# 7inch Capacitive 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

深圳微雪电子有些公司

# 目录

1. 模块简介 ..... 1
1.1 AT070TN92 ..... 1
1.2 FT5X06 ..... 3
2. 硬件说明 ..... 6
3. 软件说明 ..... 8
4. 程序验证效果 ..... 17
5. 模块尺寸大小 ..... 17

# 1. 模块简介

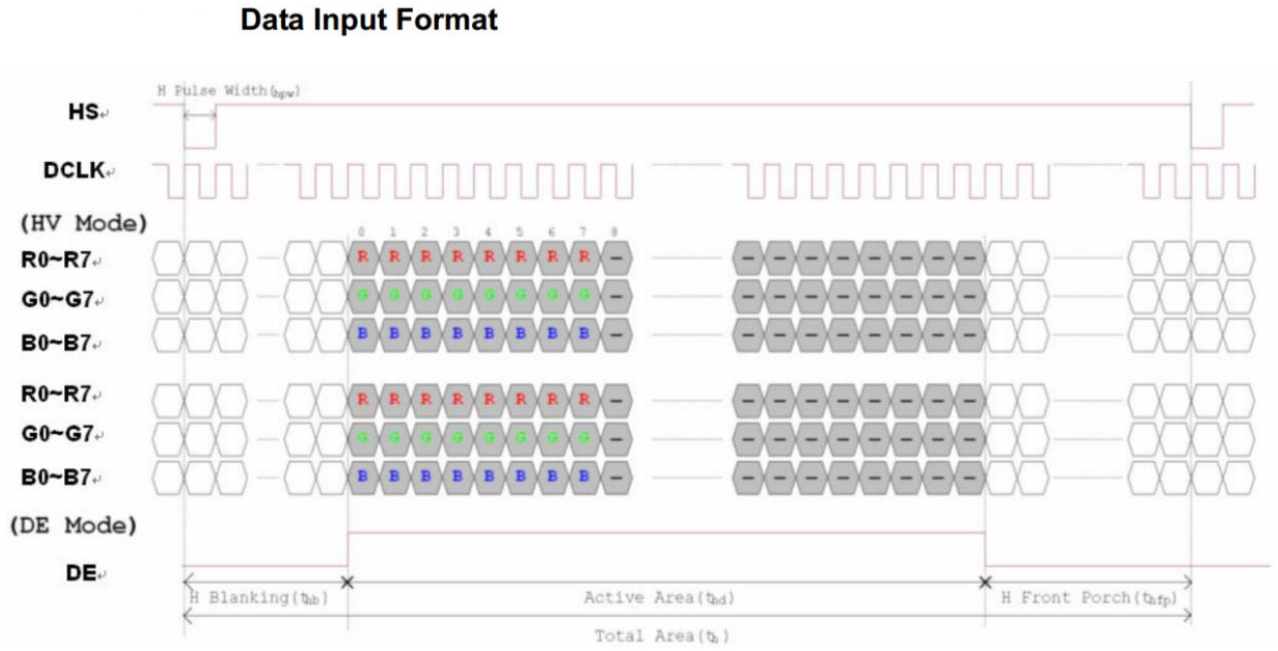
7inch Capacitive Touch LCD 模块是不带 LCD 控制器的 7 inch TFT-LCD，TFT-LCD 液晶屏型号是 AT070TN92。AT070TN92 的分别率是 800*480，像素显示深度是 24 bit；板载电容触摸控制芯片 FT5206GE1（FT5x06 系列的电容触摸控制芯片）；可以同时进行 5 点触控。

