SN8F2280 系列

用户参考手册

SN8F2288
SN8F2283

Version 1.4

SONiX 8 位单片机

修改记录

版本	时间	修改说明
VER 1.0	2009/7	初版。
VER 1.1	2009/9	1、 删除 ADC 外部高参考电压源。 2、 调整唤醒时间的描述。 3、 调整 UART 波特率前置分频的描述。 4、 调整 SIO 转换率选择位。
VER 1.2	2009/10	1、 增加 SN8F2283 的相关信息。 2、 调整低压复位的最小值。（17.2 电气特性）
VER1.3	2009/12	1、 升级 QFN 24 封装信息。（19.3 QFN 24 PIN） 2、 修改电气特性中低电压复位电平相关信息。 3、 删除编译选项表（CODE OPTION）中的 LVD_L 选项。） 4、 将编译选项表（CODE OPTION）中的 Ext_OSC 更改成 High_CLK。 5、 增加 SN8ICE2K Plus II 和 EV-Kit V3.0 的信息。
VER 1.4	2010/1	1、升级 QFN48 PIN 的封装信息。

目　　录

1 产品简介

1.1 功能特性

- **存储器配置**
  Flash ROM：12K x 16位，包括系统内置的可编程功能。可擦除/写入20000次。
  RAM：512 x 8 位。

- **8 层堆栈缓存器**

- **I/O 引脚配置**
  双向输入输出端口：P0、P1、P2、P4、P5。
  具有唤醒功能的端口：P0/P1 电平变换。
  P0、P1、P2、P4、P5 具有上拉功能。
  外部中断：P0、P1（电平变换）。
  P0.5、P0.6、P1.0、P1.1 具有漏极开路功能。

- **全速 USB 2.0**
  遵循 USB 规范 V2.0。
  3.3V 校准输出/驱动 60mA。
  内部 D+ 1.5K 欧姆上拉电阻。
  1 个控制端点。
  2 个双向 INT 端点。
  1 个双向 INT/BULK IN 端点。
  1 个双向 INT/BULK OUT 端点。
  可编程的 EP1~EP4 FIFO depth。

- **功能强大的指令系统**
  单时钟指令系统（1T）。
  大部分指令仅需要一个周期。
  JMP 指令可寻址整个 ROM 区。
  CALL 指令可寻址整个 ROM 区。
  查表指令（MOVC）可寻址整个 ROM 区。

- **系统内置可重复编程功能**
  方便韧件升级

- **内置看门狗定时器**

- **15 个中断源**
  11 个内部中断：T0、T1、TC0、TC1、TC2、USB、SIO、I/O 引脚唤醒、UART、MSP、A/D。
  4 个外部中断：INT0、INT1、P0、P1。

- **SIO 功能，用于数据传送（串行外围设备）**

- **1 个 8 位定时计数器 T0**

- **3 个 8 位定时计数器（TC0、TC1、TC2）**
  TC0、TC1、TC2，具有 8 位 PWM 功能（占空比/周期可编程控制）。

- **1 个 16 位定时计数器（T1）**

- **1 通道的 UART 功能**

- **1 通道 MSP 功能**

- **8 通道 12 位 AD 功能**

- **2 个系统时钟**
  外部高速时钟：晶体模式 6MHz/12MHz/16MHz。
  内部低速时钟：RC 模式，12KHz。

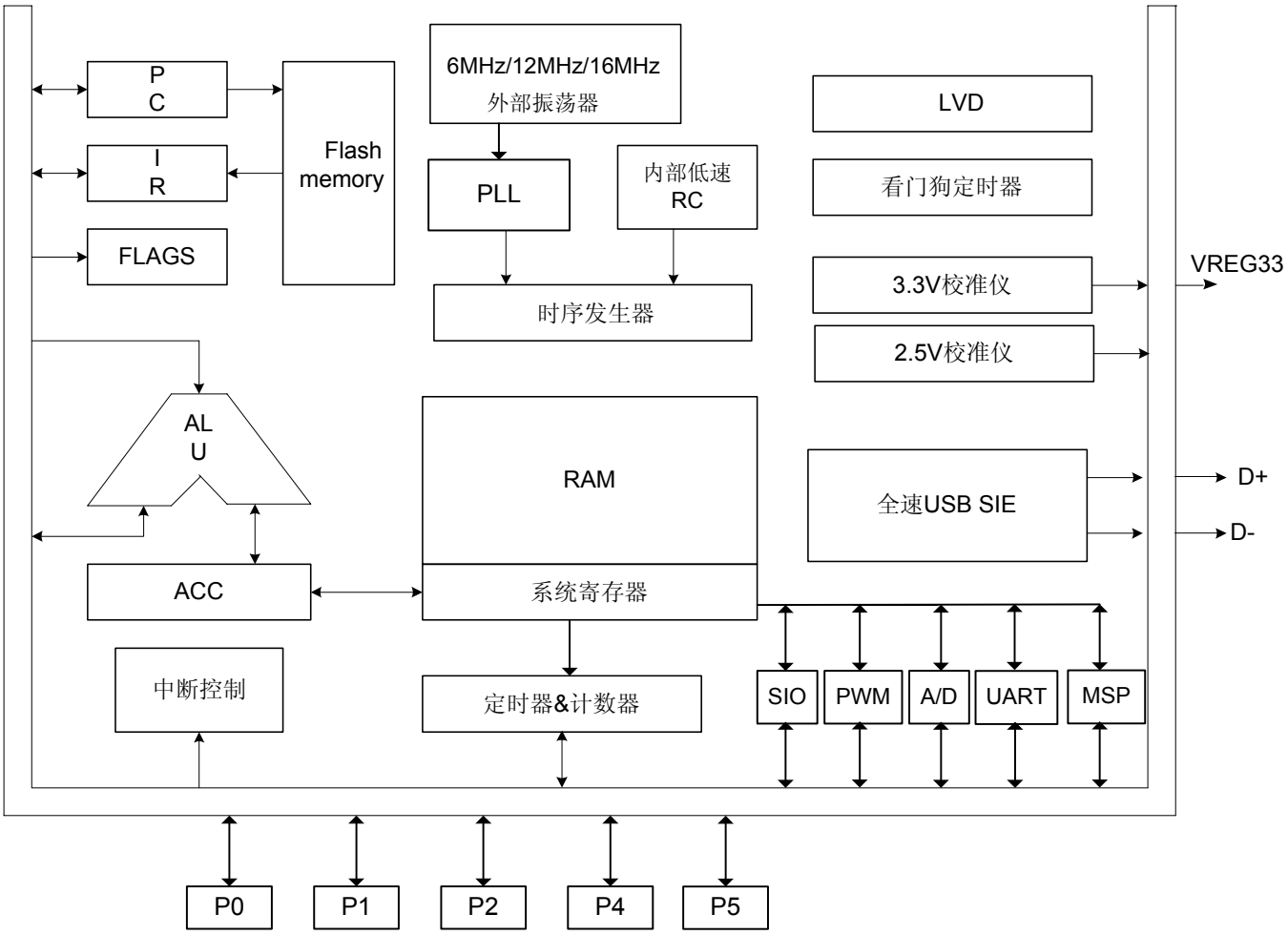
- **4 种工作模式**
  普通模式：高低速时钟都正常工作。
  低速模式：仅低速时钟正常工作。
  睡眠模式：高低速时钟都停止工作。
  绿色模式：有定时器周期性的唤醒。

- **封装形式**
  LQFP48/QFN48/QFN24

☞ 产品性能列表

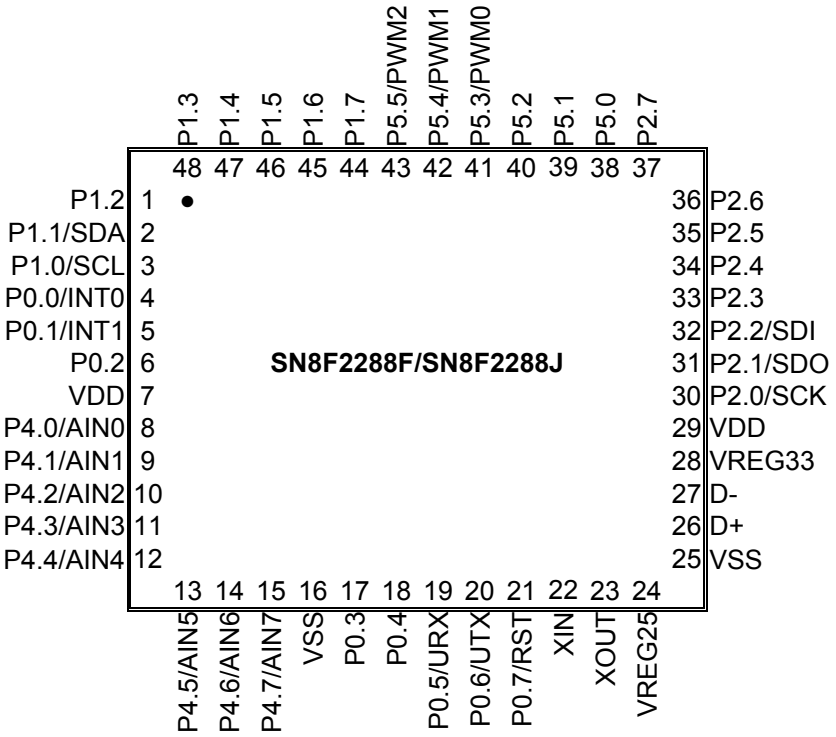
单片机名称	ROM	RAM	定时器					SIO	UART	PWM	A/D	USB	MSP	具有唤醒功能的引脚数目	I/O 引脚数目	封装形式
			T0	T1	TC0	TC1	TC2									
SN8F2288	12K*16	512*8	V	V	V	V	V	V	V	V	12bit	V	V	16	38	LQFP/QFN
SN8F2283	12K*16	512*8	V	V	V	V	V	X	V	X	X	V	X	10	12	QFN

1.2 系统框图

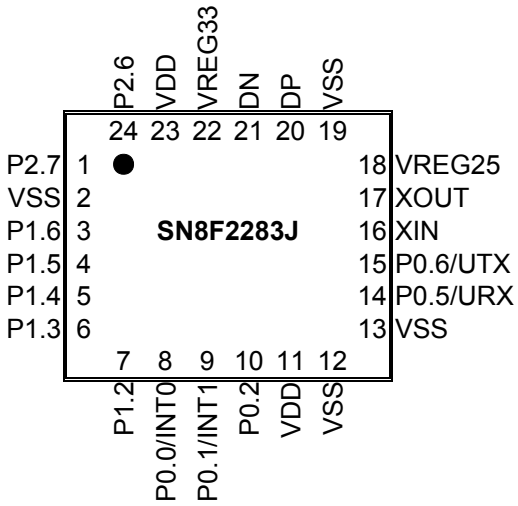

P
C
I
R
FLAGS
Flash
memory
6MHz/12MHz/16MHz
外部振荡器
PLL
内部低速
RC
时序发生器
LVD
看门狗定时器
3.3V校准仪
2.5V校准仪
VREG33
AL
U
RAM
全速USB SIE
D+
D-
ACC
系统寄存器
中断控制
定时器&计数器
SIO
PWM
A/D
UART
MSP
P0
P1
P2
P4
P5

1.3 引脚配置

SN8F2288F (LQFP 48 pins)
SN8F2288J (QFN 48 pins)



SN8F2283J（QFN 24 Pins）

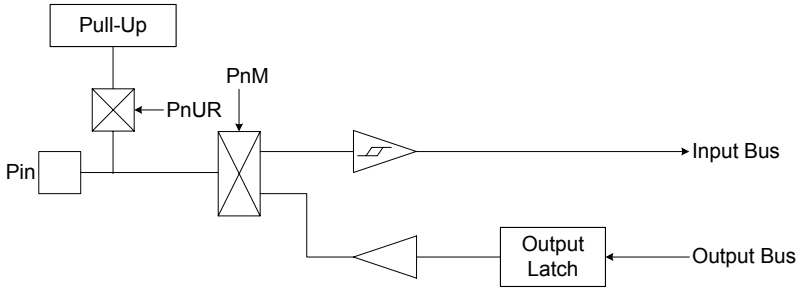


1.4 引脚说明

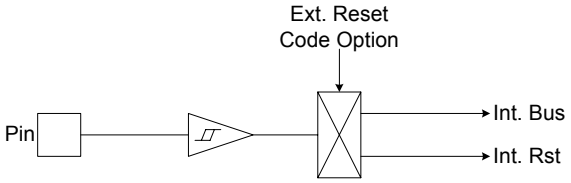
引脚名称	引脚类型	功能说明
VDD, VSS	P	电源输入端。
P0.0/INT0	I/O	P0.0：双向输入输出引脚，施密特触发，输入模式时内置上拉电阻，具有唤醒功能。 INT0：外部中断输入引脚。
P0.1/INT1	I/O	P0.1：双向输入输出引脚，施密特触发，输入模式时内置上拉电阻，具有唤醒功能。 INT1：外部中断输入引脚。
P0[4:2]	I/O	P0：双向输入输出引脚，施密特触发，输入模式时内置上拉电阻，具有唤醒功能。
P0.5/URX	I/O	P0.5：双向输入输出引脚，施密特触发，输入模式时内置上拉电阻，具有唤醒功能。 UART 功能：URX 引脚。 漏极开路功能由寄存器 P1OC 寄存器控制。
P0.6/UTX	I/O	P0.6：双向输入输出引脚，施密特触发，输入模式时内置上拉电阻，具有唤醒功能。 UART 功能：UTX 引脚。 漏极开路功能由寄存器 P1OC 寄存器控制。
P0.7/RST	I/O	RST：Ext_RST 模式时系统外部复位输入引脚，施密特触发，低电平有效，通常保持高电平。 具有唤醒功能。
P1.0/SCL	I/O	P1.0：双向输入输出引脚，施密特触发，输入模式时内置上拉电阻，具有唤醒功能。 漏极开路功能由 P1OC 寄存器控制。 MSP 功能：SCL 功能。
P1.1/SDA	I/O	P1.1：双向输入输出引脚，施密特触发，输入模式时内置上拉电阻，具有唤醒功能。 漏极开路功能由 P1OC 寄存器控制。 MSP 功能：SDA 功能。
P1[7:2]	I/O	双向输入输出引脚，施密特触发，输入模式时内置上拉电阻，具有唤醒功能。
P2.0/SCK	I/O	P2.0：双向输入输出引脚，施密特触发，输入模式时内置上拉电阻。 SCK：SIO 输出时钟引脚。
P2.1/SDO	I/O	P2.1：双向输入输出引脚，施密特触发，输入模式时内置上拉电阻。 SDO：SIO 数据输出引脚。
P2.2/SDI	I/O	P2.2：双向输入输出引脚，施密特触发，输入模式时内置上拉电阻。 SDI：SIO 数据输入引脚。
P2[5:3]	I/O	双向输入输出引脚，施密特触发，输入模式时内置上拉电阻。
P4[7:0]/AIN[7:0]	I/O	P4：双向输入输出引脚，输入模式时内置上拉电阻。 AIN[7:0]：ADC 输入通道。
P5[2:0]	I/O	双向输入输出引脚，施密特触发，输入模式时内置上拉电阻。
P5[5:3]/PWM2~PWM0	I/O	P5：双向输入输出引脚，施密特触发，输入模式时内置上拉电阻。 PWM0, PWM1, PWM2：PWM 功能。
XOUT	I/O	外部振荡器输出引脚。
XIN	I/O	外部振荡器输入引脚。
VREG25	P	2.5V 电源引脚，须与 GND 之间连接一个 1uF 的电容。
VREG33	P	3.3V 电源引脚，须与 GND 之间连接一个 XuF（X=1~10）的电容。
D+, D-	I/O	USB 差分数据线。

1.5 引脚电路结构图

P 0、1、2、4、5：



P0.7：

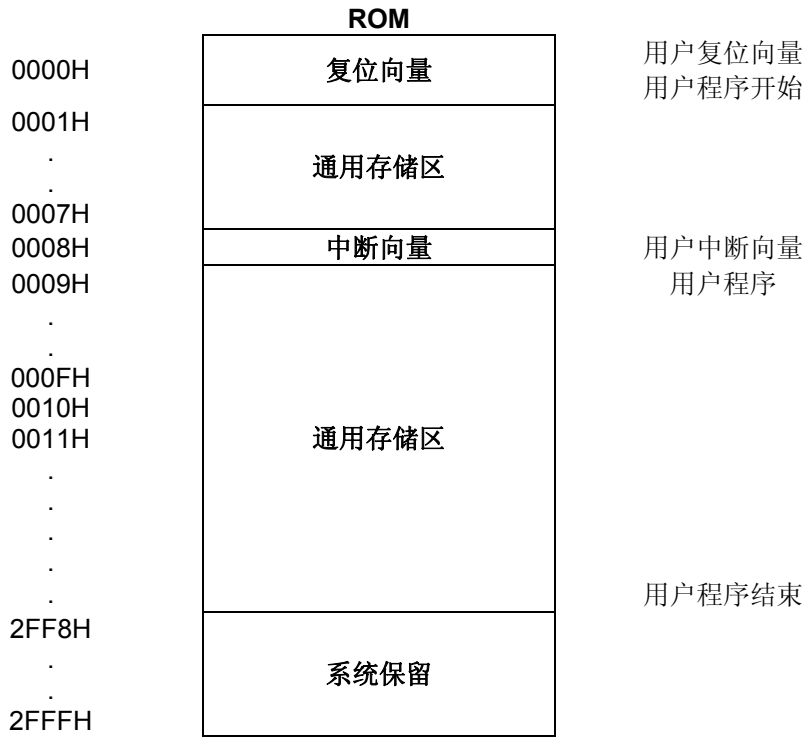


2 中央处理器（CPU）

2.1 存储器

2.1.1 程序存储器（ROM）

☞ **ROM：12K**



2.1.1.1 复位向量（0000H）

具有一个字长的系统复位向量（0000H）。

- **上电复位（NT0=1，NPD=0）；**
- **看门狗复位（NT0=0，NPD=0）；**
- **外部复位（NT0=1，NPD=1）。**

发生上述任一种复位后，程序将从 0000H 处重新开始执行，系统寄存器也都将恢复为默认值。根据 PFLAG 寄存器中的 NT0 和 NPD 标志位的内容可以判断系统复位方式。下面一段程序演示了如何定义 ROM 中的复位向量。

➢ **例：定义复位向量。**

```
                ORG     0               ;
                JMP     START           ; 跳至用户程序。
                …

                ORG     10H
START:                                  ; 0010H，用户程序起始地址。
                …                       ; 用户程序。
                …


                ENDP                    ; 程序结束。
```

2.1.1.2 中断向量（0008H）

中断向量地址为 0008H。一旦有中断响应，程序计数器 PC 的当前值就会存入堆栈缓存器并跳转到 0008H 开始执行中断服务程序。用户必须定义中断向量，下面的示例程序说明了如何编写中断服务程序。

✸ **注：“PUSH”，“POP”指令用于存储和恢复 ACC/PFLAG，不保存（NT0、NTD）。PUSH/POP 缓存器是唯一的，且仅有一层。**

➢ **例：定义中断向量，中断服务程序紧随 ORG 8 之后。**

```
.CODE
                ORG     0               ; 0000H。
                JMP     START           ; 跳转到用户程序。
                …
                ORG     8               ; 中断向量。
                PUSH                    ; 恢复 ACC 和 PFLAG。
                …                       ; 中断返回。
                …
                POP
                RETI                    ; 用户程序开始。
                …                       ; 用户程序。
START:                                  ; 用户程序结束。
                …
                …
                JMP     START           ; 程序结束。
                …                       ; 恢复 ACC 和 PFLAG。
                                        ; 中断返回。
                ENDP
```

➢ **例：定义中断向量。中断服务程序在用户程序之后。**

```
.CODE
                ORG     0               ; 0000H。
                JMP     START           ; 跳转到用户程序。
                …
                ORG     8H              ; 中断向量地址。
                JMP     MY_IRQ          ; 0008H，跳转到中断服务程序。

                ORG     10H
START:                                  ; 0010H，用户程序开始。
                …                       ; 用户程序。
                …
                …
                JMP     START           ;
                …
MY_IRQ:                                 ;中断服务程序开始。
                PUSH                    ; 保存 ACC 和 PFLAG。
                …
                …
                POP                     ; 恢复 ACC 和 PFLAG。
                RETI                    ; 中断返回。
                …

                ENDP                    ; 程序结束。
```

✸ **注：从上面的程序中容易得出 SONiX 的编程规则，有以下几点：**

1. **地址 0000H 的“JMP”指令使程序从头开始执行；**
2. **地址 0008H 是中断向量；**
3. **用户的程序应该是一个循环。**

2.1.1.3 查表

在 SONiX 单片机中，对 ROM 区中的数据进行查找，寄存器 Y 指向所找数据地址的中间字节（bit8~bit15），寄存器 Z 指向所找数据地址的低字节（bit0~bit7）。执行完 MOVC 指令后，所查找数据低字节内容被存入 ACC 中，而数据高字节内容被存入 R 寄存器。

➢ **例：查找 ROM 地址为“TABLE1”的值。**

```
                B0MOV   Y, #TABLE1$M    ; 设置 TABLE1 地址高字节。
                B0MOV   Z, #TABLE1$L    ; 设置 TABLE1 地址低字节。
                MOVC                    ; 查表，R = 00H，ACC = 35H。

                                        ; 查找下一地址。
                INCMS   Z
                JMP     @F              ; Z 没有溢出。
                INCMS   Y               ; Z 溢出（FFH → 00），→ Y=Y+1
                NOP                     ;
                                        ;
@@:             MOVC                    ; 查表，R = 51H，ACC = 05H。
                …                       ;
TABLE1:         DW      0035H           ; 定义数据表（16 位）数据。
                DW      5105H
                DW      2012H
                …
```

✸ 注：当寄存器 Z 溢出（从 0FFH 变为 00H）时，寄存器 Y 并不会自动加 1。因此， Z 溢出时，Y 必须由程序加 1，下面的宏 INC_YZ 能够对 Y 和 Z 寄存器自动处理。

➢ **例：宏 INC_YZ。**

```
INC_YZ          MACRO
                INCMS   Z
                JMP     @F              ; 没有溢出。

                INCMS   Y
                NOP                     ; 没有溢出。
@@:
                ENDM
```

➢ **例：通过“INC_YZ”对上例进行优化。**

```
                B0MOV   Y, #TABLE1$M    ; 设置 TABLE1 地址中间字节。
                B0MOV   Z, #TABLE1$L    ; 设置 TABLE1 地址低字节。
                MOVC                    ; 查表，R = 00H，ACC = 35H。

                INC_YZ                  ; 查找下一地址数据。
                                        ;
@@:             MOVC                    ; 查表，R = 51H，ACC = 05H。
                …                       ;
TABLE1:         DW      0035H           ; 定义数据表（16 位）数据。
                DW      5105H
                DW      2012H
                …
```

下面的程序通过累加器对 Y，Z 寄存器进行处理来实现查表功能，但需要特别注意进位时的处理。

➢ **例：由指令 B0ADD/ADD 对 Y 和 Z 寄存器加 1。**

```
                B0MOV   Y, #TABLE1$M    ; 设置 TABLE1 地址中间字节。
                B0MOV   Z, #TABLE1$L    ; 设置 TABLE1 地址低字节。

                B0MOV   A, BUF          ; Z = Z + BUF。
                B0ADD   Z, A

                B0BTS1  FC              ; 检查进位标志。
                JMP     GETDATA         ; FC = 0。
                INCMS   Y               ; FC = 1。
                NOP
GETDATA:                                ;
                MOVC                    ; 存储数据，如果 BUF = 0，数据为 0035H。
                                        ; 如果 BUF = 1，数据＝5105H。
                                        ; 如果 BUF = 2，数据＝2012H。
                …

TABLE1:         DW      0035H           ; 定义数据表（16 位）数据。
                DW      5105H
                DW      2012H
                …
```

2.1.1.4 跳转表

跳转表能够实现多地址跳转功能。由于 PCL 和 ACC 的值相加即可得到新的 PCL，因此，可以通过对 PCL 加上不同的 ACC 值来实现多地址跳转。ACC 值若为 n，PCL+ACC 即表示当前地址加 n，执行完当前指令后 PCL 值还会自加 1，可参考以下范例。如果 PCL＋ACC 后发生溢出，PCH 则自动加 1。由此得到的新的 PC 值再指向跳转指令列表中新的地址。这样，用户就可以通过修改 ACC 的值轻松实现多地址的跳转。

✸ 注：PCH 只支持 PC 增量运算，而不支持 PC 减量运算。当 PCL+ACC 后如有进位，PCH 的值会自动加 1。PCL-ACC 后若有借位，PCH 的值将保持不变，用户在设计应用时要加以注意。

➢ **例：跳转表。**

```
        ORG     0100H       ; 跳转表从 ROM 前端开始。

        B0ADD   PCL, A      ; PCL = PCL + ACC，PCL 溢出时 PCH 加 1。
        JMP     A0POINT     ; ACC = 0，跳至 A0POINT。
        JMP     A1POINT     ; ACC = 1，跳至 A1POINT。
        JMP     A2POINT     ; ACC = 2，跳至 A2POINT。
        JMP     A3POINT     ; ACC = 3，跳至 A3POINT。
```

SONiX 单片机提供一个宏以保证可靠执行跳转表功能，它会自动检测 ROM 边界并将跳转表移至适当的位置。但采用该宏程序会占用部分 ROM 空间。

➢ **例：宏 “MACRO3.H”中，“@JMP_A”的应用。**

```
        B0MOV   A, BUF0     ; “BUF0”从 0 至 4。
        @JMP_A  5           ; 列表个数为 5。
        JMP     A0POINT     ; ACC = 0，跳至 A0POINT。
        JMP     A1POINT     ; ACC = 1，跳至 A1POINT。
        JMP     A2POINT     ; ACC = 2，跳至 A2POINT。
        JMP     A3POINT     ; ACC = 3，跳至 A3POINT。
        JMP     A4POINT     ; ACC = 4，跳至 A4POINT。
```

如果跳转表跨越了 ROM 的边界（00FFH~0100H），宏指令“@JMP_A”会调整跳转表的位置使其从 ROM 的前端开始。

➢ **例：宏指令@JMP_A 操作。**

；编译前

```
ROM 地址
        B0MOV   A, BUF0     ; “BUF0”从 0 至 4。
        @JMP_A  5           ; 列表个数为 5。
00FDH   JMP     A0POINT     ; ACC = 0，跳至 A0POINT。
00FEH   JMP     A1POINT     ; ACC = 1，跳至 A1POINT。
00FFH   JMP     A2POINT     ; ACC = 2，跳至 A2POINT。
0100H   JMP     A3POINT     ; ACC = 3，跳至 A3POINT。
0101H   JMP     A4POINT     ; ACC = 4，跳至 A4POINT。
```

；编译后

```
ROM 地址
        B0MOV   A, BUF0     ; “BUF0”从 0 至 4。
        @JMP_A  5           ; 列表个数为 5。
0100H   JMP     A0POINT     ; ACC = 0，跳至 A0POINT。
0101H   JMP     A1POINT     ; ACC = 1，跳至 A1POINT。
0102H   JMP     A2POINT     ; ACC = 2，跳至 A2POINT。
0103H   JMP     A3POINT     ; ACC = 3，跳至 A3POINT。
0104H   JMP     A4POINT     ; ACC = 4，跳至 A4POINT。
```

2.1.1.5 CHECKSUM 计算

ROM 区域的最后一个地址是系统保留区域，用户在进行 Checksum 计算时应该跳过这一区域。

➢ **例：下面的程序给出了在进行 Checksum 计算时如何跳过系统保留区域。**

```
                MOV         A,#END_USER_CODE$L
                B0MOV       END_ADDR1, A            ; 保存结束地址的低字节。
                MOV         A,#END_USER_CODE$M
                B0MOV       END_ADDR2, A            ; 保存结束地址的高字节。
                CLR         Y                       ; 清 Y。
                CLR         Z                       ; 清 Z。
@@:
                MOVC
                B0BCLR      FC                      ; 清标志 C。
                ADD         DATA1, A                ;
                MOV         A, R
                ADC         DATA2, A                ;
                JMP         END_CHECK               ; 检查 YZ 的值是否指向结束地址。
AAA:
                INCMS       Z                       ; Z=Z+1。
                JMP         @B                      ;
                JMP         Y_ADD_1                 ;
END_CHECK:
                MOV         A, END_ADDR1
                CMPRS       A, Z                    ; 检查 Z 是否等于结束地址的低字节。
                JMP         AAA                     ; 不是继续计算。
                MOV         A, END_ADDR2
                CMPRS       A, Y                    ; 是则检查 Y 是否等于结束地址的高字节。
                JMP         AAA                     ; 不是继续计算。
                JMP         CHECKSUM_END            ; 是则结束计算。

Y_ADD_1:
                INCMS       Y                       ;
                NOP
                JMP         @B                      ;
CHECKSUM_END:
                …
                …
END_USER_CODE:                                      ; 程序结束。
```

2.1.2 编译选项列表（CODE OPTION）

编译选项	内容	功能说明
High_CLK	6MHz	外部振荡器采用 6MHz 晶体/陶瓷振荡器。
	12MHz	外部振荡器采用 12MHz 晶体/陶瓷振荡器。
	16MHz	外部振荡器采用 16MHz 晶体/陶瓷振荡器。
Watch_Dog	Always_On	始终开启看门狗定时器，即使在睡眠模式和绿色模式下也处于开启状态。
	Enable	使能看门狗定时器，但在睡眠模式和绿色模式下关闭看门狗定时器。
	Disable	关闭看门狗定时器。
Fcpu	Fhosc/1	指令周期为 **12 MHz 时钟。**
	Fhosc/2	指令周期为 **6 MHz 时钟。**
	Fhosc/4	指令周期为 **3 MHz 时钟。**
	Fhosc/8	指令周期为 **1.5 MHz 时钟。**
Reset_Pin	Reset	使能外部复位引脚。
	P07	使能 P0.7 的普通 I/O 功能。
Fslow	Flosc/2	低速模式时钟 = Flosc/2。
	Flosc/4	低速模式时钟 = Flosc/4。
Rst_Length	No	无外部复位 de-bounce 时间。
	128*ILRC	外部复位 de-bounce 时间 = 128*ILRC。
LVD	LVD_M	2.4V 低电压检测。
	LVD_H	3.6V 低电压检测。
Security	Enable	ROM 代码加密。
	Disable	ROM 代码不加密。

2.1.3 数据存储器（RAM）

☞ **RAM：512×8 位**

	地址	RAM	
BANK 0	000h “ “ “ “ “ 07Fh	通用存储区	BANK 0
	080h “ “ “ “ “ 0FFh	系统寄存器 Bank0 结束区	Bank 0 的 80h~FFh 存储系统寄存器（128 字节）。
BANK1	100h “ “ “ “ “ 1FFh	通用存储区	BANK1
BANK2	200h “ “ “ “ “ 27Fh	通用存储区	BANK2

☞ **RAM（USB DATA FIFO）：136 x 8 位**

- 端点 0 仅支持控制管道－8 字节。
- 端点 1 至端点 4 支持中断数据的传送，Bulk 数据的传送－配置 FIFO Depth，由 USB FIFO 控制寄存器设置。

136 x 8 RAM (FIFO)

00h ~ 07h	端点 0 RAM（8 字节）
08h	端点 1 RAM（W 字节） 中断 IN/OUT
	端点 2 RAM（X 字节） 中断 IN/OUT
	端点 3 RAM（Y 字节） 中断 IN/OUT BULK IN/OUT
	端点 4 RAM（Z 字节） 中断 IN/OUT BULK IN/OUT

2.1.4 系统寄存器

2.1.4.1 系统寄存器列表

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	-	-	R	Z	Y	-	PFLAG	RBANK	TC0M	TC0C	TC0R	TC1M	TC1C	TC1R	TC2M	TC2C
9	TC2R	UDA	USTATUS	EP0OUT_CNT	USB_INT_EN	EP_ACK	EP_NAK	UE0R	UE1R	UE1R_C	UE2R	UE2R_C	UE3R	UE3R_C	UE4R	UE4R_C
A	EP2FIFO_ADDR	EP3FIFO_ADDR	EP4FIFO_ADDR	UDP0	-	UDR0_R	UDR0_W	UPID	UToggle	URTX	URRX	URBRC	URTXD1	URTXD2	URRXD1	URRXD2
B	SIOM	SIOR	SIOB	-	-	P0M	ADM	ADB	ADR	P4CON	PECMD	PEROML	PEROMH	PERAML	PERAMCNT	PEDGE
C	P1W	P1M	P2M	-	P4M	P5M	INTRQ1	INTEN1	INTRQ	INTEN	OSCM	-	WDTR	-	PCL	PCH
D	P0	P1	P2	-	P4	P5	-	-	T0M	T0C	T1M	T1CL	T1CH	-	-	STKP
E	P0UR	P1UR	P2UR	-	P4UR	P5UR	-	@YZ	-	P1OC	MSPSTAT	MSPM1	MSPM2	MSPBUF	MSPADR	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.1.4.2 系统寄存器说明

R = 工作寄存器，ROM 查表寄存器
PFLAG = ROM 页和特殊标志寄存器
UDA = USB 控制寄存器
UEXR_C = EPX 字节计数寄存器
UDP0 = USB FiFo 地址指针
UDR0_W = UDP0 所指 USB FiFo 写入读取数据缓存器
EP_ACK = 端点 ACK 标志寄存器
UToggle = USB 端点触发位控制寄存器
USTATUS = USB 状态寄存器
EP0OUT_CNT = USB 端点 0 OUT Token 数据字节计数器
SIOM = SIO 模式控制寄存器
SIOB = SIO 数据缓存器
PnM = Pn 模式控制寄存器
INTRQ = 中断请求寄存器
INTRQ1 = 中断请求寄存器
OSCM = 振荡模式寄存器
TC0R = TC0 自动装载数据缓存器
Pn = Pn 数据缓存器
TnC = Tn 计数寄存器，n = 0、1、C0、C1
PnUR = Pn 上拉电阻控制寄存器
P1W = P1 唤醒控制寄存器
PEROM = ISP ROM 地址
PERAMCNT = ISP RAM 可编程计数寄存器
UTRX = UART RX 控制寄存器
URXXDX = UART 数据缓存器
ADB, ADR = A/D 转换数据缓存器
MSPMX = MSP 模式寄存器

Y, Z = 工作寄存器，@YZ 和 ROM 寻址寄存器
RBANK = RAM bank 选择寄存器
UEXR = EPX 控制寄存器
EPXFIFO_ADDR = EPX FIFO USB FIFO 的起始地址
UDR0_R = UDP0 所指 USB FiFo 读取数据缓存器
EP_NAK = 端点 NAK 标志寄存器
UPID = USB 总线控制寄存器
USB_INT_EN = USB 中断控制寄存器
SIOR = SIO 时钟装载缓存器
PEDGE = P0.0、P0.1 边沿控制寄存器
INTEN = 中断使能寄存器
INTEN1 = 中断使能寄存器
WDTR = 看门狗清零寄存器
PCH, PCL = 程序计数器
TnM = Tn 模式寄存器，n= 0、1、C0、C1、C2
TnR = Tn 寄存器，n = C0、C1、C2
STKP = 堆栈指针
@YZ = RAM YZ 间接寻址寄存器
STK0~STK7 = 堆栈
PECMD = ISP 命令寄存器
PERAM = ISP RAM 映射地址
URTX = UART TX 控制寄存器
URBRC = UART Baud rate register
ADM = A/D 转换模式控制寄存器
MSPSTAT = MSP 状态寄存器
MSPBUF = MSP 缓存器
MSPADR = MSP 地址

2.1.4.3 系统寄存器的位定义

地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	备注
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
086H	NT0	NPD				C	DC	Z	R/W	PFLAG
087H							RBNKS1	RBNKS0	R/W	RBANK
088H	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT	R/W	TC0M
089H	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
08AH	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	R/W	TC0R
08BH	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS	ALOAD1	TC1OUT	PWM1OUT	R/W	TC1M
08CH	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0	R/W	TC1C
08DH	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0	R/W	TC1R
08EH	TC2ENB	TC2rate2	TC2rate1	TC2rate0	TC2CKS	ALOAD2	TC2OUT	PWM2OUT	R/W	TC2M
08FH	TC2C7	TC2C6	TC2C5	TC2C4	TC2C3	TC2C2	TC2C1	TC2C0	R/W	TC2C
090H	TC2R7	TC2R6	TC2R5	TC2R4	TC2R3	TC2R2	TC2R1	TC2R0	R/W	TC2R
091H	UDE	UDA6	UDA5	UDA4	UDA3	UDA2	UDA1	UDA0	R/W	UDA
092H	CRCERR	PKTERR	SOF	BUS_RST	SUSPEND	EP0SETUP	EP0IN	EP0OUT	R/W	USTATUS
093H				UEP0OC4	UEP0OC3	UEP0OC2	UEP0OC1	UEP0OC0	R/W	EP0OUT_CNT
094H	REG_EN	DP_PU_EN	SOF_INT _EN		EP4NAK _INT_EN	EP3NAK _INT_EN	EP2NAK _INT_EN	EP1NAK _INT_EN	R/W	USB_INT_EN
095H					EP4_ACK	EP3_ACK	EP2_ACK	EP1_ACK	R/W	EP_ACK
096H					EP4_NAK	EP3_NAK	EP2_NAK	EP1_NAK	R/W	EP_NAK
097H		UE0M1	UE0M0		UE0C3	UE0C2	UE0C1	UE0C0	R/W	UE0R
098H	UE1E	UE1M1	UE1M0						R/W	UE1R
099H		UE1C6	UE1C5	UE1C4	UE1C3	UE1C2	UE1C1	UE1C0	RW	UE1R_C
09AH	UE2E	UE2M1	UE2M0						R/W	UE2R
09BH		UE2C6	UE2C5	UE2C4	UE2C3	UE2C2	UE2C1	UE2C0	RW	UE2R_C
09CH	UE3E	UE3M1	UE3M0						R/W	UE3R
09DH		UE3C6	UE3C5	UE3C4	UE3C3	UE3C2	UE3C1	UE3C0	RW	UE3R_C
09EH	UE4E	UE4M1	UE4M0						R/W	UE4R
09FH		UE4C6	UE4C5	UE4C4	UE4C3	UE4C2	UE4C1	UE4C0	RW	UE4R_C
0A0H	EP2FIFO7	EP2FIFO6	EP2FIFO5	EP2FIFO4	EP2FIFO3	EP2FIFO2	EP2FIFO1	EP2FIFO0	R/W	EP2FIFO_ADDR
0A1H	EP3FIFO7	EP3FIFO6	EP3FIFO5	EP3FIFO4	EP3FIFO3	EP3FIFO2	EP3FIFO1	EP3FIFO0	R/W	EP3FIFO_ADDR
0A2H	EP4FIFO7	EP4FIFO6	EP4FIFO5	EP4FIFO4	EP4FIFO3	EP4FIFO2	EP4FIFO1	EP4FIFO0	R/W	EP4FIFO_ADDR
0A3H	UDP07	UDP06	UDP05	UDP04	UDP03	UDP02	UDP01	UDP00	R/W	UDP0
0A5H	UDR0_R7	UDR0_R6	UDR0_R5	UDR0_R4	UDR0_R3	UDR0_R2	UDR0_R1	UDR0_R0	R/W	UDR0_R
0A6H	UDR0_W7	UDR0_W6	UDR0_W5	UDR0_W4	UDR0_W3	UDR0_W2	UDR0_W1	UDR0_W0	R/W	UDR0_W
0A7H						UBDE	DDP	DDN	R/W	UPID
0A8H					EP4_DATA 01	EP3_DATA 01	EP2_DATA 01	EP1_DATA 01	R/W	Utoggle
0A9H	UCLKS			UTXEN	UTXPEN	UTXPS	UTXM		R/W	URTX
0AAH	URXEN	URXS1	URXS0	URXPEN	URXPS	URXPC	URXM		R/W	URRX
0ABH	UDIV4	UDIV3	UDIV2	UDIV1	UDIV0	UPCS2	UPCS1	UPCS0	R/W	URBRC
0ACH	UTXD17	UTXD16	UTXD15	UTXD14	UTXD13	UTXD12	UTXD11	UTXD10	R/W	URTXD1
0ADH	UTXD27	UTXD26	UTXD25	UTXD24	UTXD23	UTXD22	UTXD21	UTXD20	R/W	URTXD2
0AEH	URXD17	URXD16	URXD15	URXD14	URXD13	URXD12	URXD11	URXD10	R	URRXD1
0AFH	URXD27	URXD26	URXD25	URXD24	URXD23	URXD22	URXD21	URXD20	R	URRXD2
0B0H	SENB	START	SRATE1	SRATE0	MLSB	SCKMD	CPOL	CPHA	R/W	SIOM
0B1H	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0	W	SIOR
0B2H	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0	R/W	SIOB
0B5H	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M	R/W	P0M
0B6H	ADENB	ADS	EOC	GCHS	CHS3	CHS2	CHS1	CHS0	R/W	ADM
0B7H	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	R/W	ADB
0B8H	ADCKS2	ADCKS1	ADCKS0	ADLEN	ADB3	ADB2	ADB1	ADB0	RW	ADR
0B9H	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0	R/W	P4CON
0BAH	PECMD7	PECMD6	PECMD5	PECMD4	PECMD3	PECMD2	PECMD1	PECMD0	W	PECMD
0BBH	PEROML7	PEROML6	PEROML5	PEROML4	PEROML3	PEROML2	PEROML1	PEROML0	R/W	PEROML
0BCH	PEROMH7	PEROMH6	PEROMH5	PEROMH4	PEROMH3	PEROMH2	PEROMH1	PEROMH0	R/W	PEROMH
0BDH	PERAML7	PERAML6	PERAML5	PERAML4	PERAML3	PERAML2	PERAML1	PERAML0	R/W	PERAML
0BEH	PERAMCNT 4	PERAMCNT 3	PERAMCNT 2	PERAMCNT 1	PERAMCNT 0		PERAML9	PERAML8	R/W	PERAMCNT
0BFH					P01G1	P01G0	P00G1	P00G0	R/W	PEDGE
0C0H	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W	R/W	P1W
0C1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C2H	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M	R/W	P2M
0C4H	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M	R/W	P4M
0C5H			P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5M
0C6H	P1IRQ	P0IRQ	MSPIRQ	UTRXIRQ	UTTXIRQ	T2CIRQ	TC1IRQ	TC0IRQ	R/W	INTRQ1
0C7H	P1IEN	P0IEN	MSPIEN	UTRXIEN	UTTXIEN	TC2IEN	TC1IEN	TC0IEN	R/W	INTEN1
0C8H	ADCIRQ	USBIRQ	T1IRQ	T0IRQ	SIOIRQ	WAKEIRQ	P01IRQ	P00IRQ	R/W	INTRQ
0C9H	ADCIEN	USBIEN	T1IEN	T0IEN	SIOIEN	WAKEIEN	P01IEN	P00IEN	R/W	INTEN

