7 TFT LCD 屏显示实验

7.1. 背景介绍

在外部 SRAM 实验中简单的介绍了一下 AHB 总线，并且通过 AHB 总线成功访问了外部的 SRAM。其实 LCD 的操作时序与 SRAM 控制类似，唯一不同的就是 SRAM 有地址控制信号，而 TFT LCD 只有 RS 信号，没有地址信号，但是可以选用 SRAM 的其中任意一条地址信号来作为 TFT LCD 的 RS 信号。

本次实验使用的 4.3 寸的 TFT LCD 屏，该屏的接口是通用的，显示屏的控制器为 NT35510，下面我们来简单的了解一下 TFT LCD 显示屏。

7.1.1. TFT LCD 简介

TFT(ThinFilm Transistor)LCD 即薄膜场效应晶体管 LCD，是有源矩阵类型液晶显示器(AM-LCD)中的一种。TFT-LCD 的主要特点是为每个像素配置一个半导体开关器件。由于每个像素都可以通过点脉冲直接控制，因而每个节点都相对独立，并可以进行连续控制。这样的设计方法不仅提高了显示屏的反应速度，同时也可以精确控制显示灰度，这就是 TFT 色彩较 DSTN 更为逼真的原因。

本次实验使用到的是 4.3 寸的 TFT LCD 电容触摸屏模块。该屏幕的分辨率为 800*480，16 位真彩显示，采用 NT35510 驱动，该芯片自带 GRAM，无需外加驱动器。该模块具有如下几个特点：

1. 高分辨率：800*480；
2. 自带驱动，无需外加驱动器；
3. 速度快，理论上最高刷屏速率可达 78.9 帧/秒；
4. 采用电容触摸屏，最大支持 5 点触摸；
5. 采用背光电路，只需加 3.3/5V 供电即可，无需外加高压；

TFT LCD 模块采用 2*17 的 2.54 公排针与外部连接（即开发板的接口），接口定义如下图 7. 1 所示。

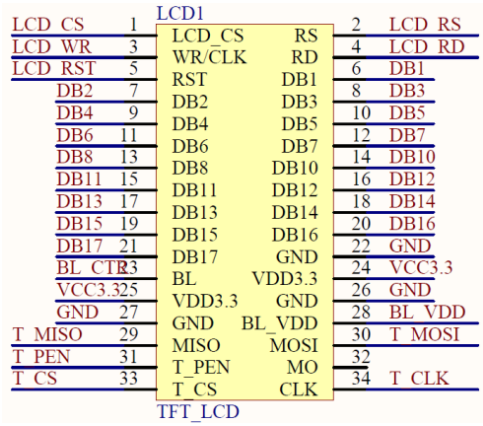

图 7. 1 4.3 寸 TFTLCD 模块接口图

在 AC208 开发板上对应的连接接口如下图 7. 2 所示，使用时，只需将对应的接口相接即可。

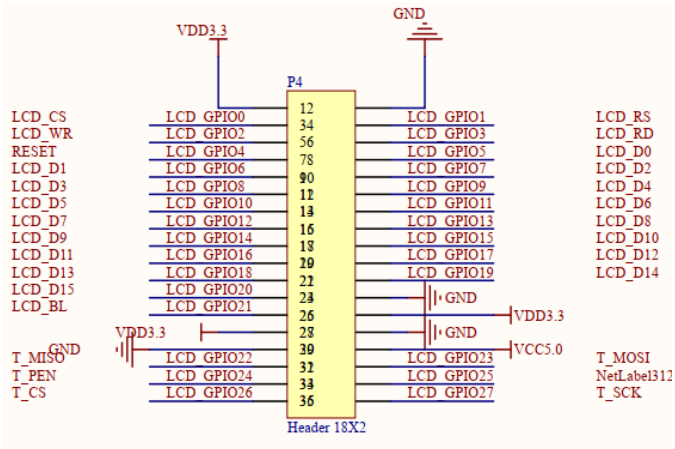

图 7. 2AC208 开发板上 LCD 屏接口

对模块的引脚说明如下表 7- 1 所示。

表 7- 1TFT LCD 模块引脚说明表

序号	名称	说明
1	CS	LCD 片选信号（低电平有效）
2	RS	命令/数据控制信号（0：命令；1：数据）
3	WR	写使能信号（低电平有效）
4	RD	读使能信号（低电平有效）
5	RST	复位信号（低电平信号）
6~21	D0~D15	双向数据总线
22,26,27	GND	地线
23	BL_CTR	背光控制引脚（高电平点亮背光，低电平关闭）
24,25	VCC3.3	主电源供电引脚（3.3V）
28	VCC5	背光供电引脚（5V）
29	MISO	NC，电容触摸屏未用到
30	MOSI	电容触摸屏 IIC_SDA 信号（CT_SDA）

31	PEN	电容触摸屏中断信号（CT_INT）
32	BUSY	NC，电容触摸屏未用到
33	CS	电容触摸屏复位信号（CT_RST）
34	CLK	电容触摸屏 IIC_SCL 信号（CT_SCL）

7.1.2. NT35510 控制器简介

NT35510 支持 MDDI 接口、MIPI 接口，16/18/24 位 RGB 接口、8/16/18/24 位系统接口、串行外围接口（SPI）和 I2C 接口。可以选择性地更新指定的窗口区域，以便可以在显示静止图像的同时也可以显示运动的图像。

NT35510 自带 LCD GRAM(480* 864* 3 字节)，并且最高支持 24 位颜色深度(1600 万色)，一般使用 16 位颜色深度(65K 色)，即 RGB565 格式，这样，在 16 位模式下，可以达到最快的速度。此时 NT35510 的低 16 位数据总线(高 8 位没有用到)与 MCU 的 16 位数据线以及 24 位 LCD GRAM 的对应关系如下表 7- 2 所示：

