ATK-1.3’TFTLCD 液晶模块使用说明

本应用文档将教大家如何在 MINI STM32F103 开发板上使用 ATK-1.3’TFTLCD 液晶显示模块。

本文档分为如下几部分：

1， ATK-1.3’TFTLCD 模块简介

2， 硬件连接

3， 软件实现

1.ATK-1.3’TFTLCD 模块简介

ATK-1.3’ TFTLCD 是 ALIENTEK 推出的一款高性能 1.3 寸液晶显示模块。该模块分辨率高达 240*240，支持 16 位真彩色显示，采用 ST7789V2 驱动，该芯片自带 RAM，无需外加驱动器，单片机只需要使用 SPI 接口就可以轻易驱动该液晶屏幕。

1.1 模块引脚说明

ATK-1.3’TFTLCD 显示屏通过 2*4P 的排针（2.54mm 间距）同外部连接，模块可以与 ALIENTEK 的 STM32 开发板直接连接，我们也提供了相应的例程，用户可以在 ALIENTEK STM32 开发板上直接测试。ATK-1.3’ TFTLCD 外观如图 1.1.1 所示：

图 1.1.1 ATK-1.3’ TFTLCD 模块正面图

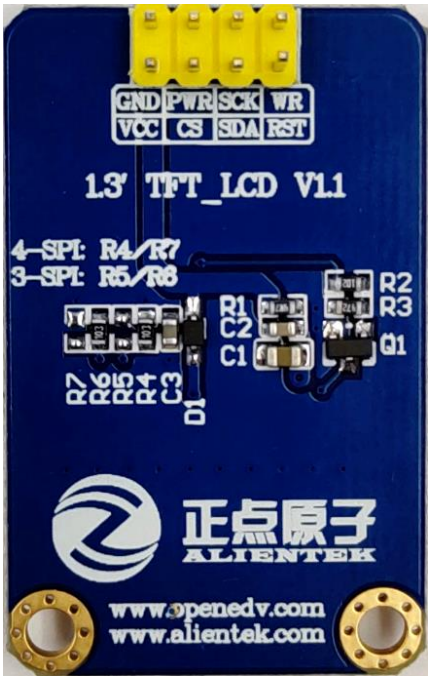

图 1.1.2 ATK-1.3’ TFTLCD 模块背面图

ATK-1.3’ TFTLCD 模块通过 8（2*4）个引脚同外部连接，对外接口原理图如图 1.1.3 所示：

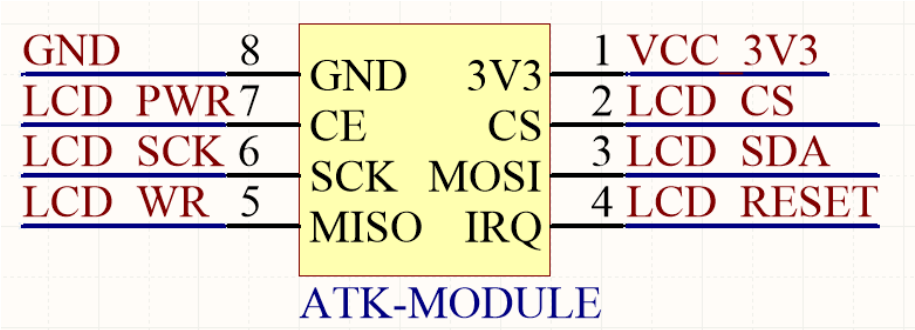

图 1.1.3 模块对外接口原理图

对应引脚功能详细描述如表 1.1.1 所示：

序号	名称	说明
1	VCC	LCD 供电电源引脚（3.3V）
2	CS	LCD 片选信号（低电平有效）
3	SDA	LCD 的 SDA 信号线
4	RESET	LCD 的复位信号(低电平有效)
5	WR(DC)	写命令/写数据信号（0：写命令；1：写数据）
6	SCK	LCD 的 SCK 时钟线
7	PWR	LCD 背光控制引脚（0：关闭；1：打开）
8	GND	电源地