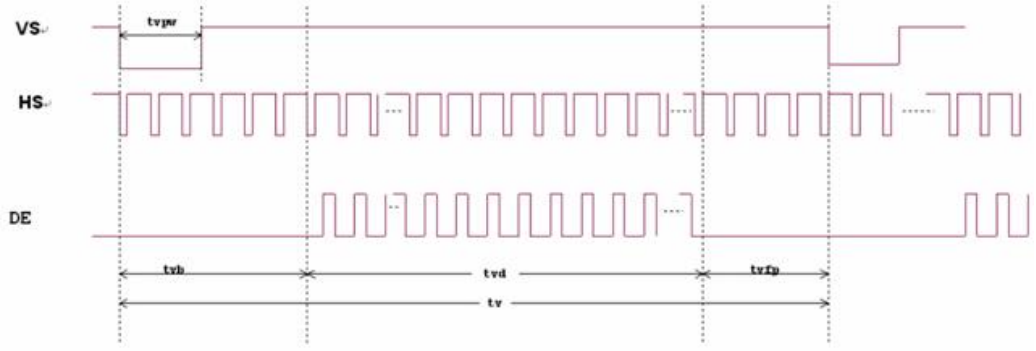
## 1.1 AT070TN92

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：






外部引脚说明：

| 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。 |

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。

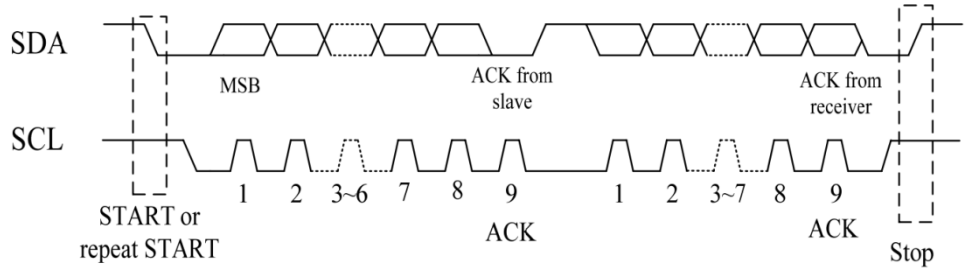
其他参数可以稍微变动，但必须得按照上面表格范围进行配置。

## 1.2 FT5X06

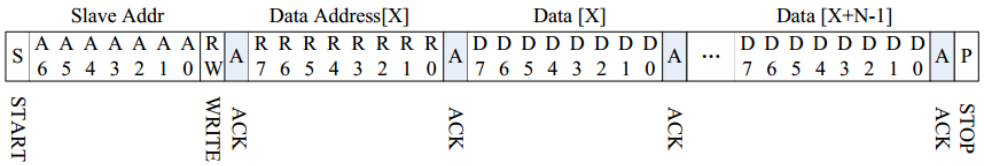
- 接口方式：I2C/SPI （此屏只引出 I2C 接口）
- 2.8V-3.6V 的工作电压
- 内置 8051 的 MCU，此 8051 具有 32KB 的程序存储，6KB 的数据内存和 256KB 的内部数据空间
- 工作温度范围：-40℃-85℃。
- 支持 5 点触摸

**FT5X06 的 I2C 的读写时序**

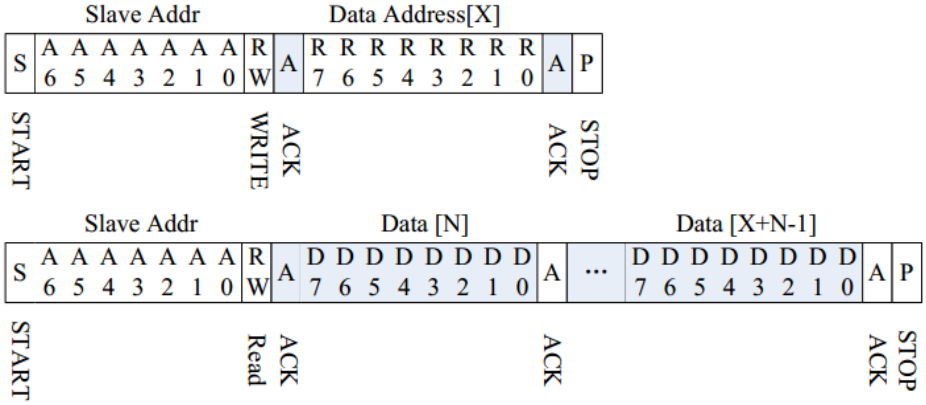
如下图. FT5X06 的 I2C 读写时序




**FT5X06 写 N 字节**




**FT5X06 读 N 字节**




I2C 的时间参数

| 参数 | 最小 | 最大 | 单位 |
|---|---|---|---|
| SCL 时钟 | 0 | 400 | KHZ |
| STOP 和 START | 4.7 | \ | us |
| 数据建立时间 | 250 | \ | ns |

