# 好钜润液晶屏与 TK499 接线通用说明

好钜润科技的显示屏例程一般主要分为 STM32 及 TK499 两大类。STM32 的例程相对功能少一点，TK499 的例程有简单有复杂，相对丰富一些，应用范围也更广更灵活。

常用的驱动接口有 STM32 的 FSMC， TKM32F499 的 TK80，及 LTDC，SPI，IIC 等，其中 IIC 主要用于驱动小的 OLED 屏，中小尺寸的可以用 SPI，分辨率高一点的例如 800x480 一般都用并口。另外，GPIO 也可以用来模拟时序来驱动 IIC、SPI 及 8080 接口的屏。好钜润科技的 TKM32F499 也常常弃用 TK80 等高级接口，用 GPIO 模拟 8080 时序的方式驱动液晶屏，这种例程主要针对 51 单片机应用场合，在 M4 上面模拟 51 编程方式以便参考。

建议从 TK499 的简单例程看起，LCD.C 及 LCD.H 文件是液晶驱动，驱动及移植看这两个文件就行。液晶屏驱动流程：配置相关 GPIO→复位液晶屏→写入初始化参数到液晶屏→液晶屏正常启动完成。

注意：①如果是 IPS 的液晶屏，BL 背光引脚给高电平使能后，背光虽然亮了，但由于 IPS 是常黑背景，所以只有少量光能透出来，你要对液晶屏初始化后，刷入颜色才能亮；②MCU 屏（又名 8080 接口屏），初始化后会是雪花状的，因为没有刷入颜色，所以屏会显示随机三原色，表达出雪花状，这时候已经表明液晶屏初始化已经完成；③如果背光是并联的，可以用 3.3V 串联限流电阻点亮，如果是背光是串联的，则要升压恒流源点亮。

常见的接线方式：

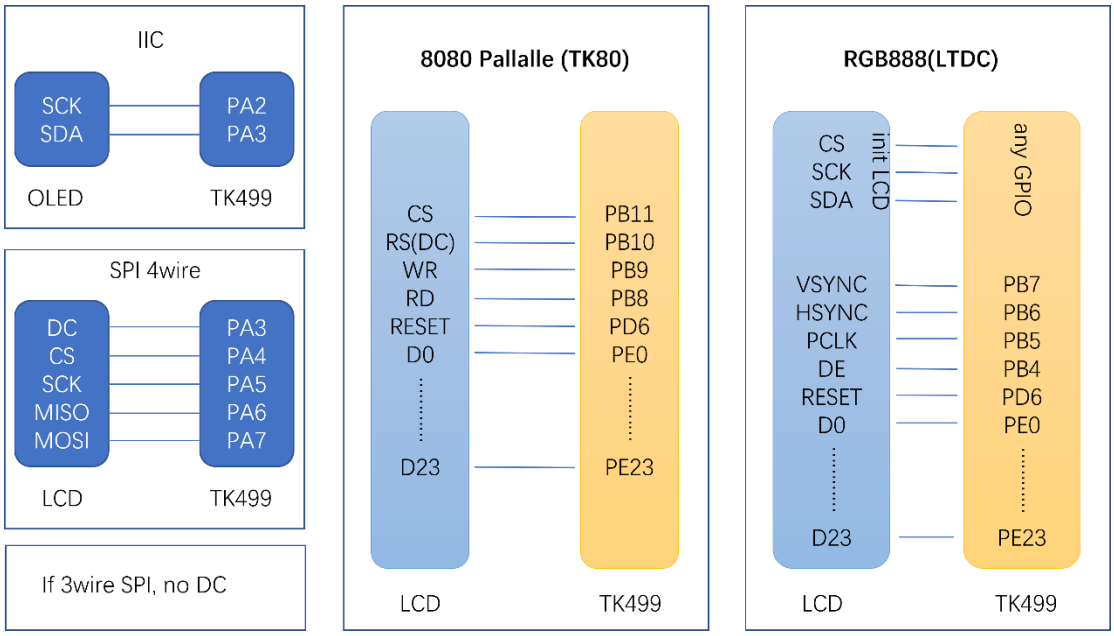
IIC：SCK、SDA

SPI：三线 SPI：CS、SCK、MISO(SDA)；四线 SPI：DC(RS)、CS、SCK、MISO(SDA)；

8080 并口：CS、RS(DC)、WR、RD、RESET、D0~D23（8 位：D0~D7；16 位 D0~D15）

LTDC(RGB)：初始化用 CS、SCK、MISO(SDA)；初始化后，进入正常刷数据：HSYNC、VSYNC、DE、PCLK、D0~D23（RGB888 可以向下兼容 RGB565 及 RGB666，只要高位对齐就行）；

