文档版本	V1.0.0
发布日期	20221026

APT32F110x 基于 CSI 库 LCD 应用指南

目录

1 概述

本文介绍了在APT32F110x中LCD控制器。

2. 适用的硬件

该例程使用于 APT32F110x 系列学习板

3. 应用方案代码说明

基于 APT32F110x 完整的 CSI 库文件系统，进行配置 LCD 模块。

3.1 LCD 资源

- **硬件配置**

LCD 模块有 2-COM x 32-SEG、 4-COM x 30-SEG 或 8-COM x26-SEG 驱动能力，即最大能驱动 64、120 或 208 LCD 段（面板），每个段都可由 LCD 内部显示数据寄存器对应的位控制。

支持 1/2、1/3、1/4 和 1/8 占空比。支持 1/2、1/3 和 1/4 偏置。

内置升压电路，通过软件选择的 LCD 输出电压（对比度）：VLCDMIN 到 VLCDMAX

双缓冲显示存储器，允许软件随时更新 LCD_DMRx（LCD 显示存储寄存器）

可配置闪烁功能， 1、2、3、4、8 个或所有像素以可配置的频率闪烁。

- **结构示意图：**

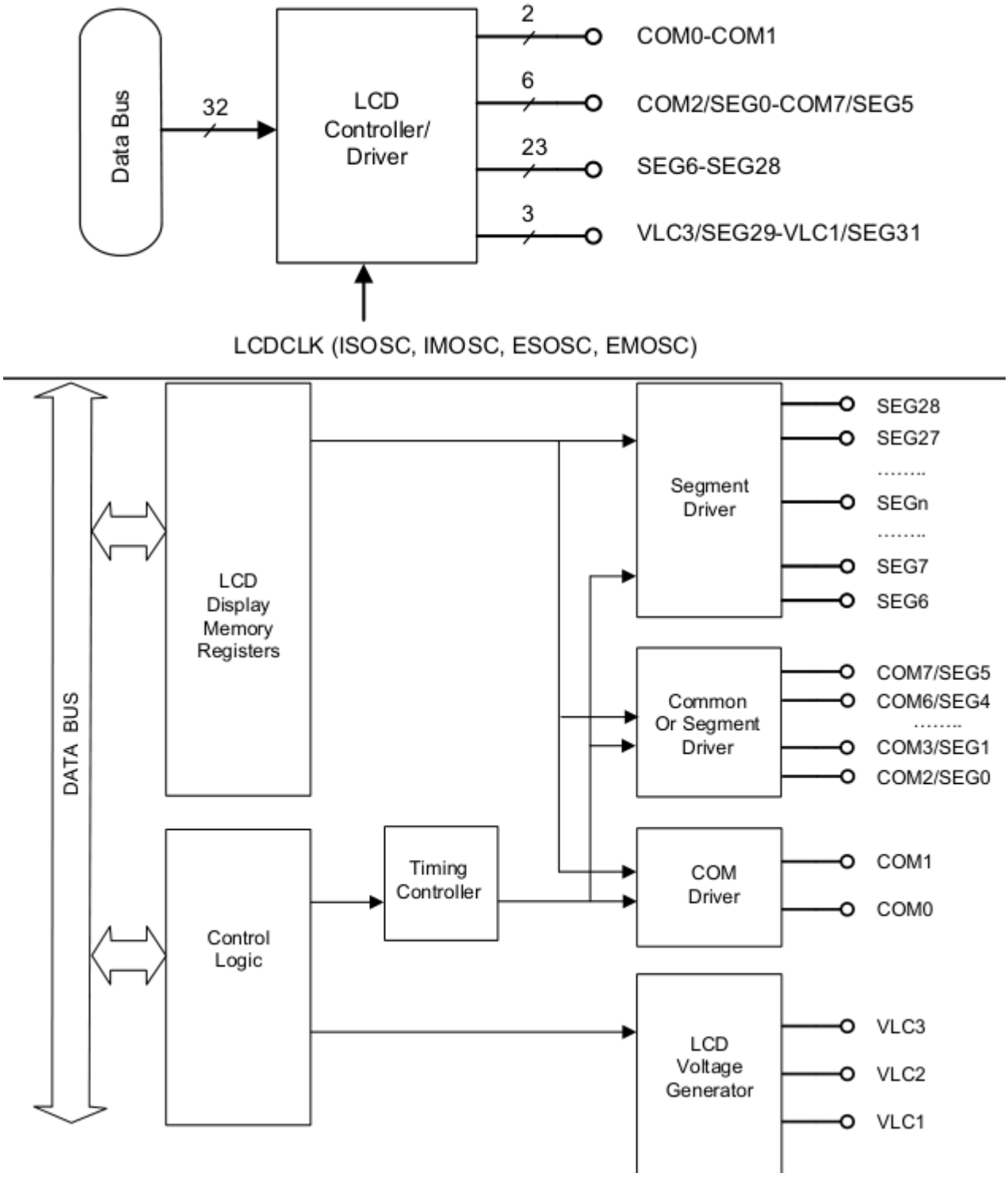

图 3.1.1 功能框图

- **管脚描述：**

Pin Name	Function	I/O Type	Note
COM[1:0]	LCD段（面板）公用端驱动信号	O	
COM[7:2]/SEG[5:0]	LCD段（面板）公用 / 区段端驱动信号	O	
SEG[28:6]	LCD段（面板）区段端驱动信号	O	
VLC3/SEG29	内部升压去耦电容引脚3 / LCD段（面板）区段端驱动信号	O	
VLC2/SEG30	内部升压去耦电容引脚2 / LCD段（面板）区段端驱动信号	O	