表 7- 2 NT35510、MCU 16 位数据线与 24 位 LCD GRAM 的对应关系表

35510 总线	D15	D14	D13	D12	D11				D10	D9	D8	D7	D6	D5			D4	D3	D2	D1	D0			
MCU 数据	D15	D14	D13	D12	D11				D10	D9	D8	D7	D6	D5			D4	D3	D2	D1	D0			
LCD GRAM	R4	R3	R2	R1	R0	R4	R3	R2	G5	G4	G3	G2	G1	G0	G5	G4	B4	B3	B2	B1	B0	B4	B3	B2

从上表可以看出，NT35510 的 24 位 GRAM 与 16 位 RGB565 的对应关系，其实就是分别将高位的 R、G、B 数据，搬运到低位做填充，“凑成”24 位，再显示。MCU 的 16 位数据中，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。NT35510 的指令是 16 位宽，数据除了 GRAM 读写的时候是 16 位宽，其他都是 8 位宽的(高 8 位无效)，这个和 ILI9320 等驱动器不一样，必须加以注意。

7.1.3. 常用指令说明

7.1.3.1. 读 ID 指令

NT35510 读取 ID 的指令一共有三个字节，分别是 0xDA00、0xDB00 和 0xDC00，依次读取的模块的制造商 ID、模块驱动版本 ID 和模块 ID。每个指令输出一个参数，每个 ID 以 8 位数据的形式输出，高 8 位固定为 0，不过这里输出的 ID，并不是像 5510 这样的字样，仅有指令 0xDB00 会输出 ID：0x80，其他两个指令读到的 ID 都是 0。将这 3 个指令的输出组合在一起，可以得到 NT35510 的 ID 为：0x8000。对于指令描述如下表 7- 3 所示。

表 7- 3 读 ID 指令描述表

顺序	控制			各位描述									HEX
	RS	RD	WR	D15	D14	D13	D12	D11	D10	D9	D8	D7~D0	
指令 1	0	1	上升沿	1	1	0	1	1	0	1	0	00H	DA00H
参数 1	1	上升沿	1	0	0	0	0	0	0	0	0	00H	00H
指令 2	0	1	上升沿	1	1	0	1	1	0	1	1	00H	DB00H
参数 2	1	上升沿	1	0	0	0	0	0	0	0	0	80H	80H
指令 3	0	1	上升沿	1	1	0	1	1	1	0	0	00H	DC00H
参数 3	1	上升沿	1	0	0	0	0	0	0	0	0	00H	00H

7.1.3.2. 存储器访问指令

存储器访问指令 0x3600，可以控制 NT35510 存储器的读写方向，简单的说，就是连续写 GRAM 的时候，可以控制 GRAM 指针的增长方向，从而控制显示方式（读 GRAM 也是一样）。除了读写存储器的方向，该寄存器也可以控制 GRAM 扫描刷新的方向。返回的参数含义如下表 7- 4 所示。

表 7- 4 存储器访问指令返回参数说明表

参数名称	参数含义
MY	这 3 位控制接口到内存的写入/读取方向。图案更改后立即显示。
MX	
MV	
ML	控制 TFT LCD 垂直刷新方向
RGB	控制 R,G,B 的排列顺序，0（RGB）/1（BGR）
MH	水平刷新方向
RSMX	左右反转图像（高电平有效）
RSMY	上下反转图像（高电平有效）

对于 MY、MX 和 MV 控制的写入/读取方向的说明如下表 7- 5 所示。

表 7- 5 MY、MX、 MV 设置与 LCD 扫描方向关系表

控制位			效果
MY	MX	MV	LCD 扫描方向（GRAM 自增方式）
0	0	0	从左到右，从上到下
1	0	0	从左到右，从下到上
0	1	0	从右到左，从上到下
1	1	0	从右到左，从下到上
0	0	1	从上到下，从左到右
0	1	1	从上到下，从右到左
1	0	1	从下到上，从左到右
1	1	1	从下到上，从右到左

7.1.3.3. 列地址设置

列地址设置指令 0x2A00~0x2A03，4.3 寸 LCD 的分辨率为 480*800，因此根据 NT35510 给出的 X，Y 轴坐标的限制，X，Y 坐标取值范围如下所示。

MV=“0”：Parameter range 0 ≤ XS[15:0] ≤ XE[15:0] ≤ 479(01DFh)

MV=“1”：Parameter range 0 ≤ XS[15:0] ≤ XE[15:0] ≤ 799(031Fh)

对于指令的描述如下表 7- 6 所示。

表 7- 6 列地址指令描述表

顺序	控制			各位描述									HEX
	RS	RD	WR	D15~D8	D7	D6	D5	D4	D3	D2	D1	D0	
指令 1	0	1	上升沿	2AH	0	0	0	0	0	0	0	0	2A00H
参数 1	1	1	上升沿	00H	SC15	SC14	SC13	SC12	SC11	SC10	SC9	SC8	SC[15:8]
指令 2	0	1	上升沿	2AH	0	0	0	0	0	0	0	1	2A01H
参数 2	1	1	上升沿	00H	SC7	SC6	SC5	SC4	SC3	SC2	SC1	SC0	SC[7:0]
指令 3	0	1	上升沿	2AH	0	0	0	0	0	0	1	0	2A02H
参数 3	1	1	上升沿	00H	EC15	EC14	EC13	EC12	EC11	EC10	EC9	EC8	EC[15:8]
指令 4	0	1	上升沿	2AH	0	0	0	0	0	0	1	1	2A03H
参数 4	1	1	上升沿	00H	EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0	EC[7:0]

上表中的 SC 和 EC 指的是列地址的起始值和结束值，SC 必须小于等于 EC，且 0≤SC/EC≤479。一般在设置 X 坐标的时候，我们只需要 0x2A00 和 0x2A01 两条指令即可，也就是设置 SC 即可，因为如果 EC 没有变化，我们只需要设置一次即可（在初始化的时候进行设置），从而提高速度。

