JLX400-02101-BN 使用说明书

(插接式 FPC)

目 录

1. 概述

晶联讯电子专注于液晶屏及液晶模块的研发、制造。所生产 JLX400-02101 型 TFT 模块由于使用方便、显示清晰，广泛应用于各种人机交流面板。

JLX400-02101 可以显示 480 列*800 行点阵彩色图片,或显示 20 个/行*30 行 16*16 点阵的汉字，或显示 40 个/行*60 行 8*8 点阵的英文、数字、符号。

2. JLX400-02101 图像型点阵 TFT 模块的特性

2.1 结构轻、薄、带背光。

2.2 IC 采用 NT35510,功能强大，稳定性好

2.3 显示内容:

●480*800 点阵彩色图片;

●可选用 32*32 点阵或其他点阵的图片来自编汉字，按照 32*32 点阵汉字来计算可显示 15 个字/行*25 行。

●可选用 16*16 点阵或其他点阵的图片来自编汉字，按照 16*16 点阵汉字来计算可显示 30 个字/行*50 行。

2.4 指令功能强：例如可以用指令控制显示内容顺时针旋转 90、逆时针旋转 90°或倒立竖放。

2.5 接口方式：可选择 8 位或 16 位并行接口方式。

2.6 工作温度宽:-20℃ – +70℃;

2.7 工作温度宽:-30℃ – +80℃;

3. 外形尺寸及接口引脚功能

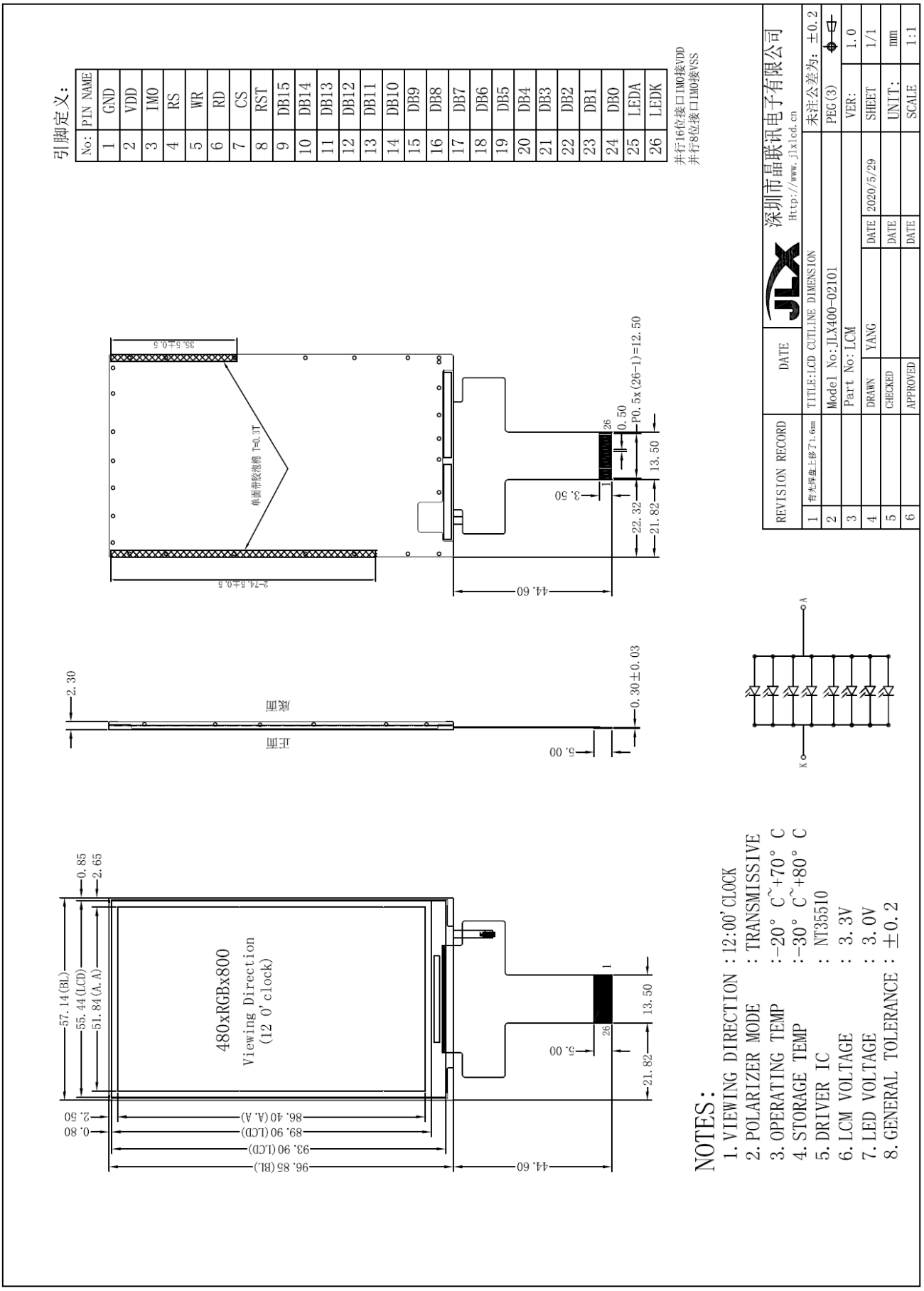


图 1. 带背光的 TFT 模块外形尺寸

模块的接口引脚功能

引 线 号	符 号	名 称	功 能
1	GND	接地	0V
2	VDD	供电电源正极	供电电源正极 3.3V
3	IM0	接口方式选择	16 位并行接口：高电平， 8 位并行接口：低电平
4	RS	寄存器选择信号	H:数据寄存器 0:指令寄存器
5	WR	写	写功能
6	RD	读	使能或读功能
7	CS	片选	低电平片选
8	RST	复位	低电平复位，复位完成后，回到高电平，TFT 模块开始工作
9	DB15	I/O	数据总线 DB15
10	DB14	I/O	数据总线 DB14
11	DB13	I/O	数据总线 DB13
12	DB12	I/O	数据总线 DB12
13	DB11	I/O	数据总线 DB11
14	DB10	I/O	数据总线 DB10
15	DB9	I/O	数据总线 DB9
16	DB8	I/O	数据总线 DB8
17	DB7	I/O	数据总线 DB7
18	DB6	I/O	数据总线 DB6
19	DB5	I/O	数据总线 DB5
20	DB4	I/O	数据总线 DB4
21	DB3	I/O	数据总线 DB3
22	DB2	I/O	数据总线 DB2
23	DB1	I/O	数据总线 DB1
24	DB0	I/O	数据总线 DB0
25	LEDA	背光电源正极	接 3.0V（接 3.3V 串 10 欧电阻，接 5.0V 串 39 欧电阻）
26	LEDK	背光电源负极	接 VSS

表 1：模块的接口引脚功能

4. 基本原理

4.1TFT 屏（LCD）

在 **LCD** 上排列着 480×800 点阵,480 个列信号与驱动 IC 相连，800 个行信号也与驱动 IC 相连，IC 邦定在 LCD 玻璃上（这种加工工艺叫 COG）.

4.2 背光参数

该型号 TFT 模块带 LED 背光源。它的性能参数如下:
工作温度:-20∽+70° C;
存储温度:-30∽+80° C;
背光板是白色。
正常工作电流为：64∽160mA（LED 灯数共 6 颗,每颗灯是 8～20 mA）
工作电压：3.0V（接 3.3V 串 10 欧电阻，接 5.0V 串 39 欧电阻）

5. 技术参数

5.1 最大极限参数（超过极限参数则会损坏 TFT 模块）

名称	符号	标准值			单位
		最小	典型	最大	
电路电源	VDD – VSS	-0.3	3.0	3.3	V
工作温度		-20		+70	℃
储存温度		-30		+80	℃

表 2：最大极限参数

5.2 直流（DC）参数

名 称	符 号	测 试 条 件	标 准 值			单位
			MIN	TYPE	MAX	
工作电压	VDD		2.4	–	3.3	V
背光工作电压	VLED		2.9	3.0	3.1	V
输入高电平	V_{IHC}	–	0.8xVDD	–	VDD	V
输入低电平	V_{ILC}	–	VSS	–	0.2xVDD	V
输出高电平	V_{OHC}	I_{OH} = -0.5mA	0.8xVDD	–	VDD	V
输出低电平	V_{OHC}	I_{OL} = -0.5mA	VSS	–	0.2xVDD	V
模块工作电流	I_{DD}	VDD = 3.3V	–		0.3	mA
背光工作电流	I_{LED}	V_{LED}=3.0V	64	120	160	mA

表 3：直流（DC）参数

6. 读写时序特性

6.1 8080 写时序特性

The write cycle means that the host writes information (command or/and data) to the display via the interface. Each write cycle (WRX high-low-high sequence) consists of 3 control (D/CX, RDX, WRX) and data signals (D[23:0]). D/CX bit is a control signal, which tells if the data is a command or a data. The data signals are the command if the control signal is low (='0') and vice versa it is data (='1').

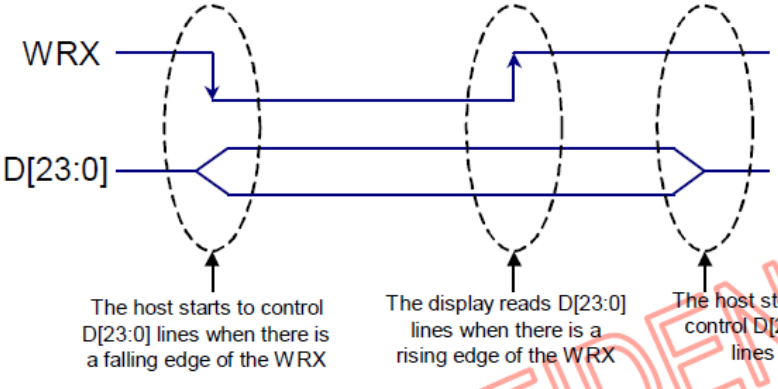


Fig. 5.1.1 80-Series WRX protocol

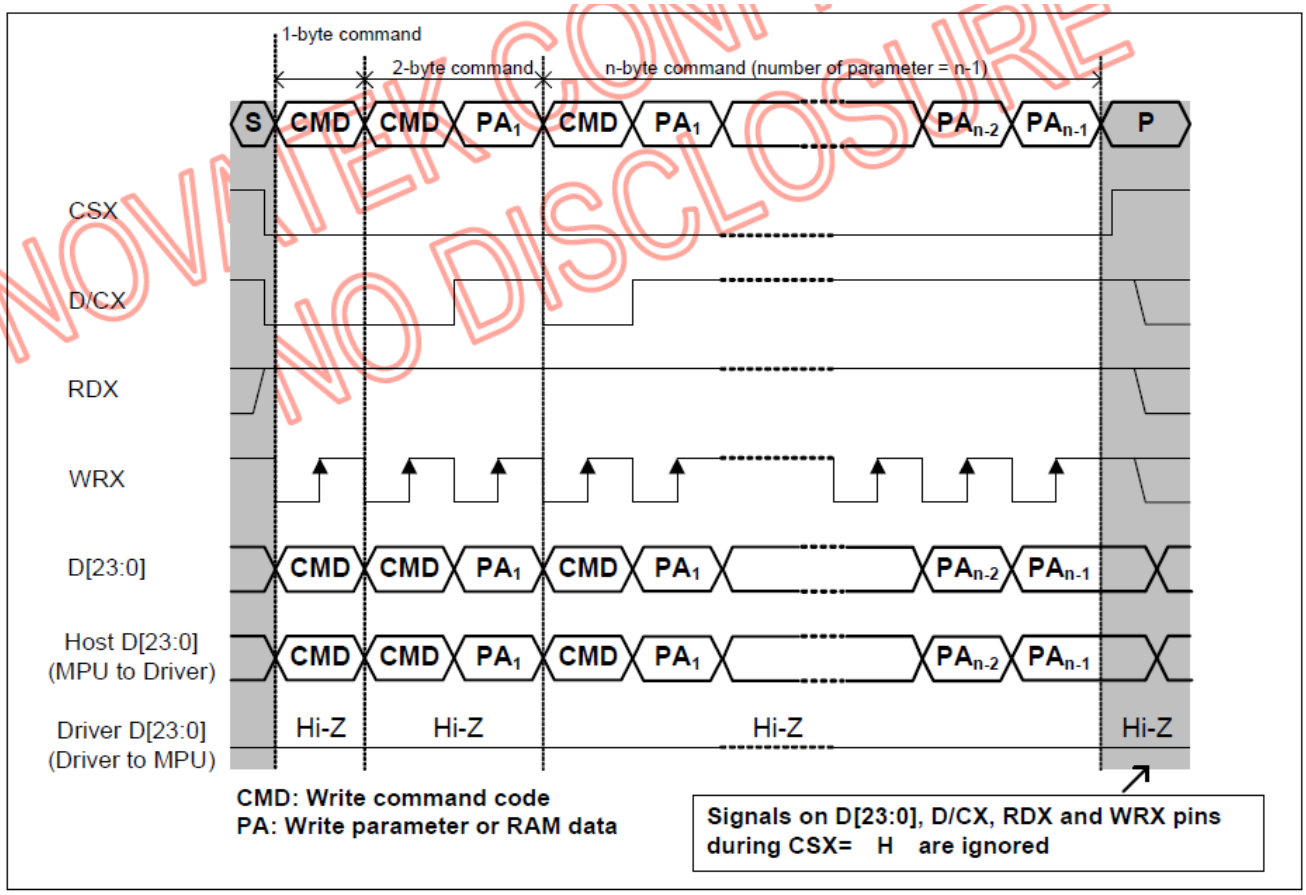


Fig. 5.1.2 80-Series parallel bus protocol, write to register or display RAM

图 1. 8080 写时序图

6.2 8080 读时序特性

The read cycle (RDX high-low-high sequence) means that the host reads information from display via interface. The display sends data (D[17:0]) to the host when there is a falling edge of RDX and the host reads data when there is a rising edge of RDX.

