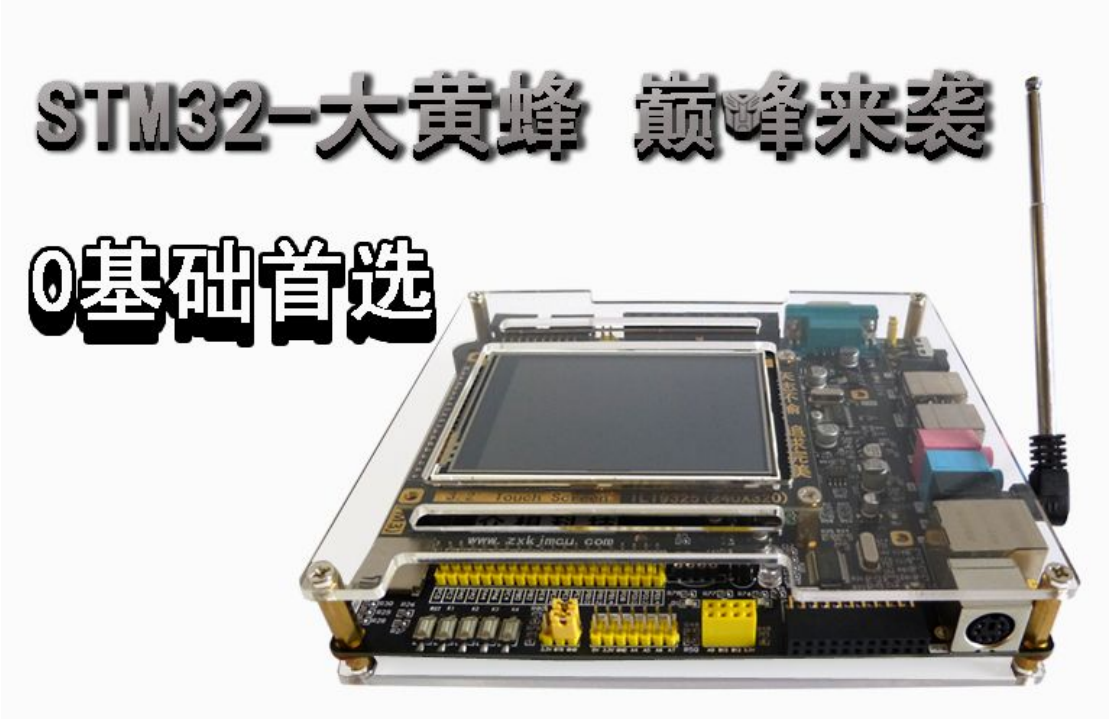
学 ARM 从 STM32 开始

STM32 开发板库函数教程--实战篇



官方网站：http://www.zxkjmcu.com
官方店铺：http://zxkjmcu.taobao.com
官方论坛：http://bbs.zxkjmcu.com
刘洋课堂：http://school.zxkjmcu.com

4.32 STM32 LCD 彩色液晶屏显示汉字、英文、数字实验

4.32.1 概述

在 LCD 液晶屏上，我们把要显示的汉字、英文、数字、图案等所要经过的像素点改变成与背景色不同的颜色，我们的眼睛就会看到我们想要得到的效果。这是实现的原理，如果我们要想实现 LCD 液晶屏的显示，我们刚开始要做的就是要建立字模型（取字模）。

4.32.2 在软件中建立字模型

我们的目的是实用软件驱动 LCD 液晶屏显示，我们首先要做的是把要改变颜色的像素点的坐标取出来，按照程汉字结构制作成列表，每一个汉字、每一个字母、每一个数字都制作成一个表格，我们需要使用哪个字模型，就可以直接调用。想要建立字模型，首先要规定汉字、字母、数字的占用像素点的大小。

4.32.2.1 字模型结构和大小

按照标准规定，字模型占用的像素点的大小一般是有习惯规定的，汉字、字母、数字占用的像素点数量描述如下：

1、汉字字模 24×24（16×16）

2、字母字模 12×24

3、数字字模 12×24

4.32.2.2 取字模

人工查像素点是很麻烦的一件事情，现在有一款专用取字模型软件，节省了我们很大部分的工作，下面我们介绍一下这款软件，并使用它获取字模代码。在我们随机光盘中有这款软件具体位置是，《外设篇 18.LCD 彩色液晶屏显示汉字、英文、数字(代码课件)/软件/PCtoLCD2002 完美版》。打开后如“图 4.32.1 PCtoLD2002 取字模型软件界面”所示。红色框图 1 区域可以设置字模像素点的大小；红色框图 2 区域显示输入字符；红色框图 3 区域是操作按钮；红色框图 4 区域是取字模型后生成的十六进制代码。

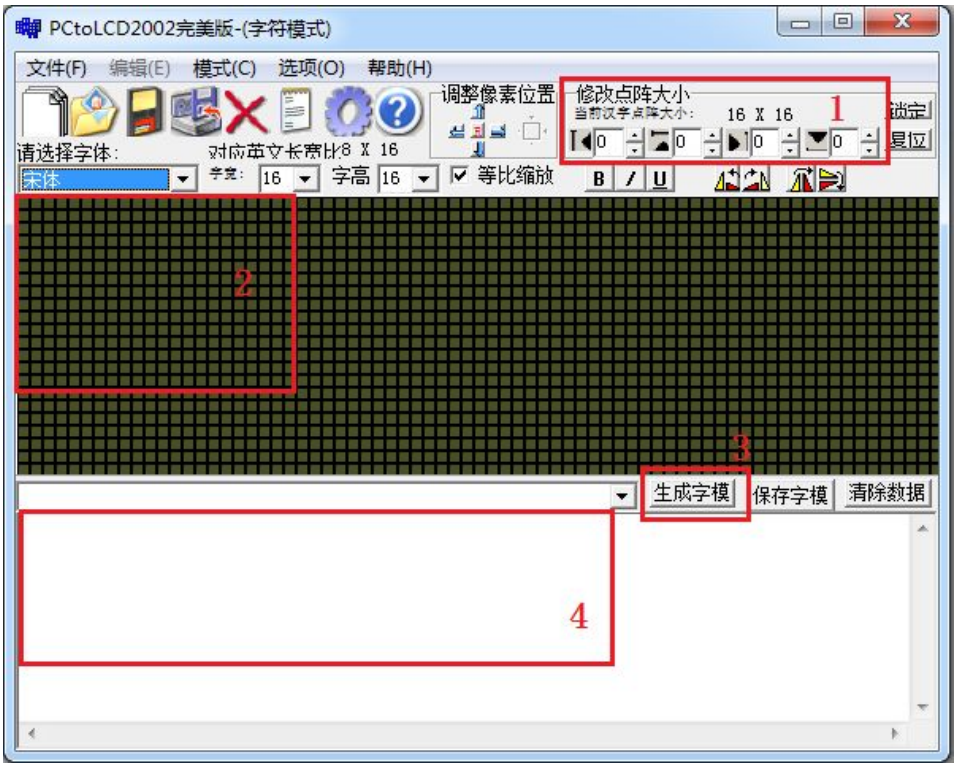


图 4.32.1 PCtoLD2002 取字模型软件界面

在工具条中点击  图标，进入设置界面，如“图 4.32.2 PCtoLD2002 取字模型软件设置界面”，完成相关设置。

图 4.32.2　PCtoLD2002 取字模型软件设置界面

下面我们来取字模型。在输入区输入汉字“众”，点击“生成字模”按钮，在代码生成区就会有汉字“众”的字模十六进制代码。如“图 4.32.3 PCtoLD2002 取字模型软件使用”。

