有梦想，
哪里都是你的舞台。
芯嵌stm32开发板

STM32 入门系列教程

点亮 LCD 液晶屏

Revision 1.01

(2013-06-08)

原想把本期《点亮 LCD 液晶屏》教程放在《GPIO 编程》之后，以提高大家的兴趣，但考虑到可能网友学习 STM32，是想更多地了解 STM32 内部工作机制，因此在之前的教程，我们先介绍了串口、外部中断、定时器等最基本的外设模块，有了这些基础，相信您再来学习 LCD 液晶，已经很轻松了。

我们使用的是芯嵌 STM32 配套的 2.8 寸 TFT 液晶触摸屏（备注：芯嵌 stm32 所用的 2.8 寸屏，对角线已经接近 3.0 寸的屏），内部驱动 IC 为 ILI9341。我们操作 LCD，实际上就是在操作 ILI9341。有关该芯片的资料，请参考光盘中的目录《芯嵌 STM32 配套芯片手册》：

注意，这里只讲述如何去点亮 LCD 液晶屏，如果您看完本期教程，能理解 LCD 驱动过程，则笔者心满意足。

要驱动 LCD，分两个部分讲解：

1、CPU 内部模块支持的 LCD 接口（这里使用 FSMC 模块）

2、LCD 控制电路

一、STM32 的 FSMC 原理

如果是单片机，相信大家再熟悉不过了，直接拿 P0 或者 P1 口用作 LCD 数据总线，再另外拿出几个 IO 口用作控制信号线 —— 一个 LCD 控制电路完成了。STM32 相对于单片机，有啥过人之处呢？

对于 STM32 系列的 CPU 来说，有两种方法给 LCD 总线赋值。第一个方法，就是给对应的 GPIOx_ODR 寄存器赋值 —— 这与单片机一样，单片机也是给 P0-P3 寄存器赋值，使得信号能从对应的 IO 端口输出。而 STM32 的另一种方法就是使用 FSMC。FSMC 全称“静态存储器控制器”。使用 FSMC 控制器后，我们可以把 FSMC 提供的 FSMC_A[25:0]作为地址线，而把 FSMC 提供的 FSMC_D[15:0]作为数据总线。

1、 FSMC 包括哪几个部分？

FSMC 包含以下四个模块：

（1）AHB 接口（包含 FSMC 配置寄存器）

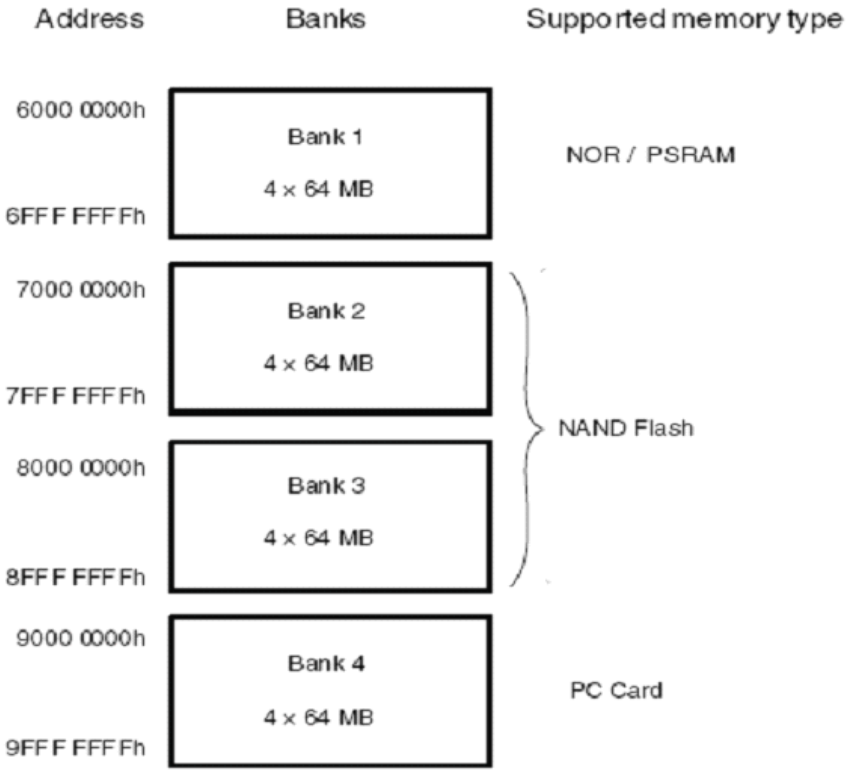
（2）NOR 闪存和 PSRAM 控制器

（3）NAND 闪存和 PC 卡控制器

（4）外部设备接口

要注意的是，FSMC 可以请求 AHB 进行数据宽度的操作。如果 AHB 操作的数据宽度大于外部设备（NOR 或 NAND 或 LCD）的宽度，此时 FSMC 将 AHB 操作分割成几个连续的较小的数据宽度，以适应外部设备的数据宽度。

