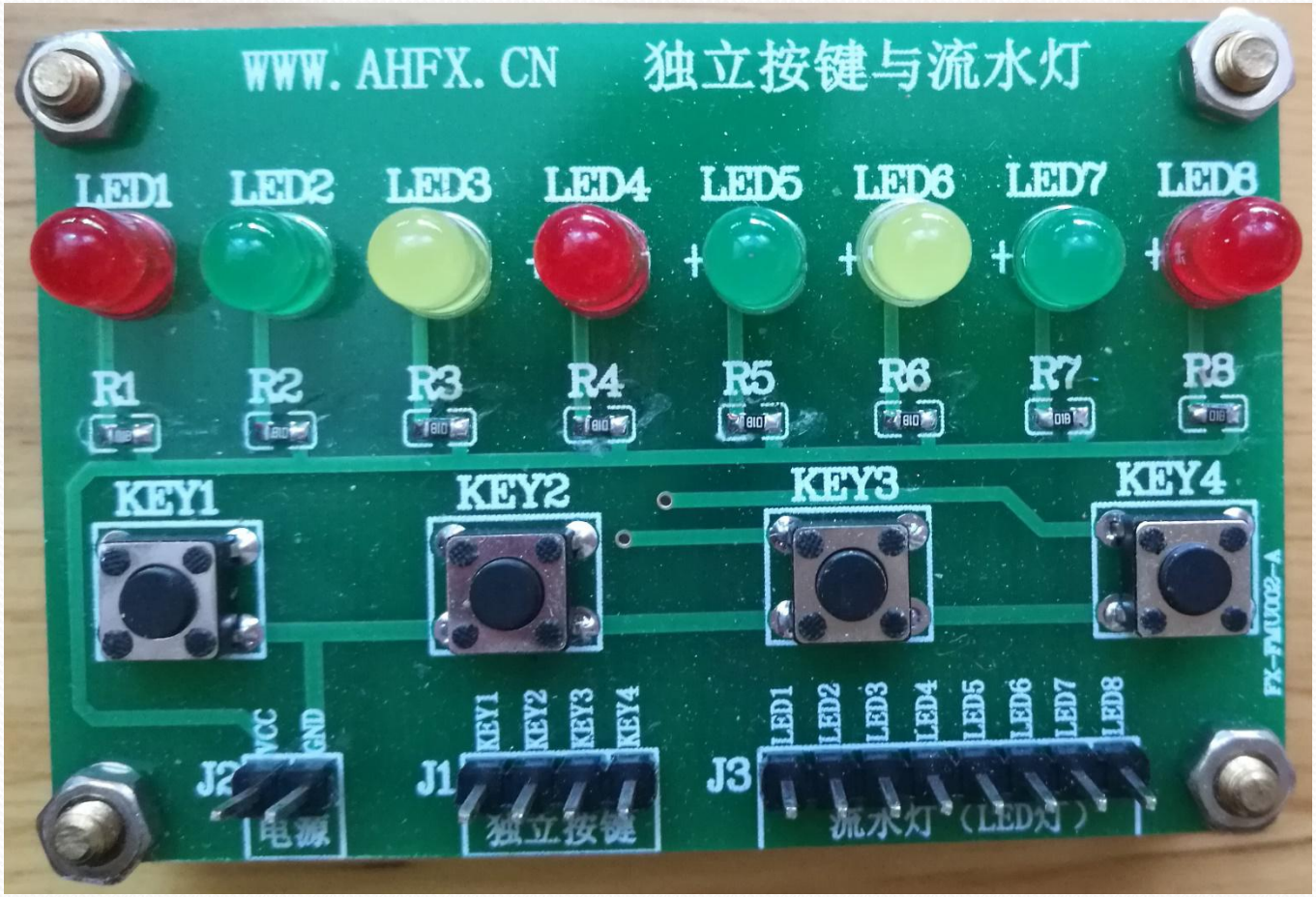
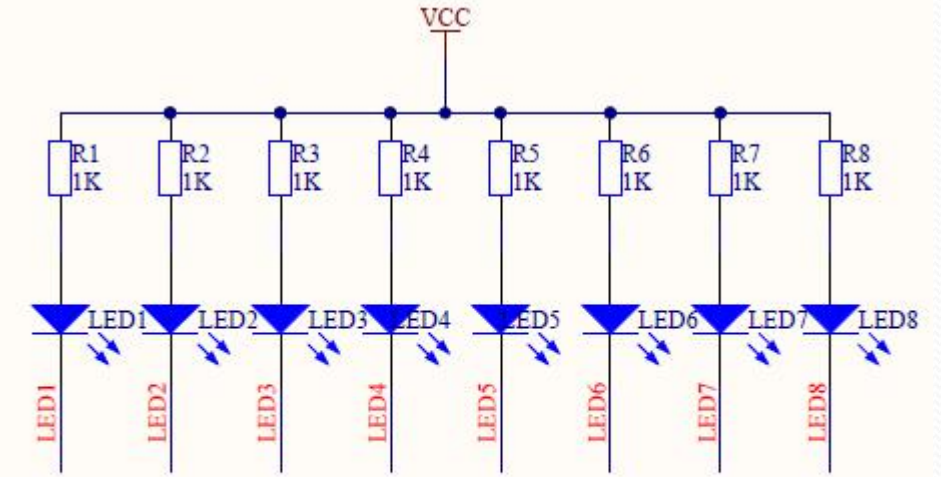
通过GPIO操纵常用输出设备

