实验二 定时器和计数器

杨凯欣 1500012805

2018 年 3 月 21 日

目录

实验目的

1. 掌握使用 C 语言进行单片机程序开发的方法，熟悉开发环境；
2. 熟悉单片机 C 语言开发中的特殊语法；
3. 了解单片机中的定时器和计数器；
4. 了解单片机的中断系统原理和编程方法。

实验原理

C51 简介

对于 MCS-51 系列的单片机，支持它的 C 语言叫做 C51，和标准 C 有一定区别，主要体现在数据类型和数据存储结构上。

数据类型

特殊的数据类型有：

1. 位类型 bit

   使用位类型可以方便进行逻辑操作，位类型可以定义一个位变量，由编译器在内部 RAM 区 20H-2FH 的 128 个位地址中分配一个位地址。位类型不能定义指针和数组。

2. 特殊功能寄存器 sfr

   必须采用直接寻址的方式来访问，离散地分布在 80H-FFH 的地址空间里。sfr 可对特殊功能寄存器进行定义，sfr 型数据占用一个字节，取值范围 0-255。

3. 16 位特殊功能寄存器 sfr16

   如 DPTR 就可以用 sfr16 定义，sfr16 型数据占用两个字节，取值范围 0-65535。

4. 可寻址位类型 sbit

   sbit 可对内部 RAM 的位寻址空间及特殊功能寄存器的可寻址位进行定义。

   eg：sbit flag = P1.0，表示将 P1.0 这条 I/O 口线定义为 flag 变量。

只有 unsigned char 和 bit 类型是 8051 CPU 可以直接用汇编语言支持的数据类型，它们的操作效率更高。其他数据类型都有多条汇编指令组合操作，需占用大量的程序存储空间和数据存储器资源。C51 还支持结构类型和联合类型等复杂类型数据。

指定变量存储区域的关键字

data	直接寻址片内 RAM，00F-7FH 空间，速度最快
bdata	可位寻址片内 RAM，20H-2FH 空间，容许位和字节混合访问
idata	间接访问片内 00H-FFH 全部 256 个地址空间
pdata	使用 MOVX @Ri 指令访问外部 RAM 分页的 00H-FFH 空间
xdata	使用 MOVX @DPTR 访问外部 RAM 的 0000H-FFFFH 全部空间
code	使用 MOVC @A+DPTR 访问程序存储器 0000H-FFFFH 全部空间
_at_	指定具体的存储位置，如 unsigned int xdata a _at_ (0x500)

中断服务程序函数

为处理单片机中断 C51 提供了 interrupt 关键字，支持直接编写中断服务程序函数。函数定义的形式为：

```
函数类型 函数名() [interrupt n] [using n]
```

函数类型一般定义为 void。

interrupt n 是中断号，指示相应的中断源。C51 编译器从 code 区的绝对地址 8n+3 处产生中断向量。n 必须是常数，不允许使用表达式。

using n 是可选的，n 为 0-3 的常数，指示选择 4 个寄存器组，即由 PSW 中 RS0/RS1 所指定的工作寄存器可以保存的 4 个位置。如果不使用 using n，中断函数所有使用的公共寄存器都入栈（保护现场）。如果使用 using n，切换的寄存器就不再入栈（因为使用了不同的寄存器组，因此不需要保护 R0-R7）。

编写中断函数时应注意的几个问题：

- 中断函数没有返回值，因此必须是 void 类型；
- 中断函数不允许进行参数传递；
- 不允许直接调用中断函数；
- 中断函数对压栈和出栈的处理由编译器完成，无需人工管理；
- 需严格注意 using n 的使用，必须确保寄存器组的正确切换。

并行端口

C8051F020 共支持 64 个数字输入输出接口，共组成 8 组 8 位的并行端口。其中 P0、P1、P2 和 P3 被称为低位端口（标准的 MCS-51 只支持 4 个端口），可以通过寄存器按字节或者按位访问；对应的 P4、P5、P6、P7 为高位端口，仅可以按字节访问。所有端口都可以配置成漏极开路输出或推挽输出。

对 P0-P7 寄存器的读写可以实现对相应端口的输入和输出，另外部分端口还支持对其工作状态的设置。低位端口可以按位设置，例如 P0MDOUT 寄存器可以用来设置 P0 端口 8 个接口的输出模式，其中设置为 1 的位表示对应的接口为推挽方式输出，设置为 0 的位表示采用漏极开路的方式输出。但对于高位端口，只有一个 P74OUT 寄存器来设置端口的输出状态，只能按照 4 位每组的方式进行输出设置。

