# AN1509 ATK-VS1053 MP3 模块使用说明

本应用文档（AN1509，对应 **ALIENTEK 精英 STM32F103 开发板 扩展实验 E1**）将教大家如何在 ALIENTEK 精英 STM32 开发板上使用 ATK-VS1053 MP3 模块。本文档我们将使用 ATK-VS1053 MP3 模块实现音频播放，设计一个属于你自己的 MP3！

本文档分为如下几部分：

1， ATK-VS1053 MP3 模块简介
2， 硬件连接
3， 软件实现
4， 验证

## 1、ATK-VS1053 MP3 模块简介

ATK-VS1053 MP3 MODULE 是 ALIENTEK 推出的一款高性能音频编解码模块，该模块采用 VS1053B 作为主芯片，支持：MP3/WMA/OGG/WAV/FLAC/MIDI/AAC 等音频格式的解码，并支持：OGG/WAV 音频格式的录音，支持高低音调节以及 EarSpeaker 空间效果设置，功能十分强大。

### 1.1 模块资源简介

ATK-VS1053 MP3 模块是 ALIENTEK 开发的一款高性能音频编解码模块，该模块接口丰富、功能完善，仅需提供电源（3.3V/5.0V），即可通过单片机（8/16/32 位单片机均可）控制模块实现音乐播放，或者录音等功能，模块其资源图如图 1.1.1 所示：

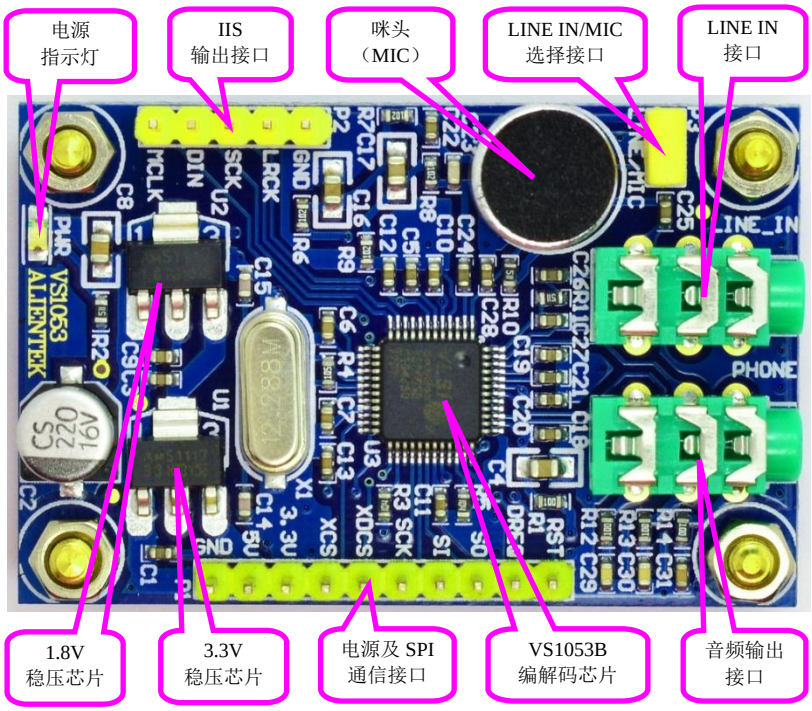


图 1.1.1 ATK-VS1053 MP3 模块资源图

从图 1.1.1 可以看出，ATK-VS1053 MP3 模块不但外观漂亮，而且功能齐全、接口丰富，

模块尺寸为 34mm*52.6mm，并带有安装孔位，非常小巧，并且利于安装，可方便应用于各种设计。

ALIENTEK ATK-VS1053 MP3 模块板载资源如下：

- 高性能编解码芯片：VS1053B
- 1 个 LINE IN/MIC 选择接口
- 1 个咪头
- 1 个电源指示灯（蓝色）
- 1 个 1.8V 稳压芯片
- 1 个 3.3V 稳压芯片
- 1 路 IIS 输出接口
- 1 路电源及 SPI 控制接口
- 1 路 3.5mm LINE IN 接口，支持双声道输入录音
- 1 路 3.5mm 音频输出接口，可直接插耳机