VLC1/SEG31	内部升压去耦电容引脚1 / LCD段（面板）区段端驱动信号	O	
VLCD	使用外部供电时的电压输入 使用内部供电时，需要接一个去耦电容C_{VLCD}	I	

图 3.1.2 管脚图

3.2 COM 和 SEG 多路复用

LCD 管脚 COM[7:2]/SEG[5:0]可用于 COM 和 SEG 多路复用，在不同占空比下，管脚多路复用如下表所示

管脚	占空比			
	1/2	1/3	1/4	1/8
COM[2]/SEG[0]	SEG[0]	COM[2]	COM[2]	COM[2]
COM[3]/SEG[1]	SEG[1]	SEG[1]	COM[3]	COM[3]
COM[4]/SEG[2]	SEG[2]	SEG[2]	SEG[2]	COM[4]
COM[5]/SEG[3]	SEG[3]	SEG[3]	SEG[3]	COM[5]
COM[6]/SEG[4]	SEG[4]	SEG[4]	SEG[4]	COM[6]
COM[7]/SEG[5]	SEG[5]	SEG[5]	SEG[5]	COM[7]

表 3.2.1 复用管脚

3.3LCD 频率及电源

时钟源必须稳定以获取精确的 LCD 时序，LCD 的时钟可以选择外部副时钟振荡器(ESOSC, 32.768KHz), 外部晶振(EMOSC), 内部主振荡器(IMOSC)和内部低速振荡器(ISOSC)。

LCD 的频率通过配置 LCD_CDR 中 CDIV[2:0] CPRE[15:0]。

帧速率计算：Fframe = LCDCLK /Duty / 2^(CDIV+1) /(CPRE+1)

举例：内部低速振荡器(ISOSC)27khz，选取 30HZ 则计算如下

Fframe = 27khz / 4 / 2^(CDIV+1) /(CPRE+1) 则 CDIV 取 2， CPRE 取 55.

