SN8PC21

用户参考手册

Version 1.0

SONiX 8 位单片机

修改记录

版本	时间	修改说明
VER 1.0	2010.07	初版。

目　录

1 产品简介

1.1 功能特性

- **存储器配置**
  OTP ROM：1K * 16 位。
  RAM：32 * 8 位。

- **4 层堆栈缓存器**
- **I/O 引脚配置**
  双向输入输出端口：P0、P1。
  400mA IR 输出引脚：IROUT。
  具有唤醒功能的引脚：P0、P1 电平变换触发。
  具有上拉电阻的端口：P0、P1。

- **1 级 LVD**
- **指令周期（Fcpu）**
  Fcpu = Fosc/16

- **功能强大的指令集**
  单周期指令系统（1T）
  绝大部分指令只需要一个周期。
  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。

- **无中断功能**
- **1 个 8 位基本定时器 T0**
  T0：基本定时器，具有标志功能。

- **1 通道可编程控制占空比/周期的 IR 输出**
- **4 种系统时钟**
  外部高速时钟：RC，高达 8MHz。
  外部高速时钟：晶体，高达 8MHz。
  内部高速时钟：RC，8MHz。
  内部低速时钟：RC，16KHz（3V）。

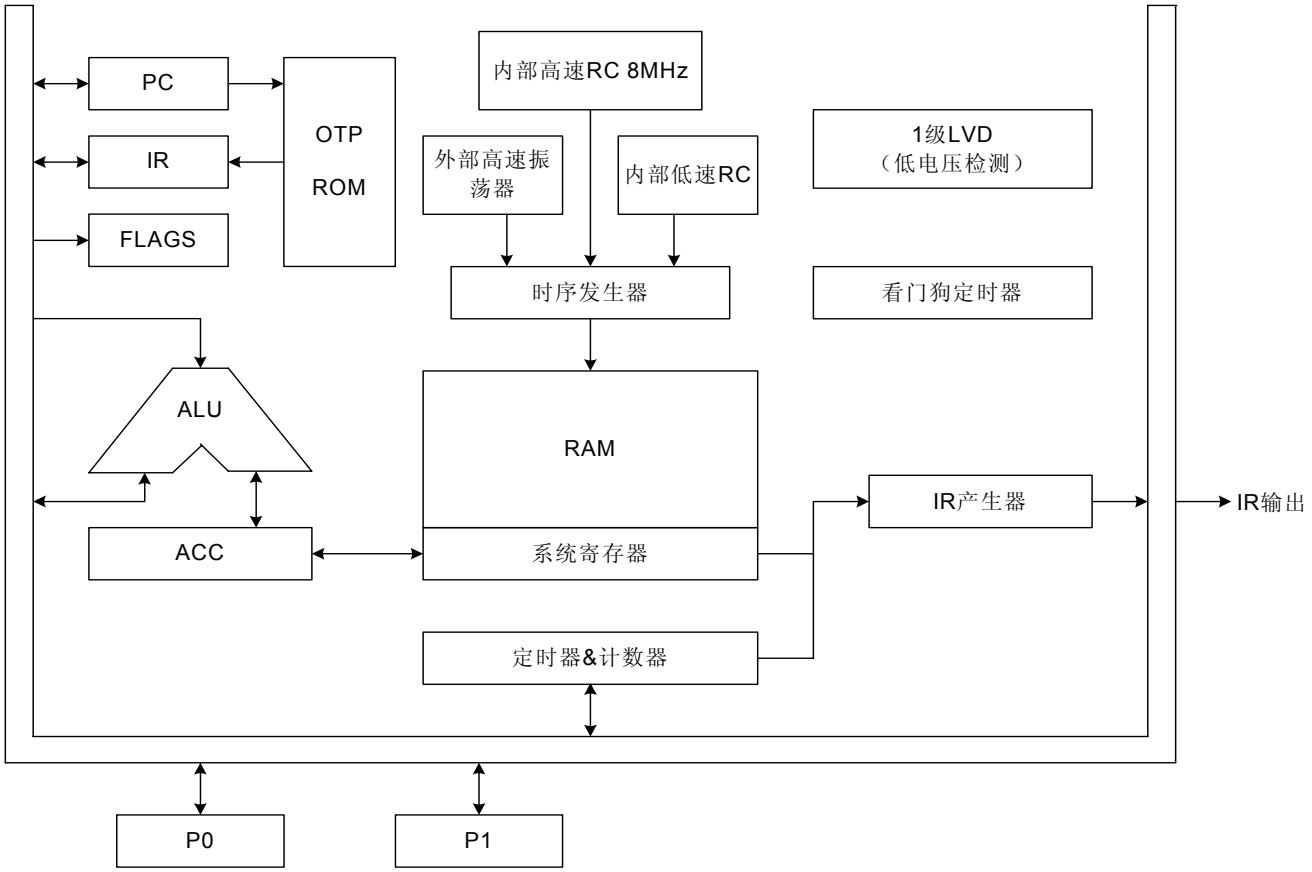
- **2 种工作模式**
  普通模式：高低速时钟都正常工作。
  睡眠模式：高低速时钟都停止工作。

- **封装形式**
  PDIP 16 pin
  SOP 16 pin

☞ **产品特性比较表**

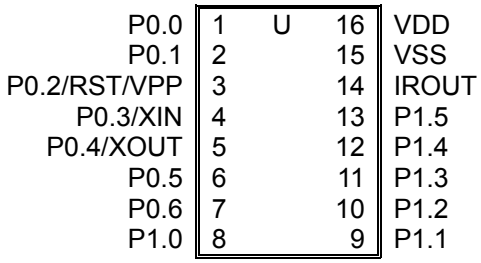
单片机型号	ROM	RAM	堆栈	T0	振荡器			I/O	IR 输出	唤醒功能引脚数目	封装形式
					Ext. 455K	Ext. 4M	Int. 8M				
SN8PC01	0.5K*16	32	4	-	V	-	-	16	固定为 38KHz，正常的电流	9	PDIP20/SOP20
SN8PC13	2K*16	48	4	V	-	V	-	16	占空比/周期可编程控制，正常的电流	15	PDIP20/SOP20/SSOP20
SN8PC20	2K*16	56	4	V	V	V	V	18	占空比/周期可编程控制，400mA 的灌电流	16	PDIP20/SOP20/SSOP20
SN8PC21	1K*16	32	4	V	-	V	V	13	占空比/周期可编程控制，400mA 的灌电流	13	PDIP16/SOP16

1.2 系统框图


PC
IR
FLAGS
OTP
ROM
内部高速RC 8MHz
外部高速振荡器
内部低速RC
1级LVD
（低电压检测）
时序发生器
看门狗定时器
ALU
RAM
IR产生器
IR输出
ACC
系统寄存器
定时器&计数器
P0
P1

1.3 引脚配置

SN8PC21P (PDIP 16 pins)
SN8PC21S (SOP 16 pins)

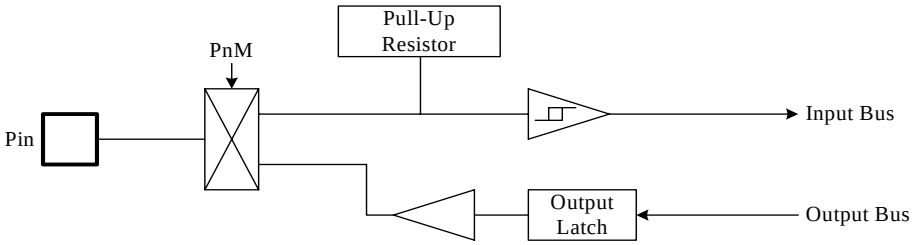


1.4 引脚说明

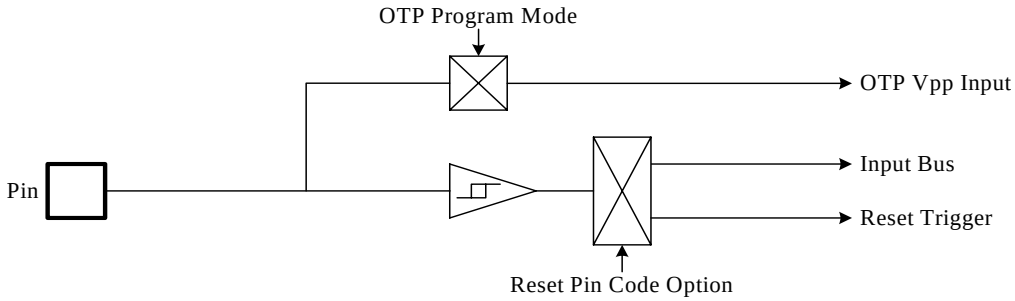
引脚名称	类型	功能说明
VDD, VSS	P	电源输入端。
P0.2/RST/ VPP	I, P	RST：系统外部复位输入引脚，施密特触发，低电平有效，通常保持高电平。
		VPP：OTP 烧录电源输入引脚。
		P0.2：单向输入引脚，施密特触发，无上拉电阻。
XIN/P0.3	I/O	XIN：外部振荡器输入引脚。
		P0.3：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时唤醒系统。
XOUT/P0.4	I/O	XOUT：外部振荡器输出引脚。
		P0.4：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时唤醒系统。
P0[6:0]	I/O	双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时唤醒系统。
P1[5:0]	I/O	双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时唤醒系统。
IROUT	O	占空比/周期可编程控制的 IR 信号输出引脚。

1.5 引脚电路结构图

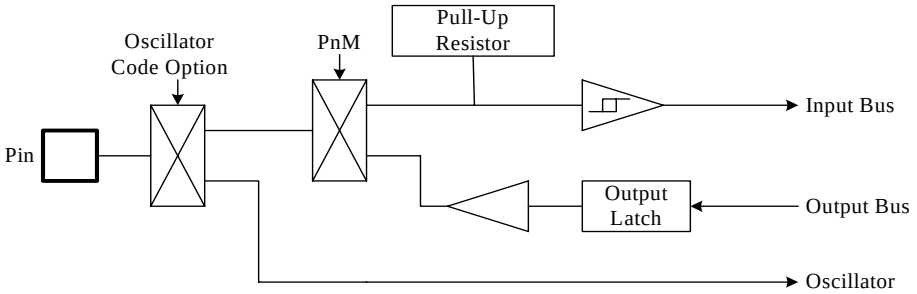
✸ **I/O 引脚，施密特触发（ViH=0.7*Vdd，ViL=0.3*Vdd），上拉电阻（200KΩ@3V）：**



● **单向输入引脚，施密特触发（ViH=0.7*Vdd，ViL=0.3*Vdd），Vpp 高电压：**



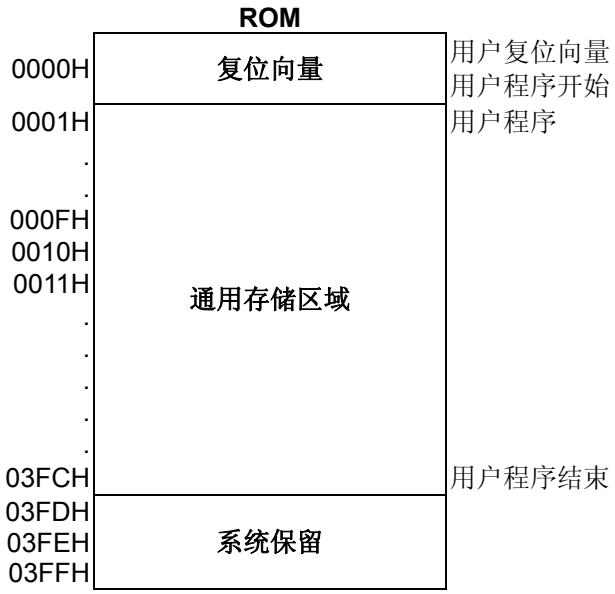
✸ **振荡器引脚，与 GPIO 引脚共用，施密特触发（ViH=0.7*Vdd，ViL=0.3*Vdd），上拉电阻（200KΩ@3V）：**



2 中央处理器（CPU）

2.1 程序存储器（ROM）

☞ **ROM：1K**



ROM 包括复位向量，通用存储区域和系统保留区域：复位向量是程序的开始地址；通用存储区域则是程序存储区，包括主循环，子程序和数据表。

2.1.1 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:                          ; 用户程序起始地址。
        …                       ; 用户程序。
        …
        ENDP                    ; 程序结束。
```

2.1.2 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
                …
```

2.1.3 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：如果跳转表跨越 ROM 边界，将引起程序错误。**

```
@JMP_A  MACRO   VAL
        IF      (($+1) !& 0XFF00) !!= (($+(VAL)) !& 0XFF00)
        JMP     ($ | 0XFF)
        ORG     ($ | 0XFF)
        ENDIF
        B0ADD   PCL, A
        ENDM
```

