3.2inch 320x240 Touch LCD (C) 用户手册

简介

驱动芯片	LCD 控制芯片 ILI9325，触摸面板控制芯片 XPT2046
类型	TFT
接口	LCD：16bit 并行，触摸面板：SPI
背光	LED 背光
色阶指数	65536 色
分辨率	320x240 （Pixel）

目录

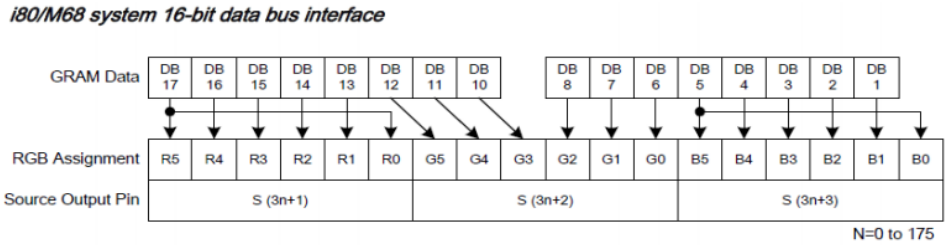
1. 硬件资源

1.1 ILI9325

- ILI9325 是一个 240x320（RGB）分辨率、262144 色的 TFT 液晶显示屏的驱动芯片；172820 (240 x 320x 18/8) 字节的 RAM。每个像素点深度可以达到 18 位。
- ILI9325 有以下几种数据接口模式：
  1) i80-system MPU 接口(8-/9-/16-/18-bit bus width)
  2) VSYNC 接口(system interface + VSYNC, internal clock, DB[17:0])
  3) serial data transfer 接口(SPI)
  4) RGB 6-/16-/18-bit 接口(DOTCLK, VSYNC, HSYNC, ENABLE, DB[17:0]).

此屏的 ILI9325 的 18 位 RGB 赋值与 LCD GRAM 的对应关系如图所示：

i80/M68 system 16-bit data bus interface



从图中可以看出，ILI9325 在 16 位模式下面，GRAM Data 有用的是：D17~D10 和 D8~D1，D9 和 D0 没有用到，实际上在我们 LCD 模块里面，ILI9341 的 D9 和 D0 没有引出，ILI9325 的 D17~D10 和 D8~D1 对应 MCU 的 D15~D0。MCU 的 16 位数据，最低 5 位代表蓝色，中间 6 位为绿色，最高 5 位为红色；数值越大，表示该颜色越深。

重要寄存器介绍

寄存器详细介绍请参阅 ILI9325 的 datasheet。这里只介绍一些重要的寄存器设置：

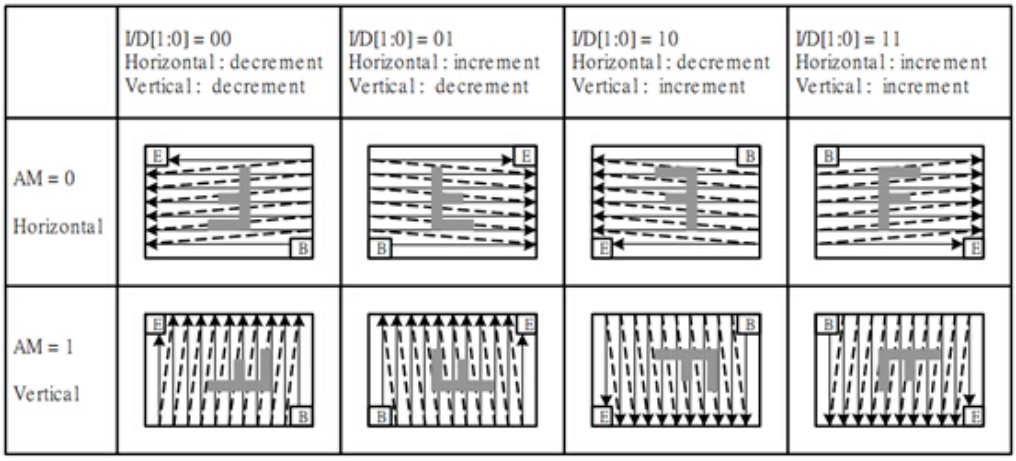
输入设置（R03h）：

R/W	RS	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
W	1	TRI	DFM	0	BGR	0	0	0	0	ORG	0	I/D1	I/D0	AM	0	0	0
Default		0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

