青风手把手教你学 stm32f030 系列教程

---------- 库函数操作版本

出品论坛： www.qfv8.com 青风电子社区

作者： 青风
出品论坛: www.qfv8.com
淘宝店：http://qfv5.taobao.com
QQ 技术群：241364123
硬件平台：QF-STM32F030 开发板

2.6 点亮 LCD 液晶屏

2.6.1 原理分析

在嵌入式开发中，目前对于人机交互比较流行使用 TFT-LCD 彩屏，这是由于触摸屏的大量普及，价格上已经到达比较低的程度。他的用户体验方面是要远远胜过之前的一些单色屏，并且加入触摸之后，可以省略按钮，用于人机交互。

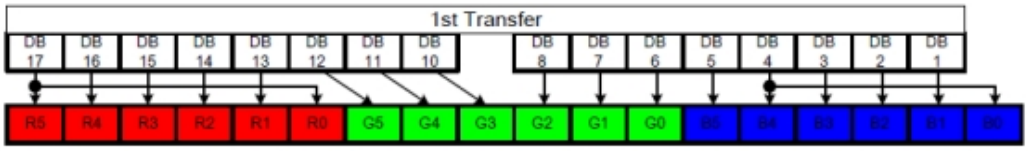
TFT-LCD 属于真彩显示屏，目前手机，掌上游戏机，平板电脑等都在使用 TFT。

QF_STM32F051 评估板使用的触摸屏的控制器为 ILI9328,采用 16BIT 并口输入数据，分辨率达到:320X240.分辨率其实是屏幕所具有的点的个数。230x240=76800 个点。而每个点都是由三原色“红 绿 蓝”RGB 组成，也就是说每个点可以配置成不同的颜色了。那么配置驱动采用 ILI9328 驱动器。ILI9328 一般存在四种总线接口控制：

1. I80 接口配置
2. VSYNC 接口配置
3. SPI 接口配置
4. RGB 接口配置

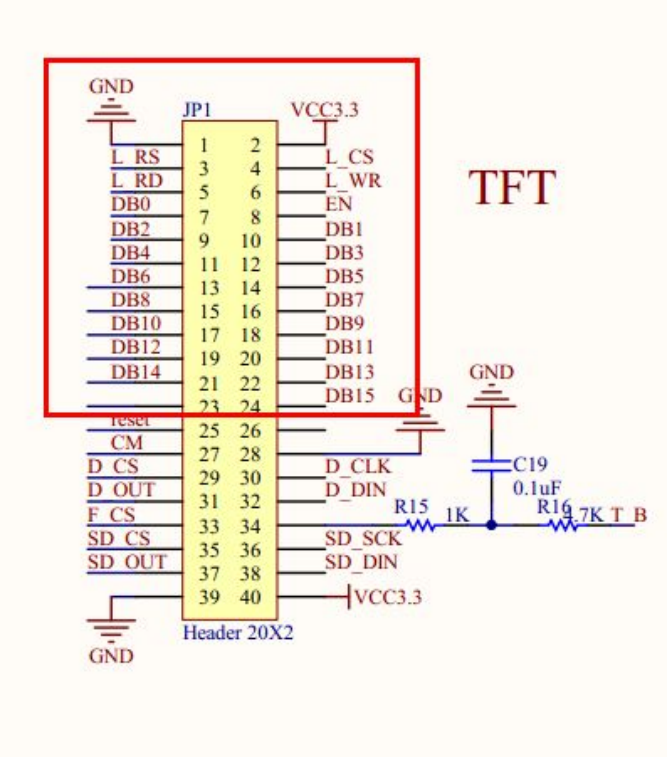
其实是采用串口和并口的不同配置启动方式。为了保证驱动速度，我们一般采用 I80 接口配置方式。本开发采用 16 位方式。

