# mini2440裸机试炼之IIS——音乐播放器

## IIS

**信号频率设置**

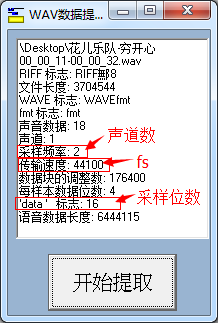
IIS（Inter-IC Sound）由飞利浦公司开发，是一种常用的音频设备接口，主要用于CD、MD、MP3等设备。

s3c2440一共有5个引脚用于IIS：IISDO、IISDI、IISSCLK、IISLRCK和CDCLK。前两个引脚用于数字音频信号的输出和输入，另外三个引脚都与音频信号的频率有关，可见要用好IIS，就要把信号频率设置正确。

**IISSCLK为串行时钟，每一个时钟信号传送一位音频信号。**因此IISSCLK的频率＝声道数×采样频率×采样位数，如采样频率fs为44.1kHz，采样的位数为16位，声道数2个（左、右两个声道），则IISSCLK的频率＝32fs＝1411.2kHz。

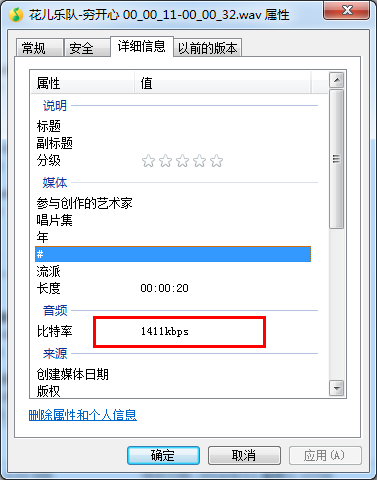
**1、Fs 2、采样的位数 3、声道数**

**使用 wav数据提取器 查看**



**采样频率**

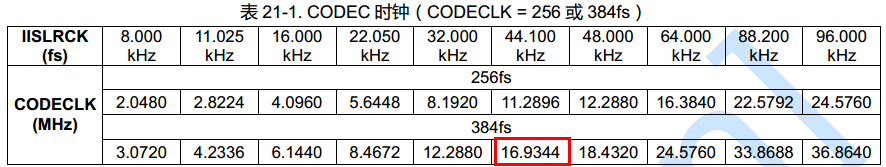
右键音乐文件🡪属性🡪详细信息



PCLK（50,000,000）经过两个预分频器处理后分别得到IISSCLK、IISLRCK和CDCLK，寄存器**IISPSR**是IIS预分频器寄存器，5~9位是预分频器A，0~4位是预分频器B，一般来说，这两个预分频器的值N相等，即只要知道一个，另一个也就知道，而这里我们是通过CDCLK来计算预分频器B的值N的，即CDCLK＝PCLK / (N＋1)。

我设置主时钟频率选择384fs，由下表，因fs=44.1kHZ，CDCLK=16.9344。可知**N=2**。

|  |
| --- |
| //预分频器为2，所以CDCLK=PCLK/(2+1)=16.66666kHz（和表中的16.9344有偏差）  rIISPSR **=** 2**<<**5**|**2**;** |



IIS配置

|  |  |
| --- | --- |
| |  | | --- | | //配置IIS接口  rGPEUP **=** rGPEUP **&** **~(**0x1f**)** **|** 0x1f**;** //上拉无效，GPE[4:0] 1 1111  rGPECON **=** rGPECON **&** **~(**0x3ff**)** **|** 0x2aa**;**    //配置s3c2440的IIS寄存器  //预分频器为2，所以CDCLK=PCLK/(2+1)=16.66666kHz  rIISPSR **=** 2**<<**5**|**2**;**  //无效DMA，输入空闲，预分频器有效，开启IIS  rIISCON **=** **(**0**<<**5**)|(**0**<<**4**)|(**0**<<**3**)|(**1**<<**2**)|(**1**<<**1**);**  //PCLK为时钟源，输出模式，IIS模式，每个声道16位，CODECLK=384fs，SCLK=32fs  rIISMOD **=** **(**0**<<**9**)|(**0**<<**8**)|(**2**<<**6**)|(**0**<<**5**)|(**0**<<**4**)|(**1**<<**3**)|(**1**<<**2**)|(**1**<<**0**);**  rIISFCON **=** **(**0**<<**15**)|(**1**<<**13**);** //输出FIFO正常模式，输出FIFO使能 | |