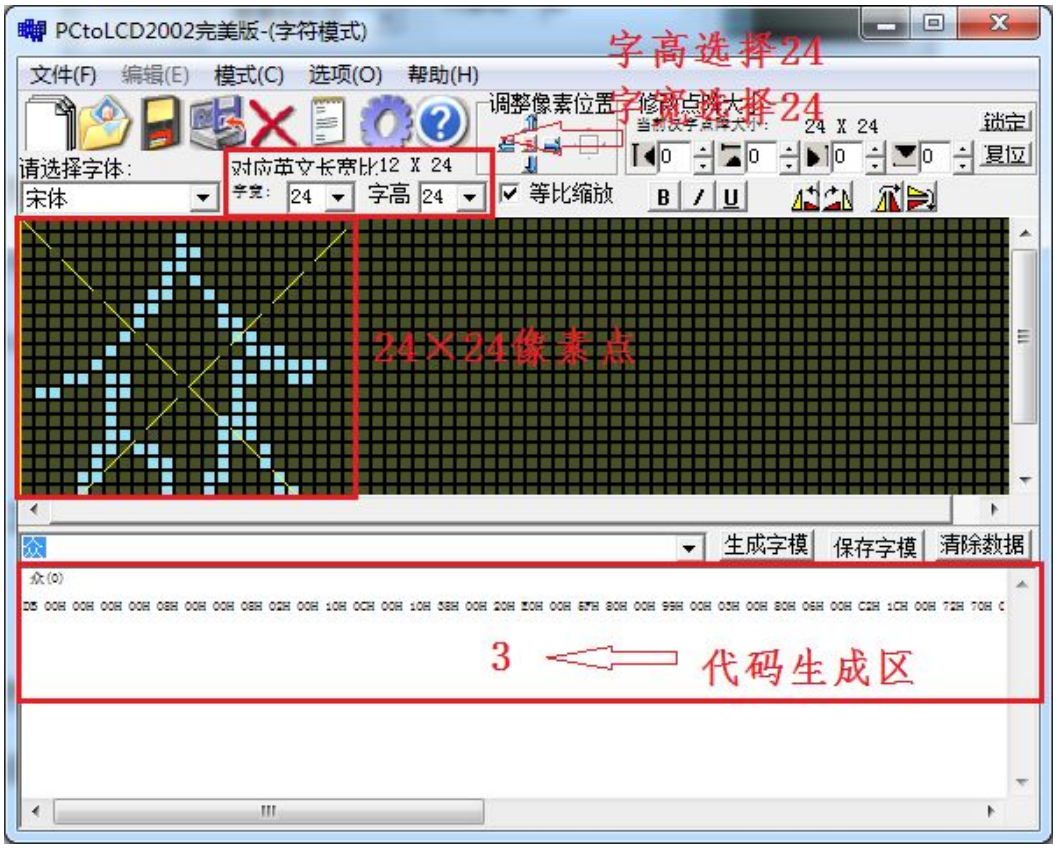


图 4.32.3　PCtoLD2002 取字模型软件使用

汉字“众”生成的代码是：

{0x00, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x02, 0x00, 0x10, 0x0C, 0x00, 0x10, 0x38, 0x00, 0x20, 0xE0, 0x00, 0x5F, 0x80, 0x00, 0x99, 0x00, 0x03, 0x00, 0x80, 0x06, 0x00, 0xC2, 0x1C, 0x00, 0x72, 0x70, 0x00, 0x04, 0x28, 0x00, 0x08, 0x04, 0x00, 0x30, 0x03, 0x03, 0xE0, 0x01, 0xBF, 0x00, 0x00, 0xD7, 0x00, 0x00, 0x60, 0xC0, 0x00, 0x60, 0x30, 0x00, 0x30, 0x18, 0x00, 0x30, 0x0E, 0x00, 0x20, 0x06, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00}, /*"众", 0*/

大括号内一共是 72 个十六进制字，他是汉字“众”的字模。

4.32.3 实验目的

我们要把“众想科技”四个汉字送到触摸屏显示；还要把“http://zxkjmcu.taobao.com”送到触摸屏显示。

4.32.4 硬件设计

触摸屏和 CPU 之间采用的是 SPI 通讯方式，从硬件电路设计上可以看出，外扩触摸屏和大黄蜂实验板通过 40 位的插座产生电气连接。具体的引脚规定在《4.29 STM32 外设篇-LCD 彩色液晶屏工作原理》中已经做了详细的介绍。

