第 2-7 讲：数码管显示

1. 学习目的

1. 了解数码管分类、工作原理及驱动电路的设计。
2. 掌握 IAP15F2K61S2/IAP15W4K61S4 单片机驱动 8 位共阳数码管的动态显示的软件设计。

2. 数码管概述

数码管是一种常用的显示设备，他有着价格便宜、使用简单的特点，在各个领域被广泛的应用，如空调、电子万年历、冰箱等等。学习数码管相关的编程之前，我们有必要了解一下数码管的一些概念和操作方式。

数码管也称 LED 数码管（LED Segment Displays），其是由多个发光二极管封装在一起组成。

1. 数码管的“段”

常用的数码管有七段数码管和八段数码管，如下图所示。7 段数码管由七个条状发光二极管组成，8 段数码管由七个条状和一个点状发光二极管，即 8 段数码管比 7 段数码管多一个发光二极管单元（多一个小数点）。如果需要显示带小数的数据，应选用 8 段数码管。开发板上使用的是 8 段数码管。

图 1：8 段数码管比 7 段数码管

2. 数码管的“位”

数码管按能显示多少个“8”可分为 1 位、2 位、4 位等数码管，如下图所示。开发板上使用的是 8 位数码管。

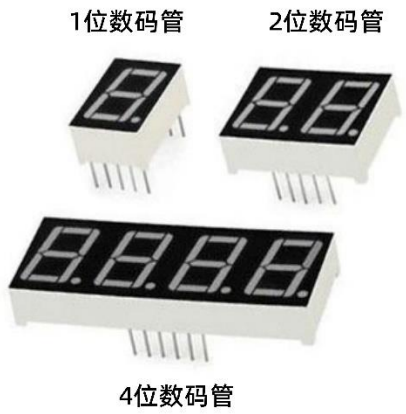

图 2：数码管

3. 共阴极和共阳极数码管

数码管按发光二极管单元连接方式分为共阴极数码管和共阳极数码管，如下图所示。

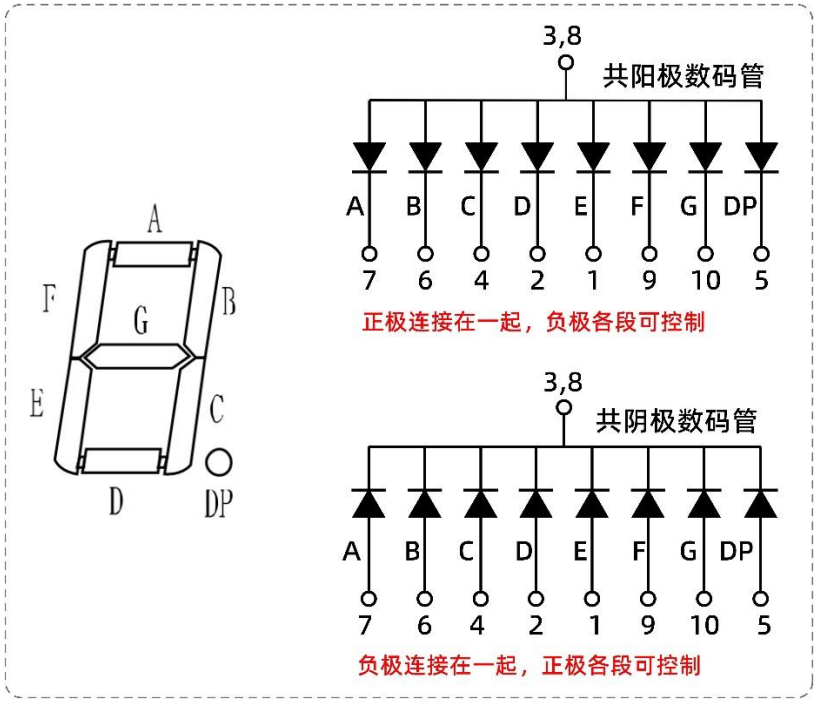

图 3：共阳极和共阴极数码管

- 共阳极：正极连接在一起，作为公共端。负极单独控制，施加低电平（逻辑 0），对应的“段”点亮，施加高电平（逻辑 1），对应的“段”熄灭。
- 共阴极：负极连接在一起，作为公共端。正极单独控制，施加低电平（逻辑 0），对应的“段”熄灭，施加高电平（逻辑 1），对应的“段”点亮。