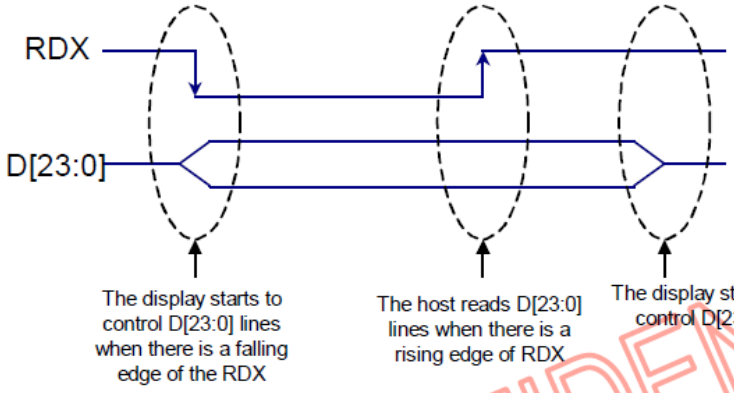


Fig. 5.1.3 80-Series RDX protocol

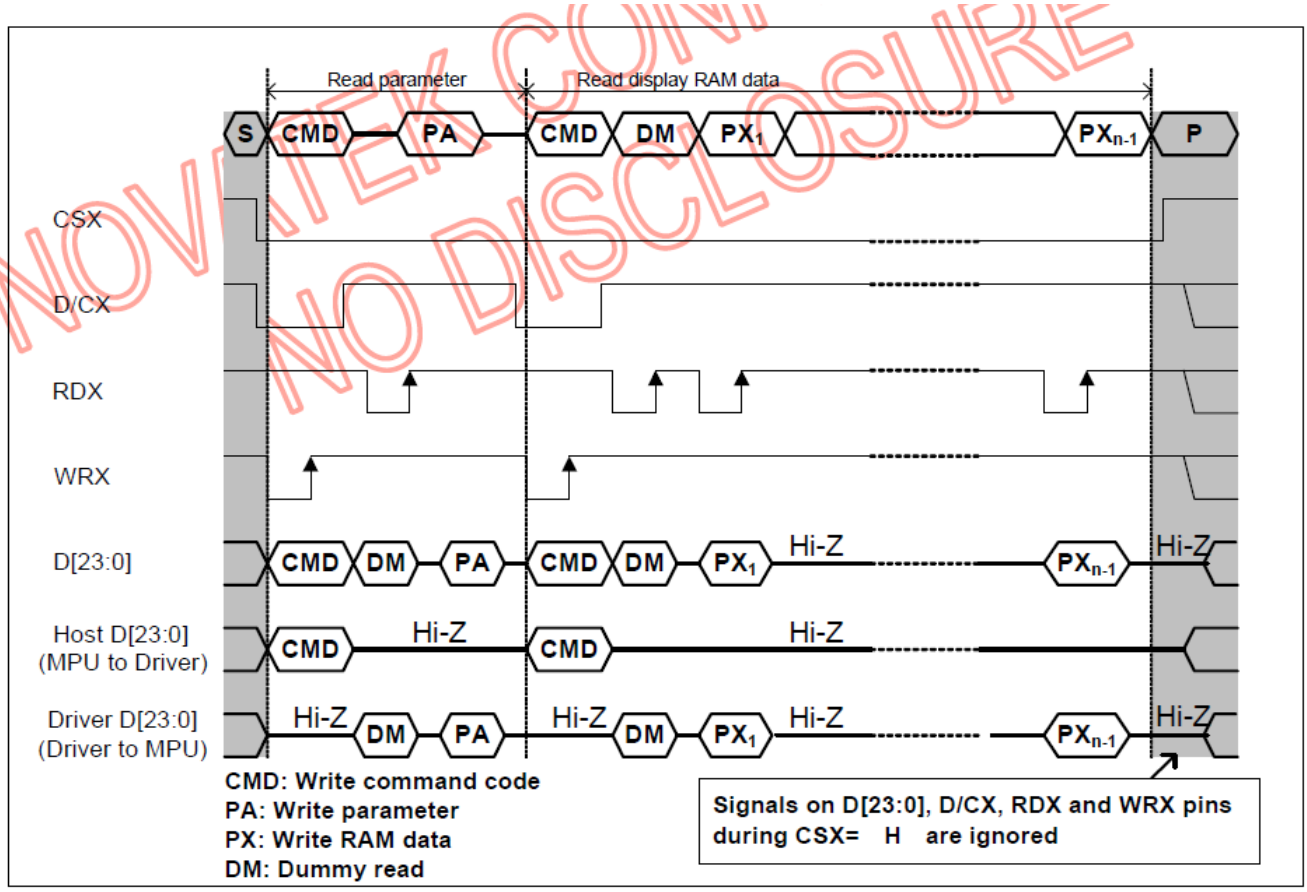


Fig. 5.1.4 80-Series parallel bus protocol, read from register or display RAM

图 2. 8080 读时序图

7．指令功能：

7.1 指令表

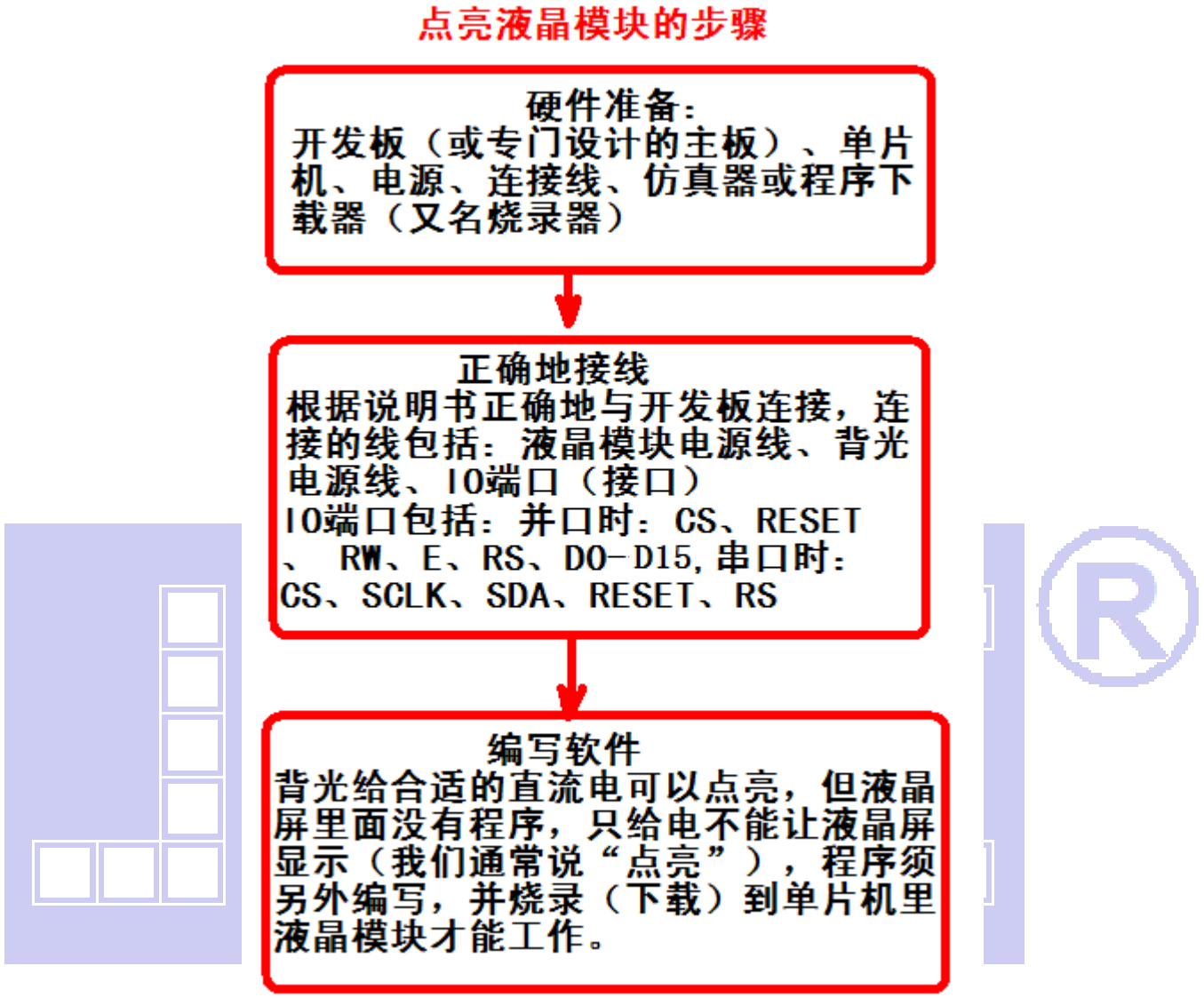
Instruction	ACT	R/W	Address		Parameter									Function
			MIPI	Others	D[15:8] (Non-MIPI)	D7	D6	D5	D4	D3	D2	D1	D0	
NOP	Dir	W	00h	0000h	No Argument (0000h in MDDI I/F)									No Operation
SWRESET	Cnd1	W	01h	0100h	No Argument (0000h in MDDI I/F)									Software reset
RDDID	Dir	R	04h	0400h	00h	ID17	ID16	ID15	ID14	ID13	ID12	ID11	ID10	Read display ID
				0401h	00h	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	
				0402h	00h	ID37	ID36	ID35	ID34	ID33	ID32	ID31	ID30	
RDNUMED	Dir	R	05h	X	X	P7	P6	P5	P4	P3	P2	P1	P0	Read No. of the Errors on DSI only
RDDPM	Dir	R	0Ah	0A00h	00h	D7	D6	D5	D4	D3	D2	D1	D0	Read Display Power Mode
RDDMADCTL	Dir	R	0Bh	0B00h	00h	D7	D6	D5	D4	D3	D2	D1	D0	Read Display MADCTR
RDDCOLMOD	Dir	R	0Ch	0C00h	00h	D7	D6	D5	D4	D3	D2	D1	D0	Read Display Pixel Format
RDDIM	Dir	R	0Dh	0D00h	00h	D7	D6	D5	D4	D3	D2	D1	D0	Read Display Image Mode
RDDSM	Dir	R	0Eh	0E00h	00h	D7	D6	D5	D4	D3	D2	D1	D0	Read Display Signal Mode
RDDSDR	Dir	R	0Fh	0F00h	00h	D7	D6	D5	D4	D3	D2	D1	D0	Read Display Self-diagnostic result
SLPIN	DVS	W	10h	1000h	No Argument (0000h in MDDI I/F)									Sleep in & booster off
SLPOUT	Dir	W	11h	1100h	No Argument (0000h in MDDI I/F)									Sleep out & booster on
PTLON	DVS	W	12h	1200h	No Argument (0000h in MDDI I/F)									Partial mode on
NORON	DVS	W	13h	1300h	No Argument (0000h in MDDI I/F)									Partial off (Normal)
INVOFF	DVS	W	20h	2000h	No Argument (0000h in MDDI I/F)									Display inversion off (normal)
INVON	DVS	W	21h	2100h	No Argument (0000h in MDDI I/F)									Display inversion on
ALLPOFF	DVS	W	22h	2200h	No Argument (0000h in MDDI I/F)									All pixel off (black)
ALLPON	DVS	W	23h	2300h	No Argument (0000h in MDDI I/F)									All pixel on (white)
GAMSET	DVS	W	26h	2600h	00h	GC7	GC6	GC5	GC4	GC3	GC2	GC1	GC0	Gamma curve select
DISPOFF	DVS	W	28h	2800h	No Argument (0000h in MDDI I/F)									Display off
DISPON	DVS	W	29h	2900h	No Argument (0000h in MDDI I/F)									Display on
CASET	Dir	W	2Ah	2A00h	00h	XS15	XS14	XS13	XS12	XS11	XS10	XS9	XS8	Column address set XS[15:0]: column start address XE[15:0]: column end address
				2A01h	00h	XS7	XS6	XS5	XS4	XS3	XS2	XS1	XS0	
				2A02h	00h	XE15	XE14	XE13	XE12	XE11	XE10	XE9	XE8	
				2A03h	00h	XE7	XE6	XE5	XE4	XE3	XE2	XE1	XE0	
RASET	Dir	W	2Bh	2B00h	00h	YS15	YS14	YS13	YS12	YS11	YS10	YS9	YS8	Row address set YS[15:0]: row start address YE[15:0]: row end address
				2B01h	00h	YS7	YS6	YS5	YS4	YS3	YS2	YS1	YS0	
				2B02h	00h	YE15	YE14	YE13	YE12	YE11	YE10	YE9	YE8	
				2B03h	00h	YE7	YE6	YE5	YE4	YE3	YE2	YE1	YE0	
RAMWR	Dir	W	2Ch	X	X	D7	D6	D5	D4	D3	D2	D1	D0	Memory write
RAMRD	Dir	R	2Eh	2E00h	00h	D7	D6	D5	D4	D3	D2	D1	D0	Memory read
PTLAR	DVS	W	30h	3000h	00h	PSL15	PSL14	PSL13	PSL12	PSL11	PSL10	PSL9	PSL8	Partial start/end address set PSL[15:0]: partial start address PEL[15:0]: partial end address
				3001h	00h	PSL7	PSL6	PSL5	PSL4	PSL3	PSL2	PSL1	PSL0	
				3002h	00h	PEL15	PEL14	PEL13	PEL12	PEL11	PEL10	PEL9	PEL8	
				3003h	00h	PEL7	PEL6	PEL5	PEL4	PEL3	PEL2	PEL1	PEL0	
TEOFF	DVS	W	34h	3400h	No Argument (0000h in MDDI I/F)									Tearing effect line off
TEON	DVS	W	35h	3500h	00h	-	-	-	-	-	-	-	M	Tearing effect mode set & on
MADCTL	Cnd2	W	36h	3600h	00h	MY	MX	MV	ML	RGB	MH	RSMX	RSMY	Memory data access control
IDMOFF	DVS	W	38h	3800h	No Argument (0000h in MDDI I/F)									Idle mode off
IDMON	DVS	W	39h	3900h	No Argument (0000h in MDDI I/F)									Idle mode on