下面就是 16 位驱动 RPG 的寄存器配置方式：



R5_R0 为红色，G5_G0 为绿色，B5_B0 配置为蓝色。

2.6.2 硬件准备：



硬件连接：PB0-PB7---DB0-DB7 低 8 位

PC0-PC7---DB8-DB7 高 8 位

PD2---WR PC12---RD PC11---CS PC10---RS PB11---CM 其中

WR :写信号输入引脚，低电平时有效。

RD :读信号输入引脚，低电平时有效。

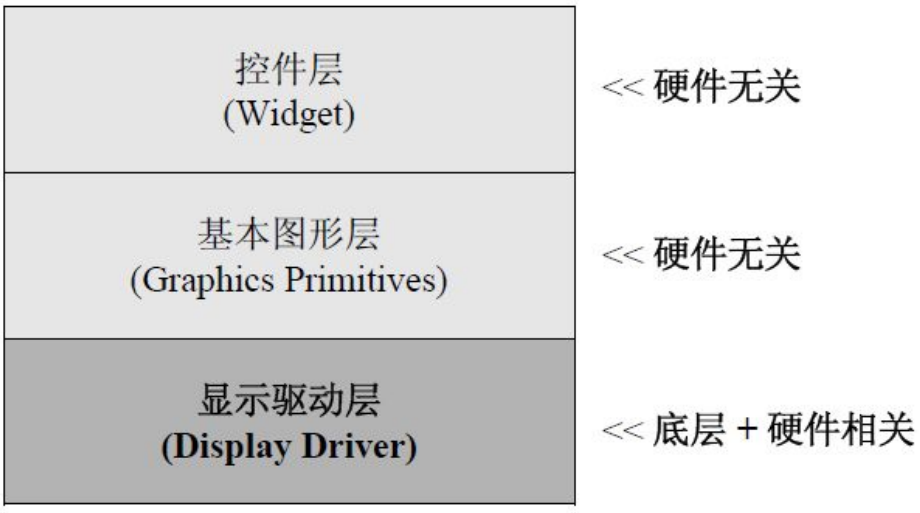
CS：TFT 液晶屏的片选信号，低电平时有效。

RS：写数据和写命令选择引脚。取高写数据，取低写命令。

CM：选择 8BIT 和 16BIT 选择引脚。

2.6.3 软件准备：

对于软件配置我们采用分层的方式：



这种方式虽然较为复杂，但是对于最底层的硬件驱动不同而已，上层的图形层和控制则不需要变换，可以很方便的移植到其他的显示屏上。这一节就是来讲解 **Display Driver** 的编写问题。

软件准备：

打开工程代码---LCD 刷屏--user 文件夹，点开工程后，keil 的工程树目录如下图所示：

整个工程目录很简单，只有会操作 **STM32F051** 的 **GPIO** 口就可以实现整个 **TFT** 的显示，由于这个工程中，为了工程代码的简便，我是直接写寄存器的，所以没有加入 **LIB** 库文件。

硬件连接好后，底层 **TFT-LCD** 的驱动是和硬件密切相关的，**TFT-LCD** 液晶屏实际上是一个 **16** 位/**8** 位的并行的数据接口。开发板上通过 **CM** 管脚选择了 **8** 位输入。**ILI9328** 的驱动简单的说包含两个重要的函数：第一：我们确定向那个寄存器写数据。第二：确定寄存器后，决定写什么样的数据。

这两个函数编写代码如下，由于直接操作 **IO** 寄存器比较简单，这里面我选择了直接写寄存器：

```
01.  void LCD_WR_DATA(uint16_t val)//写数据
02.  {
03.      Set_Rs;//开寄存器选择，选择写数据
04.      Clr_Cs;  //片选置低
05.      GPIOC->ODR &= 0xff00;   //PC 数据端口全部置低
06.      GPIOC->ODR|=(val>>8);//写入高八位
07.      Clr_nWr; //开写使能
08.      Set_nWr; //关使能
09.      GPIOC->ODR &= 0xff00;
10.      GPIOC->ODR|=((val)&(0x00ff));//写入低八位
11.      Clr_nWr; //开写使能
12.      Set_nWr; //关使能
13.      Set_Cs;  //关片选
14.  }
```

