



开源共享

携手共进

深圳普中科技有限公司

官方网站: www.prechin.cn

技术论坛: www.prechin.net

技术 QQ: 2489019400

咨询电话: 0755-61139052

PZ-VS1053 MP3 模块开发手册

前几年，MP3 曾经风行一时，几乎人手一个。如果能自己做一个 MP3，我想对于很多朋友来说是一件十分骄傲的事情。PZ-VS1503 模块内就集成了一颗非常强劲的 MP3 解码芯片：VS1053，利用该芯片我们可以实现 MP3/OGG/WMA/WAV/FLAC/AAC/MIDI 等各种音频文件的播放。本章我们将利用普中 STM32 开发板实现一个简单的 MP3 播放器。本章分为如下几个部分：

- 1 PZ-VS1053 MP3 模块介绍
- 2 硬件设计
- 3 软件设计
- 4 实验现象

普中科技STM32开发板

1 PZ-VS1053 MP3 模块介绍

1.1 特性参数

PZ-VS1053 MP3 模块是深圳普中科技有限公司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。

模块通过 SPI 接口与外部单片机通信，模块可以直接与 3.3V 单片机系统连接，也可以方便的与 5V 单片机系统连接（**特别注意：模块的信号线不能直接接 5V 单片机，如果要接，推荐在信号线上串联 1K 左右的电阻。**）。模块自带稳压芯片，外部仅需提供 5V/3.3V 电压即可，使用非常方便，该模块各参数如图所示：

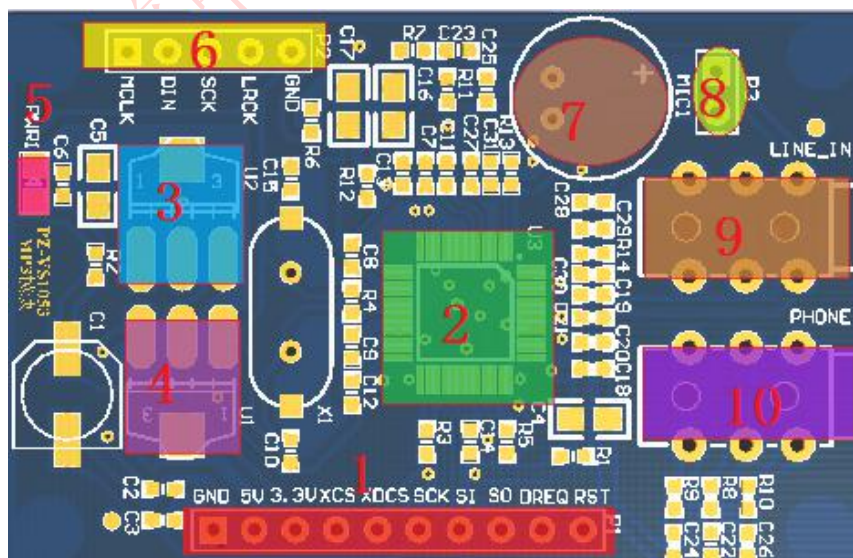
项目	说明
接口特性	3.3V(串电阻后，可与 5V 系统连接)
解码格式	MP3、OGG、WMA、WAV、MIDI、AAC、FLAC（需要加载 patch）
编码格式	WAV(PCM/IMA ADPCM)、OGG（需要加载 patch）
对外接口	1 路 3.5mm 耳机接口、1 路 3.5mm LINE IN 接口、IIS 接口、供电及控制接口
板载录音	支持
工作温度	-30℃~85℃
其他特性	音量控制、高低音控制、EarSpeaker 空间效果、解码时间输出
模块尺寸	34.64mm*52.83mm

