

第 2-12 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 STC8A8K64D4 系列单片机驱动 8 位共阴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。

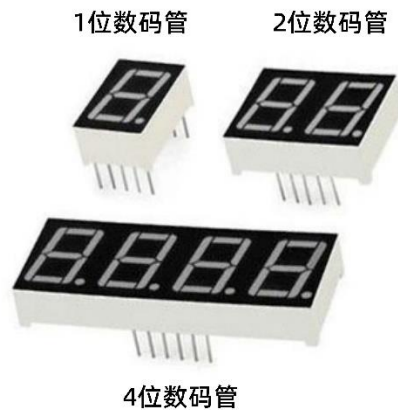


图 2：数码管

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。

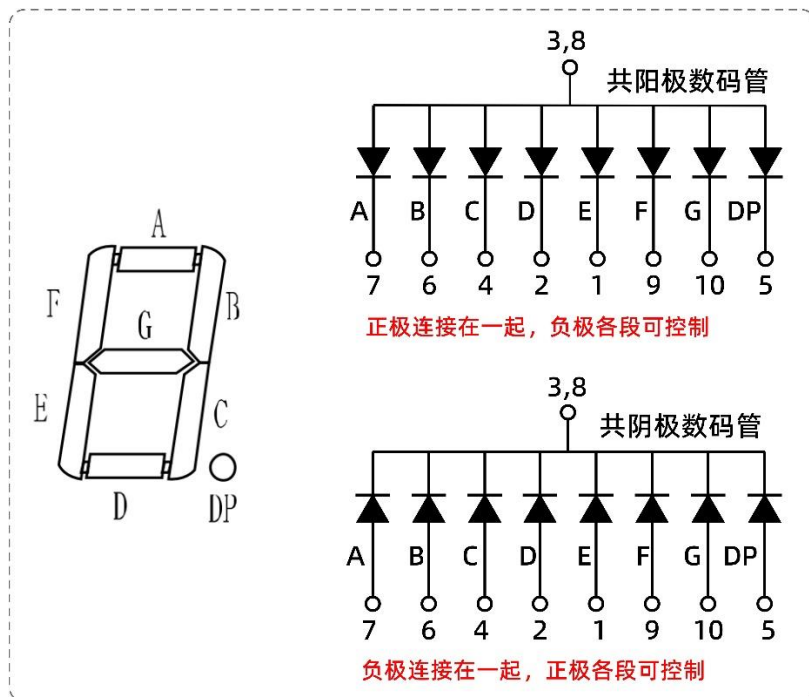


图 3：共阳极和共阴极数码管

- 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 段共阴极数码管显示数字 7 为例来描述段码。

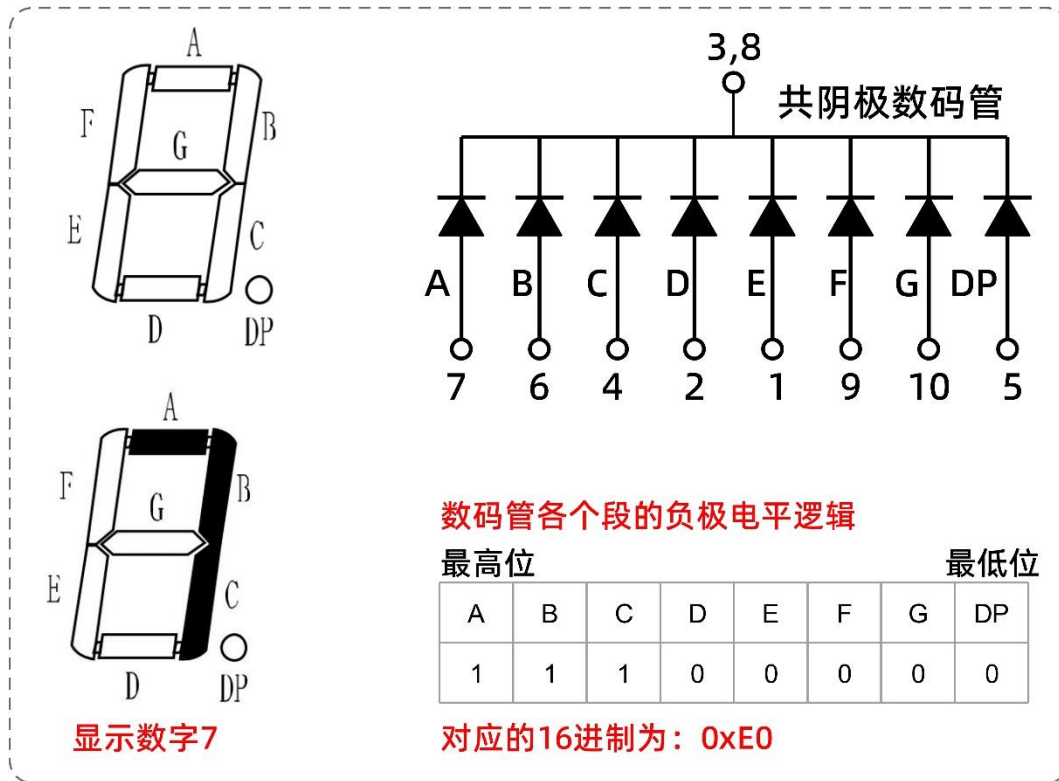


图 4：共阴极数码管显示数字 7 时的段码

对于共阴极数码管来说，段的正极为高电平“逻辑 1”时，段点亮，段的正极为低电平“逻辑 0”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 11100000，换算成 16 进制即为 0xE0。如果是共阳极数码管，正好和共阴极数码管相反，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，我们即可得出共阴极数码管显示的段码表，如下表所示。

