Sonix（松翰）8bit 单片机
26 系列 I/O 型原理及基础课件（二）
基础硬件功能模块

主要内容：

为了让更多的工程师更加方便、快捷的了解和使用 SONIX 单片机，从而我们编写 SONIX 单片机系列的培训课件，主要详细的介绍了 SONIX 26 系列芯片的硬件模块、指令以及开发环境，供业界工程师交流经验，由于时间仓促，请提出宝贵的意见。

SONIX **官方网站：**http:www.sonix.com.tw
Sonix **单片机咨询邮箱：**howard_tone@hotmail.com
Sonix **单片机咨询电话：**13725134515

1．1．I/O端口概述：

26XX系列单片机最多有4组8位并行I/O端口：端口P0、端口P1、端口P2、端口P5；

26系列单片机的每一位I/O端口都可单独被定义为输入或输出端口，输入时可设置为有上拉；

各端口均配有数据缓存寄存器（锁存器），端口可进行直接的位操作；

特殊功能：各端口与内部硬件功能模块复用；

*P5.4端口可做PWM OR TCOUT输出

*P0。0端口可做外部中断/计数输入

*RST脚可做端口（P1.4）使用，只能作为输入口，内部无上拉电阻；

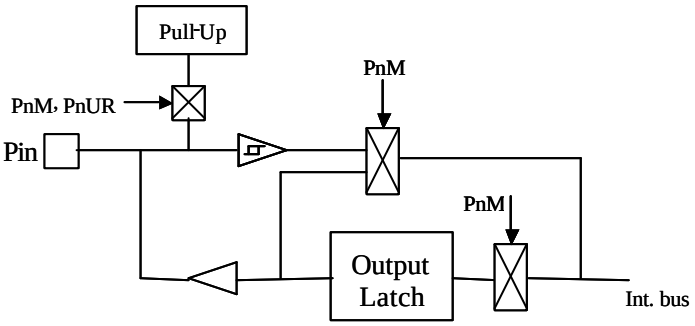
I/O端口基的基本描述/电气参数：

（所有电压以 Vss, Vdd=5.0V 为参考值， fosc=3.579545MHz，环境温度为 25℃）

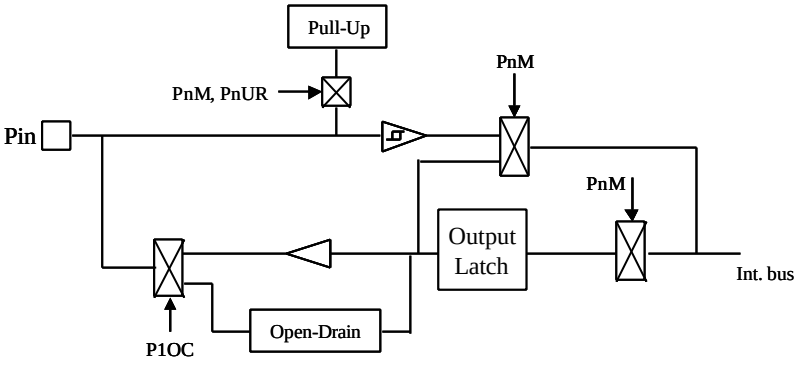
参数	符号	说明	最小指	标准值	最大值	单位
输入低电压	ViL	施密特触发的输入端	Vss	-	0.2Vdd	V
输入高电压	ViH	施密特触发的输入端	0.8Vdd	-	Vdd	V
复位引脚漏电流	Ilekg	Vin = Vdd	-	-	2	uA
I/O 口的上拉电阻	Rup	Vin = Vss, Vdd = 3v	-	200	-	KΩ
		Vin = Vss , Vdd = 5V	-	100	-	
I/O 口的输入漏电流	Ilekg	禁止上拉电阻, Vin = Vdd	-	-	2	uA
输出所有端口的源电流	IoH	Vop = Vdd – 0.5V	-	12	-	mA
输出所有端口的灌电流	IoL	Vop = Vss + 0.5V	-	15	-	
INTn 触发脉冲宽度	Tint0	INT0 ~ INT2 中断请求脉冲宽度	2/fcpu	-	-	cycle

1．2．I/O 端口结构：

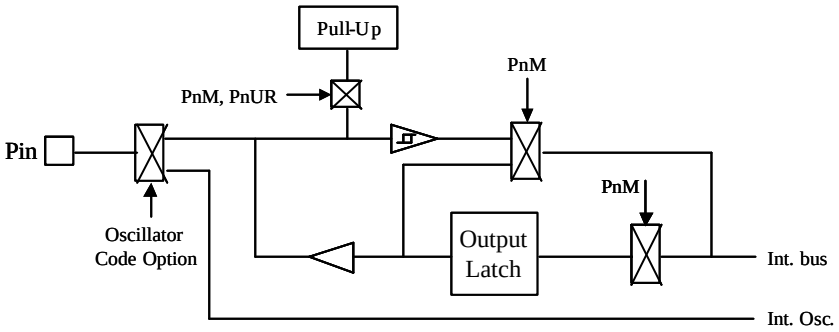
Port 0, 2, 5结构



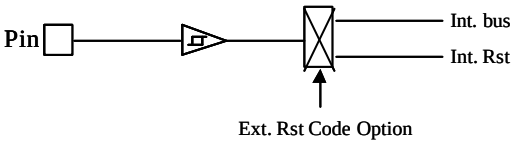
Port 1.0结构



Port 1.2和 Port 1.3结构



Port 1.1结构



I/O端口的设置：

通常，对某一位的设定包括以下3个基本项：数据寄存器Data、上拉寄存器PUXR和方向控制寄存器PXM，唤醒寄存器P1W（注P0口默认为唤醒）；

➢其具体作用如下：

➢方向控制寄存器将管脚设置为输入或输出

➢上拉寄存器PUXR将管脚设置为上拉或不上拉

➢唤醒寄存器P1W将管脚设置为唤醒或不唤醒

➢当管脚作为输入时，上拉寄存器PUXR将其设置为上拉，读数据寄存器Data；

➢当管脚作为输出时，往数据寄存器Data里面写入数值便可以将其输出；

方向控制寄存器8位PnM[7:0](n=0,1,2,5)1为输出，0为输入；

例：输入/输出模式选择

```
CLR         P0M             ; 设置为输入模式
CLR         P1M
CLR         P2M
CLR         P5M

MOV         A, #0FFH        ; 设置为输出模式
B0MOV       P0M, A
B0MOV       P1M, A
B0MOV       P2M, A
B0MOV       P5M, A

B0BCLR      P1M.2           ; 设置 P1.2 为输入模式

B0BSET      P1M.2           ; 设置 P1.2 为输出模式
```

上拉寄存器PunR[7:0] (n=0,1,2,5)1为为将管脚设置上拉，0为将管脚设置为不上拉;
注：RST脚可做端口（P1.4）使用，只能作为输入口，内部无上拉电阻

