对应于US_LIR寄存器定义的字节数（NDATA）。此参数允许定义超时计数器，用于"无响应错误"标志的超时限制的计算。对于LIN1.2或LIN2.0版本，超时限制定义如下：limit_cpt =LCPndata + SYNC_BRK －13. x = 0–3
注意：信息传输期间不能写此寄存器。			

16.3.24 US_LCP2(LIN限制计数协议2寄存器)

Address = Base Address+ 0x58, Reset Value = 0xafa09284

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCP7								LCP6								LCP5								LCP4							
1	0	1	0	1	1	1	1	1	0	1	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	1	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
LCP7~LCP4	[31:0]	RW	LIN限制计数协议 x对应于US_LIR寄存器定义的字节数（NDATA）。此参数允许定义超时计数器，用于"无响应错误"标志的超时限制的计算。对于LIN1.2或LIN2.0版本，超时限制定义如下：limit_cpt =LCPndata + SYNC_BRK －13. x = 4–7
注意：信息传输期间不能写此寄存器。			

16.3.25 US_DMACR(DMA控制寄存器)

Address = Base Address+ 0x5C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																												TXMODE	RXMODE	TXDMAE	RXDMAE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	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
TXMODE	[3]	RW	USART TX DMA 模式 0=发送FIFO未满，产生DMA数据请求 1=发送FIFO数据达到中断触发点，产生DMA数据请求
RXMODE	[2]	RW	USART RX DMA 模式 0=接收FIFO非空，产生DMA数据请求 1=接收FIFO数据达到中断触发点，产生DMA数据请求
TXDMAE	[1]	RW	发送DMA使能 0 = 禁止 1 = 使能
RXDMAE	[0]	RW	接收DMA使能 0 = 禁止 1 = 使能

19.3.26 US_IRDA(IRDA寄存器)

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

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

Name	Bit	Type	Description
PLUSEWIDE	[11:8]	RW	脉宽 0 ： 3/16波特率 1 ： 2 PCLK 2 ： 4 PCLK 3 ： 8 PCLK 4 ： 16 PCLK 5 ： 32 PCLK 6 ： 64 PCLK 7 ： 128 PCLK 8 ： 256 PCLK OTHER ： 3/16波特率
RXPOL	[1]	RW	IRDA接收极性位 0 ： 接收的数据与usart rx上的数据相反 1 ： 接收的数据与usart rx上的数据相同
IRDA_EN	[0]	RW	IRDA使能位 0 ： 失能 1 ： 使能

17 通用异步收发器 (UART)

17.1 概述

UART 是一个简单通用的异步串行接收和发送接口，支持 8 位的数据通信，支持校验位，每次发送都以一个停止位结束。

注：如果系列内芯片不具有某一外围，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 主要特性

- 可配置的波特率
- 固定的8位发送长度，支持8个单独的收发FIFO
- 固定一个(位)停止位
- 发送接收溢出检测
- 发送接收完成中断和溢出中断
- 支持4种校验位，奇偶校验和0/1校验
- 使用DMA实现连续通信

17.1.2 管脚描述

Table 17-1 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	–	–

17.2 功能描述

17.2.1 模块框图

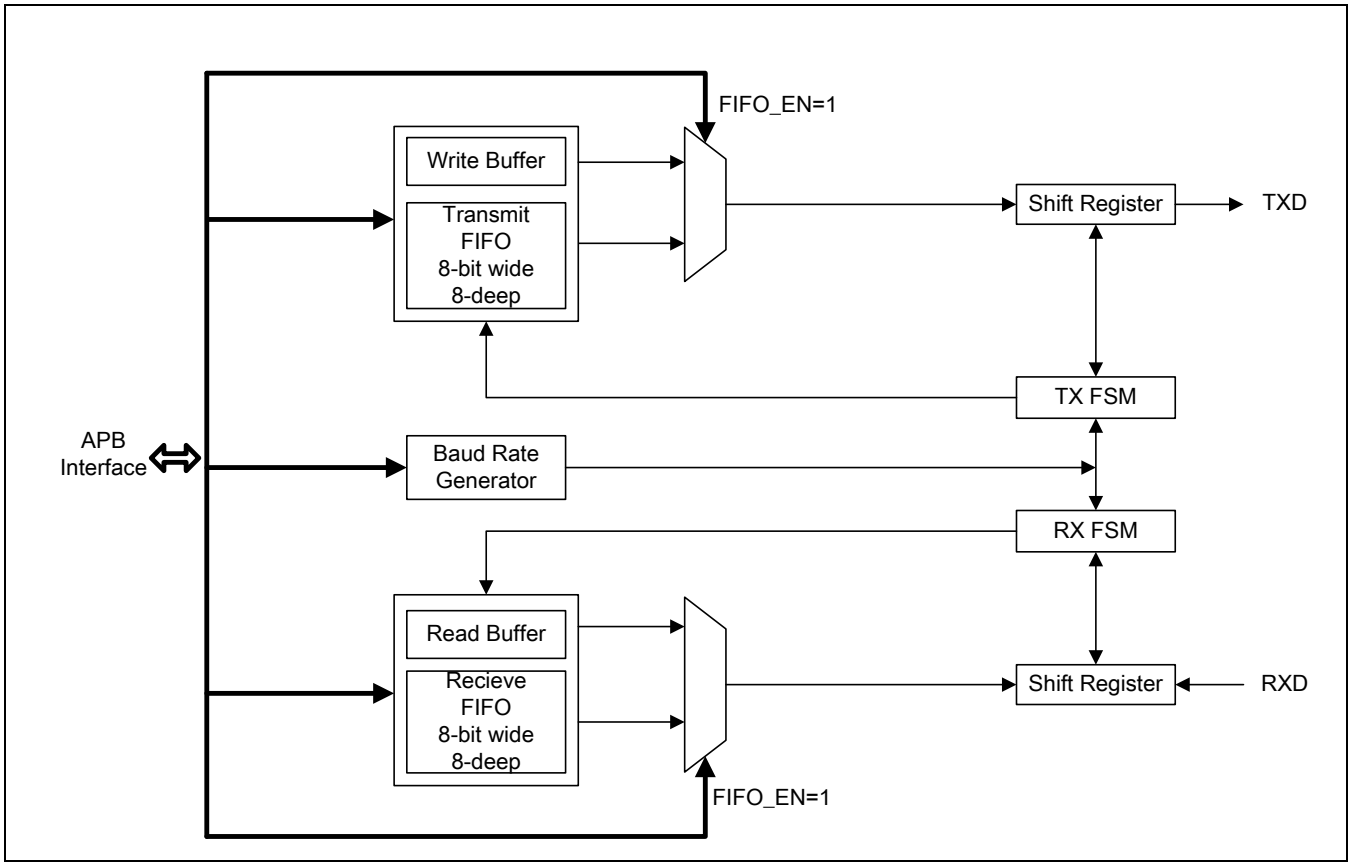


Figure 17-1 UART模块框图

17.2.2 功能说明

17.2.2.1 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 17-2 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%

17.2.2.2 接收

UART 通过检测 RXD 信号来判断接收字节的起始位。如果 RXD 上的低电平超过 7 个采样时钟的周期，那么这个低电平则被认为是有效的起始位。采样时钟的频率为波特率的 16 倍。所以长于 7/16 采样周期的低电平为有效，而比 7/16 个采样周期短的低电平则会被忽略，忽略后 UART 会继续等待有效的起始位。

当检测到一个有效的起始位，接收端开始在理论上每位的中心点读取 RXD 信号。假设每个数据位有 16 个采样周期的宽度，那么采样点则在起始位后的第 8 个采样周期(0.5 个数据位)处。所以第一个采样点是在 RXD 下降沿后的第 24 个采样周期(1.5 个数据位)时，之后每个采样点则每隔 16 个采样周期(1 个数据位)。

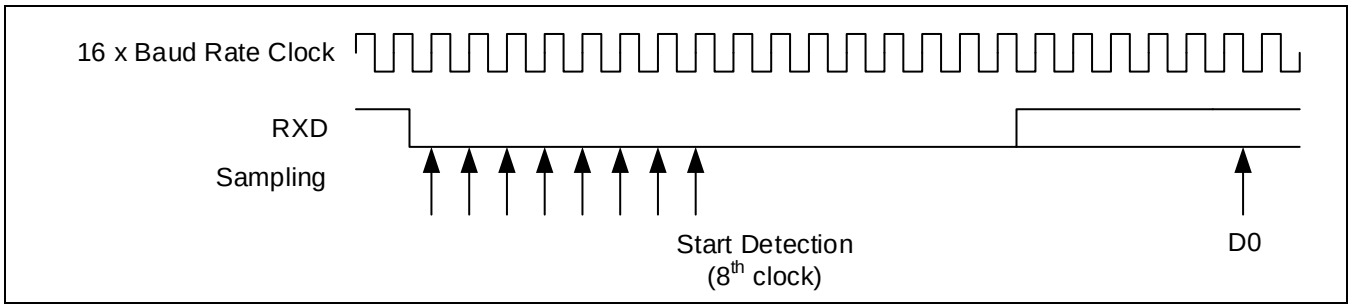


Figure 17-2 起始位检测

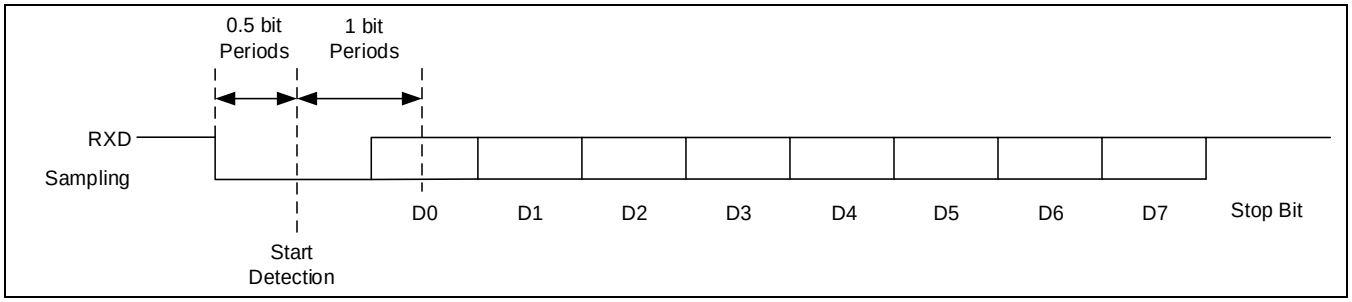


Figure 17-3 接收数据

当模块中的FIFO功能有效时，即当寄存器UART_CTRL中的FIFO_EN设置为1时，UART发送和接收都会分别通过发送FIFO和接收FIFO。接收FIFO是一个8位宽，8地址深的先进先出缓冲区。从串行接口接收到的数据存在缓冲区里，直到被CPU通过总线读出。

17.2.2.3 发送

发送过程中，起始位，数据位和停止位按顺序被移出，最低位(LSB)优先。需要发送数据时，先将 UART 模块使能(UART_CTRL 中的 TX 使能位)，再将数据写入数据寄存器(UART_DATA)即可。当写完数据寄存器 UART_DATA 后，数据会被立即发送出去。

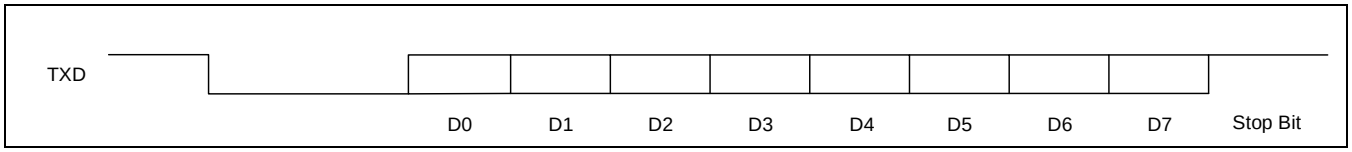


Figure 17-4 数据发送

当模块中的FIFO功能有效时，即当寄存器UART_CTRL中的FIFO_EN设置为1时，UART发送和接收数据都会分别通过发送FIFO和接收FIFO。发送FIFO是一个8位宽，8地址深的先进先出缓冲区。在需要通过UART_TX管脚串行发送数据时，先将数据写入发送FIFO，之后发送逻辑会自动将FIFO缓冲中的数据逐一发出。

17.2.2.4 校验位

UART_CTRL寄存器中的PARITY位用来设置校验方式。PARITY的第2位PARITY[2]如果为0，那么校验位被禁止，发送和接收都没有校验位。如果PARITY[2]为1，则校验位使能，根据PARITY[1:0]的设置，校验的模式不同：

PARITY[1:0]为00：偶校验，数据位(8位)与校验位中1的个数为偶数

PARITY[1:0]为01：奇校验，数据为(8位)与校验位中1的个数为奇数

PARITY[1:0]为10：0校验，校验位一直为0

PARITY[1:0]为11：1校验，校验位一直为1

17.2.2.5 中断

当接收到一个数据或者发送完一个数据后，状态寄存器 UART_SR 中的相应位会被置 1。如果收到的数据没有来得及被 CPU 读取而又再收到另一个数据时，或者如果当前数据还没发送完 CPU 就又往 UART_DATA 里写数据，那么 UART_SR 中的溢出位将会被置 1。

如果相应的中断被使能，那么 UART_ISR 里的寄存器也会被置位，同时 CPU 将会收到中断请求。

当模块中的 FIFO 使能时，即 UART_CTRL 中的 FIFO_EN 置为 1 时，UART 可以产生 3 个中断：

- UARTRXINTR_FIFO: UART 接收 FIFO 中断
- UARTTXINTR_FIFO: UART 发送 FIFO 中断
- UARTRORINTR_FIFO: UART 接收溢出中断

用户可以通过 UART_CTRL 寄存器中的行应为使能或禁止这些中断。

- UARTINTR_FIFO
  当接收 FIFO 占用到一定数量之后就会触发该中断。这个数量可以通过 UART_CTRL 中的 RXIFLSEL 位设置。
- UARTTXINTR_FIFO
  当发送 FIFO 的占用为 4 或者更少时，会触发该中断。数据的发送可以用两种方法操作。一是数据可以在中断前就写入发送 FIFO，二是中断使能后在发送 FIFO 中断服务子程序中写入数据。

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

17.2.2.6 DMA连续传输

将UART_DMACR寄存器的RXDMAEN位置1，在接收FIFO状态满足UART_DMACR寄存器的RXMODE位设置的条件，将发出DMA请求信号。该信号经过ETCB（事件触发控制器）模块选择后发送到DMA对应的通道x（0=0~3），（ETCB（事件触发控制器）相关设置，请参考ETCB章节）。将DMA_RSRx（DMA通道x请求选择寄存器）的REQ位置1，将触发该通道的硬件请求，（DMA其他设置，请参考DMA章节）。

将UART_DMACR寄存器的TXDMAEN位置1，在发送FIFO状态满足UART_DMACR寄存器的TXMODE位设置的条件，将发出DMA请求信号。该信号经过ETCB（事件触发控制器）模块选择后发送到DMA对应的通道x（0=0~3），（ETCB（事件触发控制器）相关设置，请参考ETCB章节）。将DMA_RSRx（DMA通道x请求选择寄存器）的REQ位置1，将触发该通道的硬件请求，（DMA其他设置，请参考DMA章节）。

17.3 寄存器说明

17.3.1 寄存器表

Base Address of UART0: 0x40082000
Base Address of UART1: 0x40083000
Base Address of UART2: 0x40084000

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	0x00004000
UART_ISR	0x00C	中断状态寄存器	0x00000000
UART_BRDIV	0x010	波特率分频寄存器	0x00000000
UART_DMACR	0x014	DMA控制寄存器	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
UAT_RX_FL	0x024	接收FIFO状态寄存器	0x00000000
UAT_TX_FL	0x028	发送FIFO状态寄存器	0x00000000

17.3.2 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

17.3.3 UART_SR(状态寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																					RXBRK	TIMEOUT	RFF	RNE	TNF	TFE	PARITY_SR	RX_OVER	TX_OVER	RX_FULL	TX_FULL
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	R	R

Name	Bit	Type	Description
RXBRK	[10]	R	接收端Break 0 = 在上一次状态复位后，还没有检测到Break 1 = 在上一次状态复位后，检测到Break（读） 1 = 清除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）
RFF	[8]	R	接收FIFO是否已满状态位 0 = 接收FIFO未满 1 = 接收FIFO已满
RNE	[7]	R	接收FIFO是否为空状态位 0 = 接收FIFO为空 1 = 接收FIFO非空
TNF	[6]	R	发送FIFO是否已满状态位 0 = 发送FIFO已满 1 = 发送FIFO未满
TFE	[5]	R	发送FIFO是否为空状态位 0 = 发送FIFO非空 1 = 发送FIFO为空
PARITY_SR	[4]	RW	PARITY 错误状态位 0 = 校验没有错误 1 = 校验出错 1 = 清除校验错误状态位(写)
RX_OVER	[3]	RW	RX缓冲区溢出状态 0 = RX缓冲区没有溢出 1 = RX缓冲区溢出(读取) 1 =清除RX缓冲区溢出标志(写)
TX_OVER	[2]	RW	TX缓冲区溢出状态 0 = TX缓冲区没有溢出 1 = TX缓冲区溢出(读取)

			1 = 清除TX缓冲区溢出标志(写)
RX_FULL	[1]	R	RX缓冲区状态 0 = RX缓冲区没有满(未收到数据或数据已被读取) 1 = RX缓冲区已满(收到数据，并且未被读取)
TX_FULL	[0]	R	TX缓冲区状态 0 = TX缓冲区没有满(可以发送数据) 1 = TX缓冲区已满(正在发送数据)

17.3.4 UART_CTRL(控制寄存器)

Address = Base Address+ 0x008, Reset Value = 0x00004000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBGEN	RSVD						STPBRK	STTBRK	INT_RXBRK	INT_RXTO	STTT0	INT_TX_DONE_EN	INT_OVER	RSVD	RXIFLSEL			INT_FIFO_RX	INT_FIFO_TX	FIFO_EN	PARITY			INT_PARITY	TEST	INT_OVER_RX	INT_OVER_TX	INT_RX	INT_TX	RX	TX
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	R	R	R	RW	RW	RW	W	W	RW	RW	W	W	W	R	W	W	W	W	W	W	W	W	W	RW	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
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
INT_OVER	[18]	W	FIFO使能下的RX溢出中断使能/禁止 0= 禁止RX溢出中断 1= 使能RX溢出中断
RXIFLSEL	[16:14]	W	接收FIFO中断触发点选择位 001 接收FIFO占用>=1/8

			010 接收FIFO占用>=1/4 100 接收FIFO占用>=1/2 Others=保留
INT_FIFO_RX	[13]	W	[13]：FIFO使能下的RX中断使能/禁止 0= 禁止RX中断 1= 使能RX中断
INT_FIFO_TX	[12]	W	[12]：FIFO使能下的TX中断使能/禁止 0= 禁止TX中断 1= 使能TX中断
FIFO_EN	[11]	W	FIFO模块有效，接收和发送模式下都需要经过相应的FIFO模块 0= 禁用FIFO 1= 使能FIFO
PARITY	[10:8]	W	校验位类型 0XX：无校验位 100：偶校验 101：奇校验 110：0校验，校验位一直为0(Space) 111：1校验，校验位一直为1(Mark)
INT_PARITY	[7]	RW	PARITY中断使能/禁止 0 = 禁止PARITY中断 1 = 使能PARITY中断
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

17.3.5 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								UART_RXBRK_INT_S	UART_RXTO_INT_S	RSVD	RXFIFO_OV_INT	RXFIFO_INT	TXFIFO_INT	PARITY_ERR	RX_OVER_INT	TX_OVER_INT	RX_INT	TX_INT
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	R	R	R	R	R	RW	R	R	RW	R	R	RW	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)
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)
RXFIFO_OV_INT	[7]	RW	接收FIFO溢出中断 0= 溢出中断没发生 1= 溢出中断发生(读取) 1= 清除溢出中断(写)
RXFIFO_INT	[6]	R	接收FIFO中断 0= RX中断没发生 1= RX中断发生(读取)
TXFIFO_INT	[5]	R	发送FIFO中断 0= TX中断没发生 1= TX中断发生(读取)
PARITY_ERR	[4]	RW	PARITY错误中断 0= PARITY错误中断没发生 1= PARITY错误中断发生(读取) 1= 清除PARITY中断(写)
RX_OVER_INT	[3]	RW	RX溢出中断

			0 = RX溢出中断没发生 1 = RX溢出中断发生(读取) 1 = 清除RX溢出中断(写)
TX_OVER_INT	[2]	RW	TX溢出中断 0 = TX溢出中断没发生 1 = TX溢出中断发生(读取) 1 = 清除TX溢出中断(写)
RX_INT	[1]	RW	RX中断 0 = RX中断没发生 1 = RX中断发生(读取) 1 = 清除RX中断(写)
TX_INT	[0]	RW	TX中断 0 = TX中断没发生 1 = TX中断发生(读取) 1 = 清除TX中断(写)

17.3.6 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

17.3.7 UART_DMACR(DMA控制寄存器)

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																												TXMODE	RXMODE	TXDMAEN	RXDMAEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	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
TXMODE	[3]	RW	UART TX DMA MODE 0=发送FIFO未满，产生DMA数据请求 1=发送FIFO数据占用<=1/2时，产生DMA数据请求
RXMODE	[2]	RW	UART RX DMA MODE 0=接收FIFO非空，产生DMA数据请求 1=接收FIFO数据达到中断触发点，产生DMA数据请求
TXDMAEN	[1]	RW	UART TX DMA enable
RXDMAEN	[0]	RW	UART RX DMA enable

17.3.8 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

17.3.9 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"。

17.3.10 UART_SRR(软件复位寄存器)

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

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

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

17.3.11 UAT_RX_FL(接收FIFO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																												RXFF_LEVER			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
RXFF_LEVER	[3:0]	R	

17.3.12 UAT_TX_FL(发送FIFO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																												TXFF_LEVER			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
TXFF_LEVER	[3:0]	R	

18 LED控制器(LED DRIVER)

18.1 概述

该型处理器内嵌一个 10x8 点的 LED 自动扫描控制器模块。它可以自动产生最大 10 个 COM 的扫描信号，通过不同时间槽分割来点亮 10 个 8 段数码管。在每个扫描时间槽（单独一个 COM）内，SEGMENT 输出口会将相应的预设值输出到 IO 口上，COM 扫描的通道可以自由定义。

18.1.1 特性

- 可配置的 COM 通道数（支持从1个通道到10个通道扫描模式）。
- 通过寄存器可配置 LED 的刷新率。
- 可选择的亮度调节功能。
- 支持 LED 闪烁扫描。

18.2 功能描述

18.2.1 功能模块

LED 扫描控制模块功能图如下所示。

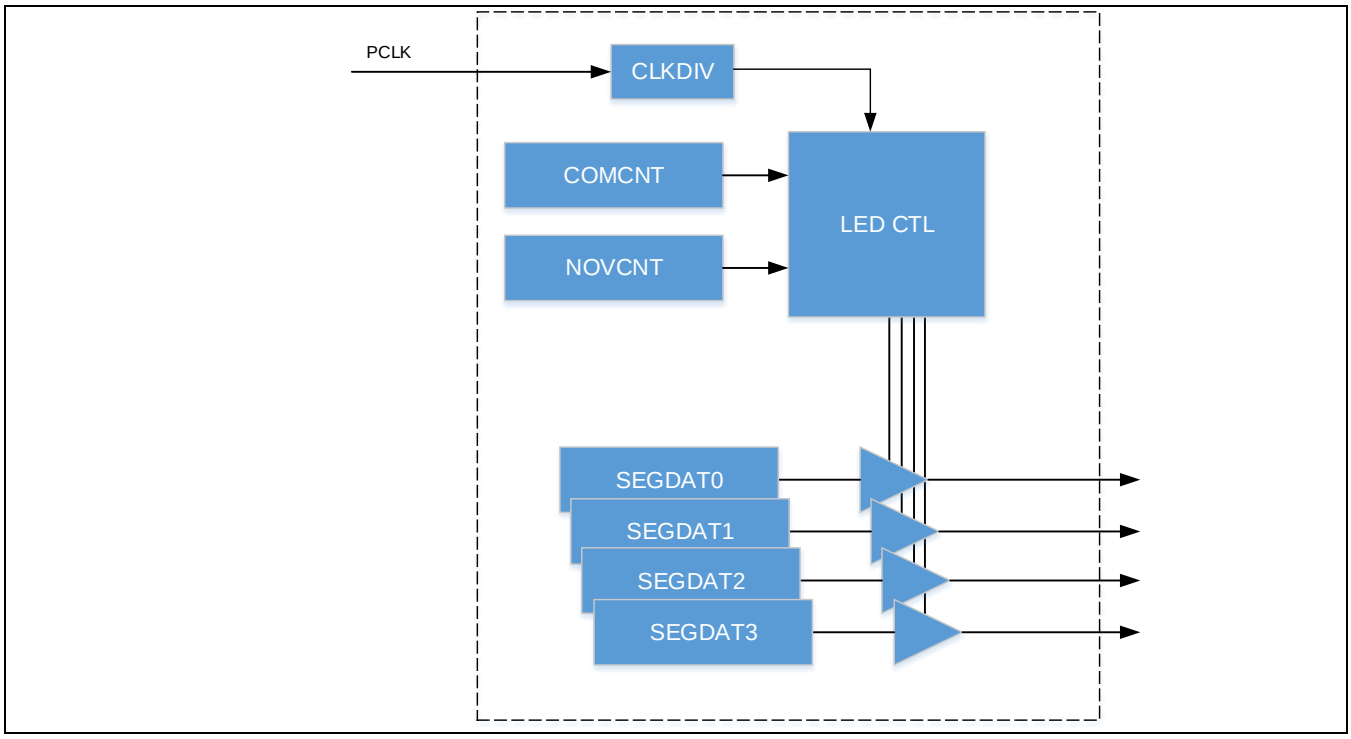


Figure 18-1 LED 扫描模块功能图

18.2.2 LED 扫描的工作模式

LED 自动扫描驱动器模块可以自动产生不互相叠加的扫描信号，用以驱动多个8段数码管。多个8段数码管的 SEG 端复用同一个驱动源。本驱动模块支持最大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。

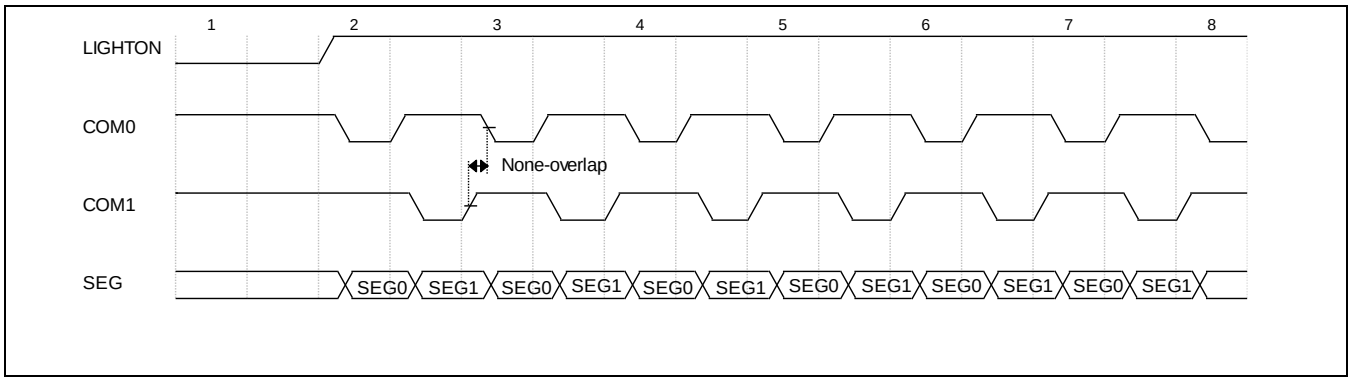


Figure 18-2 LED 扫描显示 （COM_EN = 8'b0000_0011模式）

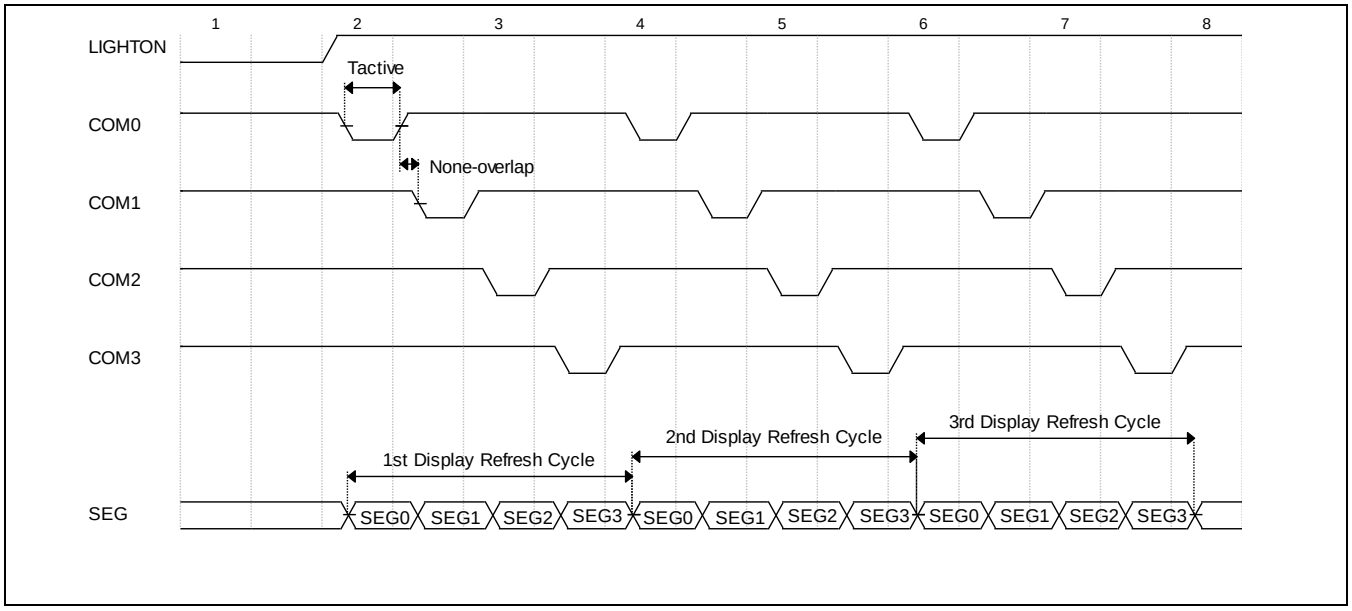


Figure 18-3 LED 扫描显示 （COM_EN = 8'b0000_1111模式）

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。

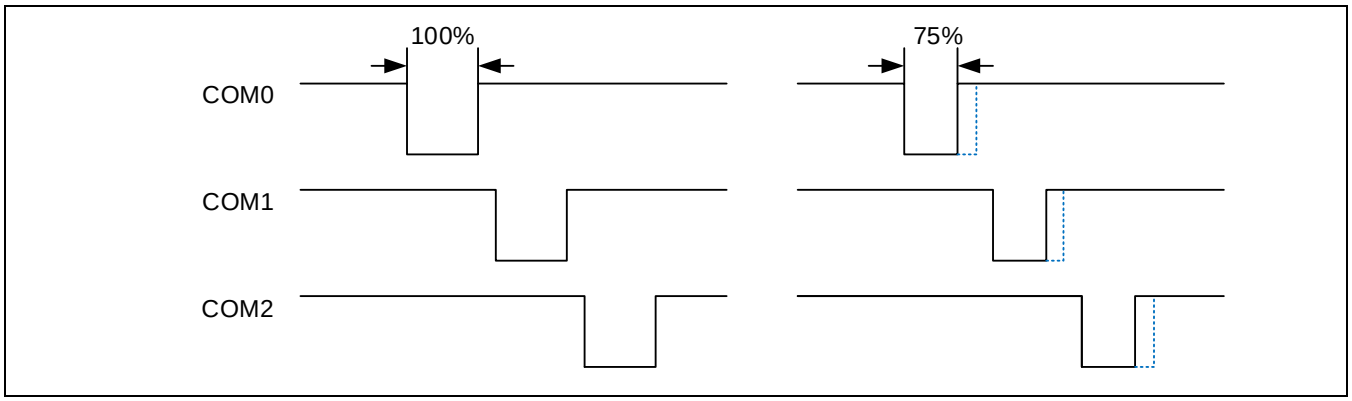


Figure 18-4 LED 显示亮度调节

18.2.3 LED 闪烁控制

LED 的每个扫描 COM 端口，可以独立设置该周期内是否禁止输出，从而实现在不影响扫描显示效果的情况下（扫描顺序和周期不变化），实现特定 COM 的刷新停止，从而实现特定 COM 控制端的闪烁效果。COM 的输出禁止可以通过设置 BLKER 来设置，通过 BLKDR 来清除禁止标志。

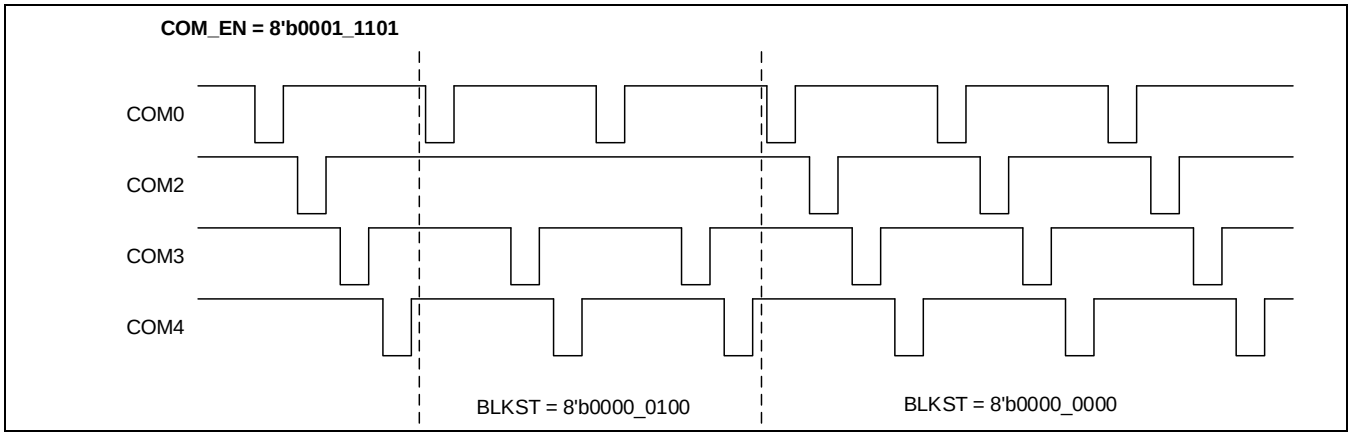


Figure 18-5 LED闪烁的工作模式

18.2.4 中断

LED 模式模块有三个中断。一个为 ICEND 中断，此中断将在每个 COM 扫描结束时触发。一个为 IPEND 中断，此中断将在所有的 COM 扫描结束后触发。另外一个为 IKEYDET 中断，此中断将在外部机械按键扫描时发现有按键状态发生变化时触发。当前按键的状态可以通过 KEYSR 寄存器获得。

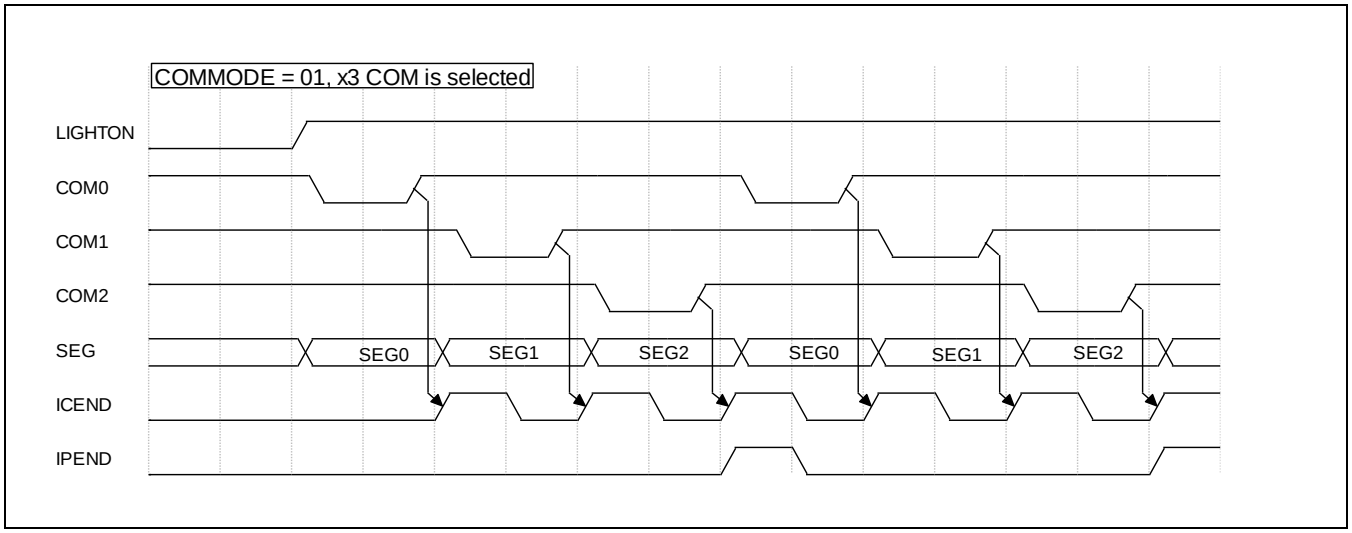


Figure 18-6 中断产生示意图

18.3 寄存器说明

18.3.1 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_BRIGHT	0x04	显示亮度控制寄存器	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_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_SEGDATx	0x2C~0x50	段码输出数据寄存器x	0x00000000
注意：x取值范围为0~9，表示当COMx使能时，SEG数据的输出			

18.3.2 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						COM_EN										RSVD												LEDCLK			LIGHTON
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
COM_EN	[25:16]	RW	COM扫描的使能选择。COM_EN的每一位对应一个COM扫描信号，当该位为‘1’时，所对应的COM扫描即被打开。 COM_EN[0]：COM0扫描使能 COM_EN[1]：COM1扫描使能 …… COM_EN[8]：COM8扫描使能 COM_EN[9]：COM9扫描使能
LEDCLK	[3:1]	RW	LED模块工作时钟频率设置（基于PCLK的分频）。 000：Fled = PCLK/4 001：Fled = PCLK/8 010：Fled = PCLK/16 011：Fled = PCLK/32 100：Fled = PCLK/64 101：Fled = PCLK/128 110：Fled = PCLK/256 111：Fled = PCLK/32
LIGHTON	[0]	RW	LED扫描启动。 1：启动LED自动扫描 0：停止LED自动扫描

18.3.3 LED_BRIGHT(显示亮度控制寄存器)

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															BRT		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW

Name	Bit	Type	Description
BRT	[2:0]	RW	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%

18.3.4 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																														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
IPEND	[1]	R	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]	R	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

18.3.5 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																														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	RW	RW

Name	Bit	Type	Description
IPEND	[1]	RW	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	一个COM扫描结束中断使能控制。 0：禁止中断 1：使能中断

18.3.6 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																														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
IPEND	[1]	R	所有COM扫描结束中断。 0：中断未发生 1：中断发生
ICEND	[0]	R	一个COM扫描结束中断。 0：中断未发生 1：中断发生

18.3.7 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																														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	W	W

Name	Bit	Type	Description
IPEND	[1]	W	清除所有COM扫描结束中断。
ICEND	[0]	W	清除一个COM扫描结束中断。

18.3.8 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
RSVD																NOVCNT								DCOMCNT							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
NOVCNT	[15:8]	RW	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(NOVCNT+7)
DCOMCNT	[7:0]	RW	在显示期间，计数器值决定了一个COM显示周期的长度。该计数器的计数时钟为LEDCLK/8。计数器的计数值为DCOMCNT+7

18.3.9 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	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
COMx_DIS	[9:0]	W	COM扫描周期内输出禁止（只写寄存器） 0：无效果 1：禁止该COM在扫描周期内输出

18.3.10 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	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
COMx_CLR	[9:0]	W	COM扫描周期内输出使能（只写寄存器） 0：无效果 1：使能该COM在扫描周期内输出

18.3.11 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	[9:0]	R	COM扫描周期内输出禁止状态 0：扫描输出使能 1：扫描输出禁止

18.3.12 LED_SEGDATx(段码输出数据寄存器x)

Address = Base Address+ 0x2C~0x50, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																								SEGDATA							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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
SEGDATA	[7:0]	RW	当COMx使能时，SEG端口输出的数据

19 串行外设接口 (SPI)

19.1 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。

19.2 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，16位宽，8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16位宽，8地址深
- 接收到的数据串行转并行，缓存到一个16位宽，8地址深的FIFO

19.2.1 管脚描述

Table 19-1 SPI 管脚描述

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

19.3 功能描述

19.3.1 模块框图

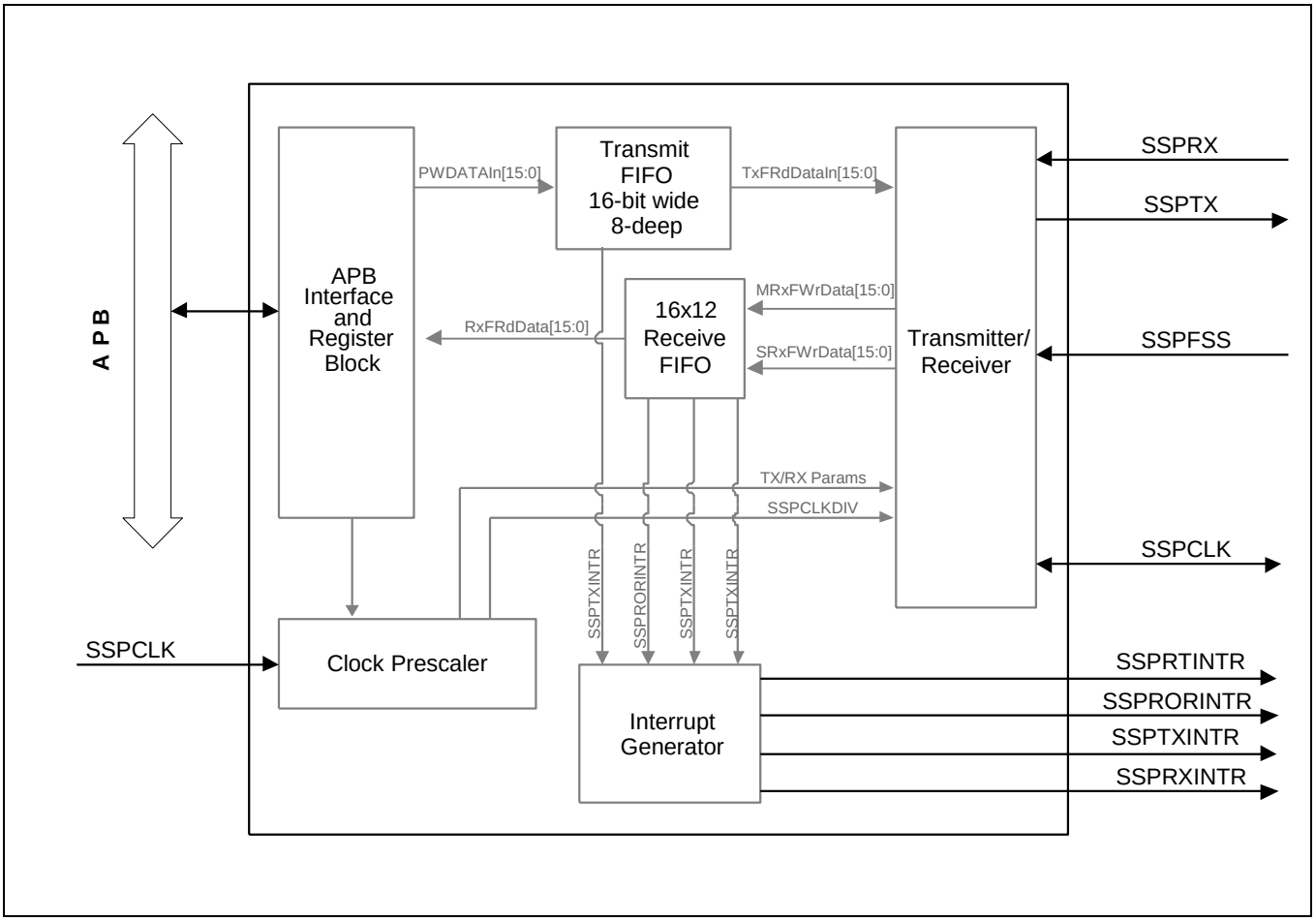


Figure 19-1 SPI 模块框图

19.3.2 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。

19.3.2.1 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SPI_SCK提供时钟源。

用户可以通过SPI_CPSR来配置预分频器，分两步给F_{SPICLK}进行2到254分频。SPI_CPSR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SPI_CR0可配置，这个分频器的输出最终输出到SPI_SCK管脚。

19.3.2.2 发送FIFO

发送FIFO是一个16位宽，8地址深的先进先出缓冲区。CPU通过AMBA APB接口将数据写入该缓存，直到发送逻辑将数据读出。在通过SPI_TX（主机模式是SPI_MOSI，从机模式是SPI_MISO）管脚串行发送数据前，需要先将并行的数据写入发送FIFO。

19.3.2.3 接收FIFO

接收FIFO是一个16位宽，8地址深的先进先出缓冲区。从串行接口接收到的数据存在缓冲区里，直到被CPU通过AMBA APB总线读出。在通过SPI_RX（主机模式是SPI_MISO，从机模式是SPI_MOSI）管脚收到串行数据后，才能读取FIFO的并行数据。

19.3.2.4 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[LPMD]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，SPI_RX和SPI_TX都将接在SPI_MOSI管脚上，SPI_RX将会一直接收SPI_MOSI上的信号，SPITX则通过SPI_CR1[LPTXOE]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。

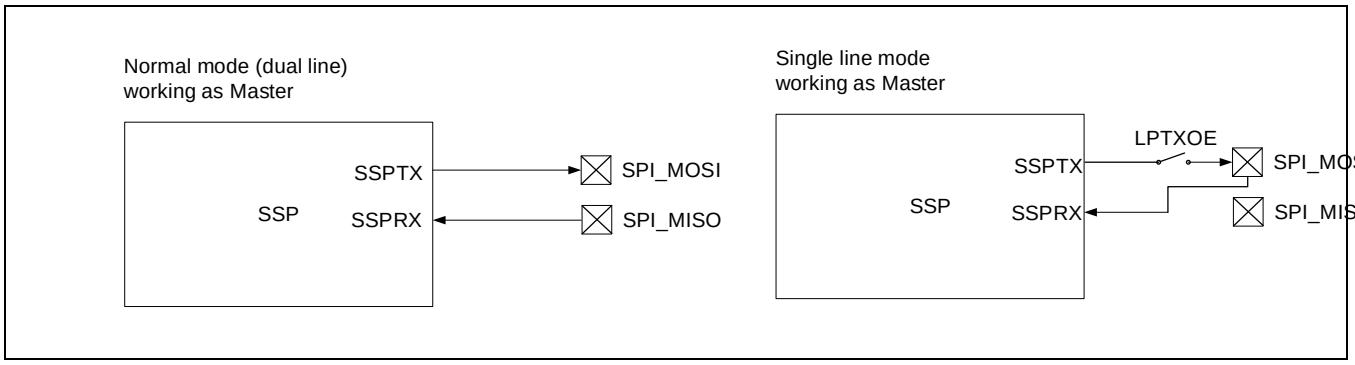


Figure 19-2 SPI 单线通讯模式

19.3.2.5 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[RXIFLSEL]位设置。

- SPITXINTR: SPI发送FIFO中断服务请求
  当发送FIFO的占用为4或者更少时，会触发该中断。这个发送中断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。SPIICR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_IMSCR寄存器中的相应位使能或者禁止这些中断。

19.3.3 SPI操作方法

19.3.3.1 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，可以通过写 SPI_SRR[SCLK_SWRST]实现。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，可以通过写 SPI_SSR[PCLK_SWRST]实现。

19.3.3.2 主从配置

通过控制寄存器SPI_CR1[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。

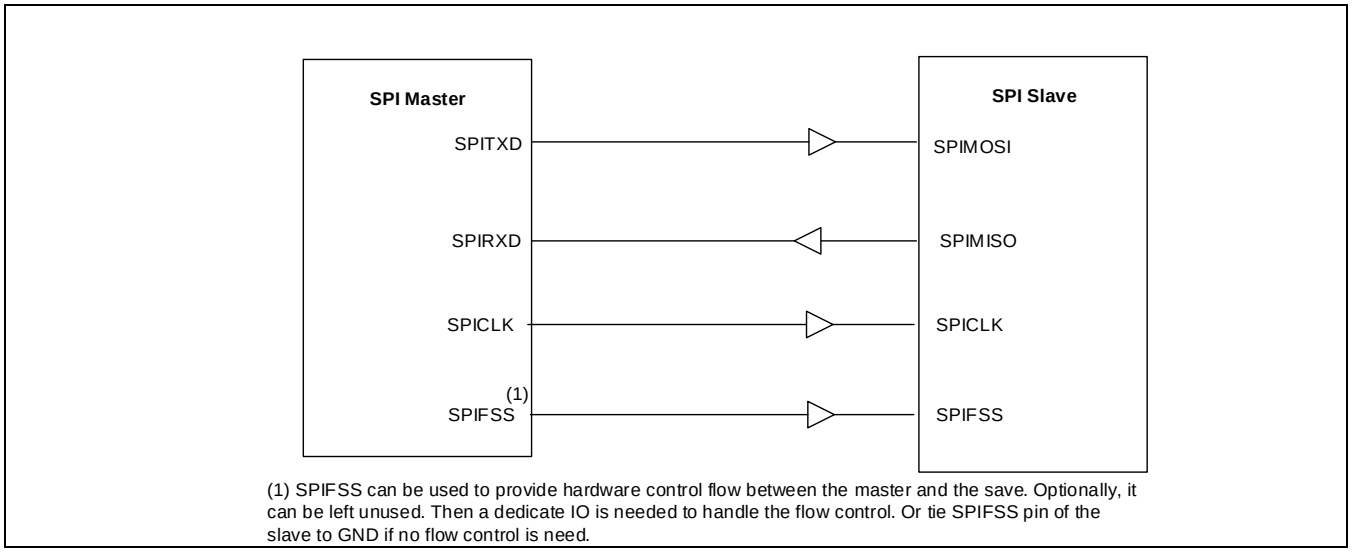


Figure 19-3 SPI主机连接1个SPI从机

上图为SPI配置成主机，连接了一个Motorola SPI从机。

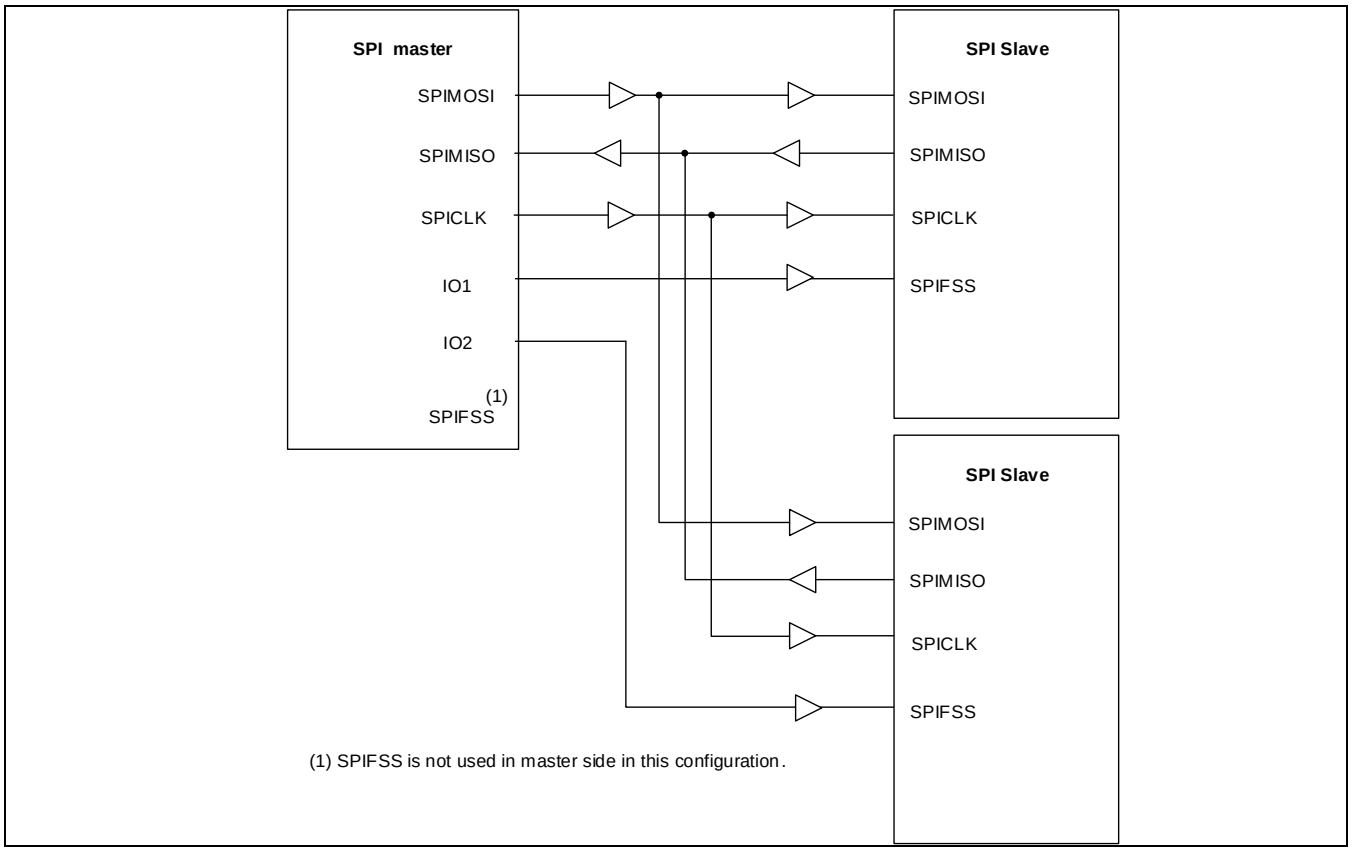


Figure 19-4 SPI 主机连接到两个SPI从机

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。

19.3.3.3 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 ≥ 12 x SPI_SCK。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 SPI_SCK。

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12MHz。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] = 12，SPI_CR0[SCR] = 0。

Table 19-2 PCLK和SPICLK的关系限制

SPI_SCK和PCLK的关系	主机模式	从机模式
	PCLK ≥ 2 x SPI_SCK	PCLK ≥ 12 x SPI_SCK

19.3.3.4 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SPI_FSS信号用作从机选择。对于所有格式，当SPI在空闲时，串行时钟(SPI_SCK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。SPI_SCK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SPI_SCK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。

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

SPO = 0，无数据传输时，SPI_SCK 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。

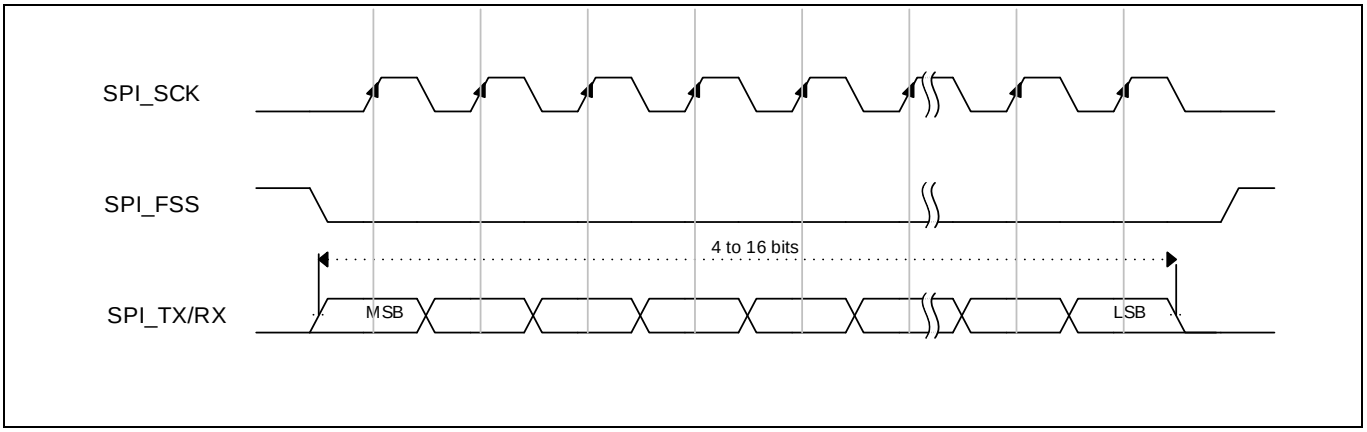


Figure 19-5 Motorola SPI 帧格式 (单帧传输) SPO = 0, SPH = 0

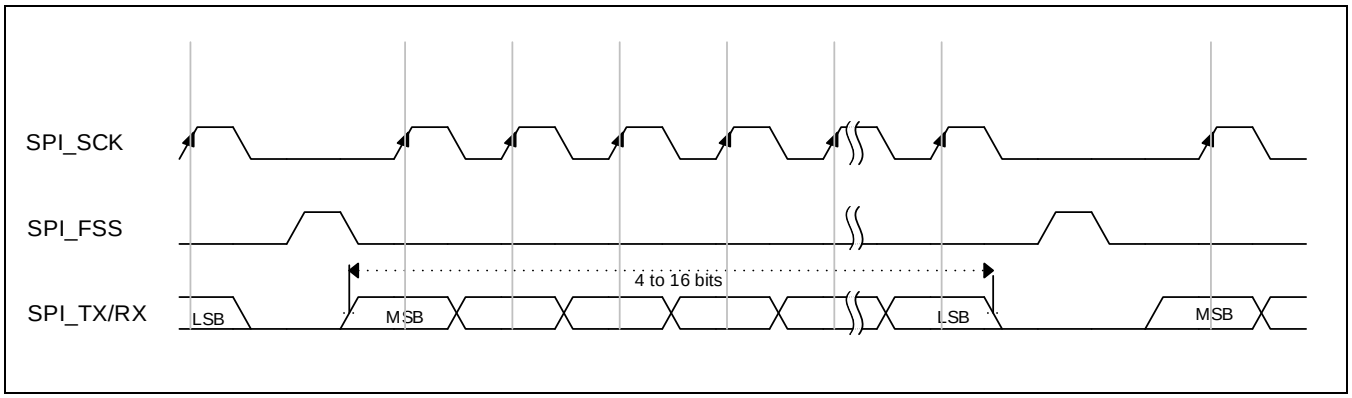


Figure 19-6 Motorola SPI 帧格式 (连续传输) SPO = 0, SPH = 0

这个配置中，在空闲时间：

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

在单字传输的情况下，当所有数据位都传输完后，SPI_FSS信号线在最后一位被捕捉后的一个SPI_SCK周期后回到它的空闲状态 - 高电平。但是在连续的传输中，SPI_FSS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SPI_FSS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，SPI_FSS管脚在最后一位被捕捉后的一个SPI_SCK周期后又会回到它的空闲状态。

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

SPO = 0， 无数据传输时，SPI_SCK 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。

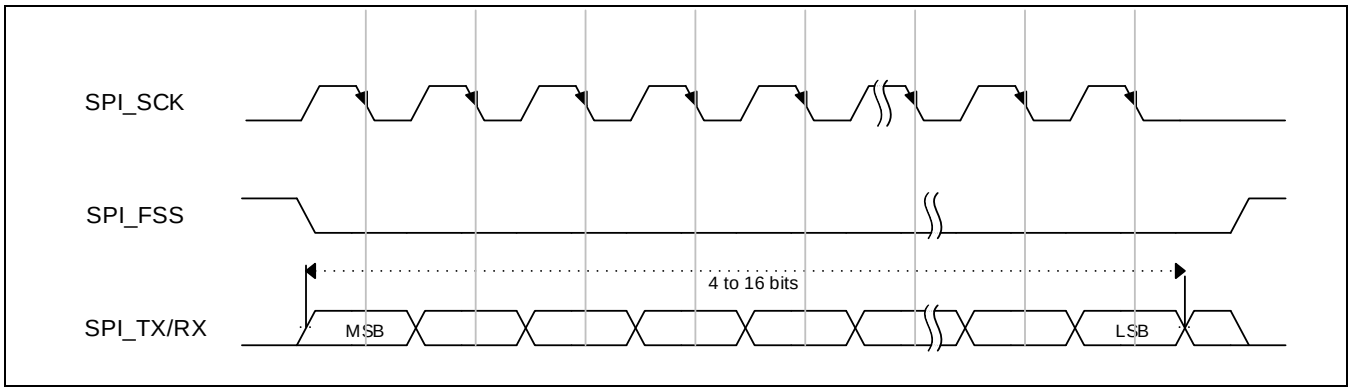


Figure 19-7 Motorola SPI 帧格式（单帧传输）， SPO = 0 和 SPH = 1

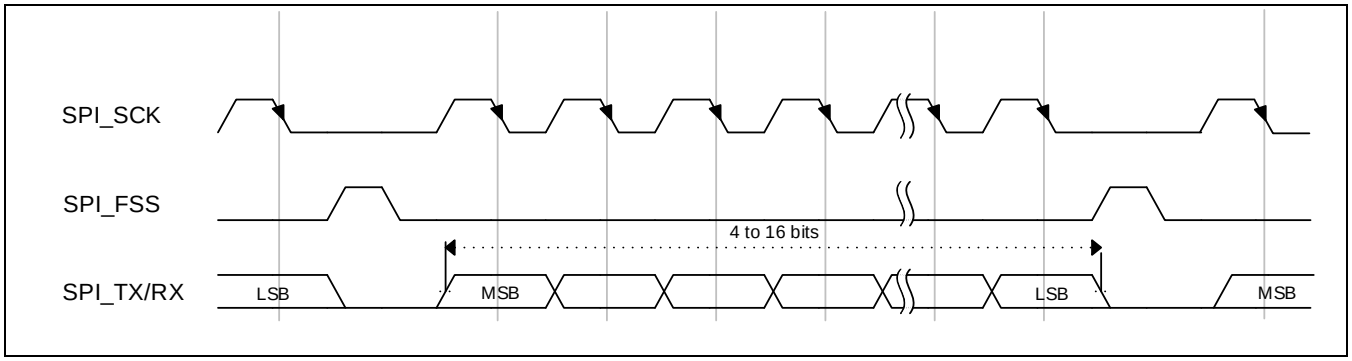


Figure 19-8 Motorola SPI 帧格式（连续传输）， SPO = 0 和 SPH = 1

这个配置中，在空闲时间：

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

在单次传输的情况下，当所有数据位都传输完后，SPI_FSS信号线在最后一位被捕捉后的一个SPI_SCK周期后回到它的空闲状态 - 高电平。在连续的传输中，SPI_FSS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。

19.3.3.4.3 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，SPI_SCK 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。

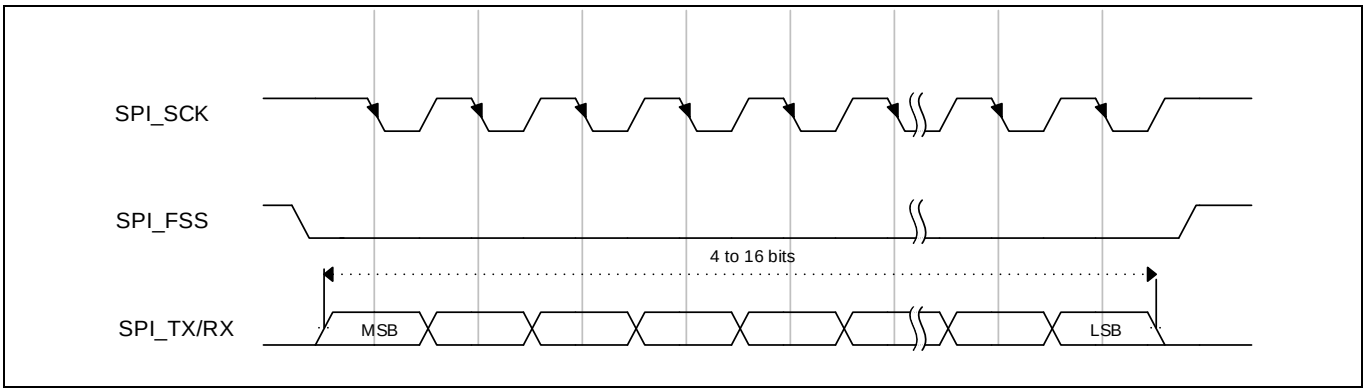


Figure 19-9 Motorola SPI 帧格式 (单帧传输)，SPO = 1 和 SPH = 0

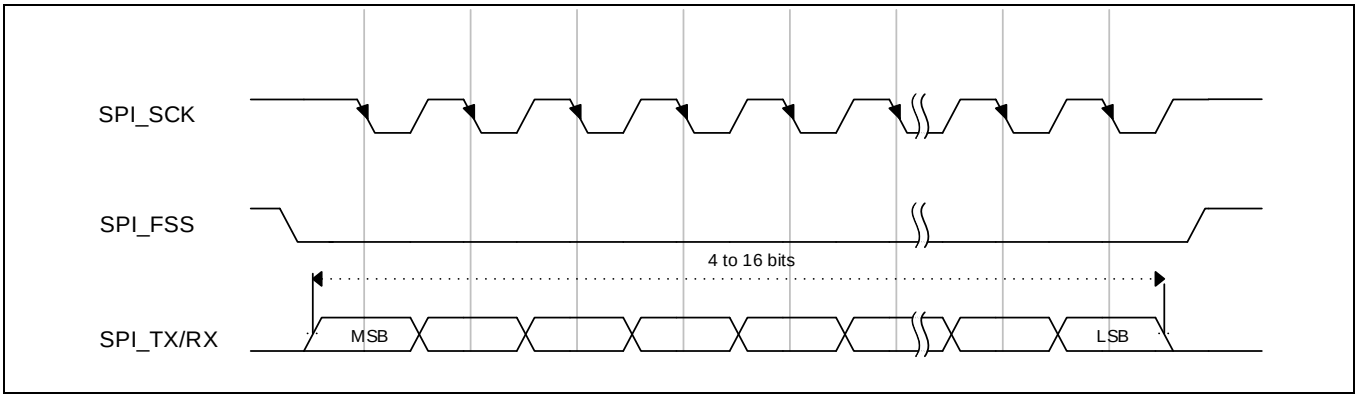


Figure 19-10 Motorola SPI 帧格式 (连续传输)， SPO = 1 和 SPH = 0

这种情况下，在空闲时间：

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

在单次传输的情况下，当所有数据位都传输完后，SPI_FSS信号线在最后一位被捕捉后的一个SPI_SCK周期后回到它的空闲状态 - 高电平。但是在连续的传输中，SPI_FSS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SPI_FSS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，SPI_FSS管脚在最后一位被捕捉后的一个SPI_SCK周期后又会回到它的空闲状态。