MCS-51 单片机的中断系统

MCS-51 单片机有 5 个中断源（2 个外部中断源 INT0 和 INT1，2 个片内定时器/计数器溢出中断 TF0 和 TF1，1 个片内串行口的收/发中断 RI）。C8051F020 具有多达 22 个中断源。

与中断有关的专用寄存器有中断允许寄存器 IE、中断优先级控制寄存器 IP、控制寄存器 TCON 等。其中有关定时器/计数器 2 的相关控制位是 C8051F020 扩展的。

D7	D6	D5	D4	D3	D2	D1	D0
EA		ET2	ES	ET1	EX1	ET0	EX0
总中断允许位		定时器/计数器 2 中断允许位	串行通信口中断允许位	定时器/计数器 1 中断允许位	外部中断 INT1 中断允许位	定时器/计数器 0 中断允许位	外部中断 INT0 中断允许位

Table 1: 中断允许寄存器 IE

D7	D6	D5	D4	D3	D2	D1	D0
		PT2	PS	PT1	PX1	PT0	PX0
		定时器/计数器 2	串行口	定时器/计数器 1	外部中断 1	定时器/计数器 0	外部中断 0

Table 2: 中断优先级控制寄存器 IP

MCS-51 仅支持一级中断嵌套，靠 IP 寄存器可以将中断优先级分为高、低两级，他们遵

从下面两条规则：1. 低优先级中断可被高优先级中断所中断，反之不能；2. 一种中断（不论哪个优先级）一旦得到响应，与它同级的中断不能再中断它。

当同时收到几个同一优先级的中断请求时，取决于内部的查询顺序，相当于在每个优先级内还有另一辅助优先级结构，中断序号小的中断具有较高的优先级。

D7	D6	D5	D4	D3	D2	D1	D0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
定时器/计数器 1 溢出标志	定时器/计数器 1 运行控制	定时器/计数器 0 溢出标志	定时器/计数器 0 运行控制	外部沿触发中断 1 请求标志	外部中断 1 触发类型控制	外部沿触发中断 0 请求标志	外部中断 0 触发类型控制
溢出时硬件置位申请中断，进入 ISR 后被硬件自动清除	软件，置位时启动定时器/计数器 1 工作，清除时停止工作	同 TF1	同 TR1	INT1 引脚出现外部中断信号下降沿，硬件置位请求中断，进入 ISR 后被硬件自动清除	软件，置位下降沿触发，清除低电平触发	同 IE1	同 IT1

Table 3: 控制寄存器 TCON

当单片机检测到中断事件发生并且相应的中断已经在控制寄存器中被允许，则 PC 指针跳转到中断所对应的入口地址执行那里的代码。中断入口一般是一个跳转指令，跳转到相应的中断处理程序运行。在中断程序最后，会执行 IRET 指定返回到中断前的地址继续运行。

中断源	外部中断 INT0	定时器/计数器 T0	外部中断 INT1	定时器/计数器 T1	串行口
入口地址	0003H	000BH	0013H	001BH	0023H

Table 4: 中断入口地址 (8n+3)

时钟和振荡器

C8051F020 内部包含一个振荡器，当系统复位时，单片机首先使用内部的振荡器作为系统时钟（缺省工作频率为 2MHz，用 SYSCLK 表示），而如果要使用其他的工作频率或者使用外部的晶体振荡器，都必须通过寄存器进行设置。

D7	D6	D5	D4	D3	D2	D1	D0
MSCLKE			IFRDY	CLKSL	IOSCEN	IFCN1	IFCN0
时钟失效检测，100μs 未检测到时钟信号时复位			检测内部振荡器是否工作在设置的频率下	选择系统使用的振荡器（1 外部 0 内部）	设置内部振荡器工作状态（1 使用 0 禁止）	IFCN1-0 设置内部振荡器工作频率	00:2MHz, 01:4MHz, 10:8MHz, 11:16MHz

Table 5: OSCICN 寄存器结构

D7	D6	D5	D4	D3	D2	D1	D0
XTLVLD	XOSCMD2	XOSCMD1	XOSCMD0		XFCN2	XFCN1	XFCN0
外部晶体振荡器工作状态（1 稳定 0 非稳定）	XOSCMD2-0 设置外部振荡器的模式	00x:XTAL1 接地（外部振荡器不工作）; 01x: 使用外部时钟源；10x: 使用 RC 振荡器，二分频；11x: 使用外部晶体振荡器	是否要进行二分频（对 RC 振荡器模式无效）		XFCN2-0 设置外部振荡器的频率范围	111 表示频率大于 6.7MHz	其它设置参考数据手册