➲ **例：I/O 上拉寄存器**

```
MOV         A, #0FFH        ; 使能 P0,P1,P2,P5 口上拉
B0MOV       P0UR, A         ;
B0MOV       P1UR, A
B0MOV       P2UR, A
B0MOV       P5UR, A
```

唤醒寄存器P1W[7：0] 1为为将管脚势能唤醒，0为将管脚禁止唤醒；

省电（睡眠）模式下，具有唤醒功能的P0和P1都能将系统唤醒，P0永远具有唤醒功能，而P1的唤醒功能受寄存器P1W控制。改变其电平就可产生唤醒信号

I/O口数据寄存器（P0，P1，P2，P5）：

例：从输入端读取数据

```
B0MOV       A, P0           ; 读 P0 口的数据
B0MOV       A, P1           ; 读 P1 口的数据
B0MOV       A, P2           ; 读 P2 口的数据
B0MOV       A, P5           ; 读 P5 口的数据
```

➲ **例：写数据到输出端**

```
MOV         A, #0FFH        ; 所有的端口写 FFH
B0MOV       P0, A
B0MOV       P1, A
```

```
B0MOV       P2, A
B0MOV       P5, A
```

➲ **例：写 1-bit 的数据到输出端口**

```
B0BSET      P1.3        ; 置 P1.3 和 P2.5 为“1”.
B0BSET      P2.5

B0BCLR      P1.3        ; 置 P1.3 和 P2.5 为“0”.
B0BCLR      P2.5
```

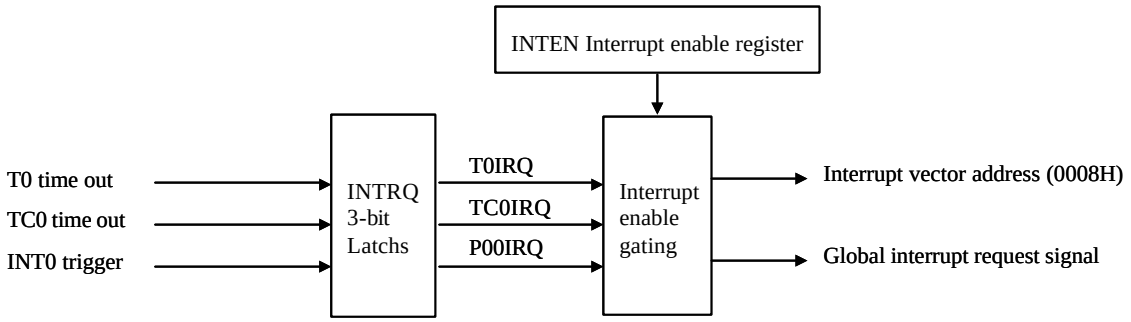
2．中断系统：

***中断系统的中断源T0，TC0，INT0：**

***与中断系统的寄存器：**INTEN**中断使能寄存器，**INTRQ**中断请求寄存器，**INT0(P0.0)**中断操作寄存器**PEDG**；**

2．1**概述：**

26XX **系列有** 3 **个中断源：两个内部中断源（**T0/TC0**），一个外部中断源（**INT0**）。外部中断** INT0 **能够唤醒睡眠模式进入高速模式。当系统进入到中断服务程序时，全局中断控制位** GIE **清零，系统退出中断服务后，**GIE **被置为**“1”**以准备响应下一个中断请求。所有的中断请求存放于** INTRQ **中，用户可编程设置中断优先级。**



✷ **注：GIE 使能后，才能响应各中断。**

26系列单片机中断服务一般流程：

1. **保护现场（包括寄存器）**
2. **中断源判断**
3. **中断服务程序处理**
4. **清除中断标志位**
5. **恢复现场**
6. **返回**

1.1 INTEN 中断使能寄存器

INTEN **为中断使能控制寄存器，包括内部中断和外部中断的控制位。**INTEN **的某位被置**“1”， **则相对应的中断请求便能够被响应。一旦有中断发生，程序将跳至** ORG 8 **处执行中断服务程序。当执行到中断服务返回指令（**RETI**）时，将退出中断程序。**

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	-	-	TC0IEN	T0IEN	-	-	-	P00IEN
读/写	-	-	R/W	R/W	-	-	-	R/W
复位后	-	-	0	0	-	-	-	0

P00IEN：**外部** P0.0 **中断控制位，**0 = **禁止，**1 = **使能**
T0IEN：**定时器** T0 **中断控制位，**0 = **禁止，**1 = **使能**
TC0IEN：**定时器** TC0 **中断控制位，**0 = **禁止，** 1 =**使能**

1.2 INTRQ 中断请求寄存器

INTRQ 为中断请求寄存器，包含了所有的中断请求标志，当有中断发生时，INTRQ 寄存器中的相应位会置为“1”。中断请求标志需要用软件清零。用户通过检查中断请求寄存器可以知道中断的种类，从而执行相应的中断服务程序。

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ	-	-	TC0IRQ	T0IRQ	-	-	-	P00IRQ
读/写	-	-	R/W	R/W	-	-	-	R/W
复位后	-	-	0	0	-	-	-	0

P00IRQ：外部 P0.0 中断请求位，0 = 无中断请求，1 = 请求中断服务
T0IRQ：T0 定时器中断请求位，0 = 无中断请求，1 = 请求中断服务
TC0IRQ：TC0 定时器中断请求位，0 = 无中断请求，1 = 请求中断服务

1.3 中断操作举例

1.3.1 GIE 全局中断操作

GIE 是总中断控制位。所有的中断在 GIE 使能的前提下才能够得到响应。一旦有中断请求发生，程序计数器 PC 指向中断向量地址（ORG 8），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

GIE：全局中断控制位。0=禁止，1=使能

➲ **例：设置全局中断控制位（GIE）**

```
B0BSET      FGIE            总中断使能
```

✷ **注：GIE 必须使能，所有中断才能被响应。**

1.3.2 PUSH, POP 程序

中断发生时，系统跳到 ORG 8 处执行中断服务程序，此时必须保存 ACC、PFLAG 的值。当进入中断服务程序后，芯片没有任何指定的指令处理 ACC、PFLAG，用户必须由程序保存 ACC 和 PFLAG。在中断服务程序完成后，用“B0XCH”保存/恢复 ACC，“B0MOV”保存/恢复 PFLAG 可以避免主程序出错。

➲ **例：在中断服务程序发生时由程序保存 ACC 和 PFLAG 的值。**

```
ACCBUF      EQU     00H         ; 定义 ACCBUF 保存 ACC 的值
PFLAGBUF    EQU     01H         ; 定义 PFLAGBUF 保存 PFLAG 的值

            ORG     0
            JMP     START

            ORG     8
            JMP     INT_SERVICE

            ORG     10H
START:
            …
INT_SERVICE:
            B0XCH   A, ACCBUF   ; 保存 ACC
            B0MOV   A, PFLAG
            BOMOV   PFLAGVUF, A ; 保存 PFLAG
            .
            .
```

```
B0MOV       A, PFLAGBUF     ; 恢复 PFLAG
B0MOV       PFLAG, A
B0XCH       A, ACCBUF       ; 恢复 ACC
RETI                        ; 中断返回
…
ENDP
```

