life.augmented

AN5086
应用笔记

使用标准SPI外设，在STM32L0系列微控制器上执行I2S协议模拟

引言

I^2S协议广泛用于从微控制器/DSP到音频编解码器传输音频数据，以播放（储存于存储器中的）音频内容，或者（从麦克风）捕获模拟声音。

本应用笔记描述了标准SPI（串行协议接口）和TIMER外设如何能够模拟I^2S接口。当应用程序由于物理限制（小封装）或因为已经使用（全双工音频交换），而不允许使用I^2S功能时，此解决方案非常有用。

作为基于STM32L0系列产品的应用示例，我们提出了一个常见的用例，其中，从UART外设异步接收数据，并通过所提出的模拟I^2S主接口同步传输到外部IC（集成电路）。

目录

表格索引

图片索引

1 I²S外设仿真原理

当I²S功能（包含在SPI外设中）不可使用或者已经被使用的时候（全双工或多音频编解码器系统），可以使用标准SPI和定时器外设模拟I²S协议主机或从机。本应用笔记描述了主I²S接口的仿真。

图1. 主发射器I²S仿真器概述

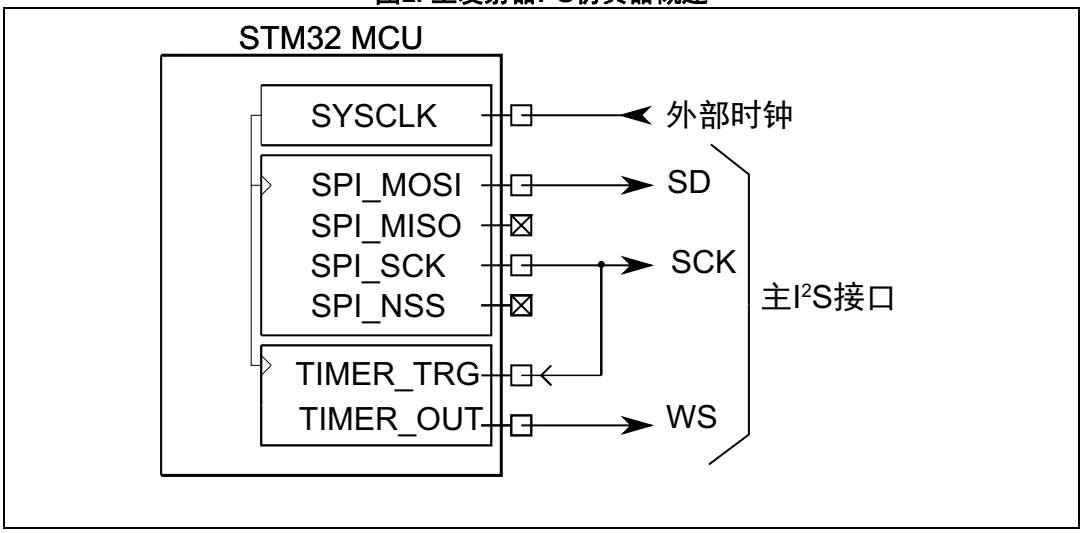


图2. 主接收器I²S仿真器概述

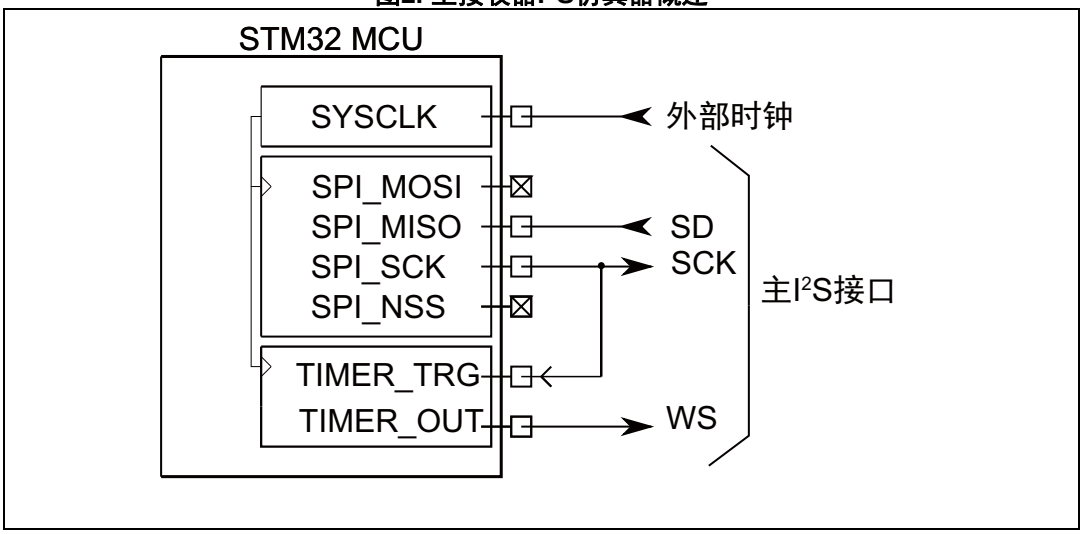


表 1 描述充分实现I²S接口仿真所需的外设清单：

表1. 用于I²S协议仿真的STM32外设约束条件

目的	外设	注释
系统时钟	HSI/MSI或HSE	用于I²S仿真外设的主时钟源。I²S协议频率源自此时钟
