LCD 的驱动不像 LED 那样，加上电压（LED 实际上是电流驱动）就可以长期显示的。LCD 驱动必须使用交流电压驱动才能保持稳定的显示，如果在 LCD 上加上稳定的直流电压，不但不能正常显示，时间久了还会损坏 LCD。一段 LCD 由背电极和段电极组成，需要显示时，在背电极和段电极之间加上合适的交流电压（通常使用方波）。为了调节对比度，可以调节方波中每半个周期中显示的时间（即占空比）来实现。

通常，为了节约驱动口，将多个背电极连在一起，形成公共背电极端：COM。另外，再将属于不同 COM 的段电极连接在一起，形成公共段电极端：SEG。当在某个 COM 和某个 SEG 之间加了足够的交流电压之后，就会将对应的段点亮（实际上是变黑）。

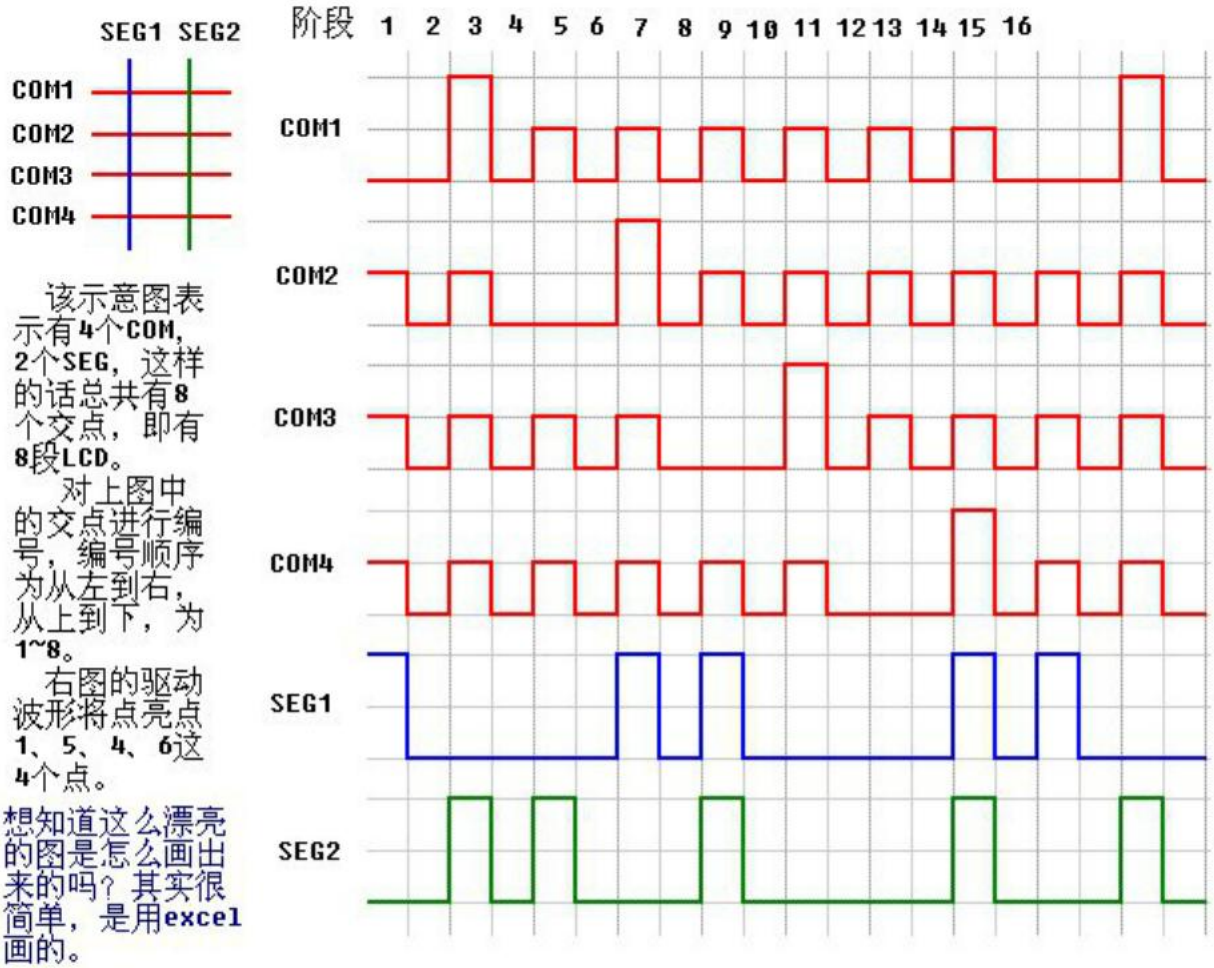
像万利的板子上使用的这种 LCD，有 4 个 COM，还有 16 个 SEG。要想某一 SEG 显示时，需要在对应的 SEG 和 COM 之间加上足够的交流电压。在万利的板子上，COM 驱动使用了两个电阻分压，输出电压为 1/2Vcc，当不想让某位显示时，就将它的电压设置为 1/2Vcc（通过设置 IO 口为高阻态来完成），这样加在对应的 SEG 和 COM 之间的电压只有 1/2Vcc，不足以点亮对应的 SEG。需要显示的，就将 COM 电压设置为 0 或者 1，这样 SEG 电压跟 COM 电压相反的段就被点亮了（变黑），因为它们之间的电压为 Vcc。通过定期扫描每个 COM，即可稳定的在 LCD 上显示需要的图形了。需要显示字符或者数字时，自己先将对应的图案设计好，在显示时，发送到相应的 SEG 和 COM 上即可。但是如果使用 100%的时间都驱动的话，会造成对比度太高，甚至出现不该显示的地方也显示了。因此在显示一段时间后，就将 COM 和 SEG 都设置为低，以关闭它的显示，降低对比度。通过调节关闭时间的长短（PWM），可以调节对比度。在下面的测试程序中，为了简化程序，使用了 50%固定的占空比。