✷ **注：为了保存和恢复 ACC，必须使用“ B0XCH”指令，否则会影响 PFLAG。**

1.3.3 INT0(P0.0)中断操作

当 INT0 **中断发生时，不管** P00IEN **是否使能，**P00IRQ **都会置**"1"**。若** P00IEN=1**，且** P00IRQ=1**，那么系统就进入中断向量地址（**ORG 8**）执行中断服务程序。但若** P00IEN=0**，不管** P00IRQ **是否等于** 1**，系统都不进入中断。用户应注意多种中断下的处理。**

✷ **注：中断触发的方式由 PEDGE 寄存器控制。**

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

P00G[1:0]: P0.0 **中断触发边沿控制位：**00=**保留，**01=**上升沿，**10=**下降沿，**11=**上升/下降沿双向（改变电平）**

➲ **例：INT0 中断请求**

```
                MOV         A, #18H
                B0MOV       PEDGE, A        ; 设置 INT0 中断触发为双向触发

                B0BSET      FP00IEN         ; INT0 中断使能
                B0BCLR      FP00IRQ         ; 清 INT0 中断请求标志
                B0BSET      FGIE            ; 总中断使能
```

➲ **例：INT0 中断服务程序**

```
ACCBUF          DS          1               ; 定义 ACCBUF 保存 ACC
PFLAGBUF        DS          1               ; 定义 PFLAGBUF 保存 PFLAG

                ORG         8               ; 中断向量地址
                JMP         INT_SERVICE
INT_SERVICE:
                B0XCH       A, ACCBUF       ; 保存 ACC 和 PFLAG
                B0MOV       A, PFLAG
                B0MOV       PFLAGBUF, A

                B0BTS1      FP00IRQ         ; 判断是否有外部中断请求
                JMP         EXIT_INT        ; P00IRQ=1，退出中断

                B0BCLR      FP00IRQ         ; 清中断标志
                .           .               ; 中断服务程序
EXIT_INT:
                B0MOV       A, PFLAGBUF     ; 恢复 ACC 和 PFLAG
                B0MOV       PFLAG, A
                B0XCH       A, ACCBUF       ;

                RETI                        ;中断返回
```

1.3.4 T0 中断操作

当 T0C 计数器溢出，无论 T0IEN 是否使能，T0IRQ 都会置“1”。若 T0IEN=1，且 T0IRQ =1，那么系统就进入中断向量地址（ORG 8）执行中断服务程序。若 T0IEN =0，不管 T0IRQ 是否等于 1，系统都不进入中断。用户应注意多种中断下的处理。

➲ **例：初始化 T0**

```
        B0BCLR      FT0IEN          ; 禁止 T0 中断
        B0BCLR      FT0ENB          ; 停止 T0 计数
        MOV         A, #20H         ;
        B0MOV       T0M, A          ; 设置定时模式 Fcpu / 64
        MOV         A, #74H         ; 设置时间常数
        B0MOV       T0C, A          ; 定时中断为 10 ms

        B0BCLR      FT0IRQ          ; 清 T0 中断请求标志
        B0BSET      FT0IEN          ; 使能 T0 中断
        B0BSET      FT0ENB          ; 开始 T0 计数

        B0BSET      FGIE            ; 使能总中断
```

➢ **注：避免错误的响应中断，在设置定时/计数器中断时必须先清除定时/计数器的中断请求标志位 T0IRQ，再开放中断和定时/计数器，如上例。**

➲ **例：T0 中断服务程序**

```
ACCBUF      DS          1           ; 定义 ACCBUF 保存 ACC
PFLAGBUF    DS          1           ; 定义 PFLAGBUF 保存 PFLAG

            ORG         8           ; 中断向量地址
            JMP         INT_SERVICE
INT_SERVICE:

            B0XCH       A, ACCBUF   ; 保存 ACC 和 PFLAG
            B0MOV       A, PFLAG
            B0MOV       PFLAGBUF, A

            B0BTS1      FT0IRQ      ; 检查是否是 T0 中断请求
            JMP         EXIT_INT    ;

            B0BCLR      FT0IRQ      ; 清 T0 中断标志
            MOV         A, #74H
            B0MOV       T0C, A      ; 重新装载时间常数
            .           .           ; T0 中断服务程序
            .           .
EXIT_INT:
            B0MOV       A, PFLAGBUF ; 恢复 ACC 和 PFLAG
            B0MOV       PFLAG, A
            B0XCH       A, ACCBUF

            RETI                    ; 中断返回
```

1.3.5 TC0 中断操作

当计数器 TC0C 溢出时，无论 TC0IEN 是否使能，TC0IRQ 都会置“1”。若 TC0IEN =1，且 TC0IRQ =1，那么系统就进入中断向量地址（ORG 8）执行中断服务程序。若 TC0IEN =0，不管 TC0IRQ 是否等于 1，系统都不进入中断。用户应注意多种中断下的处理。

➲ **例：初始化 TC0**

```
        B0BCLR      FTC0IEN         ; 禁止 TC0 中断
        B0BCLR      FTC0ENB         ; 停止 TC0 计数
        MOV         A, #20H         ;
        B0MOV       TC0M, A         ; 设置 TC0 模式 Fcpu / 64
        MOV         A, #74H         ; 设置时间常数
        B0MOV       TC0C, A         ; TC0 每 10ms 中断一次

        B0BCLR      FTC0IRQ         ; 清 TC0 中断标志
        B0BSET      FTC0IEN         ; 使能 TC0 中断
        B0BSET      FTC0ENB         ; 开始 TC0 计数

        B0BSET      FGIE            ; 使能总中断
```

➢ **注：避免错误的响应中断，在设置定时/计数器中断时必须先清除定时/计数器的中断请求标志位 TC0IRQ，再开放中断和定时/计数器，如上例。**

➲ **例：TC0 中断服务程序**

```
ACCBUF      DS          1               ; 定义 ACCBUF 保存 ACC
PFLAGBUF    DS          1               ; 定义 PFLAGBUF 保存 PFLAG

            ORG         8               ;中断向量地址
            JMP         INT_SERVICE
INT_SERVICE:

            B0XCH       A, ACCBUF       ; 保存 ACC 和 PFLAG
            B0MOV       A, PFLAG
            B0MOV       PFLAGBUF, A

            B0BTS1      FTC0IRQ         ; 检查是否是 TC0 中断
            JMP         EXIT_INT        ; 若不是则中断返回

            B0BCLR      FTC0IRQ         ; 清 TC0 中断标志
            MOV         A, #74H
            B0MOV       TC0C, A         ; 恢复时间常数
            .           .               ; TC0 中断服务程序
            .           .
EXIT_INT:
            B0MOV       A, PFLAGBUF
            B0MOV       PFLAG, A
            B0XCH       A, ACCBUF       ; 恢复 ACC 和 PFLAG

            RETI                        ; 中断返回
```

