



深圳富芯电子

TX8M226X 用户手册

增强型 8 位电控单片机

Rev. V2.0

重要声明：本公司保留对以下所有产品在可靠性、功能和设计方面作进一步说明的权利，同时保留在未通知的情况下，对本产品所有文档做更改的权利。客户在使用本产品时，请向我司销售人员索要最新文档，特此

修订记录

日期	版本	描 述	修订人
2024-09-04	V2.0	1、修改 CLK_CON5[5:4]描述;	TX
2023-10-16	V1.18	1、修改比较器部分的功能框图;	TX
2023-08-21	V1.17	1、修改 ADC 参考电压的描述;	TX
2023-05-30	V1.16	1、修改 ADC 参考电压的描述;	TX
2023-05-17	V1.15	1、修改 ADC_ACON0 的 bit2, bit3 的默认值描述错误; 2、修改 SPI 模块的使用流程说明; 3、修改 ADC 内部参考电压的描述;	TX
2023-04-27	V1.14	1、修改 ADC 内部参考描述;	TX
2023-02-03	V1.13	1、解决部分设备打开阅读时, 出现乱码问题;	TX
2023-01-09	V1.12	1、修改 CMP_STA[5:2]和 I2C_STA[2]的寄存器错误描述。补充对 CMP_STA[7:6]的描述。修改 CMP0_CON4[4], CMP0_CON4[6], CMP1_CON4[4], CMP1_CON4[6]的描述使之更容易理解。修改对 Simple Timer0-3 的 TMRx_PRL 和 TMRx_PRH 的可读性描述。修改低功耗管理章节的关于进入 IDLE 模式的描述;	TX
2022-08-10	V1.11	1、修改 ADC_CFG1, IO 端口和运放模块的部分内容描述;	TX
2022-07-06	V1.10	1、修改 CMP0_CON5[3:2]和 SPI_CON[2]的寄存器错误描述; 2、第三章存储器描述中增加 XSFR 实际地址的说明;	TX
2022-06-15	V1.9	1、增加 FLASH 控制模块的 NVR 系统信息区域说明;	TX
2022-06-13	V1.8	1、修改工作温度范围为-40℃ ~ 105℃; 并增加了 IO_MAP[2:0]写保护说明;	TX
2022-06-07	V1.7	1、增加 LED 硬件模块的内容;	TX



目录

1. 产品概述.....	1
1.1. 说明.....	1
1.2. 特性.....	1
2. 中央处理器.....	4
2.1. 累加器 (ACC)	4
2.2. 寄存器 (B)	4
2.3. 堆栈指针寄存器 (SP)	5
2.4. 堆栈指针寄存器 (SPH)	5
2.5. 数据指针寄存器 (DPTR0/DPTR1)	5
2.6. 数据指针控制寄存器 (DPCFG)	6
2.7. 程序状态寄存器 (PSW)	9
2.8. PCON1	9
2.9. 程序计数器 (PC)	10
3. 存储器.....	11
3.1. 程序存储器.....	11
3.2. XDATA 数据存储器	12
3.3. IDATA	12
3.4. SFR 空间.....	14
4. 时钟系统.....	15
4.1. 时钟系统概述.....	15
4.2. 时钟系统主要功能.....	15
4.3. 时钟系统框图.....	16
4.4. 系统振荡器.....	17
5. 复位系统.....	18
5.1. 上电复位.....	18
5.2. 掉电复位.....	18
5.3. 看门狗复位.....	18
5.4. 低电检测复位.....	21
6. 低功耗管理.....	25
6.1. Idle Mode 及唤醒	25
6.2. Stop Mode 及唤醒	25
6.3. Sleep Mode 及唤醒	25
6.4. 低功耗唤醒单元结构图.....	26
6.5. 寄存器详细说明.....	26
7. 系统控制模块.....	30
7.1. 功能概述.....	30
7.2. 寄存器列表.....	30
7.3. 寄存器详细说明.....	31
8. 中断系统.....	54
8.1. 中断概述.....	54
8.2. 中断向量表.....	54
8.3. 寄存器列表.....	56
8.4. 寄存器详细说明.....	56
8.5. 中断优先级及中断嵌套.....	64
9. I/O 端口	64
9.1. 功能描述.....	64
9.2. 结构框图.....	65
9.3. 引脚功能复用.....	65



9.4.	寄存器列表.....	69
9.5.	寄存器详细说明.....	75
10.	SPI 模块.....	190
10.1.	功能概述.....	190
10.2.	模块框图.....	191
10.3.	寄存器列表.....	191
10.4.	寄存器详细说明.....	192
10.5.	使用流程说明.....	193
11.	UART0/1 模块.....	195
11.1.	功能概述.....	195
11.2.	模块框图.....	195
11.3.	寄存器列表.....	195
11.4.	寄存器详细说明.....	196
11.5.	使用流程说明.....	203
12.	I2C 模块.....	204
12.1.	功能概述.....	204
12.2.	功能描述.....	204
12.3.	寄存器列表.....	209
12.4.	寄存器详细说明.....	210
13.	MDU 运算加速模块.....	212
13.1.	功能概述.....	212
13.2.	运算性能.....	212
13.3.	模块框图.....	213
13.4.	寄存器列表.....	213
13.5.	寄存器详细说明.....	213
13.6.	使用流程说明.....	216
14.	Simple Timer 模块.....	217
14.1.	功能概述.....	217
14.2.	模块框图.....	219
14.3.	寄存器列表.....	219
14.4.	寄存器详细说明.....	221
14.5.	使用流程说明.....	239
15.	Normal Timer 模块.....	240
15.1.	功能概述.....	240
15.2.	模块框图.....	243
15.3.	寄存器列表.....	243
15.4.	寄存器详细说明.....	244
15.5.	使用流程说明.....	252
16.	Super timer 模块（增强型 PWM 模块）.....	255
16.1.	功能概述.....	255
16.2.	模块框图.....	277
16.3.	寄存器列表.....	277
16.4.	寄存器详细说明.....	282
16.5.	使用流程说明.....	304
17.	CRC 校验模块.....	305
17.1.	功能概述.....	305
17.2.	基本功能.....	305
17.3.	模块框图.....	306
17.4.	寄存器列表.....	306



17.5.	寄存器详细说明.....	307
17.6.	使用流程说明.....	308
18.	Flash 控制器模块	310
18.1.	功能概述.....	310
18.2.	基本功能.....	310
18.3.	模块框图.....	315
18.4.	寄存器列表.....	316
18.5.	寄存器详细说明.....	317
18.6.	使用流程说明.....	324
19.	模数转换器 (ADC)	325
19.1.	功能概述.....	325
19.2.	基本功能.....	326
19.3.	模块框图.....	331
19.4.	寄存器列表.....	333
19.5.	寄存器详细说明.....	333
19.6.	使用流程说明.....	347
20.	模拟比较器 (CMP0/1)	348
20.1.	功能概述.....	348
20.2.	模块框图.....	349
20.3.	引脚复用映射表.....	350
20.4.	功能配置流程图.....	351
20.5.	基本功能使用说明.....	351
20.6.	寄存器列表.....	355
20.7.	寄存器详细说明.....	355
21.	运放模块.....	366
21.1.	功能概述.....	366
21.2.	引脚复用表.....	366
21.3.	基本运放功能.....	367
21.4.	增强功能.....	368
21.5.	模块框图.....	378
21.6.	寄存器列表.....	379
21.7.	寄存器详细说明.....	380
22.	LED 模块.....	390
22.1.	功能概述.....	390
22.2.	功能框图.....	390
22.3.	数据结构.....	391
22.4.	寄存器列表.....	392
22.5.	寄存器详细说明.....	393
22.6.	使用流程说明.....	395

1. 产品概述

1.1. 说明

TX8M226x 是一款高性能低功耗的 8051 内核 MCU，工作主频最高为 48MHz，内置 16K 字节 LogicFlash（以下简称 FLASH）存储器，支持类 EEPROM 功能，2K 字节 SRAM。

模拟资源：

1 个 12 位 500KSPS 的 SARADC、2 个多功能比较器，3 个运算放大器。

定时器资源：

6 个 16 位高级定时器（3 对互补 PWM、带死区控制或 6 路独立 PWM）、

5 个 16 位通用定时器（都支持 Capture、Count、PWM 功能）、

1 个 16 位唤醒定时器（都支持 Capture、Count、PWM 功能）、

1 个 8 位蜂鸣器（支持 PWM、Count 功能）、

1 个看门狗定时器。

LED 显示功能：

支持多达 8COM x 12SEG。

硬件加速模块：

支持有符号数/无符号数的数学计算功能包括 32 位/16 位除法，16 位/16 位除法，16 位 x16 位乘法，16 位+16 位加法，16 位-16 位减法，32 位开平方。

标准的通信接口：

1 个 SPI 接口、1 个 IIC 接口和 2 个 UART 接口（其中 UART1 支持 DMA 工作方式）。

GPIO：

内置 30K 上下拉电阻，多个驱动档位可配置，每个 IO 都可以作为 ADC 的输入，每个 IO 都可以作为 IO 中断唤醒口。

支持宽范围电压供电，工作电压为 2.4V ~ 5.5V，工作温度范围-40℃ ~ +105℃。多种省电工作模式保证低功耗应用的要求，最低功耗模式 5uA。

TX8M226x 产品系列包含 TX8M2260，TX8M2261，TX8M2262 等产品型号，不同产品型号资源和封装形式都不相同。

应用场合：

- 小家电
- 电机控制系列 MCU 产品应用

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：48MHz
- 32 个中断源，支持硬件两级优先级
- 支持在线调试接口
- 支持代码加密



- 支持带电烧录
- **工作电压**
 - 2.4V ~ 5.5V宽电压范围供电
- **存储器**
 - 16K字节Flash，用于存储用户代码
 - 2K字节RAM
 - 支持EEPROM功能
- **时钟**
 - 内部 1~48MHz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/8~40MHz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
 - LVD低压检测复位，提供 8 级低压检测电压
- **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
 - 2 个UART接口，最大支持 4Mbps，其中UART1 支持DMA模式
- **定时器资源**
 - 6 个 16 位高级定时器，支持 3 对互补输出，支持死区插入和事件刹车功能，支持单脉冲模式。或支持 6 个独立PWM输出
 - 5 个 16 位通用定时器，都支持Capture、Count、PWM功能
 - 1 个 16 位唤醒定时器
 - 1 个 8 位蜂鸣器定时器
 - 1 个看门狗定时器
- **高安全性**
 - 支持 32 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 5uA (@25℃, 5V供电)，3uA (@25℃, 3.3V供电)
 - 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 10MHz，最大采样率 500KSPS
 - 失调校正step 2mV, DNL +-2 INL +-4
 - 外部输入通道任意IO可选，2 个模拟通路
 - 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，正端支持 6 个GPIO，负端支持 2 个GPIO
- **3 个可编程增益放大器 (PGA)**
 - 3 个可编程高增益放大器，多级可配置增益 (1/2/4/8/16/32/64/128/256/512)
 - 支持OP工作模式，外接电阻调节放大增益



- LED显示功能
 - 支持多达 8COMx 12SEG
- 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
 - 均支持上/下拉电阻功能
 - 均支持唤醒功能
 - 可编程驱动能力，驱动电流范围 4mA ~ 64mA
 - 支持OD输出低/高模式。
 - 支持独立控制的上下拉电阻，阻值 30K Ω
- 高可靠性
 - ESD HBM 6KV
 - Latch-up $\pm 200\text{mA}$ @25°C
- 96 位的芯片唯一 ID (UID)
- 工作温度范围
 - -40°C ~ +105°C



2. 中央处理器

TX8M226x 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。TX8M226x 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。

TX8M226x 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 3bit，复位后默认值位 0x7，与 SP 组合使用，意味着堆栈的区域从 RAM 地址的 0x707 开始。该值可以修改，如果将堆栈区域设置为 0x10B 开始，则在复位后将 SPH 和 SP 的值分别设置为 0x1 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: 3	—	—	RO	0x0
2: 0	SPH	堆栈指针寄存器高位	RW	0x7

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	0x00

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	0x00



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	0x00

Addr = 0x85 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	DPH1	DPTR1 数据指针寄存器高八位	RW	0x00

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	中断地址起始区域选择 0x0: 中断起始地址为 0x000C 0x1: 中断起始地址为 0x80C 0x2: 中断起始地址为 0x100C 0x3: 中断起始地址为 0x600C	RW	0x0
5	DPID0	DPTR0 加 1 或减 1 选择 0x0: DPTR0 加 1 0x1: DPTR0 减 1 Note: DPSEL 为 0, DPAID 为 1 时 DPTR0 指针会在以下指令之后根据 DPID0 选择自动加 1 或者减 1; MOVC A, @A+DPTR MOVX A, @DPTR MOVX @DPTR, A INC DPTR 如果 DPSEL 为 0, DPAID 为 0, 则只在指令 INC DPTR 后根据 DPID0 选择自动加 1 或者减 1;	RW	0x0
4	DPID1	DPTR1 加 1 或减 1 选择 0x0: DPTR1 加 1 0x1: DPTR1 减 1 Note: DPSEL 为 1, DPAID 为 1 时 DPTR1 指针会在	RW	0x0

		以下指令之后根据 DPID1 选择自动加 1 或者减 1; MOVC A, @A+DPTR MOVX A, @DPTR MOVX @DPTR, A INC DPTR 如果 DPSEL 为 1, DPAID 为 0, 则只在指令 INC DPTR 后根据 DPID1 选择自动加 1 或者减 1;		
3	DPAID	DPTR0/DPTR1 自加自减使能 0x0: 不使能 0x1: 使能 Note: 使能之后, 以下指令会根据 DPID0 和 DPID1 以及 DPSEL 的配置使 DPTR0 或者 DPTR1 在指令后自 加 1 自减 1; MOVC A, @A+DPTR MOVX A, @DPTR MOVX @DPTR, A	RW	0x0
2	DPTSL	DPSEL 自动翻转使能 0x0: 不使能 0x1: 使能 Note: 使能之后, 以下指令会使 DPTR 指针在 DPTR0 和 DPTR1 之间自动切换; MOVC A, @A+DPTR MOVX A, @DPTR MOVX @DPTR, A INC DPTR MOV DPTR, #DATA16 如当前指令使用 DPTR0 作为指针, 下一条 DPTR 操 作指令将使用 DPTR1 作为指针; 此功能下更改 DPSEL 的值可影响下一条 DPTR 指令的指针选择;	RW	0x0
1	-	-	RW	0x0
0	DPSEL	选择 DPTR0 /DPTR1 0x0: 指针 DPTR0 有效 0x1: 指针 DPTR1 有效	RW	0x0



		Note: DPTSL 使能后会使 DPSEL 在使用 DPTR 操作指令后自动翻转，相关指令见 DPTSL； MOV DPTR, #DATA16 指令可以根据 DPSEL 的选择将立即数输入 DPTR0 或者 DPTR1 相关寄存器；		



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: 加法位 7 有进位或减法位 7 有借位	RW	0x0
6	AC	辅助进位标志位 0x0: 无进位 0x1: 加法位 3 有进位或减法位 3 有借位	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. PCON1

Addr = 0x9C (SFR)

Bit(s)	Name	Description	RW	Reset
7	PCON1_SEL	MOVX @RI 寻址选择位 0x0: MOVX @RI 寻址 IRAM 0x1: MOVX @RI 寻址 XSFR	RW	0x1
6	—	—	RW	0x0



5	ADR_OV_CLR	地址越界清零 写 1 清掉地址越界标志位，写零无效	RW	0x0
4	ADR_OV_IF	地址越界标志 0x0: PC 地址未越界 0x1: PC 地址越界	RW	0x0
3	ADROV_EN	PC 地址越界功能使能 0x0: 地址越界功能不使能 0x1: 地址越界功能使能 Note: 使能后，当 PC 运行在 0x0-0x4000 和 0x6000-0x67FF 之外的区域，将会产生越界标志，并复位；	RW	0x0
2: 0	PCON1_VAL	MOVX @RI 寻址高位 可使该指令在 IRAM 或者 XSFR 的 0~2K 空间进行寻址，寻址地址为 {PCON1_VAL, @RI}	RW	0x0

2.9. 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

TX8M226x 有内部有 3 种存储器：XDATA，IDATA，程序存储器。

IDATA 大小为 256 字节，XDATA 大小为（2K-256）字节，程序存储器大小为 16K 字节。

3.1. 程序存储器

TX8M226x 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 16K 字节的程序存储空间。



图 3-1 程序存储空间地址映射

复位后，MCU 从地址 0x0000 开始执行。0x0003~0x000F 之间为系统配置区。从 0x000F 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。Flash 储存区的结束地址为 0x4000。0x4000~0x407F 为 NVR 区。

3.2. XDATA 数据存储器

XDATA 分为两部分：0x6000~0x67FF 为数据存储空间，0x7000~0x77FF 为 XSFR 空间。（注意：本文中所有 XSFR 的地址都是指偏移地址，XSFR 实际逻辑地址=对应 XSFR 偏移地址+0x7000【基地址】；）

3.3. IDATA

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 字节可以直接访问，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。

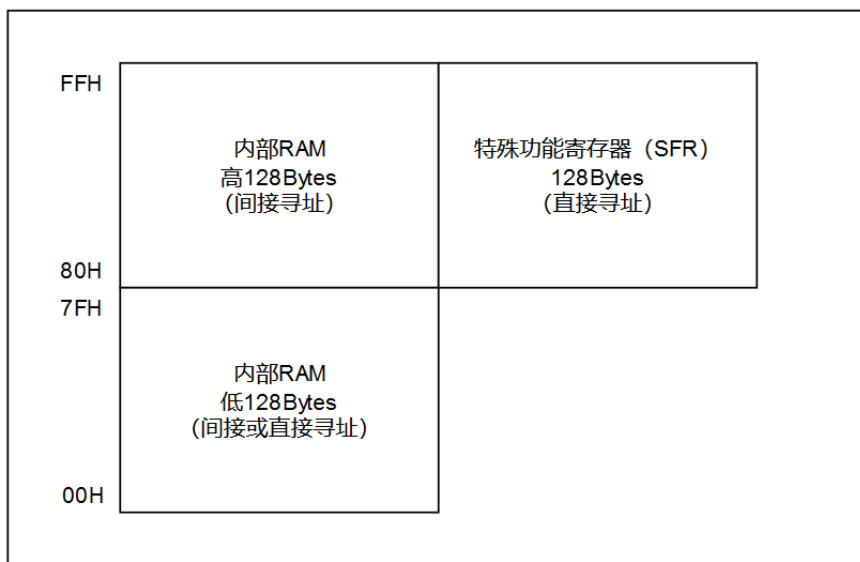


图 3-2 数据存储器地址映射

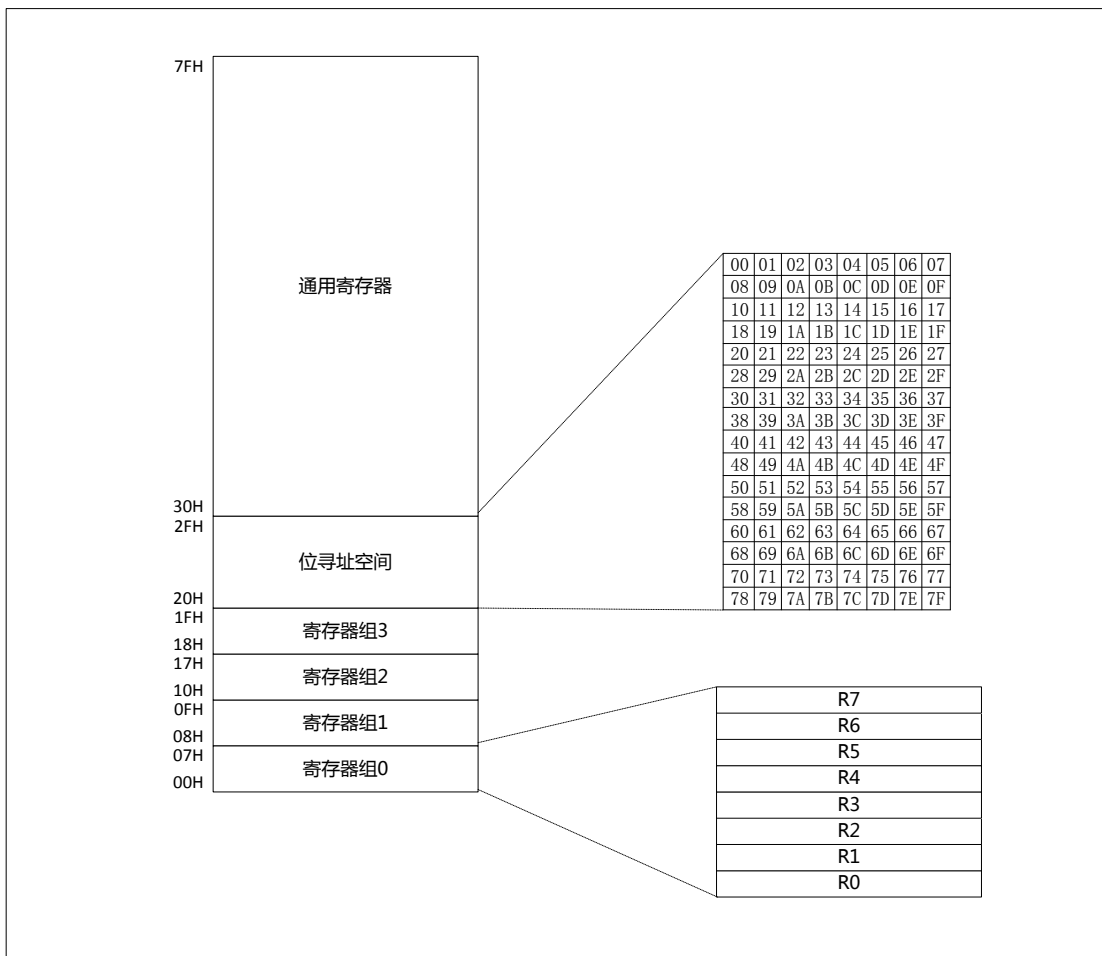


图 3-3 内部低 128 字节数据空间映射



3.4. SFR 空间

表 3-1 SFR 地址空间

	0H/8H	1H/9H	2H/AH	3H/BH	4H/CH	5H/DH	6H/EH	7H/FH
F8H	CMP1_CON0	UART1_DMAADDRH	UART1_DMAADDRL	UART1_DMALEN	ADC_CFG4	UART1_DMACON	–	FLASH_FUNCON
F0H	B	MDU_STA	UART1_CON	UART1_CON1	UART1_STA	UART1_BAUD0	UART1_BAUD1	UART1_DATA
E8H	TMR4_CAP20	TMR4_CAP21	TMR4_CAP30	TMR4_CAP31	TMR4_CAP40	TMR4_CAP41	TMR4_FLAG0	–
E0H	ACC	–	TMR4_CLR0	TMR4_CLR1	TMR4_CNT0	TMR4_CNT1	TMR4_CAP10	TMR4_CAP11
D8H	WDT_CON	WDT_KEY	TMR4_CON0	TMR4_CON1	TMR4_CON2	TMR4_CON3	TMR4_EN	TMR4_IE0
D0H	PSW	TMR_ALLCON	UART0_CON	UART0_CON1	UART0_STA	UART0_BAUD0	UART0_BAUD1	UART0_DATA
C8H	CMP0_CON0	SPI0_BAUD	SPI0_STA	SPI0_DAT	I2C_CON	I2C_DAT	I2C_ADR	I2C_STA
C0H	FLASH_LOCK	CRC_CON	CRC_REG	CRC_FIFO	CRC_DATA0	CRC_DATA1	CRC_DATA2	CRC_DATA3
B8H	IP0	SPI0_CON	LP_CON	FLASH_TRIM	FLASH_DMASTADR	FLASH_DMALEN	FLASH_BOOTCON	FLASH_ERRSTA
B0H	P3	IP1	IP2	IP3	IP4	IP5	IP6	IP7
A8H	IE0	IE1	IE2	IE3	FLASH_TIMERE01	FLASH_CRCLN	FLASH_PASSWORD	FLASH_ADDR
A0H	P2	ADC_TRGS2	ADC_CMPDATAH	ADC_CMPDATAL	FLASH_CON	FLASH_STA	FLASH_DATA	FLASH_TIMEREG0
98H	ADC_DATA12	ADC_CHS0	ADC_CHS1	SPH	PCON1	ADC_CHS2	ADC_TRGS0	ADC_TRGS1
90H	P1	ADC_CFG2	ADC_CFG3	ADC_DATAH0	ADC_DATA10	ADC_DATAH1	ADC_DATA11	ADC_DATAH2
88H	MDU_CTL0	MDU_CTL1	MDU_ADR0	MDU_ADR1	MDU_ADR2	WKUPCON0	WKPEN	SYSPEN
80H	P0	SP	DPL0	DPH0	DPL1	DPH1	DPCFG	PCON0



4. 时钟系统

4.1. 时钟系统概述

系统片上有一个 48MHz 的高速高精度 RC 振荡器，以及内部 PMU 里面集成了一个 64KHz 的低速 RC 振荡器，支持一个外接的 32.768KHz/8~40MHz 的晶体振荡器。

4.2. 时钟系统主要功能

TX8M226x 芯片的时钟源来自于 3 个不同的时钟，分别是片外 32.768KHz 的低速晶振或者 8~40MHz 的高速晶振，片内 64K 低速 RC 和片内 48M 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、CRC32 等外设都使用 sys_clk。如图 4-2 所示，GPIO 口的滤波时钟，WUT 模块时钟和低电检测这几个特殊的模块会使用 sys_clk，片外晶振，片内 64K 低速 RC，片内 48M 高速 RC 分频后时钟进行选择。

The clock tree diagram of the XMC4500 shows the following components and connections:

- rc64KHz** input splits to **DIV2** (outputting **wdt_clk** and **rc64k**) and **XSOCM**.
- 32.768KHz/4~40MHz** input splits to **XSOCM** and **hirc48MHz**.
- XSOCM** output splits to **DIV2** (outputting **xosc_div2**) and a 4-to-1 multiplexer. The multiplexer selects between **rc64k**, **xosc**, **hirc_div_clk**, and **hirc_clk** based on **CLK_CON2[3:0]** and **CLK_CON0[1:0]**. Its output is **sys_clk_pre**, which goes to **DIVX** (outputting **sys_clk**) and **SLEEP** (outputting **fclk**).
- hirc48MHz** input splits to **DIV2** and a 4-to-1 multiplexer. The multiplexer selects between the **DIV2** output, and three other **DIV2** outputs based on **CLK_CON6[7:6]**. Its output is **hirc_clk**, which goes to **CKG** (outputting **hirc_div_clk**) and **DIVX** (outputting **hirc_div_clk**).
- CLK_CON4[5]** selects between **fclk** and **LP_CON[7]** for **CKG** (outputting **sram0_clk**).
- CLK_CON3[2]** selects between **fclk** and **CLK_CON3[1]** for **CKG** (outputting **spi0_clk**).
- CLK_CON3[4]** selects between **fclk** and **CLK_CON3[4]** for **CKG** (outputting **crc_clk**).
- CLK_CON2[6]** selects between **fclk** and **CLK_CON2[6]** for **CKG** (outputting **mdm_clk**).
- CLK_CON3[5]** selects between **sys_clk_pre** and **timer0_clk** for **CKG** (outputting **timer0_clk**).
- CLK_CON3[6]** selects between **sys_clk_pre** and **timer1_clk** for **CKG** (outputting **timer1_clk**).
- CLK_CON4[0]** selects between **sys_clk_pre** and **sup_timer0_clk** for **CKG** (outputting **sup_timer0_clk**).
- CLK_CON4[1]** selects between **sys_clk_pre** and **sup_timer1_clk** for **CKG** (outputting **sup_timer1_clk**).
- CLK_CON4[2]** selects between **sys_clk_pre** and **sup_timer2_clk** for **CKG** (outputting **sup_timer2_clk**).
- CLK_CON5[6]** selects between **sys_clk_pre** and **tk_clk** for **CKG** (outputting **tk_clk**).
- CLK_CON4[3]** selects between **sys_clk_pre** and **adc_clk_pre** for **CKG** (outputting **adc_clk_pre**).
- CLK_CON2[5]** selects between **adc_clk_pre** and **adc_clk_pre_inv** for **INV** (outputting **adc_clk**).
- CLK_CON4[4]** selects between **sys_clk** and **eflash_mem_clk** for **CKG** (outputting **eflash_mem_clk**).
- CLK_CON6[5:0]** selects between **eflash_mem_clk** and **eflash_mem_clk** for **DIVX** (outputting **eflash_mem_clk**).
- CLK_CON0[6]** selects between **eflash_mem_clk** and **cmp_dbs_clk** for a 2-to-1 multiplexer (outputting **cmp_dbs_clk**).
- CLK_CON2[4]** selects between **eflash_mem_clk** and **led_clk** for **CKG** (outputting **cmp_hy_clk**).

图 4-1 系统时钟域时钟结构框图

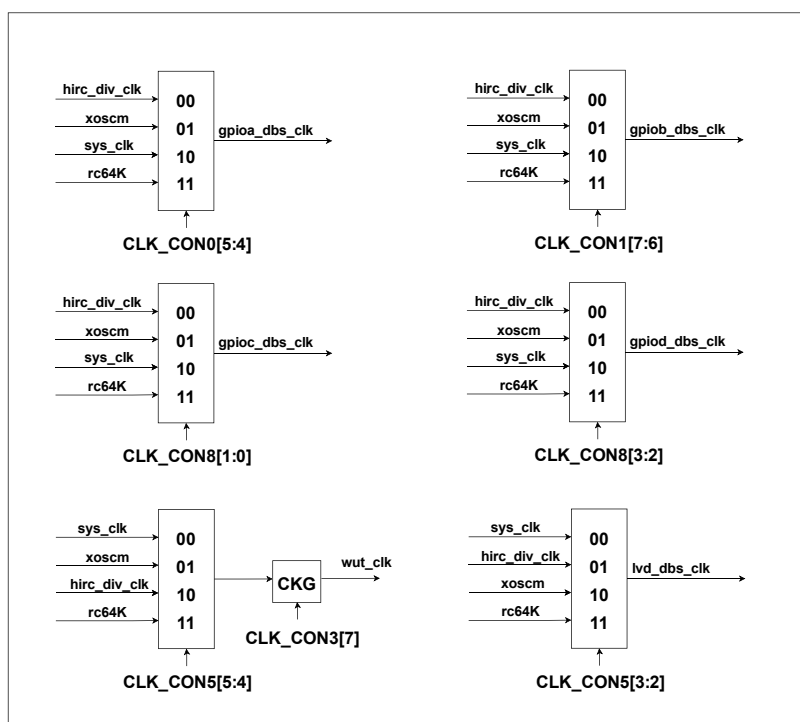


图 4-2 模块异步时钟域时钟结构框图

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 48MHz 的输出时钟给系统用，默认是关闭的，需要用户在程序中通过配置系统寄存器 CLK_ACON0[7]=1，打开高速 RC 振荡器，也可以通过配置 CLK_ACON0[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支持外部低速 32.768KHz 的无源晶体或者 8~40MHz 的高速无源晶体，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 CLK_XOSC[3]=1，打开 32.768KHz 的低速时钟源；也可以配置 CLK_XOSC[3]=0，关闭 32.768KHz 的低速时钟源。可以通过配置系统寄存器 CLK_XOSC[7]=1，打开 8~40MHz 的高速时钟源；也可以配置 CLK_XOSC[7]=0，关闭 8~40MHz 的高速时钟源。

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

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 7.8mS ~ 256S。可以选择看门狗产生中断，不复位。中断和复位只能二选一。

表 5-1 WDT 寄存器列表

Address	Register Name	Description
0xD8 (SFR)	WDT_CON	Watchdog Control Register
0xD9 (SFR)	WDT_KEY	Watchdog Key Register



5.3.1. WDT_CON

Addr = 0xD8 (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	WDT_PND	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	看门狗定时时间 每次配置该位域之前必须先写 WDT_KEY=0x55 0x0: 7.8125 毫秒钟 0x1: 15.625 毫秒钟 0x2: 31.25 毫秒钟 0x3: 62.5 毫秒钟 0x4: 125 毫秒钟 0x5: 250 毫秒钟	RW	0x8



		0x6: 500 毫秒钟		
		0x7: 1 秒钟		
		0x8: 2 秒钟		
		0x9: 4 秒钟		
		0xA: 8 秒钟		
		0xB: 16 秒钟		
		0xC: 32 秒钟		
		0xD: 64 秒钟		
		0xE: 128 秒钟		
		0xF: 256 秒钟		

5.3.2. WDT_KEY

Addr = 0xD9 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	WDT_KEY	喂狗数据寄存器 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 Note: 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!	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。

表 5-2 LVD 寄存器列表

Address	Register Name	Description
0x15A (XSFR)	LVD_CON0	LVD_CON0 register
0x15B (XSFR)	LVD_CON1	LVD_CON1 register
0x15C (XSFR)	LVD_CON2	LVD_CON2 register
0x15D (XSFR)	LVD_CON3	LVD_CON3 register

5.4.1. LVD_CON0

Addr = 0x15A (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	LVD0E	LVD 中断和复位功能输出到系统使能位 Note: 使用 LVD 所有相关功能，必须把 LVD0E 设置位 1 0x0: 关闭 0x1: 打开	RW	0x1
6	LVDVDDRSTEN	LVD VDD 低电压复位功能使能位 0x0: 低电中断功能使能 0x1: 低电复位功能使能	RW	0x1
5	LVDVCCRSTEN	LVD VCC 低电压复位功能使能位 0x0: 低电中断功能使能	RW	0x1



		0x1: 低电复位功能使能		
4: 2	PMULVD5SET	VCCA 电源电压低电检测阈值设置 0x0: 2.0V 0x1: 2.2V 0x2: 2.4V 0x3: 2.7V 0x4: 3.0V 0x5: 3.7V 0x6: 4.0V 0x7: 4.3V	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. LVD_CON1

Addr = 0x15B (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	—	—	—	—
6	VDDOCPND	VDD 过流检测标记位 0x0: VDD 没有过流 0x1: VDD 过流 Note: 写 1 清除标记位!	RW	0x0
5	LVDVDDPND	VDD 低电检测标记位 0x0: VDD 没有低电 0x1: VDD 低电 Note: 写 1 清除标记位!	RW	0x0
4	LVDVCCPND	VCC 低电检测标记位	RW	0x0



		0x0: VCC 没有低电 0x1: VCC 低电 Note: 写 1 清除标记位!		
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

5.4.3. LVD_CON2

Addr = 0x15C (XSFR)

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



5.4.4. LVD_CON3

Addr = 0x15D (XSFR)

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

TX8M226x 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流可以低至 5uA。

6.1. Idle Mode 及唤醒

通过配置系统寄存器 LP_CON[7]=0，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 电平变化触发唤醒、比较器唤醒、WUT 唤醒、看门狗唤醒、触摸按键中断唤醒、LVDVCC(电源的低电压检测信号)中断唤醒。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 以外的模拟模块都可以关闭，关闭所有时钟源包括 HRCOSC、LRCOSC、XOSC。通过选择多种唤醒源来唤醒 Stop Mode，唤醒源包括：所有 GPIO 电平变化触发唤醒、比较器唤醒、WUT 定时中断唤醒、LVDVCC(电源的低电压检测信号)中断唤醒。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。

如果在进入低功耗 Sleep 模式之前打开看门狗复位模式，则当看门狗计时到了以后会复位整个系统重新跑用户程序。建议用户在进入低功耗 Sleep 模式之前喂狗，并根据用户应用



场景设置看门狗计时长度，并打开看门狗复位工作模式，则可以保护低功耗模式下发生小概率不可预测的外部环境造成的异常情况。

6.4. 低功耗唤醒单元结构图

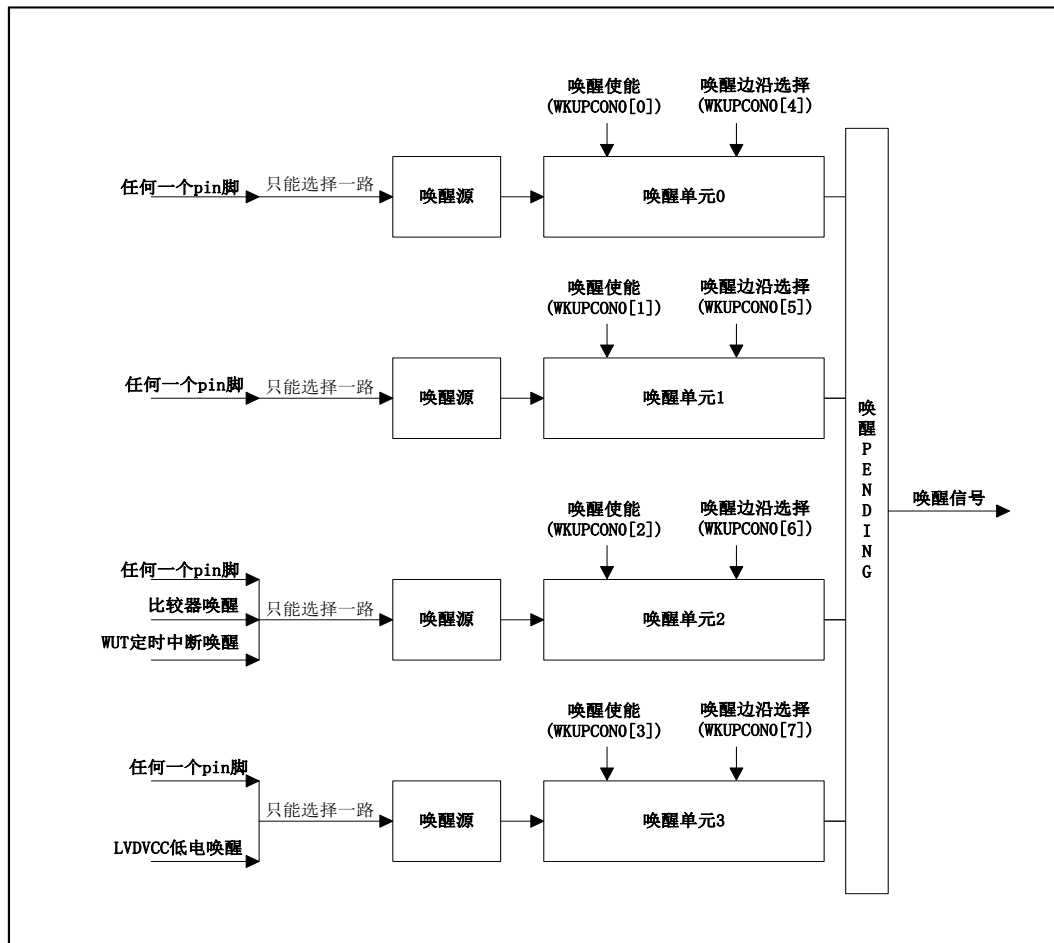


图 6-1 低功耗唤醒结构图

6.5. 寄存器详细说明

表 6-1 低功耗和唤醒功能配置寄存器列表

Address	Register Name	Description
0x0D (SFR)	WKUP_CON0	WKUP_CON0 register
0x0E (SFR)	WKUP_PND	WKUP_PND register



0x3A (SFR)	LP_CON	LP_CON register
0x116 (XSFR)	SYS_CON6	SYS_CON6 register
0x117 (XSFR)	SYS_CON7	SYS_CON7 register
0x118 (XSFR)	SYS_CON8	SYS_CON8 register

6.5.1. WKUP_CON0

Addr = 0x0D (SFR)

Bit(s)	Name	Description	R/W	Reset
7	WKUP3EDG	低功耗 SLEEP 模式唤醒通道 3 触发条件设置 0x0: 高电平触发唤醒 0x1: 低电平触发唤醒	RW	0x0
6	WKUP2EDG	低功耗 SLEEP 模式唤醒通道 2 触发条件设置 0x0: 高电平触发唤醒 0x1: 低电平触发唤醒	RW	0x0
5	WKUP1EDG	低功耗 SLEEP 模式唤醒通道 1 触发条件设置 0x0: 高电平触发唤醒 0x1: 低电平触发唤醒	RW	0x0
4	WKUP0EDG	低功耗 SLEEP 模式唤醒通道 0 触发条件设置 0x0: 高电平触发唤醒 0x1: 低电平触发唤醒	RW	0x0
3	WKUP3EN	低功耗 SLEEP 模式唤醒通道 3 功能使能位 0x0: 关闭 0x1: 打开	RW	0x0
2	WKUP2EN	低功耗 SLEEP 模式唤醒通道 2 功能使能位 0x0: 关闭 0x1: 打开	RW	0x0
1	WKUP1EN	低功耗 SLEEP 模式唤醒通道 1 功能使能位 0x0: 关闭 0x1: 打开	RW	0x0



0	WKUP0EN	低功耗 SLEEP 模式唤醒通道 0 功能使能位 0x0: 关闭 0x1: 打开	RW	0x0

6.5.2. WKUP_PND

Addr = 0x0E (SFR)

Bit(s)	Name	Description	R/W	Reset
7	WKUP3PCLR	低功耗 SLEEP 模式唤醒通道 3 清 pending 位 0x0: 无操作 0x1: 清 pending	RW	0x0
6	WKUP2PCLR	低功耗 SLEEP 模式唤醒通道 2 清 pending 位 0x0: 无操作 0x1: 清 pending	RW	0x0
5	WKUP1PCLR	低功耗 SLEEP 模式唤醒通道 1 清 pending 位 0x0: 无操作 0x1: 清 pending	RW	0x0
4	WKUP0PCLR	低功耗 SLEEP 模式唤醒通道 0 清 pending 位 0x0: 无操作 0x1: 清 pending	RW	0x0
3	WKUP3PND	低功耗 SLEEP 模式唤醒通道 3 唤醒 pending 位 0x0: 无 pending 0x1: 有 pending	RW	0x0
2	WKUP2PND	低功耗 SLEEP 模式唤醒通道 2 唤醒 pending 位 0x0: 无 pending 0x1: 有 pending	RW	0x0
1	WKUP1PND	低功耗 SLEEP 模式唤醒通道 1 唤醒 pending 位 0x0: 无 pending 0x1: 有 pending	RW	0x0
0	WKUP0PND	低功耗 SLEEP 模式唤醒通道 0 唤醒 pending 位 0x0: 无 pending 0x1: 有 pending	RW	0x0



6.5.3. LP_CON

Addr = 0x3A (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, 停 CPU 时钟 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 0x0: 打开 (默认状态) 0x1: 关闭 Note: CPU 和 ISD 都无法写寄存器!	RO	0x0
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 WUT 定时唤醒场景下不可以关闭该时钟, 造成 WUT 没有工作时钟而不能唤醒系统。	RW	0x0
3	ISDDISLPEN	ISD 模式下低功耗功能关闭位 0x0: 使能 0x1: 关闭 Note: 该寄存器配置为 0x1 时, ISD 调试模式下进不了 SLEEP 低功耗模式!	RW	0x1
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!!!	RW	0x0



		0x0: 不 hold CPU 0x1: hold CPU		
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
0x110 (XSFR)	SYS_CON0	SYS_CON0 register
0x111 (XSFR)	SYS_CON1	SYS_CON1 register
0x112 (XSFR)	SYS_CON2	SYS_CON2 register
0x113 (XSFR)	SYS_CON3	SYS_CON3 register
0x114 (XSFR)	SYS_CON4	SYS_CON4 register
0x115 (XSFR)	SYS_CON5	SYS_CON5 register
0x116 (XSFR)	SYS_CON6	SYS_CON6 register
0x117 (XSFR)	SYS_CON7	SYS_CON7 register
0x118 (XSFR)	SYS_CON8	SYS_CON8 register



0x0F (SFR)	SYS_PND	SYS_PND register
0x15E (XSFR)	IO_MAP	IO_MAP register
0x162 (XSFR)	CLK_XOSC	XOSC Control register (晶振配置寄存器)
0x120 (XSFR)	CLK_ACON0	CLK_ACON0 register
0x121 (XSFR)	CLK_ACON1	CLK_ACON1 register
0x122 (XSFR)	ADC_ACON0	ADC_ACON0 register (ADC 模拟相关配置)
0x123 (XSFR)	ADC_ACON1	ADC_ACON1 register (ADC 模拟相关配置)
0x124 (XSFR)	ADC_ACON2	ADC_ACON2 register (ADC 模拟相关配置)
0x130 (XSFR)	CLK_CON0	CLK_CON0 register
0x131 (XSFR)	CLK_CON1	CLK_CON1 register
0x132 (XSFR)	CLK_CON2	CLK_CON2 register
0x133 (XSFR)	CLK_CON3	CLK_CON3 register
0x134 (XSFR)	CLK_CON4	CLK_CON4 register
0x135 (XSFR)	CLK_CON5	CLK_CON5 register
0x136 (XSFR)	CLK_CON6	CLK_CON6 register
0x137 (XSFR)	CLK_CON7	CLK_CON7 register (ADC 模拟相关配置)
0x138 (XSFR)	CLK_CON8	CLK_CON8 register

7.3. 寄存器详细说明

7.3.1. SYS_CON0

Addr = 0x110 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	STMROSOFTRST	高级 Timer0 软复位	RW	0x1



		0x0: 软复位 0x1: 软复位释放		
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 0x1: 软复位释放	RW	0x1
3	I2CSOFTRST	I2C 软复位 0x0: 软复位 0x1: 软复位释放	RW	0x1
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 = 0x111 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	I0DBSSFTRST	GPI0 Debounce 模块软复位 0x0: 软复位 0x1: 软复位释放	RW	0x1
6	—	—	RW	0x1



5	GPIO SofTrst	GPIO 模块软复位 0x0: 软复位 0x1: 软复位释放	RW	0x1
4	ADC SofTrst	ADC 软复位 0x0: 软复位 0x1: 软复位释放	RW	0x1
3	WDT SofTrst	Watchdog 软复位 0x0: 软复位 0x1: 软复位释放	RW	0x1
2	CRC SofTrst	CRC 软复位 0x0: 软复位 0x1: 软复位释放	RW	0x1
1	STM2 SofTrst	高级 Timer2 软复位 0x0: 软复位 0x1: 软复位释放	RW	0x1
0	STM1 SofTrst	高级 Timer1 软复位 0x0: 软复位 0x1: 软复位释放	RW	0x1

7.3.3. SYS_CON2

Addr = 0x112 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	TMR4CAPRCEN	Timer4 捕获 rc64k_div8 使能位 0x0: 关闭 0x1: 打开	RW	0x0
6	XOSCHWENSEL	XOSC 硬件使能功能选择位 0x0: 关闭 0x1: 打开	RW	0x0
5	TMR24IREN	基本 Timer2 与 Timer4 联合完成红外发送功能使能位 0x0: 关闭	RW	0x0



		0x1: 打开 Note: 基本 Timer2 作为载波 PWM, Timer4 通道 A 作为调制波 PWM		
4	LVDVCCWKEN	LVDVCC 唤醒使能位 0x0: 关闭 0x1: 打开	RW	0x0
3	ISPIODEBEN	ISP IO 输入滤波功能使能位 0x0: 关闭 0x1: 打开	RW	0x0
2	—	—	—	—
1	—	—	—	—
0	MDUSOFTRST	MDU 模块软复位 0x0: 软复位 0x1: 软复位释放	RW	0x1

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 = 0x113 (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	RW	0x0



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

7.3.5. SYS_CON4

Addr = 0x114 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P17DBSEN	P17 输入滤波功能使能位 0x0: 关闭 0x1: 打开	RW	0x0
6	P16DBSEN	P16 输入滤波功能使能位 0x0: 关闭 0x1: 打开	RW	0x0
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

7.3.6. SYS_CON5

Addr = 0x115 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P27DBSEN	P27 输入滤波功能使能位 0x0: 关闭 0x1: 打开	RW	0x0
6	P26DBSEN	P26 输入滤波功能使能位 0x0: 关闭 0x1: 打开	RW	0x0
5	P25DBSEN	P25 输入滤波功能使能位 0x0: 关闭 0x1: 打开	RW	0x0
4	P24DBSEN	P24 输入滤波功能使能位 0x0: 关闭 0x1: 打开	RW	0x0



3	P23DBSEN	P23 输入滤波功能使能位 0x0: 关闭 0x1: 打开	RW	0x0
2	P22DBSEN	P22 输入滤波功能使能位 0x0: 关闭 0x1: 打开	RW	0x0
1	P21DBSEN	P21 输入滤波功能使能位 0x0: 关闭 0x1: 打开	RW	0x0
0	P20DBSEN	P20 输入滤波功能使能位 0x0: 关闭 0x1: 打开	RW	0x0

7.3.7. SYS_CON6

Addr = 0x116 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 4	MEMDVS[3: 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！！！！	-	-
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 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） Note: 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。	RW	0x3
1	P31DBSEN	P31 输入滤波功能使能位	RW	0x0



		0x0: 关闭 0x1: 打开		
0	P30DBSEN	P30 输入滤波功能使能位 0x0: 关闭 0x1: 打开	RW	0x0

7.3.8. SYS_CON7

Addr = 0x117 (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 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） Note: 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。	RW	0x0
5: 4	FLASHUPCNT	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 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） Note: 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。	RW	0x0
3: 2	OPMLDOCNT	低功耗 Sleep Mode 流程打开主 LDO 延迟时间配置 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） Note: 低功耗模式进入之前必须将系统时钟切换	RW	0x0



		成低速的 64KHz 的 RC，所以延迟时间= $n \times T_{64k}$ 。		
1: 0	CLSMLDOCNT	低功耗 Sleep Mode 流程关闭主 LDO 延迟时间配置 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） Note: 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= $n \times T_{64k}$ 。	RW	0x0

7.3.9. SYS_CON8

Addr = 0x118 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	LPSLPDISANA	低功耗 sleep mode 一键关模拟模块功能使能位 0x0: 关闭 0x1: 打开 Note: 通过一键关闭模拟模块，可以节省进入低功耗 sleep 模式前的程序代码，简化关闭模拟模块的流程。一键关闭模拟模块包括：包括触摸按键，运放，比较器，ADC。	RW	0x0
6	MEMDVSE	片上 SRAM 动态电压调整使能位 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
5	FASTRSTEN	快速复位唤醒 Sleep Mode 使能 0x0: 关闭 0x1: 打开 Note: 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	RW	0x0
4	SPIOMAP2EN	SPIO io map2 的使能位	RW	0x0



		0x0: 关闭 0x1: 打开		
3	SPIOMAP1EN	SPI0 io map1 的使能位 0x0: 关闭 0x1: 打开	RW	0x0
2	DBGEN	DEBUG 功能使能位 0x0: 关闭 0x1: 打开	RW	0x0
1: 0	PMUREV54	PMU 保留寄存器的位 5: 4 的值	RW	0x0

7.3.10. SYS_PND

Addr = 0x0F (SFR)

Bit(s)	Name	Description	R/W	Reset
7	—	—	—	—
6	SFTRST1CLR	写 1 清掉系统软复位 1 标志位	RW	0x0
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	SFTRST1PND	系统软复位 1 标志位 写 1 系统软复位。	RW	0x0
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

7.3.11. IO_MAP

Addr = 0x15E (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	STMRCMPCEQEN	高级 timer 比较点 C 相等触发 timer0 功能使 能 0x0: 关闭	RW	0x0



		0x1: 使能		
6	–	–	RW	0x0
5	MCLRMAP3EN	MCLR MAP3 选择位 0x0: 不选择 P15 0x1: 选择 P15	RW	0x0
4	MCLRMAP2EN	MCLR MAP2 选择位 0x0: 不选择 P25 0x1: 选择 P25	RW	0x0
3	MCLRMAP1EN	MCLR MAP1 选择位 0x0: 不选择 P23 0x1: 选择 P23	RW	0x0
2	LEDDMAEN	LED DMA 使能位 0x0: 关闭 0x1: 使能 Note: 该位寄存器配置前需要打开保护 KEY， 否则配置不了。寄存器 WDT_KEY = 0x55，则关 闭寄存器写保护功能，配置完这位寄存器后， 需寄存器 WDT_KEY = 0x00，则打开寄存器写保 护功能！	RW	0x0
1	MCLREN	MCLR 功能使能位 0x0: 关闭 0x1: 使能 Note: 该位寄存器配置前需要打开保护 KEY， 否则配置不了。寄存器 WDT_KEY = 0x55，则关 闭寄存器写保护功能，配置完这位寄存器后， 需寄存器 WDT_KEY = 0x00，则打开寄存器写保 护功能！	RW	0x0
0	ISPMAP1	烧写/调试 pin 脚选择位 0x0: 不选择 0x1: 选择 P31【ISP_CLK】，P17【ISP_DAT】 Note: 该位寄存器配置前需要打开保护 KEY， 否则配置不了。寄存器 WDT_KEY = 0x55，则关 闭寄存器写保护功能，配置完这位寄存器后，	RW	0x1



		需寄存器 WDT_KEY = 0x00，则打开寄存器写保护功能！		

7.3.12. CLK_XOSC

Addr = 0x162 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	HXOSCEN	高速晶振使能 0x0: 不使能 0x1: 使能	RW	0x0
6: 4	HXOSCDR	高速晶振驱动能力选择 0x0: x2 0x1: x3 0x2: x4 0x3: x5 0x4: x6 0x5: x7 0x6: x8 0x7: x9	RW	0x3
3	LXOSCEN	32.768KHz 低速晶振使能 0x0: 不使能 0x1: 使能	RW	0x0
2: 0	LXOSCDR	32.768KHz 低速晶振驱动能力选择 0x0: X1 0x1: X2 0x2: X3 0x3: X4 0x4: X5 0x5: X6 0x6: X7 0x7: X8	RW	0x3



7.3.13. CLK_ACON0

Addr = 0x120 (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=0.5%) 0x00: low 0x7F: high	RW	0x48

7.3.14. CLK_ACON1

Addr = 0x121 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	XOSCHY	输出时钟迟滞窗口选择 0x0: 没有迟滞 0x1: 有+/-10%的迟滞	RW	0x1
6	HRCTESTEN	HRC 内部模拟电压测试使能信号 0x0: 关闭 0x1: 打开	RW	0x0
5: 4	HRCTEMPSEL	HRC 时钟频率温度系数调节 0x0: -1.17% (100℃) 0x1: -0.82% (65℃) 0x2: -0.47% (40℃) 0x3: +0.67% (20℃)	RW	0x1
3: 2	HRCSCADD	HRC 时钟频率细调 (step=0.3%) 0x0: low 0x3: high	RW	0x1



1: 0	HRC SR	HRC 时钟频率粗调 0x0: 24MHz 0x1: 48MHz 0x2: 48MHz 0x3: 72MHz	RW	0x1

7.3.15. ADC_ACON0

Addr = 0x122 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	—	—	—	—
6: 4	ADCDETCHSEL	ADC 内部检测信号选择 0x0: 保留, 未定义 0x1: VREF_OP6 0x2: 保留, 未定义 0x3: VCCA_D5 0x4: AMP0 (运放 0 的输出) 0x5: AMP1 (运放 1 的输出) 0x6: AMP2 (运放 2 的输出) 0x7: 不使能内部检测信号	RW	0x7
3	ADCBIASSEL	ADC 偏置电流选择位 0x0: 1.25X 0x1: 1X Note: ADC 测试用信号, 用户不用关注此配置, 使用时用默认值即可!	RW	0x0
2	ADCINREFSEL	ADC 输入基准参考电压选择位 0x0: 0.6V 0x1: 1.05V;	RW	0x1
1	ADCBIASEN	ADC 偏置电流使能信号 0x0: 关闭 0x1: 打开 Note: ADC 模拟电路要正常工作, 必须使能内部偏置电流模块, 即必须配置该寄存器为 1!	RW	0x0



0	ADCCMPEN	ADC 中 CMP 使能信号 0x0: 关闭 0x1: 打开 Note: ADC 模拟电路要正常工作，必须使能内部比较器，即必须配置该寄存器为 1！	RW	0x0

7.3.16. ADC_ACON1

Addr = 0x123 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	ADCTRIMIBSEL	ADC 校准电流选择 0x0: 1X 0x1: 2X Note: 芯片出厂时已经经过校准，校准值用户程序需要在对应 NVR 存储器中固定位置读取出来配置到该寄存器！	RW	0x0
6	ADCSELINREF	ADC 中内部参考选择信号 0x0: 关闭内部参考 0x1: 选择内部参考 Note: 选择内部参考时，必须先断开外部参考！	RW	0x1
5	ADCSELEXREF	ADC 外部参考选择信号 0x0: 关闭外部参考 0x1: 选择外部 EXREF (P07) 为参考电压 Note: 选择外部参考时，必须选关闭内部参考！	RW	0x0
4: 3	ADCTENSEL	ADC 测试信号选择 0x0: 测试信号 0x1: 保留 0x2: 保留 0x3: 关闭测试信号 Note: 用户在使用 ADC 时，需要确保关闭所有测试信号，即保持默认值 0x3 不改变！	RW	0x3
2: 0	ADCVREFSEL	ADC 中内部参考电压选择信号	RW	0x1



		0x0: 保留, 未定义 0x1: 2.0V (未校准) 0x2: 2.4V 0x3: 3.0V (未校准) 0x4: 3.6V (未校准) 0x5: 4.2V (未校准) 0x6: VCCA 0x7: 保留, 未定义 Note: 当使用 VCCA 作为参考时, 需要进行以下配置: ADCSELINREF=0 ADCSELEXREF=0; 当使用其他内部参考时, 需要进行以下配置: ADCSELINREF=1 ADCSELEXREF=0;		

7.3.17. ADC_ACON2

Addr = 0x124 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	–	–	RW	0x0
6	ADCCMPTRIMEN	ADC 比较器校准功能使能信号 0x0: 关闭 0x1: 打开 Note: 芯片 ADC 校准值生效使能, 用户程序从对应 NVR 存储器固定位置读取 ADC 校准值, 并配置到对应的寄存器以后, 必须打开这个使能信号, 才能生效!	RW	0x0
5: 0	ADCCMPTRIM	ADC 比较器校准值配置位 MSB: 符号位, 低 5 为数值。 Note: 芯片出厂时已经经过校准, 校准值用户程序需要在对应 NVR 存储器中固定位置读取出	RW	0x0



来配置到该寄存器！

7.3.18. CLK_CON0

Addr = 0x130 (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	PODBSCLKSEL	P0 滤波时钟选择位 0x0: 选择 hirc_div_clk 0x1: 选择 xosc 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 xosc	RW	0x0
1: 0	SYCLKSEL	系统时钟选择位 0x0: 选择 rc64k 0x1: 选择 xosc 0x2: 选择 hirc_div_clk 0x3: 选择 hirc_clk	RW	0x0

7.3.19. CLK_CON1

Addr = 0x131 (XSFR)

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



		0x0: 选择 hirc_div_clk 0x1: 选择 xoscm 0x2: 选择 sys_clk 0x3: 选择 rc64k		
5: 3	HIRCCLKDIV	高速 HRCOSC 时钟源分频设置 0x0: 不分频 0x1: 2 分频 0x2: 3 分频 0x6: 7 分频 0x7: 关闭 Note: 配置比为 n+1 时钟。	RW	0x6
2: 0	CLKTOIODIV	I/O 输出时钟源分频设置 0x0: 不分频 0x1: 2 分频 0x2: 3 分频 0x6: 7 分频 0x7: 关闭 Note: 配置比为 n+1 时钟。	RW	0x0

7.3.20. CLK_CON2

Addr = 0x132 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	—	—	—	—
6	MDUCLKEN	MDU 模块时钟使能位 0x0: 关闭时钟 0x1: 打开时钟	RW	0x1
5	ADCCLKSEL	Adc clk 时钟选择位 0x0: 选择 adc_clk_pre 0x1: 选择 adc_clk_pre_inv	RW	0x0



4	LEDCLKEN	LED clk 时钟使能位 0x0: 关闭时钟 0x1: 打开时钟	RW	0x0
3: 0	SYSCLKDIV	系统时钟分频设置 0x0: 不分频 0x1: 2 分频 0x2: 3 分频 0xE: 15 分频 0xF: 关闭 Note: 配置比为 n+1 时钟。	RW	0x0

7.3.21. CLK_CON3

Addr = 0x133 (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	TMROCLKEN	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	I2CCLKEN	I2C 模块时钟使能位 0x0: 关闭时钟 0x1: 打开时钟	RW	0x1
2	SPI0CLKEN	SPI0 模块时钟使能位	RW	0x1



		0x0: 关闭时钟 0x1: 打开时钟		
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

7.3.22. CLK_CON4

Addr = 0x134 (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	SPIDISYEN	SPI 数据线输入同步使能 0x0: 数据线输入不经过同步器 0x1: 数据线输入经过同步器	RW	0x1
5	RAMCLKEN	片上 SRAM 时钟使能位 0x0: 关闭时钟 0x1: 打开时钟	RW	0x1
4	AHB1CLKEN	AHB1 CLK 时钟使能位 0x0: 关闭时钟 0x1: 打开时钟	RW	0x1
3	ADCCLKEN	ADC 模块时钟使能位 0x0: 关闭时钟 0x1: 打开时钟	RW	0x1
2	STMR2CLKEN	高级 Timer2 模块时钟使能位 0x0: 关闭时钟 0x1: 打开时钟	RW	0x1
1	STMR1CLKEN	高级 Timer1 模块时钟使能位	RW	0x1



		0x0: 关闭时钟 0x1: 打开时钟		
0	STMROCLKEN	高级 Timer0 模块时钟使能位 0x0: 关闭时钟 0x1: 打开时钟	RW	0x1

7.3.23. CLK_CON5

Addr = 0x135 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	HIRCCLKEN	hirc_clk 时钟使能位 0x0: 关闭时钟 0x1: 打开时钟	RW	0x1
6	—	—	RW	0x1
5: 4	WUTCLKSEL	WUT 模块时钟选择位 0x0: 选择 sys_clk 0x1: 选择 xosc 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 0x1: 选择 hirc_div_clk 0x2: 选择 xosc 0x3: 选择 rc64k	RW	0x0
1	—	—	—	—
0	TCLKEN	测试时钟 1 使能位 0x0: 关闭时钟 0x1: 打开时钟	RW	0x0



7.3.24. CLK_CON6

Addr = 0x136 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	HIRCDIVSEL	Hirc clk 时钟分频选择 0x0: 不分频 0x1: 2 分频 0x2: 4 分频 0x3: 8 分频	RW	0x0
5: 0	MCLKDIV	存储器烧写时钟分频设置 0x00: 不分频 0x01: 2 分频 0x02: 3 分频 0x3E: 63 分频 0x3F: 关闭 Note: 配置比为 n+1 时钟。	RW	0x0

7.3.25. CLK_CON7

Addr = 0x137 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	ADCTRMCLKSEL	ADC_TRIMCKSEL_VDD 配置 0x0: Normal 0x1: Digital/软件校准时选 CK_VDD 直接作为时钟	RW	0x0
6: 0	—	—	—	—

7.3.26. CLK_CON8

Addr = 0x138 (XSFR)



Bit(s)	Name	Description	R/W	Reset
7	ADC_SFTRM_EN	ADC 软件 trim 功能使能位 0x0: 选择关闭 0x1: 选择打开	RW	0x0
6	ADC_SFTRM_SOC	ADC 软件 trim SOC 控制位	RW	0x0
5	ADC_SFTRM_TRM	ADC 软件 trim trimming 值控制位	RW	0x0
4	ADC_SFTRM_CLK	ADC 软件 trim 时钟控制位	RW	0x0
3: 2	P3DBSCLKSEL	P3 滤波时钟选择位 0x0: 选择 hirc_div_clk 0x1: 选择 xosc 0x2: 选择 sys_clk 0x3: 选择 rc64k	RW	0x0
1: 0	P2DBSCLKSEL	P2 滤波时钟选择位 0x0: 选择 hirc_div_clk 0x1: 选择 xosc 0x2: 选择 sys_clk 0x3: 选择 rc64k	RW	0x0



8. 中断系统

8.1. 中断概述

TX8M226x 支持多达 29 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。通过设置 DPCFG[7: 6] 可调节中断起始地址。首个中断地址为：起始地址+（中断向量号+1）*3。

中断控制器有以下特性：

- 从 29 个中断源接收中断
- 支持 2 级嵌套，等级越高优先级越高
- 中断向量号起始地址可选

8.2. 中断向量表

表 8-1 中断向量表

Name	Physical Base Address	Base Address
INT_BASE	000CH/080CH/100CH/600CH	中断向量起始地址，可配置

Address	Register Name	Description
000FH	LVD	LVD Interrupt
0012H	GPI00	GPI00 Interrupt
0015H	GPI01	GPI01 Interrupt
0018H	GPI02	GPI02 Interrupt
001BH	GPI03	GPI03 Interrupt
001EH	TMR0	Timer0 Interrupt
0021H	TMR1	Timer1 Interrupt
0024H	TMR2	Timer2 Interrupt
0027H	TMR3	Timer3 Interrupt



002AH	TMR4	Timer4 Interrupt
002DH	STMR0	Super Timer0 Interrupt
0030H	STMR1	Super Timer1 Interrupt
0033H	STMR2	Super Timer2 Interrupt
0036H	STMR3	Super Timer3 Interrupt
0039H	STMR4	Super Timer4 Interrupt
003CH	STMR5	Super Timer5 Interrupt
003FH	WUT	Wake Up Timer Interrupt
0042H	ADC	ADC Interrupt
0045H	COMP	Comparer Interrupt
0048H	I2C	I2C Interrupt
004BH	UART0	UART0 Interrupt
004EH	UART1	UART1 Interrupt
0051H	SPI	SPI Interrupt
0054H	WKPND	Low Power Wake Up Interrupt
0057H	WDT	WatchDog Interrupt
005AH	BUZ/FLASH	BUZ/FLASH Interrupt
005DH	—	—
0060H	AMP	AMP Interrupt
0063H	LED	LED Interrupt

