目录

北方蓝芯 2.4”（Version2.0）TFT 使用文档

一、TFT 和转接板电路的性能指标介绍

该款 TFT 屏采用信利公司的 2.4″超薄超清 2.4″TFT 模组，其采用 R61505U 控制器（该款控制器为日本 RENESAS 公司生产，可以与 ILI9320、ILI9325 相互替换，时序一致），显示屏质量可靠、不带山寨图标、外观小巧、美观，是 2.4″FTF 模组的。

TFT 尺寸：2.4 寸 TFT 彩屏。

分辨率：320x240，262K 色。

转接板上 TFT 接口类型：为了方便 8 为单片机的控制，在硬件电路上已经设定为 8 位数据口；转换板接口可兼容 12864 液晶接口，插上后需根据实际接口来修改程序中的接口定义方可正常显示。

电压类型：支持 5V 或 3.3V 供电，可以与 5V 或 3.3V 单片机一起工作。板上含有 3.3V 稳压芯片，通过跳帽来切换 5V 和 3.3V 工作模式。

SD 卡座：板上载有 SD 卡，电路设计为 SPI 工作模式。

不带触摸功能的 TFT 转接板没有焊接触摸控制电路的相关器件。

注：用 51 内核单片机做 SD 卡实验时建议选用增强型或高档单片机，这样显示速度会大大提高，因为 TFT 常常是与高级控制器一起使用的，用普通 8 位单片机刷屏特别是真彩图片的显示通常很吃力。

二、转接电路上的接口说明

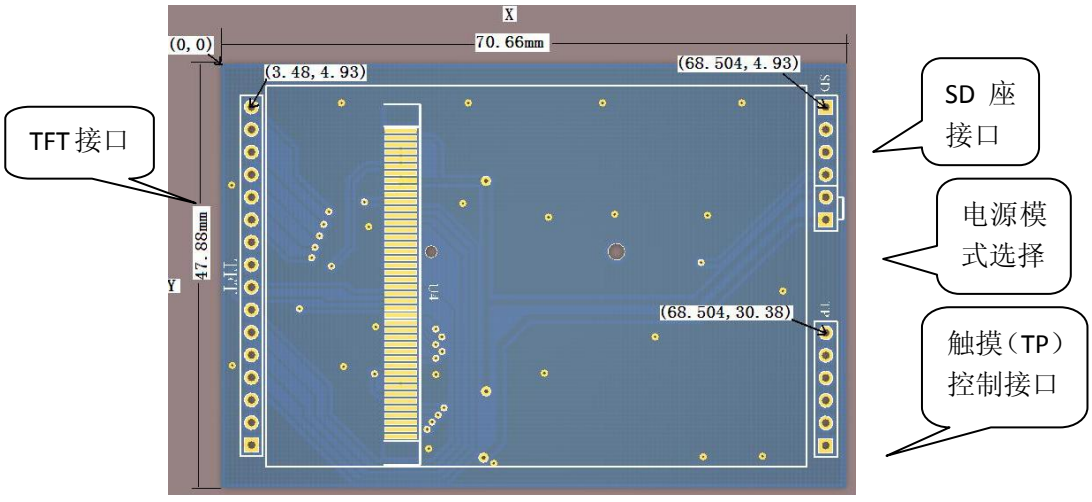

图 1 TFT 转接板的尺寸图（mm）

1、TFT 接口功能说明如下:

REST:液晶复位信号;

CS:片选信号;

DB0～BD7:8 位数据口;

RD:读控制信号;

WR:写控制信号;

RS:命令/数据选择，RS=0 时可读/写命令，RS=1 时不可以读/写命令;

BL_EN:液晶背光控制，BL_EN=0 关背光，BL_EN=1 点亮背光;

POWER:接电源,注意根据接入的电源电压来选择跳帽是否短接，3v 供电时电源模式选择接口 VCC_SEL 短接，5v 供电时 VCC_SEL 断开;