为了方便描述，我们把 COM 为低电平时点亮叫做正亮，COM 为高电平时点亮叫做负亮。扫描每个 COM 分成 4 个阶段：正亮，关闭，负亮，关闭。因此对于本板子上的 LCD 驱动，总共有 16 个状态，每个 COM 都有上面所说的 4 个状态。我们每隔 2ms 就切换一次状态，这样整个扫描周期就是 2*16=32ms，基本上感觉不到闪烁。

但是需要注意的是，这个 LCD 中的每个 COM 并不是刚好对应着显示图案中的一个字符的位置。每个 COM 都对应着每个显示字符中的相同 4 段！换句话说，要显示第一个字符位置的字符，每个 COM 都要被用到。因此，要改变某个字符位置的显示，就需要改变每次 COM 输出时对应的 SEG 中的 4 段。为此，建立一个缓冲区，当需要修改显示字符时，就修改缓冲区中的内容。这个缓冲区有 4 行，每行中有 16 个 SEG，对应着一个 COM。需要修改显示时，把每行中对应的 4 个 SEG 设置为需要的值，这样就实现了某个显示位置图案的修改。

为了显示字符，需要事先把需要显示的字符按照 SEG 和 COM 的分布，制作成数据保存起来，需要显示时，就把它复制到显示缓冲区中对应的位置去。另外，由于输入的参数是字符的 ASCII 码，因此还需要将 ASCII 码转换为对应的字符图案的索引值。使用一个专门的函数来完成这些转换和填充缓冲区，在需要修改显示数据时，就调用该函数。

为了方便大家对这个 LCD 的驱动方式和编程，下面简单的画一下驱动的波形图。



驱动波形 (原文件名:LcdDriver.GIF)

这里只画出 2 个 SEG 波形图，实际有 16 个 SEG，只要你理解了 2 个 SEG 的，那么 16 个的也是一样的意思。

如图所示，所有偶数阶段都是关闭显示阶段，这时 COM 和 SEG 都是 0，将不会有段被点亮，通过调节关闭

显示阶段所占的时间百分比，即可调节总体显示的对比度。SEG 和 COM 之间电平相差 1 格的显示不出来或者

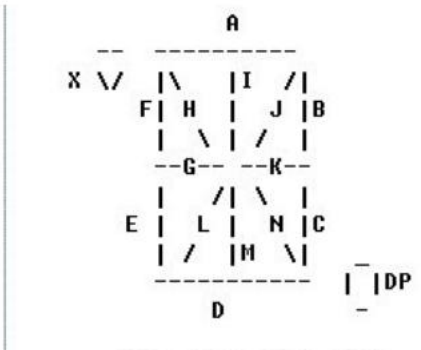
浓度不够，而 SEG 和 COM 之间电平相差 2 格的则可以显示出来或者浓度较深。例如第一阶段中的 SEG1 和 COM1

之间相差 2 格，第三阶段中 COM1 和 SEG1 相差 2 格，因而 SEG1 和 COM1 之间的交叉点（即点 1）被显示。又如

第九阶段的 SEG1 和 COM3 之间相差 2 格，第十一阶段中的 COM3 和 SEG1 之间相差 2 格，因而 SEG1 和 COM3 之间的

交叉点（即点 5）被显示出来。其它点以此类推。

最后，再来看看万利板子上的 LCD 的 COM 和 SEG 之间的关系图，如下图所示。



该LCD的每个字符位置有16段，
用符号表示如左图所示。

(SEG0~SEG15分别连接在STM32的PE0~PE15口上)

