深圳富芯电子科技有限公司

RC6F800X

数据手册

版本 V2.6

1 总体概述

本芯片是一款高性能的 8 位单片机。内部包含 4K 字节 Flash，256 字节 SRAM，1 个 8 位定时器、2 个 16-bit 定时器/计数器，具有独立时钟的看门狗，12-bit ADC，2 个模拟比较器，1 路 UART 和 1 路 I2C 通讯接口，片内 POR，BOR 和 LVD，内部 16MHz RC 振荡器和 32KHz WDT 振荡器，支持外部 32.768KHz 晶振。具有两种低功耗模式。

2 主要功能

内核：

- 超高速 8051 内核（1T）
- 指令周期可配：
  - 4MHz，VDD≥2.4V
  - 16MHz，VDD≥4.5V

工作电压：2.4V~5.5V
工作温度：-40℃~85℃
Flash ROM：4K 字节 Flash ROM(擦写次数典型值 1000 次)
SRAM：256 字节 SRAM
时钟：

- 内建高频 16MHz RC 振荡器(可微调)
  - 作为系统时钟源时，可通过寄存器设定为 8/4/2/1MHz 等
  - 误差不超过±2.5%(-40℃~85℃)
- 内部 32KHz 低速 RC(误差不超过±10%)
- 外部 32.768 KHz 低速晶振

复位：

- 上电复位、复位脚复位、看门狗溢出复位
- 欠压复位(2.25V、2.6V、4.2V)

低电压检测：LVD 共 4 级(2.3V、2.5V、2.7V、4.3V)
中断(INT)：

- Timer0、Timer1、Timer2、SCK3、WDT、ADC、UART、I2C、LVD、CMP0~1 、PT0~PT2 共 14 个中断源，全部 GPIO 可设上升沿、下降沿、双沿中断

数字外设：

- 1 个 8 位基本定时器
  - 预分频 1,2,4,8,16,32,64,128
- 2 个 16 位高级定时器，支持 4 路 PWM 输出功能
  - 支持捕获和刹车功能
  - 支持周期中断和占空比中断
- 1 个 16 位看门狗定时器
- 1 个 UART
- 1 个 I2C：支持主机模式和从机模式
  - 速率 100KHz/400KHz

12 位 ADC:

- 外部输入：18 路
- 参考源：外部参考(PT10)，内部参考 1.2V 和 2V，电源参考
- 采样可以通过 PWM 的上升沿或者下降沿触发

模拟比较器：

- 2 个模拟比较器
  - 6 位的 DAC，参考电压来自 VBG（1.2V）

18 个 GPIO:

- P0[7:0]、P1[7:0]、P2[1:0]
- PT00、PT13、PT14 默认开漏上拉输出，其余 I/O 默认为输入高阻态
- 所有 IO 可单独配置上下拉 10K 电阻(匹配精度 5%)
  - 上下拉可同时打开
- 2 个大电流驱动 IO(灌电流可达 100mA,拉电流可达 65mA)

省电模式：

- 深度休眠可由看门狗复位、睡眠定时器中断、引脚中断唤醒
- 深度休眠电流：3.2uA(典型值)

Flash 烧写：

- 5 线烧写(VDD,GND,SDA,SCL,VPP)

封装：

- SSOP20/QFN20/SOP16

回流焊：

- 建议回流焊最高温度设置为 245±5℃，持续时间 10 秒，可参考回流焊温度曲线
  注：受生产工艺限制，超过 245±5℃，部分芯片频率偏移会超标（±2%）

目　录

3 系统功能框图及脚位图

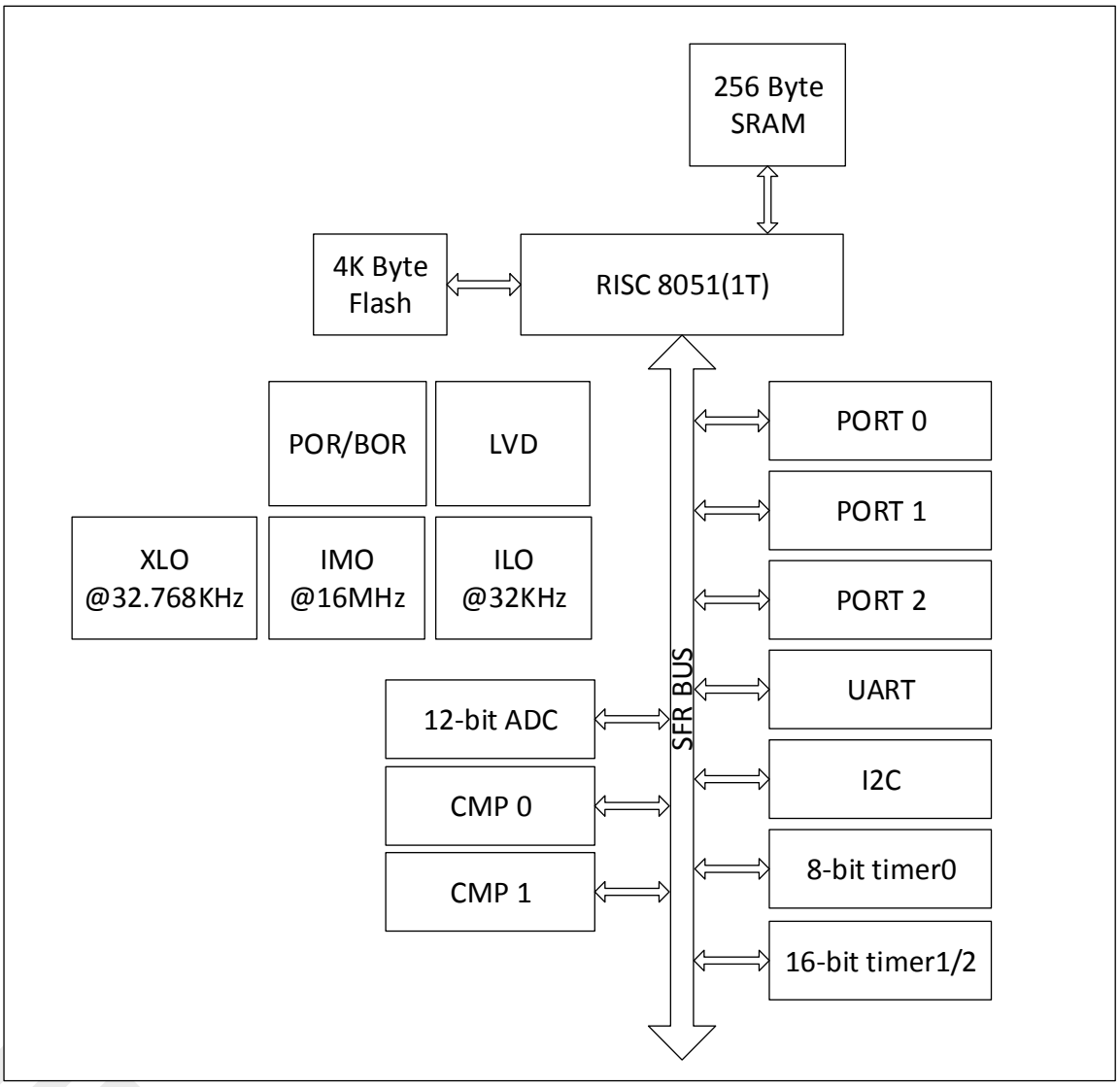


图 3-1 系统功能结构框图

3.1 脚位图

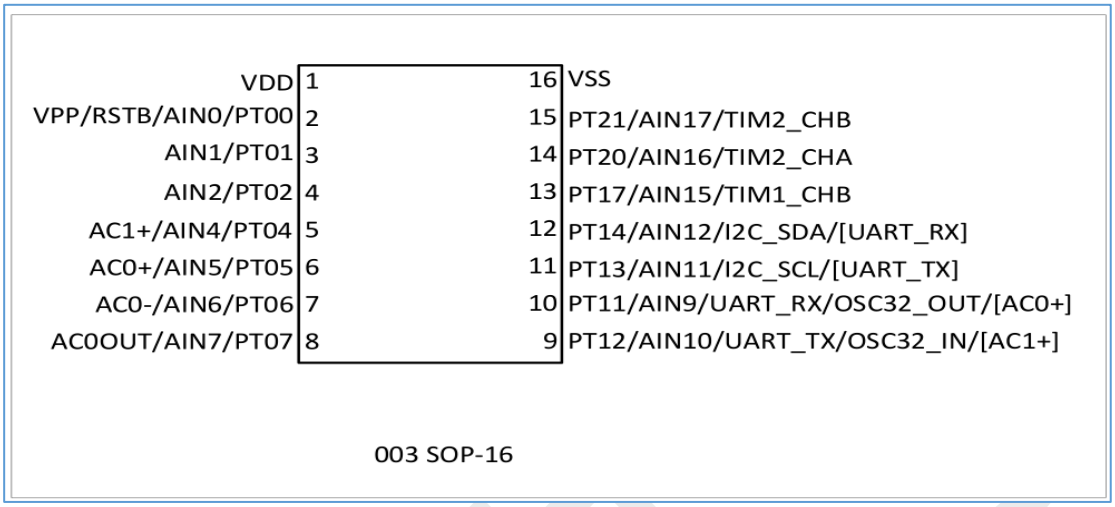


图 3-2 SOP16 封装脚位图

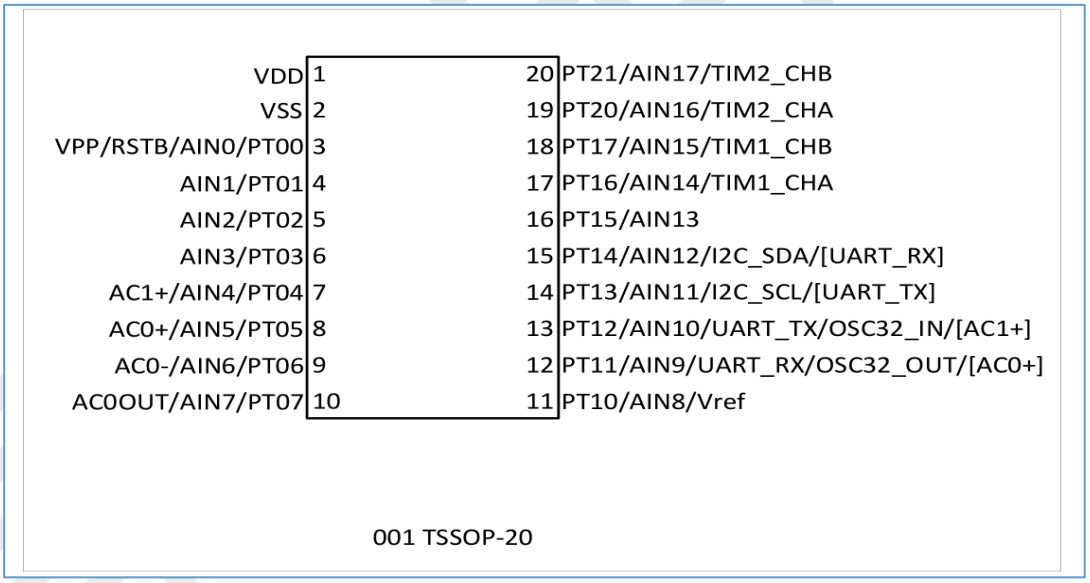


图 3-3 TSSOP20 封装脚位图

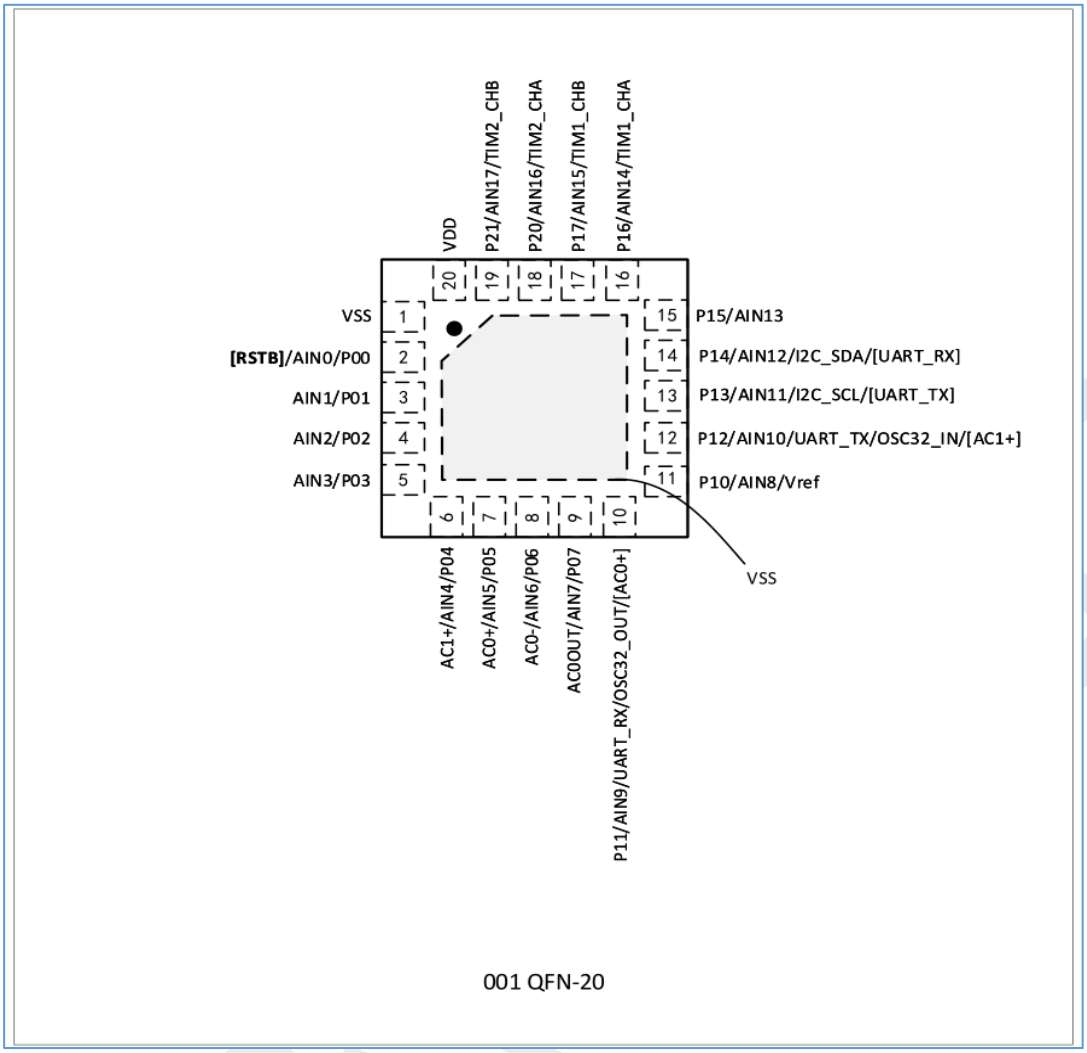


图 3-4 QFN20 封装脚位图

注意：

1. UART_RX 和 UART_TX 可以选择两个不同的位置输出
2. 所有的 GPIO 都可以产生中断。

3.2 引脚描述

引脚名字	I/O	功能描述	复位状态	复用功能
VDD	A	电源	-	无
GND	A	地	-	无
PT00	I/O	PT0 口的每个口都可以设置为输入或者输出模式 输入模块可以使能内部上拉 输出模块可以设置开漏输出	PT00 默认为开漏输出且上拉打开，其他引脚默认为高阻输出	AIN0/RST/VPP
PT01				AIN1
PT02				AIN2
PT03				AIN3
PT04				AIN4/AC1+

PT05				AIN5/AC0+
PT06				AIN6/AC0-
PT07				AIN7/AC0OUT
PT10	I/O	PT1 口的每个口都可以设置为输入或者输出模式 输入模块可以使能内部上拉 输出模块可以设置开漏输出 （注意：上电 3MS，I2C_SDA 脚会输出 60uS 左右的低电平）	PT13 和 PT14 默认为开漏输出且上拉打开，其他引脚默认为高阻输出	AIN8/Vref(外部参考电压)
PT11				AIN9/UART_RX/OSC32_OUT [AC0+]
PT12				AIN10/UART_TX/OSC32_IN [AC1+]
PT13				AIN11/I2C_SCL/[UART_TX]
PT14				AIN12/I2C_SDA/[UART_RX]
PT15				AIN13/BKIN
PT16				AIN14/TIM1_CHA
PT17				AIN15/TIM1_CHB
PT20	I/O	PT2 口的每个口都可以设置为输入或者输出模式 输入模块可以使能内部上拉 输出模块可以设置开漏输出 该组端口的输出能力可以达到 100mA	高阻输出	AIN16/TIM2_CHA
PT21				AIN17/TIM2_CHB

4 GPIO

4.1 GPIO 结构框图

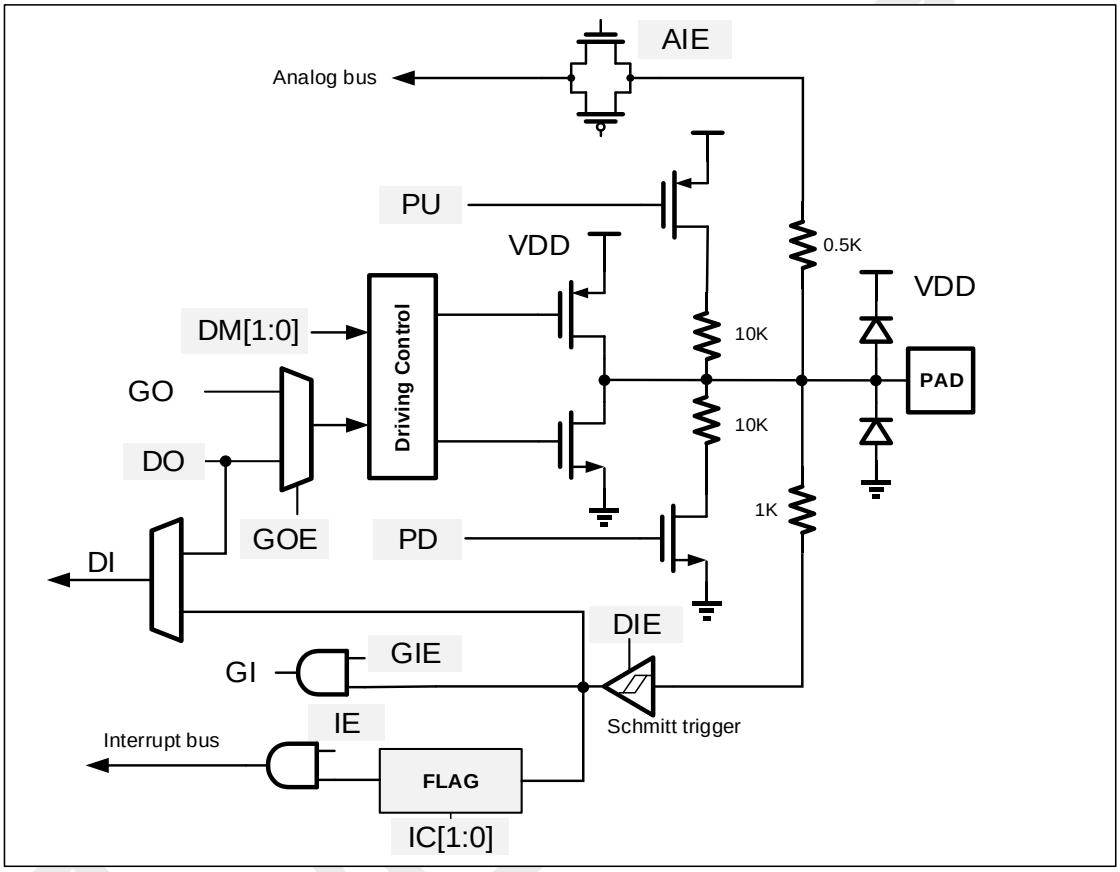


图 4-1 GPIO 结构图

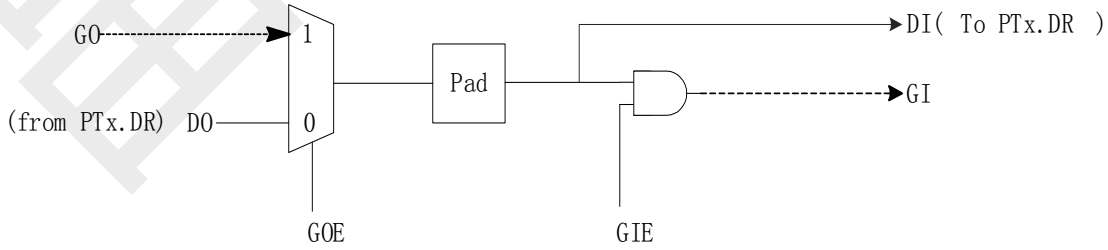


图 4-2 GPIO 复用结果说明

4.2 配置 I/O 口

每个 I/O 的配置都需要使用两个寄存器进行配置。

以 PT0 口为例，配置 PT0 口需要使用 PT0_DM0 和 PT0_DM1 两个寄存器进行配置，如下图所示：

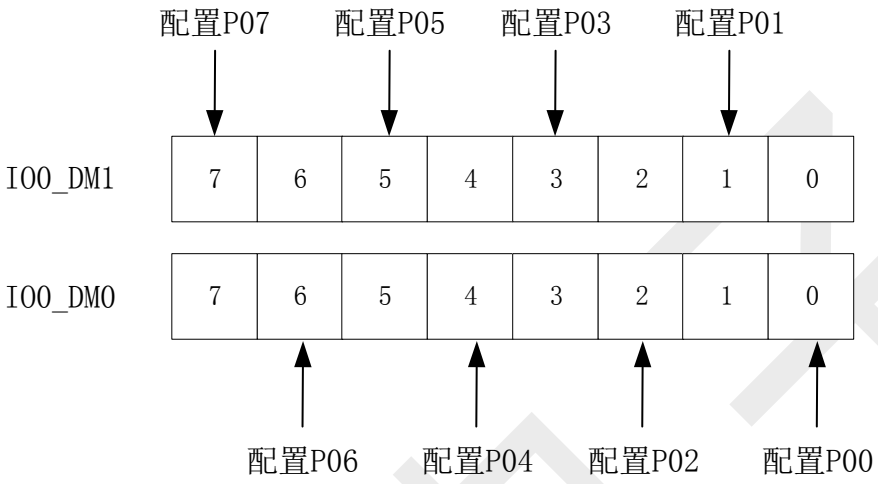


图 4-3 I/O 配置图

即 PT0_DM0 的第 0 位和 PT0_DM1 的第 0 位合起来配置 PT00 的模式；

即 PT0_DM0 的第 1 位和 PT0_DM1 的第 1 位合起来配置 PT01 的模式；

GPIO 模式的配置说明如下表和下图所示

PTx_DM1	PTx_DM0	驱动模式
0	0	配置 PTx 的对应 I/O 为高阻输出
0	1	配置 PTx 的对应 I/O 为强推挽输出
1	0	配置 PTx 的对应 I/O 为开漏高输出
1	1	配置 PTx 的对应 I/O 为开漏低输出

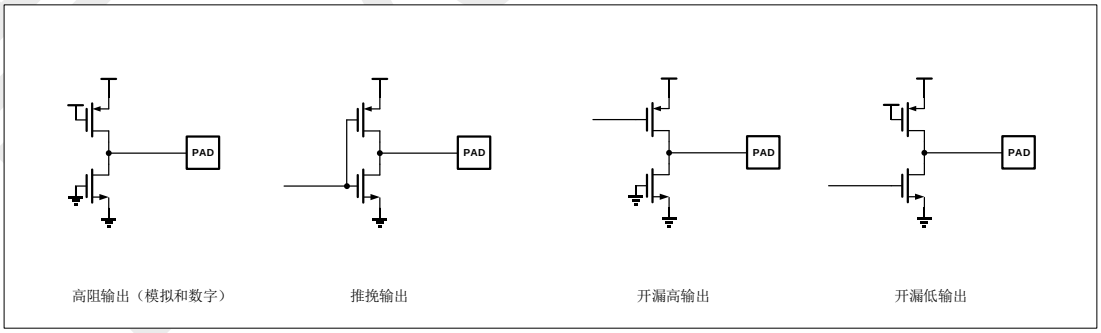


图 4-4 GPIO 驱动模式

4.3 与 GPIO 相关的寄存器定义

名字	地址	读写	复位值	描述
PT0_DR	0xFF20	读写	00000000	端口 0 数据寄存器
PT0_GIE	0xFF21	读写	00000001	端口 0 数字复用输入使能寄存器
PT0_GOE	0xFF22	读写	00000001	端口 0 数字复用输出使能寄存器
PT0_DM0	0xFF23	读写	00000001	端口 0 模式 0 位
PT0_DM1	0xFF24	读写	00000001	端口 0 模式 1 位
PT0_DIE	0xFF25	读写	00000001	端口 0 数字输入使能
PT0_AIE	0xFF26	读写	00000000	端口 0 模拟输入使能
PT0_PU	0xFF27	读写	00000001	端口 0 上拉控制寄存器
PT0_PD	0xFF28	读写	00000000	端口 0 下拉控制寄存器
PT0_IE	0xFF29	读写	00000000	端口 0 中断使能寄存器
PT0_IC0	0xFF2A	读写	00000000	端口 0 中断控制 0 位
PT0_IC1	0xFF2B	读写	00000000	端口 0 中断控制 1 位
PT0_FLAG	0xFF2C	读写	00000000	端口 0 中断标志位
PT1_DR	0xFF30	读写	00000000	端口 1 数据寄存器
PT1_GIE	0xFF31	读写	00011000	端口 1 数字复用输入使能寄存器
PT1_GOE	0xFF32	读写	00011000	端口 1 数字复用输出使能寄存器
PT1_DM0	0xFF33	读写	00011000	端口 1 模式 0 位
PT1_DM1	0xFF34	读写	00011000	端口 1 模式 1 位
PT1_DIE	0xFF35	读写	00011000	端口 1 数字输入使能
PT1_AIE	0xFF36	读写	00000000	端口 1 模拟输入使能
PT1_PU	0xFF37	读写	00011000	端口 1 上拉控制寄存器
PT1_PD	0xFF38	读写	00000000	端口 1 下拉控制寄存器
PT1_IE	0xFF39	读写	00000000	端口 1 中断使能寄存器
PT1_IC0	0xFF3A	读写	00000000	端口 1 中断控制 0 位
PT1_IC1	0xFF3B	读写	00000000	端口 1 中断控制 1 位
PT1_FLAG	0xFF3C	读写	00000000	端口 1 中断标志位
PT2_DR	0xFF40	读写	00000000	端口 2 数据寄存器
PT2_GIE	0xFF41	读写	00000000	端口 2 数字复用输入使能寄存器
PT2_GOE	0xFF42	读写	00000000	端口 2 数字复用输出使能寄存器
PT2_DM0	0xFF43	读写	00000000	端口 2 模式 0 位

PT2_DM1	0xFF44	读写	00000000	端口 2 模式 1 位
PT2_DIE	0xFF45	读写	00000000	端口 2 数字输入使能
PT2_AIE	0xFF46	读写	00000000	端口 2 模拟输入使能
PT2_PU	0xFF47	读写	00000000	端口 2 上拉控制寄存器
PT2_PD	0xFF48	读写	00000000	端口 2 下拉控制寄存器
PT2_IE	0xFF49	读写	00000000	端口 2 中断使能寄存器
PT2_IC0	0xFF4A	读写	00000000	端口 2 中断控制 0 位
PT2_IC1	0xFF4B	读写	00000000	端口 2 中断控制 1 位
PT2_FLAG	0xFF4C	读写	00000000	端口 2 中断标志位

4.3.1 PT0_DR (0xFF20)

Bit	7	6	5	4	3	2	1	0
Name	PT0_DR							
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	PT0_DR	端口 0 的数据寄存器，写该寄存器会更新端口输出，读该寄存器得到端口输出值。 注意：如果端口配置为强推挽输出或者开漏输出，那么读端口得到端口寄存器的值；如果端口配置为高阻模式，数字输入使能打开，那么读端口得到端口电平。

4.3.2 PT0_GIE (0xFF21)

Bit	7	6	5	4	3	2	1	0
Name	GIE.7	GIE.6	GIE.5	GIE.4	GIE.3	GIE.2	GIE.1	GIE.0
Reset	0	0	0	0	0	0	0	1
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	GIE.x	端口 0 的复用输入使能。 0 关闭复用输入使能 1 打开复用输入使能

4.3.3 PT0_GOE (0xFF22)

Bit	7	6	5	4	3	2	1	0
Name	GOE.7	-	-	-	-	-	-	GOE.0
Reset	0	-	-	-	-	-	-	1
Type	R/W	-	-	-	-	-	-	R/W

Bit	Name	Function
7:0	GOE.x	端口 0 的复用输出使能。 0 关闭复用输出使能，输出由 PTx.DR 决定 1 打开复用输出使能，输出由 GDO 决定

4.3.4 PT0_DM0 (0xFF23)

Bit	7	6	5	4	3	2	1	0
Name	PT0_DM0							
Reset	0x01							
Type	R/W							

Bit	Name	Function
7:0	PT0_DM0	P0 模式控制寄存器。

4.3.5 PT0_DM1 (0xFF24)

Bit	7	6	5	4	3	2	1	0
Name	PT0_DM1							
Reset	0x01							
Type	R/W							

Bit	Name	Function
7:0	PT0_DM1	P0 模式控制寄存器。

4.3.6 PT0_DIE (0xFF25)

Bit	7	6	5	4	3	2	1	0
Name	PT0_DIE							
Reset	0x01							
Type	R/W							

Bit	Name	Function
7:0	PT0_DIE	数字端口使能寄存器 0 关闭数字输入，高阻模式下读端口得到 0 1 打开数字输入

4.3.7 PT0_AIE (0xFF26)

Bit	7	6	5	4	3	2	1	0
Name	PT0_AIE							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	PT0_AIE	模拟端口使能： 0 关闭模拟通道 1 打开模拟通道