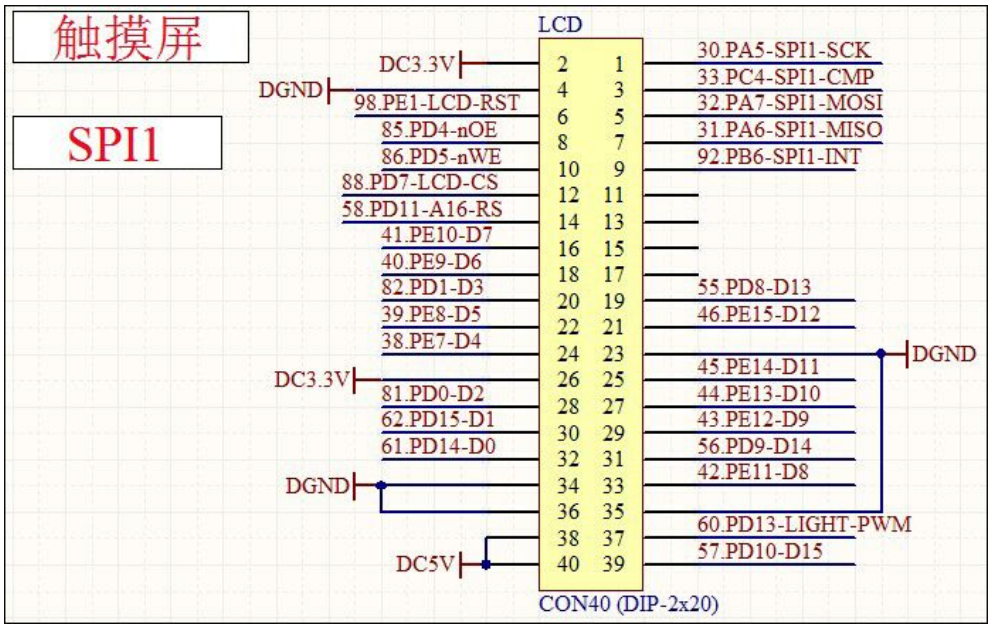


图 4.31.1　外扩触摸屏接口槽原理图

4.32.5 软件设计

4.32.5.1 软件设计说明

1、采用 SPI 通讯方式。

2、这套程序严格按照 SPI 的工作时序编写。

在这节程序设计中，用到了外部中断函数；USART 串口通讯函数；定时器函数等。

4.32.5.2 STM32 库函数文件

```
stm32f10x_gpio.c
stm32f10x_rcc.c
Misc.c  // 中断控制字（优先级设置）库函数
stm32f10x_exti.c // 外部中断库处理函数
stm32f10x_tim.c  // 定时器库处理函数
stm32f10x_usart.c  // 串口通讯函数
stm32f10x_fsmc.c  // FSMC 通讯函数
```

以上库文件包含了本次实验所有要用到的函数功能。

4.32.5.3 自定义头文件

```
pbdata.h
pbdata.c
```

我们已经创建了两个公共的文件，这两个文件主要存放我们自定义的公共函数和全局变量，以方便以后每个功能模块之间传递参数。

4.32.5.4 pbdata.h 文件里的内容是

```
#ifndef _pbdata_H
#define _pbdata_H

#include "stm32f10x.h"
#include "misc.h"
#include "stm32f10x_exti.h"
#include "stm32f10x_tim.h"
#include "stm32f10x_usart.h"
#include "stm32f10x_spi.h"
#include "stdio.h"
#include "stm32_fsmc.h" //自定义的 stm32_fsmc 专属函数
#include "lcd_ILI9325.h" //自定义的 LCD-ILI9325 专属函数

extern u8 dt;       //定义中间变量

void RCC_HSE_Configuration(void);  //定义函数
void delay(u32 nCount);          //定义函数
void delay_us(u32 nus);          //定义函数
void delay_ms(u16 nms);          //定义函数

#endif
```

语句 #ifndef、#endif 是为了防止 pbdata.h 文件被多个文件调用时出现错误提示。如果不加这两条语句，当两个文件同时调用 pbdata 文件时，会提示重复调用错误。“stm32_fsmc.h”和“lcd_ILI9325.h”是我们自定义的，为了是在大程序设计中方便移植和功能分类而独立新建。

4.32.5.5 pbdata.c 文件里的内容是

下面是 pbdata.c 文件详细内容，在文件开始还是引用“pbdata.h”文件。

```
#include "pbdata.h"
```

```
    u8 dt=0;    //中间变量赋值

    void RCC_HSE_Configuration(void) //HSE 作为 PLL 时钟，PLL 作为 SYSCLK
    {
       RCC_DeInit(); /*将外设 RCC 寄存器重设为缺省值 */
       RCC_HSEConfig(RCC_HSE_ON);    /*设置外部高速晶振（HSE） HSE 晶振打开
(ON)*/

       if(RCC_WaitForHSEStartUp() == SUCCESS) {  /*等待 HSE 起振,  SUCCESS：HSE
晶振稳定且就绪*/

       RCC_HCLKConfig(RCC_SYSCLK_Div1);/*设置 AHB 时钟(HCLK)RCC_SYSCLK_Div1——
—AHB 时钟 = 系统时*/
       RCC_PCLK2Config(RCC_HCLK_Div1); /*设置高速 AHB 时钟(PCLK2)RCC_HCLK_Div1
———APB2 时钟 = HCLK*/
       RCC_PCLK1Config(RCC_HCLK_Div2); /*设置低速 AHB 时钟(PCLK1)RCC_HCLK_Div2
———APB1 时钟 = HCLK / 2*/

       RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);/*设置 PLL 时钟源及
倍频系数*/
       RCC_PLLCmd(ENABLE);    /*使能 PLL */
       while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET) ; /*检查指定的 RCC
标志位(PLL 准备好标志)设置与否*/

       RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);  /*设置系统时钟（SYSCLK） */
       while(RCC_GetSYSCLKSource() != 0x08);      /*0x08：PLL 作为系统时钟 */
     }
    }

    void delay(u32 nCount)
    {
       for(;nCount!=0;nCount--);
    }
    /*******************************************************************************
    * 名     称：delay_us(u32 nus)
    * 功     能：微秒延时函数
    * 入口参数：u32  nus
    * 出口参数：无
    * 说     明：
    * 调用方法：无
    *******************************************************************************/
    void delay_us(u32 nus)
    {
        u32 temp;
```

```
    SysTick->LOAD = 9*nus;
    SysTick->VAL=0X00;//清空计数器
    SysTick->CTRL=0X01;//使能，减到零是无动作，采用外部时钟源
    do
    {
     temp=SysTick->CTRL;//读取当前倒计数值
    }while((temp&0x01)&&(!(temp&(1<<16))));//等待时间到达

    SysTick->CTRL=0x00; //关闭计数器
    SysTick->VAL =0X00; //清空计数器
}

/*****************************************************************************
* 名    称：delay_ms(u16 nms)
* 功    能：毫秒延时函数
* 入口参数：u16 nms
* 出口参数：无
* 说    明：
* 调用方法：无
*****************************************************************************/
void delay_ms(u16 nms)
{
     //注意 delay_ms 函数输入范围是 1-1863
    //所以最大延时为 1.8 秒
    u32 temp;
    SysTick->LOAD = 9000*nms;
    SysTick->VAL=0X00;//清空计数器
    SysTick->CTRL=0X01;//使能，减到零是无动作，采用外部时钟源
    do
    {
     temp=SysTick->CTRL;//读取当前倒计数值
    }while((temp&0x01)&&(!(temp&(1<<16))));//等待时间到达
    SysTick->CTRL=0x00; //关闭计数器
    SysTick->VAL =0X00; //清空计数器
}
```

4.32.6 STM32 系统时钟配置 SystemInit()

每个工程都必须在开始时配置并启动 STM32 系统时钟，这点很重要。

4.32.7 GPIO 引脚时钟使能

```
    SystemInit();//72m
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD,ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE,ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE);//设置串口 1 时钟
使能
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);//功能复用 IO 时钟
使能
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_FSMC,ENABLE);//功能复用 FSMC 时钟
使能
    }
```

本节实验用到了 PA 端口、PD 端口、PE 端口，所以要打开这些端口的使能；串口 1 时钟源是通过 APB2 预分频器得到的，串口 1 时钟初始化；因为要与外部芯片通讯，所以要打开功能复用时钟；重要的是一定要打开 FSMC 功能服用，FSMC 时钟是 AHB 产生，这点要注意。

