嵌入式计算机系统与实验 | 作业 5

Shuiyuan@Noroshi

1 定时器

1.1 外部方波宽度的测量

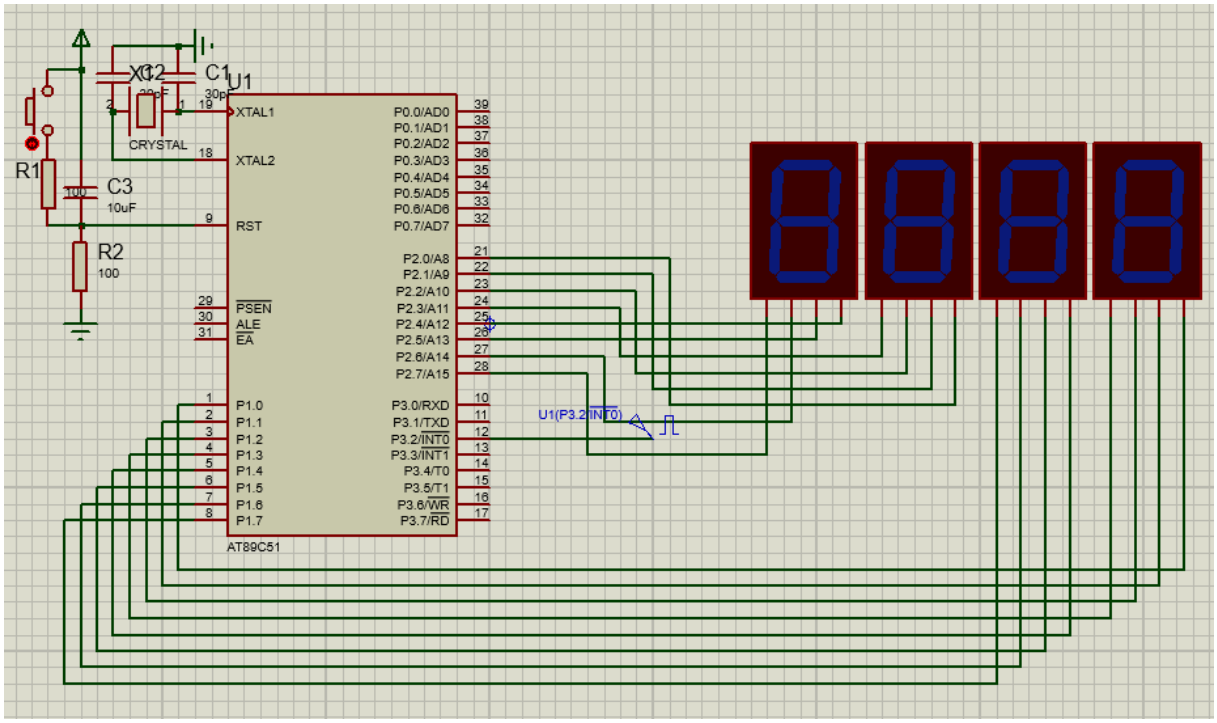
这个相对来说比较简单，就是用一个计数器来计算外部方波处在高电平状态时的时间就可以，先看看代码

```
ORG 0000H
LJMP START

ORG 0100H
  START:
  MOV TMOD, #00001001B
  MOV TH0, #00H
  MOV TL0, #00H
  CLR P3.2
  SETB TR0
  WAIT_HIGH:
  JNB P3.2, WAIT_HIGH
  WAIT_LOW:
  JNB P3.2, FINISH
  LJMP WAIT_LOW

  FINISH:
  MOV A, TL0
  MOV R0, A
  MOV A, TH0
  MOV R1, A

  SHOW_LED:
  MOV A, R0
  MOV B, #10
  DIV AB
  PUSH ACC
  MOV A, B
  MOV P2, A
  POP ACC
  MOV B, #10
  DIV AB
  MOV R4, B

  MOV R5, #4
  SHIFT:
  RL A
  DJNZ R5, SHIFT

  ORL A, R4
```

```
  MOV P1, A

  LJMP START
END
```

我们先聚焦于 SHOW_LED 以外的部分，那么就是一开始先把 TMOD 寄存器设置为 00001001，具体也就是置一门空位，使定时器只在外部方波拉高时启动，然后由于我们采用的是计数器模式，所以 TH 设置为 0，之后我们打开 TR0 的内部启动开关，这样外部一但拉高就会开始计数，然后在检测到外部置零的时候自动终止计数，并将高低位分别保存到 R0，R1 中.