表 1.1.1 ATK-1.3’ TFTLCD 模块引脚说明

特别注意：模块出厂默认选择四线 SPI，例程源码也是使用四线 SPI 进行通讯的，如果需要使用三线 SPI，请按照用户手册修改电路。

温馨提示，V1.1 版本 PCB 丝印存在一处错误：R6 和 R7 的丝印位置标注反了，实际出厂时默认焊接的四线模式并没有问题。

1.2 LCD 接口时序

ATK-1.3’TFTLCD 模块支持三线 SPI 和四线 SPI 通讯，出厂默认选择四线 SPI 通讯。通讯模式选择：

序号	名称	说明
1	三线 SPI	焊接：R5 和 R6
2	四线 SPI	焊接：R4 和 R7（出厂默认）

温馨提示，V1.1 版本 PCB 丝印存在一处错误：R6 和 R7 的丝印位置标注反了，实际出厂时默认焊接的四线模式并没有问题。

由于我们模块出厂默认使用四线 SPI，所以我们这里只讲解四线 SPI 的时序，三线 SPI 的时序请参考模块用户手册。

ATK-1.3’TFTLCD 在四线 SPI 通讯模式下，最少需要四根线就可以与 LCD 通讯：CS/SCL/SDA/WR(DC)，该模块出厂默认使用四线 SPI 通讯，也就是默认焊接 R7 和 R10 电阻。四线 SPI 接口时序如图 2.3.1.1 所示：

