深圳富芯电子

TX8C101x 用户手册

增强型 8 位通用单片机

Rev. V3.3

深圳富芯电子科技有限公司 http://www.fxmcu.com/

修订记录

日期	版本	描　　述	修订人
2026-5-07	V3.3	1、 更新 FLASH_TRIM 寄存器描述；	FX
2024-11-07	V3.2	1、更新 LVD 检测电压档位的描述；	FWF
2024-09-04	V3.1	1、更新 LVD_CON0[3:2]的上电掉电阈值描述 2、更新 AIP_ACON4[1]的内部参考电压描述 3、更新 WKUP_CON0 和 WKUP_PND 寄存器符号	FWF
2024-08-22	V3.0	1、更新 LVD_CON0[3:2]的 LVD 检测电压档位调整描述	ZJ
2023-10-20	V2.9	1、更新 P07 和 P10 的多功能模式描述	FWF
2023-08-31	V2.8	1、更新页脚的公司英文名描述；	FWF
2023-08-30	V2.7	1、更新 logo 图案；	FWF
2023-08-21	V2.6	1、修改 ADC 参考电压的描述；	FWF
2023-05-31	V2.5	1、修改 ADC 参考电压的描述；	FWF
2023-05-17	V2.4	1、修改 GPIO 低功耗唤醒的描述，更改为电平唤醒； 2、将唤醒寄存器的详细说明移至低功耗管理； 3、修改 ADC 内部参考电压的描述；	FWF
2023-02-03	V2.3	1、解决部分设备打开阅读时，出现乱码问题；	HLW
2022-10-20	V2.2	1、修改比较器部分功能说明；2、修改 LP_CON 说明；	ZJ、HLW
2022-08-10	V2.1	修改 ADC 模块的分频比描述。	HLW
2022-07-28	V2.0	将型号改为 TX8C101x，统一了 TX8C1010 和 TX8C1011 系列。	HLW
2022-07-21	V1.13	修改第一章节，缩减封装型号。	ZJ
2022-04-17	V1.12	修改 SPI_CON 寄存器 bit2 对采样模式的错误描述。	YLC
2022-03-18	V1.11	修改 AIP_CON1 寄存器第 1 和 0 位的错误描述。	ZJ
2022-03-10	V1.10	修改模拟比较器 CON4 的 N 管 P 管开关功能的描述。	YLC
2022-02-25	V1.9	修改模拟比较器 P03 PAD 短路保护功能的描述。	ZJ
2022-02-23	V1.8	修改 TX8C101xSO14 PIN8 引脚 P14 的错误描述	ZJ
2022-02-11	V1.7	Flash 控制模块添加地址说明	HLW

2021-12-28	V1.6	修改 SOP8 的封装信息	ZJ、HLW
2021-12-17	V1.5	修改 SYS_CON2 第 6 和 7 位描述错误，修改引脚图描述	ZJ
2021-12-07	V1.4	添加 TX8C101xQF16 的引脚图.	HLW
2021-08-23	V1.0	初始版本，未来有更新时恕不另行通知，请联系我司人员获取最新版本.	ZJ、HLW

目 录

1. 产品概述

1.1. 说明

TX8C101x 是一款高性能低功耗的 8051 内核 MCU，工作主频最高为 32MHz，内置 4K+256 字节闪存存储器（支持类 EEPROM），512 字节 SRAM。

模拟资源：1 个 12 位 200Ksps 的 ADC、2 个多功能比较器。

定时器、PWM 资源（两者是互斥功能，同一个 Timer 不能同时使用）：

- 2 个 16 位高级定时器，能支持 2 对互补输出或 4 个独立 PWM 输出（周期相同，占空比独立配置）
- 1 个 16 位通用定时器（都支持 Capture、Count、PWM 功能）
- 2 个 8 位通用定时器（可合成 1 个 16 位通用定时器，都支持 Capture、Count、PWM 功能）

标准的通信接口：1 个 SPI 接口、2 个 UART 接口、1 个 IIC 接口（只有 TX8C1011 系列支持）。

支持宽范围电压供电，工作电压为 2.4V ~ 5.5V（可以支持电池应用场景），工作温度范围-40℃ ~ 85℃。多种省电工作模式保证低功耗应用的要求，最低功耗模式 3uA。

TX8C1010 提供 SOP8、MSOP10、SOP14、SOP16、QFN16 共计 5 种封装形式，根据不同的封装形式，器件中的外设资源配置不尽相同。

应用场合：

- 小家电
- 电子烟
- 蓝牙充电仓、无线充
- 玩具

1.2. 特性

- **内核**
  - 超高速 8051 内核（1T）
  - 指令全兼容传统 8051
  - 工作最大主频：32MHz
  - 14 个中断源，支持硬件两级优先级
  - 支持在线下载
  - 支持代码加密
  - 支持带电烧录
  - TX8C1011 系列支持在线调试功能
- **工作电压**
  - 2.4V~5.5V 宽电压范围供电
- **存储器**
  - 4K+256 字节 Flash，用于存储用户代码，并且支持类 EEPROM（擦写次数典型值 10 万次）
  - 512 字节 RAM
- **时钟**
  - 内部 1~32MHz 高精度 HIRC，支持校准（误差±1%）
  - 内部 64KHz 低速 LIRC，支持校准（误差±1%）
  - 外部 32.768 KHz 低速晶振，需要外部加电容
- **复位**
  - 上电复位
  - 欠压复位
  - 复位脚复位
  - 看门狗溢出复位

- GPIO
  - 最多可达 14 个 GPIO
  - 所有端口均可输入输出 5V 信号
  - 均支持上升沿/下降沿/双边沿中断
  - 均支持唤醒功能
  - 有全驱动和小驱动两个档位。
  - 支持 OD 输出低模式。
  - 支持独立控制的上下拉电阻，阻值 30KΩ
- LVD低压检测复位
  - 提供 4 级低压检测电压（1.8/2.0V、2.4/2.6V、2.9/3.1V、3.4/3.6V）
- 数字外设
  - 1 个 SPI 高速串行接口，支持主从模式
  - 1 个 I2C 接口，支持多主和从机模式（只有 TX8C1011 系列支持）
  - 2 个 UART 接口，最大支持 4Mbps
- 定时器资源
  - 2 个 16 位高级定时器，能支持 2 对互补输出或 4 个独立 PWM 输出（周期相同，占空比独立配置），支持死区插入和事件刹车功能，支持单脉冲模式
  - 1 个 16 位通用定时器，都支持 Capture、Count、PWM 功能
  - 2 个 8 位通用定时器（可合成 1 个 16 位通用定时器，都支持 Capture、Count、PWM 功能），可以支持红外发送和接收功能（需要两个 Timer）
  - 1 个看门狗定时器
- 高安全性
  - 支持 16 bit CRC 效验，保证数据准确性
- 低功耗
  - 支持 Idle、Stop、Sleep 低功耗模式

- 静态功耗 3uA @25℃
- 低功耗唤醒时间小于 100us

- **1 个高精度 12 位模数转换器（ADC）**
  - 转换时钟最快支持 4MHz，最快速度 200Ksps
  - 失调校正 step 2mV，DNL +-2 INL +-4
  - 13 个外部输入通道，2 条模拟通路
  - ADC 有效位约 10bit（5V 稳压器供电，ADC 通过内部开关接到芯片的 VCC，以此电压作为 ADC 的参考电压，ADC 满量程等于 VCC）
- **2 个模拟比较器(ACMP)**
  - 2 个低失调比较器，校正 step 1mV
  - 比较器支持负端输入精准 BG 或者 VDDADC 的 120 个分压档位
  - 两个比较器都支持轨到轨输入模式，正负端各支持 2 个 GPIO 可选
  - 支持干吸保护
  - 支持短路保护
- **高可靠性**
  - ESD HBM 8KV
  - Latch-up ±200mA @25℃
- **96 位的芯片唯一ID（UID）**
- **封装**
  - Die Form
  - SOP8/MSOP10/SOP14/SOP16/QFN16
- **工作温度范围**
  - -40℃ ~ 85℃

2. 中央处理器

TX8C101x 全兼容传统的 8051 微控制器，所有指令的助记符和二进制码都和 8051 兼容。TX8C101x 的处理器采用了一些体系结构上的优化，扩展了 SP、DPTR 等常用的寄存器，相比传统的 8051 在性能上面有了很大的提升。

TX8C101x 内部的 ALU 配合内部的 ACC（0xE0）、B（0xF0）、PSW（0xD0）寄存器可以实现各种 8 位运算操作。

ALU 可以进行典型操作如下：

- 基本算术运算：加法、减法、乘法、除法
- 其他算术运算：自加、自减、BCD调整、比较
- 逻辑运算：与、或、异或、取反、移位
- 布尔比特运算：置位、清零、取反、按位判断跳转、进位操作

2.1. 累加器（ACC）

ALU 是 8Bit 宽的算术逻辑单元，MCU 所有的数学、逻辑运算均通过它来完成。它可以对数据进行加、减、移位及逻辑运算；ALU 也控制状态位（PSW 状态寄存器中），用来表示运算结果的状态。

ACC 寄存器是一个 8Bit 的寄存器，ALU 的运算结果可以存放在此。

Addr = 0xE0（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	ACC	累加器寄存器	RW	0x0

2.2. 寄存器（B）

B 寄存器在使用乘法和除法指令时使用,乘法结果高 8bit，除法结果低 8bit。如不使用乘除法指令，也可作为通用寄存器使用。

Addr = 0xF0（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	B	B 寄存器	RW	0x0

2.3. 堆栈指针寄存器（SP）

SP 寄存器指向堆栈的低 8bit 地址，复位后默认值为 0x07，该 SP 的值可以修改。影响 SP 的操作有：指令 PUSH、LCALL、ACALL、POP、RET、RETI 以及进入中断。

Addr = 0x81（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	SP	堆栈指针寄存器	RW	0x7

2.4. 堆栈指针寄存器（SPH）

SPH 寄存器指向堆栈高 8bit 地址,有效位 1bit，复位后默认值位 0x0，与 SP 组合使用，意味着堆栈的区域从 RAM 地址的 0x07 开始。该值可以修改，如果将堆栈区域设置为 0x0B 开始，则在复位后将 SPH 和 SP 的值分别设置为 0x0 和 0x0A。

影响 SPH 的操作有：指令 PUSH、LCALL、ACALL、POP、RET、RETI 以及进入中断。

Addr = 0x9B（SFR）

Bit(s)	Name	Description	R/W	Reset
7:1	-	-	-	-
0	SPH	堆栈指针寄存器高位	RW	0x0

2.5. 数据指针寄存器（DPTR0/DPTR1）

数据指针主要用在 MOVX，MOVC 指令中，其作用是定位 RAM 与 ROM 的地址。芯片内部有两个数据指针寄存器 DPTR0 与 DPTR1，通过 DPSEL 寄存器选择。

每组指针包括两个 8 位寄存器：DPTR0={DPH0,DPL0}、DPTR1={DPH1,DPL1}。

Addr = 0x82（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	DPL0	DPTR0 数据指针寄存器低八位	RW	0x0

Addr = 0x83（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	DPH0	DPTR0 数据指针寄存器高八位	RW	0x0

Addr = 0x84（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	DPL1	DPTR1 数据指针寄存器低八位	RW	0x0

Addr = 0x85（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	DPL1	DPTR1 数据指针寄存器高八位	RW	0x0

2.6. 数据指针控制寄存器（DPCFG）

Addr = 0x86（SFR）

Bit(s)	Name	Description	R/W	Reset
7:6	IA	**中断起始地址和增加方式选择位**	RW	0x2

		IA[0] 0x0：中断地址=（中断向量*8）+3 0x1：中断地址=（中断向量+1）*3 IA[1] 0x0：中断起始地址为 0x300 0x1：中断起始地址为 0x8000		
5	DPID0	**DPTR0 加 1/减 1** 0x0：DPTR0 加 1 0x1：DPTR0 减 1	RW	0x0
4	DPID1	**DPTR1 加 1/减 1** 0x0：DPTR1 加 1 0x1：DPTR1 减 1	RW	0x0
3	DPAID	**DPTR0/DPTR1 自加自减使能位** 0x0：关闭 0x1：打开	RW	0x0
2	DPTSL	**DPSEL 自动翻转使能位** 0x0：关闭 0x1：打开	RW	0x0
1	-	-	-	-
0	DPSEL	**选择 DPTR0 /DPTR1 位** 0x0：DPTR0 有效 0x1：DPTR1 有效	RW	0x0

2.7. 程序状态寄存器（PSW）

Addr = 0xD0（SFR）

Bit(s)	Name	Description	R/W	Reset
7	CY	**进位标志位** 0x0：无进位 0x1：有进位	RW	0x0
6	AC	**辅助进位标志位** 0x0：无进位 0x1：有进位	RW	0x0
5	F0	**通用标志位 0**	RW	0x0
4:3	RS1,RS0	**寄存器组选择位** 0x0：寄存器组 0 0x1：寄存器组 1 0x2：寄存器组 2 0x3：寄存器组 3	RW	0x0
2	OV	**溢出标志位** 0x0：算术或逻辑运算无溢出 0x1：算术或逻辑运算有溢出	RW	0x0
1	F1	**通用标志位 1**	RW	0x0
0	P	**奇偶校验标志位** 0x0：ACC 中 1 为偶数 0x1：ACC 中 1 为奇数	RW	0x0

2.8. 程序计数器（PC）

程序计数器（PC）控制程序内存 FLASH 中的指令执行顺序，它可以寻址整个 FLASH 的范围，取得指令码后，程序计数器（PC）会自动加一，指向下一个指令码的地址。但如果执行跳转、条件跳转、向 PCL 赋值、子程序调用、初始化复位、中断、中断返回、子程序返

回等操作时，PC 会加载与指令相关的地址而不是下一条指令的地址。

当遇到条件跳转指令且符合跳转条件时，当前指令执行过程中读取的下一条指令将会被丢弃，且会插入一个空指令操作周期，随后才能取得正确的指令。反之，就会顺序执行下一条指令。

3. 存储器

TX8C101x 有内部有 3 种存储器：IDATA、XDATA、程序存储器。

程序存储只能读不能写，程序存储器大小为 4K 字节。XDATA 大小为 1K 字节（其中 XSFR 为 512 字节，XDATA 大小为 256 字节，IDATA 大小为 256 字节）。

3.1. 程序存储器

TX8C101x 的程序指针为 16 位，最大寻址空间可达 64K 字节，实际只实现了 4K 字节的程序存储空间。

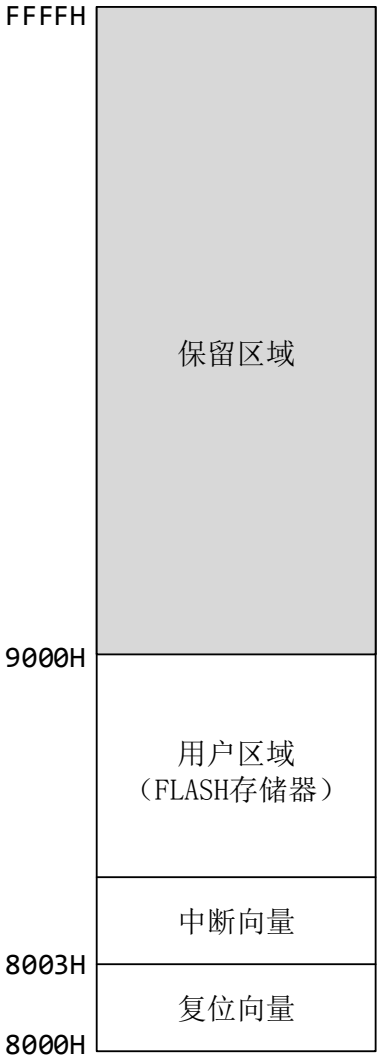


图 3-1 程序存储空间

复位后，MCU 从 0x8000 开始执行。从 0x8003 开始是中断向量表，当发生中断且中断使能后，PC 会跳转到对应的中断向量位置去执行。

3.2. XDATA

XDATA 共有 256 字节（包含于 1K 字节的 XDATA），地址为 0x300~0x3FF，可用于间接寻址的数据存储。

3.3. IDATA

内部数据存储器空间大小为 256 字节。

内部数据存储器的地址空间的低 128 字节可以字节访问，高 128 字节和 SFR 共用一个地址空间，直接访问高 128 字节会访问到 SFR 空间，高 128 字节数据存储器只能通过间接寻址方式访问。

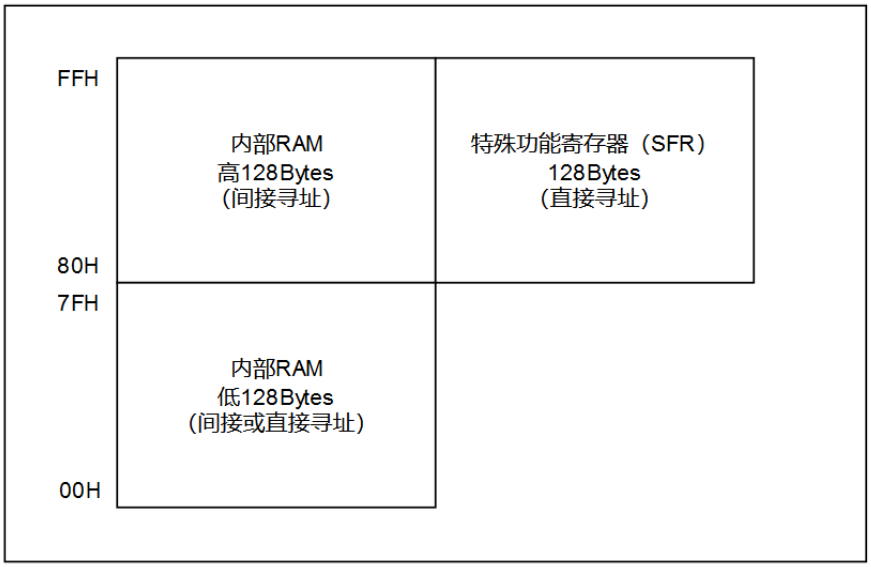


图 3-2 数据存储器

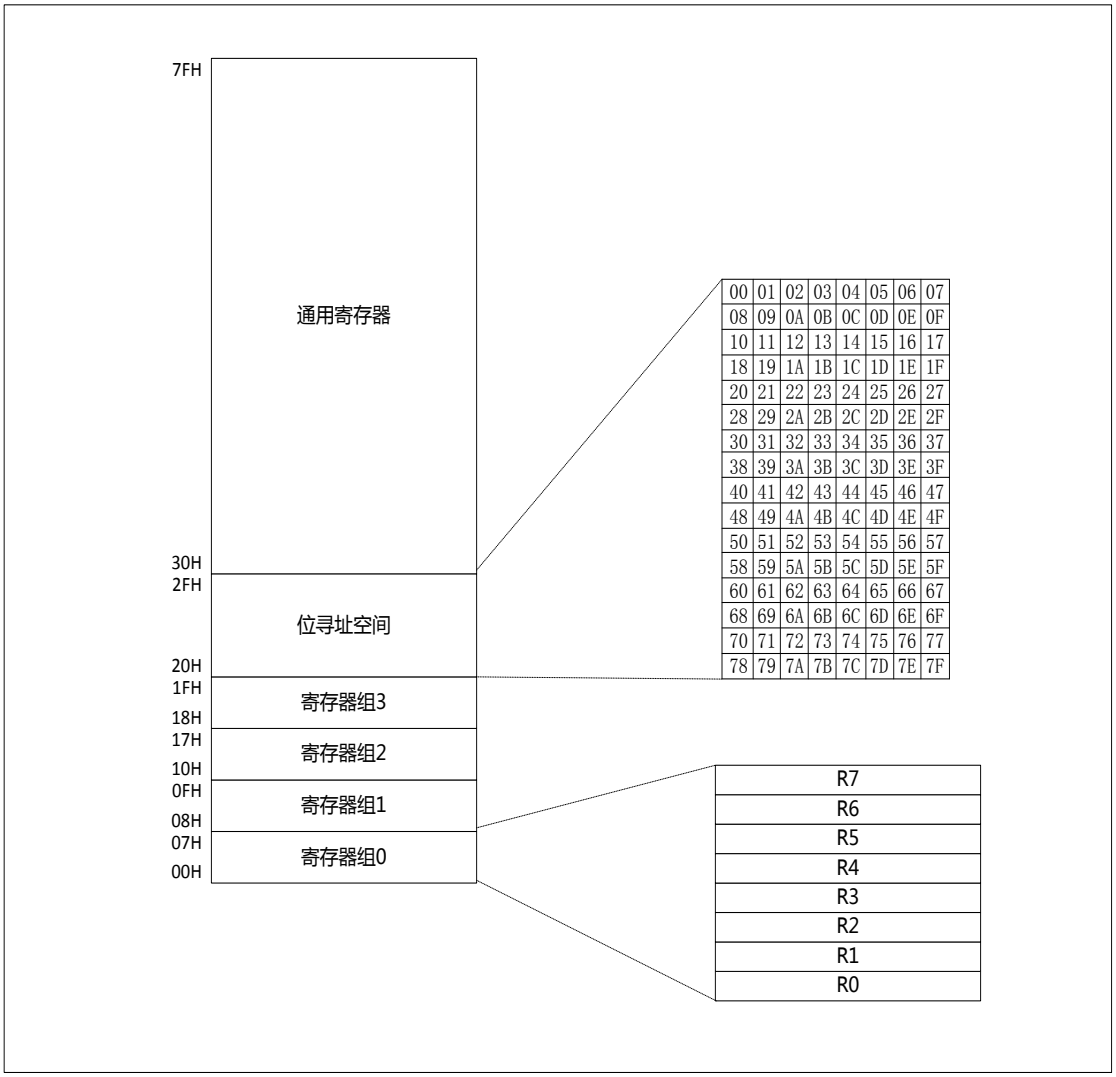


图 3-3 内部低 128 字节数据空间分配

3.4. SFR 空间

	0H/8H	1H/9H	2H/AH	3H/BH	4H/CH	5H/DH	6H/EH	7H/FH
F8H	UART0_BAUD0	UART0_BAUD1	UART0_DATA	UART1_CON	UART1_STA	UART1_BAUD0	UART1_BAUD1	UART1_DATA
F0H	B	SPI0_CON	SPI0_BAUD	SPI0_DAT	SPI0_STA	STMR_ALLCON	UART0_CON	UART0_STA
E8H	STMR2_BRAKE	STMR2_DTR	STMR2_PCONRA	STMR2_PCONRB	STMR2_IE	STMR2_SR	ADK_CFG3	-
E0H	ACC	STMR2_CMPAH	STMR2_CMPBL	STMR2_CMPBH	STMR2_CR	STMR2_FCONR	STMR2_VPERR	STMR2_DTUA
D8H	STMR1_PCONRB	STMR1_IE	STMR1_SR	STMR2_CNTL	STMR2_CNTH	STMR2_PRL	STMR2_PRH	STMR2_CMPAL
D0H	PSW	STMR1_CR	STMR1_FCONR	STMR1_VPERR	STMR1_DTUA	STMR1_BRAKE	STMR1_DTR	STMR1_PCONRA
C8H	STMR1_CNTL	STMR1_CNTH	STMR1_PRL	STMR1_PRH	STMR1_CMPAL	STMR1_CMPAH	STMR1_CMPBL	STMR1_CMPBH
C0H	TMR2_CONL	TMR2_CONH	TMR2_CNTL	TMR2_CNTH	TMR2_PRL	TMR2_PRH	TMR2_PWML	TMR2_PWMH
B8H	IP0	IP1	LVD_CON0	LVD_CON1	LVD_CON2	LVD_CON3	LP_CON	SYS_PND
B0H	TMR1_CONL	TMR1_CONH	TMR1_CNTL	TMR1_CNTH	TMR1_PRL	TMR1_PRH	TMR1_PWML	TMR1_PWMH
A8H	IE0	IE1	FLASH_TRIM	WKUP_CON0	WKUP_PND	WDT_CON	WDT_KEY	-
A0H	FLASH_CON	FLASH_STA	FLASH_DATA	FLASH_TIM0	FLASH_TIM1	FLASH_CRCLEN	FLASH_PASSWORD	FLASH_ADDR
98H	ADK_CHS0	ADK_CHS1	ADK_CFG2	SPH	PCON1	ADK_CON	CRC_REG	CRC_FIFO
90H	P1	ADC_CFG0	ADC_CFG1	ADC_STA	ADC_DATAH0	ADC_DATAL0	ADC_DATAH1	ADC_DATAL1
88H	TMR0_CONL	TMR0_CONH	TMR0_CNTL	TMR0_CNTH	TMR0_PRL	TMR0_PRH	TMR0_PWML	TMR0_PWMH
80H	P0	SP	DPL0	DPH0	DPL1	DPH1	DPS	PCON0

4. 系统时钟

4.1. 时钟系统概述

系统片上有一个 32MHz 的高速高精度 RC 振荡器，和一个外接的 32.768KHz 的低速晶体振荡器，以及内部 PMU 里面集成了一个 64KHz 的低速 RC 振荡器。

4.2. 时钟系统主要功能

TX8C101x 芯片的时钟源来自于 3 个不同的时钟，分别是片外 32.768KHz 的低速晶振，片内 64K 低速 RC 和片内 32M 高速 RC。如图 4-1 所示，系统时钟可以通过 CLK_CON0[1:0]对上述三个时钟源进行选择，选择后的时钟为芯片工作最快时钟（下文称为 sys_clk_pre）。如图 4-2 所示，sys_clk_pre 再经过 CLK_CON2[3:0]进行分频，分频后时钟为系统时钟（下文称为 sys_clk），系统中大部分外设与模数混合模块都将使用 sys_clk，例如 UART、SPI、CRC16 等外设都使用 sys_clk。如图 4-2 所示，GPIO 口的滤波时钟，Timer2 模块时钟和低电检测这几个特殊的模块会使用 sys_clk，片外 32.768KHz 的低速晶振，片内 64K 低速 RC，片内 32M 高速 RC 分频后时钟进行选择。

4.3. 时钟系统框图

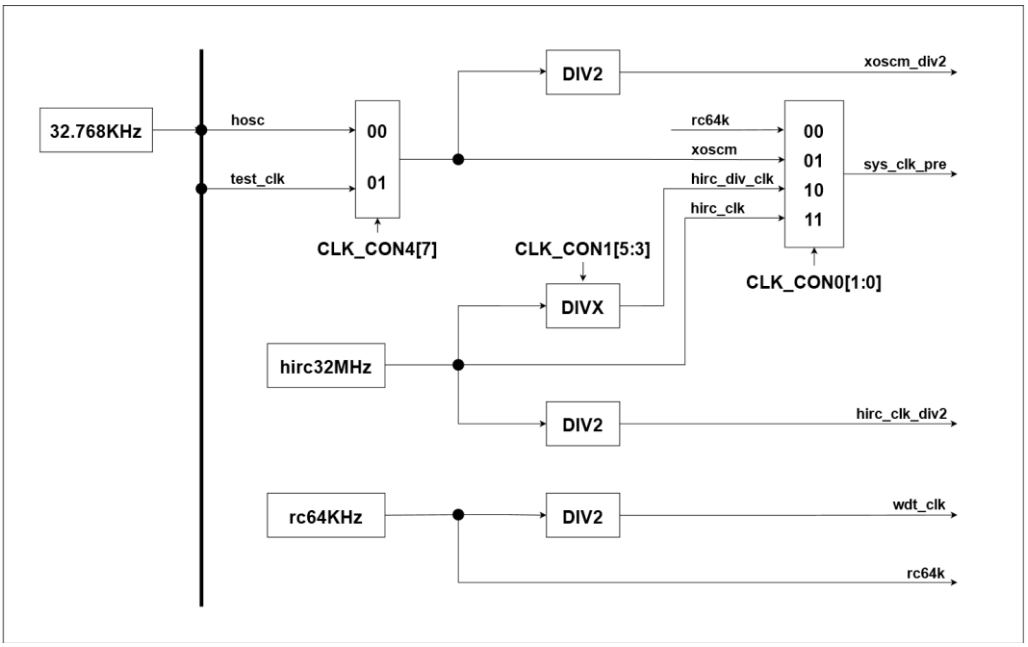


图 4-1 系统时钟框图

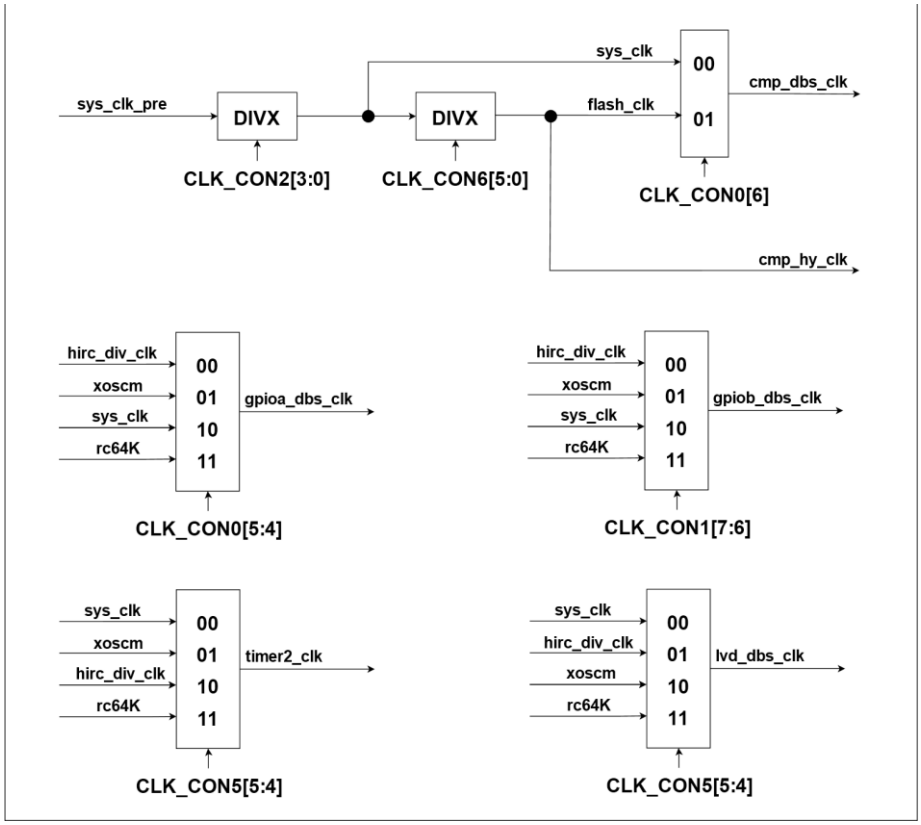


图 4-2 模块时钟框图

4.4. 系统振荡器

4.4.1. 内部低速 RC 振荡器

PMU 内部集成了一个 64KHz 的常开低速 RC 振荡器，当芯片上电时，系统工作在该 64KHz 的时钟，完成上电复位过程，等到系统复位释放以后才开始跑程序。

4.4.2. 内部高速 RC 振荡器

芯片内部集成了一个片上高速 RC 振荡器，支持最大 32MHz 的输出时钟给系统用，默认时关闭的，需要用户在程序中通过配置系统寄存器 AIP_CON0[7]=1，打开高速 RC 振荡器，也可以通过配置 AIP_CON0[7]=0，关闭该时钟源。注意在关闭该时钟前，系统需要先切换到低速 RC 振荡器。该高速 RC 振荡器可以通过量产过程中的校准程序校准，保证其精度满足应用方案需求。

4.4.3. 外部低速晶体振荡器

芯片内部集成了一个晶体振荡器启振电路，可以外部接一个低速 32.768KHz 的无源晶体振荡器，作为系统的工作时钟源。默认时关闭的，可以通过配置系统寄存器 AIP_CON1[3]=1，打开该时钟源；也可以配置 AIP_CON1[3]=0，关闭。

5. 复位系统

5.1. 上电复位

芯片上电的 POR 复位。

5.2. 掉电复位掉电

芯片掉电的 BOR 复位。

5.3. 看门狗复位

芯片有一个独立于系统运行的看门狗模块，用于保护系统异常发生之后的复位重启系统。看门狗模块工作时钟是常开的 64KHz 的低速 RC 的 2 分频时钟，即工作在 32KHz 的独立于系统时钟的时钟。默认配置是 2 秒钟复位一次系统。所以在用户程序中需要在看门狗复位之前要喂狗，使其重新计时。用户可以配置看门狗复位时间间隔范围从 8ms ~ 262s，可以选择看门狗产生中断，不复位。中断和复位只能二选一。

5.3.1. 控制寄存器列表

表 5-1 WDT 寄存器列表

Address	Register Name	Description
0xAD (SFR)	WDT_CON	WDT_CON register
0xAE (SFR)	WDT_KEY	WDT_KEY register

5.3.2. 寄存器详细说明

5.3.2.1. WDT_CON

Addr = 0xAD (SFR)

Bit(s)	Name	Description	R/W	Reset
7	WAKEEN	**WDT 唤醒功能使能位** 写 WDT_KEY=0xEE，置位 写 WDT_KEY=0x22，复位 0x0：关闭 0x1：打开	RO	0x0
6	WDTPND	**WDT 计数器计满标记位** 写 WDT_KEY=0xAA，清掉该标记位 0x0：计数器未计满 0x1：计数器计满	RO	0x0
5	INTEN	**WDT 中断功能使能位** 写 WDT_KEY=0x5A，置位 写 WDT_KEY=0xA5，复位 0x1：打开中断功能 0x0：打开复位功能	RO	0x0
4	WDTE	**WDT 使能位** 写 WDT_KEY=0xCC，置位 写 WDT_KEY=0xDD，复位 0x0：关闭 watchdog 功能 0x1：打开 watchdog 功能	RW	0x1
3:0	PSR	**预分频系数** 每次配置该位域之前必须先写 WDT_KEY=0x55 0x0：不分频 0x1：2 分频 0x2：4 分频 0x3：8 分频	RW	0x8

		0x4：16 分频 0x5：32 分频 0x6：64 分频 0x7：128 分频 0x8：256 分频 0x9：512 分频 0xA：1024 分频 0xB：2048 分频 0xC：4096 分频 0xD：8192 分频 0xE：16384 分频 0xF：32768 分频 看门狗复位时间=1/32K*256*分频系数		

5.3.2.2. WDT_KEY

Addr = 0xAE（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	WDT_KEY	**喂狗数据寄存器** 软件必须以一定的间隔写入 0xAA 完成喂狗操作，否则，当计数器为 0 时，看门狗会产生复位 当 pending 为 1 的时候，写入 0xAA 清除 pending 0x55：表示允许访问和设置 wdt_psr 0xDD：关闭看门狗 0xCC：启动看门狗工作 0xAA：喂狗并清除 wdt_pending 0xA5：关闭中断 0x5A：开启中断 0x22：关闭 wake up 0xEE：开启 wake up	WO	0x0

5.4. 低电检测复位

PMU 内部集成了低电压检测和过流检测功能电路，用于检测 PMU 供电部分异常情况，并可以把检测到低电压和过流等异常情况通过中断方式上报给 CPU 进行系统异常处理程序。另外低电压异常信号可以产生复位信号去复位系统，以免在低电压的情况下电路工作不正常而导致用户程序跑飞。低电压检测的阈值可以通过 LVD 控制寄存器设置。可以通过设置 LVD 控制寄存器对异常信号进行滤波去抖动，避免系统瞬态变化导致的正常电源压降而误发生意外低电复位系统的情况发生。

5.4.1. 控制寄存器列表

表 5-2 LVD 寄存器列表

Address	Register Name	Description
0xBA (SFR)	LVD_CON0	LVD_CON0 register
0xBB (SFR)	LVD_CON1	LVD_CON1 register
0xBC (SFR)	LVD_CON2	LVD_CON2 register
0xBD (SFR)	LVD_CON3	LVD_CON3 register

5.4.2. 寄存器详细说明

5.4.2.1. LVD_CON0

Addr = 0xBA (SFR)

