

启光电子——VS1053 音乐播放器

VS1053B 简介

VS1053B 是 **VS1003** 的升级版芯片，功能、解码强度和音质效果上都有了很大的提升。

VS1053b 是单片 **Ogg Vorbis/MP3/AAC/WMA/MIDI** 音频解码器，它包含一个高性能，自主产权的低功耗 **DSP** 处理器核 **VS_DSP**，工作数据存储器，为用户应用提供 **16KB** 的指令 **RAM** 和 **0.5KB** 的数据 **RAM**。具有 **8** 个可用的通用 **IO** 口和一个串口，芯片内部还带了一个可变采样率的立体声 **ADC**（支持咪头/咪头+线路/2 线路）、一个高性能立体声 **DAC** 及音频耳机放大器。

对于 **vs1003** 来说指令 **RAM** 只有 **5kb**，常规用途 **IO** 口只有 **4** 个。

VS1053 相对前者，其支持的音乐文件解码是不用程序判断是哪种文件后缀，而是自动识别的。而且 **VS1053** 支持 **EarSpeaker** 空间效果（**VS1003** 不支持）。同样的歌曲用 **VS1053** 播放出来的效果，确实要比 **VS1003** 好很多，有很多同学都感觉 **VS1003** 有杂音的问题，但是这样的问题在 **VS1053** 上已经不在有了，他的效果会让你感觉和你买的品牌 **MP3** 的效果没有什么区别。所以说用 **VS1053** 完全可以做一部属于自己的 **music player**！

VS1053 的特性如下：

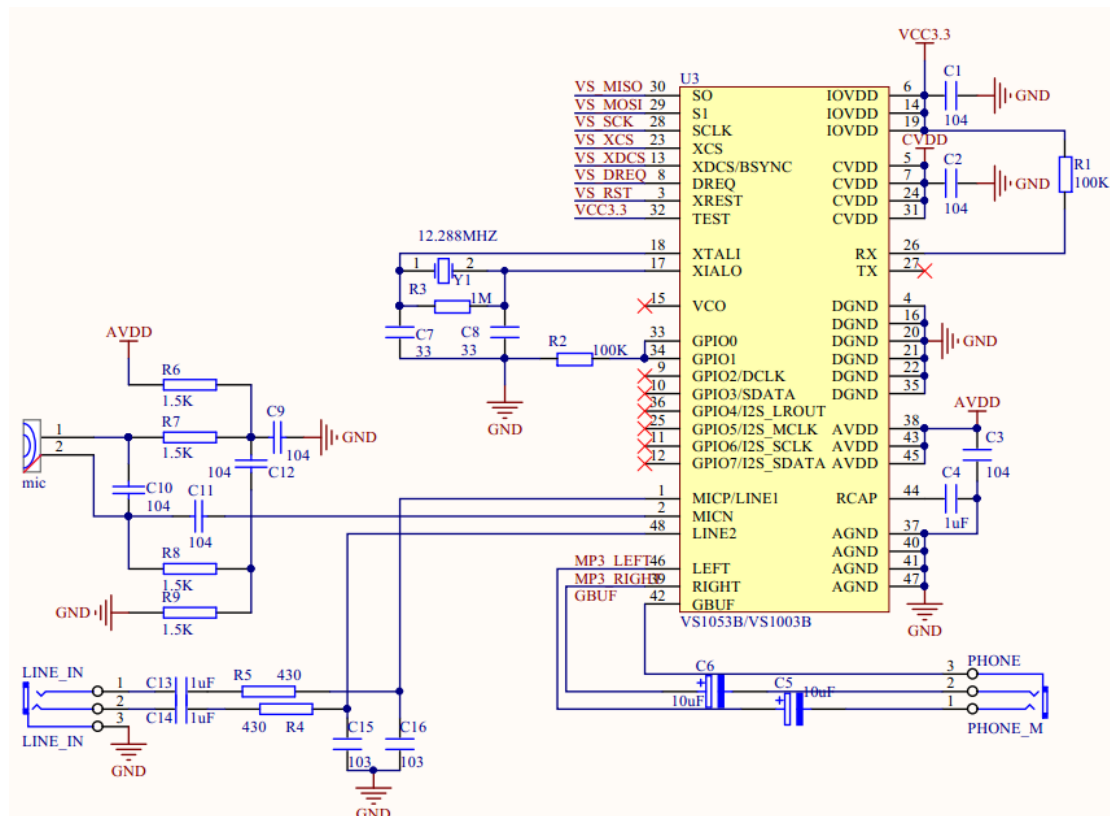
- 支持众多音频格式解码，包括 **OGG/MP3/WMA/WAV/FLAC**（需要加载 **patch**）/**MIDI/AAC** 等。
- 对话筒输入或线路输入的音频信号进行 **OGG**（需要加载 **patch**）/**IMA**