7.1.3.4. 行地址设置

行地址设置指令 0x2B00~0x2B03，对于指令的描述如下表 7- 7 所示。

表 7- 7 行地址指令描述表

顺序	控制			各位描述									HEX
	RS	RD	WR	D15~D8	D7	D6	D5	D4	D3	D2	D1	D0	
指令 1	0	1	上升沿	2AH	0	0	0	0	0	0	0	0	2B00H
参数 1	1	1	上升沿	00H	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SP[15:8]
指令 2	0	1	上升沿	2AH	0	0	0	0	0	0	0	1	2B01H
参数 2	1	1	上升沿	00H	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SP[7:0]
指令 3	0	1	上升沿	2AH	0	0	0	0	0	0	1	0	2B02H
参数 3	1	1	上升沿	00H	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP[15:8]
指令 4	0	1	上升沿	2AH	0	0	0	0	0	0	1	1	2B03H
参数 4	1	1	上升沿	00H	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0	EP[7:0]

上表中的 SP 和 EP 指的是页地址的起始值和结束值，SP 必须小于等于 EP，且 0≤SP/EP≤799。一般在设置 Y 坐标的时候，我们只需要 0X2B00 和 0X2B01 两条指令即可，也就是设置 SP 即可，因为如果 EP 没有变化，我们只需要设置一次即可（在初始化的时候设置），从而提高速度。

7.1.3.5. 指定坐标位置填充颜色

在指定坐标位置填充颜色指令 0x2C00，当我们输入指令 0X2C00 之后，我们可以肆无忌惮地输入颜色数据，GRAM 会按照我们既定好的扫描顺序对颜色点进行填充。当达到（Xmax,Ymax）时，GRAM 会回到（X=0,Y=0）重新进行自增并进行颜色填充操作。配置的步骤如下所示：

1. 先配置颜色自增时的 X 轴的上下界；
2. 配置颜色自增时的 Y 轴上下界；
3. 当不断写入 RG565 的数据，GRAM 会按照我们配置的既定自增方式进行颜色填充。其中，16 位颜色数据=5 位 R+6 位 G+5 位 B。

7.1.3.6. GRAM 读取颜色点信息指令

从 GRAM 中读取颜色点信息的指令 0x2E00，除了写 GRAM 的颜色数据时使用的是 16 位有效数据，其他全为 8 位有效数据（虽然是 16 位数据，但是高 8 位无效）。因此，我们需要将读取出来的 8 位有效数据进行整理，最后得到 RGB565 的 16 位数据。

NT35510 是 24 位的，即 RGB=8+8+8，但是存储在帧存储器中的 D[23:0]数据不可以一下子送出来，只能半字半字的从 16 位并行数据线中传输出来，如下表 7- 8 所示。

表 7- 8 读取颜色点信息指令描述表

顺序	控制			各位描述												HEX
	RS	RD	WR	D15~D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
指令	0	1	上升沿	2EH				0	0	0	0	0	0	0	0	2E00
参数 1	1	上升沿	1	XX												dummy
参数 2	1	上升沿	1	R1[4:0]		XX		G1[5:0]						XX		R1G1
参数 3	1	上升沿	1	B1[4:0]		XX		R2[4:0]					XX			B1R2
参数 4	1	上升沿	1	G2[5:0]			XX	B2[4:0]					XX			G2B2
参数 5	1	上升沿	1	R3[4:0]		XX		G3[5:0]						XX		R3G3
参数 N	1	上升沿	1	按以上规律输出												

如上表所示，NT35510 在收到指令之后，第一次输出的是 dummy 数据，也就是无效的数据，第二次开始，读取到的才是有效的 GRAM 数据（从坐标：SC，SP 开始），输出规律为：每个颜色分量占 8 个位，一次数据 2 个颜色分量。比如：第一次输出的 R1G1，随后的规律为：B1R2→G2B2→R3G3→B3R4→G4R4.....以此类推。如果我们只需要读取一个点的颜色值，那么只需要接收到参数 3 即可，如果需要连续读取（利用 GRAM 地址自增，方法同上），那么就按照上诉规律去接收颜色数据。

7.2. 实验介绍

本章实验将在外部 SRAM 实验的基础之上，将之前连接 SRAM 的接口连接至 TFT LCD 屏的接口，并将 SRAM 地址控制信号的 SRAM_ADDR[0]作为 TFT LCD 屏的 RS 信号，最终点亮 TFT LCD，并实现 ASCII 字符和彩色显示等功能，并在串口打印 LCD 控制器 ID。

7.3. 建立 HQFPGA 工程

将外部 SRAM 实验的工程复制至存放本次实验的文件夹之下，并且对其进行修改，修改方式参考实验二的 2.3 节中的内容。

7.4. FPGA 侧代码修改

7.4.1. 修改顶层文件

点击设计管理，进入设计管理器，找到顶层文件进行修改，修改步骤如下所示：

1. 定义 TFT LCD 屏的端口

在 7.1.1 节中的表 7- 1 中列出了 TFT LCD 模块的端口信号，在顶层文件中定义需要使用到的端口信号（本次实验不包含触摸）如下所示：

```
output wire        lcd_bl,     //背光信号
output wire        lcd_wr_n,   //写使能
output wire        lcd_rd_n,   //读使能
inout  wire [15:0] lcd_data,   //数据信号线
output wire        lcd_cs_n,   //片选信号
output wire        lcd_rs,     //RS 为 1 表示写数据，为 0 表示写命令
output wire        lcd_rst     //复位信号
```

2. 设置背光和复位信号

在顶层模块文件中点亮背光信号，并将 LCD 屏的复位信号连接至整个系统的复位信号，如下所示。

```
assign lcd_bl = 1'b1;//设置背光信号为高电平
assign lcd_rst = MTXRSTN; //设置 TFT 的 RST 引脚
```

