数据手册

APT32F003

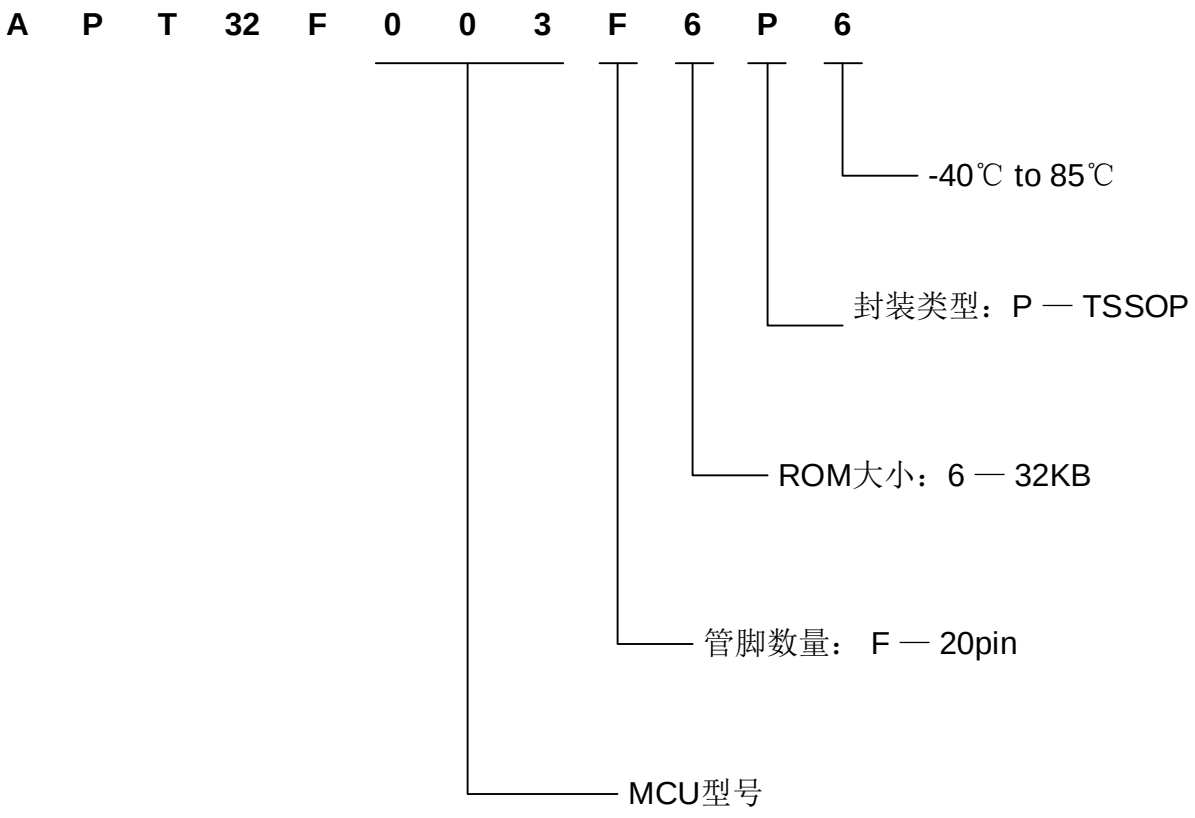
通用型 32 位微处理控制器

Revision 2.2

May 2018

产品订购信息

型号	FLASH 大小	SRAM 大小	封装
APT32F003F6P6	32KB	2KB	TSSOP20



历史版本说明

版本	修改日期	修改概要
V1.0	2016-9-10	初版
V1.1	2016-9-19	修正了一些前后表达不一致的错误
V1.2	2016-9-22	增加了系统定时器CORET章节
V1.3	2016-11-15	增加PC0.1作为复位管脚的一些描述，管脚定义和UserOption
V1.4	2017-01-12	- 电气特性中增加上电和掉电复位章节 - 根据量产测试情况，增加或修正电气特性中各个条目的参数值 - 修正一些前后表达不一致的错误
V1.5	2017-03-16	- 增加封装章节
V1.6	2017-5-11	- 增加触摸按键控制章节 (14 - TOUCH)
V1.7	2017-06-09	- ADC章节：完善一些描述，修正错误说明 - 增加烧录工具复位脚 (PA0.0 / F_RST) 的描述
V1.8	2017-08-01	- 修正SYSCON章节中IWDCR寄存器IWDT_INTW位的描述错误
V1.9	2017-12-01	- 管脚重映射部分的内容改为全部管脚复用 - 电容式触摸传感器章节增加应用参考电路
V2.0	2018-02-08	**- 上电时LVD改为默认使能，LVDCR寄存器第0位写1为禁止LVD模块 (E版本芯片)**
V2.1	2018-03-12	- TC32增加寄存器使用说明，PRDR/PULR不能设置为0 - 在PC0.3/PC0.2上增加ADC12/13通道
V2.1	2018-05-16	- 更新上电示意图，删除POR电压检测值(VPOR)

1 概述

1.1 文档用途

本文档是APT32F003产品的一个完整，详细的规格书。

1.2 APT32F003介绍

APT32F003是一个低成本，高性能的系列单片机，使用了C-Sky Microsystems Co., Ltd开发的32位单片机核。APT32F003单片机面向的应用为家电以及便携设备等。

- C-Sky 32位CPU核(0.9DMIPS)
- 片载32K(16Kbytes可选)程序闪存，4Kbytes数据闪存(大小可配置)
- 内含2Kbytes SRAM，可用于堆栈，数据存储，代码存储
- 工作温度： – 40 to 85°C
- 工作电压范围： 2.4 to 5.5V
- 中断控制器：支持动态配置的可嵌套中断 (NVIC)
- 用户友好的时钟和功耗控制器(SYSCON)
- 1 x 独立看门狗定时器(IWDT)
- 3 × 16位通用定时器/计数器，支持PWM功能 (GTC)
- 1 × 32位通用定时器/计数器，支持PWM功能 (GTC)
- 1 x 16位计数器(COUNTERA)，支持自动重载功能以及单次或者循环计数功能(载波发生器)
- 1 × I2C 和 2 × UART
- 多达14路的12位ADC
- 多大12路的触摸按键控制
- 3 x 大电流驱动的管脚(每个管脚最大120mA)
- 支持RUN，SLEEP，和DEEP-SLEEP模式

1.3 主要特性

1.3.1 CPU

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

1.3.2 存储

- 多达32Kbytes的内部程序闪存，支持ISP保护，保护区域的大小可配置
- 多达4Kbytes的数据闪存，大小可通过User Option配置
- 闪存的User Option配置支持修改复位源，允许或者禁止看门狗，修改数据闪存大小以及使能程序代码保护功能
- 专用闪存烧录工具，支持快速的量产烧录
- 多达2Kbytes的内部SRAM
- 只支持小端(little-endian)存储方式

1.3.3 可嵌套中断控制器

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

1.3.4 系统控制器(SYSCON)

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

1.3.5 IWDT: 独立看门狗定时器

- 复位时间可配置
- 可配置定时提醒中断
- 使用内部辅晶振的可编程18位递减计数器

1.3.6 GTC: 16位/32位通用定时器/计数器

- 多达3个通道的16位和1个通道的32位定时器/计数器(GTC0 to GTC3)
- 定时器的计数位数可配置
- 可以工作在捕捉，匹配和溢出，输出翻转和PWM输出模式
- 匹配和溢出中断
- 可选择内部或者外部时钟源

1.3.7 载波频率发生器 (Counter A)

- 1个16位的计数器，支持自动重载功能以及单次或者循环计数功能
- 软件/硬件可选择的载波频率输出使能/禁止控制
- 在一个周期波形内，输出高/低电平脉冲宽度可配置
- 输出极性可配置
- 可以用于驱动扬声器或者远程IR数据传输

1.3.8 UART: 通用异步收发器

- 8位数据长度，无校验位
- 可编程的波特率

1.3.9 I2C

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

1.3.10 ADC: 模数转换器

- 多达14个模拟输入通道供选择
- 12位模式支持最快500KSPS转换速度，10位模式支持最快1MSPS
- 支持连续转换模式和硬件比较转换结果
- 启动转换支持外部管脚触发或者内部定时器触发

1.3.11 触摸按键

- 针对电源敏感的应用和噪声敏感的应用，提供两种不同的检测模式
- RC模式：支持多达12个触摸按键扫描通道，分成2组
- 外部C模式：支持11个触摸按键扫描通道
- 每个通道都支持可编程的灵敏度调节
- 支持同一个管脚上的触摸扫描和LED驱动同时工作
- 可配置扫描时钟频率，用于优化功耗
- 多种扫描触发模式
- 内嵌独立的硬件算法模块，支持硬件自动按键检测和系统唤醒

1.3.12 通用IO (GPIO)

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

1.3.13 两个低功耗模式

- SLEEP: 关闭选择的系统时钟和CPU时钟
- DEEP-SLEEP: 关闭所有系统时钟和CPU时钟
- 可配置的DEEP-SLEEP唤醒源：外部中断，iWDT中断，LVD中断或者触摸按键中断

1.3.14 POR: 上电复位

1.3.15 LVD: 低电压检测

- 可配置成低电压复位功能，可选4个电压值 (2.2V/2.7V/3.3V/3.6V).
- 可配置成低电压产生中断，可选4个电压值 (2.5V/3.0V/3.9V/4.1V).

1.3.16 工作电压范围

- 2.4V to 5.5V

1.3.17 工作频率范围

- 外部主晶振 24 MHz
- 内部主晶振 20 MHz
- 内部辅晶振 500KHz 或 3MHz

1.3.18 工作温度范围

- – 40 to 85°C

1.3.19 封装

- 20-TSSOP

1.4 模块框图

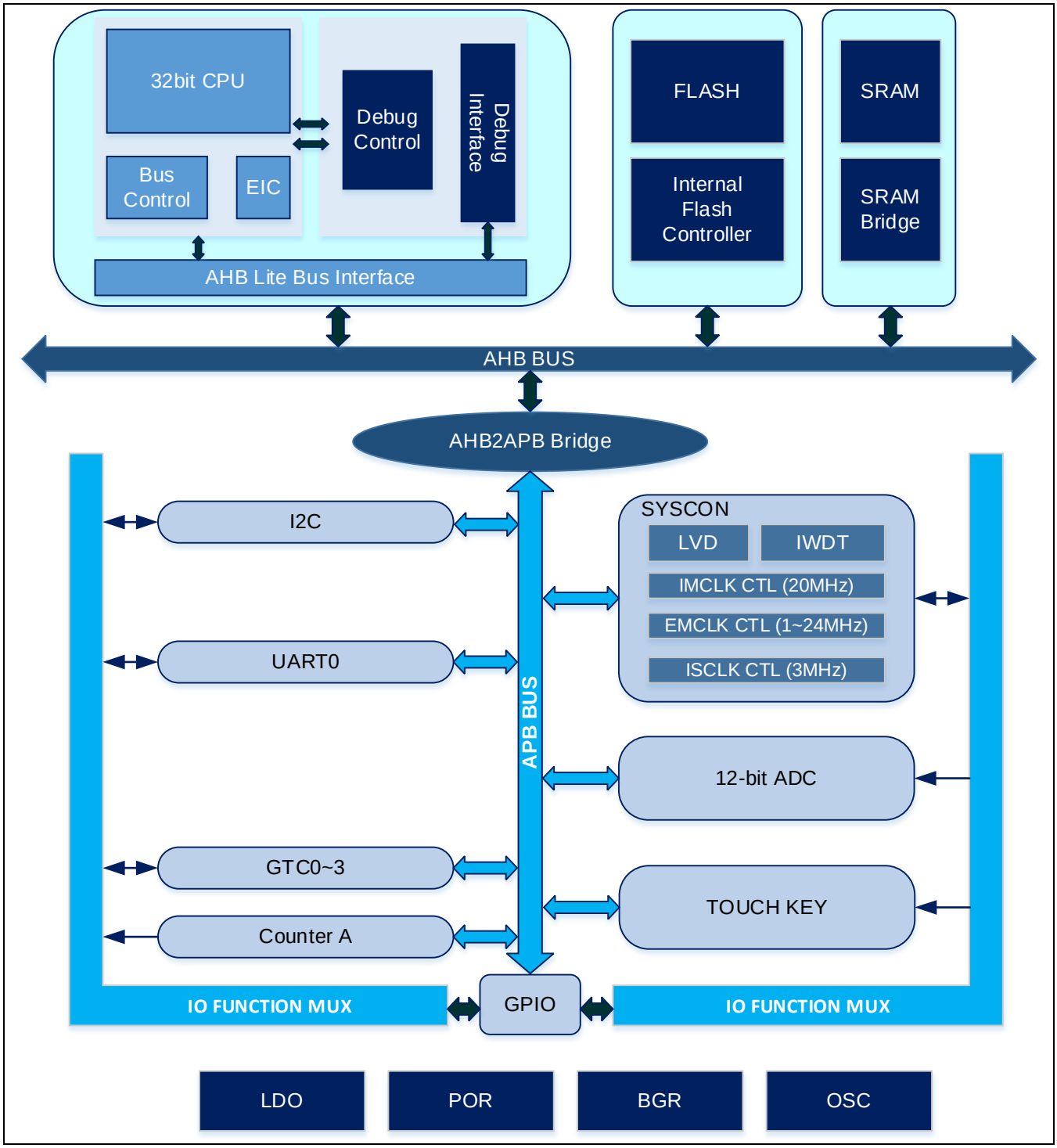


Figure 1-1 APT32F003模块框图

2 管脚配置

2.1 概要

本章节描述APT32F003产品的管脚功能信息。

包含：

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

2.2 管脚定义图

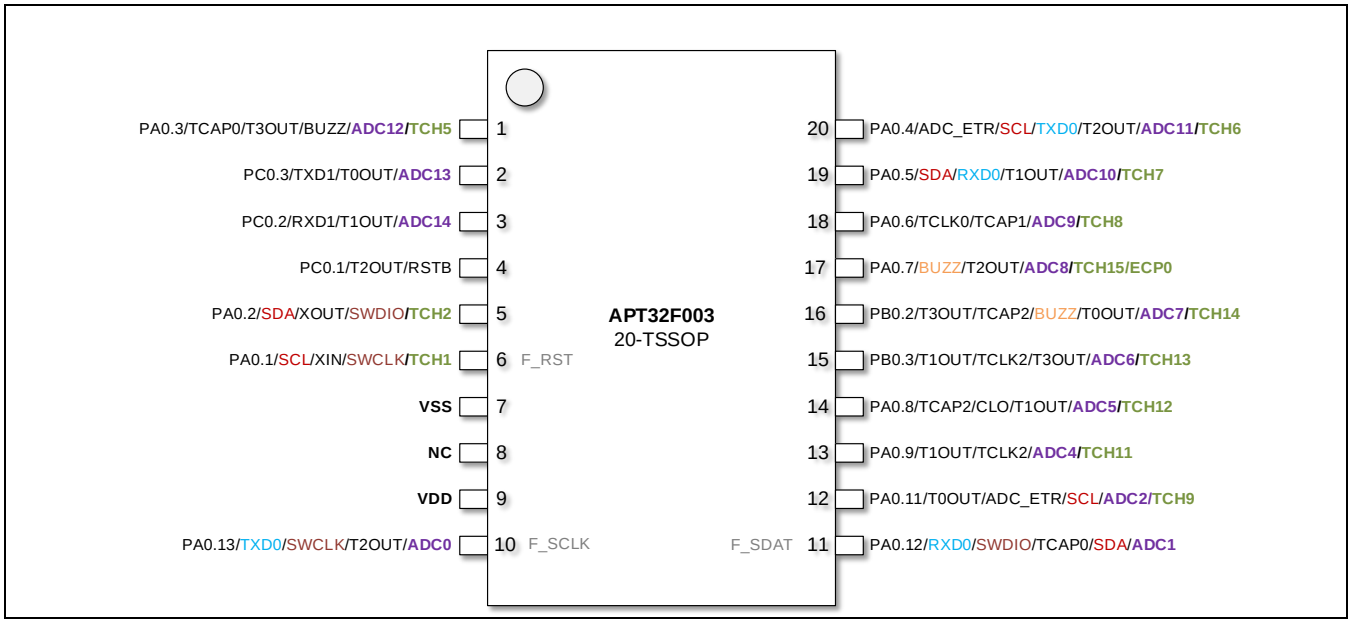


Figure 2-1 管脚定义图

2.3 管脚功能分配

Table 2-1 描述了管脚功能的详细分配。

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

Table 2-1 管脚功能分配，依照管脚号排序

管脚号	管脚名称									默认功能	上拉/下拉	复位状态
	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	EXI			
9	VDD	-	-	-	-	-	-	-	-	VDD	-	P
10	PA0.13	TXD0	SWCLK	T2OUT	SCL	-	-	ADC0	EXI13	SWCLK(4)	UP	I
11	PA0.12	RXD0	SWDIO	TCAP0	SDA	SDA	-	ADC1	EXI12	SWDIO(4)	UP	I
12	PA0.11	T0OUT	-	ADC_ETR	-	SCL	TCH9	ADC2	EXI11	IO	-	Z
13	PA0.9	T1OUT	-	TCLK2	-	-	TCH11	ADC4	EXI9	IO	-	Z
14	PA0.8	TCAP2	-	CLO	T1OUT	-	TCH12	ADC5	EXI8	IO	-	Z
15	PB0.3	T1OUT	TCLK2	-	T3OUT	-	TCH13	ADC6	EXI3	IO	-	Z
16	PB0.2	T3OUT	TCAP2	BUZZ	T0OUT	-	TCH14	ADC7	EXI2	IO	-	Z
17	PA0.7	BUZZ	-	-	T2OUT	-	TCH15 (ECP0)	ADC8	EXI7	IO	-	Z
18	PA0.6	TCLK0	-	TCAP1	-	-	TCH8	ADC9	EXI6	IO	-	Z
19	PA0.5	-	-	SDA	RXD0	T1OUT	TCH7	ADC10	EXI5	IO	-	Z
20	PA0.4	ADC_ETR	-	SCL	TXD0	T2OUT	TCH6	ADC11	EXI4	IO	-	Z
1	PA0.3	TCAP0	-	T3OUT	BUZZ	-	TCH5	ADC12	EXI3	IO	-	Z
2	PC0.3	TXD1	T0OUT	-	-	-	-	ADC13	EXI3	IO	DN	Z
3	PC0.2	RXD1	T1OUT	-	-	-	-	ADC14	EXI2	IO	DN	Z
4	PC0.1	-	T2OUT	-	-	-	-	-	EXI1	IO(1)	UP	Z
5	PA0.2	SDA	-	XOUT	SWDIO	-	TCH2	-	EXI2	IO(4)	-	Z
6	PA0.1	SCL	-	XIN	SWCLK	-	TCH1	-	EXI1	IO(4)	-	Z
7	VSS	-	-	-	-	-	-	-	-	VSS	-	P

注意：

1) 外部复位功能和PC0.1管脚复用，可以使用User Option功能选择配置
2) F_SCLK, F_SDAT为外部闪存烧录工具接口信号
3) 每个IO管脚只要配置成数字IO功能，都可以使用EXI功能来触发中断
4) 总共有两组可选SWD接口，默认的SWD接口复用管脚可以使用User Option功能配置

2.4 管脚复用

下表归纳了各种功能的管脚复用，方便用户在各种不同应用下使用各种不同的功能。

Table 2-2 复用功能概要

功能模块	功能管脚	管脚分配
TIMER0	T0OUT(O)	PA0.11(AF1) PC0.3(AF2) PB0.2(AF4)
TIMER1	T1OUT(O)	PA0.9(AF1) PB0.3(AF1) PC0.2(AF2) PA0.8(AF4) PA0.5(AF5)
TIMER2	T2OUT(O)	PC0.1(AF2) PA0.13(AF3) PA0.7(AF4) PA0.4(AF5)
TIMER3	T3OUT(O)	PA0.3(AF2) PB0.2(AF1) PB0.3(AF4)
TIMER Common	TCAP0(I)	PA0.3(AF1) PA0.12(AF3)
	TCAP1(I)	PA0.6(AF3)
	TCAP2(I)	PA0.8(AF1) PB0.2(AF2)
	TCLK2(I)	PB0.3(AF2) PA0.9(AF3)
CNTA	BUZZ(O)	PA0.7(AF1) PB0.2(AF3) PA0.3(AF4)
I2C	SCL(B)	PA0.1(AF1) PA0.4(AF3) PA0.13(AF4)
	SDA(B)	PA0.2(AF1) PA0.5(AF3) PA0.12(AF4)
UART0	TXD0(O)	PA0.13(AF1) PA0.4(AF4)
	RXD0(I)	PA0.12(AF1) PA0.5(AF4)
UART1	TXD1(O)	PC0.3(AF1)

	RXD1(I)	PC0.2(AF1)
ADC	ADC_ETR(I)	PA0.4(AF1) PA0.11(AF3)
	ADCx (A)	PA0.3~PA0.9, PA0.11~PA0.13, PB0.2~PB0.3, PC0.2~PC0.3 (AF7)
TOUCH	TCHx (A)	PA0.1~PA0.9, PA0.11, PB0.2~PB0.3 (AF6)
SYSTEM	CLO(O)	PA0.8(AF3) PA0.0(AF4)

注意：

1) 对于输出功能，如果多个管脚都被配置成同一个功能，那么所有这些管脚都会输出相同的信号。
2) 对于输入功能，如果多个管脚都被配置成同一个功能，那么AF编号小的管脚有更高的优先权呢。例如，当PA0.12和PA0.5都被配置成RXD0时，只有PA0.12(AF1)是RXD0，而PA0.5(AF4)的RXD0配置无效。
3) 功能管脚括号中的 I：输入， O: 输出， B: 双向， A: 模拟。

2.5 管脚功能说明

本段落描述了以下管脚的功能：

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

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

2.5.1 电源管脚

Table 2-3 电源管脚说明

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

2.5.2 系统功能管脚

Table 2-4 系统功能管脚说明

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

2.5.3 普通模块功能管脚

Table 2-5 普通模块功能管脚说明

模块	管脚名称	I/O	管脚说明	D/A
GPIO	PA0.x	I/O	通用IO A	D
	PB0.x	I/O	通用IO B	D
	PC0.x	I/O	通用IO C	D

TIMER	TCLKx	I	定时器的外部时钟输入	D
	TCAPx	I	定时器的外部捕捉输入	D
	TxOUT	O	定时器的翻转输出或者PWM输出	D
CNTA	BUZZ	O	计数器A的载波频率输出	D
I2C	SCL	I	I2C串行时钟	D
	SDA	I/O	I2C串行数据	D
UART	RXD	I	UART串行数据接收	D
	TXD	O	UART串行数据发送	D
ADC	ADCx	I	ADC模拟输入通道	A
	ADC_ETR	I	ADC外部启动触发输入	D
TOUCH	TCHx	I/O	触摸按键扫描通道	A
	ECP0	I/O	触摸按键在外部电容模式中的外接电容管脚	A

2.5.4 调试接口管脚

Table 2-6 调试接口管脚说明

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

2.5.5 闪存烧录工具管脚

Table 2-7 闪存烧录工具管脚说明

模块	管脚名称	I/O	管脚说明	D/A
FLASH	F_SCLK	I	串行时钟	D
	F_SDAT	I/O	串行数据	D
	F_RST	I	复位	D

3 系统存储空间

3.1 概述

本章节介绍了 APT32F003 系统存储空间管理。

本章包含内容如下：

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

3.2 默认存储地址表

Table 3-1 存储地址

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

3.3 特殊功能寄存器表

特殊功能寄存器表有如下两种形式

- CPU特殊功能寄存器表
- 外围设备特殊功能寄存器表

3.3.1 CPU特殊功能寄存器表

Table 3-2 CPU SFR 表

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

3.3.2 外围设备特殊功能寄存器表

Table 3-3 外围设备SFR表

Peripheral	Base Address	Function Description
I2C	0x400A_0000	I2C串行接口 (I2C)
UART	0x4008_1000	通用异步收发器1 (UART1)
	0x4008_0000	通用异步收发器0 (UART0)
CNTA	0x4007_0000	计数器 A (COUNTERA)
LED	0x4006_0000	LED显示控制器 (LED)
TC	0x4005_3000	通用定时器/计数器 模块-3 (TIMER3)
	0x4005_2000	通用定时器/计数器 模块-2 (TIMER2)
	0x4005_1000	通用定时器/计数器 模块-1 (TIMER1)
	0x4005_0000	通用定时器/计数器 模块-0 (TIMER0)
GPIO	0x4004_1000	通用IO端口-B (GPIO B)
	0x4004_0000	通用IO端口-A (GPIO A)
ADC	0x4003_0000	模数转换器 (ADC)
TKEY	0x4002_0000	电容式触摸按键传感器 (TOUCH)
SYSTEM	0x4001_2000	系统控制器 (SYSCON)
	0x4001_1000	闪存控制器 (IFC)
	0x4001_0000	设备信息寄存器 (Device ID)

4 中断向量控制器(INTC)

4.1 概述

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

4.1.1 特性

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

4.2 中断向量表

Table 4-1 System Interrupt Vectors

Number	Address	Vector	Interrupt Sources
32	0x0000_0080	CORET	CK802 CPU Core Timer
33	0x0000_0084	SYSCON	System controller interrupt
34	0x0000_0088	IFC	Program flash controller interrupt
35	0x0000_008C	ADC	ADC Interrupt
36	0x0000_0090	GTC0	General Timer/Counter 0 Interrupt
37	0x0000_0094	GTC1	General Timer/Counter 1 Interrupt
38	0x0000_0098	GTC2	General Timer/Counter 2 Interrupt
39	0x0000_009C	EXI0	External interrupt 0
40	0x0000_00A0	EXI1	External interrupt 1
41	0x0000_00A4	GTC3	General Timer/Counter 3 Interrupt
42	0x0000_00A8	-	Reserved
43	0x0000_00AC	-	Reserved
44	0x0000_00B0	-	Reserved
45	0x0000_00B4	UART0	UART 0 interrupt
46	0x0000_00B8	UART1	UART 1 interrupt
47	0x0000_00BC	-	Reserved
48	0x0000_00C0	-	Reserved
49	0x0000_00C4	I2C	I2C interrupt
50	0x0000_00C8	-	Reserved
51	0x0000_00CC	-	Reserved
52	0x0000_00D0	-	Reserved
53	0x0000_00D4	EXI2	External Interrupt 2 ~ 3
54	0x0000_00D8	EXI3	External Interrupt 4 ~ 8
55	0x0000_00DC	EXI4	External Interrupt 9 ~ 13
56	0x0000_00E0	CNTA	COUNTER A interrupt
57	0x0000_00E4	TKEY	Touch Key interrupt
58	0x0000_00E8	-	Reserved
59	0x0000_00EC	LED	LED interrupt
60	0x0000_00F0	-	Reserved
61	0x0000_00F4	-	Reserved
62	0x0000_00F8	-	Reserved
63	0x0000_00FC	-	Reserved

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

4.3 工作原理

4.3.1 中断处理

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

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

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

中断的处理过程可以分以下几个步骤进行：

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

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

4.3.2 中断优先级和中断抢占

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

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

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

中断抢占的优先级条件可分为两种：

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

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

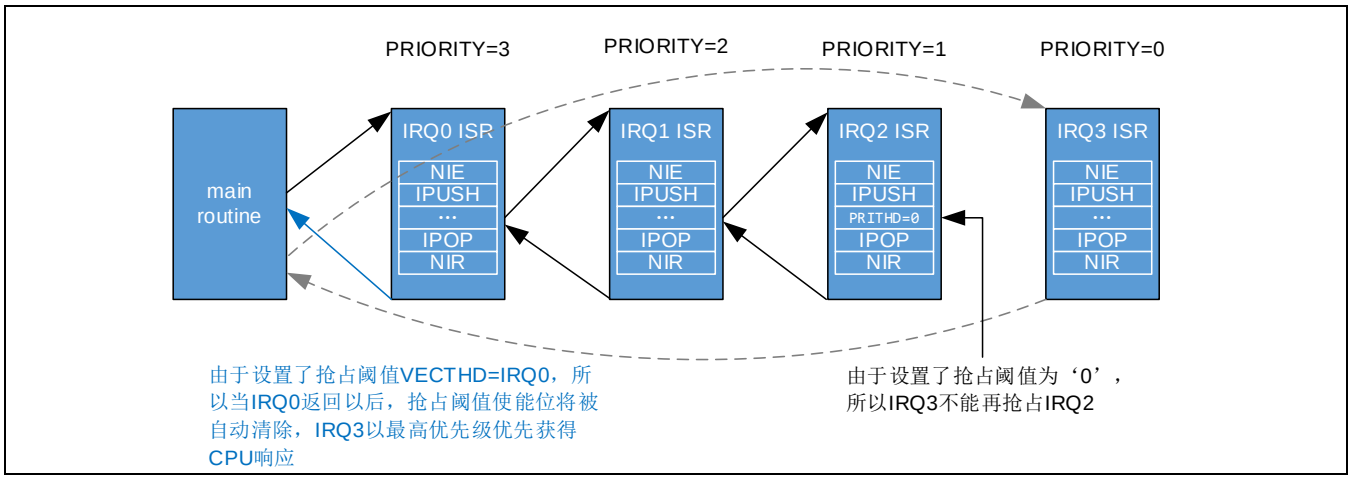


Figure 4-1 中断嵌套优先级示意图

4.3.3 中断响应时间和中断嵌套条件

一般中断在指令的边界上被确认，如下图所示。

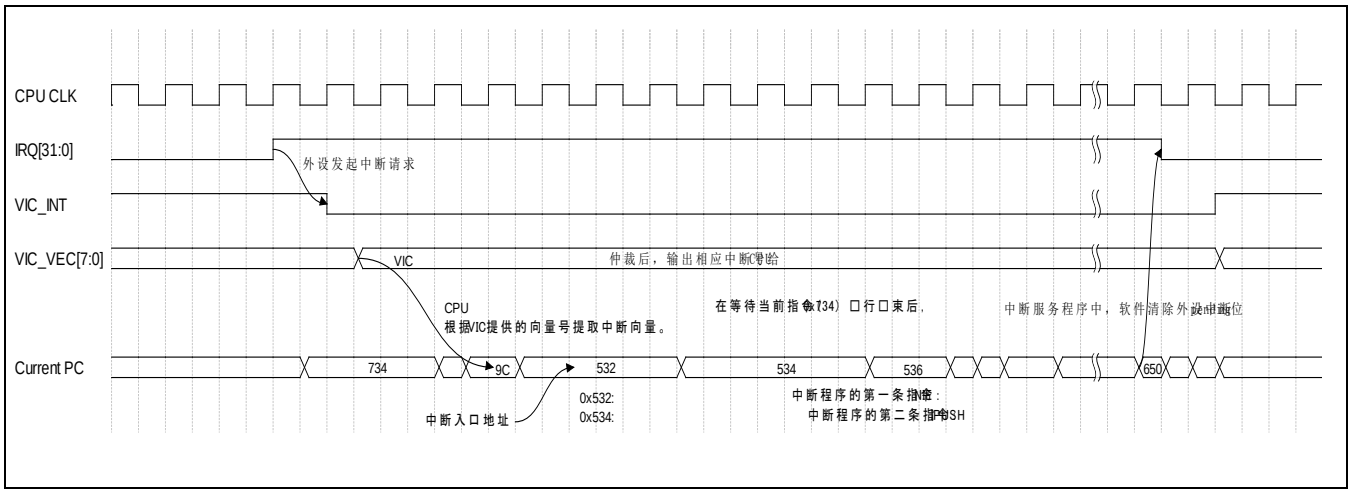


Figure 4-2 中断响应过程

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

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

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

为保证中断嵌套现场的保护和恢复，CPU在以下阶段不能被中断打断：

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

CPU在以下阶段可以安全的响应新的中断：

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

下图给出了IRQ0/IRQ1/IRQ2/IRQ3中断的嵌套过程。

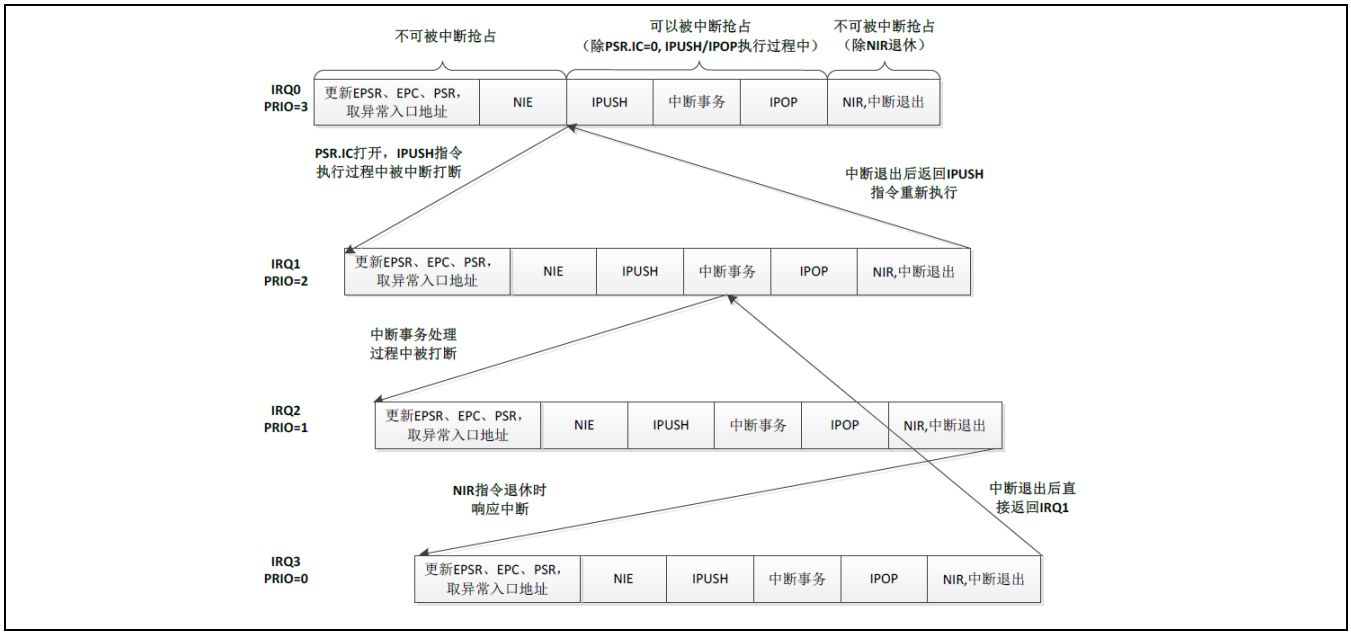


Figure 4-3 中断嵌套条件示例

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

4.3.4 中断唤醒

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

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

.

4.3.5 中断操作步骤

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

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

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

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

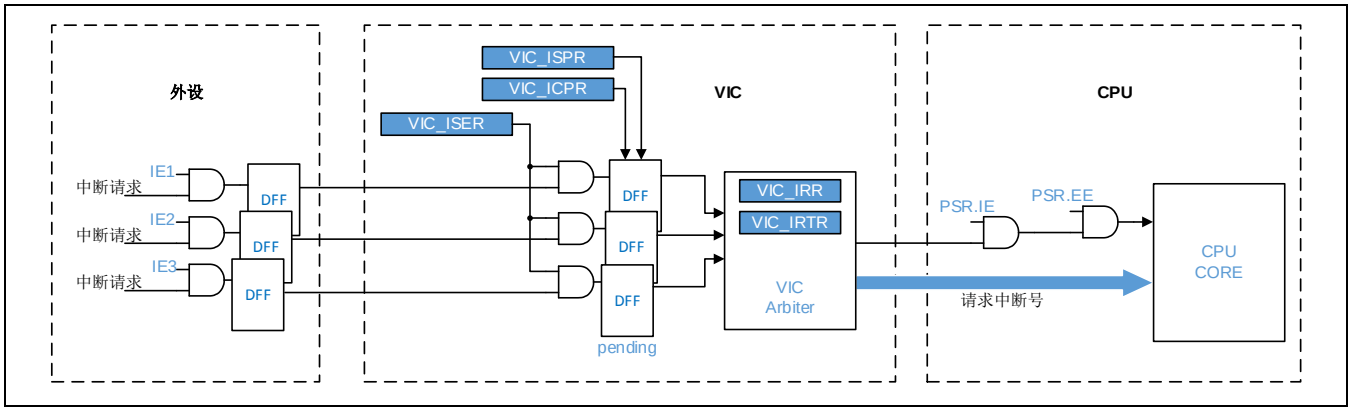


Figure 4-4 中断配置结构示意图

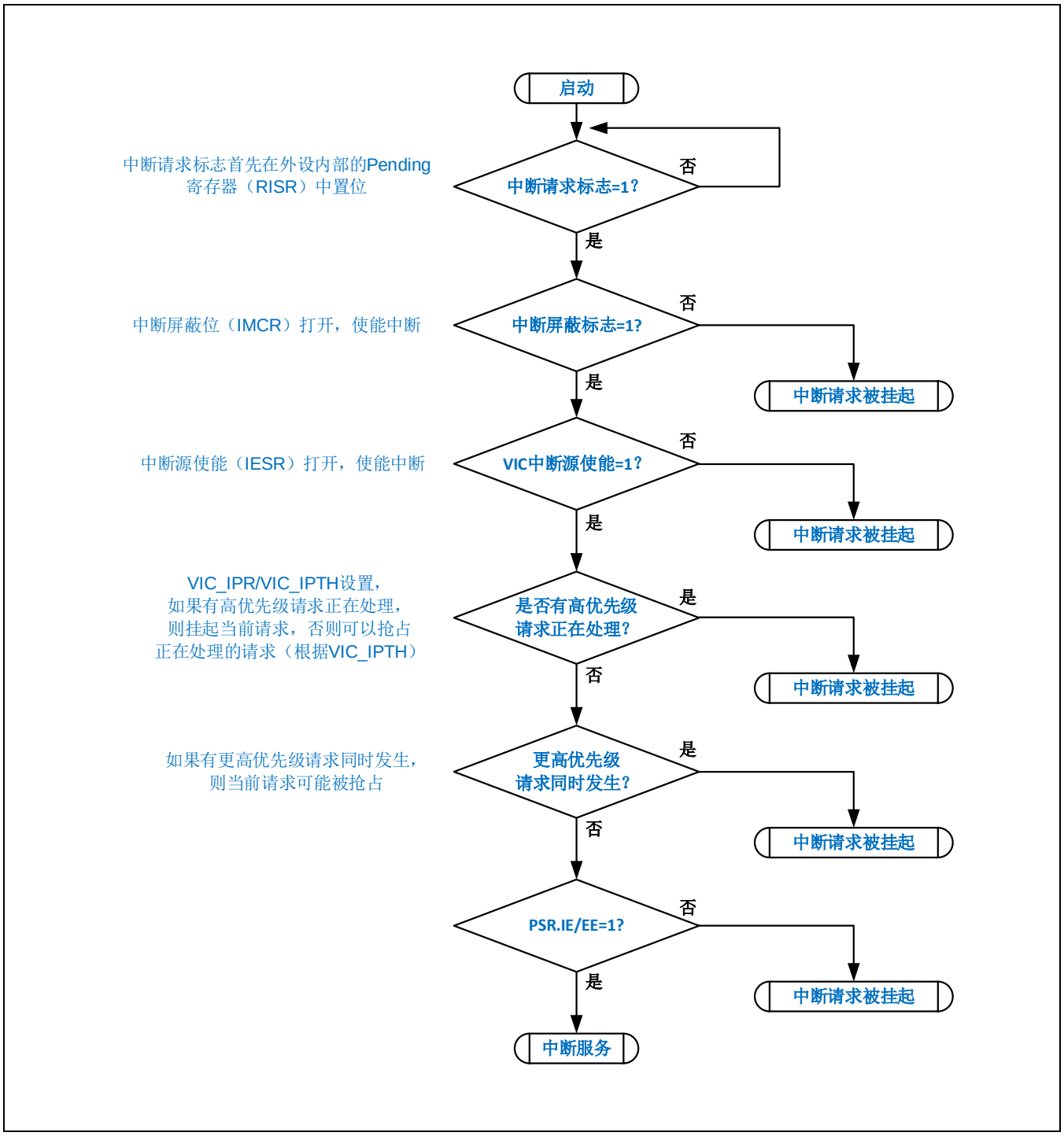


Figure 4-5 中断请求处理流程

4.4 寄存器说明

4.4.1 寄存器表

- Base Address: 0xE000_E000

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

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

NOTE:

4.4.2 VIC_ISER (中断设置使能寄存器)

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

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

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

4.4.3 VIC_IWER (中断低功耗唤醒使能寄存器)

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

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

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

4.4.4 VIC_ICER (中断使能清除寄存器)

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

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

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

4.4.5 VIC_IWDR (中断低功耗唤醒清除寄存器)

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

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

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

4.4.6 VIC_ISPR (中断等待设置寄存器)

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

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

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

4.4.7 VIC_ICPR (中断等待设置寄存器)

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

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

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

4.4.8 VIC_IABR (中断响应状态寄存器)

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

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

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

4.4.9 VIC_IPR0 (中断优先级设置寄存器0)

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

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

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

4.4.10 VIC_IPR1 (中断优先级设置寄存器1)

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

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

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

4.4.11 VIC_IPR2 (中断优先级设置寄存器2)

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

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

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

4.4.12 VIC_IPR3 (中断优先级设置寄存器3)

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

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

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

4.4.13 VIC_IPR4 (中断优先级设置寄存器4)

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

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

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

4.4.14 VIC_IPR5 (中断优先级设置寄存器5)

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

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

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

4.4.15 VIC_IPR6 (中断优先级设置寄存器6)

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

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

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

4.4.16 VIC_IPR7 (中断优先级设置寄存器7)

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

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

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

4.4.17 VIC_ISR (中断状态寄存器)

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

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

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

4.4.18 VIC_IPTR (中断优先级阈值寄存器)

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

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

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

5 系统定时器 (CORET)

5.1 概述

系统定时器是 CK802 CPU一个内部模块，它主要用于计时。系统定时器提供了一个简单易用的 24 位循环递减的计数器，当系统计时器使能时，计数器开始工作。当计数器递减到 0 时，会向中断控制器发起中断请求。