4.32.8 GPIO 管脚电平控制函数

在主程序中采用 while(1)循环语句，每次进入 while(1)循环语句就是延时 1 秒就退出循环。

```
  while(1)
   {
      delay_ms(200);
      TFT_Draw_Rectangle(10,150,230,160,RED);
      delay_ms(200);
      TFT_Draw_Rectangle(10,150,230,160,BLUE);
      delay_ms(200);
      TFT_Draw_Rectangle(10,150,230,160,GREEN);
   }
```

4.32.9 stm32f10x_it.c 文件里的内容

在中断处理 stm32f10x_it.c 文件里中有串口 1 子函数。

```
#include "stm32f10x_it.h"
```

```
#include "stm32f10x_exti.h"
#include "stm32f10x_rcc.h"
#include "misc.h"
#include "pbdata.h"

void NMI_Handler(void)
{
}

void USART1_IRQHandler(void)
{
   if(USART_GetITStatus(USART1,USART_IT_RXNE)!=RESET)
   {
      USART_SendData(USART1,USART_ReceiveData(USART1));
      while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);
   }
}
```

4.32.10 stm32_fsmc.h 文件里的内容是

函数 stm32_fsmc.h 在这里是为了声明 FSMC 模式形式，方便其他程序调用。它是一个通用的 FSMC 声明函数。stm32_fsmc.h 的内容如下：

```
#ifndef _STM32_FSMC_H
#define _STM32_FSMC_H

#include "pbdata.h"
void FSMC_Configuration(void);//声明 FSMC 初始化函数
#endif
```

4.32.11 stm32_fsmc.c 文件里的内容是

自定义函数 stm32_fsmc.c 函数中，它是配置 FSMC 功能的核心程序，基本的时序都体现在这个程序中。这段程序编写起来很复杂，需要查找很多资料，想在网络发达，我们在网上找到了一份比较详细的关于 FSMC 功能配置的说

明，请参考附件《FSMC 配置》文档。在这里函数 stm32_fsmc.c 其内容如下：

```
#include "pbdata.h"

void FSMC_Configuration(void)
{
  FSMC_NORSRAMInitTypeDef  FSMC_NORSRAMInitStructure;
  FSMC_NORSRAMTimingInitTypeDef  p;

  p.FSMC_AddressSetupTime = 0x02;//地址建立时间
  p.FSMC_AddressHoldTime = 0x00;//地址保持时间
  p.FSMC_DataSetupTime = 0x05;//数据建立时间
  p.FSMC_BusTurnAroundDuration = 0x00;//总线恢复时间
  p.FSMC_CLKDivision = 0x00;//时钟分频
  p.FSMC_DataLatency = 0x00;//数据保持时间
  p.FSMC_AccessMode = FSMC_AccessMode_B;

  //NOR FLASH 的 BANK1
  FSMC_NORSRAMInitStructure.FSMC_Bank = FSMC_Bank1_NORSRAM1;
  //数据线与地址线不复用
  FSMC_NORSRAMInitStructure.FSMC_DataAddressMux =
FSMC_DataAddressMux_Disable;
  //存储器类型 NOR FLASH
  FSMC_NORSRAMInitStructure.FSMC_MemoryType = FSMC_MemoryType_NOR;
  //数据宽度为 16 位
  FSMC_NORSRAMInitStructure.FSMC_MemoryDataWidth =
FSMC_MemoryDataWidth_16b;
  //使用异步写模式，禁止突发模式
  FSMC_NORSRAMInitStructure.FSMC_BurstAccessMode =
FSMC_BurstAccessMode_Disable;
  //本成员的配置只在突发模式下有效，等待信号极性为低
  FSMC_NORSRAMInitStructure.FSMC_WaitSignalPolarity =
FSMC_WaitSignalPolarity_Low;
  //禁止非对齐突发模式
  FSMC_NORSRAMInitStructure.FSMC_WrapMode = FSMC_WrapMode_Disable;
  //本成员配置仅在突发模式下有效。NWAIT 信号在什么时期产生
  FSMC_NORSRAMInitStructure.FSMC_WaitSignalActive =
FSMC_WaitSignalActive_BeforeWaitState;
  //本成员的配置只在突发模式下有效，禁用 NWAIT 信号
  FSMC_NORSRAMInitStructure.FSMC_WriteOperation =
FSMC_WriteOperation_Enable;
  //禁止突发写操作
  FSMC_NORSRAMInitStructure.FSMC_WaitSignal = FSMC_WaitSignal_Disable;
```

```
    //写使能
    FSMC_NORSRAMInitStructure.FSMC_ExtendedMode =
FSMC_ExtendedMode_Disable;
    //禁止扩展模式，扩展模式可以使用独立的读、写模式
    FSMC_NORSRAMInitStructure.FSMC_WriteBurst = FSMC_WriteBurst_Disable;
    //配置读写时序
    FSMC_NORSRAMInitStructure.FSMC_ReadWriteTimingStruct = &p;
    //配置写时序
    FSMC_NORSRAMInitStructure.FSMC_WriteTimingStruct = &p;

    //初始化 FSMC
    FSMC_NORSRAMInit(&FSMC_NORSRAMInitStructure);
    //使能 FSMC BANK1_SRAM 模式
    FSMC_NORSRAMCmd(FSMC_Bank1_NORSRAM1, ENABLE);
}
```

4.32.12 lcd_ILI9325.h 文件里的内容是

函数 lcd_ili9325.h 在这里是针对于 ILI9325 芯片自定义的一份专用于这款芯片的头文件。在 lcd_ili9325.h 中我们要定义几个地址，具体说明在程序注释中。lcd_ili9325.h 的内容如下：

```
#ifndef _LCD_ILI9325_H
#define _LCD_ILI9325_H
#include "pbdata.h"

//6000 0000
//A16  16 位通讯

#define Bank1_LCD_D (u32)0x60020000 //命令位是“1”，所以在 16 位通讯命令格式时地址是 2000
#define Bank1_LCD_C (u32)0x60000000 //数据位是“0”，所以在 16 位通讯数据格式时地址是 0

#define RGB565(r, g, b) ((r >> 3) << 11 | (g >> 2) << 5 | (b >> 3))

#define RED   RGB565(255,0,0)      //红色
#define GREEN RGB565(0,255,0)      //绿色
#define BLUE  RGB565(0,0,255)      //蓝色
#define BLACK RGB565(0,0,0)        //黑色
```

```
#define WHITE RGB565(255,255,255)  //白色

void ILI9325_Init(void);
void LCD_WR_REG(u16 index);
void LCD_WR_Data(u16 val);
void LCD_WR_CMD(u16 index,u16 val);

void ILI_9325_Draw_Point(u8 x,u16 y,u16 color);
void ILI_9325_CLEAR(u16 color);

void show_Font(u16 x,u16 y,u8 id,u16 qj,u16 bj);
void show_Str(u16 x,u16 y,u8 id,u16 qj,u16 bj);
void TFT_Draw_Rectangle(u16 x1,u16 y1,u16 x2,u16 y2,u16 color);

#endif
```