19.3.3.4.4 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，SPI_SCK 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。

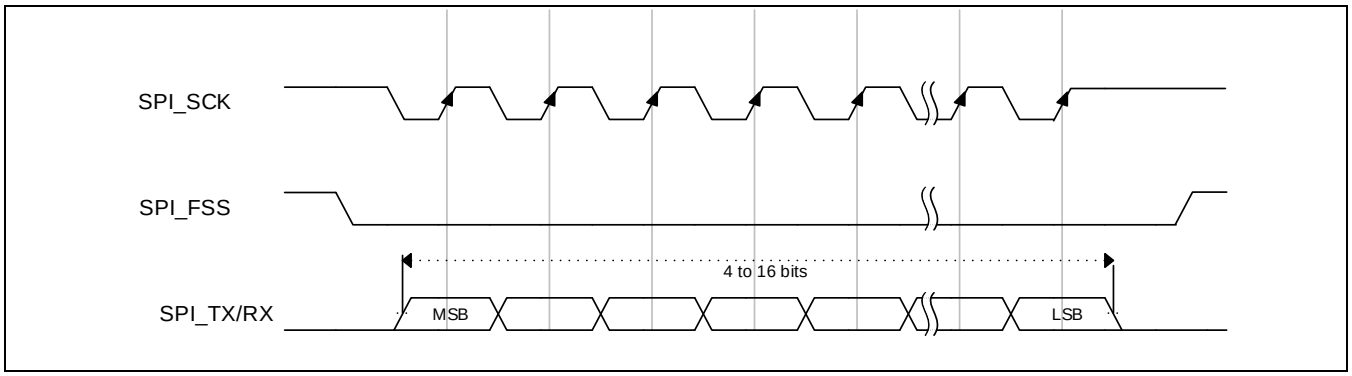


Figure 19-11 Motorola SPI 帧格式(单帧传输)， SPO = 1 和 SPH = 1

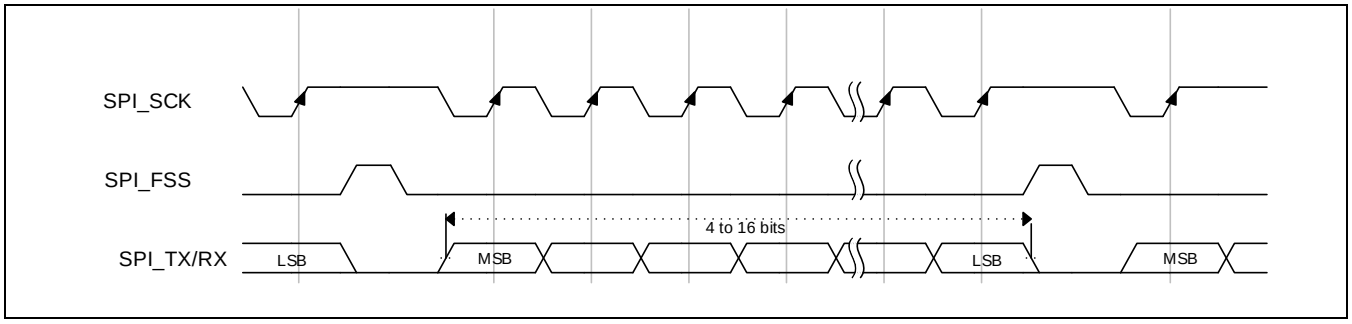


Figure 19-12 Motorola SPI 帧格式(连续传输)， SPO = 1 和 SPH = 1

这种情况下，在空闲时间：

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

在单次传输的情况下，当所有数据位都传输完后，SPI_FSS信号线在最后一位被捕捉后的一个SPI_SCK周期后回到它的空闲状态 - 高电平。在连续的传输中，SPI_FSS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。

19.3.3.5 使能SPI传送

要启动SPI的发送接收，可以往发送FIFO里写8个16位宽的数据，或者允许发送FIFO给CPU产生一个中断。一旦SPI被使能，数据的发送或者接收就会立即在SPI_TX和SPI_RX管脚上启动。

19.3 寄存器说明

19.3.1 寄存器表

Base Address of SPI0: 0x40090000

Base Address of SPI1: 0x40091000

Register	Offset	Description	Reset Value
SPI_CR0	0x0000	SSP控制寄存器0	0x00000000
SPI_CR1	0x0004	SSP控制寄存器1	0x00000010
SPI_DR	0x0008	SSP数据寄存器	0x00000000
SPI_SR	0x000C	SSP状态寄存器	0x00000003
SPI_CPSR	0x0010	SSP时钟分频寄存器	0x00000000
SPI_IMCR	0x0014	SSP中断使能/禁止寄存器	0x00000000
SPI_RISR	0x0018	SSP中断原始状态寄存器	0x00000008
SPI_MISR	0x001C	SSP中断状态寄存器	0x00000000
SPI_ICR	0x0020	SSP中断清除寄存器	0x00000000
SPI_DMACR	0x0024	SSP软件复位寄存器	0x00000000
SPI_SRR	0x0028	SSP软件复位寄存器	0x00000000

19.3.2 SPI_CR0(SSP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																SCR								SPH	SPO	FRF		DSS			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

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

19.3.3 SPI_CR1(SSP控制寄存器1)

Address = Base Address+ 0x0004, Reset Value = 0x00000010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																							LPTXOE	LPMD	RXIFLSEL			RSVD	MS	SSE	LBM
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	R	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
RXIFLSEL	[6:4]	RW	接收FIFO中断触发点选择位 001 接收FIFO占用>= 1/8 010 接收FIFO占用>= 1/4 100 接收FIFO占用>= 1/2 Others = 保留
MS	[2]	RW	主机或者从机模式 0 = 配置为主机 1 = 配置为从机
SSE	[1]	RW	SSP使能位 0 = SSP禁止 1 = SSP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

19.3.4 SPI_DR(SSP数据寄存器)

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																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

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SSPDR时，接收FIFO的数据(当前FIFO读指针所指的值)被读取。当SSP接收逻辑把接收到的数据移除时，该数据会被存到接收FIFO中当前读指针所指的空间。
写SSPDR时，数据被写入发送FIFO中当前指针所指的空间。发送逻辑每发送一次就将发送FIFO中的数据移除一个，移除的数据载入到发送串行移位寄存器，以配置好的比特率串行发送到SSPTXD管脚。
当数据位小于16时，用户在写入发送FIFO的时候必须右对齐，发送逻辑会忽略没用到的高位。 接收到的数据如果小于16位，在接收缓冲区内也是右对齐的。

19.3.5 SPI_SR(SSP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	SSP工作状态标志位 0 = SSP空闲 1 = SSP正在发送并且/或者正在接收，或者发送FIFO非空
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

19.3.6 SPI_CPSR(SSP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																								CPSDVSR							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CPSDVSR	[7:0]	RW	时钟分频位 必须是2到254之间的偶数，根据PCLK的频率来决定。在读取的时候最低位总是返回0

时钟分频位

必须是2到254之间的偶数，根据PCLK的频率来决定。在读取的时候最低位总是返回0。

SSPCPSR位时钟预分频寄存器，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。

19.3.7 SPI_IMCR(SSP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																												TXIM	RXIM	RTIM	RORIM
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
TXIM	[3]	RW	发送FIFO中断 0 = 禁止发送FIFO中断 1 = 使能发送FIFO中断
RXIM	[2]	RW	接收FIFO中断 0 = 禁止接收FIFO中断 (1/2, 1/4, or 1/8可选) 1 = 使能接收FIFO中断 (1/2, 1/4, or 1/8可选)
RTIM	[1]	RW	接收超时中断 0 = 禁止RxFIFO超时中断 1 = 使能RxFIFO超时中断
RORIM	[0]	RW	接收溢出中断 0 = 禁止RxFIFO溢出中断 1 = 使能RxFIFO溢出中断
读此寄存器返回相关中断的使能状态。写1使能相应中断，写0则禁止相应中断。			

19.3.8 SPI_RISR(SSP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																												TXRIS	RXRIS	RTRIS	RORRIS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
TXRIS	[3]	R	发送FIFO中断原始状态 SSPTXINTR中断的原始状态，不管该中断使能与否
RXRIS	[2]	R	接收FIFO中断原始状态 SSPRXINTR中断的原始状态，不管该中断使能与否
RTRIS	[1]	R	接收超时中断原始状态 SSPRTINTR中断的原始状态，不管该中断使能与否
RORRIS	[0]	R	接收中断原始状态 SSPRORINTR中断的原始状态，不管该中断使能与否
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。			

19.3.9 SPI_MISR(SSP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																												TXMIS	RXMIS	RTMIS	ROMIS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
TXMIS	[3]	R	发送FIFO中断原始状态 SSPTXINTR中断的原始状态，该中断使能后才能读到，否则一直为0
RXMIS	[2]	R	接收FIFO中断原始状态 SSPRXINTR中断的原始状态，该中断使能后才能读到，否则一直为0
RTMIS	[1]	R	接收超时中断原始状态 SSPRTINTR中断的原始状态，该中断使能后才能读到，否则一直为0
ROMIS	[0]	R	接收中断原始状态 SSPRORINTR中断的原始状态，该中断使能后才能读到，否则一直为0
读该寄存器返回相应中断的状态，该中断使能后才能读到，否则一直为0。写寄存器无效。			

19.3.10 SPI_ICR(SSP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																														RTIC	RORIC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W

Name	Bit	Type	Description
RTIC	[1]	W	接收超时中断清除 0 = 无效 1 = 清除SSPRTINTR中断
RORIC	[0]	W	接收溢出中断清除 0 = 无效 1 = 清除SSPRORINTR中断
写1清除该中断，写0无效。			

19.3.11 SPI_DMACR(SSP软件复位寄存器)

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																												TXMODE	RXMODE	TXDMAEN	RXDMAEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	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
TXMODE	[3]	RW	TX DMA MODE 0=发送FIFO未满，产生DMA数据请求 1=发送FIFO数据占用<=1/2时，产生DMA数据请求
RXMODE	[2]	RW	RX DMA MODE 0=接收FIFO非空，产生DMA数据请求 1=接收FIFO数据达到中断触发点，产生DMA数据请求
TXDMAEN	[1]	RW	TX DMA使能
RXDMAEN	[0]	RW	RX DMA使能

19.3.12 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	SSPCLK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

20 I2C总线

20.1 概述

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

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

I2C 接口可以工作在标准模式，快速模式和超快速模式(或称快速模式 plus)。标准模式支持的波特率范围为 0 到 100Kbit/s，快速模式支持的波特率范围为 0 到 400Kbit/s，超快速模式支持的波特率范围为 0 到 1000Kbit/s。该模块支持 4 种模式：主机发送，主机接收，从机发送，从机接收；支持 7 位寻址和 10 位寻址，并且支持检测本机地址功能和 General Call 寻址功能(从机模式)。

注：如果系列内芯片不具有某一外围，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。

20.1.1 主要特性

- 主机或者从机工作模式，支持多主机总线
- 串行，8位的双向数据传输
- 两种速度：
  - 标准模式 (0 到 100Kbits/s)
  - 快速模式 (<=400Kbit/s) 或者超快速模式 (<=1000Kbit/s)
- 7位或者10位寻址方式
- 7位或者10位地址组合传输
- 从机发送模式下支持大量传输模式
- 支持发送和接收缓冲 (FIFO)
- 可编程的SDA保持时间
- 支持总线清除功能

20.1.2 管脚描述

Table 20-1 管脚描述

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

20.2 功能描述

20.2.1 模块框图

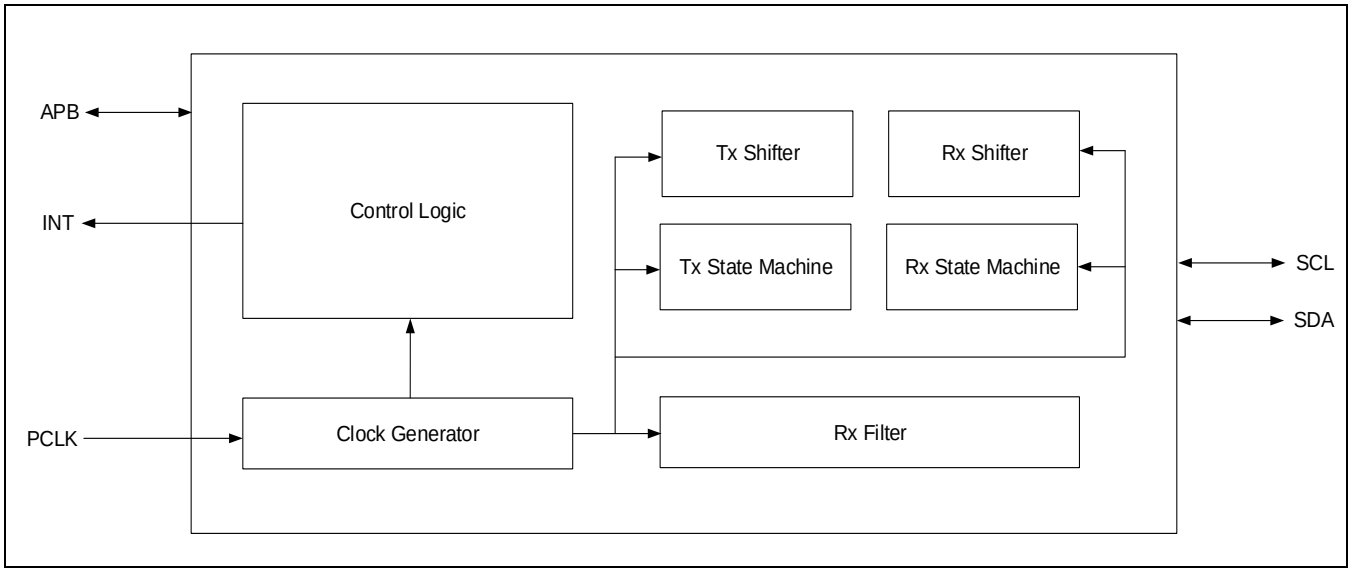


Figure 20-1 I2C模块框图

20.2.2 功能介绍

20.2.2.1 I2C总线概念

20.2.2.1.1 协议概念

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

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

1: 假设单片机A希望给单片机B发送信息

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

2: 如果单片机A希望从单片机B接收数据

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

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

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

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

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

20.2.2.2 位传输

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

20.2.2.2.1 数据有效性

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

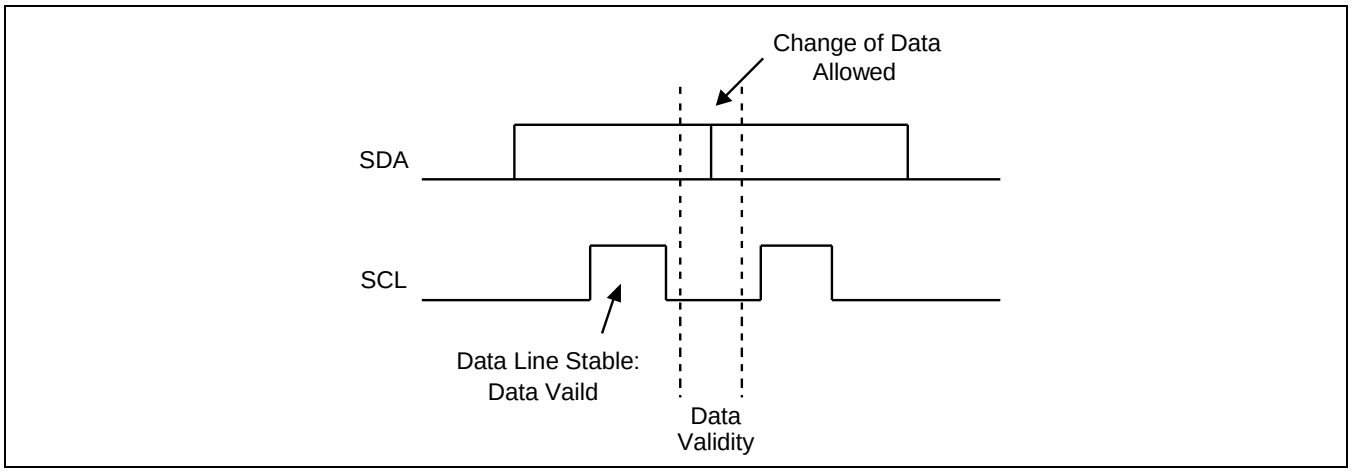


Figure 20-2 数据有效性

20.2.2.2.2 起始位和停止位

在I2C总线的传输过程中，一些特殊的情况被定义成起始位和停止位。

当SCL是高的时候，SDA从高变低，被定义为起始位。

当SCL是高的时候，SDA从低变高，被定义为停止位。

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

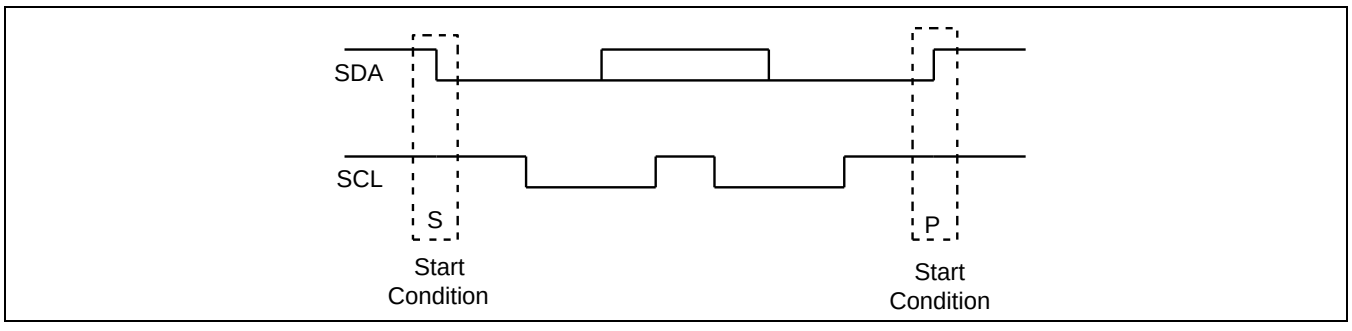


Figure 20-3 起始位和停止位

20.2.2.3 数据传输

20.2.2.3.1 传输字节格式

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

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

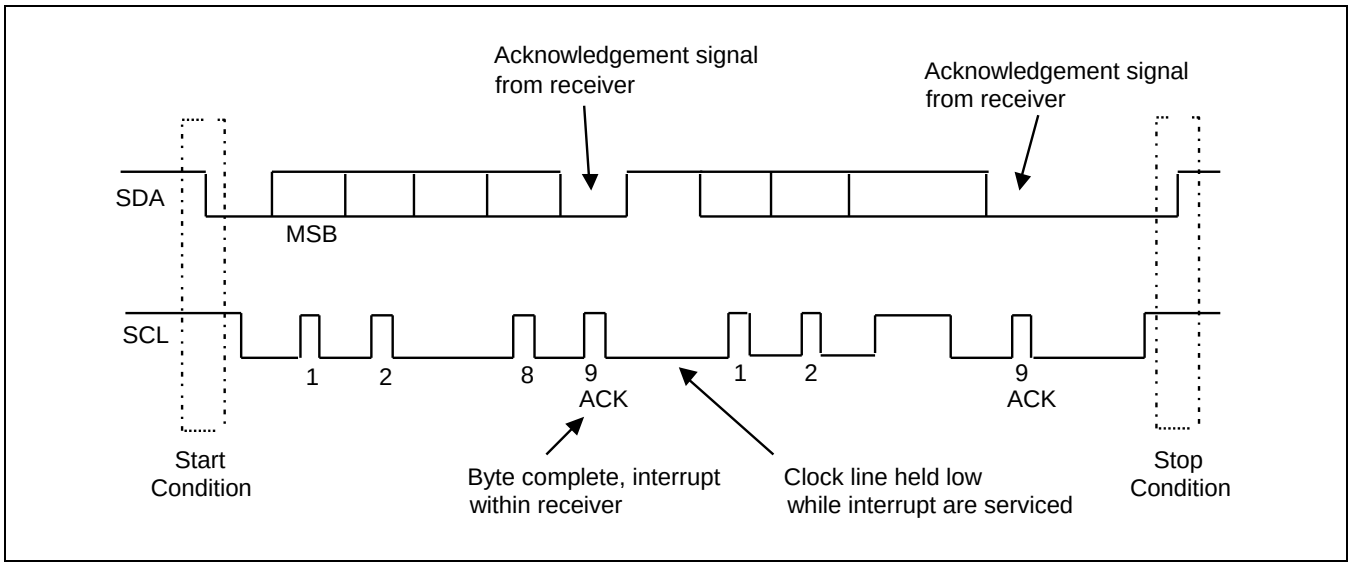


Figure 20-4 I2C总线的数据传输

20.2.2.3.2 应答

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

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

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

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

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

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

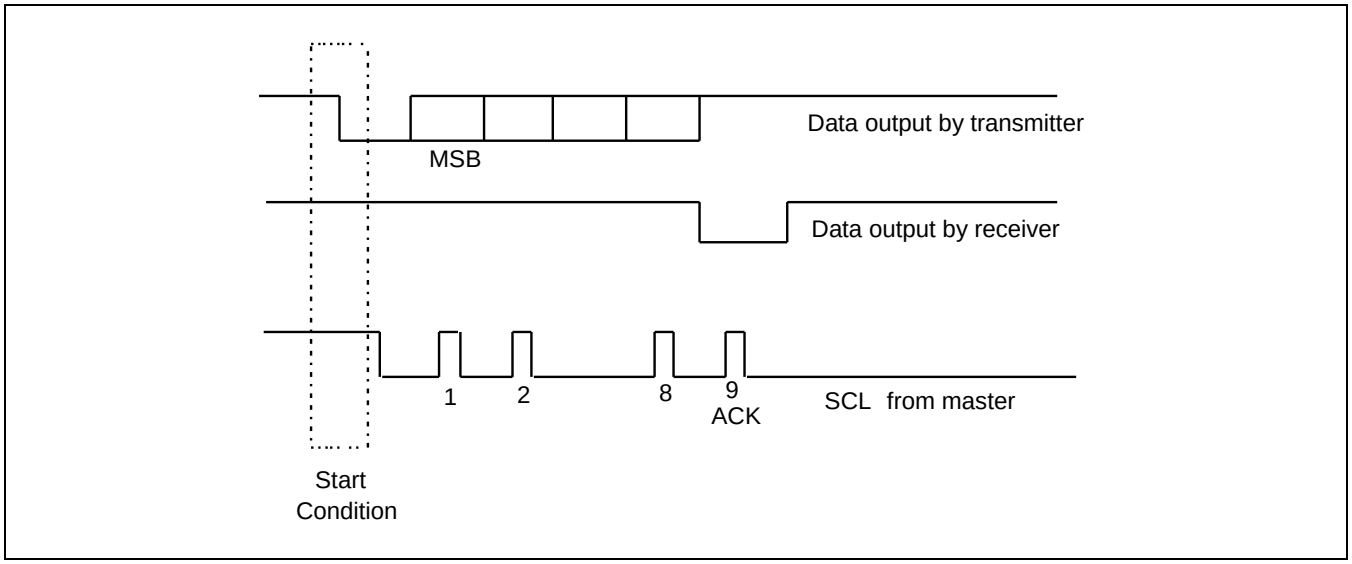


Figure 20-5 应答

20.2.2.4 时钟仲裁

20.2.2.4.1 同步

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

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

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

20.2.2.4.2 仲裁

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

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

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

失去仲裁的主机会在丢失仲裁的那个字节传输中一直产生时钟脉冲。

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

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

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

- 重复起始位和数据位
- 停止位和数据位
- 重复起始位和停止位

20.2.2.4.3 使用时钟同步机制作为握手

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

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

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

20.2.2.4.4 7位寻址格式

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

可以传输的格式为：

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

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

停止位由主机产生。

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

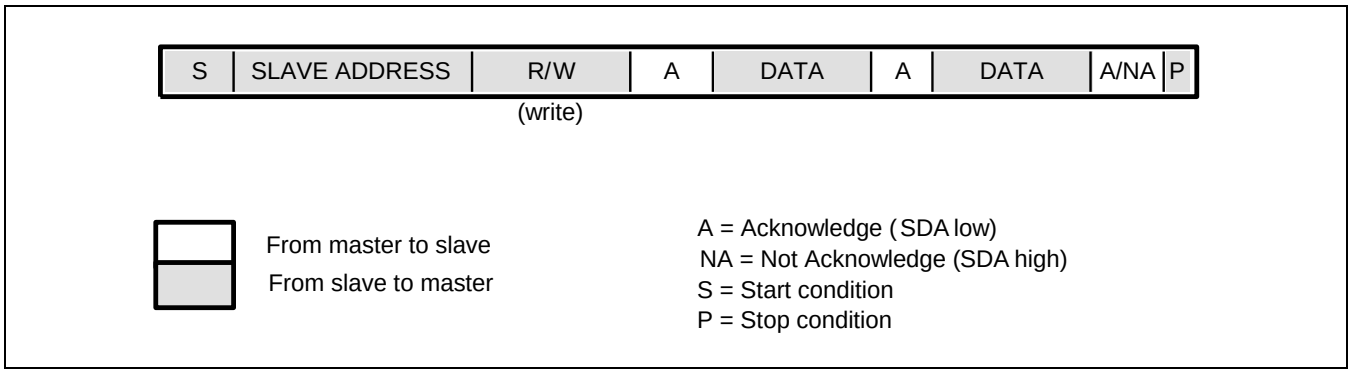


Figure 20-6 主机-发送端寻址从机

20.2.2.5 7位寻址

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

20.2.2.5.1 定义第一个字节的各个位

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

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

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

I2C总线协议委员会负责协调I2C地址的分配。

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

Table 20-2 第一个字节定义

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

注意：

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

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

第二个字节和后面的字节都会被能处理该数据的从机-接收端应答。

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

需要考虑两种情况：

- LSB最低位B是0
- LSB最低位B是1

当最低位B是0，那么第二个字节有以下定义：

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

剩下所有的代码组合都还没有定义，并且所有器件都必须忽略它们。

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

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

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

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

20.2.2.5.2 起始字节

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

在这种情况下，数据传输可以由一个比通常时间要长的起始过程来进行。

这个起始过程由下面几个步骤组成：

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

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

硬件接收端在收到重复起始位Sr后会复位，所以会忽略起始字节。

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

20.2.3 I2C总线规范的扩展

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

- 支持高达400kbit/s传输速度的快速模式和高达1000kbit/s的超快速模式(或称快速模式plus)
- 10位寻址模式，支持1024个地址空间

扩展I2C总线规范有两个原因：

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

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

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

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

20.2.4 快速模式和超快速模式

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

- 最大比特率增加到400kbit/s 或 1000kbit/s
- 串行数据SDA和串行时钟SCL信号的时序不同。不需要兼容其它总线系统比如CBUS，因为它们不能工作在这个速度。
- 工作在快速模式的器件必须在输入上抑制毛刺信号，并且输入端需要施密特触发器。
- 工作在快速模式的器件必须在输出端设计SDA和SCL信号的下降沿斜率控制。
- 如果工作在快速模式的器件掉电了，那么SDA和SCL的IO管脚必须处于悬空状态，避免干扰总线。
- 接到总线上的外部上拉器件必须适配快速模式的I2C总线所允许的信号上升时间。对于总线负载电容小于200pF的情况，上拉器件可以是一个电阻；对于总线负载电容在200pF到400pF之间的情况，上拉器件可以是一个电流源(最大3mA)或者一个开关电阻。

20.2.4.1 10位寻址

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

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

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

A – 头两个字节的位定义

10位地址由起始位(S)或者重复起始位(Sr)后的头两个字节组成。

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

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

B – 10位寻址方式

10位寻址的传输中可能包含各种读写的组合。可能涉及到的数据传输格式有：

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

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

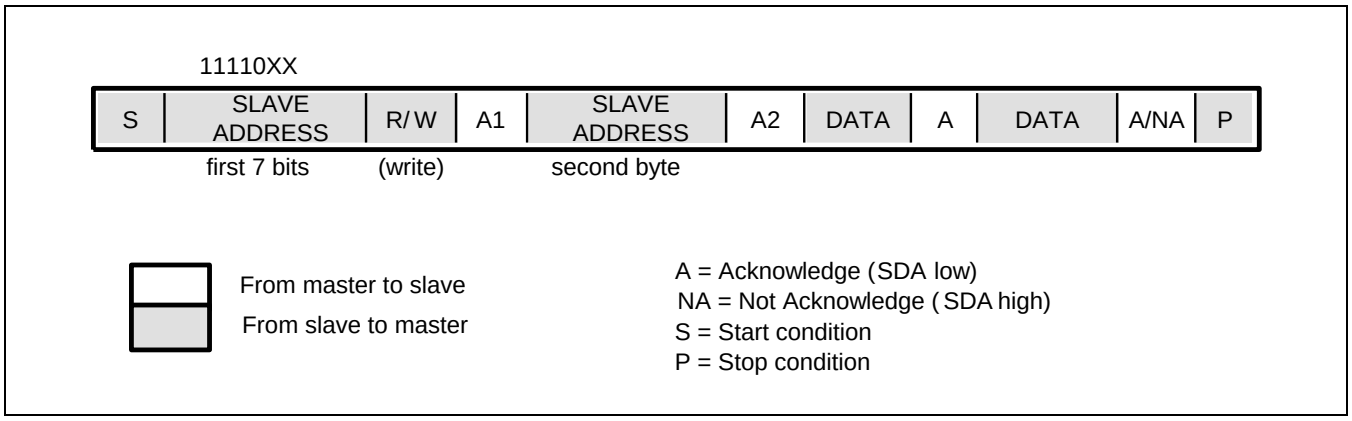


Figure 20-7 主机-发送端用10位地址寻址从机-接收端

20.2.4.2 General Call寻址和起始字节

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

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

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

10位寻址中起始字节00000001 (H’01’)的产生跟7位寻址一样。

20.2.5 发送FIFO管理

该I2C模块在发送FIFO为空的时候，不会自动产生停止位，而是将SCL拉低并保持，直到发送FIFO有新数据为止。停止位只有在用户对I2C_DATA_CMD寄存器的第9位写1发送停止指令的时候才会产生。

RESTART	STOP	CMD	DATA
10	9	8	7 ... 0

DATA - 可读写；发送和接收的数据位。

CMD - 只可写；该位决定传输的操作为读操作(CMD=1)还是写操作(CMD=0)。

STOP - 只可写；该位决定在数据被发送或者接收后，是否产生停止位。

RESTART - 只可写；该位决定在数据被发送或者接收后，是否产生重复起始位(如果重复起始位功能没有使能，那么该位决定是否在起始位后产生停止位)

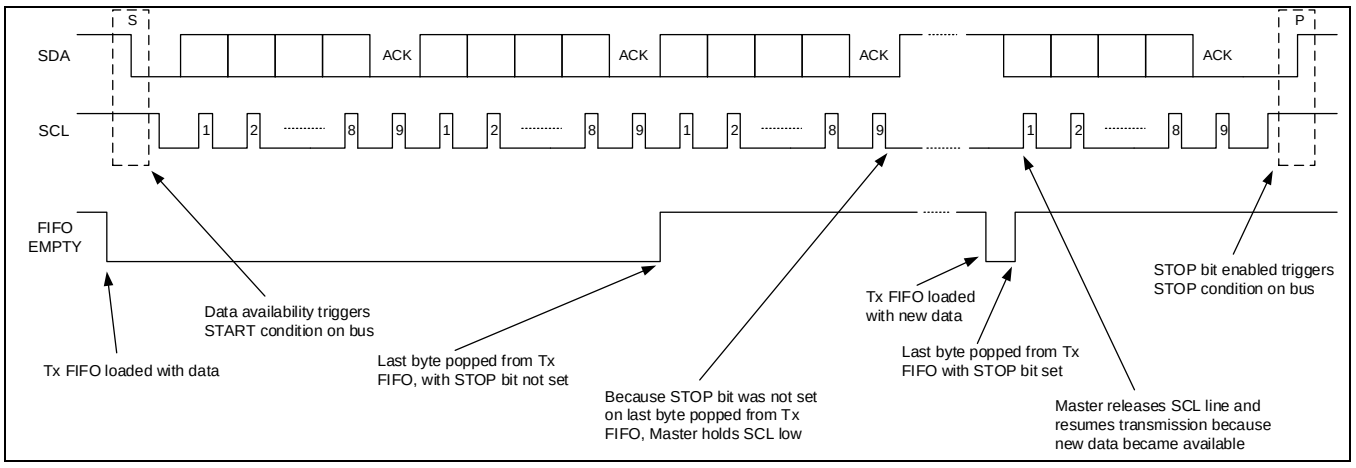


Figure 20-8 主机 – 发送 FIFO 为空 / 停止位的产生

上图说明了该I2C模块工作为主机发送端或者接收端，并且发送FIFO为空时的波形，以及停止位的产生。

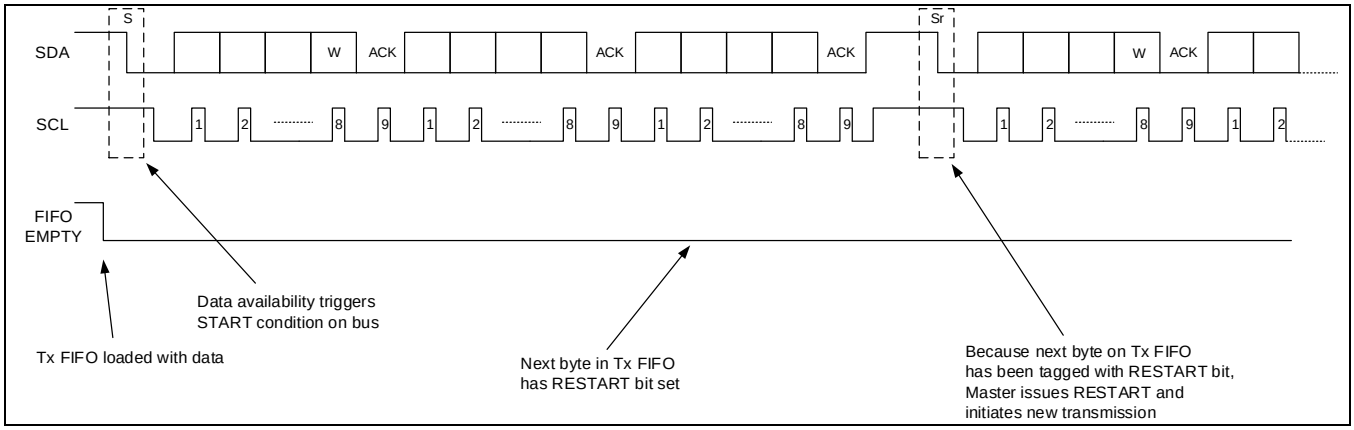


Figure 20-9 主机发送端 – I2C_DATA_CMD的RESTART位置1

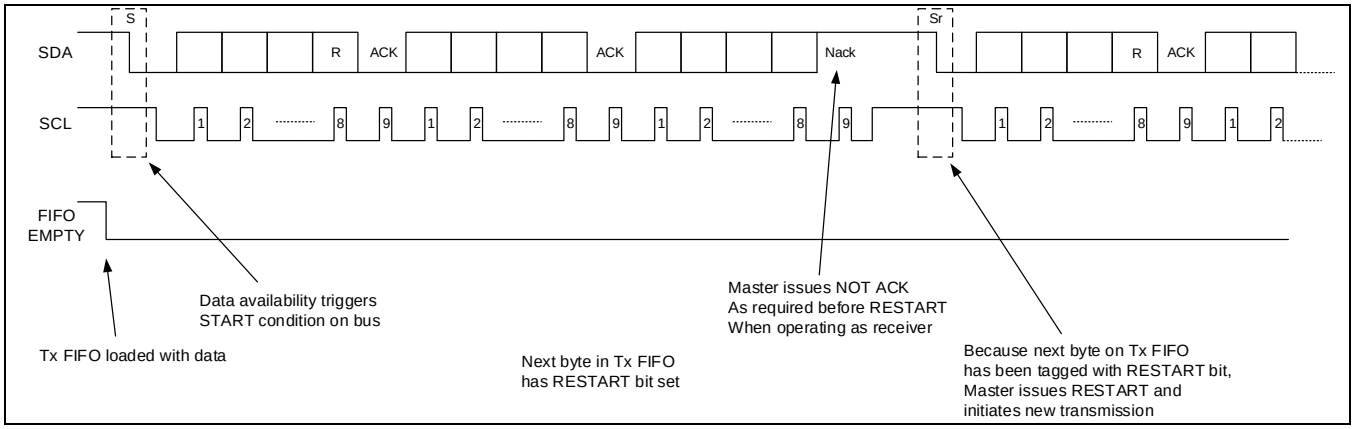


Figure 20-10 主机接收端 – I2C_DATA_CMD的RESTART位置1

上面两幅图演示了用户如何控制 I2C 总线上的重复起始位。如果 I2C_DATA_CMD 寄存器的第 10 位(RESTART)为 1 并且重复起始位功能使能(I2C_CR.RESTART_EN=1)，那么在发送数据或者读取数据前，该模块会产生一个重复起始位。如果重复起始位的功能没有被使能，那么替代重复起始位的是跟在起始位后的一个停止位。

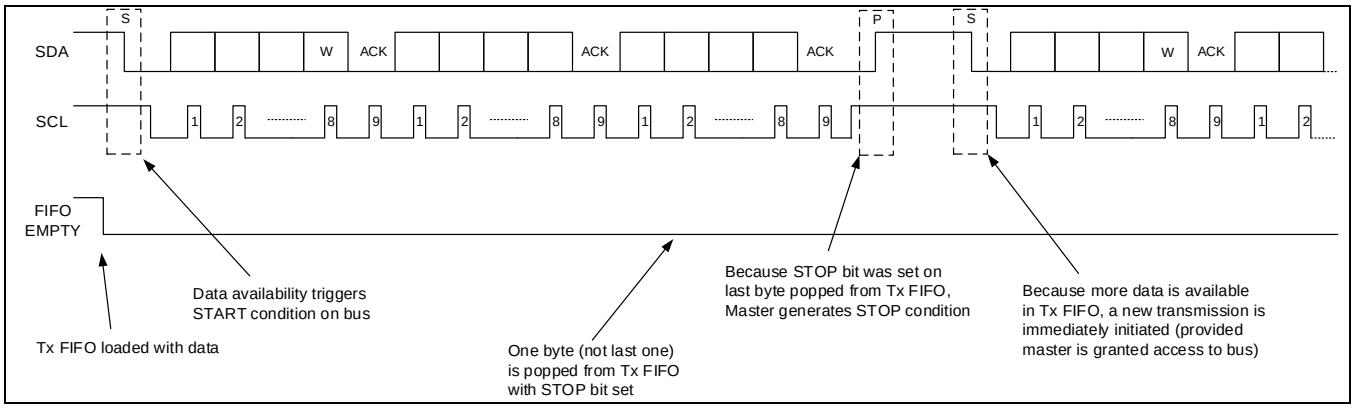


Figure 20-11 主机发送端 – I2C_DATA_CMD的停止位为1 / 发送 FIFO 非空

上图为I2C_DATA_CMD寄存器中STOP位置1并且发送FIFO非空的情况下，该I2C作为主机发送端的工作波形。

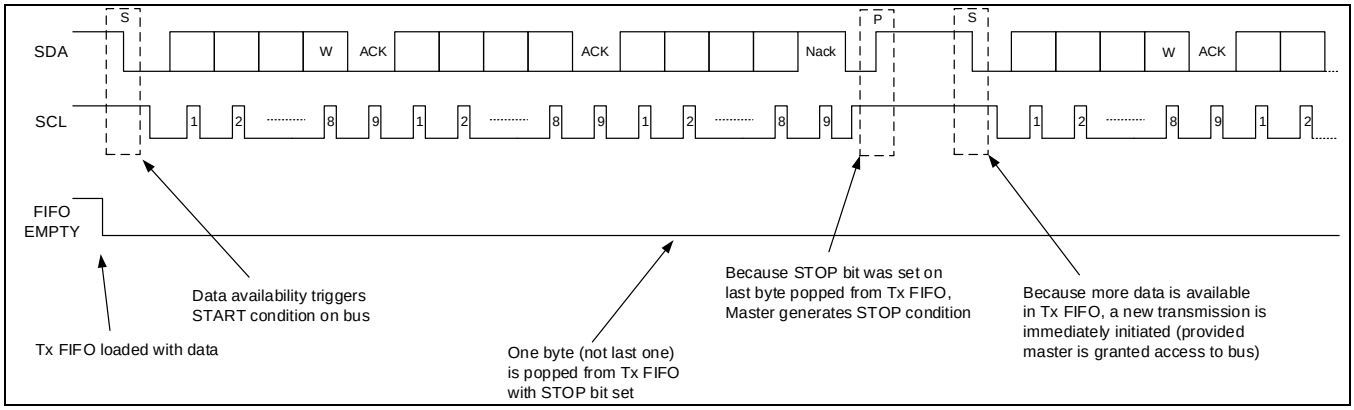


Figure 20-12 主机接收端 – I2C_DATA_CMD的停止位为1 / 发送 FIFO 非空

上图为I2C_DATA_CMD寄存器中STOP位置1并且发送FIFO非空的情况下，该I2C作为主机接收端的工作波形。

20.2.6 工作模式

20.2.6.1 主机模式

20.2.6.1.1 初始化配置

让I2C工作于主机模式，需要进行如下操作：

1. 将I2C_ENABLE寄存器的第0位写0，禁止I2C模块。
2. 配置I2C_CR寄存器，设置需要的工作速度(I2C_CR[2:1])以及寻址模式(7位或者10位)，并且确认I2C_CR的第6位(SLAVEn)为1(默认值)，第0位(MASTER)为1(默认值)。(该I2C模块的默认工作模式为主机模式。)
3. 将目标I2C器件的地址写入I2C_TADDR寄存器。这个寄存器也可以用来产生general call命令或者起始字节命令。
4. 将I2C_ENABLE寄存器的第0位写1，使能I2C模块。
5. 将传输方向和数据写入I2C_DATA_CMD寄存器。如果在I2C使能前写I2C_DATA_CMD寄存器，那么写入的数据和命令将会丢失，因为I2C在禁止状态下，所有缓存都是处于清除的状态。这个步骤会产生起始位并且发送数据字节。一旦I2C使能并且发送FIFO中有数据，I2C会开始发送数据。

注意：从机和主机不需要使用相同的寻址模式，例如作为从机时使用7位地址，而作为主机时使用10位地址。

20.2.6.1.2 主机发送和接收

该I2C模块支持在读和写之间动态切换。需要发送数据时，将数据写入I2C_DATA_CMD寄存器的低字节，并且将该寄存器的第8位写0。同时，如果需要读数据，那么只需要将I2C_DATA_CMD寄存器的第8位写1，低字节位可以为任意值。只要发送FIFO中有数据或者命令，I2C主机就会一直不停的将数据或者命令发送出去。如果数据发送完成，发送FIFO变空，那么主机根据I2C_DATA_CMD[9]这位的状态发送相应的动作：

– 如果为1，那么完成当前数据传输后，发送一个停止位。
– 如果为0，那么将SCL拉低，直到下一个数据或者命令被写入发送FIFO。

更多细节，请参考“发送FIFO管理”章节。

20.2.6.2 从机模式

20.2.6.2.1 初始化配置

让I2C工作于从机模式，需要进行如下操作：

1. 将I2C_ENABLE寄存器的第0位写0，禁止I2C模块。
2. 将从机地址写入I2C_SADDR寄存器。
3. 配置I2C_CR寄存器选择寻址模式 (7位或者10位)。将第6位(SLAVEn)写0，第0位(MASTER)写0，使能从机模式。
4. 将I2C_ENABLE寄存器的第0位写1，使能I2C模块。

注意：从机和主机不需要使用相同的寻址模式，例如作为从机时使用7位地址，而作为主机时使用10位地址。

[特别注意]

如果需要复位，那么建议在I2C总线处于空闲状态时，再对I2C从机进行复位操作。如果在总线有数据传输的时候进行复位，那么当复位结束时，内部用来同步SDA和SCL的锁存器会从复位值1变成总线上实际的值，这样就有可能导致在SCL为1的时候，SDA从1变成0，从而导致I2C从机会监测到一个错误的起始位。用户也可以先配置I2C_CR寄存器为主机模式(SLAVEn=1和MASTER=1)来避免这个问题，然后在复位结束后的大概6个PCLK同步时钟周期后再将这两位配置为0来使能从机模式。

20.2.6.2.2 从机发送端发送单个字节

当另一个总线上的I2C主机寻址到了该I2C从机并且要求数据传输，那么该I2C工作为从机发送端，需要按照以下步骤操作：

1. 另一个I2C主机以一个主机地址启动I2C传输，该地址匹配I2C_SADDR中设置的从机地址。
2. I2C从机应答该从机地址，并且识别传输的方向，知道自己是作为一个从机发送端。
3. I2C从机产生RD_REQ中断(I2C_RISR寄存器的第5位)并且将SCL拉低。I2C进入一个等待状态，直到软件响应。如果RD_REQ中断没有使能，那么I2C_MISR[5]寄存器(RD_REQ位)不会置1，只有I2C_RISR的第5位会置1，所以需要硬件或者软件的查询程序让CPU周期性的读取I2C_RISR寄存器的值。
    a. 读取I2C_RISR[5] (RD_REQ位)，判断是否为1，也就是判断RD_REQ事件是否发生。
    b. 软件必须响应并且满足I2C传输的需求。
    c. 软件查询的间隔必须是最快SCL时钟周期的10倍。例如，对于400kb/s的速度，间隔时间是25us。(推荐10这个值是因为这个值大概是I2C总线上传输单个字节所需要的时间。)
4. 如果在收到读请求(RD_REQ)之前，发送FIFO里仍然还有数据，那么I2C模块会产生一个TX_ABRT中断(I2C_RISR的第6位)，并且把发送FIFO中已有的数据清除。

    注意：因为只要TX_ABRT事件发生，I2C发送FIFO就会被强制进入清除/复位状态，所以在将数据写入发送FIFO之前，软件需要将TX_ABRT状态清除(I2C_ICR寄存器的相应位写1)。更多信息请参考I2C_RISR寄存

器。
如果TX_ABRT中断没有使能，那么I2C_MISR[6]寄存器(TX_ABRT位)不会置1，只有I2C_RISR的第6位会置1，所以需要硬件或者软件的查询程序让CPU周期性的读取I2C_RISR寄存器的值(像上一步一样)。

a. 读取I2C_RISR[6] (TX_ABRT位)，判断是否为1，也就是判断TX_ABRT事件是否发生。

b. 软件查询的间隔跟上面查询I2C_RISR[5]寄存器一样。

5. 软件将需要发送的数据写入I2C_DATA_CMD寄存器(第8位写0)。
6. 软件必须清除I2C_RISR寄存器中RD_REQ和TX_ABRT的中断状态(第5和第6位)。
7. I2C释放SCL并且发送数据。
8. 主机可能使用一个重复起始位占据I2C总线或者使用一个停止位释放总线。

20.2.6.2.3 从机接收端接收单个字节

当另一个总线上的I2C主机寻址到了该I2C从机并且向它发送数据，那么该I2C工作为从机接收端，需要按照以下步骤操作：

1. 另一个I2C主机以一个主机地址启动I2C传输，该地址匹配I2C_SADDR中设置的从机地址。
2. I2C从机应答该从机地址，并且识别传输的方向，知道自己是作为一个从机接收端。
3. I2C从机接收到数据并且将数据存入接收缓冲(接收FIFO)中。
4. I2C产生RX_FULL中断(I2C_RISR[2]寄存器)。
   如果RX_FULL中断没有使能，建议使用一个软件查询程序周期性的查询I2C_STATUS寄存器的值(参考“从机发送端发送单个字节”)。如果读到I2C_STATUS寄存器第3位RFNE的值为1，那么软件可以认为系统产生了RX_FULL中断。
5. 软件从I2C_DATA_CMD寄存器中读取接收到的数据。
6. 主机可能使用一个重复起始位占据I2C总线或者使用一个停止位释放总线。

20.2.6.2.4 从机的大量数据传输

在标准I2C协议中，所有传输都是单字节传输，远程主机端要求读取数据时，程序员将一个字节写入从机的发送FIFO中。当从机(发送端)每次从远程主机(接收端)收到一个读请求(RD_REQ)，都需要有至少有一个数据被写入从机发送端的发送FIFO中。本I2C模块设计成能处理发送FIFO中的大量数据，不需要在每次的数据请求时都产生一个中断，避免了每次中断都往发送FIFO中写数据的大量不必要的延时。

该模式只有I2C工作于从机发送端时有效。如果远程主机端应答了从机发送端发送的数据，并且从机发送FIFO中已经没有数据了，那么从机会拉低SCL并且产生一个读请求(RD_REQ)的中断，等待数据写入发送FIFO。

如果RD_REQ中断没有使能，建议使用一个软件查询程序周期性的查询I2C_STATUS寄存器的值(参考"从机发送端发送单个字节")。如果读到I2C_STATUS寄存器第5位RD_REQ的值为1，那么软件可以认为系统产生了RD_REQ中断。

RD_REQ中断是由读请求产生的，像其它中断一样，在退出中断服务子程序(ISR)的时候，该中断状态需要被清除。在ISR中允许将1个或者多个数据写入发送FIFO中。在这些数据发送到主机的过程中，如果主机应答了最后一个字节，那么从机必须再产生一个RD_REQ中断，因为主机在要求读更多的数据。

如果程序员提前知道远程主机端需要的数据包有n字节，那么当远程主机端寻址到该从机并且请求数据时，发送FIFO可以直接填入n字节的数据，让远程主机一次性连续读取出去。例如，只要远程主机一直在应答数据并且从机的发送FIFO中一直有数据时，从机会连续不停的将数据发送出去，而不是去拉低SCL等待数据或者产生RD_REQ中断。

如果远程主机需要从I2C从机读取n字节数据，但是程序员写入发送FIFO的数据超过了n字节，那么当从机完成n字节的发送后，它会清除发送FIFO，忽略多余的字节。这时，I2C会产生一个TX_ABRT事件，以表示发送FIFO被清除。在需要收到应答位时，如果收到一个非应答(NACK)位，那么说明远程主机已经完成了数据读取，这时候从机会产生一个标志位，告诉从机的工作状态机去清除发送FIFO中多余的数据。

20.2.6.3 关闭I2C

20.2.6.3.1 关闭流程

1. 定义一个 10 倍于系统中最高传输速度对应周期的时间间隔 (ti2c_poll)，例如，对于 400kb/s 的速度，这个 ti2c_poll 值为 25us.
2. 定义一个最大的超时参数, MAX_T_POLL_COUNT，如果重复查询的操作时长超过了这个最大值，那么报错。
3. 执行一个阻塞的线程/进程/函数，软件上不允许启动任何 I2C 主机传输，只允许未完成的传输结束。(如果 I2C 只配置为从机，那么此步骤可以忽略。)
4. 初始化 POLL_COUNT 变量为 0.
5. 将 I2C_ENABLE 寄存器的第 0 位置 0.
6. 读取 I2C_STATUS 寄存器并且查询第 0 位 ENABLE 位，POLL_COUNT 加 1，如果 POLL_COUNT >= MAX_T_POLL_COUNT，退出并且返回相关的错误代码。
7. 如果 I2C_STATUS[0] 为 1，那么睡眠等待 ti2c_poll 时长，然后继续上一个步骤，否则退出并且返回相关的成功代码。

20.2.6.4 中止I2C传输

I2C_ENABLE寄存器中的ABORT控制位允许软件在完成发送FIFO中的传输命令之前刷新I2C总线。控制器在收到ABORT请求后，会在I2C总线上发送一个停止位，并且随后清除发送FIFO。只有在主机模式下，允许中止传输。

20.2.6.4.1 中止流程

1. 停止对发送 FIFO (I2C_DATA_CMD)的写入。
2. 将 I2C_ENABLE 寄存器的第 1 位 (ABORT) 置 1.
3. 等待 TX_ABRT 中断。
4. 读取 I2C_TX_ABRT 寄存器确认中止源为 USER_ABRT.

20.2.6.5 干扰过滤

干扰过滤逻辑基于对输入信号(SCL 和 SDA)监控的一个计数器，检测它们在预设数量的 PCLK 周期内是否保持稳定的状态。每个信号(SCL 和 SDA)都有单独计数器。PCLK 周期的数量可以由用户设定，需要根据 PCLK 的频率和毛刺干扰信号的长度进行计算。

当计数器的输入变化时，计数器会重新开始计数。根据输入信号的行为，会有下面的情况。

- 输入信号保持不变，直到计数器计数到了预设的最大值。在这个情况下，内部信号(过滤后的输出信号)被更新成输入信号，计数器复位并且停止，直到输入信号再次变化的时候，计数器才会重新开始计数。
- 输入信号在计数器计数到预设最大值之前，发生了变化。在这个情况下，计数器复位并且停止计数，内部信号不会更新成输入信号。计数器保持停止，直到输入信号再次变化。

如下图所示：

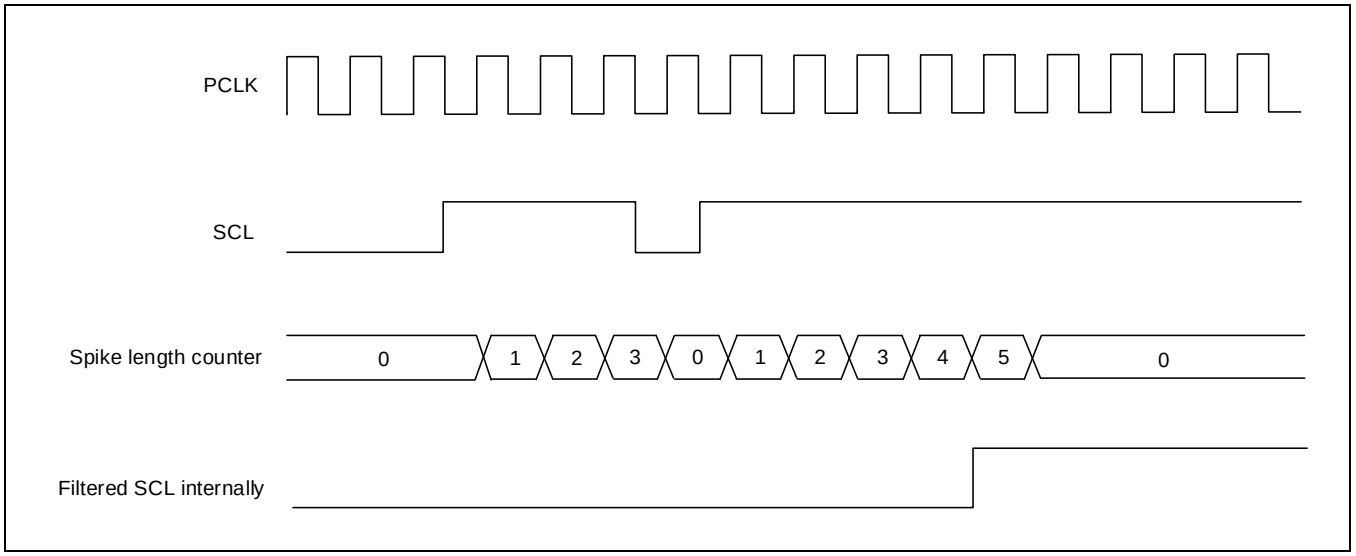


Figure 20-13 毛刺干扰信号的过滤

在图中的例子中，计数器的预设最大值为5，基于PCLK周期50ns，毛刺250ns计算得到。

注意：SCL输入上有一个两级的同步电路，但为了清楚的描述过滤电路，该同步电路的延时没有表示在上图中。

毛刺干扰过滤配置寄存器I2C_SPKLEN只有在I2C关闭的时候可写，该寄存器最小值为1（也是默认值），写0无效。

20.2.6.6 总线清除特性

该I2C模块支持总线清除的特性，当时钟(SCL)或者数据(SDA)传输线被意外拉低锁死时，可以进行恢复。

下面章节描述了SDA和SCL被拉低锁死时的恢复机制。

20.2.6.6.1 SDA拉低锁死的恢复

在SDA被锁死并且一直拉低的情况下，主机会使用下面的方法进行恢复。

1. 主机发送最大9个时钟脉冲，用来从这9个时钟造成的低电平中恢复。
   - 时钟脉冲的个数会有不同，跟从机需要发送的剩余位数有关。由于最大的位数是9，所以主机最多发9个，让从机进行恢复。
   - 主机在SDA上发送一个高电平，然后检测SDA是否恢复。如果SDA没有恢复(读到0)，那么主机会继续发送SCL，直到最大9个SCL时钟脉冲。
2. 如果SDA在9个时钟脉冲内恢复，那么主机发送一个停止位，释放总线的控制。
3. 如果在9个时钟脉冲内，SDA还没有恢复，那么系统需要一个硬件复位。

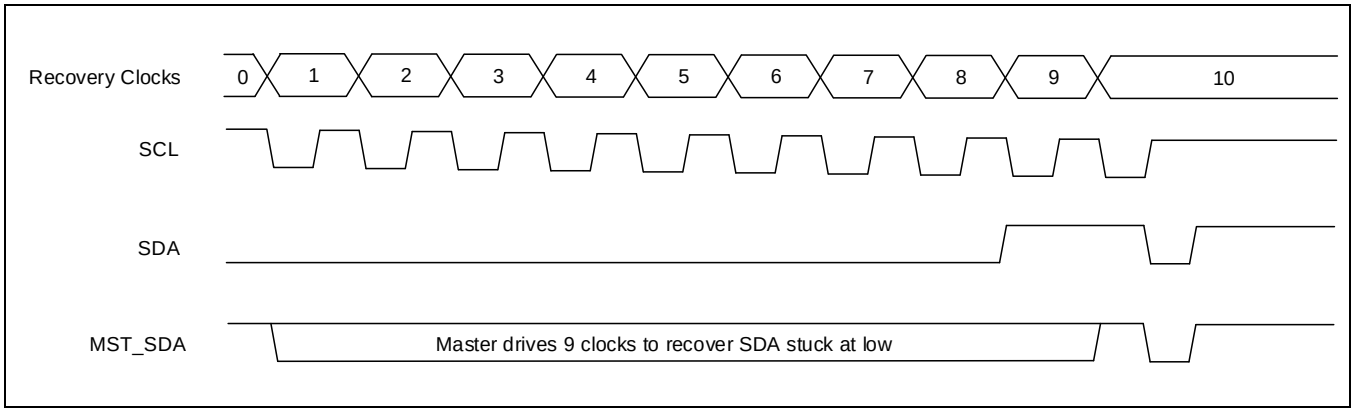


Figure 20-14 SDA在9个SCL时钟内恢复

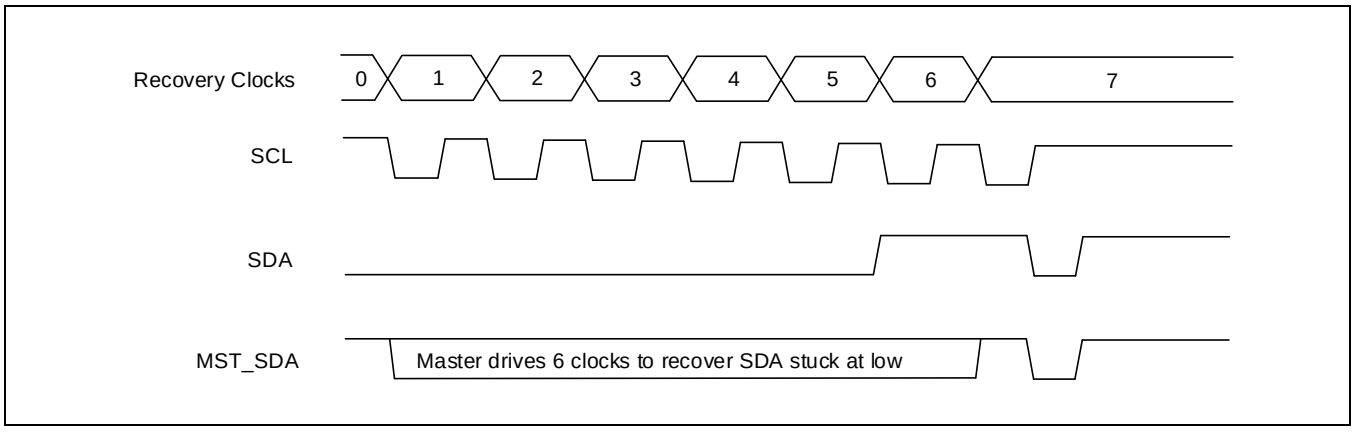


Figure 20-15 SDA在6个SCL时钟内恢复

20.2.6.6.2 SCL拉低锁死的恢复

如果时钟信号SCL被意外锁死拉低，那么除了使用硬件复位信号去复位总线意外，没有其它办法去解决。关于SCL拉低锁死恢复的详细说明，请参考编程示例。

20.2.6.7 SDA Hold时间

I2C协议标准要求标准模式和快速模式下，在SDA信号(tHD:DAT)上有300ns的hold时间，并且在快速模式或者超快速模式下hold时间足够长以覆盖SCL从1变到0下降沿的不稳定区间。

SCL和SDA信号在板上的延时会导致I2C主机上虽然满足hold时间的要求，但在I2C从机上不满足 (反之亦然)。由于每个应用的环境都不同，延时也都不同，所以该I2C模块含有一个软件可编程的寄存器(I2C_SDA_THOLD)，用来动态调节SDA的hold时间。

低16位[15:0]用来控制SDA在从机和主机模式下发送时的hold时间 (在SCL从高变低后)。

高8位[23:16]用来在主机和从机的接收状态下，扩展SDA的变化时间节点(如果有的话)。

如果不同速度模式下需要不同的SDA hold时间，那么I2C_SDA_THOLD寄存器必须在速度模式改变后重新写入。I2C_SDA_THOLD寄存器只有在I2C被禁止(I2C_ENABLE[0] = 0)的时候才可以写入。

20.2.6.7.1 SDA在接收端的hold时序

当I2C工作为接收端，根据I2C协议，设备必须在内部保持SDA的状态足够长时间，以覆盖SCL从1变道0时的不稳定区间。

SDA_RX_THOLD位用来改变内部对输入SDA信号的hold时间，寄存器的值表示以PCLK周期为单位的保持时长。SDA_RX_THOLD的最小值为0.这个hold时间只有在SCL是高电平的时候有效，接收端在SCL内部变低后就不会去扩展SDA信号了。