5.2 功能描述

5.2.1 模块框图

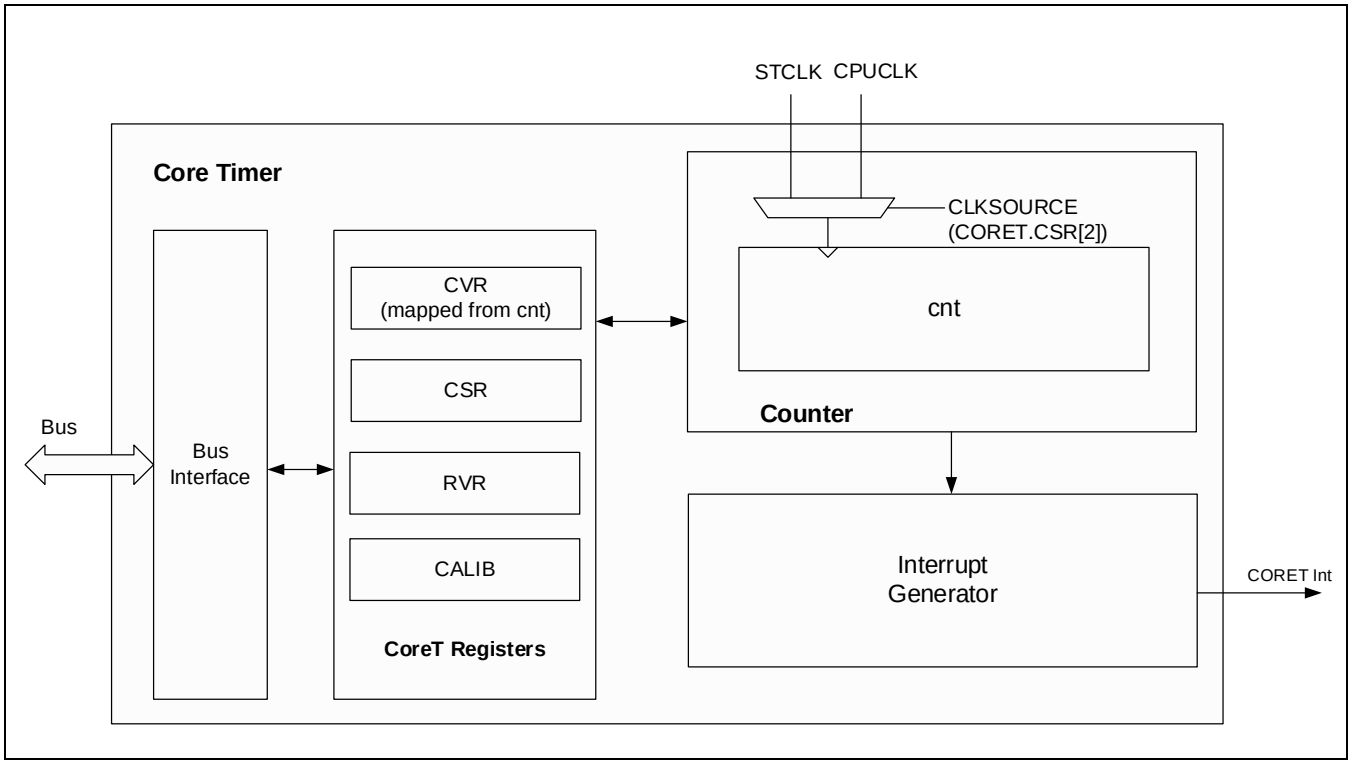


Figure 5-1 CORET模块框图

5.2.2 功能说明

5.2.2.1 定时器的时钟源

CORET 定时器有两个可选时钟源：

- CPU 时钟 (CORECLK)
- 系统时钟的 8 分频 STCLK

时钟源的选择通过 CSR 寄存器的第 2 位 CLKSOURCE 来实现。

时钟的使能/禁止和各种配置请参考 SYSCON 章节。

5.2.2.2 定时器的工作原理

CORET定时器是CK802 CPU提供的一个简单易用的24位循环递减的计数器，包含在CPU Core内部，产生的中断具有最高的优先级。CORET定时器可以用作任何简单的计时，或者可以作为操作系统的SYSTICK定时器。

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

5.3 寄存器说明

5.3.1 寄存器表

Base Address: 0xE000_E000

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

5.3.1.1 CORET_CSR (控制寄存器)

- Address = Base Address + 0x0010

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

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

5.3.1.2 CORET_RVR (回填值寄存器)

- Address = Base Address + 0x0014

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

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

5.3.1.3 CORET_CVR (当前值寄存器)

- Address = Base Address + 0x0008

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

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

6 闪存控制器(IFC)

6.1 概述

本章节描述用来控制内部程序存储的闪存控制器。APT32F003 系列片上带有 32K/16K 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。APT32F003 系列还支持额外的数据闪存(DROM)存储空间，让用户在掉电之前存储一些应用程序需要的数据。数据闪存(DROM)的大小可以通过 User Option 配置。

6.1.1 主要特性

- 程序闪存(PROM)大小: 32K/16K Bytes
- 数据闪存(DROM)大小: 4K/2K/1K Bytes
- 编程支持ISP模式和专用的工具模式
- 页大小: 1K Bytes
- 可擦除单元: 页
- 可靠性: PROM和DROM都为200,000次
- 可自定义的选项(称为User Option)支持iWDT使能和禁止，配置复位管脚
- 支持各种保护：调试接口保护，硬件保护和读保护

6.2 功能描述

6.2.1 模块框图

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

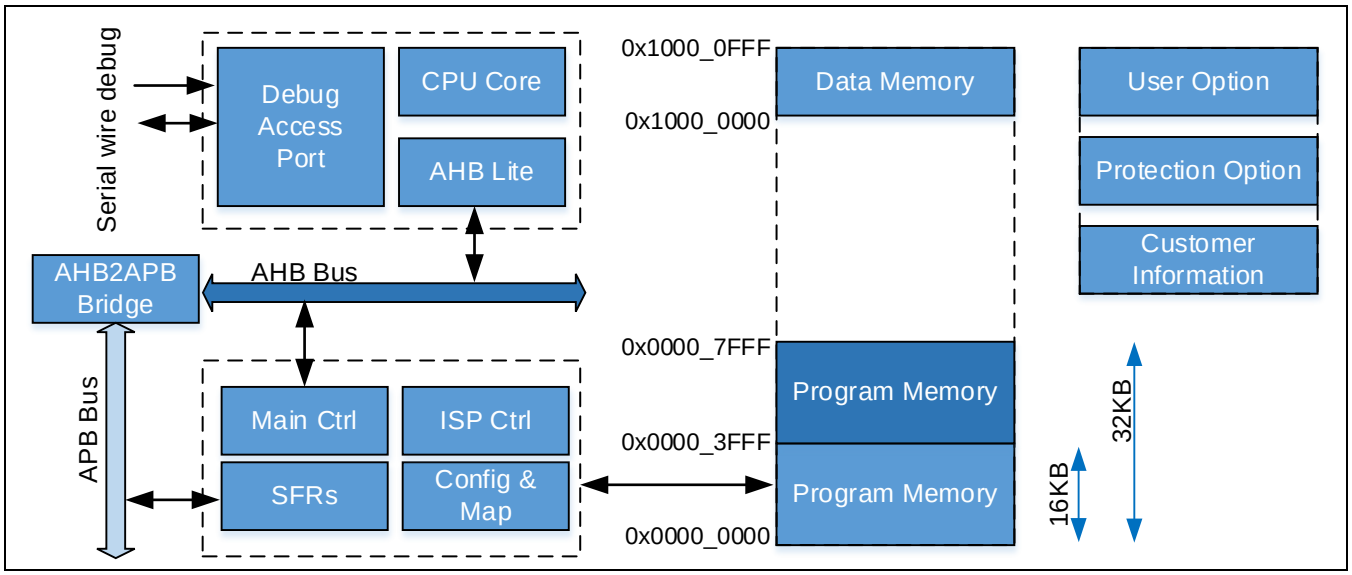


Figure 6-1 IFC模块框图

6.2.2 模块结构

APT32F003 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 36/35/34/32/16 个页空间，每页有 1K 字节。最小的擦除单元为页空间，用户可以每次指定一个页空间的单位地址进行页擦除操作，指定一个字(Word)的单位地址进行写操作。

区域	页名称	大小	起始地址	结束地址
PROM Within 16KB	Page 0	1KB	0x0000_0000	0x0000_03FF
	Page 1	1KB	0x0000_0400	0x0000_07FF
	Page 2	1KB	0x0000_0800	0x0000_0BFF
	Page 3	1KB	0x0000_0C00	0x0000_0FFF
	Page 4	1KB	0x0000_1000	0x0000_13FF
	Page 5	1KB	0x0000_1400	0x0000_17FF
	Page 6	1KB	0x0000_1800	0x0000_1BFF
	Page 7	1KB	0x0000_1C00	0x0000_1FFF
	:	:	:	:

	Page 14	1KB	0x0000_3800	0x0000_3BFF
	Page 15	1KB	0x0000_3C00	0x0000_3FFF
PROM Within 32KB	Page 16	1KB	0x0000_4000	0x0000_43FF
	Page 17	1KB	0x0000_4400	0x0000_47FF
	Page 18	1KB	0x0000_4800	0x0000_4BFF
	Page 19	1KB	0x0000_4C00	0x0000_4FFF
	:	:	:	:
	Page 30	1KB	0x0000_7800	0x0000_7BFF
	Page 31	1KB	0x0000_7C00	0x0000_7FFF
34KB	Page 32	1KB	0x0000_8000	0x0000_83FF
	Page 33	1KB	0x0000_8400	0x0000_87FFF
35KB	Page 34	1KB	0x0000_8800	0x0000_8BFF
36KB	Page 35	1KB	0x0000_8C00	0x0000_8FFF
DROM	Page 0	1KB	0x1000_0000	0x1000_03FF
	Page 1	1KB	0x1000_0400	0x1000_07FF
	Page 2	1KB	0x1000_0800	0x1000_0BFF
	Page 3	1KB	0x1000_0C00	0x1000_0FFF

Table 6-1 闪存地址映射

闪存结构如下图所示：

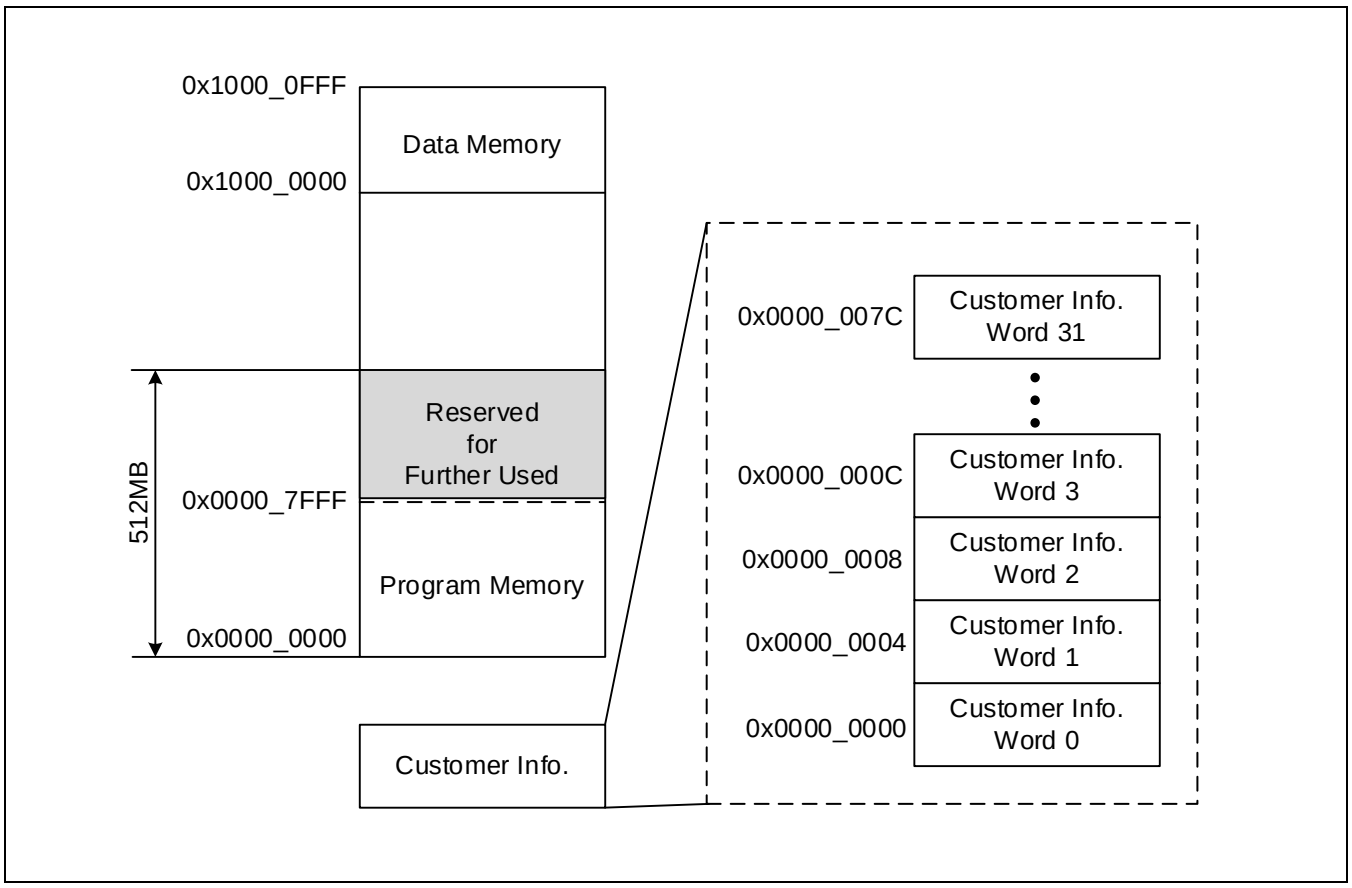


Figure 6-2 闪存地址空间结构

6.2.3 数据闪存

APT32F003 系列支持数据闪存，给用户存储普通数据。数据闪存可以通过 ISP 编程进行读写。擦除的最小单位为页。当需要改变某个字节时，该页中所有 256 个字节都需要提前复制到另一页里或者 SRAM 里暂存，或者使用特殊的软件算法在多页中轮循，模拟 EEPROM 的操作。尽管 PROM 也支持 ISP 功能，但为了数据的安全性和程序代码的完成性，我们强烈建议使用数据闪存空间来存放应用所需要存储的信息，而不是使用程序闪存。在进行全芯片擦除操作时，数据闪存和程序闪存一样都会被擦除掉。

在 32KB 闪存的产品里，用户可以选择数据闪存的大小，支持的选项为 4KB/2KB/1KB 和没有数据闪存(0KB)。最大位 4KB。如果不需要 4KB 数据闪存，可以选择 2KB/1KB/0KB，这样剩下的数据闪存空间则可以被用来当做程序闪存空间。参考 User Option 选项里的 DSIZE 位。

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

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

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

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

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

名称	位	描述
IWDT	[31:16]	独立看门狗电路使能/禁止 **IWDT[15:0]** / **功能** 0x5555 / 禁止 Other / 使能
DSIZE	[9:8]	数据闪存大小控制 **DSIZE[1:0]** / **数据闪存大小** / **程序闪存大小** 11 / 4KB / 32KB 10 / 2KB / 34KB 01 / 1KB / 35KB 00 / 0KB / 36KB
EXTRST	[7:4]	外部复位管脚功能 **EXTRST[3:0]** / **功能** 0x5 / PA0.1为外部复位管脚，程序中禁止使用该PA0.1的IO功能 0xA / PC0.1为外部复位管脚，IO功能被自动禁用 其它值 / PA0.1/PC0.1禁用外部复位功能，当作IO使用
SWDCFG	[3:2]	SWD 复位管脚配置 **SWDCFG[1:0]** / **功能** 0x0 / 切换到管脚 PA0.1, PA0.2 Other / 默认管脚 PA0.12, PA0.13

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

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

- 硬件(Hard-lock)保护

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

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

  IFC_FULL: 保护闪存全部PROM区域的内容

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

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

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

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

- 读保护

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

- 调试接口(SWD)保护

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

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

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

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

6.2.5 读操作

闪存控制器支持最大 25MHz 系统频率下的 0-wait 读取。当频率超过 25MHz 时，读取闪存时需要增加额外的等待周期。APT32F003 系列只支持最大 24MHz 的系统频率，所以 0-wait 不需要设置保持默认值 0 即可。

6.2.6 烧写方法

用户可以通过下面几种方法将数据或者代码(烧)写进闪存

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

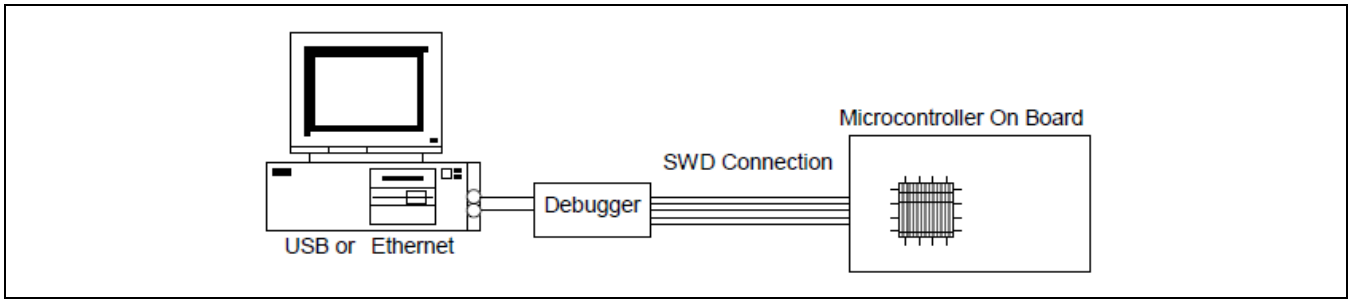


Figure 6-3 通过调试接口的烧写

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

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

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

信号	管脚名称	I/O	描述
VDD	VDD	P	芯片电源
VSS	VSS	G	芯片地
SDAT	PA0.12	I/O	串行双向数据管脚
SCLK	PA0.13	I	串行时钟输入管脚，内部默认上拉

Table 6-2 闪存烧写信号

6.2.7 ISP功能

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

6.2.7.1 写闪存操作

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

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

示例：

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

6.2.7.2 页擦除操作

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

示例：

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

6.2.7.3 片擦除操作

片擦除操作会擦除整个闪存的程序存储和数据存储区域，但不会擦除自定义选项的区域。片擦除操作只能在用户特权模式下才能执行。在片擦除中，不需要指令 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 0x3(片擦除操作)，HMODE[1:0]写为 0x1(用户特权模式)，最后将 IFC_CR 的 START 位置 1 启动该操作的执行。在 ISP 操作完成后，IFC_RISR 里的 END 位会置 1。用户同时也可以查询 IFC_CR 里的 START 位来判断 ISP 操作是否完成。

示例：

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

6.2.7.4 烧写自定义选项操作

共有 4 种自定义的选项支持通过 ISP 操作(HDP, RDP, DBP, EXTRST)。在 ISP 操作前，用户必须将秘钥 0x5A5A_5A5A 写入 IFC_KEY 寄存器以禁止闪存模块的擦除/烧写保护。之后，将 IFC_CMR 里的指令 CMD[3:0]写为 0x09/0x0A/0x0B/0x0C/0x0D/0x0E/0x0F，HMODE[1:0]写为 0x1(用户特权模式)，最后将 IFC_CR 的 START 位置 1 启动该操作的执行。在 ISP 操作完成后，IFC_RISR 里的 END 位会置 1。用户同时也可以查询 IFC_CR 里的 START 位来判断 ISP 操作是否完成。在这个操作中，不需要设置 ISP 的地址和数据寄存器。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：

```
CSP_IFC_SET_KR(IFC, USER_KEY);                  // Write Key
CSP_IFC_SET_CMR(IFC, HIDM1|EXTRST_EN);          // External reset enable
CSP_IFC_SET_CR(IFC, START);                     // Start Program
while ( CSP_IFC_GET_CR(IFC) != 0x0 );           // Wait for operation done
```

6.2.7.5 擦除自定义选项区域

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

示例：

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

6.2.8 闪存控制器的中断

闪存操作有 5 个中断源，如下所示。

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

Table 6-3 中断源描述

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

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

6.2.9 闪存控制流程图

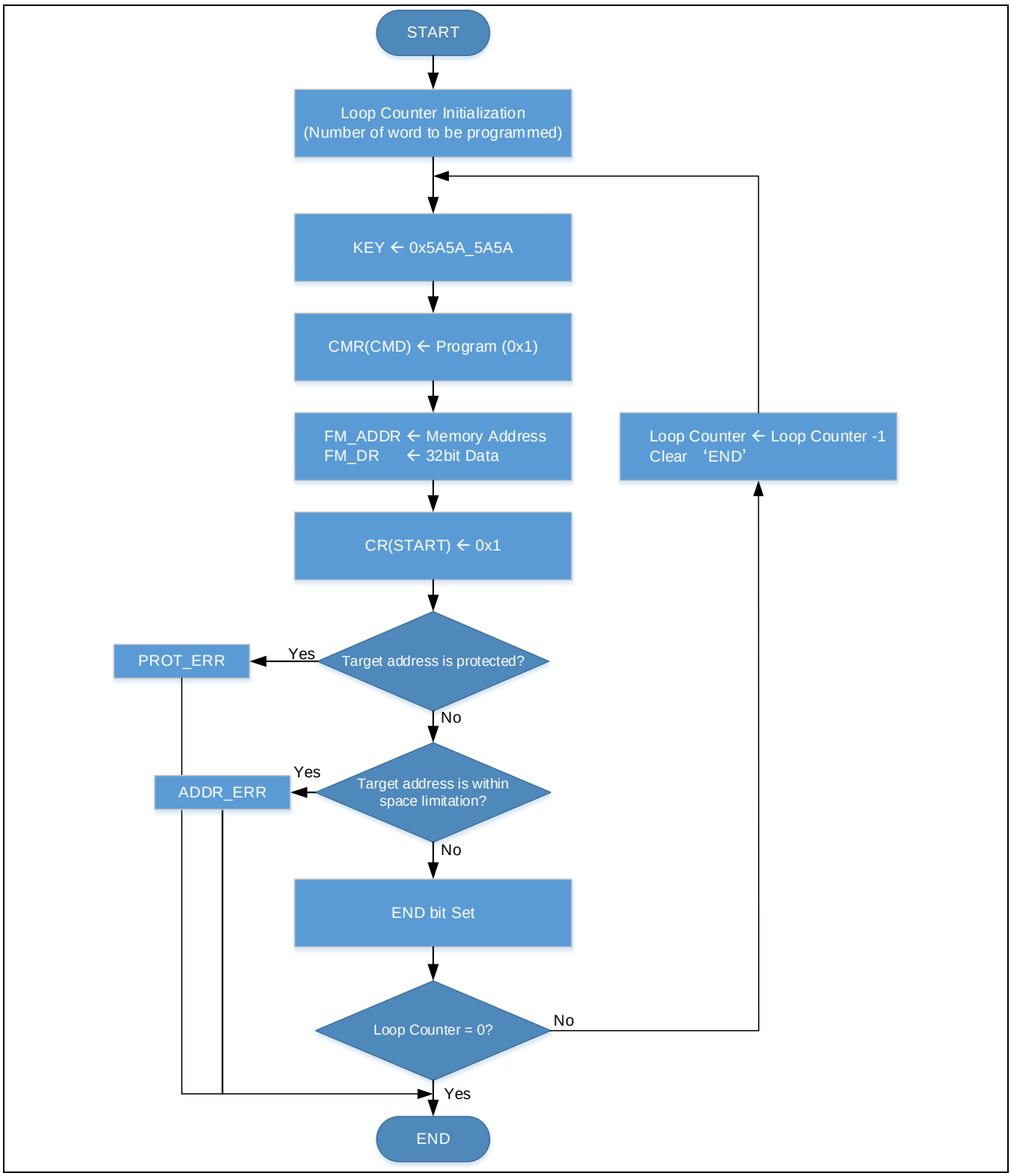


Figure 6-4 写操作流程图

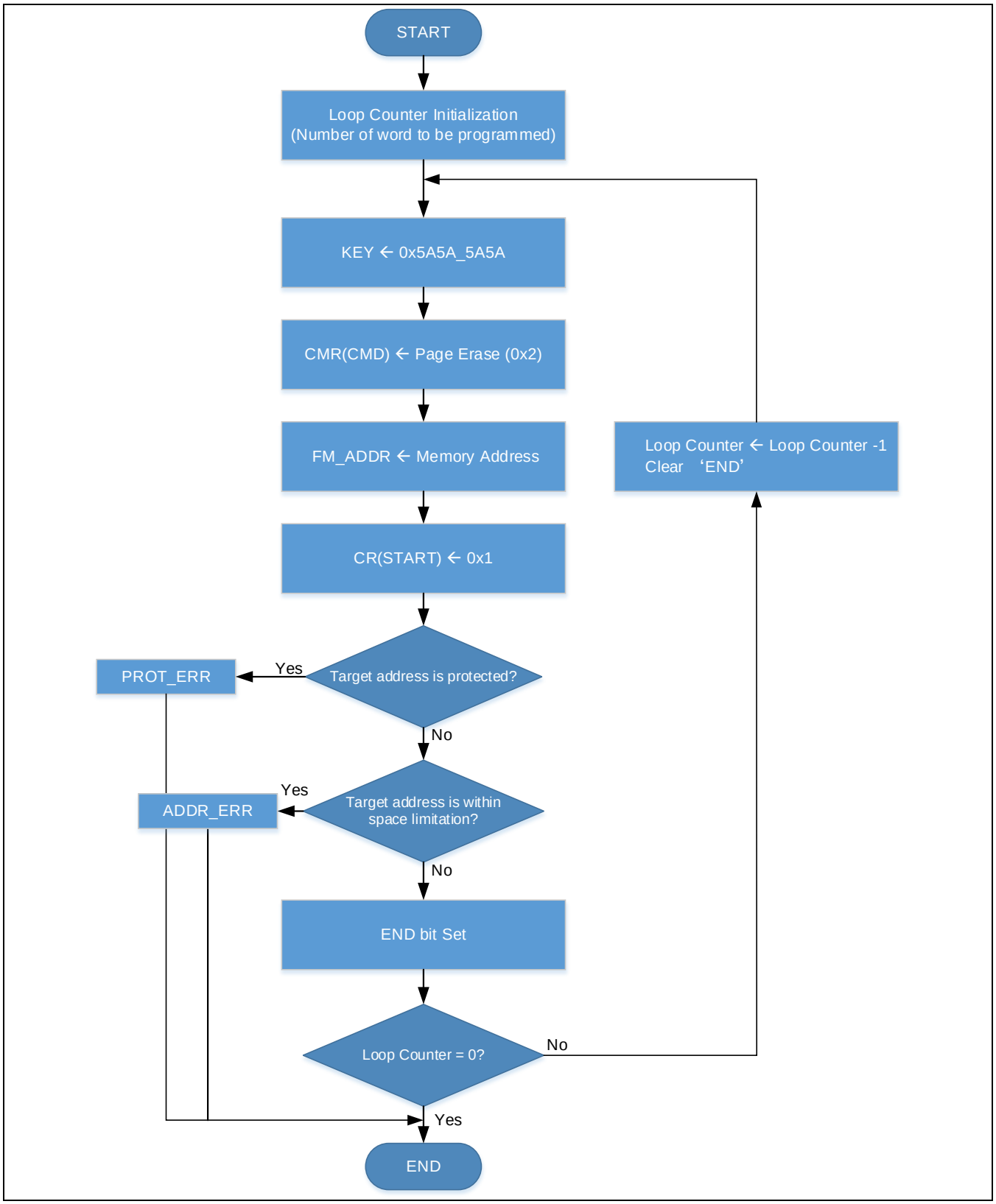


Figure 6-5 页擦除操作流程图

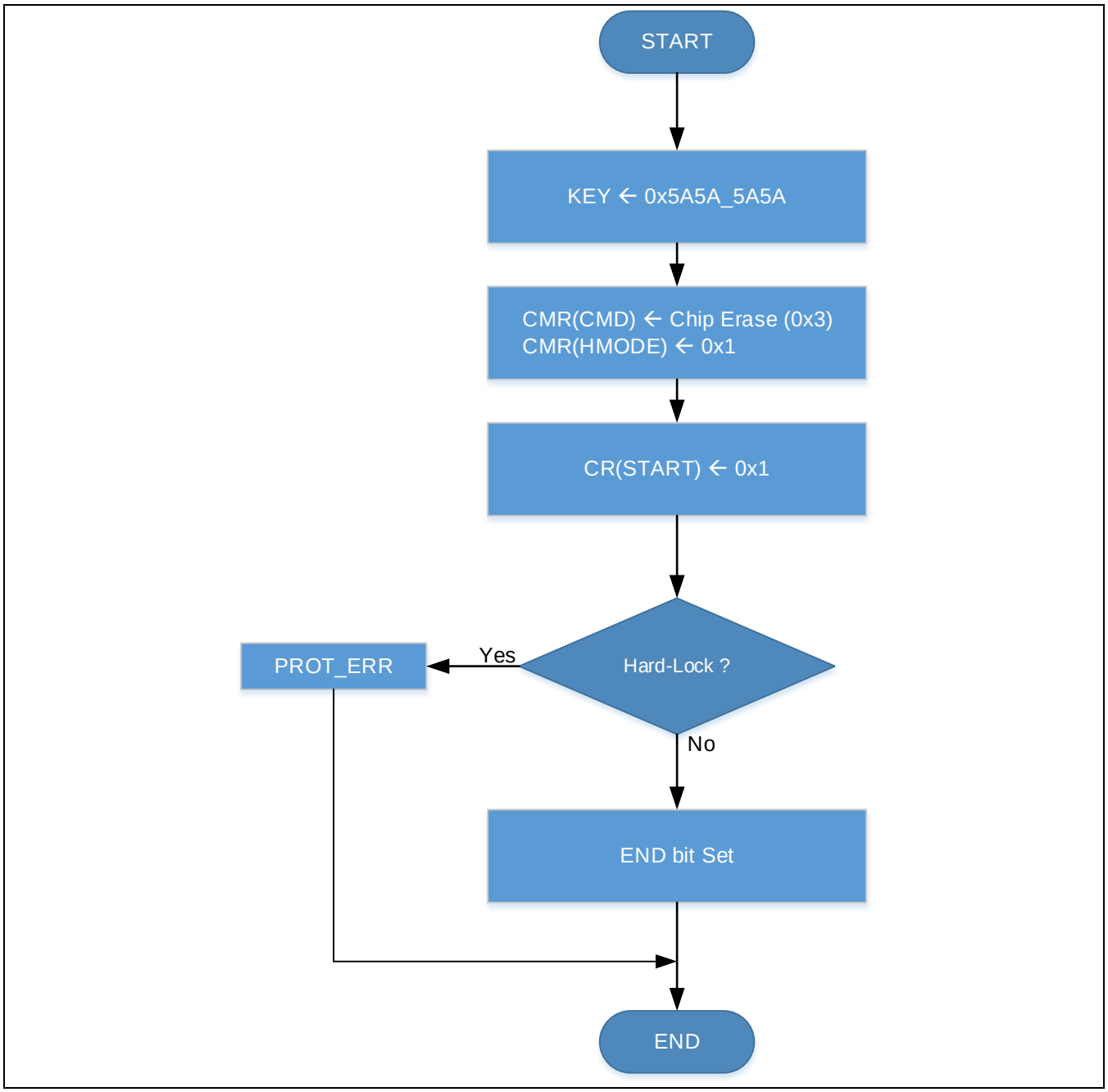


Figure 6-6 片擦除操作流程图

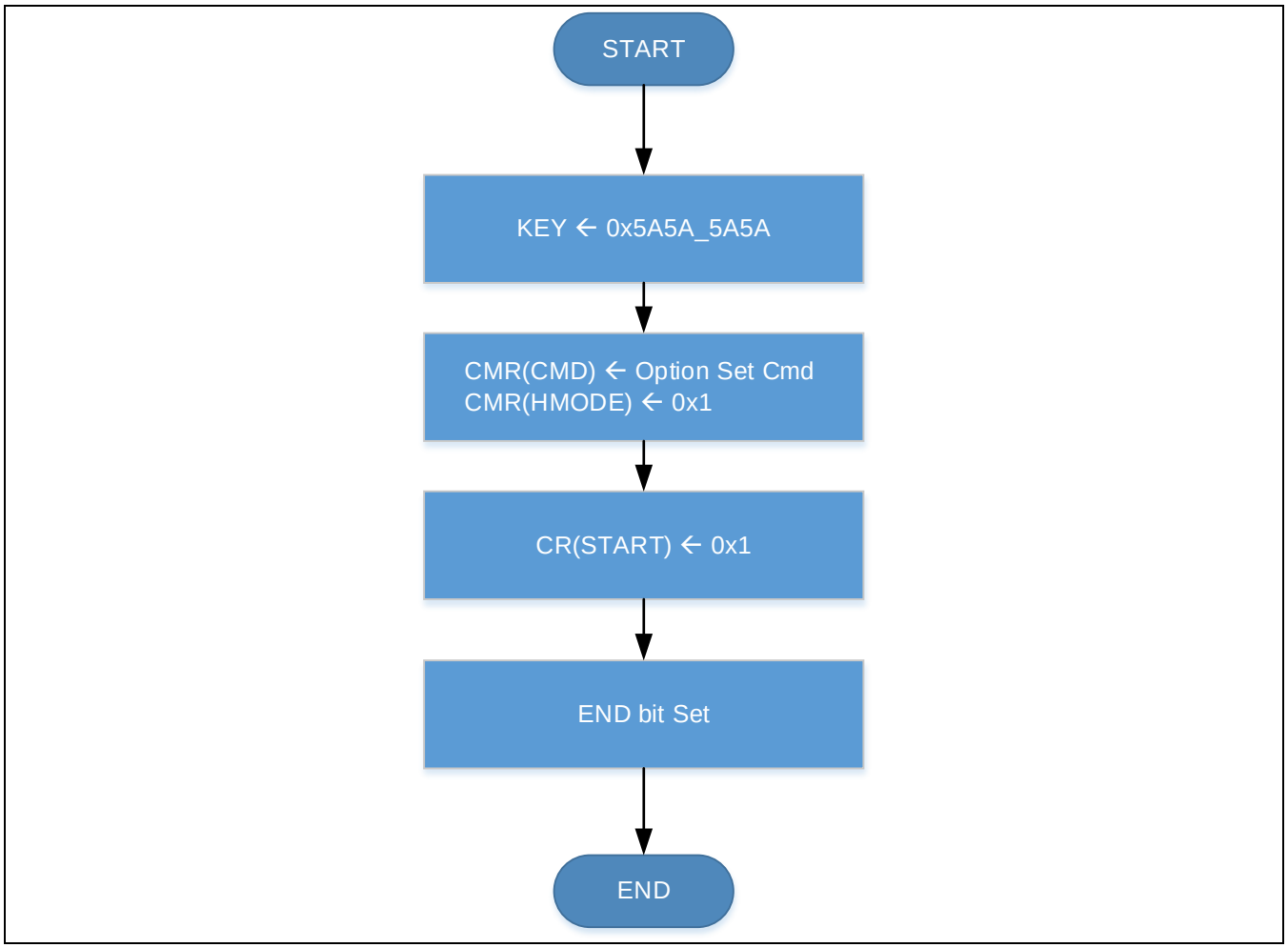


Figure 6-7 自定义选项写操作流程图

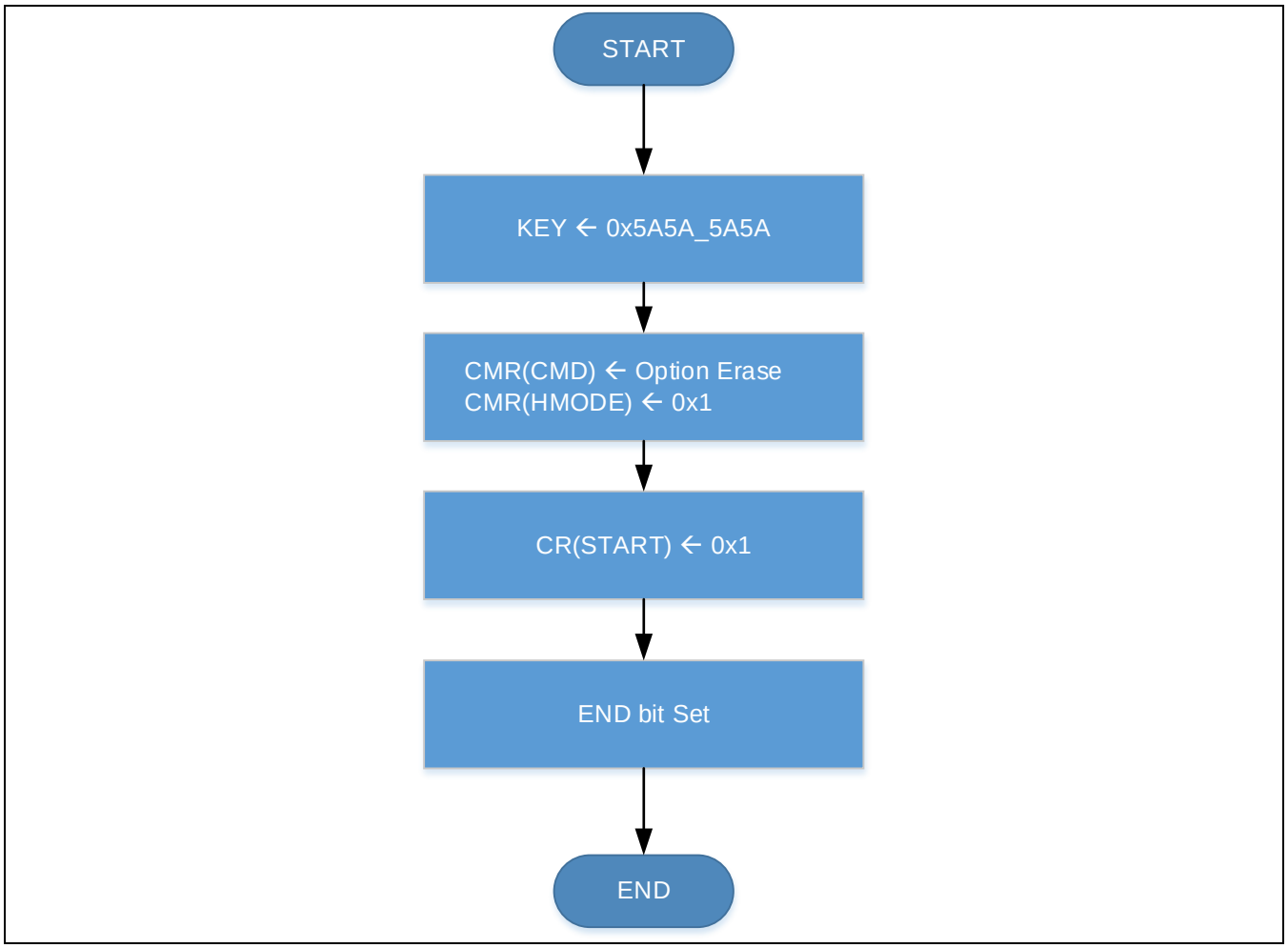


Figure 6-8 自定义选项擦除操作流程图

6.3 寄存器说明

6.3.1 寄存器表

- Base Address: 0x4001_1000

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

6.3.2 IFC_IDR (ID寄存器)

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

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

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

6.3.3 IFC_CEDR (时钟使能/禁止寄存器)

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

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

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

6.3.4 IFC_SRR (软件复位寄存器)

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

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

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

6.3.5 IFC_CMR (指令寄存器)

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

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

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

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

6.3.6 IFC_CR (控制寄存器)

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

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

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

6.3.7 IFC_MR (工作模式寄存器)

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

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

Name	Bit	Type	Description
WAIT	[2:0]	RW	**闪存读等待周期** 0: 闪存读取中等待 0 个周期 n: 闪存读取中等待 n 个周期 注意： 1. 工作频率在 0~25MHz 时，使用 0 等待周期 (APT32F003x 系列) 2. 工作频率在 25~48MHz 时，使用 1 个等待周期

6.3.8 IFC_FM_ADDR (ISP地址寄存器)

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

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

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

6.3.9 IFC_FM_DR (ISP数据寄存器)

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

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

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

6.3.10 IFC_KR (ISP秘钥寄存器)

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

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

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

6.3.11 IFC_ICR (中断控制寄存器)

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

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

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

			0: 禁止中断 1: 使能中断

6.3.12 IFC_RISR (中断原始状态寄存器)

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

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

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

			0: 该状态没有发生 1: 该状态发生

6.3.13 IFC_MISR (中断状态寄存器)

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

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

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

6.3.14 IFC_ICLR (中断状态清除寄存器)

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

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

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

			0: 无效 1: 清除中断

7 系统控制器（SYSCON）

7.1 概述

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

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

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

7.1.1 特性

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

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

7.2 功能描述

7.2.1 时钟管理和控制

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

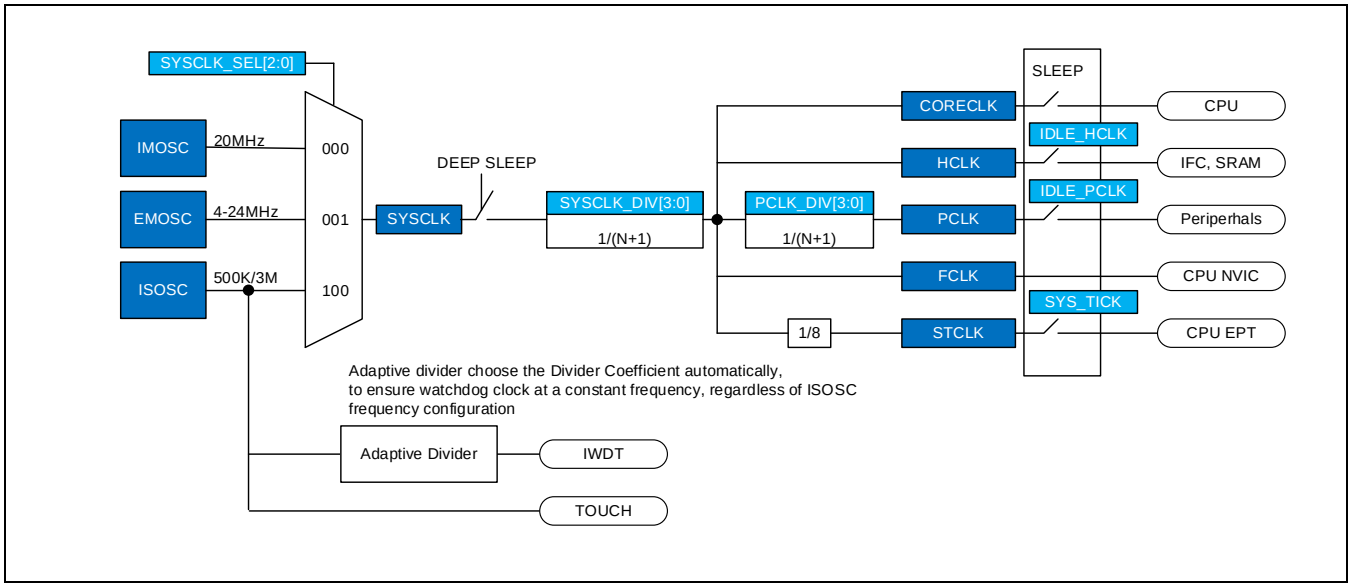


Figure 7-1 时钟结构示意图

NOTE:

