

2.4 寸 TFT 液晶屏(带触摸屏)

点屏工作笔记

作者: luocheng.sz@gamil.com

1. 液晶屏硬件

1.1 液晶屏外观



图 1

软排线共有 37 脚，左边为第 1 脚。

1.2 信号定义

管脚号	定义	说明
1	DB0	数据线
2	DB1	数据线
3	DB2	数据线
4	DB3	数据线
5	GND	地线
6	VCC	电源 3.3V
7	#CS	片选信号（低有效）
8	RS	寄存器选择
9	#WR	写信号（低有效）
10	#RD	读信号（低有效）
11	NC	空脚
12	TP_X+	触摸屏 X+
13	TP_Y+	触摸屏 Y+
14	TP_X-	触摸屏 X-
15	TP_Y-	触摸屏 Y-
16	LED_VCC	LED 背光电源 3.3V
17	LED1_GND	LED 背光地
18	LED2_GND	LED 背光地
19	LED3_GND	LED 背光地
20	LED4_GND	LED 背光地
21	NC	空脚
22	DB4	数据线
23	DB10	数据线
24	DB11	数据线
25	DB12	数据线
26	DB13	数据线
27	DB14	数据线
28	DB15	数据线
29	DB16	数据线
30	DB17	数据线
31	#RESET	复位(低有效)
32	VCC	电源 3.3V
33	VCC	电源 3.3V
34	GND	地线
35	DB5	数据线
36	DB6	数据线
37	DB7	数据线

表 1

1.3 数据总线 8/16 位选择

该屏出厂时，数据总线默认是 16 位的：

高 8 位：DB17 – DB10，

低 8 位：DB7 – DB0

如果要工作在 8 位模式，需要将软排线背面的电阻 R2 取下，焊到电阻 R1 的位置上。

当工作在 8 位模式时，数据总线为：DB17 – DB10

2. 内部驱动 IC

该屏内置的驱动 IC 为 OTM3225A 或者 ILI9325，二者功能相互兼容。在网络上可搜索到这两种 IC 的 Datasheet.

