7inch Resistive Touch LCD

用户手册

中文官方网站：www.waveshare.net
英文官方网站：www.wvshare.com
资料下载网站：www.waveshare.net/wiki

深圳微雪电子有限公司

目录

1. 模块简介

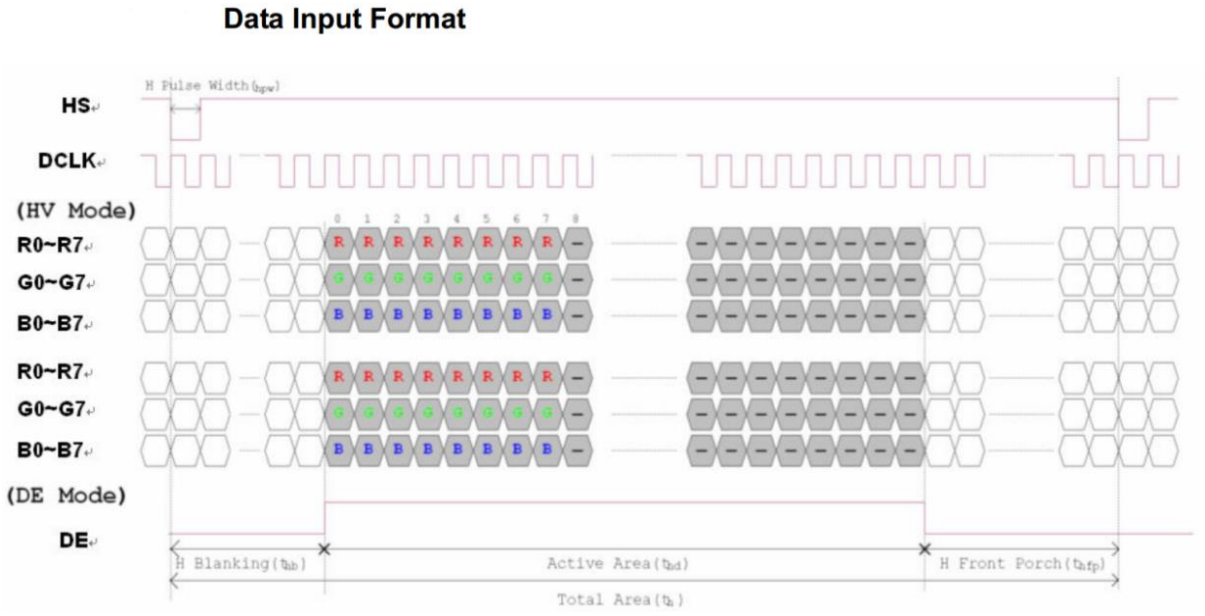
7inch Resistive Touch LCD 模块是不带 LCD 控制器的 7 inch TFT-LCD，TFT-LCD 液晶屏型号是 AT070TN92。AT070TN92 的分别率是 800*480，像素显示深度是 24 bit；板载的 XPT2046 是一款 4 线制电阻式触摸屏控制器。

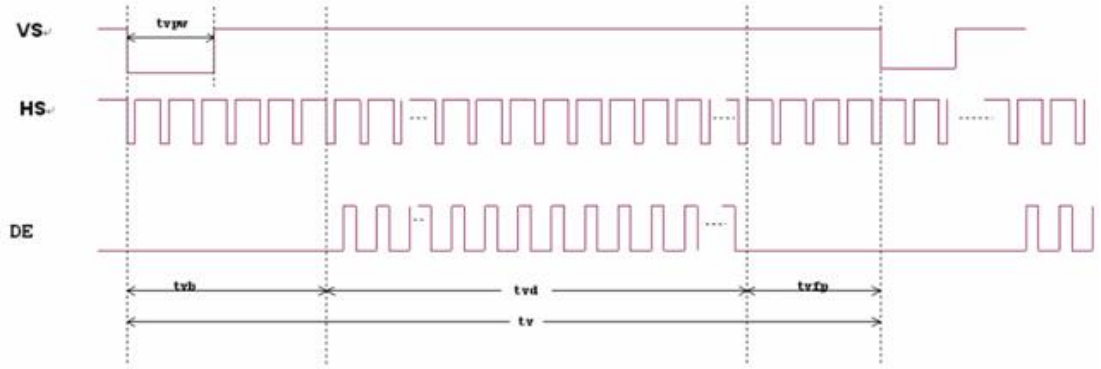
1.1 AT070TN92

AT070TN92 的基本参数如下：

类型	TFT
接口	24-bit 并行
背光	LED
可视区域（mm）	154.08（w） 85.92（H）
点距（mm）	0.0642（W） 0.1790（H）
尺寸比例	8：5
分辨率	800*480（Pixel）
扫描频率	60Hz
显示颜色	16.7M

这个 LCD 不带 LCD 控制器，需要自带控制器的 MUC 来控制；基本时序如下：





符号	注解
HS	水平同步信号，表示扫描 1 行的开始。
VS	垂直同步信号，表示扫描 1 帧的开始，一帧也就是 LCD 显示的一个画面。
DCLK	LCD 像素时钟
R0-R7	红色调色板数据线
G0-G7	绿色调色板数据线
B0-B7	蓝色调色板数据线
DE	数据使能信号。

表 1. 时序图符号含义

Item	Symbol	Values			Unit	Remark
		Min.	Typ.	Max.		
Horizontal Display Area	thd	-	800	-	DCLK	
DCLK Frequency	fclk	26.4	33.3	46.8	MHz	
One Horizontal Line	th	862	1056	1200	DCLK	
HS pulse width	thpw	1	-	40	DCLK	
HS Blanking	thb	46	46	46	DCLK	
HS Front Porch	thfp	16	210	354	DCLK	

Item	Symbol	Values			Unit	Remark
		Min.	Typ.	Max.		
Vertical Display Area	tvd	-	480	-	TH	
VS period time	tv	510	525	650	TH	
VS pulse width	tvpw	1	-	20	TH	
VS Blanking	tvb	23	23	23	TH	
VS Front Porch	tvfp	7	22	147	TH	

说明：

- 上面表格参见 AT070TN92.pdf。
- CLK 单位是一个像素的时间，CLK=1/fclk，H 单位是一行的时间，H=th
- 从上表看出：LCD 的时钟是 26.4-46.8MHz，由外部提供时钟；注意的是水平后沿+水平脉冲=46；垂直后沿+垂直脉冲=23.