DAC 分辨率	18 位
总谐波失真 (THD)	0.07% (Max)
动态范围 (A-加权)	100dB
信噪比	94dB
通道隔离度 (串扰)	80dB@600欧+GBUF 53dB@30欧+GBUF
咪头 (MIC) 放大增益	26dB
咪头 (MIC) 总谐波失真	0.07% (Max)
咪头 (MIC) 信噪比	70dB
LINE IN 信号幅度	2800mVpp (Max)
LINE IN 总谐波失真	0.014% (Max)
LINE IN 信噪比	90dB
LINE IN 阻抗	80K欧
工作电压	DC3.3V/5.0V (推荐 5.0V 供电)
工作电流	15mA~40mA
Voh	2.31V (Min)
Vol	0.99V (Max)
Vih	1.26V (Min)
Vil	0.54V (Max)

1.2 使用说明

1.2.1 模块介绍

PZ-VS1053 MP3 模块是深圳普中科技有限公司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，模块资源图如图所示：



序号	模块资源
1	电源及SPI通信接口
2	VS1053B编解码芯片
3	1.8V稳压芯片
4	3.3V稳压芯片
5	模块电源指示灯
6	IIS输出接口
7	咪头（MIC）
8	LINE IN/MIC选择接口
9	LINE IN接口
10	音频输出接口

从上图可以看出，PZ-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 34.64mm*52.83mm，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。

PZ-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

PZ-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 34.64mm*52.83mm，并带有安装孔位，小巧精致；