**FT5X06 的寄存器**

FT5X06 系列只需要配置好工作模式，MCU 就可以直接从对应的寄存器里读取出 5 个触摸点的坐标 x、y 的值；以下对主要的几个寄存器进行介绍，其他寄存器可以参阅芯片的数据手册。

| 地址 | 名称 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | 类型 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | DEVIDE_MODE | | Device Mode[2:0] | | | | | | | RW |
| 0x01 | GEST_ID | Gesture ID[7:0] | | | | | | | | R |
| 0x02 | TD_STATUS | | | | | Number of touch points[3:0] | | | | R |
| 0x03 | TOUCH1_XH | 1st Event Flag | | | | 1st Touch X Position[11:8] | | | | R |
| 0x04 | TOUCH1_XL | 1 st Touch X Position[7:0] | | | | | | | | R |
| 0x05 | TOUCH1_YH | 1st Touch ID[3:0] | | | | 1st Touch Y Position[11:8] | | | | R |
| 0x06 | TOUCH1_YL | 1st Touch Y Position[7:0] | | | | | | | | R |
| 0x09 | TOUCH2_XH | 2nd Event Flag | | | | 2nd Touch X Position[11:8] | | | | R |
| 0x0A | TOUCH2_XL | 2 st Touch X Position[7:0] | | | | | | | | R |
| 0x0B | TOUCH2_YH | 2nd Touch ID[3:0] | | | | 2nd Touch Y Position[11:8] | | | | R |
| 0x0C | TOUCH2_YL | 2nd Touch Y Position[7:0] | | | | | | | | R |
| 0x0D | | | | | | | | | | R |
| 0x0E | | | | | | | | | | R |
| 0x0F | TOUCH3_XH | 3rd Event Flag | | | | 3rd Touch X Position[11:8] | | | | R |
| 0x10 | TOUCH3_XL | 3 st Touch X Position[7:0] | | | | | | | | R |
| 0x11 | TOUCH3_YH | 3rd Touch | | | | 3rd Touch | | | | R |

| | | | | | |
|---|---|---|---|---|---|
| | | ID[3:0] | | Y Position[11:8] | |
| 0x12 | TOUCH3_YL | 3st Touch Y Position[7:0] | | | R |
| 0x13 | | | | | R |
| 0x14 | | | | | R |
| 0x15 | TOUCH4_XH | 4th Event Flag | | 4th Touch X Position[11:8] | R |
| 0x16 | TOUCH4_XL | 4th Touch X Position[7:0] | | | R |
| 0x17 | TOUCH4_YH | 4th Touch ID[3:0] | | 4th Touch Y Position[11:8] | R |
| 0x18 | TOUCH4_YL | 4th Touch Y Position[7:0] | | | R |
| 0x19 | | | | | R |
| 0x1A | | | | | R |
| 0x1B | TOUCH5_XH | 5th Event Flag | | 5th Touch X Position[11:8] | R |
| 0x1C | TOUCH5_XL | 5 st Touch X Position[7:0] | | | R |
| 0x1D | TOUCH5_YH | 5th Touch ID[3:0] | | 5th Touch Y Position[11:8] | R |
| 0x1E | TOUCH5_YL | 5th Touch Y Position[7:0] | | | R |

DEVIDE_MODE：FT5x06 模式配置寄存器；DEVIDE_MODE[6:4]有三种模式：正常模式、系统信息模式(保留)、测试模式——读取原始数据(保留) ；一般设置为正常模式即可；正常模式下正常读取触摸点的坐标。

GEST_ID：这个寄存器记录怎么样的一个姿势触摸电容屏

| 寄存器 | 值 | 描述 |
|---|---|---|
| Gesture ID[7:0] | 0x10 | 向上移动 |
| | 0x14 | 向左移动 |
| | 0x18 | 向下移动 |
| | 0x1C | 向右移动 |
| | 0x48 | 放大 |
| | 0x49 | 缩小 |
| | 0x00 | 没姿势 |

TD_STATUS：检测同一时刻有多少个触摸点

| 寄存器 | 位地址 | 寄存器名字 | 描述 |
|---|---|---|---|
| TD_STATUS | 3:0 | Number of touch point[3:0] | How many points detected 1~5 is valid |
| | 7:4 | | |