ATK-VS1053 模块采用高准设计，特点包括：

- 板载 VS1053B 高性能编解码芯片，支持众多音频格式解码，支持 OGG/WAV 编码。
- 板载稳压电路，仅需外部提供一路 3.3V 或 5V 供电即可正常工作；
- 板载 3.5mm 耳机插口，可直接插入耳机欣赏高品质音乐；
- 板载咪头（MIC），无需外部麦克风，即可实现录音；
- 板载 IIS 输出，可以接外部 DAC，获得更高音质；
- 板载电源指示灯，上电状态一目了然；
- 采用国际 A 级 PCB 料，沉金工艺加工，稳定可靠；
- 采用全新元器件加工，纯铜镀金排针，坚固耐用；
- 人性化设计，各个接口都有丝印标注，使用起来一目了然；接口位置设计安排合理，方便顺手。
- PCB 尺寸为 34mm*52.6mm，并带有安装孔位，小巧精致；

ATK-VS1053 MP3 模块的资源介绍，我们就介绍到这里，详细的介绍，请看《ATK-VS1053 MP3 模块用户手册》相关章节。

## 1.2 模块使用

模块通过SPI接口来接受输入的音频数据流，它可以是一个系统的从机，也可以作为独立的主机。这里我们只把它当成从机使用。我们通过SPI口向VS1053不停的输入音频数据，它就会自动帮我解码了，然后从输出通道输出音乐，这时我们接上耳机就能听到所播放的歌曲了。

模块（VS1053）通过7根信号线同主控芯片连接，分别是：XCS、XDCS、SCK、SI、SO、DREQ、和RST。其中RST是VS1053的复位信号线，低电平有效。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。SCK、SI(MOSI)和SO(MISO)则是VS1053的SPI接口，他们在XCS和XDCS的控制下面来执行不同的数据通信。另外，模块需要外部提供5V/3.3V供电，推荐采用5V供电，这样，总共需要9根线来连接。