从上图可以看出：

扫描一行所需要的时间为：th = thb + thd + thfp；在 thd 范围内，每来一个时钟，通过并行数据接口传输一个像素点的数据，此 LCD 是每行 800 个像素点，所以 thd=800；

扫描一帧所需要的时间为：tv = tvb + tvd + tvfp；Hsync 相当于垂直信号的时钟，Hsync 的一个时钟周期就是 LCD 显示一行的时间，Hsync 每来一个下降沿，显示就增加一行；只有在 tvd 期间，才有实际显示数据传输，每增加一行的数据在会在 LCD 上显示；此 LCD 一共 480 行，所以 tvd = 480。

其他参数可以稍微变动，但必须得按照上面表格范围进行配置。

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 5V 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。
- XPT2046 片内集成有一个温度传感器。
- XPT2046 详细使用，请参考数据手册。
- 四线电阻屏的原理和校准算法请参考附件内容。

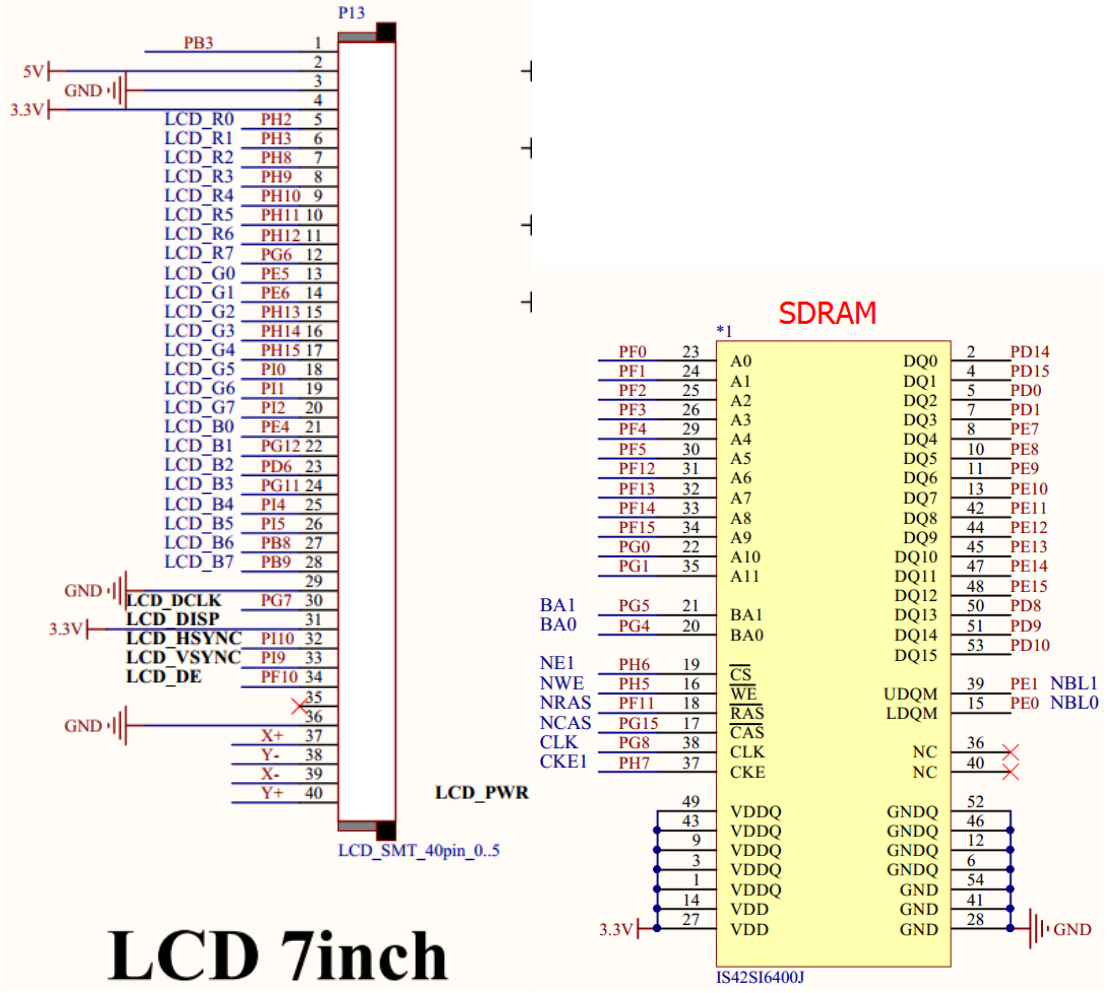
注：7inch Resistive Touch LCD 没有板载 XPT2046，需要添加外部电路。

2. 硬件说明

引脚号	标识	描述	类型	功能
1	VLED-	背光负极	电源型	背光负极，一般接 GND 或者 PWM 控制
2	VLED+	背光正极	电源型	背光正极，一般接 5V
3	GND	电源地	电源型	接地
4	VCC	电源正	电源型	电源，接 3.3V
5	R0	数据线	输入	红色调色板数据线
6	R1			
7	R2			
8	R3			
9	R4			
10	R5			

11	R6			
12	R7			
13	G0	数据线	输入	绿色调色板数据线
14	G1			
15	G2			
16	G3			
17	G4			
18	G5			
19	G6			
20	G7			
21	B0	数据线	输入	蓝色调色板数据线
22	B1			
23	B2			
24	B3			
25	B4			
26	B5			
27	B6			
28	B7			
29	GND	电源地	电源型	GND
30	DCLK	LCD 时钟	输入	LCD 时钟信号源
31	DISP	背光控制使能	输入	控制使能背光控制，一般接 VCC
32	HSYNC	行同步	输入	水平同步信号输入
33	VSYNC	帧同步	输入	垂直同步信号输入
34	DE	控制模式选择	输入	DE=0:SYNC 模式 DE=1:DE 模式
35	NC			
36	GND	电源地	电源型	GND
37	X+	触摸面板 X+	输出	电阻式触摸面板 X+模拟量输出
38	Y-	触摸面板 Y-	输出	电阻式触摸面板 Y-模拟量输出
39	X-	触摸面板 X-	输出	电阻式触摸面板 X-模拟量输出
40	Y+	触摸面板 Y+	输出	电阻式触摸面板 Y+模拟量输出