AM：控制 GRAM 的更新方向

- 当 AM=0 地址在水平写入方向得以更新
- 当 AM=1 地址在垂直写入方向得以更新

I/D[1:0]： 当更新一个像素数据时， I/D[1:0]位控制地址计数器（AC）自动增加或者减少 1。详细请见下图：



ORG ： 当窗口地址区域产生后，原点地址的移动根据 ID 的设定。当使用高速写 RAM 模式写数据到窗口地址区域这个功能被使能。

- ORG=0 ：原点地址不会移动，在这种情况下，在窗口地址区域根据 GRAM 的地址映射指定一个地址开始写操作。
- ORG=1 ： 原始地址为“00000h” 根据 ID[1 :0]的 设定来移动。

BGR : 根据被写入的数据交换 R 和 B 的顺序

- BGR=0 ： 根据 RGB 的顺序写入像素数据
- BGR=1 ： 交换 RGB 数据为 BGR 写到 GRAM

GRAM 水平垂直变址设置(R20h, R21h)

水平位置寄存器变址 0x20 垂直变址寄存器变址 0x21 GRAM Horizontal/Vertical Address Set (R20h, R21h)

R/W	RS	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
W	1	0	0	0	0	0	0	0	0	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
W	1	0	0	0	0	0	0	0	AD16	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
Default		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

AD[16:0]：用于设置变址计数器（AC）的初始化数值。当数据写入内部的 GARM 中时，变址计数器（AC）会根据 AM 和 I/D 位的设置来自动的更新其数值。当从内部 GRAM 中读数据时变址计数器不会自动更新。

GRAM 内部数据映射图

AD[16:0]	GRAM Data Map
0x00000--0x000EF	第 1 行 GRAM Data
0x00100--0x001EF	第 2 行 GRAM Data
0x00200--0x002EF	第 3 行 GRAM Data
0x00300--0x003EF	第 4 行 GRAM Data
……	……
0x13D00--0x13DEF	第 318 行 GRAM Data
0x13E00--0x13EEF	第 319 行 GRAM Data
0x13F00--0x13FEF	第 320 行 GRAM Data

写数据到 GRAM (R22h)

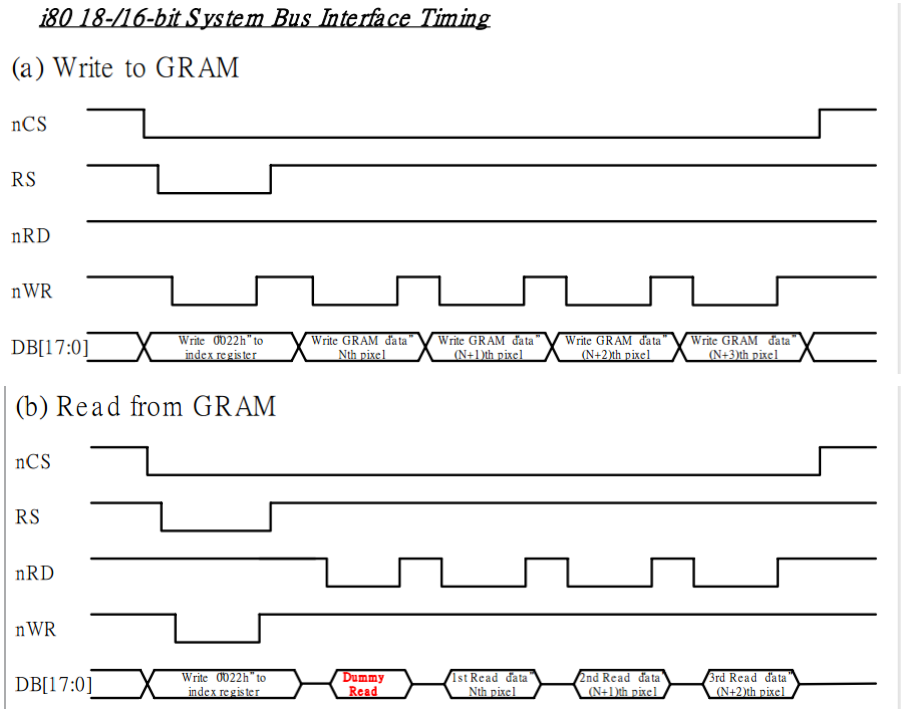
R/W	RS	D17	D16	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
W	1	RAM write data (WD[17:0], the DB[17:0] pin assignment differs for each interface.																	

