高通汉显®

GTHMI

GT-GUI LCD 0.96 寸液晶模组

GT-GL12864O096-XXXX

数据手册

V1.0

www.hmi.gaotongfont.cn

GUI-LCD 数据手册导读——开启液晶显示开发的极简之旅

■ 硬件篇：

1) **LCD 显示屏基本规格：**包含尺寸、分辨率、亮度、接口类型、驱动 IC 等；研发工程师可参考该规格初步确定该 LCD 显示屏是否符合技术要求；
2) **LCD 显示屏参考图纸：**显示屏结构，包括 LCD 显示屏外形尺寸， AA 尺寸，FPC 外形尺寸，FPC PIN 参数等；参考图纸可用于 PCB 布局和封装设计；
3) **LCD 屏接口原理图设计/PCB 布线：**
   参考规格书的目录 4 接口定义部分、目录 7 屏模块->I2C 通信/SPI 通信->引脚介绍部分、目录 8 GUI 芯片/字库芯片->引脚介绍部分，注意若使用型号为搭载了 GUI 芯片的液晶模组时，为方便开发过程中烧录到 GUI 芯片修改内容，您在原理图设计时最好预留 8PIN 接口方便连接到 GUI 芯片

■ 软件篇：

1) **LCD 显示屏 SPI/I2C 接口驱动开发：**根据 LCD 显示屏的示例代码和接口定义，开发屏 SPI/I2C 屏驱动代码。参考目录 4 接口定义部分、目录 7 屏模块->I2C 通信/SPI 通信->引脚部分；
2) **LCD 显示屏初始化设置：**根据 LCD 显示屏的示例代码完成显示屏的初始化配置，初始化后一般会有乱点出现代表初始化正常。参考目录 7 屏模块->I2C/SPI 驱动程序；
3) **GUI 芯片/字库芯片驱动开发：**
   根据 GUI 芯片/字库芯片示例代码和接口定义开发芯片读写驱动；
   参考目录 8 GUI 芯片/字库芯片下面的 SPI 例程。

修订记录

Rev.	Contents	Date
V1.0	First release	2023/09/20

目录

1.产品型号

GT- GL　12864　T/O　+096 —　X +　X/G +　X +　X

GT-GL=高通 GUI-LCD

屏幕分辨率

T：TFT 彩屏
O: OLDE

屏幕尺寸

产品代码

消费类：X
工控类：G

C：电容触摸
R：电阻触摸
N：无触摸

容量：
4Mb
16Mb
32Mb
64Mb
128Mb

描述	型号
+液晶模组	GT-GL12864O096-S0XN
+GUI 芯片	GT-GL12864O096-S0XN16

表 1-1

2.概述

GT-GUI LCD 0.96 寸液晶模组是高通开发的 GUI-LCD 轻量级嵌入式交互系统显示屏，使用 0.96 寸显示屏，分辨率为 128*64，搭配高通 GUI-LCD 开发板可快速搭建起嵌入式 GUI 交互系统，驱动芯片为 SSD1315 芯片，支持 I2C 端口通信和 SPI 通信,并搭载有高通字库芯片，显示屏搭配我司 GT21L16S2Y 字库，支持 GB2312 国标简体 12X12 点阵和 16X16 点阵的汉字库芯片、ASCII 字符及 GB2312 与 UNICODE 编码互转表，强大的显示和字库支持，帮助您快速实现嵌入式 GUI 交互系统的开发。

3.基本规格

No	Items	Parameter	Unit
1	LCD size	0.96	Inch
2	Number of Dots	128*64	Dots
3	Active Area	21.74 (W) x10.86(H)	mm
4	Dimensional Outline	26.7(W) *19.2(H)*1.05(T)	mm
5	Pixel Pitch	0.17 × 0.17	mm
6	Pixel size	0.15*0.15	mm
7	Duty	1/64 Dutr	
8	Display Mode	Passive Matrix	
9	Display Color	White	
10	display driver IC	SSD1315	
11	GUI IC	16	Mb
12	Back Light	White OLED	
13	Screen communication mode	I2C/SPI	
14	Flash communication mode	SPI	
15	operating temperature	-40- +70℃	℃
16	Storage Temperature	-40- +85℃	℃

表 3-1

4.接口定义

引脚	符号/名称	功能说明
1	GND	接地 0V
2	C2N	接升压电容
3	C2P	接升压电容
4	C1P	接升压电容
5	C1N	接升压电容
6	VCC	NC
7	GND	接地 0V
8	BS0	显示屏通讯设置 BS0=0 BS1=0 (SPI)
9	BS1	显示屏通讯设置 BS0=0 BS1=1 (I2C)
10	CS	片选，低电平选中
11	RESET	复位，低电平复位；高电平正常工作
12	D/C#/SA0	SPI：数据/命令控制 I2C：从地址选择
13	VDD	OLED 供电
14	D0(SCL)	显示屏的时钟信号
15	D1(SDA_IN)	当选择 SPI 模式时，D1 作为数据输入 SDIN，当选
16	D2(SDA_OUT)	择 I2C 模式时，D2 应与 D1 连接作为 SDA 使用
17	IREF	输出电流参考
18	DI	字库芯片（接收数据）
19	GND	接地 0V
20	CLK	字库芯片（时钟信号）
21	GND	接地 0V
22	V33	字库芯片供电（+3.3V)
23	CS#	字库芯片(片选)
24	DO	字库芯片（发送数据)

表 4-1

5.电气特性

5.1 极限参数

Item	Min	Max	Unit	Condition	Remark
Supply Voltage (V_{DD})	-0.3	4.0	V	Ta = 25℃	IC maximum rating
Supply Voltage (V_{BAT})	-0.3	6.0	V	Ta = 25℃	IC maximum rating
SupplyVoltage (Vcc)	0	18	V	Ta = 25℃	IC maximum rating
Operating Temp.	-40	70	℃		-
Storage Temp	-40	85	℃		Note (2)

表 5-1

Note:

(1) Maximum ratings are those values beyond which damages to the OLED module may occur.

