STM32Cube高效开发教程（基础篇）

第8章 FSMC连接TFT LCD

王维波
中国石油大学（华东）控制科学与工程学院

STM32Cube高效开发教程（基础篇）

作者：王维波，鄢志丹，王钊

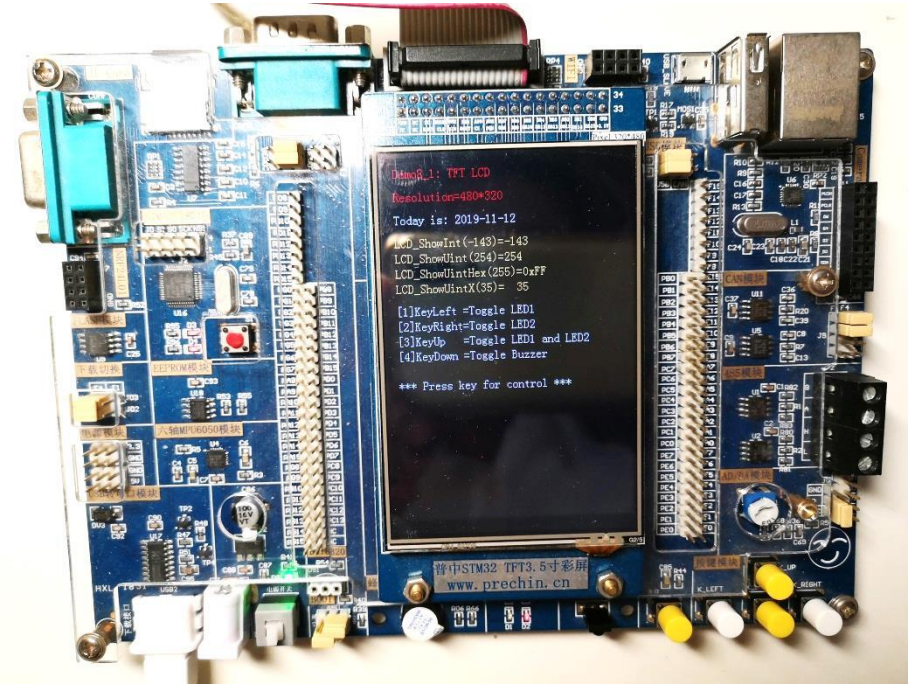
人民邮电出版社

2021年9月出版

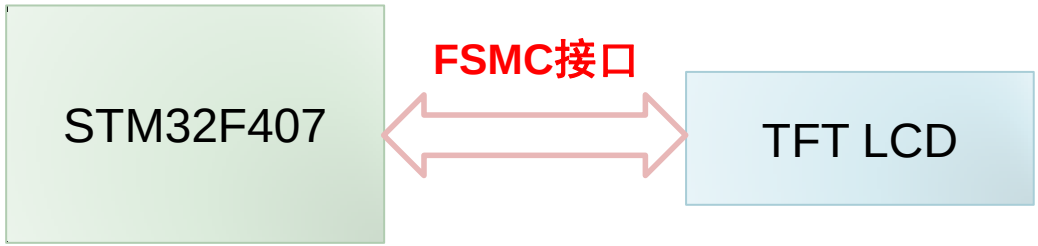
如果有读者需要本书课件的PPT版本用于备课，可以给作者发邮件免费获取，并可加入专门的教学和技术交流QQ群