该寄存器是 GRAM 的访问端口，当通过这个寄存器更新显示数据的时候，地址计数器会自动增加或者减少。

GRAM 地址映射和读/写

ILI9325 有一个容量为 172800 bytes 的内部图片 RAM (GRAM)，用来存储显示数据。一像素由十八位数据构成， GRAM 可以通过 i80 系统接口，SPI 或者是 RGB 接口来访问，以下是 GRAM 在 i80 系统接口下读写的时序“

i80 18-/16-bit System Bus Interface Timing



1.2 XPT2046

- XPT2046 是一款 4 线制电阻式触摸屏控制器，内含 12 位分辨率 125KHz 转换速率逐步逼近型 A/D 转换器。
- XPT2046 支持从 1.5V 到 5.25V 的低电压 I/O 接口。
- XPT2046 能通过执行两次 A/D 转换查出被按的屏幕位置，除此之外，还可以测量加在触摸屏上的压力。内部自带 2.5V 参考电压，可以作为辅助输入、温度测量和电池监测之用，电池监测的电压范围可以从 0V 到 5V。
- XPT2046 片内集成有一个温度传感器。在 2.7V 的典型工作状态下，关闭参考电压，功耗可小于 0.75mW。XPT2046 采用微小的封装形式：TSSOP-16,QFN-16 和 VFBGA－48。 工作温度范围为-40℃～+85℃。与 ADS7846、TSC2046、AK4182A 完全兼容。

2. 硬件说明

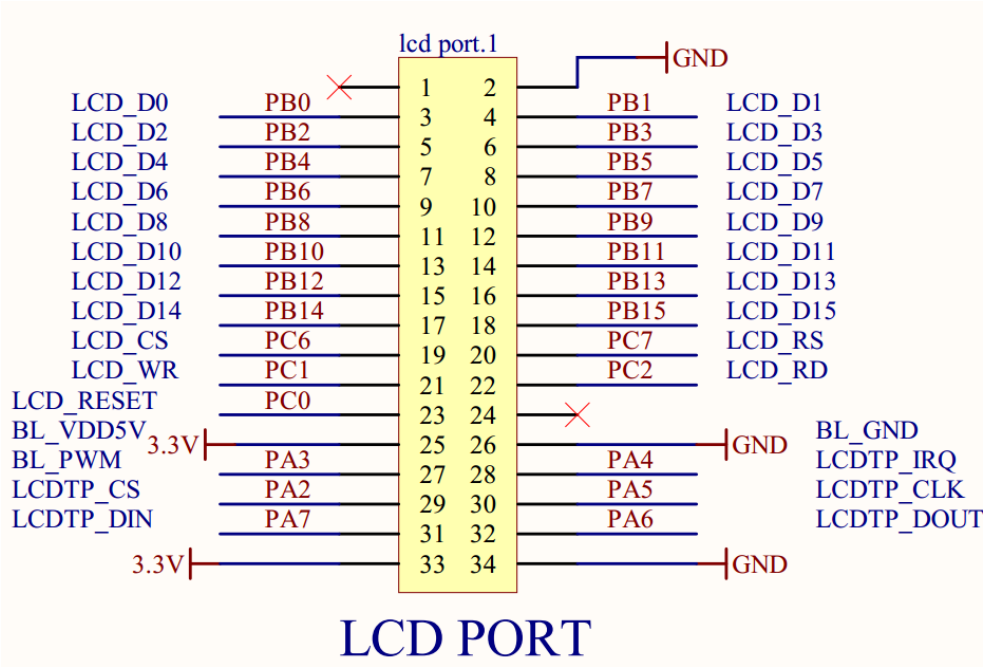
引脚号	标识	描述	功能
1	5V	5V 电源	当 5V 供电时（1，2 脚接 5V 电源），3.3V 端（33，34 脚输出 3.3V 电压）
2	GND	接地	GND
3	D0	数据线	D0-D15
4	D1		
5	D2		
6	D3		
7	D4		
8	D5		
9	D6		
10	D7		
11	D8		
12	D9		
13	D10		
14	D11		
15	D12		
16	D13		
17	D14		
18	D15		
19	CS	LCD 片选信号	低电平选择 LCD
20	RS	指令/数据 寄存器选择	RS = 0：指令寄存器 RS = 1：数据寄存器
21	WR	写动作	WR = 0，RD = 1
22	RD	读动作	WR = 1，RD = 0
23	RESET	芯片重启	低电平重启芯片
24	NC		
25	BLVCC	5V 或 3.3V	背光灯 VCC
26	BLGND	接地	背光灯 GND
27	BLCNT	背光灯亮度调节	可以使用 PWM 来控制背光灯亮度
28	TP_IRQ	触摸面板中断	检测到触摸面板有按下则为低电平
29	TP_CS	触摸面板片选信号	低电平选择触摸面板
30	TP_SCK	触摸面板 SPI 时钟信号	连接到 SPI 的 SCK
31	TP_SI	触摸面板 SPI 数据输入	连接到 SPI 的 MOSI
32	TP_SO	触摸面板 SPI 数	连接到 SPI 的 MISO