1.3.6 多个中断操作

大部分情况下，用户需要同时处理多个中断。处理多个中断就需要设置中断的优先权。中断请求由不同的事件控制，但是，有中断请求并不意味着系统就会去执行中断服务程序。不管中断是否使能，都可触发中断请求，一旦有中断发生，相应的中断请求标志就会被置为"1"。各中断与对应的触发事件关系如下表所示：

中　断	触　发　信　号
P00IRQ	由 PEDGE 寄存器控制 P0.0 触发
T0IRQ	T0C 溢出
TC0IRQ	TC0C 溢出

在处理多中断请求下，用户必须对各中断进行优先权的设置，并根据 IEN 和 IRQ 的状态决定系统是否响应中断请求。用户必须在中断向量里检查中断控制位和中断请求标志位。

➲ 例：在多中断情况下，检查是否响应各中断请求

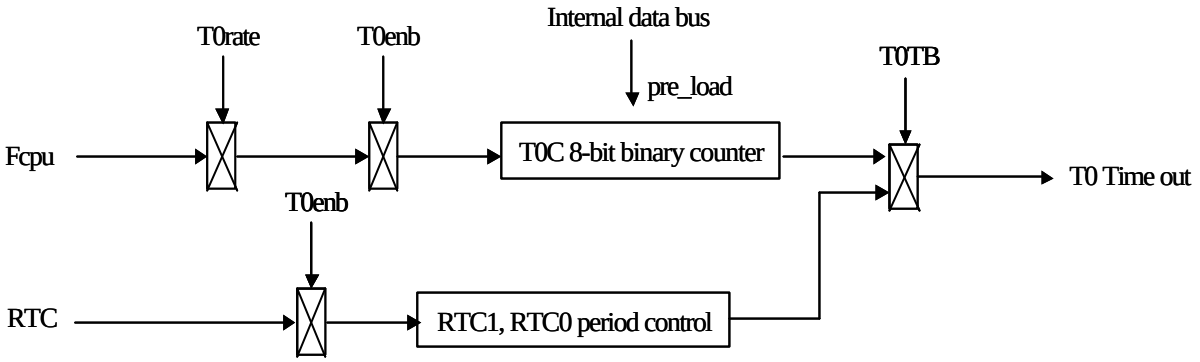
```
ACCBUF      DS      1               ; 定义 ACCBUF 保存 ACC
PFLAGBUF    DS      1               ; 定义 PFLAGBUF 保存 PFLAG

            ORG     8               ; 中断向量地址
            NOP                     ; ORG 8 处的第一条指令
            B0XCH   A, ACCBUF       ; 保存 ACC 和 PFLAG
            B0MOV   A, PFLAG
            B0MOV   PFLAGBUF,A

INTP00CHK:                          ; 检查是否有 INT0 中断
            B0BTS1  FP00IEN         ; 检查是否允外部中断 0
            JMP     INTTC0CHK       ; 跳转到下一个中断
            B0BTS0  FP00IRQ         ; 检测是否有外部中断 0 的请求
            JMP     INTP00          ; 跳转到 INT0 的中断服务程序
INTT0CHK:                           ; 检查是否有 T0 中断
            B0BTS1  FT0IEN          ; 检查是否允 T0 中断
            JMP     INTTC0CHK       ; 跳转到下一个中断
            B0BTS0  FT0IRQ          ; 检测是否有 T0 中断请求
            JMP     INTT0           ; 跳转到 T0 中断服务程序
INTTC0CHK:                          ; 检查是否有 TC0 中断
            B0BTS1  FTC0IEN         ; 检查是否允 TC0 中断
            JMP     INT_EXIT        ; 中断返回
            B0BTS0  FTC0IRQ         ; 检测是否有 TC0 中断请求
            JMP     INTTC0          ; 跳转到 TC0 中断服务程序
INT_EXIT:
            B0MOV   A, PFLAGBUF     ; 恢复 ACC 和 PFLAG
            B0MOV   PFLAG,A
            B0XCH   A, ACCBUF

            RETI                    ; 中断返回
```

1.4 基本定时器 T0

1.4.1 概述

基本定时器 T0 **是一个** 8 **位二进制加一计数器，由寄存器** T0M **选择** T0C **的输入时钟。当** T0 **溢出 （从** FFH **至** 00H**）时，产生一个信号触发** T0 **中断。**T0 **基本定时器的功能如下：**

- **8 位可编程定时器：根据所选的时钟频率，定时发出中断请求信号。**
- **RTC 定时器：以设定的钟频率为基础，在一定的间隔时间内产生实时中断。RTC 功能仅在 High_Clk 选择 “ IHRC_RTC”时可行。**
- **绿色模式唤醒功能：**T0ENB=1，T0 **定时器溢出使系统从绿色模式返回到上一个操作模式。**



1.4.2 T0M 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	-	-	-	T0TB
读/写	R/W	R/W	R/W	R/W	-	-	-	R/W
复位后	0	0	0	0	-	-	-	0

Bit0 **T0TB：** RTC **时钟源控制位。**
0 = **禁止（由** Fcpu **产生）**
1 = **使能（由** RTC **产生）**

Bit[6:4] **T0RATE2~T0RATE0：**T0 **内部时钟选择位。**
000 = fcpu/256,
001 = fcpu/128
, …,
110 = fcpu/4,
111 = fcpu/2。

Bit7 **T0ENB：**T0 **计数器控制位。**
0 = **禁止，**
1 = **使能。**

1.4.3 8.2.3T0C 计数寄存器

T0C 是一个 8 位计数寄存器。

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

T0 基本时间常数表

T0RATE	T0CLOCK	高速模式 Fcpu = 3.58MHz / 4)		低速模式 Fcpu = 32768Hz / 4)	
		最大溢出间隔时间	单步时间= max/256	最大溢出间隔时间	单步时间= max/256
000	fcpu/256	73.2 ms	286us	8000 ms	31.25 ms
001	fcpu/128	36.6 ms	143us	4000 ms	15.63 ms
010	fcpu/64	18.3 ms	71.5us	2000 ms	7.8 ms
011	fcpu/32	9.15 ms	35.8us	1000 ms	3.9 ms
100	fcpu/16	4.57 ms	17.9us	500 ms	1.95 ms
101	fcpu/8	2.28 ms	8.94us	250 ms	0.98 ms
110	fcpu/4	1.14 ms	4.47us	125 ms	0.49 ms
111	fcpu/2	0.57 ms	2.23us	62.5 ms	0.24 ms

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：3.58MHZ 高速模式下，设 TC0 的时间间隔为 10ms。 TC0C (74H) = 256 - (10ms× fcpu/64)**

初始值 = 256 - (TC0 中断间隔时间×输入时钟)

= 256 - (10ms× 3.58× 10^6 ÷ 4÷ 64)

= 256 - (10^{-2} × 3.58× 10^6 ÷ 4÷ 64)

= 116

= 74H

1.4.4 RTC 寄存器（OPTION）

RTC 的周期由寄存器 OPTION 控制，若使能 RTC 功能，则 T0 的间隔时间由 OPTION 固定为 0.5/1/2/4。

088H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPTION	-	-	-	-	RTC1	RTC0	-	-
读/写	-	-	-	-	R/W	R/W	-	-
复位后	-	-	-	-	0	0	-	-