2、FSMC 对外部设备的地址映像



从上图可以看出，FSMC 对外部设备的地址映像从 0x6000 0000 开始，到 0x9FFF FFFF 结束，共分 4 个地址块，每个地址块 256M 字节。可以看出，每个地址块又分为 4 个分地址块，大小 64M。对 NOR 的地址映像来说，我们可以通过选择 HADDR[27:26]来确定当前使用的是哪个 64M 的分地址块，如下页表格。而这四个分存储块的片选，则使用 NE[4:1]来选择。数据线/地址线/控制线是共享的。

HADDR[27:26](1)	选择的存储块
00	存储块1 NOR/PSRAM 1
01	存储块1 NOR/PSRAM 2
10	存储块1 NOR/PSRAM 3
11	存储块1 NOR/PSRAM 4

这里的 HADDR 是需要转换到外部设备的内部 AHB 地址线，每个地址对应一个字节单元。因此，若外部设备的地址宽度是 8 位的，则 HADDR[25:0]与 STM32 的 CPU 引脚 FSMC_A[25:0]一一对应，最大可以访问 64M 字节的空间。若外部

设备的地址宽度是 16 位的，则是 HADDR[25:1]与 STM32 的 CPU 引脚 FSMC_A[24:0]一一对应。在应用的时候，可以将 FSMC_A 总线连接到存储器或其他外设的地址总线引脚上。

二、LCD 控制电路设计

1、信号线的连接

STM32F10xxx FSMC 有四个不同的 banks（每个 64M 字节）可支持 NOR 以及其他类似的存储器。这些外部设备的地址线，数据先和控制线是共享的。每个设备的访问通过片选来决定，而每次只能访问一个设备。

FSMC 提供了所有的 LCD 控制器的信号：

FSMC_D[16:0] → 16bit 的数据总线

FSMC NEx：分配给 NOR 的 256M，再分为 4 个区，每个区用来分配一个外设，这四个外设的片选分为是 NE1-NE4，对应的引脚为：PD7—NE1，PG9—NE2，PG10-NE3，PG12—NE4

FSMC NOE：输出使能，连接 LCD 的 RD 脚。

FSMC NWE：写使能，连接 LCD 的 RW 脚。

FSMC Ax：用在 LCD 显示 RAM 和寄存器之间进行选择的地址线，即该线用于选择 LCD 的 RS 脚，该线可用地址线的任意一根线，范围：FSMC_A[25:0]。

注：RS = 0 时，表示读写寄存器；RS = 1 表示读写数据 RAM。

举例 1：选择 NOR 的第一个存储区，并且使用 FSMC_A16 来控制 LCD 的 RS 引脚，则我们访问 LCD 显示 RAM 的基址为 0x6002 0000，访问 LCD 寄存器的地址为：0x6000 0000。

举例 2：选择 NOR 的第四个存储区，使用 FSMC_A0 控制 LCD 的 RS 脚，则访问 LCD 显示 RAM 的基址为 0x6000 0002，访问 LCD 寄存器的地址为：0x6000 0000。

实际上，可用于 LCD 接口的 NOR 存储块信号如下：

FSMC_D[15:0]，连 16bit 数据线

FSMC_NE1，连片选：只有 bank1 可用

FSMC NOE：输出使能

FSMC NEW：FSMC 写使能

FSMC Ax：连接 RS，可用范围 FSMC_A[23:16]

2、时序问题

一般使用模式 B 来做 LCD 的接口控制，不适用外扩模式。并且读写操作的时序一样。此种情况下，我们需要使用三个参数：ADDSET，DATAST，ADDHOLD。这三个参数在位域 FSMC_TCRx 中设置。

当 HCLK 的频率是 72MHZ，使用模式 B，则有如下时序：

地址建立时间：0x1

地址保持时间：0x0

数据建立时间：0x5

好像有点理论化，呵呵，我们来编程看看就理解了。

三、LCD 驱动编写

请大家在阅读此部分之前，务必先阅读 LCD 的驱动 IC：ILI9325。查看在本期教程开始，我们给出的两个网址即可。

我们的思路是：既然想使用 STM32 的 FSMC 模块，就首先要使能它的时钟，并初始化这个模块。然后初始化 LCD 启动配置，这时候，我们才可以编写用户程序，来控制 LCD 显示各种字符、图形。