GND:接地;

TFT 屏和 TFT 控制器的规格和具体的参数请参考 “TFT 规格说明书.pdf”和“R61505U_1.0.pdf”文档;

****

2、SD 卡接口功能说明:

CS:片选信号;

DIN:串行数据输入;

SCK:始终信号;

DOUT:串行数据输出。

****

3、触摸屏（Touch panel）控制接口（此部分供带触摸功能 TFT 用户参考）

PENIRQ:中断输出;

DOUT:串行接口引脚，在时钟下降沿数据移出;

BUSY:忙指示，低电平有效;

DIN:串行接口引脚，在时钟上升沿数据移进;

CS:片选信号;

CLK:外部时钟输入。

三、与程序配套的实验接线说明

BL_EN:液晶背光控制，BL_EN 为低电平关背光，BL_EN 接高电平或直接连接 VCC 来点亮背光。

VCC:接电源,具体接5v输入还是3v输入电压,请参看液晶转接板上的丝印说明。

GND:接地

1、程序1-2.4寸彩屏显示 12864信息、程序2-2.4寸彩屏显示渐变颜色彩条、程序3-16x16和32x32字符显示例程: 只需将TFT接口与单片机实验板连接。

DB0¯7	P0.0— P0.7	TFT数据口
CS	P2.2	TFT片选
RES	P2.0	TFT复位
RD	P2.3	TFT读数据/命令
RS	P2.5	TFT数据/命令选择
RW	P2.4	TFT写数据/命令

2、程序4-电子相册-1G 51单片机SD卡读图片到彩屏。

DB0¯7	P0.0— P0.7	TFT数据口
CS	P2.2	TFT片选
RES	P2.0	TFT复位
RD	P2.3	TFT读数据/命令
RS	P2.5	TFT数据/命令选择
RW	P2.4	TFT写数据/命令
SCK	P1.1	SD卡接口连线
DIN	P1.2	SD卡接口连线
DOUT	P1.0	SD卡接口连线
CS	P1.3	SD卡接口连线
KEY	P3.7	连一个按键下翻图片

3、程序5-画板和写字板程序of带触摸TFT、程序6-显示触摸控制芯片当前采样值和显示屏的当前坐标of带触摸TFT。

DB0¯7	P0.0— P0.7	TFT数据口
CS	P2.2	TFT片选
REST	P2.0	TFT复位
RD	P2.3	TFT读数据/命令
RS	P2.5	TFT数据/命令选择
WR	P2.4	TFT写数据/命令
DCLK	P1.0	触摸外部时钟输入
CS	P1.1	触摸使能信号
DIN	P1.2	触摸输入数据口
BUSY	P1.3	触摸忙信号
DOUT	P1.4	触摸输出数据口
Penirq	P3.2	触摸中断

四、SD卡使用方法

程序4的功能是将SD卡中存储的图片显示到TFT，故此处SD卡主要用来存储图片。其使用方法说明如下:

2.4寸彩屏分辨率为320x240，程序控制中用16bit表示一个点的颜色，即为2个字节。

320x240x2=153600 字节，即为 150k，所以如果完整写一幅图片，需要 150k 存储空间，51 系列单片机一般最大内部 ROM 为 64k，显然是不够用的，所以需要 SD 卡存储图片。 SD 卡使用如下，SD 卡要格式化成 FAT(即 FAT16 格式)，然后把需要显示的大小为 320x240 像素的 bmp 格式图片通过 Image2LCD 软件转换成 bin 格式并存储到 SD 卡中。