4. 位选和段选

一般地，操作数码管时，先执行段选再执行位选。位选是选择待操作的数码管，如开发板上的是 8 位数码管，位选就是选择 8 位数码管中的某一个。段选是选择数码管里面的 LED 灯，即通过选择点亮响应的 LED 灯以达到显示需要的数据的目的。

5. 段码

数码管的段码指的是数码管在显示不同的数据时，段选信号对应的二进制数据。下图是以 8 段共阳极数码管显示数字 7 为例来描述段码。

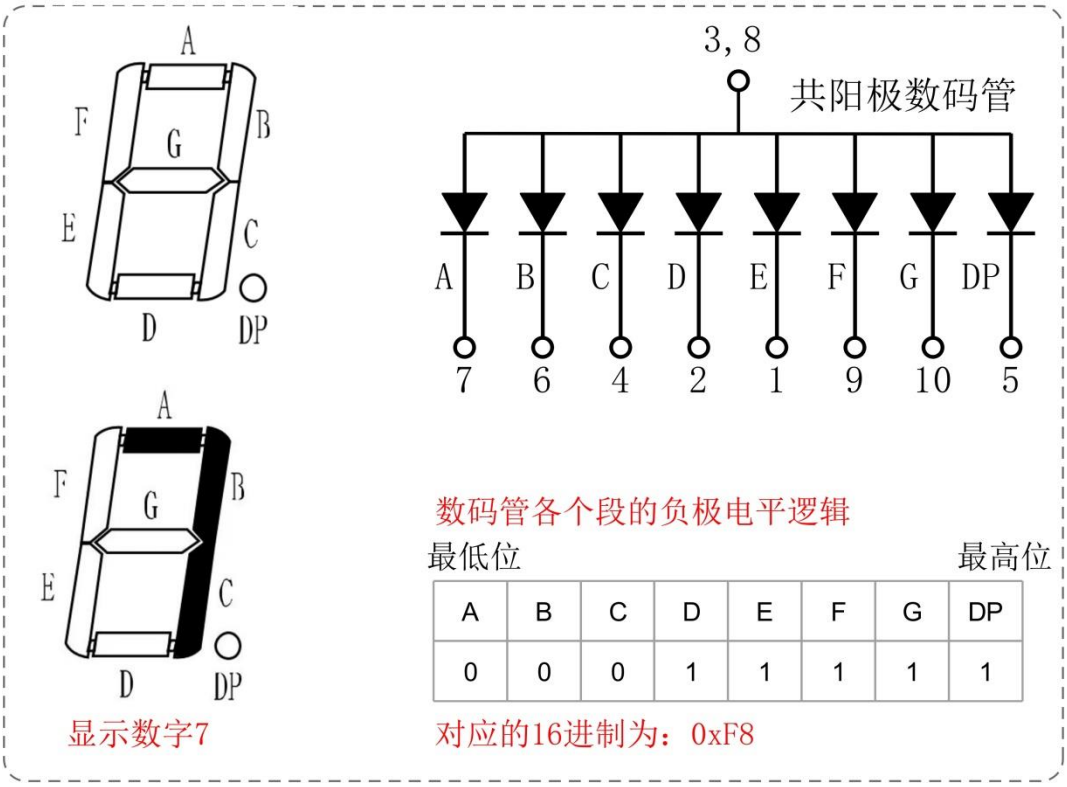

图 4：共阳极数码管显示数字 7 时的段码

对于共阳极数码管来说，段的负极为低电平“逻辑 0”时，段点亮，段的负极为高电平“逻辑 1”时，段熄灭。如要显示数字“7”，需要点亮段 A、段 B 和段 C，8 个段的负极对应的二进制数据为 11111000，换算成 16 进制即为 0xF8。如果是共阴极数码管，正好和共阳极数码管相反，8 个段的正极对应的二进制数据为 00000111，换算成 16 进制即为 0x07。

由此，我们即可得出共阳极数码管显示的段码表，如下表所示。

表 1：共阳极数码管显示的段码表

字形	A	B	C	D	E	F	G	DP	段码(共阳)
0	0	0	0	0	0	0	1	1	C0 H
1	1	0	0	1	1	1	1	1	F9 H
2	0	0	1	0	0	1	0	1	A4 H
3	0	0	0	0	1	1	0	1	B0 H
4	1	0	0	1	1	0	0	1	99 H
5	0	1	0	0	1	0	0	1	92 H
6	0	1	0	0	0	0	0	1	82 H
7	0	0	0	1	1	1	1	1	F8 H
8	0	0	0	0	0	0	0	1	80 H