- 流水灯操纵
- 8段数码管操纵
- 8*8点阵操纵

流水灯模块

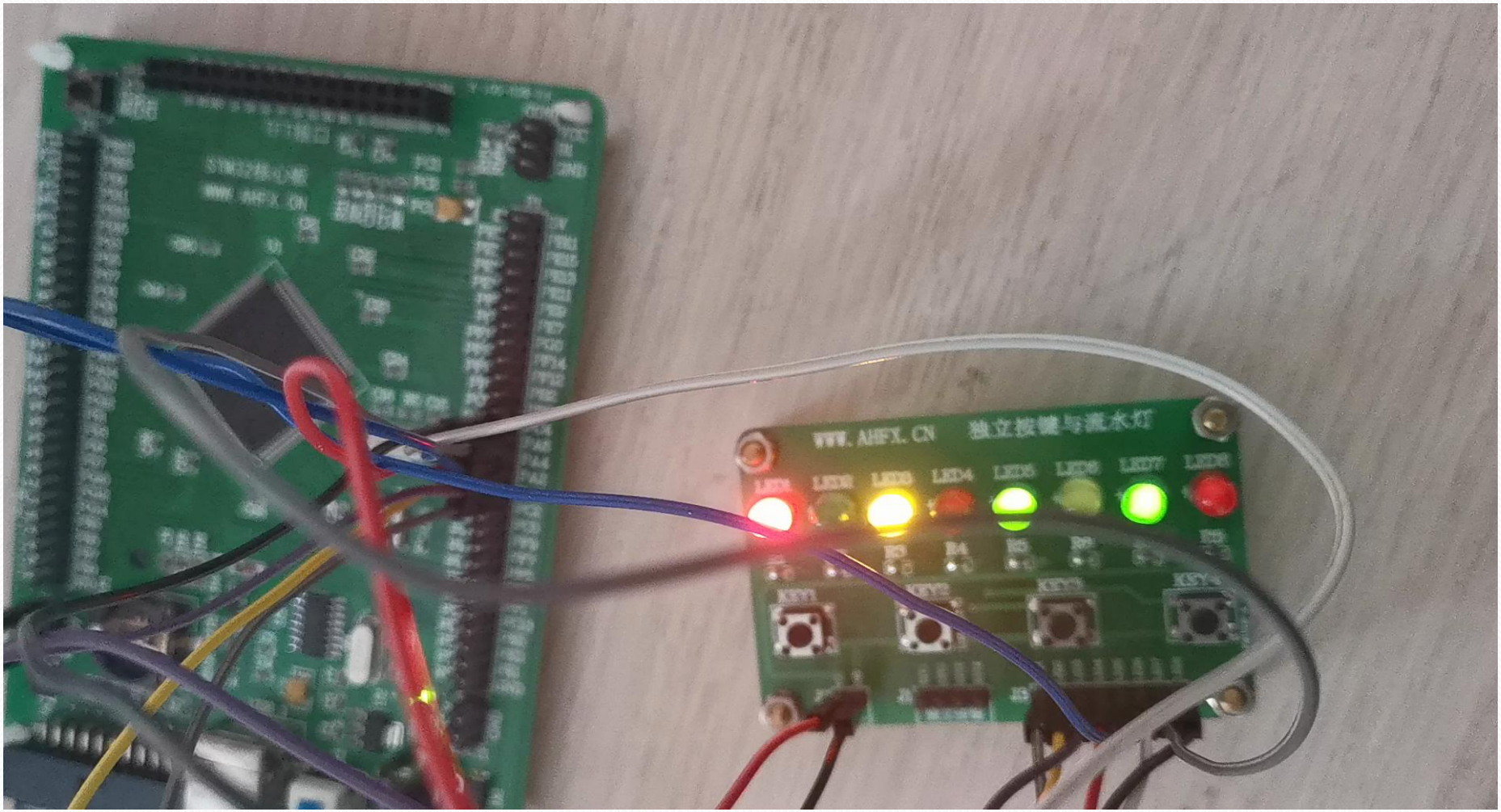

WWW.AHFX.CN
独立按键与流水灯
LED1
LED2
LED3
LED4
LED5
LED6
LED7
LED8
R1
R2
R3
R4
R5
R6
R7
R8
KEY1
KEY2
KEY3
KEY4
J2
VCC
GND
电源
J1
KEY1
KEY2
KEY3
KEY4
独立按键
J3
LED1
LED2
LED3
LED4
LED5
LED6
LED7
LED8
流水灯（LED灯）



8路流水灯

- **流水灯工作原理：**将 8 位流水灯 LED1--LED8 的阴极与单片机的 I/O 口端口（如PA口）相连，当单片机 端口相应的位输出低电平（与共阳接法相对应）时，对应LED 灯被点亮；反之则熄灭。

举例：8路流水灯分两组，间隔点亮，交替轮换



硬件原理

- 设定PA口用作输出口，选择PA0控制LED1，PA1控制LED2，...,PA7控制LED8
- 对于共阳接法，PA0=0，则LED1点亮，PA1=1，则LED2熄灭

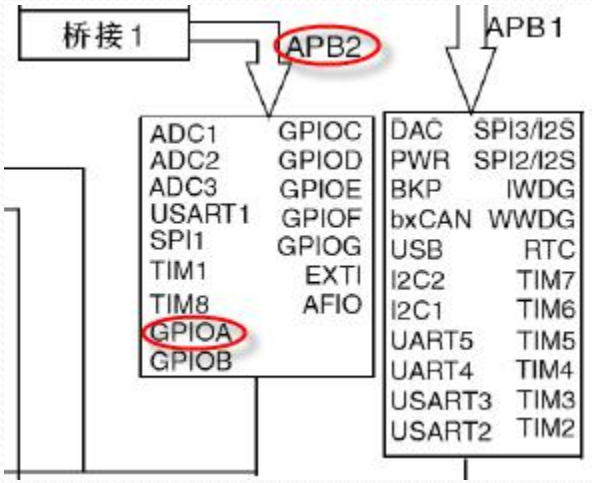
相关库函数

Table 179. GPIO 库函数

函数名	描述
GPIO_DeInit	将外设 GPIOx 寄存器重设为缺省值
GPIO_AFIODeInit	将复用功能（重映射事件控制和 EXTI 设置）重设为缺省值
GPIO_Init	根据 GPIO_InitStruct 中指定的参数初始化外设 GPIOx 寄存器
GPIO_StructInit	把 GPIO_InitStruct 中的每一个参数按缺省值填入
GPIO_ReadInputDataBit	读取指定端口管脚的输入
GPIO_ReadInputData	读取指定的 GPIO 端口输入
GPIO_ReadOutputDataBit	读取指定端口管脚的输出
GPIO_ReadOutputData	读取指定的 GPIO 端口输出
GPIO_SetBits	设置指定的数据端口位
GPIO_ResetBits	清除指定的数据端口位
GPIO_WriteBit	设置或者清除指定的数据端口位
GPIO_Write	向指定 GPIO 数据端口写入数据
GPIO_PinLockConfig	锁定 GPIO 管脚设置寄存器
GPIO_EventOutputConfig	选择 GPIO 管脚用作事件输出
GPIO_EventOutputCmd	使能或者失能事件输出
GPIO_PinRemapConfig	改变指定管脚的映射
GPIO_EXTILineConfig	选择 GPIO 管脚用作外部中断线路