Instruction	ACT	R/W	Address		Parameter									Function
			MIPI	Others	D[15:8] (Non-MIPI)	D7	D6	D5	D4	D3	D2	D1	D0	
COLMOD	Dir	W	3Ah	3A00h	00h	VIPF3	VIPF2	VIPF1	VIPF0	IFPF3	IFPF2	IFPF1	IFPF0	Interface pixel format
RAMWRC	Dir	W	3Ch	3C00h	00h	D7	D6	D5	D4	D3	D2	D1	D0	Memory write Continue
RAMRDC	Dir	R	3Eh	3C00h	00h	D7	D6	D5	D4	D3	D2	D1	D0	Memory read Continue
STESL	DVS	W	44h	4400h	00h	N15	N14	N13	N12	N11	N10	N9	N8	Set tearing effect scan line
				4401h	00h	N7	N6	N5	N4	N3	N2	N1	N0	
GSL	Dir	R	45h	4500h	00h	N15	N14	N13	N12	N11	N10	N9	N8	Get scan line
				4501h	00h	N7	N6	N5	N4	N3	N2	N1	N0	
DSTBON	DVS	W	4Fh	4F00h	00h	0	0	0	0	0	0	0	DSTB	Deep standby mode on
WRPFD	DVS	W	50h	5000h	00h	V017	V016	V015	V014	V013	V012	V011	V010	Write profile value for display
				5001h	00h	V027	V026	V025	V024	V023	V022	V021	V020	
				:	:	:	:	:	:	:	:	:	:	
				500Eh	00h	V157	V156	V155	V154	V153	V152	V151	V150	
				500Fh	00h	V167	V166	V165	V164	V163	V162	V161	V160	
WRDISBV	DVS	W	51h	5100h	00h	DBV7	DBV6	DBV5	DBV4	DBV3	DBV2	DBV1	DBV0	Write display brightness
RDDISBV	Dir	R	52h	5200h	00h	DBV7	DBV6	DBV5	DBV4	DBV3	DBV2	DBV1	DBV0	Read display brightness value
WRCTRLD	DVS	W	53h	5300h	00h	0	0	BCTRL	A	DD	BL	DB	G	Write control display
RDCTRLD	Dir	R	54h	5400h	00h	0	0	BCTRL	A	DD	BL	DB	G	Read control display value
WRCABC	DVS	W	55h	5500h	00h	0	0	0	0	0	0	C1	C0	Write CABC mode
RDCABC	Dir	R	56h	5600h	00h	0	0	0	0	0	0	C1	C0	Read CABC mode
WRHYSTE	DVS	W	57h	5700h	00h	I017	I016	I015	I014	I013	I012	I011	I010	Write hysteresis
				5701h	00h	I027	I026	I025	I024	I023	I022	I021	I020	
				:	:	:	:	:	:	:	:	:	:	
				570Eh	00h	I157	I156	I155	I154	I153	I152	I151	I150	
				570Fh	00h	I167	I166	I165	I164	I163	I162	I161	I160	
				5710h	00h	D017	D016	D015	D014	D013	D012	D011	D010	
				5711h	00h	D027	D026	D025	D024	D023	D022	D021	D020	
				:	:	:	:	:	:	:	:	:	:	
				571Eh	00h	D157	D156	D155	D154	D153	D152	D151	D150	
				571Fh	00h	D167	D166	D165	D164	D163	D162	D161	D160	
WRGAMMSET	DVS	W	58h	5800h	00h	G023	G022	G021	G020	G013	G012	G011	G010	Write gamma setting
				5801h	00h	G043	G042	G041	G040	G033	G032	G031	G030	
				:	:	:	:	:	:	:	:	:	:	
				5806h	00h	G143	G142	G141	G140	G133	G132	G131	G130	
				5807h	00h	G163	G162	G161	G160	G153	G152	G151	G150	
RDFSVM	Dir	R	5Ah	5A00h	00h	FSV15	FSV14	FSV13	FSV12	FSV11	FSV10	FSV9	FSV8	Read FS value MSBs
RDFSVL	Dir	R	5Bh	5B00h	00h	FSV7	FSV6	FSV5	FSV4	FSV3	FSV2	FSV1	FSV0	Read FS value LSBs
RDMFFSVM	Dir	R	5Ch	5C00h	00h	FFSV15	FFSV14	FFSV13	FFSV12	FFSV11	FFSV10	FFSV9	FFSV8	Read median filter FS value MSBs
RDMFFSVL	Dir	R	5Dh	5D00h	00h	FFSV7	FFSV6	FFSV5	FFSV4	FFSV3	FFSV2	FFSV1	FFSV0	Read median filter FS value LSBs
WRCABCMB	DVS	W	5Eh	5E00h	00h	CMB7	CMB6	CMB5	CMB4	CMB3	CMB2	CMB1	CMB0	Write CABC minimum brightness
RDCABCMB	Dir	R	5Fh	5F00h	00h	CMB7	CMB6	CMB5	CMB4	CMB3	CMB2	CMB1	CMB0	Read CABC minimum brightness

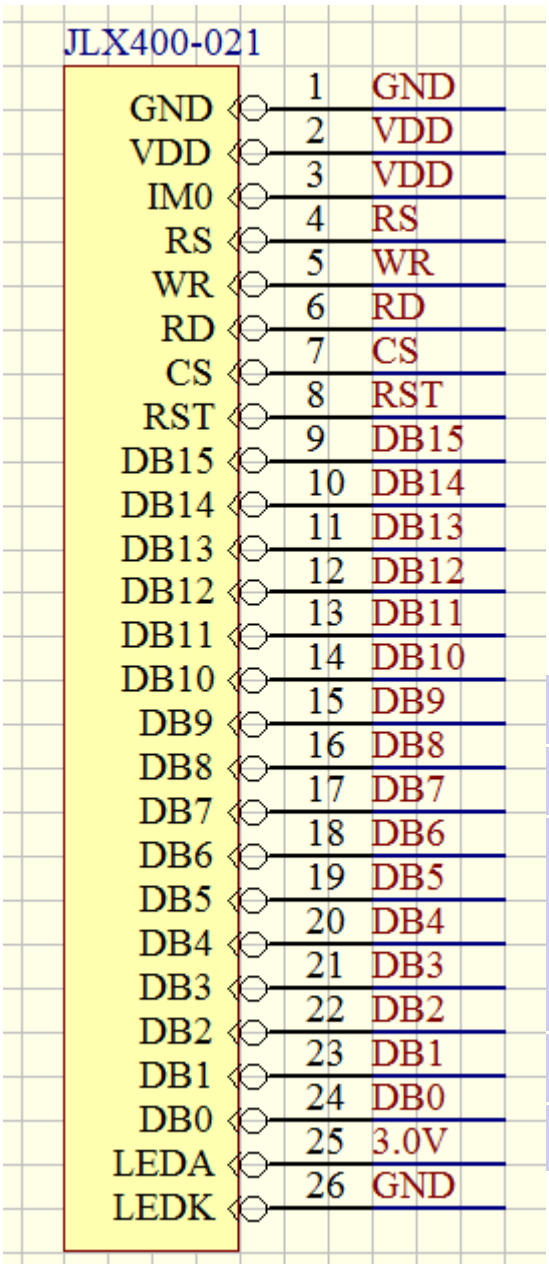
Instruction	ACT	R/W	Address		Parameter									Function
			MIPI	Others	D[15:8] (Non-MIPI)	D7	D6	D5	D4	D3	D2	D1	D0	
WRLSCC	DVS	W	65h	6500h	00h	CC15	CC14	CC13	CC12	CC11	CC10	CC9	CC8	Write light sensor compensation coefficient
				6501h	00h	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0	
RDLSCCM	Dir	R	66h	6600h	00h	CC15	CC14	CC13	CC12	CC11	CC10	CC9	CC8	Read LSCC value MSBs
RDLSCCL	Dir	R	67h	6700h	00h	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0	Read LSCC value LSBs
RDBWLB	Dir	R	70h	7000h	00h	Bkx1	Bkx0	Bky1	Bky0	Wx1	Wx0	Wy1	Wy0	Read Black/White low bit
RDBkx	Dir	R	71h	7100h	00h	Bkx9	Bkx8	Bkx7	Bkx6	Bkx5	Bkx4	Bkx3	Bkx2	Read Bkx
RDBky	Dir	R	72h	7200h	00h	Bky9	Bky8	Bky7	Bky6	Bky5	Bky4	Bky3	Bky2	Read Bky
RDWx	Dir	R	73h	7300h	00h	Wx9	Wx8	Wx7	Wx6	Wx5	Wx4	Wx3	Wx2	Read Wx
RDWy	Dir	R	74h	7400h	00h	Wy9	Wy8	Wy7	Wy6	Wy5	Wy4	Wy3	Wy2	Read Wy
RDRGLB	Dir	R	75h	7500h	00h	Rx1	Rx0	Ry1	Ry0	Gx1	Gx0	Gy1	Gy0	Read Red/Green low bit
RDRx	Dir	R	76h	7600h	00h	Rx9	Rx8	Rx7	Rx6	Rx5	Rx4	Rx3	Rx2	Read Rx
RDRy	Dir	R	77h	7700h	00h	Ry9	Ry8	Ry7	Ry6	Ry5	Ry4	Ry3	Ry2	Read Ry
RDGx	Dir	R	78h	7800h	00h	Gx9	Gx8	Gx7	Gx6	Gx5	Gx4	Gx3	Gx2	Read Gx
RDGy	Dir	R	79h	7900h	00h	Gy9	Gy8	Gy7	Gy6	Gy5	Gy4	Gy3	Gy2	Read Gy
RDBALB	Dir	R	7Ah	7A00h	00h	Bx1	Bx0	By1	By0	Ax1	Ax0	Ay1	Ay0	Read Blue/AColor low bit
RDBx	Dir	R	7Bh	7B00h	00h	Bx9	Bx8	Bx7	Bx6	Bx5	Bx4	Bx3	Bx2	Read Bx
RDBy	Dir	R	7Ch	7C00h	00h	By9	By8	By7	By6	By5	By4	By3	By2	Read By
RDAx	Dir	R	7Dh	7D00h	00h	Ax9	Ax8	Ax7	Ax6	Ax5	Ax4	Ax3	Ax2	Read Ax
RDAy	Dir	R	7Eh	7E00h	00h	Ay9	Ay8	Ay7	Ay6	Ay5	Ay4	Ay3	Ay2	Read Ay
RDDDBS	Dir	R	A1h	A100h	00h	SID7	SID6	SID5	SID4	SID3	SID2	SID1	SID0	Read DDB start
				A101h	00h	SID15	SID14	SID13	SID12	SID11	SID10	SID9	SID8	
				A102h	00h	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	
				A103h	00h	MID15	MID14	MID13	MID12	MID11	MID10	MID9	MID8	
				A104h	00h	1	1	1	1	1	1	1	1	
RDDDBC	Dir	R	A8h	A800h	00h	SID7	SID6	SID5	SID4	SID3	SID2	SID1	SID0	Read DDB continue
				A801h	00h	SID15	SID14	SID13	SID12	SID11	SID10	SID9	SID8	
				A802h	00h	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	
				A803h	00h	MID15	MID14	MID13	MID12	MID11	MID10	MID9	MID8	
				A804h	00h	1	1	1	1	1	1	1	1	
RDFCS	Dir	R	AAh	AA00h	00h	FCS7	FCS6	FCS5	FCS4	FCS3	FCS2	FCS1	FCS0	Read first checksum
RDCCS	Dir	R	AFh	AF00h	00h	CCS7	CCS6	CCS5	CCS4	CCS3	CCS2	CCS1	CCS0	Read continue checksum
RDID1	Dir	R	DAh	DA00h	00h	ID17	ID16	ID15	ID14	ID13	ID12	ID11	ID10	Read ID1
RDID2	Dir	R	DBh	DB00h	00h	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	Read ID2
RDID3	Dir	R	DCh	DC00h	00h	ID37	ID36	ID35	ID34	ID33	ID32	ID31	ID30	Read ID3

7.2 初始化方法

用户所编的显示程序, 开始必须进行初始化, 否则模块无法正常显示, 过程请参考程序



7. 3 并行原理图


JLX400-021
GND 1 GND
VDD 2 VDD
IM0 3 VDD
RS 4 RS
WR 5 WR
RD 6 RD
CS 7 CS
RST 8 RST
DB15 9 DB15
DB14 10 DB14
DB13 11 DB13
DB12 12 DB12
DB11 13 DB11
DB10 14 DB10
DB9 15 DB9
DB8 16 DB8
DB7 17 DB7
DB6 18 DB6
DB5 19 DB5
DB4 20 DB4
DB3 21 DB3
DB2 22 DB2
DB1 23 DB1
DB0 24 DB0
LEDA 25 3.0V
LEDK 26 GND