TOUCHn_XH（n=1-5）：最高两位是检测第 n 个触摸点的触摸状态；低四位是第 n 个触摸点位置坐标 X 值的高 4 位

| 寄存器 | 位地址 | 寄存器名字 | 描述 |
|---|---|---|---|
| TOUCHn_XH | 7:6 | Event Flag | 00b：按下 |

| | | | |
|---|---|---|---|
| （n=1-5） | | | 01b：离开<br>10b：一直接触屏幕<br>11b：保留 |
| | 5:4 | | |
| | 3:0 | Touch X Position [11:8] | 第 n 个触摸点位置坐标 X 值的高 4 位 |

说明：

TOUCHn_XL（n=1-5）：第 n 个触摸点位置坐标 X 值的低 8 位

TOUCHn_YH（n=1-5）：高四位是第 n 个触摸点相应的 ID；低四位是第 n 个触摸点位置坐标 Y 值的高 4 位

TOUCHn_YL（n=1-5）：第 n 个触摸点位置坐标 Y 值的低 8 位

**FT5x06 读取数据流程**

初始化 I2C；

配置 FT5x06 工作模式；

如果有触摸屏中断，则读取触摸点位置坐标 XY 相对应的寄存器的值。

**注意：**相邻两个触摸点的坐标 XY 的寄存器不是连续的，比如第一个触摸点的坐标 XY 寄存器复制地址是 0x03，0x04，0x05，0x06，第二个触摸点的坐标 X，Y 寄存器地址是 0x09，0x0A，0x0B，0x0C，中间的地址 0x07 和 0x08 是没有值的。

## 2. 硬件说明

电容触摸屏接口定义

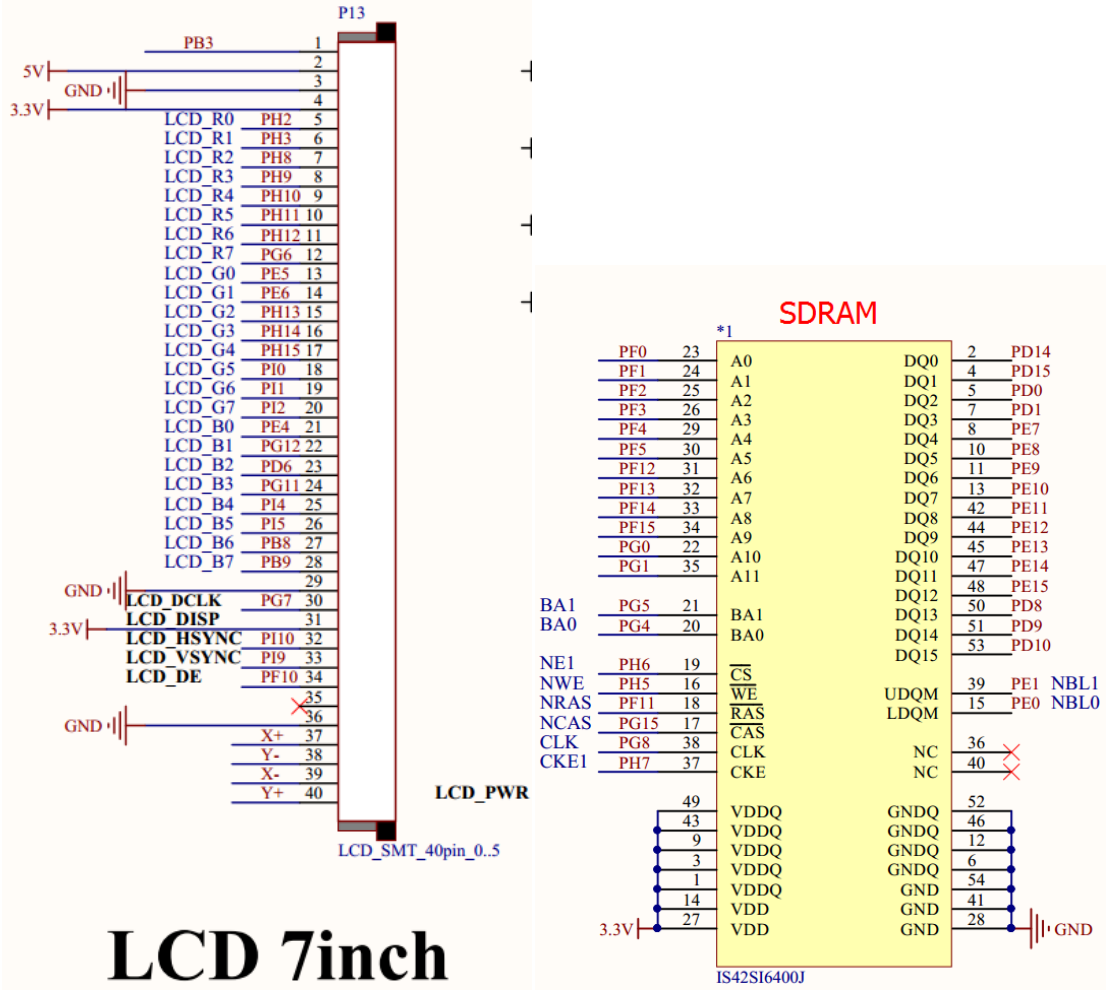
| 引脚号 | 标识 | 描述 | 类型 | 功能 |
|---|---|---|---|---|
| 1 | CAP_INT | 中断引脚 | 输入 | 外部触摸中断控制输入引脚 |
| 2 | CAP_WAKE | WAKEUP 引脚 | 输入 | 用于唤醒外部 TP 控制器 |
| 3 | I2C_SDA | I2C 数据脚 | 输出/输入 | I2C 数据传输脚，读写数据 |
| 4 | I2C_SCL | I2C 时钟脚 | 输入 | I2C 时钟控制脚，控制时钟 |