表 1：共阴极数码管显示的段码表

字形	A	B	C	D	E	F	G	DP	段码(共阴)
0	1	1	1	1	1	1	0	0	FC H
1	0	1	1	0	0	0	0	0	60 H
2	1	1	0	1	1	0	1	0	DA H
3	1	1	1	1	0	0	1	0	F2 H
4	0	1	1	0	0	1	1	0	66 H
5	1	0	1	1	0	1	1	0	B6 H
6	1	0	1	1	1	1	1	0	BE H
7	1	1	1	0	0	0	0	0	E0 H
8	1	1	1	1	1	1	1	0	FE H
9	1	1	1	1	0	1	1	0	F6 H

A	1	1	1	0	1	1	1	0	EE H
B	0	0	1	1	1	1	1	0	3E H
C	1	0	0	1	1	1	0	0	9C H
D	0	1	1	1	1	0	1	0	7A H
E	1	0	0	1	1	1	1	0	9E H
F	1	0	0	0	1	1	1	0	8E H
小数点	0	0	0	0	0	0	0	1	01 H
不显示	0	0	0	0	0	0	0	0	00 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

IK-64D4 开发板上设计了 8 位共阴极数码管显示电路，该显示电路中使用 74HC138 译码器实现数码管的位选，74HC595 芯片实现数码管的段选，电图原理图如下。