Table 6: OSCXCN 寄存器结构

```
void SYSCLK_Init (void){
    int i;
    OSCXCN = 0x67;                  //启动外部晶振
    for (i = 0; i < 256; i++);      //延时一段时间
    while (!(OSCXCN & 0x80));       //等待振荡稳定
    OSCICN = 0x88;                  //使用外部振荡器
}
```

定时器和计数器

MCS-51 系列单片机有两个定时器/计数器，在模式控制寄存器 TMOD 中各有一个控制位（C/T），分别用于控制定时器/计数器 0 和 1 是否工作在定时器方式还是计数器方式，当计数值发生溢出的时候会触发中断。

1. 选择定时器工作方式时，计数输入是内部系统时钟，每个机器周期使寄存器的值加 1；

2. 选择计数器工作方式时，计数脉冲来自相应的外部输入引脚 T0 或 T1，当输入信号由 1 跳变至 0 时，计数寄存器的值加 1。

除了可以选择定时器或计数器工作方式外，每个定时器/计数器还有 4 种操作模式，其中前三种模式对两者都是一样的，只有模式 3 对两者不同。

1. 模式 0：13 位定时器/计数器，控制逻辑 $TR_x \times (\overline{GATE} + INT_x)$

2. 模式 1：16 位定时器/计数器，控制逻辑同模式 0

3. 模式 2：把定时器寄存器 TLx 配置成一个可以自动重载的 8 位计数器

4. 模式 3：定时器/计数器 1 将保持原有的计数值；定时器/计数器 0 将使 TL0 和 TH0 成为两个互相独立的 8 位计数器。

D7	D6	D5	D4	D3	D2	D1	D0
GATE	C/T	M1	M0	GATE	C/T	M1	M0

Table 7: TMOD 寄存器

D7	D6	D5	D4	D3	D2	D1	D0
	T4M	T2M	T1M	T0M			

Table 8: CKCON 寄存器

实验内容

控制 LED 闪烁

主板上的 LED 与 CPU 的 P7 端口最高位相连，通过控制 P7 的输出，可以让这个 LED 闪烁。

调试过程中可以在反汇编窗口看到相应的汇编代码，通过对比可以更加深刻地了解单片机的工作原理。

在工程设置中的 Listing 页，勾选 C Compiler Listing 中的 Assembly Code 选项，就可以在编译工程时生成的列表文件中包含生成的汇编代码。

```
#include <C8051F020.h>
void PORT_Init(void){
    P74OUT = 0x80;  //10000000B, set P7.7 as push-pull output
}

//function 'Delay' to hold on the state
void Delay(){
    int i;
    int j;
    for (i = 0; i < 500; ++i)
        for (j = 0; j < 180; ++j);
    return;
}

void main(void){
    WDTCN = 0xDE;
    WDTCN = 0xAD;  //disable the watchdog

    //initial
    PORT_Init();

    while(1){
        P7 = 0x80;   //turn on LED
        Delay();
        P7 = 0x00;   //turn off LED
        Delay();
    }
    return;
}
```

使用计数器延时

使用计数器 0 控制 LED 的闪烁频率为 1Hz。为了保证闪烁频率的准确，要使用外部晶振。

```
void PORT_Init(void){
    P74OUT = 0x80;  //10000000B, set P7.7 as push-pull output
    return;
}

```

```
void SYSCLK_Init(void){
    int i;
    OSCXCN = 0x67; //the external oscillator is not stable; the frequency is more than 6.7MHz
    for (i = 0; i < 256; i++);
    while (!(OSCXCN & 0x80));
    OSCICN = 0x88;
    return;
}

void INTERRUPT_Init(void){
    IE = IE | 0x82;   //IE = 10000010, enable counter0
    IP = IP | 0x02;   //IP = 00000010, set interrupt of counter0 the priority
    return;
}

void COUNTER_Init(void){
    TMOD = TMOD | 0x02; //set the counter0 in the mode 2
    TH0 = 0x00;
    TL0 = 0x00;
}

void main(void){
    WDTCN = 0xDE;
    WDTCN = 0xAD; //disable the watchdog

    //initial
    PORT_Init();
    SYSCLK_Init();
    INTERRUPT_Init();
    COUTER_Init();

    TR0 = 1;   //turn on the counter
    P7 = 0x80;   //output HIGH voltage
    while(1);
    return;
}

void COUNTER_ISR (void) interrupt 1{
    static int count;   //total of interruption times
    count++;
    if (count > 3600){
        count = 0;
```

```
48          P7 = ~P7;
49      }
50  }
```