Table 337. RCC 库函数

函数名	描述
RCC_DeInit	将外设 RCC 寄存器重设为缺省值
RCC_HSEConfig	设置外部高速晶振（HSE）
RCC_WaitForHSEStartUp	等待 HSE 起振
RCC_AdjustHSICalibrationValue	调整内部高速晶振（HSI）校准值
RCC_HSICmd	使能或者失能内部高速晶振（HSI）
RCC_PLLConfig	设置 PLL 时钟源及倍频系数
RCC_PLLCmd	使能或者失能 PLL
RCC_SYSCLKConfig	设置系统时钟（SYSCLK）
RCC_GetSYSCLKSource	返回用作系统时钟的时钟源
RCC_HCLKConfig	设置 AHB 时钟（HCLK）
RCC_PCLK1Config	设置低速 AHB 时钟（PCLK1）
RCC_PCLK2Config	设置高速 AHB 时钟（PCLK2）
RCC_ITConfig	使能或者失能指定的 RCC 中断
RCC_USBCLKConfig	设置 USB 时钟（USBCLK）
RCC_ADCCLKConfig	设置 ADC 时钟（ADCCLK）
RCC_LSEConfig	设置外部低速晶振（LSE）
RCC_LSICmd	使能或者失能内部低速晶振（LSI）
RCC_RTCCLKConfig	设置 RTC 时钟（RTCCLK）
RCC_RTCCLKCmd	使能或者失能 RTC 时钟
RCC_GetClocksFreq	返回不同片上时钟的频率
RCC_AHBPeriphClockCmd	使能或者失能 AHB 外设时钟
RCC_APB2PeriphClockCmd	使能或者失能 APB2 外设时钟
RCC_APB1PeriphClockCmd	使能或者失能 APB1 外设时钟
RCC_APB2PeriphResetCmd	强制或者释放高速 APB（APB2）外设复位
RCC_APB1PeriphResetCmd	强制或者释放低速 APB（APB1）外设复位
RCC_BackupResetCmd	强制或者释放后备域复位
RCC_ClockSecuritySystemCmd	使能或者失能时钟安全系统
RCC_MCOConfig	选择在 MCO 管脚上输出的时钟源
RCC_GetFlagStatus	检查指定的 RCC 标志位设置与否
RCC_ClearFlag	清除 RCC 的复位标志位
RCC_GetITStatus	检查指定的 RCC 中断发生与否
RCC_ClearITPendingBit	清除 RCC 的中断待处理位



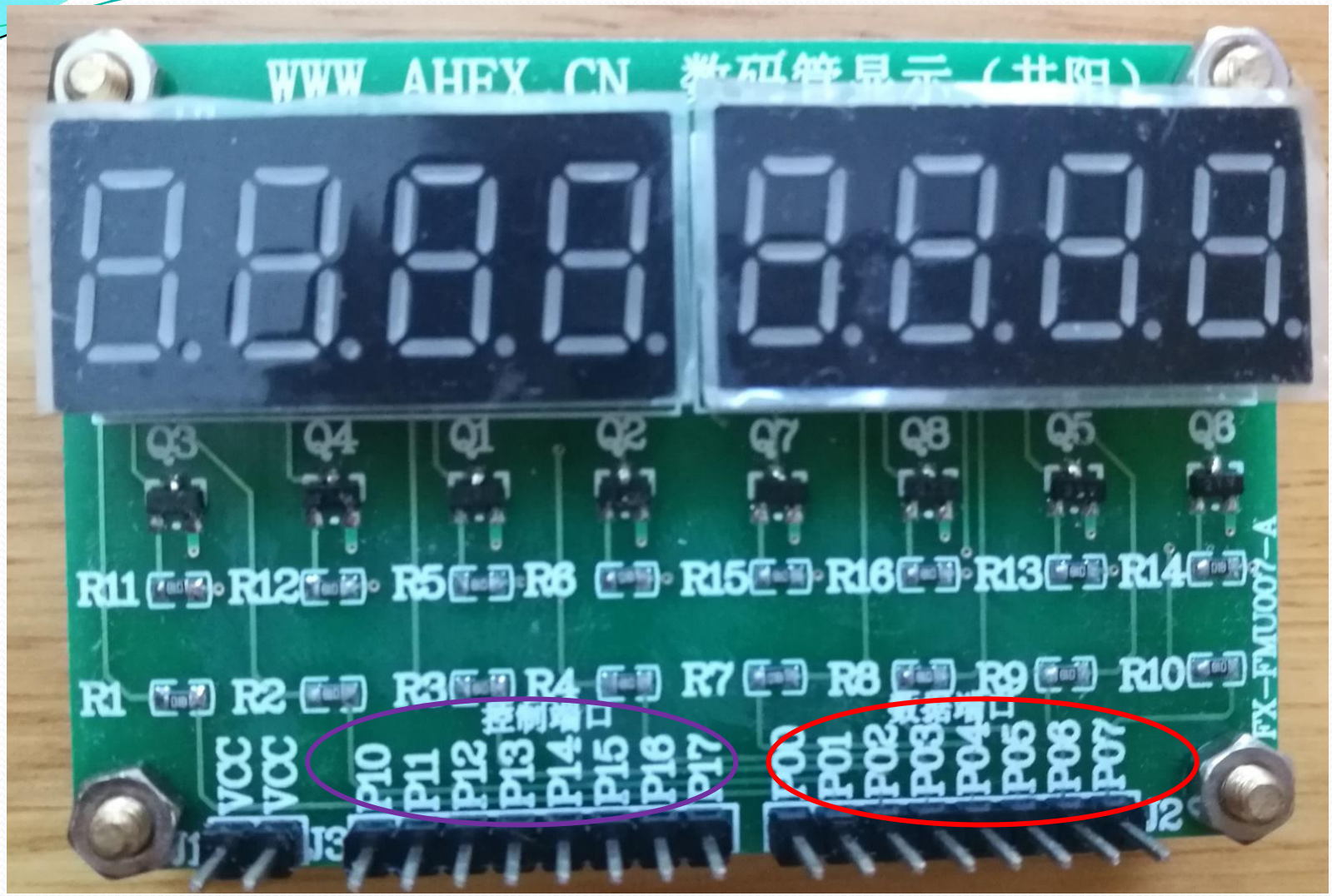
配套程序编写--主程序

```
12  int main(void)
13  {
14    MyDelay(0x7fffff);//软件延时
15
16    LED_Init();         //PA0--PA7对应控制LED1--LED8，PA口初始化为推挽式输出
17    while(1)
18    {
19      GPIO_ResetBits(GPIOA, GPIO_Pin_0);//对于共阳接法，PA0输出0代表点亮LED1
20      GPIO_SetBits(GPIOA, GPIO_Pin_1);//对于共阳接法，PA1输出1代表熄灭LED2
21      GPIO_ResetBits(GPIOA, GPIO_Pin_2);
22      GPIO_SetBits(GPIOA, GPIO_Pin_3);
23      GPIO_ResetBits(GPIOA, GPIO_Pin_4);
24      GPIO_SetBits(GPIOA, GPIO_Pin_5);
25      GPIO_ResetBits(GPIOA, GPIO_Pin_6);
26      GPIO_SetBits(GPIOA, GPIO_Pin_7);
27
28      MyDelay(0x7fffff);//软件延时
29      GPIO_SetBits(GPIOA, GPIO_Pin_0);//对于共阳接法，PA0输出1代表熄灭LED1
30      GPIO_ResetBits(GPIOA, GPIO_Pin_1);//对于共阳接法，PA1输出0代表点亮LED2
31      GPIO_SetBits(GPIOA, GPIO_Pin_2);
32      GPIO_ResetBits(GPIOA, GPIO_Pin_3);
33      GPIO_SetBits(GPIOA, GPIO_Pin_4);
34      GPIO_ResetBits(GPIOA, GPIO_Pin_5);
35      GPIO_SetBits(GPIOA, GPIO_Pin_6);
36      GPIO_ResetBits(GPIOA, GPIO_Pin_7);
37
38      MyDelay(0x7fffff);//软件延时
39    }
40  }
```