1.2.2 模块引脚说明

PZ-VS1053 MP3 模块总共有 3 组排针：P1、P2 和 P3，均采用纯铜镀金排针，2.54mm 间距，方便与外部设备连接。

P1 排针为模块的供电与通信接口，采用 1*10P 排针，各引脚详细描述如图所示：

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）

P2 排针为模块的 IIS 输出接口，采用 1*5P 排针，各引脚详细描述如图所示：

序号	名称	说明
1	MCLK	主时钟
2	DIN	数据输出
3	SCLK	位时钟
4	LRCK	帧时钟
5	GND	地

P3 排针为 LINE IN/MIC 选择接口，采用 1*2P 排针，各引脚详细描述如图所示：

序号	名称	说明
1	MIC	咪头正极信号
2	MICP/LINE1	咪头正极输入/线路输入 1

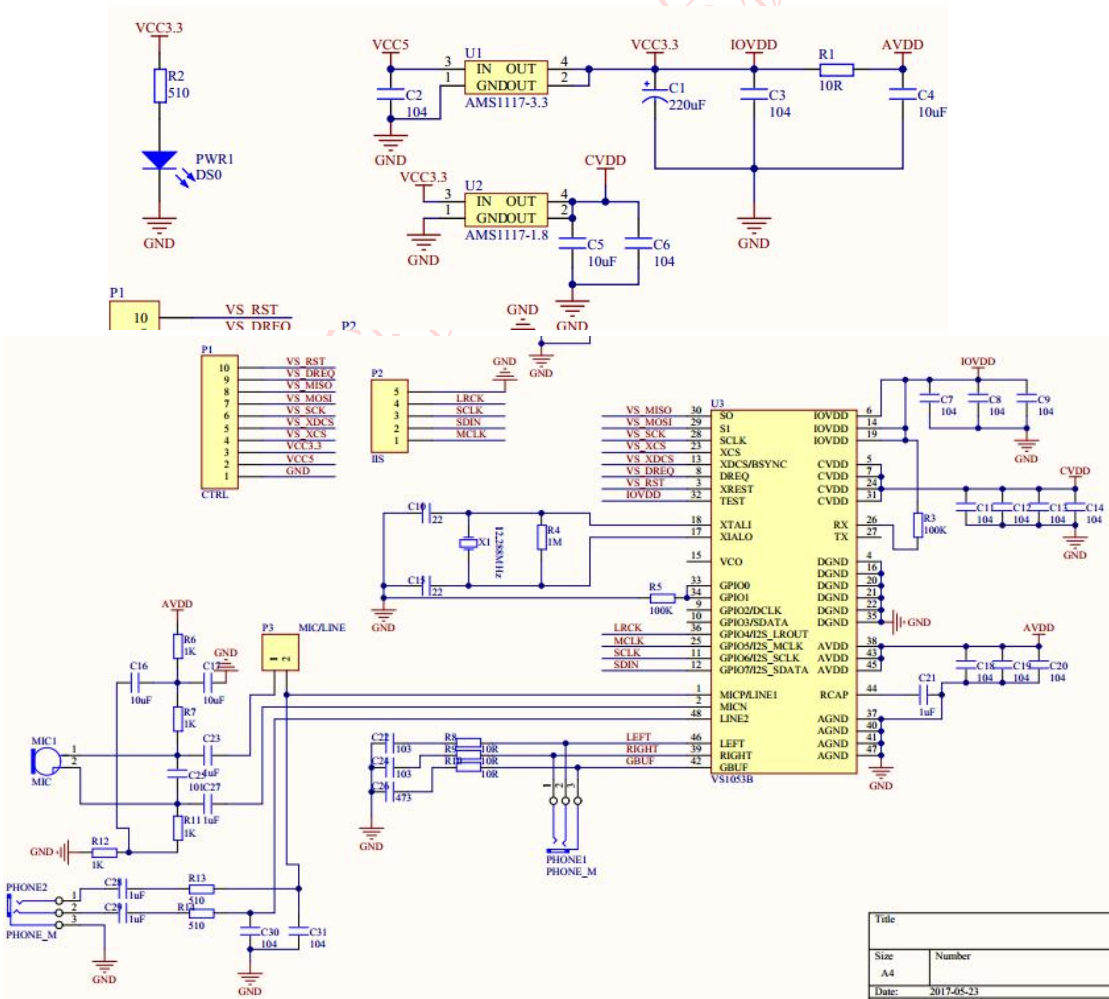
P3 接口不对外连接，当用跳线帽短接 P3 的 1 脚和 2 脚的时候（默认设

置)，咪头是直接连接在 VS1053 芯片上的，录音的时候，我们可以直接通过咪头实现声音采集。当不采用咪头拾音，而采用外部线路输入的时候， 为了防止 MIC 拾音器对线路输入的影响，此时我们拔了 P3 的跳线帽即可。

1.2.3 模块使用

VS1053 通过 SPI 接口来接收输入的音频数据流，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，它就会自动帮我们解码了，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。

PZ-VS1053 MP3 模块原理图如图所示：（看不清楚的话可以打开模块资料-原理图查看）



VS1053 芯片需要通过 7 根线同 STM32 连接,他们是: VS_MISO、VS_MOSI、VS_SCK、 VS_XCS、 VS_XDCS、 VS_DREQ 和 VS_RST。该模块同 STM32 的连接关

系如图所示：

PZ-VS1053 MP3模块	普中STM32F1开发板 (IO)
RST	PG8
DREQ	PG7
MISO	PA6
MOSI	PA7
SCK	PA5
XDCS	PG6
XCS	PE6
VCC3.3	
VCC5	5V
GND	GND

其中 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。VS_MISO、VS_MOSI 和 VS_SCK 则是 VS1053 的 SPI 接口，他们在 VS_XCS 和 VS_XDCS 下面来执行不同的操作。从上图可以看出，VS1053 的 SPI 是接在 STM32 的 SPI1 上面的。

VS1053 的 SPI 支持两种模式：1，VS1002 有效模式（即新模式）。2，VS1001 兼容模式。这里我们仅介绍 VS1002 有效模式（此模式也是 VS1053 的默认模式）。新模式下 VS1053 的 SPI 信号线功能描述如图所示：