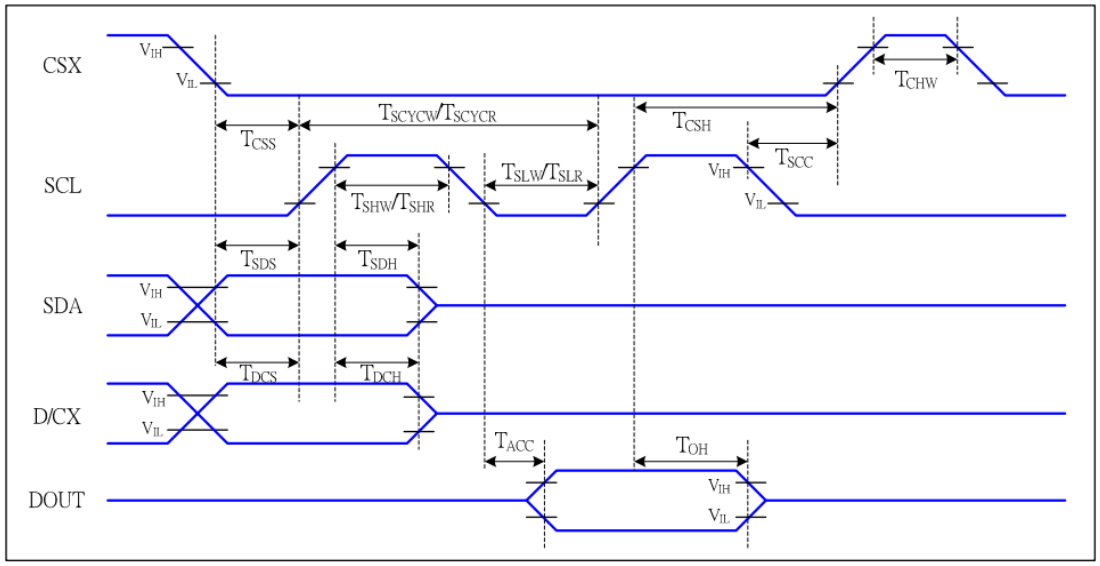

图 2.3.1.1　四线 SPI 接口时序图

图中各个时间参数见表 2.3.1.2 所示：

Signal	Symbol	Parameter	MIN	MAX	Unit	Description
CSX	T_{CSS}	Chip select setup time (write)	15		ns	
	T_{CSH}	Chip select hold time (write)	15		ns	
	T_{CSS}	Chip select setup time (read)	60		ns	
	T_{SCC}	Chip select hold time (read)	65		ns	
	T_{CHW}	Chip select “H” pulse width	40		ns	
SCL	T_{SCYCW}	Serial clock cycle (Write)	16		ns	-write command & data ram
	T_{SHW}	SCL “H” pulse width (Write)	7		ns	
	T_{SLW}	SCL “L” pulse width (Write)	7		ns	
	T_{SCYCR}	Serial clock cycle (Read)	150		ns	-read command & data ram
	T_{SHR}	SCL “H” pulse width (Read)	60		ns	
	T_{SLR}	SCL “L” pulse width (Read)	60		ns	
D/CX	T_{DCS}	D/CX setup time	10		ns	
	T_{DCH}	D/CX hold time	10		ns	
SDA (DIN)	T_{SDS}	Data setup time	7		ns	
	T_{SDH}	Data hold time	7		ns	
DOUT	T_{ACC}	Access time	10	50	ns	For maximum CL=30pF
	T_{OH}	Output disable time	15	50	ns	For minimum CL=8pF

表 2.3.1.2　　四线 SPI 时间参数

从表中可以看出，模块的写周期是非常快的，写周期为：16ns，而模块的读周期相对较慢，读周期为：150ns。

LCD 四线 SPI 的详细读写时序，请看 ST7789V2 数据手册第 56 页和 60 页。

1.3 LCD 驱动说明

ATK-1.3’TFTLCD 模块采用 ST7789V2 作为 LCD 驱动器，显示数据可以直接存储在 240*320*18 位片上的 RAM 中，它可以在没有外部操作时钟的情况下执行显示数据 RAM 读/写操作，以最小化功耗。该驱动芯片采用 SPI 接口与外部连接，需要使用的信号线如下：

CS：LCD 的片选信号线

SCK：SPI 的时钟信号线

SDA：SPI 的数据信号线

WR(DC)：命令/数据标志（0：写命令；1：写数据）

除了以上信号，我们一般还需要用到这个 2 个信号：RESET 和 PWR，其中：RST 是 LCD 的硬复位脚，低电平有效，用于复位 ST7789V2 芯片，实现液晶的复位，在每次初始化之前，我们强烈建议大家先执行硬复位，再做初始化。而 PWR 则是 LCD 的背光控制引脚，高电平有效，这个引脚自带了下拉电阻，所以如果这个引脚悬空，背光是不会亮的。必须接高电平背光才会亮，另外可以用 PWM 控制 PWR 脚，从而控制背光的亮度。

ST7789V2 自带 LCD RAM（240*320*3 字节），并且最高支持 18 位颜色深度（262K 色），不过我们一般使用 16 位颜色深度（65K 色），RGB565 格式，这样可以在 16 位模式下达到最快的速度。在 16 位模式下，ST7789V2 采用 RGB565 格式存储颜色数据，此时 MCU 的 16 位数据与 LCD RAM 的对应关系如图 2.4.1 所示：