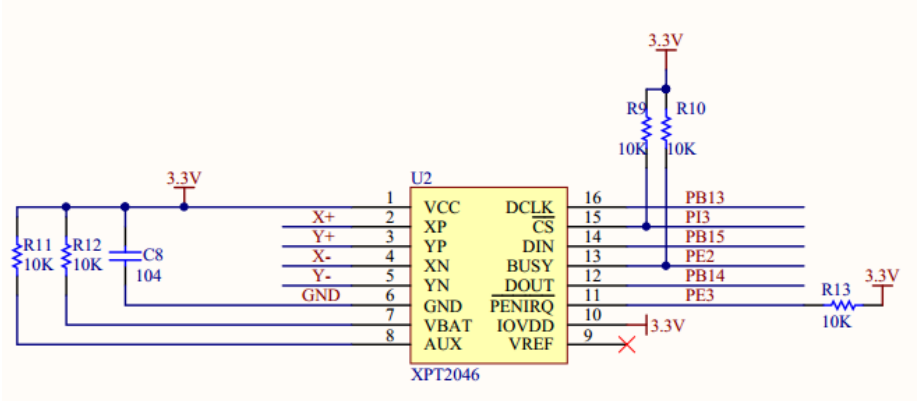
基于 Open429I-C 的硬件接口如下：



以上硬件接法是根据 STM32F429IGT6 的 TFT-LCD 控制器来接的，由 STM32F429IGT6 控制 LCD 的显示；触摸屏是 4 线电阻触摸；LCD 自带触摸屏芯片 XPT2046，MCU 通过 SPI 来读写触摸屏芯片的数据。

SDRAM 是用作 LCD 的数据缓冲区，TFT-LCD 控制器不断地从 SDRAM 读取数据，然后显示到 LCD 上，TFT-LCD 控制器会不断的刷新数据，当 SDRAM 的数据改变时，LCD 显示的内容马上也会改变，所以我们配置好 TFT-LCD 控制器的寄存器后，就只需要改变 SDRAM 的数据就可以控制 LCD 的显示，刷新的频率是有 LCD 的像素时钟决定的。

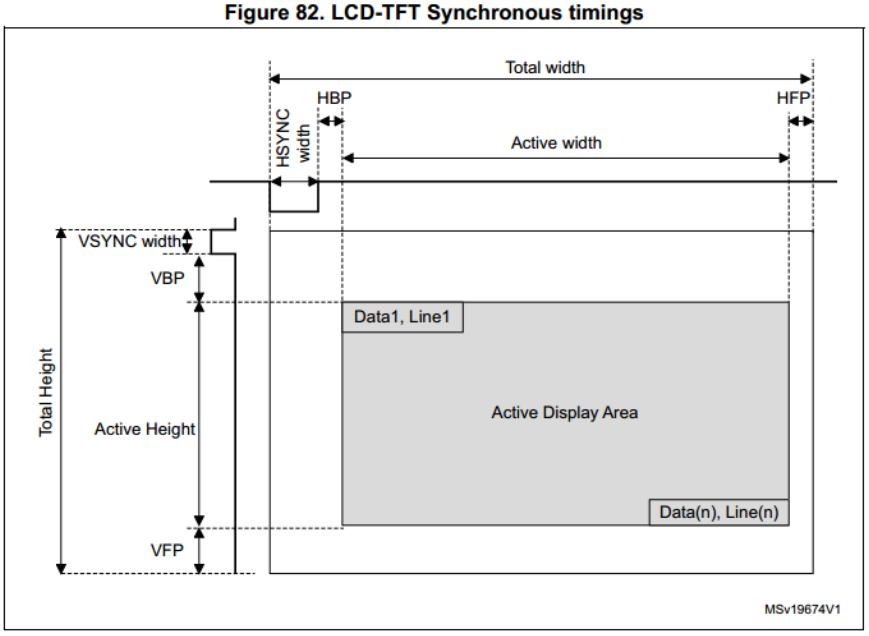
电阻触摸屏接口如下：



3. 软件说明

以下程序是基于 Open407I-C 的板子的硬件写的，Open407I-C 的板子的核心芯片是 STM32F429IGT6，STM32F429IGT6 自带 800*600 分辨率的 TFT-LCD 控制器。
STM32F429IGT6 自带的控制器如下：