4.3.8 PT0_PU (0xFF27)

Bit	7	6	5	4	3	2	1	0
Name	PT0_PU							
Reset	0x01							
Type	R/W							

Bit	Name	Function
7:0	PT0_PU	上拉使能： 0 关闭上拉 1 打开上拉

4.3.9 PT0_PD (0xFF28)

Bit	7	6	5	4	3	2	1	0
Name	PT0_PD							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	PT0_PD	下拉使能： 0 关闭下拉 1 打开下拉

4.3.10 PT0_IE (0xFF29)

Bit	7	6	5	4	3	2	1	0
Name	PT0_IE							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	PT0_IE	中断使能： 0 关闭端口中断 1 打开端口中断

4.3.11 PT0_IC0/PT0_IC1 (0xFF2A/0xFF2B)

Bit	7	6	5	4	3	2	1	0
Name	P0_IC0							
Reset	1	1	1	1	1	1	1	1
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Name	P0_IC1							
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	P0_IC0	[P0_IC1：P0_IC0] 中断触发模式控制： 00 保留 01 下降沿中断 10 上升沿中断 11 上下沿中断
7:0	P0_IC1	

		注意：IO 中断使能后需要读一下 IO 状态，作为后一状态对比； 如果使用的是上下沿中断，在中断函数中也需要读一下 IO 状态。

4.3.12 PT0_FLAG (0xFF2C)

Bit	7	6	5	4	3	2	1	0
Name	P0_FLAG							
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	P0_FLAG	中断标志： 0 没有中断发生 1 有中断发生 写任何值清除该中断标志

4.3.13 PT1_DR (0xFF30)

Bit	7	6	5	4	3	2	1	0
Name	PT1_DR							
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	PT1_DR	端口 1 的数据寄存器，写该寄存器会更新端口输出，读该寄存器得到端口输出值。 注意：如果端口配置为强推挽输出或者开漏输出，那么读端口得到端口寄存器的值；如果端口配置为高阻模式，数字输入使能打开，那么读端口得到端口电平。

4.3.14 PT1_GIE (0xFF31)

Bit	7	6	5	4	3	2	1	0
Name	GIE.7	GIE.6	GIE.5	GIE.4	GIE.3	GIE.2	GIE.1	GIE.0
Reset	0	0	0	1	1	0	0	1
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	GIE.x	端口 1 的复用输入使能。 0 关闭复用输入使能 1 打开复用输入使能

4.3.15 PT1_GOE (0xFF32)

Bit	7	6	5	4	3	2	1	0
Name	GOE.7	GOE.6	GOE.5	GOE.4	GOE.3	GOE.2	GOE.1	GOE.0
Reset	0x18							
Type	R/W							

Bit	Name	Function
7:0	GOE.x	端口 1 的复用输出使能。

		0 关闭复用输出使能，输出由 PTx.DR 决定 1 打开复用输出使能，输出由 GDO 决定

4.3.16 PT1_DM0 (0xFF33)

Bit	7	6	5	4	3	2	1	0
Name	PT1_DM0							
Reset	0x18							
Type	R/W							

Bit	Name	Function
7:0	PT1_DM0	P1 模式控制寄存器。

4.3.17 PT1_DM1 (0xFF34)

Bit	7	6	5	4	3	2	1	0
Name	PT1_DM1							
Reset	0x18							
Type	R/W							

Bit	Name	Function
7:0	PT1_DM1	P1 模式控制寄存器。

4.3.18 PT1_DIE (0xFF35)

Bit	7	6	5	4	3	2	1	0
Name	PT1_DIE							
Reset	0x18							
Type	R/W							

Bit	Name	Function
7:0	PT1_DIE	数字端口使能寄存器 0 关闭数字输入，高阻模式下读端口得到 0 1 打开数字输入

4.3.19 PT1_AIE (0xFF36)

Bit	7	6	5	4	3	2	1	0
Name	PT1_AIE							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	PT1_AIE	模拟端口使能寄存器 0 关闭模拟通道 1 打开模拟通道

4.3.20 PT1_PU (0xFF37)

Bit	7	6	5	4	3	2	1	0

Name	PT1_PU
Reset	0x18
Type	R/W

Bit	Name	Function
7:0	PT1_PU	上拉使能： 0 关闭上拉 1 打开上拉

4.3.21 PT1_PD (0xFF38)

Bit	7	6	5	4	3	2	1	0
Name	PT1_PD							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	PT1_PD	下拉使能： 0 关闭下拉 1 打开下拉

4.3.22 PT1_IE (0xFF39)

Bit	7	6	5	4	3	2	1	0
Name	PT1_IE							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	PT1_IE	中断使能： 0 关闭端口中断 1 打开端口中断

4.3.23 PT1_IC0/PT1_IC1 (0xFF3A/0xFF3B)

Bit	7	6	5	4	3	2	1	0
Name	P1_IC0							
Reset	1	1	1	1	1	1	1	1
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Name	P1_IC1							
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function

7:0	P1_IC0	[P1_IC1：P1_IC0] 中断触发模式控制： 00 保留 01 下降沿中断 10 上升沿中断 11 上下沿中断 注意：IO 中断使能后需要读一下 IO 状态，作为后一状态对比； 如果使用的是上下沿中断，在中断函数中也需要读一下 IO 状态。
7:0	P1_IC1	

4.3.24 PT1_FLAG (0xFF3C)

Bit	7	6	5	4	3	2	1	0
Name	P1_FLAG							
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	P1_FLAG	中断标志： 0 没有中断发生 1 有中断发生 写任何值清除该中断标志

4.3.25 PT2_DR (0xFF40)

Bit	7	6	5	4	3	2	1	0
Name	PT2_DR							
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	PT2_DR	端口 2 的数据寄存器，写该寄存器会更新端口输出，读该寄存器得到端口输出值。 注意：如果端口配置为强推挽输出或者开漏输出，那么读端口得到端口寄存器的值； 如果端口配置为高阻模式，数字输入使能打开，那么读端口得到端口电平。

4.3.26 PT2_GIE (0xFF41)

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	GIE.1	GIE.0
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:2	N/A	保留位，读 0
1:0	GIE.x	端口 2 的复用输入使能。 0 关闭复用输入使能 1 打开复用输入使能

4.3.27 PT2_GOE (0xFF42)

Bit	7	6	5	4	3	2	1	0

Name	–	–	–	–	–	–	GOE.1	GOE.0
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:2	N/A	保留位，读 0
1:0	GOE.x	端口 2 的复用输出使能。 0 关闭复用输出使能，输出由 PTx.DR 决定 1 打开复用输出使能，输出由 GDO 决定

4.3.28 PT2_DM0 (0xFF43)

Bit	7	6	5	4	3	2	1	0
Name	PT2_DM0							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	PT2_DM0	P2 模式控制寄存器。

4.3.29 PT2_DM1 (0xFF44)

Bit	7	6	5	4	3	2	1	0
Name	PT2_DM1							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	PT2_DM1	P2 模式控制寄存器。

4.3.30 PT2_DIE (0xFF45)

Bit	7	6	5	4	3	2	1	0
Name	PT2_DIE							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	PT2_DIE	数字端口使能寄存器 0 关闭数字输入，高阻模式下读端口得到 0 1 打开数字输入

4.3.31 PT2_AIE (0xFF46)

Bit	7	6	5	4	3	2	1	0
Name	PT2_AIE							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	PT2_AIE	模拟端口使能寄存器 0 关闭模拟通道 1 打开模拟通道

4.3.32 PT2_PU (0xFF47)

Bit	7	6	5	4	3	2	1	0
Name	PT2_PU							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	PT2_PU	上拉使能： 0 关闭上拉 1 打开上拉

4.3.33 PT2_PD (0xFF48)

Bit	7	6	5	4	3	2	1	0
Name	PT2_PD							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	PT2_PD	下拉使能： 0 关闭下拉 1 打开下拉

4.3.34 PT2_IE (0xFF49)

Bit	7	6	5	4	3	2	1	0
Name	PT2_IE							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	PT2_IE	中断使能： 0 关闭端口中断 1 打开端口中断

4.3.35 PT2_IC0/PT2_IC1 (0xFF4A/0xFF4B)

Bit	7	6	5	4	3	2	1	0
Name	P2_IC0							
Reset	1	1	1	1	1	1	1	1
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0

Name	P2_IC1							
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	P2_IC0	[P2_IC1：P02_IC0] 中断触发模式控制： 00 保留 01 下降沿中断 10 上升沿中断 11 上下沿中断 注意：IO 中断使能后需要读一下 IO 状态，作为后一状态对比； 如果使用的是上下沿中断，在中断函数中也需要读一下 IO 状态。
7:0	P2_IC1	

4.3.36 4.3.36 PT2_FLAG (0xFF4C)

Bit	7	6	5	4	3	2	1	0
Name	P2_FLAG							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	P2_FLAG	中断标志: 0 没有中断发生 1 有中断发生 写 1 清除该中断标志

5 CPU

5.1 CPU 内核概述

本芯片全兼容传统的 8051 微控制器，所有指令的助记符和二进制码都和 8051 兼容。处理器采用了一些体系结构上的优化，相比传统的 8051 在性能上面有了很大的提升。

本芯片内部的 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

5.2 CPU 内核 SFR 寄存器概述

名字	地址	读写	复位值	描述
ACC	0xE0	读写	00000000	累加寄存器
B	0xF0	读写	00000000	B 寄存器
PSW	0xD0	读写	00000000	程序状态字寄存器
P2	0xA0	读写	00000000	P2 读写寄存器

5.2.1 ACC 寄存器 (0xE0)

Bit	7	6	5	4	3	2	1	0
Name	ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	ACC	累加寄存器。

5.2.2 B 寄存器 (0xF0)

Bit	7	6	5	4	3	2	1	0
Name	B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	B	乘法运算和除法运算的时候使用，其他情况用作普通寄存器。

5.2.3 PSW 寄存器 (0xD0)

Bit	7	6	5	4	3	2	1	0
Name	CY	AC	F0	RS[1:0]		OV	F1	P

Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	CY	进位标志
6	AC	辅助进位标志
5	F0	通用标志 0
4:3	RS[1:0]	寄存器组选择： 00　寄存器组 0，数据地址 0x00-0x07 01　寄存器组 1，数据地址 0x08-0x0F 10　寄存器组 2，数据地址 0x10-0x17 11　寄存器组 3，数据地址 0x18-0x1F
2	OV	溢出标志
1	F1	通用标志 1
0	P	奇偶校验标志

5.2.4　P2 寄存器 (0xA0)

Bit	7	6	5	4	3	2	1	0
Name	P2							
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	P2	使用 MOVX 指令使用 R0 或者 R1 的时候访问 XRAM 空间的时候标志地址的[15:8]位。

6 存储器

本芯片内部有 3 种存储器：SFR，内部数据存储器，程序存储器。

程序存储器只能读不能写，该存储器大小为 4K 字节。内部数据存储器大小为 256 字节。SFR 为内部特殊功能寄存器。

6.1 程序存储器

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

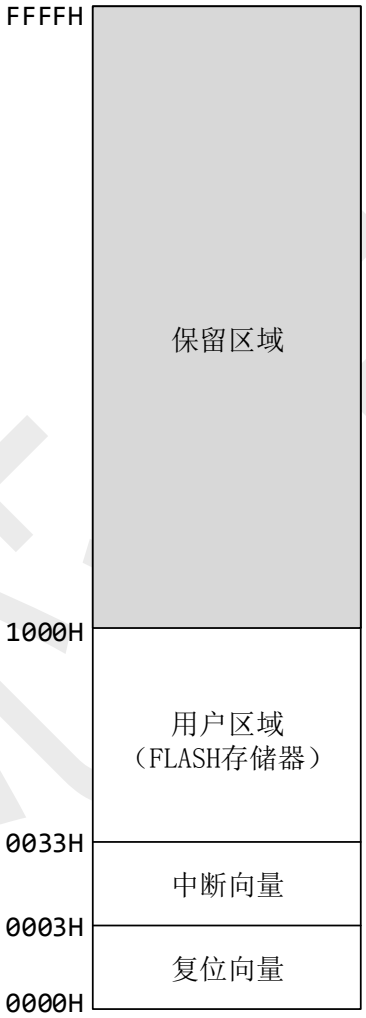


图 2-1 程序存储空间

复位后，MCU 从 0000H 开始执行。从 0003H 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。

6.2 数据存储器

数据存储器分内部数据存储器和外部数据存储器。内部数据存储器空间大小为 256 字节。内部数据存储器的地址空间的低 128 字节可以字节访问，高 128 字节和 SFR 共用一个地址空间，直接访问高 128 字节会访问到 SFR

空间，高 128 字节数据存储器只能通过间接寻址方式访问。

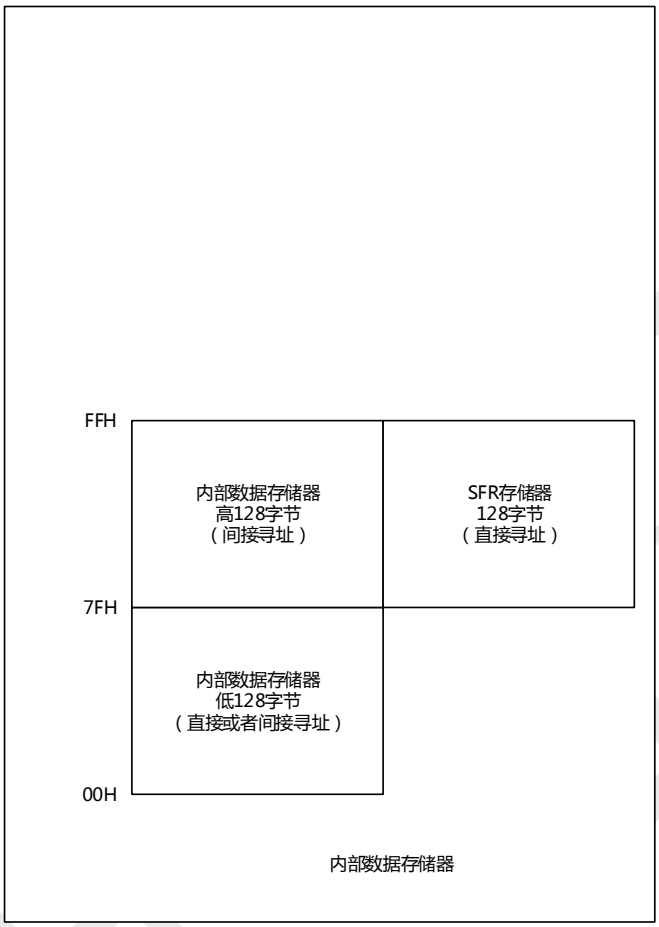


图 6-2 数据存储器

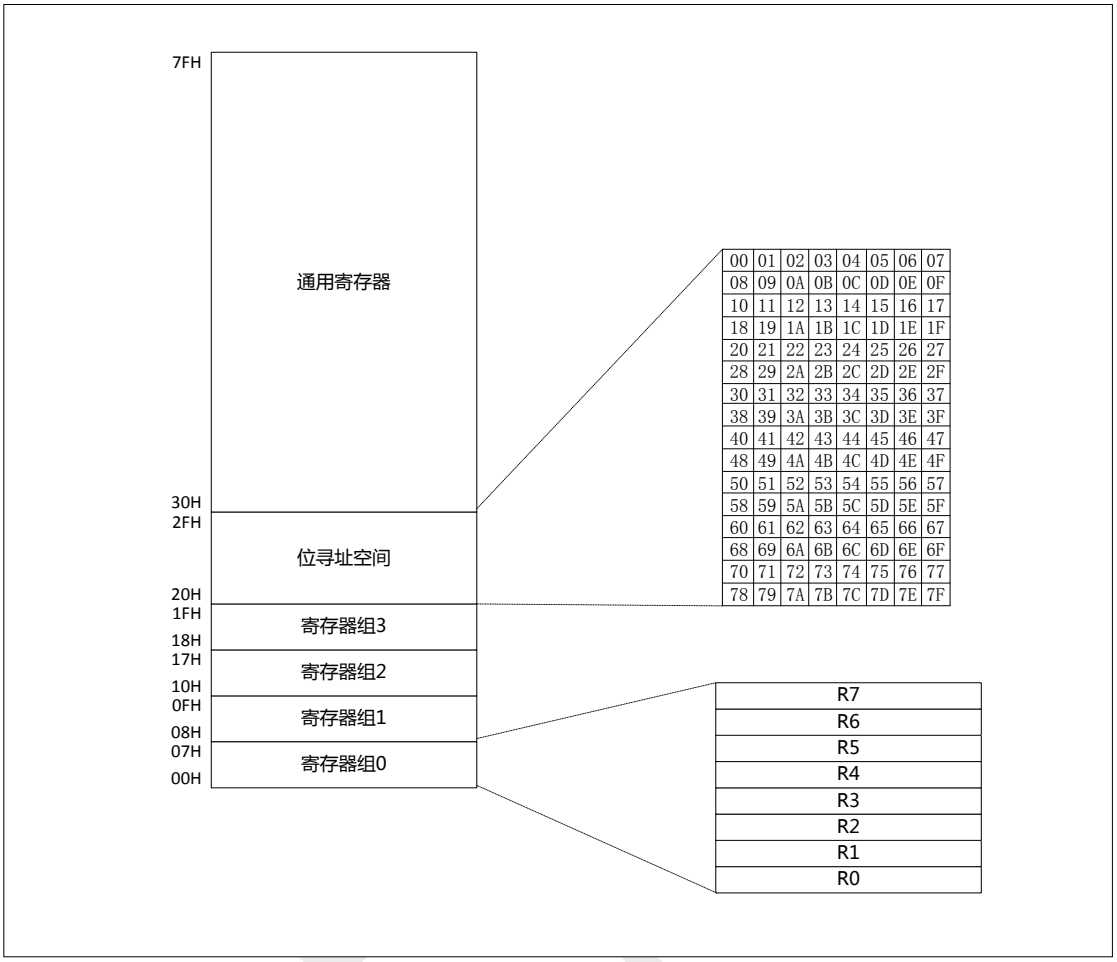


图 6-3 内部低 128 字节数据空间分配

6.3 SFR 空间

	0H/8H	1H/9H	2H/AH	3H/BH	4H/CH	5H/DH	6H/EH	7H/FH
F8H	TIMTER0_CR	TIM0_ARR	TIM0_IE	TIM0_SR	TIMER_SSCONR			
F0H	B							
E8H	ADC_CR0	ADC_CR1	ADC_CR2	ADC_RESL	ADC_RESH			
E0H	ACC				UART0CR	UART0DR	UART0SR	UART0CFG
D8H	AC0_CR1	AC0_CR2	AC0_DASEL	AC1_CR1	AC1_CR2	AC1_DASEL		
D0H	PSW							
C8H	TIM2_VPERR	TIM2_DTUA	TIM2_BRAKE	TIM2_DTR	TIM2_PCONRA	TIM2_PCONRB	TIM2_IE	TIM2_SR
C0H	TIM2_CR	TIM2_FCONR	TIM2_ARRL	TIM2_ARRH	TIM2_GCMARL	TIM2_GCMARH	TIM2_GCMBRL	TIM2_GCMBRH
B8H	TIM1_VPERR	TIM1_DTUA	TIM1_BRAKE	TIM1_DTR	TIM1_PCONRA	TIM1_PCONRB	TIM1_IE	TIM1_SR
B0H	TIM1_CR	TIM1_FCONR	TIM1_ARRL	TIM1_ARRH	TIM1_GCMARL	TIM1_GCMARH	TIM1_GCMBRL	TIM1_GCMBRH
A8H	IE				INT_MSK0	INT_MSK1	INT_FWSET0	INT_FWSET1
A0H	P2	I2C_ADDR	I2C_CR	I2C_STAT	I2C_DR	I2C_MCR		
98H								
90H	SCR_CFG	SCR_SLEEP	SCR_CALI		SCR_CLK_CR	SCR_PCLK_CR	SCR_PCLK_DIV12	SCR_PCLK_DIV3
88H	SLPTIM_CR	SLPTIM_SR	SLPTIM_CLR	SLPTIM_WDT	SLPTIM_CNTL	SLPTIM_CNTH	SLPTIM_PRDRL	SLPTIM_PRDRH
80H		SP	DPL0	DPH0	DPL1	DPH1	DPS	

6.4 XDATA 空间

芯片中一部分寄存器放在外部数据存储器空间，该部分地址空间大小 256 字节，地址范围 0xFF00~0xFFFF。下面表所示

	0H/8H	1H/9H	2H/AH	3H/BH	4H/CH	5H/DH	6H/EH	7H/FH
FFF8H								
FFF0H								
FFE8H								
FFE0H								
FFD8H								
FFD0H								
FFC8H								
FFC0H								
FFB8H								
FFB0H								
FFA8H								
FFA0H								
FF98H								
FF90H								
FF88H	IMO_CR	IMO_TRIM	ILO_TRIM	ILO_TEST	IMO_SLTRIM	XTAL_CR		
FF80H	BG_CR	BG_VTRIM	BG_ITRIM	BG_TCTRIM	BG_TEST	BORLVD_TEST	BORLVD_STAT	ANA_TEST

	0H/8H	1H/9H	2H/AH	3H/BH	4H/CH	5H/DH	6H/EH	7H/FH
FF78H								
FF70H								
FF68H								
FF60H								
FF58H								
FF50H								
FF48H	PT2_PD	PT2_IE	PT2_ICO	PT2_ICO	PT2_FLAG			
FF40H	PT2_DR	PT2_GIE	PT2_GOE	PT2_DMO	PT2_DM1	PT2_DIE	PT2_AIE	PT2_PU
FF38H	PT1_PD	PT1_IE	PT1_ICO	PT1_ICO	PT1_FLAG			
FF30H	PT1_DR	PT1_GIE	PT1_GOE	PT1_DMO	PT1_DM1	PT1_DIE	PT1_AIE	PT1_PU
FF28H	PT0_PD	PT0_IE	PT0_ICO	PT0_ICO	PT0_FLAG			
FF20H	PT0_DR	PT2_GIE	PT2_GOE	PT0_DMO	PT0_DM1	PT0_DIE	PT0_AIE	PT0_PU
FF18H								
FF10H								
FF08H								
FF00H	FLASH_CR	FLASH_CFG		FLASH_ADL	FLASH_ADH	FLASH_PBUFL	FLASH_PBUFH	FLASH_DR

6.5 FLASH 控制器

本芯片内部实现了一个大小为 4KB 的 FLASH 存储器，编程次数可达 1000 次。FLASH 控制器用来控制 8051 访问的 FLASH 存储器的读时序和编程器通过编程接口编程 FLASH 存储器。

6.5.1 与 FLASH 控制器相关寄存器定义

名字	地址	读写	复位值	描述
FLASH_CR	0xFF00	读写	00000000	FLASH control register
FLASH_CFG	0xFF01	读写	00110010	FLASH configuration register
FLASH_ADL	0xFF03	读写	00000000	FLASH program address low byte
FLASH_ADH	0xFF04	读写	00000000	FLASH program address high byte
FLASH_PBUFL	0xFF05	读写	00000000	FLASH pogram buffer low byte
FLASH_DR	0xFF07	读	xxxxxxxx	FLASH read data register

6.5.1.1 FLASH_CR (0xFF00)

Bit	7	6	5	4	3	2	1	0
Name	-	-	WRSZ[1:0]		CKEN	-	IFREN	BUSY
Reset	-	-	0	0	0	-	0	0
Type	-	-	R/W	R/W	R/W	-	R/W	R/W

Bit	Name	Function
7:6	N/A	保留位，读 0
5:4	WRSZ[1:0]	FLASH 存储器编程数据大小，当选择 EEPROM 区域时，单位为字节，FLASH 用户区域或信息区域单位为半字（2 个字节） 00 = 1 01 = 32 10 = 64 11 = 128
3	CKEN	FLASH 时钟使能: 0 = 关闭时钟 1 = 使能时钟
2	N/A	保留位，读 0
1	IFREN	0 = 选择 FLASH 用户区域 1 = 选择 FLASH 信息区域
0	BUSY	读模式下 BUSY 的值表示： 0 = FLASH 编程完成 1 = FLASH 编程没有完成 写 1 开始编程操作。

6.5.1.2 FLASH_CFG (0xFF01)

Bit	7	6	5	4	3	2	1	0
Name	FWSEL	CLEAN	ISAVB	-	SAVPWR1	SAVPWR0	RDCYC[1:0]	
Reset	0	0	0	-	0	0	1	1
Type	R/W	R/W	R/W	-	R/W	R/W	R/W	R/W

Bit	Name	Function
7	FWSEL	FLASH 控制信号选择: 0 使用默认的 FLASH CLEN, SRL, MRGN, ISAVB, STATICEN 信号 1 使用寄存器定义的 FLASH CLEN, SRL, MRGN, ISAVB, STATICEN 信号
6	CLEAN	FLASH 测试模式
5	ISAVB	FLASH 读模式 0 选择低功耗模式 1 选择高速模式
4	N/A	保留位，读 0
3	SAVPWR1	SLEEP 模式门控 READ 信号: 0 READ 信号打开 1 READ 信号关闭 注意：SLEEP 模式时建议此位置 1，休眠功耗较未置 1 会更小。
2	SAVPWR0	SLEEP 模式门控 CS 信号: 0 CS 信号打开 1 CS 信号关闭 注意：SLEEP 模式时建议此位置 1，休眠功耗较未置 1 会更小。
1	RDCYC[1:0]	FLASH 访问周期: 00 1 个周期 01 2 个周期 11 3 个周期 10 4 个周期 注意:当芯片电压低于 4.5V 时候，要配置 RDCYC 为 10（4 个周期）。芯片 FLASH 内部实现了 2 个字节的缓冲，VDD 电压大于 4.5V 时使用 01（2 个周期）配置即可，这样可以保证性能和功耗的平衡。

6.5.1.3 FLASH_ADL (0xFF03)

Bit	7	6	5	4	3	2	1	0
Name	ADL							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	ADL	FLASH 访问地址低 8 位。

6.5.1.4 FLASH_ADH (0xFF04)

Bit	7	6	5	4	3	2	1	0
Name	-	-	ADH					

Reset	-	-	0	0	0	0	0	0
Type	-	-	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:6	N/A	保留位，读 0
5:0	ADH	FLASH 访问地址高 6 位。

6.5.1.5 FLASH_PBUFL (0xFF05)

Bit	7	6	5	4	3	2	1	0
Name	PBUFL							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	PBUFL	FLASH 编程缓冲地址。

6.5.1.6 FLASH_DR (0xFF07)

Bit	7	6	5	4	3	2	1	0
Name	DR							
Reset	0xXX							
Type	R							

Bit	Name	Function
7:0	DR	FLASH 读数据。

7 中断控制器

7.1 概述

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

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

7.2 GPIO 中断

GPIO 中断来自引脚，可以根据寄存器配置来选择中断发生的条件。GPIO 中断可以通过 EIEDGE 来选择中断沿来触发中断。寄存器 PTx_IF 保存每个中断的中断标志。GPIO 中断必须在使能前读取对应的 PTx_DR，这样 GPIO 中断状态会更新，下次 GPIO 有电平变化的时候才能正确触发 GPIO 中断。

