2012

ZX043P 规格书

版本	修改日期	备注
01	2011/08/30	初始版本
02	2012/09/20	详细规格书

www.zxlcd.com

中显液晶并口模组

ZX043P480272 彩色液晶模组规格书

简述：

本模组的控制核心部分采用 ACTEL 的 FPGA+高速 SRAM 架构编程实现 ,性能稳定可靠 ,抗干扰极强，功耗低。

该方案专门针对工业产品及嵌入式应用而做的优化设计，具有简单易用、稳定可靠等特点，是工业应用的理想选择。使用该模组可大幅度降低系统复杂度、简化用户设计、降低成本和加快产品上市，可显著提高用户产品的竞争优势 ,并提供完整的演示应用程序和 24 小时的技术支持 ,方便客户产品迅速完成开发。

功能特性：

- ✓ 接口完全兼容 RA8835/SED1335 等 IC 的通信接口，不需修改硬件，就能直接从单色升级为彩色；
- ✓ 8 位高速 8080 并行总线接口，读写无需专门的忙信号 IO，大数据量传输无雪花；
- ✓ 指令和 CPU 接口完全兼容本公司其他型号并口彩屏，升级更换方便；
- ✓ X、Y 地址输入（X、Y 地址与显示屏水平象素、垂直象素一一对应）;
- ✓ 显示页和读写页可以分开操作，切换显示无延时无雪花；
- ✓ 硬件 8 点写入加速功能(比传统的单点写入的速度快十几倍 ,特别适合于字体的写入 ,还可透明写入);
- ✓ 硬件全屏填充加速功能（对当前页瞬间填充）;
- ✓ 读/写时可独立设置 X 方向或 Y 方向自动增量；
- ✓ 16 位色彩深度（RGB565 格式），工业 65K 色应用显示；
- ✓ 控制板上附有背光升压电路和 VGH/VGL/VCOM 电压电路，用户无需添加额外的电路板；
- ✓ 可选配触摸屏（TI 的 ADS7843 芯片，SPI 接口）;
- ✓ 工业级卡口固定方式，方便用户固定；
- ✓ 高亮 LED 背光，2 万小时以上寿命；
- ✓ 通用 2.54MM 双排针接口或 1.0MM FFC 软排线接口。