Figure 82. LCD-TFT Synchronous timings



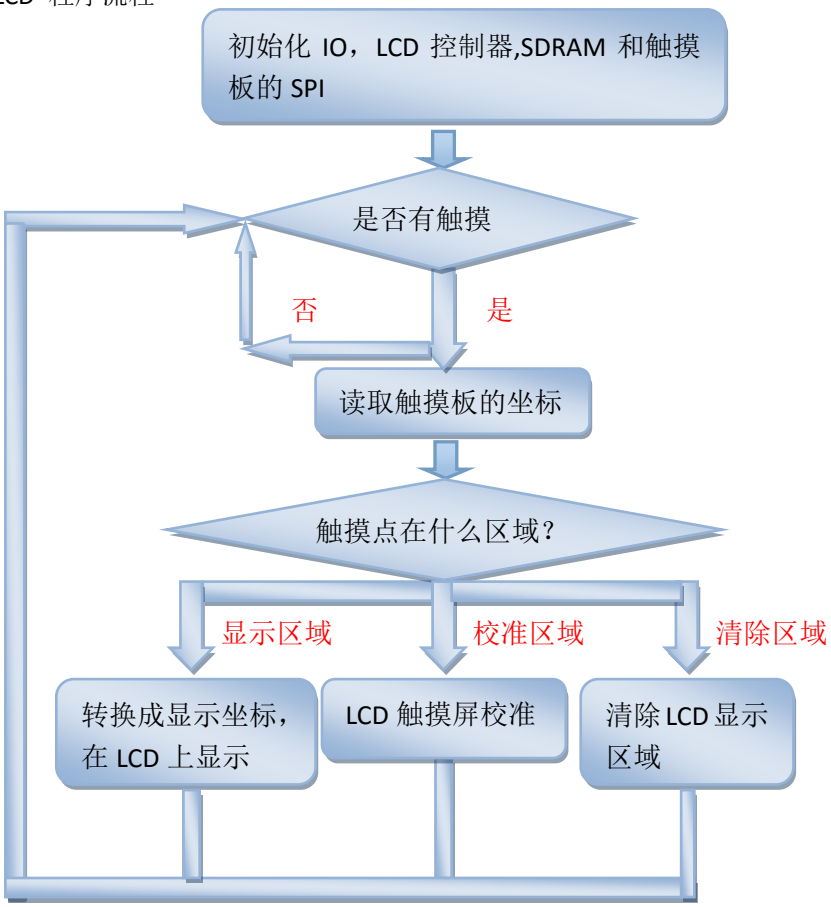
HBP 和 HFP 分别是水平后沿和水平前沿；
VBP 和 VFP 分别是垂直后沿和垂直前沿。
- HSYNC Width 和 VSYNC Height：水平和垂直同步宽度可以通过 LTDC_SSCR 寄存器下的 HSW（LTDC_SSCR[27:16]）和 VSH（LTDC_SSCR[10:0]）的值来设置;赋值方式为：HSW= HSYNC Width - 1，VSH = VSYNC Height - 1。

- HBP 和 VBP 可以通过 LTDC_BPCR 寄存器的 AHBP（LTDC_BPCR[27:16]）和 AVBP（LTDC_BPCR[10:0]）的值来设置。赋值方式为：AHBP= HSYNC Width + HBP - 1，AVBP= VSYNC Height + VBP - 1；

- Active Width 和 Active Height 可以通过 LTDC_AWCR 寄存器的 AAW（LTDC_AWCR[27:16]）和 AAH（LTDC_AWCR[10:0]）的值来设置。赋值方式为： AAW= HSYNC Width + HBP+ Active Width - 1， AAH= VSYNC Height + VBP + Active Height - 1；

- Total Width 和 Total Height 可以通过 LTDC_TWCR 寄存器的 TOTALW（LTDC_TWCR[27:16]）和 TOTALH（LTDC_TWCR[10:0]）的值来设置。赋值方式为：
TOTALW= HSYNC Width + HBP+ Active Width +HFP - 1， TOTALH= VSYNC Height + VBP + Active Height +VFP - 1；