✷ **注：“VAL”为跳转表列表中列表个数。**

➢ **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。
```

如果跳转表恰好位于 ROM BANK 边界处（00FFH~0100H），宏指令“@JMP_A”将调整跳转表到适当的位置（0100H）。

➢ **例：“@JMP_A”运用举例**

```
; 编译前
ROM 地址
        B0MOV   A, BUF0         ; “BUF0”从 0 到 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 到 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。
```

2.1.4 CHECKSUM 计算

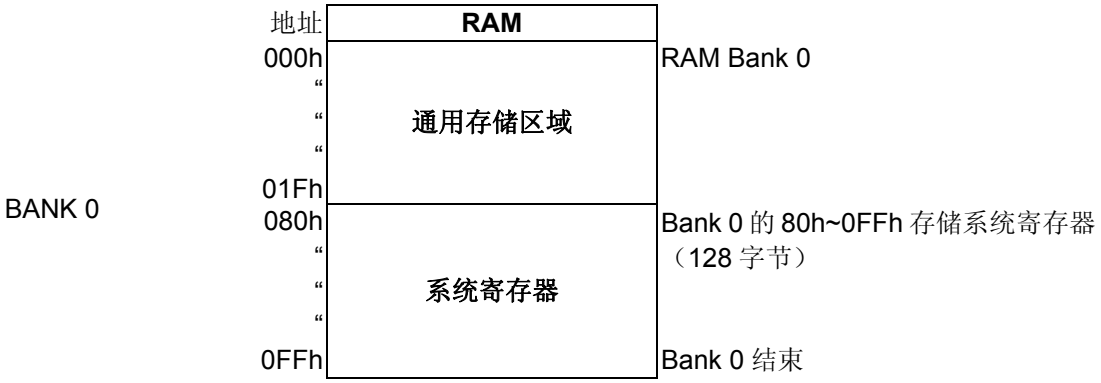
ROM 的末端位置的几个字限制使用，进行 Checksum 计算时，用户应避免对该单元的访问。

➢ **例：示例程序演示了如何对 00H 到用户程序结束进行 Checksum 计算。**

```
                MOV     A,#END_USER_CODE$L
                B0MOV   END_ADDR1, A        ; 用户程序终端地址低位字节存入 end_addr1。
                MOV     A,#END_USER_CODE$M
                B0MOV   END_ADDR2, A        ; 用户程序终端地址中间字节存入 end_addr2。
                CLR     Y                   ; 清 Y。
                CLR     Z                   ; 清 Z。
@@:
                MOVC
                B0BCLR  FC                  ; 清标志位 C。
                ADD     DATA1, A            ;
                MOV     A, R
                ADC     DATA2, A            ;
                JMP     END_CHECK           ; 检查 YZ 地址是否为代码的结束地址。
AAA:
                INCMS   Z                   ;
                JMP     @B                  ; 如果 Z != 00H，进行下一个计算。
                JMP     Y_ADD_1             ; 如果 Z = 00H，Y 加 1。
END_CHECK:
                MOV     A, END_ADDR1
                CMPRS   A, Z                ; 检查 Z 地址是否为用户程序结束地址低位地址。
                JMP     AAA                 ; 否，则进行 checksum 计算。
                MOV     A, END_ADDR2
                CMPRS   A, Y                ; 是则检查 Y 的地址是否为用户程序结束地址中间地址。
                JMP     AAA                 ; 否，则进行 Checksum 计算。
                JMP     CHECKSUM_END        ; 是则 Checksum 计算结束。

Y_ADD_1:
                INCMS   Y
                NOP
                JMP     @B                  ; 转至 checksum 计算。
CHECKSUM_END:
                …
                …
END_USER_CODE:
```

2.2 数据存储器（RAM）

☞ **RAM：32 X 8 位**



Sonix 提供“Bank 0”型的指令（如 B0MOV、B0ADD、B0BTS1、B0BTS0 等）直接访问 Bank 0 RAM。

2.2.1 系统寄存器

2.2.1.1 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	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B	-	-	-	-	-	-	-	-	P0M	-	-	-	-	-	-	-
C	-	P1M	-	-	-	-	-	-	INTRQ	-	OSCM	-	WDTR	IRR	PCL	PCH
D	P0	P1	-	-	-	-	-	-	T0M	T0C	IRM	IRC	-	-	-	STKP
E	-	-	-	-	-	-	-	@YZ	IRD	-	-	-	-	-	-	-
F	-	-	-	-	-	-	-	-	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.2.1.2 系统寄存器说明

R = 工作寄存器和 ROM 查表数据缓存器
PnM = Pn 模式控制寄存器
INTRQ = 中断请求寄存器
OSCM = 振荡器模式寄存器
IRR = IR 计数器自动重装数据缓存器
Pn = Pn 数据缓存器
T0C = T0 计数寄存器
IRC = IR 计数寄存器
@YZ = 间接寻址寄存器

Y, Z = 专用寄存器，@YZ 间接寻址寄存器，ROM 寻址寄存器
PFLAG = 特殊标志寄存器
WDTR = 看门狗定时器清零寄存器
PCH, PCL = 程序计数器
T0M = T0 模式寄存器
IRM = IR 模式寄存器
STKP = 堆栈指针
IRD = IR 占空比控制寄存器
STK0~STK3 = 堆栈缓存器

2.2.1.3 系统寄存器的位定义

地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	注释
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
0B8H		P06M	P05M	P04M	P03M		P01M	P00M	R/W	P0M
0C1H			P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C8H				T0IRQ					R/W	INTRQ
0CAH					CPUM0				R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CDH	IRR7	IRR6	IRR5	IRR4	IRR3	IRR2	IRR1	IRR0	W	IRR
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH							PC9	PC8	R/W	PCH
0D0H		P06	P05	P04	P03	P02	P01	P00	R/W	P0
0D1H			P15	P14	P13	P12	P11	P10	R/W	P1
0D8H	T0ENB	T0rate2	T0rate1	T0rate0					R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH								CREN	R/W	IRM
0DBH	IRC7	IRC6	IRC5	IRC4	IRC3	IRC2	IRC1	IRC0	R/W	IRC
0DFH	GIE						STKPB1	STKPB0	R/W	STKP
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0E8H	IRD7	IRD6	IRD5	IRD4	IRD3	IRD2	IRD1	IRD0	W	IRD
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H							S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH							S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH							S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH							S0PC9	S0PC8	R/W	STK0H

✷ 注：

1. 所有寄存器名都已在SN8ASM编译器中做了宣告；

2. 用户使用SN8ASM编译器对寄存器的位进行操作时，须在该寄存器的位前加“F”；

3. 指令“b0bset”，“b0bclr”，“bset”，“bclr”只能用于可读写的寄存器（“R/W”）。

2.2.2 累加器

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。**

; 将立即数写入 ACC。

```
MOV     A, #0FH
```

;把 ACC 中的数据存入 BUF 中。

```
MOV     BUF, A
B0MOV   BUF, A
```

; 把 BUF 中的数据送到 ACC 中。

```
MOV     A, BUF
B0MOV   A, BUF
```

2.2.3 程序状态寄存器 PFLAG

寄存器 PFLAG 中包含 ALU 运算状态信息和系统复位状态信息，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。

NT0	NPD	复位状态
0	0	看门狗复位
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/分支运算的结果为零；

0 = 算术/逻辑/分支运算的结果非零。

✷ **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。**

2.2.4 程序计数器

程序计数器 PC 是一个 10 位二进制程序地址寄存器，分高 2 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	-	-	-	-	-	-	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
复位后	-	-	-	-	-	-	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     ; 如果 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。
                …
                …
```

2.2.5 Y, Z 寄存器

寄存器 Y 和 Z 都是 8 位缓存器，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, #7FH     ; 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:                    ;
        …
```

2.2.6 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 寄存器查表应用请参考查表章节。**

2.3 寻址模式

2.3.1 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。**

2.3.2 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
```

2.3.3 间接寻址模式

通过指针寄存器（Y/Z）访问 RAM 数据。

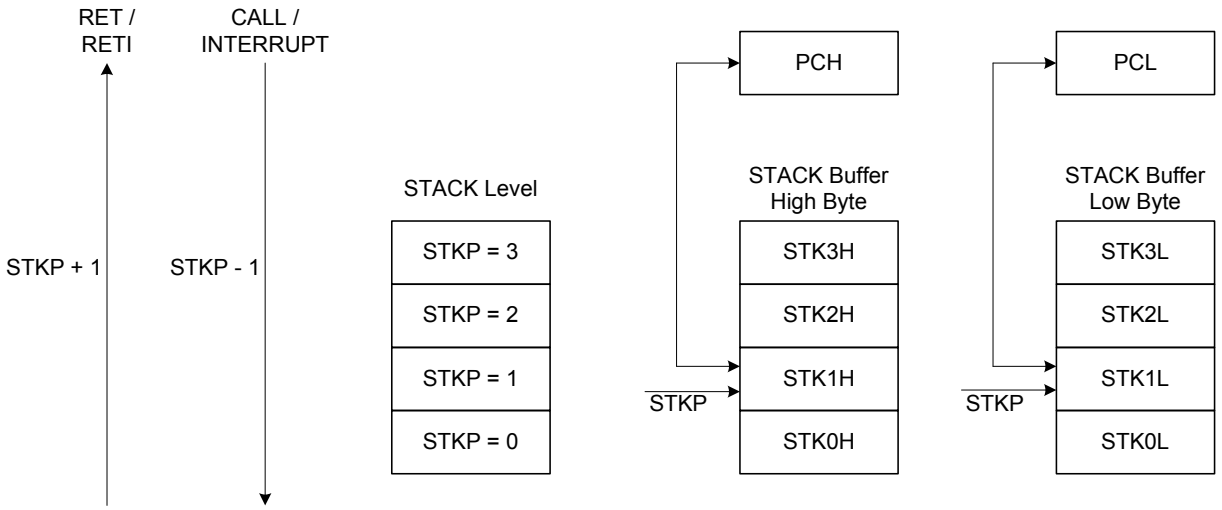
- **例：用 @YZ 实现间接寻址。**

```
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
```

2.4 堆栈

2.4.1 概述

SN8PC21 的堆栈缓存器共有 4 层，程序执行 CALL 指令时，用来存储程序计数器 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。



2.4.2 堆栈寄存器

堆栈指针 STKP 是一个 2 位寄存器，存放被访问的堆栈单元地址，10 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。

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	-	-	-	-	-	-	STKPB1	STKPB0
读/写	-	-	-	-	-	-	R/W	R/W
复位后	-	-	-	-	-	-	1	1