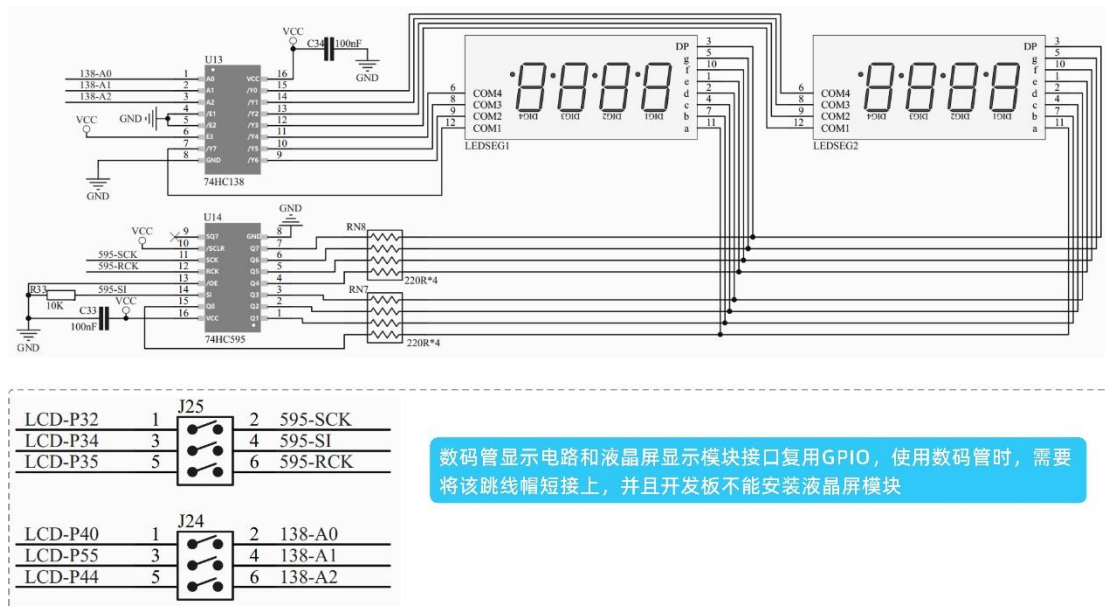


图 5：数码管显示电路

数码管显示电路占用的 STC8A8K64D4 的引脚如下表：

表 2：引脚分配

名称	引脚	说明
SCK	P3.2	和显示屏接口共用
SI	P3.4	和显示屏接口共用
RCK	P3.5	和显示屏接口共用
A0	P4.0	和显示屏接口共用
A1	P5.5	和显示屏接口共用
A2	P4.4	和显示屏接口共用

1. 数码管段选的实现

数码管的段选是通过 74HC595 芯片实现的，74HC595 是一个 8 位串行输入、并行输出的位移寄存器。数码管显示电路中，为了节省 GPIO，74HC595 的使能控制端在硬件上设置为恒有效（/OE 连接到 GND，恒为低电平）。74HC595 的 Q0~Q7 八位数据并行输出端并接到 8 个数码管的段选信号线上，这样，使用 3 个 GPIO 就可以向数码管发送段选信号。这 3 个 GPIO 各自的作用如下表所示。

表 3：74HC595 驱动信号

GPIO	连接（74HC595）	描述
P3.4	SI	发送串行数码管段选数据，经 74HC595 转为 8 位并行数据向数码管提供段选信号。
P3.2	SCK	产生数据移位时钟，上升沿时数据寄存器的数据移位，Q0→Q1→Q2→Q3→...→Q7，下降沿移位寄存器数据不变。
P3.5	RCK	产生数据锁存时钟，上升沿时将 8 位锁存移位寄存器中的状态值并行输出，下降沿时存储寄存器数据不变。

传输段码时，单片机的 P3.4 输出 8 位串行段码数据，每传输一个数据位，P3.2 产生一个上升沿时钟将数据移位，8 个上升沿移位时钟后，8 位段码数据传输完成，之后，P3.5 产生一个上升沿，段码数据并行输出，由此，完成数码管的段选。

2. 数码管位选的实现

数码管显示电路使用 74HC138 译码器产生位选信号，同样，为了节省 GPIO，74HC138 的使能控制端在硬件上均设置为恒有效（E1、E2 连接 GND，恒为低电平；E3 连接 VCC，恒为高电平）。因此，程序中操作数码管时只需控制三个地址数据输入端（A0、A1、A2）去选中 8 个数码管中的待操作的数码管即可。

三个地址数据输入端（A0、A1、A2）是如何选中 8 个数码管中的一个的，这就要看 74HC138 译码器的真值表了（74HC138 译码器的真值表如下图所示）。

输入						输出							
$\overline{E1}$	$\overline{E2}$	E3	A0	A1	A2	$\overline{Y0}$	$\overline{Y1}$	$\overline{Y2}$	$\overline{Y3}$	$\overline{Y4}$	$\overline{Y5}$	$\overline{Y6}$	$\overline{Y7}$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

H: 高电平 L: 低电平 X: 任意电平

图 6: 74HC138 真值表

由 74HC138 译码器的真值表可以看出，当 74HC138 使能后，A0、A1、A2 的输入电平决定了 Y0~Y7 的输出（低电平有效），而且从 Y0 到 Y7 是和 A2、A1、A0 的值对应的，因此，当我们需要 Y0~Y7 中某个输出端输出低电平时，输入端 A2、A1、A0 输入对应的数值即可，如需要 Y1 输出低电平，则 A2、A1、A0 输入为 001（16 进制：0x01）即可。电路中的 8 个数码管是共阴极数码管，8 个数码管的公共端依次连接到了 Y7~Y0，由此，通过 A0、A1、A2 的“逻辑值”即可完成对待操作的数码管的位选。