注：此向量表以 0000H 为起始基地址；如选择不同的起始地址，请根据计算方式加上相应的偏移量。



8.3. 寄存器列表

Address	Register Name	Description
0xA8 (SFR)	IE0	Interrupt Enable 0 Register
0xA9 (SFR)	IE1	Interrupt Enable 1 Register
0xAA (SFR)	IE2	Interrupt Enable 2 Register
0xAB (SFR)	IE3	Interrupt Enable 3 Register
0xB8 (SFR)	IP0	Interrupt Priority 0 Register
0xB1 (SFR)	IP1	Interrupt Priority 1 Register
0xB2 (SFR)	IP2	Interrupt Priority 2 Register
0xB3 (SFR)	IP3	Interrupt Priority 3 Register
0xB4 (SFR)	IP4	Interrupt Priority 4 Register
0xB5 (SFR)	IP5	Interrupt Priority 5 Register
0xB6 (SFR)	IP6	Interrupt Priority 6 Register
0xB7 (SFR)	IP7	Interrupt Priority 7 Register

8.4. 寄存器详细说明

8.4.1. 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: 允许所有中断	RW	0x0
6	TMR1	TMR1 中断使能 0x0: 禁止 TMR1 中断	RW	0x0



		0x1: 允许 TMR1 中断		
5	TMR0	TMR0 中断使能 0x0: 禁止 TMR0 中断 0x1: 允许 TMR0 中断	RW	0x0
4	GPI03	GPI03 中断使能 0x0: 禁止 GPI03 中断 0x1: 允许 GPI03 中断	RW	0x0
3	GPI02	GPI02 中断使能 0x0: 禁止 GPI02 中断 0x1: 允许 GPI02 中断	RW	0x0
2	GPI01	GPI01 中断使能 0x0: 禁止 GPI01 中断 0x1: 允许 GPI01 中断	RW	0x0
1	GPI00	GPI00 中断使能 0x0: 禁止 GPI00 中断 0x1: 允许 GPI00 中断	RW	0x0
0	LVD	LVD 中断使能 0x0: 禁止 LVD 中断 0x1: 允许 LVD 中断	RW	0x0

8.4.2. IE1

Addr = 0xA9 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	STMR4	STMR4 中断使能 0x0: 禁止 STMR4 中断 0x1: 允许 STMR4 中断	RW	0x0
6	STMR3	STMR3 中断使能 0x0: 禁止 STMR3 中断 0x1: 允许 STMR3 中断	RW	0x0
5	STMR2	STMR2 中断使能 0x0: 禁止 STMR2 中断	RW	0x0



		0x1: 允许 STMR2 中断		
4	STMR1	STMR1 中断使能 0x0: 禁止 STMR1 中断 0x1: 允许 STMR1 中断	RW	0x0
3	STMR0	STMR0 中断使能 0x0: 禁止 STMR0 中断 0x1: 允许 STMR0 中断	RW	0x0
2	TMR4	TMR4 中断使能 0x0: 禁止 TMR4 中断 0x1: 允许 TMR4 中断	RW	0x0
1	TMR3	TMR3 中断使能 0x0: 禁止 TMR3 中断 0x1: 允许 TMR3 中断	RW	0x0
0	TMR2	TMR2 中断使能 0x0: 禁止 TMR2 中断 0x1: 允许 TMR2 中断	RW	0x0

8.4.3. IE2

Addr = 0xAA (SFR)

Bit(s)	Name	Description	R/W	Reset
7	SPI	SPI 中断使能 0x0: 禁止 SPI 中断 0x1: 允许 SPI 中断	RW	0x0
6	UART1	UART1 中断使能 0x0: 禁止 UART1 中断 0x1: 允许 UART1 中断	RW	0x0
5	UART0	UART0 中断使能 0x0: 禁止 UART0 中断 0x1: 允许 UART0 中断	RW	0x0
4	I2C	I2C 中断使能 0x0: 禁止 I2C 中断	RW	0x0



		0x1: 允许 I2C 中断		
3	COMP	比较器中断使能 0x0: 禁止比较器中断 0x1: 允许比较器中断	RW	0x0
2	ADC	ADC 中断使能 0x0: 禁止 ADC 中断 0x1: 允许 ADC 中断	RW	0x0
1	WUT	WUT 中断使能 0x0: 禁止 WUT 中断 0x1: 允许 WUT 中断	RW	0x0
0	STMR5	STMR5 中断使能 0x0: 禁止 STMR5 中断 0x1: 允许 STMR4 中断	RW	0x0

8.4.4. IE3

Addr = 0xAB (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	—
5	LED	LED 中断使能 0x0: 禁止 LED 中断 0x1: 允许 LED 中断	RW	0x0
4	AMP	AMP 中断使能 0x0: 禁止 AMP 中断 0x1: 允许 AMP 中断	RW	0x0
3	—	—	RW	0x0
2	BUZ/FLASH	BUZ/FLASH 中断使能 0x0: 禁止 BUZ/FLASH 中断 0x1: 允许 BUZ/FLASH 中断	RW	0x0
1	WDT	WDT 中断使能 0x0: 禁止 WDT 中断 0x1: 允许 WDT 中断	RW	0x0



0	WKPND	WKPND 中断使能 0x0: 禁止 WKPND 中断 0x1: 允许 WKPND 中断	RW	0x0

8.4.5. IP0

Addr = 0xB8 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	GPI02	GPI02 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0
5: 4	GPI01	GPI01 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0
3: 2	GPI00	GPI00 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0
1: 0	LVD	LVD 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0

8.4.6. IP1

Addr = 0xB1 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	—
5: 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: 2	TMR0	TMR0 中断优先级 0x0: 优先等级为 0	RW	0x0



		0x1: 优先等级为 1		
1: 0	GPI03	GPI03 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0

8.4.7. IP2

Addr = 0xB2 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	STMRO	STMRO 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0
5: 4	TMR4	TMR4 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0
3: 2	TMR3	TMR3 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0
1: 0	TMR2	TMR2 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0

8.4.8. IP3

Addr = 0xB3 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	STM4	STM4 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0
5: 4	STM3	STM3 中断优先级 0x0: 优先等级为 0	RW	0x0



		0x1: 优先等级为 1		
3: 2	STMR2	STMR2 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0
1: 0	STMR1	STMR1 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0

8.4.9. IP4

Addr = 0xB4 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	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
5: 4	ADC	ADC 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0
3: 2	WUT	WUT 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0
1: 0	STMR5	STMR5 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0

8.4.10. IP5

Addr = 0xB5 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	SPI	SPI 中断优先级 0x0: 优先等级为 0	RW	0x0



		0x1: 优先等级为 1		
5: 4	UART1	UART1 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0
3: 2	UART0	UART0 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0
1: 0	I2C	I2C 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0

8.4.11. IP6

Addr = 0xB6 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	RW	0x0
5: 4	BUZ/FLASH	BUZ/FLASH 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0
3: 2	WDT	WDT 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0
1: 0	WKPND	WKPND 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0

8.4.12. IP7

Addr = 0xB7 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 4	—	—	—	—



3: 2	LED	LED 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1 0x2: 优先等级为 2 0x3: 优先等级为 3	RW	0x0
1: 0	AMP	AMP 中断优先级 0x0: 优先等级为 0 0x1: 优先等级为 1	RW	0x0

8.5. 中断优先级及中断嵌套

芯片规定两个中断优先级，可实现两级中断嵌套。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。通过设置 IP0-IP7 寄存器，来进行各个中断优先级的设置。IP0[1: 0] 设置 LVD 中断优先级，IP0[3: 2] 设置 GPIO0 优先级，依次类推。其中 IP1 只有 6bits，即[5: 0]有效。其他 IP0，IP2-IP7 均为 8 位。

每个中断可配置优先等级 0-1。中断优先级等级设置数越大，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。每个中断有一个中断屏蔽位，用户通过设置中断屏蔽位可以屏蔽对应的中断。在每个功能模块中，中断置位和清零是沿敏感；在中断控制器中，中断是电平敏感。

9. I/O 端口

9.1. 功能描述

- 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
- 均支持上/下拉电阻功能
- 均支持唤醒功能
- 可编程驱动能力，驱动电流范围 4mA ~ 64mA，每个档位调节 4mA。
- 支持OD输出低/高模式。
- 支持独立控制的上下拉电阻，阻值 30KΩ

9.2. 结构框图

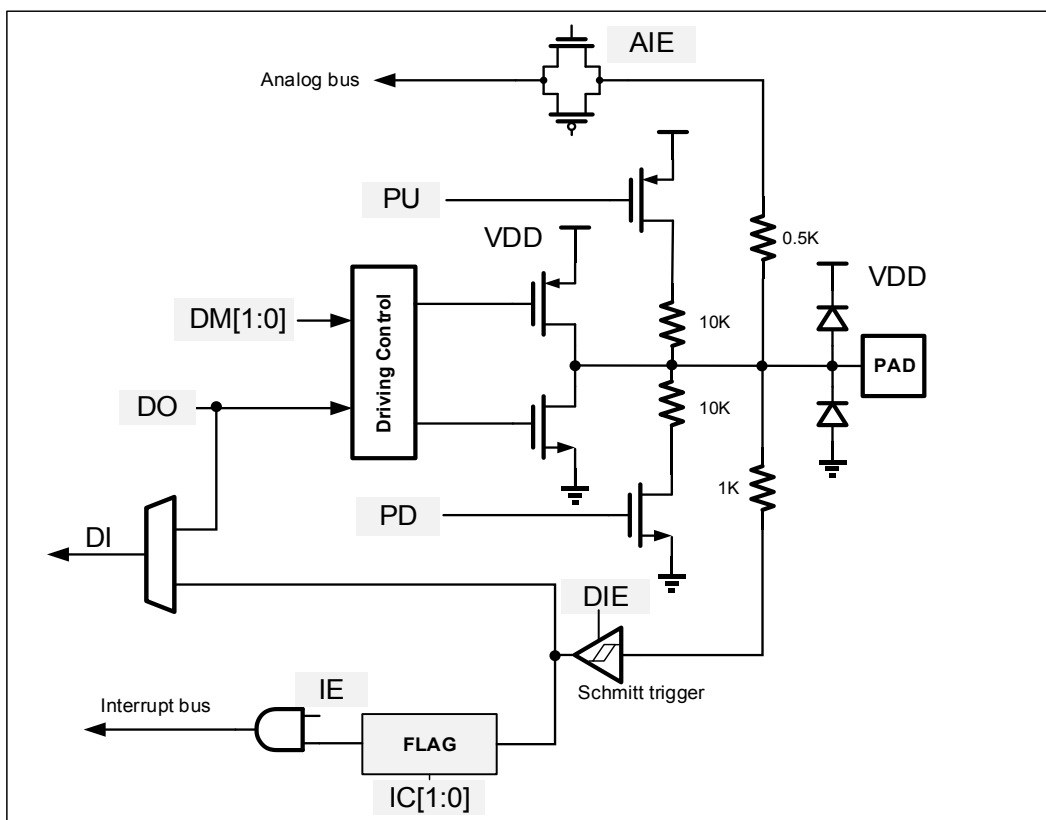


图 9-1 IO 结构图

9.3. 引脚功能复用

9.3.1. IO 引脚定义说明

- 系统有 4 组 IO，即 P0，P1，P2，P3 组；每组内 IO 有 8 个编号从 0~7。
P0 组 IO 有 P00,P01,P02,P03,P04,P05,P06,P07 这 8 个引脚；
P1 组 IO 有 P10,P11,P12,P13,P14,P15,P16,P17 这 8 个引脚；
P2 组 IO 有 P20,P21,P22,P23,P24,P25,P26,P27 这 8 个引脚；
P3 组 IO 只有 P30,P31 这 2 个引脚；
加起来一共有 26 个 GPIO 引脚；
- 引脚编号定义说明：某个具体引脚为 P[x][y]定义，x 表示 IO 组的编号，y 表示每组 IO 内的具体编号，例如如果 IO 组编号 x=1，IO 组内的具体编号 y=7，则 P[x][y]即表示引脚 P17；（以下描述 IO 功能复用的内容基于此引脚编号定义）



9.3.2. 模拟功能引脚复用表

表 9-1 引脚模拟功能复用表

引脚	比较器	运放	ADC	TOUCH	其他
P00			AIN0	TK0	
P01	C1N1 (AI0)		AIN1	TK1	
P02	C1P3 (AI0)		AIN2	TK2	
P03	C0P3 (AI0)		AIN3	TK3	
P04	CON1 (AI0)		AIN4	TK4	
P05	C0N0 (AI0)		AIN5	TK5	
P06	C0P0 (AI0)		AIN6	TK6	
P07	C0P1 (AI0)		AIN7	TK7	ADCEXREF
P10	C0P2 (AI0)		AIN8	TK8	
P11	C1P2 (AI0)		AIN9	TK9	
P12	C1P1 (AI0)		AIN10	TK10	
P13	C1P0 (AI0)		AIN11	TK11	
P14	C1N0 (AI0)		AIN12	TK12	
P15	AIN_DACMP		AIN13	TK13	
P16	CCS	OP2_N	AIN14	TK14	
P17			AIN15	TK15	
P20		OP2OUTF	AIN16	TK16	
P21		OP0_P/OP2_N	AIN17	TK17	OSCOUT
P22		OP0_N/OP2_P	AIN18	TK18	OSCIN
P23	C0P4/C1P4 (AI0)	OP0_0	AIN19	TK19	
P24		OP2_0	AIN20	TK20	
P25		OP2_P	AIN21	TK21	
P26	C0P5/C1P5 (AI0)	OP1_0	AIN22	TK22	
P27		OP1_N	AIN23	TK23	EXCAP
P30		OP1_P	AIN24	TK24	
P31			AIN25	TK25	

9.3.3. 外设数字输出功能复用图

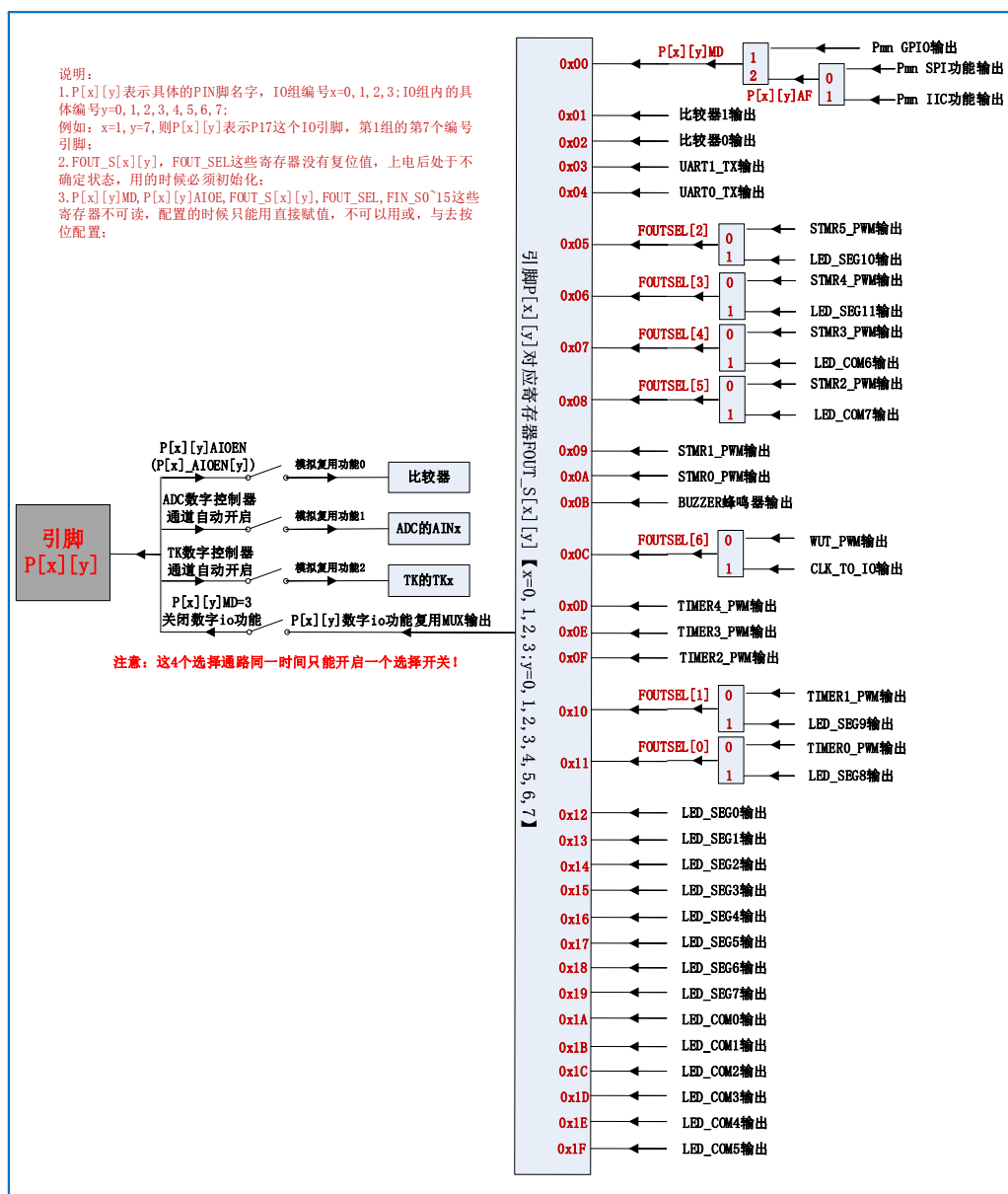


图 9-2 引脚输出功能复用数据通路图

9.3.4. 外设数字输入功能复用表

表 9-2 外设数字输入功能复用



外设输入功能	配置寄存器	引脚说明
Tmr0_cap_pin	FIN_S0	0x0: 选择固定输入低电平
Tmr1_cap_pin	FIN_S1	0x1: 选择 P00
Tmr2_cap_pin	FIN_S2	0x2: 选择 P01
Tmr3_cap_pin	FIN_S3	0x3: 选择 P02
Tmr4_cap0_pin	FIN_S4	0x4: 选择 P03
Tmr4_cap1_pin	FIN_S5	0x5: 选择 P04
Tmr4_cap2_pin	FIN_S6	0x6: 选择 P05
uart0_rx	FIN_S7	0x7: 选择 P06
uart1_rx	FIN_S8	0x8: 选择 P07
wut_cap_pin (唤醒定时器捕获输入)	FIN_S9	0x9: 选择 P10
port_wkup_in0 (唤醒通道 0 输入)	FIN_S10	0xA: 选择 P11
port_wkup_in1 (唤醒通道 1 输入)	FIN_S11	0xB: 选择 P12
port_wkup_in2 (唤醒通道 2 输入)	FIN_S12	0xC: 选择 P13
port_wkup_in3 (唤醒通道 3 输入)	FIN_S13	0xD: 选择 P14
fb_in(外部引脚触发刹车)	FIN_S14	0xE: 选择 P15
adc_etr (外部引脚触发 ADC)	FIN_S15	0xF: 选择 P16
		0x10: 选择 P17
		0x11: 选择 P20
		0x12: 选择 P21
		0x13: 选择 P22
		0x14: 选择 P23
		0x15: 选择 P24
		0x16: 选择 P25
		0x17: 选择 P26
		0x18: 选择 P27
		0x19: 选择 P30
		0x1A: 选择 P31

9.3.5. 引脚功能复用具体配置示例

(1) 示例一: UART0 功能引脚复用配置, UART0_TX 选择 P15, UART0_RX 选择 P26

//第一步: 配置 UART0_TX 功能输出选择引脚 P15, P15 方向设置为输出

P1_MD1 |= 0x1<<2; //P15MD=1, 设置 P15 为数字 IO 功能输出

FOUT_S15 = 0x4; //配置 UART0_TX 输出通路到 P15 引脚, 参见本章节“外设数字输出功能复用图”

//第二步: 配置 UART0_RX 功能输入选择引脚 P26, P26 方向设置为输入

P2_MD1 &= ~(0x3<<4); //P26MD=0, 设置 P26 为数字 IO 功能输入

FIN_S7 = 0x17; //配置 UART0_RX 功能复用寄存器 FIN_S7=0x17, 选择 P26 引脚, 参见本章节“外



设数字输入功能复用表”

通过上述配置 UART0 的功能 IO 复用，修改不同的配置可以将其任意选择映射到所有 IO 引脚上。

- (2) 示例二：IIC 功能引脚复用配置，I2C_SCL 选择 P13, I2C_SDA 选择 P14

//第一步：配置 P13,P14 模式寄存器值为 2，选择数字功能复用功能模式

P1_MD0 |= 0x2<<6; //P13 模式寄存器选择数字功能复用模式，参见本章节的 P1_MD0 寄存器说明

P1_MD1 |= 0x2<<0; //P14 模式寄存器选择数字功能复用模式，参见本章节的 P1_MD1 寄存器说明

//第二步：配置 P13,P14 复用功能选择寄存器选择对应的 IIC 的功能

P1_AF0 = 0x10; //P13 选择 I2C_SCL 功能，P14 选择 I2C_SDA 功能；注意 P1_AF0 是只写寄存器

//第三步：配置 P13,P14 的数字输出通道选择 0 通道，注意 0 通道是 GPIO, IIC 和 SPI 功能输出通道

FOUT_S13 = 0x0; //配置 I2C_SCL 通路输出到 P13 引脚，参见本章节“外设数字输出功能复用图”

FOUT_S14 = 0x0; //配置 I2C_SDA 通路输出到 P14 引脚，参见本章节“外设数字输出功能复用图”

- (3) 示例三：比较器 0 模拟功能引脚 IO 复用配置，COP0 选择 P06, CON0 选择 P05

//第一步：关闭比较器 0 复用功能引脚的数字 IO 功能

P0_MD1 |= (0x3<<4) | (0x3<<2); //关闭 P05,P06 的数字 IO 功能，参见本章节“外设数字输出功能复用图”

//第二步：使能 P05,P06 的模拟比较器模拟 IO 通道

P0_AI0EN = 0x60; //使能 P05,P06 模拟比较器复用功能，参见本章节“外设数字输出功能复用图”

9.4. 寄存器列表

表 9-3 gpio register list

address	Register Name	Description
0x80 (SFR)	P0	P0 数据寄存器
0xC0 (XSFR)	P0_PU	P0 上拉电阻使能寄存器
0xC1 (XSFR)	P0_PD	P0 下拉电阻使能寄存器
0xC2 (XSFR)	P0_MD0	P0 工作模式寄存器 0
0xC3 (XSFR)	P0_MD1	P0 工作模式寄存器 1
0xC4 (XSFR)	P0_TRG0	P0 中断触发配置寄存器 0
0xC5 (XSFR)	P0_TRG1	P0 中断触发配置寄存器 1
0xC6 (XSFR)	P0_PND	P0 中断 PENDING 寄存器



0xC7 (XSFR)	P0_IMK	P0 中断屏蔽寄存器
0xC8 (XSFR)	P0_AIOEN	P0 比较器模拟 IO 功能使能
0xC9 (XSFR)	P0_DRV0	P0 驱动电流配置寄存器
0xCA (XSFR)	P0_ODN	P0 开漏低使能寄存器
0xCB (XSFR)	P0_ODP	P0 开漏高使能寄存器
0xA8 (XSFR)	P0_DRV1	P0 驱动电流配置寄存器 1
0xA9 (XSFR)	P0_DRV2	P0 驱动电流配置寄存器 2
0xAA (XSFR)	P0_DRV3	P0 驱动电流配置寄存器 3
0xAB (XSFR)	P0_DRV4	P0 驱动电流配置寄存器 4
0xAC (XSFR)	P0_DRV5	P0 驱动电流配置寄存器 5
0xAD (XSFR)	P0_DRV6	P0 驱动电流配置寄存器 6
0xAE (XSFR)	P0_DRV7	P0 驱动电流配置寄存器 7
0x166 (XSFR)	P0_AF0	P0 数字外设功能复用选择配置寄存器 0
0x167 (XSFR)	P0_AF1	P0 数字外设功能复用选择配置寄存器 1
0x90 (SFR)	P1	P1 数据寄存器
0xD0 (XSFR)	P1_PU	P1 上拉电阻使能寄存器
0xD1 (XSFR)	P1_PD	P1 下拉电阻使能寄存器
0xD2 (XSFR)	P1_MD0	P1 工作模式寄存器 0
0xD3 (XSFR)	P1_MD1	P1 工作模式寄存器 1
0xD4 (XSFR)	P1_TRG0	P1 中断触发配置寄存器 0
0xD5 (XSFR)	P1_TRG1	P1 中断触发配置寄存器 1
0xD6 (XSFR)	P1_PND	P1 中断 PENDING 寄存器
0xD7 (XSFR)	P1_IMK	P1 中断屏蔽寄存器



0xD8 (XSFR)	P1_AIOEN	P1 比较器模拟 IO 功能使能寄存器
0xD9 (XSFR)	P1_DRV0	P1 驱动电流配置寄存器
0xDA (XSFR)	P1_ODN	P1 开漏低使能寄存器
0xDB (XSFR)	P1_ODP	P1 开漏高使能寄存器
0xAF (XSFR)	P1_DRV1	P1 驱动电流配置寄存器 1
0xB0 (XSFR)	P1_DRV2	P1 驱动电流配置寄存器 2
0xB1 (XSFR)	P1_DRV3	P1 驱动电流配置寄存器 3
0xB2 (XSFR)	P1_DRV4	P1 驱动电流配置寄存器 4
0xB3 (XSFR)	P1_DRV5	P1 驱动电流配置寄存器 5
0xB4 (XSFR)	P1_DRV6	P1 驱动电流配置寄存器 6
0xB5 (XSFR)	P1_DRV7	P1 驱动电流配置寄存器 7
0x168 (XSFR)	P1_AF0	P1 数字外设功能复用选择配置寄存器 0
0x169 (XSFR)	P1_AF1	P1 数字外设功能复用选择配置寄存器 1
0xA0 (SFR)	P2	P2 数据寄存器
0xE0 (XSFR)	P2_PU	P2 上拉电阻使能寄存器
0xE1 (XSFR)	P2_PD	P2 下拉电阻使能寄存器
0xE2 (XSFR)	P2_MD0	P2 工作模式寄存器 0
0xE3 (XSFR)	P2_MD1	P2 工作模式寄存器 1
0xE4 (XSFR)	P2_TRG0	P2 中断触发配置寄存器 0
0xE5 (XSFR)	P2_TRG1	P2 中断触发配置寄存器 1
0xE6 (XSFR)	P2_PND	P2 中断 PENDING 寄存器
0xE7 (XSFR)	P2_IMK	P2 中断屏蔽寄存器
0xE8 (XSFR)	P2_AIOEN	P2 比较器模拟 IO 功能使能寄存器



0xE9 (XSFR)	P2_DRV0	P2 驱动电流配置寄存器
0xEA (XSFR)	P2_ODN	P2 开漏低使能寄存器
0xEB (XSFR)	P2_ODP	P2 开漏高使能寄存器
0xB6 (XSFR)	P2_DRV1	P2 驱动电流配置寄存器 1
0xB7 (XSFR)	P2_DRV2	P2 驱动电流配置寄存器 2
0xB8 (XSFR)	P2_DRV3	P2 驱动电流配置寄存器 3
0xB9 (XSFR)	P2_DRV4	P2 驱动电流配置寄存器 4
0xBA (XSFR)	P2_DRV5	P2 驱动电流配置寄存器 5
0xBB (XSFR)	P2_DRV6	P2 驱动电流配置寄存器 6
0xBC (XSFR)	P2_DRV7	P2 驱动电流配置寄存器 7
0x16A (XSFR)	P2_AF0	P2 数字外设功能复用选择配置寄存器 0
0x16B (XSFR)	P2_AF1	P2 数字外设功能复用选择配置寄存器 1
0xB0 (SFR)	P3	P3 数据寄存器
0xF0 (XSFR)	P3_PU	P3 上拉电阻使能寄存器
0xF1 (XSFR)	P3_PD	P3 下拉电阻使能寄存器
0xF2 (XSFR)	P3_MD0	P3 工作模式寄存器 0
0xF3 (XSFR)	P3_MD1	P3 工作模式寄存器 1
0xF4 (XSFR)	P3_TRG0	P3 中断触发配置寄存器 0
0xF5 (XSFR)	P3_TRG1	P3 中断触发配置寄存器 1
0xF6 (XSFR)	P3_PND	P3 中断 PENDING 寄存器
0xF7 (XSFR)	P3_IMK	P3 中断屏蔽寄存器
0xF8 (XSFR)	P3_AIOEN	P3 比较器模拟 I/O 功能使能寄存器
0xF9 (XSFR)	P3_DRV0	P3 驱动电流配置寄存器



0xFA (XSFR)	P3_ODN	P3 开漏低使能寄存器
0xFB (XSFR)	P3_ODP	P3 开漏高使能寄存器
0xBD (XSFR)	P3_DRV1	P3 驱动电流配置寄存器 1
0x16C (XSFR)	P3_AF0	P3 数字外设功能复用选择配置寄存器 0
0x16D (XSFR)	P3_AF1	P3 数字外设功能复用选择配置寄存器 1
0x17E (XSFR)	FOUT_S00	P00 数字功能输出选择寄存器
0x17F (XSFR)	FOUT_S01	P01 数字功能输出选择寄存器
0x180 (XSFR)	FOUT_S02	P02 数字功能输出选择寄存器
0x181 (XSFR)	FOUT_S03	P03 数字功能输出选择寄存器
0x182 (XSFR)	FOUT_S04	P04 数字功能输出选择寄存器
0x183 (XSFR)	FOUT_S05	P05 数字功能输出选择寄存器
0x184 (XSFR)	FOUT_S06	P06 数字功能输出选择寄存器
0x185 (XSFR)	FOUT_S07	P07 数字功能输出选择寄存器
0x186 (XSFR)	FOUT_S10	P10 数字功能输出选择寄存器
0x187 (XSFR)	FOUT_S11	P11 数字功能输出选择寄存器
0x188 (XSFR)	FOUT_S12	P12 数字功能输出选择寄存器
0x189 (XSFR)	FOUT_S13	P13 数字功能输出选择寄存器
0x18A (XSFR)	FOUT_S14	P14 数字功能输出选择寄存器
0x18B (XSFR)	FOUT_S15	P15 数字功能输出选择寄存器
0x18C (XSFR)	FOUT_S16	P16 数字功能输出选择寄存器
0x18D (XSFR)	FOUT_S17	P17 数字功能输出选择寄存器
0x18E (XSFR)	FOUT_S20	P20 数字功能输出选择寄存器
0x18F (XSFR)	FOUT_S21	P21 数字功能输出选择寄存器



0x190 (XSFR)	FOUT_S22	P22 数字功能输出选择寄存器
0x191 (XSFR)	FOUT_S23	P23 数字功能输出选择寄存器
0x192 (XSFR)	FOUT_S24	P24 数字功能输出选择寄存器
0x193 (XSFR)	FOUT_S25	P25 数字功能输出选择寄存器
0x194 (XSFR)	FOUT_S26	P26 数字功能输出选择寄存器
0x195 (XSFR)	FOUT_S27	P27 数字功能输出选择寄存器
0x196 (XSFR)	FOUT_S30	P30 数字功能输出选择寄存器
0x197 (XSFR)	FOUT_S31	P31 数字功能输出选择寄存器
0x198 (XSFR)	FOUT_SEL	数字功能输出选择寄存器
0x16E (XSFR)	FIN_S0	数字功能输入 I0 映射寄存器 0
0x16F (XSFR)	FIN_S1	数字功能输入 I0 映射寄存器 1
0x170 (XSFR)	FIN_S2	数字功能输入 I0 映射寄存器 2
0x171 (XSFR)	FIN_S3	数字功能输入 I0 映射寄存器 3
0x172 (XSFR)	FIN_S4	数字功能输入 I0 映射寄存器 4
0x173 (XSFR)	FIN_S5	数字功能输入 I0 映射寄存器 5
0x174 (XSFR)	FIN_S6	数字功能输入 I0 映射寄存器 6
0x175 (XSFR)	FIN_S7	数字功能输入 I0 映射寄存器 7
0x176 (XSFR)	FIN_S8	数字功能输入 I0 映射寄存器 8
0x177 (XSFR)	FIN_S9	数字功能输入 I0 映射寄存器 9
0x178 (XSFR)	FIN_S10	数字功能输入 I0 映射寄存器 10
0x179 (XSFR)	FIN_S11	数字功能输入 I0 映射寄存器 11
0x17A (XSFR)	FIN_S12	数字功能输入 I0 映射寄存器 12
0x17B (XSFR)	FIN_S13	数字功能输入 I0 映射寄存器 13



0x17C (XSFR)	FIN_S14	数字功能输入 IO 映射寄存器 14
0x17D (XSFR)	FIN_S15	数字功能输入 IO 映射寄存器 15

9.5. 寄存器详细说明

9.5.1. P0

Addr = 0x80 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	P0	PORT0 数据寄存器	RW	0x00

9.5.2. P0_PU

Addr = 0xC0 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P07PU	P07 30K Ω 上拉电阻使能位 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
6	P06PU	P06 30K Ω 上拉电阻使能位 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
5	P05PU	P05 30K Ω 上拉电阻使能位 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
4	P04PU	P04 30K Ω 上拉电阻使能位 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
3	P03PU	P03 30K Ω 上拉电阻使能位 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0



2	P02PU	P02 30KΩ 上拉电阻使能位 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
1	P01PU	P01 30KΩ 上拉电阻使能位 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
0	P00PU	P00 30KΩ 上拉电阻使能位 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0

9.5.3. P0_PD

Addr = 0xC1 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P07PD	P07 30KΩ 下拉电阻使能位 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
6	P06PD	P06 30KΩ 下拉电阻使能位 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
5	P05PD	P05 30KΩ 下拉电阻使能位 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
4	P04PD	P04 30KΩ 下拉电阻使能位 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
3	P03PD	P03 30KΩ 下拉电阻使能位 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
2	P02PD	P02 30KΩ 下拉电阻使能位 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0



1	P01PD	P01 30KΩ 下拉电阻使能位 0x0: 关闭 30KΩ 下拉电阻 0x1: 使能 30KΩ 下拉电阻	RW	0x0
0	P00PD	P00 30KΩ 下拉电阻使能位 0x0: 关闭 30KΩ 下拉电阻 0x1: 使能 30KΩ 下拉电阻	RW	0x0

9.5.4. P0_MD0

Addr = 0xC2 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	P03MD	P03 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式 (数字功能关闭模式)	RW	0x0
5: 4	P02MD	P02 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式 (数字功能关闭模式)	RW	0x0
3: 2	P01MD	P01 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式 (数字功能关闭模式)	RW	0x0
1: 0	P00MD	P00 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式	RW	0x0



		0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）		

9.5.5. P0_MD1

Addr = 0xC3 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	P07MD	P07 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）	RW	0x0
5: 4	P06MD	P06 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）	RW	0x0
3: 2	P05MD	P05 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）	RW	0x0
1: 0	P04MD	P04 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）	RW	0x0



9.5.6. P0_AF0

Addr = 0x166 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P07AF	P07 数字外设功能复用选择配置寄存器. 0x0: SPI MOSI(I00) 功能选择 0x1: I2C_SCL 功能选择	WO	0x0
6	P06AF	P06 数字外设功能复用选择配置寄存器. 0x0: SPICLK 功能选择 0x1: I2C_SDA 功能选择	WO	0x0
5	P05AF	P05 数字外设功能复用选择配置寄存器. 0x0: SPI MISO(I01) 功能选择 0x1: I2C_SCL 功能选择	WO	0x0
4	P04AF	P04 数字外设功能复用选择配置寄存器. 0x0: SPI MOSI(I00) 功能选择 0x1: I2C_SDA 功能选择	WO	0x0
3	P03AF	P03 数字外设功能复用选择配置寄存器. 0x0: SPICLK 功能选择 0x1: I2C_SCL 功能选择	WO	0x0
2	P02AF	P02 数字外设功能复用选择配置寄存器. 0x0: SPI MISO(I01) 功能选择 0x1: I2C_SDA 功能选择	WO	0x0
1	P01AF	P01 数字外设功能复用选择配置寄存器. 0x0: SPI MOSI(I00) 功能选择 0x1: I2C_SCL 功能选择	WO	0x0
0	P00AF	P00 数字外设功能复用选择配置寄存器. 0x0: SPICLK 功能选择 0x1: I2C_SDA 功能选择	WO	0x0



9.5.7. P0_TRG0

Addr = 0xC4 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	P03TRG	P03 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
5: 4	P02TRG	P02 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
3: 2	P01TRG	P01 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
1: 0	P00TRG	P00 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0

9.5.8. P0_TRG1

Addr = 0xC5 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	P07TRG	P07 中断触发配置寄存器. 0x0: 双边沿触发中断	RW	0x0



		0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断		
5: 4	P06TRG	P06 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
3: 2	P05TRG	P05 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
1: 0	P04TRG	P04 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0

9.5.9. P0_PND

Addr = 0xC6 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P07PND	P07 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
6	P06PND	P06 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
5	P05PND	P05 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0



4	P04PND	P04 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
3	P03PND	P03 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
2	P02PND	P02 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
1	P01PND	P01 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
0	P00PND	P00 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0

Note: CPU 写 P0_PND 操作，则清掉 P00~P07 所有的中断标志位。所以中断使用时，中断处理函数优先将 P0_PND 通过变量进行保存，在处理完函数后，才统一进行标志位清除。如果对中断标志位 P0_PND 使用比较严格应用，建议使用不同组的 GPIO 中断。

9.5.10. P0_IMK

Addr = 0xC7 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P07IMK	P07 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能 0x1: 打开 I0 中断触发功能	RW	0x0
6	P06IMK	P06 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能 0x1: 打开 I0 中断触发功能	RW	0x0
5	P05IMK	P05 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能	RW	0x0



		0x1: 打开 I0 中断触发功能		
4	P04IMK	P04 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能 0x1: 打开 I0 中断触发功能	RW	0x0
3	P03IMK	P03 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能 0x1: 打开 I0 中断触发功能	RW	0x0
2	P02IMK	P02 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能 0x1: 打开 I0 中断触发功能	RW	0x0
1	P01IMK	P01 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能 0x1: 打开 I0 中断触发功能	RW	0x0
0	P00IMK	P00 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能 0x1: 打开 I0 中断触发功能	RW	0x0

9.5.11. P0_AIOEN

Addr = 0xC8 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P07AIOEN	P07 模拟 I0 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
6	P06AIOEN	P06 模拟 I0 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
5	P05AIOEN	P05 模拟 I0 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
4	P04AIOEN	P04 模拟 I0 功能使能位. 0x0: 不使能	WO	0x0



		0x1: 使能		
3	P03AIOEN	P03 模拟 IO 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
2	P02AIOEN	P02 模拟 IO 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
1	P01AIOEN	P01 模拟 IO 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
0	P00AIOEN	P00 模拟 IO 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0

9.5.12. P0_AIOEN1

Addr = 0xCC (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P07AIOEN1	P07 模拟 IO 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
6	P06AIOEN1	P06 模拟 IO 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
5	P05AIOEN1	P05 模拟 IO 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
4	P04AIOEN1	P04 模拟 IO 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
3	P03AIOEN1	P03 模拟 IO 功能 1 使能位. 0x0: 不使能	WO	0x0



		0x1: 使能		
2	P02AIOEN1	P02 模拟 IO 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
1	P01AIOEN1	P01 模拟 IO 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
0	P00AIOEN1	P00 模拟 IO 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0

9.5.13. P0_AIOEN2

Addr = 0xCD (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P07AIOEN2	P07 模拟 IO 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
6	P06AIOEN2	P06 模拟 IO 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
5	P05AIOEN2	P05 模拟 IO 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
4	P04AIOEN2	P04 模拟 IO 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
3	P03AIOEN2	P03 模拟 IO 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
2	P02AIOEN2	P02 模拟 IO 功能 2 使能位. 0x0: 不使能	WO	0x0



		0x1: 使能		
1	P01AIOEN2	P01 模拟 IO 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
0	P00AIOEN2	P00 模拟 IO 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0

9.5.14. P0_DRV0

Addr = 0xC9 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P00DRVE	P00 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P00DRV	P00 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0

9.5.15. P0_DRV1

Addr = 0xA8 (XSFR)

Bit(s)	Name	Description	R/W	Reset



7: 5	—	—	—	—
4	P01DRVE	P01 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P01DRV	P01 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0

9.5.16. P0_DRV2

Addr = 0xA9 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P02DRVE	P02 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P02DRV	P02 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0



9.5.17. P0_DRV3

Addr = 0xAA (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P03DRVE	P03 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P03DRV	P03 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0

9.5.18. P0_DRV4

Addr = 0xAB (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P04DRVE	P04 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P04DRV	P04 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA	WO	0x0



		0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!		

9.5.19. P0_DRV5

Addr = 0xAC (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P05DRVE	P05 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P05DRV	P05 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0

9.5.20. P0_DRV6

Addr = 0xAD (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P06DRVE	P06 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力	WO	0x0



		0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!		
3: 0	P06DRV	P06 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0

9.5.21. P0_DRV7

Addr = 0xAE (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P07DRVE	P07 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P07DRV	P07 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0



9.5.22. P0_ODN

Addr = 0xCA (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P070DN	P07 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
6	P060DN	P06 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
5	P050DN	P05 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
4	P040DN	P04 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
3	P030DN	P03 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
2	P020DN	P02 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
1	P010DN	P01 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0



0	P000DN	P00 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0

9.5.23. P0_ODP

Addr = 0xCB (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P07ODP	P07 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
6	P06ODP	P06 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
5	P05ODP	P05 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
4	P04ODP	P04 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
3	P03ODP	P03 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
2	P02ODP	P02 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1;	WO	0x0



		DIN_VDD=0 时输出高阻态)		
1	P010DP	P01 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
0	P000DP	P00 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0

9.5.24. P1

Addr = 0x90 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	P1	PORT1 数据寄存器	RW	0x00

9.5.25. P1_PU

Addr = 0xD0 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P17PU	P17 30K Ω 上拉电阻使能位. 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
6	P16PU	P16 30K Ω 上拉电阻使能位. 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
5	P15PU	P15 30K Ω 上拉电阻使能位. 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
4	P14PU	P14 30K Ω 上拉电阻使能位.	RW	0x0



		0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻		
3	P13PU	P13 30KΩ 上拉电阻使能位. 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
2	P12PU	P12 30KΩ 上拉电阻使能位. 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
1	P11PU	P11 30KΩ 上拉电阻使能位. 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
0	P10PU	P10 30KΩ 上拉电阻使能位. 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0

9.5.26. P1_PD

Addr = 0xD1 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P17PD	P17 30KΩ 下拉电阻使能位. 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
6	P16PD	P16 30KΩ 下拉电阻使能位. 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
5	P15PD	P15 30KΩ 下拉电阻使能位. 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
4	P14PD	P14 30KΩ 下拉电阻使能位. 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
3	P13PD	P13 30KΩ 下拉电阻使能位.	RW	0x0



		0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻		
2	P12PD	P12 30KΩ 下拉电阻使能位. 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
1	P11PD	P11 30KΩ 下拉电阻使能位. 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
0	P10PD	P10 30KΩ 下拉电阻使能位. 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0

9.5.27. P1_MD0

Addr = 0xD2 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	P13MD	P13 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式 (数字功能关闭模式)	RW	0x0
5: 4	P12MD	P12 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式 (数字功能关闭模式)	RW	0x0
3: 2	P11MD	P11 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式	RW	0x0



		0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）		
1: 0	P10MD	P10 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）	RW	0x0

9.5.28. P1_MD1

Addr = 0xD3 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	P17MD	P17 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）	RW	0x0
5: 4	P16MD	P16 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）	RW	0x0
3: 2	P15MD	P15 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）	RW	0x0
1: 0	P14MD	P14 工作模式寄存器.	RW	0x0



		0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）		

9.5.29. P1_AF0

Addr = 0x168 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P17AF	P17 数字外设功能复用选择配置寄存器. 0x0: SPICLK 功能选择 0x1: I2C_SDA 功能选择	WO	0x0
6	P16AF	P16 数字外设功能复用选择配置寄存器. 0x0: SPI MISO(I01) 功能选择 0x1: I2C_SDA 功能选择	WO	0x0
5	P15AF	P15 数字外设功能复用选择配置寄存器. 0x0: SPI MOSI(I00) 功能选择 0x1: I2C_SCL 功能选择	WO	0x0
4	P14AF	P14 数字外设功能复用选择配置寄存器. 0x0: SPICLK 功能选择 0x1: I2C_SDA 功能选择	WO	0x0
3	P13AF	P13 数字外设功能复用选择配置寄存器. 0x0: I2C_SCL 功能选择 0x1: I2C_SDA 功能选择	WO	0x0
2	P12AF	P12 数字外设功能复用选择配置寄存器. 0x0: SPI MOSI(I00) 功能选择 0x1: I2C_SCL 功能选择	WO	0x0
1	P11AF	P11 数字外设功能复用选择配置寄存器. 0x0: SPICLK 功能选择 0x1: I2C_SDA 功能选择	WO	0x0
0	P10AF	P10 数字外设功能复用选择配置寄存器.	WO	0x0



		0x0: SPI MISO(I01) 功能选择		
		0x1: I2C_SDA 功能选择		

9.5.30. P1_TRG0

Addr = 0xD4 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	P13TRG	P13 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
5: 4	P12TRG	P12 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
3: 2	P11TRG	P11 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
1: 0	P10TRG	P10 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0

9.5.31. P1_TRG1

Addr = 0xD5 (XSFR)



Bit(s)	Name	Description	R/W	Reset
7: 6	P17TRG	P17 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
5: 4	P16TRG	P16 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
3: 2	P15TRG	P15 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
1: 0	P14TRG	P14 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0

9.5.32. P1_PND

Addr = 0xD6 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P17PND	P17 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
6	P16PND	P16 中断 PENDING 寄存器. 0x0: 没有中断 PENDING	RW	0x0



		0x1: 有中断 PENDING		
5	P15PND	P15 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
4	P14PND	P14 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
3	P13PND	P13 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
2	P12PND	P12 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
1	P11PND	P11 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
0	P10PND	P10 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0

Note: CPU 写 P1_PND 操作，则清掉 P10~P17 所有的中断标志位。所以中断使用时，中断处理函数优先将 P1_PND 通过变量进行保存，在处理完函数后，才统一进行标志位清除。如果对中断标志位 P1_PND 使用比较严格应用，建议使用不同组的 GPIO 中断。

9.5.33. P1_IMK

Addr = 0xD7 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P17IMK	P17 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能 0x1: 打开 I0 中断触发功能	RW	0x0
6	P16IMK	P16 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能	RW	0x0



		0x1: 打开 I0 中断触发功能		
5	P15IMK	P15 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能 0x1: 打开 I0 中断触发功能	RW	0x0
4	P14IMK	P14 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能 0x1: 打开 I0 中断触发功能	RW	0x0
3	P13IMK	P13 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能 0x1: 打开 I0 中断触发功能	RW	0x0
2	P12IMK	P12 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能 0x1: 打开 I0 中断触发功能	RW	0x0
1	P11IMK	P11 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能 0x1: 打开 I0 中断触发功能	RW	0x0
0	P10IMK	P10 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能 0x1: 打开 I0 中断触发功能	RW	0x0

9.5.34. P1_AIOEN

Addr = 0xD8 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P17AIOEN	P17 模拟 I0 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
6	P16AIOEN	P16 模拟 I0 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
5	P15AIOEN	P15 模拟 I0 功能使能位. 0x0: 不使能	WO	0x0



		0x1: 使能		
4	P14AIOEN	P14 模拟 I/O 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
3	P13AIOEN	P13 模拟 I/O 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
2	P12AIOEN	P12 模拟 I/O 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
1	P11AIOEN	P11 模拟 I/O 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
0	P10AIOEN	P10 模拟 I/O 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0

9.5.35. P1_AIOEN1

Addr = 0xDC (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P17AIOEN1	P17 模拟 I/O 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
6	P16AIOEN1	P16 模拟 I/O 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
5	P15AIOEN1	P15 模拟 I/O 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
4	P14AIOEN1	P14 模拟 I/O 功能 1 使能位. 0x0: 不使能	WO	0x0



		0x1: 使能		
3	P13AIOEN1	P13 模拟 I0 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
2	P12AIOEN1	P12 模拟 I0 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
1	P11AIOEN1	P11 模拟 I0 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
0	P10AIOEN1	P10 模拟 I0 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0

9.5.36. P1_AIOEN2

Addr = 0xDD (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P17AIOEN2	P17 模拟 I0 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
6	P16AIOEN2	P16 模拟 I0 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
5	P15AIOEN2	P15 模拟 I0 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
4	P14AIOEN2	P14 模拟 I0 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
3	P13AIOEN2	P13 模拟 I0 功能 2 使能位. 0x0: 不使能	WO	0x0



		0x1: 使能		
2	P12AIOEN2	P12 模拟 IO 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
1	P11AIOEN2	P11 模拟 IO 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
0	P10AIOEN2	P10 模拟 IO 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0

9.5.37. P1_DRV0

Addr = 0xD9 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P10DRVE	P10 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P10DRV	P10 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0



9.5.38. P1_DRV1

Addr = 0xAF (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	—
4	P11DRVE	P11 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P11DRV	P11 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0

9.5.39. P1_DRV2

Addr = 0xB0 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P12DRVE	P12 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P12DRV	P12 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA	WO	0x0



		0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!		

9.5.40. P1_DRV3

Addr = 0xB1 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	—
4	P13DRVE	P13 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P13DRV	P13 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0

9.5.41. P1_DRV4

Addr = 0xB2 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P14DRVE	P14 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力	WO	0x0



		0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!		
3: 0	P14DRV	P14 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0

9.5.42. P1_DRV5

Addr = 0xB3 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	—
4	P15DRVE	P15 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P15DRV	P15 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0



9.5.43. P1_DRV6

Addr = 0xB4 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P16DRVE	P16 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P16DRV	P16 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0

9.5.44. P1_DRV7

Addr = 0xB5 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P17DRVE	P17 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P17DRV	P17 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA	WO	0x0



		0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!		

9.5.45. P1_ODN

Addr = 0xDA (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P17ODN	P17 开漏功能使能位. 0x0: 不使能 0x1: 使能(DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
6	P16ODN	P16 开漏功能使能位. 0x0: 不使能 0x1: 使能(DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
5	P15ODN	P15 开漏功能使能位. 0x0: 不使能 0x1: 使能(DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
4	P14ODN	P14 开漏功能使能位. 0x0: 不使能 0x1: 使能(DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
3	P13ODN	P13 开漏功能使能位. 0x0: 不使能 0x1: 使能(DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
2	P12ODN	P12 开漏功能使能位. 0x0: 不使能 0x1: 使能(DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0



		输出高阻态)		
1	P110DN	P11 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
0	P100DN	P10 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0

9.5.46. P1_ODP

Addr = 0xDB (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P170DP	P17 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
6	P160DP	P16 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
5	P150DP	P15 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
4	P140DP	P14 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
3	P130DP	P13 开漏功能使能位.	WO	0x0



		0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)		
2	P120DP	P12 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
1	P110DP	P11 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
0	P100DP	P10 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0

9.5.47. P2

Addr = 0xA0 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	P2	PORT2 数据寄存器	RW	0x00

9.5.48. P2_PU

Addr = 0xE0 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P27PU	P27 30K Ω 上拉电阻使能位. 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
6	P26PU	P26 30K Ω 上拉电阻使能位.	RW	0x0



		0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻		
5	P25PU	P25 30K Ω 上拉电阻使能位. 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
4	P24PU	P24 30K Ω 上拉电阻使能位. 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
3	P23PU	P23 30K Ω 上拉电阻使能位. 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
2	P22PU	P22 30K Ω 上拉电阻使能位. 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
1	P21PU	P21 30K Ω 上拉电阻使能位. 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0
0	P20PU	P20 30K Ω 上拉电阻使能位. 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0

9.5.49. P2_PD

Addr = 0xE1 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P27PD	P27 30K Ω 下拉电阻使能位. 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
6	P26PD	P26 30K Ω 下拉电阻使能位. 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
5	P25PD	P25 30K Ω 下拉电阻使能位.	RW	0x0



		0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻		
4	P24PD	P24 30KΩ 下拉电阻使能位. 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
3	P23PD	P23 30KΩ 下拉电阻使能位. 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
2	P22PD	P22 30KΩ 下拉电阻使能位. 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
1	P21PD	P21 30KΩ 下拉电阻使能位. 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
0	P20PD	P20 30KΩ 下拉电阻使能位. 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0

9.5.50. P2_MD0

Addr = 0xE2 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	P23MD	P23 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式 (数字功能关闭模式)	RW	0x0
5: 4	P22MD	P22 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式	RW	0x0



		0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）		
3: 2	P21MD	P21 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）	RW	0x0
1: 0	P20MD	P20 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）	RW	0x0

9.5.51. P2_MD1

Addr = 0xE3 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	P27MD	P27 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）	RW	0x0
5: 4	P26MD	P26 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）	RW	0x0
3: 2	P25MD	P25 工作模式寄存器.	RW	0x0



		0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）		
1: 0	P24MD	P24 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式（数字功能关闭模式）	RW	0x0

9.5.52. P2_AF0

Addr = 0x16A (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P27AF	P27 数字外设功能复用选择配置寄存器. 0x0: SPI MISO(I01) 功能选择 0x1: I2C_SCL 功能选择	WO	0x0
6	P26AF	P26 数字外设功能复用选择配置寄存器. 0x0: SPI MOSI(I00) 功能选择 0x1: I2C_SDA 功能选择	WO	0x0
5	P25AF	P25 数字外设功能复用选择配置寄存器. 0x0: SPICLK 功能选择 0x1: I2C_SCL 功能选择	WO	0x0
4	P24AF	P24 数字外设功能复用选择配置寄存器. 0x0: SPI MISO(I01) 功能选择 0x1: I2C_SDA 功能选择	WO	0x0
3	P23AF	P23 数字外设功能复用选择配置寄存器. 0x0: SPI MOSI(I00) 功能选择 0x1: I2C_SCL 功能选择	WO	0x0
2	P22AF	P22 数字外设功能复用选择配置寄存器.	WO	0x0



		0x0: SPICLK 功能选择 0x1: I2C_SDA 功能选择		
1	P21AF	P21 数字外设功能复用选择配置寄存器. 0x0: SPI MISO(I01) 功能选择 0x1: I2C_SCL 功能选择	WO	0x0
0	P20AF	P20 数字外设功能复用选择配置寄存器. 0x0: SPI MOSI(I00) 功能选择 0x1: I2C_SCL 功能选择	WO	0x0

9.5.53. P2_TRG0

Addr = 0xE4 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	P23TRG	P23 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
5: 4	P22TRG	P22 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
3: 2	P21TRG	P21 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
1: 0	P20TRG	P20 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0



9.5.54. P2_TRG1

Addr = 0xE5 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	P27TRG	P27 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
5: 4	P26TRG	P26 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
3: 2	P25TRG	P25 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0
1: 0	P24TRG	P24 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0

9.5.55. P2_PND

Addr = 0xE6 (XSFR)

Bit(s)	Name	Description	R/W	Reset



7	P27PND	P27 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
6	P26PND	P26 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
5	P25PND	P25 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
4	P24PND	P24 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
3	P23PND	P23 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
2	P22PND	P22 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
1	P21PND	P21 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
0	P20PND	P20 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0

Note: CPU 写 P2_PND 操作，则清掉 P20~P27 所有的中断标志位。所以中断使用时，中断处理函数优先将 P2_PND 通过变量进行保存，在处理完函数后，才统一进行标志位清除。如果对中断标志位 P2_PND 使用比较严格应用，建议使用不同组的 GPIO 中断。

9.5.56. P2_IMK

Addr = 0xE7 (XSFR)



Bit(s)	Name	Description	R/W	Reset
7	P27IMK	P27 中断屏蔽寄存器. 0x0: 屏蔽 IO 中断触发功能 0x1: 打开 IO 中断触发功能	RW	0x0
6	P26IMK	P26 中断屏蔽寄存器. 0x0: 屏蔽 IO 中断触发功能 0x1: 打开 IO 中断触发功能	RW	0x0
5	P25IMK	P25 中断屏蔽寄存器. 0x0: 屏蔽 IO 中断触发功能 0x1: 打开 IO 中断触发功能	RW	0x0
4	P24IMK	P24 中断屏蔽寄存器. 0x0: 屏蔽 IO 中断触发功能 0x1: 打开 IO 中断触发功能	RW	0x0
3	P23IMK	P23 中断屏蔽寄存器. 0x0: 屏蔽 IO 中断触发功能 0x1: 打开 IO 中断触发功能	RW	0x0
2	P22IMK	P22 中断屏蔽寄存器. 0x0: 屏蔽 IO 中断触发功能 0x1: 打开 IO 中断触发功能	RW	0x0
1	P21IMK	P21 中断屏蔽寄存器. 0x0: 屏蔽 IO 中断触发功能 0x1: 打开 IO 中断触发功能	RW	0x0
0	P20IMK	P20 中断屏蔽寄存器. 0x0: 屏蔽 IO 中断触发功能 0x1: 打开 IO 中断触发功能	RW	0x0

9.5.57. P2_AIOEN

Addr = 0xE8 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P27AIOEN	P27 模拟 IO 功能使能位.	WO	0x0



		0x0: 不使能 0x1: 使能		
6	P26AIOEN	P26 模拟 I0 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
5	P25AIOEN	P25 模拟 I0 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
4	P24AIOEN	P24 模拟 I0 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
3	P23AIOEN	P23 模拟 I0 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
2	P22AIOEN	P22 模拟 I0 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
1	P21AIOEN	P21 模拟 I0 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
0	P20AIOEN	P20 模拟 I0 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0

9.5.58. P2_AIOEN1

Addr = 0xEC (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P27AIOEN1	P27 模拟 I0 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
6	P26AIOEN1	P26 模拟 I0 功能 1 使能位.	WO	0x0



		0x0: 不使能 0x1: 使能		
5	P25AIOEN1	P25 模拟 IO 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
4	P24AIOEN1	P24 模拟 IO 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
3	P23AIOEN1	P23 模拟 IO 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
2	P22AIOEN1	P22 模拟 IO 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
1	P21AIOEN1	P21 模拟 IO 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
0	P20AIOEN1	P20 模拟 IO 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0

9.5.59. P2_AIOEN2

Addr = 0xED (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P27AIOEN2	P27 模拟 IO 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
6	P26AIOEN2	P26 模拟 IO 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
5	P25AIOEN2	P25 模拟 IO 功能 2 使能位.	WO	0x0



		0x0: 不使能 0x1: 使能		
4	P24AIOEN2	P24 模拟 I0 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
3	P23AIOEN2	P23 模拟 I0 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
2	P22AIOEN2	P22 模拟 I0 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
1	P21AIOEN2	P21 模拟 I0 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
0	P20AIOEN2	P20 模拟 I0 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0

9.5.60. P2_DRV0

Addr = 0xE9 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P20DRVE	P20 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P20DRV	P20 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA	WO	0x0



		 0xF: 64mA Note: 每增加一个档位，驱动能力增加 4mA！		

9.5.61. P2_DRV1

Addr = 0xB6 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P21DRVE	P21 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1，在原来配置驱动电流档位上再增加 12mA 的驱动能力！	WO	0x0
3: 0	P21DRV	P21 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位，驱动能力增加 4mA！	WO	0x0

9.5.62. P2_DRV2

Addr = 0xB7 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	—
4	P22DRVE	P22 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力	WO	0x0



		Note: 该位寄存器配置为 1，在原来配置驱动电流档位上再增加 12mA 的驱动能力！		
3: 0	P22DRV	P22 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位，驱动能力增加 4mA！	WO	0x0

9.5.63. P2_DRV3

Addr = 0xB8 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P23DRVE	P23 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1，在原来配置驱动电流档位上再增加 12mA 的驱动能力！	WO	0x0
3: 0	P23DRV	P23 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位，驱动能力增加 4mA！	WO	0x0

9.5.64. P2_DRV4

Addr = 0xB9 (XSFR)



Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	—
4	P24DRVE	P24 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P24DRV	P24 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0

9.5.65. P2_DRV5

Addr = 0xBA (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P25DRVE	P25 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P25DRV	P25 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA	WO	0x0



		Note: 每增加一个档位，驱动能力增加 4mA！		

9.5.66. P2_DRV6

Addr = 0xBB (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	—
4	P26DRVE	P26 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1，在原来配置驱动电流档位上再增加 12mA 的驱动能力！	WO	0x0
3: 0	P26DRV	P26 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位，驱动能力增加 4mA！	WO	0x0

9.5.67. P2_DRV7

Addr = 0xBC (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P27DRVE	P27 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1，在原来配置驱动电流档位上再增加 12mA 的驱动能力！	WO	0x0



3: 0	P27DRV	P27 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0

9.5.68. P2_ODN

Addr = 0xEA (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P27ODN	P27 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
6	P26ODN	P26 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
5	P25ODN	P25 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
4	P24ODN	P24 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
3	P23ODN	P23 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0



2	P220DN	P22 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
1	P210DN	P21 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
0	P200DN	P20 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0

9.5.69. P2_ODP

Addr = 0xEB (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	P270DP	P27 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
6	P260DP	P26 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
5	P250DP	P25 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
4	P240DP	P24 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0



		出高阻态)		
3	P230DP	P23 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
2	P220DP	P22 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
1	P210DP	P21 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
0	P200DP	P20 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0

9.5.70. P3

Addr = 0xB0 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	P3	PORT3 数据寄存器	RW	0x00

9.5.71. P3_PU

Addr = 0xF0 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 2	—	—	—	—
1	P31PU	P31 30KΩ 上拉电阻使能位.	RW	0x0



		0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻		
0	P30PU	P30 30K Ω 上拉电阻使能位. 0x0: 关闭 30K Ω 上拉电阻 0x1: 使能 30K Ω 上拉电阻	RW	0x0

9.5.72. P3_PD

Addr = 0xF1 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 2	—	—	—	—
1	P31PD	P31 30K Ω 下拉电阻使能位. 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0
0	P30PD	P30 30K Ω 下拉电阻使能位. 0x0: 关闭 30K Ω 下拉电阻 0x1: 使能 30K Ω 下拉电阻	RW	0x0

9.5.73. P3_MD0

Addr = 0xF2 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 4	—	—	—	—
3: 2	P31MD	P31 工作模式寄存器. 0x0: GPIO 输入模式 0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式 (数字功能关闭模式)	RW	0x0
1: 0	P30MD	P30 工作模式寄存器. 0x0: GPIO 输入模式	RW	0x0



		0x1: GPIO 输出模式 0x2: GPIO 数字功能复用选择模式 0x3: GPIO 模拟 IO 工作模式 (数字功能关闭模式)		

9.5.74. P3_MD1

Addr = 0xF3 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	—	—	—	—

9.5.75. P3_AF0

Addr = 0x16C (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 2	—	—	—	—
1	P31AF	P31 数字外设功能复用选择配置寄存器. 0x0: SPI MISO(I01) 功能选择 0x1: I2C_SCL 功能选择	WO	0x0
0	P30AF	P30 数字外设功能复用选择配置寄存器. 0x0: SPI MOSI(I00) 功能选择 0x1: I2C_SDA 功能选择	WO	0x0

9.5.76. P3_TRG0

Addr = 0xF4 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 4	—	—	—	—
3: 2	P31TRG	P31 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断	RW	0x0



		0x2: 上升沿触发中断 0x3: 下降沿触发中断		
1: 0	P30TRG	P30 中断触发配置寄存器. 0x0: 双边沿触发中断 0x1: 下降沿触发中断 0x2: 上升沿触发中断 0x3: 下降沿触发中断	RW	0x0

9.5.77. P3_TRG1

Addr = 0xF5 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	—	—	—	—

9.5.78. P3_PND

Addr = 0xF6 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 2	—	—	—	—
1	P31PND	P31 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0
0	P30PND	P30 中断 PENDING 寄存器. 0x0: 没有中断 PENDING 0x1: 有中断 PENDING	RW	0x0

Note: CPU 写 P3_PND 操作，则清掉 P30~P31 所有的中断标志位。所以中断使用时，中断处理函数优先将 P3_PND 通过变量进行保存，在处理完函数后，才统一进行标志位清除。如果对中断标志位 P3_PND 使用比较严格应用，建议使用不同组的 GPIO 中断。



9.5.79. P3_IMK

Addr = 0xF7 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 2	–	–	–	–
1	P31IMK	P31 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能 0x1: 打开 I0 中断触发功能	RW	0x0
0	P30IMK	P30 中断屏蔽寄存器. 0x0: 屏蔽 I0 中断触发功能 0x1: 打开 I0 中断触发功能	RW	0x0

9.5.80. P3_AIOEN

Addr = 0xF8 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 2	–	–	–	–
1	P31AIOEN	P31 模拟 I0 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0
0	P30AIOEN	P30 模拟 I0 功能使能位. 0x0: 不使能 0x1: 使能	WO	0x0

9.5.81. P3_AIOEN1

Addr = 0xFC (XSFR)

Bit(s)	Name	Description	R/W	Reset



7: 2	—	—	—	—
1	P31AIOEN1	P31 模拟 I/O 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
0	P30AIOEN1	P30 模拟 I/O 功能 1 使能位. 0x0: 不使能 0x1: 使能	WO	0x0

9.5.82. P3_AIOEN2

Addr = 0xFD (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 2	—	—	—	—
1	P31AIOEN2	P31 模拟 I/O 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0
0	P30AIOEN2	P30 模拟 I/O 功能 2 使能位. 0x0: 不使能 0x1: 使能	WO	0x0

9.5.83. P3_DRV0

Addr = 0xF9 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P30DRVE	P30 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0



3: 0	P30DRV	P30 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0

9.5.84. P3_DRV1

Addr = 0x5B (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	P31DRVE	P31 驱动电流增强配置寄存器. 0x0: 不额外增加 12mA 驱动能力 0x1: 再额外增加 12mA 驱动能力 Note: 该位寄存器配置为 1, 在原来配置驱动电流档位上再增加 12mA 的驱动能力!	WO	0x0
3: 0	P31DRV	P31 驱动电流档位配置寄存器. 0x0: 4mA 0x1: 8mA 0x2: 12mA 0xF: 64mA Note: 每增加一个档位, 驱动能力增加 4mA!	WO	0x0