## UDA1341

在我看来，**IIS**就是实现将音频（WAV）数据按一定频率发送一定大小的数据给FIFO或者DMA，**UDA1341**就是将这些音频数据转换成电信号通过音频接口发送出去，而关于uda1341的配置是参考来的

|  |
| --- |
| //通过io口模拟L3总线写数据  //mode：1为地址模式，0为数据模式  //关于地址模式和数据模式以及传输时序注意参考数据手册  static void write\_UA1341**(**U8 data**,** U8 address**)**  **{**  int i**,**j**;**  **if(**address **==** 1**)** **{**  rGPBDAT **=** rGPBDAT**&(~(**L3D **|** L3M **|**L3C**))** **|**L3C**;** //地址模式，根据手册L3M为LOW,L3C为high  **}** **else** **{**  rGPBDAT **=** rGPBDAT **&** **(~(**L3D **|**L3M **|**L3C**))** **|(**L3M**|**L3C**);** //数据模式 L3M为高  **}**  Delay**(**1**);**  //传输数据  **for(**i**=**0**;**i**<**8**;**i**++)**  **{**  **if(**data **&** 0x1**)** // H  **{**  rGPBDAT **&=** **~**L3C**;** //L3C=L  rGPBDAT **|=** L3D**;** //L3D=H  **for(**j**=**0**;**j**<**5**;**j**++)**  **;** //等待一段时间  rGPBDAT **|=** L3C**;** //L3C=H  rGPBDAT **|=** L3D**;** //L3D=H  **for(**j**=**0**;**j**<**5**;**j**++)**  **;** //等待一段时间  **}**  **else** // L  **{**  rGPBDAT **&=** **~**L3C**;** //L3C=L  rGPBDAT **&=** **~**L3D**;** //L3D=L  **for(**j**=**0**;**j**<**5**;**j**++)**  **;** //等待一段时间  rGPBDAT **|=** L3C**;** //L3C=H  rGPBDAT **&=** **~**L3D**;** //L3D=L  **for(**j**=**0**;**j**<**5**;**j**++)**  **;** //等待一段时间  **}**  data **>>=** 1**;**  **}**  rGPBDAT **=** rGPBDAT **&** **~(**L3D **|** L3M **|** L3C**)** **|** **(**L3C **|** L3M**);** //L3M=H,L3C=H |

UDA1341初始化

|  |
| --- |
| //UDA1341初始化  //配置L3接口总线，GPB2:L3MODE, GPB3:L3DATA, GPB4:L3CLOCK  rGPBCON **=** 0x015550**;** //输出  rGPBUP **=** 0x7ff**;** //上拉无效  rGPBDAT **=** 0x1e4**;**    rGPBDAT **=** rGPBDAT **&** **(~(**L3M **|**L3C **|**L3D**))** **|(**L3M**|**L3C**);** //将L3CLOCK和L3MODE置高，准备开始传输  ////根据UDA1341TS数据手册14页中的操作顺序，首先在地址模式下，  //选择操作地址000101xx +10(STATUS)=0X16  write\_UA1341**(**0x16**,**1**)** **;**  write\_UA1341**(**0x60**,**0**);** // 0，1 ，10，000，0 复位    write\_UA1341**(**0x16**,**1**)** **;**  write\_UA1341**(**0x10**,**0**);** //0,0,01, 000,0 : 状态0, 384fs,IIS,no DC-filtering    write\_UA1341**(**0x16**,**1**)** **;**  write\_UA1341**(**0xc1**,**0**);** //1,0,0,0, 0,0,01:状态1,  //Gain of DAC 6 dB,Gain of ADC 0dB,ADC non-inverting,  //DAC non-inverting,Single speed playback,ADC-Off DAC-On |

