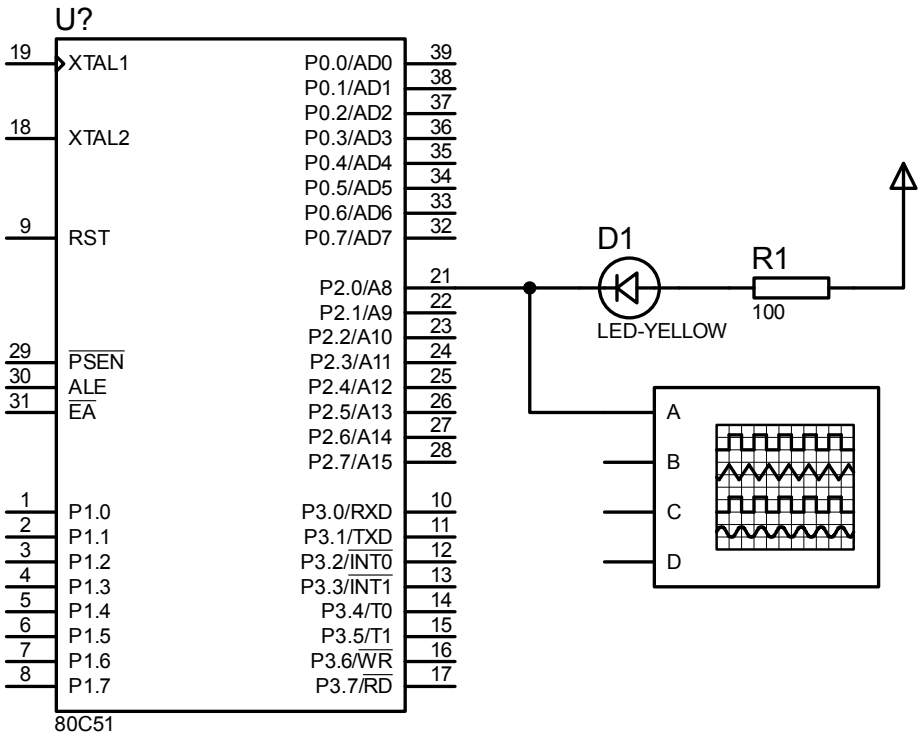
实例 1 设单片机的 fosc=12MHz，采用 T1 定时方式 1 在 P2.0 脚上输出周期为 2ms 的方波。



分析：周期为 2ms 的方波由 2 个半周期为 1ms 的正负脉冲组成

方波输出原理：定时 1ms 后将端口输出电平取反。

1ms 定时的计数初值应为：$a = 2^{16} - t^* fos / 12 = 2^{16} - 1000^* 12/ 12 = 64536 = 0xfc18$

TH1 = 0xfc TL1 = 0x18

注意：需要不断重装计数初值。

（1）查询方式

```
#include <reg51.h>
sbit P2_0 = P2^0;
main () {
    TMOD = 0x10;              //设置 T1 定时方式 1(0001 0000B)
    TR1=1;                    //启动 T0
    for(;;){
        TH1 = 0xfc;           //装载计数初值
        TL1 = 0x18;
        do{ } while(!TF1); //等待 TF1 溢出
        P2_0 =!P2_0;          //定时时间到 P2.0 反相
        TF1 = 0;              //TF1 标志清 0
      }
}
```

（2）中断方式

```
#include <reg51.h>
sbit P2_0=P2^0;
timer0 () interrupt 3 {       //T1 中断函数
    P2_0 = !P2_0;             //P2.0 取反
    TH1 = 0xfc;               //装载计数初值
    TL1 = 0x18;
}
main () {
    TMOD = 0x10;              //T1 定时方式 1
     TH1 = 0xfc;              //装载计数初值
    TL1 = 0x18;
    EA=1;                     //开总中断
    ET1=1;                    //开 T1 中断
    TR1=1;                    //启动 T1
     while(1);
}
```