3. TFT LCD 控制信号的连接

在之前我们也说过，LCD 的操作时序和 SRAM 是类似的，唯一不同的就是 TFT LCD 只有 RS 信号，没有地址信号，在这里，我们可以通过 SRAM_ADDR[0] 来控制 RS 信号，TFT LCD 其他的控制信号则是和 SRAM 对应的信号相连接。

最终得到的顶层模块代码如下所示：

```
module TFT_LCD(
  input    wire                MAIN_CLK,        //CM3 时钟
  output   wire                CLKOUT,
  output   wire                PCLK,
  output   wire                MTX_CLK,
  input    wire                MTXRSTN,       //CM3 复位

  input    wire                swdclk,        //debug 接口
  inout    wire                swddio,        //debug 接口
  inout[31:0]                  GPIO,         //32 位通用 GPIO
```

```
  output wire        lcd_bl,     //背光信号
  output wire        lcd_wr_n,   //写使能
  output wire        lcd_rd_n,   //读使能
  inout  wire [15:0] lcd_data,   //数据信号线
  output wire        lcd_cs_n,   //片选信号
  output wire        lcd_rs,     //RS 为 1 表示写数据，为 0 表示写命令
  output wire        lcd_rst     //复位信号
);
wire CLK100M;
wire PLL_Locked;
wire [18:0]SRAM_ADDR;
assign lcd_bl = 1'b1;//设置背光信号为高电平
assign lcd_rst = MTXRSTN; //设置 TFT 的 RST 引脚
assign lcd_rs = SRAM_ADDR[0]; //LCD 的 RS 信号对应 SRAM 地址信号的最低位
cm3_system cm3_system(
  .MAIN_CLK(MAIN_CLK),
  .CLKOUT(CLKOUT),
  .PCLK(PCLK),
  .MTX_CLK(MTX_CLK),
  .MTXRSTN(MTXRSTN),
  .PLL_OUT(CLK100M),
  .swdclk(swdclk),
  .swddio(swddio),
  .GPIO(GPIO),
  .SRAM_ADDR(SRAM_ADDR),
  .SRAM_DQ(lcd_data),
  .SRAM_nWE(lcd_wr_n),
  .SRAM_nOE(lcd_rd_n),
  .SRAM_nCE(lcd_cs_n),
  .SRAM_nLB(),
  .SRAM_nUB()
);
PLL PLL(
  .CLKI(MAIN_CLK),
  .CLKOP(CLK100M),
  .LOCK(PLL_Locked)
);
endmodule
```

7.4.2. LCD 控制器读写时序说明

LCD 控制器的读写时序和 SRAM 的虽然差不多，但是在移植过来使用的时候，需要注意的是其读写时序的差别。在 7.1.2 节中介绍过实验用到的 TFT LCD

屏的控制芯片为 NT35510，通过查阅对应的器件手册得到对应的读写时序如下图 7. 3 所示， 表 7- 9 对图中的参数进行了说明。

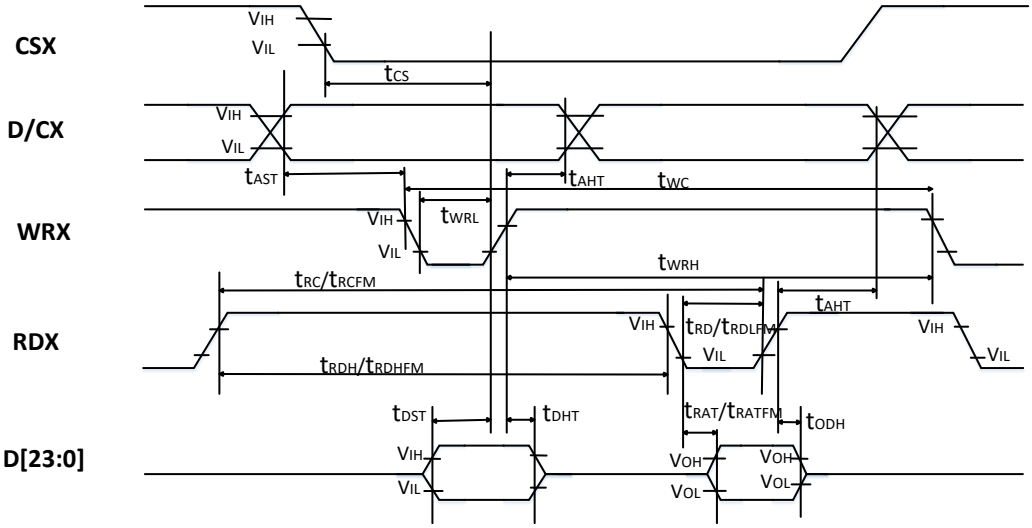

图 7. 3 NT35510 时序参数图

表 7- 9 时序参数说明表

Signal	Symbol	Parameter	MIN	MAX	Unit	Description
WRX	t_{WC}	Write cycle	33	-	ns	
	t_{WRH}	Control pulse “H” duration	15	-	ns	
	t_{WRL}	Control pulse “L” duration	15	-	ns	
RDX(ID)	t_{RC}	Read cycle (ID)	160	-	ns	When read ID data
	t_{RDH}	Control pulse “H” duration (ID)	90	-	ns	
	t_{RDL}	Control pulse “L” duration (ID)	45	-	ns	
RDX(FM)	t_{RCFM}	Read cycle (FM)	400	-	ns	When read from frame memory
	t_{RDHFM}	Control pulse “H” duration (FM)	250	-	ns	
	t_{RDLFM}	Control pulse “L” duration (FM)	150	-	ns	
D/CX	t_{AST}	Address setup time (Write)	0	-	ns	
		Address setup time (Read)	10	-	ns	
	t_{AHT}	Address hole time	2	-	ns	
D[17:0]	t_{DST}	Data setup time	15	-	ns	
	t_{DHT}	Data hold time	10	-	ns	
	t_{RAT}	Read access time (ID)	-	40	ns	
	t_{RATFM}	Read access time (FM)	-	150	ns	
	t_{ODH}	Output disable time	5	-	ns	