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

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

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

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

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

7.2.2 工作时钟切换

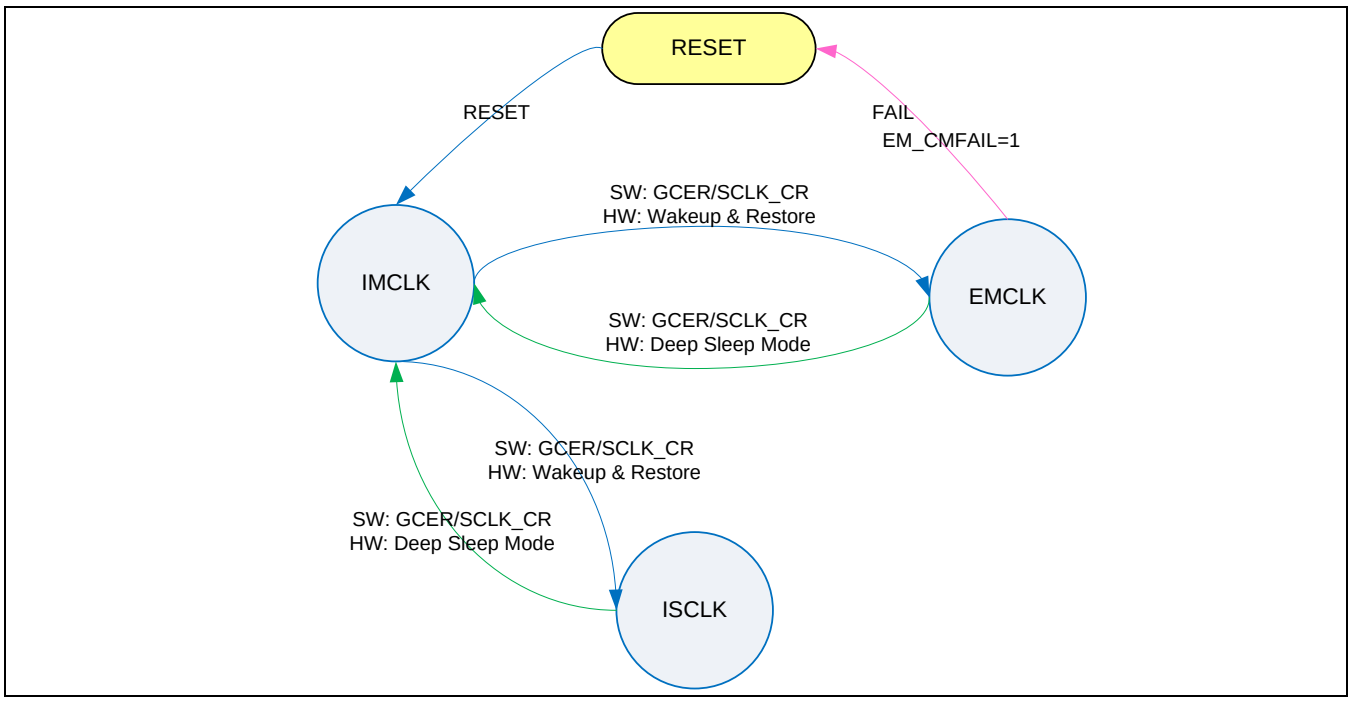


Figure 7-2 时钟切换状态机

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

当芯片执行到 STOP 指令（工作模式切换到 DEEP-SLEEP 模式）时，当前的时钟配置将会被自动保存，然后系统硬件自动切换系统时钟到 IMCLK，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），在完成所有初始化后，IMOSC 会自动停止。芯片自此进入 DEEP-SLEEP 模式。DEEP-SLEEP 的初始化过程根据系统时钟设置的不同会有差异，如果 IMOSC 在 DEEP-SLEEP 进入前没有被关闭，则初始化的时间为35个系统时钟周期（在 IMCLK 为20MHz 的情况下，大约为 1.8us），IMOSC 在 DEEP-SLEEP 进入前是被关闭的，则初始化过程将加入 IMOSC 的使能和稳定时间（初始化总共消耗的时间大约为4.6us）。一旦芯片被唤醒，IMOSC 将作为缺省时钟首先工作，然后再 IMCLK 的控制下，系统会自动恢复 DEEP-SLEEP 前保存的时钟配置。整个唤醒的过程包括了8000个 IMCLK Cycle(380us 左右)的内部电源稳定时间和 IMOSC 稳定时间，以及4.6us 的系统时钟配置恢复时间。所有的这些 DEEP-SLEEP 前后的时钟切换都由硬件自动完成，对用户程序是完全透明的。

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

7.2.3 外部主时钟（EMOSC，EMCLK）

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

Table 7-1 CYOSC_GM[2:0] 设置说明

EMOSC 频率	CYO_GM[2:0]
16MHz	111
10MHz	
8MHz	
4MHz	
1MHz	
500KHz	
32.768KHz	000

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

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

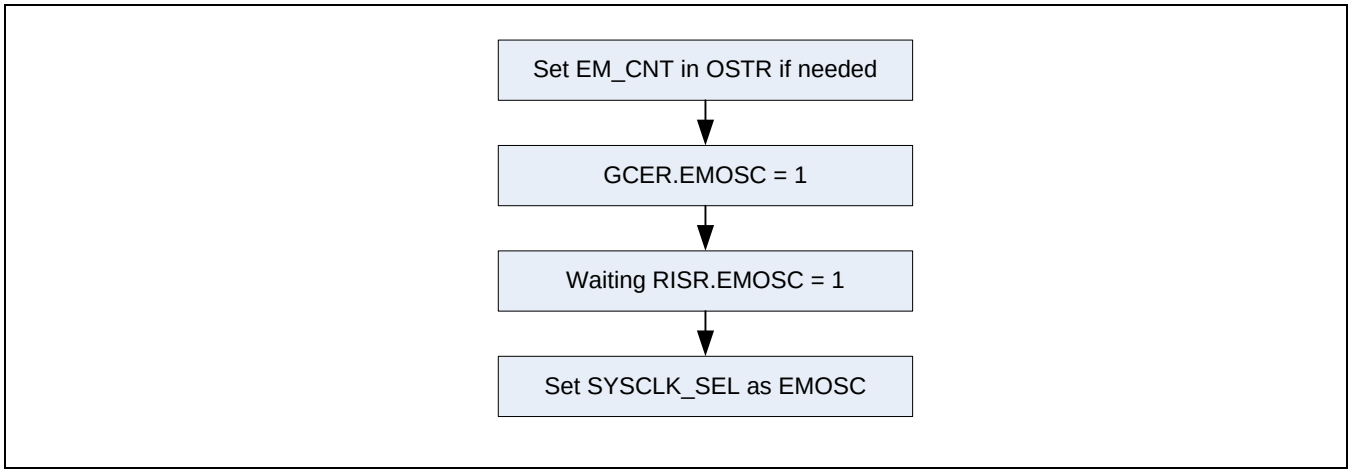


Figure 7-3 EMOSC 使能和系统时钟选择设置流程

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

7.2.4 内部主时钟振荡器（IMOSC，IMCLK）

芯片内部有一个高速的 RC 振荡器，可以作为系统工作的主要时钟源。当 IMOSC 使能时，GCSR 寄存器的 IMOSC 状态标志位将始终为‘1’。在芯片上电后复位以后，IMOSC 是芯片的缺省工作时钟。由于 IMOSC 的稳定时间非常短（相比较于外部晶振），所以系统从 DEEP-SLEEP 或者 POR 以后进入正常工作模式的时间可以大幅的缩短，有效的提升系统响应时间。

如果当前系统工作时钟选择 IMOSC 作为系统时钟源，那么在芯片 SLEEP 模式下，IMOSC 不会改变工作状态。在芯片进入 DEEP-SLEEP 模式后，IMOSC 会自动停止工作，并在系统被唤醒以后，被硬件自动使能。

7.2.5 内部副时钟振荡器（ISOSC，ISCLK）

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

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

7.2.6 外部时钟可靠性监测

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

外部时钟失效监测通过设置 GCER 寄存器中的 EM_CM 位来使能。当失效被检测到，可以通过设置 CMRST 位来使能自动产生系统的复位。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）
- 标志位置位 （在 RISR 寄存器中的 EM_CMFAIL 位置位）

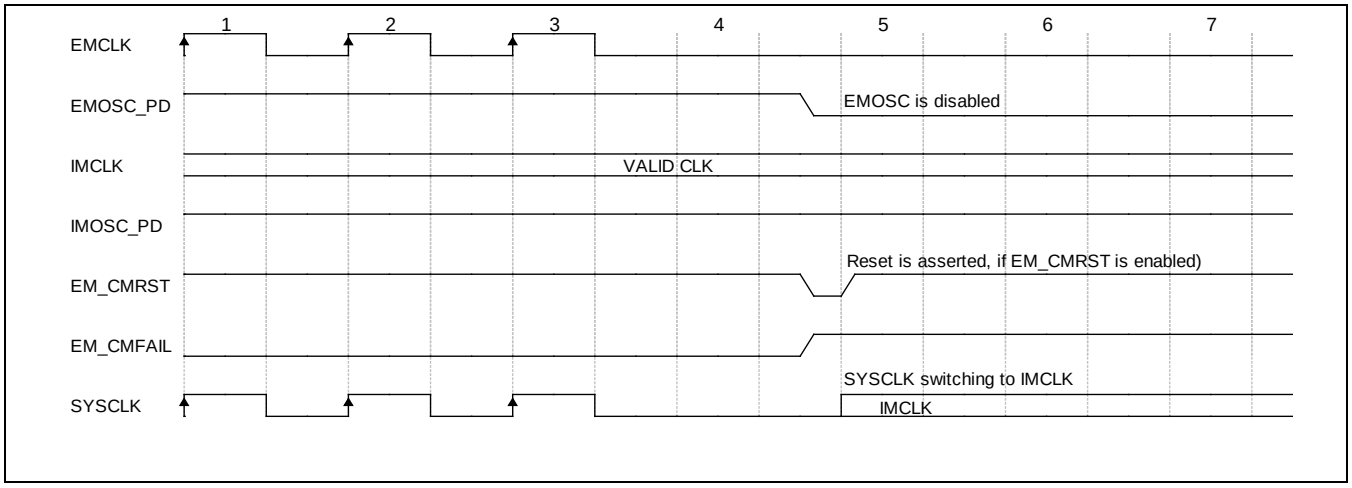


Figure 7-4 EMOSC 失效监测

外部时钟失效监测需要在 EMOSC 稳定以后才可以使能，EM_CMRCV 中断表示 EMOSC 的时钟失效已经恢复。由于外部时钟失效而引发的系统复位，不会清除 EM_CMFAIL 标志（此标志位为异常事件，所以必须软件强制清除，但是断电后上电复位，外部复位，低电压复位和看门狗复位可以清除这个标志）。在系统复位以后，如果 EM_CMRCV 标志已经置位，软件可以继续尝试恢复到 EMOSC 作为系统时钟工作。恢复的工作可以参考如下顺序进行：

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

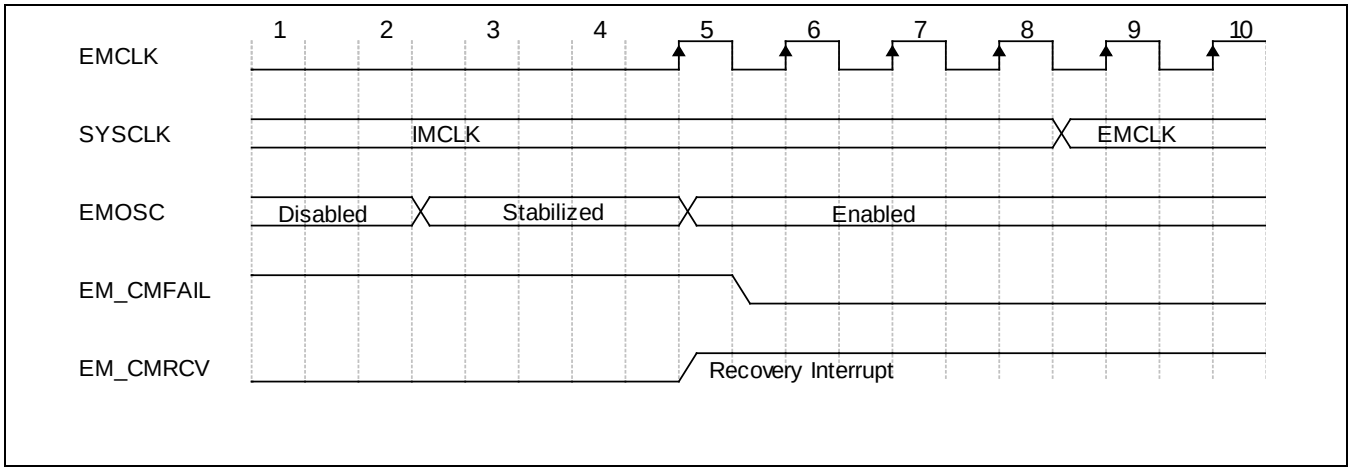


Figure 7-5 EMOSC 从 CM Fail 恢复以后重新使能

7.2.7 功耗管理

芯片支持三种不同的工作模式：普通（RUN）、睡眠（SLEEP）、深睡眠（DEEP-SLEEP）。针对不同的工作模式，可以设置不同的功耗调整策略。

7.2.7.1 RUN 模式

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

7.2.7.2 SLEEP 模式

在 SLEEP 模式下，系统控制器将挂起 CORE 模块的时钟（CPU 将不会工作）。缺省模式下，所有的逻辑外设仍旧可以工作，但是可以通过设置 IDLE_PCLK 位来修改 SLEEP 模式下，PCLK 是否挂起。所有振荡器的工作状态在该模式下不会做任何改变。

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

SLEEP 模式可以在系统应用需要快速唤醒，并对功耗有一定要求的应用中采用。

7.2.7.3 DEEP-SLEEP 模式

在 DEEP-SLEEP 模式下，系统控制器将挂起所有的时钟源，但是内部逻辑电源仍将保持不变。在 IWDT 或 TOUCH 模块使能前提下，ISOSC 可以在该模式下继续工作。IMOSC 和 EMOSC 在进入 DEEP-SLEEP 模式以后，会被自动关闭。当处理器从 DEEP-SLEEP 模式退出时，系统工作时钟（SYSCLK）将会自动恢复到之前的状态。外部 IO，IWDT，TOUCH 等外部中断或事件可以唤醒处理器，并从该模式退出。具体可以参考中断章节的说明。

7.2.8 外部供电监测 （LVD）

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

LVD 在芯片上电后默认为打开状态，如果需要关闭 LVD 功能，可以置高 LVDCR 的 LVD 控制位 LVD_OFF 关闭 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。当中断使能时（通过 LVD_INT 控制位设置），处理器会在外部供电电压低于 INTDET_LVL 设置值时，产生中断请求（IER、IDR 寄存器中的 LVD_INT 位可以设置或者清除中断标志）。当前外部供电电压的状态，可以通过 LVDCR 的 LVDFLAG 位检测到。当外部供电电压低于检测 level 时，该标志位为‘1’，当高于检测 level 时，该标志位为‘0’。

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。为保证 LVD 的精度，LVD 模块在使能时会消耗额外的功耗。

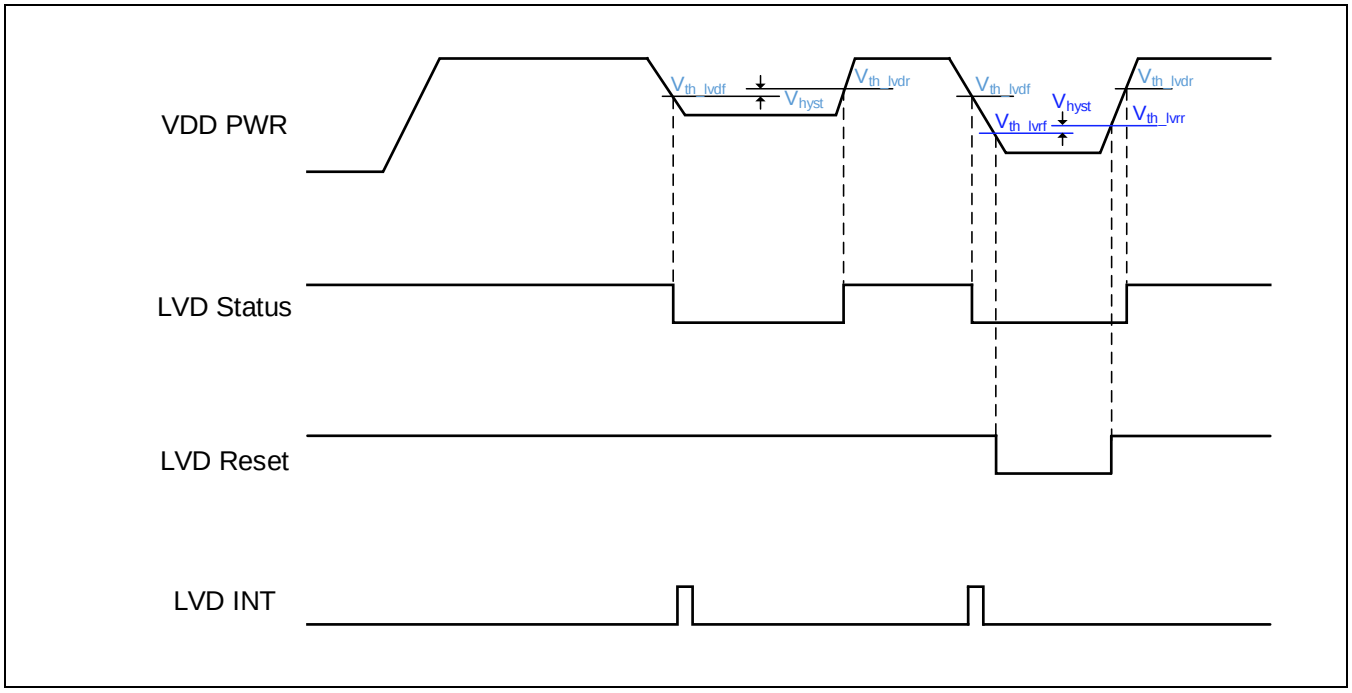


Figure 7-6 LVD 工作时序图

7.2.9 复位管理 （RESET ID）

处理器内嵌一个复位历史纪录控制器，专门用于记录引起系统复位的 RESET 源。下表中描述了处理器的所有可能的复位信号源。

Table 7-2 处理器复位信号源表

信号名	描述
EXTRST	外部输入的硬件 RESET 信号（低电平有效）。在外部复位脚有效时可用。
CMRST	由内部产生的 EMOSC 时钟异常复位信号。可以通过软件使能或关闭该功能。
LVDRST	由低电压监测模块（LVD）产生的系统复位信号。可以通过软件势能或者关闭该功能。
IWDTRST	由内部看门狗电路产生的复位信号。
SWRST	由系统控制器产生的软件复位信号。(IDCCR 寄存器中的 SWRST 控制位)
SYSRSTREQ	由 CPU 产生的系统复位请求（通过 MTCR 指令对 CPU 中的 SRCR 寄存器写入 0xABCD1234）。
POR	上电复位

每一个复位信号都对应 RSR 寄存器中的一个状态位。可以通过软件读取该寄存器鉴别处理器的复位信号源。该寄存器中的所有位信息在电源上电复位（POR）以后，会自动清除。有效的复位可以在复位成功后清除上次的历史记录，并记录当前复位的触发源。

7.2.10 外部中断管理 （EXI）

处理器的所有 GPIO 都可以设置为外部中断输入。在该芯片中，最多有14个外部中断信号线。同一个组的管脚可以被配置为该组中断信号线的触发源。外部中断管脚分组的定义如下图所示。详细的管脚设置和分组设置可以参考 GPIO 章节。系统控制器内部，负责对外部中断信号线的中断使能控制。通过 EXIER/EXIDR 寄存器可以使能或者关闭指定的外部中断信号线。外部中断的触发边沿选择，可以通过 EXIRT 和 EXIFT 寄存器进行设置。

Table 7-3 外部中断源分组

EXI 中断信号线	中断源
EXI0	PA0.0 or PB0.0 or PC0.0
EXI1	PA0.1 or PB0.1 or PC0.1
EXI2	PA0.2 or PB0.2 or PC0.2
EXI3	PA0.3 or PB0.3 or PC0.3
EXI4	PA0.4
EXI5	PA0.5
EXI6	PA0.6
EXI7	PA0.7
EXI8	PA0.8
EXI9	PA0.9
EXI10	PA0.10
EXI11	PA0.11

EXI12	PA0.12
EXI13	PA0.13

7.2.11 独立的看门狗定时器 （IWDT）

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

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

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

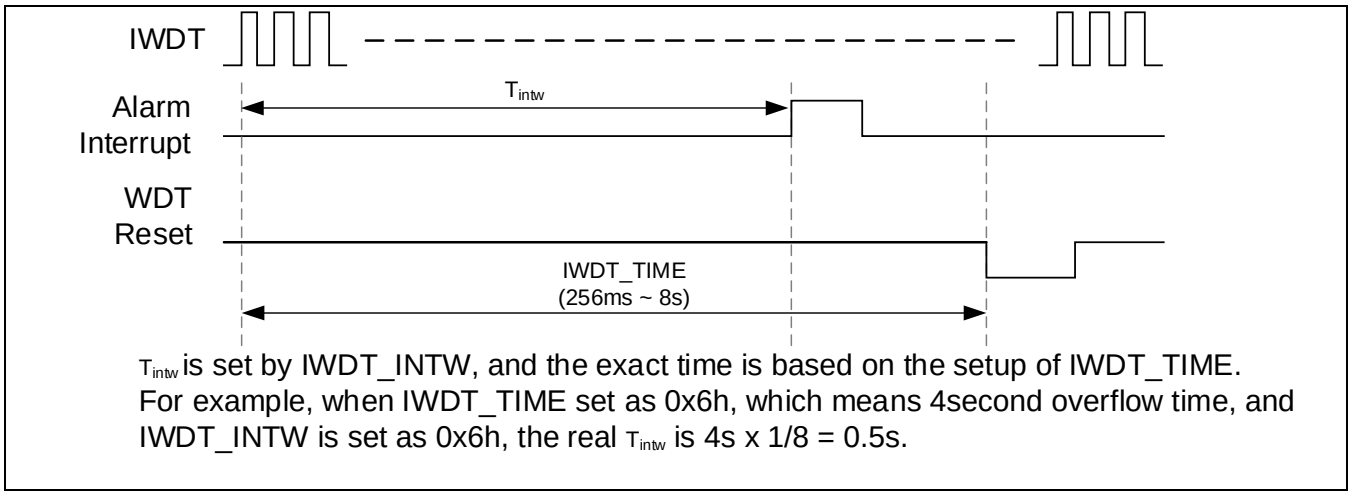


Figure 7-7 IWDT 溢出和中断间隔

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

7.2.12 错误命令处理

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

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

7.3 寄存器说明

7.3.1 寄存器表

- Base Address: 0x4001_2000

Table 7-4 寄存器表

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

Register	Offset	Description	Reset Value
SYSCON_ICR	0x078	中断清除寄存器	0x0000_0000
SYSCON_RISR	0x07C	原始中断标志状态寄存器	0x0000_0000
SYSCON_ISR	0x080	中断标志状态寄存器	0x0000_0000
SYSCON_RSR	0x084	复位记录状态寄存器	-
SYSCON_EXIRT	0x088	外部中断上升沿选择寄存器	0x0000_0000
SYSCON_EXIFT	0x08C	外部中断下降沿选择寄存器	0x0000_0000
SYSCON_EXIER	0x090	外部中断使能寄存器	0x0000_0000
SYSCON_EXIDR	0x094	外部中断禁止寄存器	0x0000_0000
SYSCON_EXIMR	0x098	外部中断使能/禁止状态寄存器	0x0000_0000
SYSCON_EXIAR	0x09C	外部中断软件触发寄存器	0x0000_0000
SYSCON_EXICR	0x0A0	外部中断清除寄存器	0x0000_0000
SYSCON_EXIRS	0x0A4	外部中断原始标志状态寄存器	0x0000_0000
SYSCON_IWDCR	0x0A8	看门狗控制寄存器	0x0000_070C
SYSCON_IWDCNT	0x0AC	看门狗控制计数器值	0x0003_FFFF
SYSCON_IWDEDR	0x0B0	看门狗使能寄存器	0x0000_XXXX
SYSCON_CINF0	0x0B4	客户信息区0[3]	-
SYSCON_CINF1	0x0B8	客户信息区1	-
SYSCON_FINF0	0x0BC	工程信息区0[4]	-
SYSCON_FINF1	0x0C0	工程信息区1	-
SYSCON_ERRINF	0x0E0	错误命令信息查询寄存器	0x0000_0000
SYSCON_SFCR	0x200	特殊功能寄存器	0x0000_0078

NOTE:

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

7.3.2 SYSCON_IDCCR (ID 和控制器模块时钟控制寄存器)

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

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

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

7.3.3 SYSCON_GCER/GCDR (通用使能/禁止控制寄存器)

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

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

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

NOTE:

GCER 和 GCDR 寄存器只有对写入的‘1’敏感，写入‘0’时无效。

7.3.4 SYSCON_GCSR (通用状态寄存器)

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

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

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

7.3.5 SYSCON_SCLKCR (系统时钟控制寄存器)

- Address = Base Address + 0x001C, Reset Value = 0x0000_0800

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

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

7.3.6 SYSCON_PCLKCR (外设时钟控制寄存器)

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

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

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

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

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

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

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

7.3.8 SYSCON_PCSR0 (外设时钟状态寄存器0)

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

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

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

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD					GPIOC0	GPIOB0	GPIOA0	RSVD							LED	RSVD		CNTA	RSVD	GTC3	GTC2	GTC1	GTC0	RSVD							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	W	W	W	R	R	R	R	R	R	R	W	R	R	W	R	W	W	W	W	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
GTC0 GTC1 GTC2 GTC3 CNTA LED GPIOA0 GPIOB0 GPIOC0	[-]	W	使能/禁止相应外设模块的 PCLK 时钟。 只有对相应位写‘1’时才有效，写‘0’时无效 PCER 相应位写‘1’时，使能相应模块 PCLK 时钟， PCDR 相应位写‘1’时，禁止相应模块 PCLK 时钟。

7.3.10 SYSCON_PCSR1 (外设时钟状态寄存器1)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD					GPIOC0	GPIOB0	GPIOA0	RSVD							LED	RSVD		CNTA	RSVD	GTC3	GTC2	GTC1	GTC0	RSVD							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	Type	Description
GTC0 GTC1 GTC2 GTC3 CNTA LED GPIOA0 GPIOB0 GPIOC0	[-]	R	相应外设模块的 PCLK 时钟的使能/禁止状态。 0：该对应模块的时钟被禁止。 1：该对应模块的时钟被使能。

7.3.11 SYSCON_OSTR (外部振荡器稳定时间配置寄存器)

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

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

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

7.3.12 SYSCON_LVDCR (低电压检测控制寄存器)

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

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

Name	Bit	Type	Description
LVD_OFF	[0]	R/W	使能/禁止 LVD 模块。 0：使能 LVD 模块。 1：禁止 LVD 模块。
INTDET_LVL	[10:8]	R/W	LVD 中断触发检测电平 **INTDET_LVL[2:0]** / **VDD(v)** x00 / 2.5 x01 / 3.0 x10 / 3.9 x11 / 4.1
LVD_INT	[11]	R/W	使能/禁止 LVD 中断功能。 0：LVD 中断禁止。 1：LVD 中断使能。 在需要产生 LVD 中断时，还必须使能中断使能寄存器 IECR 中的 LVD_INT 位。
RSTDET_LVL	[14:12]	R/W	LVD 复位触发检测电平 **RSTDET_LVL[2:0]** / **VDD(v)** x00 / 2.2 x01 / 2.7 x10 / 3.3 x11 / 3.6
LVDFLAG	[15]	R/W	LVD 的检测状态。

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

NOTE:

1. 由 LVD 模块产生的处理器复位，不会复位 LVD 控制寄存器。
2. 从芯片 E 版本开始，芯片上电后 LVD 默认为打开状态，将 LVD_OFF 位写1会禁止 LVD 模块。

7.3.13 SYSCON_IMFT (内部主振荡器（IMOSC）精调寄存器)

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

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

Name	Bit	Type	Description
IMO_TUNE	[8:0]	R/W	IMOSC 频率的微调设置。 通过 IMO_TUNE 控制位，可以精细调整 IMOSC 的输出频率。调整的步进大约为12KHz。 0x1FF：标准输出频率（工程校准后的输出值）。 0x1FE：标准输出频率-12KHz。 0x1FD：标准输出频率-24KHz。 0x1FC：标准输出频率-36KHz。 ……
CLOMX	[19:16]	R/W	CLO 输出选择。

CLOMX[3:0]	CLOCK
0x0	ISCLK
0x1	IMCLK
0x2	RSVD
0x3	EMCLK
0x4	TKEYCLK
0x5	TKEYCLK_DV
0x6	IWDTCLK
0x7	SYSCLK
0x8	FRNCLK
0x9	DAPCLK
0xA	CPUCLK
0xB	AHBCLK
0xC	APBCLK
Others	RSVD

7.3.14 SYSCON_PWRCR (功耗控制寄存器)

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

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

Name	Bit	Type	Description
STOP_CFG	[0]	R/W	在 DEEP-SLEEP 模式下功耗控制。 0：驱动负载能力较强，功耗较大。 1：驱动负载能力较弱，功耗小。
IDLE_CFG	[1]	R/W	在 SLEEP 模式下功耗控制 0：驱动负载能力较强，功耗较大。 1：驱动负载能力较弱，功耗小。
SIDLE_CFG	[2]	R/W	在低速 SLEEP 模式下功耗控制 0：驱动负载能力较强，功耗较大。 1：驱动负载能力较弱，功耗小。
SUBOPT	[3]	R/W	低速模式选项。 0：普通模式 1：低速模式（当 ISOSC 作为系统工作时钟时）

7.3.15 SYSCON_OPT1 (系统配置寄存器1)

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

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

Name	Bit	Type	Description
IMOSC_TRM	[14:8]	R/W	IMOSC 的输出频率调整。 在上电时，FLASH 内的工厂校准值会被自动加载到此寄存器中。通过调整此寄存器，程序可以再次校准 IMOSC 的输出。

7.3.16 SYSCON_OPT0 (系统配置寄存器0)

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

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

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

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

- IECR：Address = Base Address + 0x0068, Reset Value = 0x0000_0000
- IEDR：Address = Base Address + 0x006C, Reset Value = 0x0000_0000
- IAR：Address = Base Address + 0x0074, Reset Value = 0x0000_0000
- ICR：Address = Base Address + 0x0078, Reset Value = 0x0000_0000

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

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

NOTE:

1） 寄存器只有在写入‘1’时有效，写入‘0’时无效。

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

3） IAR 在对应位写入‘1’时，对应中断被触发。

4） ICR 在对应位写入‘1’时，对应中断状态标示被清除。

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

- IMSR：Address = Base Address + 0x0070, Reset Value = 0x0000_0000
- RISR：Address = Base Address + 0x007C, Reset Value = 0x0000_0000
- ISR：Address = Base Address + 0x0080, Reset Value = 0x0000_0000

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

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

NOTE:

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

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

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

7.3.19 SYSCON_RSR (复位记录状态寄存器)

- Address = Base Address + 0x0084, Reset Value = 0xXXXX_XXXX

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

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

NOTE:

对相应位写‘1’来清除。

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

- EXIRT：Address = Base Address + 0x0088, Reset Value = 0x0000_0000
- EXIFT：Address = Base Address + 0x008C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
EXI0 ~ EXI13	[13:0]	R/W	外部中断上升沿/下降沿使能。 0：上升沿（EXIRT）/ 下降沿（EXIFT）未使能。 1：上升沿（EXIRT）/ 下降沿（EXIFT）使能。

NOTE:

1） EXIRT 是上升沿选择寄存器。

2） EXIFT 是下降沿选择寄存器。

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

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

- EXIER：Address = Base Address + 0x0090, Reset Value = 0x0000_0000
- EXIDR：Address = Base Address + 0x0094, Reset Value = 0x0000_0000
- EXIAR：Address = Base Address + 0x009C, Reset Value = 0x0000_0000
- EXICR：Address = Base Address + 0x00A0, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
EXI0 ~ EXI13	[13:0]	R/W	IER/IDR：使能/禁止外部中断 IAR/ICR：软件触发/清除外部中断

NOTE:

1） 寄存器只有在写入‘1’时有效，写入‘0’时无效。

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

3） EXIAR 在对应位写入‘1’时，对应外部中断被触发。

4） EXICR 在对应位写入‘1’时，对应外部中断状态标示被清除。

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

- EXIMR：Address = Base Address + 0x0098, Reset Value = 0x0000_0000
- EXIRS：Address = Base Address + 0x00A4, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description
EXI0 ~ EXI13	[13:0]	R/W	EXIMR：外部中断使能、禁止状态寄存器。 0：该外部中断被禁止。 1：该外部中断被使能。 EXIRS：外部中断原始标志位 0：该中断未发生。 1：该中断发生。

7.3.23 SYSCON_IWDCR（看门狗控制寄存器）

- Address = Base Address + 0x00A8, Reset Value = 0x0000_070C

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

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

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

NOTE:

在看门狗工作时，任何尝试关闭 ISOSC 的操作将导致命令错误中断发生。

7.3.24 SYSCON_IWDCNT（看门狗控制计数器值）

- Address = Base Address + 0x00AC, Reset Value = 0x0003_FFFF

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

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

7.3.25 SYSCON_IWDEDR（看门狗使能寄存器）

- Address = Base Address + 0x00B0, Reset Value = 0x0000_XXXX

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

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

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

- CINF0：Address = Base Address + 0x00B4, Reset Value = 0xXXXX_XXXX
- CINF1：Address = Base Address + 0x00B8, Reset Value = 0xXXXX_XXXX
- FINF0：Address = Base Address + 0x00BC, Reset Value = 0xXXXX_XXXX
- FINF1：Address = Base Address + 0x00C0, Reset Value = 0xXXXX_XXXX

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

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

7.3.27 SYSCON_ERRINF（错误命令信息查询寄存器）

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

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

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

7.3.28 SYSCON_SFCF（特殊功能寄存器）

- Address = Base Address + 0x0200, Reset Value = 0x0000_0078

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

Name	Bit	Type	Description
CYO_CD	[2]	R/W	外部晶振驱动调整。
CYO_GM	[5:3]	R/W	外部晶振增益调整。
TCHREF	[6]	R/W	电容检测模块充电模式以及检测阈值设置。
SFCR_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 SFCR_KEY 等于0x5AA5时，对本寄存器的写入才有效。

8 模数转换器（ADC)

8.1 概述

本章节描述ADC控制器的功能，从用户的角度详细说明如何操作ADC。

8.1.1 主要特性

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

- 带逐次逼近逻辑的模拟比较器
- 支持多路输入AIN[14:4], [2:0]
- 12位转换结果寄存器(ADC_DR)
- 最大转换速度: 12位500KSPS，10位1MSPS
- 模拟输入范围：AVSS 到 AVREF (VDD)
- 可配置采样时间，并且可以配置成10位的ADC以加快转换速度

8.1.2 管脚描述

Table 8-1 ADC管脚描述

管脚名称	功能	I/O类型	有效电平	说明
AVREF	模拟参考电压	模拟	-	-
AIN0 to AIN2, AIN4 to AIN14	模拟信号输入	模拟	-	-

8.1.3 模块框图

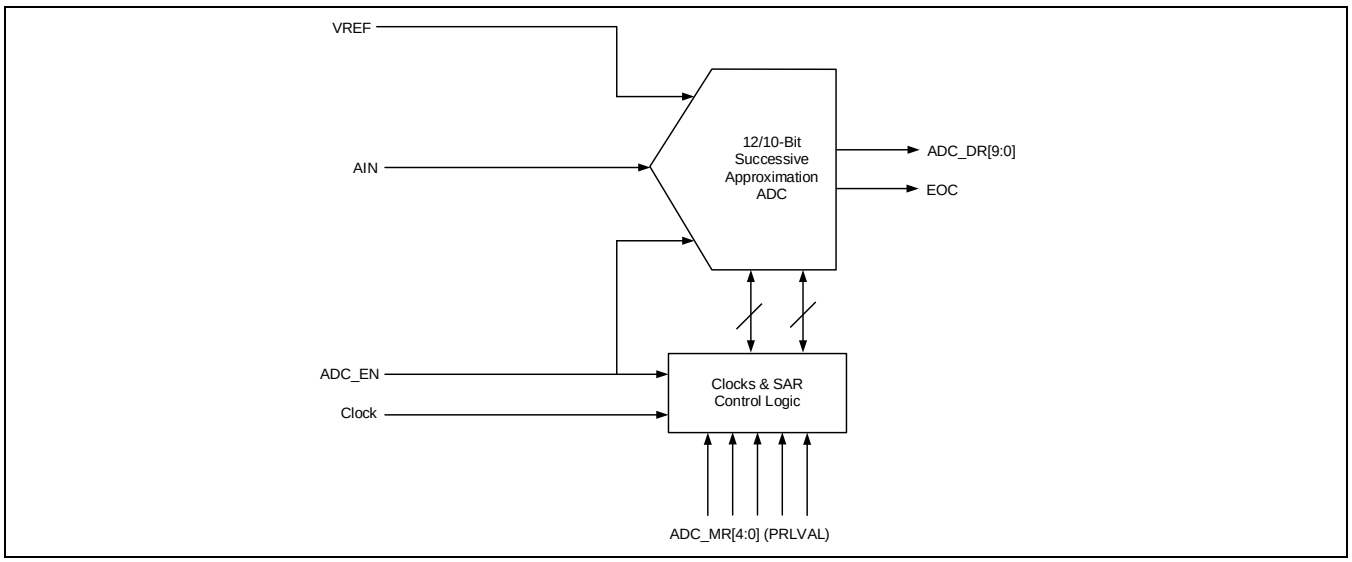


Figure 8-1 ADC模块框图

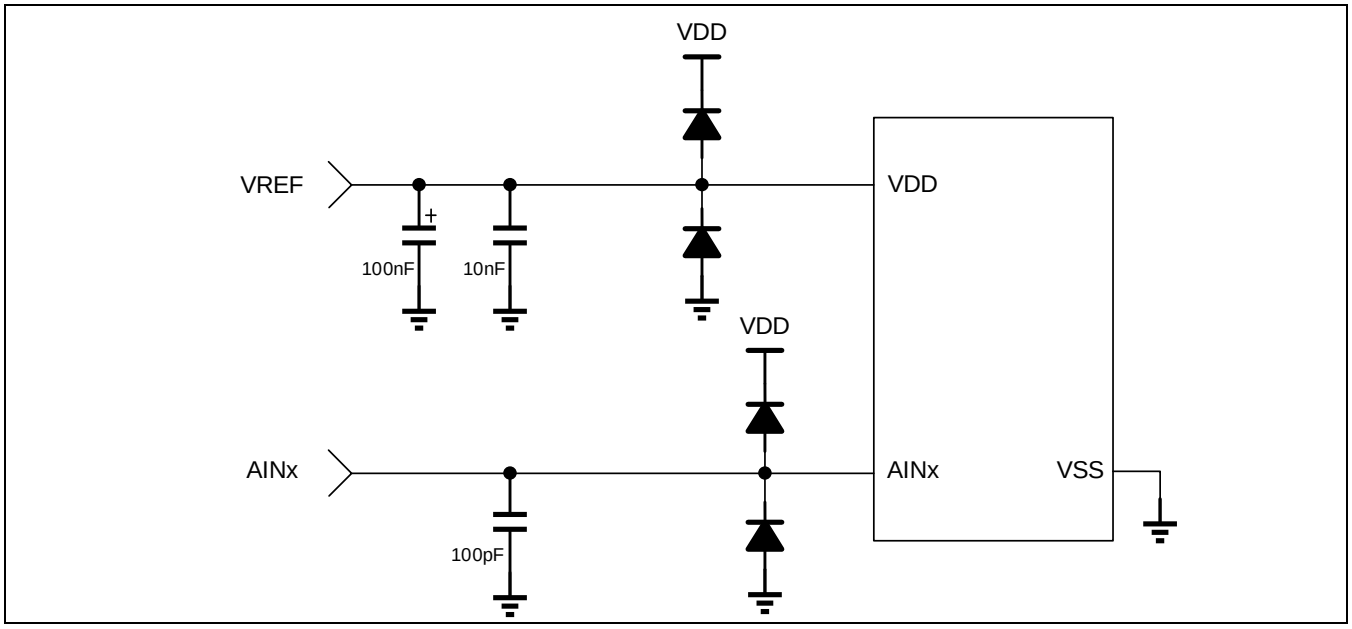


Figure 8-2 参考电路

8.1.4 输入和输出

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

Input Voltage Range: 0.0 V ~ 5.0 V

Reference Bottom Voltage: 0.0 V

Reference Top Voltage: 5.0 V

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

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

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

Table 8-2 12位模式的输入和输出范围 (VREF = 5V)

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

Table 8-3 10位模式的输入和输出范围 (VREF = 5V)

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

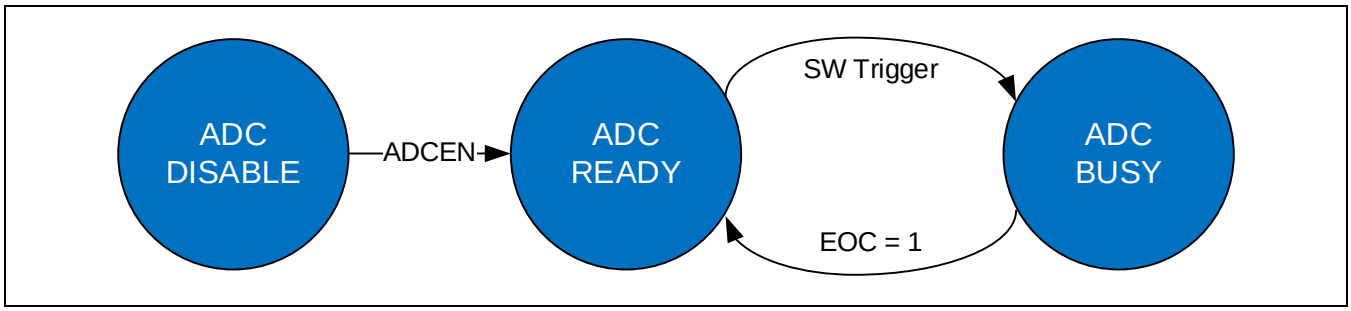


Figure 8-3 ADC状态机

8.1.5 时钟频率和转换时间

ADC 工作的时钟是从 PCLK 获得的。AD 转换的过程需要总共(setup+12/10)个时钟周期。setup 时间可以通过 ADC_CR 寄存器里的“SAMPLE”位(ADC_CR[6:5])设置。ADC 模块提供一个时钟分频器，该分频器是一个 6 位计数器，由模式寄存器里的 PRLVAL 控制。 下面的表达式给出了系统频率和 ADC 模拟模块时钟频率之间的关系。

如果 PRLVAL 是 0，那么 F_ANA = PCLK

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

PRLVAL 的值必须保证采样速度不超过手册规定的最大值(12 位 500KSPS，10 位 1MSPS)。如果 PCLK/2 被选择位转换时钟，并且 PCLK 频率是 21MHz，那么一个时钟周期就是 95.2ns。转换速度计算如下(假设 setup 时间为默认值 6 个周期)：

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

21 x 95.2ns = 2us (500ksps)

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

19 x 95.2ns = 1.7us (588ksps)