邮箱：wangwb@upc. edu. cn

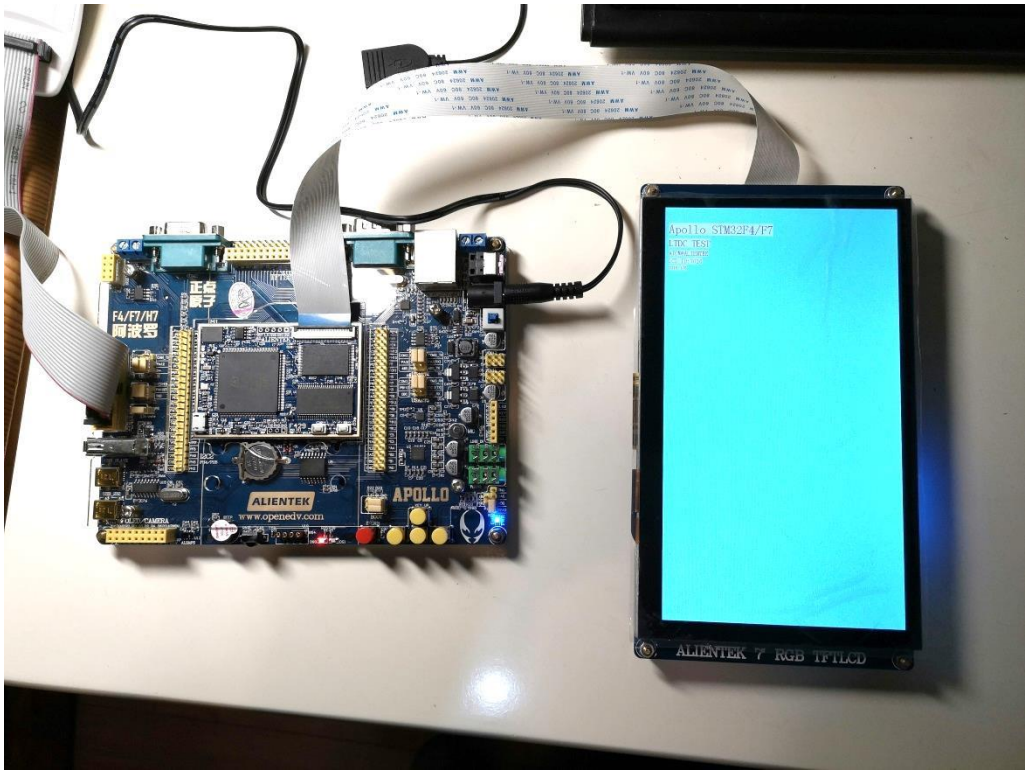
补充资料：各种连接TFT LCD的接口



- 普中STM32F407开发板
- 3.5寸 TFT LCD
- FSMC接口

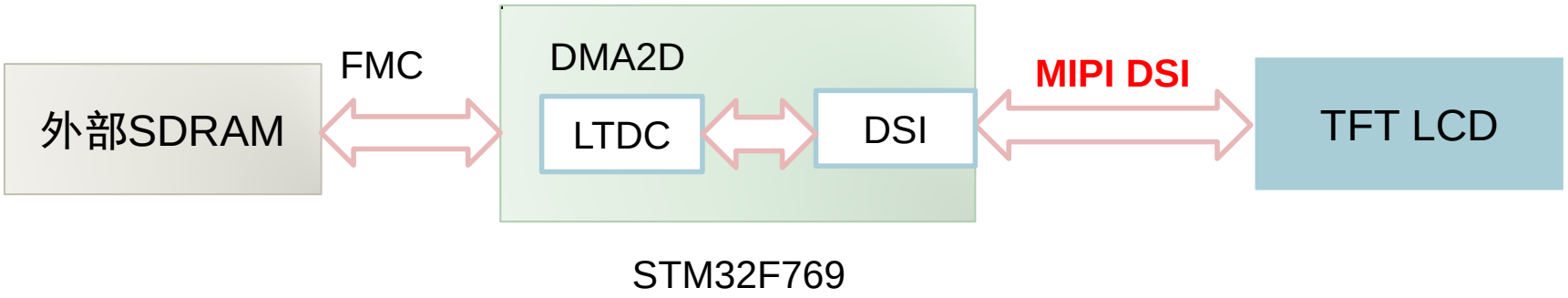


MCU屏
（带LCD控制器）



- 正点原子STM32F429开发板
- 7寸 TFT LCD
- LTDC接口

- ST官方的STM32F769I DICO开发板
- 4.3寸 TFT LCD
- MIPI DSI接口



第8章 FSMC连接TFT LCD

8.1 FSMC连接TFT LCD的原理

8.2 FSMC连接LCD的电路以及接口初始化

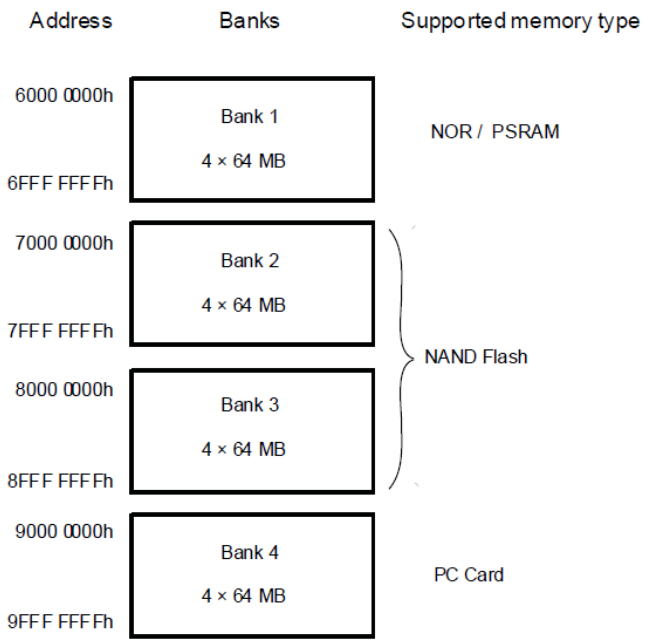
8.3 使用LCD驱动程序

8.1.1 FSMC接口

FSMC（Flexible static memory controller，灵活的静态存储控制器）接口。它能够连接NOR/PSRAM、NAND Flash、PC Card、TFT LCD等。

FSMC连接的所有外部存储器共享地址、数据和控制信号，但有各自的片选信号。

FSMC外部存储器被划分为4个固定大小的存储区域（memory bank），每个区域大小为256MB



Bank 1被分为4个子区域，每个子区域容量64MB，有专用的片选信号。

- Bank 1- NOR/PSRAM 1，片选信号NE1
- Bank 1- NOR/PSRAM 2，片选信号NE2
- Bank 1- NOR/PSRAM 3，片选信号NE3（开发板上用于连接外部SRAM）
- Bank 1- NOR/PSRAM 4，片选信号NE4（开发板上用于连接TFT LCD）

- Bank 2和Bank 3用于访问NAND Flash存储器，每个区域连接一个设备
- Bank 4用于连接PC卡设备

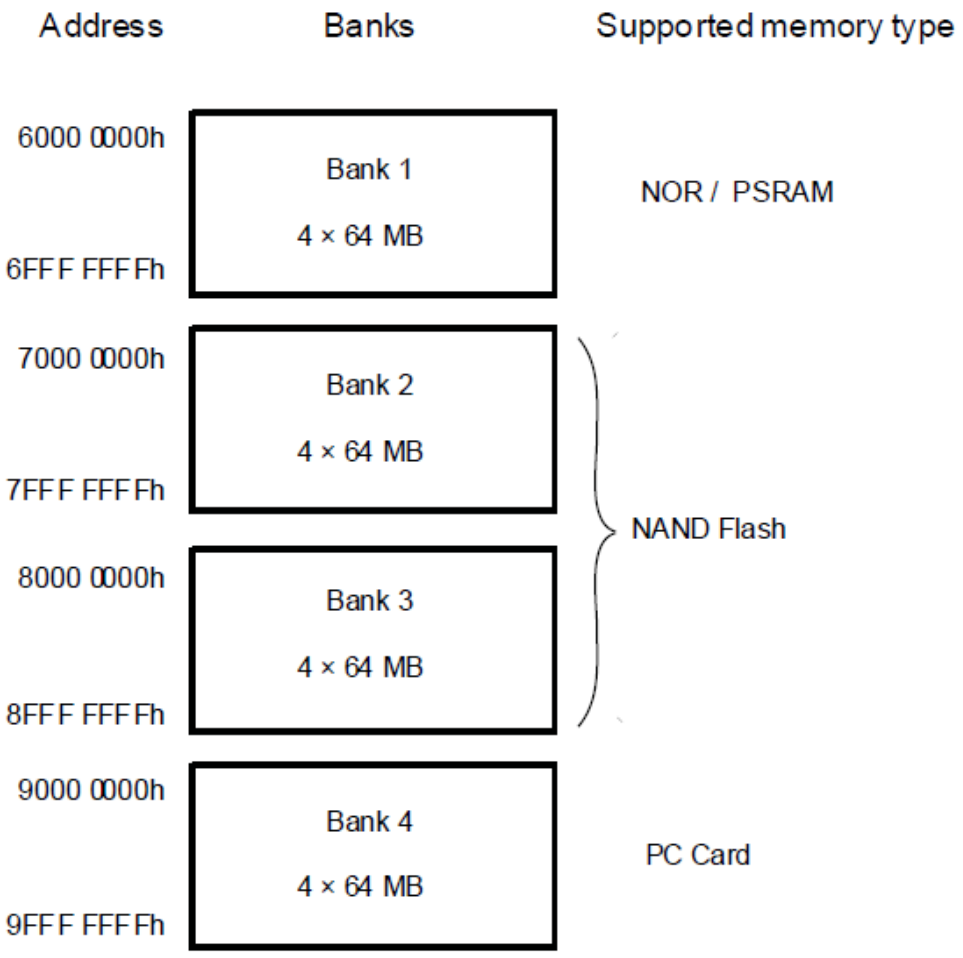


表8-1 非复用PSRAM/SRAM接口信号

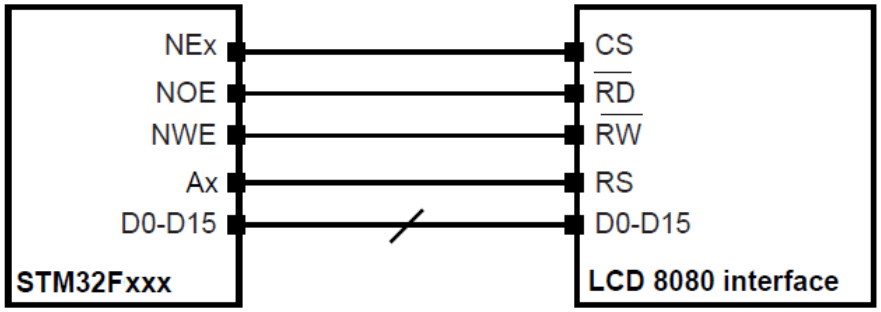
FSMC信号	I/O	功能
CLK	O	时钟（仅用于PSRAM同步突发）
A[25:0]	O	26位地址总线，寻址空间64M，也就是在一个子区域的空间
D[15:0]	I/O	16位数据双向总线
NE[x]	O	片选信号，x=1..4，即NE1, NE2, NE3, NE4，用于选择子区域
NOE	O	输出使能（output enable），低有效
NWE	O	写入使能（write enable），低有效
NL(=NADV)	O	仅用于PSRAM输入的地址有效信号（存储器信号名称：NADV）
NWAIT	I	PSRAM发送给FSMC的等待输入信号
NBL[1]	O	高字节使能（存储器信号名称：NUB）
NBL[0]	O	低字节使能（存储器信号名称：NLB）