0CAH				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH			PC13	PC12	PC11	PC10	PC9	PC8	R/W	PCH
0D0H	P07	P06	P05	P04	P03	P02	P01	P00	R/W	P0
0D1H	P17	P16	P15	P14	P13	P12	P11	P10	R/W	P1
0D2H	P27	P26	P25	P24	P23	P22	P21	P10	R/W	P2
0D4H	P47	P46	P45	P44	P43	P42	P41	P40	R/W	P4
0D5H			P55	P54	P53	P52	P51	P50	R/W	P5
0D8H	T0ENB	T0rate2	T0rate1	T0rate0					R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	T1ENB	T1rate2	T1rate1	T1rate0					R/W	T1M
0DBH	T1C7	T1C6	T1C5	T1C4	T1C3	T1C2	T1C1	T1C0	R/W	T1CL
0DCH	T1C15	T1C14	T1C13	T1C12	T1C11	T1C10	T1C9	T1C8	R/W	T1CH
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H	P07R	P06R	P05R	P04R	P03R	P02R	P01R	P00R	W	P0UR
0E1H	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E2H	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R	W	P2UR
0E4H	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R	W	P4UR
0E5H			P55R	P54R	P53R	P52R	P51R	P50R	W	P5UR
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0E9H					P06OC	P05OC	P11OC	P10OC	R/W	P1OC
0EAH		CKE	D_A	P	S	RED_WRT		BF	R	MSPSTAT
0EBH	WCOL	MSPOV	MSPENB	CKP	SLRXCKP	MSPWK		MSPC	R/W	MSPM1
0ECH	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	R/W	MSPM2
0EDH	MSPBUF7	MSPBUF6	MSPBUF5	MSPBUF4	MSPBUF3	MSPBUF2	MSPBUF1	MSPBUF0	R/W	MSPBUF
0EEH	MSPADR7	MSPADR6	MSPADR5	MSPADR4	MSPADR3	MSPADR2	MSPADR1	MSPADR0	R/W	MSPADR
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H			S7PC13	S7PC12	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H			S6PC13	S6PC12	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H			S5PC13	S5PC12	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H			S4PC13	S4PC12	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H			S3PC13	S3PC12	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH			S2PC13	S2PC12	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH			S1PC13	S1PC12	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH			S0PC13	S0PC12	S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

✸ **注:**

1. **所有系统寄存器的名称都已经在 SN8ASM 编译器中宣告过。**
2. **寄存器各位的名称都已经以“F”为前缀在 SN8ASM 宣告过。**
3. **指令“b0bset”, “b0bclr”, ”bset”, ”bclr”仅对“R/W”寄存器有效。**
4. **详细内容请参考“系统寄存器快速参照表”。**

2.1.4.4 累加器 ACC

8 位数据寄存器 ACC 用来执行 ALU 与数据存储器之间数据的传送操作。如果操作结果为零（Z）或有进位产生（C 或 DC），程序状态寄存器 PFLAG 中相应位会发生变化。

ACC 并不在 RAM 中，因此在立即寻址模式中不能用“B0MOV”指令对其进行读写。

➢ **例：读/写 ACC。**

```
; 数据写入 ACC。
                MOV         A, #0FH

; 读取 ACC 中的数据并存入 BUF。
                MOV         BUF, A
                B0MOV       BUF, A

; BUF 中的数据写入 ACC。
                MOV         A, BUF
                B0MOV       A, BUF
```

系统执行中断操作时，ACC 和 PFLAG 中的数据不会自动存储，用户需通过程序将中断入口处的 ACC 和 PFLAG 中的数据送入存储器进行保存。可通过“PUSH”和“POP”指令对 ACC 和 PFLAG 等系统寄存器进行存储及恢复。

➢ **例：ACC 和工作寄存器中断保护操作。**

```
INT_SERVICE:
                PUSH                    ; 保存 ACC 和 PFLAG。
                …           .
                …
                POP                     ; 恢复 ACC 和 PFLAG。

                RETI                    ; 退出中断。
```

2.1.4.5 程序状态寄存器 PFLAG

寄存器PFLAG中包含ALU运算状态信息、系统复位状态信息和LVD低电压检测状态信息。其中，位NT0和NPD显示系统复位状态信息，包括上电复位、LVD复位、外部复位和看门狗复位；位C、DC和Z显示ALU的运算信息。

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	-	-	-	C	DC	Z
读/写	R/W	R/W	-	-	-	R/W	R/W	R/W
复位后	-	-	-	-	-	0	0	0

Bit [7:6] **NT0, NPD**：复位状态标志位。

NT0	NPD	复位状态
0	0	看门狗定时器溢出
0	1	保留
1	0	LVD 复位
1	1	外部复位

Bit 2 **C**：进位标志。
1 =加法运算后有进位、减法运算没有借位发生或移位后移出逻辑“1”或比较运算的结果≥0；
0 =加法运算后没有进位、减法运算有借位发生或移位后移出逻辑“0”或比较运算的结果<0。

Bit 1 **DC**：辅助进位标志。
1 =加法运算时低四位有进位，或减法运算后没有向高四位借位；
0 =加法运算时低四位没有进位，或减法运算后有向高四位借位。

Bit 0 **Z**：零标志。
1 =算术/逻辑/分支转移运算的结果为零；
0 =算术/逻辑/分支转移运算的结果非零。

✸ **注：关于标志位 C、DC 和 Z 的更多信息请参阅指令集相关内容。**

2.1.4.6 程序计数器

程序计数器 PC 是一个 14 位二进制程序地址寄存器，分高 6 位和低 8 位。专门用来存放下一条需要执行指令的内存地址。通常，程序计数器会随程序中指令的执行自动增加。

若程序执行 CALL 和 JMP 指令时，PC 指向特定的地址。

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC	-	-	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
复位后	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PCH								PCL							

☞ **单地址跳转**

在 SONiX 单片机里面，有 9 条指令（CMPRS、INCS、INCMS、DECS、DECMS、BTS0、BTS1、B0BTS0 和 B0BTS1）可完成单地址跳转功能。如果这些指令执行结果为真，那么 PC 值加 2 以跳过下一条指令。

如果位测试为真，PC 加 2。

```
                B0BTS1      FC          ; 若 Carry_flag = 1 则跳过下一条指令。
                JMP         C0STEP      ; 否则执行 C0STEP。
                …
                …
C0STEP:         NOP

                B0MOV       A, BUF0     ; BUF0 送入 ACC。
                B0BTS0      FZ          ; Zero flag = 0 则跳过下一条指令。
                JMP         C1STEP      ; 否则执行 C1STEP。
                …
                …
C1STEP:         NOP
```

如果 ACC 等于指定的立即数则 PC 值加 2，跳过下一条指令。

```
                CMPRS       A, #12H     ; 若 ACC = 12H，则跳过下一条指令。
                JMP         C0STEP      ; 否则跳至 C0STEP。
                …
                …
C0STEP:         NOP
```

执行加 1 指令后，结果为零时，PC 的值加 2，跳过下一条指令。

INCS：

```
                INCS        BUF0
                JMP         C0STEP
                …
C0STEP:         NOP
```

INCMS：

```
                INCMS       BUF0
                JMP         C0STEP
                …
C0STEP:         NOP
```

执行减 1 指令后，结果为零时，PC 的值加 2，跳过下一条指令。

DECS：

```
                DECS        BUF0
                JMP         C0STEP
                …
C0STEP:         NOP
```

DECMS：

```
                DECMS       BUF0
                JMP         C0STEP
                …
C0STEP:         NOP
```

☞ **多地址跳转**

执行 JMP 或 ADD M,A（M=PCL）指令可实现多地址跳转。执行 ADD M, A、ADC M, A 或 B0ADD M, A 后，若 PCL 溢出，PCH 会自动进位。对于跳转表及其它应用，用户可以通过上述 3 条指令计算 PC 的值而不需要担心 PCL 溢出的问题。

✸ **注：PCH 仅支持 PC 的递增运算而不支持递减运算。当 PCL+ACC 执行完 PCL 有进位时，PCH 会自动加 1；但执行 PCL-ACC 有借位发生，PCH 的值会保持不变。**

➢ **例：PC = 0323H（PCH = 03H，PCL = 23H）。**

```
; PC = 0323H
                MOV     A, #28H
                B0MOV   PCL, A          ; 跳到地址 0328H。
                …

; PC = 0328H
                MOV     A, #00H
                B0MOV   PCL, A          ; 跳到地址 0300H。
                …
```

➢ **例：PC = 0323H（PCH = 03H，PCL = 23H）。**

```
; PC = 0323H
                B0ADD   PCL, A          ; PCL = PCL + ACC，PCH 的值不变。
                JMP     A0POINT         ; ACC = 0，跳到 A0POINT。
                JMP     A1POINT         ; ACC = 1，跳到 A1POINT。
                JMP     A2POINT         ; ACC = 2，跳到 A2POINT。
                JMP     A3POINT         ; ACC = 3，跳到 A3POINT。
                …
                …
```

2.1.4.7 Y, Z 寄存器

8 位缓存器 Y，Z 的主要用途如下：

- 普通工作寄存器；
- RAM 数据寻址指针@YZ；
- 配合指令 MOVC 对 ROM 数据进行查表。

084H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Y	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

083H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Z	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

➢ **例：用 Y、Z 作为数据指针，访问 bank0 中 025H 处的内容。**

```
        B0MOV   Y, #00H         ; Y 指向 RAM bank 0。
        B0MOV   Z, #25H         ; Z 指向 25H。
        B0MOV   A, @YZ          ; 数据送入 ACC。
```

➢ **例：利用数据指针@YZ 对 RAM 数据清零。**

```
        B0MOV   Y, #0           ; Y = 0，指向 bank 0。
        B0MOV   Z, #07FH        ; Z = 7FH，RAM 区的最后单元。

CLR_YZ_BUF:
        CLR     @YZ             ; @YZ 清零。

        DECMS   Z               ;
        JMP     CLR_YZ_BUF      ; 不为零。

        CLR     @YZ
END_CLR:                        ;
        …
```

2.1.4.8 R 寄存器

8 位缓存器 R 主要有以下两个功能：

- **作为工作寄存器使用；**
- **存储执行查表指令后的高字节数据。**

（执行 MOVC 指令，指定 ROM 单元的高字节数据会被存入 R 寄存器而低字节数据则存入 ACC。）

082H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

✸ **注：关于 R 寄存器的应用，可参考“查表”的相关章节。**

2.2 寻址模式

2.2.1 立即寻址模式

将立即数送入 ACC 或指定的 RAM 单元。

- **例：立即数 12H 送入 ACC。**

```
MOV     A, #12H
```

- **例：立即数 12H 送入寄存器 R。**

```
B0MOV   R, #12H
```

注：立即数寻址中，指定的 RAM 单元必须是 80H~87H 的工作寄存器。

2.2.2 直接寻址模式

通过 ACC 对 RAM 单元数据进行操作。

- **例：地址 12H 处的内容送入 ACC。**

```
B0MOV   A, 12H
```

- **例：ACC 中数据写入 RAM 的 12H 单元。**

```
B0MOV   12H, A
```

2.2.3 间接寻址模式

通过指针寄存器（Y/Z）访问 RAM 数据。

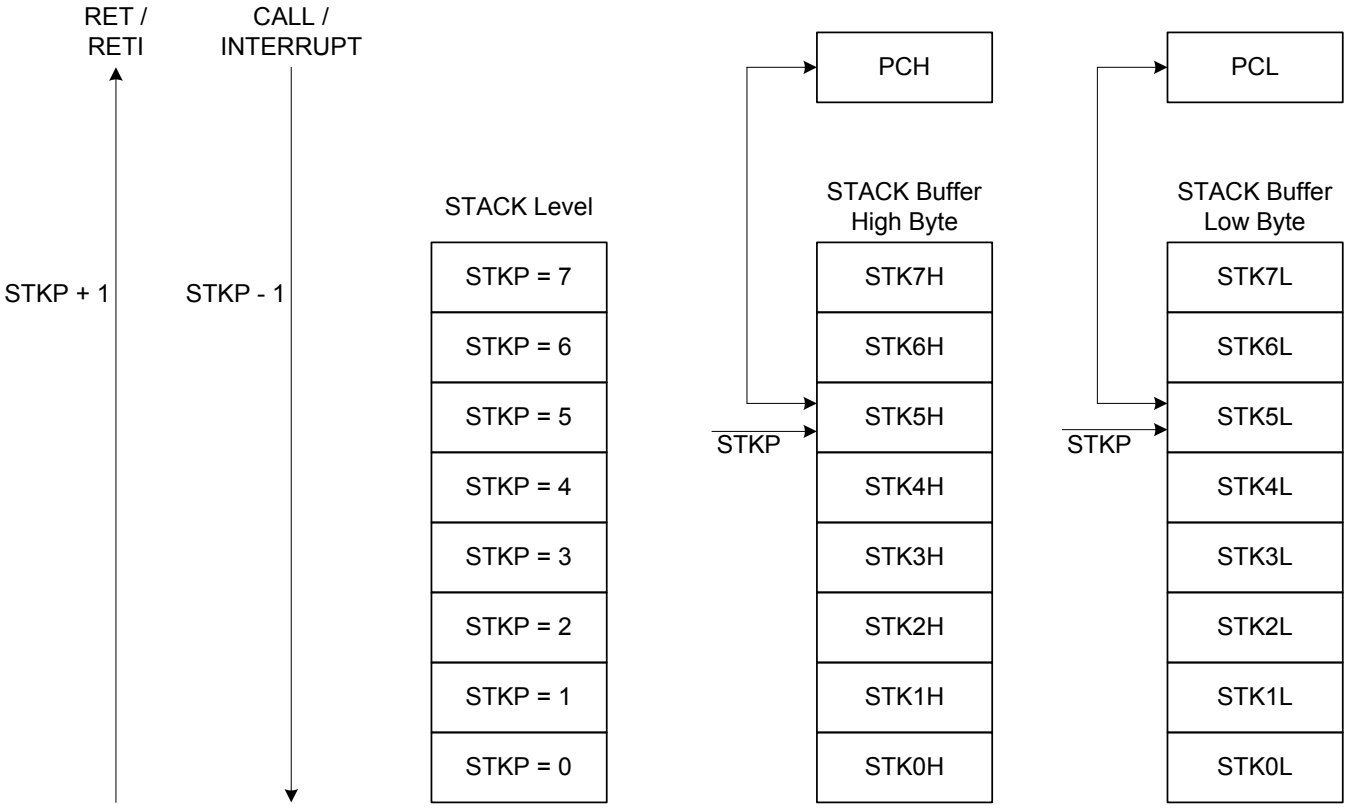
- **例：用 @YZ 实现间接寻址。**

```
B0MOV   Y, #0       ; Y 清零以寻址 RAM bank 0。
B0MOV   Z, #12H     ; 设定寄存器地址。
B0MOV   A, @YZ
```

2.3 堆栈操作

2.3.1 概述

SN8F2280 系列的堆栈缓存器共 8 层，程序进入中断或执行 CALL 指令时，用于存储程序计数器 PC 的值。寄存器 STKP 为堆栈指针，指向堆栈缓存器顶层，STKnH 和 STKnL 分别是各堆栈缓存器高、低字节。



2.3.2 堆栈寄存器

堆栈指针 STKP 是一个 3 位寄存器，存放被访问的堆栈单元地址，14 位数据存储器 STKnH 和 STKnL 用于暂存堆栈数据。以上寄存器都位于 bank 0。

使用入栈指令 PUSH 和出栈指令 POP 可对堆栈缓存器进行操作。堆栈操作遵循后进先出（LIFO）的原则，入栈时堆栈指针 STKP 的值减 1，出栈时 STKP 的值加 1，这样，STKP 总是指向堆栈缓存器顶层单元。

系统进入中断或执行 CALL 指令之前，程序计数器 PC 的值被存入堆栈缓存器中进行入栈保护。

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0
读/写	R/W	-	-	-	-	R/W	R/W	R/W
复位后	0	-	-	-	-	1	1	1

Bit[2:0] **STKPBn**：堆栈指针（n = 0 ~ 2）。

Bit 7 **GIE**：全局中断控制位。
0 = 禁止；
1 = 使能。

➢ **例：系统复位时，堆栈指针寄存器内容为默认值，但强烈建议在程序初始部分重新设定，如下面所示：**

```
MOV     A, #00000111B
B0MOV   STKP, A
```

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnH	-	-	SnPC13	SnPC12	SnPC11	SnPC10	SnPC9	SnPC8
读/写	-	-	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	0	0	0	0	0	0

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnL	SnPC7	SnPC6	SnPC5	SnPC4	SnPC3	SnPC2	SnPC1	SnPC0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

STKn = STKnH , STKnL (n = 7 ~ 0)

2.3.3 堆栈操作举例

执行程序调用指令 CALL 和响应中断服务时，堆栈指针 STKP 的值减 1，指针指向下一个堆栈缓存器。同时，对程序计数器 PC 的内容进行入栈保存。

堆栈层数	STKP 寄存器			堆栈缓存器		说明
	STKPB2	STKPB1	STKPB0	高字节	低字节	
0	1	1	1	Free	Free	-
1	1	1	0	STK0H	STK0L	-
2	1	0	1	STK1H	STK1L	-
3	1	0	0	STK2H	STK2L	-
4	0	1	1	STK3H	STK3L	-
5	0	1	0	STK4H	STK4L	-
6	0	0	1	STK5H	STK5L	-
7	0	0	0	STK6H	STK6L	-
8	1	1	1	STK7H	STK7L	-
> 8	1	1	0	-	-	**堆栈溢出，出错**

对应每个入栈操作，都有一个出栈操作来恢复程序计数器PC的值。RETI指令用于中断服务程序中，RET用于子程序调用。出栈时，STKP加1并指向下一个空闲堆栈缓存器。堆栈恢复操作如下表所示：

堆栈层数	STKP 寄存器			堆栈缓存器		说明
	STKPB2	STKPB1	STKPB0	高字节	低字节	
8	1	1	1	STK7H	STK7L	-
7	0	0	0	STK6H	STK6L	-
6	0	0	1	STK5H	STK5L	-
5	0	1	0	STK4H	STK4L	-
4	0	1	1	STK3H	STK3L	-
3	1	0	0	STK2H	STK2L	-
2	1	0	1	STK1H	STK1L	-
1	1	1	0	STK0H	STK0L	-
0	1	1	1	Free	Free	-

3 复位

3.1 概述

SN8F2280 系列单片机有以下几种复位方式：

- 上电复位；
- 看门狗复位；
- 掉电复位；
- 外部复位（仅支持外部复位引脚使能的状态）。

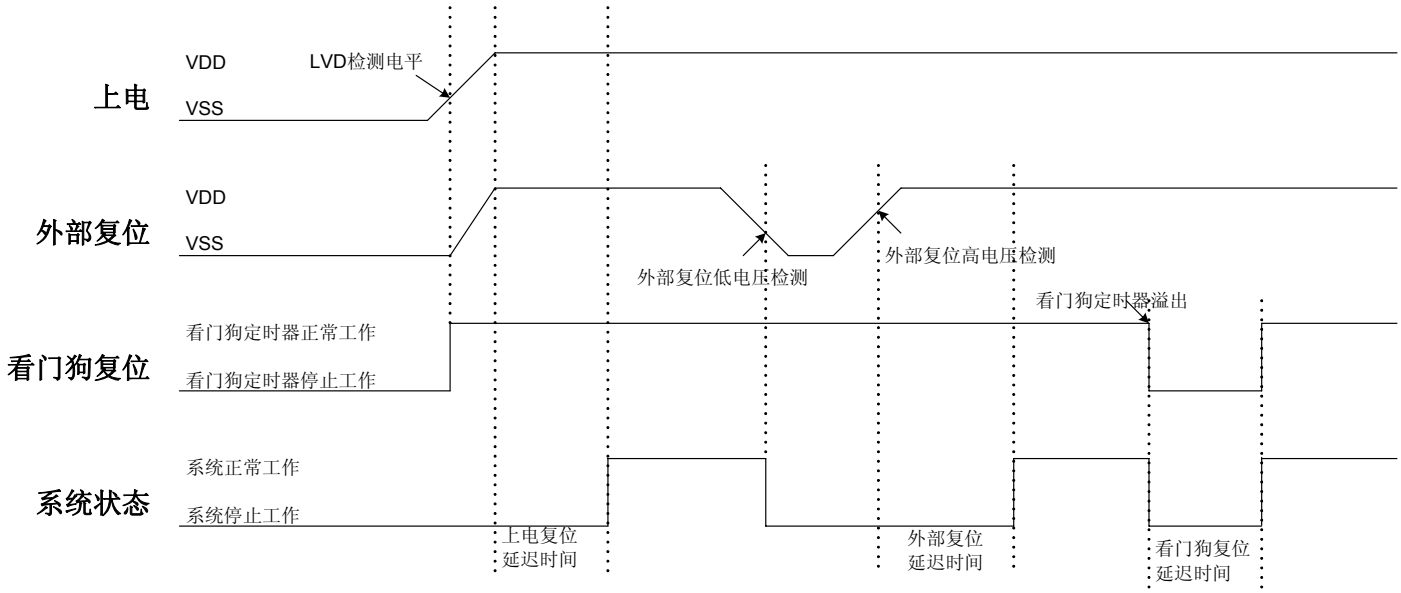
上述任一种复位发生时，所有的系统寄存器恢复默认状态，程序停止运行，同时程序计数器 PC 清零。复位结束后，系统从向量 0000H 处重新开始运行。PFLAG 寄存器的 NT0 和 NPD 两个标志位能够给出系统复位状态的信息。用户可以编程控制 NT0 和 NPD，从而控制系统的运行路径。

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	-	-	-	C	DC	Z
读/写	R/W	R/W	-	-	-	R/W	R/W	R/W
复位后	-	-	-	-	-	0	0	0

Bit [7:6] **NT0, NPD**：复位状态标志。

NT0	NPD	条件	说明
0	0	看门狗复位	看门狗定时器溢出
0	1	保留	-
1	0	上电复位和 LVD 复位	电源电压低于 LVD 检测电平电压
1	1	外部复位	外部复位引脚检测到低电平

任何一种复位方式都需要一定的复位时间，系统为上电复位提供了完整的复位程序。对于不同的晶振类型，系统需要的复位时间也有差异。因此，VDD 的上升速度和各振荡器的启动时间都不确定。RC 振荡器的启动时间非常之短，晶体振荡器的启动时间则相对较长。对于终端应用，用户必须注意主机端对上电复位时间的要求。下面给出了复位时序图。



3.2 上电复位

上电复位通常都与 LVD 密切相关。系统上电是一个逐渐上升的过程，需要经过一段时间才能达到正常的电压值。上电复位的时序如下：

- **上电：**系统检测到电源电压直到电压稳定；
- **外部复位（开启外部复位引脚的情况下）：**系统检查外部复位引脚的状态。如果外部复位引脚不为高电平，则系统保持复位状态；
- **系统初始化：**初始化所有的系统寄存器；
- **振荡器起振：**振荡器正常工作并提供系统时钟；
- **执行程序：**上电复位结束，程序从 ORG 0 开始执行。

3.3 看门狗复位

看门狗复位是系统的一种保护设置。在正常状态下，由程序将看门狗定时器清零。若出错，系统处于未知状态，看门狗定时器溢出，此时系统复位。看门狗复位后，系统重启进入正常状态。看门狗复位的时序如下：

- **看门狗定时器状态：**系统检测看门狗定时器是否溢出，若溢出，则系统复位；
- **系统初始化：**初始化所有的系统寄存器；
- **振荡器起振：**振荡器正常工作并提供系统时钟；
- **执行程序：**上电复位结束，程序从 ORG 0 开始执行。

看门狗定时器应用时需注意：

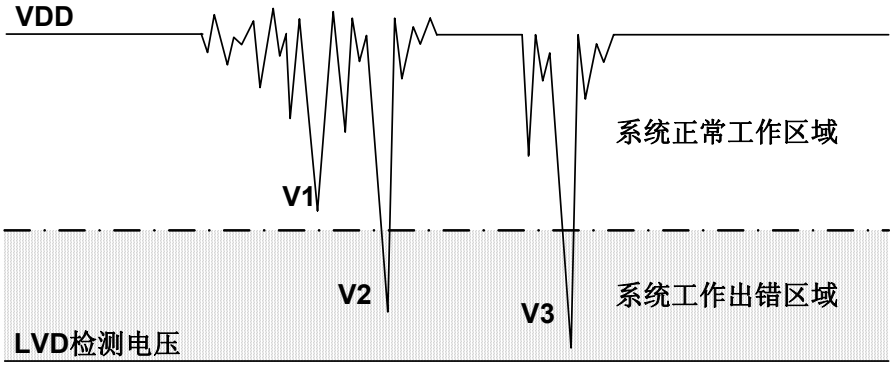
- 清看门狗定时器之前，请先检查 I/O 口的状态和 RAM 数据；
- 不能在中断里清看门狗定时器，否则无法起到侦测程序跑飞的目的；
- 程序中应该只有一个清看门狗的动作，这种架构能够最大限度的发挥看门狗的保护功能。

✸ **注：关于看门狗定时器的详细内容，请参阅“看门狗定时器有关章节。**

3.4 掉电复位

3.4.1 概述

掉电复位针对外部因素引起的系统电压跌落情形（例如：干扰或外部负载的变化），掉电复位可能会引起系统工作状态不正常或程序执行错误。



掉电复位示意图

电压跌落可能会进入系统死区。系统死区意味着电源不能满足系统的最小工作电压要求。上图是一个典型的掉电复位示意图。图中，VDD 受到严重的干扰，电压值降的非常低。虚线以上区域系统正常工作，在虚线以下的区域内，系统进入未知的工作状态，这个区域称作死区。当 VDD 跌至 V1 时，系统仍处于正常状态；当 VDD 跌至 V2 和 V3 时，系统进入死区，则容易导致出错。以下情况系统可能进入死区：

DC 运用中：

DC 运用中一般都采用电池供电，当电池电压过低或单片机驱动负载时，系统电压可能跌落并进入死区。这时，电源不会进一步下降到 LVD 检测电压，因此系统维持在死区。

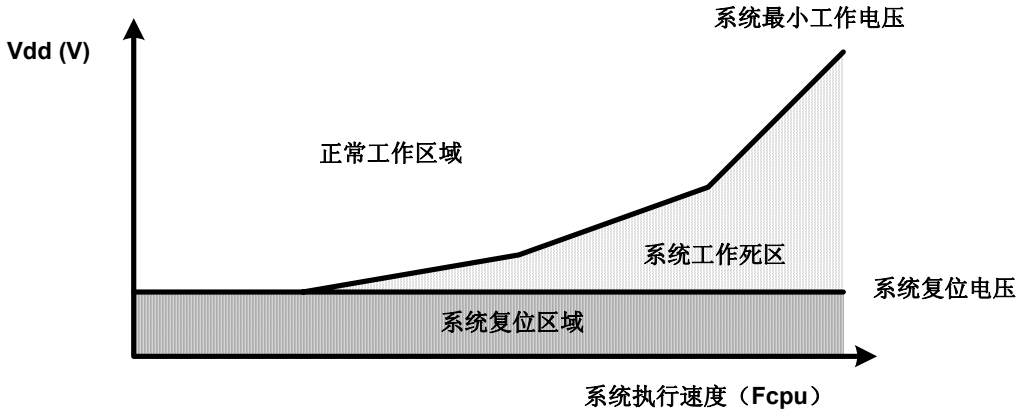
AC 运用中：

系统采用 AC 供电时，DC 电压值受 AC 电源中的噪声影响。当外部负载过高，如驱动马达时，负载动作产生的干扰也影响到 DC 电源。VDD 若由于受到干扰而跌落至最低工作电压以下时，则系统将有可能进入不稳定工作状态。

在 AC 运用中，系统上、下电时间都较长。其中，上电时序保护使得系统正常上电，但下电过程却和 DC 运用中情形类似，AC 电源关断后，VDD 电压在缓慢下降的过程中易进入死区。

3.4.2 系统工作电压

为了改善系统掉电复位的性能，首先必须明确系统具有的最低工作电压值。系统最低工作电压与系统执行速度有关，不同的执行速度下最低工作电压值也不同。



如上图所示，系统正常工作电压区域一般高于系统复位电压，同时复位电压由低电压检测（LVD）电平决定。当系统执行速度提高时，系统最低工作电压也相应提高，但由于系统复位电压是固定的，因此在系统最低工作电压与系统复位电压之间就会出现一个电压区域，系统不能正常工作，也不会复位，这个区域即为死区。

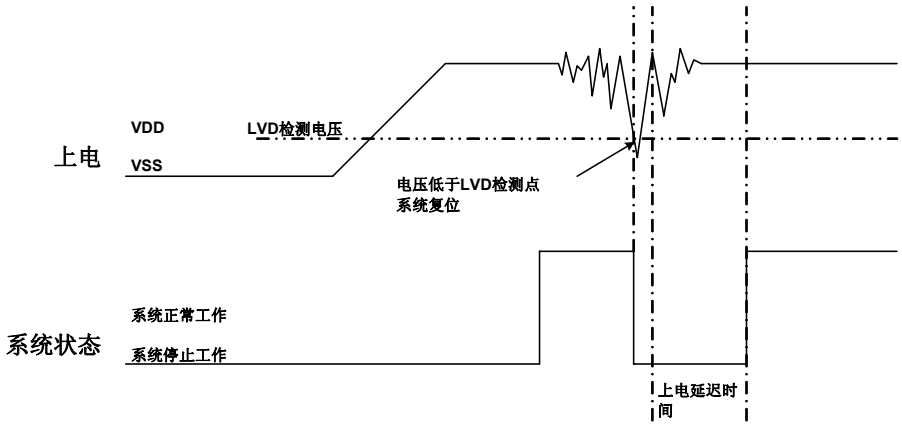
3.4.3 掉电复位性能改进

如何改善系统掉电复位性能，有以下几点建议：

- **LVD 复位；**
- **看门狗复位；**
- **降低系统工作速度；**
- **采用外部复位电路（稳压二极管复位电路，电压偏移复位电路，外部 IC 复位电路）。**

✸ **注：“稳压二极管复位电路”、“电压偏移复位电路”和“外部 IC 复位电路”能够完全避免掉电复位出错；**

LVD 复位：



低电压检测（LVD）是 SONiX 8 位单片机内置的掉电复位保护装置，当 VDD 跌落并低于 LVD 检测电压值时，LVD 被触发，系统复位。不同的单片机有不同的 LVD 检测电平，LVD 检测电平值仅为一个电压点，并不能覆盖所有死区范围。因此采用 LVD 依赖于系统要求和环境状况。电源变化较大时，LVD 能够起到保护作用，如果电源变化触发 LVD，系统工作仍出错，那么 LVD 就不能起到保护作用，就需要采用其它复位方法。

看门狗复位：

看门狗定时器用于保证系统正常工作。通常，会在主程序中将看门狗定时器清零，但不要在多个分支程序中清看门狗。若程序正常运行，看门狗不会复位。当系统进入死区或程序运行出错的时候，看门狗定时器继续计数直至溢出，系统复位。

如果看门狗复位后电源仍处于死区，则系统复位失败，保持复位状态，直到系统工作状态恢复到正常值。

降低系统工作速度：

系统工作速度越快最低工作电压值越高，从而加大工作死区的范围，因此降低系统工作速度不失为降低系统进入死区几率的有效措施。所以，可选择合适的工作速度以避免系统进入死区，这个方法需要调整整个程序使其满足系统要求。

附加外部复位电路：

外部复位也能够完全改善掉电复位性能。有三种外部复位方式可改善掉电复位性能：稳压二极管复位电路，电压偏移复位电路和外部 IC 复位。它们都采用外部复位信号控制单片机可靠复位。

3.5 外部复位

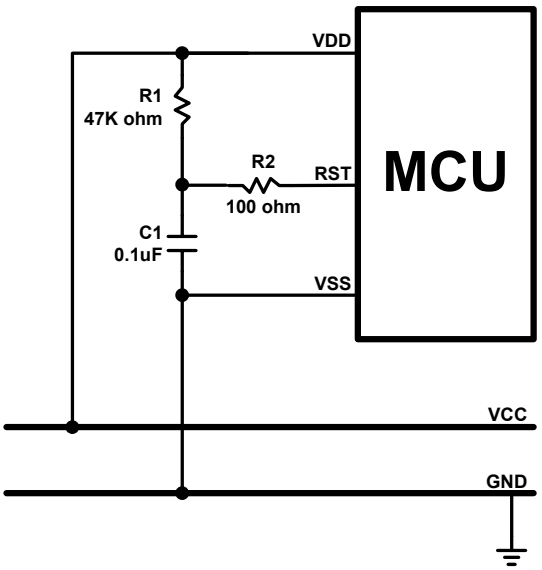
外部复位功能由编译选项“Reset_Pin”控制。将该编译选项置为“Reset”，可使能外部复位功能。外部复位引脚为施密特触发结构，低电平有效。复位引脚处于高电平时，系统正常运行。当复位引脚输入低电平信号时，系统复位。外部复位操作在上电和正常工作模式时有效。需要注意的是，在系统上电完成后，外部复位引脚必须输入高电平，否则系统将一直保持在复位状态。外部复位的时序如下：

- **外部复位（当且仅当外部复位引脚为使能状态）：**系统检测复位引脚的状态，如果复位引脚不为高电平，则系统会一直保持在复位状态，直到外部复位结束；
- **系统初始化：**所有的系统寄存器被置为初始状态；
- **振荡器开始工作：**振荡器开始提供系统时钟；
- **执行程序：**上电结束，程序开始运行。

外部复位可以在上电过程中使系统复位。良好的外部复位电路可以保护系统以免进入未知的工作状态，如 AC 应用中的掉电复位等。

3.6 外部复位电路

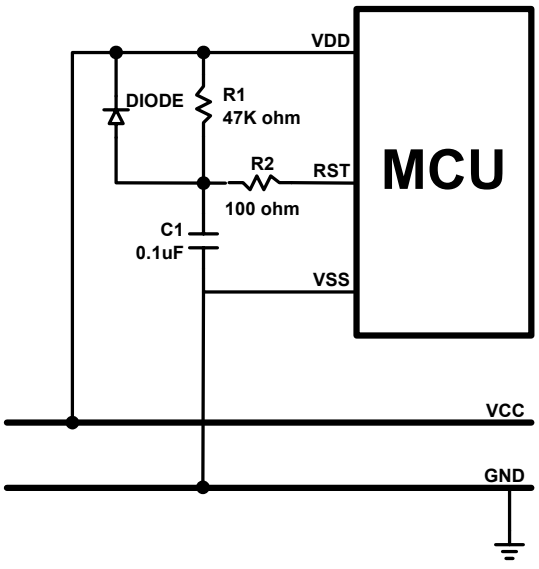
3.6.1 基本 RC 复位电路



上图为一个由电阻 R1 和电容 C1 组成的基本 RC 复位电路，它在系统上电的过程中能够为复位引脚提供一个缓慢上升的复位信号。这个复位信号的上升速度低于 VDD 的上电速度，为系统提供合理的复位时序，当复位引脚检测到高电平时，系统复位结束，进入正常工作状态。

✸ **注：此 RC 复位电路不能解决非正常上电和掉电复位问题。**

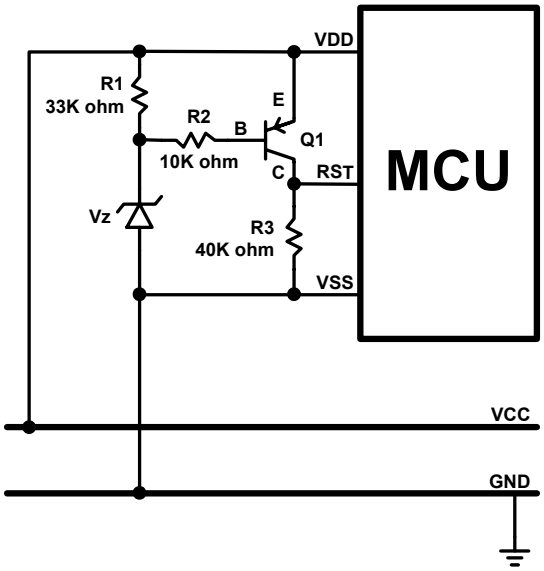
3.6.2 二极管& RC 复位电路



上图中，R1 和 C1 同样是为复位引脚提供输入信号。对于电源异常情况，二极管正向导通使 C1 快速放电并与 VDD 保持一致，避免复位引脚持续高电平、系统无法正常复位。

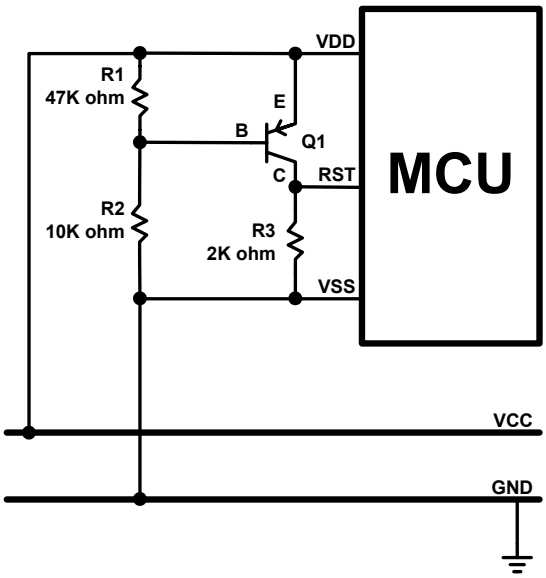
✸ **注：“基本 RC 复位电路”和“二极管及 RC 复位电路”中的电阻 R2 都是必不可少的限流电阻，以避免复位引脚 ESD（Electrostatic Discharge）或 EOS（Electrical Over-stress）击穿。**

3.6.3 稳压二极管复位电路



稳压二极管复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。如上图电路中，利用稳压管的击穿电压作为电路复位检测值，当 VDD 高于“Vz + 0.7V”时，三极管集电极输出高电平，单片机正常工作；当 VDD 低于“Vz + 0.7V”时，三极管集电极输出低电平，单片机复位。稳压管规格不同则电路复位检测值不同，根据电路的要求选择合适的二极管。

3.6.4 电压偏移复位电路

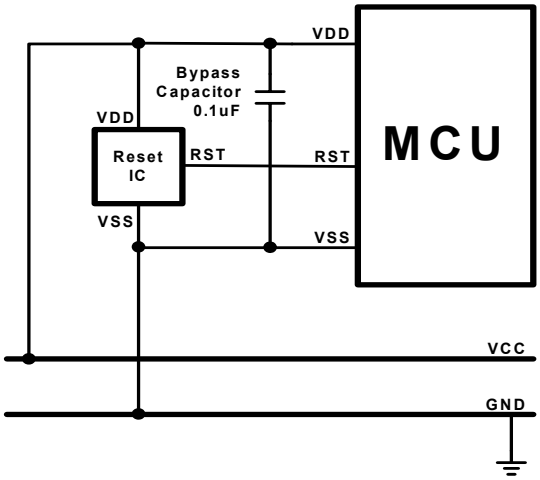


电压偏置复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。与稳压二极管复位电路相比，这种复位电路的检测电压值的精确度有所降低。电路中，R1 和 R2 构成分压电路，当 VDD 高于和等于分压值“0.7V x（R1 + R2）/ R1”时，三极管集电极 C 输出高电平，单片机正常工作；VDD 低于“0.7V x（R1 + R2）/ R1”时，集电极 C 输出低电平，单片机复位。

对于不同应用需求，选择适当的分压电阻。单片机复位引脚上电压的变化与 VDD 电压变化之间的差值为 0.7V。如果 VDD 跌落并低于复位引脚复位检测值，那么系统将被复位。如果希望提升电路复位电平，可将分压电阻设置为 R2>R1，并选择 VDD 与集电极之间的结电压高于 0.7V。分压电阻 R1 和 R2 在电路中要耗电，此处的功耗必须计入整个系统的功耗中。

✸ **注：在电源不稳定或掉电复位的情况下。“稳压二极管复位电路”和“偏压复位电路”能够保护电路在电压跌落时避免系统出错。当电压跌落至低于复位检测值时，系统将被复位。从而保证系统正常工作。**

3.6.5 外部 IC 复位电路



外部 IC 复位可以增强 MCU 复位的可靠性。

4 系统时钟

4.1 概述

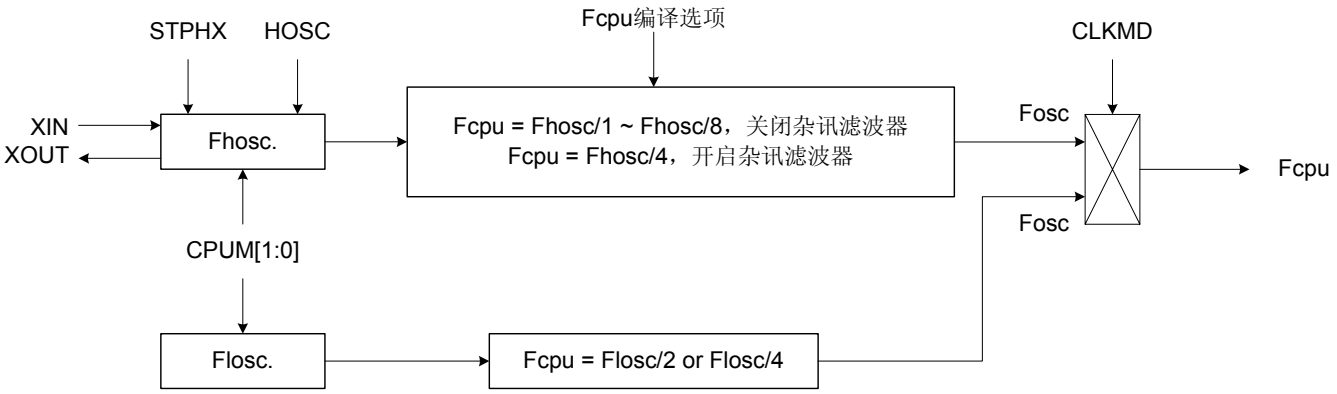
SN8F2280 系列单片机内带双时钟系统：高速时钟和低速时钟。高速时钟由外部振荡电路和内置 PLL 电路提供。低速时钟则由内置低速 RC 振荡电路（ILRC 12KHZ）提供。

高低速时钟都可以作为系统时钟（Fosc），当系统工作在低速模式下时，时钟信号 2 分频或者 4 分频之后作为系统指令周期（Fcpu）。

- **普通模式（高速时钟）：Fcpu = Fhosc / N**，N = 1 ~ 8，N 的值由 Fcpu 编译选项决定。
- **低速模式（低速时钟）：Fcpu = Flosc/4**，N=2、4，N 的值由编译选择决定。

在干扰较严重的运行条件下，SONiX 提供的杂讯滤波器能够对外部干扰进行隔离以保护系统的正常工作。但在杂讯滤波器有效时，高速时钟的 Fcpu 被限制为 **Fcpu＝Fhosc/4**。

4.2 系统时钟框图



- HOSC：High_Clk 编译选项。
- Fhosc：外部高速时钟频率。
- Flosc：内部低速 RC 时钟频率（典型值：12KHz）。
- Fosc：系统时钟频率。
- Fcpu：指令执行频率。

4.3 OSCM 寄存器

寄存器 OSCM 控制振荡器的状态和系统模式。

0CAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OSCM	-	-	-	CPUM1	CPUM0	CLKMD	STPHX	-
读/写	-	-	-	R/W	R/W	R/W	R/W	-
复位后	-	-	-	0	0	0	0	-

Bit 1 **STPHX**：外部高速振荡器控制位。
0 = 运行；
1 = 停止。内部低速 RC 振荡器仍然运行。

Bit 2 **CLKMD**：高/低速时钟模式控制位。
0 = 普通（双时钟）模式，系统时钟来自高速时钟；
1 = 低速模式，系统时钟来自低速时钟。

Bit[4:3] **CPUM[1:0]** ：CPU 工作模式控制位。
00 = 普通模式；
01 = 睡眠模式；
10 = 绿色模式；
11 = 系统保留。

- **例：停止高速振荡器。**

```
B0BSET      FSTPHX          ; 停止外部高速振荡器
```

- **例：系统进入睡眠模式时，高速振荡器（外部振荡器和内部 PLL 电路）和内部低速振荡器都停止运行。**

```
B0BSET      FCPUM0
```