Bit[1:0] **STKPBn**：堆栈指针（n = 0 ~ 1）。

- **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, #00000011B
B0MOV   STKP, A
```

0F0H~0F8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnH	-	-	-	-	-	-	SnPC9	SnPC8
读/写	-	-	-	-	-	-	R/W	R/W
复位后	-	-	-	-	-	-	0	0

0F0H~0F8H	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 = 3 ~ 0)

2.4.3 堆栈操作举例

执行程序调用指令 CALL 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		堆栈缓存器		备注
	STKPB1	STKPB0	高字节	低字节	
0	1	1	Free	Free	-
1	1	0	STK0H	STK0L	-
2	0	1	STK1H	STK1L	-
3	0	0	STK2H	STK2L	-
4	1	1	STK3H	STK3L	-
> 4	1	0	-	-	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。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冲器。堆栈恢复操作如下表所示：

堆栈层数	STKP		堆栈缓存器		备注
	STKPB1	STKPB0	高字节	低字节	
4	1	1	STK3H	STK3L	-
3	0	0	STK2H	STK2L	-
2	0	1	STK1H	STK1L	-
1	1	0	STK0H	STK0L	-
0	1	1	Free	Free	-

2.5 编译选项列表（CODE OPTION）

编译选项（CODE OPTION）是一种系统的硬件配置，包括振荡器类型，看门狗定时器的操作，复位引脚选项以及 OTP ROM 的安全控制。如下表所示：

编译选项	配置项目	功能说明
High_Clk	IHRC_8M	内部高速 8MHz RC，XIN/XOUT 引脚为 GPIO 引脚。
	RC	外部高速振荡器采用廉价的 RC 振荡电路，XIN 引脚外接 RC 电路，XOUT 引脚为 GPIO 引脚。
	4M X'tal	外部高速振荡器采用标准陶瓷/石英振荡器（如 4M~8MHz）。
Watch_Dog	Always_On	始终开启看门狗定时器，睡眠模式下也不例外。.
	Enable	开启看门狗定时器，但在睡眠模式下会处于关闭状态。
	Disable	关闭看门狗定时器。
Reset_Pin	Reset	使能外部复位引脚。
	P02	使能 P0.2 单向输入功能，无上拉电阻。
Security	Enable	ROM 代码加密。
	Disable	ROM 代码不加密。

2.5.1 Reset_Pin 编译选项

复位引脚与单向输入引脚共用，由编译选项控制。

- **Reset**：使能外部复位引脚功能。当下降沿触发时，系统复位。
- **P02**：使能 P0.2 为单向输入引脚。此时禁止外部复位引脚功能。

2.5.2 Security 编译选项

Security 编译选项是对 OTP ROM 的一种保护，当使能 Secuity 编译选项，ROM 代码加密，可以保护 ROM 的内容。

3 复位

3.1 概述

SN8PC21 系列的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（使能外部复位引脚时有效）。

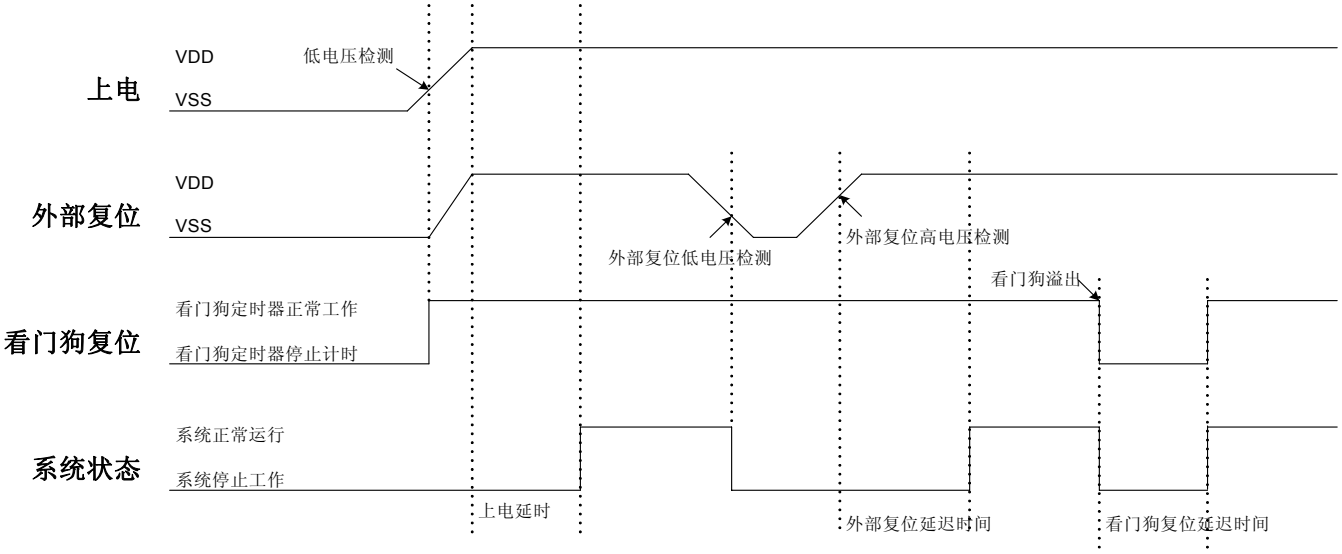
上述任一种复位发生时，所有的系统寄存器恢复默认状态，程序停止运行，同时程序计数器 PC 清零。复位结束后，系统从向量 0000H 处重新开始运行。PFLAG 寄存器的 NT0 和 NPD 两个标志位能够给出系统复位状态的信息。用户可以根据 NT0 和 NPD 的状态，编程控制系统的运行路径。

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	复位类型	复位条件
0	0	看门狗复位	看门狗定时器溢出
0	1	系统保留	-
1	0	上电复位和 LVD 复位	电源电压低于 LVD 检测电压
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响应时间，系统提供完善的复位流程以保证复位动作的顺利进行。对于不同类型的振荡器，完成复位所需要的时间也不同。因此，VDD 的上升速度和不同晶振的起振时间都不固定。RC 振荡器的起振时间最短，晶体振荡器的起振时间则较长。在用户使用的过程中，应考虑系统对上电复位时间的要求。系统复位时序图如下：



3.2 上电复位

上电复位与 LVD 操作密切相关。系统上电过程呈逐渐上升的曲线形式，需要一定时间才能达到正常电平值。下面给出上电复位的正常时序：

- **上电**：系统检测到电源电压上升并等待其稳定；
- **外部复位（使能外部复位引脚时才有效）**：系统检测外部复位引脚状态。如果不为高电平，系统保持复位状态直到外部复位引脚的复位结束。
- **系统初始化**：所有的系统寄存器被置为默认状态；
- **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。

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**：所有的系统寄存器被置为默认状态；
- **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。

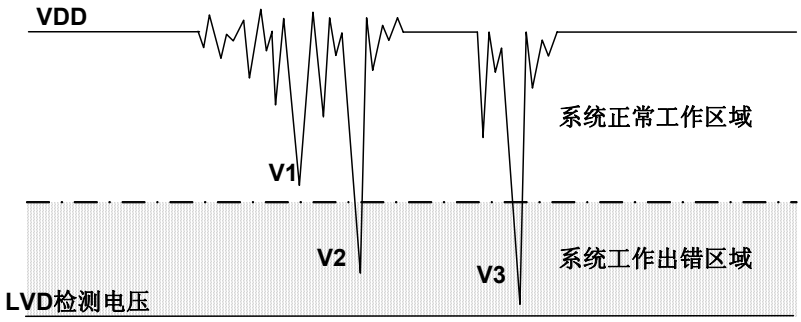
看门狗定时器应用注意事项如下：

- 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；
- 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。

✷ **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

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。

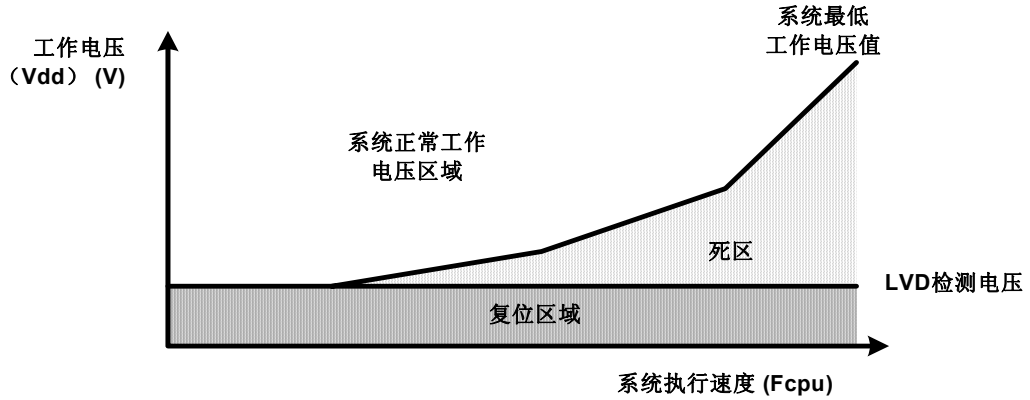
AC 运用中：

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。

3.4.1 系统工作电压

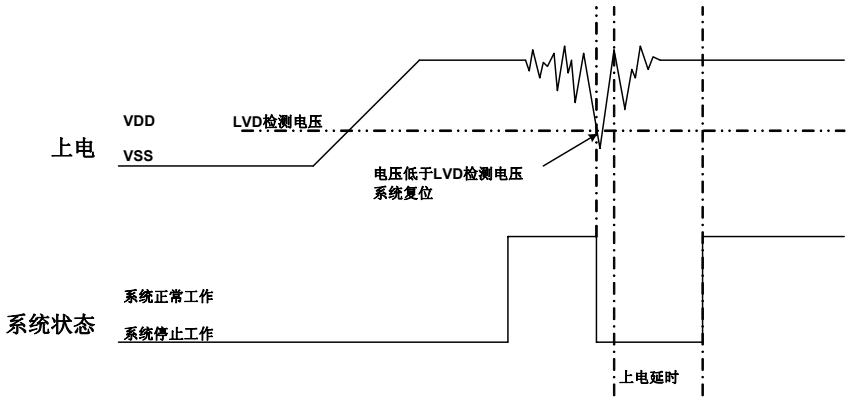
为了改善系统掉电复位的性能，首先必须明确系统具有的最低工作电压值。系统最低工作电压与系统执行速度有关，不同的执行速度下最低工作电压值也不同。



系统工作电压与执行速度关系图

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。

3.4.2 低电压检测（LVD）



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，则 LVD 就不能起到保护作用，就需要采用其它复位方法。

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；**

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 复位电路。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

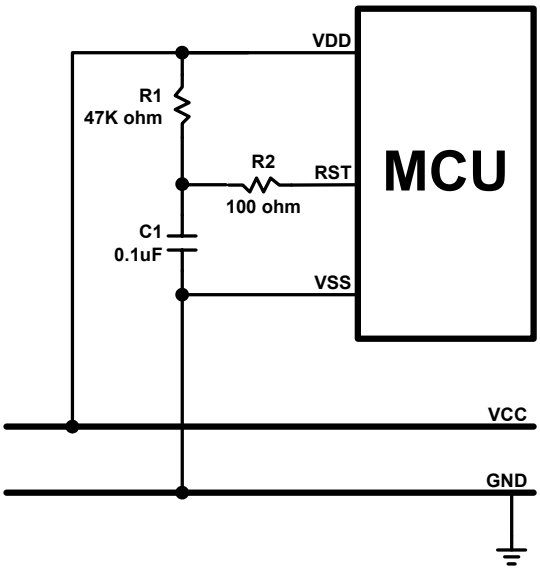
外部复位功能由编译选项“Reset_Pin”控制。将该编译选项置为“Reset”，可使能外部复位功能。外部复位引脚为施密特触发结构，低电平有效。复位引脚处于高电平时，系统正常运行。当复位引脚输入低电平信号时，系统复位。外部复位操作在上电和正常工作模式时有效。需要注意的是，在系统上电完成后，外部复位引脚必须输入高电平，否则系统将一直保持在复位状态。外部复位的时序如下：

- **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

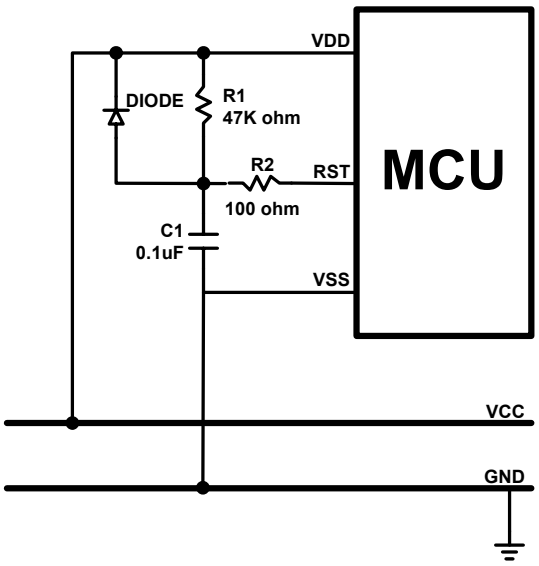
3.6.1 基本 RC 复位电路



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。**

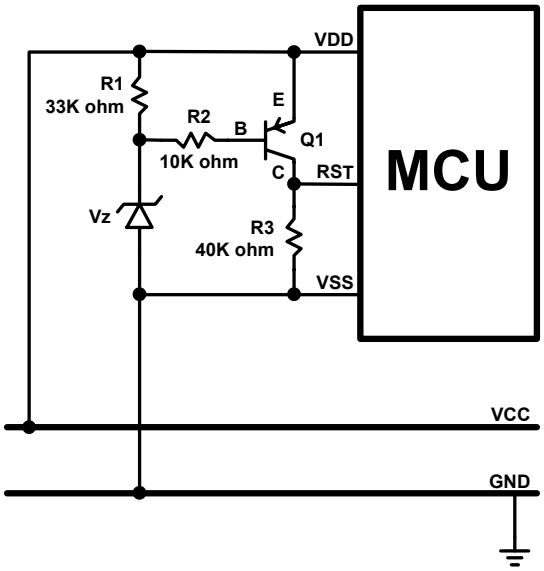
3.6.2 二极管& RC 复位电路



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。

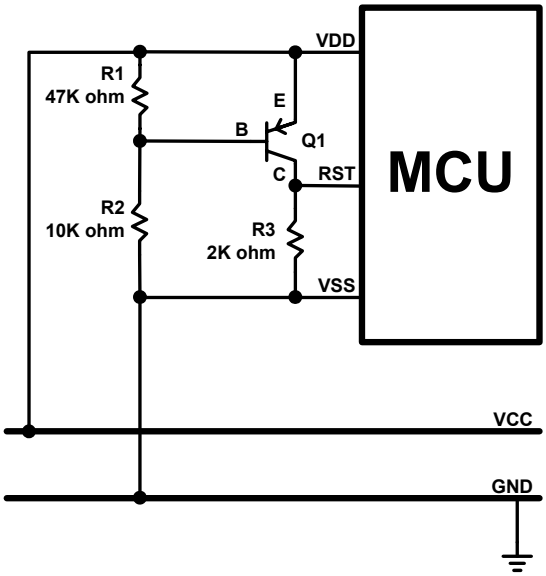
✷ **注："基本 RC 复位电路"和"二极管及 RC 复位电路"中的电阻 R2 都是必不可少的限流电阻，以避免复位引脚 ESD（Electrostatic Discharge）或 EOS（Electrical Over-stress）击穿 。**

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 电压偏置复位电路

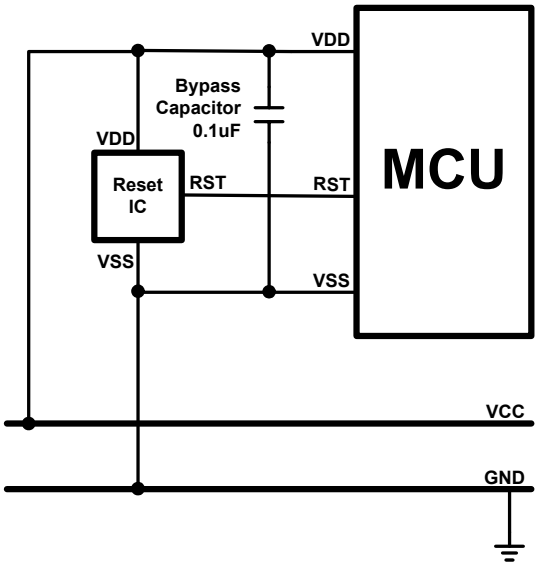


电压偏置复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。与稳压二极管复位电路相比，这种复位电路的检测电压值的精确度有所降低。电路中，R1 和 R2 构成分压电路，当 VDD 高于和等于分压值“0.7V x (R1 + R2) / R1”时，三极管集电极 C 输出高电平，单片机正常工作；VDD 低于“0.7V x (R1 + R2) / R1”时，集电极 C 输出低电平，单片机复位。

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



外部复位也可以选用 IC 进行外部复位，但是这样一来系统成本将会增加。针对不用的应用要求选择适当的复位 IC，如上图所示外部 IC 复位电路，能够有效的降低电源变化对系统的影响。

4 系统时钟

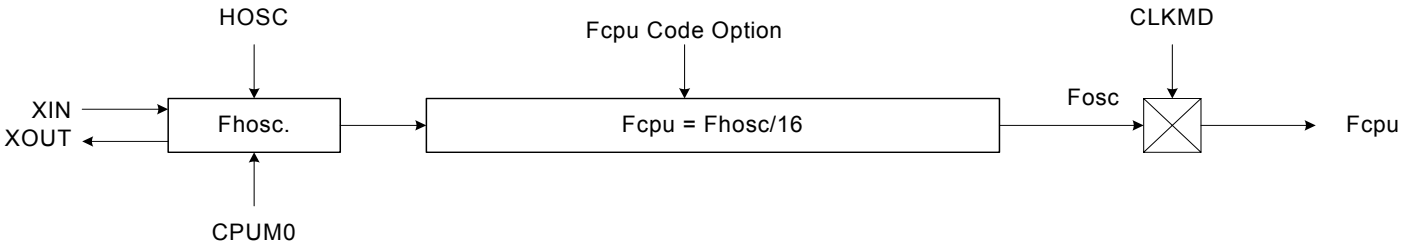
4.1 概述

SN8PC21 内置双时钟系统：高速时钟和低速时钟。高速时钟包括外部高速时钟和内置 8MHz RC 时钟（IHRC 8MHz）。低速时钟由内部低速 RC 时钟提供（ILRC 16KHz @3V）。

高速时钟可以作为系统时钟（Fosc），低速时钟仅支持看门狗定时器。

- **普通模式（高速时钟）：Fcpu = Fhosc/16。**

4.2 时钟框图



- HOSC：High_CLK 编译选项。
- Fhosc：外部高速时钟/内部高速 RC 时钟。
- Fosc：系统时钟源。
- Fcpu：指令周期。

4.3 OSCM 寄存器

寄存器 OSCM 控制振荡器的状态和系统的工作模式。

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

Bit3 **CPUM0**：CPU 工作模式控制位。
0 = 普通模式；
1 = 睡眠模式。

- **例：进入睡眠模式后，高速时钟和内部低速时钟都停止工作。**