8.1.2 TFT LCD接口

TFT LCD通常使用标准的8080并口，有16位数据线，还有几根控制线。除RST信号外，其他信号都可以由FSMC接口提供

LCD信号	I/O	功能
RST	I	复位信号，低电平复位
CS	I	片选信号，低有效
RS	I	LCD寄存器选择（register select），RS=0 控制寄存器；RS=1数据寄存器
RD	I	读操作，低有效
WR	I	写操作，低有效
D0-D15	I/O	16位数据线

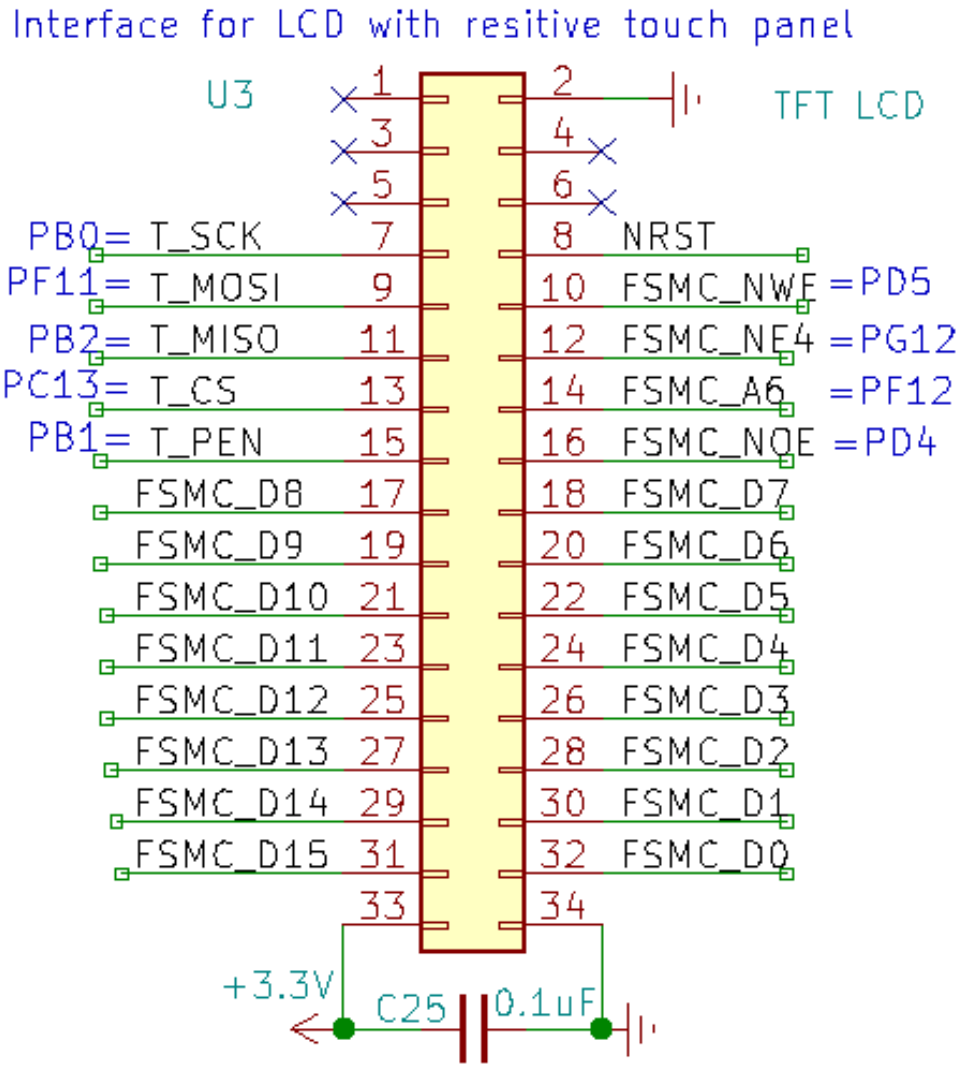
8.1.3 FSMC与TFT LCD的连接



LCD信号	功能	连接的FSMC信号
RST	复位信号，低电平复位	由MCU控制或连接MCU的RST信号
CS	片选信号，低有效	NEx，使用NE1，NE2，NE3，NE4中的某个信号
RS	LCD寄存器选择	Ax，使用A[25..0]中的某根地址线
RD	读操作，低有效	NOE，FSMC的输出使能就是读取LCD
WR	写操作，低有效	NWE，FSMC的写入使能就是写入LCD
D0-D15	16位数据线	D[15:0]数据线