I²S串行数据和时钟	SPI（标准）	I²S协议SD和SCK信号生成基于SPI SCK和SPI MOSI信号。请注意，I²S仿真器也可以用于使用SPI MISO输入引脚的从配置。
I²S帧率	TIMER	定时器外设应具有以下特征： – extern ETR输入引脚 – 2 个通道 – 用于正确生成I²S WS信号的从配置

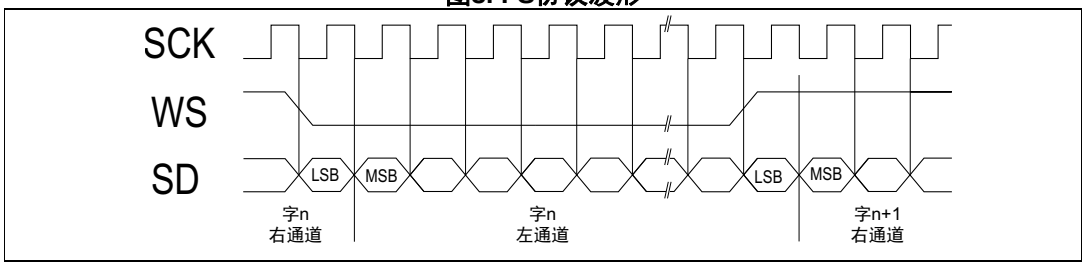
1.1 I²S数字音频协议

I²S接口（Inter-IC-Sound)是一种同步串行总线接口标准，用于数字音频设备的互连。在小端格式（MSB优先）下，可使用I²S传输音频数据，其速率与发射器的下降时钟沿（SCK信号）同步，并与接收器的上升沿同步。请注意，单独的时钟和数据会导致少量抖动。

（继承自CD音频格式的）数据代表立体声数字音频，因此，每个样本包含两个字——右声道样本和左声道样本。它并不使用两个数据通道，而是执行复用，在半个采样周期传输一个字，这样就将采样率加倍，可在每周期传输两个字。

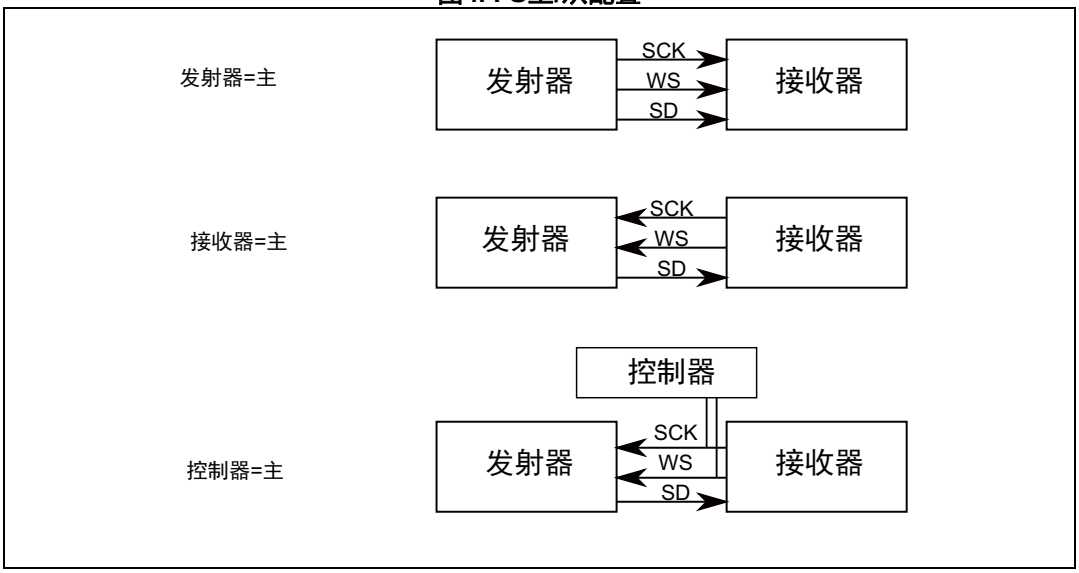
它使用控制信号WS（字选择）确定正在发送的字是右声道还是左声道。此信号还决定了数据的开始和结束；因此无需固定数据长度。因此，接收器和发射器数据长度可以不同，右声道和左声道数据长度也可以不同。在SCK的下降沿同步WS，比MSB早一个SCK周期，以便有足够时间执行储存和移位操作。