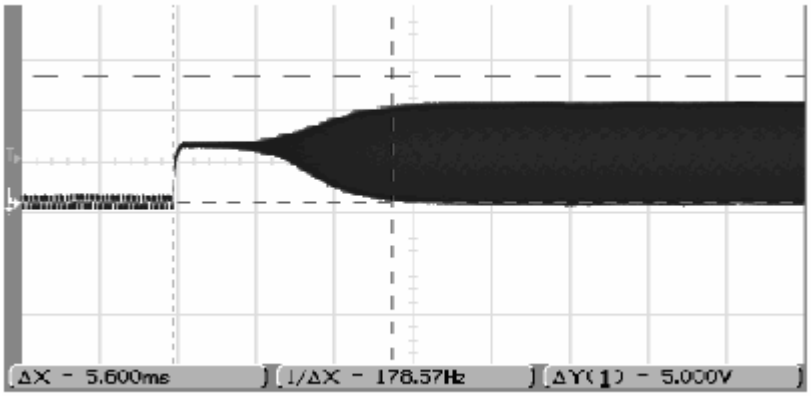
4.4 系统高速时钟

系统高速时钟有 PLL 电路提供。用户必须通过编译选项“Ext_OSC”将外部振荡器设为 6MHz X’tal、12MHz X’tal 和 16MHz X’tal，输入给内部 PLL 电路，然后再输出 12MHz 作为系统时钟（Fosc）。

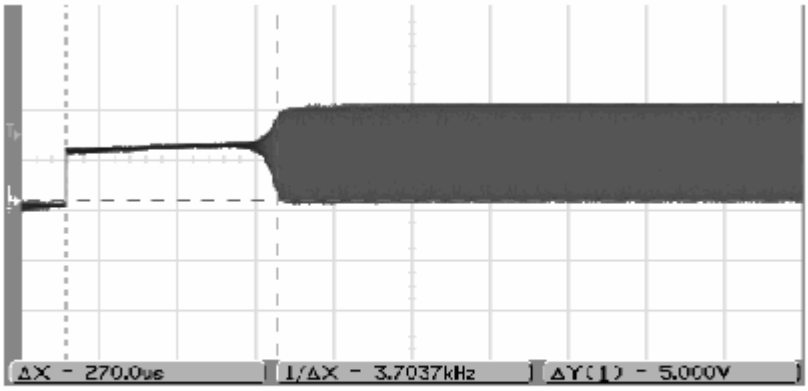
4.4.1 外部高速时钟

外部高速时钟包括 2 种模式：晶体/陶瓷和外部时钟信号。晶体和陶瓷振荡器的起振时间不相同，振荡器的起振时间决定复位时间的长短。

4MHz Crystal

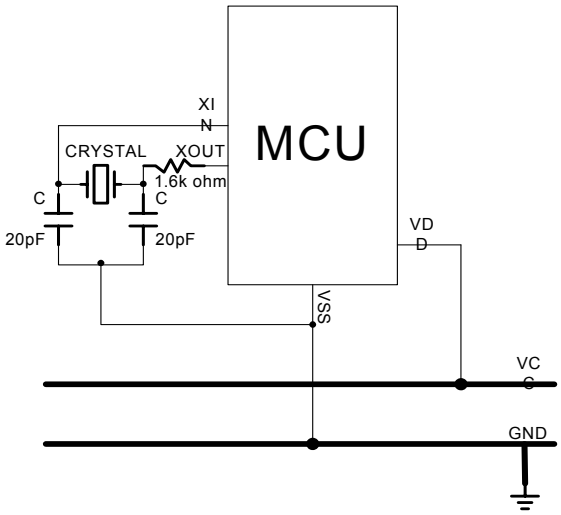


4MHz Ceramic



4.4.1.1 晶体/陶瓷振荡器

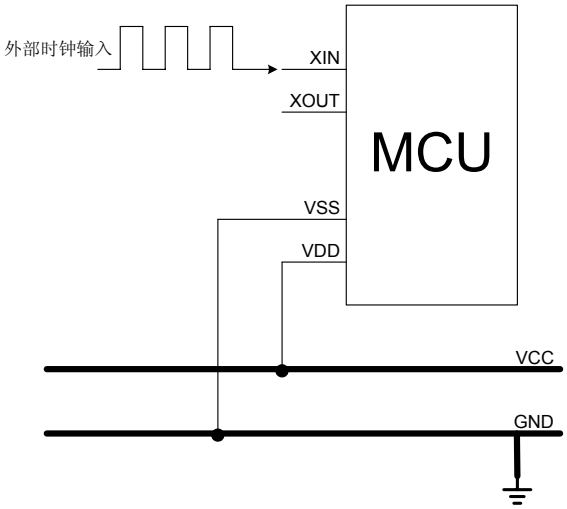
晶体振荡器由 XIN、XOUT 引脚驱动。



✸ **注：上图中， XIN/XOUT/VSS 引脚与石英振荡器以及电容 C 之间的距离越近越好。**

4.4.1.2 外部时钟信号

可通过编译选项“Ext_OSC”选项外部时钟信号作为时钟源。外部时钟信号由 XIN 引脚输入，XOUT 引脚则作为普通的 I/0 引脚。



✸ **注：外部振荡电路中的 GND 必须尽可能的接近单片机的 VSS 端口。**

4.5 系统低速时钟

系统低速时钟源来自内置的低速振荡器，采用 RC 振荡电路。低速时钟的输出频率受系统电压和环境温度的影响，通常输出 12KHz。

低速时钟可作为看门狗定时器的时钟源。由 CLKMD 控制系统低速工作模式。

☞ **Flosc = 内部低速 RC 振荡器（12KHz）。**
☞ **低速模式 Fcpu = Flosc / 4，**N=2、4，N 的值由编译选择决定。

系统工作在睡眠模式和绿色模式下禁止看门狗时，可以停止低速 RC 振荡器。如果系统工作频率为 12K 且禁止看门狗，那么这种情况下只有 12K 振荡器处于工作状态，系统功耗相应较低。

➢ **例：停止内部低速振荡器。**

```
B0BSET      FCPUM0
```

✸ **注：不可以单独停止内部低速时钟；由寄存器 OSCM 的位 CPUM0 和 CPUM1（12K，禁止看门狗）的设置决定内部低速时钟的状态。**

4.5.1 系统时钟测量

在设计过程中，用户可通过软件指令周期 Fcpu 对系统时钟速度进行测试。此测试仅适用于 RC 模式下。

➢ **例：外部振荡器的 Fcpu 指令周期测试。**

```
        B0BSET      P0M.0           ; P0.0 置为输出模式以输出 Fcpu 的触发信号。
@@:
        B0BSET      P0.0
        B0BCLR      P0.0
        JMP         @B
```

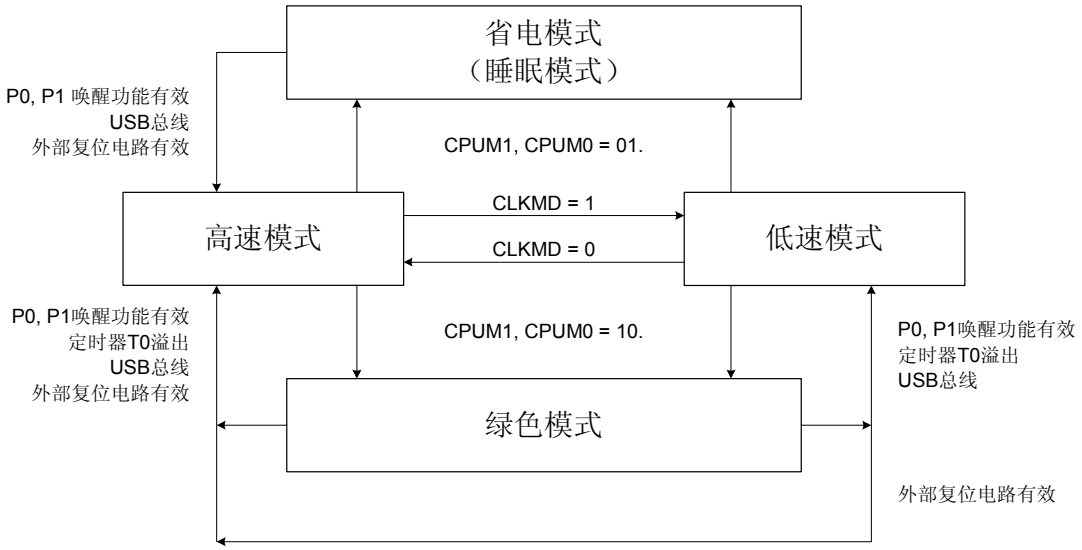
✸ **注：不能直接从 XIN 引脚测试 RC 振荡频率，因为探针的连接会影响测试的准确性。**

5 系统工作模式

5.1 概述

SN8F2280 系列单片机可在如下四种工作模式之间进行切换：

- 高速模式；
- 低速模式；
- 省电模式（睡眠模式）；
- 绿色模式。



系统工作模式切换示意图

系统工作模式说明

工作模式	高速模式	低速模式	绿色模式	睡眠模式	备注
HOSC	运行	STPHX	STPHX	停止	
ILRC	运行	运行	运行	停止	
CPU 指令	执行	执行	停止	停止	
定时器 T0	*有效	*有效	*有效	无线	* T0ENB=1 时有效
定时器 T1	*有效	*有效	无效	无效	* T1ENB=1 时有效
定时器 TC0	*有效	*有效	无效	无效	* TC0ENB=1 时有效
定时器 TC1	*有效	*有效	无效	无效	* TC1ENB=1 时有效
USB	运行	无效	无效	无效	* USBE=1 时有效
看门狗定时器	由 Watch_Dog 编译选项控制	由 Watch_Dog 编译选项控制	由 Watch_Dog 编译选项控制	由 Watch_Dog 编译选项控制	请参考编译选项说明
内部中断	全部有效	全部有效	T0	全部无效	
外部中断	全部有效	全部有效	全部有效	全部无效	
唤醒功能	-	-	P0、P1、T0、复位	P0、P1、复位	

- **HOSC**：高速时钟（Fosc=12MHz）
- **ILRC**：内部低速时钟（12KHz RC）

5.2 系统工作模式切换举例

➢ **例：高速/低速模式切换到睡眠模式。**

```
        B0BSET      FCPUM0
```

✸ **注：在睡眠模式下，只有具有唤醒功能的引脚和复位才能将系统唤醒进入高速模式。**

➢ **例：高速模式切换到低速模式。**

```
        B0BSET      FCLKMD      ; CLKMD = 1，系统进入低速模式。
        B0BSET      FSTPHX      ; 停止外部高速振荡器。
```

➢ **例：系统由低速模式切换到普通模式（外部高速振荡器仍然正常工作）。**

```
        B0BCLR      FCLKMD
```

➢ **例：系统由低速模式切换到普通模式（外部高速振荡器停止工作）。**

外部高速时钟停止工作，系统欲返回普通模式，须延迟至少 10ms 的时间等待外部时钟稳定。

```
        B0BCLR      FSTPHX      ; 开启外部高速振荡器。

        MOV         A, #10      ;内部 RC=12KHz（典型值）延迟。
        B0MOV       Z, A
@@:     DECMS       Z           ; 0.33ms X 30≈10ms 等待外部时钟稳定。
        JMP         @B
                                ;
        B0BCLR      FCLKMD      ; 系统返回到普通模式。
```

➢ **例：系统由普通/低速模式切换到绿色模式。**

```
        B0BSET      FCPUM1
```

✸ **注：绿色模式下如果禁止 T0 的唤醒功能，则只有具有唤醒功能的引脚和复位引脚可以将系统唤醒（具有唤醒功能的引脚将系统返回到上一个工作模式，复位引脚将系统返回到普通模式)。**

➢ **例：系统由普通/低速模式切换到绿色模式，并使能 T0 的唤醒功能。**

```
; 设置 T0 的唤醒功能。
        B0BCLR      FT0IEN      ; 禁止 T0 中断。
        B0BCLR      FT0ENB      ; 禁止 T0 定时器。
        MOV         A,#20H      ;
        B0MOV       T0M,A       ; 设置 T0 时钟 = Fcpu / 64。
        MOV         A,#74H
        B0MOV       T0C,A       ; 设置 T0C 初始值 = 74H（设置 T0 的中断间隔时间为 10ms）。
        B0BCLR      FT0IEN      ; 禁止 T0 的中断请求。
        B0BCLR      FT0IRQ      ; 清 T0 的中断请求。
        B0BSET      FT0ENB      ; 使能 T0 定时器。
; 进入绿色模式。
        B0BCLR      FCPUM0      ; 设置 CPUMx = 10。
        B0BSET      FCPUM1
```

✸ **注：开启绿色模式下 T0 的唤醒功能，具有唤醒功能的引脚和 T0 都可将系统唤醒。T0 的唤醒时间由程序控制。**

5.3 系统唤醒

5.3.1 概述

睡眠模式或绿色模式下，系统并不执行程序。唤醒触发信号可以将系统唤醒进入普通模式或低速模式。唤醒触发信号来自外部触发（P0、P1 的电平变化）、内部触发（T0 定时器溢出）和 USB 总线触发。

- 从睡眠模式唤醒后只能进入普通模式，且将其唤醒的触发只能是外部触发信号（P0、P1 电平变化）和 USB 触发信号。
- 系统由绿色模式唤醒返回到上一个工作模式（普通模式或低速模式），可以是外部触发信号（P0、P1 电平变化）、内部触发信号（T0 溢出）和 USB 总线触发信号。

5.3.2 唤醒时间

系统进入睡眠模式后，高速时钟振荡器停止运行。把系统从睡眠模式唤醒时，单片机需要等待 16384 个外部 6MHz 时钟周期的时间等待中断电路稳定工作，等待的这一段时间就称为唤醒时间。唤醒时间结束后，系统进入普通模式。

✸ **注：从绿色模式下唤醒系统不需要唤醒时间，因为系统时钟在绿色模式下仍然正常工作。**

唤醒时间的计算如下：

“16M_X’tal/12M_X’tal/6M_X’tal”模式:

*唤醒时间 = 1/Fosc * 16384 (sec) +高速时钟起振时间*

✸ **注：高速时钟的启动时间决定于 VDD 和振荡器的类型。**

➢ **例：在 16M_X’tal/12M_X’tal/6M_X’tal 和睡眠模式下，系统被唤醒进入普通模式。唤醒时间的计算如下：**

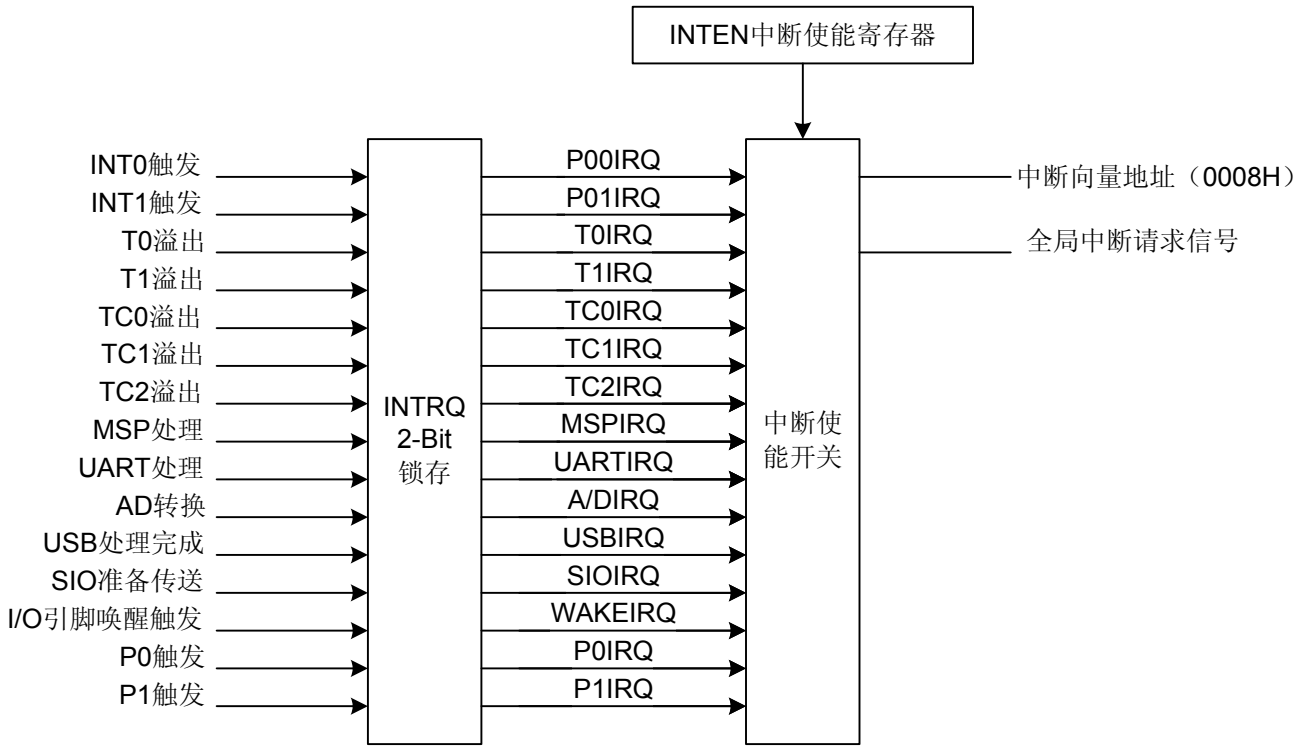
唤醒时间 = 1/Fosc * 16384 = 2.72ms（Fosc = 6MHz）

总的唤醒时间 = 2.72ms + 振荡器启动时间

6 中断

6.1 概述

SN8F2280 提供 15 个中断源：11 个内部中断（T0/T1/TC0/TC1/TC2/USB/SIO/MSP/UART/AD/唤醒中断）和 4 个外部中断（INT0/INT1/P0/P1）。外部中断可以将系统从睡眠模式中唤醒进入高速普通模式。一旦程序进入中断，寄存器 STKP 的位 GIE 将被硬件自动清零以避免再次响应其它中断。系统退出中断后，即执行完 RETI 指令后，硬件自动将 GIE 置“1”，以响应下一个中断。中断请求存放在寄存器 INTRQ 中。



✸ **注：程序响应中断时，位 GIE 必须处于有效状态。**

6.2 INTEN 中断使能寄存器

中断请求控制寄存器 INTEN 包括所有中断的使能控制位，即 INTEN 有效位中的每一位均控制单片机相应资源的中断功能。某一位被置为“1”，则使能该位的中断功能。一旦产生中断，堆栈加 1，程序转至中断向量处（0008H）。程序运行到指令 RETI 时，中断结束，系统退出中断服务。

0C7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN1	P1IEN	P0IEN	MSPIEN	UTRXIEN	UTTXIEN	TC2IEN	TC1IEN	TC0IEN
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **P1IEN**：P1 I/O 电平变换中断触发控制位。
0 = 禁止 P1 I/O 电平变换触发中断；
1 = 使能 P1 I/O 电平变换触发中断。

Bit 6 **P0IEN**：P0 I/O 电平变换中断触发控制位。
0 = 禁止 P0 I/O 电平变换触发中断；
1 = 使能 P0 I/O 电平变换触发中断。

Bit 5 **MSPIEN**：MSP 中断控制位。
0 = 禁止 MSP 中断；
1 = 使能 MSP 中断。

Bit 4 **UTRXIEN**：UART RX 中断控制位。
0 = 禁止 UART RX 中断；
1 = 使能 UART RX 中断。

Bit 3 **UTTXIEN**：UART TX 中断控制位。
0 = 禁止 UART TX 中断；
1 = 使能 UART TX 中断。

Bit 2 **TC2IEN**：TC2 中断控制位。
0 = 禁止 TC2 中断；
1 = 使能 TC2 中断。

Bit 1 **TC1IEN**：TC1 中断控制位。
0 = 禁止 TC1 中断；
1 = 使能 TC1 中断。

Bit 0 **TC0IEN**：TC0 中断控制位。
0 = 禁止 TC0 中断；
1 = 使能 TC0 中断。

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	ADCIEN	USBIEN	T1IEN	T0IEN	SIOIEN	WAKEIEN	P01IEN	P00IEN
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **ADCIEN**：ADC 中断控制位。
0 = 禁止 ADC 中断；
1 = 使能 ADC 中断。

Bit 6 **USBIEN**：USB 中断控制位。
0 = 禁止 USB 中断；
1 = 使能 USB 中断。

Bit 5 **T1IEN**：T1 中断控制位。
0 = 禁止 T1 中断；
1 = 使能 T1 中断。

Bit 4 **T0IEN**：T0 中断控制位。
0 = 禁止 T0 中断；
1 = 使能 T0 中断。

Bit 3 **SIOIEN**：SIO 中断控制位。
0 = 禁止 SIO 中断；
1 = 使能 SIO 中断。

Bit 2 **WAKEIEN**：I/O P0 & P1 WAKEUP 中断控制位。
0 = 禁止 WAKEUP 中断；
1 = 使能 WAKEUP 中断。

Bit 1 **P01IEN**：INT1 中断控制位。
0 = 禁止 INT1 中断；
1 = 使能 INT1 中断。

Bit 0 **P00IEN**：INT0 中断控制位。
0 = 禁止 INT0 中断；
1 = 使能 INT0 中断。

6.3 INTRQ 中断请求寄存器

中断请求寄存器 INTRQ 中存放各中断请求标志。一旦有中断请求发生，则 INTRQ 中对应位将被置“1”，该请求被响应后，程序应将该标志位清零。根据 INTRQ 的状态，程序判断是否有中断发生，并执行相应的中断服务。

0C6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ1	P1IRQ	P0IRQ	MSPIRQ	UTRXIRQ	UTTXIRQ	TC2IRQ	TC1IRQ	TC0IRQ
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **P1IRQ**：P1 I/O 电平变换中断请求控制位。
0 = 无中断请求；
1 = 请求中断。

Bit 6 **P0IRQ**：P0 I/O 电平变换中断请求控制位。
0 = 无中断请求；
1 = 请求中断。

Bit 5 **MSPIRQ**：MSP 中断请求控制位。
0 = 无中断请求；
1 = 请求中断。

Bit 4 **UTRXIRQ**：UART RX 中断请求控制位。
0 = 无中断请求；
1 = 请求中断。

Bit 3 **UTTXIRQ**：UART TX 中断请求控制位。
0 = 无中断请求；
1 = 请求中断。

Bit 2 **TC2IRQ**：TC2 中断请求控制位。
0 = 无中断请求；
1 = 请求中断。

Bit 1 **TC1IRQ**：TC1 中断请求控制位。
0 = 无中断请求；
1 = 请求中断。

Bit 0 **TC0IRQ**：TC0 中断请求控制位。
0 = 无中断请求；
1 = 请求中断。

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ	ADCIRQ	USBIRQ	T1IRQ	T0IRQ	SIOIRQ	WAKEIRQ	P01IRQ	P00IRQ
读/写	RW	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **ADCIRQ**：ADC 中断请求控制位。
0 = 无中断请求；
1 = 请求中断。

Bit 6 **USBIRQ**：USB 中断请求控制位。
0 = 无中断请求；
1 = 请求中断。

Bit 5 **T1IRQ**：T1 中断请求控制位。
0 = 无中断请求；
1 = 请求中断。

Bit 4 **T0IRQ**：T0 中断请求控制位。
0 = 无中断请求；
1 = 请求中断。

Bit 3 **SIOIRQ**：SIO 中断请求控制位。
0 = 无中断请求；
1 = 请求中断。

Bit 2 **WAKEIRQ**：I/O P0 & P1 WAKEUP 中断请求控制位。
0 = 无中断请求；
1 = 请求中断。

Bit 1 **P01IRQ**：INT1 中断请求控制位。
0 = 无中断请求；
1 = 请求中断。

Bit 0 **P00IRQ**：INT0 中断请求控制位。
0 = 无中断请求；
1 = 请求中断。

6.4 全局中断控制位 GIE

只有当全局中断控制位 GIE 置“1”的时候程序才能响应中断请求。 一旦有中断发生，程序计数器（PC）指向中断向量地址（ORG8），堆栈层数加 1。

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0
读/写	R/W	-	-	-	-	R/W	R/W	R/W
复位后	0	-	-	-	-	1	1	1

Bit 7 **GIE**：全局中断控制位。
0 = 禁止全局中断；
1 = 使能全局中断。

➢ **例：设置全局中断控制位（GIE）。**

```
B0BSET      FGIE            ; 使能 GIE。
```

✸ **注：在所有中断中，GIE 都必须处于使能状态。**

6.5 PUSH, POP

有中断请求发生并被响应后，程序转至 0008H 执行中断子程序。响应中断之前，必须保存 ACC、PFLAG 的内容。芯片提供 PUSH 和 POP 指令进行入栈保存和出栈恢复，从而避免中断结束后可能的程序运行错误。

✸ **注：“PUSH”、“POP”指令仅对 ACC 和 PFLAG 作中断保护，而不包括 NT0 和 NPD。PUSH/POP 缓存器是唯一的且仅有一层。**

➢ **例：对 ACC 和 PAFLG 进行入栈保护。**

```
            ORG     0
            JMP     START

            ORG     8H
            JMP     INT_SERVICE

            ORG     10H
START:
            …

INT_SERVICE:
            PUSH                    ; 保存 ACC 和 PFLAG。
            …
            …
            POP                     ; 恢复 ACC 和 PFLAG。

            RETI                    ; 退出中断。
            …
            ENDP
```

6.6 INT0（P0.0）& INT1（P0.1）中断

INT0/INT1 被触发，则无论 P00IEN/P01IEN 处于何种状态，P00IRQ/P01IRQ 都会被置“1”。如果 P00IRQ/P01IRQ=1 且 P00IEN/P01IEN=1，系统响应该中断；如果 P00IRQ/P01IRQ=1 而 P00IEN/P01IEN=0，系统并不会执行中断服务。在处理多中断时尤其需要注意。

如果中断的触发方向和唤醒功能的触发方向是一样的，则在系统由 P0.0 从睡眠模式和绿色模式唤醒时，INT0/INT1 的中断请求（INT0IRQ/INT1IRQ）就会被锁定。系统会在唤醒后马上进入中断向量地址执行中断服务程序。

✸ **注：INT0 的中断请求被 P0.0 的唤醒触发功能锁定。**
✸ **注：INT1 的中断请求被 P0.1 的唤醒触发功能锁定。**

✸ **注：P0.0/P0.1 的中断触发边沿由 PEDGE 控制。**

0BFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDGE					P01G1	P01G0	P00G1	P00G0
读/写					R/W	R/W	R/W	R/W
复位后					1	0	1	0

Bit[1:0] **P00G[1:0]**：P0.0 中断触发控制位。
00 = 保留；
01 = 上升沿触发；
10 = 下降沿触发；
11 = 上升/下降沿触发（电平变换触发）。

Bit[3:2] **P01G[1:0]:** P0.1 中断触发控制位。
00 = 保留；
01 = 上升沿触发；
10 = 下降沿触发；
11 = 上升/下降沿触发（电平变换触发）。

➢ **例：INT0 中断请求设置，电平触发。**

```
        MOV     A, #03H
        B0MOV   PEDGE, A        ; 设置 INT0 为电平触发。

        B0BCLR  FP00IRQ         ; 清 INT0 中断请求标志。
        B0BSET  FP00IEN         ; 允许 INT0 中断。
        B0BSET  FGIE            ; 使能 GIE。
```

➢ **例：INT0 中断。**

```
        ORG     8H              ;
        JMP     INT_SERVICE
INT_SERVICE:

        …                       ; ACC 和 PFLAG 入栈保护。

        B0BTS1  FP00IRQ         ; 检查是否有 P00 中断请求标志。
        JMP     EXIT_INT        ; P00IRQ = 0，退出中断。

        B0BCLR  FP00IRQ         ; 清 P00IRQ。
        …                       ; INT0 中断服务程序。
        …
EXIT_INT:
        …                       ; ACC 和 PFLAG 出栈恢复。

        RETI                    ; 退出中断。
```

6.7 T0 中断

T0C 溢出时，无论 T0IEN 处于何种状态，T0IRQ 都会置“1”。若 T0IEN 和 T0IRQ 都置“1”，系统就会响应 T0 中断；若 T0IEN = 0，则无论 T0IRQ 是否置“1”，系统都不会响应 T0 中断。尤其需要注意多种中断下的情形。

➢ **例：设置 T0 中断。**

```
        B0BCLR      FT0IEN          ; 禁止 T0 中断。
        B0BCLR      FT0ENB          ; 关闭 T0 定时器。
        MOV         A, #20H         ;
        B0MOV       T0M, A          ; 设置 T0 时钟= Fcpu / 64。
        MOV         A, #74H         ; 初始化 T0C = 74H。
        B0MOV       T0C, A          ; 设置 T0 间隔时间= 10 ms。

        B0BCLR      FT0IRQ          ; T0IRQ 清零。
        B0BSET      FT0IEN          ; 使能 T0 中断。
        B0BSET      FT0ENB          ; 开启定时器 T0。

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：T0 中断服务程序。**

```
        ORG         8H
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; 保存 ACC 和 PFLAG。

        B0BTS1      FT0IRQ          ; 检查是否有 T0 中断请求标志。
        JMP         EXIT_INT        ; T0IRQ = 0，退出中断。

        B0BCLR      FT0IRQ          ; 清 T0IRQ。
        MOV         A, #74H
        B0MOV       T0C, A          ;
        …
        …
EXIT_INT:
        …                           ; 恢复 ACC 和 PFLAG。

        RETI                        ; 退出中断。
```

6.8 T1 中断

T1C 溢出时，无论 T1IEN 处于何种状态，T1IRQ 都会置“1”。若 T1IEN 和 T1IRQ 都置“1”，系统就会响应 T1 中断；若 T1IEN = 0，则无论 T1IRQ 是否置“1”，系统都不会响应 T1 中断。尤其需要注意多种中断下的情形。

➢ **例：设置 T1 中断。**

```
        B0BCLR      FT1IEN          ; 禁止 T1 中断。
        B0BCLR      FT1ENB          ; 关闭 T1 定时器。
        MOV         A, #00H         ;
        B0MOV       T1M, A          ; 设置 T1 时钟= Fcpu / 256。
        MOV         A, #0E5H        ; 设置 T1CL 初始值为 0E5H。
        B0MOV       T1CL, A
        MOV         A, #48H         ; 设置 T1CH 初始值为 48H。
        B0MOV       T1CH, A         ; 设置 T1 间隔时间为 1S。

        B0BCLR      FT1IRQ          ; 清 T1IRQ。
        B0BSET      FT1IEN          ; 使能 T1 中断。
        B0BSET      FT1ENB          ; 开启 T1 定时器。

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：T1 中断。**

```
        ORG         8               ; 中断向量。
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; 保存 ACC 和 PFLAG。

        B0BTS1      FT1IRQ          ; 检查是否有 T1 中断请求。
        JMP         EXIT_INT        ; T1IRQ = 0，退出中断。

        B0BCLR      FT1IRQ          ; 清 T1IRQ。
        MOV         A, #0E5H
        B0MOV       T1CL, A         ; 初始化 T1CL。
        MOV         A, #48H
        B0MOV       T1CH, A         ; 初始化 T1CH。
        …                           ; T1 中断程序。
        …
EXIT_INT:
        …                           ; 恢复 ACC 和 PFLAG。

        RETI                        ; 退出中断。
```

6.9 TC0 中断

TC0C 溢出时，无论 TC0IEN 处于何种状态，TC0IRQ 都会置“1”。若 TC0IEN 和 TC0IRQ 都置“1”，系统就会响应 TC0 中断；若 TC0IEN = 0，则无论 TC0IRQ 是否置“1”，系统都不会响应 TC0 中断。尤其需要注意多种中断下的情形。

➢ **例：设置 TC0 中断。**

```
        B0BCLR      FTC0IEN         ; 禁止 TC0 中断。
        B0BCLR      FTC0ENB         ; 关闭 TC0 定时器。
        MOV         A, #20H         ;
        B0MOV       TC0M, A         ; 设置 TC0 时钟= Fcpu / 64。
        MOV         A, #74H         ; 设置 TC0C 的初始值为 74H。
        B0MOV       TC0C, A         ; 设置 TC0 的间隔时间为 10ms。

        B0BCLR      FTC0IRQ         ; 清 TC0IRQ。
        B0BSET      FTC0IEN         ; 使能 TC0 中断。
        B0BSET      FTC0ENB         ; 开启 TC0 定时器。

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：TC0 中断。**

```
        ORG         8               ; 中断向量。
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; 保存 ACC 和 PFLAG。

        B0BTS1      FTC0IRQ         ; 检查是否有 TC0 中断请求。
        JMP         EXIT_INT        ; TC0IRQ = 0，退出中断。

        B0BCLR      FTC0IRQ         ; 清 TC0IRQ。
        MOV         A, #74H
        B0MOV       TC0C, A         ; 初始化 TC0C。
        …                           ; TC0 中断程序。
        …
EXIT_INT:
        …                           ; 恢复 ACC 和 PFLAG。

        RETI                        ; 退出中断。
```

6.10 TC1 中断

TC1C 溢出时，无论 TC1IEN 处于何种状态，TC1IRQ 都会置“1”。若 TC1IEN 和 TC1IRQ 都置“1”，系统就会响应 TC1 中断；若 TC1IEN = 0，则无论 TC1IRQ 是否置“1”，系统都不会响应 TC1 中断。尤其需要注意多种中断下的情形。

➢ **例：设置 TC1 中断。**

```
        B0BCLR      FTC1IEN         ; 禁止 TC1 中断。
        B0BCLR      FTC1ENB         ; 关闭 TC1 定时器。
        MOV         A, #20H         ;
        B0MOV       TC1M, A         ; 设置 TC1 时钟= Fcpu / 64
        MOV         A, #74H         ; 设置 TC1C 初始值为 74H。
        B0MOV       TC1C, A         ; 设置 TC1 间隔时间为 10ms。

        B0BCLR      FTC1IRQ         ; 清 TC1IRQ。
        B0BSET      FTC1IEN         ; 使能 TC1 中断。
        B0BSET      FTC1ENB         ; 开启 TC1 定时器。

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：TC1 中断。**

```
        ORG         8               ; 中断向量。
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; 保存 ACC 和 PFLAG。

        B0BTS1      FTC1IRQ         ; 检查是否有 TC1 中断请求。
        JMP         EXIT_INT        ; TC1IRQ = 0，退出中断。

        B0BCLR      FTC1IRQ         ; 清 TC1IRQ。
        MOV         A, #74H
        B0MOV       TC1C, A         ; 初始化 TC1C。
        …                           ; TC1 中断程序。
        …
EXIT_INT:
        …                           ; 恢复 ACC 和 PFLAG。

        RETI                        ; 退出中断。
```

6.11 TC2 中断

TC2C 溢出时，无论 TC2IEN 处于何种状态，TC2IRQ 都会置“1”。若 TC2IEN 和 TC2IRQ 都置“1”，系统就会响应 TC2 中断；若 TC2IEN = 0，则无论 TC2IRQ 是否置“1”，系统都不会响应 TC2 中断。尤其需要注意多种中断下的情形。

➢ **例：设置 TC2 中断。**

```
        B0BCLR      FTC2IEN         ; 禁止 TC2 中断。
        B0BCLR      FTC2ENB         ; 关闭 TC2 定时器。
        MOV         A, #20H         ;
        B0MOV       TC2M, A         ; 设置 TC2 时钟= Fcpu / 64
        MOV         A, #74H         ; 设置 TC2C 初始值为 74H。
        B0MOV       TC2C, A         ; 设置 TC2 间隔时间为 10ms。

        B0BCLR      FTC2IRQ         ; 清 TC2IRQ。
        B0BSET      FTC2IEN         ; 使能 TC2 中断。
        B0BSET      FTC2ENB         ; 开启 TC2 定时器。

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：TC2 中断。**

```
        ORG         8               ; 中断向量。
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; 保存 ACC 和 PFLAG。

        B0BTS1      FTC2IRQ         ; 检查是否有 TC2 中断请求。
        JMP         EXIT_INT        ; TC2IRQ = 0，退出中断。

        B0BCLR      FTC2IRQ         ; 清 TC2IRQ。
        MOV         A, #74H
        B0MOV       TC2C, A         ; 初始化 TC2C。
        …                           ; TC2 中断程序。
        …
EXIT_INT:
        …                           ; 恢复 ACC 和 PFLAG。

        RETI                        ; 退出中断。
```

6.12 USB 中断

USB 处理完成后，无论 USBIEN 处于何种状态，USBIRQ 都会置“1”。若 USBIEN 和 USBIRQ 都置“1”，系统就会响应 USB 中断；若 USBIEN = 0，则无论 USBIRQ 是否置“1”，系统都不会响应 USB 中断。尤其需要注意多种中断下的情形。

➢ **例：设置 USB 中断。**

```
        B0BCLR      FUSBIEN     ; 禁止 USB 中断。
        B0BCLR      FUSBIRQ     ; 清 USB 中断请求标志。
        B0BSET      FUSBIEN     ; 使能 USB 中断。

        …                       ; 初始化 USB。
        …                       ; USB 操作。

        B0BSET      FGIE        ; 使能 GIE。
```

➢ **例：USB 中断。**

```
        ORG         8           ; 中断向量。
        JMP         INT_SERVICE
INT_SERVICE:

        PUSH                    ; 保存 ACC 和 PFLAG。

        B0BTS1      FUSBIRQ     ; 检查是否有 USB 中断请求。
        JMP         EXIT_INT    ; USBIRQ = 0，退出中断。

        B0BCLR      FUSBIRQ     ; 清 USBIRQ。

        …                       ; USB 中断。
        …
EXIT_INT:
        POP                     ; 恢复 ACC 和 PFLAG。

        RETI                    ; 退出中断。
```

6.13 WAKEUP 中断

P1/P0 将系统从睡眠模式唤醒后，无论 WAKEIEN 处于何种状态，WAKEIRQ 都会置“1”。若 WAKEIEN 和 WAKEIRQ 都置“1”，系统就会响应 WAKE 中断；若 WAKEIEN = 0，则无论 WAKEIRQ 是否置“1”，系统都不会响应 WAKE 中断。尤其需要注意多种中断下的情形。

➢ **例：设置 WAKE 中断。**

```
        B0BCLR      FWAKEIEN        ; 禁止 WAKE 中断。
        B0BCLR      FWAKEIRQ        ; 清 WAKE 中断请求标志。
        B0BSET      FWAKEIEN        ; 使能 WAKE 中断。

        …                           ; 初始化 WAKEUP 引脚。
        …                           ; WAKEUP 操作。

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：WAKE 中断。**

```
        ORG         8               ; 中断向量。
        JMP         INT_SERVICE
INT_SERVICE:

        PUSH                        ; 保存 ACC 和 PFLAG。

        B0BTS1      FWAKEIRQ        ; 检查是否有 WAKE 中断请求。
        JMP         EXIT_INT        ; WAKEIRQ = 0，退出中断。

        B0BCLR      FWAKEIRQ        ; 清 WAKEIRQ。

        …                           ; WAKE 中断程序。
        …
EXIT_INT:
        POP                         ; 恢复 ACC 和 PFLAG。

        RETI                        ; 退出中断。
```

6.14 SIO 中断

SIO 成功传送后，无论 SIOIEN 处于何种状态，SIOIRQ 都会置“1”。若 SIOIEN 和 SIOIRQ 都置“1”，系统就会响应 SIO 中断；若 SIOIEN = 0，则无论 SIOIRQ 是否置“1”，系统都不会响应 SIO 中断。尤其需要注意多种中断下的情形。

➢ **例：设置 SIO 中断。**

```
        B0BSET      FSIOIEN     ; 使能 SIO 中断。
        B0BCLR      FSIOIRQ     ; 清 SIO 中断请求标志。
        B0BSET      FGIE        ; 使能 GIE。
```

➢ **例：SIO 中断，**

```
        ORG         8           ; 中断向量。
        JMP         INT_SERVICE
INT_SERVICE:
        …                       ; 保存 ACC 和 PFLAG。

        B0BTS1      FSIOIRQ     ; 检查是否有 SIO 中断请求。
        JMP         EXIT_INT    ; SIOIRQ = 0，退出中断。

        B0BCLR      FSIOIRQ     ; 清 SIOIRQ。
        …                       ; SIO 中断程序。
        …
EXIT_INT:
        …                       ; 恢复 ACC 和 PFLAG。

        RETI                    ; 退出中断。
```

6.15 多中断操作

在同一时刻，系统中可能出现多个中断请求。处理这种多中断情况时，必须预先对各中断请求设置不同的优先级别。中断请求标志 IRQ 由中断事件触发，当 IRQ 处于有效值“1”时，系统并不一定会响应该中断。各中断触发事件列表如下：

中断名称	*触发说明*
P00IRQ	P0.0 触发，由 PEDGE 控制。
T0IRQ	T0C 溢出。
T1IRQ	T1C 溢出。
USBIRQ	USB 处理完成。
WAEKIRQ	I/O P1&P0 唤醒系统。
SIOIRQ	SIO 处理完成。

多个中断同时发生时，需要注意的是：首先，必须预先设定好各中断的优先级。其次，利用 IEN 和 IRQ 控制系统是否响应该中断。在程序中，必须对中断控制位和中断请求标志进行检测。

➢ **例：多中断操作。**

```
                ORG         8                   ; 中断向量。
                JMP         INT_SERVICE
INT_SERVICE:
                …                               ; 保存 ACC 和 PFLAG。

INTP00CHK:                                      ; 检查是否有 INT0 中断请求。
                B0BTS1      FP00IEN             ; 检查是否使能 INT0 中断。
                JMP         INTT0CHK            ; 跳转到下一个中断。
                B0BTS0      FP00IRQ             ; 检查是否有 INT0 中断请求。
                JMP         INTP00
INTT0CHK:                                       ; 检查是否有 T0 中断请求。
                B0BTS1      FT0IEN              ; 检查是否使能 T0 中断。
                JMP         INTT1CHK            ; 停止到下一个中断。
                B0BTS0      FT0IRQ              ; 检查是否有 T0 中断请求。
                JMP         INTT0               ; 进入 T0 中断程序。
INTT1CHK:                                       ; 检查是否有 T1 中断请求。
                B0BTS1      FT1IEN              ; 检查是否使能 T1 中断。
                JMP         INTTC1CHK           ; 跳转到下一个中断。
                B0BTS0      FT1IRQ              ; 检查是否有 T1 中断请求。
                JMP         INTT1               ; 进入 T1 中断程序。
INTUSBCHK:                                      ; 检查是否有 USB 中断请求。
                B0BTS1      FUSBIEN             ; 检查是否使能 USB 中断。
                JMP         INTWAKECHK          ; 跳转到下一个中断。
                B0BTS0      FUSBIRQ             ; 检查是否有 USB 中断请求。
                JMP         INTUSB              ; 进入 USB 中断程序。
INTWAKECHK:                                     ; 检查是否有 WQKE 中断请求。
                B0BTS1      FWAKEIEN            ; 检查是否使能 WAKE 中断。
                JMP         INTSIOCHK           ; 跳转到下一个中断。
                B0BTS0      FWAKEIRQ            ; 检查是否有 WAKE 中断请求。
                JMP         INTWAKEUP           ; 进入 WAKE 中断程序。
INTSIOCHK:                                      ; 检查是否有 SIO 中断请求。
                B0BTS1      FSIOIEN             ; 检查是否使能 SIO 中断。
                JMP         INT_EXIT            ; 跳转到下一个中断。
                B0BTS0      FSIOIRQ             ; 检查是否有 SIO 中断请求。
                JMP         INTSIO              ; 进入 SIO 中断程序。
INT_EXIT:
                …                               ; 虎伏 ACC 和 PFLAG。

                RETI                            ; 退出中断。
```

7 I/O 端口

7.1 I/O 口模式

寄存器 PnM 的设置决定各 I/O 口的数据传送方向，所有的 I/O 口必须设置相应的输入、输出方向。

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1M	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0C2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2M	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0C4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4M	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	-	-	P55M	P54M	P53M	P52M	P51M	P50M
读/写	-	-	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	0	0	0	0	0	0