4.32.13 lcd_ILI9325.c 文件里的内容是

函数 lcd_ILI9325.c 在这里是针对于 ILI9325 芯片自定义的一份专用于这款芯片的库函数。他是读写 ILI9325 芯片的核心程序，基本的时序都体现在这个程序中。每一款新的触摸屏生产出来后，生产厂家都会为这款新产品配好了初始化子函数，我们在做开发程序的时候，直接使用就可以。在 lcd_ILI9325.c 中就包含了厂家针对于这款产品开发的初始化子函数。

lcd_ILI9325.c 的内容如下：

```
#include "pbdata.h"
const u8 font[4][72]={
{0x00,0x00,0x00,0x00,0x08,0x00,0x00,0x08,0x02,0x00,0x10,0x0C,0x00,0x10,0x38,0x00,0x20,0xE0,0x00,0x5F,0x80,0x00,0x99,0x00,0x03,0x00,0x80,0x06,0x00,0xC2,0x1C,0x00,0x72,0x70,0x00,0x04,0x28,0x00,0x08,0x04,0x00,0x30,0x03,0x03,0xE0,0x01,0xBF,0x00,0x00,0xD7,0x00,0x00,0x60,0xC0,0x00,0x60,0x30,0x00,0x30,0x18,0x00,0x30,0x0E,0x00,0x20,0x06,0x00,0x00,0x04,0x00,0x00,0x00},/*"众",0*/
{0x00,0x00,0x00,0x00,0x02,0x00,0x02,0x04,0x00,0x02,0x08,0x0C,0x02,0x30,0x38,0x02,0xE0,0x00,0x7F,0xFE,0x00,0x3F,0xFE,0x00,0x02,0xC0,0xFC,0x06,0x60,0x04,0x02,0x20,0x04,0x00,0x01,0x04,0x1F,0xFE,0xC4,0x09,0x24,0xE4,0x09,0x24,0x04,0x09,0x24,0x04,0x09,0x24,0x04,0x09,0x24,0x3C,0x09,0x24,0x00,0x1F,0xFE,0x40,0x1F,0xFE,0x30,0x00,0x00,0x18,0x00,0x00,0x00,0x00,0x00,0x00},/*"想",1*/
```

```
    {0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x08, 0x40, 0x40, 0x08, 0x41, 0x80, 0x08, 0x46, 0
x00, 0x08, 0x5C, 0x00, 0x1F, 0xFF, 0xFE, 0x1F, 0xFF, 0xFE, 0x10, 0x48, 0x00, 0x30, 0x8C, 0x
00, 0x10, 0x86, 0x80, 0x00, 0x00, 0x80, 0x00, 0x20, 0x80, 0x04, 0x10, 0x80, 0x03, 0x1C, 0x8
0, 0x03, 0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x01, 0x00, 0x3F, 0xFF, 0xFE, 0x20, 0x01, 0x00,
0x00, 0x02, 0x00, 0x00, 0x06, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00},/*"科",2*/
    {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x06, 0x00, 0x01, 0x04, 0x04, 0x01, 0x04, 0
x06, 0x7F, 0xFF, 0xFE, 0x3F, 0xFF, 0xFC, 0x21, 0x08, 0x00, 0x02, 0x10, 0x02, 0x02, 0x10, 0x
02, 0x01, 0x10, 0x02, 0x01, 0x10, 0x04, 0x01, 0x1C, 0x08, 0x01, 0x13, 0x08, 0x01, 0x10, 0xD
0, 0x3F, 0xF0, 0x30, 0x21, 0x10, 0x70, 0x01, 0x11, 0xC8, 0x01, 0x17, 0x0C, 0x01, 0x1C, 0x04,
0x02, 0x10, 0x06, 0x02, 0x00, 0x06, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00}/*"技",3*/
    };

    const u8 str[15][36]={
    {0x00, 0x20, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x3F, 0xE0, 0x00, 0x23, 0xF8, 0x00, 0x00, 0
xE0, 0x00, 0x27, 0x00, 0x00, 0x3E, 0x00, 0x00, 0x3F, 0xE0, 0x00, 0x21, 0xF8, 0x00, 0x01, 0x
E0, 0x00, 0x3E, 0x00, 0x00, 0x20, 0x00},/*"w",0*/
    {0x00, 0x20, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x3F, 0xE0, 0x00, 0x23, 0xF8, 0x00, 0x00, 0
xE0, 0x00, 0x27, 0x00, 0x00, 0x3E, 0x00, 0x00, 0x3F, 0xE0, 0x00, 0x21, 0xF8, 0x00, 0x01, 0x
E0, 0x00, 0x3E, 0x00, 0x00, 0x20, 0x00},/*"w",1*/
    {0x00, 0x20, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x3F, 0xE0, 0x00, 0x23, 0xF8, 0x00, 0x00, 0
xE0, 0x00, 0x27, 0x00, 0x00, 0x3E, 0x00, 0x00, 0x3F, 0xE0, 0x00, 0x21, 0xF8, 0x00, 0x01, 0x
E0, 0x00, 0x3E, 0x00, 0x00, 0x20, 0x00},/*"w",2*/
    {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x38, 0x00, 0x00, 0
x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x
00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},/*".",3*/
    {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x08, 0x00, 0x20, 0x38, 0x00, 0x20, 0
xF8, 0x00, 0x23, 0xE8, 0x00, 0x2F, 0x88, 0x00, 0x3E, 0x08, 0x00, 0x38, 0x08, 0x00, 0x20, 0x
18, 0x00, 0x00, 0x70, 0x00, 0x00, 0x00},/*"z",4*/
    {0x00, 0x00, 0x00, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x00, 0x38, 0x38, 0x00, 0x3E, 0
x68, 0x00, 0x27, 0x80, 0x00, 0x03, 0xC8, 0x00, 0x2C, 0xF8, 0x00, 0x38, 0x38, 0x00, 0x20, 0x
18, 0x00, 0x20, 0x08, 0x00, 0x00, 0x00},/*"x",5*/
    {0x00, 0x00, 0x00, 0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x0F, 0xFF, 0xF8, 0x00, 0x01, 0
x88, 0x00, 0x03, 0x00, 0x00, 0x2F, 0xC0, 0x00, 0x38, 0xF8, 0x00, 0x20, 0x38, 0x00, 0x20, 0x
08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00},/*"k",6*/
    {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x03, 0x00, 0x20, 0
x01, 0x00, 0x20, 0x01, 0x00, 0x20, 0x03, 0x06, 0x3F, 0xFE, 0x06, 0x3F, 0xFC, 0x00, 0x00, 0x
00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},/*"j",7*/
    {0x00, 0x20, 0x08, 0x00, 0x3F, 0xF8, 0x00, 0x3F, 0xF8, 0x00, 0x10, 0x08, 0x00, 0x20, 0
x00, 0x00, 0x3F, 0xF8, 0x00, 0x3F, 0xF8, 0x00, 0x10, 0x08, 0x00, 0x20, 0x00, 0x00, 0x3F, 0x
F8, 0x00, 0x3F, 0xF8, 0x00, 0x00, 0x08},/*"m",8*/
    {0x00, 0x00, 0x00, 0x00, 0x07, 0xC0, 0x00, 0x1F, 0xF0, 0x00, 0x18, 0x30, 0x00, 0x20, 0
x08, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x00, 0x3C, 0x08, 0x00, 0x1C, 0x10, 0x00, 0x00, 0x
60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},/*"c",9*/
    {0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x3F, 0xF0, 0x00, 0x7F, 0xF8, 0x00, 0x00, 0
```

```
x18,0x00,0x00,0x08,0x00,0x00,0x08,0x00,0x20,0x10,0x00,0x3F,0xF8,0x00,0x7F,0x
F0,0x00,0x00,0x10,0x00,0x00,0x00},/*"u",10*/
    {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x00,0x00,0x38,0x00,0x00,0
x38,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,0x00,0x00,0x00,0x00,0x00,0x00},/*".",11*/
    {0x00,0x00,0x00,0x00,0x07,0xC0,0x00,0x1F,0xF0,0x00,0x18,0x30,0x00,0x20,0
x08,0x00,0x20,0x08,0x00,0x20,0x08,0x00,0x3C,0x08,0x00,0x1C,0x10,0x00,0x00,0x
60,0x00,0x00,0x00,0x00,0x00,0x00},/*"c",12*/
    {0x00,0x00,0x00,0x00,0x07,0xC0,0x00,0x0F,0xF0,0x00,0x18,0x30,0x00,0x30,0
x08,0x00,0x20,0x08,0x00,0x20,0x08,0x00,0x30,0x08,0x00,0x18,0x30,0x00,0x0F,0x