ADPCM 编码

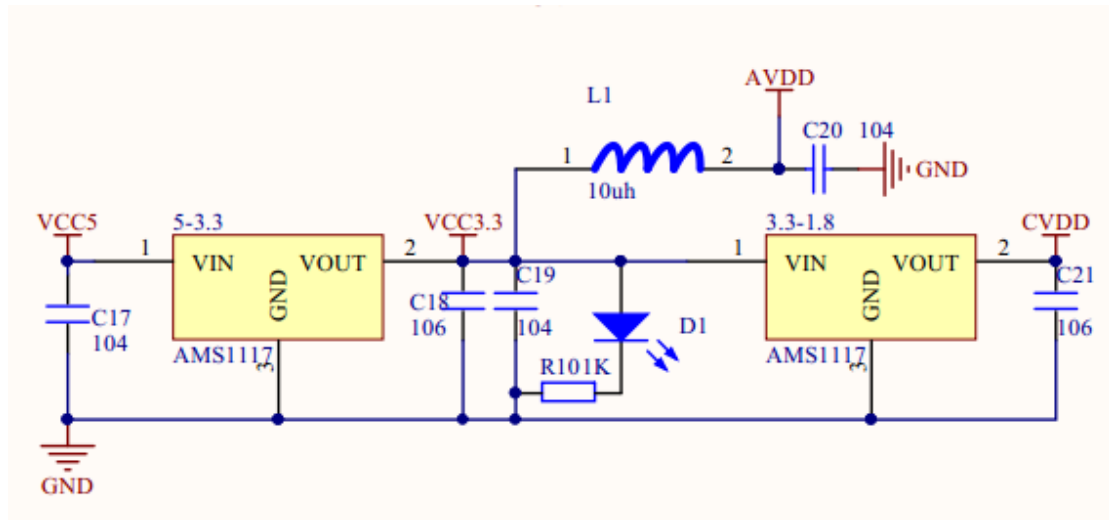
- 高低音控制
- 带有 **EarSpeaker** 空间效果（用耳机虚拟现场空间效果）
- 单时钟操作 **12..13MHz**
- 内部 **PLL** 锁相环时钟倍频器
- 低功耗
- 内含高性能片上立体声 **DAC**，两声道间无相位差
- 过零交差侦测和平滑的音量调整
- 内含能驱动 **30** 欧负载的耳机驱动器
- 模拟，数字，**I/O** 单独供电
- 为用户代码和数据准备的 **16KB** 片上 **RAM**
- 可扩展外部 **DAC** 的 **I2S** 接口
- 用于控制和数据的串行接口（**SPI**）
- 可被用作微处理器的从机
- 特殊应用的 **SPI Flash** 引导
- 供调试用途的 **UART** 接口
- 新功能可以通过软件和 **8 GPIO** 添加