Bit[7:0] **PnM[7:0]**：Pn 模式控制位（n = 0~5）。
0 = 输入模式；
1 = 输出模式。

✸ **注： 可通过位控制指令 B0BSET、B0BCLR 设置模式寄存器。**

➢ **例：I/O 模式设置。**

```
CLR     P0M             ; 设置为输入模式。
CLR     P1M
CLR     P5M

MOV     A, #0FFH        ; 设置为输出模式。
B0MOV   P0M, A
B0MOV   P1M, A
B0MOV   P5M, A

B0BCLR  P1M.2           ; 设置为输入模式。

B0BSET  P1M.2           ; 设置为输出模式。
```

7.2 I/O 口上拉电阻

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	P07R	P06R	P05R	P00R	P03R	P02R	P01R	P00R
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	P17R	P16R	P15R	P16R	P13R	P12R	P11R	P10R
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

0E2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2UR	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

0E4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4UR	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

0E5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5UR	-	-	P55R	P54R	P53R	P52R	P51R	P50R
读/写	-	-	W	W	W	W	W	W
复位后	-	-	0	0	0	0	0	0

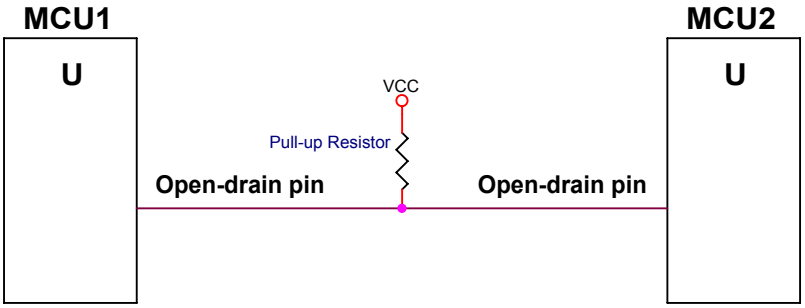
✸ **注：P0.4 是单向输入引脚，内置上拉电阻。**

➢ **例：I/O 口上拉电阻。**

```
MOV     A, #0FFH        ; 使能 P0、P1 和 P5 的上拉电阻。
B0MOV   P0UR, A         ;
B0MOV   P1UR, A
B0MOV   P5UR, A
```

7.3 I/O 口漏极开路寄存器

P1.0/P1.1/P0.6/P0.5 内置漏极开路功能，当使能 P1.0/P1.1/P0.6/P0.5 的漏极开路功能是，必须将 P1.1/P1.0/P0.6/P0.5 设置为输出模式。漏极开路功能的外部电路如下所示：



上图中的上拉电阻是必不可少的。

0E9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1OC	-	-	-	-	P06OC	P05OC	P11OC	P10OC
读/写	-	-	-	-	W	W	W	W
复位后	-	-	-	-	0	0	0	0

Bit [3:2] **P0nOC**：P0 漏极开路控制位。
0 = 禁止漏极开路功能；
1 = 使能漏极开路功能。

Bit [1:0] **P1nOC**：P1 漏极开路控制位。
0 = 禁止漏极开路功能；
1 = 使能漏极开路功能。

➢ **例：使能 P1.0 的漏极开路功能，输出高电平。**

```
B0BSET      P1.0          ; 设置 P1.0 为输出模式。

B0BSET      P10M          ;
MOV         A, #01H       ; 使能 P1.0 的漏极开路功能。
B0MOV       P1OC, A
```

✸ **注：P1OC 是只写寄存器，通过指令 MOV 设置 P10OC。**

➢ **例：禁止 P1.0 的漏极开路功能，输出低电平。**

```
MOV         A, #0         ; 禁止 P1.0 的漏极开路功能。
B0MOV       P1OC, A
```

✸ **注：禁止 P1.0 的漏极开路功能后，P1.0 返回到上一个 I/O 模式。**

7.4 I/O 口数据寄存器

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	P07	P06	P05	P04	P03	P02	P01	P00
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1	P17	P16	P15	P14	P13	P12	P11	P10
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0D2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2	P27	P26	P25	P24	P23	P22	P21	P20
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0D4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4	P47	P46	P45	P44	P43	P42	P41	P40
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	-	-	P55	P54	P53	P52	P51	P50
读/写	-	-	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	0	0	0	0	0	0

➢ **例：读取输入口的数据。**

```
B0MOV   A, P0     ; 读取 P0 口的数据。
B0MOV   A, P1     ; 读取 P1 口的数据。
B0MOV   A, P5     ; 读取 P5 口的数据。
```

➢ **例：写入数据到输出口。**

```
MOV     A, #0FFH  ; 写入数据 0FFH 到 P0、P1 和 P5。
B0MOV   P0, A
B0MOV   P1, A
B0MOV   P5, A
```

➢ **例：写入 1 位数据到输出口。**

```
B0BSET  P1.3      ; 设置 P1.3 和 P5.3 为 1。
B0BSET  P5.3

B0BCLR  P1.3      ; 清 P1.3 和 P5.3。
B0BCLR  P5.3
```

7.5 P1 唤醒功能寄存器

0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **P1nW**：P1 唤醒功能控制位。
0 = 禁止唤醒功能；
1 = 使能唤醒功能。

8 定时器

8.1 看门狗定时器

看门狗定时器 WDT 是一个 4 位二进制计数器，用于监控程序的正常执行。如果由于干扰，程序进入了未知状态，看门狗定时器溢出，系统复位。看门狗的工作模式由编译选项控制，其时钟源由内部低速振荡器（12KHz）提供。

看门狗溢出时间 = 8192 /内部低速振荡器周期（s）

VDD	内部低速 RC 频率	编译选项－低速设置	看门狗溢出时间
5V	12KHz	Flosc/2	341ms
5V	12KHz	Flosc/4	682ms

✸ **注：如果看门狗被置为“Always_On”模式，那么看门狗在睡眠模式和绿色模式下仍然运行。**

看门狗由 WDTR 寄存器清零，写入 5AH 到 WDTR 寄存器就可以将看门狗定时器清零。

0CCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WDTR	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

➢ **例：如下是对看门狗定时器的操作，在主程序开头对看门狗清零。**

```
Main:
        MOV     A,#5AH          ; 看门狗定时器清零。
        B0MOV   WDTR,A
        …
        CALL    SUB1
        CALL    SUB2
        …
        …
        …
        JMP     MAIN
```

看门狗定时器应用注意事项如下：

- 对看门狗清零之前，检查 I/O 口的状态和 RAM 的内容可增强程序的可靠性；
- 不能在中断中对看门狗清零，否则无法侦测到主程序跑飞的状况；
- 程序中应该只在主程序中有一次清看门狗的动作，这种架构能够最大限度的发挥看门狗的保护功能。

➢ **例：如下是对看门狗定时器的操作，在主程序开头对看门狗清零。**

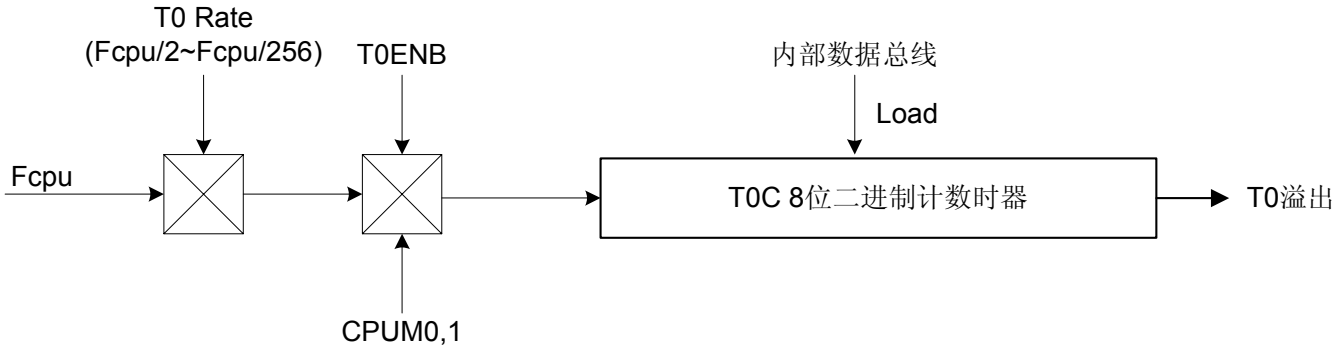
```
Main:
        …                       ; 检查 I/O 口状态。
        …                       ; 检查 RAM 状态。
Err:    JMP $                   ; I/O 或 RAM 出错，程序停止运行。
                                ; 等待看门狗定时器溢出，并复位。

Correct:                        ; I/O 或 RAM 处于正确状态，清看门狗定时器，执行程序。

        MOV     A,#5AH          ; 清看门狗定时器。
        B0MOV   WDTR,A
        …
        CALL    SUB1
        CALL    SUB2
        …
        …
        …
        JMP     MAIN
```

8.2 定时器 T0

8.2.1 概述

8 位二进制定时/计数器 T0 溢出（从 0FFH 到 00H）时，T0 继续计数并给出一个溢出信号触发 T0 中断请求。定时器 T0 的主要用途如下：

- **8 位可编程计数定时器：**根据选择的时钟频率周期性的产生中断请求；
- **绿色模式唤醒功能：**T0ENB = 1 时，T0 的溢出信号将系统从绿色模式下唤醒。



8.2.2 T0M 模式寄存器

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	-	-	-	
读/写	R/W	R/W	R/W	R/W	-	-	-	
复位后	0	0	0	0	-	-	-	

Bit [6:4] **T0RATE[2:0]：**T0 内部时钟选择位。
000 = fcpu/256；
001 = fcpu/128；
… ；
110 = fcpu/4；
111 = fcpu/2。

Bit 7 **T0ENB：**T0 计数控制位。
0 = 关闭 T0；
1 = 开启 T0。

8.2.3 T0C 计数寄存器

8 位二进制计数器 T0C 控制 T0 的间隔时间。

0D9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0C	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

T0C 初始值的计算方法如下：

T0C 初始值 = 256 -（T0 中断间隔时间 * 输入时钟）

➢ **例：T0 的中断间隔时间为 1ms，高速时钟为内部 12MHz，Fcpu = Fosc/2，T0RATE = 010（Fcpu/64）。**

T0C 初始值 = 256 - (T0 中断间隔时间 * 输入时钟)
= 256 - (1ms * 6MHz / 1 / 64)
= 256 - (10^{-3} * 6 * 10^{6} / 1 / 64)
= 162
= A2H

T0 间隔时间表

T0RATE	T0CLOCK	高速模式（Fcpu = 12MHz / 2）	
		最大间隔溢出时间	单步间隔时间 = max/256
000	Fcpu/256	10.923 ms	42.67 us
001	Fcpu/128	5.461 ms	21.33 us
010	Fcpu/64	2.731 ms	10.67 us
011	Fcpu/32	1.365 ms	5.33 us
100	Fcpu/16	0.683 ms	2.67 us
101	Fcpu/8	0.341 ms	1.33 us
110	Fcpu/4	0.171 ms	0.67 us
111	Fcpu/2	0.085 ms	0.33 us

8.2.4 T0 操作流程

定时器 T0 的操作流程如下：

☞ **T0 停止计数，禁止 T0 中断，清 T0 中断请求标志。**

```
B0BCLR    FT0ENB      ; T0 停止计数。
B0BCLR    FT0IEN      ; 禁止 T0 中断。
B0BCLR    FT0IRQ      ; 清 T0 中断请求标志。
```

☞ **设置 T0 速率。**

```
MOV       A, #0xxx0000b   ;T0M 的 bit4~bit6 位控制 T0 的速率，
                          ; 范围为 x000xxxxB~x111xxxxB。
B0MOV     T0M,A           ; 禁止 T0 定时器。
```

☞ **设置 T0 中断间隔时间。**

```
MOV       A,#7FH
B0MOV     T0C,A       ; 设置 T0C 的值。
```

☞ **设置 T0 定时器的功能。**

```
B0BSET    FT0IEN      ; 使能 T0 中断。
```

☞ **使能定时器 T0。**

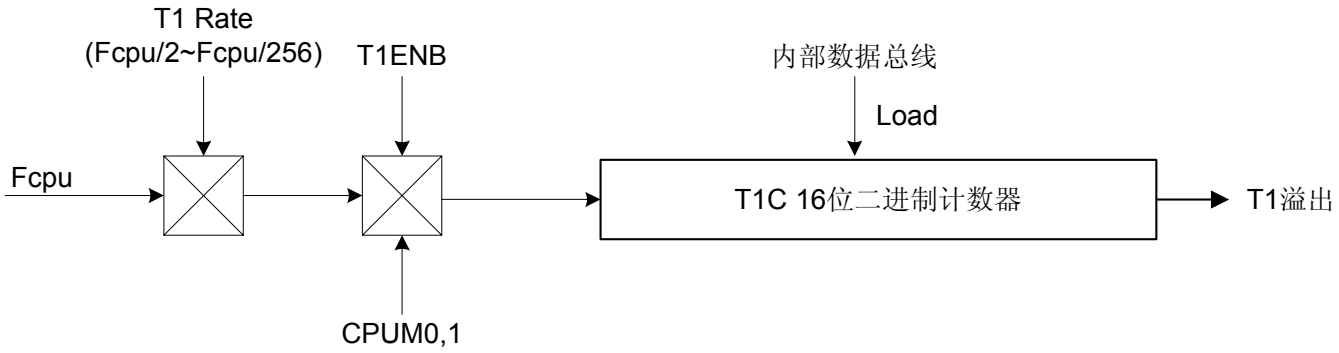
```
B0BSET    FT0ENB      ;
```

8.3 定时器 T1

8.3.1 概述

16 位二进制定时器 T1 溢出（从 0FFFFH 到 0000H）， T1 继续计数并给出信号请求中断。T1 的注意用途如下：

- **16 位可编程计数定时器：**根据选择的时钟频率周期性的请求中断。
- **绿色模式唤醒功能：**T1ENB = 1，将系统从绿色模式下唤醒。



8.3.2 T1M 模式寄存器

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1M	T1ENB	T1rate2	T1rate1	T1rate0	-	-	-	
读/写	R/W	R/W	R/W	R/W	-	-	-	
复位后	0	0	0	0	-	-	-	

Bit [6:4] **T1RATE[2:0]：**T1 内部时钟选择位。
000 = fcpu/256；
001 = fcpu/128；
… ；
110 = fcpu/4；
111 = fcpu/2。

Bit 7 **T1ENB：**T1 计数控制位。
0 = 禁止 T1；
1 = 使能 T1。

8.3.3 T1C 计数寄存器

16 位计数寄存器 T1CL 和 T1CH 控制 T1 的间隔时间。

0DBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CL	T1C7	T1C6	T1C5	T1C4	T1C3	T1C2	T1C1	T1C0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0DCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CH	T1C15	T1C14	T1C13	T1C12	T1C11	T1C10	T1C9	T1C8
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

T1C 的初始值的计算方法如下：

T1C 初始值 = 65536 -（T1 中断间隔时间 * 输入时钟）

☞ **例：设置 T1 的间隔时间为 1ms，高速时钟为 12MHz，Fcpu=Fosc/2，T1RATE=001（Fcpu/128）。**

T1C 初始值 = 65536 -（T1 中断间隔时间 * 输入时钟）
= 65536 - (1s * 12MHz / 2 / 128)
= 65536 - (10 * 6 * 10^6 / 1 / 128)
= 18661
= 48E5H

T1 间隔时间列表

T1RATE	T1CLOCK	高速时钟（Fcpu = 12MHz / 2）	
		最大间隔时间	单步间隔时间 = max/256
000	Fcpu/256	2.796 s	42.67 us
001	Fcpu/128	1.398 s	21.33 us
010	Fcpu/64	699.051 ms	10.67 us
011	Fcpu/32	349.525 ms	5.33 us
100	Fcpu/16	174.763 ms	2.67 us
101	Fcpu/8	87.381 ms	1.33 us
110	Fcpu/4	43.691 ms	0.67 us
111	Fcpu/2	21.845 ms	0.33 us

8.3.4 T1 操作流程

定时器 T1 的操作流程如下：

☞ **T1 停止计数，禁止 T1 中断，清 T1 中断请求标志。**

```
B0BCLR      FT1ENB      ; T1 停止计数。
B0BCLR      FT1IEN      ; 禁止 T1 中断。
B0BCLR      FT1IRQ      ; 清 T1 中断请求标志。
```

☞ **设置 T1 速率。**

```
MOV         A, #0xxx0000b   ;T1M 的 bit4~bit6 位控制 T1 的速率，
                            ; 范围为 x000xxxxB~x111xxxxB。
B0MOV       T1M,A           ; 禁止 T1 定时器。
```

☞ **设置 T0 中断间隔时间。**

```
MOV         A,#0E5H
B0MOV       T1CL,A      ; 设置 T1C_L 的值。
MOV         A,#48H
BOMOV       T1CH,A      ; 设置 T1C_H 的值。
```

☞ **设置 T1 定时器的功能。**

```
B0BSET      FT1IEN      ; 使能 T1 中断。
```

☞ **使能定时器 T1。**

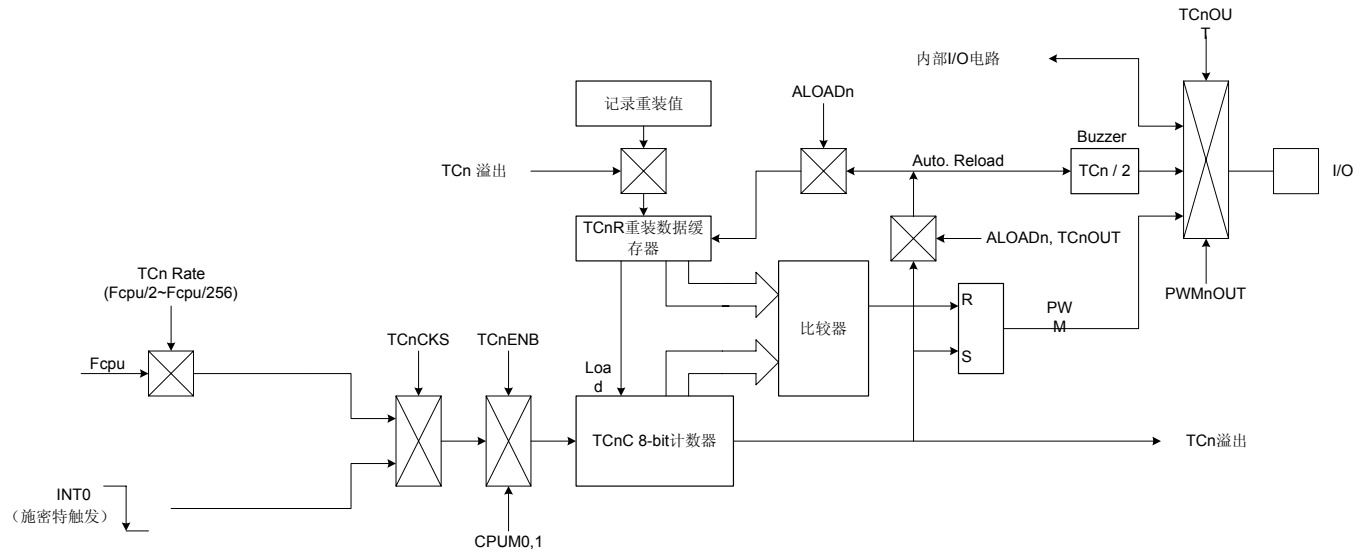
```
B0BSET      FT1ENB      ;
```

8.4 定时/计数器 TC0~TC2

8.4.1 概述

定时/计数器 TCn（n=0~2）具有双时钟源，可根据实际需要选择内部时钟或外部时钟作为计时标准。其中，内部时钟源来自 Fcpu。外部时钟源 INT0 从 P0.0 端输入（下降沿触发）。寄存器 TCnM 控制 TCn 时钟源的选择。当 TCn 从 0FFH 溢出到 00H 时，TCn 在继续计数的同时产生一个溢出信号，触发 TCn 中断请求。PWM 模式下，TCn 的溢出由 PWM 周期（由 ALOADn 和 TCnOUT 位控制）决定。TC0 的主要功能如下：

- **8 位可编程计数定时器：**根据选择的时钟频率在特定的时间请求中断；
- **外部事件计数器：**对外部事件计数；
- **蜂鸣器输出；**
- **PWM 输出。**



8.4.2 TCnM 模式寄存器

088H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **TC0ENB**：TC0 启动控制位。
0 = 关闭 TC0 定时器；
1 = 开启 TC0 定时器。

Bit [6:4] **TC0RATE[2:0]**：TC0 分频选择位。
000 = fcpu/256；
001 = fcpu/128；
… ；
110 = fcpu/4；
111 = fcpu/2。

Bit 3 **TC0CKS**：TC0 时钟信号控制位。
0 = 内部时钟（Fcpu 或 Fosc）；
1 = 外部时钟信号。

Bit 2 **ALOAD0**：自动装载控制位，仅当 PWM0OUT = 0 时有效。
0 = 禁止；
1 = 使能。

Bit 1 **TC0OUT**：TC0 溢出信号输出控制位，仅当 PWM0OUT = 0 时有效。
0 = 禁止，P5.3 作为普通的 I/O 口；
1 = 使能，P5.3 输出 TC0OUT 信号。

Bit 0 **PWM0OUT**：PWM 输出控制位。
0 = 禁止 PWM 输出；
1 = 允许 PWM 输出，PWM 的输出占空比由 TC0OUT 和 ALOAD0 控制。

✸ **注：若 TC0CKS=1，TC0 则用作外部事件计数器，此时不需考虑 TC0RATE 的设置。P0.0 无中断请求（P00IRQ＝0）。**

08BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1M	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS	ALOAD1	TC1OUT	PWM1OUT
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **TC1ENB**：TC1 启动控制位。
0 = 关闭 TC1 定时器；
1 = 开启 TC1 定时器。

Bit [6:4] **TC1RATE[2:0]**：TC1 分频选择位。
000 = fcpu/256；
001 = fcpu/128；
… ；
110 = fcpu/4；
111 = fcpu/2。

Bit 3 **TC1CKS**：TC1 时钟信号控制位。
0 = 内部时钟（Fcpu 或 Fosc）；
1 = 外部时钟信号。

Bit 2 **ALOAD1**：自动装载控制位，仅当 PWM1OUT = 0 时有效。
0 = 禁止；
1 = 使能。

Bit 1 **TC1OUT**：TC1 溢出信号输出控制位，仅当 PWM1OUT = 0 时有效。
0 = 禁止，P5.4 作为普通的 I/O 口；
1 = 使能，P5.4 输出 TC1OUT 信号。

Bit 0 **PWM1OUT**：PWM 输出控制位。
0 = 禁止 PWM 输出；
1 = 允许 PWM 输出，PWM 的输出占空比由 TC1OUT 和 ALOAD1 控制。

✸ **注：若 TC1CKS=1，TC1 则用作外部事件计数器，此时不需考虑 TC1RATE 的设置。P0.1 无中断请求（P01IRQ=0）。**

08EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC2M	TC2ENB	TC2rate2	TC2rate1	TC2rate0	TC2CKS	ALOAD2	TC2OUT	PWM2OUT
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **TC2ENB**：TC2 启动控制位。
0 = 关闭 TC2 定时器；
1 = 开启 TC2 定时器。

Bit [6:4] **TC2RATE[2:0]**：TC2 分频选择位。
000 = fcpu/256；
001 = fcpu/128；
… ；
110 = fcpu/4；
111 = fcpu/2。

Bit 3 **TC2CKS**：TC2 时钟信号控制位。
0 = 内部时钟（Fcpu 或 Fosc）；
1 = 外部时钟信号。

Bit 2 **ALOAD2**：自动装载控制位，仅当 PWM2OUT = 0 时有效。
0 = 禁止；
1 = 使能。

Bit 1 **TC2OUT**：TC2 溢出信号输出控制位，仅当 PWM2OUT = 0 时有效。
0 = 禁止，P5.5 作为普通的 I/O 口；
1 = 使能，P5.5 输出 TC2OUT 信号。

Bit 0 **PWM2OUT**：PWM 输出控制位。
0 = 禁止 PWM 输出；
1 = 允许 PWM 输出，PWM 的输出占空比由 TC2OUT 和 ALOAD2 控制。

8.4.3 TCnC 计数寄存器

8 位计数寄存器 TCnC（n=0，1，2）控制 TCn0 的中断间隔时间。

089H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0C	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

08CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1C	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

08FH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC2C	TC2C7	TC2C6	TC2C5	TC2C4	TC2C3	TC2C2	TC2C1	TC2C0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

TCnC 初始值的计算方法如下：

TCnC 初始值 = N -（TCn 中断间隔时间 * 输入时钟）

N 为 TCn 溢出的临界数值。TCn 的溢出时间和有效值参考下表：

TCnCKS	PWMn	ALOADn	TCnOUT	N	TCnC 有效值	TCnC 有效值	备注
0	0	x	x	256	00H~0FFH	00000000b~11111111b	每计数 256 次溢出
	1	0	0	256	00H~0FFH	00000000b~11111111b	每计数 256 次溢出
	1	0	1	64	00H~3FH	xx000000b~xx111111b	每计数 64 次溢出
	1	1	0	32	00H~1FH	xxx00000b~xxx11111b	每计数 32 次溢出
	1	1	1	16	00H~0FH	xxxx0000b~xxxx1111b	每计数 16 次溢出
1	-	-	-	256	00H~0FFH	00000000b~11111111b	每计数 256 次溢出

➢ **例：设置 TCn 的中断间隔时间为 1ms，TCn 时钟源来自 Fcpu（TCnCKS=0），无 PWM 输出（PWMn=0），高速时钟由内部 6MHz 提供，Fcpu=Fosc/2，TC0RATE=010（Fcpu/64）。**

TC0C 初始值 = N -（TC0 中断间隔时间 * 输入时钟）
= 256 - (1ms * 6MHz / 1 / 64)
= 256 - (10^{-3} * 6 * 10^{6} / 1 / 64)
= 162
= A2H

TCn 中断间隔时间列表

TCnRATE	TCnCLOCK	高速时钟（Fcpu = 6MHz / 1）	
		最大间隔时间	单步间隔时间 = max/256
000	Fcpu/256	10.923 ms	42.67 us
001	Fcpu/128	5.461 ms	21.33 us
010	Fcpu/64	2.731 ms	10.67 us
011	Fcpu/32	1.365 ms	5.33 us
100	Fcpu/16	0.683 ms	2.67 us
101	Fcpu/8	0.341 ms	1.33 us
110	Fcpu/4	0.171 ms	0.67 us
111	Fcpu/2	0.085 ms	0.33 us

8.4.4 TCnR 自动装载寄存器

TCn（n=0，1，2）的自动重装功能由 TCnM 的 ALOADn 位控制。当 TCnC 溢出时，TCnR 的值自动装入 TCnC 中。这样，用户在使用的过程中就不需要在中断中复位 TCnC。

TCn 为双重缓存器结构。若程序对 TCnR 进行了修改，那么修改后的 TCnR 值首先被暂存在 TCnR 的第一个缓存器中，TCn 溢出后，TCnR 的新值就会被存入 TCnR 缓存器中，从而避免 TCn 中断时间出错以及 PWM 和蜂鸣器误动作。

✸ **注：在 PWM 模式下，系统自动开启重装功能，ALOADn 用于控制溢出范围。**

08AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

08DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1R	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

090H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC2R	TC2R7	TC2R6	TC2R5	TC2R4	TC2R3	TC2R2	TC2R1	TC2R0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

TCnR 初始值的计算方法如下：

TCnR 初始值 = N -（TCn 中断间隔时间 * 输入时钟）

N 为 TCn 溢出的临界数值。TCn 的溢出时间和有效值参考下表：

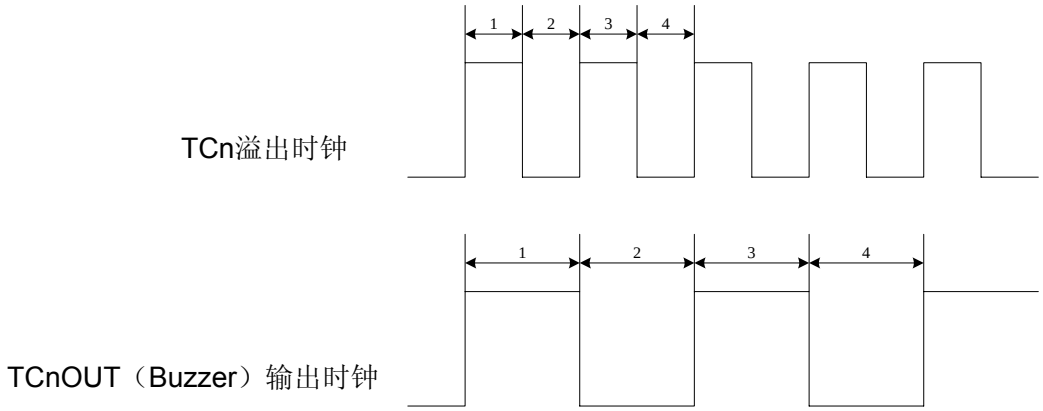
TCnCKS	PWMn	ALOADn	TCnOUT	N	TCnC 有效值	TCnC 有效值
0	0	x	x	256	00H~0FFH	00000000b~11111111b
	1	0	0	256	00H~0FFH	00000000b~11111111b
	1	0	1	64	00H~3FH	xx000000b~xx111111b
	1	1	0	32	00H~1FH	xxx00000b~xxx11111b
	1	1	1	16	00H~0FH	xxxx0000b~xxxx1111b
1	-	-	-	256	00H~0FFH	00000000b~11111111b

➢ **例：设置 TCn 的中断间隔时间为 1ms，TCn 时钟源来自 Fcpu（TCnCKS=0），无 PWM 输出（PWMn=0），高速时钟由内部 6MHz 提供，Fcpu=Fosc/1，TCnRATE=010（Fcpu/64）。**

TC0R 初始值 = N -（TC0 中断间隔时间 * 输入时钟）
= 256 - (1ms * 6MHz / 1 / 64)
= 256 - (10^{-3} * 6 * 10^{6} / 1 / 64)
= 162
= A2H

8.4.5 TCn 时钟频率输出（BUZZER）

对 TCn 时钟频率进行适当设置可得到特定频率的蜂鸣器输出（TCnOUT），并通过引脚 P5.3/4/5 输出。单片机内部设置 TCn 的溢出频率经过 2 分频后作为 TCnOUT 的频率，即 TCn 每溢出 2 次 TCnOUT 输出一个完整的脉冲，此时，P5.3/4/5 的 I/O 功能自动被禁止。TCnOUT 输出波形如下：



若外部高速时钟选择 4MHz，系统时钟源采用外部时钟 Fosc/4，程序中设置 TC0RATE2~TC0RATE1 = 110， TC0C = TC0R = 131，则 TC0 的溢出频率为 1KHz，TC0OUT 的输出频率为 0.5KHz。下面给出范例程序。

➢ **例：设置 TC0OUT（P5.4）。**

```
MOV     A,#01100000B
B0MOV   TC0M,A          ; TC0 速率 = Fcpu/4。

MOV     A,#131          ; 自动装载参考值设置。
B0MOV   TC0C,A
B0MOV   TC0R,A

B0BSET  FTC0OUT         ; TC0 的输出信号由 P5.4 输出，禁止 P5.4 的普通 I/O 功能。
B0BSET  FALOAD0         ; 允许 TC0 自动重装功能。
B0BSET  FTC0ENB         ; 开启 TC0 定时器。
```

✸ **注：蜂鸣器的输出有效时，“PWM0OUT”必须被置为“0”。**

8.4.6 TCn 操作流程

TCn 定时器可用于定时器中断、事件计数、TCnOUT 和 PWM。下面以 TC0 为例进行说明。

☞ **停止 TC0 计数，禁止 TC0 中断，并清 TC0 中断请求标志。**

```
B0BCLR    FTC0ENB     ; 停止 TC0 计数、TC0OUT 和 PWM。
B0BCLR    FTC0IEN     ; 禁止 TC0 中断。
B0BCLR    FTC0IRQ     ; 清 TC0 中断请求标志。
```

☞ **设置 TC0 的速率 (不包含事件计数模式)。**

```
MOV       A, #0xxx0000b   ; TC0M 的 bit4~bit6 控制 TC0 的速率为 x000xxxxb~x111xxxxb。
B0MOV     TC0M,A          ; 禁止 TC0 中断。
```

☞ **设置 TC0 的时钟源。**

```
; 选择 TC0 内部/外部时钟源。
B0BCLR    FTC0CKS     ; 内部时钟。
```

或

```
B0BSET    FTC0CKS     ; 外部时钟。
```

☞ **设置 TC0 的自动装载模式。**

```
B0BCLR    FALOAD0     ; 禁止 TC0 自动装载功能。
```

或

```
B0BSET    FALOAD0     ; 使能 TC0 自动装载功能。
```

☞ **设置 TC0 中断间隔时间，TC0OUT（Buzzer）频率或 PWM 占空比。**

```
; 设置 TC0 中断间隔时间，TC0OUT（Buzzer）频率或 PWM 占空比。
MOV       A,#7FH          ; TC0 的模式决定 TC0C 和 TC0R 的值。
B0MOV     TC0C,A          ; 设置 TC0C 的值。
B0MOV     TC0R,A          ; 在自动装载模式或 PWM 模式下设置 TC0R 的值。

; PWM 模式下设置 PWM 的周期。
B0BCLR    FALOAD0     ; ALOAD0，TC0OUT = 00，PWM 周期 = 0~255。
B0BCLR    FTC0OUT
```

或

```
B0BCLR    FALOAD0     ; ALOAD0，TC0OUT = 01，PWM 周期 = 0~63。
B0BSET    FTC0OUT
```

或

```
B0BSET    FALOAD0     ; ALOAD0，TC0OUT = 10，PWM 周期 = 0~31。
B0BCLR    FTC0OUT
```

或

```
B0BSET    FALOAD0     ; ALOAD0，TC0OUT = 11，PWM 周期 = 0~15。
B0BSET    FTC0OUT
```

☞ **设置 TC0 的模式。**

```
B0BSET    FTC0IEN     ; 使能 TC0 中断。
```

或

```
B0BSET    FTC0OUT     ; 使能 TC0OUT（Buzzer）功能。
```

或

```
B0BSET    FPWM0OUT    ; 使能 PWM。
```

☞ **开启 TC0 定时器。**

```
B0BSET    FTC0ENB
```

8.5 PWMn 模式

8.5.1 概述

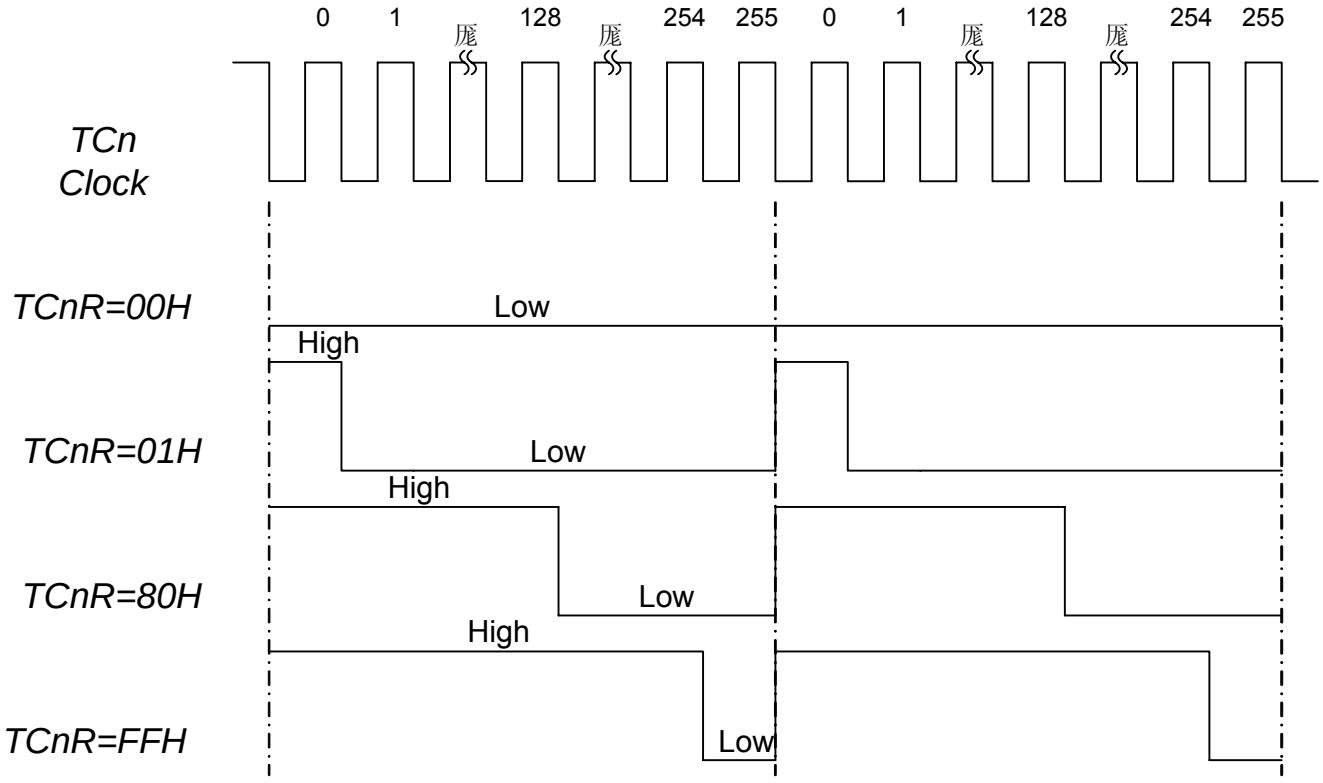
PWM 信号输出到 PWM0OUT（P5.3 引脚）、PWM1OUT（P5.4 引脚）和 PWM2OUT（P5.5 引脚），位 TCnOUT 和 ALOADn 控制 PWM 输出的阶数（256、64、32 和 16）。8 位计数器 TCnC 计数过程中不断与 TCnR 相比较，当 TCnC 计数到两者相等时，PWM 输出低电平，当 TCnC 再次从零开始计数时，PWM 被强制输出高电平。PWMn 输出占空比 = TCnR/计数量程（计数量程 = 256、64、32 或 16）。

参考寄存器保持输入 00H 可使 PWM 的输出长时间维持在低电平，通过修改 TCnR 可改变 PWM 输出占空比。

✸ **注：TCn 为双重缓存器结构，通过调整 TCnR 的值可以改变 PWM 的占空比，在 PWM 输出的波形中不会出现错误的信号。用户可以随时改变 TCnR 的值，在 TCn 溢出时，新的修改值才会存入 TCnR 寄存器。**

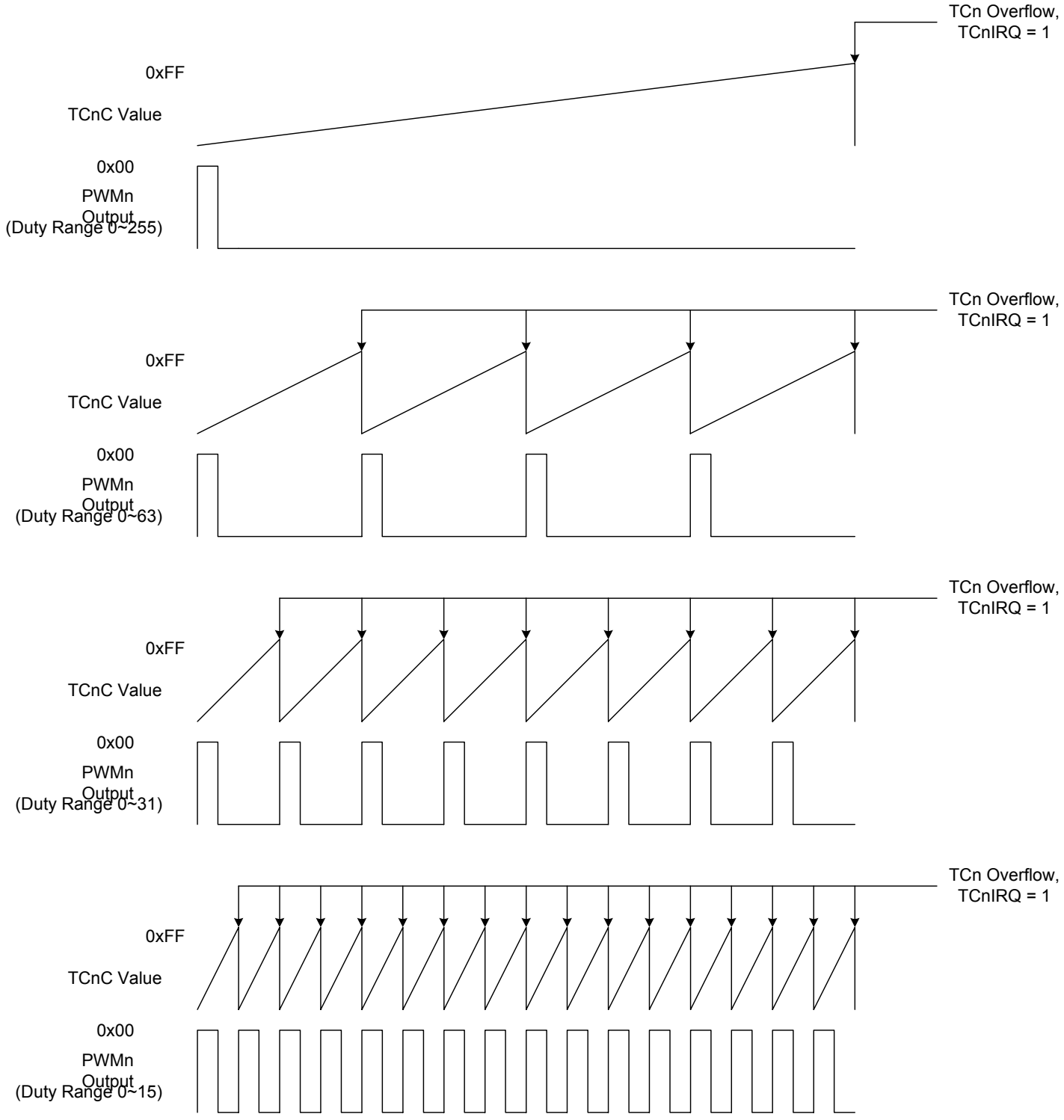
ALOADn	TCnOUT	PWM 占空比范围	TCnC 有效值	TCnR 有效值	MAX. PWM 频率 (Fcpu = 6MHz)	备注
0	0	0/256~255/256	00H~0FFH	00H~0FFH	11.719K	每计数 256 次溢出
0	1	0/64~63/64	00H~3FH	00H~3FH	46.875K	每计数 64 次溢出
1	0	0/32~31/32	00H~1FH	00H~1FH	93.75K	每计数 32 次溢出
1	1	0/16~15/16	00H~0FH	00H~0FH	187.5K	每计数 16 次溢出