## 界面素材



## WAV音频文件的制作

1. 网上下载无损音乐（我下载了WAV格式和APE格式的）
2. 打开格式工厂🡪音频🡪->WAV🡪添加文件🡪截取片段（无损音乐太大了，需要截取一小部分，20秒一首就可以了）🡪确定🡪输出配置（如下图）🡪确定🡪点击开始。



1. 打开截取WAV文件夹
2. 使用DataToHex将WAV文件下的WAV转换成音频数据数组文件，修改数组文件名Array[]，
3. 将wav音频数据文件改为c文件，请不要改为h，不然debug要好久。
4. 制作完成！

## 主要逻辑块

|  |
| --- |
| buffer**=**music1**;** //初始化buffer指向music1音乐数组地址    **while(**1**){**  **if(**flag**==**1**){** rIISCON **|=** 0x1**;** //如果点击播放(flag==1) 开启IIS song\_num1初值为1 播放第一首    //处理点击播放时音乐图片的显示(点击播放时 flag1=1)  **if(**flag1**==**1 **&&** song\_num**==**1 **){**flag1**=**0**,** Pait\_Bmp**(** 70**,** 70**,** 100**,** 100**,** music1\_bmp**);}**  **if(**flag1**==**1 **&&** song\_num**==**2 **){**flag1**=**0**,** Pait\_Bmp**(** 70**,** 70**,** 100**,** 100**,** music2\_bmp**);}**  **if(**flag1**==**1 **&&** song\_num**==**3 **){**flag1**=**0**,** Pait\_Bmp**(** 70**,** 70**,** 100**,** 100**,** music3\_bmp**);}**    // 音乐标志song\_num1不为零时(歌曲播放完song\_num1赋值或者通过键值(上一首/下一首)给song\_num1赋值)  // buffer音乐播放地址赋初值 length音乐长度重新赋值 并显示音乐图片 下一FIFO字节位置count初始化 song\_num1归零  **if(**song\_num1**==**1**)** **{** buffer**=**music1**;** length**=**3704572**;** count**=**0**;** Pait\_Bmp**(** 70**,** 70**,** 100**,** 100**,** music1\_bmp**);** song\_num1**=**0**;}**  **if(**song\_num1**==**2**)** **{** buffer**=**music2**;** length**=**3704552**;** count**=**0**;** Pait\_Bmp**(** 70**,** 70**,** 100**,** 100**,** music2\_bmp**);** song\_num1**=**0**;}**  **if(**song\_num1**==**3**)** **{** buffer**=**music3**;** length**=**5644880**;** count**=**0**;** Pait\_Bmp**(** 70**,** 70**,** 100**,** 100**,** music3\_bmp**);** song\_num1**=**0**;}**    **if((**rIISCON **&** **(**1**<<**7**))==**0**)** //检查输出FIFO是否为空  **{** //FIFO中的数据为16位，深度为32  //当输出FIFO为空时，一次性向FIFO写入32个16位数据  **for(**i**=**0**;**i**<**32**;**i**++){**  //rIISFIFO=(buffer[i+count]);  rIISFIFO**=(**buffer**[**2**\***i**+**count**])+(**buffer**[**2**\***i**+**1**+**count**]<<**8**);** // 一次循环向FIFO存储16位数据于FIFO  **}**    count**+=**64**;**  // 64为32次循环，每次循环指向两个不同字节的字节总和    //音乐播放完成后 音乐标志song\_num1指向下一首歌  **if(**count**>**length**){**  song\_num2**=**song\_num2**+**1**;**  **if(**song\_num2**==**4**){**song\_num2**=**1**;}**  song\_num1**=**song\_num2**;**  **}**  **}**    **}**    //暂停时关闭IIS 显示欢迎图片  **if(**flag**==**0**)** **{** rIISCON **|=** 0x0**;** Pait\_Bmp**(** 70**,** 70**,** 100**,** 100**,** hellomusic**);** **}** //关闭IIS;  **}** |