9.5.85. P3_ODN

Addr = 0xFA (XSFR)

Bit(s)	Name	Description	R/W	Reset



7: 2	—	—	—	—
1	P310DN	P31 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0
0	P300DN	P30 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=0 时输出 0; DIN_VDD=1 时输出高阻态)	RW	0x0

9.5.86. P3_ODP

Addr = 0xFB (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 2	—	—	—	—
1	P310DP	P31 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0
0	P300DP	P30 开漏功能使能位. 0x0: 不使能 0x1: 使能 (DIN_VDD=1 时输出 1; DIN_VDD=0 时输出高阻态)	WO	0x0

9.5.87. FOUT_S00

Addr = 0x17E (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P00FOUTS	P00 输出功能选择.	WO	—



		0x0: 选择 P00AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7(FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9(FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8(FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0		



		0x1B: 选择 led_com1		
		0x1C: 选择 led_com2		
		0x1D: 选择 led_com3		
		0x1E: 选择 led_com4		
		0x1F: 选择 led_com5		

9.5.88. FOUT_S01

Addr = 0x17F (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	–	–	–	–
4: 0	P01FOUTS	P01 输出功能选择. 0x0: 选择 P01AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7(FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o	WO	–



		0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9 (FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8 (FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.89. FOUT_S02

Addr = 0x180 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P02FOUTS	P02 输出功能选择. 0x0: 选择 P02AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or	WO	—



		led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7(FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9(FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8(FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		



9.5.90. FOUT_S03

Addr = 0x181 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P03FOUTS	P03 输出功能选择. 0x0: 选择 P03AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10 (FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11 (FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6 (FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7 (FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io (FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9 (FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8 (FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1	WO	—



		0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.91. FOUT_S04

Addr = 0x182 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P04FOUTS	P04 输出功能选择. 0x0: 选择 P04AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10 (FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11 (FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6 (FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7 (FOUT_SEL[5]=1)	WO	—



		0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9(FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8(FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.92. FOUT_S05

Addr = 0x183 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—



4: 0	P05FOUTS	P05 输出功能选择. 0x0: 选择 P05AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7(FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9(FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8(FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7	WO	—



		0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.93. FOUT_S06

Addr = 0x184 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P06FOUTS	P06 输出功能选择. 0x0: 选择 P06AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7(FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o	WO	—



		0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9 (FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8 (FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.94. FOUT_S07

Addr = 0x185 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P07FOUTS	P07 输出功能选择. 0x0: 选择 P07AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx	WO	—



		0x5: 选择 stmr5_pwmout or led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7(FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9(FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8(FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		



9.5.95. FOUT_S10

Addr = 0x186 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P10FOUTS	P10 输出功能选择. 0x0: 选择 P10AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10 (FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11 (FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6 (FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7 (FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io (FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9 (FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8 (FOUT_SEL[0]=1)	WO	—



		0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.96. FOUT_S11

Addr = 0x187 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P11FOUTS	P11 输出功能选择. 0x0: 选择 P11AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1)	WO	—



		0x8: 选择 stmr2_pwmout or led_com7 (FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io (FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9 (FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8 (FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.97. FOUT_S12

Addr = 0x188 (XSFR)



Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P12FOUTS	P12 输出功能选择. 0x0: 选择 P12AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10 (FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11 (FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6 (FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7 (FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io (FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9 (FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8 (FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4	WO	—



		0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.98. FOUT_S13

Addr = 0x189 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P13FOUTS	P13 输出功能选择. 0x0: 选择 P13AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10 (FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11 (FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6 (FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7 (FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out	WO	—



		0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9(FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8(FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.99. FOUT_S14

Addr = 0x18A (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P14FOUTS	P14 输出功能选择. 0x0: 选择 P14AF 功能输出 0x1: 选择 cmp1_dout_dig	WO	—



		0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7(FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9(FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8(FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2		



		0x1D: 选择 led_com3		
		0x1E: 选择 led_com4		
		0x1F: 选择 led_com5		

9.5.100. FOUT_S15

Addr = 0x18B (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P15FOUTS	P15 输出功能选择. 0x0: 选择 P15AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7(FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or	WO	—



		led_seg9(FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8(FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.101. FOUT_S16

Addr = 0x18C (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P16FOUTS	P16 输出功能选择. 0x0: 选择 P16AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or	WO	—



		led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7(FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9(FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8(FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		



9.5.102. FOUT_S17

Addr = 0x18D (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P17FOUTS	P17 输出功能选择. 0x0: 选择 P17AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10 (FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11 (FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6 (FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7 (FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io (FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9 (FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8 (FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1	WO	0x0



		0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.103. FOUT_S20

Addr = 0x18E (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P20FOUTS	P20 输出功能选择. 0x0: 选择 P20AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10 (FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11 (FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6 (FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7 (FOUT_SEL[5]=1)	WO	—



		0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9(FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8(FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.104. FOUT_S21

Addr = 0x18F (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—



4: 0	P21FOUTS	P21 输出功能选择. 0x0: 选择 P21AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7(FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9(FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8(FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7	WO	—



		0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.105. FOUT_S22

Addr = 0x190 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P22FOUTS	P22 输出功能选择. 0x0: 选择 P22AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7(FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o	WO	—



		0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9 (FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8 (FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.106. FOUT_S23

Addr = 0x191 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P23FOUTS	P23 输出功能选择. 0x0: 选择 P23AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx	WO	—



		0x5: 选择 stmr5_pwmout or led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7(FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9(FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8(FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		



9.5.107. FOUT_S24

Addr = 0x192 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P24FOUTS	P24 输出功能选择. 0x0: 选择 P24AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10 (FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11 (FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6 (FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7 (FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io (FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9 (FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8 (FOUT_SEL[0]=1)	WO	—



		0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.108. FOUT_S25

Addr = 0x193 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	–	–	–	–
4: 0	P25FOUTS	P25 输出功能选择. 0x0: 选择 P25AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1)	WO	–



		0x8: 选择 stmr2_pwmout or led_com7 (FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io (FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9 (FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8 (FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.109. FOUT_S26

Addr = 0x194 (XSFR)



Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P26FOUTS	P26 输出功能选择. 0x0: 选择 P26AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10 (FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11 (FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6 (FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7 (FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io (FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9 (FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8 (FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4	WO	—



		0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.110. FOUT_S27

Addr = 0x195 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P27FOUTS	P27 输出功能选择. 0x0: 选择 P27AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7(FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out	WO	—



		0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9(FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8(FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.111. FOUT_S30

Addr = 0x196 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P30FOUTS	P30 输出功能选择. 0x0: 选择 P30AF 功能输出 0x1: 选择 cmp1_dout_dig	WO	—



		0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7(FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or led_seg9(FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8(FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2		



		0x1D: 选择 led_com3		
		0x1E: 选择 led_com4		
		0x1F: 选择 led_com5		

9.5.112. FOUT_S31

Addr = 0x197 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	P31FOUTS	P31 输出功能选择. 0x0: 选择 P31AF 功能输出 0x1: 选择 cmp1_dout_dig 0x2: 选择 cmp0_dout_dig 0x3: 选择 uart1_tx 0x4: 选择 uart0_tx 0x5: 选择 stmr5_pwmout or led_seg10(FOUT_SEL[2]=1) 0x6: 选择 stmr4_pwmout or led_seg11(FOUT_SEL[3]=1) 0x7: 选择 stmr3_pwmout or led_com6(FOUT_SEL[4]=1) 0x8: 选择 stmr2_pwmout or led_com7(FOUT_SEL[5]=1) 0x9: 选择 stmr1_pwmout 0xA: 选择 stmr0_pwmout 0xB: 选择 buz_out 0xC: 选择 wut_pwm_o or Clk_to_io(FOUT_SEL[6]=1) 0xD: 选择 tmr4_pwm_o 0xE: 选择 tmr3_pwm_o 0xF: 选择 tmr2_pwm_o 0x10: 选择 tmr1_pwm_o or	WO	—



		led_seg9 (FOUT_SEL[1]=1) 0x11: 选择 tmr0_pwm_o or led_seg8 (FOUT_SEL[0]=1) 0x12: 选择 led_seg0 0x13: 选择 led_seg1 0x14: 选择 led_seg2 0x15: 选择 led_seg3 0x16: 选择 led_seg4 0x17: 选择 led_seg5 0x18: 选择 led_seg6 0x19: 选择 led_seg7 0x1A: 选择 led_com0 0x1B: 选择 led_com1 0x1C: 选择 led_com2 0x1D: 选择 led_com3 0x1E: 选择 led_com4 0x1F: 选择 led_com5		

9.5.113. FOUT_SEL

Addr = 0x198 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	FOUT_SEL	功能输出 io 复用通道的选择位	WO	—

9.5.114. FIN_S0

Addr = 0x16E (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—



4: 0	TOCAPFINS	tmr0_cap_pin 输入功能 pin 脚选择. 0x0: 选择固定输入低电平 0x1: 选择 P00 0x2: 选择 P01 0x3: 选择 P02 0x4: 选择 P03 0x5: 选择 P04 0x6: 选择 P05 0x7: 选择 P06 0x8: 选择 P07 0x9: 选择 P10 0xA: 选择 P11 0xB: 选择 P12 0xC: 选择 P13 0xD: 选择 P14 0xE: 选择 P15 0xF: 选择 P16 0x10: 选择 P17 0x11: 选择 P20 0x12: 选择 P21 0x13: 选择 P22 0x14: 选择 P23 0x15: 选择 P24 0x16: 选择 P25 0x17: 选择 P26 0x18: 选择 P27 0x19: 选择 P30 0x1A: 选择 P31	W0	0x0

9.5.115. FIN_S1

Addr = 0x16F (XSFR)



Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	T1CAPFINS	tmr1_cap_pin 输入功能 pin 脚选择. 0x0: 选择固定输入低电平 0x1: 选择 P00 0x2: 选择 P01 0x3: 选择 P02 0x4: 选择 P03 0x5: 选择 P04 0x6: 选择 P05 0x7: 选择 P06 0x8: 选择 P07 0x9: 选择 P10 0xA: 选择 P11 0xB: 选择 P12 0xC: 选择 P13 0xD: 选择 P14 0xE: 选择 P15 0xF: 选择 P16 0x10: 选择 P17 0x11: 选择 P20 0x12: 选择 P21 0x13: 选择 P22 0x14: 选择 P23 0x15: 选择 P24 0x16: 选择 P25 0x17: 选择 P26 0x18: 选择 P27 0x19: 选择 P30 0x1A: 选择 P31	WO	0x0



9.5.116. FIN_S2

Addr = 0x170 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	T2CAPFINS	tmr2_cap_pin 输入功能 pin 脚选择. 0x0: 选择固定输入低电平 0x1: 选择 P00 0x2: 选择 P01 0x3: 选择 P02 0x4: 选择 P03 0x5: 选择 P04 0x6: 选择 P05 0x7: 选择 P06 0x8: 选择 P07 0x9: 选择 P10 0xA: 选择 P11 0xB: 选择 P12 0xC: 选择 P13 0xD: 选择 P14 0xE: 选择 P15 0xF: 选择 P16 0x10: 选择 P17 0x11: 选择 P20 0x12: 选择 P21 0x13: 选择 P22 0x14: 选择 P23 0x15: 选择 P24 0x16: 选择 P25 0x17: 选择 P26 0x18: 选择 P27 0x19: 选择 P30 0x1A: 选择 P31	WO	0x0



9.5.117. FIN_S3

Addr = 0x171 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	–	–	–	–
4: 0	T3CAPFINS	tmr3_cap_pin 输入功能 pin 脚选择. 0x0: 选择固定输入低电平 0x1: 选择 P00 0x2: 选择 P01 0x3: 选择 P02 0x4: 选择 P03 0x5: 选择 P04 0x6: 选择 P05 0x7: 选择 P06 0x8: 选择 P07 0x9: 选择 P10 0xA: 选择 P11 0xB: 选择 P12 0xC: 选择 P13 0xD: 选择 P14 0xE: 选择 P15 0xF: 选择 P16 0x10: 选择 P17 0x11: 选择 P20 0x12: 选择 P21 0x13: 选择 P22 0x14: 选择 P23 0x15: 选择 P24 0x16: 选择 P25 0x17: 选择 P26 0x18: 选择 P27	WO	0x0



		0x19: 选择 P30 0x1A: 选择 P31		

9.5.118. FIN_S4

Addr = 0x172 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	–	–	–	–
4: 0	T4CAPOFINS	tmr4_cap0_pin 输入功能 pin 脚选择. 0x0: 选择固定输入低电平 0x1: 选择 P00 0x2: 选择 P01 0x3: 选择 P02 0x4: 选择 P03 0x5: 选择 P04 0x6: 选择 P05 0x7: 选择 P06 0x8: 选择 P07 0x9: 选择 P10 0xA: 选择 P11 0xB: 选择 P12 0xC: 选择 P13 0xD: 选择 P14 0xE: 选择 P15 0xF: 选择 P16 0x10: 选择 P17 0x11: 选择 P20 0x12: 选择 P21 0x13: 选择 P22 0x14: 选择 P23 0x15: 选择 P24 0x16: 选择 P25	WO	0x0



		0x17: 选择 P26 0x18: 选择 P27 0x19: 选择 P30 0x1A: 选择 P31		

9.5.119. FIN_S5

Addr = 0x173 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	T4CAP1FINS	tmr4_cap1_pin 输入功能 pin 脚选择. 0x0: 选择固定输入低电平 0x1: 选择 P00 0x2: 选择 P01 0x3: 选择 P02 0x4: 选择 P03 0x5: 选择 P04 0x6: 选择 P05 0x7: 选择 P06 0x8: 选择 P07 0x9: 选择 P10 0xA: 选择 P11 0xB: 选择 P12 0xC: 选择 P13 0xD: 选择 P14 0xE: 选择 P15 0xF: 选择 P16 0x10: 选择 P17 0x11: 选择 P20 0x12: 选择 P21 0x13: 选择 P22 0x14: 选择 P23	WO	0x0



		0x15: 选择 P24		
		0x16: 选择 P25		
		0x17: 选择 P26		
		0x18: 选择 P27		
		0x19: 选择 P30		
		0x1A: 选择 P31		

9.5.120. FIN_S6

Addr = 0x174 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	T4CAP2FINS	tmr4_cap2_pin 输入功能 pin 脚选择. 0x0: 选择固定输入低电平 0x1: 选择 P00 0x2: 选择 P01 0x3: 选择 P02 0x4: 选择 P03 0x5: 选择 P04 0x6: 选择 P05 0x7: 选择 P06 0x8: 选择 P07 0x9: 选择 P10 0xA: 选择 P11 0xB: 选择 P12 0xC: 选择 P13 0xD: 选择 P14 0xE: 选择 P15 0xF: 选择 P16 0x10: 选择 P17 0x11: 选择 P20 0x12: 选择 P21	WO	0x0



		0x13: 选择 P22		
		0x14: 选择 P23		
		0x15: 选择 P24		
		0x16: 选择 P25		
		0x17: 选择 P26		
		0x18: 选择 P27		
		0x19: 选择 P30		
		0x1A: 选择 P31		

9.5.121. FIN_S7

Addr = 0x175 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	UORXFINS	uart0_rx 输入功能 pin 脚选择. 0x0: 选择固定输入低电平 0x1: 选择 P00 0x2: 选择 P01 0x3: 选择 P02 0x4: 选择 P03 0x5: 选择 P04 0x6: 选择 P05 0x7: 选择 P06 0x8: 选择 P07 0x9: 选择 P10 0xA: 选择 P11 0xB: 选择 P12 0xC: 选择 P13 0xD: 选择 P14 0xE: 选择 P15 0xF: 选择 P16 0x10: 选择 P17	WO	0x0



		0x11: 选择 P20 0x12: 选择 P21 0x13: 选择 P22 0x14: 选择 P23 0x15: 选择 P24 0x16: 选择 P25 0x17: 选择 P26 0x18: 选择 P27 0x19: 选择 P30 0x1A: 选择 P31		

9.5.122. FIN_S8

Addr = 0x176 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	U1RXFINS	uart1_rx 输入功能 pin 脚选择. 0x0: 选择固定输入低电平 0x1: 选择 P00 0x2: 选择 P01 0x3: 选择 P02 0x4: 选择 P03 0x5: 选择 P04 0x6: 选择 P05 0x7: 选择 P06 0x8: 选择 P07 0x9: 选择 P10 0xA: 选择 P11 0xB: 选择 P12 0xC: 选择 P13 0xD: 选择 P14 0xE: 选择 P15	WO	0x0



		0xF: 选择 P16 0x10: 选择 P17 0x11: 选择 P20 0x12: 选择 P21 0x13: 选择 P22 0x14: 选择 P23 0x15: 选择 P24 0x16: 选择 P25 0x17: 选择 P26 0x18: 选择 P27 0x19: 选择 P30 0x1A: 选择 P31		

9.5.123. FIN_S9

Addr = 0x177 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	WUCAPFINS	wut_cap_pin 输入功能 pin 脚选择. 0x0: 选择固定输入低电平 0x1: 选择 P00 0x2: 选择 P01 0x3: 选择 P02 0x4: 选择 P03 0x5: 选择 P04 0x6: 选择 P05 0x7: 选择 P06 0x8: 选择 P07 0x9: 选择 P10 0xA: 选择 P11 0xB: 选择 P12 0xC: 选择 P13	WO	0x0



		0xD: 选择 P14 0xE: 选择 P15 0xF: 选择 P16 0x10: 选择 P17 0x11: 选择 P20 0x12: 选择 P21 0x13: 选择 P22 0x14: 选择 P23 0x15: 选择 P24 0x16: 选择 P25 0x17: 选择 P26 0x18: 选择 P27 0x19: 选择 P30 0x1A: 选择 P31		

9.5.124. FIN_S10

Addr = 0x178 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	PWKOFINS	port_wkup_in0 输入功能 pin 脚选择. 0x0: 选择固定输入低电平 0x1: 选择 P00 0x2: 选择 P01 0x3: 选择 P02 0x4: 选择 P03 0x5: 选择 P04 0x6: 选择 P05 0x7: 选择 P06 0x8: 选择 P07 0x9: 选择 P10 0xA: 选择 P11	WO	0x0



		0xB: 选择 P12 0xC: 选择 P13 0xD: 选择 P14 0xE: 选择 P15 0xF: 选择 P16 0x10: 选择 P17 0x11: 选择 P20 0x12: 选择 P21 0x13: 选择 P22 0x14: 选择 P23 0x15: 选择 P24 0x16: 选择 P25 0x17: 选择 P26 0x18: 选择 P27 0x19: 选择 P30 0x1A: 选择 P31		

9.5.125. FIN_S11

Addr = 0x179 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	PWK1FINS	port_wkup_in1 输入功能 pin 脚选择. 0x0: 选择固定输入低电平 0x1: 选择 P00 0x2: 选择 P01 0x3: 选择 P02 0x4: 选择 P03 0x5: 选择 P04 0x6: 选择 P05 0x7: 选择 P06 0x8: 选择 P07	WO	0x0



		0x9: 选择 P10 0xA: 选择 P11 0xB: 选择 P12 0xC: 选择 P13 0xD: 选择 P14 0xE: 选择 P15 0xF: 选择 P16 0x10: 选择 P17 0x11: 选择 P20 0x12: 选择 P21 0x13: 选择 P22 0x14: 选择 P23 0x15: 选择 P24 0x16: 选择 P25 0x17: 选择 P26 0x18: 选择 P27 0x19: 选择 P30 0x1A: 选择 P31		

9.5.126. FIN_S12

Addr = 0x17A (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	PWK2FINS	port_wkup_in2 输入功能 pin 脚选择. 0x0: 选择固定输入低电平 0x1: 选择 P00 0x2: 选择 P01 0x3: 选择 P02 0x4: 选择 P03 0x5: 选择 P04 0x6: 选择 P05	WO	0x0



		0x7: 选择 P06 0x8: 选择 P07 0x9: 选择 P10 0xA: 选择 P11 0xB: 选择 P12 0xC: 选择 P13 0xD: 选择 P14 0xE: 选择 P15 0xF: 选择 P16 0x10: 选择 P17 0x11: 选择 P20 0x12: 选择 P21 0x13: 选择 P22 0x14: 选择 P23 0x15: 选择 P24 0x16: 选择 P25 0x17: 选择 P26 0x18: 选择 P27 0x19: 选择 P30 0x1A: 选择 P31		

9.5.127. FIN_S13

Addr = 0x17B (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	PWK3FINS	port_wkup_in3 输入功能 pin 脚选择. 0x0: 选择固定输入低电平 0x1: 选择 P00 0x2: 选择 P01 0x3: 选择 P02 0x4: 选择 P03	WO	0x0



		0x5: 选择 P04		
		0x6: 选择 P05		
		0x7: 选择 P06		
		0x8: 选择 P07		
		0x9: 选择 P10		
		0xA: 选择 P11		
		0xB: 选择 P12		
		0xC: 选择 P13		
		0xD: 选择 P14		
		0xE: 选择 P15		
		0xF: 选择 P16		
		0x10: 选择 P17		
		0x11: 选择 P20		
		0x12: 选择 P21		
		0x13: 选择 P22		
		0x14: 选择 P23		
		0x15: 选择 P24		
		0x16: 选择 P25		
		0x17: 选择 P26		
		0x18: 选择 P27		
		0x19: 选择 P30		
		0x1A: 选择 P31		

9.5.128. FIN_S14

Addr = 0x17C (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	FBFINS	fb_in 输入功能 pin 脚选择. 0x0: 选择固定输入低电平 0x1: 选择 P00 0x2: 选择 P01	WO	0x0



		0x3: 选择 P02		
		0x4: 选择 P03		
		0x5: 选择 P04		
		0x6: 选择 P05		
		0x7: 选择 P06		
		0x8: 选择 P07		
		0x9: 选择 P10		
		0xA: 选择 P11		
		0xB: 选择 P12		
		0xC: 选择 P13		
		0xD: 选择 P14		
		0xE: 选择 P15		
		0xF: 选择 P16		
		0x10: 选择 P17		
		0x11: 选择 P20		
		0x12: 选择 P21		
		0x13: 选择 P22		
		0x14: 选择 P23		
		0x15: 选择 P24		
		0x16: 选择 P25		
		0x17: 选择 P26		
		0x18: 选择 P27		
		0x19: 选择 P30		
		0x1A: 选择 P31		

9.5.129. FIN_S15

Addr = 0x17D (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	ADCETRFINS	adc_etr 输入功能 pin 脚选择. 0x0: 选择固定输入低电平	WO	0x0



		0x1: 选择 P00 0x2: 选择 P01 0x3: 选择 P02 0x4: 选择 P03 0x5: 选择 P04 0x6: 选择 P05 0x7: 选择 P06 0x8: 选择 P07 0x9: 选择 P10 0xA: 选择 P11 0xB: 选择 P12 0xC: 选择 P13 0xD: 选择 P14 0xE: 选择 P15 0xF: 选择 P16 0x10: 选择 P17 0x11: 选择 P20 0x12: 选择 P21 0x13: 选择 P22 0x14: 选择 P23 0x15: 选择 P24 0x16: 选择 P25 0x17: 选择 P26 0x18: 选择 P27 0x19: 选择 P30 0x1A: 选择 P31		

10.SPI 模块

10.1. 功能概述

- 支持两线模式和三线模式
- 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 12Mbps的通讯速率 ($F_{osc}=48\text{MHz}$)

10.2. 模块框图

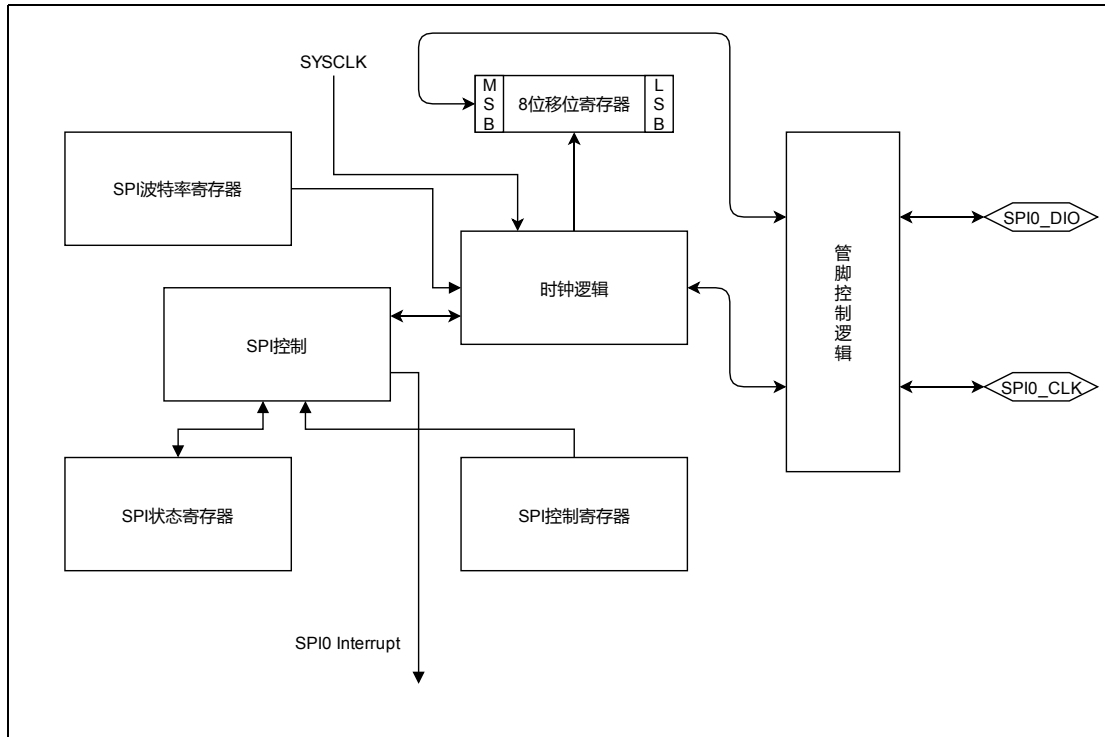


图 10-1 SPI 模块框图

10.3. 寄存器列表

表 10-1 SPI0 register list

Address	Register Name	Description
0xB9 (SFR)	SPI_CON	SPI control register
0xC9 (SFR)	SPI_BAUD	SPI baud rate register
0xCB (SFR)	SPI_DATA	SPI data register
0xCA (SFR)	SPI_STA	SPI status register



10.4. 寄存器详细说明

10.4.1. SPI_CON

Addr = 0xB9 (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	SPI2W3W	2 线或 3 线选择位 0x0: 3 线模式 0x1: 2 线模式	RW	0x0
3	SPIINTEN	SPI 中断使能位 0x0: 不使能 0x1: 使能	RW	0x0
2	SPISMPSEL	采样模式选择位, 1 代表第一个边沿采样, 0 代表第二个边沿采样 0x0: 第二个边沿采样 0x1: 第一个边沿采样	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: 不使能 0x1: 使能	RW	0x0



10.4.2. SPI_BAUD

Addr = 0xC9 (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 = $\text{clk} / (2 * (\text{BAUD} + 1))$	WO	0x00

10.4.3. SPI_DATA

Addr = 0xCB (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	DATA	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	0x00

10.4.4. SPI_STA

Addr = 0xCA (SFR)

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

10.5. 使用流程说明

- 1) 主机 TX: 配置 SPI_CON 使能位，SPIRXTX 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_CON 使能位，SPIRXTX 配 1 表示接收。写入任意数据到 DATA 触发接收，接收完成 (SPIPEND == 1) 读 DATA 读出数据。
- 3) 从机 TX: 配置 SPI_CON 使能位，SPISM 配 1 表示从机模式，SPIRXTX 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_CON 使能位, SPISM 配 1 表示从机模式, SPIRXTX 配 1 表示接收。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

- 支持全双工
- 支持发送 9bit 数据
- 支持软件奇偶校验
- UART1 支持 DMA

11.2. 模块框图

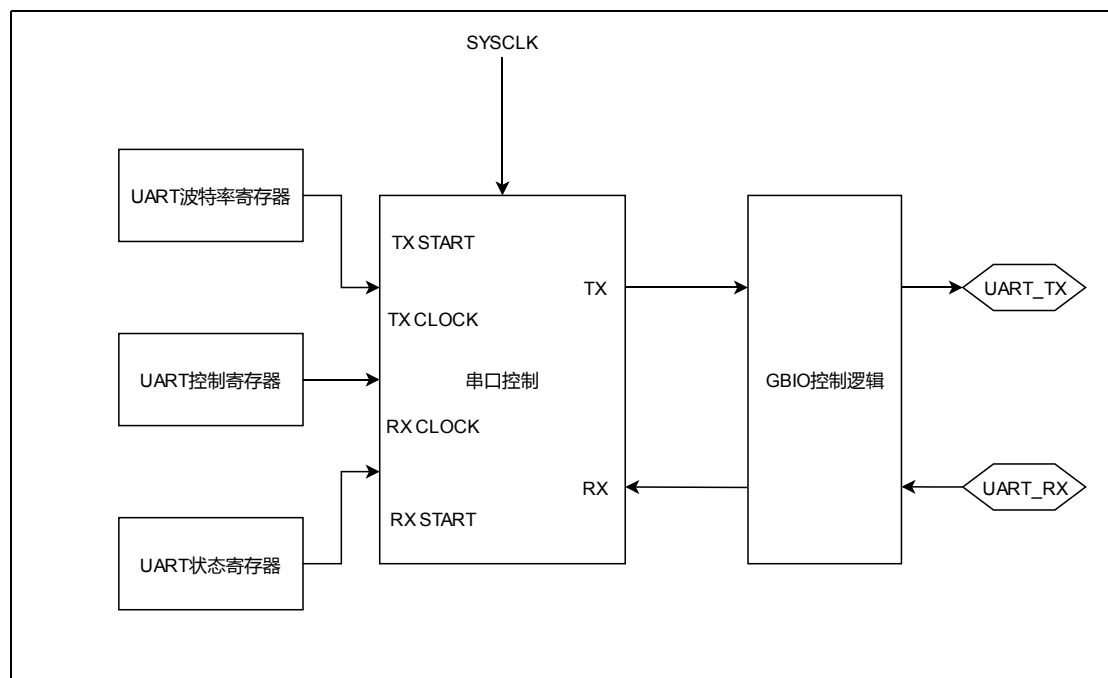


图 11-1 UART0/1 模块框图

11.3. 寄存器列表

表 11-1 SPI0 register list

address	Register Name	Description
0xD2 (SFR)	UART0_CON0	UART0 control register 0
0xD3 (SFR)	UART0_CON1	UART0 control register 1
0xD4 (SFR)	UART0_STA	UART0 status register



0xD5 (SFR)	UART0_BAUD0	The low eight bits of the UART0 baud rate register
0xD6 (SFR)	UART0_BAUD1	The high eight bits of the UART0 baud rate register
0xD7 (SFR)	UART0_DATA	UART0 data register
0xF2 (SFR)	UART1_CON0	UART1 control register 0
0xF3 (SFR)	UART1_CON1	UART1 control register 1
0xF4 (SFR)	UART1_STA	UART1 status register
0xF5 (SFR)	UART1_BAUD0	The low eight bits of the UART1 baud rate register
0xF6 (SFR)	UART1_BAUD1	The high eight bits of the UART1 baud rate register
0xF7 (SFR)	UART1_DATA	UART1 data register
0xFD (SFR)	UART1_DMACON	UART1 DMA control register
0xF9 (SFR)	UART1_DMAADRH	UART1 DMA addr high eight register
0xFA (SFR)	UART1_DMAADRL	UART1 DMA addr low eight register
0xFB (SFR)	UART1_DMALEN	UART1 DMA lenght register

11.4. 寄存器详细说明

11.4.1. UART0_CON0

Addr = 0xD2 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	STOPBIT	停止位控制位 0x0: 发送 1bit 停止位 0x1: 发送 2bit 停止位	RW	0x0
6	NINTHBIT	将需要发送的第 9bit 数据写入该寄存器	RW	0x1
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: 不使能 0x1: 使能	RW	0x0
3	TXINV	TX 电平取反控制位 0x0: 不取反	RW	0x0



		0x1: 取反		
2	RXINV	RX 电平取反控制位 0x0: 不取反 0x1: 取反	RW	0x0
1	UARTTXIE	TX 中断使能 0x0: 不使能 0x1: 使能	RW	0x0
0	UARTRXIE	RX 中断使能位 0x0: 不使能 0x1: 使能	RW	0x0

11.4.2. UART0_CON1

Addr = 0xD3 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 3	—	—	—	—
2	RXADRIE	RX 接收到地址标志位中断使能 0x0: 不使能 0x1: 使能	RW	0x0
1	FERRIE	帧错误中断使能 0x0: 不使能 0x1: 使能	RW	0x0
0	—	—	—	—

11.4.3. UART0_STA

Addr = 0xD4 (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	—
6	FERR	帧错误标志位 0x0: 没有帧错误 0x1: 接收到错误的停止位	RC	0x0



		写 1 清零		
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 DATA 清零	RC	0x0
4	TXDONE	TX 状态标志位 0x0: 正在发送数据 0x1: 空闲	RO	0x1
3	—	—	—	—
2	—	—	—	—
1	ADDRPEND	RX 状态标志位 0x0: 接收的是数据 0x1: 接收的是地址	RO	0x0
0	—	—	—	—

11.4.4. UART0_BAUD0

Addr = 0xD5 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	UARTBAUDL	波特率寄存器低八位, UART 波特率寄存器, 计算公式: $\text{sysclk}/(\text{baud}+1)$	WO	0x0

11.4.5. UART0_BAUD1

Addr = 0xD6 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	UARTBAUDH	波特率寄存器高八位, UART 波特率寄存器, 计算公式: $\text{sysclk}/(\text{baud}+1)$	WO	0x0

11.4.6. UART0_DATA

Addr = 0xD7 (SFR)

Bit(s)	Name	Description	R/W	Reset



7: 0	DATA	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

11.4.7. UART1_CON0

Addr = 0xF2 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	STOPBIT	停止位控制位: 0x0: 发送 1bit 停止位 0x1: 发送 2bit 停止位	RW	0x0
6	NINTHBIT	将需要发送的第 9bit 数据写入该寄存器	RW	0x1
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: 不使能 0x1: 使能	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	UARTTXIE	TX 中断使能 0x0: 不使能 0x1: 使能	RW	0x0
0	UARTRXIE	RX 中断使能位 0x0: 不使能 0x1: 使能	RW	0x0

11.4.8. UART1_CON1

Addr = 0xF3 (SFR)



Bit(s)	Name	Description	R/W	Reset
7: 3	—	—	—	—
2	RXADRIE	RX 接收到地址标志位中断使能 0x0: 不使能 0x1: 使能	RW	0x0
1	FERRIE	帧错误中断使能 0x0: 不使能 0x1: 使能	RW	0x0
0	DMAIE	DMA 中断使能位 0x0: 不使能 0x1: 使能	RW	0x0

11.4.9. UART1_STA

Addr = 0xF4 (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	—
6	FERR	帧错误标志位 0x0: 没有帧错误 0x1: 接收到错误的停止位 写 1 清零	RC	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 DATA 清零	RC	0x0
4	TXDONE	TX 状态标志位 0x0: 正在发送数据 0x1: 空闲	RO	0x1
3	—	—	—	—
2	DMAPEND	DMA 状态标志位 0x0: 正在忙 0x1: 空闲	RC	0x0
1	ADDRPEND	RX 状态标志位	RO	0x0



		0x0: 接收的是数据 0x1: 接收的是地址		
0	—	—	—	—

11.4.10. UART1_BAUD0

Addr = 0xF5 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	UARTBAUDL	波特率寄存器低八位, UART 波特率寄存器, 计算公式: $\text{sysclk}/(\text{baud}+1)$	WO	0x0

11.4.11. UART1_BAUD1

Addr = 0xF6 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	UARTBAUDH	波特率寄存器高八位, UART 波特率寄存器, 计算公式: $\text{sysclk}/(\text{baud}+1)$	WO	0x0

11.4.12. UART1_DATA

Addr = 0xF7 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	DATA	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

11.4.13. UART1_DMACON

Addr = 0xFD (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 4	—	—	—	—



3	TXDMAKEY	TXDMA 的使能 KEY 使能 TXDMA 时这一位需要同时置 1，否则使能无效	WO	0x0
2	RXDMAKEY	RXDMA 的使能 KEY 使能 RXDMA 时这一位需要同时置 1，否则使能无效	WO	0x0
1	TXDMAEN	TXDMA 使能信号 DMA 完成后硬件自动清零	RW	0x0
0	RXDMAEN	RXDMA 使能信号 DMA 完成后硬件自动清零	RW	0x0

11.4.14. UART1_DMAADRH

Addr = 0xF9 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	DMAADRH	DMA 地址高八位	RW	0x0

11.4.15. UART1_DMAADRL

Addr = 0xFA (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	DMAADRL	DMA 地址低八位	RW	0x0

11.4.16. UART1_DMALEN

Addr = 0xFB (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4: 0	DMALEN	DMA 长度寄存器 最大支持 31byte	RW	0x0



11.5. 使用流程说明

TX:

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 NINTHBIT 再将前 8 位数据写入 DATA 开始发送。

RX:

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 RXDONE 写 1 清零才会接收下一帧数据 (UART0->STA = BIT(5))

如需使用 DMA 则配置 DMA 控制寄存器使能 DMA，TXDMA 和 RXDMA 不能同时使用。

12.I2C 模块

12.1. 功能概述

- 支持主机模式和从机模式
- 支持主机仲裁

12.2. 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。

注意：对 I2C_STA 的值的描述(如 08H, f8H 等)都是默认寄存器低 3 位为 0 的，也就是说描述的是 I2C_STA[7: 3]左移 3 位的值。

12.2.1. 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 (I2C_CON[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(I2C_STA)被设为 08H。软件此时应该将从机地址和写命令(SLA+W)写入 I2C_DATA。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，I2C_STA 此时是 18H。此时根据需要来进行下一步操作。

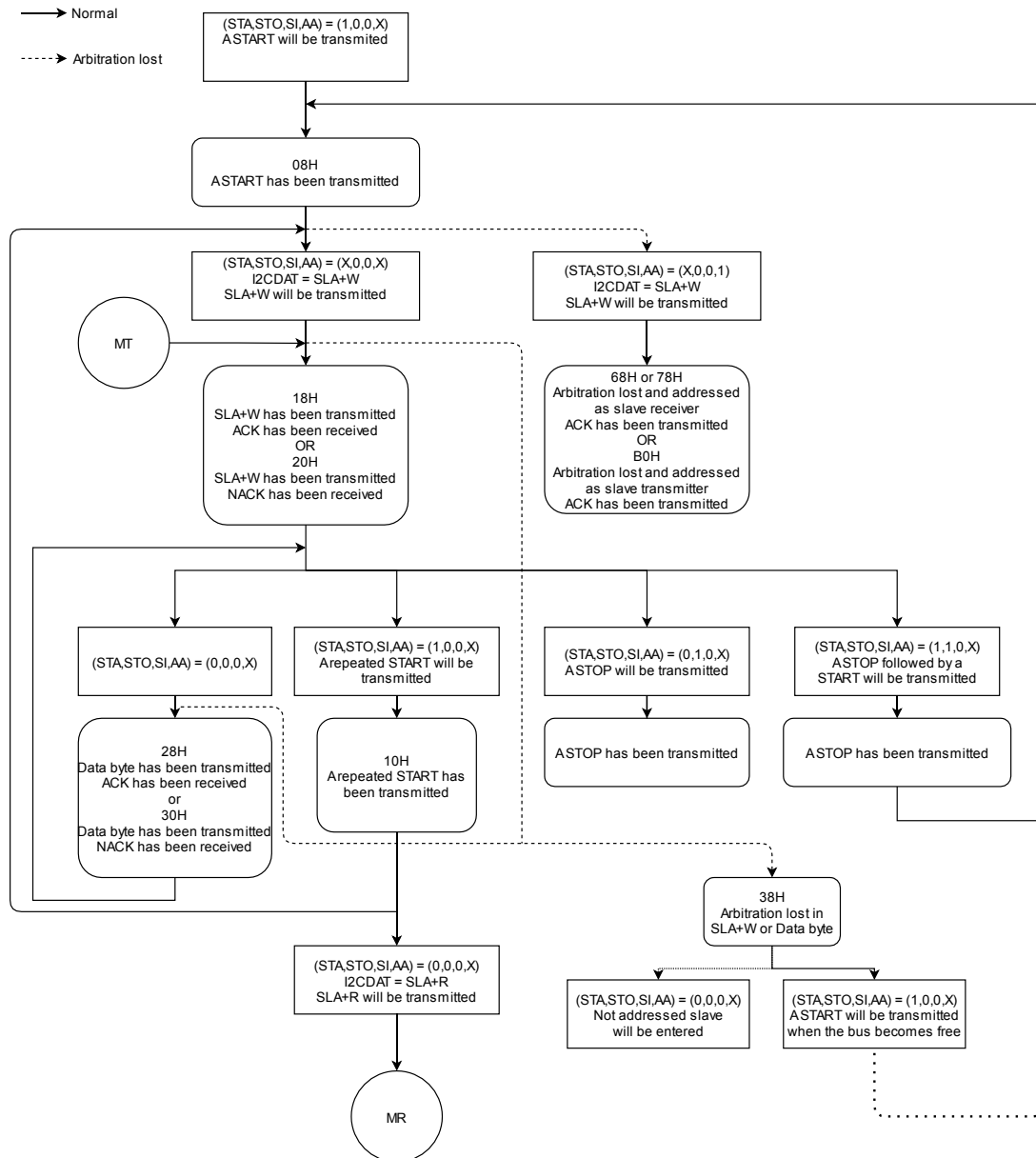


图 12-1 主机发送流程图

12.2.2. 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 I2C_DATA 的数据变为 (SLA+R)，发送成功并接收到 ACK 后 SI 会被置位，此时 I2C_STA 值为 40H。

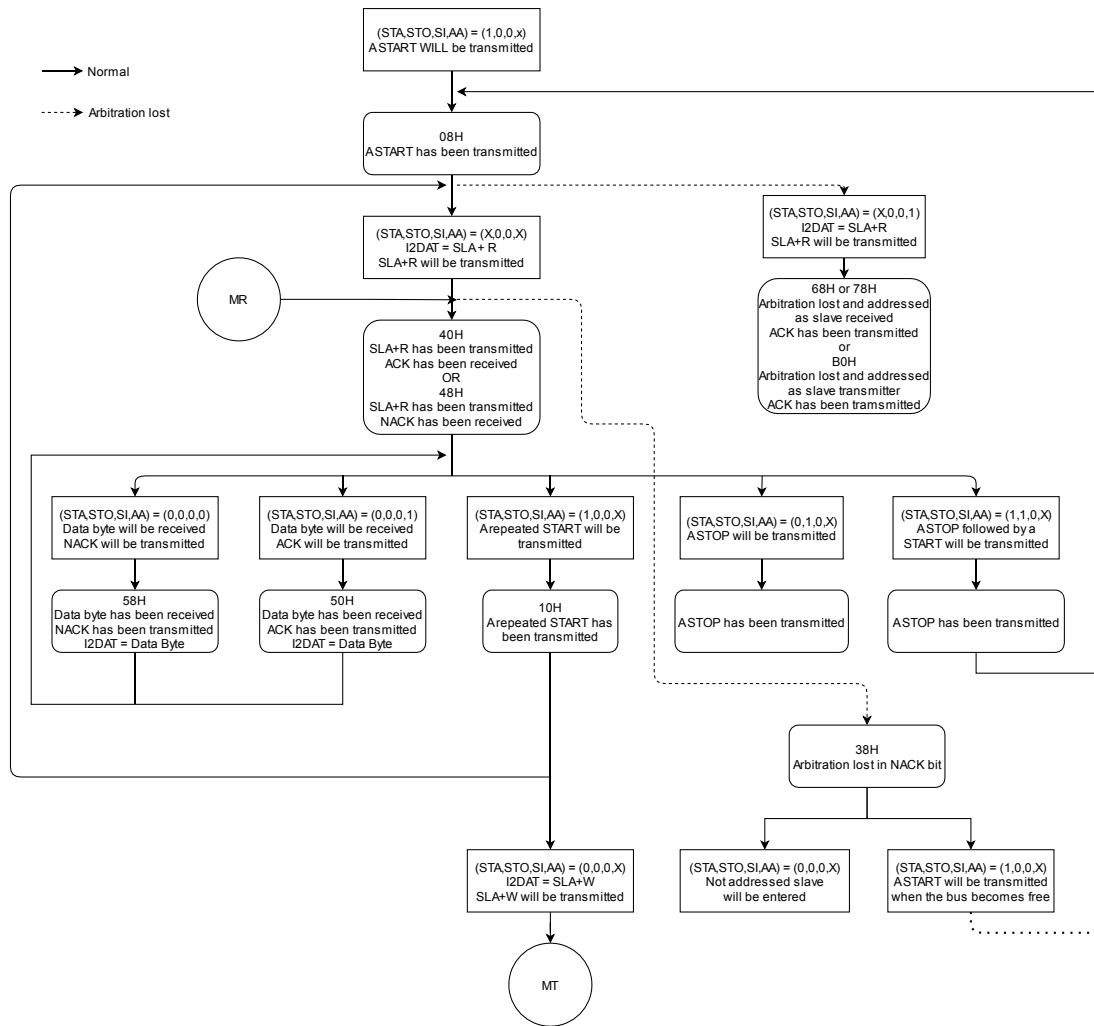


图 12-2 主机接收流程图

12.2.3. 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 I2C_ADR，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。

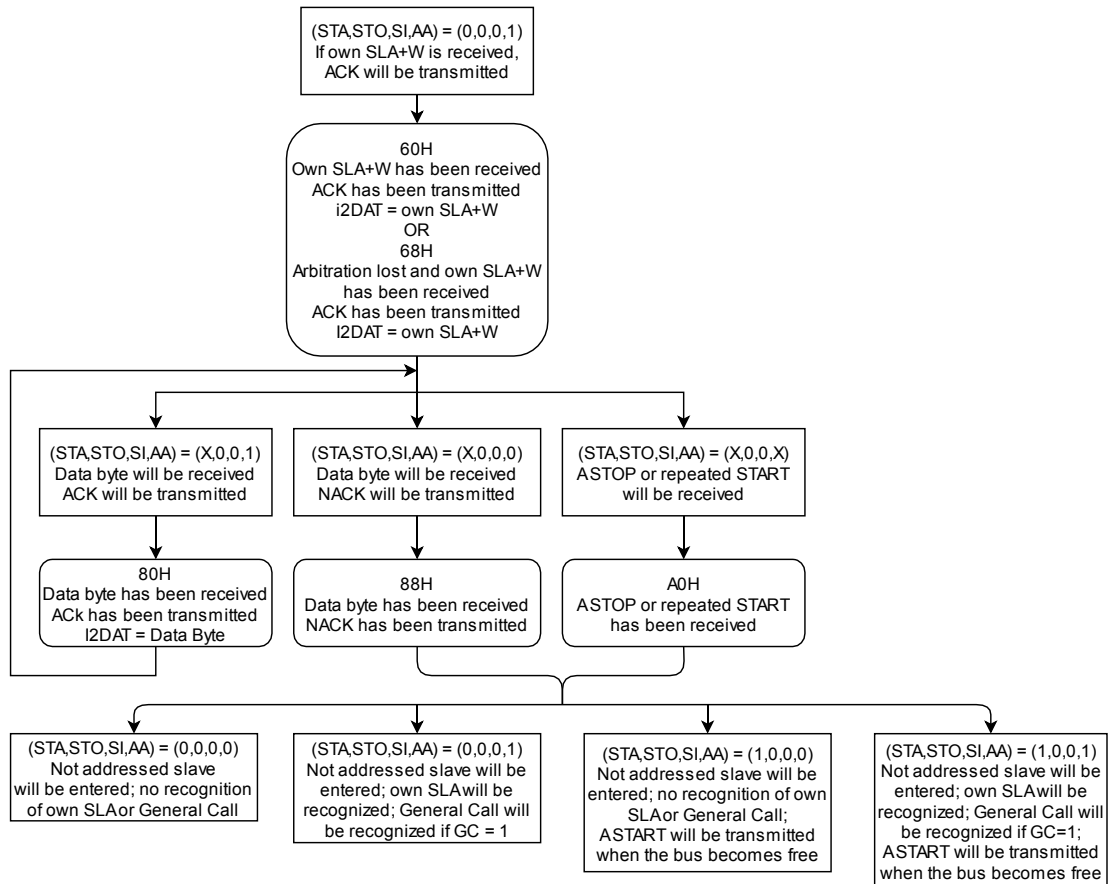


图 12-3 从机接收流程图

12.2.4. 从机发送

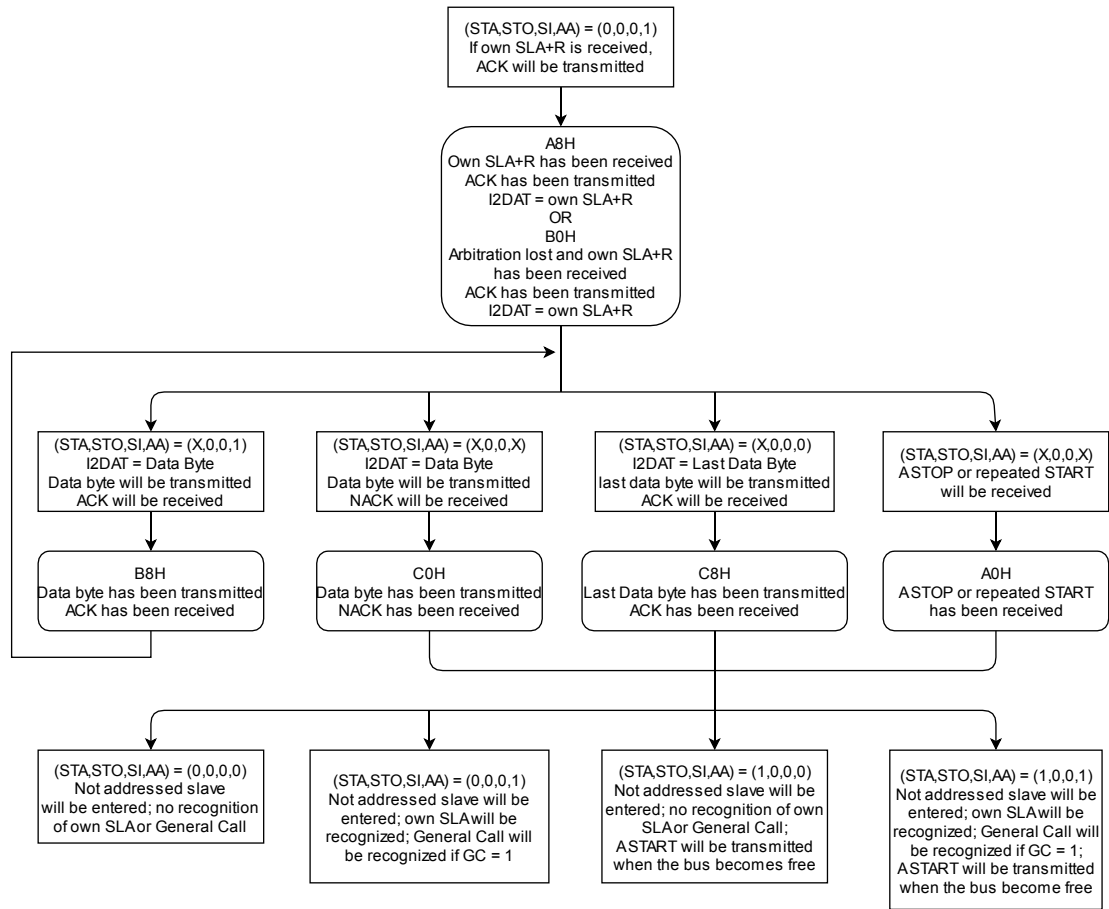


图 12-4 从机发送流程图



12.2.5. 广播模式

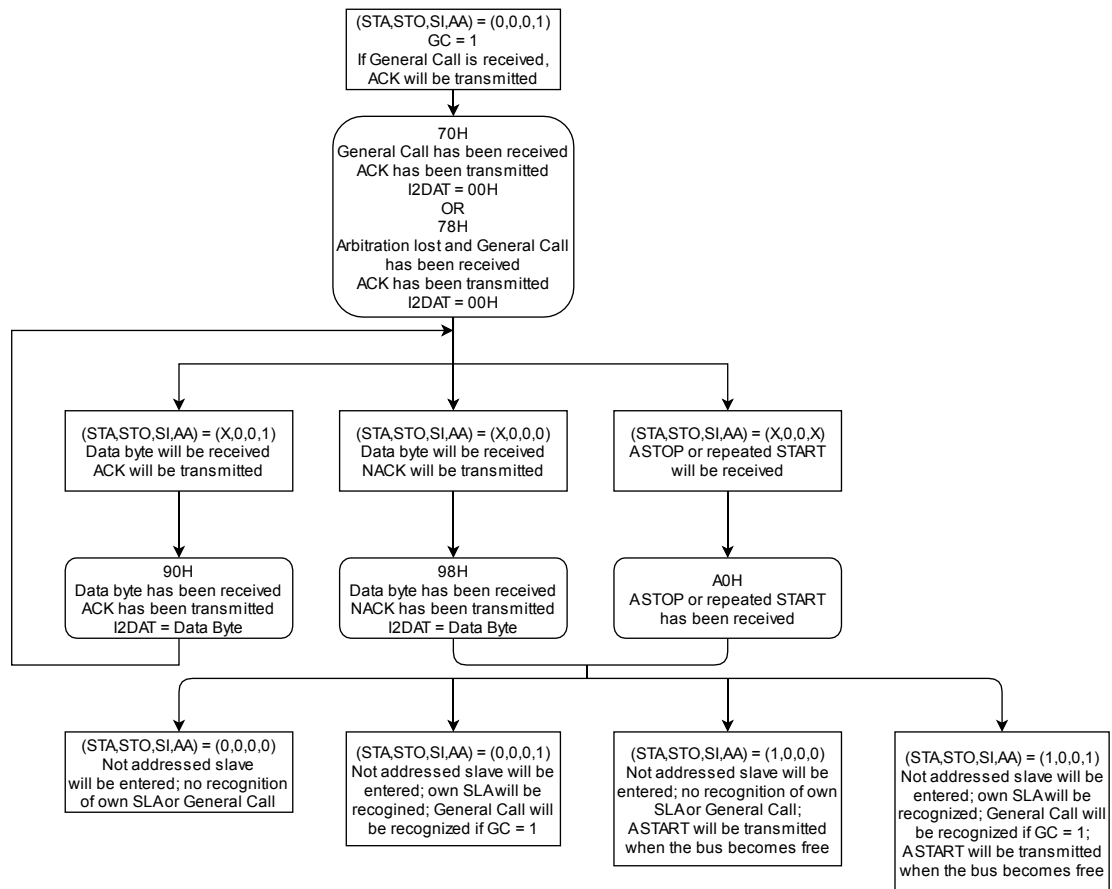


图 12-5 广播模式流程图

12.3. 寄存器列表

表 12-1 SPI0 register list

address	Register Name	Description
0xCC (SFR)	I2C_CON	I2C control register
0xCD (SFR)	I2C_DATA	I2C data register
0xCE (SFR)	I2C_ADR	I2C address register
0xCF (SFR)	I2C_STA	I2C status register



12.4. 寄存器详细说明

12.4.1. I2C_CON

Addr = 0xCC (SFR)

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	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	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	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: $\text{baud (kHz)} = \text{sysclk} / x$, 其中 x 的值由 CR 寄存器决定, 分别是: 0x00: 256 0x01: 224	RW	0x0



		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		

12.4.2. I2C_STA

Addr = 0xCD (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 3	STA	模块状态标志位	RO	0x8
2	SI	除了 f8H 状态之外的所有状态都会将 SI 置位, SI_CLR 写 1 和读写 I2C_DATA 寄存器都会清除 SI	RO	0x0
1	INT	中断标志位, 当使能了中断后, 随 SI 置位	RC	0x0
0	INTEN	中断使能位	RW	0x0

注意: 对 I2C_STA 的值的描述(如 08H, f8H 等)都是默认寄存器低 3 位为 0 的, 也就是说描述的是 I2C_STA[7: 3]左移 3 位的值

12.4.3. I2C_ADR

Addr = 0xCE (SFR)

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 0x1: 响应广播地址	RW	0x0

12.4.4. I2C_DATA

Addr = 0xCF (SFR)



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

13.MDU 运算加速模块

13.1. 功能概述

- 支持有符号数/无符号数的算术计算功能
- 支持 32 位/16 位除法，16 位/16 位除法
- 支持 16 位x16 位乘法
- 支持 16 位+16 位加法
- 支持 16 位-16 位减法
- 支持 32 位硬件快速开平方

13.2. 运算性能

假设系统时钟为 48M，每一个时钟周期为 1/48M，即 20.8333ns，以下表格给出的运算时间则以系统时钟设置 48M为例：

运算类型	32/32	开方	16-16	32-32	16+16	16x16	16/16	移位
所需时钟数	32	43	10	14	10	21	21	10
运算时间(us)	0.67	0.90	0.21	0.29	0.21	0.44	0.44	0.21

注意：以上数据仅为估算时间，代表的是从使能运算开始到运算结束所需时间，可能存在误差。操作数地址和结果地址为 2 byte对齐时可节省模块内部DMA的时间，从而获得最大性能。



13.3. 模块框图

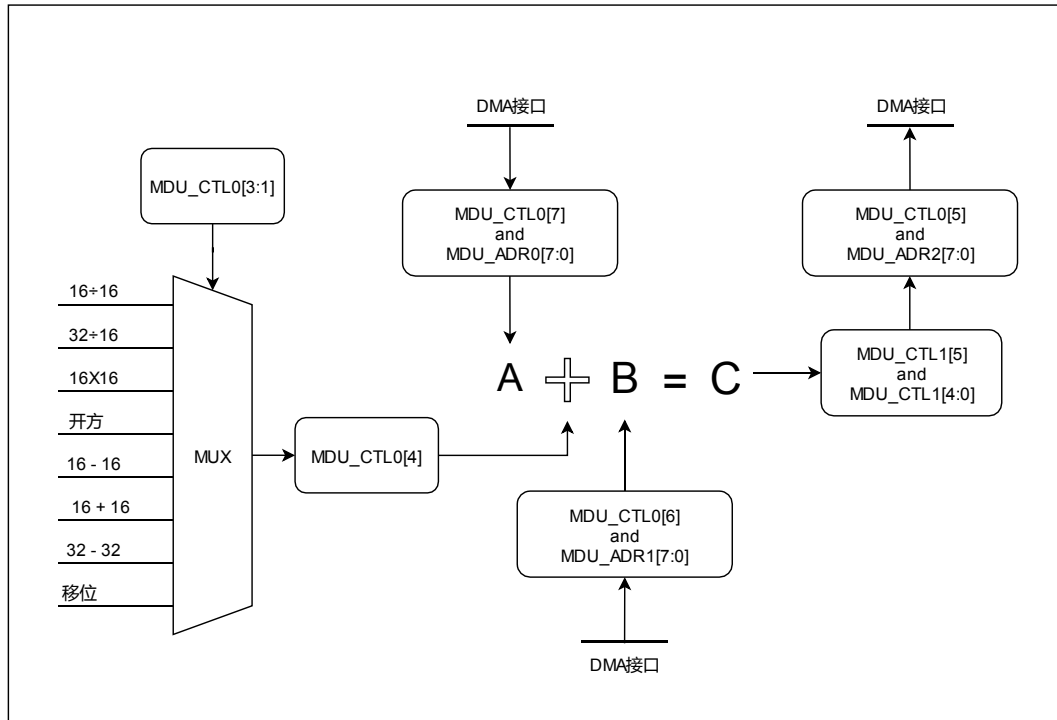


图 13-1 MDU 运算加速模块框图

13.4. 寄存器列表

表 13-1 MDU register list

Address	Register Name	Description
0x88 (SFR)	MDU_CTL0	MDU control register
0x89 (SFR)	MDU_CTL1	MDU control register
0x8A (SFR)	MDU_ADR0	MDU operand 0 address control register
0x8B (SFR)	MDU_ADR1	MDU operand 1 address control register
0x8C (SFR)	MDU_ADR2	MDU operand 2 address control register
0xF1 (SFR)	MDU_STA	MDU state register

13.5. 寄存器详细说明

13.5.1. MDU_CTL0

Addr =0x88 (SFR)



Bit(s)	Name	Description	R/W	Reset
7	ADROH	操作数 0 地址翻页控制位 0x0: 操作数 0 的地址页选择 data 区的 0x00~0xFF 0x1: 操作数 0 的地址页选择 xdata 区的 0x6000~60FF 注意: 操作数 0 地址由这一 bit 和 MDU_ADRO 共同决定	RW	0x0
6	ADR1H	操作数 1 地址翻页控制位 0x0: 操作数 1 的地址页选择 data 区的 0x00~0xFF 0x1: 操作数 1 的地址页选择 xdata 区的 0x6000~60FF 注意: 操作数 1 地址由这一 bit 和 MDU_ADR1 共同决定	RW	0x0
5	ADR2H	结果地址翻页控制位 0x0: 结果的地址页选择 data 区的 0x00~0xFF 0x1: 结果的地址页选择 xdata 区的 0x6000~60FF 注意: 结果地址由这一 bit 和 MDU_ADR2 共同决定	RW	0x0
4	SIGNSEL	操作数种类选择 0x0: 无符号操作数 0x1: 有符号操作数 开方不支持有符号操作, 此位不可置 1	RW	0x0
3: 1	OPTSEL	操作选择寄存器. 0x0: 16/16 运算 0x1: 32/16 运算 0x2: 16x16 运算 0x3: 开方运算 0x4: 16-16 运算 0x5: 16+16 运算 0x6: 32-32 运算 0x7: 移位运算	RW	0x0
0	MDUEN	MDU 使能位, 每次使能进行一次运算, 运算完自动清零 MDU 可被软件连续使能, 不需要等任何 pending	RW	0x0



13.5.2. MDU_CTL1

Addr = 0x89 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	—
5	SIFTSEL	左移右移控制位。 0x0: 左移 0x1: 右移	RW	0x0
4: 0	SIFTNUM	移位位数控制位，最大 31 位	RW	0x0

Note: 每一次操作完都会检查 SIFT_NUM 寄存器，如果不为零则将该次操作结果移位. 如不需移位要将 SIFT_NUM 置零. 移位是算术移位，溢出会补成最大值。

13.5.3. MDU_ADR0

Addr = 0x8A (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	ADR0	操作数 0 的地址，和 ADR0H 共同决定操作数 0 的地址	WO	0x0

13.5.4. MDU_ADR1

Addr = 0x8B (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	ADR1	操作数 1 的地址，和 ADR1H 共同决定操作数 1 的地址	WO	0x0

13.5.5. MDU_ADR2

Addr = 0x8C (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	ADR2	计算结果保存地址，和 ADR2H 共同决定结果的地	WO	0x0



		址		

13.5.6. MDU_STA

Addr = 0xF1 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 3	—	—	—	—
2	DIVERR	被除数为 0 错误标志位, 写 1 清零	RW	0x0
1	SIFTOVFL	移位溢出标志位, 写 1 清零	RW	0x0
0	ADDOVFL	加减法溢出标志位, 写 1 清零	RW	0x0

13.6. 使用流程说明

- 1) 配置操作数和结果地址
- 2) 选择要进行的运算, 使能 MDU

Note: MDU 可被软件连续使能, 不需要等任何 pending, 操作数和结果遵循大端格式.