PWM 的输出占空比随 TCnR 的变化而变化：0/256~255/256



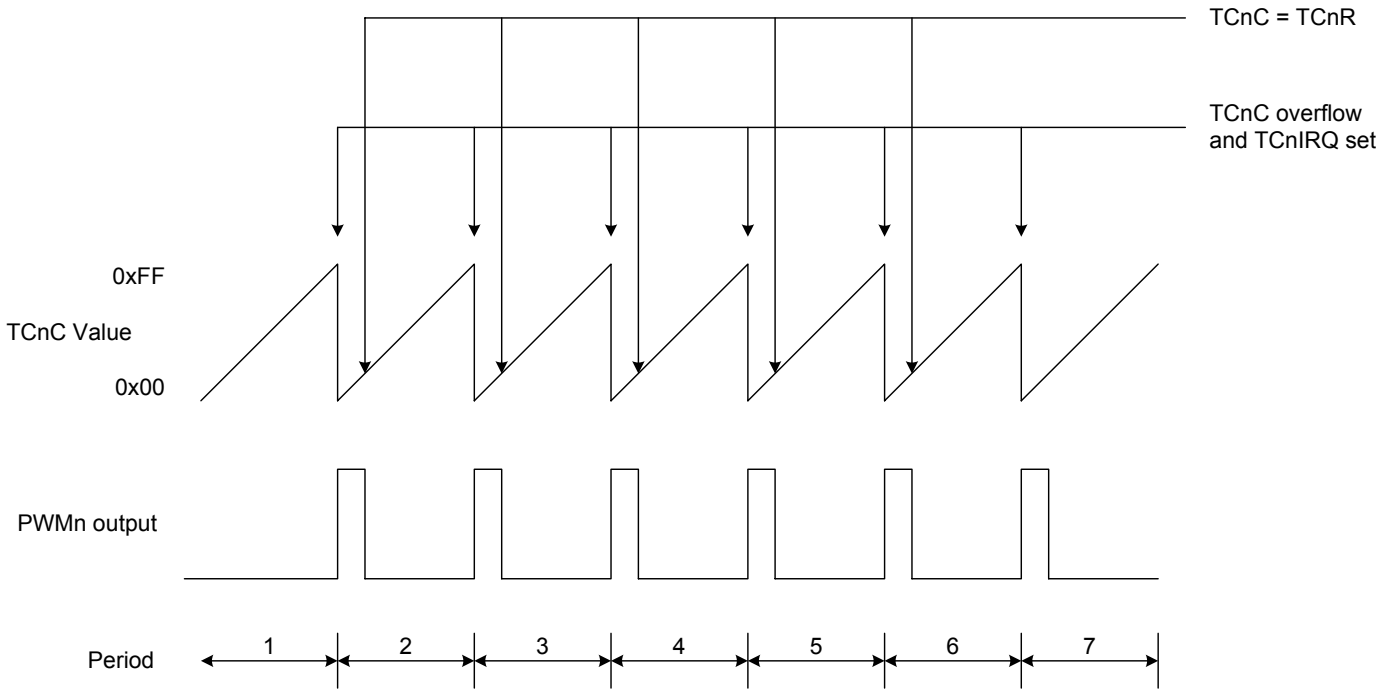
8.5.2 TCnIRQ 和 PWM 占空比

在 PWM 模式下，TCnIRQ 的频率与 PWM 的占空比有关，具体情况如下图所示：

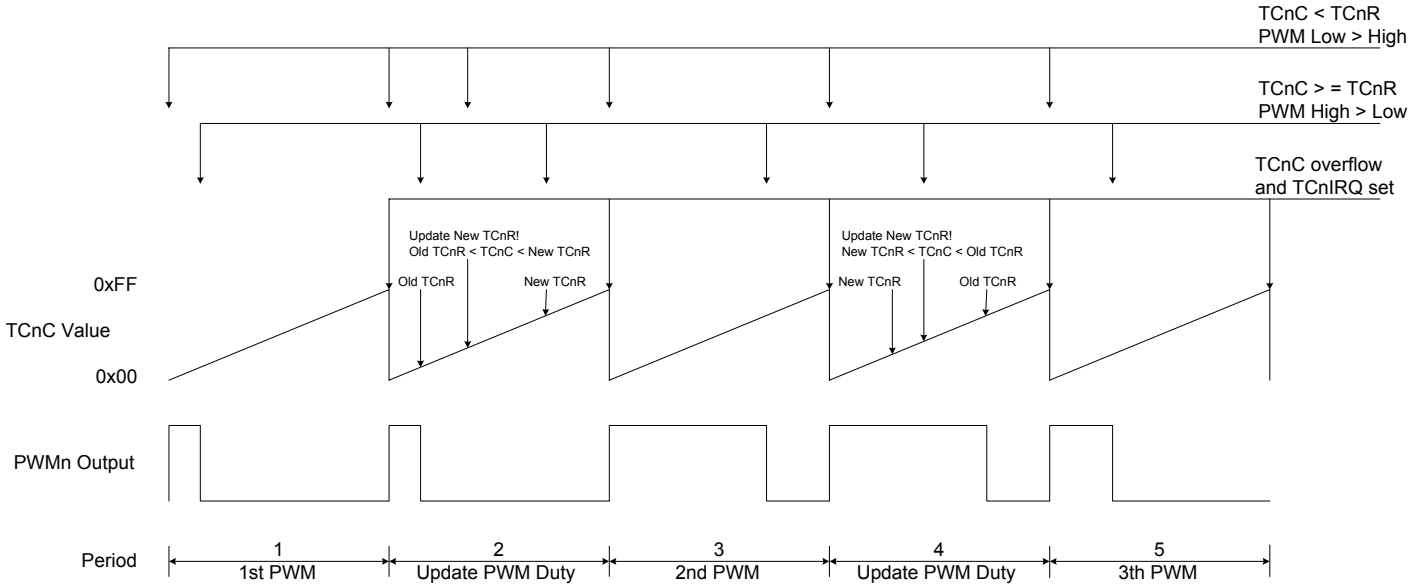


8.5.3 PWM 输出占空比与 TCnR 值的变化

在 PWM 模式下，系统随时比较 TCnC 和 TCnR 的异同。若 TCnC<TCnR，PWM 输出高电平，反之则输出低电平。当 TCnC 发生改变的时候，PWM 将在下一周期改变输出占空比。如果 TCnR 保持恒定，那么 PWM 输出波形也保持稳定。



上图所示是 TCnR 恒定时的波形。每当 TCnC 溢出时，PWM 都输出高电平，TCnC≧TCnR 时，PWM 即输出低电平。下面所示是 TCnR 发生变化时对应的波形图：



在 period 2 和 period 4 中，显示新的占空比（TCnR），但 PWM 在 period 2 和 period 4 的占空比要在下一个 period 才会改变。这时，系统可以使 PWM 变化或者在同一个周期内 H/L 改变两次，这样系统可以预防未知的的系统错误。

8.5.4 PWM 编程举例

☞ **例：PWM0 输出设置。时钟频率为内部 12MHz，Fcpu = Fosc/2=6MHz，PWM 输出占空比为 30/256，PWM 输出频率约为 6KHz，PWM 时钟源来自外部振荡时钟。TC0 速率 = Fcpu/4，TC0RATE2~TC0RATE1 = 110. TC0C = TC0R = 30.**

```
MOV     A,#01100000B
B0MOV   TC0M,A          ; 设置 TC0 速率为 Fcpu/4。

MOV     A,#30           ; 设置 PWM 输出占空比为 30/256。
B0MOV   TC0C,A
B0MOV   TC0R,A

B0BCLR  FTC0OUT         ; 设置 PWM 输出占空比的范围为 0/256~255/256。
B0BCLR  FALOAD0
B0BSET  FPWM0OUT        ; 使能 PWM0 输出到 P5.3，同时禁止 P5.3 普通的 I/O 功能。
B0BSET  FTC0ENB         ; 开启 TC0 定时器。
```

✸ **注：TCnR 为只写寄存器，不能用 INCMS 和 DECMS 指令对其进行操作。**

➢ **例：调整 TC0R 寄存器的值。**

```
MOV     A, #30H         ; 用 B0MOV 指令输入一个立即数。
B0MOV   TC0R, A

INCMS   BUF0            ; 从程序定义的缓存器 BUF0 得到新的 TC0R 的值。
NOP
B0MOV   A, BUF0
B0MOV   TC0R, A
```

✸ **注：PWM 可以在中断模式下工作。**

9 USB

9.1 概述

USB 做为 PC 与外围通信的一种接口标准，以其快速、双向、同步、低成本、热插拔等特点大大满足了 PC 平台未来发展的需要。SONiX USB 控制芯片正将诸如鼠标、摇杆，游戏垫等计算机外设的人机交互推向最优化。

USB遵循的规范：

- 遵循USB规范V2.0。
- 支持1个全速USB设备地址。
- 支持1个控制端点、2个中断端点和2个中断/bulk端点。
- 集成USB收发器。
- 使能USB功能后，VREG将为D+ 的1.5K上拉电阻提供3.3V电压。

9.2 USB 机构

USB 机构实现单片机与 USB 主机的通信，硬件可独立地完成如下 USB 动作：

USB 机构将完成：

- 对接收到的数据译码，对将要发送的数据编码；
- CRC 的校验和产生由硬件完成。若 CRC 错误，硬件将不给主机任何回应；
- 硬件自动更新发送数据同步校准位；
- USB 控制寄存器发送相应的 ACK/NAK/STALL 握手信号；
- 不同类型令牌包（SETUP、IN、OUT）的识别。接收到有效令牌包后，对相应寄存器的状态位置“1”；
- 将接收到的有效数据存放在相应的端点 FIFO 中；
- 填充校验位。
- 地址检查。丢掉无效地址的传输。
- 端点检查。检查到来自主机的请求后，对寄存器的相关位进行设置。

韧件完成以下功能：

- 通过接收 USB 设备请求上传相对应的列举数据；
- 填写/清空 FIFO。
- 挂起/唤醒功能。
- 设备端唤醒功能；
- USB 传输中，决定采用何种中断请求。

9.3 USB 中断

USB 功能接受 USB 主机的命令并产生相关的中断，程序计数器跳转到中断向量地址，此时要求韧件检查 USB 的状态位以了解产生的是哪一种中断。

在以下情况下会产生 USB 中断：

- 端点 0 接收到设置令牌包（token SETUP）；
- 成功完成输入事务后，设备会接收到一个 ACK 应答信号。
- 端点在 ACK OUT 模式下，接收到数据后产生中断；
- USB 主机送出 USB 挂起请求；
- USB 复位；
- USB处理完成后端点中断；
- 使能SOF中断，接收到SOF包裹；
- 使能NAK中断时NAK握手。

9.4 USB 枚举

典型的USB枚举流程如下：

1. USB主机发送Setup包后，再发送DATA包到地址0，请求设备发送设备描述符Device descriptor.
2. 韧件收到请求，从ROM表中找到设备描述符。
3. USB主机发送输入控制时序，程序找到FIFO中的设备描述符通过USB回传给主机。
4. USB主机接收到描述符后，发送SETUP包和DATA包给地址0以给设备分配新USB地址。
5. 无数据的状态控制阶段结束后，程序会将新地址存储在USB设备地址寄存器中。
6. 主机利用新的USB地址请求发送设备描述符；
7. 韧件解码得到请求命令，并从程序存储列表中找出设备描述符；
8. 主机电脑发送读取命令，韧件将设备描述符发送到USB总线上；
9. 主机发出控制读命令，请求配置和说明描述符；
10. 一旦设备接收到一个Set Configuration请求后，USB功能开始使用；
11. 程序要完成端点0~3的各交易操作。

9.5 USB 寄存器

9.5.1 USB 设备地址寄存器

USB地址寄存器（UDA）包括7位USB设备地址和1位USB功能使能位。该寄存器在复位后被清零并禁止USB功能。

091H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UDA	UDE	UDA6	UDA5	UDA4	UDA3	UDA2	UDA1	UDA0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **UDE：USB设备的功能使能位。**使能USB功能后，该位必须由韧件设置。
0 = 禁止USB功能；
1 = 使能USB功能。

Bit [6:0] **UDA [6:0]：**在 USB 枚举过程（如 SetAddress）中，因 USB 主机分配非零地址，而使相应位被置 1。

9.5.2 USB 状态寄存器

USB 状态寄存器显示 USB 的状态。

092H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
USTATUS	CRCERR	PKTERR	SOF	BUS_RST	SUSPEND	EP0SETUP	EP0IN	EP0OUT
读/写	R/W	R/W	R/W	R	R	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **CRCERR：**USB 数据 CRC 检测位。
0 = 没有检测到 USB 数据 CRC 出错，由韧件清零；
1 = 检测到 USB 数据 CRC 出错后由硬件置 1。

Bit 6 **PKTERR：**USB 包裹错误检查位。
0 = USB 包没有出错，由韧件清零；
1 = USB 包出错后由硬件置 1。

Bit 5 **SOF：**USB SIE SOF 包接收指示位。
0 = 没有接收到 USB SIE SOF 包；
1 = 接收到 SOF 包且 SOF_INT_EN=1 时，该位由硬件设为 1，否则该位将一直为 0。

Bit 4 **BUS_RST：**USB 复位位。
0 = 无 USB 复位；
1 = USB 请求中断时由硬件设为 1。

Bit 3 **SUSPEND：**USB 挂起状态位。
0 = 非挂起状态。当单片机由 USB 将其从睡眠模式下唤醒后，该位自动设为 0；
1 = USB 挂起请求时由硬件设为 1。

Bit 2 **EP0_SETUP：**端点 0 SETUP Token 接收指示位。
0 = 端点 0 没有接收到 SETUP Token；
1 = 端点 0 接收到有效 SETUP Token 包，接收到最后一个 SETUP 包后，该位被置 1。当该位为 1 时，主机不能向 EP0 FIFO 写入任何数据。这将阻止 SIE 在韧件读到 SETUP 数据之前，再次写入 SETUP 输入事务。

Bit 1 **EP0_IN：**端点 0 IN Token 接收指示位。
0 = 端点 0 没有接收到 IN Token；
1 = 端点 0 接收到有效的 IN Token 包，接收到最后一个 IN 包后，该位被置 1。

Bit 0 **EP0_OUT：**端点 0 OUT Token 接收指示位。
0 = 端点 0 没有接收到 OUT Token；
1 = 端点 0 接收到有效的 OUT Token 包，接收到最后一个 OUT 包后，该位被置 1。

9.5.3 USB 数据计数寄存器

USB EP0 OUT token 数据字节计数器。

093H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP0OUT_CNT	-	-	-	UEP0OC4	UEP0OC3	UEP0OC2	UEP0OC1	UEP0OC0
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

Bit [4:0] UEP0C [4:0]：USB 端点 0 OUT Token 数据计数器。

9.5.4 USB 使能控制寄存器

USB 使能寄存器控制 3.3V 电压输出，SOF 包接收中断，NAK 握手中断和 D+内部 1.5K 上拉电阻。

094H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
USB_INT_EN	REG_EN	DP_UP_EN	SOF_INT_EN	-	EP4NAK _INT_EN	EP3NAK _INT_EN	EP2NAK _INT_EN	EP1NAK _INT_EN
读/写	R/W	R/W	R/W	-	R/W	R/W	R/W	R/W
复位后	1	0	0	-	0	0	0	0

Bit 7 **REG_EN**：3.3V 调整仪控制位。
0 = 禁止输出 3.3V；
1 = 使能输出 3.3V。

Bit 6 **DN_UP_EN**：D+内部 1.5K 上拉电阻控制位。
0 = 禁止接 1.5K 上拉电阻到 3.3V；
1 = 使能接 1.5K 上拉电阻到 3.3V。

Bit 5 **SOF_INT_EN**：USB SIE SOF 包接收中断使能位。
0 = 禁止 USB SIE SOF 中断请求；
1 = 使能 USB SIE SOF 中断请求。

Bit [3:0] **EPnNAK_INT_EN [3:0]**：EP1~EP4 NAK处理中断控制位。N=1~4。
0 = 禁止 NAK 中断请求；
1 = 使能 NAK 中断请求。

9.5.5 USB 端点 ACK 握手标志寄存器

该寄存器显示端点 ACK 处理事务的状态。

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP_ACK	-	-	-	-	EP4_ACK	EP3_ACK	EP2_ACK	EP1_ACK
读/写	-	-	-	-	R/W	R/W	R/W	R/W
复位后	-	-	-	-	0	0	0	0

Bit [3:0] **EPn_ACK [3:0]**：EP1~EP4 ACK处理事务。n= 1～4，只要完成ACK处理事务，该位置1。
0 = 端点（中断通道）没有完成 ACK 处理事务；
1 = 端点（中断通道）完成 ACK 处理事务。

9.5.6 USB 端点 NAK 握手标志寄存器

该寄存器显示端点 NAK 处理事务的状态。

096H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP_NAK	-	-	-	-	EP4_NAK	EP3_NAK	EP2_NAK	EP1_NAK
读/写	-	-	-	-	R/W	R/W	R/W	R/W
复位后	-	-	-	-	0	0	0	0

Bit [3:0] **EPn_NAK [3:0]**：EP1~EP4 NAK处理事务。n = 1～4，只要完成ACK处理事务，该位就置1。
0 = EPnNAK_INT_EN = 0 或端点（中断通道）没有完成 NAK 处理事务；
1 = EPnNAK_INT_EN = 1 和端点（中断通道）完成 NAK 处理事务。

9.5.7 USB 端点 0 使能寄存器

端点0（EP0）用来初始化USB和控制USB。EP0是一个双向控制设备，可以发送和接收数据， 用来控制USB设备的配置和USB的状态。

097H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE0R	-	UE0M1	UE0M0	-	UE0C3	UE0C2	UE0C1	UE0C0
读/写	-	R/W	R/W	-	R/W	R/W	R/W	R/W
复位后	-	0	0	-	0	0	0	0

Bit [6:5] **UE0M [1:0]**：EP0 模式决定了 SIE 对主机发送给 EP0 的 USB 包作出何种响应。例如，当 EP0 模式位置为 00 时（如以下列表中的 NAK 对应项），USB SIE 将发送 NAK 信号以回应给端点 0 的任一 IN/OUT token。

USB EP0 模式列表

UE0M1	UE0M0	IN/OUT Token 握手
0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

Bit [3:0] **UE0C [3:0]**：表明了事务处理的数据字节长度：对输入交易而言，指的是韧件要从端点 0 FIFO 上传给 USB 主机的数据字节的个数。

9.5.8 USB 端点 1 使能寄存器

利用EP1和USB主机之间进行通讯，专用RAM的W字节来执行EP1 FIFO。EP1是一个中断端点。

098H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE1R	UE1E	UE1M1	UE1M0	-	-	-	-	-
读/写	R/W	R/W	R/W	-	-	-	-	-
复位后	0	0	0	-	-	-	-	-

Bit 7 **UE1E**：USB 端点 1 功能使能位。
0 = 禁止 USB 端点 1 的功能；
1 = 使能 USB 端点 1 的功能。

Bit [6:5] **UE1M [1:0]**：EP1 模式决定了 SIE 对主机发送给 EP1 的 USB 包作出何种响应。例如，当 EP1 模式位置为 00 时（如以下列表中的 NAK 对应项），USB SIE 将发送 NAK 信号以回应给端点 1 的任一 IN/OUT token。当 ACK 事务处理完成时，UE1M0 位会自动复位清零。

USB EP1 模式列表

UE1M1	UE1M0	IN/OUT Token 握手
0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

099H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE1R_C	-	-	UE1C5	UE1C4	UE1C3	UE1C	UE1C1	UE1C0
读/写	-	-	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	0	0	0	0	0	0

Bit [5:0] **UE1C [5:0]**：指示交易中的数据字节数。在输入交易中，指的是韧件从端点1的FIFO要传给主机的字节数。

9.5.9 USB 端点 2 使能寄存器

利用EP2和USB主机之间进行通讯，专用RAM的X字节来执行EP2 FIFO。EP2是一个中断端点。

09AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE2R	UE2E	UE2M1	UE2M0	-	-	-	-	-
读/写	R/W	R/W	R/W	-	-	-	-	-
复位后	0	0	0	-	-	-	-	-

Bit 7 **UE2E**：USB 端点 2 功能使能位。
0 = 禁止 USB 端点 2 的功能；
1 = 使能 USB 端点 2 的功能。

Bit [6:5] **UE2M [1:0]**：EP2 模式决定了 SIE 对主机发送给 EP2 的 USB 包作出何种响应。例如，当 EP2 模式位置为 00 时（如以下列表中的 NAK 对应项），USB SIE 将发送 NAK 信号以回应给端点 2 的任一 IN/OUT token。当 ACK 事务处理完成时，UE2M0 位会自动复位清零。

USB EP2 模式列表

UE2M1	UE2M0	IN/OUT Token 握手
0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

09BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE2R_C	-	-	UE2C5	UE2C4	UE2C3	UE2C	UE2C1	UE2C0
读/写	-	-	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	0	0	0	0	0	0

Bit [5:0] **UE2C [5:0]**：指示交易中的数据字节数。在输入交易中，指的是韧件从端点2的FIFO要传给主机的字节数。

9.5.10 USB 端点 3 使能寄存器

利用EP3和USB主机之间进行通讯，专用RAM的Y字节来执行EP3 FIFO。EP3是一个中断端点和bulk端点。

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE3R	UE3E	UE3M1	UE3M0	-	-	-	-	-
读/写	R/W	R/W	R/W	-	-	-	-	-
复位后	0	0	0	-	-	-	-	-

Bit 7 **UE3E**：USB 端点 3 功能使能位。
0 = 禁止 USB 端点 3 的功能；
1 = 使能 USB 端点 3 的功能。

Bit [6:5] **UE3M [1:0]**：EP3 模式决定了 SIE 对主机发送给 EP3 的 USB 包作出何种响应。例如，当 EP3 模式位置为 00 时（如以下列表中的 NAK 对应项），USB SIE 将发送 NAK 信号以回应给端点 3 的任一 IN/OUT token。当 ACK 事务处理完成时，UE3M0 位会自动复位清零。

USB EP3 模式列表

UE3M1	UE3M0	IN/OUT Token 握手
0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

09DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE3R_C	-	-	UE3C5	UE3C4	UE3C3	UE3C	UE3C1	UE3C0
读/写	-	-	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	0	0	0	0	0	0

Bit [5:0] **UE3C [5:0]**：指示交易中的数据字节数。在输入交易中，指的是韧件从端点3的FIFO要传给主机的字节数。

9.5.11 USB 端点 4 使能寄存器

利用EP4和USB主机之间进行通讯，专用RAM的Z字节来执行EP4 FIFO。EP4是一个中断端点和bulk端点。

09EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE4R	UE4E	UE4M1	UE4M0	-	-	-	-	-
读/写	R/W	R/W	R/W	-	-	-	-	-
复位后	0	0	0	-	-	-	--	-

Bit 7 **UE4E**：USB 端点 4 功能使能位。
0 = 禁止 USB 端点 4 的功能；
1 = 使能 USB 端点 4 的功能。

Bit [6:5] **UE4M [1:0]**：EP4 模式决定了 SIE 对主机发送给 EP4 的 USB 包作出何种响应。例如，当 EP4 模式位置为 00 时（如以下列表中的 NAK 对应项），USB SIE 将发送 NAK 信号以回应给端点 4 的任一 IN/OUT token。当 ACK 事务处理完成时，UE4M0 位会自动复位清零。

USB EP4 模式列表

UE4M1	UE4M0	IN/OUT Token 握手
0	0	NAK
0	1	ACK
1	0	STALL
1	1	STALL

09FH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UE4R_C	-	-	UE4C5	UE4C4	UE4C3	UE4C	UE4C1	UE4C0
读/写	-	-	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	0	0	0	0	0	0

Bit [5:0] **UE4C [5:0]**：指示交易中的数据字节数。在输入交易中，指的是韧件从端点4的FIFO要传给主机的字节数。

9.5.12 USB 端点 FIFO 地址设置寄存器

0A0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP2FIFO_ADDR	EP2FIFO7	EP2FIFO6	EP2FIFO5	EP2FIFO4	EP2FIFO3	EP2FIFO2	EP2FIFO1	EP2FIFO0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **EP2FIFO_ADDR [7:0]**：EP2 FIFO开始地址。

0A1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP3FIFO_ADDR	EP3FIFO7	EP3FIFO6	EP3FIFO5	EP3FIFO4	EP3FIFO3	EP3FIFO2	EP3FIFO1	EP3FIFO0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **EP3FIFO_ADDR [7:0]**：EP3 FIFO开始地址。

0A2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EP4FIFO_ADDR	EP4FIFO7	EP4FIFO6	EP4FIFO5	EP4FIFO4	EP4FIFO3	EP4FIFO2	EP4FIFO1	EP4FIFO0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **EP4FIFO_ADDR [7:0]**：EP4 FIFO开始地址。

9.5.13 USB 数据指示寄存器

USB FIFO 的地址指示器。使用该指示器设置 FIFO 的地址以读取 USB FIFO 的数据和写入数据到 USB FIFO。

0A3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UDP0	UDP07	UDP06	UDP05	UDP04	UDP03	UDP02	UDP01	UDP00
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

USB 数据指示器：通过数据指示器访问 USB 数据。

9.5.14 USB 数据读/写寄存器

0A5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UDR0_R	UDR0_R7	UDR0_R6	UDR0_R5	UDR0_R4	UDR0_R3	UDR0_R2	UDR0_R1	UDR0_R0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

UDR0_R：读取 UDP0 所指 USB FIFO 地址的数据。

0A6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UDR0_W	UDR0_W7	UDR0_W6	UDR0_W5	UDR0_W4	UDR0_W3	UDR0_W2	UDR0_W1	UDR0_W0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

UDR0_W：写入数据到 UDP0 所指 USB FIFO 地址。

9.5.15 UPID 寄存器

强制允许韧件可以直接驱动D+和D-引脚。

0A7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UPID	-	-	-	-	-	UBDE	DDP	DDN
读/写	-	-	-	-	-	R/W	R/W	R/W
复位后	-	-	-	-	-	0	0	0

Bit 2 **UBDE**：直接驱动 USB 总线使能位。
0 = 禁止；
1 = 使能。

Bit 1 **DDP**：在 USB 总线上驱动 D+。
0 = 驱动 D+低电平；
1 = 驱动 D+高电平。.

Bit 0 **DDN**：在 USB 总线上驱动 D-。
0 = 驱动 D-低电平；
1 = 驱动 D-高电平。.

9.5.16 端点 TOGGLE 位控制寄存器

0ACH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UTOGGLE	-	-	-	-	EP4 _DATA01	EP3 _DATA01	EP2 _DATA01	EP1 _DATA01
读/写	-	-	-	-	R/W	R/W	R/W	R/W
复位后	-	-	-	-	1	1	1	1

Bit [3:0] 端点 1～4 DATA0/1 TOGGLE 控制位。
0 = 清除端点 1～4 TOGGLE 位到 DATA0；
1 = 硬件自动设置 TOGGLE 位为 1。

10 通用异步收发器（UART）

10.1 概述

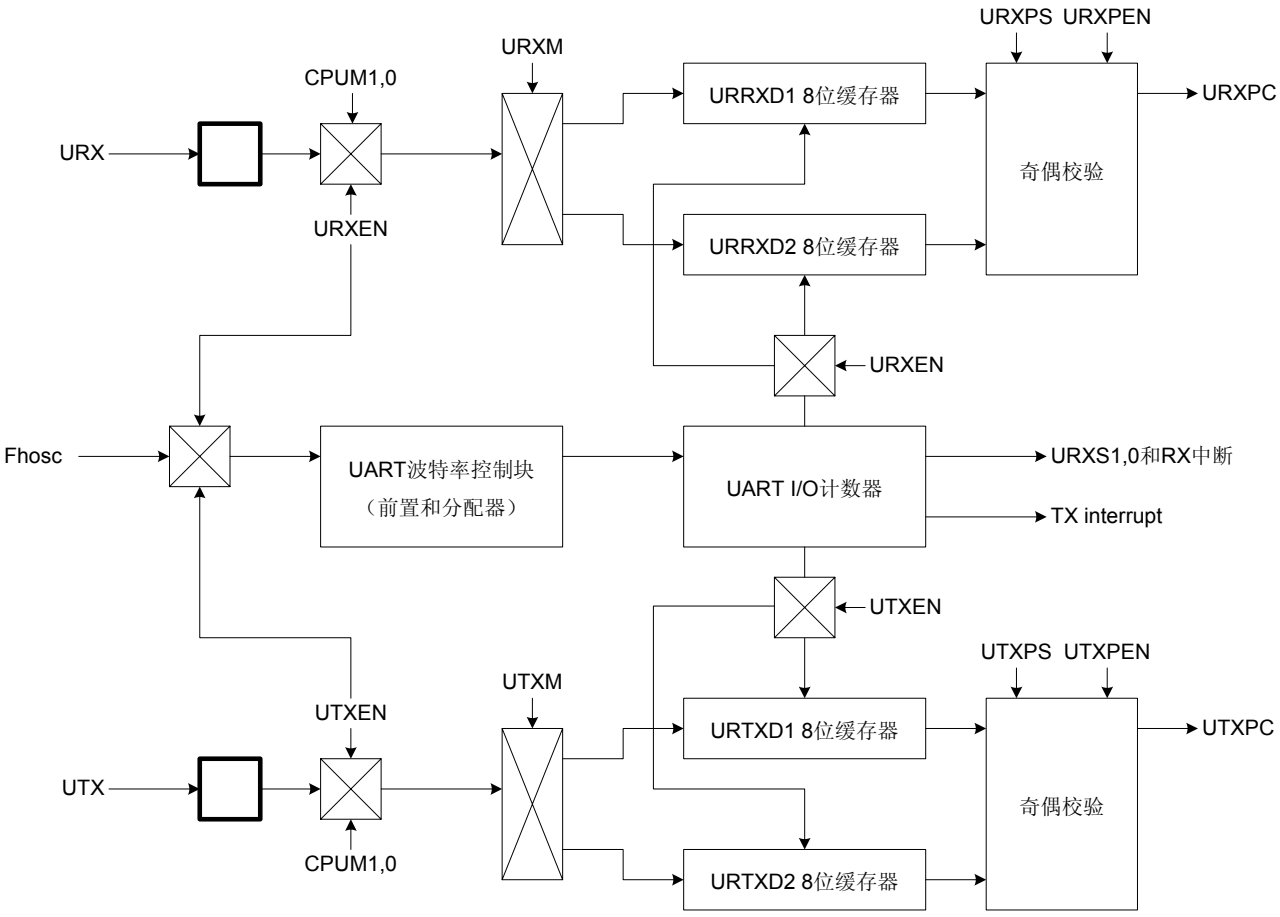
UART（通用异步收发器）可提供来传送低速数据及与低速外部设备通信。 SN8F2288 支持 RS232 规范，可传输 1 字节及 2 字节的数据，数据传送格式为：起始位，8/16 位数据位，校验位及停止位，并能设置各种波特率，在使用上灵活方便。UART I/O 引脚支持推挽式和漏极开漏结构，可以通过串行口控制寄存器设置。

UART 特性包括：

- **全双工，2 线异步数据传输；**
- **可编程波特率；**
- **支持 8 位和 16 位数据长度；**
- **奇偶校验位；**
- **传输结束中断。**

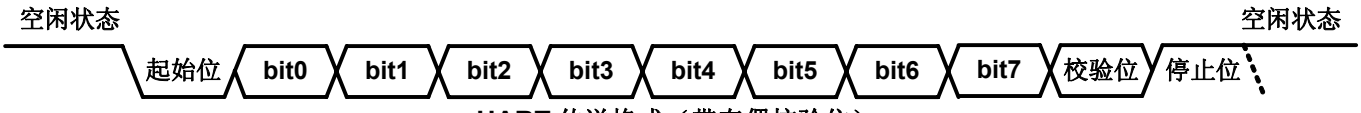
10.2 UART 操作

UART RX 和 TX 引脚与普通 I/O 引脚共享。使能 UART 功能时（RXDEN=1, TXDEN=1），共享引脚为 UART 功能引脚并自动禁止该引脚的 GPIO 功能。禁止 UART 功能时，该引脚返回到普通 I/O 模式，引脚状态改变为 UART 模式前状态。UART 数据缓存器支持 1 字节和 2 字节数据。UART 接收（RX）操作完成后，RXIRQ 置“1”；UART 发送（TX）操作完成后，TXIRQ 置“1”。UART 的 IRQ 位需要用程序清零。如果 RXIEN 或 TXIEN 为使能状态，RXIRQ 和 TXIRQ 触发中断请求，程序计数器将跳至中断向量处，执行中断服务程序。

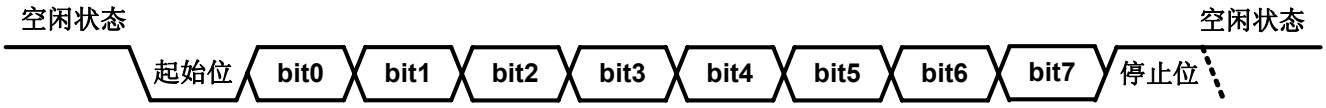


UART 接口电路示意图

UART 发送数据格式包括：空闲状态，起始位，8 位数据位，奇偶校验位和停止位，如下图所示：



UART 传送格式（带奇偶校验位）



UART 传送格式（无奇偶校验位）

总线空闲状态

总线空闲状态即总线上无任何操作的状态。单片机 UART 接收总线空闲状态为悬浮状态，由终端发送设备置高，发送总线空闲状态为高。UART 总线由 URXEN 和 UTXEN 使能后设置。

起始位

UART 为异步通信，因而需要向接收器提供一个标志位以说明传送开始的信息，起始位起到这样的作用，数据格式从高变低即标志传送开始，该信息位持续一位的时间。接收器可以通过起始位识别到将有数据传送来。

8 位数据位

数据格式为 8 位长度，发送起始位后，接着发送最低位（LSB）。数据位持续的时间为 UART 波特率一个单元的时间，由寄存器控制。

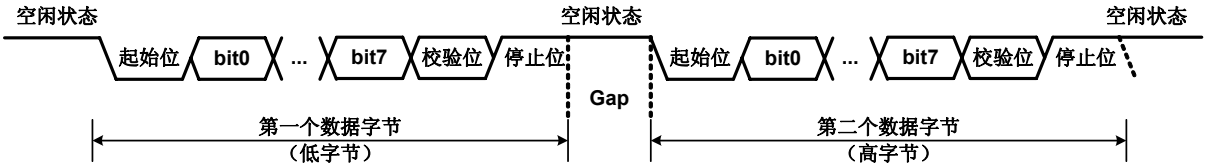
奇偶校验位

奇偶校验位用来检测数据传送是否出错，附加在需要发送数据之后。奇偶校验位的数据由校验方法得到，且由 URXPS/UTXPS 位控制。接收到数据及其校验位后，程序自动检查奇偶校验位是否正确，由 UPXPC 位显示校验结果。奇偶校验位功能由 URXPEN/UTXPEN 位控制。如果奇偶校验位功能禁止，UART 传送时将不传送奇偶检验位，发送完数据后接着发送停止位。

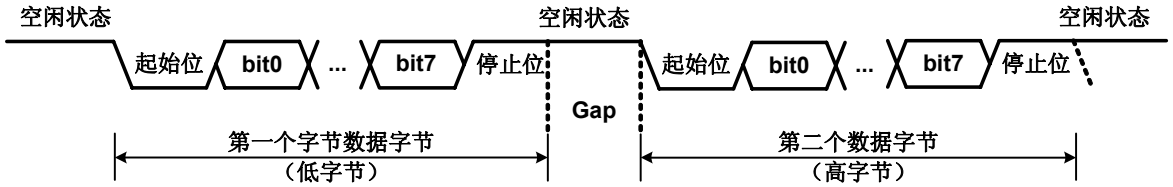
停止位

与起始位格式相同，停止位用来指示 UART 已传送结束。停止位以电平由低变高为准，并且持续一位的时间。

UART通讯支持2字节数据长度。它可以实时传输连续的数据。2 字节数据格式为连续字节数据格式。2 字节数据之间的间隙时间为波特率的一个单元。第一个数据字节存储在URRXD1（接收器）和URTXD1（发送器）。第二个数据字节存储在URRXD2和URTXD2（发送器）， 2 字节数据格式如下：



2 字节传送格式（带奇偶校验位）



2 字节传送格式（无奇偶校验位）

UART 支持中断功能，UTRXIEN/UTTXIEN 为传送数据中断功能控制位。UTRXIEN=0 时，禁止 UART 接收中断功能；UTTXIEN=0 时，禁止 UART 发送中断功能；UTRXIEN=1 时，使能 UART 接收中断功能；UTTXIEN=1 时，使能 UART 发送中断功能；UART 中断功能使能时，程序计数器指向中断向量 0008H 处，在 UART 操作完成后执行 UART 中断服务程序，UTRAIRQ 和 UTTXIRQ 是 UART 的中断请求标志位，在 UTRXIEN=0 或 UTTXIEN=0 使能时也可以显示 UART 操作状态，但必须由程序清零。UART 操作结束时，UTRXIRQ/UTTXIRQ 被置“1”。

✸ **注：UART 操作首先要设置 UART 各引脚的模式，使能 URXEN/UTXEN 引脚以控制 UART 引脚的模式。**

10.3 UART 发送控制寄存器

URTX 初始值 = 0xx0000x

0A9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URTX	UCLKS			UTXEN	UTXPEN	UTXPS	UTXM	
读/写	R/W			R/W	R/W	R/W	R/W	
复位后	0			0	0	0	0	

Bit 7 **UCLKS**：UART 时钟源选择位。
0 = UART 时钟源为 16MHz；
1 = UART 时钟源为 24MH。

Bit 4 **UTXEN**：UART 发送控制位。
0 = 禁止 UART 发送，UTX 引脚返回普通 I/O 功能；
1 = 使能 UART 发送，UTX 引脚发送 UART 数据。

Bit 3 **UTXPEN**：UART 发送校验位检测控制位。
0 = 禁止 UART 发送检验位检测功能，数据位后不发送校验位；
1 = 使能 UART 发送检验位检测功能，数据位后发送校验位。

Bit 2 **UTXPS**：UART 发送校验位格式控制位。
0 = UART 发送检验位为偶校验；
1 = UART 发送检验位为奇校验。

Bit 1 **UTXM**：UART 发送数据缓存器长度控制位。
0 = 1 字节；
1 = 2 字节。

10.4 UART 接收控制寄存器

URRX 初始值 = 0000000x

0AAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URRX	URXEN	URXS1	URXS0	URXPEN	URXPS	URXPC	URXM	
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
复位后	0	0	0	0	0	0	0	

Bit 7 **URXEN**：UART 接收控制位。
0 = 禁止 UART 接收，URX 引脚返回普通 I/O 功能；
1 = 使能 UART 接收，URX 引脚接收 UART 数据。

Bit[6:5] **URXS1, URXS0**：UART 接收状态指示位。
00 = 没有接收到数据；
01 = 接收到数据，但奇偶检验位出错；
10, 11 = 数据成功接收到。

Bit 4 **URXPEN**：UART 接收时的校验位检测控制位。
0 = 禁止 UART 检验位检测功能。数据位后不发送校验位。
1 = 使能 UART 校验位检测功能。数据位后发送校验位。

Bit 3 **URXPS**：UART 接收校验位格式控制位。
0 = UART 接收检验位为偶校验；
1 = UART 接收校验位为奇校验。

Bit 2 **URXPC**：UART 接收校验位状态检测指示位。
0 = UART 接收校验位出错；
1 = UART 接收校验位正确。

Bit 1 **URXM**：UART 接收数据缓存器长度控制位。
0 = 1 字节；
1 = 2 字节。

10.5 UART 波特率控制寄存器

URBRC 初始值 =11010101

0ABH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URBRC	UDIV4	UDIV3	UDIV2	UDIV1	UDIV0	UPCS2	UPCS1	UPCS0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	1	1	0	1	0	1	0	1

Bit[7:3] **UDIV[4:0]**：UART 波特率分频控制位。

Bit[2:0] **UPCS[2:0]**：UART 波特率预分频选择位。

UART 波特率时钟源为 Fhosc，可由预分频器分频后选择，计算方式如下：

UART 波特率 = (Fhosc / $2^{PreScaler}$ / (分频率+1)) / 16 ∈ 分频率≠0

波特率	UART 时钟源 = 16MHz （UCLKS = 0）		UART 时钟源 = 24MHz （UCLKS = 1）	
	UPCS[2:0]	UDIV[4:0]	UPCS[2:0]	UDIV[4:0]
1200	101	11010	-	-
2400	100	11010	-	-
4800	011	11010	-	-
9600	010	11010	-	-
19200	001	11001	-	-
38400	000	11001	-	-
51200	000	10011	-	-
57600	000	10000	-	-
102400	000	01001	-	-
115200	-	-	000	01100

10.6 UART 数据缓存器

URTXD1 初始值 = 00000000

0ACH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URTXD1	UTXD17	UTXD16	UTXD15	UTXD14	UTXD13	UTXD12	UTXD11	UTXD10
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **URTXD1**：UART 发送数据缓存器字节 1。

URTXD2 初始值 = 00000000

0ADH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URTXD2	UTXD27	UTXD26	UTXD25	UTXD24	UTXD23	UTXD22	UTXD21	UTXD20
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **URTXD2**：UART 发送数据缓存器字节 2。

URRXD1 初始值 = 00000000

0AEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URRXD1								
读/写	R	R	R	R	R	R	R	R
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **URRXD1**：UART 接收数据缓存器字节 1。

URRXD2 初始值 = 00000000

0AFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URRXD2								
读/写	R	R	R	R	R	R	R	R
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **URRXD2**：UART 接收数据缓存器字节 2。

UART 数据模式	URTXD2	URTXD1	URRXD2	URRXD1
1 字节	00H	1 字节数据	00H	1 字节数据
2 字节	高字节数据	低字节数据	高字节数据	低字节数据

11 串行输入/输出收发器（SIO）

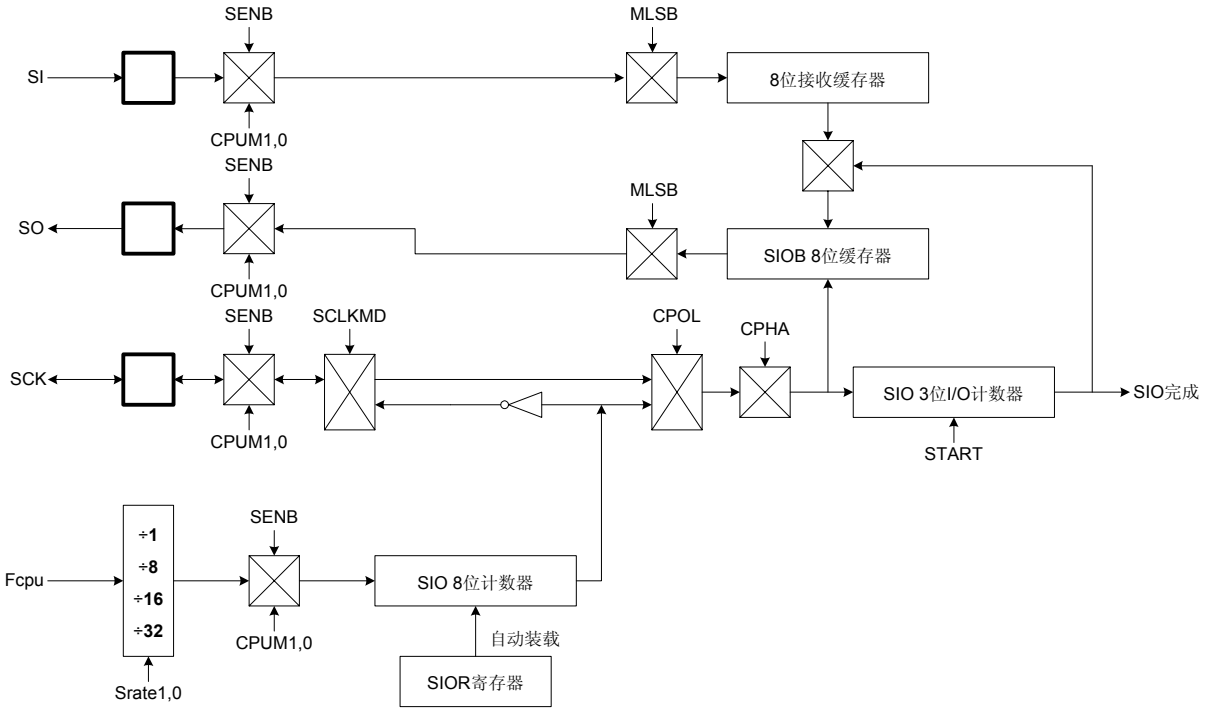
11.1 概述

串行输入/输出SIO收发器允许高速同步数据在SN8F2280系列单片机和外围装置之间或者几个SN8F2280装置之间传送。外围装置可以是：串行EEPROMs，移位寄存器，显示驱动芯片等。