图3. I²S协议波形



与大多数通信协议一样，必须有主设备和从设备。主设备提供并控制SCK时钟和WS信号，从设备仅发送或接收数据。主设备可以是接收器、发送器或第三设备：

图4. I²S主/从配置



1.1.1 串行协议接口外设（SPI）

内部标准外设接口，即SPI，提供简单的通信接口，允许微控制器与外部设备通信。此接口具有高度可配置性，用于支持多重标准协议。SCK(SPI)可直接用作SCK(I²S)，MOSI(SPI)用作SD (I²S)。

第三个信号WS（字选择）相对于SCK信号同步和延迟，因此需要使用定时器外设。

表2. SPI外设信号

信号名称	SPI SCK	MOSI	SPI nSS	MISO
信号说明	SPI输出串行时钟	主设备输出 / 从设备输入（数据）	SPI输出 / 从设备选择	主设备输入 / 从设备输出（数据）
I²S仿真	SCK	SD	-	-
信号方向	输出	输出	输出	输入

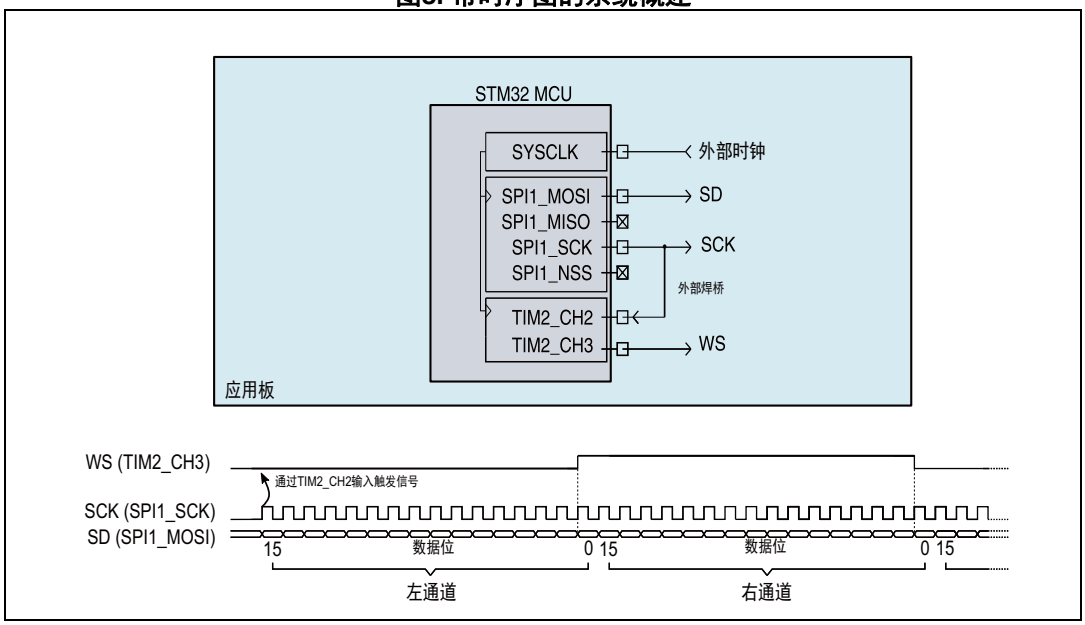
1.1.2 定时器外设

STM32内嵌多个定时器，为I/O相关的硬件任务提供定时资源。定时器的输出可生成波形，其输入可对外部事件做出反应。定时器功能齐全，提供多种工作模式，可以减少CPU的重复性和时间关键型任务，同时最大限度地减少接口电路的需求。定时器内核由一个16位向上计数器，一个用于编程计数周期的自动重载寄存器和一个用于调整计数器翻转中断率的重复计数器组成。定时器是产生WS（定时器CH3）信号的适当资源，该信号相对于SCK信号同步和延迟。定时器由SYSCLK内部定时（SPI外设使用相同的时钟源），其启动触发器（定时器CH2）通过STM32 GPIO连接到SCK信号。