7.4 程序

//STM32F103 例程序

```
//管理 LCD 重要参数
//默认为竖屏
_lcd_dev lcddev;

//画笔颜色,背景颜色
u16 POINT_COLOR = 0x001F,BACK_COLOR = 0xFFFF;
u16 DeviceCode;

void LCD_write(u16 VAL)
{
    LCD_CS_CLR;
    DATAOUT(VAL);
    LCD_WR_CLR;
    LCD_WR_SET;
    LCD_CS_SET;
}

u16 LCD_read(void)
{
    u16 data;
    LCD_CS_CLR;
    LCD_RD_CLR;
//  delay_us(1);//延时 1us
    data = DATAIN;
    LCD_RD_SET;
    LCD_CS_SET;
    return data;
}

/*****************************************************************************
 * @name       :void LCD_WR_REG(u16 data)
 * @date       :2018-08-09
 * @function   :Write an 16-bit command to the LCD screen
 * @parameters :data:Command value to be written
 * @retvalue   :None
******************************************************************************/
void LCD_WR_REG(u16 data)
{
  LCD_RS_CLR;
    #if LCD_USE8BIT_MODEL
    LCD_write(data<<8);
    #else
    LCD_write(data);
```

```
    #endif
}

/*****************************************************************************
 * @name       :void LCD_WR_DATA(u16 data)
 * @date       :2018-08-09
 * @function   :Write an 16-bit data to the LCD screen
 * @parameters :data:data value to be written
 * @retvalue   :None
*****************************************************************************/
void LCD_WR_DATA(u16 data)
{
    LCD_RS_SET;
    #if LCD_USE8BIT_MODEL
    LCD_write(data<<8);
    #else
    LCD_write(data);
    #endif
}

/*****************************************************************************
 * @name       :u16 LCD_RD_DATA(void)
 * @date       :2018-11-13
 * @function   :Read an 16-bit value from the LCD screen
 * @parameters :None
 * @retvalue   :read value
*****************************************************************************/
u16 LCD_RD_DATA(void)
{
    LCD_RS_SET;
    #if LCD_USE8BIT_MODEL
    return (LCD_read()>>8);
    #else
    return LCD_read();
    #endif
}

/*****************************************************************************
 * @name       :void LCD_WriteReg(u16 LCD_Reg, u16 LCD_RegValue)
 * @date       :2018-08-09
 * @function   :Write data into registers
 * @parameters :LCD_Reg:Register address
                LCD_RegValue:Data to be written
 * @retvalue   :None
*****************************************************************************/
void LCD_WriteReg(u16 LCD_Reg, u16 LCD_RegValue)
```

```
{
    LCD_WR_REG(LCD_Reg);
    LCD_WR_DATA(LCD_RegValue);
}

//等待按键,3个按键，有任何一个按键按下都可以进入下一步
void waitkey(void)
{
    repeat:
        if(KEY0==1&&KEY1==1&&KEY2==1&&KEY3==1)    goto repeat;
        else delay_ms(500);
}



/*****************************************************************************
 * @name       :void LCD_ReadReg(u16 LCD_Reg,u16 *Rval,int n)
 * @date       :2018-11-13
 * @function   :read value from specially registers
 * @parameters :LCD_Reg:Register address
 * @retvalue   :read value
*****************************************************************************/
void LCD_ReadReg(u16 LCD_Reg,u16 *Rval,int n)
{
    LCD_WR_REG(LCD_Reg);
    GPIOC->CRL=0X88888888; //PC0-7  上拉输入
    GPIOA->CRL=0X88888888; //PA0-7 上拉输入
    GPIOC->ODR=0X0000;     //全部输出高
    GPIOA->ODR=0X0000;     //全部输出高
    while(n--)
    {
        *(Rval++) = LCD_RD_DATA();
    }

    GPIOC->CRL=0X33333333;      //PC0-7  上拉输出
    GPIOA->CRL=0X33333333;      //PA0-7 上拉输出
    GPIOC->ODR=0XFFFF;          //全部输出高
    GPIOA->ODR=0XFFFF;          //全部输出高
}

/*****************************************************************************
 * @name       :void LCD_WriteRAM_Prepare(void)
 * @date       :2018-08-09
 * @function   :Write GRAM
 * @parameters :None
 * @retvalue   :None
```

```
*****************************************************************************/
void LCD_WriteRAM_Prepare(void)
{
    LCD_WR_REG(lcddev.wramcmd);
}

/*****************************************************************************
 * @name       :void LCD_ReadRAM_Prepare(void)
 * @date       :2018-11-13
 * @function   :Read GRAM
 * @parameters :None
 * @retvalue   :None
*****************************************************************************/
void LCD_ReadRAM_Prepare(void)
{
    LCD_WR_REG(lcddev.rramcmd);
}

/*****************************************************************************
 * @name       :void Lcd_WriteData_16Bit(u16 Data)
 * @date       :2018-08-09
 * @function   :Write an 16-bit command to the LCD screen
 * @parameters :Data:Data to be written
 * @retvalue   :None
*****************************************************************************/
void Lcd_WriteData_16Bit(u16 Data)
{
   LCD_RS_SET;
    #if LCD_USE8BIT_MODEL
        LCD_CS_CLR;
        DATAOUT(Data);
        LCD_WR_CLR;
        LCD_WR_SET;
        DATAOUT(Data<<8);
        LCD_WR_CLR;
        LCD_WR_SET;
        LCD_CS_SET;
 //  LCD_write(Data&0xFF00);
 //   LCD_write(Data<<8);
    #else
    LCD_write(Data);
    #endif
}

//快速画点
//x,y:坐标
```

```
//color:颜色
void LCD_Fast_DrawPoint(u16 x,u16 y,u16 color)
{
    LCD_WR_REG(0x2a00);
    LCD_WR_DATA(x>>8);
    LCD_WR_DATA(x&0XFF);
    LCD_WR_REG(0x2b00);
    LCD_WR_DATA(y>>8);
    LCD_WR_DATA(y&0XFF);

    LCD_RS_CLR;
    LCD_CS_CLR;
    LCD_WR_REG(0x2c00);
//  DATAOUT(0x2c);//写指令
    LCD_WR_CLR;
    LCD_WR_SET;
    LCD_CS_SET;
    Lcd_WriteData_16Bit(color);     //写数据
}


u16 Color_To_565(u8 r, u8 g, u8 b)
{
    return ((r & 0xF8) << 8) | ((g & 0xFC) << 3) | ((b & 0xF8) >> 3);
}

/*****************************************************************************
 * @name       :u16 Lcd_ReadData_16Bit(void)
 * @date       :2018-11-13
 * @function   :Read an 16-bit value from the LCD screen
 * @parameters :None
 * @retvalue   :read value
*****************************************************************************/
u16 Lcd_ReadData_16Bit(void)
{
    u16 r,g,b;
    LCD_RS_SET;
    LCD_CS_CLR;

    //dummy data
    LCD_RD_CLR;
    delay_us(1);//延时 1us
    r = DATAIN;
    LCD_RD_SET;

    //8bit:red data
```

```
    //16bit:red and green data
    LCD_RD_CLR;
    delay_us(1);//延时 1us
    r = DATAIN;
    LCD_RD_SET;

    //8bit:green data
    //16bit:blue data
    LCD_RD_CLR;
    delay_us(1);//延时 1us
    g = DATAIN;
    LCD_RD_SET;

    #if LCD_USE8BIT_MODEL
    //blue data
    LCD_RD_CLR;
    delay_us(1);//延时 1us
    b = DATAIN;
    LCD_RD_SET;
    r >>= 8;
    g >>= 8;
    b >>= 8;
    #else
    b = g>>8;
    g = r&0xFF;
    r = r>>8;
    #endif
    LCD_CS_SET;
    return Color_To_565(r, g, b);
}

/*****************************************************************************
 * @name       :void LCD_DrawPoint(u16 x,u16 y)
 * @date       :2018-08-09
 * @function   :Write a pixel data at a specified location
 * @parameters :x:the x coordinate of the pixel
                y:the y coordinate of the pixel
 * @retvalue   :None
*****************************************************************************/
void LCD_DrawPoint(u16 x,u16 y)
{
    LCD_SetCursor(x,y);//设置光标位置
    Lcd_WriteData_16Bit(POINT_COLOR);
}

/*****************************************************************************
```

```
 * @name       :u16 LCD_ReadPoint(u16 x,u16 y)
 * @date       :2018-11-13
 * @function   :Read a pixel color value at a specified location
 * @parameters :x:the x coordinate of the pixel
                y:the y coordinate of the pixel
 * @retvalue   :the read color value
******************************************************************************/
u16 LCD_ReadPoint(u16 x,u16 y)
{
    u16 color;
    if(x>=lcddev.width||y>=lcddev.height)
    {
        return 0;//超过了范围,直接返回
    }
    LCD_SetCursor(x,y);//设置光标位置
    LCD_ReadRAM_Prepare();
    GPIOA->CRL=0X88888888; //PC0-7  上拉输入
    GPIOC->CRL=0X88888888; //PA0-7 上拉输入
    GPIOA->ODR=0XFFFF;     //全部输出高
    GPIOC->ODR=0XFFFF;     //全部输出高

    color = Lcd_ReadData_16Bit();

    GPIOA->CRL=0X33333333;      //PC0-7  上拉输出
    GPIOC->CRL=0X33333333;      //PA0-7 上拉输出
    GPIOA->ODR=0XFFFF;          //全部输出高
    GPIOC->ODR=0XFFFF;          //全部输出高
    return color;
}

/******************************************************************************
 * @name       :void LCD_Clear(u16 Color)
 * @date       :2018-08-09
 * @function   :Full screen filled LCD screen
 * @parameters :color:Filled color
 * @retvalue   :None
******************************************************************************/
void LCD_Clear(u16 Color)
{
  unsigned int i;//,m;
    LCD_SetWindows(0,0,lcddev.width-1,lcddev.height-1);
    for(i=0;i<lcddev.height*lcddev.width;i++)
    {
 //   for(m=0;m<lcddev.width;m++)
  //  {
              Lcd_WriteData_16Bit(Color);
```

```
    //  }
    }
}

/*****************************************************************************
 * @name       :void LCD_GPIOInit(void)
 * @date       :2018-08-09
 * @function   :Initialization LCD screen GPIO
 * @parameters :None
 * @retvalue   :None
*****************************************************************************/
void LCD_GPIOInit(void)
{
    GPIO_InitTypeDef  GPIO_InitStructure;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC|RCC_APB2Periph_GPIOA|RCC_APB2Periph_GPIOB|RCC_APB2Periph_AFIO,EN
ABLE);
    GPIO_PinRemapConfig(GPIO_Remap_SWJ_JTAGDisable , ENABLE);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11| GPIO_Pin_10| GPIO_Pin_2| GPIO_Pin_1 | GPIO_Pin_0; //GPIOC10,6,7,8,9,4
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;   //推挽输出
    GPIO_Init(GPIOB, &GPIO_InitStructure);
    GPIO_SetBits(GPIOB,GPIO_Pin_11|GPIO_Pin_10|GPIO_Pin_2|GPIO_Pin_1|GPIO_Pin_0);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_All;   //
    GPIO_Init(GPIOA, &GPIO_InitStructure); //GPIOA
    GPIO_SetBits(GPIOA,GPIO_Pin_All);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_All;   //
    GPIO_Init(GPIOC, &GPIO_InitStructure); //GPIOA
    GPIO_SetBits(GPIOC,GPIO_Pin_All);
}

/*****************************************************************************
 * @name       :void LCD_RESET(void)
 * @date       :2018-08-09
 * @function   :Reset LCD screen
 * @parameters :None
 * @retvalue   :None
*****************************************************************************/
void LCD_RESET(void)
{
    LCD_RST_CLR;
    delay_ms(100);
    LCD_RST_SET;
```

```
    delay_ms(50);
}

/*********************************************************************************
 * @name        :void LCD_Init(void)
 * @date        :2018-08-09
 * @function    :Initialization LCD screen
 * @parameters  :None
 * @retvalue    :None
*********************************************************************************/
void LCD_Init(void)
{
    LCD_GPIOInit();//LCD GPIO 初始化
    delay_ms(50);
    LCD_RESET(); //LCD 复位
//************ NT35510 初始化**********//
    LCD_WR_REG(0xF000); LCD_WR_DATA(0x55);
    LCD_WR_REG(0xF001); LCD_WR_DATA(0xAA);
    LCD_WR_REG(0xF002); LCD_WR_DATA(0x52);
    LCD_WR_REG(0xF003); LCD_WR_DATA(0x08);
    LCD_WR_REG(0xF004); LCD_WR_DATA(0x01);
    //# AVDD: manual); LCD_WR_DATA(
    LCD_WR_REG(0xB600); LCD_WR_DATA(0x34);
    LCD_WR_REG(0xB601); LCD_WR_DATA(0x34);
    LCD_WR_REG(0xB602); LCD_WR_DATA(0x34);

    LCD_WR_REG(0xB000); LCD_WR_DATA(0x0D);//09
    LCD_WR_REG(0xB001); LCD_WR_DATA(0x0D);
    LCD_WR_REG(0xB002); LCD_WR_DATA(0x0D);
    //# AVEE: manual); LCD_WR_DATA( -6V
    LCD_WR_REG(0xB700); LCD_WR_DATA(0x24);
    LCD_WR_REG(0xB701); LCD_WR_DATA(0x24);
    LCD_WR_REG(0xB702); LCD_WR_DATA(0x24);

    LCD_WR_REG(0xB100); LCD_WR_DATA(0x0D);
    LCD_WR_REG(0xB101); LCD_WR_DATA(0x0D);
    LCD_WR_REG(0xB102); LCD_WR_DATA(0x0D);
    //#Power Control for
    //VCL
    LCD_WR_REG(0xB800); LCD_WR_DATA(0x24);
    LCD_WR_REG(0xB801); LCD_WR_DATA(0x24);
    LCD_WR_REG(0xB802); LCD_WR_DATA(0x24);

    LCD_WR_REG(0xB200); LCD_WR_DATA(0x00);
    LCD_WR_REG(0xB201); LCD_WR_DATA(0x00);
    LCD_WR_REG(0xB202); LCD_WR_DATA(0x00);
```

```
    //# VGH: Clamp Enable); LCD_WR_DATA(
    LCD_WR_REG(0xB900); LCD_WR_DATA(0x24);
    LCD_WR_REG(0xB901); LCD_WR_DATA(0x24);
    LCD_WR_REG(0xB902); LCD_WR_DATA(0x24);

    LCD_WR_REG(0xB300); LCD_WR_DATA(0x05);
    LCD_WR_REG(0xB301); LCD_WR_DATA(0x05);
    LCD_WR_REG(0xB302); LCD_WR_DATA(0x05);

    ///LCD_WR_REG(0xBF00); LCD_WR_DATA(0x01);

    //# VGL(LVGL):
    LCD_WR_REG(0xBA00); LCD_WR_DATA(0x34);
    LCD_WR_REG(0xBA01); LCD_WR_DATA(0x34);
    LCD_WR_REG(0xBA02); LCD_WR_DATA(0x34);
    //# VGL_REG(VGLO)
    LCD_WR_REG(0xB500); LCD_WR_DATA(0x0B);
    LCD_WR_REG(0xB501); LCD_WR_DATA(0x0B);
    LCD_WR_REG(0xB502); LCD_WR_DATA(0x0B);
    //# VGMP/VGSP:
    LCD_WR_REG(0xBC00); LCD_WR_DATA(0X00);
    LCD_WR_REG(0xBC01); LCD_WR_DATA(0x78);
    LCD_WR_REG(0xBC02); LCD_WR_DATA(0X00);
    //# VGMN/VGSN
    LCD_WR_REG(0xBD00); LCD_WR_DATA(0x00);
    LCD_WR_REG(0xBD01); LCD_WR_DATA(0x78);
    LCD_WR_REG(0xBD02); LCD_WR_DATA(0x00);
    //# VCOM=-0.1
    LCD_WR_REG(0xBE00); LCD_WR_DATA(0x00);
    LCD_WR_REG(0xBE01); LCD_WR_DATA(0x50);//4f
    //  VCOMH+0x01;
//#R+
    LCD_WR_REG(0xD100); LCD_WR_DATA(0x00);
    LCD_WR_REG(0xD101); LCD_WR_DATA(0x37);
    LCD_WR_REG(0xD102); LCD_WR_DATA(0x00);
    LCD_WR_REG(0xD103); LCD_WR_DATA(0x52);
    LCD_WR_REG(0xD104); LCD_WR_DATA(0x00);
    LCD_WR_REG(0xD105); LCD_WR_DATA(0x7B);
    LCD_WR_REG(0xD106); LCD_WR_DATA(0x00);
    LCD_WR_REG(0xD107); LCD_WR_DATA(0x99);
    LCD_WR_REG(0xD108); LCD_WR_DATA(0x00);
    LCD_WR_REG(0xD109); LCD_WR_DATA(0xB1);
    LCD_WR_REG(0xD10A); LCD_WR_DATA(0x00);
    LCD_WR_REG(0xD10B); LCD_WR_DATA(0xD2);
    LCD_WR_REG(0xD10C); LCD_WR_DATA(0x00);
```

```
LCD_WR_REG(0xD10D); LCD_WR_DATA(0xF6);
LCD_WR_REG(0xD10E); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD10F); LCD_WR_DATA(0x27);
LCD_WR_REG(0xD110); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD111); LCD_WR_DATA(0x4E);
LCD_WR_REG(0xD112); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD113); LCD_WR_DATA(0x8C);
LCD_WR_REG(0xD114); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD115); LCD_WR_DATA(0xBE);
LCD_WR_REG(0xD116); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD117); LCD_WR_DATA(0x0B);