SN8F2280的SIO特性包括：

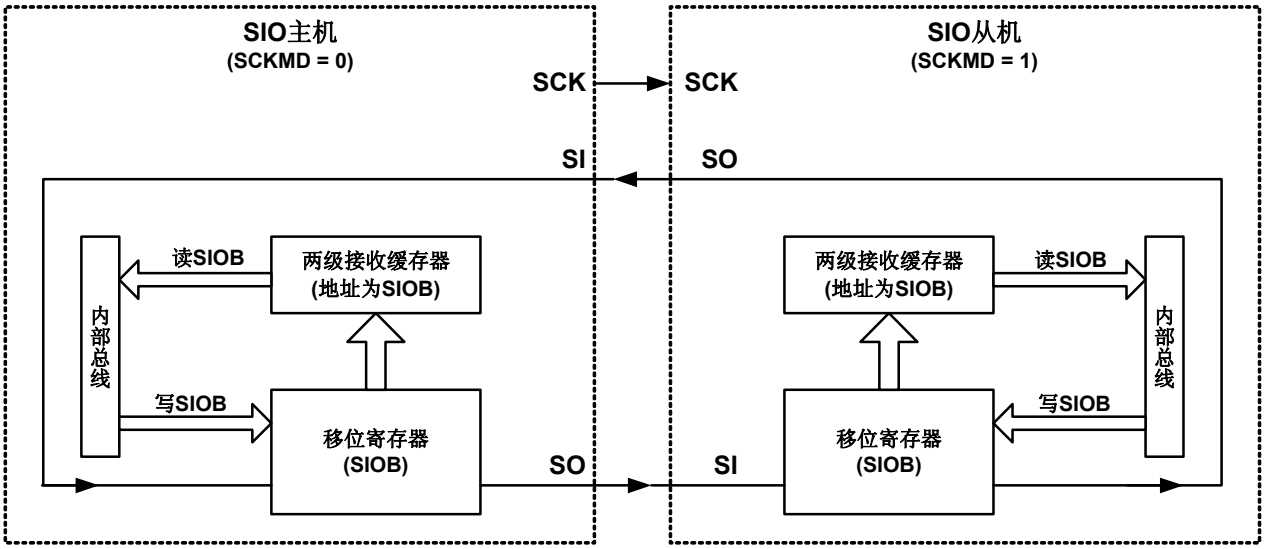
- 全双工 3 线同步传输；
- TX/RX 模式或单向 TX 模式；
- 主控模式（SCK 为时钟输出）或从动模式（SCK 为时钟输入）；
- MSB/LSB 数据优先传送；
- 在多路从动装置应用时，SDO（P2.1）是可编程漏极开路输出引脚；
- 主控模式时可设置数据传输速率；
- 传送结束时产生 SIO 中断。

寄存器SIOM用来控制SIO功能，如发送/接收、时钟速率、触发边沿等。通过设置寄存器SIOM的SENB和START位，SIO就可自动发送和接收8位数据。SIOB是一个8位数据缓存器，用于存储发送/接收的数据，SIOC和SIOR具有自动装载功能，能够产生SIO的时钟源。3位的I/O计数器可以监控SIO的操作，每接收/发送8位数据后，会产生一个中断请求。一次发送或接收结束后，SIO电路将自动禁止，可以通过重新编程SIOM寄存器启动下一次的数据传输。



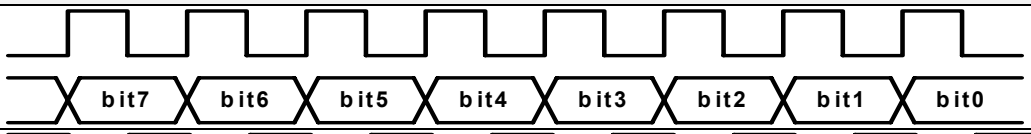
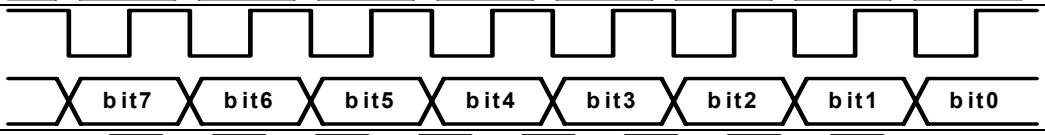
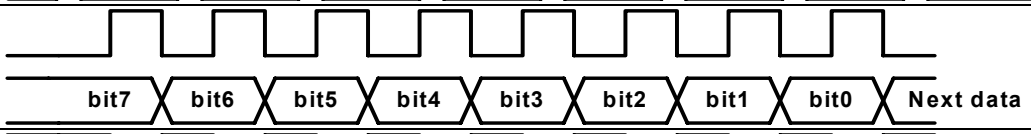
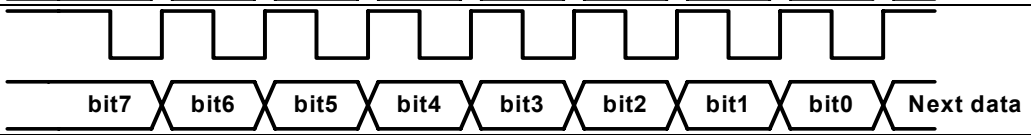
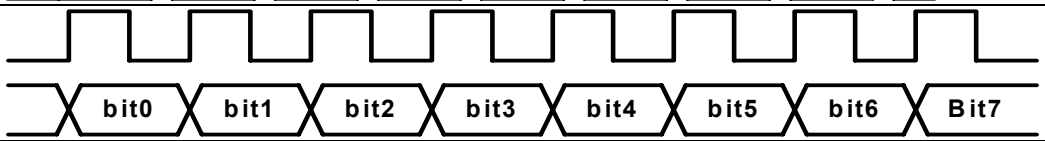
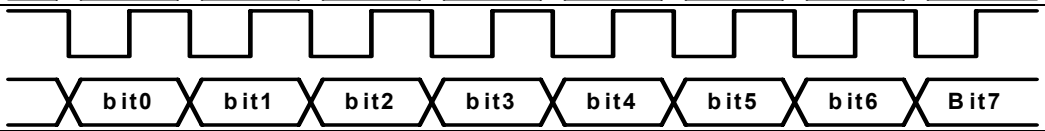
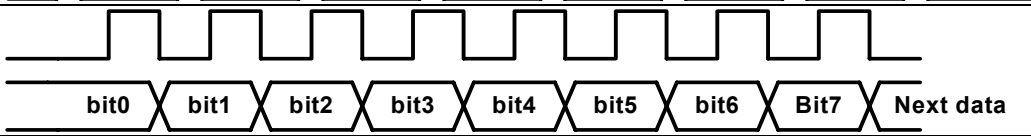
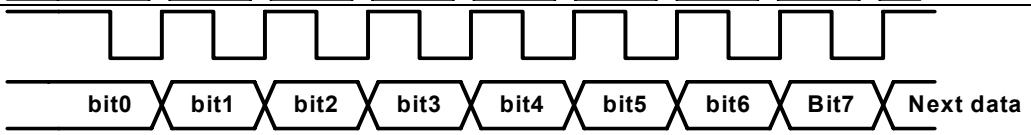
SIO 接口电路示意图

系统发送时使用一次缓存，而在接收时使用两次缓存。也就是说在整个移位周期结束前，新的数据不能写入SIOB数据寄存器中；而在接收数据时，在新的数据完全移入前，必须从SIOB数据寄存器中读出接收的数据，否则，前一个数据将会丢失。下图是一个典型的单片机之间的数据通信。由主控单片机发送SCK启动数据传输，两个单片机必须有相同的时钟沿触发方式，并将在同一时刻发送和接收数据。



SIO 数据传送示意图

SIO 数据传送时序如下所示：

MLSB	CPOL	CPHA	SCK 空闲状态	示意图
0	0	1	Low	
0	1	1	High	
0	0	0	Low	
0	1	0	High	
1	0	1	Low	
1	1	1	High	
1	0	0	Low	
1	1	0	High	

SIO 数据传送时序图

11.2 SIOM 模式寄存器

0B0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOM	SENB	START	SRATE1	SRATE0	MLSB	SCKMD	CPOL	CPHA
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **SENB**：SIO 功能控制位。
0 = 禁止（P2.0~P2.2 作为普通输入输出引脚）；
1 = 使能（P2.0~P2.2 作为SIO引脚）。

Bit 6 **START**：SIO 状态控制位。
0 = 传送结束；
1 = 传送过程中。

Bit [5:4] **SRATE1,0**：SIO 传送时钟分频位，**当 SCKMD=1，这两个位的功能是无效的。**
00 = Fcpu/2；
01 = Fcpu/64；
10 = Fcpu/32；
11 = Fcpu/16。

Bit 3 **MLSB**：MSB/LSB 传送优先控制位。
0 = MSB 优先传送；
1 = LSB 优先传送。

Bit 2 **SCKMD**：SIO 时钟模式选择位。
0 = 内部时钟（主控模式）；
1 = 外部时钟（从动模式）。

Bit 1 **CPOL**：SIO 传送时钟沿选择位。
0 = SCK 空闲状态是低电平状态；
1 = SCK 空闲状态是高电平状态；

Bit 0 **CPHA**：接收数据的时钟相位控制位。
0 = 在第一个时钟相位接收数据；
1 = 在第二个时钟相位接收数据。

✸ 注 1：在外部时钟模式时，若 SCKMD=1，则 SIO 处于从动模式；内部时钟时，若 SCKMD = 0 则为主控模式。
✸ 注 2：不能同时设置 SENB 和 START 位为 1，否则会导致 SIO 传送出错。

SIO 功能是与 P2 口共用：P2.0/SCK、P2.1/SDO、P2.2/SDI。下表列出了在使能或禁止 SIO 功能时，P2[2:0]的 I/O 口的模式状态和设置。

SENB=1 （使能 SIO 功能）		
P2.0/SCK	(SCKMD=1) SIO 时钟源来自外部时钟	P2.0 被自动设为输入模式，不管 P2M 如何设置
	(SCKMD=0) SIO 时钟源来自内部时钟	P2.0 被自动设为输出模式，不管 P2M 如何设置
P2.1/SDO	SIO =发送/接收	P2.1 被自动设为输出模式，不管 P2M 如何设置
P2.2/SDI	P2.2 必须设为输入模式，否则 SIO 功能会出错	
SENB=0（禁止 SIO 功能）		
P2.0/P2.1/P2.2	禁止 SIO 功能时，自动由 P2M 完全控制 P2[2:0]的 I/O 模式	

11.3 SIOB 数据缓存器

0B2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOB	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

SIOB是SIO的数据缓存器，存储串行发送/接收的数据。

11.4 SIOR 寄存器说明

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOR	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

寄存器SIOR用于SIO的自动装载，类似于后置分频器，能够提高SIO的时钟精度。用户可对SIOR进行写操作，从而控制SIO的通信速率。SIO的时钟频率计算如下：

SCK 频率 = SIO 速率 / (256 - SIOR)

SIOR=00H~0FDH

SIOR = 256 - (1 / (SCK 频率) * SIO 速率)

➢ **例：设置 SIO 时钟频率为 2MHz，Fosc = 12MHz，SIO’s 速率 = Fcpu/2，Fcpu=Fosc/1=12MHz。**

SIOR = 256 –（1/（2MHz）* 12MHz/2）
= 256 – 3
= 253
= 0FDH

➢ **例：主控模式下，上升沿发送/接收数据。**

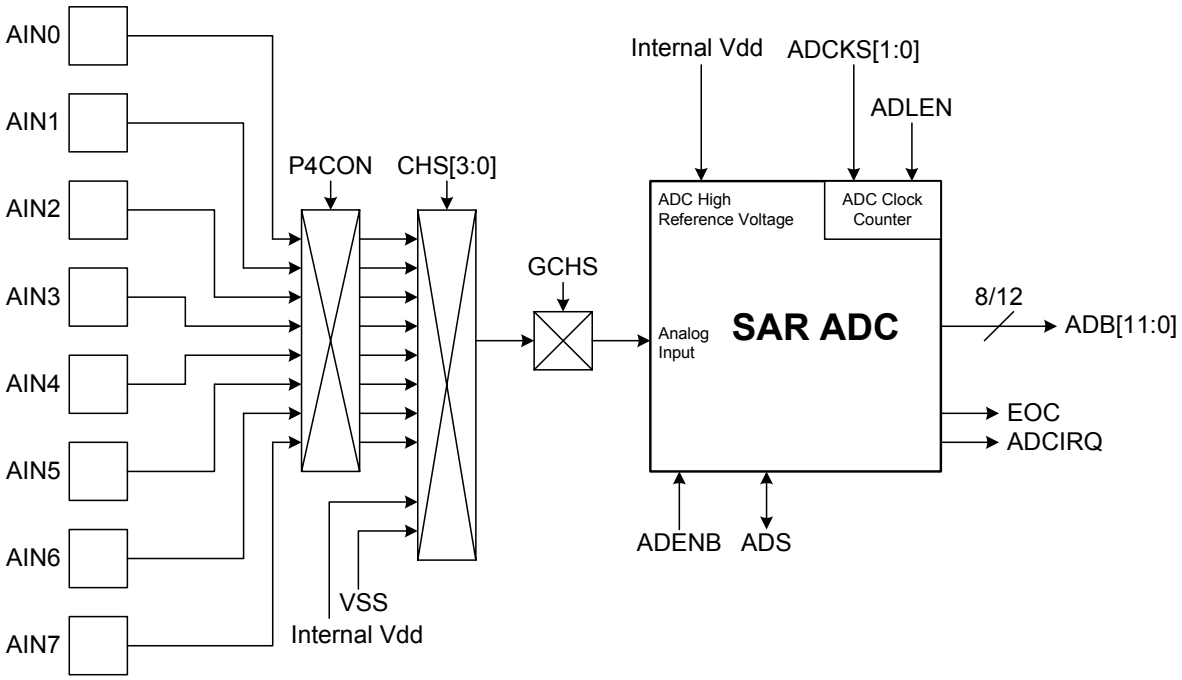
```
        MOV     A,TXDATA        ; 将需要传送的数据存入 SIOB 寄存器。
        B0MOV   SIOB,A
        MOV     A,#0FEH         ; 设置 SIO 时钟。
        B0MOV   SIOR,A
        MOV     A,#10000000B    ; 设置 SIOM 寄存器并使能 SIO 功能。
        B0MOV   SIOM,A
        B0BSET  FSTART          ; 开始传送并接收 SIO 数据。
CHK_END:
        B0BTS0  FSTART          ; 等待 SIO 结束。
        JMP     CHK_END
        B0MOV   A,SIOB          ; 保存 SIOB 的数据到 RXDATA 缓存器中。
        MOV     RXDATA,A
```

➢ **例：从动模式下，上升沿发送/接收数据。**

```
        MOV     A,TXDATA        ; 将需要传送的数据存入 SIOB 寄存器。
        B0MOV   SIOB,A
        MOV     A,# 10000100B   ; 设置 SIOM 寄存器并使能 SIO 功能。
        B0MOV   SIOM,A
        B0BSET  FSTART          ; 开始传送并接收 SIO 数据。
CHK_END:
        B0BTS0  FSTART          ; 等待 SIO 结束。
        JMP     CHK_END
        B0MOV   A,SIOB          ; 保存 SIOB 的数据到 RXDATA 缓存器中。
        MOV     RXDATA,A
```

12 8 通道 ADC

12.1 概述



模拟数字转换（ADC）是一个 SAR 结构，内置 8 个模拟通道（AIN0~AIN7），高达 4096 阶的分辨率，能将一个模拟信号转换成相应的 12 位数字信号。通过 CHS[2:0]选择模拟信号输入引脚（AIN 引脚），GCHS 位使能全局 ADC 通道，模拟信号通过 SAR ADC 输入。ADC 的分辨率可以选择 8 位或者 12 位，由 ADLEN 位决定。且由 ADCKS[1:0]控制 ADC 的转换速率以决定 ADC 的转换时间。ADC 内置 P4CON 寄存器来设置模拟输入引脚，必须由程序将 P4 设为输入引脚。设置好 ADENB 和 ADS 位后，ADC 开始转换，转换结束时，ADC 电路将 EOC 和 ADCIRQ 置 1，并将转换结果存入 ADB 和 ADR 寄存器中。若 ADCIEN=1，ADC 请求中断，AD 转换完成后，ADCIRQ=1 时，程序计数器跳转中断向量地址执行中断服务程序。

✸ **注：**
1.将 ADC 输入引脚设置为输入模式，且无上拉电阻。
2.进入睡眠模式前禁止 ADC（设置 ADENB=0）以省电。
3.设置号 P4CON 寄存器的相关位，以避免在睡眠模式下产生额外的功耗。
4.使能 ADC 后（设置 ADENB=1）延时 100us 以等待 ADC 电路准备进行转换。

12.2 ADM 寄存器

0B6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	CHS3	CHS2	CHS1	CHS0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **ADENB**：ADC 控制位。
0 = 禁止 ADC；
1 = 使能 ADC。

Bit 6 **ADS**：ADC 启动控制位。
0 = 停止 AD 转换；
1 = 开始 AD 转换。

Bit 5 **EOC**：ADC 状态位。
0 = AD 转换中；
1 = AD 转换结束，ADS 位复位。

Bit 4 **GCHS**：ADC 全局通道控制位。
0 = 禁止 AIN 通道；
1 = 使能 AIN 通道。

Bit [3:0] **CHS[3:0]**：ADC 输入通道选择位。
0000 = AIN0； 0001 = AIN1；
0010 = AIN2； 0011 = AIN3；
0100 = AIN4； 0101 = AIN5；
0110 = AIN6； 0111 = AIN7；
1000 = VSS； 1001 = VDD。

12.3 ADR 寄存器

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	ADCKS2	ADCKS1	ADCKS0	ADLEN	ADB3	ADB2	ADB1	ADB0
读/写	R/W	R/W	R/W	R/W	R	R	R	R
复位后	0	0	0	0	-	-	-	-

Bit [7:5] **ADCKS [2:0]**：ADC 时钟源选择位。

ADCKS2	ADCKS1	ADCKS0	ADC Clock Source
0	0	0	Fcpu/16
0	0	1	Fcpu/8
0	1	0	Fcpu/1
0	1	1	Fcpu/2
1	0	0	Fcpu/64
1	0	1	Fcpu/32
1	1	0	Fcpu/4
1	1	1	保留

Bit 4 **ADLEN**：ADC 分辨率选择位。
0 = 8-位；
1 = 12 位。

Bit [3:0] **ADB [3:0]**：12 位数据缓存器的低字节数据。

12.4 ADB 寄存器

0B7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADB	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4
读/写	R	R	R	R	R	R	R	R
复位后	-	-	-	-	-	-	-	-

Bit[7:0] **ADB[7:0]**：8 位 ADC 数据缓存器，12 位 ADC 数据缓存器的高字节数据。

ADB 为 ADC 数据缓存器，用来存储 ADC 的转换结果，ADB 寄存器只有 8 位，可以存储 12 位 ADC 的 bit4~bit11。将 ADB 寄存器和 ADR 寄存器的低 4 位的数据合起来就是 12 位 ADC。ADC 缓存器是只读寄存器，其初始值是未定的。

- **ADB[11:4]：8 位 ADC 的转换结果存储在 ADB 寄存器中。**
- **ADB[11:0]：12 位 ADC 的转换结果存储在 ADB 寄存器和 ADR 寄存器中。**

✷ **注：ADC 数据缓存器（包括 ADB 寄存器和 ADR 寄存器的低 4 位）的初始值是未定的。**

AIN 输入电压 v.s. ADB 输出数据

AIN n	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
0/4096*VREFH	0	0	0	0	0	0	0	0	0	0	0	0
1/4096*VREFH	0	0	0	0	0	0	0	0	0	0	0	1
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
4094/4096*VREFH	1	1	1	1	1	1	1	1	1	1	1	0
4095/4096*VREFH	1	1	1	1	1	1	1	1	1	1	1	1

针对不同的应用，用户可能需要精度介于 8 位到 12 位之间的 AD 转换器。对于这种情况，可以通过对保存在 ADR 和 ADB 中的转换结果进行处理得到。首先，用户必须选择 12 位分辨率的模式，进行 AD 转换，然后在转换结果中去掉最低的几位得到需要的结果。如下表所示：

ADC 分辨率	ADB								ADR			
	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
8-bit	O	O	O	O	O	O	O	O	x	x	x	x
9-bit	O	O	O	O	O	O	O	O	O	x	x	x
10-bit	O	O	O	O	O	O	O	O	O	O	x	x
11-bit	O	O	O	O	O	O	O	O	O	O	O	x
12-bit	O	O	O	O	O	O	O	O	O	O	O	O
O = Selected, x = Delete												

12.5 P4CON 寄存器

P4 口和 ADC 的输入口共享。同一时间只能设置 P4 口的一个引脚作为 ADC 的测量信号输入口（通过 ADM 寄存器来设置），其它引脚则作为普通 I/O 使用。具体应用中，当向 CMOS 结构端口输入一个模拟信号，尤其当模拟信号为 1/2 VDD 时，将可能产生额外的漏电流。在睡眠模式下，上述漏电流会严重影响到系统的整体功耗。尤其当 P4 口外接多个模拟信号时，也会产生额外的漏电流。P4CON 为 P4 口的配置寄存器。将 P4CON[7:0]置“1”，其对应的 P4 口将被设置为纯模拟信号输入口，从而避免上述漏电流的情况。

0B9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4CON	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **P4CON[7:0]**：P4.n 配置控制位。
0 = P4.n 作为模拟输入（ADC 输入）引脚或者数字 I/O 引脚；
1 = P4.n 只能作为单纯的模拟输入引脚，不能作为数字 I/O 引脚。

✸ **注：当 P4.n 为普通 I/O 而不是 ADC 通道时，P4CON.n 必须置“0”，否则 P4.n 的数字 I/O 信号会被隔离。**

12.6 ADC 转换时间

ADC 转换时间是指从 ADS=1（开始 ADC）到 EOC=1（ADC 结束）所用的时间，由 ADC 分辨率控制，12 位 ADC 的转换时间为 1/（ADC 时钟/4）*16 S；8 位 ADC 的转换时间为 1/（ADC 时钟/4）*12 S。**ADC 的转换时间会影响 ADC 的性能，如果输入高 Rate 的模拟信号，必须要选择一个高 Rate 的 ADC 转换 Rate。如果 ADC 的转换时间比模拟信号的转换 Rate 慢，则 ADC 的结果出错。故选择合适的 ADC 时钟 Rat 和 ADC 分辨率才能得到合适的 ADC 转换 Rate。**

12 位 AD 转换时间 = 1/(ADC 时钟 /4)*16 sec

8 位 AD 转换时间 = 1/(ADC 时钟 /4)*12 sec

Fcpu = 4MHz（高速时钟振荡器频率（Fosc）为 16MHz，Fcpu=Fosc/4）

ADLEN	ADCKS1	ADCKS0	ADC 时钟	AD 转换时间
0（8 位）	0	0	Fcpu/16	1/(4MHz/16/4)*12 = 192 us
	0	1	Fcpu/8	1/(4MHz/8/4)*12 = 96 us
	1	0	Fcpu	1/(4MHz/4)*12 = 12 us
	1	1	Fcpu/2	1/(4MHz/2/4)*12 = 24 us
1（12 位）	0	0	Fcpu/16	1/(4MHz/16/4)*16 = 256 us
	0	1	Fcpu/8	1/(4MHz/8/4)*16 = 128 us
	1	0	Fcpu	1/(4MHz/4)*16 = 16 us
	1	1	Fcpu/2	1/(4MHz/2/4)*16 = 32 us

12.7 ADC 注意事项

12.7.1 ADC 信号

ADC 的高参考电压为内部 Vdd，ADC 的低参考电压为 GND。

ADC 输入信号的电压范围必须在高参考电压和低参考电压之间。

12.7.2 ADC 编程注意事项

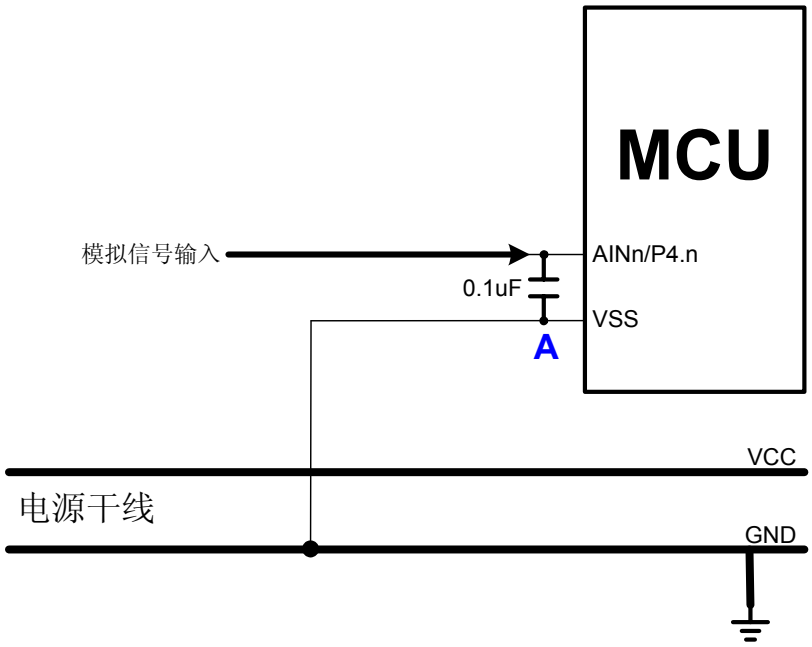
执行 ADC 首先要设置 ADC 的配置，ADC 编程的设置流程和注意事项如下：

- **步骤 1：**使能 ADC。由 ADENB 位控制，ADENB=1 使能 ADC；ADENB=0 禁止 ADC。**当使能 ADENB 位后，系统必须延时 100us 作为 ADC 的准备时间，然后设置 ADS，开始 AD 转换。设置 ADENB（不是 ADS）后的 100us 的延迟时间是必需的，否则 ADC 转换结果会出错。**通常情况下，在系统正常运行时，只需设置 ADENB 一次，故也只有一次的延时。
- **步骤 2：**通过 CHS[2:0]选择 ADC 输入引脚，使能 P4CON 的相关位以选择 ADC 输入引脚，并使能 ADC 全局输入。**当一个 AIN 引脚设置为模拟信号输入引脚时，该引脚必须设置为输入模式，并由程序禁止该引脚的上拉电阻。同时也需设置 P4CON，普通 I/O 功能包括上拉电阻都会被隔离。**
- **步骤 3：**设置 ADS=1 后，AD 开始转换。
- **步骤 4：**通过检测 EOC=1 或者 ADCIRQ=1，等待 ADC 完成。如使能 ADC 中断功能，当 ADC 中断发生时，系统执行 ADC 中断。**当 ADC 完成后，自动将 ADS 清零，EOC 显示 ADC 转换过程中的实时状态，当 ADS=1 时，将 EOC 清零，用户不需由程序清零 EOC。**

➢ **例：设置 AIN0 为 12 位 ADC 输入通道，进入睡眠模式后进行 AD 转换。**

```
ADC0:
        B0BSET      FADENB          ; 使能 ADC 电路。
        CALL        Delay100uS      ; 延时 100us 等待 ADC 电路准备以进行转换。
        MOV         A, #0FEh
        B0MOV       P4UR, A         ; 禁止 P4.0 的上拉电阻。
        B0BCLR      FP40M           ; 设置 P4.0 为输入引脚。
        MOV         A, #01h
        B0MOV       P4CON, A        ; 设置 P4.0 为纯模拟输入引脚。
        MOV         A, #60H
        B0MOV       ADR, A          ; 设置 12 位 ADC，ADC 时钟为 Fosc。
        MOV         A,#90H
        B0MOV       ADM,A           ; 使能 ADC，设置 AIN0 输入。
        B0BSET      FADS            ; 开始转换。
WADC0:
        B0BTS1      FEOC            ; 若转换结束=1，跳过。
        JMP         WADC0           ; 否则，跳到 WADC0。
        B0MOV       A,ADB           ; 获得 AIN0 输入数据 bit11 ~ bit4。
        B0MOV       Adc_Buf_Hi, A
        B0MOV       A,ADR           ; 获得 AIN0 输入数据 bit3 ~ bit0。
        AND         A, 0Fh
        B0MOV       Adc_Buf_Low, A
Power_Down  .       .
        B0BCLR      FADENB          ; 禁止 ADC 电路。
        B0BCLR      FCPUM1
        B0BSET      FCPUM0          ; 进入数目模式。
```

12.8 ADC 电路



模拟信号从 ADC 输入引脚 AINn/P4.n 输入。在 ADC 输入引脚和 VSS 之间（A）必须连接一个 0.1uF 的电容，且要尽可能的靠近 ADC 输入引脚。不能将电容的 GND 直接连接到电源干线上的 GND，必须通过 VSS 引脚。该电容可以减少电源干扰对模拟信号的影响。

13 MSP

13.1 概述

MSP 是一个串行的通信接口，用来进行几个单片机之间或者外围殷鉴之间的数据交换。外围设备可以是串行 EEPROM、ADC，显示设备等。

MSP 模块可以在下面模式下进行操作：

- **全主控模式；**
- **从动模式（无通用地址调用）。**

MSP 的主要性能如下：

- **2 线同步数据发送/接收；**
- **主控（SCL 为时钟输出）/从动（SCL 为时钟输入）操作；**
- **在多路从动设备应用时，SCL、SDA 为可编程漏极开路输出引脚；**
- **指示 400K 的时钟频率@Fcpu=4MIPs；**
- **发送/接收结束时产生 MSP 中断。**

13.2 MSP 状态寄存器

MSPSTAT 初始值 =x00000x0

0EAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPSTAT	-	CKE	D_A	P	S	RED_WRT	-	BF
读/写	-	R/W	R	R	R	R	-	R
复位后	-	0	0	0	0	0	-	0

Bit 6 **CKE**：从机边沿控制位。
从动模式下接收地址或数据字节。
0 = SCL 上升沿有效（默认）；
1 = SCL 下降沿有效。

✸ **注 1：从动模式下发送数据时，接收地址取决于 CKE 的设置；在 SCL 下降沿发送数据。**
✸ **注 2：从动模式下接收数据时，接收的地址和数据都由 CKE 设置。**

Bit 5 **D_A**：数据/地址标志位。
0 = 表示接收或者发送的上一字节是地址；
1 = 表示接收或者发送的上一字节是数据。

Bit 4 **P**：停止标志位。
0 = 没有检测到停止位；
1 = 检测到停止位。

✸ **注 1：检测到起始位后将停止位清零。**

Bit 3 **S**：起始标志位。
0 = 没有检测到起始位；
1 = 检测到起始位。

✸ **注 1：检测到停止位后将起始位清零。**

Bit 2 **RED_WRT**：读/写标志位。该标志位是和上一个地址信息的读写标志位相匹配的，只在该区间有效，当下一个 START 命令、STOP 命令或者 NO ACK 命令出现时，自动无效。
从动模式下：0 = 写；1 = 读。
主控模式下：0 = 没有发送；1 = 发送过程中。
该位和 SEN、RSEN、PEN、RCEN、ACKEN 位指明 MSP 是否在 IDLE 状态。

Bit 0 **BF**：缓存器状态位。
接收：1 = 接收完成，MSPBUF 填满；
0 = 接收没有完成，MSPBUF 空置。
发送：1 = 正在发送数据（不包括 ACK 和停止位），MSPBUF 填满；
0 = 完成发送数据（不包括 ACK 和停止位），MSPBUF 空置。

13.3 MSP 模式寄存器 1

MSPM1 初始值 =000000x0

0EBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPM1	WCOL	MSPOV	MSPENB	CKP	SLRXCKP	MSPWK		MSPC
读/写	R/W	R/W	R/W	R/W	R/W	R/W		R/W
复位后	0	0	0	0	0	0		0

Bit 7 **WCOL**：写数据检测标志位。
主控模式：0 = 未检测到有数据写入 MSPBUF；
1 = MSP 通讯还未开始，已经写入数据到 MSPBUF 寄存器，此位被置 1。
从动模式：0 = 未检测到有数据写入 MSPBUF；
1 = 正在发送上一个数据时，有写入新的数据到 MSPBUF。（必须由软件清零）

Bit 6 **MSPOV**：接收溢出显示位。
0 = 没有溢出；
1 = 当 MSPBUF 仍在处理前一个数据时，接收到新数据，此时此位被置 1，发送模式下 MSPOV 可以忽略，任一模式下都必须在软件下将该位清零。

Bit 5 **MSPENB**：MSP 通信使能位。
0 = 禁止 MSP，相关引脚为普通 I/O 引脚；
1 = 使能 MSP，相关引脚为 SCL、SDA 串口引脚。

Bit 4 **CKP**：SCL 时钟优先控制位。
MSP 从动模式：
0 = SCL 保持低电平（保证数据设置时间和从动设备准备就绪）；
1 = 释放 SCL 时钟（从动发送模式时时钟使能 CKP 功能，从动接收模式时由 SLRXCKP 控制 CKP 功能）。
MSP 主控模式：无效。

Bit 3 **SLRXCKP**：从动接收模式时 SCL 时钟优先控制位。
MSP 从动接收模式：
0 = 禁止 CKP 功能；
1 = 使能 CKP 功能。
MSP 主控和从动发送模式：无效。

Bit 2 **MSPWK**：MSP 唤醒显示位。
0 = MSP 没有唤醒单片机；
1 = MSP 唤醒单片机。

✸ **注：在进入睡眠模式之前将 MSPWK 清零，以便在唤醒后知道是否由 MSP 唤醒。**

Bit 0 **MSPC**：MSP 模式控制位。
0 = MSP 在从动模式下工作，7 位地址；
1 = MSP 在主控模式下工作。

13.4 MSP 模式寄存器 2

MSPM2 初始值 =00000000

0ECH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPM2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **GCEN**：通用地址应答使能位（仅在从动模式下有效）。
0 = 禁止通用地址应答功能；
1 = 使能通用地址应答功能和中断功能。

Bit 6 **ACKSTAT**：应答状态位（仅在主控模式下有效）。
主控发送模式下：
0 = 接收到从动端的应答；
1 = 没有接收到从动端的应答。

Bit 5 **ACKDT**：应答数据位（仅在主控模式下有效）。
主控接收模式下：在接收到一个字节数据后，执行应答操作时，此位的数据将被发送。
0 = 应答；
1 = 无应答。

Bit 4 **ACKEN**：应答流程使能位（仅在主控模式下有效）。
主控接收模式下：
0 = 应答流程空闲；
1 = 在 SDA 和 SCL 引脚开始应答流程，发送 ACKDT 位，自动由硬件清零。

✸ **注：若 MSP 模块不在空闲模式，则该位不会被设置（脱机），不能写入数据到 MSPBUF（禁止写入 MSPBUF）。**

Bit 3 **RCEN**：接收使能位（仅在主控模式下有效）。
0 = 接收空闲；
1 = 使能 MSP 接收模式。

✸ **注：若 MSP 模块不在空闲模式，则该位不会被设置（脱机），不能写入数据到 MSPBUF（禁止写入 MSPBUF）。**

Bit 2 **PEN**：停止条件使能位（仅在主控模式下有效）。
0 = 停止条件空闲；
1 = 初始化 SDA/SCL 停止信号，由硬件自动清零。

✸ **注：若 MSP 模块不在空闲模式，则该位不会被设置（脱机），不能写入数据到 MSPBUF（禁止写入 MSPBUF）。**

Bit 1 **RSEN**：重复起始条件使能位（仅在主控模式下有效）。
0 = 重复起始条件空闲；
1 = 初始化重新发送 SDA/SCL 信号，由硬件自动清零。

✸ **注：若 MSP 模块不在空闲模式，则该位不会被设置（脱机），不能写入数据到 MSPBUF（禁止写入 MSPBUF）。**

Bit 0 **SEN**：起始条件使能位（仅在主控模式下有效）。
0 = 起始条件空闲；
1 = 在 SDA 和 SCL 引脚显示起始条件，由硬件自动清零。

✸ **注：若 MSP 模块不在空闲模式，则该位不会被设置（脱机），不能写入数据到 MSPBUF（禁止写入 MSPBUF）。**

13.5 MSP 缓存寄存器

MSPBUF 初始值 =00000000

0EDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPBUF	MSPBUF7	MSPBUF6	MSPBUF5	MSPBUF4	MSPBUF3	MSPBUF2	MSPBUF1	MSPBUF0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **MSPBUF**：MSP 缓存器。

13.6 MSP 地址寄存器

MSPADR 初始值=00000000

0EEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPADR	MSPADR7	MSPADR6	MSPADR5	MSPADR4	MSPADR3	MSPADR2	MSPADR1	MSPADR0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **MSPADR**：MSP 地址。

13.7 从动模式操作

当地址数据被发送或地址接收到，硬件将自动产生 ACK 信号，将接收到的数据从 MSPSR 装载到 MSPBUF（MSP 缓存寄存器）中。

在下列情形下，MSP 功能不会应答 ACK_信号：

- **数据缓存器全满：接收到另一组发送的内容，BF = 1（MSPSTAT bit 0）。**
- **数据溢出：接收到另一组发送的内容，MSP0V=1（MSPM1 bit 6）。**

BF=1，即 MCU 还未读取 MSPBUF 的数据，故 MSPSR 中的数据还未装入 MSPBUF，但此时 MSPIRQ 和 MSPOV 位仍置 1，当读取 MSPBUF 寄存器后，BF 自动被清零，MSPOV 必须由软件清零。

13.7.1 寻址

使能 MSP 从动功能时，需等待产生一个 START 信号，START 信号产生后，8 位地址装入 MSPSR 寄存器，MSPSR[7:1] 的数据与 MSPADR 寄存器在 SCL 脉冲的第 8 个下降沿进行比较，若地址相同，BF 和 MSPOV 都清零，产生下列事件：

1、 在 SCL 的第 8 个下降沿，MSPSR 寄存器的数据装入 MSPBFU。
2、 在 SCL 的第 8 个下降沿，缓存器状态位 BF 置 1。
3、 产生一个 ACK_信号。
4、 在 SCL 的第 9 个下降沿，MSP 中断请求 MSPIRQ 置 1。

接收数据时的状态		MSPSP→ MSPBUF	响应 ACK_信号	设置 MSPIRQ
BF	MSPOV			
0	0	是	是	是
*0	*1	是	否	是
1	0	否	否	是
1	1	否	否	是

接收数据效果表

注：BF=0，MSPOV=1 显示出软件设置不合适，清除溢出寄存器。

13.7.2 从动模式接收数据

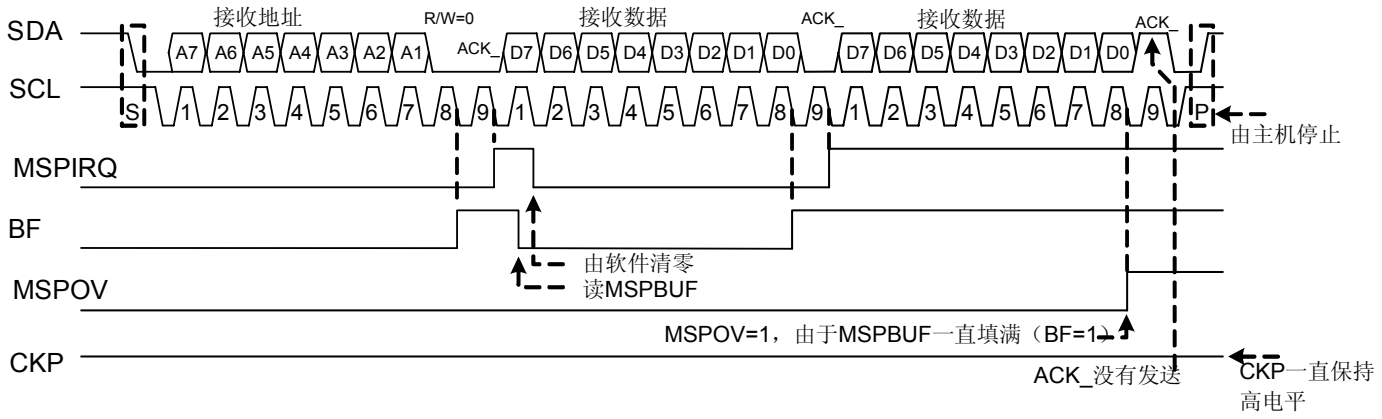
当地址字节的 R/W 位=0 时，地址匹配时，MSPSTAT 的 R/W 位被清零，该地址装入 MSPBUF。响应 ACK_信号后，MSP 每 8 个时钟接收一次数据，由 SCRXCKP 位控制是否使能 CKP 功能，由 CKE 位控制数据的边沿锁存功能－上升沿或下降沿。

当溢出时，不管 BF=1 还是 MSPOV=1，都没有应答信号。

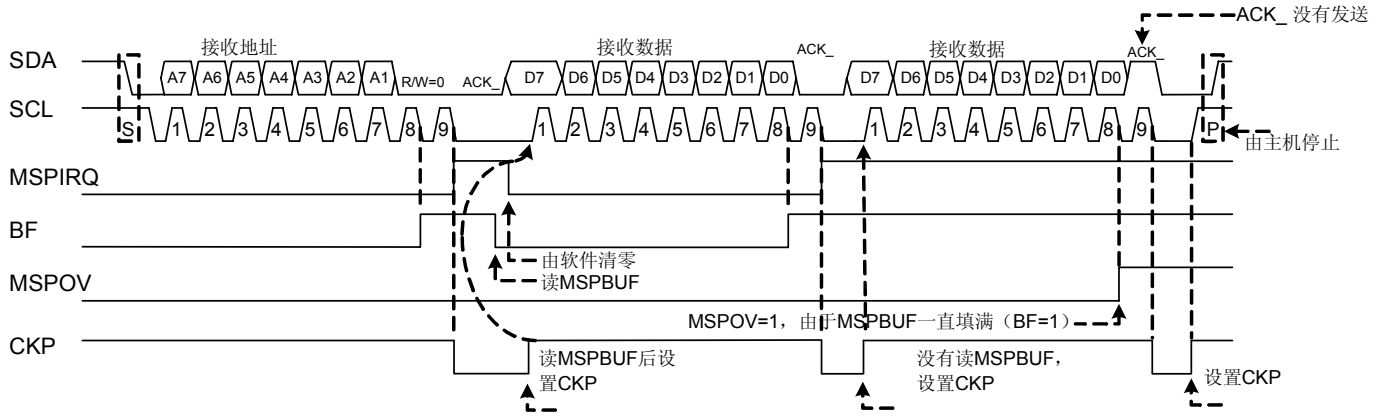
每发送一次数据会产生一次 MSP 中断，必须由软件清 MSPIRQ。

从动模式接收示意图如下所示：

SLRXCKP=0



SLRXCKP=1

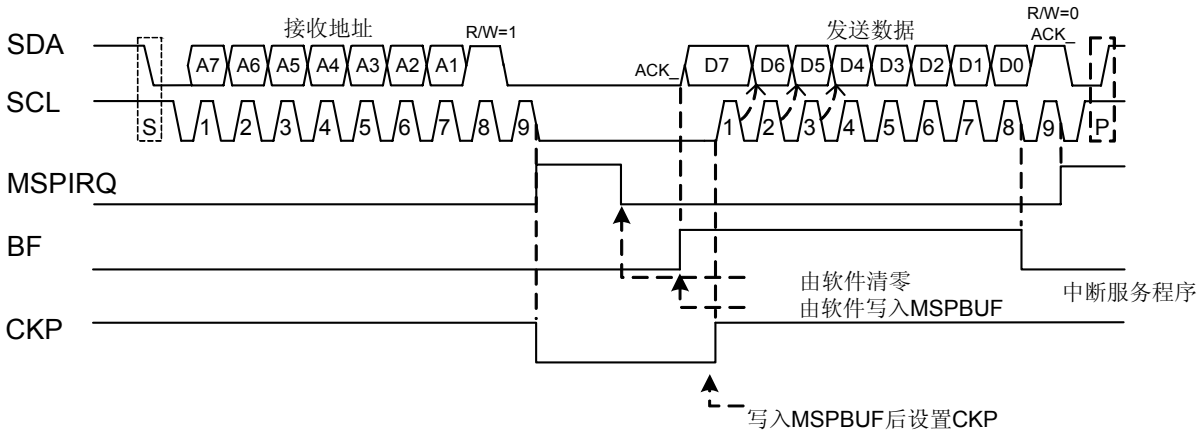


13.7.3 从动模式发送数据

匹配地址后，设置 R/W 位，MSPSTAT bit 2 R/W 置 1，接收到的地址装入 MSPBUF，然后在第 9 个时钟发送一个 ACK 信号，SCL 保持低电平。发送的数据装入 MSPBUF，MSPBUF 装入 MSPSR 寄存器。主控装置监控 SCL 引脚信号，从动设备保持 CKP 低电平。装入 MSPBUF 后，设置 CKP 位，MSPBUF 数据在 SCL 信号的下降沿移出，确保 SDA 信号在 SCL 高电平状态是有效的。

