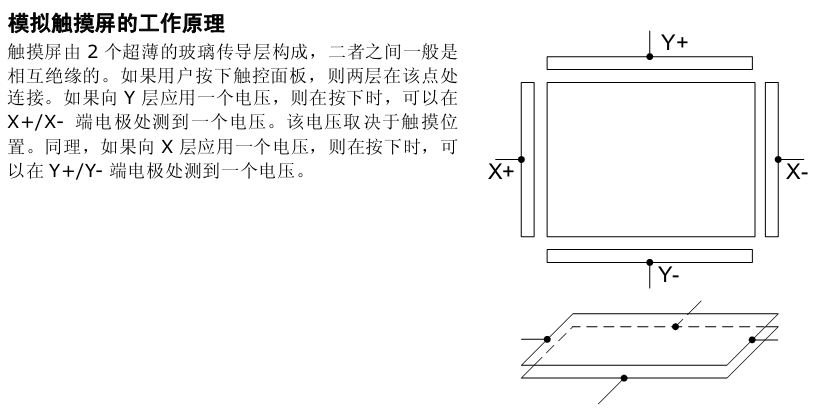
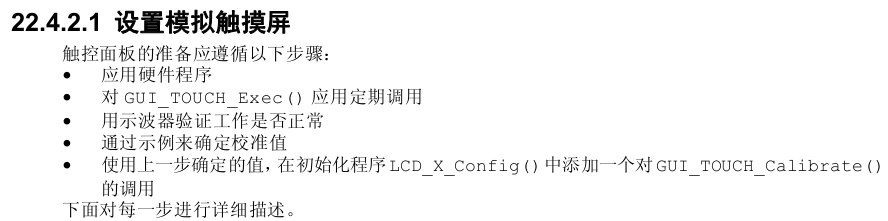
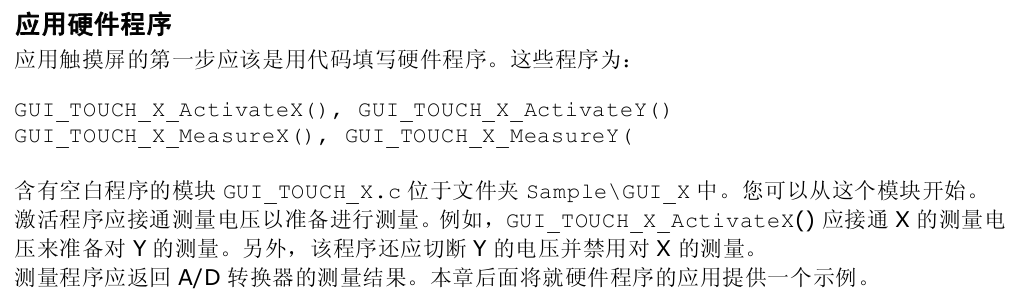
# 1.模拟触摸屏





# 2 测量触摸屏电压



在文件“GUI\_X\_Touch\_Analog.c”中：

int GUI\_TOUCH\_X\_MeasureX(void) {

return(touch\_read\_x());

}

int GUI\_TOUCH\_X\_MeasureY(void) {

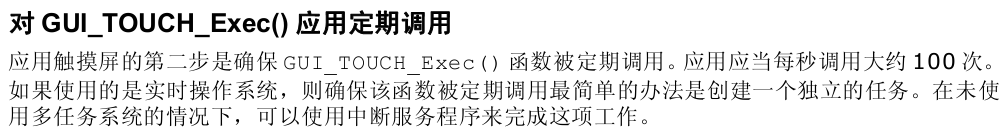
return(touch\_read\_y());

}

其余2个函数GUI\_TOUCH\_X\_ActivateX/ GUI\_TOUCH\_X\_ActivateY不用

# 3对GUI\_TOUCH\_Exec（）定期调用。

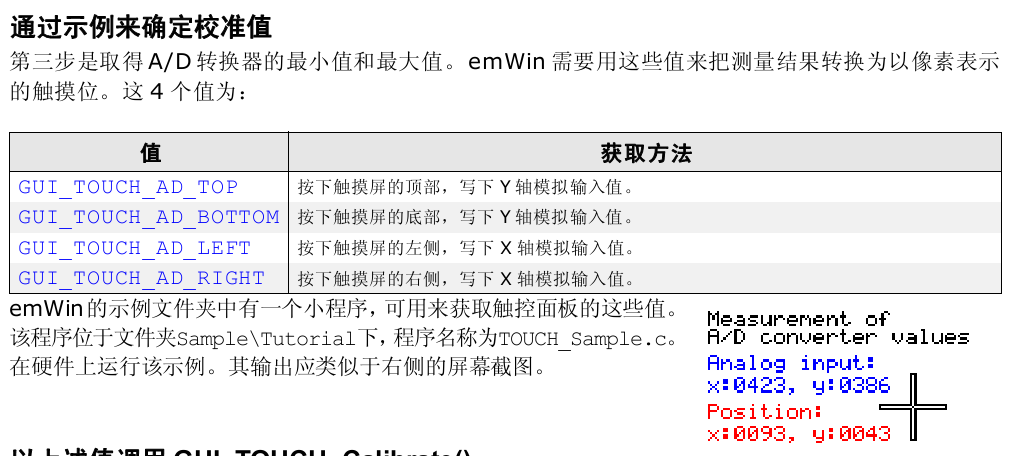
触摸屏触点位置的获得是通过调用GUI\_X\_Touch.c文件中的 GUI\_TOUCH\_Exec( )函数来实现的，该函数要在系统中开一个定时器，由于GUI说明书中要求每秒钟不少于100次的，访问GUI\_TOUCH\_Exec( )，用以记录触摸屏的点击状态与坐标。在操作系统中单独开一条线程定时操作处理。