开发板上TFT LCD模块接口线定义

触摸面板接口



FSMC有多种时序模型用于NOR/Flash/PSRAM/SRAM的访问，对LCD的访问使用模式A比较方便，因为模式A支持独立的读写时序控制。

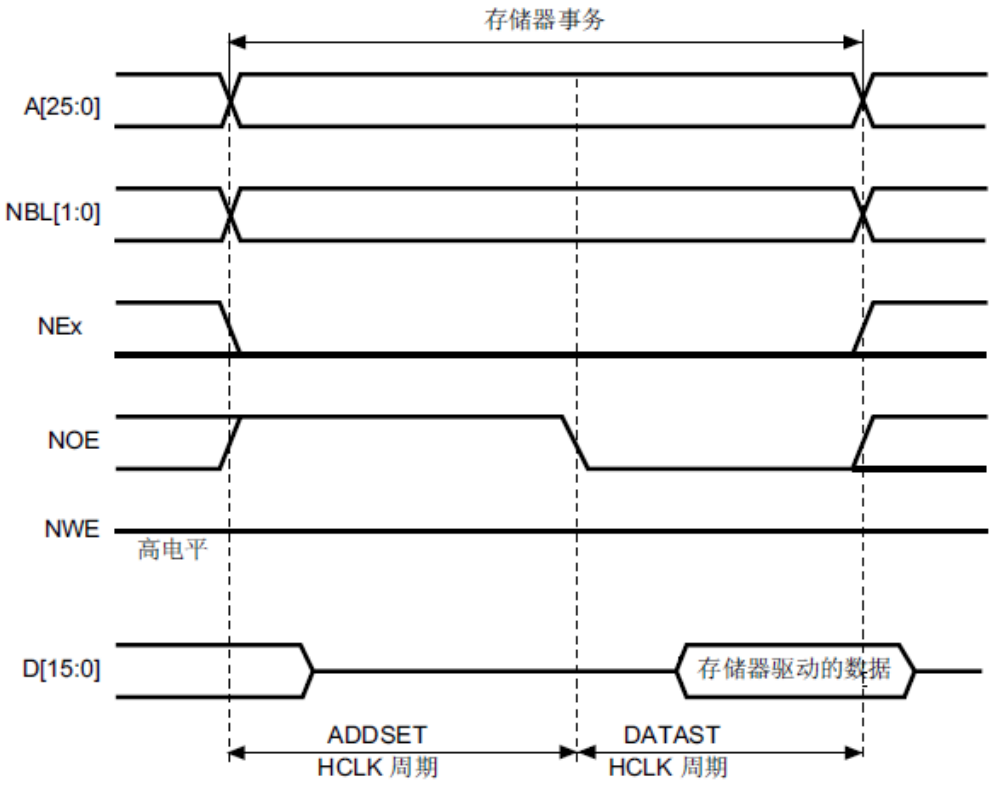


图8-3 SRAM/PSRAM模式A读取访问时序

时序中需要设置两个参数：地址建立时间ADDSET和数据建立时间DATAST，它们都用HCLK的时钟周期个数表示。

第8章 FSMC连接TFT LCD

8.1 FSMC连接TFT LCD的原理

8.2 FSMC连接LCD的电路以及接口初始化

8.3 使用LCD驱动程序

8.2.1 电路连接

3.5寸TFT LCD，驱动芯片是ILI9481/9486，分辨率480*320

FSMC使用Bank 1的子区4访问LCD，使用FSMC_NE4连接LCD的片选信号，FSMC_A6作为LCD的RS信号。

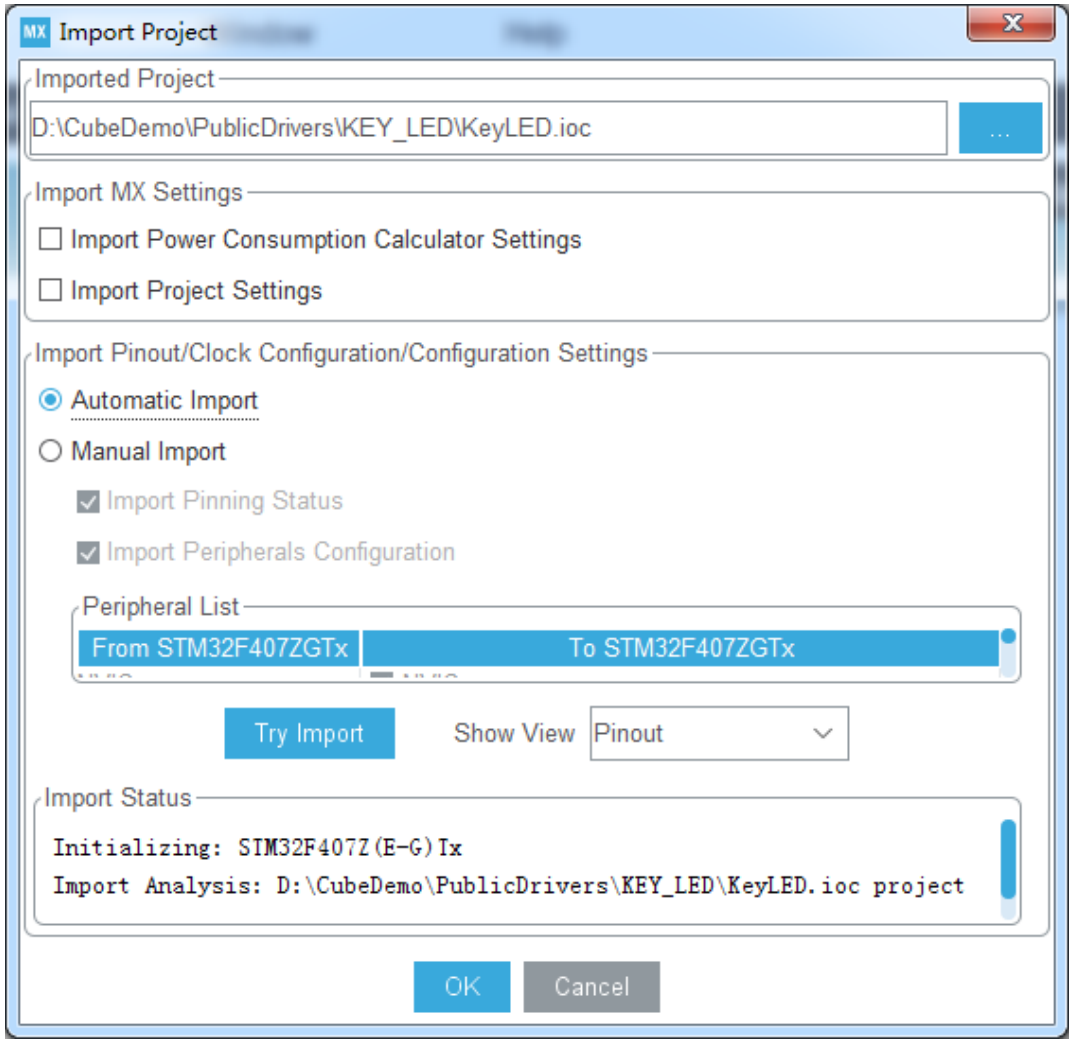
LCD信号	功能	连接的FSMC信号
RST	复位信号，低电平复位	与MCU共用NRST信号，即按键复位
CS	片选信号，低有效	FSMC_NE4，使用Bank1子区4
RS	LCD寄存器选择	FSMC_A6
RD	读操作，低有效	FSMC_NOE
WR	写操作，低有效	FSMC_NWE
D0-D15	16位数据线	FSMC_D[0..15]

8.2.2 CubeMX项目设置

1. 导入KEY_LED项目

使用4个按键和2个LED，需要导入Key_LED项目

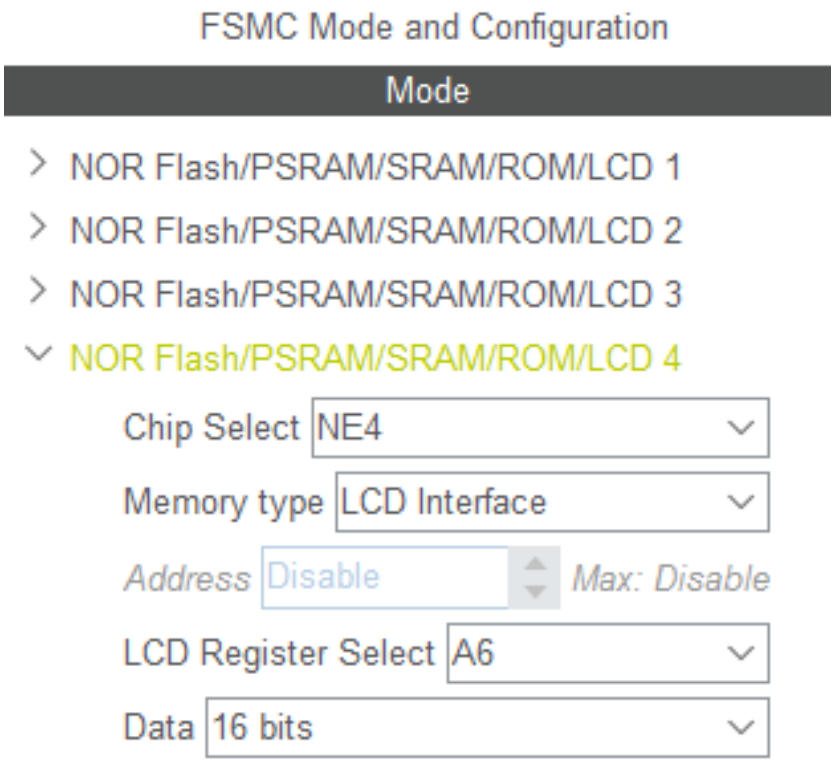
新建项目，未做任何修改的时候导入



2. FSMC设置

使用Bank1的子区4连接LCD

- Chip Select设置为NE4，使用FSMC_NE4作为LCD的片选信号
- Memory type设置为LCD Interface
- LCD Register Select设置为A6，使用FSMC_A6作为LCD的寄存器选择信号
- Data设置为16 bits，也就是使用FSMC_D[15..0]作为访问LCD的16位数据线