Bit(s)	Name	Description	R/W	Reset
7	-	-	-	-
6	LVDOE	LVD 中断和复位功能输出到系统使能位	RW	0x1

		使用 LVD 所有相关功能，必须把 LVDOE 设置位 1 0x0：关闭 0x1：打开		
5	LVDVDDRSTEN	**LVD VDD 低电压复位功能使能位** 0x0：关闭 0x1：打开	RW	0x1
4	LVDVCCRSTEN	**LVD VCC 低电压复位功能使能位** 0x0：关闭 0x1：打开	RW	0x1
3:2	PMULVD5SET	**VCC 电源电压低电检测阈值设置** 0x0：1.8/2.0V（掉电阈值/上电阈值） 0x1：2.4/2.6V（掉电阈值/上电阈值） 0x2：2.9/3.1V（掉电阈值/上电阈值） 0x3：3.4/3.6V（掉电阈值/上电阈值）	RW	0x0
1	PMULVD15EN	**1.5V 数字逻辑系统工作电压 VDD 低电检测功能使能位** 0x0：关闭 0x1：打开	RW	0x1
0	PMULVD5EN	**VCC 电源 VCC 电压低电检测功能使能位** 0x0：关闭 0x1：打开	RW	0x1

5.4.2.2. LVD_CON1

Addr = 0xBB（SFR）

Bit(s)	Name	Description	R/W	Reset
7	-	-	-	-
6	VDDOCPND	**VDD 过流检测标记位** 写 1 清除标记位 0x0：VDD 没有过流 0x1：VDD 过流	RW	0x0

5	LVDVDDPND	**VDD 低电检测标记位** 写 1 清除标记位 0x0：VDD 没有低电 0x1：VDD 低电	RW	0x0
4	LVDVCCPND	**VCC 低电检测标记位** 写 1 清除标记位 0x0：VCC 没有低电 0x1：VCC 低电	RW	0x0
3	LVDVCCSYNDIS	**LVD VCC 低电检测同步器关闭位** 0x0：打开同步器 0x1：关闭同步器	RW	0x1
2	VDDOCBPSEN	**VDD 过流滤波去抖功能关闭位** 0x0：打开滤波器 0x1：关闭滤波器	RW	0x1
1	LVDVDDBPSEN	**VDD 低电滤波去抖功能关闭位** 0x0：打开滤波器 0x1：关闭滤波器	RW	0x1
0	LVDVCCBPSEN	**VCC 低电滤波去抖功能关闭位** 0x0：打开滤波器 0x1：关闭滤波器	RW	0x1

5.4.2.3. LVD_CON2

Addr = 0xBC（SFR）

Bit(s)	Name	Description	R/W	Reset
7	-	-	-	-
6:0	DBSHLMT	**LVD 低电和过流异常检测滤波器高电平滤波时钟周期设置数目** **Note：** LVD 滤波时钟可以通过系统配置寄存器 CLKCON5[3:2]来选择。用户可以根据使用场景来选择滤波功能。滤波会导致异常发生到系统收到异常会有延迟时间，延迟时间会由设置的滤波的	RW	0x2

		时钟周期和配置滤波高电平和低电平滤波周期数目共同决定，设置的滤波时钟周期越长，滤波周期数目越多会导致该延迟越长。用户在使用时可以通过对该延迟的容忍度来合理配置。在有些对延迟比较敏感的应用场景可以关闭该滤波功能。		

5.4.2.4. LVD_CON3

Addr = 0xBD（SFR）

Bit(s)	Name	Description	R/W	Reset
7	-	-	-	-
6:0	DBSLLMT	**LVD 低电和过流异常检测滤波器低电平滤波时钟周期设置数目** **Note：** LVD 滤波时钟可以通过系统配置寄存器 CLKCON5[3:2]来选择。用户可以根据使用场景来选择滤波功能。滤波会导致异常发生到系统收到异常会有延迟时间，延迟时间会由设置的滤波的时钟周期和配置滤波高电平和低电平滤波周期数目共同决定，设置的滤波时钟周期越长，滤波周期数目越多会导致该延迟越长。用户在使用时可以通过对该延迟的容忍度来合理配置。在有些对延迟比较敏感的应用场景可以关闭该滤波功能。	RW	0x2

6. 低功耗管理

TX8C101x 芯片系统支持 3 个不同功耗等级的低功耗模式，从高到低依次是：Idle Mode、Stop Mode 和 Sleep Mode。其中功耗最低的是 Sleep 低功耗工作模式，该模式下常温整个芯片漏电可以约为 3uA。

6.1. Idle Mode 及唤醒

通过配置系统寄存器 LP_CON[7]=1，进入 Idle Mode。在 Idle 模式下只有 CPU 工作时钟被关闭，CPU 停止工作。通过中断方式唤醒 Idle Mode，唤醒之后会进入当前唤醒 Idle Mode 的中断服务子程序并执行。

6.2. Stop Mode 及唤醒

通过配置系统寄存器 LP_CON[1]= 1，进入 Stop Mode。在 Stop 模式下系统时钟被关闭，CPU 及大部分系统时钟域的外设停止工作。通过选择多种唤醒源来唤醒 Stop Mode，唤醒源包括：所有 GPIO、比较器、基本 Timer2、看门狗、LVDVCC(电源的低电压检测信号)。Stop Mode 唤醒之后会继续跑后面的用户程序。

6.3. Sleep Mode 及唤醒

通过配置系统寄存器 LP_CON[0]=1，进入最低功耗的 Sleep Mode。在 Sleep 模式下系统时钟被关闭，CPU 及大部分系统时钟域的外设停止工作，除了 PMU 以外的模拟模块都要关闭，但是 XOSC 可选择开/关。通过选择多种唤醒源来唤醒 Sleep Mode，唤醒源包括：所有 GPIO 电平变换唤醒、基本 Timer2 定时中断唤醒、看门狗复位唤醒。Sleep Mode 唤醒之后可以通过进入 Sleep 模式之前配置的 LP_CON[6]=1，会继续跑后面的用户程序，如果 LP_CON[6]=0，则会复位系统重新跑用户程序。

6.4. 低功耗唤醒单元结构图

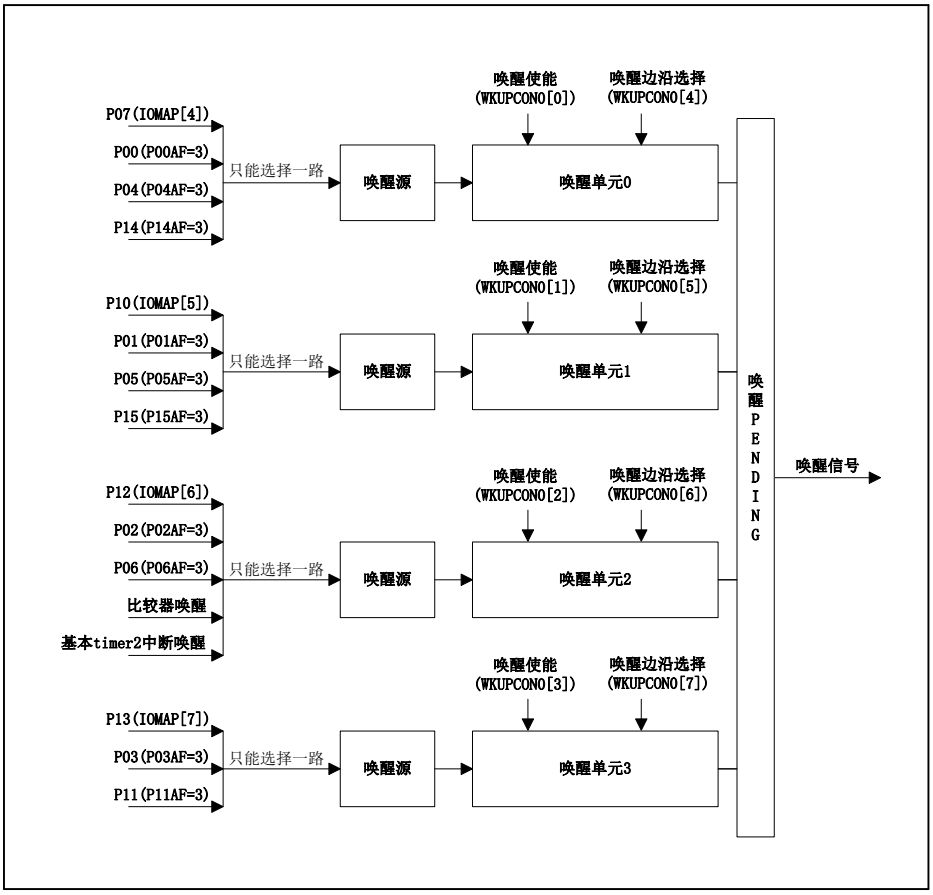


图 6-1 低功耗唤醒结构图

6.5. 寄存器详细说明

6.5.1. WKUP_CON0

Addr = 0xAB（SFR）

Bit(s)	Name	Description	R/W	Reset
7	WKUP3EDG	**低功耗模式 wakeup 通道 3 输入触发电平设置** 0x0：高电平触发唤醒 0x1：低电平触发唤醒	RW	0x0
6	WKUP2EDG	**低功耗模式 wakeup 通道 2 输入触发电平设置** 0x0：高电平触发唤醒	RW	0x0

		0x1：低电平触发唤醒		
5	WKUP1EDG	**低功耗模式 wakeup 通道 1 输入触发电平设置** 0x0：高电平触发唤醒 0x1：低电平触发唤醒	RW	0x0
4	WKUP0EDG	**低功耗模式 wakeup 通道 0 输入触发电平设置** 0x0：高电平触发唤醒 0x1：低电平触发唤醒	RW	0x0
3	WKUP3EN	**低功耗模式 wakeup 通道 3 功能使能位** 0x0：关闭 0x1：打开	RW	0x0
2	WKUP2EN	**低功耗模式 wakeup 通道 2 功能使能位** 0x0：关闭 0x1：打开	RW	0x0
1	WKUP1EN	**低功耗模式 wakeup 通道 1 功能使能位** 0x0：关闭 0x1：打开	RW	0x0
0	WKUP0EN	**低功耗模式 wakeup 通道 0 功能使能位** 0x0：关闭 0x1：打开	RW	0x0

6.5.2. WKUP_PND

Addr = 0xAC（SFR）

Bit(s)	Name	Description	R/W	Reset
7	WKUP3PCLR	**低功耗模式 wakeup 通道 3 清 pending 位** 0x0：无操作 0x1：清 pending	RW	0x0
6	WKUP2PCLR	**低功耗模式 wakeup 通道 2 清 pending 位** 0x0：无操作 0x1：清 pending	RW	0x0
5	WKUP1PCLR	**低功耗模式 wakeup 通道 1 清 pending 位**	RW	0x0

		0x0：无操作 0x1：清 pending		
4	WKUP0PCLR	**低功耗模式 wakeup 通道 0 清 pending 位** 0x0：无操作 0x1：清 pending	RW	0x0
3	WKUP3PND	**低功耗模式 wakeup 通道 3 唤醒 pending 位** 0x0：无 pending 0x1：有 pending	RW	0x0
2	WKUP2PND	**低功耗模式 wakeup 通道 2 唤醒 pending 位** 0x0：无 pending 0x1：有 pending	RW	0x0
1	WKUP1PND	**低功耗模式 wakeup 通道 1 唤醒 pending 位** 0x0：无 pending 0x1：有 pending	RW	0x0
0	WKUP0PND	**低功耗模式 wakeup 通道 0 唤醒 pending 位** 0x0：无 pending 0x1：有 pending	RW	0x0

6.5.3. LP_CON

Addr = 0xBE（SFR）

Bit(s)	Name	Description	R/W	Reset
7	IDLE	**Idle 低功耗模式使能** 0x0：打开，进入 Idle 低功耗模式 0x1：关闭	RW	0x1
6	SLEEPGOEN	**Sleep 低功耗模式唤醒后继续跑后续程序使能位** 0x0：Sleep 模式唤醒后复位重新跑程序 0x1：Sleep 模式唤醒后继续跑后续程序	RW	0x1
5	CPDIS	**用户程序保护功能关闭位** CPU 和 ISD 都无法写寄存器 0x0：打开 （默认状态）	RO	0x0

		0x1：关闭		
4	LPGLIRCEN	**低功耗进入低速 RC 选择** 低功耗 Sleep 模式下，可以通过配置该寄存器为 1，在系统进入 Sleep 模式后自动 gate 住 RC64K 低速时钟，目的是减少 Sleep 的漏电功耗。也可以选择不关闭 RC64K. **Note:**该功能只能在 Sleep 低功耗模式下，由 GPIO 唤醒系统的场景下才能使用该功能。如果是 TIMER2 定时唤醒场景下不可以关闭该时钟，造成 TIMER2 没有工作时钟而不能唤醒系统。	RW	0x0
3	CMCEDIS	**程序存储器空片硬件自动检查功能关闭位** 默认芯片上电以后会自动检查程序存储器是否有烧写程序，如果没有烧写程序，则 hold 住 cpu 不跑任何程序。 0x0：开启空片自动检查功能 0x1：关闭空片自动检查功能	RW	0x0
2	TMHCPU	**测试模式下 hold 住 CPU 使能** **Note:** 用户程序不要随便写这个寄存器，会造成系统功能异常的风险！！！ 0x0：不 hold CPU 0x1：hold CPU	RW	0x0
1	STOP	**Stop 低功耗模式使能** 0x0：关闭 0x1：打开，进入 Stop 低功耗模式	RW	0x0
0	SLEEP	**Sleep 低功耗模式使能** 0x0：关闭 0x1：打开，进入 Sleep 低功耗模式	RW	0x0

7. 系统控制模块

7.1. 功能概述

系统控制模块主要是用来管理和配置系统功能的，包括系统中模拟模块，时钟源，供电系统，时钟管理系统，低功耗及唤醒系统等系统功能配置。

7.2. 寄存器列表

表 7-1 系统寄存器列表

Address	Register Name	Description
0x08 (XSFR)	SYS_CON0	SYS_CON0 register
0x09 (XSFR)	SYS_CON1	SYS_CON1 register
0x0A (XSFR)	SYS_CON2	SYS_CON2 register
0x0B (XSFR)	SYS_CON3	SYS_CON3 register
0x0C (XSFR)	SYS_CON4	SYS_CON4 register
0x0D (XSFR)	SYS_CON5	SYS_CON5 register
0x10 (XSFR)	CLK_CON0	CLK_CON0 register
0x11 (XSFR)	CLK_CON1	CLK_CON1 register
0x12 (XSFR)	CLK_CON2	CLK_CON2 register
0x13 (XSFR)	CLK_CON3	CLK_CON3 register
0x14 (XSFR)	CLK_CON4	CLK_CON4 register
0x15 (XSFR)	CLK_CON5	CLK_CON5 register
0x16 (XSFR)	CLK_CON6	CLK_CON6 register
0x28 (XSFR)	IO_MAP	IO_MAP register
0x2A (XSFR)	IO_MAP1	IO_MAP1 register

0x30 (XSFR)	AIP_CON0	AIP_CON0 register
0x31 (XSFR)	AIP_CON1	AIP_CON1 register
0x32 (XSFR)	AIP_CON2	AIP_CON2 register
0x33 (XSFR)	AIP_CON3	AIP_CON3 register
0x34 (XSFR)	AIP_CON4	AIP_CON4 register
0xAB (SFR)	WKUP_CON0	WKUP_CON0 register
0xAC (SFR)	WKUP_PND	WKUP_PND register
0xBE (SFR)	LP_CON	LP_CON register
0xBF (SFR)	SYS_PND	SYS_PND register

7.3. 寄存器详细说明

7.3.1. SYS_CON0

Addr = 0x08 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	STMR1SOFTRST	**高级 Timer1 软复位** 0x0：软复位 0x1：软复位释放	RW	0x1
6	TMR2SOFTRST	**基本 Timer2 软复位** 0x0：软复位 0x1：软复位释放	RW	0x1
5	TMR1SOFTRST	**基本 Timer1 软复位** 0x0：软复位 0x1：软复位释放	RW	0x1
4	TMR0SOFTRST	**基本 Timer0 软复位** 0x0：软复位	RW	0x1

		0x1：软复位释放		
3	-	-	-	-
2	SPI0SOFTRST	**SPI0 软复位** 0x0：软复位 0x1：软复位释放	RW	0x1
1	UART1SOFTRST	**UART1 软复位** 0x0：软复位 0x1：软复位释放	RW	0x1
0	UART0SOFTRST	**UART0 软复位** 0x0：软复位 0x1：软复位释放	RW	0x1

7.3.2. SYS_CON1

Addr = 0x09（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	MEMDVS[1:0]	**片上 SRAM 动态电压调整值** **Note**：用户千万不要配置该寄存器，否则会导致芯片不确定行为！！！！	RW	0x0
5	FASTRSTEN	**快速复位唤醒 Sleep Mode 使能** 配置此功能主要是为了能在 sleep 低功耗模式下，设置通过复位唤醒 sleep 时，可以节省复位时间 0x0：关闭 0x1：打开	RW	0x0
4	GPIOSOFTRST	**GPIO 模块软复位** 0x0：软复位 0x1：软复位释放	RW	0x1
3	ADCSOFTRST	**ADC 软复位**	RW	0x1

		0x0：软复位 0x1：软复位释放		
2	WDTSOFTRST	**Watchdog 软复位** 0x0：软复位 0x1：软复位释放	RW	0x1
1	CRCSOFTRST	**CRC 软复位** 0x0：软复位 0x1：软复位释放	RW	0x1
0	STMR2SOFTRST	**高级 Timer2 软复位** 0x0：软复位 0x1：软复位释放	RW	0x1

7.3.3. SYS_CON2

Addr = 0x0A（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	-	-	-	-
6	TMR2SWSYNCBPS	**基础 Timer2 软件同步 bypass 使能位** 0x0：关闭 0x1：打开	RW	0x1
5	IODBSSOFTRST	**IO debouce 模块软复位** 0x0：软复位 0x1：软复位释放	RW	0x1
4	-	-	-	-
3	TMR22IREN	**基本 Timer2 与高级 Timer2 联合完成红外发送功能使能位** 基本 Timer2 作为载波 PWM，高级 Timer2 通道 A 作为调制波 PWM 0x0：关闭 0x1：打开	RW	0x0
2:1	-	-	-	-
0	LVDVCCWKEN	**LVDVCC 唤醒使能位**	RW	0x0

		0x0：关闭 0x1：打开		

Note：SYS_CON2 寄存器中保留位用于特殊测试功能，用户程序不能随意写操作，可能会带来系统风险！

7.3.4. SYS_CON3

Addr = 0x0B（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	EXTSLPCNT	**低功耗 Sleep Mode 流程退出低功耗 LDO 延迟的时间配置** 低功耗模式进入之前必须将系统时钟切换成低速的 64KHz 的 RC，所以延迟时间=n*T64k 0x0：1 个系统周期 0x1：2 个系统周期 0x2：3 个系统周期 0x3：4 个系统周期（推荐配置）	RW	0x0
5:4	CMPUPCNT	**低功耗 Sleep Mode 流程退出低功耗流程中打开程序存储器供电的延迟时间配置** 低功耗模式进入之前必须将系统时钟切换成低速的 64KHz 的 RC，所以延迟时间=n*T64k 0x0：1 个系统周期 0x1：2 个系统周期 0x2：3 个系统周期 0x3：4 个系统周期（推荐配置）	RW	0x0
3:2	OPMLDOCNT	**低功耗 Sleep Mode 流程打开主 LDO 延迟时间配置** 低功耗模式进入之前必须将系统时钟切换成低速的 64KHz 的 RC，所以延迟时间=n*T64k 0x0：1 个系统周期 0x1：2 个系统周期 0x2：3 个系统周期	RW	0x0

		0x3：4 个系统周期（推荐配置）		
1:0	CLSMLDOCNT	**低功耗 Sleep Mode 流程关闭主 LDO 延迟时间配置** 低功耗模式进入之前必须将系统时钟切换成低速的 64KHz 的 RC，所以延迟时间=n*T64k 0x0：1 个系统周期 0x1：2 个系统周期 0x2：3 个系统周期 0x3：4 个系统周期（推荐配置）	RW	0x0

7.3.5. SYS_CON4

Addr = 0x0C（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	P07DBSEN	**P07 输入滤波功能使能位** 0x0：关闭 0x1：打开	RW	0x0
6	P06DBSEN	**P06 输入滤波功能使能位** 0x0：关闭 0x1：打开	RW	0x0
5	P05DBSEN	**P05 输入滤波功能使能位** 0x0：关闭 0x1：打开	RW	0x0
4	P04DBSEN	**P04 输入滤波功能使能位** 0x0：关闭 0x1：打开	RW	0x0
3	P03DBSEN	**P03 输入滤波功能使能位** 0x0：关闭 0x1：打开	RW	0x0
2	P02DBSEN	**P02 输入滤波功能使能位** 0x0：关闭 0x1：打开	RW	0x0
1	P01DBSEN	**P01 输入滤波功能使能位**	RW	0x0

		0x0：关闭 0x1：打开		
0	P00DBSEN	**P00 输入滤波功能使能位** 0x0：关闭 0x1：打开	RW	0x0

7.3.6. SYS_CON5

Addr = 0x0D（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	-	-	-	-
5	P15DBSEN	**P15 输入滤波功能使能位** 0x0：关闭 0x1：打开	RW	0x0
4	P14DBSEN	**P14 输入滤波功能使能位** 0x0：关闭 0x1：打开	RW	0x0
3	P13DBSEN	**P13 输入滤波功能使能位** 0x0：关闭 0x1：打开	RW	0x0
2	P12DBSEN	**P12 输入滤波功能使能位** 0x0：关闭 0x1：打开	RW	0x0
1	P11DBSEN	**P11 输入滤波功能使能位** 0x0：关闭 0x1：打开	RW	0x0
0	P10DBSEN	**P10 输入滤波功能使能位** 0x0：关闭 0x1：打开	RW	0x0

7.3.7. CLK_CON0

Addr = 0x10 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	-	-	-	-
6	CMPDBSSEL	**比较器滤波时钟选择位** 0x0：选择 eflash_clk 0x1：选择 sys_clk	RW	0x0
5:4	P0DBSCLKSEL	**P0 滤波时钟选择位** 0x0：选择 hirc_div_clk 0x1：选择 xoscm 0x2：选择 sys_clk 0x3：选择 rc64k	RW	0x0
3:2	CLKTOIOSEL	**IO 输出时钟源选择位** 0x0：选择 sys_clk 0x1：选择 hirc_div_clk 0x2：选择 lirc 0x3：选择 xoscm	RW	0x0
1:0	SYSCLKSEL	**系统时钟选择位** 0x0：选择 rc64k 0x1：选择 xoscm 0x2：选择 hirc_div_clk 0x3：选择 hirc_clk	RW	0x0

7.3.8. CLK_CON1

Addr = 0x11 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7:6	P1DBSCLKSEL	**P1 滤波时钟选择位** 0x0：选择 hirc_div_clk 0x1：选择 xoscm	RW	0x0

		0x2：选择 sys_clk 0x3：选择 rc64k		
5:3	HIRCCLKDIV	**高速 HRCOSC 时钟源分频设置** 配置比为 n+1 时钟 0x0：不分频 0x1：2 分频 0x2：3 分频 …… 0x6：7 分频 0x7：关闭	RW	0x6
2:0	CLKTOIODIV	**IO 输出时钟源分频设置** 配置比为 n+1 时钟 0x0：不分频 0x1：2 分频 0x2：3 分频 …… 0x6：7 分频 0x7：关闭	RW	0x0

7.3.9. CLK_CON2

Addr = 0x12（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:4	-	-	-	-
3:0	SYSCLKDIV	**系统时钟分频设置** 配置比为 n+1 时钟 0x0：不分频 0x1：2 分频 0x2：3 分频 …… 0xE：15 分频 0xF：关闭	RW	0x0

7.3.10. CLK_CON3

Addr = 0x13（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	TMR2CLKEN	**基本 Timer2 模块时钟使能位** 0x0：关闭时钟 0x1：打开时钟	RW	0x1
6	TMR1CLKEN	**基本 Timer1 模块时钟使能位** 0x0：关闭时钟 0x1：打开时钟	RW	0x1
5	TMR0CLKEN	**基本 Timer0 模块时钟使能位** 0x0：关闭时钟 0x1：打开时钟	RW	0x1
4	CRCCLKEN	**CRC 模块时钟使能位** 0x0：关闭时钟 0x1：打开时钟	RW	0x1
3	-	-	-	-
2	SPI0CLKEN	**SPI0 模块时钟使能位** 0x0：关闭时钟 0x1：打开时钟	RW	0x1
1	UART1CLKEN	**UART1 模块时钟使能位** 0x0：关闭时钟 0x1：打开时钟	RW	0x1
0	UART0CLKEN	**UART0 模块时钟使能位** 0x0：关闭时钟 0x1：打开时钟	RW	0x1

7.3.11. CLK_CON4

Addr = 0x14（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	TESTCLKEN	**测试时钟使能位** 0x0：关闭时钟 0x1：打开时钟	RW	0x0
6	-	-	-	-
5	RAMCLKEN	**片上 SRAM 时钟使能位** 0x0：关闭时钟 0x1：打开时钟	RW	0x1
4	CMPDBSCLKEN	**比较器 debouce 时钟使能位** 0x0：关闭时钟 0x1：打开时钟	RW	0x1
3:2	-	-	-	-
1	STMR2CLKEN	**高级 Timer2 模块时钟使能位** 0x0：关闭时钟 0x1：打开时钟	RW	0x1
0	STMR1CLKEN	**高级 Timer1 模块时钟使能位** 0x0：关闭时钟 0x1：打开时钟	RW	0x1

7.3.12. CLK_CON5

Addr = 0x15（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	-	-	-	-
5:4	TMR2CLKSEL	**基本 Timer2 模块时钟选择位** 0x0：选择 sys_clk 0x1：选择 xoscm 0x2：选择 hirc_div_clk 0x3：选择 rc64k	RW	0x0
3:2	LVDDBSCLKSEL	**LVD 模块滤波时钟源选择位** 0x0：选择 sys_clk	RW	0x0

		0x1：选择 hirc_div_clk 0x2：选择 xoscm 0x3：选择 rc64k		
1	-	-	-	-
0	TCLKEN	**测试时钟 1 使能位** 0x0：关闭时钟 0x1：打开时钟	RW	0x1

7.3.13. CLK_CON6

Addr = 0x16（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	-	-	-	-
5:0	MCLKDIV	**存储器烧写时钟分频设置** 配置比为 n+1 时钟 0x00：不分频 0x01：2 分频 0x02：3 分频 …… 0x3E：63 分频 0x3F：关闭	RW	0x0

7.3.14. AIP_CON0

Addr = 0x30（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	HRCEN	**HRC 时钟使能信号** 0x0：关闭 0x1：打开	RW	0x0
6:0	HRCSC	**HRC 时钟频率细调(step=1%)** 0x00：low	RW	0x48

		…… 0x7F：high		

7.3.15. AIP_CON1

Addr = 0x31 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	-	-	-	-
6:5	HRCSCADD_B2TO1	**HRC 时钟频率细调的高 2 位(step=0.5%)** 0x0：low …… 0x3：high	RW	0x1
4	HRCTESTEN	**HRC 内部模拟电压测试使能信号** 0x0：关闭 0x1：打开	RW	0x0
3	XOSCEN	**晶振的使能信号** 0x0：关闭 0x1：打开	RW	0x0
2	XOSCHY	**输出时钟迟滞窗口选择** 0x0：没有迟滞 0x1：有+-10%的迟滞	RW	0x1
1	HRCSCADD_B0	**HRC 时钟频率细调的低 1 位(step=0.5%)**	RW	0x0
0	HRCSR	**HRC 时钟频率粗调** 0x0：16MHz 0x1：32MHz	RW	0x1

7.3.16. AIP_CON2

Addr = 0x32 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	ADCCMPEN	**ADC 中 CMP 使能信号**	RW	0x0

		0x0：关闭 0x1：打开		
6	ADCCMPTRIMEN	**ADC 中比较器校准功能使能信号** 0x0：关闭 0x1：打开	RW	0x0
5:0	ADCCMPTRIM	**ADC 中比较器校准值** MSB：符号为，低 5 为数值	RW	0x0

7.3.17. AIP_CON3

Addr = 0x33（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	ADCDUMMY	**保留寄存器** <bit 6>VDDACMP 测试是能开关 <bit 7>保留	RW	0x0
5:4	ADCTENSEL	**ADC 测试信号选择** 0x0：VREFP 0x1：保留 0x2：保留 0x3：关闭测试信号	RW	0x3
3:2	ADCSELEXREF	**ADC 外部参考选择信号** 0x0：不选择外部参考 0x1：选择 EXREF0 为参考电压(P10 点电压) 0x2：选择 EXREF1 为参考电压(VCCA) 0x3：非法态 **Note**：选择外部参考时，必须选关闭内部参考！	RW	0x0
1:0	ADCBIASSEL	**ADC 偏置电流选择** 0x0：0.75X 0x1：1X 0x2：1X 0x3：1.25X	RW	0x1

7.3.18. AIP_CON4

Addr = 0x34 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	-	-	-	-
6	ADCPADFORCE	**强行将 ATO 连接到 ADC 输入口** 用于测试	RW	0x0
5	ADCSELTRIMIB	**ADC 校准电流选择** 0x0：1X 0x1：2X	RW	0x0
4	ADCSELINREF	**ADC 中内部参考能使信号** 0x0：关闭 0x1：打开 选择内部参考时，必须先断开外部参考	RW	0x1
3	ADCCMPBSSEL	**ADC 中比较器偏置电流选择** 0x0：1X 0x1：1.25X	RW	0x0
2	ADCVCMSEL	**ADC 共模电压选择** 0x0：0.375 fullscale 0x1：0.5 fullscale	RW	0x1
1	ADCVREFSEL	**ADC 中内部参考电压选择信号** 0x0：reserved 0x1：2.4V(5V 电压供电下)	RW	0x1
0	ADCBIASEN	**ADC 偏置电流能使信号** 0x0：关闭 0x1：打开	RW	0x0

7.3.19. IO_MAP

Addr = 0x28 (XSFR)

Bit(s)	Name	Description	R/W	Reset

7	WKUPIN3	**低功耗模式 wakeup 通道 3 输入引脚选择** 0x0：不选择 P13 0x1：选择 P13	RW	0x0
6	WKUPIN2	**低功耗模式 wakeup 通道 2 输入引脚选择** 0x0：不选择 P12 0x1：选择 P12	RW	0x0
5	WKUPIN1	**低功耗模式 wakeup 通道 1 输入引脚选择** 0x0：不选择 P10 0x1：选择 P10	RW	0x0
4	WKUPIN0	**低功耗模式 wakeup 通道 0 输入引脚选择** 0x0：不选择 P07 0x1：选择 P07	RW	0x0
3:2	CLKTOIOMAP	**IO 输出时钟源功能 pin 脚选择位** 0x0：关闭 IO 输出时钟功能 0x1：选择 P04 输出时钟 0x2：选择 P05 输出时钟 0x3：选择 P12 输出时钟	RW	0x0
1:0	ISPMAP	**烧写/调试 pin 脚选择位** 0x0：关闭烧写/调试功能 0x1：选择 P10【HCK】,P07【HDA】 0x2：选择 P00【HCK】,P01【HDA】 0x3：关闭烧写/调试功能	RW	0x1

7.3.20. IO_MAP1

Addr = 0x2A（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	MEMDVSE	**片上 SRAM 动态电压调整值使能位** **Note**：用户千万不要配置该寄存器，否则会导致芯片不确定行为！！！！	RW	0x0
6:5	MEMDVS[3:2]	**片上 SRAM 动态电压调整值** **Note**：用户千万不要配置该寄存器，否则会导致	RW	0x0

		芯片不确定行为！！！！		
4	RSTBEN	**RSTB 复位功能使能位** 0x0：关闭 RSTB 复位功能 0x1：打开 RSTB 复位功能	RW	0x0
3:2	MPDNCNT	**低功耗 Sleep Mode 流程进入关闭程序存储器供电的延迟时间配置** 低功耗模式进入之前必须将系统时钟切换成低速的 64KHz 的 RC，所以延迟时间=n*T64k。 0x0：1 个系统周期 0x1：2 个系统周期 0x2：3 个系统周期 0x3：4 个系统周期（推荐配置）	RW	0x0
1:0	RSTBSEL	**RSTB 复位引脚选择** 0x0：不选择任何 pin 脚作为复位功能脚 0x1：选择 P00 作为复位引脚 0x2：选择 P05 作为复位引脚 0x3：选择 P00，P05 作为复位引脚，实际 P00 有效复位	RW	0x0

7.3.21. SYS_PND

Addr = 0xBF（SFR）

Bit(s)	Name	Description	R/W	Reset
7:6	-	-	-	-
5	SLPSTACLR	**写 1 清掉系统 sleep 标志位**	RW	0x0
4	SFTRSTCLR	**写 1 清掉系统软复位标志位**	RW	0x0
3:2	-	-	-	-
1	SLPPND	**系统 sleep 标志位**	RW	0x0
0	SFTRSTPND	**系统软复位标志位** 写 1 系统软复位。	RW	0x0

8. 中断系统

8.1. 中断概述

TX8C101x 支持多达 14 个中断源。每个中断源都有独立的中断使能信号，可以通过软件来控制其使能开关。中断控制器有以下特性：

- 从 14 个中断源接收中断
- 每个中断有固定的中断号，中断号越小优先级越高
- 中断延时：5~8 机器周期

8.2. 结构框图

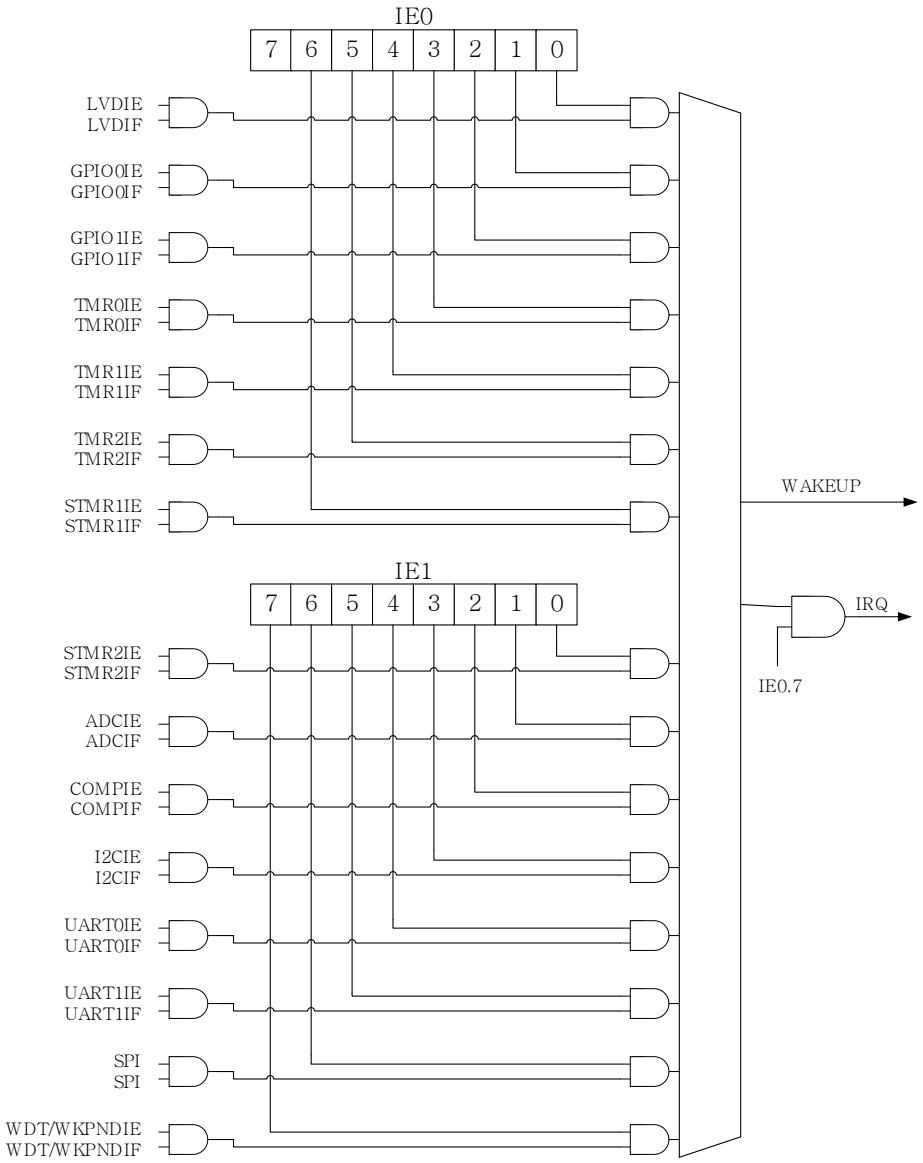


图 8-1 中断结构图

8.3. 中断向量表

中断控制器支持 14 个中断源。当中断发生且中断使能之后，跳转到对应向量地址去执行 LCALL 指令来进入中断服务程序。

表 8-1 中断向量表

中断源	中断等级	中断号	中断地址	说明
LVD	低	0	0003H	低压检测
P0	低	1	0006H	P0 端口中断
P1	低	2	0009H	P1 端口中断
TMR0	低	3	000CH	TMR0 中断
TMR1	低	4	000FH	TMR1 中断
TMR2	低	5	0012H	TMR2 中断
STMR1	低	6	0015H	STMR1 中断
STMR2	低	7	0018H	STMR2 中断
ADC	低	8	001BH	ADC 转换完成中断
COMP	低	9	001EH	模拟比较器中断
-	-	-	0021H	-
UART0	低	11	0024H	UART0 状态中断
UART1	低	12	0027H	UART1 状态中断
SPI0	低	13	002AH	SPI0 中断
WDT/WKUP_PND	低	14	002DH	看门狗中断