F0,0x00,0x07,0xC0,0x00,0x00,0x00},/*"o",13*/
    {0x00,0x20,0x08,0x00,0x3F,0xF8,0x00,0x3F,0xF8,0x00,0x10,0x08,0x00,0x20,0
x00,0x00,0x3F,0xF8,0x00,0x3F,0xF8,0x00,0x10,0x08,0x00,0x20,0x00,0x00,0x3F,0x
F8,0x00,0x3F,0xF8,0x00,0x00,0x08}/*"m",14*/
    };

    void LCD_WR_REG(u16 index)
    {
       *(__IO u16 *)(Bank1_LCD_C)=index;
    }

    void LCD_WR_Data(u16 val)
    {
       *(__IO u16 *)(Bank1_LCD_D)=val;
    }

    void LCD_WR_CMD(u16 index,u16 val)
    {
       *(__IO u16 *)(Bank1_LCD_C)=index;
       *(__IO u16 *)(Bank1_LCD_D)=val;
    }


    void ILI9325_Init(void)
    {
        GPIO_ResetBits(GPIOE, GPIO_Pin_1);      //硬件复位
        delay(0xAFFf);
        GPIO_SetBits(GPIOE, GPIO_Pin_1 );
       delay(0xAFFf);

       LCD_WR_CMD(0x0001, 0x0100); // set SS and SM bit
       LCD_WR_CMD(0x0002, 0x0700); // set 1 line inversion
       LCD_WR_CMD(0x0003, 0x1030); // set GRAM write direction and BGR=1.
       LCD_WR_CMD(0x0004, 0x0000); // Resize register
```

```
    LCD_WR_CMD(0x0008, 0x0207); // set the back porch and front porch
    LCD_WR_CMD(0x0009, 0x0000); // set non-display area refresh cycle
ISC[3:0]
    LCD_WR_CMD(0x000A, 0x0000); // FMARK function
    LCD_WR_CMD(0x000C, 0x0000); // RGB interface setting
    LCD_WR_CMD(0x000D, 0x0000); // Frame marker Position
    LCD_WR_CMD(0x000F, 0x0000); // RGB interface polarity

    LCD_WR_CMD(0x0010, 0x0000); // SAP, BT[3:0], AP, DSTB, SLP, STB
    LCD_WR_CMD(0x0011, 0x0007); // DC1[2:0], DC0[2:0], VC[2:0]
    LCD_WR_CMD(0x0012, 0x0000); // VREG1OUT voltage
    LCD_WR_CMD(0x0013, 0x0000); // VDV[4:0] for VCOM amplitude
    LCD_WR_CMD(0x0007, 0x0001);
    delay(12000); // Dis-charge capacitor power voltage
    LCD_WR_CMD(0x0010, 0x1490); // SAP, BT[3:0], AP, DSTB, SLP, STB
    LCD_WR_CMD(0x0011, 0x0227); // DC1[2:0], DC0[2:0], VC[2:0]
    delay(15500); // Delay 50ms
    LCD_WR_CMD(0x0012, 0x001C); // Internal reference voltage= Vci;
    delay(15000); // Delay 50ms
    LCD_WR_CMD(0x0013, 0x1A00); // Set VDV[4:0] for VCOM amplitude
    LCD_WR_CMD(0x0029, 0x0025); // Set VCM[5:0] for VCOMH
    LCD_WR_CMD(0x002B, 0x000C); // Set Frame Rate
    delay(15000); // Delay 50ms
    LCD_WR_CMD(0x0020, 0x0000); // GRAM horizontal Address
    LCD_WR_CMD(0x0021, 0x0000); // GRAM Vertical Address
    // ----------- Adjust the Gamma Curve ----------//
    LCD_WR_CMD(0x0030, 0x0000);
    LCD_WR_CMD(0x0031, 0x0506);
    LCD_WR_CMD(0x0032, 0x0104);
    LCD_WR_CMD(0x0035, 0x0207);
    LCD_WR_CMD(0x0036, 0x000F);
    LCD_WR_CMD(0x0037, 0x0306);
    LCD_WR_CMD(0x0038, 0x0102);
    LCD_WR_CMD(0x0039, 0x0707);
    LCD_WR_CMD(0x003C, 0x0702);
    LCD_WR_CMD(0x003D, 0x1604);
    //------------------ Set GRAM area ---------------//
    LCD_WR_CMD(0x0050, 0x0000); // Horizontal GRAM Start Address
    LCD_WR_CMD(0x0051, 0x00EF); // Horizontal GRAM End Address
    LCD_WR_CMD(0x0052, 0x0000); // Vertical GRAM Start Address
    LCD_WR_CMD(0x0053, 0x013F); // Vertical GRAM Start Address
    LCD_WR_CMD(0x0060, 0xA700); // Gate Scan Line
    LCD_WR_CMD(0x0061, 0x0001); // NDL,VLE, REV
```

```
    LCD_WR_CMD(0x006A, 0x0000); // set scrolling line
    //-------------- Partial Display Control ---------//
    LCD_WR_CMD(0x0080, 0x0000);
    LCD_WR_CMD(0x0081, 0x0000);
    LCD_WR_CMD(0x0082, 0x0000);
    LCD_WR_CMD(0x0083, 0x0000);
    LCD_WR_CMD(0x0084, 0x0000);
    LCD_WR_CMD(0x0085, 0x0000);
    //-------------- Panel Control -------------------//
    LCD_WR_CMD(0x0090, 0x0010);
    LCD_WR_CMD(0x0092, 0x0600);
    LCD_WR_CMD(0x0007, 0x0133); // 262K color and display ON
}

//240*320

void ILI_9325_Draw_Point(u8 x,u16 y,u16 color)
{
    LCD_WR_CMD(0x50,x);  //x 起始
    LCD_WR_CMD(0x51,x);  //x 结束
    LCD_WR_CMD(0x52,y);  //y 起始
    LCD_WR_CMD(0x53,y);  //y 结束

    LCD_WR_CMD(0x20,x);
    LCD_WR_CMD(0x21,y);
    LCD_WR_REG(0x22);
    LCD_WR_Data(color);
}

void ILI_9325_CLEAR(u16 color)
{
    u16 i=0,j=0;

    for(i=0;i<240;i++)
    {
        for(j=0;j<320;j++)
        {
            ILI_9325_Draw_Point(i,j,color);
        }
    }
}

void show_Font(u16 x,u16 y,u8 id,u16 qj,u16 bj)
{
```

```
    u8 k=0,temp=0,t=0,y0=0;

    y0=y;//初值

    for(k=0;k<72;k++)
    {
        temp=font[id][k];
        for(t=0;t<8;t++)
        {
            if(temp&0x80)
                ILI_9325_Draw_Point(x,y,qj);
            else
                ILI_9325_Draw_Point(x,y,bj);

            temp<<=1;
            y++;
            if((y-y0)==24)
            {
                y=y0;
                x++;
                break;
            }
        }
    }
}

void show_Str(u16 x,u16 y,u8 id,u16 qj,u16 bj)
{
    u8 k=0,temp=0,t=0,y0=0;

    y0=y;//初值

    for(k=0;k<36;k++)
    {
        temp=str[id][k];
        for(t=0;t<8;t++)
        {
            if(temp&0x80)
                ILI_9325_Draw_Point(x,y,qj);
            else
                ILI_9325_Draw_Point(x,y,bj);

            temp<<=1;
            y++;
```

```
            if((y-y0)==24)
            {
                y=y0;
                x++;
                break;
            }
        }
    }
}

void TFT_Draw_Rectangle(u16 x1,u16 y1,u16 x2,u16 y2,u16 color)
{
    u16 i=0,j=0;

    for(i=x1;i<x2;i++)
    {
        for(j=y1;j<y2;j++)
        {
            ILI_9325_Draw_Point(i,j,color);
        }
    }
}
```