LCD_WR_REG(0xD118); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD119); LCD_WR_DATA(0x48);
LCD_WR_REG(0xD11A); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD11B); LCD_WR_DATA(0x4A);
LCD_WR_REG(0xD11C); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD11D); LCD_WR_DATA(0x7E);
LCD_WR_REG(0xD11E); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD11F); LCD_WR_DATA(0xBC);
LCD_WR_REG(0xD120); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD121); LCD_WR_DATA(0xE1);
LCD_WR_REG(0xD122); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD123); LCD_WR_DATA(0x10);
LCD_WR_REG(0xD124); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD125); LCD_WR_DATA(0x31);
LCD_WR_REG(0xD126); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD127); LCD_WR_DATA(0x5A);
LCD_WR_REG(0xD128); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD129); LCD_WR_DATA(0x73);
LCD_WR_REG(0xD12A); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD12B); LCD_WR_DATA(0x94);
LCD_WR_REG(0xD12C); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD12D); LCD_WR_DATA(0x9F);
LCD_WR_REG(0xD12E); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD12F); LCD_WR_DATA(0xB3);
LCD_WR_REG(0xD130); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD131); LCD_WR_DATA(0xB9);
LCD_WR_REG(0xD132); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD133); LCD_WR_DATA(0xC1);
//#G+
LCD_WR_REG(0xD200); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD201); LCD_WR_DATA(0x37);
LCD_WR_REG(0xD202); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD203); LCD_WR_DATA(0x52);
LCD_WR_REG(0xD204); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD205); LCD_WR_DATA(0x7B);
```

```
LCD_WR_REG(0xD206); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD207); LCD_WR_DATA(0x99);
LCD_WR_REG(0xD208); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD209); LCD_WR_DATA(0xB1);
LCD_WR_REG(0xD20A); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD20B); LCD_WR_DATA(0xD2);
LCD_WR_REG(0xD20C); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD20D); LCD_WR_DATA(0xF6);
LCD_WR_REG(0xD20E); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD20F); LCD_WR_DATA(0x27);
LCD_WR_REG(0xD210); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD211); LCD_WR_DATA(0x4E);
LCD_WR_REG(0xD212); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD213); LCD_WR_DATA(0x8C);
LCD_WR_REG(0xD214); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD215); LCD_WR_DATA(0xBE);
LCD_WR_REG(0xD216); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD217); LCD_WR_DATA(0x0B);
LCD_WR_REG(0xD218); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD219); LCD_WR_DATA(0x48);
LCD_WR_REG(0xD21A); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD21B); LCD_WR_DATA(0x4A);
LCD_WR_REG(0xD21C); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD21D); LCD_WR_DATA(0x7E);
LCD_WR_REG(0xD21E); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD21F); LCD_WR_DATA(0xBC);
LCD_WR_REG(0xD220); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD221); LCD_WR_DATA(0xE1);
LCD_WR_REG(0xD222); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD223); LCD_WR_DATA(0x10);
LCD_WR_REG(0xD224); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD225); LCD_WR_DATA(0x31);
LCD_WR_REG(0xD226); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD227); LCD_WR_DATA(0x5A);
LCD_WR_REG(0xD228); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD229); LCD_WR_DATA(0x73);
LCD_WR_REG(0xD22A); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD22B); LCD_WR_DATA(0x94);
LCD_WR_REG(0xD22C); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD22D); LCD_WR_DATA(0x9F);
LCD_WR_REG(0xD22E); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD22F); LCD_WR_DATA(0xB3);
LCD_WR_REG(0xD230); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD231); LCD_WR_DATA(0xB9);
LCD_WR_REG(0xD232); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD233); LCD_WR_DATA(0xC1);
```

```
//#B+
LCD_WR_REG(0xD300); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD301); LCD_WR_DATA(0x37);
LCD_WR_REG(0xD302); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD303); LCD_WR_DATA(0x52);
LCD_WR_REG(0xD304); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD305); LCD_WR_DATA(0x7B);
LCD_WR_REG(0xD306); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD307); LCD_WR_DATA(0x99);
LCD_WR_REG(0xD308); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD309); LCD_WR_DATA(0xB1);
LCD_WR_REG(0xD30A); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD30B); LCD_WR_DATA(0xD2);
LCD_WR_REG(0xD30C); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD30D); LCD_WR_DATA(0xF6);
LCD_WR_REG(0xD30E); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD30F); LCD_WR_DATA(0x27);
LCD_WR_REG(0xD310); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD311); LCD_WR_DATA(0x4E);
LCD_WR_REG(0xD312); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD313); LCD_WR_DATA(0x8C);
LCD_WR_REG(0xD314); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD315); LCD_WR_DATA(0xBE);
LCD_WR_REG(0xD316); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD317); LCD_WR_DATA(0x0B);
LCD_WR_REG(0xD318); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD319); LCD_WR_DATA(0x48);
LCD_WR_REG(0xD31A); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD31B); LCD_WR_DATA(0x4A);
LCD_WR_REG(0xD31C); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD31D); LCD_WR_DATA(0x7E);
LCD_WR_REG(0xD31E); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD31F); LCD_WR_DATA(0xBC);
LCD_WR_REG(0xD320); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD321); LCD_WR_DATA(0xE1);
LCD_WR_REG(0xD322); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD323); LCD_WR_DATA(0x10);
LCD_WR_REG(0xD324); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD325); LCD_WR_DATA(0x31);
LCD_WR_REG(0xD326); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD327); LCD_WR_DATA(0x5A);
LCD_WR_REG(0xD328); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD329); LCD_WR_DATA(0x73);
LCD_WR_REG(0xD32A); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD32B); LCD_WR_DATA(0x94);
LCD_WR_REG(0xD32C); LCD_WR_DATA(0x03);
```

```
LCD_WR_REG(0xD32D); LCD_WR_DATA(0x9F);
LCD_WR_REG(0xD32E); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD32F); LCD_WR_DATA(0xB3);
LCD_WR_REG(0xD330); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD331); LCD_WR_DATA(0xB9);
LCD_WR_REG(0xD332); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD333); LCD_WR_DATA(0xC1);

//#R-////////////////////////////////////////
LCD_WR_REG(0xD400); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD401); LCD_WR_DATA(0x37);
LCD_WR_REG(0xD402); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD403); LCD_WR_DATA(0x52);
LCD_WR_REG(0xD404); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD405); LCD_WR_DATA(0x7B);
LCD_WR_REG(0xD406); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD407); LCD_WR_DATA(0x99);
LCD_WR_REG(0xD408); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD409); LCD_WR_DATA(0xB1);
LCD_WR_REG(0xD40A); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD40B); LCD_WR_DATA(0xD2);
LCD_WR_REG(0xD40C); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD40D); LCD_WR_DATA(0xF6);
LCD_WR_REG(0xD40E); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD40F); LCD_WR_DATA(0x27);
LCD_WR_REG(0xD410); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD411); LCD_WR_DATA(0x4E);
LCD_WR_REG(0xD412); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD413); LCD_WR_DATA(0x8C);
LCD_WR_REG(0xD414); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD415); LCD_WR_DATA(0xBE);
LCD_WR_REG(0xD416); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD417); LCD_WR_DATA(0x0B);
LCD_WR_REG(0xD418); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD419); LCD_WR_DATA(0x48);
LCD_WR_REG(0xD41A); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD41B); LCD_WR_DATA(0x4A);
LCD_WR_REG(0xD41C); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD41D); LCD_WR_DATA(0x7E);
LCD_WR_REG(0xD41E); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD41F); LCD_WR_DATA(0xBC);
LCD_WR_REG(0xD420); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD421); LCD_WR_DATA(0xE1);
LCD_WR_REG(0xD422); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD423); LCD_WR_DATA(0x10);
LCD_WR_REG(0xD424); LCD_WR_DATA(0x03);
```

```
LCD_WR_REG(0xD425); LCD_WR_DATA(0x31);
LCD_WR_REG(0xD426); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD427); LCD_WR_DATA(0x5A);
LCD_WR_REG(0xD428); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD429); LCD_WR_DATA(0x73);
LCD_WR_REG(0xD42A); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD42B); LCD_WR_DATA(0x94);
LCD_WR_REG(0xD42C); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD42D); LCD_WR_DATA(0x9F);
LCD_WR_REG(0xD42E); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD42F); LCD_WR_DATA(0xB3);
LCD_WR_REG(0xD430); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD431); LCD_WR_DATA(0xB9);
LCD_WR_REG(0xD432); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD433); LCD_WR_DATA(0xC1);

//#G-//////////////////////////////////////////////
LCD_WR_REG(0xD500); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD501); LCD_WR_DATA(0x37);
LCD_WR_REG(0xD502); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD503); LCD_WR_DATA(0x52);
LCD_WR_REG(0xD504); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD505); LCD_WR_DATA(0x7B);
LCD_WR_REG(0xD506); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD507); LCD_WR_DATA(0x99);
LCD_WR_REG(0xD508); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD509); LCD_WR_DATA(0xB1);
LCD_WR_REG(0xD50A); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD50B); LCD_WR_DATA(0xD2);
LCD_WR_REG(0xD50C); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD50D); LCD_WR_DATA(0xF6);
LCD_WR_REG(0xD50E); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD50F); LCD_WR_DATA(0x27);
LCD_WR_REG(0xD510); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD511); LCD_WR_DATA(0x4E);
LCD_WR_REG(0xD512); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD513); LCD_WR_DATA(0x8C);
LCD_WR_REG(0xD514); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD515); LCD_WR_DATA(0xBE);
LCD_WR_REG(0xD516); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD517); LCD_WR_DATA(0x0B);
LCD_WR_REG(0xD518); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD519); LCD_WR_DATA(0x48);
LCD_WR_REG(0xD51A); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD51B); LCD_WR_DATA(0x4A);
LCD_WR_REG(0xD51C); LCD_WR_DATA(0x02);
```

```
LCD_WR_REG(0xD51D); LCD_WR_DATA(0x7E);
LCD_WR_REG(0xD51E); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD51F); LCD_WR_DATA(0xBC);
LCD_WR_REG(0xD520); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD521); LCD_WR_DATA(0xE1);
LCD_WR_REG(0xD522); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD523); LCD_WR_DATA(0x10);
LCD_WR_REG(0xD524); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD525); LCD_WR_DATA(0x31);
LCD_WR_REG(0xD526); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD527); LCD_WR_DATA(0x5A);
LCD_WR_REG(0xD528); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD529); LCD_WR_DATA(0x73);
LCD_WR_REG(0xD52A); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD52B); LCD_WR_DATA(0x94);
LCD_WR_REG(0xD52C); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD52D); LCD_WR_DATA(0x9F);
LCD_WR_REG(0xD52E); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD52F); LCD_WR_DATA(0xB3);
LCD_WR_REG(0xD530); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD531); LCD_WR_DATA(0xB9);
LCD_WR_REG(0xD532); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD533); LCD_WR_DATA(0xC1);
//#B-////////////////////////////////
LCD_WR_REG(0xD600); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD601); LCD_WR_DATA(0x37);
LCD_WR_REG(0xD602); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD603); LCD_WR_DATA(0x52);
LCD_WR_REG(0xD604); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD605); LCD_WR_DATA(0x7B);
LCD_WR_REG(0xD606); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD607); LCD_WR_DATA(0x99);
LCD_WR_REG(0xD608); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD609); LCD_WR_DATA(0xB1);
LCD_WR_REG(0xD60A); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD60B); LCD_WR_DATA(0xD2);
LCD_WR_REG(0xD60C); LCD_WR_DATA(0x00);
LCD_WR_REG(0xD60D); LCD_WR_DATA(0xF6);
LCD_WR_REG(0xD60E); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD60F); LCD_WR_DATA(0x27);
LCD_WR_REG(0xD610); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD611); LCD_WR_DATA(0x4E);
LCD_WR_REG(0xD612); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD613); LCD_WR_DATA(0x8C);
LCD_WR_REG(0xD614); LCD_WR_DATA(0x01);
LCD_WR_REG(0xD615); LCD_WR_DATA(0xBE);
```

```
LCD_WR_REG(0xD616); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD617); LCD_WR_DATA(0x0B);
LCD_WR_REG(0xD618); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD619); LCD_WR_DATA(0x48);
LCD_WR_REG(0xD61A); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD61B); LCD_WR_DATA(0x4A);
LCD_WR_REG(0xD61C); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD61D); LCD_WR_DATA(0x7E);
LCD_WR_REG(0xD61E); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD61F); LCD_WR_DATA(0xBC);
LCD_WR_REG(0xD620); LCD_WR_DATA(0x02);
LCD_WR_REG(0xD621); LCD_WR_DATA(0xE1);
LCD_WR_REG(0xD622); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD623); LCD_WR_DATA(0x10);
LCD_WR_REG(0xD624); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD625); LCD_WR_DATA(0x31);
LCD_WR_REG(0xD626); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD627); LCD_WR_DATA(0x5A);
LCD_WR_REG(0xD628); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD629); LCD_WR_DATA(0x73);
LCD_WR_REG(0xD62A); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD62B); LCD_WR_DATA(0x94);
LCD_WR_REG(0xD62C); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD62D); LCD_WR_DATA(0x9F);