9	0	0	0	0	1	0	0	1	90 H
A	0	0	0	1	0	0	0	1	88 H
B	1	1	0	0	0	0	0	1	83 H
C	0	1	1	0	0	0	1	1	C6 H
D	1	0	0	0	0	1	0	1	A1 H
E	0	1	1	0	0	0	0	1	86 H
F	0	1	1	1	0	0	0	1	8E H
小数点	1	1	1	1	1	1	1	0	7F H
不显示	1	1	1	1	1	1	1	1	FF H

6. 动态显示和静态显示

数码管的驱动显示方式有多种，大致可分为动态显示和静态显示。

动态显示其实就是利用 LED 的余辉效应和人眼的视觉暂留效应来实现的。动态显示方式外围驱动电路相对简单，占用单片机 I/O 较少，但是需要不断刷新数码管显示，因此，占用 CPU 时间较长。

静态显示是单片机发送数据之后，数据如何稳定有效的显示由外围锁存器件实现，这样可以大幅降低占用 CPU 的时间，但他会占用更多的 I/O，并且外围驱动电路也相对复杂。

3. 硬件设计

PK107D 开发板上设计了 8 位共阳极数码管显示电路，该显示电路中使用 74HC573 锁存器 U13 实现数码管的位选，74HC573 锁存器 U14 实现数码管的段选，电路原理图如下。

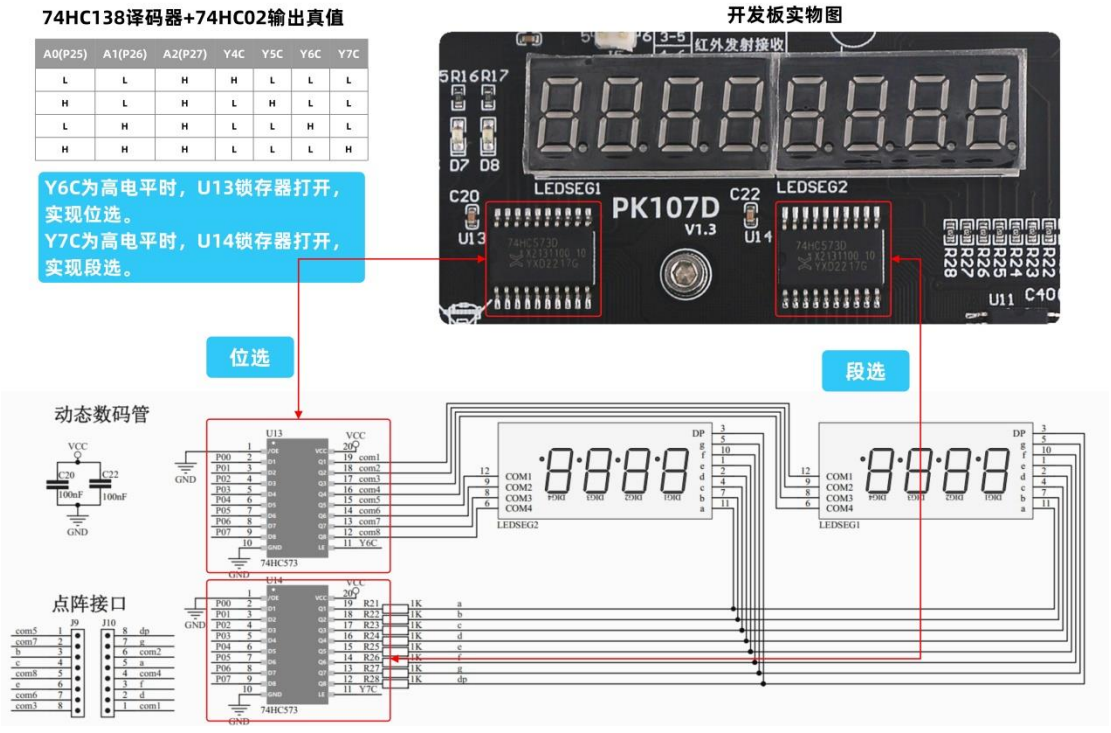

图 5：数码管显示电路

1. 数码管段选的实现

数码管的段选是通过 74HC573 锁存器 U14 实现的，P0 口的 P0.0～P0.7 连接到 8 个数码管的段选信号线上，这样，一旦锁存器 U14 使能后，可通过对 P0 口赋值向数码管发送段选信号。P0 口与数码管段选信号连接如下表所示。