4.32.14 main.c 文件里的内容是

```
#include "pbdata.h"

void RCC_Configuration(void);
void GPIO_Configuration(void);
void NVIC_Configuration(void);
void USART_Configuration(void);

int fputc(int ch,FILE *f)
{
    USART_SendData(USART1,(u8)ch);
    while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);
    return ch;
}

int main(void)
{
    u8 i=0;
```

```
    RCC_Configuration();  //系统时钟初始化
    GPIO_Configuration();//端口初始化
    USART_Configuration();
    NVIC_Configuration();
    FSMC_Configuration();
    ILI9325_Init();
    delay_ms(1000);
    ILI_9325_CLEAR(WHITE);

    for(i=0;i<4;i++)
    {
        show_Font(30+i*50,50,i,RED,WHITE);
    }

    for(i=0;i<15;i++)
    {
       show_Str(10+i*15,100,i,RED,WHITE);
    }

    while(1)
    {
        delay_ms(200);
        TFT_Draw_Rectangle(10,150,230,160,RED);
        delay_ms(200);
        TFT_Draw_Rectangle(10,150,230,160,BLUE);
        delay_ms(200);
        TFT_Draw_Rectangle(10,150,230,160,GREEN);
    }
}

void RCC_Configuration(void)
{
    SystemInit();//72m
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD,ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE,ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_FSMC,ENABLE);
}

void GPIO_Configuration(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
```

```
    //LED
    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_9;//TX
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA,&GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_10;//RX
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA,&GPIO_InitStructure);

    //FSMC  管脚初始化

    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_13;
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;
    GPIO_Init(GPIOD,&GPIO_InitStructure);
    GPIO_SetBits(GPIOD,GPIO_Pin_13);//打开背光

    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_1;//复位
    GPIO_Init(GPIOE,&GPIO_InitStructure);

    //启用 fsmc 复用功能  复用上拉模式

    GPIO_InitStructure.GPIO_Pin =GPIO_Pin_14 //D0
                                |GPIO_Pin_15 //D1
                                |GPIO_Pin_0    //D2
                                |GPIO_Pin_1    //D3
                                |GPIO_Pin_8    //D13
                                |GPIO_Pin_9    //D14
                                |GPIO_Pin_10;//D15
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AF_PP;
    GPIO_Init(GPIOD,&GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin =GPIO_Pin_7    //D4
                                |GPIO_Pin_8    //D5
                                |GPIO_Pin_9    //D6
                                |GPIO_Pin_10 //D7
                                |GPIO_Pin_11 //D8
                                |GPIO_Pin_12 //D9
                                |GPIO_Pin_13 //D10
                                |GPIO_Pin_14 //D11
                                |GPIO_Pin_15;//D12
    GPIO_Init(GPIOE,&GPIO_InitStructure);
```

```
    GPIO_InitStructure.GPIO_Pin =GPIO_Pin_11 //RS
                               |GPIO_Pin_4    //nOE
                               |GPIO_Pin_5; //nWE
    GPIO_Init(GPIOD,&GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin =GPIO_Pin_7; //NE1
    GPIO_Init(GPIOD,&GPIO_InitStructure);
}

void NVIC_Configuration(void)
{
    NVIC_InitTypeDef NVIC_InitStructure;

    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);

    NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}

void USART_Configuration(void)
{
     USART_InitTypeDef  USART_InitStructure;

    USART_InitStructure.USART_BaudRate=9600;
    USART_InitStructure.USART_WordLength=USART_WordLength_8b;
    USART_InitStructure.USART_StopBits=USART_StopBits_1;
    USART_InitStructure.USART_Parity=USART_Parity_No;
    USART_InitStructure.USART_HardwareFlowControl=USART_HardwareFlowControl_
None;
    USART_InitStructure.USART_Mode=USART_Mode_Rx|USART_Mode_Tx;

    USART_Init(USART1,&USART_InitStructure);
    USART_ITConfig(USART1,USART_IT_RXNE,ENABLE);
    USART_Cmd(USART1,ENABLE);
    USART_ClearFlag(USART1,USART_FLAG_TC);
}
```