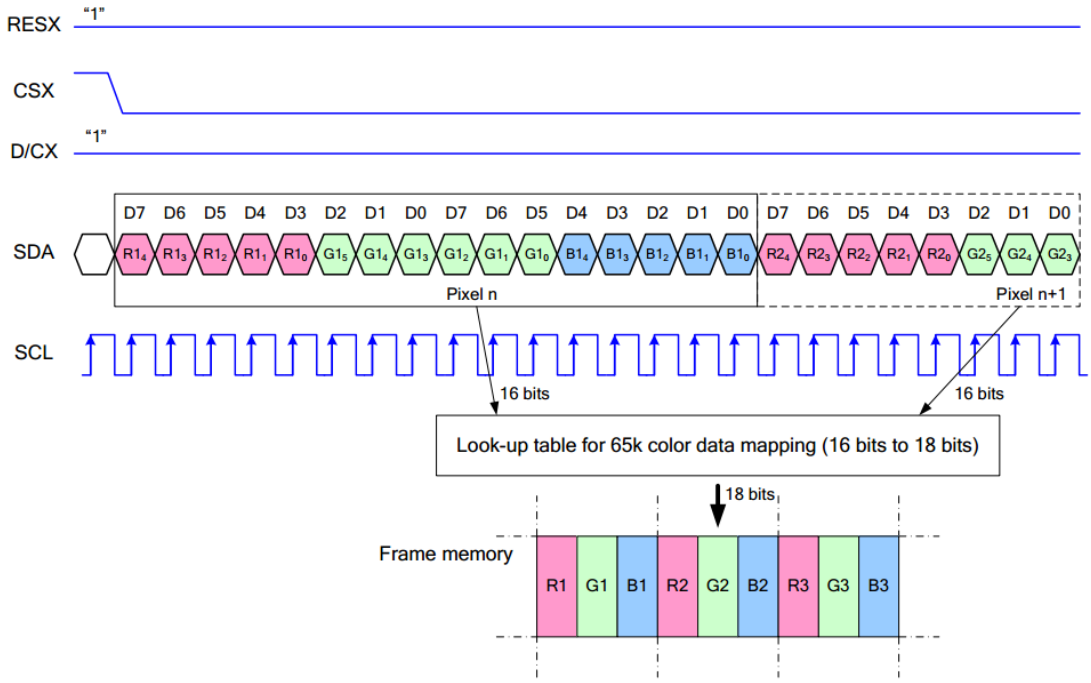

图 2.4.1 16 位数据与 LCD RAM 的对应关系

图示 MCU 的 16 位数据中，最低 5 位代表蓝色，中间 6 位位绿色，最高 5 位为红色。数值越大，表示该颜色越深。

接下来，就来介绍一下 ST7789V2 的几条重要指令，因为该芯片的命令有很多，我们这里就不全部介绍了，有兴趣的可以找数据手册看看，里面对命令有详细介绍。

首先来看一下指令：0x36，这是存储访问控制指令，可以控制 ST7789V2 存储器的读写方向，简单的来说，就是在连续写 LCD RAM 数据的时候，可以控制 RAM 指针的增长方向，从而控制显示方式（读操作也是一样）。该指令如图 2.4.2 所示：

36H	MADCTL (Memory Data Access Control)												
Inst / Para	D/CX	WRX	RDX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	HEX
MADCTL	0	↑	1	-	0	0	1	1	0	1	1	0	(36h)
parameter	1	↑	1	-	MY	MX	MV	ML	RGB	MH	-	-	

-This command defines read/ write scanning direction of frame memory.

Bit	NAME	DESCRIPTION
D7	MY	Page Address Order
D6	MX	Column Address Order
D5	MV	Page/Column Order
D4	ML	Line Address Order
D3	RGB	RGB/BGR Order
D2	MH	Display Data Latch Order

图 2.4.2 0x36 指令描述

从上图可以看出。0x36 指令下可以配置 6 个参数，这里我们主要关心：MY、MX 和 MV 这三位，通过这三个位的设置，我们可以控制整个 ST7789V2 的全部扫描方向。如表 2.4.1 所示：

控制位			效果
MY	MX	MV	LCD 扫描方向（RAM 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

表 2.4.1 MX、MY、MV 设置与 LCD 扫描方向关系表

这样，我们在使用 ST7789V2 显示内容的时候，就有很大灵活性了，比如显示 BMP 图片，BMP 解码数据，就是从图片的左下角开始，慢慢显示到右上角，如果设置 LCD 扫描方向为从左到右，从下到上，那么我 们只需要设置一次坐标，然后就不停的往 LCD 填充颜色数据即可，这样可以大大提高显示速度。

接下来看一下指令：0x2A，这是列地址设置指令，在从左到右，从上到下的扫描方式（默认）下面，该指令用于设置横坐标（x 坐标），该指令如图 2.4.3 所示：

2AH	CASET (Column Address Set)												
Inst / Para	D/CX	WRX	RDX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	HEX
CASET	0	↑	1	-	0	0	1	0	1	0	1	0	(2Ah)
1st parameter	1	↑	1	-	XS15	XS14	XS13	XS12	XS11	XS10	XS9	XS8	
2nd parameter	1	↑	1	-	XS7	XS6	XS5	XS4	XS3	XS2	XS1	XS0	
3rd parameter	1	↑	1	-	XE15	XE14	XE13	XE12	XE11	XE10	XE9	XE8	
4th parameter	1	↑	1	-	XE7	XE6	XE5	XE4	XE3	XE2	XE1	XE0	