表 2：数码管段选信号连接

GPIO	数码管段	备注
P0.0	A	数码管的段码最低位
P0.1	B	
P0.2	C	
P0.3	D	
P0.4	E	
P0.5	F	
P0.6	G	
P0.7	DP	数码管的段码最高位

✧ **举例**：如果选择好了位选信号，那么如果想让该位数码管显示数字 0，则需要对 P0 口这样赋值：

```
1. P0 = 0xC0;       //控制 a、b、c、d、e、f、g、dp 输出
```

2. 数码管位选的实现

数码管的位选是通过 74HC573 锁存器 U13 实现的，P0 口的 P0.0～P0.7 连接到 8 个数码管的位选信号线上，这样，一旦锁存器 U13 使能后，可通过对 P0 口赋值向数码管发送位选信号。P0 口与数码管位选信号连接示意如下：

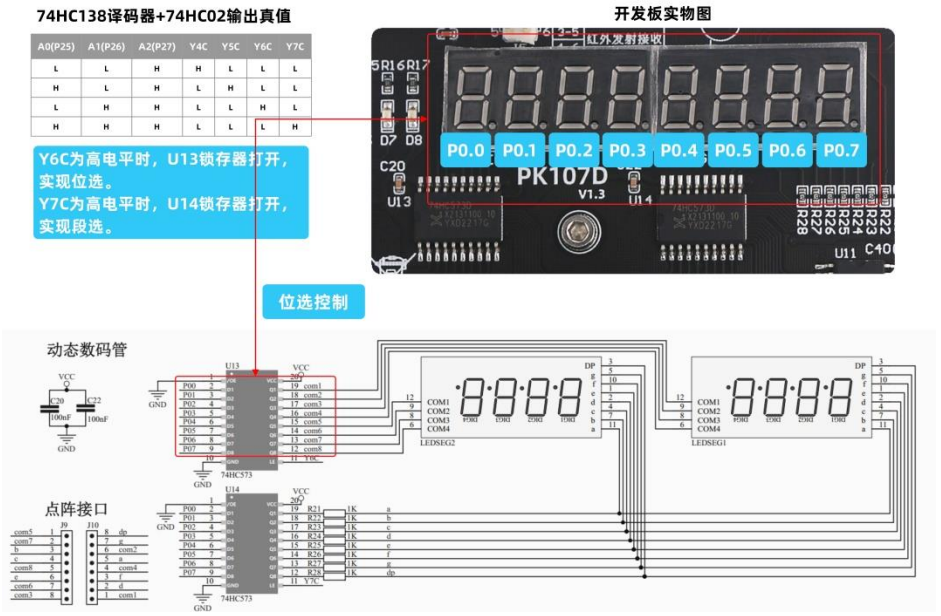

图 6：数码管位选控制

✧ **说明**：开发板 8 个数码管从左到右分别由 P0.0、P0.1、P0.2、P0.3、P0.4、P0.5、P0.6、P0.7 和 P0.8 引脚控制，在编程程序的时候需要注意到，不要把顺序搞反了。

4. 软件设计

4.1. 数码管动态显示程序结构

对于数码管动态显示来说，主要考虑的有两个方面：数码管刷新和显示数据更新。

1) 数码管刷新：因为是动态显示，所以要不断刷新数码管，利用人眼的视觉暂留效应实现数码管的显示，并且刷新的速度不能过慢，否则，显示会有闪烁。

2) 数码管显示内容的更新：8 位数码管中每位数码管都可以单独更新数据。

数码管动态显示驱动程序设计的方法很多，下面是一种基于定时器刷新的方法，供读者借鉴。

数码管驱动程序原理如下图所示，定义一个数组，该数组共有 8 个元素，分别用于保存 8 位数码管的段码，即数组中第 1 个元素用于保存 8 位数码管中第 1 位数码管的段码，第 2 个元素用于保存第 2 位数码管的段码，以此类推。