原理图

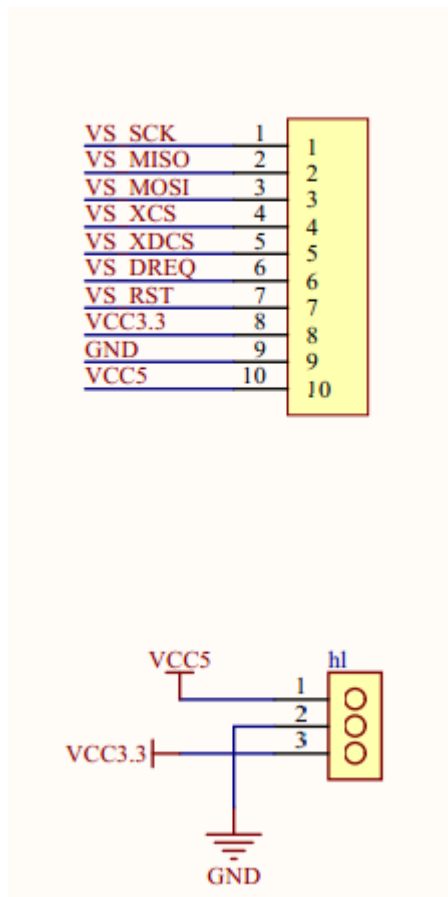
主控制部分



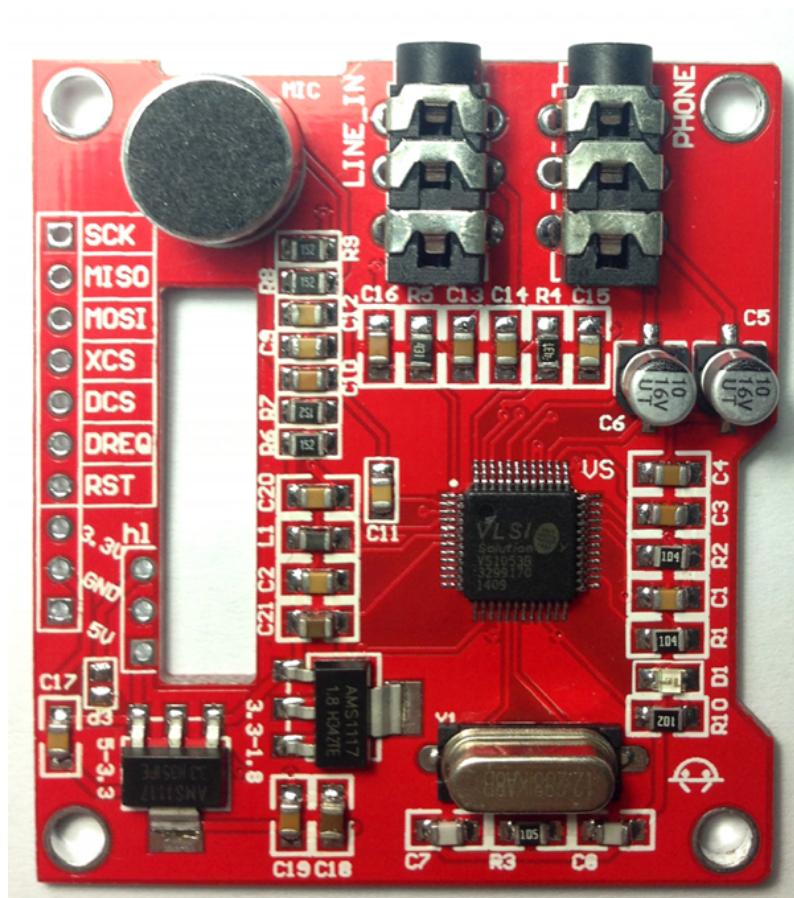
电源部分



插口部分



模块实物图



主要程序流程

我们的程序主要是在 51 单片机上开发的，其实不管是 STM32 还是 51 或者其他单片机，他们的程序流程都是一样的。

主要步骤如下：

挂载文件系统（我们用的文件系统是 Petit Fatfs）只有挂载系统后才能打开 tf 卡上的音乐文件、

初始化 VS1053 芯片（包括硬件初始化和软件初始化，并且进行一些值的测试）、

读取指定音乐文件的数据到缓存数组中、

通过判断方式将读出的数据流以每 32 个字节一次的方式丢

入 VS1053 中，这时 VS1053 音频输出口就会有音乐播出。

总线介绍：

VS1053 主要用的的有 7 根数据线，分别是：VS_MISO 、VS_MOSI、VS_SCK、VS_XCS 、VS_XDCS 、VS_DREQ 和 VS_RST 。