		据输出	
33	3.3V	+3.3 电源	当 3.3V 供电时（33，34 脚输入 3.3V） 1,2 脚悬空
34	GND	接地	

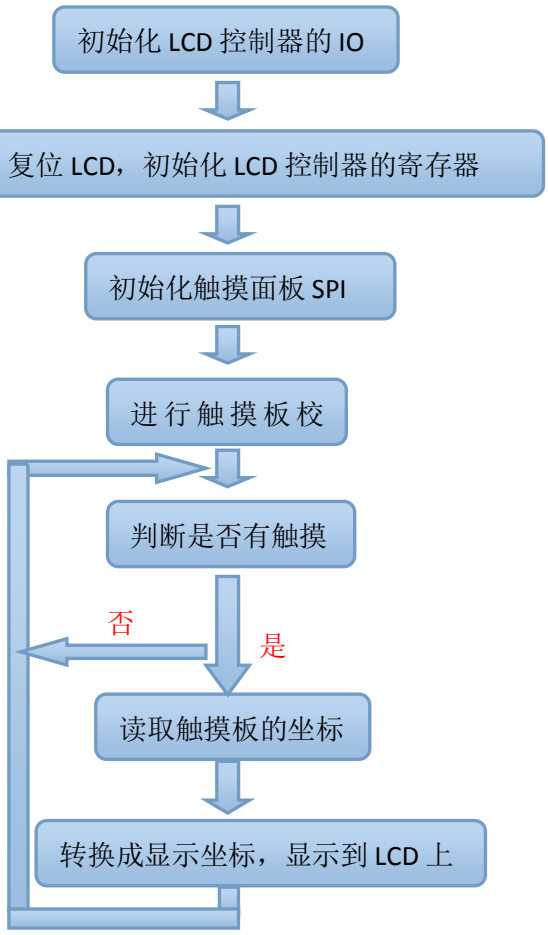
3. 示例程序

本手册使用主控芯片 STM32F103RCT6 的开发板说明本款 LCD 的基本使用方法。用户也可以采用其他类似的开发板进行开发。

3.2inch 320x240 Touch LCD (C)和 STM32F103RCT6 连接接口图：



程序流程：


初始化 LCD 控制器的 IO
复位 LCD，初始化 LCD 控制器的寄存器
初始化触摸面板 SPI
进 行 触 摸 板 校
判断是否有触摸
否
是
读取触摸板的坐标
转换成显示坐标，显示到 LCD 上

源代码解析：

```
/*下面宏定义的是图像的旋转角度*/
//#define DISP_ORIENTATION                    0
//#define DISP_ORIENTATION                    90
//#define DISP_ORIENTATION                    180
#define DISP_ORIENTATION                    270
#define Set_Cs       GPIO_SetBits(GPIOC, GPIO_Pin_6);    //CS=1;
#define Clr_Cs       GPIO_ResetBits(GPIOC, GPIO_Pin_6);  //CS=0;

#define Set_Rs       GPIO_SetBits(GPIOC, GPIO_Pin_7);    //RS=1;
#define Clr_Rs       GPIO_ResetBits(GPIOC, GPIO_Pin_7);  //RS=0;

#define Set_nWr      GPIO_SetBits(GPIOC, GPIO_Pin_1);    //WR=1;
#define Clr_nWr      GPIO_ResetBits(GPIOC, GPIO_Pin_1);  //WR=0;

#define Set_nRd      GPIO_SetBits(GPIOC, GPIO_Pin_2);    //RD=1;
#define Clr_nRd      GPIO_ResetBits(GPIOC, GPIO_Pin_2);  // RD=0;
/* 写命令函数 */
__inline void LCD_WriteIndex(uint16_t index)
{
    Clr_Rs;           //RS=0
    Set_nRd;          //RD=0
    LCD_Delay(0);      //延时
    GPIOB->ODR = index; /*写命令 */
    LCD_Delay(0);      //延时
    Clr_nWr;          //WR=0
    Set_nWr;          //WR=1
}
/* 写数据函数 */
__inline void LCD_WriteData(uint16_t data)
{
    Set_Rs;            //RS=1
    LCD_Delay(0);       //延时
    GPIOB->ODR = data;   /*写数据*/
    LCD_Delay(0);      //延时
    Clr_nWr;          //WR=0
    Set_nWr;          //WR=1
}
/* 读数据函数 */
__inline uint16_t LCD_ReadData(void)
{
    uint16_t value;
    Set_Rs;
```

```
    Set_nWr;
    Clr_nRd;
    GPIOB->CRH = 0x44444444;   //设置 PB0-PB15 为输入
    GPIOB->CRL = 0x44444444;
    value = GPIOB->IDR;        //读取数据
    GPIOB->CRH = 0x33333333;   //设置 PB0-PB15 为输出
    GPIOB->CRL = 0x33333333;
    Set_nRd;
    return value;
}
/*************************************************************
**************
```