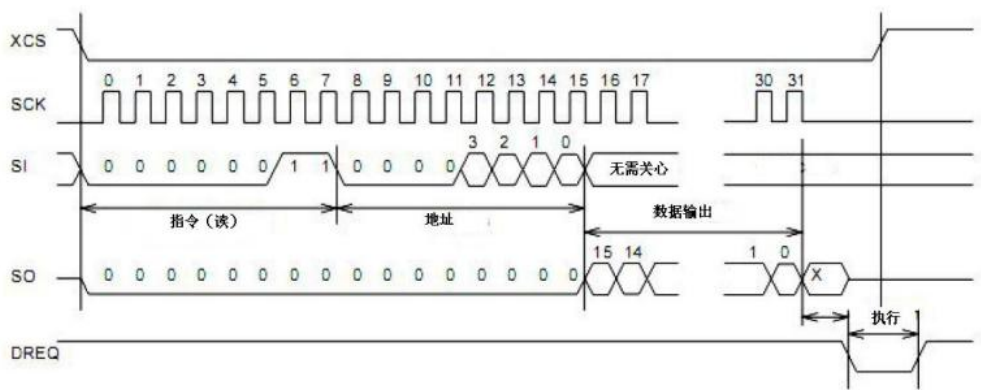
SDI	SCI	描述
XDCS	XCS	SDI/SCI 片选信号，低电平有效。高电平强制 SPI 进入 Standby 模式，结束当前操作，且 SO 变为高阻态。如果 SM_SDISHARE 位设置为 1，则不使用 XDCS，而有 XCS 内部反相后代替 XDCS。不过不推荐这种设置方式。
SCK		串行时钟输入。SCK 在传输的时候可以被打断，但是必须保持 XCS/XDCS 低电平不变，否则传输将中断。
SI		串行数据输入。在片选有效的情况下，SI 在 SCK 的上升沿处采样。所以，MCU 必须在 SCK 的下降沿上更新数据。
SO		串行数据输出。在读操作时，数据在 SCK 的下降沿从此引脚输出，在写操作时为高阻态。

VS1053 的 SPI 数据传送，分为 SDI 和 SCI，分别用来传输数据/命令。SDI 和前面介绍的 SPI 协议一样的，不过 VS1053 的数据传输是通过 DREQ 控制的，主机在判断 DREQ 有效（高电平）之后，直接发送即可（一次可以发送 32 个字节）。

这里我们重点介绍一下 SCI。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。读写操作可以读写单个寄存器，在 SCK 的上升沿读出数据位，所以主机必须在下降沿刷新数据。SCI 的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

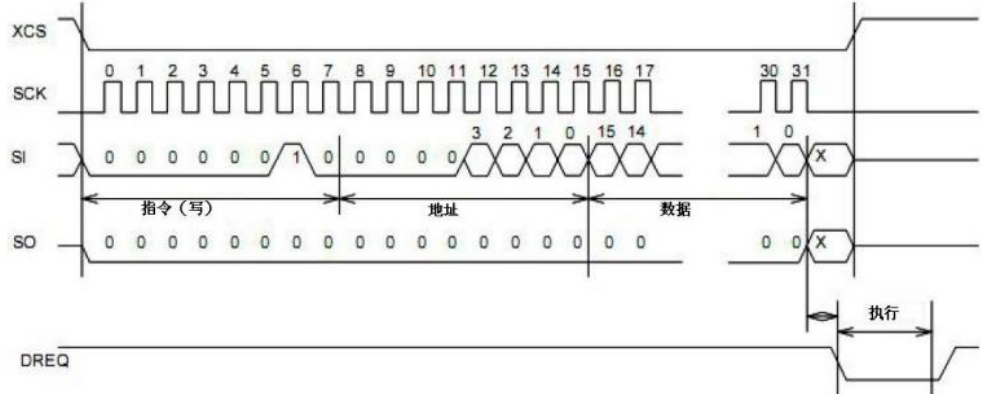
为 0X03，写为 0X02。

一个典型的 SCI 读时序如图所示：



从上图可以看出，向 VS1053 读取数据，通过先拉低 XCS (VS_XCS)，然后发送读指令 (0X03)，再发送一个地址，最后，我们在 SO 线 (VS_MISO) 上就可以读到输出的数据了。而同时 SI (VS_MOSI) 上的数据将被忽略。

