OLED点阵屏驱动方法

8 段数码管、1602、5110、12864 点阵屏……这些都是最最常见的、可用于简单电子开发的显示屏。我最喜欢用点阵屏，因为它相比其他字符式的屏来说，可以用来显示任意大小的字符（包括汉字）、图片等。但点阵屏的操作方式略比其他屏复杂……如图，点阵屏要显示的内容是由相应的点按照一定的顺序“拼凑”出来的！下面要说的就是这“拼凑”的方法！

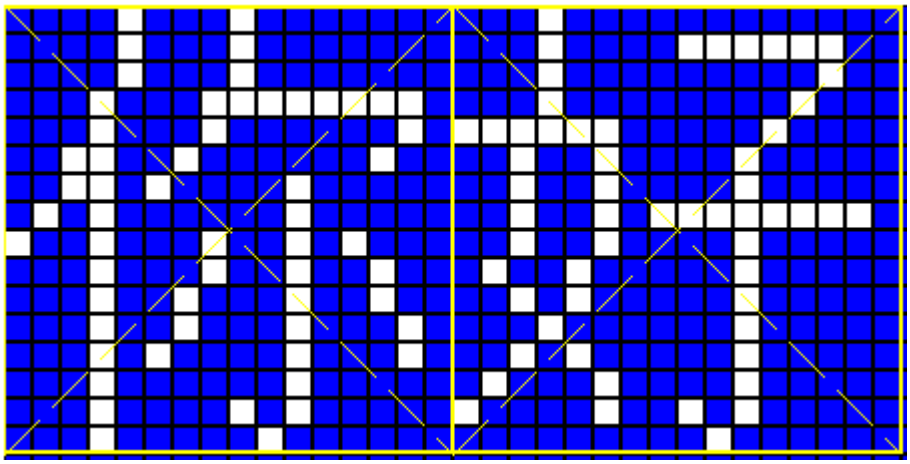

图 1

操作方式大致按图 2 流程来：

图 2

1. 通信方式

SPI、I2C、并行(8080)……这些都是向屏幕发送数据的方式，首先你应该根据自己屏的情况，选择合适的通信方式，然后将 MCU 的通信方式配置好！注意：

- 传统的 51 单片机没有 I2C 或 SPI 功能，只能使用 8080 的并口方式，或者用 IO 口来模拟 I2C 或 SPI 通信；
- 8080 方式的速度是最快的，但是也是最占用 IO 口的（通信就要 8 个 IO 口），这种屏不适合用在 430、STM32 这样的 IO 口不是很丰富的 MCU 上，但这些 MCU 一般都有硬件 I2C 或 SPI 功能；
- 硬件 I2C 或 SPI 比模拟 I2C 货 SPI 速度快，且更省资源，但不同芯片的配置方法不同，这里介绍用 IO 口模拟 SPI 和 I2C 通信的方式。

1.1 模拟 SPI 写数据

```
01 void SPI_Write(unsigned char data)
02 {
03  unsigned char i;
04 
05  CS = 0;//选中SPI设备，一般都是低电平选中
06  for(i=0;i<8;i++) //通过IO口产生高低电平，发送一个byte的数据(8位)
07  {
08    CLK = 0
09    if((data << i) & 0x80)
10    {
11      SI = 1; // SI -- slave input 从机输入
12    }
13    else
14    {
15      SI = 0;
16    }
17    CLK = 1;
18  }
19  CS = 1;
20 }
```

1.2 模拟 I2C 通信

模拟 I2C 就比 SPI 略复杂一些，这就是为什么它比模拟 SPI 略慢

```
01 void IIC_Start() //开启I2C总线
02 {
03  SCL = 1;
04  SDA = 1;
05  SDA = 0;
06  SCL = 0;
07 }
08 void IIC_Stop() //停止I2C总线
09 {
10  SCL = 0;
11  SDA = 0;
12  SCL = 1;
13  SDA = 1;
14 }
15 void Write_IIC_Byte(unsigned char IIC_Byte) //通过I2C向外发送一个byte的数据
16 {
17  unsigned char i;
18  for(i=0;i<8;i++)
19  {
20    if(IIC_Byte & 0x80)
21      SDA = 1;
22    else
23      SDA=0;
24    SCL = 1;
25    SCL=0;
26    IIC_Byte<<=1;
27  }
28  SDA=1;
29  SCL=1;
30  SCL=0;
31 }
```

2. 初始化

要操作点阵显示屏，这里又有一个重要的东西——写数据和写命令！

让屏幕休眠、让屏幕滚动之类的操作，就要向屏幕写命令；要让屏幕显示内容，就要写数据。**怎样让屏幕知道你发送的东西是数据还是命令？有的屏幕有类似 D/C(Data/Command)这样的引脚，说明是通过高低电平来区分，有的屏幕是向屏幕写入特定的值来区分。**