16 位操作数格式:

ADDR	ADDR+1
数据第 15~8	数据第 7~0 位

32 位操作数格式:

ADDR	ADDR+1	ADDR+2	ADDR+3
数据第 31~24 位	数据第 23~16 位	数据第 15~8 位	数据第 7~0 位

结果格式和操作数格式一致, 除法有余数, 除法格式如下:

32/16 除法结果格式:

ADDR	ADDR+1	ADDR+2	ADDR+3	ADDR+4	ADDR+5
结 果 31~24 位	结 果 23~16 位	结果 15~8 位	结果 7~0 位	余数 15~8 位	余数 7~0 位

16/16 除法结果格式:

ADDR	ADDR+1	ADDR+2	ADDR+3



结果 15~8 位

结果 7~0 位

余数 15~8 位

余数 7~0 位

14.Simple Timer 模块

14.1. 功能概述

Simple Timer 模块是由 Timer0、Timer1、Timer2、Timer3 以及一个 Wake Up Timer 五个 16 位的基础功能定时器以及一个 8 位的 Buzzer 组成。除了 Buzzer 之外，其他 5 个 Timer 均支持多种计数源选择，支持多种工作模式：计数器模式、捕获模式和 PWM 模式等。

Simple Timer 模块均带有 8 位分频器，且计数模式均为递增计数，当计数器值等于设置的周期值时，计数器从零开始重新计数。16 位定时器，带有 1 级捕获功能，能将捕获值存入比较值寄存器中。

计数源选择：

- 1) 32K 低速 RC (LIRC)
- 2) 八分频高速 RC (HIRC)
- 3) 二分频 XOSC
- 4) 外部 GPIO 输入
- 5) 系统时钟

14.1.1. Timer0-3

14.1.1.1. 特殊计数源

Timer0 和 Timer1 可以与 Super Timer 模块中的 Stmr0 同步计数。通过配置 Super Timer 的 STMR_CNTTYPE[7: 6] 和 STMR_CNTEN[7: 6] 可以使能 Timer0 和 Timer1 与 Stmr0 进行同步计数，即 Stmr0 计数器加 1，Timer0 和 Timer1 计数器加 1。

将 IO_MAP[7] 置 1，可以使能 Timer0 计数源为 Stmr0 比较值 C 点等于计数值，即每一次 Stmr0 计数值等于比较值 C 时，Timer0 计数器加 1。

Timer0 计数值等于周期可以作为 Timer1 计数源。

Timer1 计数值等于周期可以作为 Timer2 计数源。

Timer2 计数值等于周期可以作为 Timer3 计数源。

Timer3 计数值等于周期可以作为 Wake Up Timer 计数源。



14.1.1.2. 影子寄存器

Timer0 和 Timer1 比较值寄存器拥有影子寄存器，在 PWM 模式下，计数器值等于周期值时可以将影子寄存器的值加载到比较值寄存器中。当比较值寄存器写入值时，同时也会将该值写入影子寄存器。当影子寄存器写入值时，不影响比较值寄存器。

Timer2、Timer3 比较值寄存器没有影子寄存器。

14.1.1.3. 计数器清零

通过寄存器 TMR_ALLCON[3: 0]写 1 可以清零 Timer0-Timer3 计数。

14.1.2. Wake Up Timer

Wake Up Timer 和 Timerx (x 表示 0-3) 功能一样，可以作为正常的 PWM 输出，支持 1 级捕获模式。通过寄存器 TMR_ALLCON[5]写 1 可以清零计数。其中断可以作为芯片休眠模式下的唤醒源之一，即当芯片休眠时，计数器中断有效可唤醒芯片开始工作。

14.1.3. Buzzer

Buzzer 是一个八位计数器，支持计数模式和 PWM 模式。PWM 模式占空比为 50%。计数 pending 和 PWM 输出二选一。



14.2. 模块框图

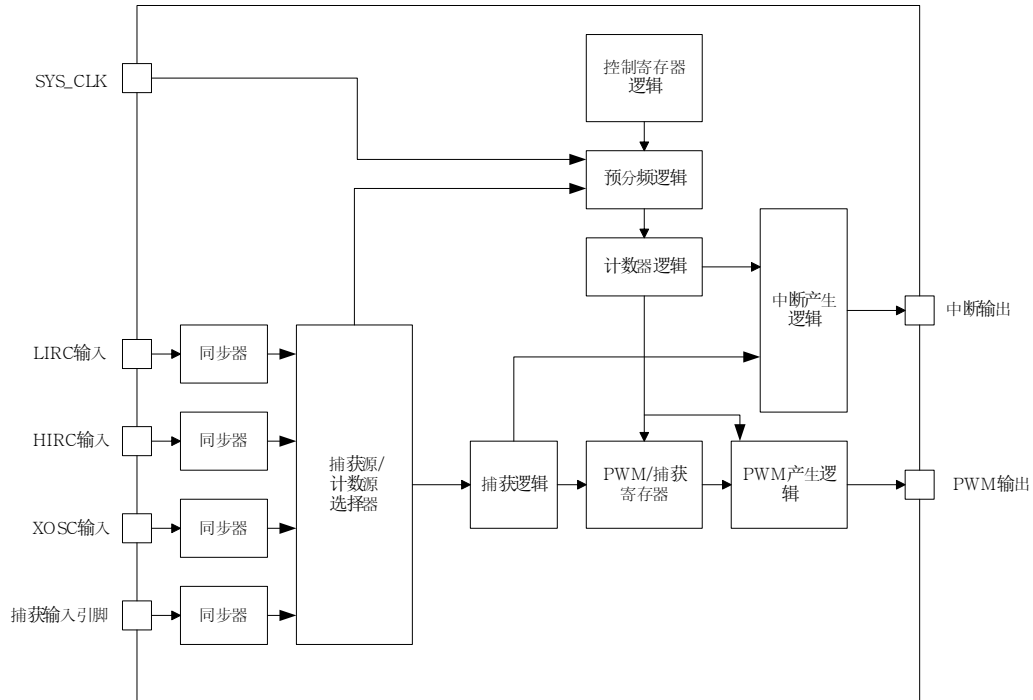


图 14-1 Simple Timer 模块框图

14.3. 寄存器列表

表 14-1 Timer0/1 register list

address	Register Name	Description
0x090 (XSFR)	TMRO_CONL	TIMER0 Control Low Register
0x091 (XSFR)	TMRO_CONH	TIMER0 Control High Register
0x092 (XSFR)	TMRO_CNTL	TIMER0 Counter Low Register
0x093 (XSFR)	TMRO_CNTH	TIMER0 Counter High Register
0x094 (XSFR)	TMRO_PRL	TIMER0 Period Low Register
0x095 (XSFR)	TMRO_PRH	TIMER0 Period High Register



0x096 (XSFR)	TMR0_PWML	TIMER0 PWM Low Register
0x097 (XSFR)	TMR0_PWMH	TIMER0 PWM High Register
0x09a (XSFR)	TMR0_PWML1	TIMER0 PWM Low Shadow Register
0x09b (XSFR)	TMR0_PWMH1	TIMER0 PWM High Shadow Register
0x09c (XSFR)	TMR1_CONL	TIMER1 Control Low Register
0x09d (XSFR)	TMR1_CONH	TIMER1 Control High Register
0x09e (XSFR)	TMR1_CNTL	TIMER1 Counter Low Register
0x09f (XSFR)	TMR1_CNTH	TIMER1 Counter High Register
0x0a0 (XSFR)	TMR1_PRL	TIMER1 Period Low Register
0x0a1 (XSFR)	TMR1_PRH	TIMER1 Period High Register
0x0a2 (XSFR)	TMR1_PWML	TIMER1 PWM Low Register
0x0a3 (XSFR)	TMR1_PWMH	TIMER1 PWM High Register
0x0a6 (XSFR)	TMR1_PWML1	TIMER1 PWM Low Shadow Register
0x0a7 (XSFR)	TMR1_PWMH1	TIMER1 PWM High Shadow Register
0x100 (XSFR)	TMR2_CONL	TIMER2 Control Low Register
0x101 (XSFR)	TMR2_CONH	TIMER2 Control High Register
0x102 (XSFR)	TMR2_CNTL	TIMER2 Counter Low Register
0x103 (XSFR)	TMR2_CNTH	TIMER2 Counter High Register
0x104 (XSFR)	TMR2_PRL	TIMER2 Period Low Register
0x105 (XSFR)	TMR2_PRH	TIMER2 Period High Register
0x106 (XSFR)	TMR2_PWML	TIMER2 PWM Low Register
0x107 (XSFR)	TMR2_PWMH	TIMER2 PWM High Register
0x108 (XSFR)	TMR3_CONL	TIMER3 Control Low Register



0x109 (XSFR)	TMR3_CONH	TIMER3 Control High Register
0x10a (XSFR)	TMR3_CNTL	TIMER3 Counter Low Register
0x10b (XSFR)	TMR3_CNTH	TIMER3 Counter High Register
0x10c (XSFR)	TMR3_PRL	TIMER3 Period Low Register
0x10d (XSFR)	TMR3_PRH	TIMER3 Period High Register
0x10e (XSFR)	TMR3_PWML	TIMER3 PWM Low Register
0x10f (XSFR)	TMR3_PWMH	TIMER3 PWM High Register
0x152 (XSFR)	WUT_CONL	Wake Up Timer Control Low Register
0x153 (XSFR)	WUT_CONH	Wake Up Timer Control High Register
0x154 (XSFR)	WUT_CNTL	Wake Up Timer Counter Low Register
0x155 (XSFR)	WUT_CNTH	Wake Up Timer Counter High Register
0x156 (XSFR)	WUT_PRL	Wake Up Timer Period Low Register
0x157 (XSFR)	WUT_PRH	Wake Up Timer Period High Register
0x158 (XSFR)	WUT_PWML	Wake Up Timer PWM Low Register
0x159 (XSFR)	WUT_PWMH	Wake Up Timer PWM High Register
0x164 (XSFR)	BUZ_CON	Buzzer Control Register
0x165 (XSFR)	BUZ_DIV	Buzzer Period Register

14.4. 寄存器详细说明

14.4.1. TMR0_CONL

Addr = 0x090 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	SOURCESEL	计数源选择	RW	0x0



		0x0: 选择 GPIO (CAP PIN) 上升沿作为计数源; 0x1: 选择 GPIO (CAP PIN) 下降沿作为计数源; 0x2: 选择 HIRC 上升沿和下降沿作为计数源; 0x3: 选择 LIRC 上升沿和下降沿作为计数源; 0x4: 选择 XOSC 上升沿和下降沿作为计数源; 0x5: 无计数源; 0x6: 选择系统时钟作为计数源; 0x7: 选择系统时钟作为计数源;		
4: 2	PSC	TIMER 预分频配置. 0x0: 1/1 0x1: 1/2 0x2: 1/4 0x3: 1/8 0x4: 1/16 0x5: 1/32 0x6: 1/64 0x7: 1/128	RW	0x0
1: 0	TMRMODE	模式选择及计数使能 0x0: 不使能计数 0x1: 计数器模式 0x2: PWM 模式 0x3: 捕获模式 Note: 非零使能计数, 其他使能方式见功能概述	RW	0x0

14.4.2. TMR0_CONH

Addr = 0x091 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	TMRPND	TIMER 计数 pending 位 (写 1 清 pending) 0x0: 没有发生计数等于周期, 或已清零 0x1: 发生过计数等于周期	RW	0x0
6	CAPPND	TIMER 捕获 pending 位 (写 1 清 pending) 0x0: 没有发生捕获事件	RW	0x0



		0x1: 有发生捕获事件		
5	TMRIE	TIMER 计数中断使能位 0x0: 不使能计数中断 0x1: 使能计数中断, 计数等于周期时允许发生中断	RW	0x0
4	CAPIE	TIMER 捕获中断使能位 0x0: 不使能捕获中断 0x1: 使能捕获中断, 发生捕获事件时允许发生中断	RW	0x0
3: 2	CAPSRC	TIMER 捕获源选择配置. 0x0: 引脚作为捕获源 0x1: 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	CAPEDGESEL	TIMER 捕获引脚的边沿触发设置. 0x0: 上升沿触发捕获 0x1: 下降沿触发捕获 0x2: 双边沿触发捕获 0x3: 双边沿触发捕获	RW	0x0

14.4.3. TMR0_CNTL

Addr = 0x092 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CNTL	TIMER0 计数器低八位	RW	—

14.4.4. TMR0_CNTH

Addr = 0x093 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CNTH	TIMER0 计数器高八位	RW	—



14.4.5. TMR0_PRL

Addr = 0x094 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PRDL	TIMER0 计数周期低八位	WO	—

14.4.6. TMR0_PRH

Addr = 0x095 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PRDH	TIMER0 计数周期高八位	WO	—

14.4.7. TMR0_PWML

Addr = 0x096 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PWML	TIMER0 占空比低八位 PWM 工作模式时，该值是 PWM 的占空比设置值；捕获工作模式时，当捕获到捕获源之后抓取的计数器低八位的值锁存在此寄存器中。	RW	—

14.4.8. TMR0_PWMH

Addr = 0x097 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PWMH	TIMER0 占空比高八位 PWM 工作模式时，该值是 PWM 的占空比设置值；捕获工作模式时，当捕获到捕获源之后抓取的计数器高八位的值锁存在此寄存器中。	RW	—



14.4.9. TMR0_PWML1

Addr = 0x09A (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PWML1	TIMER0 占空比影子寄存器低八位 PWM 工作模式时, 当计数器值等于周期值时, 将加载此寄存器中的值到占空比寄存器 PWML。	RW	—

14.4.10. TMR0_PWMH1

Addr = 0x09B (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PWMH1	TIMER0 占空比影子寄存器高八位 PWM 工作模式时, 当计数器值等于周期值时, 将加载此寄存器中的值到占空比寄存器 PWMH。	RW	—

14.4.11. TMR1_CONL

Addr = 0x09C (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	SOURCESEL	计数源选择 0x0: 选择 GPIO (CAP PIN) 上升沿作为计数源; 0x1: 选择 GPIO (CAP PIN) 下降沿作为计数源; 0x2: 选择 HIRC 上升沿和下降沿作为计数源; 0x3: 选择 LIRC 上升沿和下降沿作为计数源; 0x4: 选择 XOSC 上升沿和下降沿作为计数源; 0x5: Timer0 计数 CNT 等于周期作为计数源; 0x6: 选择系统时钟作为计数源; 0x7: 选择系统时钟作为计数源;	RW	0x0
4: 2	PSC	TIMER 预分频配置. 0x0: 1/1 0x1: 1/2	RW	0x0



		0x2: 1/4 0x3: 1/8 0x4: 1/16 0x5: 1/32 0x6: 1/64 0x7: 1/128		
1: 0	TMRMODE	模式选择及计数使能 0x0: 不使能计数 0x1: 计数器模式 0x2: PWM 模式 0x3: 捕获模式 Note: 非零使能计数, 其他使能方式见功能概述	RW	0x0

14.4.12. TMR1_CONH

Addr = 0x09D (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	TMRPND	TIMER 计数 pending 位 (写 1 清 pending) 0x0: 没有发生计数等于周期, 或已清零 0x1: 发生过计数等于周期	RW	0x0
6	CAPPND	TIMER 捕获 pending 位 (写 1 清 pending) 0x0: 没有发生捕获事件 0x1: 有发生捕获事件	RW	0x0
5	TMRIE	TIMER 计数中断使能位 0x0: 不使能计数中断 0x1: 使能计数中断, 计数等于周期时允许发生中断	RW	0x0
4	CAPIE	TIMER 捕获中断使能位 0x0: 不使能捕获中断 0x1: 使能捕获中断, 发生捕获事件时允许发生中断	RW	0x0
3: 2	CAPSRC	TIMER 捕获源选择配置. 0x0: 引脚作为捕获源	RW	0x0



		0x1: 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	CAPEDGESEL	TIMER 捕获引脚的边沿触发设置. 0x0: 上升沿触发捕获 0x1: 下降沿触发捕获 0x2: 双边沿触发捕获 0x3: 双边沿触发捕获	RW	0x0

14.4.13. TMR1_CNTL

Addr = 0x09E (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CNTL	TIMER1 计数器低八位	RW	—

14.4.14. TMR1_CNTH

Addr = 0x09F (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CNTH	TIMER1 计数器高八位	RW	—

14.4.15. TMR1_PRL

Addr = 0x0A0 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PRDL	TIMER1 计数周期低八位	WO	—

14.4.16. TMR1_PRH

Addr = 0x0A1 (XSFR)

Bit(s)	Name	Description	R/W	Reset



7: 0	PRDH	TIMER1 计数周期高八位	WO	—

14.4.17. TMR1_PWML

Addr = 0x0A2 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PWML	TIMER1 占空比低八位 PWM 工作模式时, 该值是 PWM 的占空比设置值; 捕获工作模式时, 当捕获到捕获源之后抓取的计数器低八位的值锁存在此寄存器中。	RW	—

14.4.18. TMR1_PWMH

Addr = 0x0A3 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PWMH	TIMER1 占空比高八位 PWM 工作模式时, 该值是 PWM 的占空比设置值; 捕获工作模式时, 当捕获到捕获源之后抓取的计数器高八位的值锁存在此寄存器中。	RW	—

14.4.19. TMR1_PWML1

Addr = 0x0A6 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PWML1	TIMER1 占空比影子寄存器低八位 PWM 工作模式时, 当计数器值等于周期值时, 将加载此寄存器中的值到占空比寄存器 PWML。	RW	—

14.4.20. TMR1_PWMH1

Addr = 0x0A7 (XSFR)

Bit(s)	Name	Description	R/W	Reset



7: 0	PWMH1	TIMER1 占空比影子寄存器高八位 PWM 工作模式时, 当计数器值等于周期值时, 将加载此寄存器中的值到占空比寄存器 PWMH。	RW	—

14.4.21. TMR2_CONL

Addr = 0x100 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	SOURCESEL	计数源选择 0x0: 选择 GPIO (CAP PIN) 上升沿作为计数源; 0x1: 选择 GPIO (CAP PIN) 下降沿作为计数源; 0x2: 选择 HIRC 上升沿和下降沿作为计数源; 0x3: 选择 LIRC 上升沿和下降沿作为计数源; 0x4: 选择 XOSC 上升沿和下降沿作为计数源; 0x5: Timer1 计数 CNT 等于周期作为计数源; 0x6: 选择系统时钟作为计数源; 0x7: 选择系统时钟作为计数源;	RW	0x0
4: 2	PSC	TIMER 预分频配置. 0x0: 1/1 0x1: 1/2 0x2: 1/4 0x3: 1/8 0x4: 1/16 0x5: 1/32 0x6: 1/64 0x7: 1/128	RW	0x0
1: 0	TMRMODE	模式选择及计数使能 0x0: 不使能计数 0x1: 计数器模式 0x2: PWM 模式 0x3: 捕获模式	RW	0x0



14.4.22. TMR2_CONH

Addr = 0x101 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	TMRPND	TIMER 计数 pending 位 (写 1 清 pending) 0x0: 没有发生计数等于周期, 或已清零 0x1: 发生过计数等于周期	RW	0x0
6	CAPPND	TIMER 捕获 pending 位 (写 1 清 pending) 0x0: 没有发生捕获事件 0x1: 有发生捕获事件	RW	0x0
5	TMRIE	TIMER 计数中断使能位 0x0: 不使能计数中断 0x1: 使能计数中断, 计数等于周期时允许发生中断	RW	0x0
4	CAPIE	TIMER 捕获中断使能位 0x0: 不使能捕获中断 0x1: 使能捕获中断, 发生捕获事件时允许发生中断	RW	0x0
3: 2	CAPSRC	TIMER 捕获源选择配置. 0x0: 引脚作为捕获源 0x1: 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	CAPEDGESEL	TIMER 捕获引脚的边沿触发设置. 0x0: 上升沿触发捕获 0x1: 下降沿触发捕获 0x2: 双边沿触发捕获 0x3: 双边沿触发捕获	RW	0x0

14.4.23. TMR2_CNTL

Addr = 0x102 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CNTL	TIMER2 计数器低八位	RW	—



14.4.24. TMR2_CNTH

Addr = 0x103 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CNTH	TIMER2 计数器高八位	RW	—

14.4.25. TMR2_PRL

Addr = 0x104 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PRDL	TIMER2 计数周期低八位	WO	—

14.4.26. TMR2_PRH

Addr = 0x105 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PRDH	TIMER2 计数周期高八位	WO	—

14.4.27. TMR2_PWML

Addr = 0x106 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PWML	TIMER2 占空比低八位 PWM 工作模式时，该值是 PWM 的占空比设置值；捕获工作模式时，当捕获到捕获源之后抓取的计数器低八位的值锁存在此寄存器中。	RW	—

14.4.28. TMR2_PWMH

Addr = 0x107 (XSFR)

Bit(s)	Name	Description	R/W	Reset



7: 0	PWMH	TIMER2 占空比高八位 PWM 工作模式时, 该值是 PWM 的占空比设置值; 捕获工作模式时, 当捕获到捕获源之后抓取的计数器高八位的值锁存在此寄存器中。	RW	—

14.4.29. TMR3_CONL

Addr = 0x108 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	SOURCESEL	计数源选择 0x0: 选择 GPIO (CAP PIN) 上升沿作为计数源; 0x1: 选择 GPIO (CAP PIN) 下降沿作为计数源; 0x2: 选择 HIRC 上升沿和下降沿作为计数源; 0x3: 选择 LIRC 上升沿和下降沿作为计数源; 0x4: 选择 XOSC 上升沿和下降沿作为计数源; 0x5: Timer2 计数 CNT 等于周期作为计数源; 0x6: 选择系统时钟作为计数源; 0x7: 选择系统时钟作为计数源;	RW	0x0
4: 2	PSC	TIMER 预分频配置. 0x0: 1/1 0x1: 1/2 0x2: 1/4 0x3: 1/8 0x4: 1/16 0x5: 1/32 0x6: 1/64 0x7: 1/128	RW	0x0
1: 0	TMRMODE	模式选择及计数使能 0x0: 不使能计数 0x1: 计数器模式 0x2: PWM 模式 0x3: 捕获模式	RW	0x0



14.4.30. TMR3_CONH

Addr = 0x109 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	TMRPND	TIMER 计数 pending 位 (写 1 清 pending) 0x0: 没有发生计数等于周期, 或已清零 0x1: 发生过计数等于周期	RW	0x0
6	CAPPND	TIMER 捕获 pending 位 (写 1 清 pending) 0x0: 没有发生捕获事件 0x1: 有发生捕获事件	RW	0x0
5	TMRIE	TIMER 计数中断使能位 0x0: 不使能计数中断 0x1: 使能计数中断, 计数等于周期时允许发生中断	RW	0x0
4	CAPIE	TIMER 捕获中断使能位 0x0: 不使能捕获中断 0x1: 使能捕获中断, 发生捕获事件时允许发生中断	RW	0x0
3: 2	CAPSRC	TIMER 捕获源选择配置. 0x0: 引脚作为捕获源 0x1: 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	CAPEDGESEL	TIMER 捕获引脚的边沿触发设置. 0x0: 上升沿触发捕获 0x1: 下降沿触发捕获 0x2: 双边沿触发捕获 0x3: 双边沿触发捕获	RW	0x0

14.4.31. TMR3_CNTL

Addr = 0x10A (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CNTL	TIMER3 计数器低八位	RW	—



14.4.32. TMR3_CNTH

Addr = 0x10B (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CNTH	TIMER3 计数器高八位	RW	—

14.4.33. TMR3_PRL

Addr = 0x10C (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PRDL	TIMER3 计数周期低八位	WO	—

14.4.34. TMR3_PRH

Addr = 0x10D (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PRDH	TIMER3 计数周期高八位	WO	—

14.4.35. TMR3_PWML

Addr = 0x10E (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PWML	TIMER3 占空比低八位 PWM 工作模式时, 该值是 PWM 的占空比设置值; 捕获工作模式时, 当捕获到捕获源之后抓取的计数器低八位的值锁存在此寄存器中。	RW	—

14.4.36. TMR3_PWMH

Addr = 0x10F (XSFR)



Bit(s)	Name	Description	R/W	Reset
7: 0	PWMH	TIMER3 占空比高八位 PWM 工作模式时，该值是 PWM 的占空比设置值；捕获工作模式时，当捕获到捕获源之后抓取的计数器高八位的值锁存在此寄存器中。	RW	—

14.4.37. WUT_CONL

Addr = 0x152 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	SOURCESEL	时钟源及计数源选择 0x0: 选择 GPIO (CAP PIN) 作为时钟源； 选择上升沿作为计数源； 0x1: 选择 GPIO (CAP PIN) 作为时钟源； 选择下降沿作为计数源； 0x2: 选择 HIRC 作为时钟源； 选择上升沿和下降沿作为计数源； 0x3: 选择 LIRC 作为时钟源； 选择上升沿和下降沿作为计数源； 0x4: 选择 XOSC 作为时钟源； 选择上升沿和下降沿作为计数源； 0x5: timer3 计数 CNT 等于周期作为计数源； 0x6: 选择系统时钟作为计数源； 0x7: 选择系统时钟作为计数源；	RW	0x0
4: 2	PSC	TIMER 预分频配置. 0x0: 1/1 0x1: 1/2 0x2: 1/4 0x3: 1/8 0x4: 1/16 0x5: 1/32 0x6: 1/64	RW	0x0



		0x7: 1/128		
1: 0	TMRMODE	模式选择及计数使能 0x0: 不使能计数 0x1: 计数器模式 0x2: PWM 模式 0x3: 捕获模式	RW	0x0

14.4.38. WUT_CONH

Addr = 0x153 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	TMRPND	TIMER 计数 pending 位 (写 1 清 pending) 0x0: 没有发生计数等于周期, 或已清零 0x1: 发生过计数等于周期	RW	0x0
6	CAPPND	TIMER 捕获 pending 位 (写 1 清 pending) 0x0: 没有发生捕获事件 0x1: 有发生捕获事件	RW	0x0
5	TMRIE	TIMER 计数中断使能位 0x0: 不使能计数中断 0x1: 使能计数中断, 计数等于周期时允许发生中断	RW	0x0
4	CAPIE	TIMER 捕获中断使能位 0x0: 不使能捕获中断 0x1: 使能捕获中断, 发生捕获事件时允许发生中断	RW	0x0
3: 2	CAPSRC	TIMER 捕获源选择配置. 0x0: 引脚作为捕获源 0x1: 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	CAPEDGESEL	TIMER 捕获引脚的边沿触发设置. 0x0: 上升沿触发捕获 0x1: 下降沿触发捕获 0x2: 双边沿触发捕获 0x3: 双边沿触发捕获	RW	0x0



14.4.39. WUT_CNTL

Addr = 0x154 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CNTL	TIMER 计数器低八位	RW	—

14.4.40. WUT_CNTH

Addr = 0x155 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CNTH	TIMER 计数器高八位	RW	—

14.4.41. WUT_PRL

Addr = 0x156 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PRDL	TIMER 计数周期低八位	WO	—

14.4.42. WUT_PRH

Addr = 0x157 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PRDH	TIMER 计数周期高八位	WO	—

14.4.43. WUT_PWML

Addr = 0x158 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PWML	TIMER 占空比低八位 PWM 工作模式在，该值是 PWM 的占空比设置值；捕获	RW	—



		工作模式时，当捕获到捕获源之后抓取的计数器低八位的值锁存在此寄存器中。		

14.4.44. WUT_PWMH

Addr = 0x159 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PWMH	TIMER 占空比高八位 PWM 工作模式时，该值是 PWM 的占空比设置值；捕获工作模式时，当捕获到捕获源之后抓取的计数器高八位的值锁存在此寄存器中。	RW	—

14.4.45. BUZ_CON

Addr = 0x164 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	BUZEN	计数使能 0x0: 不使能 0x1: 使能	RW	0x0
6	COVPEND	计数标志位 0x0: 没有产生标志 0x1: 计数等于 BUZDIV 时产生标志 写 1 清零，写 0 无效。	RW	0x0
5	COVIE	计数中断使能 0x0: 不使能 0x1: 当计数等于 BUZDIV 时允许中断发生	RW	0x0
4	TMRMODE	模式选择及计数使能 0x0: PWM 模式 0x1: 计数器模式	RW	0x0
3	—	保留	RO	0x0
2: 0	PSC	TIMER 预分频配置。 0x0: 1/1	RW	0x0



		0x1: 1/2		
		0x2: 1/4		
		0x3: 1/8		
		0x4: 1/16		
		0x5: 1/32		
		0x6: 1/64		
		0x7: 1/128		

14.4.46. BUZ_DIV

Addr = 0x165 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	BUZDIV	TIMER 计数周期 计数模式下为周期, PWM 模式下当计数值等于 BUZDIV 时 PWM 输出翻转。	RW	—

14.5. 使用流程说明

14.5.1. 计数器/定时器工作模式

- 1) 配置 SOURCESEL;
- 2) 配置计数器、周期, 比较值寄存器;
- 3) 配置 TMRPSC;
- 4) 配置 TMRIE;
- 5) TMRMODE = 0x1;

14.5.2. 捕获工作模式

- 1) 配置 SOURCESEL;
- 2) 配置计数器、周期寄存器;
- 3) 配置 TMRPSC;
- 4) 配置 TMRIE, CAPIE;
- 5) 配置 CAPSEL、EDGESEL;



- 6) 配置 TMRMODE = 0x3;

14.5.3. PWM 工作模式

- 1) 配置 SOURCESEL;
- 2) 配置计数器、周期、比较值寄存器;
- 3) 配置 TMRPSC;
- 4) 配置 TMRIE, ;
- 5) 配置 TMRMODE = 0x2;

15.Normal Timer 模块

15.1. 功能概述

Normal Timer 模块是由一个 16 位的基础功能定时器 Timer4 组成, 其支持多种计数源选择, 支持计数器模式, 捕获模式, 和 PWM 模式等多种工作模式。通过寄存器 TMR_ALLCON[4] 写 1 可以清零 Timer4 计数。

15.1.1. 计数源选择

计数源选择:

- 1) 32K 低速 RC (LIRC)
- 2) 八分频高速 RC (HIRC)
- 3) 二分频 XOSC
- 4) 外部 GPIO 输入/8K RC
- 5) 系统时钟

通过配置 TMR4_CON0 寄存器的 INCSRCSEL 选择不同的计数源, 配置 TMR4_CON0 的 PSC 配置不同的分频系数。

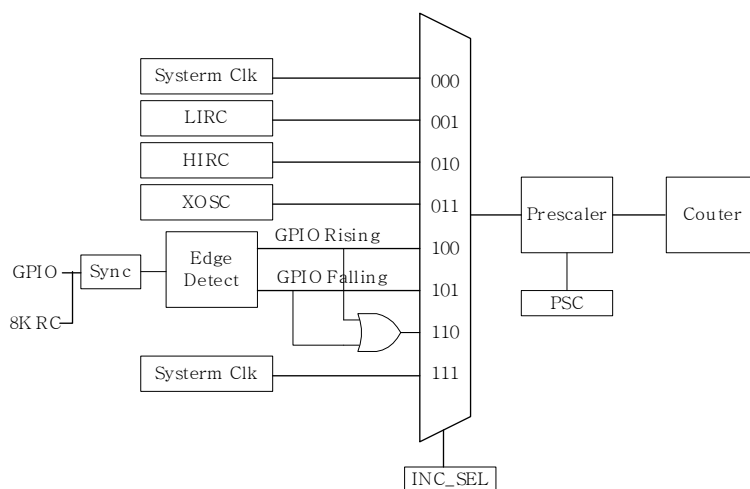


图 15-1 分频寄存器计数源选择逻辑示意图

15.1.2. 输入捕获源

输入捕获，可选 8KRC 时钟、外部 GPIO 触发或者比较器触发。通过配置 SYS_CON2[7]置 1，选择 8K RC 作为输入源。外部 GPIO 触发可以选择单一 Timer4 的 CAP0 PIN 作为触发源，也可以选择 CAP0 PIN 异或 CAP1 PIN 异或 CAP2 PIN 作为触发源。

15.1.3. 输入捕获模式

捕获触发信号首先通过一个选择器选择输入后，经过同步器同步得到 CAP_SRC，然后边沿检测器产生边沿触发信号，最后通过一个选择器选出最终捕获信号 CAP_EV。当捕获信号有效时，自动把当前计数器的值存入 TMR4_CAPx 寄存器，并且产生捕获中断信号。

Timer4 能保存 4 级捕获事件，即可以连续保存 4 次捕获事件发生时计数器的值。每当捕获事件发生时，把当前计数值保存到对应捕获寄存器寄存器（TMR4_CAPx）。

1 级：比较值存入 TMR4_CAP10 和 TMR4_CAP11 寄存器中；

2 级：比较值存入 TMR4_CAP20 和 TMR4_CAP21 寄存器中；

3 级：比较值存入 TMR4_CAP30 和 TMR4_CAP31 寄存器中；

4 级：比较值存入 TMR4_CAP40 和 TMR4_CAP41 寄存器中；

通过配置寄存器 CAPxPOL 可以独立配置每个捕获事件极性，选择上升沿还是下降沿捕获。

通过配置寄存器 CTRRSTx 可以独立配置对应的捕获事件发生时，是否复位计数器的值。

每次捕获事件发生，都会产生对应的捕获标志。通过配置寄存器 TMR4_IE0 可以独立配置每次捕获事件发生是否产生中断。

捕获模式下，支持计数值溢出中断。该模式下，计数周期固定为 16' hffff。当计数值达到 16' hffff，会产生溢出标志 OVFFLAG（TMR4_FLG[4]），通过配置 TMR4_IE0 寄存器可以产生溢出中断。

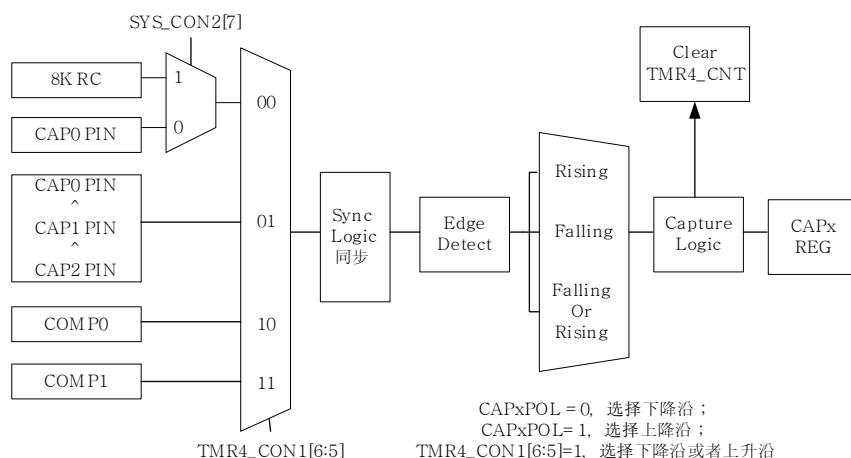


图 15-2 分频寄存器捕获模式选择逻辑示意图

15.1.4. PWM 模式

PWM 工作模式可以产生一个由 TMR4_CAP10 和 TMR4_CAP11 寄存器确定周期，TMR4_CAP20 和 TMR4_CAP21 寄存器确定占空比的信号。

当使用计数模式和 PWM 模式时，TMR4_CAP30 和 TMR4_CAP31 为 TMR4_CAP10 和 TMR4_CAP11 的影子寄存器，TMR4_CAP40 和 TMR4_CAP41 为 TMR4_CAP20 和 TMR4_CAP21 的影子寄存器。当向 TMR4_CAP10 和 TMR4_CAP11 写入值时，该值同时会写入 TMR4_CAP30 和 TMR4_CAP31；当向 TMR4_CAP20 和 TMR4_CAP21 写入值时，该值同时会写入 TMR4_CAP40 和 TMR4_CAP41。而写影子寄存器时，不影响比较值寄存器。

每次计数值等于周期时，自动把 TMR4_CAP30 和 TMR4_CAP31 的值赋值到 TMR4_CAP10 和 TMR4_CAP11，把 TMR4_CAP40 和 TMR4_CAP41 的值赋值到 TMR4_CAP20 和 TMR4_CAP21。

使能 SYS_CON2[5], Timer4 可以和 Simple Timer 中的 Timer2 配合使用完成红外 PWM 调制。将红外编码中的 1 或者 0 的周期和占空比填入 Timer4 对应的寄存器, 将 Timer2 作为载波发生器, 可调制所需求的红外输出波形。

The block diagram illustrates the internal architecture of the AD7175. It features four input sources on the left: SYS_CLK, LIRC输入, HIRC输入, and XOSC输入. Each input passes through a 同步器 (synchronizer) before entering a large 捕获源/计数源选择器 (capture source/counter source selector). The SYS_CLK input also branches off to provide clock signals to the 控制寄存器逻辑 (control register logic), 预分频逻辑 (prescaler logic), and 计数器逻辑 (counter logic). The 控制寄存器逻辑 feeds into the 预分频逻辑, which then feeds into the 计数器逻辑. The 计数器逻辑 outputs to the 中断产生逻辑 (interrupt generation logic) and also provides a signal to the 捕获逻辑 (capture logic). The 捕获源/计数源选择器 outputs to the 捕获逻辑. The 捕获逻辑 feeds into the PWM/捕获寄存器 (PWM/capture register), which then feeds into the PWM产生逻辑 (PWM generation logic). The 中断产生逻辑 outputs to the 中断输出 (interrupt output). The PWM产生逻辑 outputs to the PWM输出 (PWM output).

图 15-3 Normal Timer 模块框图

表 15-1 Timer2 register list

address	Register Name	Description
0xD1 (SFR)	TMR_ALLCON	TIMER ALL Control Register
0xDA (SFR)	TMR4_CON0	TIMER4 Control 0 Register
0xDB (SFR)	TMR4_CON1	TIMER4 Control 1 Register



0xDC (SFR)	TMR4_CON2	TIMER4 Control 2 Register
0xDD (SFR)	TMR4_CON3	TIMER4 Control 3 Register
0xDE (SFR)	TMR4_EN	TIMER4 Enable Register
0xDF (SFR)	TMR4_IE0	TIMER4 Interrupt Enable Register
0xE2 (SFR)	TMR4_CLR0	TIMER4 Clear 0 Register
0xE4 (SFR)	TMR4_CNT0	TIMER4 Counter 0 Register
0xE5 (SFR)	TMR4_CNT1	TIMER4 Counter 1 Register
0xE6 (SFR)	TMR4_CAP10	TIMER4 Capture 10 Register
0xE7 (SFR)	TMR4_CAP11	TIMER4 Capture 11 Register
0xE8 (SFR)	TMR4_CAP20	TIMER4 Capture 20 Register
0xE9 (SFR)	TMR4_CAP21	TIMER4 Capture 21 Register
0xEA (SFR)	TMR4_CAP30	TIMER4 Capture 30 Register
0xEB (SFR)	TMR4_CAP31	TIMER4 Capture 31 Register
0xEC (SFR)	TMR4_CAP40	TIMER4 Capture 40 Register
0xED (SFR)	TMR4_CAP41	TIMER4 Capture 41 Register
0xEE (SFR)	TMR4_FLAG0	TIMER4 Flag Register

15.4. 寄存器详细说明

15.4.1. TMR_ALLCON

Addr = 0xD1 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	–	–	–	0x0
5	WUTWSYNC	Wake Up Timer 计数清零	WO	0x0



		写 1 清零，写 0 无效		
4	TMR4SWSYNC	Timer4 计数清零 写 1 清零，写 0 无效	WO	0x0
3	TMR3SWSYNC	Timer3 计数清零 写 1 清零，写 0 无效	WO	0x0
2	TMR2SWSYNC	Timer2 计数清零 写 1 清零，写 0 无效	WO	0x0
1	TMR1SWSYNC	Timer1 计数清零 写 1 清零，写 0 无效	WO	0x0
0	TMR0SWSYNC	Timer0 计数清零 写 1 清零，写 0 无效	WO	0x0

15.4.2. TMR4_CON0

Addr = 0xDA (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	PSC	计数预分频 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	INCSEL	计数信号选择 0x0: 系统时钟 0x1: LIRC 上升沿 0x2: HIRC 上升沿 0x3: XOSC 上升沿 0x4: GPIO 输入上升沿 0x5: GPIO 输入下降沿 0x6: GPIO 输入边沿（上升下降）	RW	0x0



		0x7: 系统时钟		
1: 0	TMRMODE	模式选择 0x0: 定时器计数模式 0x1: PWM 输出模式 0x2: 捕获模式 0x3: 保留	RW	0x0

15.4.3. TMR4_CON1

Addr = 0xDB (SFR)

Bit(s)	Name	Description	R/W	Reset
7	—	保留		0x0
6: 5	CAPSEL	捕获信号源选择 0x0: GPIO 输入或 LIRC 8 分频 (8K) 0x1: CAP PIN0/1/2 0x2: 比较器 0 0x3: 比较器 1	RW	0x0
4: 0	—	保留, 如写入非 0, 可能引起计数器异常	RW	0x0

15.4.4. TMR4_CON2

Addr = 0xDC (SFR)

Bit(s)	Name	Description	R/W	Reset
7	CAP3POL	3 级捕获信号极性 0x0: 选择上升沿 0x1: 选择下降沿 Note: 要选择双沿请配置 CAPSEL == 0x1;	RW	0x0
6	CAP2POL	2 级捕获信号极性 0x0: 选择上升沿 0x1: 选择下降沿 Note: 要选择双沿请配置 CAPSEL == 0x1;	RW	0x0
5	CAP1POL	1 级捕获信号极性	RW	0x0



		0x0: 选择上升沿 0x1: 选择下降沿 Note: 要选择双沿请配置 CAPSEL == 0x1;		
4	CTRRST4	4 级捕获信号有效时, 计数复位使能 0x0: 不使能 0x1: 使能	RW	0x0
3	CTRRST3	3 级捕获信号有效时, 计数复位使能 0x0: 不使能 0x1: 使能	RW	0x0
2	CTRRST2	2 级捕获信号有效时, 计数复位使能 0x0: 不使能 0x1: 使能	RW	0x0
1	CTRRST1	1 级捕获信号有效时, 计数复位使能 0x0: 不使能 0x1: 使能	RW	0x0
0	—	保留		0x0

15.4.5. TMR4_CON3

Addr = 0xDD (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 4	—	保留	RW	0x0
3: 2	CAPCNT	捕获有效级数 0x0: 1 级 0x1: 2 级 0x2: 3 级 0x3: 4 级	RW	0x0
1	PWMPOL	PWM 输出极性 0x0: 正常输出 0x1: 取反输出	RW	0x0
0	CAP4POL	4 级捕获信号极性 0x0: 选择上升沿 0x1: 选择下降沿	RW	0x0



Note: 要选择双沿请配置 CAPSEL == 0x1;

15.4.6. TMR4_EN

Addr = 0xDE (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 1	—	保留	RW	0x0
0	TMREN	TIMER4 计数器使能 0x0: 不使能 0x1: 使能	RW	0x0

15.4.7. TMR4_IE0

Addr = 0xDF (SFR)

Bit(s)	Name	Description	R/W	Reset
7			RW	0x0
6	CMP1E	计数值等于比较值中断使能 0x0: 不使能 0x1: 使能	RW	0x0
5	PRD1E	计数值等于周期值中断使能 0x0: 不使能 0x1: 使能	RW	0x0
4	OVF1E	捕获模式下计数值溢出中断使能 0x0: 不使能 0x1: 使能	RW	0x0
3	CAP4IE	4 级捕获信号有效中断使能 0x0: 不使能 0x1: 使能	RW	0x0
2	CAP3IE	3 级捕获信号有效中断使能 0x0: 不使能 0x1: 使能	RW	0x0
1	CAP2IE	2 级捕获信号有效中断使能	RW	0x0



		0x0: 不使能 0x1: 使能		
0	CAP1IE	1 级捕获信号有效中断使能 0x0: 不使能 0x1: 使能	RW	0x0

15.4.8. TMR4_CLR0

Addr = 0xE2 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	—	保留	WO	—
6	CMPFLGC	计数值等于比较值标志清零 写 1 清零, 写 0 无效	WO	—
5	PRDFLGC	计数值等于周期值标志清零 写 1 清零, 写 0 无效	WO	—
4	OVFFLGC	捕获模式下计数值溢出标志清零 写 1 清零, 写 0 无效	WO	—
3	CAP4FLGC	4 级捕获信号有效标志清零 写 1 清零, 写 0 无效	WO	—
2	CAP3FLGC	3 级捕获信号有效标志清零 写 1 清零, 写 0 无效	WO	—
1	CAP2FLGC	2 级捕获信号有效标志清零 写 1 清零, 写 0 无效	WO	—
0	CAP1FLGC	1 级捕获信号有效标志清零 写 1 清零, 写 0 无效	WO	—

15.4.9. TMR4_CNT0

Addr = 0xE4 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CNT0	TIMER4 计数器低 8bit	RW	—



15.4.10. TMR4_CNT1

Addr = 0xE5 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CNT1	TIMER4 计数器高 8bit	RW	—

15.4.11. TMR4_CAP10

Addr = 0xE6 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CAP10	TIMER4 计数周期低 8bit 捕获模式下, 1 级捕获值低 8 位	RW	—

15.4.12. TMR4_CAP11

Addr = 0xE7 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CAP11	TIMER4 计数周期高 8bit 捕获模式下, 1 级捕获值高 8 位	RW	—

15.4.13. TMR4_CAP20

Addr = 0xE8 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CAP20	TIMER4 比较值低 8bit 捕获模式下, 2 级捕获值低 8 位	RW	—

15.4.14. TMR4_CAP21

Addr = 0xE9 (SFR)

Bit(s)	Name	Description	R/W	Reset



7: 0	CAP21	TIMER4 比较值高 8bit 捕获模式下, 2 级捕获值高 8 位	RW	—

15.4.15. TMR4_CAP30

Addr = 0xEA (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CAP30	TIMER4 计数周期影子寄存器低 8bit 捕获模式下, 3 级捕获值低 8 位	RW	—

15.4.16. TMR4_CAP31

Addr = 0xEB (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CAP31	TIMER4 计数周期影子寄存器高 8bit 捕获模式下, 3 级捕获值高 8 位	RW	—

15.4.17. TMR4_CAP40

Addr = 0xEC (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CAP40	TIMER4 比较值影子寄存器低 8bit 捕获模式下, 4 级捕获值低 8 位	RW	—

15.4.18. TMR4_CAP41

Addr = 0xED (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CAP41	TIMER4 比较值影子寄存器高 8bit 捕获模式下, 4 级捕获值高 8 位	RW	—



15.4.19. TMR4_FLAG0

Addr = 0xEE (SFR)

Bit(s)	Name	Description	R/W	Reset
7	—	保留	RO	—
6	CMPFLAG	计数值等于比较值标志 0x0: 没有产生标志 0x1: 产生标志	RO	—
5	PRDFLAG	计数值等于周期值标志 0x0: 没有产生标志 0x1: 产生标志	RO	—
4	OVFFLAG	捕获模式下计数值溢出标志 0x0: 没有产生标志 0x1: 产生标志	RO	—
3	CAP4FLAG	4级捕获信号有效标志 0x0: 没有产生标志 0x1: 产生标志	RO	—
2	CAP3FLAG	3级捕获信号有效标志 0x0: 没有产生标志 0x1: 产生标志	RO	—
1	CAP2FLAG	2级捕获信号有效标志 0x0: 没有产生标志 0x1: 产生标志	RO	—
0	CAP1FLAG	1级捕获信号有效标志 0x0: 没有产生标志 0x1: 产生标志	RO	—

15.5. 使用流程说明

15.5.1. 计数器/定时器工作模式

- 1) 配置 INCSEL;
- 2) 配置计数器、周期，比较值寄存器;



- 3) 配置 PSC;
- 4) 配置 TMR1E;
- 5) TMRMODE = 0x0;
- 6) 使能;

15.5.2. 捕获工作模式

- 1) 配置 INCSEL;
- 2) 配置计数器、周期, 比较值寄存器;
- 3) 配置 PSC;
- 4) 配置 TMR1E, CAP1E, OV1E;
- 5) 配置 CAPSEL、CAPPOL、CAPNUM;
- 6) 配置 TMRMODE = 0x2;
- 7) 使能;

15.5.3. PWM 工作模式

- 1) 配置 INCSEL;
- 2) 配置计数器、周期、比较值寄存器;
- 3) 配置 PSC;
- 4) 配置 TMR1E;
- 5) 配置 TMRMODE = 0x1;
- 6) 使能;

15.5.4. 红外工作模式

- 1) 配置 INCSEL;
- 2) 配置计数器、周期、比较值寄存器;
- 3) 配置 PSC;
- 4) 配置 Simple Timer 模块中 Timer2 的周期、比较值, 配置 PWM 模式, 作为载波;
- 5) 配置 SYSCON2[5]使能红外功能;
- 6) 配置 TMRMODE = 0x1;
- 7) 使能两个 Timer;



16. Super timer 模块（增强型 PWM 模块）

16.1. 功能概述

增强型 STMR 模块支持 6 路 PWM 发生器，可以配置成相互独立的 6 路 PWM 输出（STMR0-STMR5），也可以配置成 3 对分别带有编程死区发生器的互补 PWM（STMR0-STMR1，STMR2-STMR3，STMR4-STMR5）。

每一路 PWM 拥有 1 个 8 位预分频器。每一路 PWM 输出有独立的 16 位计数器进行控制，另外 16 位的比较器用以调节占空比。6 路 PWM 发生器提供 30 个中断标志，相关 PWM 通道的周期或占空比与计数器相符，将产生中断标志，每一路 PWM 有单独的使能位。

每路 PWM 可配置成单次模式（产生一个 PWM 信号周期）或者循环模式（连续输出 PWM 波形）。

增强型 STMR 模块具有如下特性：

- 6 路独立 PWM 输出：STMR0-STMR5；
- 3 组互补 PWM 对输出：STMR0-STMR1，STMR2-STMR3，STMR4-STMR5；
- 可插入可编程死区时间，8 种死区模式可选；
- 3 组同步 PWM 对输出：STMR0-STMR1，STMR2-STMR3，STMR4-STMR5；
- 支持群组控制，STMR0，STMR2，STMR4 输出同步，STMR1，STMR3，STMR5 输出同步；
- 单次模式或者自动装载模式；
- 支持边沿对齐，中心对齐 2 种模式；中心对齐模式支持对称计数和非对称计数；
- 每路 PWM 可独立选择大于比较值或者小于比较值输出；
- 硬件刹车保护（外部 FB 触发，比较器通道 0 或者 1，ADC，支持软件触发）；

请先开启互补、同步、群组模式后，再进行周期比较值配置。

16.1.1. 基本动作

基本波形模式

增强型 STMR 模块是由计数器模块、输出比较单元、波形发生器、故障检测和输出控制器组成。

计数器：

时钟输入增强型 STMR 模块，通过预分频器（STMRn_PSC）配置计数器的计数频率，周期寄存器（STMRn_PRH，STMRn_PRL）设置其计数周期。为了防止在 PWM 运行的过程中随意修改周期设置，采用缓冲寄存器（Period Buffer）对周期进行缓存。如果 PWM 设置为连续运

行模式（STMR_CNTMD 寄存器中的 STMRnCNTM=1），在每个 PWM 的自动重载点会自动将周期寄存器的值加载到缓冲寄存器（Period Buffer）当中。

PWM 计数器有两种计数模式：边沿对齐向下计数（Down count mode）和中心对齐计数（UP-Down count mode）。

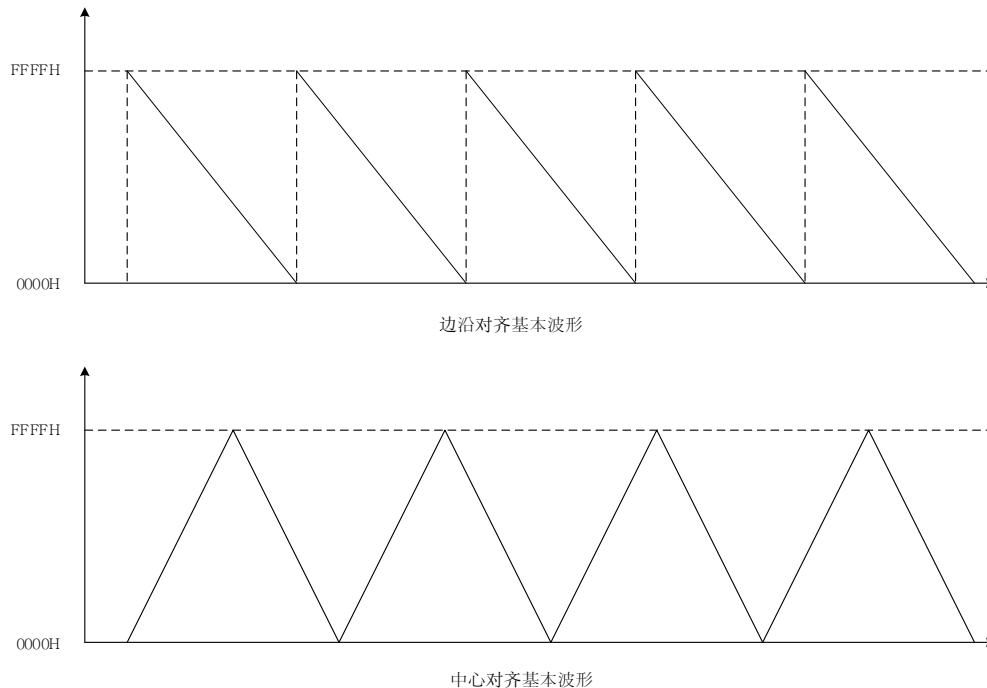


图 16-1 PWM 计数模式波形图

输出比较单元：

输出比较单元是由占空比寄存器（STMRn_CMPAH，STMRn_CMPAL，STMRn_CMPBH，STMRn_CMPBL）组成，用于设置 PWM 占空比。STMRn_CMPBH，STMRn_CMPBL 仅在上下计数模式中有效。同样地，为了防止在运行的过程中随意修改 PWM 的占空比设置，采用缓冲寄存器（Duty Buffer）和 PWM 计数器进行比较，以进行输出电平的翻转。如果设置为连续运行模式，在每个自动装载点会自动将占空比寄存器的值加载到缓冲寄存器（Duty Buffer）当中。

装载点：

边沿模式，CNT=0 时自动装载；

上下计数模式可选装载点：STMR_CNTMD[7]=1 选择周期点时加载；STMR_CNTMD[7]=0 选择 0 点自动装载；

STMR_CNTEN=0 时，写占空比和周期寄存器会直接更新至缓冲寄存器中。



波形发生单元:

波形发生器是由死区控制单元和输出选择控制单元组成。针对带死区的互补输出, STMR01_DT/ STMR23_DT/ STMR45_DT 用于设置死区时间; 再结合输出选择控制单元对比较值 A 点或 B 点的输出值进行控制。

故障检测(刹车功能):

故障检测模块内嵌在增强型 STMR 中, 配置为输入故障侦测, 是为了保护系统防止器件损坏。一旦检测到有效的故障信号输入, 则强制关断 PWM 的输出。为了适应不同的驱动要求, 关断的电平是可以进行选择配置。

掩码输出:

针对类似方波电机控制这种特殊的应用场合, 掩码输出显得尤为重要。STMR 每个通道都有单独的掩码控制位和掩码数据位, 通过掩码控制寄存器 STMR_PWMMSKEN 和掩码数据寄存器 STMR_PWMMSKD 设置。

当掩码输出禁止 STMR_PWMMSKEN =0 时, STMRn 输出正常的 PWM 波形;

当掩码输出使能 STMR_PWMMSKEN =1 时, STMRn 输出掩码寄存器 STMR_PWMMSKD 的数据。

输出控制器:

输出控制器, 用于对 PWM 的输出状态进行控制。PWM 输出使能控制寄存器 STMR_PWMEN 用于设置各通道的输出使能, STMR_PWM BEN 用于设置 B 点输出使能。发生故障需要强制关断 PWM 时, MCU 可根据刹车数据寄存器 STMR_BRKDAT 中的设置输出相应的电平以适应不同外设的需求。

16.1.2. 增强型 STMR 操作

16.1.2.1. 加载更新模式

计数器加载模式有两种: 单次模式与自动加载模式。单次模式下, 周期和占空比数据在计数使能前自动加载; 自动加载模式下, 周期和占空比相关数据在自动加载点自动加载。

由于 STMR 存在双缓存结构, 在运行的过程中, 改变相关运行寄存器: STMRn_PRL/ STMRn_PRH/STMRn_CMPAL/STMRn_CMPAH/ STMRn_CMPBL/ STMRn_CMPBH/的值, PWM 输出波形不会立即改变, 只有在装载点时这些寄存器的值才会加载到相应的缓存中。这样的结构在改变周期占空比数据后, 不会立即改变当前 PWM 周期的输出波形, PWM 波形在下个周期才会做出相应的变化。即任何 PWM 相关数据的改变不会影响当前一个完整 PWM 周期。

在高速的应用中，有可能会出现加载点已经到来，但写入运行寄存器的操作还未完成的情况。此时不期望出现部分运行数据已经加载，另外一部分运行数据没有加载的情况。针对该高速应用情况，PWM 模块提供了加载使能位。

当改变相关运行寄存器后，需要将加载使能位 STMR_LOADEN 置 1，周期和占空比加载完毕后 STMR_LOADEN 位自动清零。即可以通过读取该位来判断是否将相关寄存器的值加载到实际电路中。如果 STMR_LOADEN=0，则表示已经加载，将影响正在输出的 PWM 波形；如果 STMR_LOADEN=1，则表示还未加载，当前的 PWM 波形还未发生变化，将在下一个加载点加载之前改变的寄存器的值。如果再次改变相关运行寄存器的值，也需重新将 STMR_LOADEN 置 1。

注：当 STMR_LOADEN=1 时，对周期和占空比寄存器内容的更改，可能引发无法预测的结果。建议先更改周期和占空比寄存器内容，再将加载使能位 STMR_LOADEN 置 1，最后等待加载完成。

16.1.2.2. 单次计数模式

单次计数模式是 STMR 计数器只工作一个 PWM 周期，而后 PWM 计数器停止运行的模式。单次计数模式完成，STMR 计数使能控制位硬件清 0 (STMR_CNTEN 寄存器中的 STMRnCNTEN=0)，若再次开启单次模式需使能 STMR 计数使能控制位 (STMRnCNTEN=1)。通过 STMR 计数器模式控制寄存器 STMR_CNTMD 中的 STMRnCNTM=0 可选择单次计数模式。

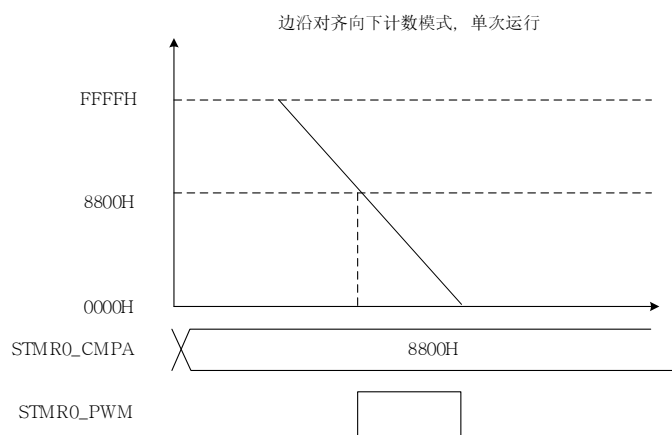


图 16-2 PWM 单次计数模式波形图-中心对齐模式

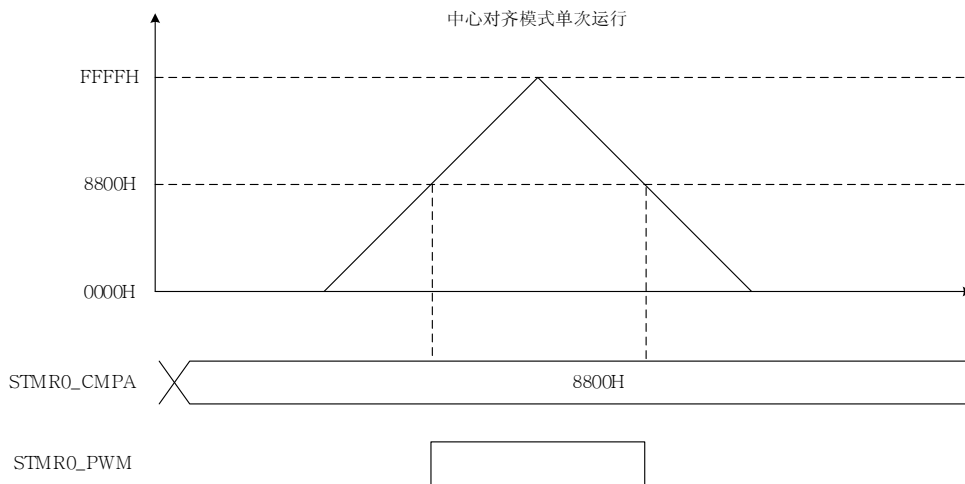


图 16-3 PWM 单次计数模式波形图-边沿对齐向下数模式

16.1.2.3. 边沿对齐模式

边沿对齐模式下，STMR 计数器向下计数模式（Down count）。16 位 STMRn 计数器的初始值为周期值，以此开始向下计数直至计数值变为 0，此时计数器自动将周期寄存器的值加载到计数寄存器中，继而开始下一个 PWM 周期的计数。

通过设置 STMR_PWMVALA 寄存器来控制 PWM 输出电平高低。STMRnPWMVA=0，表示当计数寄存器的值小于占空比寄存器的值时输出高电平，大于等于时输出低电平；STMRnPWMVA=1，表示当计数寄存器的值大于占空比寄存器的值时输出高电平，小于等于时输出低电平。

16.1.2.4. 中心对齐模式

对称计数

中心对齐对称计数模式下，STMRn 计数器采用上下计数模式（Up-Down count），16 位计数器寄存器从 0 开始向上计数，当计数寄存器的值等于周期值后又自动开始向下计数直到 0，后续的 PWM 周期重复这样的计数操作。（系统复位重新使能计数器时，计数先减至 0，然后开始正常的上下计数模式。如果中途停止计数，再使能计数器，计数将根据停止时的方向向上计数到周期或者向下计数到 0，再开始正常的上下计数。）

无论是向上计数还是向下计数，当计数寄存器的值与占空比寄存器 STMRn_CMPA 的值相等时，STMRn 的 PWM 输出电平就会发生翻转。翻转前 PWM 输出值由相应的 STMR_PWMVALA 寄存器设置。

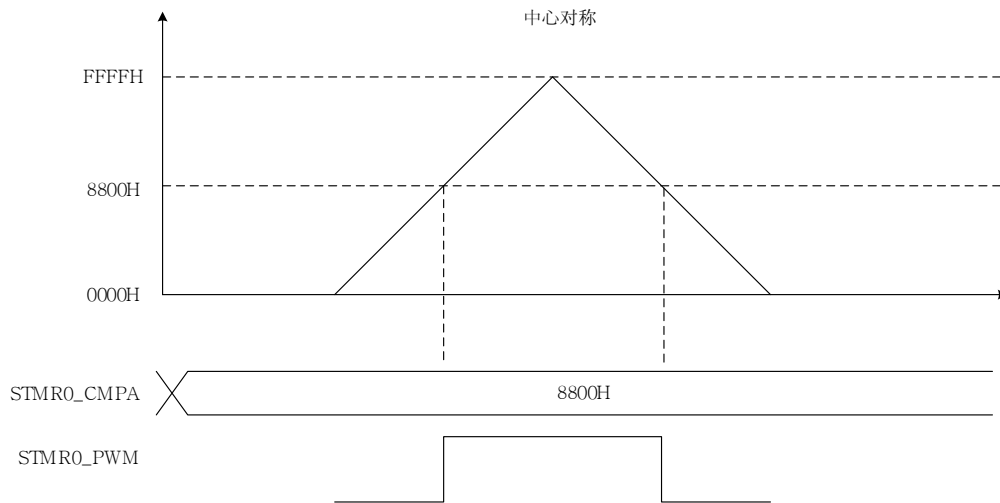


图 16-4 PWM 中心对齐模式波形图

非对称计数

中心对齐非对称计数模式，是电机控制里非常重要的一个特性。PWM 计数器的工作方式依然采用上下计数模式（Up-Down count）。

在这种模式下，有两个 16 位比较寄存器：STMRn_CMPA，STMRn_CMPB。STMRn 计数器寄存器从 0 开始向上计数，当计数寄存器的值等于 STMRn_CMPA 时，STMRn 的 PWM 输出电平翻转，之后计数寄存器继续向上计数至周期，然后开始向下计数，在向下计数的过程中当计数寄存器的值等于 STMRn_CMPB 时，STMRn 的 PWM 输出电平发生翻转，之后继续向下计数至 0。开启非对称模式需要将 STMR_PWMBEN 寄存器中的 STMRnPWMBEN 置 1。

中心对齐非对称模式的时序图如下所示：

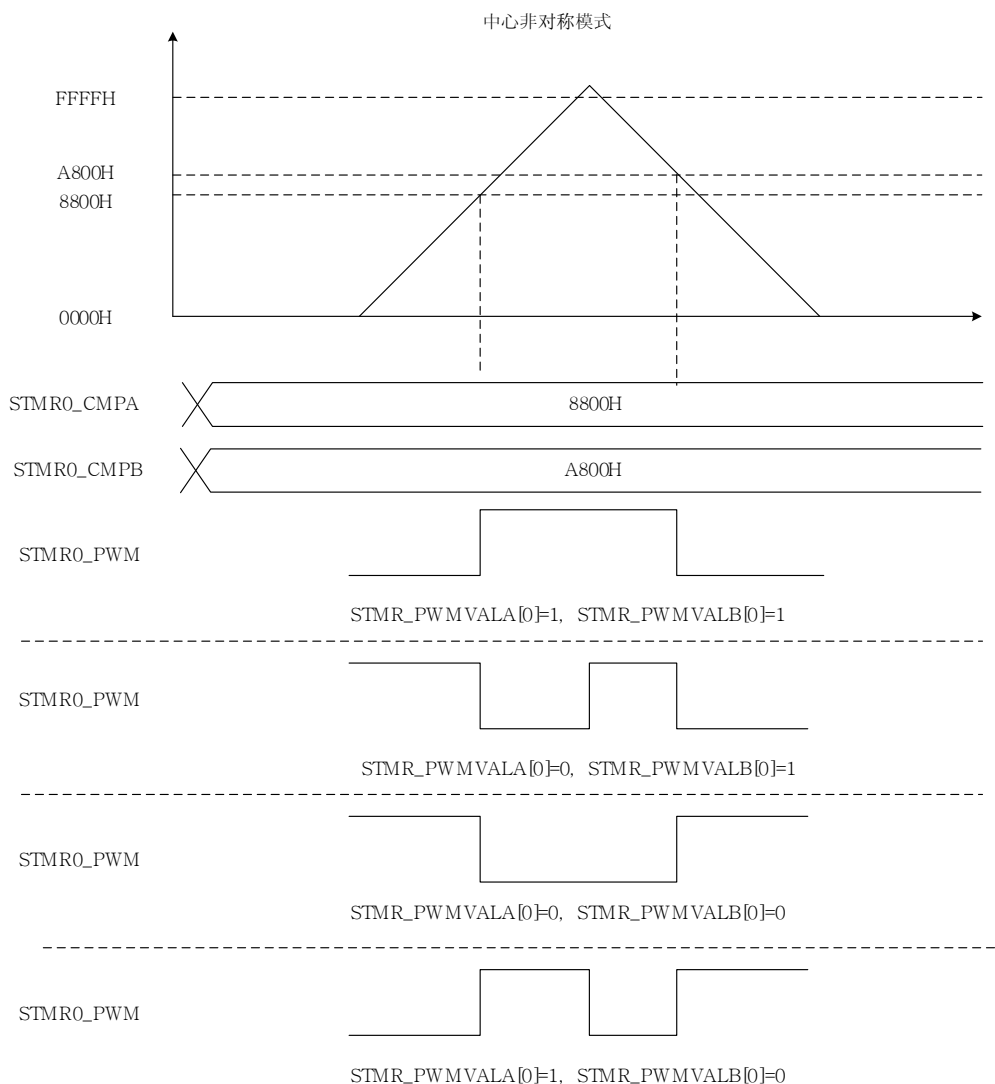


图 16-5 PWM 非中心对称模式

16.1.2.5. 带死区的互补模式

在实际的电机控制应用中，用于驱动逆变桥的 PWM 信号需要具备互补输出的模式，即上桥臂的驱动信号正好和下桥臂的驱动信号反相。

在增强型 STMR 模块中，6 通道 PWM 可设置为 3 对互补信号：STMR0 和 STMR1，STMR2 和 STMR3，STMR4 和 STMR5。STMR1，STMR3，STMR5 的周期与占空比分别由 STMR0，STMR2，STMR4 相关寄存器决定。在设置 STMR0，STMR2，STMR4 的周期和占空比前，需要先配置 STMR_CON0 寄存器，选择互补模式。

