STM32 之 UCGUI 移植

2012 年 7 月 6 日　　author:wzt

在移植之前当然要明白 ucgui 是什么，它就是一种嵌入式图形软件，用它可以制作操作界面，图像……等等，用于在显示设备上显示。要在自己的嵌入式设备中应用必须要进行移植……摸索了近一个星期今天终于移植完美，并成功绘制出一幅图像。

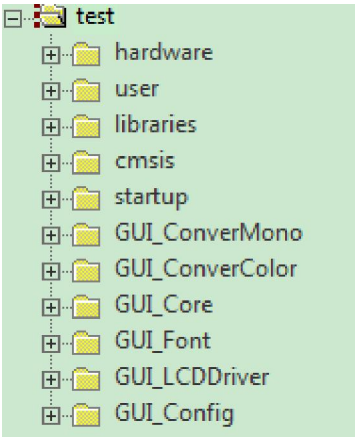
在移植前看过很多资料和例程。可能每个人的编程风格各具特色，从而大相径庭，让自己也甚是苦恼。看到有的完全不同风格的代码我还是宁愿自己重头开始自己一点点移植，当然遇到问题也只能靠自己。过程中遇到的个个令人抓狂但网上有搜不到解决办法的问题我会进行详细说明，希望对初次进行移植的同学能有所帮助。下面是我的吐血总结：

一、　做好准备工作：下载 UCGUI 的源代码（我用的是 3.90 版的）；建立好自己的 stm32 的工程并且编译无误;

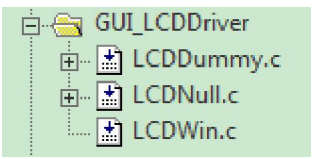
二、　打开源代码文件夹，

三、　复制 START\GUI 粘贴到工程文件夹中，复制 START\CONFIG 粘贴到工程目录下的 GUI 文件夹下，复制 SAMPLE\GUI_X 文件夹到工程目录下的 GUI\CONFIG 文件夹中。这样 ucgui 中的文件已经够了，接下来就是修改了。

四、　打开之前建好的 mdk 工程，把拷贝入工程中的 ucgui 文件加入工程中。整个工程的结构如下：

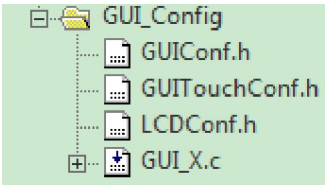


可以看到前面的 GUI 的都是 ucgui 的文件。细心地人可能会发现和 gui 文件夹里面的相比我们这里没有包含 JPEG, MemDev , MultiLayer , Widget , Wm 这 5 个文件夹的内容。因为这些文件起到的是扩展功能，在我们移植阶段可以先不添加，等到以后用到其中的功能时再选择添加。当然前提是在配置时要把相应的功能开关关掉，在下面的步骤中会提到。在上面的工程文件中 ConverMono , ConverColor ,Core ,Font 这四个目录下的文件是不用修改的。而我们要修改的文件在 LCDDriver , Config 这两个目录下。



LCDDriver 目录下的结构如图所示，其中我们用到的也仅仅只有

LCDDummy.c（也可以用其他两个，这三个文件实际上就是三个不同的模板，但功能一样，只是移植时修改的细节不一样。我选择修改 LCDDummy.c 这个文件）。前几天我看别人的例程，很多人用的是官方给出的对应型号的配置文件……比如 ili9320.c，ili9320_api.c，ili9320_gui.c，用这个的话就不用 LCDDmmy.c 了。那到底该用哪种呢？实际上官方给出了一些型号 lcd 控制器的配置文件，假如你的 lcd 有对应的配置文件，那么你两种方法都可以用。要是找不到你对应的型号，就必须用我使用的这种方法。我也更赞同用修改 LCDDummy.c 文件这种方法，应为它的通用性更强，它可以配置任何型号的 lcd 前提是你要编写好该 lcd 最底层的驱动。



这个是 Config 目录下的文件结构，别忘了把 GUI_X.C 添加进来，要不然编译的时候会有错误。

整个移植修改的文件都在这里了，其实也没几个，我修改的 LCDDummy.c GUIConf.h LCDConf.h 也就这三个文件。下面讲解如何修改。

五、 上层配置：也就是 GUI 一些功能的开关，打开 GUIConf.h 文件：按照如下代码配置：（注：0 是关，1 是开）

```
#ifndef GUICONF_H
#define GUICONF_H

#define GUI_OS                    (0) //操作系统的支持，当用到 ucos 时需要打开
#define GUI_SUPPORT_TOUCH         (0)   /* 触摸屏的支持，这里关闭*/
#define GUI_SUPPORT_UNICODE       (1)   /* 建议初始关闭，以后用汉字库时再打开*/

#define GUI_DEFAULT_FONT          &GUI_Font6x8     //定义字体大小
#define GUI_ALLOC_SIZE            5000   /*分配的动态内存空间 */

/*********************************************************************

#define GUI_WINSUPPORT            0   /* 窗口功能支持*/
#define GUI_SUPPORT_MEMDEV        0   /* 内存管理，建议先关闭*/
#define GUI_SUPPORT_AA            0   /* 抗锯齿功能，打开后可以提高显示效果，建议先关闭*/
#endif
```