接下来我们看看电路



这里可以稍微提一下我在这里选择的 LED 显示方式，其采用了四个 7-SEG-BCD-BLUE 元件，其与之前我们使用的常规七段 LED 的灯管的区别就是其只有四个引脚，就通过这四个引脚输入的二进制数来输出对应的数值，例如输入了 0010B 就显示 2，输入了 1010B 就显示 A，在代码方面，我 P1 口来显示低八位的数值，P2 口来显示高八位的数值，让我们来看看效果:

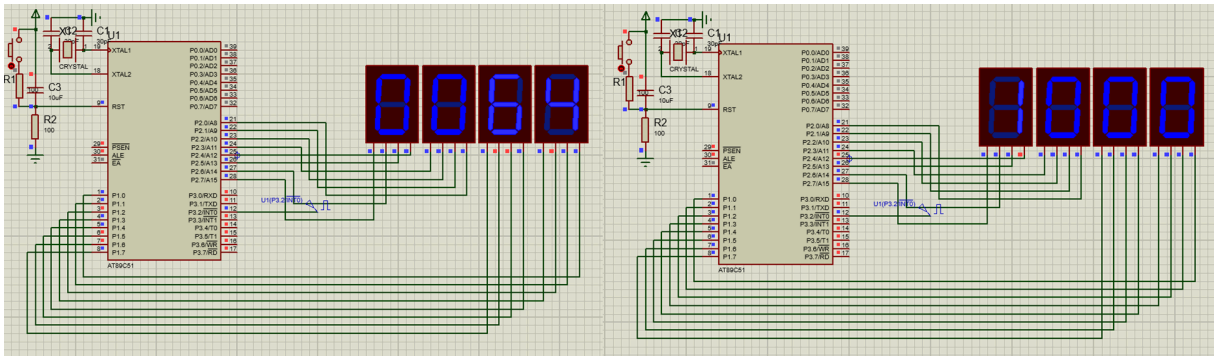


Figure 2: 运行效果展示，左边为方波长度为 100us(16 进制表示为 64)，右边为 4096us(十六进制表示为 1000)，此处数码管显示的是 16 进制的表示方法

1.2 中断输出方波

这部分我们目的是输出一个周期为 20ms 的方波，具体需要使用中断的方式来调节，先看代码

```
ORG 0000H
LJMP MAIN

ORG 000BH
LJMP DVT0

ORG 0100H
  MAIN:
  MOV TMOD, #00000001B
  MOV TH0, #0D8H
  MOV TL0, #0F0H
  SETB ET0
  SETB EA
  SETB TR0
  SJMP $

  DVT0:
  CPL P1.0
  MOV TH0, #0D8H
  MOV TL0, #0F0H
  RETI

END
```

主程序部分，我们在一开始将 TMOD 设为了 00000001B，也就是把定时器零设置为了 16 位计时器，定时器零的初值为 D8F0，这样当其从初值计数到触发中断的时候大概耗时就是 10ms，之后我们启动定时器，并打开中断使能

当触发中断时，其会从为定时器零中断预留 000B 处自动跳转到我们自己编写的中断处理程序，具体内容就是反转 P1.0 的输出，并重设定时器初值，之后返回.