写寄存器命令如下：

```
15.  void LCD_WR_REG(uint16_t cmd)//写命令
16.  {
17.        Clr_Rs;;//开寄存器选择，选择写命令
18.        Clr_Cs;
19.        GPIOC->ODR &= 0xff00;
20.        GPIOC->ODR|=(cmd>>8);
21.        Clr_nWr;
22.        Set_nWr;
23.        GPIOC->ODR &= 0xff00;
24.        GPIOC->ODR|=((cmd)&(0x00ff));
25.         Clr_nWr; //开写使能
26.        Set_nWr; //关使能
27.         Set_Cs;
28.        }
```

下面来说明下上面两个函数：写命令和写数据进程大体相当，不同的在于最开始通过 Rs 置低或者置高来进行选择，取高写数据，取低写命令。GPIOB->ODR 采用 ODR 寄存器，表示 IO 口输出值。

对 ILI9328Q 驱动器常用的命令如下表：

编号	指令	各位描述																命令
	HEX	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
R0	0X00	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	OSC	打开振荡器/读取控制器型号
		1	0	0	1	0	0	1	1	0	0	1	0	0	0	0	0	
R3	0X03	TRI	DFM	0	BGR	0	0	HWM	0	ORG	0	I/D1	I/D0	AM	0	0	0	入口模式
R7	0X07	0	0	PTDE1	PTDE0	0	0	0	BASEE	0	0	GON	DTE	CL	0	D1	D0	显示控制
R32	0X20	0	0	0	0	0	0	0	0	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	行地址（X）设置
R33	0X21	0	0	0	0	0	0	0	AD16	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8	列地址（Y）设置
R34	0X22	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	写数据到GRAM
R80	0X50	0	0	0	0	0	0	0	0	HSA7	HSA6	HSA5	HSA4	HSA3	HSA2	HSA1	HSA0	行起始地址（X）设置
R81	0X51	0	0	0	0	0	0	0	0	HEA7	HEA6	HEA5	HEA4	HEA3	HEA2	HEA1	HEA0	行结束地址（X）设置
R82	0X52	0	0	0	0	0	0	0	VSA8	VSA7	VSA6	VSA5	VSA4	VSA3	VSA2	VSA1	VSA0	列起始地址（Y）设置
R83	0X53	0	0	0	0	0	0	0	VEA8	VEA7	VEA6	VEA5	VEA4	VEA3	VEA2	VEA1	VEA0	列结束地址（Y）设置

那么要向某个确定寄存器写入数据可以编写成如下，直接调用上面两个子函数：

```
29.  void LCD_WR_REG_DATA(uint16_t reg, uint16_t data)
30.  {
31.        LCD_WR_REG(reg);//确定要写入的寄存器
32.        LCD_WR_DATA(data);//确定写入寄存器的数据
33.        }
```