## 主要代码

Main.c

|  |
| --- |
| #define GLOBAL\_CLK 1  #include "def.h"  #include "option.h"  #include "2440addr.h"  #include "2440lib.h" //函数声明  #include "2440slib.h"  #include "mmu.h"  #include "profile.h"  //功能函数声明  extern void music\_player**(**void**);**    void Main**(**void**)**  **{**  U32 mpll\_val **=** 0**,**consoleNum**;**  Port\_Init**();**    mpll\_val **=** **(**92**<<**12**)|(**1**<<**4**)|(**1**);**    //init FCLK=400M,  ChangeMPllValue**((**mpll\_val**>>**12**)&**0xff**,** **(**mpll\_val**>>**4**)&**0x3f**,** mpll\_val**&**3**);**  ChangeClockDivider**(**14**,** 12**);** //the result of rCLKDIVN [0:1:0:1] 3-0 bit  cal\_cpu\_bus\_clk**();** //HCLK=100M PCLK=50M    consoleNum **=** 0**;** // Uart 0 select for debug.  Uart\_Init**(** 0**,**115200 **);**  Uart\_Select**(** consoleNum **);**    Port\_Init**();**  MMU\_Init**();** //地址映射初始化    Beep**(**2000**,** 100**);**  music\_player**();**  **}** |