使用一个定时器用于刷新数码管显示，在定时器中断服务函数中从数组中取数码管的段码，完成对数码管显示的刷新。这样，当我们需要修改数码管显示内容时，只需要修改数组中的段码即可。

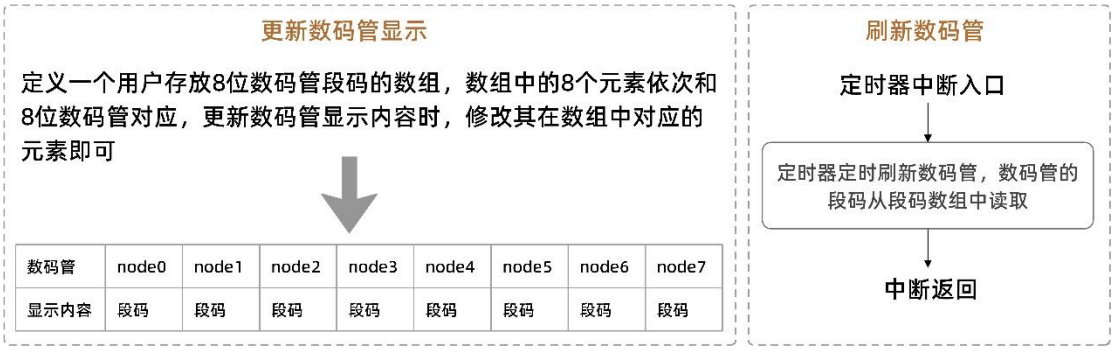

图 7：数码管软件驱动原理

定时器刷新数码管显示的流程图如下，每次进入定时器中断服务函数后刷新 8 位数码管中的一位。这里，定义一个变量“ledseg_nod”用于记录数码管的位，每次刷新后“ledseg_nod”加 1，到达 8 时，表示 8 位数码管全部刷新，“ledseg_nod”的值设置为 0，开始新一轮刷新。

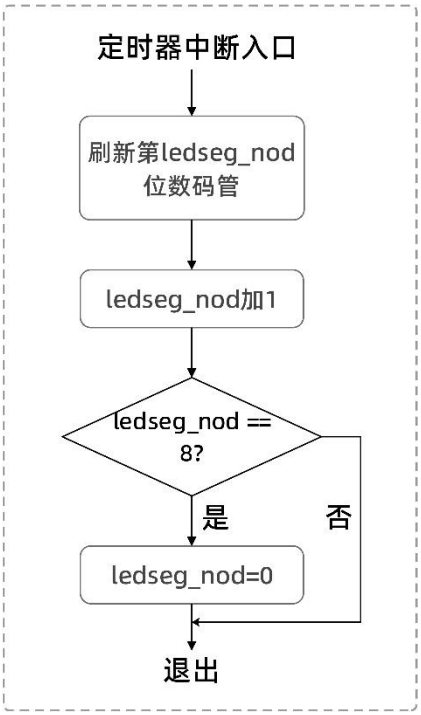

图 8：定时器刷新数码管流程

4.2. 数码管显示实验

✧ 注：本节的实验是在“实验 2-5-1：串口 1 数据收发实验”的基础上修改，本节对应的实验源码是：“实验 2-7-1：数码管显示实验”。

4.2.1. 实验内容

通过 4 个独立按键 S4~S7 控制数码管显示内容，S7 控制第 1 个和第 2 个数码管，S6 控制第 3 个和第 4 个数码管，S5 控制第 5 个和第 6 个数码管，S4 控制第 7 个和第 8 个数码管。

程序复位运行后，8 个数码管全部显示数字 0，每按动一次按键，该按键控制的数码管显示内容加 1，加到 F 后再次按动按键返回数字 0，如此反复。

4.2.2. 代码编写

1. 新建一个名称为“ledseg.c”的文件及其头文件“ledseg.h”并保存到工程的“Source”文件夹，并将“ledseg.c”加入到 Keil 工程中的“SOURCE”组。
2. 引用头文件