VS_MISO 、VS_MOSI、VS_SCK、为 SPI 总线，通过 VS_XCS 、VS_XDCS 的不同片选脚，来控制总线的不同操作。

VS_RST 是 VS1053 的复位信号线，低电平有效。

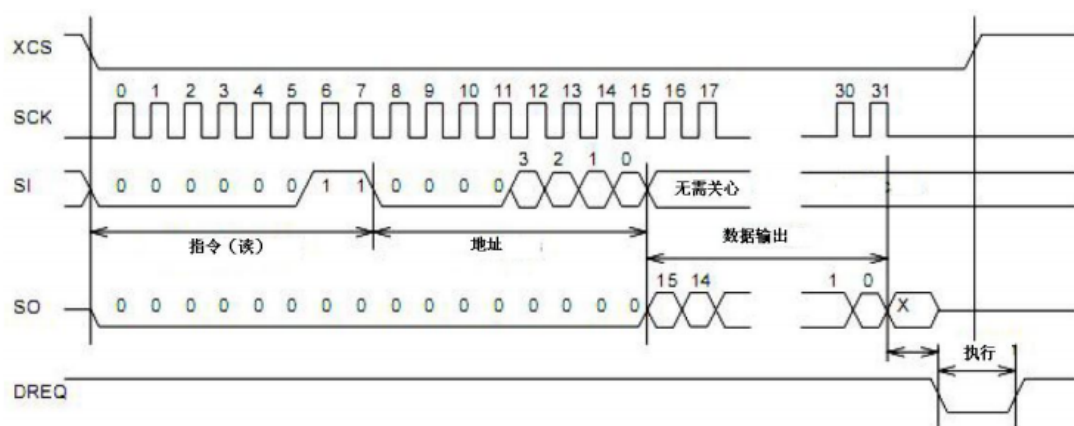
VS_DREQ 是一个数据请求信号，用来通知主机，VS1053 可以接收数据与否。（也就是是否可以发送音乐数据的判断口）

VS1053 支持两种 SPI 模式：1，VS1002 本地模式（新模式）2，VS1001 兼容模式（不推荐），这里只介绍本地模式。SPI 信号描述如图：

SDI 引脚	SCI 引脚	说明
XDCS	XCS	低有效的片选信号。高电平将强制串行接口结束当前操作，并进入备用模式，它将强行使串行输出进入高阻状态。如果 SM_SDISHARE 是 1，则 XDCS 引脚是不使用的，但这个信号是通过将 XCS 反向来获得的。
SCK		串行时钟输入。此时钟也是用作内部寄存器接口的主时钟。SCK 电平在平时可以是脉冲状或平静的。不管在何种情况之下，只要在 XCS 信号变低之后，首个时钟上升沿将被定义为首个位。
SI		串行输入。如果片选有效，SI 是在时钟 SCK 的上升沿上取样的。
-	SO	串行输出。在读取时，数据是逐位移动输出在时钟 SCK 的下降沿上。而在写入时，它是处于高阻态的。

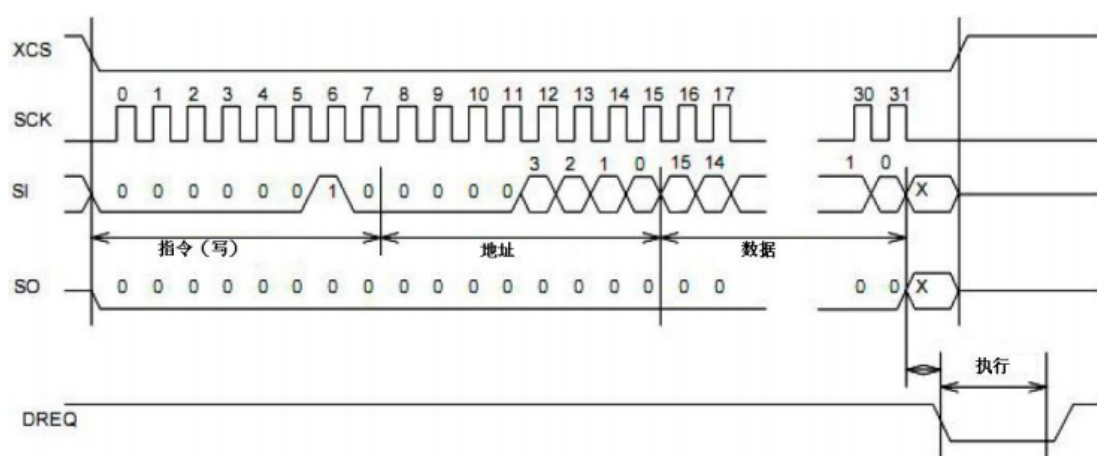
从上表的 SPI 数据传送上看，SDI 是发送数据，SCI 是发送命令。SDI 是在主机判断 DREQ 有效（高电平）时，才控制 SPI 总线向 VS1053 中写入数据（一次可发送 32 个字节数据）。