下图为I2C作为接收端，SDA_RX_THOLD大于等于3时的情况。

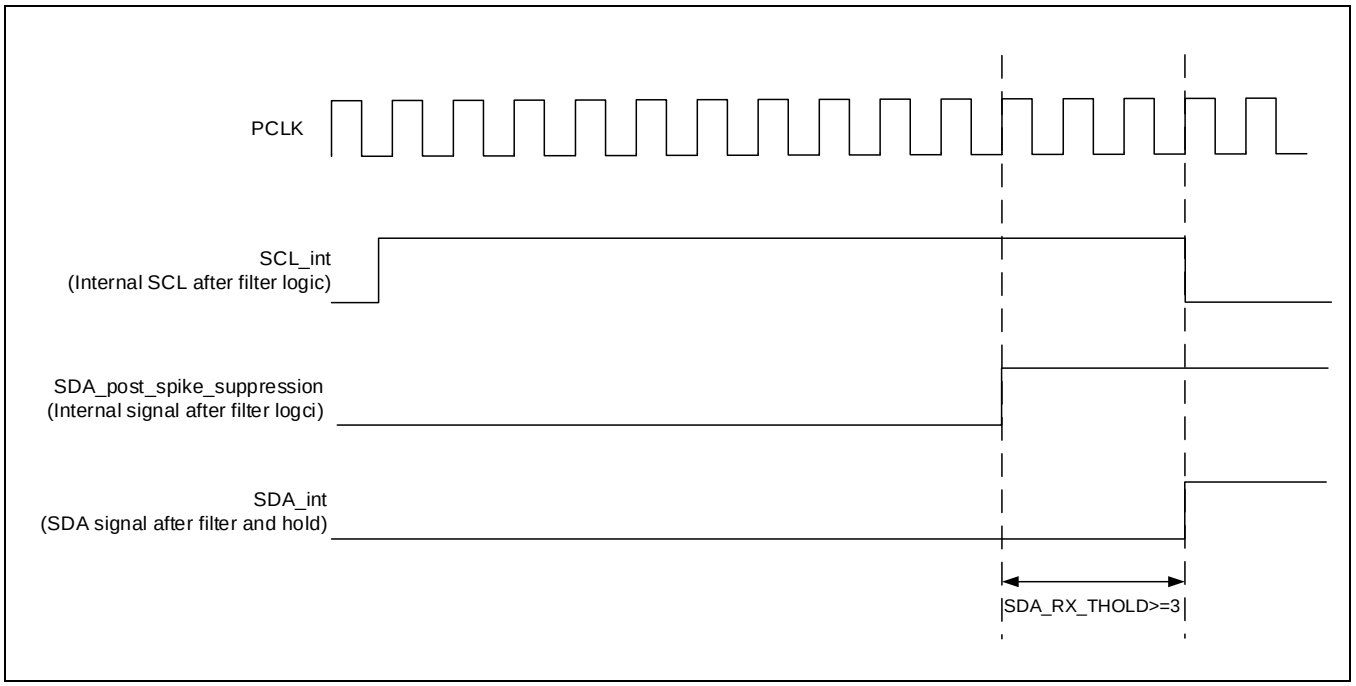


Figure 20-16 SDA Hold时序 SDA_RX_THOLD>=3

如果SDA_RX_THOLD大于3，I2C在3个PCLK周期后就不会保持SDA了，因为内部SCL已经变低。

下图为I2C作为接收端，SDA_RX_THOLD等于2时的情况。

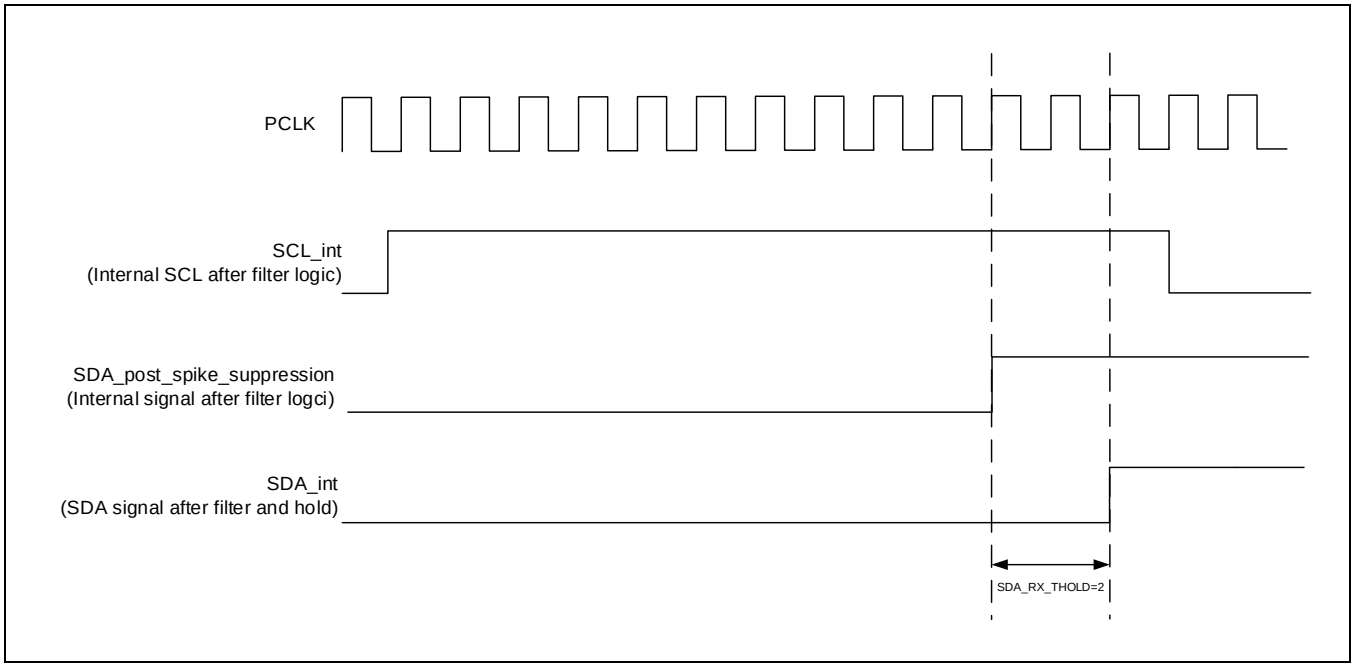


Figure 20-17 SDA Hold时序 SDA_RX_THOLD=2

SDA_RX_THOLD的最大值根据速度的不同，可以根据下面公式计算：

标准模式: I2C_SS_SCLH – I2C_SPKLEN – 3

快速模式或者超快速模式: I2C_FS_SCLH – I2C_SPKLEN – 3

注意： 最大值只在主机模式下有效。在从机模式下，必须保证 SDA_RX_THOLD 不超过SCL的下降沿。

20.2.6.7.2 SDA在发送端的Hold时序

SDA_TX_THOLD位可以用来改变I2C产生的SDA信号的时序，SDA_TX_THOLD寄存器值的单位为PCLK周期。

当I2C工作在主机模式，最小的tHD:DAT时间是1个PCLK周期。所以即使SDA_TX_THOLD的值为0，I2C也会在SCL变0后的下一个周期再驱动SDA。对于其它所有SDA_TX_THOLD的值：

– SDA的驱动输出发生在SCL变0后，再过SDA_TX_THOLD个PCLK周期

当I2C工作在从机模式，最小的tHD:DAT时间是I2C_SPKLEN + 7 PCLK周期。在这段时间内，在SCL信号上需要进行同步和毛刺噪声的过滤，所以即使SDA_TX_THOLD的值小于I2C_SPKLEN + 7，I2C也会在SCL变0的I2C_SPKLEN + 7 PCLK周期后再驱动输出SDA。对于其它所有SDA_TX_THOLD值：

– SDA的驱动输出发生在SCL变0后，再过SDA_TX_THOLD个PCLK周期

下图为I2C工作在主机模式，SDA_TX_THOLD等于3时的tHD:DAT时序。

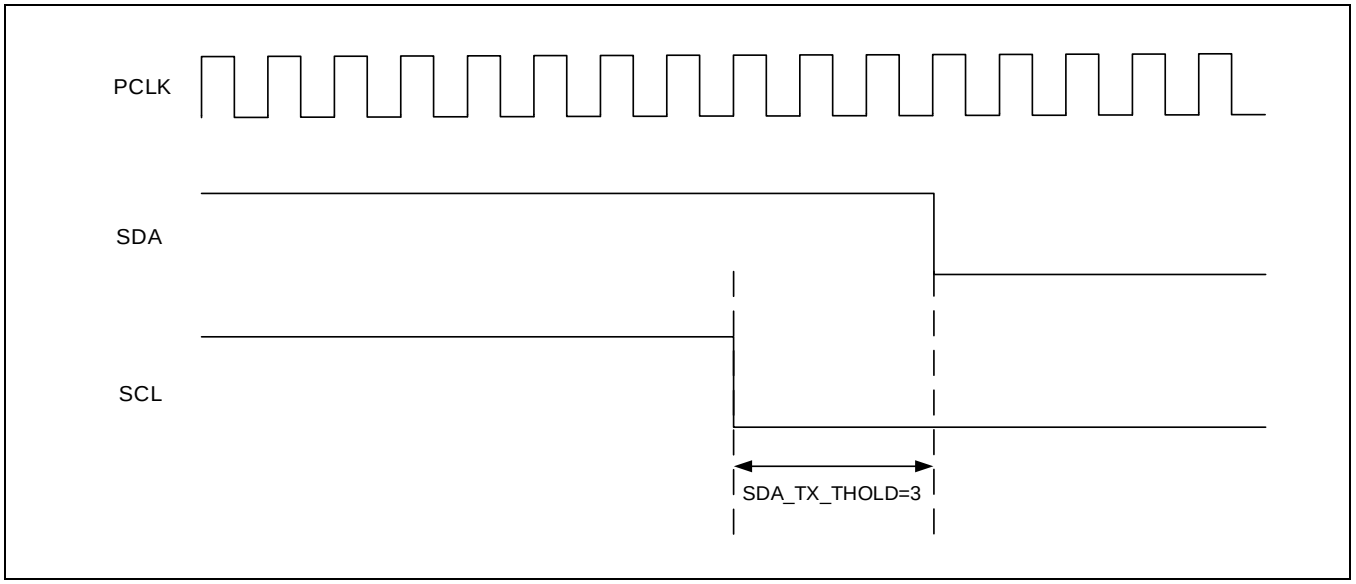


Figure 20-18 I2C主机模式SDA_TX_THOLD = 3下实现tHD;DAT

注意： 配置的SDA hold时间不能超过SCL低电平的时间长度，所以配置的值不能大于N_SCL_LOW-2，N_SCL_LOW是SCL低电平宽度，以PCLK周期为单位。

20.3 编程示例

下面的流程图为 I2C 主机模式的编程示例。

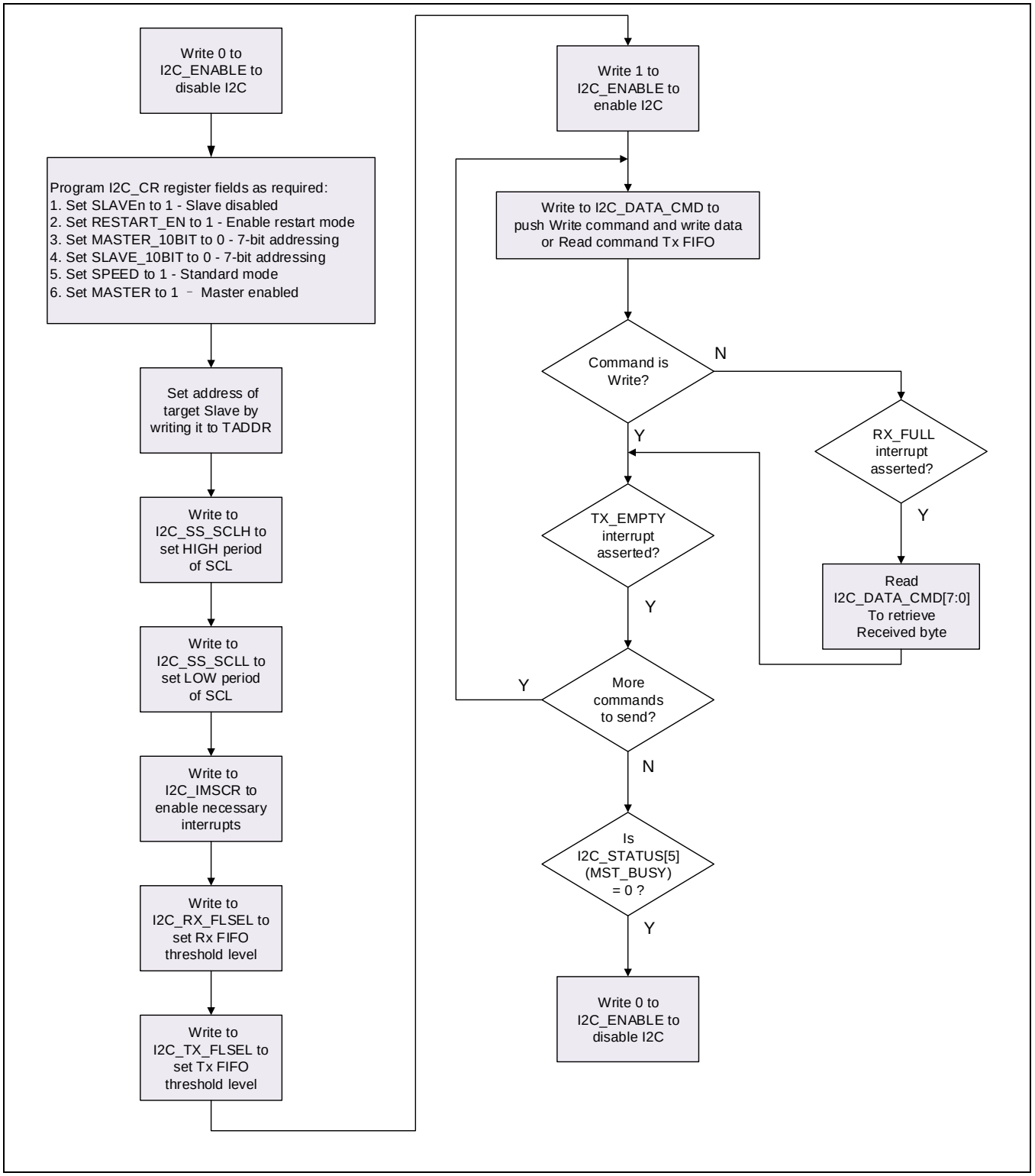


Figure 20-19 I2C主机的流程图

下面的流程图为I2C从机模式的编程示例。

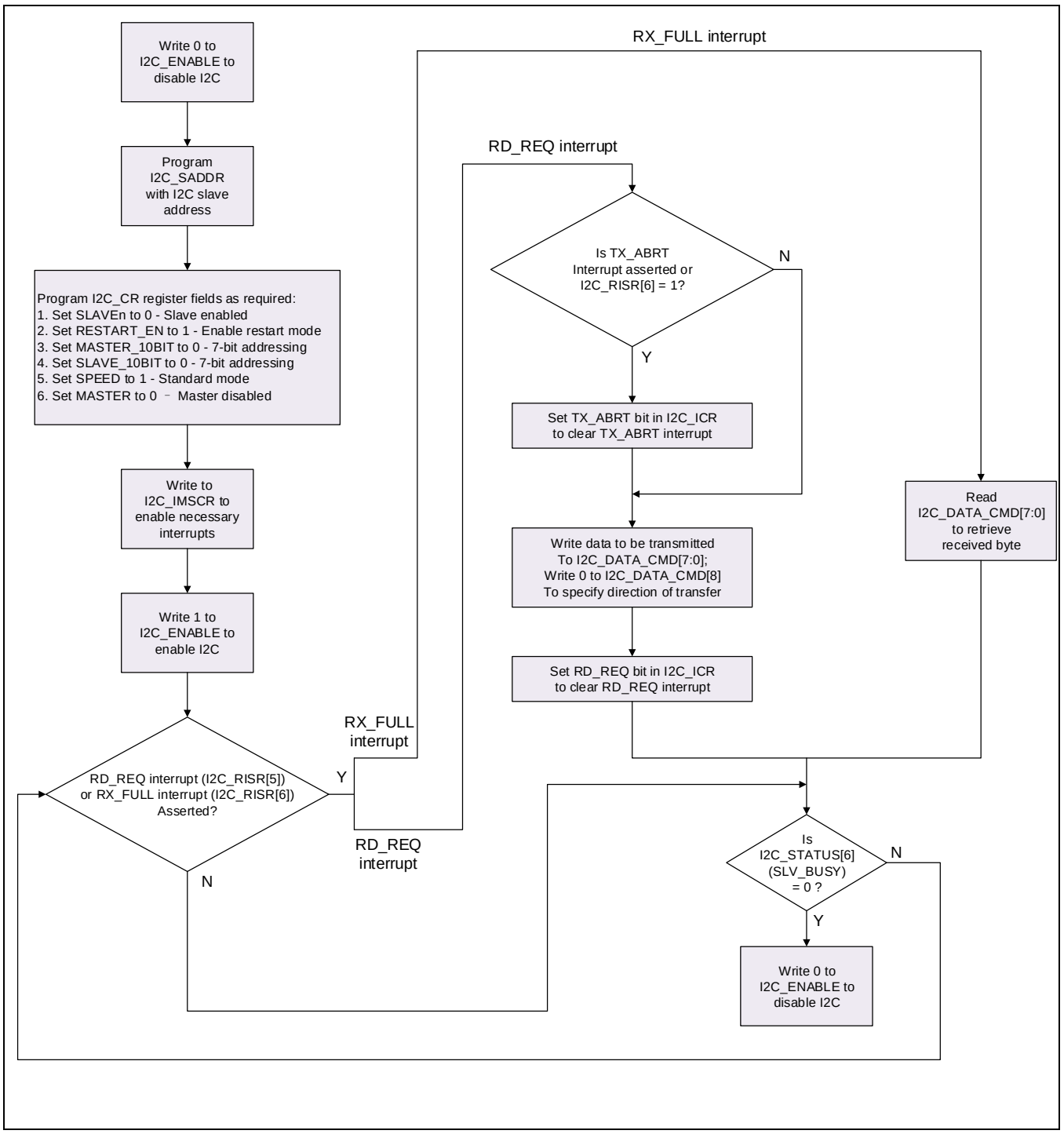


Figure 20-20 I2C从机的流程图

下面的流程图为SCL和SDA总线恢复的编程示例。

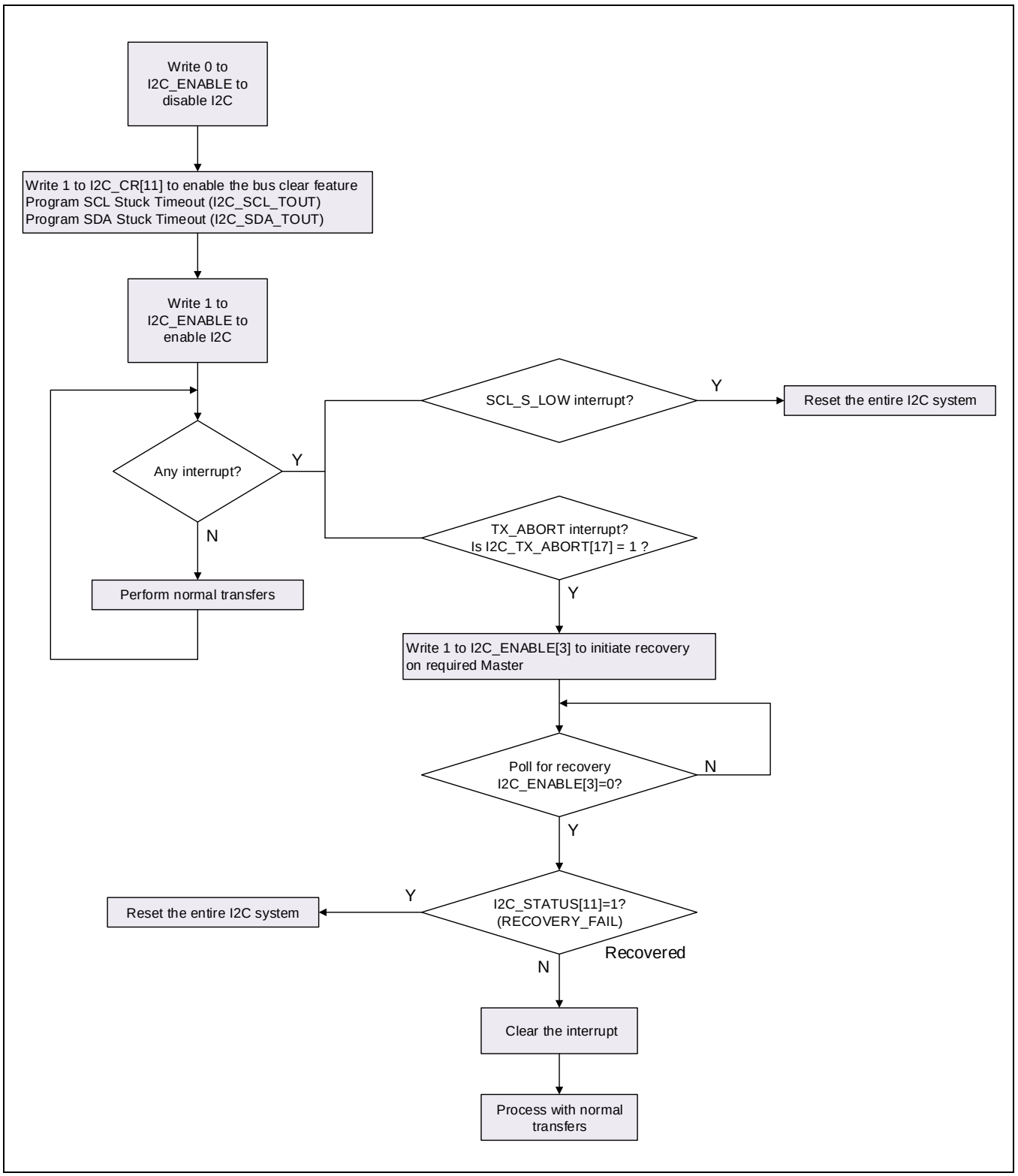


Figure 20-21 SCL和SDA总线恢复的流

20.4 寄存器说明

20.4.1 寄存器表

Base Address of I2C: 0x400A0000

Register	Offset	Description	Reset Value
I2C_CR	0x000	控制寄存器	0x0000007D
I2C_TADDR	0x004	目标地址寄存器	0x00000055
I2C_SADDR	0x008	从机地址寄存器	0x00000055
I2C_DATA_CMD	0x010	数据和命令寄存器	0x00000000
I2C_SS_SCLH	0x014	标准模式SCL高电平长度计数寄存器	0x00000050
I2C_SS_SCLL	0x018	标准模式SCL低电平长度计数寄存器	0x0000005E
I2C_FS_SCLH	0x01C	高速模式SCL高电平长度计数寄存器	0x0000000C
I2C_FS_SCLL	0x020	高速模式SCL低电平长度计数寄存器	0x0000001A
I2C_RX_FLSEL	0x02C	接收FIFO阈值寄存器	0x00000000
I2C_TX_FLSEL	0x030	发送FIFO阈值寄存器	0x00000000
I2C_RX_FL	0x034	接收FIFO状态寄存器	0x00000000
I2C_TX_FL	0x038	发送FIFO状态寄存器	0x00000000
I2C_ENABLE	0x03C	使能寄存器	0x00000000
I2C_STATUS	0x040	状态寄存器	0x00000006
I2C_SDA_TSETUP	0x048	SDA Setup时间寄存器	0x00000064
I2C_SDA_THOLD	0x04C	SDA Hold 时间寄存器	0x00000001
I2C_SPKLEN	0x050	毛刺干扰滤波控制寄存器	0x00000001
I2C_MISR	0x058	中断状态寄存器	0x00000000
I2C_IMCR	0x05C	中断使能寄存器	0x00000000
I2C_RISR	0x060	原始中断状态寄存器	0x00000000
I2C_ICR	0x064	中断清除寄存器	0x00000000
I2C_SCL_TOUT	0x06C	SCL锁死超时控制寄存器	0xFFFFFFFF
I2C_SDA_TOUT	0x070	SDA锁死超时控制寄存器	0xFFFFFFFF
I2C_TX_ABRT	0x074	发送中止状态寄存器	0x00000000
I2C_GCALL	0x078	General Call控制寄存器	0x00000001
I2C_NACK	0x07C	从机NACK控制寄存器	0x00000000
I2C_DMACR	0x080	DMA控制寄存器	0x00000000
I2C_SRR	0x08C	软件复位控制寄存器	0x00000000
I2C_SADDRQUAL	0x090	从机地址限定寄存器	0x00000000
I2C_ADDRMON	0x094	地址监控寄存器	0x00000000

20.4.2 I2C_CR(控制寄存器)

Address = Base Address+ 0x000, Reset Value = 0x0000007D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																				BUS_CLR_EN	RSVD	RXFULL_HLD	TX_EMPTY_CON	STOP_DET_CON	SLAVEn	RESTART_EN	MASTER_10BIT	SLAVE_10BIT	SPEED		MASTER
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
BUS_CLR_EN	[11]	RW	总线清除功能控制(只有在主机模式下有效)： 0: 禁用 1: 使能 从机模式下该功能不工作。
RXFULL_HLD	[9]	RW	当接收FIFO溢出时的总线控制 0: 不占用总线 1: 占用总线 (hold bus)
TX_EMPTY_CON	[8]	RW	TX_EMPTY (I2C_RISR)状态控制 0: 当发送缓冲的指针小于或则等于I2C_TX_FLSEL寄存器设置的阈值时，TX_EMPTY位置1 1: 当发送缓冲的指针小于或则等于I2C_TX_FLSEL寄存器设置的阈值，并且最近一个地址或者数据的传输完成时，TX_EMPTY位置1
STOP_DET_CON	[7]	RW	STOP_DET中断控制 (只在从机模式下有效): 0: 不管被寻址成功与否，都产生STOP_DET中断 1: 只有被成功寻址到，才产生STOP_DET 注意：在general call寻址模式下，如果STOP_DET_CON=1，即使该从机设备产生了ACK去响应general call地址，该设备也不会产生STOP_DET中断。 只有当传输地址跟从机地址(SADDR)匹配的时候，才会产生STOP_DET中断。
SLAVEn	[6]	RW	从机模式控制位 0 = 使能从机模式 1 = 禁用从机模式 注意如果此位写了0，那么第0位MASTER也必须写0。
RESTART_EN	[5]	RW	重复起始位控制 0: 禁止 1: 使能 控制主机模式下是否发送重复起始位，有些老的从机设备不支持重复起始位。
MASTER_10BIT	[4]	RW	主机模式寻址控制

			0: 7位寻址 1: 10位寻址
SLAVE_10BIT	[3]	RW	从机模式寻址控制 0: 7位寻址 1: 10位寻址
SPEED	[2:1]	RW	I2C工作速度 0x0: 无效 0x1: 标准模式 (0 到 100 Kb/s) 0x2: 高速模式 (<=400 Kb/s) 或超高速模式 (<=1000 Kb/s) 0x3: 无效
MASTER	[0]	RW	主机模式控制位 0: 禁用主机模式 1: 使能主机模式 注意如果此位写了1，那么第6位SLAVEn也必须写1.

20.4.3 I2C_TADDR(目标地址寄存器)

Address = Base Address+ 0x004, Reset Value = 0x00000055

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

Name	Bit	Type	Description
SPECIAL	[11:10]	RW	I2C地址的特殊命令控制，表示I2C的操作类型：普通地址，general call或者起始字节。 0x0: 普通I2C_TADDR地址 0x1: 普通I2C_TADDR地址 0x2: General Call地址 – 在发送General Call后，只有写操作有效，任何尝试读的操作都会导致RISR寄存器中的第6位(TX_ABRT)位变1。I2C会一直保持在General Call模式直到SPECIAL位被配置为普通地址。 0x3: 起始字节 (START BYTE)
TADDR	[9:0]	RW	主机模式下传输的目标地址。 当发送general call的时候，该位无效。 产生起始字节时，CPU只需要对该位写一次。

20.4.4 I2C_SADDR(从机地址寄存器)

Address = Base Address+ 0x008, Reset Value = 0x00000055

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

Name	Bit	Type	Description
SADDR	[9:0]	RW	当I2C工作为从机时， SADDR为从机地址。7位寻址模式下，只有I2C_SADDR[6:0]有效。 该寄存器只有在I2C接口被禁用(I2C_ENABLE[0]=0)的时候可写。 注意：从机地址不能配置成保留地址：0x00 to 0x07, 或者 0x78 到0x7f. 如果I2C_SADDR或者I2C_TADDR被配置成以上保留地址，那么I2C有可能工作异常。

20.4.5 I2C_DATA_CMD(数据和命令寄存器)

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																					RESTART	STOP	CMD	DATA							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
RESTART	[10]	W	重复起始位(RESTART)控制。 0: 如果I2C_CR.RESTART_EN为1，并且传输的方向发生改变，那么I2C会发送一个重复起始位；如果I2C_CR.RESTART_EN为0，那么I2C会发送停止位并且跟着再重新发送一个起始位。 1: 如果I2C_CR.RESTART_EN为1，不管传输方向是否发生改变，I2C都会在数据发送或者接收前(根据CMD位的值)发送一个重复起始位；如果I2C_CR.RESTART_EN为0，那么I2C会发送停止位并且跟着再重新发送一个起始位。
STOP	[9]	W	停止位(STOP)控制。 0: 不管发送FIFO是否为空，都不发送停止位。如果发送FIFO非空，那么主机根据CMD位的值继续当前的发送或者接收传输。如果发送FIFO为空，那么主机将SCL拉低占用总线，直到发送FIFO中有新的命令。 1: 不管发送FIFO是否为空，都在当前字节传输完成后发送停止位。如果发送FIFO非空，那么主机会立即发送起始位申请总线仲裁来尝试重新开始一个新的传输。
CMD	[8]	W	读写控制。 1: 读 0: 写 在I2C工作在从机模式下，该位不能控制传输的方向。在工作在主机模式时，该位用来控制传输方向。 当一个命令被写入发送FIFO时，该位被用来区分读和写。在从机接收模式下，该位为无效操作位，因为不需要对该位进行操作。在从机发送模式下，0表示I2C_DATA_CMD中的数据需要被发送。
DATA	[7:0]	RW	I2C总线发送或者收到的数据。 在I2C进行读操作的时候，如果写这个寄存器，那么写操作将无效，如果读这个寄存器，那么返回的是I2C接口从总线上收到的数据。

20.4.6 I2C_SS_SCLH(标准模式SCL高电平长度计数寄存器)

Address = Base Address+ 0x014, Reset Value = 0x00000050

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

Name	Bit	Type	Description
SCL_HCNT	[15:0]	RW	该寄存器设置标准速度模式下，SCL时钟高电平的周期长度计数值。 为了保证IO时序的正确性，该寄存器必须在任何I2C总线传输开始之前设置。 该寄存器必须在I2C接口被禁用(I2C_ENABLE[0]=0)时进行配置。 可配置的最小值为6，硬件上禁止写任何比6小的数，如果尝试写比6小的数，结果就是寄存器的值为6。 注意：该寄存器的值不能超过65525，因为该I2C模块使用了一个16位的计数器来检测I2C总线的空闲状态，当这个计数器的值达到SCL_HCNT+10的时候，标记为空闲。

20.4.7 I2C_SS_SCLL(标准模式SCL低电平长度计数寄存器)

Address = Base Address+ 0x018, Reset Value = 0x0000005E

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

Name	Bit	Type	Description
SCL_LCNT	[15:0]	RW	该寄存器设置标准速度模式下，SCL时钟低电平的周期长度计数值。 为了保证IO时序的正确性，该寄存器必须在任何I2C总线传输开始之前设置。 该寄存器必须在I2C接口被禁用(I2C_ENABLE[0]=0)时进行配置。 可配置的最小值为8，硬件上禁止写任何比8小的数，如果尝试写比8小的数，结果就是寄存器的值为8。

20.4.8 I2C_FS_SCLH(高速模式SCL高电平长度计数寄存器)

Address = Base Address+ 0x01C, Reset Value = 0x0000000C

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

Name	Bit	Type	Description
SCL_HCNT	[15:0]	RW	该寄存器设置高速模式下，SCL时钟高电平的周期长度计数值。为了保证IO时序的正确性，该寄存器必须在任何I2C总线传输开始之前设置。 该寄存器必须在I2C接口被禁用(I2C_ENABLE[0]=0)时进行配置。 可配置的最小值为6，硬件上禁止写任何比6小的数，如果尝试写比6小的数，结果就是寄存器的值为6。

20.4.9 I2C_FS_SCLL(高速模式SCL低电平长度计数寄存器)

Address = Base Address+ 0x020, Reset Value = 0x0000001A

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

Name	Bit	Type	Description
SCL_LCNT	[15:0]	RW	该寄存器设置高速或者超高速模式下，SCL时钟低电平的周期长度计数值。为了保证IO时序的正确性，该寄存器必须在任何I2C总线传输开始之前设置。 该寄存器必须在I2C接口被禁用(I2C_ENABLE[0]=0)时进行配置。 可配置的最小值为8，硬件上禁止写任何比8小的数，如果尝试写比8小的数，结果就是寄存器的值为8。

20.4.10 I2C_RX_FLSEL(接收FIFO阈值寄存器)

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																								RX_FLSEL							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
RX_FLSEL	[7:0]	RW	接收FIFO阈值。 FIFO中数据的数量超过或者等于该阈值时，将触发RX_FULL中断(I2C_RISR的第2位)。该阈值不能超过FIFO深度(8)，如果设置的值超过了FIFO深度，那么实际写入的值为FIFO的深度值。 该寄存器的值0表示有1个数据，7表示阈值为8个数据。

20.4.11 I2C_TX_FLSEL(发送FIFO阈值寄存器)

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																								TX_FLSEL							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
TX_FLSEL	[7:0]	RW	发送FIFO阈值。 FIFO中数据的数量少于或者等于该阈值时，将触发TX_EMPTY中断(I2C_RISR的第4位)。该阈值不能超过FIFO深度(8)，如果设置的值超过了FIFO深度，那么实际写入的值为FIFO的深度值。 该寄存器的值0表示0个数据，7表示阈值为8个数据。

20.4.12 I2C_RX_FL(接收FIFO状态寄存器)

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																												RX_FL			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
RX_FL	[3:0]	R	接收FIFO中有效数据的个数。

20.4.13 I2C_TX_FL(发送FIFO状态寄存器)

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																												TX_FL			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
TX_FL	[3:0]	R	发送FIFO中有效数据的个数。

20.4.14 I2C_ENABLE(使能寄存器)

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																												RECOVER_EN	RSVD	ABORT	ENABLE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	RW	RW

Name	Bit	Type	Description
RECOVER_EN	[3]	RW	SDA被拉低锁死后的恢复功能控制 0: 禁用 1: 使能 如果TX_ABRT中断(I2C_ABRT[17])显示SDA被拉低锁死，那么该位可以控制使能SDA的恢复机制(发送最多9个SCL时钟和停止位尝试释放SDA)，然后该位自动清零。
ABORT	[1]	RW	中止I2C传输。 0: 中止操作没有发生或则中止操作已完成 1: 中止操作正在进行 在主机模式下，软件可以将该位置1，用来中止I2C传输。只有在I2C使能状态下(ENABLE=1)，才可以对该位置1，否则任何写操作都无效。发起中止操作后，I2C会在当前传输完成后产生一个停止位并且清空发送FIFO，然后在中止操作后产生TX_ABRT中断。ABORT位会在中止操作后自动清零。
ENABLE	[0]	RW	使能I2C模块 0：禁用 1：使能 当I2C被禁用，会发生下面的事件： - 发送FIFO和接收FIFO被清空 - 状态标志位仍然保持有效状态直到I2C进入空闲状态 如果模块正在发送状态，那么它会在当前传输完成后停止并且删除发送缓冲中的内容。如果模块正在接收，那么I2C会在当前字节接收完成后停止，并且不发送应答位。

20.4.15 I2C_STATUS(状态寄存器)

Address = Base Address+ 0x040, Reset Value = 0x00000006

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																ENABLE				RECOVER_FAIL	RSVD				SLV_BUSY	MST_BUSY	RFF	RFNE	TFE	TFNF	BUSY
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
ENABLE	[15:12]	R	同I2C_ENABLE
RECOVER_FAIL	[11]	R	SDA拉低锁死的恢复状态 0: 恢复没有失败 1: 恢复失败 该位表示SDA拉低锁死后的恢复机制是否成功。
SLV_BUSY	[6]	R	从机模式的状态机工作状态 0: 从机模式状态机处于空闲状态，I2C的从机部分不在工作 1: 从机模式状态机处于工作状态，I2C的从机部分正在工作
MST_BUSY	[5]	R	主机模式的状态机工作状态 0: 主机模式状态机处于空闲状态，I2C的主机部分不在工作 1: 主机模式状态机处于工作状态，I2C的主机部分正在工作 注意：I2C_STATUS[0] – 也就是工作状态位 – 是MST_BUSY和SLV_BUSY位的或逻辑结果。
RFF	[4]	R	接收FIFO已满 0: 接收FIFO未满 1: 接收FIFO已满 当接收FIFO已满时置1，当FIFO含有至少1个或者多个空位时会被清0。
RFNE	[3]	R	接收FIFO非空 0: 接收FIFO为空 1: 接收FIFO非空 当接收FIFO含有至少1个或者多个数据时置1，当FIFO为空时会被清0。
TFE	[2]	R	发送FIFO为空 0: 发送FIFO非空 1: 发送FIFO为空

			当发送FIFO完全为空时置1，当FIFO含有至少1个或者多个数据时会被清0。该位不会产生中断。
TFNF	[1]	R	发送FIFO未满 0: 发送FIFO已满 1: 发送FIFO未满 当发送FIFO含有至少1个或者多个数据时置1，当FIFO为满时会被清0。
BUSY	[0]	R	I2C工作状态 0: 空闲 1: 工作中

20.4.16 I2C_SDA_TSETUP(SDA Setup时间寄存器)

Address = Base Address+ 0x048, Reset Value = 0x00000064

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

Name	Bit	Type	Description
SDA_TSETUP	[7:0]	RW	Setup时间的长度使用下面公式计算： [(SDA_TSETUP - 1) * (PCLK周期)] 如果需要的延时是1000ns，PCLK频率是10MHz，那么建议设置SDA_TSETUP的值为11. SDA_TSETUP的最小值为2.

20.4.17 I2C_SDA_THOLD(SDA Hold 时间寄存器)

Address = Base Address+ 0x04C, 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								SDA_RX_THOLD								SDA_TX_THOLD															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
SDA_RX_THOLD	[23:16]	RW	当I2C作为接收端时，设置SDA的hold时间，单位为PCLK的周期。
SDA_TX_THOLD	[15:0]	RW	当I2C作为发送端时，设置SDA的hold时间，单位为PCLK的周期。 SDA_TX_THOLD的值必须大于最小的hold时间： - 主机模式下1个时钟周期 - 从机模式下7个时钟周期

20.4.18 I2C_SPKLEN(毛刺干扰滤波控制寄存器)

Address = Base Address+ 0x050, 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																								SPKLEN							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
SPKLEN	[7:0]	RW	该寄存器设置毛刺干扰过滤逻辑可以过滤的SCL和SDA上最长的毛刺信号长度，以PCLK周期为单位。 该寄存器必须在I2C总线传输开始之前设置，以保证稳定的传输。 该寄存器只有在I2C接口被禁止(I2C_ENABLE[0]=0)的时候可以设置，否则写操作无效。可以设置的最小值为1，硬件禁止写比1小的值，写0仍然为1。

20.4.19 I2C_MISR(中断状态寄存器)

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
RSVD																	SCL_S_LOW	RSVD	RESTART_DET	GEN_CALL	START_DET	STOP_DET	BUSY	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
SCL_S_LOW	[14]	R	
RESTART_DET	[12]	R	
GEN_CALL	[11]	R	
START_DET	[10]	R	
STOP_DET	[9]	R	
BUSY	[8]	R	
RX_DONE	[7]	R	
TX_ABRT	[6]	R	
RD_REQ	[5]	R	
TX_EMPTY	[4]	R	
TX_OVER	[3]	R	
RX_FULL	[2]	R	
RX_OVER	[1]	R	
RX_UNDER	[0]	R	

中断状态位 (I2C_IMSCR寄存器中使能后才有效)

参考I2C_RISR寄存器中的具体说明。

20.4.20 I2C_IMCR(中断使能寄存器)

Address = Base Address+ 0x05C, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																	SCL_S_LOW	RSVD	RESTART_DET	GEN_CALL	START_DET	STOP_DET	BUSY	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
SCL_S_LOW	[14]	RW	
RESTART_DET	[12]	RW	
GEN_CALL	[11]	RW	
START_DET	[10]	RW	
STOP_DET	[9]	RW	
BUSY	[8]	RW	
RX_DONE	[7]	RW	
TX_ABRT	[6]	RW	
RD_REQ	[5]	RW	
TX_EMPTY	[4]	RW	
TX_OVER	[3]	RW	
RX_FULL	[2]	RW	
RX_OVER	[1]	RW	
RX_UNDER	[0]	RW	

相应的中断使能

0: 禁止

1: 使能

参考I2C_RISR寄存器中的具体说明。

20.4.21 I2C_RISR(原始中断状态寄存器)

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																	SCL_S_LOW	RSVD	RESTART_DET	GEN_CALL	START_DET	STOP_DET	BUSY	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
SCL_S_LOW	[14]	R	SCL锁死状态 0: SCL没有被拉低锁死 1: SCL被拉低锁死 表示SCL是否被拉低锁死，锁死时间超过了I2C_SCL_TOUT个PCLK周期的时长。
RESTART_DET	[12]	R	重复起始位检测状态 (从机模式) 0: 没有检测到重复起始位 1: 检测到重复起始位 当I2C工作在从机模式时，表示I2C接口上是否产生了重复起始位，并且该从机是被寻址到的从机。
GEN_CALL	[11]	R	General call状态 0: 没有收到general call 1: 收到general call 当收到general call地址并且应答了该地址，该位置1。该状态位会一直保持，直到关闭I2C或者对I2C_ICR寄存器中相应位写1来清零该位。I2C将接收到的数据保存到接收缓冲区中。
START_DET	[10]	R	起始位检测状态 0: 没有检测到起始位 1: 检测到起始位 表示I2C总线上是否产生了起始位，不管是主机模式还是从机模式。
STOP_DET	[9]	R	停止位检测状态 0: 没有检测到停止位 1: 检测到停止位

			表示I2C总线上是否产生了停止位，不管是主机模式还是从机模式。 注意在从机模式： - 如果I2C_CR[7]=1'b1 (STOP_DET_CON)，只有从机被成功寻址时，才会产生STOP_DET中断。 在general call寻址模式下，如果STOP_DET_CON=1，即使该从机设备产生了ACK去响应general call地址，该设备也不会产生STOP_DET中断。只有当传输地址跟从机地址(SADDR)匹配的时候，才会产生STOP_DET中断。 - 如果I2C_CR[7]=1'b0 (STOP_DET_CON)，不管是否被寻址，都会产生STOP_DET中断。
BUSY	[8]	R	I2C工作状态 0: I2C空闲 1: I2C工作中 该位是I2C的工作状态，状态会一直保持直到被下面3种方式清除： - 禁用I2C模块 - 将I2C_ICR寄存器的BUSY位写1 - 系统复位 一旦该位被置1后，只有通过上面3中方法清零。即使I2C模块处于空闲状态，该位仍然为1，除非被上述3种方法清零。
RX_DONE	[7]	R	RX_DONE状态 0: RX_DONE中断没有发生 1: RX_DONE中断发生 当I2C作为接收发送端，如果主机不应答发送的字节，那么该位置1。该状态一般发生在传输的最后一个字节，表示传输已经完成。
TX_ABRT	[6]	R	TX_ABRT状态 0: TX_ABRT中断没有发生 1: TX_ABRT中断发生 该位表示I2C作为发送端，不能够完成发送FIFO中要求完成的动作。这个情况在主机和从机模式下都有可能发生，意思是“发送被中止”。 当该位被置1时，I2C_TX_ABRT寄存器会保存发送中止的原因。
RD_REQ	[5]	R	RD_REQ状态 (从机模式) 0: 没有读请求 1: 收到读请求

			当I2C工作在从机模式并且另外一个I2C主机正在尝试从该从机设备读取数据时，该位置1。I2C会占用并且让总线处于等待状态(SCL=0)，直到该中断被响应并处理，意思是该从机被远端主机寻址到并且被要求传输数据。处理器必须响应该中断并且将要求传输的数据写入I2C_DATA_CMD寄存器。
TX_EMPTY	[4]	R	TX_EMPTY状态 0: TX缓冲区非空 (>TX_FLSEL) 1: TX缓冲区为空 (<=TX_FLSEL) TX_EMPTY中断状态的工作行为跟I2C_CR寄存器中TX_EMPTY_CON的选择有关。 - 当TX_EMPTY_CON = 0: 如果发送缓冲等于或者小于I2C_TX_FLSEL寄存器中设置的阈值，该位置1。 - 当TX_EMPTY_CON = 1: 如果发送缓冲等于或者小于I2C_TX_FLSEL寄存器中设置的阈值，并且当前传输的地址或者数据完成，该位置1。 当缓冲区数据个数大于阈值时，该位会自动被清零。当I2C_ENABLE[0]=0时，发送FIFO会被清除并且处于复位状态，所以该位被置1。
TX_OVER	[3]	R	TX_OVER状态 0: TX溢出中断未发生 1: TX溢出中断发生 如果发送缓冲被填满8个数据，而且处理器仍然尝试往I2C_DATA_CMD寄存器里写命令时置1。如果I2C模块被关闭(I2C_ENABLE[0]=0)，那么这位的值会一直保持，直到主机或从机的状态机变为空闲状态。 TX_OVER状态 0: TX溢出中断未发生 1: TX溢出中断发生 如果发送缓冲被填满8个数据，而且处理器仍然尝试往I2C_DATA_CMD寄存器里写命令时置1。如果I2C模块被关闭(I2C_ENABLE[0]=0)，那么这位的值会一直保持，直到主机或从机的状态机变为空闲状态。
RX_FULL	[2]	R	RX_FULL状态 0: RX缓冲区未满 (<RX_FLSEL) 1: RX缓冲区已满 (>=RX_FLSEL)

			当缓冲区数据个数等于或者大于I2C_RX_FLSEL寄存器中RX_FLSEL阈值时置1，当数据个数小于阈值时会自动清零。当该模块被禁止(I2C_ENABLE[0]=0)时，接收FIFO会被清除并且处于复位状态，也就是接收FIFO是未满状态，所以一旦I2C_ENABLE[0]=0，该位就会被清零。
RX_OVER	[1]	R	RX_OVER状态 0：RX溢出中断未发生 1：RX溢出中断发生
RX_UNDER	[0]	R	RX_UNDER状态 0: RX under-flow中断没有发生 1: RX under-flow中断发生 当接收FIFO为空，处理器尝试通过I2C_DATA_CMD寄存器去读取接收数据时置1。如果I2C模块被关闭 (I2C_ENABLE[0]=0)，那么这位的值会一直保持，直到主机或从机的状态机变为空闲状态。 RX_UNDER状态 0: RX under-flow中断没有发生 1: RX under-flow中断发生 当接收FIFO为空，处理器尝试通过I2C_DATA_CMD寄存器去读取接收数据时置1。如果I2C模块被关闭 (I2C_ENABLE[0]=0)，那么这位的值会一直保持，直到主机或从机的状态机变为空闲状态。
相应的中断使能 0: 禁止 1: 使能 参考I2C_RISR寄存器中的具体说明。			

20.4.22 I2C_ICR(中断清除寄存器)

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																	SCL_S_LOW	RSVD	RESTART_DET	GEN_CALL	START_DET	STOP_DET	BUSY	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	R	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	Type	Description
SCL_S_LOW	[14]	W	
RESTART_DET	[12]	W	
GEN_CALL	[11]	W	
START_DET	[10]	W	
STOP_DET	[9]	W	
BUSY	[8]	W	
RX_DONE	[7]	W	
TX_ABRT	[6]	W	
RD_REQ	[5]	W	
TX_EMPTY	[4]	W	
TX_OVER	[3]	W	
RX_FULL	[2]	W	
RX_OVER	[1]	W	
RX_UNDER	[0]	W	

相应的中断清除

0: 无效

1: 清除中断状态

参考I2C_RISR寄存器中的具体说明。

20.4.23 I2C_SCL_TOUT(SCL锁死超时控制寄存器)

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

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

Name	Bit	Type	Description
SCL_TOUT	[31:0]	RW	
如果I2C检测到SCL被锁死在低电平超过SCL_TOUT个PCLK周期时，会产生SCL拉低锁死中断。			

20.4.24 I2C_SDA_TOUT(SDA锁死超时控制寄存器)

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

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

Name	Bit	Type	Description
SDA_TOUT	[31:0]	RW	
如果I2C检测到SDA被锁死在低电平超过SDA_TOUT个PCLK周期，那么I2C可以通过使能RECOVER_EN (I2C_ENABLE[3])寄存器来打开SDA的恢复功能。			

20.4.25 I2C_TX_ABRT(发送中止状态寄存器)

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														SDA_S_LOW	USER_ABRT	SLVRD_INTX	SLV_ARBLOST	SLVFLUSH_TX	ARB_LOST	MASTER_DIS	10B_RD_NRSTRT	SBYTE_NRSTRT	RSVD	SBYTE_ACK	RSVD	GCALL_READ	GCALL_NACK	TXDATA_NACK	10ADDR2_NACK	10ADDR1_NACK	7ADDR_NACK
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
SDA_S_LOW	[17]	R	只在主机模式有效。主机检测到SDA被锁死在低电平超过I2C_SDA_TOUT个PCLK周期。
USER_ABRT	[16]	R	只在主机模式有效。主机检测到发送中止(IC_ENABLE[1]).
SLVRD_INTX	[15]	R	1: 当处理器(从机模式)响应了一个给远程主机发送数据的请求时，用户对I2C_DATA_CMD寄存器的第8位CMD写1。
SLV_ARBLOST	[14]	R	1: 从机在发送数据到远端主机时丢失了总线。I2C_TX_ABRT[12]会被同时置1。
SLVFLUSH_TX	[13]	R	1: 从机收到了一个读命令并且发送FIFO中有数据，所以从机产生了一个TX_ABRT中断去清除发送FIFO中的旧数据。
ARB_LOST	[12]	R	1: 主机丢失了仲裁，或者在I2C_TX_ABRT[14]置1的情况下从机发送端丢失了仲裁
MASTER_DIS	[11]	R	1: 用户在主机模式没有使能的情况下尝试主机操作
10B_RD_NRSTRT	[10]	R	1: 重复起始位被禁止(RESTART_EN (I2C_CR[5]) = 0)并且主机在10位寻址模式发送了一个读命令。
SBYTE_NRSTRT	[9]	R	1: 重复起始位被禁止(RESTART_EN (I2C_CR[5]) = 0)并且用户正在尝试发送一个起始字节。 如果要清除第9位，SBYTE_NRSTRT的触发源必须固定；重复起始位必须使能(I2C_CR[5]=1)，SPECIAL位不能为0x3 (I2C_TADDR[11:10])。一旦触发源固定，该位就可以跟清除其它位的方法一样进行清除，否则该位在清除后又会被置起。
SBYTE_ACK	[7]	R	1: 主机发送了一个起始字节并且这个起始字节被应答了(错误行为)
GCALL_READ	[5]	R	1: I2C在主机模式发送了一个general call但是用户的程序在general call后从总线上读数据 (I2C_DATA_CMD[9]被设置为1)
GCALL_NACK	[4]	R	1: I2C在主机模式发送了一个general call并且总线上没有从机应答该general call
TXDATA_NACK	[3]	R	1: 该位只在主机模式下有效。主机收到了地址的应答，但是当它发送数据到该地址时，却没有收到任何从机的应答。
10ADDR2_NACK	[2]	R	1: 主机工作在10位寻址模式并且没有任何从机应答主机发送的10位

			地址的第二个字节
10ADDR1_NACK	[1]	R	1: 主机工作在10位寻址模式并且没有任何从机应答主机发送的10位地址的第一个字节
7ADDR_NACK	[0]	R	1: 主机工作在7位寻址模式并且没有任何从机应答主机发送的地址

20.4.26 I2C_GCALL(General Call控制寄存器)

Address = Base Address+ 0x078, 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																															ACK_GCALL
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
ACK_GCALL	[0]	RW	
应答General Call 1: I2C使用ACK应答General Call 0: I2C不产生General Call中断			

20.4.27 I2C_NACK(从机NACK控制寄存器)

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																NACK
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW

Name	Bit	Type	Description
NACK	[0]	RW	NACK的产生控制 1: 在数据字节收到后产生NACK 0: 按正常情况产生NACK/ACK 该位只有在I2C工作在从机接收端时有效。如果该位为1，I2C在收到一个数据字节后只会产生一个NACK，此后的数据传输会被中止并且收到的数据不会被存入接收缓冲区内。如果该位为0，I2C按照正常的情况产生NACK/ACK。

20.4.28 I2C_DMACR(DMA控制寄存器)

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																												TX_DMA_SEL	RX_DMA_SEL	TX_DMAEN	RX_DMAEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W

Name	Bit	Type	Description
TX_DMA_SEL	[3]	W	TX DMA功能选择 0: TX_FL小于等于7触发DMA请求 1: TX_FL小于等于TX_FLSEL触发DMA请求
RX_DMA_SEL	[2]	W	RX DMA功能选择 0: RX_FL大于等于1触发DMA请求 1: RX_FL大于等于RX_FLSEL触发DMA请求
TX_DMAEN	[1]	W	TX DMA功能使能 0: 禁止 1: 使能
RX_DMAEN	[0]	W	RX DMA功能使能 0: 禁止 1: 使能

20.4.29 I2C_SRR(软件复位控制寄存器)

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																														SWRST_C	SWRST_R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	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
SWRST_C	[1]	W	控制逻辑软件复位 0 = 无效 1 = 软件复位 注意：该位只复位控制逻辑，不复位寄存器
SWRST_R	[0]	W	寄存器软件复位 0 = 无效 1 = 软件复位 注意：该位复位整个I2C模块，包括寄存器和控制逻辑

20.4.30 I2C_SADDRQUAL(从机地址限定寄存器)

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																					QUALMODE	SLVQUAL									
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
QUALMODE	[10]	RW	从机地址限定模式 0x0：地址屏蔽模式，SLVQUAL用于匹配地址的掩码。 0x1：地址扩展模式，SLVQUAL用于扩展地址的匹配范围。
SLVQUAL	[9:0]	RW	从机地址限定值，值为0时，I2C_SADDR中的地址按原样使用，即仅匹配I2C_SADDR中的设定值，此项优先级高于QUALMODE，值为0时，QUALMODE设置任意值均无效。 如果QUALMODE = 0, 则SLVQUAL中设置为 1 的任何位在I2C_SADDR寄存器中的对应位将被MASK，无论主机呼叫地址的此对应位为1或者0均有效。 如果 QUALMODE = 1，则地址范围与地址 0 匹配。此范围从2C_SADDR定义的值扩展到SLVQUAL定义的地址（当 SADDR [9:0] ≤ 接收地址 ≤ SLVQUAL[9:0] 时地址匹配）

20.4.31 I2C_ADDRMON(地址监控寄存器)

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																						MONADDR									
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
MONADDR	[9:0]	R	地址监控：一旦I2C使能，I2C总线上呼叫的地址将会被存储于MONADDR字段

21 串行输入输出 (SIO)

21.1 概述

本章节描述SIO控制器的功能，从用户的角度详细说明如何操作SIO。

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。

21.1.1 主要特性

串行输入输出(SIO)可模拟多种串行通信协议。

- 单线通信管脚，支持双向数据传输。
- 可通过配置时钟分频得到多种通信速率。
- 接收模式需同步开始标志。
- 接收模式下，可调整每位(bit)的采样数和抽取点。
- 接收模式下，可灵活配置输入滤波。

21.1.2 管脚描述

Table 21-1 SIO管脚描述

管脚名称	功能		I/O类型	有效电平	说明
SIO	SIO输入输出口		数字	-	-

21.1.3 模块框图

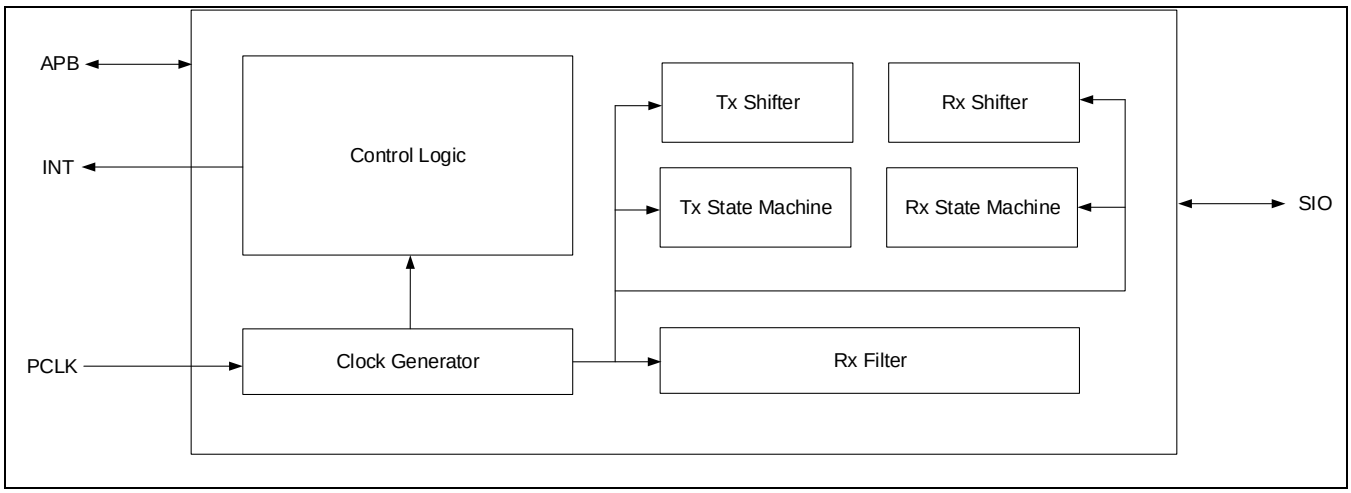


Figure 21-1 SIO模块框图

21.1.4 发送功能说明

21.1.4.1 发送流程框图

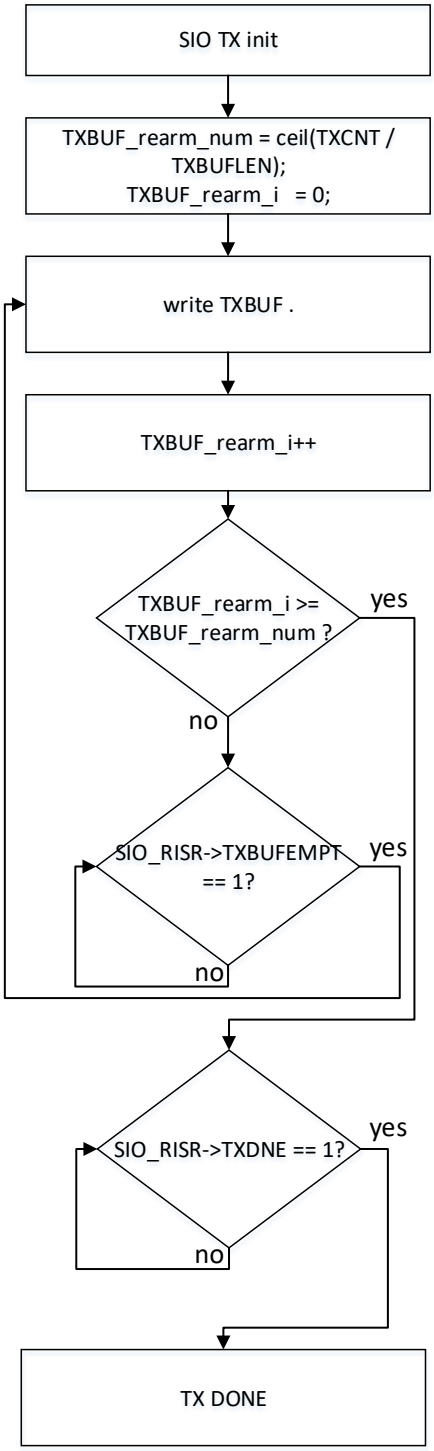


Figure 21-2 SIO 发送流程框图

21.1.4.2 设置发送时钟分频

设置SIO_CR[31:16] (TCKPRS)以设置发送时钟分频系数，发送分频时钟TXCLK的频率：

FTXCLK = FPCLK / (TCKPRS +1)。

21.1.4.3 发送相关设置

SIO定义了D0, D1, DL, DH四个对象，使用这四个对象组合编码，可灵活设置出多种发送波形。

其中D0为全0序列，D1为全1序列，DL、DH序列可通过SIO_TXCR1的LSQ[7:0]和HSQ[7:0]自定义。

D0, D1, DL, DH四个对象的序列长度可分别通过SIO_TXCR1[4:2] (D0DUR)，SIO_TXCR1[7:5] (D1DUR)，SIO_TXCR1[13:11] (LENOBL)和SIO_TXCR1[10:8] (LENOBH)分别独立设置。

发送DL (DH)时，按LSQ[0] (HSQ[0])到LSQ[7] (HSQ[7])的顺序依次移出SIO_TXCR1[13:11] (LENOBL)和SIO_TXCR1[10:8] (LENOBH)位LSQ (HSQ)序列到SIO端口。

当SIO_CR[8] (MODE) 为0时使能发送模块。SIO模块有发送缓存寄存器，写入TXBUF的数据将被载入发送缓存寄存器，SIO模块将按SIO_TXCR0和SIO_TXCR1的设置将发送缓存寄存器移位到SIO端口，并将SIO_RISR[2] (TXBUFEMPT)位被置1，此时可以往TXBUF写下一次要发送数据，

当SIO_TXCR0[2] （TDIR）为0时，SIO_TXBUF数据按LSB到MSB方式移出，即按发送缓存寄存器[1:0]，发送缓存寄存器[3:2] …方式依次译码并移出序列至SIO管脚。当SIO_TXCR0[2] （TDRI）为1时，发送缓存寄存器数据按MSB到LSB方式移出，即按发送缓存寄存器[31:30]，发送缓存寄存器[29:28] …方式依次译码并移出序列至SIO管脚。

为了方便发送非字节/字(Byte/Word)对齐的数据，可设置SIO_TXCR0[7:4] (TXBUFLEN)以配置TXBUF重载个数，即发送移出TXBUFLEN+1个对象后重新载入TXBUF寄存器到发送缓存。设置SIO_TXCR0[15:8] (TXCNT)以配置总发送对象个数。累计发送个数等于SIO_TXCR0[15:8] (TXCNT)+1时，SIO_RISR[0] (TXDNE)将置1，并结束当前发送。

若要发送如Figure 21-3所示发送波形，其中“T”为一个发送时钟TXCLK的长度，则D1，D0对象的长度为1T，DL，DH对象的长度为3T。整个发送的数据长度为9bits。

部分发送相关寄存器设置如下：

```
SIO_TXCR1[4:2] (D0DUR)    = 0b000;          // D0 length = 1 TXCLK
SIO_TXCR1[7:5] (D1DUR)    = 0b000;          // D0 length = 1 TXCLK
SIO_TXCR1[13:11] (LENOBL) = 0b010;          // DL length = 3 TXCK
SIO_TXCR1[31:24] (LSQ) = 0b00000001;      // DL sequence
SIO_TXCR1[10:8] (LENOBH) = 0b010;           // DH length = 3 TXCK
SIO_TXCR1[23:16] (HSQ) = 0b00000011;      // DH sequence
```

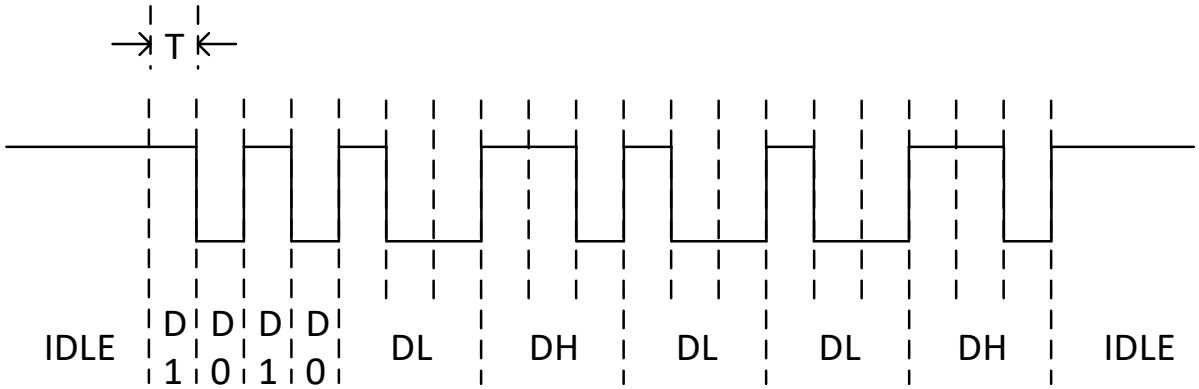


Figure 21-3 SIO 发送示例波形

整个发送的数据长度为9bits，可设置SIO_TXCR0[7:4] (TBUFLEN) = 0x8, SIO_TXCR0[15:8] (TXCNT) = 0x8。

也可以设置SIO_TXCR0[7:4] (TBUFLEN) = 0x4，SIO_TXCR0[15:8] (TXCNT) = 0x8，这样在写TXBUF后查到SIO_RISR[2] (TXBUFEMPT)为1时，需将发送数据的第8位到第5位重新编码后写入TXBUF。

21.1.4.4 发送状态标示

SIO_RISR[0] (TXDNE) 和SIO_RISR[2] (TXBUFEMPT)显示当前发送状态，发送控制器处于空闲状态，未往TXBUF写入数据时, TXBUFEMPT为1；发送控制器处于空闲状态，往TXBUF写入数据时, TXBUFEMPT置0；当发送控制器载入TXBUF到内部发送缓存寄存器后，TXBUFEMPT会被置1。

在发送多个数据时，在发送控制器处于发送状态且TXBUFEMPT位为1时，写TXBUF会清除TXBUFEMPT位为0，等发送缓存寄存器的数据被全部发送后，TXBUF会被载入发送缓存寄存器并将TXBUFEMPT位置1。如此循环发送完所有数据。在TXBUFEMPT位为0时，写入数据到TXBUF会导致发送结果和预期不一致。

在发送多个数据时，在发送控制器处于发送状态且TXBUFEMPT位置为1时，如果未能及时写入TXBUF，发送控制器在发送完发送缓存寄存器后，将重新载入TXBUF。可能会导致发送结果和预期不一致。建议在发送多个数据时禁止全局中断或调整SIO中断为最高优先级，以保证写入TXBUF不会被其他中断打断而不能及时更新TXBUF。

当发送的bit数目等于SIO_TXCR0[15:8] (TXCNT) + 1时,发送控制器退出发送状态,并将SIO_RISR[0] (TXDNE)置1。

21.1.5 接收功能说明

21.1.5.1 接收流程框图

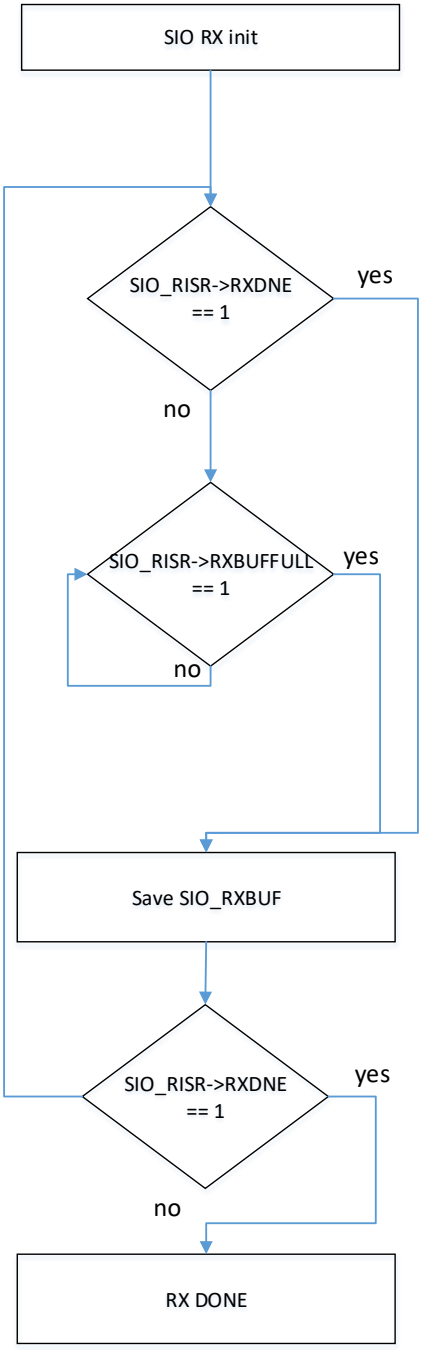


Figure 21-4 SIO 接收流程框图

21.1.5.2 设置接收时钟分频

设置SIO_RXCR1[31:16] (RCKPRS)以设置接收钟分频系数，接收分频时钟RXCLK的频率：

$F_{RXCLK} = F_{PCLK} / (RCKPRS + 1)$。

21.1.5.3 接收采样与抽取

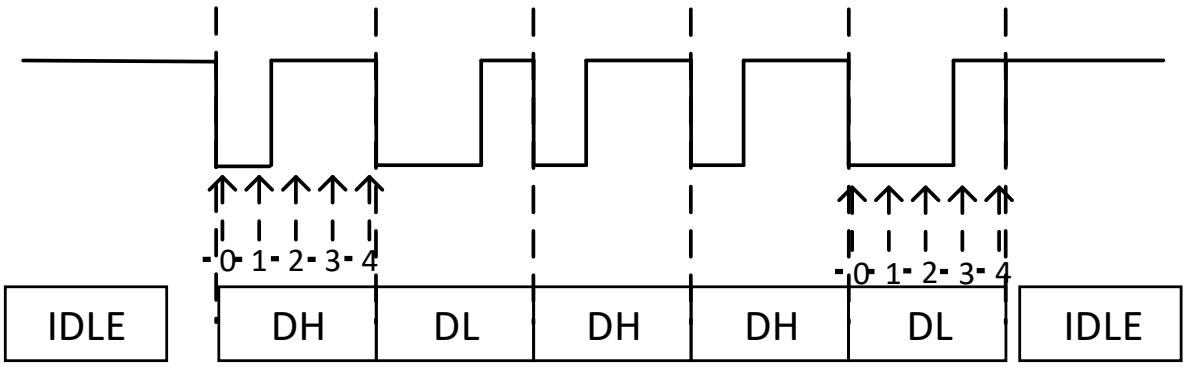


Figure 21-5 SIO 接收采样和抽取示意图

由SIO_RXCR0[1:0] (BSTSEL)决定采样触发边沿，在采样触发边沿启动采样，按接收分频时钟RXCLK的频率对SIO端口采样SIO_RXCR0[8:4] (SPLCNT) + 1次，在第SPLCNT次采样时，根据SIO_RXCR0[15:10] (EXTRACT)的设置抽取采样结果为H或L，并移入接收缓存寄存器。

如Figure 21-5所示的SIO信号，在下降沿触发接收,可设置SIO_RXCR0[8:4] (SPLCNT)为4，设置SIO_RXCR0[15:10] (EXTRACT)为2，如果DH和DL对象的差异较小时，可提高接收分频时钟RXCLK的频率，同时相应地增加SIO_RXCR0[8:4] (SPLCNT)以增加采样次数。

21.1.5.4 接收相关设置

按通信协议设置好采样时钟RXCLK分频，触发边沿，每bit采样数目和抽取条件。

对于通信协议每bit均有触发沿的接收，设置SIO_RXCR0[31:30] (RMODE)位为00b；若通信协议有的bit没有触发沿，则设置SIO_RXCR0[31:30] (RMODE)为01b。

在实际通信中，接收采样周期与主机发送的信号周期有一定误差，特别是通信协议中有的bit没有触发沿，误差将会被累计，可以使能SIO_RXCR0[28] (ALIGNEN)，在采样触发沿将对整采样已减少采样误差。

设置SIO_CR[8] (MODE)为1时，启动SIO接收模块。

SIO模块内置接收缓存寄存器，在每bit采样结束时，如果SIO_RXCR0[29] = 0时，接收的数据从第0位左移入接收缓存寄存器，如果SIO_RXCR0[29] = 1时，接收的数据从第31位右移入接收缓存寄存器.