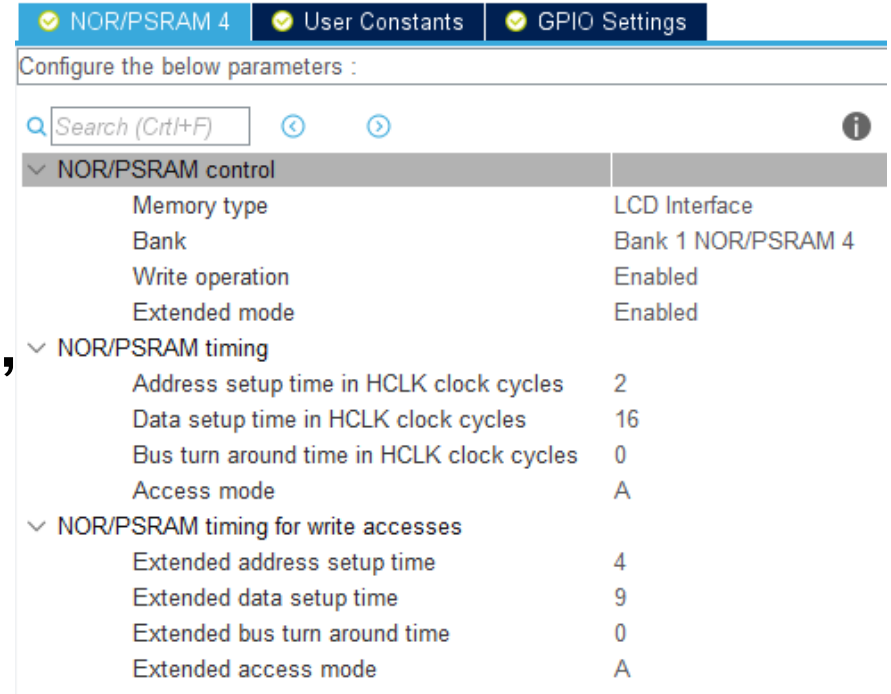


模式设置

3组参数设置

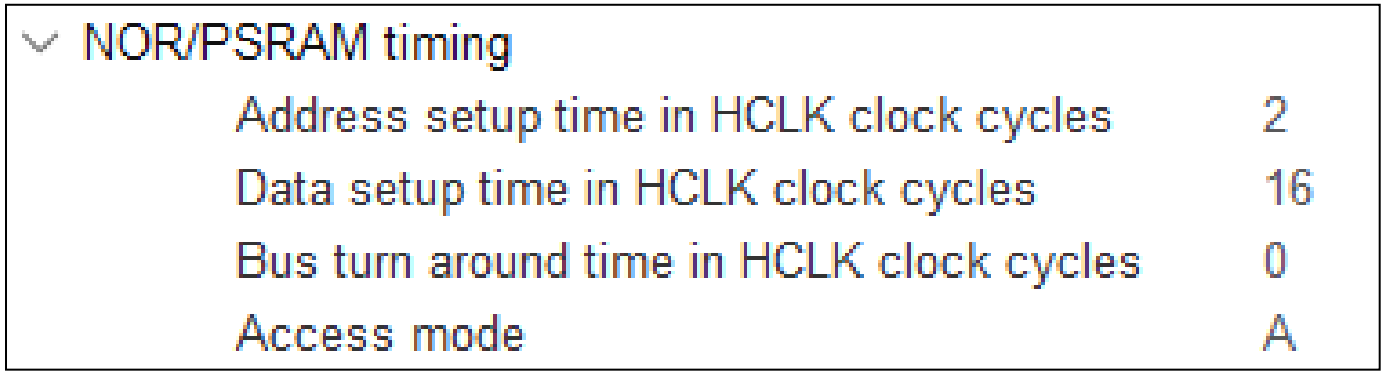
（1）NOR/PSRAM control组参数

- Memory type，选择LCD
- Write operation设置为Enabled
- Extended mode设置为Enabled，使用扩展模式才会出现下面的第3组参数，对读取操作和写入操作分别定义时序参数。



参数设置

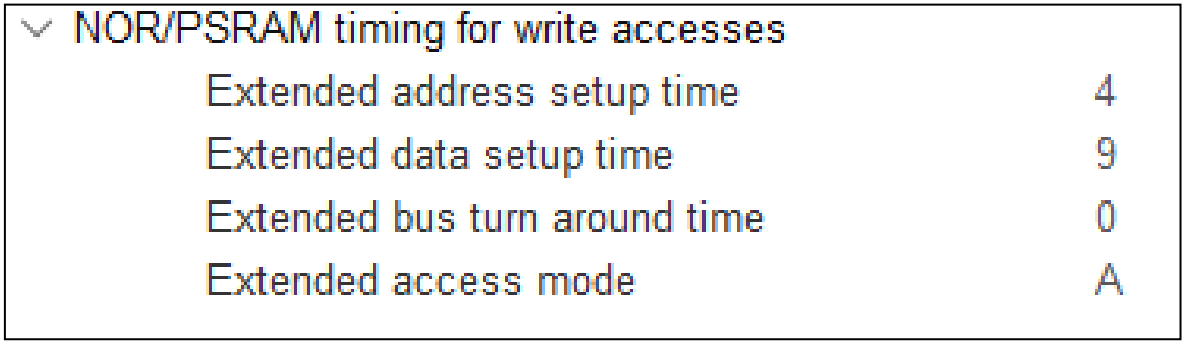
（2）NOR/PSRAM timing组参数，设置读取操作时序的参数



- Address setup time in HCLK clock cycles，即地址建立时间参数ADDSET
- Data setup time in HCLK clock cycles，即数据建立时间参数DATAST
- Bus turn around time in HCLK clock cycles，总线翻转时间
- Access mode，访问模式

（3）NOR/PSRAM timing for writing access组参数，设置

写入操作时序的参数



- Extended address setup time，即地址建立时间参数ADDSET
- Extended data setup，即数据建立时间参数DATAST
- Extended bus turn around ，总线翻转时间
- Extended access mode，访问模式

8.2.3 初始化代码分析

1. 主程序

```
#include "main.h"
#include "gpio.h"
#include "fsmc.h"
void SystemClock_Config(void);

int main(void)
{
  HAL_Init();  //复位所有外设，初始化Flash接口和Systick
  SystemClock_Config();                    //配置系统时钟

  /*  初始化所有已配置外设  */
  MX_GPIO_Init();                  //GPIO初始化
  MX_FSMC_Init();                  //FSMC初始化
  while (1)
  {
  }
}
```

2. 函数MX_GPIO_Init()

这个函数只是使能了一些端口时钟，FSMC接口用到的引脚的初始化设置在文件fsmc.c中完成。

```
void MX_GPIO_Init(void)
{
  /* GPIO 端口时钟使能 */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOE_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOG_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
}
```

3. FSMC接口初始化

MX_FSMC_Init()是进行FSMC初始化，它又会调用函数HAL_SRAM_MspInit()对FSMC引脚进行GPIO初始化。

代码较长，看源程序

（1）定义了表示SRAM存储器Bank1子区4的变量hsram4

```
SRAM_HandleTypeDef hsram4;  //表示Bank1子区4的变量
```

```
typedef struct
{
  FMC_NORSRAM_TypeDef           *Instance;          // 寄存器基址
  FMC_NORSRAM_EXTENDED_TypeDef  *Extended;  // 扩展模式寄存器基址
  FMC_NORSRAM_InitTypeDef        Init;              // SRAM 设备控制配置参数
  HAL_LockTypeDef                Lock;              //SRAM 锁定对象
  __IO HAL_SRAM_StateTypeDef     State;             // SRAM 设备访问状态
  DMA_HandleTypeDef              *hdma;             // DMA 指针
} SRAM_HandleTypeDef;
```

其中，Instance赋值寄存器基址，指向 Bank1

```
hsram4.Instance = FSMC_NORSRAM_DEVICE;
```

成员变量hsram4.Init是SRAM设备控制具体参数，是一个FMC_NORSRAM_InitTypeDef结构体类型，其中重要的几个参数的设置语句是：

```
hsram4.Init.NSBank = FSMC_NORSRAM_BANK4;    //指向Bank1的子区4
hsram4.Init.MemoryType = FSMC_MEMORY_TYPE_SRAM;   //存储器类型SRAM
hsram4.Init.MemoryDataWidth = FSMC_NORSRAM_MEM_BUS_WIDTH_16;
      //16位数据总线
```