LCD_WR_REG(0xD62E); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD62F); LCD_WR_DATA(0xB3);
LCD_WR_REG(0xD630); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD631); LCD_WR_DATA(0xB9);
LCD_WR_REG(0xD632); LCD_WR_DATA(0x03);
LCD_WR_REG(0xD633); LCD_WR_DATA(0xC1);



//#Enable Page0
LCD_WR_REG(0xF000); LCD_WR_DATA(0x55);
LCD_WR_REG(0xF001); LCD_WR_DATA(0xAA);
LCD_WR_REG(0xF002); LCD_WR_DATA(0x52);
LCD_WR_REG(0xF003); LCD_WR_DATA(0x08);
LCD_WR_REG(0xF004); LCD_WR_DATA(0x00);
//# RGB I/F Setting
LCD_WR_REG(0xB000); LCD_WR_DATA(0x08);
LCD_WR_REG(0xB001); LCD_WR_DATA(0x05);
LCD_WR_REG(0xB002); LCD_WR_DATA(0x02);
LCD_WR_REG(0xB003); LCD_WR_DATA(0x05);
LCD_WR_REG(0xB004); LCD_WR_DATA(0x02);
//## SDT:
```

```
LCD_WR_REG(0xB600); LCD_WR_DATA(0x08);
LCD_WR_REG(0xB500); LCD_WR_DATA(0x50);//0x6b ???? 480x854        0x50 ???? 480x800

//## Gate EQ:
LCD_WR_REG(0xB700); LCD_WR_DATA(0x00);
LCD_WR_REG(0xB701); LCD_WR_DATA(0x00);

//## Source EQ:
LCD_WR_REG(0xB800); LCD_WR_DATA(0x01);
LCD_WR_REG(0xB801); LCD_WR_DATA(0x05);
LCD_WR_REG(0xB802); LCD_WR_DATA(0x05);
LCD_WR_REG(0xB803); LCD_WR_DATA(0x05);

//# Inversion: Column inversion (NVT)
LCD_WR_REG(0xBC00); LCD_WR_DATA(0x00);
LCD_WR_REG(0xBC01); LCD_WR_DATA(0x00);
LCD_WR_REG(0xBC02); LCD_WR_DATA(0x00);

//# BOE's Setting(default)
LCD_WR_REG(0xCC00); LCD_WR_DATA(0x03);
LCD_WR_REG(0xCC01); LCD_WR_DATA(0x00);
LCD_WR_REG(0xCC02); LCD_WR_DATA(0x00);

//# Display Timing:
LCD_WR_REG(0xBD00); LCD_WR_DATA(0x01);
LCD_WR_REG(0xBD01); LCD_WR_DATA(0x84);
LCD_WR_REG(0xBD02); LCD_WR_DATA(0x07);
LCD_WR_REG(0xBD03); LCD_WR_DATA(0x31);
LCD_WR_REG(0xBD04); LCD_WR_DATA(0x00);

LCD_WR_REG(0xBA00); LCD_WR_DATA(0x01);

LCD_WR_REG(0xFF00); LCD_WR_DATA(0xAA);
LCD_WR_REG(0xFF01); LCD_WR_DATA(0x55);
LCD_WR_REG(0xFF02); LCD_WR_DATA(0x25);
LCD_WR_REG(0xFF03); LCD_WR_DATA(0x01);

LCD_WR_REG(0x3500); LCD_WR_DATA(0x00);
LCD_WR_REG(0x3600); LCD_WR_DATA(0x00);
LCD_WR_REG(0x3a00); LCD_WR_DATA(0x55);  ////55=16?/////66=18?
LCD_WR_REG(0x1100);
delay_ms(120);
LCD_WR_REG(0x2900 );
LCD_WR_REG(0x2c00);
LCD_direction(USE_HORIZONTAL);//设置 LCD 显示方向
```

```
//   LCD_Clear(WHITE);//清全屏白色
}

/*****************************************************************************
 * @name       :void LCD_SetWindows(u16 xStar, u16 yStar,u16 xEnd,u16 yEnd)
 * @date       :2018-08-09
 * @function   :Setting LCD display window
 * @parameters :xStar:the bebinning x coordinate of the LCD display window
                yStar:the bebinning y coordinate of the LCD display window
                xEnd:the endning x coordinate of the LCD display window
                yEnd:the endning y coordinate of the LCD display window
 * @retvalue   :None
******************************************************************************/
void LCD_SetWindows(u16 xStar, u16 yStar,u16 xEnd,u16 yEnd)
{
    LCD_WR_REG(lcddev.setxcmd);LCD_WR_DATA(xStar>>8);
    LCD_WR_REG(lcddev.setxcmd+1);LCD_WR_DATA(xStar&0XFF);
    LCD_WR_REG(lcddev.setxcmd+2);LCD_WR_DATA(xEnd>>8);
    LCD_WR_REG(lcddev.setxcmd+3);LCD_WR_DATA(xEnd&0XFF);
    LCD_WR_REG(lcddev.setycmd);LCD_WR_DATA(yStar>>8);
    LCD_WR_REG(lcddev.setycmd+1);LCD_WR_DATA(yStar&0XFF);
    LCD_WR_REG(lcddev.setycmd+2);LCD_WR_DATA(yEnd>>8);
    LCD_WR_REG(lcddev.setycmd+3);LCD_WR_DATA(yEnd&0XFF);

    LCD_WriteRAM_Prepare();//开始写入GRAM
}

/*****************************************************************************
 * @name       :void LCD_SetCursor(u16 Xpos, u16 Ypos)
 * @date       :2018-08-09
 * @function   :Set coordinate value
 * @parameters :Xpos:the  x coordinate of the pixel
                Ypos:the  y coordinate of the pixel
 * @retvalue   :None
******************************************************************************/
void LCD_SetCursor(u16 Xpos, u16 Ypos)
{
    LCD_SetWindows(Xpos,Ypos,Xpos,Ypos);
}

/*****************************************************************************
 * @name       :void LCD_direction(u8 direction)
 * @date       :2018-08-09
 * @function   :Setting the display direction of LCD screen
 * @parameters :direction:0-0 degree
                         1-90 degree
```

```
                                                    2-180 degree
                                                    3-270 degree
 * @retvalue    :None
******************************************************************************/
void LCD_direction(u8 direction)
{
     lcddev.setxcmd=0x2A00;
     lcddev.setycmd=0x2B00;
     lcddev.wramcmd=0x2C00;
     lcddev.rramcmd=0x2E00;
     switch(direction){
          case 0:
               lcddev.width=LCD_W;
               lcddev.height=LCD_H;
               LCD_WriteReg(0x3600,0x00);
          break;
          case 1:
               lcddev.width=LCD_H;
               lcddev.height=LCD_W;
               LCD_WriteReg(0x3600,(1<<5)|(1<<6));
          break;
          case 2:
               lcddev.width=LCD_W;
               lcddev.height=LCD_H;
               LCD_WriteReg(0x3600,(1<<7)|(1<<6));
          break;
          case 3:
               lcddev.width=LCD_H;
               lcddev.height=LCD_W;
               LCD_WriteReg(0x3600,(1<<7)|(1<<5));
          break;
          default:break;
     }
}

/******************************************************************************
 * @name        :u16 LCD_Read_ID(void)
 * @date        :2018-11-13
 * @function    :Read ID
 * @parameters  :None
 * @retvalue    :ID value
******************************************************************************/
u16 LCD_Read_ID(void)
{
     u16 val;
     LCD_WR_REG(0xF000);
```

```
    LCD_WR_DATA(0x55);
    LCD_WR_REG(0xF001);
    LCD_WR_DATA(0xAA);
    LCD_WR_REG(0xF002);
    LCD_WR_DATA(0x52);
    LCD_WR_REG(0xF003);
    LCD_WR_DATA(0x08);
    LCD_WR_REG(0xF004);
    LCD_WR_DATA(0x01);

   LCD_ReadReg(0xC500,&lcddev.id,1);//读回 0X55
    lcddev.id<<=8;
    LCD_ReadReg(0xC501,&val,1);//读回 0X10
    lcddev.id |= val;
    return lcddev.id;
}
/*****************************************************************
 * @name       :void GUI_DrawPoint(u16 x,u16 y,u16 color)
 * @date       :2018-08-09
 * @function   :draw a point in LCD screen
 * @parameters :x:the x coordinate of the point
                y:the y coordinate of the point
                                color:the color value of the point
 * @retvalue   :None
*****************************************************************/
void GUI_DrawPoint(u16 x,u16 y,u16 color)
{
    LCD_SetCursor(x,y);//设置光标位置
    Lcd_WriteData_16Bit(color);
}

/*****************************************************************
 * @name       :void LCD_Fill(u16 sx,u16 sy,u16 ex,u16 ey,u16 color)
 * @date       :2018-08-09
 * @function   :fill the specified area
 * @parameters :sx:the bebinning x coordinate of the specified area
                sy:the bebinning y coordinate of the specified area
                                ex:the ending x coordinate of the specified area
                                ey:the ending y coordinate of the specified area
                                color:the filled color value
 * @retvalue   :None
*****************************************************************/
void LCD_Fill(u16 sx,u16 sy,u16 ex,u16 ey,u16 color)
{
    u16 i,j;
    u16 width=ex-sx+1;          //得到填充的宽度
```

```
    u16 height=ey-sy+1;          //高度
    LCD_SetWindows(sx,sy,ex,ey);//设置显示窗口
    for(i=0;i<height;i++)
    {
        for(j=0;j<width;j++)
        Lcd_WriteData_16Bit(color); //写入数据
    }
    LCD_SetWindows(0,0,lcddev.width-1,lcddev.height-1);//恢复窗口设置为全屏
}

/*****************************************************************
 * @name       :void LCD_DrawLine(u16 x1, u16 y1, u16 x2, u16 y2)
 * @date       :2018-08-09
 * @function   :Draw a line between two points
 * @parameters :x1:the bebinning x coordinate of the line
                y1:the bebinning y coordinate of the line
                                  x2:the ending x coordinate of the line
                                  y2:the ending y coordinate of the line
 * @retvalue   :None
*****************************************************************/
void LCD_DrawLine(u16 x1, u16 y1, u16 x2, u16 y2)
{
    u16 t;
    int xerr=0,yerr=0,delta_x,delta_y,distance;
    int incx,incy,uRow,uCol;

    delta_x=x2-x1; //计算坐标增量
    delta_y=y2-y1;
    uRow=x1;
    uCol=y1;
    if(delta_x>0)incx=1; //设置单步方向
    else if(delta_x==0)incx=0;//垂直线
    else {incx=-1;delta_x=-delta_x;}
    if(delta_y>0)incy=1;
    else if(delta_y==0)incy=0;//水平线
    else{incy=-1;delta_y=-delta_y;}
    if( delta_x>delta_y)distance=delta_x; //选取基本增量坐标轴
    else distance=delta_y;
    for(t=0;t<=distance+1;t++ )//画线输出
    {
        LCD_DrawPoint(uRow,uCol);//画点
        xerr+=delta_x ;
        yerr+=delta_y ;
        if(xerr>distance)
        {
            xerr-=distance;
```

```
            uRow+=incx;
        }
        if(yerr>distance)
        {
            yerr-=distance;
            uCol+=incy;
        }
    }
}

/*****************************************************************************
 * @name       :void LCD_DrawRectangle(u16 x1, u16 y1, u16 x2, u16 y2)
 * @date       :2018-08-09
 * @function   :Draw a rectangle
 * @parameters :x1:the bebinning x coordinate of the rectangle
               y1:the bebinning y coordinate of the rectangle
                                  x2:the ending x coordinate of the rectangle
                                  y2:the ending y coordinate of the rectangle
 * @retvalue   :None
*****************************************************************************/
void LCD_DrawRectangle(u16 x1, u16 y1, u16 x2, u16 y2)
{
    LCD_DrawLine(x1,y1,x2,y1);
    LCD_DrawLine(x1,y1,x1,y2);
    LCD_DrawLine(x1,y2,x2,y2);
    LCD_DrawLine(x2,y1,x2,y2);
}

/*****************************************************************************
 * @name       :void LCD_DrawFillRectangle(u16 x1, u16 y1, u16 x2, u16 y2)
 * @date       :2018-08-09
 * @function   :Filled a rectangle
 * @parameters :x1:the bebinning x coordinate of the filled rectangle
               y1:the bebinning y coordinate of the filled rectangle
                                  x2:the ending x coordinate of the filled rectangle
                                  y2:the ending y coordinate of the filled rectangle
 * @retvalue   :None
*****************************************************************************/
void LCD_DrawFillRectangle(u16 x1, u16 y1, u16 x2, u16 y2)
{
    LCD_Fill(x1,y1,x2,y2,POINT_COLOR);
}

/*****************************************************************************
 * @name       :void _draw_circle_8(int xc, int yc, int x, int y, u16 c)
 * @date       :2018-08-09
```

```
 * @function   :8 symmetry circle drawing algorithm (internal call)
 * @parameters :xc:the x coordinate of the Circular center
                yc:the y coordinate of the Circular center
                                x:the x coordinate relative to the Circular center
                                y:the y coordinate relative to the Circular center
                                c:the color value of the circle
 * @retvalue   :None
******************************************************************************/
void _draw_circle_8(int xc, int yc, int x, int y, u16 c)
{
    GUI_DrawPoint(xc + x, yc + y, c);

    GUI_DrawPoint(xc - x, yc + y, c);

    GUI_DrawPoint(xc + x, yc - y, c);

    GUI_DrawPoint(xc - x, yc - y, c);

    GUI_DrawPoint(xc + y, yc + x, c);

    GUI_DrawPoint(xc - y, yc + x, c);

    GUI_DrawPoint(xc + y, yc - x, c);

    GUI_DrawPoint(xc - y, yc - x, c);
}

/******************************************************************************
 * @name       :void gui_circle(int xc, int yc,u16 c,int r, int fill)
 * @date       :2018-08-09
 * @function   :Draw a circle of specified size at a specified location
 * @parameters :xc:the x coordinate of the Circular center
                yc:the y coordinate of the Circular center
                                r:Circular radius
                                fill:1-filling,0-no filling
 * @retvalue   :None
******************************************************************************/
void gui_circle(int xc, int yc,u16 c,int r, int fill)
{
    int x = 0, y = r, yi, d;

    d = 3 - 2 * r;


    if (fill)
    {
```

```
        // 如果填充（画实心圆）
        while (x <= y) {
            for (yi = x; yi <= y; yi++)
                _draw_circle_8(xc, yc, x, yi, c);

            if (d < 0) {
                d = d + 4 * x + 6;
            } else {
                d = d + 4 * (x - y) + 10;
                y--;
            }
            x++;
        }
    } else
    {
        // 如果不填充（画空心圆）
        while (x <= y) {
            _draw_circle_8(xc, yc, x, y, c);
            if (d < 0) {
                d = d + 4 * x + 6;
            } else {
                d = d + 4 * (x - y) + 10;
                y--;
            }
            x++;
        }
    }
}

/************************************************************************
 * @name        :void Draw_Triangel(u16 x0,u16 y0,u16 x1,u16 y1,u16 x2,u16 y2)
 * @date        :2018-08-09
 * @function    :Draw a triangle at a specified position
 * @parameters :x0:the bebinning x coordinate of the triangular edge
                y0:the bebinning y coordinate of the triangular edge
                                x1:the vertex x coordinate of the triangular
                                y1:the vertex y coordinate of the triangular
                                x2:the ending x coordinate of the triangular edge
                                y2:the ending y coordinate of the triangular edge
 * @retvalue    :None
************************************************************************/
void Draw_Triangel(u16 x0,u16 y0,u16 x1,u16 y1,u16 x2,u16 y2)
{
    LCD_DrawLine(x0,y0,x1,y1);
    LCD_DrawLine(x1,y1,x2,y2);
    LCD_DrawLine(x2,y2,x0,y0);
```

```
}

static void _swap(u16 *a, u16 *b)
{
    u16 tmp;
  tmp = *a;
    *a = *b;
    *b = tmp;
}

/*************************************************************************
