作者：.COM 缺氧® 来源：博客园 发布时间：2010-12-27 22:59 阅读：478 次 原文链接 [收藏]

实在很抱歉，时间紧张，我只讲怎样从SD卡内读取bin文件（二进制文件），然后现在TFT-LCD上。

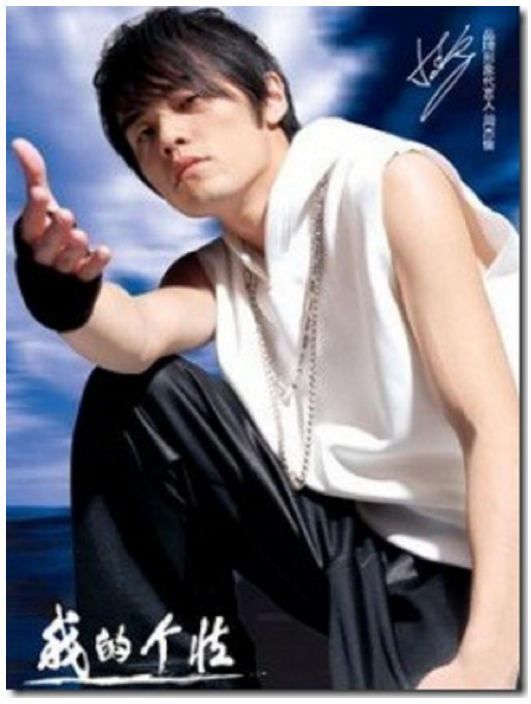
准备工具

1. Image2Lcd.zip

操作步骤

步骤1 寻找或制作240x320的图片

简单起见，我在谷歌图片里搜索240x320手机壁纸。随便选2张作为样本。



步骤2 使用Image2Lcd将图片转换为bin文件

我们先算一下，每个像素需要高8位+低8位，即16位数据，也就是2Byte；那么一张图片的话，就是240*320*2=15, 3600 Byte = 150 KB。

使用Image2Lcd依次打开图片，勾上如下所示的选项，保存为bin文件。

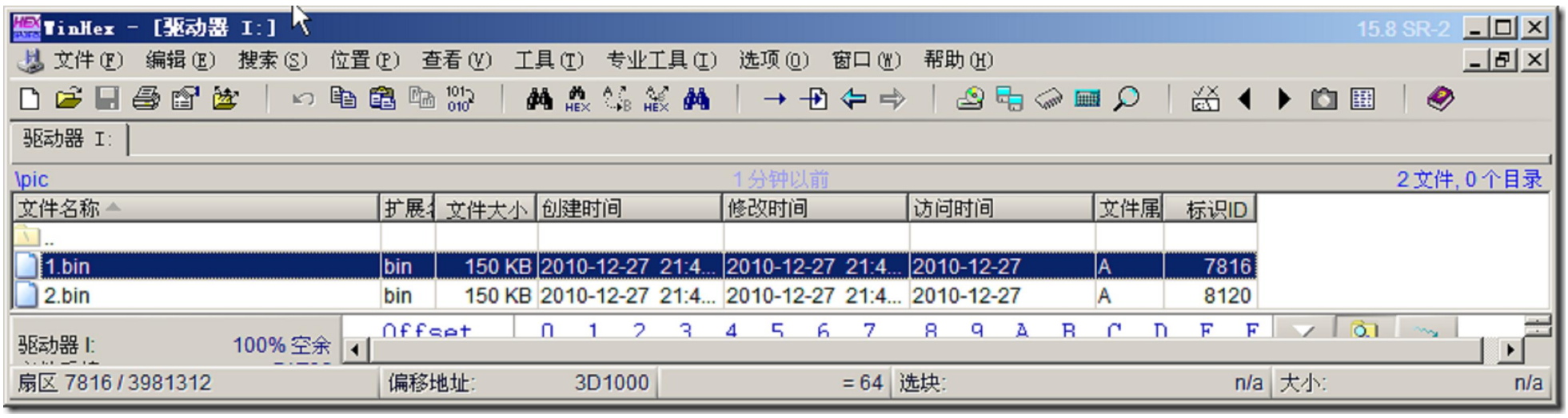
将生成的2个bin文件，拷贝到SD内的任意目录，比方说pic目录下。

步骤3 使用WinHex查找存放1.bin和2.bin的起始扇区及扇区内容

现在还不涉及文件系统，虽然我的sd卡被格式化为fat32。需要文件系统的朋友，请自行研究。

如何使用WinHex，我在前面的SD卡驱动一节有讲。在查看之前，我们先计算一下。1.bin是150KB，而每个扇区仅能存储512B，那么1.bin需要连续存储300个扇区。2.bin紧随1.bin之后，其存储的起始扇区地址应该滞后1.bin的起始扇区地址300个偏移量。但是我们发现 