```
B0BSET      FCPUM0
```

4.4 系统高速时钟

系统高速时钟包括内部 8MHz RC 时钟和外部振荡时钟，由编译选项 High_Clk 控制。

High_Clk 编译选项	功能说明
IHRC_8M	内部高速 8MHz RC，XIN/XOUT 引脚为 GPIO 引脚。
RC	外部高速振荡器采用廉价的 RC 振荡电路，XIN 引脚外接 RC 电路，XOUT 引脚为 GPIO 引脚。
4M	外部高速振荡器采用标准陶瓷/石英振荡器，典型值为 4M。

4.4.1 内部高速 RC 时钟

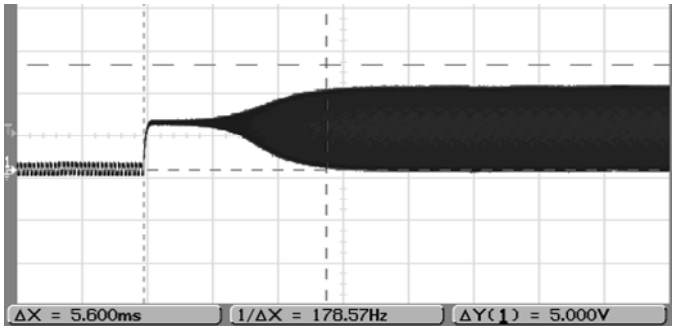
内部高速 RC 8MHz 时钟由编译选项 IHRC_8M 控制，在 IHRC_8M 模式下，系统时钟源由内部 8MHz RC 时钟提供，XIN/XOUT 引脚为 GPIO 引脚。

- **IHRC_8M**：系统时钟为内部 8MHz RC 时钟，XIN/XOUT 引脚为 GPIO 引脚。

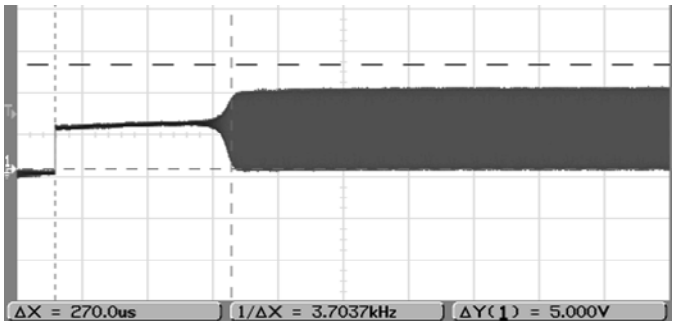
4.4.2 外部高速时钟

外部高速时钟由外部晶体振荡器和 RC 振荡电路提供，由编译选项 High_Clk 控制。晶体/陶瓷振荡器和 RC 振荡器的启动时间并不一致，RC 的启动时间很短，晶体振荡器的启动时间要稍长。单片机的复位时间与振荡器的启动时间息息相关。

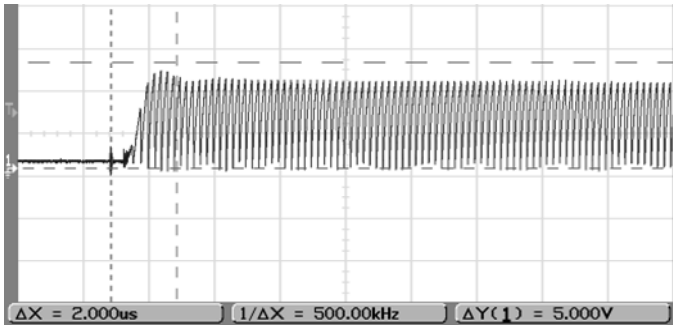
4MHz Crystal



4MHz Ceramic

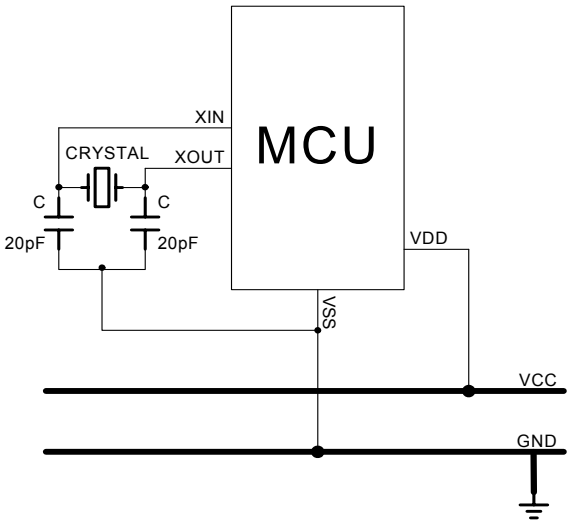


RC



4.4.2.1 晶体/陶瓷振荡器

晶体/陶瓷振荡器连接到 XIN，XOUT 引脚，针对高速/标准/低速频率而言，其驱动电流是不同的。编译选项 High_Clk 支持不同的频率，如高速 8MHz，标准 4MHz。

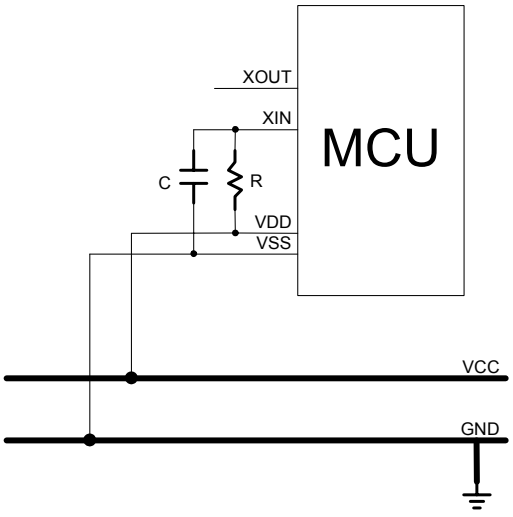


4M/8M 配置的电容为 20pF

✷ **注：晶体振荡器和电容 C 要尽可能的靠近单片机的 XIN/XOUT/VSS 引脚。**

4.4.2.2 RC

通过编译选项 High_Clk 选择 RC 振荡器，频率为 10MHz，电阻 R 的值决定频率的大小，电容 C 的值建议为 50P~100P。XOUT 引脚为 GPIO 引脚。



✷ **注：电阻 R 和电容 C 要尽可能的靠近单片机的 VDD 和 XIN 引脚。**

4.5 系统时钟测试

在设计过程中，用户可通过软件指令周期对系统时钟速度进行测试。

➢ **例：外部振荡器的 Fcpu 指令周期测试。**

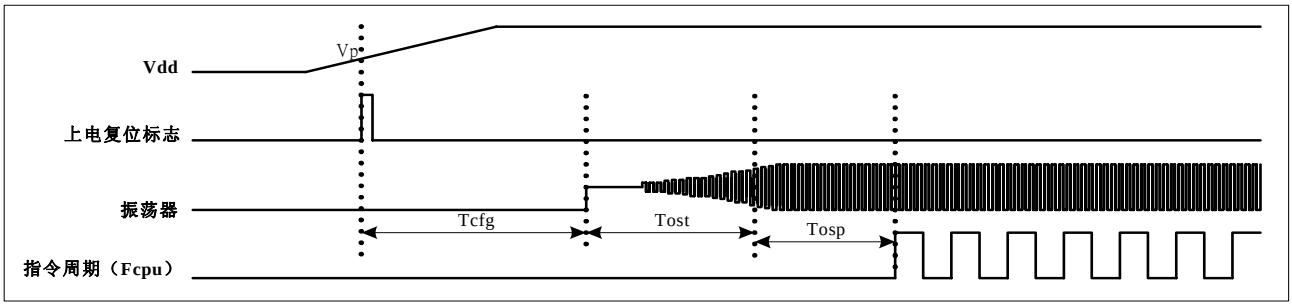
```
        B0BSET      P0M.0           ; P0.0 置为输出模式以输出 Fcpu 的触发信号。