 * @name       :void Fill_Triangel(u16 x0,u16 y0,u16 x1,u16 y1,u16 x2,u16 y2)
 * @date       :2018-08-09
 * @function   :filling a triangle at a specified position
 * @parameters :x0:the bebinning x coordinate of the triangular edge
               y0:the bebinning y coordinate of the triangular edge
                                  x1:the vertex x coordinate of the triangular
                                  y1:the vertex y coordinate of the triangular
                                  x2:the ending x coordinate of the triangular edge
                                  y2:the ending y coordinate of the triangular edge
 * @retvalue   :None
*************************************************************************/
void Fill_Triangel(u16 x0,u16 y0,u16 x1,u16 y1,u16 x2,u16 y2)
{
    u16 a, b, y, last;
    int dx01, dy01, dx02, dy02, dx12, dy12;
    long sa = 0;
    long sb = 0;
    if (y0 > y1)
    {
    _swap(&y0,&y1);
        _swap(&x0,&x1);
    }
    if (y1 > y2)
    {
    _swap(&y2,&y1);
        _swap(&x2,&x1);
    }
  if (y0 > y1)
    {
    _swap(&y0,&y1);
        _swap(&x0,&x1);
  }
    if(y0 == y2)
    {
        a = b = x0;
```

```
        if(x1 < a)
    {
            a = x1;
    }
    else if(x1 > b)
    {
            b = x1;
    }
    if(x2 < a)
    {
            a = x2;
    }
        else if(x2 > b)
    {
            b = x2;
    }
        LCD_Fill(a,y0,b,y0,POINT_COLOR);
    return;
     }
     dx01 = x1 - x0;
     dy01 = y1 - y0;
     dx02 = x2 - x0;
     dy02 = y2 - y0;
     dx12 = x2 - x1;
     dy12 = y2 - y1;

     if(y1 == y2)
     {
        last = y1;
     }
  else
     {
        last = y1-1;
     }
     for(y=y0; y<=last; y++)
     {
        a = x0 + sa / dy01;
        b = x0 + sb / dy02;
        sa += dx01;
    sb += dx02;
    if(a > b)
    {
            _swap(&a,&b);
        }
        LCD_Fill(a,y,b,y,POINT_COLOR);
     }
```

```
    sa = dx12 * (y - y1);
    sb = dx02 * (y - y0);
    for(; y<=y2; y++)
    {
        a = x1 + sa / dy12;
        b = x0 + sb / dy02;
        sa += dx12;
        sb += dx02;
        if(a > b)
        {
            _swap(&a,&b);
        }
        LCD_Fill(a,y,b,y,POINT_COLOR);
    }
}

/*****************************************************************************
 * @name       :void LCD_ShowChar(u16 x,u16 y,u16 fc, u16 bc, u8 num,u8 size,u8 mode)
 * @date       :2018-08-09
 * @function   :Display a single English character
 * @parameters :x:the bebinning x coordinate of the Character display position
               y:the bebinning y coordinate of the Character display position
                 fc:the color value of display character
                 bc:the background color of display character
                 num:the ascii code of display character(0~94)
                 size:the size of display character
                 mode:0-no overlying,1-overlying
 * @retvalue   :None
*****************************************************************************/
void LCD_ShowChar(u16 x,u16 y,u16 fc, u16 bc, u8 num,u8 size,u8 mode)
{
    u8 temp1,temp2;
    u8 pos,t;
    u16 colortemp=POINT_COLOR;
    num=num-' ';//得到偏移后的值
    LCD_SetWindows(x,y,x+size/2-1,y+size-1);//设置单个文字显示窗口
    if(!mode) //非叠加方式
    {
        for(pos=0;pos<size;pos++)
        {
            if(size==12)temp1=asc2_1206[num][pos];//调用 1206 字体
            else if(size==16) temp1=asc2_1608[num][pos];        //调用 1608 字体
            else if(size==32) temp1=asc2_3216[num][pos*2];//调用 3216 字体
            else return;
            for(t=0;t<8;t++)
            {
```

```
            if(temp1&0x01)
             {
                  Lcd_WriteData_16Bit(fc);
                  delay_ms(10);
             }
             else Lcd_WriteData_16Bit(bc);
             temp1>>=1;
             y++;

        }

    }
}else//叠加方式
{
    for(pos=0;pos<size;pos++)
    {
         if(size==12)temp1=asc2_1206[num][pos];//调用 1206 字体
         else if(size==16)temp1=asc2_1608[num][pos];          //调用 1608 字体
         else if(size==32)
         {
            temp1=asc2_3216[num][pos*2];
            temp2=asc2_3216[num][pos*2+1];
         }//调用 3216 字体
         else return;
         for(t=0;t<8;t++)
         {
            POINT_COLOR=fc;
            if(temp1&0x01)
             {
                  LCD_DrawPoint(x+t,y+pos);//画一个点
            }
             temp1>>=1;
        }

         for(t=0;t<8;t++)
         {
              POINT_COLOR=fc;
            if(temp2&0x01)
             {
                  LCD_DrawPoint(x+8+t,y+pos);//画一个点
            }
             temp2>>=1;
        }
    }
}
POINT_COLOR=colortemp;
```

```
    LCD_SetWindows(0,0,lcddev.width-1,lcddev.height-1);//恢复窗口为全屏
}



/*****************************************************************************
 * @name       :void LCD_ShowString(u16 x,u16 y,u8 size,u8 *p,u8 mode)
 * @date       :2018-08-09
 * @function   :Display English string
 * @parameters :x:the bebinning x coordinate of the English string
                y:the bebinning y coordinate of the English string
                                    p:the start address of the English string
                                    size:the size of display character
                                    mode:0-no overlying,1-overlying
 * @retvalue   :None
*****************************************************************************/
void LCD_ShowString(u16 x,u16 y,u8 size,u8 *p,u8 mode)
{
    while((*p<='~')&&(*p>=' '))//判断是不是非法字符!
    {
        if(x>(lcddev.width-1)||y>(lcddev.height-1))
        return;
        LCD_ShowChar(x,y,POINT_COLOR,BACK_COLOR,*p,size,mode);
        x+=size/2;
        p++;
    }
}
/*****************************************************************************
 * @name       :u32 mypow(u8 m,u8 n)
 * @date       :2018-08-09
 * @function   :get the nth power of m (internal call)
 * @parameters :m:the multiplier
                n:the power
 * @retvalue   :the nth power of m
*****************************************************************************/
u32 mypow(u8 m,u8 n)
{
    u32 result=1;
    while(n--)result*=m;
    return result;
}

/*****************************************************************************
 * @name       :void LCD_ShowNum(u16 x,u16 y,u32 num,u8 len,u8 size)
 * @date       :2018-08-09
```

```
 * @function   :Display number
 * @parameters :x:the bebinning x coordinate of the number
                y:the bebinning y coordinate of the number
                                num:the number(0~4294967295)
                                len:the length of the display number
                                size:the size of display number
 * @retvalue   :None
******************************************************************************/
void LCD_ShowNum(u16 x,u16 y,u32 num,u8 len,u8 size)
{
    u8 t,temp;
    u8 enshow=0;
    for(t=0;t<len;t++)
    {
        temp=(num/mypow(10,len-t-1))%10;
        if(enshow==0&&t<(len-1))
        {
            if(temp==0)
            {
                LCD_ShowChar(x+(size/2)*t,y,POINT_COLOR,BACK_COLOR,' ',size,0);
                continue;
            }else enshow=1;

        }
        LCD_ShowChar(x+(size/2)*t,y,POINT_COLOR,BACK_COLOR,temp+'0',size,0);
    }
}

/******************************************************************************
 * @name       :void GUI_DrawFont16(u16 x, u16 y, u16 fc, u16 bc, u8 *s,u8 mode)
 * @date       :2018-08-09
 * @function   :Display a single 16x16 Chinese character
 * @parameters :x:the bebinning x coordinate of the Chinese character
                y:the bebinning y coordinate of the Chinese character
                                fc:the color value of Chinese character
                                bc:the background color of Chinese character
                                s:the start address of the Chinese character
                                mode:0-no overlying,1-overlying
 * @retvalue   :None
******************************************************************************/
void GUI_DrawFont16(u16 x, u16 y, u16 fc, u16 bc, u8 *s,u8 mode)
{
    u8 i,j;
    u16 k;
    u16 HZnum;
    u16 x0=x;
```

```
    HZnum=sizeof(tfont16)/sizeof(typFNT_GB16);     //自动统计汉字数目


    for (k=0;k<HZnum;k++)
    {
      if ((tfont16[k].Index[0]==*(s))&&(tfont16[k].Index[1]==*(s+1)))
      { LCD_SetWindows(x,y,x+16-1,y+16-1);
            for(i=0;i<16*2;i++)
            {
                 for(j=0;j<8;j++)
                {
                      if(!mode) //非叠加方式
                      {
                           if(tfont16[k].Msk[i]&(0x80>>j)) Lcd_WriteData_16Bit(fc);
                           else Lcd_WriteData_16Bit(bc);
                      }
                      else
                      {
                           POINT_COLOR=fc;
                           if(tfont16[k].Msk[i]&(0x80>>j)) LCD_DrawPoint(x,y);//画一个点
                           x++;
                           if((x-x0)==16)
                           {
                                x=x0;
                                y++;
                                break;
                           }
                      }
                }

            }


        }
        continue;  //查找到对应点阵字库立即退出，防止多个汉字重复取模带来影响
    }

    LCD_SetWindows(0,0,lcddev.width-1,lcddev.height-1);//恢复窗口为全屏
}

/*****************************************************************************
 * @name       :void GUI_DrawFont24(u16 x, u16 y, u16 fc, u16 bc, u8 *s,u8 mode)
 * @date       :2018-08-09
 * @function   :Display a single 24x24 Chinese character
 * @parameters :x:the bebinning x coordinate of the Chinese character
```

```
                y:the bebinning y coordinate of the Chinese character
                                fc:the color value of Chinese character
                                bc:the background color of Chinese character
                                s:the start address of the Chinese character
                                mode:0-no overlying,1-overlying
 * @retvalue    :None
*******************************************************************************/
void GUI_DrawFont24(u16 x, u16 y, u16 fc, u16 bc, u8 *s,u8 mode)
{
    u8 i,j;
    u16 k;
    u16 HZnum;
    u16 x0=x;
    HZnum=sizeof(tfont24)/sizeof(typFNT_GB24);     //自动统计汉字数目

            for (k=0;k<HZnum;k++)
            {
              if ((tfont24[k].Index[0]==*(s))&&(tfont24[k].Index[1]==*(s+1)))
              { LCD_SetWindows(x,y,x+24-1,y+24-1);
                for(i=0;i<24*3;i++)
                {
                        for(j=0;j<8;j++)
                        {
                            if(!mode) //非叠加方式
                            {
                                if(tfont24[k].Msk[i]&(0x80>>j)) Lcd_WriteData_16Bit(fc);
                                else Lcd_WriteData_16Bit(bc);
                            }
                        else
                        {
                            POINT_COLOR=fc;
                            if(tfont24[k].Msk[i]&(0x80>>j)) LCD_DrawPoint(x,y);//画一个点
                            x++;
                            if((x-x0)==24)
                            {
                                x=x0;
                                y++;
                                break;
                            }
                        }
                    }
                }


            }
            continue;  //查找到对应点阵字库立即退出，防止多个汉字重复取模带来影响
```

```
            }

    LCD_SetWindows(0,0,lcddev.width-1,lcddev.height-1);//恢复窗口为全屏
}

/*****************************************************************************
 * @name       :void GUI_DrawFont32(u16 x, u16 y, u16 fc, u16 bc, u8 *s,u8 mode)
 * @date       :2018-08-09
 * @function   :Display a single 32x32 Chinese character
 * @parameters :x:the bebinning x coordinate of the Chinese character
                y:the bebinning y coordinate of the Chinese character
                                    fc:the color value of Chinese character
                                    bc:the background color of Chinese character
                                    s:the start address of the Chinese character
                                    mode:0-no overlying,1-overlying
 * @retvalue   :None
*****************************************************************************/
void GUI_DrawFont32(u16 x, u16 y, u16 fc, u16 bc, u8 *s,u8 mode)
{
    u8 i,j;
    u16 k;
    u16 HZnum;
    u16 x0=x;
    HZnum=sizeof(tfont32)/sizeof(typFNT_GB32);     //自动统计汉字数目
    for (k=0;k<HZnum;k++)
            {
             if ((tfont32[k].Index[0]==*(s))&&(tfont32[k].Index[1]==*(s+1)))
             { LCD_SetWindows(x,y,x+32-1,y+32-1);
                    for(i=0;i<32*4;i++)
                    {
                         for(j=0;j<8;j++)
                     {
                              if(!mode) //非叠加方式
                              {
                                   if(tfont32[k].Msk[i]&(0x80>>j)) Lcd_WriteData_16Bit(fc);
                                   else Lcd_WriteData_16Bit(bc);
                              }
                              else
                              {
                                   POINT_COLOR=fc;
                                   if(tfont32[k].Msk[i]&(0x80>>j)) LCD_DrawPoint(x,y);//画一个点
                                   x++;
                                   if((x-x0)==32)
                                   {
                                        x=x0;
                                        y++;
```

```
                                    break;
                                }
                            }
                        }
                    }


                }
                continue;  //查找到对应点阵字库立即退出，防止多个汉字重复取模带来影响
            }

    LCD_SetWindows(0,0,lcddev.width-1,lcddev.height-1);//恢复窗口为全屏
}


void GUI_DrawFont48(u16 x, u16 y, u16 fc, u16 bc, u8 *s,u8 mode)
{
    u16 i,j;
    u16 k;
    u16 HZnum;
    u16 x0=x;
    HZnum=sizeof(tfont48)/sizeof(typFNT_GB48);     //自动统计汉字数目
    for (k=0;k<HZnum;k++)