图 2.4.3 0x2A 指令描述

这条指令用于设置 x 坐标，x 坐标有两个坐标值：XS 和 XE（XS 和 XE 都是 16 位的，由 2 个 8 位组成），即列地址的起始值和结束值，当“MV=0”时，0<XS<XE<239，当“MV=1”时，0<XS<XE<319。

与 0x2A 指令类似，指令：0x2B，是设置行地址的指令，在从左到右，从上到下的扫描方式（默认）下面，该指令用于设置纵坐标（y 坐标）。该指令如图 2.4.3 所示:

2BH	RASET (Row Address Set)												
Inst / Para	D/CX	WRX	RDX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	HEX
RASET	0	↑	1	-	0	0	1	0	1	0	1	1	(2Bh)
1st parameter	1	↑	1	-	YS15	YS14	YS13	YS12	YS11	YS10	YS9	YS8	
2nd parameter	1	↑	1	-	YS7	YS6	YS5	YS4	YS3	YS2	YS1	YS0	
3rd parameter	1	↑	1	-	YE15	YE14	YE13	YE12	YE11	YE10	YE9	YE8	
4th parameter	1	↑	1	-	YE7	YE6	YE5	YE4	YE3	YE2	YE1	YE0	

图 2.4.3 0x2B 指令描述

这条指令用于设置 y 坐标，y 坐标有两个坐标值：YS 和 YE（YS 和 YE 都是 16 位的，由 2 个 8 位组成），即页地址的起始值和结束值，当“MV=0”时，0<YS<YE<319，当“MV=1”时，0<YS<YE<239。

一般 TFTLCD 模块的使用流程如图 2.4.1 所示：

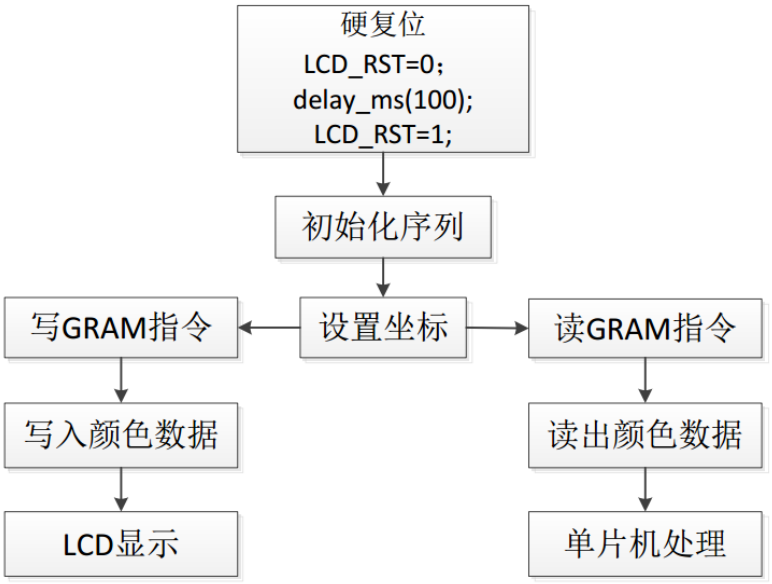

图 2.4.4 TFTLCD 使用流程

任何 LCD，使用流程都可以简单的用以上流程图表示。其中硬复位和初始化序列，只需要执行一次即可。而画点流程就是：坐标->写 GRAM 指令->写入颜色数据，然后在 LCD 上面，我们就可以看到对应的点显示我们写入的颜色了。读点流程为：设置坐标->读 GRAM 指令->读取颜色数据，这样就可以获取到对应点的颜色数据了。