@@:
        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。**

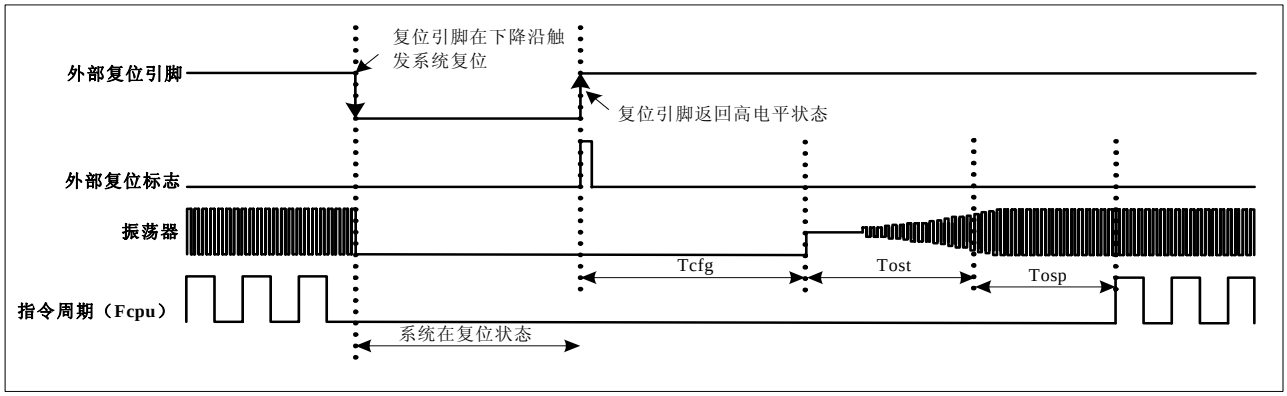
4.6 系统时钟时序

参数	符号	说明	典型值
硬件配置时间	Tcfg	2048*FILRC	128ms @ FILRC = 16KHz
振荡器启动时间	Tost	启动时间取决于振荡器的材料、工艺等。通常情况下，低速振荡器的启动时间要比高速振荡器的启动时间慢，RC 振荡器的启动时间要比晶体/陶瓷振荡器的启动时间快。	-
振荡器起振时间	Tosp	复位情况下的振荡器起振时间为 2048*Fhosc（使能上电复位，LVD 复位，看门狗复位，外部复位引脚）	512us @ Fhosc = 4MHz 256us @ Fhosc = 8MHz
		睡眠模式唤醒情况的振荡器起振时间为： 2048*Fhosc ……晶体/陶瓷振荡器，如 32768Hz 晶振，4MHz 晶振，16MHz 晶振等； 32*Fhosc……RC 振荡器，如外部 RC 振荡电路，内部高速 RC 振荡器。	512us @ Fhosc = 4MHz 256us @ Fhosc = 8MHz

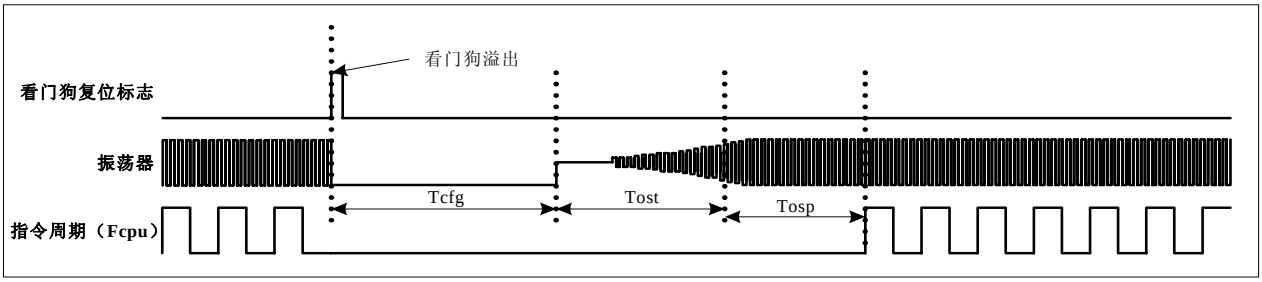
- 上电复位时序：



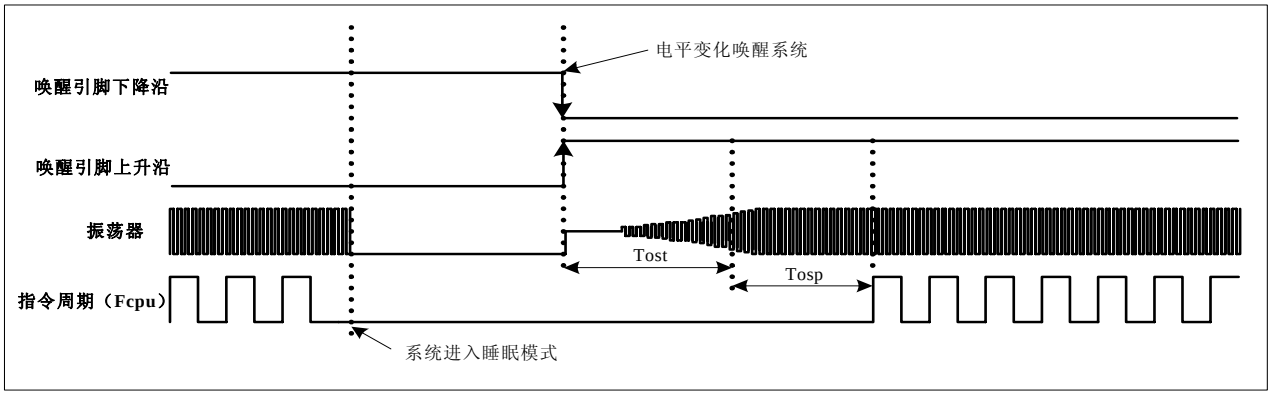
- 外部复位引脚复位时序：



- 看门狗复位时序：

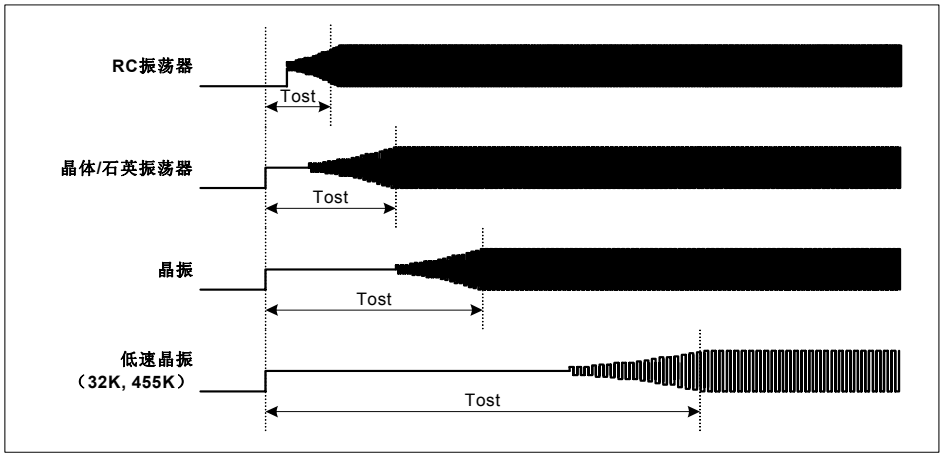


- **睡眠模式唤醒时序：**



- **振荡器启动时间**

启动时间取决于振荡器的材料、工艺等。通常情况下，低速振荡器的启动时间要比高速振荡器的启动时间慢，RC 振荡器的启动时间要比晶体/陶瓷振荡器的启动时间快。

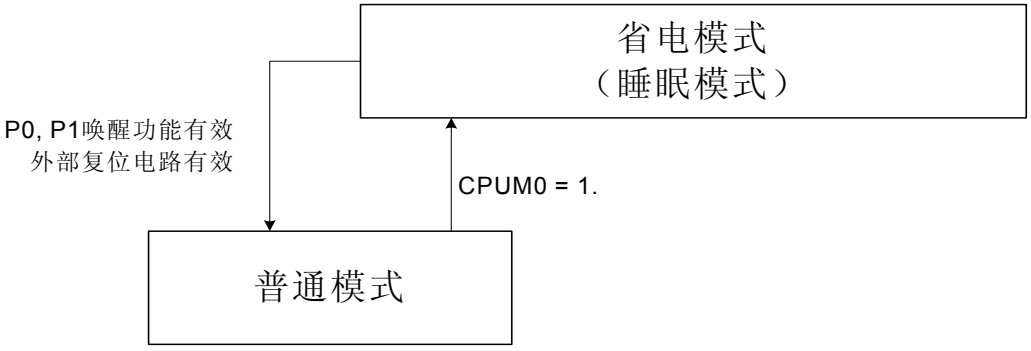


5 系统工作模式

5.1 概述

SN8PC21 具有低功耗的特性，可以在两种不同的工作模式下工作：

- **普通模式；**
- **省电模式（睡眠模式）。**



系统工作模式切换示意图

工作模式说明

工作模式	普通模式	睡眠模式	备注
EHOSC	运行	停止	
IHRC	运行	停止	
ILRC	运行	停止	
CPU 指令	执行	停止	
T0 定时器	*有效	无效	* T0ENB=1 时有效
看门狗定时器	Watch_Dog 编译选项	Watch_Dog 编译选项	参考编译选项章节
唤醒源	-	P0，P1，复位	

- **EHOSC**：外部高速振荡器。
- **IHRC**：内部高速振荡器（8M RC 振荡器）。
- **ILRC**：内部低速振荡器（16KHz RC @3V）。

5.2 普通模式

普通模式是系统高速时钟正常工作模式，系统时钟源由高速振荡器提供。程序被执行。上电复位或任意一种复位触发后，系统进入普通模式执行程序。当系统从睡眠模式被唤醒后进入普通模式。普通模式下，高速振荡器正常工作，功耗最大。

- 程序被执行，所有的功能都可控制。
- 系统速率为高速。
- 高速振荡器和内部低速 RC 振荡器都正常工作。
- 通过 OSCM 寄存器，系统可以从普通模式切换到其它任何一种工作模式。
- 系统从睡眠模式唤醒后进入普通模式。

5.3 睡眠模式

睡眠模式是系统的理想状态，不执行程序，振荡器也停止工作。整个芯片的功耗低于 1uA。睡眠模式可以由 P0、P1 的电平变换触发唤醒。P1 的唤醒功能由 P1W 寄存器控制。从任何工作模式进入睡眠模式，被唤醒后都返回到普通模式。由 OSCM 寄存器的 CPUM0 位控制是否进入睡眠模式，当 CPUM0=1，系统进入睡眠模式。当系统从睡眠模式被唤醒后，CPUM0 被自动禁止（0 状态）。

- 程序停止执行，所有的功能被禁止。
- 所有的振荡器，包括外部高速振荡器、内部高速振荡器和内部低速振荡器都停止工作。
- 功耗低于 1uA。
- 系统从睡眠模式被唤醒后进入普通模式。
- 睡眠模式的唤醒源为 P0 和 P1 电平变换触发。

5.4 工作模式控制宏

Sonix 提供工作模式控制宏以方便系统工作模式的切换。

宏名称	长度	说明
SleepMode	1-word	系统进入睡眠模式。

➢ **例：从普通模式切换进入睡眠模式。**

```
SleepMode                    ; 直接宣告“SleepMode”宏。
```

5.5 系统唤醒

5.5.1 概述

睡眠模式下，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。唤醒触发信号为外部触发信号（P0的电平变换）。

- 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）。

5.5.2 唤醒时间

系统进入睡眠模式后，高速时钟停止运行。把系统从睡眠模式下唤醒时，单片机需要等待 2048 个外部高速振荡器时钟周期以使振荡电路进入稳定工作状态，等待的这一段时间就称为唤醒时间。唤醒时间结束后，系统才进入到普通模式。

唤醒时间计算如下：

唤醒时间 = 1/Fosc * 2048（sec）+ 高速时钟启动时间

✷ **注：高速时钟的启动时间与 VDD 和振荡器类型有关。**

➢ **例：将系统从睡眠模式中唤醒，并设置系统进入普通模式。唤醒时间计算如下。**

唤醒时间= 1/Fosc * 2048 = 0.512 ms (Fosc = 4MHz)
总的唤醒时间 = 0.512 ms + 振荡器启动时间

6 I/O 口

6.1 概述

SN8PC20 内带 13 个 I/O 引脚，包括 P0 和 P1。这些 I/O 引脚在设置为输入模式时都内置上拉电阻。

✷ **注：GPIO 引脚设置为输入模式时，使能内置上拉电阻。**

部分 I/O 引脚都和模拟引脚及特殊功能的引脚共用，I/O 引脚的共用情况如下表所示：

I/O 引脚		共用引脚		共用引脚控制条件
引脚名称	类型	引脚名称	类型	
P0.2	I	RST	DC	Reset_Pin code option = Reset
		VPP	HV	OTP Programming
P0.4	I/O	XOUT	AC	High_CLK code option = 455K, 4M, 8M
P0.3	I/O	XIN	AC	High_CLK code option = RC, 455K, 4M, 8M

*** DC：数字特性，AC：模拟特性，HV：高压特性。**

6.2 I/O 口模式

寄存器 PnM 控制 I/O 的工作模式。

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

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1M	-	-	P15M	P14M	P13M	P12M	P11M	P10M
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 模式选择控制位（n = 0~1）。
0 = 输入模式；
1 = 输出模式。