PA口初始化：LED_Init()函数

```
3  ////////////////////////////////////////////////////////////////////////////////
4  //PA0--PA7对应控制LED1--LED8
5  ////////////////////////////////////////////////////////////////////////////////
6  void LED_Init(void)
7  {
8     GPIO_InitTypeDef  GPIO_InitStructure;
9
10    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);//开启PA口对应时钟
11   //使用PA0--PA7
12    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
13    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; //推挽输出模式
14    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; //输出频率50Mhz
15    GPIO_Init(GPIOA, &GPIO_InitStructure); //根据制定参数初始化PA口
16 }
17
18
```

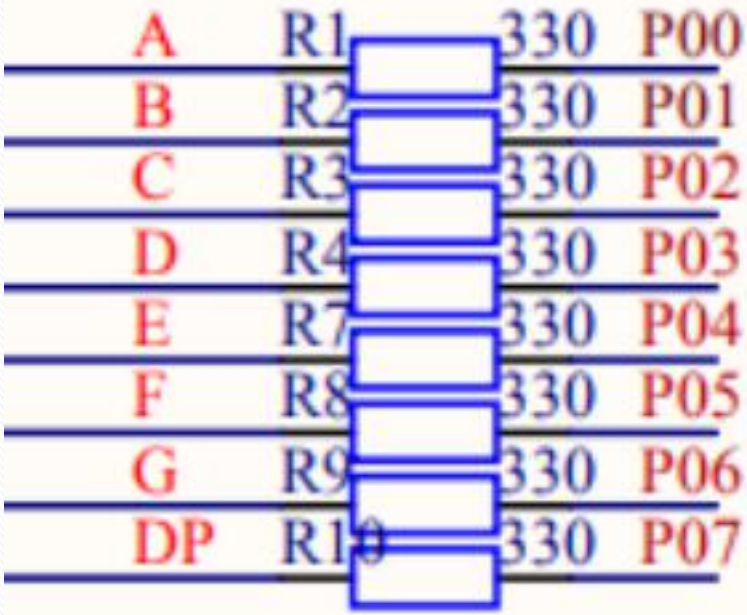
8段数码管模块



P00--P07控制字形（对应A--H）

P10--P17控制字位（从左到右）

字形控制原理

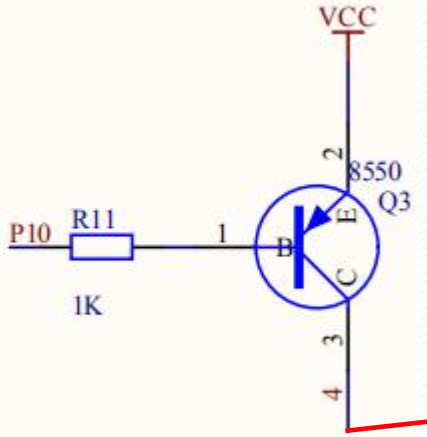


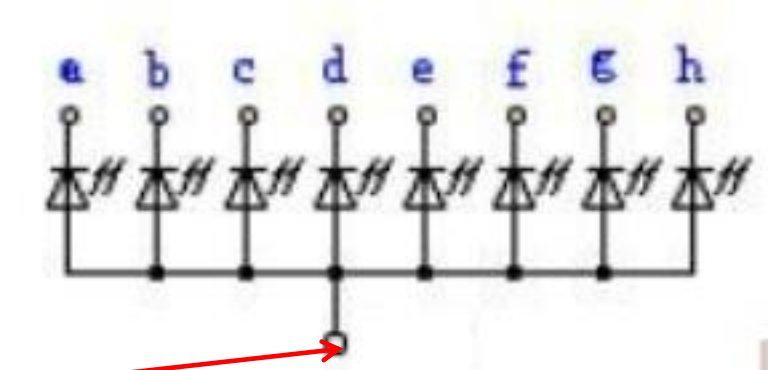
P00--P07控制字形（对应A--H）

可以采用STM32的 PA 口作为数码管的字形控制口（比如：PA7接P07，PA6接P06，...,PA0接P00）；

如果想显示 ’1’，对于共阳接法，需要将第数码管的b,c 位点亮，其他位不点亮，可对 PA端口赋值 0xf9；如果是共阴接法，控制逻辑相反。

显示位置控制原理（以共阳为例）





已知8个数码管同时收到8段字形数据，但可以通过P10--P17控制字位（即从左到右控制8个数码管是否允许显示）

可以采用单片机的 PE 口作为数码管的字位控制口（比如PE0接P10，PE1接P11，...,PE7接P17），当 PE0=0时，PNP三极管 Q3 导通，对应数码管的公共端置高电平，此时该数码管可将收到的字形数据显示出来，而其他PEx=1导致对应的三极管截止，对应的数码管公共端为低电平，数码管即使收到字形数据也无法。

如果打算在左边第一个数码管上显示 1，除了PA口输出字形控制码为0xF9，还需要令PE口输出0xFE。

举例:8段数码管扫描显示0--7

控制字位（从左至右）

控制字形（对应段A—段H，共阳驱动方式）

定义适用于共阳数码管的字形表

```
u8 seg_tab[10] =
{0xC0,
0xF9,
0xA4,
0xB0,
0x99,
0x92,
0x82,
0xF8,
0x80,
0x90};
```

- 将PA口低8位和PE口低8位定义为推挽输出方式

```
18  void LED_Init(void)
19  {
20
21      GPIO_InitTypeDef  GPIO_InitStructure;
22
23      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);  //使能PA口时钟
24
25      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
26      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
27      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
28      GPIO_Init(GPIOA, &GPIO_InitStructure);
29      GPIO_SetBits(GPIOA,GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7);
30
31
32      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE,ENABLE);//使能PE口时钟
33
34      GPIO_InitStructure.GPIO_Pin  = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
35      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; //推挽输出
36      GPIO_Init(GPIOE, &GPIO_InitStructure);//
37      GPIO_SetBits(GPIOE,GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7);
38
39  }
```