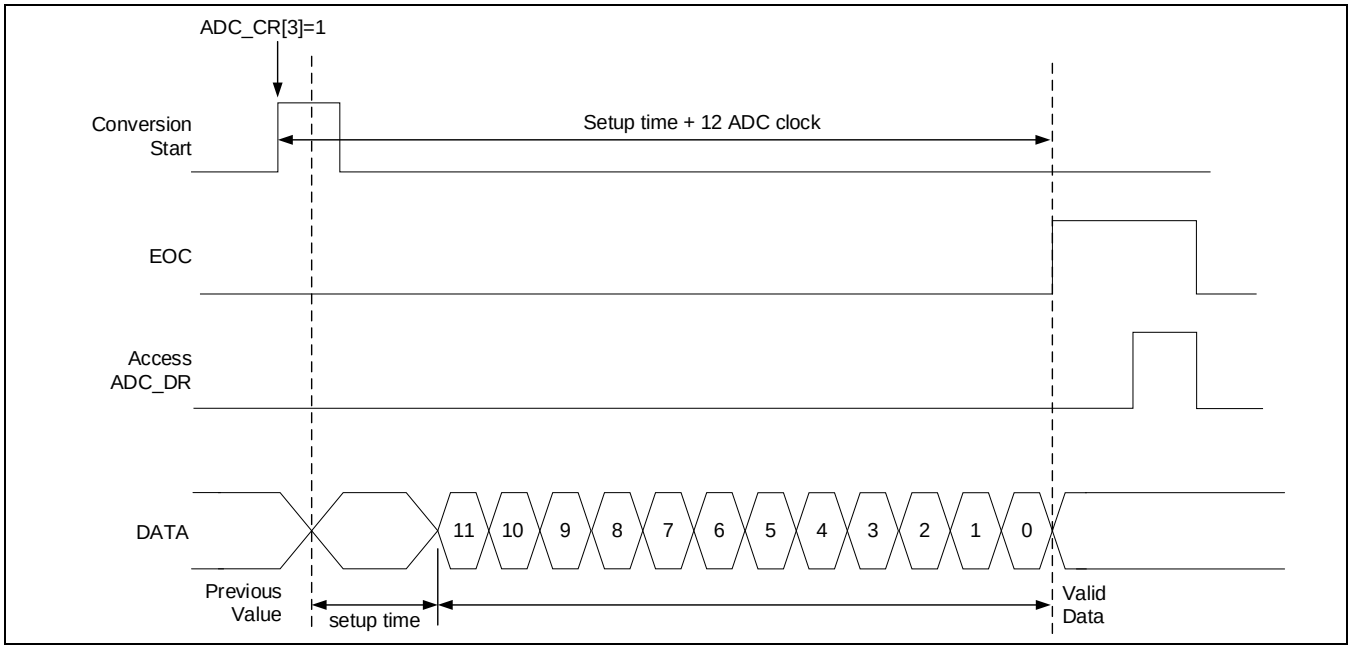


Figure 8-4 ADC工作时序图

8.1.6 转换序列定义

转换序列是指若干个需要转换的模拟输入的组合。用户可以设置ADC模块以任意顺序转换14个通道中的某几个选择好的输入信号。序列的长度(或者说转换的次数)可以由ADC_MR(ADC模式寄存器)中的NBRCH位定义。下表列出了NBRCH和转换次数的关系：

Table 8-4 NBRCH[5:0]的值和转换次数

NBRCH[5:0]	转换次数
00_0000	1
00_0001	2
00_0010	3
00_0011	4
00_0100	5
00_0101	6
00_0110	7
00_0111	8
00_1000	9
00_1001	10
00_1010	11
00_1011	12

要注意的是，即使是在单次转换(one shot)模式下，ADC 也会在启动后转换设置好的转换次数。转换结果寄存器会更新成序列中最后一次转换的结果值。

序列中转换通道的组合由 ADC_CMRx 寄存器设定。CV1 定义转换序列中的第一个通道，CV2 为第二个通道，以此类推。下表列出了 CVx 值和输入通道选择的关系：

Table 8-5 CVx值和输入选择通道

CVx值	选择的输入通道	选择的管脚
00_0000	Input 0	AIN0
00_0001	Input 1	AIN1
00_0010	Input 2	AIN2
00_0011	N/A	N/A
00_0100	Input 4	AIN4
00_0101	Input 5	AIN5
00_0110	Input 6	AIN6
00_0111	Input 7	AIN7
00_1000	Input 8	AIN8
00_1001	Input 9	AIN9
00_1010	Input 10	AIN10
00_1011	Input 11	AIN11

00_1100	Input 12	AIN12
00_1101	Input 13	AIN13
00_1110	Input 14	AIN14

例如，假设：

NBRCH = 0x2,

CV1 = 0x5, CV2 = 0x2 and CV3 = 0x0

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

8.1.7 单次转换或者连续转换模式

ADC 可以配置成两种模式：单次转换模式和连续转换模式。

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

在序列转换完成前，ADC 不可以被停止。

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

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

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

8.1.8 ADC的比较功能

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

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

在 APT32F003 系列中，可以用来比较的阈值寄存器有两个：ADC_CMP0 和 ADC_CMP1。在连续转换模式下，也有两个相应的 NBRCMP0(ADC_MR[21:16])和 NBRCMP1(ADC_MR[27:22])寄存器。

Table 8-6 NBRCMPx[5:0]的值和需要比较的转换次数

NBRCMPx[5:0]	需要比较的转换次数
00_0000	1
00_0001	2
00_0010	3
00_0011	4
00_0100	5
00_0101	6
00_0110	7

00_0111	8
00_1000	9
00_1001	10
00_1010	11
00_1011	12

例如，假设：

NBRCH = 0x4,

CV1 = 0x5, CV2 = 0x2, CV3 = 0x0, CV4 = 0x5, CV5 = 0x2

NBRCMP0 = 0x1, NBRCMP1 = 0x3

ADC_CMP0 = 0x200, ADC_CMP1 = 0x700

(ADC_IER) CMP0H = 1, CMP1L = 1

那么 ADC 会进行 5 次转换。

1. ADC 将 CV2 (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 CV4 (AIN5)的转换结果和 0x700 (ADC_CMP1)比较，如果结果小于 0x700，那么 CMP1L 中断产生。

8.1.9 ADC转换启动的触发源

ADC转换启动可以由3种事件触发：软件启动，定时器1脉冲匹配中断，外部管脚输入。

1. 由CPU软件启动(ADC_MR寄存器中的IES位为0，ADC_CR寄存器中START位写1)。

2. 在ADC_MR寄存器的IES为1的情况下，定时器1的脉冲匹配中断发生后，ADC会开始转换。

3. 在ADC_MR寄存器的IES为1的情况下，外部ADC_ETR管脚的电平发生变化时，ADC会开始转换。ADC_ETR管脚的上升沿或者下降沿或者上升下降沿同时触发，由ADC_MR寄存器中的ETRG位控制。

8.1.10 功耗管理

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

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

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

下表列出了功耗管理的各种状态：

Table 8-7 功耗管理的状态位

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

8.1.11 EOC标志 (End of Conversion)

状态寄存器中的 EOC 位表示转换结果寄存器中有新的值。

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

注意：每次读转换结果寄存器(ADC_DR)都会将 EOC 标志位清零。

8.1.12 Ready标志

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

8.1.13 OVR标志 (转换溢出)

这个标志表示某个转换完成的数据还没有被读取，就被新的数据覆盖了。

OVR 标志可以被 CPU 清除(在状态清除寄存器里写 OVR 位)。

8.1.14 CMPxH/L标志

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

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

8.1.15 工作流程

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

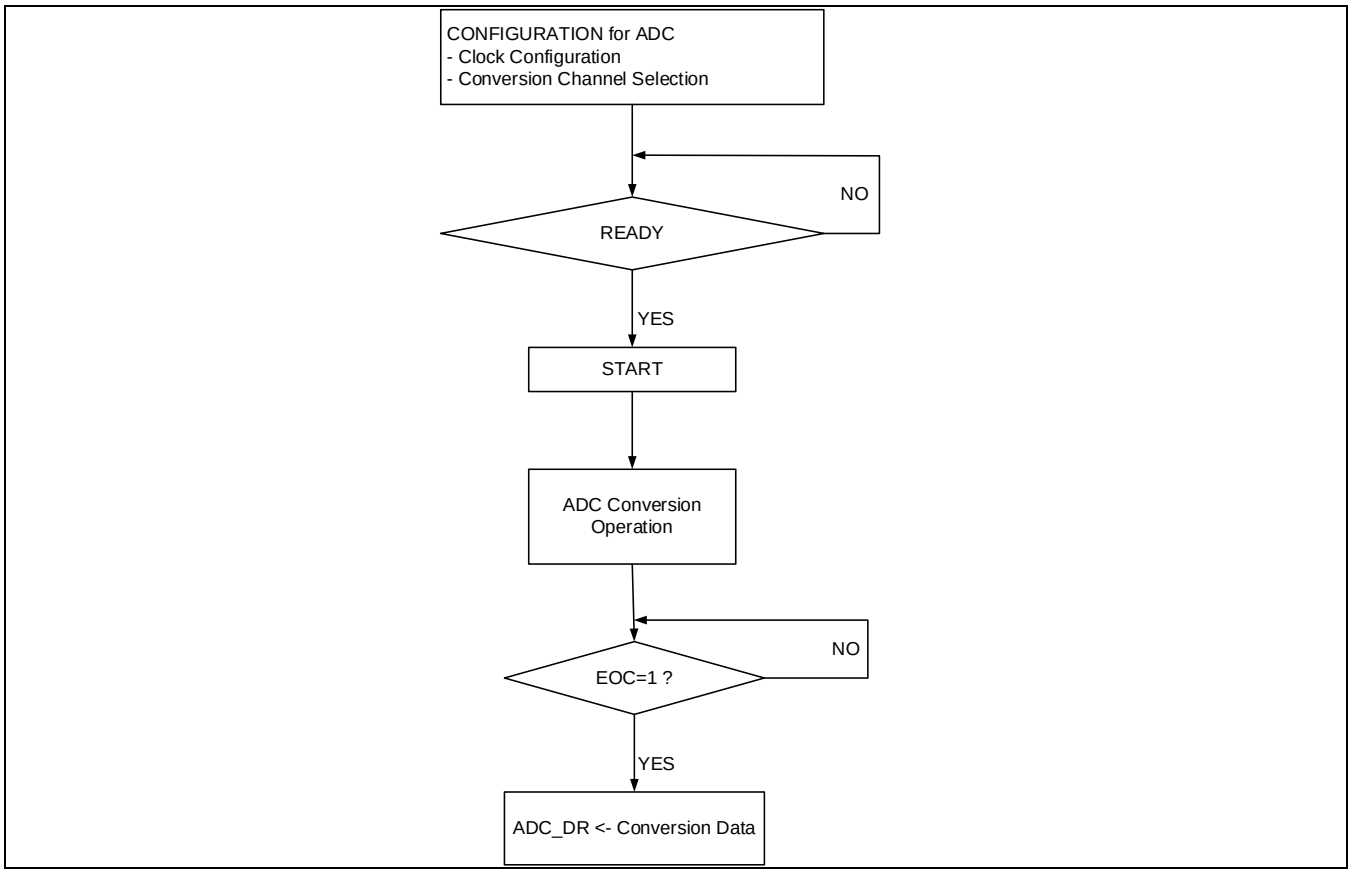


Figure 8-5 ADC工作流程图

8.1.16 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：转换次数(NBRCH)和哪些输入通道需要被转换(ADC_CMRx 中的 CVx)
3. 使能 ADC 模块(ADC_CR 中的 ADC_EN)
4. 等待 ADC_SR 中的 READY 位。只有当这个标志位被置 1 后，ADC 才能正常的开始转换。如果 ADC_IMR 中的相应中断被使能，那么当 READY 标志置起的时候，会产生一个中断
5. 通过写 ADC_CR 中的 START 位，开始转换
6. ADC 选择转换序列中的第一个模拟输入通道
7. 模拟输入电压被采样并且在 20/18 个时钟周期后，转换完成。12 位/10 位的数字转换结果被存入到 ADC_DR 中，并且 ADC_SR 中的 EOC 位被置 1。如果 EOC 标志已经是 1，那么 OVR 位会被置 1。
8. 然后 CPU 就可以读取 ADC_DR 中的数字值，并且自动清除 EOC。在连续转换模式中，如果 CPU 判断不需要更多的转换了，那么它可以写 STOP 位停止转换。这样 ADC 就会停止工作并且等待下一个开始转换的请求。注意在单次转换模式，ADC 不可以被停止，它会转换完所有的序列后自己停止。
9. 如果 NBRCH 不是 0，那么 ADC 会选择下一个需要转换的模拟输入通道，然后从上面第 6 步重新开始。
10. 如果 CONTCV 是 1，那么 ADC 会从第 5 步重新开始另一个转换序列。

8.2 寄存器说明

8.2.1 寄存器表

- Base Address: 0x4003_0000

Register	Offset	Description	Reset Value
–	0x0000 ~ 0x0050	Reserved	
ADC_ECR	0x0050	时钟使能寄存器	–
ADC_DCR	0x0054	时钟禁止寄存器	–
ADC_PMSR	0x0058	功耗管理状态寄存器	–
–	0x005C	Reserved	–
ADC_CR	0x0060	控制寄存器	0x00000040
ADC_MR	0x0064	模式寄存器	0x00000000
–	0x0068	Reserved	–
ADC_CSR	0x006C	状态清除寄存器	–
ADC_SR	0x0070	状态寄存器	0x00000000
ADC_IER	0x0074	中断使能寄存器	–
ADC_IDR	0x0078	中断禁止寄存器	–
ADC_IMR	0x007C	中断使能状态寄存器	0x00000000
ADC_CMR0	0x0080	转换模式寄存器0	0x00000000
ADC_CMR1	0x0084	转换模式寄存器1	0x00000000
ADC_CMR2	0x0088	转换模式寄存器2	0x00000000
–	0x008C ~ 0x00FC	Reserved	
ADC_DR	0x0100	转换结果寄存器	0x00000000
ADC_CMP0	0x0104	比较数据0寄存器	0x00000000
ADC_CMP1	0x0108	比较数据1寄存器	0x00000000

8.2.2 ADC_ECR (时钟使能寄存器)

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

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

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

8.2.3 ADC_DCR (时钟禁止寄存器)

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

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

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

8.2.4 ADC_PMSR (功耗管理状态寄存器)

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

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

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

8.2.5 ADC_CR (控制寄存器)

- Address = Base Address + 0x0060, Reset Value = 0x0000_0040

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

Name	B	Type	Description	Reset Value
SWRST	[0]	W	**SWRST : ADC 软件复位** 0: 无效 1: 复位 ADC 模块 当软件复位发生时，除了 ADC_PMSR 寄存器以外，其它所有寄存器都会恢复初始值。	-
ADCEN	[1]	W	**ADCEN : ADC 使能** 0: 无效 1: 使能 ADC 模块	-
ADCDIS	[2]	W	**ADCDIS : ADC 禁止** 0: 无效 1: 关闭 ADC 模块(待机模式) 如果 ADCEN 和 ADCDIS 都写 1，那么 ADC 会被禁用。	-
START	[3]	W	**START : 开始转换** 0: 无效 1: 开始模数转换，清除 EOC 标志位 **注意：**在开始转换前，用户必须保证 ADC 已经处于准备好转换的状态(ADC_SR 中的 READY 位必须为 1)	-
STOP	[4]	W	**STOP: 在连续转换模式下停止转换** 0: 无效 1: 停止连续转换	-
SAMPLE	[6:5]	R/W	**SAMPLE: ADC 采样周期 (setup 时间)** 00: 3 个周期 01: 4 个周期 10: 6 个周期 11: 8 个周期	10

			注意：采样时间不能小于(TBD) us	
ACCURACY	[31]	R/W	**ACCURACY: ADC 转换精度选择位** 0: 10 位 1: 12 位	1

8.2.6 ADC_MR (模式寄存器)

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

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

Name	B	Type	Description	Reset Value
PRLVAL	[4:0]	R/W	**PRLVAL[4:0]: 分频设置** 用来将 PCLK 分频，给 ADC 数字模块以及模拟模块的时钟使用。 如果 PRLVAL == 0，那么 FADC = PCLK 否则 FADC = PCLK / (2*PRLVAL)	00000
IES	[5]	R/W	**IES: 内部或者外部启动转换** 0：使用内部启动(软件触发) 1：使用外部启动(硬件触发)	0
ETRG	[9:8]	R/W	**ETRG[1:0]: 外部触发类型** 00：上升沿 01：下降沿 10：上升沿和下降沿 11：无效选择	00
NBRCH	[15:10]	R/W	**NBRCH[5:0]: 转换次数** **NBRCH[5:0]** / **转换次数** 000000b / 1 000001b / 2 … / … 001011b / 12 **注意：** 即使在单次转换模式，如果 NBRCH[5:0]的值大于 0，ADC 也会进行多次转换。	000000

NBRCMP0	[21:16]	R/W	**NBRCMP0[5:0]: 需要比较的转换序数** 当该次转换结果大于或者小于 ADC_CMP0 寄存器时，将产生一个 CMPxH/CMPxL 中断	000000
NBRCMP1	[27:22]	R/W	**NBRCMP1[5:0]: 需要比较的转换序数** 当该次转换结果大于或者小于 ADC_CMP1 寄存器时，将产生一个 CMPxH/CMPxL 中断	000000
CONTCV	[31]	R/W	**CONTCV: 连续转换** 0: 单次转换模式。ADC 根据 NBRCH[5:0]中设置的值转换输入的信号并且停止 1: 连续转换模式。ADC 根据 NBRCH[5:0]中设置的值转换输入的信号并且重复不停的循环转换。 该位初始值为 0. **注意：**在连续转换模式下，ADC 收到停止指令后，仍然会完成当前正在进行的转换，看起来像是多转换了一次。	0

8.2.7 ADC_CSR (状态清除寄存器)

- Address = Base Address + 0x006C, Reset Value = 0x0000_0000

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

Name	B	Type	Description	Reset Value
OVR	[2]	W	**OVR：转换溢出中断** 0: 无效 1: 清除该中断	-
CMP0H	[4]	W	**CMP0H：转换结果大于 ADC_CMP0 中断** 0: 无效 1: 清除该中断	-
CMP0L	[5]	W	**CMP0L：转换结果小于 ADC_CMP0 中断** 0: 无效 1: 清除该中断	-
CMP1H	[6]	W	**CMP1H：转换结果大于 ADC_CMP1 中断** 0: 无效 1: 清除该中断	-
CMP1L	[7]	W	**CMP1L：转换结果小于 ADC_CMP1 中断** 0: 无效 1: 清除该中断	-

8.2.8 ADC_SR (状态寄存器)

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

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

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

ADCENS	[8]	R	**ADCENS：ADC 使能状态** 0: ADC 被禁止 1: ADC 被使能	0
CTCVS	[9]	R	**CTCVS： 连续转换模式状态** 0: 单次模式 1: 连续模式	0

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

Table 8-8 是否准备好进行转换

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

8.2.9 ADC_IER (中断使能寄存器)

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

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

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

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

8.2.10 ADC_IDR (中断禁止寄存器)

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

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

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

8.2.11 ADC_IMR (中断使能状态寄存器)

- Address = Base Address + 0x007C, Reset Value = 0x0000_0000

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

Name	B	Type	Description	Reset Value
EOC	[0]	W	**EOC：转换结束中断** 0: 该中断没有使能 1: 该中断使能	-
READY	[1]	W	**READY：ADC READY 中断** 0: 该中断没有使能 1: 该中断使能	-
OVR	[2]	W	**OVR：转换溢出中断** 0: 该中断没有使能 1: 该中断使能	-
CMP0H	[4]	W	**CMP0H：转换结果高于 ADC_CMP0 中断** 0: 该中断没有使能 1: 该中断使能	-
CMP0L	[5]	W	**CMP0L：转换结果低于 ADC_CMP0 中断** 0: 该中断没有使能 1: 该中断使能	-
CMP1H	[6]	W	**CMP1H：转换结果高于 ADC_CMP1 中断** 0: 该中断没有使能 1: 该中断使能	-
CMP1L	[7]	W	**CMP1L：转换结果低于 ADC_CMP1 中断** 0: 该中断没有使能 1: 该中断使能	-

8.2.12 ADC_CMRx (转换模式寄存器)

- ADC_CMR0 Address = Base Address + 0x0080, Reset Value = 0x0000_0000
- ADC_CMR1 Address = Base Address + 0x0084, Reset Value = 0x0000_0000
- ADC_CMR2 Address = Base Address + 0x0088, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		CVx						RSVD		CVx						RSVD		CVx						RSVD		CVx					
0	0	0	0	0	0	0	0	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	B	Type	Description	Reset Value
CVx	[5:0] [13:8] [21:16] [29:24]	RW	模拟输入通道选择 ADC_CMR0 x = {1, 2, 3, 4} ADC_CMR1 x = {5, 6, 7, 8} ADC_CMR2 x = {9, 10, 11, 12}	-

CVx 值	选择的通道
000000	AIN0
000001	AIN1
000010	AIN2
……	……
001011	AIN11
001100	AIN12
001101	AIN13
001110	AIN14

注意：目前 APT32F003 系列最多只支持前 12 次转换的通道设置，如果 NBRCH 的值大于 12 (0xC)，那么比第 12 以后的转换将会锁定在 CV1 设定的通道上。

8.2.13 ADC_DR (转换结果寄存器)

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

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

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

8.2.14 ADC_CMP0 (比较数据0寄存器)

- Address = Base Address + 0x0104, Reset Value = 0x0000_0000

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

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

8.2.15 ADC_CMP1 (比较数据1寄存器)

- Address = Base Address + 0x0108, Reset Value = 0x0000_0000

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

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

9 I/O 端口

9.1 概述

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

- Port A0: 14 位输入/输出端口, PA0.0 ~ PA0.13 (16/20脚)
- Port B0: 4 位输入/输出端口, PB0.0 ~ PB0.3 (20脚)
- Port C0: 4 位输入/输出端口, PC0.0 ~ PC0.3 (24脚)

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。

9.1.1 主要特性

- 5种 I/O 模式:
    - 禁止输入 & 输出模式（高阻）
    - 输入模式.
    - 输出模式(禁止输入)
    - 带输入监测的输出模式（输出的同时输入路径也使能）
    - 多功能复用模式
- 在各个模式下，都可对上拉/下拉进行使能/禁用
- 输出模式下，推挽式输出和开漏式输出可选（输出模式和复用功能无关，可单独配置）
- 每一个 I/O 口都可被配置成外部中断源
- 其中 4 个管脚（PA0.8~PA0.11）支持灌入大电流

9.1.2 管脚描述

Table 9-1 管脚描述

Pin Name	Function	I/O Type	Active Level	Comments
PA0[13:0]	通用 I/O 口 A0	I/O	-	-
PB0[3:0]	通用 I/O 口 B0	I/O	-	-
PC0[3:0]	通用 I/O 口 C0	I/O	-	-

注意：

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

9.2 功能描述

9.2.1 电路图

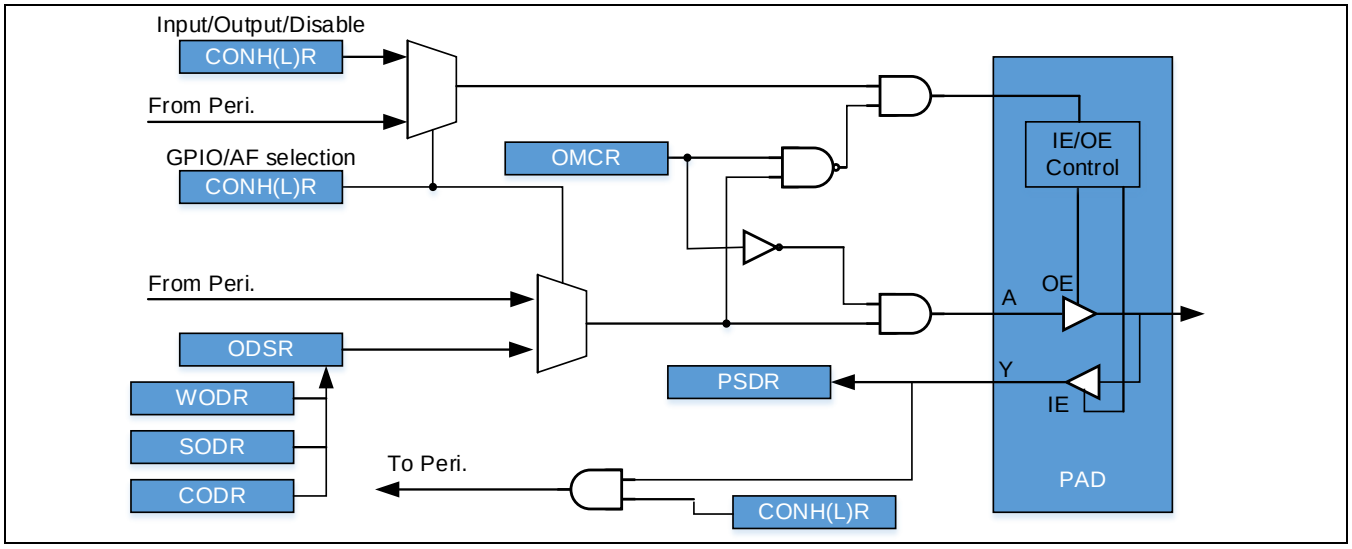


Figure 9-1 GPIO原理图

9.2.2 工作原理

9.2.2.1 功能性描述

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

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

9.2.2.2 输入与输出的配置

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

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

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

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

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

9.2.3 工作模式

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

9.2.4 外部中断与唤醒功能

通过设置寄存器 GPIO_IEN 位和 GPIO_IGRP，任何一个 GPIO 管脚都可以设置成外部中断源。中断触发方式可由与 SYSCON EXI 相关的控制寄存器来进行设置。中断路径中的自适应噪声滤波器能对输入信号进行去抖处理。去抖处理不依赖于系统时钟，当系统处于 DEEP-SLEEP 模式时，仍旧有效。所有的 GPIO 按后缀进行分组。每组中 4 个管脚中的其中一个都可以通过配置寄存器 GPIO_IGRP 来被设置成 EXI。

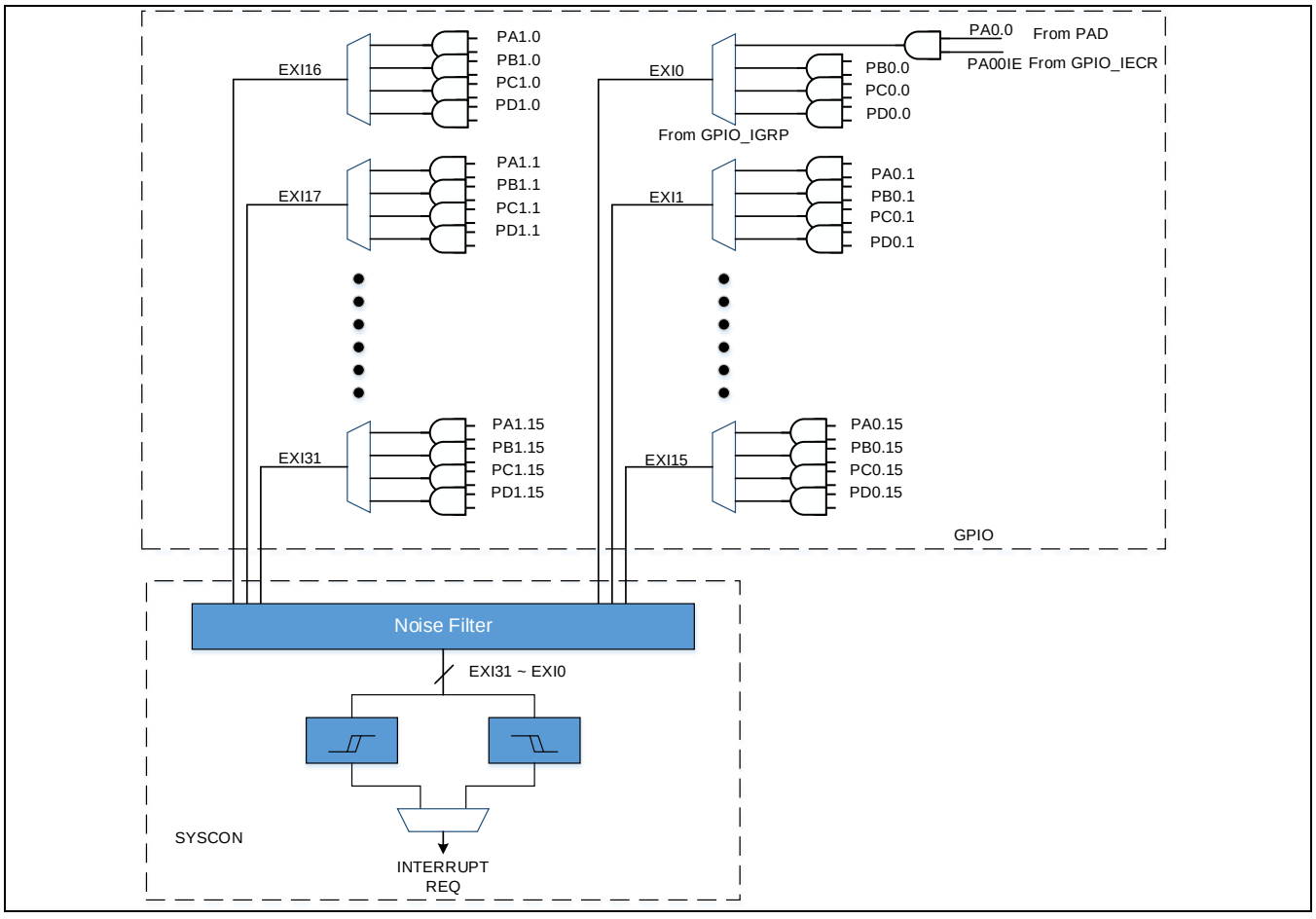


Figure 9-2 GPIO外部中断原理图

当芯片处于 SLEEP 模式或 DEEP-SLEEP 模式下，GPIO 可以被作为唤醒源使用。需要使能 GPIO 外部中断功能， 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。

9.3 寄存器说明

9.3.1 寄存器表

- Base Address of A0: 0x4004_0000
- Base Address of B0: 0x4004_1000
- Base Address of C0: 0x4004_2000

Register	Offset	Description	Reset Value
GPIO_CONLR	0x00	低位控制寄存器	A0: B0:
GPIO_CONHR	0x04	高位控制寄存器	A0: B0:
GPIO_WODR	0x08	输出数据寄存器	0x0000_0000
GPIO_SODR	0x0C	输出置位寄存器	0x0000_0000
GPIO_CODR	0x10	输出清除寄存器	0x0000_0000
GPIO_ODSR	0x14	输出状态寄存器	0x0000_0000
GPIO_PSDR	0x18	管脚状态寄存器	0x0000_0000
RSVD	0x1C	Reserved	-
GPIO_PUDR	0x20	上拉/下拉配置寄存器	A0: B0:
GPIO_DSCR	0x24	驱动强度配置寄存器	0x0000_0000
GPIO_OMCR	0x28	输出模式配置寄存器	0x0000_0000
GPIO_IECR	0x2C	外部中断使能寄存器	0x0000_0000

注意：

- GPIO_IGRP 跟 GPIOA0 使用同一个 PCLK

- Base Address of GPIO_IGRP: 0x4004_4000

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

9.3.2 GPIO_CONLR (低位控制寄存器)

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

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

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

9.3.3 GPIO_CONHR (高位控制寄存器)

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

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

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

9.3.4 GPIO_WODR (输出数据寄存器)

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

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

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

9.3.5 GPIO_SODR (输出置位寄存器)

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

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

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

9.3.6 GPIO_CODR (输出清除寄存器)

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

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

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

9.3.7 GPIO_ODSR (输出状态寄存器)

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

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

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

9.3.8 GPIO_PSDR (管脚状态寄存器)

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

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

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

9.3.9 GPIO_PUDR (上拉/下拉配置寄存器)

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

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

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

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

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

9.3.10 GPIO_DSCR (驱动强度配置寄存器)

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

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

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

'b0x: 驱动强度 15mA
'b10: 驱动强度 15mA
'b11: 驱动强度 120mA

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

驱动强度调整只适用于大电流工作模式。

9.3.11 GPIO_OMCR (输出模式配置寄存器)

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

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

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

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

9.3.12 GPIO_IECR (外部中断使能寄存器)

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

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

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

NOTE:
该寄存器只支持写入功能。

9.3.13 GPIO_IGRP (外部中断组配置寄存器)

- Address = Base Address + 0x4000, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				GRP13		GRP12		GRP11		GRP10		GRP9		GRP8		GRP7		GRP6		GRP5		GRP4		GRP3		GRP2		GRP1		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	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
GRP13	[27:26]	RW	选择外部中断组 13	'b00
GRP12	[25:24]	RW	选择外部中断组 12	'b00
GRP11	[23:22]	RW	选择外部中断组 11	'b00
GRP10	[21:20]	RW	选择外部中断组 10	'b00
GRP9	[19:18]	RW	选择外部中断组 9	'b00
GRP8	[17:16]	RW	选择外部中断组 8	'b00
GRP7	[15:14]	RW	选择外部中断组 7	'b00
GRP6	[13:12]	RW	选择外部中断组 6	'b00
GRP5	[11:10]	RW	选择外部中断组 5	'b00
GRP4	[9:8]	RW	选择外部中断组 4	'b00
GRP3	[7:6]	RW	选择外部中断组 3	'b00
GRP2	[5:4]	RW	选择外部中断组 2	'b00
GRP1	[3:2]	RW	选择外部中断组 1	'b00
GRP0	[1:0]	RW	选择外部中断组 0	'b00
'b00: GPIOA0.x 被选中 'b01: GPIOB0.x 被选中 'b10: GPIOC0.x 被选中 'b11: GPIOD0.x 被选中 'x' 表示组数				

NOTE:

在本产品中，只有 GPIOA0 和 GPIOB0 是可用的，所以当外部中断源不存在时，GPIOA0 会被认作默认外部中断源。

10 16/32位 定时器/计数器

10.1 概述

本章节介绍16/32位定时器/计数器的使用。定时器/计数器(简称TC)有三种工作模式：比较匹配模式，输入捕捉模式，输出翻转或者称为PWM模式。TC可以通过特定的管脚输出PWM信号，并且TC的时钟源支持从外部引入。

10.1.1 主要特性

- 可编程的时钟源，支持从外部管脚输入
- 位数可编程的计数器，支持16位或者32位
  - TC0, TC1 & TC2 支持16位
  - TC3 支持16位或者32位
- 单次计数或者重复循环计数
- 计数值匹配和溢出
- 捕捉
- 支持上升沿，下降沿和上升下降同时捕捉
- 两个捕捉寄存器，可捕捉两个沿
- 输出翻转功能
- PWM输出
- 周期和频率可编程
- 有效电平和空闲电平可编程
- 32位计数器可支持最高38位分辨率，16位计数器可支持最高22位分辨率，支持扩展功能
- 带Debug选项
- 支持作为计数器A的触发源

10.1.2 管脚描述

Table 10-1 管脚描述

Pin Name	Function	I/O Type	Comments
TCLK[2], TCLK[0]	外部时钟源	I	
TCAP[3:0]	捕捉输入	I	
TPWM[3:0]	PWM / 计数器输出管脚	O	

10.2 功能描述

10.2.1 模块框图

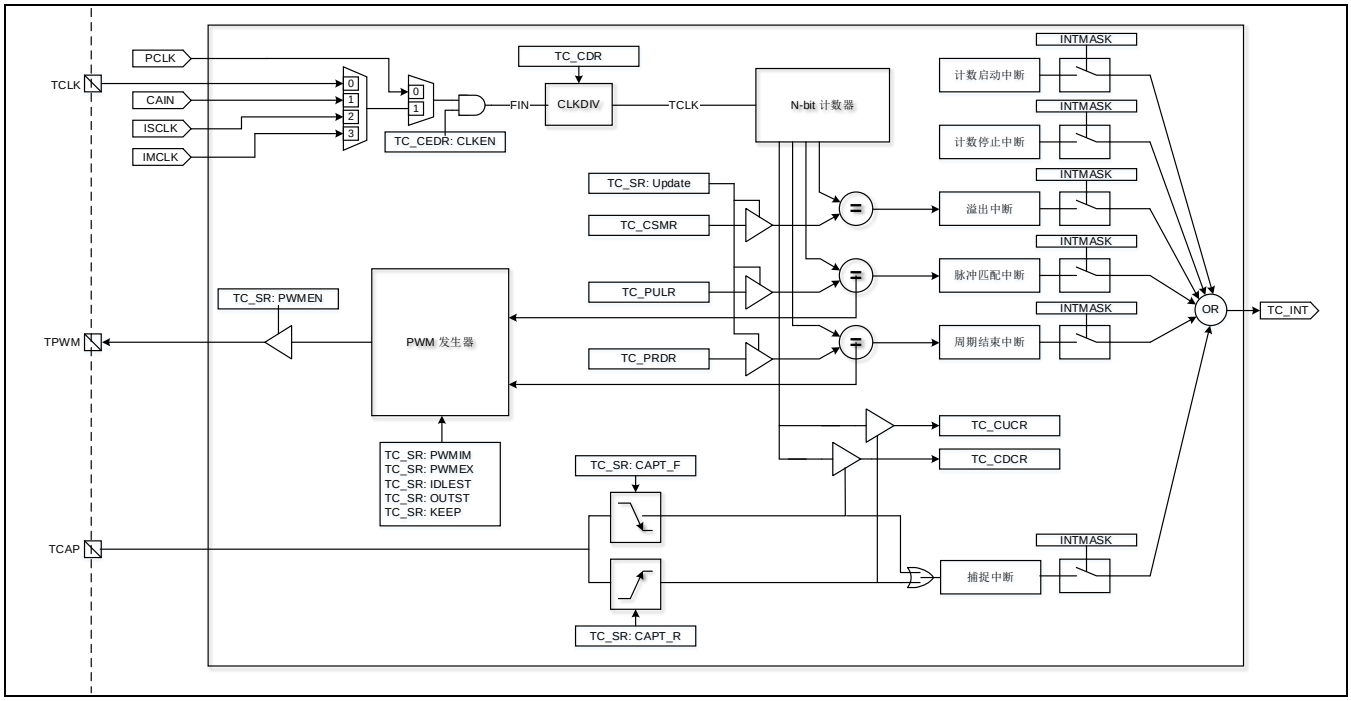


Figure 10-1 TC模块框图

10.2.2 计数器位数

计数器的位数可以由一个寄存器控制，TC_CSMR (Counter Size Mask Register). 如果SIZE[4:0]的值是n，那么TC将变成一个(n+1)位数的定时器，也就是说，定时器将从1计数到2^(n+1)-1为止。

10.2.3 计数器时钟

10.2.3.1 时钟源

TC可以使用两种不同的时钟(同步或者异步时钟)。同步时钟为PCLK，而异步时钟则是从下列源中任意选择的一个：晶振，TCLK管脚，计数器A的CAOUT输出。如果要使用外部时钟源TCLK，那么在开始定时器之前，需要先将IO管脚配置好(配置成TCLK)。

10.2.3.2 计数器时钟

基于FIN的计数器时钟由TC_CDR (Clock Divider Register)中的DIVM[7:0]和DIVN[3:0]决定。计数器时钟的频率TCCLK由FIN和内部时钟分频器(DIVM和DIVN)共同确定：

TCCLK = FIN / 2^DIVN / (DIVM +1) (例如，DIVN=3/DIVM=3，FIN=16MHz，那么TCCLK=500KHz)

计数器分辨率 = 1 / TCCLK

注意 ：在DIVN不等于0的时候，禁止将DIVM设为0

10.2.3.3 Debug选项

TC debug选项用来选择当CPU被调试器暂停的时候，计数器是否也同时停止计数。

10.2.4 芯片内部的触发源

当TC_SR寄存器中的HWTRIG位被置位后，TC0/1可以作为计数器A或者ADC的触发源使用。TC0的脉冲匹配和周期结束中断可以作为计数器A的载波打开和关闭触发源。TC1的脉冲比较中断可以作为ADC的转换开始触发源。详细内容请参考计数器A和ADC章节。

10.2.5 更新控制寄存器

所有跟计数器工作有关的寄存器值都会在定时器的开始位被置位的时候载入到计数的逻辑电路中，而当计数器在正常工作的时候，所有寄存器都会被冻结住无法更改，除非发生了软件复位或者操作了时钟使能控制位。如果需要在计数器工作的时候(on-the-fly)更改配置，那么请使用"UPDATE"位的更新机制。当TC_SR中的UPDATE位被置位的时候才可以修改寄存器，当寄存器修改完成后，需要清除TC_SR中的UPDATE位，告诉计数逻辑电路去载入寄存器的值。在内部逻辑中，所有的改动只有在循环模式中的计数开始或者周期结束事件发生后，才会生效。

注意时钟源选择寄存器是个例外，这个寄存器只有当计数器没有开始，并且时钟被禁止的情况下，才可以进行修改，否则修改无效。

10.2.6 连续计数模式

当TC_SR中CNTM位被置位时，定时器工作在连续计数模式。在这个模式下，计数值逐次增加，直到2^(SIZE+1)-1为止，其中SIZE在TC_CSMR (Counter Size Mask Register)寄存器中设置。当TC_SR寄存器中的REPEAT位是0的时候，计数值在计数到2^(SIZE+1)-1后会被清零，这时定时器会产生停止中断和溢出中断。

10.2.6.1 匹配和溢出

定时器有两个比较值可以分别用来产生两种匹配中断，一个叫脉冲匹配中断，另一个叫周期结束中断。

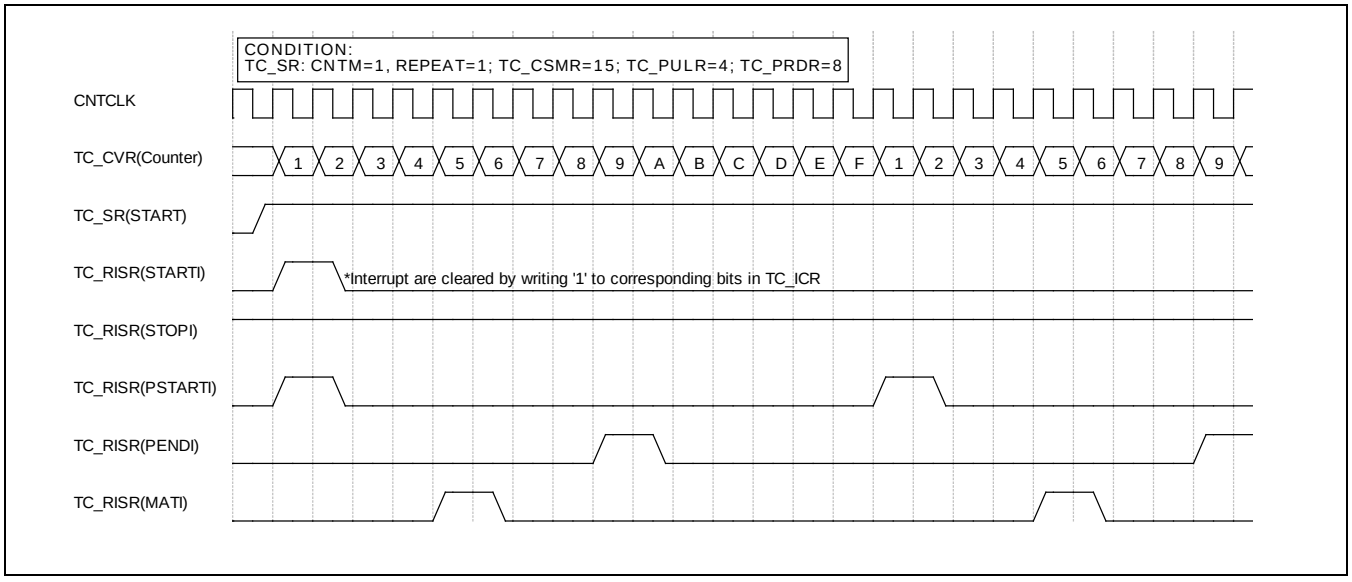


Figure 10-2 匹配和溢出时序

当定时器开始计数，立即产生计数启动中断和周期开始中断。当计数值跟TC_PULR中PULSE的值相等的时候，定时器产生脉冲匹配中断，而当计数值跟TC_PRDR中PERIOD值相等的时候，产生周期结束中断。当计数器溢出的

时候，则产生溢出中断。TC_SR中REPEAT位用来配置定时器的工作模式是单次(one-short)还是循环重复。如果REPEAT位被置位，那么计数溢出后计数器的值会被重置到1，然后自动重新开始计数。