指定的地址写入数据，LCD_Reg 是地址，LCD_RegValue 是写入的值。

```
*************************************************************
*************/
__inline void LCD_WriteReg(uint16_t LCD_Reg,uint16_t LCD_RegValue)
{
Clr_Cs;
LCD_WriteIndex(LCD_Reg);        //写指令；即要写入数据的地址；
LCD_WriteData(LCD_RegValue);    //数据写入；
Set_Cs;
}

/*************************************************************
**************
```

从指定的地址读取数据，LCD_Reg 是地址，函数返回读取出来的值。

```
*************************************************************
*************/
__inline uint16_t LCD_ReadReg(uint16_t LCD_Reg)
{
uint16_t LCD_RAM;
Clr_Cs;
LCD_WriteIndex(LCD_Reg);    //写指令；即要读出数据的地址；
LCD_RAM = LCD_ReadData();  //数据读出；
Set_Cs;
return LCD_RAM;
}
```

//以上是最基本的读写函数；IO 模拟操作，如果想 STM32 的 FSMC 来控制的，参考另外一个例程 LCD + TouchPanel(8080 FSMC)

```
/*************************************************************
**************
```

LCD 寄存器的初始化，以下寄存器的初始化值由 LCD 原厂家提供，按照如下配置就可以正常显示，寄存器请参考芯片手册。

```
*************************************************************
```

```
**************/
void LCD_Initializtion(void)
{
  uint16_t DeviceCode;
  LCD_Configuration();                  //管脚初始化
  GPIO_ResetBits(GPIOC, GPIO_Pin_0);   /* LCD 复位*/
  delay_ms(100);
  GPIO_SetBits(GPIOC, GPIO_Pin_0);
  GPIO_SetBits(GPIOA, GPIO_Pin_3);     /*使能背光 */
  DeviceCode = LCD_ReadReg(0x0000);    /* 读取 ID*/
  if( DeviceCode == 0x9325 || DeviceCode == 0x9328 )
  {
    LCD_WriteReg(0x00e7,0x0010);
    LCD_WriteReg(0x0000,0x0001);
    LCD_WriteReg(0x0001,(0<<10)|(1<<8));
    LCD_WriteReg(0x0002,0x0700);
    #if (DISP_ORIENTATION == 0)
    LCD_WriteReg(0x0003,(1<<12)|(1<<5)|(1<<4)|(0<<3));
    #elif (DISP_ORIENTATION == 90)
    LCD_WriteReg(0x0003,(1<<12)|(0<<5)|(1<<4)|(1<<3));
    #elif (DISP_ORIENTATION == 180)
    LCD_WriteReg(0x0003,(1<<12)|(0<<5)|(0<<4)|(0<<3));
    #elif (DISP_ORIENTATION == 270)
    LCD_WriteReg(0x0003,(1<<12)|(1<<5)|(0<<4)|(1<<3));
    #endif
    LCD_WriteReg(0x0004,0x0000);
    LCD_WriteReg(0x0008,0x0207);
    LCD_WriteReg(0x0009,0x0000);
    LCD_WriteReg(0x000a,0x0000);
    LCD_WriteReg(0x000c,0x0001);
    LCD_WriteReg(0x000d,0x0000);
    LCD_WriteReg(0x000f,0x0000);
    /* Power On sequence */
    LCD_WriteReg(0x0010,0x0000);
    LCD_WriteReg(0x0011,0x0007);
    LCD_WriteReg(0x0012,0x0000);
    LCD_WriteReg(0x0013,0x0000);
    delay_ms(50);  /* delay 50 ms */
    LCD_WriteReg(0x0010,0x1590);
    LCD_WriteReg(0x0011,0x0227);
    delay_ms(50);  /* delay 50 ms */
    LCD_WriteReg(0x0012,0x009c);
    delay_ms(50);  /* delay 50 ms */
    LCD_WriteReg(0x0013,0x1900);
```

```
    LCD_WriteReg(0x0029,0x0023);