- PA0—PA7与P00—P07对应连接，通过查字形表，往PA口输出对应字形数据

```
16  #define DX0 PAout(0)
17  #define DX1 PAout(1)
18  #define DX2 PAout(2)
19  #define DX3 PAout(3)
20  #define DX4 PAout(4)
21  #define DX5 PAout(5)
22  #define DX6 PAout(6)
23  #define DX7 PAout(7)
24  u8 seg_tab[10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90};
25  void Display(u8 index)
26  {
27    DX0 = seg_tab[index]&0x01;
28    DX1 = (seg_tab[index]>>1)&0x01;
29    DX2 = (seg_tab[index]>>2)&0x01;
30    DX3 = (seg_tab[index]>>3)&0x01;
31    DX4 = (seg_tab[index]>>4)&0x01;
32    DX5 = (seg_tab[index]>>5)&0x01;
33    DX6 = (seg_tab[index]>>6)&0x01;
34    DX7 = (seg_tab[index]>>7)&0x01;
35  }
```

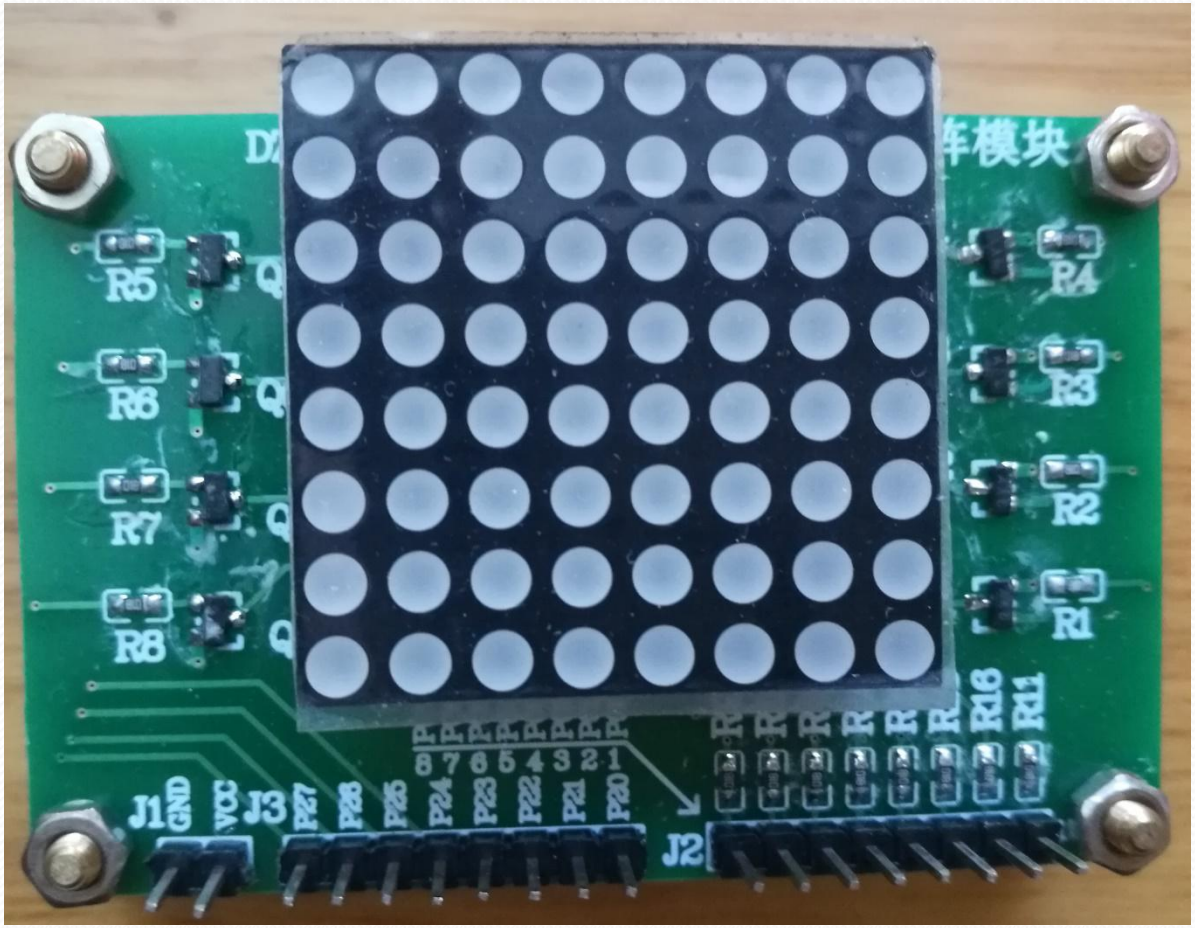
PE0—PE7与P10—P17对应连接，通过控制PE口的对应位控制显示位置

```
#define D0 PEout(0)
#define D1 PEout(1)
#define D2 PEout(2)
#define D3 PEout(3)
#define D4 PEout(4)
#define D5 PEout(5)
#define D6 PEout(6)
#define D7 PEout(7)
int main(void)
 {
    delay_init();       //延时初始化函数
    LED_Init();         //数码管初始化函数
  while(1)
  {
        Display(0);D0=0;delay_ms(1);D0=1;   //第一个数码管显示数字0
        Display(1);D1=0;delay_ms(1);D1=1;   //第二个数码管显示数字1
        Display(2);D2=0;delay_ms(1);D2=1;   //第三个数码管显示数字2
        Display(3);D3=0;delay_ms(1);D3=1;   //第四个数码管显示数字3
        Display(4);D4=0;delay_ms(1);D4=1;   //第五个数码管显示数字4
        Display(5);D5=0;delay_ms(1);D5=1;   //第六个数码管显示数字5
        Display(6);D6=0;delay_ms(1);D6=1;   //第六个数码管显示数字6
        Display(7);D7=0;delay_ms(1);D7=1;   //第七个数码管显示数字7
  }
 }
```

从原理上说，任意时刻只有一个管子允许显示，且收到它需要的字形。

先送出1#管子的字形，打开1#位置的显示，短暂延时后再关闭1#位置的显示，再切换到2#管子的显示处理，...,只要延时时间合适，由于视觉暂留，就会出现8个管子同时显示不同内容的效果（扫描显示）