注意：SD 卡中不能存储其他任何文件，使用之前一般需要先格式化，否则可能导致图片输出不完整。

图片存储到 SD 卡后，可以通过 winhex 软件查看各图片存储的地址。打开软件如图 2 所示，点击第一张图片“车.bin”，可以看到右端第 1 扇区地址是 520，这是数据区最小的地址。所有图片在 SD 卡中是依次存放的，读图片也是依次进行的。看上图中左下角圈起来的 2 个数字，上面的物理扇区编号，下面的是逻辑扇区编号，配套的程序中由于没有使用完整的 FAT 格式所以这里我们选择物理编号 769，那么对应的地址就是 769x512=393728，这个是 1G 卡 FAT 格式化后的初始数据。不同容量的 SD 卡的初始地址不同，请使用 winhex 软件查看对应的物理扇区编号，并计算出对应的地址，然后在样例中图示标记位置进行更改。

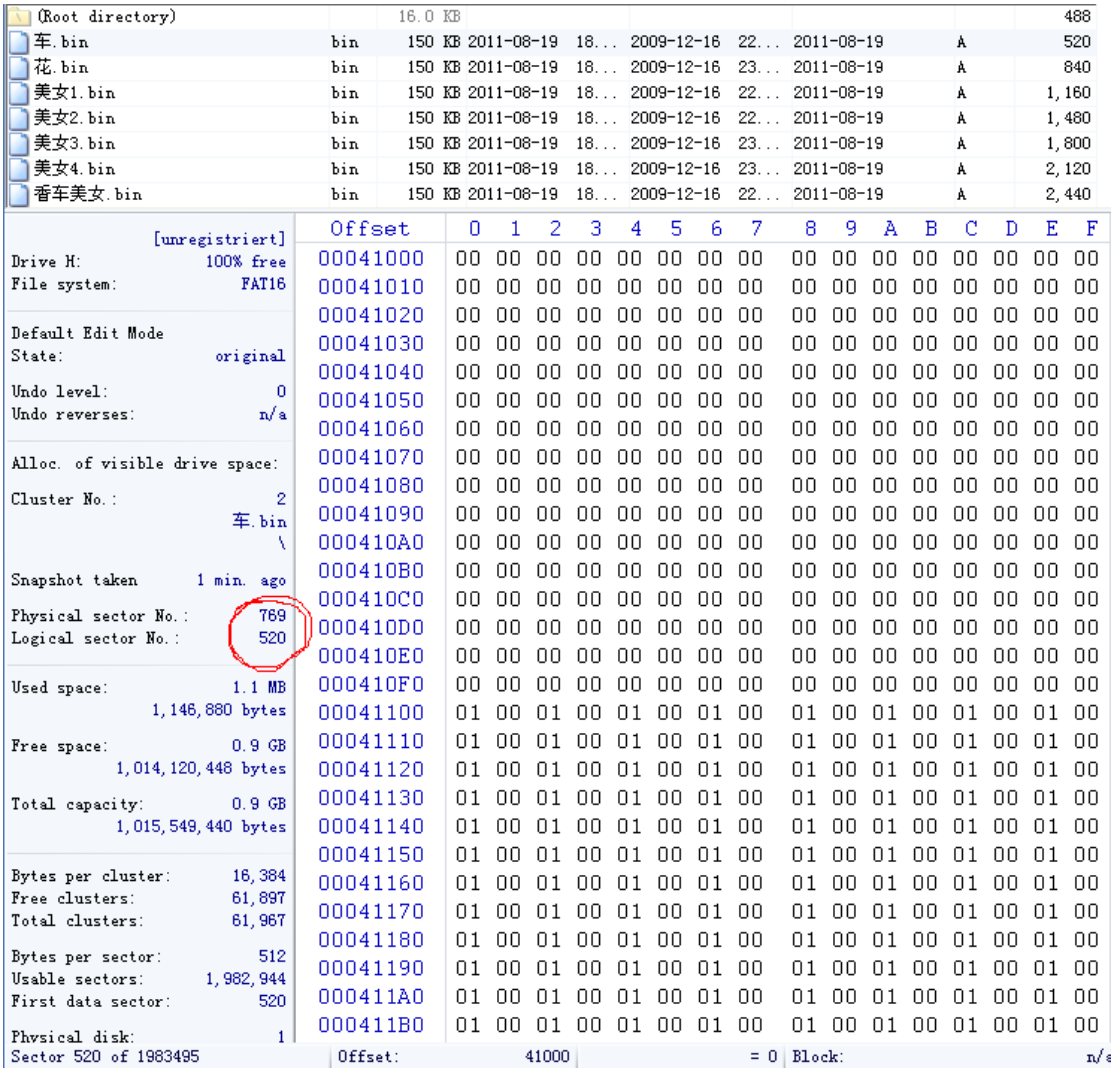

图 2

```
unsigned int x,y; //定义液晶屏坐标
unsigned long j;  //执行循环需要的临时变量
 unsigned int i;
 unsigned long AddTemp=393728;//SD卡地址第一个数据物理地址初始值，可以用winhex查看，
                              //这里是512扇区，769x512=393728，根据实际SD卡内容更改
   CS=1;
   delayms(5);
   RES=0;
   delayms(5);
   RES=1;
   delayms(5);
   ILI9325_Initial();//液晶屏初始化
   SdInit();         //SD卡初始化
while(1)
{

for(j=0;j<300;j++)    //300表示一幅图片含有300x512字节的信息
   {
   SdReadBlock(DATA,AddTemp+(j*512),512);//每次读出512字节放到缓冲区