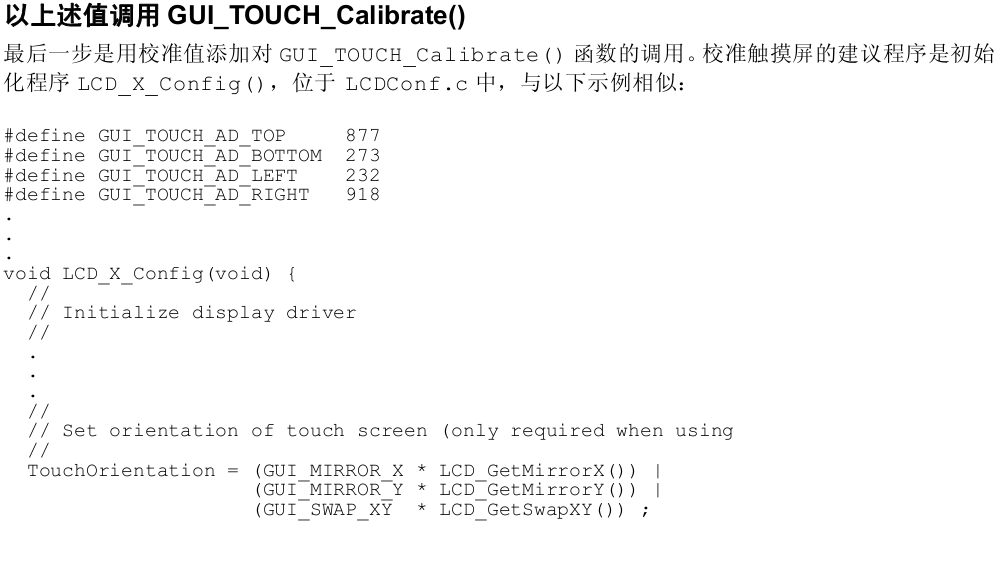


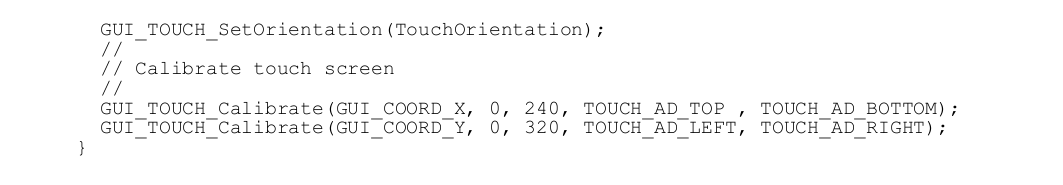
# 4 标定

## 4.1编写校准程序，获取校准值

获取显示区范围，用函数将显示区画出：







## 4.2运行时校准

运行时校准调用时间函数，在GUI\_X.c中

GUI\_TIMER\_TIME GUI\_X\_GetTime(void) {

return OS\_TimeMS;

}

void GUI\_X\_Delay(int ms) {

int tEnd = OS\_TimeMS + ms;

while ((tEnd - OS\_TimeMS) > 0);

}

在定时器，实现对OS\_TimeMS数值的1ms更新加1.

void TIM5\_IRQHandler(void)

{

//if (TIM\_GetITStatus(TIM5, TIM\_IT\_Update) != RESET)

{

TIM\_ClearITPendingBit(TIM5, TIM\_IT\_Update);

OS\_TimeMS++; //1.1ms更新GUI时间OS\_TimeMS

if(OS\_TimeMS % 10 == 0)