LCD 程序流程



源代码解析：

```
/*LCD 显示的像素大小（长宽）*/
#define    LCD_PIXEL_WIDTH          ((uint16_t)800)
#define    LCD_PIXEL_HEIGHT         ((uint16_t)480)
/*LCD 的缓冲区 SDRAM 的地址*/
#define LCD_FRAME_BUFFER          ((uint32_t)0xD0000000)
#define BUFFER_OFFSET                ((uint32_t)0x200000)
/**********************************************
LCD 控制器初始化
**********************************************/
void LCD_Init(void)
{
  LTDC_InitTypeDef          LTDC_InitStruct;

  /*初始化 LCD 的使能管脚 ----------------------------------------*/
  LCD_CtrlLinesConfig();
  LCD_ChipSelect(DISABLE);
  LCD_ChipSelect(ENABLE);
```

```
  /*使能 LTDC 时钟  */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_LTDC, ENABLE);

  /*使能 DMA2D 时钟  */
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_DMA2D, ENABLE);

  /*  初始化  LCD 控制管脚*/
  LCD_AF_GPIOConfig();

  /*  初始化 FMC 的 SDRAM 接口*/
  SDRAM_Init();

  /* LTDC  配置*********************************************************/
  /*水平同步信号在低电平有效*/
  LTDC_InitStruct.LTDC_HSPolarity = LTDC_HSPolarity_AL;
  /*垂直同步信号在低电平有效  */
  LTDC_InitStruct.LTDC_VSPolarity = LTDC_VSPolarity_AL;
  /*数据信号在低电平有效  */
  LTDC_InitStruct.LTDC_DEPolarity = LTDC_DEPolarity_AL;
  /*  输入像素时钟  */
  LTDC_InitStruct.LTDC_PCPolarity = LTDC_PCPolarity_IPC;

  /*  初始化背景颜色的 R，G，B 的值*/
  LTDC_InitStruct.LTDC_BackgroundRedValue = 0;
  LTDC_InitStruct.LTDC_BackgroundGreenValue = 0;
  LTDC_InitStruct.LTDC_BackgroundBlueValue = 0;

  /*  使能像素点的时钟  */
  /* PLLSAI_VCO Input = HSE_VALUE/PLL_M = 1 Mhz */
  /* PLLSAI_VCO Output = PLLSAI_VCO Input * PLLSAI_N = 192 Mhz */
  /* PLLLCDCLK = PLLSAI_VCO Output/PLLSAI_R = 192/4 = 48 Mhz */
  /* LTDC clock frequency = PLLLCDCLK / RCC_PLLSAIDivR = 48/2 = 24 Mhz */
  RCC_PLLSAIConfig(192, 7, 4);
  RCC_LTDCCLKDivConfig(RCC_PLLSAIDivR_Div2);

  /*使能 PLLSAI  时钟*/
  RCC_PLLSAICmd(ENABLE);
  /*  等待 PLLSAI 时钟激活  */
  while(RCC_GetFlagStatus(RCC_FLAG_PLLSAIRDY) == RESET)
  {
  }
/* HSYNC Width =10, HBP    = 36,    Active Width = LCD_PIXEL_WIDTH, HFP =210 */
/* VSYNC Height = 3, VBP    =20,    Active Height    = LCD_PIXEL_HEIGHT , VFP    = 22*/
  /*LCD 控制器的时间设置*/
```

```
  /*配置水平脉冲同步宽度 HSYNC Width =10，寄存器配置的值为 HSYNC Width - 1 */
  LTDC_InitStruct.LTDC_HorizontalSync = 9;
  /* 配置垂直脉冲同步高度 VSYNC Height = 3 ，寄存器配置的值为 VSYNC Height - 1*/
  LTDC_InitStruct.LTDC_VerticalSync = 2;
  /*配置水平后沿 HBP  = 36，寄存器配置的值为 VSYNC Height + VBP - 1*/
  LTDC_InitStruct.LTDC_AccumulatedHBP = 45;
  /*配置垂直后沿 VBP  =20， 寄存器配置的值为 HSYNC Width + HBP - 1 */
  LTDC_InitStruct.LTDC_AccumulatedVBP = 22;
  /*水平显示 Active Width = LCD_PIXEL_WIDTH，
寄存器配置的值为 VSYNC Height + VBP+Active Height - 1 */
  LTDC_InitStruct.LTDC_AccumulatedActiveW =  45 + LCD_PIXEL_WIDTH;
  /*垂直显示 Active Height  = LCD_PIXEL_HEIGHT，
寄存器配置的值为 HSYNC Width + HBP+ Active Width - 1*/
  LTDC_InitStruct.LTDC_AccumulatedActiveH =  22 + LCD_PIXEL_HEIGHT;
  /* 配置总共水平宽度 Totalwidth= VSYNC Height + VBP+Active Height +VFP– 1*/
  LTDC_InitStruct.LTDC_TotalWidth = 45 + LCD_PIXEL_WIDTH + 210;
/* 配置总垂直高度 TotalHeight= HSYNC Width + HBP+ Active Width +HFP- 1*/
  LTDC_InitStruct.LTDC_TotalHeigh = 22 + LCD_PIXEL_HEIGHT + 22;

  LTDC_Init(&LTDC_InitStruct);

}
/************************************************************
LC 层的初始化设置
************************************************************/
void LCD_LayerInit(void)
{
  LTDC_Layer_InitTypeDef LTDC_Layer_InitStruct;

  /* 显示窗口配置 */
  /*
  Horizontal start = horizontal synchronization + Horizontal back porch = 46
  Horizontal stop = Horizontal start + window width -1 = 46 + 800 -1
  Vertical start    = vertical synchronization + vertical back porch       = 23
  Vertical stop     = Vertical start + window height -1   = 23 + 480 -1        */
  LTDC_Layer_InitStruct.LTDC_HorizontalStart =46;
  LTDC_Layer_InitStruct.LTDC_HorizontalStop = (LCD_PIXEL_WIDTH + 46 - 1);
  LTDC_Layer_InitStruct.LTDC_VerticalStart = 23;
  LTDC_Layer_InitStruct.LTDC_VerticalStop = (LCD_PIXEL_HEIGHT + 23 - 1);

  /* 像素点的数据模式是 RGB565*/
  LTDC_Layer_InitStruct.LTDC_PixelFormat = LTDC_Pixelformat_RGB565;
  /* 配置透明度 (255 属于完全不透明) */
  LTDC_Layer_InitStruct.LTDC_ConstantAlpha = 255;
```