8.4. 寄存器列表

表 8-2 中断寄存器列表

Address	Register Name	Description
0xA8 (SFR)	IE0	Interrupt Enable 0 Register
0xA9 (SFR)	IE1	Interrupt Enable 1 Register

0xB8 (SFR)	IP0	Interrupt Priority 0 Register
0xB9 (SFR)	IP1	Interrupt Priority 1 Register

8.5. 寄存器详细说明

8.5.1. IE0

Addr = 0xA8 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	EA	**全局中断使能** 0x0：禁止所有中断 0x1：允许所有未被屏蔽的中断	RW	0x0
6	STMR1	**STMR1 中断使能** 0x0：禁止 STMR1 中断 0x1：允许 STMR1 未被屏蔽的中断	RW	0x0
5	TMR2	**TMR2 中断使能** 0x0：禁止 TMR2 中断 0x1：允许 TMR2 未被屏蔽的中断	RW	0x0
4	TMR1	**TMR1 中断使能** 0x0：禁止 TMR1 中断 0x1：允许 TMR1 未被屏蔽的中断	RW	0x0
3	TMR0	**TMR0 中断使能** 0x0：禁止 TMR0 中断 0x1：允许 TMR0 未被屏蔽的中断	RW	0x0
2	GPIO1	**GPIO1 中断使能** 0x0：禁止 GPIO1 中断 0x1：允许 GPIO1 未被屏蔽的中断	RW	0x0
1	GPIO0	**GPIO0 中断使能** 0x0：禁止 GPIO0 中断 0x1：允许 GPIO0 未被屏蔽的中断	RW	0x0

0	LVD	**LVD 中断使能** 0x0：禁止 LVD 中断 0x1：允许 LVD 未被屏蔽的中断	RW	0x0

8.5.2. IE1

Addr = 0xA9（SFR）

Bit(s)	Name	Description	R/W	Reset
7	WDT/WKUP	**WDT/WKUP 中断使能** 0x0：禁止 WDT/WKUP 中断 0x1：允许 WDT/WKUP 未被屏蔽的中断	RW	0x0
6	SPI	**SPI 中断使能** 0x0：禁止 SPI 中断 0x1：允许 SPI 未被屏蔽的中断	RW	0x0
5	UART1	**UART1 中断使能** 0x0：禁止 UART1 中断 0x1：允许 UART1 未被屏蔽的中断	RW	0x0
4	UART0	**UART0 中断使能** 0x0：禁止 UART0 中断 0x1：允许 UART0 未被屏蔽的中断	RW	0x0
3	-	-	-	-
2	COMP	**比较器中断使能** 0x0：禁止比较器中断 0x1：允许比较器未被屏蔽的中断	RW	0x0
1	ADC	**ADC 中断使能** 0x0：禁止 ADC 中断 0x1：允许 ADC 未被屏蔽的中断	RW	0x0
0	STMR2	**STMR2 中断使能** 0x0：禁止 STMR2 中断 0x1：允许 STMR2 未被屏蔽的中断	RW	0x0

8.5.3. IP0

Addr = 0xB8 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	-	-	-	-
6	STMR1	**STMR1 中断优先级** 0x0：优先等级为 0 0x1：优先等级为 1	RW	0x0
5	TMR2	**TMR2 中断优先级** 0x0：优先等级为 0 0x1：优先等级为 1	RW	0x0
4	TMR1	**TMR1 中断优先级** 0x0：优先等级为 0 0x1：优先等级为 1	RW	0x0
3	TMR0	**TMR0 中断优先级** 0x0：优先等级为 0 0x1：优先等级为 1	RW	0x0
2	GPIO1	**GPIO1 中断优先级** 0x0：优先等级为 0 0x1：优先等级为 1	RW	0x0
1	GPIO0	**GPIO0 中断优先级** 0x0：优先等级为 0 0x1：优先等级为 1	RW	0x0
0	LVD	**LVD 中断优先级** 0x0：优先等级为 0 0x1：优先等级为 1	RW	0x0

8.5.4. IP1

Addr = 0xB9 (SFR)

Bit(s)	Name	Description	R/W	Reset

7	WDT/WKUP	**WDT/WKUP 中断优先级** 0x0：优先等级为 0 0x1：优先等级为 1	RW	0x0
6	SPI	**SPI 中断优先级** 0x0：优先等级为 0 0x1：优先等级为 1	RW	0x0
5	UART1	**UART1 中断优先级** 0x0：优先等级为 0 0x1：优先等级为 1	RW	0x0
4	UART0	**UART0 中断优先级** 0x0：优先等级为 0 0x1：优先等级为 1	RW	0x0
3	-	-	-	-
2	COMP	**比较器中断优先级** 0x0：优先等级为 0 0x1：优先等级为 1	RW	0x0
1	ADC	**ADC 中断优先级** 0x0：优先等级为 0 0x1：优先等级为 1	RW	0x0
0	STMR2	**STMR2 中断优先级** 0x0：优先等级为 0 0x1：优先等级为 1	RW	0x0

8.6. 中断优先级及中断嵌套

芯片规定 2 个中断优先级，可实现 2 级中断嵌套。当一个中断已经响应，若有高级别中断发出请求，后者可以中断前者，实现中断嵌套。每个中断可配置优先等级 0-1。中断优先级越大，中断的优先级越高。同一级别没有嵌套，采用时间优先的原则，先到先执行。

9. I/O 端口

9.1. 结构框图

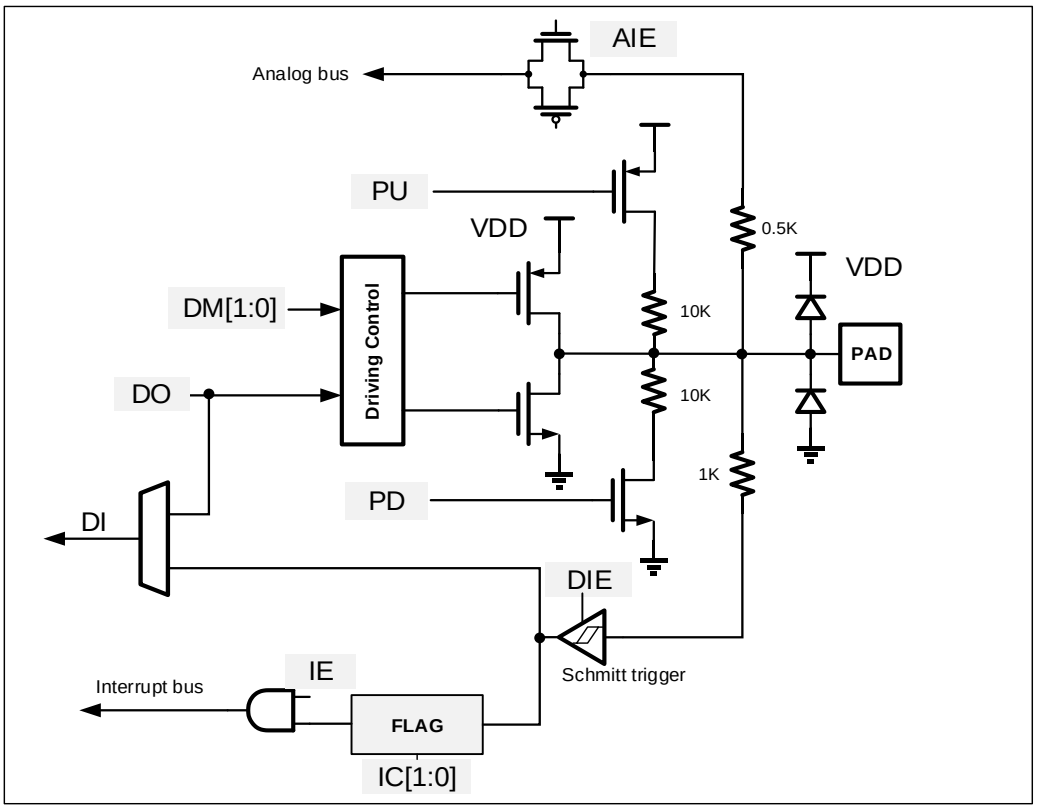


图 9-1 IO 结构图

9.2. 寄存器列表

表 9-1 GPIO 寄存器列表

Address	Register Name	Description
0x80 (SFR)	P0	P0 data register
0x50 (XSFR)	P0_PU	P0 pull-up enable register
0x51 (XSFR)	P0_PD	P0 pull-down enable register
0x52 (XSFR)	P0_MD0	P0 mode register 0

0x53 (XSFR)	P0_MD1	P0 mode register 1
0x54 (XSFR)	P0_AF0	P0 alternal function config register 0
0x55 (XSFR)	P0_AF1	P0 alternal function config register 1
0x56 (XSFR)	P0_TRG0	P0 interrupt trigger config register 0
0x57 (XSFR)	P0_TRG1	P0 interrupt trigger config register 1
0x58 (XSFR)	P0_PND	P0 interrupt pending register
0x59 (XSFR)	P0_IMK	P0 interrupt mask register
0x5A (XSFR)	P0_AIOEN	P0 analog function enable register
0x5B (XSFR)	P0_DRV	P0 driving current config register
0x5C (XSFR)	P0_OD	P0 open-drain enable register
0x90 (SFR)	P1	P1 data register
0x60 (XSFR)	P1_PU	P1 pull-up enable register
0x61 (XSFR)	P1_PD	P1 pull-down enable register
0x62 (XSFR)	P1_MD0	P1 mode register 0
0x63 (XSFR)	P1_MD1	P1 mode register 1
0x64 (XSFR)	P1_AF0	P1 alternal function config register 0
0x65 (XSFR)	P1_AF1	P1 alternal function config register 1
0x66 (XSFR)	P1_TRG0	P1 interrupt trigger config register 0
0x67 (XSFR)	P1_TRG1	P1 interrupt trigger config register 1
0x68 (XSFR)	P1_PND	P1 interrupt pending register
0x69 (XSFR)	P1_IMK	P1 interrupt mask register
0x6A (XSFR)	P1_AIOEN	P1 analog function enable register
0x6B (XSFR)	P1_DRV	P1 driving current config register

0x6C（XSFR）	P1_OD	P1 open-drain enable register

9.3. 寄存器详细说明

9.3.1. P0

Addr = 0x80（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	P0	**P0 数据寄存器**	RW	0x00

9.3.2. P0_PU

Addr = 0x50（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	P07PU	**P07 上拉电阻（30K）使能控制位** 0x0：关闭 0x1：打开	RW	0x0
6	P06PU	**P06 上拉电阻（30K）使能控制位** 0x0：关闭 0x1：打开	RW	0x0
5	P05PU	**P05 上拉电阻（30K）使能控制位** 0x0：关闭 0x1：打开	RW	0x0
4	P04PU	**P04 上拉电阻（30K）使能控制位** 0x0：关闭 0x1：打开	RW	0x0
3	P03PU	**P03 上拉电阻（30K）使能控制位** 0x0：关闭 0x1：打开	RW	0x0
2	P02PU	**P02 上拉电阻（30K）使能控制位**	RW	0x0

		0x0：关闭 0x1：打开		
1	P01PU	**P01 上拉电阻（30K）使能控制位** 0x0：关闭 0x1：打开	RW	0x0
0	P00PU	**P00 上拉电阻（30K）使能控制位** 0x0：关闭 0x1：打开	RW	0x0

9.3.3. P0_PD

Addr = 0x51（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	P07PD	**P07 下拉电阻（30K）使能控制位** 0x0：关闭 0x1：打开	RW	0x0
6	P06PD	**P06 下拉电阻（30K）使能控制位** 0x0：关闭 0x1：打开	RW	0x0
5	P05PD	**P05 下拉电阻（30K）使能控制位** 0x0：关闭 0x1：打开	RW	0x0
4	P04PD	**P04 下拉电阻（30K）使能控制位** 0x0：关闭 0x1：打开	RW	0x0
3	P03PD	**P03 下拉电阻（30K）使能控制位** 0x0：关闭 0x1：打开	RW	0x0
2	P02PD	**P02 下拉电阻（30K）使能控制位** 0x0：关闭 0x1：打开	RW	0x0
1	P01PD	**P01 下拉电阻（30K）使能控制位**	RW	0x0

		0x0：关闭 0x1：打开		
0	P00PD	**P00 下拉电阻（30K）使能控制位** 0x0：关闭 0x1：打开	RW	0x0

9.3.4. P0_MD0

Addr = 0x52（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	P03MD	**P03 模式配置位** 0x0：输入模式 0x1：输出模式 0x2：多功能 IO 模式 0x3：模拟 IO 模式	RW	0x0
5:4	P02MD	**P02 模式配置位** 0x0：输入模式 0x1：输出模式 0x2：多功能 IO 模式 0x3：模拟 IO 模式	RW	0x0
3:2	P01MD	**P01 模式配置位** 0x0：输入模式 0x1：输出模式 0x2：多功能 IO 模式 0x3：模拟 IO 模式 **Note：** 如果烧录口为 P01，则需要对应将 IO_MAP[0:1]，即 ISPMAP 设置为 0x03，才能切换为 GPIO 功能	RW	0x0
1:0	P00MD	**P00 模式配置位** 0x0：输入模式 0x1：输出模式 0x2：多功能 IO 模式	RW	0x0

		0x3：模拟 IO 模式 Note：如果烧录口为 P00，则需要对应将 IO_MAP[0:1]，即 ISPMAP 设置为 0x03，才能切换为 GPIO 功能		

9.3.5. P0_MD1

Addr = 0x53（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	P07MD	**P07 模式配置位** 0x0：输入模式 0x1：输出模式 0x2：多功能 IO 模式 0x3：模拟 IO 模式 Note：如果烧录口为 P07，则需要对应将 IO_MAP[0:1]，即 ISPMAP 设置为 0x03，才能切换为 GPIO 功能	RW	0x0
5:4	P06MD	**P06 模式配置位** 0x0：输入模式 0x1：输出模式 0x2：多功能 IO 模式 0x3：模拟 IO 模式	RW	0x0
3:2	P05MD	**P05 模式配置位** 0x0：输入模式 0x1：输出模式 0x2：多功能 IO 模式 0x3：模拟 IO 模式	RW	0x0
1:0	P04MD	**P04 模式配置位** 0x0：输入模式 0x1：输出模式 0x2：多功能 IO 模式 0x3：模拟 IO 模式	RW	0x0

9.3.6. P0_AF0

Addr = 0x54 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7:6	P03AF	**P03 多功能模式下，功能配置位** 0x0: STMR1_CHA function 0x1: SPI0_DIO function 0x2: STMR2_CHB function 0x3: PORT_WKUP_IN3 function	RW	0x0
5:4	P02AF	**P02 多功能模式下，功能配置位** 0x0: STMR1_CHB function 0x1: SPI0_CLK function 0x2: 保留 0x3: PORT_WKUP_IN2 function	RW	0x0
3:2	P01AF	**P01 多功能模式下，功能配置位** 0x0: TMR2_PWM function 0x1: UART1_TX function 0x2: 保留 0x3: PORT_WKUP_IN1 function	RW	0x0
1:0	P00AF	**P00 多功能模式下，功能配置位** 0x0: TMR2_PWM function 0x1: UART1_RX function 0x2: 保留 0x3: PORT_WKUP_IN0 function	RW	0x0

9.3.7. P0_AF1

Addr = 0x55 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7:6	P07AF	**P07 多功能模式下，功能配置位**	RW	0x0

		0x0：CMP0_DIG_OUT function 0x1：I2C_SDA function 0x2：TMR2_PWM function 0x3：UART0_RX function		
5:4	P06AF	**P06 多功能模式下，功能配置位** 0x0：STMR1_CHB function 0x1：TMR0_PWM function 0x2：STMR2_CHA function 0x3：PORT_WKUP_IN2 function	RW	0x0
3:2	P05AF	**P05 多功能模式下，功能配置位** 0x0：STMR2_CHA function 0x1：TMR0_CAP function 0x2：保留 0x3：PORT_WKUP_IN1 function	RW	0x0
1:0	P04AF	**P04 多功能模式下，功能配置位** 0x0：STMR2_CHB function 0x1：CMP0_DIG_OUT function 0x2：保留 0x3：PORT_WKUP_IN0 function	RW	0x0

9.3.8. P0_TRG0

Addr = 0x56（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	P03TRG	**P03 中断触发源配置位** 0x0：上升沿和下降沿触发 0x1：下降沿触发 0x2：上升沿触发 0x3：上升沿和下降沿触发	RW	0x0
5:4	P02TRG	**P02 中断触发源配置位** 0x0：上升沿和下降沿触发 0x1：下降沿触发	RW	0x0

		0x2：上升沿触发 0x3：上升沿和下降沿触发		
3:2	P01TRG	**P01 中断触发源配置位** 0x0：上升沿和下降沿触发 0x1：下降沿触发 0x2：上升沿触发 0x3：上升沿和下降沿触发	RW	0x0
1:0	P00TRG	**P00 中断触发源配置位** 0x0：上升沿和下降沿触发 0x1：下降沿触发 0x2：上升沿触发 0x3：上升沿和下降沿触发	RW	0x0

9.3.9. P0_TRG1

Addr = 0x57（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	P07TRG	**P07 中断触发源配置位** 0x0：上升沿和下降沿触发 0x1：下降沿触发 0x2：上升沿触发 0x3：上升沿和下降沿触发	RW	0x0
5:4	P06TRG	**P06 中断触发源配置位** 0x0：上升沿和下降沿触发 0x1：下降沿触发 0x2：上升沿触发 0x3：上升沿和下降沿触发	RW	0x0
3:2	P05TRG	**P05 中断触发源配置位** 0x0：上升沿和下降沿触发 0x1：下降沿触发 0x2：上升沿触发	RW	0x0

		0x3：上升沿和下降沿触发		
1:0	P04TRG	**P04 中断触发源配置位** 0x0：上升沿和下降沿触发 0x1：下降沿触发 0x2：上升沿触发 0x3：上升沿和下降沿触发	RW	0x0

9.3.10. P0_PND

Addr = 0x58（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	P07PND	**P07 中断标志位** 0x0：无触发中断 0x1：触发中断	RW	0x0
6	P06PND	**P06 中断标志位** 0x0：无触发中断 0x1：触发中断	RW	0x0
5	P05PND	**P05 中断标志位** 0x0：无触发中断 0x1：触发中断	RW	0x0
4	P04PND	**P04 中断标志位** 0x0：无触发中断 0x1：触发中断	RW	0x0
3	P03PND	**P03 中断标志位** 0x0：无触发中断 0x1：触发中断	RW	0x0
2	P02PND	**P02 中断标志位** 0x0：无触发中断 0x1：触发中断	RW	0x0
1	P01PND	**P01 中断标志位** 0x0：无触发中断	RW	0x0

		0x1：触发中断		
0	P00PND	**P00 中断标志位** 0x0：无触发中断 0x1：触发中断	RW	0x0

Note： CPU write P0_PND operation, and clear all pending! ! !

9.3.11. P0_IMK

Addr = 0x59（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	P07IMK	**P07 中断屏蔽位** 0x0：关闭中断 0x1：打开中断	RW	0x0
6	P06IMK	**P06 中断屏蔽位** 0x0：关闭中断 0x1：打开中断	RW	0x0
5	P05IMK	**P05 中断屏蔽位** 0x0：关闭中断 0x1：打开中断	RW	0x0
4	P04IMK	**P04 中断屏蔽位** 0x0：关闭中断 0x1：打开中断	RW	0x0
3	P03IMK	**P03 中断屏蔽位** 0x0：关闭中断 0x1：打开中断	RW	0x0
2	P02IMK	**P02 中断屏蔽位** 0x0：关闭中断 0x1：打开中断	RW	0x0
1	P01IMK	**P01 中断屏蔽位** 0x0：关闭中断 0x1：打开中断	RW	0x0

0	P00IMK	**P00 中断屏蔽位** 0x0：关闭中断 0x1：打开中断	RW	0x0

9.3.12. P0_AIOEN

Addr = 0x5A（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	P07AIOEN	**P07 模拟使能位** 0x0：关闭 0x1：打开	RW	0x0
6	P06AIOEN	**P06 模拟使能位** 0x0：关闭 0x1：打开	RW	0x0
5	P05AIOEN	**P05 模拟使能位** 0x0：关闭 0x1：打开	RW	0x0
4	P04AIOEN	**P04 模拟使能位** 0x0：关闭 0x1：打开	RW	0x0
3	P03AIOEN	**P03 模拟使能位** 0x0：关闭 0x1：打开	RW	0x0
2	P02AIOEN	**P02 模拟使能位** 0x0：关闭 0x1：打开	RW	0x0
1	P01AIOEN	**P01 模拟使能位** 0x0：关闭 0x1：打开	RW	0x0
0	P00AIOEN	**P00 模拟使能位** 0x0：关闭 0x1：打开	RW	0x0

9.3.13. P0_DRV

Addr = 0x5B（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	P07DRV	**P07 电流驱动能力配置** 0x0：10mA 0x1：50mA	RW	0x0
6	P06DRV	**P06 电流驱动能力配置** 0x0：10mA 0x1：50mA	RW	0x0
5	P05DRV	**P05 电流驱动能力配置** 0x0：10mA 0x1：50mA	RW	0x0
4	P04DRV	**P04 电流驱动能力配置** 0x0：10mA 0x1：50mA	RW	0x0
3	P03DRV	**P03 电流驱动能力配置** 0x0：10mA 0x1：50mA	RW	0x0
2	P02DRV	**P02 电流驱动能力配置** 0x0：10mA 0x1：50mA	RW	0x0
1	P01DRV	**P01 电流驱动能力配置** 0x0：10mA 0x1：50mA	RW	0x0
0	P00DRV	**P00 电流驱动能力配置** 0x0：10mA 0x1：50mA	RW	0x0

9.3.14. P0_OD

Addr = 0x5C（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	P07OD	P07 open-drain 使能位 0x0：关闭 0x1：打开	RW	0x0
6	P06OD	P06 open-drain 使能位 0x0：关闭 0x1：打开	RW	0x0
5	P05OD	P05 open-drain 使能位 0x0：关闭 0x1：打开	RW	0x0
4	P04OD	P04 open-drain 使能位 0x0：关闭 0x1：打开	RW	0x0
3	P03OD	P03 open-drain 使能位 0x0：关闭 0x1：打开	RW	0x0
2	P02OD	P02 open-drain 使能位 0x0：关闭 0x1：打开	RW	0x0
1	P01OD	P01 open-drain 使能位 0x0：关闭 0x1：打开	RW	0x0
0	P00OD	P00 open-drain 使能位 0x0：关闭 0x1：打开	RW	0x0

9.3.15. P1

Addr = 0x90（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	P1	P0 数据寄存器	RW	0x1

9.3.16. P1_PU

Addr = 0x60 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7:6	-	-	-	-
5	P15PU	P15 上拉电阻（30K）使能位 0x0：关闭 0x1：打开	RW	0x0
4	P14PU	P14 上拉电阻（30K）使能位 0x0：关闭 0x1：打开	RW	0x0
3	P13PU	P13 上拉电阻（30K）使能位 0x0：关闭 0x1：打开	RW	0x0
2	P12PU	P12 上拉电阻（30K）使能位 0x0：关闭 0x1：打开	RW	0x0
1	P11PU	P11 上拉电阻（30K）使能位 0x0：关闭 0x1：打开	RW	0x0
0	P10PU	P10 上拉电阻（30K）使能位 0x0：关闭 0x1：打开	RW	0x0

9.3.17. P1_PD

Addr = 0x61 (XSFR)

Bit(s)	Name	Description	R/W	Reset

7:6	-	-	-	-
5	P15PD	**P15 下拉电阻（30K）使能位** 0x0：关闭 0x1：打开	RW	0x0
4	P14PD	**P14 下拉电阻（30K）使能位** 0x0：关闭 0x1：打开	RW	0x0
3	P13PD	**P13 下拉电阻（30K）使能位** 0x0：关闭 0x1：打开	RW	0x0
2	P12PD	**P12 下拉电阻（30K）使能位** 0x0：关闭 0x1：打开	RW	0x0
1	P11PD	**P11 下拉电阻（30K）使能位** 0x0：关闭 0x1：打开	RW	0x0
0	P10PD	**P10 下拉电阻（30K）使能位** 0x0：关闭 0x1：打开	RW	0x0

9.3.18. P1_MD0

Addr = 0x62（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	P13MD	**P13 模式配置位** 0x0：输入模式 0x1：输出模式 0x2：多功能 IO 模式 0x3：模拟 IO 模式	RW	0x0
5:4	P12MD	**P12 模式配置位** 0x0：输入模式 0x1：输出模式	RW	0x0

		0x2：多功能 IO 模式 0x3：模拟 IO 模式		
3:2	P11_MD	**P11 模式配置位** 0x0：输入模式 0x1：输出模式 0x2：多功能 IO 模式 0x3：模拟 IO 模式	RW	0x0
1:0	P10_MD	**P10 模式配置位** 0x0：输入模式 0x1：输出模式 0x2：多功能 IO 模式 0x3：模拟 IO 模式 **Note：**如果烧录口为 P10，则需要对应将 IO_MAP[0:1]，即 ISPMAP 设置为 0x03，才能切换为 GPIO 功能	RW	0x0

9.3.19. P1_MD1

Addr = 0x63（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:4	-	-	-	-
3:2	P15MD	**P15 模式配置位** 0x0：输入模式 0x1：输出模式 0x2：多功能 IO 模式 0x3：模拟 IO 模式	RW	0x0
1:0	P14MD	**P14 模式配置位** 0x0：输入模式 0x1：输出模式 0x2：多功能 IO 模式 0x3：模拟 IO 模式	RW	0x0

9.3.20. P1_AF0

Addr = 0x64 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7:6	P13AF	**P13 多功能模式下，功能配置位** 0x0: STMR2_CHA function 0x1: STMR1_CHA function 0x2: TMR2_PWM function 0x3: UART0_TX function	RW	0x0
5:4	P12AF	**P12 多功能模式下，功能配置位** 0x0: STMR2_CHB function 0x1: STMR1_CHB function 0x2: TMR1_PWM 0x3: UART0_RX function	RW	0x0
3:2	P11AF	**P11 多功能模式下，功能配置位** 0x0: CMP1_DIG_OUT function 0x1: TMR2_CAP function 0x2: 保留 0x3: PORT_WKUP_IN3 function	RW	0x0
1:0	P10AF	**P10 多功能模式下，功能配置位** 0x0: STMR2_CHA function 0x1: I2C0_SCK function 0x2: TMR2_PWM 0x3: UART0_TX function	RW	0x0

9.3.21. P1_AF1

Addr = 0x65 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7:4	-	-	-	-
3:2	P15AF	**P15 多功能模式下，功能配置位** 0x0: CMP0_DIG_OUT function	RW	0x0

		0x1：TMR1_CAP function 0x2：保留 0x3：PORT_WKUP_IN1 function		
1:0	P14AF	**P14 多功能模式下，功能配置位** 0x0：CMP1_DIG_OUT function 0x1：TMR0_CAP function 0x2：保留 0x3：PORT_WKUP_IN0 function	RW	0x0

9.3.22. P1_TRG0

Addr = 0x66（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	P13TRG	**P13 中断触发源配置位** 0x0：上升沿和下降沿触发 0x1：下降沿触发 0x2：上升沿触发 0x3：上升沿和下降沿触发	RW	0x0
5:4	P12TRG	**P12 中断触发源配置位** 0x0：上升沿和下降沿触发 0x1：下降沿触发 0x2：上升沿触发 0x3：上升沿和下降沿触发	RW	0x0
3:2	P11TRG	**P11 中断触发源配置位** 0x0：上升沿和下降沿触发 0x1：下降沿触发 0x2：上升沿触发 0x3：上升沿和下降沿触发	RW	0x0
1:0	P10TRG	**P10 中断触发源配置位** 0x0：上升沿和下降沿触发 0x1：下降沿触发 0x2：上升沿触发	RW	0x0

		0x3：上升沿和下降沿触发		

9.3.23. P1_TRG1

Addr = 0x67（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:4	-	-	-	-
3:2	P15TRG	**P15 中断触发源配置位** 0x0：上升沿和下降沿触发 0x1：下降沿触发 0x2：上升沿触发 0x3：上升沿和下降沿触发	RW	0x0
1:0	P14TRG	**P14 中断触发源配置位** 0x0：上升沿和下降沿触发 0x1：下降沿触发 0x2：上升沿触发 0x3：上升沿和下降沿触发	RW	0x0

9.3.24. P1_PND

Addr = 0x68（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	-	-	-	-
5	P15PND	**P15 中断标志位** 0x0：无触发中断 0x1：触发中断	RW	0x0
4	P14PND	**P14 中断标志位** 0x0：无触发中断 0x1：触发中断	RW	0x0
3	P13PND	**P13 中断标志位**	RW	0x0

		0x0：无触发中断 0x1：触发中断		
2	P12PND	**P12 中断标志位** 0x0：无触发中断 0x1：触发中断	RW	0x0
1	P11PND	**P11 中断标志位** 0x0：无触发中断 0x1：触发中断	RW	0x0
0	P10PND	**P10 中断标志位** 0x0：无触发中断 0x1：触发中断	RW	0x0

Note： CPU write P1_PND operation, and clear all pending! ! !

9.3.25. P1_IMK

Addr = 0x69 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7:6	-	-	-	-
5	P15IMK	**P15 中断屏蔽位** 0x0：关闭中断 0x1：打开中断	RW	0x0
4	P14IMK	**P14 中断屏蔽位** 0x0：关闭中断 0x1：打开中断	RW	0x0
3	P13IMK	**P13 中断屏蔽位** 0x0：关闭中断 0x1：打开中断	RW	0x0
2	P12IMK	**P12 中断屏蔽位** 0x0：关闭中断 0x1：打开中断	RW	0x0
1	P11IMK	**P11 中断屏蔽位**	RW	0x0

		0x0：关闭中断 0x1：打开中断		
0	P10IMK	**P10 中断屏蔽位** 0x0：关闭中断 0x1：打开中断	RW	0x0

9.3.26. P1_AIOEN

Addr = 0x6A（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	-	-	-	-
5	P15AIOEN	**P15 模拟使能位** 0x0：关闭 0x1：打开	RW	0x0
4	P14AIOEN	**P14 模拟使能位** 0x0：关闭 0x1：打开	RW	0x0
3	P13AIOEN	**P13 模拟使能位** 0x0：关闭 0x1：打开	RW	0x0
2	P12AIOEN	**P12 模拟使能位** 0x0：关闭 0x1：打开	RW	0x0
1	P11AIOEN	**P11 模拟使能位** 0x0：关闭 0x1：打开	RW	0x0
0	P10AIOEN	**P10 模拟使能位** 0x0：关闭 0x1：打开	RW	0x0

9.3.27. P1_DRV

Addr = 0x6B（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	-	-	-	-
5	P15DRV	**P15 电流驱动能力配置** 0x0：10mA 0x1：50mA	RW	0x0
4	P14DRV	**P14 电流驱动能力配置** 0x0：10mA 0x1：50mA	RW	0x0
3	P13DRV	**P13 电流驱动能力配置** 0x0：10mA 0x1：50mA	RW	0x0
2	P12DRV	**P12 电流驱动能力配置** 0x0：10mA 0x1：50mA	RW	0x0
1	P11DRV	**P11 电流驱动能力配置** 0x0：10mA 0x1：50mA	RW	0x0
0	P10DRV	**P10 电流驱动能力配置** 0x0：10mA 0x1：50mA	RW	0x0

9.3.28. P1_OD

Addr = 0x6C（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	-	-	-	-
5	P15OD	**P15 open-drain 使能位** 0x0：关闭	RW	0x0

		0x1：打开		
4	P14OD	P14 open-drain 使能位 0x0：关闭 0x1：打开	RW	0x0
3	P13OD	P13 open-drain 使能位 0x0：关闭 0x1：打开	RW	0x0
2	P12OD	P12 open-drain 使能位 0x0：关闭 0x1：打开	RW	0x0
1	P11OD	P11 open-drain 使能位 0x0：关闭 0x1：打开	RW	0x0
0	P10OD	P10 open-drain 使能位 0x0：关闭 0x1：打开	RW	0x0

10. SPI 模块

10.1. 功能概述

SPI模块的功能特点如下：

- 支持两线模式
- 支持主从半双工收发
- 极性相位可编程的串行时钟
- 带MCU中断的传输结束标志
- 主模式支持高达 8Mbps的通讯速率（F_{osc}=32MHz）

10.2. 模块框图

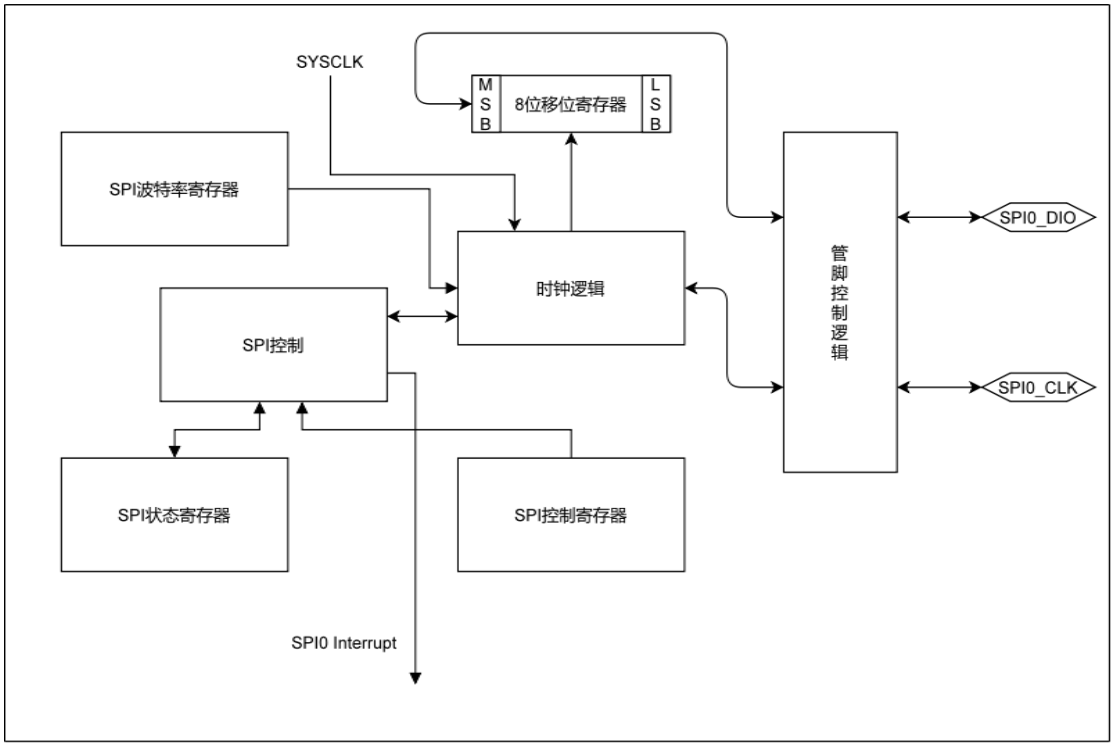


图 10.2.1 SPI 模块框图

10.3. 寄存器列表

表 10-1 SPI 寄存器列表

Address	Register Name	Description
0xF1 (SFR)	SPI0_CON	SPI0 control register
0xF2 (SFR)	SPI0_BAUD	SPI0 baud rate register
0xF3 (SFR)	SPI0_DATA	SPI0 data register
0xF4 (SFR)	SPI0_STA	SPI0 status register