弄清楚如何向寄存器内部写命令后，下面就可以开始对 TFT-LCD 进行初始化，主要是多 ILI9328 驱动器进行初始化，首先初始化 IO 端口，然后按照 ILI9328 数据手册上的要求进行开机设置，详细信息请查看 PDF。那么我们设置 LCD 初始化函数为：

```
34.  void LCD_init (void)
35.  void LCD_init (void)
36.  { uint16_t i;
37.       //////////////////////////////////////////////////////////////////////管脚初始化
38.    RCC->AHBENR&=0xffe3ffff;
39.       RCC->AHBENR|=0x001c0000;
40.       GPIOC->MODER&=0xffff0000;            //
```

```
41.       GPIOC->MODER|=0x00005555;
42.       GPIOC->ODR &= 0xff00;
43.         GPIOC->MODER&= 0xfc0fffff;   //PD2---WR    PC12---RD PC11---CS    PC10---RS
44.      GPIOC->MODER|=0x01500000;
45.         GPIOC->BSRR = 0x00001c00;
46.         GPIOD->MODER&= 0xfffffffcf;
47.      GPIOD->MODER|=0x00000010;
48.         GPIOD->BSRR = 0x00000004;
49.       ////////////////////////////////////////////////////////////////
50.       delay(5);                                   /* Delay 50 ms                                  *
51.       LCD_WR_REG_DATA(0x00e7,0x0010);
52.       LCD_WR_REG_DATA(0x0000,0x0001);                 //start internal osc
53.       LCD_WR_REG_DATA(0x0001,0x0100);
54.       LCD_WR_REG_DATA(0x0002,0x0700);                     //power on sequence
55.       LCD_WR_REG_DATA(0x0003,(1<<12)|(1<<5)|(1<<4) );  //65K
56.       LCD_WR_REG_DATA(0x0004,0x0000);
57.       LCD_WR_REG_DATA(0x0008,0x0207);
58.       LCD_WR_REG_DATA(0x0009,0x0000);
59.       LCD_WR_REG_DATA(0x000a,0x0000);                     //display setting
60.       LCD_WR_REG_DATA(0x000c,0x0001);                     //display setting
61.       LCD_WR_REG_DATA(0x000d,0x0000);                     //0f3c
62.       LCD_WR_REG_DATA(0x000f,0x0000);
63.       //Power On sequence //
64.       LCD_WR_REG_DATA(0x0010,0x0000);
65.       LCD_WR_REG_DATA(0x0011,0x0007);
66.       LCD_WR_REG_DATA(0x0012,0x0000);
67.       LCD_WR_REG_DATA(0x0013,0x0000);
68.       for(i=50000;i>0;i--);
69.       for(i=50000;i>0;i--);
70.      LCD_WR_REG_DATA(0x0010,0x1590);
71.       LCD_WR_REG_DATA(0x0011,0x0227);
72.       for(i=50000;i>0;i--);
73.       for(i=50000;i>0;i--);
74.       LCD_WR_REG_DATA(0x0012,0x009c);
75.       for(i=50000;i>0;i--);
76.       for(i=50000;i>0;i--);
77.       LCD_WR_REG_DATA(0x0013,0x1900);
78.       LCD_WR_REG_DATA(0x0029,0x0023);
79.       LCD_WR_REG_DATA(0x002b,0x000e);
80.       for(i=50000;i>0;i--);
81.       for(i=50000;i>0;i--);
82.       LCD_WR_REG_DATA(0x0020,0x0000);
83.       LCD_WR_REG_DATA(0x0021,0x0000);
84.       ////////////////////////////////////////////////////////
```

```
85.         for(i=50000;i>0;i--);
86.         for(i=50000;i>0;i--);
87.         LCD_WR_REG_DATA(0x0030,0x0007);
88.         LCD_WR_REG_DATA(0x0031,0x0707);
89.         LCD_WR_REG_DATA(0x0032,0x0006);
90.         LCD_WR_REG_DATA(0x0035,0x0704);
91.         LCD_WR_REG_DATA(0x0036,0x1f04);
92.         LCD_WR_REG_DATA(0x0037,0x0004);
93.         LCD_WR_REG_DATA(0x0038,0x0000);
94.         LCD_WR_REG_DATA(0x0039,0x0706);
95.         LCD_WR_REG_DATA(0x003c,0x0701);
96.         LCD_WR_REG_DATA(0x003d,0x000f);
97.         for(i=50000;i>0;i--);
98.         for(i=50000;i>0;i--);
99.         LCD_WR_REG_DATA(0x0050,0x0000);
100.        LCD_WR_REG_DATA(0x0051,0x00ef);
101.        LCD_WR_REG_DATA(0x0052,0x0000);
102.        LCD_WR_REG_DATA(0x0053,0x013f);
103.        LCD_WR_REG_DATA(0x0060,0xa700);
104.        LCD_WR_REG_DATA(0x0061,0x0001);
105.        LCD_WR_REG_DATA(0x006a,0x0000);
106.        LCD_WR_REG_DATA(0x0080,0x0000);
107.        LCD_WR_REG_DATA(0x0081,0x0000);
108.        LCD_WR_REG_DATA(0x0082,0x0000);
109.        LCD_WR_REG_DATA(0x0083,0x0000);
110.        LCD_WR_REG_DATA(0x0084,0x0000);
111.        LCD_WR_REG_DATA(0x0085,0x0000);
112.        LCD_WR_REG_DATA(0x0090,0x0010);
113.        LCD_WR_REG_DATA(0x0092,0x0000);
114.        LCD_WR_REG_DATA(0x0093,0x0003);
115.        LCD_WR_REG_DATA(0x0095,0x0110);
116.        LCD_WR_REG_DATA(0x0097,0x0000);
117.        LCD_WR_REG_DATA(0x0098,0x0000);
118.        LCD_WR_REG_DATA(0x0007,0x0133);
119.        LCD_WR_REG_DATA(0x0020,0x0000);
120.        LCD_WR_REG_DATA(0x0021,0x0000);
121. }
```

