作者：.COM 缺氧® 来源：博客园 发布时间：2010-12-29 11:35 阅读：182 次 原文链接 [收藏]

准备资料

1 触摸屏控制芯片ADS7843中文资料[1]_百度文库

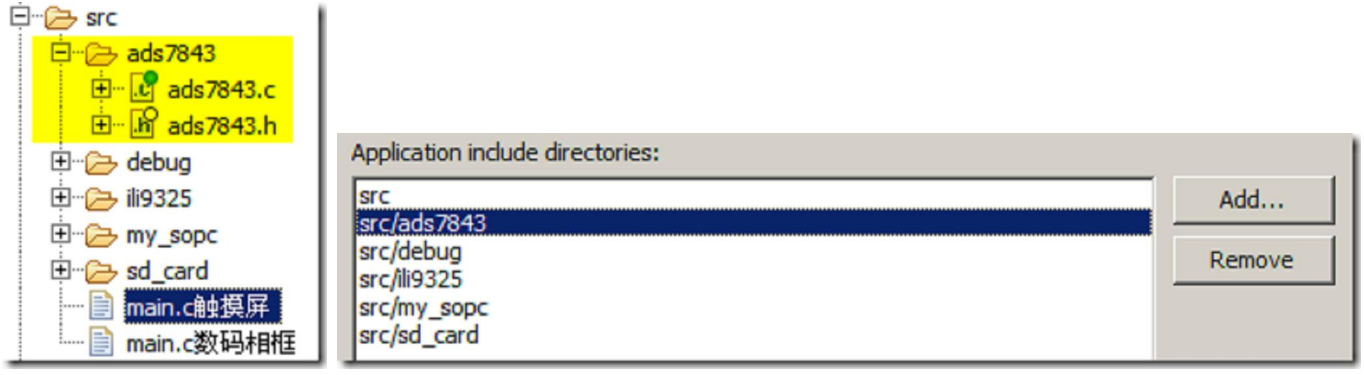
2 触摸屏原理与分类_百度文库

3 [原创].触摸屏滤波的一点心得

4 [笔记].如何使用Nios II的中断：PIO中断与定时器中断

操作步骤

步骤1 将ADS7843的驱动文件夹加入APP路径中

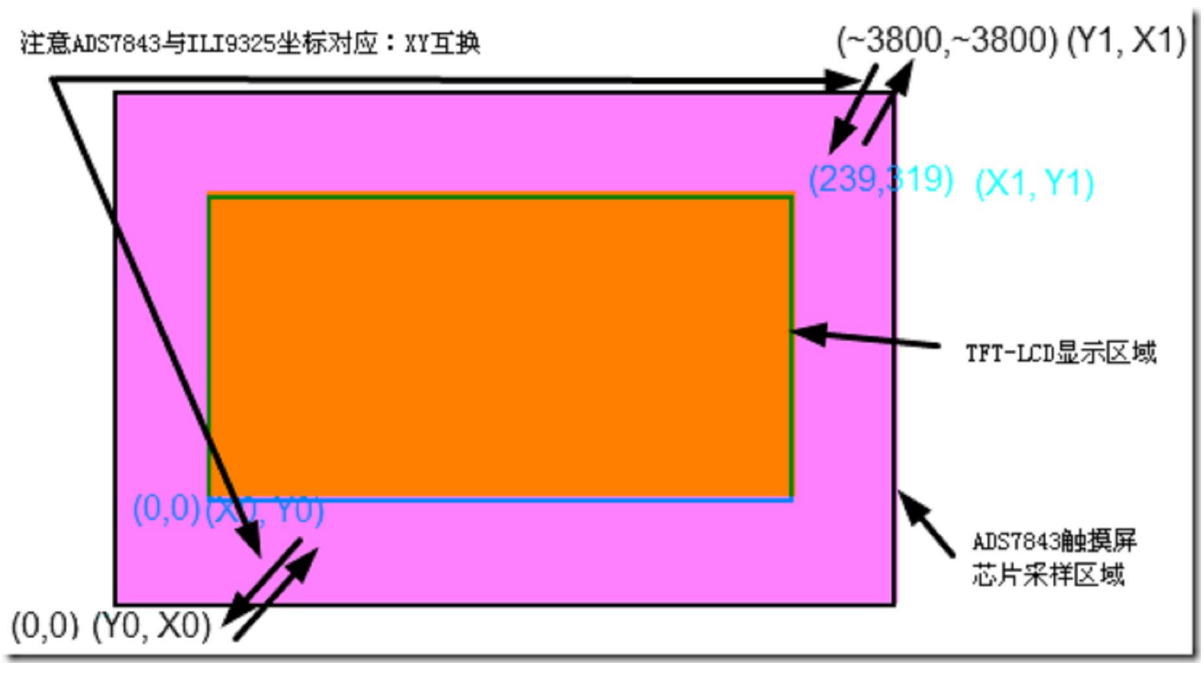


步骤2 编写ADS7843的驱动

我们先看下触摸屏芯片采样的坐标与TFT-LCD显示的坐标的区别和联系。图中的TFT-LCD方向为所定义方向，亦即

```
1 #define ID_AM    110
```