music\_player.c

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  实现功能 music播放器  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include "2440lib.h"  #include "2440slib.h"  #include "LCD\_init.h"  #include "2440addr.h"  #define L3C (1<<4) //gpb4:L3CLOCK  #define L3D (1<<3) //gpb3:L3DATA  #define L3M (1<<2) //gpb2:L3MODE  #define rIISFIFO (\*(volatile unsigned long\*)0x55000010)  extern unsigned char music\_interface**[];**  extern unsigned char hellomusic**[];** //hellomusic  extern unsigned char music1\_bmp**[];**  extern unsigned char music2\_bmp**[];**  extern unsigned char music3\_bmp**[];**  extern unsigned char button1**[];** //暂停  extern unsigned char button1\_1**[];**  extern unsigned char button2**[];** //下一首  extern unsigned char button2\_2**[];**  extern unsigned char button3**[];** //上一首  extern unsigned char button3\_3**[];**  extern unsigned char button4**[];** //播放  extern unsigned char button4\_4**[];**  extern unsigned char music1**[**3704572**];**  extern unsigned char music2**[**3704552**];**  extern unsigned char music3**[**5644880**];**  //extern unsigned char WindowsXP\_Wav[243552];  //xdata, ydata用于存储屏幕坐标(不是屏幕像素点阵)  volatile static int xdata**,** ydata**;** //volatile的作用；作为指令关键字，确保本条指令不会因编译器的优化而省略，且要求每次直接读值.  volatile static int botten1**=**0**,** flag**=**0**,** flag1**,** song\_num**=**1**,** song\_num1**=**1**;**  volatile static int length**,** count**,** song\_num2**=**1**,** i**;**  volatile unsigned char **\***buffer**;**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  键值处理函数  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  static void Button\_Handle**(**void**){**      **if(** xdata **>=** 160 **&&** xdata **<=** 360 **&&** ydata **>=** 280 **&&** ydata **<** 420 **)** //上一首并播放  **{** Beep**(**2000**,** 100**);** botten1 **=** 1**,** song\_num**--;** flag**=**1**;;**//xdata = ydata = 0;  **if(**song\_num**==**0**)** **{** song\_num**=**3**;}** song\_num1**=**song\_num**;**  Pait\_Bmp**(** 20**,** 190**,** 60**,** 60**,** button2\_2**);** Delay**(**200**);** Pait\_Bmp**(** 90**,** 190**,** 60**,** 60**,** button1**);** Pait\_Bmp**(** 20**,** 190**,** 60**,** 60**,** button2**);** **}**    **if(** xdata **>=** 660 **&&** xdata **<=** 860 **&&** ydata **>=** 280 **&&** ydata **<** 420 **)** //下一首并播放  **{** Beep**(**2000**,** 100**);** botten1 **=** 2**,** song\_num**++;** flag**=**1**;**//xdata = ydata = 0;  **if(**song\_num**==**4**)** **{** song\_num**=**1**;}** song\_num1**=**song\_num**;**  Pait\_Bmp**(**160**,** 190**,** 60**,** 60**,** button3\_3**);** Delay**(**200**);** Pait\_Bmp**(** 90**,** 190**,** 60**,** 60**,** button1**);** Pait\_Bmp**(**160**,** 190**,** 60**,** 60**,** button3**);** **}**    **if(** xdata **>=** 410 **&&** xdata **<=** 610 **&&** ydata **>=** 280 **&&** ydata **<** 420 **)** //播放  **{** Beep**(**2000**,** 100**);** flag**++;** //xdata = ydata = 0;  //下一段很凑巧，当点击(上一首/下一首) flag赋值为1时，下面这段也会执行  //我想应该是 ydata >= 280 && ydata < 420 相与为1后， xdata >= 410 && xdata <= 610 && ydata >= 280 && ydata < 420的值也为1  //刚好实现了我的功能  **if(**flag**==**1**){**botten1 **=** 0**;** flag1**=**flag**;**  Pait\_Bmp**(** 90**,** 190**,** 60**,** 60**,** button4\_4**);** Delay**(**200**);** Pait\_Bmp**(** 90**,** 190**,** 60**,** 60**,** button1**);}** //为1时播放，显示暂停按钮    **else** **{**flag**=**0**;** botten1 **=** 0**;**  Pait\_Bmp**(** 90**,** 190**,** 60**,** 60**,** button1\_1**);** Delay**(**200**);** Pait\_Bmp**(** 90**,** 190**,** 60**,** 60**,** button4**);}** //为0时暂停，显示播放按钮  **}**  **}**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  TFT LCD \*触摸屏中断函数\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    static void \_\_irq Adc\_Tc\_Handler**(**void**)**  **{**    rADCTSC**|=(**1**<<**3**)|(**1**<<**2**);** //XP上拉电阻无效, 自动连续测量X坐标和Y坐标.  rADCCON**|=(**1**<<**0**);**//ADC转换开始    **while(**rADCCON**&(**1**<<**0**));**//检测ADC转换是否开始且ADCCON[0]自动清0  **while(!(**rADCCON**&(**0x1**<<**15**)));** //检测ADCCON[15]是否为1,ADC转换是否结束,(必须)  **while(!(**rINTPND**&((**U32**)**0x1**<<**31**)));**//检测ADC中断是否已请求    xdata**=**rADCDAT0**&**0x3ff**;**//读x坐标 >>xdata并不是像素点，而是模拟信号 0-1000  ydata**=**rADCDAT1**&**0x3ff**;**//读y坐标  //Beep(2000, 100);  Button\_Handle**();** //button处理函数  Uart\_Printf**(**"\n Xdata=%04d, Ydata=%04d\n"**,** xdata**,** ydata**);**  Uart\_Printf**(**"\n flag=%d, 1:播放 0：暂停"**,** flag**);**  Uart\_Printf**(**"\n botten1=%d, 1:上一首 2：下一首"**,**botten1**);**  Uart\_Printf**(**"\n \*\*正在播放第%d首歌曲\*\* \n\n"**,** song\_num**);**          rSUBSRCPND**|=(**0x1**<<**9**);** //清除中断  rSRCPND**|=((**U32**)**0x1**<<**31**);**  rINTPND**|=((**U32**)**0x1**<<**31**);**    rADCTSC **=**0xd3**;** //ADC等待中断模式  rADCTSC**|=(**0x1**<<**8**);** //ADCTSC[8]=1,设置抬起中断信号    **while(!(**rSUBSRCPND**&(**0x1**<<**9**)));** //检测触屏抬起中断是否已请求    rADCTSC **&=~(**0x1**<<**8**);**//ADCTSC[8]=0光标按下中断信号    // 由于下面这段代码和上面这段代码是看到大神们都会加上去的，调试得到  // 结论：上面的清除中断实现触屏中断，而下面这段则是加快中断后的响应  // (我的实验结果是：没下面的时候，触屏后的数值显示时间变长，蜂鸣器的  // 响声也变长，有种慢一拍的感觉)    rSUBSRCPND**|=(**0x1**<<**9**);**  rSRCPND**|=((**U32**)**0x1**<<**31**);**  rINTPND**|=((**U32**)**0x1**<<**31**);**  **}**    static void Touch\_Init**(**void**)**  **{**  rADCCON**=((**1**<<**14**)|(**9**<<**6**));** //A/D分频时钟有效，其值为9  rADCTSC**=**0xd3**;** //光标按下中断信号,YM有效，YP无效，XM有效，XP无效，XP上拉电阻，普通ADC转换，等待中断模式  rADCDLY**=**50000**;** //正常转换模式转换延时大约为(1/3.6864M)\*50000=13.56ms    rINTSUBMSK **&=~(**0x1**<<**9**);**//TC中断使能  rINTMSK **&=~((**U32**)**0x1**<<**31**);**//ADC总中断使能    pISR\_ADC**=(**U32**)**Adc\_Tc\_Handler**;**//指向中断向量表    **}**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  IIS初始化  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void iis\_init**(**void**){**  //配置IIS接口  rGPEUP **=** rGPEUP **&** **~(**0x1f**)** **|** 0x1f**;** //上拉无效，GPE[4:0] 1 1111  rGPECON **=** rGPECON **&** **~(**0x3ff**)** **|** 0x2aa**;**    //配置s3c2440的IIS寄存器  //预分频器为2，所以CDCLK=PCLK/(2+1)=16.66666kHz  rIISPSR **=** 2**<<**5**|**2**;**  //无效DMA，输入空闲，预分频器有效，开启IIS  rIISCON **=** **(**0**<<**5**)|(**0**<<**4**)|(**0**<<**3**)|(**1**<<**2**)|(**1**<<**1**);**  //PCLK为时钟源，输出模式，IIS模式，每个声道16位，CODECLK=384fs，SCLK=32fs  rIISMOD **=** **(**0**<<**9**)|(**0**<<**8**)|(**2**<<**6**)|(**0**<<**5**)|(**0**<<**4**)|(**1**<<**3**)|(**1**<<**2**)|(**1**<<**0**);**  rIISFCON **=** **(**0**<<**15**)|(**1**<<**13**);** //输出FIFO正常模式，输出FIFO使能      **}**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  UA1341函数  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  //通过io口模拟L3总线写数据  //mode：1为地址模式，0为数据模式  //关于地址模式和数据模式以及传输时序注意参考数据手册  static void write\_UA1341**(**U8 data**,** U8 address**)**  **{**  int i**,**j**;**  **if(**address **==** 1**)**  **{**  rGPBDAT **=** rGPBDAT**&(~(**L3D **|** L3M **|**L3C**))** **|**L3C**;** //地址模式，根据手册L3M为LOW,L3C为high    **}** **else** **{**    rGPBDAT **=** rGPBDAT **&** **(~(**L3D **|**L3M **|**L3C**))** **|(**L3M**|**L3C**);** //数据模式 L3M为高    **}**  Delay**(**1**);**  //传输数据  **for(**i**=**0**;**i**<**8**;**i**++)**  **{**  **if(**data **&** 0x1**)** // H  **{**  rGPBDAT **&=** **~**L3C**;** //L3C=L  rGPBDAT **|=** L3D**;** //L3D=H  **for(**j**=**0**;**j**<**5**;**j**++)**  **;** //等待一段时间  rGPBDAT **|=** L3C**;** //L3C=H  rGPBDAT **|=** L3D**;** //L3D=H  **for(**j**=**0**;**j**<**5**;**j**++)**  **;** //等待一段时间  **}**  **else** // L  **{**  rGPBDAT **&=** **~**L3C**;** //L3C=L  rGPBDAT **&=** **~**L3D**;** //L3D=L  **for(**j**=**0**;**j**<**5**;**j**++)**  **;** //等待一段时间  rGPBDAT **|=** L3C**;** //L3C=H  rGPBDAT **&=** **~**L3D**;** //L3D=L  **for(**j**=**0**;**j**<**5**;**j**++)**  **;** //等待一段时间  **}**  data **>>=** 1**;**  **}**  rGPBDAT **=** rGPBDAT **&** **~(**L3D **|** L3M **|** L3C**)** **|** **(**L3C **|** L3M**);** //L3M=H,L3C=H    **}**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  UDA1341初始化  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void UDA1341\_init**(**void**){**  //UDA1341初始化  //配置L3接口总线，GPB2:L3MODE, GPB3:L3DATA, GPB4:L3CLOCK  rGPBCON **=** 0x015550**;** //输出  rGPBUP **=** 0x7ff**;** //上拉无效  rGPBDAT **=** 0x1e4**;**    rGPBDAT **=** rGPBDAT **&** **(~(**L3M **|**L3C **|**L3D**))** **|(**L3M**|**L3C**);** //将L3CLOCK和L3MODE置高，准备开始传输  ////根据UDA1341TS数据手册14页中的操作顺序，首先在地址模式下，  //选择操作地址000101xx +10(STATUS)=0X16  write\_UA1341**(**0x16**,**1**)** **;**  write\_UA1341**(**0x60**,**0**);** // 0，1 ，10，000，0 复位    write\_UA1341**(**0x16**,**1**)** **;**  write\_UA1341**(**0x10**,**0**);** //0,0,01, 000,0 : 状态0, 384fs,IIS,no DC-filtering    write\_UA1341**(**0x16**,**1**)** **;**  write\_UA1341**(**0xc1**,**0**);** //1,0,0,0, 0,0,01:状态1,  //Gain of DAC 6 dB,Gain of ADC 0dB,ADC non-inverting,  //DAC non-inverting,Single speed playback,ADC-Off DAC-On  **}**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  music \*子main函数\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    void music\_player**(**void**)**  **{**      Lcd\_Port\_Init**();** //端口初始化  LCD\_Init**();** //TFT LCD功能模块初始化  Lcd\_PowerEnable**(**0**,** 1**);** // TFT LCD 电源控制引脚使能  Lcd\_EnvidOnOff**(**1**);** //LCD视频和控制信号输出或者停止，1开启视频输出        /\*红(255:0:0);绿(0:255:0);蓝(0:0:255);黑(0:0:0);白(255,255,255)\*/    /\*在屏幕上显示三基色\*/    Lcd\_ClearScr**((**0x1f**<<**11**)** **|** **(**0x00**<<**5**)** **|** **(**0x00**));** //red  Delay**(**500**);**    