定时器的极性、计数器周期及其初始值设置为与I²S协议配置匹配：

- 周期值 = 31
- 脉冲值 = 15
- 触发器极性=上升
- 输出比较器极性= TIM_OCPOLARITY_HIGH

图5. 带时序图的系统概述

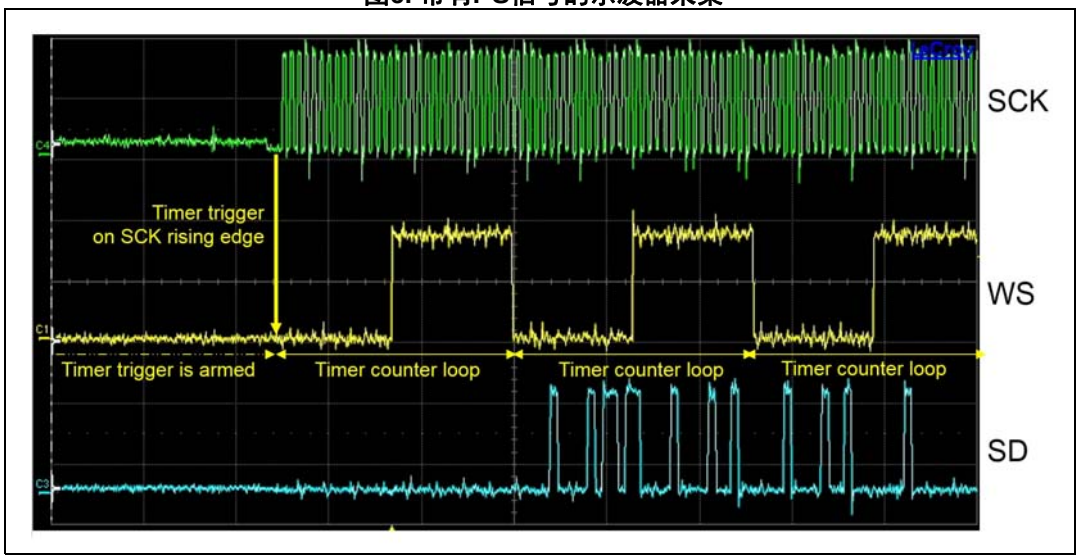


1.1.3 结果

使用标准示波器验证信号波形和时序。通过这种方法，我们还可以验证外设时序是否正确。我们还验证了固件与以下IS模式列表的兼容性：

- I 2 S标准 - 立体声通道
- I 2 S左对齐 - 立体声通道
- I 2 S右对齐 - 立体声通道

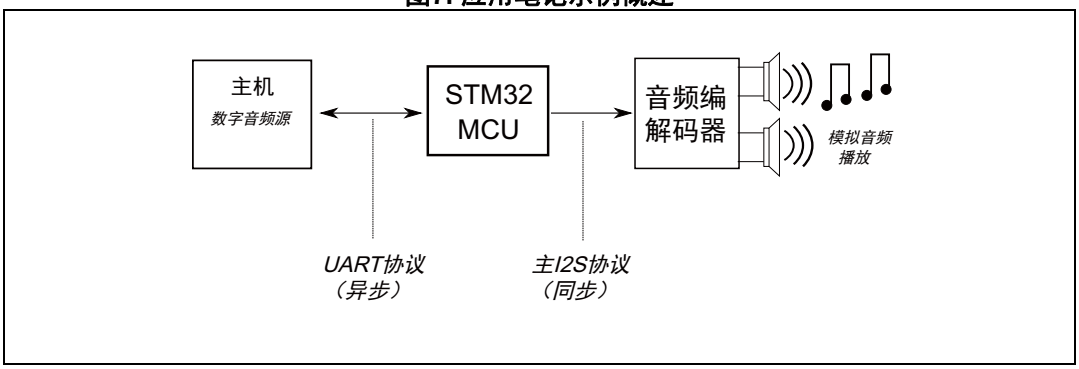
图6. 带有I²S信号的示波器采集



2 应用描述

本节介绍一种仿真的应用程序，它采用STM32MCU管理的UART和I²S协议，将主机存储的数字音频内容传输到音频编解码器。

图7. 应用笔记示例概述



以下应用示例旨在应用于相关目标市场。

2.1 48 kHz应用：CD音频质量回放

对于连接到音频编解码器的高端音频应用而言，常见的做法是在48 kHz采样率下使用I²S接口，以提供出色的音频质量（相当于音频光盘），如MP3播放器、便携式扬声器或无线耳机。