10.4. 寄存器详细说明

10.4.1. SPI0_CON

Addr = 0xF1 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	-	-	-	-
6	SPISM	**主从机控制位** 0x0：主机 0x1：从机	RW	0x0
5	SPIRXTX	**发送接收控制位** 0x0：发送数据 0x1：接收数据	RW	0x0
4	-	-	-	-
3	SPIINTEN	**SPI 中断使能位** 0x0：关闭 0x1：打开	RW	0x0
2	SPISMPSEl	**采样模式选择位** 0x0：从第二个 CLK 时钟边沿开始采样 0x1：从第一个 CLK 时钟边沿开始采样	RW	0x0
1	SPIIDST	**时钟线空闲状态选择位** 0x0：CLK 空闲为低电平 0x1：CLK 空闲为高电平	RW	0x0
0	SPIEN	**SPI 使能位** 0x0：关闭 0x1：打开	RW	0x0

10.4.2. SPI0_BAUD

Addr = 0xF2 (SFR)

Bit(s)	Name	Description	R/W	Reset

7:0	BAUD	**波特率控制寄存器** 计算公式：波特率 = clk/(2*(BAUD+1))	WO	0x0

10.4.3. SPI0_DATA

Addr = 0xF3 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	DATA	**数据寄存器** 使能后将数据写入DATA触发发送,读DATA则读出接收到的数据	RW	-

10.4.4. SPI0_STA

Addr = 0xF4 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:2	-	-	-	-
1	SPIINT	**SPI 中断标志** 写 1 清零	RO	0x0
0	SPIPEND	**SPI 状态标志位** 0x0：正在发送或正在接收 0x1：空闲	RO	0x1

10.5. 使用流程说明

1. 主机 TX：配置 SPI_ENABLE 使能位，SPI_RX_TX 配 0 表示发送,将要发送的数据写入 DATA 触发发送。
2. 主机 RX：配置 SPI_ENABLE 使能位，SPI_RX_TX 配 1 表示接收。写入任意数据到 DATA 触发接收，接收完成（SPI_PENGING=1）读 DATA 读出数据。

3. 从机 TX：配置 SPI_ENABLE 使能位，SPI_SM 配 1 表示从机模式，SPI_RX_TX 配 0 表示发送。将要发送的数据写入 DATA 触发 SPI 等待主机时钟。
4. 从机 RX：配置 SPI_ENABLE 使能位，SPI_SM 配 1 表示从机模式，SPI_RX_TX 配 1 表示发送。写入任意数据到 DATA 触发 SPI 等待主机时钟。

11. UART0/1 模块

11.1. 功能概述

UART 模块功能特点：

- 支持半双工
- 支持发送 9bit 数据
- 支持软件奇偶校验

11.2. 模块框图

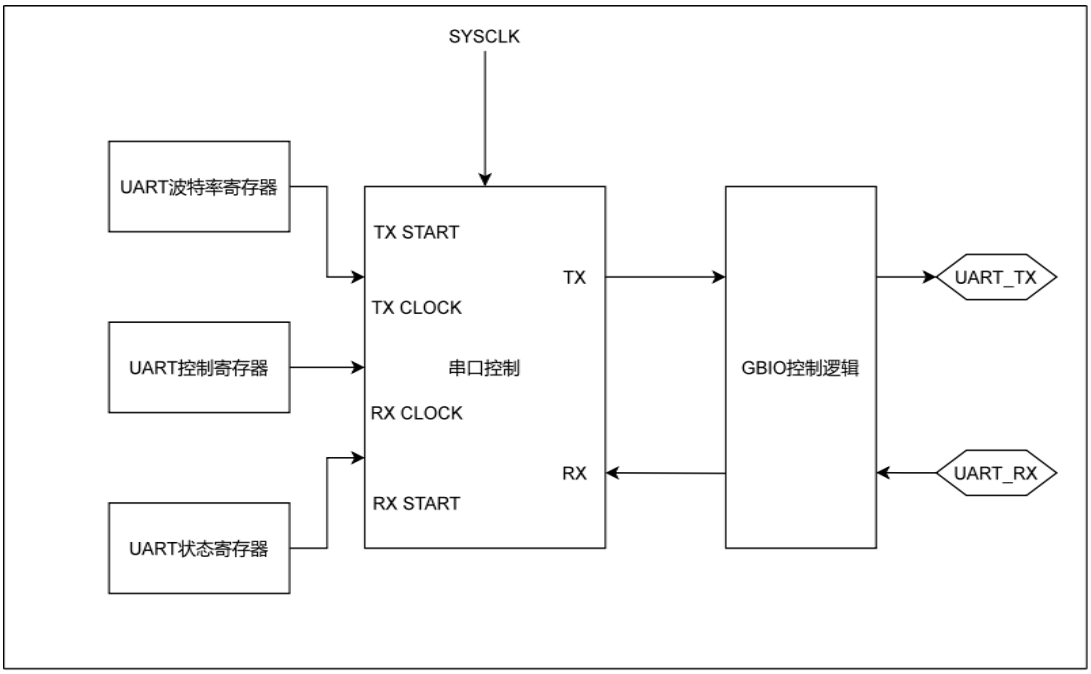


图 11.2.1 UART 模块框图

11.3. 寄存器列表

表 11-1 UART 寄存器列表

Address	Register Name	Description
0xF6 (SFR)	UART0_CON	UART0 control register
0xF7 (SFR)	UART0_STA	UART0 status register
0xF8 (SFR)	UART0_BAUD0	The low eight bits of the UART0 baud rate register
0xF9 (SFR)	UART0_BAUD1	The high eight bits of the UART0 baud rate register
0xFA (SFR)	UART0_DATA	UART0 data register
0xFB (SFR)	UART1_CON	UART1 control register
0xFC (SFR)	UART1_STA	UART1 status register
0xFD (SFR)	UART1_BAUD0	The low eight bits of the UART1 baud rate register
0xFE (SFR)	UART1_BAUD1	The high eight bits of the UART1 baud rate register
0xFF (SFR)	UART1_DATA	UART1 data register

11.4. 寄存器详细说明

11.4.1. UART0_CON

Addr = 0xF6 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	STOPBIT	**停止位控制位** 0x0：不发送停止位 0x1：发送 1bit 停止位	RW	0x0

6	NINTHBIT	**将需要发送的第 9bit 数据写入该寄存器**	RW	0x0
5	BIT9EN	**发送 9bit 数据使能位** 0x0：一次发送 8bit 数据 0x1：一次发送 9bit 数据	RW	0x0
4	UARTEN	**UART 使能位** 0x0：关闭 0x1：打开	RW	0x0
3	TXINV	**TX 电平取反控制位** 0x0：不取反 0x1：取反	RW	0x0
2	RXINV	**RX 电平取反控制位** 0x0：不取反 0x1：取反	RW	0x0
1	TXRXSEL	**TX/RX 选择位** 0x0：TX 模式 0x1：RX 模式	RW	0x0
0	UARTIE	**RX 中断使能位** 0x0：关闭 0x1：打开	RW	0x0

11.4.2. UART0_STA

Addr = 0xF7 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	RXBIT9	**接收的第 9bit,读该位读出第 9bit 数据**	RO	0x0
6	FERR	**接收到的奇偶校验位数据** 奇偶校验由软件进行(接收 8bit 数据加奇偶校验位时不使能 BIT9_EN 的情况下，在此位读出奇偶校验位)	RO	0x0
5	RXDONE	**RX 状态标志位** 该位为 1 表示 buff 接收满数据，写 1 清零或读取之后才会开始下一帧数据的接收	RO	0x0

4	TXDONE	**TX 状态标志位** 0x0：正在发送数据 0x1：空闲	RO	0x0
3	UARTINT	**中断标志位** 高电平表示请求中断,写 1 清除中断	RO	0x0
2:0	-	-	-	-

11.4.3. UART0_BAUD0

Addr = 0xF8（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	UARTBAUDL	**波特率寄存器低 8Bytes** UART 波特率寄存器,计算公式：sysclk/(baud+1)	WO	0x0

11.4.4. UART0_BAUD1

Addr = 0xF9（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	UARTBAUDH	**波特率寄存器高 8Bytes** UART 波特率寄存器,计算公式：sysclk/(baud+1)	WO	0x0

11.4.5. UART0_DATA

Addr = 0xFA（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	DATA	**数据寄存器** 使能之后向该寄存器写入数据则触发该数据的发送,读该寄存器取得接收的数据	RW	0x0

11.4.6. UART1_CON

Addr = 0xFB（SFR）

Bit(s)	Name	Description	R/W	Reset
7	STOPBIT	**停止位控制位** 0x0：不发送停止位 0x1：发送 1bit 停止位	RW	0x0
6	NINTHBIT	**将需要发送的第 9bit 数据写入该寄存器**	RW	0x0
5	BIT9EN	**发送 9bit 数据使能位** 0x0：一次发送 8bit 数据 0x1：一次发送 9bit 数据	RW	0x0
4	UARTEN	**UART 使能位** 0x0：关闭 0x1：打开	RW	0x0
3	TXINV	**TX 电平取反控制位** 0x0：不取反 0x1：取反	RW	0x0
2	RXINV	**RX 电平取反控制位** 0x0：不取反 0x1：取反	RW	0x0
1	TXRXSEL	**TX/RX 选择位** 0x0：关闭 0x1：打开	RW	0x0
0	UARTIE	**RX 中断使能位** 0x0：关闭 0x1：打开	RW	0x0

11.4.7. UART1_STA

Addr = 0xFC（SFR）

Bit(s)	Name	Description	R/W	Reset

7	RXBIT9	**接收的第 9bit,读该位读出第 9bit 数据**	RO	0x0
6	FERR	**接收到的奇偶校验位数据** 奇偶校验由软件进行(接收 8bit 数据加奇偶校验位时不使能 BIT9_EN 的情况下，在此位读出奇偶校验位)	RO	0x0
5	RXDONE	**RX 状态标志位** 该位为 1 表示 buff 接收满数据，写 1 清零或读取之后才会开始下一帧数据的接收	RO	0x0
4	TXDONE	**TX 状态标志位** 0x0：正在发送数据 0x1：空闲	RO	0x0
3	UARTINT	**中断标志位** 高电平表示请求中断,写 1 清除中断	RO	0x0
2:0	-	-	-	-

11.4.8. UART1_BAUD0

Addr = 0xFD (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	UARTBAUDL	**波特率寄存器低 8Bytes** UART 波特率寄存器,计算公式：sysclk/(baud+1)	WO	0x0

11.4.9. UART1_BAUD1

Addr = 0xFE (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	UARTBAUDH	**波特率寄存器高 8Bytes** UART 波特率寄存器,计算公式：sysclk/(baud+1)	WO	0x0

11.4.10. UART1_DATA

Addr = 0xFF（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	DATA	**数据寄存器** 使能之后向该寄存器写入数据则触发该数据的发送,读该寄存器取得接收的数据	RW	0x0

11.5. 使用流程说明

发送数据：

使能模块（UART0->CON |= BIT(4)），将需要发送的数据写入 DATA 即开始发送(UART0->DATA=x)，如需发送 9bit 数据则使能 BIT9_EN 并先将第 9bit 数据写入 NINTH_BIT 再将前 8 位数据写入 DATA 开始发送。

接收数据：

只需使能模块即开始检测起始位，当接收满一帧数据 RX_DONE 会置 1 表示 buff 收满,此时可将接收的数据读走，必需将 RX_DONE 写 1 清零才会接收下一帧数据(UART0->STA=BIT(5))。

12. 基本 Timer 0/1 模块

12.1. 功能概述

基本 Timer0/1 是两个 8bit 的基础功能定时器，支持多种计数时钟源选择，支持定时器模式，计数器模式，捕获模式，和 PWM 模式等多种工作模式。支持 Timer0 和 Timer1 级联组

成 16bit 的定时器工作模式。

12.2. 模块框图

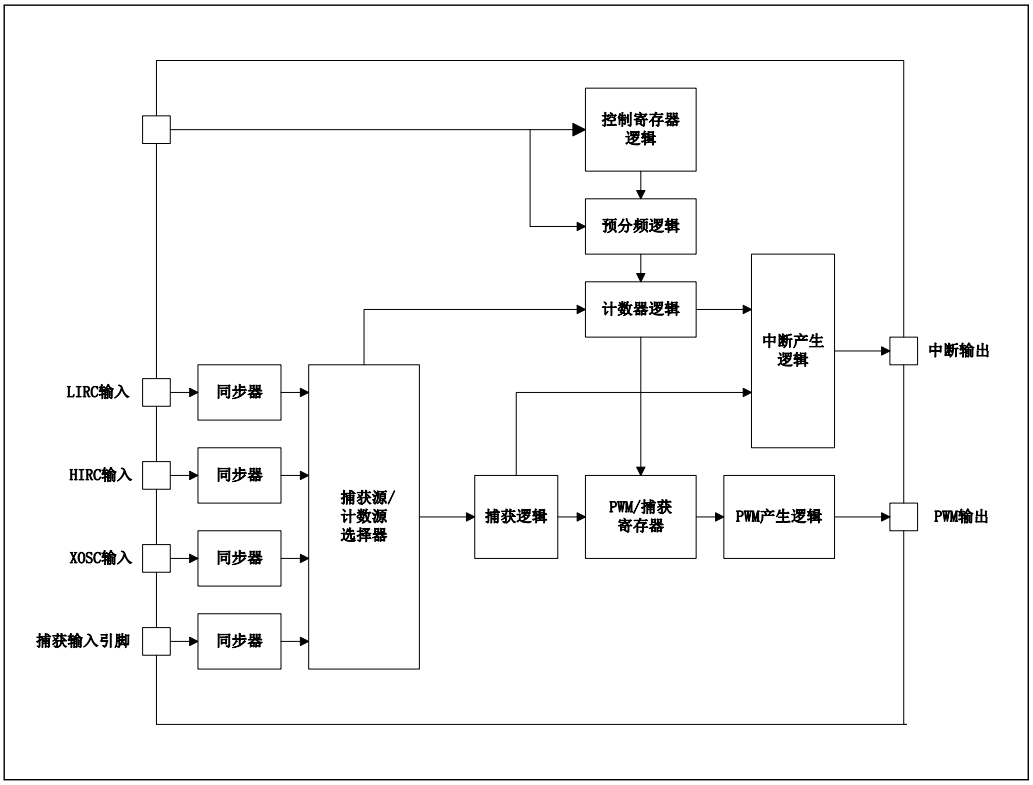


图 12-1 Timer0/1 模块框图

12.3. 寄存器列表

表 12-1 Timer0/1 寄存器列表

Address	Register Name	Description
0x88 (SFR)	TMR0_CONL	TIMER0 control low 8bit register
0x89 (SFR)	TMR0_CONH	TIMER0 control high 8bit register
0x8A (SFR)	TMR0_CNTL	TIMER0 counter low 8bit register

0x8C (SFR)	TMR0_PRL	TIMER0 period low 8bit register
0x8E (SFR)	TMR0_PWML	TIMER0 PWM low 8bit register
0xB0 (SFR)	TMR1_CONL	TIMER1 control low 8bit register
0xB1 (SFR)	TMR1_CONH	TIMER1 control high 8bit register
0xB2 (SFR)	TMR1_CNTL	TIMER1 counter low 8bit register
0xB4 (SFR)	TMR1_PRL	TIMER1 period low 8bit register
0xB6 (SFR)	TMR1_PWML	TIMER1 PWM low 8bit register

12.4. 寄存器详细说明

12.4.1. TMR0_CONL

Addr = 0x88 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:5	PSC	**TIMER0 预分频配置** 0x0：不分频 0x1：2 分频 0x2：4 分频 0x3：8 分频 0x4：16 分频 0x5：32 分频 0x6：64 分频 0x7：128 分频	RW	0x0
4:2	INCSRC	**TIMER0 计数源选择配置** 0x0：T0 rising edge 0x1：T0 falling edge 0x2：hirc_clk_div2 edge(rising & falling) 0x3：rc64k_div2 edge(rising & falling) 0x4：xoscm_div2 edge(rising & falling)	RW	0x0

		0x5：Timer1 ov 0x6：sys_clk 0x7：sys_clk		
1:0	MODE	**TIMER0 工作模式配置** 0x0：关闭 0x1：COUNTER MODE 0x2：PWM MODE 0x3：CAPTURE MODE	RW	0x0

12.4.2. TMR0_CONH

Addr = 0x89（SFR）

Bit(s)	Name	Description	R/W	Reset
7	TMRPND	**TIMER0 计数 pending 位（写 1 清 pending）** 0x0：没有计数 pending 0x1：有计数 pending	RW	0x0
6	CAPPND	**TIMER0 捕获 pending 位（写 1 清 pending）** 0x0：没有捕获 pending 0x1：有捕获 pending	RW	0x0
5	TMRIE	**TIMER0 计数中断使能位** 0x0：关闭 0x1：打开	RW	0x0
4	CAPIE	**TIMER0 捕获中断使能位** 0x0：关闭 0x1：打开	RW	0x0
3:2	INCSRC	**TIMER0 捕获源选择配置** 0x0：T0 引脚作为捕获源 0x1：T0 引脚作为捕获源 0x2：比较器 0 的数字输出作为捕获源 0x3：比较器 1 的数字输出作为捕获源	RW	0x0
1:0	CAPEDG	**TIMER0 捕获 T0 引脚的边沿触发设置** 0x0：T0 上升沿触发捕获	RW	0x0

		0x1：T0 下降沿触发捕获 0x2：T0 双边沿触发捕获 0x3：T0 双边沿触发捕获		

12.4.3. TMR0_CNTL

Addr = 0x8A（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	CNT	**TIMER0 计数器初始值**	RW	0x0

12.4.4. TMR0_PRL

Addr = 0x8C（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	PRD	**TIMER0 计数周期设置值**	RW	0xFF

12.4.5. TMR0_PWML

Addr = 0x8E（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	PWM	**TIMER0 占空比设置值** 在 PWM 工作模式是，该值是设置位 PWM 的占空比设置值；在捕获工作模式时，当捕获到捕获源之后抓取的计数器的值锁存在 PWM 寄存器中。	RW	0x0

12.4.6. TMR1_CONL

Addr = 0xB0（SFR）

Bit(s)	Name	Description	R/W	Reset
7:5	PSC	**TIMER1 预分频配置** 0x0：不分频 0x1：2 分频 0x2：4 分频 0x3：8 分频 0x4：16 分频 0x5：32 分频 0x6：64 分频 0x7：128 分频	RW	0x0
4:2	INCSRC	**TIMER1 计数源选择配置** 0x0：T0 rising edge 0x1：T0 falling edge 0x2：hirc_clk_div2 edge(rising & falling) 0x3：rc64k_div2 edge(rising & falling) 0x4：xoscm_div2 edge(rising & falling) 0x5：Timer1 over 0x6：sys_clk 0x7：sys_clk	RW	0x0
1:0	MODE	**TIMER1 工作模式配置** 0x0：关闭 0x1：COUNTER MODE 0x2：PWM MODE 0x3：CAPTURE MODE	RW	0x0

12.4.7. TMR1_CONH

Addr = 0xB1 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	TMRPND	**TIMER1 计数 pending 位（写 1 清 pending）** 0x0：没有计数 pending 0x1：有计数 pending	RW	0x0

6	CAPPND	**TIMER1 捕获 pending 位（写 1 清 pending）** 0x0：没有捕获 pending 0x1：有捕获 pending	RW	0x0
5	TMRIE	**TIMER1 计数中断使能位** 0x0：关闭 0x1：打开	RW	0x0
4	CAPIE	**TIMER1 捕获中断使能位** 0x0：关闭 0x1：打开	RW	0x0
3:2	INCSRC	**TIMER1 捕获源选择配置** 0x0：T1 引脚作为捕获源 0x1：T1 引脚作为捕获源 0x2：比较器 0 的数字输出作为捕获源 0x3：比较器 1 的数字输出作为捕获源	RW	0x0
1:0	CAPEDG	**TIMER1 捕获 T0 引脚的边沿触发设置** 0x0：T1 上升沿触发捕获 0x1：T1 下降沿触发捕获 0x2：T1 双边沿触发捕获 0x3：T1 双边沿触发捕获	RW	0x0

12.4.8. TMR1_CNTL

Addr = 0xB2 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	CNT	**TIMER1 计数器初始值**	RW	0x0

12.4.9. TMR1_PRL

Addr = 0xB4 (SFR)

Bit(s)	Name	Description	R/W	Reset

7:0	PRD	TIMER1 计数周期设置值	RW	0xFF

12.4.10. TMR1_PWML

Addr = 0xB6 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	PWM	TIMER1 占空比设置值 在 PWM 工作模式是，该值是设置位 PWM 的占空比设置值；在捕获工作模式时，当捕获到捕获源之后抓取的计数器的值锁存在 PWM 寄存器中。	RW	0x0

12.5. 使用流程说明

以 Timer0 为例，Timer1 同 Timer0。

12.5.1. 计数器/定时器工作模式

(1) 写 TMR0_CONH.bit7，TMR0_CONH.bit6 1 清 pending;

(2) 配置计数器初始值，写寄存器 TMR0_CNTL;

(3) 配置计数周期值，写寄存器 TMR0_PRL;

(4) 选择计数器的计数源，写寄存器 TMR0_CONL[4:2];

(5) 配置计数源的预分频，写寄存器 TMR0_CONL[7:5];

(6) 如果选择使用计数中断，写寄存器 TMR0_CONH[5]配置 TMR_IE=1;

(7) 配置 TMR0_CONL[1:0]=2’b01,工作在 Timer mode;

(8) 等 TMR0_CONH[7]=1,即产生了 TIMER PENDING;如果中断使能了，会进入中断，执行对应的中断服务子程序。

12.5.2. 捕获工作模式

(1) 写 TMR0_CONH.bit7，TMR0_CONH.bit6 1 清 pending；

(2) 配置计数器初始值，写寄存器 TMR0_CNTL；

(3) 配置计数周期值，写寄存器 TMR0_PRL；

(4) 配置 TMR0_PWML=0；

(5) 选择计数器的计数源，写寄存器 TMR0_CONL[4:2]；

(6) 配置计数源的预分频，写寄存器 TMR0_CONL[7:5]；

(7) 配置 TMR0_CONH[3:2]选择捕获源；

(8) 如果捕获源选择的是 T0 引脚，需要配置 TMR0_CONH[1:0]选择捕获边沿；

(9) 如果选择使用捕获中断，写寄存器 TMR0_CONH[4]配置 CAP_IE=1；

(10) 配置 TMR0_CONL[1:0]=2’b11,工作在捕获模式；

(11) 等 TMR0_CONH[6]=1,即产生了 CAP PENDING;如果中断使能了，会进入中断，执行对应的中断服务子程序；

(12) 读取捕获事件发生时的计数器值，通过读寄存器 TMR0_PWML。

12.5.3. PWM 工作模式

(1) 写 TMR0_CONH.bit7，TMR0_CONH.bit6 1 清 pending；

(2) 配置计数器初始值，写寄存器 TMR0_CNTL；

(3) 配置计数周期值，写寄存器 TMR0_PRL；

(4) 配置 PWM 的占空比，写寄存器 TMR0_PWML；

(5) 选择计数器的计数源，写寄存器 TMR0_CONL[4:2]；

(6) 配置计数源的预分频，写寄存器 TMR0_CONL[7:5]；

(7) 配置 TMR0_CONL[1:0]=2’b10,工作在 PWM 模式。

13. 基本 Timer2 模块

13.1. 功能概述

基本 Timer2 是 16bit 的基础功能定时器，支持多种计数时钟源选择，支持定时器模式、计数器模式、捕获模式和 PWM 模式等多种工作模式。

13.2. 模块框图

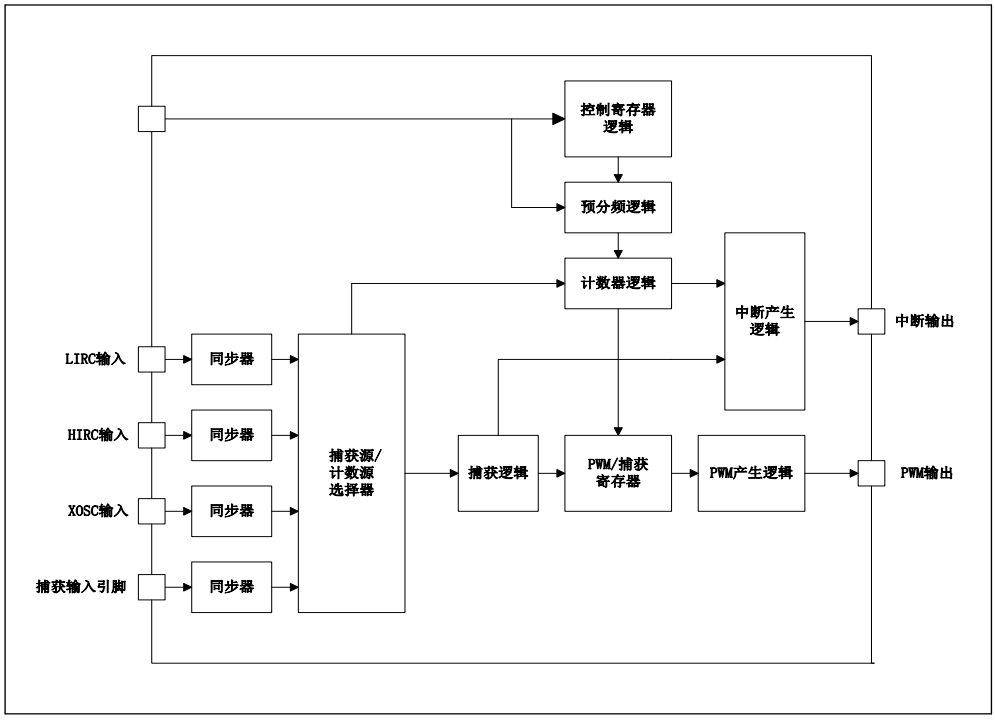


图 13-1 Timer2 的模块框图

13.3. 寄存器列表

表 13-1 Timer2 寄存器列表

Address	Register Name	Description
0xC0 (SFR)	TMR2_CONL	TIMER2 control low 8bit register
0xC1 (SFR)	TMR2_CONH	TIMER2 control high 8bit register
0xC2 (SFR)	TMR2_CNTL	TIMER2 counter low 8bit register
0xC3 (SFR)	TMR2_CNTH	TIMER2 counter high 8bit register
0xC4 (SFR)	TMR2_PRL	TIMER2 period low 8bit register
0xC5 (SFR)	TMR2_PRH	TIMER2 period high 8bit register
0xC6 (SFR)	TMR2_PWML	TIMER2 PWM low 8bit register
0xC7 (SFR)	TMR2_PWMH	TIMER2 PWM high 8bit register

13.4. 寄存器详细说明

13.4.1. TMR2_CONL

Addr = 0xC0 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:5	PSC	**TIMER2 预分频配置** 0x0：不分频 0x1：2 分频 0x2：4 分频 0x3：8 分频 0x4：16 分频 0x5：32 分频 0x6：64 分频 0x7：128 分频	RW	0x0

4:2	INCSRC	**TIMER2 计数源选择配置** 0x0：T2 rising edge 0x1：T2 falling edge 0x2：hirc_clk_div2 edge(rising & falling) 0x3：rc64k_div2 edge(rising & falling) 0x4：xoscm_div2 edge(rising & falling) 0x5：sys_clk 0x6：sys_clk 0x7：sys_clk	RW	0x0
1:0	MODE	**TIMER2 工作模式配置** 0x0：关闭 0x1：COUNTER MODE 0x2：PWM MODE 0x3：CAPTURE MODE	RW	0x0

13.4.2. TMR2_CONH

Addr = 0xC1 (SFR)

Bit(s)	Name	Description	RW	Reset
7	TMRPND	**TIMER2 计数 pending 位（写 1 清 pending）** 0x0：没有计数 pending 0x1：有计数 pending	RW	0x0
6	CAPPND	**TIMER2 捕获 pending 位（写 1 清 pending）** 0x0：没有捕获 pending 0x1：有捕获 pending	RW	0x0
5	TMRIE	**TIMER2 计数中断使能位** 0x0：关闭 0x1：打开	RW	0x0
4	CAPIE	**TIMER2 捕获中断使能位** 0x0：关闭 0x1：打开	RW	0x0
3:2	INCSRC	**TIMER2 捕获源选择配置**	RW	0x0

		0x0：T2 引脚作为捕获源 0x1：T2 引脚作为捕获源 0x2：比较器 0 的数字输出作为捕获源 0x3：比较器 1 的数字输出作为捕获源		
1:0	CAPEDG	**TIMER2 捕获 T2 引脚的边沿触发设置** 0x0：T2 上升沿触发捕获 0x1：T2 下降沿触发捕获 0x2：T2 双边沿触发捕获 0x3：T2 双边沿触发捕获	RW	0x0

13.4.3. TMR2_CNTL

Addr = 0xC2（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	CNTL	**TIMER2 计数器低 8bit 初始值**	RW	0x0

13.4.4. TMR2_CNTH

Addr = 0xC3（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	CNTH	**TIMER2 计数器高 8bit 初始值**	RW	0x0

13.4.5. TMR2_PRL

Addr = 0xC4（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	PRDL	**TIMER2 计数周期低 8bit 设置值**	RW	0xFF

13.4.6. TMR2_PRH

Addr = 0xC5 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	PRDH	**TIMER2 计数周期高 8bit 设置值**	RW	0xFF

13.4.7. TMR2_PWML

Addr = 0xC6 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	PWM	**TIMER2 占空比低 8bit 设置值** 在 PWM 工作模式是，该值是设置位 PWM 的占空比设置值；在捕获工作模式时，当捕获到捕获源之后抓取的计数器的值锁存在 PWM 寄存器中。	RW	0x0

13.4.8. TMR2_PWMH

Addr = 0xC7 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	PWM	**TIMER2 占空比高 8bit 设置值** 在 PWM 工作模式是，该值是设置位 PWM 的占空比设置值；在捕获工作模式时，当捕获到捕获源之后抓取的计数器的值锁存在 PWM 寄存器中。	RW	0x0

13.5. 使用流程说明

13.5.1. 计数器/定时器工作模式

(1) 写 TMR2_CONH.bit7，TMR2_CONH.bit6 1 清 pending；

(2) 配置计数器初始值，写寄存器 TMR2_CNTL，TMR2_CNTH；

(3) 配置计数周期值，写寄存器 TMR2_PRL，TMR2_PRH；

(4) 选择计数器的计数源，写寄存器 TMR2_CONL[4:2]；

(5) 配置计数源的预分频，写寄存器 TMR2_CONL[7:5]；

(6) 如果选择使用计数中断，写寄存器 TMR2_CONH[5]配置 TMR_IE=1；

(7) 配置 TMR2_CONL[1:0]=2’b01,工作在 Timer mode；

(8) 等 TMR2_CONH[7]=1，即产生了 TIMER PENDING;如果中断使能了，会进入中断，执行对应的中断服务子程序。

13.5.2. 捕获工作模式

(1) 写 TMR2_CONH.bit7，TMR2_CONH.bit6 1 清 pending；

(2) 配置计数器初始值，写寄存器 TMR2_CNTL，TMR2_CNTH；

(3) 配置计数周期值，写寄存器 TMR2_PRL，TMR2_PRH；

(4) 配置 TMR2_PWML=0，TMR2_PWMH=0；

(5) 选择计数器的计数源，写寄存器 TMR2_CONL[4:2]；

(6) 配置计数源的预分频，写寄存器 TMR2_CONL[7:5]；

(7) 配置 TMR2_CONH[3:2]选择捕获源；

(8) 如果捕获源选择的是 T2 引脚，需要配置 TMR2_CONH[1:0]选择捕获边沿；

(9) 如果选择使用捕获中断，写寄存器 TMR2_CONH[4]配置 CAP_IE=1；

(10) 配置 TMR2_CONL[1:0]=2’b11,工作在捕获模式;

(11) 等 TMR2_CONH[6]=1,即产生了 CAP PENDING;如果中断使能了，会进入中断，执行对应的中断服务子程序;

(12) 读取捕获事件发生时的计数器值，通过读寄存器 TMR2_PWML，TMR2_PWMH。

13.5.3. PWM 工作模式

(1) 写 TMR2_CONH.bit7，TMR2_CONH.bit6 1 清 pending;

(2) 配置计数器初始值，写寄存器 TMR2_CNTL，TMR2_CNTH;

(3) 配置计数周期值，写寄存器 TMR2_PRL，TMR2_PRH;

(4) 配置 PWM 的占空比，写寄存器 TMR2_PWML，TMR2_PWMH;

(5) 选择计数器的计数源，写寄存器 TMR2_CONL[4:2];

(6) 配置计数源的预分频，写寄存器 TMR2_CONL[7:5];

(7) 配置 TMR2_CONL[1:0]=2’b10，工作在 PWM 模式。

14. 高级 Timer 1/2 模块

14.1. 功能概述

高级定时器是一个包含两个定时器 STMR1/2。STMR1/2 是功能相同的高级计数器，可用于产生不同形式的时钟波形，一个定时器可以产生同频的一组互补 PWM，或者周期相同、占空比不同的 2 路 PWM。可以捕获外界输入进行脉冲宽度或周期测量。

主要特性如下：

- 内置 16 位计数器，向上或者向下计数，自动重装
- 时钟源可选择来自系统系统时钟或者内部晶振或者外部RTC时钟

- 预分频 1-16
- 后分频 1、2、4、8、16、32、64、128(周期间隔响应)
- 和外部信号同步（外部时钟，复位）
- 输入捕捉（上升沿，下降沿和双沿）和比较功能
- 对输入沿计数，可选上升沿，下降沿和双沿
- 刹车输入，可以将TMR1/2 的输出置位特定的状态
- 支持PWM输出功能
- 可输出周期相同、占空比不同的 2 路PWM，或者 1 路互补PWM，互补输出可编程死区
- 支持刹车功能，刹车输入为比较器输出
- 影子寄存器，触发更新事件才能更新
- 中断，在以下事件产生中断：
  - 计数器上溢
  - 计数器下溢
  - CHA/CHB 输入捕捉
  - CHA/CHB 输出比较
  - 刹车产生（软件或比较器）

14.1.1. 基本动作

1、基本波形模式

TIMER1/2 有 2 种基本计数波形模式，锯齿波模式和三角波模式。波形模式又由于不同的内部计数动作有所细分，三角波模式分为三角波 A 模式、三角波 B 模式。锯齿波和三角波的基本波形如所示。三角波 A 模式与三角波 B 模式区别在于缓存传送有差别，三角波 A 模式一个周期只发生一次缓存传送（谷点），而三角波 B 模式一个周期发生两次缓存传送（峰点和谷点）。