将TC_CCR寄存器的START位写1后，可以清除TC_SR中的START启动状态，并且同时停止定时器。定时器停止的方式由TC_SR寄存器中STOPHOLD和STOPCLEAR位决定。

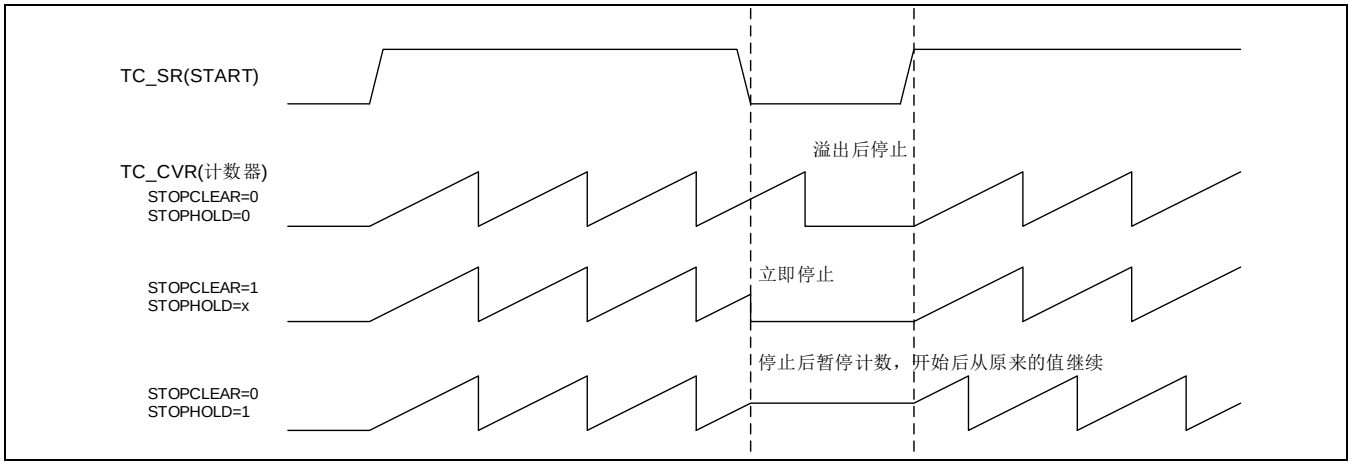


Figure 10-3 基于STOP, STOPCLEAR, STOPHOLD的计数控制

如果STOPHOLD和STOPCLEAR都是0，清除TC_SR的START位后，计数器会计数到溢出后再停止，并且计数值自动清零。如果STOPCLEAR被置位，清除TC_SR的START位后，计数器会立即停止。如果需要暂停计数，需要将STOPHOLD置1并且保持STOPCLEAR为0 (STOPCLEAR位的优先权高于STOPHOLD)，这种情况下停止计数后，计数器将保留计数值，再重新开始时会从保留的值开始继续计数。

10.2.6.2 输入捕捉

TC可以捕捉外部的输入信号，将计数值存入捕捉寄存器TC_CUCR (Capture Up Counter Register)和TC_CDCR (Capture Down Counter Register)中。这个工作模式用来计算外部信号的时间差异。外部输入触发信号的上升或者下降沿可以由寄存器预先定义好的值进行选择，当检测到预设好的上升或下降沿的时候，相应的计数器计数值会被复制到寄存器TC_CUCR或TC_CDCR中。

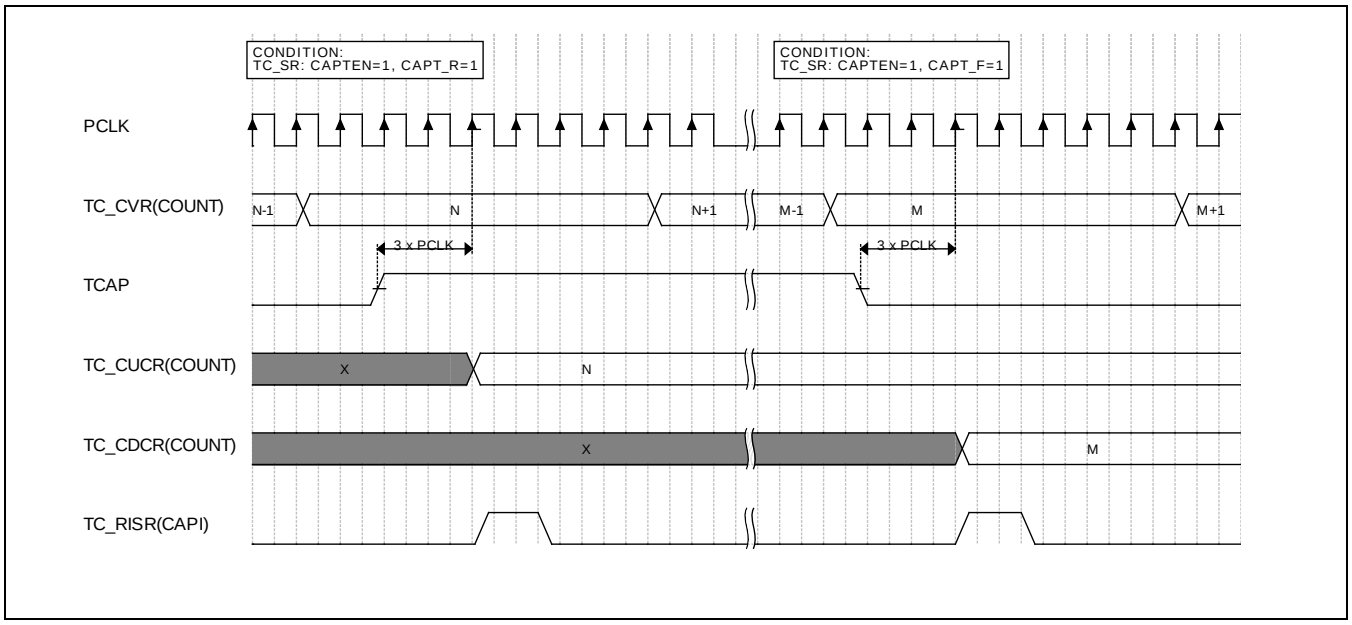


Figure 10-4 输入信号捕捉时序

对于输入捕捉模式，定时器的时钟必须选择PCLK，并且TCAP管脚输入的外部信号必须至少保持3个PCLK的时钟周期长度，以免被当作毛刺信号过滤掉。

10.2.7 周期匹配模式

当TC_SR寄存器中的CNTM位为0，TC工作在周期匹配模式。这个模式下，计数器从1一直计数到TC_PRDR中PERIOD设置的值。当计数到PERIOD值时，定时器产生周期结束中断。如果TC_SR寄存器中REPEAT位为1，计数器在周期结束后会自动从1开始继续循环计数，如果REPEAT为0，计数器则被清零并且停止计数。

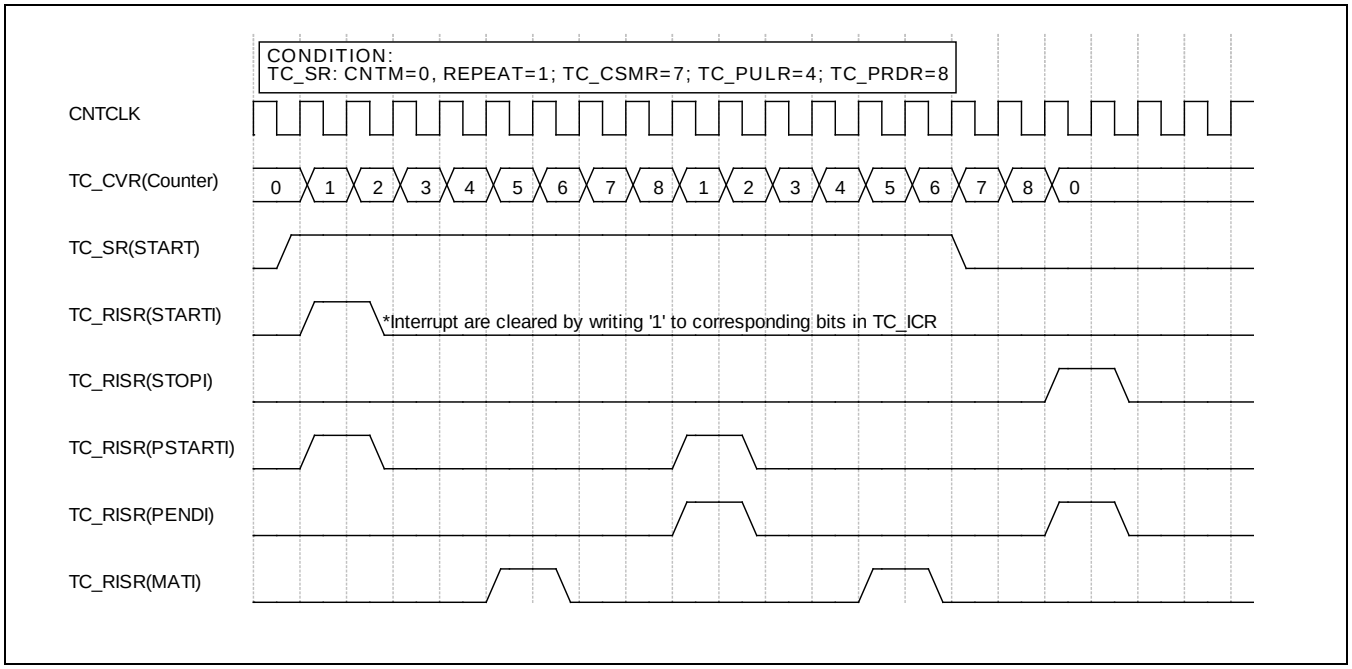


Figure 10-5 周期匹配模式时序

该模式下，当计数器的值跟TC_PULR和TC_PRDR相等时，定时器会分别产生脉冲匹配中断和周期结束中断。

基于TC_SR的PWMIM位，TC可以产生两种类型的输出信号，输出翻转信号和PWM信号。注意为了将信号输出到芯片的管脚上，TC_SR中的PWMEN位必须为1，并且管脚的复用功能设置也必须正确。

10.2.7.1 输出翻转功能

使能输出翻转功能后，定时器的输出会在检测到周期结束时翻转。比如，写0x08到TC_PRDR，计数值会一直增加直到0x08，并且在此时翻转，所以输出的波形周期为：

2 x TCCLK x PERIOD

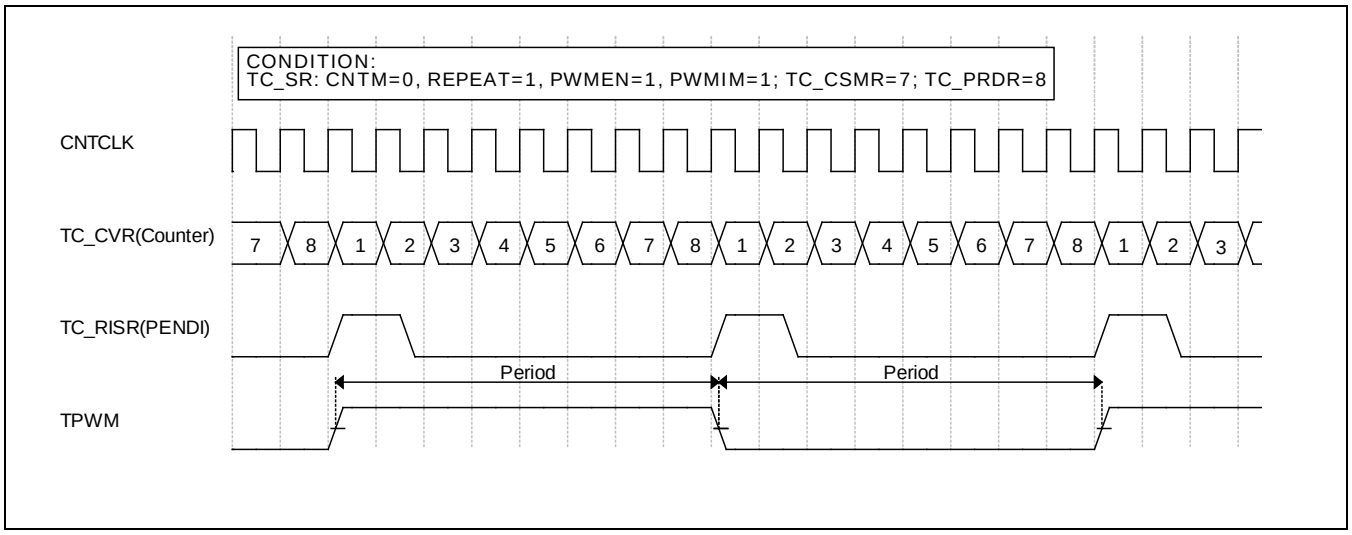


Figure 10-6 输出翻转功能的时序

10.2.7.2 PWM功能

TC可以用来产生PWM (Pulse Width Modulation)输出。在这模式下，当定时器启动时，TC_PULR寄存器的PULSE会被载入到PWM逻辑电路中。 TC_PULR的值必须小于或等于TC_PRDR的值。

PWM输出信号的初始值由TC_SR寄存器中的OUTST位决定。当OUTST为1的时候，输出信号为高；当OUTST为0的时候，输出信号为低。当计数器的值等于TC_PULR的时候，定时器产生脉冲匹配事件，并且此时PWM输出翻转成与OUTST位相反的状态。之后计数器继续工作，直到等于TC_PRDR中的PERIOD值，此时定时器产生周期结束事件，并且PWM输出再次翻转。这时如果REPEAT位为1，计数器则从1开始重新计数，如果REPEAT为0，那么定时器停止并且计数器清零。

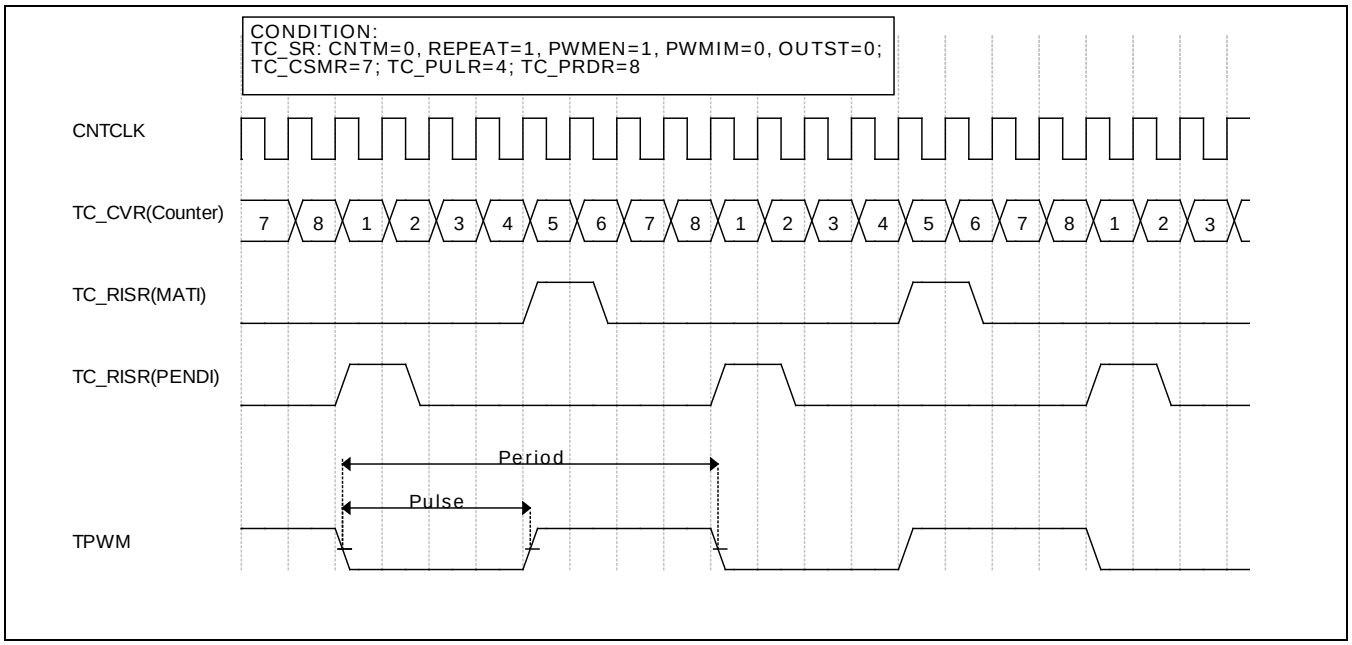


Figure 10-7 PWM时序

10.2.7.2.1 PWM扩展位

扩展计数器的值与扩展逻辑，PWMEX寄存器的值一起，用来“拉伸”PWM输出的周期。“拉伸”的值为1个时钟周期。

由于PWMEX是一个6位的寄存器，所以每64个PWM周期为一个扩展周期。扩展周期中，有一个或多个PWM输出会被“拉伸”一个时钟周期的长度，也就是比普通周期多一个时钟周期。扩展的周期位置由TC_SR中的PWMEX位决定。由于PWMEX有6位，所以PWM输出能够扩展到约38位的分辨率(32位计数器+6位扩展)。

Table 10-2 PWM扩展位

PWMEX Bit	扩展周期(第x个周期拉伸1个周期)
0	32
1	16, 48
2	8, 24, 40, 56
3	4, 12, 20, 28, 36, 44, 52, 60
4	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
5	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63

下图演示了基于PWMEX的PWM输出信号。

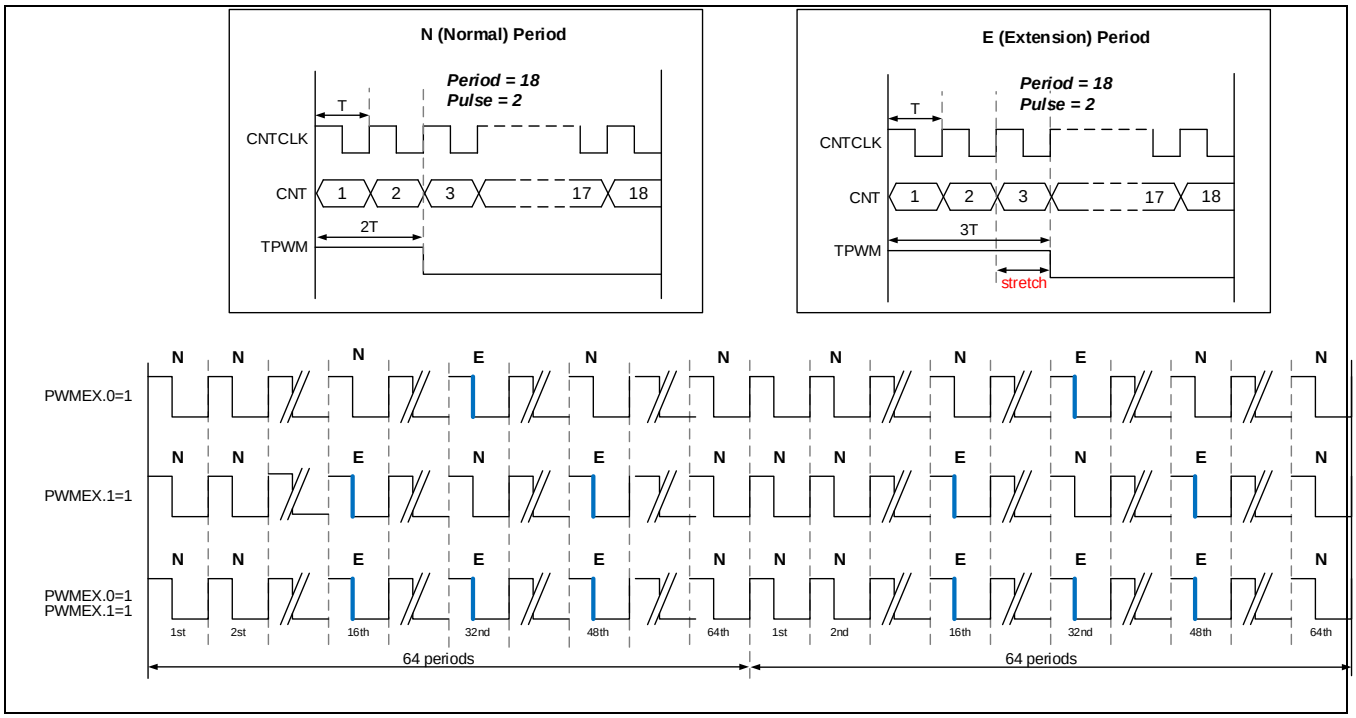


Figure 10-8 扩展PWM的波形

PWM 信号周期可以由下面的公式计算得出。

$$\text{Duty (\%)} = \left(\frac{(\text{PULSE} \times (64 - \text{E})) + (\text{PULSE} + 1) \times \text{E}}{\text{PERIOD} \times 64} \right) \times 100 = \left(\frac{\text{PULSE}}{\text{PERIOD}} + \frac{\text{E}}{\text{PERIOD} \times 64} \right) \times 100$$

，‘E’是64个周期中设置了扩展的周期数

例如，正常情况下，当PERIOD为100，PULSE为50的时候，PWM输出占空比位50%。如果PWMEX只有bit0为1，那么第32个和第64个脉冲周期的周期宽度为51个计数时钟周期。这个情况下，PWM输出的占空比为50.015625%，因为1/64是0.015625。如果只有PWMEX的bit5为1，那么每64个周期中的32个周期为扩展周期，这样占空比则为50.5%。

10.2.7.2.2 PWM波形

下图展示了基于PERIOD和PULSE的PWM波形。正常周期中，当PULSE为0的时候占空比为0%，当PULSE等于PERIOD的时候占空比100%。

PWM的输出电平由OUTST位决定。当OUTST为0的时候，PWM输出电平从低开始。可以看出，PWM扩展的电平为高还是低可以用OUTST位来控制。

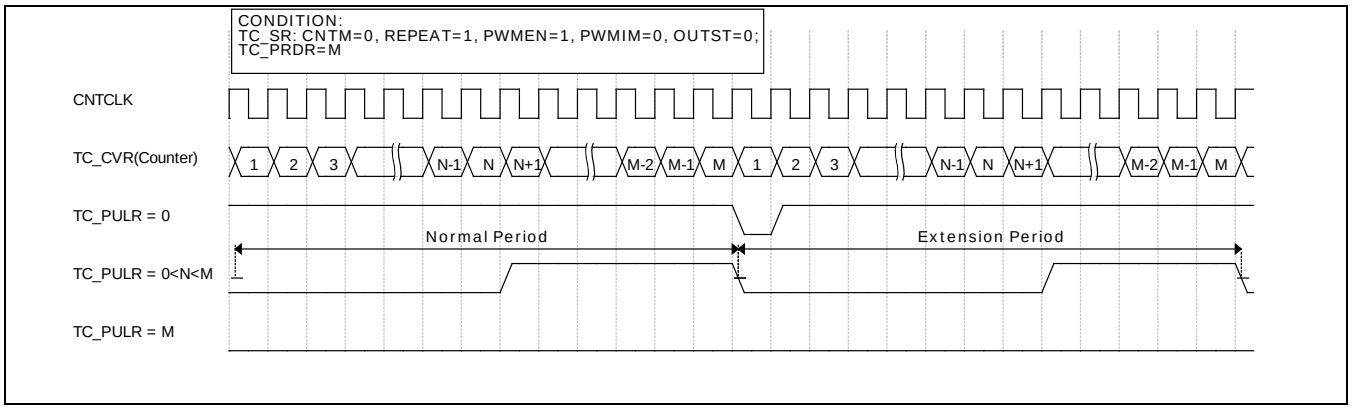


Figure 10-9 OUTST=0时PWM波形

如果OUTST为1，PWM输出信号从高电平开始。

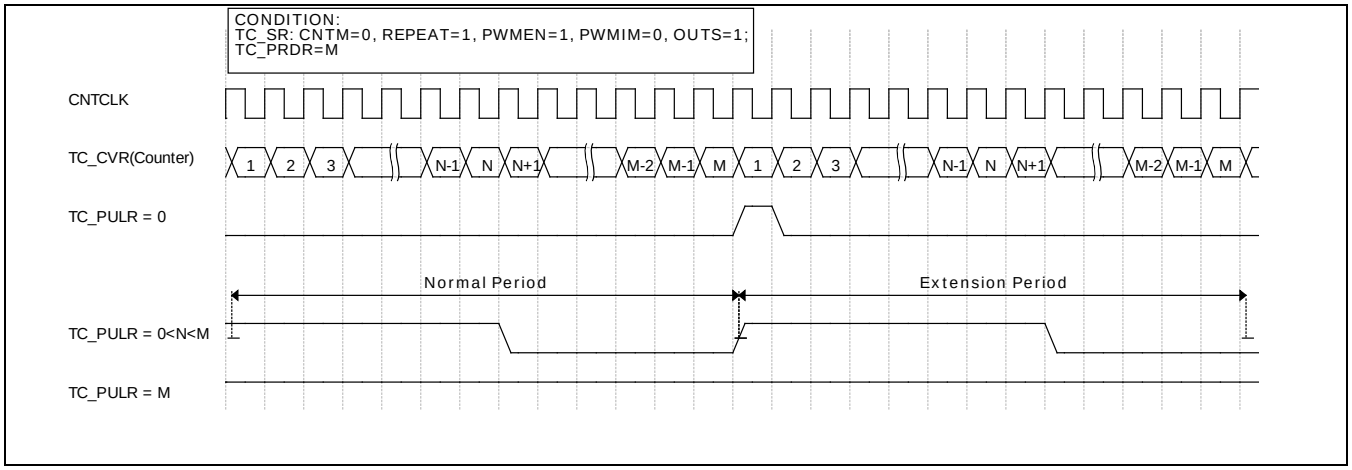


Figure 10-10 OUTST=1时PWM波形

10.2.7.2.3 PWM输出极性

当定时器停止时，PWM输出的状态由TC_SR寄存器中的STOPCLEAR, STOPHOLD, KEEP, IDLEST和OUTST各个控制位决定。下表列出了各个控制位下输出极性的不同。

Table 10-3 PWM输出极性

STOP CLEAR	STOP HOLD	KEEP	IDLEST	OUTST	TPWM	说明
0	0	0	0	X	L	定时器在周期结束时停止，输出状态由IDLEST决定
0	0	0	1	X	H	
0	0	1	X	0	H	定时器在周期结束时停止，输出状态为OUTST的反
0	0	1	X	1	L	
0	1	X	X	X	L/H	定时器立即暂停计数，PWM输出保持为停止前的状态

STOP CLEAR	STOP HOLD	KEEP	IDLEST	OUTST	TPWM	说明
1	X	X	0	X	L	定时器立即停止计数，PWM输出由IDLEST决定
1	X	X	1	X	H	

注意：

控制位的优先权为：STOPCLEAR > STOPHOLD > KEEP > IDLESL.

如果在STOPHOLD和STOPCLEAR为0的情况下清除START位，那么定时器会在当前PERIOD周期结束后才停止。在这个状态下，如果TC_SR中的KEEP位为1，那么TC保持PWM输出状态，也就是与OUTST相反的状态，如果KEEP位为0，那么PWM的输出状态则由IDLEST来决定。

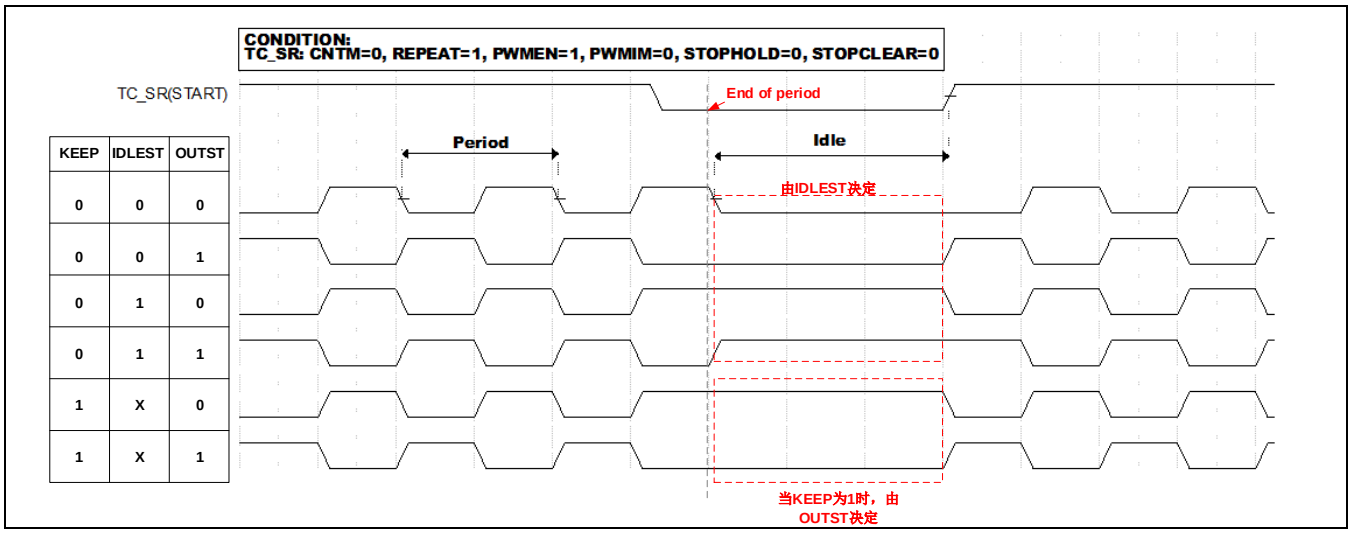


Figure 10-11 Idle状态下的PWM波形

如果STOPHOLD为1，但STOPCLEAR为0，当清除START的时候，定时器会停止并且计数器会保留停止前的值，PWM输出也保持停止前的状态。当再次给START置1的时候，定时器从之前保留的值重新开始计数。

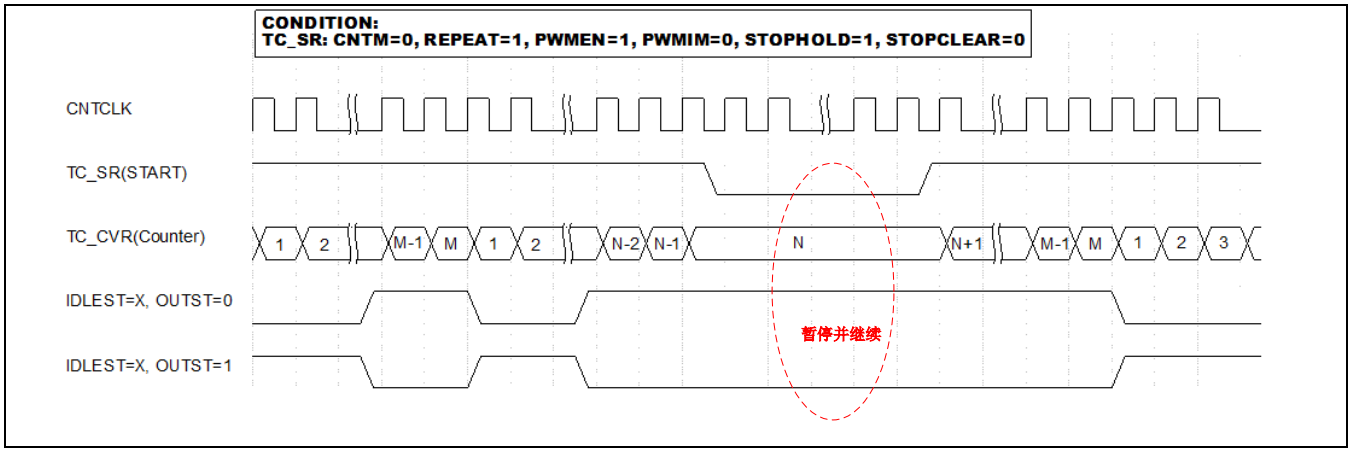


Figure 10-12 STOPHOLD=1, STOPCLEAR=0时PWM波形

如果STOPCLEAR为1，当清除START位时，定时器立即停止，并且输出电平由IDLEST决定。

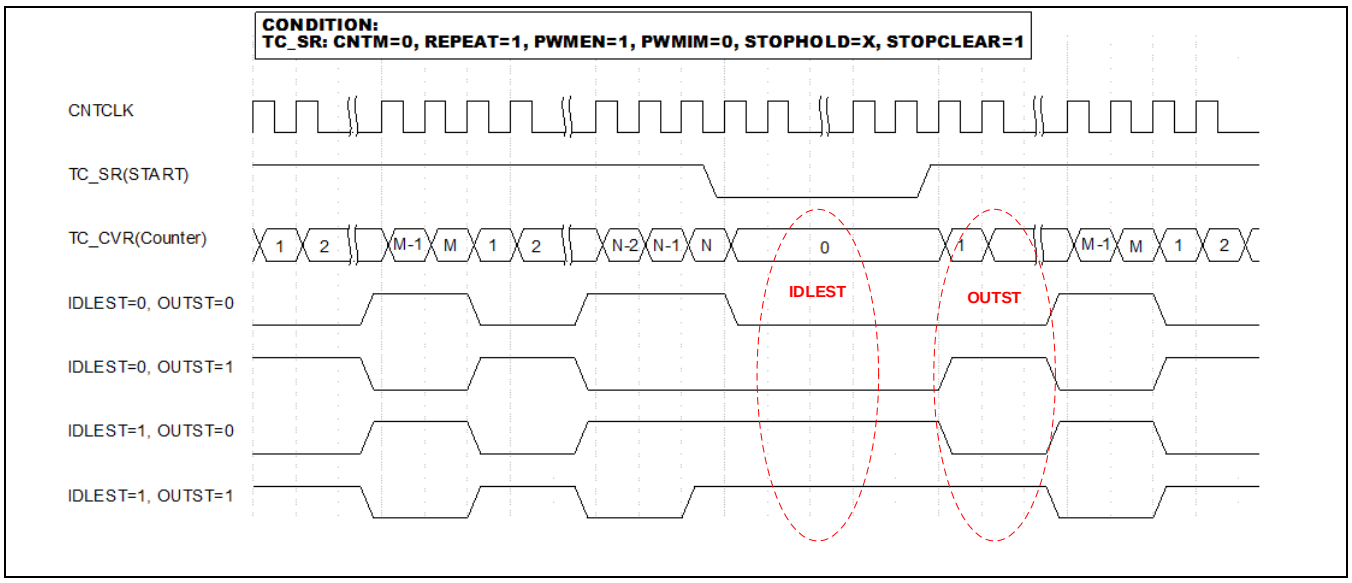


Figure 10-13 STOPCLEAR=1时PWM波形

定时器复位后，输出信号的初始值为低电平。当STOPCLEAR为1并且定时器不在工作的时候，通过改变IDLEST的值可以马上改变管脚的输出电平。

10.2.8 中断

定时器/计数器可以产生7种中断：STARTI, STOPI, PSTARTI, PENDI, MATI, OVFI 和 CAPTI。

10.2.8.1 计数启动中断

当定时器开始计数时，产生启动中断。

10.2.8.2 计数停止中断

当定时器停止计数时，产生停止中断。

10.2.8.3 周期开始中断

当计数周期开始时，产生周期开始中断。

10.2.8.4 周期结束中断

当计数周期结束时，产生周期结束中断。

10.2.8.5 脉冲匹配中断

当计数器的值等于脉冲寄存器TC_PULSE设定的值时，产生脉冲匹配中断。

10.2.8.6 溢出中断

当定时器计数溢出时，产生溢出中断。

10.2.8.7 捕捉中断

捕捉模式下，当外部捕捉信号上升或下降沿触发时，产生捕捉中断。

10.2.8.8 中断处理

- 进入中断处理程序(ISR)并且调用C函数
- 读寄存器TC_IMSR，确定中断源
- 写TC_ICR清除相应中断的状态位
- 中断处理
- 退出中断服务程序

10.3 寄存器说明

10.3.1 寄存器表

BASE Address: TC0: 0x4005_0000; TC1: 0x4005_1000; TC2: 0x4005_2000; TC3: 0x4005_3000

Table 10-4 寄存器表

Register	Offset	Description	Reset Value
TC_IDR	0x000	ID寄存器	0x0011_000A
TC_CSSR	0x004	时钟源选择寄存器	0x0000_0000
TC_CEDR	0x008	时钟使能/禁止寄存器	0x0000_0000
TC_SRR	0x00C	软件复位寄存器	0x0000_0000
TC_CSR	0x010	控制使能寄存器	0x0000_0000
TC_CCR	0x014	控制清除寄存器	0x0000_0000
TC_SR	0x018	控制状态寄存器	0x0000_0002
TC_IMSCR	0x01C	中断使能/禁止寄存器	0x0000_0000
TC_RISR	0x020	原始中断状态寄存器	0x0000_0000
TC_MISR	0x024	中断状态寄存器	0x0000_0000
TC_ICR	0x028	中断状态清除寄存器	0x0000_0000
TC_CDR	0x02C	时钟分频寄存器	0x0000_0000
TC_CSMR	0x030	计数器位数控制寄存器	0x0000_001F
TC_PRDR	0x034	周期值寄存器	0x0000_0000
TC_PULR	0x038	脉冲比较值寄存器	0x0000_0000
TC_CUCR	0x04C	捕捉上升沿计数寄存器	0x0000_0000
TC_CDCR	0x050	捕捉下降沿计数寄存器	0x0000_0000
TC_CVR	0x054	当前计数值寄存器	0x0000_0000

10.3.1.1 TC_IDR (ID寄存器)

- Address = Base Address + 0x0000, Reset Value = 0x0011_000A

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

Name	Bit	Type	Description
IDCODE	[25:0]	R	ID Code寄存器 相应IP的ID code，无法更改

10.3.1.2 TC_CSSR (时钟源选择寄存器)

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

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

Name	Bit	Type	Description
CLKSRC[2:0]	[2:0]	RW	时钟源选择寄存器 000: 计数器时钟源为PCLK 001: 计数器时钟源为外部TCLK0管脚 010: 无效选择 011: 计数器时钟源为外部TCLK2管脚 100: 无效选择 101: 计数器时钟源为计数器A的CAOUT 110: 无效选择 111: 计数器时钟源为IMOSC

注意：

- 定时器/计数器时钟使能后，用户不能修改CLKSRC位，所以在修改CLKSRC前，用户必须先禁止时钟。

10.3.1.3 TC_CEDR (时钟使能/禁止寄存器)

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

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

Name	Bit	Type	Description
CLKEN	[0]	RW	时钟使能/禁止控制位 0: 禁止计数器时钟 1: 使能计数器时钟 SWRST不影响CLKEN位的状态
DBGEN	[31]	RW	Debug模式使能/禁止控制位 0: 禁止Debug模式 1: 使能Debug模式 如果DBGEN置1，当CPU在debug模式被暂停后，计数器计数将停止。

注意：

- 设置其它寄存器(除了CSSR)之前，必须要先将CLKEN置1。如果CLKEN为0，那么其它寄存器(除了CSSR)的写操作都无效。但读寄存器，写DBGEN和SWRST位跟CLKEN的状态无关。CSSR寄存器必须在CLKEN置1前设置。

10.3.1.4 TC_SRR (软件复位寄存器)

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

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

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

10.3.1.5 TC_CSR (控制使能寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		PWMEX5	PWMEX4	PWMEX3	PWMEX2	PWMEX1	PWMEX0	RSVD			CAPT_SEL		CAPT_R	CAPT_F	RSVD	HWTRIG	CNTM	REPEAT	PWMEN	PWMIM	KEEP	OUTST	IDLEST	RSVD				STOPCLEAR	STOPHOLD	UPDATE	START
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	W	W	W	W	W	W	R	R	R	W	W	W	W	W	W	W	W	W	W	W	W	W	R	R	R	R	W	W	W	W

Name	Bit	Type	Description
START	[0]	W	启动计数器
UPDATE	[1]	W	请求更新寄存器
STOPHOLD	[2]	W	停止计数后保留计数器值
STOPCLEAR	[3]	W	停止计数后清除计数器
IDLEST	[8]	W	IDLE时输出高电平
OUTST	[9]	W	输出状态为高电平
KEEP	[10]	W	停止后保持输出电平
PWMIM	[11]	W	使能输出翻转功能/禁止PWM输出功能
PWMEN	[12]	W	使能PWM输出功能
REPEAT	[13]	W	使能循环重复模式(溢出后重新开始计数)
CNTM	[14]	W	使能连续计数模式(计数到计数器的最大值)
HWTRIG	[15]	W	使能硬件启动触发(其它模块)功能
CAPT_F	[17]	W	当检测到外部输入信号的下降沿时，将当前计数值存入下降沿捕捉寄存器中。
CAPT_R	[18]	W	当检测到外部输入信号的上升沿时，将当前计数值存入上升沿捕捉寄存器中。
CAPT_SEL	[20:19]	W	捕捉输入选择 00: 捕捉输入源为TCAP0 01: 捕捉输入源为TCAP1 10: 捕捉输入源为TCAP2 11: 捕捉输入源为TCAP3
PWMEX[5:0]	[29:24]	W	PWM扩展位

注意：

- 对该寄存器写0无效

10.3.1.6 TC_CCR (控制清除寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		PWMEX5	PWMEX4	PWMEX3	PWMEX2	PWMEX1	PWMEX0	RSVD			CAPT_SEL		CAPT_R	CAPT_F	RSVD	HWTRIG	CNTM	REPEAT	PWMEN	PWMIM	KEEP	OUTST	IDLEST	RSVD				STOPCLEAR	STOPHOLD	UPDATE	START
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	W	W	W	W	W	W	R	R	R	W	W	W	W	W	W	W	W	W	W	W	W	W	R	R	R	R	W	W	W	W

Name	Bit	Type	Description
START	[0]	W	停止计数器
UPDATE	[1]	W	完成更新寄存器
STOPHOLD	[2]	W	停止计数后不保留计数器值
STOPCLEAR	[3]	W	停止计数后不清除计数器
IDLEST	[8]	W	IDLE时输出低电平
OUTST	[9]	W	输出状态为低电平
KEEP	[10]	W	停止后不保持输出电平
PWMIM	[11]	W	禁止输出翻转功能/ 使能PWM输出功能
PWMEN	[12]	W	禁止PWM输出功能
REPEAT	[13]	W	禁止循环重复模式
CNTM	[14]	W	禁止连续计数模式
HWTRIG	[15]	W	禁止硬件启动触发(其它模块)功能
CAPT_F	[17]	W	禁止下降沿捕捉功能
CAPT_R	[18]	W	禁止上升沿捕捉功能
CAPT_SEL	[20:19]	W	捕捉输入选择 00: 捕捉输入源为TCAP0 01: 捕捉输入源为TCAP1 10: 捕捉输入源为TCAP2 11: 捕捉输入源为TCAP3
PWMEX[5:0]	[29:24]	W	PWM扩展位