   for(i=0;i<256;i++)                    //然后写到液晶屏，可以显示256个像素，每个像素16位即2个字节
      {
      LCD_SetPos(x,x,y,y);
      Write_Data(DATA[2*i+1],DATA[2*i]);
      x++;
      if(x==240)                         //检测是否写到屏的边缘 240x320
        {
        y++;
        x=0;
        if(y==320)
          y=0;
        }
      }
    }
    AddTemp = AddTemp+((j+20)*512);          //写完一幅图片后把SD地址加300x512到下一个图片地址
    while(KEY);                             //等待按键按下继续执行循环显示下一幅图片，如果没有按下则等待
   // while(AddTemp>2621440)
    // j=0,AddTemp=721408;
}
}
```

图 3

图 3 中所示的数据简略说明如下：

按键可以定义到单片机任意 I/O 口，这里表示写完一幅图片后如果按下按键，则显示下一张图片，否则停留在当前图片。

“j+20”的解释如下：一张完整的图片占用 150k 存储空间，每个扇区大小为 512，也就是说存储一张图片需要 300 个扇区，据此可推，第一张图片的逻辑地址为 520，下一张图片的逻辑地址则应该为 520+300=820，但通过 winhex 可以看出实际的逻辑地址是 840，所以要通过在程序中加 20 来跳过这个空白区。这里的程序仅作从 SD 卡中读取图片测试用，实际 SD 卡读写需要读出引导程序，然后确定下一组数据的位置，这需要复杂的写 FAT 文件系统，有兴趣的可另去研究。