接收缓存寄存器移入的数据位数等于SIO_RXCR1[12:8] (BUFCNT) + 1时，接收缓存寄存器数据将移入RXBUF，并将SIO_RISR[3] (RXBUFFLL)置1，当累计接收个数等于SIO_RXCR1[7:0] (RXCNT) + 1时，结束当前接收，并将SIO_RISR[1] (RXDNE)置1。

21.1.5.5 退出(BREAK)复位

设置SIO_RXCR2[0] (BREAKEN)为1使能退出(BREAK)复位，在检测到触发边沿后，在SIO_RXCR2[7:3] (BREAKCNT)+1个采样长度内，接收到数据都为SIO_RXCR2[1] (BREAKLVL)则复位接收模块，并产生BREAK中断。

21.1.5.6 超时(TO)复位

设置SIO_RXCR0[24] (TORSTEN)为1使能超时(TO)复位，在检测到触发边沿后，在SIO_RXCR0[23:16] (TOCNT)+1个采样长度内，未再次检测到触发边沿则复位接收模块，并产生TO中断。

21.2 寄存器说明

21.2.1 寄存器表

Base Address of SIO0: 0x400B0000
Base Address of SIO1: 0x400B1000

Register	Offset	Description	Reset Value
SIO_CR	0x0000	控制寄存器	0x00000000
SIO_TXCR0	0x0004	发送控制寄存器0	0x000000F0
SIO_TXCR1	0x0008	发送控制寄存器1	0x00000000
SIO_TXBUF	0x000C	发送数据缓存寄存器	0x00000000
SIO_RXCR0	0x0010	接收控制寄存器0	0x001001F0
SIO_RXCR1	0x0014	接收控制寄存器1	0x00001FFF
SIO_RXCR2	0x0018	接收控制寄存器2	0x00FF00F8
SIO_RXBUF	0x001C	接收数据缓存寄存器	0x00000000
SIO_RISR	0x0020	原始中断状态寄存器	0x00000000
SIO_MISR	0x0024	中断状态寄存器	0x00000000
SIO_IMCR	0x0028	中断使能控制寄存器	0x00000000
SIO_ICR	0x002C	中断清除寄存器	0x00000000
SIO_SRR	0x0030	软件复位寄存器	0x00000000

21.2.2 SIO_CR(控制寄存器)

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
TCKPRS																RST_KEY			WOKE_RST	RXDMAEN	TXDMAEN	RSVD	MODE	DEBCKS				DEBDEP			CLKEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	W	W	W	W	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
TCKPRS	[31:16]	RW	发送时钟分频系数。 发送分频时钟TXCLK的频率：FTXCLK = FPCLK / (TCKPRS +1)
RST_KEY	[15:13]	W	只有此位写0x05时，写WOKE_RST才有效
WOKE_RST	[12]	W	IP工作复位 0：无效 1：除寄存器外所有逻辑复位 注：此位只有同时写[15:13]为0x5才有效
RXDMAEN	[11]	RW	RX DMA使能
TXDMAEN	[10]	RW	TX DMA使能
MODE	[8]	RW	SIO发送与接收控制。 0h：发送模式。 1h：接收模式。
DEBCKS	[7:4]	RW	接收去抖滤波时钟分频系数。 去抖滤波分频时钟DBCLK的频率：FDBCLK = FPCLK / (DEBCKS +1)
DEBDEP	[3:1]	RW	接收去抖滤波检查周期数。 0h：1个周期 1h：2个周期 2h：3个周期 3h：4个周期 4h：5个周期 5h：6个周期 6h：7个周期 7h：8个周期
CLKEN	[0]	RW	SIO时钟使能控制。 0h：工作时钟禁止。 1h：工作时钟使能。

21.2.3 SIO_TXCR0(发送控制寄存器0)

Address = Base Address+ 0x0004, Reset Value = 0x000000F0

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

Name	Bit	Type	Description
TXCNT	[15:8]	RW	发送模式时，发送序列的长度控制。 当发送开始时，发送计数器在完成一个对象的发送后自动增加1。当发送到TXCNT+1个对象完成后，结束当前发送。
TXBUFLEN	[7:4]	RW	发送数据BUFFER长度。 当发送开始时，发送计数器在完成一个对象的发送后自动增加1。当发送到TXBUFLEN +1个对象完成后，若发送BIT数仍小于TXCNT+1则重新载入SIO_TXBUF并发送。
TDIR	[2]	RW	发送数据方向。 0h：将SIO_TXBUF数据按LSB到MSB方式移出。 1h：将SIO_TXBUF数据按MSB到LSB方式移出。
IDLEST	[1:0]	RW	空闲时输出状态选择。 0h：高阻态。 1h：高电平输出。 2h：低电平输出。 3h：高阻态。

21.2.4 SIO_TXCR1(发送控制寄存器1)

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
LSQ								HSQ								RSVD		LENOBL			LENOBH			D1DUR			D0DUR			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	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R

Name	Bit	Type	Description
LSQ	[31:24]	RW	DL对象的序列定义 发送LSQ序列时,按LSQ[0]到LSQ[7]的顺序依次移位发送，发送序列长度由SIO_TXCR1[13:11] (LENOBL)决定。
HSQ	[23:16]	RW	DH对象的序列定义 发送HSQ序列时,按HSQ[0]到HSQ[7]的顺序依次移位发送，发送序列长度由SIO_TXCR1[10:8] (LENOBH)决定。
LENOBL	[13:11]	RW	DL对象序列长度选择 0h：1bit。 1h：2bit。 2h：3bit。 …… 7h：8bit。
LENOBH	[10:8]	RW	DH对象序列长度选择 0h：1bit。 1h：2bit。 2h：3bit。 …… 7h：8bit。
D1DUR	[7:5]	RW	D1状态的时间长度 (D1DUR+1) x Ttxshft
D0DUR	[4:2]	RW	D0状态的时间长度 (D0DUR+1) x Ttxshft

21.2.5 SIO_TXBUF(发送数据缓存寄存器)

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
DATA15		DATA14		DATA13		DATA12		DATA11		DATA10		DATA9		DATA8		DATA7		DATA6		DATA5		DATA4		DATA3		DATA2		DATA1		DATA0	
0	0	0	0	0	0	0	0	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
DATA15~DATA0	[31:0]	RW	

发送数据格式选择。

0h：D0。

1h：D1。

2h：DL。

3h：DH。

若TXCR0的TDIR位为0，将依次发送DATA0，DATA1 … DATAn。

若TXCR0的TDIR位为1，将依次发送DATA15，DATA14 … DATAn。

21.2.6 SIO_RXCR0(接收控制寄存器0)

Address = Base Address+ 0x0010, Reset Value = 0x001001F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RMODE		RDIR	ALIGNEN	RSVD							HITHR					EXTRACT						RSVD	SPLCNT					TRGMODE	RSVD	BSTSEL	
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
RW	RW	RW	RW	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	R	RW	RW

Name	Bit	Type	Description
RMODE	[31:30]	RW	采样模式 0h: 在当前bit采样结束后，检测到SIO端口有沿跳变，才开始采样下一个数据 1h: 在当前bit采样结束后，无论SIO端口是否有沿跳变均立即开始采样下一个数据 其他: 无效模式
RDIR	[29]	RW	接收数据方向。 0h: 接收到的数据，即发送端的数据是从MSB到LSB移出。依次从第0位开始移入RXBUF。 1h: 接收到的数据，即发送端的数据是从LSB到MSB移出。依次从第31位开始移入RXBUF
ALIGNEN	[28]	RW	采样对准使能 0h: 不使能采样对准. 1h: 使能采样对准，在采样触发沿对准采样结果
HITHR	[20:16]	RW	接收原始数据抽取高的判断阈值 当SIO_RXCR0[15:10] （EXTRACT）为20H时，在接收到的原始数据的1的个数大于HITHR时，采样提取为1否则抽取为 0。
EXTRACT	[15:10]	RW	采样提取策略设置。 0h：接收到的原始数据的第0位作为提取值。 1h：接收到的原始数据的第1位作为提取值。 2h：接收到的原始数据的第3位作为提取值。 …… 1Dh：接收到的原始数据的第29位作为提取值。 1Eh：接收到的原始数据的第30位作为提取值。 1Fh：接收到的原始数据的第31位作为提取值。 20h：接收到的原始数据的1的个数大于SIO_RXCR2[20:16] (HITHR)时提取为H，否则提取为L 其他：保留
SPLCNT	[8:4]	RW	采样的长度，即一个BIT数据由SPLCNT+1个采样决定。
TRGMODE	[3]	RW	采样触发模式选择。 0h：选择去抖后采样信号。

			1h：选择30ns滤波后的采样信号。
BSTSEL	[1:0]	RW	采样触发边沿。 0h：上升沿触发。 1h：下降沿触发。 2h：上升、下降沿均触发。 3h：上升沿触发。

21.2.7 SIO_RXCR1(接收控制寄存器1)

Address = Base Address+ 0x0014, Reset Value = 0x00001FFF

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

Name	Bit	Type	Description
RCKPRS	[31:16]	RW	接收时钟分频系数。 接收分频时钟RXCLK的频率：FRXCLK = FPCLK / (RCKPRS +1)
BUFCNT	[12:8]	RW	接收数据BUF的长度，最大支持32个bit。 当接收到的bit数等于BUFCNT+1时，保存接收到的数据到SIO_RXBUF，并产生RXBUFFULL中断。
RXCNT	[7:0]	RW	接收数据的长度，最大支持256个bit。 当接收到的BIT数等于RXCNT+1时，则产生接收完成(RXDNE)中断

21.2.8 SIO_RXCR2(接收控制寄存器2)

Address = Base Address+ 0x0018, Reset Value = 0x00FF00F8

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

Name	Bit	Type	Description
TOCNT	[23:16]	RW	采样超时长度，在采样触发后，TOCNT+1个周期内未检测到采样触发边沿，则复位接收模块。
TORSTEN	[15]	RW	采样超时复位。 0h：禁止采样超时复位。 1h：使能采样超时复位。
BREAKCNT	[7:3]	RW	复位检测周期数。 使能复位检测，在BREAKCNT+1个采样长度内，接收到数据都为BREAKLVL则复位接收模块，并产生BREAK中断。
BREAKLVL	[1]	RW	复位检测电平。 0h：低电平。 1h：高电平
BREAKEN	[0]	RW	复位检测使能控制。 0h：禁止复位检测。 1h：使能复位检测，在BREAKCNT+1个采样长度内，接收到数据都为BREAKLVL则复位接收模块，并产生BREAK中断。

21.2.9 SIO_RXBUF(接收数据缓存寄存器)

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
RXBUF																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
RXBUF	[31:0]	R	

接收数据缓存

当SIO_RXCR0的RDIR位为0时，接收抽取后的bit位依次从第0位移入。

当SIO_RXCR0的RDIR位为1时，接收抽取后的bit位依次从第31位移入。

21.2.10 SIO_RISR(原始中断状态寄存器)

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																										TIMEOUT	BREAK	RXBUFFULL	TXBUFEMPT	RXDNE	TXDNE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
TIMEOUT	[5]	R	TIMEOUT中断请求原始标志状态
BREAK	[4]	R	BREAK中断请求原始标志状态
RXBUFFULL	[3]	R	RXBUFFULL中断请求原始标志状态
TXBUFEMPT	[2]	R	TXBUFEMPT中断请求原始标志状态
RXDNE	[1]	R	RXDNE中断请求原始标志状态
TXDNE	[0]	R	TXDNE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

21.2.11 SIO_MISR(中断状态寄存器)

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																										TIMEOUT	BREAK	RXBUFFULL	TXBUFEMPT	RXDNE	TXDNE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
TIMEOUT	[5]	R	TIMEOUT中断请求标志状态
BREAK	[4]	R	BREAK中断请求标志状态
RXBUFFULL	[3]	R	RXBUFFULL中断请求标志状态
TXBUFEMPT	[2]	R	TXBUFEMPT中断请求标志状态
RXDNE	[1]	R	RXDNE中断请求标志状态
TXDNE	[0]	R	TXDNE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

21.2.12 SIO_IMCR(中断使能控制寄存器)

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																										TIMEOUT	BREAK	RXBUFFULL	TXBUFEMPT	RXDNE	TXDNE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	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
TIMEOUT	[5]	RW	TIMEOUT中断使能控制位
BREAK	[4]	RW	BREAK中断使能控制位
RXBUFFULL	[3]	RW	RXBUFFULL中断使能控制位
TXBUFEMPT	[2]	RW	TXBUFEMPT中断使能控制位
RXDNE	[1]	RW	RXDNE中断使能控制位
TXDNE	[0]	RW	TXDNE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

21.2.13 SIO_ICR(中断清除寄存器)

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																										TIMEOUT	BREAK	RXBUFFULL	TXBUFEMPT	RXDNE	TXDNE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	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
TIMEOUT	[5]	w	清除TIMEOUT中断请求原始标志状态
BREAK	[4]	w	清除BREAK中断请求原始标志状态
RXBUFFULL	[3]	w	清除RXBUFFULL中断请求原始标志状态
TXBUFEMPT	[2]	w	清除TXBUFEMPT中断请求原始标志状态
RXDNE	[1]	w	清除RXDNE中断请求原始标志状态
TXDNE	[0]	w	清除TXDNE中断请求原始标志状态

中断清除控制位。

对该寄存器写‘0’时，无效；对该寄存器写‘1’时，清除相应中断标志位

读取时，总是返回‘0’

21.2.14 SIO_SRR(软件复位寄存器)

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
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

22 控制器局域网络 (CAN)

22.1 概述

控制器局域网络（CAN，Controller Area Network）是一种具有很高安全性能的串行通信协议，适用于对实时性要求比较高的分布式系统，如汽车电子应用中的引擎控制单元、传感器、雨刮、电子车窗等。本章节描述CAN的协议以及从用户的角度详细说明如何操作CAN。

22.1.1 主要特性

CAN模块由CAN核、报文RAM、报文处理器（报文过滤、CAN核和报文RAM的通信、发送请求处理及中断生成）、控制寄存器组成。主要特性如下：

- 支持CAN 2.0A和2.0B协议。
- 比特率最大支持1MBit/s
- 32个报文对象，每个都有独立的识别符掩码以实现报文过滤。
- 可编程FIFO模式以实现报文对象的级联
- 自动重发机制，支持事件触发的应用场合
- Loop-back模式，支持测试使用
- 功耗控制模块，支持唤醒
- CAN_TX管脚可以设置为开漏，以实现无外部收发器的连接
- 各种中断事件

22.1.2 管脚描述

Table 22-1 CAN管脚描述 Pin Description

管脚名称	信号方向	功能	有效电平
CAN_TX	数据输出	数据输出	Low
CAN_RX	数据输入	数据输入	Low

22.2 功能描述

22.2.1 模块框图

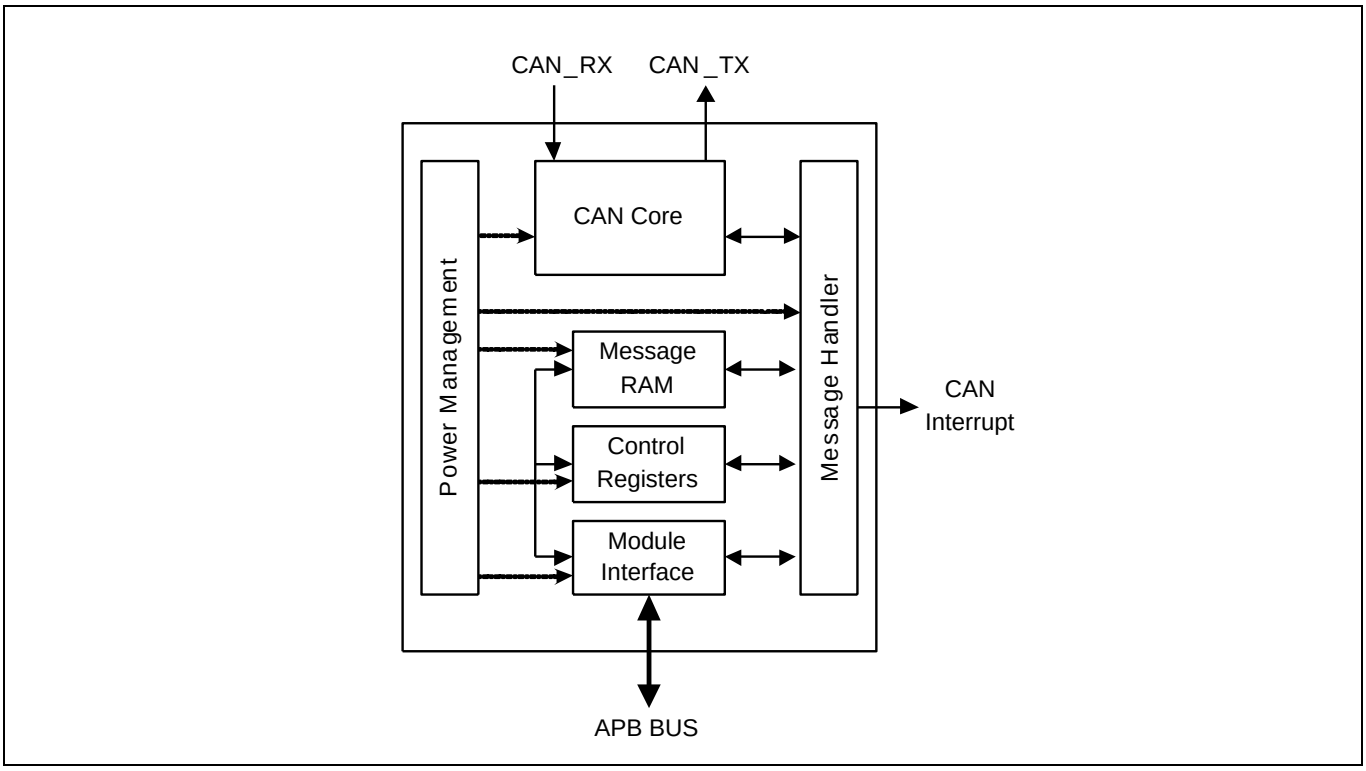


Figure 22-1 CAN 模块框图

22.2.2 基本功能描述

22.2.2.1 CAN协议介绍

CAN协议的目的是为了使任何两个CAN的实现主体间可以实现通信。尽管如此，这并不意味着只要遵守CAN协议就能实现兼容，因为兼容性涉及到很多不同的方面，如电器特性、数据解析等。

22.2.3 基本概念

22.2.3.1 CAN 的属性

CAN具有下列属性:

- 报文可以有不同的优先级
- 时延保证
- 配置灵活
- 有时间同步的多主机接收
- 系统内数据一致性
- 多主
- 错误检测和提醒
- 一旦总线空闲，自动重发已损坏的报文
- 可以识别错误的发生是暂时性的或由节点故障导致。一旦发现故障节点，自动让节点脱离网络

22.2.3.2 CAN节点的分层结构

CAN根据ISO/OSI参考模型，将通信分为两层实现

- 数据链路层

– 逻辑链路子层（LLC），负责报文过滤、过载提醒和恢复管理

– 媒体访问控制子层（MAC）。是CAN协议的核心所在，处理来自LLC子层以及将要发送到LLC子层的报文。MAC子层负责报文成帧、仲裁、确认（ACK）、错误检测和报告。该层由故障界定（Fault Confinement）单元来管理。故障界定可以区分暂时的干扰和永久性的错误。

- 物理层。物理层定义了位时序、位编码和同步方式。一个网络中所有节点的物理层必须是一样的。但协议并没有限制物理层的实现。在保证一致性的前提下，仍然可以根据应用有不同的实现方式。

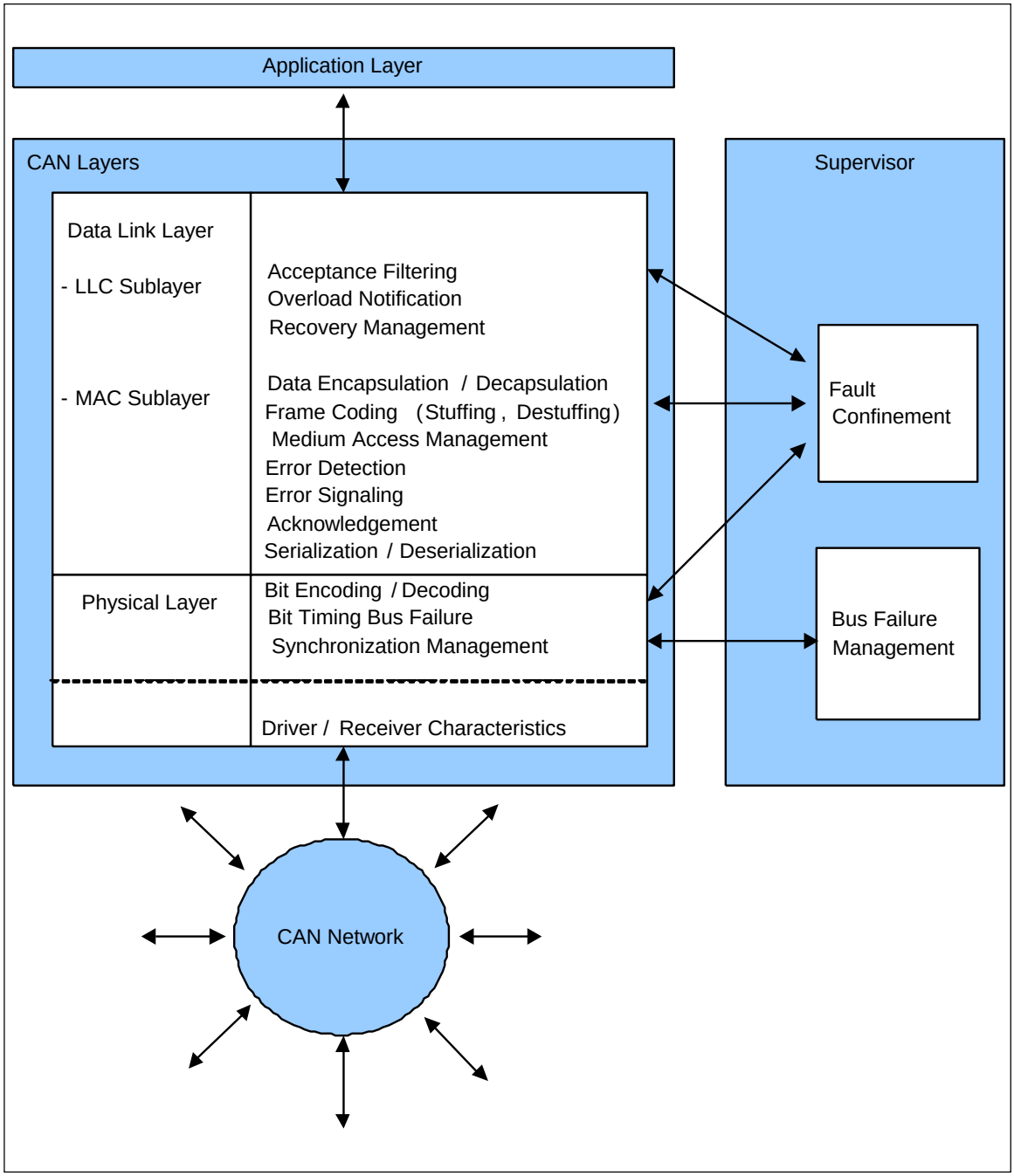


Figure 22-2 CAN 分层描述

- LLC = Logical Link Control
- MAC = Medium Access Control

22.2.3.3 报文

在总线上传输的报文有固定的格式，报文长度可以不同，但有限制（详见 报文传输）。任何网络内的单元都可以在总线空闲的时候开始一次报文发送。

22.2.3.4 信息路由

在一个CAN系统中，任何一个CAN节点都不需要知道系统的配置信息（如节点地址），好处有：

- 系统具有灵活性。如果要在网络中增加节点，软件和硬件层面都不需要做任何更改。
- 天然的报文路由。报文的内容由一个标识符来标定。这个标识符并不指定报文的目标地址。网络中所有的节点都可以根据自己的过滤条件来决定是否要对其进行处理。
- 多路广播。因为节点具有报文过滤机制，而不是限定地址，任何数量的节点都可以同时收到相同的报文并做出回应。
- 数据一致性。在一个CAN网络中，报文必然同时被所有节点收到，所以藉由多路广播和错误处理机制，系统中的数据将始终保持一致。

22.2.3.5 比特率

在不同的系统中，CAN的传输数率不尽相同。但在同一个系统中，所有节点必须具有相同的比特率。

22.2.3.6 远程数据请求

一个节点可以通过发送一个远程请求帧来要求另一个节点发送相应的数据帧。数据帧和相应的远程请求帧由相同的标识来标记，以实现配对。

22.2.3.7 多主机和仲裁

任何一个节点可以在总线空闲时开始一次数据发送。当多个设备同时发送时，具有最高优先级的报文单元获得总线使用权。优先级由标识符决定，以位与的方式进行。在整个仲裁过程中，所有的发送器都会对自己发送出去的位和总线上监测到的位进行比较。如果相同，发送器将继续发送下一位并比较。如果不相同，发送器将退出竞争，停止发送。如果具有相同标识的数据帧和远程帧同时发起数据传输，数据帧将获得总线使用权。

这种方式使得冲突发生时，最高优先级的报文仍然可以完成发送，而不造成任何时间损失。

22.2.3.8 错误检测

为了最大程度地实现数据的安全传输，每个CAN节点都具备错误检测机制。包括：

- 监测(对自己发送出去的位和总线上监测到的位进行比较)。通过监测，可以侦探到所有的网络错误和发送器的本地错误。
- 冗余校验 (CRC)。通过CRC，可以检测到一个报文内部最多5处随机分布的错误以及报文中错误位个数为奇数的错误。当报文长度小于15时，CRC还可以检测到突发错误。

- 位填充
- 报文帧格式检查

综合考虑，总体错误失检率将 < 报文错误率× 4.7×10^{-11}。

22.2.3.9 错误报告及恢复时间

任何一个节点一旦检测到错误都会给错误报文贴上标签，随即停止错误报文的发送，并开始自动重发。如果没有再次遇到错误，从检测到错误到重发之间的延时不会超过31比特的时间。

22.2.3.10 错误限制

CAN节点具有区分暂时性干扰和永久错误的能力，并且可以自动关闭错误的节点。

22.2.3.11 连接

CAN串行通信是在一条总线上进行的，很多单元可以接入总线参与通信。接入的单元数量理论上没有上限。实际应用时，可能会受到时延和负载能力的限制。

22.2.3.12 单信道

CAN总线包含一个双向的单信道，可以通过这个信号获得数据（再次）同步信息。但是协议没有定义物理层，所以就没有对这个信道如何实现进行规范。比如，可以是单芯线（加上接地），或者一对差分线，或者光纤等。

22.2.3.13 总线上的逻辑值

CAN 总线上总是呈现两种互补的逻辑值：显性和隐性。当显性位和隐性位同时传输时，总线上最终呈现的是显性位。比如，在执行“线与”时，逻辑“0”将代表显性电平，逻辑“1”代表隐性电平。规范没有限制标识这些逻辑电平的物理状态（比如电压，光）。

22.2.3.14 应答

所有接收器都会对接收到的小心进行连贯性检查。如果检查结果为连贯，则接收器发送应答，如果检查结果为不连贯，接收器会做出标志。

22.2.3.15 睡眠模式/唤醒

为了减少系统功耗，可以将CAN设备设为睡眠模式以停止其所有内部活动，并断开与总线的连接。CAN设备可以被总线唤醒，也可以在一些预设的系统条件满足时被唤醒。唤醒时，内部逻辑重新开始。传输层（MAC）要等待系统时钟的稳定外，还要花费一定的时间通过检查11个连续的隐性位，将自己和总线活动同步起来。随后，CAN设备才算是处于“总线在线(on-bus)”状态。

22.2.4 报文传输

22.2.4.1 发送器和接收器的定义

一个发起报文传输的节点被称为发送器，并维持发送器的角色直到这个节点仲裁失败或是总线开始空闲。而当总线非空闲时，任何一个不是发送器的节点都被称为报文的接收器。

22.2.4.2 帧格式

依据识别符长度的不同，报文有两种帧格式：

Table 22-2 Identifier Length within Standard and Extended Frame

Frame Format	Identifier Length
标准帧 Standard frame	11 bit
扩展帧 Extended frame	29 bit

22.2.4.3 帧类型

报文传输由以下四种不同的帧类型所表示和控制：

- 数据帧：携带从发送器至接收器的数据
- 远程帧：总线上的一个单元发送数据传输的请求，请求发送具有统一识别符的数据帧
- 错误帧：任何一个节点监测到总线错误时都会发送错误帧
- 过载帧：用来在先行和后续的数据帧或远程帧之间提供一段额外的延时

数据帧和远程帧都可以有标准帧格式或是扩展帧格式，帧与帧之间通过帧间距和先行帧分开。

22.2.4.3.1 数据帧

数据帧由下面7个部分组成：

- 帧头 Start of Frame (SOF)
- 仲裁区 Arbitration field
- 控制区 Control field
- 数据区 Data field
- CRC区 CRC field
- ACK区 ACK field
- 帧尾 End of Frame (EOF)

数据区的长度可以为0。

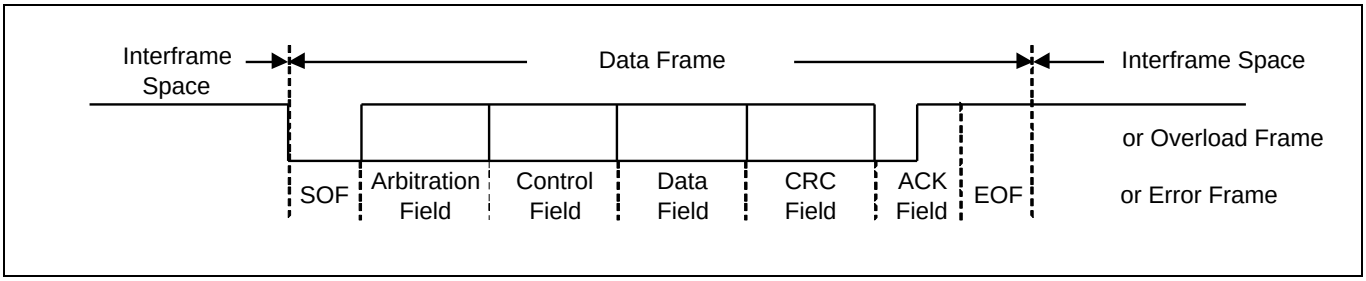


Figure 22-3 Data Frame

22.2.4.3.2 帧头（SOF，标准和扩展帧格式）

帧头用来标记数据和远程帧的开始，仅由一个显性位组成。

一个节点只有在总线处于空闲的时候才可以发起传输(参考 帧间距)。所有挂在总线上的节点都必须依据首先开始发送信息的传输器发送帧头导致的沿变化做出同步调整（参考 硬同步）。

22.2.4.3.3 仲裁区

标准帧和扩展帧有不一样的仲裁区格式。

标准帧的仲裁区由11位识别符和RTR位组成。识别符由ID[28:18]来表示。

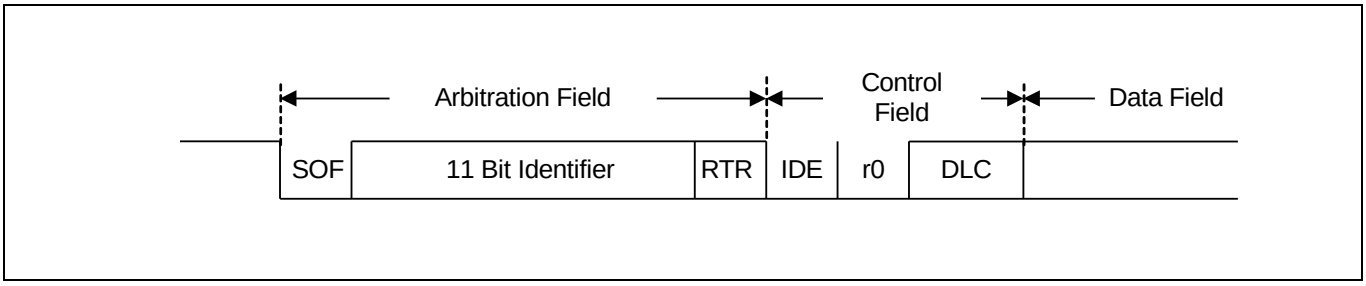


Figure 22-4 Standard Format

扩展帧的仲裁区由29位识别符、SSR位，IDE位和RTR位组成。识别符由ID[28:0]来表示。

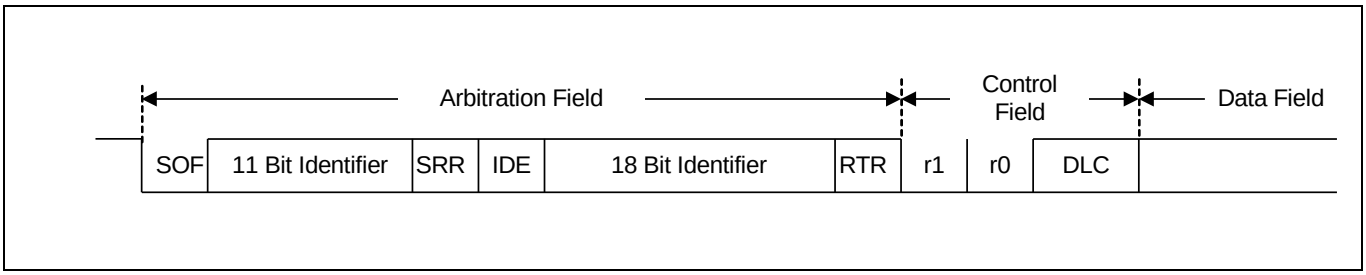


Figure 22-5 Extended Format

为了区分标准帧格式和扩展帧格式，之前版本CAN规范（1.0~1.2）中没有使用的r1位现在表示为IDE位。

- 识别符 Identifier
  - 标准帧格式下的识别符是11位的，对应于扩展帧格式下的基本ID。这些位的发送顺序是从ID[28]（MSB）到ID[18]（LSB）。其中最高的7位（ID[28:22]）必须不能全是隐性。
  - 扩展帧格式下的识别符是29位的。这29位由11位基本ID和18位扩展 ID组成。
    - 基本ID：11位，定义了扩展帧的基本优先级。发送顺序是从ID[28]（MSB）到ID[18]（LSB）。
    - 扩展ID：18位。发送顺序也是高位先发。

在标准帧中，紧随识别符的是RTR位。

– RTR位

Table 22-3 RTR Bit

Frame Type	RTR Bit Logical Value
数据帧 Data frame	显性 ‘dominant’
远程帧 Remote frame	隐性 ‘recessive’

而在扩展帧中，基本ID最先发送，然后是IDE和SRR位，然后才是扩展ID。

- SRR 位

SRR（Substitute Remote Request）位是一个隐性位。在扩展帧中的位置和标准帧中RTR的位置是一样的。因此，如果有一个标准帧的识别符和另一个扩展帧的基本ID相同，当它们发生冲突时，仲裁的结果就是标准帧胜出。

- IDE 位 (扩展帧)

下表说明了IDE位的逻辑值和帧格式类型之间的关系。

Table 22-4 IDE Bit

Frame Format Type	IDE Bit Belongs to	IDE Bit Logical Value
标准帧格式 Standard frame format	控制区 the control field	显性 ‘dominant’
扩展帧格式Extended frame format	仲裁区 the arbitration field	隐性 ‘recessive’

22.2.4.3.4 控制区 (标准和扩展帧格式)

控制区由6位组成。标准帧和扩展帧的控制区是不相同的。在标准帧中，控制区包含数据长度代码，IDE位（必须是显性的），以及一个保留位r0。在扩展帧中，控制区包括数据长度代码和两个保留位r1和r0。保留位必须显性发送，但是发送器不会把接收到隐性位认为是一种错误。

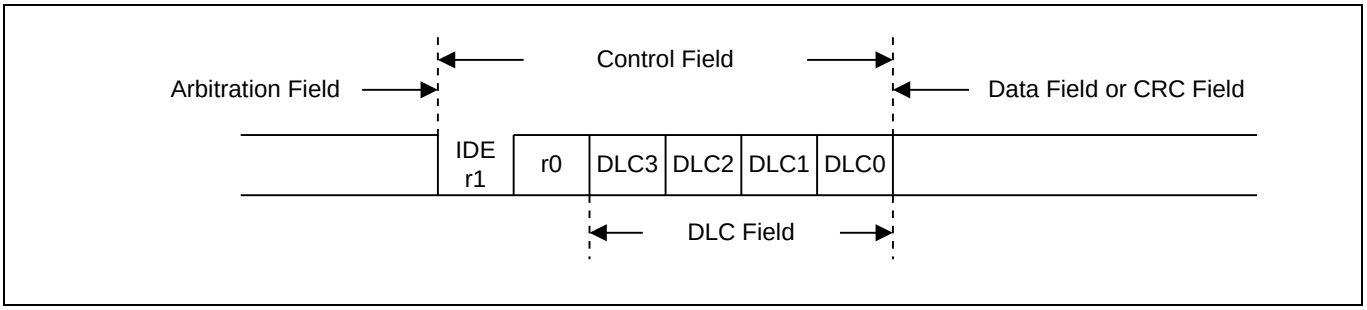


Figure 22-6 CAN Control Field

- 数据长度代码 (标准和扩展帧格式)

数据长度代码（DLC）指示了数据区中的字节数，由4位组成，包含在控制区内。DLC和数据长度的关系见下图。

Table 22-5 Data Length Code (Note)

Number of Data (Bytes)	Data Length Code			
	DLC 3	DLC 2	DLC 1	DLC 0
0	D	D	D	D
1	D	D	D	R
2	D	D	R	D
3	D	D	R	R
4	D	R	D	D
5	D	R	D	R
6	D	R	R	D
7	D	R	R	R
8	R	D	D	D

NOTE: 上图中，D 表示显性位，R 表示隐性位。

数据区的合法长度可以是 0 ~ 8。其他的数值不允许使用。

22.2.4.3.5 数据区 (标准和扩展帧格式)

数据区是数据帧中需要发送的数据。可以是0~8个字节，每个字节都是高位先发。

22.2.4.3.6 CRC区 (标准和扩展帧格式)

CRC区包含了一个CRC序列（CRC SEQUENCE）以及CRC界定符（CRC DELIMITER）。

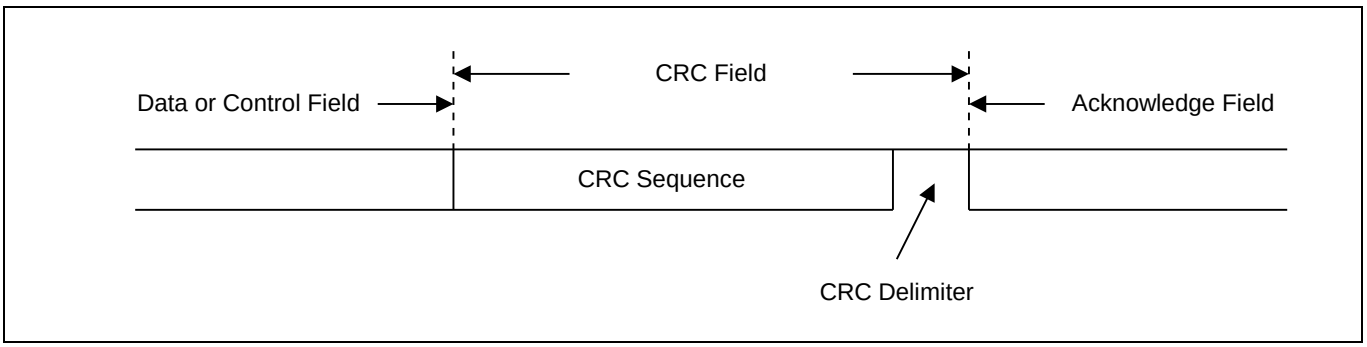


Figure 22-7 CRC Field

- CRC 序列 (标准和扩展格式)

由循环冗余码求得的帧检查序列最适用于位数低于127位（BCH码）的帧。为进行CRC计算，被除的多项式系数由无填充位流给定，组成这些位流的有：帧头，仲裁区，控制区，数据区（如果有的话），而15个最低位的系数都为“0”。将此多项式除以下面的生成多项式（其系数以2为模）：

$X^{15} + X^{14} + X^{10} + X^{8} + X^{7} + X^{4} + X^{3} + 1$

余数即为总线上传送的CRC序列。

- CRC 界定符 (标准和扩展格式)

CRC界定符是跟在CRC序列之后的由一个隐性位组成的部分。

22.2.4.3.7 ACK 区(标准和扩展格式)

应答（ACK）区长度为2位，包含应答间隙（ACK SLOT）和应答界定符（ACK DELIMITER）。在应答区，发送器发送两个隐性位。当接收器正确地接收到有效的报文时，会在应答间隙期间向发送器发送一个显性位以应答。

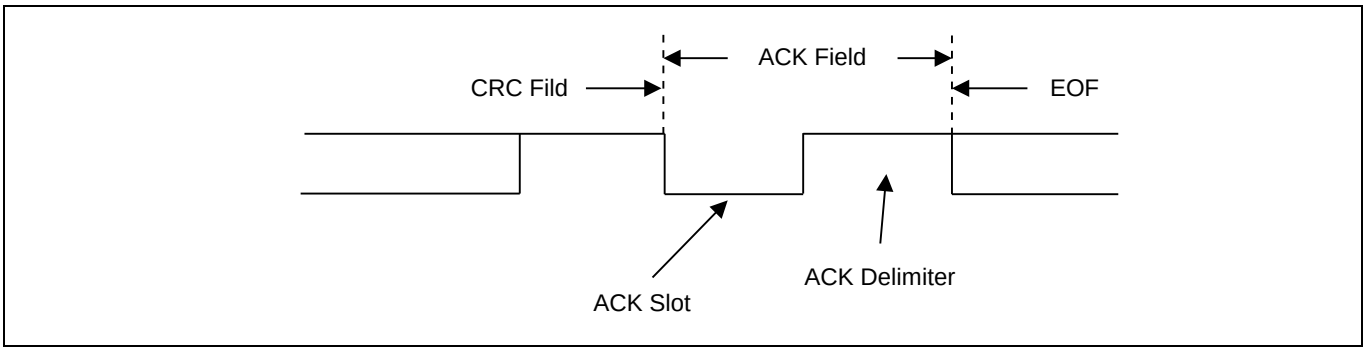


Figure 22-8 ACK Field

- ACK间隙（ACK SLOT）：

所有接收到匹配CRC序列的接收器都会在应答间隙期间发送一个显性位（发送器报文ACK间隙为隐性位）来应答。

- ACK界定符（ACK DELIMITER）：

ACK界定符是ACK区的第二位，必须为隐性位。因此，ACK间隙实际上是被两个隐性位所包围 (CRC界定符和ACK界定符)。

22.2.4.3.8 帧尾 (标准和扩展格式)

每一个数据帧和远程帧由7个隐性位来界定结束。这些位就是帧尾（EOF）。

22.2.4.3.9 远程帧

一个接收特定数据的接收器节点可以发起传输，通过发送远程帧向其资源节点发送数据请求。标准格式和扩展格式均具有远程帧。在两种格式中，远程帧都有6个不同位区组成：

帧头 (SOF)

- 仲裁区 Arbitration field
- 控制区 Control field
- CRC区 CRC field
- ACK区 ACK field
- End of Frame

和数据帧不同，远程帧的RTR位是隐性的，并且无论CLC的内容，远程帧都不具有数据区。CLC可以包含任何0~8之间的内容，表示的是其要求的数据帧中的数据长度。

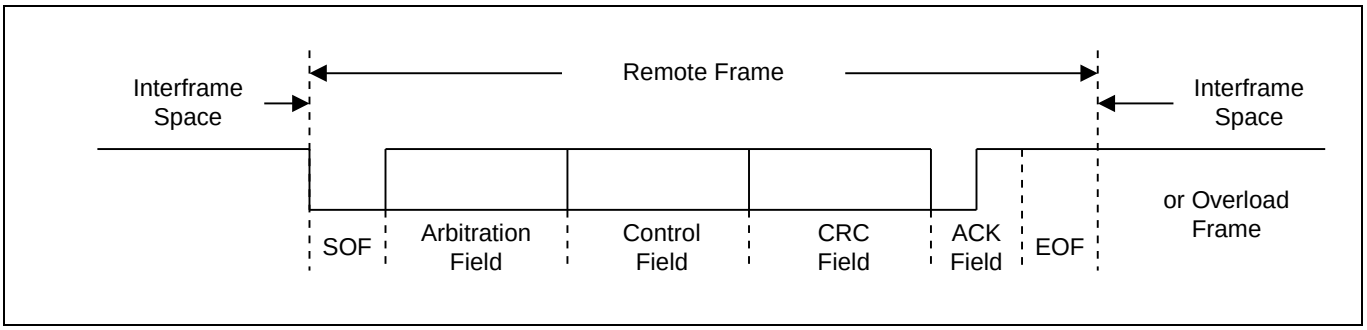


Figure 22-9 Remote Frame

RTR位的极性可以表明当前发送的帧是数据帧（RTR 显性）或者一个远程帧（RTR隐性）。

22.2.4.3.10 错误帧

错误帧包含两个位区。第一个位区是所有节点错误标志的叠加。第二个位区是错误界定符。为了能正确的判断错误帧的结束，一个“错误被动”的节点（不允许发送主动错误的节点）要求总线必须呈现至少3bit ？？时长的空闲（如果一个“错误被动”节点本地有错误的话）。因此，总线负荷不应为100%。

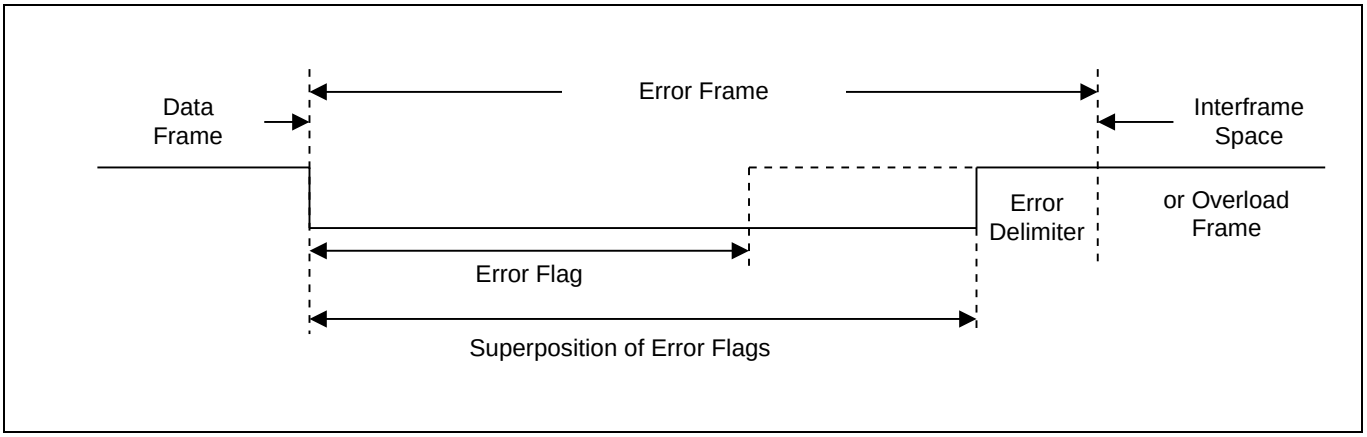


Figure 22-10 Error Frame

22.2.4.3.11 错误标志

有两种形式的错误标志：主动错误标志和被动错误标志。

- 主动错误标志包括6个连续的显性位
- 被动错误由6个连续的隐性位组成。隐性位可能被来自其他节点的显性位覆盖。

一个检测到错误条件的“错误主动”节点通过发送主动错误标志来指示错误。错误标志的形式会破坏从帧头到CRC界定符的位填充规则（参考 编码），或者破坏ACK区和EOF区固定的格式。随后，所有接收到错误条件的节

点都会发出主动错误标志。所有主动错误的叠加，使得总线上将可以看到一个由显性位组成的序列。该序列的长度最小为6位，最大为12位。

一个检测到错误条件的“错误被动”节点，会尝试通过发送被动错误标志来通知总线。但“错误被动”节点的发送其实是通过等待来实现的。一个“错误被动”节点会从主动错误标志起始开始等待6个具有相同极性的连续位。一旦检测到，即认为被动错误标志发送完毕。

22.2.4.3.12 错误帧界定符

错误帧界定符包含8个隐性位。错误标志传送之后，每个节点都会发送一个隐性位，同时监控总线直到检测出一个隐性位（没有被其他显性位覆盖）。随后发送剩下的7个隐性位。

22.2.4.3.13 过载帧

过载帧由过载标志和过载界定符组成。有两种过载条件都会导致过载标志的发送：

- 接收器的内部条件，此接收器要求在下一个数据帧或远程帧之前有一个延时。
- 在间歇场期（INTERMISSION）第一和第二位检测到1个显性位。
- 如果一个CAN节点在一个错误帧或过载帧的第八位（最后一位）检测到了一个显性位，它将开始发送过载帧（不是错误帧）。错误计数器会自加一。

由过载条件1引发的过载帧在那个期待的间歇场的第一位时间开始发送。而由过载条件2和3引发的过载帧，在检测到显性位后开始第一个比特的发送。

为了让下一个数据帧或远程帧延时一段时间后再发送，最多可以产生两个过载帧。

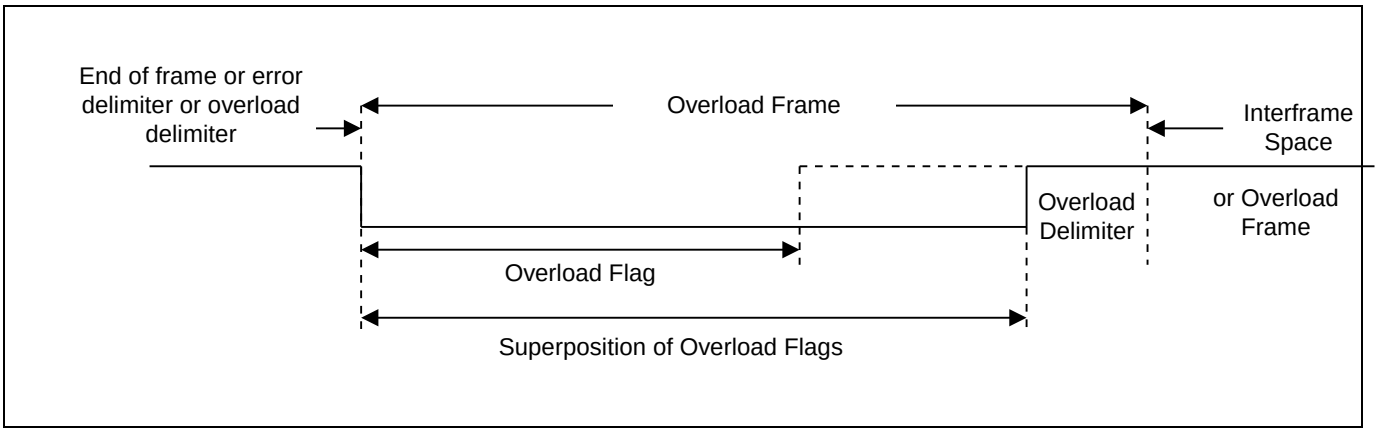


Figure 22-11 Overload Frame

22.2.4.3.14 过载标志

由6个显性位组成。过载标志和主动错误标志的形式一样。

过载标志的形式破坏了间歇场的固定形式。因而，所有其他节点都会在其自身端检测到过载条件，从而也开始发送过载标志。如果节点在间歇场的第三位期间检测到了显性位，这个显性位将被认为是帧头。

22.2.4.3.15 过载界定符

过载界定符由8个隐性位组成。过载界定符和错误界定符的形式一样。完成过载标志以及第一位过载界定符的传输后，节点会监控总线直到检测到显性位到隐性位的变化。这意味着所有的节点都已经完成了过载标志的发送，所有的节点都将同时开始发送剩下的7个隐性位。

22.2.4.3.16 帧间距

无论数据帧还是远程帧，帧与帧之间都有一个帧间距。与此不同的是，过载帧和错误帧之间或是几个过载帧之间都没有帧间距。

帧间距包括间隙区、总线空闲。对“错误被动”节点来说，如果它是上一报文的发送器的话，还要增加挂起传输（SUSPEND TRANSMISSION）位。

所以，对于不是“错误被动”的节点或是上一报文的接收器的话，其帧间距如下图所示：

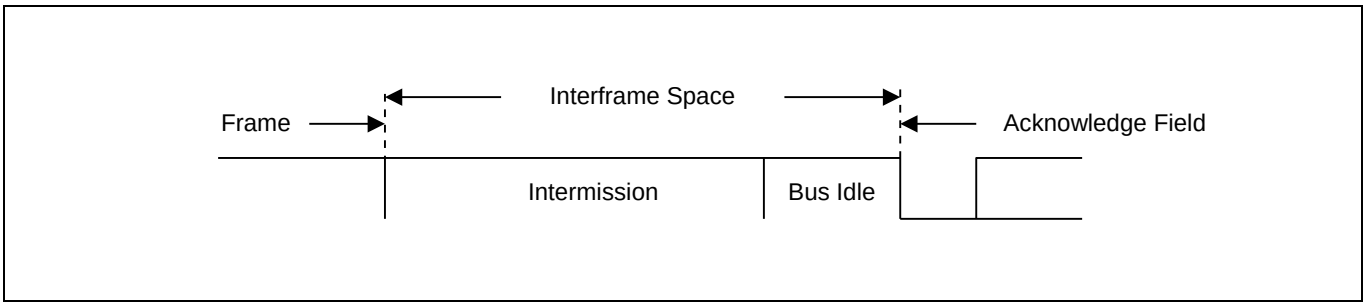


Figure 22-12 Interframe Space for Receiver

对“错误被动”节点来说，如果它是上一报文的发送器的话，其帧间距如下图所示：

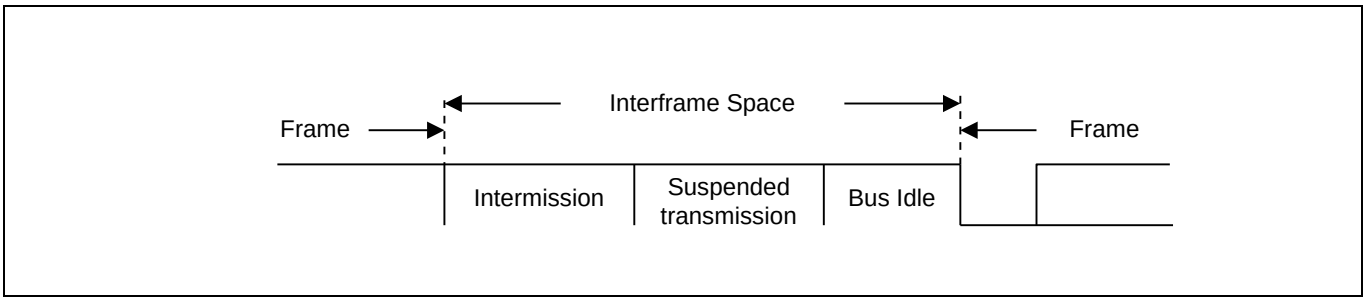


Figure 22-13 Interframe Space for Transmitter

Table 22-6 帧间距的组成位段

位段	组成	说明
间隙（intermission）	3个隐性位	间隙期间，所有的节点除了发起过载信号之外，不能做其他任何操作，包括数据帧或远程帧的发送。 **NOTE:** 如果一个 CAN 节点在等待发送一个报文期间检测到间隙的第三位是一个显性位。它将会把这个显性位视作帧头，但它并不会因此将自己的角色转换为接收器，而是会从下一个位开始发送自己报文中识别符的第一位（不再重复发送帧头）。
总线空闲（Bus Idle）	任意长度	总线空闲时，任何等待发送的节点都可以访问总线。一个因为其他报文的发送而被挂起的报文，将会在下一个间隙之后的第一位开始发送。 一旦空闲的总线上检测到一个显性位，会被视作帧头（SOF）。
挂起传输（Suspended transmission）	8个隐性位	一个“错误被动”节点在发送完一个报文后，会在间歇后发送挂起传输，随后再发送其他的报文或去判断总线是否空闲。如果另一个节点在同一时刻也发起了一个传输，那这个“错误被动”节点就转而成为接收器。

22.2.5 报文过滤

报文过滤基于整个识别符，可以用来选择将具有特定识别符的报文放入接收缓存。掩码寄存器可以将识别符的任何对应位设为“不过滤”。一旦使用掩码寄存器，需要同时对所有位进行设置，即是否需要开启掩码。掩码寄存器的有效位宽可以等于或小于整个识别符。

22.2.6 报文校验

发送器和接收器检验报文是否有效的时间点是不一样的。

- **发送器**

对发送器而言，如果直到帧尾都没有错误的话，就会认为报文发送成功。如果报文破损，会自动根据优先级发起自动重发。为了能够与其他报文争夺总线使用权，重发必须在检测到总线空闲时立即发送。

- **接收器**

对接收器而言，如果直到帧尾的倒数第二位接收都没有错误的话，就会认为报文接收成功。帧尾的最后一位会被认为是无需关注位，即便呈现显性位，也不会认为是错误。

22.2.7 编码

比特流编码：帧的部分，包括帧头、仲裁区、控制区、数据区、CRC，均通过位比特填充的方法编码。无论何时只要发送器检测到发送比特流中有5个连续的相同极性的位，就会自动加入一个相反极性的位。

数据帧或远程帧CRC界定符、ACK区、EOF中的剩余比特区的格式都是固定的，不采用填充策略。错误帧和过载帧也是这样，不会使用填充编码。

报文中比特流按照NRZ（不归零）的方式编码。这意味着在整个比特流中，位电平要么为隐性，要么为显性。

22.2.8 错误处理

22.2.8.1 错误检测

总线上可能有五种不同类型的错误，这些错误间不会相互排斥（可能同时发生）：

- 比特错误：

一个单元在发送每一位的同时也在对总线进行监视。当一个发送的位值和监视到实际的位值不同的时候，即认为发生了比特错误。有一个例外，就是仲裁区或 ACK 间隙因为填充编码而存在的隐性位。此时，即使监视到总线上实际是显性位，也不认为发生了比特错误。发送器会发送一个被动错误标志。

- 填充错误：

如果使用填充编码的区域检测到连续 6 个相同极性位，那么在第 6 位极性判断后就会认为发生了填充错误。

- CRC 错误：

CRC 序列包含了发送器端 CRC 计算的结果。接收器以和发送器相同的方式对接收到的报文进行 CRC 计算。一旦本地计算到的结果和接收到报文中的 CRC 序列不一致时，即认为发生了 CRC 错误。

- 形式错误：

当检测到一个本该有固定形式的区域出现了一个或多个不负荷固定形式的位时，即认为发生了形式错误。（注意：对接收器而言，帧尾的最后一位即便呈现显性位，也不会认为是错误。）

- 应答错误：

一旦一个发送器没有在 ACK 间隙检测到显性位，即认为发生了应答错误。

22.2.8.2 错误标定

一个检测到错误条件的节点会通过发送错误标志的形式去标定错误。对“错误主动”节点，这会是一个主动错误标志，对“被动错误”节点，这会是一个被动错误标志。任何节点检测到比特错误、填充错误、形式错误或应答错误的时候，会在紧随的一个比特期间发送错误标志。而如果检测到CRC错误，除非其他错误条件的错误标志已经置起，否则错误标志会等到ACK界定符后的一个比特器件发送错误标志。

22.2.9 故障界定

如果考虑故障界定，一个单元可能处于下列某一个状态：

- “错误主动”
- “错误被动”
- “总线关闭”

一个“错误主动”节点可以正常地参与总线通信，并在检测到错误时发送主动错误标志。

一个“错误被动”节点是不可以发送主动错误标志的。它虽然也参与总线通信，但是在检测到错误时只能发送被动错误标志。在一次发送结束后，一个“错误主动”节点会在下次传输前等待一段时间（见挂起传输）。

一个“总线关闭”节点不可以对总线产生任何影响。（比如，关闭输出）

为界定故障，每个总线单元都集成了两种计数：发送错误计数和接受错误计数。

这两个计数值在满足下列规则时改变（注意，在同一个报文传输中，可能会用到超过一个规则）

- 当一个接收器检测到一个错误，接收错误计数会增加1。但如果这个检测到的比特错误发生于主动错误标志和过载标志中，则计数值不改变。
- 当一个接收器在发送完一个错误标志后，即刻检测到一个显性位，接受错误计数加8。
- 当一个发送器发送一个错误标志时，发送错误计数加8。

但当发生下述例外情况时，发送错误计数值保持不变：

– 例外情况1：如果一个错误被动发送器因为没有检测到一个显性ACK并且在发送隐性错误标志时没有检测到显性位，而出现应答错误时

– 例外情况2: 如果一个发送器因为在仲裁期间检测到本该填充隐性位的地方检测到了显性位，并因为这样的填充错误而发送错误标志时。

- 如果一个发送器在发送主动错误标志或过载错误标志时检测到发生了比特错误，发送错误计数加8
- 如果一个接收器在发送主动错误标志或过载错误标志时检测到发生了比特错误，接收错误计数加8
- 任何一个节点在发送主动错误标志、被动错误标志或过载标志后都允许最多7个连续的显性位。一旦检测到第14个连续显性位（主动错误标志或过载标志的情况），或者在被动错误标志后收到连续8个显性位时，每检测到一个连续8位显性位序列，所有发送器和接收器的错误计数都加8。
- 当成功发送一个报文之后（收到ACK且在EOF结束后也没有错误发生），发送错误计数减一，除非已经为0。
- 当成功接收一个报文之后（知道ACK间隙都没有错误发生且成功发送应答位），如果当前接收错误计数值在1到127之间，则减一；如果已经为零，则保持零；如果超过127，则会直接设为介于119和127之间的一个值。

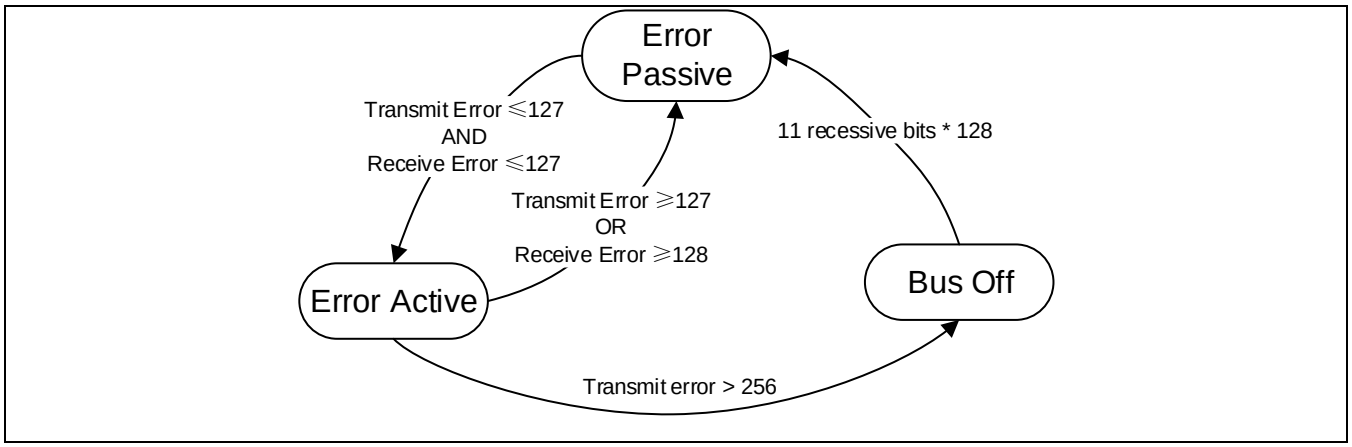


Figure 22-14 状态转变和错误计数值

- 当一个节点的发送错误计数值或接收错误计数值等于或超过128时，这个节点就成为一个“错误被动”节点。该节点在完成状态转换前发送一次主动错误标志。
- 当一个节点的发送错误超过或等于256时，这个节点就“总线关闭”。
- 一个“错误被动”节点的发送错误计数值和接收错误计数值都小于或等于127时，这个“错误被动”节点重新成为“错误主动”节点。
- 当监测到总线上128次出现11个连续的隐性位时，一个“总线关闭”的节点可以变成“错误主动”节点，它的发送和接收错误计数值也被设为0。