Bit[3:2] **RTC[1:0]：**CPU 运行模式控制位

00 = 0.5sec

01 = 1sec

10 = 2sec

11 = 4sec

1.4.5 T0 操作流程

T0 定时器就是一个定时器中断，设置 T0 的操作流程如下：

☞ **T0 定时器停止计数，禁止 T0 的中断功能并清 T0 的中断请求标志。**

程序		注释
B0BCLR	FT0ENB	T0 定时器停止
B0BCLR	FT0IEN	禁止 T0 中断功能
B0BCLR	FT0IRQ	清 T0 中断请求标志

☞ **设置 T0 定时器使能或禁止 RTC 功能**

程序		注释
B0BSET	FT0TB	使能 T0 的 RTC 功能
B0BCLR	FT0TB	禁止 T0 的 RTC 功能

☞ **设置 T0 定时器的速率。若使能 T0 的 RTC 功能，就不设置 T0 的速率。**

程序		注释
MOV B0MOV	A, #00H T0M, A	T0 的速率由 T0M 的第 4~6 位控制，范围为 00h~70h

☞ **设置 T0 定时器的功能模式**

程序		注释
B0BSET	FT0IEN	使能 T0 中断功能

☞ **设置 T0 的间隔时间。若使能 T0 的 RTC 功能，则不设置 T0C。**

程序		注释
MOV B0MOV	A, #7FH T0C, A	设置 T0 的间隔时间

☞ **使能 T0 定时器**

程序		注释
B0BSET	FT0ENB	使能 T0 定时器计数

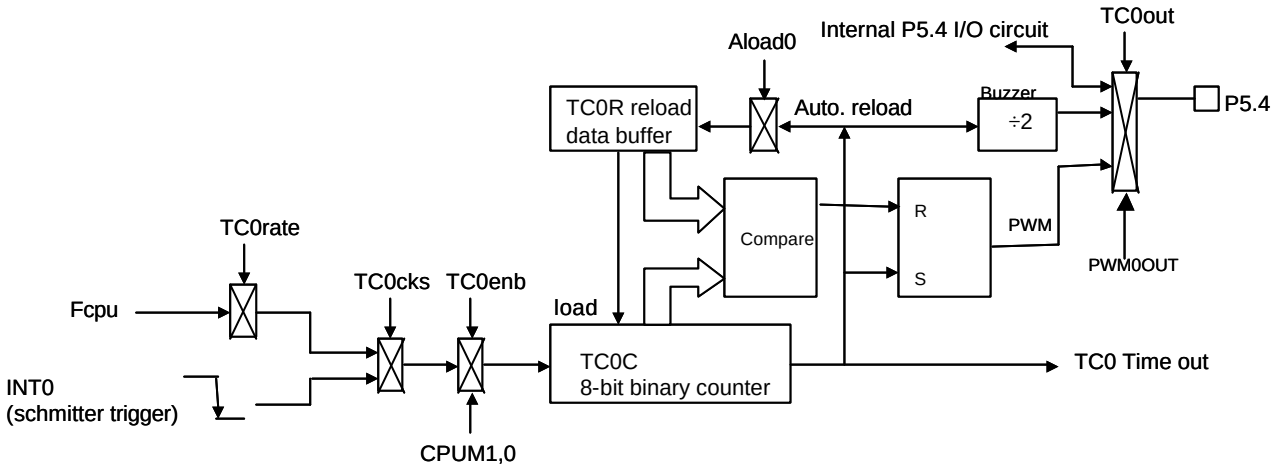
1.5 定时/计数器 TC0

1.5.1 概述

TC0 是一个二进制定时器/计数器，利用 TC0M 寄存器从 Fcpu 或者外部 INT0 引脚（下降沿触发）选择 TC0C 的时钟源，以进行精确的计时/计数。如果 TC0 溢出（从 FFH 到 00H），TC0 将继续计数并产生溢出触发信号，请求 TC0 中断服务。

TC0 定时器的主要功能如下：

- 8 位可编程定时器：根据设定的时钟频率，产生定时中断。
- 外部事件计数器：在 INT0 输入引脚端，用外部时钟信号的下降沿记录系统"事件"。
- BUZZER 输出
- PWM 输出



1.5.2 TC0M 模式寄存器

0DAH	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

Bit7 **TC0ENB：**TC0 **计数器控制位。**
0 = **禁止**
1 = **使能**

Bit[6:4] **TC0RATE2~ TC0RATE0：**TC0 **内部时钟选择位。**
000 = fcpu/256
001 = fcpu/128
…
110 = fcpu/4
111 = fcpu/2

Bit3 **TC0CKS：**TC0 **时钟源选择位。**
0 = Fcpu
1 = **来自** INT0/P0.0 **引脚的外部时钟**

Bit2 **ALOAD0：自动加载控制位。**
0 = **禁止**
1 = **使能**

Bit1 **TC0OUT：**TC0 **溢出信号输出控制位。仅在 PWM0OUT=0 时有效。**
0 = **禁止，**P5.4 **为基本输入/输出端**
1 = **使能，**P5.4 **为** TC0OUT **信号的输出端。**

Bit0 **PWM0OUT：**PWM **输出控制位。**
0 = **禁止**
1 = **使能**

✷ **注：当 TC0CKS=1，TC0 就是一个外部事件计数器，P00IRQ 始终为 0。**

1.5.3 TC0C 计数寄存器

TC0C 是一个 8 位计数寄存器，用来控制 TC0 的间隔时间。

0DBH	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

TC0 基本时间常数表

TC0RATE	TC0CLOCK	高速模式 Fcpu = 3.58MHz / 4)		低速模式 Fcpu = 32768Hz / 4)	
		最大溢出间隔时间	单步时间= max/256	最大溢出间隔时间	单步时间= max/256
000	fcpu/256	73.2 ms	286us	8000 ms	31.25 ms
001	fcpu/128	36.6 ms	143us	4000 ms	15.63 ms
010	fcpu/64	18.3 ms	71.5us	2000 ms	7.8 ms
011	fcpu/32	9.15 ms	35.8us	1000 ms	3.9 ms
100	fcpu/16	4.57 ms	17.9us	500 ms	1.95 ms
101	fcpu/8	2.28 ms	8.94us	250 ms	0.98 ms
110	fcpu/4	1.14 ms	4.47us	125 ms	0.49 ms
111	fcpu/2	0.57 ms	2.23us	62.5 ms	0.24 ms

TC0C 初始值的计算方法如下：

TC0C 初始值=256-（TC0 中断间隔时间×输入时钟）

➲ **例：3.58MHZ 高速模式下，设 TC0 的时间间隔为 10ms。TC0C(74H) = 256-(10ms× fcpu÷64)。(Fcpu=Fosc/4)**

TC0C 初始值 = 256- (TC0 中断间隔时间×输入时钟)

= 256-(10ms× 3.58× 10^6÷4÷64)