注意：

- 对该寄存器写0无效

10.3.1.7 TC_SR (控制状态寄存器)

- CCR: Address = Base Address + 0x0018, Reset Value = 0x0000_0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		PWMEX5	PWMEX4	PWMEX3	PWMEX2	PWMEX1	PWMEX0	RSVD					CAPT_R	CAPT_F	RSVD	HWTRIG	CNTM	REPEAT	PWMEN	PWMIM	KEEP	OUTST	IDLEST	RSVD				STOPCLEAR	STOPHOLD	UPDATE	START
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
START	[0]	R	0: 计数器处于停止状态 1: 计数器处于工作状态
UPDATE	[1]	R	0: 寄存器更新被禁止 1: 允许更新寄存器
STOPHOLD	[2]	R	0: 停止后保留计数值模式被禁止 1: 停止后保留计数值模式被使能 当STOPHOLD置1时，计数器停止后会保留当前计数值和当前输出状态，所以当计数器再次启动后，计数会从保留的状态继续开始
STOPCLEAR	[3]	R	0: 停止后清除计数值模式被禁止 1: 停止后清除计数值模式被使能 当STOPCLEAR置1时，计数器停止后会清除计数值，输出电平由IDLEST决定
IDLEST	[8]	R	0: Idle状态下输出电平为低 1: Idle状态下输出电平为高
OUTST	[9]	R	0: 计数开始的时候输出电平为低 1: 计数开始的时候输出电平为高
KEEP	[10]	R	0: 保持状态模式被禁止 1: 保持状态模式被使能 不管IDLEST为0还是1，计数停止后都保持输出电平的状态
PWMIM	[11]	R	0: PWM输出 1: 翻转输出
PWMEN	[12]	R	0: 禁止PWM输出 1: 使能PWM输出
REPEAT	[13]	R	0: 禁止循环重复模式 1: 使能循环重复模式 使能该模式后，计数器在连续计数模式溢出或者周期模式周期结束后，会自动重新开始计数

<table>
<tr><td>CNTM</td><td>[14]</td><td>R</td><td>0: 工作在周期模式，计数值自增直到周期结束
1: 工作在连续计数模式，计数值自增直到溢出</td></tr>
<tr><td>HWTRIG</td><td>[15]</td><td>R</td><td>0: 硬件自动触发功能被禁止
1: 硬件自动触发功能被使能</td></tr>
<tr><td>CAPT_F</td><td>[17]</td><td>R</td><td>下降沿触发捕捉
0: 外部输入信号下降沿捕捉被禁止
1: 外部输入信号下降沿捕捉被使能
当检测到外部输入信号的下降沿时，将当前计数值存入下降沿捕捉寄存器中。</td></tr>
<tr><td>CAPT_R</td><td>[18]</td><td>R</td><td>上升沿触发捕捉
0: 外部输入信号上升沿捕捉被禁止
1: 外部输入信号上升沿捕捉被使能
当检测到外部输入信号的上升沿时，将当前计数值存入上升沿捕捉寄存器中。</td></tr>
<tr><td>PWMEX[5:0]</td><td>[29:24]</td><td>R</td><td>PWM扩展位
<table>
<tr><td>PWMEX</td><td>“拉伸”的周期数</td></tr>
<tr><td>PWMEX0</td><td>32</td></tr>
<tr><td>PWMEX1</td><td>16, 48</td></tr>
<tr><td>PWMEX2</td><td>8, 24, 40, 56</td></tr>
<tr><td>PWMEX3</td><td>4, 12, 20, 28, 36 , 44, 52, 60</td></tr>
<tr><td>PWMEX4</td><td>2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62</td></tr>
<tr><td>PWMEX5</td><td>1, 3, 5, 7, 9,11,13,15,17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63</td></tr>
</table></td></tr>
</table>

10.3.1.8 TC_IMSCR (中断使能/禁止寄存器)

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

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

Name	Bit	Type	Description
STARTI	[0]	RW	计数启动中断 0: 禁止该中断 1: 使能该中断
STOPI	[1]	RW	计数停止中断 0: 禁止该中断 1: 使能该中断
PSTARTI	[2]	RW	周期开始中断 0: 禁止该中断 1: 使能该中断
PENDI	[3]	RW	周期结束中断 0: 禁止该中断 1: 使能该中断
MATI	[4]	RW	脉冲匹配中断 0: 禁止该中断 1: 使能该中断
OVFI	[5]	RW	溢出中断 0: 禁止该中断 1: 使能该中断
CAPTI	[6]	RW	捕捉中断 0: 禁止该中断

			1: 使能该中断

10.3.1.9 TC_ RISR (原始中断状态寄存器)

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

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

Name	Bit	Type	Description
STARTI	[0]	R	计数启动中断的原始状态(即使该中断没有使能，也会置位)
STOPI	[1]	R	计数停止中断的原始状态(即使该中断没有使能，也会置位)
PSTARTI	[2]	R	周期开始中断的原始状态(即使该中断没有使能，也会置位)
PENDI	[3]	R	周期结束中断的原始状态(即使该中断没有使能，也会置位)
MATI	[4]	R	脉冲匹配中断的原始状态(即使该中断没有使能，也会置位)
OVFI	[5]	R	溢出中断的原始状态(即使该中断没有使能，也会置位)
CAPTI	[6]	R	捕捉中断的原始状态(即使该中断没有使能，也会置位)

注意：

- 0: 中断没有发生; 1: 中断已发生

10.3.1.10 TC_ MISR (中断状态寄存器)

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

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

Name	Bit	Type	Description
STARTI	[0]	R	计数启动中断的状态(该中断使能后才会置位)
STOPI	[1]	R	计数停止中断的状态(该中断使能后才会置位)
PSTARTI	[2]	R	周期开始中断的状态(该中断使能后才会置位)
PENDI	[3]	R	周期结束中断的状态(该中断使能后才会置位)
MATI	[4]	R	脉冲匹配中断的状态(该中断使能后才会置位)
OVFI	[5]	R	溢出中断的状态(该中断使能后才会置位)
CAPTI	[6]	R	捕捉中断的状态(该中断使能后才会置位)

注意：

- TC_MISR = TC_IMSCR & TC_RISR
- 读该寄存器返回当前相应中断的状态，写操作无效。
- 0: 中断没有发生; 1: 中断已发生

10.3.1.11 TC_ ICR (中断状态清除寄存器)

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

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

Name	Bit	Type	Description
STARTI	[0]	W	0: 无效 1: 清除计数启动中断
STOPI	[1]	W	0: 无效 1: 清除计数停止中断
PSTARTI	[2]	W	0: 无效 1: 清除周期开始中断
PENDI	[3]	W	0: 无效 1: 清除周期结束中断
MATI	[4]	W	0: 无效 1: 清除脉冲匹配中断
OVFI	[5]	W	0: 无效 1: 清除溢出中断
CAPTI	[6]	W	0: 无效 1: 清除捕捉中断

10.3.1.12 TC_CDR (时钟分频寄存器)

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

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

Name	Bit	Type	Description
DIVN	[3:0]	RW	时钟分频器DIVN的值
DIVM	[11:4]	RW	时钟分频器DIVM的值

注意：

- 当DIVN不等于0的时候，禁止将DIVM设成0

10.3.1.13 TC_CSMR (计数器位数控制寄存器)

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

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

Name	Bit	Type	Description
SIZE	[4:0]	RW	设置计数器位数，例如： 如果SIZE为0x07，那么定时器/计数器是一个8位的定时器/计数器 如果SIZE为0x09，那么定时器/计数器是一个10位的定时器/计数器 如果SIZE为0x0F，那么定时器/计数器是一个16位的定时器/计数器 如果SIZE为0x1F，那么定时器/计数器是一个32位的定时器/计数器

10.3.1.14 TC_PRDR (周期值寄存器)

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

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

Name	Bit	Type	Description
PERIOD	[31:0]	RW	PERIOD的值

注意：由于该计数器在周期匹配或者溢出后，都是从1开始计数的，所以该寄存器的值不能设置为0，否则计数器无法正常工作。

10.3.1.15 TC_PULR (脉冲比较值寄存器)

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

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

Name	Bit	Type	Description
PULSE	[31:0]	RW	PULSE的值

注意：由于该计数器在周期匹配或者溢出后，都是从1开始计数的，所以该寄存器的值不能设置为0，否则计数器无法正常工作。

10.3.1.16 TC_CUCR (捕捉上升沿计数寄存器)

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

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

Name	Bit	Type	Description
COUNT	[31:0]	R	检测到上升沿时的计数值

10.3.1.17 TC_CDCR (捕捉下降沿计数寄存器)

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

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

Name	Bit	Type	Description
COUNT	[31:0]	R	检测到下降沿时的计数值

10.3.1.18 TC_CVR (当前计数值寄存器)

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

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

Name	Bit	Type	Description
COUNT	[31:0]	R	当前计数值

注意：

当时钟源是一个异步时钟时，无法保证该寄存器的值为当前完全正确的值(由于同步逻辑的存在，可能有一定的误差

11 计数器A

11.1 概述

本章节介绍计数器 A，一个 16 位的计数器，用来产生载波频率。

11.1.1 主要特性

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

注意：
CPU 时钟必须比计数器 A 的时钟快。

11.2 功能描述

11.2.1 模块框图

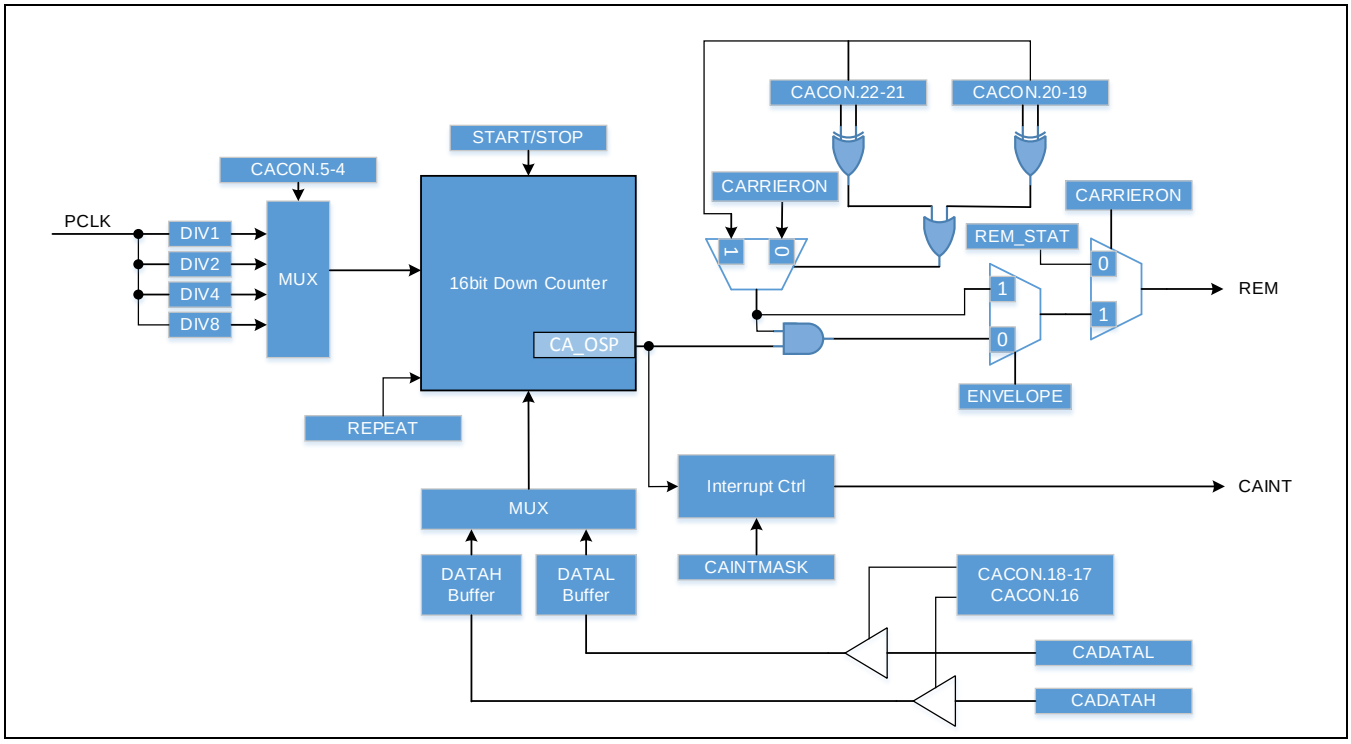


Figure 11-1 计数器A模块框图

11.2.2 工作原理

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

Table 11-1 REM管脚输出状态

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

注意：

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

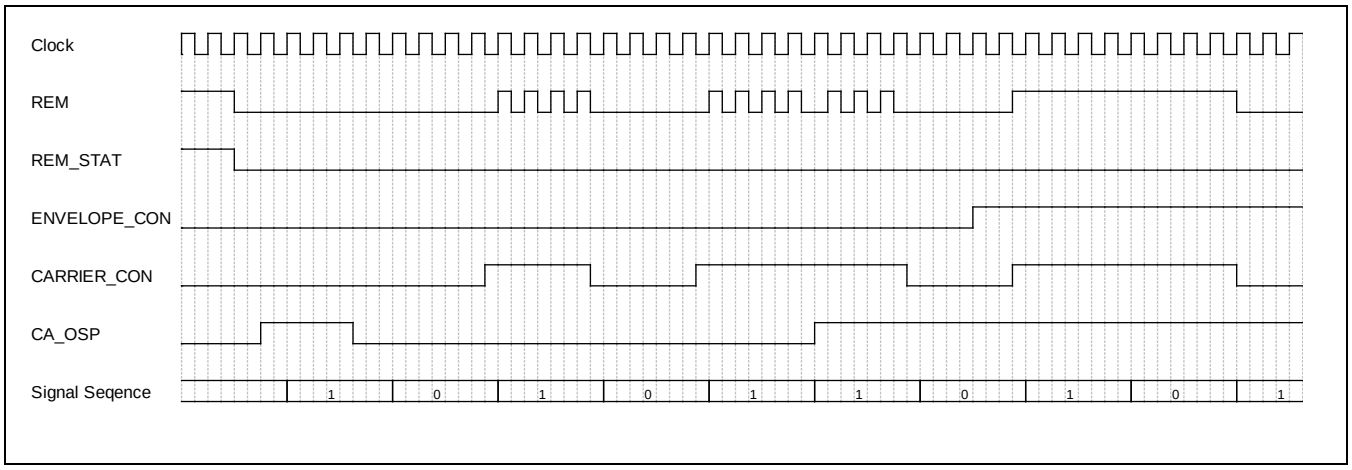


Figure 11-2 计数器A输出信号波形

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

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

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

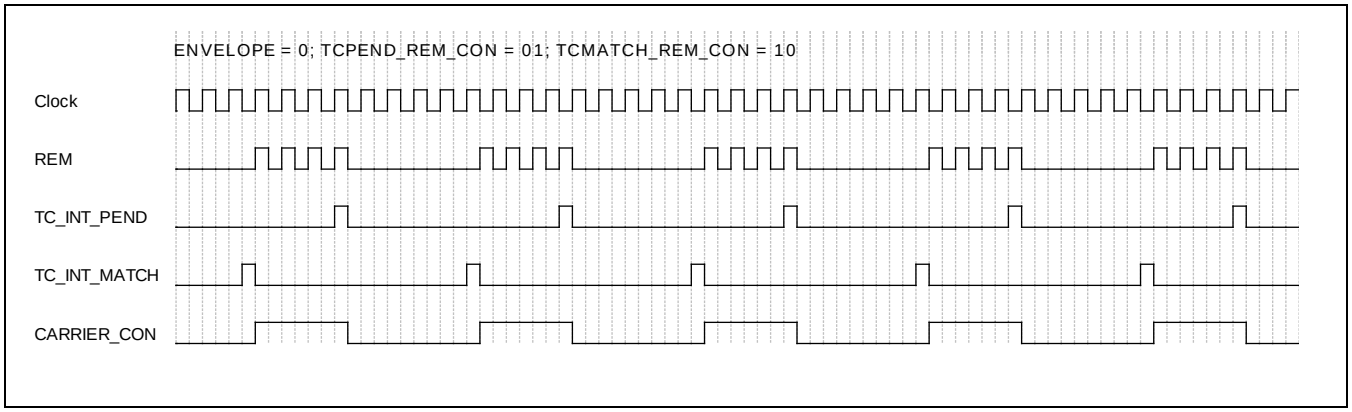


Figure 11-3 硬件触发的波形

11.2.3 脉冲宽度计算

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

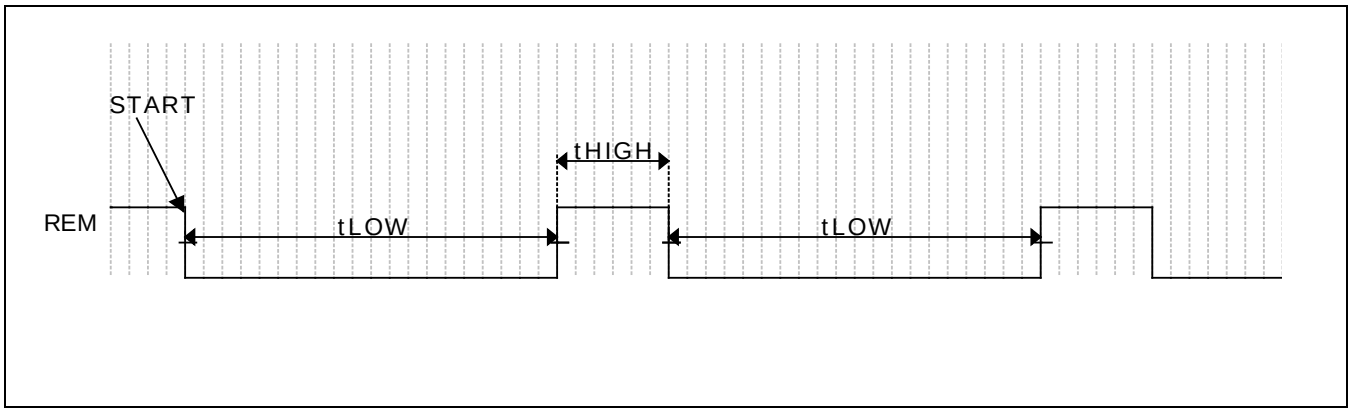


Figure 11-4 重复模式波形

当 CA_OSP = 0,

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

当 CA_OSP = 1,

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

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

[方法 1] 当 CA_OSP = 0,

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

[方法 2] 当 CA_OSP = 1,

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

11.2.4 中断

计数器 A 产生 2 个中断。

- 低电平结束中断
- 高电平结束中断

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

11.2.5 编程提示

下面的例子演示如何产生一个 38KHz, 1/3 占空比的信号。

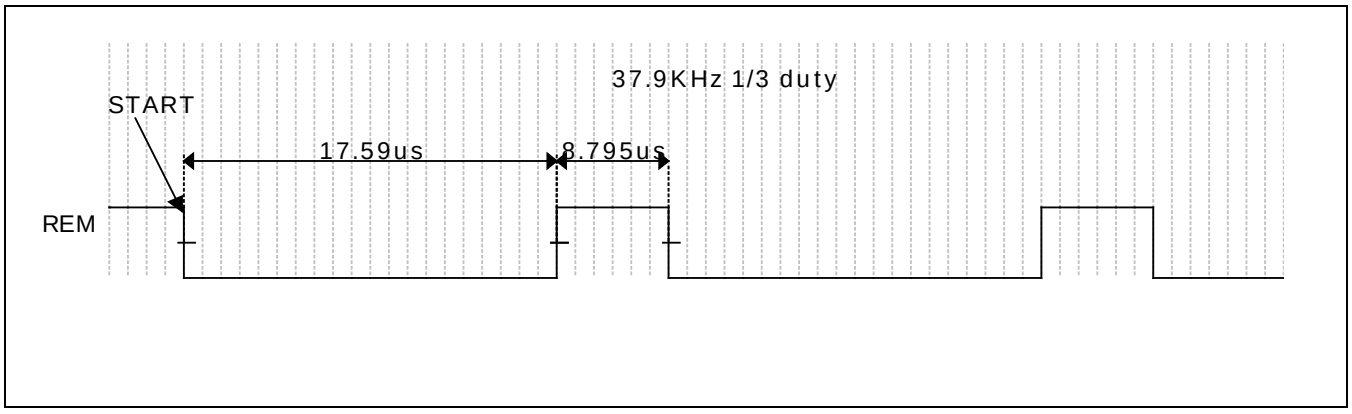


Figure 11-5 38KHz, 1/3占空比的REM输出波形

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

11.3 寄存器说明

11.3.1 寄存器表

- Base Address: 0x4007_0000

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

注意：

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

11.3.2 CADATAH (DATAH寄存器)

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

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

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

11.3.3 CADATAL (DATAL寄存器)

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

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

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

11.3.4 CACON (控制寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD						CARRIERON	ENVELOPE	REM_STAT	TCPEND_REM_CON		TCMATCH_REM_CON		HW_STROBE_DATA		SW_STROBE_DATA	RSVD										CA_CLK		STOP	START	MODE	OSP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description
CARRIERON	[25]	RW	载波信号控制位 0: 关闭载波 1: 打开载波
ENVELOPE	[24]	RW	REM输出信号选择位 0: 选择载波信号为输出 1: 选择包络信号为输出
REM_STAT	[23]	RW	当关闭载波时REM的输出状态 0: 低 1: 高
TCPEND_REM_CON	[22:21]	RW	当TC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: TC周期结束中断发生时，CARRIERON位会被硬件自动清零 10: TC周期结束中断发生时，CARRIERON位会被硬件自动置位
TCMATCH_REM_CON	[20:19]	RW	当TC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: TC脉冲匹配中断发生时，CARRIERON位会被硬件自动清零 10: TC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TC脉冲匹配中断发生时，计数值会自动载入计数器 1X: 当TC周期结束中断发生时，计数值会自动载入计数器

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

11.3.5 CAINTMASK (中断控制寄存器)

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

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

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

12 通用异步收发器 (UART)

12.1 概述

UART是一个简单通用的异步串行接收和发送接口，支持8位的数据通信，不支持校验位，每次发送都以一个停止位结束。

12.1.1 主要特性

- 可配置的波特率
- 固定的8位发送长度
- 发送接收溢出检测
- 发送接收完成中断和溢出中断

12.1.2 管脚描述

Table 12-1 UART 管脚描述

管脚名称	功能	I/O类型	有效电平	说明
UART_RX	UART发送数据线	O	–	–
UART_TX	UART接收数据线	I	–	–

12.2 功能描述

12.2.1 模块框图

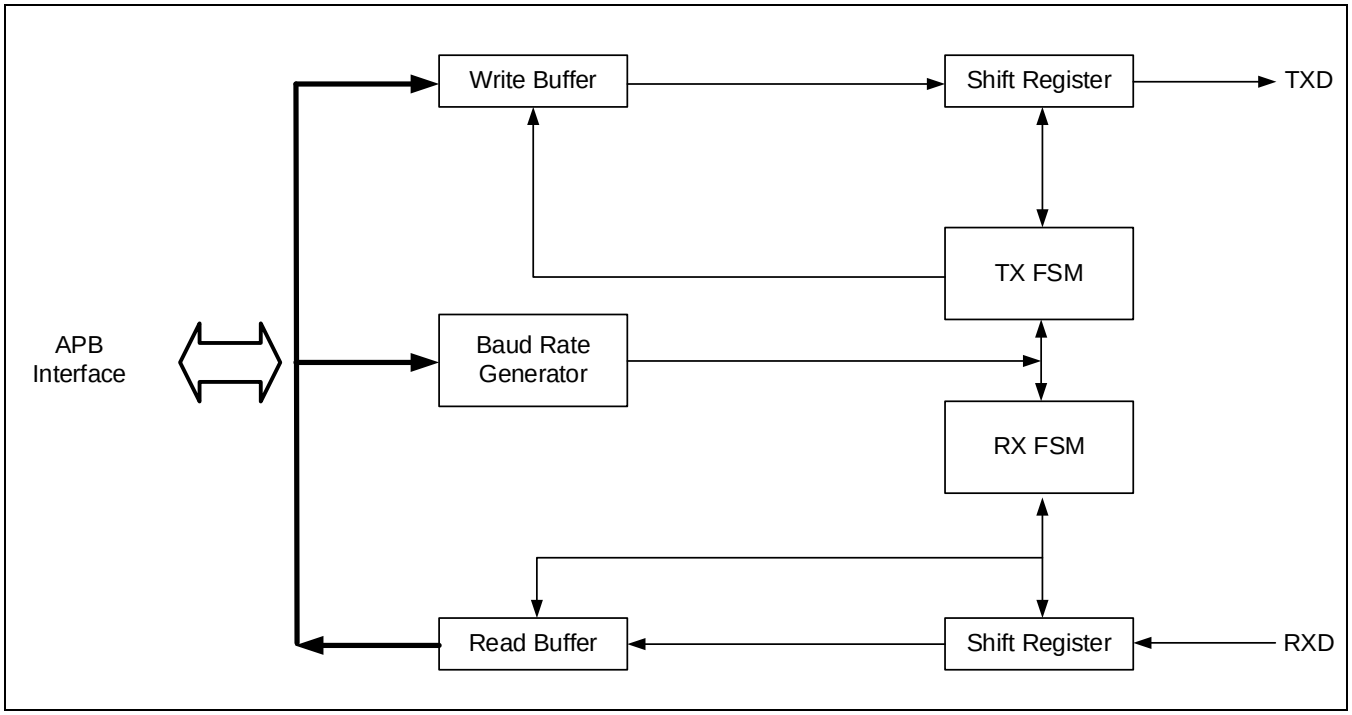


Figure 12-1 UART模块框图

12.2.2 功能说明

12.2.2.1 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 12-2 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%

12.2.2.2 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)。

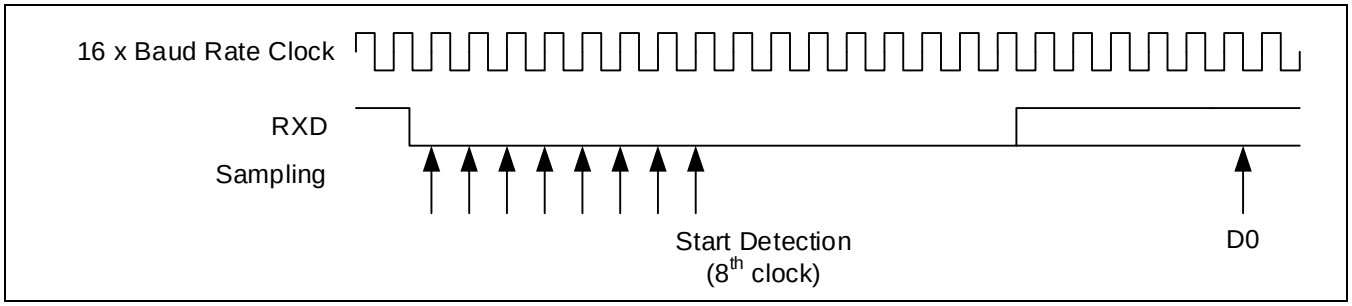


Figure 12-2 起始位检测

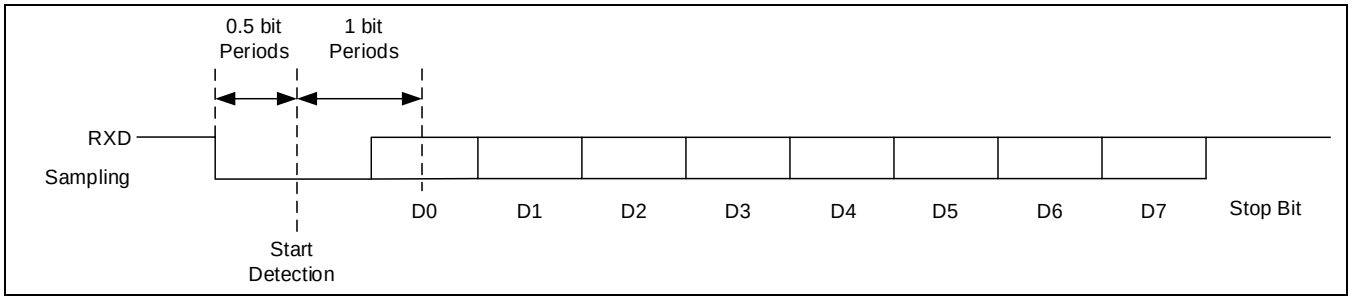


Figure 12-3 接收数据

12.2.2.3 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 12-4 数据发送

12.2.2.4 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。

12.3 寄存器说明

12.3.1 寄存器表

Base Address: 0x4008_0000

Offset Address	Name	Description	R/W	Reset State
0x000	UART_DATA	数据寄存器	R/W	0x00000000
0x004	UART_SR	状态寄存器	R/W	0x00000000
0x008	UART_CTRL	控制寄存器	R/W	0x00000000
0x00C	UART_ISR	中断状态寄存器	R/W	0x00000000
0x010	UART_BRDIV	波特率分频寄存器	R/W	0x00000000

12.3.1.1 UART_DATA (数据寄存器)

- Address = Base Address + 0x0000

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

Name	Bit	Type	Description	Reset Value
DATA	[7:0]	R/W	发送或接收到的数据 读 = 接收到的数据 写 = 发送的数据	-

12.3.1.2 UART_SR (状态寄存器)

- Address = Base Address + 0x0004

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

Name	Bit	Type	Description	Reset Value
TX_FULL	[0]	R	TX缓冲区状态 0 = TX缓冲区没有满(可以发送数据) 1 = TX缓冲区已满(正在发送数据)	0
RX_FULL	[1]	R	RX缓冲区状态 0 = RX缓冲区没有满(未收到数据或数据已被读取) 1 = RX缓冲区已满(收到数据，并且未被读取)	0
TX_OVER	[2]	R/W	TX缓冲区溢出状态 0 = TX缓冲区没有溢出 1 = TX缓冲区溢出(读取) 1 = 清除TX缓冲区溢出标志(写)	0
RX_OVER	[3]	R/W	RX缓冲区溢出状态 0 = RX缓冲区没有溢出 1 = RX缓冲区溢出(读取) 1 =清除RX缓冲区溢出标志(写)	0

12.3.1.3 UART_CTRL (控制寄存器)

- Address = Base Address + 0x0008

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

Name	Bit	Type	Description	Reset Value
TX	[0]	R/W	TX 使能/禁止 0 = 禁止TX 1 = 使能TX	0
RX	[1]	R/W	RX 使能/禁止 0 = 禁止RX 1 = 使能RX	0
INT_TX	[2]	R/W	TX 中断使能/禁止 0 = 禁止TX中断 1 = 使能TX中断	0
INT_RX	[3]	R/W	RX 中断使能/禁止 0 = 禁止RX中断 1 = 使能RX中断	0
INT_OVER_TX	[4]	R/W	TX溢出中断使能/禁止 0 = 禁止TX溢出中断 1 = 使能TX溢出中断	0
INT_OVER_RX	[5]	R/W	RX溢出中断使能/禁止 0 = 禁止RX溢出中断 1 = 使能RX溢出中断	0
TEST	[6]	R/W	测试模式 此为请保持为0	0

12.3.1.4 UART_ISR (中断状态寄存器)

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

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

Name	Bit	Type	Description	Reset Value
TX_INT	[0]	RW	TX中断 0 = TX中断没发生 1 = TX中断发生(读取) 1 = 清除TX中断(写)	0
RX_INT	[1]	RW	RX中断 0 = RX中断没发生 1 = RX中断发生(读取) 1 = 清除RX中断(写)	0
TX_OVER_INT	[2]	RW	TX溢出中断 0 = TX溢出中断没发生 1 = TX溢出中断发生(读取) 1 = 清除TX溢出中断(写)	0
RX_OVER_INT	[3]	RW	RX溢出中断 0 = RX溢出中断没发生 1 = RX溢出中断发生(读取) 1 = 清除RX溢出中断(写)	0

12.3.1.5 UART_BRDIV (波特率分频寄存器)

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

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

Name	Bit	Type	Description	Reset Value
DIV	[19:0]	RW	波特率分频 最小值为16	0x00000

13 I2C总线

13.1 概述

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

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

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

13.1.1 主要特性

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

13.1.2 管脚描述

Table 13-1 I2C 管脚描述

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

13.2 功能描述

13.2.1 模块框图

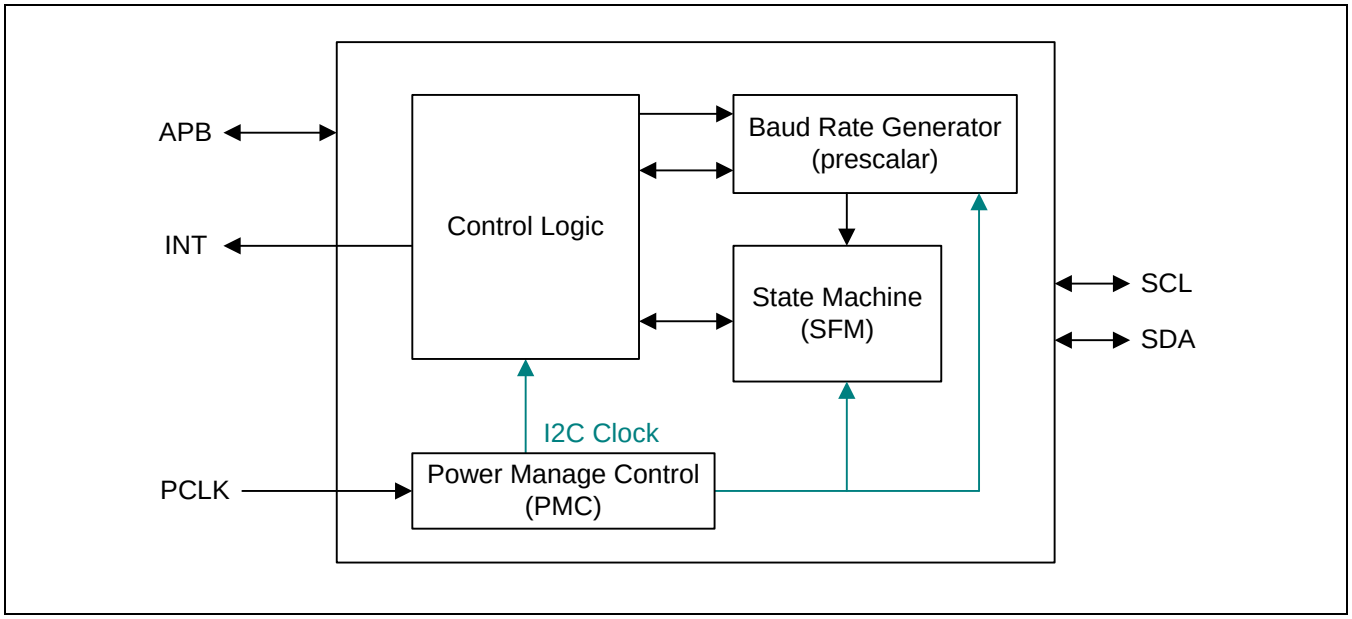


Figure 13-1 I2C模块框图

13.2.2 功能介绍

13.2.2.1 I2C总线概念

13.2.2.1.1 协议概念

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

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

1: 假设单片机A希望给单片机B发送信息

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

2: 如果单片机A希望从单片机B接收数据

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

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

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

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

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

13.2.2.1.2 一般特性

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

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

Table 13-2 波特率设置示例

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

13.2.2.2 位传输

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

13.2.2.2.1 数据有效性

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

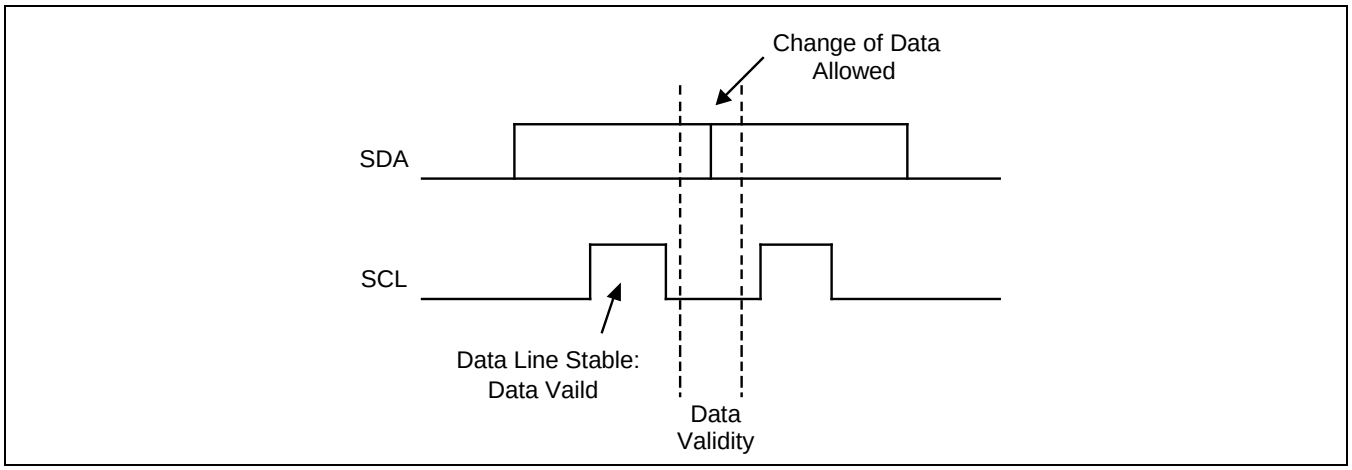


Figure 13-2 数据有效性

13.2.2.2.2 起始位和停止位

在I2C总线的传输过程中，一些特殊的情况被定义成起始位和停止位。

当SCL是高的时候，SDA从高变低，被定义为起始位。

当SCL是高的时候，SDA从低变高，被定义为停止位。

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

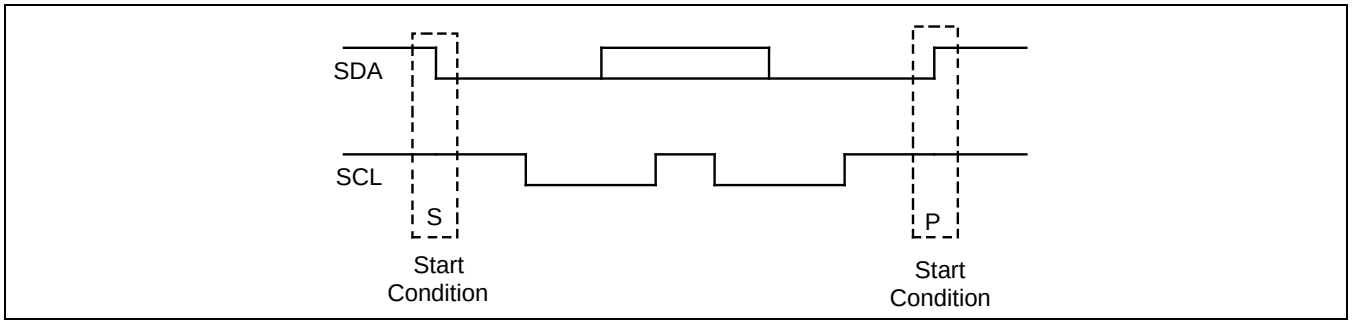


Figure 13-3 起始位和停止位

13.2.2.3 数据传输

13.2.2.3.1 传输字节格式

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

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

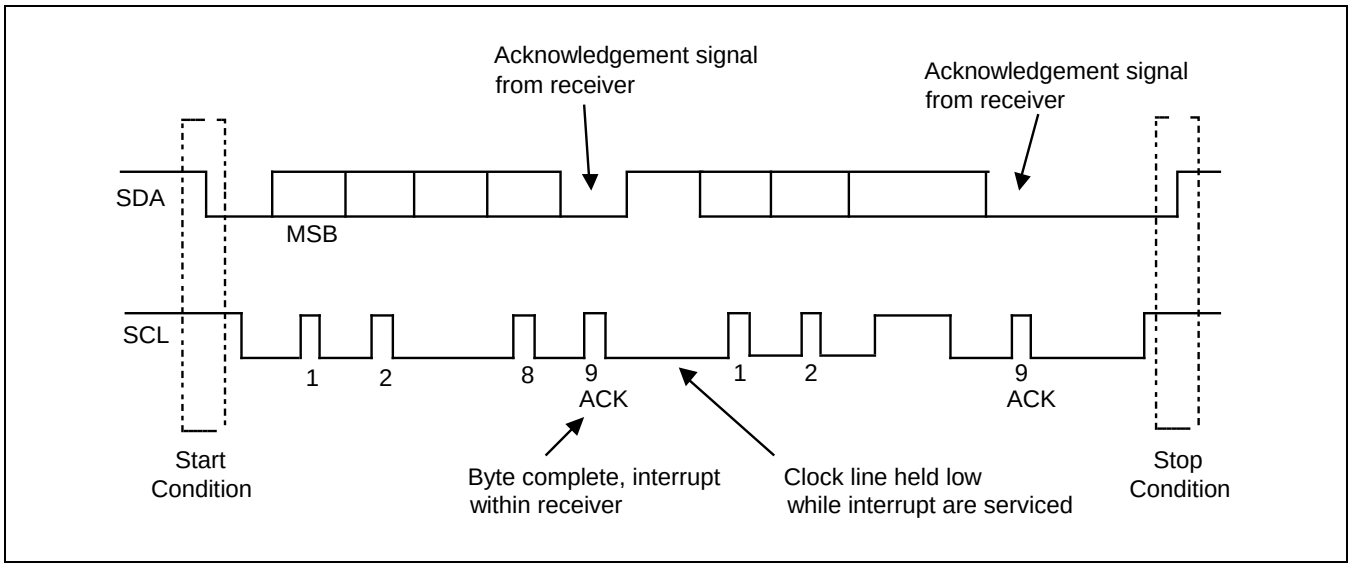


Figure 13-4 I2C总线的数据传输

13.2.2.3.2 应答

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

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

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

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

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

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

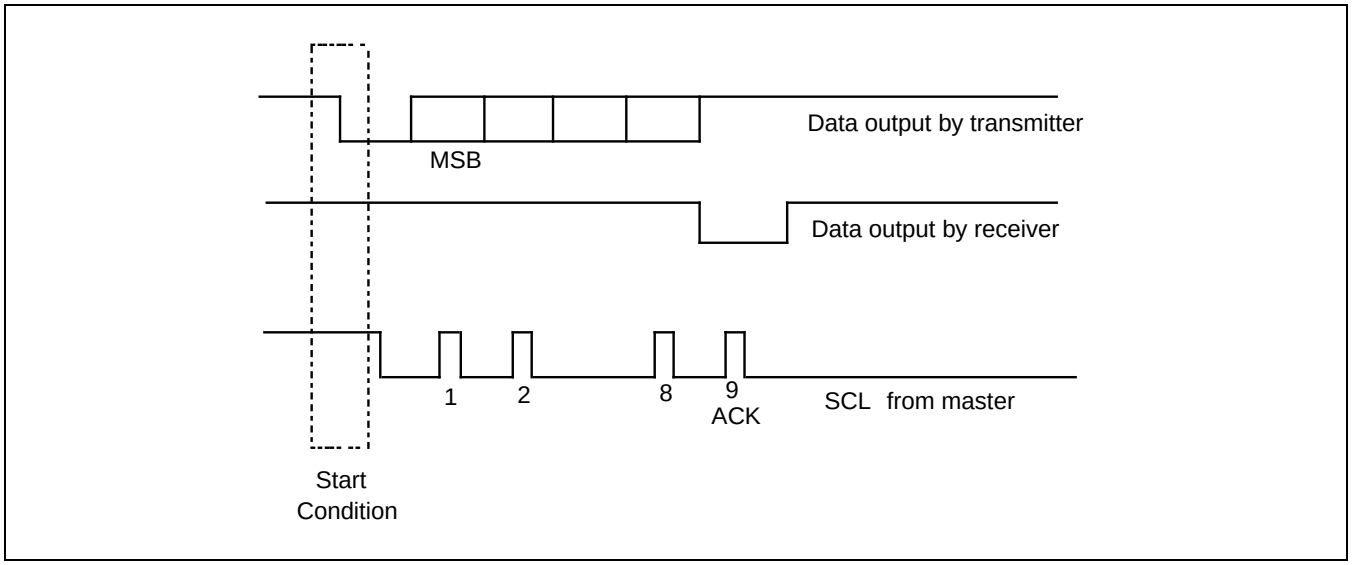


Figure 13-5 应答

13.2.2.4 时钟仲裁

13.2.2.4.1 同步

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

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

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

13.2.2.4.2 仲裁

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

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

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

失去仲裁的主机会在丢失仲裁的那个字节传输中一直产生时钟脉冲。

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

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

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

- 重复起始位和数据位
- 停止位和数据位
- 重复起始位和停止位

13.2.2.4.3 使用时钟同步机制作为握手

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

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

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

13.2.2.4.4 7位寻址格式

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

可以传输的格式为：

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

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

停止位由主机产生。

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

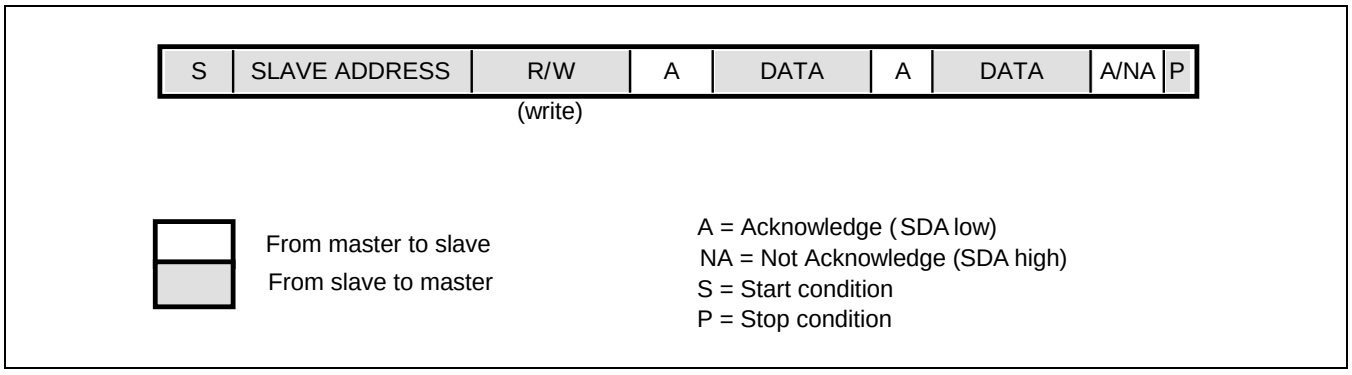


Figure 13-6 主机-发送端寻址从机

13.2.2.5 7位寻址

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

13.2.2.5.1 定义第一个字节的各个位

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

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

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

I2C总线协议委员会负责协调I2C地址的分配。

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

Table 13-3 第一个字节定义

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

注意：

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

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

第二个字节和后面的字节都会被能处理该数据的从机-接收端应答。

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

需要考虑两种情况：

- LSB最低位B是0
- LSB最低位B是1

当最低位B是0，那么第二个字节有以下定义：

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

剩下所有的代码组合都还没有定义，并且所有器件都必须忽略它们。

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

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

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

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

13.2.2.5.2 起始字节

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

在这种情况下，数据传输可以由一个比通常时间要长的起始过程来进行。

这个起始过程由下面几个步骤组成：

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

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

硬件接收端在收到重复起始位Sr后会复位，所以会忽略起始字节。

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

13.2.3 I2C总线规范的扩展

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

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

扩展I2C总线规范有两个原因：

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

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

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

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

13.2.4 1 快速模式

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

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

13.2.4.1 10位寻址

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

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

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

A – 头两个字节的位定义

10位地址由起始位(S)或者重复起始位(Sr)后的头两个字节组成。

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

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

B – 10位寻址方式

10位寻址的传输中可能包含各种读写的组合。可能涉及到的数据传输格式有：

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

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

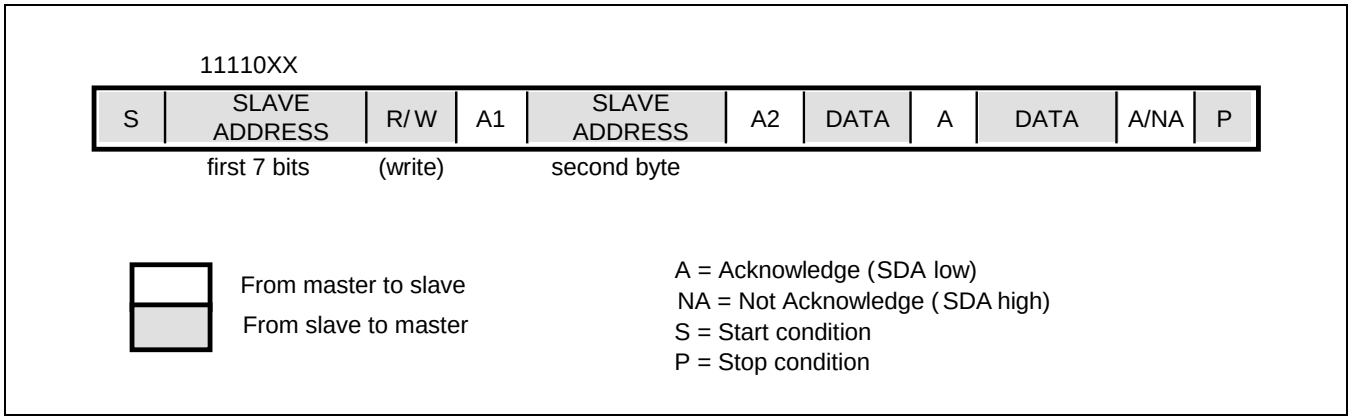


Figure 13-7 主机-发送端用10位地址寻址从机-接收端

13.2.4.2 General Call寻址和起始字节

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

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

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

10位寻址中起始字节00000001 (H’01’)的产生跟7位寻址一样。

13.3 I2C时序

Table 13-4 时序要求

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

13.4 寄存器说明

13.4.1 寄存器表 (Base Address: 0x400A_0000)