应用领域：

- ✓ 人机界面
- ✓ 仪表仪器
- ✓ 医疗产品
- ✓ 设备控制
- ✓ 智能电表
- ✓ 美容仪器

1. 基本特性

➢ 电气特性

项目	特性	单位
电源	5V 或者 3.3V	伏特
电流	--	毫安
逻辑电平	3.3V LVCOM	
接口类型	8080 并口（8 位数据线）	

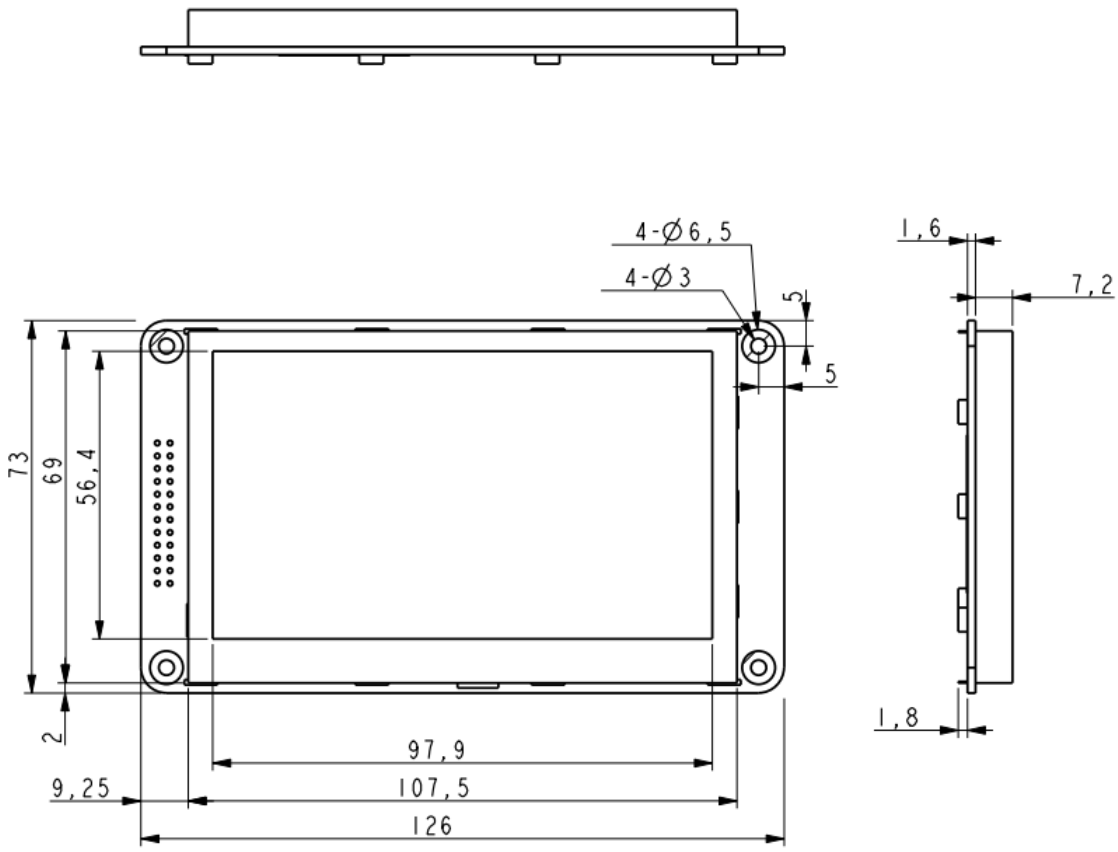
➢ 液晶特性

项目	特性	单位
尺寸	4.3	英寸
液晶分辨率	480RGB×272	点
背光类型	LED 背光	
温度	-20～70	摄氏度

2. 产品图片

注：图片跟实物可能有微小变化，请以实物为准。

3. 结构图

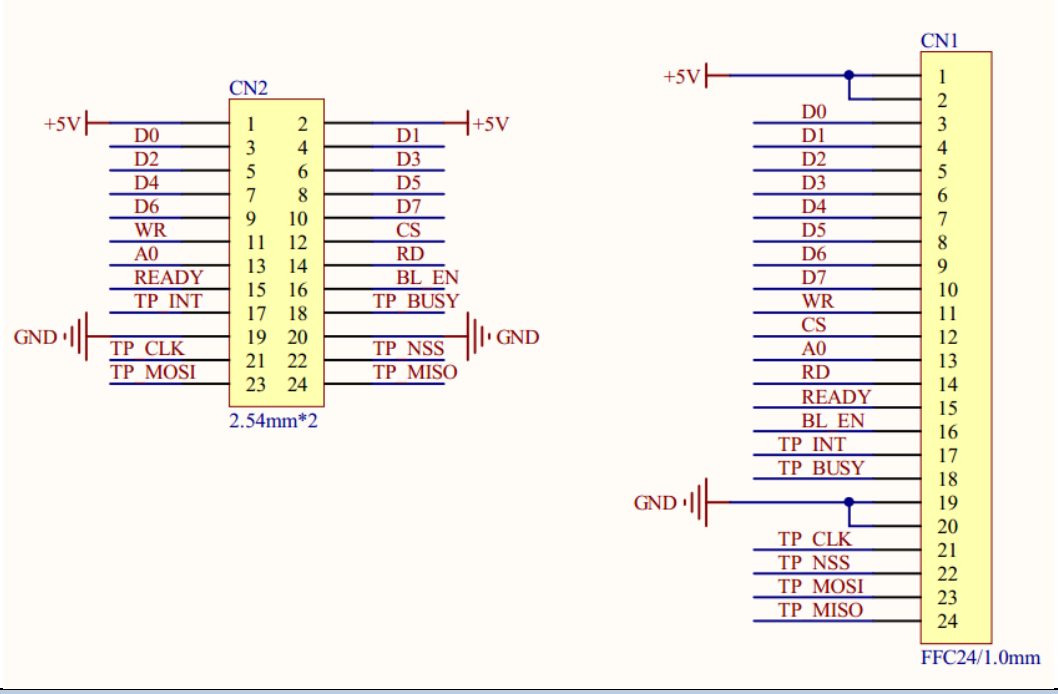


4. 接口及时序：

➢ CPU 接口引脚描述

模块	引脚名称	类型	功能				备注
液晶通信接口	GND	电源	电源地				
	VCC	电源	电源正（+5V）				
	BL_EN	输入	背光控制信号，高开通，可用 PWM 控制背光亮度				3.3V CMOS 电平
	A0	输入	**A0**	**RD**	**WR**	**功 能**	3.3V CMOS 电平
			0	0	1	读数据	
			1	0	1	读状态	
			0	1	0	写数据	
			1	1	0	写指令	
	WR/WE	输入	写信号，低有效，上升沿时，锁存当前数据				3.3V CMOS 电平
	RD	输入	读信号，低有效				3.3V CMOS 电平
	D0～D7	输入/输出	数据总线				3.3V CMOS 电平
	CS	输入	片选，低有效				3.3V CMOS 电平
	READY	输出	忙信号，高表示忙状态，如用寄存器判忙，此引脚可以不接				
触摸信号接口	TP_BUSY	输出	触摸屏忙信号输出				3.3V CMOS 电平
	TP_INT	输出	触摸屏中断信号				3.3V CMOS 电平
	TP_MISO	输出	串行数据输出				3.3V CMOS 电平
	TP_MOSI	输入	串行数据输入				3.3V CMOS 电平
	TP_NSS	输入	触摸屏片选				3.3V CMOS 电平
	TP_CLK	输入	触摸屏串行时钟信号				3.3V CMOS 电平