SCI 读时序：



SCI 串行总线命令接口包含了一个指令字节、一个地址字节和一个 16 位的数据字，并且字节数据总是高位在前低位在后的。第一个字节指令字节，只有 2 个指令，也就是读和写，读为 0X03，写为 0X02。

SCI 写时序：



这个就是 SCI 先发送写指令 0x02，然后是一个地址，通过 SI 写入 VS1053，SO 一直保持低电平。

读和写时序都有在最后 DREQ 有一个短暂的 0 过程，这是 VS1053 的一个处理过程。这部分是不允许被打断的，所以在 SCI 操作之前最好进行 DREQ 的判断。

SCI 寄存器：

SCI 有 16 个寄存器，如下表：

SCI 寄存器				
寄存器	类型	复位值	缩写	描述
0X00	RW	0X0800	MODE	模式控制
0X01	RW	0X000C	STATUS	VS0153 状态
0X02	RW	0X0000	BASS	内置低音/高音控制
0X03	RW	0X0000	CLOCKF	时钟频率+倍频数
0X04	RW	0X0000	DECODE_TIME	解码时间长度（秒）
0X05	RW	0X0000	AUDATA	各种音频数据
0X06	RW	0X0000	WRAM	RAM 写/读
0X07	RW	0X0000	WRAMADDR	RAM 写/读的基址
0X08	R	0X0000	HDATA0	流的数据标头 0
0X09	R	0X0000	HDATA1	流的数据标头 1
0X0A	RW	0X0000	AIADDR	应用程序起始地址
0X0B	RW	0X0000	VOL	音量控制
0X0C	RW	0X0000	AICTRL0	应用控制寄存器 0
0X0D	RW	0X0000	AICTRL1	应用控制寄存器 1
0X0E	RW	0X0000	AICTRL2	应用控制寄存器 2
0X0F	RW	0X0000	AICTRL3	应用控制寄存器 3

下面介绍几个重要的寄存器：

MODE 寄存器用于控制 VS1053 的操作，是最关键的寄存器之一，该寄存器的复位值为 0x0800，其实就是默认设置为新模式。

位元	名称	功能	值	说明
0	SM_DIFF	差分	0	正常的同相音频
			1	左通道反相
1	SM_LAYER12	允许 MPEG layers I & II	0	不允许
			1	允许
2	SM_RESET	软件复位	0	不用复位
			1	复位
3	SM_CANCEL	取消当前的文件解码	0	不取消
			1	取消
4	SM_EARSPEAKER_LO	EarSpeaker 低设定	0	关闭
			1	激活
5	SM_TESTS	允许 SDI 测试	0	不允许
			1	允许
6	SM_STREAM	流模式	0	不是
			1	是
7	SM_EARSPEAKER_HI	EarSpeaker 高设定	0	关闭
			1	激活
8	SM_DACT	DCLK 的有效边沿	0	上升沿
			1	下降沿
9	SM_SDIORD	SDI 位顺序	0	MSb 在前
			1	MSb 在后
10	SM_SDISHARE	共享 SPI 片选	0	不共享
			1	共享
11	SM_SDINEW	VS1002 本地 SPI 模式	0	非本地模式
			1	本地模式
12	SM_ADPCM	ADPCM 录音激活	0	不激活
			1	激活
13	-	-	0	正确的
			1	错误的
14	SM_LINE1	咪 / 线路1 选择	0	MICP
			1	LINE1
15	SM_CLK_RANGE	输入时钟范围	0	12..13 MHz
			1	24..26 MHz