NOTE:

a. 如果一个错误计数值超过 96，就代表这个总线已经被严重干扰。此时最好能采取措施去测试这个条件。

b. 启动/唤醒：如果系统启动时只有一个在线节点，且这个节点发送一些报文，将不会受到任何应答。因此会检测到错误，而重新发送。这种情况时，这个节点可以成为“错误被动”节点，而不需要“总线关闭”。

22.2.10 振荡器精度要求

振荡器最大的偏移不能超过1.58%。因此一般来说，一个速度不超过125Kb/s的总线可以使用陶瓷振荡器。更高精度的评估方法，请参
考: Dais, S; Chapman, M "Impact of Bit Representation on Transport Capacity and Clock Accuracy in Serial Data Streams", SAE Technical Paper Series 890532, Multiplexing in Automobile SP-773, March 1989。

对应CAN协议能支持的完全总线，则需要一个晶体振荡器。CAN网络中对振荡频率要求最高的芯片决定了所有其他节点的振荡器精度。

22.2.11 位定时要求

22.2.11.1 标称比特率

标称比特率是一个理想发送器在没有重同步的情况下一秒钟发送的比特数。

22.2.11.2 标称比特时间

标称比特时间在整个网络中是一致的。

标称比特时间 = 1 / 标称比特率

可以把标准比特时间划分成几个不重叠的时间片段。这些片段包括：

- 同步段 (SYNC_SEG)
- 传播段 (PROP_SEG)
- 相位缓冲段1 (PHASE_SEG1)
- 相位缓冲段2 (PHASE_SEG2)

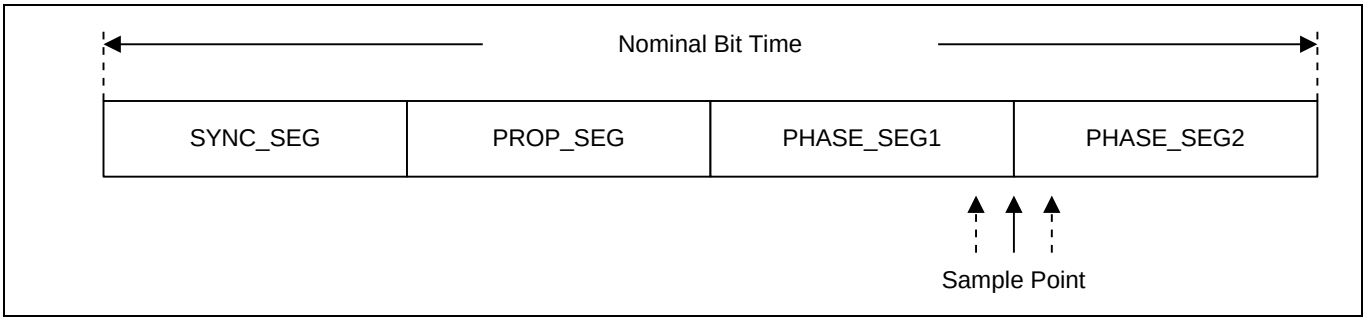


Figure 22-15 Partition of Bit Time

有一个模式寄存器可以设置每个片段的持续时间，以时间份额（TCAN）的整数倍来表示。

22.2.11.3 时间份额

时间份额（T_Q，TCAN）是基于振荡周期的一个固定的最小时间单位。可以通过一个波特率预分频器（CAN_MR[BD9:0]）来编程。T_Q和CPU时钟，以及CAN_MR[BD9:0]的关系如下：

- T_Q = (BD[9:0] +1) / CORECLK

每个时间片段（SYNC_SEG, PROP_SEG, PHASE_SEG1 and PHASE_SEG2）都是T_Q的整数倍。

Table 22-7 比特时间的组成描述

位段	组成	说明
同步段	1 T_Q	这个片段中需要有一个跳变沿，用来同步总线上不同的节点。
传播段	1~8 T_Q	用来补偿网络中的物理时延，需要将它设置为信号在总线上传播时间的两倍，包括总线驱动器和输入比较器的延迟时间。可以通过 CAN_MR[PROP]来设置。
相位缓冲段1	1~8 T_Q	补偿边沿相位误差，重同步的时候 PS1

相位缓冲段2	相位缓冲段1和信息处理时间两个中的大值	补偿边沿相位误差
采样点		总线电平被采样的时刻，紧随相位缓冲段 1。
信息处理时间	≤2 T_Q	以采样点为起始，用于判断后续位的电平值。

综上，整个时间段的长度可以设置为8~25。

NOTE: 通常，通信双方会使用相同的振荡器，因此一个 CAN 设备的振荡频率通常由总线上的控制单元决定。比特时间需要可以调整以获得想要的比特率。但如果本地没有 CPU，比特时间将不能调整，而只是一个固定的值。另一方面，这些设备可以通过选择不同频率的外部振荡器，从而让设备具有合适的比特率，这样振荡频率可编程就不是必须的了。但是，采样点的位置，所有节点必须是一样的。这样，没有 CPU 的 CAN 设备的比特时序应该符合下图定义的比特时间。

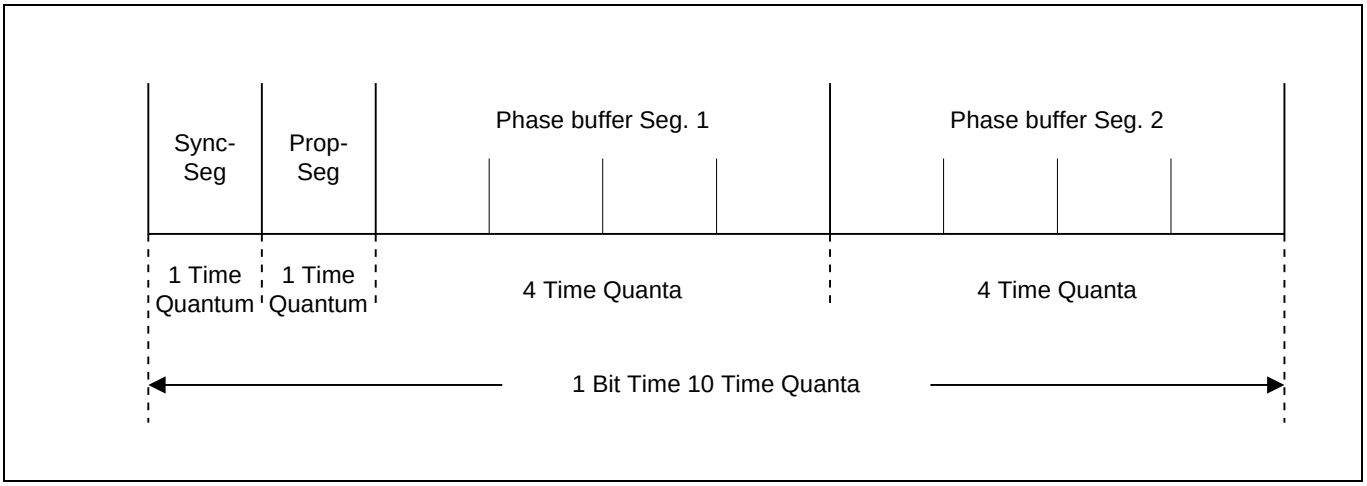


Figure 22-16 Example of Nominal Bit Time

22.2.12 硬同步

硬同步发生后，内部的比特时间从同步段重新开始。因此，硬同步会强迫引起硬同步的跳变沿成为重新开始的比特时间的同步段的起始时刻。

22.2.12.1 重同步跳转宽度

重新同步的结果可能是相位缓冲段1的增长或是相位缓冲段2的缩短。这个增长或是缩短的时长是受重同步跳转宽度限制的。重同步跳转宽度可以设为1和最小值之间（此最小值为4，相位缓冲段1）。时钟信息可以从位和位之间的跳变中得到。具有相同值的连续位个数有一个固定的最大值，这个特性使得一个总线节点可以在一个帧传输的过程中重新同步到比特流。两次传输之间可以用来做重同步的最大长度是29个比特时间。

22.2.12.2 一个跳变的相位误差

一个跳变的相位误差由同步段相关的跳变的位置给出，以时间额度为量度。定义如下：

- E = 0，如果跳变发生在同步段内，没有相位误差
- E > 0，如果跳变发生在采样点前，接收器会认为这是一个慢速发送器发送的滞后边沿（我超前了）。
- E < 0，如果跳变发生在上一个比特的采样点之后，这次的同步段前。接收器会认为这是一个快速发送器发出的下一个比特周期的提前边沿。

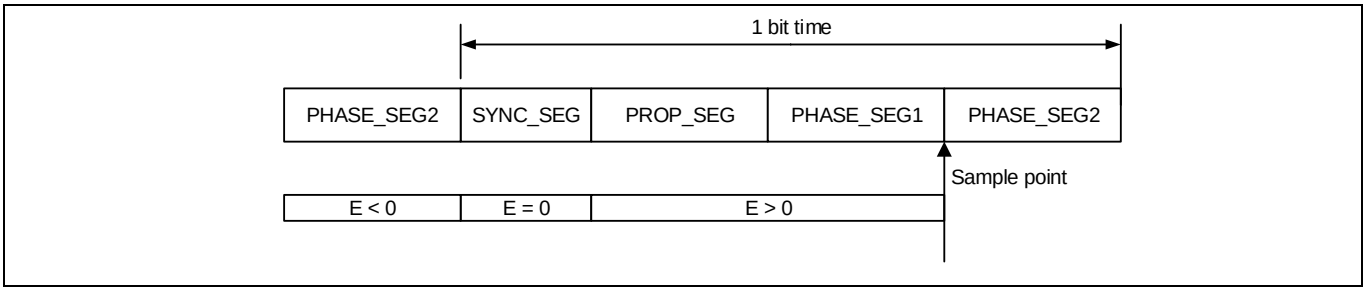


Figure 22-17 phase error

22.2.12.3 重新同步

在报文帧的随后位中，每当有从隐性位到显性位的跳变，并且该跳变落在了同步段之外，就会引起一次重同步。

当引起重新同的跳变的相位误差幅度小于或等于预先设置的重同步跳转宽度时，重新同步和硬同步一样。当相位误差大于重同步跳转宽度时：

- 如果相位误差为正，同步相位缓冲段1自加一个同步跳转宽度值。
- 如果相位误差为负，同步相位缓冲段2自减一个同步跳转宽度值。

22.2.12.4 同步的规则

硬同步和重新同步是实现同步的两种方式。它们都遵循下列规则：

1. 在一个比特时间内只允许发生一次同步。
2. 当且仅当上一个采样点（上一个总线位值）采到的值和跳变沿之后的值不一样时，这个跳变沿才会被用作同步。
3. 任何时候，只要在总线空闲时检测到一个隐性位到显性位的跳变沿，就会执行硬同步。
4. 所有其他情况时，发生满足规则1和2的隐性位到显性位的跳变，就会执行重新同步。有一个例外是，一个发送显性位的节点不会因为有正相位误差的隐性位到显性位的跳变而执行重新同步。

22.3 工作模式

22.3.1 软件初始化

软件初始化以关闭CAN控制器开始，可以是通过置位CAN_CR[CANDIS]软件关闭，也可以是硬件复位，或者是成为总线脱离节点。

当CAN被禁用时，CAN总线上所有的收发都会停止，CAN_TX输出呈隐性值（高电平），CAN_ERCR（CAN错误计数寄存器）计数器保持原有计数值不变。同时，禁止CAN不会改变原有的配置。

初始化CAN控制器时，CPU需要先配置模式寄存器和每个报文对象。每个要用到的报文对象都要初始化，因为其对应的RAM区在上电的时候并不会被初始化。如果不需要某个报文对象，只要把它对应的CAN_IRx[MSGVAL]设为无效即可。

当CAN处于禁止状态（CAN_SR[CANENS] = 0），并且CAN_SR[CCENS] = 1时，仍然可以通过模式寄存器对比特定时进行配置。

只有通过CPU软件使能（CAN_CR[CANEN]=1）会进行软件初始化。初始化之后，比特流处理器（BSP）会等待CAN总线上出现连续的11个隐性位（总线空闲），以将自己同步到CAN总线上的数据传输，之后才会参与总线上的活动或开始报文传输。

报文对象的初始化与CAN_SR[CANENS]的值无关，随时可以进行，但是在BSP开始报文传输前，所有的报文对象必须都赋以特定的识别符或者设为无效（CAN_IRX[MSGVAL] = 0）。如果在正常的操作时要改变某个报文对象的设置，CPU需要先清除CAN_IRx[MAGVAL]，而在设置完成后重新置位，恢复其有效性。

22.3.2 CAN报文传输

一旦CAN被使能且完成初始化后，CAN核会首先将自己同步于CAN总线，然后才参与报文传输。

接收到的报文，一旦通过报文过滤处理，就会被存储于其对应的报文对象内。整个报文包括所有的仲裁位、数据长度代码DLC以及8个字节的数据。

如果使用了识别符掩码，报文对象中那些被屏蔽的仲裁位可能会被重写。

CPU可以通过接口寄存器随时读取每一个报文，并行存取时数据的一致性由报文处理器保证。

CPU更新将要被发送的报文内容。如果一个报文存在永久性报文对象（arbitration and control bits set up during configuration？？？），只有数据区会被更新，TXRQST位和NEWDAT位置位后即开始发送（参考CAN_MCR）。如果几个要发送的报文都属于同一个报文对象（当报文对象的个数不够用时），整个报文对象需要在这个报文发送请求前重新支配？？？

Messages to be transmitted are updated by the CPU. If a permanent Message Object (arbitration and control bits set up during configuration) exists for the message, only the data bytes are updated and then TXRQST bit with NEWDAT bit are set to start the transmission (See CAN_MCR (CAN Interface X Message Control Register)). If the whole Message Object has to be configured before the transmission of this message is requested.

同一时刻，可以同时有任意数量的报文对象的发送请求。它们会根据内在优先级依次发送。任何时刻都可以更新报文或将报文设为无效，即使它们的发送请求正被挂起。当更新发生于一个报文被挂起的发送开始前，之前的数据会被丢弃。

根据报文对象的设置，在接收到具有匹配识别符的远程帧后，报文可以自动发出发送请求。

Depending on the configuration of the Message Object, the transmission of a message may be requested automatically by the reception of a remote frame with a matching identifier.

Guide: 重要提示：发送时，不可以使用第32个报文对象。

22.3.3 报文接口寄存器组

CPU和报文RAM区的交互受两组接口寄存器的控制。接口寄存器避免了CPU读取报文RAM、CAN报文缓存将要发送的数据以及CAN报文的接收之间的冲突。一个完整的报文对象（见31.3.4报文对象）或报文对象的一部分数据在一次发送时可能在报文RAM和IFx报文缓存寄存器之间转移。

这两组接口寄存器的功能（除了基本模式Basic Mode）是完全相同的。一种可能的使用方法是：一组用于将数据转移到报文RAM，另一组用于将数据从报文RAM取出。任何一个操作都可以被其他一个操作中断。表22-8 概述了两组接口寄存器的内容。

每一组接口寄存器都包括了报文缓存寄存器，报文缓存寄存器受它自己的发送管理寄存器控制。发送管理寄存器规定了数据的传输方向、报文对象中的哪个部分会被传输，并用来选择与报文RAM交互的目标报文对象或源报文对象，开始指定的动作。

Table 22-9 Interface Register Sets

Offset	Interface Registers	Name	Access
Interface 0 Registers			
0x0100	Interface 0 Transfer Management Register	CAN_TMR0	Read/Write
0x0104	Interface 0 Data A Register	CAN_DAR0	Read/Write
0x0108	Interface 0 Data B Register	CAN_DBR0	Read/Write

0x010C	Interface 0 Mask Register	CAN_MSKR0	Read/Write
0x0110	Interface 0 Identifier Register	CAN_IR0	Read/Write
0x0114	Interface 0 Message Control Register	CAN_MCR0	Read/Write
0x0118	Interface 0 Stamp Register	CAN_STPR0	Read only
Interface 1 Registers			
0x120	Interface 1 Transfer Management Register	CAN_TMR1	Read/Write
0x124	Interface 1 Data A Register	CAN_DAR1	Read/Write
0x128	Interface 1 Data B Register	CAN_DBR1	Read/Write
0x12C	Interface 1 Mask Register	CAN_MSKR1	Read/Write
0x130	Interface 1 Identifier Register	CAN_IR1	Read/Write
0x134	Interface 1 Message Control Register	CAN_MCR1	Read/Write
0x138	Interface 1 Stamp Register	CAN_STPR1	Read only

22.3.4 报文对象

报文RAM区一共有32个报文对象。它们不能被CPU直接访问，必须通过IFx接口寄存器来交互数据。下表描述了一个报文对象的内容。

Table 22-10 Message Object Content

Name	Description
DAR	First data bytes message
DBR	Last data bytes message
MSKR	Identifier mask
IR	Message identifier
MCR	Message control
STPR	Stamp date of message

22.3.5 禁止自动重发

根据CAN规范 (see ISO11898, 6.3.3 Recovery Management), CAN可以对那些在仲裁时失去控制权或传输过程中被错误干扰的帧自动进行重发。此时，帧发送服务在传输成功前不会通知用户。默认情况，自动重新传输是使能的。但是在一个时间触发的CAN(TTCAN, see ISO11898-1) 环境下，这个机制是可以被禁止的（CAN_MR[AR] = 0）。在这个模式下，代码需要考虑报文缓存控制寄存器的TXRQST和NEWDAT位：

- 当接收报文缓存的发送起始位（TXRQST）= 0，且NEWDAT 仍然为 1时
- 当传输成功完成，NEWDAT = 0 时
- 当传输失败（因为仲裁失败或错误），NEWDAT = 1时。CPU需要重新将TXRQST设为1以开始重新发送。

22.3.6 测试模式

CAN模块由三种可以组合的测试模式：

- 静默模式 （The Silent mode）
- 回环模式（The Loop back mode）
- 基础模式（The Basic mode）

与此同时，代码可以控制 CAN_TX 管脚的状态。CAN_TX可以设置为开漏结构，允许多个CAN控制器直接连接，而无需外部收发器（external transceiver）。

测试寄存器控制CAN_TX管脚的状态和各种测试模式。任何测试寄存器的写操作都受到一个访问钥匙的控制。如果要使能或禁止某个测试功能，都需要先设置TESTKEY = 0x0C75，然后写入控制位。

RX位监控着CAN_RX的状态，所以是只读的。

22.3.6.1 静默模式

可以通过置位SILENT位将CAN核设置为静默模式。

静默模式下的CAN节点可以接收任何有效的数据帧和远程帧，但是只允许往总线上发隐性位，也不可以发送任何报文。如果CAN核被要求发送显性位（如应答位、过载标志、主动错误标志），这个显性位将在内部重新连接，这样CAN核仍然可以监视到这个显性位，但是CAN总线仍然不受影响，呈现隐性位。静默模式可以用来在不干扰总线通信的情况下分析CAN总线上的通信量。下图描述了处于静默模式下的CAN核CAN_TX和CAN_RX的信号连接情况。

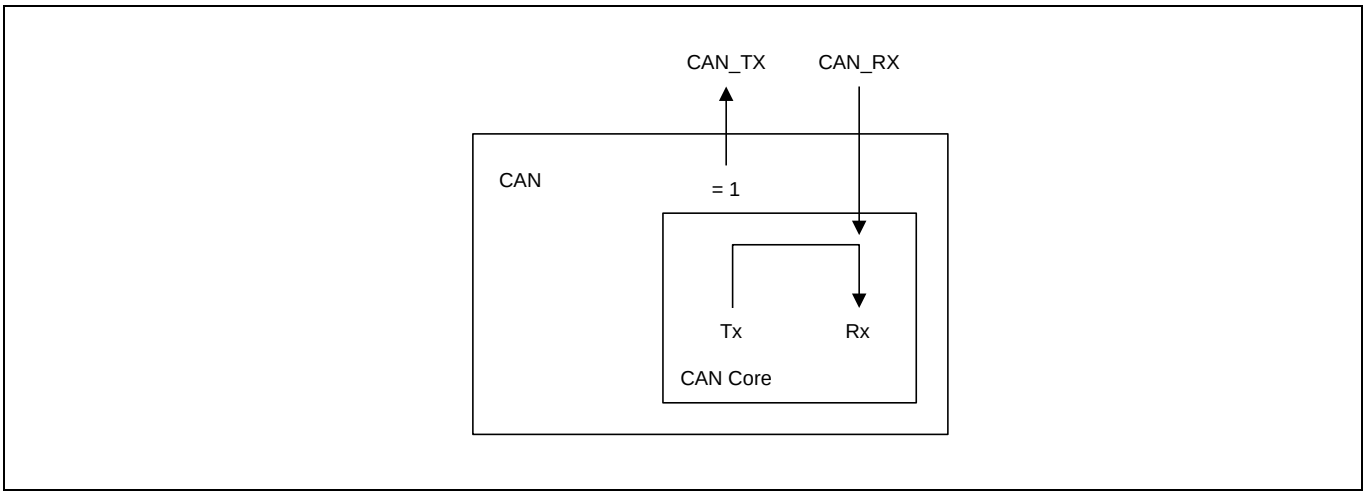


Figure 22-18 CAN Core in Silent Mode

在ISO 11898-1标准中，静默模式又被称作总线监控模式（Bus Monitoring Mode）。

22.3.6.2 回环模式

可以通过置位LBACK位将CAN核设置为回环模式。在回环模式下，CAN核只接收自己的发送报文（如果报文成功通过过滤处理的话），并将报文存储于接收缓存。下图描述了处于回环模式下的CAN核CAN_TX和CAN_RX的信号连接情况。

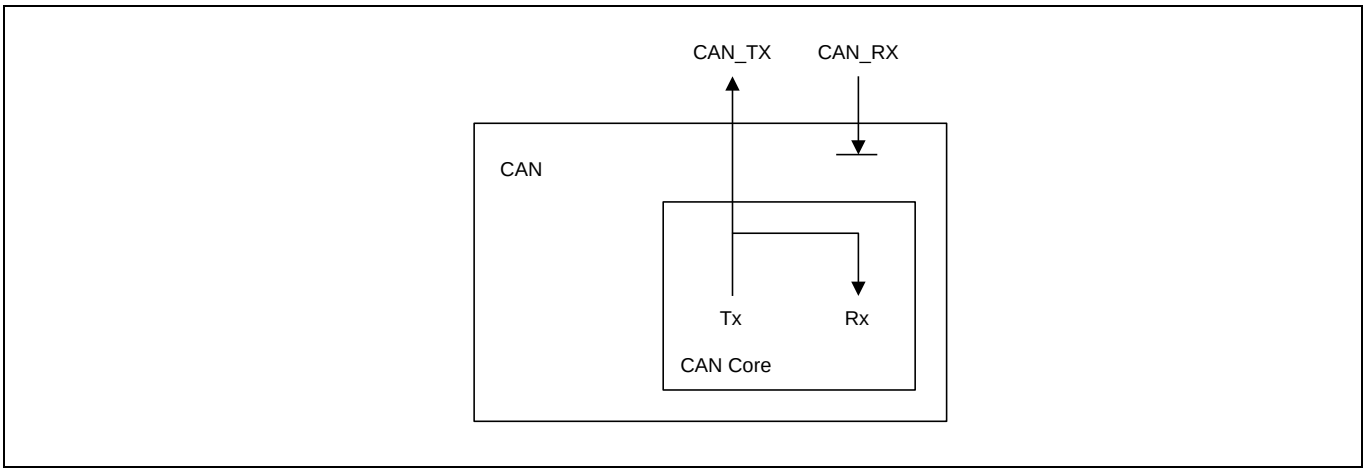


Figure 22-19 CAN Core in Loop Back Mode

这个模式适用于自测。为了不受外部刺激的影响，处于回环模式的CAN核会忽略应答错误（在数据/远程帧的应答时隙检测到隐性位）。并且，CAN核会自己内部生成从TX输出到RX输入的应答，而真正CAN_RX上的输入电平会被CAN核无视。发送报文可以在CAN_TX上监控到。

22.3.6.3 回环模式+静默模式

回环模式和静默模式可以同时开启，只要同时置位LBACK和SILENT控制位即可。这个模式可以用作“Hot Self Test”，即可以在不影响CAN_TX和CAN_RX连接到的正在运行的CAN系统的情况下，对CAN节点进行测试。在这个模式下，CAN_RX实际是和CAN核断开的，而CAN_TX则一直保持隐性值。图22-20描述了处于回环+静默模式下的CAN核CAN_TX和CAN_RX的信号连接情况。

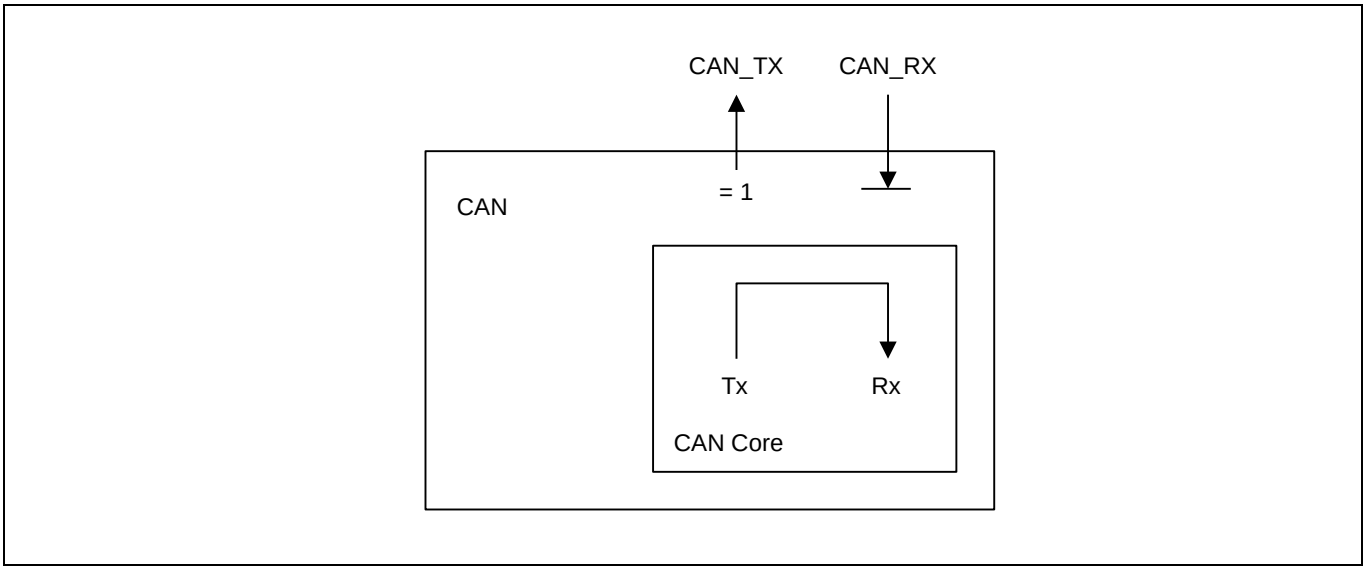


Figure 22-21 CAN Core in Loop Back Combined with Silent Mode

22.3.6.4 基础模式

可以通过置位BASIC位将CAN核设置为基础模式。在基础模式下，CAN模块脱离报文RAM运行。

IF0的寄存器被用于发送缓存。置位CAN_CR[RQBTX]即发起发送请求。当CAN_SR[BTXPD]位置高时，表示发送被挂起，此时，IF0的寄存器即被锁定。

一旦CAN总线限制，IF0的寄存器即被加载到CAN核的移位寄存器，从而开始数据发送。当发送结束时，BTXPD位会清零，IF0寄存器的锁定即接触了。

随时可以取消一个被挂起的发送（CAN_CR[ABBTX] = 1）。如果CPU取消一个挂起的发送，因为仲裁失败或错误而可能发起的重传也不会发生。

IF1的寄存器被用于接收缓存。在接收到一个报文后，移位寄存器里的内容会旁路过滤功能，进入IF1的寄存器。

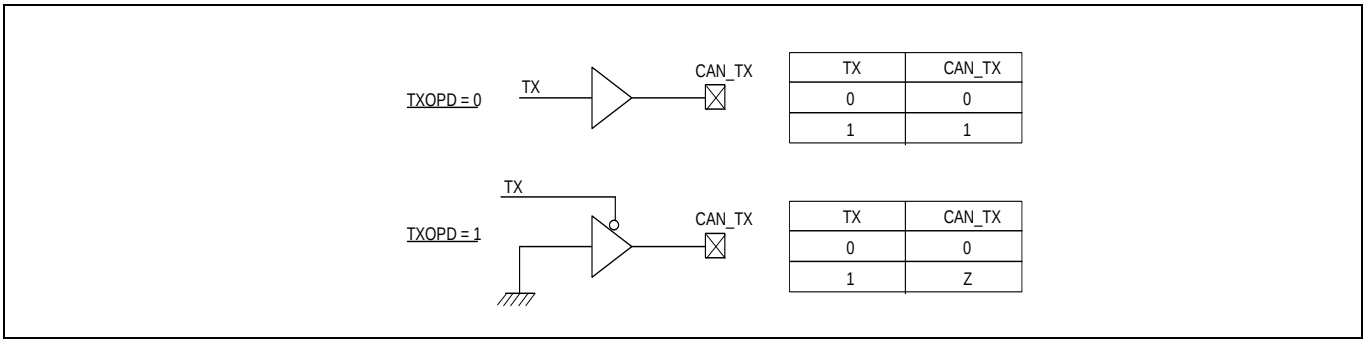
另外，报文传输期间，移位寄存器中真实的内容是可以被监视的。每次置位CAN_CR[STSR]即发起一次报文对象的读取，此时移位寄存器里的内容会进入IF1的寄存器。

在基础模式下，不再对所有报文对象相关的控制、状态位、IFx传输管理寄存器的控制位进行评估，传输管理寄存器中报文的数量也是一样。IF1报文控制寄存器中的NEWDAT和MSGLST位仍然有作用，DLC显示已经接收到的数据长度。读取其他控制位都会返回0。

22.3.7 CAN_TX管脚控制

CAN_TX作为CAN的发送管脚有4种输出功能。CAN_TX的默认功能是串行数据输出。除此之外，它监控CAN的采样点信号，从而监控CAN核的比特时间；也可以持续输出显性值或隐性值。最后的这两个功能，结合可读取的CAN接收管脚CAN_RX，可以用来检查CAN总线的物理层。

CAN_TX的输出模式可以通过配置测试寄存器CAN_TSRT[TX1/TX0]来实现。



TX	CAN_TX
0	0
1	1

TX	CAN_TX
0	0
1	Z

Figure 22-22 CAN_TX 管脚控制（CAN_TSTR[TX]和CAN_TSTR[TXOPD]）

当CAN在进行报文传输，或处于任何一个测试模式时（基础、回环或静默模式），CAN_TX必须处理其默认功

能。CAN_TX的三种测试功能（监控、输出0，输出1）可以在其他所有CAN协议功能下使用。

通过置位CAN_TSTR[TXOPD]将CAN_TX配置为开漏输出（只能输出低电平）。用户必须在开漏和正常输出模式切换前禁止CAN（CAN_CR[CANDIS]），以避免CAN_TX输出毛刺。此时，外接的上拉电阻使得这个管脚在没有被驱动时输出高电平。这个功能在下面两种情形时很有用:

- 这使得本地的CAN控制器可以在没有外部收发器的情况下，直接和总线上其他也是开漏输出的节点相连
- 这使得本地CAN控制工作于3.3V时，可以通过上拉到5V的电阻，驱动5V的收发器

22.3.8 总线关闭的恢复过程

当一个CAN控制器进入到总线关闭的状态时，它即处于被禁止的状态（CANENS = 0），并且不能继续参与总线活动。为了从这个状态恢复到正常的工作模式，CPU需要先使能CAN控制器（CANENS = 1），这时，CAN控制器会等待总线空闲，即连续128次出现11个连续隐性位。

最后，清除读写错误计数器，清除BUSOFF位。仅仅复位CAN（CAN_CR[SWRST] = 1）是不能结束节点的总线关闭状态的。

在CPU使能CAN控制器后，会有一个等待时间。一旦检测到连续的11个隐性位，状态寄存器的BIT0会置位，这将使得CPU开始检查CAN总线是否不能呈现显性位，或者不断被干扰，并开始监测总线关闭恢复的条件。

总线关闭恢复过程可以通过读REC计数器来获知：当CPU使能CAN的时候，REC计数器是0，每次检测到总线上出现连续11个隐性位，REC计数器就会加1。在恢复过程中，BUSOFF，ERPASS和CANENS位都处于置位状态，ERWARN位则处于清除状态。（参考CAN_SR寄存器描述）

22.4 CAN应用

22.4.1 报文对象的管理

芯片复位也不会对RAM区报文对象的设置产生影响，MSGVAL, NEWDAT, ITPND和TXRQST位除外。CPU需要对所有的报文对象进行初始化，不然就不能使用（CAN_IRx[MSGVAL] = 0）。同时在CPU置位CAN_CR[CANEN]位前必须完成对比特时间的配置。

报文对象的设置包括：往两个接口寄存器中的一组中写入想要的掩码、仲裁区、控制区和数据区。通过写入IFx的管理寄存器CAN_TMRx，IFx报文缓存寄存器的值会载入到报文RAM中可寻址的报文对象中。CPU写入CAN_TMRx即可发起一次报文RAM和IFx报文缓存寄存器之间的数据传输。

当CAN控制寄存器中的CANEN位置起时，CAN核的CAN协议控制器的状态机和报文处理状态机就开始一起控制CAN内部的数据流动。接收到的报文一旦通过过滤就会存储于报文RAM，发送请求被挂起的报文则被移入CAN核的移位寄存器，随后被发送到CAN总线上。

CPU通过IFx接口寄存器读取收到的报文并将要发送的报文发送出去。根据设置，CPU可以对一些CAN报文或错误事件形成的中断进行响应。

22.4.2 报文处理状态机

报文处理机控制着CAN核中Rx/Tx移位寄存器、报文RAM和IFx寄存器之间的数据传输，包括：

- 从IFx寄存器到报文RAM的数据传输
- 从报文RAM到IFx寄存器的数据传输
- 从移位寄存器到报文RAM的数据传输
- 从报文RAM到移位寄存器的数据传输
- 从移位寄存器到接收过滤单元的数据传输
- 在报文RAM区扫描匹配的报文对象
- 处理TXRQST 标志
- 处理中断

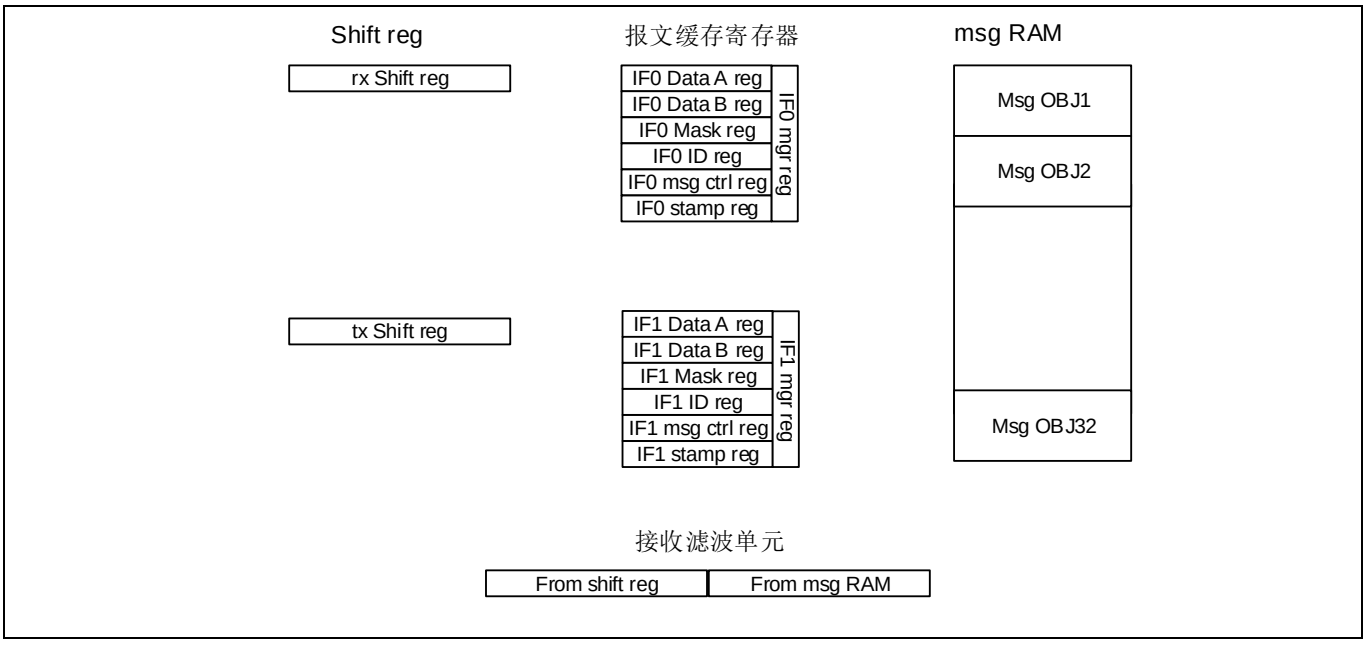


Figure 22-23 CAN核中的数据存储单元

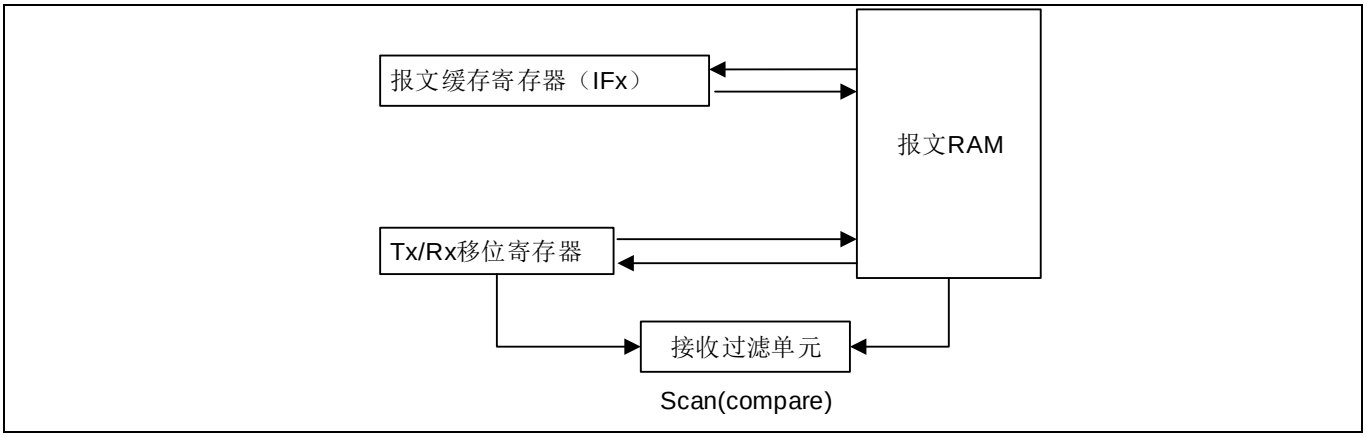


Figure 22-24 报文处理状态机管理的数据流动

22.4.2.1 接口寄存器和报文RAM间的数据传输

当CPU发起一次IFx寄存器和报文RAM区的数据传输时，报文处理机会将状态寄存器的BUSYx位置高。在传输完成后，即清除BUSYx位。 (参考 *Figure 22-25*).

各报文独立的传输管理寄存器（CAN_TMRx）规定了发送完整或是部分报文对象。但是报文RAM的结构决定了不可以只写入一个报文对象的几个比特或几位，而是要始终写入完整的报文对象。因此从IFx寄存器到报文RAM的数据传输需要执行一个读-改-写的操作。报文对象中不需要修改的部分从报文RAM区中读取，更新要更新的内容后，再将报文换存寄存器的内容写入报文对象。

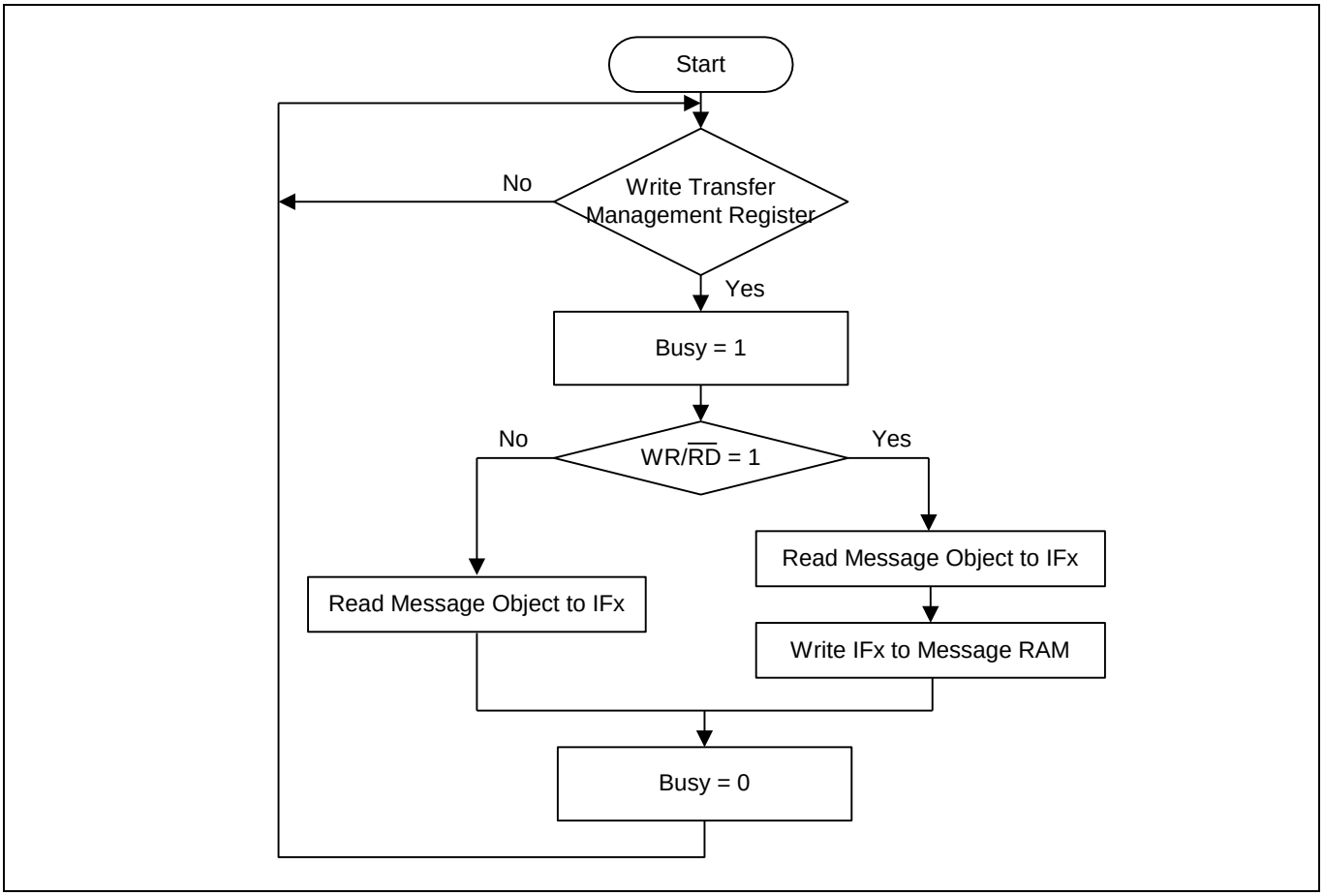


Figure 22-25 Data Transfer Between IFx Registers and Message RAM

在部分写入报文对象后，那个报文缓存寄存器，如果不被传输管理寄存器选中，就会完整呈现对应报文对象的真实内容。

在部分读取报文对象后，那个报文缓存寄存器，如果不能传输管理寄存器选中，将保持不变。

22.4.2.2 从报文对象到移位寄存器的报文传输

如果CAN核的移位寄存器已经做好加载的准备，且IFx寄存器和报文RAM之间没有数据传输，需要考虑CAN_IRx[MSGVAL]位以及CAN_MCR[TXRQST]位的状态。具有最高优先级发送请求（see *22.4.2.4* ）的有效报文对象将被报文处理机载入移位寄存器，从而开始发送。随后报文对象的NEWDAT被清除（参考CAN_MCR寄存器说明）。

发送成功后，且从上次写入后没有新的数据被再次写入报文对象（NEWDAT = 0），TXRQST位会被清除（参考CAN_MCR寄存器说明）。此时如果使能了TXIE，ITPND位会置起。如果CAN节点仲裁失败或者在传输过程中发生了错误，报文将被挂起，且在CAN总线重新空闲后立即开始重传。如果同时还有更高优先的报文在等待发送，多个报文将按照它们的优先级依次发送。

22.4.2.3 对移位寄存器中接收到报文的过滤处理

当正在接受报文的仲裁和控制区(ID + IDE + RTR + DLC)已完全移位进入CAN核的Rx/Tx移位寄存器时，报文处理状态机即开始从报文RAM中搜索扫描匹配的报文对象。

为了在报文RAM中扫描匹配的报文对象，会将CAN核中移位寄存器的仲裁位载入到接受滤波单元。随后报文对象1的仲裁区和掩码区（包括MSGVAL, UMASK, NEWDAT和OVERWRITE）也被载入接受滤波单元，并与来自移位寄存器的仲裁区内容进行比较。如此反复，直到在报文RAM区找到匹配的报文对象x或者到达报文RAM区的底部。

一旦找到匹配，扫描即会停止，报文处理状态机会根据接收到的帧类型（数据帧和远程帧）进行不同的处理。

22.4.2.3.1 数据帧的接收

报文处理状态机将来自CAN核移位寄存器的报文存储于报文RAM区独立的报文对象中。不仅仅是数据字节，还包括所有的仲裁位和数据长度码（DLC）。这么做是为了即便使用了仲裁掩码寄存器，仍然可以保留数据字节和它的标识符之间的联系。

NEWDAT位在接收到新的数据时（还未被CPU看见）会置高。CPU必须在读取一个报文对象后清除它的NEWDAT。如果接收到新数据时，NEWDAT已经为1，MSGLST为会置起，提示上一个还未被CPU处理的数据丢失。如果使能RXIE，ITPND位会置起，引起中断寄存器指向该报文对象。

报文对象的TXRQST位需要被清除以避免在刚刚接收到请求的数据帧时，发送一个远程帧。？？？The TXRQST bit of this Message Object is reset to prevent the transmission of a Remote Frame, while the requested Data Frame has just been received.

22.4.2.3.2 远程帧的接收

当接收到一个匹配的远程帧时，需要考虑对应报文对象的3种配置：

Configuration			
CAN_IR[MDIR]	CAN_MCRx[RMTEN]	CAN_MCRx[UMASK]	
1(tx)	1(enable)	0/1	TXRQST = 1
1(tx)	0(disable)	0(not use mask)	Ignore the remote frame

1(tx)	0(disable)	1(use mask)	TXRQST = 0； The arbitration and control field (ID + IDE + RTR + DLC) from the shift register is stored into the Message Object in the Message RAM and the NEWDAT bit of this Message Object is set. The data field of the Message Object remains unchanged; the Remote Frame is treated similar to a received **Data Frame**.

22.4.2.4 接收/发送优先级

报文对象的接收/发送优先级取决于这个报文的序号。报文对象1具有最高优先级，报文对象32有最小的优先级。如果有超过一个传输请求被挂起，它们被处理的顺序由对应的报文对象决定。

22.4.3 发送对象的配置

下表说明了如何初始化一个发送对象。

Table 22-11 Initialization of a Transmit Object

MSGVAL	ID	DATA	MSK	OVER WRITE	MDIR	NEWDAT	MSGLST	RXIE	TXIE	ITPND	RMTEN	TXRQST
1	Appl	Appl	Appl	1	1	0	0	0	Appl	0	Appl	0

引用代码决定了标识符寄存器（ID和CAN_IRx[XTD]），决定了标识符合发出报文的额类型。如果使用标准帧，需要使用11位的标识符，此时只需写入ID[28:18]，无需考虑ID[17:0]。

如果使能了TXIE，ITPND位会在报文对象的一次成功发送后置起。

如果CAN_MCRxp[RMTEN]被置高，扫描到一个与接收远程帧匹配的报文将会置高CAN_MCRx[TXRQST]；并主动发送对应的数据帧对远程帧做出应答。

数据寄存器（CAN_MCRx[DLC]，DATA[0:7]）也是应用决定的。在数据有效前，TXRQST和RMTEN不应该被置起。

如果UMASK = 1，掩码寄存器（MSK, UMASK, MXTD, 和MMDIR 位）可以用于接收具有相同标识符的远程帧，随后置位TXRQST。请小心处理。MDIR位不可以被屏蔽。

22.4.4 更新一个发送对象

CPU可以通过IFx接口寄存器在任意时刻更新一个发送对象的数据比特。在注意更新前要检查CAN_IRX[MSGVAL]和CAN_MCRx[TXRQST]，如果有任何一个为0，则不能进行更新。

即便只需要更新一部分数据字节，对用IFx数据A寄存器或IF数据B寄存器的四个字节在寄存器内容传输到报文

对象前都必须有效。或者CPU向IFx数据寄存器写入4个字节，或者在CPU写入新的数据前，将报文对象传入IFx数据寄存器。

为了避免TXRQST在传输结束被清零时，已经有数据正在更新，NEWDAT必须和TXRQST一起置高。详细请参考“从报文对象到移位寄存器的报文传输”。

当NEWDAT和TXRQST一起置高时，NEWDAT会在新的传输开始是立即清除。

22.4.5 接收对象的配置

下表说明了如何初始化一个接收对象。

Table 22-12 Initialization of a Receive Object

MSGVAL	ID	DATA	MSK	OVER WRITE	MDIR	NEWDAT	MSGLST	RXIE	TXIE	ITPND	RMTEN	TXRQST
1	Appl	Appl	Appl	1	0	0	0	Appl	0	0	0	0

应用代码决定了识别符寄存器的CAN_IRx[ID]和CAN_IRx [XTD]位，由此定义了识别符以及允许接受报文的类型。如果使用标准帧（11位识别符），只需要写ID[28:18]，而无需关注ID[17:0]。当收到一个11位识别符的数据帧时，ID[17:0]会被重写为零。

如果使能了RXIE，当接收到一个数据帧并存入报文对象后，ITPND标志位会置起。

数据长度码（CAN_MCRx[DLC]）也是由应用代码决定的。报文处理器在将一个数据帧存入数据对象后，会同时存入接收到数据的长度和8个数据字节。如果数据长度码小于8，剩下的数字字节会被非特定值填充。

当UMASK = 1时，掩码寄存器（UMASK, MXTD, MDIR位）可以用来屏蔽识别符不一致的数据帧。详细请参考“数据帧的接收”。在典型应用中，MDIR位不应该被屏蔽。

22.4.6 对接收报文的处理

CAN核在没有发送报文的时候，CPU可以在任意时刻通过IFx接口寄存器读取接收到的报文——检查CAN_SR[TS]标志位。报文处理状态机可以保证数据的一致性。

当CPU写传输管理寄存器时，将会把所有接收到的报文从报文RAM区移到报文缓存寄存器。同时，清除报文RAM区该报文对象的NEWDAT和ITPND位（不是报文缓存）。

如果这个报文对象使用了接收过滤，仲裁区的内容表示收到的是哪一个匹配的对象。NEWDAT表示自从上一个报文对象被读取后，是否收到了新的报文。

MSGLST表示自从上一个报文对象被读取后，是否接收到了超过一个报文（是否有报文丢失）。MSGLST不会被自动清除。

如果是远程帧，CPU可能要求其他CAN节点为接收到的对象提供新的数据。讲一个接收到对象的TXRQST位置高，将会发起一个远程帧的发送，且这个远程帧具有和接收到对象相同的标识符。这个远程帧的发送会使得其他拥有匹配识别符的数据帧的CAN节点开始发送报文。如果在远程帧发送之前收到匹配的数据帧，TXRQST位会自动复位

22.4.7 FIFO缓存的配置

除非重写CAN_MCRx，一个FIFO缓存记录的接收对象的配置应该和一个接受对象的配置是一样的。请参考“接受对象的配置”。

为了将两个或多个报文对象连接到一个FIFO缓冲区，这些报文对象的识别符和掩码（如果使用的话）必须写入匹配的值。因为报文对象的隐性优先级，具有最小序号的报文对象将会成为FIFO缓存的第一个报文对象。一个FIFO缓存中，除了最后一个，所有报文对象的OVERWRITE位必须写入零。而最后一个报文对象的这一位必须为1，以此表示数据块的结束。

22.4.8 使用FIFO缓存的报文接收

如果接收到报文的识别符匹配到一个FIFO缓存，这个报文将会被存储于这个FIFO缓存中具有最小报文序号的报文对象。这个报文的NEWDAT会被置起。当OVERWRITE为零且NEWDAT为1时，这个报文对象就被锁住了，以供后续报文处理器的访问，直到CPU清除NEWDAT位。

报文不断被存储于FIFO缓存，直到到达缓存最后一个报文对象。如果所有报文对象都没有被释放（清除NEWDAT），所有后来的属于这个FIFO缓存的报文都会覆盖最后一个报文对象，即上一个报文数据会丢失。

22.4.8.1 从FIFO缓存读取报文

当CPU通过将报文序号写入IFx传输管理寄存器（CAN_TMRx），将报文对象中的内容传输到IFx报文缓存寄存器时，NEWDAT和ITPND位都应该写为零（TRND = 1 且 CAN_TMRx[CLRIT] = 1）。报文控制寄存器的这些位始终反应被清除前的状态。

为了保证FIFO缓存的正常工作，CPU必须从具有最小报文序号的报文开始读取FIFO中的报文对象。

下图说明了CPU如何处理与一个FIFO缓存相关的一组报文对象。

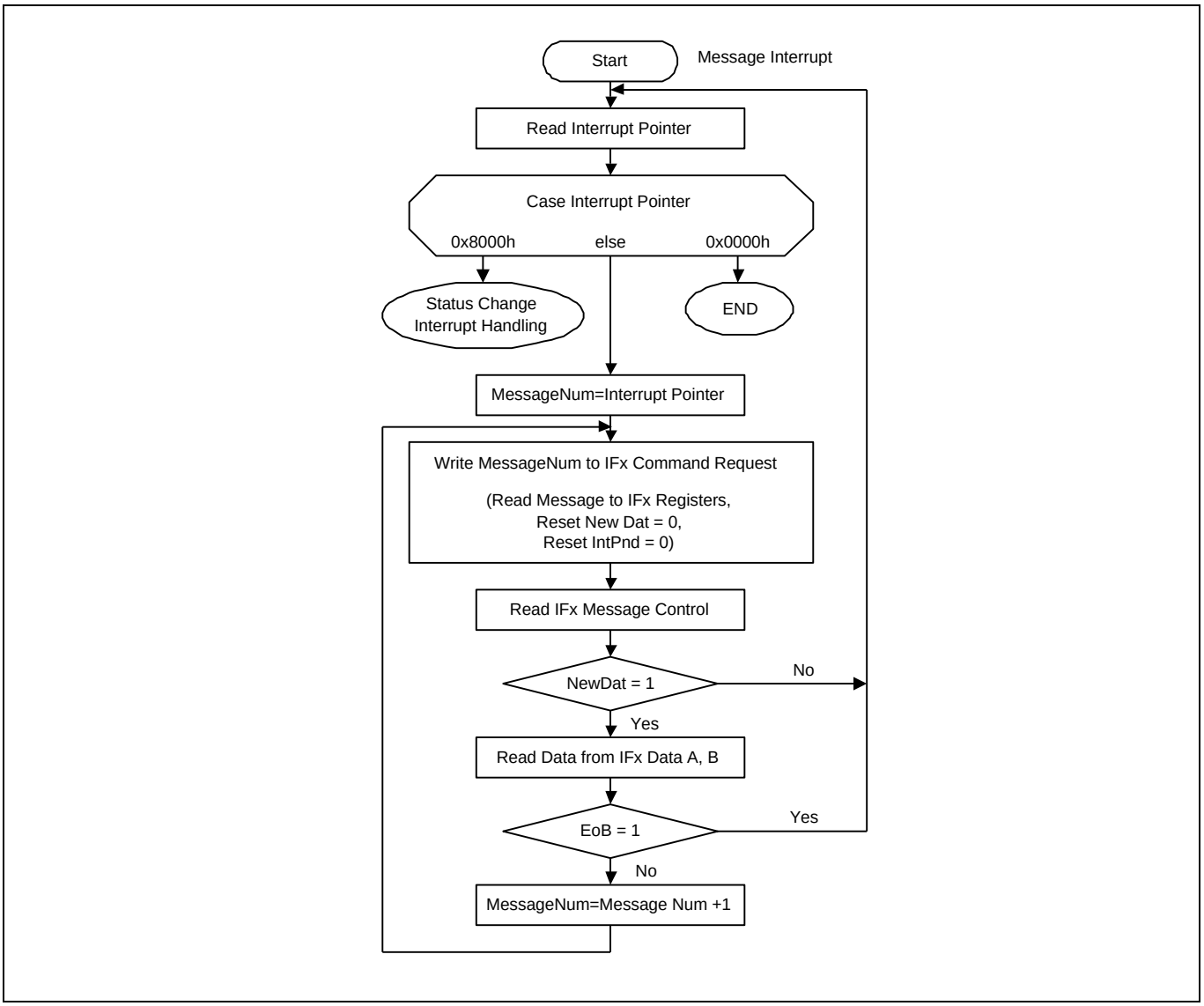


Figure 22-26 CPU Handling of a FIFO Buffer

22.4.9 中断处理

一个报文的中断请求状态可以通过清除该报文对象的ITPND位来清除。状态中断请求可以通过写清除状态寄存器来清除。

而是否发出中断请求，则可以通过中断掩码寄存器以及报文控制寄存器（CAN_TMRx[TXIE]和 [RXIE]）来实现。

22.4.10 比特时间的配置

尽管CAN比特时间一些轻微的配置错误不会导致立即失败，但是会很大程度地降低一个CAN网络的性能。

在很多情况下，CAN比特同步可以一定程度（通过偶尔产生一个错误帧）地弥补比特时间配置上的错误。仲裁时，如果两个或两个以上的CAN接口同时发起帧传输，错误的采样点可以引起其中的一个发送器成为错误被动节点。

要分析这种零散的错误需要对CAN节点内部的位同步以及CAN节点和CAN总线的交互有非常详细的知识。

22.4.10.1 时间份额

比特时间可以拆分的最小单位是时间份额（T_Q），其长度由CAN控制器系统时钟F_{CANCLK}以及波特率预分频器决定：$T_Q = (BD+1) \times t_{CANCLK}$。其中，BD取决于CAN_MR[BD]的设置，$F_{CANCLK}$来自CAN_CLK的输入。

22.4.10.2 比特时间和比特率

Table 22-13 Bit rate and Minimal Core Frequency

Bit Rate	Minimal CANCLK
125Kbits	1MHz
250Kbits	2MHz
500Kbits	4MHz
1Mbits	8MHz

CAN支持的比特率区间是1 kBit/s 到1000 kBit/s。每一个CAN网络中的成员都有自己的时钟发生器，通常情况下是晶振。每个CAN节点的比特时间的参数可以独立配置，即使CAN节点的振荡周期可能不同，也可以创建一个共同的比特率。

这些振荡器的频率并不是绝对稳定的，温度、电压或其他组件都可能引起一些小的偏差。只要这个偏差在一个特性振荡器的公差范围（df）内，CAN节点都可以通过重新同步比特流对不同的比特率进行补偿。

根据CAN的规范，比特时间被分为4个段（参考*Figure 22-27*）：同步段、传播时间段、相位缓冲段1和相位缓

冲段2。每个段都包含了个数或固定或可编程的时间份额（参考*Table 22-14*）。

在理想的传输环境下，CAN总线上电平沿应该出现在同步段（SYNC_SEQ）内，实际电平沿距同步段的距离称为相位误差。传播时间段是用来补偿CAN网络中的物理时延的。相位缓冲段1和2分别位于位采样点的左右。节点在位采样点上识别总线前位的电平状态。

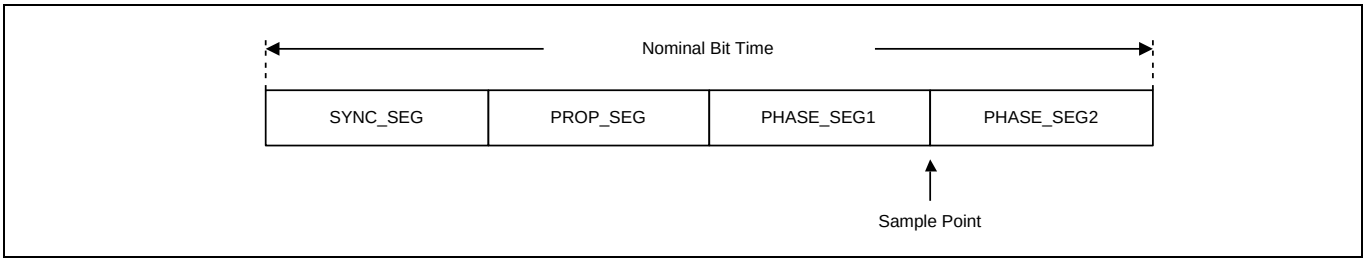


Figure 22-27 Bit Timing

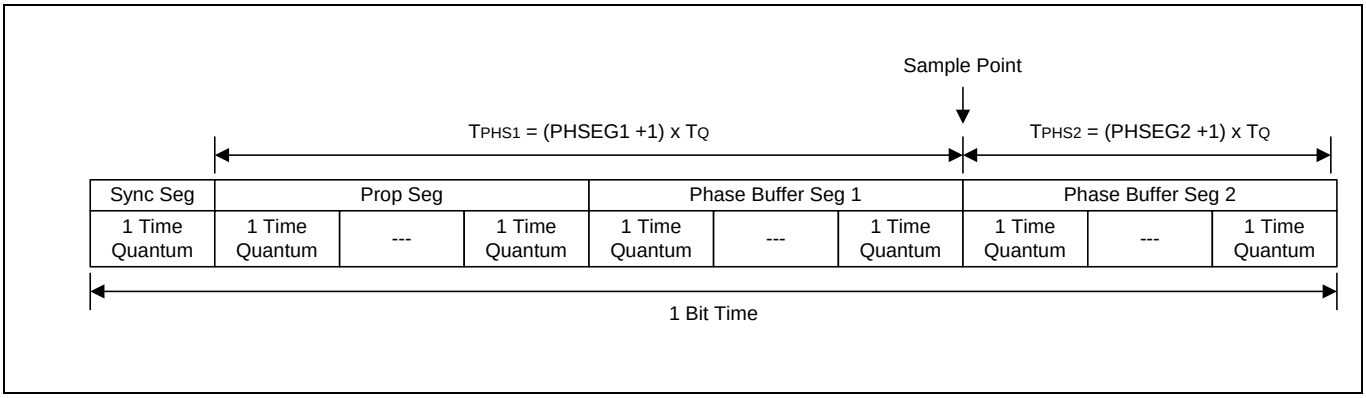


Figure 22-28 CAN Bit Time Built up @Time Quantum

Table 22-14 Parameters of the CAN Bit Time

Parameter	Range	Remark
BRP	[1..32]	Defines the length of the time quantum T_Q
Sync_Seg	1 T_Q	Fixed length, synchronization of bus input to system clock
Prop_Seg	[1..8] T_Q	Compensates for the physical delay times
Phase_Seg1	[1..8] T_Q	May be lengthened temporarily by synchronization
Phase_Seg2	[1..8] T_Q	May be shortened temporarily by synchronization
SJW (1)	[1..8] T_Q	May not be longer than either phase buffer segment
This table describes the minimum programmable ranges required by the CAN protocol		

Note (1) SJW（重同步跳转宽度）规定了重同步时，位采样点在由两个相位缓冲段限定的范围内，可以左右移动的距离。具体参考相位缓冲段和同步。

实现一个给定的比特率可能可以由多个比特时间配置实现，为了确保CAN网络的正常工作，需要考虑物理时延以及振荡器的容差范围。

22.4.10.3 传播时间段

传播时间段用于补偿网络内的物理时延。这个时延包括总线上的信号传播时间和CAN节点内部的处理时延。

Prog_Seg ≥ 2[max(node output delay + bus line delay + node input delay)]

22.4.10.4 相位缓冲段和同步

相位缓冲段1和2是用来补偿振荡器误差的。同步发生于隐性到显性位的跳变沿。同步时，相位缓冲段可能伸长或缩短，以此控制这个跳变沿和位采样点的距离。

节点在每个时间份额都会采样实际的总线电平，并和上一个时间份额采样到的总线电平比较，以此检测跳变沿。当且仅当上一次采样点采到隐性位，且当前时间份额采到显性位时，才会开始同步操作。

如果这个跳变发生于同步段内，则认为已经同步，否则认为存在相位误差。具体来说，同步段尾到跳变的距离即为相位误差，以时间份额来标定。跳变沿在同步段前就发生时相位误差为负，代表节点慢了；反之则为正，代表节点快了。

有两种同步行为：硬同步和重同步。硬同步只在帧开始时进行一次，帧内只可能发生重同步。在两个位采样点间，即每个位内只会进行一次同步。

- 硬同步

  总线处于空闲状态时，当接收节点检测出帧头（SOF）时，会将自己的同步段调整到和发送节点SOF的同步段一样，这个过程即硬同步。硬同步迫使引起硬同步的那个跳变位于重新开始的那个比特时间的同步段内，比特时间从同步段尾重新开始。