(2) The defined temperature ranges do not include the polarizer. The maximum withstood temperature of the polarizer should be 80 °C.

5.2 电气规格

Symbol	Parameter	Test Condition	Min	Typ	Max	Unit
V_{DD}	Logic Supply Voltage	Ta = 25 °C	1.65	3.3	3.5	V
V_{BAT}	Charge Pump Regulator Supply Voltage	Ta = 25 °C	3.0	3.6	4.5	V
V_{CC}	Operating Voltage (for OLED panel)	Ta = 25 °C	7.5		16.5	V
V_{OH}	High Logic Output Level	IOUT = 100uA, 3.3MHz	0.9* VDD	-	VDD	V
V_{OL}	Low Logic Output Level	IOUT = 100uA, 3.3MHz	VSS		0.1*VDD	V
V_{IH}	High Logic Input Level		0.8* VDD		VDD	V
V_{IL}	Low Logic Input Level	-	VSS	-	0.2*VDD	V

表 5-2

5.3 液晶面板功耗

Parameter	Min	Typ.	Max	Unit	Comments
Normal mode current consumption (IBAT) (Charge Pump)	-	32	35	mA	All pixels on (1)
	-	9	11	mA	20% pixels on (1)
IDD sleep mode current	-		10	uA	Sleep mode Current (2)
ICC sleep mode current	-	-	10	uA	Sleep mode Current (2)
Pixel Luminance(Charge Pump)	100	120		cd/m2	Display Average
CIEx (White)	0.24	0.28	0.32		CIE1931
CIEy (White)	0.28	0.32	0.36		CIE1931
Dark Room Contrast	2000:1				
Viewing Angle	160			degree	
Response Time		10		μs	

表 5-3

(1) Normal mode condition : (Charge Pump)

- V_{BAT} = 3.6V
- Contrast setting : 0XAF
- Charge Pump Setting:0x14
- Frame rate : 105Hz
- Duty setting : 1/64

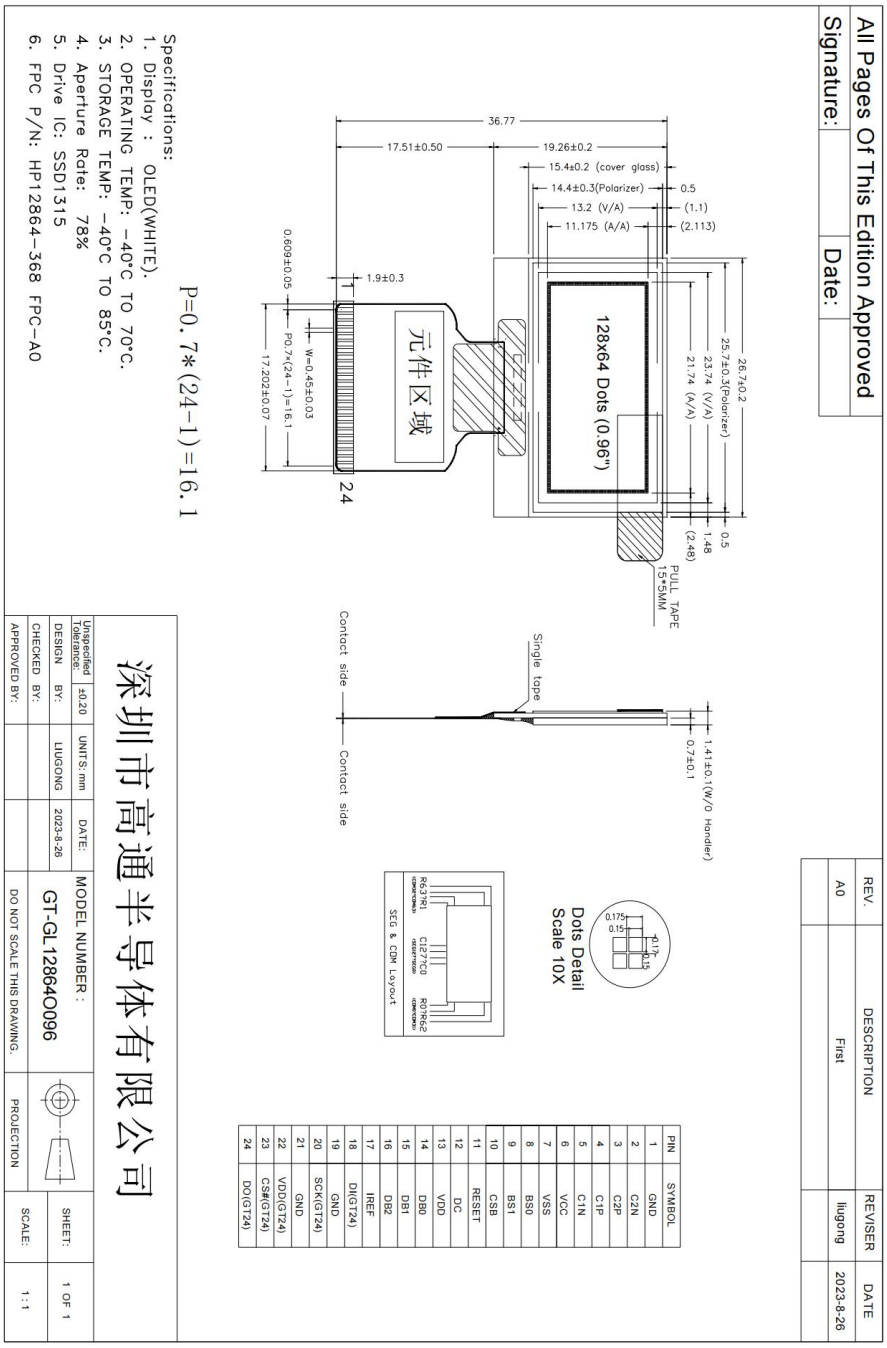
(2) Sleep mode condition :

When send 0xae command OLED display off and memory data will be maintained.