看完了 SCI 的读，我们再来看看 SCI 的写时序，如图所示：



写时序图和读时序图基本类似，都是先发指令再发地址。不过写时序中，我们的指令是写指令 (0X02)，并且数据是通过 SI 写入 VS1053 的，SO 则一直维持低电平。细心的读者可能发现了，在这两个图中，DREQ 信号上都产生了一个短暂的低脉冲，也就是执行时间。这个不难理解，我们在写入和读出 VS1053 的数据之后，它需要一些时间来处理内部的事情，这段时间是不允许外部打断的，所以，我们在 SCI 操作之前，最好判断一下 DREQ 是否为高电平，如果不是，则等待 DREQ 变为高。

了解了 VS1053 的 SPI 读写，我们再来看看 VS1053 的 SCI 寄存器，VS1053 的所有 SCI 寄存器如图所示：

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	HDAT0	流的数据标头 0
0X09	R	0X0000	HDAT1	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

VS1053 总共有 16 个 SCI 寄存器，这里我们不介绍全部的寄存器，仅仅介绍几个我们实验需要用到的寄存器。

首先是 MODE 寄存器，该寄存器用于控制 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。MODE 寄存器的各位描述如下所示：

位	0	1	2	3	4	5	6	7
名称	SM_DIFF	SM_LAYER12	SM_RESET	SM_CANCEL	SM_EARSPEAKER_LO	SM_TEST	SM_STREAM	SM_EARSPEAKER_HI
功能	差分	允许MPEG I&II	软件复位	取消当前文件的解码	EarSpeaker 低设定	允许SDI 测试	流模式	EarSpeaker 高设定
描述	0, 正常的同相音频 1, 左通道反相	0, 不允许 1, 允许	0, 不复位 1, 复位	0, 不取消 1, 取消	0, 关闭 1, 激活	0, 禁止 1, 允许	0, 不是 1, 是	0, 关闭 1, 激活
位	8	9	10	11	12	13	14	15
名称	SM_DACT	SM_SDIORD	SM_SDISHARE	SM_SDINEW	SM_ADPCM	-	SM_LINE1	SM_CLK_RANGE
功能	DCLK的有效边沿	SDI位顺序	共享SPI片选	VS1002本地SPI模式	ADPCM激活	-	咪/线路1选择	输入时钟范围
描述	0, 上升沿 1, 下降沿	0, MSB在前 1, MSB在后	0, 不共享 1, 共享	0, 非本地模式 1, 本地模式	0, 不激活 1, 激活	-	0, MICP 1, LINE1	0, 12..13Mhz 1, 24..26Mhz

这个寄存器，我们这里只介绍一下第 2 和第 11 位，也就是 SM_RESET 和 SM_SDINEW。其他位，我们用默认的即可。这里 SM_RESET，可以提供一次软复位，建议在每播放一首歌曲之后，软复位一次。SM_SDINEW 为模式设置位，这里我们选择的是 VS1002 新模式(本地模式)，所以设置该位为 1（默认的设置）。其他位的详细介绍，请参考 VS1053 的数据手册。

接着我们看看 BASS 寄存器，该寄存器可以用于设置 VS1053 的高低音效。

该寄存器的各位描述如图所示：

名称	位	描述
ST_AMPLITUDE	15: 12	高音控制, 1.5dB 步进 (-8..7 ,为 0 表示关闭)
ST_FREQLIMIT	11: 8	最低频限 1000Hz 步进 (0..15)
SB_AMPLITUDE	7: 4	低音加重, 1dB 步进 (0..15 ,为 0 表示关闭)
SB_FREQLIMIT	3: 0	最低频限 10Hz 步进 (2..15)

通过这个寄存器以上位的一些设置，我们可以随意配置自己喜欢的音效（其实就是高低音的调节）。VS1053 的 EarSpeaker 效果则由 MODE 寄存器控制，其各位的功能在前面已列出。

接下来，我们介绍一下 CLOCKF 寄存器，这个寄存器用来设置时钟频率、倍频等相关信息，该寄存器的各位描述如图所示：

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	$CLKI = XTALI \times (SC_MULT \times 0.5 + 1)$		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