	PE0	PE1	PE2	PE3	...	PE15
	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
COM1	1X	1I	1A	1DP	2X	2I	2A	2DP	3X	3I	3A	3DP	4X	4I	4A	4DP
COM2	1F	1H	1J	1B	2F	2H	2J	2B	3F	3H	3J	3B	4F	4H	4J	4B
COM3	1E	1G	1K	1C	2E	2G	2K	2C	3E	3G	3K	3C	4E	4G	4K	4C
COM4	1L	1M	1N	1D	2L	2M	2N	2D	3L	3M	3N	3D	4L	4M	4N	4D

段码分布 (原文件名:LcdDriver1.GIF)

图中显示，S0、S1、S2、S3 属于第一个字符，在显示第一个字符时，只要在对应的 COM 选中时，将需要显示的 SEG 放在上面即可。其余几个字符类似。例如要显示一个数字 3，则应该将 A 段、B 段、C 段、D 段、G 段、K 段显示。

某段显示，用 1 表示，不显示用 0 表示，得到的各段值如下：

（低--------》高）

X=0 I="0" A="1" DP="0"　　0 0 1 0

F=0 H="0" J="0" B="1"　　0 0 0 1

E=0 G="1" K="1" C="1"　　0 1 1 1

L=0 M="0" N="0" D="1"　　0 0 0 1

注意是低位在先的，把每行用十六进制来表示（高位在先），就是 0x4，0x8，0xE，0x8。它们分别对应着 COM1~COM3 选中时 S3~S0 的输出值。为了方便管理，将这 4 个十六进制值合并为一个 2 字节的值 0x48E8 保存。