(3) Wake up condition :

When send 0xaf command OLED will be turned on.

6.模组图纸-无 TP


All Pages Of This Edition Approved
Signature:
Date:
REV: A0
DESCRIPTION: First
REVISER: liugong
DATE: 2023-8-26
36.77
17.51±0.50
19.26±0.2
15.4±0.2 (cover glass)
14.4±0.3(Polarizer)
13.2 (V/A)
11.175 (A/A)
0.5
(1.1)
(2.113)
26.7±0.2
25.7±0.3(Polarizer)
23.74 (V/A)
21.74 (A/A)
1.48
(2.48)
PULL TAPE 15*5MM
128x64 Dots (0.96")
元件区域
1.9±0.3
0.609±0.05
P0.7*(24-1)=16.1
17.202±0.07
W=0.45±0.03
1
24
P=0.7*(24-1)=16.1
Specifications:
1. Display : OLED(WHITE).
2. OPERATING TEMP: -40°C TO 70°C.
3. STORAGE TEMP: -40°C TO 85°C.
4. Aperture Rate: 78%
5. Drive IC: SSD1315
6. FPC P/N: HP12864-368 FPC-A0
Contact side
Contact side
Single tape
1.41±0.1(W/O Handler)
0.7±0.1
Dots Detail
Scale 10X
0.175
0.15
0.175
0.15
SEG & COM Layout
PIN SYMBOL
1 GND
2 C2N
3 C2P
4 C1P
5 C1N
6 VCC
7 VSS
8 BS0
9 BS1
10 CSB
11 RESET
12 DC
13 VDD
14 DB0
15 DB1
16 DB2
17 IREF
18 DI(GT24)
19 GND
20 SCK(GT24)
21 GND
22 VDD(GT24)
23 CS#(GT24)
24 DO(GT24)
深圳市高通半导体有限公司
Unspecified Tolerance: ±0.20
DESIGN BY:
CHECKED BY:
APPROVED BY:
UNITS: mm
LIUGONG
DATE: 2023-8-26
MODEL NUMBER :
GT-GL12864O096
DO NOT SCALE THIS DRAWING.
PROJECTION
SCALE: 1:1
SHEET: 1 OF 1