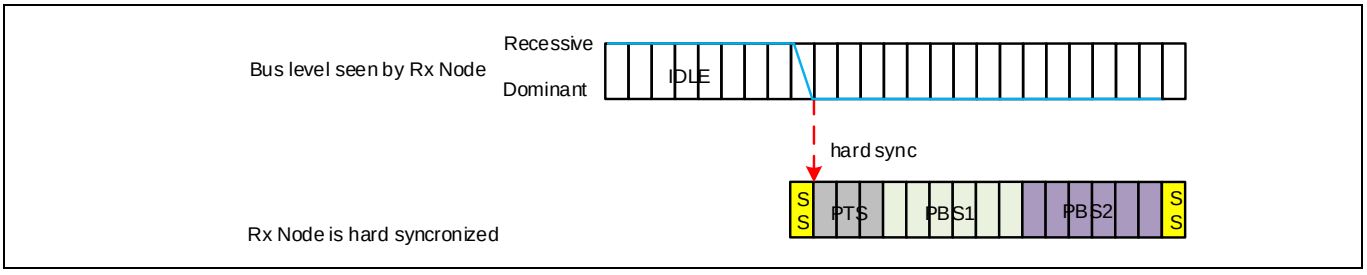


Figure 22-29 Hard Synchronization

- 重同步

  重同步指接收节点在除了硬同步以外的其他时间段进行的同步调整。重同步通过延长相位缓冲段1，或缩

短相位缓冲段2来调整同步，保证位采样点的准确性。调整的范围受同步跳转宽度（SJW）限制：

- SJW ＜ min(相位缓冲段1，相位缓冲段2)
- SJW ≤ 4

当引起重同步的相位误差为正，说明节点快了，需延长相位缓冲段1。当引起重同步的相位误差为负，说明节点慢了，需要缩短相位缓冲段2。小于SJW时按需延长或缩短，大于SJW时，调整的极限即为SJW。

重同步对相位缓冲段的影响只在当前位周期内有效。在接下来的位周期，只要没有重同步，各段仍然保持位时间的编程预设值。

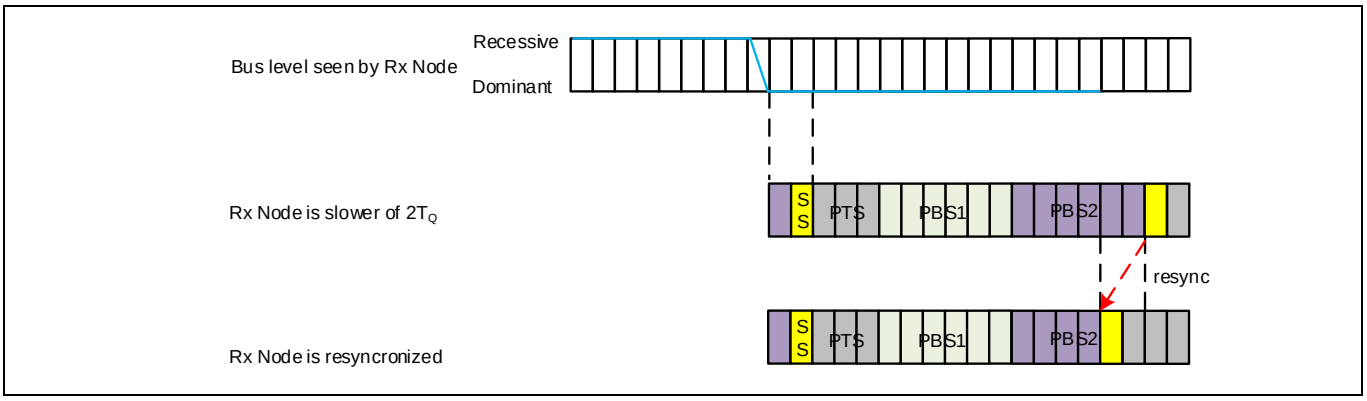


Figure 22-30 接收慢时的重同步

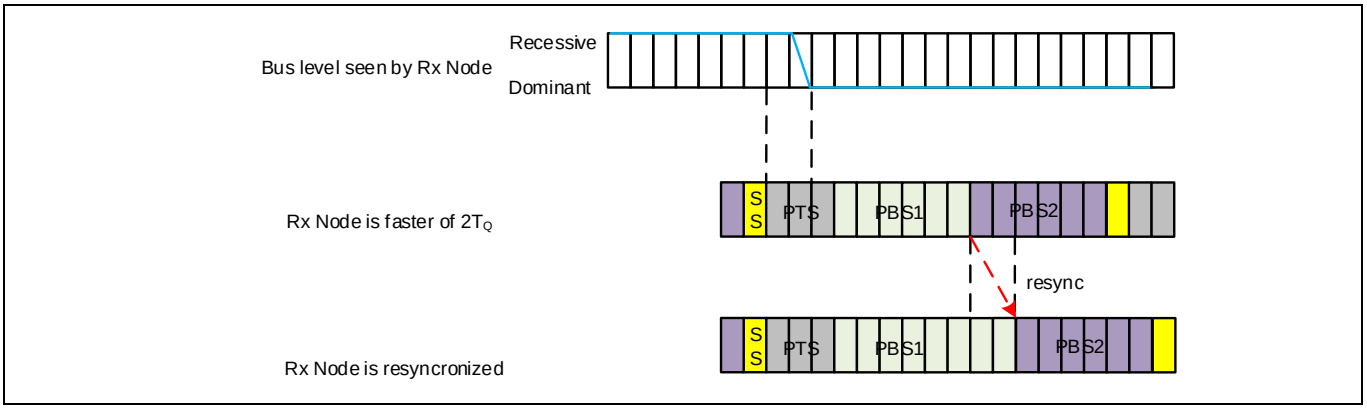


Figure 22-31 接收快时的重同步

22.4.10.5 振荡器容差范围

CAN协议从V1.1升级到V1.2时，振荡器容差范围增加了。曾经作为可选项的显性位到隐性位跳变的同步成为了必有项，只有从隐性位到显性位的跳变仍然保持可选。

从V1.2升级到V2.0（A和B）时，振荡器容差方面没有调整。

振荡器容差（df）是振荡器的振荡频率（f osc）偏移标称频率的范围：$(1 – df) \times f_{nom} \leq f_{osc} \leq (1 + df) \times f_{nom}$

由相位缓冲段1、2，SJW以及比特时间的比例决定。最大的容差取决于下面两个条件（同时满足）：

I: df ≤ [min (Phase_Seg1, Phase_Seg2)/2 x (13 x bit_time-Phase_Seg2)]

I: df ≤ [SJW/20 x bit_time]

需要考虑SJW可能小于或等于相位缓冲段中小的那个段的情况，还有传播时间段会限制可用于相位缓存段的那部分时间。

传播段 = 1，相位缓冲段1 = 相位缓冲段2 = SJW = 4，这个组合配置下，允许的最大振荡容差为1.58%。此时如果传播时间段长仅仅为10%的比特时间，对相对短的比特时间来说是不合适的。它可以用于中总线长度为40米，比特率最大为125kBit/s (bit time = 8ms)的场合。

22.4.10.6 CAN协议控制器的配置

在大多数CAN实现中，包括本实现，比特时间都是通过两个寄存器字节来控制的。

传播时间段和相位缓冲时间段1（TSEG1）和相位缓冲时间段2（TSEG2）位于同一个字节中；SJW和BRP在另一个字节中（见下图）。

在这两个时序寄存器中，必须将TSEG1，TSEG2，SJW和BRP这4个部分的标称值设为比实际值小1。因为可选范围是[0..n-1]，而不是[1..n]。这样，以SJW为例，实际值范围是[1..4]，只需要用2bit来表征就可以。

所以，比特时间的长度 = [TSEG1 + TSEG2 + 3] T_Q（写入值）

或 = [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] T_Q（实际值）

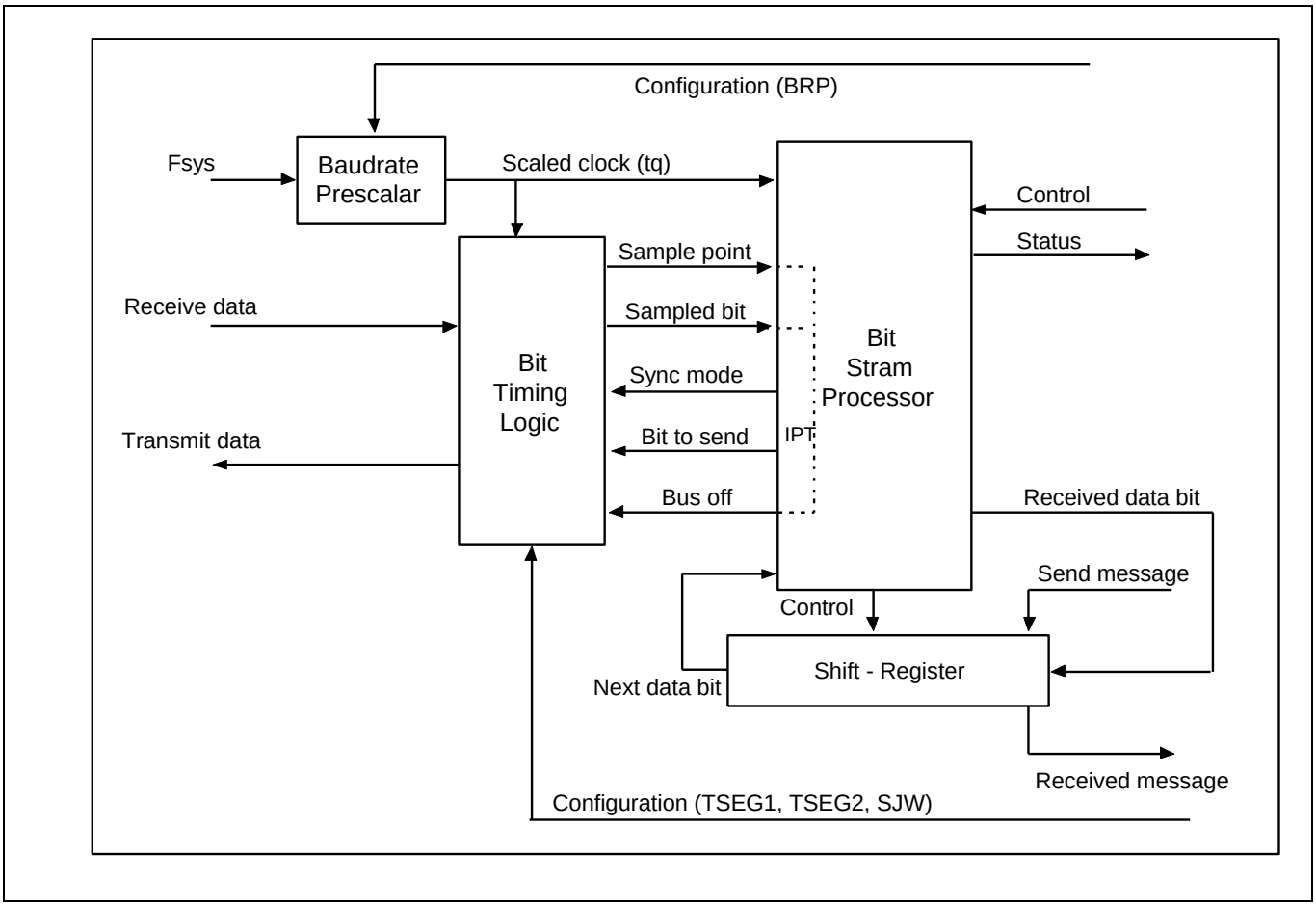


Figure 22-32 Structure of the CAN Controller Core

比特时间寄存器里的值作为输入进入CAN协议控制器。波特率预分频器（BRP）规定了时间份额的长度，即比特时间的基本时间单位。比特时序逻辑（TSEG1、TSEG2和SJW）决定了比特时间中各段的比特份额数。

比特时间的处理，采样点的计算，偶尔发生的同步都是右BTL（比特时间逻辑）状态机控制的。BTL状态机在每一个时间份额都会对上述处理进行评估。CAN协议控制器的其他部分，如BSP（比特流处理器）状态机则在每个比特时间进行一次评估，评估时刻即采样点。

移位寄存器对将要发送的报文串行输出，对接收的报文并行输入。BSP控制它的移位和加载。

BSP将报文解释为帧，或将帧打包为报文。它产生或剥离报文中固定的格式位，加入或提取填充位，计算或校验CRC码，进行错误管理并且决定需要使用哪种同步。BSP在每一个采样点工作，对采样到的总线输入位进行处理。采样点之后的时间，用于计算下一个将要被发送的位（如数据位，CRC位，填充位，错误标志或空闲）。这个时间被称为信息处理时间（IPT）。

IPT是由应用决定的，但不管怎么样都不能超过2T_Q；这个长度是相位缓冲段2可编程长度的下限。在发生同步时，相位缓冲段2可能会被缩减到小于IPT，这不会对总线时序产生影响。

22.4.10.7 比特时间参数的计算

通常情况下，比特时间配置的计算会从希望达到的比特率或比特时间开始。计算得到的比特时间（1/比特率）必须为系统时钟周期的整数倍。

比特时间可能包含4到25个时间份额，一个时间份额的长度由波特率预分频器决定：T_Q = (Baud Rate Prescalar + 1) / F_{CANCLK}。一样的比特时间可能可以通过不同的配置组合实现。以下步骤是可以迭代反复的：

最先要确定的部分是传输时间段。这将由系统中测量到的延时来决定。对一个可以扩展的CAN总线系统来说，需要事先规定最大的总线长度和最大的节点延时。测量（或规划）得到的传播时间段长度需要转换为以时间份额为单位（四舍五入到最近的整数）的数值。

同步段固定为1个时间份额，所以留给两个相位缓冲段的时间就是(bit time - Prop_Seg - 1)T_Q。如果剩下了偶数个时间份额，两个相位缓冲段就平分剩下的时间，Phase_Seg2 = Phase_Seg1。如果剩下了计数个时间份额，Phase_Seg2 = Phase_Seg1 + 1。

同时需要考虑相位缓冲时间段最小的标称长度。相位缓冲时间段不可以小于CAN控制器的信息处理时间（IPT）。IPT在不同的实现中，范围为[0..2] t_Q

将SJW的长度设为最大值，即4和相位缓冲段1两者中的小值。？？？The length of the Synchronization Jump Width is set to its maximum value, which is the minimum of 4 and Phase_Seg1.

振荡器容差范围参考“振荡器容差范围”章节给出的公式。

如果超过一种配置组合都可行，应该振荡器容差范围最大的那种。

具有不同系统时钟的CAN节点需要有不同的配置才能实现相同的比特率。CAN网络基于节点间最大延时的传播时延计算，对于整个网络只会进行一次。

CAN系统振荡器的容差范围由所有节点中具有最低容差的那个节点决定。

计算结果可能会提示为了找到符合协议的CAN比特时间配置，需要缩短总线长度或降低比特率或提高振荡频率的稳定性。

22.4.10.8 高波特率的比特时间举例

在这个例子中，CAN模块的时钟频率为10MHz，波特率预分频器设为0，比特率固定为1M Bit/s。

T_Q = t_{CANCLK} = 100ns
Bit time = 1us = 10 × T_Q
Delay of bus driver: 50ns
Delay of receiver circuit: 30ns
Delay of bus line (40m): 220ns
Prog_Seg = 2 × (Delay of bus driver) + 2 × (Delay of receiver circuit) + 2 × (Delay of bus line) = 600ns = 6 × T_Q

同步段固定为1个时间份额，所以留给两个相位缓冲段的时间就是(bit time - Prop_Seg - 1)T_Q。如果剩下了偶数个时间份额，两个相位缓冲段就平分剩下的时间，Phase_Seg2 = Phase_Seg1。如果剩下了计数个时间份额，Phase_Seg2 = Phase_Seg1 + 1。

$T_{Sync_Seg} = 1 \times T_Q$
T_Q remaining = 3, then
Phase buffer Seg 1 = $1 \times T_Q$
Phase buffer Seg 2 = $2 \times T_Q$

所以:
T_{PHS1} = Prop_Seg + Phase buffer Seg 1 = $7 \times T_Q$
T_{PHS2} = Phase buffer Seg 2 = $2 \times T_Q$
Bit time 1us = $T_{Sync_Seg} + T_{PHS1} + T_{PHS2}$
TSJW = min (4, Phase buffer Seg 1) = $1 \times T_Q$
PHSEG2[2:0] = (T_{PHS2} / T_Q) – 1 = 1
PHSEG1[3:0] = (T_{PHS1} / T_Q) – 1 = 6
AR = 1 (为了和CAN标准兼容)
SJW[1:0] = (T_{SJW} / T_Q) – 1 = 0
BD [9:0] = 0
CAN_MR = 0x00164000

22.4.10.1 低波特率的比特时间举例

在这个例子中，CAN模块的时钟频率为2MHz，波特率预分频器设为1，比特率固定为100K Bit/s。

$T_Q = 2 \times t_{CANCLK} = 1us$
Bit time = 10us = $10 \times T_Q$
Delay of bus driver: 200ns
Delay of receiver circuit: 80ns
Delay of bus line (40m): 220ns
Prog_Seg = $2 \times$ (Delay of bus driver) + $2 \times$ (Delay of receiver circuit) + $2 \times$ (Delay of bus line) = $1us = T_Q$

同步段固定为1个时间份额，所以留给两个相位缓冲段的时间就是(bit time - Prop_Seg - 1)T_Q。如果剩下了偶数个时间份额，两个相位缓冲段就平分剩下的时间，Phase_Seg2 = Phase_Seg1。如果剩下了计数个时间份额，
Phase_Seg2 = Phase_Seg1 + 1。

$T_{Sync_Seg} = 1\ us = 1 \times T_Q$
TQ remaining = 8, then
Phase buffer Seg 1 = $4 \times T_Q$
Phase buffer Seg 2 = $4 \times T_Q$

所以:
T_{PHS1} = Prop_Seg + Phase buffer Seg 1 = $5 \times T_Q$
T_{PHS2} = Phase buffer Seg 2 = $4 \times T_Q$
Bit time 10us = $T_{Sync_Seg} + T_{PHS1} + T_{PHS2}$
T_{SJW} = min (4, Phase buffer Seg 1) = $4 \times T_Q$
PHSEG2[2:0] = (T_{PHS2} / T_Q) – 1 = 3
PHSEG1[3:0] = (T_{PHS1} / T_Q) – 1 = 4
AR = 1 (to be compliant with CAN standard)
SJW[1:0] = (T_{SJW} / T_Q) – 1 = 3
BD [9:0] = 1
CAN_MR = 0x00347001

22.4.10.2 时间戳

所有接收到的报文都可以加盖一个32位宽的时间戳。

这个32位的寄存器时时间戳定时器（STT）的第二个计数器。在接收到每一个CAN帧后，这个计数器的计数值都会自动写入对应CAN通道的CAN_STPRX寄存器。

22.5 寄存器说明

22.5.1 寄存器表

Base Address of CAN: 0x400F0000

Register	Offset	Description	Reset Value
CAN_ECR	0x050	CAN时钟使能寄存器	0x00000000
CAN_DCR	0x054	CAN时钟禁止寄存器	0x00000000
CAN_PMSR	0x058	CAN时钟状态寄存器	0x00000000
CAN_CR	0x060	CAN控制寄存器	0x00000000
CAN_MR	0x064	CAN模式寄存器	0x00234001
CAN_CSR	0x06C	CAN清除状态寄存器	0x00000000
CAN_SR	0x070	CAN状态寄存器	0x00000010
CAN_IER	0x074	CAN中断使能寄存器	0x00000000
CAN_IDR	0x078	CAN中断使能寄存器	0x00000000
CAN_IMR	0x07C	CAN中断使能状态寄存器	0x00000000
CAN_ISSR	0x084	CAN源中断状态寄存器	0x00000000
CAN_SIER	0x088	CAN源中断使能寄存器	0x00000000
CAN_SIDR	0x08C	CAN源中断禁止寄存器	0x00000000
CAN_SIMR	0x090	CAN源中断使能状态寄存器	0x00000000
CAN_HPIR	0x094	CAN最高优先级中断寄存器	0x00000000
CAN_ERCR	0x098	CAN错误计数器	0x00000000
CAN_TMRx	0x100+0x020*x	CAN IF0/1传输管理寄存器	0x00000001
CAN_DARx	0x104+0x020*x	CAN IF0/1 Data A寄存器	0x00000000
CAN_DBRx	0x108+0x020*x	CAN IF0/1 Data B寄存器	0x00000000
CAN_MSKRx	0x10C+0x020*x	CAN IF0/1 掩码寄存器	0xDFFFFFFF
CAN_IRx	0x110+0x020*x	CAN IF0/1 识别符寄存器	0x00000000
CAN_MCRx	0x114+0x020*x	CAN IF0/1 报文控制寄存器	0x00000000
CAN_STPR	0x118+0x020*x	CAN IF0/1 时间戳寄存器	0x00000000
CAN_TRR	0x140	CAN发送请求寄存器	0x00000000
CAN_NDR	0x144	CAN接收报文有效寄存器	0x00000000
CAN_MVR	0x148	CAN报文有效寄存器	0x00000000
CAN_TSTR	0x14C	CAN测试寄存器	0x00000020

22.5.2 CAN_ECR(CAN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
R	R	R	R	R	R	R	R	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	调试模式使能状态： 0h: 无效 1h: 调试模式被使能
CLKEN	[1]	R	CAN模块时钟使能状态： 0h: 无效 1h：时钟被使能
NOTE: CORECLK 是SYSTEM CLOCK经过分频后，供CPU工作使用的clock，和HCLK同频。			

22.5.3 CAN_DCR(CAN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	调试模式使能： 0h: 无效 1h: 禁止调试模式
CLKEN	[1]	W	时钟使能： 0h: 无效 1h: 禁止CAN模块的时钟
NOTE: CORECLK 是SYSTEM CLOCK经过分频后，供CPU工作使用的clock，和HCLK同频。			

22.5.4 CAN_PMSR(CAN时钟状态寄存器)

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																													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	调试模式使能状态： 0h: 调试模式被禁止 1h：调试模式被使能
CLKEN	[1]	W	时钟使能： 0h: 时钟被禁止 1h：时钟被使能
NOTE: CORECLK 是SYSTEM CLOCK经过分频后，供CPU工作使用的clock，和HCLK同频。			

22.5.5 CAN_CR(CAN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																					STSR	ABBTX	RQBTX	RSVD			CCDIS	CCEN	CANDIS	CANEN	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	W	W	W	R			W	W	W	W	W

Name	Bit	Type	Description
STSR	[10]	W	加载移位寄存器： 0h：无效 1h：将移位寄存器的值加载到IF1寄存器（基础模式）。这样就可以监控移位寄存器内的真实值。 这位只在基础模式（CAN_TSTR[BASIC] = 1)时有效。
ABBTX	[9]	W	退出基础发送： 0h：无效 1h:退出IF0寄存器中内容的发送（基础模式） 这位只在基础模式（CAN_TSTR[BASIC] = 1)时有效。 当ABBTX和RQBTX同时置高时，ABBTX有更高的优先级。
RQBTX	[8]	W	请求基础发送： 0h：无效 1h：请求发送IF0寄存器中的内容（基础模式） 这位只在基础模式（CAN_TSTR[BASIC] = 1)时有效。
CCDIS	[4]	W	禁止改动配置： 0h：无效 1h：禁止对CAN_MR寄存器中比特时间配置的改动 当CCEN和CCDIS同时被置高时，CCDIS有更高的优先级。
CCEN	[3]	W	配置改动使能： 0h：无效 1h：允许对CAN_MR寄存器中比特时间配置的修改（当CAN处于禁止状态时，CAN_SR[CANENS] = 0)
CANDIS	[2]	W	禁止CAN： 0h：无效 1h：禁止CAN。禁止数据收发。如果在模块传输期间禁止CAN，会立即停止当前的传输。输出到CAN总线的CAN_TX呈阴性状态（高）。ERCR（错误计数器）保持不变。禁止CAN不会对任何配置寄存器产生影响。 当CANEN和CANDIS同时置高时，CANDIS有更高的优先级。
CANEN	[1]	W	使能CAN： 0h: 无效 1h：使能CAN模块。使能即开始CAN的软件初始化，发送和接收状态机回到复位状态。

			CAN使能后，BSP会等待总线上出现连续11个隐性位（总线空闲），以此将自己同步于CAN总线上的数据传输。在这之前，不能参与任何总线上的活动，也不能发起传输。这个时延是CAN标准的一部分。如果CAN节点因为进入总线关闭状态而被禁止，它仍然有机会重新被使能，但是需要等待129次检测到总线空闲（连续11个隐性位）。之后，CAN_SR[CANENS] = 1。 可以通过读取REC计数器来监控从总线关闭状态恢复的过程。CPU使能CAN时，会清零REC计数器，每当检测到总线上出现11个连续隐性位的时候，REC计数器加1。
SWRST	[0]	W	CAN软件复位： 0h：无效 1h：复位CAN 软件复位会触发CAN的硬件复位，复位除PMSR外所有的寄存器。存储于报文RAM区的数据不受硬件复位影响，但每个报文对象的MSGVAL，TXRQST, NEWDAT 和ITPND会回到复位状态。POR后， 报文RAM区的内容是不确定的。

22.5.6 CAN_MR(CAN模式寄存器)

Address = Base Address+ 0x064, Reset Value = 0x00234001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									PHSEG2			PHSEG1					AR	SJW			CSSEL	BD										
0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
									RW	RW	RW	RW	RW	RW	RW		RW	RW	RW		RW	RW										

Name	Bit	Type	Description
	[31:23]		
PHSEG2	[22:20]	RW	相位段2的长度。 这是位于采样之后的时间段。 TPHS2 = TQ □ (PHSEG2 + 1)
PHSEG1	[19:16]	RW	相位段1的长度。 这是位于采样点之前的时间段。包括了传播时间段和相位缓冲段1的长度。该值不能为0。如果写入0，则会自动改为1。
	[15]		
AR	[14]	RW	自动重传。 0h：不进行自动重传 1h：开启自动重发（如果要符合CAN标准，此位必须为1）
SJW	[13:12]	RW	同步跳转步长： CAN控制器在每一个发送跳变沿会进行一次再同步。（SJW+1）的值规定了一个比特周期可以加长或缩短的最大CAN时钟数。 TSWJ = TQ □ (SJW + 1)
	[11]		
CSSEL	[10]	RW	CAN时钟源选择： 0h: PCLK 1h：EMCLK
BD	[9:0]	RW	波特率预分频器： 时间份额计算公式：TQ = (BD + 1) / CANCLK 比特时间是由整数个时间份额组成的。
注：除了AR外，该寄存器所有位只有在CCENS = 1且CANENS = 0的时候可以修改。			

22.5.7 CAN_CSR(CAN清除状态寄存器)

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
																CRC	BIT0	BIT1	ACK	FORM	STUFF	TXOK	RXOK	RSVD			ACTVT	BUSOFFTR	ERPASSTR	ERWARNTR	RSVD
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
																W	W	W	W	W	W	W	W	R	R	R	W	W	W	W	R

Name	Bit	Type	Description
	[31:16]		
CRC	[15]	W	清除CRC错误标志位 0h：无效 1h：清除CRC错误标志位
BIT0	[14]	W	清除位低（bit to zero）错误标志位 0h：无效 1h：清除位低错误标志位
BIT1	[13]	W	清除位高（bit to one）错误标志位 0h：无效 1h：清除位高错误标志位
ACK	[12]	W	清除应答错误标志位 0h：无效 1h：清除应答错误标志位
FORM	[11]	W	清除形式错误标志位 0h：无效 1h：清除形式错误标志位
STUFF	[10]	W	清除填充错误标志位 0h：无效 1h：清除填充错误标志位
TXOK	[9]	W	清除报文发送成功标志位 0h：无效 1h：清除报文发送成功标志位
RXOK	[8]	W	清除报文接收成功标志位 0h：无效 1h：清除报文接收成功标志位
ACTVT	[4]	W	清除活动状态中断标志位 0h：无效 1h：清除活动状态中断标志位
BUSOFFTR	[3]	W	清除总线关闭转换中断标志位。 0h：无效 1h：清除总线关闭转换中断标志位
ERPASSTR	[2]	W	清除错误被动转换中断标志位

			0h：无效 1h：清除错误被动转换中断标志位
ERWARNTR	[1]	W	清除错误被动警告转换中断标志位 0h：无效 1h：清除错误被动警告转换中断标志位

22.5.8 CAN_SR(CAN状态寄存器)

Address = Base Address+ 0x070, Reset Value = 0x00000010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						BTXPD	CCENS	TS	RS	BUSY1	BUSY0	BUSOFF	ERPASS	ERWARN	CANENS	CRC	BIT0	BIT1	ACK	FORM	STUFF	TXOK	RXOK	RSVD			ACTVT	BUSOFFTR	ERPASSTR	ERWARNTR	ISS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
						R	R	R	R	R	R	R	R	R	R	R	R	W	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
	[31:26]		
BTXPD	[25]	R	基础传输挂起 0h：基础模式下没有发送请求或者传输已经完成或者已经离开基础模式 1h：IF0寄存器的发送被挂起或正在进行？？ 改位只在基础模式（CAN_TSTR[BASIC] = 1)下有效。
CCENS	[24]	R	配置更改使能 0h：禁止CPU修改CAN_MR中的比特时间配置位 1h：当CAN处于禁止状态时，CPU可以修改CAN_MR中比特时间配置位
TS	[23]	R	发送状态 0h：没有在发送模式。如果RS也为0，那么CAN控制器进入空闲状态。 1h：CAN控制器已经开始报文发送
RS	[22]	R	接收状态 0h：没有在接收模式。如果TS也为0，那么CAN控制器进入空闲状态。 1h：CAN控制器从空闲进入接收模式，或在传输中仲裁失败 注：如果CAN控制器在传输中仲裁失败，RS和TS位将在1个CAN时钟周期内置起。
BUSY1	[21]	R	IF1的BUSY标志 0h：IF1和报文RAM的读/写交互已完成 1h：IF0和报文RAM之间正在进行一个读/写交互
BUSY0	[20]	R	IF0的BUSY标志 0h：IF0和报文RAM的读/写交互已完成 1h：IF0和报文RAM之间正在进行一个读/写交互
BUSOFF	[19]	R	总线关闭 0h：节点没有处于总线关闭状态 1h：节点处于总线关闭状态 当CAN控制器进入总线关闭状态，它将被禁止（CANENS = 0），而不再参与任何总线活动。为了恢复到正常的工作状态，CPU必须重新使能CAN控制器（CAN_CR[CANEN] = 1)。然后，CAN控制器必须

			等待129次总线空闲状态（11个连续隐性位）的出现。总线关闭恢复过程最后一步，读写错误计数器都被清零，同时BUSOFF位也被清除。总线关闭恢复过程并会因为复位CAN（CAN_CR[SWRST] =1 ）而缩短。 CPU使能CAN总线后的等待期间，每次检测到11个连续的隐性位，SR中的BIT0就会置起，这使得CPU可以监控总线关闭的恢复过程（说明总线没有堵塞或被连续干扰）。
ERPASS	[18]	R	错误被动 0h：节点没有处于错误被动状态 1h：节点处于错误被动状态
ERWARN	[17]	R	错误被动警告 0h：没有错误被动警告 1h：至少一个错误计数器到达错误被动警告上限96
CANENS	[16]	R	CAN使能状态 0h：CAN被禁止。所有收发都不能进行，管脚CAN_TX隐性，所有错误计数器维持之前的计数。禁止CAN不会改变配置寄存器的内容。 1h：CAN被使能。节点可以参与总线活动，可以发起传输。
CRC	[15]	R	CRC错误标志位 0h：上一次报文传输没有出现CRC错误 1h：上一次报文传输中，出现了CRC错误：计算得到的校验和和报文中的CRC内容不一致
BIT0	[14]	R	位高（bit to zero）错误标志位 0h：上一次报文传输没有出现位低错误 1h：上一次报文传输中，出现了位低错误：除了标识符外，本节点发送的是显性位却监测到总线呈隐性位。在总线关闭恢复过程中，改位在每次检测到11个隐性位出现的时候都会置高。这使得CPU可以监控总线关闭恢复过程（说明总线没有堵塞或被连续干扰）。
BIT1	[13]	W	位高（bit to one）错误标志位 0h：上一次报文传输没有出现位高错误 1h：上一次报文传输中，出现了位高错误：除了标识符外，本节点发送的是隐性位却监测到总线呈显性位
ACK	[12]	R	应答错误标志位 0h：上一次报文传输没有出现应答错误 1h：上一次报文传输中，出现了应答错误：发送的报文没有收到来自其他节点的应答
FORM	[11]	R	清除形式错误标志位 0h：上一次报文传输没有出现形式错误 1h：上一次报文传输中，出现了形式错误（接收帧中某些具有固定形式的部分形式错误）
STUFF	[10]	R	填充错误标志位 0h：上一次报文传输没有出现填充错误 1h：上一次报文传输中，出现了填充错误（连续5个相同电平）

TXOK	[9]	R	清除报文发送成功标志位 0h：没有发送报文 1h：至少成功发送一个报文
RXOK	[8]	R	报文接收成功标志位 0h：没有接收到报文 1h：至少成功接收到一个报文
ACTVT	[4]	R	活动状态中断标志位 0h：CAN_RX上没有接收到活动（没有显性电平） 1h：CAN_RX上接收了活动状态（显性电平）
BUSOFFTR	[3]	R	总线关闭转换中断标志位。 0h：没有总线关闭转换事件发生 1h：至少发生一次总线关闭事件或从总线关闭恢复的事件
ERPASSTR	[2]	R	错误被动转换中断标志位 0h：没有错误被动转换事件发生 1h：至少发生一次错误被动事件或从错误被动恢复事件
ERWARNTR	[1]	R	错误被动警告转换中断标志位 0h：没有错误被动警告转换事件发生 1h：至少发生一次错误被动警告事件（有一个错误计数器记到了错误被动警告上限96）或从错误被动警告恢复事件
ISS	[0]	R	中断源状态 0h：任何通道都没有中断事件 1h：至少有一个通道产生中断事件且该中断已（在SIMR中)使能

22.5.9 CAN_IER(CAN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																CRC	BIT0	BIT1	ACK	FORM	STUFF	TXOK	RXOK	RSVD			ACTVT	BUSOFFTR	ERPASSTR	ERWARNTR	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	W	W	W	W	W	W	W	W	R	R	R	W	W	W	W	R

Name	Bit	Type	Description
CRC	[15]	W	使能CRC错误中断 0h：无效 1h：使能中断
BIT0	[14]	W	使能位低错误中断 0h：无效 1h：使能中断
BIT1	[13]	W	使能位高错误中断 0h：无效 1h：使能中断
ACK	[12]	W	使能应答错误中断 0h：无效 1h：使能中断
FORM	[11]	W	使能填充错误中断 0h：无效 1h：使能中断
STUFF	[10]	W	使能填充错误中断 0h：无效 1h：使能中断
TXOK	[9]	W	使能报文发送成功中断 0h：无效 1h：使能中断
RXOK	[8]	W	使能报文接收成功中断 0h：无效 1h：使能中断
ACTVT	[4]	W	使能活动转换中断 0h：无效 1h：使能中断
BUSOFFTR	[3]	W	使能总线关闭转换中断 0h：无效 1h：使能中断
ERPASSTR	[2]	W	使能错误被动转换中断 0h：无效

			1h：使能中断
ERWARNTR	[1]	W	使能错误被动警告转换中断 0h：无效 1h：使能中断

22.5.10 CAN_IDR(CAN中断使能寄存器)

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																CRC	BIT0	BIT1	ACK	FORM	STUFF	TXOK	RXOK	RSVD			ACTVT	BUSOFFTR	ERPASSTR	ERWARNTR	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	W	W	W	W	W	W	W	W	R	R	R	W	W	W	W	R

Name	Bit	Type	Description
CRC	[15]	W	使能CRC错误中断 0h：无效 1h：使能中断
BIT0	[14]	W	使能位低错误中断 0h：无效 1h：使能中断
BIT1	[13]	W	使能位高错误中断 0h：无效 1h：使能中断
ACK	[12]	W	使能应答错误中断 0h：无效 1h：使能中断
FORM	[11]	W	使能填充错误中断 0h：无效 1h：使能中断
STUFF	[10]	W	使能填充错误中断 0h：无效 1h：使能中断
TXOK	[9]	W	使能报文发送成功中断 0h：无效 1h：使能中断
RXOK	[8]	W	使能报文接收成功中断 0h：无效 1h：使能中断
ACTVT	[4]	W	使能活动转换中断 0h：无效 1h：使能中断
BUSOFFTR	[3]	W	使能总线关闭转换中断 0h：无效 1h：使能中断
ERPASSTR	[2]	W	使能错误被动转换中断 0h：无效

			1h：使能中断
ERWARNTR	[1]	W	使能错误被动警告转换中断 0h：无效 1h：使能中断

22.5.11 CAN_IMR(CAN中断使能状态寄存器)

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																CRC	BIT0	BIT1	ACK	FORM	STUFF	TXOK	RXOK	RSVD			ACTVT	BUSOFFTR	ERPASSTR	ERWARNTR	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
CRC	[15]	R	CRC错误中断使能状态 0h：禁止 1h：使能
BIT0	[14]	R	位低错误中断使能状态 0h：禁止 1h：使能
BIT1	[13]	R	位高错误中断使能状态 0h：禁止 1h：使能
ACK	[12]	R	应答错误中断使能状态 0h：禁止 1h：使能
FORM	[11]	R	填充错误中断使能状态 0h：禁止 1h：使能
STUFF	[10]	R	填充错误中断使能状态 0h：禁止 1h：使能
TXOK	[9]	R	报文发送成功中断使能状态 0h：禁止 1h：使能
RXOK	[8]	R	报文接收成功中断使能状态 0h：禁止 1h：使能
ACTVT	[4]	R	活动转换中断使能状态 0h：禁止 1h：使能
BUSOFFTR	[3]	R	总线关闭转换中断使能状态 0h：禁止 1h：使能
ERPASSTR	[2]	R	错误被动转换中断使能状态 0h：禁止

			1h：使能
ERWARNTR	[1]	R	错误被动警告转换中断使能状态 0h：禁止 1h：使能

22.5.12 CAN_ISSR(CAN源中断状态寄存器)

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
CH32	CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1
0	0	0	0	0	0	0	0	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
CH32	[31]		通道32中断
CH31	[30]		通道31中断
CH30	[29]		通道30中断
CH29	[28]		通道29中断
CH28	[27]		通道28中断
CH27	[26]		通道27中断
CH26	[25]		通道26中断
CH25	[24]		通道25中断
CH24	[23]		通道24中断
CH23	[22]		通道23中断
CH22	[21]		通道22中断
CH21	[20]		通道21中断
CH20	[19]		通道20中断
CH19	[18]		通道19中断
CH18	[17]		通道18中断
CH17	[16]		通道17中断
CH16	[15]		通道16中断
CH15	[14]		通道15中断
CH14	[13]		通道14中断
CH13	[12]		通道13中断
CH12	[11]		通道12中断
CH11	[10]		通道11中断
CH10	[9]		通道10中断
CH9	[8]		通道9中断
CH8	[7]		通道8中断
CH7	[6]		通道7中断
CH6	[5]		通道6中断
CH5	[4]		通道5中断
CH4	[3]		通道4中断
CH3	[2]		通道3中断
CH2	[1]		通道2中断
CH1	[0]		通道1中断
0h：该通道没有中断发生 1h：该通道上有中断发生：该通道 ITPND = 1			

22.5.13 CAN_SIER(CAN源中断使能寄存器)

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
CH32	CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1
0	0	0	0	0	0	0	0	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
CH32	[31]		通道32中断
CH31	[30]		通道31中断
CH30	[29]		通道30中断
CH29	[28]		通道29中断
CH28	[27]		通道28中断
CH27	[26]		通道27中断
CH26	[25]		通道26中断
CH25	[24]		通道25中断
CH24	[23]		通道24中断
CH23	[22]		通道23中断
CH22	[21]		通道22中断
CH21	[20]		通道21中断
CH20	[19]		通道20中断
CH19	[18]		通道19中断
CH18	[17]		通道18中断
CH17	[16]		通道17中断
CH16	[15]		通道16中断
CH15	[14]		通道15中断
CH14	[13]		通道14中断
CH13	[12]		通道13中断
CH12	[11]		通道12中断
CH11	[10]		通道11中断
CH10	[9]		通道10中断
CH9	[8]		通道9中断
CH8	[7]		通道8中断
CH7	[6]		通道7中断
CH6	[5]		通道6中断
CH5	[4]		通道5中断
CH4	[3]		通道4中断
CH3	[2]		通道3中断
CH2	[1]		通道2中断
CH1	[0]		通道1中断

0h：无效 1h：使用通道中断

22.5.14 CAN_SIDR(CAN源中断禁止寄存器)

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
CH32	CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1
0	0	0	0	0	0	0	0	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
CH32	[31]	W	通道32中断
CH31	[30]	W	通道31中断
CH30	[29]	W	通道30中断
CH29	[28]	W	通道29中断
CH28	[27]	W	通道28中断
CH27	[26]	W	通道27中断
CH26	[25]	W	通道26中断
CH25	[24]	W	通道25中断
CH24	[23]	W	通道24中断
CH23	[22]	W	通道23中断
CH22	[21]	W	通道22中断
CH21	[20]	W	通道21中断
CH20	[19]	W	通道20中断
CH19	[18]	W	通道19中断
CH18	[17]	W	通道18中断
CH17	[16]	W	通道17中断
CH16	[15]	W	通道16中断
CH15	[14]	W	通道15中断
CH14	[13]	W	通道14中断
CH13	[12]	W	通道13中断
CH12	[11]	W	通道12中断
CH11	[10]	W	通道11中断
CH10	[9]	W	通道10中断
CH9	[8]	W	通道9中断
CH8	[7]	W	通道8中断
CH7	[6]	W	通道7中断
CH6	[5]	W	通道6中断
CH5	[4]	W	通道5中断
CH4	[3]	W	通道4中断
CH3	[2]	W	通道3中断
CH2	[1]	W	通道2中断
CH1	[0]	W	通道1中断

0h：无效 1h：禁止通道中断发生

22.5.15 CAN_SIMR(CAN源中断使能状态寄存器)

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
CH32	CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1
0	0	0	0	0	0	0	0	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
CH32	[31]		通道32中断
CH31	[30]		通道31中断
CH30	[29]		通道30中断
CH29	[28]		通道29中断
CH28	[27]		通道28中断
CH27	[26]		通道27中断
CH26	[25]		通道26中断
CH25	[24]		通道25中断
CH24	[23]		通道24中断
CH23	[22]		通道23中断
CH22	[21]		通道22中断
CH21	[20]		通道21中断
CH20	[19]		通道20中断
CH19	[18]		通道19中断
CH18	[17]		通道18中断
CH17	[16]		通道17中断
CH16	[15]		通道16中断
CH15	[14]		通道15中断
CH14	[13]		通道14中断
CH13	[12]		通道13中断
CH12	[11]		通道12中断
CH11	[10]		通道11中断
CH10	[9]		通道10中断
CH9	[8]		通道9中断
CH8	[7]		通道8中断
CH7	[6]		通道7中断
CH6	[5]		通道6中断
CH5	[4]		通道5中断
CH4	[3]		通道4中断
CH3	[2]		通道3中断
CH2	[1]		通道2中断
CH1	[0]		通道1中断

0h：该通道中断已被禁止 1h：该通道中断已被使能

22.5.16 CAN_HPIR(CAN最高优先级中断寄存器)

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																INTID															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
INTID	[15:0]	R	最高优先级中断事件记录 0h：没有中断挂起 0x0001-0x0020：引起中断的报文对象序号 0x0021-0x7FFF：保留 0x8000：状态中断——状态寄存器中（ACTVT和ISS除外）至少有一个使能了的中断请求。参考CAN_SR。 0x8001-0xFFFF：保留 如果有超过一个中断被挂起，该寄存器会指定具有最高优先级的那个中断，即便这个中断不是最先到来的。中断状态位在CPU清除前一直为高。 相比通道中断，状态中断具有最高优先级。在通道中断中，报文对象序号越小优先级越高。

22.5.17 CAN_ERCR(CAN错误计数器)

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																TEC								REP	REC						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
TEC	[15:8]	R	发送错误计数值
REP	[7]	R	接收错误被动 0h：接受错误计数值<CAN规范中定义的错误被动上限 1h：接收错误已经超过错误被动上限
REC	[6:0]	R	接收错误计数值

22.5.18 CAN_TMRx(CAN IF0/1传输管理寄存器)

Address = Base Address+ 0x100, Reset Value = 0x00000001

Address = Base Address+ 0x120, 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																CLRIT	TRND	RSVD	AMCR	AIR	AMSKR	ADAR		WR	RSVD	NUMBER					
0	0	0	0	0	0	0	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	R	RW	RW	RW	RW	RW	RW	R	W	W	W	W	W	W

Name	Bit	Type	Description
CLRIT	[15]	RW	清除中断标志位 0h：ITPND位保持不变 1h：清除报文对象的ITPND 在写入模式时，该位无效。
TRND	[14]	RW	TXRQST和NEWDAT的操作 0h：TXRQST和NEWDAT不变 1h：写入时（WR=1），置位报文对象的TXRQST；读取时（WR=0），清除报文对象的NEWDAT 在写入模式下（WR=1），如果通过置位本寄存器的TXRQST来发起一次传输请求，将会忽略CAN_MCRx中的TXRQST位。 在读取模式下（WR=0），对报文对象的读取会和ITPND和NEWDAT的清除同时发生。 CAN_MCRx始终反应这些位在被清除之前的状态。
AMCR	[12]	RW	访问报文控制寄存器。 0h：报文控制位不变 1h：报文对象和CAN_MSKRx寄存器间需要传输报文控制位
AIR	[11]	RW	访问识别符寄存器。 0h：识别符不变 1h：报文对象和CAN_MSKRx寄存器间需要传输识别符
AMSKR	[10]	RW	访问屏蔽寄存器。 0h：屏蔽位不变 1h：报文对象和CAN_MSKRx寄存器间需要传输屏蔽位
ADAR	[9:8]	RW	访问Data A寄存器。 0h：Data 字节4~7 不变 1h：报文对象和CAN_DARx间需要传输 Data 字节4~7
WR	[7]	RW	读/写方向。 0h：读。传输从报文对象y到选中的IFx寄存器。 1h：写。传输从选中的IFx寄存器到报文对象y。
NUMBER	[5:0]	W	报文序号。 有效值为0x01到0x20（含）。

注意：

1、一旦CPU写入该寄存器，随即开始报文RAM区中和IFx报文缓存寄存器的报文传输。写入时，CAN_SR[BUSYx]自动置位，一旦传输完成就自动清除。当BUSYx=1时，不可以操作IFx的寄存器（DAR, DBR, MSKR, IR 和MCR）。该寄存器中大多数控制位的实际作用都与WR的状态有关。
2、报文序号32不可以用终于发送。

22.5.19 CAN_DARx(CAN IF0/1 Data A寄存器)

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

Address = Base Address+ 0x124, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA3								DATA2								DATA1								DATA0							
0	0	0	0	0	0	0	0	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
DATA3	[31:24]	RW	IFx的Data3 在一个CAN数据帧中，无论收发，总是DATA0在先，DATA7在末。 在一个字节中，MSB先发。
DATA2	[23:16]	RW	IFx的Data2 在一个CAN数据帧中，无论收发，总是DATA0在先，DATA7在末。 在一个字节中，MSB先发。
DATA1	[15:8]	RW	IFx的Data1 在一个CAN数据帧中，无论收发，总是DATA0在先，DATA7在末。 在一个字节中，MSB先发。
DATA0	[7:0]	RW	IFx的Data0 在一个CAN数据帧中，无论收发，总是DATA0在先，DATA7在末。 在一个字节中，MSB先发。
注：BUSYx = 1时，不可以操作该寄存器。			

22.5.20 CAN_DBRx(CAN IF0/1 Data B寄存器)

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

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
DATA3								DATA2								DATA1								DATA0							
0	0	0	0	0	0	0	0	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
DATA3	[31:24]	RW	IFx的Data3 在一个CAN数据帧中，无论收发，总是DATA0在先，DATA7在末。 在一个字节中，MSB先发。
DATA2	[23:16]	RW	IFx的Data2 在一个CAN数据帧中，无论收发，总是DATA0在先，DATA7在末。 在一个字节中，MSB先发。
DATA1	[15:8]	RW	IFx的Data1 在一个CAN数据帧中，无论收发，总是DATA0在先，DATA7在末。 在一个字节中，MSB先发。
DATA0	[7:0]	RW	IFx的Data0 在一个CAN数据帧中，无论收发，总是DATA0在先，DATA7在末。 在一个字节中，MSB先发。
注：BUSYx = 1时，不可以操作该寄存器。			

22.5.21 CAN_MSKRx(CAN IF0/1 掩码寄存器)

Address = Base Address+ 0x10C, Reset Value = 0xDFFFFFFF

Address = Base Address+ 0x12C, Reset Value = 0xDFFFFFFF

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

Name	Bit	Type	Description
MXTD	[31]	RW	XTD位掩码 0h：CAN_IRx[XTD] 位不参与接收滤波 1h：CAN_IRx[XTD] 位参与接收滤波
MMDIR	[30]	RW	报文方向位掩码 0h：CAN_IRx[MDIR] 位不参与接收滤波 1h：CAN_IRx[MDIR] 位参与接收滤波
BASEMASK	[28:18]	RW	基本标识符掩码 基本识别符和扩展识别符都会用到的11位基本识别符掩码。 如果CAN控制器接收到1个11位的识别符（标准帧），只需要用到高11位。当BASEMASK中的某一位为1时，标识符的对应位就会被用于接收滤波。当某一个为0时，则标识符的该位不参与过滤比较。
EXTMASK	[17:0]	RW	扩展标识符掩码 只用于扩展识别符的18位掩码。如果CAN控制器接收到29位识别符（扩展帧），可以使用所有的掩码位（包括EXTMASK和BASEMASK）。当EXTMASK中的某一位为1时，标识符的对应位就会被用于接收滤波。当某一个为0时，则标识符的该位不参与过滤比较。

22.5.22 CAN_IRx(CAN IF0/1 识别符寄存器)

Address = Base Address+ 0x110, Reset Value = 0x00000000

Address = Base Address+ 0x130, Reset Value = 0x00000000

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

Name	Bit	Type	Description
MSGVAL	[31]	RW	报文是否有效 0h：报文无效，报文处理器忽略该报文 1h：报文对象已完成配置，报文处理器需要考虑该报文 CPU必须在使能CAN（CAN_CR[CANEN]=1）前，清除所有不需要用到的报文对象的MSGVAL位。在对BASEID、EXTID、XTD、MDIR及DLC进行修改前，也必须前清除改位。不再需要用到某个报文时，也要清除该报文的MSGVAL位。
XTD	[30]	RW	扩展识别符 0h：该报文使用11位识别符（标准帧） 1h：该报文对象使用29位识别符（扩展帧）
MDIR	[29]	RW	报文方向 0h：接收。如果TXRQST=1，将会发送具有该报文对象识别符的远程帧。如果接收到标识符匹配的数据帧，该报文将会存储于报文对象中。 1h：发送 。如果TXRQST=1，报文对象将会作为数据帧发送。 在接收到识别符匹配的远程帧时： 如果RMTEN = ‘1’，置位该报文的TXRQST 如果RMTEN = ‘0’且UMASK=0，该远程帧被忽略 如果RMTEN = ‘0’且UMASK=1，接收报文的仲裁区和控制区（ID + IDE + RTR + DLC）将被存储于报文RAM的报文对象中，且该报文对象的NEWDAT置位。
BASEID	[28:18]	RW	IFx的基本识别符 基本识别符使用的11位，标准和扩展识别符都会用到。如果CAN发送11位识别符，只使用高11位。如果CAN发送一个29位的识别符，先发送BASEID，再发送EXTID。
EXTID	[17:0]	RW	IFx的扩展识别符 扩展识别符使用的18位。如果CAN发送一个29位的识别符，需要同时用到BASEID和EXTID。此时，先发送BASEID，再发送EXTID。

22.5.23 CAN_MCRx(CAN IF0/1 报文控制寄存器)

Address = Base Address+ 0x114, Reset Value = 0x00000000

Address = Base Address+ 0x134, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																NEWDAT	MSGLST	ITPND	UMASK	TXIE	RXIE	RMTEN	TXRQST	OVERWRITE	RSVD			DLC			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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	R	R	R	RW	W	W	W

Name	Bit	Type	Description
NEWDAT	[15]	RW	新数据 0h：没有新的数据到达 1h：报文处理器或者CPU往报文对象的数据区部分写入了新的数据
MSGLST	[14]	RW	报文丢失 0h：没有报文丢失 1h：报文处理器在NEWDAT仍然为1的时候载入了一个新的报文。CPU丢失报文。 该位只有在报文方向为接收时有效。
ITPND	[13]	RW	中断挂起 0h：该报文没有发生中断事件 1h：该报文发生了中断事件
UMASK	[12]	RW	使用接收过滤掩码 0h：不使用掩码 1h：使用掩码（BASEMASK, EXTMASK, MXTD and MMDIR）
TXIE	[11]	RW	发送中断使能 0h：在成功发送一个帧时，ITPND保持不变 1h：在成功发送一个帧时，ITPND置位
RXIE	[10]	RW	接收中断使能 0h：在成功接收到一个帧时，ITPND保持不变 1h：在成功接收到一个帧时，ITPND置位
RMTEN	[9]	RW	远程使能 0h：在接收到远程帧时，TXRQST位保持不变 1h：在接收到远程帧时，置位TXRQST
TXRQST	[8]	RW	传输请求 0h：报文对象没有有等待发送 1h：报文对象已经请求发送，但还没有发送完成
OVERWRITE	[7]	RW	覆盖模式 0h：报文对象处于正常模式。此模式下，新来的帧不会覆盖之前的帧 1h：报文对象处于覆盖模式。在该模式下，新来的帧会覆盖之前的帧 这个控制位，用来将2个或2个以上的报文（最多32个）级联成一个FIFO缓存。对于不属于FIFO缓存的报文对象，，这一位必须始终为

			0。
DLC	[3:0]	W	数据长度码 有效值为0~8。DLC值在每次接收到帧（数据/远程帧）的时候都会更新。
注：BUSYx = 1时，不可以操作该寄存器。基础模式下，IF0（发送接口）只有DLC位有效，IF1（接收接口）只有NEWDAT, MSGLST和DLC位有效。			

22.5.24 CAN_STPR(CAN IF0/1 时间戳寄存器)

Address = Base Address+ 0x118, Reset Value = 0x00000000

Address = Base Address+ 0x138, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STAMP																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
STAMP	[31:0]	R	时间戳值 这个32位的时间戳数据记录了IFx接收到报文的时间。该时间来源于STT秒计数器。

22.5.25 CAN_TRR(CAN发送请求寄存器)

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
CH32	CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1
0	0	0	0	0	0	0	0	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
CH32	[31]		通道32发送请求
CH31	[30]		通道31发送请求
CH30	[29]		通道30发送请求
CH29	[28]		通道29发送请求
CH28	[27]		通道28发送请求
CH27	[26]		通道27发送请求
CH26	[25]		通道26发送请求
CH25	[24]		通道25发送请求
CH24	[23]		通道24发送请求
CH23	[22]		通道23发送请求
CH22	[21]		通道22发送请求
CH21	[20]		通道21发送请求
CH20	[19]		通道20发送请求
CH19	[18]		通道19发送请求
CH18	[17]		通道18发送请求
CH17	[16]		通道17发送请求
CH16	[15]		通道16发送请求
CH15	[14]		通道15发送请求
CH14	[13]		通道14发送请求
CH13	[12]		通道13发送请求
CH12	[11]		通道12发送请求
CH11	[10]		通道11发送请求
CH10	[9]		通道10发送请求
CH9	[8]		通道9发送请求
CH8	[7]		通道8发送请求
CH7	[6]		通道7发送请求
CH6	[5]		通道6发送请求
CH5	[4]		通道5发送请求
CH4	[3]		通道4发送请求
CH3	[2]		通道3发送请求
CH2	[1]		通道2发送请求
CH1	[0]		通道1发送请求
0h：该通道没有在等待发送 1h：该通道已经发出发送请求但还没有完成发送			

22.5.26 CAN_NDR(CAN接收报文有效寄存器)

Address = Base Address+ 0x144, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH32	CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
CH32	[31]	R	通道x报文有效性
CH31	[30]	R	通道x报文有效性
CH30	[29]	R	通道x报文有效性
CH29	[28]	R	通道x报文有效性
CH28	[27]	R	通道x报文有效性
CH27	[26]	R	通道x报文有效性
CH26	[25]	R	通道x报文有效性
CH25	[24]	R	通道x报文有效性
CH24	[23]	R	通道x报文有效性
CH23	[22]	R	通道x报文有效性
CH22	[21]	R	通道x报文有效性
CH21	[20]	R	通道x报文有效性
CH20	[19]	R	通道x报文有效性
CH19	[18]	R	通道x报文有效性
CH18	[17]	R	通道x报文有效性
CH17	[16]	R	通道x报文有效性
CH16	[15]	R	通道x报文有效性
CH15	[14]	R	通道x报文有效性
CH14	[13]	R	通道x报文有效性
CH13	[12]	R	通道x报文有效性
CH12	[11]	R	通道x报文有效性
CH11	[10]	R	通道x报文有效性
CH10	[9]	R	通道x报文有效性
CH9	[8]	R	通道x报文有效性
CH8	[7]	R	通道x报文有效性
CH7	[6]	R	通道x报文有效性
CH6	[5]	R	通道x报文有效性
CH5	[4]	R	通道x报文有效性
CH4	[3]	R	通道x报文有效性
CH3	[2]	R	通道x报文有效性
CH2	[1]	R	通道x报文有效性
CH1	[0]	R	通道x报文有效性
0h：报文处理器不会处理该通道 1h：该通道的报文对象已经被配置，报文处理器需要处理该通道			

说明：接收到报文到对应的通道中时，对应的通道会被置位，直到用户去处理该报文(读取报文数据，并清除报文NEWDATA和ITPND)，对应的通道位会被清除

22.5.27 CAN_MVR(CAN报文有效寄存器)

Address = Base Address+ 0x148, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH32	CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
CH32	[31]	R	通道x新数据状态
CH31	[30]	R	通道x新数据状态
CH30	[29]	R	通道x新数据状态
CH29	[28]	R	通道x新数据状态
CH28	[27]	R	通道x新数据状态
CH27	[26]	R	通道x新数据状态
CH26	[25]	R	通道x新数据状态
CH25	[24]	R	通道x新数据状态
CH24	[23]	R	通道x新数据状态
CH23	[22]	R	通道x新数据状态
CH22	[21]	R	通道x新数据状态
CH21	[20]	R	通道x新数据状态
CH20	[19]	R	通道x新数据状态
CH19	[18]	R	通道x新数据状态
CH18	[17]	R	通道x新数据状态
CH17	[16]	R	通道x新数据状态
CH16	[15]	R	通道x新数据状态
CH15	[14]	R	通道x新数据状态
CH14	[13]	R	通道x新数据状态
CH13	[12]	R	通道x新数据状态
CH12	[11]	R	通道x新数据状态
CH11	[10]	R	通道x新数据状态
CH10	[9]	R	通道x新数据状态
CH9	[8]	R	通道x新数据状态
CH8	[7]	R	通道x新数据状态
CH7	[6]	R	通道x新数据状态
CH6	[5]	R	通道x新数据状态
CH5	[4]	R	通道x新数据状态
CH4	[3]	R	通道x新数据状态
CH3	[2]	R	通道x新数据状态
CH2	[1]	R	通道x新数据状态
CH1	[0]	R	通道x新数据状态
0h：报文处理器或CPU没有向该通道数据区写入新数据 1h：报文处理器或CPU向该通道数据区写入了新的数据			

说明：通道报文的MSGVAL配置为有效时，对应的通道位会被置位；发送/接收报文都有效

22.5.28 CAN_TSTR(CAN测试寄存器)

Address = Base Address+ 0x14C, Reset Value = 0x00000020

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

Name	Bit	Type	Description
TSTKEY	[31:16]	RW	测试寄存器KEY 只有在写入CAN_TSTR时有效。读时为0。 0x0C75：CAN_TSTR可以写入 其他值：CAN_TSTR不可以写入
RX	[6]	R	CAN_RX管脚电平值 0h：显性（CAN_RX = ‘0’） 1h：隐性（CAN_RX = ‘1’）
TXOPD	[5]	RW	CAN_TX管脚开漏 0h：CAN_TX管脚不配置为开漏输出 1h：CAN_TX管脚配置为开漏输出。这允许将CAN控制器在没有外部收发器的情况下直接连接或工作于3.3V时仍然可以通过外部上拉电阻正常驱动5V的收发器。 注意：这个控制位在切换时，需要关闭CAN，从而避免在CAN_TX管脚上次产生毛刺。
TX	[4:3]	RX	CAN_TX管脚的控制 当CAN在进行报文传输，或处于任何一个测试模式时（基础、回环或静默模式），CAN_TX必须处理其默认功能。CAN_TX的三种测试功能（监控、输出0，输出1）可以在其他所有CAN协议功能下使用。 TX[1:0] 控制 00 CAN核控制CAN_TX的状态 01 CAN_TX监控采样点 (CAN控制器应处于使能状态） 10 CAN_TX 输出显性值（‘0’） 11 CAN_TX 输出隐性值（‘1’）
LBACK	[2]	RW	回环模式 0h：不使用回环模式 1h：使用回环模式
SILENT	[1]	RW	静默模式 0h：不使用静默模式 1h：CAN模块处理静默模式
BASIC	[0]	RW	基础模式 0h：不使用基础模式 1h：IF0用作发送缓存，IF1用作接收缓存。参考“基础模式”中的详细描述。

23 时间戳计数器 (Stamp Timer)

23.1 概述

时间戳计数器（Stamp Timer）是一个 32 位计数器。Timer 工作在递增模式下，并支持自动重载功能。Stamp Timer 可提供基础定时/计数功能。

23.1.1 主要特性

- 32 位可编程递增计数器。
- 32 位预设计数器时钟分频器 (支持 On-the-fly 修改配置)。

23.2 功能描述

23.2.1 模块框图

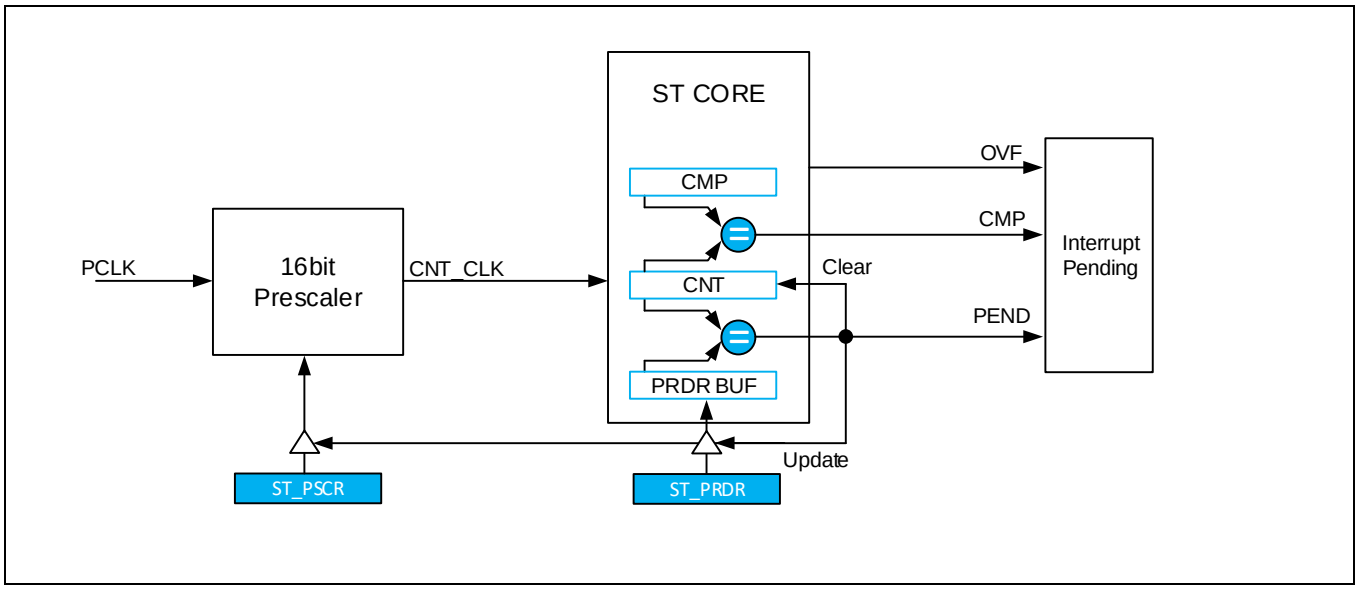


Figure 23-1 模块结构示意图

23.2.2 基本功能描述

Stamp 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。

当清除START控制位时，可以停止ST的工作。START控制位具有SHADOW功能，当START的SHADOW使能时，START被清除后，ST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ST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，当周期事件(PEND)发生时，ST会将SHADOW寄存器中的设置更新到ACTIVE寄存器中。除了PEND事件，还可以通过软件写CR[UPDATE]控制位，触发强制更新。或者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）。

ST支持One Pulse工作模式，通过CR[OPM]控制位选择。当OPM被使能后，ST在启动后只进行一次计数。当CNT=PRDR后，计数器停止计数。当CR[CNTRLD]被禁止时，OPM无效。