它们用于设置hsram4操作的是Bank1的子区4，存储器类型为SRAM（也就是用于控制LCD的类型），16位数据总线。

（2）时序参数定义

函数MX_FSMC_Init()中定义了两个时序参数定义变量，分别用于定义读取时序参数和写入时序参数。

```
FSMC_NORSRAM_TimingTypeDef  Timing;          //读取时序
FSMC_NORSRAM_TimingTypeDef  ExtTiming;       //写入时序
```

时序的主要参数包括地址建立时间、数据保持时间等（与图10-7中的设置对应），访问模式为模式A。

（3）初始化函数HAL_SRAM_Init()

函数HAL_SRAM_Init()用于对FSMC连接SRAM的接口时序进行初始化设置

在函数MX_FSMC_Init()里，完成了hsram4、Timing和ExtTiming这3个变量的属性赋值后，执行下面的函数调用进行了FSMC连接LCD的接口时序的初始化设置。

```
HAL_SRAM_Init(&hsram4, &Timing, &ExtTiming)
```

HAL_SRAM_Init()里要调用MSP函数HAL_SRAM_MspInit()进行MCU相关的初始化，这个函数在fsmc.c文件里重新实现了，用于对FSMC接口引脚进行GPIO初始化设置。

第8章 FSMC连接TFT LCD

8.1 FSMC连接TFT LCD的原理

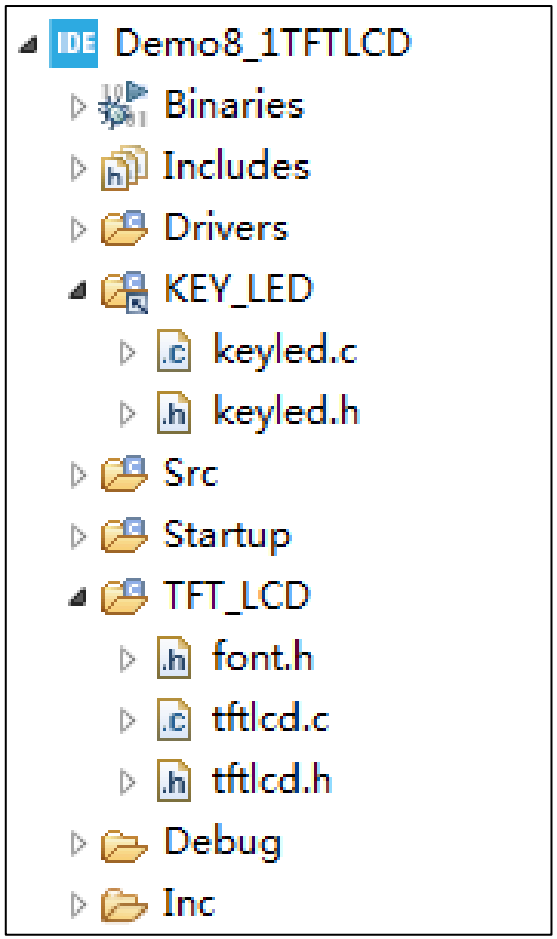
8.2 FSMC连接LCD的电路以及接口初始化

8.3 使用LCD驱动程序

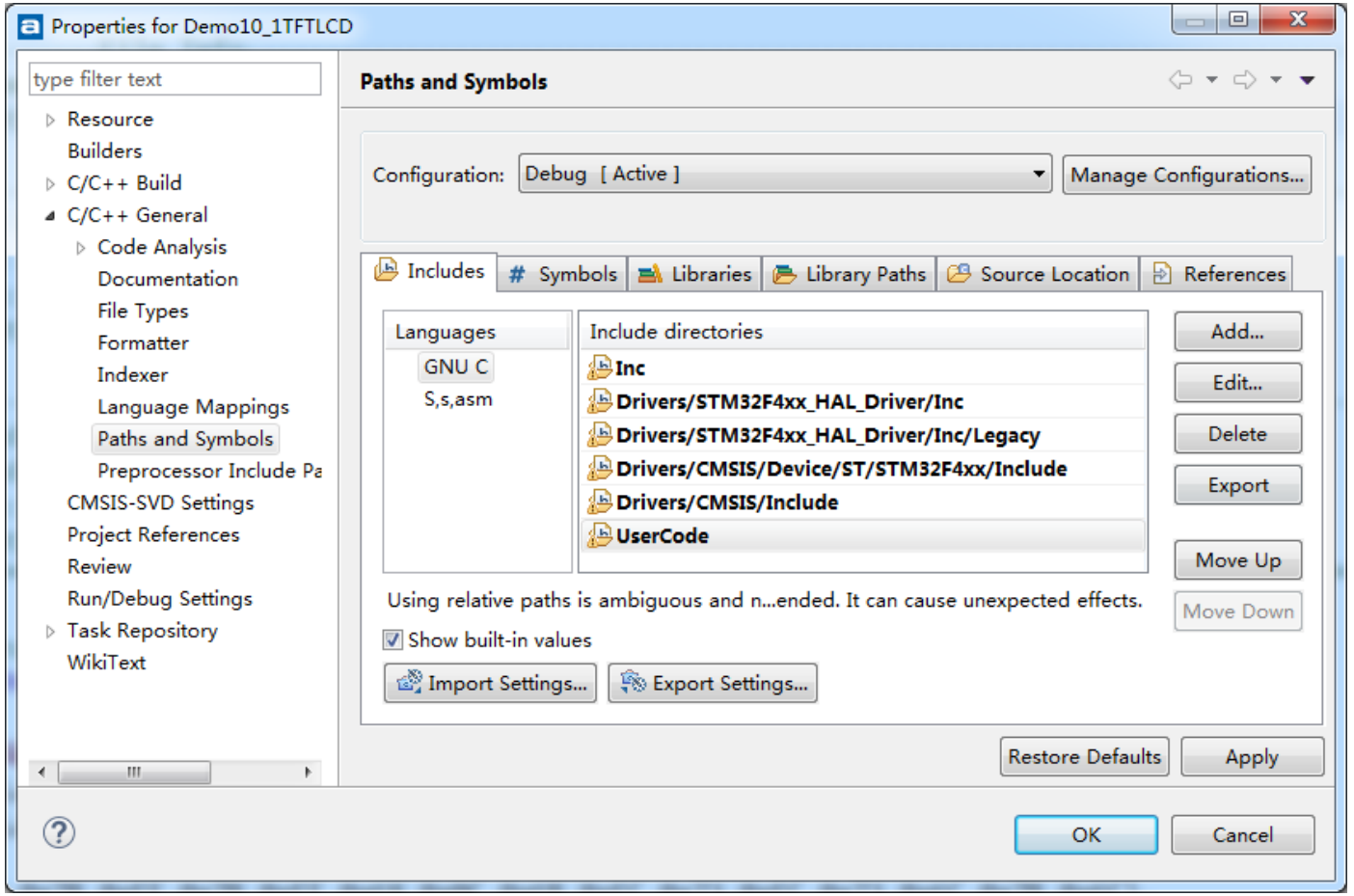
8.3.1 LCD驱动程序文件组成

在项目根目录下新建一个文件夹\UserDrivers\TFT_LCD，下面有3个文件

- font.h文件，这个文件里定义了三种大小的ASCII字符集的点阵数据，分别是12*12、16*16、24*24
- tftlcd.h和tftlcd.c，是LCD驱动程序文件，包括LCD底层的读写函数，以及显示字符串、显示数字、显示汉字的函数，还有清屏、画线、画矩形等绘图函数