控制 LED 亮度

PWM 是脉冲宽度调制技术，通过控制输出方波信号的占空比，来达到输出功率的变化，从而很方便地采用数字输出来进行模拟电路的控制，经常用来控制发光器件的亮度和电机的转速等等。

```
1   void PORT_Init(void){
2       P74OUT = 0x80;  //10000000B, set P7.7 as push-pull output
3       return;
4   }
5
6   void SYSCLK_Init(void){
7       int i;
8       OSCXCN = 0x67;  //the external oscillator is not stable; the frequency is more than 6.7MHz
9       for (i = 0; i < 256; i++);
10      while (!(OSCXCN & 0x80));
11      OSCICN = 0x88;
12      return;
13  }
14
15  void INTERRUPT_Init(void){
16      IE = IE | 0x82;   //IE = 10000010, enable counter0
17      IP = IP | 0x02;   //IP = 00000010, set interrupt of counter0 the priority
18      return;
19  }
20
21  void COUNTER_Init(void){
22      TMOD = TMOD | 0x02;  //set the counter0 in the mode 2
23      TH0 = 0x00;
24      TL0 = 0x00;
25  }
26
27  void main(void){
28      WDTCN = 0xDE;
29      WDTCN = 0xAD;  //disable the watchdog
30
31      //initial
```

```
    PORT_Init();
    SYSCLK_Init();
    INTERRUPT_Init();
    COUTER_Init();

    TR0 = 1;  //turn on the counter
    P7 = 0x80;  //output HIGH voltage
    while(1);
    return;
}


#define period_init 24

void COUNTER_ISR (void) interrupt 1{
    static int count;  //total of interruption times
    static signed int step = -1;
    static int last_time = 0;  //increase the period of amplitude's change
    static int flag_p_n = 1;  //sign now is positive period or negative period
    static signed int period_p = period_init / 2;
    //the length of positive period, can use it to change duty cycle
    static signed int period = period_init / 2;
    //threshold that sign the time to flip output
    count++;
    if (count > period){
        last_time++;
        if (flag_p_n == 1){
            period = period_init - period_p;
            flag_p_n = 0;
        }
        else{
            period = period_p;
            flag_p_n = 1;
        }
        if (last_time == 50){
            last_time = 0;
            period_p += step;
        }
        if (period_p <= 0)
            step = 1;
        else if(period_p > period_init / 2)
            step = -1;
```

```
74          count = 0;
75          P7 = ~P7;
76      }
77  }
```

思考题

1. 如何估算 C 代码中 Delay 函数的延时?

机器执行单条基本指令需要的机器周期是时钟周期的 12 倍，因而在 Delay 函数中，估算公式为：

$$t_D = \frac{1}{F_{CPU}} * I * J = \frac{12}{F_{crystal}} * I * J$$

其中 I 表示第一重循环次数，J 表示第二重循环次数。如果采用默认振荡器 $F_{crystal}$ = 2MHz，求得：

$$t_D = \frac{12}{22.1184MHz} * 500 * 100 = 300ms$$

2. 在 PWM 输出的实验中，如果方波频率过低（比如低于 20Hz）会有什么现象发生?

会观察到光强不稳定，以一定的频率抖动。

心得体会

1. 最后一段程序中的中断函数设计得过于复杂，使用太多变量且可读性差，可以简化为：

```
#define period 24

void COUNTER_ISR (void) interrupt 1{
    static int count;
    static int pos_period = period / 2;
    static int step = -1;
    count++;
    if (count % 1200 == 0)  //change the duty cycle
        pos_period += step;
    if (count >= 2 * 12 * 1200){
        //pos_period == 12 and begin to decrease the pos_period
        count = 0;
        step == -1;
```

```
    }
    if (count == 14400)
        //pos_period == 0 and begin to increase the pos_period
        step = 1;
    if (count % period == 0)
        //turn on the LED
        P7 = 0x80;
    else if ((pos_period == 0) || ((count % period) % pos_period == 0))
        //turn off the LED
        P7 = 0x00;
}
```

2. 设置 P7 的最高位要用`P7 = 0x80`，不能使用`P7.7 = 1`，会报语法错。
3. 推挽输出才有驱动能力。
4. while 死循环反汇编结果：使用 SJMP 指令跳转回循环起始命令处。
5. C8051 只有两个中断优先级，同级序号小的优先级高。
6. 自装载的工作模式使得中断发生后计数器可以重新计数，保证每次时间间隔基本不变。