➢ 连接器接口定义



CN2（2.54mm 双排针）/ CN1（1.0mm FFC 连接器）		
引脚	**符号**	**备注**
1~2	VCC	
3~10	D0～D7	
11	WR	
12	CS	
13	A0	
14	RD	
15	READY	
16	BL_EN	
17	TP_INT	
18	TP_BUSY	
19~20	GND	
21	TP_CLK	
22	TP_NSS	
23	TP_MOSI	
24	TP_MISO	

注 1: BL_EN 可以通过一个 10Khz 左右的 PWM 的信号来控制亮度等级，如不需要控制亮度等级，接高电平时（不超过 3.3V 逻辑电平）背光全开，接低电平时背光关闭。

5. 指令表

指令助记符	指令码 (CMD)	数据低字节 (LDATA)	数据高字节 (HDATA)	数据长度 (NUMBER)	备注

SET_X	0x01	X[7:0]	X[8]	2	设置 X 坐标寄存器
SET_Y	0x02	Y[7:0]	Y[8]	2	设置 Y 坐标寄存器
SET_BACK_COLOR	0x03	BCOLOR[7:0]	BCOLOR[15:8]	2	设置背景色寄存器
SET_FORE_COLOR	0x04	FCOLOR[7:0]	FCOLOR[15:8]	2	设置前景色寄存器
SET_PMODE	0x05	PMODE	-	1	设置页码模式寄存器
SET_DMODE	0x06	DMODE	-	1	设置数据模式寄存器
SET_DATA	0x07	RDATA[7:0]	RDATA[15:8](注)	N	写入数据寄存器
CLR_SCREEN	0x08	CCOLOR[7:0]	CCOLOR[15:8]		清屏

注：单点写入模式的时候才要写入 RDATA[15:8]。

➢ 设置 X 坐标寄存器指令（SET_X）：

X：X 坐标寄存器															
HDATA								LDATA							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
--（注 3）	--	--	--	--	--	--	X8	X7	X6	X5	X4	X3	X2	X1	X0

注 3：--为无关参数，未使用，建议置 0；下同。

函数示例：

```
// 设置 X 坐标
void SetX(uint16 x)
{
    WriteCmd(CMD_X);  // 写入指令头
    WriteData(x & 0xff); // 写入低 8 位数据
    WriteData((x>>8)& 0xff); // 写入高 8 位数据
}
```

➢ 设置 Y 坐标寄存器指令（SET_Y）：

Y：Y 坐标寄存器															
HDATA								LDATA							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
--	--	--	--	--	--	--	Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

函数示例：

```
// 设置 Y 坐标
void SetY(uint16 y)
{
    WriteCmd(CMD_Y);  // 写入指令头
    WriteData(y & 0xff); // 写入低 8 位数据
    WriteData((y>>8)& 0xff); // 写入高 8 位数据
}
```

➢ 设置背景色寄存器指令（SET_BACK_COLOR）：

BCOLOR：背景色寄存器															
HDATA								LDATA							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
BCOLOR15	BCOLOR14	BCOLOR13	BCOLOR12	BCOLOR11	BCOLOR10	BCOLOR9	BCOLOR8	BCOLOR7	BCOLOR6	BCOLOR5	BCOLOR4	BCOLOR3	BCOLOR2	BCOLOR1	BCOLOR0

注：颜色模式为 RGB565 格式，即红色占 5bit（D15~D11），绿色占 6bit(D10~D5)，蓝色占 5bit(D4~D0)。一共可显示 65K 种颜色。

函数示例：

```
// 设置背景色
void SetBackColor(uint16 color)
{
    WriteCmd(CMD_BACK_COLOR);  // 写入指令头
    WriteData(color & 0xff); // 写入低 8 位数据
    WriteData((color>>8)& 0xff); // 写入高 8 位数据
}
```

➢ 设置前景色寄存器指令（SET_FORE_COLOR）：

FCOLOR：背景色寄存器															
HDATA								LDATA							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
FCOLOR15	FCOLOR14	FCOLOR13	FCOLOR12	FCOLOR11	FCOLOR10	FCOLOR9	FCOLOR8	FCOLOR7	FCOLOR6	FCOLOR5	FCOLOR4	FCOLOR3	FCOLOR2	FCOLOR1	FCOLOR0

函数示例：

```
// 设置前景色
void SetForeColor(uint16 color)
{
    WriteCmd(CMD_FORE_COLOR);  // 写入指令头
    WriteData(color & 0xff); // 写入低 8 位数据
    WriteData((color>>8)& 0xff); // 写入高 8 位数据
}
```

➢ 设置页码模式寄存器指令（SET_PMODE）：

PMODE：页码模式寄存器							
D7	D6	D5	D4	D3	D2	D1	D0
--	--	--	--	--	opt1	--	disp0

disp=0，显示第一页显存数据；

disp=1，显示第二页显存数据；

opt=0，操作第一页显存数据；

opt=1，操作第二页显存数据；

函数示例：

```
// 设置页面模式寄存器
void SetModePage(uint8 data)
{
    WriteCmd(CMD_MODE_PAGE);  // 写入指令头
    WriteData(data); // 写入低 8 位数据
}
```