我们重点看下该寄存器的 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：

$$SC_FREQ = (XTALI - 8000000) / 4000$$

式中为 XTALI 的单位为 Hz。上图中 CLKI 是内部时钟频率，XTALI 是外部晶振的时钟频率。由于我们模块使用的是 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。

接下来，我们看看 DECODE_TIME 这个寄存器。该寄存器是一个存放解码时间的寄存器，以秒钟为单位，我们通过读取该寄存器的值，就可以得到解码时间了。不过它是一个累计时间，所以我们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。

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。

最后我们介绍一下 VOL 这个寄存器，该寄存器用于控制 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！

关于 VS1053 的介绍，我们就介绍到这里，更详细的介绍请看 VS1053 的数据手册。

接下来我们说说如何通过最简单的步骤，来控制 VS1053 播放一般的音频文件（MP3/WMA/OGG/WAV/MIDI/AAC 等）音乐。

（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，我们就可以播放音乐了。这一部分就先介绍到这里。

2 硬件设计

本实验使用到硬件资源如下：

（1）D1 指示灯

（2）K_UP、K_DOWN、K_LEFT、K_RIGHT 按键

- (3) 串口 1
- (4) TFTLCD 模块
- (5) SD 卡
- (6) 外部 FLASH (EN25QXX)
- (7) PZ-VS1053 MP3 模块

这些电路在前面章节都介绍过，这里就不多说，下面我们再来看下 PZ-VS1053 MP3 模块与开发板如何连接的。前面我们介绍了该模块的接口管脚功能，我们通过杜邦线将 PZ-VS1053 MP3 模块与 STM32 开发板的管脚连接，连接关系如图所示：

PZ-VS1053 MP3模块	普中STM32F1开发板 (IO)
RST	PG8
DREQ	PG7
MISO	PA6
MOSI	PA7
SCK	PA5
XDCS	PG6
XCS	PE6
VCC3.3	
VCC5	5V
GND	GND

注意：大家需要准备 1 个 TF 卡（在里面新建一个 MUSIC 文件夹，并存放一些歌曲在此文件夹下）和一个耳机，将 TF 卡插入到 STM32 开发板上，将耳机插入到 PZ-VS1053 MP3 模块的 PHONE 接口，然后下载本实验就可以通过耳机来听歌了。

3 软件设计

本实验所实现的功能为：开机先检测 SD 卡是否存在，如果检测无问题，则对 VS1053 进行 RAM 测试和正弦测试，测试完后开始循环播放 SD 卡 MUSIC 文件夹里面的歌曲（必须在 SD 卡根目录建立一个 MUSIC 文件夹，并存放歌曲在里面），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。K_RIGHT 用于选择下一曲，K_LEFT 用于选择上一曲，K_UP 和 K_DOWN 用来调节音量。D1 用于指示程序运行状态。

我们打开本实验工程，可以看到我们的工程 APP 列表中多了 vs10xx.c 和 mp3player.c 源文件，以及头文件 vs10xx.h、mp3player.h 和 flac.h 等 5 个

文件。本实验工程代码比较多，我们就不一一列出了，仅挑几个重要的地方进行讲解。

首先打开 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) // 设置 VS10XX 的时钟, 3 倍
    频 , 1.5xADD
    {
        VS_WR_Cmd(SPI_CLOCKF, 0X9800); // 设置 VS10XX 的时钟, 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，再设置好时钟，完成一次软复位。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 1/7 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
//address: 寄存器地址
//返回值: 读到的值
//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); //非等待空闲状态
    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;
}

```

该函数的作用和 VS_WR_Cmd 的作用基本相反，用于读取寄存器的值。vs10xx.c 的剩余代码、vs10xx.h 以及 flac.h 的代码，这里就不贴出来了，其中 flac.h 仅仅用来存储播放 flac 格式所需要的 patch 文件，以支持 flac 解码。大家可以打开查看他们的详细源码。然后我们打开 mp3player.c，该文件我们仅介绍一个函数，其他代码请看工程源码。这里要介绍的是 mp3_play_song 函数，该函数代码如下：