在上表中我们需要特别注意的就是读写过程中低电平保持时间：在写操作时，其低电平保持时间 t_{WRL} 的最小值为 15ns；读 ID 的的时候，低电平保持时间最小为 45ns；读寄存器时，低电平保持时间最小为 150ns。所以在整个读的过程中，只要低电平保持时间大于 150ns，那么就能成功读取 ID 和寄存器中的值。

7.4.3. 修改读写周期

点击 cm3_system，找到 cmsdk_ahb_to_extmem16 模块例化部分，找到配置

信号部分，修改读写周期，本次实验 PLL 使用的是 100M 的时钟，周期就是 10ns，那么读周期就需要至少 2 个周期，写周期至少 15 个周期，设置如下所示：

```
.CFGREADCYCLE      (15),   // Read cycle
.CFGWRITECYCLE     (2),  // Write cycle
```

需要注意的是，原本默认 0，读写为 1 周期的时间，这里分别设置为 15 和 2，那么实际写数据的时间为 3 个周期，也就是 30ns，读数据的时间为 16 个周期，也就是 160ns，满足时序要求。

7.5. 物理管脚约束

在设计管理器界面，点击设计文件中的“ac208.upc”文件，添加对于 TFT LCD 屏的引脚分配，如下所示：

```
phycst.pin.set {lcd_bl} G5
phycst.pin.set {lcd_cs_n} L12
phycst.pin.set {lcd_data[15]} G3
phycst.pin.set {lcd_data[14]} H3
phycst.pin.set {lcd_data[13]} G10
phycst.pin.set {lcd_data[12]} E6
phycst.pin.set {lcd_data[11]} E7
phycst.pin.set {lcd_data[10]} J4
phycst.pin.set {lcd_data[9]} F6
phycst.pin.set {lcd_data[8]} K3
phycst.pin.set {lcd_data[7]} J5
phycst.pin.set {lcd_data[6]} K5
phycst.pin.set {lcd_data[5]} J6
phycst.pin.set {lcd_data[4]} R11
phycst.pin.set {lcd_data[3]} N10
phycst.pin.set {lcd_data[2]} P11
phycst.pin.set {lcd_data[1]} K6
phycst.pin.set {lcd_data[0]} R12
phycst.pin.set {lcd_rd_n} K11
phycst.pin.set {lcd_rs} L13
phycst.pin.set {lcd_rst} K10
phycst.pin.set {lcd_wr_n} P12
```

7.6. 编译设计

保存完修改后的顶层文件和物理约束文件之后，点击全部运行，生成本次实验需要的 FPGA 侧的 bin 文件，操作如图 7. 4 所示。

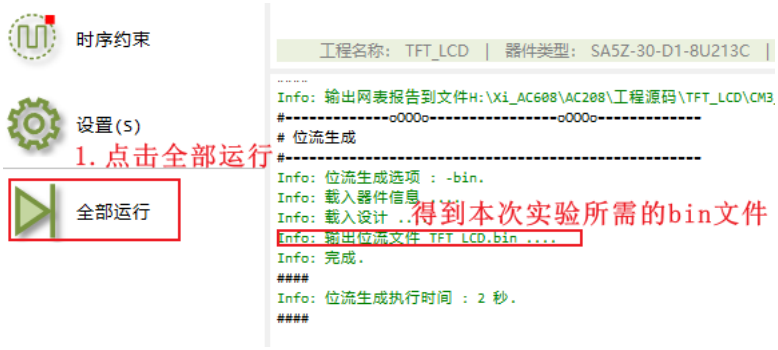

图 7. 4 编译运行整个文件

7.7. 建立 MDK 工程

将已有的 MDK 工程复制至本次工程的文件下，并对其进行修改，操作方式参考实验二中 2.8 节的内容。

7.8. 软件设计

在 HARDWARE 文件夹之下，添加本次实验需要的库文件“TFT_LCD”,添加方式参考 3.6.1 节中的内容。还需要添加在 TFT_LCD 库中使用到的字体大小设置文件“font.h”。

“TFT_LCD.c”中的代码比较多，这里就不一一介绍，仅针对本次实验使用到的的函数进行讲解，完整的代码请自行查看对应的文件。

在前面的顶层文件中，我们已经将 TFT LCD 的 RS 信号连接至 SRAM_ADDR[0]信号上，那么在控制 RS 信号的时候，其实也就是控制 AHB 总线扩展端口 0 的地址最低位（AHB Expansion0：0xA000_0000）。于是我们就可以定义两个地址，一个用来传命令（RS 信号为 0）,一个用来传数据（RS 信号为 1），在“TFT_LCD.h”文件中定义如下：

```
#define LCD_CMD_BASE  ((uint32_t)0xA0000000)
#define LCD_DATA_BASE ((uint32_t)0xA0000002)
```

在上述定义中，我们可以看到，当需要 RS 信号为 1 的时候，并不是对应的使 AHB Expansion0 地址的第[0]位为 1，而是使 AHB Expansion0 地址的第[1]位为 1，这是因为 SRAM 芯片的地址是以字节为单位的，SRAM 的地址 A0 其实连接的是 AHB 地址信号的 A1，如下所示：

```
wire [19:0]SRAM_ADDR_Byte;  //以字节为单位的 SRAM 地址
assign SRAM_ADDR = SRAM_ADDR_Byte[19:1];     //19 位 SRAM，地址去掉最低位
```

通过上诉的定义我们得到，控制 TFT LCD 控制器写命令就是向“LCD_CMD_BASE”地址中写入值，写数据就是向“LCD_DATA_BASE”地址中写入值，读数据就是读取“LCD_DATA_BASE”地址上的值。定义如下所示：

```
//写命令函数
#define LCD_WR_CMD(x)  *(__IO uint16_t *)LCD_CMD_BASE  = x
//写数据函数
#define LCD_WR_DATA(x)  *(__IO uint16_t *)LCD_DATA_BASE = x
//读数据函数
#define LCD_RD_DATA()  *(__IO uint16_t *)LCD_DATA_BASE
```