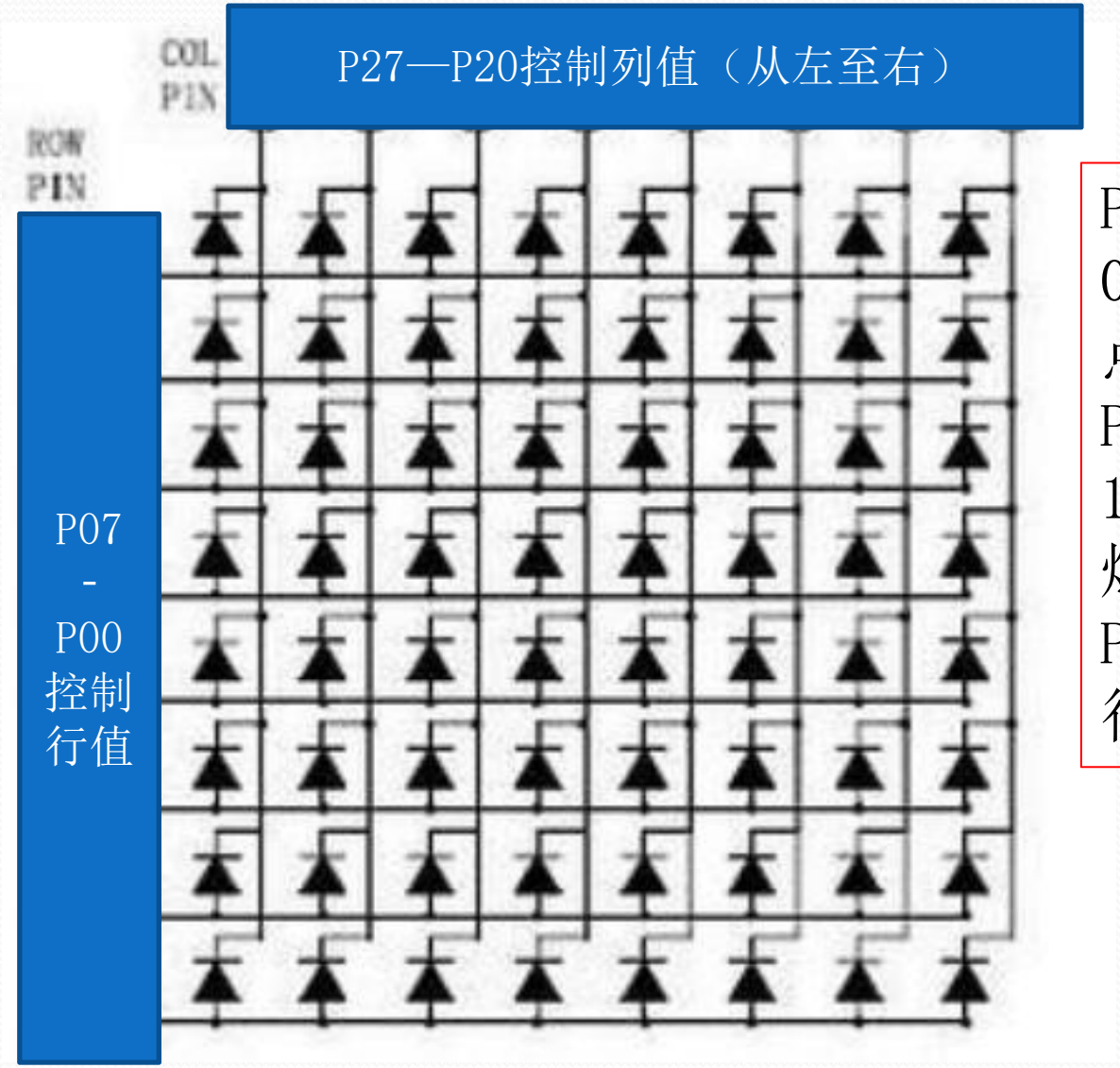
8*8点阵模块



p27--p20处理列值(从左到右）

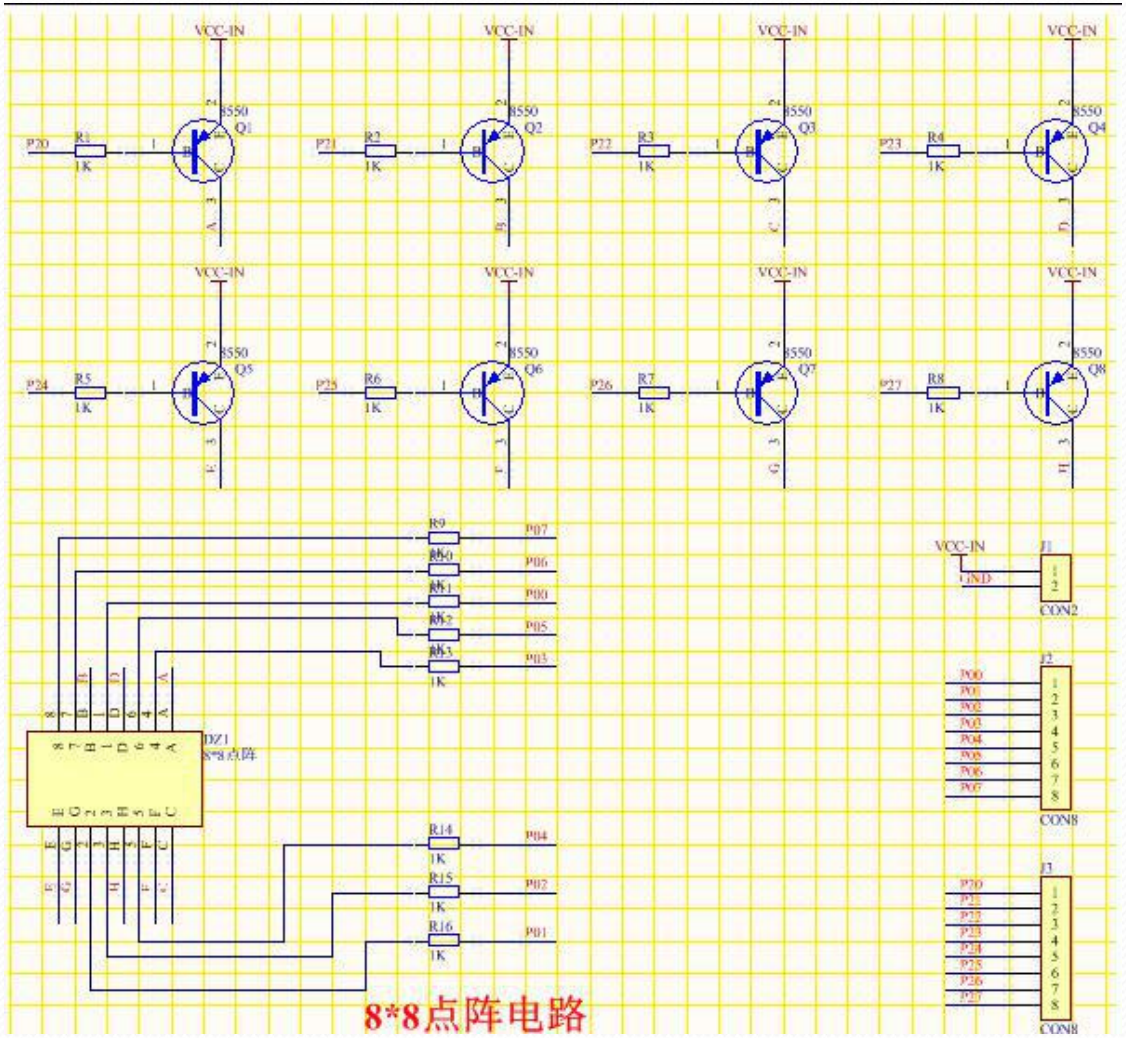
p07--p00处理行值（从上到下）

8*8点阵模块—内部结构



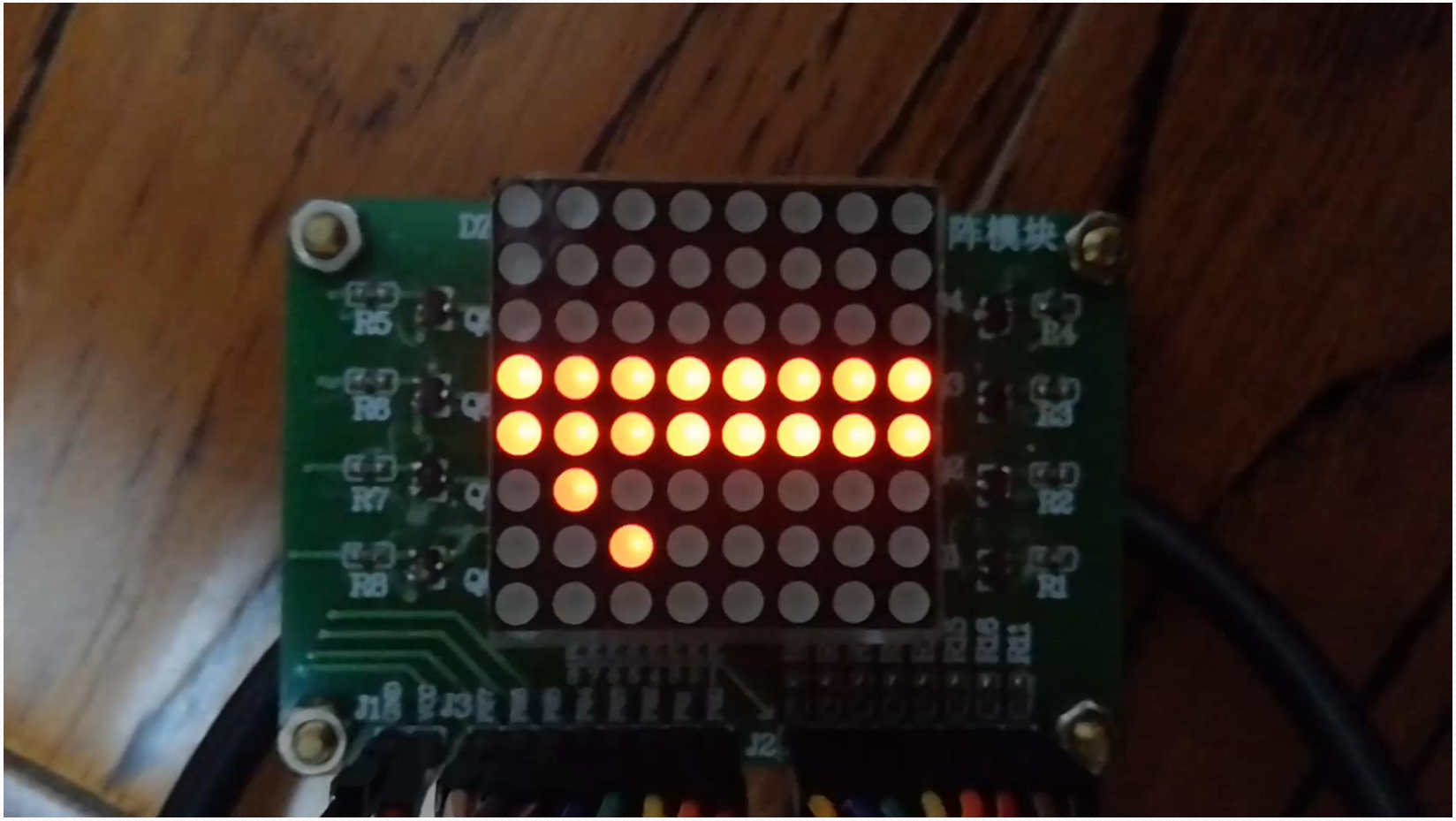
P07输出1，P27输出0，则左上角第一点点亮；
P07输出1，p27输出1，则左上角第一点熄灭；