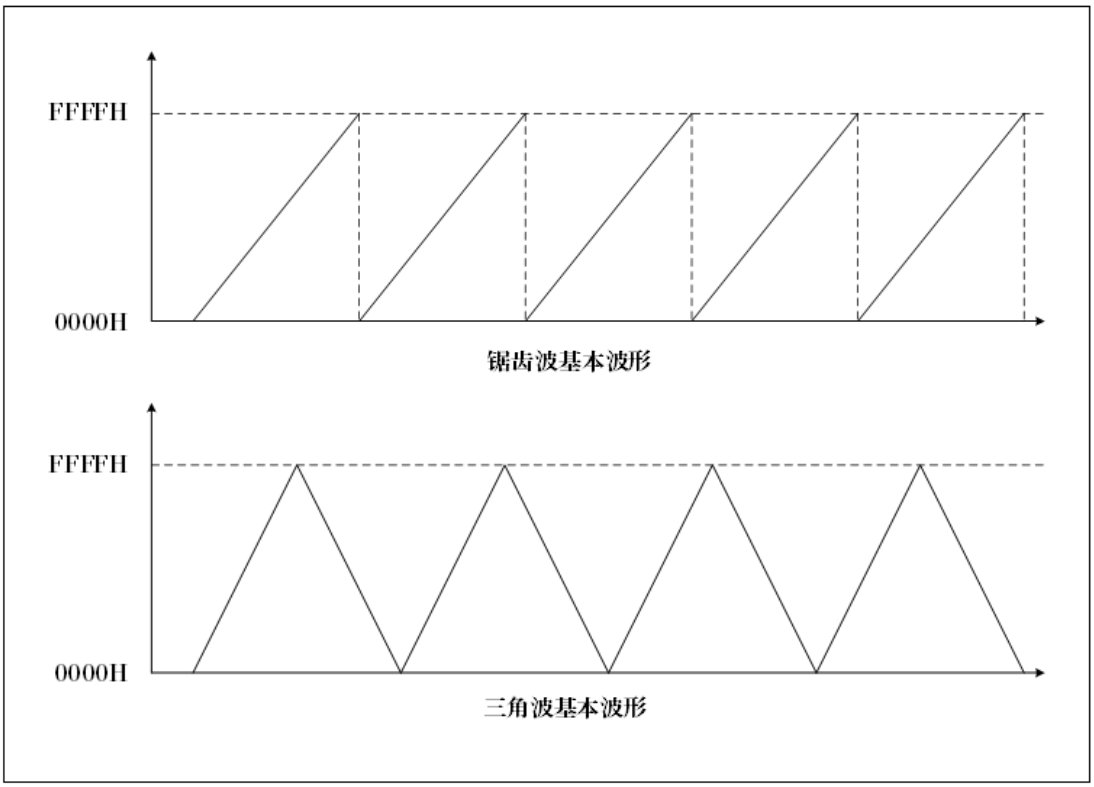


图 14-1 基础波形图

2、比较输出

STMR1/2 一个定时器有 2 个比较输出端口（CHA、CHB），可在计数值与计数基准值比较匹配时输出指定的电平。CMPA_S（{CMPAH_S,CMPAL_S}）、CMPB_S（{CMPBH_S,CMPBL_S}）寄存器分别对应了 CHA、CHB 的计数比较基准值。当计数器的计数值和 CMPA_S 相等时，CHA 端口输出指定的电平；当计数器的计数值和 CMPB_S 相等时，CHB 端口输出指定电平。

CHA、CHB 端口的计数起始电平和计数比较匹配时的电平由 PAINITVAL/PBINITVAL 和 CAPAVAL/CAPBVAL 定义。图为比较输出的动作例。

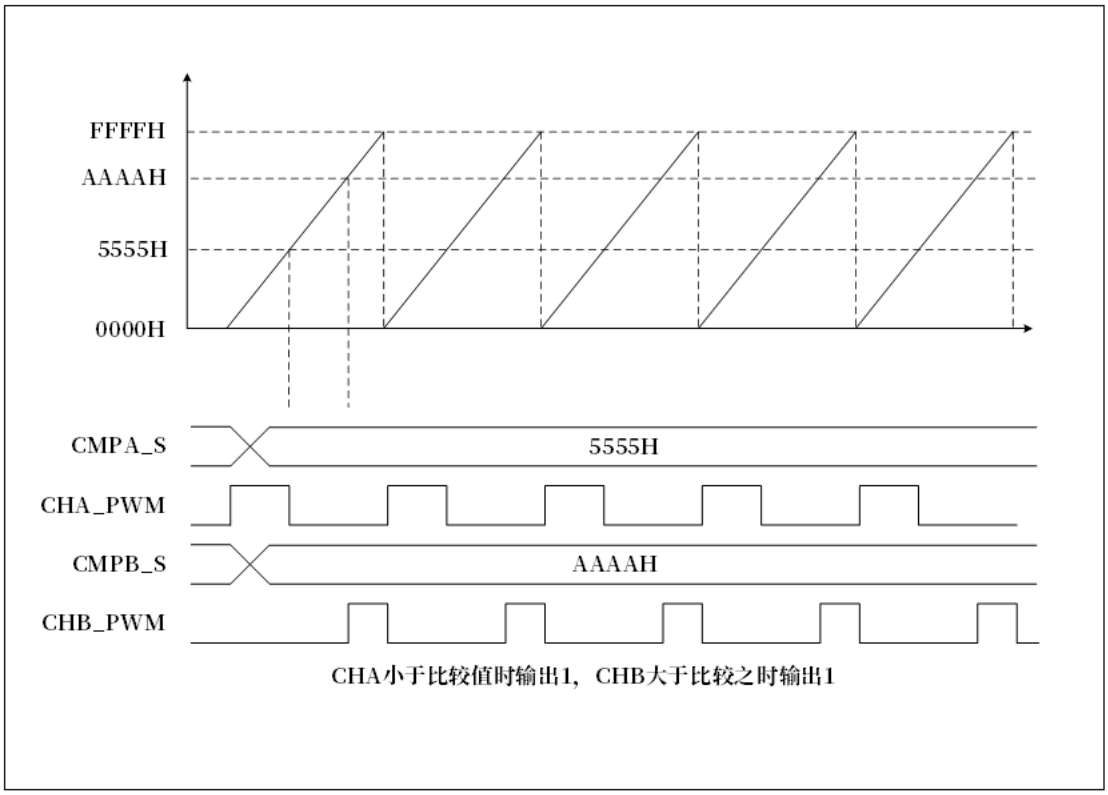


图 14-2 比较输出波形图

3、捕获输入

STMR1/2 都具有捕获输入功能，具备 2 组捕获输入寄存器（CMPA_S、CMPB_S），用于保存捕获到的计数值。设定端口控制寄存器（PCONRA/PCONRB）的 CAPA_EN/CAPB_EN 位为 1，对应端口的捕获输入功能就有效了。当设定了对应的捕获输入条件且该条件有效时，当前的计数值就被保存到相应的寄存器（CMPA_S、CMPB_S）中。每组捕获输入的条件可选 CHA 或 CHB 的上升沿，下降沿或上升下降沿，通过 CAPA_MODE/CAPB_MODE 来设定对应端口的捕获条件。

STMRxPRL_S 和 STMRxPRH_S 这两个寄存器决定了定时器内部计数器的溢出时间，捕获模式状态下，建议两个寄存器都设置成 0xFF，捕获模式推荐使用锯齿波模式，捕获图参考下图。

捕获模式下读取基准寄存器 STMRxPRL_S、STMRxPRH_S、STMRxCMPAL_S、STMRxCMPAH_S、STMRxCMPBL_S、STMRxCMPBH_S (周期值和比较值)时，通过 STMRxCR 的 SEL_SREG 设置来决定

是读取影子寄存器还是基准寄存器的值。SEL_SREG 设置成 0，读取基准寄存器的值（即捕获值），否则读取的是影子寄存器的值。捕获模式下写比较值影子寄存器没有意义。

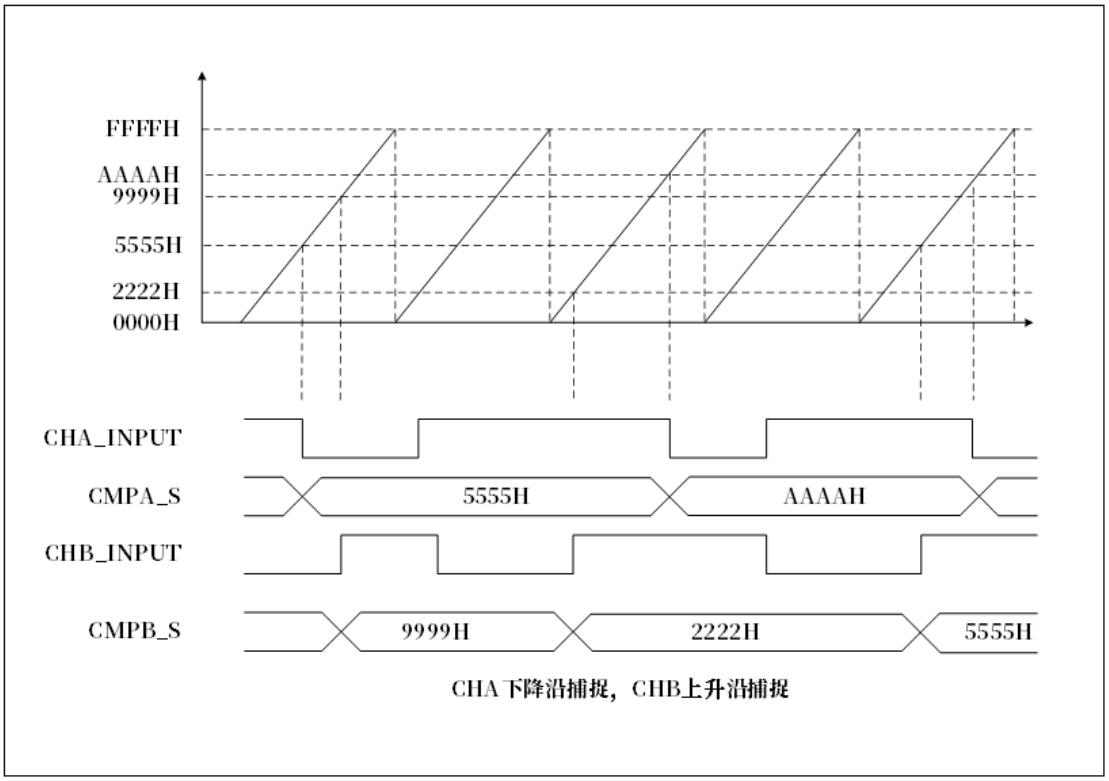


图 14-3 捕获模式波形

14.1.2. 时钟源选择

STMR1/2 的计数时钟可以有以下几种选择：

- 系统时钟（SYSCLK）
- 内部低速RC振荡器 32kHz时钟
- 外部晶振（计数时钟为外部晶振/2）
- CHA/ CHB

时钟分频 1-16 可选，可对内部低速 RC、外部低速 RC 、CHA、 CHB 上升沿，下降沿，上升下降沿可选计数。

14.1.3. 计数方向

STMR1/2 的计数器计数方向可通过软件方式改变。不同波形模式时，改变计数方向的方法略有不同。

14.1.3.1. 锯齿波计数方向

锯齿波模式时，计数方向可在计数器计数中或停止时设定。

在递加计数中时，设定 STMRxCR.DIR=0（递减计数），则计数器计数到上溢后变为递减计数模式；在递减计数中时，设定 STMRxCR.DIR=1（递加计数），则计数器计数到下溢后变为递加计数模式。

在计数停止时，设定 STMRxCR.DIR 位。则计数开始后，STMRxCR.DIR 的值立即更新。

Note：在计数模式开启状态下改变 STMRxCR.DIR 的值，如果 CR 寄存器有其他位需要配置，请配置好其它位之后最后配置 DIR 位。或者不使用（或约与与约），配置 CR 寄存器。

修改 DIR 值时如果同时修改周期、比较值，可能会出现需要等待按照原周期原计数至顶部或谷底溢出（根据 DIR 方向决定）才会按照新修改的周期和比较值进行。

14.1.3.2. 三角波计数方向

三角波模式时，计数方向只能在计数器停止时设定。在计数中设定计数方向无效。在计数停止时，设定 STMRxCR.DIR 位，立即生效。

14.1.4. 数字滤波

STMR1/2 的 CHA、CHB 端口输入都有数字滤波功能。可通过设定 STMRxPCONRA.PA_FILTER_EN/STMRxPCONRB.PB_FILTER_EN 开启对应端口的滤波功能。滤波时

钟为计数器当前工作时钟。

在滤波采样基准时钟采样到端口上 3 次一致的电平时，该电平被当作有效电平传送到模块内部；小于 3 次一致的电平会被当作外部干扰滤掉，不传送到模块内部。其动作例如所示。

数字滤波也用于对电压比较器传过来的信号滤波，通过 STMR1_CR.THA_FILTER_EN/STMR1_CR.THB_FILTER_EN 开启，此时滤波时钟为系统时钟。

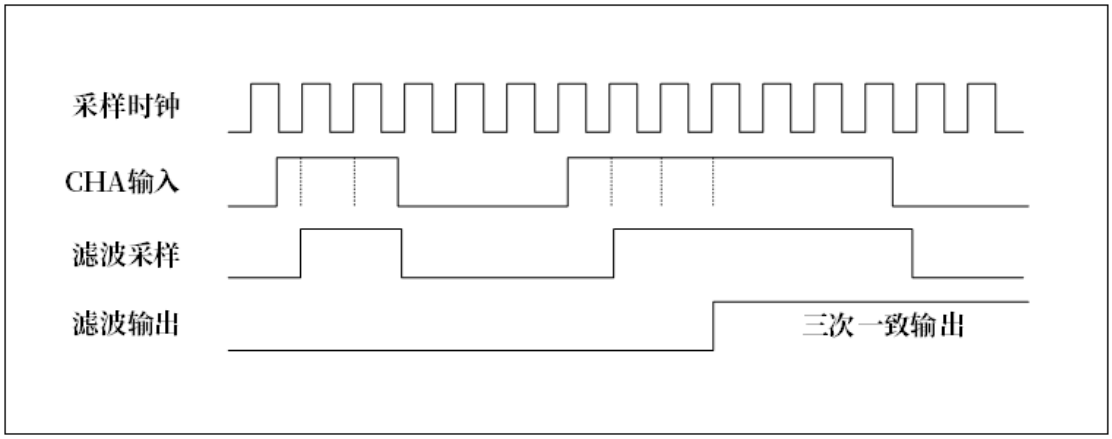


图 14-4 数字滤波波形图

14.1.5. 软件同步

STMR1/2 可通过设定软件同步启动寄存器 SSTAR（STMR_ALLCON[3]），实现目标 STMR1/2 的同步启动。

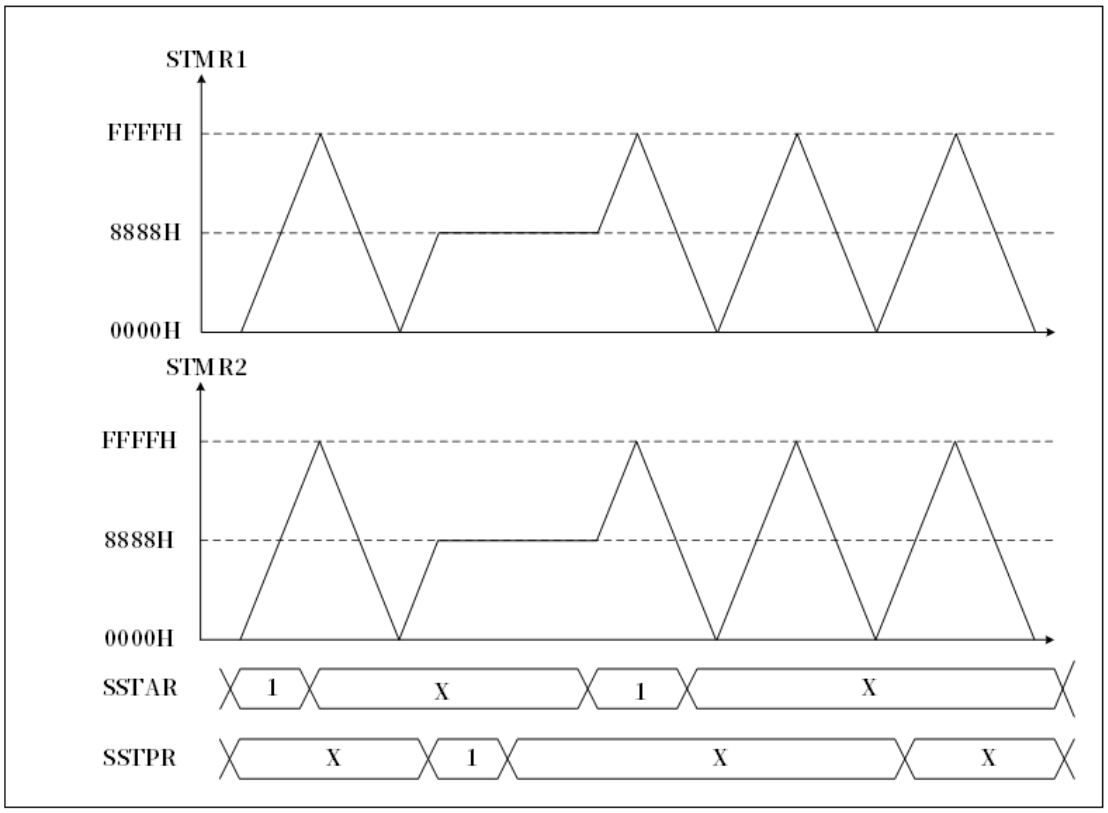


图 14-5 同步启动和停止波形图

14.1.5.1. 软件同步停止

STMR1/2 可通过设定软件同步停止寄存器 SSTPR（STMR_ALLCON[4]），实现目标 STMR1/2 的同步停止，此时计数器处于暂停状态，对同步启动寄存器 SSTAR（STMR_ALLCON[3]）写 1 可以继续计数。

14.1.5.2. 软件同步清零

STMR1/2 可通过设定软件同步清零寄存器（STMR_ALLCON[5]），实现目标 STMR1/2 的同步清零，此时计数器会复位到初始状态。

若设定 STMR_ALLCON[3],即可实现 STMR1/2 的软件同步启动。

软件同步动作相关寄存器是一组独立于 STMR1/2 外、各个 STMR 间共用的寄存器，这组寄存器的各个位只在写 1 时有效，写 0 无效。

14.1.6. 缓存功能

缓存动作是指在发生更新事件时，发生以下事件：

PR={PRH,PRL}； CMPA={CMPAH,CMPAL}；CMPB={CMPBH,CMPBL}；

a.通用周期基准值影子寄存器 PR 的值自动传送到通用周期基准值寄存器 PR_S（{PRH_S,PRL_S}）中；

b.通用比较基准值影子寄存器（CMPA、CMPB）的值自动传送到通用比较基准值寄存器（CMPA_S、CMPB_S）中（比较输出时）；

c.通用比较基准值寄存器（CMPA_S、CMPB_S）的值自动传送到通用比较基准值影子寄存器（CMPA、CMPB）中（捕获输入时）；

如图所示，是比较输出动作时、通用比较基准值寄存器的单缓存方式的时序图。从中可以看到，在计数期间改变通用比较基准值寄存器（CMPA_S）的值可以调整输出占空比，改变通用周期基准值寄存器（PR）的值可以调整输出周期。

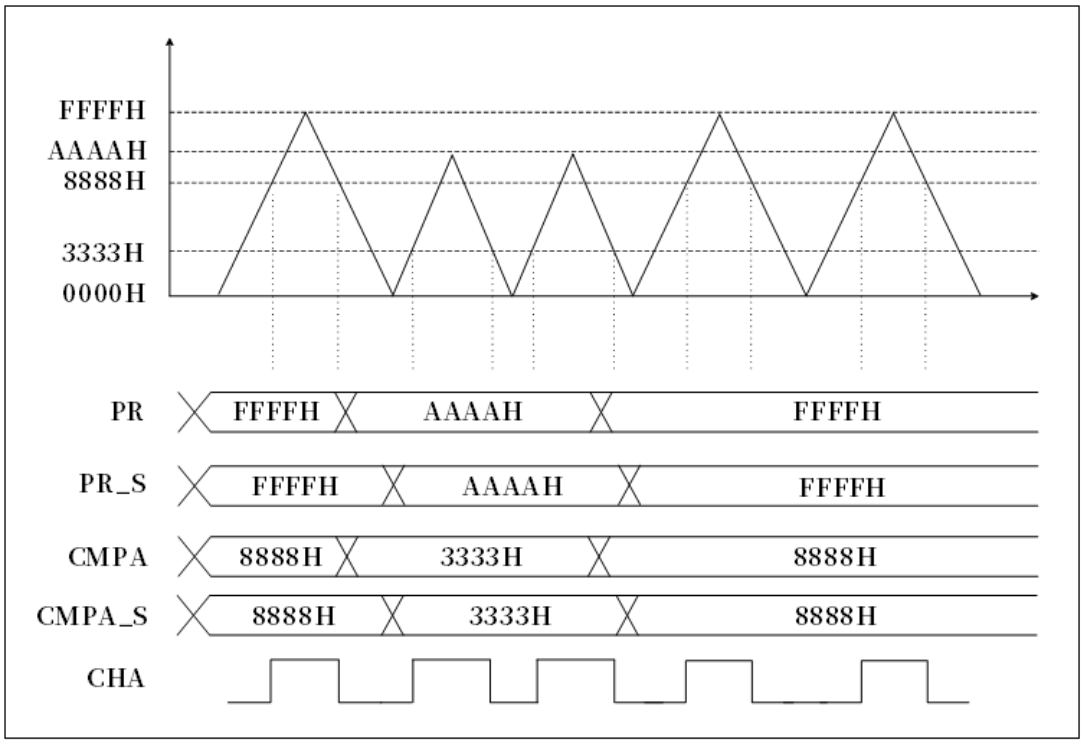


图 14-6 自动装载波形图

14.1.6.1. 缓存传送时间点

周期基准值缓存传送时间点为锯齿波时递加计数上溢点或递减计数下溢点。

三角波模式,缓存传送发生在（CNT=PR_S）上溢点或（CNT=0）下溢点。

三角波 A 模式时，缓存传送发生在计数下溢点。

三角波 B 模式时，缓存传送发生在计数下溢点和计数上溢点。

捕获输入动作缓存传送时间点为捕获输入动作时。

在计数模式启动（STMRxCR.TMREN）以及正常的比较期间若有清零动作产生时，通用周期基准值、通用比较基准值、等会根据相应的缓存动作设定状况发生一次缓存传送（更新事件）。

14.1.7. 通用 PWM 输出

14.1.7.1. 独立 PWM 输出

每个定时器的 2 个端口 CHA、CHB 能独立的输出 PWM 波。如图所示，定时器 CHA 端口输出 PWM 波。

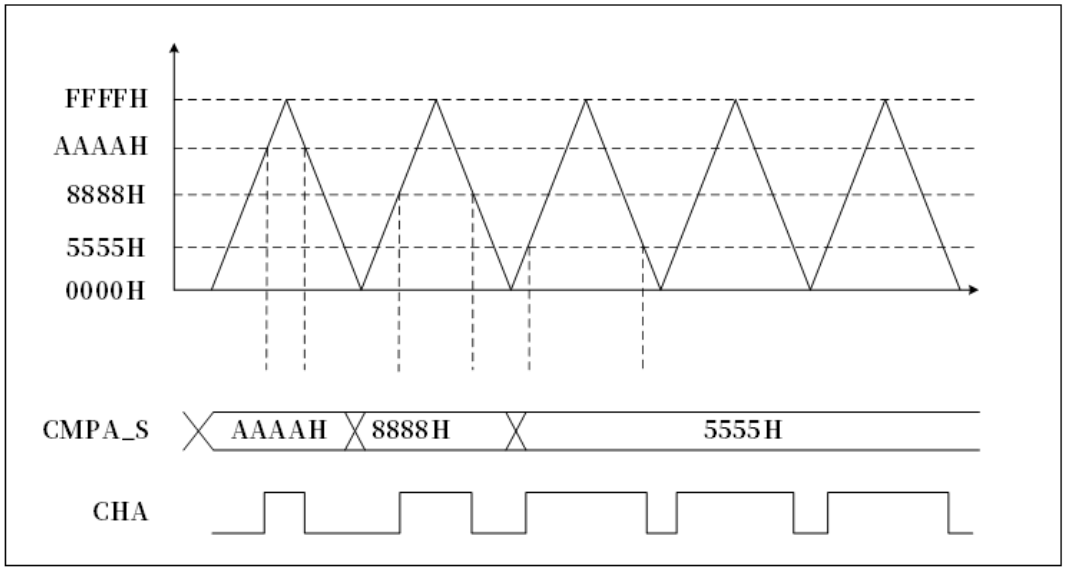


图 14-7 定时器 CHA 端口输出 PWM

14.1.7.2. 互补 PWM 输出

CHA 端口和 CHB 端口，在不同的模式下通过设定端口的极性可组合输出互补 PWM 波形。

软件设定 CMPB_S 互补 PWM 输出。

软件设定 CMPB_S 互补 PWM 输出是指在锯齿波模式和三角波 A 模式、三角波 B 模式下，通过设定相反的输出极性，形成互补 PWM 输出。用于 CHB 端口波形输出的通用比较基准值寄存器（CMPB_S）的值由寄存器(CMPB)直接设定，与通用比较基准值寄存器（CMPA_S）的值没有直接关系。下图为软件设定 CMPB_S 互补 PWM 波的示例。

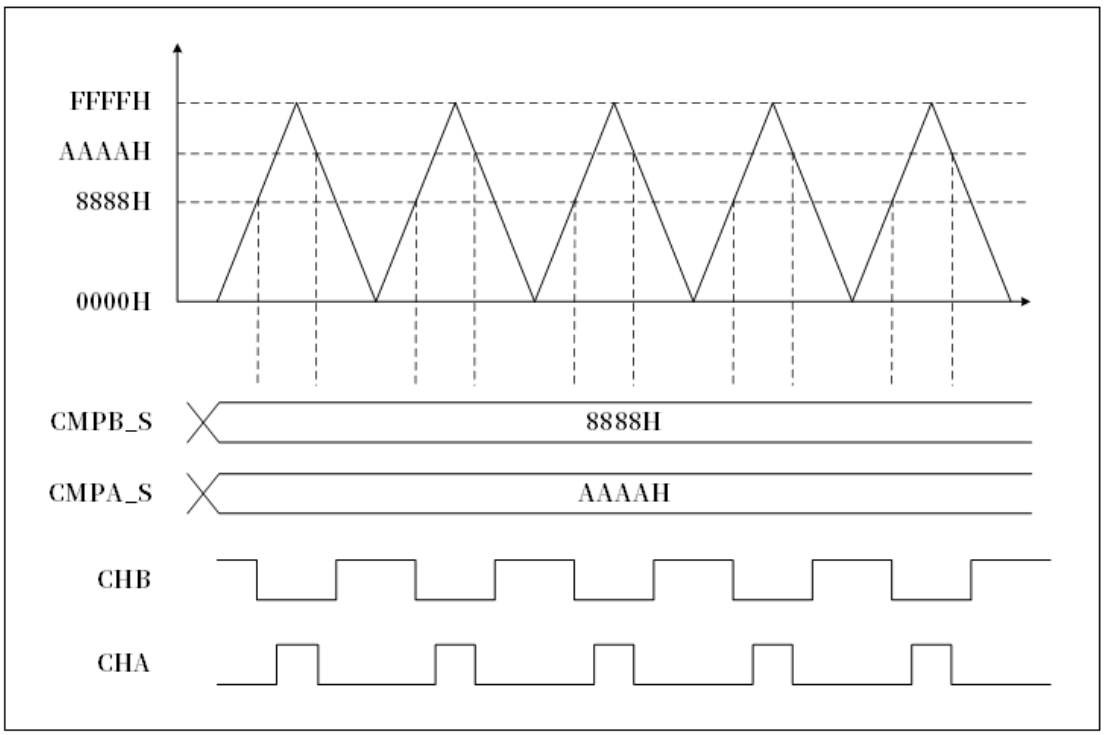


图 14-8 软件设定 CMPB_S 互补 PWM 波的示例

硬件设定 CMPB_S 互补 PWM 输出

硬件设定 CMPB_S 互补 PWM 输出是指在三角波 A 模式、三角波 B 模式下，用于 CHB 端口波形输出的通用比较基准值寄存器（CMPB_S）的值由通用比较基准值寄存器（CMPA_S）减去死区时间基准值寄存器（DTUA）的值运算决定。图为硬件设定 CMPB_S 互补 PWM 波输出例。死区时间基准值寄存器（DTUA）为 8bit，调整范围为 0~255。

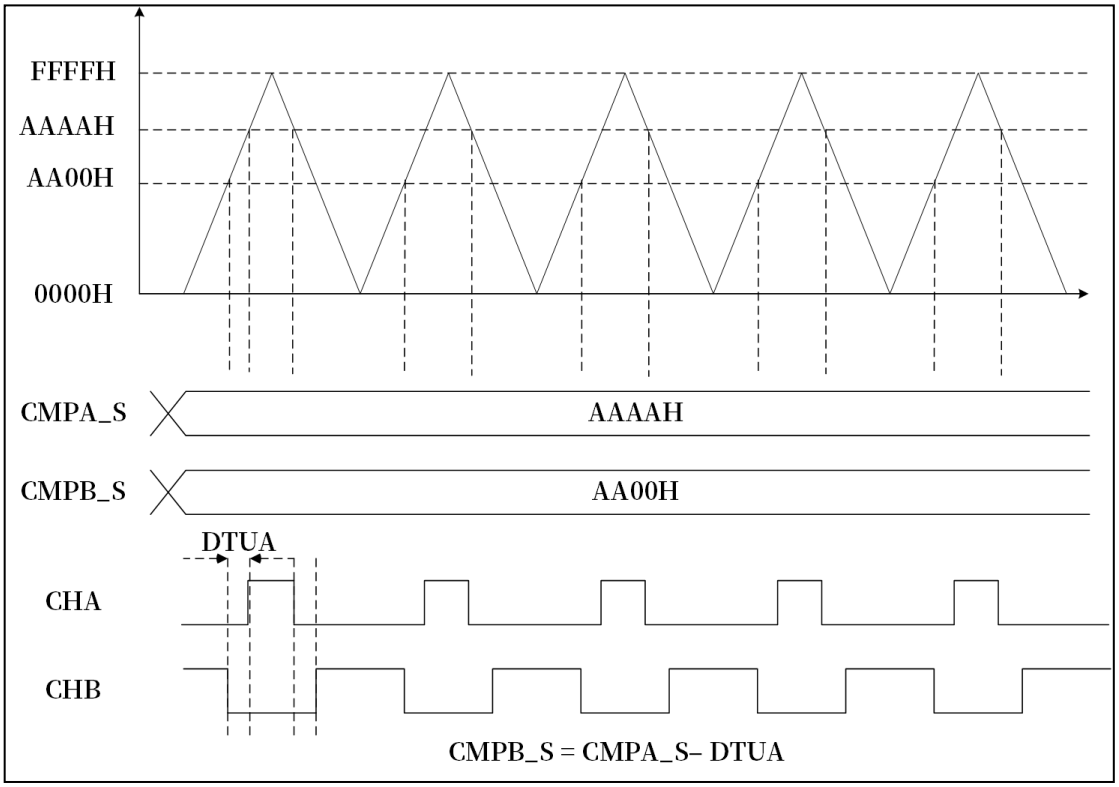


图 14-9 带死区互补波形图

14.1.8. 周期间隔响应

STMR1/2 的通用比较基准值寄存器（CMPA_S，CMPB_S），在计数比较匹配时可分别产生专用有效请求信号（CMPA_ACK/CMPB_ACK）。

该请求信号可以每间隔几个周期后产生一次有效的请求信号。通过设定有效周期寄存器（STMRxVPERR）的 STMRxVPERR.PCNTS 位来指定每隔多少个周期请求信号有效一次，其它周期内即使计数值和比较基准值寄存器 CMPA_S 或 CMPB_S 的值相等，也不会输出有效的请求信号。图所示是周期间隔有效请求信号的动作例。

CHA/ CHB 的比较值匹配的请求信号分别输出。

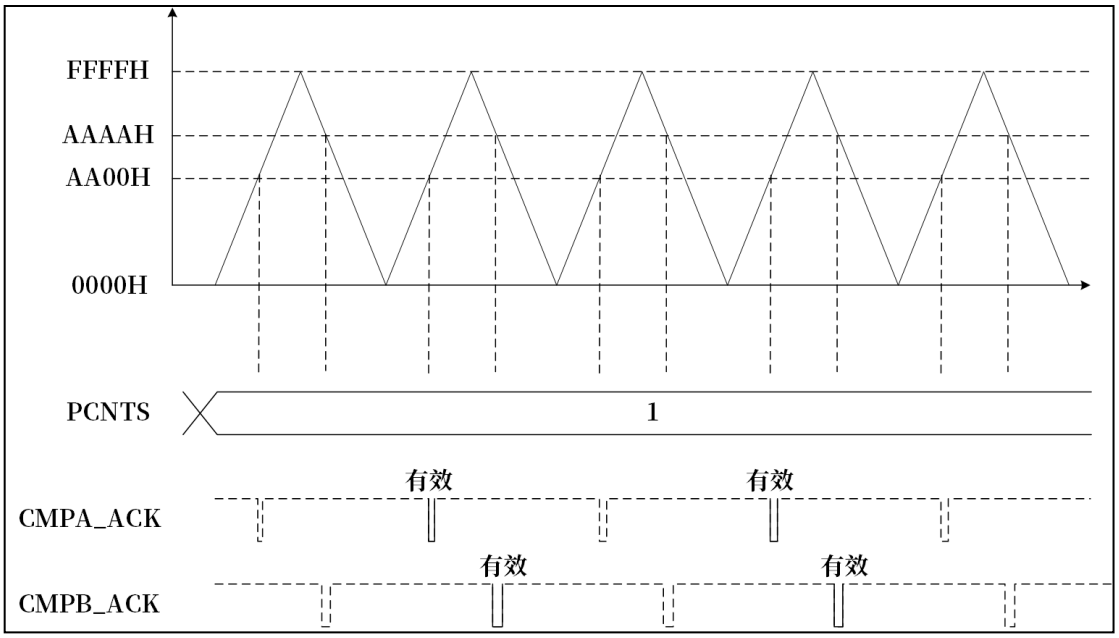


图 14-10 周期间隔有效请求信号的动作波形图

14.1.9. 保护机制

STMRx 可以对端口的输出状态进行保护控制。

STMRx 有 2 个共用的端口输入无效事件（来自模拟比较器，或者软件配置），每个接口上选通的异常状况事件可从刹车控制设定，当这些接口上监测到异常状况时，可以实现对通用 PWM 输出的控制。

端口在正常输出期间，若监测到从刹车控制过来的刹车事件，则端口的输出状态可变为预先设定好的状态。通用 PWM 输出端口在刹车控制异常事件发生时，端口状态可以变为输出高阻态、输出低电平或输出高电平（STMRxVPERR.BRAKEA_VAL/ STMRxDTR.BRAKEB_VAL 的设定决定）。

例如，若设定 BRAKEA_VAL=2’b10 时，则在 CHA 端口正常输出期间，若产生刹车事件，则 CHA 端口上输出变为高阻态。

Note：软件单独刹车 CHA，请设置 STMRxVPERR.BRAKEA_SF ；单独刹车 CHB 请设置

STMRxDTR.BRAKEB_SF;将这两个寄存器设置为 1，配合 T1A_EN、T1B_EN 使能则是软件刹车信号有效。STMRx_BRAKE.T1A_AOE/TIB_AOE 设置是软件复位，还是硬件复位，BRAKEA_VAL / BRAKEB_VAL 位设置输出端输出状态。

14.1.10. 中断说明

STMR1/2 各含有 4 类共计 6 个中断。分别是 2 个通用计数比较匹配中断（含 2 个捕获输入中断）、2 个计数周期匹配中断、2 个刹车保护中断。

14.1.11. 刹车保护

发生刹车事件时（来自电压比较器），硬件自动将端口状态改变为预设状态（高电平、低电平、高阻态）。

14.1.12. 内部互连

- 电压比较器可以触发刹车功能。
- STMR1/2 中断可以触发ADC采样功能。

14.1.13. 红外功能

高级 Timer2 通道 A 的 PWM 与基本 Timer2 的 PWM 组合可以作为红外发生器使用。通过将 SYS_CON2[3]置 1，使能红外功能。此时，基本 Timer2 作为载波 PWM，高级 Timer2 通道 A 作为调制 PWM，通过软件判断红外码为 1 或者 0，更改高级 Timer2 通道 A 的周期和比较值影