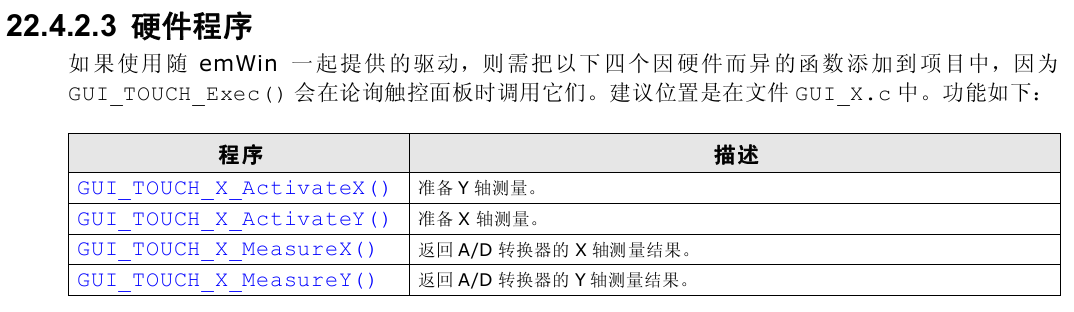
{

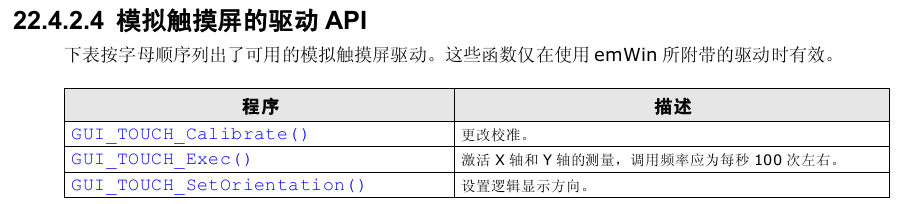
GUI\_TOUCH\_Exec(); //2.10ms调用GUI\_TOUCH\_Exec

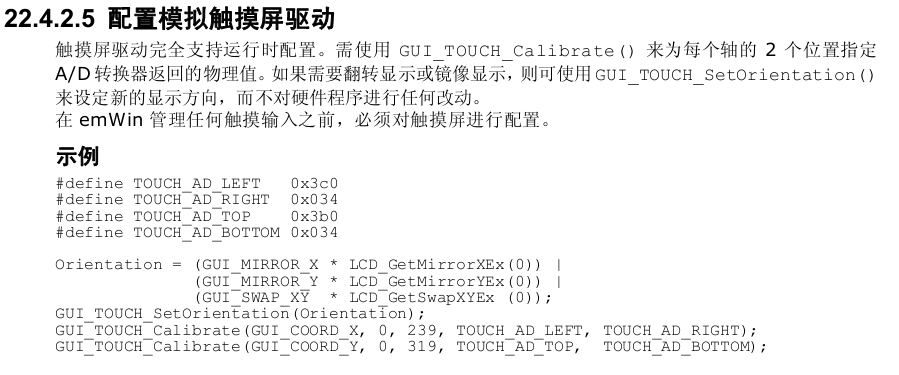
}

}

}

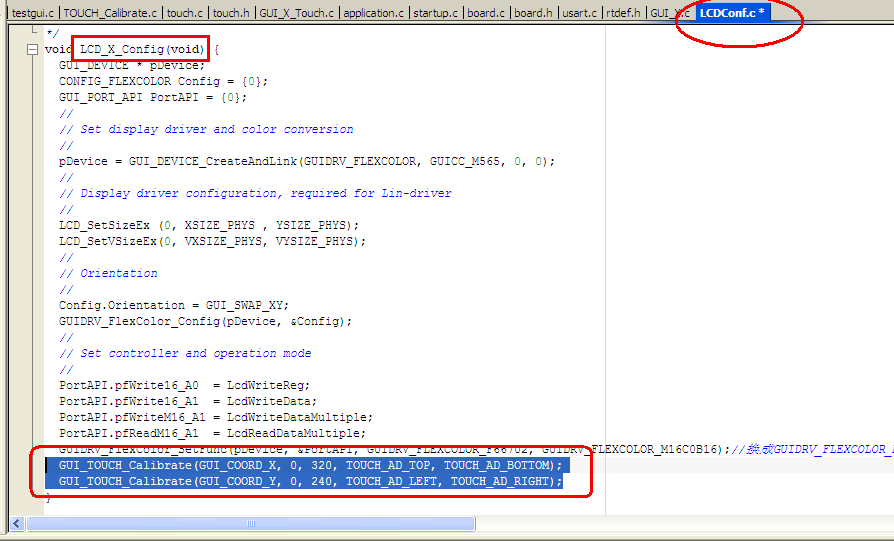






# 5.初始化中添加GUI\_TOUCH\_Calibrate()函数

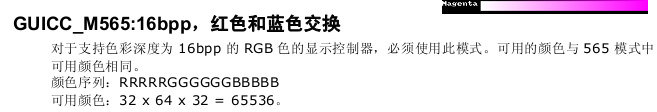
在LCDConf.c中：



# 6. LCDconf.c的配置说明

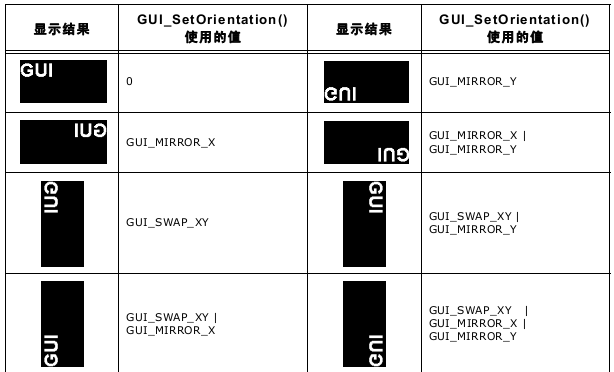
1）设置红蓝反色

pDevice = GUI\_DEVICE\_CreateAndLink(GUIDRV\_FLEXCOLOR, GUICC\_M565, 0, 0);