其XY坐标与ADS7843采样之坐标翻了。故ADS7843采样及滤波之后，需要把XY坐标翻回来。此外我们可以看到TFT-LCD显示区域是触摸屏采样芯片采样区域的子集，因此如若将程序移植到你的平台上，请坐相应的坐标校准动作。这和我们的触摸屏手机的校准功能是类似的的。



废话不多说，直接贴代码，有什么不明白的地方，请给我留言。

代码2.1 ads7843.h

```
01 #ifndef ADS7843_H_
02 #define ADS7843_H_
03
04 #include "my_types.h"
05 #include "my_regs.h"
06
07
08 #define CHX 0x90
09 #define CHY 0xD0
```

```
10
11
12  void ads_SPIStart(void);
13  void ads_SPIWrite(u8 cmd);
14  u16 ads_SPIRead(void);
15  bool ads_ReadXY(void);
16  bool ads_GetXY(void);
17  u8 *intostr(u16 n);
18
19  #endif /* ADS7843_H_ */
```

代码2.2 ads7843.c

```
001 #include "ads7843.h"
002 #include <unistd.h>
003
004
005 // 全局变量，用以储存坐标信息
006 u16 X=0, Y=0;
007
008
009 // SPI开始状态
010 void ads_SPIStart(void)
011 {
012   ads_CLK=0;
013   ads_nCS=1;
014   ads_DIN=1;
015   ads_CLK=1;
016   ads_nCS=0;
017 }
018
019
020 // SPI写一个byte
021 void ads_SPIWrite(u8 cmd)
022 {
023   u8 i;
024   ads_CLK=0;
025   for(i=0; i<8; i++) // 上升沿有效
026   {
027     ads_DIN = (cmd >> (7-i)) & 0x1; // MSB在前，LSB在后
028     ads_CLK=0; usleep(1);
029     ads_CLK=1; usleep(1);
030   }
031 }
032
033
034 // SPI读12个bit
035 u16 ads_SPIRead(void)
036 {
037   u8 i;
038   u16 temp=0;
039   for(i=0; i<12; i++) // 下降沿有效
040   {
041     temp<<=1;
042     ads_CLK=1; usleep(1);
043     ads_CLK=0; usleep(1);
044     if(ads_DOUT) temp++;
```

```
045   }
046   return temp;
047 }
048
049
050 // 读取ADS7843采集到X、Y值
051 // 返回：超出屏幕范围，则返回0
052 bool ads_ReadXY(void)
053 {
054   ads_SPIStart();
055   ads_SPIWrite(CHX);
056   ads_CLK=1; usleep(1);
057   ads_CLK=0; usleep(1);
058   X = ads_SPIRead();
059   ads_SPIWrite(CHY);
060   ads_CLK=1; usleep(1);
061   ads_CLK=0; usleep(1);
062   Y = ads_SPIRead();
063   ads_nCS=1;
064   if((X>350 && X<3800) && (Y>300 && Y<3800)) // 根据自己的屏自行矫正
065     return 1; // 读数成功(范围限制)
066   else
067     return 0; // 读数失败
068 }
069
070
071 // 处理从ADS7843读取的X、Y值，然后互换
072 #define SAMP_CNT      4
073 #define SAMP_CNT_DIV2 2
074 bool ads_GetXY(void)
075 {
076   u8 i, j, k, min;
077   u16 temp;
078   u16 tempXY[2][SAMP_CNT], XY[2];
079   // 采样
080   for(i=0; i<SAMP_CNT; i++)
081   {
082     if(ads_ReadXY())
083     {
084       tempXY[0][i] = X;
085       tempXY[1][i] = Y;
086     }
087   }
088   // 滤波
089   for(k=0; k<2; k++)
090   { // 降序排列
091     for(i=0; i<SAMP_CNT-1; i++)
092     {
093       min=i;
094       for (j=i+1; j<SAMP_CNT; j++)
095       {
096         if (tempXY[k][min] > tempXY[k][j]) min=j;
097       }
098       temp = tempXY[k][i];
099       tempXY[k][i] = tempXY[k][min];
100       tempXY[k][min] = temp;
```

```
    }
    // 设定阈值
    if((tempXY[k][SAMP_CNT_DIV2]-tempXY[k][SAMP_CNT_DIV2-1]) > 5)
      return 0;
    // 求中间值的均值
    XY[k] = (tempXY[k][SAMP_CNT_DIV2]+tempXY[k][SAMP_CNT_DIV2-1]) / 2;
  }
  // 矫正坐标
  Y = ((XY[0]-350)/11);
  X = ((XY[1]-400)/14);
  return 1;
}


// 整型转字符串（显示X、Y坐标，3个ASCII码）
u8 *intostr(u16 n)
{
  u8 *p;
  static u8 buf[3];
  p = &buf[3];
  *p     = (n/100) - ((n/1000)*10)+48;
  *(p+1) = (n/10) - ((n/100)*10) +48;
  *(p+2) = n- ((n/10) *10) +48;
  *(p+3) = 0;
  return  p;
}
```