    LCD_WriteReg(0x002b,0x000e);
    delay_ms(50);  /* delay 50 ms */
    delay_ms(50);  /* delay 50 ms */
    LCD_WriteReg(0x0030,0x0007);
    LCD_WriteReg(0x0031,0x0707);
    LCD_WriteReg(0x0032,0x0006);
    LCD_WriteReg(0x0035,0x0704);
    LCD_WriteReg(0x0036,0x1f04);
    LCD_WriteReg(0x0037,0x0004);
    LCD_WriteReg(0x0038,0x0000);
    LCD_WriteReg(0x0039,0x0706);
    LCD_WriteReg(0x003c,0x0701);
    LCD_WriteReg(0x003d,0x000f);
    delay_ms(50);  /* delay 50 ms */
    LCD_WriteReg(0x0050,0x0000);
    LCD_WriteReg(0x0051,0x00ef);
    LCD_WriteReg(0x0052,0x0000);
    LCD_WriteReg(0x0053,0x013f);
    LCD_WriteReg(0x0060,0xa700);
    LCD_WriteReg(0x0061,0x0001);
    LCD_WriteReg(0x006a,0x0000);
    LCD_WriteReg(0x0080,0x0000);
    LCD_WriteReg(0x0081,0x0000);
    LCD_WriteReg(0x0082,0x0000);
    LCD_WriteReg(0x0083,0x0000);
    LCD_WriteReg(0x0084,0x0000);
    LCD_WriteReg(0x0085,0x0000);
    LCD_WriteReg(0x0090,0x0010);
    LCD_WriteReg(0x0092,0x0600);
    LCD_WriteReg(0x0093,0x0003);
    LCD_WriteReg(0x0095,0x0110);
    LCD_WriteReg(0x0097,0x0000);
    LCD_WriteReg(0x0098,0x0000);
    /* display on sequence */
    LCD_WriteReg(0x0007,0x0133);
  }
  delay_ms(50);
}
/*****************************************************************
**************
```

设置显示窗口的位置 X、Y；

```
*****************************************************************
************/
```

```
static void LCD_SetCursor( uint16_t Xpos, uint16_t Ypos )
{
  uint16_t temp;
  #if (DISP_ORIENTATION == 0)
  #elif (DISP_ORIENTATION == 90)
    temp = Xpos;
    Xpos =Ypos;
    Ypos = MAX_X - 1 - temp;
  #elif (DISP_ORIENTATION == 180)
    Xpos = MAX_X - 1 - Xpos;
    Ypos = MAX_Y - 1 - Ypos;
  #elif (DISP_ORIENTATION == 270)
    temp = Ypos;
    Ypos = Xpos;
    Xpos = MAX_Y - 1 - temp;
  #endif
  LCD_WriteReg(0x0020, Xpos); //水平位置 X 的设置
  LCD_WriteReg(0x0021, Ypos); //垂直位置 Y 的设置
  }
/*********************************************************************
**************
清屏函数，作用是让整个屏显示某一种颜色，
*********************************************************************
************/
void LCD_Clear(uint16_t Color)
{
  uint32_t index=0;
  LCD_SetCursor(0,0);    //设置光标的初始位置 X、Y
  Clr_Cs;
  LCD_WriteIndex(0x0022);//开始向 GRAM 写数据
  for( index = 0; index < MAX_X * MAX_Y; index++ )
  {
    LCD_WriteData(Color);
  }
  Set_Cs;
}

int main(void)
{
  //延时和初始化系统
  LCD_Initializtion();     //LCD 初始化
  //LCD 触摸板初始化
  LCD_Clear(Red);      //清屏为红色
```

```
  //您可以编写函数来校准屏幕。
  /* Infinite loop */
  while (1)
  {
        //您可以编写函数，把触摸点坐标显示在 LCD 上
  }
}
```