➢ 设置数据模式寄存器指令（SET_DMODE）：

DMODE：数据模式寄存器							
D7	D6	D5	D4	D3	D2	D1	D0
--	--	--	--	yinc	xinc	wrcon1	wrcon0

wrcon 标志位说明:

wrcon1	wrcon0	功能
0	0	单点操作模式
0	1	8 点操作模式
1	0	8 点透明操作模式

1	1	未定义

xinc 标志位说明：X 坐标自动递增控制位。

xinc	读/写	单点操作模式	8 点透明操作模式	8 点操作模式
1	写数据	X 坐标自动加 1，满行时自动切换到下一行	X 坐标自动加 8，满行时自动切换到下一行	X 坐标自动加 8，满行时自动切换到下一行
	读数据	X 坐标自动加 1，满行时自动切换到下一行	未定义，请勿使用	未定义，请勿使用
0	写数据	X 坐标保持当前值不变	X 坐标保持当前值不变	X 坐标保持当前值不变
	读数据	X 坐标保持当前值不变	X 坐标保持当前值不变	X 坐标保持当前值不变

yinc 标志位说明：Y 坐标自动递增控制位。

yinc	读/写	单点操作模式	8 点透明操作模式	8 点操作模式
1	写数据	Y 坐标自动加 1	Y 坐标自动加 1	Y 坐标自动加 1
	读数据	Y 坐标自动加 1	未定义，请勿使用	未定义，请勿使用
0	写数据	Y 坐标保持当前值不变	Y 坐标保持当前值不变	Y 坐标保持当前值不变
	读数据	Y 坐标保持当前值不变	Y 坐标保持当前值不变	Y 坐标保持当前值不变

详细解释：

单点操作模式：一次只能写入一个点，直接将颜色值写入数据寄存器 RDATA[15:0]中，当前坐标点颜色被写入 RDATA[15:0]，而与前景色、背景色寄存器内容无关；