23.2.3 工作模式

Stamp 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。

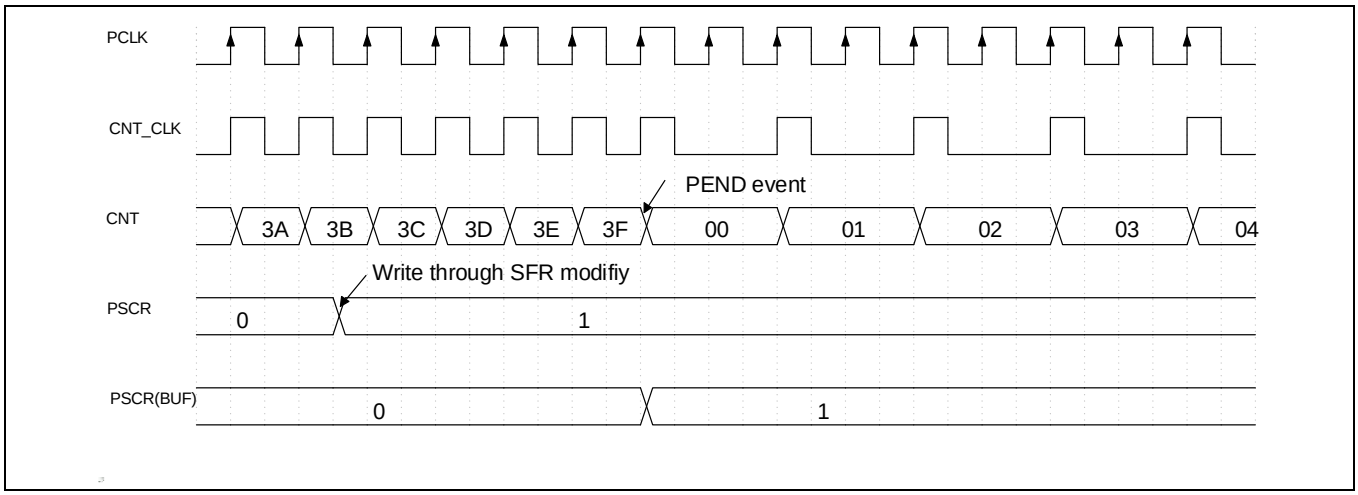


Figure 23-2 ST 周期计数模式

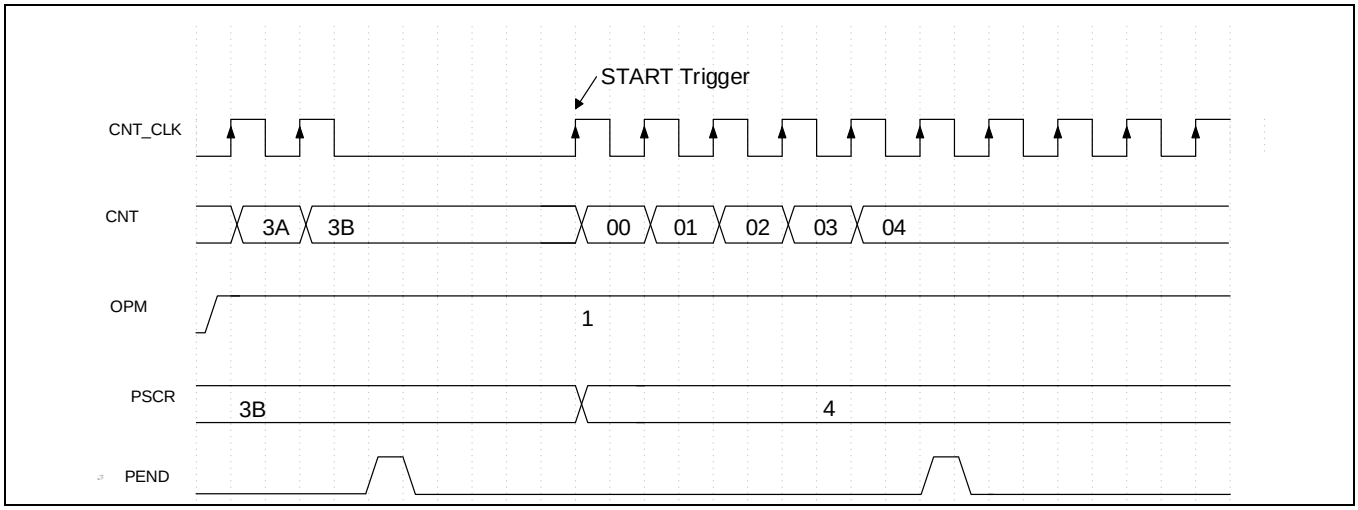


Figure 23-3 ST One Pulse 计数模式

23.2.4 中断控制

Stamp Timer支持3种中断，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。
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。

23.3 寄存器说明

23.3.1 寄存器表

Base Address of STAMP: 0x400F1000

Register	Offset	Description	Reset Value
ST_RSSR	0x000	复位/启停控制寄存器	0x00000000
ST_CR	0x004	通用控制寄存器	0x00000000
ST_PSCR	0x008	计数器预分频寄存器	0x00000000
ST_PRDR	0x00C	周期值寄存器	0x00000000
ST_CMP	0x010	比较值寄存器	0x00000000
ST_CNT	0x014	计数值寄存器	0x00000000
ST_RISR	0x028	原始中断状态寄存器	0x00000000
ST_ICR	0x034	中断清除寄存器	0x00000000

23.3.2 ST_RSSR(复位/启停控制寄存器)

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

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

23.3.3 ST_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															CNTRLD	RSVD										EXTCKM	OPM	SHDWSTP	UPDATE	DBGEN	CLKEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	W	RW	RW

Name	Bit	Type	Description
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。
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]	W	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。

23.3.4 ST_PSCR(计数器预分频寄存器)

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)

23.3.5 ST_PRDR(周期值寄存器)

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。

23.3.6 ST_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，下个计数周期开始BT输出将翻转。

23.3.7 ST_CNT(计数值寄存器)

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。

23.3.8 ST_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															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
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			

23.3.9 ST_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															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	W	W	W

Name	Bit	Type	Description
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’			

24 模数转换器 (ADC)

24.1 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。

24.1.1 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
- 自带固定电压参考源(INTVREF)
- 支持多路外部模拟输入AIN[23:0]，BIN[22:0]，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: 3MSPS
- 模拟输入范围：AVSS 到 AVREF

24.1.2 管脚描述

Table 24-1 ADC管脚描述

管脚名称	功能	I/O类型	有效电平	说明
INTVREF	模拟参考电压	模拟	-	-
AIN0 to AIN23	模拟信号输入	模拟	-	-
BIN0 to BIN22	模拟信号输入	模拟	-	-

24.1.3 模块框图

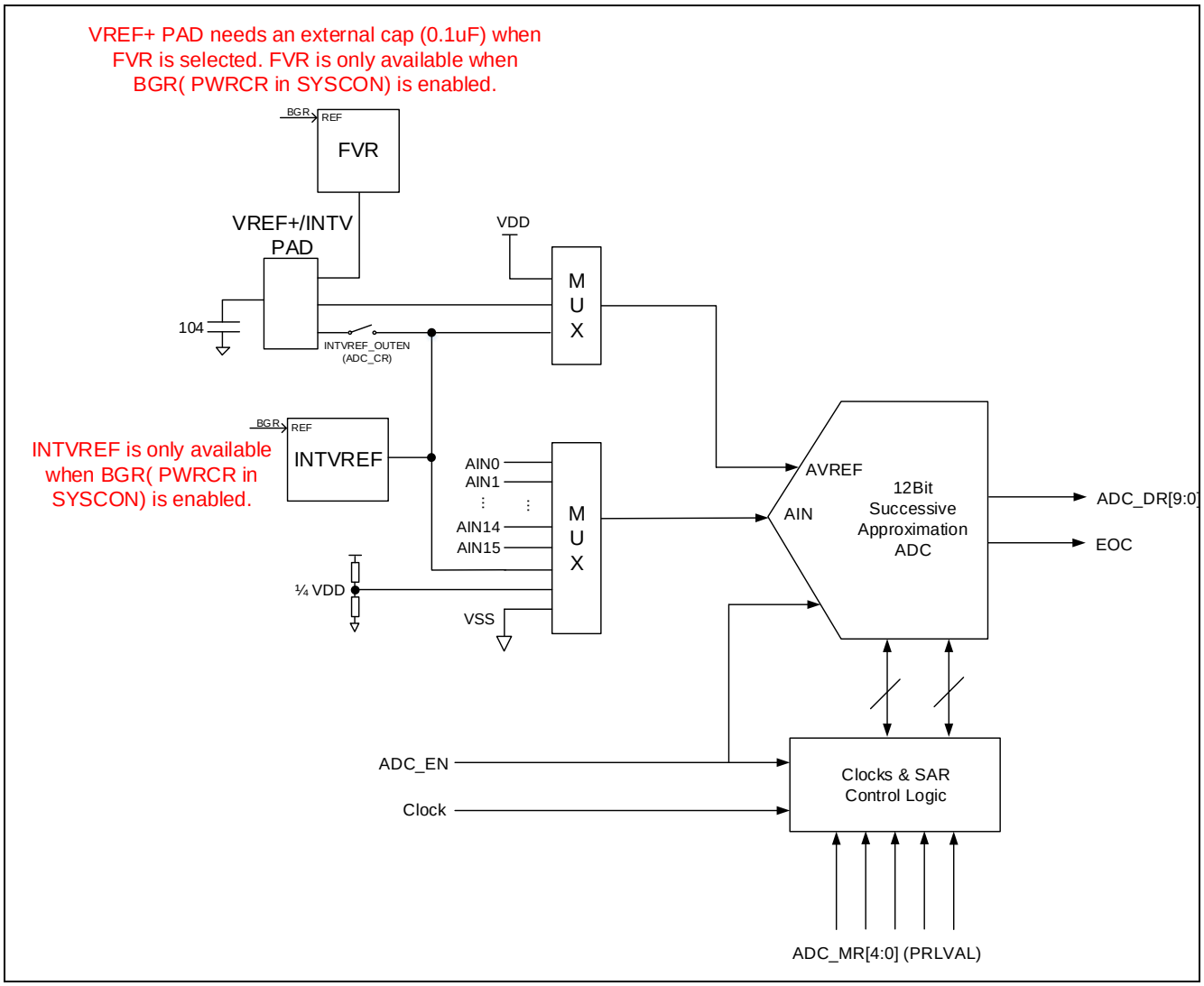


Figure 24-1 ADC模块框图

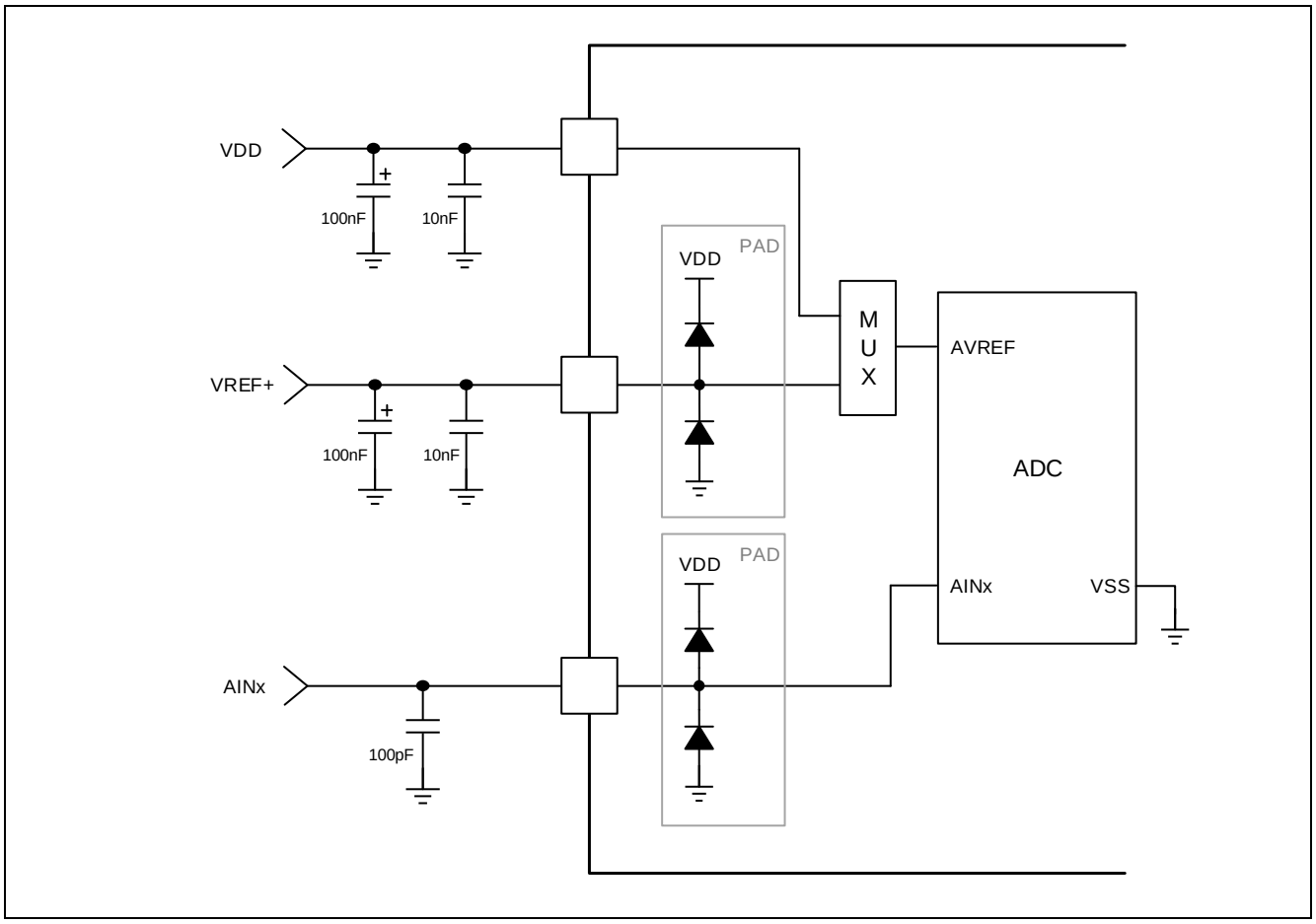


Figure 24-2 参考电路

24.1.4 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 24-2 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

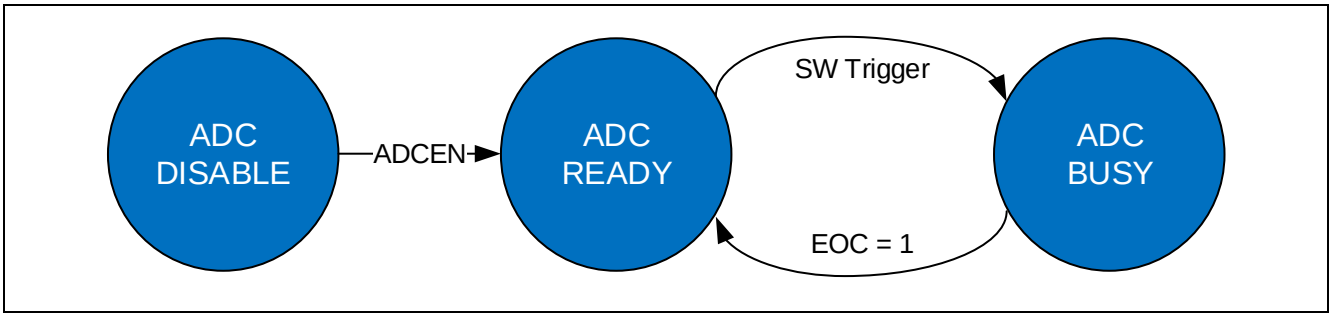


Figure 24-3 ADC状态机

24.1.5 参考电压源(AVREF)选择

ADC的支持参考电压源选择，由ADC_CR寄存器中的VREF_SEL位控制。正向电压参考源提供芯片VDD，固定电压源FVR（使用FVR做参考时，外部需要接100nF的电容），内部参考电压INTVREF输出，各种参考源的选择参见ADC_CR寄存器的VREF_SEL控制位。

如果希望使用固定电压源FVR提供参考电压，有两个配置需要满足：

1. 在GPIO的配置中使能对应的AF功能

2. ADC_CR中VREF_SEL选择FVR作为正向参考电压.

使用固定电压源作为ADC的参考电压，实际是将FVR电压输出到BUF，再通过BUF连到ADC的参考电压上。固定电压源模块提供三个不会随着芯片电源VDD变化的固定电压2.048V，3.072V和4.096V，可以通过FVR进行选择。

如果希望使用内部参考电压INTVREF作为参考，只需要使用ADC_CR中的VREF_SEL选择INTVREF作为相应参考即可。

注意，如果使用FVR作为参考，需要在VREF管脚上增加一个0.1uF的电容，参考 Figure 11-1 ADC模块框图。

24.1.6 时钟频率和转换时间

ADC 工作的时钟是从 PCLK 或 PLLQ 获得的。AD 转换的过程需要总共(S/H+13+3)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 ADCCLK = PCLK or PLLQ

否则 PRLVAL 是其它任何值的话，ADCCLK = $\frac{PCLK}{PRLVAL}$ or $\frac{PLLQ}{PRLVAL}$

如果选择使用 PLLQ，且 PLLQ 频率是 96MHz，PRLVAL = 2，那么一个 PLLQ 周期约为 10ns，一个 ADCCLK 周期约为 21ns。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) * $\frac{1}{ADCCLK}$ + 3 个同步和结果处理时钟周期 x $\frac{1}{\mathrm{PLLQ}}$ =

18 x 21ns + 3 x 10ns = 408ns (2.45MSPS)

PRLVAL 的值必须保证 ADCCLK ≤ 48MHz。

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)) x $\frac{1}{ADCCLK}$

例如：ADCCLK = 48MHz, ADC_SHR 设置 0x5 即 6+(5-3)=8 个周期，那么 S/H 时间为：8*21ns = 168ns

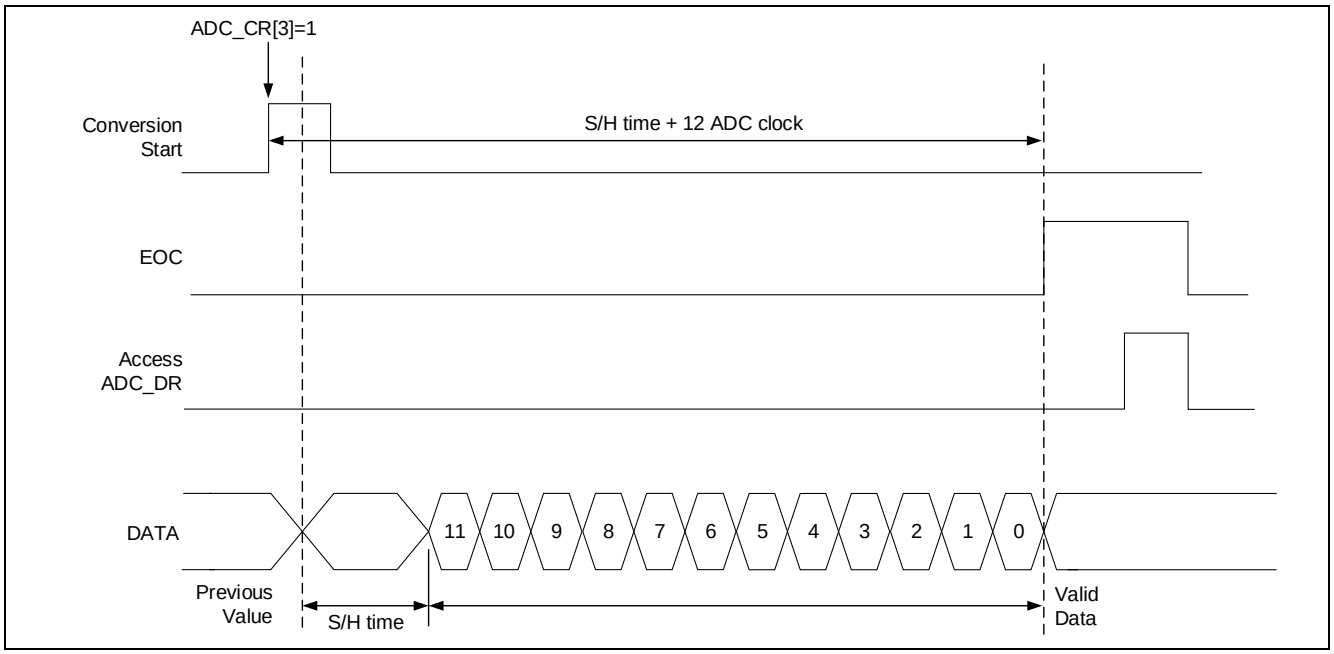


Figure 24-4 ADC工作时序图

24.1.7 转换序列定义

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

Table 24-3 NBRCH[3:0]的值和转换序列个数

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

…	…
1110	15
1111	16

要注意的是，即使是在单次转换(one shot)模式下，ADC 也会在启动后转换设置好的转换序列。16 个转换结果寄存器会保存每个序列的转换结果供读取。

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

Table 24-4 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
0000_0000	Input 0	AIN0
0000_0001	Input 1	AIN1
0000_0010	Input 2	AIN2
0000_0011	Input 3	AIN3
0000_0100	Input 4	AIN4
0000_0101	Input 5	AIN5
0000_0110	Input 6	AIN6
0000_0111	Input 7	AIN7
0000_1000	Input 8	AIN8
0000_1001	Input 9	AIN9
0000_1010	Input 10	AIN10
0000_1011	Input 11	AIN11
0000_1100	Input 12	AIN12
0000_1101	Input 13	AIN13
0000_1110	Input 14	AIN14
0000_1111	Input 15	AIN15
0001_0000	Input 16	AIN16
0001_0001	Input 17	AIN17
0001_0010	Input 18	AIN18
0001_0011	Input 19	AIN19
0001_0100	Input 20	AIN20
0001_0101	Input 21	AIN21
0001_0110	Input 22	AIN22
0001_0111	Input 23	AIN23
…	No Input (input floating)	N/A
0100_0000	Input 64	BIN0
0100_0001	Input 65	BIN1
0100_0010	Input 66	BIN2

0100_0011	Input 67	BIN3
0100_0100	Input 68	BIN4
0100_0101	Input 69	BIN5
0100_0110	Input 70	BIN6
0100_0111	Input 71	BIN7
0100_1000	Input 72	BIN8
0100_1001	Input 73	BIN9
0100_1010	Input 74	BIN10
0100_1011	Input 75	BIN11
0100_1100	Input 76	BIN12
0100_1101	Input 77	BIN13
0100_1110	Input 78	BIN14
0100_1111	Input 79	BIN15
0101_0000	Input 80	BIN16
0101_0001	Input 81	BIN17
0101_0010	Input 82	BIN18
0101_0011	Input 83	BIN19
0101_0100	Input 84	BIN20
0101_0101	Input 85	BIN21
0101_0110	Input 86	BIN22
…	No Input (input floating)	N/A
0111_1001	Input 121	OPA0X
0111_1010	Input 122	OPA1X
0111_1011	Input 123	OPA2X
0111_1100	Input 124	OPA3X
0111_1101	Input 125	INTVREF（可通过FVR配置）
0111_1110	Input 126	1/5VDD
0111_1111	Input 127	VSS

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。

注：本产品有两个 ADC 模块，其中一个 ADC 选择 ADC_INAx 输入通道时，另一个 ADC 只能选择 ADC_INBx 输入通道。

24.1.8 转换模式

ADC 可以工作在单次转换模式、连续转换模式或等待转换模式。

24.1.8.1 单次转换模式

将 ADC_MR[MODE]位设 0 为单次转换模式。在这个模式下，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。

24.1.8.2 连续转换模式

将 ADC_MR[MODE]位设 1 为连续转换模式。这个模式下，转换开始后，ADC 不停地对序列进行循环转换，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 会完成当前通道的转换，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通过 ADC_SR[SEQ_INDEX]位查看。

24.1.8.3 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。

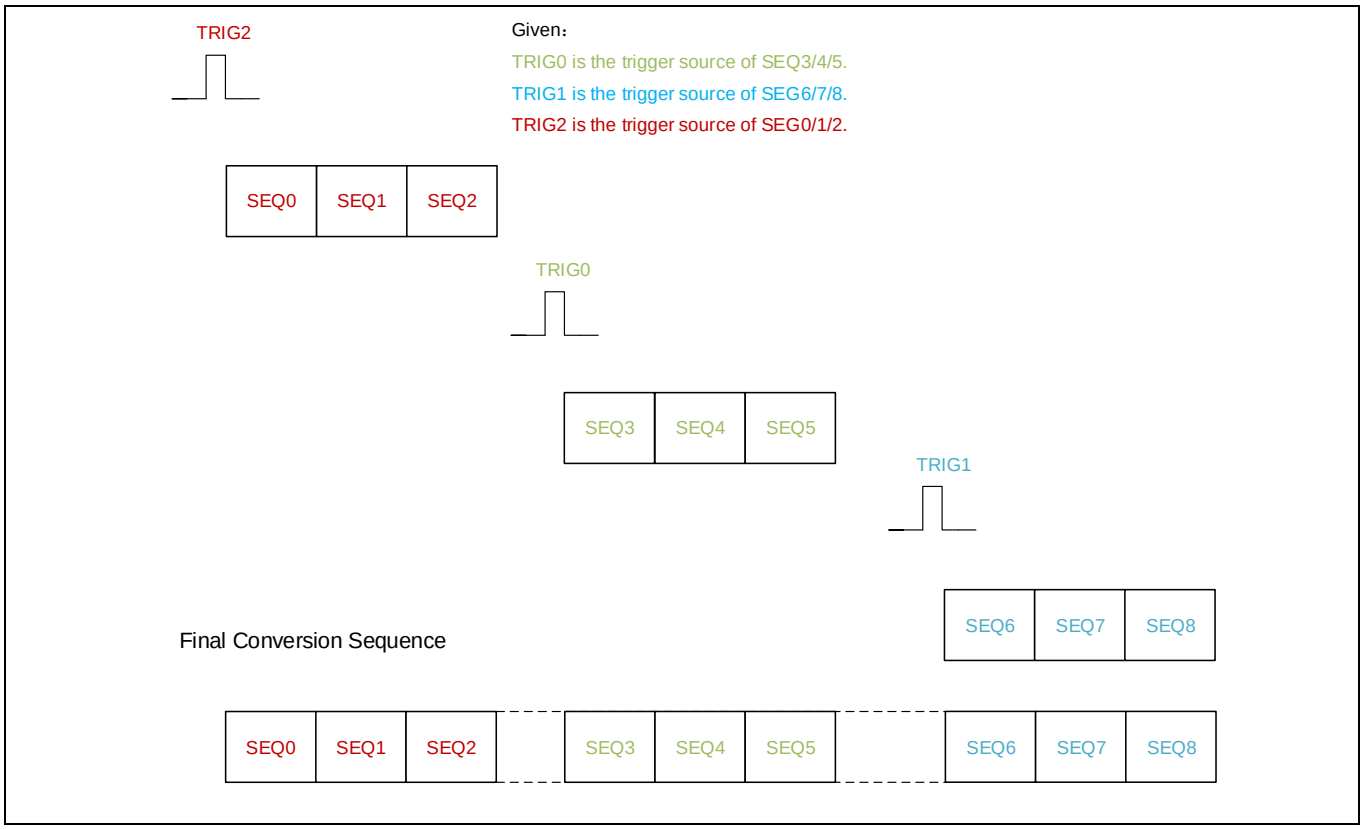


Figure 24-5 等待模式 ADC 工作时序图

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。

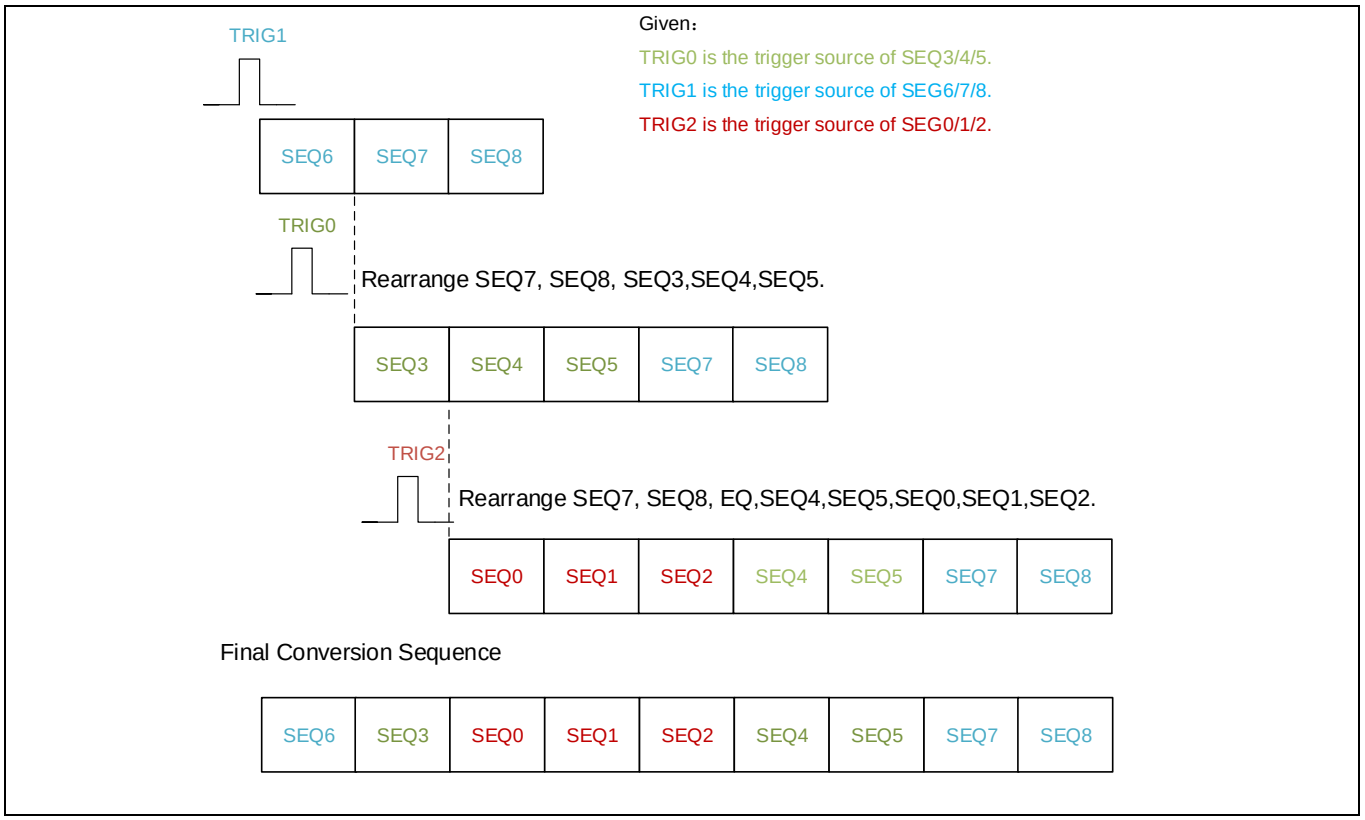


Figure 24-6 等待模式 ADC 工作时序图

24.1.9 重复转换和平均值计算

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

Table 24-5 CV_CNT的值和重复采样次数

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

如果使能计算平均值功能(AVG_CAL=1)，那么 ADC_DRx 寄存器将会保存多次转换的平均值，否则 ADC_DRx 保存的是最后一次转换的结果。平均值的计算方法可以选择，由 AVG_SEL 位决定。如果 AVG_SEL 选择 0，即为

不平均，那么 ADC_DRx 结果寄存器的值为多次转换后的所有结果之和；如果 AVG_SEL 选择 1，那么 ADC_DRx 的值则为多次转换之和除以 2，也即多次转换之和做一次右移操作(前端补 0)。

Table 24-6 AVG_SEL的值和ADC_DRx结果

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

例如，在 ADC_SEQ0 中设置 CV_CNT=3'b011，AVG_CAL=1，AVG_SEL=0，那么该 SEQ0 序列中，ADC 会转换 8 次，假设转换结果为 DATA0~DATA7，在转换结束后，ADC_DR0 的值为 (DATA0+DATA1+…+DATA7)/1；如果 AVG_SEL=1，那么 ADC_DR0 的值为(DATA0+DATA1+…+DATA7)/2；如果 AVG_SEL=3，那么 ADC_DR0 的值为(DATA0+DATA1+…+DATA7)/8；如果设置 AVG_CAL=0，那么 SEQ0 序列转换结束后，ADC_DR0 的值为 DATA7。

24.1.10 转换结果处理

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

ADC_DRMASK 寄存器除了通过软件赋值之外，还能通过配置 ADC_SEQx 的 DRMASK_EN 和 DRMASK_OFF，选择相应的同步事件来置 1（打开）或置 0（关闭）ADC_DRMASK 相应 BIT 位。（通过此项功能可以实现数据的自动保持功能）。

24.1.11 ADC的比较功能

ADC 的比较功能可以让 ADC 在转换结果达到某个设定的值时触发一个中断。将 ADC_CMPx 设定为想要的阈值，一旦转换完成后，ADC 就会将转换结果和这个阈值比较，如果转换结果比 ADC_CMPx 寄存器的值大(电压高)，那么 CMPxH 状态位会被置 1，并且触发相应的中断；如果转换结果比 ADC_CMPx 寄存器的值小(电压低)，那么 CMPxL 状态位会被置 1，并且触发相应的中断。在 ADC_MR 第 30 位 CMP_OS 位为 0 的持续触发模式下，只要 ADC 转换结果比设定的阈值高或者低，就会持续触发相应的 H 或者 L 中断；而在 CMP_OS 位为 1 的单次触发模

式下，只有当 ADC 转换结果从比阈值低变成比阈值高(或者从比阈值高变成比阈值低)的当次转换后会触发中断，后续如果一直保持在比阈值高或者低的状态，那么相应的 H 或者 L 中断不会被触发。

ADC_MR 寄存器中的 NBRCMPx 位用来指定要比较的转换序列号。

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

Table 24-7 NBRCMPx[3:0]的值和需要比较的转换序列号

NBRCMPx[3:0]	需要比较的转换序列号
0000	1
0001	2
0010	3
0011	4
0100	5
0101	6
0110	7
0111	8
1000	9
…	…
1110	15
1111	16

例如，假设：

NBRCH = 0x4,

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

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

ADC_SEQ4.AIN_SEL = 0x2

NBRCMP0 = 0x1, NBRCMP1 = 0x3

ADC_CMP0 = 0x200, ADC_CMP1 = 0x700

(ADC_IMCR) CMP0H = 1, CMP1L = 1

那么 ADC 会进行 5 次转换。

1. ADC 将 SEQ1 (AIN2)的转换结果和 0x200 (ADC_CMP0)比较，如果结果大于 0x200，那么 CMP0H 中断产生。

2. ADC 将 SEQ3 (AIN5)的转换结果和 0x700 (ADC_CMP1)比较，如果结果小于 0x700，那么 CMP1L 中断产生。

24.1.12 ADC转换启动的触发源和触发优先级

ADC转换序列可以选择各种事件作为触发源，如下表格所示：

Table 24-8 TRG_SRC[2:0]的值和选择的触发源

TRG_SRC[5:0]	触发源
0000	无触发
0001	软件触发(ADC_CR中的SWTRG位)
0010	ADC_SYNCIN0触发源 (参考ETCB章节)
0011	ADC_SYNCIN1触发源 (参考ETCB章节)
0100	ADC_SYNCIN2触发源 (参考ETCB章节)
0101	ADC_SYNCIN3触发源 (参考ETCB章节)
0110	ADC_SYNCIN4触发源 (参考ETCB章节)
0111	ADC_SYNCIN5触发源 (参考ETCB章节)
...	...
1111	ADC_SYNCIN14触发源 (参考ETCB章节)

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

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

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

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

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

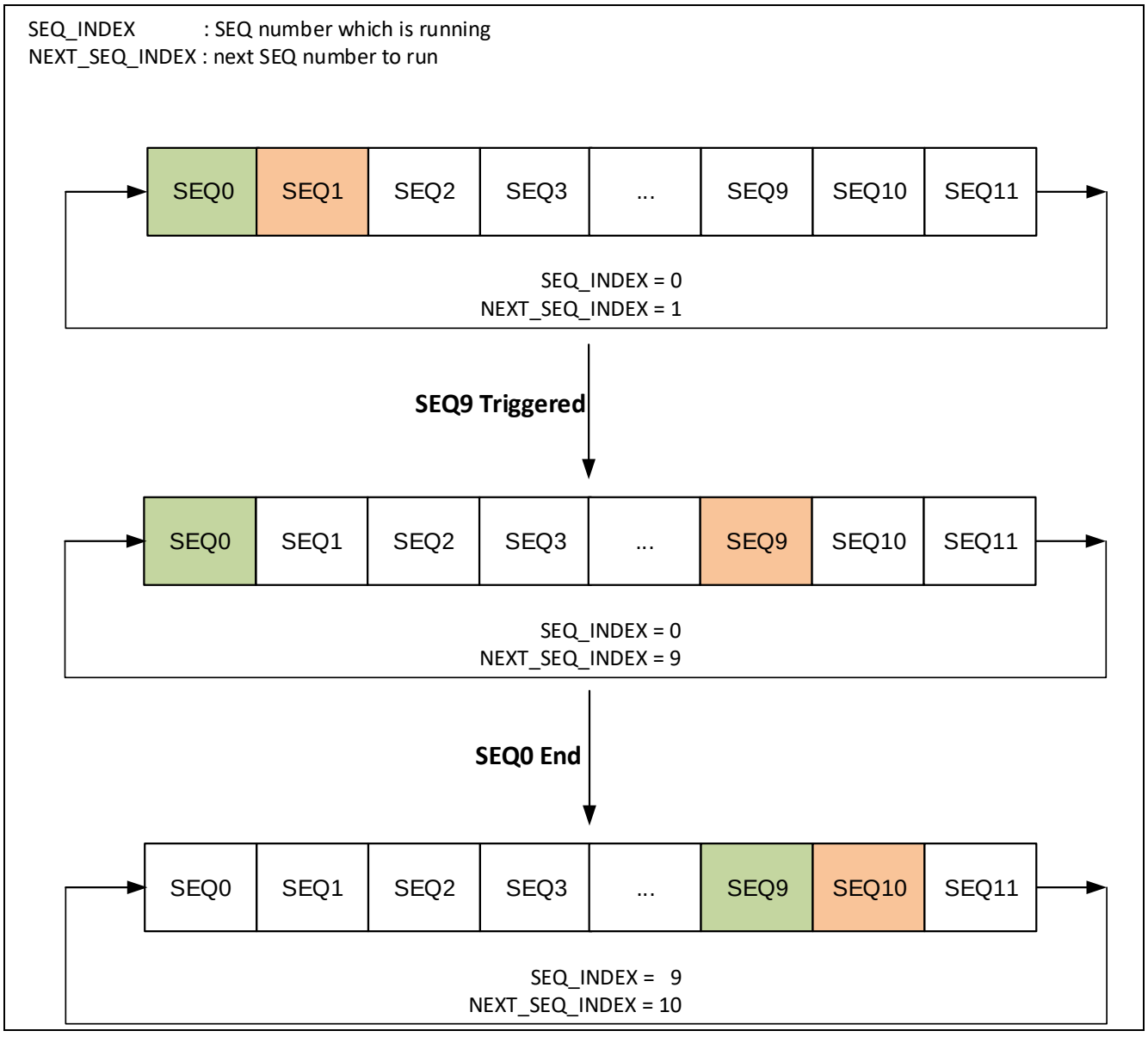


Figure 24-5 触发原理示意图

如果两个序列的触发事件同时产生，那么序列号低(小)的优先级高。如下图，当SEQ0在转换时，SEQ2和SEQ9同时被触发了，那么这两个转换都会被执行，但是序号小的SEQ2会被先转换，然后再继续转换SEQ9，接着再按照顺序继续执行下去。

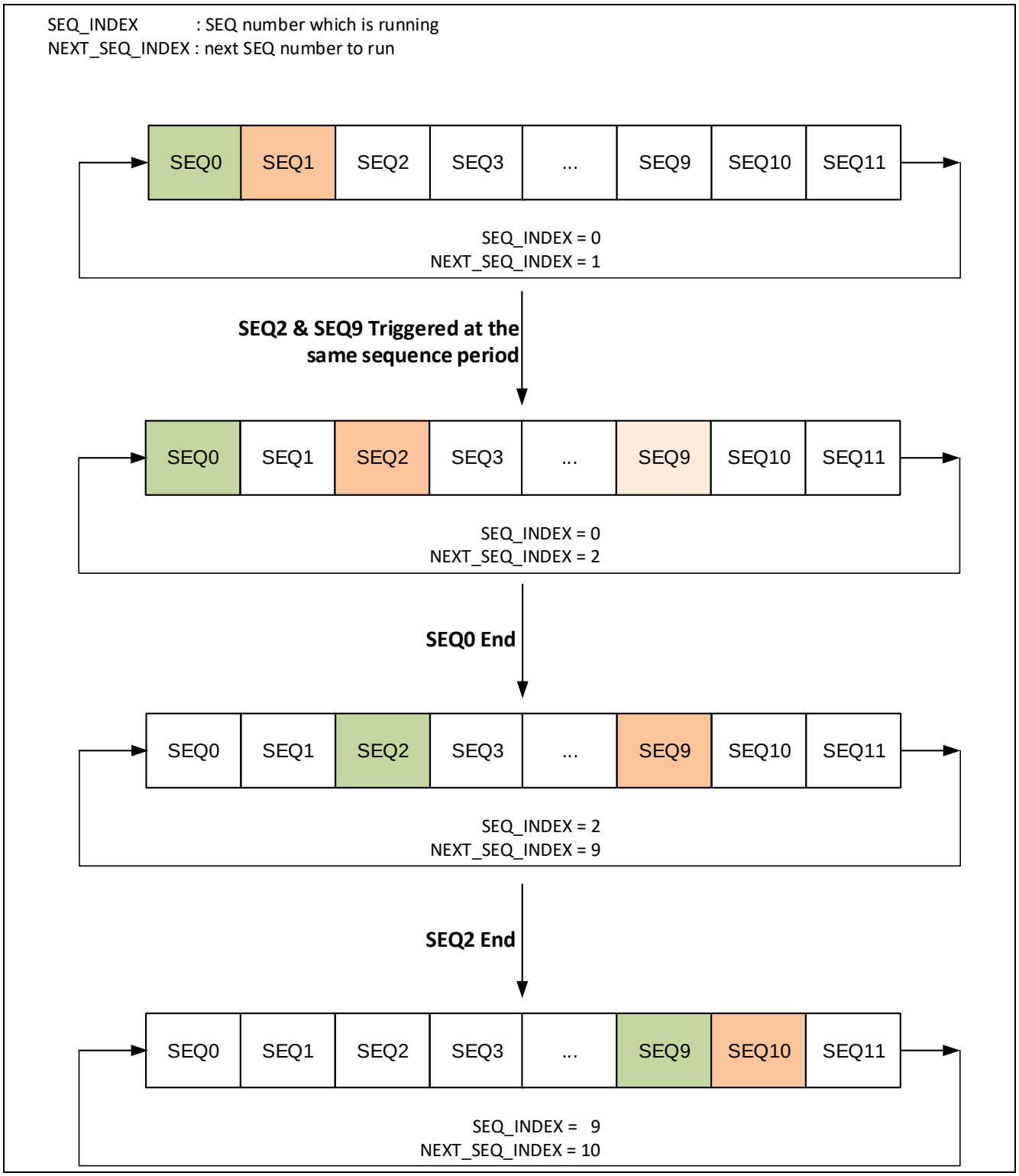


Figure 24-6 同时触发示意图

当某些特殊的转换序列不需要连续转换，而又比普通序列需要有更高的触发转换优先级时，可以使用ADC_PRI寄存器来设置优先级。比ADC_PRI寄存器中设置的值小的序列，会从转换序列中剔除，并且有更高的触发优先级。参考下图的例子，ADC_PRI寄存器设置为0x3，那么SEQ0 ~ SEQ2将不会在转换序列当中，ADC启动时，会直接转换SEQ3。如果在SEQ3转换时，SEQ2和SEQ9同时发生，那么SEQ2会先被转换，之后再转换SEQ9，接着再按照顺序继续转换下去。也就是说SEQ0 ~ SEQ2只有在被触发的时候(需要的时候)才会转换，而不会在转换序列中一直不停的被循环执行。

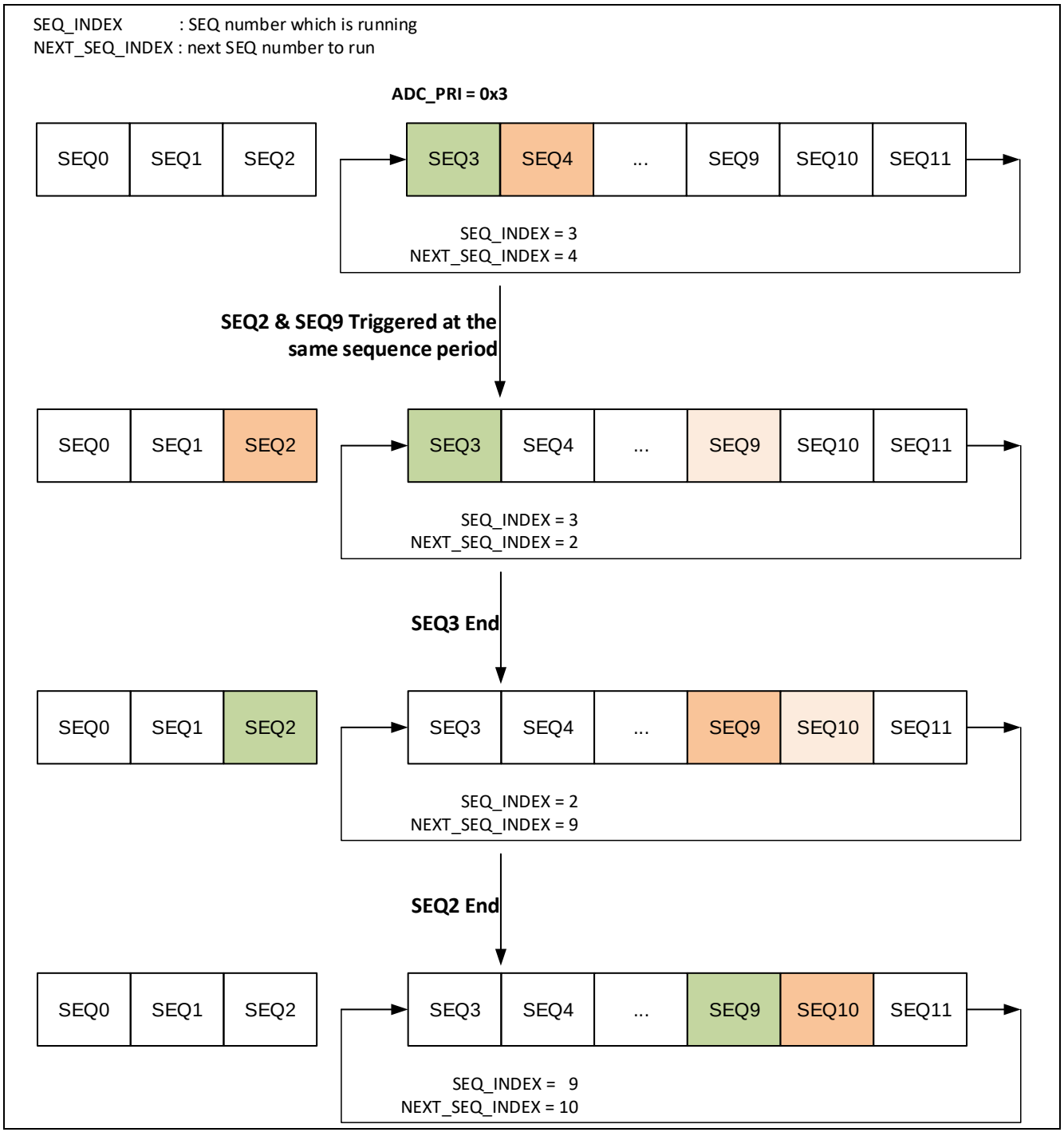


Figure 24-7 触发优先级示意图

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

例如，如果设置PWM触发SEQ4, SEQ10, SEQ11，CMP触发SEQ5，那么当PWM和CMP触发同时发生时，CMP的SEQ5会抢在SEQ10和SEQ11之前转换。所以如果要设置PWM触发三个序列，那么必须设置触发SEQ4, SEQ5,

SEQ6，CMP触发SEQ7，这样当PWM和CMP触发同时发生时，会先转换PWM的连续3个序列SEQ4, SEQ5, SEQ6，然后再转换SEQ7。

24.1.13 功耗管理

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

- 减少模拟功耗：为了降低模拟功耗，CPU 需要禁用 ADC 模块(写 ADC_CR 中的 ADCDIS 位)，让 ADC 处于待机模式。
- 减少数字功耗：为了降低数字功耗，CPU 需要关闭 ADC 时钟(写 ADC_DCR 中的 CLK 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 CLK 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 CLKDIS 位)，然后再打开 ADC 模拟模块(写 ADC_CR 中的 ADCEN 位)。

可以通过读取 ADC_PMSR 和 ADC_SR 中对应位获得功耗管理的状态信息：

Table 24-9 功耗管理的状态位

寄存器中的状态位	状态位为1时	状态位为0时
ADC_PMSR中的CLKEN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

24.1.14 EOC标志 (End of Conversion)

状态寄存器中的 EOC 位表示转换结果寄存器中有新的值。

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

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

24.1.15 Ready标志

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

24.1.16 OVR标志 (转换溢出)

这个标志表示某个转换完成的数据还没有被读取，就被新的数据覆盖了。

OVR 标志可以被 CPU 清除(在状态清除寄存器里写 OVR 位)。

24.1.17 CMPxH/L标志

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

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

24.1.18 SEQ_ENDx标志

这个标志表示序列 x 的转换已经完成。

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

24.1.19 ADC事件输出接口 (ETCB接口)

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

24.1.20 工作流程

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

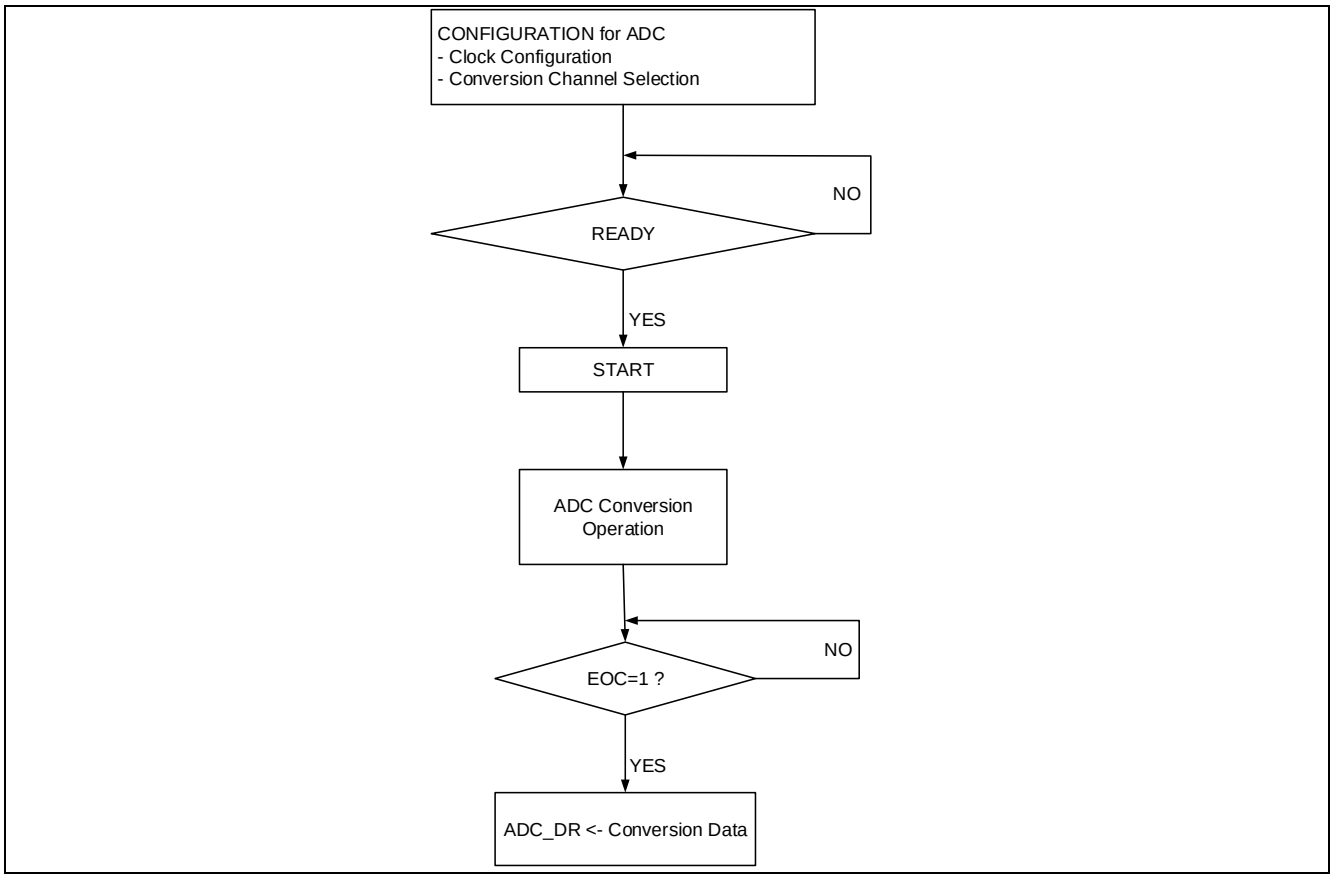


Figure 24-8 ADC工作流程图

24.1.21 转换的软件操作流程

下面描述了在复位后使用 ADC 模块的基本操作流程：

1. 在 ADC_ECR 中使能时钟
2. 在 ADC_MR 中设置 ADC 工作模式。PRLVAL 的值不能让模拟模块的工作时钟频率超过 10MHz。设置单次转换还是连续转换模式。定义转换序列：转换序列个数(NBRCH)和哪些输入通道需要被转换(ADC_SEQx 中的 AIN_SEL)
3. 使能 ADC 模块(ADC_CR 中的 ADC_EN)
4. 等待 ADC_SR 中的 READY 位。只有当这个标志位被置 1 后，ADC 才能正常的开始转换。如果 ADC_IMCR 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 22/20 个时钟周期后，转换完成。12 位/10 位的数字转换结果被存入到 ADC_DR 中，并且 ADC_SR 中的 EOC 位被置 1。如果 EOC 标志已经是 1，那么 OVR 位会被置 1。
8. 然后 CPU 就可以读取 ADC_DR 中的数字值，并且自动清除 EOC。在连续转换模式中，如果 CPU 判断不需要更多的转换了，那么它可以写 STOP 位停止转换。这样 ADC 就会停止工作并且等待下一个开始转换的请求。注意在单次转换模式，ADC 不可以被停止，它会转换完所有的序列后自己停止。

9. 如果 NBRCH 不是 0，那么 ADC 会选择下一个需要转换的模拟输入通道，然后从上面第 6 步重新开始。

10. 如果 CONTCV 是 1，那么 ADC 会从第 5 步重新开始另一个转换序列。

24.1.22 温度传感器

温度传感器是依赖ADC模块实现的。

24.1.22.1 温度传感器的采集输入

如果希望使用内部温度传感器，需要把任意一个ADC输入通道配置成ADCIN_INTVREF，并且FVR_CR中的BUFLVL选择温度传感器（ADC_CR[BUFLVL] = 3）。

24.1.22.2 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。为获得当前温度Tx，需要知道

1） INTVREF TS的当前采样值 （ADC_{TS}）

2） INTVREF 1V的当前采样值 （ADC_{1V}）

3） 出厂前在两个温度（T1，T2）上做的标定值。T1，T2和对应标定值存储于寄存器TS1，TS2，IR1，IR2中。

对INTVREF 1V的采样是为了排除参考电压随温度变化对采样结果造成的影响。对INTVREF TS和INTVREF 1V进行采样时，必须选择相同的参考电压。

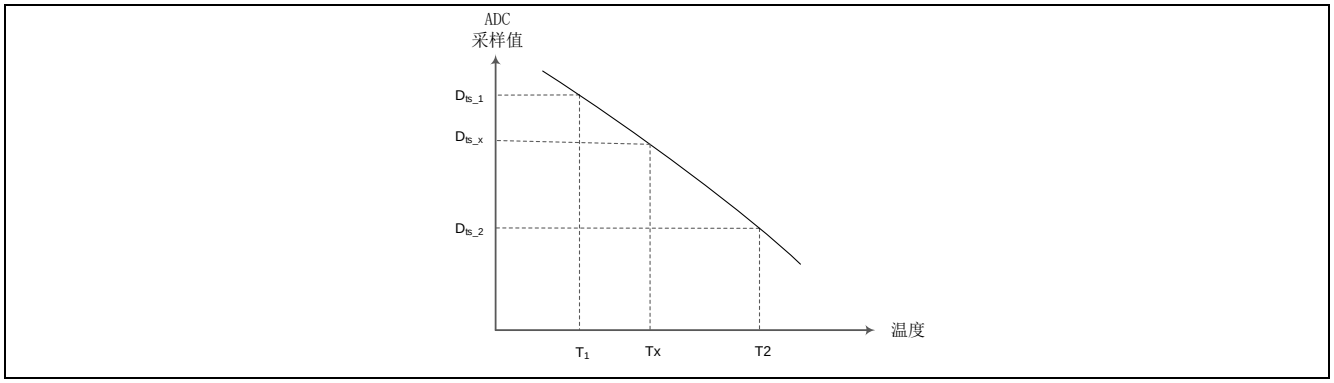


Figure 24-9 ADC采样内部温度传感器

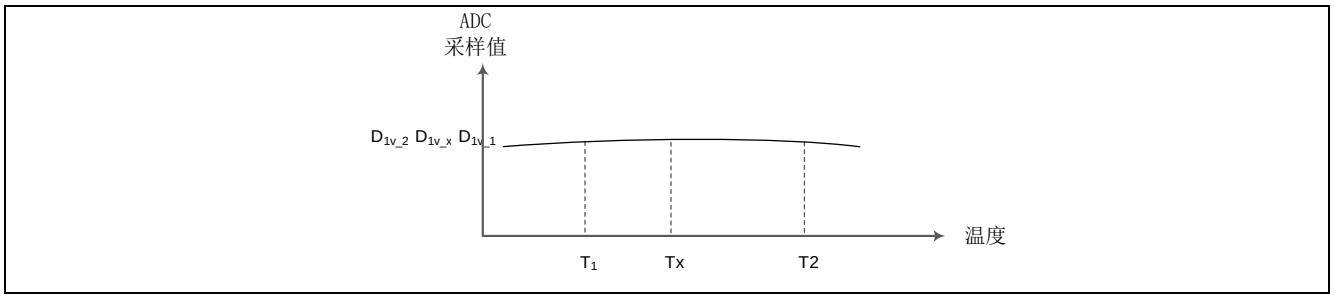


Figure 24-10 ADC采样内部1.0V参考源

24.1.22.3 温度的计算方法

当前温度Tx可通过Figure 24-11的公式计算。

$$T_x = (TS_2[TEMP]) \times 0.0625 - \frac{(TS_2[TEMP] - TS_1[TEMP]) \times 0.0625}{\frac{TS_2[DATA]}{IR_2[DATA]} - \frac{TS_1[DATA]}{IR_1[DATA]}} \times \left(\frac{TS_2[DATA]}{IR_2[DATA]} - \frac{ADC_{TS}}{ADC_{1V}}\right)$$

Figure 24-11 计算Tx 的公式

ADC_{TS}: 采集当前温度传感器 ADC 值

ADC_{1V}: 采集当前温度下 INTVREF 1V 的 ADC 值

24.1.22.4 标定的温度数据格式

出厂标定数据 TS_TS1/2、TS_IR1/2 的数据格式相同。其中 TS_TSx[FLAG]表示标定数据是否有效，TS_TSx[TEMP]表示标定时的温度，TS_TSx[DATA]表示标定时的 ADC 采样值。TS_IRx 保存对应温度下 ADC 采样 INTVREF 1.0V 的电压值。

其中 TS_TSx[TEMP]和温度的对应关系如 Table 24-10所示

Table 24-10 温度数据格式

Temperature（℃）	ADC Result of INTVREF TS（binary）	ADC Result of INTVREF TS（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

24.1.22.5 温度采集基本步骤

- ADC初始化
- ADC输入源选择INTVREF，且ADCC_CR[INTVREF_SEL] = 0x3时，采样到的值为ADC_{TS}
- ADC输入源选择INTVREF，且ADCC_CR[INTVREF_SEL] = 0x2时，采样到的值为ADC_{1V}
- 读取芯片内部标定值
- 代入公式计算

$$T_x = (TS_2[TEMP]) \times 0.0625 - \frac{(TS_2[TEMP] - TS_1[TEMP]) \times 0.0625}{\frac{TS_2[DATA]}{IR_2[DATA]} - \frac{TS_1[DATA]}{IR_1[DATA]}} \times (\frac{TS_2[DATA]}{IR_2[DATA]} - \frac{ADC_{TS}}{ADC_{1V}})$$

24.2 寄存器说明

24.2.1 寄存器表

Base Address of ADC0: 0x40030000
Base Address of ADC1: 0x40031000

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	0x00110800
ADC_CR	0x0010	控制寄存器	0x00000000
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_IMCR	0x0024	中断使能寄存器	0x00000000
ADC_MISR	0x002C	中断状态寄存器	0x00000000
ADC_SEQx	0x0030 - 0x006C	转换序列寄存器x (x=0~15)	0x0000007F
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_EVPS	0x008C	事件触发计数寄存器	0x00000000
ADC_EVSWF	0x0090	软件强制触发事件控制寄存器	0x00000000
ADC_DRx	0x0100- 0x013C	转换结果寄存器x (x=0~15)	0x00000000
ADC_CMP0	0x0140	比较数据0寄存器	0x00000000
ADC_CMP1	0x0144	比较数据1寄存器	0x00000000
ADC_DRMASK	0x0148	禁止转换结果更新寄存器	0x00000000

24.2.2 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													CLK_SEL		RSVD														CLK	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	w	w	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
CLK_SEL	[17:16]	w	ADC模拟模块的时钟源（分频前）的选择： 0: HFOSC 1: PLLQ 2/3: PCLK 若要达到ADC最快速度3MHZ，则要选择PLL频率为96MHZ，使得ADC CLK分频能够为48MHZ
CLK	[1]	W	ADC时钟使能 0: 无效 1: 使能ADC时钟

24.2.3 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																													CLK	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
CLK	[1]	W	ADC时钟禁止 0: 无效 1: 禁止ADC时钟

24.2.4 ADC_PMSR(功耗管理状态寄存器)

Address = Base Address+ 0x0008, Reset Value = 0x00110800

31	30	29	28	27	26	25	24	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		CLKEN	RSVD
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
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
CLKEN	[1]	R	ADC : ADC时钟状态 0: ADC时钟被禁止 1: ADC时钟被使能

24.2.5 ADC_CR(控制寄存器)

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

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

Name	Bit	Type	Description
ACCURACY	[31]	RW	ACCURACY: ADC转换精度选择位 0: 12位 1: 10位
VREF_SEL	[7:6]	RW	VREF: ADC电压参考电源选择 00: 内部VDD 01: 内部VDD 10: FVR 11: INTVREF 注意：使用FVR做参考时，外部需要接100nF的电容。
SWTRG	[5]	W	SWTRG : 软件触发 0: 无效 1: 触发转换序列
STOP	[4]	W	STOP: 在连续转换模式下停止转换 0: 无效 1: 停止连续转换
START	[3]	W	START : 开始转换 0: 无效 1: 开始模数转换，清除EOC标志位 注意：在开始转换前，用户必须保证ADC已经处于准备好转换的状态(ADC_SR中的READY位必须为1)
ADCDIS	[2]	W	ADCDIS : ADC模拟模块禁止 0: 无效 1: 关闭ADC模块(待机模式) 如果ADCEN和ADCDIS都写1，那么ADC会被禁用。
ADCEN	[1]	W	ADCEN : ADC模拟模块使能 0: 无效 1: 使能ADC模块
SWRST	[0]	W	SWRST : ADC软件复位 0: 无效 1: 复位ADC模块 当软件复位发生时，除了ADC_PMSR寄存器以外，其它所有寄存器都会恢复初始值。

24.2.6 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		CMP_OS	RSVD			NBRCMP1				RSVD		NBRCMP0				RSVD		NBRCH				RSVD				PRLVAL					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
RW	RW	RW	R	R	R	RW	RW	RW	RW	R	R	RW	RW	RW	RW	R	R	RW	RW	RW	RW	R	R	R	R	RW	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。 该位初始值为0. 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。 注意：在连续转换模式下，ADC收到停止指令后，仍然会完成当前正在进行的转换，看起来像是多转换了一次。
CMP_OS	[29]	RW	CMP_OS: 一次性比较 0: 每次得到比较结果时，都会产生ADC_CMPx中断 1: 只有转换结果从比ADC_CMPxH小的值变成比它大的值，或者从比ADC_CMPxL大的值变成比它小的值时，才会产生ADC_CMPx中断。
NBRCMP1	[25:22]	RW	NBRCMP1[3:0]: 需要比较的转换序列 当该次转换结果大于或者小于ADC_CMP1寄存器时，将产生一个CMPxH/CMPxL中断
NBRCMP0	[19:16]	RW	NBRCMP0[3:0]: 需要比较的转换序列 当该次转换结果大于或者小于ADC_CMP0寄存器时，将产生一个CMPxH/CMPxL中断
NBRCH	[13:10]	RW	NBRCH[3:0]: 转换序列个数 0000b:1 0001b:2 … 1111b:16 注意： 即使在单次转换模式，如果NBRCH[3:0]的值大于0，ADC也会进行多次转换。
PRLVAL	[5:0]	RW	PRLVAL[5:0]: 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或HFOSC或PLLQ 否则 FADC =（PCLK或HFOSC或PLLQ ）/ PRLVAL

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

24.2.7 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。

24.2.8 ADC_CSR(状态清除寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQ_END15	SEQ_END14	SEQ_END13	SEQ_END12	SEQ_END11	SEQ_END10	SEQ_END9	SEQ_END8	SEQ_END7	SEQ_END6	SEQ_END5	SEQ_END4	SEQ_END3	SEQ_END2	SEQ_END1	SEQ_END0	RSVD								CMP1L	CMP1H	CMP0L	CMP0H	RSVD	OVR	READY	RSVD
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	R	R	R	R	R	R	R	R	W	W	W	W	R	W	W	R

Name	Bit	Type	Description
SEQ_END15~SEQ_END0	[31:16]	W	SEQ_END[x](x=0~15): SEQx序列转换完成中断 0: 无效 1: 清除该中断
CMP1L	[7]	W	CMP1L : 转换结果小于ADC_CMP1中断 0: 无效 1: 清除该中断
CMP1H	[6]	W	CMP1H : 转换结果大于ADC_CMP1中断 0: 无效 1: 清除该中断
CMP0L	[5]	W	CMP0L : 转换结果小于ADC_CMP0中断 0: 无效 1: 清除该中断
CMP0H	[4]	W	CMP0H : 转换结果大于ADC_CMP0中断 0: 无效 1: 清除该中断
OVR	[2]	W	OVR : 转换溢出中断 0: 无效 1: 清除该中断
READY	[1]	W	READY : ADC已准备好可以转换中断 0: 无效 1: 清除该中断