点阵屏内部都有一个驱动芯片，这个驱动芯片就负责按照程序的要求，在某一个点来进行操作。所谓的初始化就是这个驱动芯片的初始化，初始化中注意两个地方：

- 一个是复位！这个非常重要，如果没有完全复位，可能导致屏幕不稳定、显示乱码等畸形问题！
- 另一个就是初始化代码！这个初始化代码一般都是一些 16 进制数，可以在显示屏的规格书之类的文档里面找到，可以通过“写命令”的方式能够把初始

化代码发送给显示屏。

完成了这些步骤，就可以让它显示你想要的东西了！先来看看写到屏幕上的数据是如何显示的：

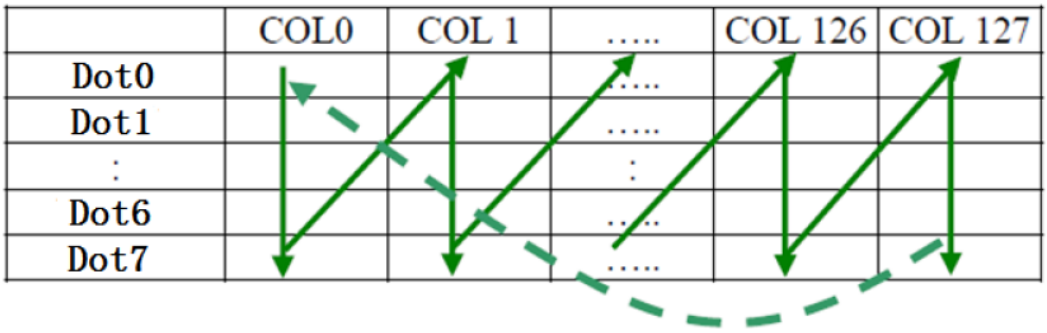

图 3

- 因为每次向屏幕写入一个 byte 的数据，每个 byte 的数据都有 8 位，所以每次至少要操作 8 个点，像图 3 这样：

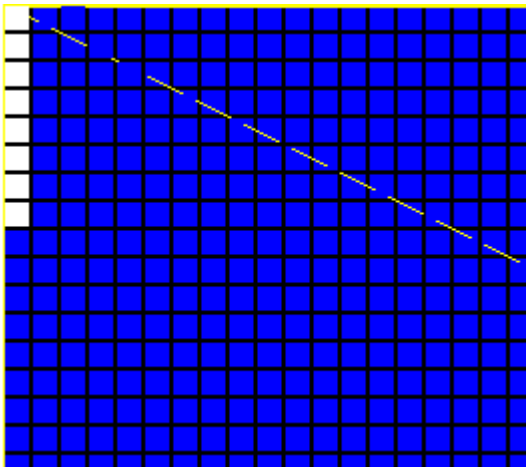

图 4

- 一个 byte 数据的每一位分别对应哪一个点？我分别向屏幕写入了“0x0F”、“0xF0”、“0x01”、“0x02”、“0x04”、“0x08”这样一些有代表性的数据，显示效果是这样的：