= 256-(10^{-2}× 3.58× 10^6÷4÷64)

= 116

= 74H

8.3.4TC0 溢出时间

TC0 的溢出时间有两种（PWM 模式和无 PWM 模式）。无 PWM 模式，其溢出边界为 0~255；PWM 模式，溢出边界包括四种(0~16，0~32，0~64，0~255)，由 TC0M 寄存器的 TC0OUT，ALOAD0 位控制。

TC0C 溢出时间列表

PWM0	ALOAD0	TC0OUT	TC0C 有效值	TC0C 的边界类型	备注
0	x	x	0x00~0xFF	00000000b~11111111b	每计数 256 次溢出
1	0	0	0x00~0xFF	00000000b~11111111b	每计数 256 次溢出
1	0	1	0x00~0x3F	xx000000b~xx111111b	每计数 64 次溢出
1	1	0	0x00~0x1F	xxx00000b~xxx11111b	每计数 32 次溢出
1	1	1	0x00~0x0F	xxxx0000b~xxxx1111b	每计数 16 次溢出

8.3.5 TC0R 自动装载寄存器

TC0 的自动装载功能由 TC0M 的 ALOAD0 位控制，当 TC0C 溢出时，TC0R 的值装载到 TC0C 中。易于产生一个精确时间，在中断发生时用户不需复位 TC0C。

✷ **注：在 PWM 模式下，自动使能自动装载功能，ALOAD0 位选择溢出的边界。**

0CDH	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

TC0R 初始值的计算如下：

TC0R 初始值 = N -（TC0 间隔时间*输入时钟）

➲ **例：设 Fosc=3.58MHz，时间间隔为 1ms。TC0R(74H) = 256- (10ms*fcpu/64)。Fcpu=Fosc/4**

TC0R 初始值 = 256 - (TC0 中断间隔时间 * 输入时钟)

= 256 - (10ms * 3.58 * 10^6 / 4 / 64)

= 256 - (10^{-2} * 3.58 * 10^6 / 4 / 64)

= 116

= 74H

8.3.6 TC0 操作流程

TC0 的操作包括定时器的中断，事件的计数。TC0OUT 和 PWM。TC0 的初始化操作如下：

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

程序		注释
B0BCLR	FTC0ENB	停止 TC0 定时器、TC0OUT 和 PWM
B0BCLR	FTC0IEN	禁止 TC0 中断功能。
B0BCLR	FTC0IRQ	清 TC0 中断请求标志

☞ **设置 TC0 的速率 (事件计数器模式除外)**

程序		注释
MOV B0MOV	A, #00H TC0M,A	由 TC0M 的第 4～6 位控制 TC0 的速率，其值处于 00h~70h 之间

☞ **设置 TC0 定时功能模式**

程序		注释
B0BSET	FTC0IEN	使能 TC0 中断
B0BSET	FTC0OUT	使能 TC0OUT 功能
B0BSET	FPWM0OUT	使能 PWM 功能

☞ **设置 TC0 定时时钟源**

程序		注释
B0BCLR	FTC0CKS	从系统时钟中选择 TC0 的时钟源
B0BSET	FTC0CKS	从外部事件计数器中选择 TC0 的时钟源

☞ **设置 TC0 的自动装载模式.**

程序		注释
B0BCLR	FALOAD0	禁止自动装载模式
B0BSET	FALOAD0	使能自动装载模式

☞ **设置 TC0 的间隔时间和 PWM 占空比**

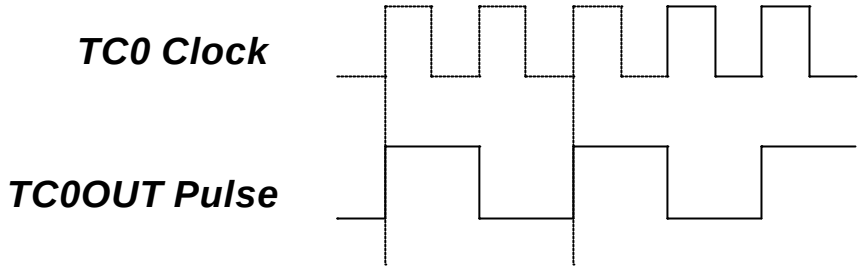
程序		注释
MOV B0MOV B0MOV	A, #7FH TC0C, A TC0R, A	设置 TC0 的间隔时间，并由 TC0C 设置 PWM 的占空比。TC0R 处于自动装载模式（PWM 模式下）
B0BCLR B0BCLR	FTC0OUT FALOAD0	设置 PWM 的占空比为 0~255

☞ **使能 TC0 定时器，TC0OUT 和 PWM 输出**

程序		注释
B0BSET	FTC0ENB	使能 TC0 定时/计数器

1.6 TC0 时钟频率输出（BUZZER 输出）

TC0 定时器/计数器提供了一个频率输出功能。通过设置 TC0 的时钟频率，可以从 P5.4 输出时钟信号，同时 P5.4 的基本输入/输出功会自动屏蔽。TC0 的输出信号是 2 分频的。由于 TC0 时钟源可以有多种选择，相应就可产生多种频率输出，这个功能常用作蜂鸣器输出。



➲ **例：设置 TC0 的 TC0OUT (P5.4)输出。外部高速时钟为 4MHz，TC0OUT 输出信号的频率为 1KHz。 因为 TC0OUT 输出信号是经过了 2 分频，因此设置 TC0 的时钟频率 2KHz。TC0 时钟源来自外部振荡器，T0C 的基频为 Fcpu/4. TC0RATE2~TC0RATE1 = 110. TC0C = TC0R = 131.**

```
MOV     A,#01100000B
B0MOV   TC0M,A          ; 设置 TC0 的时钟分频数 Fcpu/4

MOV     A,#131          ; 设置自动重新转载的时间常数
B0MOV   TC0C,A
B0MOV   TC0R,A

B0BSET  FTC0OUT         ; 使能 TC0OUT 功能
B0BSET  FALOAD0         ; 使能自动装载
B0BSET  FTC0ENB         ; TC0 开始计数
```