实例 2 采用 T0 定时方式 2 在 P2.0 口输出周期为 0.5ms 的方波(设 fosc=12MHz)。

分析：计数初值 TL0= ((256-250)*12/12)%256 = 0x06，TMOD = 0x02

（1）查询方式

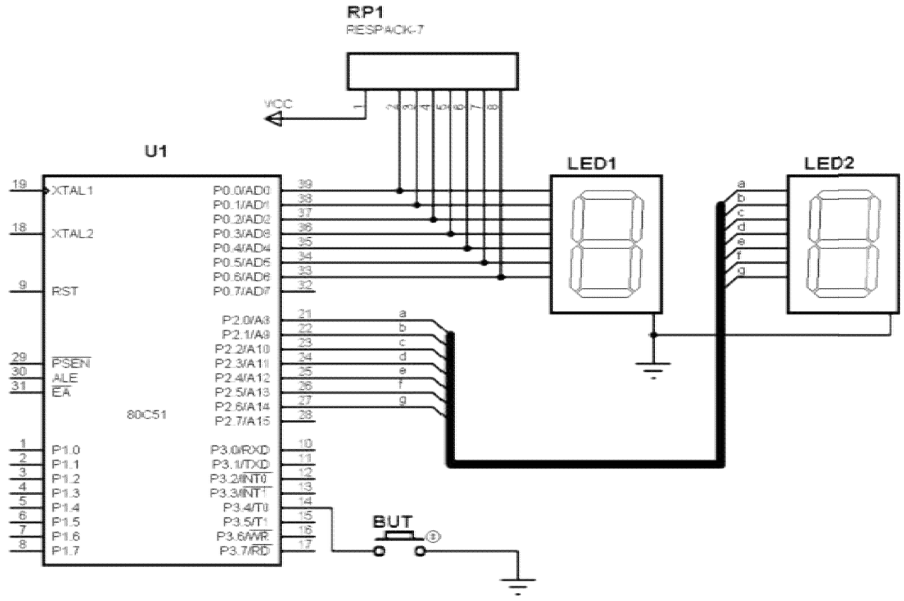
```
#include <reg51.h>
sbit P2_0 = P2^0;
main(){
    TMOD = 0x02;
     TH0= TL0 = 0x06;
    TR0=1;
    for(;;){
          do{} while(!TF0);
        P2_0 =!P2_0;
        TF0 = 0;
     }
}
```

（2）中断方式

```
#include <reg51.h>
sbit P2_0=P2^0;
timer0 () interrupt 1 {
    P2_0 = !P2_0;
}
main(){
    TMOD = 0x02;
     TH0 = TL0 = 0x06;
    EA= ET0 = 1;          TR0=1;
     while(1);
}
```

实例 3　改进“计数显示器” 的按键查询检测法，改用 T0 计数方式 2 + 中断法实现原有功能。

【解】电路改造：按键由 P3.7 改为 P3.4（T0）接入。



分析：T0 计数方式 2 的初始化；定数计数 N=1 时的初值计算；T0 中断初始化。

T0 计数方式 2：TMOD = 0000 0110B = 0x06

计数初值：　$a = 2^8 - 1 = 255 = 0xff$

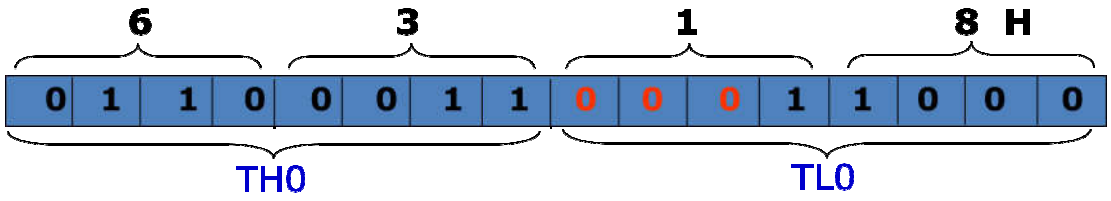
T0 中断初始化：ET0 = EA = 1

```
#include <reg51.h>
unsigned char code table[]={0x3f,0x06,0x5b,0x4f,
                            0x66,0x6d,0x7d,0x07,0x7f,0x6f};
unsigned char count=0;              //计数器赋初值

int0_srv () interrupt 1{       //T0 中断函数
      if(++count==100) count=0; //判断循环是否超限
      P0=table[count/10];              //显示十位数
      P2=table[count%10];              //显示个位数
}

main(){
    P0=P2=table[0];              //显示初值“00”
    TMOD=0x06;                        //T0 计数方式 2
    TH0=TL0=0xff;                //计数初值
    ET0=1;                            //开中断
    EA=1;
    TR0=1;                            //启动 T0
    while(1);
}
```