主要设置第 2 位 软复位 和第 11 位 为本地模式，关于软复位，建议在每首歌播放之前执行一次。至于本地模式就不用解释了，是最新模式。

BASS 寄存器可以用于设置 VS1053 的高低音效。

名称	位域	说明
ST_AMPLITUDE	15:12	高音控制，步长为 1.5 dB (-8..7, 0 = 关闭)
ST_FREQLIMIT	11:8	下限频率，步长为 1000 Hz (1..15)
SB_AMPLITUDE	7:4	低音增强，步长为 1 dB (0..15, 0 = 关闭)
SB_FREQLIMIT	3:0	频率上限，步长为 10 Hz (2..15)

通过该寄存器的高低音调节，可以调节出我们喜欢的音效。它和 EarSpeaker 效果不同，EarSpeaker 是由 MODE 寄存器控制。

CLOCKF 寄存器用来设置时钟频率、倍频等相关信息。

CLOCKF 寄存器			
位	15:13	12:11	10:0
名称	SC_MULT	SC_ADD	SC_FREQ
描述	时钟倍频数	允许倍频	时钟频率
说明	$CLKI = XTALI \times (SC_MULT \times 0.5 + 1)$	倍频增量 $= SC_ADD \times 0.5$	当时钟频率不为 12.288M 时，外部时钟的频率。外部时钟为 12.288M 时，此部分设置为 0 即可

重点说明 SC_FREQ, SC_FREQ 是以 4KHz 为步进的一个时钟寄存器，当外部时钟不是 12.288M 的时候，

其计算公式为： $SC_FREQ = (XTALI - 8000000) / 4000$

式中 XTALI 的单位为 Hz。CLKI 是内部时钟频率，XTALI 是外部晶振的时钟频率。由于我们使用的是 12.288M 的晶振，在这里设置此寄存器的值为 0x9800，也就是设置内部时钟频率为输入时钟频率的 3 倍，倍频增量为 1.0 倍。

注：0x9 相当于 1001 15:13 就是 100 也就是 4，12:11 就是 10 也就是 2

DECODE_TIME 寄存器是一个存放解码时间的寄存器，

以秒钟为单位，通过读取该寄存器的值，就可以得到解码时间。不过它是一个累计时间，所以需要在每首歌播放之前把它清空一下，以得到这首歌的准确解码时间，当然如果歌曲有进度跳转时，它就不能跟踪时间了，所以后期准确的当前时间来源，还是以计算码率为主。

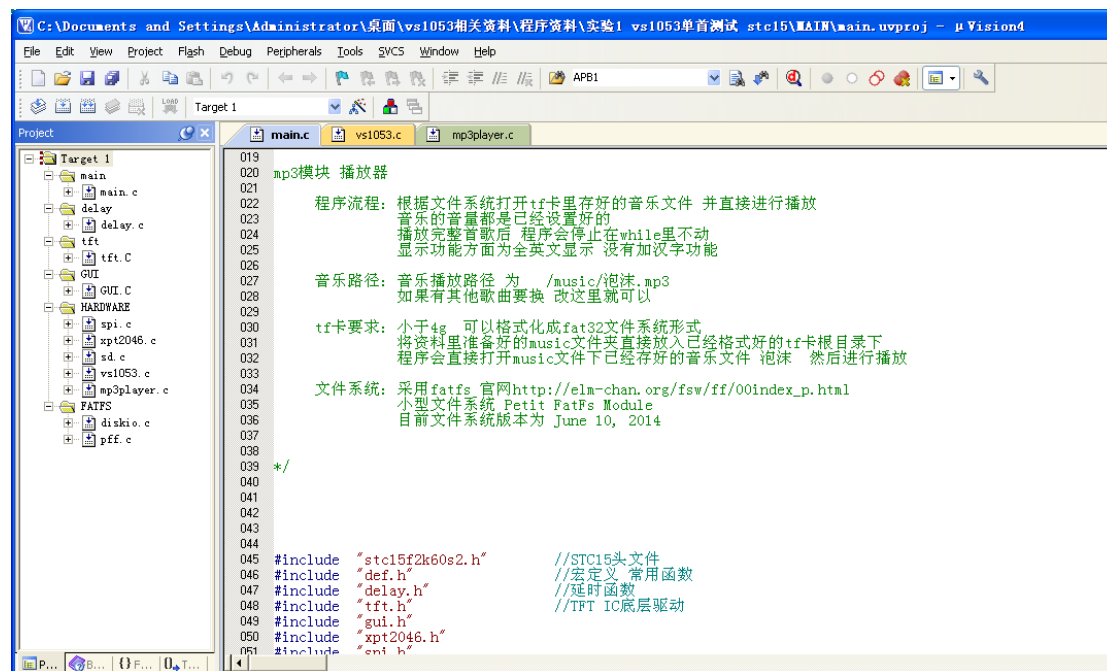
HDATA0 和 **HDATA1** 是两个数据流头寄存器，不同的音频文件，读出来的值意义不一样，我们可以通过这两个寄存器来获取音频文件的码率，从而可以计算音频文件的总长度。这两个寄存器的详细介绍，请参考 **VS1053** 的数据手册。