每个字节发送时产生 MSP 中断，中断请求 MSPIRQ 在第 9 个 SCL 时钟置 1，由软件清零。MSPSTAT 寄存器可以监控数据发送的状态。

从动发送模式下，从主机接收的 ACK_信号在第 9 个 SCL 时钟的上升沿被锁存，如 ACK_为高电平，发送完成，从动设备逻辑复位并等待另一个 START 信号。如 ACK_为低电平，从设备应该重新导入 MSPBUF，设置 CKP=1，重新开始发送数据。



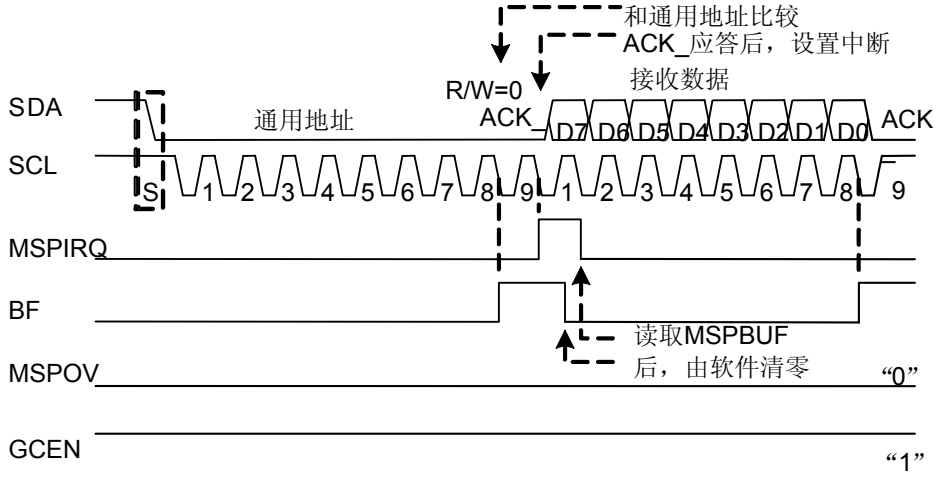
MSP 从动模式发送时序图

13.7.4 通用地址应答

MSP 总线的第一个 7 字节为从动地址，只有该地址和 MSPADR 的数据相符时才会响应一个 ACK_信号。现有一个通用地址可以作为所有从动设备的硬件地址，发生该地址时，所有的从动设备都响应一个应答信号。

通用地址是一个全为 0 的特殊地址。通用地址功能由 GCEN 位控制。该位置 1 则使能通用地址功能，反之则禁止。当 GCEN=1，START 信号后，8 位地址装入 MAPSR，并与 MSPADR 进行比较，且通用地址由硬件固定。

如通用地址匹配，MSPSR 数据发送至 MSPBUF，BF 置 1，在第 9 个时钟（ACK_）下降沿请求 MSP 中断，执行中断时，读取 MSPBUF，检测该地址是否为通用地址或者特殊地址。

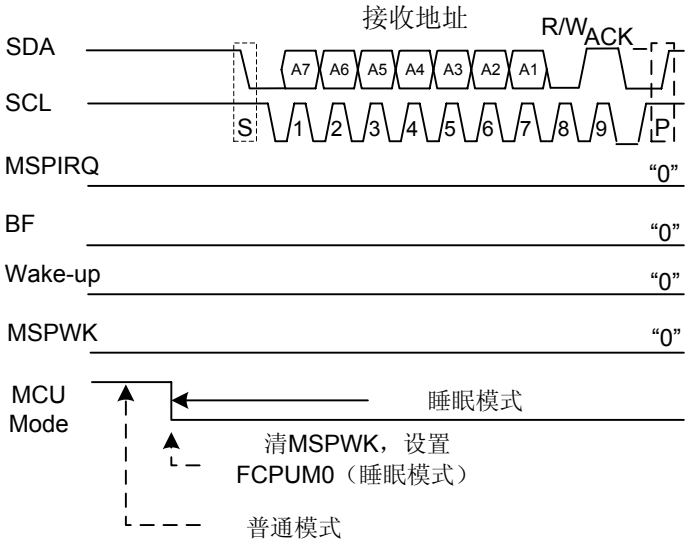


普通调用地址时序示意图

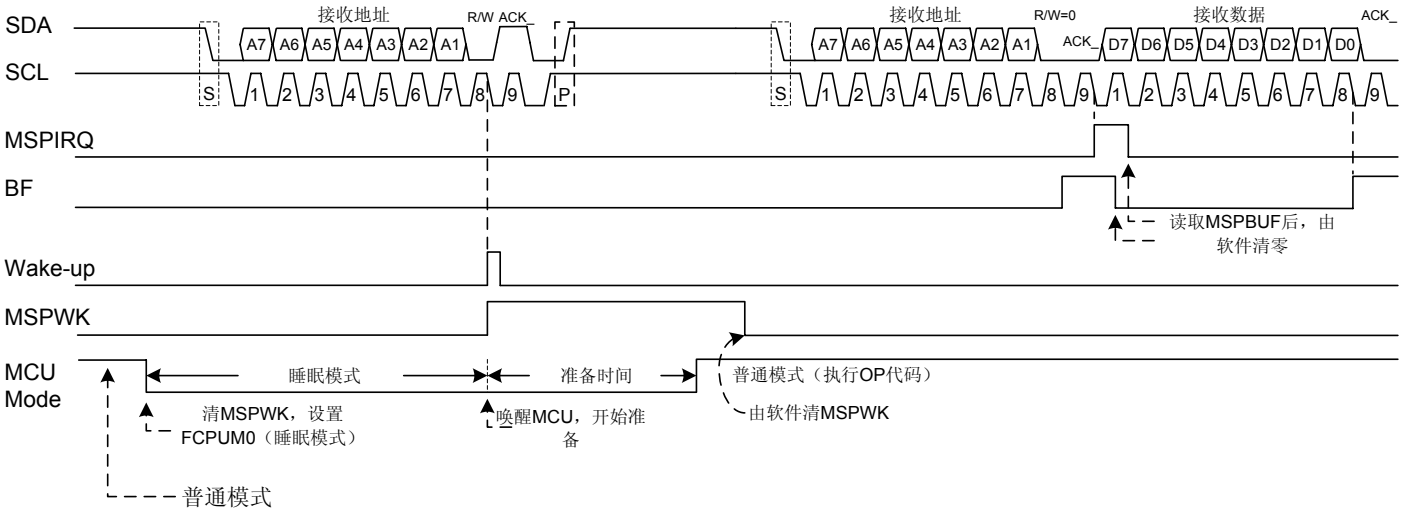
13.7.5 从动模式唤醒

睡眠模式下，若 MSPENB 置 1，则匹配的设备地址可以将单片机唤醒。

8 位 MSP 总线地址跟随 START 之后，装入 MSPSR，若地址匹配，在第 9 个 SCL 时钟时响应一个 NOT 应答信号，单片机被唤醒，MSPWK 置位，开始唤醒处理，但 MSPIRQ 不会置 1，且 MSPSR 数据也不会装入 MSPBUF。单片机被唤醒后，MSP 处于空闲状态，并等待主机的 START 信号。进入睡眠模式之前控制寄存器 BF、MSPIRQ、MSPOV 和 MSPBUF 的状态/数据相同。若地址不匹配，在第 9 个 SCL 时钟时发送一个 NOT 应答信号，但不会唤醒单片机，系统仍处于睡眠状态。



MSP 唤醒时序示意图：地址不匹配



MSP 唤醒时序示意图：地址匹配

- **注 1：MSP 功能只能在普通模式下工作，当系统从睡眠模式被唤醒后，单片机必须在主机发送 START 信号之前进入普通模式下进行工作。**
- **注 2：MSP 唤醒时，若地址不匹配，则单片机仍处于睡眠模式。**
- **注 3：在进入睡眠模式之前，先由软件清 MSPWK，以显示唤醒状态。**

13.8 主控模式操作

MSP 主控模式的操作从 START 信号开始，STOP 信号结束。
系统复位或者禁止 MSP 功能时，START（S）和 STOP（P）被清零。
主控模式下，SCL 和 SDA 线性由 MSP 硬件控制。

以下事件将触发中断请求（MSPIRQ），如果 MSPIEN 置 1，将触发中断：

- **START 情形。**
- **STOP 情形。**
- **发送/接收数据字节。**
- **发送应答信号。**
- **重复 START。**

13.8.1 主控模式支持格式

MSPC 和 MSPENB 置位时使能 MSP 主控模式，有 6 种情形：

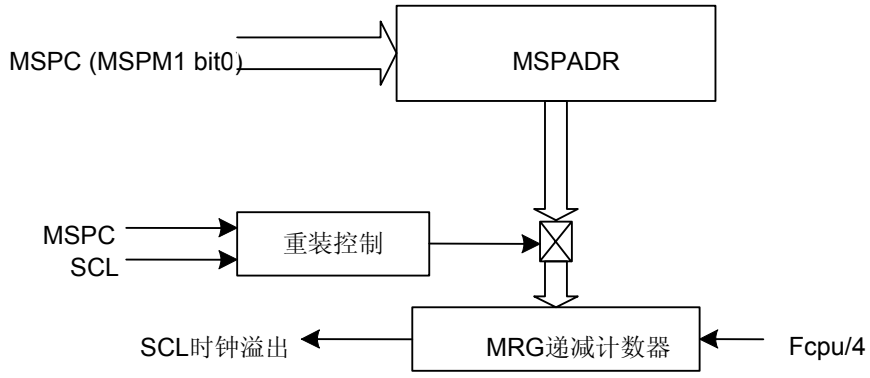
- **在 SCL 和 SDA 线上发送一个 START 信号。**
- **在 SCL 和 SDA 线上发送一个重复的 START 信号。**
- **将发送的数据和地址字节写入 MSPBUF 寄存器中。**
- **在 SCL 和 SDA 线上发送一个 STOP 信号。**
- **配置 MSP 端口以接收数据。**
- **在接收数据的末端发送一个应答信号。**

13.8.2 MSP 速率发生器（MRG）

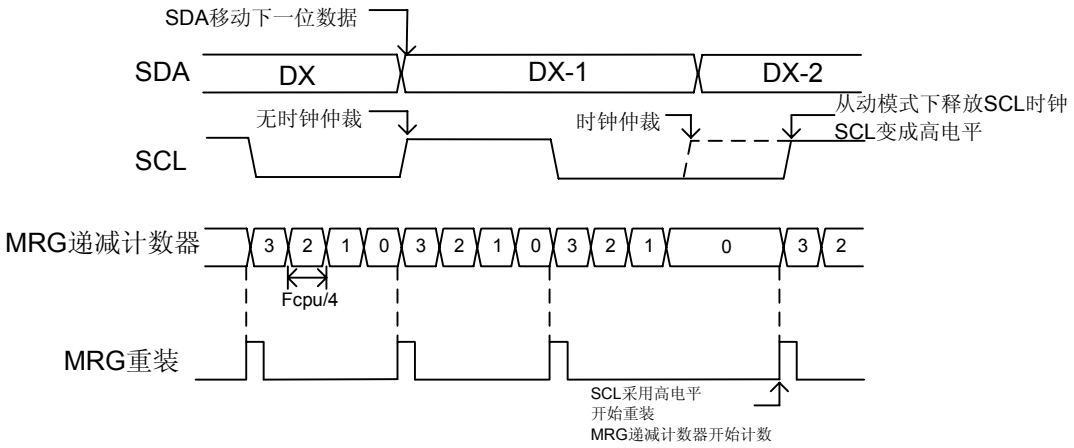
MSP 模式下，MSP 速率发生器的重装值位于 MSPADR 寄存器的低 7 位地址。当 MRG 装入寄存器，MRG 计数到 0，到另一个重装值时停止。在 MSP 主控模式下，MRG 自动重装 MSPADR 的值。若时钟仲裁发送（如 SCL 引脚由从动设备保持低电平），当 SCL 引脚检测到高电平时 MRG 重新装载。

SCL 时钟速率 = Fcpu/(MSPADR)*2

例如：设置 400KHz 4MHz Fcpu，MSPADR 必须设置为 05H
MSPADR=4Mhz/400Khz*2=5



MSP 速率发生器示意框图



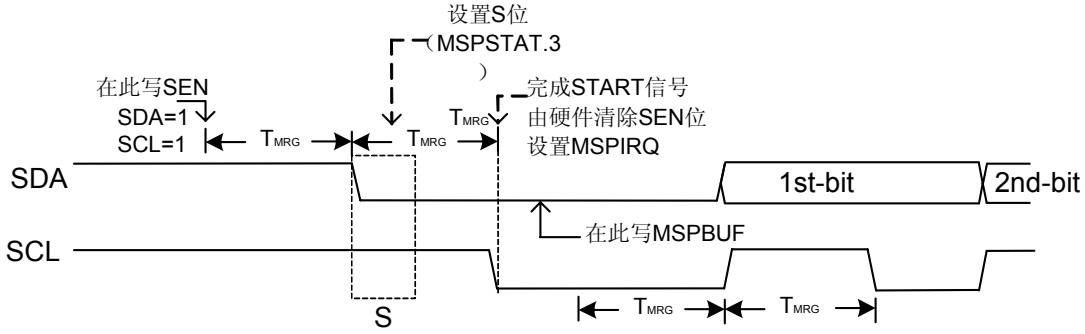
MRG 时序图，有无时钟仲裁（MSPADR=03H）

13.8.3 MSP 主控模式 START 操作

为产生一个 START 信号，用户可设置 SEN 位（MSPM2.0），当 SDA 和 SCL 引脚都采样高电平时，MSP 速率发生器重装 MSPADR[6:0]，然后开始往下计数。当 SDA 和 SCL 引脚都采样高电平且 MRG 溢出时，SDA 引脚驱动低电平。当 SCL 采样高电平，SDA 在 START 信号发送低电平并设置 S 位（MSPSTAT.3），MRG 再次重装并开始往下计数。MRG 溢出时 SEN 自动清零，MRG 挂起，SDA 保持低电平，START 完成。

WCOL 状态标志

若用户在处理 START 时写入数据到 MSPBUF，WCOL 置位，MSPBUF 的数据不作任何改变（写动作无效）。



START 操作时序示意图

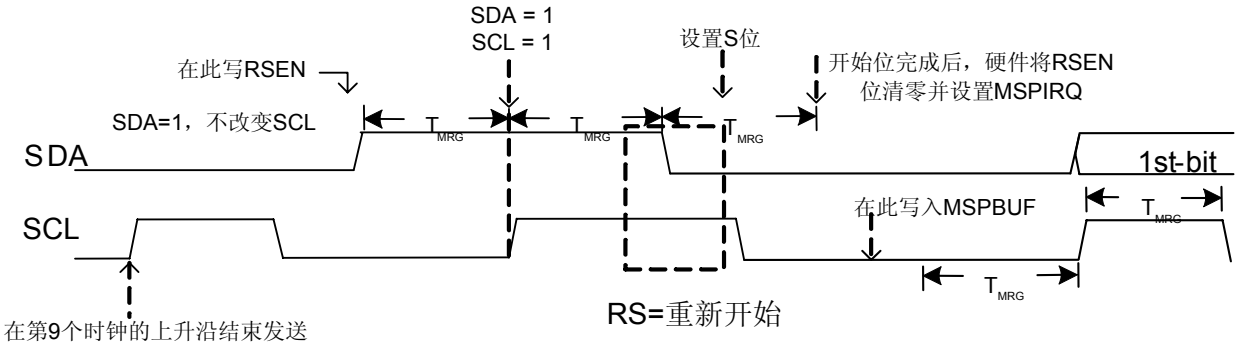
13.8.4 MSP 主控模式重复 START 操作

当 MSP 逻辑模块空闲且 RSEN 置 1 时，重复 START 操作。RSEN 置 1 且 SCL 引脚采样低电平时，MSPADR[6:0] 数据重装到 MSP 速率发生器中并开始往下计数。SDA 引脚在 MSP 速率发生计数器（T_{MRG}）释放高电平。当 MRG 溢出，且 SDA 采样高电平时，SCL 引脚输出高电平。当 SCL 采样高电平，MSPADR 重装到 MRG 并开始往下计数时，SDA 和 SCL 必须在 T_{MRG} 期间保持高电平。在下一个 T_{MRG} 期间，SCL 采用高电平时，SDA 输出低电平，RSEN 由硬件自动清零，MRG 不会重装，SDA 保持低电平。一旦检测到 SDA 和 SCL 发生 START 操作，S 位被置 1（MSPSTAT.3），MRG 溢出时，MSPIRQ 置 1。

✸ **注 1：在处理过程中发生任何其它的事件，设置 RSEN 以保证不造成影响。**
✸ **注 2：在重复 START 操作时会有总线冲突；SCL 开始输出高电平时，SDA 为低电平。**

WCOL 状态标志

若用户在重复 START 时写入数据到 MSPBUF，WCOL 置位，MSPBUF 的数据不作任何改变（写动作无效）。



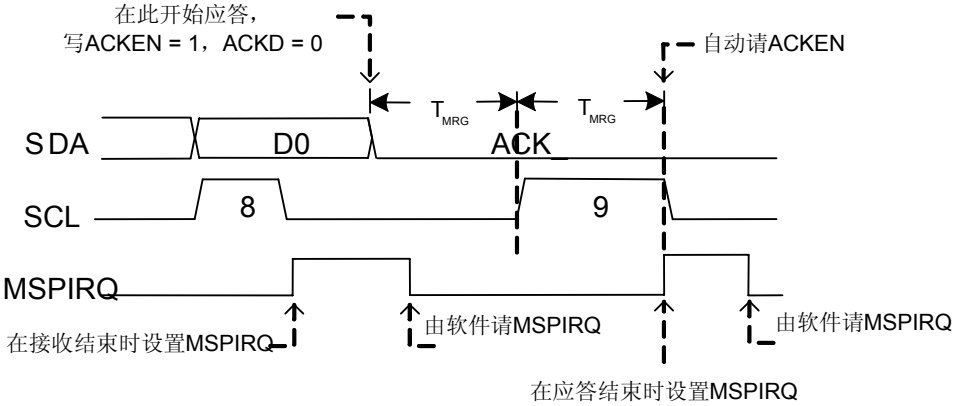
重复 START 操作时序示意图

13.8.5 应答流程时序

设置 ACKEN（MSPM2.4）时使能应答流程，SCL 输出低电平，应答数据位在 SDA 引脚为目前状态。若用户想要响应一个应答信号，需将 ACKDT 位清零，如若不然，在开始应答流程之前，置 ACKDT 位。MSP 速率发生器溢出时，释放 SCL 引脚（输出高电平）。SCL 采样高电平时，MSP 速率发生器开始 T_{MRG} 周期，往下计数。此段周期结束后，SCL 输出低电平，ACKEN 位被硬件自动清零。MRG 再次溢出时，MSP 进入空闲模式。

WCOL 状态标志

若用户在处理应答流程时写入数据到 MSPBUF，WCOL 置位，MSPBUF 的数据不作任何改变（写动作无效）。



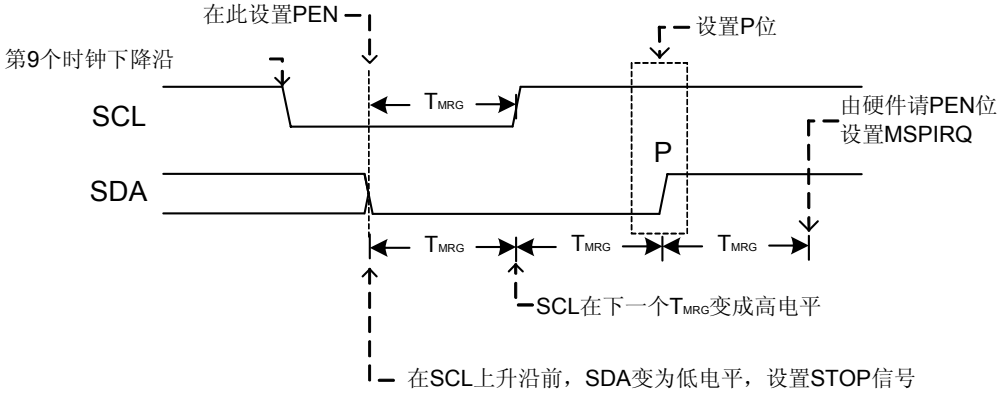
应答流程时序示意图

13.8.6 MSP 主控模式 STOP 操作流程

发送/接收结束时，通过设置 STOP 位 PEN（MSPM2.2），在 SDA 引脚出现 STOP 信号。发送/接收结束时，SCL 在第 9 个时钟的下降沿开始输出低电平。PEN 位置 1 时主机设置 SDA 输出低电平。SDA 采样低电平时，MSP 速率发生器重装并开始往下计数到 0。MRG 溢出时，SCL 引脚输出高电平。一个 T_{MRG} 周期结束后，SDA 输出高电平。SCL 输出高电平时，SDA 采用高电平，P 位置 1，下一个 T_{MRG} 周期结束后，PEN 位被清零，MSPIRQ 置 1。

WCOL 状态标志

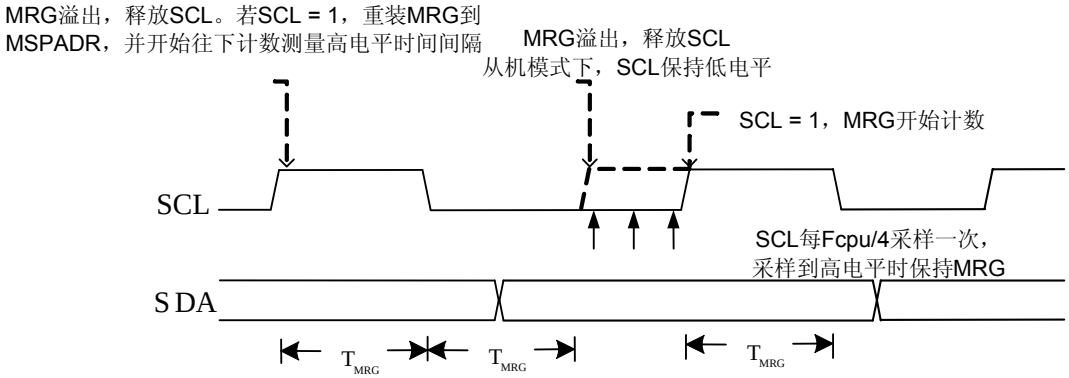
若用户在处理 STOP 时写入数据到 MSPBUF，WCOL 置位，MSPBUF 的数据不作任何改变（写动作无效）。



STOP 操作流程时序示意图

13.8.7 时钟仲裁

当主机接收/发送任何数据或者重复 START、STOP 信号时，产生时钟仲裁，此时 SCL 引脚处于高悬浮状态，MSP 速率发生器（MRG）一直挂起，从开始计数一直到 SCL 引脚采样到高电平。MRG 重装 MSPADR[6:0]，并开始往下计数，保证 SCL 高电平时间至少为一个 MRG 溢出时间，时钟由外部设备控制保持为低电平。



时钟仲裁流程时序示意图

13.8.8 主控模式发送数据

完全写入 MSPBUF 寄存器，表示一个数据字节、7 位地址或者 8 位数据发送完成，该操作设置缓存器 BF 标志，并允许 SMP 速率发生器开始计数。

写入 MSPBUF 后，地址的每一位都将在 SCL 的下降沿移出，直到 7 位地址和 R/W 位都完全移出。在第 8 个时钟的下降沿，主机将 SDA 拉低并给从机一个应答信号。在第 9 个时钟的下降沿，采样的 SDA 显示已经由从机接收到地址。ACK 的状态装入 ACKSTAT，MSPIRQ 置 1，BF 清零，MRG 持续直到另外的数据写入 MSPBUF，SCL 保持低电平，并允许 SDA 悬浮。

BF 状态标志

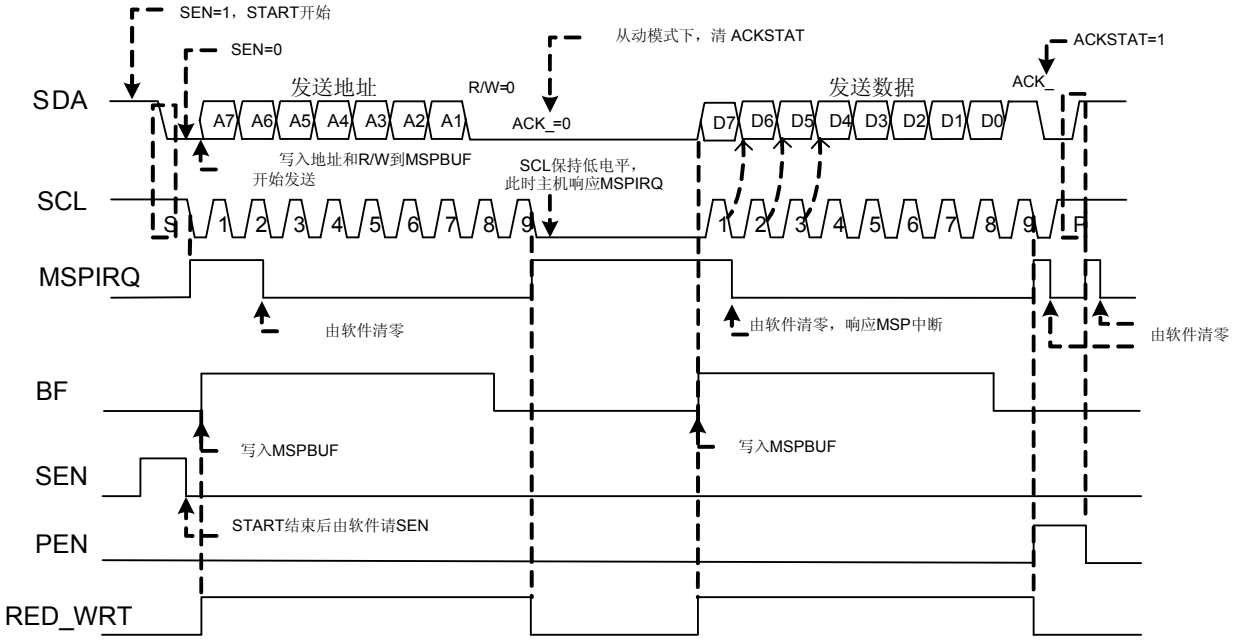
发送模式下，BF 位在数据写入 MSPBUF 时置 1，8 位数据全部移出时自动清零。

WCOL 标志

若用户在处理发送流程时写入数据到 MSPBUF，WCOL 置位，MSPBUF 的数据不作任何改变。

ACKSTAT 状态标志

发送模式下，从机发送应答信号后（ACK_=0）ACKSTAT 清零，没有发送应答信号（ACK_=1），ACKSTAT 则置 1。识别地址（包括普通调用地址）时，或者已经完全接收数据后，从机发送一个应答信号。



MSP 主机发送模式时序示意图

13.8.9 主控模式接收数据

主控接收模式由 RCEN 位设置。

当 SCL 从低变为高时，MRG 开始计数，数据存入 MSPSR。在第 8 个时钟的下降沿时，接收使能位 RECN 被自动清零，MSP 的内容被载入 MSPBUF，BF 位置 1，MSPIRQ 置 1，MRG 计数器挂起，SCL 保持低电平。MSP 处于空闲状态，并等待下一个操作命令。当由软件读取 MSPBUF 的数据后，BF 被自动清零。通过设置 ACKEN 位，用户可以在接收结束时发送应答信号。

BF 状态标志

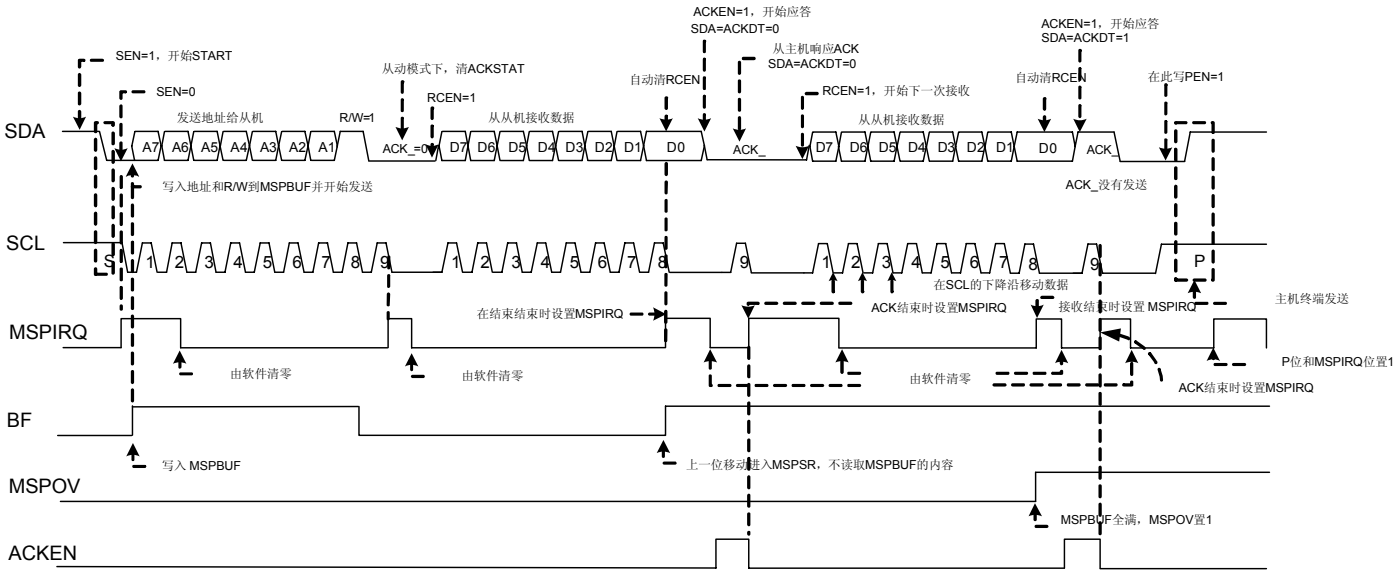
接收模式下，BF 在地址或者数据字节装入 MSPBUF 后被置 1，MSPBUF 的内容被读取后则自动清零。

MSPOV 标志

接收模式下，接收到另一个 8 位数据装入 MSPSR 后，MSPOV 置 1，BF 从前一次接收就一直置 1。

WCOL 标志

若用户在处理已经接收到的数据时写入数据到 MSPBUF，WCOL 置位，MSPBUF 的数据不作任何改变。



MSP 主机接收模式时序示意图

14 Flash

14.1 概述

SN8F2288 系列的 USB 单片机内置可编程的(ISP)的 FLASH 存储器以方便 CODE 升级。FLASH 存储器可经由 SONiX 8 位单片机编程界面进行编程。SN8F2280 提供安全的选项以保护用户代码的安全性。

- 单片机系统在进行写/擦除操作时会停止工作，虽然其他外围设备（USB、定时器、WDT、I/O、PWM 等）仍在工作。
- 在进行写/擦除操作时会由韧件禁止中断。
- 在使能编译选项的 Sucurity1 时，不能擦除 Flash 页的编译选项（CODE OPTION），ROM 地址为 2F80H~2FFFH。
- 在进行写/擦除操作之前清除看门狗定时器。
- 擦除操作会将 Flash 页中的所有位设置为逻辑 1。
- 硬件会保持系统时钟并自动将数据从 RAM 中导出，然后进行编程。编程完成后，硬件释放系统时钟并让系统执行下一条指令（建议在此增加 2 条 NOP 指令）。

14.2 FLASH 编程/擦除控制寄存器

0BAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PECMD	PECMD7	PECMD6	PECMD5	PECMD4	PECMD3	PECMD2	PECMD1	PECMD0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **PECMD[7:0]：5AH 页编程（32 字/页），C3H 页擦除（128 字/页）。**

14.3 编程/擦除地址寄存器

0BBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEROML	PEROML7	PEROML6	PEROML5	PEROML4	PEROML3	PEROML2	PEROML1	PEROML0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

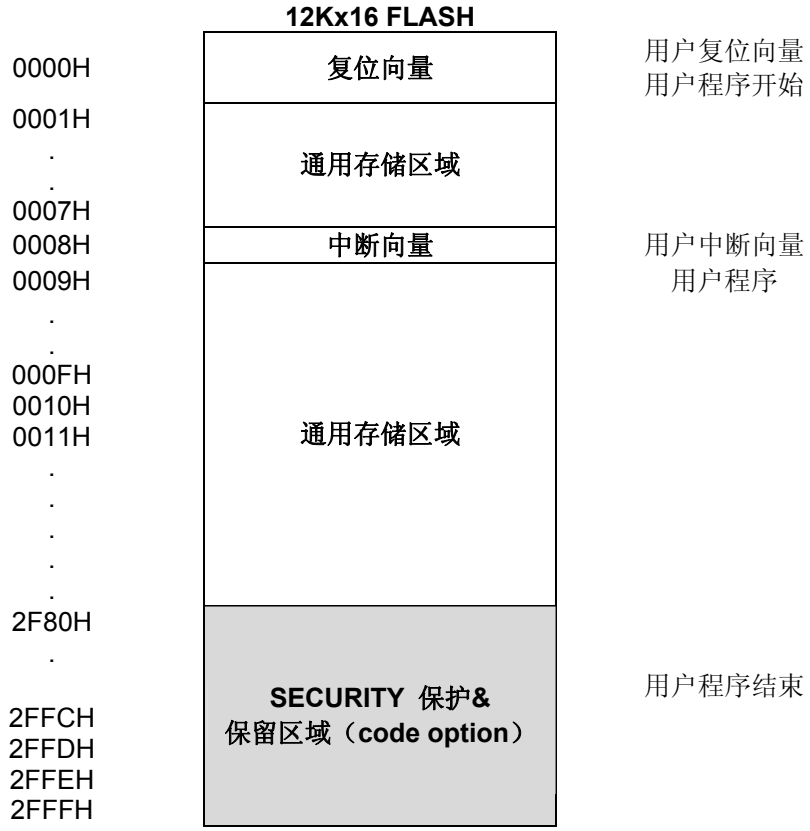
Bit [7:0] **PEROML[7:0]**：定义 Flash 存储器（12K*16）需要编程/擦除的低字节地址[7:0]。

0BCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEROMH	PEROMH7	PEROMH6	PEROMH5	PEROMH4	PEROMH3	PEROMH2	PEROMH1	PEROMH0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **PEROMH [7:0]**：定义 Flash 存储器（12K*16）需要编程/擦除的高字节地址[15:8]。

有效的**页擦除**的开始地址可以是 **00H、80H、100H、180H、200H、280H、300H、380H … 2F80H**。页擦除可以从 FLASH ROM 中擦除一页（128 个字）。

有效的页编程的开始地址可以时 00H、20H、40H、60H、80H、A0H、C0H、E0H … 2FE0H。页编程可以编写一页 32 个字到 FLASH ROM。



Flash ROM 分配

✸ **注：如编译选项的 SECURITY1=1（使能 SECURITY1）时，FLASH ROM 地址为 2F80H~2FFFH，不能执行页擦除和页写操作。**

14.4 编程/擦除数据寄存器

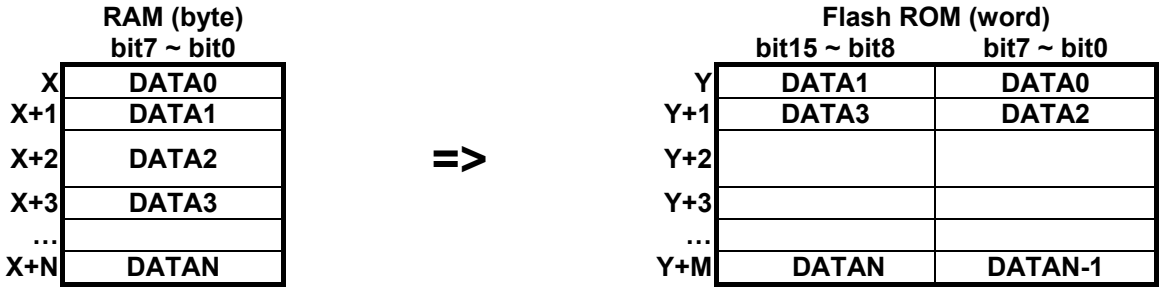
0BDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PERAML	PERAML7	PERAML6	PERAML5	PERAML4	PERAML3	PERAML2	PERAML1	PERAML0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0BEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PERAMCNT	PERAMCNT4	PERAMCNT3	PERAMCNT2	PERAMCNT1	PERAMCNT0	-	PERAML9	PERAML8
读/写	R/W	R/W	R/W	R/W	R/W	-	-	R/W
复位后	0	0	0	0	0	-	-	0

{PERAMCNT [1:0], PERAML [7:0]}：定义存储需要被编程的数据的 RAM 地址[9:0]，RAM 的有效地址为 00H~7FH 和 100H~27FH。

PERAMCNT [7:3]：定义需要编程字的长度，PERAMCNT[7:3]的最大值为 1FH，可以编程 32 个字（**64 字节 RAM**）到 Flash，PERAMCNT[7:3]的最小值为 00H，可以编程 1 个字到 Flash。

14.4.1 Flash 内置可编程功能分配地址



15 指令集

Field	Mnemonic		指令说明	C	DC	Z	Cycle
MOVE	MOV	A,M	A ← M	-	-	√	1
	MOV	M,A	M ← A	-	-	-	1
	B0MOV	A,M	A ← M (bank 0)	-	-	√	1
	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I，“M”指 80H~87H 的寄存器（如 PFLAG、R、Y、Z …）。	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1+N
	B0XCH	A,M	A ←→M (bank 0)	-	-	-	1+N
	MOVC		R, A ← ROM [Y,Z]	-	-	-	2
ARITHMETIC	ADC	A,M	A ← A + M + C，若有进位，则 C=1，否则 C=0。	√	√	√	1
	ADC	M,A	M ← A + M + C，若有进位，则 C=1，否则 C=0。	√	√	√	1+N
	ADD	A,M	A ← A + M，若有进位，则 C=1，否则 C=0。	√	√	√	1
	ADD	M,A	M ← A + M，若有进位，则 C=1，否则 C=0。	√	√	√	1+N
	B0ADD	M,A	M (bank 0) ← M (bank 0) + A，若有进位，则 C=1，否则 C=0。	√	√	√	1+N
	ADD	A,I	A ← A + I，若有进位，则 C=1，否则 C=0。	√	√	√	1
	SBC	A,M	A ← A - M - /C，若有借位，则 C=0，否则 C=1。	√	√	√	1
	SBC	M,A	M ← A - M - /C，若有借位，则 C=0，否则 C=1。	√	√	√	1+N
	SUB	A,M	A ← A - M，若有借位，则 C=0，否则 C=1。	√	√	√	1
	SUB	M,A	M ← A - M，若有借位，则 C=0，否则 C=1。	√	√	√	1+N
	SUB	A,I	A ← A - I，若有借位，则 C=0，否则 C=1。	√	√	√	1
LOGIC	AND	A,M	A ← A 与 M。	-	-	√	1
	AND	M,A	M ← A 与 M。	-	-	√	1+N
	AND	A,I	A ← A 与 I。	-	-	√	1
	OR	A,M	A ← A 或 M。	-	-	√	1
	OR	M,A	M ← A 或 M。	-	-	√	1+N
	OR	A,I	A ← A 或 I。	-	-	√	1
	XOR	A,M	A ← A 异或 M。	-	-	√	1
	XOR	M,A	M ← A 异或 M。	-	-	√	1+N
	XOR	A,I	A ← A 异或 I。	-	-	√	1
PROCESS	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1+N
	RRC	M	A ← RRC M	√	-	-	1
	RRCM	M	M ← RRC M	√	-	-	1+N
	RLC	M	A ← RLC M	√	-	-	1
	RLCM	M	M ← RLC M	√	-	-	1+N
	CLR	M	M ← 0	-	-	-	1
	BCLR	M.b	M.b ← 0	-	-	-	1+N
	BSET	M.b	M.b ← 1	-	-	-	1+N
	B0BCLR	M.b	M(bank 0).b ← 0	-	-	-	1+N
	B0BSET	M.b	M(bank 0).b ← 1	-	-	-	1+N
BRANCH	CMPRS	A,I	ZF,C ← A – I，若 A=1，则跳到下一条指令。	√	-	√	1 + S
	CMPRS	A,M	ZF,C ← A – M 若 A=M，则跳到下一条指令。	√	-	√	1 + S
	INCS	M	A ← M + 1，若 A=0，则跳到下一条指令。	-	-	-	1+ S
	INCMS	M	M ← M + 1，若 M = 0，则跳到下一条指令。	-	-	-	1+N+S
	DECS	M	A ← M – 1，若 A = 0，则跳到下一条指令。	-	-	-	1+ S
	DECMS	M	M ← M – 1，若 M = 0，则跳到下一条指令。	-	-	-	1+N+S
	BTS0	M.b	If M.b = 0，则跳到下一条指令。	-	-	-	1 + S
	BTS1	M.b	If M.b = 1，则跳到下一条指令。	-	-	-	1 + S
	B0BTS0	M.b	If M(bank 0).b = 0，则跳到下一条指令。	-	-	-	1 + S
	B0BTS1	M.b	If M(bank 0).b = 1，则跳到下一条指令。	-	-	-	1 + S
	JMP	d	PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALL	d	Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
MISC	RET		PC ← Stack	-	-	-	2
	RETI		PC ← Stack，使能全局中断。	-	-	-	2
	PUSH		保存 ACC 和 PFLAG。	-	-	-	1
	POP		恢复 ACC 和 PFLAG。	√	√	√	1
	NOP		空操作。	-	-	-	1