7 屏模块

7.1 SPI 通信

7.1.1 SPI 通信连接图

本 SPI 通信例程使用模拟 SPI，如客户使用硬件 SPI 需要修改 SPI 驱动。图 7-1 为 SPI 通信连接图。

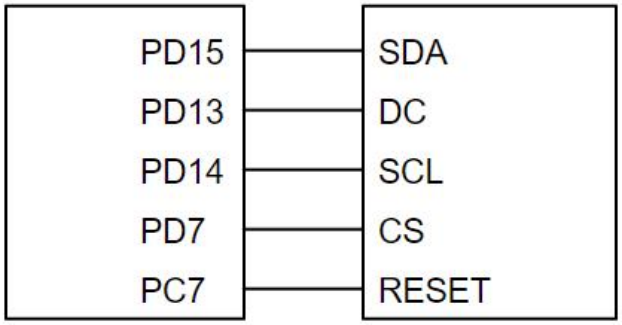


图 7-1

7.1.2 SPI 通信原理图

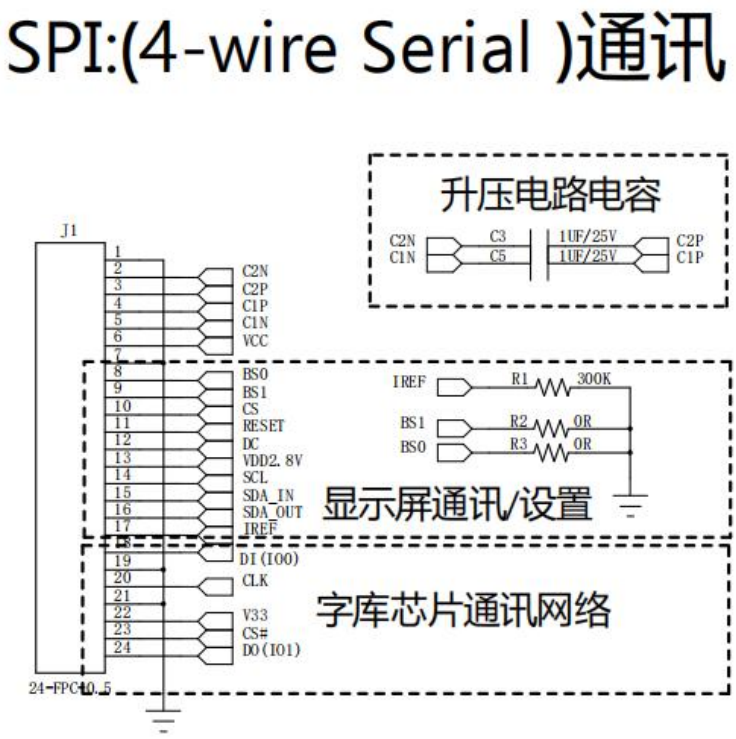


图 7-2

7.1.3 点亮屏流程

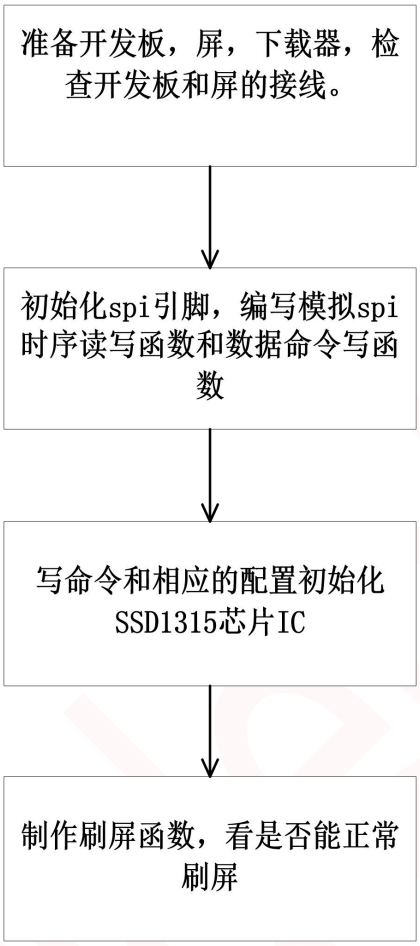


图 7-3

屏需要初始化屏 IC 才能正常显示，图 7-3 是 spi 驱动屏幕的步骤。

7.1.4 SPI 驱动程序

```
/**
* @brief SPI send data/command
* @param dat
*/
void   SendDataSPI(unsigned char dat)
{
    unsigned char i;

    for(i=0; i<8; i++)
    {
        LCD_SCL_LOW;
    if( (dat&0x80)!=0 ){
        LCD_SDA_HIGH;
    }else{
```

```
        LCD_SDA_LOW;
    }
        dat <<= 1;
        LCD_SCL_HIGH;
    }
}
/**
* @brief   this function is write command to lcd.
* @param   command : 写进去的命令值.
* @retval none
*/
void WriteComm(unsigned char command)
{
    LCD_CS_LOW;
    LCD_RS_LOW;//拉低选择发送命令
    SendDataSPI(command);
    LCD_CS_HIGH;
}
/**
* @brief   this function is write data to lcd.
* @param   data : the data to write.
* @retval none
*/
void WriteData(unsigned char data)
{
    LCD_CS_LOW;
    LCD_RS_HIGH;//拉高选择发送数据
    SendDataSPI(data);
    LCD_CS_HIGH;
}
/**
* @brief   configures the oled.
*          this function must be called before any write/read operation
*          on the oled.
*/
void OLED_init(void)
{
    lcd_port_init();

    delay_ms(1);
    LCD_RS_HIGH;
    LCD_CS_HIGH;

    LCD_RESET_LOW; // OLED 复位
    delay_ms(100);    //复位延时
```

```
    LCD_RESET_HIGH;  //结束复位
    WriteComm(0xAE); //Display Off
    WriteComm(0xD5); //SET DISPLAY CLOCK
    WriteComm(0x80); //105HZ
    WriteComm(0xA8); //Select Multiplex Ratio
    WriteComm(0x3F); //Default => 0x3F (1/64 Duty)
    WriteComm(0xD3); //Setting Display Offset
    WriteComm(0x00); //00H Reset
    WriteComm(0x40); //Set Display Start Line
    WriteComm(0x8D); //Set Charge Pump
    //Write_Command(0x10); //Disable Charge Pump
    WriteComm(0x14); //Enable Charge Pump
    WriteComm(0xAD); // Internal IREF Setting
    WriteComm(0x20); // Disable internal IREF
    //Write_command(0x30); // Enable internal IREF
    WriteComm(0xA1); //Set Segment Re-Map Default
    WriteComm(0xC8); //Set COM Output Scan Direction
    WriteComm(0xDA); //Set COM Hardware Configuration
    WriteComm(0x12); //Alternative COM Pin
    WriteComm(0x81); //Set Contrast Control
    WriteComm(0X0F);
    WriteComm(0xD9); //Set Pre-Charge period
    WriteComm(0xF1);
    WriteComm(0xDB); //Set Deselect Vcomh level
    WriteComm(0x30);
    WriteComm(0xA4); //Entire Display ON
    WriteComm(0xA6); //Set Normal Display
    //Clear_Ram();
    WriteComm(0xAF); //Display ON
}
void lcd_address(unsigned char page,unsigned char column)
{
    column=column-1; //我们平常所说的第 1 列，在 LCD 驱动 IC 里是第 0 列。所以在这里减去 1.
    page=page-1;
    //设置页地址。每页是 8 行。一个画面的 64 行被分成 8 个页。我们平常所说的第 1 页，在 LCD 驱动 IC
里是第 0 页，所以在这里减去 1
    WriteComm(0xb0+page);
    WriteComm(((column>>4)&0x0f)+0x10); //设置列地址的高 4 位
    WriteComm(column&0x0f); //设置列地址的低 4 位
}
//全屏清屏
void full_display(unsigned short color)
{
    unsigned char data1,data2, i,j;
    data1 = color >> 8;
```

```
    data2 = color;
    for(i=0;i<8;i++)
    {
        lcd_address(i+1,1);
        for(j=0;j<64;j++)
        {
            WriteData(data1);
            WriteData(data2);
        }
    }
}
uint8_t OLED_GRAM1[128][8];

//更新显存到 LCD
void OLED_Refresh_Gram(void)
{
    uint8_t i,n;
    for(i=0;i<8;i++)
    {
        WriteComm (0xB0+i);    //设置页地址（0~7）
        WriteComm (0x00);      //设置显示位置—列低地址
        WriteComm (0x10);      //设置显示位置—列高地址

        for(n=0;n<128;n++)
            WriteData(OLED_GRAM1[n][i]);
    }
}

//画点
//x:0~127
//y:0~63
//t:1 填充 0,清空
void OLED_DrawPoint1(uint8_t x,uint8_t y,uint8_t t)
{
    uint8_t pos,bx,temp=0;
    if(x>127||y>63)return;//超出范围了.
    pos=7-y/8;
    bx=y%8;
    temp=1<<(7-bx);
    if(t)OLED_GRAM1[x][pos]|=temp;
    else OLED_GRAM1[x][pos]&=~temp;
    OLED_Refresh_Gram();
}
```