2）设置x y坐标轴交换

Config.Orientation = GUI\_SWAP\_XY;

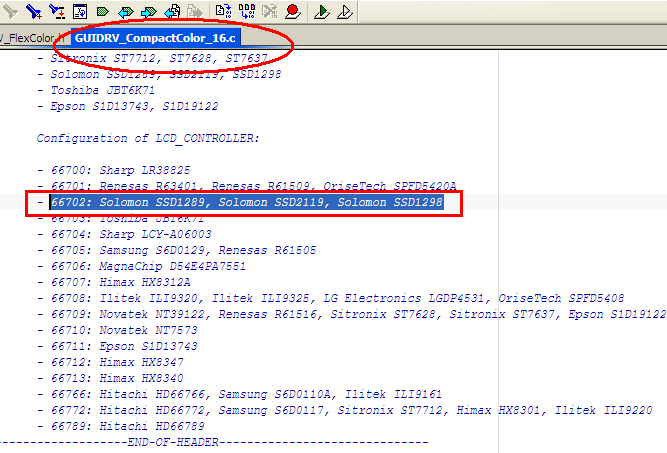


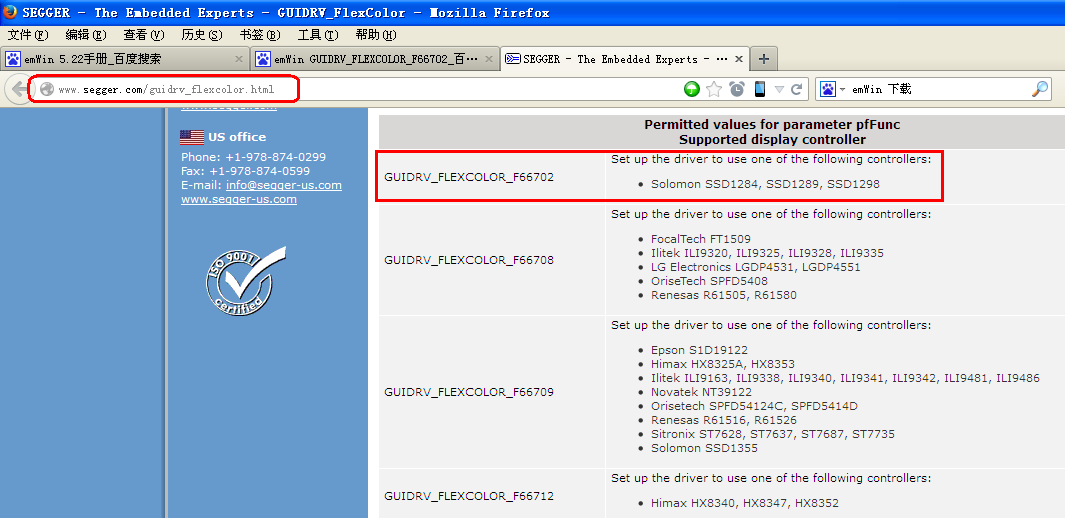


3）设置控制器

GUIDRV\_FlexColor\_SetFunc(pDevice, &PortAPI, GUIDRV\_FLEXCOLOR\_F66702, GUIDRV\_FLEXCOLOR\_M16C0B16);

在GUIDRV\_CompactColor\_16.c中，GUIDRV\_FLEXCOLOR\_F66702表示使用SSD1289控制器。



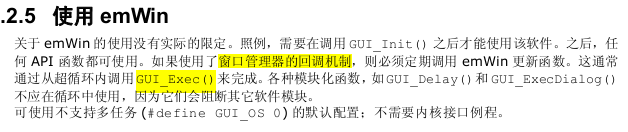
http://www.segger.com/guidrv\_flexcolor.html

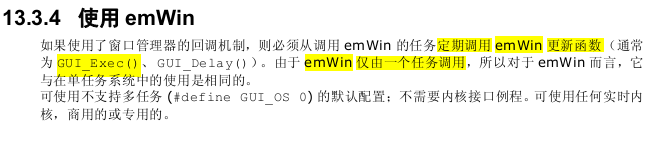
# 7.其它相关定义

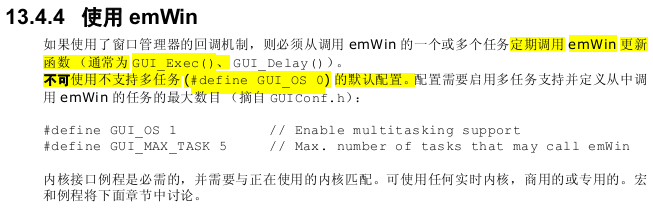
首先动GUIConf.c中的GUI\_NUMBYTES 为1024\*50，50可以小点，不要太大，太大编译器会编译会错误的，本例中改为30.