关于VS1053 SPI通信的详细介绍，请参考《ATK-VS1053 MP3模块用户手册》2.3节。这里我们不做详细阐述了。

ATK-VS1053 MP3 模块的原理图如图 1.2.1 所示：

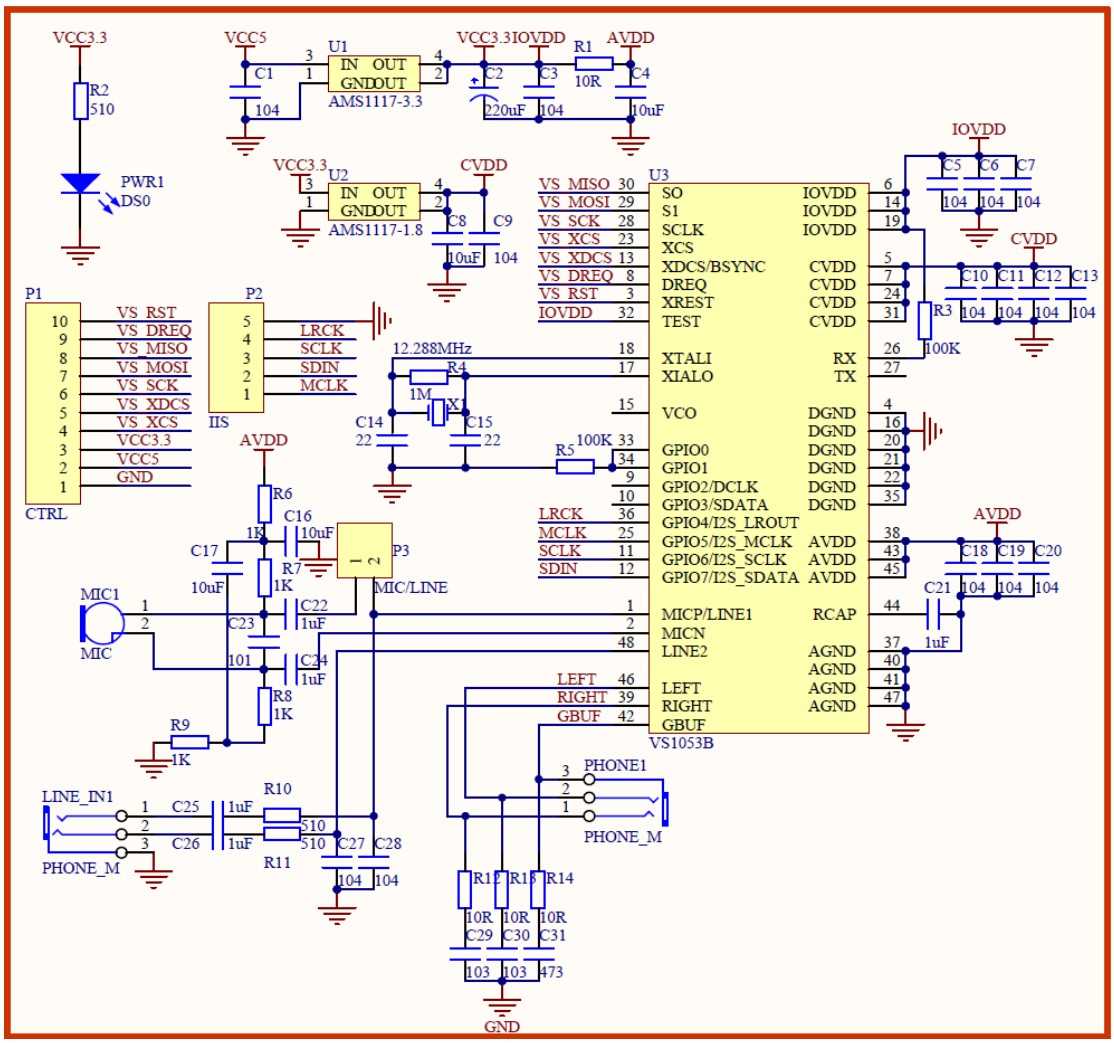


图1.2.1 ATK-VS1053 MP3模块原理图

模块通过 P1 接口与外部单片机系统连接，P1 引脚描述如表 1.2.1 所示：

| 序号 | 名称 | 说明 |
|---|---|---|
| 1 | GND | 地 |
| 2 | 5V | 5V 供电口，只可以供电 |
| 3 | 3.3V | 3.3V 供电口，当使用 5V 供电的时候，这里可以输出 3.3V 电压给外部使用 |
| 4 | XCS | 片选输入（低有效） |
| 5 | XDCS | 数据片选/字节同步 |
| 6 | SCK | SPI 总线时钟线 |
| 7 | SI | SPI 总线数据输入线 |
| 8 | SO | SPI 总线数据输出线 |
| 9 | DREQ | 数据请求 |
| 10 | RST | 复位引脚（硬复位，低电平有效） |

表 1.2.1 供电与通信接口 P1 口各引脚功能表

用 ATK-VS1043 MP3 模块来播放音频文件是非常简单的，一般的音频文件（MP3/WMA/OGG/WAV/MIDI/AAC 等），只需要简单的 3 步操作即可实现音频播放。

**1）复位 VS1053**

这里包括了硬复位（拉低 RST）和软复位（设置 MODE 寄存器的 SM_RESET 位为 1），是为了让 VS1053 的状态回到原始状态，准备解码下一首歌曲。这里建议大家在每首歌曲播放

之前都执行一次硬件复位和软件复位，以便更好的播放音乐。

**2）配置 VS1053 的相关寄存器**

这里我们配置的寄存器包括 VS1053 的模式寄存器（MODE）、时钟寄存器（CLOCKF）、音调寄存器（BASS）、音量寄存器（VOL）等。

**3）发送音频数据**

当经过以上两步配置以后，我们剩下来要做的事情，就是往 VS1053 里面扔音频数据了，只要是 VS1053 支持的音频格式，直接往里面丢就可以了，VS1053 会自动识别，并进行播放。不过发送数据要在 DREQ 信号的控制下有序的进行，不能乱发。这个规则很简单：只要 DREQ 变高，就向 VS1053 发送 32 个字节。然后继续等待 DREQ 变高，直到音频数据发送完。

经过以上三步，我们就可以利用模块来播放音乐了。

# 2、硬件连接

本章实验功能简介：开机检测SD卡和字库是否存在，如果检测无问题，则对VS1053进行正弦测试和寄存器测试，之后开始循环播放SD卡MUSIC文件夹下面的歌曲，并在TFTLCD上显示歌曲名字、播放时间、歌曲总时间、歌曲总数目、当前歌曲的编号等信息。KEY0 用于选择下一曲，KEY1 用于选择上一曲。本实验用DS0来象征性的指示程序的运行。