7.2 I2C 通信

7.2.1 I2C 通信连接图

如果使用 I2C 方式驱动屏幕，则需将 D2 与 D1 连接起来作为 SDA 数据引脚使用，原 DC 数据命令选择引脚作为 SA0 从地址选择使用，本例程拉低使用,图 7-4 为 I2C 通信连接图。

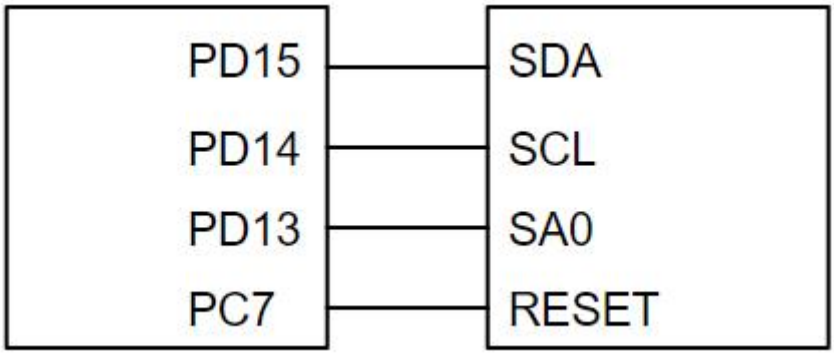


图 7-4

7.2.2 I2C 通信原理图

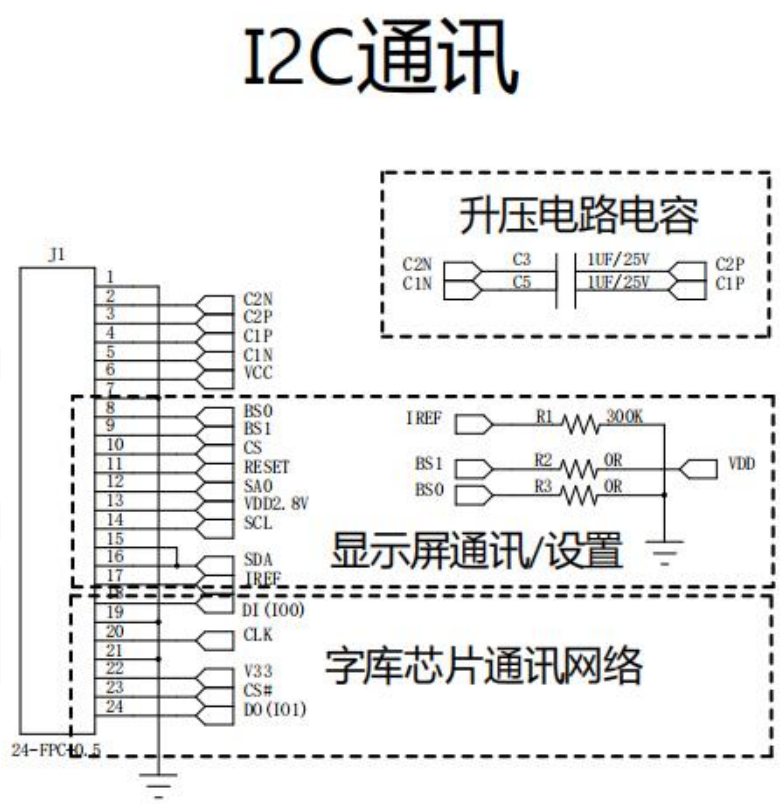


图 7-5

7.2.3 点亮屏流程

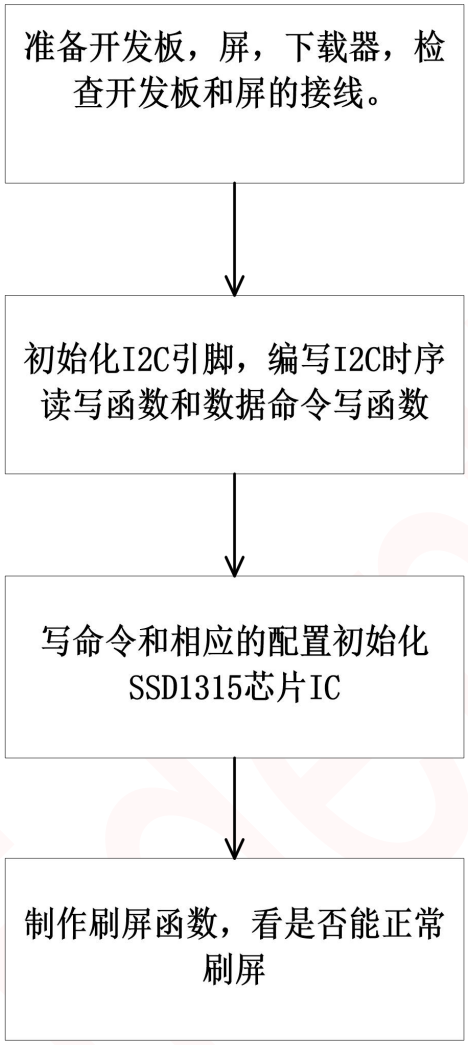


图 7-6

屏需要初始化屏 IC 才能正常显示，图 7-6 是 I2C 驱动屏幕的步骤。

7.2.4 I2C 驱动程序

```
#define OLED_CMD  0 //写命令
#define OLED_DATA 1 //写数据

//延时
void IIC_delay(void)
{
    u16 t=90;
    while(t--);
}

//起始信号
```

```
void I2C_Start(void)
{
    OLED_SDA_Set();
    OLED_SCL_Set();
    IIC_delay();
    OLED_SDA_Clr();
    IIC_delay();
    OLED_SCL_Clr();
    IIC_delay();
}

//结束信号
void I2C_Stop(void)
{
    OLED_SDA_Clr();
    OLED_SCL_Set();
    IIC_delay();
    OLED_SDA_Set();
}

//等待信号响应
void I2C_WaitAck(void) //测数据信号的电平
{
    OLED_SDA_Set();
    IIC_delay();
    OLED_SCL_Set();
    IIC_delay();
    OLED_SCL_Clr();
    IIC_delay();
}

//写入一个字节
void Send_Byte(u8 dat)
{
    u8 i;
    for(i=0;i<8;i++)
    {
        if(dat&0x80)//将 dat 的 8 位从最高位依次写入
        {
            OLED_SDA_Set();
        }else{
            OLED_SDA_Clr();
        }
        IIC_delay();
        OLED_SCL_Set();
```

```
        IIC_delay();
        OLED_SCL_Clr();//将时钟信号设置为低电平
        dat<<=1;
    }
}

//发送一个字节
//mode:数据/命令标志 0,表示命令;1,表示数据;
void OLED_WR_Byte(u8 dat,u8 mode)
{
    I2C_Start();
    Send_Byte(0x78);
    I2C_WaitAck();
    if(mode){Send_Byte(0x40);}
    else{Send_Byte(0x00);}
    I2C_WaitAck();
    Send_Byte(dat);
    I2C_WaitAck();
    I2C_Stop();
}
//OLED 的初始化
void OLED_Init(void)
{
    IIC_Init();

    OLED_RES_Clr();
    delay_ms(200);
    OLED_RES_Set();
    OLED_WR_Byte(0xAE,OLED_CMD); //Display Off
    OLED_WR_Byte(0xD5,OLED_CMD); //SET DISPLAY CLOCK
    OLED_WR_Byte(0x80,OLED_CMD); //105HZ
    OLED_WR_Byte(0xA8,OLED_CMD); //Select Multiplex Ratio
    OLED_WR_Byte(0x3F,OLED_CMD); //Default => 0x3F (1/64 Duty)
    OLED_WR_Byte(0xD3,OLED_CMD); //Setting Display Offset
    OLED_WR_Byte(0x00,OLED_CMD); //00H Reset
    OLED_WR_Byte(0x40,OLED_CMD); //Set Display Start Line
    OLED_WR_Byte(0x8D,OLED_CMD); //Set Charge Pump
    //OLED_WR_Byte(0x10,OLED_CMD);//Disable Charge Pump
    OLED_WR_Byte(0x14,OLED_CMD); //Enable Charge Pump
    OLED_WR_Byte(0xAD,OLED_CMD); // Internal IREF Setting
    OLED_WR_Byte(0x20,OLED_CMD); // Disable internal IREF