✷ **注：**
1. **用户可通过位操作指令（B0BSET、B0BCLR）对 I/O 口进行编程控制；**
2. **P0.2 是单向输入引脚，P0M.2 未定义。**

➢ **例：I/O 模式选择。**

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

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

B0BCLR  P1M.0       ; 设置 P1.0 为输入模式。

B0BSET  P1M.0       ; 设置 P1.0 为输出模式。
```

6.3 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	-	P06	P05	P04	P03	P02	P01	P00
读/写	-	R/W	R/W	R/W	R/W	R	R/W	R/W
复位后	-	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	-	-	P15	P14	P13	P12	P11	P10
读/写	-	-	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	0	0	0	0	0	0

✷ **注：当使能外部复位功能后，P02 保持为“1”。**

例：读取输入口的数据。

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

例：写入数据到输出端口。

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

例：写入 1 位数据到输出端口。

```
B0BSET  P1.0        ; P1.0 置“1”。

B0BCLR  P1.0        ; P1.0 置“0”。
```

7 定时器

7.1 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 RC 振荡器（16KHz @3V，32KHz @5V）提供。

看门狗溢出时间 = 8192 /内部低速振荡器周期（sec）

VDD	内部低速 RC Freq.	看门狗溢出时间
3V	16KHz	512ms
5V	32KHz	256ms

看门狗定时器的 3 种工作模式由编译选项“WatchDog”控制：

- **Disable**：禁止看门狗定时器功能。
- **Enable**：使能看门狗定时器功能，在普通模式和低速模式下有效，在睡眠模式和绿色模式下看门狗停止工作。
- **Always_On**：使能看门狗定时器功能，在睡眠模式和绿色模式下，看门狗仍会正常工作。

在高干扰环境下，强烈建议将看门狗设置为“Always_On”以确保系统在出错状态和重启时正常复位。

看门狗清零的方法是对看门狗计数器清零寄存器 WDTR 写入清零控制字 5AH。

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。**

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

➢ **例：用宏指令@RST_WDT 清看门狗定时器。**

```
Main:
        @RST_WDT                ; 清看门狗定时器。
        …
        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；
- 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。**

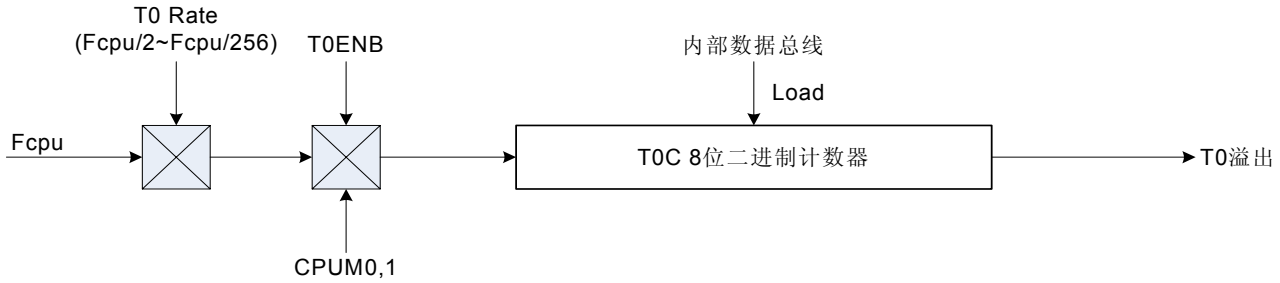
```
main:
        …                       ; 检测 I/O 口的状态。
        …                       ; 检测 RAM 的内容。
Err:    JMP $                   ; I/O 或 RAM 出错，不清看门狗等看门狗计时溢出。

Correct:                        ; I/O 和 RAM 正常，看门狗清零 。
                                ;
        MOV     A, #5AH         ; 在整个程序中只有一处地方清看门狗。
        B0MOV   WDTR, A
        …
        CALL    SUB1
        CALL    SUB2
        …
        JMP     MAIN
```

7.2 定时器 T0

7.2.1 概述

8 位二进制基本定时器 T0 无中断功能。T0 溢出时，则将系统唤醒返回到上一个工作模式。

☞ **8 位可编程计数定时器：**根据选择的时钟频率周期性的溢出。



7.2.2 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 分频选择位。
000 = fcpu/256；
001 = fcpu/128；
…；
110 = fcpu/4；
111 = fcpu/2。

Bit 7 **T0ENB**：T0 启动控制位。
0 = 禁止；
1 = 使能。

7.2.3 T0C 计数寄存器

8 位计数寄存器 T0C 用于控制 T0 的溢出间隔时间。

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 初始值的计算公式如下：

T0C 初始值 = 256 -（T0 间隔时间 * 输入时钟）

➢ **例：T0 的间隔时间为 10ms，高速时钟为外部 4MHz，Fcpu = Fosc/4，T0RATE = 010（Fcpu/64）。**

T0C 初始值 = 256 -（T0 溢出间隔时间 * 输入时钟）
= 256 -（10ms * 4MHz / 1 / 64）
= 256 -（10^{-2} * 4 * 10^{6} / 1 / 64）
= 100
= 64H

T0 间隔时间列表

T0RATE	T0CLOCK	高速模式（Fcpu = 4MHz / 4）	
		最大间隔时间	单步时间 = max/256
000	Fcpu/256	65.536 ms	256 us
001	Fcpu/128	32.768 ms	128 us
010	Fcpu/64	16.384 ms	64 us
011	Fcpu/32	8.192 ms	32 us
100	Fcpu/16	4.096 ms	16 us
101	Fcpu/8	2.048 ms	8 us
110	Fcpu/4	1.024 ms	4 us
111	Fcpu/2	0.512 ms	2 us

7.2.4 T0 操作流程

T0 的操作流程如下：

☞ **T0 停止计数。**

```
B0BCLR      FT0IRQ          ; 清 T0IRQ。
```

☞ **设置 T0 速率。**

```
MOV         A, #0xxx0000b   ; T0M 的 bit4~bit6 将 T0 的速率控制在 x000xxxxb~x111xxxxb。
B0MOV       T0M,A           ; 关闭 T0 定时器。
```

☞ **设置 T0 的溢出间隔时间。**

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

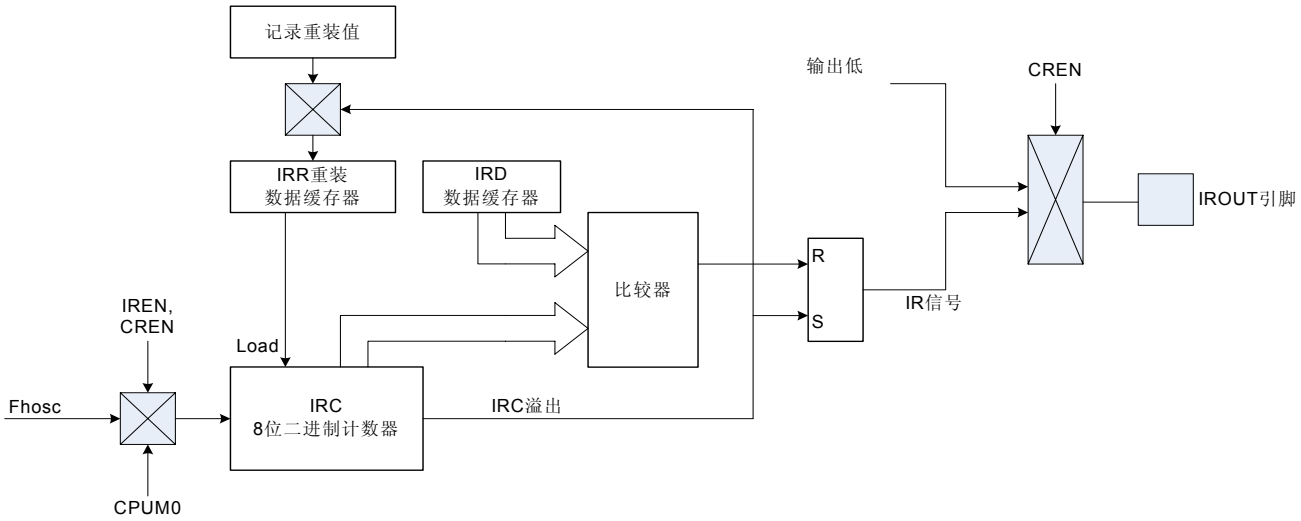
☞ **开启 T0 定时器。**

```
B0BSET      FT0ENB          ;
```

8 IR 输出

8.1 概述

红外信号由 IR 定时器产生。当 IREN 位为“1”时，IROUT 引脚输出 IR 信号。当系统处于睡眠模式时，IROUT 输出高电平。IR 定时器是 8 位二进制定时器。由寄存器 IRR 和 IRD 来控制 IR 信号的周期和占空比。IRR 控制红外信号的周期，IRD 控制红外信号的占空比。IR 时钟源来自外部高速时钟源（Fhosc），如 4MHz 振荡器。当 Fhosc = 4MHz 时，IR 计数时钟速率就是 4MHz。IR 定时器只能产生红外输出信号，不支持中断功能。当 CREN = 1 时，IR 输出低电平，同时 IRC 开始计数，IRC 的初始值为 IRR。当 IRC = IRD，IR 输出从低电平转变为高电平输出；当 IRC 溢出（0FFH 到 00H）时，IROUT 引脚输出从高电平转为低电平。同时系统自动将 IRR 中的值装载到 IRC 寄存器中并进入下一个循环周期。



✷ **注：系统进入睡眠模式之前，必须禁止 IR 载波输出（CREN=0）以避免 IR 载波信号出错。睡眠模式下，系统自动强制 IROUT 输出高电平。**

8.2 IRM 模式寄存器

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRM	-	-	-	-	-	-	-	CREN
读/写	-	-	-	-	-	-	-	R/W
复位后	-	-	-	-	-	-	-	0

Bit 0 **CREN**：IR 载波输出控制位。
0 =禁止，IROUT 输出高电平；
1 =使能，IROUT 输出 IR 载波信号。

8.3 IRC 计数寄存器

8 位计数寄存器 IRC 控制 IR 的间隔时间。

0DBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRC	IRC7	IRC6	IRC5	IRC4	IRC3	IRC2	IRC1	IRC0
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

✷ **注：在使能 IR 信号输出功能之前设置 IRC=IRR，以保证第一次循环是正确的。**

8.4 IRR 自动装载寄存器

寄存器 IRR 控制红外信号的频率。IR 定时器具有自动装载功能，当 IRC 溢出，IRR 的值会自动装载到 IRC，为 IR 信号的周期提供了精确的计时标准。

IR 为双重缓存器设计，当程序设置一个新的 IRR 值时，其新值就保存在第一个缓存器中，直到 IR 溢出，其新值就移动到真正的 IRR 缓存器中。

0CDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRR	IRR7	IRR6	IRR5	IRR4	IRR3	IRR2	IRR1	IRR0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

IRR 初始值的计算如下：

IRR 初始值 = 256 –（IR 间隔时间 * 输入时钟）

✷ **注：输入时钟来自于外部晶振 4MHz。**

➢ **例：设置 IR 的频率是 38KHz，输入时钟 = 4MHz。**

IRR 初始值= 256 -（IR 间隔时间*输入时钟）

IR 间隔时间= 1/38KHz = 26.3us

输入时钟=外部晶振 4MHz

IRR = 256 - (26.3us * 4MHz)
= 150.8
≈ 151
=97H

8.5 IRD IR 占空比控制寄存器

寄存器 IRD 可以控制红外信号的占空比和高电平脉冲的宽度。当 IRC = IRD 时，红外信号由低电平脉冲变成高电平脉冲。当 IRC 溢出，高电平脉冲停止，低电平脉冲的宽度是 IRD ~ IRR，高电平脉冲的宽度是 256 ~ IRD。这样通过 IRR 和 IRD 寄存器就可以调整红外信号的周期和占空比（duty / cycle）。

0E8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRD	IRD7	IRD6	IRD5	IRD4	IRD3	IRD2	IRD1	IRD0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

IRD 初始值的计算如下：

IRD 初始值 **= IRR + (256-IRR) / (1/**IR 占空比**)**

- **例：IR 的频率是 38KHz，占空比是 1/3，输入时钟 = 4MHz。**

IRD 初始值= IRR + (256-IRR) / (1/IR 占空比)

IRR（IR 的频率是 38KHz） = 151

IRD = 151 + (256-151)/(1/ (1/3))

= 186

= BAH

IR 信号的基本列表，系统时钟为 4MHz。

IR Freq. (KHz)	IRC IRD		IRD						Freq. Error Rate
			1/2 占空比		1/3 占空比		1/4 占空比		
	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	
32	131	83	193.50	C1	172.67	AC	162.25	A2	0.00%
36	145	91	200.50	C8	182.00	B6	172.75	AC	0.10%
38	151	97	203.50	CB	186.00	BA	177.25	B1	0.25%
39.2	154	9A	205.00	CD	188.00	BC	179.50	B3	0.04%
40	156	9C	206.00	CE	189.33	BD	181.00	B5	0.00%
56	185	B9	220.50	DC	208.67	D0	202.75	CA	0.60%