2.硬件连接

本章实验功能简介：开机时先初始化 LCD，然后开始显示正点原子 LOGO，12/16/24/32 号字体等信息，同时使用 LED 灯来指示程序正在运行。

本章需要用到的硬件资源如下：

1、指示灯 LED

2、TFTLCD 模块

3、ATK-WIRELESS 接口

ATK-1.3’ TFTLCD 模块可以直接插在 ALIENTEK STM32 开发板上的 ATK-WIRELESS 接口处。在 MINI 开发板上，ATK-1.3’ TFTLCD 模块与 MINI 开发板对应的关系如下：

SCK：对应 PA5，即 SPI1_SCK

SDA：对应 PA7，即 SPI1_SDA

WR：连接 PA6

CS：连接 PC4

PWR：连接 PA4

RESET：连接 PA1

3.软件实现

本实验我们使用四线 SPI 来驱动 ATK-1.3’ TFTLCD 模块，所以我们通过 WR（DC）信

号线来控制是发送命令还是发送数据到 LCD。代码如下所示：

```
/**
 * @brief       写命令到 LCD
 *
 * @param       cmd         需要发送的命令
 *
 * @return      void
 */
static void LCD_Write_Cmd(u8 cmd)
{
    LCD_DC = 0;

    LCD_SPI_Send(&cmd, 1);
}

/**
 * @brief       写数据到 LCD
 *
 * @param       cmd         需要发送的数据
 *
 * @return      void
 */
static void LCD_Write_Data(u8 data)
{
    LCD_DC = 1;

    LCD_SPI_Send(&data, 1);
}
```

LCD 的 SPI 通讯时序大家可以通过 ST7789V2 数据手册进行学习，这里就不多介绍了。

下面我们就来重点关注一下上面我们提到的 0x36 指令（存储器访问控制指令，即 RAM 指针增长方向），这里只粘贴了部分代码。

```
    /* Memory Data Access Control */
    LCD_Write_Cmd(0x36);
    LCD_Write_Data(0x00);
```

从以上代码可以看到，LCD 的 RAM 指令增长方向被设置成了从左到右，从上到下的方式，这个方式决定了字库取模方式和图片显示等问题。如果方向设置的好，我们只需要将字库和图片数据不停的往 LCD 填充就好了，就可以大大提高显示速度。

下面我们就来看看画点函数，该函数的实现代码如下：

```
/**
 *  设置数据写入 LCD 缓存区域
 *
 * @param       x1,y1       起点坐标
 * @param       x2,y2       终点坐标
 *
```

```
 * @return    void
 */
void LCD_Address_Set(u16 x1, u16 y1, u16 x2, u16 y2)
{
    LCD_Write_Cmd(0x2a);
    LCD_Write_Data(x1 >> 8);
    LCD_Write_Data(x1);
    LCD_Write_Data(x2 >> 8);
    LCD_Write_Data(x2);

    LCD_Write_Cmd(0x2b);
    LCD_Write_Data(y1 >> 8);
    LCD_Write_Data(y1);
    LCD_Write_Data(y2 >> 8);
    LCD_Write_Data(y2);

    LCD_Write_Cmd(0x2C);
}
/**
 * @brief       写半个字的数据到 LCD
 *
 * @param      cmd        需要发送的数据
 *
 * @return    void
 */
void LCD_Write_HalfWord(const u16 da)
{
    u8 data[2] = {0};

    data[0] = da >> 8;
    data[1] = da;

    LCD_DC = 1;
    LCD_SPI_Send(data, 2);
}
/**
 *  画点函数
 *
 * @param      x,y        画点坐标
 *
 * @return    void
 */
void LCD_Draw_Point(u16 x, u16 y)
{
```

```
    LCD_Address_Set(x, y, x, y);
    LCD_Write_HalfWord(POINT_COLOR);
}
```

该函数实现比较简单，就是先设置坐标，然后往坐标写颜色数据。其中 POINT_COLOR 是我们定义的一个全局变量，用于存放画笔颜色，顺带介绍一下另外一个全局变量 BACK_COLOR，该变量代表 LCD 的背景颜色。LCD_Draw_Point 函数虽然简单，但是至关重要，其他函数都可以调用这个函数实现。在例程源代码中，为了提高显示速度，很少用到画点函数来实现上层函数功能，因为画点函数的效率有点低。但是可以供到大家学习使用。