    //OLED_WR_Byte(0x30,OLED_CMD);// Enable internal IREF
    OLED_WR_Byte(0xA1,OLED_CMD); //Set Segment Re-Map Default
    OLED_WR_Byte(0xC8,OLED_CMD); //Set COM Output Scan Direction
    OLED_WR_Byte(0xDA,OLED_CMD); //Set COM Hardware Configuration
```

```
    OLED_WR_Byte(0x12,OLED_CMD); //Alternative COM Pin
    OLED_WR_Byte(0x81,OLED_CMD); //Set Contrast Control
    OLED_WR_Byte(0XCF,OLED_CMD);
    OLED_WR_Byte(0xD9,OLED_CMD); //Set Pre-Charge period
    OLED_WR_Byte(0xF1,OLED_CMD);
    OLED_WR_Byte(0xDB,OLED_CMD); //Set Deselect Vcomh level
    OLED_WR_Byte(0x30,OLED_CMD);
    OLED_WR_Byte(0xA4,OLED_CMD); //Entire Display ON
    OLED_WR_Byte(0xA6,OLED_CMD); //Set Normal Display
    //Clear_Ram();
    OLED_WR_Byte(0xAF,OLED_CMD); //Display ON
}
void lcd_addr(unsigned char page,unsigned char column)
{
    column=column-1; //我们平常所说的第 1 列，在 LCD 驱动 IC 里是第 0 列。所以在这里减去 1.
    page=page-1;
    //设置页地址。每页是 8 行。一个画面的 64 行被分成 8 个页。我们平常所说的第 1 页，在 LCD 驱动 IC
里是第 0 页，所以在这里减去 1
    OLED_WR_Byte(0xb0+page,OLED_CMD);
    OLED_WR_Byte(((column>>4)&0x0f)+0x10,OLED_CMD); //设置列地址的高 4 位
    OLED_WR_Byte(column&0x0f,OLED_CMD); //设置列地址的低 4 位
}
//全屏清屏
void full_display(unsigned short color)
{
    unsigned char data1,data2, i,j;
    data1 = color >> 8;
    data2 = color;
    for(i=0;i<8;i++)
    {
        lcd_addr(i+1,1);
        for(j=0;j<64;j++)
        {
            OLED_WR_Byte(data1, OLED_DATA);
            OLED_WR_Byte(data2, OLED_DATA);
        }
    }
}
uint8_t OLED_GRAM1[128][8];

//更新显存到 OLED
void OLED_Refresh(void)
{
    u8 i,n;
    for(i=0;i<8;i++)
```

```
    {
        OLED_WR_Byte(0xb0+i,OLED_CMD); //设置行起始地址
        OLED_WR_Byte(0x00,OLED_CMD);   //设置低列起始地址
        OLED_WR_Byte(0x10,OLED_CMD);   //设置高列起始地址
        I2C_Start();
        Send_Byte(0x78);
        I2C_WaitAck();
        Send_Byte(0x40);
        I2C_WaitAck();
        for(n=0;n<128;n++)
        {
            Send_Byte(OLED_GRAM[n][i]);
            I2C_WaitAck();
        }
        I2C_Stop();
    }
}
//画点
//x:0~127
//y:0~63
//t:1 填充 0,清空
void OLED_DrawPoint(u8 x,u8 y,u8 t)
{
    u8 i,m,n;
    i=y/8;
    m=y%8;
    n=1<<m;
    if(t){OLED_GRAM[x][i]|=n;}
    else
    {
        OLED_GRAM[x][i]=~OLED_GRAM[x][i];
        OLED_GRAM[x][i]|=n;
        OLED_GRAM[x][i]=~OLED_GRAM[x][i];
    }
    OLED_Refresh();//更新显示
}
```