LCD 电源可以来自内部 VDD 分压过来的内部升压转换器，或来自施加在 VLCD 引脚上的外部电压。

LCD 电源	内部升压转换器	外部电压
电压范围（VLCDMIN-VLCDMAX）	2.6-4V	2.6-4V
外部去耦	VLC1，VLC2，VLC3 管脚上添加去耦电容	

表 3.3.1 LCD 电源

3.4 LCD 配置

- **软件配置：**

1/4 占空比、1/3 偏置模式下 COM/SEG 驱动 LCD 屏软件初始配置如下

```
void lcd_config(void)
{
    int iRet = 0;
    //uint8_t i;
    uint32_t wSegMsk = 0x0Cfffffc;
    uint8_t   wComMsk = 0x0f;
    csi_lcd_config_t tLcdCfg;
    csi_lcd_gpio_init(wSegMsk, wComMsk);

    tLcdCfg.byClkSrc       = LCD_CLKSRC_ISOSC;
    tLcdCfg.byFreq         = 80;
    tLcdCfg.byVlcd         = LCD_VLCD_IN_4V0;
    tLcdCfg.byDutyBias     = LCD_DUTY1_4_BIAS1_3;
    tLcdCfg.byDead         = LCD_DEAD_2PHASE;
    tLcdCfg.byDrvNet       = LCD_DRVNET_PWLEV1;
```

```
    tLcdCfg.byDpEn      = DISABLE;
    tLcdCfg.byInt       = LCD_INTSRC_NONE;

    iRet = csi_lcd_init(LCD, &tLcdCfg);
    if(iRet == CSI_OK)
        csi_lcd_start(LCD);
    //csi_lcd_set_blink(LCD, LCD_BLINK_SEG8, LCD_BLINK_FRE_F2, 2);
    csi_lcd_write_data(LCD, bySendBata, 2, 22);
    for(i = 2; i < 24; i++)
    {
        bySendBata[i] = 0x0f;
    }
    csi_lcd_write_data(LCD, bySendBata, 2, 22);
    //
    lcd_rtc_data(bySendBata, 2, 20/10);                 //hour
    lcd_rtc_data(bySendBata, 4, 20%10);                 //hour
    lcd_rtc_data(bySendBata, 6, 45/10);                 //min
    lcd_rtc_data(bySendBata, 8, 45%10);                 //min

    csi_lcd_write_data(LCD, bySendBata, 2, 22);
}
```

- **代码说明：**

  csi_lcd_gpio_init(); ----用于配置 LCD GPIO

  csi_lcd_init(); ----用于初始化 LCD 参数

  csi_lcd_start(); ----用于启动 LCD 模块

  csi_lcd_write_data(); ----用于写入显示数据

  lcd_rtc_data(); ----用于数据转换

- **函数参数说明：**

  csi_lcd_gpio_init(wSegMsk, wComMsk);

  uint32_t wSegMsk = 0x0Cffffc; //lcd seg2->26

  uint8_t wComMsk = 0x0f; //lcd com0->30

  LCD GPIO 配置：com0-com3 seg2-seg31 4*30

csi_lcd_init(LCD, &tLcdCfg);

tLcdCfg.byClkSrc = LCD_CLKSRC_ISOSC; //LCD 时钟源选择 ISOSC

tLcdCfg.byFreq = 80; //LCD 刷新频率 =80Hz，选择范围：30~100Hz

tLcdCfg.byVlcd = LCD_VLCD_IN_4V0; //VLCD 选择内部 4.0V

tLcdCfg.byDutyBias = LCD_DUTY1_4_BIAS1_3; //LCD Duty = 1/4, Bias = 1/3

tLcdCfg.byDead = LCD_DEAD_2PHASE; //LCD 帧死区控制，选择 2 个相位周期，死区可以控制对比度

tLcdCfg.byDrvNet = LCD_DRVNET_PWLEV1; //LCD 驱动网络(电阻网络)，共有 4 档，功耗 PWLEV0 > PWLEV1 > PWLEV2 > PWLEV3

tLcdCfg.byDpEn = DISABLE; //去耦电容使能控制

tLcdCfg.byInt = LCD_INTSRC_NONE; //是否用中断，不用

模块源　显示数据　显示个数

csi_lcd_write_data(LCD, VarSendBata, 2, 22);