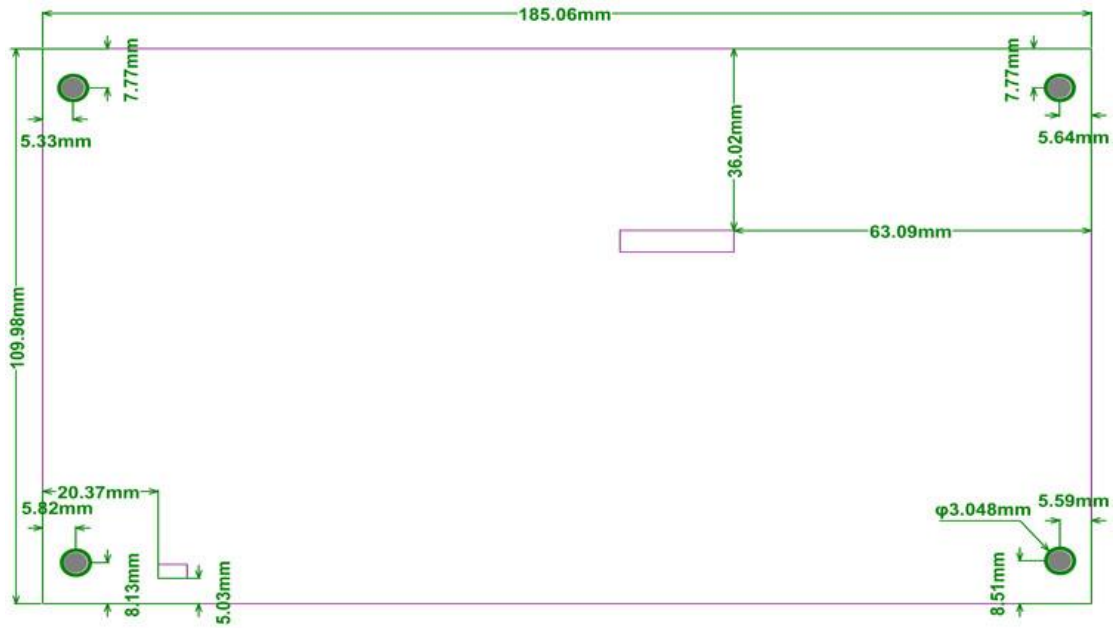
```
/*  初始化颜色 RGB 的值*/
LTDC_Layer_InitStruct.LTDC_DefaultColorBlue = 0;
LTDC_Layer_InitStruct.LTDC_DefaultColorGreen = 0;
LTDC_Layer_InitStruct.LTDC_DefaultColorRed = 0;
LTDC_Layer_InitStruct.LTDC_DefaultColorAlpha = 0;
/*  配置混合因素  */
LTDC_Layer_InitStruct.LTDC_BlendingFactor_1 = LTDC_BlendingFactor1_CA;
LTDC_Layer_InitStruct.LTDC_BlendingFactor_2 = LTDC_BlendingFactor2_CA;
/*
Line Lenth = Active high width x number of bytes per pixel + 3
Active high width              = LCD_PIXEL_WIDTH
number of bytes per pixel = 2       (pixel_format : RGB565)
*/
LTDC_Layer_InitStruct.LTDC_CFBLineLength = ((LCD_PIXEL_WIDTH * 2) + 3);
/* pitch 是一行的开始到下一行的开始的字节数。
Pitch = Active high width x number of bytes per pixel */
LTDC_Layer_InitStruct.LTDC_CFBPitch = (LCD_PIXEL_WIDTH * 2);

/*  配置总共多少行  */
LTDC_Layer_InitStruct.LTDC_CFBLineNumber = LCD_PIXEL_HEIGHT;

/*  起始地址的配置  : LCD 缓冲区 SDRAM 地址  */
LTDC_Layer_InitStruct.LTDC_CFBStartAdress = LCD_FRAME_BUFFER;

/*初始化 LTDC layer 1 */
LTDC_LayerInit(LTDC_Layer1, &LTDC_Layer_InitStruct);

/*  配置  Layer2 */
/*起始地址的配置:  在  layer 1 的起始地址上偏移 BUFFZER_OFFSET*/
LTDC_Layer_InitStruct.LTDC_CFBStartAdress = LCD_FRAME_BUFFER + BUFFER_OFFSET;

/*  配置混合因素  */
LTDC_Layer_InitStruct.LTDC_BlendingFactor_1 = LTDC_BlendingFactor1_PAxCA;
LTDC_Layer_InitStruct.LTDC_BlendingFactor_2 = LTDC_BlendingFactor2_PAxCA;

/*初始化  LTDC layer 2 */
LTDC_LayerInit(LTDC_Layer2, &LTDC_Layer_InitStruct);

/*  使能 LTDC_Layer1 和 LTDC_Layer1 */
LTDC_LayerCmd(LTDC_Layer1, ENABLE);
LTDC_LayerCmd(LTDC_Layer2, ENABLE);

/* LTDC  更新配置*/
LTDC_ReloadConfig(LTDC_IMReload);
```

```
  /*设置默认字体 */
  LCD_SetFont(&LCD_DEFAULT_FONT);

   LTDC_DitherCmd(ENABLE);
}

/*******************************************
LCD 清屏，也就是对 SDRAM 的缓冲区清除数据
*******************************************/
void LCD_Clear(uint16_t Color)
{
  uint32_t index = 0;
  /*擦除存储 */
  for (index = 0x00; index < BUFFER_OFFSET; index++)
  {
     *(__IO uint16_t*)(CurrentFrameBuffer + (2*index)) = Color;
  }
}
int main(void)
{
初始化系统设置
  /* LCD 初始化设置 */
  LCD_Init();

  /* LCD 层初始化设置 */
  LCD_LayerInit();

  /* 使能 LTDC */
  LTDC_Cmd(ENABLE);
LCD_SetLayer(LCD_FOREGROUND_LAYER);
  /*清屏 LCD*/
LCD_Clear(LCD_COLOR_RED);
While(1)
{
      getDisplayPoint(&display, Read_Ads7846(), &matrix );      //得到触摸点坐标 X，Y 的值
      if(((display.y < Touch_Y-10) && (display.y >= 2)))        //判断是否超出屏的界限
     {
        if((display.x >= 798) || (display.x < 2))
        {}
        else
        {
                    if(display.y<20)display.y=20;
                    if(display.y>Touch_Y-22)display.y=Touch_Y-22;
```

```
                    if(display.x<20)display.x=20;
                    if(display.x>780)display.x=780;
                    LCD_GetColors(&Current_Color,0);
                    /*清除之前点的显示*/
                    LCD_SetColors(LCD_COLOR_BLACK,LCD_COLOR_BLACK);
                    LCD_DrawFullCircle(display_p.x,display_p.y,20);
                    LCD_DrawLine(0, display_p.y, 800, LCD_DIR_HORIZONTAL);
                    LCD_DrawLine(display_p.x, 0, 52*8, LCD_DIR_VERTICAL);
                    /*在当前的触摸点显示*/
                    LCD_SetColors(LCD_COLOR_BLUE,LCD_COLOR_BLACK);
                    LCD_DrawFullCircle(display.x,display.y,20);
                    LCD_DrawLine(0, display.y, 800, LCD_DIR_HORIZONTAL);
                    LCD_DrawLine(display.x, 0, 52*8, LCD_DIR_VERTICAL);

                    Xpos_Ypos(display.x,display.y);
                    display_p.x=display.x;
                    display_p.y=display.y;
        }
    }
    else if ((display.y <= Touch_Y+50) && (display.y >= Touch_Y) && (display.x >= 300) &&
(display.x <= 390) )   //判断触摸点是否在 Clear 区域
    {/*清除整个显示区域*/
        LCD_SetFont(&Font8x8);
        for(linenum = 0; linenum < 52; linenum++)
        {
            LCD_ClearLine(LINE(linenum));
        }
    }
            else if ((display.y <= Touch_Y+50) && (display.y >= Touch_Y) && (display.x >= 390) &&
(display.x <= 500))    //判断触摸点是否在 Adjust 区域
            {
                getDisplayPoint(&display, Read_Ads7846(), &matrix );
                if ((display.y <= Touch_Y+50) && (display.y >= Touch_Y) && (display.x >= 390) &&
(display.x <= 500))
                {/*进行触摸屏校准*/
                TouchPanel_Calibrate();
                TP_Config();
                }
            }
    else
    {
    }
  }
}
```