✷ **注：使能 buzzer 输出，“ PWM0OUT”必须为“ 0”。**

1.6.1 TC0OUT 频率表

Fosc=4MHz，TC0 Rate=Fcpu/8

TC0R	TC0OUT (KHz)	TC0R	TC0OUT (KHz)	TC0R	TC0OUT (KHz)	TC0R	TC0OUT (KHz)	TC0R	TC0OUT (KHz)
0	0.2441	56	0.3125	112	0.4340	168	0.7102	224	1.9531
1	0.2451	57	0.3141	113	0.4371	169	0.7184	225	2.0161
2	0.2461	58	0.3157	114	0.4401	170	0.7267	226	2.0833
3	0.2470	59	0.3173	115	0.4433	171	0.7353	227	2.1552
4	0.2480	60	0.3189	116	0.4464	172	0.7440	228	2.2321
5	0.2490	61	0.3205	117	0.4496	173	0.7530	229	2.3148
6	0.2500	62	0.3222	118	0.4529	174	0.7622	230	2.4038
7	0.2510	63	0.3238	119	0.4562	175	0.7716	231	2.5000
8	0.2520	64	0.3255	120	0.4596	176	0.7813	232	2.6042
9	0.2530	65	0.3272	121	0.4630	177	0.7911	233	2.7174
10	0.2541	66	0.3289	122	0.4664	178	0.8013	234	2.8409
11	0.2551	67	0.3307	123	0.4699	179	0.8117	235	2.9762
12	0.2561	68	0.3324	124	0.4735	180	0.8224	236	3.1250
13	0.2572	69	0.3342	125	0.4771	181	0.8333	237	3.2895
14	0.2583	70	0.3360	126	0.4808	182	0.8446	238	3.4722
15	0.2593	71	0.3378	127	0.4845	183	0.8562	239	3.6765
16	0.2604	72	0.3397	128	0.4883	184	0.8681	240	3.9063
17	0.2615	73	0.3415	129	0.4921	185	0.8803	241	4.1667
18	0.2626	74	0.3434	130	0.4960	186	0.8929	242	4.4643
19	0.2637	75	0.3453	131	0.5000	187	0.9058	243	4.8077
20	0.2648	76	0.3472	132	0.5040	188	0.9191	244	5.2083
21	0.2660	77	0.3492	133	0.5081	189	0.9328	245	5.6818
22	0.2671	78	0.3511	134	0.5123	190	0.9470	246	6.2500
23	0.2682	79	0.3531	135	0.5165	191	0.9615	247	6.9444
24	0.2694	80	0.3551	136	0.5208	192	0.9766	248	7.8125
25	0.2706	81	0.3571	137	0.5252	193	0.9921	249	8.9286
26	0.2717	82	0.3592	138	0.5297	194	1.0081	250	10.4167
27	0.2729	83	0.3613	139	0.5342	195	1.0246	251	12.5000
28	0.2741	84	0.3634	140	0.5388	196	1.0417	252	15.6250
29	0.2753	85	0.3655	141	0.5435	197	1.0593	253	20.8333
30	0.2765	86	0.3676	142	0.5482	198	1.0776	254	31.2500
31	0.2778	87	0.3698	143	0.5531	199	1.0965	255	62.5000
32	0.2790	88	0.3720	144	0.5580	200	1.1161		
33	0.2803	89	0.3743	145	0.5631	201	1.1364		
34	0.2815	90	0.3765	146	0.5682	202	1.1574		
35	0.2828	91	0.3788	147	0.5734	203	1.1792		
36	0.2841	92	0.3811	148	0.5787	204	1.2019		
37	0.2854	93	0.3834	149	0.5841	205	1.2255		
38	0.2867	94	0.3858	150	0.5896	206	1.2500		
39	0.2880	95	0.3882	151	0.5952	207	1.2755		
40	0.2894	96	0.3906	152	0.6010	208	1.3021		
41	0.2907	97	0.3931	153	0.6068	209	1.3298		
42	0.2921	98	0.3956	154	0.6127	210	1.3587		
43	0.2934	99	0.3981	155	0.6188	211	1.3889		
44	0.2948	100	0.4006	156	0.6250	212	1.4205		
45	0.2962	101	0.4032	157	0.6313	213	1.4535		
46	0.2976	102	0.4058	158	0.6378	214	1.4881		
47	0.2990	103	0.4085	159	0.6443	215	1.5244		
48	0.3005	104	0.4112	160	0.6510	216	1.5625		
49	0.3019	105	0.4139	161	0.6579	217	1.6026		
50	0.3034	106	0.4167	162	0.6649	218	1.6447		
51	0.3049	107	0.4195	163	0.6720	219	1.6892		
52	0.3064	108	0.4223	164	0.6793	220	1.7361		
53	0.3079	109	0.4252	165	0.6868	221	1.7857		
54	0.3094	110	0.4281	166	0.6944	222	1.8382		
55	0.3109	111	0.4310	167	0.7022	223	1.8939		

Fosc=16MHz，TC0 Rate=Fcpu/8

TC0R	TC0OUT (KHz)	TC0R	TC0OUT (KHz)	TC0R	TC0OUT (KHz)	TC0R	TC0OUT (KHz)	TC0R	TC0OUT (KHz)
0	0.9766	56	1.2500	112	1.7361	168	2.8409	224	7.8125
1	0.9804	57	1.2563	113	1.7483	169	2.8736	225	8.0645
2	0.9843	58	1.2626	114	1.7606	170	2.9070	226	8.3333
3	0.9881	59	1.2690	115	1.7730	171	2.9412	227	8.6207
4	0.9921	60	1.2755	116	1.7857	172	2.9762	228	8.9286
5	0.9960	61	1.2821	117	1.7986	173	3.0120	229	9.2593
6	1.0000	62	1.2887	118	1.8116	174	3.0488	230	9.6154
7	1.0040	63	1.2953	119	1.8248	175	3.0864	231	10.0000
8	1.0081	64	1.3021	120	1.8382	176	3.1250	232	10.4167
9	1.0121	65	1.3089	121	1.8519	177	3.1646	233	10.8696
10	1.0163	66	1.3158	122	1.8657	178	3.2051	234	11.3636
11	1.0204	67	1.3228	123	1.8797	179	3.2468	235	11.9048
12	1.0246	68	1.3298	124	1.8939	180	3.2895	236	12.5000
13	1.0288	69	1.3369	125	1.9084	181	3.3333	237	13.1579
14	1.0331	70	1.3441	126	1.9231	182	3.3784	238	13.8889
15	1.0373	71	1.3514	127	1.9380	183	3.4247	239	14.7059
16	1.0417	72	1.3587	128	1.9531	184	3.4722	240	15.6250
17	1.0460	73	1.3661	129	1.9685	185	3.5211	241	16.6667
18	1.0504	74	1.3736	130	1.9841	186	3.5714	242	17.8571
19	1.0549	75	1.3812	131	2.0000	187	3.6232	243	19.2308
20	1.0593	76	1.3889	132	2.0161	188	3.6765	244	20.8333
21	1.0638	77	1.3966	133	2.0325	189	3.7313	245	22.7273
22	1.0684	78	1.4045	134	2.0492	190	3.7879	246	25.0000
23	1.0730	79	1.4124	135	2.0661	191	3.8462	247	27.7778
24	1.0776	80	1.4205	136	2.0833	192	3.9063	248	31.2500
25	1.0823	81	1.4286	137	2.1008	193	3.9683	249	35.7143
26	1.0870	82	1.4368	138	2.1186	194	4.0323	250	41.6667
27	1.0917	83	1.4451	139	2.1368	195	4.0984	251	50.0000
28	1.0965	84	1.4535	140	2.1552	196	4.1667	252	62.5000
29	1.1013	85	1.4620	141	2.1739	197	4.2373	253	83.3333
30	1.1062	86	1.4706	142	2.1930	198	4.3103	254	125.0000
31	1.1111	87	1.4793	143	2.2124	199	4.3860	255	250.0000
32	1.1161	88	1.4881	144	2.2321	200	4.4643		
33	1.1211	89	1.4970	145	2.2523	201	4.5455		
34	1.1261	90	1.5060	146	2.2727	202	4.6296		
35	1.1312	91	1.5152	147	2.2936	203	4.7170		
36	1.1364	92	1.5244	148	2.3148	204	4.8077		
37	1.1416	93	1.5337	149	2.3364	205	4.9020		
38	1.1468	94	1.5432	150	2.3585	206	5.0000		
39	1.1521	95	1.5528	151	2.3810	207	5.1020		
40	1.1574	96	1.5625	152	2.4038	208	5.2083		
41	1.1628	97	1.5723	153	2.4272	209	5.3191		
42	1.1682	98	1.5823	154	2.4510	210	5.4348		
43	1.1737	99	1.5924	155	2.4752	211	5.5556		
44	1.1792	100	1.6026	156	2.5000	212	5.6818		
45	1.1848	101	1.6129	157	2.5253	213	5.8140		
46	1.1905	102	1.6234	158	2.5510	214	5.9524		
47	1.1962	103	1.6340	159	2.5773	215	6.0976		
48	1.2019	104	1.6447	160	2.6042	216	6.2500		
49	1.2077	105	1.6556	161	2.6316	217	6.4103		
50	1.2136	106	1.6667	162	2.6596	218	6.5789		
51	1.2195	107	1.6779	163	2.6882	219	6.7568		
52	1.2255	108	1.6892	164	2.7174	220	6.9444		
53	1.2315	109	1.7007	165	2.7473	221	7.1429		
54	1.2376	110	1.7123	166	2.7778	222	7.3529		
55	1.2438	111	1.7241	167	2.8090	223	7.5758		