❖ **说明：**开发板为了布线方便，8 个数码管中的第 1 位连接到 Y7，第 2 位连接到 Y6，以此类推，这一点，在编程程序的时候需要注意一下。

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。

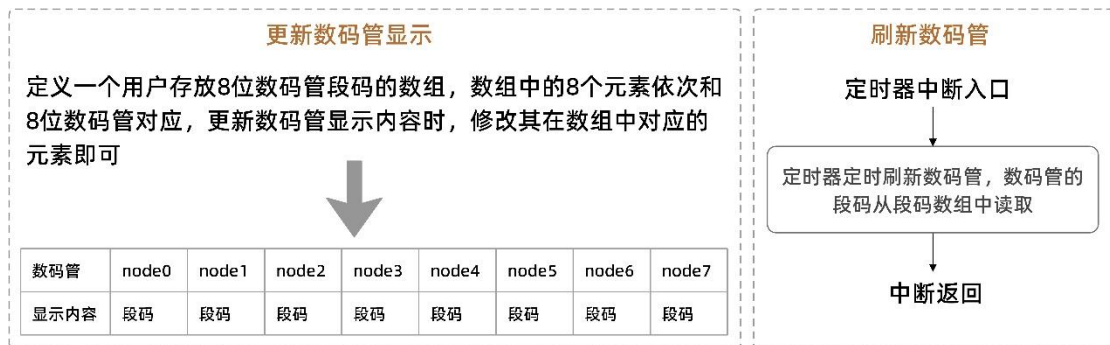


图 7：数码管软件驱动原理

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。

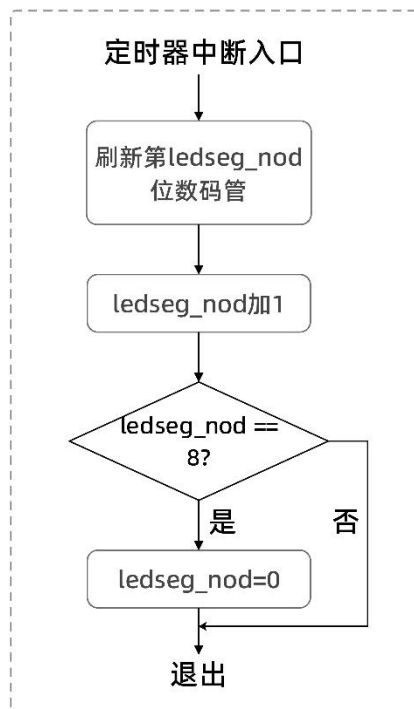


图 8：定时器刷新数码管流程

4.2. 数码管显示实验

✧ 注：本节的实验是在“实验 2-6-1：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 2-12-1：数码管显示实验”。

4.2.1. 实验内容

通过 4 个轻触按键 KEY1~KEY4 控制数码管显示内容，KEY1 控制第 1 个和第 2 个数码

管，KEY2 控制第 3 个和第 4 个数码管，KEY3 控制第 5 个和第 6 个数码管，KEY4 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 SEG8_Code[] = {
3.     0xFC, // 0
4.     0x60, // 1
5.     0xDA, // 2
6.     0xF2, // 3
7.     0x66, // 4
8.     0xB6, // 5
9.     0xBE, // 6
10.    0xE0, // 7
11.    0xFE, // 8
12.    0xF6, // 9
13.    0xEE, // A
14.    0x3E, // b
15.    0x9C, // C
16.    0x7A, // d
17.    0x9E, // E
18.    0x8E, // F
19. };
```

定义一个名称为“SEG8_DisplayArray”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“0xFC”，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 位数数码管段码存放数组

1. //存放 8 个数数码管显示的段码，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. u8 SEG8_DisgArray[8] = {0xFC,0xFC,0xFC,0xFC,0xFC,0xFC,0xFC,0xFC};

4. 数码管初始化

数码管初始化主要包括引脚配置和数码管初始显示状态的设置，代码清单如下。这里要注意的是，如果需要改动数码管初始显示内容，修改数组 SEG8_DisgArray 中的值即可，默认值是显示数字“0”。

代码清单：数码管初始化