4. 程序验证效果

5. 模块尺寸大小



附件：四线触摸屏的原理与校准

1) 电阻式触摸屏的工作原理

典型的电阻式触摸屏一般由三部分构成：两层透明的阻性导电层，在两层导电层之间的隔离层以及电极。电阻式触摸屏示意图如图 1 所示。电阻式触摸屏就相当于一种传感器，利用压力感应进行控制，将矩形区域中触摸点（x, y）的物理位置转换为代表 x 坐标和 y 坐标的电压。触摸屏工作时，上下导电层相当于电阻网络，当某一层电极加上电压时，会在该网络上形成电压梯度。如果有外力使得上下两层在某一点接触，则在电极未加电压的另一层可以检测到接触点处的电压，经过 A/D 转换知道接触点处的坐标。比如，在 Y+ 电极上加驱动电压 VCC，Y- 电极接地，则顶层导电层（Y+, Y-）上形成电压梯度，X+ 作为引出端测量接触点的电压，当有外力使得上下两层导电层有在某一点（x1,y1）接触，则在 X+ 处可测得电压为 VX+，由于导电层均匀导电，则可以认为触点电压与驱动电压之比即为触点 Y 坐标与触摸屏高度之比，即 y1=(VX+ / VCC)*height。同理，将驱动电压施加在 X+ 电极，并在 Y+ 处测量触点电压，从而可以获得该点的 x 坐标。

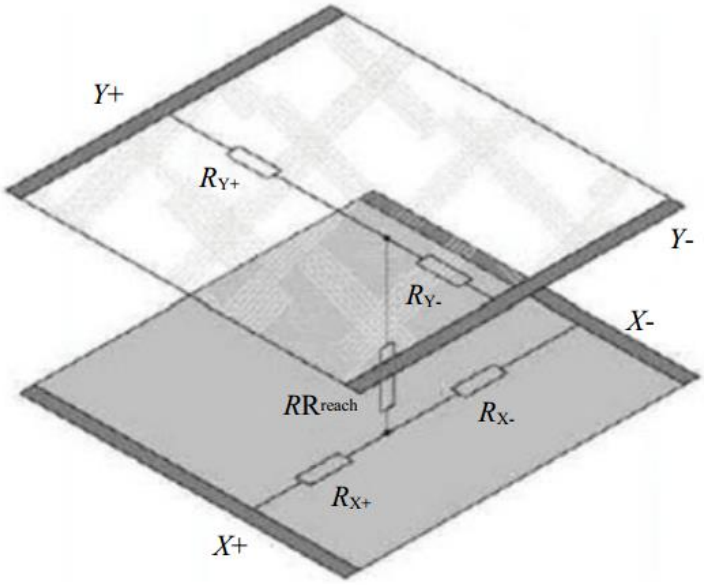


图1　电阻式触摸屏示意图

由压力感应得到坐标值的并不能达到 100%的精度，它存在着误差，尤其是触摸屏本身电阻材料的均匀性以及出厂安装时存在的机械误差，直接影响到了触摸屏的精度。因此，在使用触摸屏时，需要将触摸屏上的图形经过一定的变换，换算出与 LCD 显示屏相一致的点集合，这种图形重建的过程就是校准。

2) 触摸屏三点校准

触摸屏和 LCD 显示屏叠加在配套使用时，由于存在误差，触摸屏坐标系和显示屏坐标系不重合，校准的目的就是在这两种坐标系之间找到一种合适正确的映射关系，使触摸屏上显示

的图形经过变换，与 LCD 显示的图形保持一致。这里触摸屏和 LCD 显示图形的点都用矢量来表示：$Q(x, y)$为触摸屏上的点，称为物理坐标；Qd（xd，yd）为 LCD 显示屏上的点，称为显示坐标。设物理坐标：

$x= R\cos\theta$

$y= R\sin\theta$

由于触摸屏和 LCD 显示屏接触点之间存在角度误差，同时考虑到每个点的 x 和 y 坐标都存在不同的因子缩放，并且触摸屏和 LCD 显示屏之间还存在移动误差，则假设角度差为 φ，缩放因子为 k_x 和 k_y，位移因子为 S_x 和 S_y，可得到显示坐标：

$x_d= k_xR\cos(\theta+\varphi)+ S_x$

$y_d= k_yR\sin(\theta+\varphi)+ S_y$

一般情况，触摸屏和 LCD 显示屏之间的角度误差 ϕ 极小，则 sinϕ≈ϕ， cosϕ≈1。那么， LCD 显示屏上点坐标可以化简为：

$x_d = k_xR[\cos\theta\cos\varphi- \sin\theta\sin\varphi]+ S_x$

$= k_xR\cos\theta-\varphi k_xR\sin\theta+ S_x= k_x \cdot x-\varphi kx \cdot y+ Sx$

$y_d = k_yR[\sin\theta\cos\varphi+ \cos\theta\sin\varphi]+ S_y$

$= k_yR\sin\theta+\varphi k_yR\cos\theta+ S_y=k_y \cdot y+\varphi k_y \cdot x+ S_y$

由上式中可以看出，除了 x 和 y，方程式右边各项均为常量，即触摸屏和显示屏的坐标系可以认为是线性的，基于此方程实现的校准也称为线性校准。现在用一般情况来代替各乘积项的系数，

则可以得到：

x 坐标方程： xd = A1· x+ B1· y+ C1

y 坐标方程： yd = A2· x+ B2· y+ C2

显然，如果能求出线性变换的参数 (A1,B1,C1,A2,B2,C2)，就可以通过上述等式来校准从触摸屏那里得来的显示坐标了 [5]。为了求出这六个参数，在触摸屏上任意取三个点（由于边界点的线性度差，所以要尽量避免），设物理坐标和显示坐标分别为(x1, y1)、 (x2, y2)、 (x3, y3) 和 (xd1, yd1)、 (xd2, yd2)、 (xd3, yd3)，可以得到方程组：

A1· x1 + B1· y1 + C1 = xd1；

A1· x2 + B1· y2 + C1 = xd2；

A1· x3 + B1· y3 + C1 = xd3；

A2· x1 + B2· y1 + C2 = yd1；

A2· x2 + B2· y2 + C2 = yd2；

A2· x3 + B2· y3 + C2 = yd3；

解方程组可得：

δ= (x1- x2)(y2- y3)- (x2- x3)(y1- y2)