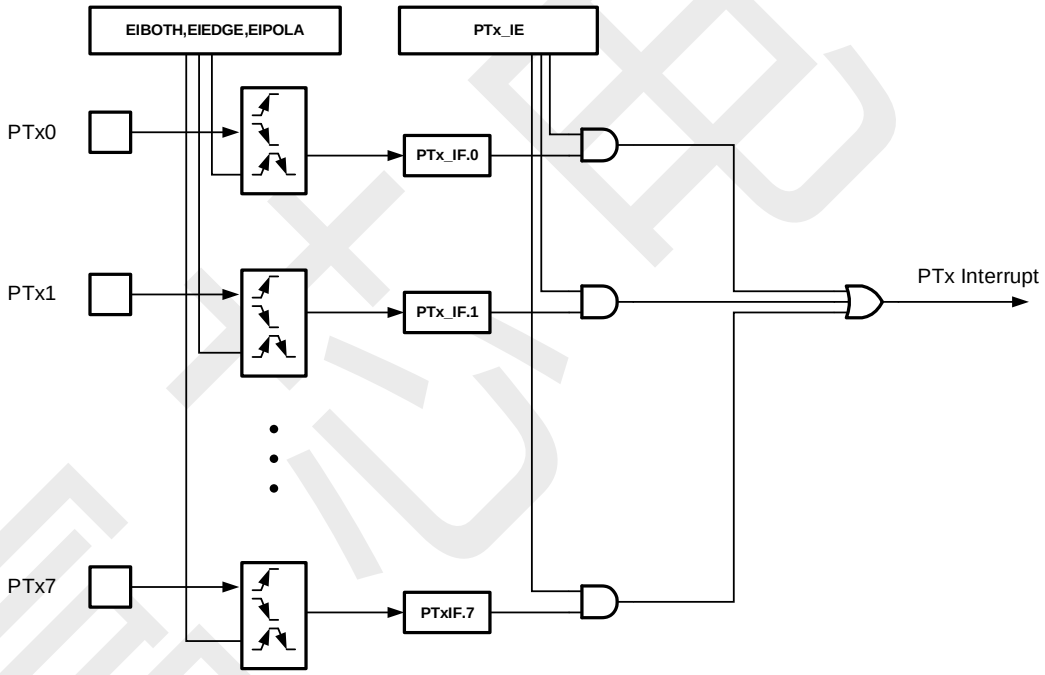


图 7-1 GPIO 中断源

7.3 中断结构框图

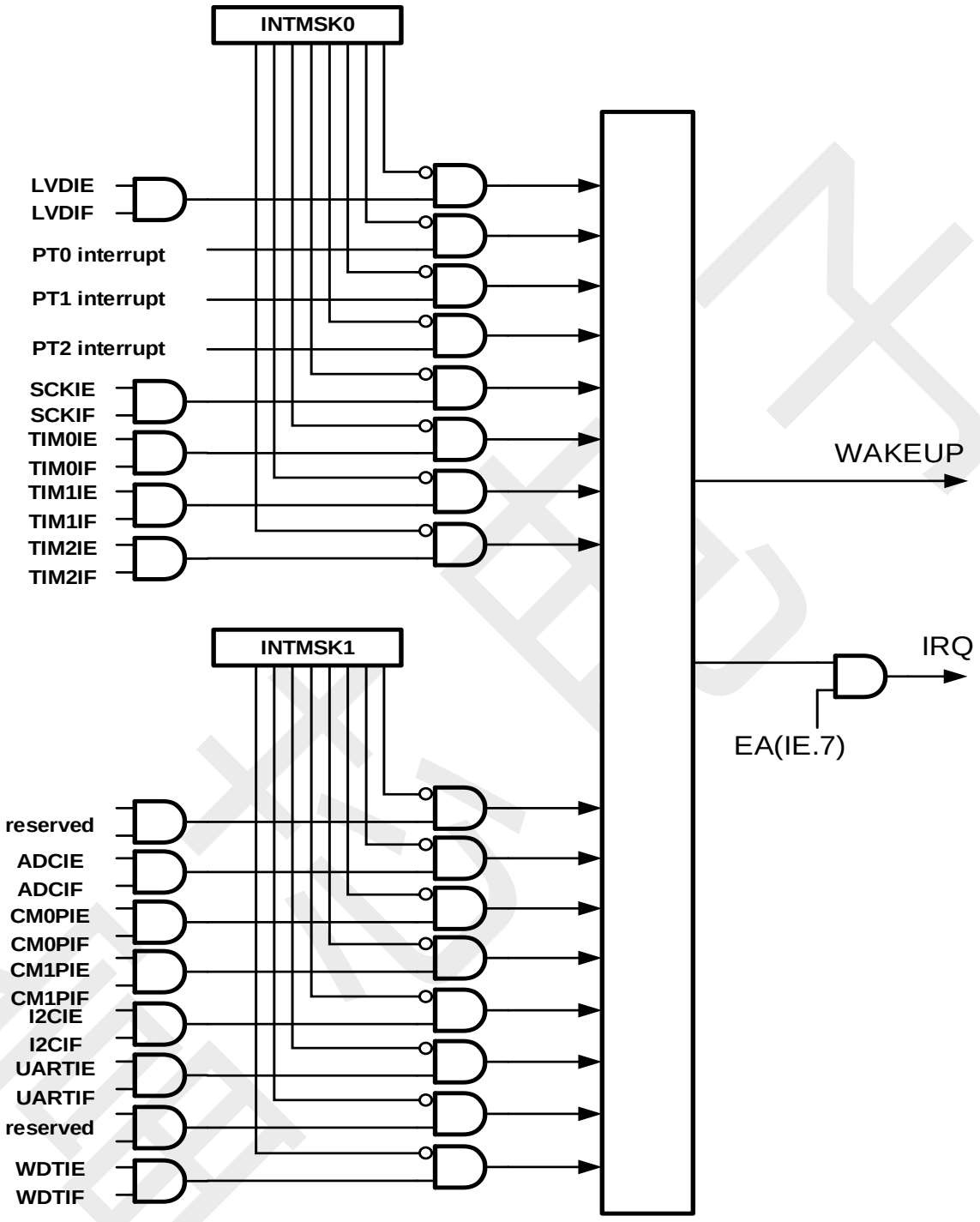


图 7-2 中断结构框图

7.4 中断向量表

中断控制器支持 14 个中断源如图 7-2 中断结构框图。当中断发生且中断使能之后，跳转到对应向量地址去执行 LCALL 指令来进入中断服务程序。

表 1 中断向量表

中断源	中断等级	中断号	中断地址	说明
LVD	低	0	0003H	低压检测
PT0	低	1	0006H	GPIO0 脚中断
PT1	低	2	0009H	GPIO1 脚中断
PT2	低	3	000CH	GPIO2 脚中断
SCK3	低	4	000FH	SCK3 时钟有效中断
Timer0	低	5	0012H	定时器 0 中断
Timer1	低	6	0015H	定时器 1 中断
Timer2	低	7	0018H	定时器 2 中断
Reserved	低	8	001BH	保留
ADC	低	9	001EH	ADC 转换完成中断
CMP0	低	10	0021H	模拟比较器 0 中断
CMP1	低	11	0024H	模拟比较器 1 中断
I2C	低	12	0027H	I2C 状态中断
UART	低	13	002AH	UART 状态中断
Reserved	低	14	002DH	保留
WDT	低	15	0030H	看门狗中断

7.5 中断优先级和中断屏蔽

每个中断有一个唯一的中断优先级编号。中断优先级编号越小，中断的优先级更高。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。

7.6 中断触发模式

如图所示，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。

7.7 与中断相关寄存器定义

名字	地址	读写	复位值	描述
INT_MSK0	0xAC	读写	00000000	中断屏蔽寄存器 0
INT_MSK1	0xAD	读写	00000000	中断屏蔽寄存器 1
INT_FWSET0	0xAE	读写	00000000	软件触发中断寄存器 0

INT_FWSET1	0xAF	读写	00000000	软件触发中断寄存器 1

7.7.1 INT_MSK0 (0xAC)

Bit	7	6	5	4	3	2	1	0
Name	T2MSK	T1MSK	TOMSK	SCK3MSK	PT2MSK	PT1MSK	PT0MSK	LVDMSK
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	T2MSK	0 = 不屏蔽 Timer2 中断 1 = 屏蔽 Timer2 中断
6	T1MSK	0 = 不屏蔽 Timer1 中断 1 = 屏蔽 Timer1 中断
5	TOMSK	0 = 不屏蔽 Timer0 中断 1 = 屏蔽 Timer0 中断
4	SC3KMSK	0 = 不屏蔽 SCK3 中断 1 = 屏蔽 SCK3 中断
3	PT2MSK	0 = 不屏蔽 GPIO 2 中断 1 = 屏蔽 GPIO 2 中断
2	PT1MSK	0 = 不屏蔽 GPIO 1 中断 1 = 屏蔽 GPIO 1 中断
1	PT0MSK	0 = 不屏蔽 GPIO 0 中断 1 = 屏蔽 GPIO 0 中断
0	LVDMSK	0 = 不屏蔽 LVD/SCM 中断 1 = 屏蔽 LVD/SCM 中断

7.7.2 INT_MSK1 (0xAD)

Bit	7	6	5	4	3	2	1	0
Name	WDTMSK	-	UARTMSK	I2CMSK	CMP1MSK	CMP0MSK	ADCMSK	-
Reset	0	-	0	0	0	0	0	-
Type	R/W	-	R/W	R/W	R/W	R/W	R/W	-

Bit	Name	Function
7	WDTMSK	0 = 不屏蔽 WDT 中断 1 = 屏蔽 WDT 中断
6	N/A	保留位，读 0
5	UARTMSK	0 = 不屏蔽 UART 中断 1 = 屏蔽 UART 中断
4	I2CMSK	0 = 不屏蔽 I2C 中断 1 = 屏蔽 I2C 中断
3	CMP1MSK	0 = 不屏蔽比较器 1 中断 1 = 屏蔽比较器 1 中断
2	CMP0MSK	0 = 不屏蔽比较器 0 中断 1 = 屏蔽比较器 0 中断

1	ADCMSK	0 = 不屏蔽 ADC 中断 1 = 屏蔽 ADC 中断
0	N/A	保留位，读 0

7.7.3 INT_FWSET0 (0xAE)

Bit	7	6	5	4	3	2	1	0
Name	T2TRG	T1TRG	T0TRG	SCK3TRG	PT2TRG	PT1TRG	PT0TRG	LVDTRG
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	T2TRG	0 = 不触发 Timer2 中断 1 = 触发 Timer2 中断
6	T1TRG	0 = 不触发 Timer1 中断 1 = 触发 Timer1 中断
5	T0TRG	0 = 不触发 Timer0 中断 1 = 触发 Timer0 中断
4	SCK3TRG	0 = 不触发 SCK3 中断 1 = 触发 SCK3 中断
3	PT2TRG	0 = 不触发 GPIO 2 中断 1 = 触发 GPIO 2 中断
2	PT1TRG	0 = 不触发 GPIO 1 中断 1 = 触发 GPIO 1 中断
1	PT0TRG	0 = 不触发 GPIO 0 中断 1 = 触发 GPIO 0 中断
0	LVDTRG	0 = 不触发 LVD 中断 1 = 触发 LVD 中断

7.7.4 INT_FWSET1 (0xAF)

Bit	7	6	5	4	3	2	1	0
Name	WDTTRG	-	UARTTRG	I2CTRG	CMP1TRG	CMP0TRG	ADCTRG	-
Reset	0	-	0	0	0	0	0	-
Type	R/W	-	R/W	R/W	R/W	R/W	R/W	-

Bit	Name	Function
7	WDTTRG	0 不触发 WDT 中断 1 触发 WDT 中断
6	N/A	保留位，读 0
5	UARTTRG	0 不触发 UART 中断 1 触发 UART 中断
4	I2CTRG	0 不触发 I2C 中断 1 触发 I2C 中断
3	CMP1TRG	0 不触发比较器 1 中断 1 触发比较器 1 中断

2	CMPOTRG	0 不触发比较器 0 中断 1 触发比较器 0 中断
1	ADCTRG	0 不触发 ADC 中断 1 触发 ADC 中断
0	N/A	保留位，读 0

8 时钟

8.1 概述

系统有三个时钟源，来自内部的 16MHz 高速 RC 振荡器和 32KHz 低速 RC 振荡器和来自外部的 32.768KHz 低速晶体振荡器。

8.2 结构框图

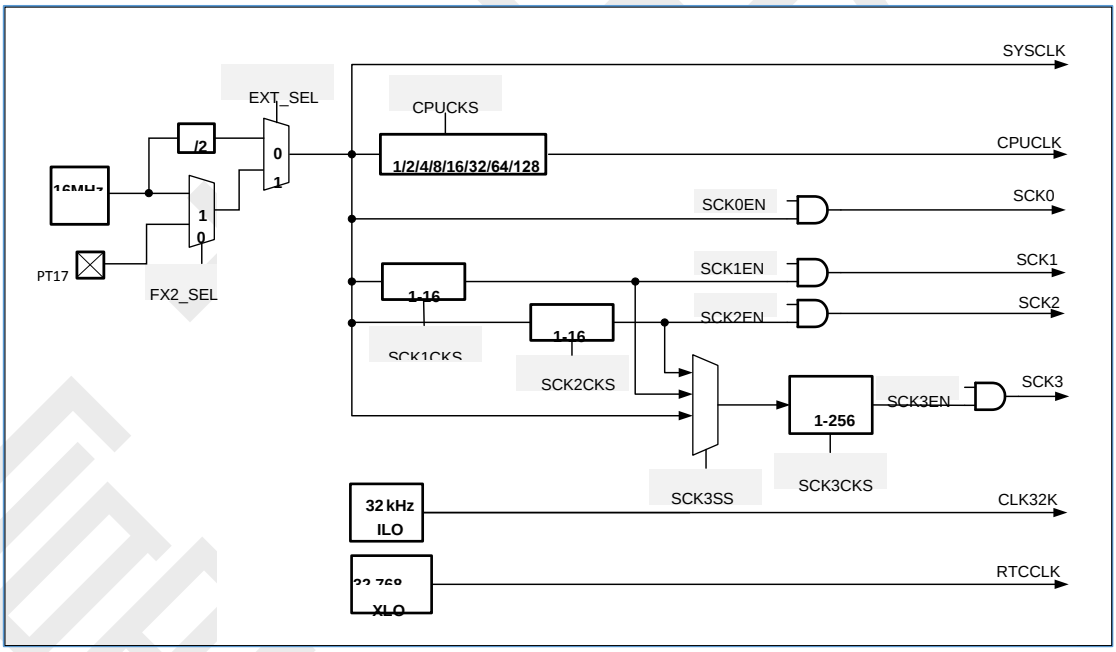


图 8-1 时钟结构框图

8.3 CPU 时钟

CPU 时钟源来自系统时钟 SYSCLK，分频比可以通过寄存器配置为 1、2、4、8、16、32、64、128。CPUCLK 时钟提供 8051 内核工作时钟。

8.4 SCK1 和 SCK2 时钟

SCK1 可以对 SYSCLK 做 1 到 16 分频，SCK2 可以对 SYSCLK 做 1 到 16 分频，每个都带使能控制。

8.5 SCK3 时钟

SCK3 有 3 个时钟源，分别可以来自 SYSCLK，SCK1，SCK2。SCK3 带一个使能控制，通过使能位可以控制 SCK3 时钟的开关。SCK3 时钟自带一个中断，可以单独使能，每次当 SCK3 的上升沿到来的时候产生一次中断，用户可用该中断来做定时器。

8.6 32K 时钟

如果 SYSCLK 域使用了，32K 时钟（来自 ILO）会同步到 SYSCLK。

8.7 32.768KHz 时钟

32.768KHz 时钟要配置外部振荡器使用，模块的控制参考 12.2.8 相关描述。该时钟源不能选择作 SYSCLK。

8.8 与时钟相关寄存器定义

名字	地址	读写	复位值	描述
IMO_CR	0xFF88	读写	00000000	IMO 控制寄存器
SCR_CLK_CR	0x94	读写	00000011	系统时钟控制寄存器
SCR_PCLK_CR	0x95	读写	11110001	外设时钟控制寄存器
SCR_PCLK_DIV12	0x96	读写	00001111	SCK1、SCK2 时钟控制寄存器
SCR_PCLK_DIV3	0x97	读写	00110001	SCK3 时钟控制寄存器

8.8.1 IMOCR (0xFF88)

Bit	7	6	5	4	3	2	1	0
Name	EXT_SEL	FX2_SEL	-	-	IMO_TSTEN	-	SLOW	IMO_EN
Reset	0	0	-	-	0	-	0	0
Type	R/W	R/W	-	-	R/W	-	R/W	R/W

Bit	Name	Function
7:6	EXT_SEL：FX2_SEL	系统时钟源选择： 0x 选择内部 8MHz 时钟 10 选择 PT17 做系统时钟 11 选择内部 16MHz 时钟
5:4	N/A	保留位，读 0
3	IMO_TSTEN	0 IMO 测试功能关闭

		1 IMO 测试功能打开，选择 SCK1 到 PT10 口。 注意要把 PT10 的 GPIO 复用输出功能打开。
2	N/A	保留位，读 0
1	SLOW	0 选择快速时钟修调值 1 选择慢速时钟修调值
0	IMO_EN	写模式下： 0 使能 IMO 1 关闭 IMO 读模式下： 0 IMO 关闭 1 IMO 使能

8.8.2 SCR_CLK_CR (0x94)

Bit	7	6	5	4	3	2	1	0
Name	SCK3IF	–	–	–	–	CPUCKS[2:0]		
Reset	1	–	–	–	–	0	1	1
Type	R/W	–	–	–	–	R/W	R/W	R/W

Bit	Name	Function
7	SCK3IF	0 = 没有 SCK3 中断发生 1 = 有 SCK3 中断发生 对该位写 1 会将其清零 注意：SCK3IF 复位值为 0，而 SCK3 默认情况下是有效的，而且会在软件启动之前就起振，因此软件看到的复位值为 0x83。
6:3	N/A	保留位，读 0
2:0	CPUCKS[2:0]	内核工作频率选择: 000 SYSCLK/8 001 SYSCLK/4 010 SYSCLK/2 011 SYSCLK 100 SYSCLK/16 101 SYSCLK/32 110 SYSCLK/64 111 SYSCLK/128

8.8.3 SCR_PCLK_CR (0x95)

Bit	7	6	5	4	3	2	1	0
Name	SCK0EN	SCK1EN	SCK2EN	SCK3EN	SCK3_IE	–	SCK3SS[1:0]	
Reset	1	1	1	1	0	–	0	1
Type	R/W	R/W	R/W	R/W	R/W	–	R/W	R/W

Bit	Name	Function
7	SCK0EN	0 = 禁止 SCK0 时钟 1 = 使能 SCK0 时钟

6	SCK1EN	0 = 禁止 SCK1 时钟 1 = 使能 SCK1 时钟
5	SCK2EN	0 = 禁止 SCK2 时钟 1 = 使能 SCK2 时钟
4	SCK3EN	0 = 禁止 SCK3 时钟 1 = 使能 SCK3 时钟
3	SCK3_IE	0 = 禁止 SCK3 时钟中断 1 = 使能 SCK3 时钟中断
2	N/A	保留位，读 0
1:0	SCK3SS[1:0]	SCK3 时钟源选择： 00 = 关闭 SCK3 时钟 01 = 来自 SYSCLK 10 = 来自 SCK1 时钟 11 = 来自 SCK2 时钟

8.8.4 SCR_PCLK_DIV12 (0x96)

Bit	7	6	5	4	3	2	1	0
Name	SCK1CKS				SCK2CKS			
Reset	0x0F							
Type	R/W							

Bit	Name	Function
7:4	SCK1CKS	控制 SCK1 时钟分频 $f_{SCK1}=f_{SYSCLK}/(SCK1CKS+1)$
3:0	SCK2CKS	控制 SCK2 时钟分频 $f_{SCK2}=f_{SYSCLK}/(SCK2CKS+1)$

8.8.5 SCR_PCLK_DIV3 (0x97)

Bit	7	6	5	4	3	2	1	0
Name	SCK3CKS							
Reset	0x31							
Type	R/W							

Bit	Name	Function
7:0	SCK3CKS	控制 SCK3 时钟的分频，频率和 SCK3SS 的值相关，具体计算方法如下： 当 SCK3SS 等于： 00 关闭 SCK3 时钟 01 $f_{SCK3}=f_{SYSCLK}/(SCK3CKS+1)$ 10 $f_{SCK3}=f_{SYSCLK}/(SCK3CKS+1)/(SCK1CKS+1)$ 11 $f_{SCK3}=f_{SYSCLK}/(SCK3CKS+1)/(SCK2CKS+1)$

9 复位

复位源有 4 个，POR 复位，BOR 复位，引脚复位，看门狗复位。

9.1 外部 RST 复位

PT00 脚可以用作外部复位引脚，外部给 PT00 一定宽度的复位脉冲信号，高电平有效(高电平时间需大于 100us)。复位状态下 PT00 默认用作普通 GPIO，通过软件配置寄存器 PT00RST 可以使 PT00 用作引脚复位。注意：在强干扰环境中，建议使用外加抗干扰复位电路。另外 P00 上电默认状态是开漏上拉，当该 PIN 脚作为外部 RST 脚时，如果在外部电路中对 GND 接有电容时，在上电时 P00 的默认高电平会对电容充电，当程序中使能复位功能后，电容维持的电平可能会触发复位，所以在程序中需要注意的是上电后需要把 P00 的上拉关闭。

9.2 看门狗复位

参考 11.2。

9.3 欠压复位

芯片内建欠压复位（BOR）模块，如果检测到了电源电压低于欠压复位所设定的点会触发欠压复位。欠压复位模块复位后默认使能，只要发生上电复位该模块都会处于使能状态。欠压电压 4 档可调。欠压复位的寄存器描述见 12.2.2。

10 外设

10.1 8-bit 基本计数器

10.1.1 概述

8 位基本定时器内部包含一个 8 位自动重装向上计数器，带预分频。可以用作基本的间隔定时器中断，计时溢出可以产生中断。主要特性如下：

- 8-bit 自动重装向上计数器
- 3-bit 可编程预分频，分频比 1, 2, 4, 8, 16, 32, 64, 128
- 计数器溢出产生中断同时重装计数器
- 计数时钟可选系统时钟，32K 看门狗时钟，RTC 时钟，其中 RTC 时钟支持休眠唤醒

10.1.2 结构框图

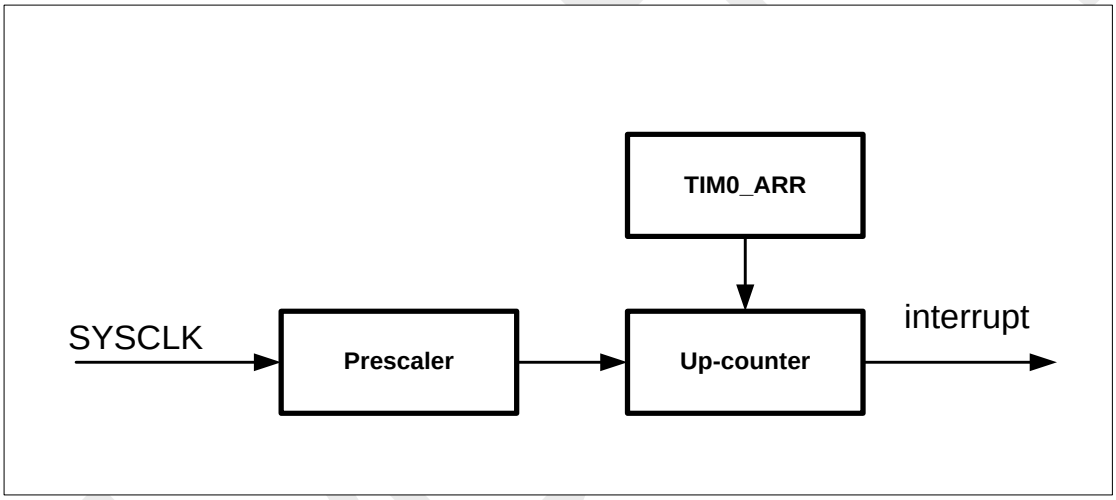


图 10-1 TIMER0 结构框图

10.1.3 与 TIM0 相关寄存器定义

名字	地址	读写	复位值	描述
TIM0_CR	0xF8	读写	00000000	Timer0 控制寄存器
TIM0_ARR	0xF9	读写	00000000	Timer0 自动重装寄存器
TIM0_IE	0xFA	读写	00000000	Timer0 中断控制寄存器
TIM0_SR	0xFB	读写	00000000	Timer0 状态寄存器
SSCONR	0xFC	读写	00000000	Timer1/2 软件同步控制寄存器

10.1.3.1 TIM0_CR (0xF8)

Bit	7	6	5	4	3	2	1	0

Name	-	-	TIM0_CLKSEL[1:0]		TIM0_CLKDIV[2:0]			TIM0_EN
Reset	-	-	0	0	0	0	0	0
Type	-	-	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:6	N/A	保留位，读 0
5:4	TIM0_CLKSEL[1:0]	TIMER0 时钟选择： 00 CLK_SYS 01 内部 32K 时钟 10 外部 RTC 时钟 11 保留
3:1	TIM0_CLKDIV[2:0]	TIMER0 预分频选择: 000 1 分频 001 2 分频 010 4 分频 011 8 分频 100 16 分频 101 32 分频 110 64 分频 111 128 分频
0	TIM0_EN	0 = TIMER0 关 1 = TIMER0 开

10.1.3.2 TIM0_ARR (0xF9)

Bit	7	6	5	4	3	2	1	0
Name	TIM0_ARR							
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	TIM0_ARR	TIMER0 自动重装寄存器。

10.1.3.3 TIM0_IE (0xFA)

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	TIM0_TCIE
Reset	-	-	-	-	-	-	-	0
Type	-	-	-	-	-	-	-	R/W

Bit	Name	Function
7:1	N/A	保留位，读 0
0	TIM0_TCIE	0 = 溢出中断关 1 = 溢出中断开

10.1.3.4 TIM0_SR (0xFB)

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	TIM0_TC
Reset	-	-	-	-	-	-	-	0
Type	-	-	-	-	-	-	-	R/W

Bit	Name	Function
7:1	N/A	保留位，读 0
0	TIM0_TC	定时器 0 溢出标志位: 0 TIMER0 未发生溢出 1 TIMER0 发生溢出 写 1 清零该标志位。

10.1.3.5 SSCONR (0xFC)

Bit	7	6	5	4	3	2	1	0
Name	-	SSCLR2	SSTPR2	SSTAR2	-	SSCLR1	SSTPR1	SSTAR1
Reset	-	0	0	0	-	0	0	0
Type	-	R/W	R/W	R/W	-	R/W	R/W	R/W

Bit	Name	Function
7	N/A	保留位，读 0
6	SSCLR2	写 1 TIMER2 停止计数 写 0 无效 读出值为 0
5	SSTPR2	写 1 TIMER2 暂停计数，当前计数值保持 写 0 无效 读出值为 0
4	SSTAR2	写 1 TIMER2 开始计数 写 0 无效 读出的值为 TIMER2 的使能状态
3	N/A	保留位，读 0
2	SSCLR1	写 1 TIMER1 停止计数 写 0 无效 读出值为 0
1	SSTPR1	写 1 TIMER1 暂停计数，当前计数值保持 写 0 无效 读出值为 0
0	SSTAR1	写 1 TIMER1 开始计数 写 0 无效 读出的值为 TIMER1 的使能状态

10.2 16-bit 高级计数器

10.2.1 概述

高级定时器是一个包含两个定时器 TIMER1/2。TIMER1/2 是功能相同的高级计数器，可用于产生不同形式的时钟波形，一个定时器可以产生同频的一组互补 PWM 或者 2 路 PWM 独立输出。可以捕获外界输入进行脉冲宽度或周期测量。

10.2.2 主要特性

主要特性如下：

- 内置 16 位计数器，向上或者向下计数，自动重装
- 时钟源可选择来自系统时钟或者看门狗时钟或者外部 RTC 时钟
- 预分频 1-16
- 后分频 1, 2, 4, 8, 16, 32, 64, 128(周期间隔响应)
- 和外部信号同步（外部时钟，复位）
- 输入捕捉（上升沿，下降沿和双沿）和比较功能
- 对输入沿计数，可选上升沿，下降沿和双沿
- 刹车输入，可以将 TIMER1/2 的输出置位复位状态或者特定的状态
- 支持 PWM 输出功能
  - 可输出 2 路独立 PWM 或者 1 路互补 PWM，互补输出可编程死区
  - 支持刹车功能，刹车输入为比较器输出
  - 影子寄存器，计数周期只有按顺序写入才能更新
- 中断，在以下事件产生中断：
  - 计数器上溢或下溢
  - 输入捕捉
  - 输出比较
  - 刹车产生

10.2.3 结构框图

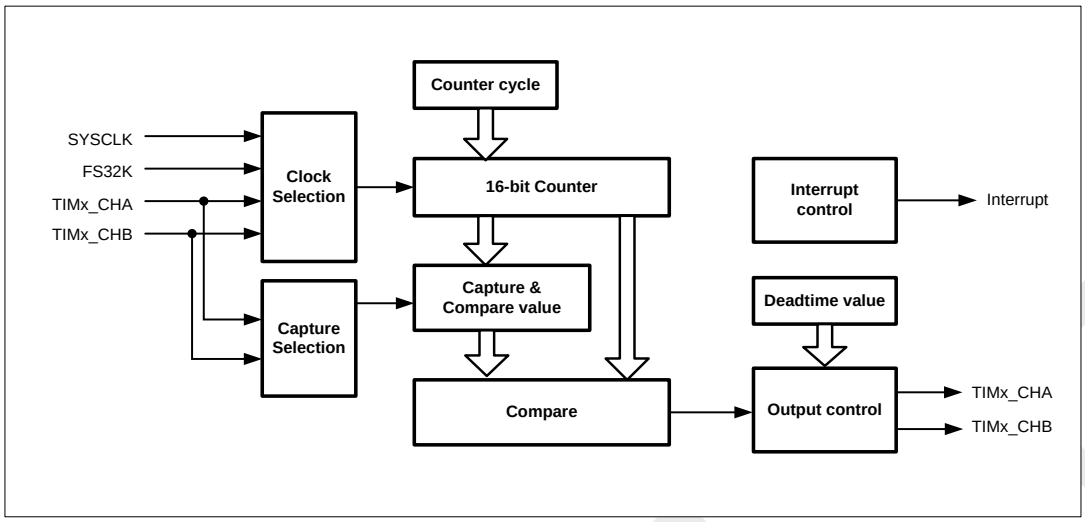


图 10-2

10.2.4 基本动作

基本波形模式

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

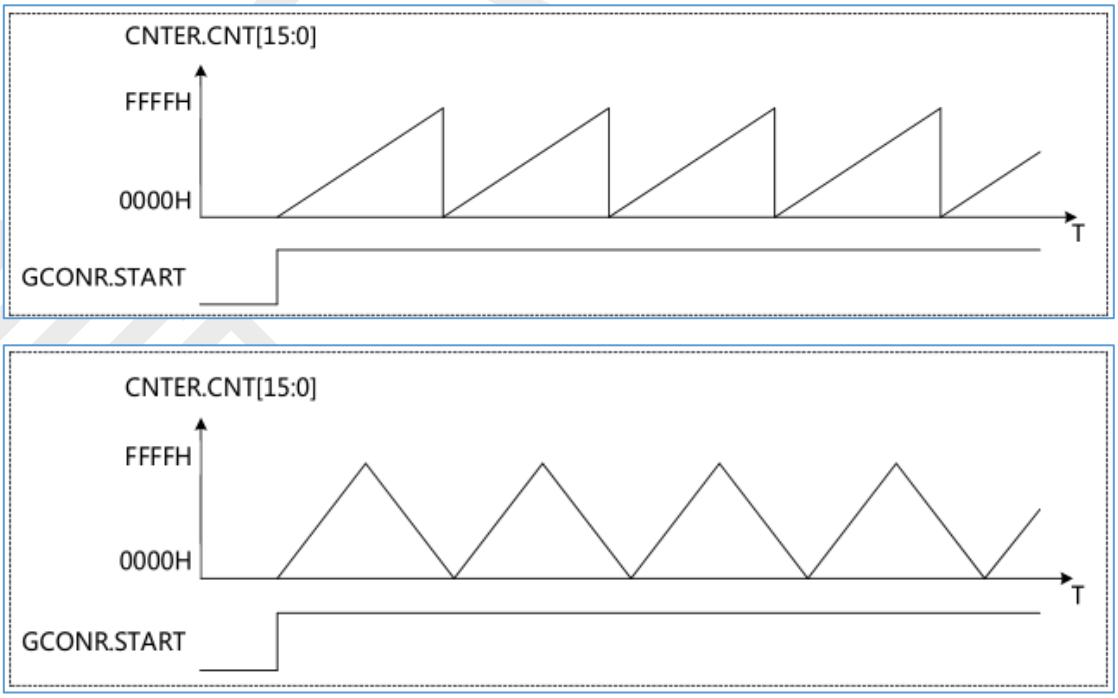


图 10-3

比较输出

TIMER1/2 一个定时器有 2 个比较输出端口（TIMx_CHA、TIMx_CHB），可在计数值与计数基准值比较匹配时输出指定的电平。GCMAR、GCMBR 寄存器分别对应了 TIMx_CHA、TIMx_CHB 的计数比较基准值。当计数器的计数值和 GCMAR 相等时，TIMx_CHA 端口输出指定的电平；当计数器的计数值和 GCMBR 相等时，TIMx_CHB 端口输出指定电平。