8 点操作模式：一次写入 8 个点，将点位信息写入数据寄存器 RDATA[7:0]中，对应点位信息为 1 的写入前景色，为 0 的写入背景色。

如写入数据寄存器为‘01010101b’则对应点位显示‘背景色、前景色、背景色、前景色、背景色、前景色、背景色、前景色’。

8 点透明操作模式：一次写入 8 个点，将点位信息写入数据寄存器 RDATA[7:0]中，对应点位信息为 1 的写入前景色，为 0 的不写入，保持原有色不变。

如写入数据寄存器为‘01010111b’则对应点位显示‘原有色、前景色、原有色、前景色、原有色、前景色、前景色、前景色’。

8 点操作模式和 8 点透明操作模式的点位信息与 XY 坐标的对应关系：

xinc=1 时，

RDATA7	RDATA 6	RDATA 5	RDATA 4	RDATA 3	RDATA 2	RDATA 1	RDATA 0
X	X+1	X+2	X+3	X+4	X+5	X+6	X+7

yinc=1 时，

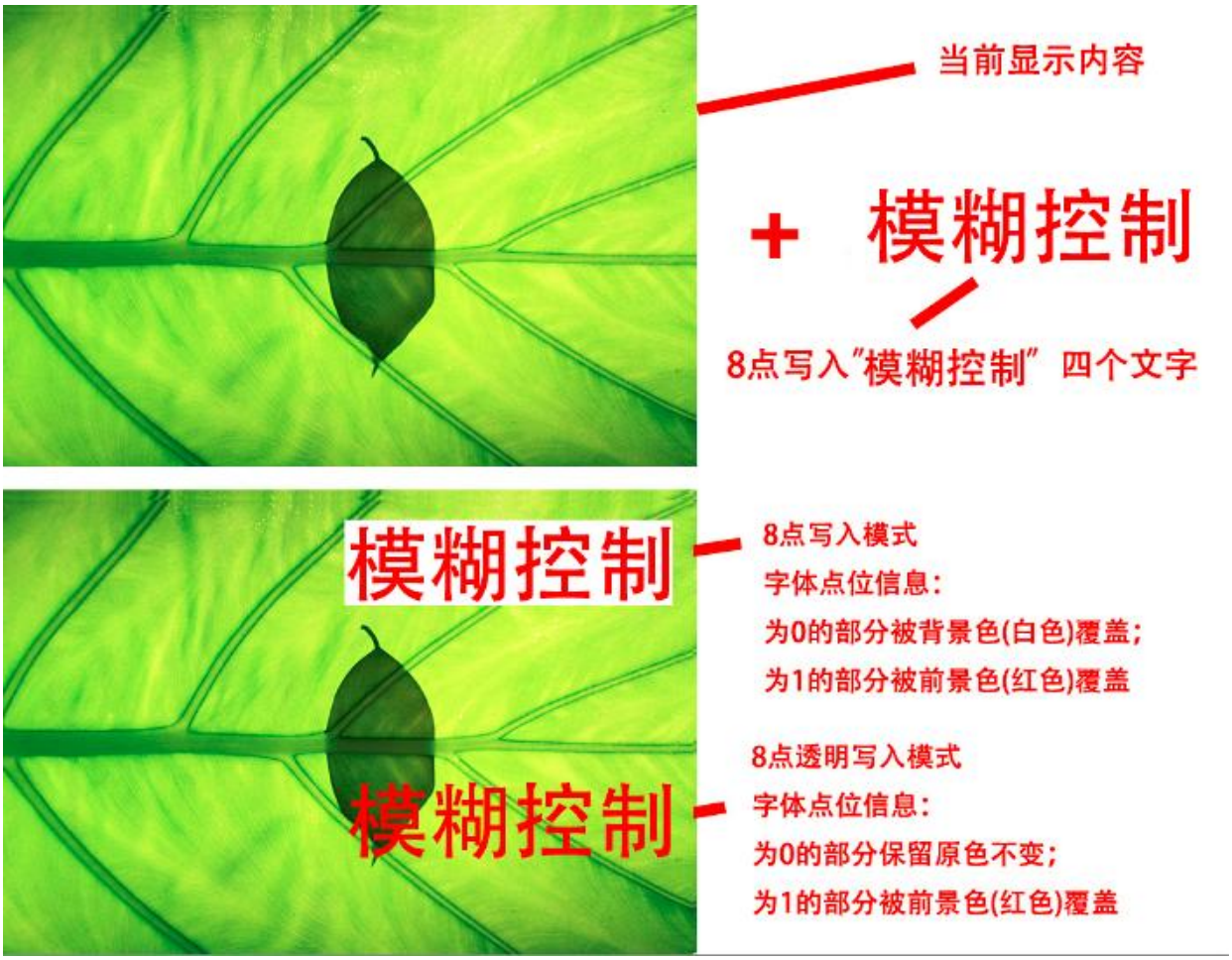
RDATA7	RDATA 6	RDATA 5	RDATA 4	RDATA 3	RDATA 2	RDATA 1	RDATA 0
Y	Y+1	Y+2	Y+3	Y+4	Y+5	Y+6	Y+7

函数示例：

```
// 设置数据模式寄存器
void SetModeData(uint8 data)
{
```

```
    WriteCmd(CMD_MODE_DATA);  // 写入指令头
    WriteData(data); // 写入低 8 位数据
}
```

8 点功能效果图



➢ 写入数据寄存器指令（SET_DATA）：

RDATA：数据寄存器															
HDATA								LDATA							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
RDATA15	RDATA14	RDATA13	RDATA12	RDATA11	RDATA10	RDATA9	RDATA8	RDATA7	RDATA6	RDATA5	RDATA4	RDATA3	RDATA2	RDATA1	RDATA0

将数据总线上的数据暂存到 RDATA 寄存器中。配合其他指令，可以实现单点写，单点连续写，8 点写，8 点连续写等功能。

函数示例：

```
// 写数据
void SetRdata(uint16 data)
{
    WriteCmd(CMD_DATA);  // 写入指令头
    WriteData(data & 0xff); // 写入低 8 位数据
    WriteData((data>>8)& 0xff); // 写入高 8 位数据
```

```
}
```

➢ 硬件清屏指令（CLR_SCREEN）：

CCOLOR：清屏颜色寄存器															
HDATA								LDATA							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
CCOLOR15	CCOLOR14	CCOLOR13	CCOLOR12	CCOLOR11	CCOLOR10	CCOLOR9	CCOLOR8	CCOLOR7	CCOLOR6	CCOLOR5	CCOLOR4	CCOLOR3	CCOLOR2	CCOLOR1	CCOLOR0

在极短时间内，当前操作页面被清屏成指定色。

函数示例：

```
// 用指定色清屏当前操作层
void ClrScreen(uint8 color)
{
    //SetForeColor(color);  // 设置前景色
    SetModeData(reg_XINC | reg_SDOT);  // 设置 Mdata 模式寄存器
    WriteCmd(CMD_CLR_SCREEN);  // 写入指令头
    WriteData(color & 0xff); // 写入低 8 位数据
    WriteData((color>>8)& 0xff); // 写入高 8 位数据
    while((ReadStatus() & 0x01)==0x01); // 读取清屏满标志位，数据总线的 D0 位
}
```