根据这个思路，我们调用函数：

RCC_AHBPeriphClockCmd(RCC_AHBPeriph_FSMC, ENABLE);

来使能 FSMC 模块所使用的时钟。呵呵，STM32 固件库果然给我们提供了超方便的库函数，我们无需了解任何东西，只要知道调用这个函数即可。项目开发进度大大加快。

下面配置 FSMC 初始化部分，采用的函数是 FSMC_LCD_Init();，来看下它的实现吧！

```
void FSMC_LCD_Init(void)
{
  FSMC_NORSRAMInitTypeDef  FSMC_NORSRAMInitStructure;
  FSMC_NORSRAMTimingInitTypeDef  FSMC_TimingInitStructure;

  FSMC_TimingInitStructure.FSMC_AddressSetupTime = 0x02;
  FSMC_TimingInitStructure.FSMC_AddressHoldTime = 0x00;
  FSMC_TimingInitStructure.FSMC_DataSetupTime = 0x05;
  FSMC_TimingInitStructure.FSMC_BusTurnAroundDuration = 0x00;
  FSMC_TimingInitStructure.FSMC_CLKDivision = 0x00;
  FSMC_TimingInitStructure.FSMC_DataLatency = 0x00;
  FSMC_TimingInitStructure.FSMC_AccessMode = FSMC_AccessMode_B;

  FSMC_NORSRAMInitStructure.FSMC_Bank = FSMC_Bank1_NORSRAM1;
```

```
  FSMC_NORSRAMInitStructure.FSMC_DataAddressMux = FSMC_DataAddressMux_Disable;
  FSMC_NORSRAMInitStructure.FSMC_MemoryType = FSMC_MemoryType_NOR;
  FSMC_NORSRAMInitStructure.FSMC_MemoryDataWidth = FSMC_MemoryDataWidth_16b;
  FSMC_NORSRAMInitStructure.FSMC_BurstAccessMode = FSMC_BurstAccessMode_Disable;
  FSMC_NORSRAMInitStructure.FSMC_WaitSignalPolarity = FSMC_WaitSignalPolarity_Low;
  FSMC_NORSRAMInitStructure.FSMC_WrapMode = FSMC_WrapMode_Disable;
  FSMC_NORSRAMInitStructure.FSMC_WaitSignalActive = FSMC_WaitSignalActive_BeforeWaitState;
  FSMC_NORSRAMInitStructure.FSMC_WriteOperation = FSMC_WriteOperation_Enable;
  FSMC_NORSRAMInitStructure.FSMC_WaitSignal = FSMC_WaitSignal_Disable;
  FSMC_NORSRAMInitStructure.FSMC_ExtendedMode = FSMC_ExtendedMode_Disable;
  FSMC_NORSRAMInitStructure.FSMC_WriteBurst = FSMC_WriteBurst_Disable;
  FSMC_NORSRAMInitStructure.FSMC_ReadWriteTimingStruct = &FSMC_TimingInitStructure;
  FSMC_NORSRAMInitStructure.FSMC_WriteTimingStruct = &FSMC_TimingInitStructure;

  FSMC_NORSRAMInit(&FSMC_NORSRAMInitStructure);
  FSMC_NORSRAMCmd(FSMC_Bank1_NORSRAM1, ENABLE);
}
```