8.6 IR 输出操作流程

- **设置 IRC 和 IRR，得到 IR 的周期。**

```
MOV     A, #IRCYCVAL    ; 设置 IRC 和 IRR 的值，得到 IR 的周期。
MOV     IRC, A
MOV     IRR, A
```

- **设置 IRC，得到 IR 的占空比。**

```
MOV     A, #IRDUTYVAL   ; 设置 IRD 的值，得到 IR 的占空比。
MOV     IRD, A
```

- **使能 IR 输出。**

```
BSET    FCREN           ; 设置 IR 载波信号输出。
```

9 指令集

Field	指令格式		描述	C	DC	Z	周期
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 仅适用地址是 0x80~0x87 的系统寄存器，如 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，如果产生进位则 C = 1，否则 C = 0。	√	√	√	1
	ADC	M,A	M ← A + M + C，如果产生进位则 C = 1，否则 C = 0。	√	√	√	1+N
	ADD	A,M	A ← A + M，如果产生进位则 C = 1，否则 C = 0。	√	√	√	1
	ADD	M,A	M ← A + M，如果产生进位则 C = 1，否则 C = 0。	√	√	√	1+N
	B0ADD	M,A	M (bank 0) ← M (bank 0) + A，如果产生进位则 C = 1，否则 C = 0。	√	√	√	1+N
	ADD	A,I	A ← A + I，如果产生进位则 C = 1，否则 C = 0。	√	√	√	1
	SBC	A,M	A ← A - M - /C，如果产生借位则 C=0，否则 C=1。	√	√	√	1
	SBC	M,A	M ← A - M - /C，如果产生借位则 C=0，否则 C=1。	√	√	√	1+N
	SUB	A,M	A ← A – M，如果产生借位则 C=0，否则 C=1。	√	√	√	1
	SUB	M,A	M ← A – M，如果产生借位则 C=0，否则 C=1。	√	√	√	1+N
	SUB	A,I	A ← A – I，如果产生借位则 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 ← M 带进位右移。	√	-	-	1
	RRCM	M	M ← M 带进位右移。	√	-	-	1+N
	RLC	M	A ← M 带进位左移。	√	-	-	1
	RLCM	M	M ← M 带进位左移。	√	-	-	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	比较，如果相等则跳过下一条指令，C 与 ZF 标志位可能受影响。	√	-	√	1 + S
	CMPRS	A,M	比较，如果相等则跳过下一条指令，C 与 ZF 标志位可能受影响。	√	-	√	1 + S
	INCS	M	A ← M + 1，如果 A = 0，则跳过下一条指令。	-	-	-	1+ S
	INCMS	M	M ← M + 1，如果 M = 0，则跳过下一条指令。	-	-	-	1+N+S
	DECS	M	A ← M - 1，如果 A = 0，则跳过下一条指令。	-	-	-	1+ S
	DECMS	M	M ← M - 1，如果 M = 0，则跳过下一条指令。	-	-	-	1+N+S
	BTS0	M.b	如果 M.b = 0，则跳过下一条指令。	-	-	-	1 + S
	BTS1	M.b	如果 M.b = 1，则跳过下一条指令。	-	-	-	1 + S
	B0BTS0	M.b	如果 M(bank 0).b = 0，则跳过下一条指令。	-	-	-	1 + S
	B0BTS1	M.b	如果 M(bank 0).b = 1，则跳过下一条指令。	-	-	-	1 + S
	JMP	d	跳转指令，PC15/14 ← RomPages1/0，PC13~PC0 ← d。	-	-	-	2
	CALL	d	子程序调用指令，Stack ← PC15~PC0，PC15/14 ← RomPages1/0，PC13~PC0 ← d。	-	-	-	2
MISC	RET		子程序跳出指令，PC ← Stack。	-	-	-	2
	NOP		空指令，无特别意义。	-	-	-	1

注： **1. “M”是系统寄存器或 RAM，M 为系统寄存器时 N = 0，否则 N = 1。**
2. 条件跳转指令的条件为真，则 S = 1，否则 S = 0。

10 电气特性

10.1 极限参数

Supply voltage (Vdd)..... - 0.3V ~ 6.0V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8PC21P, SN8PC21S,..... 0°C ~ + 70°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

10.2 电气特性

- **DC CHARACTERISTIC**

(All of voltages refer to Vss, Vdd = 3.0V, fosc = 4MHz,fcpu=1MHZ,ambient temperature is 25°C unless otherwise note.)

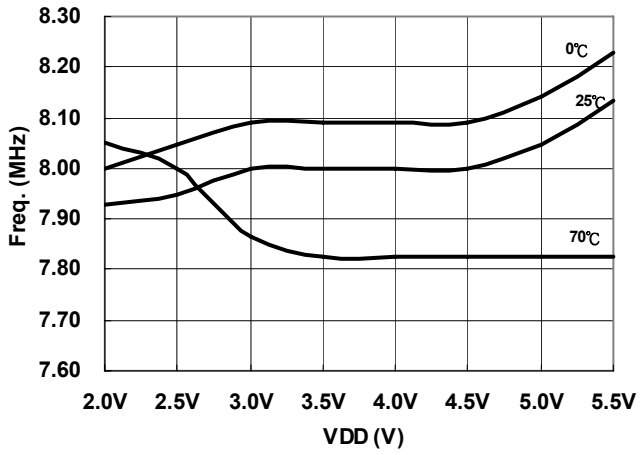
PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd, 25°C, Fcpu = 2mips.		2.0	-	5.5	V
RAM Data Retention voltage	Vdr			1.5	-	-	V
*Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL1	All input ports		Vss	-	0.3Vdd	V
	ViL2	Reset pin		Vss	-	0.2Vdd	V
Input High Voltage	ViH1	All input ports		0.7Vdd	-	Vdd	V
	ViH2	Reset pin		0.9Vdd	-	Vdd	V
Reset pin leakage current	Ilekg	Vin = Vdd		-	-	2	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 3V		100	200	300	KΩ
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O output source current sink current	IoH	Vop = Vdd – 0.5V		8	10	-	mA
	IoL1	Vop = Vss + 0.5V		8	12	-	
	IoL2	Vop = Vss + 1.5V, IR output pin		300	400	-	mA
Supply Current	Idd1	Run Mode, IHRC 8MHz.	Vdd= 3V Vdd= 5V	-	0.5 1	1 2	mA
	Idd2	Sleep Mode	Vdd= 3V Vdd= 5V	-	1	2	uA
Internal High Oscillator Freq.	Fihrc	Internal Hihg RC (IHRC)	25°C,Vdd= 2V~5V	7.84	8	8.16	Mhz
*LVD Voltage	Vdet0	Low voltage reset level.		-	-	1.8	V

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

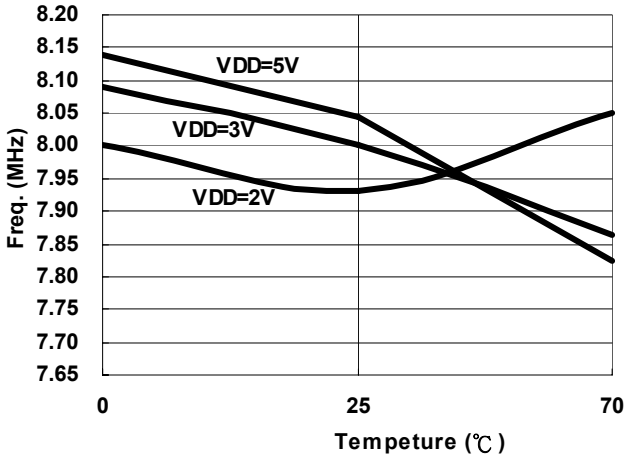
10.3 特性曲线图

本章所列的各曲线图仅作设计参考，其中给出的部分数据可能超出了芯片指定的工作范围，为保证芯片的正常工作，请严格参照电气特性说明。

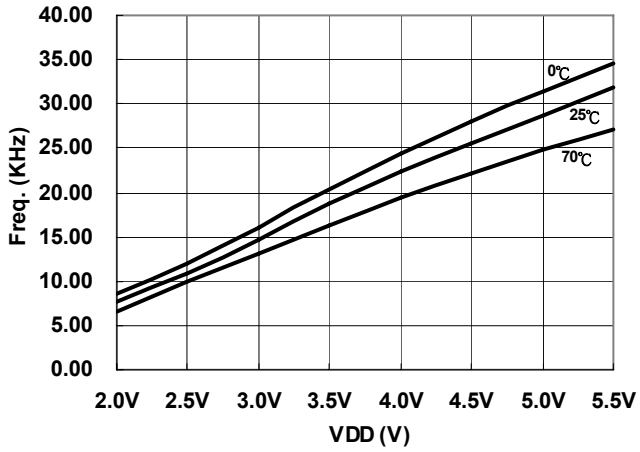
Internal High RC Oscillator (MHz)



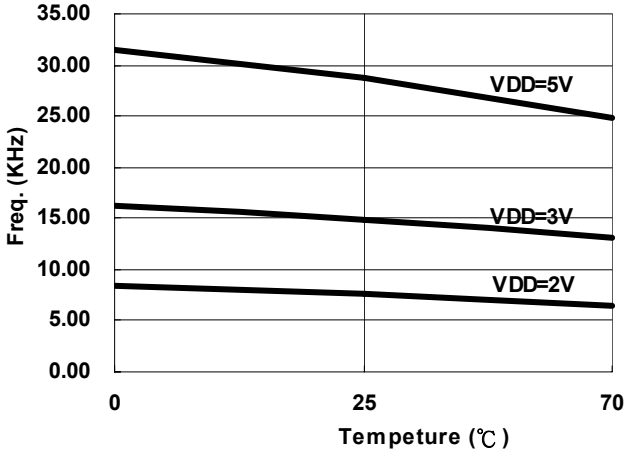
Internal High RC Oscillator (MHz)
(Fcpu=IHRC/16)



Internal Low RC Oscillator (KHz)



Internal Low RC Oscillator (KHz)



11 开发工具

在进行 SN8PC31 的开发时，SONIX 提供 ICE（在线仿真器），IDE（集成开发环境）和 EV-Kit 开发工具。ICE 和 EV-Kit 为外部硬件装置，IDE 有一个友好的用户界面进行韧件开发与仿真。各工具的版本如下所示：

- **ICE：SN8ICE2K Plus 2。**
- **EV-kit：SN8PC21 EV-kit V1.0。**
- **IDE：SONiX IDE M2IDE_V126 或更晚的版本。**
- **Writer：MPIII writer。**

11.1 ICE 和 EV-KIT 应用注意事项

SN8PC21 EV-kit 包括矩阵按键和 IR 驱动电路模块。

IR 驱动电路模块包括两种：一种是用 ICE 仿真来驱动 IR 载波信号；另一种是直接使用 SN8PC21 实际芯片，而不连接 ICE。R3 的阻值最小为 3.75Ω，Vdd=3V 时，sink 电流为 400mA。

如果 SN8PC21 实际芯片设置晶振模式，则 C4（XIN）和 C5（XOUT）的容值为 20pF。

12 OTP 烧录引脚

12.1 烧录转接板引脚配置

Easy Writer JP1/JP2

VSS	2	1	VDD
CE	4	3	CLK/PGCLK
OE/ShiftDat	6	5	PGM/OTPCLK
D0	8	7	D1
D2	10	9	D3
D4	12	11	D5
D6	14	13	D7
VPP	16	15	VDD
RST	18	17	HLS
ALSB/PDB	20	19	-

JP1 连接 MP 烧录转接板

Easy Writer JP3 (Mapping to 48-pin text tool)

DIP1	1	48	DIP48
DIP2	2	47	DIP47
DIP3	3	46	DIP46
DIP4	4	45	DIP45
DIP5	5	44	DIP44
DIP6	6	43	DIP43
DIP7	7	42	DIP42
DIP8	8	41	DIP41
DIP9	9	40	DIP40
DIP10	10	39	DIP39
DIP11	11	38	DIP38
DIP12	12	37	DIP38
DIP13	13	36	DIP36
DIP14	14	35	DIP35
DIP15	15	34	DIP34
DIP16	16	33	DIP33
DIP17	17	32	DIP32
DIP18	18	31	DIP31
DIP19	19	30	DIP30
DIP20	20	29	DIP29
DIP21	21	28	DIP28
DIP22	22	27	DIP27
DIP23	23	26	DIP26
DIP24	24	25	DIP25