在电机控制应用当中，理想的 PWM 信号是在同一时刻发生电平的翻转，由于 MOS 管的开



通和关断存在着延时,这样就容易造成电源对地直通,从而损坏功率管。为了避免这种现象,带死区时间的 PWM 就显得尤为重要。在互补模式下,每组互补 PWM 均支持插入死区时间,插入的死区时间如下:

STMR0/1 死区时间: $(\text{STMR01\_DT}+1) * \text{TSTMR0}$

STMR2/3 死区时间: $(\text{STMR23\_DT}+1) * \text{TSTMR2}$

STMR4/5 死区时间: $(\text{STMR45\_DT}+1) * \text{TSTMR4}$

TSTMR0, TSTMR2, TSTMR4 分别为 STMR0, STMR2, STMR4 的时钟源周期。死区时间为 8 位。

当 PWM 通过计数 CNT 和占空比比较,输出的波形通过死区时,可以通过配置 STMR_DTCON 和 STMR_DTEN 寄存器中的死区模式选择,以及 STMR_EDGE_SEL 和 STMR_DT_DAT 改变最终输出到 GPIO 上的 PWM 波形。

死区模式对于波形的影响,以 STMR01 互补模式为例,STMR01_DT=0xFF:

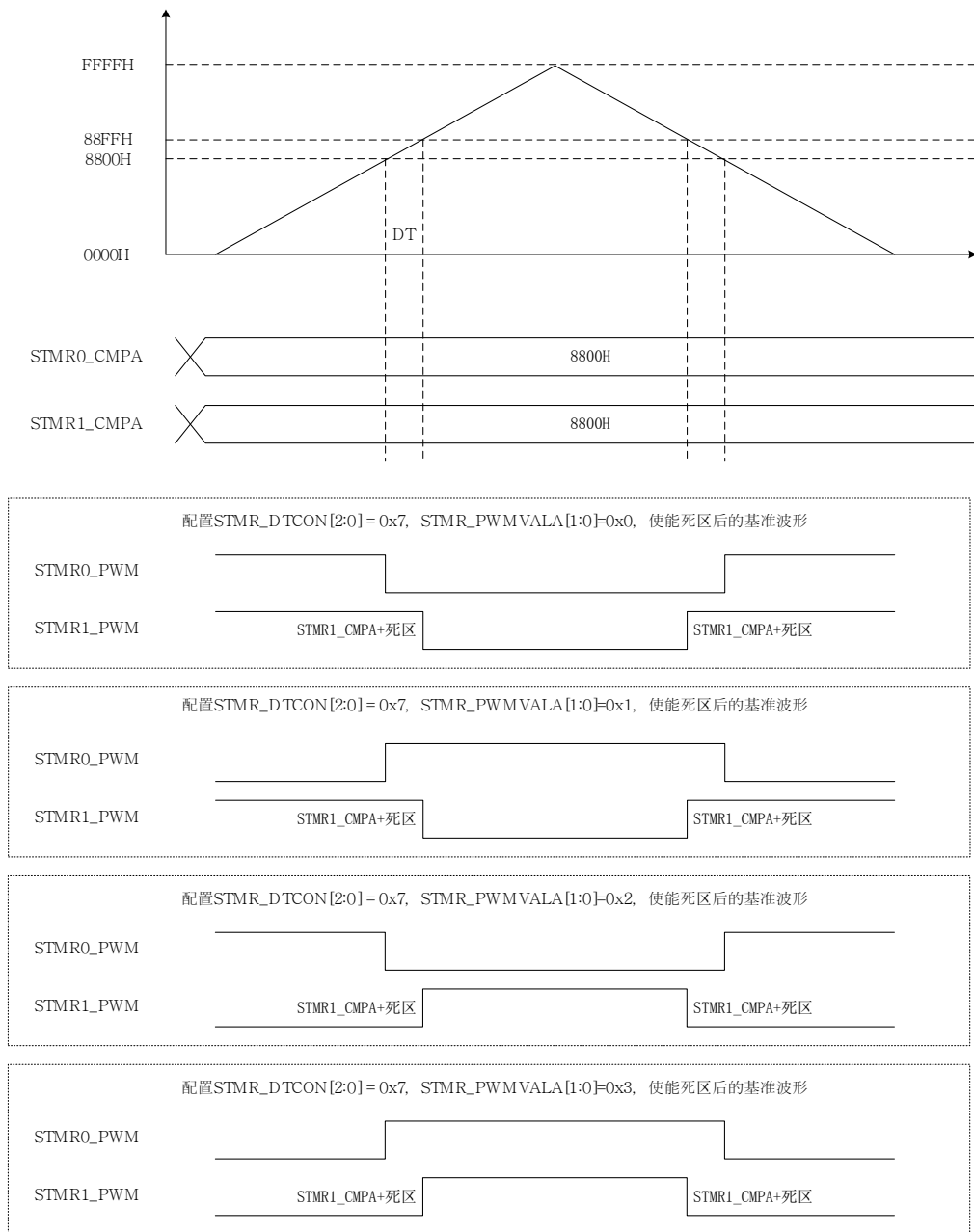


图 16-6 PWM 死区模式波形图

STM0_EDGE_SEL 和 STM0_DT_DAT 改变对最终输出到 GPIO 上的 PWM 波形影响：
 改变 STM0_EDGE_SEL[1] 和 STM0_DT_DAT[0] 的值影响 STM0_PWM 的输出波形；
 改变 STM0_EDGE_SEL[0] 和 STM0_DT_DAT[1] 的值影响 STM1_PWM 的输出波形。
 以 STM001 互补，死区模式选择 7，STM0_PWMVALA[1:0]=0x2 为例：

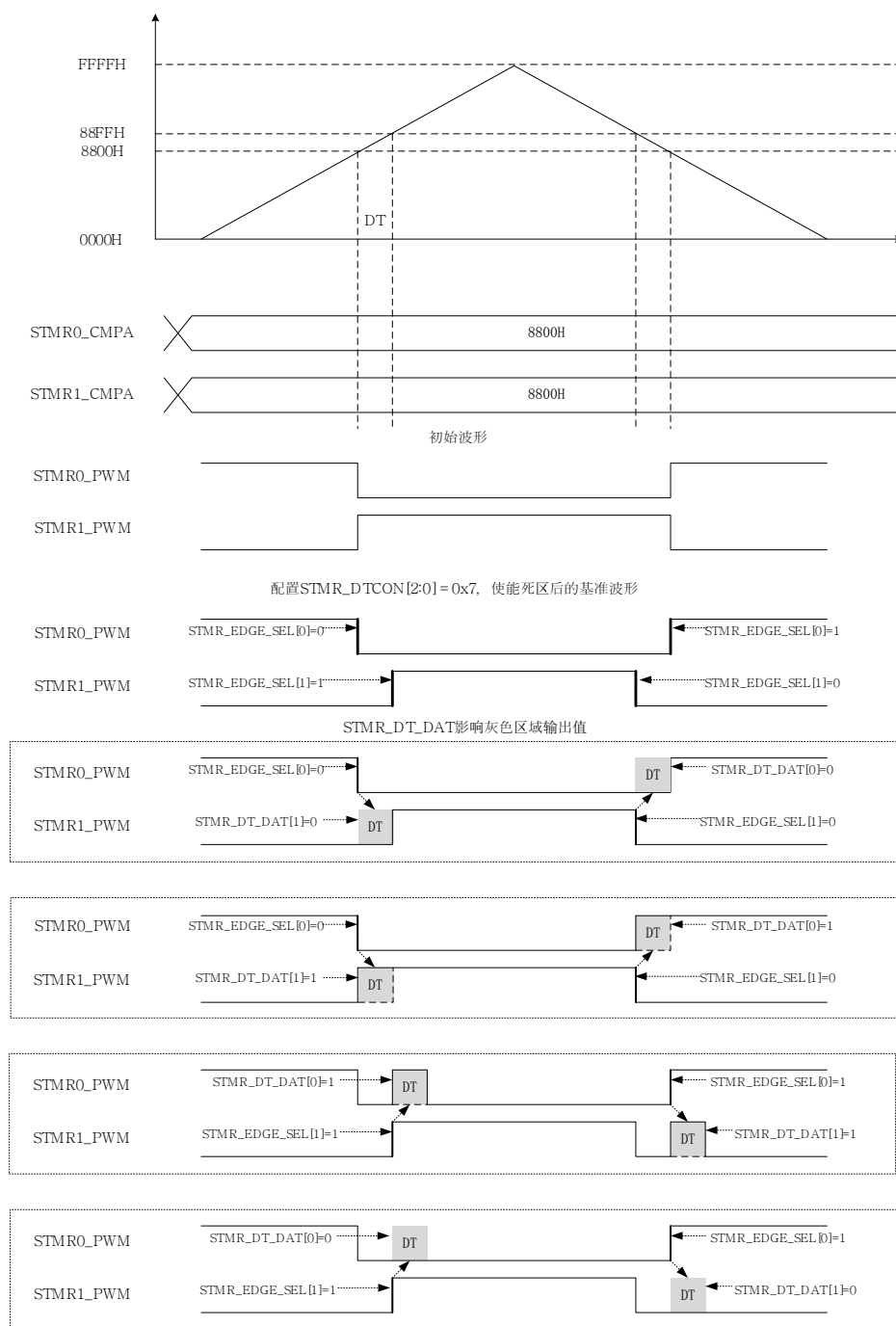


图 16-7 死区触发边缘选择与死区输出值的关系图

中心对齐与边沿对齐均支持互补模式。中心对齐支持 8 种死区模式可选。这 8 中模式的主要区别在于死区存在于互补波形的哪一个输出波形中。

以 STMR01 互补模式示意：

STMR_CON0[1: 0]=0x2, STMR_CNTMD[1: 0]=0x3, STMR_CNTTYPE[1: 0]=0x3,



STMR_PWMEN[1: 0]=0x3, STMR01_DT=0xFF, STMR_DTEN[0]=0x1

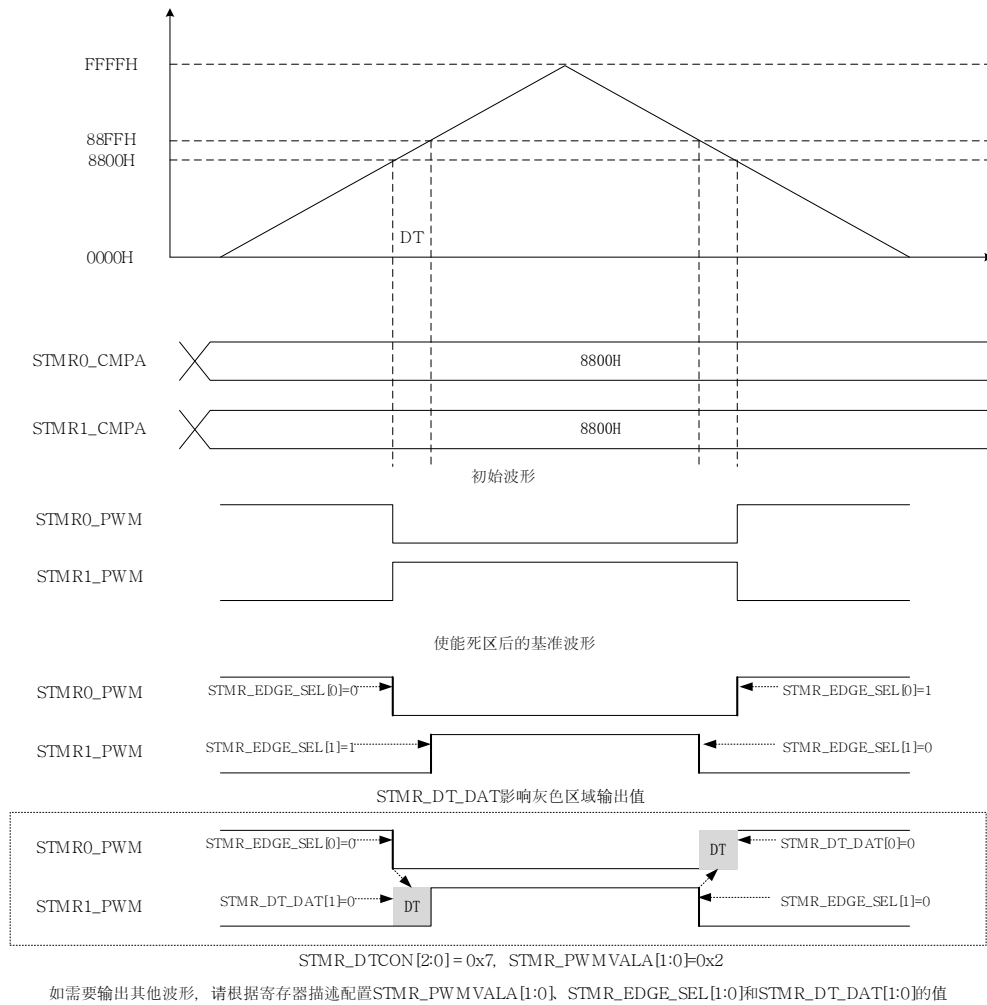


图 16-8 PWM 死区模式 7 互补波形图

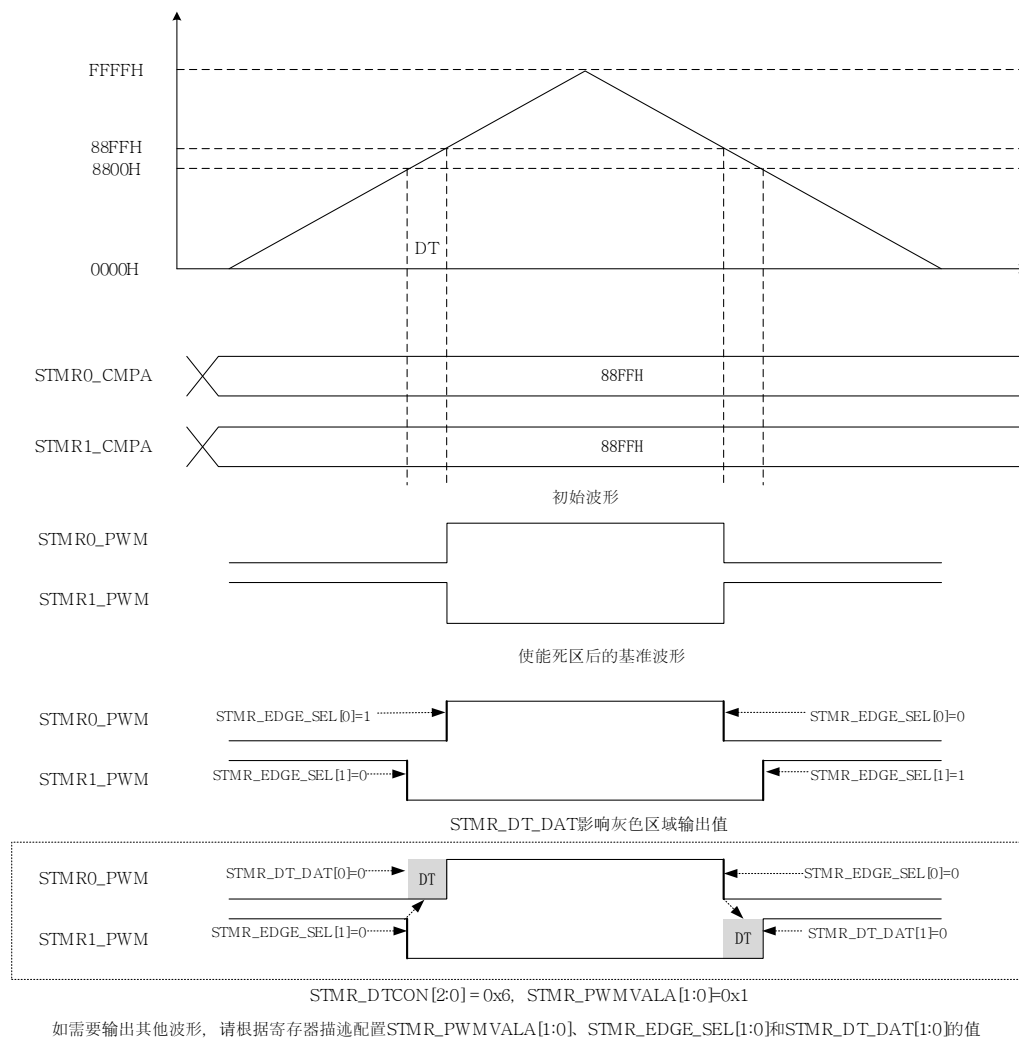


图 16-9 死区模式 6 互补波形图

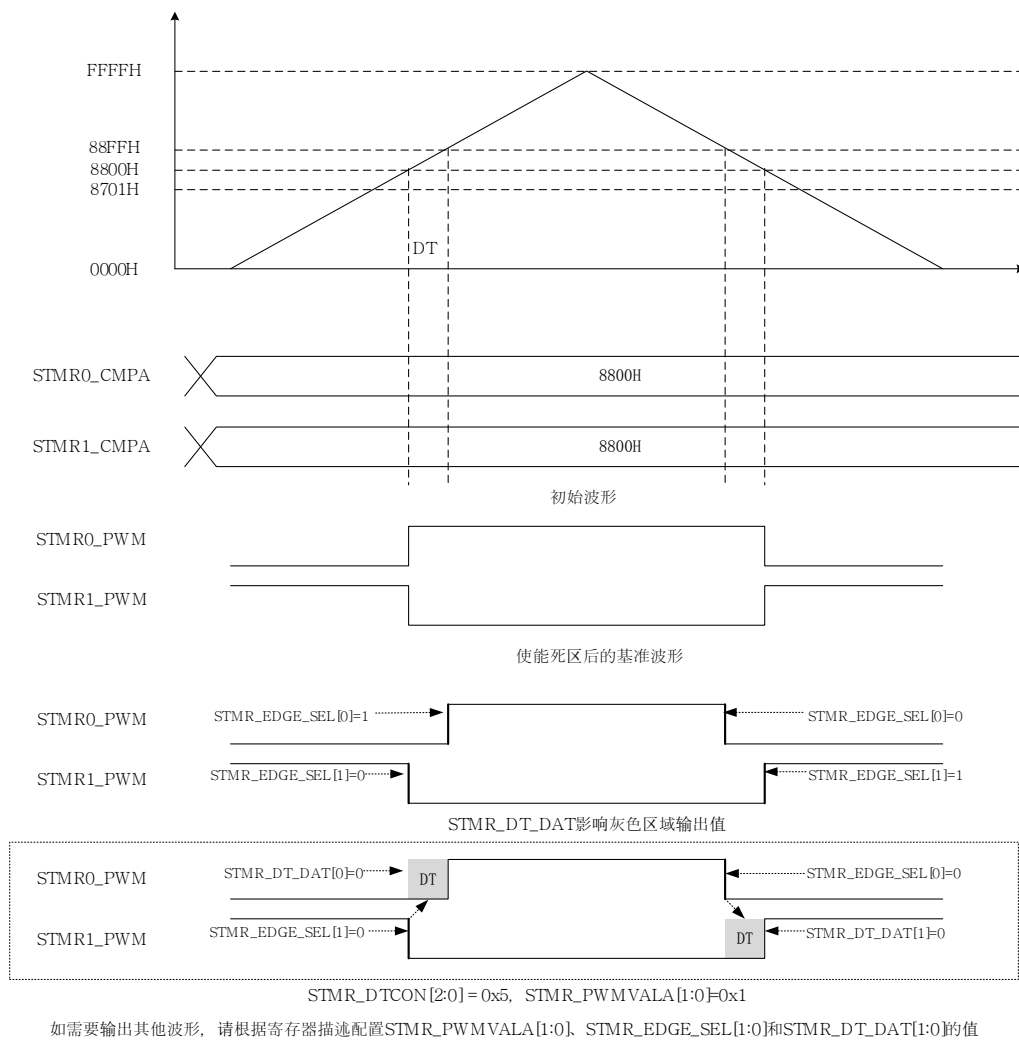


图 16-10 死区模式 5 互补波形图



深圳富芯电子科技有限公司 <http://www.fxmcu.com/>

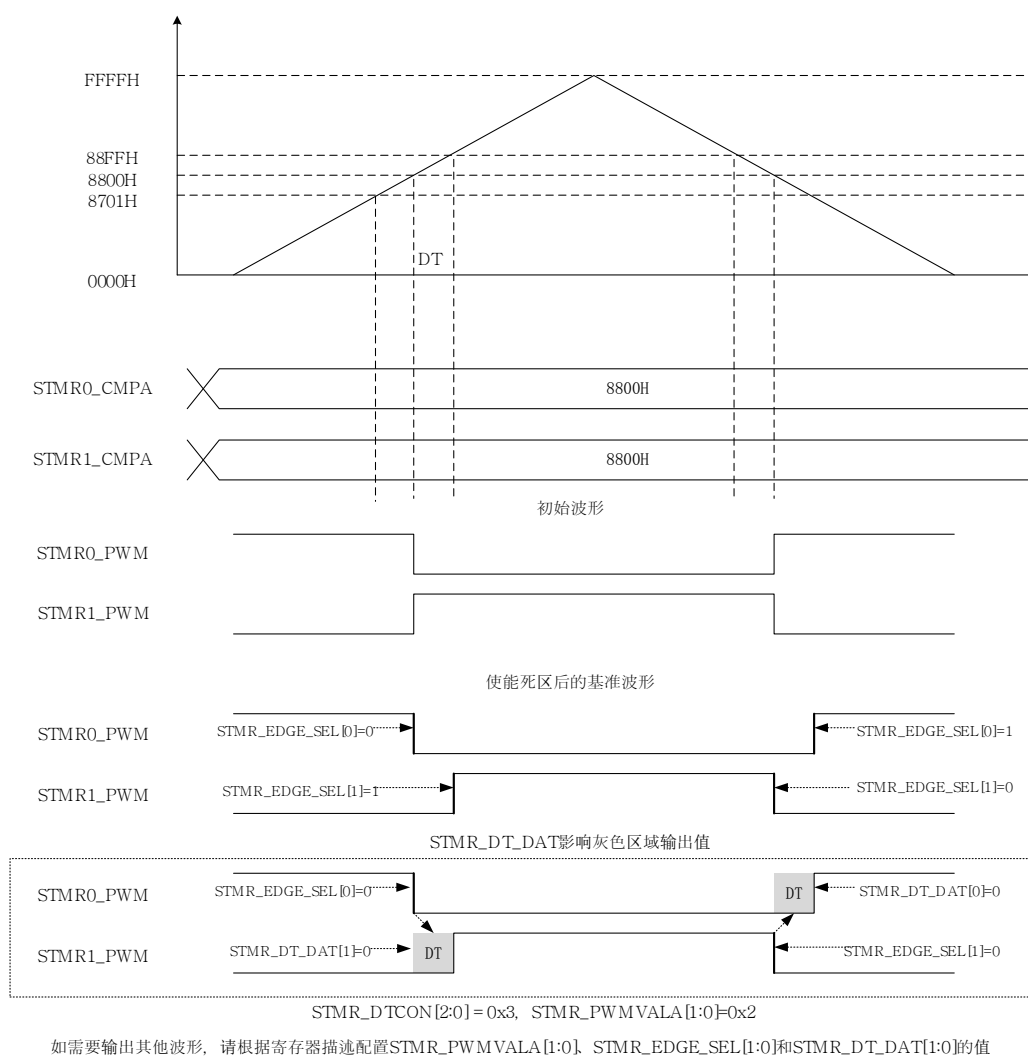


图 16-12 死区模式 3 互补波形图

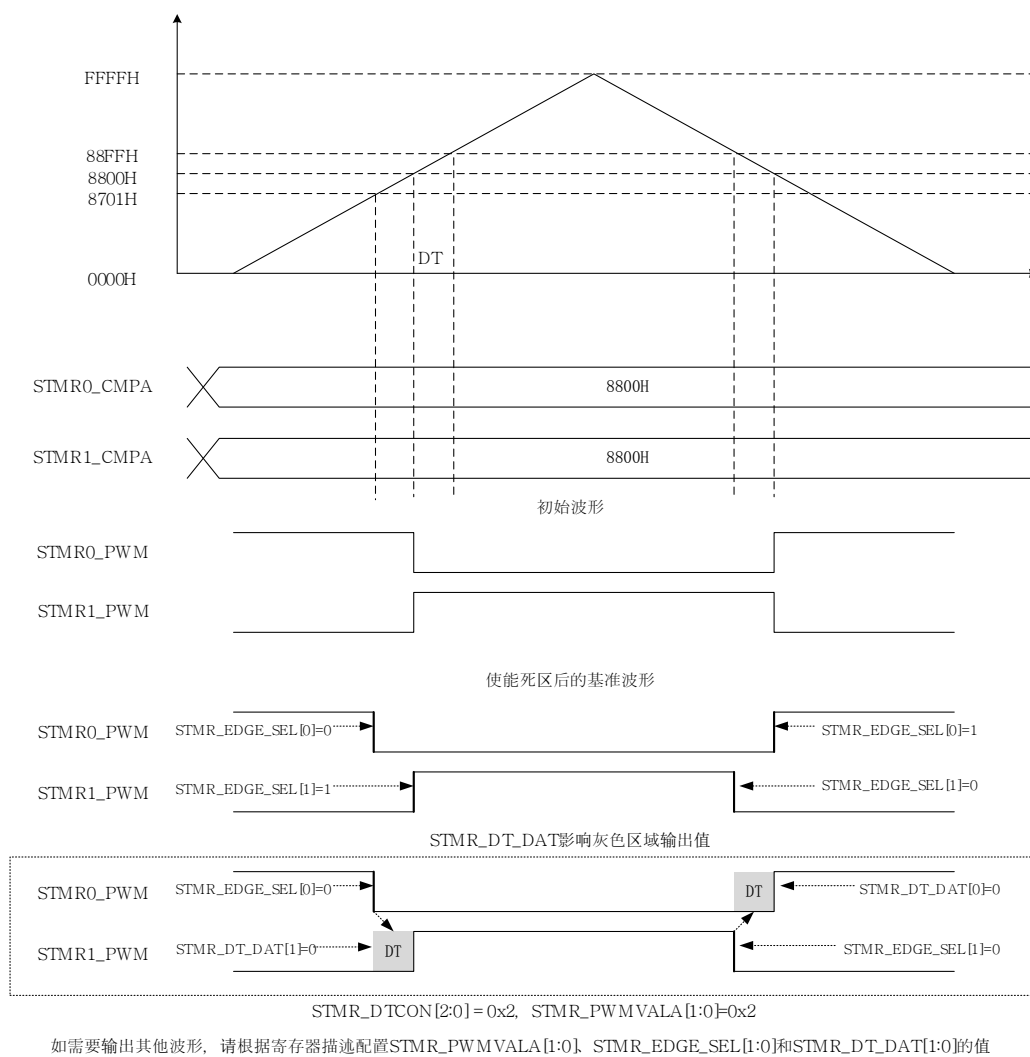


图 16-13 死区模式 2 互补波形图

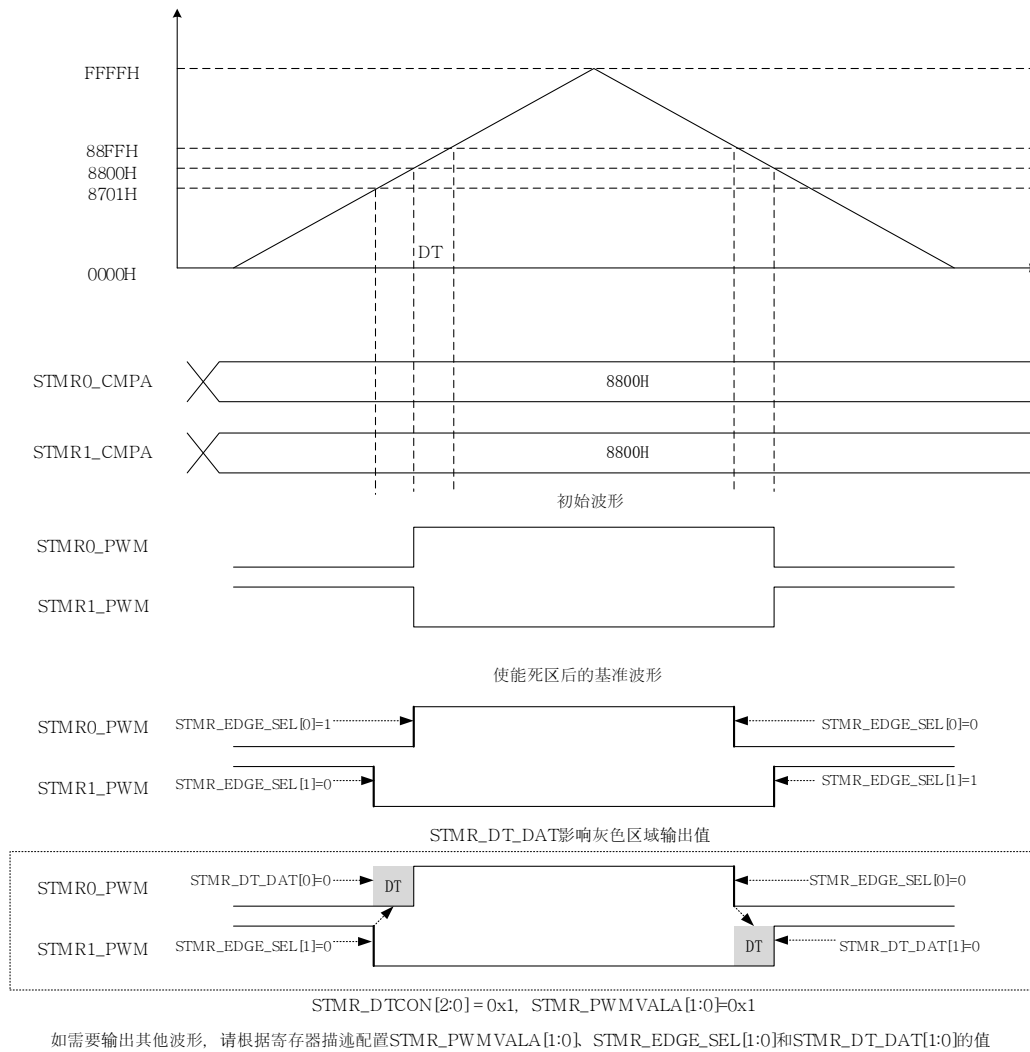


图 16-14 死区模式 1 互补波形图

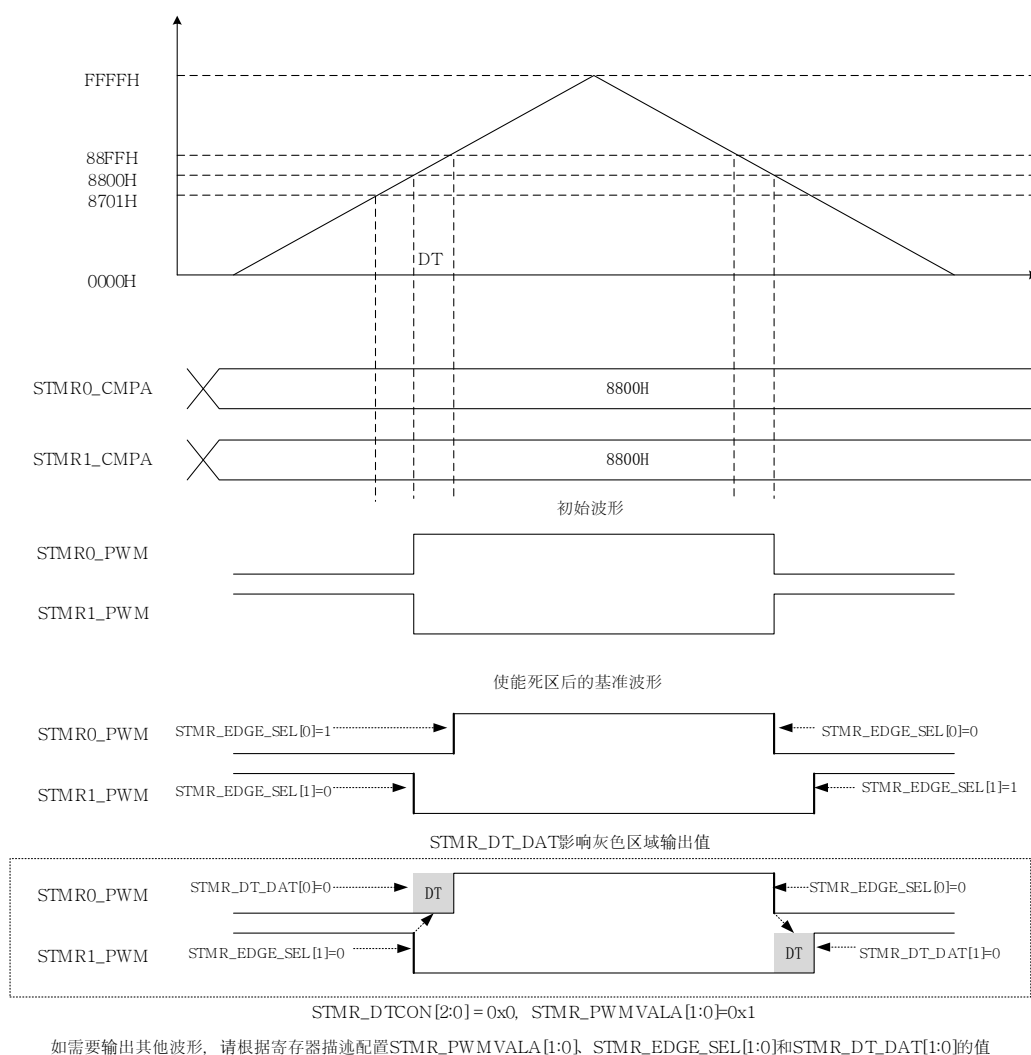


图 16-15 死区模式 0 互补波形图

在配置死区时, 请先根据 STMR_PWMVALA 和 STMR_PWMVALB 以及 STMR_DTCN 的值画出波形。然后以此波形为基础, 配置 STMR_EDGE_SEL 和 STMR_DT_DAT 的值, 以得到最终的需求波形。

在配置占空比的时候, 会出现占空比值减死区小于零的情况或者占空比加死区大于周期的情况。当占空比减死区小于零时, 得到的真实占空比为零, 示例图如下:

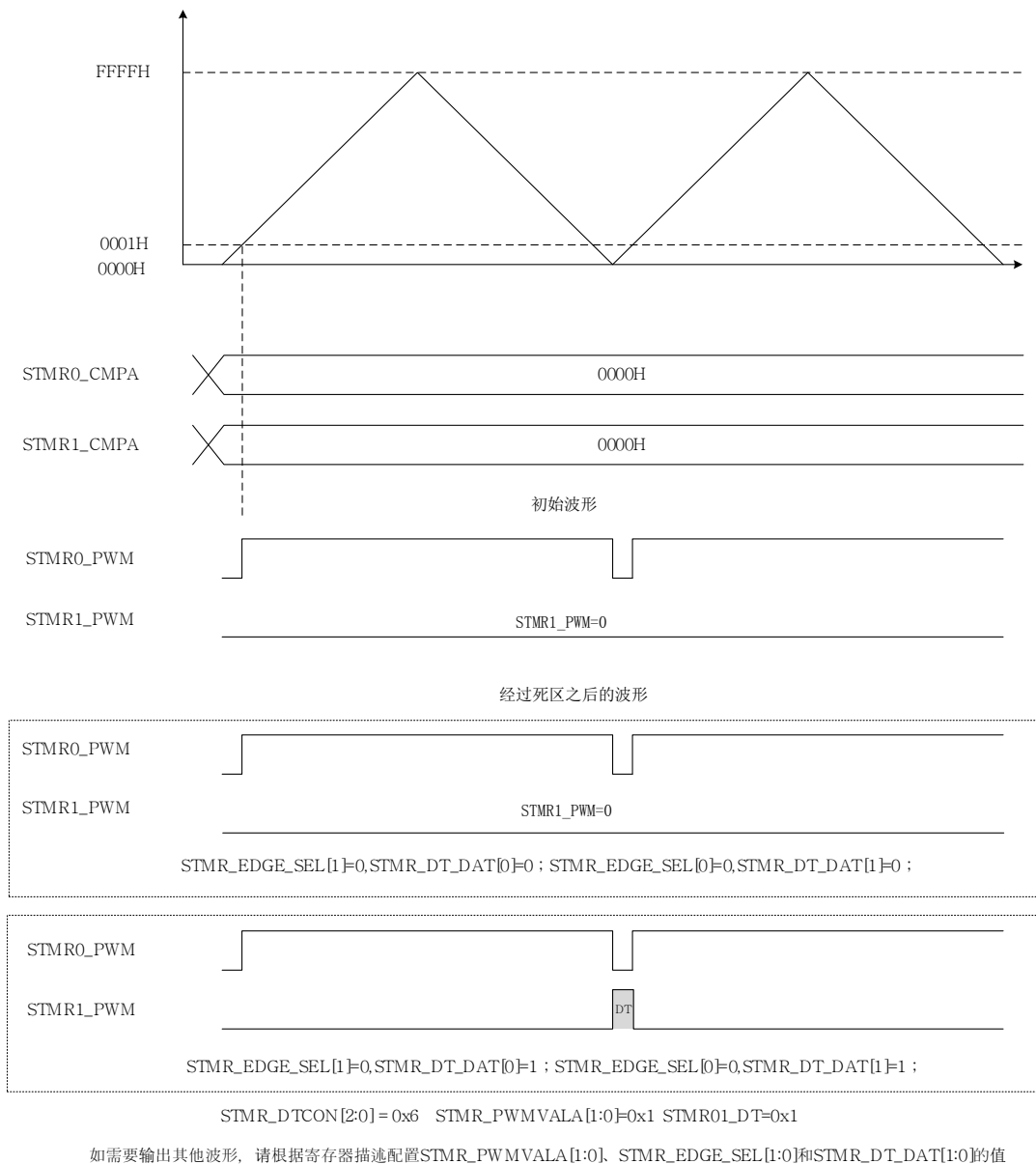
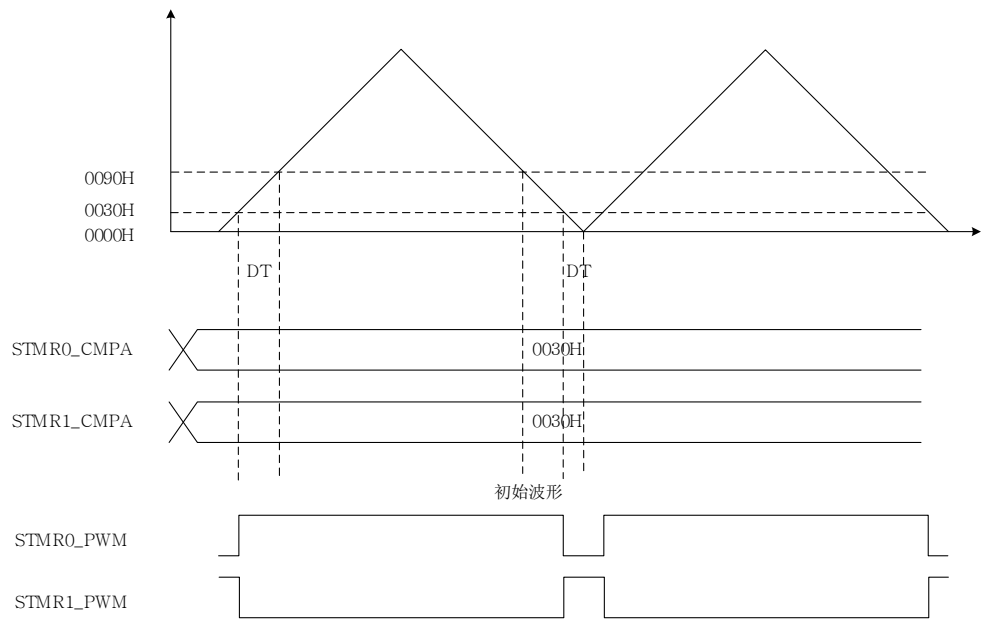
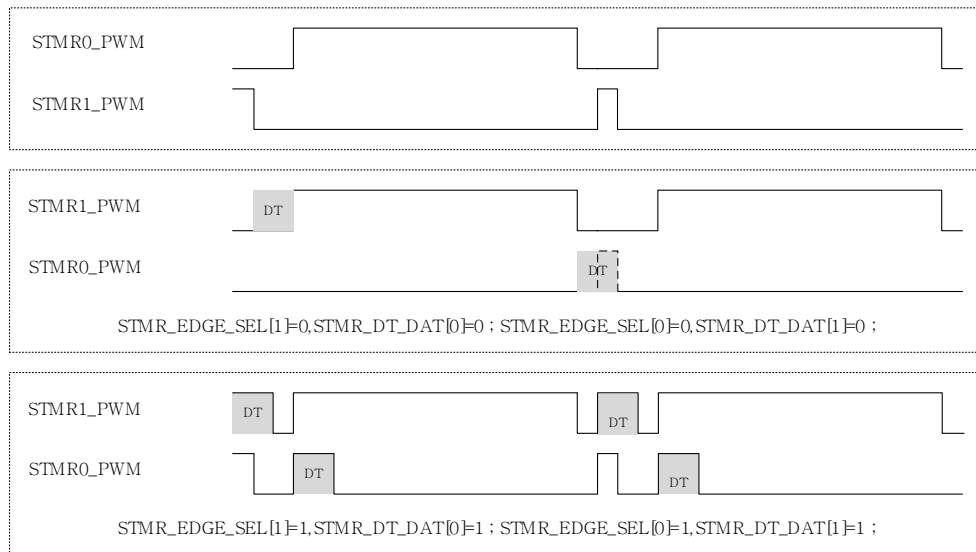


图 16-16 死区模式 6 占空比值减死区小于零时互补波形图 1



经过死区之后的基准波形



STMR_DTCON[2:0]=0x0 STMR_PWMVALA[1:0]=0x1 STMR01_DT=0x60

如需要输出其他波形，请根据寄存器描述配置STMR_PWMVALA[1:0]、STMR_EDGE_SEL[1:0]和STMR_DT_DAT[1:0]的值

图 16-17 死区模式 0 占空比值减死区小于零时互补波形图 2

当占空比加死区大于周期时，实际占空比可以大于周期，示例图如下：

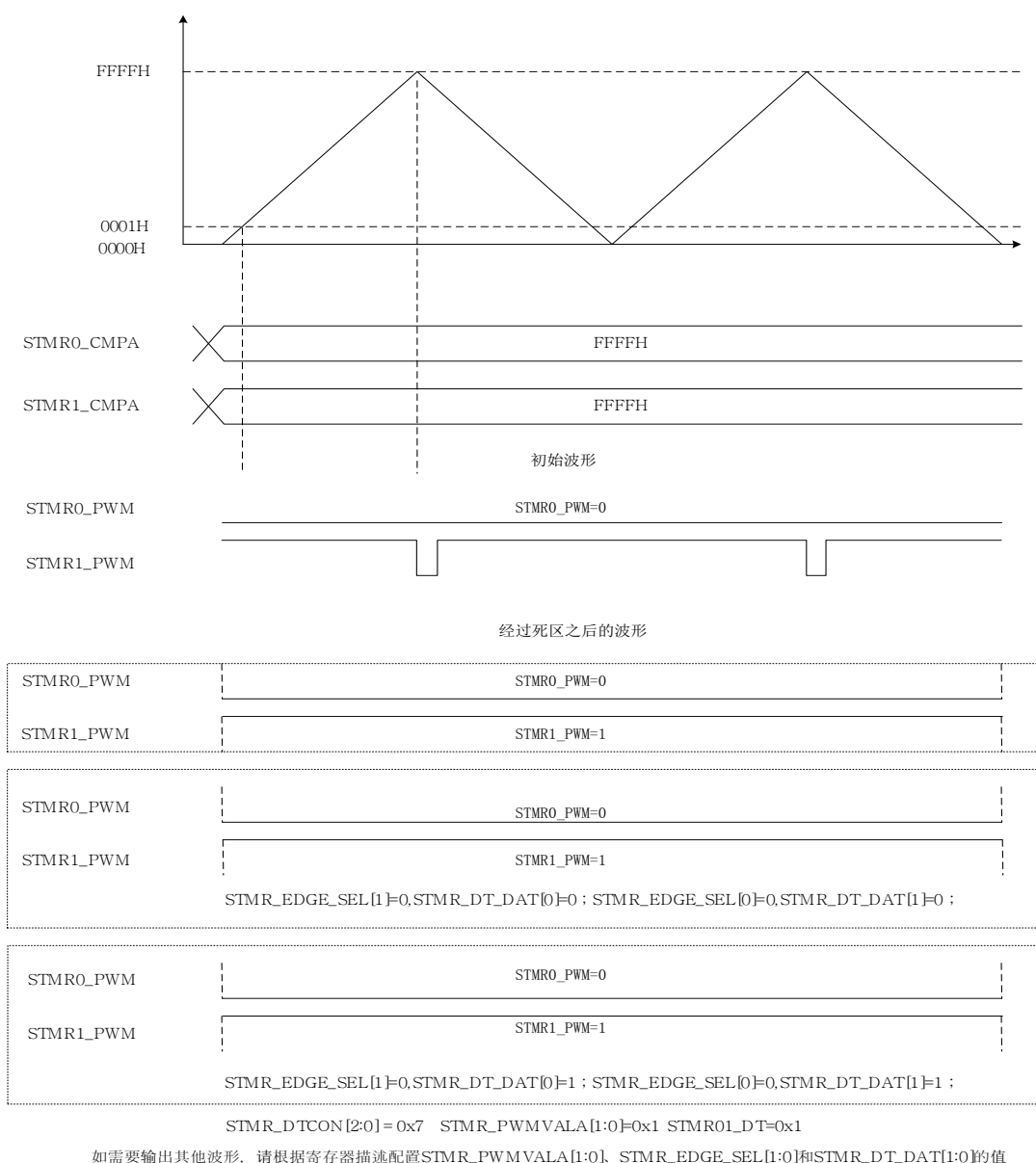


图 16-18 死区模式 7 占空比加死区大于周期时互补波形图

边沿对齐模式下, 死区的表现形式, 请参考 STMR_DTCON 寄存器描述中死区选择模式, 向下计数部分描述。

16.1.2.6. 刹车功能

可触发 PWM 刹车的信号源有如下几种:

- 外部触发端口 FB;



- ADC 结果比较输出;
- 比较器 0 的输出;
- 比较器 1 的输出;
- 软件输入;

STMRn 每个通道有独立的刹车开关, 比较器有选择位以及使能位, ADC 比较和 FB 端口有独立的使能位, 除软件输入刹车信号以外都可以选择极性。触发刹车后, 刹车标志位 STMRn_IF 刹车标志位置 1, 通道的计数器使能位清零, 且 PWM 输出预设的刹车数据。

如需恢复正常输出, 则需要将刹车标志清零。通过 STMR_BRKDAT[6]控制当刹车有效时是否关闭计数使能。(刹车相关的配置请参见刹车控制相关寄存器和刹车数据寄存器的说明)。

刹车配置流程:

1、比较器刹车

- 1) 配置 STMR_BRKCON[0]选择比较器信号源;
- 2) 配置 STMR_BRKCON[1]选择是否滤波, 如滤波, 配置 STMR_BRKFILT;
- 3) 配置 STMR_BRKCON[3]选择刹车有效时是否清零计数器;
- 4) 配置 STMR_BRKCON[6]选择刹车信号极性;
- 5) 配置 STMR_BRKDAT[6]选择刹车信号有效时, 是否关闭计数使能;
- 6) 配置 STMR_BRKEN[5: 0]选择刹车信号对哪一路 STMR 有效;
- 7) 配置 STMR_BRKDAT[5: 0]选择对应 STMRn 在刹车信号有效时的 PWM 输出值;
- 8) 配置 STMR_BRKEN[6]使能比较器刹车;

2、FB 刹车和 ADC 刹车配置除了不需要选择信号源, 其他与比较器刹车类似, 请根据寄存器描述进行相关配置;

3、软件刹车

- 1) 配置 STMR_BRKCON[3]选择刹车有效时是否清零计数器;
- 2) 配置 STMR_BRKDAT[6]选择刹车信号有效时, 是否关闭计数使能;
- 3) 配置 STMR_BRKEN[5: 0]选择刹车信号对哪一路 STMR 有效;
- 4) 配置 STMR_BRKDAT[5: 0]选择对应 STMRn 在刹车信号有效时的 PWM 输出值;
- 5) 配置 STMR_BRKCON[2]=1, 刹车;

16.1.2.7. 中断功能

本模块总共有 30 个中断标志, 其中 6 个周期中断标志, 6 个零点中断标志, 6 个向上比较中断标志, 6 个向下比较中断标志, 6 个刹车中断标志。中断标志位的产生与对应中断使能位是否开启无关。开启 STMRn 任何一种类型的中断均需打开全局中断使能位 (EA=1)、



STMRn 总中断使能位 STMRn (IE1[7: 3]和 IE2[0])，以及 STMRn_IE 才能成功配置 STMRn 中断功能。STMRn 的所有 5 个中断（周期、零点、向上和向下比较、刹车）共用一个中断向量入口，故进入中断服务程序后用户可通过中断标志位判断是哪种类型中断产生。

16.2. 模块框图

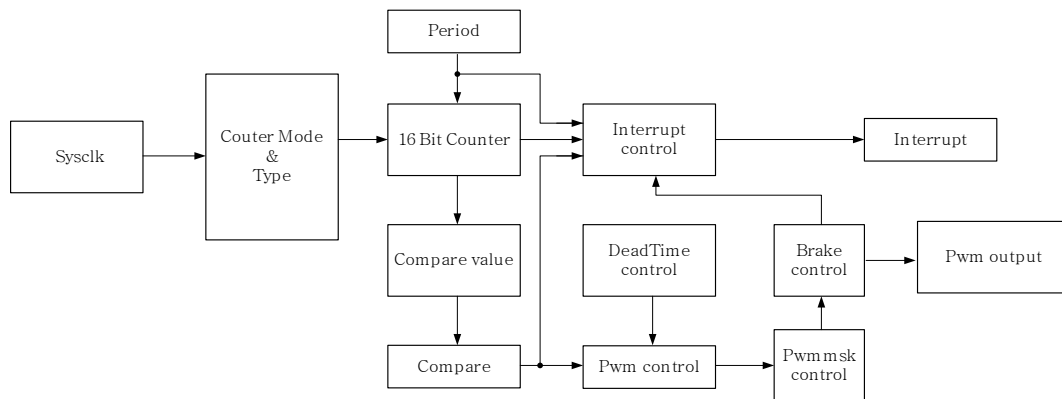


图 16-19 Super Timer 模块框图

16.3. 寄存器列表

表 16-1 Super Timer 寄存器列表

Address	Register Name	Description
0x0E (XSFR)	STMR_EDGESEL	Super Timer Pwm Edge Select Register
0x0F (XSFR)	STMR_DTDAT	Super Timer DeadTime Pwm Output Data Register
0x10 (XSFR)	STMR_CON0	Super Timer Control 0 Register
0x11 (XSFR)	STMR_CNTMD	Super Timer Counter Mode Register
0x12 (XSFR)	STMR_CNTCLR	Super Timer Counter Clear Register
0x13 (XSFR)	STMR_CNTTYPE	Super Timer Counter Type Register
0x14 (XSFR)	STMR_CNTEN	Super Timer Counter Enable Register



0x15 (XSFR)	STMR_CMPCL	Super Timer Point C Comparison Value Low Register
0x16 (XSFR)	STMR_CMPCH	Super Timer Point C Comparison Value High Register
0x17 (XSFR)	STMR_LOADEN	Super Timer Load Enable Register
0x18 (XSFR)	STMR_PWMEN	Super Timer Pwm Enable Register
0x19 (XSFR)	STMR_PWMVALA	Super Timer Point A Pwm Value Register
0x1A (XSFR)	STMR_PWMVALB	Super Timer Point B Pwm Value Register
0x1B (XSFR)	STMR_PWMBEN	Super Timer Point B Pwm Enable Register
0x1C (XSFR)	STMR_PWMMSKEN	Super Timer PWM Mask Enable Register
0x1D (XSFR)	STMR_PWMMSKD	Super Timer PWM Mask Data Register
0x1E (XSFR)	STMR_BRKEN	Super Timer Brake Enable Register
0x1F (XSFR)	STMR_BRKDAT	Super Timer Brake Data Register
0x20 (XSFR)	STMR_BRKCON	Super Timer Brake Control Register
0x21 (XSFR)	STMR01_DT	Super Timer 0 and 1 DeadTime Register
0x22 (XSFR)	STMR23_DT	Super Timer 2 and 3 DeadTime Register
0x23 (XSFR)	STMR45_DT	Super Timer 4 and 5 DeadTime Register
0x24 (XSFR)	STMR_DTCON	Super Timer DeadTime Control Register
0x25 (XSFR)	STMR_DTEN	Super Timer DeadTime Enable Register
0x26 (XSFR)	STMR0_PRL	Super Timer 0 Period Low Register
0x27 (XSFR)	STMR0_PRH	Super Timer 0 Period High Register
0x28 (XSFR)	STMR1_PRL	Super Timer 1 Period Low Register
0x29 (XSFR)	STMR1_PRH	Super Timer 1 Period High Register
0x2A (XSFR)	STMR2_PRL	Super Timer 2 Period Low Register



0x2B (XSFR)	STMR2_PRH	Super Timer 2 Period High Register
0x2C (XSFR)	STMR3_PRL	Super Timer 3 Period Low Register
0x2D (XSFR)	STMR3_PRH	Super Timer 3 Period High Register
0x2E (XSFR)	STMR4_PRL	Super Timer 4 Period Low Register
0x2F (XSFR)	STMR4_PRH	Super Timer 4 Period High Register
0x30 (XSFR)	STMR5_PRL	Super Timer 5 Period Low Register
0x31 (XSFR)	STMR5_PRH	Super Timer 5 Period High Register
0x32 (XSFR)	STMR0_CMPAL	Super Timer 0 Point A Comparison Value Low Register
0x33 (XSFR)	STMR0_CMPAH	Super Timer 0 Point A Comparison Value High Register
0x34 (XSFR)	STMR0_CMPBL	Super Timer 0 Point B Comparison Value Low Register
0x35 (XSFR)	STMR0_CMPBH	Super Timer 0 Point B Comparison Value High Register
0x36 (XSFR)	STMR1_CMPAL	Super Timer 1 Point A Comparison Value Low Register
0x37 (XSFR)	STMR1_CMPAH	Super Timer 1 Point A Comparison Value High Register
0x38 (XSFR)	STMR1_CMPBL	Super Timer 1 Point B Comparison Value Low Register
0x39 (XSFR)	STMR1_CMPBH	Super Timer 1 Point B Comparison Value High Register
0x3A (XSFR)	STMR2_CMPAL	Super Timer 2 Point A Comparison Value Low Register
0x3B (XSFR)	STMR2_CMPAH	Super Timer 2 Point A Comparison Value High



		Register
0x3C (XSFR)	STMR2_CMPBL	Super Timer 2 Point B Comparison Value Low Register
0x3D (XSFR)	STMR2_CMPBH	Super Timer 2 Point B Comparison Value High Register
0x3E (XSFR)	STMR3_CMPAL	Super Timer 3 Point A Comparison Value Low Register
0x3F (XSFR)	STMR3_CMPAH	Super Timer 3 Point A Comparison Value High Register
0x40 (XSFR)	STMR3_CMPBL	Super Timer 3 Point B Comparison Value Low Register
0x41 (XSFR)	STMR3_CMPBH	Super Timer 3 Point B Comparison Value High Register
0x42 (XSFR)	STMR4_CMPAL	Super Timer 4 Point A Comparison Value Low Register
0x43 (XSFR)	STMR4_CMPAH	Super Timer 4 Point A Comparison Value High Register
0x44 (XSFR)	STMR4_CMPBL	Super Timer 4 Point B Comparison Value Low Register
0x45 (XSFR)	STMR4_CMPBH	Super Timer 4 Point B Comparison Value High Register
0x46 (XSFR)	STMR5_CMPAL	Super Timer 5 Point A Comparison Value Low Register
0x47 (XSFR)	STMR5_CMPAH	Super Timer 5 Point A Comparison Value High Register
0x48 (XSFR)	STMR5_CMPBL	Super Timer 5 Point B Comparison Value Low Register



0x49 (XSFR)	STMR5_CMPBH	Super Timer 5 Point B Comparison Value High Register
0x4A (XSFR)	STMR0_CNTL	Super Timer 0 Counter Low Register
0x4B (XSFR)	STMR0_CNTH	Super Timer 0 Counter High Register
0x4C (XSFR)	STMR1_CNTL	Super Timer 1 Counter Low Register
0x4D (XSFR)	STMR1_CNTH	Super Timer 1 Counter High Register
0x4E (XSFR)	STMR2_CNTL	Super Timer 2 Counter Low Register
0x4F (XSFR)	STMR2_CNTH	Super Timer 2 Counter High Register
0x50 (XSFR)	STMR3_CNTL	Super Timer 3 Counter Low Register
0x51 (XSFR)	STMR3_CNTH	Super Timer 3 Counter High Register
0x52 (XSFR)	STMR4_CNTL	Super Timer 4 Counter Low Register
0x53 (XSFR)	STMR4_CNTH	Super Timer 4 Counter High Register
0x54 (XSFR)	STMR5_CNTL	Super Timer 5 Counter Low Register
0x55 (XSFR)	STMR5_CNTH	Super Timer 5 Counter High Register
0x56 (XSFR)	STMR0_PSC	Super Timer 0 Prescaler Register
0x57 (XSFR)	STMR1_PSC	Super Timer 1 Prescaler Register
0x58 (XSFR)	STMR2_PSC	Super Timer 2 Prescaler Register
0x59 (XSFR)	STMR3_PSC	Super Timer 3 Prescaler Register
0x5A (XSFR)	STMR4_PSC	Super Timer 4 Prescaler Register
0x5B (XSFR)	STMR5_PSC	Super Timer 5 Prescaler Register
0x5C (XSFR)	STMR0_IE	Super Timer 0 Interrupt Enable Register
0x5D (XSFR)	STMR1_IE	Super Timer 1 Interrupt Enable Register
0x5E (XSFR)	STMR2_IE	Super Timer 2 Interrupt Enable Register



0x5F (XSFR)	STMR3_IE	Super Timer 3 Interrupt Enable Register
0x60 (XSFR)	STMR4_IE	Super Timer 4 Interrupt Enable Register
0x61 (XSFR)	STMR5_IE	Super Timer 5 Interrupt Enable Register
0x62 (XSFR)	STMR0_IF	Super Timer 0 Interrupt Flag Register
0x63 (XSFR)	STMR1_IF	Super Timer 1 Interrupt Flag Register
0x64 (XSFR)	STMR2_IF	Super Timer 2 Interrupt Flag Register
0x65 (XSFR)	STMR3_IF	Super Timer 3 Interrupt Flag Register
0x66 (XSFR)	STMR4_IF	Super Timer 4 Interrupt Flag Register
0x67 (XSFR)	STMR5_IF	Super Timer 5 Interrupt Flag Register
0x68 (XSFR)	STMR_BRKFILT	Super Timer Brake Filter Number Register

16.4. 寄存器详细说明

16.4.1. STMR_CON0

Addr = 0x10 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	STMRGP1	STMR1/STMR3/STMR5 群组功能使能 0x0: 群组功能不使能 0x1: 群组功能使能	RW	0x0
6	STMRGP0	STMR0/STMR2/STMR4 群组功能使能 0x0: 群组功能不使能 0x1: 群组功能使能	RW	0x0
5: 4	ST45SYNCP	STMR4/STMR5 模式选择 0x0: STMR4 和 STMR5 为独立模式 0x1: STMR4 和 STMR5 为同步模式 0x2: STMR4 和 STMR5 为互补模式 0x3: 保留	RW	0x0
3: 2	ST23SYNCP	STMR2/STMR3 模式选择	RW	0x0



		0x0: STMR2 和 STMR3 为独立模式 0x1: STMR2 和 STMR3 为同步模式 0x2: STMR2 和 STMR3 为互补模式 0x3: 保留		
1: 0	ST01SYNCMP	STMR0/STMR1 模式选择 0x0: STMR0 和 STMR1 为独立模式 0x1: STMR0 和 STMR1 为同步模式 0x2: STMR0 和 STMR1 为互补模式 0x3: 保留	RW	0x0

16.4.2. STMR_CNTMD

Addr = 0x11 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	LOADSEL	自动装载点选择 0x0: 计数至 0 时自动装载 0x1: 计数至周期自动装载	RW	0x0
6	CMPCSEL	比较值 C 点选择位 0x0: 计数方向向上时, 比较值 C 有效 0x1: 计数方向向下时, 比较值 C 有效	RW	0x0
5	STMR5CNTM	STMR5 计数模式选择 0x0: 单次计数模式 0x1: 连续计数模式	RW	0x0
4	STMR4CNTM	STMR4 计数模式选择 0x0: 单次计数模式 0x1: 连续计数模式	RW	0x0
3	STMR3CNTM	STMR3 计数模式选择 0x0: 单次计数模式 0x1: 连续计数模式	RW	0x0
2	STMR2CNTM	STMR2 计数模式选择 0x0: 单次计数模式 0x1: 连续计数模式	RW	0x0
1	STMR1CNTM	STMR1 计数模式选择	RW	0x0



		0x0: 单次计数模式 0x1: 连续计数模式		
0	STMROCNTM	STMRO 计数模式选择 0x0: 单次计数模式 0x1: 连续计数模式	RW	0x0

16.4.3. STMR_CNTCLR

Addr = 0x12 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	—
5	STMR5CNTCLR	STMR5 计数清零 写 1 清零, 写 0 无效	WO	—
4	STMR4CNTCLR	STMR4 计数清零 写 1 清零, 写 0 无效	WO	—
3	STMR3CNTCLR	STMR3 计数清零 写 1 清零, 写 0 无效	WO	—
2	STMR2CNTCLR	STMR2 计数清零 写 1 清零, 写 0 无效	WO	—
1	STMR1CNTCLR	STMR1 计数清零 写 1 清零, 写 0 无效	WO	—
0	STMROCNTCLR	STMRO 计数清零 写 1 清零, 写 0 无效	WO	—

16.4.4. STMR_CNTTYPE

Addr = 0x13 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	TMR1EN	STMR_CNTEN[7] 硬件置 1 使能 0x0: 硬件置 1 无效; 0x1: 硬件置 1 使能, STMRO 计数值等于 0 时,	RW	0x0



		STMR_CNTEN[7]硬件置 1;		
6	TMR0EN	STMR_CNTEN[6]硬件置 1 使能 0x0: 硬件置 1 无效; 0x1: 硬件置 1 使能, STMR0 计数值等于 0 时, STMR_CNTEN[6]硬件置 1;	RW	0x0
5	STMR5CNTTP	STMR5 计数类型选择 0x0: 边沿对齐计数 0x1: 中心对齐计数	RW	0x0
4	STMR4CNTTP	STMR4 计数类型选择 0x0: 边沿对齐计数 0x1: 中心对齐计数	RW	0x0
3	STMR3CNTTP	STMR3 计数类型选择 0x0: 边沿对齐计数 0x1: 中心对齐计数	RW	0x0
2	STMR2CNTTP	STMR2 计数类型选择 0x0: 边沿对齐计数 0x1: 中心对齐计数	RW	0x0
1	STMR1CNTTP	STMR1 计数类型选择 0x0: 边沿对齐计数 0x1: 中心对齐计数	RW	0x0
0	STMR0CNTTP	STMR0 计数类型选择 0x0: 边沿对齐计数 0x1: 中心对齐计数	RW	0x0

16.4.5. STMR_CNTEN

Addr = 0x14 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	TMR1CNTEN	TMR1 计数使能 0x0: 计数不使能 0x1: 计数使能	RW	0x0



6	TMROCNTEN	TMRO 计数使能 0x0: 计数不使能 0x1: 计数使能	RW	0x0
5	STMR5CNTEN	STMR5 计数使能 0x0: 计数不使能 0x1: 计数使能	RW	0x0
4	STMR4CNTEN	STMR4 计数使能 0x0: 计数不使能 0x1: 计数使能	RW	0x0
3	STMR3CNTEN	STMR3 计数使能 0x0: 计数不使能 0x1: 计数使能	RW	0x0
2	STMR2CNTEN	STMR2 计数使能 0x0: 计数不使能 0x1: 计数使能	RW	0x0
1	STMR1CNTEN	STMR1 计数使能 0x0: 计数不使能 0x1: 计数使能	RW	0x0
0	STMR0CNTEN	STMR0 计数使能 0x0: 计数不使能 0x1: 计数使能	RW	0x0

16.4.6. STMR_LOADEN

Addr = 0x17 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	–	–	–	0
5	STMR5LOADEN	STMR5 自动装载使能 0x0: 自动装载不使能 0x1: 自动装载使能 Note: 每次写周期和比较值影子寄存器后, 需将此位置 1, 到装载点时新值才会有效载入! 目的是为	RW	0x0



		了保护写影子寄存器中途未完成就到了装载点自动更新导致错误装载的发生！		
4	STMR4LOADEN	STMR4 自动装载使能 0x0: 自动装载不使能 0x1: 自动装载使能 Note: 每次写周期和比较值影子寄存器后, 需将此位置 1, 到装载点时新值才会有效载入! 目的是为了 保护写影子寄存器中途未完成就到了装载点自动更新导致错误装载的发生！	RW	0x0
3	STMR3LOADEN	STMR3 自动装载使能 0x0: 自动装载不使能 0x1: 自动装载使能 Note: 每次写周期和比较值影子寄存器后, 需将此位置 1, 到装载点时新值才会有效载入! 目的是为了 保护写影子寄存器中途未完成就到了装载点自动更新导致错误装载的发生！	RW	0x0
2	STMR2LOADEN	STMR2 自动装载使能 0x0: 自动装载不使能 0x1: 自动装载使能 Note: 每次写周期和比较值影子寄存器后, 需将此位置 1, 到装载点时新值才会有效载入! 目的是为了 保护写影子寄存器中途未完成就到了装载点自动更新导致错误装载的发生！	RW	0x0
1	STMR1LOADEN	STMR1 自动装载使能 0x0: 自动装载不使能 0x1: 自动装载使能 Note: 每次写周期和比较值影子寄存器后, 需将此位置 1, 到装载点时新值才会有效载入! 目的是为了 保护写影子寄存器中途未完成就到了装载点自动更新导致错误装载的发生！	RW	0x0
0	STMR0LOADEN	STMR0 自动装载使能 0x0: 自动装载不使能 0x1: 自动装载使能	RW	0x0



		Note: 每次写周期和比较值影子寄存器后, 需将此位置 1, 到装载点时新值才会有效载入! 目的是为了 保护写影子寄存器中途未完成就到了装载点自动更新导致错误装载的发生!		