Offset Address	Name	Description	R/W	Reset State
0x000 ~ 0x04C	–	Reserved	–	–
0x050	I2C_ECR	时钟使能寄存器	W	–
0x054	I2C_DCR	时钟禁止寄存器	W	–
0x058	I2C_PMSR	电源管理状态寄存器	R	–
0x05C	–	Reserved	–	–
0x060	I2C_CR	控制寄存器	R/W	0x00000000
0x064	I2C_MR	模式寄存器	R/W	0x000001F4
0x068	–	Reserved	–	–
0x06C	–	Reserved	–	–
0x070	I2C_SR	状态寄存器	R	0x000000F8
0x074	I2C_IER	中断使能寄存器	W	–
0x078	I2C_IDR	中断禁止寄存器	W	–
0x07C	I2C_IMR	中断状态寄存器	R	0x00000000
0x080	I2C_DAT	数据寄存器	R/W	0x00000000
0x084	I2C_ADR	从机地址寄存器	R/W	0x00000000
0x088	I2C_THOLD	Hold/Setup 延时控制寄存器	R/W	0x00000001

13.4.1.1 I2C_ECR (时钟使能寄存器)

- Address = Base Address + 0x0050

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

Name	Bit	Type	Description	Reset Value
CLKEN	[1]	W	时钟使能控制位. 0 = 无效 1 = 使能I2C时钟	-
DBGEN	[1]	W	调试使能控制位 0 = 无效 1 = 使能I2C模块的调试功能	-

13.4.1.2 I2C_DCR (时钟禁止寄存器)

- Address = Base Address + 0x0054

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

Name	Bit	Type	Description	Reset Value
CLKEN	[1]	W	时钟禁止控制位. 0 = 无效 1 = 禁止I2C时钟	-
DBGEN	[1]	W	调试禁止控制位 0 = 无效 1 = 禁止I2C模块的调试功能	-

13.4.1.3 I2C_PMSR (电源管理状态寄存器)

- Address = Base Address + 0x0058

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

Name	Bit	Type	Description	Reset Value
CLKEN	[1]	R	CLKEN：时钟使能/禁止状态 0 = I2C时钟被禁止 1 = I2C时钟被使能	0
IPIDCODE	[29:4]	R	模块版本信息，共26位	-
DBGEN	[31]	R	DBGEN：调试模式 0 = dbgack_sclk输入对I2C功能无影响 1 = dbgack_sclk被使能。 当这位是低，I2C功能保持不变。当这位是高，I2C的功能被冻结无法使用，但是寄存器的读写功能不受影响，以方便调试。	0

13.4.1.4 I2C_CR (控制寄存器)

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

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

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

			用，没有任何输出和输入) 1 = 使能I2C接口	

13.4.1.5 I2C_MR (模式寄存器)

- Address = Base Address + 0x0064, Reset Value = 0x0000_01F4

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

Name	Bit	Type	Description	Reset Value
PRV	[11:0]	RW	预分频值 这个值用来设置总线的速度(FSCL). PRV (pre-scaler Value)的值用下面的公式来产生FSCL： FSCL = PCLK/(PRV+4)	0x1F4
FAST	[12]	RW	快速模式 0 = 禁用快速模式，使能标准模式。在这个模式下，高电平和低电平的比为1:1并且最大波特率为100kHz。 1 = 使能快速模式。在这个模式下，高电平和低电平的比为2:3并且最大波特率为400kHz。	0

Table 13-5 基于 PRV, FAST 和 PCLK 的 FSCL 值，单位 kHz

PRV (Decimal)	FAST	PCLK (MHz)	
		10	20
500	0	19.8	39.7
400	0	24.7	49.5
250	0	39.3	78.7
200	0	49	98
125	0	77.5	–
125	1	77.5	155
100	1	96	192
62.5	1	150	300
50	1	185	370
25	1	344	–

注意：

1. PCLK 的频率至少是 FSCL 频率的 6 倍(用于 SCL 同步+状态机)。
2. PRV 值复位后是’500’ (十进制)。
3. PRV 不可以写’000’ (十六进制)。

13.4.1.6 I2C_SR (状态寄存器)

- Address = Base Address + 0x0070, Reset Value = 0x0000_00F8

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

Name	Bit	Type	Description	Reset Value
SR	[7:3]	R	I2C状态代码 接口状态代码，共有27种可能的状态 I2C_SR复位值是0x000000F8. 当I2C_SR的值是这个复位值时，表明当前没有任何相关信息。所有其它状态值都跟某个工作状态有关。当I2C接口的状态机执行到某个状态时，这个寄存器的值也会更新到该状态代码，并且SI中断位会被置1。 所有这些状态代码的意义，软件执行的下一个动作以及I2C接口执行的下一个动作，都在下一页中描述。	0x1F

Table 13-6 主机-发送模式的状态码

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

Table 13-7 主机-接收模式状态码

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

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

Table 13-8 从机-接收模式状态码

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

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

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

Table 13-9 从机-发送模式状态码

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

状态代码	代码意义	软件执行的下一个动作					I2C接口执行的下一个动作
		STA	STO	AA	SI		
0x0000_00C8	最后一个字节的数据已经被成功发送，并且收到了ACK	0	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零	同上
		0	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1	
		1	0	0	0	将I2C_CR中的SI清零，将I2C_CR中的AA清零，将I2C_CR中的STA置1	
		1	0	1	0	将I2C_CR中的SI清零，将I2C_CR中的AA置1，将I2C_CR中的STA置1	

Table 13-10 其它状态码

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

13.4.1.7 I2C_IER (中断使能寄存器)

- Address = Base Address + 0x0074

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

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

13.4.1.8 I2C_IDR (中断禁止寄存器)

- Address = Base Address + 0x0078

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

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

13.4.1.9 I2C_IMR (中断状态寄存器)

- Address = Base Address + 0x007C, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
SI	[4]	R	SI使能后的中断状态 当这位是1时，表示有中断需要处理，这时i2c_int为高，SCL传输线则被拉低，传输被暂停，直到SI被处理完并清零。	0

读取这个寄存器返回的是中断被使能后的状态，写寄存器无效。

13.4.1.10 I2C_SDR (数据寄存器)

- Address = Base Address + 0x0080, Reset Value = 0x0000_0000

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

Name	Bit	Type	Description	Reset Value
DAT	[7:0]	RW	I2C数据. 在接收模式下，该字节是从I2C总线上收到的数据；在发送模式下，该字节是需要发送到I2C总线上的数据。 DAT总是从右向左移，也就是说，在发送模式下，总是先发送最高位MSB，而在接收模式下，也是先接收到最高位MSB。 在一个发送过程中(该模块往I2C总线发送数据)，当数据被移位出去的时候，SDA传输线上的数据同时也被移位进来。所以在丢失仲裁的情况下，DAT将会包含正确的数据字节(从总线上读到的值)。	0x00

13.4.1.11 I2C_ADR (从机地址寄存器)

- Address = Base Address + 0x0084, Reset Value = 0x0000_0000

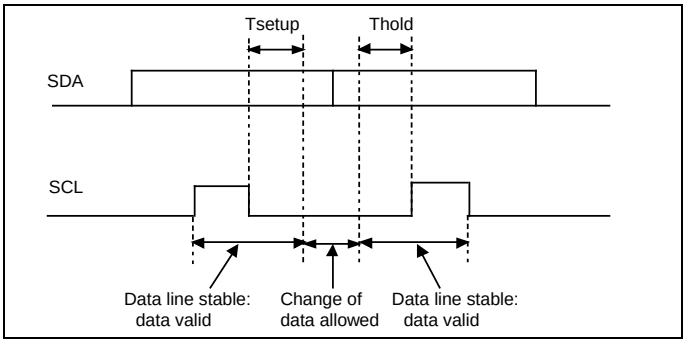
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																								ADR							GC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	Type	Description	Reset Value
GC	[0]	RW	General Call. 使能对general call地址的响应功能，当GC位置1时，如果识别到general call地址，I2C接口会产生中断。	0
ADR	[7:1]	RW	I2C地址 包含一个7位的I2C地址，如果I2C接口被设定为一个从机(发送端或者接收端)，那么将会响应这个从机地址。	0x00

13.4.1.12 I2C_THOLD (Hold/Setup延时控制寄存器)

- Address = Base Address + 0x0088, Reset Value = 0x0000_0001

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

Name	Bit	Type	Description	Reset Value
DL	[7:0]	RW	Hold/setup延时. Hold/Setup延时的值，由下面公式计算： THOLD = DL[7-0] × PCLK, TSETUP = DL[7-0] × PCLK  **Figure 1-8 Hold/Setup延时** 注意： 1. 复位后的DL值为'1' (十六进制). 2. Setup延时(TSETUP) 必须至少为250 ns (标准模式)，至少为100 ns (快速模式). 3. I2C器件必须在内部保证SDA信号上至少有300ns的hold时间。用户必须保证正确的hold值来满足慢速器件的时序。 4. DL的值不允许为0。	0x01

14 电容式触摸按键传感器

14.1 概述

此MCU内嵌了一个最大支持12个扫描通道的电容式触摸按键检测模块。该模块支持两种不同的电容检测工作原理，分别为基于张弛振荡器的电容检测和基于电荷平均分配的检测，以满足不同应用条件下电容触摸检测。

14.1.1 特性

- 最大支持12通道按键检测
  - 张弛振荡模式下，最大支持12路按键
  - 电荷转移模式下，最大支持11路按键
  - 张弛振荡模式下，支持8通道的同时扫描，加快扫描响应时间
- 内嵌独立的硬件算法模块，支持硬件自动按键检测。
  - 灵活的硬件算法配置模块
  - 很强的抗干扰能力
- 自适应的扫描速度调节，优化功耗。
- 多种扫描触发模式。
  - 软件触发
  - 硬件自动间隔触发
  - iWDT中断触发
  - LED 扫描中断触发
- 多种按键中断模式和异常处理中断。

14.1.2 管脚描述

Table 14-1 TOUCH KEY 管脚描述

Pin Name	Function	I/O Type	Active Level	Comments
TCHx	触摸检测通道	A	-	-
ECAP	电荷转移模式下的外部电容接口	A	-	-

14.2 功能描述

14.2.1 模块框图

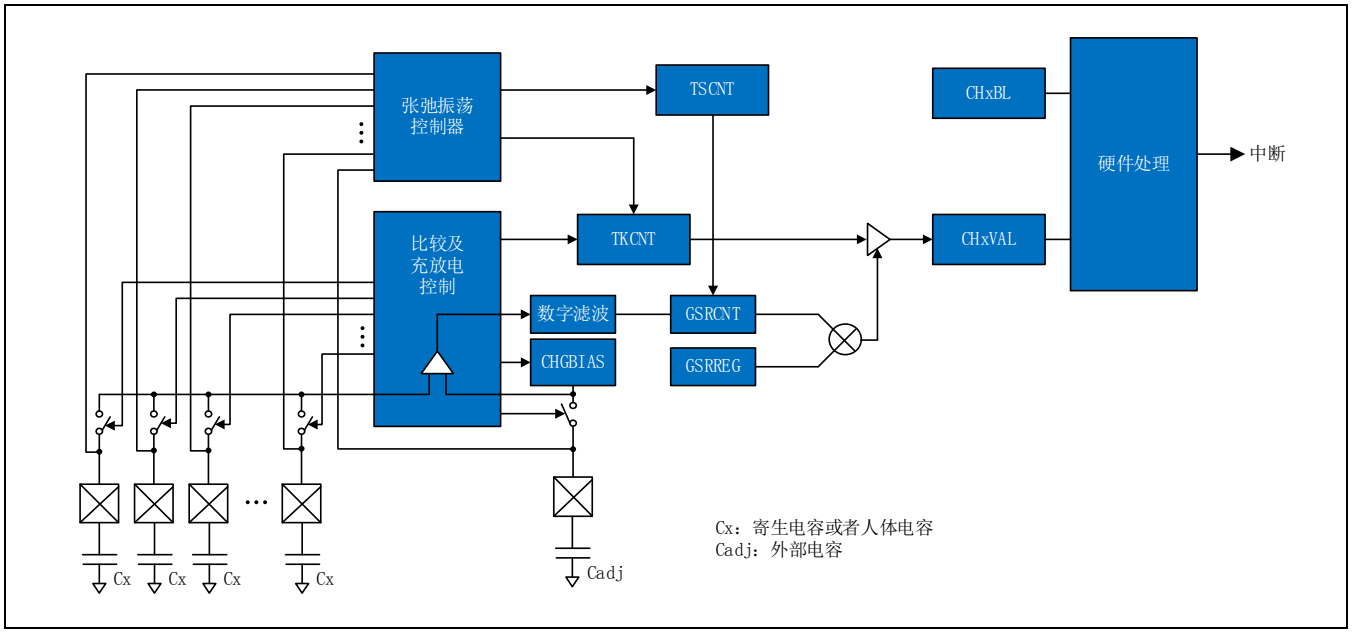


Figure 14-1 Touch Sensor模块框图

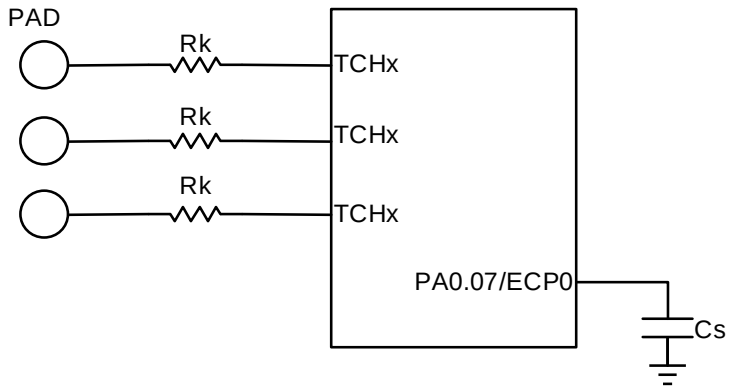
14.2.2 工作原理

电容式按键传感器是一种基于自电容检测技术，在人体或带电物体靠近传感极点时，导致自电容的变化，根据这种变化从而实现按键或者触摸滑条等应用的实现。

本芯片同时可实现两种不同的自电容检测技术，一种是基于张弛振荡器的自电容检测，一种是基于电荷转移的自电容检测。基于张弛振荡器检测技术的电容感应，可以实现在较低功耗基础上的电容检测，而且由于支持 8 通道同时扫描，可以有效减少扫描的时间，缩短响应时间。基于电荷转移的电容检测在抗干扰上有很强的优势，但是需要外接调整电容，而且只支持单路分时扫描模式。根据不同的应用场合，可以选择不同的检测方式进行工作。

在张弛振荡器模式下，所有的检测通道都连接到芯片内部的一个张弛振荡器上，外部的寄生电容和内部电路构成充放电振荡器，此振荡器的频率会随着寄生电容的变化而变化。张弛振荡器的输出连接到一个 TSCNT 计数器上，当人体（手指）靠近传感极点时，人体电容会导致寄生电容的改变，从而改变张弛振荡器输出的频率周期。通过一个内部采样计数器（CHxCNT，此计数器时钟来自系统内部参考时钟）可以记录 TSCNT 计数到某个特定值所需的时间。寄生电容变大时，CHxCNT 值也会变大；寄生电容变小时，CHxCNT 值也会变小。

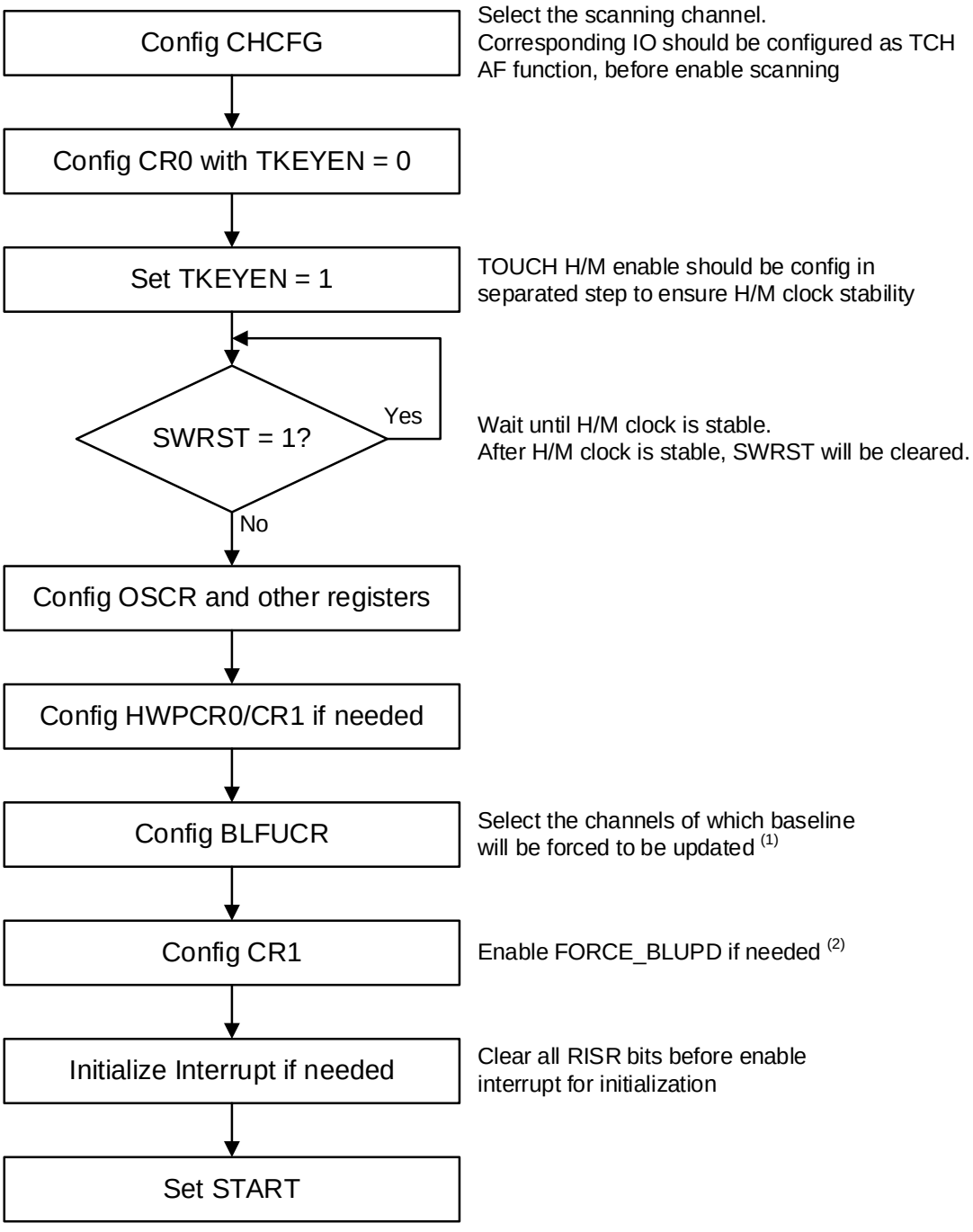
在电荷转移模式下，所有的通道都通过内部开关连接到一个比较器上。开始扫描时，内部充电电路对外部参考电容（Cadj）进行充电直到 Vth，然后被扫描通道的开关被打开，外部参考电容上的电荷将和扫描通道上的寄生电容(Cx)进行电荷平衡，平衡后，扫描通道电容(Cx)上的电荷将被放电，然后再继续平衡，如此往复直到外部参考电容上的电荷被放电到 Vtl。充放电的次数将被一个内部采样计数器记录（CHxCNT）。寄生电容变大时，CHxCNT 值会变小；寄生电容变小时，CHxCNT 值会随之变大。



- Rk is used for improving noise immunity under touch application. Suggested value of Rk is 1KΩ to 20KΩ. The bigger value of Rk is selected, the better immunity performance will be achieved, but the sensitivity will be poor as well.
- Cs is external cap which is used for charge reference, the value of Cs is suggested to be 0.01uF to 0.1uF.

Figure 14-2 电荷转移模式应用参考连接图

扫描模式的选择通过 TCH_CR0 寄存器中 MODE 和 SCMODE 来进行配置。



(1) To initialize the channels for the 1st scanning. Baseline will be forced to be initialized as current sampled data at start. Baseline register also can be directly changed to an expected value when scanning is not enabled by writing to CHxBL register.

(2) The baseline of specific channel will not be updated, if current channel is issuing a key detect event

Figure 14-3 软件配置流程

14.2.3 硬件数字处理引擎

APT32F101 内嵌硬件数字处理引擎，可以通过硬件比较内部计数器（CHxCNT）和基准的差值来确认按键是否被按下。当检测到按键时，可以通过配置寄存器产生相应CPU的中断。由于集成硬件数字处理，按键扫描可以自动完成，不需要额外的CPU运算进行干预，即使在CPU停止工作时，按键也可以继续自动扫描。当检测到有按键时，可以通过按键中断唤醒CPU。

硬件处理引擎可以对采集数据直接进行滤波和运算，自动控制基准值更新。每个通道都有一组独立的采样计数器（CHxCNT）和基准值寄存器（BLxVAL），当采样计数器和基准值寄存器之间的差值大于阈值寄存器（TCH_OSCR）所设置的值时，外部按键中断将被触发。

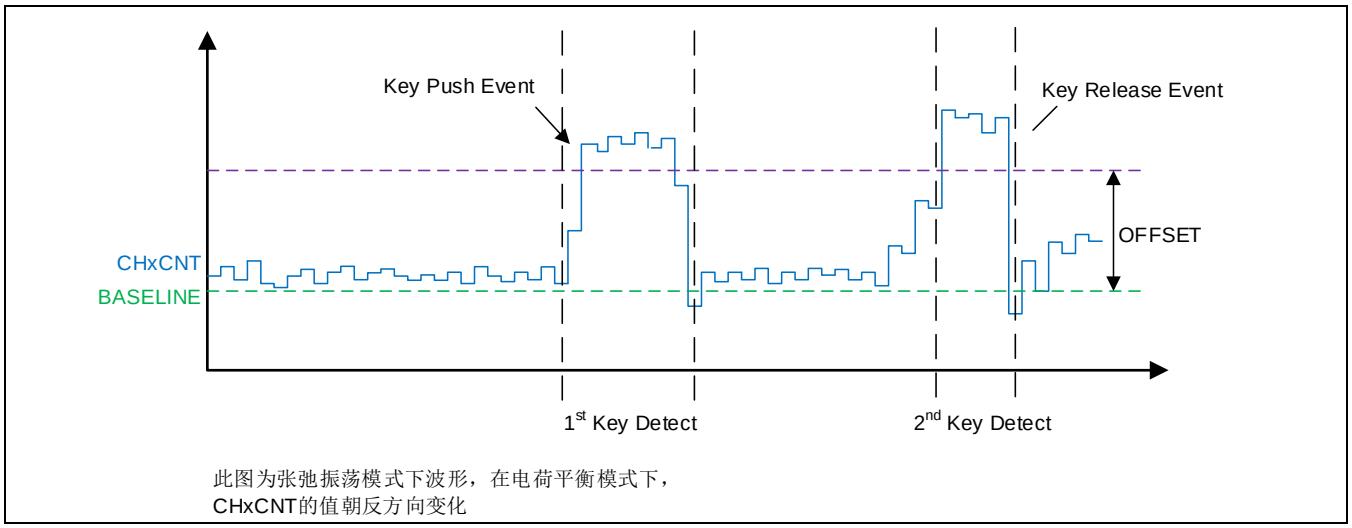


Figure 14-4 按键触发示意图

硬件处理引擎的配置通过TCH_CR1，TCH_BLFUCR，TCH_HWPCR0，TCH_HWPCR1，TCH_TSR，TCH_GSR，TCH_OSRx，TCH_TKCR 这几个寄存器来实现。

- 通过设置TCH_CR1的FORCE_BLUPD位可以强制更新硬件比较的基准值，更新基准值的通道和更新方式通过TCH_BLFUCR设置。
- 在扫描停止时，可以直接更新每个通道的基准值CHxBL，但在扫描进行时，只能通过FORCE_BLUPD来更新。
- 基准值的自动更新可以通过TCH_HWPCR0和TCH_HWPCR1来设置。当设置TCH_CR1的BLUPDIS位以后，基准值自动更新将被关闭。
- 通过设置TCH_TSR和TCH_GSR可以改变每个通道的灵敏度。一共支持4挡灵敏度设置，每个通道可以选择4挡灵敏度中的任意一个作为本通道的灵敏度设置。
- 按键的比较阈值（OFFSET）通过TCH_OSRx设置。每个通道可以根据实际需要设置不同的比较阈值。当按键采样值大于比较阈值时，在按键去抖条件满足后，将触发按键中断
- 按键中断触发条件可以设置为按下事件，松开事件，或者按下和松开事件。通过设置TCH_TKCR选择。
- 当前所有通道的按键状态，可以通过TCH_TKEYST寄存器查询。
- 按键模式分为单按键模式和多按键模式，当选择单按键模式时，当已有按键被按下时，其他按键将被忽略。模式选择通过TCH_CR1中MKEYMOD位来设置。

在张弛振荡模式下，每个感应通道的振荡计数器TSCNT值可以通过TCH_TSCCR和TCH_TSCDR来设置。在遇

到干扰时，可以通过调节TCH_CR1寄存器中的TOSCFREQ位，选择外部振荡器的中心频率。所有被配置为扫描功能，但当前不处于扫描状态的IO，其缺省状态可以通过TCH_CHDST来设置。

14.2.4 扫描模式和扫描触发方式设置

14.2.4.1 扫描工作时钟

整个通道扫描控制模块的工作时钟根据检测模式的不同分为两种。在张弛振荡模式下，工作时钟为相对应的张弛振荡器模块提供的参考时钟；在电荷转移模式下，工作时钟为ISOSC。工作时钟的频率可以通过TCH_CR0的PDIV来设置分频参数。

14.2.4.2 扫描模式

感应通道的扫描分为两种扫描模式：顺序扫描和并行扫描。顺序扫描时，按照从低到高的通道号依次扫描。并行扫描时，0~7通道同时扫描，扫描完成后，8~15通道再同时扫描。在电荷转移模式下，只支持顺序扫描方式进行工作。在扫描启动前，先应使能触控传感器模块（TCH_CR0中的TKEYEN），通过查询TCH_CR0中的SWRST位可以确认扫描控制时钟是否已经准备就绪。只有在扫描控制时钟就绪以后，才可以启动扫描。

扫描的启动通过设置TCH_START寄存器来启动。在使能TCH_START以后，可以通过查询TCH_START状态获知当前的扫描状态。所有硬件处理引擎的控制寄存器在扫描进行时都不能进行改写操作，除了TCH_CR0的ICNT位，TCH_CR1的FORCE_BLUPD位，TCH_CR1的BLUPDIS位和TCH_FUCR寄存器。

需要扫描的通道通过TCH_CHCFG寄存器设置。所有的使能通道扫描完成后，扫描将自动停止，一轮扫描结束。扫描结束后，再次设置TCH_START可以开始下一轮的扫描。

14.2.4.3 扫描触发方式

除软件启动扫描方式，芯片还支持自动扫描方式工作。自动扫描模式下，START控制位将会在上一轮扫描结束以后并满足触发条件时，自动启动下一轮扫描。扫描的触发条件可以通过TCH_CR0的STARTCFG位设置。

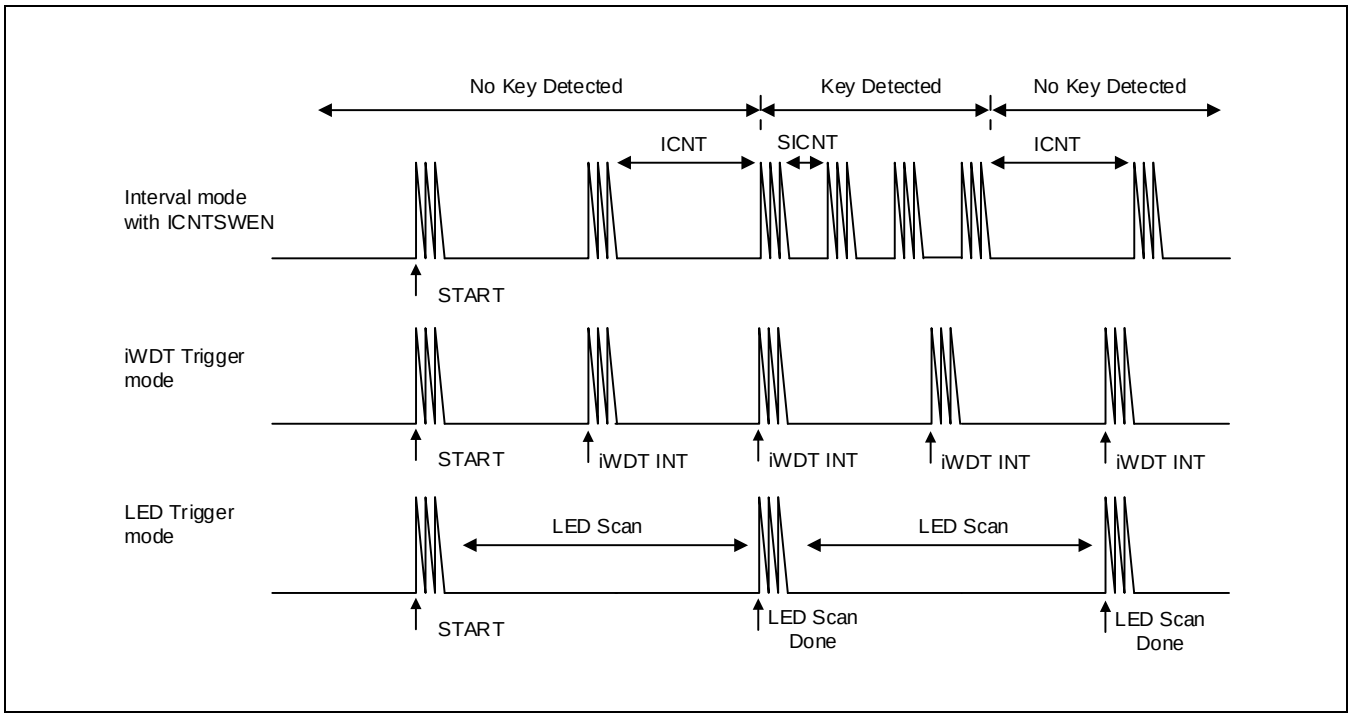


Figure 14-5 自动扫描模式

自动扫描模式启动以后，可以通过清除START位来停止自动扫描。当START位被清除以后，内部扫描不会立即停止，而是要等到本轮扫描结束后才会停止。可以通过查询START位状态来判断当前的扫描状态。

当设置为间隔触发模式时，一轮扫描结束后，系统将自动启动一个12位计数器，此计数器计数频率为扫描工作频率的128分频。当计数器值等于TCH_CR0的ICNT时，扫描将自动启动。间隔扫描还支持自动加速功能，在TCH_CR1的ICNTSWEN位使能后，一旦检测到按键（无论去抖是否满足），扫描间隔计数器比较值将由ICNT自动切换到SICNT；在按键没有被检测到时，自动切换回ICNT。

当设置为iWDT中断触发模式时，一轮扫描结束后，系统将等待iWDT的警告中断。iWDT警告中断发生后，扫描将自动启动

当设置为LED触发模式时，一轮扫描结束后，系统将等待LED扫描结束信号。LED扫描结束后，触控按键扫描将自动启动。此模式配合LED同时工作，需要将LED控制寄存器中的SHARE模式使能。具体可参考LED章节。

当设置为LED触发模式时，TOUCH的扫描时间长度将会影响到LED的显示亮度。为避免LED闪烁，TOUCH的扫描频率应尽可能设置高，推荐PDIV设置为不分频，并且ISOSC的频率设置为3MHz。

14.2.5 中断产生

14.2.5.1 各个通道扫描完成中断

每个通道都有独立的扫描完成中断。通过设置CHx_DONE 位，可以设置该通道扫描完成后触发相应的通道扫描完成中断。

14.2.5.2 按键中断

在硬件处理引擎被开启以后，一旦有按键检测到以后，该中断会被触发。触发的条件可以通过TCH_TKCR来设置。同时需要考虑设置按键去抖（TCH_CR1的KEYDETDEB）增加抗干扰能力。

14.2.5.3 异常复位中断

在硬件处理引擎被开启以后，为防止某个按键由于错误的更新了参考值（一般为干扰引起），而导致的某个按键长时间处于按下状态，可以通过硬件自动监测功能来复位处理引擎。该监测功能通过设置TCH_CR1中TKEYSTICKCHK位来启动。该监测功能使能以后，一旦有按键按下，自动监测模块即开始计时，当计时超过KEYSTICKDUR的设定值时，该按键仍旧没有释放，将会强制复位处理引擎，并触发异常复位中断。

该异常复位效果等同于软件复位中的硬件处理引擎复位，但是软件复位不会产生该异常中断。该异常中断将自动清除所有的硬件算法控制电路的状态，包括采样值和基准值寄存器。但是配置寄存器的设置将会被保持，除了CR1中的FORCE_BLUPD控制位和START寄存器（扫描需要重新通过软件启动）。

14.2.5.4 扫描超时中断

在张弛振荡扫描模式下，由于电路异常或者强干扰存在，导致感应的张弛振荡器工作异常。长时间没有振荡或者振荡频率完全不能被检测到。这时通过使能扫描时间监测功能（TCH_SCTCR），可以在当前扫描时间超出规定扫描时间后，触发该异常中断。此中断不会强制复位处理引擎，需软件处理。

14.2.5.5 基准值更新中断

在有基准值被更新后，该中断立即被触发。程序可以通过查询TCH_BLUPINF寄存器获得被更新的通道号。TCH_BLUPINF不会自动清除，在查询后，需要软件清除该寄存器。

14.2.5.6 全部通道扫描完成中断

所有扫描使能的通道在扫描处理完成后，触发该中断。该中断表示本轮扫描结束。

.