由于 ATK-1.3’TFTLCD 模块是 SPI 通讯接口的，在速度上肯定会比不上那些使用 8080 等并口的 TFTLCD 显示屏，为了提高显示速度，增加了一个 LCD 缓存，以提高显示效果，这个缓存会影响清屏函数 LCD_Clear、填充函数 LCD_Fill 和画线函数 LCD_DrawLine，修改缓存大小时，请注意！！！

```
//LCD 缓存大小设置，修改此值时请注意！！！！修改这两个值时可能会影响以下函数
LCD_Clear/LCD_Fill/ LCD_DrawLine
#define LCD_TOTAL_BUF_SIZE      (240*240*2)
#define LCD_Buf_Size 1152
static u8 lcd_buf[LCD_Buf_Size];
```

最后就来看一下字符显示函数 LCD_ShowChar，该函数可以显示 12/16/24/32 号字体，如果大家需要其他字体的话可以直接修改这个函数。特别要注意字库取模方向！

```
/**
 * @brief       显示一个 ASCII 码字符
 *
 * @param       x,y         显示起始坐标
 * @param       chr         需要显示的字符
 * @param       size 字体大小(支持 16/24/32 号字体)
 *
 * @return      void
 */
void LCD_ShowChar(u16 x, u16 y, char chr, u8 size)
{
    u8 temp, t1, t;
    u8 csize;       //得到字体一个字符对应点阵集所占的字节数
    u16 colortemp;
    u8 sta;

    chr = chr - ' '; //得到偏移后的值（ASCII 字库是从空格开始取模，所以-' '就是对应字符的字库）

    if((x > (LCD_Width - size / 2)) || (y > (LCD_Height - size)))       return;

    LCD_Address_Set(x, y, x + size / 2 - 1, y + size - 1);//(x,y,x+8-1,y+16-1)

    if((size == 16) || (size == 32) )       //16 和 32 号字体
```

```
{
    csize = (size / 8 + ((size % 8) ? 1 : 0)) * (size / 2);

    for(t = 0; t < csize; t++)
    {
        if(size == 16)temp = asc2_1608[chr][t];//调用 1608 字体
        else if(size == 32)temp = asc2_3216[chr][t]; //调用 3216 字体
        else return;                    //没有的字库

        for(t1 = 0; t1 < 8; t1++)
        {
            if(temp & 0x80) colortemp = POINT_COLOR;
            else colortemp = BACK_COLOR;

            LCD_Write_HalfWord(colortemp);
            temp <<= 1;
        }
    }
}

else if    (size == 12)      //12 号字体
{
    csize = (size / 8 + ((size % 8) ? 1 : 0)) * (size / 2);

    for(t = 0; t < csize; t++)
    {
        temp = asc2_1206[chr][t];

        for(t1 = 0; t1 < 6; t1++)
        {
            if(temp & 0x80) colortemp = POINT_COLOR;
            else colortemp = BACK_COLOR;

            LCD_Write_HalfWord(colortemp);
            temp <<= 1;
        }
    }
}

else if(size == 24)         //24 号字体
{
    csize = (size * 16) / 8;

    for(t = 0; t < csize; t++)
```

```
            {
                temp = asc2_2412[chr][t];

                if(t % 2 == 0)sta = 8;
                else sta = 4;

                for(t1 = 0; t1 < sta; t1++)
                {
                    if(temp & 0x80) colortemp = POINT_COLOR;
                    else colortemp = BACK_COLOR;

                    LCD_Write_HalfWord(colortemp);
                    temp <<= 1;
                }
            }
        }
    }
```

关于 LCD 的驱动代码问题就讲到这里，我们的源码中还提供了画矩形、画圆、显示数字和显示图片等等函数，大家可以直接 对照源码进行学习。

4.实验验证

代码编译成功后，直接下载代码到我们的 STM32 开发板上，然后将 ATK-1.3’TFTLCD 模块接到开发板上的 ATK-WIRELESS 接口上就可以测试模块的显示效果了。如图 4.1 所示：

图 4.1 LCD 显示效果图