16.4.7. STMR_CMPCL

Addr = 0x15 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	STMRCMPCL	比较值 C 点低八位 Note: 比较值 C 点只与 STMRO 的计数进行比较	WO	—

16.4.8. STMR_CMPCH

Addr = 0x16 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	STMRCMPCH	比较值 C 点高八位 Note: 比较值 C 点只与 STMRO 的计数进行比较	WO	—

16.4.9. STMR_PWMEN

Addr = 0x18 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	0
5	STMR5PWMEN	STMR5 PWM 输出使能 0x0: 不使能 0x1: 使能	RW	0x0
4	STMR4PWMEN	STMR4 PWM 输出使能 0x0: 不使能 0x1: 使能	RW	0x0
3	STMR3PWMEN	STMR3 PWM 输出使能 0x0: 不使能 0x1: 使能	RW	0x0



2	STMR2PWMEN	STMR2 PWM 输出使能 0x0: 不使能 0x1: 使能	RW	0x0
1	STMR1PWMEN	STMR1 PWM 输出使能 0x0: 不使能 0x1: 使能	RW	0x0
0	STMR0PWMEN	STMR0 PWM 输出使能 0x0: 不使能 0x1: 使能	RW	0x0

16.4.10. STMR_PWMVALA

Addr = 0x19 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	0x0
5	STMR5PWMVA	STMR5 PWM 输出值 0x0: 计数 CNT 小于比较值 A, PWM 输出 1, 大于等于输出 0; 0x1: 计数 CNT 大于等于比较值 A, PWM 输出 1, 小于输出 0;	RW	0x0
4	STMR4PWMVA	STMR4 PWM 输出值 0x0: 计数 CNT 小于比较值 A, PWM 输出 1, 大于等于输出 0; 0x1: 计数 CNT 大于等于比较值 A, PWM 输出 1, 小于输出 0;	RW	0x0
3	STMR3PWMVA	STMR3 PWM 输出值 0x0: 计数 CNT 小于比较值 A, PWM 输出 1, 大于等于输出 0; 0x1: 计数 CNT 大于等于比较值 A, PWM 输出 1, 小于输出 0;	RW	0x0
2	STMR2PWMVA	STMR2 PWM 输出值 0x0: 计数 CNT 小于比较值 A, PWM 输出 1, 大于等	RW	0x0



		于输出 0; 0x1: 计数 CNT 大于等于比较值 A, PWM 输出 1, 小于输出 0;		
1	STMR1PWMVA	STMR1 PWM 输出值 0x0: 计数 CNT 小于比较值 A, PWM 输出 1, 大于等于输出 0; 0x1: 计数 CNT 大于等于比较值 A, PWM 输出 1, 小于输出 0;	RW	0x0
0	STMR0PWMVA	STMR0 PWM 输出值 0x0: 计数 CNT 小于比较值 A, PWM 输出 1, 大于等于输出 0; 0x1: 计数 CNT 大于等于比较值 A, PWM 输出 1, 小于输出 0;	RW	0x0

16.4.11. STMR_PWMVALB

Addr =0x1A (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	0x0
5	STMR5PWMVB	STMR5 比较值 B 有效 PWM 输出值 0x0: 计数 CNT 小于比较值 B, PWM 输出 1, 大于等于输出 0; 0x1: 计数 CNT 大于等于比较值 B, PWM 输出 1, 小于输出 0;	RW	0x0
4	STMR4PWMVB	STMR4 比较值 B 有效 PWM 输出值 0x0: 计数 CNT 小于比较值 B, PWM 输出 1, 大于等于输出 0; 0x1: 计数 CNT 大于等于比较值 B, PWM 输出 1, 小于输出 0;	RW	0x0
3	STMR3PWMVB	STMR3 比较值 B 有效 PWM 输出值 0x0: 计数 CNT 小于比较值 B, PWM 输出 1, 大于等于输出 0;	RW	0x0



		0x1: 计数 CNT 大于等于比较值 B, PWM 输出 1, 小于输出 0;		
2	STMR2PWMVB	STMR2 比较值 B 有效 PWM 输出值 0x0: 计数 CNT 小于比较值 B, PWM 输出 1, 大于等于输出 0; 0x1: 计数 CNT 大于等于比较值 B, PWM 输出 1, 小于输出 0;	RW	0x0
1	STMR1PWMVB	STMR1 比较值 B 有效 PWM 输出值 0x0: 计数 CNT 小于比较值 B, PWM 输出 1, 大于等于输出 0; 0x1: 计数 CNT 大于等于比较值 B, PWM 输出 1, 小于输出 0;	RW	0x0
0	STMR0PWMVB	STMR0 比较值 B 有效 PWM 输出值 0x0: 计数 CNT 小于比较值 B, PWM 输出 1, 大于等于输出 0; 0x1: 计数 CNT 大于等于比较值 B, PWM 输出 1, 小于输出 0;	RW	0x0

16.4.12. STMR_PWMBEN

Addr = 0x1B (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	0x0
5	STMR5PWMBEN	STMR5 比较值 B PWM 输出使能 0x0: 不使能 0x1: 使能	RW	0x0
4	STMR4PWMBEN	STMR4 比较值 B PWM 输出使能 0x0: 不使能 0x1: 使能	RW	0x0
3	STMR3PWMBEN	STMR3 比较值 B PWM 输出使能 0x0: 不使能 0x1: 使能	RW	0x0



2	STMR2PWMBEN	STMR2 比较值 B PWM 输出使能 0x0: 不使能 0x1: 使能	RW	0x0
1	STMR1PWMBEN	STMR1 比较值 B PWM 输出使能 0x0: 不使能 0x1: 使能	RW	0x0
0	STMR0PWMBEN	STMR0 比较值 B PWM 输出使能 0x0: 不使能 0x1: 使能	RW	0x0

16.4.13. STMR_PWMMSKEN

Addr = 0x1C (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	0x0
5	STMR5PWMMSKEN	STMR5 PWM 掩码输出使能 0x0: 不使能 0x1: 使能	RW	0x0
4	STMR4PWMMSKEN	STMR4 PWM 掩码输出使能 0x0: 不使能 0x1: 使能	RW	0x0
3	STMR3PWMMSKEN	STMR3 PWM 掩码输出使能 0x0: 不使能 0x1: 使能	RW	0x0
2	STMR2PWMMSKEN	STMR2 PWM 掩码输出使能 0x0: 不使能 0x1: 使能	RW	0x0
1	STMR1PWMMSKEN	STMR1 PWM 掩码输出使能 0x0: 不使能 0x1: 使能	RW	0x0
0	STMR0PWMMSKEN	STMR0 PWM 掩码输出使能 0x0: 不使能 0x1: 使能	RW	0x0



16.4.14. STMR_PWMMSKD

Addr = 0x1D (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	0x0
5	STMR5PWMMSKD	STMR5 PWM 掩码输出值 0x0: 掩码输出 0 0x1: 掩码输出 1	RW	0x0
4	STMR4PWMMSKD	STMR4 PWM 掩码输出值 0x0: 掩码输出 0 0x1: 掩码输出 1	RW	0x0
3	STMR3PWMMSKD	STMR3 PWM 掩码输出值 0x0: 掩码输出 0 0x1: 掩码输出 1	RW	0x0
2	STMR2PWMMSKD	STMR2 PWM 掩码输出值 0x0: 掩码输出 0 0x1: 掩码输出 1	RW	0x0
1	STMR1PWMMSKD	STMR1 PWM 掩码输出值 0x0: 掩码输出 0 0x1: 掩码输出 1	RW	0x0
0	STMR0PWMMSKD	STMR0 PWM 掩码输出值 0x0: 掩码输出 0 0x1: 掩码输出 1	RW	0x0

16.4.15. STMR_BRKEN

Addr = 0x1E (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	BRKFBEN	FB 端口刹车使能 0x0: 不使能 0x1: 使能	RW	0x0
6	BRKCOMPEN	比较器刹车使能 0x0: 不使能	RW	0x0



		0x1: 使能		
5	STMR5BRKEN	STMR5 刹车使能 0x0: 不使能 0x1: 使能	RW	0x0
4	STMR4BRKEN	STMR4 刹车使能 0x0: 不使能 0x1: 使能	RW	0x0
3	STMR3BRKEN	STMR3 刹车使能 0x0: 不使能 0x1: 使能	RW	0x0
2	STMR2BRKEN	STMR2 刹车使能 0x0: 不使能 0x1: 使能	RW	0x0
1	STMR1BRKEN	STMR1 刹车使能 0x0: 不使能 0x1: 使能	RW	0x0
0	STMR0BRKEN	STMR0 刹车使能 0x0: 不使能 0x1: 使能	RW	0x0

16.4.16. STMR_BRKCON

Addr = 0x20 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	BRKFB_POL	FB 端口刹车极性 0x0: 刹车信号 0 有效 0x1: 刹车信号 1 有效	RW	0x1
6	BRKCOMPOL	比较器刹车极性 0x0: 刹车信号 0 有效 0x1: 刹车信号 1 有效	RW	0x1
5	BRKADCPOL	ADC 刹车极性 0x0: 刹车信号 0 有效 0x1: 刹车信号 1 有效	RW	0x1



4	BRKADCEN	ADC 刹车使能 0x0: 不使能 0x1: 使能	RW	0x0
3	BRKNTCLREN	刹车有效计数寄存器清零使能 0x0: 不使能 0x1: 使能	RW	0x0
2	BRKSOF	软件刹车信号 0x0: 不刹车 0x1: 刹车	RW	0x0
1	BRKFILTEN	ADC/比较器/FB 刹车滤波使能 0x0: 不使能 0x1: 使能	RW	0x0
0	BRKCOMPSEL	比较器刹车选择 0x0: 比较器 0 刹车有效 0x1: 比较器 1 刹车有效	RW	0x0

16.4.17. STMR_BRKDAT

Addr = 0x1F (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	0x0
5	STMR5BRKDAT	STMR5 刹车有效 PWM 输出值 0x0: 刹车有效时 PWM 输出 0 0x1: 刹车有效时 PWM 输出 1	RW	0x0
4	STMR4BRKDAT	STMR4 刹车有效 PWM 输出值 0x0: 刹车有效时 PWM 输出 0 0x1: 刹车有效时 PWM 输出 1	RW	0x0
3	STMR3BRKDAT	STMR3 刹车有效 PWM 输出值 0x0: 刹车有效时 PWM 输出 0 0x1: 刹车有效时 PWM 输出 1	RW	0x0
2	STMR2BRKDAT	STMR2 刹车有效 PWM 输出值 0x0: 刹车有效时 PWM 输出 0 0x1: 刹车有效时 PWM 输出 1	RW	0x0



1	STMR1BRKDAT	STMR1 刹车有效 PWM 输出值 0x0: 刹车有效时 PWM 输出 0 0x1: 刹车有效时 PWM 输出 1	RW	0x0
0	STMR0BRKDAT	STMR0 刹车有效 PWM 输出值 0x0: 刹车有效时 PWM 输出 0 0x1: 刹车有效时 PWM 输出 1	RW	0x0

16.4.18. STMR_BRKFILT

Addr = 0x68 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	0x0
5: 0	STMRBRKFILTNUM	刹车滤波时长 滤波时长为 STMRBRKFILTNUM + 4 个 CLK	RW	0x0

16.4.19. STMR01_DT

Addr = 0x21 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	STMR01_DT	STMR0/STMR1 死区时间寄存器	WO	—

16.4.20. STMR23_DT

Addr = 0x22 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	STMR23DT	STMR2/STMR3 死区时间寄存器	WO	—

16.4.21. STMR45_DT

Addr = 0x23 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	STMR45DT	STMR4/STMR5 死区时间寄存器	WO	—



16.4.22. STMR_DTCON

Addr = 0x24 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	0x0
5: 3	STMR23DTTYPE	STMR2/STMR3 死区模式选择 0x0: STMR2 向上计数时比较点+死区时间, 向下计数时无死区; STMR3 向上计数时无死区, 向下计数时比较点-死区时间 0x1: STMR2 向下计数时比较点+死区时间, 向上计数时无死区; STMR3 向下计数时无死区, 向上计数时比较点-死区时间 0x2: STMR2 向上计数时比较点-死区时间, 向下计数时无死区; STMR3 向上计数时无死区, 向下计数时比较点+死区时间 0x3: STMR2 向下计数时比较点-死区时间, 向上计数时无死区; STMR3 向下计数时无死区, 向上计数时比较点+死区时间 0x4: STMR2 向下和向上计数时比较点-死区时间, STMR3 无死区 0x5: STMR2 向下和向上计数时比较点+死区时间, STMR3 无死区 0x6: STMR2 无死区, STMR3 向下和向上计数时比较点-死区时间 0x7: STMR2 无死区, STMR3 向下和向上计数时比较点+死区时间	RW	0x0
2: 0	STMR01DTTYPE	STMR0/STMR1 死区模式选择 0x0: STMR0 向上计数时比较点+死区时间, 向下计数时无死区; STMR1 向上计数时无死区, 向下计数时比较点-死区时间 0x1: STMR0 向下计数时比较点+死区时间, 向上计数时无死区; STMR1 向下计数时无死区, 向上计数时比较点-死区时间	RW	0x0



		0x2: STMR0 向上计数时比较点-死区时间, 向下计数时无死区; STMR1 向上计数时无死区, 向下计数时比较点+死区时间 0x3: STMR0 向下计数时比较点-死区时间, 向上计数时无死区; STMR1 向下计数时无死区, 向上计数时比较点+死区时间 0x4: STMR0 向下和向上计数时比较点-死区时间, STMR1 无死区 0x5: STMR0 向下和向上计数时比较点+死区时间, STMR1 无死区 0x6: STMR0 无死区, STMR1 向下和向上计数时比较点-死区时间 0x7: STMR0 无死区, STMR1 向下和向上计数时比较点+死区时间		

16.4.23. STMR_DTEN

Addr = 0x25 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	0x0
5: 3	STMR45DTTYPE	STMR4/STMR5 死区模式选择 0x0: STMR4 向上计数时比较点+死区时间, 向下计数时无死区; STMR5 向上计数时无死区, 向下计数时比较点-死区时间 0x1: STMR4 向下计数时比较点+死区时间, 向上计数时无死区; STMR5 向下计数时无死区, 向上计数时比较点-死区时间 0x2: STMR4 向上计数时比较点-死区时间, 向下计数时无死区; STMR5 向上计数时无死区, 向下计数时比较点+死区时间 0x3: STMR4 向下计数时比较点-死区时间, 向上计数时无死区; STMR5 向下计数时无死区, 向上计数时比较点+死区时间	RW	0x0



		0x4: STMR4 向下和向上计数时比较点-死区时间, STMR5 无死区 0x5: STMR4 向下和向上计数时比较点+死区时间, STMR5 无死区 0x6: STMR4 无死区, STMR5 向下和向上计数时比 较点-死区时间 0x7: STMR4 无死区, STMR5 向下和向上计数时比 较点+死区时间		
2	STMR45DTEN	STMR4/STMR5 死区使能 0x0: 不使能 0x1: 使能	RW	0x0
1	STMR23DTEN	STMR2/STMR3 死区使能 0x0: 不使能 0x1: 使能	RW	0x0
0	STMR01DTEN	STMR0/STMR1 死区使能 0x0: 不使能 0x1: 使能	RW	0x0

16.4.24. STMR_EDGESEL

Addr = 0x0E (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	0x0
5	STMR5EDGESEL	STMR5 PWM 边沿选择 0x0: STMR5 PWM 下降沿触发 STMR4 死区事件 0x1: STMR5 PWM 上升沿触发 STMR4 死区事件	RW	0x0
4	STMR4EDGESEL	STMR4 PWM 边沿选择 0x0: STMR4 PWM 下降沿触发 STMR5 死区事件 0x1: STMR4 PWM 上升沿触发 STMR5 死区事件	RW	0x0
3	STMR3EDGESEL	STMR3 PWM 边沿选择 0x0: STMR3 PWM 下降沿触发 STMR2 死区事件 0x1: STMR3 PWM 上升沿触发 STMR2 死区事件	RW	0x0
2	STMR2EDGESEL	STMR2 PWM 边沿选择	RW	0x0



		0x0: STMR2 PWM 下降沿触发 STMR3 死区事件 0x1: STMR2 PWM 上升沿触发 STMR3 死区事件		
1	STMR1EDGESEL	STMR1 PWM 边沿选择 0x0: STMR1 PWM 下降沿触发 STMR0 死区事件 0x1: STMR1 PWM 上升沿触发 STMR0 死区事件	RW	0x0
0	STMR0EDGESEL	STMR0 PWM 边沿选择 0x0: STMR0 PWM 下降沿触发 STMR1 死区事件 0x1: STMR0 PWM 上升沿触发 STMR1 死区事件	RW	0x0

16.4.25. STMR_DTDAT

Addr = 0x0F (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	0x0
5	STMR5DTDAT	STMR5 死区事件触发后, PWM 死区时间内输出值 0x0: STMR5 PWM 死区时间内输出 0 0x1: STMR5 PWM 死区时间内输出 1	RW	0x0
4	STMR4DTDAT	STMR4 死区事件触发后, PWM 死区时间内输出值 0x0: STMR4 PWM 死区时间内输出 0 0x1: STMR4 PWM 死区时间内输出 1	RW	0x0
3	STMR3DTDAT	STMR3 死区事件触发后, PWM 死区时间内输出值 0x0: STMR3 PWM 死区时间内输出 0 0x1: STMR3 PWM 死区时间内输出 1	RW	0x0
2	STMR2DTDAT	STMR2 死区事件触发后, PWM 死区时间内输出值 0x0: STMR2 PWM 死区时间内输出 0 0x1: STMR2 PWM 死区时间内输出 1	RW	0x0
1	STMR1DTDAT	STMR1 死区事件触发后, PWM 死区时间内输出值 0x0: STMR1 PWM 死区时间内输出 0 0x1: STMR1 PWM 死区时间内输出 1	RW	0x0
0	STMR0DTDAT	STMR0 死区事件触发后, PWM 死区时间内输出值 0x0: STMR0 PWM 死区时间内输出 0 0x1: STMR0 PWM 死区时间内输出 1	RW	0x0



16.4.26. STMRn_IE (n=0~5)

Addr = 0x5C/0x5D/0x5E/0x5F/0x60/0x61 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	0x0
4	STMRNBRKIE	STMRn 刹车中断使能 0x0: 不使能 0x1: 使能	RW	0x0
3	STMRNCMPBIE	STMRn 计数值等于比较值 B 中断使能 0x0: 不使能 0x1: 使能	RW	0x0
2	STMRNCMPAIE	STMRn 计数值等于比较值 A 中断使能 0x0: 不使能 0x1: 使能	RW	0x0
1	STMRNUDIE	STMRn 计数值等于 0 中断使能 0x0: 不使能 0x1: 使能	RW	0x0
0	STMRNOVIE	STMRn 计数值等于周期中断使能 0x0: 不使能 0x1: 使能	RW	0x0

16.4.27. STMRn_IF (n=0~5)

Addr = 0x62/0x63/0x64/0x65/0x66/0x67 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	0x0
4	STMRNBRKIF	STMRn 刹车有效标志 0x0: 刹车无效或没有发生 0x1: 刹车已经发生且有效	RW	0x0
3	STMRNCMPBIF	STMRn 计数等于比较值 B 有效标志 0x0: 无效或计数不等于比较值 B 0x1: 计数等于比较值 B 已经发生且有效	RW	0x0
2	STMRNCMPAIF	STMRn 计数等于比较值 A 有效标志	RW	0x0



		0x0: 无效或计数不等于比较值 A 0x1: 计数等于比较值 A 已经发生且有效		
1	STMRNUDIF	STMRn 计数等于 0 有效标志 0x0: 无效或计数不等于 0 0x1: 计数等于 0 已经发生且有效	RW	0x0
0	STMRNOVIF	STMRn 计数等于周期有效标志 0x0: 无效或计数不等于周期 0x1: 计数等于周期已经发生且有效 Note: 边沿对齐模式时, 计数等于周期无中断标志	RW	0x0

16.4.28. STMRn_PRL (n=0~5)

Addr = 0x26/0x28/0x2A/0x2C/0x2E/0x30 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	STMRNPRL	STMRn 周期低八位寄存器	WO	—

16.4.29. STMRn_PRH (n=0~5)

Addr = 0x27/0x29/0x2B/0x2D/0x2F/0x31 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	STMRNPRH	STMRn 周期高八位寄存器	WO	—

16.4.30. STMRn_CMPAL (n=0~5)

Addr = 0x32/0x36/0x3A/0x3E/0x42/0x46 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	STMRNCMPAL	STMRn 比较值 A 点低八位寄存器	WO	—

16.4.31. STMRn_CMPAH (n=0~5)

Addr = 0x33/0x37/0x3B/0x3F/0x43/0x47 (XSFR)

Bit(s)	Name	Description	R/W	Reset



7: 0	STMRNCMPAH	STMRn 比较值 A 点高八位寄存器	WO	—

16.4.32. STMRn_CMPBL (n=0~5)

Addr = 0x34/0x38/0x3C/0x40/0x44/0x48 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	STMRNCMPBL	STMRn 比较值 B 点低八位寄存器	WO	—

16.4.33. STMRn_CMPBH (n=0~5)

Addr = 0x35/0x39/0x3D/0x41/0x45/0x49 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	STMRNCMPBH	STMRn 比较值 B 点高八位寄存器	WO	—

16.4.34. STMRn_PSC (n=0~5)

Addr = 0x56/0x57/0x58/0x59/0x5A/0x5B (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	STMRNPSC	STMRn 计数分频寄存器	WO	—

16.4.35. STMRn_CNTL (n=0~5)

Addr = 0x4A/0x4C/0x4E/0x50/0x52/0x54 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	STMRNCNTL	STMRn 计数低八位寄存器	WO	—

16.4.36. STMRn_CNTH (n=0~5)

Addr = 0x4B/0x4D/0x4F/0x51/0x53/0x55 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	STMRNCNTH	STMRn 计数高八位寄存器	WO	—



16.5. 使用流程说明

- 1) 选择PWM模式（同步，独立，互补，群组）；
- 2) 配置计数模式和计数类型；
- 3) 配置分频寄存器，周期寄存器；
- 4) 配置比较器寄存器以及PWM输出行为寄存器；
- 5) 死区以及刹车相关寄存器；
- 6) PWM输出使能值 1；
- 7) 计数使能位置 1；



17.CRC 校验模块

17.1. 功能概述

- 支持CRC32-MPEG-2、CRC8 计算
- 每一个系统周期计算 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。

17.2. 基本功能

17.2.1. CRC 基本介绍

CRC（循环冗余校验）是一种在是数据存储和数据通信的过程中，为了保证数据的正确新，而采用的一种数据校验的手段，常用的使用方法是在需要校验的数据后携带一个 CRC 校验结果，这个 CRC 校验结果是根据需要校验的数据和生成多项式进行运算后得到的一个结果，CRC 校验结果会和所使用的协议不同，对同一段数据产生不同的校验结果，以下举例目前所支持的两种的协议

例如：目前需要校验的数据段为[0x11, 0x12, 0x13, 0x14, 0x15]

- 使用 CRC32-MPEG-2 协议的计算结果：0xD783E03F

所以需要存储和发送的数据为：[0x11, 0x12, 0x13, 0x14, 0x15, 0xD7, 0x83, 0xE0, 0x3F]

- 使用 CRC8 协议的计算结果：0x55

所以需要存储和发送的数据为：[0x11, 0x12, 0x13, 0x14, 0x15, 0x55]

17.2.2. 支持的 CRC 协议

- CRC32-MPEG-2
 - 多项式： $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$ (0x04C11DB7)
 - INIT: 0xffffffff
 - REFIN: FALSE
 - REFOUT: FALSE
 - XOROUT: 0x00000000

- CRC8:
 - 多项式: x^8+x^2+x+1 (0x07)
 - INIT: 0xff
 - REFIN: FALSE
 - REFOUT: FALSE
 - XOROUT: 0x00

17.3. 模块框图

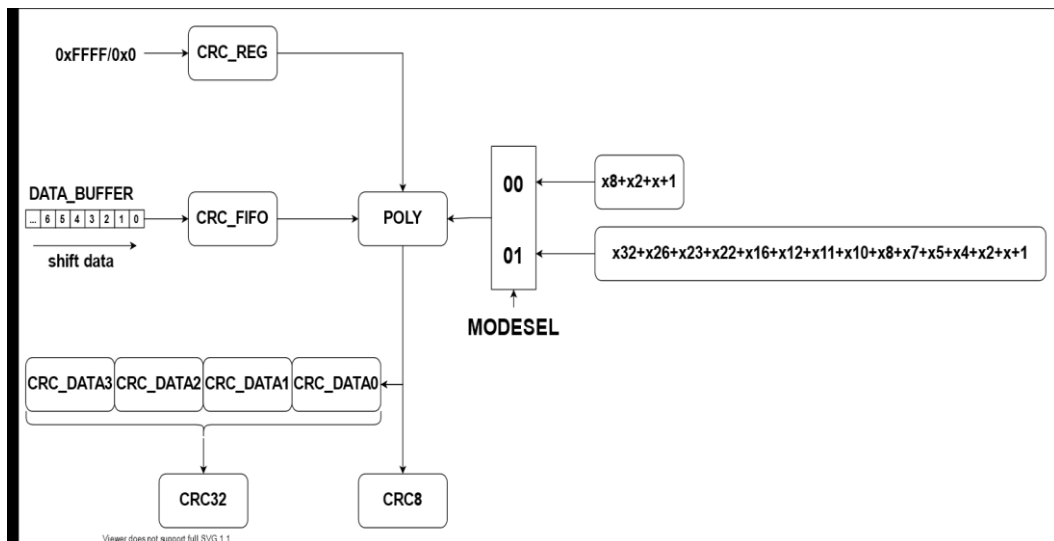


图 17-1CRC 模块结构框图

17.4. 寄存器列表

表 17-1 CRC32 register list

Offset	Register Name	Description
0xC1 (SFR)	CRC_CON	CRC configuration register
0xC2 (SFR)	CRC_REG	CRC initial register
0xC3 (SFR)	CRC_FIFO	CRC data fifo register
0xC4 (SFR)	CRC_DATA0	CRC result data 0 regiter
0xC5 (SFR)	CRC_DATA1	CRC result data 1 regiter
0xC6 (SFR)	CRC_DATA2	CRC result data 2 regiter



0xC7 (SFR)	CRC_DATA3	CRC result data 3 regiter

17.5. 寄存器详细说明

17.5.1. CRC_CON

Addr = 0xC1 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 1	—	—	—	—
0	MODESEL	选择使用 CRC32 还是 CRC8 的功能, CRC8 只需要读取一次结果, CRC32 需要读取 4 次结果, 把结果拼成一个 32bitCRC 结果。 0x0: 使用 CRC8 0x1: 使用 CRC32	RW	0x1

17.5.2. CRC_REG

Addr = 0xC2 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	INITSET	每次使用前需要先设置初始值设置 CRC 校验的初始值, 默认上电的 CRC32 的初始值为 0xffffffff, CRC8 的初始值为 0xff (第 31-8bit 的值不会影响 CRC8 的结果)。	WO	0xFF

Note: 使用 CRC32 模式时, 复位初始值需要写 4 次 CRC_REG 寄存器, CRC8 模式时, 复位初始值只需要写 1 次 CRC_REG 寄存器。

17.5.3. CRC_FIFO

Addr = 0xC3 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	DATA	写入每次需要计算的 1byte 数据	WO	0x0



17.5.4. CRC_DATA0

Addr = 0xC4 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	DATA0	CRC32 的结果的 0-7bit/CRC8 结果	RO	0x0

17.5.5. CRC_DATA1

Addr = 0xC5 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	DATA0	CRC32 的结果的 8-15bit	RO	0x0

17.5.6. CRC_DATA2

Addr = 0xC6 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	DATA0	CRC32 的结果的 16-23bit	RO	0x0

17.5.7. CRC_DATA3

Addr = 0xC7 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	DATA0	CRC32 的结果的 24-31bit	RO	0x0

17.6. 使用流程说明

- 1) 需要配置 CRC_CON 寄存器选择使用 CRC32 或者 CRC8
- 2) 配置 CRC_REG，一般配置为 0xff/0x00 (CRC32 需要四次这个寄存器/CRC8 自需要写一次这个寄存器)；



- 3) 把需要校验的 1byte 数据通过 CRC_FIFO 这个寄存器写进去;
- 4) 需要读取结果时, 读取 CRC_DATA0, CRC_DATA1, CRC_DATA2, CRC_DATA3 拼成一个 CRC32 的结果或者是在使用 CRC8 模式时, 读取 CRC_DATA0 作为 CRC8 的结果。



18. Flash 控制器模块

18.1. 功能概述

- 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/页（1K）擦除全片（16K）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
- 可以配置FLASH存储器的读每 2K加密进行单独加密
- 可以配置FLASH存储器的写/擦每 2K进行单独保护
- 可以支持DMA进行烧写的功能，同时支持自动补齐末尾CRC烧录
- 可以支持代码查空和代码区CRC的功能
- 可以支持在lvd中断打断FLASH的操作
- 可以支持有两个单独的扇区进行全擦/页擦除的保护
- 可以实现切换PC首地址的功能
- 支持类EEPROM的使用

18.2. 基本功能

18.2.1. 读时序配置

表 18-1

系统时钟	FLASH_TIMEREGO (TKH)	FLASH_TIMEREGO (TKP)
48M	0	1
32M	0	1
12M 以下	0	0

当 VCC 的电压在 2.5V 以上时，FLASH 的访问时序，配置为以上参数。

表 18-2

系统时钟	FLASH_TIMEREGO (TKH)	FLASH_TIMEREGO (TKP)
48M	1	3
32M	1	3



12M 以下	0	0

当 VCC 的电压在 2.5V 以下时，FLASH 的访问时序，配置为以上参数。

NOTE: 根据应该场景，需要按照以上参数配置 FLASH 的访问时序。

18.2.2. 写擦保护机制

通过配置 FLASH_FUNCON 寄存器可以对 FLASH 存储体进行写/擦请求的保护，配置方式为每配置 1bit 代表保护 2K 的 FLASH 存储体，建议在实际使用的过程中进行配置，防止意外发生的情况导致 CPU 跑飞从而对 FLASH 存储体进行了误操作。

18.2.3. 自举模式

可以通过配置 FLASH_BOOTCON 设置程序的初始地址，需要通过系统的软复位进行程序的初始地址的切换，例如：当 FLASH_BOOTCON 输入 0x15 的密码，将强制从 0 地址执行程序，输入 0x3A，将强制从 BOOTADDR 配置的区执行程序，BOOTLOAD 可以选择从 0/1K/2K/4K 的程序空间开始执行，但是需要配置 SYS_PEND 的第 2bit 触发其软复位。

18.2.4. 自动计算 CRC

通过配置 FLASH_CRC_LEN 设置 CRC 计算的长度，可以通过硬件自动搬运 FLASH 的内部数据进行 CRC 计算，无需配置初值，每次计算会自动配置初值进行计算。

18.2.5. 类 EEPROM 使用

FLASH支持在程序运行的过程中进行编程/扇区擦除两种操作，在这种使用场景下建议用户设置用户区配置中的代码保护位和FLASH_FUNCON中的代码保护位，以保护主程序代码在这种使用场景下代码的安全性，这种使用场景下，主要有以下两种操作：

- 不使用DMA的类EEPROM场景：

根据使用流程中的配置，配置需要保存的数据，就可以将需要保存的数据烧录到FLASH中，进行数据保护，在这中场景下需要软件进行干预。



- 使用DMA的类EEPROM

将所需要存储的数据提前写到SRAM中，配置数据段的长度，SRAM中存放数据的地址，就可以将SRAM中所需要保存的数据段烧录到FLASH中。

同时，用户如果选择打开FLASH_LOCK中的DMACRCEN将会自动计算数据段的CRC值（使用CRC校验模块中的CRC-MPEG-2 的协议），在烧录完需要保存的数据段后会自动烧录该数据段计算出来的CRC结果，可以节省用户在使用类EEPROM的场景下的代码量，减少用户在这种场景下耗费的时间。

- 在掉电过程中使用类EEPROM

在掉电过程中使用类EEPROM的场景下，用户可以选择打开FLASH_LOCK中LVDCLREN的功能，在这种情况下在LVD事件触发时，会打断当前FLASH的编程/扇区擦除，同时在掉电过程中使用类EEPROM的场景下，建议用户在电源挂载电容，以保证所需要保存的数据能够成功的编程进FLASH中。

注：最好打开用户区配置中的代码保护位和FLASH_FUNCON中的代码保护位。

18.2.6. 支持用户区配置

用户区会在芯片上电的过程中去配置芯片的某些功能，（会在 CPU 复位释放前就配置完成并且生效）这些功能会大大提高芯片的安全性和可靠性，配置功能的表如下图所示：

表 18-3

	功能	长度	描述
3	CRC_LEN_L	[7: 0]	BOOTLOAD 的 CRC 长度，默认为 0，不进行 CRC 校验
4	CRC_LEN_H	[7: 0]	BOOTLOAD 的 CRC 长度，默认为 0，不进行 CRC 校验
5	CODE_PROTECT	[7: 0]	代码加密，每 1bit 代表 2K
6	FUNTION_PROTECT	[7: 0]	代码保护，每 1bit 代表 2K
7	CHECK_EMPTY	[0]	代码查空使能
	HRCOSC_SC_48M	[7: 1]	内部高频 48M 精调：aipcon0[6: 0]，软件配置
8	VREF_TRIM	[4: 0]	ADC 内部基准源校准：pmucon5[4: 0]
	HRCOSC_SC2_48M	[6: 5]	新增 2BIT 用于软件再校准，aipcon1[3: 2]
	reserve	[7]	default: 0
9	CMP_TRIMIB	[5: 0]	CMP 20mA 电流源校准：cmp_anacon0[5: 0]
	HRCOSC_SR_48M	[7: 6]	内部高频 48M 粗调：aipcon1[1: 0]，软件配置
10	MCLR IO MAPO	[0]	0x1: 选择 P23 为 MCLR 功能 pin?



			0x0: P23 没有 MCLR 功能?
	MCLR IO MAP1	[1]	0x1: 选择 P25 为 MCLR 功能 pin? 0x0: P25 没有 MCLR 功能?
	MCLR IO MAP2	[2]	0x1: 选择 P15 为 MCLR 功能 pin? 0x0: P15 没有 MCLR 功能?
	reserve	[7: 3]	
11~14	CRC32	[31: 0]	byte0~byte12 的 CRC32

根据上表，用户可以配置以下的功能

- 代码区自动进行CRC校验：

芯片上电会自动进行代码区CRC校验，若CRC校验失败，则会让CPU不执行代码，防止在一些特殊情况导致代码丢失的情况下，CPU执行错误的代码，带来不可预测的后果。

- 代码区芯片加密：

可以对代码进行读取加密，加密代码后，FLASH内的对应区间的数据无法被读取，保证了用户的代码安全性。

- 代码区编程/擦权限保护：

可以对代码进行编程/擦保护，保护代码后，FLASH内对应的区间将无法被编程和擦除，将大大提高了CPU跑飞后修改了FLASH中的代码，提高了芯片在运行过程中的稳定性和安全性。

18.2.7. NVR 系统信息区域说明

表 18-4

访问字节地址	功能	位长度	描述
0x4004~0x400F	UUID	[95: 0]	每颗芯片 96bit 唯一的身份识别 UUID
0x4010	ADC_OFFSET _IBTRIM	[0]	ADC 失调电流校准（对应配置寄存器 AIP_CON3[7]）
0x4010	ADC_OFFSET _TRIM	[6: 1]	ADC 失调校准（对应配置寄存器 AIP_CON4[5:0]）
0x4010	ADC_CMPBS_ TRIM	[7]	ADC 内部电流校准（对应配置寄存器 AIP_CON4[7]）
0x4011	BGR_TRIM	[2: 0]	PMU 基准校准（对应配置寄存器 PMU_CON3[2:0]）



0x4011	VREF_TRIM	[7: 3]	ADC 内部基准源校准（对应配置寄存器 PMU_CON5[4:0]） 注意：上电硬件自动配置！
0x4012	CMP0_TRIM	[3: 0]	CMP0 失调校准（对应配置寄存器 CMP0_CON4[3:0]）
0x4012	CMP1_TRIM	[7: 4]	CMP1 失调校准（对应配置寄存器 CMP1_CON4[3:0]）
0x4013	CMP_TRIMIB	[5: 0]	CMP 20mA 电流源校准（对应配置寄存器 CMP_CON[5:0]）
0x4014	PGA0_OFFSE T	[7: 0]	PGA0 OFFSET 校准（对应配置寄存器 AMP_CON2[7:0]）
0x4015	PGA1_OFFSE T	[7: 0]	PGA1 OFFSET 校准（对应配置寄存器 AMP_CON4[7:0]）
0x4016	PGA2_OFFSE T	[7: 0]	PGA2 OFFSET 校准（对应配置寄存器 AMP_CON6[7:0]）
0x4017	RC64K_TRIM	[6: 0]	内置 RC64K 校准（对应配置寄存器 PMU_CON4[6:0]）
0x4018	HRCOSC_SC_ 48M	[6: 0]	内部高频 48M 精调对应配置寄存器 CLK_ACON0[6:0]） 注意：上电硬件自动配置！
0x401D	HRCOSC_SR_ 48M	[1: 0]	内部高频 48M 粗调对应配置寄存器 CLK_ACON1[1:0]） 注意：上电硬件自动配置！
0x4040	VBG06_2V_T RIM_3V	[4: 0]	使用 VBG06 ADC VREFP 2V 在 VCC=3V 时的 trim 值 （对应配置寄存器 PMU_CON5[4:0]）
0x4041	VBG06_2.4V _TRIM_3V	[4: 0]	使用 VBG06 ADC VREFP 2.4V 在 VCC=3V 时的 trim 值 （对应配置寄存器 PMU_CON5[4:0]）
0x4042	VBG06_3V_T RIM_3V	[4: 0]	使用 VBG06 ADC VREFP 3V 在 VCC=3V 时的 trim 值 （对应配置寄存器 PMU_CON5[4:0]）
0x4043	VBG06_3.6V _TRIM_3V	[4: 0]	使用 VBG06 ADC VREFP 3.6V 在 VCC=3V 时的 trim 值 （对应配置寄存器 PMU_CON5[4:0]）
0x4044	VBG06_2V_T RIM_4V	[4: 0]	使用 VBG06 ADC VREFP 2V 在 VCC=4V 时的 trim 值 （对应配置寄存器 PMU_CON5[4:0]）
0x4045	VBG06_2.4V _TRIM_4V	[4: 0]	使用 VBG06 ADC VREFP 2.4V 在 VCC=4V 时的 trim 值 （对应配置寄存器 PMU_CON5[4:0]）
0x4046	VBG06_3V_T RIM_4V	[4: 0]	使用 VBG06 ADC VREFP 3V 在 VCC=4V 时的 trim 值 （对应配置寄存器 PMU_CON5[4:0]）
0x4047	VBG06_3.6V _TRIM_4V	[4: 0]	使用 VBG06 ADC VREFP 3.6V 在 VCC=4V 时的 trim 值 （对应配置寄存器 PMU_CON5[4:0]）
0x4048	VBG06_2V_T	[4: 0]	使用 VBG06 ADC VREFP 2V 在 VCC=5V 时的 trim 值



	RIM_5V		(对应配置寄存器 PMU_CON5[4:0])
0x4049	VBG06_2.4V _TRIM_5V	[4: 0]	使用 VBG06 ADC VREFP 2.4V 在 VCC=5V 时的 trim 值 (对应配置寄存器 PMU_CON5[4:0])
0x404A	VBG06_3V_T RIM_5V	[4: 0]	使用 VBG06 ADC VREFP 3V 在 VCC=5V 时的 trim 值 (对应配置寄存器 PMU_CON5[4:0])
0x404B	VBG06_3.6V _TRIM_5V	[4: 0]	使用 VBG06 ADC VREFP 3.6V 在 VCC=5V 时的 trim 值 (对应配置寄存器 PMU_CON5[4:0])

重要说明:

- 所有市场上销售的各个型号的每颗芯片在出厂之前都是经过测试机台校准过, 校准值存储在 NVR 系统信息存储区域对应固定的地址(见上表), 用户程序中只需要通过指针读取出来并配置对应的校准值配置寄存器中, 无需自己在程序中再次校准。
- 发布的 SDK 中对应的系统初始化函数中会读取 芯片各个模块的校准值并配置对应的模拟模块配置寄存器中, 用户可以略过本章节关于校准部分的内容及配置寄存器中关于校准部分的寄存器!
- 具体举例: 如用户需要读取芯片的 96bit 的唯一身份识别 UUID 码, 可以通过以下程序读取:

```
u8 code *nvr_info_ptr;  
u8 UUID[12];  
u8 cnt=0;  
nvr_info_ptr = (u8 code *)0x4004;  
for(cnt=0;cnt<12;cnt++)  
{  
    UUID[cnt] = *nvr_info_ptr++;  
}
```

18.3. 模块框图

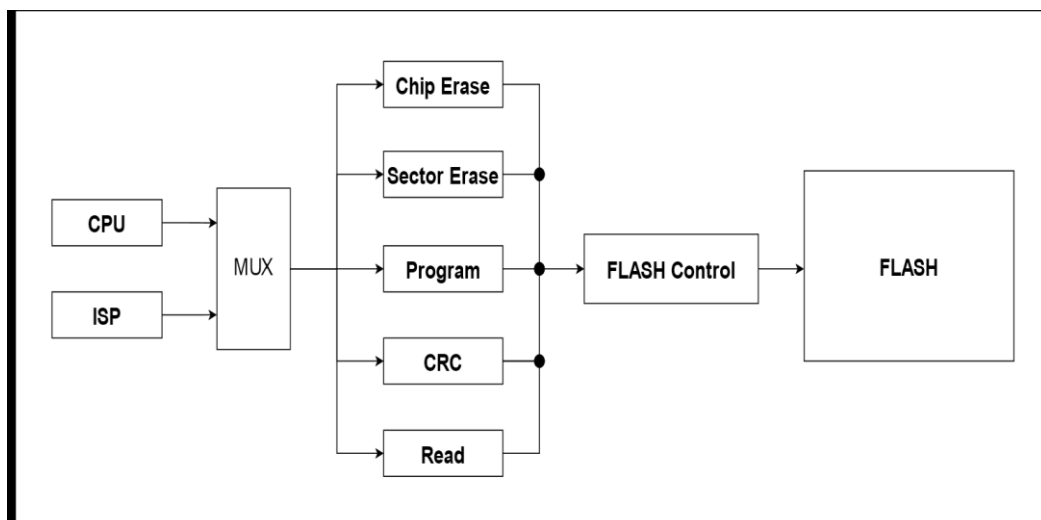




图 18-1 Flash 控制器模块框图

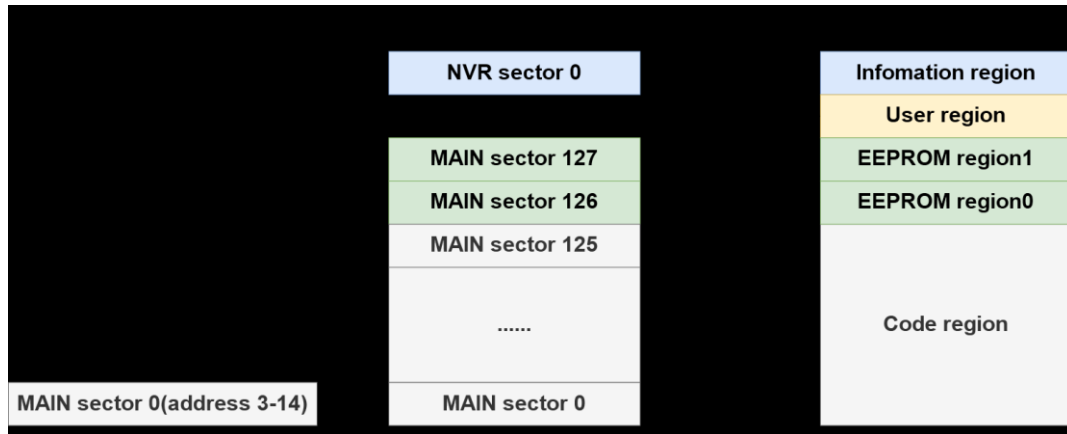


图 18-2 FLASH mapping 框图

18.4. 寄存器列表

表 18-5 FLASH register list

Offset	Register Name	Description
0xA4 (SFR)	FLASH_CON	FLASH control register
0xA5 (SFR)	FLASH_STA	FLASH state register
0xA6 (SFR)	FLASH_DATA	FLASH program data register
0xA7 (SFR)	FLASH_TIMEREG0	FLASH timing control register 0
0xAC (SFR)	FLASH_TIMEREG1	FLASH timing control register 1
0xAD (SFR)	FLASH_CRCLEN	FLASH CRC data length register
0xAE (SFR)	FLASH_PASSWORD	FLASH operation to protect register
0xAF (SFR)	FLASH_ADDR	FLASH program/erase address register
0xBB (SFR)	FLASH_TRIM	FLASH test work mode register
0xBC (SFR)	FLASH_DMASTADR	FLASH dma start address register
0xBD (SFR)	FLASH_DMALEN	FLASH dma length address register
0xC0 (SFR)	FLASH_LOCK	FLASH last 2 sector protect register
0xBE (SFR)	FLASH_BOOTCON	FLASH bootload configuration register
0xBF (SFR)	FLASH_ERRSTA	FLASH error state register
0x199 (XSFR)	FLASH_DEBUGSTA	FLASH debug state register
0xFF (SFR)	FLASH_FUNCON	FLASH funtion protect configuration register



18.5. 寄存器详细说明

18.5.1. FLASH_CON

Addr = 0xA4 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	–	–	–	–
5	DMAST	写 1 触发 DMA 模式烧录	WO	0x0
4	PERST	写 1 触发页擦除	WO	0x0
3	CRCST	写 1 触发 CRC 校验, 需要先配置寄存器 MTP_CRC_LEN 的大小才可以触发该操作	WO	0x0
2	–	–	–	–
1	SERST	写 1 触发扇区擦除	WO	0x0
0	PROGST	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.

18.5.2. FLASH_STA

Addr = 0xA5 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	OVPEND	dma 烧录等于 dma 长度的工作标志, 写 1 清 0 0x0: dma 仍未烧录完成 0x1: dma 烧录完成	RW	0x0
6	HFPEND	dma 烧录等于 dma 长度的一半的工作标志, 写 1 清 0 0x0: dma 烧录仍未烧录到一半 0x1: dma 烧录超过一半	RW	0x0
5	MAINCRCFAIL	代码区域的 CRC 是否通过 0x0: 代码区域 CRC 通过 0x1: 代码区域 CRC 不通过/不开启这	RO	0x0
4	PERPEND	页擦除模式工作标志 0x0: 正在进行页擦除	RO	0x0



		0x1: 空闲状态		
3	CRCPEND	CRC 模式工作标志位 0x0: 正在进行 CRC 校验 0x1: 空闲状态	R0	0x0
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	R0	0x0
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	R0	0x0
0	PROGPEND	烧录模式工作标志 0x0: 正在进行烧录 0x1: 空闲状态	R0	0x0

18.5.3. FLASH_DATA

Addr = 0xA6 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	DATA	FLASH 存储器的数据位宽为 8 位, 在类 EEPROM 的使用场景的时候, 需要将要编程的数据写到该寄存器中	RW	0x0

18.5.4. FLASH_TIMEREG0

Addr = 0xA7 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	TERH	扇区擦除时序控制 (默认时间为 100ms) 0x0: 默认值 0x1: 101ms 0x2: 102ms 0x3: 103ms	RW	0x0
5: 4	TPOAM	后同步信号时序控制信号 (默认时间为 2us) 0x0: 默认值	RW	0x0



		0x1: 3us 0x2: 4us 0x3: 5us		
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 30us） 0x0: 默认值 0x1: 31us 0x2: 32us 0x3: 33us	RW	0x0

18.5.5. FLASH_TIMEREGL

Addr = 0xAC (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	TCERH	全片擦除时序控制信号（默认时间为 100ms） 0x0: 默认值 0x1: 101ms 0x2: 102ms 0x3: 103ms	RW	0x0
5: 4	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
3: 2	TKP	读周期时序控制信号，具体配置看配置表	RW	0x0
1: 0	TKH	读信号高电平时序控制信号，具体配置看配置表	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）。



18.5.6. FLASH_CRCLEN

Addr = 0xAD (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	LEN	CRC 校验操作的数据大小 (单位: 字节) 0x0: 无法工作 0x1: 校验 1byte 0x2: 校验 2byte 0xff: 校验 255byte	RW	0x0

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

18.5.7. FLASH_PASSWORD

Addr = 0xAE (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PASSWORD	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

18.5.8. FLASH_ADDR

Addr = 0xAF (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	ADDR	FLASH 存储器的地址为 15 位, 需要操作的时候需要对该寄存器写两次, 第一次写高 7 位, 第二次写低 8 位。	WO	0x0

Note: 进行 FLASH 的烧录/扇区擦除/页擦除/全片擦除时, 需要将地址对齐, 否则无法进行操作。该地址映射的就为 CPU 的逻辑地址

18.5.9. FLASH_TRIM

Addr = 0xBB (SFR)



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

Note: 正常使用下不用配置这个寄存器（只供测试使用，在非测试模式下使用可能会引发问题）。

18.5.10. FLASH_LOCK

Addr = 0xC0 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	—	—	—	—
4	DMAOVIE	在 DMA 烧录模式下，DMA 完成产生系统中断使能 0x0: 不使能中断 0x1: 使能中断	RW	0x0
3	LVDCLREN	LVD 中断是否打断 FLASH 操作使能 0x0: 不使能该功能 0x1: 使能该功能	RW	0x0
2	DMACRCEN	使能 DMA 操作时自动在烧录代码后添加烧录代码的 CRC 0x0: 关闭功能 0x1: 使能功能	RW	0x0
1	LOCKSN2	使能最后倒数第 1 个 sector 的全擦除和页擦除保护 0x0: 使能保护 0x1: 关闭保护	RW	0x1
0	LOCKSN1	使能最后倒数第二个 sector 的全擦除和页擦除保护 0x0: 使能保护 0x1: 关闭保护	RW	0x1



18.5.11. FLASH_DMASTADR

Addr = 0xBC (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	DMASTADR	FLASH 的 DMA 烧录模式的初始地址 (8byte 对齐)	RW	0x0

18.5.12. FLASH_DMALEN

Addr = 0xBD (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	DMALEN	FLASH 的 DMA 烧录模式的初始地址 (8byte 对齐)	RW	0x0

Note: 当 FLASH_DMA_LEN 的配置值为 0 时, DMA 烧录无法触发。

18.5.13. FLASH_BOOTCON

Addr = 0xBE (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 2	BOOTKEY	输入 0x15, 将强制从 0 地址执行程序, 输入 0x3A, 将强制从 BOOTADDR 配置的区执行程序	WO	0x0
1: 0	BOOTADDR	跳转到代码区的地址 0x0: 默认从 0 开始 0x1: 从 2K 开始 0x2: 从 4K 开始 0x3: 从 6K 开始	RW	0x0

Note: 配置完该寄存器后, 需要执行软复位才可以进行代码区跳转。

18.5.14. FLASH_ERRSTA

Addr = 0xBF (SFR)

Bit(s)	Name	Description	R/W	Reset
7	MAINCRCOK	代码区的 CRC 是否通过	RO	0x0



		0x0: 没有通过/没有开启代码区 CRC 的功能 0x1: 通过代码区 CRC		
6	MTPSUPERKEYFLAG	SUPERKEY 是否有效 0x0: 有效 0x1: 无效	RO	0x1
5	USERNOPASSLOCK	USER 区域的 CRC 不通过的 LOCK 信号 0x0: USER 区域的 CRC 不通过 LOCK 无效 0x1: USER 区域的 CRC 不通过 LOCK 有效	RO	0x0
4	USERLOCK	USER 区域的 CRC 通过的 LOCK 信号 0x0: USER 区域的 CRC 通过 LOCK 无效 0x1: USER 区域的 CRC 通过 LOCK 有效	RO	0x0
3	NVRNOPASSLOCK	NVR 区域的 CRC 不通过的 LOCK 信号 0x0: NVR 区域的 CRC 不通过 LOCK 无效 0x1: NVR 区域的 CRC 不通过 LOCK 有效	RO	0x0
2	NVRLOCK	NVR 区域的 CRC 通过的 LOCK 信号 0x0: NVR 区域的 CRC 通过 LOCK 无效 0x1: NVR 区域的 CRC 通过 LOCK 有效	RO	0x0
1	CHECKEMPTY	代码区查空结果是否为空 0x0: 代码区有代码 0x1: 代码区没有代码	RO	0x0
0	8KMODE	进入了 8K 模式 0x0: 16K 模式 0x1: 8K 模式	RO	0x0

Note: 该寄存器默认值需要会根据出厂的配置不同而不同.

18.5.15. FLASH_DEBUGSTA

Addr = 0x199 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	TIMEOUT	上电过程超时标志 0x0: 上电过程没有超时 0x1: 上电过程超时	RO	0x0
6	FUNCONPROT	寄存器配置写/擦保护开启	RO	0x0



		0x0: 写/擦保护关闭 0x1: 写/擦保护开启		
5	FUNPROT	写/擦保护开启 (USER 区配置) 0x0: 写/擦保护关闭 0x1: 写/擦保护开启	RO	0x0
4	RDPROT	读保护开启 (USER 区配置) 0x0: 读保护关闭 0x1: 读保护开启	RO	0x0
3	PERLOCK	页擦除权限 0x0: 页擦除权限开启 0x1: 页擦除权限关闭	RO	0x0
2	SERLOCK	扇区擦除权限 0x0: 扇区擦除权限开启 0x1: 扇区擦除权限关闭	RO	0x0
1	CERLOCK	全擦权限标志 0x0: 全擦权限开启 0x1: 全擦权限关闭	RO	0x0
0	PROGLOCK	写权限标志 0x0: 写权限开启 0x1: 写权限关闭	RO	0x0

Note: 该寄存器默认值需要会根据出厂的配置不同而不同.

18.5.16. FLASH_FUNCON

Addr = 0xFF (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	FUNCON	每 1bit 代表可以保护 2K 的代码, 保护写/擦	RW	0x0

18.6. 使用流程说明

DMA 烧录模式使用:

- 1) 配置需要操作的 FLASH 的地址
- 2) 提前在对应的 RAM 中初始化需要烧录的数据



3) 配置 FLASH_DMA_STADR 和 FLASH_DMA_LEN 设定 DMA 的首地址和长度, 如果希望在烧录时末尾自动增加 CRC 校验值时, 使能 FLASH_LOCK 中的 DMACRCEN

4) 配置 FLASH_PASSWORD 寄存器使能

5) 配置 FLASH_CON 触发 DMA 操作

6) 等待对应的 FLASH_STA 中的 OVPEND 为高结束一次 DMA 操作

普通 FLASH 操作:

1) 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)

2) 配置需要操作的 FLASH 的数据

3) 配置 FLASH_PASSWORD 寄存器使能

4) 配置 FLASH_CON 触发对应的 FLASH 操作

5) 等 FLASH_STA 对应的 pending 为高, 结束一次 FLASH 操作

19. 模数转换器(ADC)

19.1. 功能概述

- ADC具有 12BIT转换精度
- 高达 500KSPS的转换速度
- 支持 26 路IO的单端模拟输入通道
- 支持外部参考电压/内部参考电压作为ADC的参考电压
- 内部参考电压为 2.0/2.4/3.0/3.6/4.2 可选, 只有 2.4V有校准, 且在VCC=5V条件下
- 外部电压默认为VCC
- ADC通过通路选择器可以实对现所有IO口输入电压进行数据转换
- 采样时间可调, 可调范围: 5~256 个ADC时钟
- 支持 3 通道仲裁触发 (可带DLY功能触发)
- 支持 15 路硬件触发源和 1 路软件触发
- 支持硬件自动切换通道, 提高连续采样的速度
- 比较器offset可以校准
- 可以通过ADC的通路选择把Analog测试信号, 输入到任意I/O口



19.2. 基本功能

19.2.1. 外部触发源

ADC 外部触发源头来自于 SUPER TIMER/TIMER0/IO 外部触发。

1) SUPER TIMER 的触发源来自于 SUPER TIMER0/2/4

- PWM 上升沿 (SUPER TIMER0/2/4 PWM 上升沿)：SUPER TIMER 的输出到 IO 的 PWM 上升沿。
- PWM 下降沿 (SUPER TIMER0/2/4 PWM 下降沿)：SUPER TIMER 的输出到 IO 的 PWM 下降沿。
- 周期点 (SUPER TIMER0/2/4 PWM 周期点)：SUPER TIMER 模块中 SEMRn_IF (n=0/2/4) 的第 0 位，所以在触发完一次后，需要手动清除标志位，才可以重新触发第二次。（具体描述可以看 SUPER TIMER 模块描述）
- 零点 (SUPER TIMER0/2/4 PWM 零点)：SUPER TIMER 模块中 SEMRn_IF (n=0/2/4) 的第 1 位，所以在触发完一次后，需要手动清除标志位，才可以重新触发第二次。（具体描述可以看 SUPER TIMER 模块描述）
- SUPER TIMER0 的 C 点 (SUPER TIMER0 C 点)：需要配置 STMR_CMPL 和 STMR_CMPH，当 SUPER TIMER0 的计数值等于配置 STMR_CMPL 和 STMR_CMPH 的值时，会触发 ADC 进行采样。（具体描述可以看 SUPER TIMER 模块描述）

2) TIMER0 的触发源 (TIMER0 降采样触发)

- TIMER0 发生 OVERFLOW 事件时会触发 ADC 进行转换。（TIMER0 不与 SUPER TIMER 进行联动时）
- TIMER0 与 SUPER TIMER 进行联动时，具体的 TIMER0 计数源可以看 SIMPLE TIMER 模块描述。

3) GPIO 触发源 (GPIO 外部触发)

- 需要将 IO 配置成输入模式，同时配置 FIN_S15，选择对应的 IO，IO 如果出现上升沿变化则会触发 ADC 转换。

19.2.2. 内部采样通道描述

ADC 内部采样通道有 5 个采样通道，分别是芯片内部参考 0.6V (VREF_0P6)，芯片电源电压的 5 分压值 (VCCA_5D)，运放 0/1/2 (AMP0/1/2) 的输出电压。

19.2.3. 单通道触发模式

单通道触发模式是只使能 ADC_CFG0 中的 CHAN0EN/CHAN1EN/CHAN2EN, 如果只使能通道 0 时, 只有配置的 ADC_TRG0 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 1/2 同理, 对应使用 ADC_TRG1/2 的触发源和 ADC_CHS1/2 的转换通道。如下图所示:

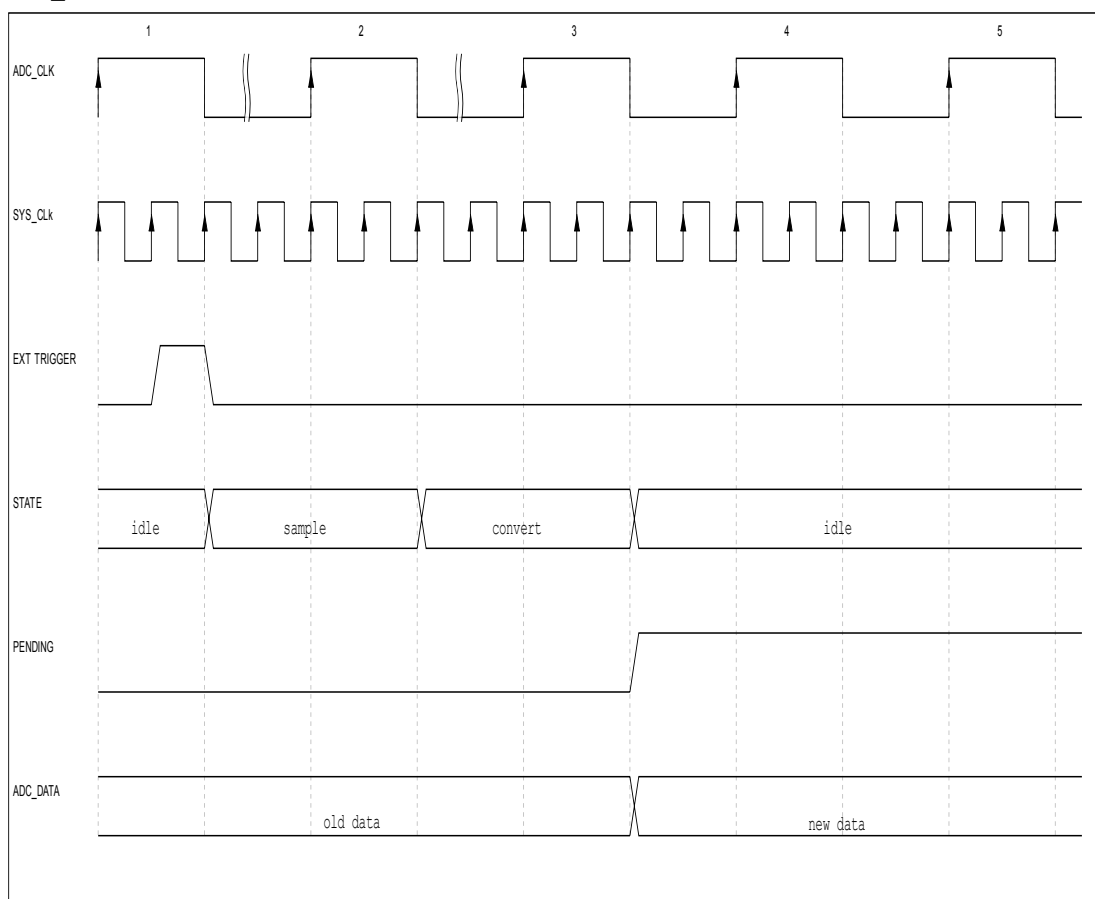


图 19-1 单通道触发模式时序图

19.2.4. 多通道触发模式

多通道触发模式是同时使能 ADC_CFG0 中 CHAN0EN/CHAN1EN 或 CHAN1EN/CHAN2EN 或 CHAN0EN/CHAN2EN 或 CHAN0EN/CHAN1EN/CHAN2EN, 如果当前只使能通道 0 和 1, 即 CHAN0EN 和 CHAN1EN 同时使能, 而 CHAN2EN 不使能, 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_TRG0 和 ADC_TRG1 中的配置的触发源在同一时刻触发 ADC 时, 会先响应 ADC_TRG0 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。如下图所示：