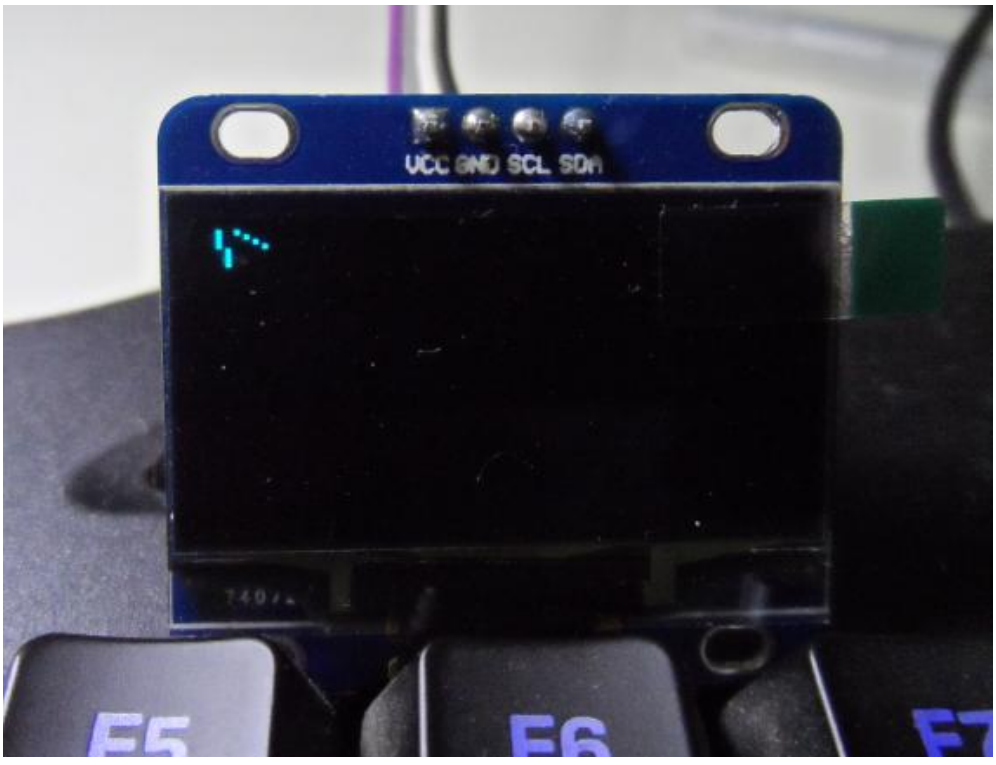

图 5

总结一下：从上到下第一个点为最低位，第 8 个点为最高位！

- 当写完一个 byte 的数据后，下一组数据从哪里开始写？如图 5 所示：

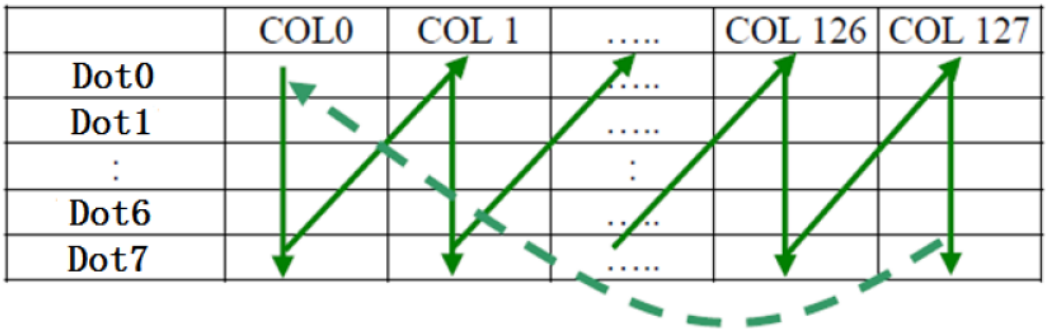

图 6

明白了这个规律，要显示一个字或符号，只要把它对应的点找出来，然后向屏幕写数据就 OK 了！怎么才能得到你想要显示内容的点阵呢？这个可以用软件来计算！这种软件一般被称为“取模软件”——下载地址