最后 **VOL** 寄存器用于控制 **VS1053** 的输出音量，该寄存器可以分别控制左右声道的音量，每个声道的控制范围为 **0~254**，每个增量代表 **0.5db** 的衰减，所以该值越小，代表音量越大。比如设置为 **0X0000** 则音量最大，而设置为 **0XFEFE** 则音量最小。

注意：如果设置 **VOL** 的值为 **0XFFFF**，将使芯片进入掉电模式！

初级程序讲解

程序讲解以 **实验1 vs1053单首测试 stc15** 为例。



```
019 mp3模块 播放器
020
021 程序流程: 根据文件系统打开tf卡里存好的音乐文件 并直接进行播放
022           音乐的音量都是已经设置好的
023           播放完整首歌后 程序会停止在while里不动
024           显示功能方面为全英文显示 没有加汉字功能
025
026 音乐路径: 音乐播放路径 为 /music/泡沫.mp3
027           如果有其他歌曲要换 改这里就可以
028
029 tf卡要求: 小于4g 可以格式化成fat32文件系统形式
030           将资料里准备好的music文件夹直接放入已经格式好的tf卡根目录下
031           程序会直接打开music文件夹下已经存好的音乐文件 泡沫 然后进行播放
032
033 文件系统: 采用fatfs 官网http://elm-chan.org/fsw/ff/00index_p.html
034           小型文件系统 Petit FatFs Module
035           目前文件系统版本为 June 10, 2014
036
037
038
039 */
040
041
042
043
044
045 #include "stc15f2k60s2.h" //STC15头文件
046 #include "def.h" //宏定义 常用函数
047 #include "delay.h" //延时函数
048 #include "tft.h" //TFT IC底层驱动
049 #include "gui.h"
050 #include "xpt2046.h"
051 #include "eni.h"
```

程序本身已经加载好了 **FATFS** 文件系统。



我们这里直接看 main 函数

```
31
32 void main()
33 {
34     FRESULT res;
35     u16 br;
36     u32 cnt=0; //音乐进度递增变量
37     u8 mp3in;
38
39     SP=0X80; //调整堆栈指向 手册286页 详解
40     Lcd_Init(); //tft初始化
41     Init_SPI(); //SPI初始化
42     pf_mount(&fatfs); //初始化petit FATFS文件系统 并提取tf卡相应数据
43     //这句非常重要,它是使用所有Petit Fatfs文件系
44
45     GUI_Clear(White); //白色清屏
46     GUI_sprintf_hzstr16x(0,0,"VS1053--TEST",Black,White);
47
48     mp3in=Mp3Player_Init();
49     if(mp3in==1)GUI_sprintf_hzstr16x(0,30,"MP3 Init OK",Red,White);
50     else GUI_sprintf_hzstr16x(0,30,"MP3 Init Error",Red,White);
51
52     res=pf_open("/music/泡沫.mp3"); //打开指定路径下的音乐文件名
53
54     if(res == FR_OK)GUI_sprintf_hzstr16x(0,80,"The music is already open!".E
```

主要步骤就是初始化 TFT

```
SP=0X80;
Lcd_Init();
Init_SPI();
pf_mount(&fat
```

初始化 SPI 总线

挂载文件系统

```
pf_mount(&fatfs);
```

然后下面就是对 VS1053 芯片的初始化 Mp3Player_Init();