子寄存器，进行红外调制。结果通过 Timer2 的 PWM 输出到 GPIO。

14.2. 模块框图

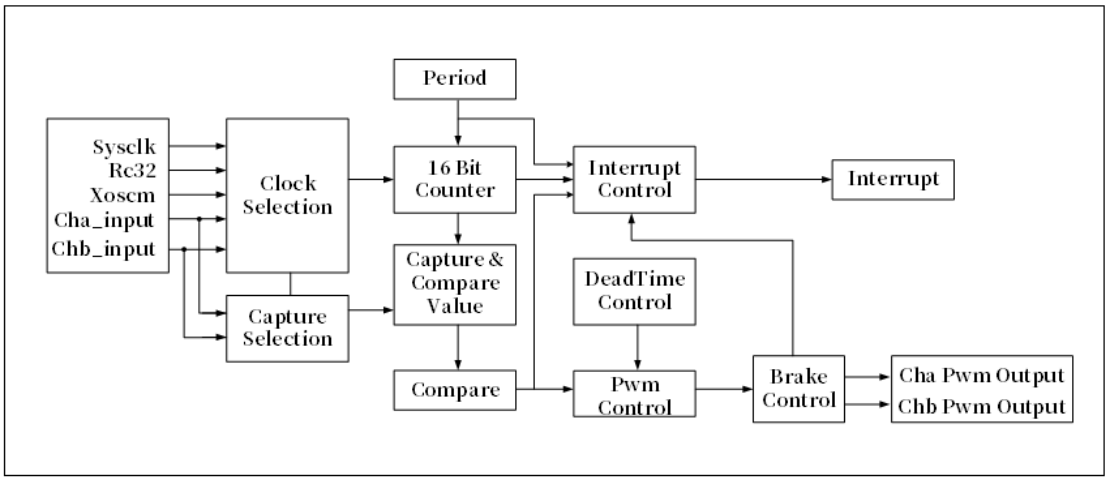


图 14-11 高级 Timer 模块框架图

14.3. 寄存器列表

表 14-1 STimer 寄存器列表

Address	Register Name	Description
0xC8 (SFR)	STMR1_CNTL	STMR1 Count Low Register
0xC9 (SFR)	STMR1_CNTH	STMR1 Count High Register
0xCA (SFR)	STMR1_PRL	STMR1 Period Low Register
0xCB (SFR)	STMR1_PRH	STMR1 Period High Register
0xCC (SFR)	STMR1_CMPAL	STMR1 Channel A Comparison Value Low Register
0xCD (SFR)	STMR1_CMPAH	STMR1 Channel A Comparison Value High Register
0xCE (SFR)	STMR1_CMPBL	STMR1 Channel B Comparison Value Low Register
0xCF (SFR)	STMR1_CMPBH	STMR1 Channel B Comparison Value High Register

0xD1 (SFR)	STMR1_CR	STMR1 Control Register
0xD2 (SFR)	STMR1_FCONR	STMR1 Time Control Register
0xD3 (SFR)	STMR1_VPERR	STMR1 Count Period Register
0xD4 (SFR)	STMR1_DTUA	STMR1 DeadTime Register
0xD5 (SFR)	STMR1_BRAKE	STMR1 Brake Control Register
0xD6 (SFR)	STMR1_DTR	STMR1 DeadTime Control Register
0xD7 (SFR)	STMR1_PCONRA	STMR1 Channel A Control Register
0xD8 (SFR)	STMR1_PCONRB	STMR1 Channel B Control Register
0xD9 (SFR)	STMR1_IE	STMR1 Interrupt Enable Register
0xDA (SFR)	STMR1_SR	STMR1 Interrupt Flag Register
0xDB (SFR)	STMR2_CNTL	STMR2 Count Low Register
0xDC (SFR)	STMR2_CNTH	STMR2 Count High Register
0xDD (SFR)	STMR2_PRL	STMR2 Period Low Register
0xDE (SFR)	STMR2_PRH	STMR2 Period High Register
0xDF (SFR)	STMR2_CMPAL	STMR2 Channel A Comparison Value Low Register
0xE1 (SFR)	STMR2_CMPAH	STMR2 Channel A Comparison Value High Register
0xE2 (SFR)	STMR2_CMPBL	STMR2 Channel B Comparison Value Low Register
0xE3 (SFR)	STMR2_CMPBH	STMR2 Channel B Comparison Value High Register
0xE4 (SFR)	STMR2_CR	STMR2 Control Register
0xE5 (SFR)	STMR2_FCONR	STMR2 Time Control Register
0xE6 (SFR)	STMR2_VPERR	STMR2 Count Period Register
0xE7 (SFR)	STMR2_DTUA	STMR2 DeadTime Register
0xE8 (SFR)	STMR2_BRAKE	STMR2 Brake Control Register

0xE9 (SFR)	STMR2_DTR	STMR2 DeadTime Control Register
0xEA (SFR)	STMR2_PCONRA	STMR2 Channel A Control Register
0xEB (SFR)	STMR2_PCONRB	STMR2 Channel B Control Register
0xEC (SFR)	STMR2_IE	STMR2 Interrupt Enable Register
0xED (SFR)	STMR2_SR	STMR2 Interrupt Flag Register
0xF5 (SFR)	STMR_ALLCON	STMR ALL Control Register

14.4. 寄存器详细说明

14.4.1. STMR1_CR

Addr = 0xD1 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	THBFILTEREN	**STMR1 刹车输入 CHB 滤波控制** 0x0：刹车输入 CHB 关数字滤波 0x1：刹车输入 CHB 开数字滤波 **Note：** STMR1/2 共用,只在 STMR1 设定，STMR2 共用 STMR1 设定。	RW	0x0
6	THAFILTEREN	**STMR1 刹车输入 CHA 滤波控制** 0x0：刹车输入 CHA 关数字滤波 0x1：刹车输入 CHA 开数字滤波 **Note：** STMR1/2 共用,只在 STMR1 设定，STMR2 共用 STMR1 设定。	RW	0x0
5	-	-	-	-
4	SELSREG	**STMR1 读影子寄存器控制** 0x0：读取寄存器 PR/CMPA/CMPB 得到 PR_S/CMPA_S/CMPB_S 寄存器的值 0x1：读取寄存器 PR/CMPA/CMPB 得到 PR/CMPA/CMPB 寄存器的值	RW	0x0

3	DIR	**STMR1 计数方向控制** 0x0：向下计数 0x1：向上计数	RW	0x1
2:1	MODE	**STMR1 计数器计数模式** 0x0：锯齿波计数模式 0x1：三角波 A 计数模式 0x2：三角波 B 计数模式 0x3：保留	RW	0x0
0	TMREN	**STMR1 计数使能控制** 0x0：关闭 STMR1 计数 0x1：打开 STMR1 计数	RW	0x0

14.4.2. STMR1_FCONR

Addr = 0xD2（SFR）

Bit(s)	Name	Description	R/W	Reset
7:6	INCSEL	**STMR1 计数源** 0x0：SYS_CLK 0x1：PIN_SEL 上升沿 0x2：PIN_SEL 下降沿 0x3：PIN_SEL 边沿（上升和下降）	RW	0x0
5:4	PINSEL	**STMR1 计数时钟源** 0x0：XOSCM/2 0x1：32KHz RC 0x2：CHA 输入 0x3：CHB 输入	RW	0x0
3:0	PREDIV	**STMR1 计数预分频** 0~15 对应 1~16 分频	RW	0x0

14.4.3. STMR1_CNTL

Addr = 0xC8 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	STMR1CNTL	STMR1 计数寄存器低 8 位	RW	0x0

14.4.4. STMR1_CNTH

Addr = 0xC9 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	STMR1CNTH	STMR1 计数寄存器高 8 位	RW	0x0

14.4.5. STMR1_PRL

Addr = 0xCA (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	STMR1PRL	STMR1 周期寄存器低 8 位	RW	0xFF

14.4.6. STMR1_PRH

Addr = 0xCB (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	STMR1PRH	STMR1 周期寄存器高 8 位	RW	0xFF

14.4.7. STMR1_CMPAL

Addr = 0xCC (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	STMR1CMPAL	STMR1 CHA 比较值寄存器低 8 位	RW	0x0

14.4.8. STMR1_CMPAH

Addr = 0xCD (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	STMR1CMPAH	STMR1 CHA 比较值寄存器高 8 位	RW	0x0

14.4.9. STMR1_CMPBL

Addr = 0xCE (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	STMR1CMPBL	STMR1 CHB 比较值寄存器低 8 位	RW	0x0

14.4.10. STMR1_CMPBH

Addr = 0xCF (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	STMR1CMPBH	STMR1 CHB 比较值寄存器高 8 位	RW	0x0

14.4.11. STMR1_VPERR

Addr = 0xD3 (SFR)

Bit(s)	Name	Description	R/W	Reset

7	BRAKEASF	**STMR1 CHA 软件刹车信号** 0x0：CHA 软件刹车信号无效 0x1：CHA 软件刹车信号有效	RW	0x0
6:5	BRAKEAVAL	**STMR1 CHA 刹车输出值** 0x0：刹车事件有效时，CHA PWM 输出 0 0x1：刹车事件有效时，CHA PWM 输出 1 0x2：刹车事件有效时，CHA PWM 输出关闭 0x3：刹车事件有效时，CHA PWM 输出关闭	RW	0x0
4:3	PCNTE	**STMR1 周期间隔响应计数条件** 0x0：周期间隔功能无效 0x1：锯齿波上或下溢点或三角波波谷 0x2：锯齿波上或下溢点或三角波波峰 0x3：锯齿波上或下溢点或三角波波峰和波谷	RW	0x0
2:0	PCNTS	**STMR1 周期间隔响应时长** 0x0：1 个周期响应一次 0x1：2 个周期响应一次 0x2：4 个周期响应一次 0x3：8 个周期响应一次 0x4：16 个周期响应一次 0x5：32 个周期响应一次 0x6：64 个周期响应一次 0x7：128 个周期响应一次	RW	0x0

14.4.12. STMR1_DTUA

Addr = 0xD4（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	STMR1DTUA	**STMR1 死区时间设定寄存器**	RW	0x0

14.4.13. STMR1_BRAKE

Addr = 0xD5 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	T1BMOE	**STMR1 CHB 刹车事件后，PWM 正常输出标志** 0x1：CHB PWM 恢复正常输出 0x0：CHB PWM 按照刹车配置输出 **Note：** 刹车事件有效时，会立即被同步清零；当刹车有效信号消失后，根据 T1BAOE 的选择，通过软件置 1 或硬件自动置 1。	RW	0x1
6	T1BAOE	**STMR1 CHB PWM 正常输出标志控制位** 0x0：有效刹车事件后，T1BMOE 只能软件置 1 0x1：有效刹车事件后，T1BMOE 可被软件和溢出置 1	RW	0x0
5	T1BSEL	**STMR1 CHB 刹车源** 0x0：CHB 选择模拟比较器 0 输出作为刹车源 0x1：CHB 选择模拟比较器 1 输出作为刹车源	RW	0x0
4	T1BEN	**STMR1 CHB 刹车使能** 0x0：CHB 刹车不使能 0x1：CHB 刹车使能	RW	0x0
3	T1AMOE	**STMR1 CHA 刹车事件后，PWM 正常输出标志** 0x0：CHA PWM 按照刹车配置输出 0x1：CHA PWM 恢复正常输出 **Note：** 刹车事件有效时，会立即被同步清零；当刹车有效信号消失后，根据 T1AAOE 的选择，通过软件置 1 或硬件自动置 1。	RW	0x1
2	T1AAOE	**STMR1 CHA PWM 正常输出标志控制位** 0x0：有效刹车事件后，T1AMOE 只能软件置 1 0x1：有效刹车事件后，T1AMOE 可被软件和溢出置 1	RW	0x0
1	T1ASEL	**STMR1 CHA 刹车源** 0x0：CHA 选择模拟比较器 0 输出作为刹车源	RW	0x0

		0x1：CHA 选择模拟比较器 1 输出作为刹车源		
0	T1AEN	**STMR1 CHA 刹车使能** 0x0：CHA 刹车不使能 0x1：CHA 刹车使能	RW	0x0

14.4.14. STMR1_DTR

Addr = 0xD6（SFR）

Bit(s)	Name	Description	R/W	Reset
7	BRAKEBSF	**STMR1 CHB 软件刹车信号** 0x0：CHB 软件刹车信号无效 0x1：CHB 软件刹车信号有效	RW	0x0
6:5	BRAKEBVAL	**STMR1 CHB 刹车输出值** 0x0：刹车事件有效时，CHB PWM 输出 0 0x1：刹车事件有效时，CHB PWM 输出 1 0x2：刹车事件有效时，CHB PWM 输出关闭 0x3：刹车事件有效时，CHB PWM 输出关闭	RW	0x0
4	HWCPWM	**STMR1 互补模式下硬件设定 CMPB_S 寄存器使能** 0x0：硬件设定 CMPB_S 寄存器不使能 0x1：硬件设定 CMPB_S 寄存器使能	RW	0x0
3	DTBHO	**STMR1 互补模式下 CHB 死区输出值** 0x1：CHB PWM 死区输出关闭 0x0：CHB PWM 死区正常输出	RW	0x0
2	DTBEN	**STMR1 CHB 死区使能** 0x0：死区设置无效 0x1：死区设置有效	RW	0x0
1	DTAHO	**STMR1 互补模式下 CHA 死区输出值** 0x0：CHA PWM 死区正常输出 0x1：CHA PWM 死区输出关闭	RW	0x0
0	DTAEN	**STMR1 CHA 死区使能**	RW	0x0

		0x0：死区设置无效 0x1：死区设置有效		

14.4.15. STMR1_PCONRA

Addr = 0xD7（SFR）

Bit(s)	Name	Description	R/W	Reset
7	PAINITVAL	**STMR1 CHA PWM 初始输出值** 0x0：CHA 计数使能后 PWM 初始状态为 0 0x1：CHA 计数使能后 PWM 初始状态为 1	RW	0x0
6:5	CMPAVAL	**STMR1 CHA PWM 输出值** 0x0：计数值小于 CHA 比较值输出 1，大于输出 0 0x1：计数值大于 CHA 比较值输出 1，小于输出 0 0x2：计数值等于 CHA 比较值，输出翻转 0x3：输出保持不变	RW	0x0
4	PAENO	**STMR1 CHA PWM 输出使能** 0x0：CHA PWM 输出不使能 0x1：CHA PWM 输出使能	RW	0x0
3	PAFILTEREN	**STMR1 CHA 输入滤波使能** 0x0：CHA 输入信号不滤波 0x1：CHA 输入信号滤波	RW	0x0
2:1	CAPAMODE	**STMR1 CHA 捕获点选择** 0x0：不捕获 0x1：捕获上升沿 0x2：捕获下降沿 0x3：捕获边沿（上升沿和下降沿）	RW	0x0
0	CAPAEN	**STMR1 CHA 捕获模式使能** 0x0：捕获模式不使能 0x1：捕获模式使能	RW	0x0

14.4.16. STMR1_PCONRB

Addr = 0xD8（SFR）

Bit(s)	Name	Description	R/W	Reset
7	PBINITVAL	**STMR1 CHB PWM 初始输出值** 0x0：CHB 计数使能后 PWM 初始状态为 0 0x1：CHB 计数使能后 PWM 初始状态为 1	RW	0x0
6:5	CMPBVAL	**STMR1 CHB PWM 输出值** 0x0：计数值小于 CHB 比较值输出 1，大于输出 0 0x1：计数值大于 CHB 比较值输出 1，小于输出 0 0x2：计数值等于 CHB 比较值，输出翻转 0x3：输出保持不变	RW	0x0
4	PBENO	**STMR1 CHB PWM 输出使能** 0x0：CHB PWM 输出不使能 0x1：CHB PWM 输出使能	RW	0x0
3	PBFILTEREN	**STMR1 CHB 输入滤波使能** 0x0：CHB 输入信号不滤波 0x1：CHB 输入信号滤波	RW	0x0
2:1	CAPBMODE	**STMR1 CHB 捕获点选择** 0x0：不捕获 0x1：捕获上升沿 0x2：捕获下降沿 0x3：捕获边沿（上升沿和下降沿）	RW	0x0
0	CAPBEN	**STMR1 CHB 捕获模式使能** 0x0：捕获模式不使能 0x1：捕获模式使能	RW	0x0

14.4.17. STMR1_IE

Addr = 0xD9（SFR）

Bit(s)	Name	Description	R/W	Reset
7:6	-	-	-	-
5	BRAKEBIE	**STMR1 CHB 刹车中断使能** 0x0：CHB 刹车中断不使能 0x1：CHB 刹车中断使能	RW	0x0
4	BRAKEAIE	**STMR1 CHA 刹车中断使能** 0x0：CHA 刹车中断不使能 0x1：CHA 刹车中断使能	RW	0x0
3	CMPBIE	**STMR1 计数值等于 CHB 比较值/发生捕获中断使能** 0x0：计数值等于 CHB 比较值/发生捕获，中断不使能 0x1：计数值等于 CHB 比较值/发生捕获，中断使能	RW	0x0
2	CMPAIE	**STMR1 计数值等于 CHA 比较值/发生捕获中断使能** 0x0：计数值等于 CHA 比较值/发生捕获，中断不使能 0x1：计数值等于 CHA 比较值/发生捕获，中断使能	RW	0x0
1	UDIE	**STMR1 计数值等于 0 中断使能** 0x0：计数值等于 0 中断不使能 0x1：计数值等于 0 中断使能 **Note：**锯齿波模式向上计数没有 0 中断	RW	0x0
0	OVIE	**STMR1 计数值等于周期中断使能** 0x0：计数值等于周期中断不使能 0x1：计数值等于周期中断使能 **Note：**锯齿波模式向下计数没有周期中断	RW	0x0

14.4.18. STMR1_SR

Addr = 0xDA（SFR）

Bit(s)	Name	Description	R/W	Reset

7:6	-	-	-	-
5	BRAKEBIF	**STMR1 CHB 刹车中断标志** 0x0：CHB 有效刹车没有发生 0x1：CHB 有效刹车已经发生 **Note**：写 1 清零，写 0 无效。读取为标志状态。	RW	0x0
4	BRAKEAIF	**STMR1 CHA 刹车中断标志** 0x0：CHA 有效刹车没有发生 0x1：CHA 有效刹车已经发生 **Note**：写 1 清零，写 0 无效。读取为标志状态。	RW	0x0
3	CMPBIF	**STMR1 计数值等于 CHB 比较值/发生捕获中断标志** 0x0：计数值等于 CHB 比较值/捕获没有发生 0x1：计数值等于 CHB 比较值/捕获已经发生 **Note**：写 1 清零，写 0 无效。读取为标志状态。	RW	0x0
2	CMPAIF	**STMR1 计数值等于 CHA 比较值/发生捕获中断标志** 0x0：计数值等于 CHA 比较值/捕获没有发生 0x1：计数值等于 CHA 比较值/捕获已经发生 **Note**：写 1 清零，写 0 无效。读取为标志状态。	RW	0x0
1	UDIF	**STMR1 计数值等于 0 中断标志** 0x0：计数值等于 0 没有发生 0x1：计数值等于 0 已经发生 **Note**：写 1 清零，写 0 无效。读取为标志状态。	RW	0x0
0	OVIF	**STMR1 计数值等于周期中断标志** 0x0：计数值等于周期没有发生 0x1：计数值等于周期已经发生 **Note**：写 1 清零，写 0 无效。读取为标志状态。	RW	0x0

14.4.19. STMR2_CR

Addr = 0xE4 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:6	-	-	-	-

5	CAPTMR1	**STMR2 捕捉 STMR1 刹车滤波** 0x0：STMR2 不捕捉 STMR1 刹车滤波控制 0x1：STMR2 捕捉 STMR1 刹车滤波控制 **Note：**捕捉后 STMR2 刹车滤波和 STMR1 刹车滤波设置同步。	RW	0x0
4	SELSREG	**STMR2 读影子寄存器控制** 0x0：读寄存器 PR/CMPA/CMPB 得到 PR_S/CMPA_S/CMPB_S 寄存器的值 0x1：读寄存器 PR/CMPA/CMPB 得到 PR/CMPA/CMPB 寄存器的值	RW	0x0
3	DIR	**STMR2 计数方向** 0x0：向下计数 0x1：向上计数	RW	0x1
2:1	MODE	**STMR2 计数器计数模式** 0x0：锯齿波计数模式 0x1：三角波 A 计数模式 0x2：三角波 B 计数模式 0x3：保留	RW	0x0
0	TMREN	**STMR2 计数使能** 0x0：STMR2 计数不使能 0x1：STMR2 计数使能	RW	0x0

14.4.20. STMR2_FCONR

Addr = 0xE5（SFR）

Bit(s)	Name	Description	R/W	Reset
7:6	INCSEL	**STMR2 计数源** 0x0：SYS_CLK 0x1：PIN_SEL 上升沿 0x2：PIN_SEL 下降沿 0x3：PIN_SEL 边沿（上升和下降）	RW	0x0
5:4	PINSEL	**STMR2 计数时钟源**	RW	0x0

		0x0：XOSCM/2 0x1：32KHz RC 0x2：CHA 输入 0x3：CHB 输入		
3:0	PREDIV	**STMR2 计数预分频** 0~15 对应 1~16 分频	RW	0x0

14.4.21. STMR2_PRL

Addr = 0xDD (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	STMR2PRL	**STMR2 周期寄存器低 8 位**	RW	0xFF

14.4.22. STMR2_PRH

Addr = 0xDE (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	STMR2PRH	**STMR2 周期寄存器高 8 位**	RW	0xFF

14.4.23. STMR2_CMPAL

Addr = 0xDF (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	STMR2CMPAL	**STMR2 CHA 比较值寄存器低 8 位**	RW	0x0

14.4.24. STMR2_CMPAH

Addr = 0xE1 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	STMR2CMPAH	**STMR2 CHA 比较值寄存器高 8 位**	RW	0x0

14.4.25. STMR2_CMPBL

Addr = 0xE2 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	STMR2CMPBL	**STMR2 CHB 比较值寄存器低 8 位**	RW	0x0

14.4.26. STMR2_CMPBH

Addr = 0xE3 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	STMR2_CMPBH	**STMR2 CHB 比较值寄存器高 8 位**	RW	0x0

14.4.27. STMR2_VPERR

Addr = 0xE6 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	BRAKEASF	**STMR2 CHA 软件刹车信号** 0x0：CHA 软件刹车信号无效 0x1：CHA 软件刹车信号有效	RW	0x0
6:5	BRAKEAVAL	**STMR2 CHA 刹车输出值** 0x0：刹车事件有效时，CHB PWM 输出 0	RW	0x0

		0x1：刹车事件有效时，CHB PWM 输出 1 0x2：刹车事件有效时，CHB PWM 输出关闭 0x3：刹车事件有效时，CHB PWM 输出关闭		
4:3	PCNTE	**STMR2 周期间隔响应计数条件** 0x0：周期间隔功能无效 0x1：锯齿波上或下溢点或三角波波谷 0x2：锯齿波上或下溢点或三角波波峰 0x3：锯齿波上或下溢点或三角波波峰和波谷	RW	0x0
2:0	PCNTS	**STMR2 周期间隔响应时长** 0x0：1 个周期响应一次 0x1：2 个周期响应一次 0x2：4 个周期响应一次 0x3：8 个周期响应一次 0x4：16 个周期响应一次 0x5：32 个周期响应一次 0x6：64 个周期响应一次 0x7：128 个周期响应一次	RW	0x0

14.4.28. STMR2_DTUA

Addr = 0xE7 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	STMR2_DTUA	**STMR2 死区时间设定寄存器**	RW	0x0

14.4.29. STMR2_BRAKE

Addr = 0xE8 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	T1BMOE	**STMR2 CHB 刹车事件后，PWM 正常输出标志**	RW	0x1

		0x0：CHB PWM 按照刹车配置输出 0x1：CHB PWM 恢复正常输出 Note：刹车事件有效时，会立即被同步清零；当刹车有效信号消失后，根据 T1BAOE 的选择，通过软件置 1 或硬件自动置 1。		
6	T1BAOE	**STMR2 CHB PWM 正常输出标志控制位** 0x0：有效刹车事件后，T1BMOE 只能软件置 1 0x1：有效刹车事件后，T1BMOE 可被软件和溢出置 1	RW	0x0
5	T1BSEL	**STMR2 CHB 刹车源** 0x0：CHB 选择模拟比较器 0 输出作为刹车源 0x1：CHB 选择模拟比较器 1 输出作为刹车源	RW	0x0
4	T1BEN	**STMR2 CHB 刹车使能** 0x0：CHB 刹车不使能 0x1：CHB 刹车使能	RW	0x0
3	T1AMOE	**STMR2 CHA 刹车事件后，PWM 正常输出标志** 0x1：CHA PWM 恢复正常输出 0x0：CHA PWM 按照刹车配置输出 Note：刹车事件有效时，会立即被同步清零；当刹车有效信号消失后，根据 T1AAOE 的选择，通过软件置 1 或硬件自动置 1。	RW	0x1
2	T1AAOE	**STMR2 CHA PWM 正常输出标志控制位** 0x0：有效刹车事件后，T1AMOE 只能软件置 1 0x1：有效刹车事件后，T1AMOE 可被软件和溢出置 1	RW	0x0
1	T1ASEL	**STMR2 CHA 刹车源** 0x0：CHA 选择模拟比较器 0 输出作为刹车源 0x1：CHA 选择模拟比较器 1 输出作为刹车源	RW	0x0
0	T1AEN	**STMR2 CHA 刹车使能** 0x0：CHA 刹车不使能 0x1：CHA 刹车使能	RW	0x0

14.4.30. STMR2_DTR

Addr = 0xE9 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	BRAKEBSF	**STMR2 CHB 软件刹车信号** 0x0：CHB 软件刹车信号无效 0x1：CHB 软件刹车信号有效	RW	0x0
6:5	BRAKEBVAL	**STMR2 CHB 刹车输出值** 0x0：刹车事件有效时，CHB PWM 输出 0 0x1：刹车事件有效时，CHB PWM 输出 1 0x2：刹车事件有效时，CHB PWM 输出关闭 0x3：刹车事件有效时，CHB PWM 输出关闭	RW	0x0
4	HWCPWM	**STMR2 互补模式下硬件设定 CMPB_S 寄存器使能** 0x0：硬件设定 CMPB_S 寄存器不使能 0x1：硬件设定 CMPB_S 寄存器使能	RW	0x0
3	DTBHO	**STMR2 互补模式下 CHB 死区输出值** 0x0：CHB PWM 死区正常输出 0x1：CHB PWM 死区输出关闭	RW	0x0
2	DTBEN	**STMR2 CHB 死区使能** 0x0：死区设置无效 0x1：死区设置有效	RW	0x0
1	DTAHO	**STMR2 互补模式下 CHA 死区输出值** 0x1：CHA PWM 死区输出关闭 0x0：CHA PWM 死区正常输出	RW	0x0
0	DTAEN	**STMR2 CHA 死区使能** 0x0：死区设置无效 0x1：死区设置有效	RW	0x0

14.4.31. STMR2_PCONRA

Addr = 0xEA (SFR)

Bit(s)	Name	Description	R/W	Reset
7	PAINITVAL	**STMR2 CHA PWM 初始输出值** 0x0：CHA 计数使能后 PWM 初始状态为 0 0x1：CHA 计数使能后 PWM 初始状态为 1	RW	0x0
6:5	CMPAVAL	**STMR2 CHA PWM 输出值** 0x0：计数值小于 CHA 比较值输出 1，大于输出 0 0x1：计数值大于 CHA 比较值输出 1，小于输出 0 0x2：计数值等于 CHA 比较值，输出翻转 0x3：输出保持不变	RW	0x0
4	PAENO	**STMR2 CHA PWM 输出使能** 0x0：CHA PWM 输出不使能 0x1：CHA PWM 输出使能	RW	0x0
3	PAFILTEREN	**STMR2 CHA 输入滤波使能** 0x0：CHA 输入信号不滤波 0x1：CHA 输入信号滤波	RW	0x0
2:1	CAPAMODE	**STMR2 CHA 捕获点选择** 0x0：不捕获 0x1：捕获上升沿 0x2：捕获下降沿 0x3：捕获边沿（上升沿和下降沿）	RW	0x0
0	CAPAEN	**STMR2 CHA 捕获模式使能** 0x0：捕获模式不使能 0x1：捕获模式使能	RW	0x0

14.4.32. STMR2_PCONRB

Addr = 0xEB（SFR）

Bit(s)	Name	Description	R/W	Reset
7	PBINITVAL	**STMR2 CHB PWM 初始输出值** 0x0：CHB 计数使能后 PWM 初始状态为 0 0x1：CHB 计数使能后 PWM 初始状态为 1	RW	0x0

6:5	CMPBVAL	**STMR2 CHB PWM 输出值** 0x0：计数值小于 CHB 比较值输出 1，大于输出 0 0x1：计数值大于 CHB 比较值输出 1，小于输出 0 0x2：计数值等于 CHB 比较值，输出翻转 0x3：输出保持不变	RW	0x0
4	PBENO	**STMR2 CHB PWM 输出使能** 0x0：CHB PWM 输出不使能 0x1：CHB PWM 输出使能	RW	0x0
3	PBFILTEREN	**STMR2 CHB 输入滤波使能** 0x0：CHB 输入信号不滤波 0x1：CHB 输入信号滤波	RW	0x0
2:1	CAPBMODE	**STMR2 CHB 捕获点选择** 0x0：不捕获 0x1：捕获上升沿 0x2：捕获下降沿 0x3：捕获边沿（上升沿和下降沿）	RW	0x0
0	CAPBEN	**STMR2 CHB 捕获模式使能** 0x0：捕获模式不使能 0x1：捕获模式使能	RW	0x0

14.4.33. STMR2_IE

Addr = 0xEC（SFR）

Bit(s)	Name	Description	R/W	Reset
7:6	-	-	-	-
5	BRAKEBIE	**STMR2 CHB 刹车中断使能** 0x0：CHB 刹车中断不使能 0x1：CHB 刹车中断使能	RW	0x0
4	BRAKEAIE	**STMR2 CHA 刹车中断使能** 0x0：CHA 刹车中断不使能 0x1：CHA 刹车中断使能	RW	0x0
3	CMPBIE	**STMR2 CHB 计数值等于比较值/发生捕获中断使**	RW	0x0

		能 0x0：计数值等于 CHB 比较值/发生捕获，中断不使能 0x1：计数值等于 CHB 比较值/发生捕获，中断使能		
2	CMPAIE	**STMR2 CHA 计数值等于比较值/发生捕获中断使能** 0x0：计数值等于 CHA 比较值/发生捕获，中断不使能 0x1：计数值等于 CHA 比较值/发生捕获，中断使能	RW	0x0
1	UDIE	**STMR2 计数值等于 0 中断使能** 0x0：计数值等于 0 中断不使能 0x1：计数值等于 0 中断使能 **Note：**锯齿波模式向上计数没有 0 中断	RW	0x0
0	OVIE	**STMR2 计数值等于周期中断使能** 0x0：计数值等于周期中断不使能 0x1：计数值等于周期中断使能 **Note：**锯齿波模式向下计数没有周期中断	RW	0x0

14.4.34. STMR2_SR

Addr = 0xED（SFR）

Bit(s)	Name	Description	R/W	Reset
7:6	-	-	-	-
5	BRAKEBIF	**STMR2 CHB 刹车中断标志** 0x0：CHB 有效刹车没有发生 0x1：CHB 有效刹车已经发生 **Note：**写 1 清零，写 0 无效。读取为标志状态。	RW	0x0
4	BRAKEAIF	**STMR2 CHA 刹车中断标志** 0x0：CHA 有效刹车没有发生	RW	0x0

		0x1：CHA 有效刹车已经发生 Note：写 1 清零，写 0 无效。读取为标志状态。		
3	CMPBIF	**STMR2 CHB 计数值等于比较值/发生捕获中断标志** 0x0：CHB 计数值等于比较值/捕获没有发生 0x1：CHB 计数值等于比较值/捕获已经发生 Note：写 1 清零，写 0 无效。读取为标志状态。	RW	0x0
2	CMPAIF	**STMR2 CHA 计数值等于比较值/发生捕获中断标志** 0x0：CHA 计数值等于比较值/捕获没有发生 0x1：CHA 计数值等于比较值/捕获已经发生 Note：写 1 清零，写 0 无效。读取为标志状态。	RW	0x0
1	UDIF	**STMR2 计数值等于 0 中断标志** 0x0：计数值等于 0 没有发生 0x1：计数值等于 0 已经发生 Note：写 1 清零，写 0 无效。读取为标志状态。	RW	0x0
0	OVIF	**STMR2 计数值等于周期中断标志** 0x0：计数值等于周期没有发生 0x1：计数值等于周期已经发生 Note：写 1 清零，写 0 无效。读取为标志状态。	RW	0x0

14.4.35. STMR_ALLCON

Addr = 0xF5（SFR）

Bit(s)	Name	Description	R/W	Reset
7:6	-	-	-	-
5	SCLRR	**STMR1 和 STMR2 计数同步清零** 写 1 清零，写 0 无效	WO	0x0
4	SSTPR	**STMR1 和 STMR2 同步停止计数** 写 1 有效，写 0 无效	WO	0x0
3	SSTAR	**STMR1 和 STMR2 同步启动计数** 写 1 有效，写 0 无效	WO	0x0
2	TMR2CLRCNT	**TMR2 计数清零**	WO	0x0

		写 1 清零，写 0 无效		
1	TMR1CLRCNT	**TMR1 计数清零** 写 1 清零，写 0 无效	WO	0x0
0	TMR0CLRCNT	**TMR0 计数清零** 写 1 清零，写 0 无效	WO	0x0

14.5. 使用流程说明

- 选择计数模式，计数方向；
- 选择计数时钟源（PIN_SEL），选择计数方式（上升沿，下降沿）INC_SEL；
- 配置分频寄存器，周期寄存器；
- 选择打开捕获模式还是PWM输出模式；
- 如果选择捕获模式，请配置捕获模式相关寄存器；
- 如果选择PWM输出模式，请配置比较器寄存器以及其他PWM输出相关寄存器：包括死区以及刹车相关寄存器；
- 计数使能位置 1；

15. CRC16 模块

15.1. 功能概述

CRC16 功能特点如下：

- 支持CRC16-CCITT-FALSE/CRC16-XMODEM计算
- 每一个系统时钟周期计算 1byte数据
- 支持更改初值，实现对不同的CRC协议的支持

Note：当FLASH使用CRC校验代码的时候，CRC模块暂时无法使用

15.2. 基本功能

CRC16-CCITT-FALSE 协议为 x^16+x^12+x^5+1，初始值为 0xffff，输入数据不取反，输出数据不取反，输出数据异或 0x0。

CRC16-XMODEM 协议为 x^16+x^12+x^5+1，初始值为 0x0，输入数据不取反，输出数据不取反，输出数据异或 0x0。

CRC16 模块没输入一次就进行 1byte 数据的并行 CRC 计算，对比于串行计算 CRC，大大提高了运算速度，CRC16 模块在使用的时候需要注意，在每次使用模块时需要使用 CRC_REG 寄存器配置 CRC16 计算的初始值,初始值为 16 位,所以需要配置两次,初始值只能为 0xffff、0x0，与此同时，CRC16 的运算结果为 16 位，所以在读取结果的时候需要读取两次 CRC_FIFO 寄存器，把两次读取的值拼接成一个 16 位的结果才是真正的 CRC16 的结果。

15.3. 模块框图

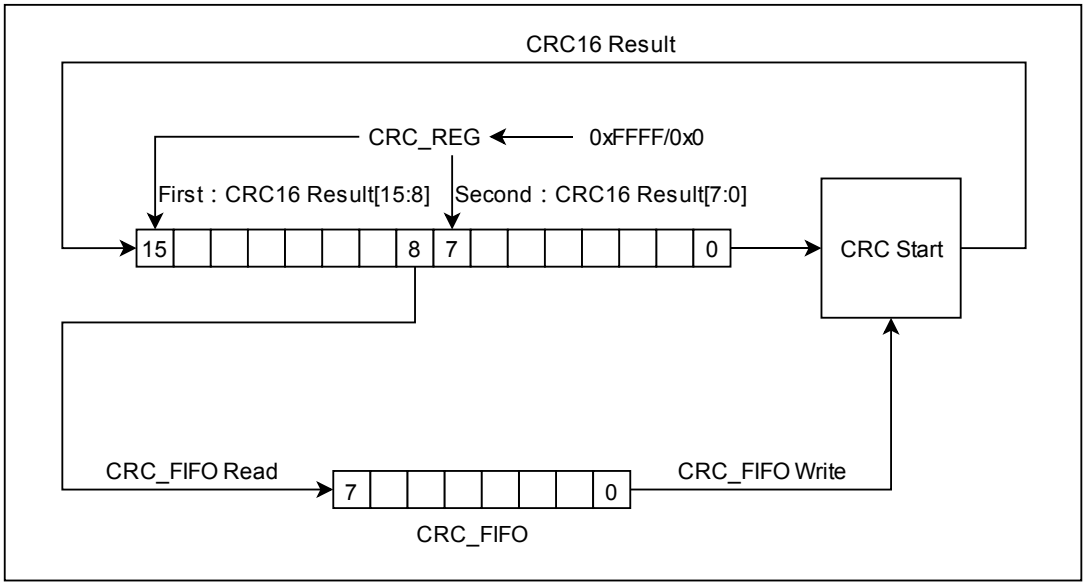


图 15-1 CRC16 结构框图

15.4. 寄存器列表

表 15-1 CRC 寄存器列表

Address	Register Name	Description
0x9E (SFR)	CRC_REG	CRC initial register
0x9F (SFR)	CRC_FIFO	CRC data fifo register