LCD 接口定义

| 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<br>DE=1:DE 模式 |
| 35 | NC | | | |
| 36 | GND | 电源地 | 电源型 | GND |
| 37 | X+ | - | - | 电容屏此处 NC |
| 38 | Y- | - | - | 电容屏此处 NC |
| 39 | X- | - | - | 电容屏此处 NC |
| 40 | Y+ | - | - | 电容屏此处 NC |

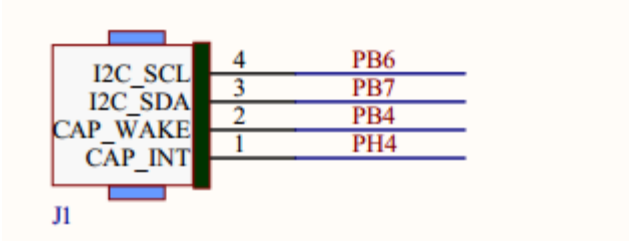
基于 Open429I-C 的硬件接口如下：




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；触摸屏是电容触摸；LCD 自带触摸屏芯片 FT5X06，MCU 通过 I2C 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。

电容触摸屏接口如下：

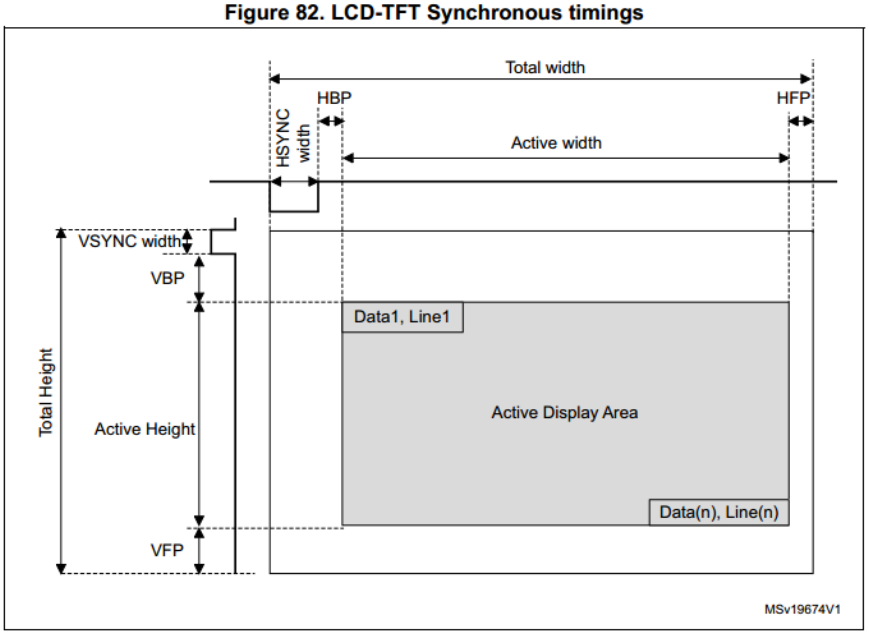



## 3. 软件说明

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。

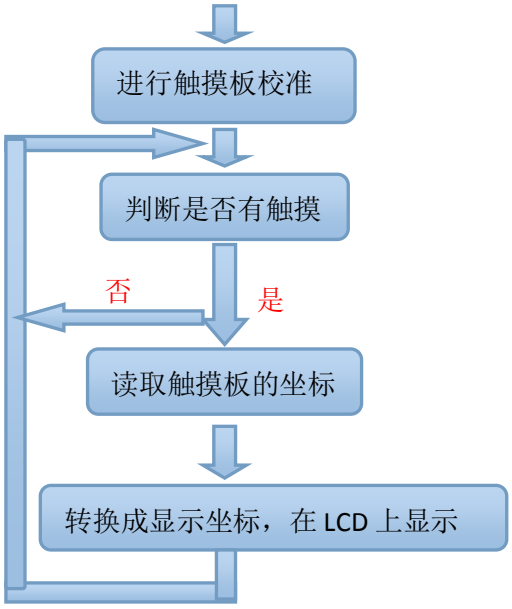
- HBP 和 VBP 可以通过 LTDC_BPCR 寄存器的 AHBP（LTDC_BPCR[27:16]）和 AVBP（LTDC_BPCR[10:0]）的值来设置。赋值方式为：AHBP= HSYNC Width + HBP - 1，AVBP= VSYNC Height + VBP - 1；