TIMx_CHA、TIMx_CHB 端口的计数起始电平和计数比较匹配时的电平由 TIM1_PCONRA.PA_INITVAL 和 TIM1_PCONRA.CAPA_OUT 定义。图为比较输出的动作例。

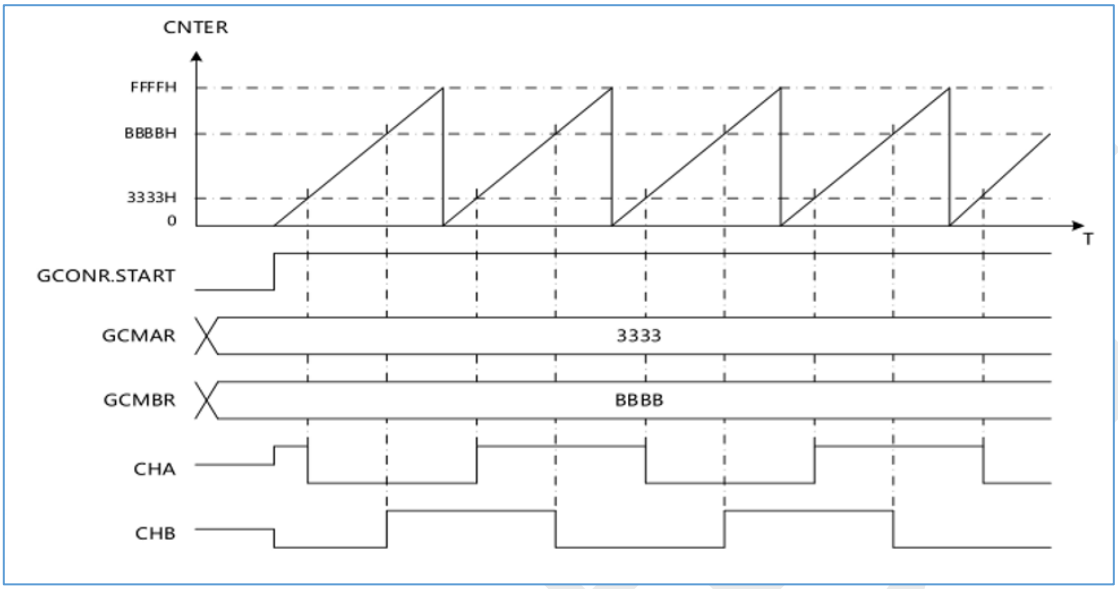


图 10-4

TIMER1/2 都具有捕获输入功能，具备 2 组捕获输入寄存器（GCMAR_S、GCMBR_S），用于保存捕获到的计数值。设定端口控制寄存器（PCONRA/ PCONRB）的 capa_en/capb_en 位为 1，对应端口的捕获输入功能就有效了。当设定了对应的捕获输入条件且该条件有效时，当前的计数值就被保存到相应的寄存器（GCMAR_S、GCMBR_S）中。每组捕获输入的条件可选 TIMx_CHA 或 TIMx_CHB 的上升沿，下降沿或上升下降沿，通过 CAPA_MODE/CAPB_MODE 来设定对应端口的捕获条件。图为捕获输入的动作例。

捕获输入

捕获是根据外部信号的沿采样内部计数器的值，TIM1_ARR_L 和 TIM1_ARR_H 这两个寄存器决定了定时器内部计数器的溢出时间，捕获模式要设置，建议两个寄存器都设置成 0xFF，捕获模式推荐使用三角波模式 A，三角波模式 A 的捕获图参考下图。

捕获模式读取这两个寄存器的值要把 TIMx_CR 的 SEL_SREG 设置成 0 才能读到真的捕获值，否则读取的是配置寄存器时写入的 GCMAR 和 GCMBR 值。SEL_SREG 只影响这两个寄存器的读，捕获模式下写这两个寄存器没有意义。

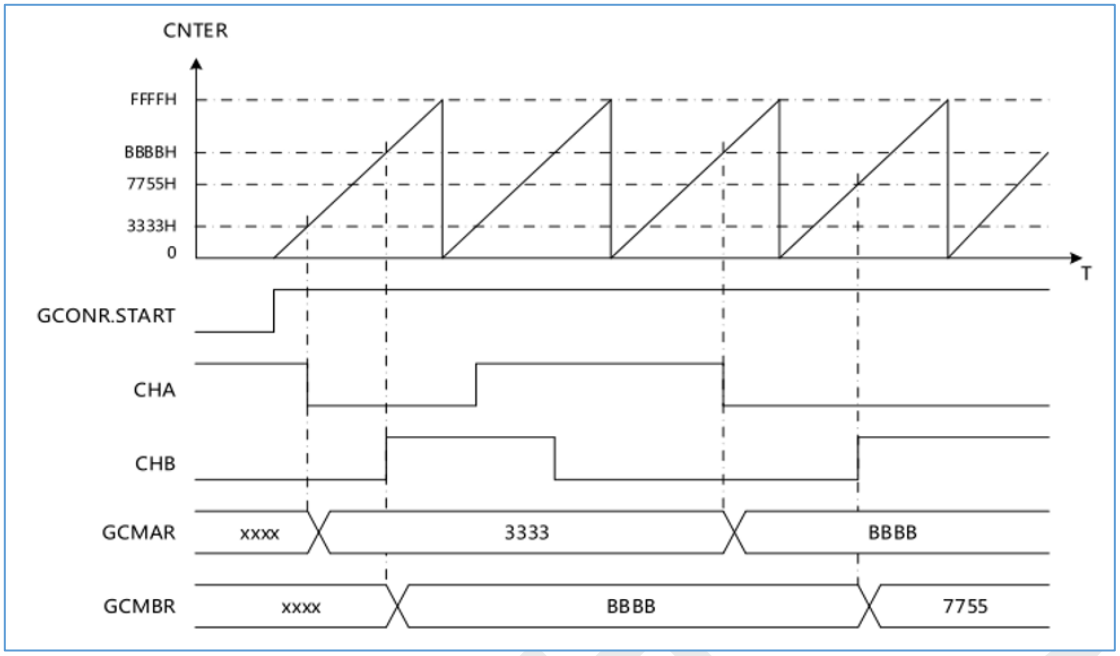


图 10-5

10.2.5 时钟源选择

Timer1/2 的计数时钟可以有以下几种选择：

- 系统时钟（SYSCLK）
- 内部低速 RC 振荡器 32kHz 时钟
- 外部低速晶体振荡器 32.768KHz 时钟

时钟分频 1-16 可选。

可选输入 CHA/CHB 作为时钟，此时可对 CHA/CHB 沿（上升沿，下降沿，上升下降沿可选）计数。

10.2.6 计数方向

TIMER1/2 的计数器计数方向可通过软件方式改变。不同波形模式时，改变计数方向的方法略有不同。

10.2.6.1 锯齿波计数方向

锯齿波模式时，计数方向可在计数器计数中或停止时设定。

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

在计数停止时，设定 GCONR.DIR 位。则计数开始后直至上溢或下溢时，GCONR.DIR 的设定才会反映到计数中。

10.2.6.2 三角波计数方向

三角波模式时，计数方向只能在计数器停止时设定。在计数中设定计数方向无效。在计数停止时，设定 CR.DIR 位。则计数开始后直至上溢或下溢时，CR.DIR 的设定才会反映到计数中。

10.2.7 数字滤波

TIMER1/2 的 TIMX_CHA、TIMX_CHB 端口输入都有数字滤波功能。可通过设定 PA_FILTER_EN/PB_FILTER_EN 开启对应端口的滤波功能。滤波时钟为计数器当前工作时钟。

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

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

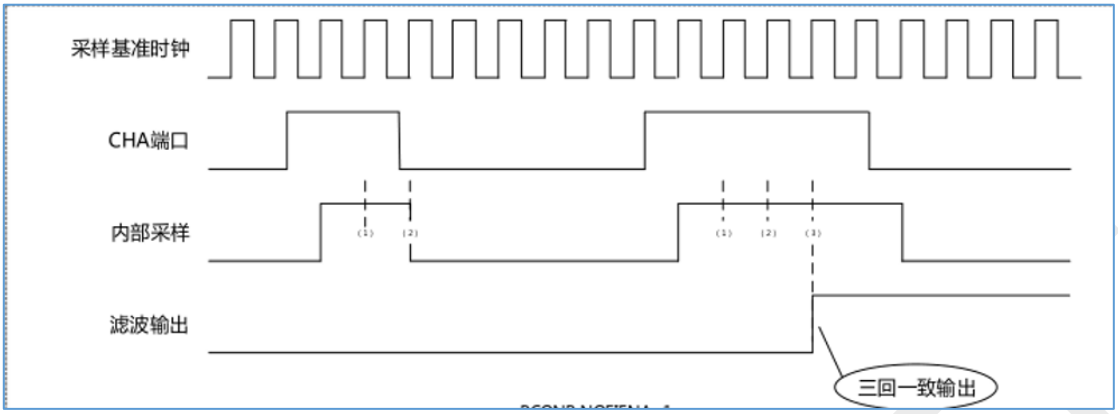


图 10-6

10.2.8 软件同步

TIMER1/2 可通过设定软件同步启动寄存器（SSTAR），实现目标 TIMER1/2 的同步启动。

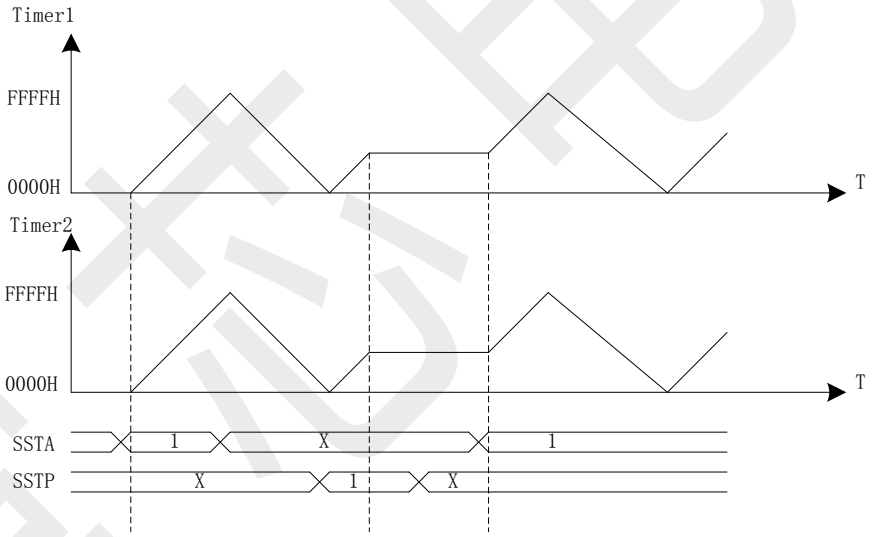


图 10-7

10.2.8.1 软件同步停止

TIMER1/2 可通过设定软件同步停止寄存器（SSTPR），实现目标 TIMER1/2 的同步停止，此时计数器处于暂停状态，对同步启动寄存器（SSTAR）写 1 可以继续计数。

10.2.8.2 软件同步清零

TIMER1/2 可通过设定软件同步清零寄存器（SCLRR），实现目标 TIMER1/2 的同步清零，此时计数器会复位到初始状态。

如图 13-7 所示、若设定 SSTAR，即可实现 TIMER1/2 的软件同步启动。

软件同步动作相关寄存器（SSTAR、SSTPR、SCLRR）是一组独立于 TIMER1/2 外、各个 TIM 间共用的寄存器，这组寄存器的各个位只在写 1 时有效，写 0 无效。在读取 SSTAR 寄存器时，会读出定时器的计数器状态（TIMER1/2 同时开始读到 1，其余情况下为 0），在读取 SSTPR 或 SCLRR 时，会读出 0。

10.2.9 缓存功能

缓存动作是指在缓存传送时间点，发生以下事件：

a. 通用周期基准值缓存寄存器（PERBR）的值自动传送到通用周期基准值寄存器（PERAR_S）中；

b. 通用比较基准值缓存寄存器（GCMAR、GCMBR）的值自动传送到通用比较基准值寄存器（GCMAR_S、GCMBR_S）中（比较输出时）；

c. 通用比较基准值寄存器（GCMAR、GCMBR）的值自动传送到通用比较基准值缓存寄存器（GCMAR_S、GCMBR_S）中（捕获输入时）；

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

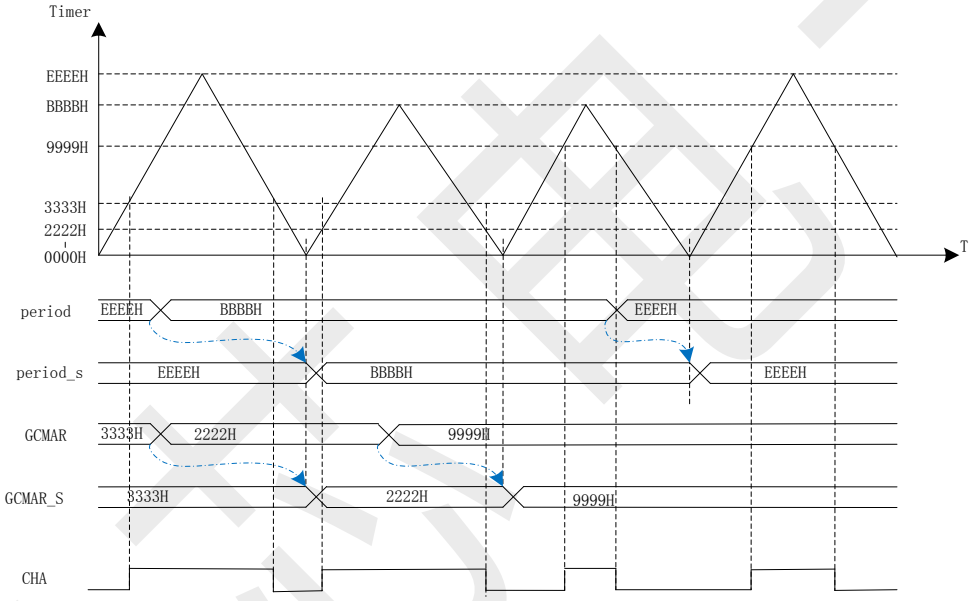


图 10-8

10.2.9.1 缓存传送时间点

周期基准值缓存传送时间点为锯齿波时递加计数上溢点或递减计数下溢点、三角波时计数谷点。

锯齿波 A 模式时，缓存传送发生在上溢点或下溢点。

三角波 A 模式时，缓存传送发生在计数谷点。

三角波 B 模式时，缓存传送发生在计数谷点和计数峰点。

捕获输入动作缓存传送时间点为捕获输入动作时。

在锯齿波计数模式或硬件计数模式时，正常的比较输出动作期间若有清零动作产生，通用周期基准值、通用比较基准值、等会根据相应的缓存动作设定状况发生一次缓存传送。

10.2.10 通用 PWM 输出

10.2.10.1 独立 PWM 输出

每个定时器的 2 个端口 TIMx_CHA、TIMx_CHB 能独立的输出 PWM 波。如图所示，定时器 Timer1 的 CHA 端口输出 PWM 波。

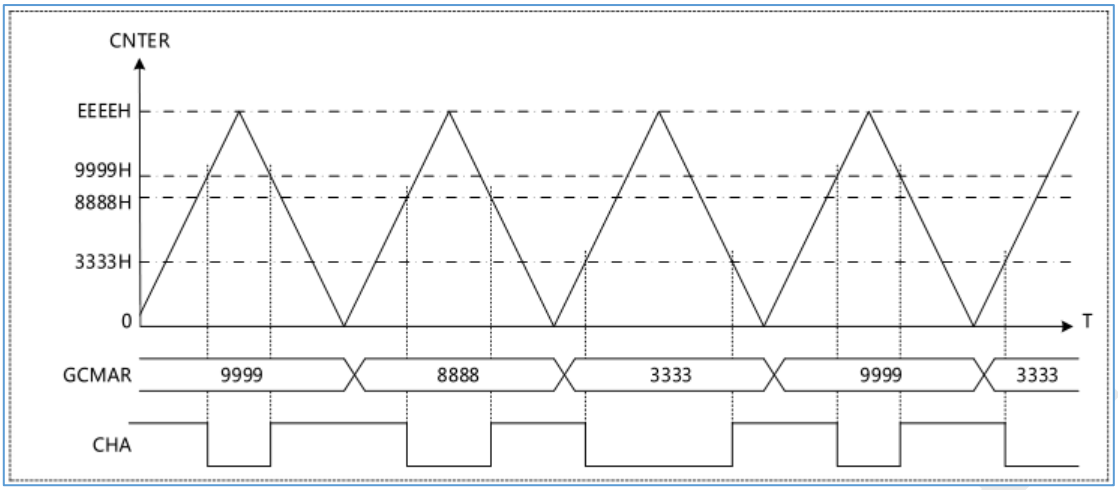


图 10-9

10.2.10.2 互补 PWM 输出

TIMx_CHA 端口和 TIMx_CHB 端口，在不同的模式下可组合输出互补 PWM 波形。

软件设定 GCMBR 互补 PWM 输出

软件设定 GCMBR 互补 PWM 输出是指在锯齿波模式和三角波 A 模式、三角波 B 模式下，用于 TIMx_CHB 端口波形输出的通用比较基准值寄存器（GCMBR）的值由寄存器直接设定，与通用比较基准值寄存器（GCMAR）的值没有直接关系。下图为软件设定 GCMBR 互补 PWM 波的示例。

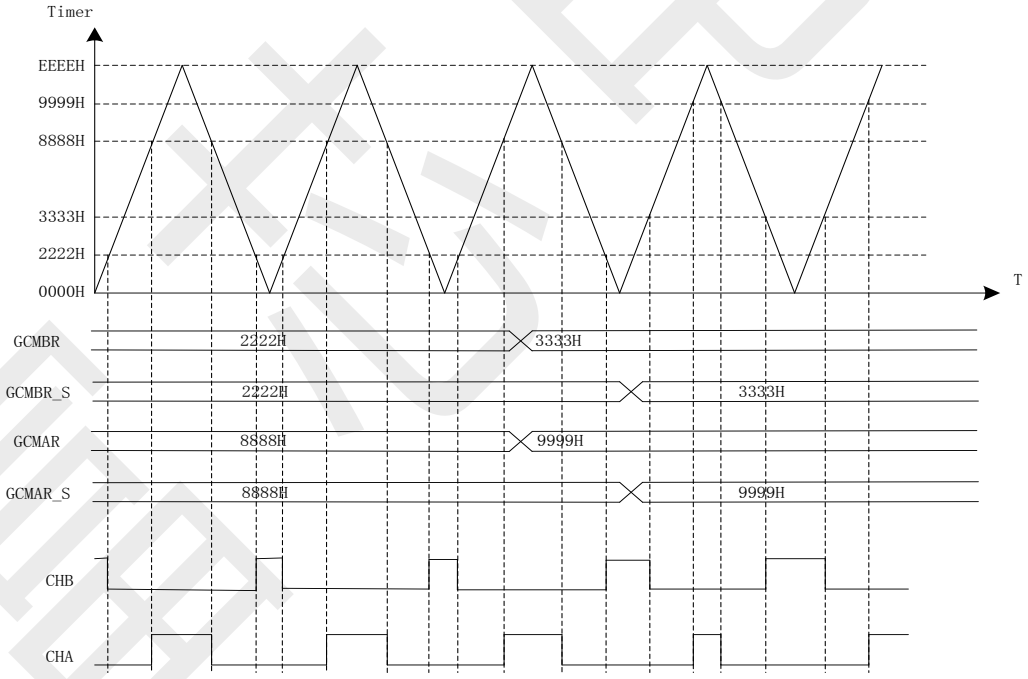


图 10-10

硬件设定 GCMBR 互补 PWM 输出

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

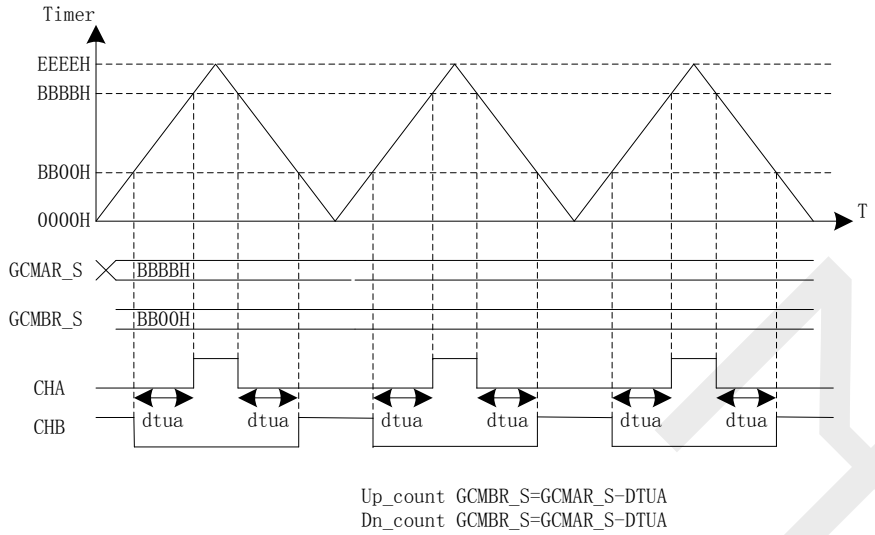


图 10-11

10.2.11 周期间隔响应

Timer1/2 的通用比较基准值寄存器（GCMAR，GCMBR），在计数比较匹配时可分别产生专用有效请求信号。

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

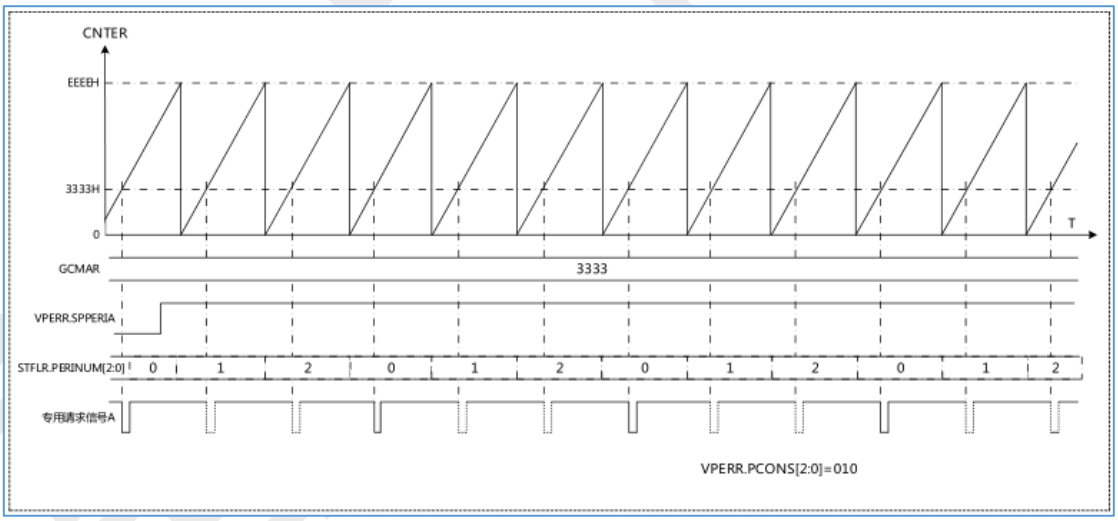


图 10-12

10.2.12 保护机制

Advanced Timer 可以对端口的输出状态进行保护控制。

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

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

例如，若设定 BRAKE_VAL=2’b10 时，则在 TIMx_CHA 端口正常输出期间，若输出无效条件 1 上产生刹车事件，则 TIMx_CHA 端口上输出变为高阻态。

10.2.13 中断说明

TIMER1/2 各含有 4 类共计 6 个中断。分别是 2 个通用计数比较匹配中断（含 2 个捕获输入中断）、2 个计数周期匹配中断、2 个刹车保护中断。

10.2.14 刹车保护

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

10.2.15 内部互连

- 电压比较器可以触发刹车功能。
- TIMER1/2 中断可以触发 ADC 采样功能。

10.2.16 与 TIM1 和 TIM2 相关寄存器定义

名字	地址	读写	复位值	描述
TIM1_CR	0xB0	读写	00000000	Timer1 控制寄存器
TIM1_FCONR	0xB1	读写	00000000	Timer1 时钟控制寄存器
TIM1_ARRL	0xB2	读写	00000000	Timer1 自动重载寄存器低八位
TIM1_ARRH	0xB3	读写	00000000	Timer1 自动重载寄存器高八位
TIM1_GCMARL	0xB4	读写	00000000	Timer1 比较捕获寄存器 A 低八位
TIM1_GCMARH	0xB5	读写	00000000	Timer1 比较捕获寄存器 A 高八位
TIM1_GCMBRL	0xB6	读写	00000000	Timer1 比较捕获寄存器 B 低八位
TIM1_GCMBRH	0xB7	读写	00000000	Timer1 比较捕获寄存器 B 高八位
TIM1_VPERR	0xB8	读写	00000000	Timer1 周期间隔响应控制寄存器
TIM1_DTUA	0xB9	读写	00000000	Timer1 死区事件寄存器
TIM1_BRAKE	0xBA	读写	00000000	Timer1 刹车控制寄存器
TIM1_DTR	0xBB	读写	00000000	Timer1 死区控制寄存器
TIM1_PCONRA	0xBC	读写	00000000	Timer1 端口 A 控制寄存器
TIM1_PCONRB	0xBD	读写	00000000	Timer1 端口 B 控制寄存器
TIM1_IE	0xBE	读写	00000000	Timer1 中断控制寄存器
TIM1_SR	0xBF	读写	00000000	Timer1 状态寄存器
TIM2_CR	0xC0	读写	00000000	Timer2 控制寄存器
TIM2_FCONR	0xC1	读写	00000000	Timer2 时钟控制寄存器

TIM2_ARRL	0xC2	读写	00000000	Timer2 自动重载寄存器低八位
TIM2_ARRH	0xC3	读写	00000000	Timer2 自动重载寄存器高八位
TIM2_GCMARL	0xC4	读写	00000000	Timer2 比较捕获寄存器 A 低八位
TIM2_GCMARH	0xC5	读写	00000000	Timer2 比较捕获寄存器 A 高八位
TIM2_GCMBRL	0xC6	读写	00000000	Timer2 比较捕获寄存器 B 低八位
TIM2_GCMBRH	0xC7	读写	00000000	Timer2 比较捕获寄存器 B 高八位
TIM2_VPERR	0xC8	读写	00000000	Timer2 周期间隔响应控制寄存器
TIM2_DTUA	0xC9	读写	00000000	Timer2 死区事件寄存器
TIM2_BRAKE	0xCA	读写	00000000	Timer2 刹车控制寄存器
TIM2_DTR	0xCB	读写	00000000	Timer2 死区控制寄存器
TIM2_PCONRA	0xCC	读写	00000000	Timer2 端口 A 控制寄存器
TIM2_PCONRB	0xCD	读写	00000000	Timer2 端口 B 控制寄存器
TIM2_IE	0xCE	读写	00000000	Timer2 中断控制寄存器
TIM2_SR	0xCF	读写	00000000	Timer2 状态寄存器

10.2.16.1 TIM1_CR (0xB0)

Bit	7	6	5	4	3	2	1	0
Name	THB_FILTER_EN	THA_FILTER_EN	-	SEL_SREG	DIR	MODE[1:0]		TIM1_EN
Reset	0	0	-	0	0	0	0	0
Type	R/W	R/W	-	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	THB_FILTER_EN	刹车输入 A 滤波控制 0 刹车输入 B 关数字滤波 1 刹车输入 B 开数字滤波 TIMER1/2 共用,只在 TIMER1 设定，TIMER2 共用 TIMER1 设定
6	THA_FILTER_EN	刹车输入 B 滤波控制 0 刹车输入 A 关数字滤波 1 刹车输入 A 开数字滤波 TIMER1/2 共用,只在 TIMER1 设定，TIMER2 共用 TIMER1 设定
5	N/A	保留位，读 0
4	SEL_SREG	影子寄存器控制 0 ARR GCMAR GCMBR 读到影子寄存器的值或捕获值 1 ARR GCMAR GCMBR 读到当前设定的值
3	DIR	计数器计数方向 0 向上计数 1 向下计数
2:1	MODE[1:0]	计数器计数模式 00 锯齿波计数模式 01 三角波 A 计数模式