TFT LCD 其余的控制信号线 CS、WR、RD 都是由 AHB 总线进行控制的，不用我们手动控制了，下面将介绍一下在“TFT_LCD.c”中本次实验使用到的几个库函数。

7.8.1. TFT LCD 库

7.8.1.1. TFTLCD_Init

TFT LCD 初始化函数，函数主要内容：首先通过 LCD_WR_CMD 函数读取 NT35510 的 ID 号，然后通过 LCD_RD_DATA 函数看返回的值是否为 0x8000，如果是 0x8000 的话，则代表读取的 ID 正确，ID 读取成功之后，则通过 LCD_WriteReg 函数进行寄存器的配置。LCD_WriteReg 函数主要内容：先通过写命令函数写入需要操作的寄存器，然后通过写数据函数写需要写入到寄存器中的数据，代码如下所示：

```
void LCD_WriteReg(uint16_t LCD_Reg,uint16_t LCD_RegValue)
{
    LCD_WR_CMD(LCD_Reg);         // 写入的寄存器
    LCD_WR_DATA(LCD_RegValue);   // 写入的数据
}
```

通过 LCD_Display_Dir 函数设置 LCD 显示方式为横屏，LCD_Display_Dir 函数内容：设置 LCD 的宽度，高度，开始写 gram 指令，设置 X 坐标和 Y 坐标指令。代码如下所示：

```
void LCD_Display_Dir(uint8_t dir)
{
    lcd_dir=dir; //屏幕方向
    if(dir==0){ //竖屏
        lcd_width=240;
        lcd_height=320;
        lcd_wramcmd=0X2C00;
        lcd_setxcmd=0X2A00;
        lcd_setycmd=0X2B00;
        lcd_width=480;
        lcd_height=800;
    }else{  //横屏
        lcd_width=320;
```

```
        lcd_height=240;
        lcd_wramcmd=0X2C00;
        lcd_setxcmd=0X2A00;
        lcd_setycmd=0X2B00;
        lcd_width=800;
        lcd_height=480;
    }
    LCD_Scan_Dir(DFT_SCAN_DIR);     //默认扫描方向
}
```

可以看到在 LCD_Display_Dir 函数中，通过 LCD_Scan_Dir 函数设置了默认的扫描方向为 DFT_SCAN_DIR（从左到右，从上到下）。LCD_Scan_Dir 函数代码请自行查看 TFT LCD 库中的内容。

最后 LCD_Clear 函数将屏幕颜色设置为白色。LCD_Clear 函数代码如下所示：

```
void LCD_Clear(uint32_t color)
{
    uint32_t index=0;
    uint32_t totalpoint=lcd_width;
    totalpoint*=lcd_height;      //得到总点数
    LCD_SetCursor(0x00,0x0000);     //设置光标位置
    LCD_WriteRAM_Prepare();         //开始写入 GRAM
    for(index=0;index<totalpoint;index++) {
        LCD_WR_DATA(color);
    }
}
```

从上述代码可以看出，该函数只有一个输入变量：color，就是选择需要设置的清屏的颜色，对于颜色的定义都在“TFT_LCD.h”文件中，这里就不进行列举。函数主要进行的操作：通过 LCD 的屏两个参数，宽度（lcd_width）和高度（lcd_height）相乘得到需要写入颜色的总点数，然后通过 LCD_SetCursor 函数设置光标的位置，LCD_WriteRAM_Prepare 函数开始写入 GRAM，最后通过 LCD_WR_DATA 函数依次相对应的点写入颜色值。

LCD_Clear 函数中使用到了比较重要的函数：

1. LCD_SetCursor 函数

设置光标的位置，该函数实现将 LCD 当前操作的点设置到指定坐标（x,y），通过该函数我们就可以在 LCD 屏上的任意一个位置进行作图。函数代码如下所示：

```
void LCD_SetCursor(uint16_t Xpos, uint16_t Ypos)
{
    LCD_WR_CMD(lcd_setxcmd);LCD_WR_DATA(Xpos>>8);
```

```
    LCD_WR_CMD(lcd_setxcmd+1);LCD_WR_DATA(Xpos&0XFF);
    LCD_WR_CMD(lcd_setycmd);LCD_WR_DATA(Ypos>>8);
    LCD_WR_CMD(lcd_setycmd+1);LCD_WR_DATA(Ypos&0XFF);
}
```

上述代码中通过 LCD_WR_CMD 函数写入设置 X 坐标指令（0X2A00）和 Y 坐标指令（0X2B00），然后通过 LCD_WR_DATA 函数写入对应的坐标位置，lcd_setxcmd 的值是在 LCD_Display_Dir 这一函数中进行设置的。

2. LCD_WriteRAM_Prepare 函数

函数主要就是通过 LCD_WR_CMD 函数写入写 GRAM 指令 lcd_wramcmd（0X2C00），lcd_wramcmd 的值同样是在 LCD_Display_Dir 这一函数中进行设置的，函数代码如下所示：

```
void LCD_WriteRAM_Prepare(void)
{
    LCD_WR_CMD(lcd_wramcmd);
}
```