其它各字符的构造方式相同。显示时，分别取出各段的值写入到对应的缓冲区去。

扫描 LCD 的程序流程如下：

①、COM1 设置为低电平，其余 COM 为 1/2 高电平，设置 PE 口为需要的电平（16 个段码），延时 2ms；

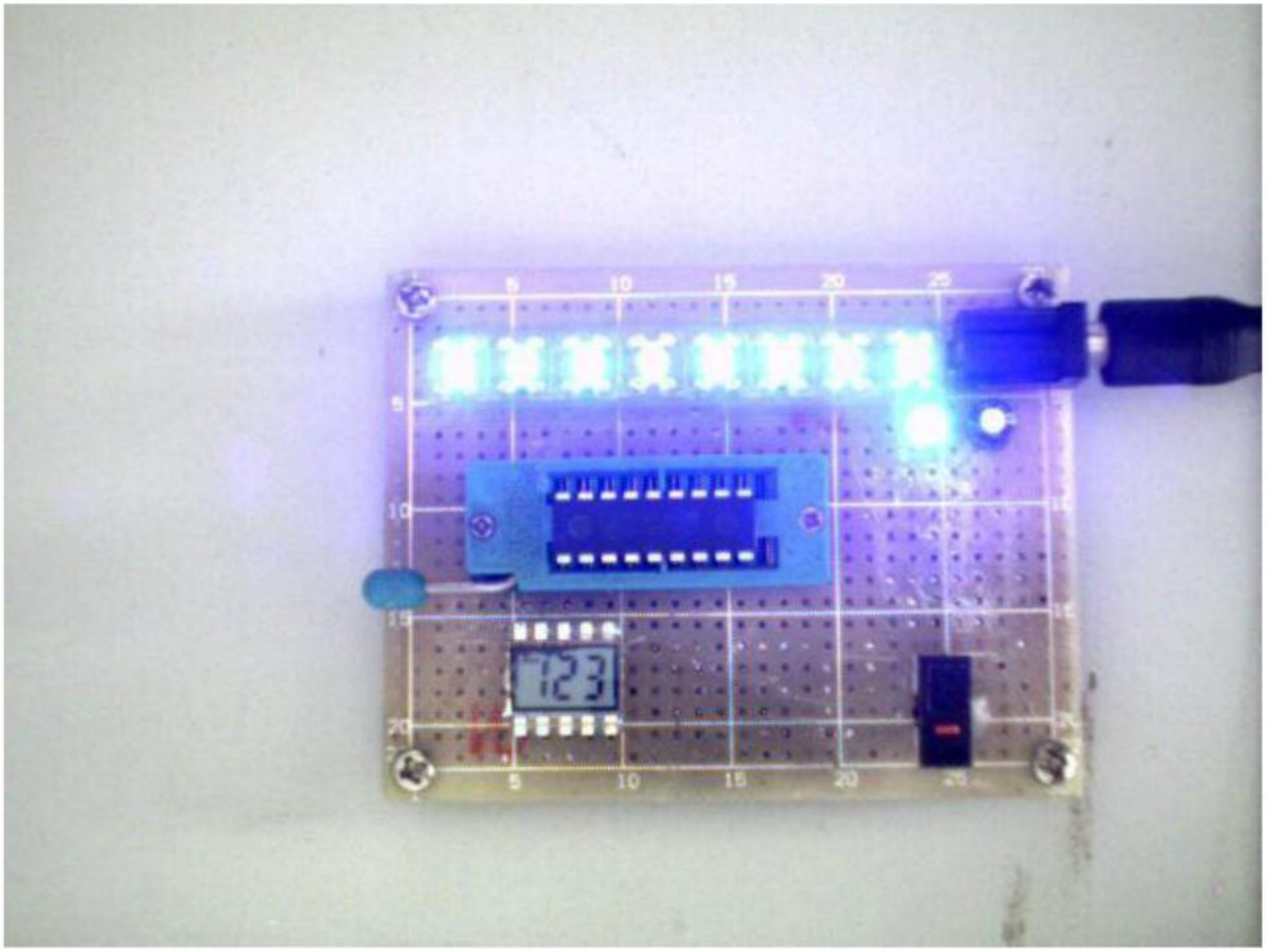
②、4 个 COM、PE 口均设置为低电平，关闭显示，延时 2ms；

③、COM1 设置为高电平，其余 COM 为 1/2 高电平，设置 PE 口为需要的电平（第一步 16 个段码的取反），延时 2ms。

④、4 个 COM、PE 口均设置为低电平，关闭显示，延时 2ms；

然后对剩下的 3 个 COM 重复前面 4 个步骤，这样一个完整的扫描就完成了。

用这个原理驱动过一个产品的 LCD，效果不错。2.5ms 中断一次，20ms 完成一次扫描，50HZ 的刷新率。
用的是 MEGA8，省掉了一片 HT1621




```
//引入头文件************************************************************
#include    "delay.h"
#include    "delay.c"
#include     <pic1684.h>

//字符查表************************************************************
    const unsigned char Lcd[10] = {0x3f, 0x06, 0x5b, 0x4f, 0x66,
                                   0x6d, 0x7d, 0x07, 0x7f, 0x6f};
```

```
//*********************************************************************
//函数名称：port_init();
//输入参数：无
//输出参数：无
//功能描述：端口设置
//建造日期：2008.06.03
//*********************************************************************
void PortInit(void)
 {
   PORTA = 0xff;                          //A 口设置
   PORTB = 0xff;                          //B 口设置

   TRISA = 0x00;                          //
   TRISB = 0x00;                          //
 }

//*********************************************************************
//函数名称：main();
//输入参数：无
//输出参数：无
//功能描述：主要程序
//建造日期：2008.06.03
//*********************************************************************
void main(void)                           //
 {
   unsigned char temp[3];
   unsigned char i, j;

   PortInit();                            //脚位设置

   temp[0] = Lcd[1];
   temp[1] = Lcd[2];
   temp[2] = Lcd[3];

   while (1)
    {
     PORTA = 0b11111110;                  //正亮显示
     TRISA = 0b11111110;

     PORTB = temp[0];
     DelayMs(2);

     PORTA = 0b11111000;                  //关闭显示
     TRISA = 0b11111000;
```

```
PORTB = 0x00;
DelayMs(2);

PORTA = 0b11111111;                    //负亮显示
TRISA = 0b11111110;

PORTB = temp[0] ^ 0xff;
DelayMs(2);

PORTA = 0b11111000;                    //关闭显示
TRISA = 0b11111000;

PORTB = 0x00;
DelayMs(2);

PORTA = 0b11111101;                    //正亮显示
TRISA = 0b11111101;

PORTB = temp[1];
DelayMs(2);

PORTA = 0b11111000;                    //关闭显示
TRISA = 0b11111000;

PORTB = 0x00;
DelayMs(2);

PORTA = 0b11111111;                    //负亮显示
TRISA = 0b11111101;

PORTB = temp[1] ^ 0xff;
DelayMs(2);

PORTA = 0b11111000;                    //关闭显示
TRISA = 0b11111000;

PORTB = 0x00;
DelayMs(2);

PORTA = 0b11111011;                    //正亮显示
TRISA = 0b11111011;

PORTB = temp[2];
```

```
    DelayMs(2);

    PORTA = 0b11111000;                    //关闭显示
    TRISA = 0b11111000;

    PORTB = 0x00;
    DelayMs(2);

    PORTA = 0b11111111;                    //负亮显示
    TRISA = 0b11111011;

    PORTB = temp[2] ^ 0xff;
    DelayMs(2);

    PORTA = 0b11111000;                    //关闭显示
    TRISA = 0b11111000;

    PORTB = 0x00;
    DelayMs(2);
    }
}
```