		10　三角波 B 计数模式 11　保留
0	TIM1_EN	TIMER1 使能控制 0　关闭 TIMER1 1　使能 TIMER1

10.2.16.2 TIM1_FCONR (0xB1)

Bit	7	6	5	4	3	2	1	0
Name	-	CLK_SEL[2:0]			PRE_DIV[3:0]			
Reset	-	0	0	0	0	0	0	0
Type	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	N/A	保留位，读 0
6:4	CLK_SEL[2:0]	TIMER1 时钟源选择: 000　SYSCLK 001　看门狗时钟 32kHz 010　RTC 011　保留 100　TIM1_CHA 上升沿 101　TIM1_CHB 上升沿 110　TIM1_CHA 下降沿 111　TIM1_CHB 下降沿
3:0	PRE_DIV[3:0]	TIMER1 预分频选择: 0~15 对应 1~16 分频

10.2.16.3 TIM1_ARRL (0xB2)

Bit	7	6	5	4	3	2	1	0
Name	TIM1_ARRL							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	TIM1_ARRL	自动重载值寄存器低 8 位，需先写高 8 位再写低 8 位。

10.2.16.4 TIM1_ARRH (0xB3)

Bit	7	6	5	4	3	2	1	0
Name	TIM1_ARRH							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	TIM1_ARRH	自动重载值寄存器高 8 位，需先写高 8 位再写低 8 位。

10.2.16.5 TIM1_GCMARL (0xB4)

Bit	7	6	5	4	3	2	1	0
Name	TIM1_GCMARL							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	TIM1_GCMARL	计数模式下比较值，捕获模式下 CHA 捕获值， GCMAR 低 8 位，需先写高 8 位再写低 8 位。

10.2.16.6 TIM1_GCMARH (0xB5)

Bit	7	6	5	4	3	2	1	0
Name	TIM1_GCMARH							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	TIM1_GCMARH	计数模式下比较值，捕获模式下 CHA 捕获值， GCMAR 高 8 位，需先写高 8 位再写低 8 位。

10.2.16.7 TIM1_GCMBRL (0xB6)

Bit	7	6	5	4	3	2	1	0
Name	TIM1_GCMBRL							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	TIM1_GCMBRL	计数模式下比较值，捕获模式下 CHB 捕获值， GCMBR 低 8 位，需先写高 8 位再写低 8 位。

10.2.16.8 TIM1_GCMBRH (0xB7)

Bit	7	6	5	4	3	2	1	0
Name	TIM1_GCMBRH							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	TIM1_GCMBRH	计数模式下比较值，捕获模式下 CHB 捕获值， GCMAR 高 8 位，需先写高 8 位再写低 8 位。

10.2.16.9 TIM1_VPERR (0xB8)

Bit	7	6	5	4	3	2	1	0
Name	-	-	PCNTE[1:0]		-	PCNTS[2:0]		
Reset	-	-	0	0	-	0	0	0

Type	-	-	R/W	R/W	-	R/W	R/W	R/W

Bit	Name	Function
7:6	N/A	保留位，读 0
5:4	PCNTE[1:0]	周期间隔响应计数条件： 00 有效周期选择功能无效 01 锯齿波计数上、下溢点或三角波波峰作为计数条件 10 锯齿波计数上、下溢点或三角波波谷作为计数条件 11 锯齿波计数上、下溢点或三角波波谷、波峰作为计数条件
3	N/A	保留位，读 0
2:0	PCNTS[2:0]	周期间隔响应周期： 000 1 个周期响应一次 001 2 个周期响应一次 010 4 个周期响应一次 011 8 个周期响应一次 100 16 个周期响应一次 101 32 个周期响应一次 110 64 个周期响应一次 111 128 个周期响应一次

10.2.16.10 TIM1_DTUA (0xB9)

Bit	7	6	5	4	3	2	1	0
Name	TIM1_DTUA							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	TIM1_DTUA	TIMER1 死区时间设定值。

10.2.16.11 TIM1_BRAKE (0xBA)

Bit	7	6	5	4	3	2	1	0
Name	TIB_MOE	TIB_AOE	TIB_SEL	TIB_EN	TIA_MOE	TIA_AOE	TIA_SEL	TIA_EN
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	TIB_MOE	TIM1_CHB 主输出使能 刹车事件有效时会立即被同步清零。根据 AOE 的选择，通过软件置 1 或硬件自动置 1 1 TIM1_CHB 主输出有效 0 TIM1_CHB 主输出关闭
6	TIB_AOE	自动输出使能 1 有刹车事件产生时，MOE 可被软件和溢出事件置 1 0 有刹车事件产生时，MOE 只被软件置 1
5	TIB_SEL	选择 TIM1_CHB 刹车来源

		0 TIM1_CHB 刹车事件选择模拟比较器 0 比较输出 1 TIM1_CHB 刹车事件选择模拟比较器 1 输入
4	TIB_EN	刹车功能控制 1 TIM1_CHB 刹车有效 0 TIM1_CHB 刹车无效
3	TIA_MOE	TIM1_CHA 主输出使能 刹车事件有效时会立即被同步清零。根据 AOE 的选择，通过软件置 1 或硬件自动置 1 1 TIM1_CHA 主输出有效 0 TIM1_CHA 主输出关闭
2	TIA_AOE	TIM1_CHA 自动输出使能 1 有刹车事件产生时，MOE 可被软件和溢出事件置 1 0 有刹车事件产生时，MOE 只被软件置 1
1	TIA_SEL	选择 TIM1_CHA 刹车来源 0 TIM1_CHA 刹车事件选择模拟比较器 0 输出 1 TIM1_CHA 刹车事件选择模拟比较器 1 输出
0	TIA_EN	TIM1_CHA 刹车功能控制 1 TIM1_CHA 刹车有效 0 TIM1_CHA 刹车无效

10.2.16.12 TIM1_DTR (0xBB)

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	HW_CPWM	DTB_HO	DTB_EN	DTH_HO	DTA_EN
Reset	-	-	-	0	0	0	0	0
Type	-	-	-	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:5	N/A	保留位，读 0
4	HW_CPWM	控制 GCMBR 互补模式 0 硬件设定 GCMBR 互补 PWM 输出模式关 1 硬件设定 GCMBR 互补 PWM 输出模式开
3	DTH_B	控制死区输出状态 1 输出 B 死区置为高阻输出 0 输出 B 死区置为 0 或 1（由 GPIO 输出决定）
2	DTB_EN	死区控制使能 1 输出 B 死区控制有效 0 输出 B 死区控制无效
1	DTA_HO	控制死区输出状态 1 输出 A 死区置为高阻输出 0 输出 A 死区置为 0 或 1（由 GPIO 输出决定）
0	DTA_EN	死区控制使能 1 输出 A 死区控制有效 0 输出 A 死区控制无效

10.2.16.13 TIM1_PCONRA (0xBC)

Bit	7	6	5	4	3	2	1	0
Name	PA_INITVAL	CMPA_VAL[1:0]		PA_ENO	PA_FILTER_EN	CAPA_MODE[1:0]		CAPA_EN
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	PA_INITVAL	设置 TIM1_CHA 的输出： 1 TIM1_CHA 的初始值为 1 0 TIM1_CHA 的初始值为 0 TIMER1 关时设定有效，TIMER1 开时中间设定无效
6:5	CMPA_VAL[1:0]	配置 TIM1_CHA 比较输出值： 00 计数值小于比较值为 1，大于为 0 01 计数值大于比较值为 1，小于为 0 10 比较值匹配，输出取反前一状态 11 比较值匹配，输出保持前一状态
4	PA_ENO	TIM1_CHA 输出控制： 1 TIM1_CHA 输出打开 0 TIM1_CHA 输出关闭
3	PA_FILTER_EN	TIM1_CHA 输入滤波使能 1 TIM1_CHA 输入数字滤波打开 0 TIM1_CHA 输入数字滤波关闭
2:1	CAPA_MODE[1:0]	TIM1_CHA 捕获模式选择： 00 不捕获 01 捕获上升沿 10 捕获下降沿 11 捕获上升沿与下降沿
0	CAPA_EN	TIM1_CHA 捕获模式使能： 1 TIM1_CHA 捕获模式开 0 TIM1_CHA 捕获模式关

10.2.16.14 TIM1_PCONRB (0xBD)

Bit	7	6	5	4	3	2	1	0
Name	PB_INITVAL	CMPB_VAL[1:0]		PB_ENO	PB_FILTER_EN	CAPB_MODE[1:0]		CAPB_EN
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	PB_INITVAL	设置 TIM1_CHB 的输出： 1 TIM1_CHB 的初始值为 1 0 TIM1_CHB 的初始值为 0 TIMER1 关时设定有效，TIMER1 开时中间设定无效
6:5	CMPB_VAL[1:0]	配置 TIM1_CHB 比较输出值： 00 计数值小于比较值为 1，大于为 0

		01 计数值大于比较值为 1，小于为 0 10 比较值匹配，输出取反前一状态 11 比较值匹配，输出保持前一状态
4	PB_ENO	TIM1_CHB 输出控制： 1 TIM1_CHB 输出打开 0 TIM1_CHB 输出关闭
3	PB_FILTER_EN	TIM1_CHB 输入滤波使能： 1 TIM1_CHB 输入数字滤波打开 0 TIM1_CHB 输入数字滤波关闭
2:1	CAPB_MODE[1:0]	TIM1_CHB 捕获模式选择： 00 不捕获 01 捕获上升沿 10 捕获下降沿 11 捕获上升沿与下降沿
0	CAPB_EN	TIM1_CHB 捕获模式使能： 1 TIM1_CHB 捕获模式开 0 TIM1_CHB 捕获模式关

10.2.16.15 TIM1_IE (0xBE)

Bit	7	6	5	4	3	2	1	0
Name	-	-	BRAKEB_IE	BRAKEA_IE	CMPB_IE	CMPA_IE	UD_IE	OV_IE
Reset	-	-	0	0	0	0	0	0
Type	-	-	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:6	N/A	保留位，读 0
5	BRAKEB_IE	TIM1_CHB 刹车中断使能： 1 TIM1_CHB 刹车中断使能开 0 TIM1_CHB 刹车中断使能关
4	BRAKEA_IE	TIM1_CHA 刹车中断使能： 1 TIM1_CHA 刹车中断使能开 0 TIM1_CHA 刹车中断使能关
3	CMPB_IE	TIM1_CHB 比较或者捕获中断使能： 1 TIM1_CHB 比较匹配或者捕获中断开 0 TIM1_CHB 比较匹配或者捕获中断关
2	CMPA_IE	TIM1_CHA 比较或者捕获中断使能： 1 TIM1_CHA 比较匹配或者捕获中断开 0 TIM1_CHA 比较匹配或者捕获中断关
1	UD_IE	下溢中断使能： 1 计数器下溢中断开 0 计数器下溢中断关
0	OV_IE	上溢中断使能： 1 计数器上溢中断开 0 计数器上溢中断关

10.2.16.16 TIM1_SR (0xBF)

Bit	7	6	5	4	3	2	1	0
Name	-	-	BRAKEB_IF	BRAKEA_IF	CMB_IF	CMA_IF	UD_IF	OV_IF
Reset	-	-	0	0	0	0	0	0
Type	-	-	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:6	N/A	保留位，读 0
5	BRAKEB_IF	TIM1_CHB 刹车中断标志： 1 TIM1_CHB 输入发生刹车事件，刹车信号无效时 0 CHB 输入未发生刹车事件 写 1 清零该标志位
4	BRAKEA_IF	TIM1_CHA 刹车中断标志： 1 TIM1_CHA 输入发生刹车事件，刹车信号无效时 0 TIM1_CHA 输入未发生刹车事件 写 1 清零该标志位
3	CMPB_IF	TIM1_CHB 比较或者捕获中断标志： 1 发生 TIM1_CHB 比较匹配或者捕获，写 1 清零 0 未发生 TIM1_CHB 比较匹配或者捕获 写 1 清零该标志位
2	CMPA_IF	TIM1_CHA 比较或者捕获中断使能： 1 TIM1_CHA 比较匹配或者捕获中断开 0 TIM1_CHA 比较匹配或者捕获中断关 写 1 清零该标志位
1	UD_IF	TIMER1 计数器下溢中断标志： 1 计数器发生下溢，写 1 清零 0 计数器未发生下溢 写 1 清零该标志位
0	OV_IF	TIMER1 计数器上溢中断标志： 1 计数器发生上溢，写 1 清零 0 计数器未发生上溢 写 1 清零该标志位

10.2.16.17 TIM2_CR (0xC0)

Bit	7	6	5	4	3	2	1	0
Name	-	-	CAP_TIM1	SEL_SREG	DIR	MODE[1:0]		TIM2_EN
Reset	-	-	0	0	0	0	0	0
Type	-	-	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:6	N/A	保留位，读 0
5	CAP_TIM1	TIMER2 捕获 TIMER1 控制： 0 TIMER2 不捕获 TIMER1 1 TIMER2 捕获 TIMER1

4	SEL_SREG	影子寄存器控制： 0 ARR GCMAR GCMBR 读到影子寄存器的值或捕获值 1 ARR GCMAR GCMBR 读到当前设定的值
3	DIR	计数器计数方向： 0 向上计数 1 向下计数
2:1	MODE[1:0]	计数器计数模式 00 锯齿波计数模式 01 三角波 A 计数模式 10 三角波 B 计数模式 11 保留
0	TIM2_EN	TIMER2 使能控制： 0 关闭 TIMER2 1 使能 TIMER2

10.2.16.18 TIM2_FCONR (0xC1)

Bit	7	6	5	4	3	2	1	0
Name	-	CLK_SEL[2:0]			PRE_DIV[3:0]			
Reset	-	0	0	0	0	0	0	0
Type	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	N/A	保留位，读 0
6:4	CLK_SEL[2:0]	TIMER2 时钟源选择： 000 SYSCLK 001 看门狗时钟 32kHz 010 RTC 011 保留 100 TIM2_CHA 上升沿 101 TIM2_CHB 上升沿 110 TIM2_CHA 下降沿 111 TIM2_CHB 下降沿
3:0	PRE_DIV[3:0]	TIMER2 预分频选择： 0~15 对应 1~16 分频

10.2.16.19 TIM2_ARRL (0xC2)

Bit	7	6	5	4	3	2	1	0
Name	TIM2_ARRL							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	TIM2_ARRL	自动重载值寄存器低 8 位，需先写高 8 位再写低 8 位。

10.2.16.20 TIM2_ARRH (0xC3)

Bit	7	6	5	4	3	2	1	0
Name	TIM2_ARRH							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	TIM2_ARRH	自动重载值寄存器高 8 位，需先写高 8 位再写低 8 位。

10.2.16.21 TIM2_GCMARL (0xC4)

Bit	7	6	5	4	3	2	1	0
Name	TIM2_GCMARL							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	TIM2_GCMARL	计数模式下比较值，捕获模式下 CHA 捕获值， GCMAR 低 8 位，需先写高 8 位再写低 8 位。

10.2.16.22 TIM2_GCMARH (0xC5)

Bit	7	6	5	4	3	2	1	0
Name	TIM2_GCMARH							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	TIM2_GCMARH	计数模式下比较值，捕获模式下 CHA 捕获值， GCMAR 高 8 位，需先写高 8 位再写低 8 位。

10.2.16.23 TIM2_GCMBRL (0xC6)

Bit	7	6	5	4	3	2	1	0
Name	TIM2_GCMBRL							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	TIM2_GCMBRL	计数模式下比较值，捕获模式下 CHB 捕获值， GCMBR 低 8 位，需先写高 8 位再写低 8 位。

10.2.16.24 TIM2_GCMBRH (0xC7)

Bit	7	6	5	4	3	2	1	0
Name	TIM2_GCMBRH							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	TIM2_GCMBRH	计数模式下比较值，捕获模式下 CHB 捕获值， GCMAR 高 8 位，需先写高 8 位再写低 8 位。

10.2.16.25 TIM2_VPERR (0xC8)

Bit	7	6	5	4	3	2	1	0
Name	-	-	PCNTE[1:0]		-	PCNTS[2:0]		
Reset	-	-	0	0	-	0	0	0
Type	-	-	R/W	R/W	-	R/W	R/W	R/W

Bit	Name	Function
7:6	N/A	保留位，读 0
5:4	PCNTE[1:0]	周期间隔响应计数条件： 00 有效周期选择功能无效 01 锯齿波计数上、下溢点或三角波波峰作为计数条件 10 锯齿波计数上、下溢点或三角波波谷作为计数条件 11 锯齿波计数上、下溢点或三角波波谷、波峰作为计数条件
3	N/A	保留位，读 0
2:0	PCNTS[2:0]	周期间隔响应周期： 000 1 个周期响应一次 001 2 个周期响应一次 010 4 个周期响应一次 011 8 个周期响应一次 100 16 个周期响应一次 101 32 个周期响应一次 110 64 个周期响应一次 111 128 个周期响应一次

10.2.16.26 TIM2_DTUA (0xC9)

Bit	7	6	5	4	3	2	1	0
Name	TIM2_DTUA							
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:0	TIM2_DTUA	TIMER2 死区时间设定值。

10.2.16.27 TIM2_BRAKE (0xCA)

Bit	7	6	5	4	3	2	1	0
Name	TIB_MOE	TIB_AOE	TIB_SEL	TIB_EN	TIA_MOE	TIA_AOE	TIA_SEL	TIA_EN
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	TIB_MOE	TIM2_CHB 主输出使能 刹车事件有效时会立即被同步清零。根据 AOE 的选择，通过软件置 1 或硬件自动置 1 1 TIM2_CHB 主输出有效 0 TIM2_CHB 主输出关闭
6	TIB_AOE	自动输出使能 1 有刹车事件产生时，MOE 可被软件和溢出事件置 1 0 有刹车事件产生时，MOE 只被软件置 1
5	TIB_SEL	选择刹车来源 1 TIM2_CHB 刹车事件选择模拟比较器 0 输出 0 TIM2_CHB 刹车事件选择模拟比较器 1 输出
4	TIB_EN	刹车功能控制 1 TIM2_CHB 刹车有效 0 TIM2_CHB 刹车无效
3	TIA_MOE	TIM1_CHA 主输出使能 刹车事件有效时会立即被同步清零。根据 AOE 的选择，通过软件置 1 或硬件自动置 1 1 TIM2_CHA 主输出有效 0 TIM2_CHA 主输出关闭
2	TIA_AOE	TIM2_CHA 自动输出使能 1 有刹车事件产生时，MOE 可被软件和溢出事件置 1 0 有刹车事件产生时，MOE 只被软件置 1
1	TIA_SEL	选择 TIM2_CHA 刹车来源 1 TIM2_CHA 刹车事件选择模拟比较器 0 输出 0 TIM2_CHA 刹车事件选择模拟比较器 1 输出
0	TIA_EN	TIM2_CHA 刹车功能控制 1 TIM2_CHA 刹车有效 0 TIM2_CHA 刹车无效

10.2.16.28 TIM2_DTR (0xCB)

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	HW_CPWM	DTB_HO	DTB_EN	DTA_HO	DTA_EN
Reset	-	-	-	0	0	0	0	0
Type	-	-	-	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:5	N/A	保留位，读 0
4	HW_CPWM	控制 GCMBR 互补模式 0 硬件设定 GCMBR 互补 PWM 输出模式关 1 硬件设定 GCMBR 互补 PWM 输出模式开
3	DTB_HO	死区控制使能 1 输出 B 死区控制有效 0 输出 B 死区控制无效
2	DTB_EN	死区控制使能 1 输出 B 死区控制有效

		0 输出 B 死区控制无效
1	DTA_HO	控制死区输出状态 1 输出 B 死区置为高阻输出 0 输出 B 死区置为 0 或 1（由 GPIO 输出决定）
0	DTA_EN	死区控制使能 1 输出 A 死区控制有效 0 输出 A 死区控制无效

10.2.16.29 TIM2_PCONRA (0xCC)

Bit	7	6	5	4	3	2	1	0
Name	PA_INITVAL	CMPA_VAL[1:0]		PA_ENO	PA_FILTER_EN	CAPA_MODE[1:0]		CAPA_EN
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	PA_INITVAL	设置 TIM2_CHA 的输出： 1 TIM2_CHA 的初始值为 1 0 TIM2_CHA 的初始值为 0 TIMER1 关时设定有效，TIMER1 开时中间设定无效
6:5	CMPA_VAL[1:0]	配置 TIM2_CHA 比较输出值： 00 计数值小于比较值为 1，大于为 0 01 计数值大于比较值为 1，小于为 0 10 比较值匹配，输出取反前一状态 11 比较值匹配，输出保持前一状态
4	PA_ENO	TIM2_CHA 输出控制： 1 TIM2_CHA 输出打开 0 TIM2_CHA 输出关闭
3	PA_FILTER_EN	TIM2_CHA 输入滤波使能 1 TIM2_CHA 输入数字滤波打开 0 TIM2_CHA 输入数字滤波关闭
2:1	CAPA_MODE[1:0]	TIM2_CHA 捕获模式选择： 00 不捕获 01 捕获上升沿 10 捕获下降沿 11 捕获上升沿与下降沿
0	CAPA_EN	TIM2_CHA 捕获模式使能： 1 TIM2_CHA 捕获模式开 0 TIM2_CHA 捕获模式关

10.2.16.30 TIM2_PCONRB (0xCD)

Bit	7	6	5	4	3	2	1	0
Name	PB_INITVAL	CMPB_VAL[1:0]		PB_ENO	PB_FILTER_EN	CAPB_MODE[1:0]		CAPB_EN
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	PB_INITVAL	设置 TIM2_CHB 的输出： 1 TIM2_CHB 的初始值为 1 0 TIM2_CHB 的初始值为 0 TIMER2 关时设定有效，TIMER2 开时中间设定无效
6:5	CMPB_VAL[1:0]	配置 TIM2_CHB 比较输出值： 00 计数值小于比较值为 1，大于为 0 01 计数值大于比较值为 1，小于为 0 10 比较值匹配，输出取反前一状态 11 比较值匹配，输出保持前一状态
4	PB_ENO	TIM2_CHB 输出控制： 1 TIM2_CHB 输出打开 0 TIM2_CHB 输出关闭
3	PB_FILTER_EN	TIM2_CHB 输入滤波使能： 1 TIM2_CHB 输入数字滤波打开 0 TIM2_CHB 输入数字滤波关闭
2:1	CAPB_MODE[1:0]	TIM2_CHB 捕获模式选择： 00 不捕获 01 捕获上升沿 10 捕获下降沿 11 捕获上升沿与下降沿
0	CAPB_EN	TIM2_CHB 捕获模式使能： 1 TIM2_CHB 捕获模式开 0 TIM2_CHB 捕获模式关

10.2.16.31 TIM2_IE (0xCE)

Bit	7	6	5	4	3	2	1	0
Name	-	-	BRAKEB_IE	BRAKEA_IE	CMPB_IE	CMPA_IE	UD_IE	OV_IE
Reset	-	-	0	0	0	0	0	0
Type	-	-	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:6	N/A	保留位，读 0
5	BRAKEB_IE	TIM2_CHB 刹车中断使能： 1 TIM2_CHB 刹车中断使能开 0 TIM2_CHB 刹车中断使能关
4	BRAKEA_IE	TIM2_CHA 刹车中断使能： 1 TIM2_CHA 刹车中断使能开 0 TIM2_CHA 刹车中断使能关
3	CMPB_IE	TIM2_CHB 比较或者捕获中断使能： 1 TIM2_CHB 比较匹配或者捕获中断开 0 TIM2_CHB 比较匹配或者捕获中断关
2	CMPA_IE	TIM2_CHA 比较或者捕获中断使能：

		1 TIM2_CHA 比较匹配或者捕获中断开 0 TIM2_CHA 比较匹配或者捕获中断关
1	UD_IE	下溢中断使能： 1 计数器下溢中断开 0 计数器下溢中断关
0	OV_IE	上溢中断使能： 1 计数器上溢中断开 0 计数器上溢中断关

10.2.16.32 TIM2_SR (0xCF)

Bit	7	6	5	4	3	2	1	0
Name	-	-	BRAKEB_IF	BRAKEA_IF	CMB_IF	CMA_IF	UD_IF	OV_IF
Reset	-	-	0	0	0	0	0	0
Type	-	-	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:6	N/A	保留位，读 0
5	BRAKEB_IF	TIM2_CHB 刹车中断标志： 1 TIM2_CHB 输入发生刹车事件，刹车信号无效时 0 CHB 输入未发生刹车事件 写 1 清零该标志位
4	BRAKEA_IF	TIM2_CHA 刹车中断标志： 1 TIM2_CHA 输入发生刹车事件，刹车信号无效时 0 TIM2_CHA 输入未发生刹车事件 写 1 清零该标志位
3	CMPB_IF	TIM2_CHB 比较或者捕获中断标志： 1 发生 TIM2_CHB 比较匹配或者捕获，写 1 清零 0 未发生 TIM2_CHB 比较匹配或者捕获 写 1 清零该标志位
2	CMPA_IF	TIM2_CHA 比较或者捕获中断使能： 1 TIM2_CHA 比较匹配或者捕获中断开 0 TIM2_CHA 比较匹配或者捕获中断关 写 1 清零该标志位
1	UD_IF	TIMER1 计数器下溢中断标志： 1 计数器发生下溢，写 1 清零 0 计数器未发生下溢 写 1 清零该标志位
0	OV_IF	TIMER1 计数器上溢中断标志： 1 计数器发生上溢，写 1 清零 0 计数器未发生上溢 写 1 清零该标志位

10.3 UART

10.3.1 概述

UART 模块可以实现和外部设备异步通讯的功能。该模块包含以下主要特性。

- 半双工
- 异步模式
- LSB 在前
- 波特率发生器
- 8 位数据
- 帧错误检测
- 接收数据超限检测
- 两个独立中断，传输完成中断和帧错误中断

10.3.2 结构框图

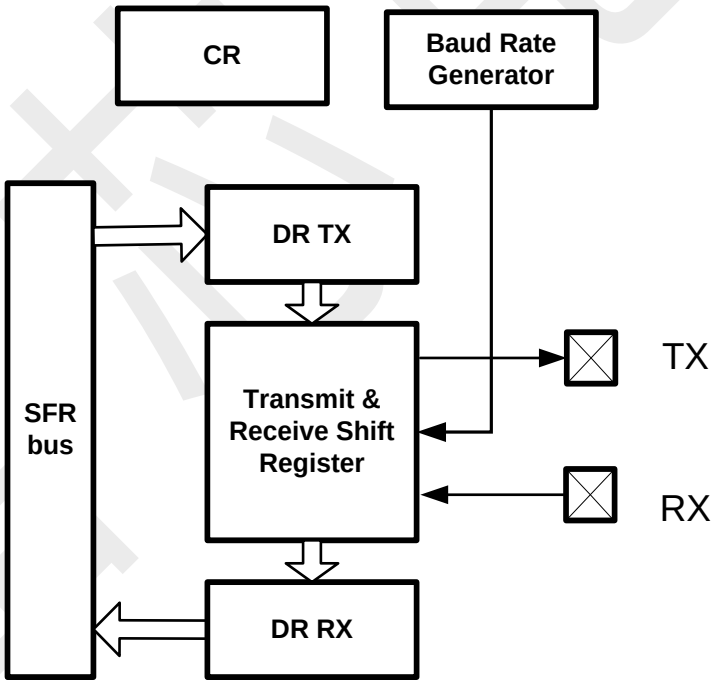


图 10-13 UART 结构框图

10.3.3 时钟发生器

时钟源来自系统时钟 SCK1、SCK2 和 SCK3 中的一个。

10.3.4 UART 发送

当配置 DIR 为 0 的时候，UART 工作在发送模式。UART 使能后，写 UART 数据寄存器会启动一次发送行为。如果发送完成，那么发送状态退回到空闲。一次发送完成后会置位发送完成标志，该标志可以触发 UART 中断。发送完成标志可以通过软件清零。

10.3.5 UART 接收

配置 DIR 为 1 的时候 UART 工作在接收模式。模块使能后开始检测 RX 数据输入。如果检测到开始信号，UART 开始接收数据。如果成功检测到停止位，那么认为这一帧数据是有效的，数据会存入到 UART 数据寄存器，同时置位接收成功标志。如果接收标志在数据接收完成更新到 UART 数据寄存器的时候有效，那么会置位接收数据超限标志。为了确保不触发错误的接收数据超限标志，用户必须在接收完成一帧数据后清除接收标志。

10.3.6 与 UART 相关寄存器定义

名字	地址	读写	复位值	描述
UART0DR	0xE4	读写	00000000	UART 数据寄存器
UART0CR	0xE5	读写	00000000	UART 控制寄存器
UART0SR	0xE6	读写	00000000	UART 状态寄存器
UART0CFG	0xE7	读写	00000000	UART 配置寄存器

10.3.6.1 UART0DR (0xE4)

Bit	7	6	5	4	3	2	1	0
Name	DATA							
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	DATA	发送模式该寄存器只能写。该寄存器只能在 UART 使能之后才能写入。 接收模式下只能读，读取内容表示接收到的数据。

10.3.6.2 UART0CR (0xE5)

Bit	7	6	5	4	3	2	1	0
Name	IE	-	-	PSEL	PAR_ODD	PAR_EN	TNR	EN
Reset	0	-	-	0	0	0	0	0
Type	R/W	-	-	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	IE	0 = 发送完成或者接受满不产生中断 1 = 发送完成或者接受满产生中断
6:5	N/A	保留位，读 0