- **显示效果：**

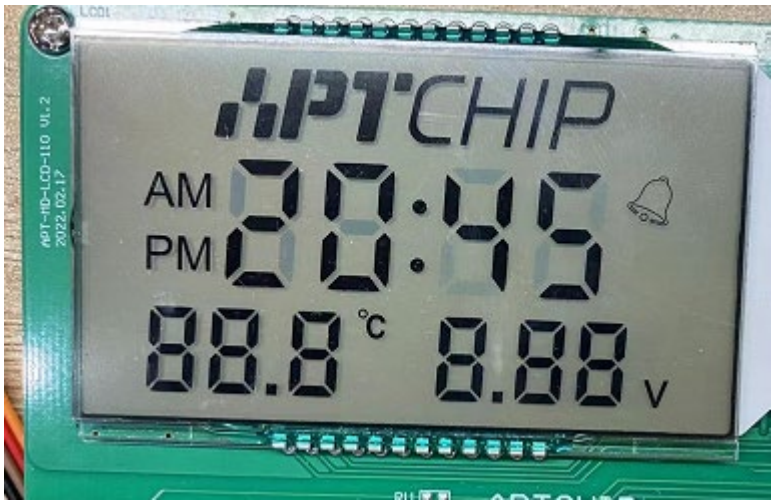

图 3.4.1 显示图

COM 波形

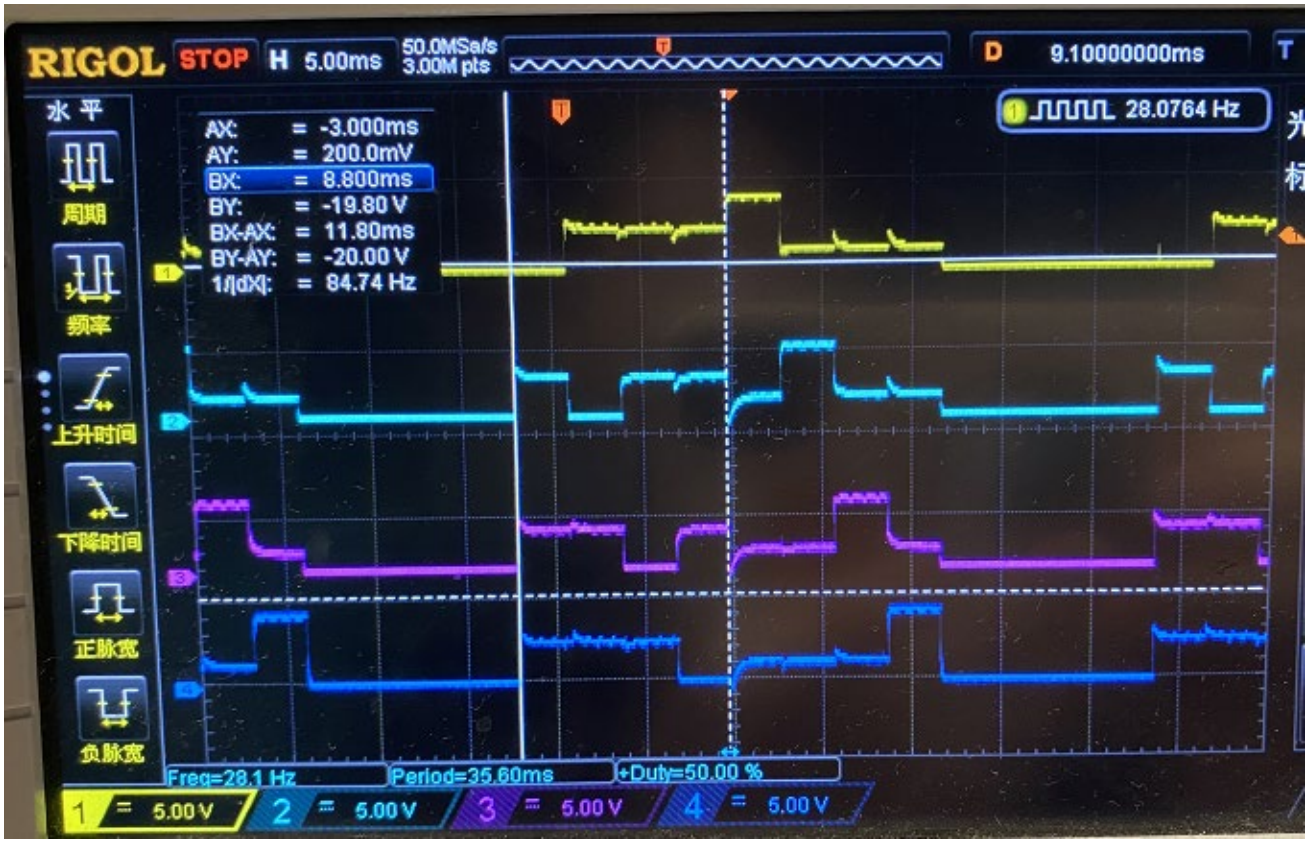

图 3.4.2 显示图

3.4 LCD 低功耗下运行

配置 LCD 及 RTC，正常工作模式下切换到 snooze 模式,通过 RTC 1MIN 中断进行唤醒一次。（SNOOZE 模式下 CPU 时钟被关闭，除了 TOUCH 和 LCD 可以配置开关以及唤醒源保持工作外，所有外设模块都被关闭。片上 SRAM 停止供电，不掉电 SRAM 保持供电。唤醒后芯片会重新复位）

```
int lcd_disp_rtc_snooze_demo(void)
{
    int iRet = 0;
    uint8_t i;
    uint32_t wSegMsk = 0x03fffffc;          //lcd seg2->26
    uint8_t  wComMsk = 0x0f;                //lcd com0->3

    uint16_t hwRstSrc;

    csi_pm_mode_e ePmMode = PM_MODE_SNOOZE;             //PM_MODE_SNOOZE/PM_MODE_DEEPSLEEP
```

```
csi_lcd_config_t tLcdCfg;
csi_rtc_config_t tRtcConfig;
csi_rtc_time_t tRtcTime,tRtcTimeRdbk;

csi_pin_set_mux(PB02, PB02_OUTPUT);                    //PB02 OUTPUT
csi_pin_set_high(PB02);
csi_pin_set_mux(PA05, PA05_OUTPUT);                    //PA05 OUTPUT

csi_pin_toggle(PA05);
mdelay(250);
csi_pin_toggle(PA05);
mdelay(250);
csi_pin_toggle(PA05);
mdelay(250);
csi_pin_toggle(PA05);
mdelay(250);
csi_pin_toggle(PA05);
mdelay(250);
csi_pin_toggle(PA05);
mdelay(250);
csi_pin_toggle(PA05);
mdelay(250);
csi_pin_toggle(PA05);
mdelay(250);
hwRstSrc = csi_get_rst_reason();
my_printf("System Reset Source = 0x%x \n", hwRstSrc);
if(!(hwRstSrc & RST_SRC_SNOOZE_WKUP))
{
        tRtcConfig.byClkSrc = RTC_CLKSRC_ISOSC;
        tRtcConfig.byFmt = RTC_24FMT;
        csi_rtc_init(RTC, &tRtcConfig);

        tRtcTime.iYear = 21;
        tRtcTime.iMon   = 12;
        tRtcTime.iMday = 9;
        tRtcTime.iHour = 15;
        tRtcTime.iMin   = 50;
        tRtcTime.iSec   = 59;
        csi_rtc_set_time(RTC, &tRtcTime);
        csi_rtc_start_as_timer(RTC, RTC_TIMER_1S);