还需要将LCD驱动程序文件所在目录添加到头文件搜索路径和源程序搜索路径里



8.3.2 LCD驱动程序的组成和功能

文件tftlcd.h和tftlcd.c是LCD驱动程序函数的主要文件，文件tftlcd.h定义了驱动程序相关的结构体、变量和函数。

```
/* 文件:tftlcd.h,  TFT LCD驱动程序头文件  */
#include "stm32f4xx_hal.h"

#define   TFTLCD_HX8352C                    //3.6寸屏的驱动芯片,分辨率400*240
//#define  TFTLCD_HX8357D
//#define  TFTLCD_ILI9486                    //3.5寸屏的驱动芯片,

#define   USE_HZ_LIB      //是否使用汉字库HzLib.c,如果不使用就注释掉这行
#define   TFTLCD_DIR   0  //0：竖屏  1：横屏  默认竖屏

//TFTLCD地址结构体
typedef struct
{
      uint16_t LCD_CMD;
      uint16_t LCD_DATA;
}TFTLCD_TypeDef;
```

（续1）

```
//TFTLCD重要参数
typedef struct
{
    uint16_t  width;                    //LCD 宽度
    uint16_t  height;                   //LCD 高度
    uint16_t  id;                       //LCD ID
    uint8_t   dir;                      //LCD 方向
}TFTLCD_ParaDef;

//一些全局变量
extern  TFTLCD_ParaDef lcdPara;   //LCD重要参数
extern  uint16_t LCD_W;             //LCD 宽度,会在LCD_Display_Dir()函数里初始化
extern  uint16_t LCD_H;             //LCD 高度
extern  uint8_t  LCD_FS;            //fontsize, 一个字符宽度为半个字体大小
extern    uint16_tLCD_SP10;         //1.0倍行距
extern    uint16_tLCD_SP15;         //1.5倍行距
extern    uint16_tLCD_SP20;         //2.0倍行距
extern  uint16_t LCD_CurX;          //当前位置X
extern  uint16_t LCD_CurY;          //当前位置Y
extern  uint16_t  LcdFRONT_COLOR;         //LCD前景颜色 默认红色
extern  uint16_t  LcdBACK_COLOR;          //LCD背景颜色.默认为白色
```

（续2）

```
/* LCD基本读写指令  */
void LCD_WriteCmd(uint16_t cmd);                          //写指令
void LCD_WriteData(uint16_t data);                        //写数据
void LCD_WriteCmdData(uint16_t cmd,uint16_t data);        //写指令数据
void LCD_WriteData_Color(uint16_t color); //写颜色数据

/*  LCD软件初始化  */
void TFTLCD_Init(void);    //LCD 初始化

/* ------  LCD显示基本功能函数定义       ------------------------------------*/
void LCD_Set_Window(uint16_t sx,uint16_t sy,uint16_t width,uint16_t height);
void LCD_Clear(uint16_t Color);                  //清屏
void LCD_DrawPoint(uint16_t x,uint16_t y);              //画点
void LCD_DrawFRONT_COLOR(uint16_t x,uint16_t y,uint16_t color);  //快速画点
void LCD_DrawLine(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2); //画线
void LCD_DrawRectangle(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2); //画矩形
void LCD_Draw_Circle(uint16_t x0,uint16_t y0,uint8_t r);  //画一个指定大小的圆
```

（续3）

```
//=========字符和数字显示=========
void LCD_setFontSize(uint8_t       fontSize);//设置字体大小，只能是12，16，或24
//在指定位置显示一个字符
void LCD_ShowChar(uint16_t x,uint16_t y,uint8_t charCode, uint8_t mode);

void LCD_ShowInt(uint16_t x,uint16_t y,int32_t num);  //显示整数
void LCD_ShowUint(uint16_t x,uint16_t y,uint32_t num);  //显示无符号整数
void LCD_ShowUintX(uint16_t x,uint16_t y,uint32_t num, uint8_t digiCount);
//固定数字位数显示，前端补0

void LCD_ShowString(uint16_t x,uint16_t y,uint8_t *p);  //显示字符串

//显示font.h里定义的汉字，字模大小32*29，需自己做字模
void LCD_ShowFontHZ(uint16_t x, uint16_t y, uint8_t *cn);

//使用HzLib.c里定义的GB2312汉字库 显示任意汉字，16*16点阵
#ifdef    USE_HZ_LIB
    void LCD_ShowHZ16(uint16_t x, uint16_t y, uint8_t *str);
#endif
```

（1）LCD驱动芯片宏定义

```
#define TFTLCD_ILI9481          //3.5寸电阻屏,分辨率480*320
```

驱动芯片的宏定义主要用于条件编译

（2）一些全局变量

```
extern  uint16_t LCD_W;         //LCD 宽度
extern  uint16_t LCD_H;         //LCD 高度
extern  uint8_t  LCD_FS;        //字体大小, 一个字符宽度为半个字体大小

extern  uint16_t LCD_SP10;      //1.0倍行距
extern  uint16_t LCD_SP15;      //1.5倍行距
extern  uint16_t LCD_SP20;      //2.0倍行距

extern  uint16_t LCD_CurX;      //当前位置X
extern  uint16_t LCD_CurY;      //当前位置Y
```

（3）LCD地址结构体

```
typedef struct
{
	uint16_t LCD_CMD;
	uint16_t LCD_DATA;
}TFTLCD_TypeDef;
#define TFTLCD_BASE		((uint32_t)(0x6C000000 | 0x0000007E))
#define TFTLCD			((TFTLCD_TypeDef *) TFTLCD_BASE)
```

TFTLCD->LCD_CMD 就是指令寄存器

TFTLCD->LCD_DATA 就是数据寄存器

LCD的驱动程序就是按照驱动芯片的指令集向指令寄存器和数据寄存器写入规定的数据实现的。

（4）基本读写函数实现代码

```
//写指令寄存器, cmd:寄存器值
void LCD_WriteCmd(uint16_t cmd)
{
    TFTLCD->LCD_CMD=cmd<<8;
}

//写数据寄存器，data:要写入的值
void LCD_WriteData(uint16_t data)
{
    TFTLCD->LCD_DATA=data<<8;
}
 void LCD_WriteCmdData(uint16_t cmd,uint16_t data)
{
    LCD_WriteCmd(cmd);
    LCD_WriteData(data);
}
void LCD_WriteData_Color(uint16_t color)
{
    TFTLCD->LCD_DATA=color & 0xFF00;
    TFTLCD->LCD_DATA=color<<8;
}
```

（5）LCD的软件初始化函数TFTLCD_Init()

它就是通过LCD_WriteCmd()、LCD_WriteCmdData等基本函数向LCD写入一系列的指令或数据来完成初始化。其中用到的LCD操作指令可查阅LCD驱动芯片的数据手册。

完成FSMC接口初始化之后必须调用此函数进行LCD软件初始化

看示例完整源程序

（6）其他操作函数

文件tftlcd.h和tftlcd.c里定义和实现的其他的一些函数都是调用前面所述的几个基本读写函数实现的，例如：

- LCD_DrawFRONT_COLOR()，使用指定颜色画一个像素点
- LCD_ShowChar()，在指定位置显示一个ASCII码字符
- LCD_ShowString()，在指定位置显示一个ASCII码字符串
- LCD_ShowHZ16()函数，显示16*16点阵任意汉字
- LCD_ShowInt()函数，用于显示一个整数
- LCD_DrawPoint()函数，使用前景色画一个像素点
- LCD_DrawLine()函数，用于画一条线
- LCD_Clear()函数，使用指定的颜色清除屏幕

8.3.3 LCD驱动程序的使用

在main()函数里调用LCD驱动程序，进行一些显示

```
/* 文件：main.c   ------------------------------------------------------------*/
#include "main.h"
#include "gpio.h"
#include "fsmc.h"
/* USER CODE BEGIN Includes */
#include "tftlcd.h"
#include "keyled.h"
/* USER CODE END Includes */

int main(void)
{
    HAL_Init();
    SystemClock_Config();
    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_FSMC_Init();
```