可以看一下初始化的底层

```

4 // 说明:
5 // =====
6 u8 Mp3Player_Init(void)
7 {
8     u16 ret;
9
10    VS_HD_Reset();           // 硬复位           ok
11    ret = VS_Ram_Test();     // 存储器测试
12    if(ret != 0X83FF && ret != 0x807F) return 0;
13    VS_Set_Vol(120);        // 设置音量
14    VS_Sine_Test();         // 正弦波测试
15    VS_HD_Reset();         // 硬复位
16    VS_Soft_Reset();        // 软复位
17    return 1;
18 }

```

这是 VS1053 初始的一个正规的步骤，至于音量也可在后面设置。

整体的这个步骤也不用太深入研究，记住步骤流程，知道怎么调用其实就足够了！

有一点需要强调一下

在读写寄存器的时候 要注意降低 SPI 总线的速度

```

215
216 //读VS10XX的寄存器
217 //address: 寄存器地址
218 //返回值: 读到的值
219 //注意不要用倍速读取,会出错
220 u16 VS_RD_Reg(u8 address)
221 {
222     u16 temp=0;
223     while(VS_DQ==0); //非等待空闲状态
224     VS_SPI_SpeedLow(); //低速           SPI降速
225     VS_XDCS=1;
226     VS_XCS=0;
227     VS_SPI_ReadWriteByte(VS_READ_COMMAND); //发送VS10XX的读命令
228     VS_SPI_ReadWriteByte(address); //地址
229     temp=VS_SPI_ReadWriteByte(0xff); //读取高字节
230     temp=temp<<8;
231     temp+=VS_SPI_ReadWriteByte(0xff); //读取低字节
232     VS_XCS=1;
233     VS_SPI_SpeedHigh(); //高速
234     return temp;
235 }
236

```

```

6
7 //向VS10XX写命令
8 //address:命令地址
9 //dat:命令数据
0 void VS_WR_Cmd(u8 address,u16 dat)
1 {
2     while(VS_DQ==0); //等待空闲
3     VS_SPI_SpeedLow(); //低速
4     VS_XDCS=1;
5     VS_XCS=0;
6     VS_SPI_ReadWriteByte(VS_WRITE_COMMAND); //发送VS10XX的写命令
7     VS_SPI_ReadWriteByte(address); //地址
8     VS_SPI_ReadWriteByte(dat>>8); //发送高八位
9     VS_SPI_ReadWriteByte(dat); //第八位
0     VS_XCS=1;
1     VS_SPI_SpeedHigh(); //高速
2 }
3

```

主要目的是为了防止丢字节 或者通讯不成功 这里虽然程序都写好了，但是还是请大家一定要注意一下！

然后回到 main 函数

初始化 VS 以后，就要打开音乐文件

至于路径什么的，在存入 tf 卡 music 文件夹里就有讲了，这里不多说。

```

res=pf_open("/music/泡沫.mp3"); //打开指定路径下的音乐文件名
if(res == FR_OK)GUI_sprintf_hzstr16x(0,80,"The music is already open!",Blue,White);

```

打开成功后会有 TFT 显示效果

最后就是播放部分：

```

while(1)
{
    res=pf_read(tbuf,512,&br); //通过文件系统读取指定文件夹下的一音乐数据
    if((res!=FR_OK)
    {
        led=0; while(1);
    }
    cnt=0;
    do{
        if(VS_Send_MusicData(tbuf+cnt)==0) //一次送32个字节音乐数据
        {
            cnt+=32;
        }
        else {led=0;}
    }while(cnt<512);
}

if(br!=512) // 文件结束
{
    while(1);
}

```

先读取 512 字节的音乐文件数据到缓存数组 tbuf 中

```
res=pf_read(tbuf, 512, &br);
```

然后向 VS1053 中写入数据流，注意一次只能写入 32 个字节，写入以后不停的判断 DREQ 是否为 1，如果为 1 表示可以继续写入下 32 个字节。

不停的循环这个过程，如果 512 的数据发送没，就在读下 512 字节到缓存数组，然后继续丢数据流到 VS1053。这样的过程，就会在 VS1053 音频输出口播放已经打开的音乐文件——泡沫。

这样一个简单的播放过程就讲完了，多说一句后面做的音乐频谱、播放时间的更新等等、都是在丢 VS1053 32 个字节后，在等 DREQ 为高时更新的。这样做好了就不会影响到音乐的正常播放，还可以加很多效果！

启光电子——叶木夕阳