字符串显示函数参考 SPI 通信驱动程序，这里不再赘述。

8 GUI 芯片模块

8.1 SPI 通信

8.1.1 引脚介绍

图 8-1 是普通 SPI 连接图，在使用 SPI 通信，HOLD 引脚和 WP 脚需要接 2K 电阻到 VCC。

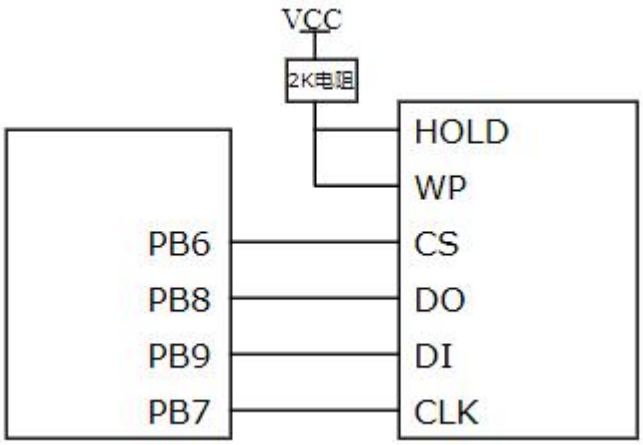


图 8-1

8.1.2 GUI 芯片指令

指令名称	字节 1	字节2	字节3	字节4	字节5	字节6	下一个字节
写使能	06h						
写禁能	04h						
读状态寄存器	05h	(S7-S0)					
写状态寄存器	01h	S7-S0					
读数据	03h	A23-A16	A15-A8	A7-A0	D7-D0	下一个字节	继续
快读	0Bh	A23-A16	A15-A8	A7-A0	伪字节	D7-D0	下一个字节
快读双输出	3Bh	A23-A16	A15-A8	A7-A0	伪字节	D7-D0	每四个时钟一个字节
页编程	02h	A23-A16	A15-A8	A7-A0	D7-D0	下一个字节	直到256个字节
块擦除(64k)	D8h	A23-A16	A15-A8	A7-A0			
扇区擦除(4k)	20h	A23-A16	A15-A8	A7-A0			
芯片擦除	C7h						
制造/器件ID	90h	伪字节	伪字节	00h	M7-M0		

图 8-2 操作存储芯片指令，可供客户写驱动做参考。

8.1.3 SPI 驱动程序

```
/**
* @brief   write a byte to flash
* @param   data: data to write
* @retval flash return data
*/
uint8_t spi_byte_write(uint8_t data)
{
    uint8_t brxbuff;
    spi_i2s_dma_transmitter_enable(SPI4, FALSE);
    spi_i2s_dma_receiver_enable(SPI4, FALSE);
    spi_i2s_data_transmit(SPI4, data);
    while(spi_i2s_flag_get(SPI4, SPI_I2S_RDBF_FLAG) == RESET);
    brxbuff = spi_i2s_data_receive(SPI4);
    while(spi_i2s_flag_get(SPI4, SPI_I2S_BF_FLAG) != RESET);
    return brxbuff;
}
/**
* @brief   read a byte to flash
* @param   none
* @retval flash return data
*/
uint8_t spi_byte_read(void)
{
    return (spi_byte_write(FLASH_SPI_DUMMY_BYTE));
}
/**
* @brief   read data from flash
* @param   pbuffer: the pointer for data buffer
* @param   read_addr: the address where the data is read
* @param   length: buffer length
* @retval none
*/
void spiflash_read(uint8_t *pbuffer, uint32_t read_addr, uint32_t length)
{
    FLASH_CS_LOW();
    spi_byte_write(0x03); /* send instruction 0x03 普通读取*/
    spi_byte_write((uint8_t)((read_addr) >> 16)); /* send 24-bit address */
    spi_byte_write((uint8_t)((read_addr) >> 8));
    spi_byte_write((uint8_t)read_addr);
    //spi_byte_write((uint8_t)0xff);//使用 0x0B 快速读取时需额外发送一个字节
    spi_bytes_read(pbuffer, length);
    FLASH_CS_HIGH();
```

```
}
/**
* @brief   erase a sector data
* @param   erase_addr: sector address to erase
* @retval none
*/
void spiflash_sector_erase(uint32_t erase_addr)
{
    spiflash_write_enable();
    spiflash_wait_busy();
    FLASH_CS_LOW();
    spi_byte_write(0x20); //擦除指令
    spi_byte_write((uint8_t)((erase_addr) >> 16));
    spi_byte_write((uint8_t)((erase_addr) >> 8));
    spi_byte_write((uint8_t)erase_addr);
    FLASH_CS_HIGH();
    spiflash_wait_busy();
}
/**
* @brief   read data continuously
* @param   pbuffer: buffer to save data
* @param   length: buffer length
* @retval none
*/
void spi_bytes_read(uint8_t *pbuffer, uint32_t length)
{
    while(length--)
    {
        while(spi_i2s_flag_get(SPI4, SPI_I2S_TDBE_FLAG) == RESET);
        spi_i2s_data_transmit(SPI4, 0xa5);//随意值皆可
        while(spi_i2s_flag_get(SPI4, SPI_I2S_RDBF_FLAG) == RESET);
        *pbuffer = spi_i2s_data_receive(SPI4);
        pbuffer++;
    }
}
/**
* @brief   read device id
* @param   none
* @retval device id
*/
uint16_t spiflash_read_id(void)
{
    uint16_t wreceivedata = 0;
    FLASH_CS_LOW();
    spi_byte_write(0x90); //0x90 读取 id 指令
```

```
    spi_byte_write(0x00);
    spi_byte_write(0x00);
    spi_byte_write(0x00);
    wreceivedata |= spi_byte_read() << 8;
    wreceivedata |= spi_byte_read();
    FLASH_CS_HIGH();
    return wreceivedata;
}
```