常用的接线方法




驱动 8080 接口液晶屏：主要抓住三个根函数，分别是 WriteComm(unsigned int CMD) //写命令、WriteData(u32 dat) //写数据、BlockWrite(unsigned int Xstart,unsigned int

Xend,unsigned int Ystart,unsigned int Yend) //开窗函数。其它应用上层，都是依据这三个函数来构建的。

```
void WriteComm(unsigned int CMD) //写命令
{
  TK80->CMDIR = CMD;while(TK80->SR & 0x10000);
}

void WriteData(u32 dat) //写数据
{
  TK80->DINR = dat;while(TK80->SR & 0x10000);
}
/************************************************
函数名：开窗函数
入口参数：XStart x方向的起点
          Xend   x方向的终点
          YStart y方向的起点
          Yend   y方向的终点
这个函数的意义是：开一个矩形框，方便接下来往这个框填充数据
************************************************/
void BlockWrite(unsigned int Xstart,unsigned int Xend,unsigned int Ystart,unsigned int Yend)
{
  WriteComm(0x2a);
  WriteData(Xstart>>8);
  WriteData(Xstart);
  WriteData(Xend>>8);
  WriteData(Xend);

  WriteComm(0x2b);
  WriteData(Ystart>>8);
  WriteData(Ystart);
  WriteData(Yend>>8);
  WriteData(Yend);

  WriteComm(0x2c);
}
```

BlockWrite 这个函数，如果果坐标地址起始与结束相等，就是打一个点；如果仅是 x 坐标起止相等，那就是画一条竖线；如果仅是 y 坐标起止相等，则画一条横线；如果 x 及 y 坐标起止都不相等，那就是写一个区域。

举例 1：打点函数

```
//================ 在x，y 坐标上打一个颜色为Color的点 ================
void DrawPixel(u16 x, u16 y, int Color)
{
  BlockWrite(x,x,y,y);
  WriteData(Color);
}
```

举例 2：填充一个区域

```
/************************************************
函数名：Lcd矩形填充函数

入口参数：xStart x方向的起始点
          ySrart y方向的终止点
          xLong 要选定矩形的x方向长度
          yLong  要选定矩形的y方向长度
返回值：无
************************************************/
void Lcd_ColorBox(u16 xStart,u16 yStart,u16 xLong,u16 yLong,u32 Color)
{
    u32 temp;

  BlockWrite(xStart,xStart+xLong-1,yStart,yStart+yLong-1);//设定区域的长宽

  for (temp=0; temp<xLong*yLong; temp++)//循环往指定区域写入长*宽（xLong*yLong）那么多个点
  {
    WriteData(Color);
  }
}
```

驱动 RGB 接口液晶屏：TKM32F499 驱动 RGB 屏基本流程如下图

```
void LCD_Initial(void) //LCD初始化函数
{
  GPIO_RGB_INIT();            //初始化液晶屏相关GPIO
  LTDC_Clock_Set();           //设置LTDC时钟
  Set_LCD_Timing_to_LTDC();   //设置同步信号线时序，定义液晶屏显存地址
  Lcd_Initialize();           //写入初始化参数，初始化液晶屏
  Lcd_Light_ON;//打开背光
}
```

其中 GPIO 初始化，时钟时序设置的参考提供的程序，下面来说一下如何写入初始化参数来初始化液晶屏。主要也是三个根函数，LCD_WriteByteSPI 是模拟 SPI 函数，SPI_WriteComm 及 SPI_WriteData 分别是写命令及写命令参数。

```
void LCD_WriteByteSPI(unsigned char byte)//用GPIO模拟SPI
{
    unsigned char n;
    for(n=0; n<8; n++)
    {
        if(byte&0x80) SPI_SDA(1)
        else SPI_SDA(0)
        byte<<= 1;
        SPI_DCLK(0);
        SPI_DCLK(1);
    }
}
void SPI_WriteComm(u16 CMD)//模拟3线9bit SPI，写命令
{
  LCD_SPI_CS(0);
  SPI_SDA(0);     //9位数据的第一位为0说明这次是写入命令
  SPI_DCLK(0);
  SPI_DCLK(1);
  LCD_WriteByteSPI(CMD);
  LCD_SPI_CS(1);
}
void SPI_WriteData(u16 tem_data)//模拟3线9bit SPI，写命令参数
{
  LCD_SPI_CS(0);
  SPI_SDA(1);     //9位数据的第一位为1说明这次是写入命令参数
  SPI_DCLK(0);
  SPI_DCLK(1);
  LCD_WriteByteSPI(tem_data);
  LCD_SPI_CS(1);
}
```