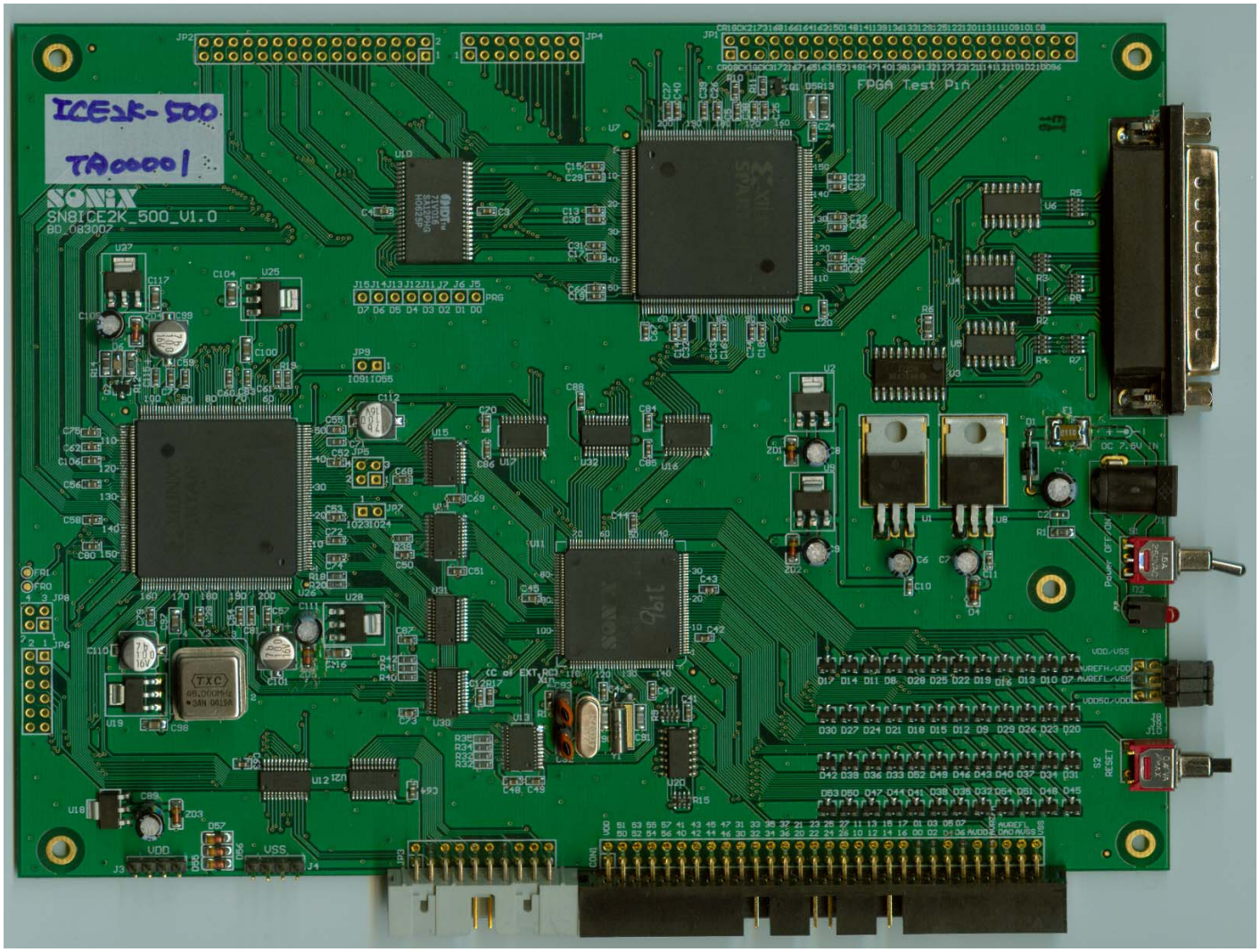
Note: ***1. M 指系统寄存器或 RAM，若 M 指系统寄存器，则 N=0，否则 N=1。***
2. 若条件为真则 S=1，否则 S=0。

16 开发工具

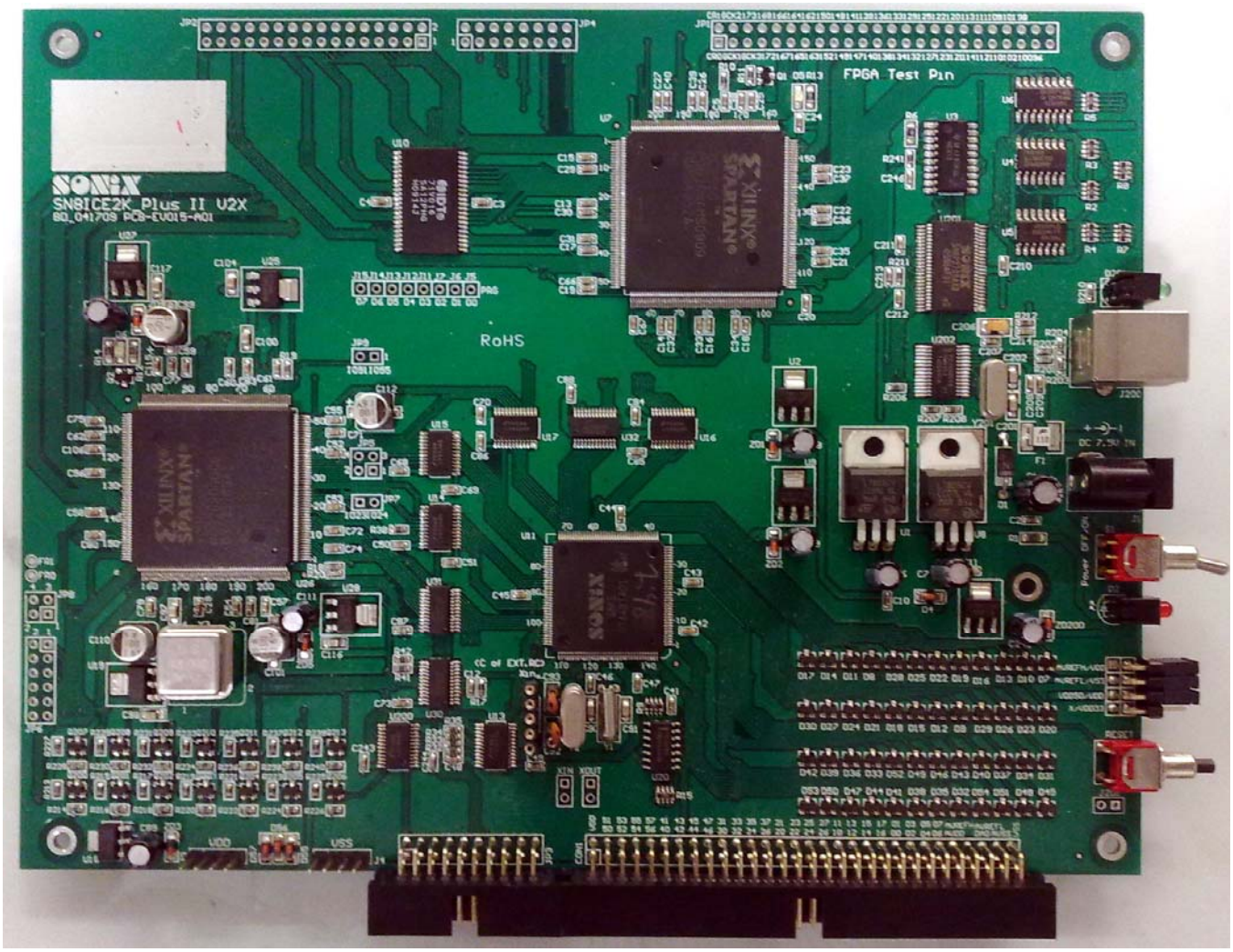
SONIX 为 USB 开发应用提供了在线仿真器 ICE，集成开发环境 IDE，EV-Kit 和韧件库，ICE 和 EV-Kit 为外部硬件，ICE 则为用户进行韧件开发和仿真提供友好的界面。

16.1 在线仿真器 ICE

下图中的 ICE 称为“SN8ICE2K Plus”。



下图中的 ICE 称为“SN8ICE2K Plus II”。



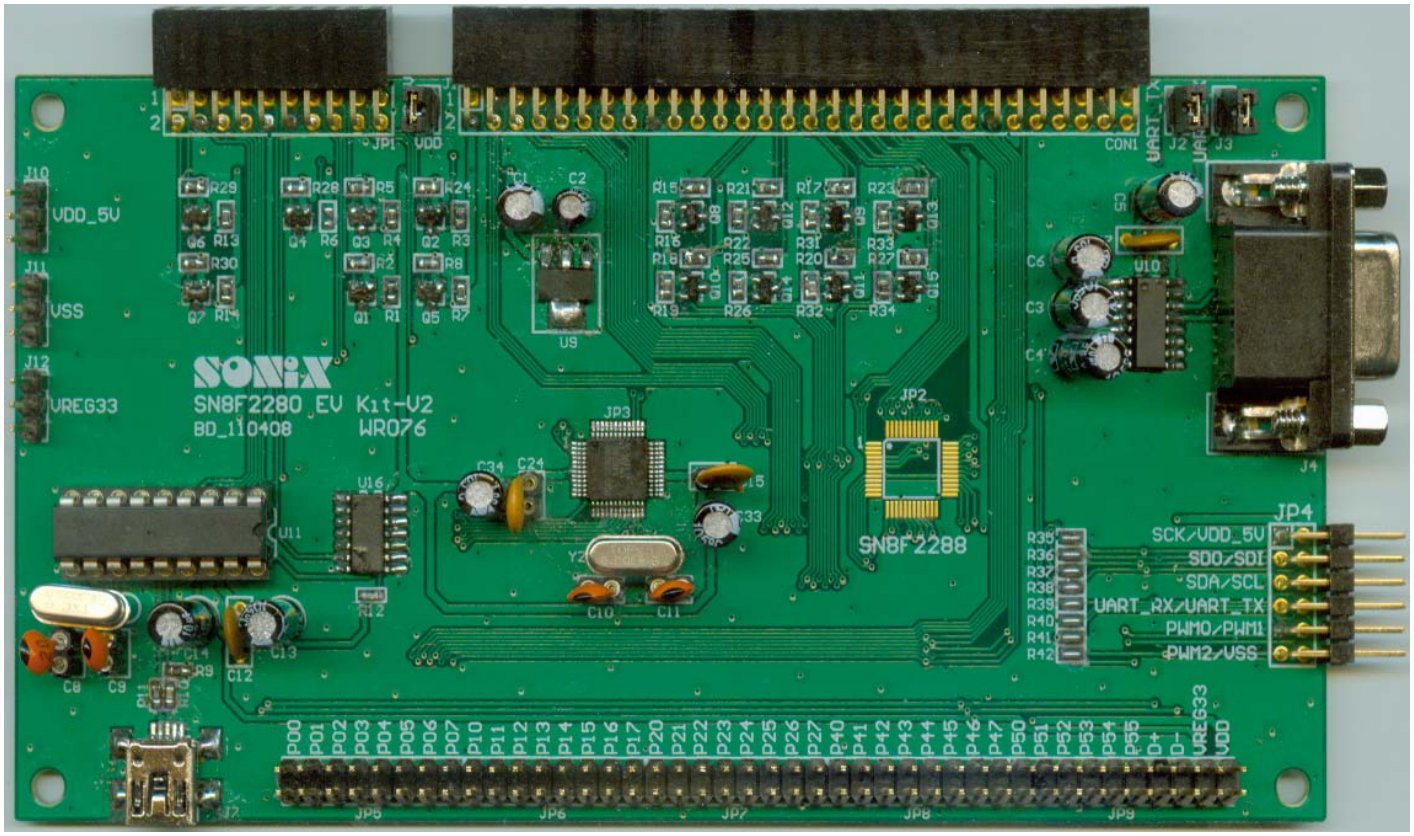
✸ **注 1：“SN8ICE2K Plus”须和 SN8F2280 EV-Kit V2 配合进行开发和仿真。**
✸ **注 2：“SN8ICE2K Plus II“须和 SN8F2280 EV-Kit V3 配合进行开发和仿真。**

16.2 SN8F2280 EV-kit

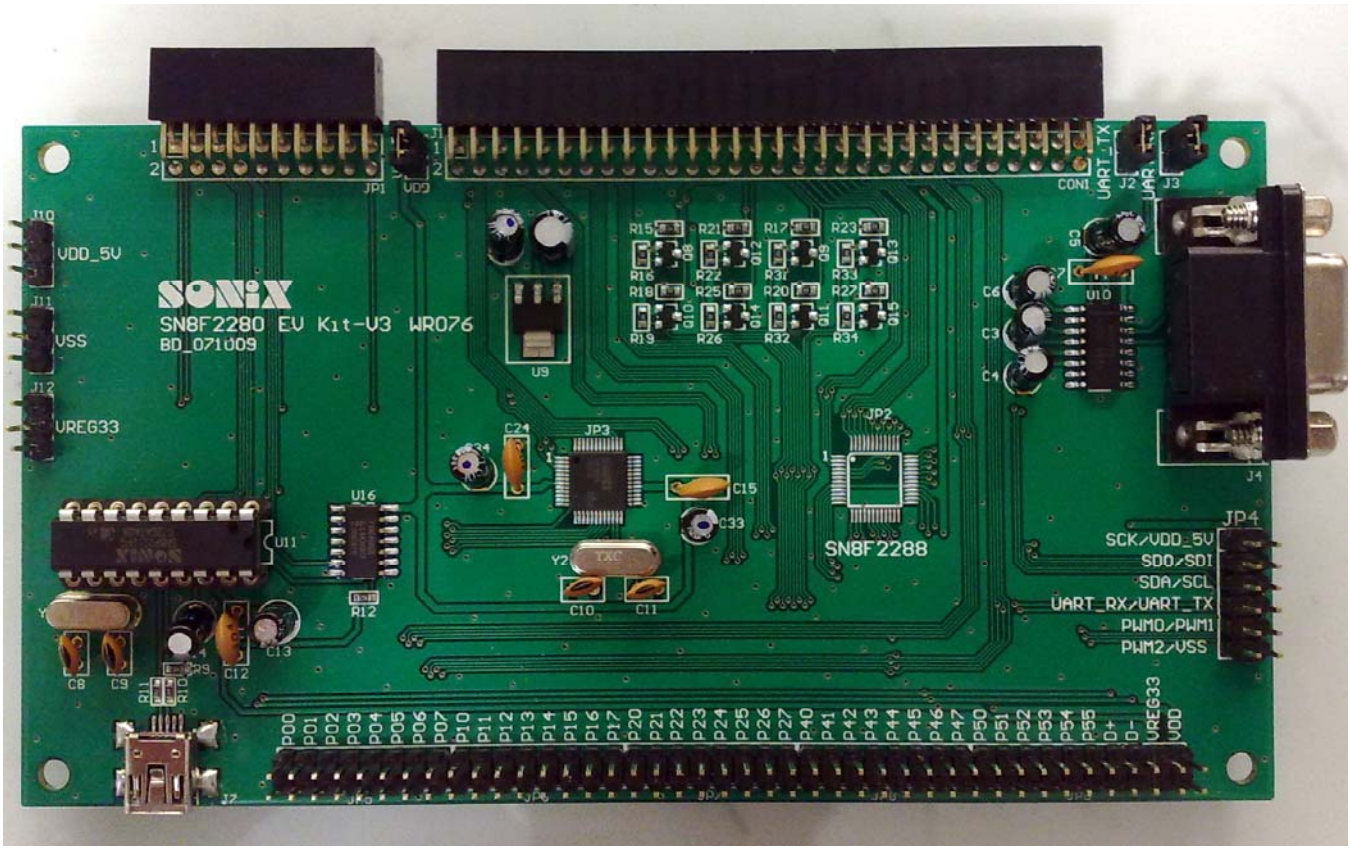
SN8F2280 EV-kit 包括 ICE 接口，GPIO 接口，USB 接口、UART 接口、SIO 接口、MSP 接口、PWM 接口和 VERG3.3V 电源输入。

- ICE 接口：连接 SN8ICE2K Plus。
- GPIO 接口：连接封装片 SN8F2288。
- USB 接口：USB Mini-B 接口。
- UART 接口：连接通用异步收发器（UART）。
- SIO 接口：连接串行输入输出收发器（SIO）。
- MSP 接口：连接 MSP。
- PWM 接口：输出 PWM 信号。
- VREG 3.3V 电源输入：利用 SN8P2212 VREG 提供 3.3V 电源给 VREG 引脚。

SN8F2280 EV-kit V2 如下图所示：



SN8F2280 EV-kit V3 如下图所示：

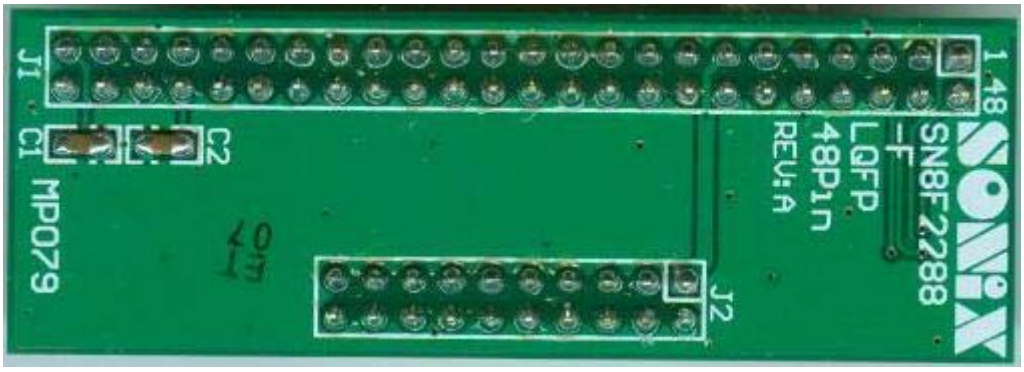


- CON1：ICE 接口，连接 SN8ICE2K Plus。
- J1：连接 SN8ICE2K Plus 上的 5V VDD 和 SN8F2288 封装片上的 VDD 跳线座。
- J7：USB Mini-B 接口。
- U11：SN8P2212 提供 3.3V 电源给 VREG 和 USB PHY。
- JP2：SN8F2288 连接用户目标板。
- JP3：SN8F2288 提供 P0 和 P1 I/O 电平变换中断功能。
- J2~J4：UART 接口，连接 RS-232。
- JP4：SIO 接口，MSP 接口，UART 接口，PWM 接口。

✸ 注 1：EV-Kit 上的 UART 和 MSP 没有内置漏极开路功能，如 P0.5/URX、P0.6/UTX、P1.0/SCL、P1.1/SDA，与实际芯片不同。
✸ 注 2：EV-Kit 上的 P2、P0.5/URX、P0.6/UTX、P1.0/SCL、P1.1/SDA 和 P5.5/PWM2 都内置 1K 外部上拉电阻。
✸ 注 3：EV-Kit 上的 P2 无施密特触发，与实际芯片不同。
✸ 注 4：“SN8ICE2K Plus”须和 SN8F2280 EV-Kit V2 配合进行开发和仿真。
✸ 注 5：“SN8ICE2K Plus II”须和 SN8F2280 EV-Kit V3 配合进行开发和仿真。

16.3 SN8F2280 转接板

SN8F2280 转接板用来烧录 SN8F2288 LQFP 封装片，如下图所示。下图中的 C1 和 C2 都必须焊接 1uF 的电容。



17 电气特性

17.1 极限参数

Supply voltage (Vdd)..... - 0.3V ~ 5.5V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8F2288F ..... 0°C ~ + 70°C
Storage ambient temperature (Tstor) ..... –30°C ~ + 125°C

17.2 电气特性

(All of voltages refer to Vss, Vdd = 5.0V, fosc = 12MHz, ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd1	Normal mode except USB transmitter		4.0	5	5.5	V
	Vdd2	USB mode		4.25	5	5.25	V
RAM Data Retention voltage	Vdr			-	1.5*	-	V
Vdd rise rate	Vpor	Vdd rise rate to ensure power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL1	P0, P1, P4, P5 input ports		Vss	-	0.2Vdd	V
	ViL2	P2 input ports		Vss	-	0.2 VREG33	V
Input High Voltage	ViH1	P0, P1, P4, P5 input ports		0.8Vdd	-	Vdd	V
	ViH2	P2 input ports		0.8 VREG33	-	VREG33	V
Input Voltage	Vin1	P0, P1, P4, P5 I/O port's input voltage range		-0.5	-	Vdd+0.5	V
	Vin2	P2 I/O port's input voltage range		-0.3	-	VREG33 +0.3	V
Output Voltage	Voh1	P0, P1, P4, P5 output ports		0	-	Vdd	V
	Voh2	P2 output ports		0	-	VREG33	V
Reset pin leakage current	Ilekg	Vin = Vdd		-	-	2	uA
I/O port pull-up resistor Rup1	Rup1	P0, P1, P4, P5 's Vin = Vss, Vdd = 5V		25	40*	70	KΩ
I/O port pull-up resistor Rup2	Rup2	P2's Vin = Vss, Vdd = 5V		30	55*	100	KΩ
D+ pull-up resistor	Rd+	Vdd = 5V, VREG = 3.3V		1	1.5	1.65	KΩ
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O output source current	IoH1	P0, P1, P4, P5 I/O port's output source current, Vop1 = Vdd – 1V		15	20*		mA
	IoH2	P2 I/O port's output source current, Vop2 = VREG33 – 1V		1	2*		
sink current	IoL1	P0, P1, P4, P5 I/O port's sink current, Vop1 = Vss + 0.4V			15*	20	
	IoL2	P2 I/O port's sink current, Vop2 = Vss + 0.4V			2*	3	
INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Page erase (128 words)	Terase	Flash ROM page erase time		-	25*	50	ms
Page program (32 words)	Tpg	Flash ROM page program time (program 32 words)		-	1*	2	ms
VREG33 Regulator current	IVREG33	VREG33 Max Regulator Output Current, Vcc > 4.35 volt with 10uF to GND		-	-	60	mA
VREG33 Regulator GND current	Ivreg33 _gnl	No loading. VREG33 pin output 3.3V ((Regulator enable)		-	70	100	uA
VREG25 Regulator GND current	Ivreg25 _gnl	No loading.VREG25 pin output 2.5V ((Regulator enable)		-	120	150	uA
VREG33 Regulator Output voltage	Vreg1	VCC > 4.35V, 0 < temp < 40°C, IVREG ≦ 60 mA with 10uF to GND		3.0	-	3.6	V
	Vreg2	VCC > 4.35V, 0 < temp < 40°C, IVREG ≦ 25 mA with 10uF to GND		3.1	-	3.6	V
Supply Current	Idd1	normal Mode (No loading, Fcpu = Fosc/1)	Vdd= 5V, 12Mhz	-	10	15	mA
	Idd2	Slow Mode (Internal low RC)	Vdd= 5V, 12Khz	-	190	250	uA
	Idd3	Sleep Mode	Vdd= 5V	-	190	250	uA
	Idd4	Green Mode (No loading, Fcpu = Fosc/4 Watchdog Disable)	Vdd= 5V, 12Mhz	-	5	10	mA
			Vdd=5V, ILRC 12Khz	-	190	250	uA
LVD Voltage	Vdet1	Low voltage reset level middle.		1.8	2.4	2.9	V
	Vdet2	Low voltage reset level high.		2.8	3.6	4.25	V

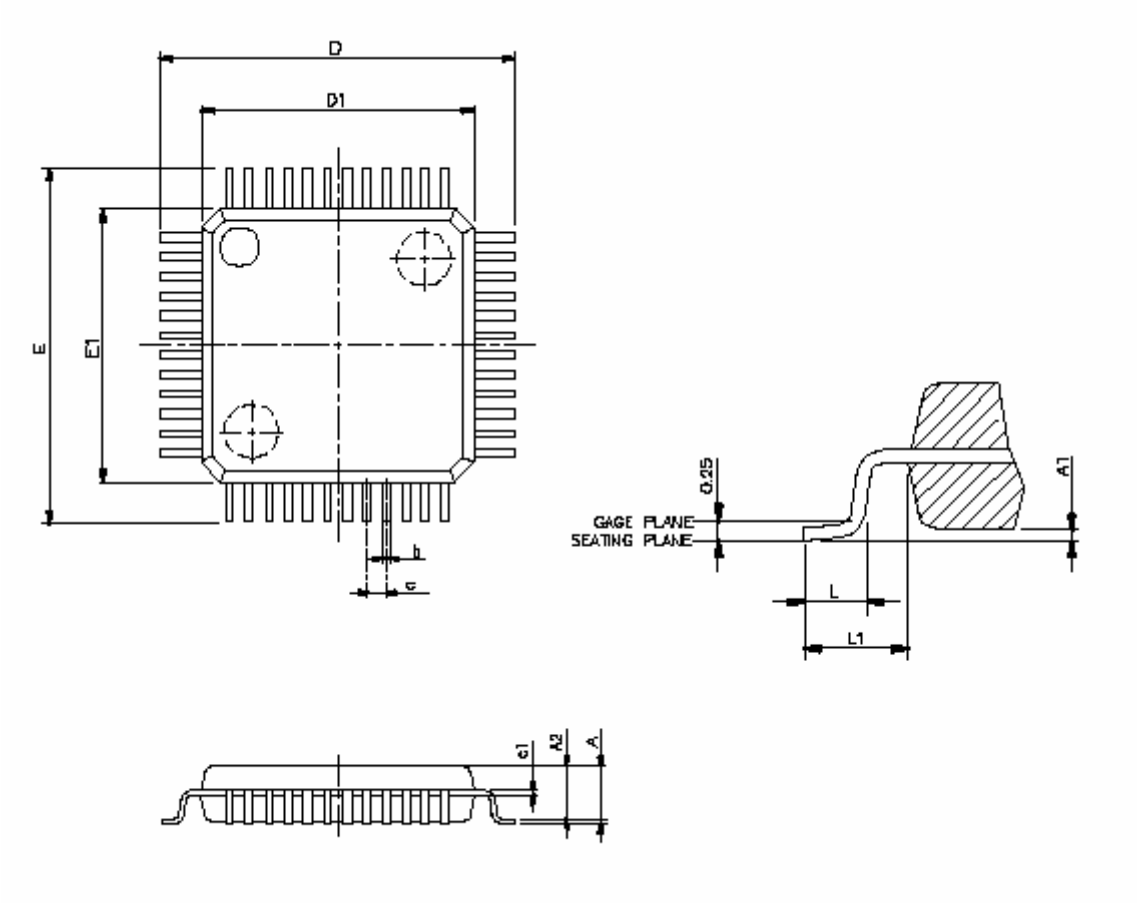
*** These parameters are for design reference, not tested.**

18 FLASH ROM 烧录引脚

SN8F2288 烧录信息					
单片机名称		SN8F2288F/J		SN8F2283J	
MPIII Writer		Flash IC / JP3 引脚配置			
引脚编号	引脚名称	引脚编号	引脚名称	引脚编号	引脚名称
1	VDD	7,29	VDD	11,23	VDD
2	GND	16,25	VSS	2,12,13,19	VSS
3	CLK	47	P1.4	5	P1.4
4	CE				
5	PGM	1	P1.2	7	P1.2
6	OE	46	P1.5	4	P1.5
7	D1				
8	D0				
9	D3				
10	D2				
11	D5				
12	D4				
13	D7				
14	D6				
15	VDD				
16	-				
17	HLS				
18	RST				
19	-				
20	ALSB/PDB	48	P1.3	6	P1.3

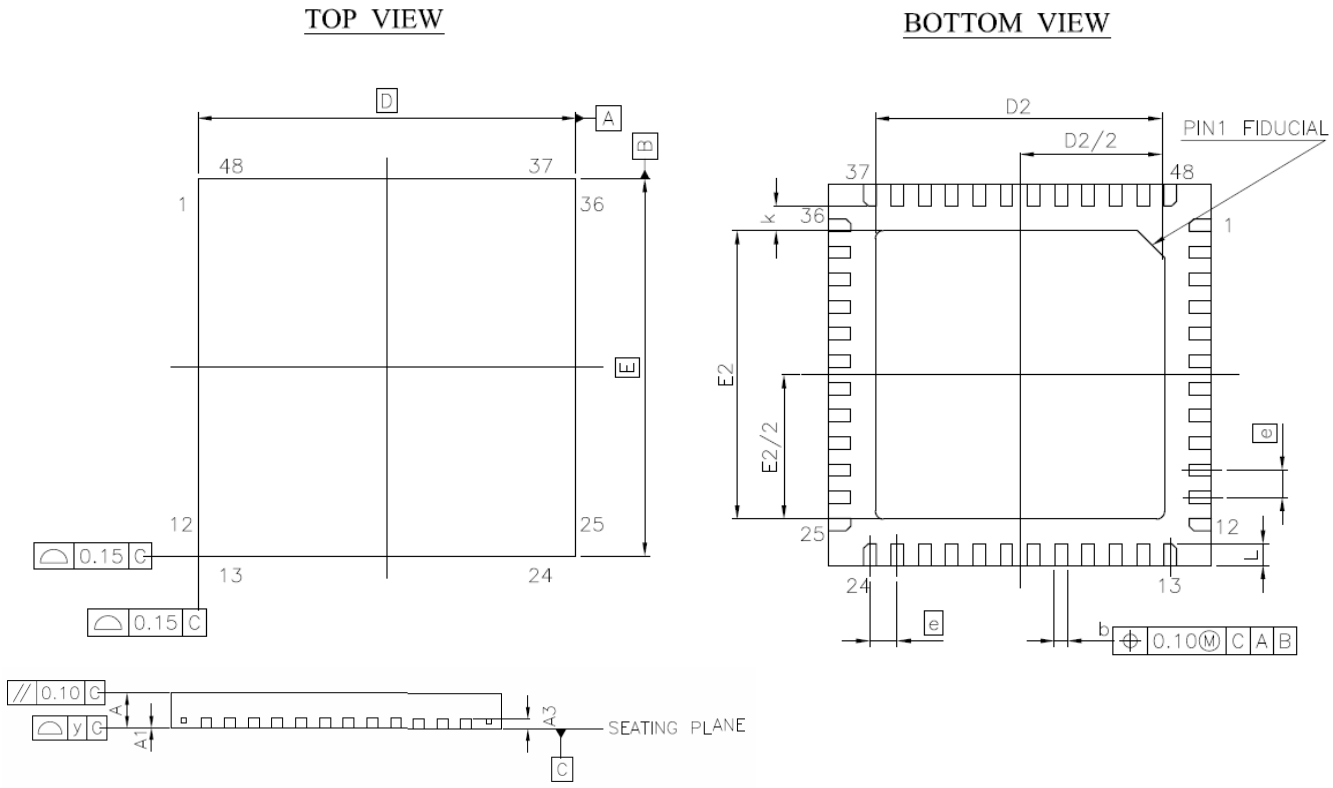
19 封装信息

19.1 LQFP 48 PIN



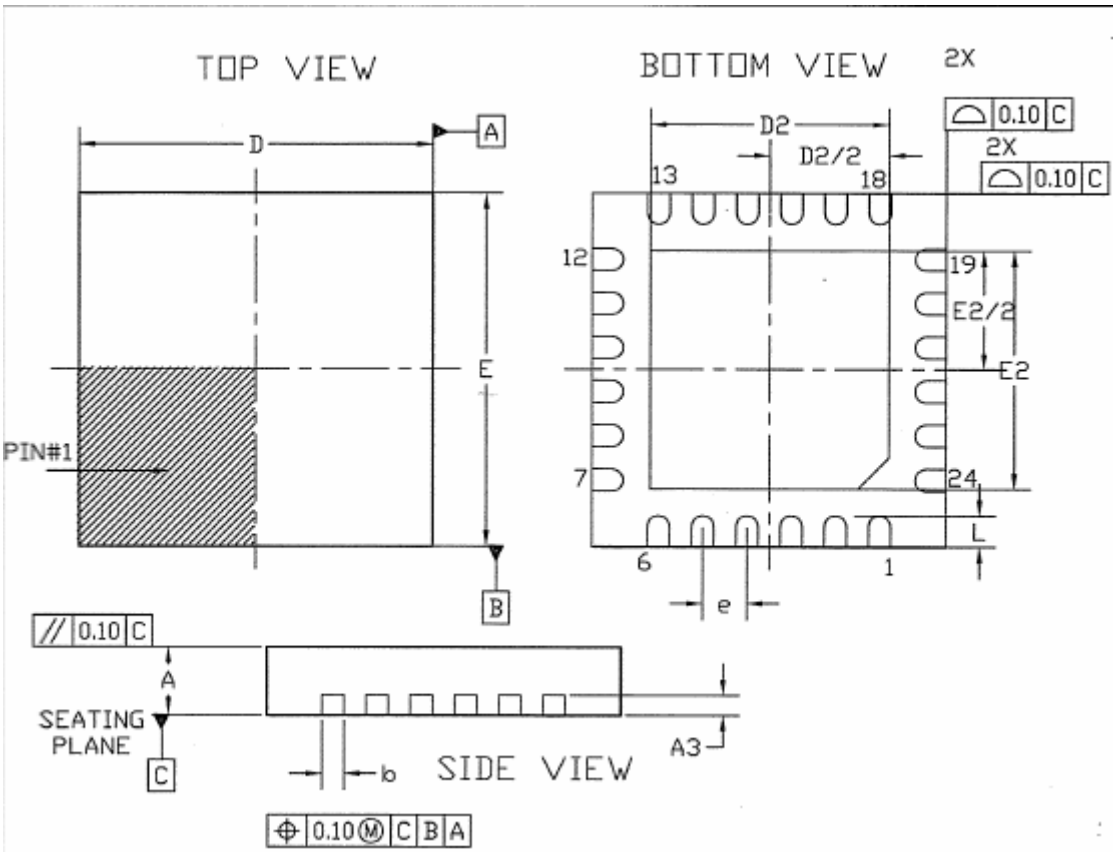
SYMBOLS	MIN	NOR	MAX
	(mm)		
A	-	-	1.6
A1	0.05	-	0.15
A2	1.35	-	1.45
c1	0.09	-	0.16
D	9.00 BSC		
D1	7.00 BSC		
E	9.00 BSC		
E1	7.00 BSC		
e	0.5 BSC		
B	0.17	-	0.27
L	0.45	-	0.75
L1	1 REF		

19.2 QFN 48 PIN



SYMBOL	DIMENSION (MM)			DIMENSION (MIL)		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	0.80	0.85	0.90	31.5	33.5	35.4
A1	0	0.02	0.05	0	0.79	1.97
A3	0.203 REF			8 REF		
b	0.18	0.25	0.30	7.1	9.8	11.8
D	7.00 BSC			276 BSC		
D2	5.10	5.20	5.60	201	205	221
E	7.00 BSC			276 BSC		
E2	5.10	5.20	5.30	201	205	209
e	0.50 BSC			19.7 BSC		
K	0.2			7.9		
L	0.30	0.40	0.50	11.8	15.7	19.7
y	0.08			3.15		

19.3 QFN 24 PIN



COMMON	DIMENSIONS MILLIMETER			DIMENSIONS INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	SEE VARIATION					
A3	0.195	0.203	0.211	0.0077	0.008	0.0083
b	0.180	0.230	0.300	0.007	0.009	0.012
D	3.925	4.0	4.075	0.154	0.157	0.160
E	3.925	4.0	4.075	0.154	0.157	0.160
e	0.50 BSC			0.020 BSC		
L	SEE VARIATION					

VARIATION ' A '	DIMENSIONS MILLIMETER			DIMENSIONS INCH			REF
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.	
TQFN	0.70	0.75	0.80	0.027	0.029	0.031	W VERY VERY THIN
QFN	0.85	0.90	0.95	0.033	0.035	0.037	V VERY THIN

PAD SIZE	VARIATION ' L '						REF
	DIMENSIONS MILLIMETER			DIMENSIONS INCH			
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.	
118x118	0.30	0.35	0.40	0.012	0.014	0.016	CUSTOMS A,B

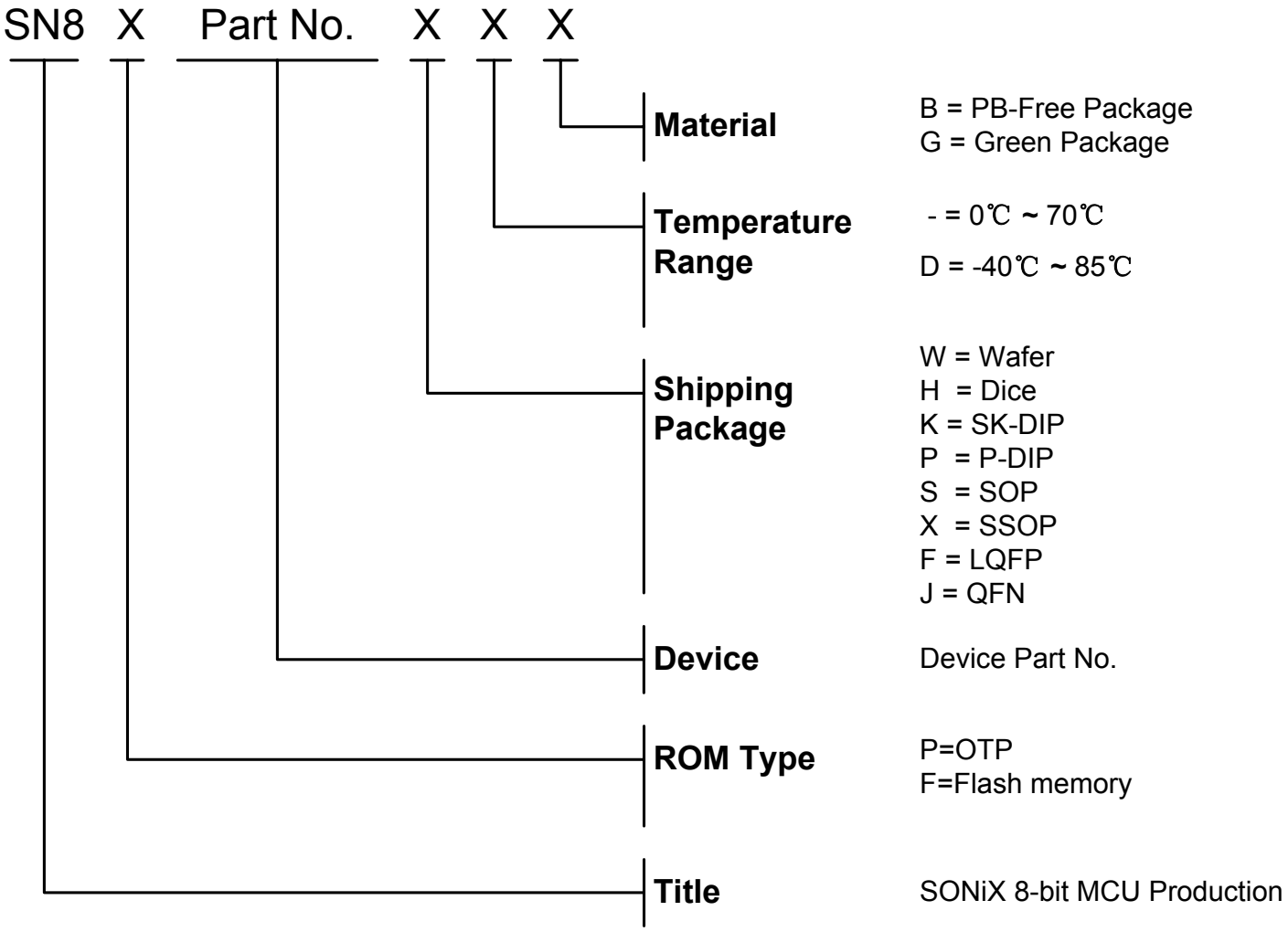
PAD SIZE	D2/E2 DIMENSIONS MILLIMETER			D2/E2 DIMENSIONS INCH			REF
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.	
118x118	2.50/2.50	2.65/2.65	2.80/2.80	0.098/0.098	0.104/0.104	0.110/0.110	VGGD-6, WGGD-6

20 芯片正印命名规则

20.1 概述

SONiX 8 位单片机产品具有多种型号，本章将给出所有 8 位单片机分类命名规则。

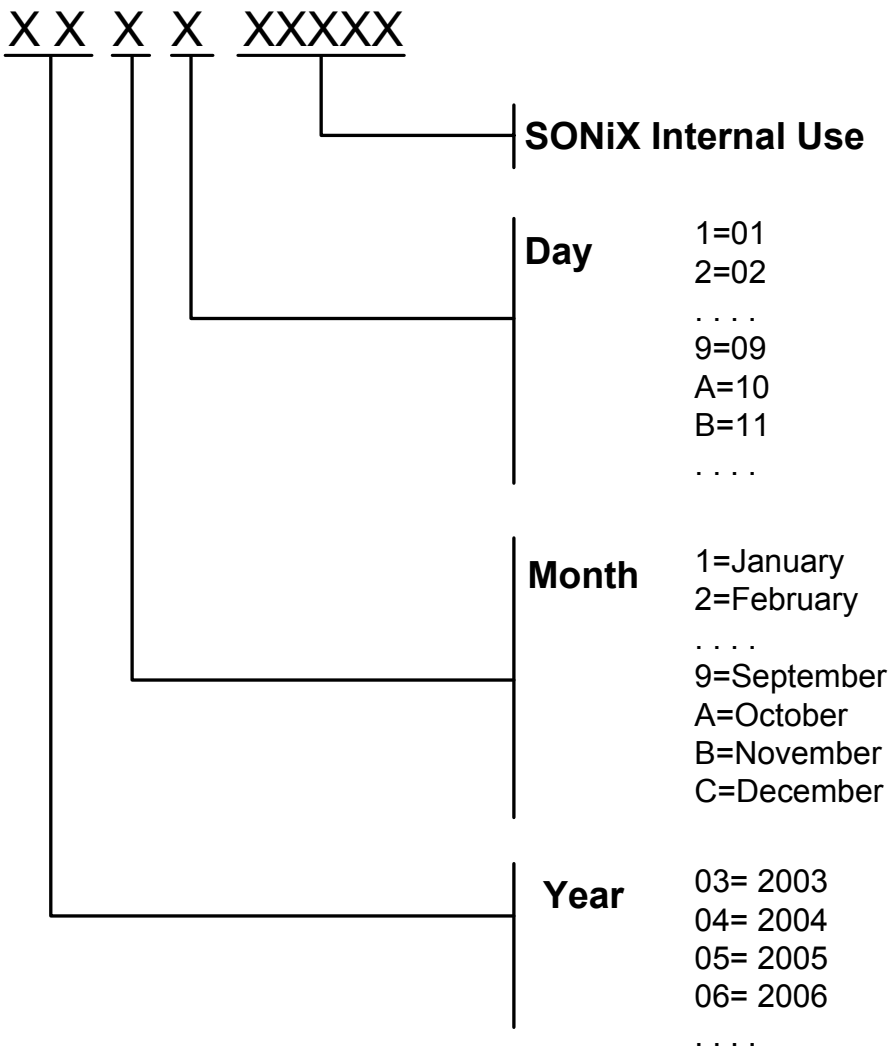
20.2 芯片型号说明



20.3 命名举例

单片机名称	ROM 类型	器件名称（Device）	封装形式	温度范围	封装材料
SN8F2288FG	Flash memory	2288	LQFP	0℃~70℃	绿色封装
SN8F2288W	Flash memory	2288	Wafer	0℃~70℃	-
SN8F2288H	Flash memory	2288	Dice	0℃~70℃	-
SN8F2283JG	Flash memory	2288	QFN	0℃~70℃	绿色封装

20.4 日期码规则



SONiX 公司保留对以下所有产品在可靠性，功能和设计方面的改进作进一步说明的权利。SONiX 不承担由本手册所涉及的产品或电路的运用和使用所引起的任何责任，SONiX 的产品不是专门设计来应用于外科植入、生命维持和任何 SONiX 产品的故障会对个体造成伤害甚至死亡的领域。如果将 SONiX 的产品应用于上述领域，即使这些是由 SONiX 在产品设计和制造上的疏忽引起的，用户应赔偿所有费用、损失、合理的人身伤害或死亡所直接或间接产生的律师费用，并且用户保证 SONiX 及其雇员、子公司、分支机构和销售商与上述事宜无关。

总公司：
地址：台湾新竹县竹北市台元街 36 号 10 楼之一
电话：886-3-5600-888
传真：886-3-5600-889
台北办事处：
地址：台北市松德路 171 号 15 楼之 2
电话：886-2-2759 1980
传真：886-2-2759 8180
香港办事处：
地址：香港新界沙田沙田乡宁会路 138＃新城市中央广场第一座 7 楼 705 室
电话：852-2723 8086
传真：852-2723 9179
松翰科技（深圳）有限公司
地址：深圳市南山区高新技术产业园南区 T2-B 栋 2 层
电话：86-755-2671 9666
传真：86-755-2671 9786
技术支持：
Sn8fae@SONiX.com.tw