本实验用到的资源如下：

1）DS0。

2）TFTLCD 液晶模块。

3）SD 卡（**并在SD卡根目录新建MUSIC文件夹，存放音乐文件在里面**）。

4）KEY0、KEY1。

5）ATK-VS1053 MP3模块

6）耳机

ALIENTEK 精英 STM32F103 开发板与 ATK-VS1053 MP3 模块的连接关系如图 2.1 所示：

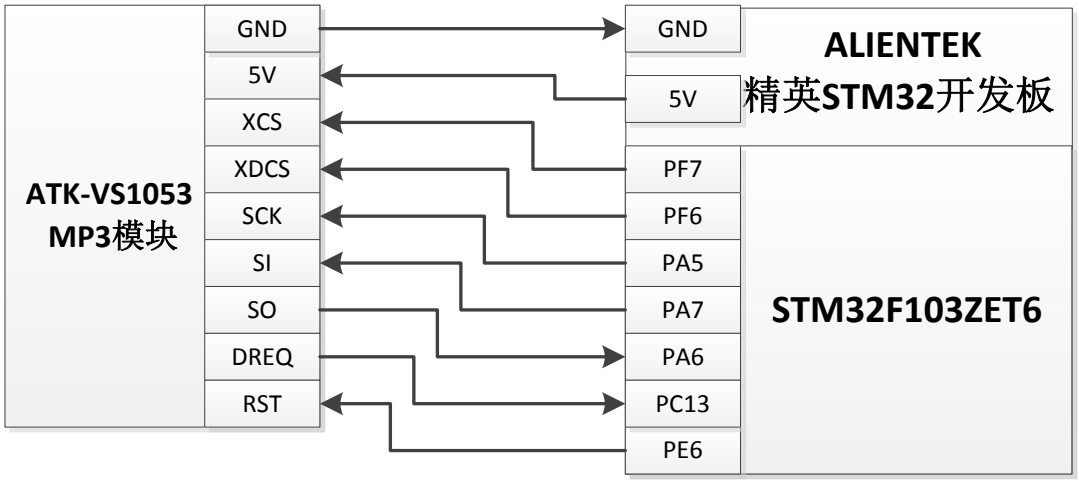


图 2.1 ATK-VS1053 MP3 模块与精英 STM32 开发板连接关系图

上表中，就是ALIENTEK 精英STM32开发板与ATK-VS1053 MP3模块的连接示意图。实物连接如图2.2所示：

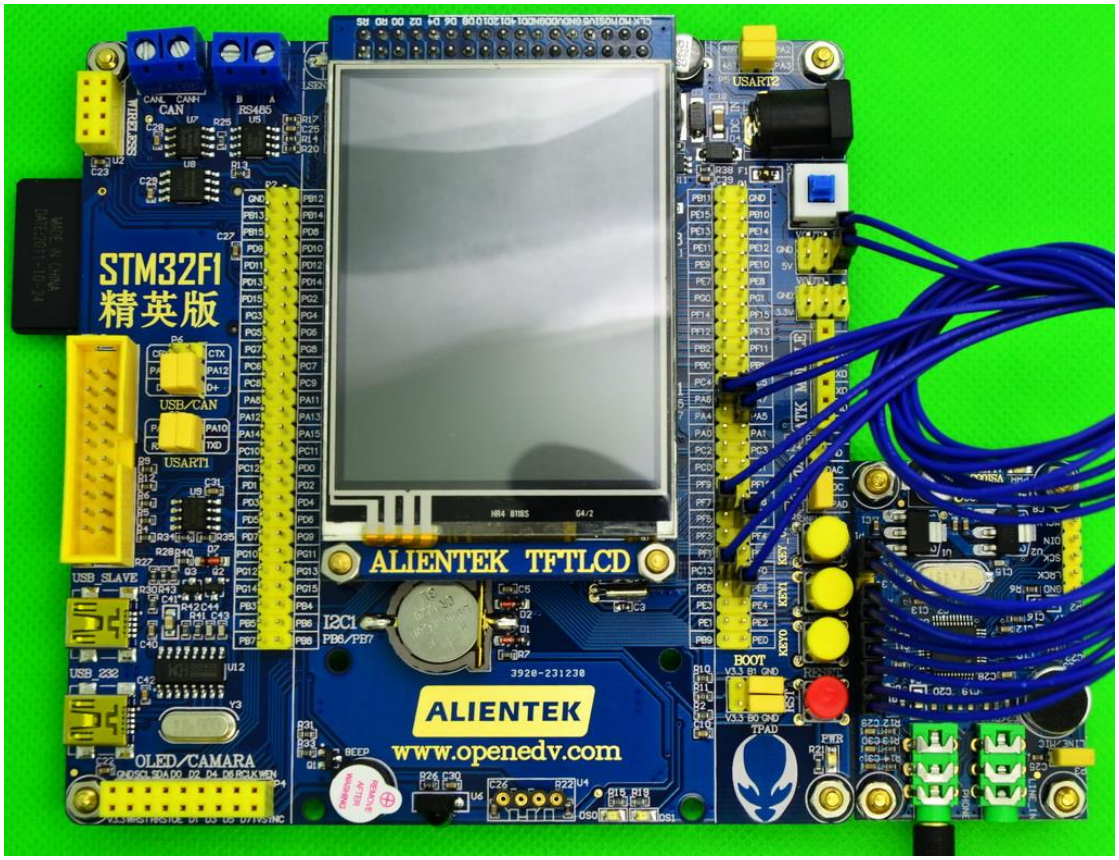


图 2.2 ATK-VS1053 模块与精英 STM32 开发板连接实物图

图中，我们总共用了 9 根杜邦线连接 ATK-VS1053 MP3 模块与精英 STM32 开发板，供电采用 5V(开发板的 VOUT2)直接供电，完全与图 2.1 所示的关系图一致。

## 3、软件实现

本章，我们在精英STM32开发板标准例程（寄存器版）第37章（汉字显示实验）实验的基础上修改，先打开第37章实验的工程，首先在HARDWARE文件夹所在的文件夹下新建一个VS10XX的文件夹。在该文件夹下新建vs10xx.c和vs10xx.h，然后在工程目录下新建：APP 文件夹，在该文件夹下新建：mp3player.c和mp3player.h 两个文件。

打开VS10XX.c，里面的代码我们不一一贴出了，这里挑几个重要的函数给大家介绍一下，首先要介绍的是VS_Soft_Reset，该函数用于软复位VS1053，其代码如下：

```
//软复位 VS10XX
void VS_Soft_Reset(void)
{
    u8 retry=0;
    while(VS_DQ==0);                        //等待软件复位结束
    VS_SPI_ReadWriteByte(0Xff);             //启动传输
    retry=0;
    while(VS_RD_Reg(SPI_MODE)!=0x0800)      // 软件复位,新模式
    {
        VS_WR_Cmd(SPI_MODE,0x0804);         // 软件复位,新模式
        delay_ms(2);//等待至少 1.35ms
```

```
            if(retry++>100)break;
        }
        while(VS_DQ==0);//等待软件复位结束
        retry=0;
        while(VS_RD_Reg(SPI_CLOCKF)!=0X9800)    //设时钟,3 倍频 ,1.5xADD
        {
            VS_WR_Cmd(SPI_CLOCKF,0X9800);      //设置时钟,3 倍频 ,1.5xADD
            if(retry++>100)break;
        }
        delay_ms(20);
    }
```

该函数比较简单，先配置一下VS1053的模式顺便执行软复位操作，在软复位结束之后，再设置时钟，待时钟配置完成,即完成软复位。接下来，我们介绍一下VS_WR_Cmd 函数，该函数用于向VS1053写命令，代码如下：