上面的函数实现字体是小五，如果需要查看完整 FSMC-TFT-LCD 例程，请查看芯嵌 STM32 的光盘。此部分可作为一个模板，复制到您的项目文件中直接使用。实际上，控制 LCD 关键在于下面的初始化序列：

```
LCD_Write_Reg(0xCF); //Power control B
LCD_Write_RAM(0x00);     //采用默认参数
LCD_Write_RAM(0x81);
LCD_Write_RAM(0X30);

LCD_Write_Reg(0xED);     //Power on sequence control
LCD_Write_RAM(0x64);
LCD_Write_RAM(0x03);
LCD_Write_RAM(0X12);
LCD_Write_RAM(0X81);

LCD_Write_Reg(0xE8);     //Driver timing control A
LCD_Write_RAM(0x85);
LCD_Write_RAM(0x10);
```

```
LCD_Write_RAM(0x7A);

LCD_Write_Reg(0xCB);      //Power control A
LCD_Write_RAM(0x39);
LCD_Write_RAM(0x2C);
LCD_Write_RAM(0x00);
LCD_Write_RAM(0x34);
LCD_Write_RAM(0x02);

LCD_Write_Reg(0xF7);      //Pump ratio control
LCD_Write_RAM(0x20);

LCD_Write_Reg(0xEA);      //Driver timing control B
LCD_Write_RAM(0x00);
LCD_Write_RAM(0x00);

LCD_Write_Reg(0xC0);    //Power control
LCD_Write_RAM(0x1B);    //VRH[5:0]

LCD_Write_Reg(0xC1);    //Power control
LCD_Write_RAM(0x01);    //BT[2:0]

LCD_Write_Reg(0xC5);    //VCOM control
LCD_Write_RAM(0x30);
LCD_Write_RAM(0x30);

LCD_Write_Reg(0xC7);    //VCOM control2
LCD_Write_RAM(0XB7);

LCD_Write_Reg(0x36);    //Memory Access Control
LCD_Write_RAM(0x08);

LCD_Write_Reg(0x3A);
LCD_Write_RAM(0x55);

LCD_Write_Reg(0xB1);
LCD_Write_RAM(0x00);
LCD_Write_RAM(0x1A);

LCD_Write_Reg(0xB6);    // Display Function Control
LCD_Write_RAM(0x0A);
LCD_Write_RAM(0xA2);

LCD_Write_Reg(0xF2);    // 3Gamma Function Disable
```

```
LCD_Write_RAM(0x00);

LCD_Write_Reg(0x26);    //Gamma curve selected
LCD_Write_RAM(0x01);

LCD_Write_Reg(0xE0);    //Set Gamma
LCD_Write_RAM(0x0F);
LCD_Write_RAM(0x2A);
LCD_Write_RAM(0x28);
LCD_Write_RAM(0x08);
LCD_Write_RAM(0x0E);
LCD_Write_RAM(0x08);
LCD_Write_RAM(0x54);
LCD_Write_RAM(0XA9);
LCD_Write_RAM(0x43);
LCD_Write_RAM(0x0A);
LCD_Write_RAM(0x0F);
LCD_Write_RAM(0x00);
LCD_Write_RAM(0x00);
LCD_Write_RAM(0x00);
LCD_Write_RAM(0x00);

LCD_Write_Reg(0XE1);    //Set Gamma
LCD_Write_RAM(0x00);
LCD_Write_RAM(0x15);
LCD_Write_RAM(0x17);
LCD_Write_RAM(0x07);
LCD_Write_RAM(0x11);
LCD_Write_RAM(0x06);
LCD_Write_RAM(0x2B);
LCD_Write_RAM(0x56);
LCD_Write_RAM(0x3C);
LCD_Write_RAM(0x05);
LCD_Write_RAM(0x10);
LCD_Write_RAM(0x0F);
LCD_Write_RAM(0x3F);
LCD_Write_RAM(0x3F);
LCD_Write_RAM(0x0F);

LCD_Write_Reg(0x2A);     //列地址设置，起始地址为0，末地址为EFH，即239
LCD_Write_RAM(0x00);     //注意，起始地址必须 比 末地址 小！
LCD_Write_RAM(0x00);     //并且，地址要小于239或319，否则大出的部分将被忽略
LCD_Write_RAM(0x00);
LCD_Write_RAM(0xef);
```

```
LCD_Write_Reg(0x2B);      //注意事项参见 0x2A命令
LCD_Write_RAM(0x00);
LCD_Write_RAM(0x00);
LCD_Write_RAM(0x01);
LCD_Write_RAM(0x3f);

LCD_Write_Reg(0x11); //Exit Sleep
Delay(0XFFFFF);
LCD_Write_Reg(0x29); //display on
```

以上初始化序列代码约有 55 个参数需要配置，每个参数为何配置成这样，由于篇幅有限，这里不一一讲述，详情请参考 ILI9341 芯片手册。实际上，如果您时间有限，可以直接 copy 这部分的内容，只需要编写具体的用户实现部分。

当然，在初始化之前，我们要注意 LCD 的复位操作。对于每个 LCD 模块来说，想初始化之前，必须先复位，ILI9341 的复位，是低电平有效。芯嵌 STM32 开发板根据版本的不同，对复位的操作也不一样。其中一个版本的复位直接采用 STM32 的 CPU 复位，另一个版本的复位采用 PA4 引脚。两者都是可以的。

经过以上步骤初始化之后，现在 LCD 可以显示图片和字符了。为了测试，我们分别编写了字符和图片的测试文件，您可以参考。

以上就是 LCD 液晶程序的过程，期待大家拍砖。如需拍砖，请直接前往 www.51stm32.com 猛拍，以促进芯嵌 STM32 进一步改善教程，谢谢！

官方网站：http://www.51stm32.com/

官方淘宝：http://shop36353570.taobao.com/