A1= [(xd1- xd2)(y2- y3)- (xd2- xd3)(y1- y2)]/δ

B1=- [(xd1- xd2)(y2- y3)- (xd2- xd3)x1- x2)]/δ

C1= xd1- A1x1- B1y1

A2= [(yd1- yd2)(y2- y3)- (yd2- yd3)(y1- y2)]/δ

B2=- [(yd1- yd2)(x2- x3)- (yd2- yd3)(x1- x2)]/δ

C2= yd1- A2x1- B2y1

值得注意的是，只有在触摸屏和 LCD 显示屏之间的角度误差 ϕ 极小的情况下，上述的基本线性校准算法才适用。为了达到更好的校准效果，本文在此基础上，对基本线性校准算法进行了优化，形成五点校准。

3) 触摸屏五点校准

为了使校准更加精确，现在触摸屏上任意取五个点，设物理坐标和显示坐标分别为 (x1, y1)、(x2, y2)、(x3, y3)、(x4, y4)、(x5, y5) 和 (xd1, yd1)、(xd2, yd2)、(xd3, yd3)、(xd4, yd4)、(xd5, yd5)，代入 x 坐标方程 xd = A1 · x+ B1 · y+ C1，可以得到方程组：

A1 · x1 + B1 · y1 + C1 = xd1 ；

A1 · x2 + B1 · y2 + C1 = xd2 ；

A1 · x3 + B1 · y3 + C1 = xd3 ；

A1 · x4 + B1 · y4 + C1 = xd4 ；

A1 · x5 + B1 · y5 + C1 = xd5 ；

对等式做如下处理：

第一步，将原方程 5 个等式直接相加，得第 1 个总等式：

A1 · (x1+x2+x3+x4+x5)+ B1 · (y1+ y2+ y3+ y4+ y4)+5C1=xd1+xd2+xd3+xd4+xd5 ；

第二步，将原方程 5 个等式分别乘以参数 x，然后 5 个等式相加，得第 2 个总等式：

A1 · (x 1 · x 1 +x 2 · x 2+x 3 · x 3+x 4 · x 4+x 5 · x 5)+B1 · (y1 · x1+y2 · x2+ y3 · x3+ y4 · x4+y5 · x5)+ C1(x1+x2+x3+x4+x5)=xd1 · x1+xd2 · x2+xd3 · x3+xd4 · x4+xd5 · x5 ；

第三步，将原方程 5 个等式分别乘以参数 y，然后 5 个等式相加，得第 3 个总等式：

A1 · (x 1 · y 1 +x 2 · y 2+x 3 · y 3+x 4 · y 4+ x 5 · y 5)+B1 · (y1 · y1+y2 · y2+ y3 · y3+ y4 · y4+y5 · y5)+C1(y1+y2+y3+y4+y5)=xd1 · y1+xd2 · y2+xd3 · y3+ xd4 · y4+ xd5 · y5 ；

由这三个总等式构成一个 x 坐标的三阶线性方程组，用克莱姆法则可以将参数 A1， B1， C1 求出。

同理，可以得到 y 坐标的三阶线性方程组：

A2 · (x1+x2+x3+x4+x5)+B2 · (y1+y2+y3+y4+y4)+5C2=yd1+yd2+yd3+yd4+yd5 ；

A2· (x1 · x1+x2· x2+x3· x3+x4· x4+x5· x5)+B2 · (y1 · x1+y2· x2+y3· x3+y4· x4+y5· x5)+C2(x1+x2+x3+x4+x5)=yd1· x1+yd2· x2+yd3 · x3+yd4 · x4+yd5 · x5 ；

A2 · (x 1 · y 1 + x 2 · y 2+ x 3 · y 3+ x 4 · y 4+ x 5 · y 5)+B2 · （ y1 · y1+ y2 · y2+ y3 · y3+ y4 · y4+ y4 · y5)+C2(y1+ y2+y3+y4+y5)=yd1 · y1+yd2 · y2+yd3 · y3+yd4 · y4+yd5 · y5；

用克莱姆法则可以将参数 A2， B2， C2 求出。

也可以使用三阶行列式求解:

4) 三阶行列式解方程组

引例 2 用消元法解关于 x,y,z 三元线性方程组

$$\begin{cases} ax+by+cz=d, \\ ex+fy+gz=h, \\ ix+jy+kz=l. \end{cases}$$

解得：

$$\begin{cases} x_1 = \dfrac{D_1}{D} = \dfrac{bgl-bhk-cfl+chj+dfk-dgj}{afk-agj-bek+bgi+cej-cfi} \\ x_2 = \dfrac{D_2}{D} = \dfrac{-agl+ahk+cel-chi-dek+dgi}{afk-agj-bek+bgi+cej-cfi} \\ x_3 = \dfrac{D_3}{D} = \dfrac{afl-ahj-bel+bhi+dej-dfi}{afk-agj-bek+bgi+cej-cfi} \end{cases}$$

为了记忆三元线性方程组的求解公式，可引入三阶行列式，三阶行列式的定义如下：

定义 设有 9 个数排成 3 行 3 列的数表

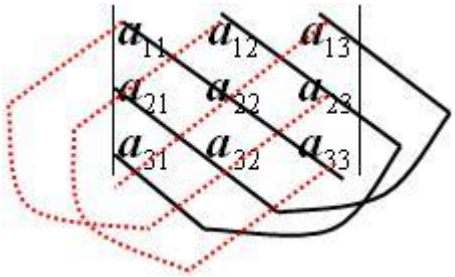
$$\begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix} \qquad (3)$$

记

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}a_{22}a_{33}+a_{12}a_{23}a_{31}+a_{13}a_{21}a_{32}- a_{13}a_{22}a_{31}-a_{12}a_{21}a_{33}-a_{11}a_{23}a_{32}, \qquad (4)$$

（4）式称为数表（3）所确定的三阶行列式.

三阶行列式的展开式也可用对角线法则得到，三阶行列式的对角线法则如下图所示：



其中每一条实线上的三个元素的乘积带正号，每一条虚线上的三个元素的乘积带负号，所得六项的代数和就是三阶行列式的展开式.