P07输出0，则第一行8个点全部熄灭

8*8点阵模块—点阵原理



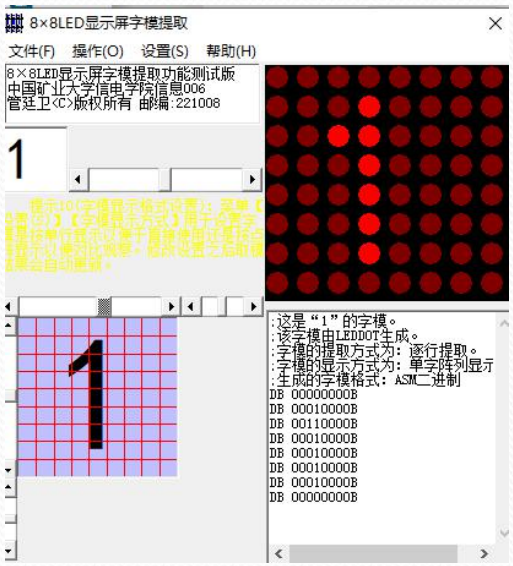
可以采用单片机的 PE 口控制点阵模块的行，单片机的 PA 口控制点阵模块的列；当 PEo 置为低电平时，三极管 Q1 导通，点阵第一行的公共端置高电平，此时可通过单片机的 PA 口对第一行点阵的 8 个 led 灯进行控制，当需要对第一行的某个 led 灯进行点亮，只需要将对应的PAx置低电平即可实现。