五、TFT 配带程序实现的显示结果

图为手机拍摄，图片色相有较严重的失真^_^，权作示意的作用，**非实际显示效果！！！**

2.4 寸彩屏显示渐变颜色彩条

51 单片机-1G SD 卡读图片到 TFT 真彩液晶， p3.7 接按键，可通过按键切换为下一图片

字符显示例程，字符显示包含了中英文，16x16 以及 32X32 点阵字符

2.4 寸彩屏显示 12864 信息，每一个方格都可容纳 128X64 点阵的信息容量

附录——内部 TFT 控制芯片 R61505U 的初始化程序：

```
void R61505U_Initial(void)
{
    Write_Cmd_Data(0x0000,0x0000);
    Write_Cmd_Data(0x0000,0x0000);
    Write_Cmd_Data(0x0000,0x0000);
    Write_Cmd_Data(0x0000,0x0000);
    Write_Cmd_Data(0x0010,0x0600);      // SLP=0,
      delayms(30);
    Write_Cmd_Data(0x0007,0x0000);
    Write_Cmd_Data(0x0012,0x011A);
    Write_Cmd_Data(0x00A4,0x0001);
    Write_Cmd_Data(0x0008,0x020E);    // FP,BP
    Write_Cmd_Data(0x000A,0x0008);
    Write_Cmd_Data(0x000D,0x0008);
    Write_Cmd_Data(0x0030,0x0707);
    Write_Cmd_Data(0x0031,0x0007);
    Write_Cmd_Data(0x0032,0x0603);
    Write_Cmd_Data(0x0033,0x0700);
    Write_Cmd_Data(0x0034,0x0202);
    Write_Cmd_Data(0x0035,0x0002);
    Write_Cmd_Data(0x0036,0x1E00);
    Write_Cmd_Data(0x0037,0x0707);
    Write_Cmd_Data(0x0038,0x0000);
    Write_Cmd_Data(0x0039,0x0000);
    Write_Cmd_Data(0x003A,0x0707);
    Write_Cmd_Data(0x003B,0x0000);
    Write_Cmd_Data(0x003C,0x0007);
    Write_Cmd_Data(0x003D,0x0000);
          delayms(30);
      Write_Cmd_Data(0x0011,0x0007);
      Write_Cmd_Data(0x0060,0x2700);
      Write_Cmd_Data(0x0090,0x0016);    // DIVI, RTNI
    Write_Cmd_Data(0x0017,0x0001);
      Write_Cmd_Data(0x0019,0x0000); // TBT[1:0]
          delayms(30);
    Write_Cmd_Data(0x0010,0x16B0);
    Write_Cmd_Data(0x0012,0x011A);
      delayms(30);
    Write_Cmd_Data(0x0013,0x1800); // VDV[4:0]
    Write_Cmd_Data(0x002A,0x000E); // VCMSEL, VCM2[4:0]
    Write_Cmd_Data(0x0029,0x000E); // VCM1[4:0]
```

```
   delayms(30);
Write_Cmd_Data(0x0012,0x013A); // VCOMR[0], VREG1R, PSON, PON, VRH[3:0]
   delayms(100);
Write_Cmd_Data(0x0050,0x0000);
Write_Cmd_Data(0x0051,0x00EF);
Write_Cmd_Data(0x0052,0x0000);
Write_Cmd_Data(0x0053,0x013F);
Write_Cmd_Data(0x0020,0x0000);
Write_Cmd_Data(0x0021,0x0000);
Write_Cmd_Data(0x0061,0x0001);
Write_Cmd_Data(0x006A,0x0000);
Write_Cmd_Data(0x0080,0x0000);
Write_Cmd_Data(0x0081,0x0000);
Write_Cmd_Data(0x0082,0x0000);
Write_Cmd_Data(0x0083,0x0000);
Write_Cmd_Data(0x0084,0x0000);
Write_Cmd_Data(0x0085,0x0000);
Write_Cmd_Data(0x0092,0x0300);
Write_Cmd_Data(0x0093,0x0005);
Write_Cmd_Data(0x0095,0x0000);
Write_Cmd_Data(0x0097,0x0000);
Write_Cmd_Data(0x0098,0x0000);
Write_Cmd_Data(0x0001,0x0100);
Write_Cmd_Data(0x0002,0x0700);
Write_Cmd_Data(0x0003,0x1030);
Write_Cmd_Data(0x0004,0x0000);
Write_Cmd_Data(0x000C,0x0000);
Write_Cmd_Data(0x000F,0x0000);
delayms(30);
Write_Cmd_Data(0x0007,0x0001);
Write_Cmd_Data(0x0007,0x0021);
delayms(30);
Write_Cmd_Data(0x0010,0x16B0);   // Write final user's setting values to BT bits
Write_Cmd_Data(0x0011,0x0007);   // Write final user's setting values to VC bits
delayms(30);
Write_Cmd_Data(0x0007,0x0061);
delayms(30);
Write_Cmd_Data(0x0007,0x0173);
delayms(30);
```