1.7 PWM 功能说明

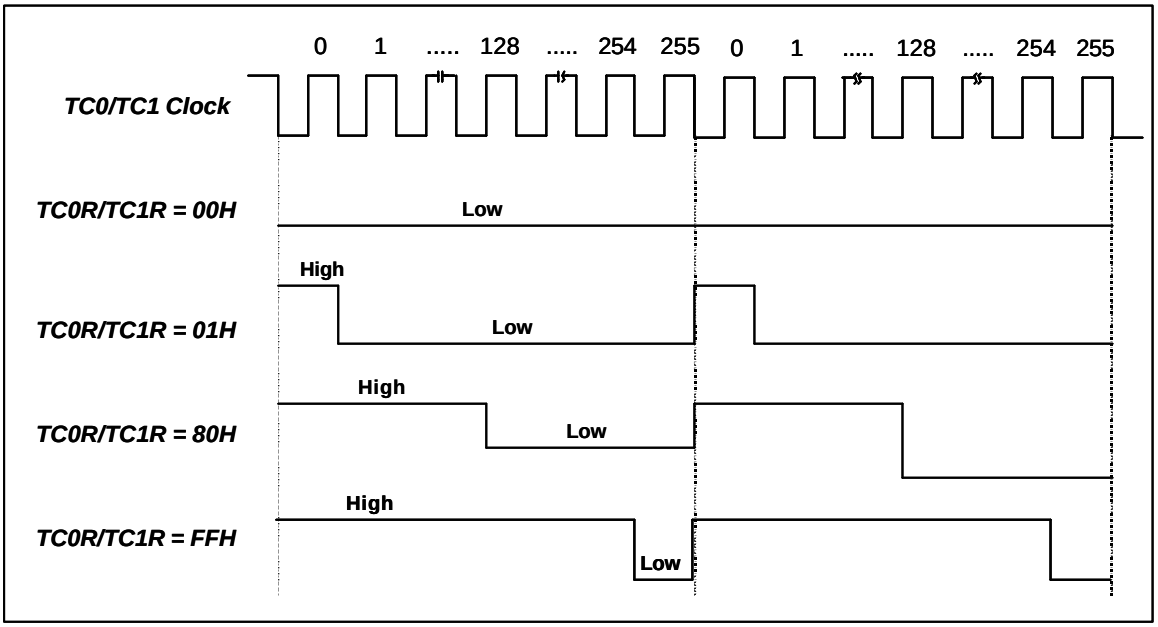
1.7.1 概述

PWM0 功能使用的时钟源为 TC0，产生的 PWM 信号通过 PWM0OUT 引脚（P5.4）输出。8 位计数器计数范围可为 256，64，32，16，由 ALOAD0 和 TC0OUT 位控制。8 位计数器的值与 TC0R 中的参考值相比较，当 TC0R 和 TC0C 的值相等时，PWM 输出低电平；当 TC0C 的值重新回到 0 时，PWM 输出高电平。PWM 的高、低电平之比（占空比）等于 TC0R/256，64，32，16。

向参考寄存器 TC0R 中写入 00H 可以使 PWM 输出保持在低电平；在 PWM 运行中，可以通过调整 TC0R 的值改变 PWM 的占空比。

ALOAD0	TC0OUT	PWM 占空比范围	TC0C 有效值	TC0R 有效位值	PWM 的最大频率 (Fcpu = 4MHz)	备注
0	0	0/256~255/256	0x00~0xFF	0x00~0xFF	7.8125K	每计数 256 次溢出
0	1	0/64~63/64	0x00~0x3F	0x00~0x3F	31.25K	每计数 64 次溢出
1	0	0/32~31/32	0x00~0x1F	0x00~0x1F	62.5K	每计数 32 次溢出
1	1	0/16~15/16	0x00~0x0F	0x00~0x0F	125K	每计数 16 次溢出

对不同的 TC0R，PWM 的输出占空比的范围为 0/256~255/256



1.7.2 PWM 程序说明

➲ **例：设置 PWM0 的输出 PWM0OUT (P5.4)，其中，外部高速振荡器时钟 4MHz，Fcpu=Fosc/4。 PWM 输出占空比为 30/256. PWM 的输出频率是 1KHz. PWM 的时钟源来自外部振荡器，TC0 的速率是 Fcpu/4. TC0RATE2~TC0RATE1 = 110. TC0C = TC0R = 30.**

```
MOV     A,#01100000B
B0MOV   TC0M,A          ; 设置 TC0 的时钟分频数 Fcpu/4

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

B0BCLR  FTC0OUT         ; 禁能 TC0OUT 功能
B0BCLR  FALOAD0
B0BSET  FPWM0OUT        ; 使能 PWM0 输出到 P5.4 并禁能 P5.4 的输入输出功能
B0BSET  FTC0ENB         ; TC0 开始计数
```

✷ **注：TC0R 是只写寄存器，不能执行 INCMS, DECMS 指令。**

➲ **例：调整 TC0R 寄存器的值。**

```
MOV     A, #30H         ;
B0MOV   TC0R, A

INCMS   BUF0            ;
B0MOV   A, BUF0         ;
B0MOV   TC0R, A
```

✷ **注：**
1. 在调整 PWM0 的占空比时，最好同时改变 TC0C 和 TC0R 的值，以避免 PWM0 的信号受到干扰；
2. PWM 功能在中断请求下可同时运行。