上层配置完当然还有下层，就是 LCDConf.h 文件，这个也很简单，它原来的看起来比较复杂，我这里把它简化了，大家用的时候可以把我的拷贝过去替换掉原来的，代码如下：

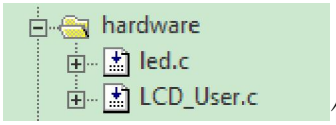
```
#ifndef LCDCONF_H
#define LCDCONF_H

#define LCD_XSIZE         (240)      /* lcd 的水平分辨率，根据自己的具体修改 */
#define LCD_YSIZE         (320)      /* lcd 的垂直分辨率，根据自己的具体修改*/
```

```
#define LCD_CONTROLLER      (-1)          /* lcd 控制器的具体型号，下面细讲*/
#define LCD_BITSPERPIXEL    (16)          /* lcd 颜色深度*/
#define LCD_FIXEDPALETTE    (565)         /* RGB 颜色位数*/
#define LCD_SWAP_RB         (1)          //红蓝反色交换
#define LCD_SWAP_XY         (1)
//LCD_MIRROR_X
#define LCD_INIT_CONTROLLER()   LCD_Config();//这里的 LCD_Config（）是你自己编写的 LCD 初始化函数，记好这里一定要替换（下面还会具体讲到自己编写的 lcd 底层文件）
#endif
```

关于配置中 LCD 型号问题，这里我们用的是通用型的方法也就是没有用到官方提供的具体型号配置文件，所以这里具体型号填-1，要是用官方配置文件那就要填具体型号了，比如我看他们的例程用到的是 ili9320，这里的-1 就改为 9320.

有的同学可能还看到 GUIConfig 目录下还有 GUITouchConf.h 文件，其实这里可以不管了，因为我们在 GUIConf.h 中有#define GUI_SUPPORT_TOUCH (0) 这条语句把触摸屏的功能已经关闭，当然以后用到触摸时还要具体配置，这里就不探讨了。其实这一部分很简单关键就是几个开关的控制和一个 lcd 初始化函数的替换。

六、麻烦的应该就是这一步的内容，这里也是最容易出问题的地方，当然按照我的步骤来一次移植成功也很容易。找到自己编写好的底层驱动。我的添加在工程中 hardware 目录下，我的是 lcd_user.c 如图所示：



你当然也可以把它加入到 GUI_LCDDriver 目录下，不管加到那个目录下，一定也把自己的接口 h 文件（我的是 lcd_user.h）所在位置包含进工程中。还有要注意的是自己的 lcd 底层驱动不能命名为 lcd.c 和 lcd.h，这样编译会出错，因为在 ucgui 内部文件中已经有 lcd.c 和 lcd.h 文件，比如我这里使用 lcd_user。

Lcd_user.c 在这里存放的是最底层的 lcd 驱动,而这个驱动程序最重要的就是三个函数：

1、lcd 初始化函数,我定义的是 void LCD_Config();记好这个初始化的函数名不能定义为 LCD_Init();因为在 UCGUI 文件中已经使用了这个函数名，所以会导致编译出错。还有就是返回看第五步配置文件最下面用到的就是这个函数，你的文件 lcd 初始化文件名换的话也别忘了给第五步配置中的也替换掉。

2、lcd 填充一个像素的函数,我定的的是 void LCD_SetPoint（int x，int y，int color）；这里有一个关键的问题，也是导致我第一次移植失败的原因，后来经过很多测试问题就在我定义的这个函数。我当时定义的这个函数只有两个参数即 x 和 y 坐标，把 color 给省略了。最终导致出现的问题就是初始化后黑屏，测试 gui 的字符显示函数，在显示区域就是一个矩形的黑条。所以记得编写底层驱动的时候这三个参数一个都不能少。

3、得到一个像素颜色的函数我定义的是 int LCD_ReadPoint（int x，int y）；这个也要根据你的 lcd 型号进行编写。

这以上的三个函数编写完后自己最好要测试一下确保能够正常使用。并

且同时在 lcd_user.h 文件之中声明这些函数以便和 ucgui 进行接口连接。
七、ucgui 配置好了，lcd 底层驱动也编写好了，所以现在最关键的就是要 ucgui 能够找到 lcd 的底层驱动并且进行连接。这一步就是修改 LCDDummy.c 文件了。