- 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；

程序流程：

初始化 IO，LCD 控制器,SDRAM 和触摸板的 I2C




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，初始化 TFT-LCD 的时间控制
**********************************************/
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，像素点的数据模式
************************************************************/
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;
```

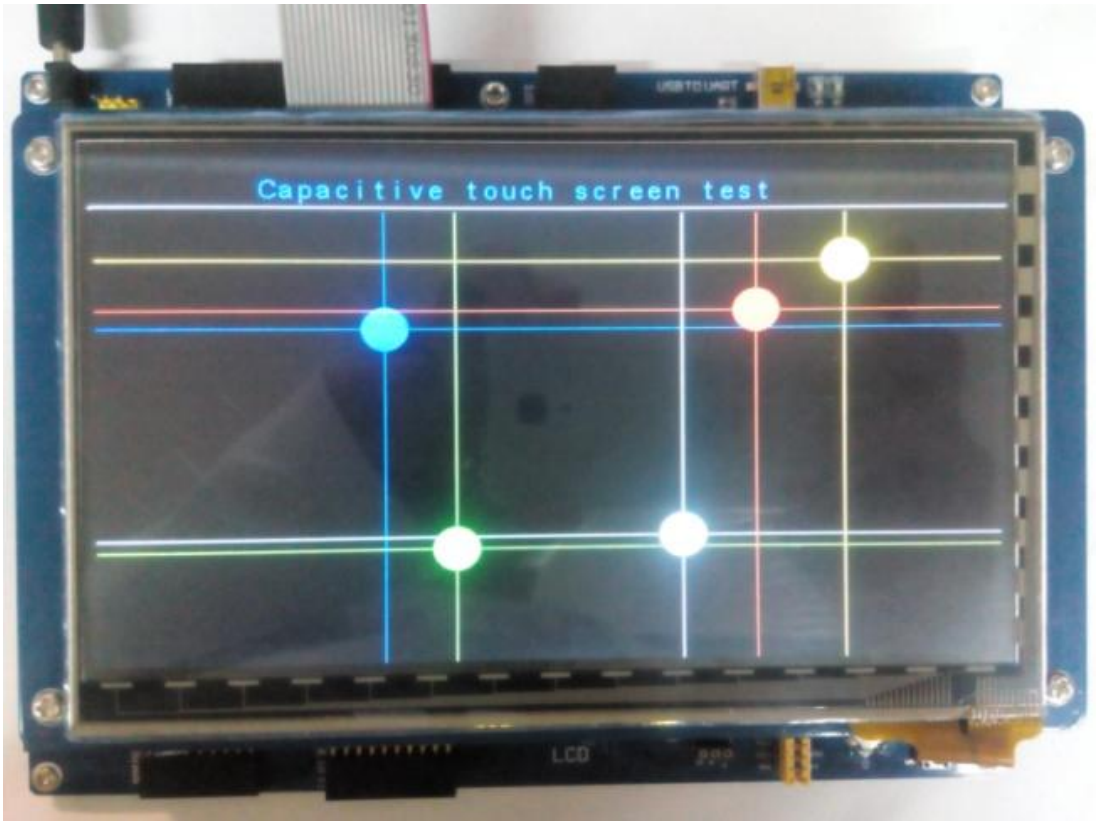
```
  /*
  Line Lenth = Active high width x number of bytes per pixel + 3
  Active high width          = LCD_PIXEL_WIDTH
  number of bytes per pixel = 2      (pixel_format : RGB565)
  */
  LTDC_Layer_InitStruct.LTDC_CFBLineLength = ((LCD_PIXEL_WIDTH * 2) + 3);
  /* pitch 是一行的开始到下一行的开始的字节数。
  Pitch = 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;
  }
}