因为在“main.c”文件中使用了“ledseg.c”文件中的函数，所以需要引用下面的头文件“ledseg.h”。

代码清单：引用头文件

```
1.  //引用数码管的头文件
2.  #include    " ledseg.h"
```

3. 定义段码表和 8 个数码管的段码数组

数码管常用来显示数字“0~9”和字符“A~F”，他们的段码定义如下：

代码清单：段码表

```
1. //数码管段码
2. u8 code SEG8_Code[] ={
3.         0xC0, // 0
4.         0xF9, // 1
5.         0xA4, // 2
6.         0xB0, // 3
7.         0x99, // 4
8.         0x92, // 5
9.         0x82, // 6
10.        0xF8, // 7
11.        0x80, // 8
12.        0x90, // 9
13.        0x88, // A
14.        0x83, // b
15.        0xC6, // C
16.        0xA1, // d
17.        0x86, // E
18.        0x8E, // F
19.        0xBF, // -
20.        0xFF, //
21.    };
```

定义一个名称为“SEG8_DispArray”的数组，用于存放 8 个数码管显示的段码，初始化值均设置为数字“0”的段码“0xC0”，数码管刷新时从该数组读取段码，代码清单如下。

代码清单： 8 位数码管段码存放数组

```
1. //存放 8 个数码管显示的段码，初始值都为数字 0 的段码。更新数码管显示内容时，只需更新该数组中的段码即可
2. static u8 SEG8_DispArray[8] ={0xC0,0xC0,0xC0,0xC0,0xC0,0xC0,0xC0,0xC0};
```

4. 发送段选和位选信号

为了程序中操作方便，我们先用宏定义对 8 位数码管中的各个位进行编号，让数码管和数组“SEG8_DispArray”关联。

代码清单：数码管位定义

```
1.  #define  LEDSEG_1       0
2.  #define  LEDSEG_2       1
3.  #define  LEDSEG_3       2
4.  #define  LEDSEG_4       3
5.  #define  LEDSEG_5       4
6.  #define  LEDSEG_6       5
7.  #define  LEDSEG_7       6
8.  #define  LEDSEG_8       7
```

程序中先发送段码信号，8 位串行段码经过 74HC573 锁存器 U14 后并行输出完成数码管的段选，之后发送位选信号，点亮数码管，代码清单如下。

代码清单：发送段选和位选信号

```
/*******************************************************************************
功能描述：向指定的数码管发送段选和位选信号
参    数：nod[in]:数码管，取值范围：0~7
返 回 值：无
*******************************************************************************/
void LEDseg_write_data(u8 nod)
{
    u8 dat;

    dat = SEG8_DispArray[nod];  //获取段码

    //控制 Y7C 输出高电平，即打开控制 U14 的锁存器（U14:74HC573）
    P25=1;          //控制 P2.5 端口输出高电平
    P26=1;          //控制 P2.6 端口输出高电平
    P27=1;          //控制 P2.7 端口输出高电平

    //发送 8 位段码
    P0 = dat;       //控制 a、b、c、d、e、f、g、dp 输出

    LEDseg_nodeSelect(nod);    //发送数码管位选信号
}
```

发送位选信号的函数代码清单如下。

代码清单：发送位选信号函数

```
/*******************************************************************************
功能描述：发送位选信号，选择指定的数码管
参    数：nod[in]:数码管，取值范围：0~7
返 回 值：无
*******************************************************************************/
void LEDseg_nodeSelect(u8 nod)
{
    //控制 Y6C 输出高电平，即打开控制 U13 的锁存器（U13:74HC573）
    P25=0;                  //控制 P2.5 端口输出低电平
    P26=1;                  //控制 P2.6 端口输出高电平
    P27=1;                  //控制 P2.7 端口输出高电平

    P0=(0x01<<nod);         //控制对应输出 com 口输出低电平 com1~com8
}
```

5. 数码管刷新

本例中使用 Timer2 刷新数码管，Timer2 定时时间配置为 2ms，每次中断刷新一位数码管。程序中使用变量“ledseg_nod”记录数码管的位，8 位数码管一轮刷新完成后，“ledseg_nod”复位（值设置为 0），进入新一轮的刷新，代码清单如下。

代码清单：定时器 2 中断服务函数中刷新数码管

```
/*******************************************************************************
 * 描  述 : 定时器 2 中断服务函数
 * 入  参 : 无
 * 返回值 : 无
 *******************************************************************************/
void timer2_isr() interrupt 12
{
    LEDseg_write_data(ledseg_nod);      //发送段码
    ledseg_nod++;
    if(ledseg_nod == 8)ledseg_nod = 0;  //8 位数码管刷新完成，ledseg_nod 复位
}
```