需要注意的是，凡事用到 TFT LCD 屏都需要在主函数中先进行初始化。调用方式如下所示：

```
TFTLCD_Init();
```

7.8.1.2. LCD_Set_Window

设置窗口，并自动设置画点坐标到窗口的左上角（sx，sy），窗口的宽度和高度分别为 width 和 height，得到的窗体大小：width*height。首先需要得到窗口的两个极限值：x 的最大值：sx+width-1；y 的最大值：sy+height-1，然后分别写入命令，得到窗口 2 个顶点的坐标。函数实现的具体代码如下：

```
void LCD_Set_Window(u16 sx,u16 sy,u16 width,u16 height)
{
    u16 twidth,theight;
    twidth=sx+width-1;
    theight=sy+height-1;
    LCD_WR_CMD(lcd_setxcmd);LCD_WR_DATA(sx>>8);
    LCD_WR_CMD(lcd_setxcmd+1);LCD_WR_DATA(sx&0XFF);
    LCD_WR_CMD(lcd_setxcmd+2);LCD_WR_DATA(twidth>>8);
    LCD_WR_CMD(lcd_setxcmd+3);LCD_WR_DATA(twidth&0XFF);
    LCD_WR_CMD(lcd_setycmd);LCD_WR_DATA(sy>>8);
    LCD_WR_CMD(lcd_setycmd+1);LCD_WR_DATA(sy&0XFF);
    LCD_WR_CMD(lcd_setycmd+2);LCD_WR_DATA(theight>>8);
    LCD_WR_CMD(lcd_setycmd+3);LCD_WR_DATA(theight&0XFF);
}
```

函数的使用方式举例如下：

```
LCD_Set_Window(10,10,100,50);//起始坐标为(10,10)，宽度为 100，长度为 50
```

7.8.1.3. LCD_DisplayPic

显示图片，函数的变量说明如下表 7- 10 所示。

表 7- 10LCD_DisplayPic 函数变量说明表

变量名称	变量意义
x	光标的 x 轴坐标
y	光标的 y 轴坐标
size	需要写入的图片的大小
*pic	图片像素点的起始位置

函数的实现方式就是连续写像素点，其实现代码如下所示。

```
void LCD_DisplayPic(uint16_t x,uint16_t y,uint32_t size,const uint8_t *pic)
{
    uint32_t i;
    LCD_SetCursor(x, y);
    LCD_WriteRAM_Prepare();     //开始写入 GRAM
    for(i=0; i < size; i++)
        LCD_WR_DATA(pic[i*2]<<8 | pic[i*2+1]);
}
```

函数的使用方式举例如下所示：

```
LCD_DisplayPic(18,28,16008,gImage_xiaomeige3);//显示图片
```

上面所示的“gImage_xiaomeige3”是一个.h 文件，可以通过 img2lcd 软件生成，使用的时候在“main.c”这个文件中添加这个头文件就可以了，img2lcd 软件的使用方法在本章的思考与总结中进行补充。

7.8.1.4. LCD_ShowString

显示字符串，函数的变量说明如下表 7- 11 所示。

表 7- 11 LCD_ShowString 函数变量说明表

变量名称	变量意义
x	起点横坐标
y	起点纵坐标
width	显示区域的宽度
height	显示区域的高度
size	字体的大小
*p	字符串起始地址
mode	叠加方式(1)还是非叠加方式(0)

函数主要是先判断是否是非法字符，如果不是非法字符的话，就通过 LCD_ShowChar 函数在指定位置显示字符串，如果是非法字符的话，退出。函数的代码如下所示：

```
void LCD_ShowString(uint16_t x,uint16_t y,uint16_t width,uint16_t
height,uint8_t size,uint8_t mode,uint8_t *p)
{
    uint8_t x0=x;
    width+=x;
    height+=y;
    while((*p<='~')&&(*p>=' ')) {   //判断是不是非法字符!
        if(x>=width) {
            x=x0;
            y+=size;
        }
        if(y>=height)
            break;//退出
        LCD_ShowChar(x,y,*p,size,mode);
        x+=size/2;
        p++;
    }
}
```

在函数中使用到了一个函数：LCD_ShowChar 显示一个字符，对于该函数的变量说明如下表 7- 12 所示：

表 7- 12 LCD_ShowChar 变量说明表

变量名称	变量意义
x	需要显示字符的 x 轴坐标
y	需要显示字符的 y 轴坐标
num	需要显示的字符内容
size	字体的大小，12/16
mode	叠加方式，1 代表叠加，0 代表不叠加

对于上述的叠加方式的选择，就是显示字符的空白区域是否与背景颜色进行叠加，1 代表的就是与背景颜色一致。对于字体的选择，其具体的设置在“font.h”中，函数的具体内容可以自行查看库函数中的代码。

LCD_ShowString 函数使用举例如下：

```
LCD_ShowString(150, 80, 180, 20, 24, 1, (uint8_t *)"TFT LCD TEST!!!");
```

7.8.2. 添加用户代码

实验功能：实现 ASCII 字符和彩色显示等功能，并在串口打印 LCD 控制器 ID。

实验代码：初始化串口和 TFT LCD 屏，使 TFT LCD 屏全屏显示蓝色，开窗显示图片，然后设置字体颜色为红色，在屏幕上显示“Welcome to use AC208”和“TFT LCD TEST!!!”。代码如下所示：

```
#include "CM3DS_rcc.h"
```

```
#include "CM3DS_gpio.h"
#include "CM3DS_MPS2.h"
#include "CM3DS_spi.h"
#include "TFT_LCD.h"
#include "uart.h"
#include "xiaomeige.h"
extern uint16_t lcd_id;
int main()
{
    Uart_Init(CM3DS_MPS2_UART0,115200);//初始化串口
    TFTLCD_Init();//初始化 LCD 屏
    printf("LCD ID:%x\r\n",lcd_id); //打印 ID
    LCD_Clear(LCD_BLUE); //全屏显示蓝色
    LCD_Set_Window(18,68,116,138);//开窗
    LCD_DisplayPic(18,68,16008,gImage_xiaomeige3);//画图
    POINT_COLOR = LCD_RED;//设置字体颜色为红色
    LCD_ShowString(150, 80, 230, 40, 24, 1, (uint8_t *)"Welcome to use AC208");
    LCD_ShowString(150, 110, 180, 40, 24, 1, (uint8_t *)"TFT LCD TEST!!!");
}
```

7.9. 板级验证

7.9.1. 实验所需硬件

(1) AC208 开发板
(2) FPGA 下载器：XIST USB Cable
(3) CM3 仿真器：DAP Link
(4) 电源线一根
(5) TFT LCD 屏一块