实例 4　计算 T0 方式 0 定时 5ms 的计数初值 a（设 fosc=12MHz）

解：计数初值 $a=2^{13}-5000\times12/12=3192=1100\ 0111\ 1000B$

由于方式 0 的 TL0 高 3 位未用（一般填 0），因此 a= 0110 0011 0001 1000 = 6318H



即，TH0 = 0x63;　　TL0 = 0x18;

除计数器位数不同外，方式 0 与方式 1 的逻辑结构并无差异。方式 0 采用 13 位计数器是为了与早期产品 MCS-48 单片机兼容。方式 0 的初值计算比较麻烦，一般采用方式 1 替代。

实例 5　由 P3.4 口输入一个低频窄脉冲信号。当该信号出现负跳变时，由 P3.0 口输出宽度为 500μs 的同步脉冲，如此往复。要求据此设计一个波形展宽程序（fosc= 6MHz）。

分析：可以采取如下做法：

1）将 T0 设置为 **1 次计数方式 2**，初值设为 0xff。这样 P3.4 一旦发生负跳变 T0 就会产生溢出；

2）查询 TF0 标志位。当 TF0=1 时将 T0 设置为 **500μs 定时方式 2**，初值 a 为 0x06(=256-500×6/12)，同时使 P3.0 输出低电平；

3）查询 TF0 标志位。待 T0 再次溢出后使 P3.0 输出高电平，然后将 T0 设置为 **1 次计数方式 2**，如此往复进行。

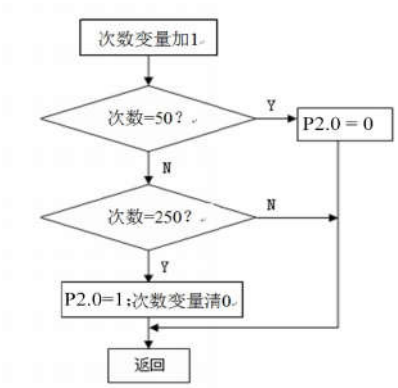
```
#include <reg51.h>
sbit P3_0=P3^0;
void main (){
    TMOD = 0x06;  //设置为 T0 计数方式 2
    TL0 = 0xff;       //初值 0xff 可使 1 个外来脉冲即产生溢出
    TR0 = 1;              //启动计数器
    while (1){
      while (!TF0); //等待首次溢出
      TF0 = 0;       //清 TF0 溢出标志
      TMOD = 0x02;     //设置为 T0 定时方式 2
      TL0=0x06;          //500 微秒定时初值
      P3_0 = 0;
      while (!TF0); //等待再次溢出
      TF0 = 0;       //清 TF0 溢出标志
      P3_0 = 1;
      TMOD = 0x06;     //设置为 T0 计数方式 2
      TL0 = 0xff;    //1 次溢出计数初值
    }
}
```