15.5. 寄存器详细说明

15.5.1. CRC_REG

Addr = 0x9E (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	INITSET	**CRC 初始值配置** 每次使用前需要先设置初始值设置 CRC 校验的初始值，默认上电时 0xffff，需要写两次初始值	W	0xFF

15.5.2. CRC_FIFO

Addr = 0x9F (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	DATA	**CRC 数据配置** 写：CRC 每次校验的 1byte 数据 读：读 1 次该寄存器就读 1byteCRC 结果，如果需要读 CRC16 的寄过需要读两次进行拼成 16bit	RW	-

15.6. 使用流程说明

- 配置CRC_REG，一般配置为 0xffff/0x0000（需要写两次这个寄存器）
- 把需要校验的 1byte数据通过CRC_FIFO这个寄存器写进去
- 需要读取结果时，读取两次CRC_FIFO拼成一个CRC16 的结果

16. FLASH 控制器模块

16.1. 功能概述

Logic Flash（以下简称 Flash）功能特点如下：

- Flash控制器带有操作保护功能（在进行Flash操作前需要配置密码寄存器使能）
- 通过配置寄存器可以实现对Flash存储器进行烧录/扇区（128byte）擦除/全片（4K）擦除的功能，同时支持在线烧录
- 通过配置寄存器可以实现对Flash存储器的数据进行CRC校验
- 支持类EEPROM的使用

16.2. 模块框图

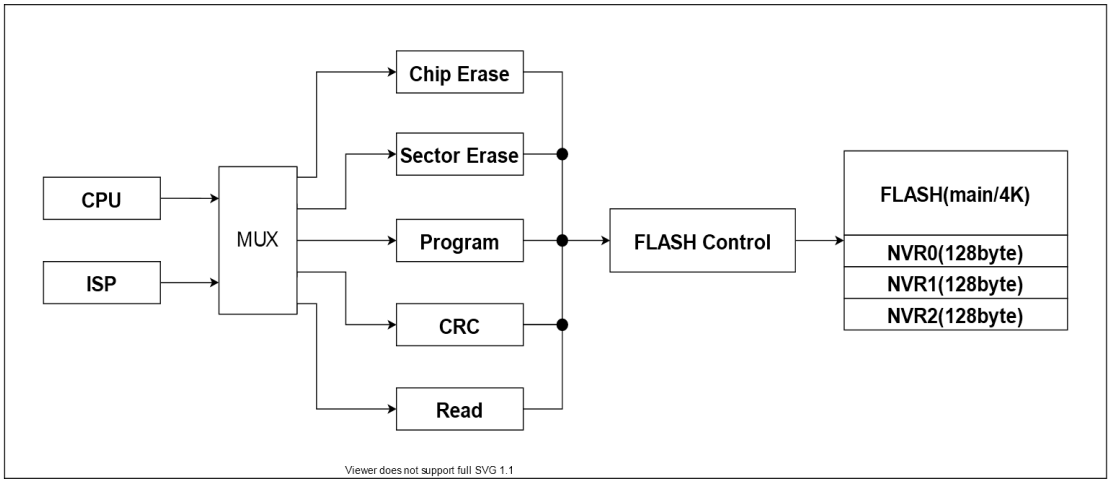


图 16-1 FLASH 结构框图

16.3. 寄存器列表

表 16-1 FLASH 寄存器列表

Address	Register Name	Description
0xA0 (SFR)	FLASH_CON	FLASH control register
0xA1 (SFR)	FLASH_STA	FLASH state register
0xA2 (SFR)	FLASH_DATA	FLASH program data register
0xA3 (SFR)	FLASH_TIM0	FLASH timing control register 0
0xA4 (SFR)	FLASH_TIM1	FLASH timing control register 1
0xA5 (SFR)	FLASH_CRCLEN	FLASH CRC data length register
0xA6 (SFR)	FLASH_PASSWORD	FLASH operation to protect register
0xA7 (SFR)	FLASH_ADDR	FLASH program/erase address register
0xAA (SFR)	FLASH_TRIM	FLASH test mode register

16.4. 寄存器详细说明

16.4.1. FLASH_CON

Addr = 0xA0（SFR）

Bit(s)	Name	Description	R/W	Reset
7:4	-	-	-	-
3	CRCST	**FLASH CRC 功能触发** 写 1 触发 CRC 校验，需要先配置寄存器 FLASH_CRCLEN 的大小才可以触发该操作，且不能在 DATA 区跑 CRC 代码	WO	0x0
2	CERST	**FLASH 全擦功能触发** 写 1 触发全片擦除	WO	0x0
1	SERST	**FLASH 扇区擦除功能触发** 写 1 触发扇区擦除，需要配置 128byte 对齐的地址才可以触发	WO	0x0
0	PROGST	**FLASH 烧写功能触发** 写 1 触发烧录操作	WO	0x0

Note：FLASH 控制器带有操作保护功能，需要先正确配置 FLASH_PASSWORD 寄存器才正确触发以上的操作。

16.4.2. FLASH_STA

Addr = 0xA1（SFR）

Bit(s)	Name	Description	R/W	Reset
7:5	-	-	-	-
4	NVRLOCK	**FLASH 的 information 区标志信号** 0x0：information 区没有通过检验（不能正常使用） 0x1：information 区通过校验（能正常使用）	RO	0x1

3	CRCPEND	**CRC 模式工作标志位** 0x0：正在进行 CRC 校验 0x1：空闲状态	RO	0x1
2	CERPEND	**全片擦除模式工作标志** 0x0：正在进行全片擦除 0x1：空闲状态	RO	0x1
1	SERPEND	**扇区擦除模式工作标志** 0x0：正在进行扇区擦除 0x1：空闲状态	RO	0x1
0	PROGPEND	**烧录模式工作标志** 0x0：正在进行烧录 0x1：空闲状态	RO	0x1

16.4.3. FLASH_DATA

Addr = 0xA2（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	DATA	**FLASH 烧写数据** FLASH 存储器的数据为 16 位，需要操作的时候需要对该寄存器写两次，第一次写低 8 位，第二次写高 8 位	RW	0x0

16.4.4. FLASH_TIM0

Addr = 0xA3（SFR）

Bit(s)	Name	Description	R/W	Reset
7:6	TERH	**扇区擦除时序控制（默认时间为 50ms）** 0x0：默认值 0x1：51ms 0x2：52ms	RW	0x0

		0x3：53ms		
5:4	TPOAM	后同步信号时序控制信号（默认时间为 2us） 0x0：默认值 0x1：3us 0x2：4us 0x3：5us	RW	0x0
3:2	TPRAM	前同步信号时序控制信号（默认时间为 2us） 0x0：默认值 0x1：3us 0x2：4us 0x3：5us	RW	0x0
1:0	TPGHF	烧录时序控制信号（默认时间为 20us） 0x0：默认值 0x1：21us 0x2：22us 0x3：23us	RW	0x0

16.4.5. FLASH_TIM1

Addr = 0xA4（SFR）

Bit(s)	Name	Description	R/W	Reset
7:6	TPEL	连续操作时序控制（默认时间为 2*Tsys） 0x0：默认值 0x1：3*Tsys 0x2：4*Tsys 0x3：5*Tsys	RW	0x0
5:4	TCERH	全片擦除时序控制信号（默认时间为 50ms） 0x0：默认值 0x1：51ms 0x2：52ms 0x3：53ms	RW	0x0

3:2	TKP	**读周期时序控制信号** 0x0：默认值 0x1：Tsys 0x2：2*Tsys 0x3：3*Tsys	RW	0x0
1:0	TKH	**读信号高电平时序控制信号** 0x0：默认值 0x1：Tsys 0x2：2*Tsys 0x3：3*Tsys	RW	0x0

Note： TKH 应配置大于等于 TKP 的值（TKP 和 TKH 的配置请查看具体的配置表）

16.4.6. FLASH_CRCLEN

Addr = 0xA5（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	LEN	**CRC 操作的数据长度** CRC 校验操作的数据大小（单位：字节）	RW	0x0

Note： 当 FLASH_CRCLEN 的配置值为 0 时，CRC 操作无法触发！

16.4.7. FLASH_PSWD

Addr = 0xA6（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	PASSWORD	**FLASH 操作保护的密码寄存器** 操作保护密码，执行 FLASH_CON 的操作之前需要配置该寄存器，密码为 0xB9	WO	0x0

16.4.8. FLASH_ADDR

Addr = 0xA7（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	ADDR	**FLASH 的地址寄存器** FLASH 存储器的地址为 16 位，有效位 12 位，需要操作的时候需要对该寄存器写两次，第一次写高 8 位，第二次写低 8 位	WO	0x0

Note：

1. 进行 FLASH 扇区擦除时，需要将地址对齐（128 bytes），否则无法进行操作。
2. 进行 FLASH 烧录时，FLASH Control 有基地址偏移（偏移量为 0x8000），换算公式为：

   FLASH_ADDR = (W_ADDR - 0x8000)/2

   其中 W_ADDR 为 Flash 物理地址，FLASH_ADDR 为寄存器，Flash 位宽是 16bit，所以地址需要除以 2。

   假设要存储的地址为 0x9000，则 FLASH_ADDR = (0x9000 - 0x8000)/2=0x800。

16.4.9. FLASH_TRIM

Addr = 0xAA（SFR）

Bit(s)	Name	Description	R/W	Reset
7:6	-	-	-	-
5	MODESEL	**测试模式下的 MODESEL 信号**	RW	0x0
4:3	TRIM	**测试模式下的 TRIM 信号**	RW	0x2
2:1	VRDCGSEL	**测试模式下的 VRDCGSEL 信号**	RW	0x1
0	TRF	**测试模式下的 TRF 信号**	RW	0x0

16.5. 使用流程说明

- 配置需要操作的FLASH的地址（需对齐对应的地址才可以触发对应的操作）
- 配置需要操作的FLASH的数据
- 配置FLASH_PASSWORD寄存器使能
- 配置FLASH_CON触发对应的FLASH操作
- 等FLASH_STA对应的pending为高，结束一次FLASH操作

17. 模数转换器(ADC)

17.1. 功能概述

该模块是一个12bit的逐次逼近式的ADC控制器。ADC支持多种工作模式：单次转换/两通道交替触发，可以选择所有的GPIO和内部通道进行转换，ADC的启动包括软件启动、以及高级Timer1/2触发启动。

ADC 的输入时钟不得超过 4Mhz 是由系统时钟经分频产生。

- ADC具有 12BIT量化，可选内部参考电压或外部参考电压作为ADC的参考电压，内部参考电压为 1.2V和 2.4V可选；外部参考电压默认为VCC
- 可以通过ADC的通路选择把Analog测试信号，输入到任意I/O口
- 转换速度可达 200Ksps
- 采样时间可调，可调范围：5~256 个ADC时钟
- 支持单通道配置延迟触发
- 比较器offset可以使用模拟/数字校准
- 支持双通道带仲裁触发（两个通道都不开启硬件延迟功能）
- 支持 1 路软件触发和 7 路硬件触发

- 支持所有GPIO和内部通道作为模拟转换通道

17.2. 基本功能

17.2.1. 单通道触发模式

单通道触发模式是只使能 ADC_CFG0 中的 CHAN0EN/CHAN1EN，如果只使能通道 0 时，只有配置的 ADC_CHS0 的通道 0 的触发源可以触发 ADC 的转换，而且此时 ADC 使用的是 ADC_CHS0 的模拟转换通道，当然，只使能通道 1 同理。

17.2.2. 双通道触发模式

双通道触发模式是同时使能 ADC_CFG0 中 CHAN0EN/CHAN1EN，在这种模式下，通道 0 的优先级高于通道 1 的优先级，在这种时候一般会出现以下两种情况：

- 仲裁：当 ADC_CHS0 和 ADC_CHS1 中的配置的触发源在同一时刻触发 ADC 时，会先响应 ADC_CHS0 的配置的触发源，同时采用 ADC_CHS0 中对应的模拟采样通道，再响应完后通道 0 的触发后才响应通道 1 的触发，响应通道 1 的触发时使用 ADC_CHS1 中对应的模拟采样通道
- 排队：在通道 0 进行转换时，此时如果再来通道 0 的触发将会忽视，不会响应也不会进行锁存，如果此时来通道 1 的触发将会锁存通道 1 的请求，等到通道 0 转换完之后再进行通道 1 的转换，当然，在通道 1 进行转换是也是同理，来通道 1 的触发将会忽视，但是如果此时来通道 0 的触发，会锁存对应的请求，等到通道 1 转换完成后才会去转换通道 0。

17.2.3. 单通道触发延迟模式

单通道触发延迟模式是只使能 ADC_CFG0 中的 CHAN0EN&DLY0EN/CHAN1EN&DLY1EN，在这种模式下当通道 0 在 ADC_CHS0 配置的触发源来了之后，并不是马上开始 ADC 转换，需要等待 ADC_CFG3 中配置的延迟时间后才会去进行 ADC 转换，与此同时，在这种模式下，ADC 的两个通道没有仲裁机制，所以在这种模式下建议不要使能另一个通道进行转换。

17.2.4. 模拟校准/数字校准

模拟校准需要先使能 ADC_CFG0 中 CALIBEN，然后将通道 0 配置成单通道触发模式，进行一次软件触发后，将转换后的结果的低 6 位写到 comp_trim_vdd，然后再关闭 CALIBEN，并且清楚 ADC_STA 中通道 0 完成标志和模拟校准标志，完成模拟校准。

数字校准只需要 ADC_CFG1 的 CPCALIBST，等待 ADC_STA 中的数字校准标志位后，清除数字校准标志位后完成数字校准。

17.3. 模块框图

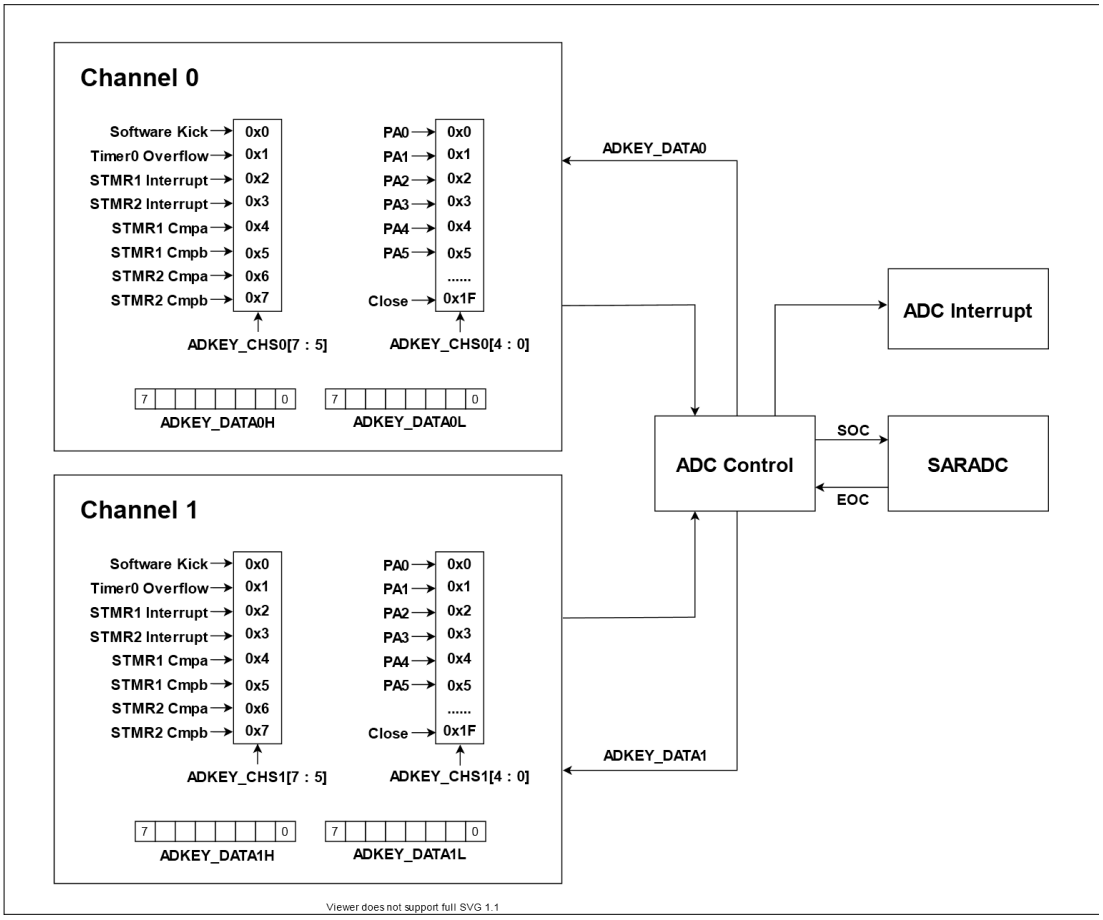


图 17-1 ADC 结构框图

17.4. 寄存器列表

表 17-1 ADC 寄存器列表

Address	Register Name	Description
0x91 (SFR)	ADC_CFG0	ADC configuration 0 register
0x92 (SFR)	ADC_CFG1	ADC configuration 1 register
0x9A (SFR)	ADC_CFG2	ADC configuration 2 register
0xEE (SFR)	ADC_CFG3	ADC configuration 3 register

0x93 (SFR)	ADC_STA	ADC state register
0x94 (SFR)	ADC_DATAH0	ADC channel0 data high 8bit register
0x95 (SFR)	ADC_DATAL0	ADC channel0 data low 4bit register
0x96 (SFR)	ADC_DATAH1	ADC channel1 data high 8bit register
0x97 (SFR)	ADC_DATAL1	ADC channel1 data low 4bit register
0x98 (SFR)	ADC_CHS0	ADC channel0 select register
0x99 (SFR)	ADC_CHS1	ADC channel1 select register

17.5. 寄存器详细说明

17.5.1. ADC_CFG0

Addr = 0x91 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	CALIBEN	**开启硬件校准** 开启后的通道 0 作为校准结果，填回去 comp_trim_vdd	RW	0x0
6	DLY1EN	**通道 1 ADC 触发延迟功能使能** 0x0：关闭 0x1：打开	RW	0x0
5	DLY0EN	**通道 0 ADC 触发延迟功能使能** 0x0：关闭 0x1：打开	RW	0x0
4	ADCEN	**A/D 转换使能** 0x0：关闭 0x1：打开	RW	0x0
3	CHAN1EN	**通道 1 转换使能** 0x0：关闭	RW	0x0

		0x1：打开		
2	CHAN0EN	**通道 0 转换使能** 0x0：关闭 0x1：打开	RW	0x0
1	ADST1	**ADC 通道 1 触发转换** 写 1 开始触发通道 1 进行转换	WO	0x0
0	ADST0	**ADC 通道 0 触发转换** 写 1 开始触发通道 0 进行转换	WO	0x0

Note：触发延迟功能没有带仲裁功能，所以在使用触发延迟功能的时候不能同时使用两个通道进行转换！

17.5.2. ADC_CFG1

Addr = 0x92（SFR）

Bit(s)	Name	Description	R/W	Reset
7	-	-	-	-
6	CPCALIBST	**数字校准功能触发位** 数字校准功能触发，写 1 触发，触发后会 ADC 模块会进行自动校准，校准完后将校准值填到 comp_trim_vdd	WO	0x0
5:2	ADCPRE	**ADC 时钟分频** 分频比为 n+1，0 的时候默认为 2 分频，ADC 最大输入时钟频率为 4MHz，Fadc=Fsys/ADCPRE，其中 Fadc 是 ADC 输入频率，Fsys 是系统时钟主频 0x0：2 分频 0x1：2 分频 0x2：3 分频 0x3：4 分频 0x4：5 分频 ..... 0xE：15 分频	RW	0x7

		0xF：16 分频		
1	CHAN1IE	**ADC 通道 1 的中断使能** 0x0：关闭 0x1：打开	RW	0x0
0	CHAN0IE	**ADC 通道 0 的中断使能** 0x0：关闭 0x1：打开	RW	0x0

17.5.3. ADC_CFG2

Addr = 0x9A（SFR）

Bit(s)	Name	Description	R/W	Reset
7:0	FSMPCYC	**采样时间配置** 配置比为 n+1 时钟 0x0：1 个 ADC 时钟 0x1：2 个 ADC 时钟 0x2：3 个 ADC 时钟 …… 0xFF：256 个 ADC 时钟	RW	0x4

Note：为了 ADC 能够正常工作，采样时间最少配置为 0x3

17.5.4. ADC_CFG3

Addr = 0xEE（SFR）

Bit(s)	Name	Description	R/W	Reset
7:4	-	-	-	-
3:0	DLYCYC	**触发延迟时间配置** 配置比为 4n+3 个 ADC 时钟 0x0：3 个 ADC 时钟	RW	0x0

		0x1：7 个 ADC 时钟 0x2：11 个 ADC 时钟 ..... 0xF：67 个 ADC 时钟		

17.5.5. ADC_STA

Addr = 0x93（SFR）

Bit(s)	Name	Description	R/W	Reset
7:5	-	-	-	-
4	EODTADCFLAG	**数字校准完成标志** 0x0：数字校准没有完成/没有使能 0x1：数字校准完成 写 1 清 0	RW	0x0
3	EOATADCFLAG	**模拟校准完成标志** 0x0：模拟校准没有完成/没有使能 0x1：模拟校准完成 写 1 清 0	RW	0x0
2	CHAN1OVPEND	**通道 1 转换完成标志** 0x0：通道 1 转换没有完成/没有使能通道 1 0x1：通道 1 转换完成 写 1 清 0	RW	0x0
1	CHAN0OVPEND	**通道 0 转换完成标志** 0x0：通道 0 转换没有完成/没有使能通道 1 0x1：通道 0 转换完成 写 1 清 0	RW	0x0
0	BUSY	**ADC 忙/空闲标志** 0x0：ADC 空闲 0x1：ADC 转换中	RO	0x0

17.5.6. ADC_DATAH0

Addr = 0x94 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	DATA0H	通道 0 的 12 位 A/D 转换结果高 8 位	RW	0x0

17.5.7. ADC_DATAL0

Addr = 0x95 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:4	-	-	-	-
3:0	DATA0L	通道 0 的 12 位 A/D 转换结果低 4 位	RW	0x0

17.5.8. ADC_DATAH1

Addr = 0x96 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:0	DATA1H	通道 1 的 12 位 A/D 转换结果高 8 位	RW	0x0

17.5.9. ADC_DATAL1

Addr = 0x97 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:4	-	-	-	-
3:0	DATA1L	通道 1 的 12 位 A/D 转换结果低 4 位	RW	0x0

17.5.10. ADC_CHS0

Addr = 0x98 (SFR)

Bit(s)	Name	Description	R/W	Reset
7:5	TRGSEL0	**硬件触发源选择** 0x0：选择软件触发 0x1：Timer0ov 0x2：高级 Timer1 中断 0x3：高级 Timer2 中断 0x4：高级 Timer1 CMPA_ACK 0x5：高级 Timer1 CMPB_ACK 0x6：高级 Timer2 CMPA_ACK 0x7：高级 Timer2 CMPB_ACK	RW	0x0
4:0	CHANSEL0	**通道 0 的模拟转换通道选择** 0x0~0x7 分别选择 P00~P07 通路， 0x8~0xD 分别选择 P10~P15 通路： 0x0：P00 0x1：P01 0x2：P02 0x3：P03 0x4：P04 0x5：P05 0x6：P06 0x7：P07 0x8：reserved 0x9：P11 0xA：P12 0xB：P13 0xC：P14 0xD：P15 0xE：reserved 0xF：reserved 0x10：VCCA 的 5 分压值	RW	0x1F

		0x11：analog test out 其他码均不选择/保留		

17.5.11. ADC_CHS1

Addr = 0x99（SFR）

Bit(s)	Name	Description	R/W	Reset
7:5	TRGSEL1	**硬件触发源选择** 0x0：选择软件触发 0x1：Timer0ov 0x2：高级 Timer1 中断 0x3：高级 Timer2 中断 0x4：高级 Timer1 CMPA_ACK 0x5：高级 Timer1 CMPB_ACK 0x6：高级 Timer2 CMPA_ACK 0x7：高级 Timer2 CMPB_ACK	RW	0x0
4:0	CHANSEL1	**通道 1 的模拟转换通道选择** 0x0~0x7 分别选择 P00~P07 通路, 0x8~0xD 分别选择 P10~P15 通路： 0x0：P00 0x1：P01 0x2：P02 0x3：P03 0x4：P04 0x5：P05 0x6：P06 0x7：P07 0x8：reserved 0x9：P11 0xA：P12 0xB：P13 0xC：P14	RW	0x1F

		0xD：P15 0xE：reserved 0xF：reserved 0x10：VCCA 的 5 分压值 0x11：analog test out 其他码均不选择/保留		

17.6. 使用流程说明

- 配置ADC_CFG1 的ADCPRE，设定ADC时钟分频
- 配置ADC_CFG0 使能通道转换
- 配置ADC_CFG2 配置采样时间
- 配置ADC_CHS0/ADC_CHS1 配置模拟转换通道和触发源
- 配置ADC_CFG0 使能ADC
- 等待 20us（等待期间不能触发ADC转换）触发ADC
- 等待ADC_STA中对应的通道的pending为
- ADC_DATAH0/ADC_DATAH1 的转换结果

18. 模拟比较器（CMP0/1）

18.1. 功能概述

比较器功能特点如下：

- DACMP 主要包括 2 个 8 bit flash DAC，2 个比较器，1 路 20mA 恒流源输出
- 每个比较器正端可选择 2 路端口（AIO）输入和 1 路 PGA 输入，负端可选择 2 路端口(AIO)输入和 DAC 输入，其中比较器 0 正端还支持阈值短路(VCCA-VTH 或 PAD-VTH)

保护输入，比较器 1 支持 CCS 采样电压输入

- DAC 的参考电压可选择内部 1.2V 参考，输出为 1.2/240*（1~240）
- 内置 2 路阈值短路保护（可选 VCCA-VTH 或者 P03 PAD-VTH），其中 VTH 档位调节输出电压可选为 80mv/200mv/320mv/480mv
- 1 路 6bit 校准精度为 2.5%的 20mA 恒流源输出，TYP 输出范围：-40.5~+42%
- 均可支持 P 输入对管或者 N 输入对端输入，支持 P 输入管的 offset 电压校准，校准 step 在 typ 下为 2mv，校准范围为-13mv~+13mv
- 支持数字滤波，滤波时间共 32 档位选择，步长为 1 微秒
- 支持数字迟滞控制，迟滞采样间隔 16 个档位可选，步长 1 微秒和 16 微秒可选
- 数字迟滞电压可在 1.2/240*（1~240）内选择
- 支持失调电压软件修调
- 输出可作为增强型 PWM 的刹车触发信号
- 支持输出改变产生中断
- 支持比较器唤醒睡眠状态
- 比较器使能可由内部 PWM 控制

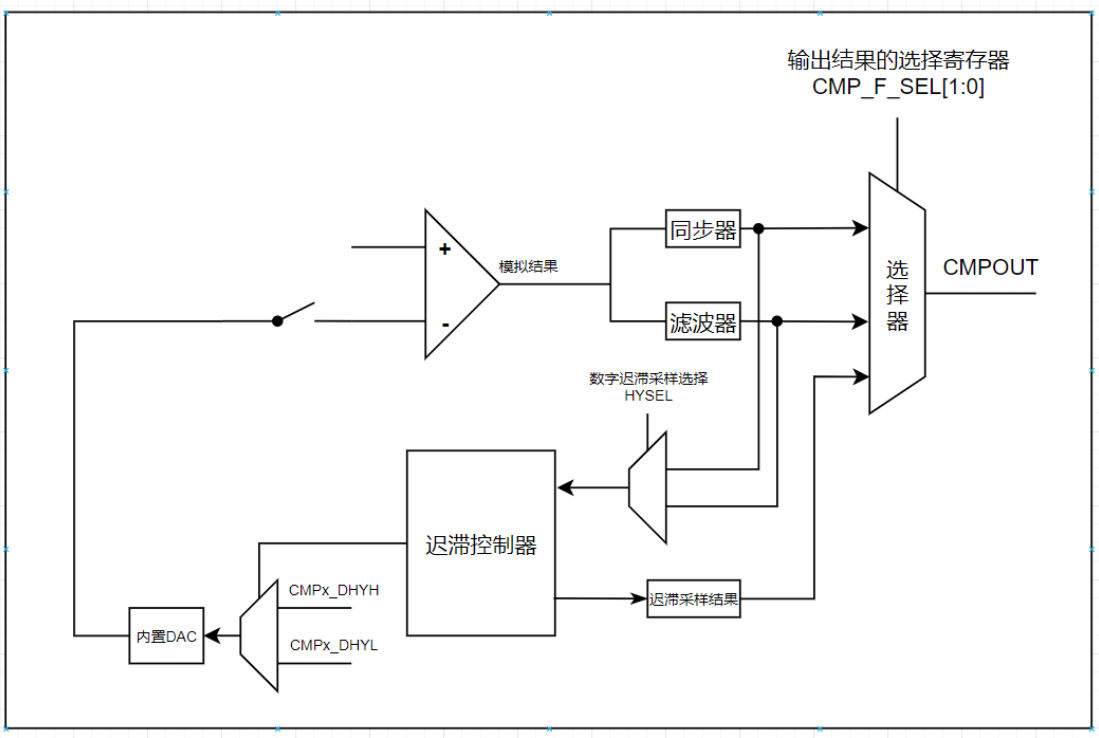
18.2. 功能使用说明

18.2.1. 关于数字滤波功能说明

比较器数字滤波器的逻辑是：比较器的模拟输出结果改变后必须持续数个周期不变，比较器的数字结果才会改变。

当比较器的模拟输出改变后，内部计数器从 0 开始计数，计数每加一都会采样一次模拟的输出结果，直到计数到配置值（CMPx_CON2[7:0]）或者中途发现模拟比较器的输出值变了。

18.2.2. 关于数字迟滞功能说明



比较器的数字迟滞实现方式是：采样比较器的输出值，判断输出是 0 还是 1，根据判断改变选择 DAC 的输入（从 CMPx_DHYH 和 CMPx_DHYL 中选一个,采样到 0 时选 CMPx_DHYL,采样到 1 时选 CMPx_DHYH），从而达到迟滞的效果。

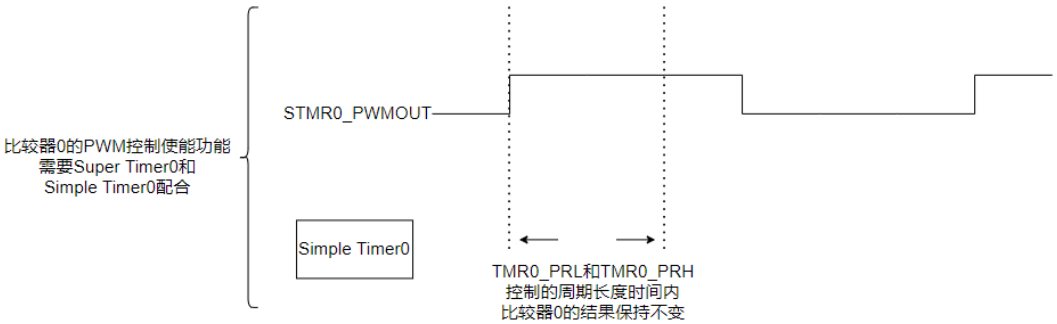
迟滞采样的间隔可以配置(CMPx_CON3[3:0])

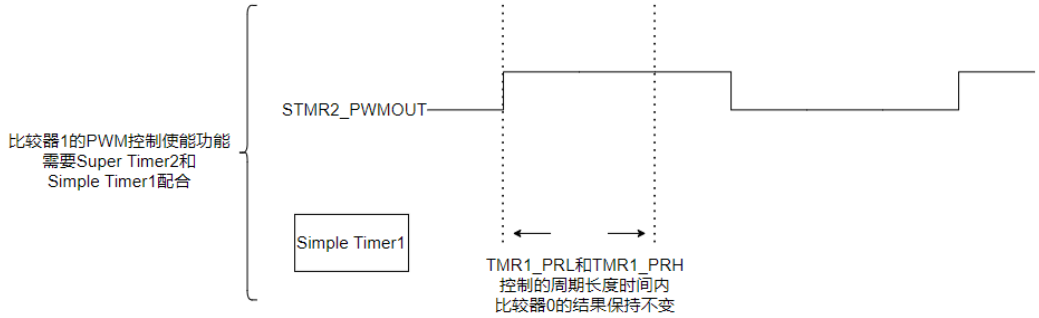
迟滞采样的结果可以作为比较器的输出（CMP_F_SEL[1:0]选择）

迟滞采样可以选择采同步器的结果或滤波器的结果（CMPx_CON3 的 HYSEL）

18.2.3. 关于 PWM 控制比较器使能

PWM 控制使能主要应用于需要使用 PWM 控制功率器件的场景，在控制功率器件的 PWM 翻转时，电源可能产生较大波动，这时比较器的结果可能不可靠，因此在 PWM 翻转之后的一段时间内，比较器的结果会保持不变。这段时间是由 timer 计时的，对应关系如下：





使用 PWM 控制使能模式的时候，Simple timer 需要开启计数模式，计数的周期长度就是比较器结果保持不变的时间，过了这段时间之后，比较器的数字电路才会采样模拟比较器的结果。

18.3. 模块框图

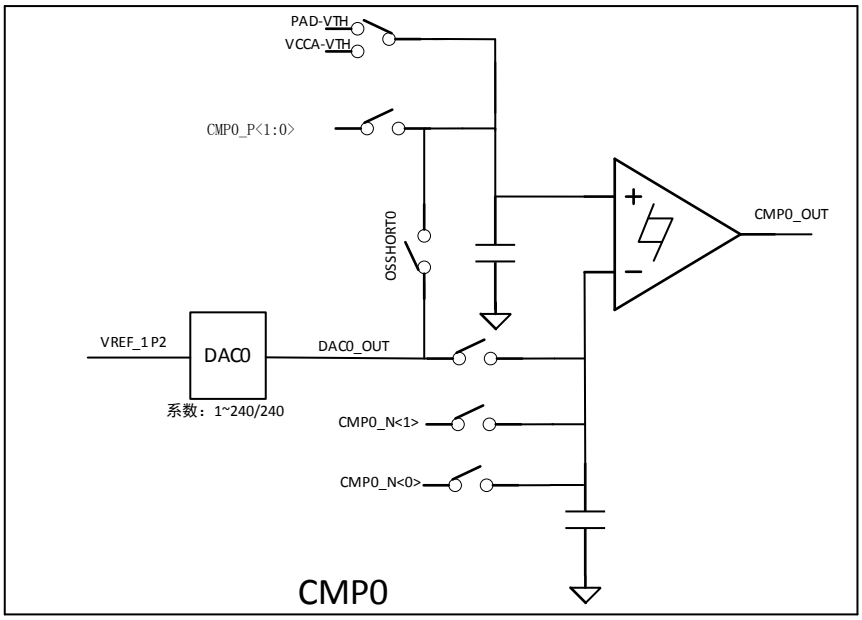


图 18-1 比较器 0 模拟的内部框图

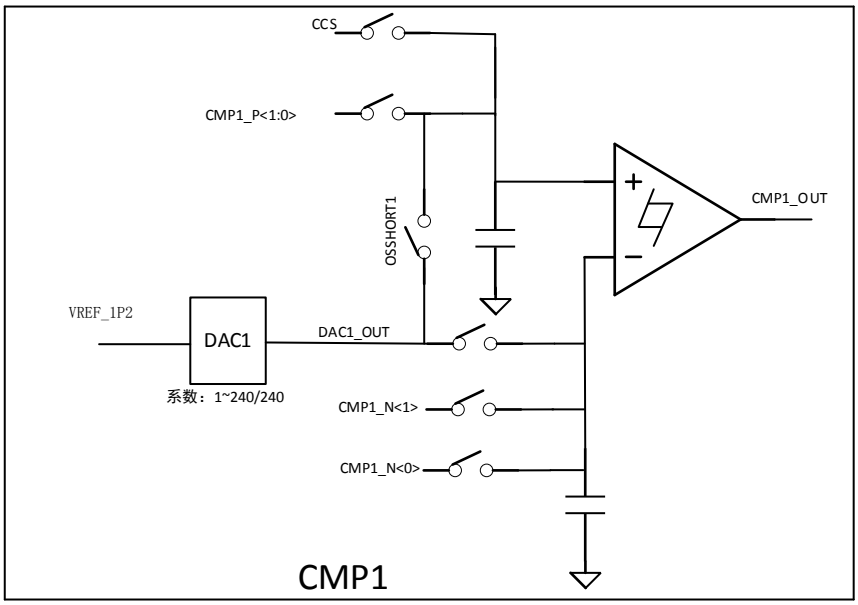


图 18-2 比较器 1 模拟的内部框图

18.4. 引脚对应表

表 18-1 CMP 寄存器列表

输入端		CMPx_CON1 [CHPSEL]	CMPx_CON1 [CHNSEL]	GPIO
CMP0	CMP0_N[0]	/	0x01	P02
	CMP0_N[1]	/	0x02	P14
	CMP0_P[0]	0x01	/	P11
	CMP0_P[1]	0x02	/	P12
CMP1	CMP1_N[0]	/	0x01	P00
	CMP1_N[1]	/	0x02	P01
	CMP1_P[0]	0x01	/	P03
	CMP1_P[1]	0x02	/	P13

