第 17 课，写入按键次数到 24c02，并读出来显示在 4 个 LED 上。并脱机运行验证结果。

这一课我们用 24c02 完成一个实际应用的场合，在 24c02 中记录按键次数并用二机制显示在 4 个 LED 上。下次开机时，将继续显示上次的按键次数。这些工作在工控领域有十分广泛的应用。

我们将在第一次运行过后，再断点，学习 DX516 的脱机运行，并看运行结果。

————————————————————————

```
#define uchar unsigned char //定义一下方便使用
#define uint    unsigned int
#define ulong unsigned long
#include <reg52.h>          //包括一个 52 标准内核的头文件

char code dx516[3] _at_ 0x003b;//这是为了仿真设置的

#define    WriteDeviceAddress 0xa0        //定义器件在 IIC 总线中的地址
#define    ReadDviceAddress 0xa1
sbit  SCL=P2^7;
sbit  SDA=P2^6;

sbit  P10=P1^0;
sbit  K1=P3^2;

//定时函数
void DelayMs(unsigned int number)
{
      unsigned char temp;
      for(;number!=0;number--)
      {
            for(temp=112;temp!=0;temp--) ;
      }
}

//开始总线
void Start()
{
      SDA=1;
      SCL=1;
      SDA=0;
      SCL=0;
}

//结束总线
void Stop()
{
```

```
    SCL=0;
    SDA=0;
    SCL=1;
    SDA=1;
}


//发 ACK0
void NoAck()
{
    SDA=1;
    SCL=1;
    SCL=0;
}

//测试 ACK
bit TestAck()
{
    bit ErrorBit;
    SDA=1;
    SCL=1;
    ErrorBit=SDA;
    SCL=0;
    return(ErrorBit);
}

//写入 8 个 bit 到 24c02
Write8Bit(unsigned char input)
{
    unsigned char temp;
    for(temp=8;temp!=0;temp--)
    {
        SDA=(bit)(input&0x80);
        SCL=1;
        SCL=0;
        input=input<<1;
    }
}

//写入一个字节到 24c02 中
void Write24c02(uchar ch,uchar address)
{
    Start();
    Write8Bit(WriteDeviceAddress);
```

```
    TestAck();
    Write8Bit(address);
    TestAck();

    Write8Bit(ch);
    TestAck();

    Stop();
    DelayMs(10);
}

//从 24c02 中读出 8 个 bit
uchar Read8Bit()
{
    unsigned char temp,rbyte=0;
    for(temp=8;temp!=0;temp--)
    {
        SCL=1;
        rbyte=rbyte<<1;
        rbyte=rbyte|((unsigned char)(SDA));
        SCL=0;
    }
    return(rbyte);
}

//从 24c02 中读出 1 个字节
uchar Read24c02(uchar address)
{
    uchar ch;

    Start();
    Write8Bit(WriteDeviceAddress);
    TestAck();
    Write8Bit(address);
    TestAck();
    Start();
    Write8Bit(ReadDviceAddress);
    TestAck();
    ch=Read8Bit();
    NoAck();
    Stop();
    return(ch);
}
```

```
 //写入按键次数到 24c02，并读出来显示在 4 个 LED 上

void main(void)     //  主程序
{
    uchar c1,c2;

    while(1)
    {
        c1=Read24c02(0x01);     //读出 24c02 第一个地址数据
        P1=c1;                  //显示在 P1 口的 4 个 LED 上

        if(!K1)       //按键处理
        {
            c1++;             //值加 1
            Write24c02(c1,0x01);    //重新写入 24c02

            while(!K1);               //等待按键松开
            for(c2=0;c2<250;c2++); //松开按键去抖
        }
    }
}
```

————————————————————————————————————

程序中，不断读出 24c02 的 0x01 位置的数据出来，并显示在 P1 口上，我们可以在 4 个 LED 上观察到低 4 位的数据变化。

当检查到按键时，就将前面读出来的值加 1，写入在 24c02 中的同一个位置中。下一个循环中，值又被读出来并显示。

编译，联机，并运行。不断按 K1，可以看到 P1 的 4 个 LED 不断以二机制变化显示。

下面我们试验**脱机运行方式**，并验证 24c02 的非挥发特性。

我们记住目前的 led 显示状态，再将 dx516 后面 usb 取电板拔下。使 Dx516 仿真器断电。再将仿真器上旁边的一个跳线跳到“ RUN ”的位置，这时仿真器就会像一个烧好了程序的 51 芯片一样工作，这就是 DX516 的脱机运行工作方式，这种方式对特殊的调试有很大的帮助，进入脱机方式后，就会直接运行上次调试过的程序。

跳好线之后，请重新插上 usb 取电板，上电。可以看到 dx516 上面的蓝灯闪烁 3 次，表示进入了脱机运行方式，并开始全速运行上次用户调试过的程序。

我们可以直接看到 4 个 LED 的显示状况，和上次断电之前是一样的。按 K1，4 个 LED 又继续按照二机制方式加 1。程序继续正常工作。

好，试验完成，我们将脱机运行的跳线，跳回到“ EMB ”方向，以便下一次课程继续仿真运行。

作业：

修改为用 24c02 的另一个位置 0x08 保存数据。并用 DX516 脱机运行方式验证。