8.1.4 字库驱动程序

从高通技术支持处获取字库相应的静态库支持文件，实现静态库 h 文件中外部声明的函数。

```
/* 外部函数声明 */
extern unsigned char r_dat_bat(unsigned long address,unsigned long byte_long,unsigned char *p_arr);
extern unsigned char gt_read_data(unsigned char* sendbuf , unsigned char sendlen , unsigned char* receivebuf, unsigned int receivelen);
/* ------------------------------------------------------------ */
//字库初始化
int GT_Font_Init(void);
```

实现示例如下：

```
/**
 * @brief 外部声明函数 从 address 地址读取 DataLen 字节数据，存储于 pBuff 数组中
 * @param address   读取地址
 * @param DataLen   读取字节
 * @param pBuff     存放数组
 * @return unsigned char
 */
unsigned char r_dat_bat(unsigned long address,unsigned short DataLen,unsigned char *pBuff)
{
    spiflash_read(pBuff,address,DataLen);
    return 1;
}
/**
 *@brief 向字库芯片发送 sendlen 字节的 sendbuf 数组数据后，读取 receivelen 字节数据存放于 receivebuf 数组
 * @param sendbuf       发送数据
 * @param sendlen       发送长度
 * @param receivebuf    接收数据
 * @param receivelen    接收长度
 * @return unsigned char
 */
unsigned char gt_read_data(unsigned char* sendbuf , unsigned char sendlen , unsigned char* receivebuf, unsigned int receivelen)
```

```
{
    unsigned int i;
    FLASH_CS_LOW();
    for(i = 0; i < sendlen;i++)
    {
       spi_byte_write(sendbuf[i]);
    }
    for(i = 0; i < receivelen;i++)
    {
       receivebuf[i] = spi_byte_read();
    }
    FLASH_CS_HIGH();
    return 1;
}
```

之后在主函数中调用静态库 h 中声明的 GT_Font_Init 字库初始化函数，返回值大于 0，则说明字库初始化成功。即可调用静态库中声明的函数接口调取不同点阵大小的字库数据并进行显示。

```
/**
 * @brief 显示 16 点阵汉字与 ASCII 码 8x16 点阵混合字符串
 * @param page 显示页
 * @param column 显示列
 * @param text 字符串内容
 */
void display_string_16(unsigned char page,unsigned char column,unsigned char *text)
{
    unsigned char i = 0,j = 0,k = 0,w = 0;
    unsigned char DZ_Buff[32] = {0};
    while(text[i] != 0)
    {
        if(column > 128)
        {
            column = 1;
            page += 2;
        }
        if(text[i] > 0x7f){//获取 16x16 中文字形数据
            gt_16_GetData (text[i],text[i+1],DZ_Buff);
            w = 16;
        }else{//获取 8x16ASCII 码字形数据
            ASCII_GetData(text[i],ASCII_8X16_A,DZ_Buff);
            w = 8;
        }

        for(k = 0; k < 2; k++) //16 的高 等于 2 页
        {
            lcd_address(page+k,column);
            for(j = 0; j < w; j++)
```

```
            {
                WriteData(DZ_Buff[k*w + j]);
            }
        }
        column += w;
    }
}
```

主函数的调用情况如下：

```
int main(void)
{
    system_clock_config();
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);
    at32_board_init();
    spiflash_init();    //SPI 接口初始化
    OLED_init();        //屏幕初始化
    GT_Font_Init();     //字库初始化
    display_string_16(1,1,"Hello，高通字库");
    while(1)
    {
    }
}
```

9 注意事项

1.请勿拆卸液晶显示模块。

2.不要在印制电路板上钻额外的孔 ， 修改形状或更改印制线路板上元件的位置。

3.除焊接接口外 ， 不要用烙铁做任何更改；焊接温度保证在 320°C-350°C ， 焊接时间控制在 10S 以内 ， 焊接时注意不要在同一处停留时间太久以免烫伤 FPC。

4.其他事项在不清楚使用之前 ， 请联系我司人员指导进行。

10 联系信息

深圳高通半导体有限公司
地址：深圳市福田区车公庙泰然九路金润大厦 12C
电话： 0755-83453881 83453855
技术支持： www.hmi.gaotongfont.cn
Call：0755-83453881

QQ 技术频道：

企业微信客服：