6. 主函数

主函数中完成相关的初始化之后，在主循环里面调用按键扫描函数 buttons_scan()查询是否有按键按下，如果有按键按下则更新数码管显示内容，代码清单如下。

代码清单：主函数

```
/*******************************************************************************
功能描述：主函数
入口参数：无
返 回 值：int 类型
*******************************************************************************/
int main(void)
{
    u8 temp;
    u8 disp_dat1 = 0,disp_dat2 = 0,disp_dat3 = 0,disp_dat4 = 0;

    P2M1 &= 0x1F;   P2M0 |= 0xE0;     //设置 P2.5、P2.6、P2.7 为推挽输出
    P0M1 &= 0x00;   P0M0 |= 0xFF;     //设置 P0.0 ~ P0.7 为推挽输出
    P3M1 &= 0xF0;   P3M0 &= 0xF0;     //设置 P3.0 ~ P0.3 为准双向口

    SEG_off();                  //控制 8 位数码管/点阵不显示
    leds_off();                 //熄灭 D1~D8 指示灯
    ULN2003_off();              //控制 ULN2003 输出高电平，关闭蜂鸣器、继电器等
    delay_ms(10);               //延时

    timer2_init();              //定时器 2 初始化
    timer2_start();             //启动定时器 2
    EA = 1;                     //使能总中断
```

```
23.    delay_ms(200);
24.
25.    while(1)
26.    {
27.        temp = buttons_scan(0);              //获取开发板用户按键检测值，不支持连按
28.        if(temp == BUTTON1_PRESSED)          //按键 S4 按下
29.        {
30.            disp_dat1++;
31.            if(disp_dat1 >0x0F)disp_dat1 = 0;
32.            LEDseg_DispUpdata(LEDSEG_1,disp_dat1,LEDSEG_DP_OFF);//更新第 8 个数码管显示内容
33.            LEDseg_DispUpdata(LEDSEG_2,disp_dat1,LEDSEG_DP_OFF);//更新第 7 个数码管显示内容
34.        }
35.        else if(temp == BUTTON2_PRESSED)    //按键 S5 按下
36.        {
37.            disp_dat2++;
38.            if(disp_dat2 >0x0F)disp_dat2 = 0;
39.            LEDseg_DispUpdata(LEDSEG_3,disp_dat2,LEDSEG_DP_ON);//更新第 6 个数码管显示内容
40.            LEDseg_DispUpdata(LEDSEG_4,disp_dat2,LEDSEG_DP_ON);//更新第 5 个数码管显示内容
41.        }
42.        else if(temp == BUTTON3_PRESSED)    //按键 S6 按下
43.        {
44.            disp_dat3++;
45.            if(disp_dat3 >0x0F)disp_dat3 = 0;
46.            LEDseg_DispUpdata(LEDSEG_5,disp_dat3,LEDSEG_DP_OFF);//更新第 4 个数码管显示内容
47.            LEDseg_DispUpdata(LEDSEG_6,disp_dat3,LEDSEG_DP_OFF);//更新第 3 个数码管显示内容
48.        }
49.        else if(temp == BUTTON4_PRESSED)    //按键 S7 按下
50.        {
51.            disp_dat4++;
52.            if(disp_dat4 >0x0F)disp_dat4 = 0;
53.            LEDseg_DispUpdata(LEDSEG_7,disp_dat4,LEDSEG_DP_OFF);//更新第 2 个数码管显示内容
54.            LEDseg_DispUpdata(LEDSEG_8,disp_dat4,LEDSEG_DP_OFF);//更新第 1 个数码管显示内容
55.        }
56.     }
57.}
```

4.2.3. 硬件连接

本实验程序的编写都是基于 IO 模式，所以 J24 端子需要使用短路帽将该端子第 1 引脚和第 2 引脚短接，即选择为 IO 模式。同时 J6 端子需要使用短路帽将该端子第 1 引脚和第 2 引脚短接，即选择独立按键。