- 系统采样频率为 48 kHz (SAMPLERATE)
- 系统位数为16（BITWIDTH）
- 系统通道数为2（立体声）

基于这些数字，我们可以计算得出I²S协议比特率的频率等于：

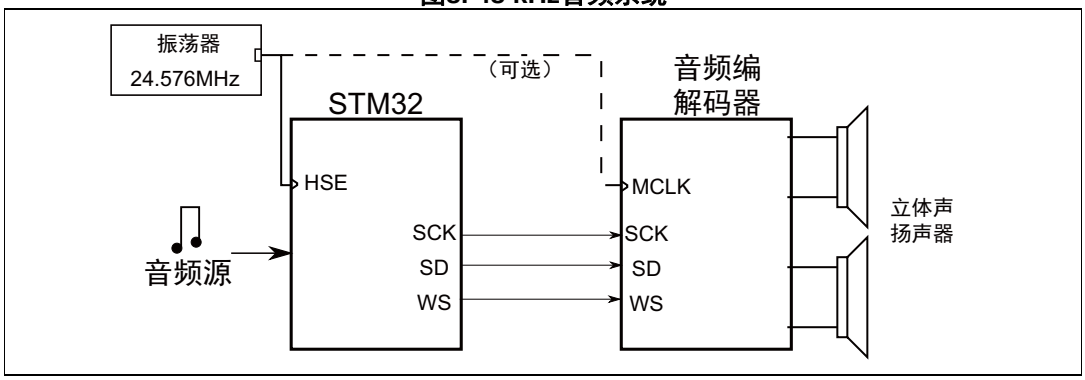
公式1- 计算I²S接口比特率

$$\text{I2S BITRATE} = \text{SAMPLE RATE} \times \text{WIDTH} \times \text{CHANNEL} = 48000 \times 16 \times 2 = 1.536 \text{ Mbit/s}$$

音频系统使用特定频率值，以满足现有标准（11 kHz、44.1 kHz、48 kHz等）。系统通常使用外部振荡器，以提供低抖动的精确时钟。

在这种特殊情况下（48kHz），我们可以从外部振荡器中选择MCLK=24.576MHz，其中，过采样率为16，采用I²S协议比特率。

图8. 48 kHz音频系统



2.2 8 kHz应用：无线AM质量

在低成本应用中，我们建议使用较低的I²S比特率和STM32内部频率振荡器，最大限度地降低系统的功耗和物料清单的成本。该应用使用I²S协议驱动D类音频放大器。

- 系统采样频率 8 kHz
- 系统位数为16（BITWIDTH）
- 系统通道数 = 2
- I²S比特率= 8 kHz x 16 x 2 = 256 Kbits / s

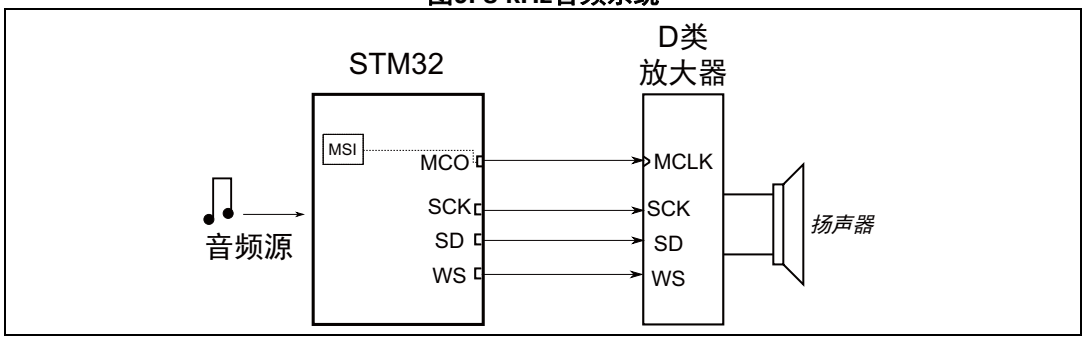
在本例中，基于STM32L0x，我们可以使用MSI（4.194MHz）或HSI（16MHz）振荡器，但二者均无法实现匹配256 kHz的倍数（I²S比特率）所需的精确和正确的系统时钟频率。另外一种替代方法是在系统中添加外部振荡器，这样会产生额外的成本。