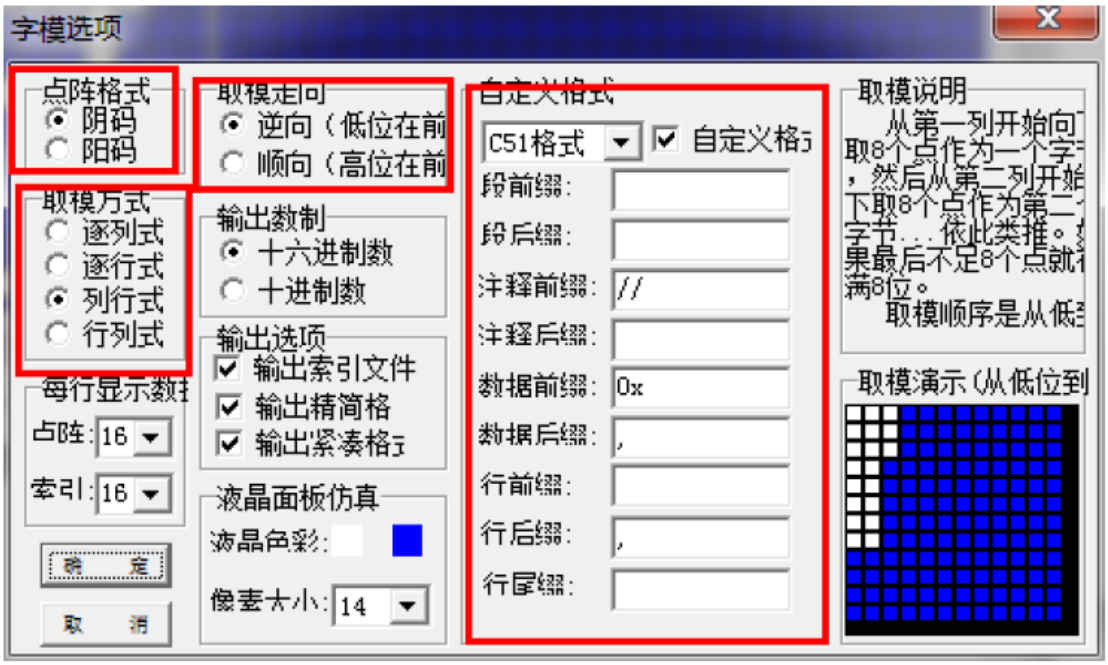

图 7

3. 在显示屏上显示一些东西

我们可以把取得的字模寸在数组里，然后用一个专门的文件来存放……比如 codetab.h(下载样例)

3.1 显示 6*8 的 ASCII 字符串

每一个字符都有独一的 ASCII 值，通过这一点，可以找到某一字符在 codetab 中的具体位置，程序中要做的就是来处理这种“查询”的过程！比如显示一个 6*8 的 ASCII 字符，可以通过下面的代码来完成：

```
01 void Char_F6x8(unsigned char x, unsigned char y, const char ch[]) //x,y对应起始点坐标，x: 0~127, y: 0~7
02 {
03  unsigned char c,i,j=0;
04  while(ch[j] != '\0')
05  {
06  c = ch[j] - 32;
07  if(x>125) //如果一行的点阵不够显示，就换行
08  {
09    x=0;
10    y++;
11  }
12  SetPos(x,y);//设置起始点坐标的方式是通过写命令来完成的，具体什么命令，需要你去查找产品
13  for(i=0;i<6;i++)
14  {
15    WriteData(font6x81);//写数据
16  }
17  x += 6;
18  j++;
19  }
20 }
```

我在 main 函数中写了如下几句代码：

Char_F6x8(0,0,"Welcome to new york, happy everyday -- I know a joke about that sentence, Do U want know? pls reply or e-mail me:hello14blog@gmail.com");

效果是这样的：

图 8

3.2 显示 8*16 的 ASCII 字符串

上面 6*8 的字符，这个倒简单……因为每一个字符竖着都只有 8 个点，刚好是一个 byte。要是想显示大一些的字符，比如说 8*16 的这样会占用两行的字符，怎么操作呢？

```
01 void Char_P8x16Str(unsigned char x, unsigned char y, unsigned char ch[])
02 {
03  unsigned char c=0,i=0,j=0;
04  while (ch[j]!='\0')
05  {
06    c =ch[j]-32;
07    if(x>120) //溢出即换行
08    {
09      x=0;
10      y++;
11    }
12  SetPos(x,y);
13  for(i=0;i<8;i++) //先写完上面一行的8个点
14    WriteData(F8X161);
15  SetPos(x,y+1);
16  for(i=0;i<8;i++) //再写下面一行的8个点
17    WriteDdata(F8X161);
18  x+=8;
19  j++;
20  }
21 }
```

显示效果是这样的：

图 9

3.3 显示 16*16 汉字

因为汉字不能通过 ASCII 码来查询，所以用了一种特殊的方式，我习惯将其称之为“字码索引”。先来看看汉字的字码是什么样子的：

```
0x80,0x80,0xC0,0x60,0x30,0x18,0x2C,0xE7,0xC7,0x0C,0x18,0x30,0x60,0xC0,0x80,0x80,
0x00,0x00,0x00,0x02,0x02,0x02,0x02,0x82,0xC2,0x62,0x3A,0x1E,0x06,0x00,0x00,0x00,//今",0

0x40,0x40,0x42,0x42,0x42,0x42,0x42,0xFE,0xFE,0x42,0x42,0x42,0x42,0x42,0x40,0x40,
0x80,0x80,0xC0,0x60,0x30,0x1C,0x0F,0x03,0x03,0x0F,0x1C,0x30,0x60,0xC0,0x80,0x80,//天",1

0x40,0x40,0x42,0x42,0x42,0x42,0x42,0xFE,0xFE,0x42,0x42,0x42,0x42,0x42,0x40,0x40,
0x80,0x80,0xC0,0x60,0x30,0x1C,0x0F,0x03,0x03,0x0F,0x1C,0x30,0x60,0xC0,0x80,0x80,//天",2

0x20,0x30,0x5C,0x4F,0x57,0x54,0x54,0x54,0x54,0x54,0x54,0xD4,0xD4,0x04,0x04,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0x3F,0x70,0xF0,0xF0,//气",3

0x00,0xFC,0xFC,0x84,0xFC,0xFC,0x44,0x54,0x54,0x54,0x7F,0x7F,0x54,0x54,0x54,0x44,
0x00,0x3F,0x3F,0x10,0x3F,0x3F,0x00,0xFF,0xFF,0x15,0x15,0x55,0xD5,0xFF,0x7F,0x00,//晴",4

0x00,0xFC,0xFC,0x25,0x27,0x26,0xFC,0xFC,0x00,0xFE,0xFE,0x22,0x22,0xFE,0xFE,0x00,
0x00,0x7F,0x7F,0x31,0x15,0x1D,0xB9,0xF1,0x70,0x3F,0x0F,0x42,0xC2,0xFF,0x7F,0x00,//朗",5
```