3. 测试电路原理图

我选择单片机 AVR Atmega16A 对该屏进行测试，CPU 外接 16MHz 晶体。

数据总线为 8 位模式。

部分电路原理图如下：



电路实物照片如下：

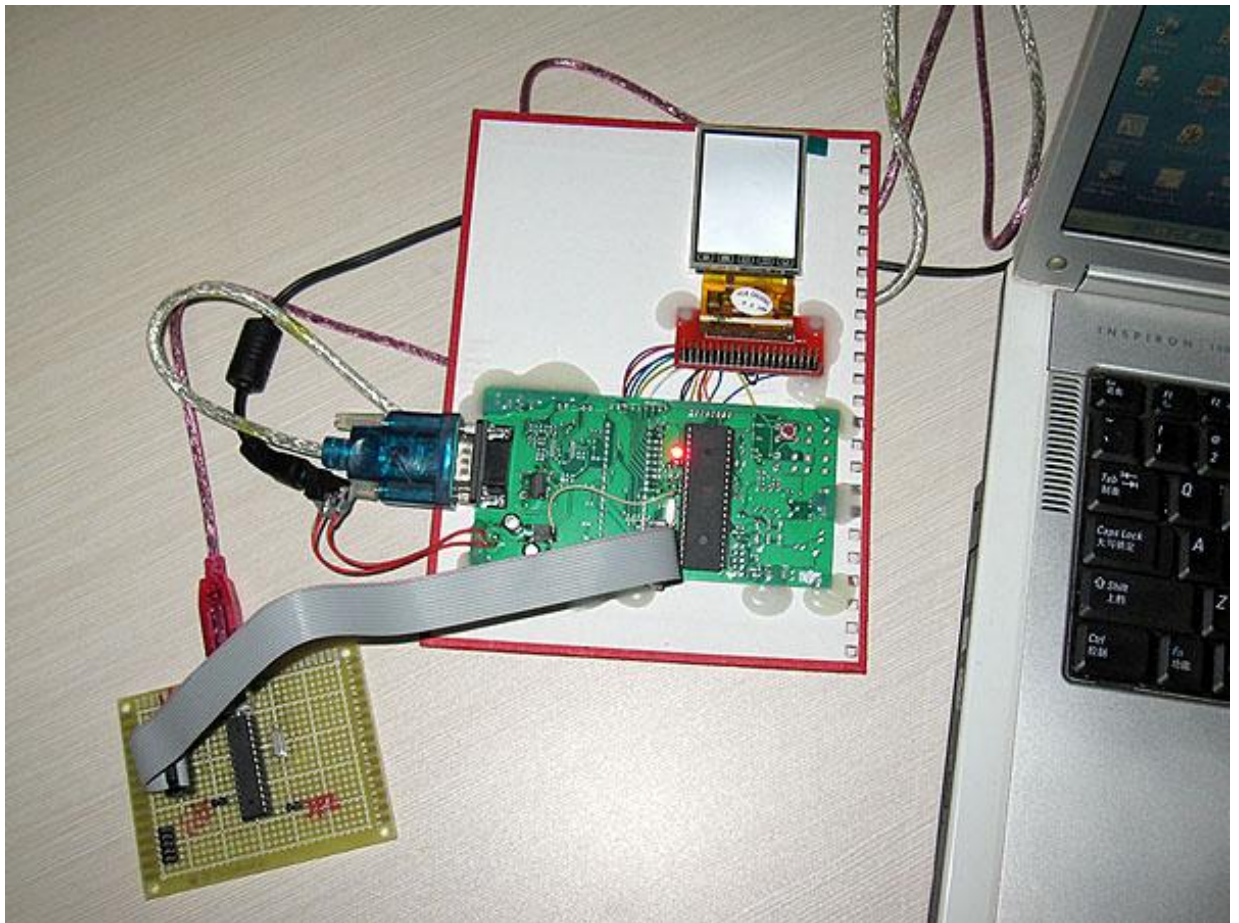


图 3

4. 软件说明

4.1 软件流程图

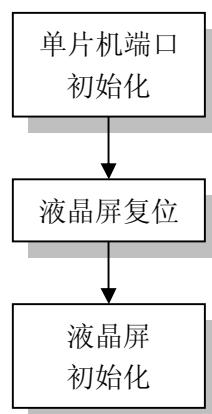


图 4

4.2 软件代码及说明

程序代码为 C 语言，编译环境为：AVR Studio4.18 + WinAVR 20100110 （AVR GCC）

4.2.1 单片机端口初始化

```
1  #define F_CPU 16000000UL
2
3  #include <avr/io.h>
4  #include <avr/interrupt.h>
5  #include <util/delay.h>
6
7  #define RESET_1      PORTD |=  _BV(PD5)
8  #define RESET_0      PORTD &=  ~_BV(PD5)
9  #define CS_1          PORTC |=  _BV(PC5)
10 #define CS_0          PORTC &=  ~_BV(PC5)
11 #define RS_1          PORTC |=  _BV(PC4)
12 #define RS_0          PORTC &=  ~_BV(PC4)
13 #define WR_1          PORTC |=  _BV(PC3)
14 #define WR_0          PORTC &=  ~_BV(PC3)
15 #define RD_1          PORTC |=  _BV(PC2)
16 #define RD_0          PORTC &=  ~_BV(PC2)
17 /*****
18 /* 端口初始化 */
19 /*****/
20 void init_IO(void)
21 {
22     DDRA = _BV(PA4) | _BV(PA3) | _BV(PA1);
23     DDRB = 0XFF;
24     DDRC = 0XFF;
25     DDRD = _BV(PD5);
26
27     RESET_1;
28     CS_1;
29     RS_0;
30     WR_1;
31     RD_1;
32 }
33 // End of init_IO()
```

源代码 1

4.2.2 液晶屏复位

```
1  /******  
2  /* 液晶屏复位 */  
3  /******  
4  void LCD_reset(void)  
5  {  
6      RESET_0;  
7      _delay_ms(50);    // 按照 delay.h 中的说明，单次延时长度不要大于 60ms  
8      _delay_ms(50);  
9      RESET_1;  
10     _delay_ms(50);  
11     _delay_ms(50);  
12 }  
13 // End of LCD_reset( )
```