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

该函数用于向VS1053发送命令，这里要注意VS1053的写操作比读操作快（写1/4 CLKI，读1/6 CLKI），虽然说写寄存器最快可以到1/4CLKI，但是经实测在1/4CLKI 的时候会出错，所以在写寄存器的时候最好把SPI 速度调慢点,然后在发送音频数据的时候,就可以1/4CLKI的速度了。有写命令的函数，当然也有读命令的函数了。VS_RD_Reg 用于读取VS1053的寄存器的内容。该函数代码如下：

```
//读 VS10XX 的寄存器
//注意不要用倍速读取,会出错
u16 VS_RD_Reg(u8 address)
{
    u16 temp=0;
    while(VS_DQ==0);      //等待空闲
    VS_SPI_SpeedLow();    //低速
    VS_XDCS=1;
    VS_XCS=0;
```

```
    VS_SPI_ReadWriteByte(VS_READ_COMMAND); //发送 VS10XX 的读命令
    VS_SPI_ReadWriteByte(address);          //地址
    temp=VS_SPI_ReadWriteByte(0xff);        //读取高字节
    temp=temp<<8;
    temp+=VS_SPI_ReadWriteByte(0xff);       //读取低字节
    VS_XCS=1;
    VS_SPI_SpeedHigh();//高速
    return temp;
}
```

该函数用于读取某个寄存器的值，我们这里不多说了。VS10XX.c 的剩余代码和VS10XX.h的代码，这里就不贴出来了。读者可以去光盘查看详细源码。

保存VS10XX.c和VS10XX.h 两个文件。把VS10XX.c 加入到HARDWARE 组下。然后我们打开mp3player.c，该文件我们仅介绍一个函数，其他代码请看光盘的源码。这里要介绍的是mp3_play_song函数，该函数代码如下：

```
//返回值:0,正常播放完成
//       1,下一曲
//       2,上一曲
//       0XFF,出现错误了
u8 mp3_play_song(u8 *pname)
{
    FIL* fmp3;
    u16 br;
    u8 res,rval;
    u8 *databuf;
    u16 i=0;
    u8 key;
    rval=0;
    fmp3=(FIL*)mymalloc(SRAMIN,sizeof(FIL));  //申请内存
    databuf=(u8*)mymalloc(SRAMIN,4096);       //开辟 4096 字节的内存区域
    if(databuf==NULL||fmp3==NULL)rval=0XFF ;  //内存申请失败.
    if(rval==0)
    {
        VS_Restart_Play();                    //重启播放
        VS_Set_All();                         //设置音量等信息
        VS_Reset_DecodeTime();                //复位解码时间
        res=f_typetell(pname);                //得到文件后缀
        if(res==0x4c)//如果是 flac,加载 patch
        {
            VS_Load_Patch((u16*)vs1053b_patch,VS1053B_PATCHLEN);
        }
        res=f_open(fmp3,(const TCHAR*)pname,FA_READ);//打开文件
        if(res==0)//打开成功.
        {
            VS_SPI_SpeedHigh();  //高速
```

```
                while(rval==0)
                {
                    res=f_read(fmp3,databuf,4096,(UINT*)&br);//读出 4096 个字节
                    i=0;
                    do//主播放循环
                    {
                        if(VS_Send_MusicData(databuf+i)==0)//给 VS10XX 发送数据
                        {
                            i+=32;
                        }else
                        {
                            key=KEY_Scan(0);
                            switch(key)
                            {
                                case KEY0_PRES:
                                    rval=1;         //下一曲
                                    break;
                                case KEY1_PRES:
                                    rval=2;         //上一曲
                            }
                            mp3_msg_show(fmp3->fsize);//显示信息
                        }
                    }while(i<4096);//循环发送 4096 个字节
                    if(br!=4096||res!=0)
                    {
                        rval=0;
                        break;//读完了.
                    }
                }
                f_close(fmp3);
            }else rval=0XFF;//出现错误
        }
        myfree(SRAMIN,databuf);
        myfree(SRAMIN,fmp3);
        return rval;
    }
```

该函数，就是我们解码音乐的核心函数了，该函数的参数为歌曲的路径+名字,比如:0:\MUSIC\海阔天空.wav。然后为文件结构体以及数据缓存区申请内存，之后重设VS1053，然后根据歌曲的类型，如果是flac则加载patch文件（VS1053解码flac要打补丁），然后打开这个文件，开始播放。

播放的时候，设置SPI工作在高速模式（9Mhz时钟），然后死循环发送音频数据给VS1053，在死循环里面等待DREQ信号的到来，每次VS_DQ变高，就向VS1053发送32 个字节，直到整个文件读完。此段代码还包含了对按键的处理（暂停/播放、上一首、下一首）及当前播放的歌曲的一些状态（编号、码率、播放时间、总时间、歌曲名字等）的显示（通过mp3_msg_show