```

//播放一曲指定的歌曲

//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=FATFS_ScanFileType(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

```

个字节

送音频数据

```
while(rval==0)
{
    res=f_read(fmp3, databuf, 4096, (UINT*)&br); //读出 4096

    i=0;
    do//主播放循环
    {
        if(VS_Send_MusicData(databuf+i)==0)//给 VS10XX 发

        {
            i+=32;
        }else
        {
            key=KEY_Scan(0);
            switch(key)
            {
                case KEY_RIGHT:
                    rval=1;        //下一曲
                    break;
                case KEY_LEFT:
                    rval=2;        //上一曲
                    break;
            }
            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 MP3 的核心函数了，该函数在初始化 VS1053 后，根据文件格式选择是否加载 patch（如果是 flac 格式，则需要加载 patch），最后在死循环里面等待 DREQ 信号的到来，每次 VS_DQ 变高，就通过 VS_Send_MusicData 函数向 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（音量调节、上一首、下一首）及当前播放的歌曲的一些状态（码率、播放时间、总时间）显示。

mp3player.c 的其他代码和 mp3player.h 在这里就不详细介绍了，请大家直接参考源码。最后我们看看 main.c 文件的内容：

```

int main()
{

    SysTick_Init(72);

    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2); //中断优先级
    分组 分 2 组

    LED_Init();

    USART1_Init(9600);

    TFTLCD_Init(); //LCD 初始化

    KEY_Init();

    EN25QXX_Init(); //初始化 EN25Q128

```

```

VS_Init(); //初始化 VS1053
my_mem_init(SRAMIN); //初始化内部内存池
FATFS_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.

FRONT_COLOR=RED; //设置字体为红色
while(SD_Init()!=0)
{

    LCD_ShowString(10, 10, tftlcd_data.width, tftlcd_data.height, 16, "SD
Card Error!");
}

LCD_ShowFont16Char(10, 10, "普中科技");
LCD_ShowFont12Char(10, 30, "www.prechin.cn");
LCD_ShowFont12Char(10, 50, "音乐播放器实验");
LCD_ShowFont12Char(10, 70, "K_UP: 音量+");
LCD_ShowFont12Char(10, 90, "K_DOWN: 音量-");
LCD_ShowFont12Char(10, 110, "K_RIGHT: 下一曲");
LCD_ShowFont12Char(10, 130, "K_LEFT: 上一曲");

while(1)
{
    led2=0;
    LCD_ShowFont12Char(10, 170, "存储器测试...");
    printf("Ram Test:0X%04X\r\n", VS_Ram_Test()); //打印 RAM 测试
    结果

    LCD_ShowFont12Char(10, 170, "正弦波测试...");

```

```

VS_Sine_Test();
LCD_ShowFont12Char(10, 170, "《MP3 音乐播放实验》");
led2=1;
mp3_play();
}
}

```

该函数先检测 SD 卡是否存在，然后执行 VS1053 的 RAM 测试和正弦测试，这两个测试结束后，调用 mp3_play 函数开始播放 SD 卡 MUSIC 文件夹里面的音乐。软件部分就介绍到这里。

4 实验现象

将工程程序编译下载到开发板内并且将模块连接到开发板上，注意一定要插上 SD 卡，并且 SD 的根目录下的 MUSIC 文件夹要存放音乐文件，可以是 .mp3、.wav 等格式。当检测到有音频文件后就开始自动播放音乐了。可以看到 TFTLCD 上显示如下界面：



从上图可以看出，当前正在播放第 1 首歌曲，总共 1 首歌曲，歌曲名、播放时间、总时长、码率、音量等信息等也都有显示。此时 D1 会随着音乐的播放而闪烁。

只要我们在模块的耳机端子插入耳机，就能听到歌曲的声音了。同时，我们可以通过按 K_RIGHT 和 K_LEFT 来切换下一曲和上一曲，通过 K_UP 按键来控制音量增加，通过 K_DOWN 控制音量减小。

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。

普中科技STM32开发板