注：通道的优先级：通道 0>通道 1>通道 2

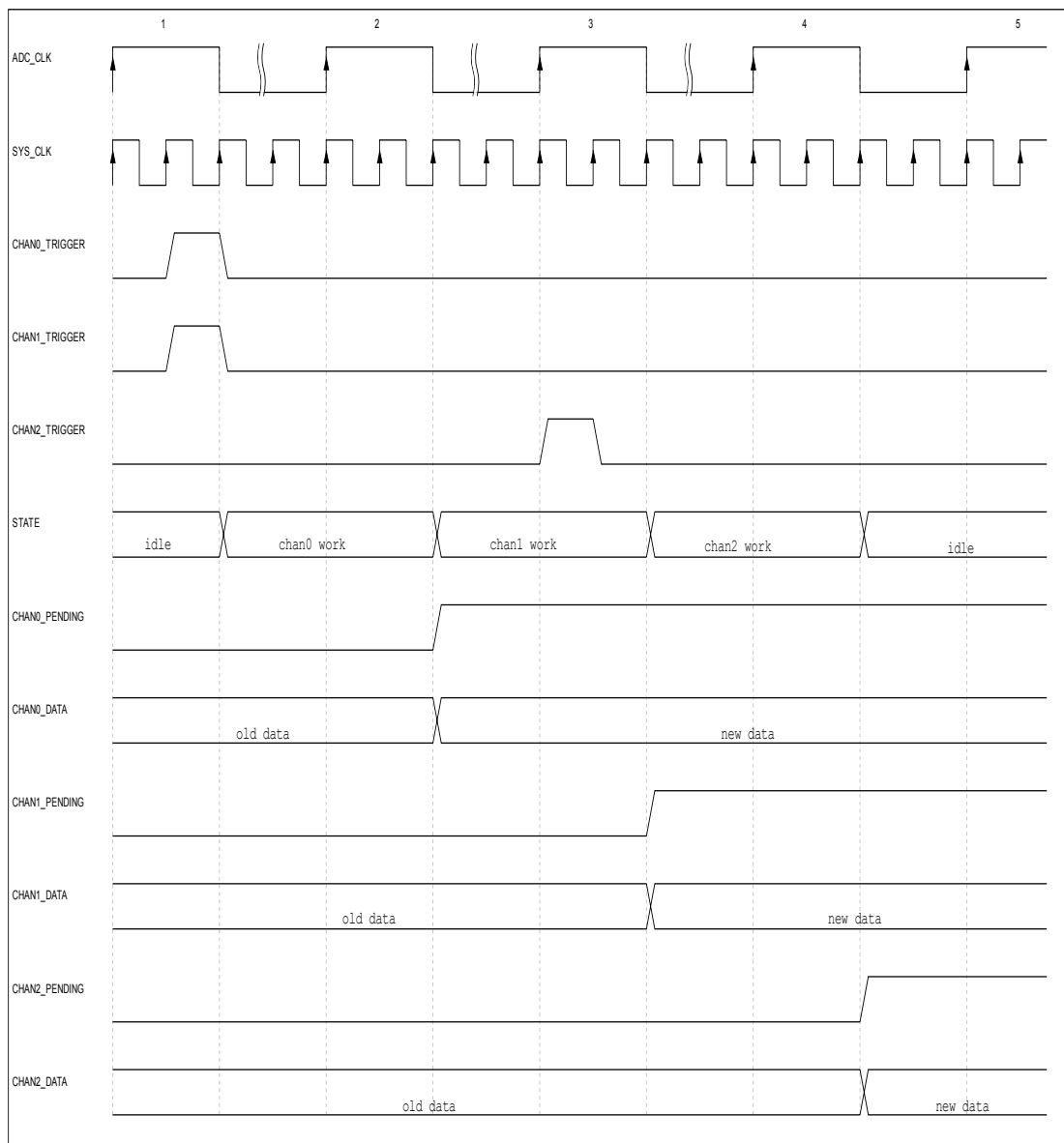


图 19-2 多通道触发模式时序图

19.2.5. 触发延迟模式

触发延迟模式在通道使能的基础上加上触发延迟的功能

CHANOEN&DLYOEN/CHAN1EN&DLY1EN/CHAN2EN&DLY2EN,在这种模式下当通道 0 在 ADC_TRG0/1/2 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_TRG0/1/2 中配置的 DLYCYC 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 的三个通道有仲裁机制,可以同时开启触发延迟模式和多通道触发模式。

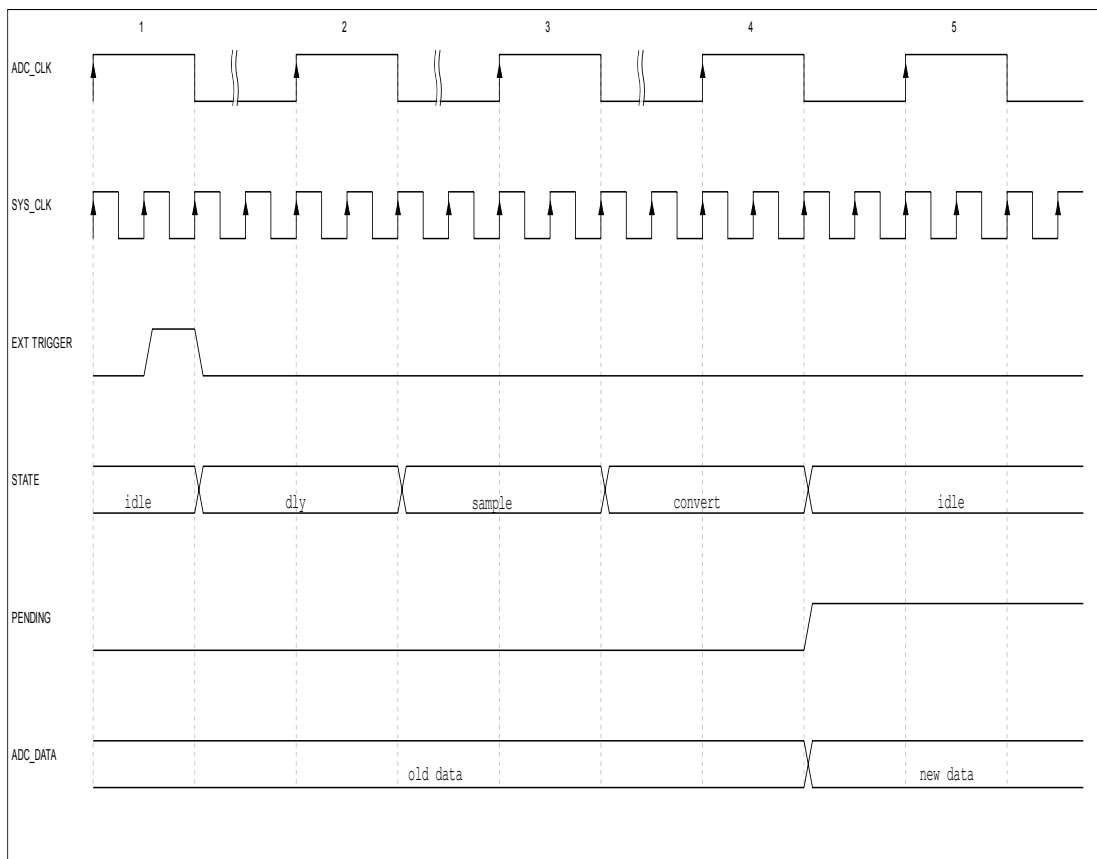


图 19-3 触发延迟模式时序图

19.2.6. 加速模式

当 ADC_CMPDATAL 的 SPEEDMODEEN 使能后,当通道 0 和 1 同时选择同一个触发源时,硬件将会在通道 0 转换完后马上进行通道 1 的转换,同时会将采样通道提前进行切换,以满足通道的建立时间,提高连续采样的采样速度。

注:在使能这种模式时,一定要将通道 0/1 配置成相同的触发源,否则会出现不可预测的后果。

19.2.7. 数字比较器

当配置数字比较器的比较值时,需要配置 ADC_CMPDATAH 和 ADC_CMPDATAL 的 CMPDATAH 和 CMPDATAL 时,同时使能 CMPEN0/CMPEN1/CMPEN2 时,对应通道的 ADC 转换结果会和设置的

比较值进行比较，但是三个通道共用同一个比较值，同时当转换的结果大于比较值时会使 ADC_STA 中的 BRKADC 的标志位为 1，此时在 Super timer 中有对应的硬件的刹车使能位。

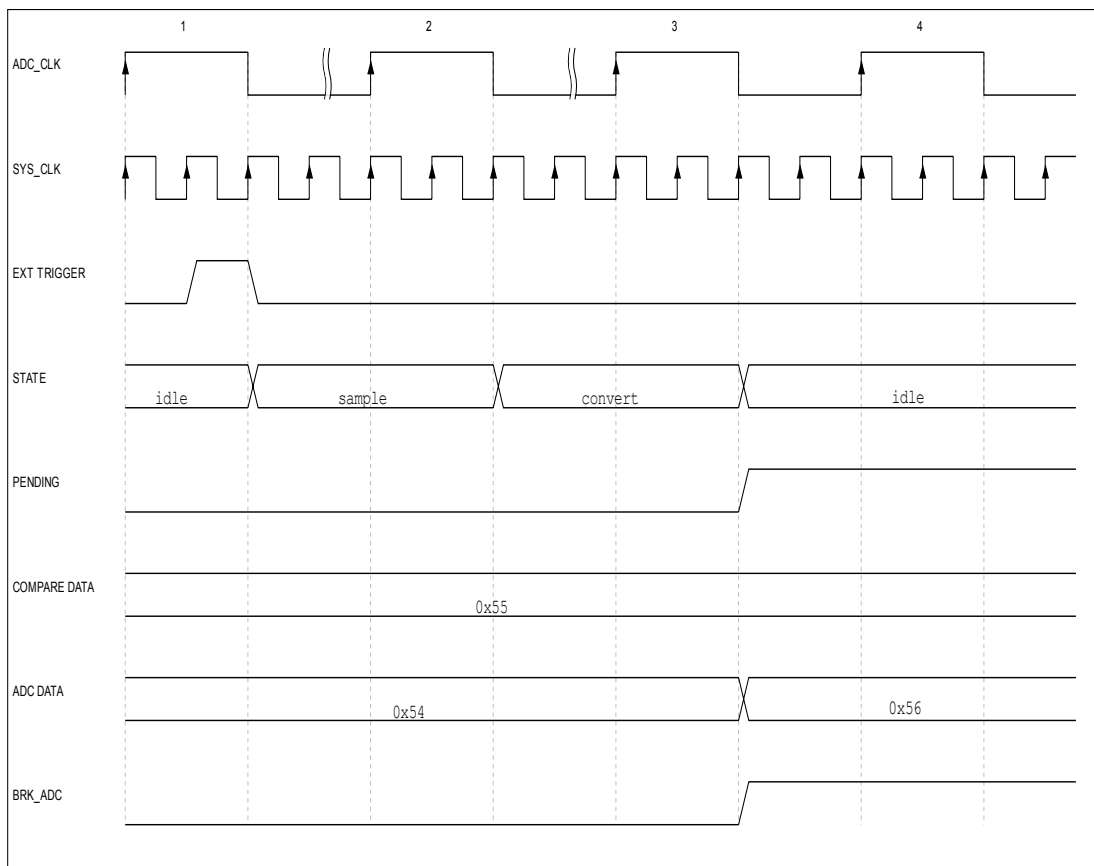


图 19-4 数字比较器模式工作时序图

19.2.8. 模拟校准/数字校准

NOTE: 所有市场上销售的各个型号的每颗芯片在出厂之前都是经过测试机台校准过，校准值存储在 NVR 存储器固定的地址，用户程序中只需要通过指针读取出来并配置对应的校准值寄存器中，无需自己在程序中再次校准。发布的 SDK 中对应的系统初始化函数中会读取芯片各个模块的校准值并配置对应的模拟模块配置寄存器中，用户可以略过本章节关于校准部分的内容及配置寄存器中关于校准部分的寄存器！

- 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，并且清除 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。

19.3. 模块框图

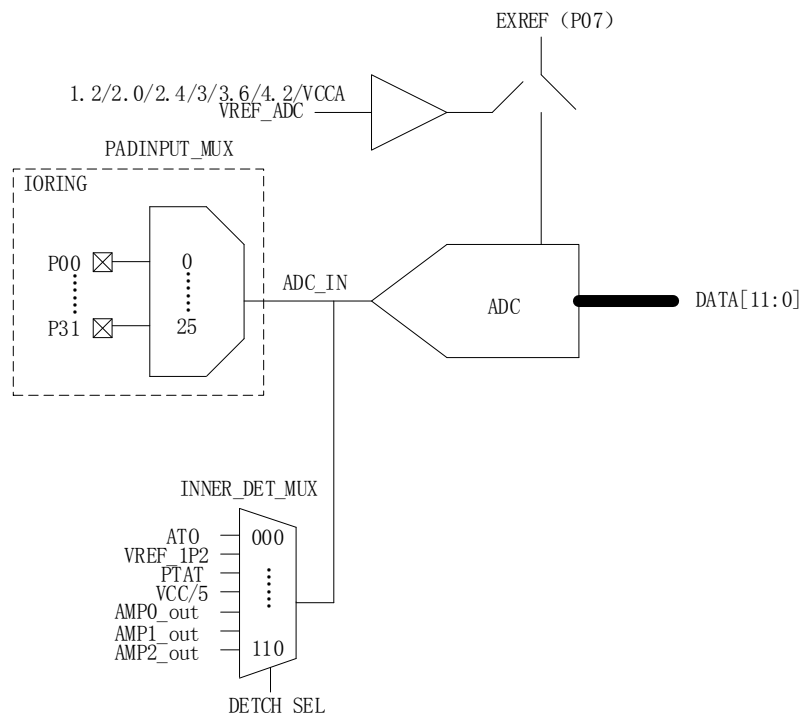


图 19-5 ADC 模块结构框图

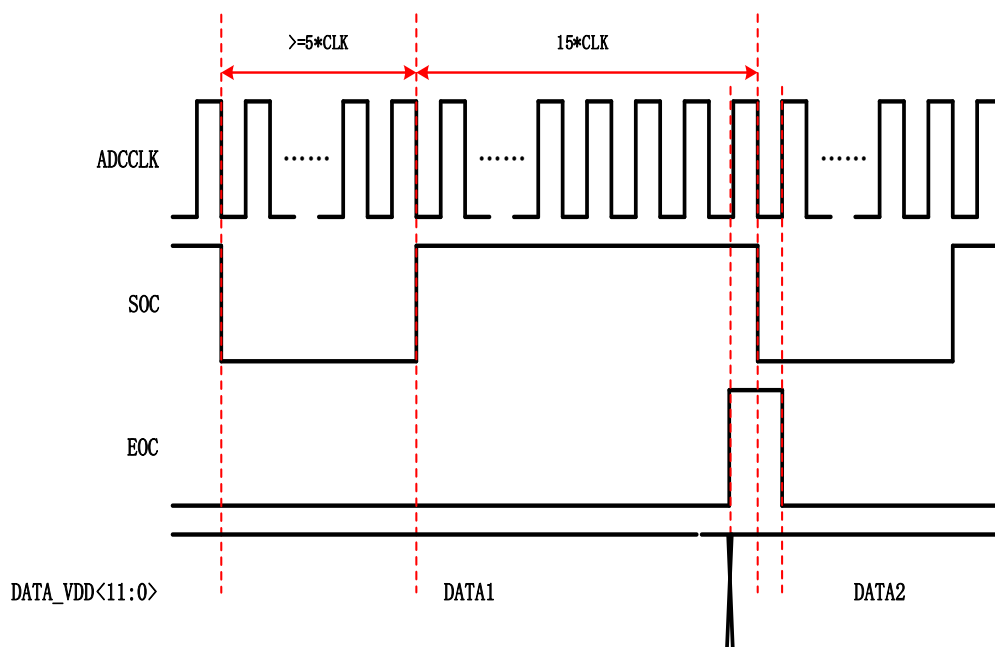


图 19-6 ADC 转换时序图

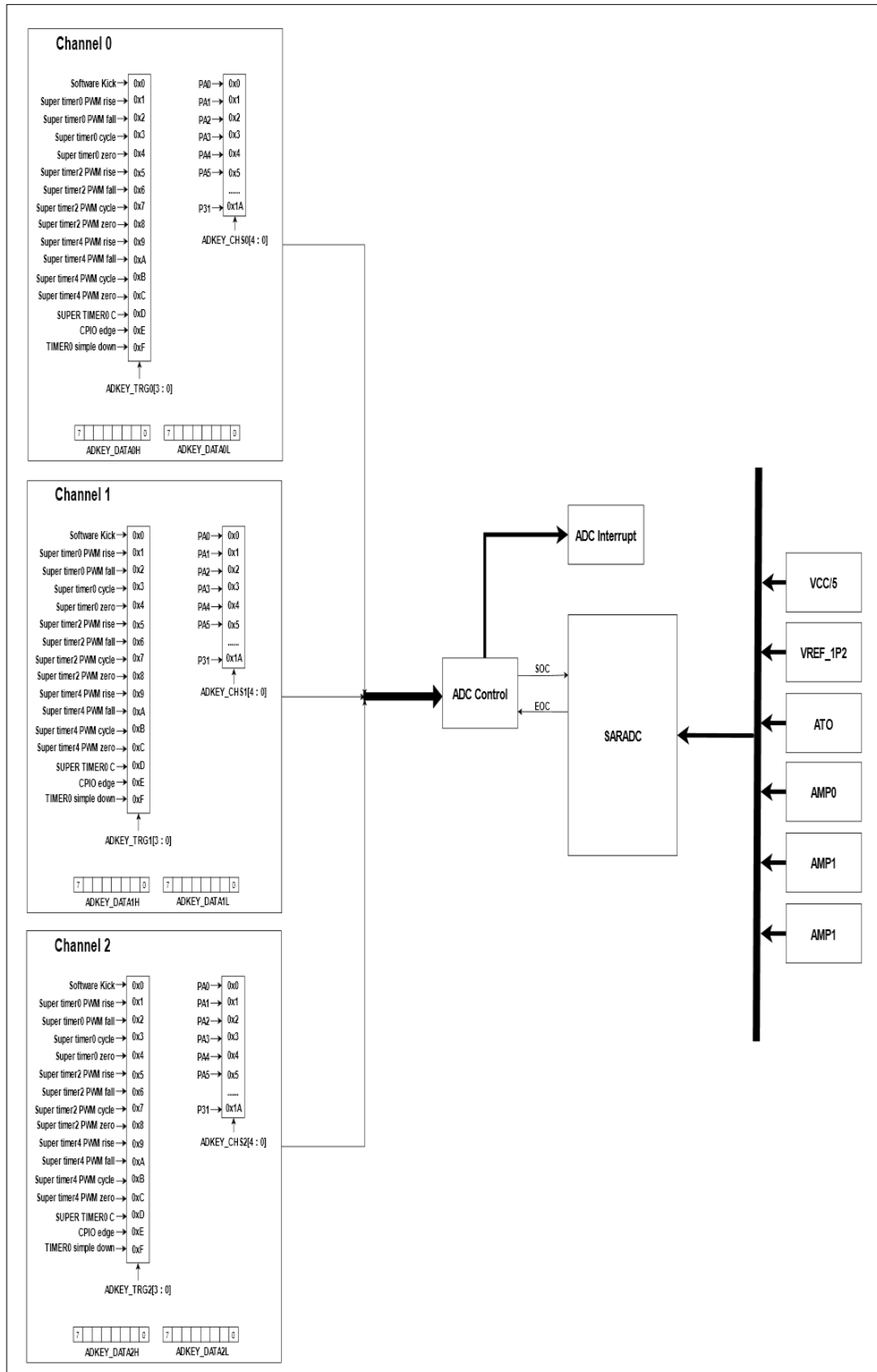


图 19-7 模块结构框图



19.4. 寄存器列表

表 19-1 ADC register list

Offset	Register Name	Description
0x69 (XSFR)	ADC_CFG0	ADC configuration 0 register
0x6A (XSFR)	ADC_CFG1	ADC configuration 1 register
0x91 (SFR)	ADC_CFG2	ADC configuration 2 register
0x92 (SFR)	ADC_CFG3	ADC configuration 3 register
0xFC (SFR)	ADC_CFG4	ADC configuration 4 register
0x6B (XSFR)	ADC_STA	ADC state register
0x93 (SFR)	ADC_DATAH0	ADC channel0 data high 8bit register
0x94 (SFR)	ADC_DATALO	ADC channel0 data low 4bit register
0x95 (SFR)	ADC_DATAH1	ADC channel1 data high 8bit register
0x96 (SFR)	ADC_DATA11	ADC channel1 data low 4bit register
0x97 (SFR)	ADC_DATAH2	ADC channel2 data high 8bit register
0x98 (SFR)	ADC_DATA22	ADC channel2 data low 4bit register
0x99 (SFR)	ADC_CHS0	ADC channel0 select register
0x9A (SFR)	ADC_CHS1	ADC channel1 select register
0x9D (SFR)	ADC_CHS2	ADC channel2 select register
0x9E (SFR)	ADC_TRGS0	ADC trigger0 select register
0x9F (SFR)	ADC_TRGS1	ADC trigger1 select register
0xA1 (SFR)	ADC_TRGS2	ADC trigger2 select register
0xA2 (SFR)	ADC_CMPDATAH	ADC digital compare data register
0xA3 (SFR)	ADC_CMPDATAL	ADC digital compare data register

19.5. 寄存器详细说明

19.5.1. ADC_CFG0

Addr = 0x69 (XSFR)

Bit(s))	Name	Description	R/W	Reset



7	CALIBEN	开启硬件校准，开启后的通道 0 低 6bit 作为校准结果， 填回去 ADC_ACON2 的 ADCCMPTRIM	RW	0x0
6	ADCEN	A/D 转换使能（ADC 使能） 0x0：关闭 0x1：使能	RW	0x0
5	CHAN2EN	通道 2 转换使能 0x0：关闭 0x1：使能	RW	0x0
4	CHAN1EN	通道 1 转换使能 0x0：关闭 0x1：使能	RW	0x0
3	CHAN0EN	通道 0 转换使能 0x0：关闭 0x1：使能	RW	0x0
2	ADST2	ADC 通道 2 触发转换 写 1 开始触发通道 3 进行转换	WO	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

19.5.2. ADC_CFG1

Addr = 0x6A (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	CPCALIBST	写 1 触发数字校准功能	WO	0x0
6: 3	ADCPRE	ADC 时钟分频，分频比为 n+1，3 分频以下无法正常工作， 需要配置大于 3 分频，ADC 输入时钟最大为 10MHz， Fadc=Fsys/ADCPRE，其中 Fadc 为 ADC 输入时钟，Fsys 为系统时钟； Note: ADC 输入时钟超过 10MHz 会导致采样不准！ 0x3: 4 分频	RW	0x4



		0x4: 5 分频 0x5: 6 分频 0xf: 16 分频		
2	CHAN2IE	ADC 通道 2 的中断使能 0x0: 中断不使能 0x1: 中断使能	RW	0x0
1	CHAN1IE	ADC 通道 1 的中断使能 0x0: 中断不使能 0x1: 中断使能	RW	0x0
0	CHAN0IE	ADC 通道 0 的中断使能 0x0: 中断不使能 0x1: 中断使能	RW	0x0

19.5.3. ADC_CFG2

Addr = 0x91 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	FSMPCYC0	通道 0 采样时间配置, 配置比为 n+1 时钟 0x0: 1 个采样时钟周期 0x1: 2 个采样时钟周期 0x2: 3 个采样时钟周期 0xff: 256 个采样时钟周期	RW	0x4

Note: 为了 ADC 能够正常工作, 采样时间最少配置为 4.

19.5.4. ADC_CFG3

Addr = 0x92 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	FSMPCYC1	通道 1 采样时间配置, 配置比为 n+1 时钟 0x0: 1 个采样时钟周期	RW	0x4



		0x1: 2 个采样时钟周期 0x2: 3 个采样时钟周期 0xff: 256 个采样时钟周期		

Note: 为了 ADC 能够正常工作, 采样时间最少配置为 4。

19.5.5. ADC_CFG4

Addr = 0xFC (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	FSMPCYC2	通道 2 采样时间配置, 配置比为 n+1 时钟 0x0: 1 个采样时钟周期 0x1: 2 个采样时钟周期 0x2: 3 个采样时钟周期 0xff: 256 个采样时钟周期	RW	0x4

Note: 为了 ADC 能够正常工作, 采样时间最少配置为 4。

19.5.6. ADC_STA

Addr = 0x6B (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	CMPOUT	ADC 中模拟比较器输出, 用于校准	RO	0x0
6	BRKADC	ADC 刹车信号标志, 写 1 清 0 0x0: ADC 没有触发刹车事件 0x1: ADC 触发刹车事件	RW	0x0
5	EODTADCFLAG	数字校准完成标志, 写 1 清 0 0x0: 数字校准没开始或者没完成 0x1: 数字校准完成	RW	0x0
4	EOATADCFLAG	模拟校准完成标志, 写 1 清 0 0x0: 模拟校准没开始或者没完成 0x1: 模拟校准完成	RW	0x0



3	CHAN2OVPEND	通道 2 转换结束，写 1 清 0 0x0: 通道 2 转换没开始或者没完成 0x1: 通道 2 转换完成	RW	0x0
2	CHAN1OVPEND	通道 1 转换结束，写 1 清 0 0x0: 通道 1 转换没开始或者没完成 0x1: 通道 1 转换完成	RW	0x0
1	CHAN0OVPEND	通道 0 转换结束，写 1 清 0 0x0: 通道 0 转换没开始或者没完成 0x1: 通道 0 转换完成	RW	0x0
0	BUSY	ADC 忙/空闲 0x0: ADC 空闲 0x1: ADC 转换中	RO	0x0

19.5.7. ADC_DATAH0

Addr = 0x93 (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

19.5.8. ADC_DATA0L

Addr = 0x94 (SFR)

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

19.5.9. ADC_DATAH1

Addr = 0x95 (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



19.5.10. ADC_DATA1

Addr = 0x96 (SFR)

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

19.5.11. ADC_DATAH2

Addr = 0x97 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	DATA2H	通道 2 的 12 位 A/D 转换结果高 8 位	RW	0x0

19.5.12. ADC_DATA2

Addr = 0x98 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 4	DATA2L	通道 2 的 12 位 A/D 转换结果低 4 位	RW	0x0
3: 0	—	—	—	—

19.5.13. ADC_CHS0

Addr = 0x99 (SFR)

Bit(s)	Name	Description	R/W	Reset
7	FRMTCHAN0	通道 0 数据格式控制 0x0: 左对齐 0x1: 右对齐	RW	0x0
6	DLYOEN	通道 0 DLY 功能使能 0x0: 关闭 0x1: 使能	RW	0x0
5	CHANOEXT	内外部通道选择	RW	0x0



		0x0: 外部通道 0x1: 内部通道		
4: 0	CHANSELO	通道 0 的模拟转换通道选择 当 CHANOEXT 为 0 时, CHANSELO 的模拟通道选择为 GPIO, 对应关系为: 0x0: P00 0x1: P01 0x2: P02 0x3: P03 0x4: P04 0x5: P05 0x6: P06 0x7: P07 0x8: P10 0x9: P11 0xA: P12 0xB: P13 0xC: P14 0xD: P15 0xE: P16 0xF: P17 0x10: P20 0x11: P21 0x12: P22 0x13: P23 0x14: P24 0x15: P25 0x16: P26 0x17: P27 0x18: P30 0x19: P31 当 CHANOEXT 为 1 时, CHANSELO 的模拟通道对应选择为内部通道, 对应关系为: 0x0: 保留, 未定义	RW	0x1F



		0x1: VREF_OP6 0x2: 保留, 未定义 0x3: VCCA_D5 0x4: AMP0 0x5: AMP1 0x6: AMP2 0x7: 不使能 Note: 采样内部通道时需要将 ADC_ACON0[6: 4] 设置成 0 采样内部通道时需要将 ADC_ACON0[6: 4] 设置成 0 VREF_OP6: 芯片内部 0.6V 参考源, 与比较器中 DAC 的参考源相同 VCCA_D5: 芯片的 VCC 电源电压的 5 分压值 AMP0: 运放 0 输出电压 AMP1: 运放 1 输出电压 AMP2: 运放 2 输出电压		

19.5.14. ADC_CHS1

Addr = 0x9A (SFR)

Bit(s)	Name	Description	R/W	Reset
7	FRMTCHAN1	通道 1 数据格式控制 0x0: 左对齐 0x1: 右对齐	RW	0x0
6	DLY1EN	通道 1DLY 功能使能 0x0: 关闭 0x1: 使能	RW	0x0
5	CHAN1EXT	内外部通道选择 0x0: 外部通道 0x1: 内部通道	RW	0x0
4: 0	CHANSEL1	通道 1 的模拟转换通道选择 当 CHAN1EXT 为 0 时, CHANSEL1 的模拟通道选择为 GPIO, 对应关系为:	RW	0x1F



		0x0: P00 0x1: P01 0x2: P02 0x3: P03 0x4: P04 0x5: P05 0x6: P06 0x7: P07 0x8: P10 0x9: P11 0xA: P12 0xB: P13 0xC: P14 0xD: P15 0xE: P16 0xF: P17 0x10: P20 0x11: P21 0x12: P22 0x13: P23 0x14: P24 0x15: P25 0x16: P26 0x17: P27 0x18: P30 0x19: P31 当 CHAN1EXT 为 1 时, CHANSEL1 的模拟通道对应选择为内部通道, 对应关系为: 0x0: 保留, 未定义 0x1: VREF_OP6 0x2: 保留, 未定义 0x3: VCCA_D5 0x4: AMP0 0x5: AMP1		



		0x6: AMP2 0x7: 不使能 Note: 采样内部通道时需要将 ADC_ACON0[6: 4] 设置成 0 采样内部通道时需要将 ADC_ACON0[6: 4] 设置成 0 VREF_OP6: 芯片内部 0.6V 参考源, 与比较器中 DAC 的参考源相同 VCCA_D5: 芯片的 VCC 电源电压 5 分压值 AMP0: 运放 0 输出电压 AMP1: 运放 1 输出电压 AMP2: 运放 2 输出电压		

19.5.15. ADC_CHS2

Addr = 0x9D (SFR)

Bit(s)	Name	Description	R/W	Reset
7	FRMTCHAN2	通道 2 数据格式控制 0x0: 左对齐 0x1: 右对齐	RW	0x0
6	DLY2EN	通道 2DLY 功能使能 0x0: 关闭 0x1: 使能	RW	0x0
5	CHAN2EXT	内外部通道选择 0x0: 外部通道 0x1: 内部通道	RW	0x0
4: 0	CHANSEL2	通道 2 的模拟转换通道选择 当 CHAN2EXT 为 0 时, CHANSEL2 的模拟通道选择为 GPIO, 对应关系为: 0x0: P00 0x1: P01 0x2: P02 0x3: P03 0x4: P04	RW	0x1F



		0x5: P05 0x6: P06 0x7: P07 0x8: P10 0x9: P11 0xA: P12 0xB: P13 0xC: P14 0xD: P15 0xE: P16 0xF: P17 0x10: P20 0x11: P21 0x12: P22 0x13: P23 0x14: P24 0x15: P25 0x16: P26 0x17: P27 0x18: P30 0x19: P31 当 CHAN2EXT 为 1 时, CHANSEL2 的模拟通道对应选择为内部通道, 对应关系为: 0x0: 保留, 未定义 0x1: VREF_OP6 0x2: 保留, 未定义 0x3: VCCA_D5 0x4: AMP0 0x5: AMP1 0x6: AMP2 0x7: 不使能 Note: 采样内部通道时需要将 ADC_ACON0[6: 4] 设置成 0 VREF_OP6: 芯片内部 0.6V 参考源, 与比较器中 DAC		



		的参考源相同 VCCA_D5: 芯片的 VCC 电源电压 5 分压值 AMP0: 运放 0 输出电压 AMP1: 运放 1 输出电压 AMP2: 运放 2 输出电压		

19.5.16. ADC_TRGS0

Addr = 0x9E (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 4	DLYCYC0	通道 0DLY 的 ADC 时钟个数选择, 配置为 4n+1 0x0: 1 个 ADC 时钟周期 0x1: 5 个 ADC 时钟周期 0x2: 9 个 ADC 时钟周期 0x7: 29 个 ADC 时钟周期	RW	0x0
3: 0	TRGSEL0	通道 0 硬件触发源选择 0x0: 选择软件触发 0x1: SUPER TIMER0 PWM 上升沿 0x2: SUPER TIMER0 PWM 下降沿 0x3: SUPER TIMER0 周期点 0x4: SUPER TIMER0 零点 0x5: SUPER TIMER2 PWM 上升沿 0x6: SUPER TIMER2 PWM 下降沿 0x7: SUPER TIMER2 周期点 0x8: SUPER TIMER2 零点 0x9: SUPER TIMER4 PWM 上升沿 0xA: SUPER TIMER4 PWM 下降沿 0xB: SUPER TIMER4 周期点 0xC: SUPER TIMER4 零点 0xD: SUPER TIMER0 C 点 0xE: GPIO 外部触发 0xF: TIMER0 降采样触发	RW	0x0



特别注意：当选择零点/周期点时，需要清除对应的标志位，具体请看外部触发源描述！！

19.5.17. ADC_TRGS1

Addr = 0x9F (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 4	DLYCYC1	通道 1DLY 的 ADC 时钟个数选择，配置为 $4n+1$ 0x0: 1 个 ADC 时钟周期 0x1: 5 个 ADC 时钟周期 0x2: 9 个 ADC 时钟周期 0x7: 29 个 ADC 时钟周期	RW	0x0
3: 0	TRGSEL1	通道 1 硬件触发源选择 0x0: 选择软件触发 0x1: SUPER TIMERO PWM 上升沿 0x2: SUPER TIMERO PWM 下降沿 0x3: SUPER TIMERO 周期点 0x4: SUPER TIMERO 零点 0x5: SUPER TIMER2 PWM 上升沿 0x6: SUPER TIMER2 PWM 下降沿 0x7: SUPER TIMER2 周期点 0x8: SUPER TIMER2 零点 0x9: SUPER TIMER4 PWM 上升沿 0xA: SUPER TIMER4 PWM 下降沿 0xB: SUPER TIMER4 周期点 0xC: SUPER TIMER4 零点 0xD: SUPER TIMERO C 点 0xE: GPIO 外部触发 0xF: TIMERO 降采样触发 特别注意：当选择零点/周期点时，需要清除对应的标志位，具体请看外部触发源描述！！	RW	0x0



19.5.18. ADC_TRGS2

Addr = 0xA1 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 4	DLYCYC2	通道 2DLY 的 ADC 时钟个数选择，配置为 $4n+1$ 0x0: 1 个 ADC 时钟周期 0x1: 5 个 ADC 时钟周期 0x2: 9 个 ADC 时钟周期 0x7: 29 个 ADC 时钟周期	RW	0x0
3: 0	TRGSEL2	通道 2 硬件触发源选择 0x0: 选择软件触发 0x1: SUPER TIMER0 PWM 上升沿 0x2: SUPER TIMER0 PWM 下降沿 0x3: SUPER TIMER0 周期点 0x4: SUPER TIMER0 零点 0x5: SUPER TIMER2 PWM 上升沿 0x6: SUPER TIMER2 PWM 下降沿 0x7: SUPER TIMER2 周期点 0x8: SUPER TIMER2 零点 0x9: SUPER TIMER4 PWM 上升沿 0xA: SUPER TIMER4 PWM 下降沿 0xB: SUPER TIMER4 周期点 0xC: SUPER TIMER4 零点 0xD: SUPER TIMER0 C 点 0xE: GPIO 外部触发 0xF: TIMER0 降采样触发 特别注意：当选择零点/周期点时，需要清除对应的标志位，具体请看外部触发源描述!!!	RW	0x0

19.5.19. ADC_CMPDATAH

Addr = 0xA2 (SFR)



Bit(s)	Name	Description	R/W	Reset
7: 0	CMPDATAH	数字比较器刹车的预设值，高 8bit	RW	0x0

19.5.20. ADC_CMPDATAH

Addr = 0xA3 (SFR)

Bit(s)	Name	Description	R/W	Reset
7: 4	CMPDATAH	数字比较器刹车的预设值，低 4bit 当转换结果大于数字比较器刹车时，会触发刹车事件	RW	0x0
3	SPEEDMODEEN	加速模式，当通道 0 和通道 1 都选择同一个触发源的时候，硬件会自动切换通道，提前保证通道的建立时间能够满足。 0x0: 关闭 0x1: 使能	RW	0x0
2	CMPEN2	通道 2 使能数字比较功能 0x0: 关闭 0x1: 使能	RW	0x0
1	CMPEN1	通道 1 使能数字比较功能 0x0: 关闭 0x1: 使能	RW	0x0
0	CMPEN0	通道 0 使能数字比较功能 0x0: 关闭 0x1: 使能	RW	0x0

Note: 只有一个预设值，所以当使能两个或以上的通道数字刹车时，只有一个预设值，所有通道共用一个预设值。

19.6. 使用流程说明

- 1) 配置 ADC_CFG1 的 ADCPRE，设定 ADC 时钟分频
- 2) 配置 ADC_CFG0 使能通道转换



- 3) 配置 ADC_CFG2 配置采样时间
- 4) 配置 ADC_CHS0/ADC_CHS1 配置模拟转换通道和触发源
- 5) 配置 ADC_CFG0 使能 ADC
- 6) 等待 20us (等待期间不能触发 ADC 转换)
- 7) 触发 ADC
- 8) 等待 ADC_STA 中对应的通道的 pending 为高
- 9) ADC_DATAH0/ADC_DATAH1 的转换结果

20. 模拟比较器 (CMP0/1)

20.1. 功能概述

- 模块功能结构
 - 主要包括 2 个 8 bit flash DAC, 2 个比较器, 1 路 20mA 恒流源输出
- 比较器正端和负端可选多种输入通路
 - 每个比较器正端可选择 6 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
 - 比较器 0 正端还支持阈值短路保护输入
 - 比较器 1 支持 CCS 采样电压输入
- 参考电压可选
 - DAC 的参考电压可选择内部 1.2V 参考或者 VCCA 参考, 输出为 $1.2/240 * (1 \sim 240)$ 或者 $VCCA/240 * (1 \sim 240)$
- 短路保护功能
 - 内置 2 路阈值短路保护 (可选 VCCA-VTH 或者 PAD-VTH), 其中 VTH 档位调节输出电压可选为 80mv/200mv/320mv/480mv。
- 恒流源功能
 - 1 路 6bit 校准精度为 2.5% 的 20mA 恒流源输出, TYP 输出范围: $-40.5 \sim +42\%$
- 迟滞功能
 - 2 路比较器支持单边 (正/负) 和双边 (正负) 模拟迟滞控制, 迟滞电压可选为 10mv/20mv/60mv。
 - 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 $1.2/240 * (1 \sim 240)$ 或者 $VCCA/240 * (1 \sim 240)$ 内选择;
- 数字滤波功能
 - 支持数字滤波, 滤波时间共 256 档位选择, 步长为 1 微秒;
- 中断与唤醒功能
 - 支持输出改变产生中断;

- 支持比较器唤醒睡眠状态
- 比较器与系统其他模块联动功能
 - 比较器使能可由内部PWM控制
 - 比较器输出可作为增强型PWM模块的刹车触发信号；
 - 比较器的数字输出可以映射到任意IO引脚

20.2. 模块框图

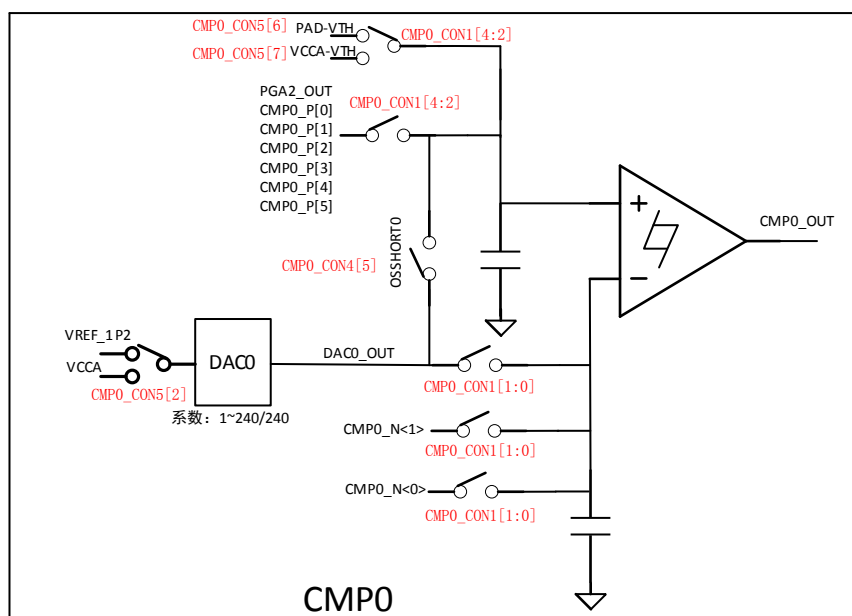


图 20-1 比较器 0 模块内部结构图

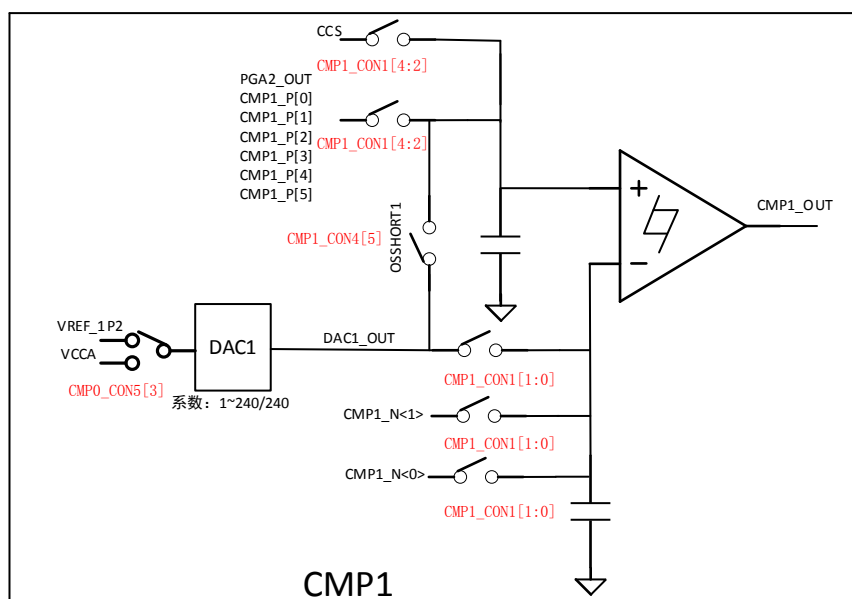


图 20-2 比较器 1 模块内部结构图



20.3. 引脚复用映射表

表 20-1 CMP 引脚复用映射表

引脚复用名字		负端通道选择 CMPx_CON1[1:0]	正端通道选择 CMPx_CON1[4:2]	关闭数字 IO 功能	使能比较器模 拟复用功能	GPIO
CMP0	COP0	/	0x01	P06MD=3	P06AIOEN=1	P06
	COP1	/	0x02	P07MD=3	P07AIOEN=1	P07
	COP2	/	0x03	P10MD=3	P10AIOEN=1	P10
	COP3	/	0x04	P03MD=3	P03AIOEN=1	P03
	COP4	/	0x05	P23MD=3	P23AIOEN=1	P23
	COP5	/	0x06	P26MD=3	P26AIOEN=1	P26
	CON0	0x01	/	P05MD=3	P05AIOEN=1	P05
	CON1	0x02	/	P04MD=3	P04AIOEN=1	P04
CMP1	C1P0	/	0x01	P13MD=3	P13AIOEN=1	P13
	C1P1	/	0x02	P12MD=3	P12AIOEN=1	P12
	C1P2	/	0x03	P11MD=3	P11AIOEN=1	P11
	C1P3	/	0x04	P02MD=3	P02AIOEN=1	P02
	C1P4	/	0x05	P23MD=3	P23AIOEN=1	P23
	C1P5	/	0x06	P26MD=3	P26AIOEN=1	P26
	C1N0	0x01	/	P14MD=3	P14AIOEN=1	P14
	C1N1	0x02	/	P01MD=3	P01AIOEN=1	P01
阈值比较	DACMP	/	/	P15MD=3	/	P15
恒流源	CCS	/	/	P16MD=3	/	P16

NOTE: 选择引脚作为通道必须把 IO 的模拟使能打开, 即配置相应的 PTx_AIOEN, 具体请

参照第 9 章节关于 I/O 复用部分的描述!

20.4. 功能配置流程图

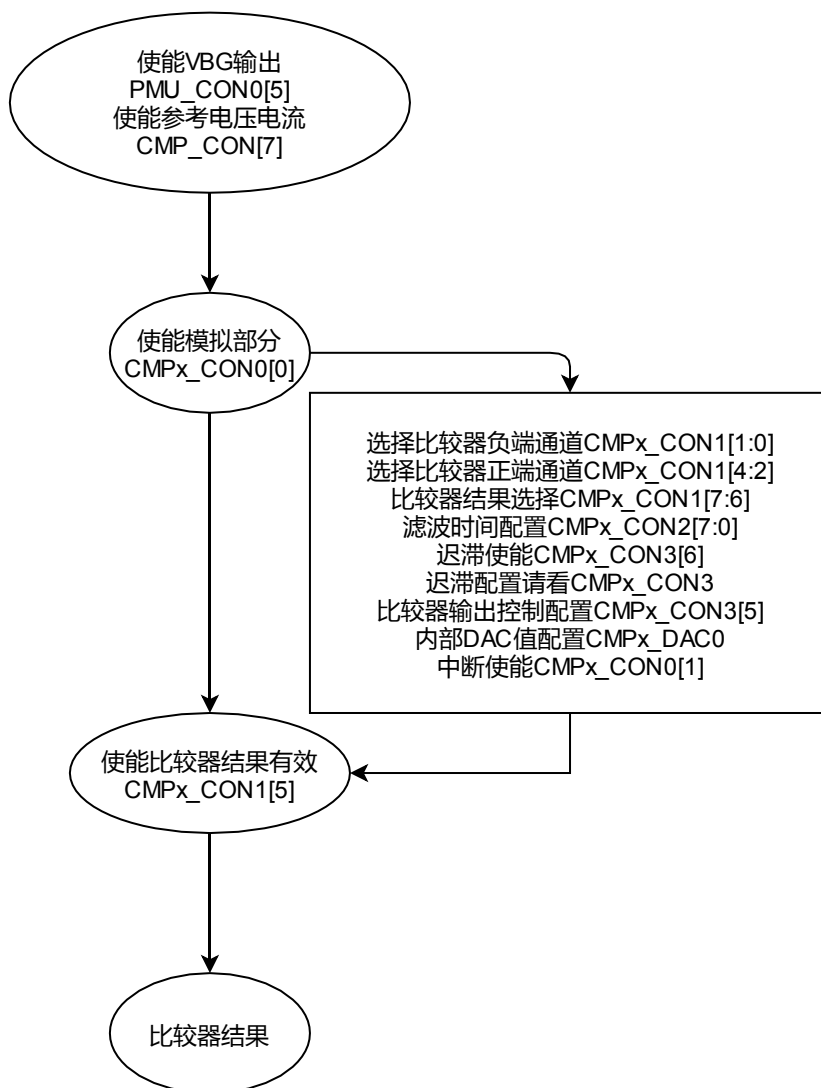


图 20-3 比较器使用配置流程说明图

Note: 图中 CMPx_CON1 是指 CMP0_CON1 和 CMP1_CON1！

20.5. 基本功能使用说明

20.5.1. 比较器工作模式使用说明

比较器工作模式分为两种比较情况：

第一种比较器工作情况是比较器外部正端输入和外部负端输入的信号之间的比较；

第二种比较器工作情况是比较器外部正端输入和内部负端选择的信号之间的比较；

**如果是第一种外部正端和外部负端之间的比较，则配置流程如下：**

- (1) 选择比较器正端和负端的功能复用引脚: 参照本章内容中关于引脚复用映射表的内容配置比较器 0 和比较器 1 的正端和负端的功能复用引脚, 每个比较器的正端最多可以选择 5 个输入引脚通路, 负端最多可以选择 2 个输入引脚通路;

以比较器 0 的配置举例, 具体代码配置如下:

//关闭比较器 0 复用功能引脚的数字 I/O 功能

P0_MD1 |= (0x3<<4) | (0x3<<2); //关闭 C0N0(P05), C0P0(P06) 的数字 I/O 功能

P0_AIOEN = 0x60; //使能 P05, P06 模拟比较器复用功能

//比较器 0 正端输入通道选择 P06, 负端输入通道选择 P05

CMP0_CON1= (CMP0_CON1 & ~(0x3<<0)) | (0x1<<0); //负端选择 P05

CMP0_CON1= (CMP0_CON1 & ~(0x7<<2)) | (0x1<<2); //正端选择 P06

- (2) 使能比较器的参考电压参考电流, 比较器模拟部分, 建立稳定状态;

以比较器 0 的配置举例, 具体代码配置如下:

PMU_CON0 |= 0x1<<5; //使能 VBG06_REF 输出

CMP_CON |= 0x1<<7; //使能比较器参考电压参考电流

CMP0_CON0 |= 0x1<<0; //使能比较器模拟部分, 建立稳定状态

- (3) 根据用户需要配置使能比较器的 P 管和 N 管, 如果想简单使用, 就把 P 管和 N 管都使能;

以比较器 0 的配置举例, 具体代码配置如下:

CMP0_CON4 &= ~(0x1<<4); //打开比较器的 N 管

CMP0_CON4 &= ~(0x1<<6); //打开比较器的 P 管

重要说明: 以 VCC/2 为基准, 共模电压比较低的时候选择使能 P 管, 共模电压比较高的时候选择使能 N 管;

- (4) 根据用户实际应用需要, 配置比较器的模拟迟滞, 数字迟滞, 数字滤波等功能;
- (5) 用户读取比较器的比较结果, 也可以使能比较器的中断, 产生中断; 也可以将比较器的结果通过选择任意 I/O 中的一个引脚输出;

如果是第二种外部正端输入信号和内部负端输入信号之间的比较，则配置流程如下：

- (1) 选择比较器正端功能复用引脚: 参照本章内容中关于引脚复用映射表的内容配置比较器 0 和比较器 1 的正端功能复用引脚, 每个比较器的正端最多可以选择 5 个输入引脚通路;

以比较器 0 的配置举例, 具体代码配置如下:

//关闭比较器 0 复用功能引脚的数字 I/O 功能

P0_MD1 |= (0x3<<4); //关闭 C0P0(P06) 的数字 I/O 功能

P0_AIOEN = 0x40; //使能 P06 模拟比较器复用功能

//比较器 0 正端输入通道选择 P06



```
CMP0_CON1= (CMP0_CON1 & ~(0x7<<2)) | (0x1<<2); //正端选择 P06
```

//比较器 0 负端输入通道选择内部 DAC0_OUT 即通过内部 DAC0 配置的电压比较值

```
CMP0_CON1= (CMP0_CON1 & ~(0x3<<0)) | (0x3<<0); //负端选择 DAC0_OUT
```

- (2) 使能比较器的参考电压参考电流，比较器模拟部分，建立稳定状态；

以比较器 0 的配置举例，具体代码配置如下：

```
PMU_CON0 |= 0x1<<5; //使能 VBG06_REF 输出
```

```
CMP_CON |= 0x1<<7; //使能比较器参考电压参考电流
```

```
CMP0_CON0 |= 0x1<<0; //使能比较器模拟部分，建立稳定状态
```

- (3) 根据用户需要配置使能比较器的 P 管和 N 管，如果想简单使用，就把 P 管和 N 管都使能；

以比较器 0 的配置举例，具体代码配置如下：

```
CMP0_CON4 &= ~(0x1<<4); //打开比较器的 N 管
```

```
CMP0_CON4 &= ~(0x1<<6); //打开比较器的 P 管
```

重要说明：以 $VCC/2$ 为基准，共模电压比较低的时候选择使能 P 管，共模电压比较高的时候选择使能 N 管；

- (4) 选择 DAC0 的参考源，配置比较器 DAC0 的电压值，DAC0 的电压值等于参考源 $/0xF0 * CMP0\_DHYH$, DAC 默认参考电压是 VCCA；

以比较器 0 的配置举例，具体代码配置如下：

//选择内部 VCCA 作为参考源

```
CMP0_CON5= (CMP0_CON5 & ~(0x1<<3)) | (0x1<<3);
```

//配置比较器 0 DAC0 的电压值，则负端输入比较信号的电压值为： $(0x78 * VCCA) / 0xF0$

```
CMP0_DHYH = 0x78;
```

重要说明：当比较器正端引脚和负端的 DAC 设置的电压值进行比较功能时，以及需要比较器使用数字迟滞功能（通过设置比较器 CMP0_DHYH 和 CMP0_DHYL 设置迟滞窗口）这两种情况下需要配置比较器 DAC 的电压值这一步骤；当比较器的正端引脚和负端引脚两种进行比较时，不需要配置 DAC 电压值这一步骤；

- (5) 根据用户实际应用需要，配置比较器的模拟迟滞，数字迟滞，数字滤波等功能；

- (6) 用户读取比较器的比较结果，也可以使能比较器的中断，产生中断；也可以将比较器的结果通过选择任意 IO 中的一个引脚输出；

20.5.2. 短路保护功能使用说明

短路保护功能主要是用来监测用户应用方案中如 MOS 管是否短路烧掉这种需要检测短路和保护的应用场景。系统检测到 MOS 管短路就产生刹车型号刹车 PWM，避免烧坏 MOS 管。

- (1) 短路保护功能是集成在模拟比较器 0 里面的功能，此时比较器 0 的正端输入选择短路检



测功能通路即 CMP0_CON1[4:2]=0x7;

- (2) 短路检测功能可以检测内部 VCC 和外部 P15 两种，可以通过配置 CMP0_CON5[7]=1，使能检测 VCC 通路或者配置 CMP0_CON5[6]=1，使能检测 P15 外部通路；VCC 或者 P15 连接到被检测对象如 MOS 管近电源的端点；
- (3) 根据应用的检测需要，可以通过被检测对象如 MOS 管的特性来配置 CMP0_CON5[5:4]来选择阈值电压 VTH 档位调节输出电压可选为 80mv/200mv/320mv/480mv 这 4 个档位，此时比较器 0 的正端的检测信号的电压为：VCC-VTH 或者 P15-VTH；
- (4) 比较器 0 的负端输入选择外部某个引脚，该引脚连接到被检测对象如 MOS 管的近地的端点；
- (5) 使能比较器的参考电压参考电流，比较器模拟部分，建立稳定状态；
- (6) 具体代码配置如下：


```
PMU_CON0 |= 0x1<<5;    //使能 VBG06_REF 输出
CMP_CON    |= 0x1<<7;    //使能比较器参考电压参考电流
CMP0_CON0  |= 0x1<<0;    //使能比较器模拟部分，建立稳定状态
```
- (7) 根据用户实际应用需要，配置比较器的迟滞，滤波等功能；
- (8) 用户读取比较器的比较结果来判断被检测对象如 MOS 管是否短路，如果短路可以直接输出到 PWM 进行刹车操作，以达到保护 MOS 管的目的；

20.5.3.恒流源功能使用说明

注意：比较器恒流源功能在芯片出厂前会经过校准，校准值会存放到 NVR 存储器的某个固定地址，SDK 中的系统初始化部分会读取出来校准值，并配置到 CMP_CON[5:0]；

恒流源功能具体流程如下：

- (1) 关闭 CCS 恒流源复用引脚 P16 的数字 I/O 功能：

```
P1_MD1 |= 0x3<<4;
```

- (2) 打开恒流源功能使能：

```
CMP_CON |= 0x3<<6;
```

重要说明：恒流源功能是集成在模拟比较器 1 里面的功能，在恒流源功能开启的时候，也可以结合比较器的功能一起使用，此时比较器 1 的正端输入选择 CCS，比较器 1 的负端输入选择设置的内部设置的电压值，用于检测 CCS 驱动的外部通路的短路或者断路检测，从而在方案上实现短路和断路保护功能；



20.6. 寄存器列表

表 20-1 CMP0/1 register list

address	Register Name	Description
0xC8 (SFR)	CMP0_CON0	CMP0 control register 0
0x143 (XSFR)	CMP0_CON1	CMP0 control register 1
0x144 (XSFR)	CMP0_CON2	CMP0 control register 2
0x145 (XSFR)	CMP0_CON3	CMP0 control register 3
0x146 (XSFR)	CMP0_CON4	CMP0 control register 4
0x149 (XSFR)	CMP0_CON5	CMP0 control register 5
0x147 (XSFR)	CMP0_DHYH	CMP0 Digital Hysteresis High Threshold Set Value
0x148 (XSFR)	CMP0_DHYL	CMP0 Digital Hysteresis Low Threshold Set Value
0xF8 (SFR)	CMP1_CON0	CMP1 control register 0
0x14C (XSFR)	CMP1_CON1	CMP1 control register 1
0x14D (XSFR)	CMP1_CON2	CMP1 control register 2
0x14E (XSFR)	CMP1_CON3	CMP1 control register 3
0x14F (XSFR)	CMP1_CON4	CMP1 control register 4
0x150 (XSFR)	CMP1_DHYH	CMP1 Digital Hysteresis High Threshold Set Value
0x151 (XSFR)	CMP1_DHYL	CMP1 Digital Hysteresis Low Threshold Set Value
0x140 (XSFR)	CMP_CON	Control the CMP0 and CMP1
0x141 (XSFR)	CMP_STA	CMP0 and CMP1 state register
0x14A (XSFR)	CMP_AHYCON	CMP0 and CMP1 Analog Hysteresis Control Register

20.7. 寄存器详细说明

20.7.1. CMP_CON

Addr = 0x140 (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: 不使能	RW	0x0



		0x1: 使能		
6	CCSEN	恒流源的使能信号. 0x0: 不使能 0x1: 使能	RW	0x0
5: 0	TRIMIB	恒流源电流调节信号(step=2.5%) 0x00: 11.8mA 0x0F: 20mA 0x3F: 28.4mA	RW	0x0

20.7.2. CMP_AHYCON

Addr = 0x14A (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	CMP0_HYSPNS_VDD	比较器 0 的输入正负迟滞选择信号 0x0: disable 迟滞输出 0x1: 正迟滞输出 0x2: 负迟滞输出 0x3: 正负迟滞输出	R/W	0x0
5: 4	CMP0_HYSVTH_VDD	比较器 0 的迟滞电压选择 0x0: 0mv 0x1: 10mv 0x2: 20mv 0x3: 60mv Note: CMP1_HYSPNS_VDD=0x0 的时候, CMP0_HYSVTH_VDD 必须选择等于 0x0;	R/W	0x0
3: 2	CMP1_HYSPNS_VDD	比较器 1 的输入正负迟滞选择信号 0x0: disable 迟滞输出 0x1: 正迟滞输出 0x2: 负迟滞输出 0x3: 正负迟滞输出	R/W	0x0
1: 0	CMP1_HYSVTH_VDD	比较器 1 的迟滞电压选择 0x0: 0mv	R/W	0x0



		0x1: 10mv		
		0x2: 20mv		
		0x3: 60mv		

Note: 模拟迟滞控制!