```
1.  /*****
2.  功能描述：始化数码管驱动电路所用的 GPIO
3.  参    数：无
4.  返 回 值：无
5.  *****/
6.  void LEDseg_init(void)
7.  {
8.      P3M1 &= 0xCB; P3M0 |= 0x34;          //设置 P3.2,P3.4, P3.5 为强推挽输出
9.      P4M1 &= 0xEE;   P4M0 &= 0xEE;        //设置 P4.0, P4.4 为准双向口
10.     P5M1 &= 0xDF;   P5M0 &= 0xDF;        //设置 P5.5 为准双向口
11.
12.     SEG_A0=1;                          //74HC138 第 1 引脚 A0 置高电平
13.     SEG_A1=1;                          //74HC138 第 2 引脚 A1 置高电平
14.     SEG_A2=1;                          //74HC138 第 3 引脚 A2 置高电平
15.     SEG_LCLK=1;                        //74HC595 第 12 引脚 LCLK 置高电平
16.     SEG_SCK=0;                         //74HC595 第 11 引脚 SLK 置低电平
17.     SEG_DATA=0;                       //74HC595 第 14 引脚 DATA 置低电平
18. }
```

5. 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 位数数码管中的各个位进行编号，让数码管和数组“SEG8_DisgArray”关联。这里要注意一下，开发板为了布线方便，8 位数数码管是反序连接到 74HC138 的输出“Y0~Y7”的。

代码清单：数码管位定义

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

程序中先发送段码信号，8位串行段码经过74HC595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

```
1.  /*****
2.  功能描述：向指定的数码管发送段选和位选信号
3.  参    数：nod[in]:数码管，取值范围：0~7
4.  返 回 值：无
5.  *****/
6.  void LEDseg_write_data(u8 nod)
7.  {
8.      u8 dat,i;
9.
10.     dat = SEG8_DispArray[nod]; //获取段码
11.     for(i=0;i<8;i++)           //循环发送8位段码
12.     {
13.         SEG_DATA=(dat>>i)&0x01;
14.         SEG_SCK=0;
15.         Delay10us();
16.         SEG_SCK=1;
17.     }
18.     LEDseg_nodeSelect(nod); //发送数码管位选信号
19. }
```

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

```
1.  /*****
2.  功能描述：发送位选信号，选择指定的数码管
3.  参    数：nod[in]:数码管，取值范围：0~7
4.  返 回 值：无
5.  *****/
6.  void LEDseg_nodeSelect(u8 nod)
7.  {
8.      SEG_A0=  nod&0x01;
9.      SEG_A1= (nod&0x02)>>1;
10.     SEG_A2= (nod&0x04)>>2;
11. }
```

6. 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

```
1.  /*****
```

```
2.  * 描 述 : 定时器 2 中断服务函数
3.  * 入 参 : 无
4.  * 返回值 : 无
5.  *****/
6. void timer2_isr() interrupt 12
7. {
8.     LEDseg_write_data(ledseg_nod);    //发送段码
9.     LEDseg_Refresh();                //发送
10.    ledseg_nod++;
11.    if(ledseg_nod == 8)ledseg_nod = 0; //8 位数数码管刷新完成, ledseg_nod 复位
12. }
```

7. 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

```
1.  /*****
2.  功能描述: 主函数
3.  入口参数: 无
4.  返回值: int 类型
5.  *****/
6. int main(void)
7. {
8.     u8 temp;
9.     u8 disp_dat1 = 0, disp_dat2 = 0, disp_dat3 = 0, disp_dat4 = 0;
10.    //省略了初始化相关的代码
11.    while(1)
12.    {
13.        temp = buttons_scan(0);    //获取开发板用户按键检测值, 不支持连接
14.        if(temp == BUTTON1_PRESSED) //按键 KEY1 按下
15.        {
16.            led_toggle(LED_1);    //用户指示灯 D1 状态翻转
17.            disp_dat1++;
18.            if(disp_dat1 > 0x0F) disp_dat1 = 0;
19.            LEDseg_DisUpdata(LEDSEG_1, disp_dat1, LEDSEG_DP_OFF); //更新第 1 个数码管显示内容
20.            LEDseg_DisUpdata(LEDSEG_2, disp_dat1, LEDSEG_DP_OFF); //更新第 2 个数码管显示内容
21.        }
22.        else if(temp == BUTTON2_PRESSED) //按键 KEY2 按下
23.        {
24.            led_toggle(LED_2);    //用户指示灯 D2 状态翻转
25.            disp_dat2++;
26.            if(disp_dat2 > 0x0F) disp_dat2 = 0;
27.            LEDseg_DisUpdata(LEDSEG_3, disp_dat2, LEDSEG_DP_ON); //更新第 3 个数码管显示内容
```