电路部分也很简单，我们就和结果一起看

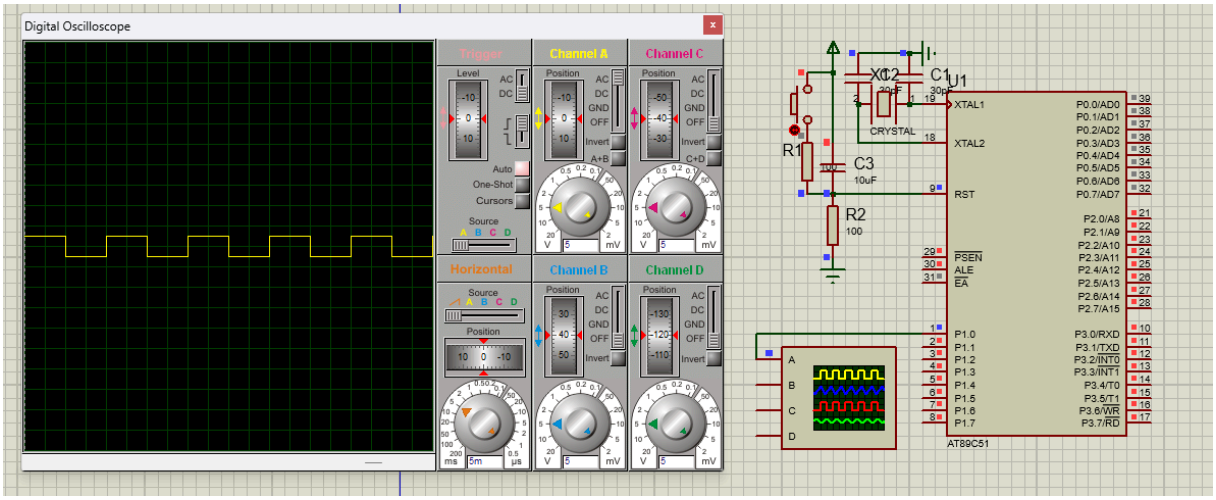


Figure 3: 这里左边的示波器每个小格为 5ms，也就是 P1.0 在输出 10ms 高电位，10ms 低电位循环的周期为 20ms 的方波

1.3 方波频率的测量

这部分我们联合使用定时器和计数器来实现方波频率的测量，具体设计也就是一个定时器作为计时器，另一个作为计数器，当方波完成一个周期之后，就计数+1，当定时器计到 1s 时，读取计数器中的数量.

代码:

```
ORG 0000H
LJMP MAIN

ORG 000BH
LJMP RESET_TIMER_0

ORG 0100H
  MAIN:
  MOV A, #00H
  MOV TMOD, #01010001B
  MOV TH0, #03CH
  MOV TL0, #0B0H

  MOV TH1, #00H
  MOV TL1, #00H

  SETB TR0
  SETB TR1

  SETB ET0
  SETB EA

  SJMP $

  RESET_TIMER_0:
  MOV TH0, #3CH
  MOV TL0, #0B0H

  INC A
  CJNE A, #20, RESET_TIMER

  DONE:
  CLR TR1
  MOV A, TL1
  LCALL SHOW_LED
  SJMP $

  RESET_TIMER:
  SETB TR0
  RETI

  SHOW_LED:
  MOV B, #10
  DIV AB
  MOV P2, B
  MOV B, #10
  DIV AB

  MOV R1, #4
  SHIFT:
  RL A
  DJNZ R1, SHIFT

  ORL A, B
  MOV P1, A

END
```

具体分析这段代码的话，第一步我们将 T1 设置为了计数模式，T0 设置为了定时模式，之后我们装填了 T0 的定时为 3CB0H 计算可得以此作为初值溢出触发中断时大约用时 0.05s， 也就是在触发二十次中断之后，满一秒，这个时候我们读 T1 的数值就行

中断程序也是这样设置的，当触发二十次中断时候才会进入计算频率和输出的部分

而 T1 方面不需要什么特殊的设置，因为计数器的自带特性之一就是可以不用内部时钟，而是引入外部的脉冲源，所以不需要什么特别的设置，把要测量的脉冲源自己接到 T1 口(P3.4)上就可以计数了