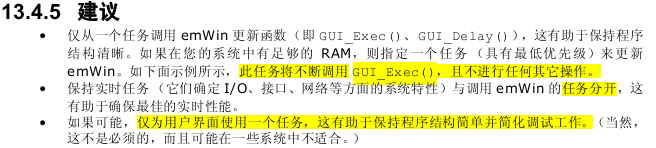
## 7.1 GUI相关

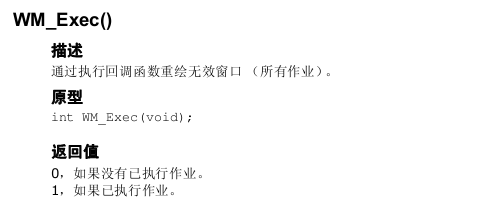
在操作系统中不用开一条线程定时调用GUI\_Exec() ，只需要定时调用GUI\_Delay().

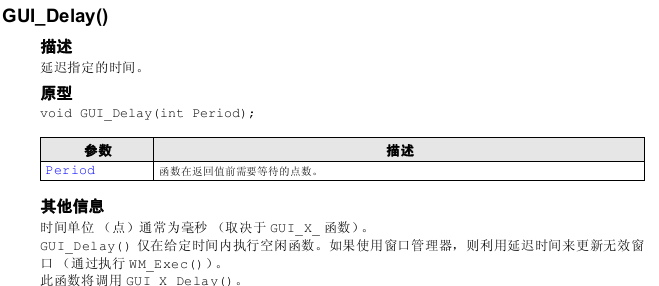








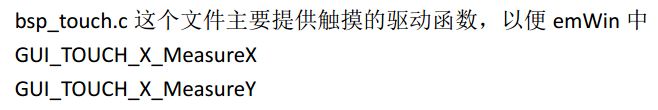


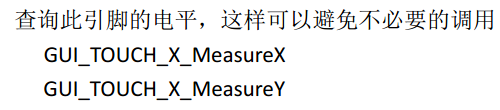


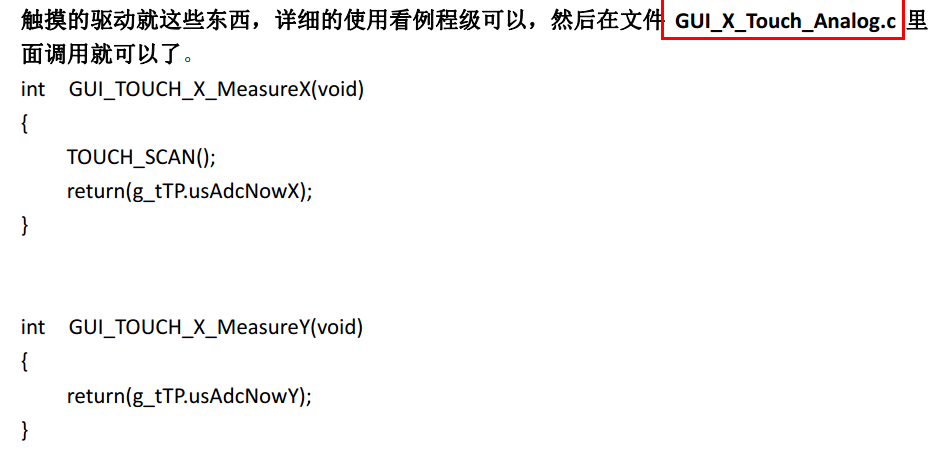
参考4.2定时器操作。

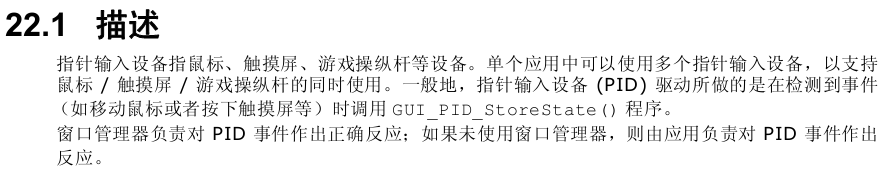
## 7.2 触摸屏相关

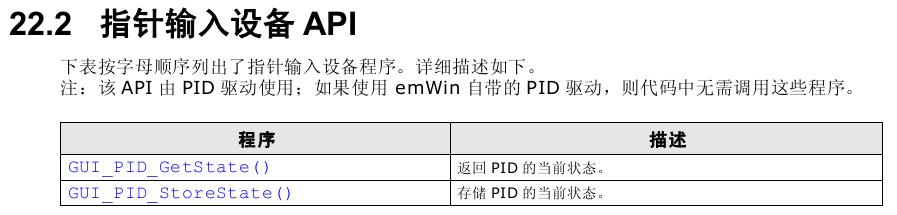
在定时器中断中，每隔10ms调用GUI\_TOUCH\_Exec() 。