7.9.2. 硬件连接

JTAG 和 DAP Link 的硬件连接参考实验一。

在连接 TFT LCD 屏的时候，由于开发板显示接口有 2*18 共 36 个接口座，而 TFT LCD 屏只有 34 个接口，所以在连接的时候，请保持靠下插接的方式，空出主板上显示扩展接口的 1、2 两个脚，即对应的开发板后面显示的 GND 和 3V3 不要接。需要空出的接口如下图 7. 5 所示。

图 7. 5 TFT LCD 屏连接时需要空出的引脚

TFT LCD 屏正确的连接方式如下图 7. 6 所示。

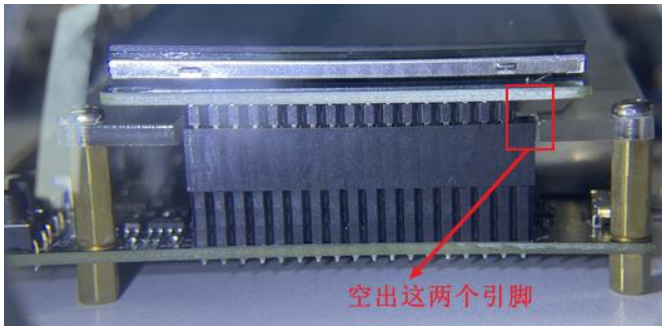

图 7. 6 TFT LCD 显示屏正确连接示意图

7.9.3. 下载文件至目标板

下载文件的方式参考实验一。

7.9.4. 功能演示

将程序下载之后，屏幕显示如下图 7. 7 所示。

图 7. 7 TFT LCD 屏显示图

串口显示 ID 如下图 7. 8 所示。

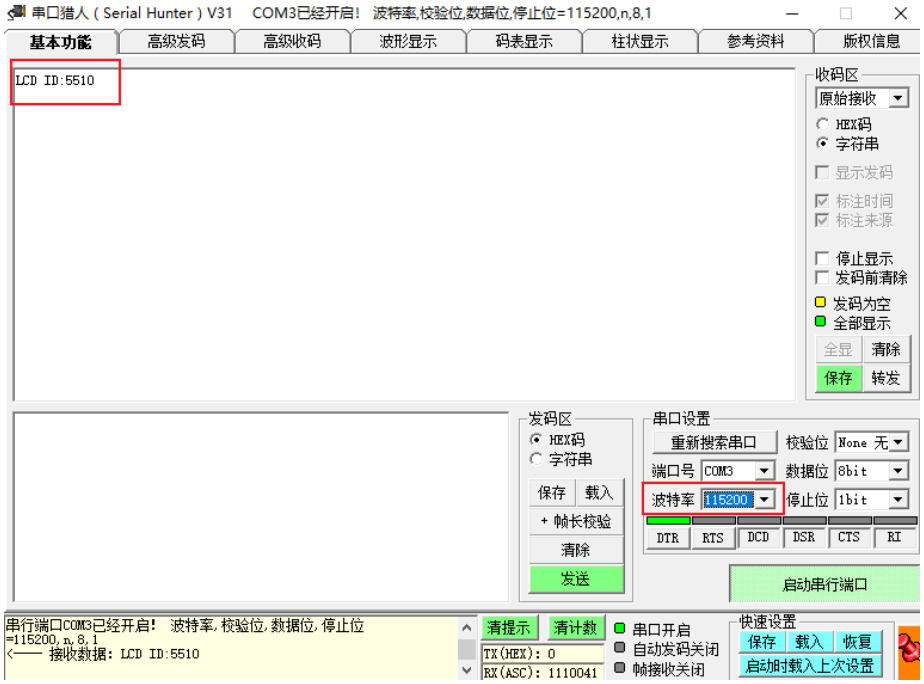

图 7. 8 串口显示 ID

7.10. 思考与总结

本次实验通过 AHB 总线成功驱动了 TFT LCD 屏。在实验中，将 SRAM 地址控制信号的 SRAM_ADDR[0]作为 TFT LCD 屏的 RS 信号，需要注意的是：当

需要 RS 信号为 1 的时候，并不是对应的使 AHB Expansion0 地址的第[0]位为 1，而是使 AHB Expansion0 地址的第[1]位为 1，这是因为 SRAM 芯片的地址是以字节为单位的，SRAM 的地址 A0 其实连接的是 AHB 地址信号的 A1。

在本次实验中需要补充一点：使用 img2lcd 软件将图片生成.h 文件方法。

打开 img2lcd 软件，然后点击左上方的“打开”按钮，选择一张图片，按照下图的方法进行设置，设置完成后点击保存（保存的时候文件加上.h 后缀），即可得到图片对应的.h 文件，具体操作方式如下图 7. 9 所示。

图 7. 9 img2lcd 软件的使用方法

在设置的时候需要注意以下几点：

1. 本次例程使用到的 LCD 的分辨率为 800*480，那么在设置的时候得到的实际输出图像的大小不能大于 800*480，实际输出图像的数据为何与自己设置的不一样，这是因为该软件输出的图像数据是基于原图片的分辨率进行等比例缩放的。

2. 以本次举例的图片为例，将输出的图像命名为“1.h”，得到的输出图像数据为一个数组，如下所示：

```
const unsigned char gImage_1[108540]
```

可以看到得到的数组的大小为“108540”，也就是实际输出图像的像素 270*201*2，这里乘以 2 是因为数组中的两个值代表一个像素值，如果需要将这张图片显示在 LCD 屏上，需要先开一个“270*201”的窗，然后通过 LCD_DisplayPic 画图，比如我们这里设置起始显示坐标为“0，0”，代码举例如下：

```
TFTLCD_Init();//初始化 LCD 屏
```

```
LCD_Set_Window(0,0,270,201);//开窗
LCD_DisplayPic(0,0,270*201,gImage_1);//画图
```