源代码 2

4.2.3 液晶屏初始化

```
1  /******  
2  /* 液晶屏初始化 */  
3  /******  
4  // 说明：本函数代码全部参考网上资料，参数未做改动  
5  void LCD_init(void)  
6  {  
7      /****** Start Initial Sequence *****/  
8      LCD_WR_CMD(0xE3, 0x3008); // Set internal timing  
9      LCD_WR_CMD(0xE7, 0x0012); // Set internal timing  
10     LCD_WR_CMD(0xEF, 0x1231); // Set internal timing  
11     LCD_WR_CMD(0x01, 0x0100); // set SS and SM bit  
12     LCD_WR_CMD(0x02, 0x0700); // set 1 line inversion  
13     LCD_WR_CMD(0x03, 0x1030); // set GRAM write direction and BGR=1.  
14     LCD_WR_CMD(0x04, 0x0000); // Resize register  
15     LCD_WR_CMD(0x08, 0x0202); // set the back porch and front porch  
16     LCD_WR_CMD(0x09, 0x0000); // set non-display area refresh cycle ISC[3:0]  
17     LCD_WR_CMD(0x0A, 0x0000); // FMARK function  
18     LCD_WR_CMD(0x0C, 0x0000); // RGB interface setting  
19     LCD_WR_CMD(0x0D, 0x0000); // Frame marker Position  
20     LCD_WR_CMD(0x0F, 0x0000); // RGB interface polarity
```

```

21  /*****Power On sequence *****/
22  LCD_WR_CMD(0x10, 0x0000); // SAP, BT[3:0], AP, DSTB, SLP, STB
23  LCD_WR_CMD(0x11, 0x0007); // DC1[2:0], DC0[2:0], VC[2:0]
24  LCD_WR_CMD(0x12, 0x0000); // VREG1OUT voltage
25  LCD_WR_CMD(0x13, 0x0000); // VDV[4:0] for VCOM amplitude
26  _delay_ms(50); // Dis-charge capacitor power voltage
27  _delay_ms(50);
28  _delay_ms(50);
29  _delay_ms(50);
30  LCD_WR_CMD(0x10, 0x1690); // SAP, BT[3:0], AP, DSTB, SLP, STB
31  LCD_WR_CMD(0x11, 0x0227); // R11h=0x0221 at VCI=3.3V, DC1[2:0], DC0[2:0], VC[2:0]
32  _delay_ms(50); // Delay 50ms
33  LCD_WR_CMD(0x12, 0x001C); // External reference voltage= Vci;
34  _delay_ms(50); // Delay 50ms
35  LCD_WR_CMD(0x13, 0x1800); // R13=1200 when R12=009D;VDV[4:0] for VCOM amplitude
36  LCD_WR_CMD(0x29, 0x001C); // R29=000C when R12=009D;VCM[5:0] for VCOMH
37  LCD_WR_CMD(0x2B, 0x000D); // Frame Rate = 91Hz
38  _delay_ms(50); // Delay 50ms
39  LCD_WR_CMD(0x20, 0x0000); // GRAM horizontal Address
40  LCD_WR_CMD(0x21, 0x0000); // GRAM Vertical Address
41  /***** Adjust the Gamma Curve *****/
42  LCD_WR_CMD(0x30, 0x0007);
43  LCD_WR_CMD(0x31, 0x0302);
44  LCD_WR_CMD(0x32, 0x0105);
45  LCD_WR_CMD(0x35, 0x0206);
46  LCD_WR_CMD(0x36, 0x0808);
47  LCD_WR_CMD(0x37, 0x0206);
48  LCD_WR_CMD(0x38, 0x0504);
49  LCD_WR_CMD(0x39, 0x0007);
50  LCD_WR_CMD(0x3C, 0x0105);
51  LCD_WR_CMD(0x3D, 0x0808);
52  /*****Set GRAM area *****/
53  LCD_WR_CMD(0x50, 0x0000); // Horizontal GRAM Start Address
54  LCD_WR_CMD(0x51, 0x00EF); // Horizontal GRAM End Address
55  LCD_WR_CMD(0x52, 0x0000); // Vertical GRAM Start Address
56  LCD_WR_CMD(0x53, 0x013F); // Vertical GRAM Start Address
57  LCD_WR_CMD(0x60, 0xA700); // Gate Scan Line
58  LCD_WR_CMD(0x61, 0x0001); // NDL, VLE, REV
59  LCD_WR_CMD(0x6A, 0x0000); // set scrolling line