下面编写一个进入 LCD 刷屏子函数，要求在屏幕上显示 8 条不同的颜色条纹。不同颜色的数据值如下面所列，我们只需要选择在不同区域填充不同颜色就可以。

```
122. #define      RED          0XF800    //红色
123. #define      GREEN        0X07E0        //绿色
124. #define      BLUE         0X001F    //蓝色
125. #define      WHITE        0XFFFF        //白色
```

```
126. #define        BLACK            0X0000        //黑色
127. #define        YELLOW           0XFFE0        //黄色
128. #define        ORANGE           0XFC08        //橙色
129. #define        GRAY             0X8430     //灰色
130. #define        LGRAY            0XC618        //浅灰色
131. #define        DARKGRAY         0X8410        //深灰色
132. #define        PORPO            0X801F        //紫色
133. #define        PINK             0XF81F     //粉红色
134. #define        GRAYBLUE         0X5458     //灰蓝色
135. #define        LGRAYBLUE        0XA651     //浅灰蓝色
136. #define        DARKBLUE         0X01CF        //深灰蓝色
137. #define    LIGHTBLUE        0X7D7C      //浅蓝色
138.
139. void GLCD_Test(void)
140. {
141.    uint16_t i,j;
142.    Clr_Cs;   //TFT 进行片选
143.    LCD_WR_REG_DATA(0x20, 0);//确定写入的 X 坐标
144.    LCD_WR_REG_DATA(0x21, 0);//确定其写入的 Y 坐标
145.    LCD_WR_REG(0x22);//开始写入 GRAM
146.
147.    for(i=0;i<320;i++)
148.       for(j=0;j<240;j++)//循环写颜色
149.       {
150.          if(i>279)LCD_WR_DATA(0x0000);
151.          else if(i>239)LCD_WR_DATA(0x001f);
152.          else if(i>199)LCD_WR_DATA(0x07e0);
153.          else if(i>159)LCD_WR_DATA(0x07ff);
154.          else if(i>119)LCD_WR_DATA(0xf800);
155.          else if(i>79)LCD_WR_DATA(0xf81f);
156.          else if(i>39)LCD_WR_DATA(0xffe0);
157.          else LCD_WR_DATA(0xffff);
158.       }
159.    Set_Cs;
160. }
```

那么上面一个简单的刷屏程序就出炉了，当然我们还可以写一些小的函数验证液晶屏的好坏。

主函数可以直接进行调用：

```
161.    #include "stm32f0xx.h"
162. #include "ili9328.h"
163.
164. void Delay()     //延迟函数
165. {
166.       int i,j;
```

```
167.    for(i=0;i<5000;i++)
168.            {
169.                    for(j=0;j<5000;j++);
170.              }
171. }
172.
173. int main()
174. {
175.        SystemInit();      // 时钟配置
176.        LCD_init();            // 液晶显示器初始化
177.        GLCD_Test();
178.        Delay();
179.
180.        LCD_Clear(WHITE);      // 全屏显示白色
181.        POINT_COLOR = BLACK; // 定义笔的颜色为黑色
182.        BACK_COLOR = WHITE;    // 定义笔的背景色为白色
183.        LCD_DrawRectage(20, 20, 180, 210, DARKBLUE);   // 画一个深蓝色边框的矩形
184.        LCD_Fill(60, 20, 179, 209, PINK);      // 画填充矩形
185.        POINT_COLOR = LIGHTBLUE; // 定义笔的颜色为浅蓝色
186.        LCD_DrawCircle(100, 100, 25); // 画一个圆
187.        LCD_ShowChar(120,20,'A');
188.        LCD_ShowString(140,20,"hello");
189.        while(1)
190.        {}
191. }
```

实验下载现象：

显示颜色条如下图所示：