第3章 从做成一个用按钮控制数码管显示的数字增减来初识单片机的中断

3.1 硬件设计及连接步骤

3.1.1 硬件设计

1. 设计思路

第2章用一个按钮来控制我们自己制作的数码管让其发生数字变化，但这个按钮没有接到外部中断输入引脚，数码管也只能显示一位数字。本章我们制作的电路要用到一个4位一体的共阴数码管，我们用到的数码管是0.36in(1in=0.0254m)的，另外需要接一个按钮用来控制数字的增减，还要接一个发光二极管电路用来在按下按钮时提醒用户，但是本书只给读者提供一个按钮控制数字增加的电路和程序，两个按钮控制数字增减的电路请读者自己完成。

有的读者可能没学过原理图设计，所以在此说明一下。本书的原理图中有的用直接连线将两个或多个元器件连接起来，有的却是用网络标号，网络标号其实就是将两个或多个需要连在一起的点用两个或多个标号标出，例如图3.1中的数码管的8个段码引脚与单片机的P1口的8个引脚都是用网络符号表明连接关系的。建议读者亲自用Proteus软件绘制此电路图并进行仿真，以后再看到网络标号就能看懂了。

2. 原理图

原理图见图3.1。

3. 元器件清单

实验用元器件见表3.1。

4. 认识排阻

在硬件连接之前首先要认识一下排阻，排阻见图3.2。排阻就是将若干相同阻值的电阻集成到一块，然后将这些电阻的一端连接起来并引出一个引脚，这些电阻的另一端分别引出引脚，排阻的一端有一个标记，表示其下面的引脚是这些电阻的公共端。

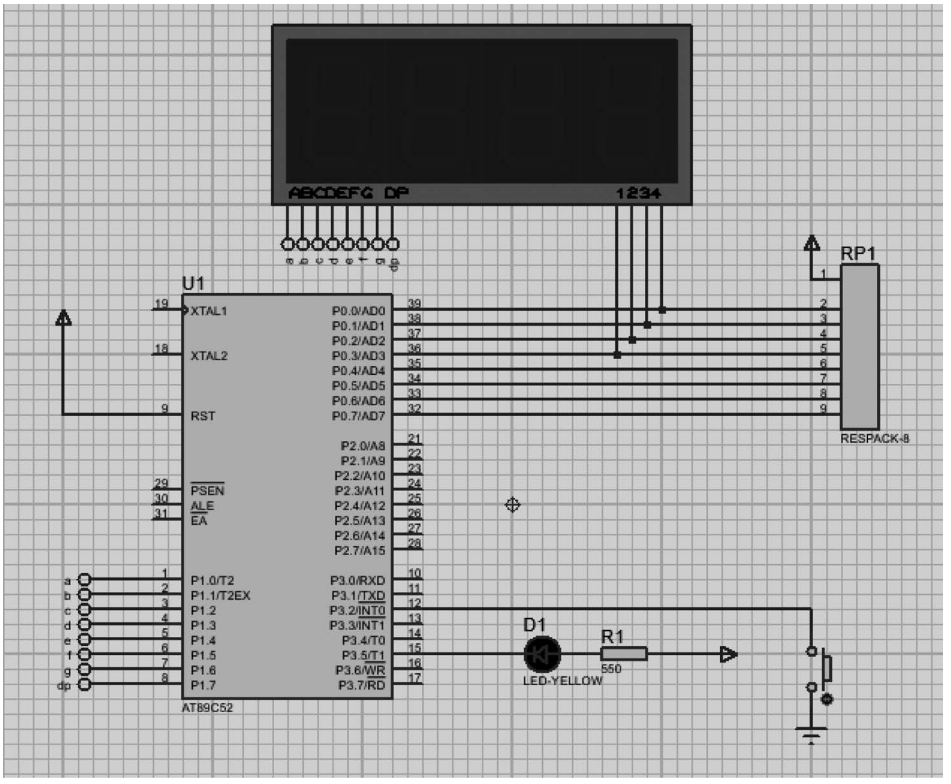

图 3.1　原理图

表 3.1　实验用元器件

序　　号	元器件名称	型号或容量	数量/个
1	单片机	STC89C52RC DIP40	1
2	晶振	12MHz	1
3	电容	30pF	2
4	数码管	4 位一体 0.36in 共阴	1
5	排阻	1kΩ 9 脚排阻	1
6	电阻	1kΩ	1
7	按钮	12×12×4.3，按键，轻触开关	2
8	发光二极管	任意颜色	1

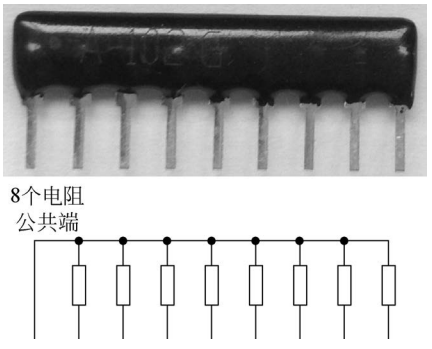

图 3.2　9 脚排阻及其内部电路

3.1.2　硬件连接步骤

4 位一体数码管的引脚排布如图 3.3 所示。这种数码管一共有 12 个引脚，分别是 8 个段选线引脚(a、b、c、d、e、f、g、dp)和 4 条位选线引脚(w1、w2、w3、w4)。