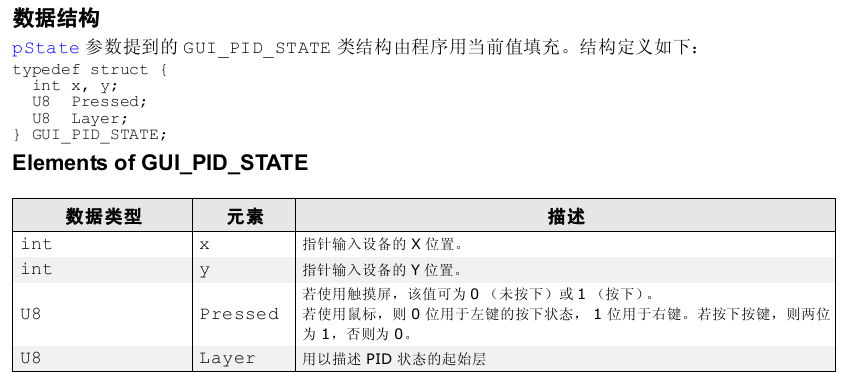


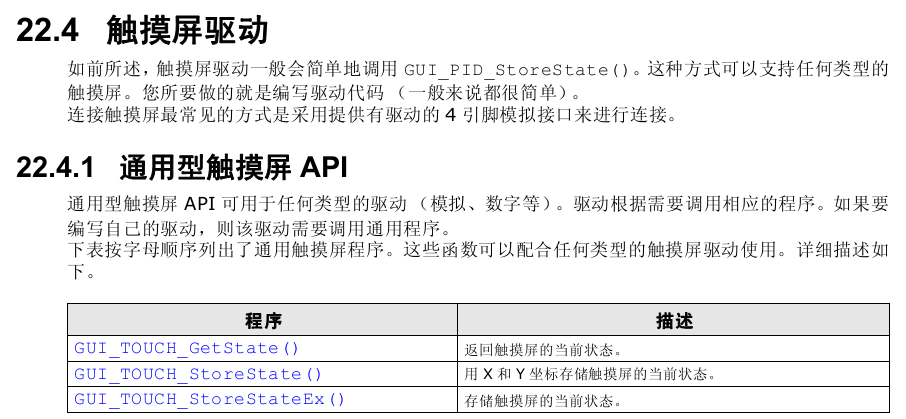


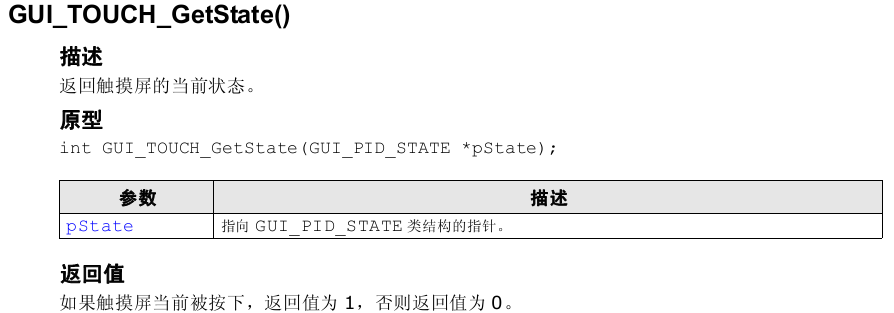


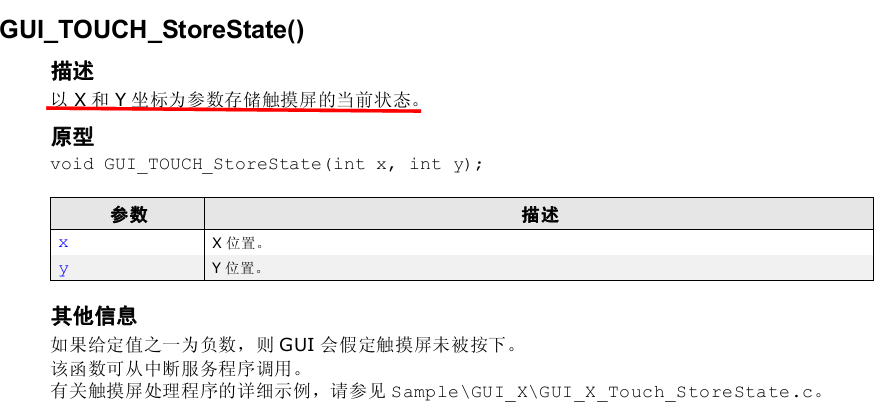


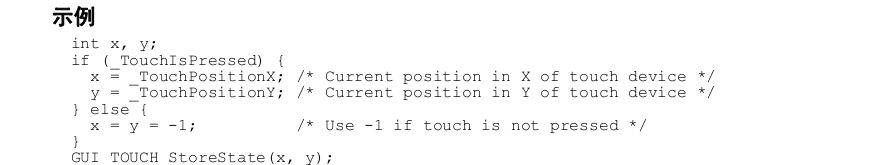




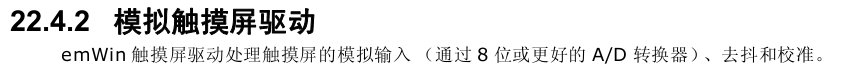












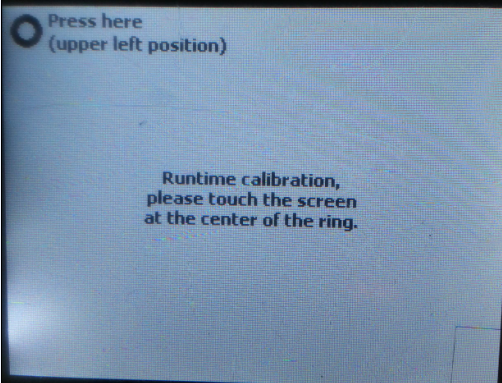
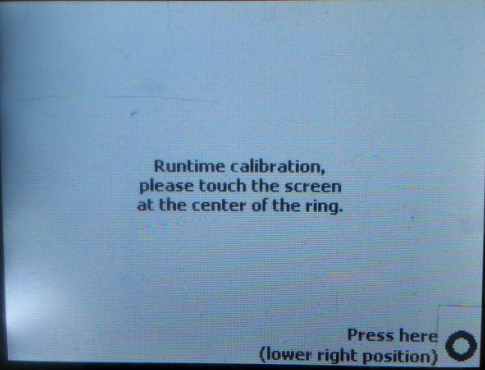
****

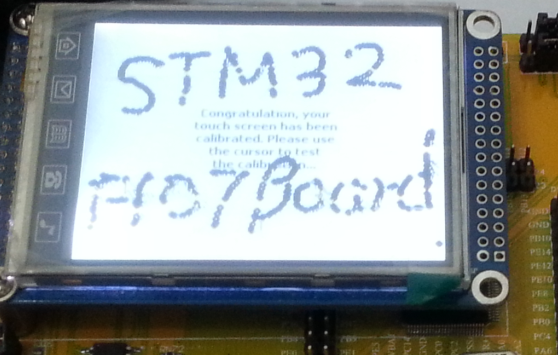
# 8.finsh shell的运行说明

1.test\_gui() ：显示外框，及中间字符 Data:

2.startgui()：初始化GUI，清屏

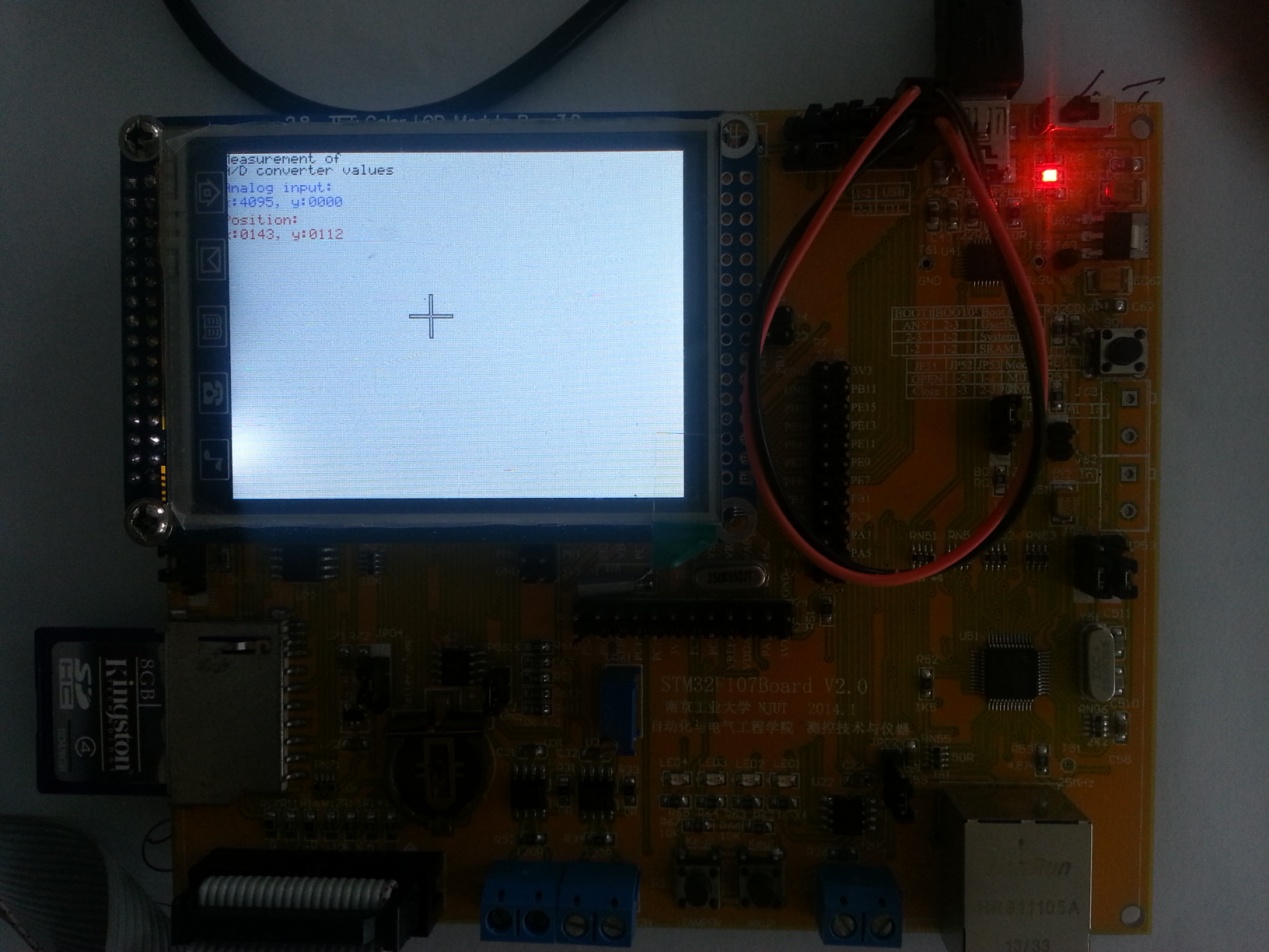
3.DynamicCali() ：动态校准函数



4.StopCali()：退出动态校准程序

5.TouchCali() ：触摸屏校准函数



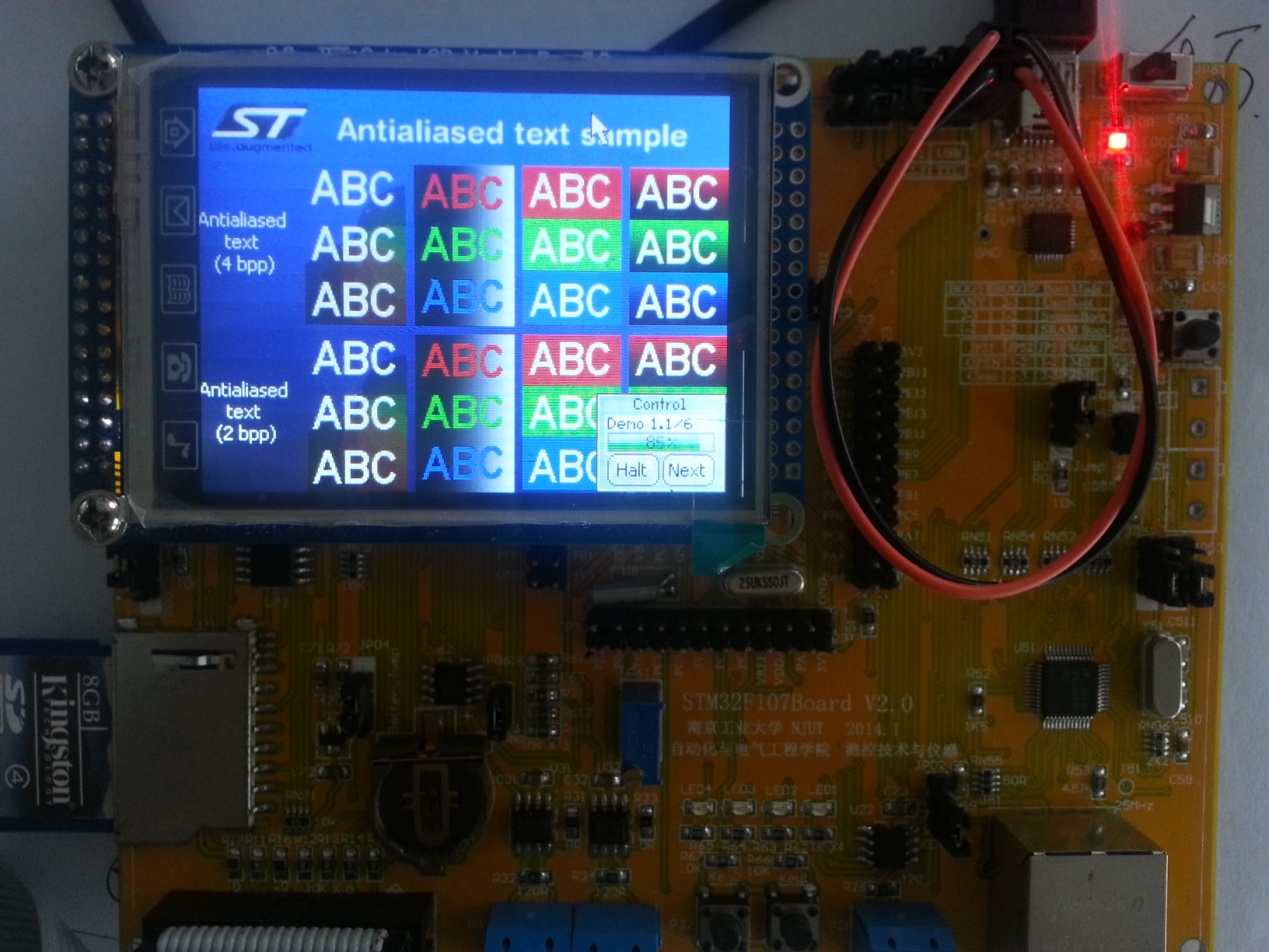
# 9. 关于硬件SPI与模拟SPI

Touch\_spi采用硬件SPI编写。

# 10．在RTThread中移植使用emWin

在finsh 中运行 startgui()

则会相继出现emWin中的Demo



注意：由于内存不够，不可以同时选择全部的demo, 在GUIDEMO.h 中选择相应的demo