            {
              if ((tfont48[k].Index[0]==*(s))&&(tfont48[k].Index[1]==*(s+1)))
              { LCD_SetWindows(x,y,x+48-1,y+48-1);
                for(i=0;i<48*6;i++)
                {
                    for(j=0;j<8;j++)
                {
                        if(!mode) //非叠加方式
                        {
                            if(tfont48[k].Msk[i]&(0x80>>j)) Lcd_WriteData_16Bit(fc);
                            else Lcd_WriteData_16Bit(bc);
                        }
                        else
                        {
                            POINT_COLOR=fc;
                            if(tfont48[k].Msk[i]&(0x80>>j)) LCD_DrawPoint(x,y);//画一个点
                            x++;
                            if((x-x0)==48)
                            {
                                x=x0;
                                y++;
                                break;
                            }
```

```
                            }
                        }
                    }


                }
                continue;  //查找到对应点阵字库立即退出，防止多个汉字重复取模带来影响
            }

    LCD_SetWindows(0,0,lcddev.width-1,lcddev.height-1);//恢复窗口为全屏
}


/*****************************************************************************
 * @name       :void Show_Str(u16 x, u16 y, u16 fc, u16 bc, u8 *str,u8 size,u8 mode)
 * @date       :2018-08-09
 * @function   :Display Chinese and English strings
 * @parameters :x:the bebinning x coordinate of the Chinese and English strings
                y:the bebinning y coordinate of the Chinese and English strings
                fc:the color value of Chinese and English strings
                bc:the background color of Chinese and English strings
                str:the start address of the Chinese and English strings
                size:the size of Chinese and English strings
                mode:0-no overlying,1-overlying
 * @retvalue   :None
******************************************************************************/
void Show_Str(u16 x, u16 y, u16 fc, u16 bc, u8 *str,u8 size,u8 mode)
{
    u16 x0=x;
    u8 bHz=0;     //字符或者中文
    while(*str!=0)//数据未结束
    {
        if(!bHz)
        {
            if(x>(lcddev.width-size/2)||y>(lcddev.height-size))
            return;
            if(*str>0x80)bHz=1;//中文
            else              //字符
            {
                if(*str==0x0D)//换行符号
                {
                    y+=size;
                      x=x0;
                    str++;
                }
                else
```

```
                {
                    if(size==16)//字库中没有集成 12X24 16X32 的英文字体,用 8X16 代替
                    {
                        LCD_ShowChar(x,y,fc,bc,*str,16,mode);
                        x+=8; //字符,为全字的一半
                    }
                    else if(size==32)
                    {
                        LCD_ShowChar(x,y,fc,bc,*str,size,mode);
                        x+=size/2; //字符,为全字的一半
                    }
                    else
                    {
                        LCD_ShowChar(x,y,fc,bc,*str,size,mode);
                        x+=size/2; //字符,为全字的一半
                    }
                    str++;
                }
//                  str++;
            }
        }else//中文
        {
            if(x>(lcddev.width-size)||y>(lcddev.height-size))
            return;
           bHz=0;//有汉字库
            if(size==48)
            GUI_DrawFont48(x,y,fc,bc,str,mode);
            else if(size==32)
            GUI_DrawFont32(x,y,fc,bc,str,mode);
            else if(size==24)
            GUI_DrawFont24(x,y,fc,bc,str,mode);
            else
            GUI_DrawFont16(x,y,fc,bc,str,mode);

           str+=2;
           x+=size;//下一个汉字偏移
        }
    }
}

/*****************************************************************************
 * @name       :void Gui_StrCenter(u16 x, u16 y, u16 fc, u16 bc, u8 *str,u8 size,u8 mode)
 * @date       :2018-08-09
 * @function   :Centered display of English and Chinese strings
 * @parameters :x:the bebinning x coordinate of the Chinese and English strings
```

```
                y:the bebinning y coordinate of the Chinese and English strings
                                fc:the color value of Chinese and English strings
                                bc:the background color of Chinese and English strings
                                str:the start address of the Chinese and English strings
                                size:the size of Chinese and English strings
                                mode:0-no overlying,1-overlying
 * @retvalue    :None
******************************************************************************/
void Gui_StrCenter(u16 x, u16 y, u16 fc, u16 bc, u8 *str,u8 size,u8 mode)
{
    u16 len=strlen((const char *)str);
    u16 x1=(lcddev.width-len*16)/2;
    Show_Str(x1,y,fc,bc,str,size,mode);
}

/******************************************************************************
 * @name        :void Gui_Drawbmp16(u16 x,u16 y,u16 w,u16 h,const unsigned char *p)
 * @date        :2018-08-09
 * @function    :Display a 16-bit BMP image
 * @parameters  :x:the bebinning x coordinate of the BMP image
                 y:the bebinning y coordinate of the BMP image
                                p:the start address of image array
 * @retvalue    :None
******************************************************************************/
void Gui_Drawbmp16(u16 x,u16 y,u16 w,u16 h,const unsigned char *p) //显示 40*40 QQ 图片
{
    int i;
    unsigned char picH,picL;
    LCD_SetWindows(x,y,x+w-1,y+h-1);//窗口设置
    for(i=0;i<w*h;i++)
    {
        picL=*(p+i*2);      //数据低位在前
        picH=*(p+i*2+1);
        Lcd_WriteData_16Bit(picH<<8|picL);
    }
    LCD_SetWindows(0,0,lcddev.width-1,lcddev.height-1);//恢复显示窗口为全屏
}

void DrawTestPage(u8 *str)
{
//绘制固定栏 up
LCD_Clear(WHITE);
LCD_Fill(0,0,lcddev.width-1,36,BLUE);
//绘制固定栏 down
LCD_Fill(0,lcddev.height-36,lcddev.width-1,lcddev.height-1,RED);
POINT_COLOR=WHITE;
```

```
Gui_StrCenter(0,2,WHITE,BLUE,str,32,1);//居中显示
Gui_StrCenter(0,lcddev.height-34,GRED,BLUE,"http://www.jlxlcd.cn",32,1);//居中显示
Show_Str(32,46,BLUE,YELLOW,"1.型号名:JLX400-021-BN",32,1);
Show_Str(32,88,BLUE,YELLOW,"2.点阵数:480(RGB)x800",32,1);
Show_Str(32,130,BLUE,YELLOW,"3.可视区:51.84x86.4mm",32,1);
Show_Str(32,172,BLUE,YELLOW,"4.驱动 IC:NT35510",32,1);
Show_Str(32,214,BLUE,YELLOW,"5.模组尺寸:4.0 寸",32,1);
Show_Str(32,256,BLUE,YELLOW,"6.MCU 接口:16 位或 8 位并行可选",32,1);
Show_Str(32,298,BLUE,YELLOW,"7.工作温度:-20-70",32,1);
Show_Str(32,340,BLUE,YELLOW,"8.储存温度:-30-80",32,1);
Show_Str(32,382,BLUE,YELLOW,"9.LCM 工作电压:3.3V",32,1);
Show_Str(32,424,BLUE,YELLOW,"10.LED 工作电压:3.0V",32,1);
Show_Str(32,466,BLUE,YELLOW,"11.视角:12:00'CLOCK",32,1);
Show_Str(32,508,BLUE,YELLOW,"12.外形尺寸:57.1x96.8x2.2mm",32,1);

}

/*****************************************************************************
 * @name       :void main_test(void)
 * @date       :2019-08-09
 * @function   :Drawing the main Interface of the Comprehensive Test Program
 * @parameters :None
 * @retvalue   :None
*****************************************************************************/
void main_test(void)
{
    DrawTestPage("深圳市晶联讯电子有限公司");
    waitkey();
}


void main_test1(void)
{
    LCD_direction(1);
    LCD_Clear(BLUE);
    Show_Str(160,0,YELLOW,BLUE,"数字无线平面管理系统",48,1);
    Show_Str(608,54,GREEN,BLUE,"主板电源:12V",32,1);
    Show_Str(2,54,WHITE,BLUE,"系统时间:2019-09-04 15:40:05",32,1);
    POINT_COLOR = 0xFFFF;
    LCD_DrawRectangle(0,92,265,414);
    LCD_DrawRectangle(1,93,264,413);
    LCD_DrawRectangle(264,92,530,414);
    LCD_DrawRectangle(265,93,531,413);
    LCD_DrawRectangle(530,92,798,414);
    LCD_DrawRectangle(531,93,799,413);
    Show_Str(5,98,WHITE,BLUE,"当前信令:",32,1);
```

```
    Show_Str(10,162,WHITE,BLUE,"调车长未对码:",32,1);
    Show_Str(10,202,WHITE,BLUE,"发停车对码:",32,1);
    Show_Str(10,272,WHITE,BLUE,"如对码不成功:",32,1);
    Show_Str(10,314,WHITE,BLUE,"请重启再对码:",32,1);

    Show_Str(270,98,WHITE,BLUE,"已对码 ID:10",32,1);
    Show_Str(270,146,WHITE,BLUE,"调号:02",32,1);
    Show_Str(270,202,WHITE,BLUE,"信道:CW05",32,1);
    Show_Str(270,250,WHITE,BLUE,"调车电台",32,1);
    Show_Str(270,284,WHITE,BLUE,"剩余电量",32,1);
    Show_Str(270,324,WHITE,BLUE,"只有收到新指令时才更新电量",16,1);
    Show_Str(270,356,WHITE,BLUE,"通话主呼 ID 号:",32,1);

    Show_Str(633,98,WHITE,BLUE,"目录:",32,1);
    Show_Str(537,138,WHITE,BLUE,"1、按键说明",32,1);
    Show_Str(537,178,WHITE,BLUE,"2、领车功能说明",32,1);
    Show_Str(537,218,WHITE,BLUE,"3、放权功能说明",32,1);
    Show_Str(537,256,WHITE,BLUE,"4、测车说明",32,1);
    Show_Str(537,298,WHITE,BLUE,"5、备注",32,1);
    Show_Str(537,338,WHITE,BLUE,"6、操作指令",32,1);

    Show_Str(5,420,YELLOW,BLUE,"返回",48,1);
    Show_Str(121,420,YELLOW,BLUE,"上调",48,1);
    Show_Str(237,420,YELLOW,BLUE,"下调",48,1);
    Show_Str(353,420,YELLOW,BLUE,"确认",48,1);
    Show_Str(469,420,YELLOW,BLUE,"功能",48,1);
    Show_Str(585,420,YELLOW,BLUE,"设置",48,1);
    Show_Str(701,420,YELLOW,BLUE,"帮助",48,1);

    waitkey();
}

/*****************************************************************************
 * @name       :void Test_Color(void)
 * @date       :2019-08-09
 * @function   :Color fill test(white,black,red,green,blue)
 * @parameters :None
 * @retvalue   :None
*****************************************************************************/
void Test_Color(void)
{
    //DrawTestPage("测试 1:纯色填充测试");
//  LCD_Fill(0,0,lcddev.width-1,lcddev.height-1,WHITE);
//  Show_Str(20,30,BLUE,YELLOW,"BL Test",16,1);delay_ms(800);
    LCD_Fill(0,0,lcddev.width-1,lcddev.height-1,RED);
    Show_Str(20,30,BLUE,YELLOW,"RED ",16,1);waitkey();
```

```
    LCD_Fill(0,0,lcddev.width-1,lcddev.height-1,GREEN);
    Show_Str(20,30,BLUE,YELLOW,"GREEN ",16,1);waitkey();
    LCD_Fill(0,0,lcddev.width-1,lcddev.height-1,BLUE);
    Show_Str(20,30,RED,YELLOW,"BLUE ",16,1);waitkey();
}

/*****************************************************************************
 * @name       :void Test_FillRec(void)
 * @date       :2019-08-09
 * @function   :Rectangular display and fill test
                                Display red,green,blue,yellow,pink rectangular boxes in turn,
                                1500 milliseconds later,
                                Fill the rectangle in red,green,blue,yellow and pink in turn
 * @parameters :None
 * @retvalue   :None
*****************************************************************************/
void Test_FillRec(void)
{
    u8 i=0;
    LCD_Clear(WHITE);
//  DrawTestPage("测试 3:GUI 矩形填充测试");
//  LCD_Fill(0,20,lcddev.width-1,lcddev.height-20,WHITE);
    Show_Str(54,100,BLACK,WHITE,"矩形填充",32,1);
    for (i=0; i<5; i++)
    {
        POINT_COLOR=ColorTab[i];

    LCD_DrawRectangle(lcddev.width/6-80+(i*15),lcddev.height/2-80+(i*15),lcddev.width/6-80+(i*15)+60,lcddev.heig
ht/2-80+(i*15)+60);
    }
    delay_ms(1000);
//  LCD_Fill(0,20,lcddev.width-1,lcddev.height-20,WHITE);
    for (i=0; i<5; i++)
    {
        POINT_COLOR=ColorTab[i];

    LCD_DrawFillRectangle(lcddev.width/6-80+(i*15),lcddev.height/2-80+(i*15),lcddev.width/6-80+(i*15)+60,lcddev.
height/2-80+(i*15)+60);
    }
    delay_ms(1000);
//  waitkey();
}

/*****************************************************************************
 * @name       :void Test_Circle(void)
 * @date       :2019-08-09
```

```
 * @function   :circular display and fill test
                                Display red,green,blue,yellow,pink circular boxes in turn,
                                1500 milliseconds later,
                                Fill the circular in red,green,blue,yellow and pink in turn
 * @parameters :None
 * @retvalue   :None
*******************************************************************************/
void Test_Circle(void)
{
    u8 i=0;
//  DrawTestPage("测试 4:GUI 画圆填充测试");
//  LCD_Fill(0,20,lcddev.width-1,lcddev.height-20,WHITE);
    Show_Str(320,100,BLUE,WHITE,"画圆填充",32,1);
    for (i=0; i<5; i++)
        gui_circle(lcddev.width/2-60+(i*25),lcddev.height/2-50+(i*25),ColorTab[i],30,0);
    delay_ms(1000);
//  LCD_Fill(0,20,lcddev.width-1,lcddev.height-20,WHITE);
    for (i=0; i<5; i++)
        gui_circle(lcddev.width/2-60+(i*25),lcddev.height/2-50+(i*25),ColorTab[i],30,1);
    delay_ms(1000);
//  waitkey();
}


/*******************************************************************************
 * @name       :void Test_Triangle(void)
 * @date       :2019-08-09
 * @function   :triangle display and fill test
                                Display red,green,blue,yellow,pink triangle boxes in turn,
                                1500 milliseconds later,
                                Fill the triangle in red,green,blue,yellow and pink in turn
 * @parameters :None
 * @retvalue   :None
*******************************************************************************/
void Test_Triangle(void)
{
    u8 i=0;
//  DrawTestPage("测试 5:GUI Triangle 填充测试");
//  LCD_Fill(0,20,lcddev.width-1,lcddev.height-20,WHITE);
    Show_Str(600,100,RED,WHITE,"三角形填充",32,1);
    for(i=0;i<5;i++)
    {
        POINT_COLOR=ColorTab[i];

    Draw_Triangel(lcddev.width-200+(i*20),lcddev.height/2-20+(i*15),lcddev.width-170-1+(i*20),lcddev.height/2-20
-52-1+(i*15),lcddev.width-140-1+(i*20),lcddev.height/2-20+(i*15));
```

```
    }
    delay_ms(1000);
//  LCD_Fill(0,20,lcddev.width-1,lcddev.height-20,WHITE);
    for(i=0;i<5;i++)
    {
        POINT_COLOR=ColorTab[i];

    Fill_Triangel(lcddev.width-200+(i*20),lcddev.height/2-20+(i*15),lcddev.width-170-1+(i*20),lcddev.height/2-20
-52-1+(i*15),lcddev.width-140-1+(i*20),lcddev.height/2-20+(i*15));
    }
    waitkey();
}

int main(void)
{
    SystemInit();//初始化 RCC 设置系统主频为 72MHZ
    delay_init();       //延时初始化
    LCD_Init();       //液晶屏初始化
  //循环测试
    while(1)
    {
        main_test();       //测试主界面
        main_test1();
        Test_FillRec();       //GUI 矩形绘图测试
        Test_Circle();       //GUI 画圆测试
        Test_Triangle();    //GUI 三角形绘图测试
        Test_Color();       //简单刷屏填充测试
        Pic_test();       //图片显示示例测试
    }
}
```

-END-