后面的 0、1、2、3、4、5 这样的数字，就是字码的索引，用代码这样来完成操作：

```
void Char_P16x16Ch(unsigned char x, y, N) //x,y是起始点坐标，N 就是对应的汉字
{
 unsigned char wm=0;
 unsigned int adder=32*N; //通过字码的索引来确定地址
 SetPos(x , y); //先写上面一行的8个点
 for(wm = 0;wm < 16;wm++)
 {
   WriteData(F16x16[adder]);
   adder += 1;
 }
 SetPos(x,y + 1); //再写下面一行的8个点
 for(wm = 0;wm < 16;wm++)
 {
   WriteData(F16x16[adder]);
   adder += 1;
 }
}
```

显示效果：

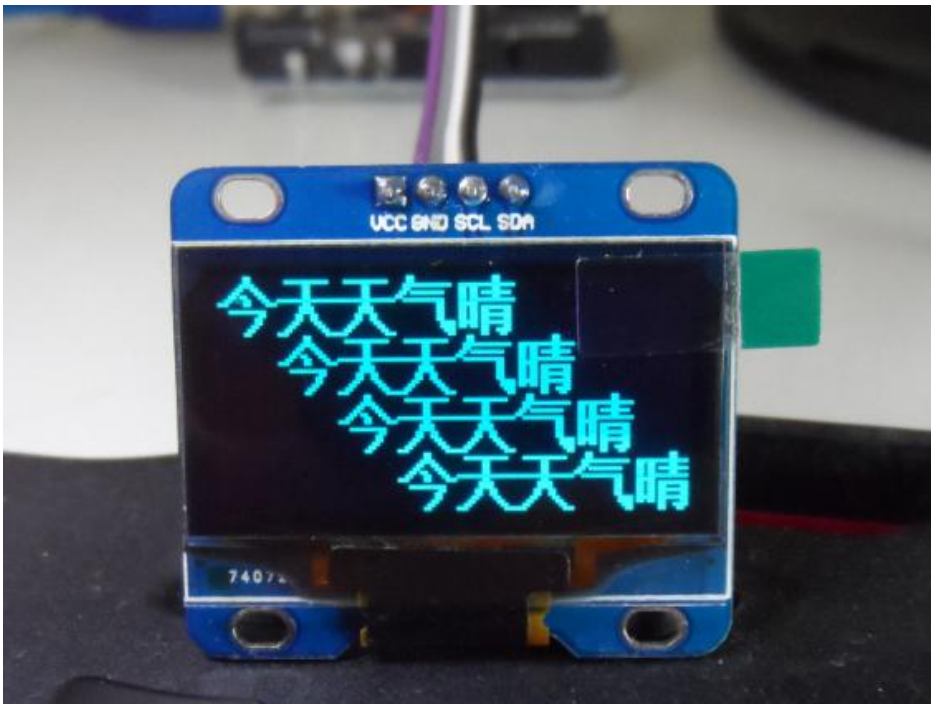

图 10

3.4 显示图片的函数

```
void Draw_BMP(unsigned char x0, y0,x1, y1,unsigned char BMP[]) //x0,y0是图片起始点的坐标，x1,y1是结束点坐标
{
 unsigned int j=0;
 unsigned char x,y;

 if(y1%8==0)
   y=y1/8;
 else
   y=y1/8+1;
 for(y=y0;y<y1;y++)
 {
   LCD_Set_Pos(x0,y);
   for(x=x0;x<x1;x++)
   {
     LCD_WrDat(BMP[j++]);
   }
 }
}
```

显示效果：

图 11

4. 挽总

- **上文介绍了一些主要显示函数，基本上可以满足一些简单的电子开发；**
- **还有更多的功能，比如要显示一个点，显示波形……以后再介绍；**
- **演示所用的点阵屏叫“OLED 显示屏”，它有体积小，显示效果锐利，超低功耗等优点……我在这家买的：**

  http://item.taobao.com/item.htm?spm=2013.1.0.0.s62LHC&id=36399661547