1、打开 LCDDummy.c 文件，在开头的头文件中增加语句#include"lcd_user.h" 把 lcd 的底层驱动接口包含进来。

2、将接下来这句话#if (LCD_CONTROLLER == -1) && (!defined(WIN32) | defined(LCD_SIMCONTROLLER)) 中后半部分的删掉，即剩下 #if (LCD_CONTROLLER == -1)这一句就行了，要是不修改的话整个 LCDDummy.c 文件都不会编译。

3、再往下找到 void LCD_L0_SetPixelIndex(int x, int y, int PixelIndex)这个函数把函数里面的都删掉然后用自己的填充一个像素的函数替换掉如下所示：

```
void LCD_L0_SetPixelIndex(int x, int y, int PixelIndex)
{
    LCD_SetPoint(x,y,PixelIndex);//画点
}
```

4、同上一步操作，只是这里用的是得到一个像素颜色的函数，如下：

```
unsigned int LCD_L0_GetPixelIndex(int x, int y)
{
  return LCD_ReadPoint(x,y);
}
```

八、到此移植的基本工作已经完成，然后就是编译测试了。编译通过后，修改 main 函数：如下

```
int main()
{
    SystemInit();           //系统时钟初始化，根据自己的定义进行修改
    GUI_Init();             //GUI 初始化
    GUI_SetBkColor(GUI_BLUE);   //设置背景颜色
    GUI_SetColor(GUI_WHITE);    //设置前景颜色，及字体和绘图的颜色
    GUI_Clear();                //按指定颜色清屏
    GUI_DrawCircle(100,100,50);    //画圆
    GUI_DispStringAt("Made By WZT!",10,10); //显示字符
    while(1);
}
```

如果屏幕上字符和圆形都可以正常显示，到这里就移植成功了。当然如果颜色和自己设定的不一样，就要修改自己的底层驱动。在七.3 步骤中，我们为了更易理解把原来的都删除掉了所以不能通过底层配置 LCD_SWAP_RB 激活红蓝基色交换开关来控制了。所以大家移植成功后可以再修改回来，这样将颜色不正常通过修改为 0 或 1 就可以控制红蓝基色位置交换，使颜色正常。

```
  #defined LCD_SWAP_RB (0)
```

九、移植成功了，发现刷屏速度很慢，没有直接操作 lcd 那样迅速。所以要想完美一点就要优化，当然优化的前提是可能要破坏掉一些原有函数的结构，导致可读性会降低。这个优化其实关键就在于 UCGUI 和 LCD 驱动连接的 LCDDummy.c 文件中。首先你要明白速度慢的原因是什么，经过我的测试刷屏使用的函数是：

LCD_L0_FillRect();而在我的 lcd 底层驱动中也编写好了和这个等效的函数 LCD_Fill();但是速度比这个要快很多倍，具体原因前者是一个一个像素填充写一个还要查找下一个，而我的底层驱动就是根据像素个数直接往 GRAM 中写数据，不用进行计算查找下一个像素点。所以就可以写成下面这个样子：

```
void LCD_L0_FillRect(int x0, int y0, int x1, int y1)
{
    LCD_Fill(x0,y0,x1,y1,LCD_COLORINDEX);
}
```

前提是这两个函数要有一样的参数。

另一个就是画线画图形函数，因为他们都是有填充像素和读取像素颜色构成的，那么我们主要就是优化这两个函数；LCD_L0_SetPixelIndex();

LCD_L0_GetPixelIndex();而这两个函数和我们的底层驱动编写的两个函数 LCD_SetPoint(); 和 LCD_ReadPoint(); 实际上是等效的。所以我们可以把 LCDDummy.c 中所有用到 LCD_L0_SetPixelIndex();LCD_L0_GetPixelIndex();的地方直接用我们的 LCD 底层驱动函数替换，减少一层函数的嵌套，也就省去了一步内部执行代码压栈出栈的步骤，也就节省了时间。这时候再测试应该可以明显感觉到速度快了很多。

到这里移植和优化也就完成了，因为本教程本着简单易理解为目的进行的移植，所以很多地方进行了简化，所以有些功能可能会丧失。当然理解了以后，某些功能不能用时我们再具体进行修改（一般修改 LCDDummy.c 文件）也就显得轻松了很多。下一步就是要把 ucgui 和 ucso 整合在一起，构成一个多任务的嵌入式软件系统。最后看一张我用 ucgui 画的图形：