扫描如下二维码在手机或平板计算机端一边观看硬件连接和用万用表检测电路的视频，一边动手进行硬件连接，硬件连接完成后一定要用万用表检测一下硬件连接得是否可靠，如果不可靠，一定要重新连接直至可靠无误。至此整个硬件电路的安装工作结束。接下来要动手做的就是编写程序了。

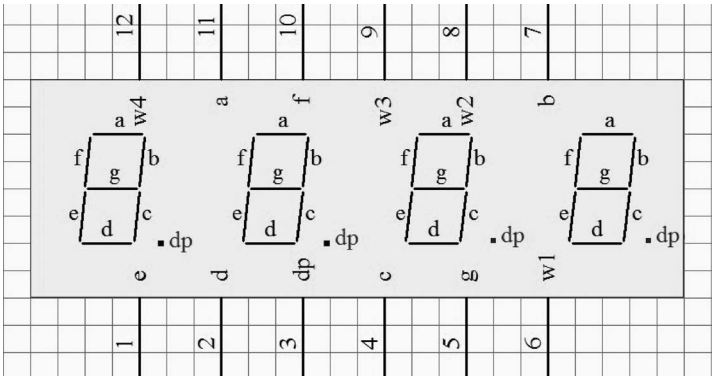

图 3.3　4 位一体数码管段排列及管脚

数码管数字加 1 电路硬件连接及运行视频

3.2　程序设计及下载

3.2.1　程序设计思路

在单片机的外部中断引脚接一个按钮，当这个按钮被按下时就会产生一个外部中断，每按下一次按钮就让一个长整型变量加 1，中断的好处是不需要 CPU 查询是否有按钮被按下，可以提高 CPU 的工作效率。

3.2.2　源程序

源程序如下：

```
#include <reg52.h>
#define u8 unsigned char
u8 d[] = {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
u8 w[] = {0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};
sbit LED = P3^5;                          //通过 P3.5 来控制发光二极管是否亮
unsigned long j = 0;                      //这个变量就是我们用按钮控制增大的那个数字

/*****************************************************************************
函数名称:        delay(u8 t)
函数功能:        产生时间延时
```

```
入口参数:        无
出口参数:        无
备 注:
*************************************************************************/
void delay(u8 t)
{
    u8 b;
    for(;t>0;t--)
        for(b=0;b<160;b++);
}
/*************************************************************************
函数名称:        display()
函数功能:        显示定时时间
入口参数:        无
出口参数:        无
备 注:
*************************************************************************/
void display(void)
{
        P1=d[j/1000];                   //通过P1口给数码管输出数字的千位数段码
        P0=w[3];                        //通过P0口让第一个数码管亮
        delay(1);
        P0=0xff;

        P1=d[j%1000/100];               //通过P1口给数码管输出百位数的段码
        P0=w[2];                        //通过P0口让第二个数码管亮
        delay(1);
     P0=0xff;

        P1=d[j%100/10];
        P0=w[1];                        //通过P0口让第三个数码管亮
        delay(1);
        P0=0xff;

        P1=d[j%10];                     //通过P2口给数码管输出个位数段码
        P0=w[0];                        //通过P0口让第四个数码管亮
        delay(1);
        P0=0xff;
}
/*************************************************************************
函数名称:        EX0_init()
函数功能:        外部中断初始化
入口参数:        无
出口参数:        无
备 注:
*************************************************************************/
void EX0_init()
{
    IT0=1;
    EX0=1;
    EA=1;
}
/*************************************************************************
```

```
函数名称:         main()
函数功能:         初始化外部中断,反复调用显示函数以显示数字
入口参数:         无
出口参数:         无
备 注:
*************************************************************************/
main()
{
    EX0_init();                          //调用外部中断初始化函数
    while(1)
    {
      display();                         //显示
  }
}
/*************************************************************************
函数名称:         EX0_int() interrupt 0
函数功能:         处理外部中断 0 发生后的事情
入口参数:         无
出口参数:         无
备 注:
*************************************************************************/
void EX0_int() interrupt 0
{
    j++;
    LED = 0;
    delay(20);
    if(j > 9999) j = 0;
    LED = 1;
}
```

3.3　初识单片机的中断

3.3.1　用按钮产生外部中断

在本章做的电路有一个按钮接到了单片机的 P3.2(即单片机的第 12 脚)。P3.2 是单片机的 0 号外部中断输入端,只要 P3.2 有一个由高电平到低电平的跳变或者直接由高电平跳到了低电平,就会产生一个外部中断。