4	PSEL	0 选择 PT11 为接收、PT12 为发送口 1 选择 PT13 为发送、PT14 为接收口
3	PAR_ODD	0 = 偶校验 1 = 奇校验 必须使能奇偶校验，校验才会生效。
2	PAR_EN	0 = 关闭奇偶校验 1 = 使能奇偶校验 接收模式下，收到的第 9 位数据数据位奇偶校验位；发送模式下，发送的第 9 位数据位前面 8 位数据的校验值。
1	TNR	0 = 接收模式 1 = 发送模式
0	EN	0 = 模块关闭 1 = 模块使能

10.3.6.3 UARTOSR (0xE6)

Bit	7	6	5	4	3	2	1	0
Name	RX_FULL	RX_ACTIVE	ERR_FRAME	ERR_PAR	OVERRUN	–	–	TX_COMPLETE
Reset	0	0	0	0	0	–	–	0
Type	R	R	R/W1C	R/W1C	R/W1C	–	–	R/W1C

Bit	Name	Function
7	RX_FULL	0 = 没有接收到数据 1 = 接收到了数据 读数据寄存器会清该标志位。发送模式下该位常为 0。
6	RX_ACTIVE	0 = 没有接收数据 1 = 正在接收数据 发送模式下该位常为 0。
5	ERR_FRAME	0 = 没有发生帧错误 1 = 发生帧错误 该位只有在接收模式下有效，接收数据时如果停止位收到低电平会触发帧错误。发送模式下该位常为 0。写 1 清零。
4	ERR_PAR	0 = 没有发生奇偶校验错误 1 = 发生奇偶校验错误 接收模式下，如果数据校验错误会置 1。发送模式下该位常为 0。写 1 清零该位。
3	OVERRUN	0 = 没有接收超限 1 = 接收超限 接收模式下，如果接收到了数据后又收到了数据会将该位置 1。 发送模式下常为 0。写 1 清零该标志位。
2	N/A	保留位，读 0
1	N/A	保留位，读 0
0	TX_COMPLETE	0 = 发送没有完成 1 = 发送完成 发送模式下，如果发送完成会将该位置 1。接收模式下常为 0。写 1 清零该位。

10.3.6.4 UART0CFG (0xE7)

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	CKSEL[1:0]	
Reset	-	-	-	-	-	-	0	0
Type	-	-	-	-	-	-	R/W	R/W

Bit	Name	Function
7:2	N/A	保留位，读 0
1:0	CKSEL[1:0]	选择 UART0 时钟源： 00 选择 SCK1 01 选择 SCK2 10/11 选择 SCK3

10.3.7 波特率设置

波特率时钟来自 SCK1、SCK2、SCK3 三个时钟源。UART 发送时，使用时钟源作为波特率发生器；UART 接收时，使用时钟源的 4 分频来作为波特率时钟。注意：发送和接收时钟频率不一样，接收是发送的 4 倍。另外在高温下波特率建议不要使用 115200，误差为 3.5%。

示例 1，使用 SCK3 配置发送 9600bps
16000000/9600=1666.6≈16*104

```
SCR_PCLK_DIV12 = 0xff; //SCK1 = Fsys/16
SCR_PCLK_DIV3 = 103; //SCK3 = Fsck1/104
SCR_PCLK_CR = 0xf2;  //sck3 select clock source is sck1
```

示例 2，使用 SCK3 配置接收 9600bps
16000000/(9600*4)=416.6≈4*104

```
SCR_PCLK_DIV12 = 0x3f; //SCK1 = Fsys/4
SCR_PCLK_DIV3 = 103; //SCK3 = Fsck1/104
SCR_PCLK_CR = 0xf2;  //sck3 select clock source is sck1
```

10.4 I2C

10.4.1 概述

I2C 是一种简单、双向的二进制同步串行总线，只需两根线即可在连接于总线上的器件之间传送信息。下图为 I2C 的架构图，MCU 通过总线访问 I2C 内部寄存器控制 I2C 的传输过程，I2C 通过两个双向的 GPIO 口与外部连接，发送或接收数据。

I2C 模块可以配置为主机或者从机模式或者主从模式。包含以下特性。

- 主机或者从机模式
- 多主机仲裁
- 速率 5Kbps、100Kbps、400Kbps
- 7 位从机地址
- 支持中断

10.4.2 结构框图

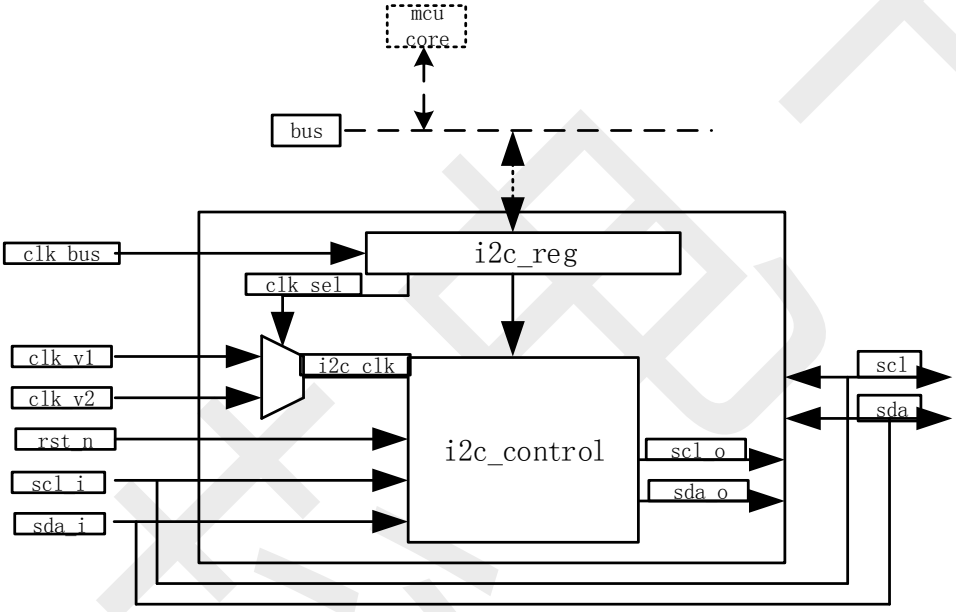


图 10-14 I2C 结构框图

10.4.3 应用描述

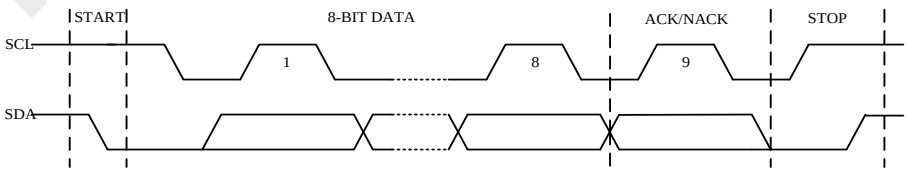
I2C 支持主从模式下的数据发送和接收。

2.2.1 基本数据传输方式

主器件产生传输用的时钟（SCL）信号，开始信号（START）和结束信号（STOP）。数据（SDA）必须在时钟的低电平时改变，并在高电平时保持。

SCL 为高时，检测到 SDA 上有由高到低的跳变，为 START；

SCL 为高时，检测到 SDA 上有由低到高的跳变，为 STOP。

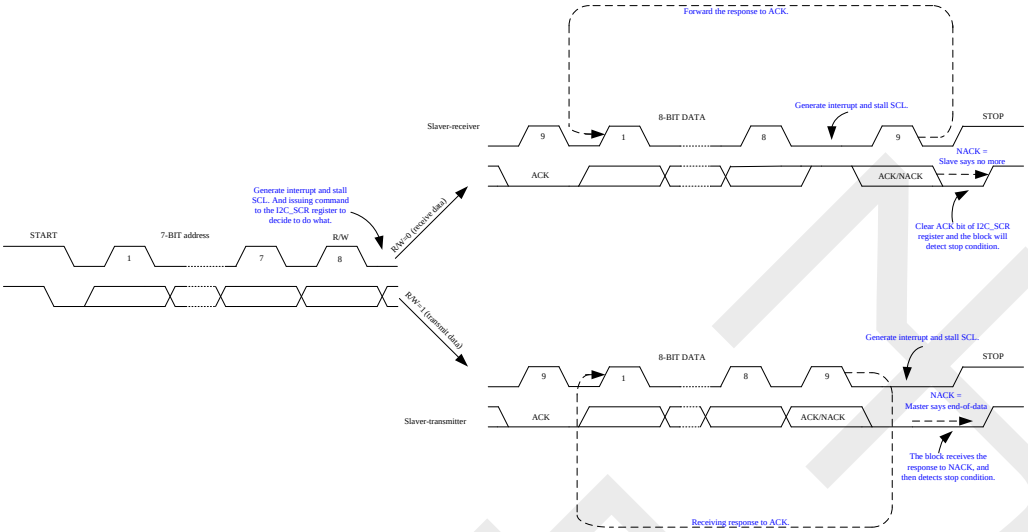


2.2.2 从模式（slave）

从模式下，会持续监听总线上是否有 START 信号。当监听到 START，会收到 8bit 的数据，其中包括 7bit

的 address 和 1bit 的 R/W 标志，从器件会根据收到的地址来确认是否响应主器件的读写请求。

如果地址正确，确认响应主器件的请求，从器件会根据 R/W 标志确认是传输数据还是接收数据，过程如图所示



从器件成功发送 1byte 数据过程如下:

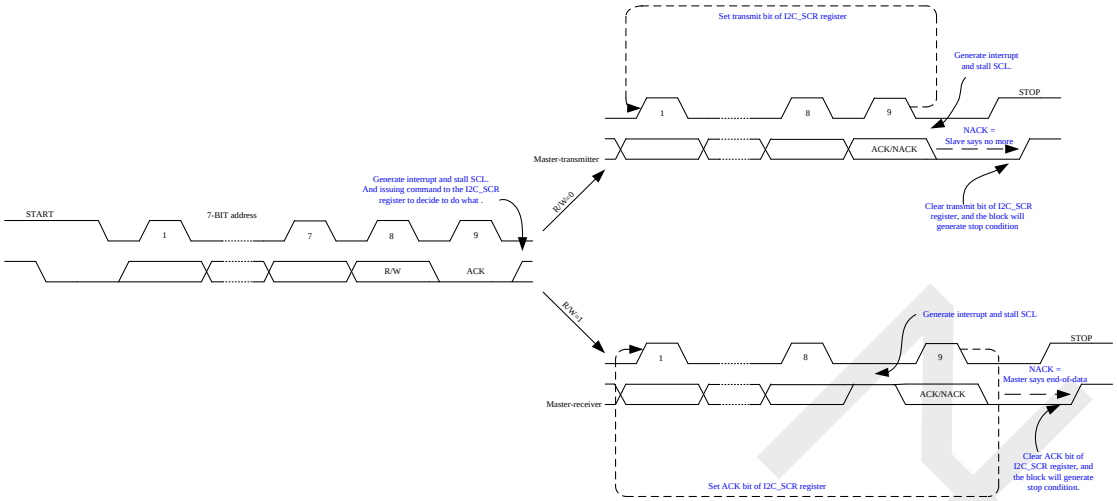
1) 确认寄存器都在初始状态.
2) 打开从模式（I2C_CR），处于监听状态.
   收到 8-bit data (slave address)后产生中断.
3) 将要发送的数据写入 I2C_DR
4) ACK bit 和 transmit bit 置 1（I2C_STAT）.
5) Byte Complete bit 置 1（I2C_STAT）.
   收到 8-bit data 和响应后产生中断.
6) 检查 LRB bit（I2C_STAT）.
   重复步骤 3~6，可以发送多 byte 数据

从器件成功接收 1byte 数据过程如下:

1) 确认寄存器都在初始状态.
2) 打开从模式（I2C_CR），处于监听状态.
   收到 8-bit data (slave address)后产生中断.
3) ACK bit 置 1，transmit bit 清 0（I2C_STAT）.
4) Byte Complete bit 置 1（I2C_STAT）.
   收到 8-bit data 后产生中断..
5) ACK bit 清 0（I2C_STAT）.
   重复步骤 3~4，可以接收多 byte 数据

2.2.3 主模式

主模式下，发起一个传送请求前，主设备必须先判断总线是否处于空闲状态。当总线上有设备在传输数据时，总线忙状态位（Bus Busy）会一直置为 1，直到检测到一个 STOP 信号，此时，当前设备获得总线使用权，启动一个读/写过程。



主器件成功发送 1byte 数据过程如下:

1) 确认寄存器都在初始状态.
2) 打开主模式（ I2C_CR ）.
3) 将数据(slave address+W)写入 I2C_DR.
4) Start Gen bit 置 1（ I2C_MCR ）.
   主设备发送完 8bit 数据并收到 ACK，产生中断.
5) 将要发送数据写入 I2C_DR.
6) Transmit bit 置 1（I2C_STAT）.
   主设备发送完 8bit 数据并收到 ACK，产生中断.
7) 发送完成，Transmit bit 清零（I2C_STAT register）.
   重复步骤 5~6，可以发送多 byte 数据。

主器件成功接收 1byte 数据过程如下:

1) 确认寄存器都在初始状态.
2) 打开主模式（ I2C_CR ）.
3) 将数据(slave address+W)写入 I2C_DR.
4) Start Gen bit 置 1（ I2C_MCR ）.
   主设备发送完 8bit 数据并收到 ACK，产生中断.
5) Transmit bit 清 0（I2C_STAT）.
   主设备收到 8bit 数据，产生中断.
6) 如果需要接收更多数据，ACK bit 置 1，接收完成 ACK bit 置 0 .
   重复步骤 5~6，能接收多 byte 数据。

10.4.4 中断

I2C 提供 5 种类型的中断:

- 总线错误中断
- 停止中断
- NACK 中断
- 硬件地址匹配中断

- 传输完成中断

10.4.5 波特率设置

主机模式下，发送时钟来自时钟源的 17 分频。

10.4.6 与 I2C 相关寄存器定义

名字	地址	读写	复位值	描述
I2C_ADDR	0xA1	读写	01100110	I2C 从机地址寄存器
I2C_CR	0xA2	读写	00000001	I2C 控制寄存器
I2C_STAT	0xA3	读写	00000000	I2C 状态寄存器
I2C_DR	0xA4	读写	00000000	I2C 数据寄存器
I2C_MCR	0xA5	读写	00000000	I2C 主机控制寄存器

10.4.6.1 I2C_ADDR (0xA1)

Bit	7	6	5	4	3	2	1	0
Name	HwAddrEn	Slave Address[6:0]						
Reset	0	1	1	0	0	1	1	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	HwAddrEn	1 = 打开地址比较功能 0 = 关掉地址比较功能 只用于从模式下。 I2C_ADDR[6:0]为当前 I2C 设备号， HwAddrEn 为 1，收到请求后，会比较收到的地址是否与 Slave Address 一致，如果一致，则响应请求，不一致则不响应； HwAddrEn 为 0，会响应收到的所有请求。
6:0	Slave Address [6:0]	只用于从模式，当前设备的地址。

10.4.6.2 I2C_CR (0xA2)

Bit	7	6	5	4	3	2	1	0
Name	I2C IE	-	Bus Error IE	Stop IE	-	Clk_sel	Enable Master	Enable Slave
Reset	0	-	0	0	-	0	0	1
Type	R/W	-	R/W	R/W	-	R/W	R/W	R/W

Bit	Name	Function
7	I2C IE	1 = 打开 I2C 全部中断 0 = 关闭 I2C 全部中断

6	N/A	保留位，读 0
5	Bus Error IE	1 = 打开 Bus Error 中断 0 = 关闭 Bus Error 中断.
4	Stop IE	1 = 打开结束中断 0 = 关闭结束中断
3	N/A	保留位，读 0
2	Clk_sel	0 = SCK1 1 = SCK2
1:0	Enable Master or Slave	00 = 主模式关&从模式关 01 = 主模式关&从模式开 10 = 主模式开&从模式关 11 = 主模式开&从模式开

10.4.6.3 I2C_STAT (0xA3)

Bit	7	6	5	4	3	2	1	0
Name	Bus Error	Lost Arb	Stop Status	ACK	Address	Transmit	LRB	Trans Complete
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	Bus Error(状态位)	只用于主模式，数据传送过程中检测到总线上有开始或结束条件时置 1。 只能通过写 0 清除。 注意：若发生了 Bus Error，则需要配置成非主机模式或关掉 I2C。
6	Lost Arb(状态位)	只用于主模式，失去对总线的控制权时置 1； 可以通过写 0 清除； 每次检测到开始信号都会自动清零。 注意：若主机失去对总线控制，则需要配置成非主机模式或关掉 I2C。
5	Stop Status(状态位)	检测到结束状态时置 1； 只能通过写 0 清除。
4	ACK(控制位)	1 = 发送 ack 0 = 不发送 ack（nack）
3	Address(状态位)	收到一个地址时置 1； 只能通过写 0 清除。
2	Transmit(状态位)	1 = 发送模式 0 = 接收模式
1	LRB(状态位)	1 = 最后收到的 bit 是 NACK 0 = 最后收到的 bit 是 ACK 写 0 清除或者检测到 START 信号清除。
0	Trans Complete (状态位)	单字节方式： 1：接收完成 发送模式：8bits 数据传送完成并收到响应（ACK 或者 NACK）. 接收模式：8bits 数据接收完成. 写 0 清除或者检测到 START 信号清除。

10.4.6.4 I2C_DR (0xA4)

Bit	7	6	5	4	3	2	1	0
Name	Data							
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	Data	主从模式接收，保存收到的数据，只读； 主模式产生开始信号前，需写入要发送到总线上的地址； 主从模式开始发送数据前，需写入要发送到客户端的数据。

10.4.6.5 I2C_MCR (0xA5)

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	Bus Busy	Master Mode	Restart Gen	Start Gen
Reset	-	-	-	-	0	0	0	0
Type	-	-	-	-	R	R	R/W	R/W

Bit	Name	Function
7:4	N/A	保留位，读 0
3	Bus Busy	检测到开始信号，状态置为 1； 检测到结束信号，状态置为 0。
2	Master Mode	产生开始信号，状态置为 1； 产生结束信号，状态置为 0。
1	Restart Gen	1 传送过程中收到响应为 NACK，重启传送过程，重新传送。
0	Start Gen	1 产生开始信号并发送地址到 i2c 总线上 传送完成后清零。

10.5 12-bit ADC

10.5.1 概述

本芯片内部集成了一个 12 位高精度，高转换速率的逐次逼近型模数转换器(SAR ADC)模块。具有以下特性：

- 12 位转换精度；
- 高达 200K SPS 的转换速度；
- 支持 18 路可选的单端输入通道
- 支持 3 路可选的参考电压源：片外参考，电源参考，片内的 1.2V/2V 参考；
- ADC 的电压输入范围：0~Vref；
- 软件可配置 ADC 的采样/转换时钟频率；
- 软件可配置 ADC 的采样时间；
- 可以通过定时器事件触发采样。

10.5.2 结构框图

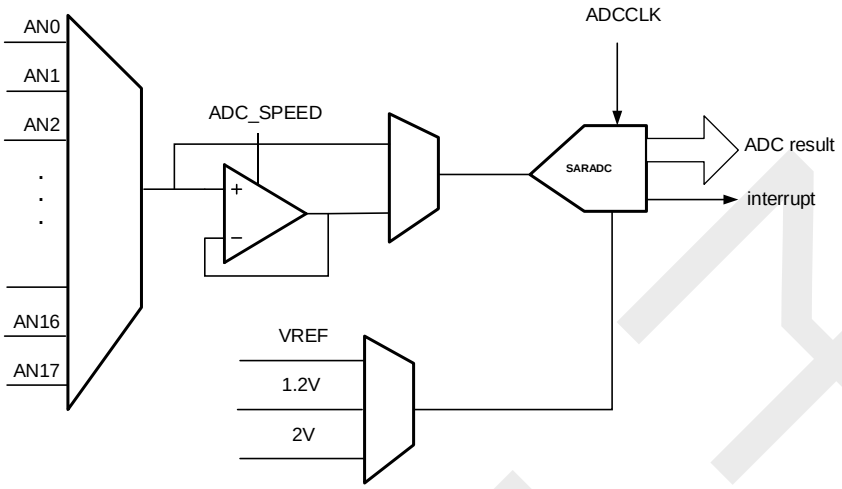


图 10-3 ADC 结构框图

10.5.3 ADC 转换时序

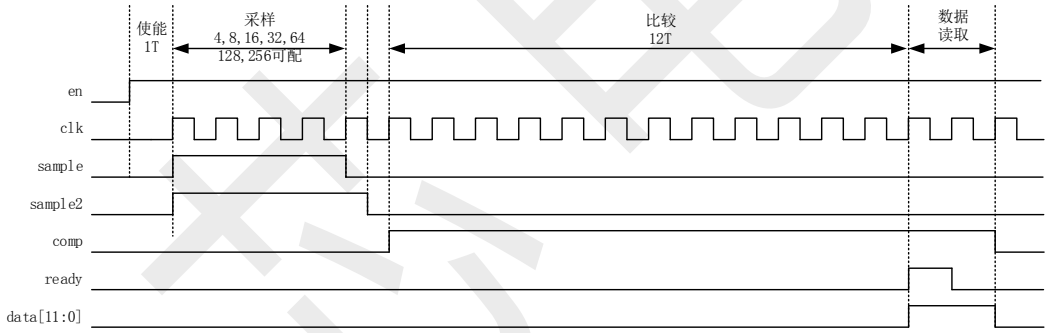


图 10-4 ADC 时序图

10.5.4 与 ADC 相关寄存器定义

名字	地址	读写	复位值	描述
ADC_CR0	0xE8	读写	00000000	ADC 转换控制寄存器 0
ADC_CR1	0xE9	读写	00000000	ADC 转换控制寄存器 1
ADC_CR2	0xEA	读写	00000010	ADC 转换控制寄存器 2
ADC_RESL	0xEB	读写	00000000	ADC 转换结果低位寄存器
ADC_RESH	0xEC	读写	00000000	ADC 转换结果高位寄存器

10.5.4.1 ADC_CR0 (0xE8)

Bit	7	6	5	4	3	2	1	0
Name	ADC_EN	ADC_START	ADC_IF	ADC_IE	-	CLKSEL[2:0]		
Reset	0	0	0	0	-	0	0	0

Type	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W

Bit	Name	Function
7	ADC_EN	ADC 使能位 0 = ADC 转换电路关闭 1 = ADC 转换电路开启
6	ADC_START	ADC 转换启动控制位。写 1 后开始 ADC 转换，转换完成后硬件自动将此位清零。 0 无影响。即使 ADC 已经开始转换工作，写 0 也不会停止 A/D 转换。 1 开始 ADC 转换，转换完成后硬件自动将此位清零。
5	ADC_IF	ADC 转换结束标志。当 ADC 完成一次转换后，硬件会自动将此位置 1，并向 CPU 发出中断请求。此标志位必须由软件写 1 清零。
4	ADC_IE	ADC 中断使能。 0 = 关闭 ADC 中断 1 = 使能 ADC 中断
3	N/A	保留位，读 0
2:0	CLKSEL[2:0]	ADC 时钟选择： 001 = 系统时钟的 2 分频 010 = 系统时钟的 4 分频 011 = 系统时钟的 6 分频 100 = 系统时钟的 8 分频 101 = 系统时钟的 16 分频 其它值 系统时钟

10.5.4.2 ADC_CR1 (0xE8)

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	TRIGSEL[1:0]		SCSEL[2:0]		
Reset	-	-	-	0	0	0	0	1
Type	-	-	-	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:5	N/A	保留位，读 0
4:3	TRIGSEL[1:0]	硬件触发模式选择 00 无硬件触发 01 选择 Timer1 中断触发 10 选择 Timer2 中断触发 11 无硬件触发
2:0	SCSEL[2:0]	ADC 采样时间周期选择寄存器： 000 = 4 个 ADC 时钟周期 001 = 8 个 ADC 时钟周期 010 = 16 个 ADC 时钟周期 011 = 32 个 ADC 时钟周期 100 = 64 个 ADC 时钟周期 101 = 128 个 ADC 时钟周期 110 = 256 个 ADC 时钟周期

		111 = 256 个 ADC 时钟周期 片外很高的输入阻抗时，增加采样时间，提高转换精度。

10.5.4.3 ADC_CR2 (0xEA)

Bit	7	6	5	4	3	2	1	0
Name	-	-	CTRL[5:0]					
Reset	-	-	0	0	0	0	1	1
Type	-	-	R/W					

Bit	Name	Function
7:6	N/A	保留位，读 0
5:0	CTRL[5:0]	ADC 控制寄存器： [5] 参考测试模式 0 正常模式； 1 测试模式 [4] 高速模式选择 0 正常模式； 1 高速模式 [3] 参考 buffer 输出选择 0 输出 buffer*1.666，如果输入 1.2V 输出 2.0V，默认； 1 输出 buffer*1，输入 1.2V 输出 1.2V [2] 参考 buffer 输入选择 0 选择内部参考 Bandgap； 1 选择外部输入 [1:0] 00 选择片外参考； 01 保留； 10 选择 VDD 参考； 11 选择 buffer 输出参考。

10.5.4.4 ADC_RESL (0xEB)

Bit	7	6	5	4	3	2	1	0
Name	ADC_RES[7:0]							
Reset	0x00							
Type	R							

Bit	Name	Function
7:0	ADC_RES[7:0]	ADC 转换结果低 8 位

10.5.4.5 ADC_RESH (0xEC)

Bit	7	6	5	4	3	2	1	0
Name	-				ADC_RES[11:8]			
Reset	-				0	0	0	0
Type	-				R	R	R	R

Bit	Name	Function
7:4	N/A	保留位，读 0
3:0	ADC_RES[11:8]	ADC 转换结果高 4 位

10.6 模拟比较器

10.6.1 概述

模拟电压比较器 VC 用于比较两个输入模拟电压的大小，并根据比较结果输出高/低电平。当“+”输入端电压高于“-”输入端电压时，电压比较器输出为高电平；当“+”输入端电压低于“-”输入端电压时，电压比较器输出为低电平。RC6F800X 内部集成的模拟电压比较器 VC 具有以下特性：

- 支持电压比较功能；
- 支持内部 64 阶 VBG 分压作为输入；
- 支持外部输入端口；
- 支持三种软件可配置的中断触发方式：上升沿触发/下降沿触发/上下沿触发；
- 电压比较器的输出可以作为 Timer1 和 Timer2 的刹车输入；
- 提供软件可配置的滤波时间以增强芯片的抗干扰能力。

10.6.2 结构框图

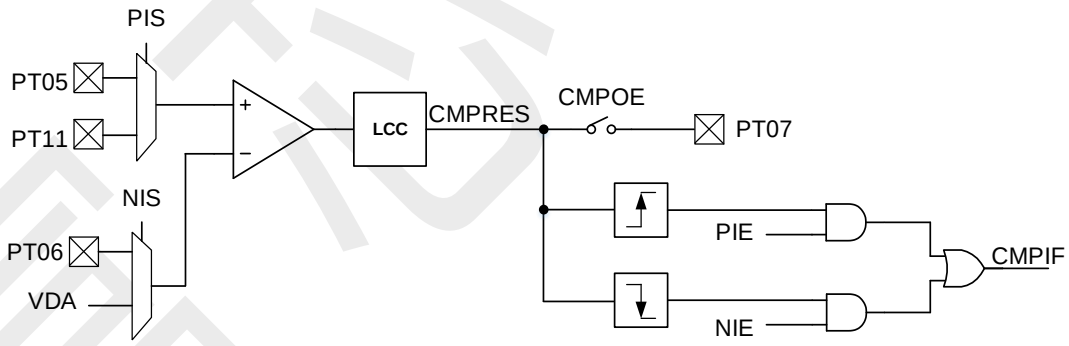


图 10-17 比较器 0 电路结构框图

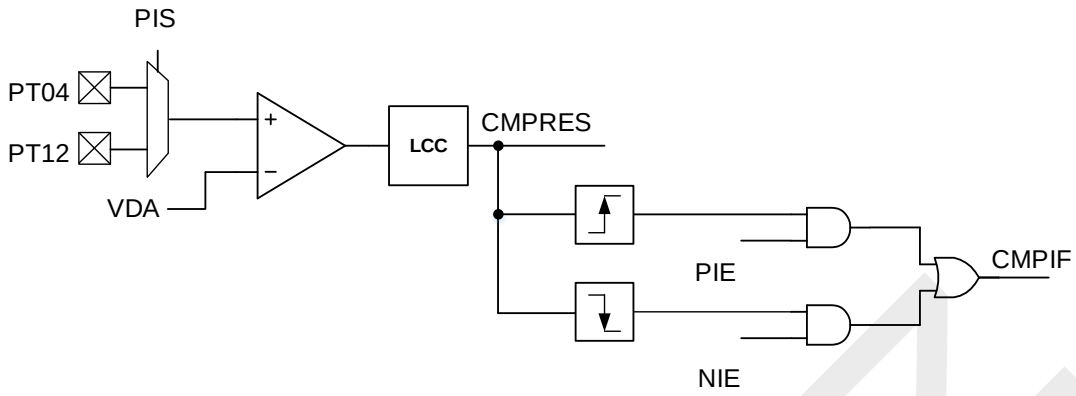


图 10-18 比较器 1 结构框图

10.6.3 比较器时钟和滤波功能

模拟比较器输出通过 SCKO 同步到内部数字系统，数字滤波时钟采用 SCKO 作为输入，如果要使用比较器要打开 SCKO。

数字滤波工作即为数字信号去抖动功能。当比较器输出变化 LCDTY 个时钟周期稳定后才认为数据变化是有效的。数字滤波功时钟来自 SCKO，所以如果要使用数字滤波功能，要使能 SCKO 时钟。