        csi_rtc_start(RTC);
        my_printf("Init Rtc Complete!\n");
}
```

```
else
{
        csi_rtc_start_as_timer(RTC, RTC_TIMER_1MIN);
}
csi_pin_set_low(PB02);
csi_clr_rst_reason(hwRstSrc);

csi_pm_config_wakeup_source(WKUP_RTC, ENABLE);
csi_pm_snooze_power_enable(SNOOZE_LCD_POWER, ENABLE);
csi_lcd_gpio_init(wSegMsk, wComMsk);
tLcdCfg.byClkSrc      = LCD_CLKSRC_ISOSC;
tLcdCfg.byFreq        = 80;
tLcdCfg.byVlcd        = LCD_VLCD_IN_3V2;
tLcdCfg.byDutyBias    = LCD_DUTY1_4_BIAS1_3;
tLcdCfg.byDead        = LCD_DEAD_NONE;
tLcdCfg.byDrvNet      = LCD_DRVNET_PWLEV1;
tLcdCfg.byDpEn        = DISABLE;
tLcdCfg.byInt         = LCD_INTSRC_NONE;
iRet = csi_lcd_init(LCD, &tLcdCfg);
if(iRet == CSI_OK)
{
        csp_lcd_en(LCD);
        my_printf("LCD Init Succeed!\n");
}
for(i = 2; i < 26; i++)
{
        bySendBata[i] = 0x0f;
}
csi_lcd_write_data(LCD, bySendBata, 2, 22);
my_printf("Light up all!\n");
mdelay(500);
do
{
        csi_rtc_get_time(RTC,   &tRtcTimeRdbk);
        for(i = 2; i < 26; i++)
        {
                if(7 == i)
                {
                        bySendBata[i] |= 0x07;

                        if(bySendBata[7] & 0x08)
                                bySendBata[7] &= 0x07;
                        else
                                bySendBata[7] |= 0x08;
```

```
                }
                else
                        bySendBata[i] = 0x0f;
        }

        lcd_rtc_data(bySendBata, 2, tRtcTimeRdbk.iMin/10);
        lcd_rtc_data(bySendBata, 4, tRtcTimeRdbk.iMin%10);
        lcd_rtc_data(bySendBata, 6, tRtcTimeRdbk.iSec/10);
        lcd_rtc_data(bySendBata, 8, tRtcTimeRdbk.iSec%10);

        csi_lcd_write_data(LCD, bySendBata,2,22);