14.3 寄存器说明

14.3.1 寄存器表

- Base Address: 0x4002_0000

Register	Offset	Description	Reset Value
TCH_CR0	0x00	Touch Sensor General Control Register0	0xFFF0_0000
TCH_CR1	0x04	Touch Sensor General Control Register1	0x0850_0100
TCH_HWPCR0	0x08	Hardware Processing Control Register0	0xFF10_7318
TCH_HWPCR1	0x0C	Hardware Processing Control Register1	0xFF10_003C
TCH_BLFUCR	0x10	Baseline Force Updating Control Register	0x0000_0000
TCH_START	0x14	Touch Sensor Start Control Register	0x0000_0000
TCH_SCTCR	0x18	Touch Scanning Time Limitation Control Register	0x0000_FFFF
TCH_CHDST	0x1C	Channel Disable Status Control Register	0x0000_0000
TCH_TSCCR	0x20	Touch Sensing Counter Configuration Register	0x0000_0000
TCH_TSCDR	0x24	Touch Sensing Counter Compare Data Register	0x0000_0000
TCH_CHCFG	0x28	Touch Sensor Channel Configuration Register	0x0000_0000
TCH_TSR	0x2C	Touch Sensor Sensitivity Select Register	0x0000_0000
TCH_GSR	0x30	Touch Sensor Global Sensitivity Register	0x0000_0000
TCH_OSR0	0x34	Offset Register for Channel 0 to 3	0x0000_0000
TCH_OSR1	0x38	Offset Register for Channel 4 to 7	0x0000_0000
TCH_OSR2	0x3C	Offset Register for Channel 8 to 11	0x0000_0000
TCH_OSR3	0x40	Offset Register for Channel 12 to 15	0x0000_0000
TCH_TKCR	0x44	Touch Key Interrupt Trigger Condition Register	0x0000_0000
TCH_RISR	0x48	Touch Senor Raw Interrupt Status Register	0x0000_0000
TCH_IMCR	0x4C	Touch Senor Interrupt Masking Control Register	0x0000_0000
TCH_MISR	0x50	Touch Senor Masked Interrupt Status Register	0x0000_0000
TCH_ICR	0x54	Touch Senor Interrupt Clear Control Register	0x0000_0000
TCH_CHxCNT[16]	0x58 – 0x94	Touch Sensor Channel x Sampling Counter Value	0x0000_0000
TCH_CHxBL[16]	0x98 – 0xD4	Touch Sensor Channel x Baseline Value	0x0000_0000
TCH_TKEYST	0xD8	Touch Key Status Register	0x0000_0000
TCH_BLUPINF	0xDC	Baseline Updating Information Register	0x0000_0000

NOTE: 由于本产品中通道 0, 3, 4 和 10 不存在，所以寄存器中这些通道对应的设置值无效，请勿操作。

14.3.2 TCH_CR0 (通用控制寄存器0)

- Address = Base Address + 0x0000, Reset Value = 0xFFF0_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICNT												STPENA	HWPROC	STARTCFG		HYST	PDIV			SICNT		RSVD	DBGEN	CLKEN	SWRST				MODE	SCMODE	TKEYEN
1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
TKEYEN	[0]	RW	触摸按键Hard-macro使能 0： 关闭触控Hard-macro 模块 1： 开启触控Hard-macro 模块	0x0
SCMODE	[1]	RW	扫描模式设置： 0： 所有通道顺序扫描 1： 所有通道分组并行扫描	0x0
MODE	[2]	RW	扫描原理设置： 0： 张弛振荡器模式 1： 电荷转移模式	0x0
SWRST	[6:3]	RW	触控模块软件复位： 0001b： 硬件处理引擎复位 0101b： 全模块复位（所有寄存器恢复初始值） Bit3 可以读取到当前复位状态，当读取到‘1’时，表示当前复位没有释放，或者Hard-macro没有使能。	0x0
CLKEN	[7]	RW	工作时钟使能控制： 0： 工作时钟停止 1： 工作时钟开启 在扫描过程中，停止工作时钟后，可以对锁定的控制寄存器进行修改。	0x0
DBGEN	[8]	RW	调试使能控制： 0： 程序调试状态中，程序挂起后，扫描继续 1： 程序调试状态中，程序挂起后，扫描暂停	0x0
SICNT	[11:10]	RW	快速间隔扫描计数器	0x0

<table>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>PDIV</td><td>[14:12]</td><td>RW</td><td>工作时钟分频控制：<table><tr><th>PDIV</th><th>FREQ</th></tr><tr><td>0</td><td>None-div</td></tr><tr><td>1</td><td>2 div</td></tr><tr><td>2</td><td>4 div</td></tr><tr><td>3</td><td>8 div</td></tr><tr><td>4</td><td>16 div</td></tr><tr><td>5</td><td>32 div</td></tr></table></td><td>0x0</td></tr>
<tr><td>HYST</td><td>[15]</td><td>RW</td><td>按键检测迟滞设置：
0： 3/4 OFFSET 作为按键释放检测点
1： 1/2 OFFSET 作为按键释放检测点</td><td>0x0</td></tr>
<tr><td>STARTCFG</td><td>[17:16]</td><td>RW</td><td>扫描触发设置：
00b： 软件启动扫描
01b： 间隔扫描触发
10b： 硬件触发（iWDT中断）
11b： 硬件触发（LED）</td><td>0x0</td></tr>
<tr><td>HWPROC</td><td>[18]</td><td>RW</td><td>硬件处理引擎使能控制：
0： 禁止硬件处理
1： 启动硬件处理</td><td>0x0</td></tr>
<tr><td>STPENA</td><td>[19]</td><td>RW</td><td>在CPU停止时，扫描模块工作方式：
0： 禁止扫描模块工作
1： 保持扫描模块工作</td><td>0x0</td></tr>
<tr><td>ICNT</td><td>[31:20]</td><td>RW</td><td>间隔扫描时间控制计数器

该计数器以工作时钟的128分频工作。
当设置为’0’时，间隔时间为两个工作时钟。</td><td>0xFFF</td></tr>
</table>

NOTE: 如果需要在程序中多次修改扫描工作模式，请务必在修改模式前执行一次硬件处理引擎复位。

14.3.3 TCH_CR1 (通用控制寄存器1)

- Address = Base Address + 0x0004, Reset Value = 0x0850_0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLUPDIS_CYCLE								BLUPDIS	ICNTSWEN	KEYDETDEB		KEYSTICKDUR			KEYSTICKCHK	TOSCFDIV				BLUPDINCT				CMPFLT		TOSCFREQ		MKEYMODE	OFFSET_MUL		FORCE_BLUPD
0	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
FORCE_BLUPD	[0]	RW	写操作时，强制更新参考值： 0： 无效果 1： 启动更新参考值 读操作时，查询当前更新参考值状态： 0： 参考值更新已结束 1： 参考值更新正在进行	0x0
OFFSET_MUL	[2:1]	RW	OFFSET值自动放大处理。在某些应用上，由于干扰或系统稳定度比较低，造成采样值不稳定。这时需要比较大的OFFSET设置值。OFFSET寄存器最大设置为0xFF，当需要设置更大的OFFSET，可以通过此寄存器对OFFSET设置进行乘积放大。 0： 不进行放大 1： 放大到2倍 2： 放大到4倍 3： 放大到8倍	0x0
MKEYMODE	[3]	RW	按键自动检测模式设置 0： 单按键模式，最多只有一个按键可以被激活 1： 多按键模式，支持多个按键同时激活	0x0
TOSCFREQ	[5:4]	RW	张弛振荡模式下，通道感应振荡器频率设置 00b： 默认频率 01b： 默认频率的2倍 1xb： 默认频率的25倍	0x0
CMPFLT	[7:6]	RW	电荷转移模式下，模拟比较器输出的数字滤波设置	0x0

			0： 无滤波 1： x1 工作频率周期的滤波 2： x2 工作频率周期的滤波 3： x3 工作频率周期的滤波	
BLUPDICNT	[11:8]	RW	自动参考值更新检测周期设置 0： 1轮扫描为一次检测周期 1： 2轮扫描为一次检测周期 2： 3轮扫描为一次检测周期 3： 4轮扫描为一次检测周期	0x1
TOSCFDIV	[15:12]	RW	张弛振荡模式下，通道感应振荡器的预分频设置 分频系数为 (TOSCFDIV+1)。	0x0
KEYSTICKCHK	[16]	RW	按键自动检测时，按键状态锁死监测设置 0： 禁止按键锁死监测 1： 开启按键锁死监测	0x0
KEYSTICKDUR	[19:17]	RW	按键锁死监测开启后，异常判读阈值设置。在按键锁死时间超过该阈值时，将自动触发数字处理引擎的复位。 时间阈值 = (KEYSTICKDUR+1) x 16秒	0x0
KEYDETDEB	[21:20]	RW	按键自动检测时，去抖设置 0： 无去抖 1： 1次去抖 2： 2次去抖 3： 3次去抖	0x1
ICNTSWEN	[22]	RW	自动间隔扫描时，扫描间隔周期自动切换设置。当该位被设置以后，在有按键时，自动间隔扫描时间由SICNT决定。在没有检测到按键时，自动间隔扫描时间由ICNT决定。 0： 禁止切换扫描间隔，扫描间隔由ICNT决定 1： 自动切换扫描间隔	0x1
BLUPDIS	[23]	RW	基准值自动更新停止寄存器 0： 总是启动基准值自动更新 1： 总是禁止基准值自动更新	0x0
BLUPDIS_CYCLE	[31:24]	RW	按键释放后，基准值自动更新禁止周期设置。当有按键释放被检测到时，可以设置接下来的几轮扫描为禁止基准值更新周期，以保证基准值的稳定。	0x8

14.3.4 TCH_HWPCR0 (硬件处理控制寄存器0)

- Address = Base Address + 0x0008, Reset Value = 0xFF10_7318

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLDUP_THVAL								RSVD		ABNFLT		BLUPD_DEB1								BLUPD_DEB0				RSVD		BLUPD_WCO1		BLUPD_WCO0		BLUP_THR	
1	1	1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	1	1	1	0	0	1	1	0	0	0	1	1	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
BLUPD_THR	[1:0]	RW	参考值自动更新阈值设置。 00b： 设置更新阈值为1/4 OFFSET 01b： 设置更新阈值为1/2 OFFSET 10b： 设置更新阈值为3/4 OFFSET 11b： 设置更新阈值为BLUPD_THVAL的值	0x0
BLUPD_WCO0	[3:2]	RW	当采样值小于BLUPD_THR所设置的阈值时，参考值更新的权重设置。 0： 参考值不进行更新 1： 以1/4 权重进行更新 2： 以1/2 权重进行更新 3： 用采样值直接进行更新	0x2
BLUPD_WCO1	[5:4]	RW	当采样值大于BLUPD_THR所设置的阈值时，参考值更新的权重设置。 0： 参考值不进行更新 1： 以1/4 权重进行更新 2： 以1/2 权重进行更新 3： 用采样值直接进行更新	0x1
BLUPD_DEB0	[11:8]	RW	当采样值连续多次小于BLUPD_THR所设置的阈值时，处理引擎将根据WCO0设置的条件对参考值进行自动更新。通过该寄存器可以设置连续检测的次数。	0x3
BLUPD_DEB1	[19:12]	RW	当采样值连续多次大于BLUPD_THR所设置的阈值时，处理引擎将根据WCO1设置的条件对参考值进行自动更新。通过该寄存器可以设置连续检测的次数。	0x7
ABNFLT	[21:20]	RW	异常采样值滤波设置，当采样值大于设置阈值，将被处理引擎忽略	0x1

			0： 大于4倍OFFSET 1： 大于8倍OFFSET 2： 大于16倍OFFSET 3： 禁止滤波功能	
BLUPD_THVAL	[31:24]	RW	自动更新阈值设定值	0xFF

NOTE: 该寄存器在采样值不小于基准值时起作用。当采样值小于基准值时，处理引擎将使用 HWPCR1 中的设置对基准值进行处理。

14.3.5 TCH_HWPCR1 (硬件处理控制寄存器1)

- Address = Base Address + 0x000C, Reset Value = 0xFF10_003C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLDUP_THVAL								RSVD		ABNFLT		BLUPD_DEB1				BLUPD_DEB0				RSVD		BLUPD_WCO1		BLUPD_WCO0		BLUP_THR			
1	1	1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
BLUPD_THR	[1:0]	RW	参考值自动更新阈值设置。 00b： 设置更新阈值为1/4 OFFSET 01b： 设置更新阈值为1/2 OFFSET 10b： 设置更新阈值为3/4 OFFSET 11b： 设置更新阈值为BLUPD_THVAL的值	0x0
BLUPD_WCO0	[3:2]	RW	当采样值小于BLUPD_THR所设置的阈值时，参考值更新的权重设置。 0： 参考值不进行更新 1： 以1/4 权重进行更新 2： 以1/2 权重进行更新 3： 用采样值直接进行更新	0x3
BLUPD_WCO1	[5:4]	RW	当采样值大于BLUPD_THR所设置的阈值时，参考值更新的权重设置。 0： 参考值不进行更新 1： 以1/4 权重进行更新 2： 以1/2 权重进行更新 3： 用采样值直接进行更新	0x3
BLUPD_DEB0	[11:8]	RW	当采样值连续多次小于BLUPD_THR所设置的阈值时，处理引擎将根据WCO0设置的条件对参考值进行自动更新。通过该寄存器可以设置连续检测的次数。	0x0
BLUPD_DEB1	[19:12]	RW	当采样值连续多次大于BLUPD_THR所设置的阈值时，处理引擎将根据WCO1设置的条件对参考值进行自动更新。通过该寄存器可以设置连续检测的次数。	0x0
ABNFLT	[21:20]	RW	异常采样值滤波设置，当采样值大于设置阈值，将被处理引擎忽略	0x1

			0： 大于4倍OFFSET 1： 大于8倍OFFSET 2： 大于16倍OFFSET 3： 禁止滤波功能	
BLUPD_THVAL	[31:24]	RW	自动更新阈值设定值	0xFF

14.3.6 TCH_BLFUCR (基准强制更新控制寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLUPD_VAL																CH15_BLUPD	CH14_BLUPD	CH13_BLUPD	CH12_BLUPD	CH11_BLUPD	CH10_BLUPD	CH9_BLUPD	CH8_BLUPD	CH7_BLUPD	CH6_BLUPD	CH5_BLUPD	CH4_BLUPD	CH3_BLUPD	CH2_BLUPD	CH1_BLUPD	CH0_BLUPD
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
CHx_BLUPD	[15:0]	RW	指定需要强制更新参考值的通道号 0： 不启动强制更新 1： 启动强制更新	0x0
BLUPD_VAL	[31:16]	RW	强制更新值设定，当BLUPD_VAL设置为全零时，当前通道的采样值将会被强制更新到参考值寄存器中。	0x0

14.3.7 TCH_START (扫描启动控制寄存器)

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

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

Name	Bit	RW	Description	Reset Value
START	[0]	RW	扫描启动控制位 写操作时： 0： 停止扫描 1： 启动扫描 读操作时： 0： 扫描没有开始 1： 扫描正在进行	0x0

14.3.8 TCH_SCTCR (扫描时间限制控制寄存器)

- Address = Base Address + 0x0018, Reset Value = 0x0000_FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																SCTDATA															
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	RW	Description	Reset Value
SCTDATA	[15:0]	RW	张弛振荡模式下，扫描超时设置。当扫描开始后，内部的监测寄存器开始计数。当计数器超过SCTDATA设置时，将会触发SCTOVF中断。 内部的监测计数器为24位，该寄存器设置为该计数器的高16位值。当该寄存器设置为'0'时，此功能被禁止。	0xFFFF

14.3.9 TCH_CHDST (扫描禁止通道状态控制寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSR15		DSR14		DSR13		DSR12		DSR11		DSR10		DSR9		DSR8		DSR7		DSR6		DSR5		DSR4		DSR3		DSR2		DSR1		DSR0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
DSRx	[1:0] [3:2] …… [31:30	RW	设置在扫描过程中，没有被激活为当前扫描通道的通道状态 00b： 高阻状态 01b： 低电平输出 10b： 高电平输出 11b： 高阻状态	0x0

14.3.10 TCH_TSCCR (张弛振荡感应计数器配置寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSC15		TSC14		TSC13		TSC12		TSC11		TSC10		TSC9		TSC8		TSC7		TSC6		TSC5		TSC4		TSC3		TSC2		TSC1		TSC0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
TSCx	[1:0] [3:2] …… [31:30	RW	在张弛振荡模式下，选择每个通道的张弛振荡器感应计数器值。 00b： 选择TSCDR中的CDR0为当前通道计数器比较值 01b： 选择TSCDR中的CDR1为当前通道计数器比较值 10b： 选择TSCDR中的CDR2为当前通道计数器比较值 11b： 选择TSCDR中的CDR3为当前通道计数器比较值	0x0

14.3.11 TCH_TSCDR (张弛振荡感应计数器值寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CDR3								CDR2								CDR1								CDR0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
CDR0	[7:0]	RW	感应通道计数器比较值0	0x0
CDR1	[15:8]	RW	感应通道计数器比较值1	0x0
CDR2	[23:16]	RW	感应通道计数器比较值2	0x0
CDR3	[31:24]	RW	感应通道计数器比较值3	0x0

14.3.12 TCH_CHCFG (扫描通道配置寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																CH15EN	CH14EN	CH13EN	CH12EN	CH11EN	CH10EN	CH9EN	CH8EN	CH7EN	CH6EN	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	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
CHxEN	[15..0]	RW	扫描通道使能控制 0： 禁止该通道扫描 1： 使能该通道扫描	0x0

NOTE: 即使在 TCH_CHCFG 中设置了通道使能，如果该通道所对应的 GPIO 功能没有被设置为 TKEY 功能，该通道扫描会被禁止。

14.3.13 TCH_TSR (通道灵敏度选择寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSR15		TSR14		TSR13		TSR12		TSR11		TSR10		TSR9		TSR8		TSR7		TSR6		TSR5		TSR4		TSR3		TSR2		TSR1		TSR0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
TSRx	[1:0] [3:2] …… [31:30	RW	选择每个通道所对应的灵敏度设置 00b： 选择GSR中的GSR0为当前通道灵敏度值 01b： 选择GSR中的GSR1为当前通道灵敏度值 10b： 选择GSR中的GSR2为当前通道灵敏度值 11b： 选择GSR中的GSR3为当前通道灵敏度值	0x0

14.3.14 TCH_GSR (全局灵敏度值寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GSR3								GSR2								GSR1								GSR0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
GSR0	[7:0]	RW	通道灵敏度值0	0x0
GSR1	[15:8]	RW	通道灵敏度值1	0x0
GSR2	[23:16]	RW	通道灵敏度值2	0x0
GSR3	[31:24]	RW	通道灵敏度值3	0x0

14.3.15 TCH_OSR0 (偏移量控制寄存器0)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OS_CH3								OS_CH2								OS_CH1								OS_CH0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
OS_CH0	[7:0]	RW	通道0的偏移（OFFSET）值	0x0
OS_CH1	[15:8]	RW	通道1的偏移（OFFSET）值	0x0
OS_CH2	[23:16]	RW	通道2的偏移（OFFSET）值	0x0
OS_CH3	[31:24]	RW	通道3的偏移（OFFSET）值	0x0

14.3.16 TCH_OSR1 (偏移量控制寄存器1)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OS_CH7								OS_CH6								OS_CH5								OS_CH4							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
OS_CH4	[7:0]	RW	通道4的偏移（OFFSET）值	0x0
OS_CH5	[15:8]	RW	通道5的偏移（OFFSET）值	0x0
OS_CH6	[23:16]	RW	通道6的偏移（OFFSET）值	0x0
OS_CH7	[31:24]	RW	通道7的偏移（OFFSET）值	0x0

14.3.17 TCH_OSR2 (偏移量控制寄存器2)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OS_CH11								OS_CH10								OS_CH9								OS_CH8							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
OS_CH8	[7:0]	RW	通道8的偏移（OFFSET）值	0x0
OS_CH9	[15:8]	RW	通道9的偏移（OFFSET）值	0x0
OS_CH10	[23:16]	RW	通道10的偏移（OFFSET）值	0x0
OS_CH11	[31:24]	RW	通道11的偏移（OFFSET）值	0x0

14.3.18 TCH_OSR3 (偏移量控制寄存器3)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OS_CH15								OS_CH14								OS_CH13								OS_CH12							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
OS_CH12	[7:0]	RW	通道12的偏移（OFFSET）值	0x0
OS_CH13	[15:8]	RW	通道13的偏移（OFFSET）值	0x0
OS_CH14	[23:16]	RW	通道14的偏移（OFFSET）值	0x0
OS_CH15	[31:24]	RW	通道15的偏移（OFFSET）值	0x0

14.3.19 TCH_TKCR (按键中断触发条件控制寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TKCR15		TKCR14		TKCR13		TKCR12		TKCR11		TKCR10		TKCR9		TKCR8		TKCR7		TKCR6		TKCR5		TKCR4		TKCR3		TKCR2		TKCR1		TKCR0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
TKCRx	[1:0] [3:2] …… [31:30	RW	选择每个通道按键中断触发方式 00b： 选择按键按下事件为中断触发方式 01b： 选择按键按下事件为中断触发方式 10b： 选择按键松开事件为中断触发方式值 11b： 选择按键按下/松开事件为中断触发方式	0x0

14.3.20 TCH_RISR (原始中断状态寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRCDNE	BLUPD	SCTOVF	HWRST	KEYINT	RSVD											CH15_DNE	CH14_DNE	CH13_DNE	CH12_DNE	CH11_DNE	CH10_DNE	CH9_DNE	CH8_DNE	CH7_DNE	CH6_DNE	CH5_DNE	CH4_DNE	CH3_DNE	CH2_DNE	CH1_DNE	CH0_DNE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	RW	Description	Reset Value
CHx_DNE	[15..0]	R	每个通道扫描完成中断	0x0
KEYINT	[27]	R	触摸按键中断	0x0
HWRST	[28]	R	异常复位中断	0x0
SCTOVF	[29]	R	扫描时间超时中断	0x0
BLUPD	[30]	R	基准值更新中断	0x0
PRCDNE	[31]	R	全部通道扫描完成中断	0x0

该寄存器用于查询原始中断标志状态。一旦事件被触发，无论中断是否使能，该寄存器都会记录状态。

0： 原始中断未发生
1： 原始中断发生

14.3.21 TCH_IMCR (中断使能控制寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRCDNE	BLUPD	SCTOVF	HWRST	KEYINT	RSVD											CH15_DNE	CH14_DNE	CH13_DNE	CH12_DNE	CH11_DNE	CH10_DNE	CH9_DNE	CH8_DNE	CH7_DNE	CH6_DNE	CH5_DNE	CH4_DNE	CH3_DNE	CH2_DNE	CH1_DNE	CH0_DNE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
CHx_DNE	[15..0]	RW	每个通道扫描完成中断	0x0
KEYINT	[27]	RW	触摸按键中断	0x0
HWRST	[28]	RW	异常复位中断	0x0
SCTOVF	[29]	RW	扫描时间超时中断	0x0
BLUPD	[30]	RW	基准值更新中断	0x0
PRCDNE	[31]	RW	全部通道扫描完成中断	0x0

该寄存器用于使能中断。

0： 关闭中断
1： 打开中断

14.3.22 TCH_MISR (中断状态控制寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRCDNE	BLUPD	SCTOVF	HWRST	KEYINT	RSVD											CH15_DNE	CH14_DNE	CH13_DNE	CH12_DNE	CH11_DNE	CH10_DNE	CH9_DNE	CH8_DNE	CH7_DNE	CH6_DNE	CH5_DNE	CH4_DNE	CH3_DNE	CH2_DNE	CH1_DNE	CH0_DNE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	RW	Description	Reset Value
CHx_DNE	[15..0]	R	每个通道扫描完成中断	0x0
KEYINT	[27]	R	触摸按键中断	0x0
HWRST	[28]	R	异常复位中断	0x0
SCTOVF	[29]	R	扫描时间超时中断	0x0
BLUPD	[30]	R	基准值更新中断	0x0
PRCDNE	[31]	R	全部通道扫描完成中断	0x0

该寄存器用于查询触发中断的中断源。

0： 中断未触发
1： 中断触发

14.3.23 TCH_ICR (中断清除控制寄存器)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRCDNE	BLUPD	SCTOVF	HWRST	KEYINT	RSVD											CH15_DNE	CH14_DNE	CH13_DNE	CH12_DNE	CH11_DNE	CH10_DNE	CH9_DNE	CH8_DNE	CH7_DNE	CH6_DNE	CH5_DNE	CH4_DNE	CH3_DNE	CH2_DNE	CH1_DNE	CH0_DNE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	W	W	W	W	R	R	R	R	R	R	R	R	R	R	R	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Name	Bit	RW	Description	Reset Value
CHx_DNE	[15..0]	W	每个通道扫描完成中断	0x0
KEYINT	[27]	W	触摸按键中断	0x0
HWRST	[28]	W	异常复位中断	0x0
SCTOVF	[29]	W	扫描时间超时中断	0x0
BLUPD	[30]	W	基准值更新中断	0x0
PRCDNE	[31]	W	全部通道扫描完成中断	0x0

该寄存器用于清除原始中断源标志位，该寄存器为只写寄存器

0： 无效果
1： 清除中断标志

14.3.24 TCH_CHxCNT (采样计数器值寄存器)

- Address = Base Address + 0x0058 ~ Base Address + 0x0094, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
TCH_CHxCNT[0]	[15..0]	R	通道0原值采样值寄存器	0x0
TCH_CHxCNT[1]	[15..0]	R	通道1原值采样值寄存器	0x0
TCH_CHxCNT[2]	[15..0]	R	通道2原值采样值寄存器	0x0
TCH_CHxCNT[3]	[15..0]	R	通道3原值采样值寄存器	0x0
TCH_CHxCNT[4]	[15..0]	R	通道4原值采样值寄存器	0x0
TCH_CHxCNT[5]	[15..0]	R	通道5原值采样值寄存器	0x0
TCH_CHxCNT[6]	[15..0]	R	通道6原值采样值寄存器	0x0
TCH_CHxCNT[7]	[15..0]	R	通道7原值采样值寄存器	0x0
TCH_CHxCNT[8]	[15..0]	R	通道8原值采样值寄存器	0x0
TCH_CHxCNT[9]	[15..0]	R	通道9原值采样值寄存器	0x0
TCH_CHxCNT[10]	[15..0]	R	通道10原值采样值寄存器	0x0
TCH_CHxCNT[11]	[15..0]	R	通道11原值采样值寄存器	0x0
TCH_CHxCNT[12]	[15..0]	R	通道12原值采样值寄存器	0x0
TCH_CHxCNT[13]	[15..0]	R	通道13原值采样值寄存器	0x0
TCH_CHxCNT[14]	[15..0]	R	通道14原值采样值寄存器	0x0
TCH_CHxCNT[15]	[15..0]	R	通道15原值采样值寄存器	0x0

14.3.25 TCH_CHxBL (基准值寄存器)

- Address = Base Address + 0x0098 ~ Base Address + 0x00D4, Reset Value = 0x0000_0000

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

Name	Bit	RW	Description	Reset Value
TCH_CHxBL[0]	[15..0]	RW	通道0参考值寄存器	0x0
TCH_CHxBL[1]	[15..0]	RW	通道1参考值寄存器	0x0
TCH_CHxBL[2]	[15..0]	RW	通道2参考值寄存器	0x0
TCH_CHxBL[3]	[15..0]	RW	通道3参考值寄存器	0x0
TCH_CHxBL[4]	[15..0]	RW	通道4参考值寄存器	0x0
TCH_CHxBL[5]	[15..0]	RW	通道5参考值寄存器	0x0
TCH_CHxBL[6]	[15..0]	RW	通道6参考值寄存器	0x0
TCH_CHxBL[7]	[15..0]	RW	通道7参考值寄存器	0x0
TCH_CHxBL[8]	[15..0]	RW	通道8参考值寄存器	0x0
TCH_CHxBL[9]	[15..0]	RW	通道9参考值寄存器	0x0
TCH_CHxBL[10]	[15..0]	RW	通道10参考值寄存器	0x0
TCH_CHxBL[11]	[15..0]	RW	通道11参考值寄存器	0x0
TCH_CHxBL[12]	[15..0]	RW	通道12参考值寄存器	0x0
TCH_CHxBL[13]	[15..0]	RW	通道13参考值寄存器	0x0
TCH_CHxBL[14]	[15..0]	RW	通道14参考值寄存器	0x0
TCH_CHxBL[15]	[15..0]	RW	通道15参考值寄存器	0x0

14.3.26 TCH_TKEYST (触控按键状态寄存器)

- Address = Base Address + 0x00D8, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																TKEYST15	TKEYST14	TKEYST13	TKEYST12	TKEYST11	TKEYST10	TKEYST9	TKEYST8	TKEYST7	TKEYST6	TKEYST5	TKEYST4	TKEYST3	TKEYST2	TKEYST1	TKEYST0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Name	Bit	RW	Description	Reset Value
TKEYSTx	[15..0]	R	当前通道按键状态查询值	0x0

14.3.27 TCH_BLUPINF (基准值更新状态寄存器)

- Address = Base Address + 0x00DC, Reset Value = 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																BLUPD_CH15	BLUPD_CH14	BLUPD_CH13	BLUPD_CH12	BLUPD_CH11	BLUPD_CH10	BLUPD_CH9	BLUPD_CH8	BLUPD_CH7	BLUPD_CH6	BLUPD_CH5	BLUPD_CH4	BLUPD_CH3	BLUPD_CH2	BLUPD_CH1	BLUPD_CH0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bit	RW	Description	Reset Value
BLUPD_CHx	[15..0]	RW	参考值自动更新通道信息查询 被自动更新过的通道对应位将被置位成'1'。该寄存器不会自动清除，只有通过软件清除。	0x0

15 电气特性

15.1 极限参数

器件在超过下述“极限参数”条件下工作可能会造成永久损坏。 器件只有在说明书所规定的条件范围内才能确保正常工作，在“极限参数”条件下工作会影响器件的可靠性。

Table 15-1 极限参数

参数	符号	条件	数值	单位
工作电压	V_{DD}	–	–0.1 to 6.5	V
输入电压	V_{IN}	–	–0.1 to V_{DD} + 0.3	V
输出电压	V_O	所有端口	–0.1 to V_{DD} + 0.3	V
		普通端口	15	mA
		强下拉驱动端口	120	mA
上拉电流	I_{OH}	所有端口	15	mA
工作环境温度	T_A	–	–40 to 85	°C
储存温度	T_{STG}	–	–65 to 150	°C

15.2 推荐工作条件

器件需要在推荐的工作条件下才能正常工作。 本章所列电气特性参数需要在推荐条件下才能得到确保。器件在超出推荐条件以外的工作条件下工作可能会降低其可靠性，甚至造成器件损坏。

Table 15-2 推荐工作条件

参数	符号	条件	数值	单位
工作电压	V_{DD}	–	2.4 to 5.5	V
工作环境温度	T_A	–	–40 to 85	°C

15.3 I/O 端口特性

Table 15-3 I/O 端口特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
输入高电压	V_{IH}	所有端口	0.7 V_{DD}	–	V_{DD}	V
输入低电压	V_{IL}	所有端口	–	–	0.3 V_{DD}	V
输出高电压	V_{OH}	I_{OH} = -15mA, V_{DD} = 5V	V_{DD} – 1.0	–	–	V
输出低电压	V_{OL1}	I_{OL1} = 15mA , V_{DD} = 5V （所有端口）	–	–	1	V
	V_{OL2}	I_{OL2} = 120mA , V_{DD} = 5V (PA0.11 ~ PA0.8强下拉驱动模式)	–	–	1	V
高输入漏电流	I_{LIH}	除X_{IN}外所有端口, V_{IN} = V_{DD}	–	–	1	uA
低输入漏电流	I_{LIL}	除X_{IN}外所有端口, V_{IN} = 0	–	–	–1	uA
上拉电阻	R_{PU}	V_{DD} = 5V, V_{IN} = 0V	25	50	75	kΩ
下拉电阻	R_{PD}	V_{DD} = 5V, V_{IN} = 5V	25	50	75	kΩ

15.4 输入复位特性

Table 15-5 输入复位特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
最小低压脉宽	T_{NRST}	-	100	300	500	nS
nRESET 迟滞电压	V_{hyst}	上升/下降		1		V

NOTE: 输入复位信号的滤波器宽度为 100ns 至 500 ns。
如果输入复位信号宽度低于 100ns 将被认为无效信号（不复位）。
如果输入复位信号宽度高于 500ns 将被认为有效信号（复位）。

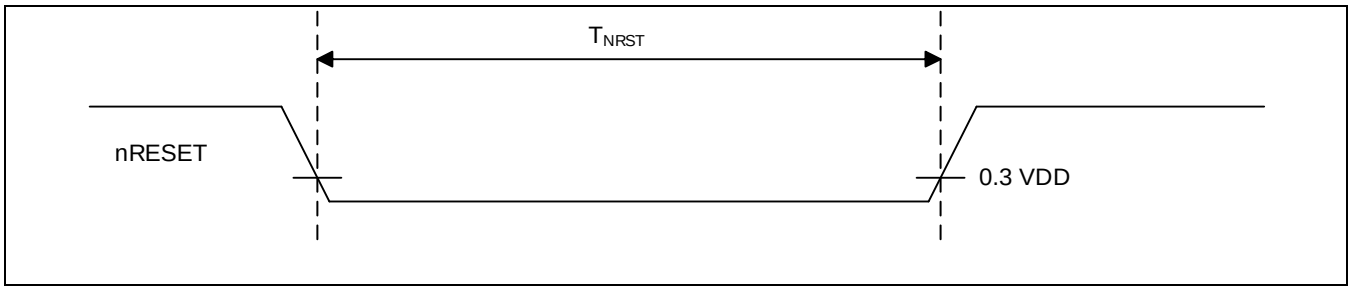


Figure 15-1 nRESET 输入时序

15.5 上电和掉电复位特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
上电电源变化速率	SR_{VDD}	-	0.1	-	-	V/mS
掉电 (Brown-out) 复位电压	V_{BO}	-	-	0.3	-	V
掉电保持时间	T_{BO}	-	10	-	-	mS

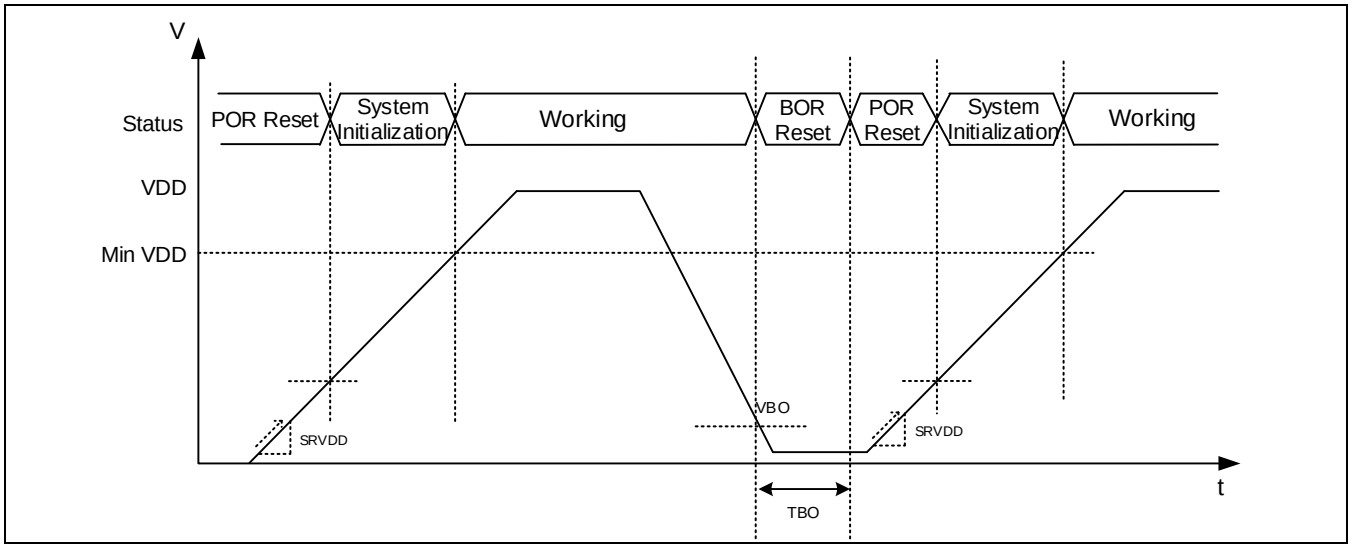


Figure 15-2 上电和掉电示意图

15.6 外部中断输入特性

Table 15-6 外部中断输入特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
中断输入高脉宽	t_{INTH}	V_{DD} = 5.0V	15	30	45	nS
中断输入低脉宽	t_{INTL}	V_{DD} = 5.0V	15	30	45	nS

NOTE: 输入复位信号的滤波器宽度为 15ns 至 45 ns。
如果输入复位信号宽度低于 15ns 将被认为无效信号。
如果输入复位信号宽度高于 45ns 将被认为有效信号。

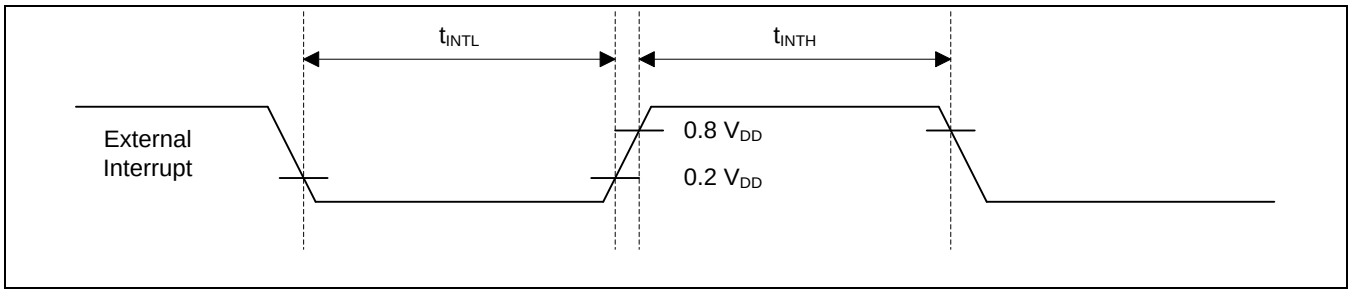


Figure 15-3 外部中断输入时序

15.7 振荡器特性

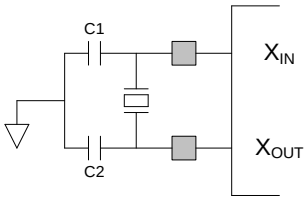
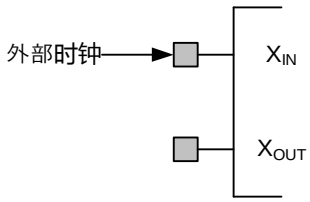
系统中包括三种振荡器：

- 外部主振荡器
- 内部主振荡器
- 内部副振荡器

15.7.1 外部主振荡器

Table 15-7 外部主振荡器特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
振荡器频率	F_{EMOSC}	-	0.4	-	24	Mhz
内部反馈电阻	R_{FD}	XIN 端口	2	4	10	MΩ
稳定时间	T_{STA}	–	–	20	-	ms
匹配电容	C1/C2	-	-	22	-	pF
外接晶振	–		0.4	–	24	MHz
外部时钟	–		0.4	–	24	MHz

15.7.2 内部主振荡器特性

Table 15-8 内部主振荡器特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
振荡器频率	F_{IMOSC}	-	-	20	-	Mhz
占空比	T_{OD}	-	40	-	60	%
校准后精度 (出厂后即为已校准状态)	T_{ACC}	V_{DD} = 5.0V , T_A = 25°C	-	-	±1	%
		V_{DD} = 2.4V to 5.5V , T_A = –40 to 85°C	-	-	±5	%
稳定时间	T_{STA}	电源电压达到最低工作值后	–	–	10	Clk

15.7.3 内部副振荡器特性

Table 15-9 内部副振荡器特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
振荡器频率	F_{IMOSC}	模式1	-	0.5	-	Mhz
		模式2		3.0		
占空比	T_{OD}	-	40	-	60	%
精度	T_{ACC}		-50		+50	%
稳定时间	T_{STA}	电源电压达到最低工作值后	–	–	10	Clk

15.8 工作电流

Table 15-10 工作电流

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	说明	条件	状态名称	最小值	典型值	最大值	单位
工作电流	I_{DD1}	正常工作	-	RUN	–	5	–	mA
	I_{DD2}	CPU 时钟关闭	-	SLEEP	–	1	–	mA
	I_{DD3}	所有时钟及模拟模块关闭	V_{DD} = 5.0V , T_A = 25°C	DEEP SLEEP	–	0.7	5	uA
			V_{DD} = 2.4V to 5.5V , T_A = –40 to 85°C		–	0.7	10	

NOTE: 工作电流不包括 I/O 端口的上拉、下拉电流.

15.9 低压复位监测特性

Table 15-11 低压复位检测特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
低压复位电压 (V_{DD} 下降沿)	Vthrf	–	2.1	2.2	2.3	V
		–	2.6	2.7	2.8	
		–	3.2	3.3	3.4	
		–	3.5	3.6	3.7	
低压监测电压 (V_{DD} 下降沿)	Vthdf	–	2.4	2.5	2.6	
		–	2.9	3.0	3.1	
		–	3.8	3.9	4.0	
		–	4.0	4.1	4.2	
迟滞电压	ΔV_{LVD}	–	–	200	–	mV
工作电流	I_{CC}	–	–	9	–	uA
关断电流	I_{PD}	–	–	0.1	–	uA

15.10 12位模/数转换器特性（ADC 12位模式）

Table 15-12 12位模/数转换器特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
精度	–	–	–	12	–	Bit
工作电压	V_{ADC}	–	3.0	5	5.5	V
输入电压范围	V_{AIN}	–	0	–	V_{ADC}	V
转换速率	F_S	–	–	–	0.5	MHz
微分非线性	DNL	F_S= 0.5MHz V_{ADC} = 5V	–	-	±2.0	LSB
积分非线性	INL		–	-	±4.0	
偏移误差	TOPOFF		–	-	±10.0	
	BOTOFF		–	-	±10.0	
工作电流	I_{OP}	–	–	1	–	mA
关断电流	I_{PD}	–	–	1	–	μA

NOTE: 以上数据为应用评估结果，非量产测试结果。

15.11 10位模/数转换器特性（ADC 10位模式）

Table 15-13 10位模/数转换器特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
精度	–	–	–	10	–	Bit
工作电压	V_{ADC}	–	3.0	5	5.5	V
输入电压范围	V_{AIN}	–	0	–	V_{ADC}	V
转换速率	F_S	–	–	–	1	MHz
微分非线性	DNL	F_S = 1MHz V_{ADC} = 5V	–	-	±1.0	LSB
积分非线性	INL		–	-	±2.0	
偏移误差	TOPOFF		–	-	±4.0	
	BOTOFF		–	-	±4.0	
工作电流	I_{OP}	–	–	1	–	mA
关断电流	I_{PD}	–	–	1	–	μA

NOTE: 以上数据为应用评估结果，非量产测试结果。

15.12 存储器特性

Table 15-13 RAM和寄存器的特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
数据保持电压(1)	V_{DDDR}	深睡眠模式	1.0	–	V_{DD}	V

NOTE: 1）保证 RAM 中的数据不丢失的最低电压值（深睡眠模式下），或者是保持寄存器的状态的最低电压值（深睡眠模式下）。由设计保证，不在量产中测试。

Table 15-14 FLASH内存的特性

(T_A = –40 to 85°C, V_{DD} = 2.4V to 5.5V)

参数	符号	条件	最小值	典型值	最大值	单位
编程大小	F_{WSIZE}	–	–	4	–	Byte
页面大小	F_{PSIZE}	–	–	1024	–	Byte
编程时间（1Word）	F_{tprog}	–	20	–	–	us
页擦除时间	F_{tpera}	–	2	–	–	ms
全芯片擦除时间	F_{tmera}	–	10	–	–	ms
编程次数	F_{nwe}	–	200,000	–	–	Times
数据保持时间	F_{tdr}	–	20	–	–	Years
功耗（编程或擦除时）	F_{idd}	–	–	–	5	mA

15.13 静电防护（ESD）特性

Table 15-15 静电防护特性

参数	符号	模型	最小值	典型值	最大值	单位
静电防护耐压	V_{ESD}	HBM	4000	–	–	V
		MM	200	–	–	V
		CDM	500	–	–	V

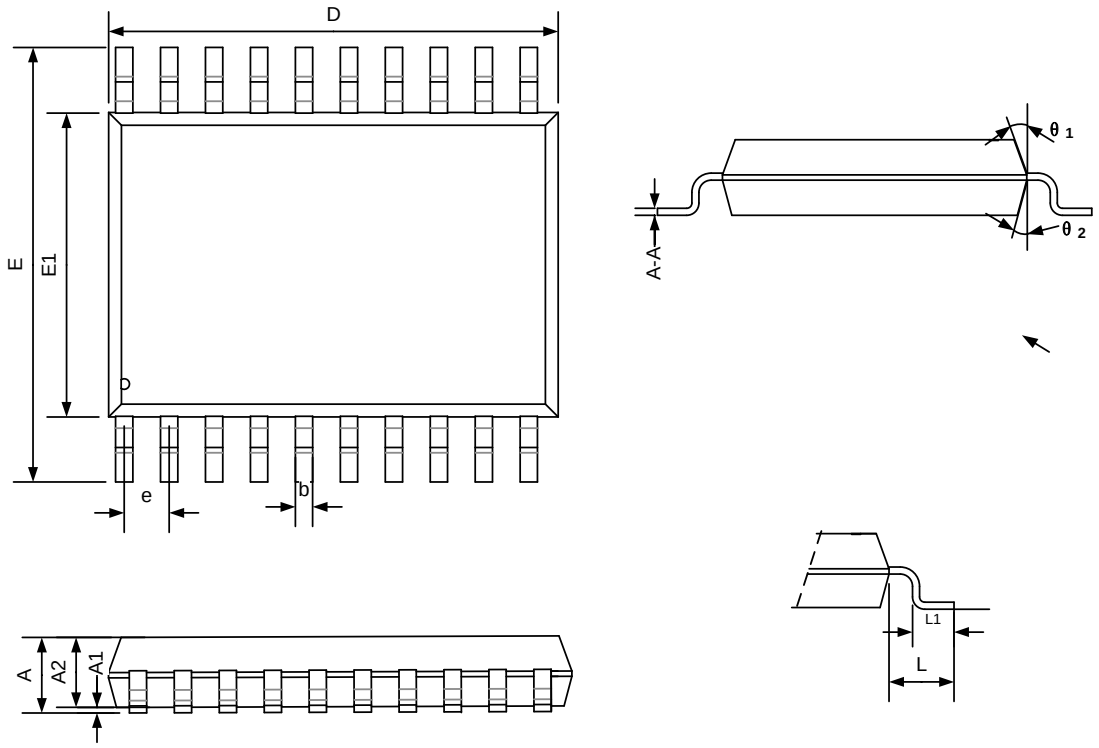
封装尺寸

Package Dimension

TSSOP20

Revision 1.0

December 2016



尺寸 / 标注	最小（mm）	最大（mm）	尺寸 / 标注	最小（mm）	最大（mm）
A		1.20	b	0.19	0.30
A1	0.05	0.15	e	0.65	
A2	0.80	1.05	A-A	0.09	0.20
E	6.25	6.55	θ1	12°TYP	
E1	4.30	4.50	θ2	12°TYP	
D	6.40	6.60			
L		1.00			
L1	0.45	0.75			

TSSOP20 封装尺寸