20.7.3. CMP_STA

Addr = 0x141 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	CMP1WKUPEN	比较器1 数字信号 wakeup 使能, 数字唤醒指当比较器中断标志位为 1 时唤醒 0x0: 不使能 0x1: 使能	RW	0x0
6	CMPOWKUPEN	比较器0 数字信号 wakeup 使能, 数字唤醒指当比较器中断标志位为 1 时唤醒 0x0: 不使能 0x1: 使能	RW	0x0
5	CMP1ANAWKP	比较器1 模拟信号 WAKEUP 使能, 模拟唤醒指使用比较器的模拟输出进行唤醒, 配置成模拟唤醒后, 可关闭比较器模块的时钟 0x0: 不使能 0x1: 使能	RW	0x0
4	CMPOANAWKP	比较器0 模拟信号 WAKEUP 使能, 模拟唤醒指使用比较器的模拟输出进行唤醒, 配置成模拟唤醒后, 可关闭比较器模块的时钟 0x0: 不使能 0x1: 使能	RW	0x0
3	CMP1ANAINVEN	比较器1 模拟 WAKEUP 取反功能使能 0x0: 不使能 0x1: 使能	RW	0x0
2	CMPOANAINVEN	比较器0 模拟 WAKEUP 取反功能使能 0x0: 不使能 0x1: 使能	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, 写 1 清零	RW	0x0

20.7.4. CMP0_CON0

Addr = 0xC8 (SFR)

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: 不使能 0x1: 使能	RW	0x0
2	CMPOUT	比较器的比较结果, 只读	RO	0x0
1	INTENA	中断使能信号 0x0: 不使能 0x1: 使能	RW	0x0
0	ENA	比较器使能信号 使能之后还需要使能数字部分, 比较器才正常工作 0x0: 不使能 0x1: 使能	RW	0x0

20.7.5. CMP0_CON1

Addr = 0x143 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	CMP_F_SEL	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	RW	0x0



		0x1: 比较器输出同步后的结果 0x2: 比较器输出迟滞采样结果		
5	OUTPUT_EN	控制比较器结果是否输出控制位 0x0: 不输出, 结果保持关闭输出前的值 0x1: 输出	RW	0x0
4: 2	CHPSEL	比较器 0 正端输入通道选择. 0x0: CMP_PGA (PGA2 的输出) 0x1: CMP0_P<0> (P06) 0x2: CMP0_P<1> (P07) 0x3: CMP0_P<2> (P10) 0x4: CMP0_P<3> (P03) 0x5: CMP0_P<4> (P23) 0x6: CMP0_P<5> (P26) 0x7: VTH_OUT_VCC (VTH_PADEN=0) / VTH_OUT_PAD (VTH_PADEN=1) (P15)	RW	0x0
1: 0	CHNSEL	比较器 0 负端输入通道选择 0x0: 不使能 0x1: CMP0_N<0> (P05) 0x2: CMP0_N<1> (P04) 0x3: DAC0_OUT	RW	0x0

20.7.6. CMP0_CON2

Addr = 0x144 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	FILT_NUM	滤波时钟个数设置, 最大值 256, 步长为 1 微秒	RW	0x0

20.7.7. CMP0_CON3

Addr = 0x145 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	HYSEL	选择迟滞信号是否经过滤波器	RW	0x0



		0x0: 经过滤波后的信号 0x1: 滤波前的同步信号		
6	HYSEN	比较器 0 迟滞功能使能位, 通过 CMP0_DHYH 和 CMP0_DHYL 的两个寄存器控制阈值	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	CMP0HYCNT	迟滞计数器, 用来设置迟滞采样间隔, 步长为 16 微秒和 1 微秒可选 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

20.7.8. CMP0_CON4

Addr = 0x146 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	DACTSEN	内部 DAC 的输出测试使能信号. 0x0: 关闭 0x1: 打开	RW	0x0
6	P_DISABLE	关闭 P 管 0x0: 不关闭 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	N_DISABLE	关闭 N 管 0x0: 不关闭 N 管	RW	0x0



		0x1: 关闭 N 管		
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	0x0

注意: 共模电压 (比较器两端的电压) 比较低 (低于 $1/2V_{CC}$) 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 $0 \sim V_{CC}$ 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。

20.7.9. CMP0_CON5

Addr = 0x149 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	VTH_VCCEN	VCC 短路保护阈值 ($V_{CC}-0.08/0.2/0.32/0.48$) 输入使能信号. 0x0: 不使能 0x1: 使能	RW	0x0
6	VTH_PADEN	P15 短路保护阈值 ($P15-0.08/0.2/0.32/0.48$) 输入使能信号. 0x0: 不使能 0x1: 使能	RW	0x0
5: 4	CMPVTHS	比较器内置阈值选择. 0x0: $V_{CC}/P15 -80\text{mv}$ 0x1: $V_{CC}/P15 -200\text{mv}$ 0x2: $V_{CC}/P15 -320\text{mv}$ 0x3: $V_{CC}/P15 -480\text{mv}$	RW	0x0
3	DAC1VREFSEL	比较器 1 的 DAC 的输入参考电压选择信号. 0x0: 选择内部 1.2V 作为参考电压 0x1: 选择内部 V_{CCA} 作为参考电压	RW	0x0
2	DAC0VREFSEL	比较器 0 的 DAC 的输入参考电压选择信号. 0x0: 选择内部 1.2V 作为参考电压 0x1: 选择内部 V_{CCA} 作为参考电压	RW	0x0
1: 0	—	—	—	—



20.7.10. CMP0_DHYH

Addr = 0x147 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CMPODHYH	比较器 0 的数字迟滞窗口高阈值设置寄存器 选择 1.2V 参考时 Step=5mV, 0x00~0xF0 对应转换输出为 0~1.2V, 0xF0~0xFF 转换输出为 1.2V, 当选择 VCCA 参考时, 同 1.2V 参考等比例换算即可。 CMP0_CON5[2]选择 DAC 的参考源! Note: 如果不开启数字迟滞, 本寄存器即为比较器内置 DAC 的输入; (比较器内置 DAC 设置值复用该寄存器)	RW	0x0

20.7.11. CMP0_DHYL

Addr = 0x148 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CMPODHYL	比较器 0 的数字迟滞窗口低阈值设置寄存器 选择 1.2V 参考时 Step=5mV, 0x00~0xF0 对应转换输出为 0~1.2V, 0xF0~0xFF 转换输出为 1.2V, 当选择 VCCA 参考时, 同 1.2V 参考等比例换算即可	RW	0x0

20.7.12. CMP1_CON0

Addr = 0xF8 (SFR)

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: 不使能 0x1: 使能	RW	0x0
2	CMPOUT	比较器的比较结果, 只读	RO	0x0
1	INTENA	中断使能信号 0x0: 不使能 0x1: 使能	RW	0x0
0	ENA	比较器使能信号 使能之后还需要使能数字部分, 比较器才正常工作 0x0: 不使能 0x1: 使能	RW	0x0

20.7.13. CMP1_CON1

Addr = 0x14C (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	CMP_F_SEL	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 0x0: 不输出, 结果保持关闭输出前的值 0x1: 输出	RW	0x0
4: 2	CHPSEL	比较器 1 正端输入通道选择 0x0: CMP_PGA (PGA2 的输出) 0x1: CMP1_P<0> (P13) 0x2: CMP1_P<1> (P12) 0x3: CMP1_P<2> (P11) 0x4: CMP1_P<3> (P02) 0x5: CMP1_P<4> (P23) 0x6: CMP1_P<5> (P26) 0x7: CCS (P16)	RW	0x0



1: 0	CHNSEL	比较器 1 负端输入通道选择 0x0: 不使能 0x1: CMP1_N<0> (P14) 0x2: CMP1_N<1> (P01) 0x3: DAC1_OUT	RW	0x0

20.7.14. CMP1_CON2

Addr = 0x14D (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	FILT_NUM	滤波时钟个数设置，最大值 256，步长为 1 微秒	RW	0x0

20.7.15. CMP1_CON3

Addr = 0x14E (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 1 迟滞功能使能位，通过 CMP1_DHYH 和 CMP1_DHYL 的两个寄存器控制阈值	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	CMP1HYCNT	迟滞计数器，用来设置迟滞采样间隔，步长为 16 微秒和 1 微秒可选 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

20.7.16. CMP1_CON4

Addr = 0x14F (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	DACTSEN	内部 DAC 的输出测试使能信号, 0x0: 关闭 0x1: 打开	RW	0x0
6	P_DISABLE	关闭 P 管 0x0: 不关闭 P 管 0x1: 关闭 P 管	RW	0x0
5	OSSHORT	短路使能 0x0: 不使能 0x1: 使能	RW	0x0
4	N_DISABLE	关闭 N 管 0x0: 不关闭 N 管 0x1: 关闭 N 管	RW	0x1
3: 0	TRIM	比较器 1 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	0x0

注意: 共模电压 (比较器两端的电压) 比较低 (低于 1/2VCC) 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。

20.7.17. CMP1_DHYH

Addr = 0x150 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CMP1DHYH	比较器 1 的数字迟滞窗口高阈值设置寄存器 选择 1.2V 参考时 Step=5mV, 0x00~0xF0 对应转换输出为 0~1.2V, 0xF0~0xFF 转换输出为 1.2V, 当选择 VCCA 参考时, 同 1.2V 参考等比例换算即可。	RW	0x0



		CMP0_CON5[2]选择 DAC 的参考源 Note: 如果不开启数字迟滞，本寄存器即为比较器内置 DAC 的输入；（比较器内置 DAC 设置值复用该寄存器）		

20.7.18. CMP1_DHYL

Addr = 0x151 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	CMP1DHYL	比较器 1 的数字迟滞窗口低阈值设置寄存器 选择 1.2V 参考时 Step=5mV, 0x00~0xF0 对应转换输出为 0~1.2V, 0xF0~0xFF 转换输出为 1.2V, 当选择 VCCA 参考时, 同 1.2V 参考等比例换算即可	RW	0x0

21. 运放模块

21.1. 功能概述

- 内部集成 3 个运算放大器，支持 OP，PGA 和内部比较器工作模式
- PGA 模式下多级可配置增益（设计理论值：1/2/4/8/16/32/64/128/256/512），实际芯片由于受到封装绑线等因素会引入额外的等效电阻，导致输入电阻会偏大，所以计算增益的时候输入电阻 R_{in} ，需要在寄存器配置的输入电阻值加上一个等效电阻 R_x （对于 PGA0, PGA1, R_x 约等于 150 欧姆；对于 PGA2, R_x 约等于 250 欧姆）；
- 3 个 PGA 都支持 OP 模式，外接电阻调节运放的放大增益
- 支持正相放大和负相放大
- 支持内部比较器模式
- 支持多个 PGA 级联，获得更高的增益，用于小信号放大应用
- 3 个运放的输出都可以输入到 ADC 的内部输入通道
- 运放 PGA2 输出可以输入到比较器 CMP1 的内部输入通道

21.2. 引脚复用表

表 21-1 运放引脚复用表



引脚复用运放名字		关闭数字 GPIO 功能	引脚
PGA0	OP0_P	P21MD=0x3	P21
	OP0_N	P22MD=0x3	P22
	OP0_0	P23MD=0x3	P23
PGA1	OP1_P	P30MD=0x3	P30
	OP1_N	P27MD=0x3	P27
	OP1_0	P26MD=0x3	P26
PGA2	OP2_P	P25MD=0x3/P22MD=0x3	P25/P22
	OP2_N	P16MD=0x3/P21MD=0x3	P16/P21
	OP2_0	P24MD=0x3	P24
	OPOUTF	P20MD=0x3	P20

21.3. 基本运放功能

建议：对于简单应用运放的用户请直接阅读本章节就可以用起来，可以忽略后续章节的内容！

运放模块基本功能是基于默认的 OP 工作模式，需要用户根据应用方案的需求的放大增益来外接电阻实现，具体配置流程和代码如下：

- (1) 关闭 OP 复用的对应的数字 GPIO 功能，关闭对应复用引脚上其他的模拟功能复用通路，具体复用关系详见本章节《21.2 引脚复用表》；

举例说明：

如果需要用 OP0，则需要关闭 OP0_P[P21], OP0_N[P22], OP0_0[P23]对应的 P21, P22, P23 的数字 GPIO 功能关闭，即设置 P21MD=0x3, P22MD=0x3, P23MD=0x3；

如果需要使用 OP0，则对应代码如下：

```
P2_MD0 |= (0x3<<6) | (0x3<<4) | (0x3<<2); //关闭 P21, P22, P23 数字 IO 功能
```

同理，如果需要使用 OP1，则对应代码如下：

```
P3_MD0 |= (0x3<<0); //关闭 P30 数字 IO 功能
```

```
P2_MD1 |= (0x3<<6) | (0x3<<4); //关闭 P26, P27 数字 IO 功能
```

同理，如果需要使用 OP2，则对应代码如下：

```
P1_MD1 |= (0x3<<4); //关闭 P16 数字 IO 功能
```




P2_MD1 |= (0x3<<2) | (0x3<<0); //关闭 P24, P25 数字 IO 功能

- (2) 设置 OP 负相输入，正相输入，和输出端到 IO 的通路使能；

如果需要用 OP0，则对应代码如下：

AMP_CON8 |= 0x1<<5; //PGA0 的负相输入到 IO 的通路使能

AMP_CON9 |= 0x1<<5; //PGA0 的正相输入到 IO 的通路使能

AMP_CON10 |= 0x1<<3; //PGA0 的输出端口到 IO 的通路使能

同理，如果需要用 OP1，则对应代码如下：

AMP_CON8 |= 0x1<<6; //PGA1 的负相输入到 IO 的通路使能

AMP_CON9 |= 0x1<<6; //PGA1 的正相输入到 IO 的通路使能

AMP_CON10 |= 0x1<<4; //PGA1 的输出端口到 IO 的通路使能

同理，如果需要用 OP2，则对应代码如下：

AMP_CON8 &= ~(0x3<<0); //关闭 PGA2 负相和正相输入默认连接 PGA0 的通路

AMP_CON8 |= 0x1<<7; //PGA2 的负相输入到 IO 的通路使能

AMP_CON9 |= 0x1<<7; //PGA2 的正相输入到 IO 的通路使能

AMP_CON10 |= 0x1<<5; //PGA2 的输出端口到 IO 的通路使能

- (3) 最后，使能对应使用的运放模块；

如果需要使用 OP0，则对应代码如下：

AMP_CON11 = (AMP_CON11 & ~(0x1<<0)) | (0x1<<0); // OP0 使能

同理，如果需要使用 OP1，则对应代码如下：

AMP_CON11 = (AMP_CON11 & ~(0x1<<1)) | (0x1<<1); // OP1 使能

同理，如果需要使用 OP2，则对应代码如下：

AMP_CON11 = (AMP_CON11 & ~(0x1<<2)) | (0x1<<2); //OP2 使能

21.4. 增强功能

建议：对于想深入使用运放各种功能的有一定能力的用户可以深入阅读本章节的内容，想简单应用运放的用户忽略本章节内容！

21.4.1.OP 工作模式使用说明

- (1) 关闭 OP 复用的对应的数字 GPIO 功能，具体复用关系详见本章节《21.2 引脚复用表》；

举例说明：如果需要用 OP0，则需要关闭 OP0_P[P21], OP0_N[P22], OP0_O[P23]对应的 P21, P22, P23 的数字 GPIO 功能关闭，即设置 P21MD=0x3, P22MD=0x3, P23MD=0x3；关闭对应复用引脚上其他的模拟功能复用通路，即设置 P21AIOEN=0x0, P22AIOEN=0x0，



P23AIOEN=0x0;

如果需要使用 OP0, 则对应代码如下:

P2_MD0 |= (0x3<<6) | (0x3<<4) | (0x3<<2); //关闭 P21, P22, P23 数字 IO 功能

P2_AIOEN = 0x0; //关闭 P21, P22, P23 其他模拟复用

同理, 如果需要使用 OP1, 则对应代码如下:

P3_MD0 |= (0x3<<0); //关闭 P30 数字 IO 功能

P2_MD1 |= (0x3<<6) | (0x3<<4); //关闭 P26, P27 数字 IO 功能

P3_AIOEN = 0x0; //关闭 P30 其他模拟复用

P2_AIOEN = 0x0; //关闭 P21, P22, P23 其他模拟复用

同理, 如果需要使用 OP2, 则对应代码如下:

P1_MD1 |= (0x3<<4); //关闭 P16 数字 IO 功能

P2_MD1 |= (0x3<<2) | (0x3<<0); //关闭 P24, P25 数字 IO 功能

P1_AIOEN = 0x0; //关闭 P16 其他模拟复用

P2_AIOEN = 0x0; //关闭 P24, P25 其他模拟复用

(2) 关闭内部反馈通路, 即使能 OP 工作模式, 默认是 OP 运放工作模式;

如果需要用 OP0, 则对应代码如下:

AMP_CON7 &= ~(0x1<<5); //OP0 的 OP 工作模式使能

同理, 如果需要使用 OP1, 则对应代码如下:

AMP_CON7 &= ~(0x1<<6); //OP1 的 OP 工作模式使能

同理, 如果需要使用 OP2, 则对应代码如下:

AMP_CON7 &= ~(0x1<<7); //OP2 的 OP 工作模式使能

(3) 设置 OP 负相输入, 正相输入, 输出端到 IO 的通路使能;

如果需要用 OP0, 则对应代码如下:

AMP_CON8 |= 0x1<<5; //PGA0 的负相输入到 IO 的通路使能

AMP_CON9 |= 0x1<<5; //PGA0 的正相输入到 IO 的通路使能

AMP_CON10 |= 0x1<<3; //PGA0 的输出端口到 IO 的通路使能

同理, 如果需要用 OP1, 则对应代码如下:

AMP_CON8 |= 0x1<<6; //PGA1 的负相输入到 IO 的通路使能

AMP_CON9 |= 0x1<<6; //PGA1 的正相输入到 IO 的通路使能

AMP_CON10 |= 0x1<<4; //PGA1 的输出端口到 IO 的通路使能

同理, 如果需要用 OP2, 则对应代码如下:

AMP_CON8 &= ~(0x3<<0); //关闭 PGA2 负相和正相输入默认连接 PGA0 的通路

AMP_CON8 |= 0x1<<7; //PGA2 的负相输入到 IO 的通路使能

AMP_CON9 |= 0x1<<7; //PGA2 的正相输入到 IO 的通路使能



AMP_CON10 |= 0x1<<5; //PGA2 的输出端口到 IO 的通路使能

- (4) OP 的增益，是通过芯片外部的电阻（负相输入电阻和反馈电阻）来实现用户需要的放大增益倍数；

OP 正相放大增益倍数计算公式为：Gain=(Rfb+Rin)/Rin;

OP 负相放大增益倍数计算公式为：Gain=-Rfb/Rin;

同理，如果需要用 OP1, OP2，则参照 OP0 的使用说明即可。

- (5) 配置 OP 的偏置电流选择，即配置对应的运放总电流，输出级电流，及 OFFSET 电流选择大电流或者小电流；

如果需要使用 OP0，则对应代码如下：

//OP0 运放总电流选择大电流

AMP_CON1 = (AMP_CON1 & ~(0x1<<0)) | (0x1<<0);

//OP0 运放输出级电流选择大电流

AMP_CON1 = (AMP_CON1 & ~(0x1<<1)) | (0x1<<1);

//OP0 运放 OFFSET 电流选择小电流

AMP_CON1 = (AMP_CON1 & ~(0x1<<2)) | (0x0<<2);

同理，如果需要使用 OP1, 则对应代码如下：

//OP1 运放总电流选择大电流

AMP_CON3 = (AMP_CON3 & ~(0x1<<0)) | (0x1<<0);

//OP1 运放输出级电流选择大电流

AMP_CON3 = (AMP_CON3 & ~(0x1<<1)) | (0x1<<1);

//OP1 运放 OFFSET 电流选择小电流

AMP_CON3 = (AMP_CON3 & ~(0x1<<2)) | (0x0<<2);

同理，如果需要使用 OP2, 则对应代码如下：

//OP2 运放总电流选择大电流

AMP_CON5 = (AMP_CON5 & ~(0x1<<0)) | (0x1<<0);

//OP2 运放输出级电流选择大电流

AMP_CON5 = (AMP_CON5 & ~(0x1<<1)) | (0x1<<1);

//OP2 运放 OFFSET 电流选择小电流

AMP_CON5 = (AMP_CON5 & ~(0x1<<2)) | (0x0<<2);

- (6) 最后，使能对应使用的运放模块；

如果需要使用 OP0，则对应代码如下：

AMP_CON11 = (AMP_CON11 & ~(0x1<<0)) | (0x1<<0); // OP0 使能

同理，如果需要使用 OP1, 则对应代码如下：

AMP_CON11 = (AMP_CON11 & ~(0x1<<1)) | (0x1<<1); // OP1 使能



同理，如果需要使用 OP2, 则对应代码如下：

```
AMP_CON11 = (AMP_CON11 & ~(0x1<<2)) | (0x1<<2); //OP2 使能
```

21.4.2.PGA 工作模式使用说明

- (1) 关闭 PGA 复用的对应的数字 GPIO 功能, 具体复用关系详见本章节《21.2 引脚复用表》;

举例说明：如果需要用 PGA0, 则需要关闭 OP0_P[P21], OP0_N[P22], OP0_O[P23]对应的 P21, P22, P23 的数字 GPIO 功能关闭, 即设置 P21MD=0x3, P22MD=0x3, P23MD=0x3; 关闭对应复用引脚上其他的模拟功能复用通路, 即设置 P21AIOEN=0x0, P22AIOEN=0x0, P23AIOEN=0x0;

对应代码如下：

```
P2_MD0 |= (0x3<<6) | (0x3<<4) | (0x3<<2); //关闭 P21, P22, P23 数字 IO 功能
P2_AIOEN = 0x0; //关闭 P21, P22, P23 其他模拟复用
```

同理，如果需要使用 PGA1, 则对应代码如下：

```
P3_MD0 |= (0x3<<0); //关闭 P30 数字 IO 功能
P2_MD1 |= (0x3<<6) | (0x3<<4); //关闭 P26, P27 数字 IO 功能
P3_AIOEN = 0x0; //关闭 P30 其他模拟复用
P2_AIOEN = 0x0; //关闭 P21, P22, P23 其他模拟复用
```

同理，如果需要使用 PGA2, 则对应代码如下：

```
P1_MD1 |= (0x3<<4); //关闭 P16 数字 IO 功能
P2_MD1 |= (0x3<<2) | (0x3<<0); //关闭 P24, P25 数字 IO 功能
P1_AIOEN = 0x0; //关闭 P16 其他模拟复用
P2_AIOEN = 0x0; //关闭 P24, P25 其他模拟复用
```

- (2) 使能内部反馈通路, 即使能 PGA 工作模式, 默认是 OP 运放工作模式;

如果需要用 PGA0, 则对应代码如下：

```
AMP_CON7 |= 0x1<<5; //PGA0 的 PGA 工作模式使能
```

同理，如果需要使用 PGA1, 则对应代码如下：

```
AMP_CON7 |= 0x1<<6; //PGA1 的 PGA 工作模式使能
```

同理，如果需要使用 PGA2, 则对应代码如下：

```
AMP_CON7 |= 0x1<<7; //PGA2 的 PGA 工作模式使能
```

- (3) 设置 PGA 负相输入, 正相输入, 输出端到 IO 的通路使能;

如果需要用 PGA0, 则对应代码如下：

```
AMP_CON8 |= 0x1<<5; //PGA0 的负相输入到 IO 的通路使能
AMP_CON9 |= 0x1<<5; //PGA0 的正相输入到 IO 的通路使能
```



AMP_CON10 |= 0x1<<3; //PGA0 的输出端口到 IO 的通路使能

同理，如果需要用 PGA1，则对应代码如下：

AMP_CON8 |= 0x1<<6; //PGA1 的负相输入到 IO 的通路使能

AMP_CON9 |= 0x1<<6; //PGA1 的正相输入到 IO 的通路使能

AMP_CON10 |= 0x1<<4; //PGA1 的输出端口到 IO 的通路使能

同理，如果需要用 PGA2，则对应代码如下：

AMP_CON8 &= ~(0x3<<0); //关闭 PGA2 负相和正相输入默认连接 PGA0 的通路

AMP_CON8 |= 0x1<<7; //PGA2 的负相输入到 IO 的通路使能

AMP_CON9 |= 0x1<<7; //PGA2 的正相输入到 IO 的通路使能

AMP_CON10 |= 0x1<<5; //PGA2 的输出端口到 IO 的通路使能

- (4) 配置 PGA 的增益，通过选择内部的可配置电阻（负相输入电阻和反馈电阻）来实现用户需要的放大增益倍数；

如果需要用 PGA0，则对应代码如下：

//PGA0 内置负相输入电阻 Rin 选择 **80K Ω**

AMP_CON1 = (AMP_CON1 & ~(0x7<<5)) | (0x1<<5);

//PGA0 内置反馈电阻 Rfb 选择 **320K Ω**

AMP_CON1 = (AMP_CON1 & ~(0x3<<3)) | (0x2<<3);

重要说明：实际芯片由于受到封装绑线等因素会引入额外的等效电阻，导致输入电阻会偏大，所以计算增益的时候输入电阻Rin，需要在寄存器配置的输入电阻值加上一个等效电阻Rx（对于PGA0,PGA1，Rx约等于 150 欧姆；对于PGA2，Rx约等于 250 欧姆）；

这样配置 PGA0 正相放大增益倍数计算公式为：

$$\text{Gain} = (\text{Rfb} + \text{Rin}) / (\text{Rin} + \text{Rx}) = (320\text{K} + 80\text{K}) / (80\text{K} + 150) = 4.99;$$

这样配置 PGA0 负相放大增益倍数计算公式为：

$$\text{Gain} = -\text{Rfb} / (\text{Rin} + \text{Rx}) = -320\text{K} / (80\text{K} + 150) = -3.99;$$

同理，如果需要用 PGA1, PGA2，则按照 PGA0 的配置对应配置即可，这里略去 PGA1, PGA2 的对应代码。

- (5) 配置 PGA 的偏置电流选择，即配置对应的运放总电流，输出级电流，及 OFFSET 电流选择大电流或者小电流；

如果需要使用 PGA0，则对应代码如下：

//PGA0 运放总电流选择大电流

AMP_CON1 = (AMP_CON1 & ~(0x1<<0)) | (0x1<<0);

//PGA0 运放输出级电流选择大电流

AMP_CON1 = (AMP_CON1 & ~(0x1<<1)) | (0x1<<1);



//PGA0 运放 OFFSET 电流选择小电流

AMP_CON1 = (AMP_CON1 & ~(0x1<<2)) | (0x0<<2);

同理，如果需要使用 PGA1, 则对应代码如下：

//PGA1 运放总电流选择大电流

AMP_CON3 = (AMP_CON3 & ~(0x1<<0)) | (0x1<<0);

//PGA1 运放输出级电流选择大电流

AMP_CON3 = (AMP_CON3 & ~(0x1<<1)) | (0x1<<1);

//PGA1 运放 OFFSET 电流选择小电流

AMP_CON3 = (AMP_CON3 & ~(0x1<<2)) | (0x0<<2);

同理，如果需要使用 PGA2, 则对应代码如下：

//PGA2 运放总电流选择大电流

AMP_CON5 = (AMP_CON5 & ~(0x1<<0)) | (0x1<<0);

//PGA2 运放输出级电流选择大电流

AMP_CON5 = (AMP_CON5 & ~(0x1<<1)) | (0x1<<1);

//PGA2 运放 OFFSET 电流选择小电流

AMP_CON5 = (AMP_CON5 & ~(0x1<<2)) | (0x0<<2);

- (6) 最后，使能对应使用的运放模块；

如果需要使用 PGA0, 则对应代码如下：

AMP_CON11 = (AMP_CON11 & ~(0x1<<0)) | (0x1<<0); // PGA0 使能

同理，如果需要使用 PGA1, 则对应代码如下：

AMP_CON11 = (AMP_CON11 & ~(0x1<<1)) | (0x1<<1); // PGA1 使能

同理，如果需要使用 PGA2, 则对应代码如下：

AMP_CON11 = (AMP_CON11 & ~(0x1<<2)) | (0x1<<2); // PGA2 使能

21.4.3.PGA+PGA 串联工作模式使用说明

该系列芯片支持 PGA0+PGA1, PGA1+PGA2, PGA0+PGA1+PGA2 三种串联工作模式。通过配置寄存器 PGA0T01EN【AMP_CON7[0]】=1, 使能 PGA0 的输出到 PGA1 的输入；通过配置寄存器 PGA1T02EN【AMP_CON7[1]】=1, 使能 PGA1 的输出到 PGA2 的输入；

具体举例 PGA0+PGA1 串联工作模式说明配置流程：

- (1) 参照 21.3.2 的 PGA 工作模式使用说明，独立配置 PGA0 和 PGA1；
- (2) 关闭 PGA0 的输出端口到 IO 的通路使能；

AMP_CON10 &= ~(0x1<<3); //关闭 PGA0 的输出端口到 IO 的通路使能



- (3) 配置寄存器 PGA0T01EN 【AMP_CON7[0]】=1, 使能 PGA0 的输出到 PGA1 的输入;
- (4) 关闭 PGA1 的负相输入到 IO 的通路和正相输入到 IO 的通路;

AMP_CON8 &= ~(0x1<<6); //PGA1 的负相输入到 IO 的通路使能

AMP_CON9 &= ~(0x1<<6); //PGA1 的正相输入到 IO 的通路使能

21.4.4.PGA+ADC 串联工作模式使用说明

该系列芯片支持 PGA0+ADC, PGA1+ADC, PGA2+ADC 串联工作模式, 可以应用在外部被控制信号内部放大后 ADC 采样的工作场景中。

具体举例 PGA0+ADC 串联工作模式说明配置流程:

- (1) 参照 21.3.2 的 PGA 工作模式使用说明, 独立配置 PGA0;
- (2) 关闭 PGA0 的输出端口到 IO 的通路使能;

AMP_CON10 &= ~(0x1<<3); //关闭 PGA0 的输出端口到 IO 的通路使能

- (3) 参照本手册中关于模数转换器的功能配置说明配置 ADC, 并通过配置 CHAN0EXT 【ADC_CHS0[5]】=1, CHANSEL0 【ADC_CHS0[4:0]】=0x4, 选择 PGA0 的输出作为 ADC 的输入采样信号;

21.4.5.PGA2+CMP 串联工作模式使用说明

该系列芯片支持 PGA2+CMP 串联工作模式, 可用于经过 PGA2 内部放大以后的信号输出到芯片内部独立的比较器 CMP 的正端输入通道进行比较的应用场合。

具体配置流程说明:

- (1) 参照 21.3.2 的 PGA 工作模式使用说明, 独立配置 PGA2;
- (2) 关闭 PGA2 的输出端口到 IO 的通路使能;

AMP_CON10 &= ~(0x1<<5); //关闭 PGA2 的输出端口到 IO 的通路使能

- (3) 参照本手册中关于模数比较器的功能配置说明配置 CMPx, 并通过配置 CHPSEL 【CMPx_CON1[4:2]】=0, 选择 PGA2 的输出作为 CMPx 的正端输入通道;

21.4.6.比较器工作模式使用说明

该芯片的运放模块内部集成了比较器功能, PGA0, PGA1, PGA2 都可以独立工作在比较器功能模式。需要说明的是, 第一种比较器模式是支持 PGA 放大以后的信号和 $0.5 \times V_{CCA}$ 做比较, 如果 PGA 放大以后的信号大于 $0.5 \times V_{CCA}$, 则比较器输出为 1, 否则输出为 0; 第二种比



较器模式是 PGA 不打开增益配置通路，即 OP_P 端和 OP_N 端输入信号进行比较，如果 OP_P 端输入信号大于 OP_N 端输入信号，则比较器输出 1，否则输出为 0；

以 PGA0 第一种比较器工作模式为例具体说明：

- (1) 关闭 PGA0 复用的对应的数字 GPIO 功能，具体复用关系详见本章节《21.2 引脚复用表》；
关闭 OP0_P[P21], OP0_N[P22], OP0_0[P23] 对应的 P21, P22, P23 的数字 GPIO 功能关闭，
即设置 P21MD=0x3, P22MD=0x3, P23MD=0x3；关闭对应复用引脚上其他的模拟功能复用
通路，即设置 P21AIOEN=0x0, P22AIOEN=0x0, P23AIOEN=0x0；

对应代码如下：

```
P2_MD0 |= (0x3<<6) | (0x3<<4) | (0x3<<2); //关闭 P21, P22, P23 数字 IO 功能
P2_AIOEN = 0x0; //关闭 P21, P22, P23 其他模拟复用
```

- (2) 使能 PGA0 内部反馈通路，即使能 PGA 工作模式，默认是 OP 运放工作模式；

```
AMP_CON7 |= 0x1<<5; //PGA0 的 PGA 工作模式使能
```

- (3) 设置 PGA0 负相输入，正相输入，输出端到 IO 的通路使能；

```
AMP_CON8 |= 0x1<<5; //PGA0 的负相输入到 IO 的通路使能
AMP_CON9 |= 0x1<<5; //PGA0 的正相输入到 IO 的通路使能
AMP_CON10 |= 0x1<<3; //PGA0 的输出端口到 IO 的通路使能
```

- (4) 配置 PGA0 的增益，通过选择内部的可配置电阻（负相输入电阻和反馈电阻）来实现用户需要的放大增益倍数；

//PGA0 内置负相输入电阻 Rin 选择 80KΩ

```
AMP_CON1 = (AMP_CON1 & ~(0x7<<5)) | (0x1<<5);
```

//PGA0 内置反馈电阻 Rfb 选择 320KΩ

```
AMP_CON1 = (AMP_CON1 & ~(0x3<<3)) | (0x2<<3);
```

这样配置 PGA0 正相放大增益倍数计算公式为： $Gain=(Rfb+Rin)/Rin=(320+80)/80=5$ ；

这样配置 PGA0 负相放大增益倍数计算公式为： $Gain=-Rfb/Rin=-320/80=-4$ ；

- (5) 配置 PGA 的偏置电流选择，即配置对应的运放总电流，输出级电流，及 OFFSET 电流选择大电流或者小电流；

如果需要使用 PGA0，则对应代码如下：

//PGA0 运放总电流选择大电流

```
AMP_CON1 = (AMP_CON1 & ~(0x1<<0)) | (0x1<<0);
```

//PGA0 运放输出级电流选择大电流

```
AMP_CON1 = (AMP_CON1 & ~(0x1<<1)) | (0x1<<1);
```

//PGA0 运放 OFFSET 电流选择小电流

```
AMP_CON1 = (AMP_CON1 & ~(0x1<<2)) | (0x0<<2);
```

- (6) 最后，使能 PGA0 运放模块；



```
AMP_CON11 = (AMP_CON11 & ~(0x1<<0)) | (0x1<<0); // PGA0 使能
```

- (7) PGA0 放大的同时，可以将放大的信号输入跟 $0.5 \times V_{CCA}$ 做内部比较，比较器的输出结果放到寄存器 AMPCMP0OUT【AMP_CON0[0]】，这样 CPU 可以通过读这个寄存器的值，来做过零检测应用。还可以配置 AMPCMP0IE【AMP_CON0[0]】=1，开启运放比较器的中断功能；还可以通过配置 TRIGSEL【AMP_CON0[6]】，来选择是比较器输出结果上升沿或者下降沿变化触发中断的功能；

以 PGA0 第二种比较器工作模式为例具体说明：

- (1) 关闭 PGA0 复用的对应的数字 GPIO 功能，具体复用关系详见本章节《21.2 引脚复用表》；关闭 OP0_P[P21], OP0_N[P22], OP0_O[P23] 对应的 P21, P22, P23 的数字 GPIO 功能关闭，即设置 P21MD=0x3, P22MD=0x3, P23MD=0x3；关闭对应复用引脚上其他的模拟功能复用通路，即设置 P21AIOEN=0x0, P22AIOEN=0x0, P23AIOEN=0x0；

对应代码如下：

```
P2_MD0 |= (0x3<<6) | (0x3<<4) | (0x3<<2); //关闭 P21, P22, P23 数字 IO 功能
P2_AIOEN = 0x0; //关闭 P21, P22, P23 其他模拟复用
```

- (2) 使能 PGA0 内部反馈通路，即使能 PGA 工作模式，默认是 OP 运放工作模式；

```
AMP_CON7 |= 0x1<<5; //PGA0 的 PGA 工作模式使能
```

- (3) 设置 PGA0 负相输入，正相输入到 IO 的通路使能；

```
AMP_CON8 |= 0x1<<5; //PGA0 的负相输入到 IO 的通路使能
AMP_CON9 |= 0x1<<5; //PGA0 的正相输入到 IO 的通路使能
```

- (4) 配置 PGA 的偏置电流选择，即配置对应的运放总电流，输出级电流，及 OFFSET 电流选择大电流或者小电流；

如果需要使用 PGA0，则对应代码如下：

//PGA0 运放总电流选择大电流

```
AMP_CON1 = (AMP_CON1 & ~(0x1<<0)) | (0x1<<0);
```

//PGA0 运放输出级电流选择大电流

```
AMP_CON1 = (AMP_CON1 & ~(0x1<<1)) | (0x1<<1);
```

//PGA0 运放 OFFSET 电流选择小电流

```
AMP_CON1 = (AMP_CON1 & ~(0x1<<2)) | (0x0<<2);
```

- (5) 最后，使能 PGA0 运放模块；

```
AMP_CON11 = (AMP_CON11 & ~(0x1<<0)) | (0x1<<0); // PGA0 使能
```

- (6) PGA0 的 OP0_P 端和 OP0_N 端输入做比较，如果 OP0_P 大于 OP0_N，则比较器输出 1，否则输出 0。比较器的输出结果放到寄存器 AMPCMP0OUT【AMP_CON0[0]】，这样 CPU 可以通过读这个寄存器的值来做其他应用处理。还可以配置 AMPCMP0IE【AMP_CON0[0]】=1，开启运放比较器的中断功能；还可以通过配置 TRIGSEL【AMP_CON0[6]】，来选择是比较



器输出结果上升沿或者下降沿变化触发中断的功能;

21.4.7.关于 PGA 正相放大和负相放大的使用说明

重要说明: 实际芯片由于受到封装绑线等因素会引入额外的等效电阻, 导致输入电阻会偏大, 所以计算增益的时候输入电阻 R_{in} , 需要在寄存器配置的输入电阻值加上一个等效电阻 R_x (对于PGA0, PGA1, R_x 约等于 150 欧姆; 对于PGA2, R_x 约等于 250 欧姆);

- 正相放大时, OPx_N 端需要接地, OPx_P 端输入信号, 放大增益公式为 $Gain = (R_{fb} + R_{in}) / (R_{in} + R_x)$ 。注意此运放为单电源运放, V_{out} 不能输出负电压, OPx_P 端输入 AC 信号时需要带偏置, 且偏置电压也会被放大;
- 负相放大时, OPx_N 端输入信号, OPx_P 端需要接内部偏置, 放大增益公式为 $Gain = -R_{fb} / (R_{in} + R_x)$ 。注意此运放为单电源运放, V_{out} 不能输出负电压, OPx_P 端接内部提供 1.2V 和 1.05V 的偏置电压, 或自行外灌偏置电压, OPx_N 端输入 AC 信号时, 注意添加隔直电容, 输入 DC 信号时, 正常输入即可;

21.4.8.关于运放内部偏置选择配置使用说明

该芯片内部 PGA0, PGA1, PGA2 都支持正相输入端口 ($OP0/1/2\_P$ 端) 选择内部偏置电压, 偏置电压内部可以提供 1.2V 和 1.05V 的选择。当被放大信号为交流小信号, 一定要开启运放内部偏置功能, 否则在运放不能正常工作!

具体选择的代码如下:

- 1.05V 偏置电压连接到 PGA 的正相输入端口:


```
SYS_CON8    |= (1<<1);    // PMU_1.05V to PGA_P
AMP_CON10   |= 0x1<<0;    //PGA0 的正相输入端口选择内部偏置
AMP_CON10   |= 0x1<<1;    //PGA1 的正相输入端口选择内部偏置
AMP_CON10   |= 0x1<<2;    //PGA2 的正相输入端口选择内部偏置
```
- 1.2V 偏置电压连接到 PGA 的正相输入端口:


```
ADKEY_CFG0  |= (1<<6);    // ADC_EN 使能
AIP_CON2    |= (1<<1);    // bias_en 使能
PMU_CON6    |= (1<<4);    // PGA 内部偏置电压输出使能
AMP_CON10   |= 0x1<<0;    //PGA0 的正相输入端口选择内部偏置
```

AMP_CON10 |= 0x1<<1; //PGA1 的正相输入端口选择内部偏置

AMP_CON10 |= 0x1<<2; //PGA2 的正相输入端口选择内部偏置

21.5. 模块框图

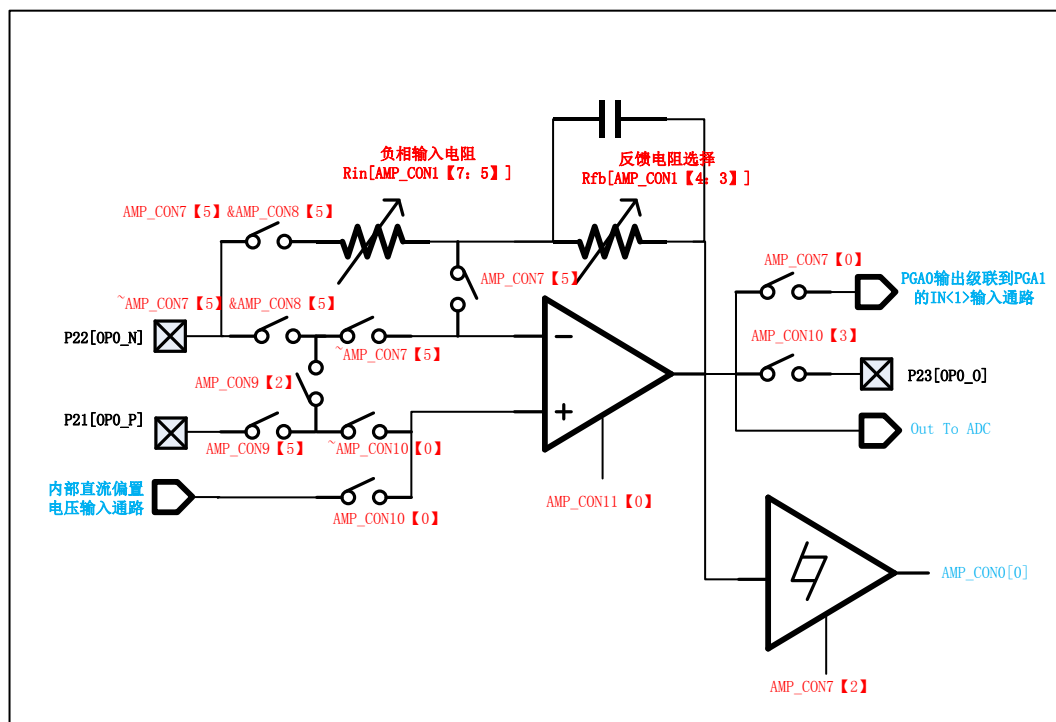


图 21-1 运放 PGA0 电路结构框图

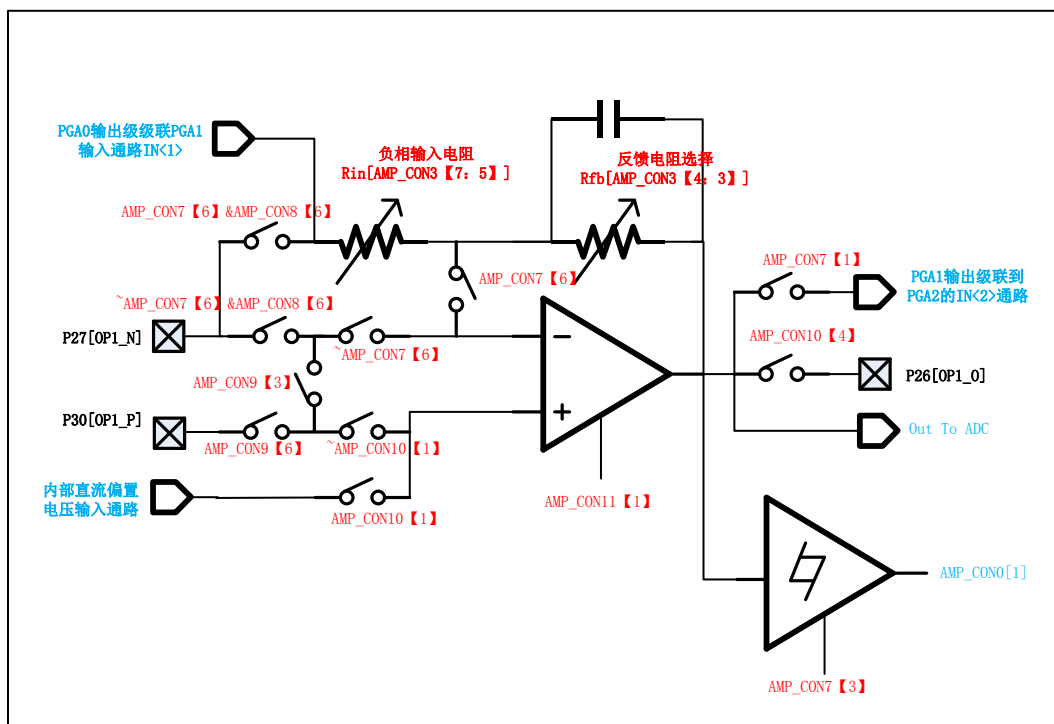


图 21-2 运放 PGA1 电路结构框图

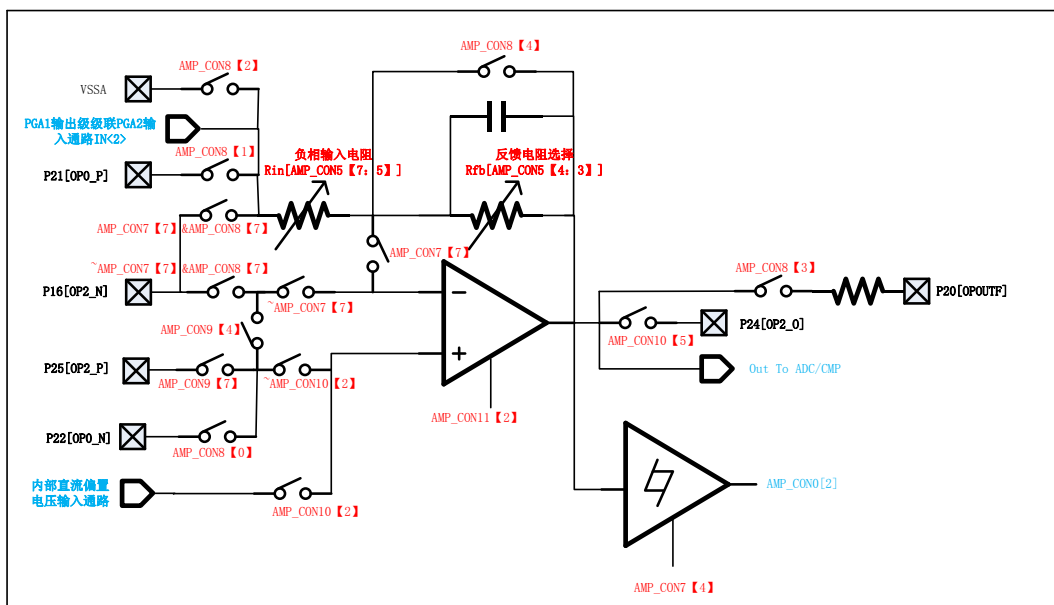


图 21-3 运放 PGA2 电路结构框图

21.6. 寄存器列表

Table 11 AMP register list



Offset	Register Name	Description
0x163 (XSFR)	AMP_CON0	AMP_CON0 register
0x125 (XSFR)	AMP_CON1	AMP_CON1 register
0x126 (XSFR)	AMP_CON2	AMP_CON2 register
0x127 (XSFR)	AMP_CON3	AMP_CON3 register
0x128 (XSFR)	AMP_CON4	AMP_CON4 register
0x129 (XSFR)	AMP_CON5	AMP_CON5 register
0x12A (XSFR)	AMP_CON6	AMP_CON6 register
0x12B (XSFR)	AMP_CON7	AMP_CON7 register
0x12C (XSFR)	AMP_CON8	AMP_CON8 register
0x12D (XSFR)	AMP_CON9	AMP_CON9 register
0x12E (XSFR)	AMP_CON10	AMP_CON10 register
0x12F (XSFR)	AMP_CON11	AMP_CON11 register

21.7. 寄存器详细说明

21.7.1. AMP_CON0

Addr = 0x163 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	—	—	—	—
6	TRIGSEL	运放比较器中断触发边沿选择 0x0: 上升沿触发 0x1: 下降沿触发	RW	0x0
5	AMPCMP2IE	运放 PGA2 的比较器触发中断使能位 0x0: 不使能 0x1: 使能中断	RW	0x0



4	AMPCMP1IE	运放 PGA1 的比较器触发中断使能位 0x0: 不使能 0x1: 使能中断	RW	0x0
3	AMPCMP0IE	运放 PGA0 的比较器触发中断使能位 0x0: 不使能 0x1: 使能中断	RW	0x0
2	AMPCMP2OUT	运放 PGA2 的比较器输出值	RO	0x0
1	AMPCMP1OUT	运放 PGA1 的比较器输出值	RO	0x0
0	AMPCMP0OUT	运放 PGA0 的比较器输出值	RO	0x0

21.7.2. AMP_CON1

Addr = 0x125 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	PGA0RIN	PGA0 内置负相输入电阻选择 0x0: 160K Ω 0x1: 80K Ω 0x2: 40K Ω 0x3: 20K Ω 0x4: 10K Ω 0x5: 5K Ω 0x6: 2.5K Ω 0x7: 1.25K Ω Note: 和 RFB 配合起来在负相输入 PGA 模式获得 0dB 增益, 在正相输入 PGA 模式获得 6dB 增益!	RW	0x1
4: 3	PGA0RFB	PGA0 内置反馈电阻选择 0x0: 80K Ω 0x1: 160K Ω 0x2: 320K Ω 0x3: 640K Ω	RW	0x0
2: 0	PGA0IB	PGA0 的偏置电流选择 BIT0 为运放总电流选择, BIT1 为输出级电流选择, BIT2 为 OFFSET 电流选择; 对应 BIT 置 1 为大电流,	RW	0x1



		0 为小电流; default=1, 即只有运放总电流选择大电流;		

21.7.3. AMP_CON2

Addr = 0x126 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PGA0TRIM	PGA0 的 offset trim 选项 其中 BIT7 为符号位, BIT6: 0 为选择位。在 trim 模式, 如果 AMPCMP0OUT=0, 对应的 BIT7 置 0, BIT6: 0 逐渐增大直至 AMPCMP0OUT=1; 如果 AMPCMP0OUT=1, 对应的 BIT7 置 1, BIT6: 0 逐渐增大直至 AMPCMP0OUT=0;	RW	0x0

21.7.4. AMP_CON3

Addr = 0x127 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	PGA1RIN	PGA1 内置负相输入电阻选择 0x0: 160K Ω 0x1: 80K Ω 0x2: 40K Ω 0x3: 20K Ω 0x4: 10K Ω 0x5: 5K Ω 0x6: 2.5K Ω 0x7: 1.25K Ω Note: 和 RFB 配合起来在负相输入 PGA 模式获得 0dB 增益, 在正相输入 PGA 模式获得 6dB 增益!	RW	0x1
4: 3	PGA1RFB	PGA1 内置反馈电阻选择	RW	0x0



		0x0: 80K Ω 0x1: 160K Ω 0x2: 320K Ω 0x3: 640K Ω		
2: 0	PGA1IB	PGA1 的偏置电流选择 BIT0 为运放总电流选择, BIT1 为输出级电流选择, BIT2 为 OFFSET 电流选择; 对应 BIT 置 1 为大电流, 0 为小电流; default=1, 即只有运放总电流选择 大电流;	RW	0x1

21.7.5. AMP_CON4

Addr = 0x128 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PGA1TRIM	PGA1 的 offset trim 选项 其中 BIT7 为符号位, BIT6: 0 为选择位。在 trim 模式, 如果 AMPCMP1OUT=0, 对应的 BIT7 置 0, BIT6: 0 逐渐增大直至 AMPCMP1OUT=1; 如果 AMPCMP1OUT=1, 对应的 BIT7 置 1, BIT6: 0 逐渐 增大直至 AMPCMP1OUT=0;	RW	0x0

21.7.6. AMP_CON5

Addr = 0x129 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 5	PGA2RIN	PGA2 内置负相输入电阻选择 0x0: 160K Ω 0x1: 80K Ω 0x2: 40K Ω 0x3: 20K Ω	RW	0x1



		0x4: 10K Ω 0x5: 5K Ω 0x6: 2.5K Ω 0x7: 1.25K Ω Note: 和 RFB 配合起来在负相输入 PGA 模式获得 0dB 增益，在正相输入 PGA 模式获得 6dB 增益！		
4: 3	PGA2RFB	PGA2 内置反馈电阻选择 0x0: 80K Ω 0x1: 160K Ω 0x2: 320K Ω 0x3: 640K Ω	RW	0x0
2: 0	PGA2IB	PGA2 的偏置电流选择 BIT0 为运放总电流选择, BIT1 为输出级电流选择, BIT2 为 OFFSET 电流选择; 对应 BIT 置 1 为大电流, 0 为小电流; default=1, 即只有运放总电流选择 大电流;	RW	0x1

21.7.7. AMP_CON6

Addr = 0x12A (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	PGA2TRIM	PGA2 的 offset trim 选项 其中 BIT7 为符号位, BIT6: 0 为选择位。在 trim 模式, 如果 AMPCMP2OUT=0, 对应的 BIT7 置 0, BIT6: 0 逐渐增大直至 AMPCMP2OUT=1; 如果 AMPCMP2OUT=1, 对应的 BIT7 置 1, BIT6: 0 逐渐增大直至 AMPCMP2OUT=0;	RW	0x0

21.7.8. AMP_CON7

Addr = 0x12B (XSFR)



Bit(s)	Name	Description	R/W	Reset
7	AMP2FBSEL	AMP2 内部反馈通路选择 0x0: 使用外部输入电阻和反馈电阻, 即 OP 模式; 0x1: 使用内部输入电阻和反馈电阻, 即 PGA 模式;	RW	0x1
6	AMP1FBSEL	AMP1 内部反馈通路选择 0x0: 使用外部输入电阻和反馈电阻, 即 OP 模式; 0x1: 使用内部输入电阻和反馈电阻, 即 PGA 模式;	RW	0x0
5	AMP0FBSEL	AMP0 内部反馈通路选择 0x0: 使用外部输入电阻和反馈电阻, 即 OP 模式; 0x1: 使用内部输入电阻和反馈电阻, 即 PGA 模式;	RW	0x0
4	AMP2CMPMDEN	AMP2 内置比较器模式使能 0x0: 不使能比较器模式 0x1: 使能比较器模式	RW	0x0
3	AMP1CMPMDEN	AMP1 内置比较器模式使能 0x0: 不使能比较器模式 0x1: 使能比较器模式	RW	0x0
2	AMP0CMPMDEN	AMP0 内置比较器模式使能 0x0: 不使能比较器模式 0x1: 使能比较器模式	RW	0x0
1	PGA1TO2EN	PGA1 输出到 PGA2 的输入使能 0x0: 不使能 0x1: 使能	RW	0x0
0	PGA0TO1EN	PGA0 输出到 PGA1 的输入使能 0x0: 不使能 0x1: 使能	RW	0x0

21.7.9. AMP_CON8

Addr = 0x12C (XSFR)

Bit(s)	Name	Description	R/W	Reset



7	AMP2IN2IO	AMP2 负相输入端到 IO 的通路选择使能 0x0: 不使能 0x1: 使能	RW	0x0
6	AMP1IN2IO	AMP1 负相输入端到 IO 的通路选择使能 0x0: 不使能 0x1: 使能	RW	0x0
5	AMP0IN2IO	AMP0 负相输入端到 IO 的通路选择使能 0x0: 不使能 0x1: 使能	RW	0x0
4	AMPPGA2IO	PGA2 的 OUTI 选通到 PGA2 的 VINI 使能 0x0: 不使能 0x1: 使能	RW	0x0
3	PGA2OUT10KEN	PGA2 的 OUT 输出通过 10K 电阻到 PGAOUTF 使能 0x0: 不使能 0x1: 使能	RW	0x0
2	AVSSTOPGA2INEN	PGA2 的 IN 选通到 AVSS (VSSPGA) 使能 0x0: 不使能 0x1: 使能	RW	0x0
1	PGA0IPTOPGA2INEN	PGA2 的 IN 选通到 PGA0 的 IP 使能 0x0: 不使能 0x1: 使能	RW	0x1
0	PGA0INTOPGA2IPEN	PGA2 的 IP 选通到 PGA0 的 IN 使能 0x0: 不使能 0x1: 使能	RW	0x1

21.7.10. AMP_CON9

Addr = 0x12D (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	AMP2IP2IOEN	AMP2 正相输入端到 IO 的通路选择使能 0x0: 不使能	RW	0x0



		0x1: 使能		
6	AMP1IP2IOEN	AMP1 正相输入端到 IO 的通路选择使能 0x0: 不使能 0x1: 使能	RW	0x0
5	AMP0IP2IOEN	AMP0 正相输入端到 IO 的通路选择使能 0x0: 不使能 0x1: 使能	RW	0x0
4	AMP2IP2IN	AMP2 内部正负输入端短接功能, 用于 offset 校正. 0x0: 不短接 0x1: 短接	RW	0x0
3	AMP1IP2IN	AMP1 内部正负输入端短接功能, 用于 offset 校正. 0x0: 不短接 0x1: 短接	RW	0x0
2	AMP0IP2IN	AMP0 内部正负输入端短接功能, 用于 offset 校正. 0x0: 不短接 0x1: 短接	RW	0x0
1	AMPLPEN	AMP 模块低功耗模式使能 (功耗<1uA) 0x0: 正常模式 0x1: 低功耗模式	RW	0x0
0	AMPPGAIB	PGA 总偏置电流选择 0x0: 1X 0x1: 2X	RW	0x0

21.7.11. AMP_CON10

Addr = 0x12E (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 6	—	—	—	—
5	AMP2OUT2IOEN	AMP2 输出端到 IO 的通路选择使能	RW	0x0



		0x0: 不使能 0x1: 使能		
4	AMP1OUT2IOEN	AMP1 输出端到 I0 的通路选择使能 0x0: 不使能 0x1: 使能	RW	0x0
3	AMP0OUT2IOEN	AMP0 输出端到 I0 的通路选择使能 0x0: 不使能 0x1: 使能	RW	0x0
2	AMP2IPSEL	AMP2 正相输入端口选择 0x0: 外部 I0 输入（注意对应的 IP2IO 要使能） 0x1: 内部直流偏置电压输入通路	RW	0x0
1	AMP1IPSEL	AMP1 正相输入端口选择 0x0: 外部 I0 输入（注意对应的 IP2IO 要使能） 0x1: 内部直流偏置电压输入通路	RW	0x0
0	AMP0IPSEL	AMP0 正相输入端口选择 0x0: 外部 I0 输入（注意对应的 IP2IO 要使能） 0x1: 内部直流偏置电压输入通路	RW	0x0

21.7.12. AMP_CON11

Addr = 0x12F (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 3	—	—	—	—
2	PGA2EN	PGA2 使能 0x0: 不使能 0x1: 使能	RW	0x0
1	PGA1EN	PGA1 使能 0x0: 不使能 0x1: 使能	RW	0x0
0	PGA0EN	PGA0 使能 0x0: 不使能 0x1: 使能	RW	0x0



22. LED 模块

22.1. 功能概述

- 最大支持 8 个 com, 12 个 seg
- 灵活的使能控制, 支持每个 com 和 seg 单独使能控制
- 支持双 buffer, 可以维护两页数据
- 支持帧中断
- 支持扫描频率配置
- 通过内部 DMA 读取显示数据

22.2. 功能框图

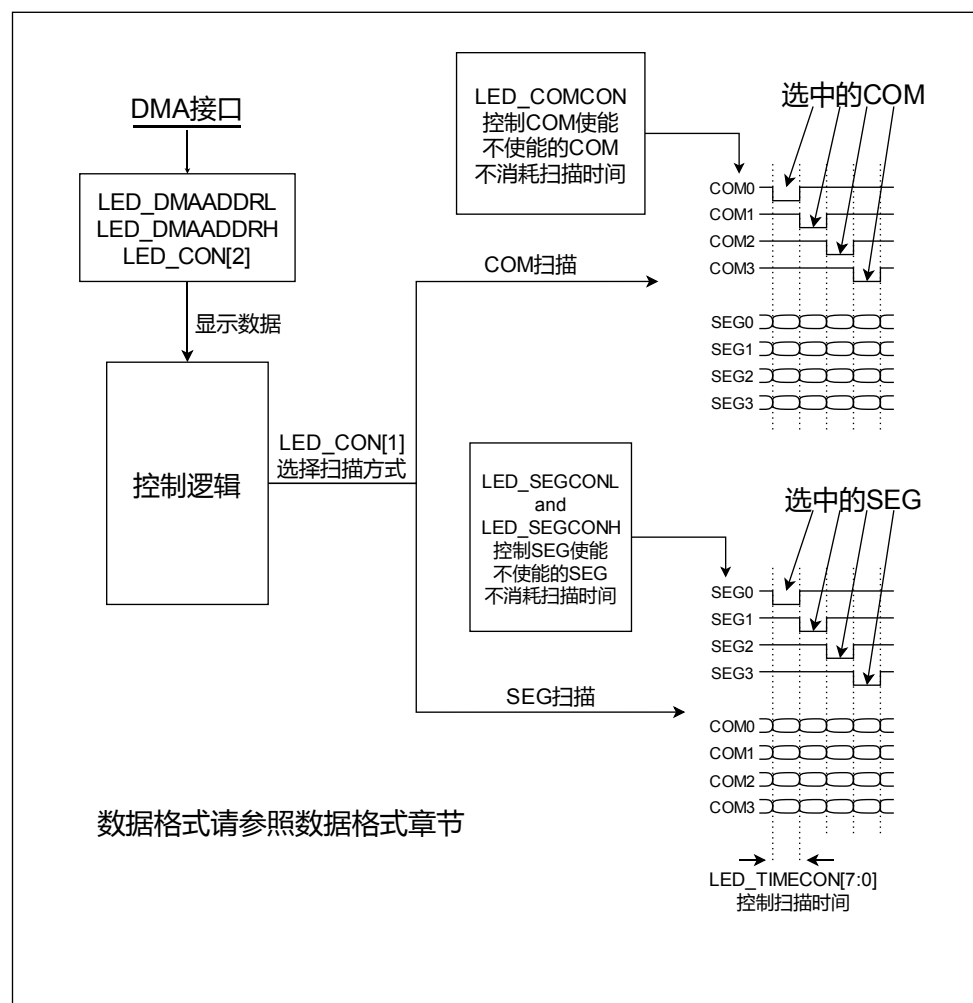


图 22-1 LED 模块功能框图

22.3. 数据结构

22.3.1. COM 扫描的数据结构

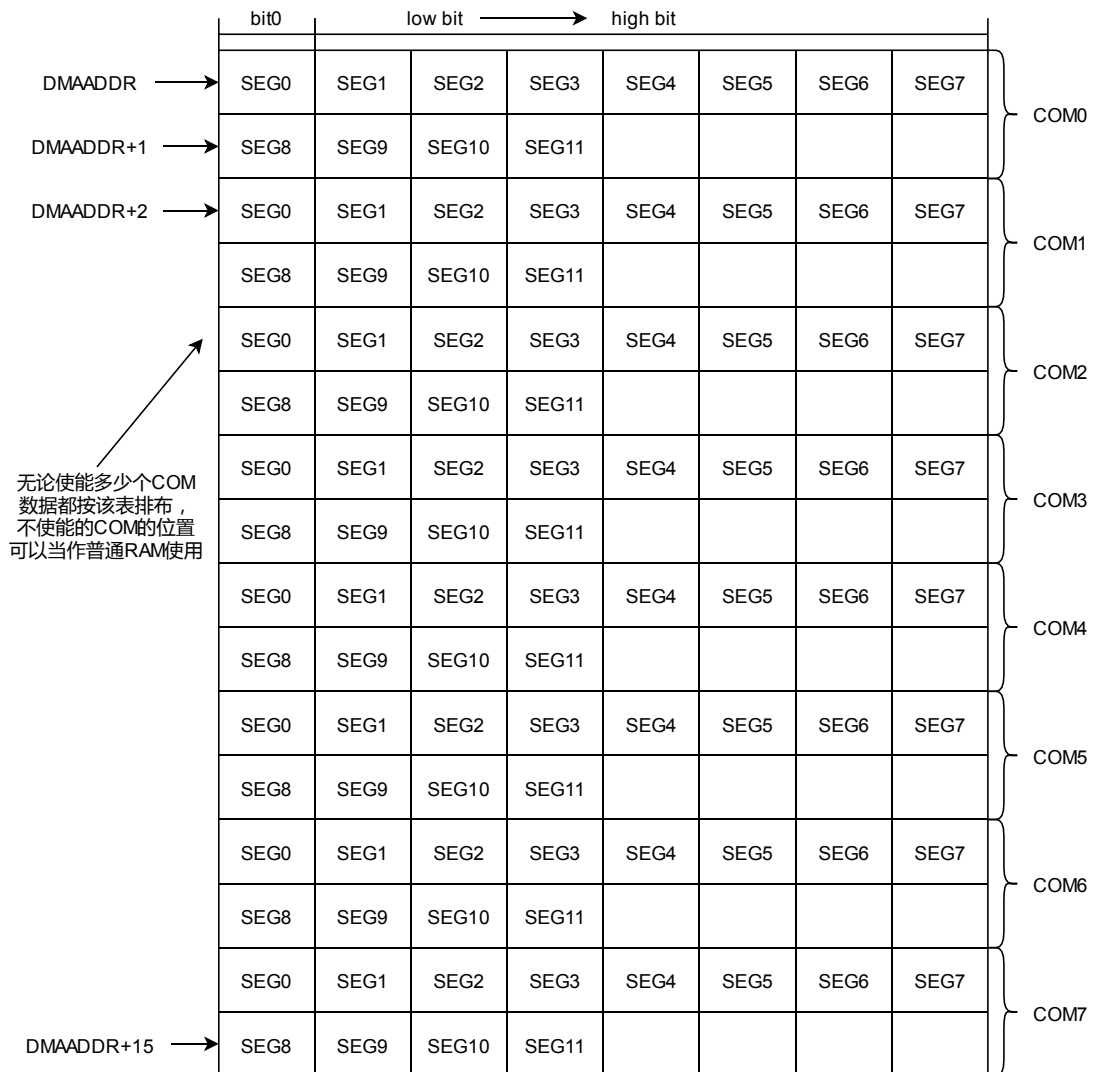


图 22-2 COM 扫描的数据结构

通过 LED_DMAADDRL 和 LED_DMAADDRH 来配置 LED 数据表的首地址，配置完首地址就确定了整个表的位置，可配置 LED_CON[2]来让 DMA 的首地址向后偏移 16byte。上表为 COM 扫描的数据储存格式。COM 扫描时选中的 COM 为低电平。设置 SEG 的值为 1 则 SEG 推高电平，设置为 0 则推低电平。不使能某个 COM 时，该 COM 的储存空间可以留作它用。



22.3.2. SEG 扫描的数据结构

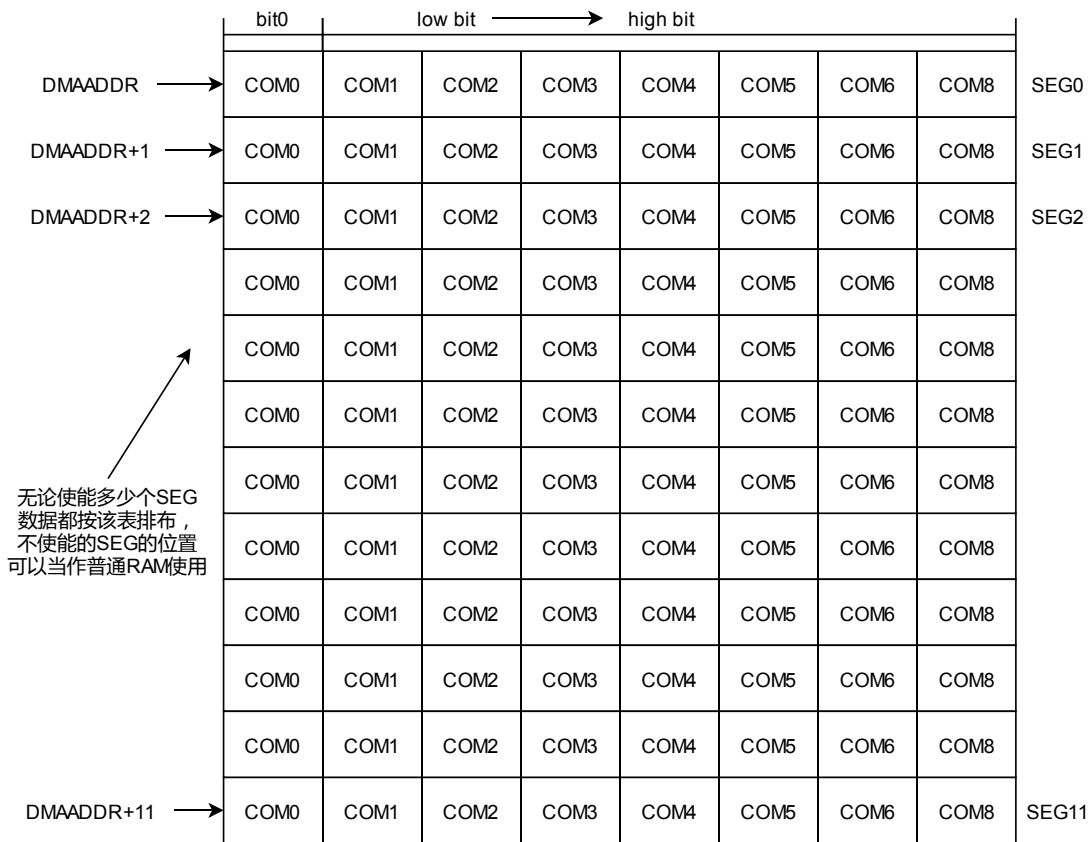


图 22-3 SEG 扫描的数据结构

通过 LED_DMAADDRL 和 LED_DMAADDRH 来配置 LED 数据表的首地址，配置完首地址就确定了整个表的位置，可配置 LED_CON[2]来让 DMA 的首地址向后偏移 16byte。上表为 SEG 扫描的数据储存格式。SEG 扫描时选中的 SEG 为低电平。设置 COM 的值为 1 则 COM 推高电平，设置为 0 则推低电平。不使能某个 SEG 时，该 SEG 的储存空间可以留作它用。

22.4. 寄存器列表

表 22-1 SPI0 register list

Address	Register Name	Description
0x07 (XSFR)	LED_SEGCONL	LED SEG enable control register
0x08 (XSFR)	LED_SEGCONH	LED SEG enable control register
0x09 (XSFR)	LED_COMCON	LED COM enable control register
0x0A (XSFR)	LED_CON	LED COM enable control register



0x0B (XSFR)	LED_TIMECON	LED display timing control register
0x0C (XSFR)	LED_DMAADDRRL	LED display data DMA start address register
0x0D (XSFR)	LED_DMAADDRH	LED display data DMA start address register

22.5. 寄存器详细说明

22.5.1. LED_SEGCONL

Addr = 0x07 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	SEGCONL	SEG 使能控制器，每一 bit 控制一个 SEG. bit7: 控制 SEG7 使能和关闭 bit6: 控制 SEG6 使能和关闭 ... bit0: 控制 SEG0 使能和关闭 Note: 每个控制 bit 置 1，则使能；写 0，则关闭使能。	RW	0x0

22.5.2. LED_SEGCONH

Addr = 0x08 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 4	—	—	—	—
3: 0	SEGCONH	SEG 使能控制器，每一 bit 控制一个 SEG. bit3: 控制 SEG11 使能和关闭 bit2: 控制 SEG10 使能和关闭 bit1: 控制 SEG9 使能和关闭 bit0: 控制 SEG8 使能和关闭 Note: 每个控制 bit 置 1，则使能；写 0，则关闭使能。	RW	0x0



22.5.3. LED_COMCON

Addr = 0x09 (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	COMCON	COM 使能控制器，每一 bit 控制一个 COM. bit7: 控制 COM7 bit6: 控制 COM6 ... bit0: 控制 COM0 Note: 每个控制 bit 置 1，则使能；写 0，则关闭使能。	RW	0x0

22.5.4. LED_CON

Addr = 0x0A (XSFR)

Bit(s)	Name	Description	R/W	Reset
7	PENDING	LED 扫完一帧的 PENDING 标志位，每扫完一帧都会置 1，写 1 清零	RC	0x0
6: 4	—	—	—	—
3	INTEN	LED 扫完一帧中断使能位。 0x0: 不使能 0x1: 使能	RW	0x0
2	DMAADDRSEL	这一位控制 DMA 拿数据的地址的偏移量。 0x0: 不偏移 0x1: 地址向后偏移 16byte (此位起到一个双 buffer 类似的作用)	RW	0x0
1	COMSEGSEL	扫描方式选择位。 0x0: com 扫描 0x1: seg 扫描	RW	0x0
0	LEDEN	LED 使能位 使能之后从 DMAADDRH 和 DMAADDRL 的地址开始拿数据 扫描点亮，用户将数据写到对应的地址即可。	RW	0x0



22.5.5. LED_TIMECON

Addr = 0x0B (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	TIMECON	扫描到某个 com 或 seg 时点亮的时间, 步长为 32 微秒.	RW	0x0

22.5.6. LED_DMAADDRL

Addr = 0x0C (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	DMAADDRL	LED 数据储存首地址的低 8 位	WO	0x0

22.5.7. LED_DMAADDRH

Addr = 0x0D (XSFR)

Bit(s)	Name	Description	R/W	Reset
7: 0	DMAADDRH	LED 数据储存首地址的高 8 位	WO	0x0

22.6. 使用流程说明

- 1) 配置 COM 或 SEG 的单次扫描时间(配置 LED_TIMECON)
- 2) 配置 LED 显示数据的首地址(配置 LED_DMAADDRH 和 LED_DMAADDRL)
- 3) 将 LED 显示数据写到首地址对应的地址
- 4) 使能 LED