        if(!((tRtcTimeRdbk.iSec+1)%2))
        {
                my_printf("Rtc Min = %d, Sec = %d \n", tRtcTimeRdbk.iMin, tRtcTimeRdbk.iSec);
                switch(ePmMode)
                {
                        case PM_MODE_SLEEP:
                                my_printf("Enter Sleep Mode\n");
                                break;
                        case PM_MODE_DEEPSLEEP:
                                my_printf("Enter Deep-Sleep mode\n");
                                break;
                        case PM_MODE_SNOOZE:
                                my_printf("Enter Snooze Mode\n");
                                break;
                        case PM_MODE_SHUTDOWN:
                                my_printf("Enter ShutDown Mode\n");
                                break;
                        default:
                                break;
                }
                csi_pin_set_high(PA05);
                csi_pin_set_high(PB02);
                csi_pm_enter_sleep(ePmMode);
                mdelay(100);
                csi_pin_set_low(PA05);
                mdelay(100);
        }

        mdelay(500);

}while(1);
return iRet;
```

```
}
```

- **代码说明：**

  csi_pm_enter_sleep()； ----用于配置休眠模式

  csi_pm_config_wakeup_source(); ----用于配置唤醒

  csi_pm_snooze_power_enable(); ----用于配置 SNOOZE 模式下运行

- **函数参数说明：**

  csi_pm_enter_sleep(PM_MODE_SNOOZE);

  PM_MODE_LPRUN //LowPower Running mode

  PM_MODE_SLEEP // Sleep mode

  PM_MODE_DEEPSLEEP // Normal DeepSleep mode

  PM_MODE_SNOOZE // Snooze mode of DeepSleep

  PM_MODE_SHUTDOWN // ShutDown mode of DeepSleep

4. 程序下载和运行

1. 将目标板与仿真器连接，分别为 VDD SCLK SWIO GND
2. 程序编译后仿真运行
3. 通过 LCD Demo 板查看显示，如图 3.4.1 所示