，两者相距的扇区地址偏移量为304，那就意味着多出来4个扇区地址偏移量。我在这里提出这个问题，无意想解答为什么多了4个，而是想说明这个多出来扇区地址偏移量在FAT16和FAT32的文件系统中是不一样的。大家在做这个实验之前最好计算一下。



有了这个信息这后，我们就可以从1.bin的扇区起始地址连续读取300个扇区的数据，并同时将这些数据打到屏上，这就实现了显示SD内的数据的目的。当1.bin显示完毕，我们只需读扇区的起始地址递增304即可。

步骤4 测试

代码4.1 main.c

```
01 #include <stdio.h>                    // printf()
02 #include <unistd.h>                   // usleep()
03 #include "my_types.h"                 // 数据类型
04 #include "debug.h"                    // debug
05 #include "sd_card.h"                  // sd_card
06 #include "ili932x.h"                  // ili9325
07
08
09 #define ENABLE_APP_DEBUG // turn on debug message
10 #ifdef ENABLE_APP_DEBUG
11     #define APP_DEBUG(x)    DEBUG(x)
12 #else
13     #define APP_DEBUG(x)
14 #endif
15
16
17 #define PIC_NUM      2    // 图片数量
18 #define START_SECTOR 7816 // 数据存储的起始扇区
19 void DispPic_Demo(void)
20 {
21   u16 i, j;
22   u8 pic_num=0;          // 照片数量
23   u8 sector_buf[512];
24   u32 sector_addr;
25
26   sector_addr=START_SECTOR;
27   do
28   {
```

```
    ili_nCS=0;
    DB_o_EN;

    ili_SetDispArea(0, 0, 240, 320, 0, 0);
    for(j=0;j<300;j++)   //300表示一幅图片含有300x512字节的信息
    {
      SD_CARD_Read_Data_LBA(sector_addr+j, 512, sector_buf);//每次读出512字节放到缓冲区
      for(i=0;i<256;i++) //然后写到液晶屏，可以显示256个像素，每个像素16位即2个字节
        ili_WrData(sector_buf[2*i+1],sector_buf[2*i]);
    }

    ili_nCS=1;

    sector_addr += 304;
    pic_num++;
    usleep(2*1000*1000); // 延时2s
  }while(pic_num < PIC_NUM);
}


int main()
{
  ili_Initial();            // 初始化ILI9325
  while(SD_CARD_Init() != 0x55);// 初始化SD卡
  while(1)
  {
    DispPic_Demo();
  }
  return 0;
}
```

在编译和运行之前，把sd_card.h里面的调试宏注释掉，以减少因打印信息而注释程序。

代码4.2 sd_card.h片段

```
//#define ENABLE_SD_CARD_DEBUG // turn on debug message
```

同样的道理，若其他模块也有调试开关宏，最好也关断。当然这样，比较麻烦。我们可以再debug.h里面统一关闭调试宏开关也行。

代码4.3 debug.h片段

```
//#define ENABLE_STDOUT_DEBUG // turn on all of debug message using standard in/out
```

测试效果如下：

源码下载

lcd_at_nios_nii_part.zip

目录

1 [原创][连载].基于SOPC的简易数码相框 - Quartus II部分（硬件部分）

2 [原创][连载].基于SOPC的简易数码相框 - Nios II SBTE部分（软件部分）- 配置工作

3 [原创][连载].基于SOPC的简易数码相框 - Nios II SBTE部分（软件部分）- SD卡（SPI模式）驱动

4 [原创][连载].基于SOPC的简易数码相框 - Nios II SBTE部分（软件部分）- TFT-LCD（控制器为ILI9325）驱动

5 [原创][连载].基于SOPC的简易数码相框 - Nios II SBTE部分（软件部分）- 从SD卡内读取图片文件，然后显示在TFT-LCD上

6 [原创][连载].基于SOPC的简易数码相框 - Nios II SBTE部分（软件部分）- 优化工作

7 [原创][连载].基于SOPC的简易数码相框 - Nios II SBTE部分（软件部分）- ADS7843触摸屏驱动测试

》点击查看原文...