看完整源程序

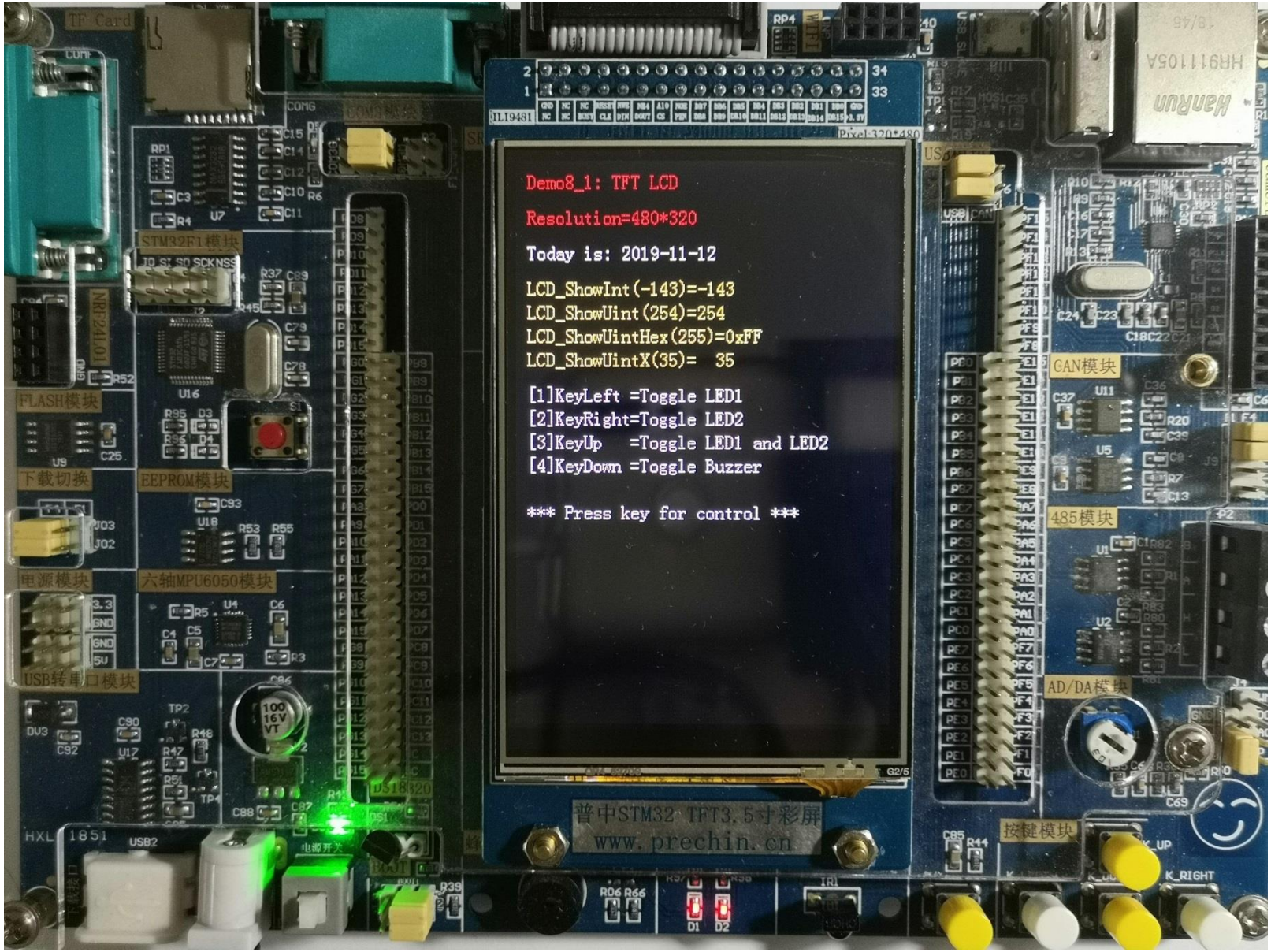
```
/* USER CODE BEGIN 2 */
TFTLCD_Init();    //LCD软件初始化
LcdFRONT_COLOR=lcdColor_RED;   //设置前景色
LCD_ShowStr(10,10, (uint8_t*)"Demo8_1: TFT LCD");
char str[50];
sprintf(str,"Resolution=%d*%d",LCD_H, LCD_W);
LCD_ShowStr(10,LCD_CurY+LCD_SP15, (uint8_t*)str);

LcdFRONT_COLOR=lcdColor_WHITE;
LCD_ShowStr(10,LCD_CurY+LCD_SP15, (uint8_t*)"Today is: ");
LCD_ShowStr(LCD_CurX,LCD_CurY, (uint8_t*)"2019-11-12");

LcdFRONT_COLOR=lcdColor_YELLOW;    //设置前景色
LCD_ShowStr(10,LCD_CurY+LCD_SP15, (uint8_t*)"LCD_ShowInt(-143)=");
LCD_ShowInt(LCD_CurX,LCD_CurY, -143);

LCD_ShowStr(10,LCD_CurY+LCD_SP10, (uint8_t*)"LCD_ShowUint(254)=");
LCD_ShowUint(LCD_CurX,LCD_CurY, 254);
```

看完整源程序

```
 LCD_ShowStr(10,LCD_CurY+LCD_SP10, (uint8_t*)"LCD_ShowUintHex(255)=");
 LCD_ShowUintHex(LCD_CurX,LCD_CurY, 255, 1);

 LCD_ShowStr(10,LCD_CurY+LCD_SP10, (uint8_t*)"LCD_ShowUintX(35)=");
 LCD_ShowUintX(LCD_CurX,LCD_CurY, 35,4);

 LcdFRONT_COLOR=lcdColor_WHITE;     //设置前景色
 LCD_SetFontSize(lcdFont_Size16);  //设置字体大小
 LCD_ShowStr(10,LCD_CurY+LCD_SP15, (uint8_t*)"[1]KeyLeft =Toggle LED1");
 LCD_ShowStr(10,LCD_CurY+LCD_SP10, (uint8_t*)"[2]KeyRight=Toggle LED2");
 LCD_ShowStr(10,LCD_CurY+LCD_SP10, (uint8_t*)"[3]KeyUp   =Toggle LED1
and LED2");
 LCD_ShowStr(10,LCD_CurY+LCD_SP10, (uint8_t*)"[4]KeyDown =Toggle
Buzzer");

 LCD_ShowStr(10,LCD_CurY+LCD_SP20, (uint8_t*)"*** Press key for control ***");
 /* USER CODE END 2 */
```

```
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    KEYS  curKey=ScanPressedKey(KEY_WAIT_ALWAYS);
    switch(curKey)
    {
    case KEY_LEFT:
      LED1_Toggle();
      break;
    case KEY_RIGHT:
      LED2_Toggle();
      break;
    case KEY_UP:
      LED1_Toggle();
      LED2_Toggle();
      break;
    case KEY_DOWN:
      Buzzer_Toggle();
    }
    HAL_Delay(300); //按键弹起阶段的延时，消除按键抖动影响
    /* USER CODE END WHILE */
  }
}
```

示例运行效果, 3.5寸LCD， 分辨率480*320



示例运行效果, 4.3寸LCD， 分辨率800*480

练习任务

1. 编写本章的示例程序，学会使用LCD的驱动程序。本书后面的各种基本都需要使用LCD显示信息，所以LCD驱动程序的使用是基础。
2. 若手头的开发板与书中的不一样，且只提供了基于标准库的LCD驱动程序，根据书中所讲的方法，将LCD的标准库驱动程序改写为CubeMX配置+HAL库驱动程序。