Lcd\_ClearScr**((**0x00**<<**11**)** **|** **(**0x3f**<<**5**)** **|** **(**0x00**));** //green  Delay**(**500**);**    Lcd\_ClearScr**((**0x00**<<**11**)** **|** **(**0x00**<<**5**)** **|** **(**0x1f**));** //blue  Delay**(**500**);**    Lcd\_ClearScr**(** **(**0x1f**<<**11**)** **|** **(**0x3f**<<**5**)** **|** **(**0x1f**)** **)** **;** //clear screen white  Delay**(**500**);**    Pait\_Bmp**(** 0**,** 0**,** 240**,** 320**,** music\_interface**);**  Pait\_Bmp**(** 70**,** 70**,** 100**,** 100**,** hellomusic**);**  Pait\_Bmp**(** 20**,** 190**,** 60**,** 60**,** button2**);** //上一首  Pait\_Bmp**(**160**,** 190**,** 60**,** 60**,** button3**);** //下一首  Pait\_Bmp**(** 90**,** 190**,** 60**,** 60**,** button4**);** //播放    Touch\_Init**();** //开触摸屏中断    iis\_init**();** // IIS初始化    UDA1341\_init**();** //UDA1341初始化      buffer**=**music1**;** //初始化buffer指向music1音乐数组地址    **while(**1**){**  **if(**flag**==**1**){** rIISCON **|=** 0x1**;** //如果点击播放(flag==1) 开启IIS song\_num1初值为1 播放第一首    //处理点击播放时音乐图片的显示(点击播放时 flag1=1)  **if(**flag1**==**1 **&&** song\_num**==**1 **){**flag1**=**0**,** Pait\_Bmp**(** 70**,** 70**,** 100**,** 100**,** music1\_bmp**);}**  **if(**flag1**==**1 **&&** song\_num**==**2 **){**flag1**=**0**,** Pait\_Bmp**(** 70**,** 70**,** 100**,** 100**,** music2\_bmp**);}**  **if(**flag1**==**1 **&&** song\_num**==**3 **){**flag1**=**0**,** Pait\_Bmp**(** 70**,** 70**,** 100**,** 100**,** music3\_bmp**);}**    // 音乐标志song\_num1不为零时(歌曲播放完song\_num1赋值或者通过键值(上一首/下一首)给song\_num1赋值)  // buffer音乐播放地址赋初值 length音乐长度重新赋值 并显示音乐图片 下一FIFO字节位置count初始化 song\_num1归零  **if(**song\_num1**==**1**)** **{** buffer**=**music1**;** length**=**3704572**;** count**=**0**;** Pait\_Bmp**(** 70**,** 70**,** 100**,** 100**,** music1\_bmp**);** song\_num1**=**0**;}**  **if(**song\_num1**==**2**)** **{** buffer**=**music2**;** length**=**3704552**;** count**=**0**;** Pait\_Bmp**(** 70**,** 70**,** 100**,** 100**,** music2\_bmp**);** song\_num1**=**0**;}**  **if(**song\_num1**==**3**)** **{** buffer**=**music3**;** length**=**5644880**;** count**=**0**;** Pait\_Bmp**(** 70**,** 70**,** 100**,** 100**,** music3\_bmp**);** song\_num1**=**0**;}**    **if((**rIISCON **&** **(**1**<<**7**))==**0**)** //检查输出FIFO是否为空  **{** //FIFO中的数据为16位，深度为32  //当输出FIFO为空时，一次性向FIFO写入32个16位数据  **for(**i**=**0**;**i**<**32**;**i**++){**  //rIISFIFO=(buffer[i+count]);  rIISFIFO**=(**buffer**[**2**\***i**+**count**])+(**buffer**[**2**\***i**+**1**+**count**]<<**8**);** // 一次循环向FIFO存储16位数据于FIFO  **}**    count**+=**64**;** // 64为32次循环，每次循环指向两个不同字节的字节总和    //音乐播放完成后 音乐标志song\_num1指向下一首歌  **if(**count**>**length**){**  song\_num2**=**song\_num2**+**1**;**  **if(**song\_num2**==**4**){**song\_num2**=**1**;}**  song\_num1**=**song\_num2**;**  **}**  **}**    **}**    //暂停时关闭IIS 显示欢迎图片  **if(**flag**==**0**)** **{** rIISCON **|=** 0x0**;** Pait\_Bmp**(** 70**,** 70**,** 100**,** 100**,** hellomusic**);** **}** //关闭IIS;    **}**    **}** |

## 附录

wav工具[下载地址](http://download.csdn.net/detail/muyang_ren/7764741)

视频[地址](http://v.youku.com/v_show/id_XNzU2NTExNjUy.html)