➢ 常见问题

问：如何在清屏或 8 点写入模式过程中读取忙状态？

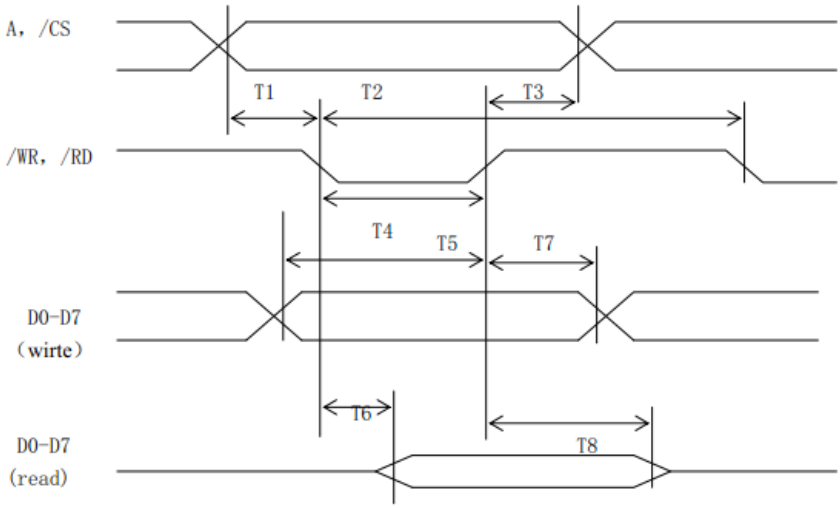
答：可通过读状态指令读取。返回数据位的最低位为忙状态标志位，为高时，系统忙；为低时，系统不忙。