Note：表格中 CMPx_CON1 是指 CMP0_CON1 和 CMP1_CON1

18.5. 寄存器列表

表 18-2 CMP 寄存器列表

Address	Register Name	Description
0x36 (XSFR)	CMP0_CON0	CMP0 configuration 0 register
0x37 (XSFR)	CMP0_CON1	CMP0 configuration 1 register
0x38 (XSFR)	CMP1_CON0	CMP1 configuration 0 register
0x39 (XSFR)	CMP1_CON1	CMP1 configuration 1 register
0x3A (XSFR)	CMP_CON	CMP configuration common register
0x3B (XSFR)	CMP_STA	CMP status register
0x74 (XSFR)	CMP0_CON2	CMP0 configuration 2 register
0x75 (XSFR)	CMP0_CON3	CMP0 configuration 3 register
0x76 (XSFR)	CMP0_CON4	CMP0 configuration 4 register
0x77 (XSFR)	CMP0_DAC0	DAC0 register of CMP0
0x78 (XSFR)	CMP0_DAC1	DAC1 register of CMP0
0x79 (XSFR)	CMP1_CON2	CMP1 configuration 2 register
0x7A (XSFR)	CMP1_CON3	CMP1 configuration 3 register
0x7b (XSFR)	CMP1_CON4	CMP1 configuration 4 register
0x7C (XSFR)	CMP1_DAC0	DAC0 register of CMP1
0x7D (XSFR)	CMP1_DAC1	DAC1 register of CMP1

18.6. 寄存器详细说明

18.6.1. CMP0_CON0

Addr = 0x36（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:5	CMPINTS	**比较器输出结果产生中断触发方式控制寄存器** 0x0：上升沿 0x1：下降沿 0x2：双边沿 0x3：高电平 0x4：低电平	RW	0x0
4	-	-	-	-
3	INVENA	**比较器结果取反使能** 0x0：比较器结果不取反 0x1：比较器结果取反	RW	0x0
2	CMPOUT	**比较器的比较结果**	RO	0x0
1	INTENA	**中断使能信号** 0x0：关闭 0x1：打开	RW	0x0
0	ENA	**比较器的模拟部分使能** 使能之后还需要使能数字部分，比较器才正常工作 0x0：关闭 0x1：打开	RW	0x0

18.6.2. CMP0_CON1

Addr = 0x37（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	VTHVCCEN	**VCC 短路保护阈值输入使能信号**	RW	0x0

		（VCC - 0.08/0.2/0.32/0.48） 0x0：关闭 0x1：打开		
6	VTHPADEN	**PAD 短路保护阈值输入使能信号** （PAD - 0.08/0.2/0.32/0.48） 0x0：关闭 0x1：打开	RW	0x0
5:4	CMPVTHS	**比较器内置阈值选择** 0x0：VCC（PAD）-80mv 0x1：VCC（PAD）-200mv 0x2：VCC（PAD）-320mv 0x3：VCC（PAD）-480mv	RW	0x0
3:2	CHPSEL	**比较器正端输入通道选择** 0x0：关闭 0x1：CMP0_P<0> -> P11 0x2：CMP0_P<1> -> P12 0x3：VTH_OUT_VCC/VTH_OUT_PAD	RW	0x0
1:0	CHNSEL	**比较器负端输入通道选择** 0x0：关闭 0x1：CMP0_N<0> -> P02 0x2：CMP0_N<1> -> P14 0x3：DAC0_OUT	RW	0x0

18.6.3. CMP0_CON2

Addr = 0x74（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	CMPFSEL	**比较器结果选择位** 0x0：比较器输出数字滤波器滤波后的结果 0x1：比较器输出同步后的结果 0x2：比较器输出迟滞采样结果	RW	0x0
5	OUTPUT_EN	**控制比较器结果是否输出控制位**	RW	0x0

		0x0：不输出,结果保持关闭输出前的值 0x1：输出		
4:0	FILTNUM	**滤波时钟个数设置** 最大值 32，步长为 1 微秒,即滤波时钟频率是 1M 0x00：滤波 1 个时钟周期 0x01：滤波 2 个时钟周期 ... 0x1F：滤波 32 个时钟周期	RW	0x0

18.6.4. CMP0_CON3

Addr = 0x75（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	HYSEL	**选择迟滞信号是否经过滤波器** 0x0：经过滤波后的信号 0x1：滤波前的同步信号	RW	0x0
6	HYSEN	**比较器数字迟滞功能使能位** 通过 DAC 的两个寄存器控制阈值 0x0：关闭迟滞 0x1：打开迟滞	RW	0x0
5	CMPENS	**比较器输出控制选择** 0x0：比较器是否输出由 OUTPUT_EN 决定 0x1：比较器是否输出由 PWM 控制	RW	0x0
4	HYSRCSEL	**迟滞计数源选择** 0x0：1M 时钟 0x1：64K 时钟	RW	0x0
3:0	CMP0COUNTREG	**迟滞计数器,用来设置迟滞采样间隔** 步长为16微秒(64K时钟)和1微秒(1M时钟)可选 0x0：间隔 1 个周期采样 0x1：间隔 2 个周期采样 ... 0xF：间隔 16 个周期采样	RW	0x0

18.6.5. CMP0_CON4

Addr = 0x76（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	DACTSEN[0]	**比较器 0 的 DAC 的输出测试使能信号** 0x0：关闭 0x1：打开	RW	0x0
6	NONLY	**关闭 P 管使能** 0x0：打开 P 管 0x1：关闭 P 管	RW	0x1
5	OSSHORT	**短路使能** 短接比较器正负端，用于比较器校准 0x0：不短接正负端 0x1：短接正负端	RW	0x0
4	PONLY	**关闭 N 管使能** 0x0：打开 N 管 0x1：关闭 N 管	RW	0x0
3:0	TRIM	**比较器 0 校正值** 芯片上电会自动填入出厂校验值，也可用户填入校验值	RW	-

注意：共模电压（比较器两端的电压）比较低（低于 1/2VDD）时应关闭 N 管打开 P 管，共模电压比较高时应关闭 P 管打开 N 管，如果电压在 0~VCC 之间，需两个同时打开。注意：CMP0 默认只打开 P 管，CMP1 默认只打开 N 管。

18.6.6. CMP0_DAC0

Addr = 0x77（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:0	CMP0_DAC0	**比较器 0 的 DAC0 寄存器** 如果不开启数字迟滞，本寄存器即为比较器 0 内置 DAC 的输入，如果开启数字迟滞，DAC 的输入值在 CMP0_DAC0 和 CMP0_DAC1 之间选择，当比较	RW	0x0

		输出为 1 时选择 CMP0_DAC0，当比较器输出为 0 时选择 CMP0_DAC1 Step=5mV,0x00~0xF0 对应转换输出为 0~1.2V，0xF0~0xFF 转换输出为 1.2V		

18.6.7. CMP0_DAC1

Addr = 0x78（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:0	CMP0_DAC1	**比较器 0 的 DAC1 寄存器** 使能数字迟滞时生效 Step=5mV,0x00~0xF0 对应转换输出为 0~1.2V，0xF0~0xFF 转换输出为 1.2V	RW	0x0

18.6.8. CMP1_CON0

Addr = 0x38（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:5	CMPINTS	**比较器输出结果产生中断触发方式控制寄存器** 0x0：上升沿 0x1：下降沿 0x2：双边沿 0x3：高电平 0x4：低电平	RW	0x0
4	-	-	-	-
3	INVENA	**比较器结果取反使能** 0x0：比较器结果不取反 0x1：比较器结果取反	RW	0x0
2	CMPOUT	**比较器的比较结果**	RO	0x0
1	INTENA	**中断使能信号** 0x0：关闭 0x1：打开	RW	0x0

0	ENA	**比较器的模拟部分使能** 使能之后还需要使能数字部分，比较器才正常工作 0x0：关闭 0x1：打开	RW	0x0

18.6.9. CMP1_CON1

Addr = 0x39（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:4	-	-	-	-
3:2	CHPSEL	**比较器正端输入通道选择** 0x0：关闭 0x1：CMP1_P<0> -> P03 0x2：CMP1_P<1> -> P13 0x3：CCS_FB	RW	0x0
1:0	CHNSEL	**比较器负端输入通道选择** 0x0：关闭 0x1：CMP1_N<0> -> P00 0x2：CMP1_N<1> -> P01 0x3：DAC0_OUT	RW	0x0

18.6.10. CMP1_CON2

Addr = 0x79（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:6	CMPFSEL	**比较器结果选择位** 0x0：比较器输出数字滤波器滤波后的结果 0x1：比较器输出同步后的结果 0x2：比较器输出迟滞采样结果	RW	0x0
5	OUTPUTEN	**控制比较器结果是否输出控制位**	RW	0x0

		0x0：不输出,结果保持关闭输出前的值 0x1：输出		
4:0	FILTNUM	**滤波时钟个数设置** 最大值 32，步长为 1 微秒,即滤波时钟频率是 1M 0x00：滤波 1 个时钟周期 0x01：滤波 2 个时钟周期 ... 0x1F：滤波 32 个时钟周期	RW	0x0

18.6.11. CMP1_CON3

Addr = 0x7A（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	HYSEL	**选择迟滞信号是否经过滤波器** 0x0：经过滤波后的信号 0x1：滤波前的同步信号	RW	0x0
6	HYSEN	**比较器数字迟滞功能使能位** 通过 DAC 的两个寄存器控制阈值 0x0：关闭迟滞 0x1：打开迟滞	RW	0x0
5	CMPENS	**比较器输出控制选择** 0x0：比较器是否输出由 OUTPUT_EN 决定 0x1：比较器是否输出由 PWM 控制	RW	0x0
4	HYSRCSEL	**迟滞计数源选择** 0x0：1M 时钟 0x1：64K 时钟	RW	0x0
3:0	CMP0COUNTREG	**迟滞计数器** 用来设置迟滞采样间隔,步长为 16 微秒(64K 时钟)和 1 微秒(1M 时钟)可选 0x0：间隔 1 个周期采样 0x1：间隔 2 个周期采样	RW	0x0

		... 0xF：间隔 16 个周期采样		

18.6.12. CMP1_CON4

Addr = 0x7B（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	DACTSEN[1]	**比较器 1 的 DAC 的输出测试使能信号** 0x0：关闭 0x1：打开	RW	0x0
6	NONLY	**关闭 P 管使能** 0x0：打开 P 管 0x1：关闭 P 管	RW	0x0
5	OSSHORT	**短路使能** 短接比较器正负端，用于比较器校准 0x0：不短接正负端 0x1：短接正负端	RW	0x0
4	PONLY	**关闭 N 管使能** 0x0：打开 N 管 0x1：关闭 N 管	RW	0x1
3:0	TRIM	**比较器 0 校正值** 芯片上电会自动填入出厂校验值，也可用户填入校验值	RW	-

注意：共模电压（比较器两端的电压）比较低（低于 1/2VDD）时应关闭 N 管打开 P 管，共模电压比较高时应关闭 P 管打开 N 管，如果电压在 0~VCC 之间，需两个同时打开。注意：CMP0 默认只打开 P 管，CMP1 默认只打开 N 管。

18.6.13. CMP1_DAC0

Addr = 0x7C（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:0	CMP0_DAC0	**比较器 0 的 DAC0 寄存器**	RW	0x0

		如果不开启数字迟滞，本寄存器即为比较器 1 内置 DAC 的输入，如果开启数字迟滞，DAC 的输入值在 CMP1_DAC0 和 CMP1_DAC1 之间选择，当比较输出为 1 时选择 CMP1_DAC0，当比较器输出为 0 时选择 CMP1_DAC1 Step=5mV,0x00~0xF0 对应转换输出为 0~1.2V，0xF0~0xFF 转换输出为 1.2V		

18.6.14. CMP1_DAC1

Addr = 0x7D（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:0	CMP0_DAC1	**比较器 1 的 DAC1 寄存器** 使能数字迟滞时生效 Step=5mV,0x00~0xF0 对应转换输出为 0~1.2V，0xF0~0xFF 转换输出为 1.2V	RW	0x0

18.6.15. CMP_CON

Addr = 0x3A（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	REFEN	**比较器的参考电压参考电流使能信号** 0x0：关闭 0x1：打开	RW	0x0
6	CCSEN	**恒流源的使能信号** 0x0：关闭 0x1：打开	RW	0x0
5:0	TRIMIB	**恒流源电流调节信号(step=2.5%)** 0x00：11.8mA 0x0F：20mA	RW	0x0

		0x1F：28.4mA		

18.6.16. CMP_STA

Addr = 0x3B（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	CMP1WKUPEN	**比较器 1 数字信号 wakeup 使能** 用于唤醒 CPU 0x0：关闭 0x1：打开	RW	0x0
6	CMP0WKUPEN	**比较器 0 数字信号 wakeup 使能** 用于唤醒 CPU 0x0：关闭 0x1：打开	RW	0x0
5	CMP1ANAINVEN	**比较器 1 模拟 WAKEUP 取反** 0x0：不取反 0x1：取反	RW	0x0
4	CMP0ANAINVEN	**比较器 0 模拟 WAKEUP 取反** 0x0：不取反 0x1：取反	RW	0x0
3	CMP1ANAWKP	**比较器 1 模拟信号 WAKEUP 使能** 用于唤醒 CPU 0x0：关闭 0x1：打开	RW	0x0
2	CMP0ANAWKP	**比较器 0 模拟信号 WAKEUP 使能** 用于唤醒 CPU 0x0：关闭 0x1：打开	RW	0x0
1	CMP1PNDCLR	**比较器 1 中断标志位** 写 1 清零	RW	0x0
0	CMP0PNDCLR	**比较器 0 中断标志位**	RW	0x0

		写 1 清零		

19. I2C 模块

19.1. 功能概述

I2C 模块功能特殊：

- 支持主机模式和从机模式
- 支持主机仲裁

19.2. 功能描述

在 I2C 协议定义中，有四种工作模式：主机发送、主机接收、从机发送、从机接收。还有广播模式，其工作方式类似于主机发送。

19.2.1. 主机发送

主机发送模式下，主机应该提供时钟，可通过设置 STA（I2CCON.5）寄存器为 1 来进入主机模式。当模块检测到总线处于空闲时，将发送一个起始位。当起始位被成功发送(时钟保持高电平，数据线拉低)，SI 寄存器将被置 1 并且状态码(I2STAT)被设为 08H。软件此时应该将从机地址和写命令(SLA+W)写入 I2DAT。接下来 SI 位应该由软件清零，触发 SLA+W 的发送。

当 SLA+W 被成功发送并且从设备返回一个 ACK 之后，SI 会被再次置位，I2STAT 此时是 18H.此时根据需要来进行下一步操作。

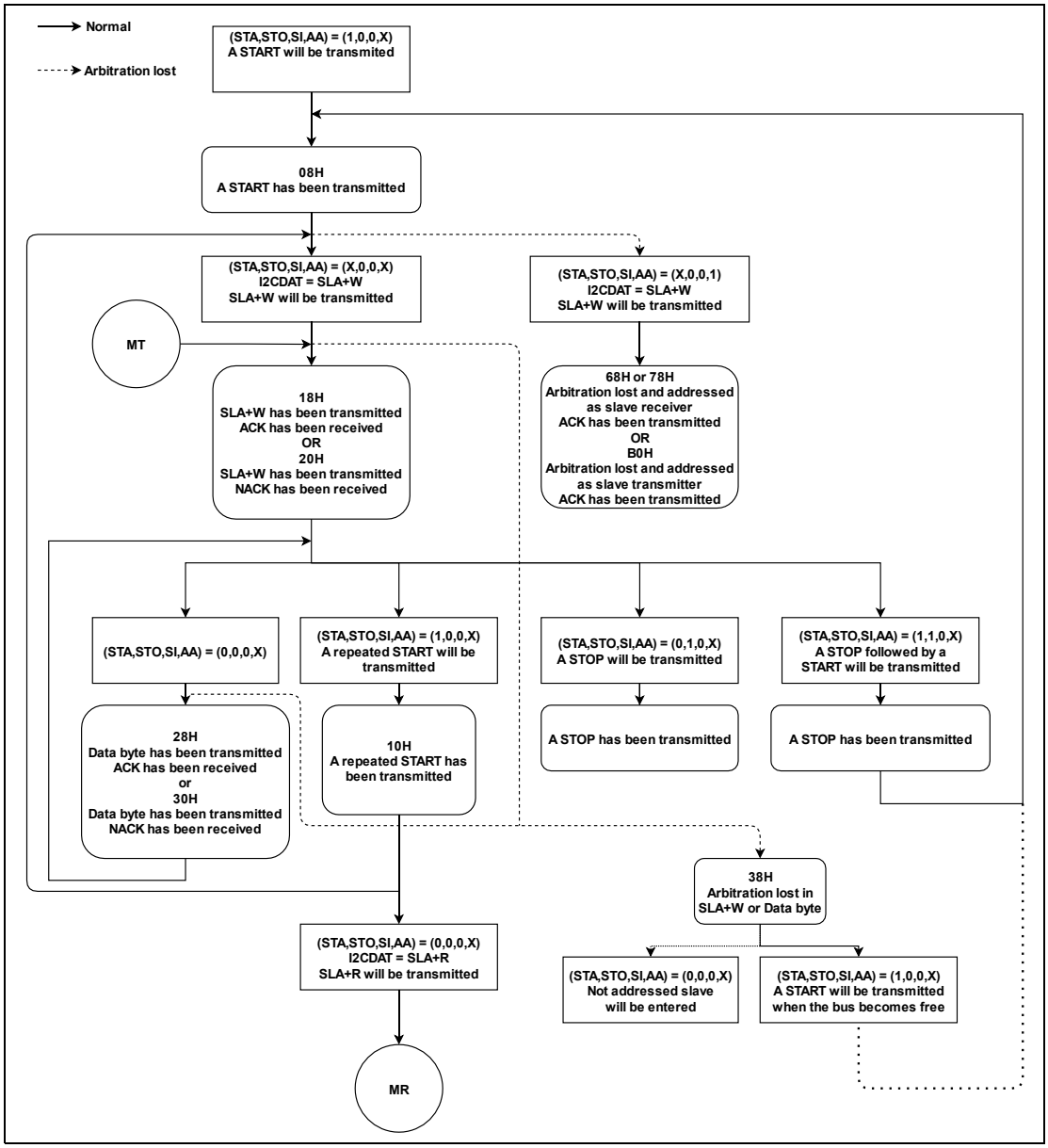


图 19-1 I2C 主机发送流程图

19.2.2. 主机接收

在主机接收模式下，起始位的发送和主机发送模式一样，不同的是写入 I2DAT 的数据变为（SLA+R），发送成功并接收到 ACK 后 SI 会被置位，此时 I2STAT 值为 40H。

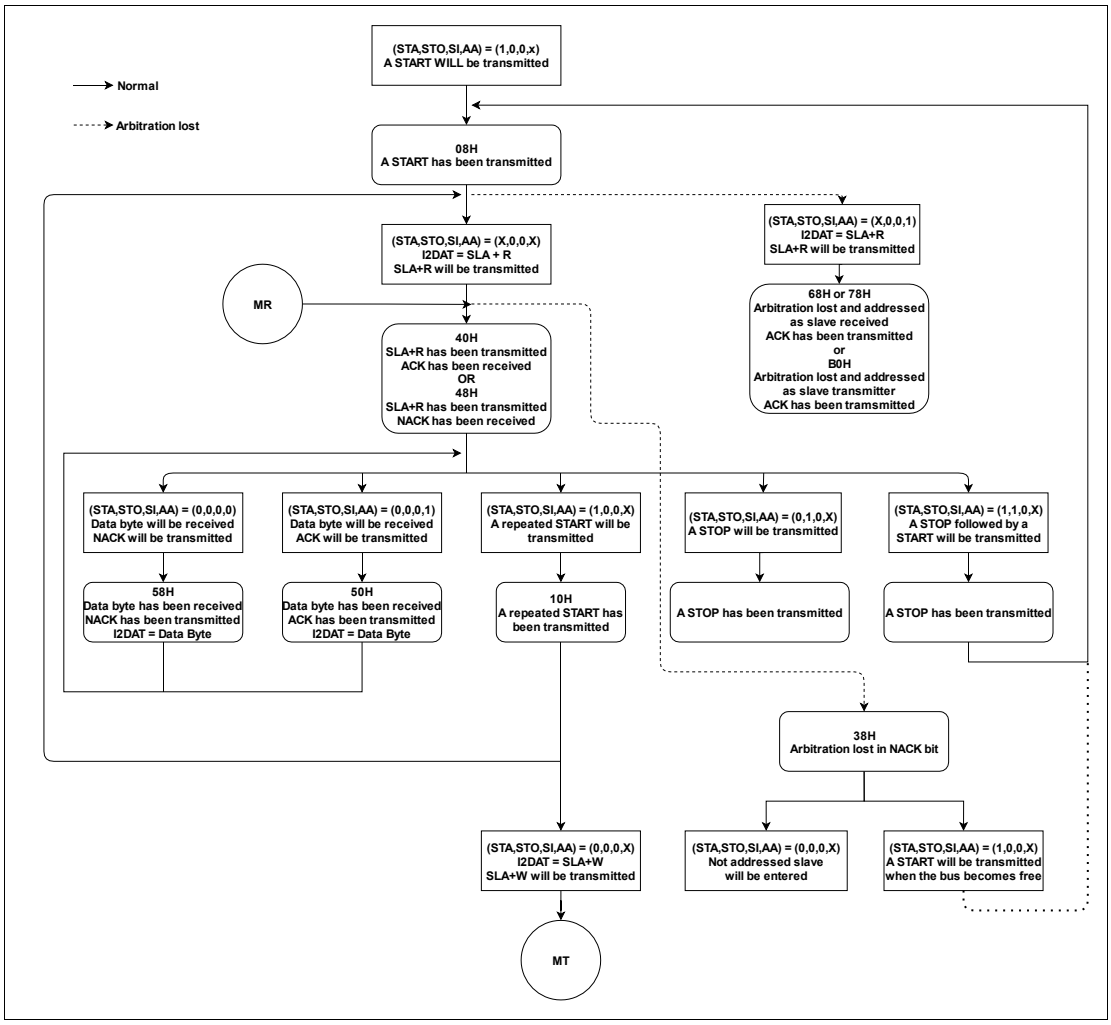


图 19-2 I2C 主机接收流程图

19.2.3. 从机接收

在从机接收模式下，传输开始之前应先将需设置的本模块从机地址写入 I2ADDR，AA 寄存器应被置 1，被置 1 后模块才会在接收到本机地址后回应 ACK。

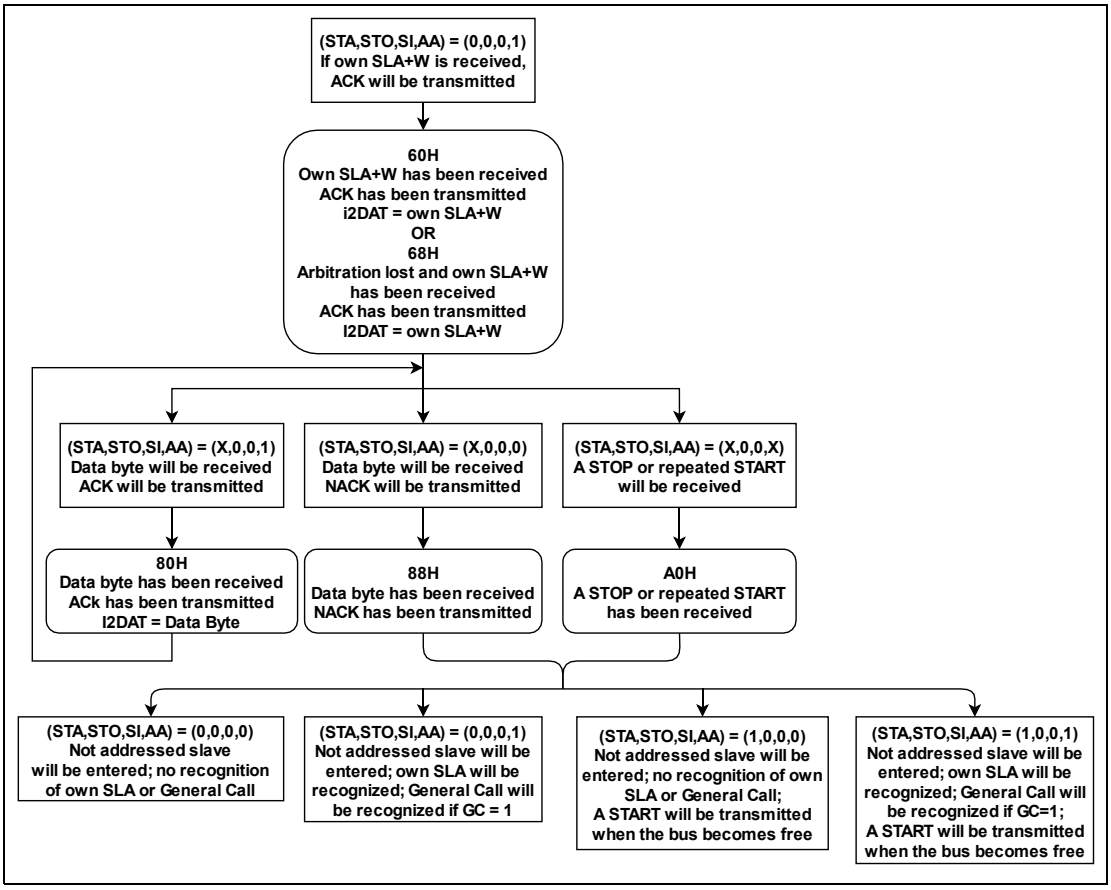


图 19-3 I2C 从机接收流程图

19.2.4. 从机发送

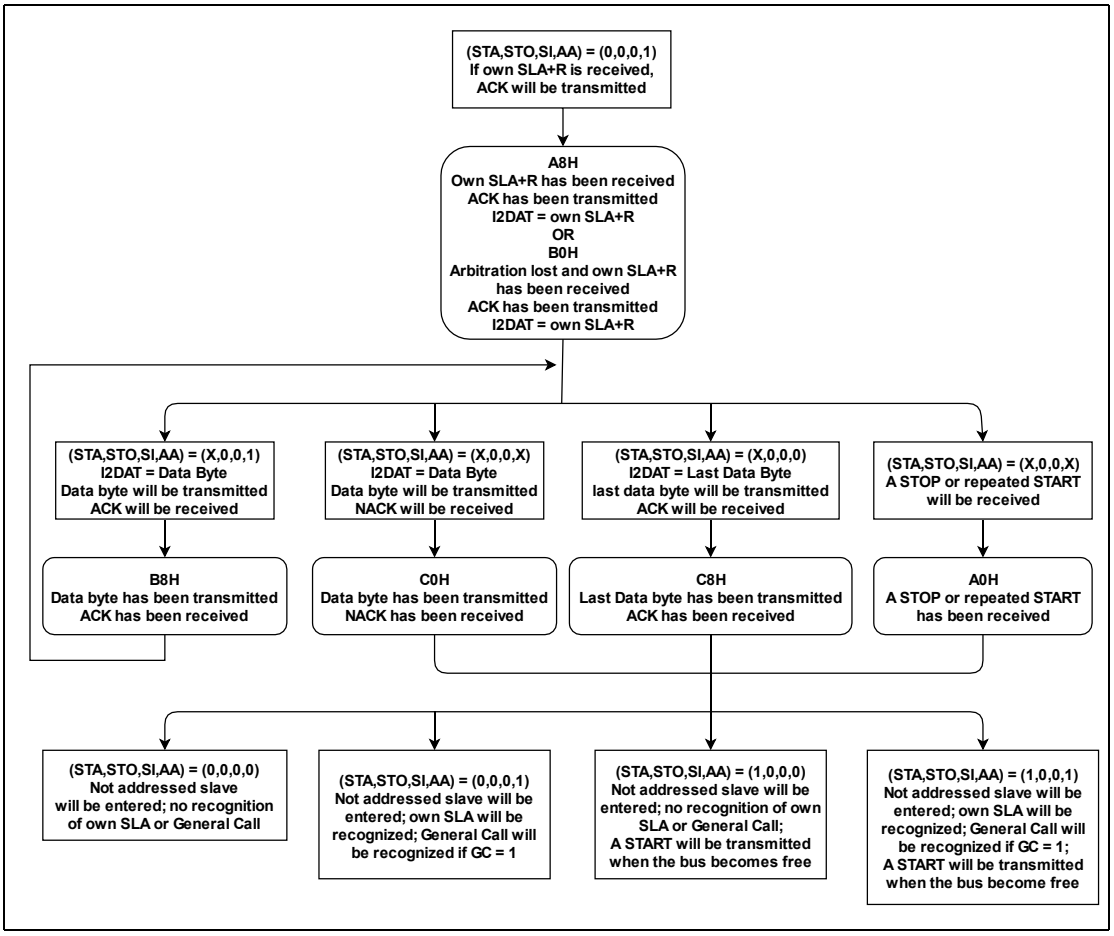


图 19-4 I2C 从机发送流程图

19.2.5. 广播模式

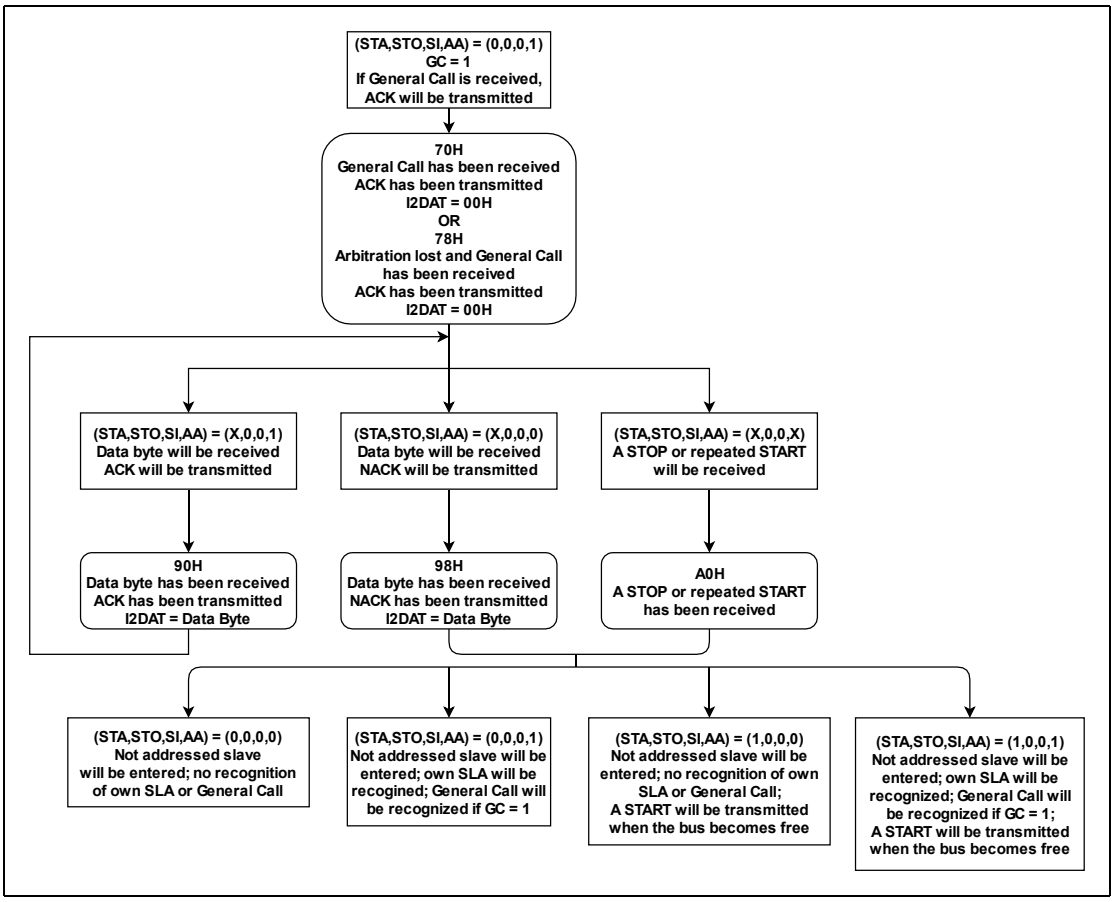


图 19-5 I2C 广播模式流程图

19.3. 寄存器列表

表 19-1 IIC 寄存器列表

Address	Register Name	Description
0x70 (XSFR)	I2C0_CON	I2C control register
0x71 (XSFR)	I2C0_DATA	I2C data register
0x72 (XSFR)	I2C0_ADR	I2C address register
0x73 (XSFR)	I2C0_STA	I2C status register

19.4. 寄存器详细说明

19.4.1. I2C0_CON

Addr = 0x70（XSFR）

Bit(s)	Name	Description	R/W	Reset
7	ENS1	**IIC 使能位** 0x0：“sdao”和“sclo”输出 1，并忽略“sdai”和“scli”输入 0x1：使能 iic 模块	RW	0x0
6	STA	**IIC 起始位** 0x0：无操作 0x1：检查 iic 总线，如果总线处于空闲状态，并且模块处于主机模式，则发送一个起始位	RW	0x0
5	STO	**IIC 停止位** 0x0：无操作 0x1：当模块处于主机模式，则发送一个停止位	RW	0x0
4	SICLR	**SI 清除位** 写 1 清除 SI，读恒为 0	WO	0x0
3	AA	**IIC 应答控制位** 0x0：当出现以下情况时发送 NACK ● 在主机接收模式下接收完 1byte ● 在从机接收模式下接收完 1byte 0x1：当出现以下情况时发送 ACK ● 接收到本机从机地址 ● 在广播地址位使能的情况下接收到广播地址 ● 主机接收情况下接收到 1byte 从机接收情况下接收到 1byte	RW	0x0
2:0	CR	**波特率控制位** I2C 波特率由以下公式计算得出:baud(kHz)=	RW	0x0

		sysclk/x，其中 x 的值由 CR 寄存器决定,分别是: 0x00:256 0x01:224 0x02:160 0x03:80 0x04:1024 0x05:120 0x06:60 0x07:当CR设置为0x07时,波特率由Timer0的pwm频率决定,计算方法为 Timer0 的 pwm 频率除以 8		

19.4.2. I2C0_STA

Addr = 0x73 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7:3	STA	**模块状态标志位**	RO	0x1F
2	SI	**SI 标志位** 除了 f8H 状态之外的所有状态都会将 SI 置位，SI_CLR 写 1 和读 DATA 都会清除 SI	RO	0x0
1	INT	**中断标志位** 当使能了中断后，随 SI 置位	RO	0x0
0	INTEN	**中断使能位**	RW	0x0

19.4.3. I2C0_ADR

Addr = 0x72 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7:1	ADR	**本模块从机地址**	RW	0x0
0	GC	**广播地址控制位** 0x0：忽略广播地址	RW	0x0

		0x1：响应广播地址		

19.4.4. I2C0_DATA

Addr = 0x71（XSFR）

Bit(s)	Name	Description	R/W	Reset
7:0	DAT	**数据寄存器**	RW	0x0