```

28.     LEDseg_DisUpdata(LEDSEG_4,disp_dat2,LEDSEG_DP_ON);//更新第 4 个数码管显示内容
29. }
30. else if(temp == BUTTON3_PRESSED) //按键 KEY3 按下
31. {
32.     led_toggle(LED_3);           //用户指示灯 D3 状态翻转
33.     disp_dat3++;
34.     if(disp_dat3 >0x0F)disp_dat3 = 0;
35.     LEDseg_DisUpdata(LEDSEG_5,disp_dat3,LEDSEG_DP_OFF);//更新第 5 个数码管显示内容
36.     LEDseg_DisUpdata(LEDSEG_6,disp_dat3,LEDSEG_DP_OFF);//更新第 6 个数码管显示内容
37. }
38. else if(temp == BUTTON4_PRESSED) //按键 KEY4 按下
39. {
40.     led_toggle(LED_4);           //用户指示灯 D3 状态翻转
41.     disp_dat4++;
42.     if(disp_dat4 >0x0F)disp_dat4 = 0;
43.     LEDseg_DisUpdata(LEDSEG_7,disp_dat4,LEDSEG_DP_OFF);//更新第 7 个数码管显示内容
44.     LEDseg_DisUpdata(LEDSEG_8,disp_dat4,LEDSEG_DP_OFF);//更新第 8 个数码管显示内容
45. }
46. }
47. }

```

4.2.3. 硬件连接

按照下图所示短接跳线帽，因为数码管和显示屏复用了 GPIO，因此，使用数码管时不能安装显示屏。

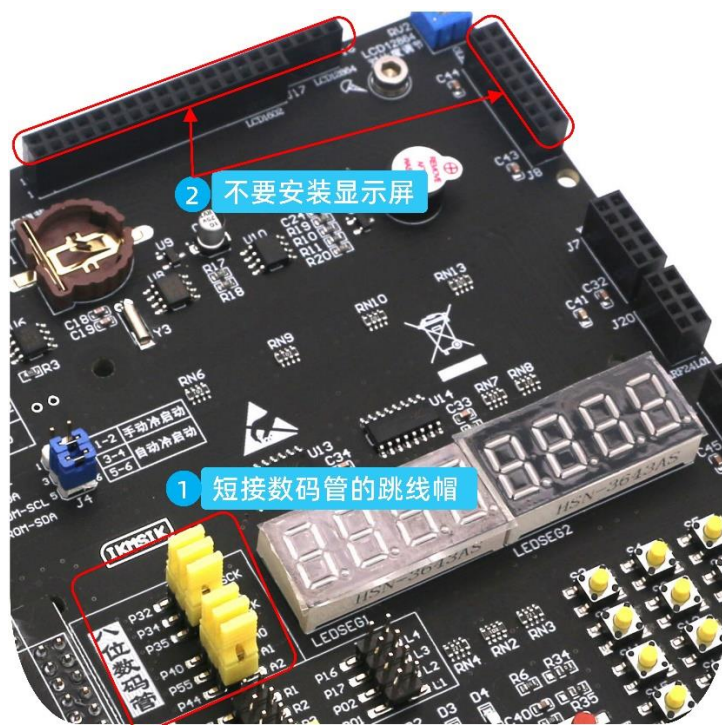


图 9：跳线帽短接

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 2-12-1：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:\STC8”（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：24MHz。
5. 程序运行后，依次按下用户按键 KEY1、KEY2、KEY3 和 KEY4，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 KEY1 按键：第 1、2 位数码管显示内容递增。
 - 按下 KEY2 按键：第 3、4 位数码管显示内容递增。
 - 按下 KEY3 按键：第 5、6 位数码管显示内容递增。
 - 按下 KEY4 按键：第 7、8 位数码管显示内容递增。