3.3.2　单片机如何处理中断

单片机内部有一个复杂的中断逻辑电路,此电路能检测到多种中断源产生的中断并进行优先级排队。当某个中断源产生中断时,如果在程序中有该中断源允许的指令并且有总中断允许的指令,该中断源产生的中断信号就可以送达 CPU,CPU 根据此中断源对应的中断号计算出其中断服务程序的入口地址,从该地址拿到该中断服务程序的首地址并放到 PC 中,该中断服务程序就会被自动执行。关于中断的详细内容在后面的章节中会讲到,现在先来编写一个中断初始化函数和中断服务函数。

1. 中断初始化函数

中断初始化函数的任务有以下几项。

（1）设置外部中断的触发方式是低电平有效还是下降沿有效，怎么设置呢？特殊功能寄存器 TCON 的 8 位里面有 1 位名称叫 IT0，见图 3.4，若 IT0＝0，则外部中断 0 产生的中断就是低电平有效；若 IT0＝1，则外部中断 0 产生的中断只能下降沿有效。

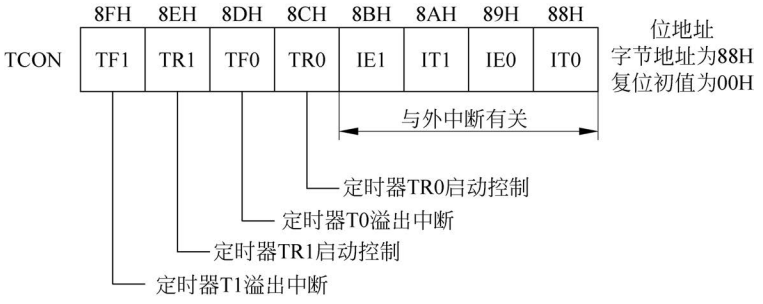

图 3.4　51 单片机的特殊功能寄存器 TCON

（2）合上外部中断的分开关。有一个名称为 IE 的特殊功能寄存器，见图 3.5，其最低位的名称叫 EX0，如果 EX0＝0，则外部中断分开关就断开，不允许外部中断 0 产生的中断信号送达 CPU；如果 EX0＝1，则外部中断分开关就被合上，允许外部中断 0 产生的中断信号送达 CPU。

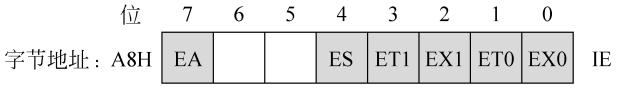

图 3.5　51 单片机的特殊功能寄存器 IE

（3）合上 CPU 总中断开关。IE 寄存器的最高位名称为 EA，如果在程序中编写 EA＝0，则即使中断源对应的中断分开关合上也不允许其中断信号送达 CPU；如果在程序中编写 EA＝1，则只要中断分开关合上就允许其中断信号送达 CPU。

下面我们就来编写本电路的外部中断 0 初始化函数。

```
void EX0_init()
{
    IT0 = 1;                          //外部中断 0 产生的中断只能下降沿有效
    EX0 = 1;                          //接通外部中断 0 的中断允许分开关
    EA = 1;                           //接通单片机中断总开关
}
```

2. 中断服务函数

中断服务函数其实就是当某个中断发生后希望 CPU 要处理的一些事情，例如图 3.1 所示的电路，当按下按钮，即外部中断 0 产生中断时，让数码管的数加 1。程序如下：

```
void EX0_int() interrupt 0
{
    j++;
    if(j > 9) j = 0;
}
```

说明：

(1) 中断服务函数的圆括号后面一定要跟上一个关键字 interrupt，此关键字后还要跟上一个数字，这个数字就是各个中断源对应的中断号，由于外部中断 0 的优先级最高，因此给其分配的中断号为 0。CPU 将中断号乘以 8 再加 3，就得到了存放中断服务程序首地址的地址。

(2) 在学习 C 语言时，除了 main()函数，其他函数都需要在某个或某几个函数中有调用它的语句，否则该函数就不会被执行，但在单片机的 C 语言程序中不需要出现中断服务函数的调用语句，所有的中断服务函数只要中断源发出了中断请求且合上该中断类型的分开关和单片机的中断总开关 EA，CPU 都会自动调用执行。本章只是概略地讲解中断，关于中断的详细知识在后面的章节还会介绍。

知识点总结

关于中断，归纳起来有以下几点。

(1) 51 单片机有 5 个中断源，按照中断的优先级顺序分别是外部中断 0、定时器 0 中断、外部中断 1、定时器 1 中断、串行中断。它们对应的中断号分别是 0、1、2、3、4。

(2) 要使用中断必须编写两个与中断有关的函数：一个是中断初始化函数；另一个是中断服务函数。中断初始化函数需要在 main()函数中调用一次；中断服务函数不需要任何函数调用，当某个中断发生时，单片机内部的 CPU 会自动根据中断号从 ROM 中找到该中断服务函数的存放地址执行。

扩展电路及创新提示

读者可以在本硬件系统的硬件基础上再增加一个按钮并将其接到外部中断 1 的输入端，然后编写程序，实现数码管显示的数字减少，但是有一个要求，减到 0 就不能再减了。