举例：8*8点阵依次显示0--9



定义适用于8*8点阵的字形数据

```
u8 seg_tab[88] = {0xff, 0x99, 0x66, 0x7e, 0x7e, 0xbd, 0xdb, 0xe7,       //心形
                  //0xe7, 0xdb, 0x99, 0x99, 0x99, 0x99, 0xdb, 0xe7,       //0
                  0xe7,0xdb,0xdb,0xdb,0xdb,0xdb,0xdb,0xe7,//0
                  0xe7, 0xc7, 0xa7, 0xe7, 0xe7, 0xe7, 0xe7, 0xe7,       //1
                  0xc3, 0x99, 0x99, 0xf1, 0xe3, 0xc7, 0x8f, 0x81,       //2
                  0xc3, 0x99, 0xf9, 0xe3, 0xe3, 0xf9, 0x99, 0xc3,       //3
                  0xf3, 0xe3, 0xc3, 0x93, 0xb3, 0x81, 0xf3, 0xf3,       //4
                  0xc3, 0x9f, 0x9f, 0x83, 0xc1, 0xf9, 0xf9, 0xc3,       //5
                  0xc3, 0x9f, 0x9f, 0x83, 0x81, 0x99, 0x99, 0xc3,       //6
                  0x81, 0x81, 0xf9, 0xf1, 0xe3, 0xe7, 0xe7, 0xe7,       //7
                  0xc3, 0x99, 0x99, 0x81, 0x81, 0x99, 0x99, 0xc3,       //8
                  0xc3, 0x99, 0x99, 0x81, 0xc1, 0xf9, 0xf9, 0xc3        //9
};
```



- 将PA口低8位和PE口低8位定义为推挽输出方式

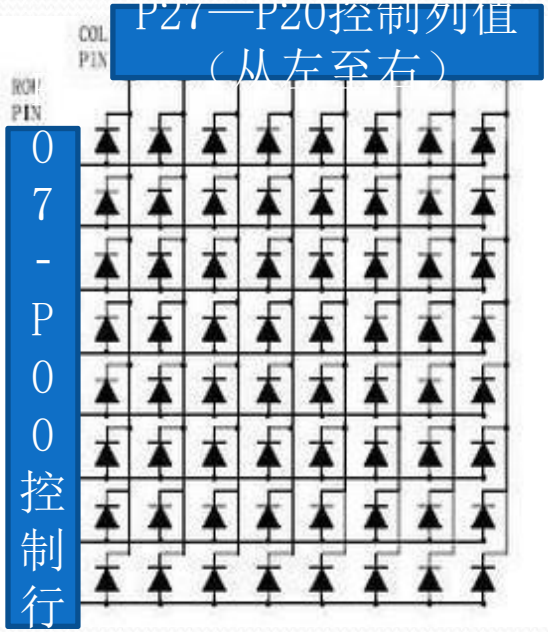
```
18  void LED_Init(void)
19  {
20
21      GPIO_InitTypeDef  GPIO_InitStructure;
22
23      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);  //使能PA口时钟
24
25      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
26      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
27      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
28      GPIO_Init(GPIOA, &GPIO_InitStructure);
29      GPIO_SetBits(GPIOA,GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7);
30
31
32      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE,ENABLE);//使能PE口时钟
33
34      GPIO_InitStructure.GPIO_Pin  = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
35      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; //推挽输出
36      GPIO_Init(GPIOE, &GPIO_InitStructure);//
37      GPIO_SetBits(GPIOE,GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7);
38
39  }
```

- PA7—PA0与P27—P20对应连接，根据起始索引值查表处理某一行的8列

```
17  u8 seg_tab[88] = {0xff, 0x99, 0x66, 0x7e, 0x7e, 0xbd, 0xdb, 0xe7,       //心形
18                    //0xe7, 0xdb, 0x99, 0x99, 0x99, 0x99, 0xdb, 0xe7,       //0
19                    0xe7,0xdb,0xdb,0xdb,0xdb,0xdb,0xdb,0xe7,//0
20                    0xe7, 0xc7, 0xa7, 0xe7, 0xe7, 0xe7, 0xe7, 0xe7,       //1
21                    //...
22  }
 96  #define DX0 PAout(0)
 97  #define DX1 PAout(1)
 98  #define DX2 PAout(2)
 99  #define DX3 PAout(3)
100  #define DX4 PAout(4)
101  #define DX5 PAout(5)
102  #define DX6 PAout(6)
103  #define DX7 PAout(7)
104  void Display(u8 index)  //从字模数组中取出数据，处理某一行的8列
105  {
106     DX0 = seg_tab[index]&0x01;//pa处理列值
107     DX1 = (seg_tab[index]>>1)&0x01;
108     DX2 = (seg_tab[index]>>2)&0x01;
109     DX3 = (seg_tab[index]>>3)&0x01;
110     DX4 = (seg_tab[index]>>4)&0x01;
111     DX5 = (seg_tab[index]>>5)&0x01;
112     DX6 = (seg_tab[index]>>6)&0x01;
113     DX7 = (seg_tab[index]>>7)&0x01;
114  }
```

- PE7—PE0与P07—P00对应连接，对应位输出0，则整行擦除；对应位输出1，则根据列数据显示对应行，连续处理8行则8*8点阵字形呈现

```
#define D0 PEout(0)
#define D1 PEout(1)
#define D2 PEout(2)
#define D3 PEout(3)
#define D4 PEout(4)
#define D5 PEout(5)
#define D6 PEout(6)
#define D7 PEout(7)
```




```
32  void zero(void) //字模表中索引值8--15的位置存放着数字0的8*8点阵数据
33  {
34          Display(8);
35          D0=0;//pe处理行值，先擦
36          delay_ms(1);
37          D0=1;//pe处理行值，后显示
38              Display(9);
39          D1=0;
40          delay_ms(1);
41          D1=1;
42              Display(10);
43          D2=0;
44          delay_ms(1);
45          D2=1;
46              Display(11);
47          D3=0;
48          delay_ms(1);
49          D3=1;
50              Display(12);
51          D4=0;
52          delay_ms(1);
53          D4=1;
54              Display(13);
55          D5=0;
56          delay_ms(1);
57          D5=1;
58              Display(14);
59          D6=0;
60          delay_ms(1);
61          D6=1;
62              Display(15);
63          D7=0;
64          delay_ms(1);
65          D7=1;
66  }
```

实验2：

GPIO外设（1）--输出设备

- 任务1：在8个8段数码管上游走显示A—H
- 任务2：在8个8段数码管上扫描显示A—H

提示：任务1和任务2的不同效果可以通过控制延时的长和短加以实现

- 任务3：在8*8点阵模块上显示自定义字符串，

如：HELLO，WORLD

AHUTJSJ2016

提示：可以借助字模提取软件快速获取字模