/*****************************************************************************
下面是 FT5x06 的基本读写函数，FT5x06 不需要校准；直接 I2C 通信成功；读取出数据即可。
*****************************************************************************/
/*****************************************************************************
FT5x06 写入数据；指定地址 reg 连续写入 buf[len]；
*****************************************************************************/
u8 FT5x06_WR_Reg(u8 reg,u8 *buf,u8 len)
{
  u8 i;
  u8 ret=0;
  CT_I2C_Start();                    //Start

  CT_I2C_Send_Byte(CT_CMD_WR);   //Slaver Addr
  CT_I2C_Wait_Ack();                 //ACK

  CT_I2C_Send_Byte(reg);              //Data Addr
  CT_I2C_Wait_Ack();

  for(i=0;i<len;i++)
  {
    CT_I2C_Send_Byte(buf[i]);         //Data
    ret=CT_I2C_Wait_Ack();            //ACK
    if(ret)break;
  }
  CT_I2C_Stop();                     //Stop
  return ret;
}
/*****************************************************************************
FT5x06 读出数据；指定地址 reg 连续读出 buf[len]；
*****************************************************************************/
void FT5x06_RD_Reg(u8 reg,u8 *buf,u8 len)
```

```
{
    u8 i;
    CT_I2C_Start();                         //Start

    CT_I2C_Send_Byte(CT_CMD_WR);    //Slaver Addr
    CT_I2C_Wait_Ack();                  //Ack

    CT_I2C_Send_Byte(reg);              // Data Addr
    CT_I2C_Wait_Ack();                  //Ack

    CT_I2C_Stop();                          //Stop

    CT_I2C_Start();                         //Start
    CT_I2C_Send_Byte(CT_CMD_RD);    //Slaver Addr
    CT_I2C_Wait_Ack();                  //Ack

    for(i=0;i<len;i++)
    {
        buf[i]=CT_I2C_Read_Byte(i==(len-1)?0:1);
    }
    CT_I2C_Stop();                          //Stop
}
/*****************************************************************************
FT5x06 初始化
*****************************************************************************/
void FT5x06_Init(void)
{
    u8 version=0x0;

    CT_I2C_Init();                          //I2C 初始化
    TP_INT_Config();                    //触摸中断初始化

    FT5x06_WR_Reg(0x0,&version,1);      //配置 FT5x06 为正常操作模式；
    delay_ms(1);
}
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);

    delay_init();
    FT5x06_Init();
    delay_ms(10);
    While(1)
    {
      /*读取触摸的 XY 的位置，显示到 LCD 上*/
    FT5x06_Test();
    }
}
```

## 4. 程序验证效果




## 5. 模块尺寸大小