函数显示），播放完成后，关闭文件，然后释放内存，完成一首歌曲的播放。

mp3player.c 的其他代码和mp3player.h在这里就不详细介绍了，请大家直接参考本例程源码。最后，我们在test.c里面修改main 函数如下：

```
int main(void)
{
    Stm32_Clock_Init(9);          //系统时钟设置
    uart_init(72,115200);         //串口初始化为 115200
    delay_init(72);               //延时初始化
    usmart_dev.init(72);          //初始化 USMA RT
    LED_Init();                   //初始化与 LED 连接的硬件接口
    KEY_Init();                   //初始化按键
    LCD_Init();                   //初始化 LCD
    W25QXX_Init();                //初始化 W25Q128
    VS_Init();                    //初始化 VS1053
    my_mem_init(SRAMIN);          //初始化内部内存池
    exfuns_init();                //为 fatfs 相关变量申请内存
    f_mount(fs[0],"0:",1);        //挂载 SD 卡
    f_mount(fs[1],"1:",1);        //挂载 FLASH.
    POINT_COLOR=RED;
    while(font_init())            //检查字库
    {
        LCD_ShowString(30,50,200,16,16,"Font Error!"); delay_ms(200);
        LCD_Fill(30,50,240,66,WHITE);//清除显示
    }
    Show_Str(30,50,200,16,"精英 STM32 开发板",16,0);
    Show_Str(30,70,200,16,"音乐播放器实验",16,0);
    Show_Str(30,90,200,16,"正点原子@ALIENTEK",16,0);
    Show_Str(30,110,200,16,"2015 年 4 月 30 日",16,0);
    Show_Str(30,130,200,16,"KEY0:NEXT     KEY1:PREV",16,0);
    while(1)
    {
        LED1=0;
        Show_Str(30,170,200,16,"存储器测试...",16,0);
        printf("Ram Test:0X%04X\r\n",VS_Ram_Test());//打印 RAM 测试结果
        Show_Str(30,170,200,16,"正弦波测试...",16,0);
        VS_Sine_Test();
        Show_Str(30,170,200,16,"<<音乐播放器>>",16,0);
        LED1=1;
        mp3_play();
    }
}
```

该函数先检测SD卡是不是在位（初始化SD卡），存在则继续，不存在在报错，然后检测外部flash是否存在字库，如果不存在字库，则报错（**此时需要先下载汉字显示实验，更新字库，更新完字库后，再下载本实验**），如果存在则继续下面的操作：对VS1053进行正弦

测试和寄存器测试，随后，开始播放MUSIC文件夹下面的歌曲。

最后，我们将VS_Set_Bass、VS_Set_Vol和VS_Set_Effect等函数加入usmart控制，这样我们就可以利用usmart来设置VS1053的音效了。

软件部分就介绍到这里。

## 4、验证

在代码编译成功之后，我们下载代码到ALIENTEK 精英STM32 开发板上，当检测到SD卡（**一定要在SD卡根目录新建MUSIC文件夹，并存放一些歌曲在MUSIC文件夹里面**）后，执行完两个测试（SIN测试和RAM测试）之后，就开始自动播放歌曲了，如图4.1所示：


图4.1 MP3播放中

从上图可以看出，当前正在播放第2首歌曲，MUSIC文件夹下总共有5首歌曲，歌曲名、播放时间、总时长及码率等信息等也都有显示。此时DS0会随着音乐的播放而闪烁。

只要我们在解码模块插入耳机，就能听到歌曲的声音了。同时，我们可以通过按KEY0和KEY1 来切换下一曲和上一曲。同时，可以通过串口调试助手（设置波特率为：115200）发送VS_Set_Bass、VS_Set_Vol和VS_Set_Effect等三个函数，来设置播放音效。

至此，我们就完成了一个简单的MP3 播放器了，在此基础上进一步完善，就可以做出一个比较实用的MP3了。大家可以自己发挥想象，做出一个你心仪的MP3。

正点原子@ALIENTEK
公司网址：www.alientek.com
技术论坛：www.openedv.com
电话：020-38271790
传真：020-36773971