实例 6　采用 10MHz 晶振，在 P2.0 脚上输出周期为 2.5s，高电平占空比为 20%的脉冲信号。

分析：10 兆晶振，方式 1 最大定时为 54.613ms；

- **定时中断与软件计数联合**法：利用定时中断进行中断次数统计；
- 若取 10ms 产生定时，则 2.5s =250 次中断之和；
- 则 0.5ms（20%占空比）相当于 50 次中断之和。
- a = 2^{16}-10000×12/10=0xd120

```
#include <reg51.h>
#define uchar unsigned char
uchar time;                  //中断次数
uchar period=250;   //1 个周期的次数
uchar high=50;               //20%高电平的次数

timer0() interrupt 1{//T0 中断函数
      TH0=0xd1;              //重装载计数初值
      TL0=0x20;
      if (++time==high) P2=0;//高电平时间到，P2 变低
      else if (time==period)
           {time=0;P2=1;}//周期时间到，P2 变高
}

void main (){
    TMOD = 0x01;  //T0 定时方式 1
    TH0 = 0xd1;            //首次装入计数初值
    TL0 = 0x20;
    EA=ET0=1;
    TR0 = 1;               //启动计数器
    do { }while (1);
}
```



实例 7　采用定时中断控制流水灯，实现每秒 1 位，自上而下循环功能（fosc=12MHz）。

分析：可以利用 20 次 50ms 的定时中断方案，计数初值为：a = 65536-50000×12/12 = 0x3cb0

仿照实例 6 做法，可以采用如下中断函数：

```
timer0() interrupt 1{    //T0中断函数
    TH0=0x3c;            //重装载计数初值
    TL0=0xb0;
    if (++time==20){
        time =0;
        P2=ledp[ledi];    //输出流水灯编码
        if (++ledi==8) ledi=0;    //刷新流水灯指针
    }
}
```

问题：中断函数内任务过多，不利于实时控制。

新方案：中断函数中仅做中断次数统计和计数初值重入，控制操作改在主函数中进行。

```
#include <reg51.h>
#define uchar unsigned char
bit ldelay=0;            //长定时溢出标记
uchar t=0;               //定时溢出次数

timer0() interrupt 1 {  //T0中断函数
    if(++t==20){t=0; ldelay=1;}//刷新长定时溢出标记
    TH0 =0x3c;TL0 =0xb0; //重置T0初值
}

void main(void) {
    uchar code ledp[8]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};
    uchar ledi;          //指示显示顺序
    TMOD=0x01;           //定义T0定时方式1
    TH0 =0x3c;TL0 =0xb0;//溢出20次=1秒（12M晶振）
    TR0=1;
    EA=ET0=1;
    while(1){
        if(ldelay) {     //发现有时间溢出标记，进入处理
            ldelay=0;    //清除标记
            P2=ledp[ledi]; //读出一个值送到P2口
            ledi++;      //指向下一个
            if(ledi==8)ledi=0;//到了最后一个灯就换到第一个
}}}
```

实例 8 测量从 P3.2（INT0）输入的正脉冲的宽度,测量结果以 BCD 码形式存放在片内 RAM 40H 开始的单元处（设 40H 地址存放个位,系统时钟为 12MHz，被测脉冲信号周期不超过 100ms）。

分析： GATE=TR0=1 时允许 INT0 的脉冲控制定时器的启停，则根据 T0 先启动、再关闭后的计数值可算出被测脉冲宽度。

```c
#include <reg51.h>
sbit P3_2=P3^2;
main(){
    unsigned char *P;
    unsigned int a;
    P=0x40;             //指针指向片内40H单元
    TMOD =0x09;         //T0定时方式1，允许INT0启动计数器
    TH0 = TL0 = 0;      //装入计数初值
    do{} while(P3_2==1);//等待INT0变低
    TR0=1;              //启动计数器（允许INT1启动计数器）
    while(P3_2==0);     //等待脉冲上升沿，上升沿启动计数器
    while(P3_2==1);     //等待脉冲下降沿，下降沿停止计数器
    TR0 = 0;            //关闭T0，防止下一个上升沿启动计数器
    a = TH0*256+TL0;//将TH0和TL0中的数合成到整形变量a中
    for(a;a!=0;){       //循环，直到a为零
        *P=a%10;        //分解a,个位存放在40单元，其它以此递增
        a=a/10;         //删除最末位
        P++;            //存放地址加1
    }
    while(1){a=0;}      //原地循环
}
```