4.32.15 程序下载

请根据下图所指向的 7 个重点区域配置。其中（1）号区域根据自己机器的实际情况选择，我的机器虚拟出来的串口号是 COM3。（2）号区域请自己选择程序所在的文件夹。（7）号区域当程序下载完后，进度条会到达最右边，并且提示一切正常。（4、5、6）号区域一定要按照上图显示的设置。当都设置好以后就可以直接点击（3）号区域的开始编程按钮下传程序了。

本节实验的源代码在光盘中：（LY-STM32 光盘资料\1.课程\2,外设篇\18. LCD 彩色液晶屏显示汉字、英文、数字\程序）

4.32.16 实验效果图

4.32.16.1 读写扩展闪存实验效果

在这个实验中我们要把触摸屏和实验板连接好，送电后下载程序。程序下载完成后触摸屏就会显示出我们设计的字符，分别是“众想科技”四个汉和“http://zxkjmcu.taobao.com”网址送到触摸屏显示。

从图 4.32.16.1 试验效果能看出和我们设计的一样，说明我们的程序设计正确，驱动触摸屏正常。

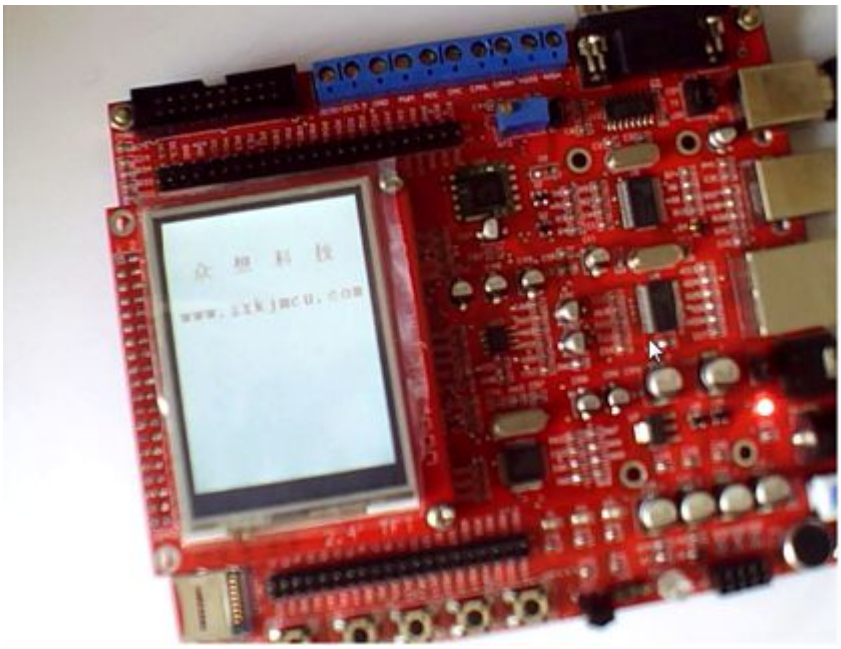


图 4.32.16.1 触摸屏显示实验效果图（一）

4.32.17 汉字代码

众(0) 想(1) 科(2) 技(3)

{0x00, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x02, 0x00, 0x10, 0x0C, 0x00,

0x10, 0x38, 0x00, 0x20, 0xE0, 0x00, 0x5F, 0x80, 0x00, 0x99, 0x00, 0x03, 0x00,

0x80, 0x06, 0x00, 0xC2, 0x1C, 0x00, 0x72, 0x70, 0x00, 0x04, 0x28, 0x00, 0x08,

0x04, 0x00, 0x30, 0x03, 0x03, 0xE0, 0x01, 0xBF, 0x00, 0x00, 0xD7, 0x00, 0x00,

0x60, 0xC0, 0x00, 0x60, 0x30, 0x00, 0x30, 0x18, 0x00, 0x30, 0x0E, 0x00, 0x20,

0x06, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00}, /*"众", 0*/

{0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x02, 0x04, 0x00, 0x02, 0x08, 0x0C, 0x02,

0x30, 0x38, 0x02, 0xE0, 0x00, 0x7F, 0xFE, 0x00, 0x3F, 0xFE, 0x00, 0x02, 0xC0,

0xFC, 0x06, 0x60, 0x04, 0x02, 0x20, 0x04, 0x00, 0x01, 0x04, 0x1F, 0xFE, 0xC4,
0x09, 0x24, 0xE4, 0x09, 0x24, 0x04, 0x09, 0x24, 0x04, 0x09, 0x24, 0x04, 0x09,
0x24, 0x3C, 0x09, 0x24, 0x00, 0x1F, 0xFE, 0x40, 0x1F, 0xFE, 0x30, 0x00, 0x00,
0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /*"想", 1*/

{0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x08, 0x40, 0x40, 0x08, 0x41, 0x80, 0x08,
0x46, 0x00, 0x08, 0x5C, 0x00, 0x1F, 0xFF, 0xFE, 0x1F, 0xFF, 0xFE, 0x10, 0x48,
0x00, 0x30, 0x8C, 0x00, 0x10, 0x86, 0x80, 0x00, 0x00, 0x80, 0x00, 0x20, 0x80,
0x04, 0x10, 0x80, 0x03, 0x1C, 0x80, 0x03, 0x01, 0x00, 0x00, 0x01, 0x00, 0x00,
0x01, 0x00, 0x3F, 0xFF, 0xFE, 0x20, 0x01, 0x00, 0x00, 0x02, 0x00, 0x00, 0x06,
0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00}, /*"科", 2*/

{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x06, 0x00, 0x01, 0x04, 0x04, 0x01,
0x04, 0x06, 0x7F, 0xFF, 0xFE, 0x3F, 0xFF, 0xFC, 0x21, 0x08, 0x00, 0x02, 0x10,
0x02, 0x02, 0x10, 0x02, 0x01, 0x10, 0x02, 0x01, 0x10, 0x04, 0x01, 0x1C, 0x08,
0x01, 0x13, 0x08, 0x01, 0x10, 0xD0, 0x3F, 0xF0, 0x30, 0x21, 0x10, 0x70, 0x01,
0x11, 0xC8, 0x01, 0x17, 0x0C, 0x01, 0x1C, 0x04, 0x02, 0x10, 0x06, 0x02, 0x00,
0x06, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00}, /*"技", 3*/