步骤3 触摸屏驱动测试

代码 main.c

```
#include <unistd.h>                   // usleep()
#include "my_types.h"                 // 数据类型
#include "my_regs.h"                  // 自定义引脚及寄存器映射
#include "debug.h"                    // debug
#include "ili932x.h"                  // ILI9325
#include "ads7843.h"                  // ADS7843
#include "sd_card.h"                  // SD Card
#include "system.h"                   // 系统
#include "altera_avalon_pio_regs.h"   // PIO，ads_nIRQ
#include "sys/alt_irq.h"              // 中断


// 变量申明
extern u16 X, Y;


// 函数申明
vu16 nirq_isr_context; // 定义全局变量以储存isr_context指针
void nIRQ_Initial(void);
void nIRQ_ISR(void* isr_context);
void ResetTouch(void);


// 调试信息显示开关
#define ENABLE_APP_DEBUG // turn on debug message
#ifdef ENABLE_APP_DEBUG
    #define APP_DEBUG(x)    DEBUG(x)
```

```
#else
    #define APP_DEBUG(x)
#endif


// nIRQ中断初始化
void nIRQ_Initial(void)
{
  // 改写timer_isr_context指针以匹配alt_irq_register()函数原型
  void* isr_context_ptr = (void*) &nirq_isr_context;
  IOWR_ALTERA_AVALON_PIO_IRQ_MASK(ADS_NIRQ_BASE, 1); // 使能中断
  IOWR_ALTERA_AVALON_PIO_EDGE_CAP(ADS_NIRQ_BASE, 0); // 清中断边沿捕获寄存器
  // 注册ISR
  alt_ic_isr_register(
      ADS_NIRQ_IRQ_INTERRUPT_CONTROLLER_ID, // 中断控制器标号，从system.h复制
      ADS_NIRQ_IRQ,     // 硬件中断号，从system.h复制
      nIRQ_ISR,         // 中断服务子函数
      isr_context_ptr,  // 指向与设备驱动实例相关的数据结构体
      0x0);             // flags，保留未用
}


// 中断服务子函数
void nIRQ_ISR(void* isr_context)
{
  IOWR_ALTERA_AVALON_PIO_EDGE_CAP(ADS_NIRQ_BASE, 0); // 清中断边沿捕获寄存器

  if(ads_GetXY())
  {
    if((X>190 && X<240 && Y>300 && Y<320)) ResetTouch();
    else
    {
      ili_PutString(34, 289, intostr(X), Blue, White);
      ili_PutString(34, 305, intostr(Y), Blue, White);
      ili_PlotBigPoint(X, Y, Red);
    }
  }
}


//
void ResetTouch(void)
{
  ili_ClearScreen(White);
  ili_PutString(190, 305, (u8 *)("Clear"), Blue, White);
  ili_PutString(10, 289, (u8 *)("X:   0"), Blue, White);
  ili_PutString(10, 305, (u8 *)("Y:   0"), Blue, White);
}


int main(void)
{
  ili_Initial();
  nIRQ_Initial();
  ResetTouch();
  while(1);
```

```
84    return 0;
85  }
```

第34~47行，初始化nIRQ引脚下降沿中断；第51~65行，编写nIRQ中断函数。其他就不多说了。

测试效果如下：

源码下载

lcd_at_nios_nii_part.zip

目录

》点击查看原文...