24.2.9 ADC_SR(状态寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQ_END15	SEQ_END14	SEQ_END13	SEQ_END12	SEQ_END11	SEQ_END10	SEQ_END9	SEQ_END8	SEQ_END7	SEQ_END6	SEQ_END5	SEQ_END4	SEQ_END3	SEQ_END2	SEQ_END1	SEQ_END0	RSVD		SEQ_INDEX				CTCVS	ADCENS	CMP1L	CMP1H	CMP0L	CMP0H	RSVD	OVR	READY	EOC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
SEQ_END15~SEQ_END0	[31:16]	R	SEQ_END[x](x=0~15): SEQx序列转换完成中断 0: 该转换序列没完成 1: 该转换序列已完成
SEQ_INDEX	[13:10]	R	SEQ_INDEX : 当前转换序列 该寄存器的值为当前转换的序列号
CTCVS	[9]	R	CTCVS : 连续转换模式状态 0: 单次模式 1: 连续模式
ADCENS	[8]	R	ADCENS : ADC使能状态 0: ADC被禁止 1: ADC 被使能
CMP1L	[7]	R	CMP1L : 比较功能的状态 0: ADC转换的结果比ADC_CMP1大 1: ADC转换的结果比ADC_CMP1小
CMP1H	[6]	R	CMP1H : 比较功能的状态 0: ADC转换的结果比ADC_CMP1小 1: ADC转换的结果比ADC_CMP1大
CMP0L	[5]	R	CMP0L : 比较功能的状态 0: ADC转换的结果比ADC_CMP0大 1: ADC转换的结果比ADC_CMP0小
CMP0H	[4]	R	CMP0H : 比较功能的状态 0: ADC转换的结果比ADC_CMP0小 1: ADC转换的结果比ADC_CMP0大
OVR	[2]	R	OVR : 转换溢出 0: 最后一次读ADC_DR时，ADC没有完成任何转换或者只完成了1次转换 1: 最后一次读ADC_DR时，ADC完成了2次或者2次以上的转换
READY	[1]	R	READY : ADC已准备好可以转换 0: ADC忽略开始或者停止指令：因为它还没有准备好或者转换未结束它还在工作中 1: ADC已经准备好，可以开始一个转换
EOC	[0]	R	EOC : 转换结束

			0: 转换未结束，仍在进行中 1: 转换完成，ADC_DR中的数据有效。当ADC_DR被读取时该位自动清零

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

24.2.10 ADC_IMCR(中断使能寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQ_END15	SEQ_END14	SEQ_END13	SEQ_END12	SEQ_END11	SEQ_END10	SEQ_END9	SEQ_END8	SEQ_END7	SEQ_END6	SEQ_END5	SEQ_END4	SEQ_END3	SEQ_END2	SEQ_END1	SEQ_END0	RSVD								CMP1L	CMP1H	CMP0L	CMP0H	RSVD	OVR	READY	EOC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	RW	RW	RW	RW	R	RW	RW	RW

Name	Bit	Type	Description
SEQ_END15~SEQ_END0	[31:16]	RW	SEQ_END[x](x=0~15): SEQx序列转换完成中断 0: 禁止该中断 1: 使能该中断
CMP1L	[7]	RW	CMP1L : 转换结果低于ADC_CMP1中断 0: 禁止该中断 1: 使能该中断
CMP1H	[6]	RW	CMP1H : 转换结果高于ADC_CMP1中断 0: 禁止该中断 1: 使能该中断
CMP0L	[5]	RW	CMP0L : 转换结果低于ADC_CMP0中断 0: 禁止该中断 1: 使能该中断
CMP0H	[4]	RW	CMP0H : 转换结果高于ADC_CMP0中断 0: 禁止该中断 1: 使能该中断
OVR	[2]	RW	OVR : 转换溢出中断 0: 禁止该中断 1: 使能该中断
READY	[1]	RW	READY : ADC READY中断 0: 禁止该中断 1: 使能该中断
EOC	[0]	RW	EOC : 转换结束中断 0: 禁止该中断 1: 使能该中断
注意： 对于CMPxH 和CMPxL中断，请勿将“H”和”L”中断同时使能。			

24.2.11 ADC_MISR(中断状态寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQ_END15	SEQ_END14	SEQ_END13	SEQ_END12	SEQ_END11	SEQ_END10	SEQ_END9	SEQ_END8	SEQ_END7	SEQ_END6	SEQ_END5	SEQ_END4	SEQ_END3	SEQ_END2	SEQ_END1	SEQ_END0	RSVD								CMP1L	CMP1H	CMP0L	CMP0H	RSVD	OVR	READY	EOC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
SEQ_END15~SEQ_END0	[31:16]	R	SEQ_END[x](x=0~15): SEQx序列转换完成中断 0: 该中断没有发生 1: 该中断发生
CMP1L	[7]	R	CMP1L : 转换结果低于ADC_CMP1中断 0: 该中断没有发生 1: 该中断发生
CMP1H	[6]	R	CMP1H : 转换结果高于ADC_CMP1中断 0: 该中断没有发生 1: 该中断发生
CMP0L	[5]	R	CMP0L : 转换结果低于ADC_CMP0中断 0: 该中断没有发生 1: 该中断发生
CMP0H	[4]	R	CMP0H : 转换结果高于ADC_CMP0中断 0: 该中断没有发生 1: 该中断发生
OVR	[2]	R	OVR : 转换溢出中断 0: 该中断没有发生 1: 该中断发生
READY	[1]	R	READY : ADC READY中断 0: 该中断没有发生 1: 该中断发生
EOC	[0]	R	EOC : 转换结束中断 0: 该中断没有发生 1: 该中断发生

24.2.12 ADC_SEQx(转换序列寄存器x (x=0~15))

Address = Base Address+ 0x0030 - 0x006C, Reset Value = 0x0000007F

31	30	29	28	27	26	25	24	23	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_OFF				DRMASK_EN				SYNC_SRC	TRG_SRC		AVG_SEL				AVG_CAL	CV_CNT				RSVD	AIN_SEL						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
DRMASK_OFF	[27:24]	RW	DRMASK_OFF：当前序列的DRMASK功能关闭同步事件源选择 0000 :无触发 0001 : ADC_SYNCIN0 (ETCB) 0010 : ADC_SYNCIN1 (ETCB) 0011 : ADC_SYNCIN2 (ETCB) 0100 : ADC_SYNCIN3 (ETCB) 0101 : ADC_SYNCIN4 (ETCB) 0110 : ADC_SYNCIN5 (ETCB) 0111 : ADC_SYNCIN6 (ETCB) 1000 : ADC_SYNCIN7 (ETCB) 1001 : ADC_SYNCIN8 (ETCB) 1010 : ADC_SYNCIN9 (ETCB) 1011 : ADC_SYNCIN10 (ETCB) 1100 : ADC_SYNCIN11 (ETCB) 1101 : ADC_SYNCIN12 (ETCB) 1110 : ADC_SYNCIN13 (ETCB) 1111 : ADC_SYNCIN14 (ETCB)
DRMASK_EN	[23:20]	RW	DRMASK_EN：当前序列的DRMASK功能开启同步事件源选择 0000 :无触发 0001 : ADC_SYNCIN0 (ETCB) 0010 : ADC_SYNCIN1 (ETCB) 0011 : ADC_SYNCIN2 (ETCB) 0100 : ADC_SYNCIN3 (ETCB) 0101 : ADC_SYNCIN4 (ETCB) 0110 : ADC_SYNCIN5 (ETCB) 0111 : ADC_SYNCIN6 (ETCB) 1000 : ADC_SYNCIN7 (ETCB) 1001 : ADC_SYNCIN8 (ETCB) 1010 : ADC_SYNCIN9 (ETCB) 1011 : ADC_SYNCIN10 (ETCB) 1100 : ADC_SYNCIN11 (ETCB) 1101 : ADC_SYNCIN12 (ETCB) 1110 : ADC_SYNCIN13 (ETCB) 1111 : ADC_SYNCIN14 (ETCB)

SYNC_SRC	[19]	RW	SYNC_SRC : 同步事件源选择 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)
TRG_SRC	[18:17]	RW	
AVG_SEL	[16:13]	RW	AVG_SEL: 平均系数选择 0000 : 1 (不平均) 0001 : 2 0010 : 4 0011 : 8 0100 : 16 0101 : 32 0110 : 64 0111 : 128 1000 : 256 1001 : 512 Other : 保留
AVG_CAL	[12]	RW	AVG_CAL : 平均值计算 0 : 禁用 1 : 使能 当这位使能时，ADC转换结果寄存器ADC_DRx将保存若干次数转换后的平均值，由CV_CNT和AVG_SEL决定。否则，ADC_DRx将保存最后一次转换的值。
CV_CNT	[11:8]	RW	CV_CNT: 连续重复采样次数 0000 : 1 0001 : 2 0010 : 4 0011 : 8 0100 : 16 0101 : 32 0110 : 64 0111 : 128 1000 : 256 1001 : 512 Other : 保留
AIN_SEL	[6:0]	RW	模拟输入通道选择 AIN_SEL值 输入选择 BIN DEC 0000000 0 INA0 0000001 1 INA1 0000010 2 INA2

			……　　……　　..... 0010111　　23　　INA23 ……　　……　　..... 0100000　　64　　INB0 ……　　……　　..... 1010110　　86　　INB22 ……　　……　　RSVD 1111001　　121　　OPA0X 1111010　　122　　OPA1X 1111011　　123　　OPA2X 1111100　　124　　OPA3X 1111101　　125　　INTVREF（可通过FVR配置） 1111110　　126　　1/5VDD 1111111　　127　　VSS 说明： 1、只使用一个ADC模块时，可以选择任意INAx和INBx作为通道的模拟输入 2、使用两个ADC模块时，不能同时选择同一组模拟输入。如ADC0选择INAx时，ADC1就只能选择INBx。

24.2.13 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

24.2.14 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。			

24.2.15 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。			

24.2.16 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	REARMx(x=0~5)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	OSTMDx(x=0~5)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	SYNCENx(x=0~5)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

24.2.17 ADC_EVTRG(事件触发选择寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD										TRG1OE	TRG0OE	RSVD							TRG1SEL					RSVD			TRG0SEL				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	RW	RW	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	R	R	R	RW	RW	RW	RW	RW

Name	Bit	Type	Description
TRG1OE	[21]	RW	触发输出端口ADC_ TRGOUT1使能 0h：禁止触发输出 1h：允许触发输出
TRG0OE	[20]	RW	触发输出端口ADC_TRGOUT0使能 0h：禁止触发输出 1h：允许触发输出
TRG1SEL	[12:8]	RW	TRGEV0，TRGEV1事件的触发源选择。 00000：无触发输出 00001：EOC事件 00010：READY事件 00011：OVR事件 00100：CMP0H事件 00101：CMP0L事件 00110：CMP1H事件 00111：CMP1L事件 01000：SEQ_END[0]事件 01001：SEQ_END[1]事件 01010：SEQ_END[2]事件 01011：SEQ_END[3]事件 01100：SEQ_END[4]事件 01101：SEQ_END[5]事件 01110：SEQ_END[6]事件 01111：SEQ_END[7]事件 10000：SEQ_END[8]事件 10001：SEQ_END[9]事件 10010：SEQ_END[10]事件 10011：SEQ_END[11]事件 10100：SEQ_END[12]事件 10101：SEQ_END[13]事件 10110：SEQ_END[14]事件 10111：SEQ_END[15]事件
TRG0SEL	[4:0]	RW	TRGEV0，TRGEV1事件的触发源选择。 00000：无触发输出

			00001：EOC事件 00010：READY事件 00011：OVR事件 00100：CMP0H事件 00101：CMP0L事件 00110：CMP1H事件 00111：CMP1L事件 01000：SEQ_END[0]事件 01001：SEQ_END[1]事件 01010：SEQ_END[2]事件 01011：SEQ_END[3]事件 01100：SEQ_END[4]事件 01101：SEQ_END[5]事件 01110：SEQ_END[6]事件 01111：SEQ_END[7]事件 10000：SEQ_END[8]事件 10001：SEQ_END[9]事件 10010：SEQ_END[10]事件 10011：SEQ_END[11]事件 10100：SEQ_END[12]事件 10101：SEQ_END[13]事件 10110：SEQ_END[14]事件 10111：SEQ_END[15]事件

24.2.18 ADC_EVPS(事件触发计数寄存器)

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								EVCNT1				EVCNT0				RSVD								TRGE1PRD				TRGE0PRD			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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
EVCNT1	[23:20]	R	产生TRGEV1触发事件次数
EVCNT0	[19:16]	R	产生TRGEV0触发事件次数
TRGE1PRD	[7:4]	RW	TRGEV0事件计数的周期设置。 当TRGEV1事件发生次数满足周期时，才产生TRGEV1触发事件
TRGE0PRD	[3:0]	RW	TRGEV0事件计数的周期设置。 当TRGEV0事件发生次数满足周期时，才产生TRGEV0触发事件

24.2.19 ADC_EVSWF(软件强制触发事件控制寄存器)

Address = Base Address+ 0x0090, Reset Value = 0x00000000

31	30	29	28	27	26	25	24	23	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

24.2.20 ADC_DRx(转换结果寄存器x (x=0~15))

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

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

Name	Bit	Type	Description
DATA	[20:0]	R	DATA[20:0] : 转换结果 模数转换的结果在转换结束后，锁存在该寄存器，直到下一个转换完成前一直有效可供读取。 当该寄存器被读取后，ADC_SR的EOC位会被自动清零。 注意： 该寄存器的有效位数跟ADC_SEQx的AVG_SEL位有关，选择的平均次数小于转换次数时，该寄存器的位数会多于12位。

- ADC_DR0 Address = Base Address + 0x0100, Reset Value = 0x0000_0000
- ADC_DR1 Address = Base Address + 0x0104, Reset Value = 0x0000_0000
- ……
- ADC_DR15 Address = Base Address + 0x013C, Reset Value = 0x0000_0000

24.2.21 ADC_CMP0(比较数据0寄存器)

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

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

Name	Bit	Type	Description
CMP0	[20:0]	RW	CMP0[20:0] : 比较阈值 模数转换结束后，转换结果会和这个寄存器的值进行比较，根据比较的结果触发相应的中断。

24.2.22 ADC_CMP1(比较数据1寄存器)

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

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

Name	Bit	Type	Description
CMP1	[20:0]	RW	CMP1[20:0] : 比较阈值 模数转换结束后，转换结果会和这个寄存器的值进行比较，根据比较的结果触发相应的中断。

24.2.23 ADC_DRMASK(禁止转换结果更新寄存器)

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

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

Name	Bit	Type	Description
DRMASK	[15:0]	RW	DRMASK : 禁止转换结果更新 如果该位为1 (MASK)，那么对应的ADC_DRx寄存器的转换结果则不会被更新，而是保持MASK之前的值。 当ADC_SEQx中DRMASK_EN/DRMASK_OFF功能被事件触发后，ADC_DRMASK寄存器的值也会相对应的改变。

24.2 寄存器说明

24.2.1 寄存器表

Base Address of TS: 0x00080100

Register	Offset	Description	Reset Value
TS1	0x00	常温下温度传感器信息1	0x00000000
IR1	0x04	常温下内部1V电压参考信息1	0x00000000
TS2	0x08	高温下温度传感器信息2	0x00000000
IR2	0x0c	高温下内部1V电压参考信息2	0x00000000

24.2.2 TS1(常温下温度传感器信息1)

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
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	标定值的温度信息，以补码形式存储 550h 85(℃) ... 010h 1(℃) ... D80h -40(℃) bit27 =0 时，温度为正，实际温度和TEMP之间的关系如下： Temperature = TEMP * 0.0625(℃) bit27 =1 时，温度为负，实际温度和TEMP之间的关系如下： Temperature = (0 - (((TEMP - 1) ^ 0x7FF) & 0x7FF)) * 0.0625(℃)
DATA	[11:0]	R	常温下温度传感器采样值1

24.2.3 IR1(常温下内部1V电压参考信息1)

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
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	标定值的温度信息，以补码形式存储 550h 85(℃) ... 010h 1(℃) ... D80h -40(℃) bit27 =0 时，温度为正，实际温度和TEMP之间的关系如下： Temperature = TEMP * 0.0625(℃) bit27 =1 时，温度为负，实际温度和TEMP之间的关系如下： Temperature = (0 - (((TEMP - 1) ^ 0x7FF) & 0x7FF)) * 0.0625(℃)
DATA	[11:0]	R	常温下内部1V电压采样值1

24.2.4 TS2(高温下温度传感器信息2)

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
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	标定值的温度信息，以补码形式存储 550h 85(℃) ... 010h 1(℃) ... D80h -40(℃) bit27 =0 时，温度为正，实际温度和TEMP之间的关系如下： Temperature = TEMP * 0.0625(℃) bit27 =1 时，温度为负，实际温度和TEMP之间的关系如下： Temperature = (0 - (((TEMP - 1) ^ 0x7FF) & 0x7FF)) * 0.0625(℃)
DATA	[11:0]	R	高温下温度传感器采样值2

24.2.5 IR2(高温下内部1V电压参考信息2)

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
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	标定值的温度信息，以补码形式存储 550h 85(℃) ... 010h 1(℃) ... D80h -40(℃) bit27 =0 时，温度为正，实际温度和TEMP之间的关系如下： Temperature = TEMP * 0.0625(℃) bit27 =1 时，温度为负，实际温度和TEMP之间的关系如下： Temperature = (0 - (((TEMP - 1) ^ 0x7FF) & 0x7FF)) * 0.0625(℃)
DATA	[11:0]	R	高温下内部1V电压采样值2

25 数模转换器 (DAC)

25.1 概述

本章节描述 DAC 控制器的功能，从用户的角度详细说明如何操作 DAC。

25.1.1 主要特性

- 12 位转换数据
- 数据输入分左对齐和右对齐
- 支持多个外部触发源，可以触发转换
- DAC 输出通道 buffer/非 buffer 模式

25.2 功能描述

25.2.1 模块框图

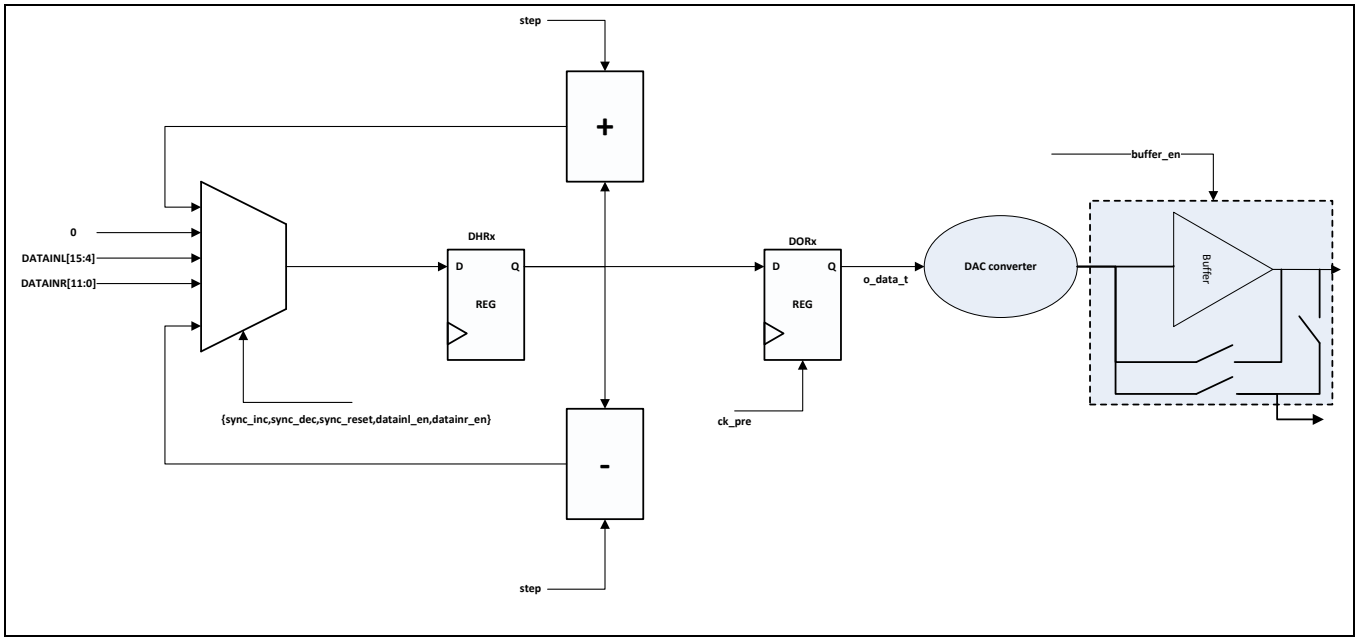


Figure 25-1 模块结构示意图

25.3 基本功能描述

DAC模块是一个12位的数字-模拟转换器。数据可以左对齐或者右对齐。支持三个同步触发事件，分别为上升、下降和复位。

25.3.1 输入数据的格式

根据所选择的配置模式，数据必须写入指定的寄存器。有两种数据格式，左对齐和又对齐，具体如下面两幅图所示。

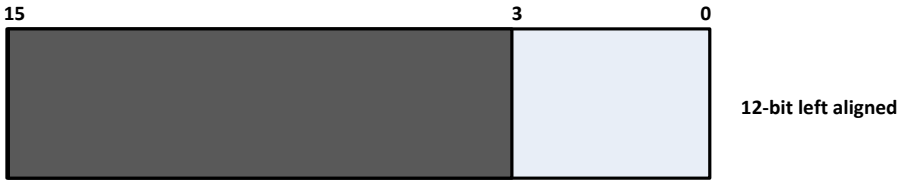


Figure 25-2 12-bit 左对齐

如图1所示，写进寄存器DATAIN_L的数据高12bit有效。经过转换逻辑的选择被转移并存储到相应的DHRx。



Figure 25-3 12-bit 右对齐

如图2所示，写进寄存器DATAIN_R的数据低12bit有效。经过转换逻辑的选择被转移并存储到相应的DHRx。

25.3.2 DAC转换

DHRx寄存器的值是用来产生需要转换的数值的。它的来源如下图所示：

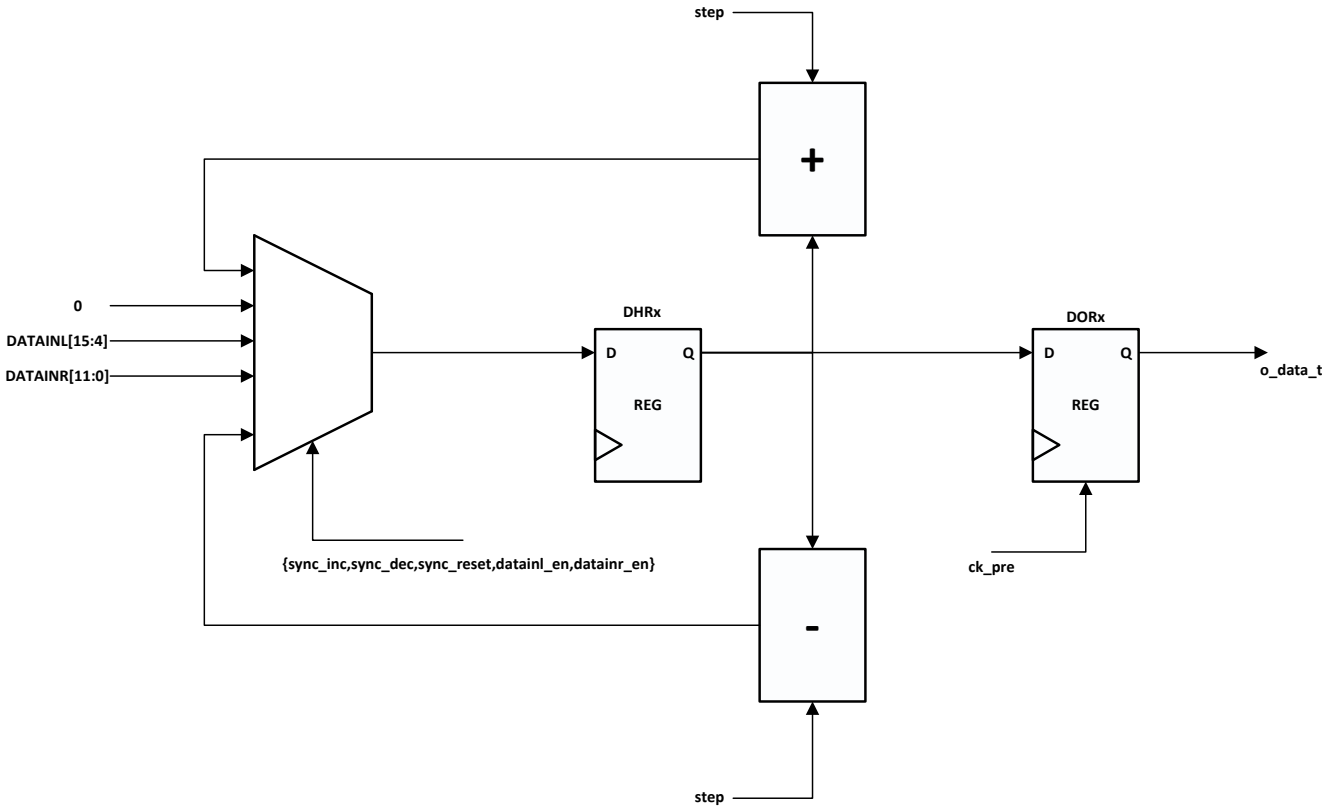


Figure 25-4 转换的逻辑

当DATAIN_L寄存器被写之后，DATAIN_L寄存器的高12bit加载到DHRx寄存器中；

当DATAIN_R寄存器被写之后，DATAIN_R寄存器的低12bit加载到DHRx寄存器中；

当有同步触发事件sync_inc发生时，将DHRx寄存器里的值加上STEP后存入DHRx中；

当有同步触发事件sync_dec发生时，将DHRx寄存器里的值减去STEP后存入DHRx中；

当有同步触发事件sync_reset发生时，将0存入DHRx中；

当同时发生sync_inc、sync_dec和sync_reset中的任何两个时，DHRx寄存器中的值不发生改变；

当DAC正在转换，即DAC_SR寄存器中BUSY位置高时，以上情况将不会影响各寄存器的值，以及当前转换。中断情况请参阅相应寄存器描述。

25.3.3 模拟模块时钟CK

DAC所控制的模拟模块需要一个用来转换的时钟CK，这个CK只要求上升沿满足一定条件，上升沿要稍晚于需要转换数据的变化，保证上升沿能够采到需要转换的数据，两个上升沿的时间差要保证模拟模块能够完成转换。本设计是将hclk进行分频来得到此CK，分频系数由配置DAC_HSCR来得到。具体逻辑如图所示，hclk分频得到dor_clk, 用dor_clk的上升沿将DHRx的值加载到DORx中用于转换，dor_clk延迟一拍生成CK。生成dor_clk有使能位，为DAC_ECR中的DACEN位。

25.3.4 同步触发事件

支持三个同步触发事件，sync_inc、sync_dec和sync_reset。三个同步触发事件均为脉冲信号。

当sync_inc来临时，模块如果处于IDLE状态，则转换数值将会更新为当前转换的数值加上step。当转换完成后可生成相应的脉冲标志信号trg_inc;

当sync_dec来临时，模块如果处于IDLE状态，则转换数值将会更新为当前转换的数值减去step。当转换完成后可生成相应的脉冲标志信号trg_dec;

当sync_reset来临时，模块如果处于IDLE状态，则转换数值将会更新为当前转换的数值加上step。当转换完成后可生成相应的脉冲标志信号trg_reset;

三个同步触发事件均可使能或关闭，通过配置DAT_SYNCR寄存器实现。

25.3.5 DAC 产生触发事件源

DAC 在转换完成后的EOC信号将默认产生DAC0_TRGOUT事件源，可以在ETCB模块中将其作为事件源用来触发目标事件。

25.4 寄存器说明

25.4.1 寄存器表

Base Address of DAC: 0x50200000

Register	Offset	Description	Reset Value
DAC_ECR	0x0000	时钟使能/禁止寄存器	0x00000000
DAC_DACR	0x0004	模拟模块控制寄存器	0x00000000
DAC_HSCR	0x0008	转换时钟分频控制寄存器	0x00000001
DAC_DATAIN_L	0x000C	初始转换左对齐数据寄存器	0x00000000
DAC_DATAIN_R	0x0010	初始转换右对齐数据寄存器	0x00000000
DAC_SYNCR	0x0014	同步控制寄存器	0x00000000
DAC_STEP	0x0018	增量控制寄存器	0x00000000
DAC_SR	0x001C	状态寄存器	0x00000000
DAC_IMCR	0x0020	中断使能控制寄存器	0x00000000
DAC_MISR	0x0024	中断状态寄存器	0x00000000
DAC_ICR	0x0028	中断清除寄存器	0x00000000

25.4.2 DAC_ECR(时钟使能/禁止寄存器)

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
IDCODE																										RSVD					DACEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	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
IDCODE	[31:6]	R	当前DAC模块的版本信息。
DACEN	[0]	RW	DAC的转换使能。 0h：无效。 1h：使能DAC的转换。

25.4.3 DAC_DACR(模拟模块控制寄存器)

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																									BST	RSVD		PWEN	REFSEL	BUFFEN	CLRB
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W	R	R	RW	RW	RW	W

Name	Bit	Type	Description
BST	[6]	W	高功耗 0h：非高功耗。 1h：高功耗。
PWEN	[3]	RW	电源使能 0h：电源关闭。 1h：使能。
REFSEL	[2]	RW	参考电压模式 0h：VDD。 1h：FVR。
BUFFEN	[1]	RW	Buffer的使能。 0h：无效。 1h：buffer使能。 PA4输出只有在BUF使能时才有效，PA8输出只有在BUF无效时才有效
CLRB	[0]	W	DA模拟模块清除控制。 写这位寄存器会清除DA目前的转换结果。

25.4.4 DAC_HSCR(转换时钟分频控制寄存器)

Address = Base Address+ 0x0008, 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																								HSC							
0	0	0	0	0	0	0	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
HSC	[7:0]	RW	时钟分频控制。 CK作为DAC模拟模块的计时时钟和工作时钟。CK的时钟从HCLK分频得到。 CK的频率：FCK = FHCLK / (HSC+1)，HSC最小值为1，即至少2分频。默认值为1

25.4.5 DAC_DATAIN_L(初始转换左对齐数据寄存器)

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
RSD																DATAINL															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
RSD	[31:16]	R	
DATAINL	[15:0]	RW	初始的转换数据，左对齐高12bits有效。

25.4.6 DAC_DATAIN_R(初始转换右对齐数据寄存器)

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
RSD																DATAINR															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
RSD	[31:16]	R	
DATAINR	[15:0]	RW	初始的转换数据，右对齐低12bits有效。

25.4.7 DAC_SYNCR(同步控制寄存器)

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																													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	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
SYNCEN2~SYNCEN0	[2:0]	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：增加STP SYNCIN1：减少STP SYNCIN2：rest到0

25.4.8 DAC_STEP(增量控制寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																				STEP											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
STEP	[11:0]	RW	外部同步触发时，每次转换增加或减少的值。

25.4.9 DAC_SR(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																BUSY
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
BUSY	[0]	R	正在转换，此时不允许写入DATAIN或产生触发事件，否则相应中断状态位会置位，如果使能了相应中断使能也会产生中断。

25.4.10 DAC_IMCR(中断使能控制寄存器)

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																													SYNCERR	WRERR	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	RW	RW	RW

Name	Bit	Type	Description
SYNCERR	[2]	RW	SYNCERR ：转换未结束时事件触发发生
WRERR	[1]	RW	WRERR：转换未结束时数据被更新
EOC	[0]	RW	EOC：转换结束中断
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			

25.4.11 DAC_MISR(中断状态寄存器)

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																													SYNCERREV	WRERREV	EOCEV
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	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
SYNCERREV	[2]	R	转换未结束时事件触发发生
WRERREV	[1]	R	转换未结束时数据被更新
EOCEV	[0]	R	转换结束
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			

25.4.12 DAC_ICR(中断清除寄存器)

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																													SYNCERREV	WRERREV	EOCEV
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	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
SYNCERREV	[2]	W	清除SYNCERREV中断状态位
WRERREV	[1]	W	清除WRERREV中断状态位
EOCEV	[0]	W	清除EOCEV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’			

26 固定电压源（FVR）

26.1 概述

芯片内部集成了一个固定电压源（FVR），可用提供固定的电压源作为参考电压。在使能 FVR_CR 寄存器中的 FVREN 位打开电压基准源模块，电压基准源模块可以提供四个不会随芯片电源 VDD 变化的固定电压 2.048 V、3.072 V 或 4.096V，可以通过 FVR_CR 寄存器中的 FVRLVL 位进行选择。

26.1.1 特性

- 提供四个不随芯片电源 VDD 变化的固定电压，固定电压可选
  - 2.048 V、3.072 V 或4.096V
- 提供比较器参考电平选择
  - AVDD 或 FVR
- 提供放大器共模电压选择
  - 1/2AVDD 或1/2FVR
- 提供 INTERVREF 电压选择
  - 外部管脚输入（PD4）、1.0V 固定电压、温敏电压

26.1.2 管脚描述

Table 26-1 FVR 管脚描述

管脚名称	功能	I/O类型	有效状态	备注
PD4	INTERVREF模拟电压信号输入	模拟	-	-

注：使用 PD4作为 INTERVREF 模拟电压信号输入管脚时，需要将 PD4配置为 INPUT 模式，并将 FVR_CR 寄存器中的 BUFIN 使能打开。

26.2 功能描述

26.2.1 固定电压源功能模块

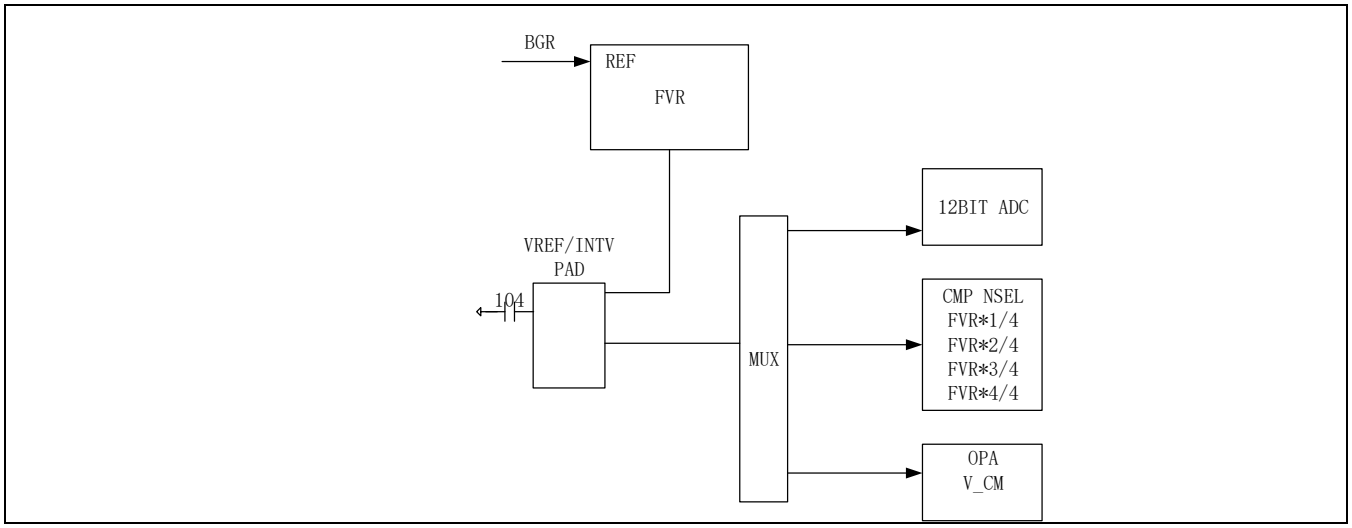


Figure 26-1 固定电压源框图

26.2.2 固定电压源功能

通过FVRLVL选择固定电压源作为FVR电平，通过CMPREF选择传输给比较器的参考电平源，通过OPACM选择传输给OPA的共模参考电压，通过BUFLVL选择INTERVREF的电平，如果选择管脚模拟输入，则通过BUFIN选择是否开启模拟输入管脚。

26.3 寄存器说明

26.3.1 寄存器表

Base Address of FVR: 0x40032000

Register	Offset	Description	Reset Value
FVR_CR	0x00	固定电压源控制寄存器	0x00000000

26.3.2 FVR_CR(固定电压源控制寄存器)

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
BUFKEY								BUFIN	RSVD	BUFLVL		RSVD			BUFEN	RSVD			OPACM	RSVD			CMPREF	RSVD		FVRLVL		RSVD			FVREN
0	0	0	0	0	0	0	0	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	R	RW	RW	R	R	R	RW	R	R	R	RW	R	R	R	RW	R	R	R/W	R/W	R	R	R	R/W

Name	Bit	Type	Description
BUFKEY	[31:24]	RW	BUF模块KEY控制。 其他值： 无效 0x5a ： 解锁BUF模块的KEY控制
BUFIN	[23]	RW	PAD输入到BUF的使能控制 (需使能BUFKEY)。 0： 禁止PAD (PD4)输入到BUF 1： 使能PAD (PD4)输入到BUF
BUFLVL	[21:20]	RW	BUF输入电平选择。(INTVREF) 0: 模拟输入(PD4) 1：1.0V 3：TS
BUFEN	[16]	RW	BUF模块的使能控制。(INTVREF) 0： 禁止BUF模块 1： 使能BUF模块
OPACM	[12]	RW	OPA模块的使能控制。 0： 1/2 VDD 1： 1/2 FVR
CMPREF	[8]	RW	比较器参考电平选择 0: AVDD 1: FVR
FVRLVL	[5:4]	R/W	固定电压源电平选择。 0： 1：2.048V 2：3.072V 3：4.096V
FVREN	[0]	R/W	FVR模块的使能控制。 0： 禁止FVR模块 1： 使能FVR模块

27 模拟比较器（CMP）

27.1 概述

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

27.1.1 特性

- 支持最大3个独立模拟比较器。
- 每个模拟比较器支持正负端外部输入，比较结果直接 IO 输出。
  - 比较器的正向端可以选择外部输入。
  - 比较器的负向端可以选择外部输入，或者内部参考电压。
- 可配置的比较器输出极性。
- 可配置的输出迟滞特性。
- 可配置的比较器输出数字滤波选择。
- 中断触发模式选择（边沿检测）。
- 支持中断硬件自动触发 ADC 转换。

27.1.2 管脚描述

Table 27-1 CMP 管脚描述

管脚名称	功能	I/O类型	有效状态	备注
CPINxP (x=0~15)	比较器模拟输入正向端通道	A	-	-
CPINxN(x=0~10)	比较器模拟输入负向端通道	A	-	-
CPx_OUT(x=0~2)	比较器输出输出	O	-	-

27.2 功能描述

27.2.1 比较器功能

处理器中内嵌 3 个一致的模拟比较器，其数字输出和模拟输入的关系如下图所示。当 VIN+的电压低于 VIN-的时候，比较器的数字输出为低电平。反之，则为高电平。

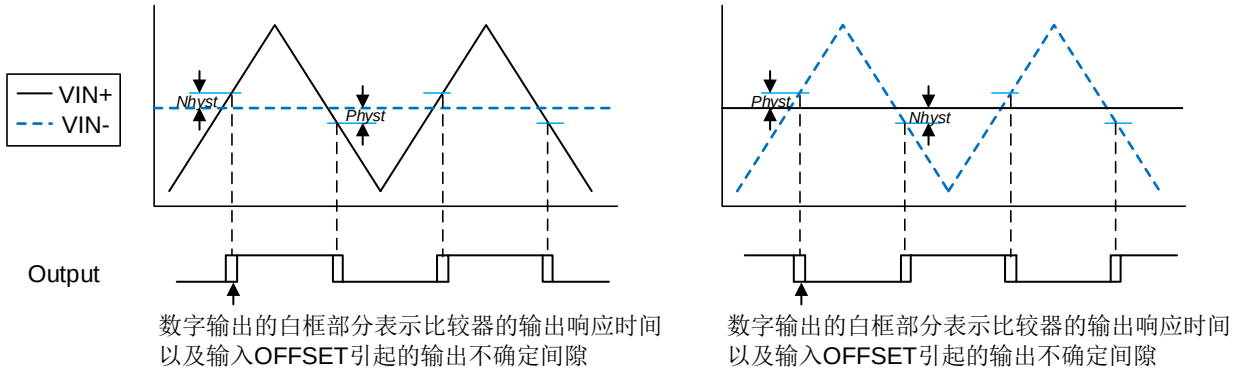


Figure 27-1 单一比较器示意图

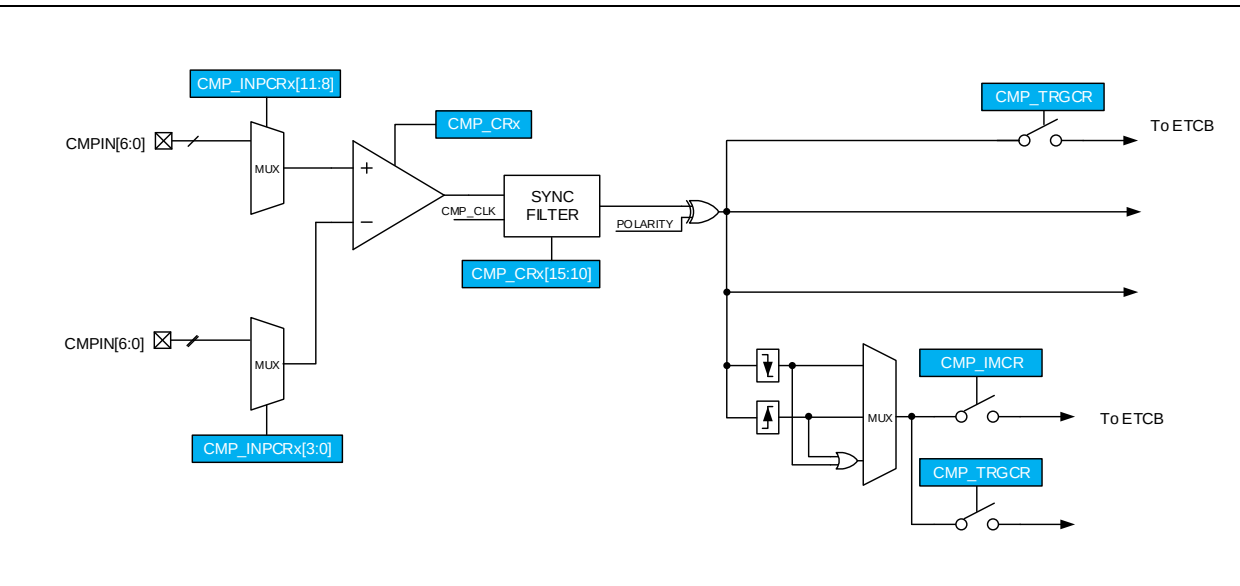


Figure 27-2 比较器结构示意图

NOTE:

1) 所有比较器的模拟输入通道都是共享的，每个通道不是单一对应某一个比较器的模拟输入端。
2) 比较器的控制时钟为比较器模块 PCLK 的时钟频率。

27.2.2 比较器控制

每一个模拟比较器都有独立的控制寄存器来进行模式配置：CMP_CRx。控制寄存器中，可以设置比较器的使能，输出极性，响应速度以及相关的输入输出特性。对于比较器的输入模拟通道的选择，可以通过相对应的通道选择控制寄存器进行设置：CMP_INPCRx。比较器有两个模拟输入端，正向和负向输入端。正向模拟输入端和负向模拟输入均可以支持外部模拟信号的直接输入。

27.2.3 比较器响应时间

在比较器的输入端发生改变时，比如通道切换、参考源变化、或者输入变化等，都会使得比较器的输出在一定时间内的不确定性输出。这个时间长度为比较器的响应时间。此响应时间包含除比较器自身内部电路引起的延时以外，还包括参考电压源的稳定时间。因此在应用设计时，确定响应时间时，需要充分考虑两种延时的影响。为提高比较器的响应速度，比较器内部具有加速功能。在加速模式下，比较器的功耗会显著增大。具体参考电器特性章节。

27.2.4 输出极性控制

比较器的支持数字滤波和极性选择功能。对比较器的输出取反输出，相当于交换比较器的正负向输入端口。比较器的输出反向可以通过控制寄存器中的 POLARITY 位进行设置。比较器的输出状态，可以通过控制寄存器的 CMPOUT 获得。

Table 27-2 比较器输出状态和输入关系

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

27.2.5 数字滤波器

比较器的输出端可以选择经过两个数字滤波后输出。两个滤波器有独立的使能控制。

滤波器 1 是一个积分滤波器。该滤波器的采样频率可以通过 CMP_DFCR1[DIVM]和 CMP_DFCR1[DIVN]设置。FLT1_CK = PCLK * 2^{DIVN}/(DIVM+1)。积分深度由 CMP_DFCR1[DEPTH]设置。工作原理如下：当滤波器采样到‘1’时，积分计数器加 1，反之，采样到‘0’时，积分计数器减 1。滤波器的输出取决于积分计数器的值。当积分计数器值≥CMP_DFCR1[DEPTH]时，输出高；当积分计数器值= 0 时，输出低。

滤波器 2 是一个去抖滤波器。该滤波器的采样频率可以通过 CMP_DFCR2[DIVM]和 CMP_DFCR2[DIVN]设置。FLT2_CK = PCLK * 2^{DIVN}/(DIVM+1). 去抖深度由 CMP_DFCR2[DEPTH]设置。工作原理如下：只有当采样到的电平连续 CMP_DFCR2[DEPTH]次保持一致，滤波器才会改变输出。

27.2.6 比较器中断

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

27.3 寄存器说明

27.3.1 寄存器表

Base Address of CMP0: 0x400D0000
Base Address of CMP1: 0x400D1000
Base Address of CMP2: 0x400D2000

Register	Offset	Description	Reset Value
CMP_CEDR	0x00	ID和时钟使能控制寄存器	0x00035000
CMP_CR	0x04	比较器的控制寄存器	0x00000000
CMP_DFCR1	0x08	数字滤波器1控制寄存器	0x00000000
CMP_DFCR2	0x0C	数字滤波器2控制寄存器	0x00000000
CMP_WFCR	0x10	窗口滤波器控制寄存器	0x00000000
CMP_INPCR	0x14	比较器输入控制寄存器	0x00000000
CMP_RISR	0x18	原始中断状态寄存器	0x00000000
CMP_IMCR	0x1C	中断使能禁止控制寄存器	0x00000000
CMP_MISR	0x20	中断状态寄存器	0x00000000
CMP_ICR	0x24	中断标志清除寄存器	0x00000000

27.3.2 CMP_CEDR(ID和时钟使能控制寄存器)

Address = Base Address+ 0x00, Reset Value = 0x00035000

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

Name	Bit	Type	Description
IDCODE/ID_KEY	[31:8]	R	ID Code寄存器。 这个区域保存了相应IP的IDCODE。
SWRST	[7]	W	软件复位。 0：没有效果 1：执行软件复位操作
CLKEN	[0]	RW	CMP的时钟使能/禁止控制位。 0： 停止控制时钟 1： 使能控制时钟

27.3.3 CMP_CR(比较器的控制寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD					CPOUT2	CPOUT1	CPOUT0	CPOSEL	RSVD		NHYSTSEL			RSVD0	RSVD1	WFOSET		WFALIGN	WFLTEN	DFLT2EN	DFLT1EN	EVE_SEL		SYNCOE	POLARITY	RSVD		PHYSTPOL		CMPEN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	RW	R	R	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW

Name	Bit	Type	Description
CPOUT2	[26]	R	比较器2输出状态（滤波后输出点状态）
CPOUT1	[25]	R	比较器1输出状态（滤波后输出点状态）
CPOUT0	[24]	R	比较器0输出状态（滤波后输出点状态）
CPOSEL	[23]	RW	CMP_OUT管脚上输出信号选择。 0h：滤波前信号直接输出 1h：滤波后信号输出
NHYSTSEL	[20:18]	RW	比较器N端上升沿输入迟滞。 0h：0mV 1h：10mV 2h：15mV 3h：25mV 4h：35mV 5h：45mV 6h：55mV 7h：65mV
RSVD0	[17]	R	
RSVD1	[16]	RW	复位为0，正常工作建议设置为1
WFOSET	[15:14]	RW	窗口滤波设置输出初始化。 0h：无效 1h：设置滤波器输出为高电平 2h：设置滤波器输出为低电平 3h：窗口只有在窗口延时期（长度由WFCR[DCNT]决定）无效
WFALIGN	[13]	RW	窗口滤波触发事件对齐设置。 0h：新触发发生时，开始新的窗口计数，不允许前一个窗口周期跨越当前周期，若前一个窗口仍有效则截断窗口 1h：新触发发生时，开始新的窗口计数，允许窗口跨越周期
WFLTEN	[12]	RW	窗口滤波设置。 0h：禁止滤波器 1h：使能滤波器
DFLT2EN	[11]	RW	输出数字滤波器设置。 0h：跳过数字滤波器

			1h：使能数字滤波器
DFLT1EN	[10]	RW	输出数字滤波器设置。 0h：跳过数字滤波器 1h：使能数字滤波器
EVE_SEL	[9:8]	RW	事件触发边沿选择。 0：下降沿 1：上升沿 2：下降沿和上升沿 3：上升沿
SYNCOE	[7]	RW	使能/禁止模拟比较器同步触发的输出。 0：禁止触发事件输出到ETCB 1：使能触发事件输出到ETCB
POLARITY	[6]	RW	比较器输出极性选择。 0：输出不反向 1：输出反向
PHYSTPOL	[3:1]	RW	比较器P端上升沿输入迟滞。 0h：0mV 1h：10mV 2h：15mV 3h：25mV 4h：35mV 5h：45mV 6h：55mV 7h：65mV
CMPEN	[0]	RW	CMP的时钟使能/禁止控制位。 0： 停止控制时钟 1： 使能控制时钟

27.3.4 CMP_DFCR1(数字滤波器1控制寄存器)

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

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

Name	Bit	Type	Description
DIVN	[20:16]	RW	数字滤波器采样时钟FLT1_CK分频系数N
DIVM	[15:8]	RW	数字滤波器采样时钟FLT1_CK分频系数M
DEPTH	[2:0]	RW	数字滤波器1积分深度设置。滤波器输入为高时，积分计数器增加1；输入低时，积分计数器减1。当前滤波器输出为低时，只有当积分计数器大于等于DEPTH设置时，滤波器输出才会翻转；当前滤波器输出为高时，只有积分计数器等于零时，滤波器输出才会翻转。滤波器的采样时钟频率为FLT1_CK的频率。 000b：N = 8 001b：N = 16 010b：N = 32 011b：N = 64 100b：N = 128 101b：N = 256 110b：N = 512 111b：N = 1024

27.3.5 CMP_DFCR2(数字滤波器2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											DIVN					DIVM								RSVD					DEPTH		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	RW	RW	RW

Name	Bit	Type	Description
DIVN	[20:16]	RW	数字滤波器采样时钟FLT2_CK分频系数N
DIVM	[15:8]	RW	数字滤波器采样时钟FLT2_CK分频系数M
DEPTH	[2:0]	RW	数字滤波器2去抖深度设置。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为FLT2_CK的频率。 000b：N = 16 001b：N = 32 010b：N = 64 011b：N = 128 100b：N = 256 101b：N = 512 110b：N = 1024 111b：N = 2048

27.3.6 CMP_WFCR(窗口滤波器控制寄存器)

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								DCNT								CLKDIV						WCNT									
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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
DCNT	[23:16]	RW	设置捕捉窗口延时计数器。在触发后，内部8位递减计数器根据CLKDIV的设置频率计数。延时的时间为(DCNT+1) x Twcnt。 注意：当DCNT为零时，延时窗口被关闭，而不是1个Twcnt。
CLKDIV	[15:10]	RW	设置捕捉滤波器的频率Fwcnt。 Value DIV Value DIV 0 不分频 28 Div208 1 Div2 29 Div224 2 Div3 30 Div240 3 Div4 31 Div256 4 Div5 32 Div288 …… …… 33 Div320 15 Div16 34 Div352 16 Div24 35 Div384 17 Div32 36 Div416 18 Div40 37 Div448 19 Div48 38 Div480 20 Div56 39 Div512 21 Div64 40 Div640 22 Div72 41 Div720 23 Div128 42 Div1024 24 Div144 43 Div2048 25 Div160 Other 不分频 26 Div176 27 Div192 Fwcnt的分频是基于比较器模块PCLK时钟的分频。
WCNT	[9:0]	RW	设置捕捉窗口宽度计数器。在触发后，当延时计数器计数完成后，内部10位计数器根据CLKDIV的设置频率计数。窗口的宽度为：(WCNT+1)x Twcnt

27.3.7 CMP_INPCR(比较器输入控制寄存器)

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

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

Name	Bit	Type	Description
FLTOFF	[16]	RW	比较器负向（VIN-）输入模拟滤波选择() 0h：使能滤波(负端是比较稳定的参考信号) 1h：关闭滤波(负端是持续变化的输入信号)
PSEL	[11:8]	RW	比较器正向（VIN+）输入选择。 0h：CP0 1h：CP1 …… 14h：CP14 15h：CP15
NSEL	[3:0]	RW	比较器负向（VIN-）输入选择。 0h：CP0 1h：CP1 …… 10h：CP10 11h：AVDD/4 or FVR/4 12h：2*AVDD/4 or 2*FVR/4 13h：3*AVDD/4 or 3*FVR/4 14h：4*AVDD/4 or 4*FVR/4 15h：DAC输出

27.3.8 CMP_RISR(原始中断状态寄存器)

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																													EDGEDET2	EDGEDET1	EDGEDET0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

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

27.3.9 CMP_IMCR(中断使能禁止控制寄存器)

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																															EDGEDET
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	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
EDGEDET	[0]	RW	CMP的输出边沿检测中断使能控制。 0：禁止中断发生 1：使能中断发生

27.3.10 CMP_MISR(中断状态寄存器)

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																													EDGEDET2	EDGEDET1	EDGEDET0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

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

27.3.11 CMP_ICR(中断标志清除寄存器)

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																															EDGEDET
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	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
EDGEDET	[0]	W	CMP的输出边沿检测中断状态清除位。（只有写入时有效） 0：无效果 1：清除CMP0中断标志位

28 模拟运算放大器（OP-AMP）

28.1 概述

芯片内部集成了四个运算放大器，可用于特定模拟信号的处理。通过内部寄存器设置，可以配置运算放大器工作在不同的工作方式。当运算放大器配置为外部增益模式时，需要增加外部反馈电路才能使得运算放大器正常工作；如果采内部增益模式，则可以选择内部增益控制。

28.1.1 特性

- 支持最大4个独立运算放大器。
- 每个运算放大器可以支持外部增益模式，或者内部增益模式。
- 可配置的内部增益放大系数
  - 单端输入模式：4/6/8/10/15/20/30/40。
  - 差分输入模式：1/2/3/5/7/9/11/19。

28.1.2 管脚描述

Table 28-1 OPA 管脚描述

管脚名称	功能	I/O类型	有效状态	备注
OPA0X	运算放大器0的输出通道	A	-	-
OPA0P	运算放大器0的正向输入通道	A	-	-
OPA0N	运算放大器0的负向输入通道	A	-	-
OPA1X	运算放大器1的输出通道	A	-	-
OPA1P	运算放大器1的正向输入通道	A	-	-
OPA1N	运算放大器1的负向输入通道	A	-	-
OPA2X	运算放大器2的输出通道	A	-	-
OPA2P	运算放大器2的正向输入通道	A	-	-
OPA2N	运算放大器2的负向输入通道	A	-	-
OPA3X	运算放大器3的输出通道	A	-	-
OPA3P	运算放大器3的正向输入通道	A	-	-
OPA3N	运算放大器3的负向输入通道	A	-	-

28.2 功能描述

28.2.1 运算放大器功能模块

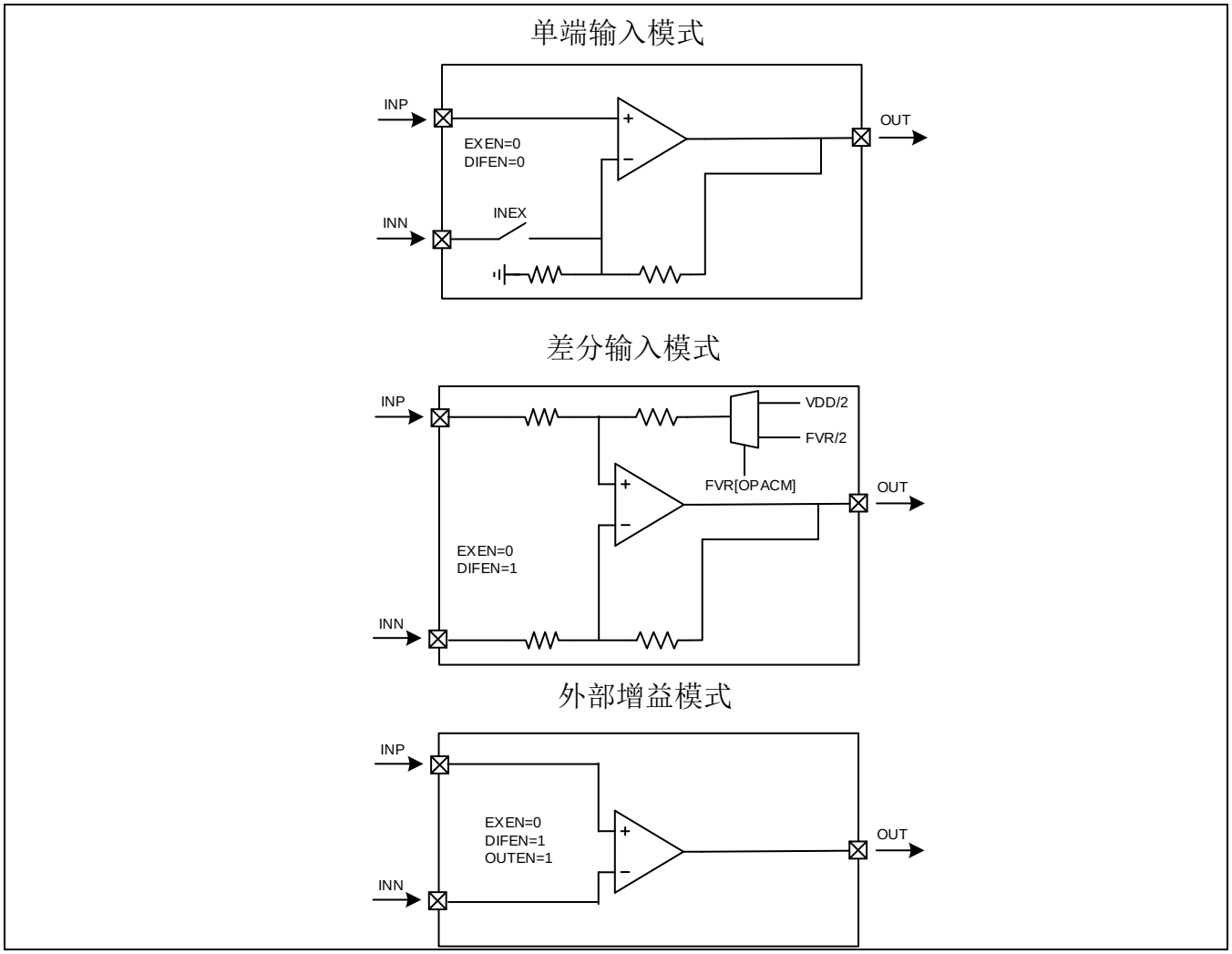


Figure 28-1 运算放大器框图

28.2.2 运算放大器工作模式

芯片内建四个高性能运算放大器，当不使用时，可以通过控制寄存器（OPA_CR）中的OPAEN位来实现软件关闭功能。每个运放提供三个外接引脚：OPAxP（正输入端口），OPAxN（负输入端口），OPAxX（输出端口）。运放使能需要先设置对应的输出引脚，只有在对应的输出端口（OPAxX）引脚选择为AF9（模拟信号通路）后，运放才能使能。在OPAxX功能未正确设置前，即使OPAEN控制位已经被置位，运放仍处于关闭状态。

运放的输入端口和ADC以及模拟比较器的正向输入端复用，当对应的GPIO设置为AF9时，可以同时使用该引脚上的所有模拟功能（输入端口的模拟信号，可以同时作为运放，ADC或者比较器的输入）。运放的输出端和ADC输入相连，可以选择是否输出至GPIO。当对应的GPIO设置为AF9时，且OUTEN=1时，运放的输出连接至GPIO引脚，可以通过选择该引脚上的ADC通道使能ADC采样。如不需要运放输出，可配置OUTEN=0，直接用ADC对运放进行采样。

运放支持内部增益控制模式工作（EXEN=0），当选择内部增益控制时，可以通过配置DIFEN使运放工作在单端输入模式或差分输入模式，并通过配置PGAC改变内部增益值。单端输入模式下，如果希望使用负向输入管脚，也可以将INEX置1，来使能IO上的负向输入功能(对应IO须设置为AF9)。

运放也支持外部部增益控制模式工作（EXEN=1，OUTEN=1），当选择外部增益控制时，可以通过外部电阻网络来控制环路增益。

在低功耗模式下，运放被强制关闭。

28.2.3 运算放大器作为ADC输入的典型应用

如下图，运算放大器使用内部增益模式，并且运算放大器的输出作为ADC的输入，同时可以配置OUTEN选择是否输出到对应的GPIO管脚。

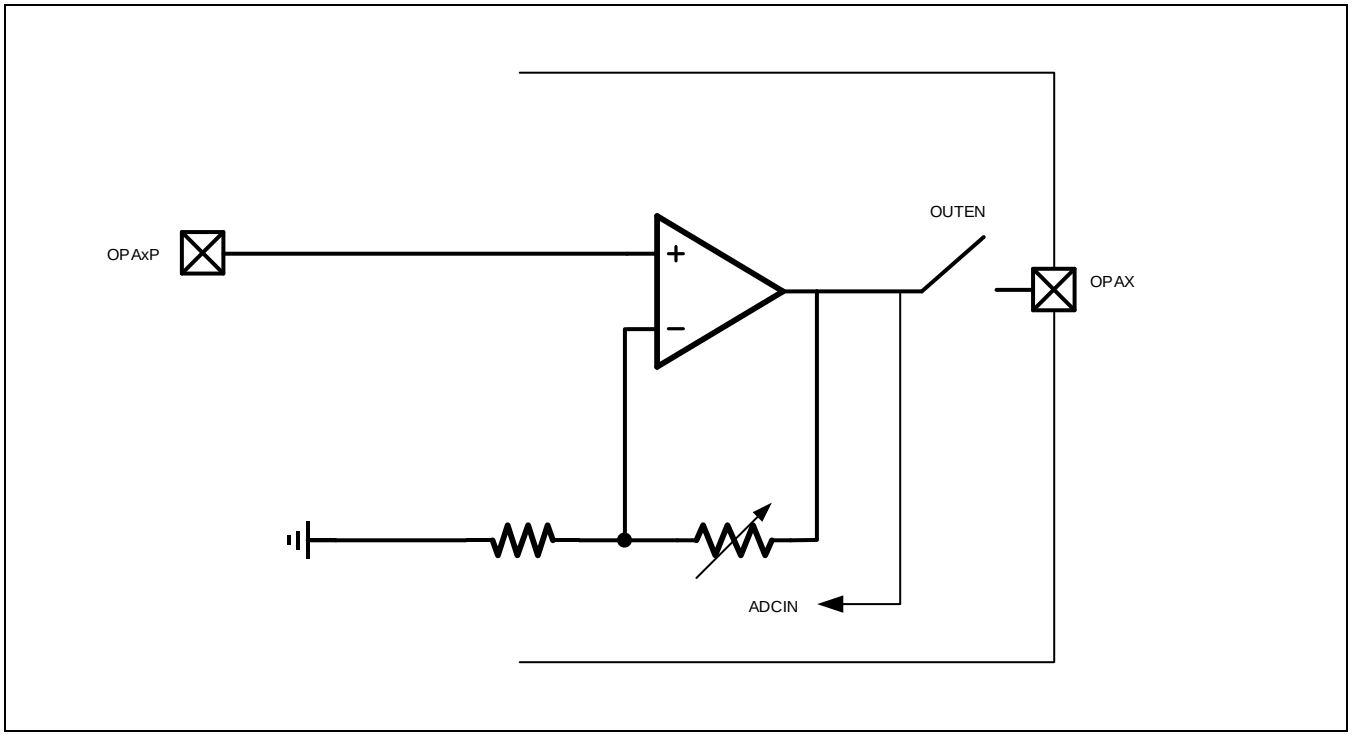


Figure 28-2 运算放大器作为 ADC 输入

28.3 寄存器说明

28.3.1 寄存器表

Base Address of OPA0: 0x400E0000
Base Address of OPA1: 0x400E1000
Base Address of OPA2: 0x400E2000
Base Address of OPA3: 0x400E3000

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

28.3.2 OPAx_CR(运算放大器0/1控制寄存器)

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

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

Name	Bit	Type	Description
OUTEN	[20]	RW	运放输出使能 0: 运放不输出至IO端口 1: 运放输出至IO端口
DIFEN	[18]	RW	差分输入模式使能 0: 单端输入模式 1: 差分输入模式
INEN	[17]	RW	负向输入使能,仅内部增益单端输入模式下使用 0: 禁止 1: 使能
EXEN	[16]	RW	外部增益模式使能 0: 内部增益模式（单端输入或差分输入模式） 1: 外部增益模式
PGAC	[4:2]	RW	内部增益控制设置。 单端输入模式下 0：X 2 1：X 4 2：X 8 3：X 16 4：X 21 5：X 32 6：X 42 7：X 64 差分输入模式下 0：X 1 1：X 2 2：X 3 3：X 5 4：X 7 5：X 9 6：X 11 7：X 19
OPAEN	[0]	RW	OPA模块的使能控制。

			0： 禁止OPA模块 1： 使能OPA模块
NOTE: （1） 内部增益控制只支持正向输入的情况			