对于单片机，因为 FPGA 的速度够快，8 点写入完全不用进行忙判断，清屏的时候，读取状态标志位就可以，也可以通过延时来实现。

问：如何快速完成显示测试？

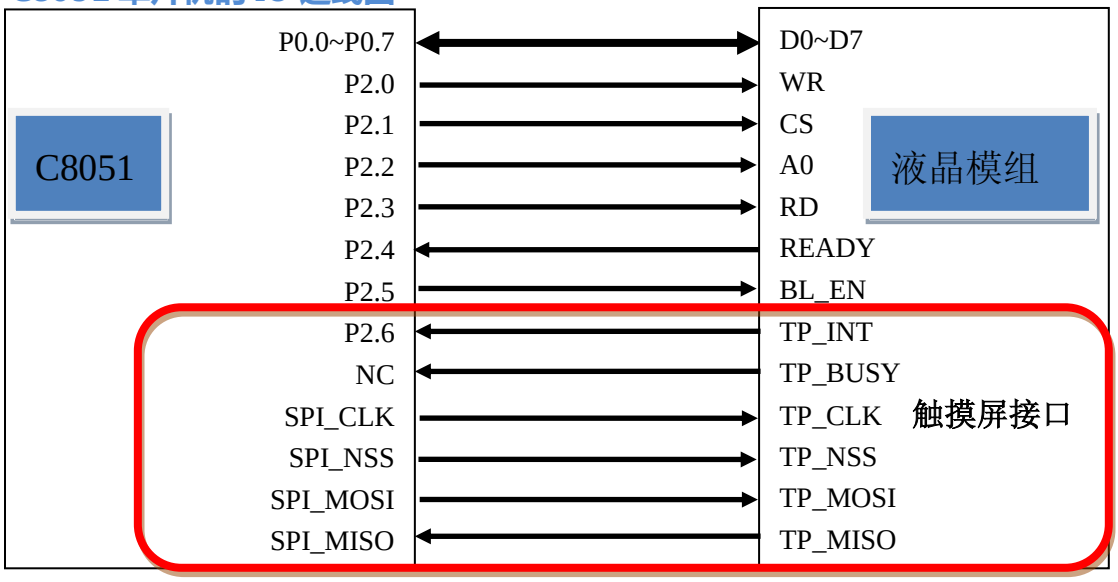
答：将演示程序移植，只需要移植读/写 4 个函数就可以了。

6. 接口时序图

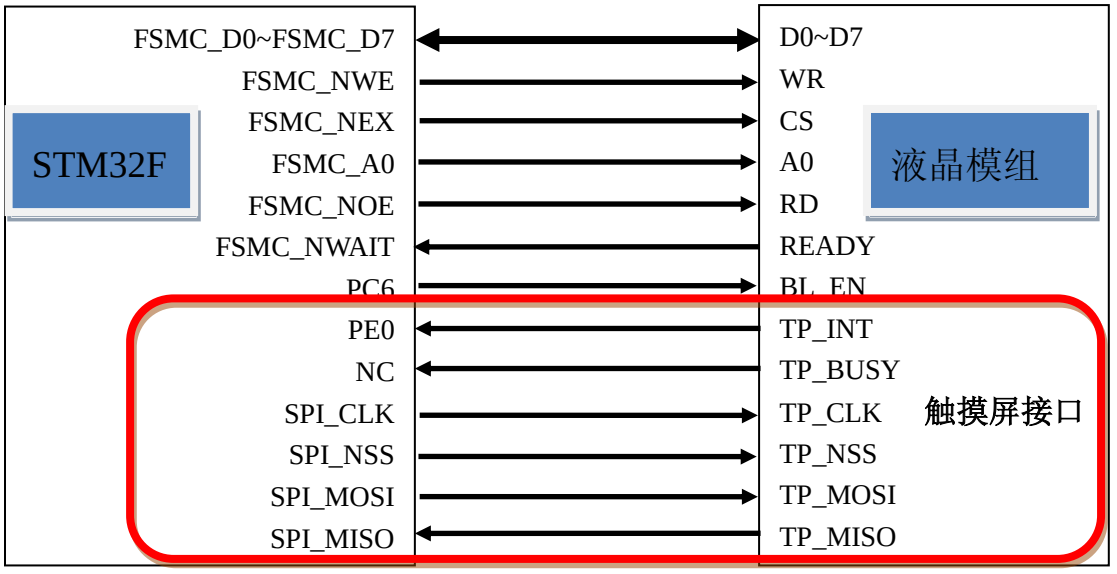


7. 典型接线图

➢ 与 C8051 单片机的 IO 连线图



➢ 模组与 STM32F103ZX 系列的总线连线图



注 1：NC 在本图中表示可不连接。

注 2：READY 可以不接，可用读取寄存器状态的方式来替代。

注 3：触摸屏接口未用到的，可以不接。

8. 演示程序

演示程序：完整的演示程序，请从网站 www.zxlcd.com 下载或者向市场人员索取。

24 小时技术支持：电话：13001234565 刘工 QQ：214496161

部分函数附录：

```
// 写入指令
void WriteCmd(uint8 cmd)
{
      // 总线模式
      *((uint8*)(FSMC_BASE+1)) = cmd;

      // IO 模式
      // RD = 1;
      // CS=0;
      // A0=1;
      // WR=0;
      // DBUS=cmd;
      // WR=1;
      // CS=1;

}

// 写入数据
void WriteData(uint8 data)
{
      // 总线模式
      *((uint8*)(FSMC_BASE)) = data;

      // IO 模式
      // RD = 1;
      // CS=0;
      // A0=0;
      // WR=0;
      // DBUS=data;
      // WR=1;
      // CS=1;
}

//读状态
uint8 ReadStatus(void)
{
uint8 data;
      // 总线模式
     data=*((uint8*)(FSMC_BASE+1));
     return (data);

      // IO 模式
      // WR = 1;
      // CS=0;
      // A0=1;
```

```
        // RD=0;
        // data=DBUS;
        // RD=1;
        // CS=1;

}

//读数据
uint8 ReadData(void)
{
uint8 data;
        // 总线模式
      data=*((uint8*)(FSMC_BASE));
      return (data);

        // IO 模式
        // WR = 1;
        // CS=0;
        // A0=0;
        // RD=0;
        // data=DBUS;
        // RD=1;
        // CS=1;

}

// 设置 X 坐标
void SetX(uint16 x)
{
        WriteCmd(CMD_X);  // 写入指令头
        WriteData(x & 0xff); // 写入低 8 位数据
        WriteData((x>>8)& 0xff); // 写入高 8 位数据
}

// 设置 Y 坐标
void SetY(uint16 y)
{
        WriteCmd(CMD_Y);  // 写入指令头
        WriteData(y & 0xff); // 写入低 8 位数据
        WriteData((y>>8)& 0xff); // 写入高 8 位数据
}

// 设置 XY 坐标
void SetXY(uint16 x, uint16 y)
{
        SetX(x);
        SetY(y);
}

// 设置背景色
void SetBackColor(uint16 color)
{
```

```
    WriteCmd(CMD_BACK_COLOR);  // 写入指令头
    WriteData(color & 0xff); // 写入低 8 位数据
    WriteData((color>>8)& 0xff); // 写入高 8 位数据
}

// 设置前景色
void SetForeColor(uint16 color)
{
    WriteCmd(CMD_FORE_COLOR);  // 写入指令头
    WriteData(color & 0xff); // 写入低 8 位数据
    WriteData((color>>8)& 0xff); // 写入高 8 位数据
}

// 设置页面模式寄存器
void SetModePage(uint8 data)
{
    WriteCmd(CMD_MODE_PAGE);  // 写入指令头
    WriteData(data); // 写入低 8 位数据
}

// 设置数据模式寄存器
void SetModeData(uint8 data)
{
    WriteCmd(CMD_MODE_DATA);  // 写入指令头
    WriteData(data); // 写入低 8 位数据
}

// 写数据
void SetRdata(uint16 data)
{
    WriteCmd(CMD_DATA);  // 写入指令头
    WriteData(data); // 写入低 8 位数据
    WriteData(x & 0xff); // 写入低 8 位数据
    WriteData((x>>8)& 0xff); // 写入高 8 位数据
}

// 画点
void DrawPixel(uint16 color, uint16 x, uint16 y)
{
    SetXY(x,y);
    WriteCmd(CMD_DATA);  // 写入指令头
    SetRdata(color);
}

// 用指定色清屏当前操作层
void ClrScreen(uint16 color)
{
    //SetForeColor(color);  // 设置前景色
    SetModeData(reg_XINC | reg_SDOT);  // 设置 Mdata 模式寄存器
    WriteCmd(CMD_CLR_SCREEN);  // 写入指令头
    WriteData(color & 0xff); // 写入低 8 位数据
    WriteData((color>>8)& 0xff); // 写入高 8 位数据