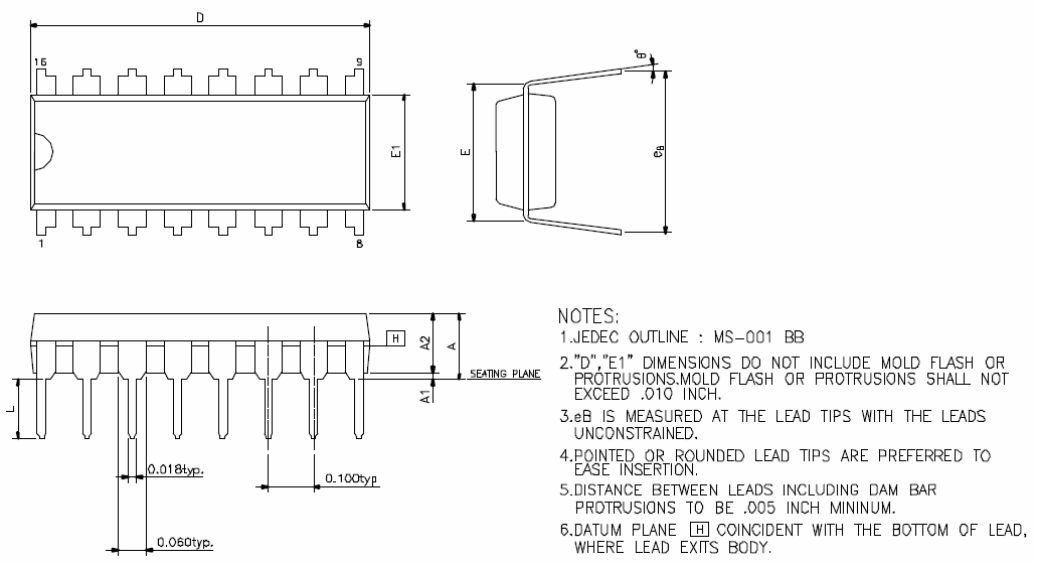
JP3 连接 MP 烧录转接板

12.2 烧录引脚配置

SN8PC21 烧录引脚信息			
单片机名称		SN8PC21P/S	
Writer 接口		OTP IC / JP3 引脚配置	
引脚编号	引脚名称	引脚编号	引脚名称
1	VDD	16	VDD
2	GND	15	VSS
3	CLK	10	P1.2
4	CE	-	-
5	PGM	8	P1.0
6	OE	11	P1.3
7	D1	-	-
8	D0	-	-
9	D3	-	-
10	D2	-	-
11	D5	-	-
12	D4	-	-
13	D7	-	-
14	D6	-	-
15	VDD	-	-
16	VPP	3	RST
17	HLS	-	-
18	RST	-	-
19	-	-	-
20	ALSB/PDB	9	P1.1

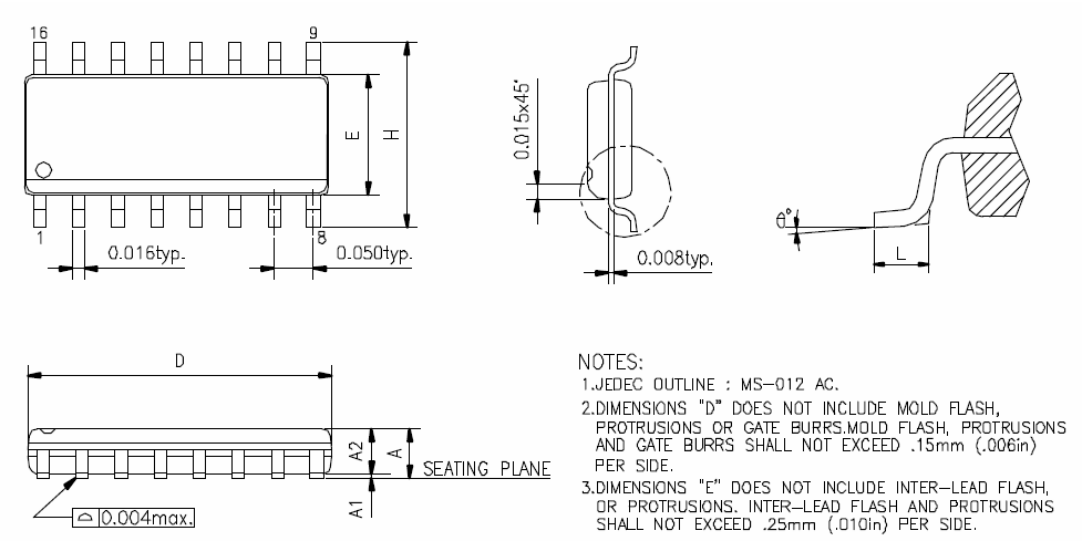
13 封裝信息

13.1 DIP16 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-		0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.735	0.755	0.775	18.669	19.177	19.685
E		0.30 BSC			7.620 BSC	
E1	0.245	0.250	0.255	6.223	6.350	6.477
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.335	0.355	0.375	8.509	9.017	9.525
θ°	0°	7°	15°	0°	7°	15°

13.2 SOP 16 PIN



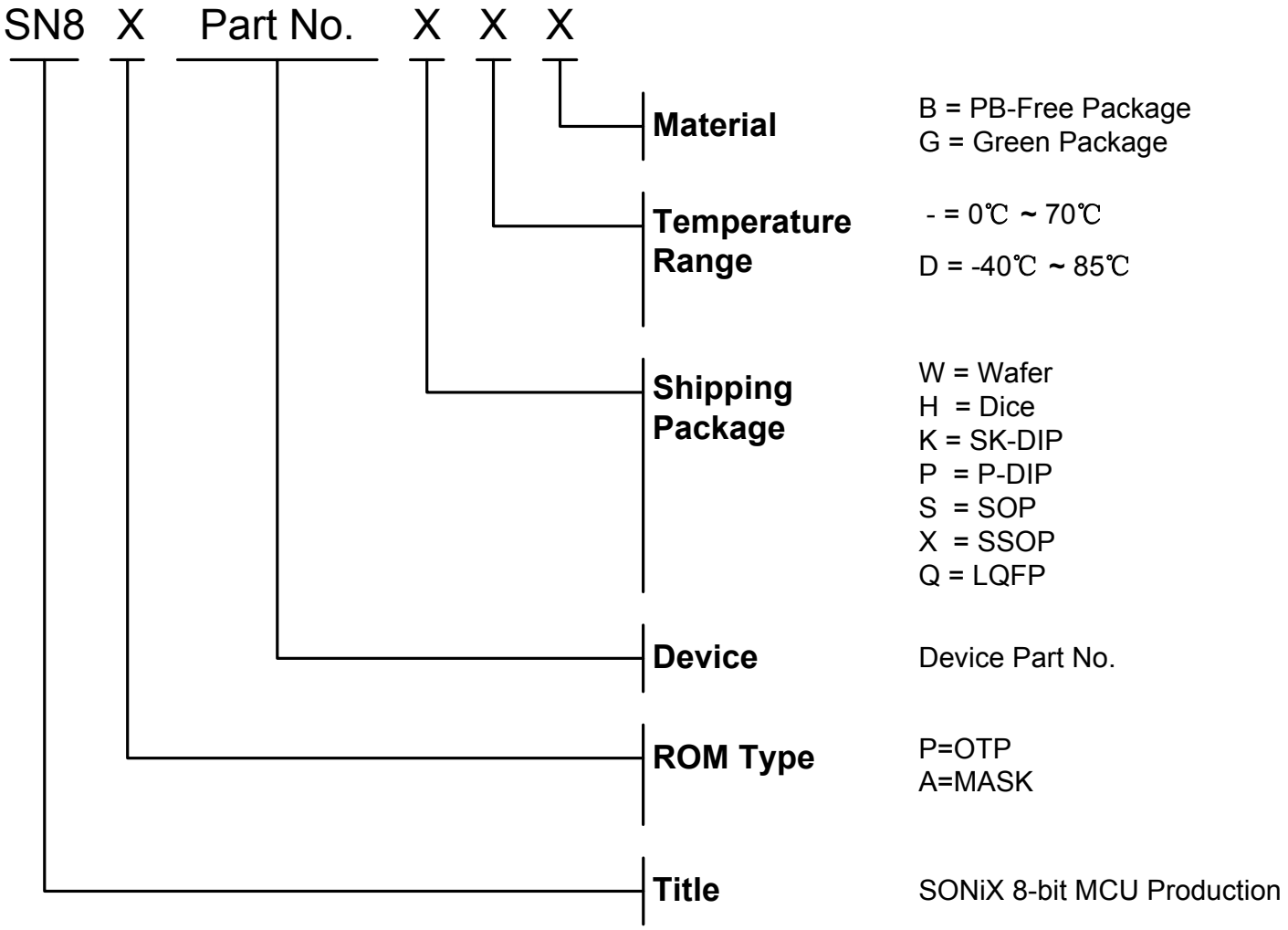
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.053	-	0.069	1.346	-	1.753
A1	0.004	-	0.010	0.102	-	0.254
D	0.386	-	0.394	9.804	-	10.008
E	0.150	-	0.157	3.810	-	3.988
H	0.228	-	0.244	5.791	-	6.198
L	0.016	-	0.050	0.406	-	1.270
θ°	0°	-	8°	0°	-	8°

14 单片机正印命名规则

14.1 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，适用于空片 OTP 型单片机。

14.2 单片机正印说明

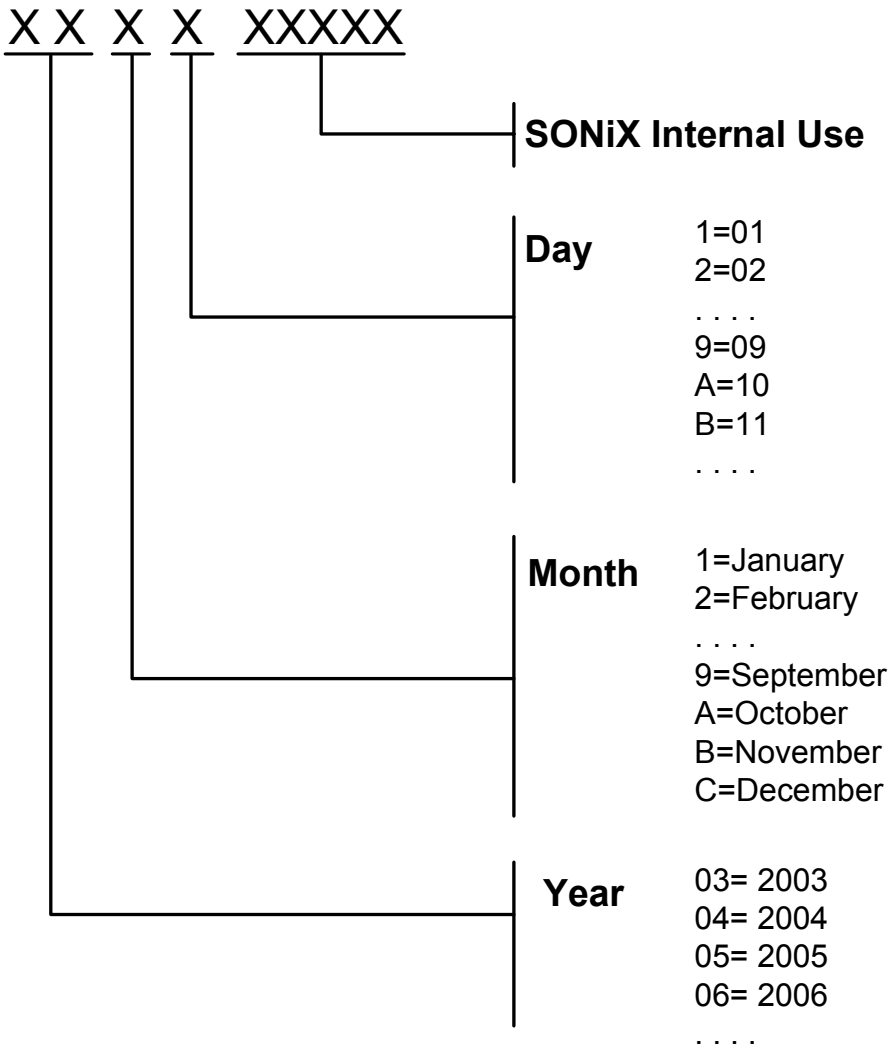


✷ 注：SN8PC21 正常的工作范围不是-40ºC~85ºC。

14.3 命名举例

单片机名称	ROM 类型	设备（Device）	封装形式	温度范围	封装材料
SN8PC21PG	OTP	C21	P-DIP	0℃~70℃	绿色封装
SN8PC21SG	OTP	C21	SOP	0℃~70℃	绿色封装
SN8PC21W	OTP	C21	Wafer	0℃~70℃	-
SN8PC21H	OTP	C21	Dice	0℃~70℃	-

14.4 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