```

```

60      /***** Partial Display Control *****/
61      LCD_WR_CMD(0x80, 0x0000);
62      LCD_WR_CMD(0x81, 0x0000);
63      LCD_WR_CMD(0x82, 0x0000);
64      LCD_WR_CMD(0x83, 0x0000);
65      LCD_WR_CMD(0x84, 0x0000);
66      LCD_WR_CMD(0x85, 0x0000);
67      /***** Panel Control *****/
68      LCD_WR_CMD(0x90, 0x0010);
69      LCD_WR_CMD(0x92, 0x0000);
70      LCD_WR_CMD(0x93, 0x0003);
71      LCD_WR_CMD(0x95, 0x0110);
72      LCD_WR_CMD(0x97, 0x0000);
73      LCD_WR_CMD(0x98, 0x0000);
74      LCD_WR_CMD(0x07, 0x0133); // 262K color and display ON
75  }
76  // End of LCD_init( )

```

源代码 3

4.2.4 函数 LCD_WR_CMD()说明

本函数参考“写寄存器时序图”（见图 10）。（时序图引用自驱动 IC 的 Datasheet）

```

1  typedef unsigned char uchar;
2  typedef unsigned int uint;
3  #define WR_DATA    PORTB
4  /*****
5  /*  函数功能描述:
6  /*  将 16bit 参数 val, 写入地址为
7  /*  index 的寄存器中。
8  /*****
9  /*  参数 1: index, 类型: 8bit
10 /*  参数说明: 目标寄存器地址
11 /*****
12 /*  参数 2: val, 类型: 16bit
13 /*  参数说明: 要写入的数据
14 /*****
15 void LCD_WR_CMD(uchar index,uint val)
16 {

```

```

17     RS_0;                // 写寄存器
18     CS_0;                // 片选有效
19
20     // WR_DATA = 0x00;    // 由于寄存器地址是 8bit,所以高 8 位无效
21     WR_0;
22     WR_1;                // 产生一次写脉冲
23
24     WR_DATA = index;     // 写低 8 位
25     WR_0;
26     WR_1;
27
28     RS_1;                // 写数据
29
30     WR_DATA = (uchar)(val>>8); // 先写高 8 位
31     WR_0;
32     WR_1;
33
34     WR_DATA = (uchar)val; // 后写低 8 位
35     WR_0;
36     WR_1;
37
38     CS_1;                // 片选无效
39 }
40 // End of LCD_WR_CMD()

```

源代码 4

4.2.5 显示彩条

完成以上初始化工作之后，就可以对液晶屏进行正常操作了。以下代码演示了在屏幕上均匀地画出 8 种颜色的彩条：

```

1     uint color[]={0xf800,0x07e0,0x001f,0xffe0,0x0000,0xffff,0x07ff,0xd343};
2     void LCD_test(void)
3     {
4         uint i,j;
5         LCD_WR_CMD(0x20,0);        // X 坐标
6         LCD_WR_CMD(0x21,0);        // Y 坐标
7         LCD_WR_REG(0x22);          // 选中 GRAM 寄存器
8         for(i=0; i<8; i++)
9             for(j=0; j<9600; j++)
10                 LCD_WR_DATA(color[i]); // 向 GRAM 写数据
11     }

```

源代码 5

4.2.6 函数 LCD_WR_REG()说明

```
1  /******  
2  /* 选中地址为 index 的 */  
3  /* 寄存器          */  
4  /******  
5  void LCD_WR_REG(uchar index)  
6  {  
7      RS_0;  
8      CS_0;  
9  
10     WR_0;  
11     WR_1;  
12  
13     WR_DATA = index;  
14  
15     WR_0;  
16     WR_1;  
17  
18     CS_1;  
19 }  
20 // End of LCD_WR_REG()
```