```

```
    while((ReadStatus() & 0x01)==0x01); // 读取清屏满标志位，数据总线的 D0 位
}

// 设置显示模式
void SetDispMode(uint8 mode)
{
    WriteCmd(CMD_DISP_MODE);  // 写入指令头
    WriteData(mode); // 随便写入一个参数
}

// 画矩形边框
void DrawRect(uint16 color,uint16 x,uint16 y,uint16 w,uint16 h)
{
uint16 i,j;
    DrawHLine(color,x,y,w);  // 画水平线
    DrawVLine(color,x,y,h);  // 画垂直线
    DrawHLine(color,x,y+h-1,w);  // 画水平线
    DrawVLine(color,x+w-1,y,h);  // 画垂直线
}

//画实心矩形
void DrawFillRect(uint16 color,uint16 x,uint16 y,uint16 w,uint16 h)
{
uint16 i,j;
    SetForeColor(color);
    SetModeData(reg_XINC | reg_SDOT); // 单点写入
    for(i=0;i<h;i++)
    {
        SetXY(x,y);
        y++;
        WriteCmd(CMD_DATA);
        for(j=0;j<w;j++) // 连续写点
        {
            WriteData(color);
            //Delayus(1);//delay 450ns
        }
    }
}

//画一条横向直线
void DrawHLine(uint16 color,uint16 x,uint16 y,uint16 w)
{
uint16 i;

    SetForeColor(color);
    SetModeData(reg_XINC | reg_SDOT); // 单点写入
    SetXY(x,y);
    WriteCmd(CMD_DATA);
    for(i=0;i<w;i++)
    {
        WriteData(color & 0xff); // 写入低 8 位数据
        WriteData((color>>8)& 0xff); // 写入高 8 位数据
```

```
    }
}

//画一条垂直直线
void DrawVLine(uint16 color,uint16 x,uint16 y,uint16 h)
{
uint16 i;

    SetForeColor(color);
    SetModeData(reg_YINC | reg_SDOT); // 单点写入
    SetXY(x,y);
    WriteCmd(CMD_DATA);
    for(i=0;i<h;i++)
    {
        WriteData(color & 0xff); // 写入低 8 位数据
        WriteData((color>>8)& 0xff); // 写入高 8 位数据
    }
}

// 显示图片
void display_pic(uint16 x,uint16 y,uint16 w,uint16 h, const unsigned char *gImage)
{
uint16 i,j;
    SetModeData(reg_XINC | reg_SDOT);
    for(i=0;i<h;i++)
    {
        SetXY(x,y+i);
        WriteCmd(CMD_DATA);
        for(j=0;j<2*w;j++)
        {
            WriteData(gImage[i*2*w+j]);
            j++;
            WriteData(gImage[i*2*w+j]);
        }
    }
}

//写汉字程序___一次写 8 点
void DrawHZMDOT(uint16 x, uint16 y, uint8 xNum, uint8 yNum, const uint8 *pData,uint16 forecolor,uint16 backcolor)
//x,y:字符左上角位置
//xNum=字符一行的点阵数/8，如 16 点阵字=2
//yNum=字符行数，如 16 点阵字符=16
//pData 字符首地址
//forecolor 字体色,backcolor 字体背景色
{
uint8 i,j;
uint16 n=0;
    SetForeColor(forecolor);
    SetBackColor(backcolor);
    SetModeData(reg_XINC | reg_MDOT); // 8 点写入

    for(i=0;i<yNum;i++)
```

```
    {
        SetXY(x,y);
        y++;
        WriteCmd(CMD_DATA);
        for(j=0;j<xNum;j++)
        {
            WriteData(pData[n++]);
            //Delayus(1);
        }
    }
}

//写汉字程序___一次写 8 点, 透明写
void DrawHZTDOT(uint16 x, uint16 y, uint8 xNum, uint8 yNum, const uint8 *pData,uint16 forecolor)
//x,y:字符左上角位置
//xNum=字符一行的点阵数/8，如 16 点阵字=2
//yNum=字符行数，如 16 点阵字符=16
//pData 字符首地址
//forecolor 字体色,backcolor 字体背景色
{
uint8 i,j;
uint16 n=0;
    SetForeColor(forecolor);
    SetModeData(reg_XINC | reg_TDOT); // 8 点写入

    for(i=0;i<yNum;i++)
    {
        SetXY(x,y);
        y++;
        WriteCmd(CMD_DATA);
        for(j=0;j<xNum;j++)
        {
            WriteData(pData[n++]);
            //Delayus(1);
        }
    }
}
```