3 位 7 段 LED 显示的方式也同第一个作业

电路部分也没有什么好说的，见图

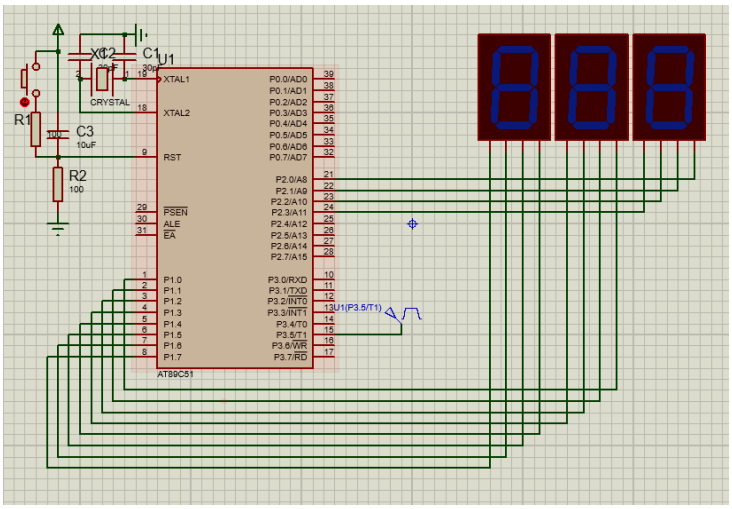


Figure 4: 电路连接图

运行结果:

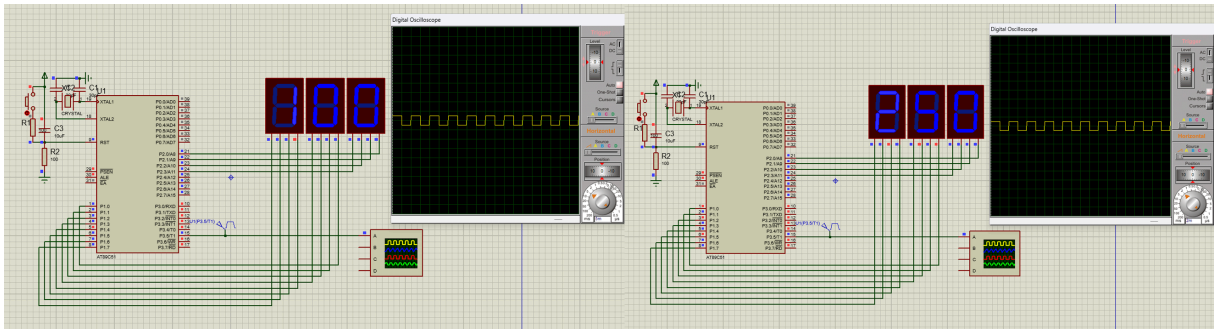


Figure 5: 如图，我们可以看见 LED 显示的频率(十进制)和观察示波器波形得到的频率是一致的

2 串口通讯

2.1 简单串口通讯实例

这部分我们实现了一个最简单的串口通讯，单片机一监听键盘，并在键盘有数字按下之后，将按下的数字发送给第二个单片机，单片机接收到了后就输出

先看代码:

输入(键盘)端:

```
ORG 0000H
LJMP INIT
```

```
ORG 0100H
  INIT:
  MOV SCON, #01010000B
  MOV TMOD, #00100000B
  MOV TH1, #0FDH
  MOV TL1, #0FDH
  SETB TR1

  MAIN:
  ACALL READ_KEYBOARD
  MOV SBUF, A
  JNB TI, $
  CLR TI
  SJMP MAIN
  READ_KEYBOARD:
  MOV P2, #11111111B
  CLR P2.0
  JNB P2.3, PRESS_1
  JNB P2.4, PRESS_4
  JNB P2.5, PRESS_7
  SETB P2.0

  CLR P2.1
  JNB P2.3, PRESS_2
  JNB P2.4, PRESS_5
  JNB P2.5, PRESS_8
  JNB P2.6, PRESS_0
  SETB P2.1

  CLR P2.2
  JNB P2.3, PRESS_3
  JNB P2.4, PRESS_6
  JNB P2.5, PRESS_9
  SETB P2.2

  RET

  PRESS_1:
  MOV A, #01H
  RET
  PRESS_2:
  MOV A, #02H
  RET
  PRESS_3:
  MOV A, #03H
  RET
  PRESS_4:
  MOV A, #04H
  RET
  PRESS_5:
  MOV A, #05H
  RET
  PRESS_6:
  MOV A, #06H
  RET
  PRESS_7:
  MOV A, #07H
  RET
  PRESS_8:
  MOV A, #08H
```

```
  RET
  PRESS_9:
  MOV A, #09H
  RET
  PRESS_0:
  MOV A, #00H
  RET

END
```