源代码 6

4.2.7 函数 LCD_WR_DATA()说明

```
1  /******  
2  /* 写 16bit 数据 val */  
3  /******  
4  void LCD_WR_DATA(uint val)  
5  {  
6      RS_1;  
7      CS_0;  
8  
9      WR_DATA = val>>8;  
10  
11     WR_0;  
12     WR_1;  
13  
14     WR_DATA = val;  
15  
16     WR_0;  
17     WR_1;  
18  
19     CS_1;  
20     RS_0;  
21 }  
22 // End of LCD_WR_DATA( )
```

源代码 7

可以看出，实际上函数 LCD_WR_REG(index)和 LCD_WR_DATA(val)合起来就是函数 LCD_WR_CMD(index, val)。把它们分开的目的是为了加快对 GRAM 的写入速度。当然还有其它的办法。

4.2.8 其它说明

要更加深入地了解如何对该屏进行操作，请仔细阅读相关的 Datasheet，以及网上公开的源码。在这里只谈一下目前我所知道的一些内容：

- ！ 该液晶屏的分辨率为 240X320，默认的坐标原点在屏幕的左上角
- ！ GRAM 接受的数据格式为 RGB565
- ！ 数据最快的写入速度（#WR 写入脉冲） $\geq 8\text{MHz}$
- ！ 可以设置窗口，快速对屏幕的指定区域进行操作

5. 实际显示效果

5.1 彩条



图 5



图 6

5.2 字符



图 7

5.3 位图



图 8



图 9

6. 写寄存器时序图

i80 9-/8-bit System Bus Interface Timing

(a) Write to register

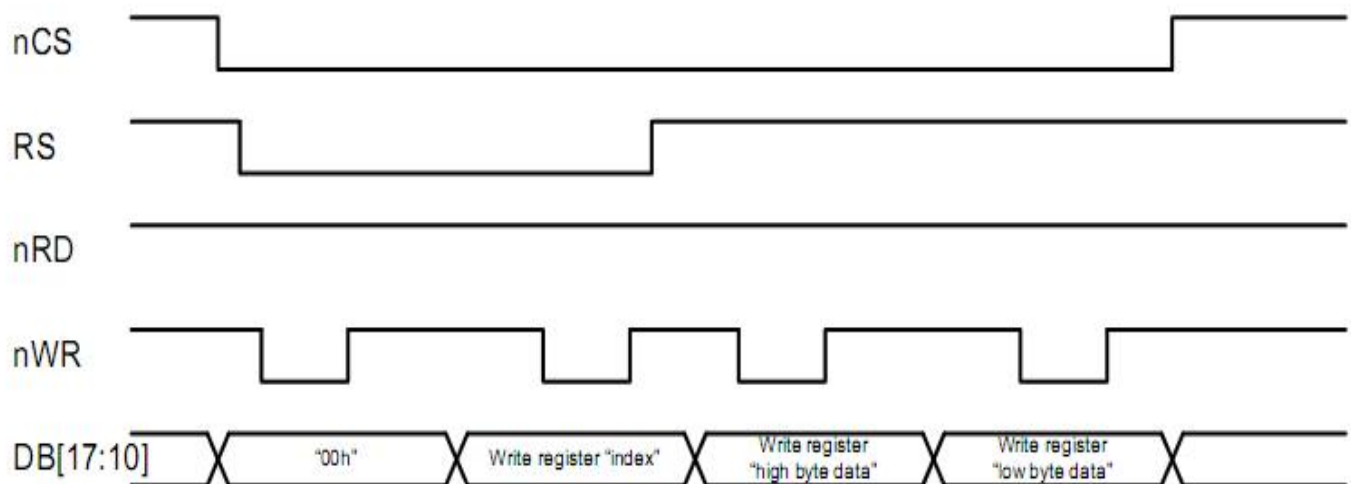


图 10