完成了上面三个简单根函数的构建以后，就可以向液晶屏写入初始化参数了，液晶屏的初始化参数会对液晶屏的内部时钟，行场极性，各种极电压，VCOM 电压，频率、gamma 值，扫描方向等等设置。通过初始化后，液晶屏不仅能启动了，色彩也调得比较漂亮。一般用户不用去理会这些初始化参数，这些多数是由驱动 IC 原厂结合电子玻璃给出的一些内部设置，有的内部参数甚至都不写在手册上。主要需要关注的寄存器一般只有两个 0x3A 及 0x36.寄存器 0x3A 是设置色彩位宽； 0x36 通常是设置扫描方向及 RGB 三原色中的 R、B 两种颜色的顺序，不同的 IC 差别有点大，有的可以设置原始坐标，有的不行，最终还不如用单片机内部的显存旋转屏幕来得方便。

初始化完液晶屏后，接下来的操作就基本与液晶屏无关了。因为在单片机内部已经映射了一块 1:1 的显存，后面只需要对内存的操作就可以显示在屏上了。

例如：打点函数；由下图可以看出，在屏上打一个点，就是往一个内存数组上写入一个颜色值，是一个纯内存操作的数学运算。

```
//================ 在x，y 坐标上打一个颜色为Color的点 ================
void DrawPixel(u16 x, u16 y, int Color)
{
  LTDC_Buf[y+YSIZE_PHYS*x] = Color;
}
```

其它应用函数也是基于内存操作原理，写图片，写文字都是一个内存块的填充。

驱动 SPI 接口液晶屏：与 8080 接口驱动方式类似，仅根函数改为 SPI 输出，这里就不再赘述。

SPI 屏又分 3 线 SPI 及 4 线 SPI，一般可以笼统地认为 3 线 SPI 是 9 位，4 线 SPI 是 8 位的数据结构。但是也有特例，3 线可以用两个 8 位标定数据类型。3 线 SPI 屏，用最高位标定是数据或者命令。4 线 SPI 屏，用 DC(RS)引脚标定是数据还是命令。

**其它液晶屏相关知识**

所有液晶屏，无论是 MCU 8080 接口的液晶屏还是 SPI 屏或者 RGB888 的屏，都有驱动 IC，网上说的 MCU、SPI 屏有驱动，RGB 屏没驱动，这个说法是不正确的。如果非要论个有无，只能说 MCU、SPI 屏有显存，RGB 屏没有显存(或者有而不用)。为什么一定要驱动 IC，单片机不直接去驱动液晶屏呢？这个从原理上看就很明白了。液晶屏本质上是一个薄膜晶体管的矩阵，例如 800*480 分辨率的三原色屏，就有 p=800*480*3=1152000 那么多点阵，至少需要行列线高达 800+480*3=2240 个，单片机没有那么多引脚。所以必须要用驱动 IC 对接液晶屏点阵，再向单片机输出 8080 接口、SPI 接口或者 RGB 接口。有显存的屏，可以实现画面静态显示，单片机可以送一次数据给驱动 IC 的显存即可。没显存的屏，单片机必须不断循环送数据给驱动 IC，这样驱动 IC 才能有源源不断的数据对液晶屏的行列线刷新以维持画面的显示。所以无显存的 RGB 屏是比较耗资源的，那么大量的数据不断在刷，充放电效应会令功耗显著增加。

其它注意事项：

1. 电容触摸一般驱动 IC 分为两大派系，均为国产，GT911 及 FT5XXX，FT6XXX，其中 FT 系列的程序全系兼容，不用太在乎后缀。
2. 有的屏自带双面胶，或者灰色导电胶；这些胶都是起做临时固定作用，方便组装用的。如果你是产品用，请勿依靠这些胶来固定，时间稍长一点胶会落或者移位。如果只是研发样品或者 DIY 用，那注意平放使用就行。

常用颜色值定义

```
//*************  24 位色（1600 万色）定义 *************//
#define WHITE            0xFFFFFF
#define BLACK            0x000000
#define BLUE             0x0000FF
#define BLUE2            0x3F3FFF
#define RED              0xFF0000
#define MAGENTA          0xFF00FF
#define GREEN            0x00FF00
#define CYAN             0x00FFFF
```

```
#define YELLOW          0xFFFF00
```