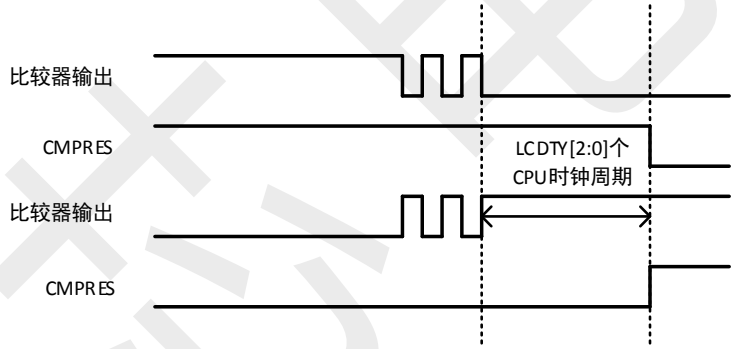


图 10-5 比较器数字滤波功能

注意：模拟比较器的输出到数字滤波电路之间有两个 SCKO 时钟周期延时。

10.6.4 与模拟比较器相关寄存器定义

名字	地址	读写	复位值	描述
ACO_CR1	0xD8	读写	00000000	模拟比较器 0 控制寄存器 0
ACO_CR2	0xD9	读写	00000000	模拟比较器 0 控制寄存器 1
ACO_DASEL	0xDA	读写	00000000	模拟比较器 0 参考电压选择寄存器
AC1_CR1	0xDB	读写	00000000	模拟比较器 1 控制寄存器 0
AC1_CR2	0xDC	读写	00000000	模拟比较器 1 控制寄存器 1
AC1_DASEL	0xDD	读写	00000000	模拟比较器 1 参考电压选择寄存器

10.6.4.1 ACO_CR1 (0xD8)

Bit	7	6	5	4	3	2	1	0
Name	CMPEN	CMPIF	PIE	NIE	PIS	NIS	CMPOE	CMPRES
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R

Bit	Name	Function
7	CMPEN	0 关闭比较功能 1 使能比较功能
6	CMPIF	比较器中断标志位。当 PIE 或者 NIE 被使能后，若产生相应的中断信号，硬件自动将 CMPIF 置 1，并向 CPU 提出中断请求。此标志位须由用户软件写 0 清零。 备注：如果没有使能比较器中断时，硬件不会设置此中断标志，即使使用查询方式访问比较器时，不能查询此中断标志
5	PIE	比较器上升沿中断使能位 0 禁止比较器上升沿中断 1 使能比较器上升沿中断。
4	NIE	比较器下降沿中断使能位 0 禁止比较器下降沿中断 1 使能比较器下降沿中断
3	PIS	比较器的正极选择位 0 选择外部端口 PT05 作为比较器正极输入 1 选择外部端口 PT11 作为比较器正极输入
2	NIS	比较器的负极选择位 0 选择外部端口 PT06 作为比较器负极的外部输入 1 选择内部 DA 的输出作为比较器负极的输入
1	CMPOE	比较器结果输出控制位 0 禁止比较器结果输出 1 使能比较器结果输出
0	CMPRES	比较器的比较结果，此位为只读位 0 表示 CMP+的电平低于 CMP-的电平 1 表示 CMP+的电平高于 CMP-的电平 CMPRES 是数字滤波后的输出信号，而不是比较器的直接输出结果。

10.6.4.2 ACO_CR2 (0xD9)

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	INVCMPO	LCDTY[2:0]		
Reset	-	-	-	-	0	0	0	0
Type	-	-	-	-	R/W	R/W	R/W	R/W

Bit	Name	Function
7;4	N/A	保留位，读 0
3	INVCMPO	0 比较器结果如果为 0 输出低电平，反之输出高电平 1 比较器结果如果为 0 输出高电平，反之输出低电平
2:0	LCDTY[2:0]	数字滤波功能。当比较器结果发生上升沿或者下降沿变化时，比较器侦测变化后的

		信号必须维持 LCDTY 所设置的 CPU 时钟数不发生变化，才认为数据变化是有效的； 否则若 LCDTY 设置为 0 时表示关闭数字滤波功能。

10.6.4.3 ACO_DASEL (0xDA)

Bit	7	6	5	4	3	2	1	0
Name	-	DAEN	DASEL[5:0]					
Reset	-	0	0	0	0	0	0	0
Type	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	N/A	保留位，读 0
6	DAEN	DA 使能： 0 = 关闭 DA 1 = 打开 DA
5:0	DASEL[5:0]	选择 DA 输出电压，电压计算公式，单位（V）： Vda=1.2*DASEL/63

10.6.4.4 AC1_CR1 (0xDB)

Bit	7	6	5	4	3	2	1	0
Name	CMPEN	CMPIF	PIE	NIE	PIS	-	-	CMPRES
Reset	0	0	0	0	0	-	-	0
Type	R/W	R/W	R/W	R/W	R/W	-	-	R

Bit	Name	Function
7	CMPEN	0 关闭比较功能 1 使能比较功能
6	CMPIF	比较器中断标志位。当 PIE 或者 NIE 被使能后，若产生相应的中断信号，硬件自动将 CMPIF 置 1，并向 CPU 提出中断请求。次标志位必须用户软件清零。（备注：如果没有使能比较器中断时，硬件不会设置此中断标志，即使使用查询方式访问比较器时，不能查询此中断标志）
5	PIE	比较器上升沿中断使能位： 0 禁止比较器上升沿中断 1 使能比较器上升沿中断。
4	NIE	比较器下降沿中断使能位： 0 禁止比较器下降沿中断 1 使能比较器下降沿中断
3	PIS	比较器的正极选择位 0 选择外部端口 PT04 作为比较器正极输入 1 选择外部端口 PT12 作为比较器负极输入
2:1	N/A	保留位，读 0
0	CMPRES	比较器的比较结果，此位为只读位 0 表示 CMP+的电平低于 CMP-的电平 1 表示 CMP+的电平高于 CMP-的电平 CMPRES 是数字滤波后的输出信号，而不是比较器的直接输出结果。

10.6.4.5 AC1_CR2 (0xDC)

Bit	7	6	5	4	3	2	1	0
Name	-					LCDTY[2:0]		
Reset	0x00							
Type	R/W							

Bit	Name	Function
7:3	N/A	保留位，读 0
2:0	LCDTY[2:0]	数字滤波功能。当比较器结果发生上升沿或者下降沿变化时，比较器侦测变化后的信号必须维持 LCDTY 所设置的 CPU 时钟数不发生变化，才认为数据变化是有效的；否则若 LCDTY 设置为 0 时表示关闭数字滤波功能。

10.6.4.6 AC1_DASEL (0xDD)

Bit	7	6	5	4	3	2	1	0
Name	-	-	DASEL[5:0]					
Reset	-	-	0	0	0	0	0	0
Type	-	-	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:6	N/A	保留位，读 0
5:0	DASEL[5:0]	选择 DA 输出电压，电压计算公式，单位（V）： Vda=1.2*DASEL/63

11 省电模式和看门狗

11.1 省电模式

本芯片有两种低功耗模式来优化设备功耗：

- 睡眠模式
- 深度睡眠模式

这两种模式下程序都停止运行。

外设	睡眠模式	深度睡眠模式
CPU	停止	停止
RAM	保持	保持
睡眠定时器	运行	运行
看门狗	运行	运行
定时器 0~2	运行	停止
ADC	运行	停止
比较器	运行	停止
UART	运行	停止
I2C	运行	停止
内部 16MHz 振荡器	运行	停止
内部 32KHz 振荡器	运行	运行
I/O 口	保持	保持
其他外设	运行	停止
唤醒条件	引脚复位、看门狗复位，所有中断	引脚复位，看门狗复位，引脚中断唤醒，睡眠定时器中断

11.1.1 睡眠模式

写 SCR 寄存器 SLEEP=1 且 SLEEPDEEP=0 进入到睡眠模式。该模式下，内部 16MHz 晶振保持工作。同时继续给外设提供时钟，但是 CPU 时钟停止。该模式可以通过复位和中断唤醒。如果使用复位唤醒，那么整个系统会复位而初始化。

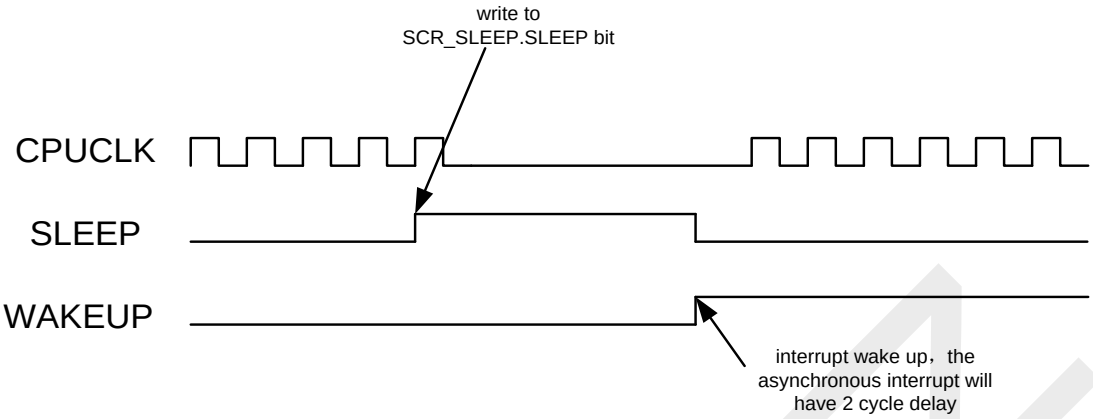


图 11-1 睡眠模式休眠和唤醒时序图

11.1.2 深度睡眠模式

深度睡眠模式通过写 SCR 的 SLEEP=1 且 SLEEPDEEP=1 进入。该模式下，16MHz 主振荡器停止工作，32KHz 低功耗振荡器继续工作。系统时钟和外设时钟停止，但是睡眠定时器和看门狗继续工作。

11.1.3 深度休眠模式唤醒

深度睡眠模式可以通过复位和中断唤醒。复位重新初始化所有的控制寄存器，所以重新工作。振荡器的重新工作需要一定时间的延时。下面的图描述了深度休眠唤醒的时序。

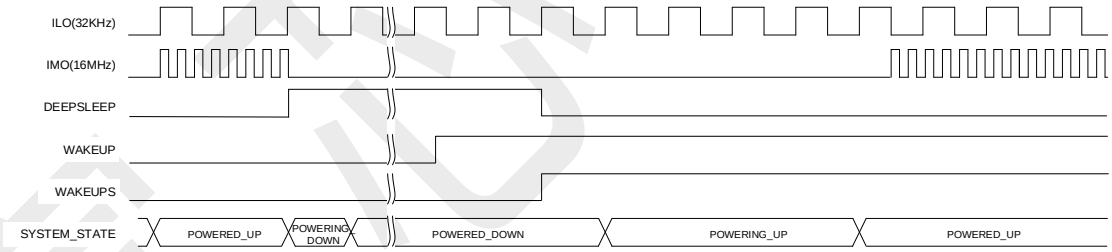


图 11-2 深度休眠唤醒时序

11.2 看门狗

看门狗定时器由 16 位睡眠定时器和 2 位看门狗定时器组成，如果看门狗使能且计数到 3 并溢出的话那么会触发看门狗复位。看门狗复位如果被触发会保持至少 2 个 32K 时钟周期。当 WDG 计数器在两个周期没有清零的话会触发 WDR。看门狗计数器可以通过写一个特殊寄存器 WDCLR 来清零。睡眠计数器也可以通过写 WDCLR 来清零。

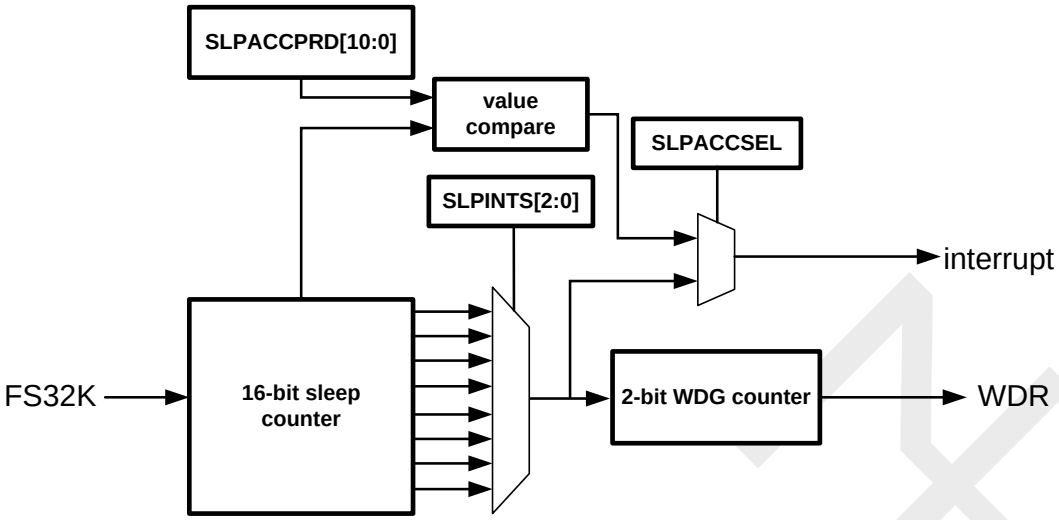


图 11-3 看门狗定时器

11.3 睡眠定时器中断

看门狗内部实现了一个 16 位的睡眠向上计数器，该定时器用作看门狗的预分频同时也可以作用定时功能。可以产生中断，中断使能可以控制。该定时器有两种用法，第一种 SLPACCSEL 等于 0 时，通过 SLPINTS 选择固定的定时周期，第二种 SLPACCSEL 等于 1 时，选择对于的溢出值来产生中，溢出值通过 SLPACCPRD[10：0]来配置。

SLPINTS[2：0]	16-bit 计数器位	定时时间(单位毫秒)
3’b000	6	4
3’b001	7	8
3’b010	8	16
3’b011	9	32
3’b100	12	253
3’b101	13	506
3’b110	14	1012
3’b111	15	2024

11.4 与省电模式和看门狗相关寄存器定义

名字	地址	读写	复位值	描述
SLPTIM_CR	0x88	读写	00000000	睡眠计数器控制寄存器
SLPTIM_STAT	0x89	读写	00000000	睡眠计数状态
SLPTIM_CLR	0x8A	写	00000000	看门狗清除寄存器
SLPTIM_WDT	0x8B	读写	00000000	看门狗计数器状态
SLPTIM_CNTL	0x8C	只读	00000000	睡眠计数器计数值
SLPTIM_CNTH	0x8D	只读	00000000	睡眠计数器计数值
SLPTIM_PRDL	0x8E	读写	00000000	睡眠计数器预分频

SLPTIM_PRDH	0x8F	读写	00000000	睡眠计数器预分频

11.4.1 SLPTIM_CR (0x88)

Bit	7	6	5	4	3	2	1	0
Name	SLPIE	-	WDTEN	-	SLEEPDIS	SLPINTS[2:0]		
Reset	0	-	0	-	0	0	0	0
Type	R/W	-	R/W	-	R/W	R/W	R/W	R/W

Bit	Name	Function
7	SLPIE	0 = 睡眠定时器中断禁止 1 = 睡眠定时器中断使能
6	N/A	保留位，读 0
5	WDTEN	0 = 看门狗定时器禁止 1 = 看门狗定时器使能
4	N/A	保留位，读 0
3	SLEEPDIS	0 = 使能睡眠定时器 1 = 禁止睡眠定时器
2:0	SLPINTS[2:0]	睡眠定时器溢出时间： 000 4ms 001 8ms 010 16ms 011 31.2ms 100 248ms 101 499.2ms 110 1s 111 2s

11.4.2 SLPTIM_STAT (0x89)

Bit	7	6	5	4	3	2	1	0
Name	SLPEV	-	-	-	-	-	-	-
Reset	0	-	-	-	-	-	-	-
Type	R/W	-	-	-	-	-	-	-

Bit	Name	Function
7	SLPEV	0 = 睡眠计数器没有溢出 1 = 睡眠计数器溢出 写 0 清除该位。
6:0	N/A	保留位，读 0

11.4.3 SLPTIM_CLR (0x8A)

Bit	7	6	5	4	3	2	1	0
Name	SLPTIM_CLR							
Reset	-	-	-	-	-	-	-	-

Type	W	W	W	W	W	W	W	W

Bit	Name	Function
7:0	SLPTIM_CLR	写任何值到该寄存器清除看门狗。

11.4.4 SLPTIM_CNTRL 和 SLPTIM_CNTRH (0x8C/0x8D)

Bit	7	6	5	4	3	2	1	0
Name	CNTRL							
Reset	0	0	0	0	0	0	0	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	Name	Function
7:0	CNTRL	看门狗计数器计数值低 8 位。

Bit	7	6	5	4	3	2	1	0
Name	CNTRH							
Reset	0	0	0	0	0	0	0	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	Name	Function
7:0	CNTRH	看门狗计数器计数值高 8 位。

11.4.5 SLPTIM_WDT (0x8B)

Bit	7	6	5	4	3	2	1	0
Name	-	WDTOV	WDTCNTR		-	-	-	-
Reset	-	0	0	0	-	-	-	-
Type	-	RO	R/W	R/W	-	-	-	-

Bit	Name	Function
7	N/A	保留位，读 0
6	WDTOV	看门狗溢出标志： 0 看门狗没有溢出 1 看门狗溢出
5:4	WDTCNTR	2 bit 看门狗计数器计数值，只能通过写 0 清除。
3:0	N/A	保留位，读 0

11.4.6 SLPTIM_PRDRL (0x8E)

Bit	7	6	5	4	3	2	1	0
Name	ACCPRDRL							
Reset	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	ACCPRDRL	睡眠定时器溢出值低 8 位

11.4.7 SLPTIM_PRDRH (0x8F)

Bit	7	6	5	4	3	2	1	0
Name	ACCSEL	–	–	–	–	ACCPRDRH		
Reset	0	–	–	–	–	0	0	0
Type	R/W	–	–	–	–	R/W	R/W	R/W

Bit	Name	Function
7	ACCSEL	0 = 选择睡眠定时器溢出值为固定值 1 = 选择睡眠定时器溢出值为 11 位可配置
6:3	N/A	保留位，读 0
2:0	ACCPRDRH	睡眠定时器高 3 位

12 系统控制

12.1 系统控制寄存器

名字	地址	读写	复位值	描述
SCR_CFG	0x90	读写	10000011	系统配置寄存器
SCR_SLEEP	0x91	读写	00000000	休眠寄存器
SCR_CALI	0x92	读写	00000000	校准寄存器

12.1.1 SCR_CFG (0x90)

Bit	7	6	5	4	3	2	1	0
Name	RAMSLOW	-	PT00RST	-	AWKEN	RSTREQ	-	-
Reset	1	-	0	-	0	0	1	1
Type	R	-	R/W	-	R/W	W	R	R

Bit	Name	Function
7	RAMSLOW	SRAM 控制寄存器： 0 SRAM 快速模式 1 SRAM 慢速模式
6	N/A	保留位，读 0
5	PT00RST	PT00 脚复用功能控制： 0 PT00 用作普通 GPIO 1 PT00 用作复位脚
4	N/A	保留位，读 0
3	AWKEN	唤醒模式控制位： 0 禁止休眠模式唤醒 1 使能休眠模式唤醒
2	RSTREQ	软件复位使能： 0 不复位系统 1 复位系统
1	N/A	保留位，读 1
0	N/A	保留位，读 1

12.1.2 SCR_SLEEP (0x91)

Bit	7	6	5	4	3	2	1	0
Name	-						SLEEPDEEP	SLEEP
Reset	-						0	0
Type	-						R/W	R/W

Bit	Name	Function
7:2	N/A	保留位，读 0

1	SLEEPDEEP	深度休眠模式控制： 0 深度休眠模式关闭 1 深度休眠模式打开
0	SLEEP	休眠模式控制： 0 正常工作模式 1 休眠模式

12.1.3 SCR_CALI (0x92)

Bit	7	6	5	4	3	2	1	0
Name	CALI_WDR	CALI_XRES	-					
Reset	0	1	-					
Type	R/W1C	R/W1C	-					

Bit	Name	Function
7	CALI_WDR	看门狗复位标志： 0 没有看门狗复位（该寄存器的清零可以通过外部引脚复位、POR、BOR、写 1 来实现） 1 看门狗复位发生 写 1 清清零 CALI_XRES, CALI_WDR
6	CALI_XRES	引脚复位标志： 0 没有外部引脚复位发生（该寄存器的清零可以通过看门狗复位、~~POR~~、~~BOR~~、CALI_WDR 写 1 来实现） 1 外部引脚复位发生
5:0	N/A	保留位，读 0

12.2 模拟控制寄存器

名字	地址	读写	复位值	描述
BG_VTRIM	0xFF81	读写	01000000	Bandgap 电压修调寄存器
BORLVD_CR	0xFF85	读写	00010101	BORLVD 控制寄存器
BORLVD_STAT	0xFF86	读写	00000000	BORLVD 状态寄存器
IMO_CR	0xFF88	读写	00000000	IMO 控制寄存器
IMO_FTRIM	0xFF89	读写	10000000	IMO 快速修调寄存器
ILO_TRIM	0xFF8A	读写	00101111	ILO 修调寄存器
IMO_STRIM	0xFF8C	读写	01000000	IMO 慢速修调寄存器
XTAL_CR	0xFF8D	读写	00000001	XTAL 控制寄存器

12.2.1 BG_VTRIM (0xFF81)

Bit	7	6	5	4	3	2	1	0

Name	-	BG_VTRIM[6:0]						
Reset	-	1	0	0	0	0	0	0
Type	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	N/A	保留位，读 0
6:0	BG_VTRIM [6:0]	Bandgap 电压修调值。注意：BG 偏离值±50mV。

12.2.2 BORLVD_CR (0xFF85)

Bit	7	6	5	4	3	2	1	0
Name	-	BOR_VSEL[1:0]		BOR_EN	-	LVD_VSEL[1:0]		LVD_EN
Reset	-	0	0	1	-	0	0	1
Type	-	R/W	R/W	R/W	-	R/W	R/W	R/W

Bit	Name	Function
7	N/A	保留位，读 0
6:5	BOR_VSEL[1:0]	BOR 电压点选择： 00 2.25V(默认值) 01 2.25V 10 2.6V 11 4.2V
4	BOR_EN	BOR 控制位 0 关闭 BOR 1 使能 BOR
3	N/A	保留位，读 0
2:1	LVD_VSEL[1:0]	LVD 电压点选择： 00 2.3V（默认值） 01 2.5V 10 2.7V 11 4.3V
0	LVD_EN	LVD 控制位 0 关闭 LVD 1 使能 LVD

12.2.3 BORLVD_STAT (0xFF86)

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	IE_LVD	STAT_BOR	-	-	STAT_LVD
Reset	-	-	-	0	0	-	-	0
Type	-	-	-	R/W	R/W	-	-	R/W

Bit	Name	Function
7:5	N/A	保留位，读 0

4	BOR_VSEL	LVD 中断使能位 0 禁止 LVD 中断 1 使能 LVD 中断
3	STAT_BOR	BOR 输出状态 0 BOR 没有发生 1 BOR 发生
2:1	N/A	保留位，读 0
0	STAT_LVD	LVD 输出状态 0 没有 LVD 事件发生 1 检测到 LVD 事件 该标志位只能读，不能清除。

12.2.4 IMOCR (0xFF88)

Bit	7	6	5	4	3	2	1	0
Name	EXT_SEL	FX2_SEL	-	-	IMO_TSTEN	-	SLOW	IMO_EN
Reset	0	0	-	-	0	-	0	0
Type	R/W	R/W	-	-	R/W	-	R/W	R/W

Bit	Name	Function
7:6	[EXT_SEL:FX2_SEL]	系统时钟源选择 0x 选择内部 8MHz 时钟 10 选择 PT17 做系统时钟 11 选择内部 16MHz 时钟
5:4	N/A	保留位，读 0
3	IMO_TETEN	0 IMO 测试功能关闭 1 IMO 测试功能打开，选择 SCK1 到 PT10 口。 注意：要把 PT0 的 GPIO 复用输出功能打开
2	N/A	保留位，读 0
1	SLOW	0 选择快速时钟修调值 1 选择慢速时钟修调值
0	IMO_EN	写模式下 0 使能 IMO 1 关闭 IMO 读模式下 0 IMO 关闭 1 IMO 使能

12.2.5 IMO_FTRIM (0xFF89)

Bit	7	6	5	4	3	2	1	0
Name	IMO_FTRIM							
Reset	0x80							
Type	R/W							

Bit	Name	Function
7:0	IMO_FTRIM	IMO 快速模式修调值

12.2.6 ILO_TRIM (0xFF8A)

Bit	7	6	5	4	3	2	1	0
Name	ITRIM			RTRIM				
Reset	0x2F							
Type	R/W							

Bit	Name	Function
7:5	ITRIM	ILO 电流修调值
4:0	RTRIM	ILO 电阻修调值

12.2.7 IMO_STRIM (0xFF8C)

Bit	7	6	5	4	3	2	1	0
Name	STRIM							
Reset	0x40							
Type	R/W							

Bit	Name	Function
7:0	STRIM	IMO 慢速修调值

12.2.8 XTAL_CR (0xFF8D)

Bit	7	6	5	4	3	2	1	0
Name	XTAL_EN	-	-	-	-	-	XTAL_OPT	
Reset	0	-	-	-	-	-	0	1
Type	R/W	-	-	-	-	-	R/W	R/W

Bit	Name	Function
7	XTAL_EN	0 禁止外部 RTC 振荡器 1 使能外部 RTC 振荡器
6:2	N/A	保留位，读 0
1:0	XTAL_OPT	XTAL 选项 可以控制 XTAL 的起振，一般配置 00 或者 01

13 电气特性

13.1 绝对最大额定值

参数	最小值	最大值	单位
存储器温度	-55	125	℃
工作温度	-40	85	℃
工作电压	2.4	5.5	V
VDD 对地电压	-0.3	6.6	V
IO 对地电压	-0.3	VDD+0.3	V

13.2 直流特性

符号	参数	测试条件	最小值	典型值	最大值	单位
		VDD=5V，常温 25℃				
f_{FLASH}	FLASH 工作频率	4.5≤VDD＜5.5			16	MHz
		2.4V≤VDD＜5.5V			4	MHz
IDD1	工作电流 1	内部 16MHz RC 振荡器工作，CPU 工作在 16MHz		6.5		mA
IDD2	工作电流 2	内部 16MHzRC 振荡器工作，CPU 关闭		1.55		mA
ISP	静态电流	内部 16MHzRC 振荡器关闭，32KHz 时钟打开，CPU 工作在 DEEPSLEEP 模式		3.2	12	uA
VIL	输入低电平				0.3VDD	
VIH	输入高电平		0.5VDD			
R_{PU}	上拉电阻			10		KΩ
R_{PD}	下拉电阻			10		KΩ
I_{OH1}	拉电流 1	PT01~PT07 和 PT10~PT17 输出 4V		25		mA
I_{OL1}	灌电流 1	PT01~PT07 和 PT10~PT17 输出 0.3V		30		mA
I_{OH2}	拉电流 2	PT00 输出 4V		10		mA
I_{OL2}	灌电流 2	PT00 输出 0.3V		8		mA
I_{OH3}	拉电流 3	PT20 和 PT21 输出 4V		65		mA
I_{OL3}	灌电流 3	PT20 和 PT21 输出 0.3V		100		mA

13.3 ADC 特性

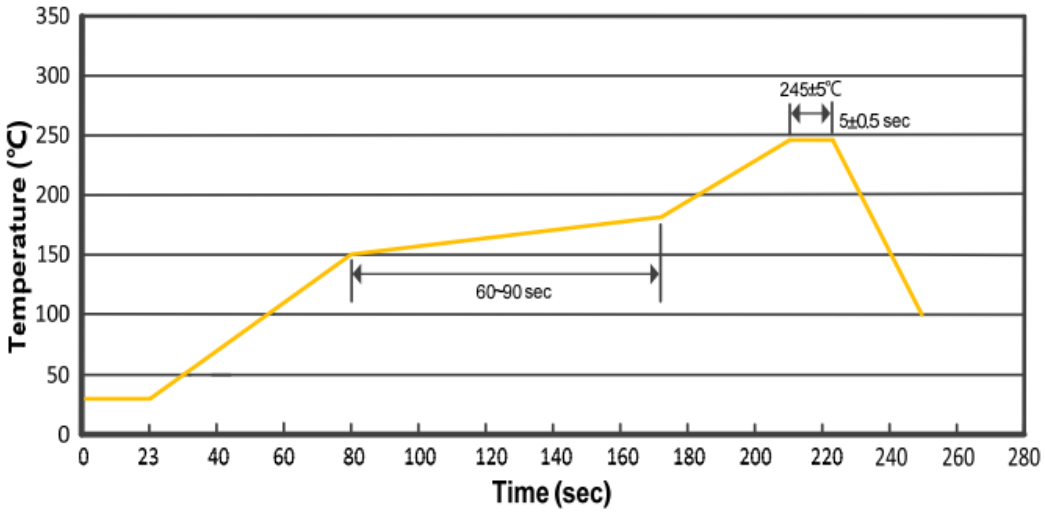
符号	参数	条件	最小值	典型值	最大值	单位
	分辨率	单端转换		12		比特
	积分非线性误差			±2	±6	LSB
	微分非线性误差			±1	±2	LSB
	增益误差			±1	±2	LSB
	偏移误差			±2	±6	LSB
	转换速度			200	300	ksps
V_{DD}	模拟供电电压		2.4	5	5.5	V
V_{REF}	参考电压		0		V_{CC}	V
V_{IN}	输入电压		0		V_{REF}	V
V_{INT}	内部电压参考			1.2		V