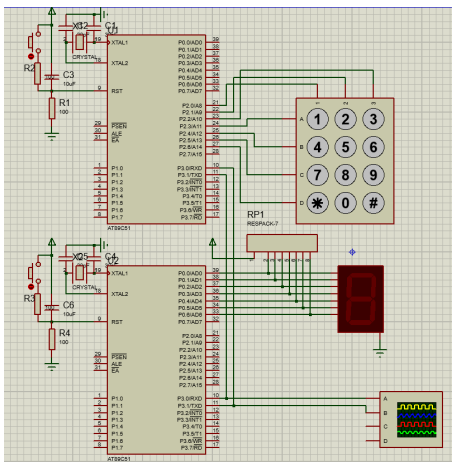
这里面最占篇幅的部分其实是读取输入部分，发送部分实际上很简单，就是直接把数据移到 SBUF 里，然后等发送，收到发送成功信号后就继续后续内容了

再看看接收端：

```
ORG 0000H
LJMP INIT

ORG 0030H
  INIT:
  MOV TMOD, #20H
  MOV TH1, #0FDH
  MOV TL1, #0FDH
  SETB TR1
  MOV SCON, #50H
  SETB ES
  SETB EA

  MAIN:
  JNB RI, $
  CLR RI
  MOV A, SBUF
  MOV DPTR, #LED_CODE
  MOVC A, @A+DPTR
  MOV P0, A
  SJMP MAIN

  LED_CODE:
  DB 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F
END
```

接收端也同样很简单，就是等待接收，当收到接收信号之后就取出 SBUF 中的值，然后之后用查表的方式取得对应的 LED 码并输出显示.

电路部分：



运行结果展示:

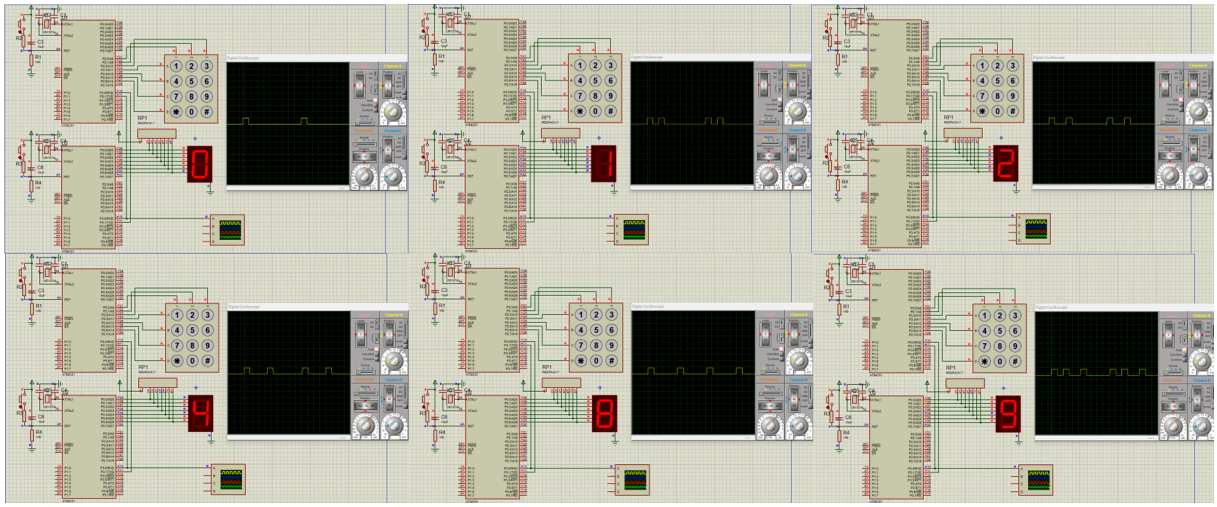


Figure 7: 除了通过观察 LED 灯以外，我们也可以从连接到数据收发线的示波器来观察更具体的情况，可以发现，在初始状态下也会有一次脉冲，这里其实上就是 TI，RI 之间发送的第一次脉冲，而之后才会发送实际的数据，以二进制的形式(从后往前)发送，例如 8 应该是 00001000，也就是 低-低-低-高-低-低-低-低，1 是 00000001，也就是 高-低-低-低-低-低-低-低，9 是 00001001，也就是 高-低-低-高-低-低-低-低