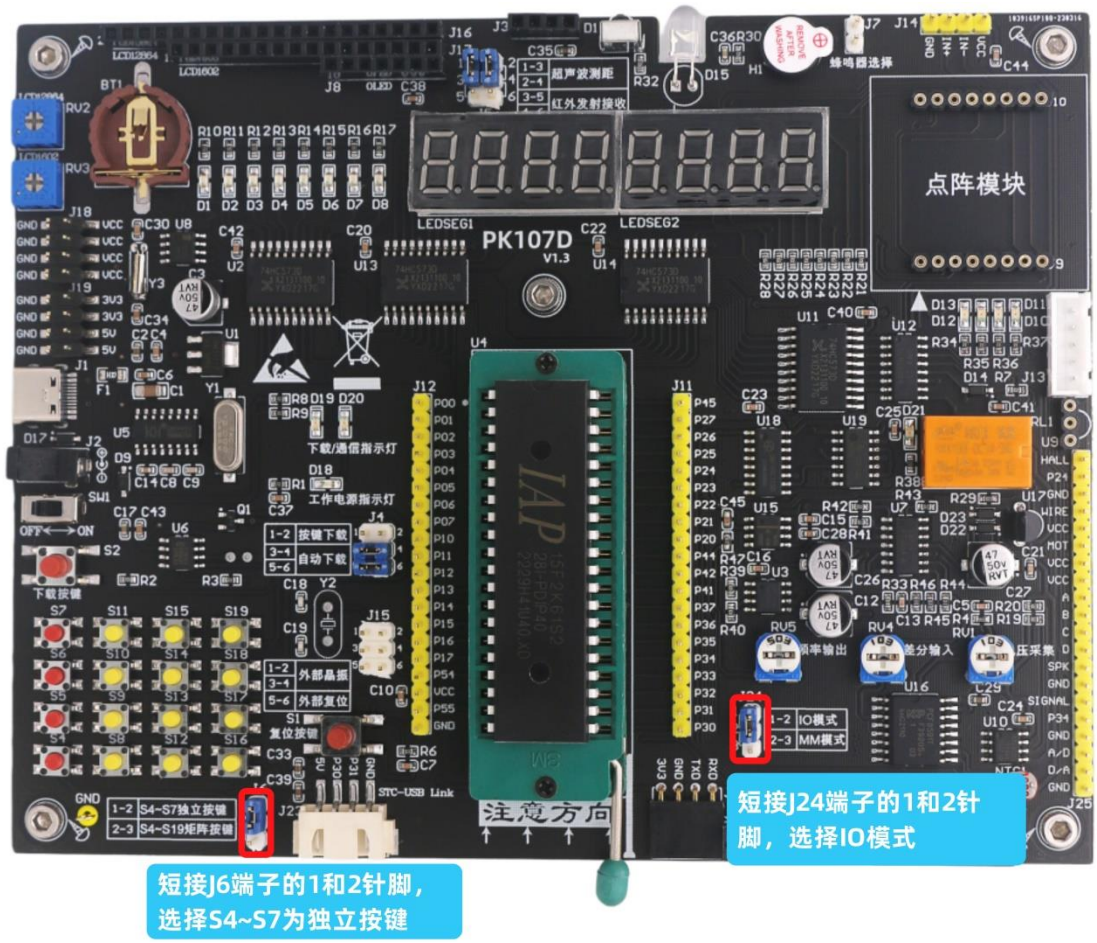

图 9：跳线帽短接

4.2.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码”目录下的压缩文件“实验 2-7-1：数码管显示实验”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”（这样做的目的是为了防止中文路径或者工程存放的路径过深导致打开工程出现问题）。
2. 双击“…\ledseg_disp\Project”目录下的工程文件“ledseg_disp.uvproj”。
3. 点击编译按钮编译工程，编译成功后生成的 HEX 文件“ledseg_disp.hex”位于工程的“…\ledseg_disp\project\Objects”目录下。
4. 打开 STC-ISP 软件下载程序，下载使用内部 IRC 时钟，IRC 频率选择：12MHz。
5. 程序运行后，依次按下独立按键 S4、S5、S6 和 S7，可以观察到每按一次按键，对应的数码管显示内容从“0~F”依次递增。
   - 按下 S7 按键：第 1、2 位数码管显示内容递增。
   - 按下 S6 按键：第 3、4 位数码管显示内容递增。
   - 按下 S5 按键：第 5、6 位数码管显示内容递增。
   - 按下 S4 按键：第 7、8 位数码管显示内容递增。