13.4 EMC 特性

Electrostatic discharge (ESD)

符号	参数	测试条件	分类	最大值	单位
$V_{ESD(HBM)}$	Electrostatic discharge voltage (Human body model)			3500	V
$V_{ESD(CDM)}$	Electrostatic discharge voltage (Charge device model)			500	V

13.5 回流焊温度曲线

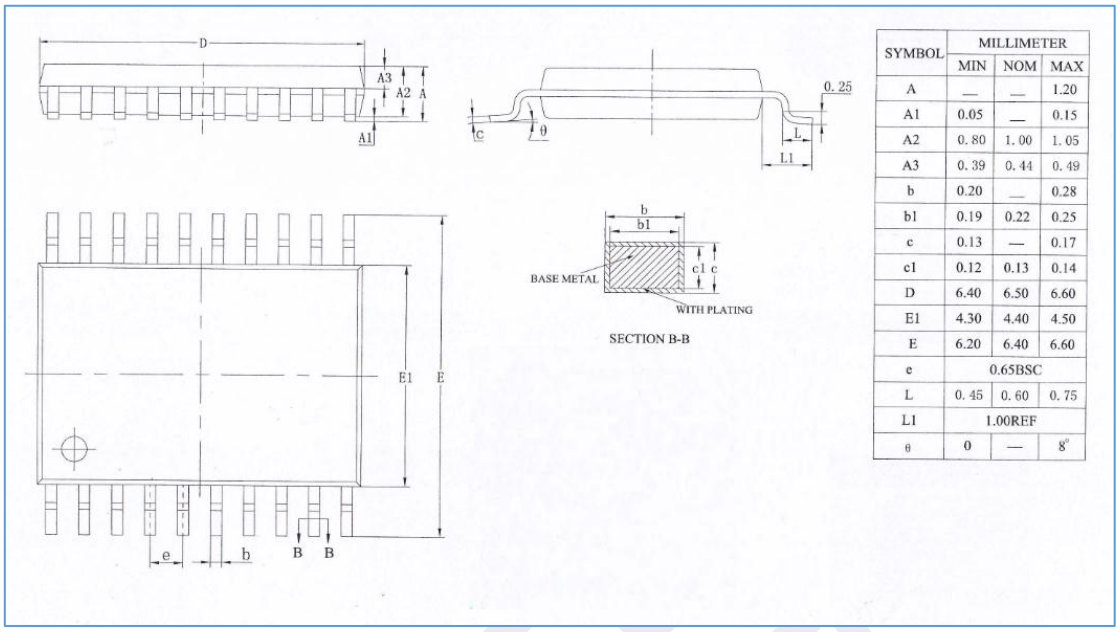


14 订单信息

订单型号	封装	丝印标记	包装
RC6F8001EC	TSSOP20	001	管装
RC6F8001EB	QFN20	001	编带卷装
RC6F8003DA	SOP16	003	管装

图 6 订单信息图

15 封装尺寸



SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	1.20
A1	0.05	—	0.15
A2	0.80	1.00	1.05
A3	0.39	0.44	0.49
b	0.20	—	0.28
b1	0.19	0.22	0.25
c	0.13	—	0.17
c1	0.12	0.13	0.14
D	6.40	6.50	6.60
E1	4.30	4.40	4.50
E	6.20	6.40	6.60
e	0.65BSC		
L	0.45	0.60	0.75
L1	1.00REF		
θ	0	—	8°

图 15-1 TSSOP20 封装外形图

标注 \ 尺寸	最小	标准	最大	标注 \ 尺寸	最小	标准	最大
A	0.70	0.75	0.80	D1	2.20	2.30	2.40
A1	0.00	—	0.05	E1	2.20	2.30	2.40
A3	0.203REF			e	0.50TYP		
b	0.20	0.25	0.30	K	0.20	—	—
D	3.90	4.00	4.10	L	0.30	0.40	0.50
E	3.90	4.00	4.10				

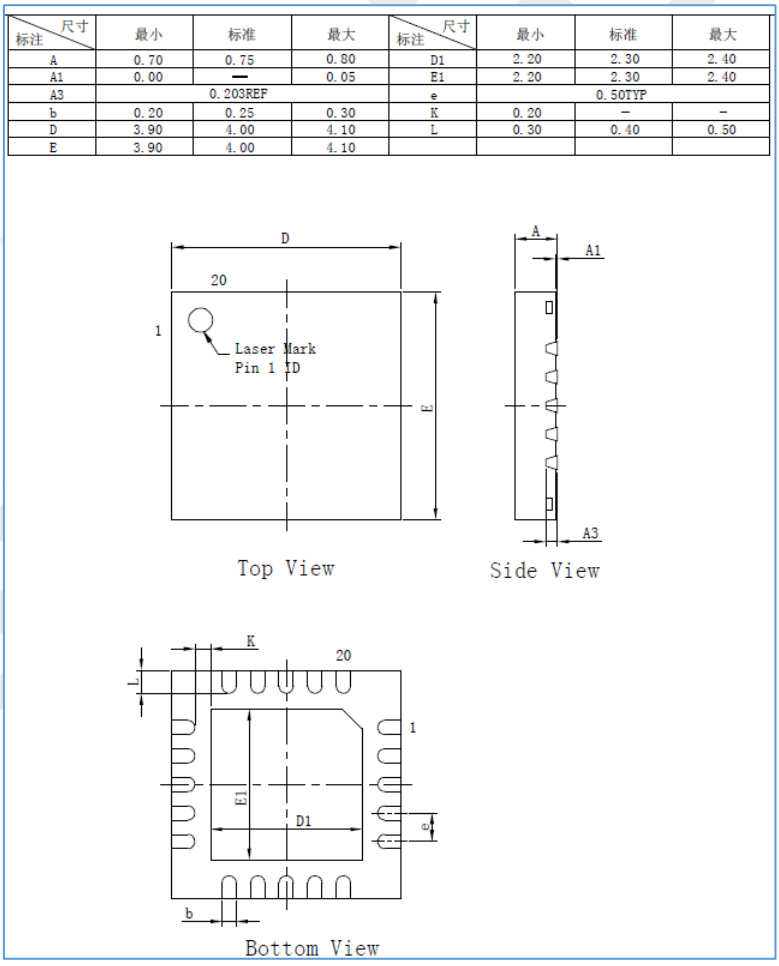
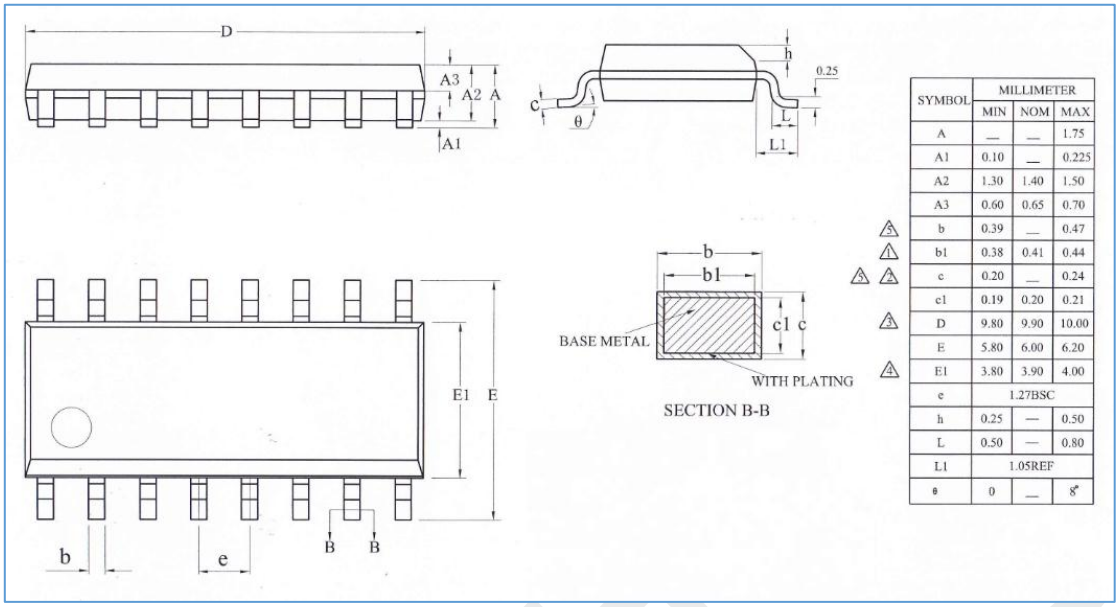


图 15-2 QFN20 封装外形图



SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	1.75
A1	0.10	—	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.39	—	0.47
b1	0.38	0.41	0.44
c	0.20	—	0.24
c1	0.19	0.20	0.21
D	9.80	9.90	10.00
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	1.27BSC		
h	0.25	—	0.50
L	0.50	—	0.80
L1	1.05REF		
θ	0	—	8°

图 15-3 SOP16 封装外形图

16 附录

16.1 INSTRUCTIONS SET BRIEF

16.1.1 INSTRUCTION SET NOTES

The RC6F800X has five different addressing modes: immediate, direct, register, indirect and relative. In the immediate addressing mode the data is contained in the opcode. By direct addressing an eight bit address is a part of the opcode, by register addressing, a register is selected in the opcode for the operation. In the indirect addressing mode, a register is selected in the opcode to point to the address used by the operation. The relative addressing mode is used for jump instructions.

The following tables give a survey about the instruction set cycles of the RC6F800X microcontroller core. **One cycle is equal to one clock period**.

Table 1 and Table 2 contain notes for mnemonics used in Instruction set tables. Tables 3 - 7 show instruction hexadecimal codes, number of bytes and machine cycles that each instruction takes to execute.

Rn	Working register R0-R7
direc	128 internal RAM locations, any Special Function Registers
@Ri	Indirect internal or external RAM location addressed by register R0 or
#dat	8-bit constant included in instruction
#dat	16-bit constant included as bytes 2 and 3 of instruction
bit	256 software flags, any bit-addressable I/O pin, control or status bit
A	Accumulator

Table 1. Notes on data addressing modes

addr16	Destination address for LCALL and LJMP may be anywhere within the 64-Kbyte of program memory address space.
addr11	Destination address for ACALL and AJMP will be within the same 2-Kbyte page of program memory as the first byte of the following instruction.
rel	SJMP and all conditional jumps include an 8-bit offset byte. Range is +127/-128 bytes relative to the first byte of the following instruction

Table 2. Notes on program addressing modes

16.1.2 INSTRUCTION SET BRIEF - FUNCTIONAL ORDER

16.1.2.1 ARITHMETIC OPERATIONS

Mnemonic	Description	Code	Byte	Cycle
ADD A,Rn	Add register to accumulator	0x28-0x2F	1	1
ADD A,direct	Add direct byte to accumulator	0x25	2	2
ADD A,@Ri	Add indirect RAM to accumulator	0x26-0x27	1	2
ADD A,#data	Add immediate data to accumulator	0x24	2	2
ADDC A,Rn	Add register to accumulator with carry flag	0x38-0x3F	1	1
ADDC A,direct	Add direct byte to A with carry flag	0x35	2	2
ADDC A,@Ri	Add indirect RAM to A with carry flag	0x36-0x37	1	2
ADDC A,#data	Add immediate data to A with carry flag	0x34	2	2
SUBB A,Rn	Subtract register from A with borrow	0x98-0x9F	1	1
SUBB A,direct	Subtract direct byte from A with borrow	0x95	2	2
SUBB A,@Ri	Subtract indirect RAM from A with borrow	0x96-0x97	1	2
SUBB A,#data	Subtract immediate data from A with borrow	0x94	2	2
INC A	Increment accumulator	0x04	1	1
INC Rn	Increment register	0x08-0x0F	1	2
INC direct	Increment direct byte	0x05	2	3
INC @Ri	Increment indirect RAM	0x06-0x07	1	3
DEC A	Decrement accumulator	0x14	1	1
DEC Rn	Decrement register	0x18-0x1F	1	2
DEC direct	Decrement direct byte	0x15	1	3
DEC @Ri	Decrement indirect RAM	0x16-0x17	2	3
INC DPTR	Increment data pointer	0xA3	1	1
MUL A,B	Multiply A and B	0xA4	1	2
DIV A,B	Divide A by B	0x84	1	6
DA A	Decimal adjust accumulator	0xD4	1	3

Table 3. Arithmetic operations

16.1.2.2 LOGIC OPERATIONS

Mnemonic	Description	Code	Byte	Cycle
ANL A,Rn	AND register to accumulator	0x58-0x5F	1	1
ANL A,direct	AND direct byte to accumulator	0x55	2	2
ANL A,@Ri	AND indirect RAM to accumulator	0x56-0x57	1	2
ANL A,#data	AND immediate data to accumulator	0x54	2	2
ANL direct,A	AND accumulator to direct byte	0x52	2	3
ANL direct,#data	AND immediate data to direct byte	0x53	3	3
ORL A,Rn	OR register to accumulator	0x48-0x4F	1	1
ORL A,direct	OR direct byte to accumulator	0x45	2	2
ORL A,@Ri	OR indirect RAM to accumulator	0x46-0x47	1	2
ORL A,#data	OR immediate data to accumulator	0x44	2	2
ORL direct,A	OR accumulator to direct byte	0x42	2	3
ORL direct,#data	OR immediate data to direct byte	0x43	3	3
XRL A,Rn	Exclusive OR register to accumulator	0x68-0x6F	1	1
XRL A,direct	Exclusive OR direct byte to accumulator	0x65	2	2
XRL A,@Ri	Exclusive OR indirect RAM to accumulator	0x66-0x67	1	2
XRL A,#data	Exclusive OR immediate data to accumulator	0x64	2	2
XRL direct,A	Exclusive OR accumulator to direct byte	0x62	2	3
XRL direct,#data	Exclusive OR immediate data to direct byte	0x63	3	3
CLR A	Clear accumulator	0xE4	1	1
CPL A	Complement accumulator	0xF4	1	1
RL A	Rotate accumulator left	0x23	1	1
RLC A	Rotate accumulator left through carry	0x33	1	1
RR A	Rotate accumulator right	0x03	1	1
RRC A	Rotate accumulator right through carry	0x13	1	1
SWAP A	Swap nibbles within the accumulator	0xC4	1	1

Table 4. Logic operations

16.1.2.3 BOOLEAN MANIPULATION

Mnemonic	Description	Code	Byte	Cycle
CLR C	Clear carry flag	0xC3	1	1
CLR bit	Clear direct bit	0xC2	2	3
SETB C	Set carry flag	0xD3	1	1
SETB bit	Set direct bit	0xD2	2	3
CPL C	Complement carry flag	0xB3	1	1
CPL bit	Complement direct bit	0xB2	2	3
ANL C,bit	AND direct bit to carry flag	0x82	2	2
ANL C,/bit	AND complement of direct bit to carry	0xB0	2	2
ORL C,bit	OR direct bit to carry flag	0x72	2	2
ORL C,/bit	OR complement of direct bit to carry	0xA0	2	2
MOV C,bit	Move direct bit to carry flag	0xA2	2	2
MOV bit,C	Move carry flag to direct bit	0x92	2	3

Table 5. Boolean manipulation

16.1.2.4 DATA TRANSFERS

Mnemonic	Description		Code	Byte	Cycle
MOV A,Rn	Move register to accumulator		0xE8-0xEF	1	1
MOV A,direct	Move direct byte to accumulator		0xE5	2	2
MOV A,@Ri	Move indirect RAM to accumulator		0xE6-0xE7	1	2
MOV A,#data	Move immediate data to accumulator		0x74	2	2
MOV Rn,A	Move accumulator to register		0xF8-0xFF	1	1
MOV Rn,direct	Move direct byte to register		0xA8-0xAF	2	3
MOV Rn,#data	Move immediate data to register		0x78-0x7F	2	2
MOV direct,A	Move accumulator to direct byte		0xF5	2	2
MOV direct,Rn	Move register to direct byte		0x88-8F	2	2
MOV direct1,dire	Move direct byte to direct byte		85	3	3
MOV direct,@Ri	Move indirect RAM to direct byte		86-87	2	3
MOV direct,#dat	Move immediate data to direct byte		75	3	3
MOV @Ri,A	Move accumulator to indirect RAM		F6-F7	1	2
MOV @Ri,direct	Move direct byte to indirect RAM		A6-A7	2	3
MOV @Ri,#data	Move immediate data to indirect RAM		76-77	2	2
MOV DPTR,#dat	Load data pointer with a 16-bit constant		90	3	3
MOVC A,@A+D	Move code byte relative to DPTR to accumulato		93	1	5
MOVC A,@A+P	Move code byte relative to PC to accumulator		83	1	4
MOVX A,@Ri	Move external RAM (8-bit address) to A		E2-E3	1	3*
MOVX A,@DPT	Move external RAM (16-bit address) to A		E0	1	2*
MOVX @Ri,A	Move A to external RAM (8-bit address)	CODE inside ROM/ RAM destination XRAM	F2-F3	1	4*
		all other cases			5*
MOVX @DPTR, A	Move A to external RAM (16-bit address)	CODE inside ROM/ RAM destination XRAM	F0	1	3*
		all other cases			4*
PUSH direct	Push direct byte onto stack		C0	2	3
POP direct	Pop direct byte from stack		D0	2	2
XCH A,Rn	Exchange register with accumulator		C8-CF	1	2
XCH A,direct	Exchange direct byte with accumulator		C5	2	3
XCH A,@Ri	Exchange indirect RAM with accumulator		C6-C7	1	3
XCHD A,@Ri	Exchange low-order nibble indirect RAM with A		D6-D7	1	3

Table 6. Data transfer

* MOVX cycles depends on STRETCH register. Table shows values with STRETCH=0.

16.1.2.5 PROGRAM BRANCHES

Mnemonic	Description	Code	Byte	Cycle
ACALL addr11	Absolute subroutine call	0x11-0xF1	2	4
LCALL addr16	Long subroutine call	03	3	4
RET	Return from subroutine	22	1	4
RETI	Return from interrupt	32	1	4
AJMP addr11	Absolute jump	01-E1	2	3
LJMP addr16	Long jump	02	3	4
SJMP rel	Short jump (relative address)	80	2	3
JMP @A+DPTR	Jump indirect relative to the DPTR	73	1	5
JZ rel	Jump if accumulator is zero	60	2	4
JNZ rel	Jump if accumulator is not zero	70	2	4
JC rel	Jump if carry flag is set	40	2	3
JNC	Jump if carry flag is not set	50	2	3
JB bit,rel	Jump if direct bit is set	20	3	5
JNB bit,rel	Jump if direct bit is not set	30	3	5
JBC bit,direct rel	Jump if direct bit is set and clear bit	10	3	5
CJNE A,direct re	Compare direct byte to A and jump if not equal	B5	3	5
CJNE A,#data r	Compare immediate to A and jump if not equal	B4	3	4
CJNE Rn,#data	Compare immediate to reg. and jump if not equ	B8-BF	3	4
CJNE @Ri,#dat	Compare immediate to ind. and jump if not equ	B6-B7	3	5
DJNZ Rn,rel	Decrement register and jump if not zero	D8-DF	2	4
DJNZ direct,rel	Decrement direct byte and jump if not zero	D5	3	5
NOP	No operation	00	1	1

Table 7. Program branches

16.1.3 INSTRATION SET BRIEF-HEXADECIMAL ORDER

Opcode	Mnemonic	Opcode	Mnemonic
00 H	NOP	30 H	JNB bit.rel
01 H	AJMP addr11	31 H	ACALL addr11
02 H	LJMP addr16	32 H	RETI
03 H	RR A	33 H	RLC A
04 H	INC A	34 H	ADDC A,#data
05 H	INC direct	35 H	ADDC A,direct
06 H	INC @R0	36 H	ADDC A,@R0
07 H	INC @R1	37 H	ADDC A,@R1
08 H	INC R0	38 H	ADDC A,R0
09 H	INC R1	39 H	ADDC A,R1
0A H	INC R2	3A H	ADDC A,R2
0B H	INC R3	3B H	ADDC A,R3
0C H	INC R4	3C H	ADDC A,R4
0D H	INC R5	3D H	ADDC A,R5
0E H	INC R6	3E H	ADDC A,R6
0F H	INC R7	3F H	ADDC A,R7
10 H	JBC bit,rel	40 H	JC rel
11 H	ACALL addr11	41 H	AJMP addr11
12 H	LCALL addr16	42 H	ORL direct,A
13 H	RRC A	43 H	ORL direct,#data
14 H	DEC A	44 H	ORL A,#data
15 H	DEC direct	45 H	ORL A,direct
16 H	DEC @R0	46 H	ORL A,@R0
17 H	DEC @R1	47 H	ORL A,@R1
18 H	DEC R0	48 H	ORL A,R0
19 H	DEC R1	49 H	ORL A,R1
1A H	DEC R2	4A H	ORL A,R2
1B H	DEC R3	4B H	ORL A,R3
1C H	DEC R4	4C H	ORL A,R4
1D H	DEC R5	4D H	ORL A,R5
1E H	DEC R6	4E H	ORL A,R6
1F H	DEC R7	4F H	ORL A,R7
20 H	JB bit.rel	50 H	JNC rel
21 H	AJMP addr11	51 H	ACALL addr11
22 H	RET	52 H	ANL direct,A
23 H	RL A	53 H	ANL direct,#data
24 H	ADD A,#data	54 H	ANL A,#data
25 H	ADD A,direct	55 H	ANL A,direct
26 H	ADD A,@R0	56 H	ANL A,@R0
27 H	ADD A,@R1	57 H	ANL A,@R1
28 H	ADD A,R0	58 H	ANL A,R0
29 H	ADD A,R1	59 H	ANL A,R1
2A H	ADD A,R2	5A H	ANL A,R2
2B H	ADD A,R3	5B H	ANL A,R3
2C H	ADD A,R4	5C H	ANL A,R4
2D H	ADD A,R5	5D H	ANL A,R5
2E H	ADD A,R6	5E H	ANL A,R6
2F H	ADD A,R7	5F H	ANL A,R7

Opcode	Mnemonic	Opcode	Mnemonic
60 H	JZ rel	90 H	MOV DPTR,#data16
61 H	AJMP addr11	91 H	ACALL addr11
62 H	XRL direct,A	92 H	MOV bit,C
63 H	XRL direct,#data	93 H	MOVC A,@A+DPTR

64 H	XRL A,#data	94 H	SUBB A,#data
65 H	XRL A,direct	95 H	SUBB A,direct
66 H	XRL A,@R0	96 H	SUBB A,@R0
67 H	XRL A,@R1	97 H	SUBB A,@R1
68 H	XRL A,R0	98 H	SUBB A,R0
69 H	XRL A,R1	99 H	SUBB A,R1
6A H	XRL A,R2	9A H	SUBB A,R2
6B H	XRL A,R3	9B H	SUBB A,R3
6C H	XRL A,R4	9C H	SUBB A,R4
6D H	XRL A,R5	9D H	SUBB A,R5
6E H	XRL A,R6	9E H	SUBB A,R6
6F H	XRL A,R7	9F H	SUBB A,R7
70 H	JNZ rel	A0 H	ORL C,bit
71 H	ACALL addr11	A1 H	AJMP addr11
72 H	ORL C,direct	A2 H	MOV C,bit
73 H	JMP @A+DPTR	A3 H	INC DPTR
74 H	MOV A,#data	A4 H	MUL AB
75 H	MOV direct,#data	A5 H	-
76 H	MOV @R0,#data	A6 H	MOV @R0,direct
77 H	MOV @R1,#data	A7 H	MOV @R1,direct
78 H	MOV R0.#data	A8 H	MOV R0,direct
79 H	MOV R1.#data	A9 H	MOV R1,direct
7A H	MOV R2.#data	AA H	MOV R2,direct
7B H	MOV R3.#data	AB H	MOV R3,direct
7C H	MOV R4.#data	AC H	MOV R4,direct
7D H	MOV R5.#data	AD H	MOV R5,direct
7E H	MOV R6.#data	AE H	MOV R6,direct
7F H	MOV R7.#data	AF H	MOV R7,direct
80 H	SJMP rel	B0 H	ANL C,bit
81 H	AJMP addr11	B1 H	ACALL addr11
82 H	ANL C,bit	B2 H	CPL bit
83 H	MOVC A,@A+PC	B3 H	CPL C
84 H	DIV AB	B4 H	CJNE A,#data,rel
85 H	MOV direct,direct	B5 H	CJNE A,direct,rel
86 H	MOV direct,@R0	B6 H	CJNE @R0,#data,rel
87 H	MOV direct,@R1	B7 H	CJNE @R1,#data,rel
88 H	MOV direct,R0	B8 H	CJNE R0,#data,rel
89 H	MOV direct,R1	B9 H	CJNE R1,#data,rel
8A H	MOV direct,R2	BA H	CJNE R2,#data,rel
8B H	MOV direct,R3	BB H	CJNE R3,#data,rel
8C H	MOV direct,R4	BC H	CJNE R4,#data,rel
8D H	MOV direct,R5	BD H	CJNE R5,#data,rel
8E H	MOV direct,R6	BE H	CJNE R6,#data,rel
8F H	MOV direct,R7	BF H	CJNE R7,#data,rel

Opcode	Mnemonic	Opcode	Mnemonic
C0	PUSH direct	E0 H	MOVX A,@DPTR
C1	AJMP addr11	E1 H	AJMP addr11
C2	CLR bit	E2 H	MOVX A,@R0
C3	CLR C	E3 H	MOVX A,@R1
C4	SWAP A	E4 H	CLR A
C5	XCH A, direct	E5 H	MOV A, direct
C6	XCH A,@R0	E6 H	MOV A,@R0
C7	XCH A,@R1	E7 H	MOV A,@R1
C8	XCH A,R0	E8 H	MOV A,R0
C9	XCH A,R1	E9 H	MOV A,R1
CA	XCH A,R2	EA H	MOV A,R2

CB	XCH A,R3	EB H	MOV A,R3
CC	XCH A,R4	EC	MOV A,R4
CD	XCH A,R5	ED	MOV A,R5
CE	XCH A,R6	EE H	MOV A,R6
CF	XCH A,R7	EF H	MOV A,R7
D0	POP direct	F0 H	MOVX @DPTR,A
D1	ACALL addr11	F1 H	ACALL addr11
D2	SETB bit	F2 H	MOVX @R0,A
D3	SETB C	F3 H	MOVX @R1,A
D4	DA A	F4 H	CPL A
D5	DJNZ direct, rel	F5 H	MOV direct, A
D6	XCHD A,@R0	F6 H	MOV @R0,A
D7	XCHD A,@R1	F7 H	MOV @R1,A
D8	DJNZ R0,rel	F8 H	MOV R0,A
D9	DJNZ R1,rel	F9 H	MOV R1,A
DA	DJNZ R2,rel	FA H	MOV R2,A
DB	DJNZ R3,rel	FB H	MOV R3,A
DC	DJNZ R4,rel	FC H	MOV R4,A
DD	DJNZ R5,rel	FD H	MOV R5,A
DE	DJNZ R6,rel	FE H	MOV R6,A
DF	DJNZ R7,rel	FF H	MOV R7,A

Table 8. Instruction set brief in hexadecimal order

17 版本说明

版本号	修改时间	修订人	修改内容
V1.0	2019.05.19	RC	初版
V1.1	2019.12.02	RC	新增加 QFN20 封装和 DFN8 封装规格。
V1.2	2019.12.03	RC	修改 ADC 参考源的描述，增加 VDD 参考选项
V1.3	2020.01.08	RC	修改深度休眠模式唤醒条件，将 LVD 中断删除；
V1.4	2020.01.17	RC	修改 TIM1_SR 描述错误；
V1.5	2020.03.19	RC	修改了 UART0CR 中 PAR_ODD 位的描述错误；修改了 DAC 中 VDA2 输出计算公式。
V1.6	2020.06.18	RC	修改了 004 SOP14 引脚复用图示错误
V1.7	2020.10.16	RC	1. 增加 GPIO 中断的操作注意事项； 2. 时钟描述章节增加对 32.768KHz 的描述； 3. 增加 FLASH 访问速度和电压关系的描述。
V1.8	2020.11.11	RC	修改 7.1 绝对最大额定值中 VDD 对地电压最大值为 6.6V。
V1.9	2021.07.26	RC	1. 修改了比较器 AC0_CR1 中 PIS 位的描述错误； 2. 修改了 SCR_PCLK_DIV12 中对 SCK2CKS 位的计算公式； 3. 修改了 SCR_PCLK_DIV3 中来自时钟 SCK2 的计算公式； 4. 增加了 ADC_CR0 中 ADC_IF 位的描述； 5. 补充了 AC0_DASEL 中 DASEL 位的电压计算公式； 6. 补充了 AC1_DASEL 中 DASEL 位的电压计算公式。
V2.0	2021.09.22	RC	1.增加了芯片在高温下串口不建议使用波特率 115200; 2.补充了 BG 偏离值为±50mV; 3.增加了烧录通讯数据 SDA 脚的注意项； 4.添加休眠电流的最大值。
V2.1	2022.04.26	RC	1. 增加外部 RST 复位描述。
V2.2	2022.11.28	RC	1、更换规格书页眉和首页 log。 2、将芯片特性描述放置目录前。
V2.3	2023.5.15	RC	1、添加公司网址至页脚处。
V2.4	2023.6.5	CYH	1、增加 ADC 外部参考电压引脚说明。
V2.5	2024.6.5	CYH	1、删除 DFN8/SOP8/SOP14 相关信息。
V2.6	2024.12.4	CYH	1. 增加回流焊温度曲线和相关说明。