公式2- 使用MSI作为时钟源计算频率不匹配

$$\text{MCLK/BITWIDTH} = \frac{4.194}{16} = 262.125\ \text{kHz (versus 256 kHz)}$$

MCLK = 4.194 MHz，得出SCK = 4.194 / 16 = 262.125 kHz，与256 kHz相比可知，绝对频率误差是2.3%。

图9. 8 kHz音频系统



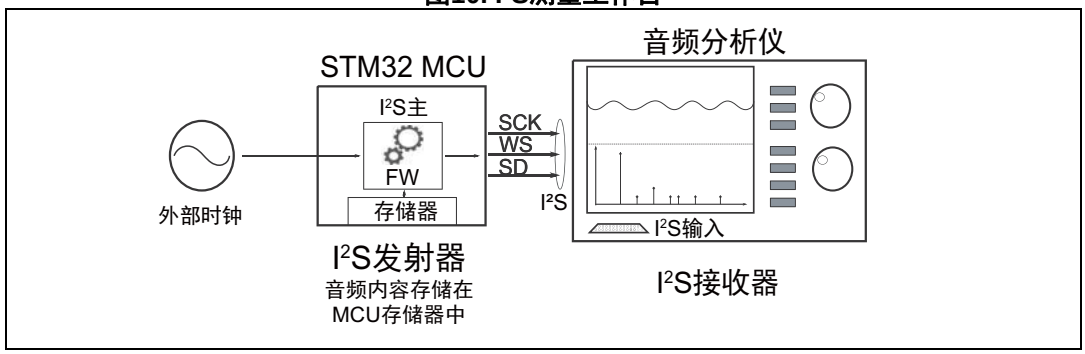
3 系统实现

3.1 带嵌入式存储器的系统实现

3.1.1 描述

为了验证仿真I²S主接口的行为是否正确，将该接口连接到带有参数I²S从接口的数字音频分析仪。首先，分析固定的常数值和存储的正弦波形（16位立体声）。其次，分析来自UART外设的相同音频信号。

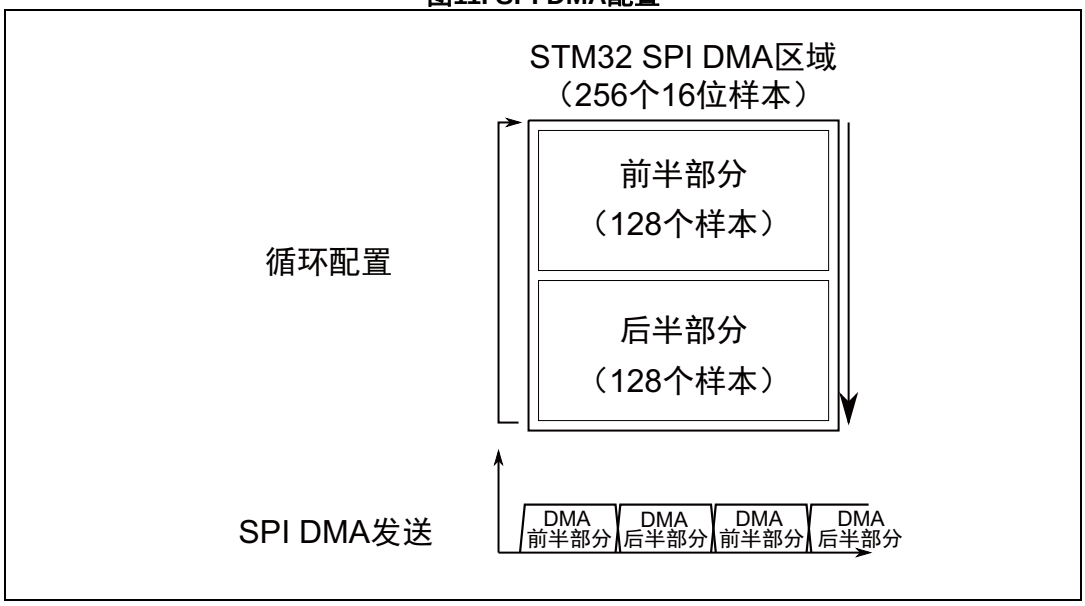
图10. I²S测量工作台



3.1.2 存储器机制实现

在SPI传输期间，以循环模式配置DMA。SPI外设持续读取缓冲的音频数据：

图11. SPI DMA配置



3.2 使用UART的系统实现

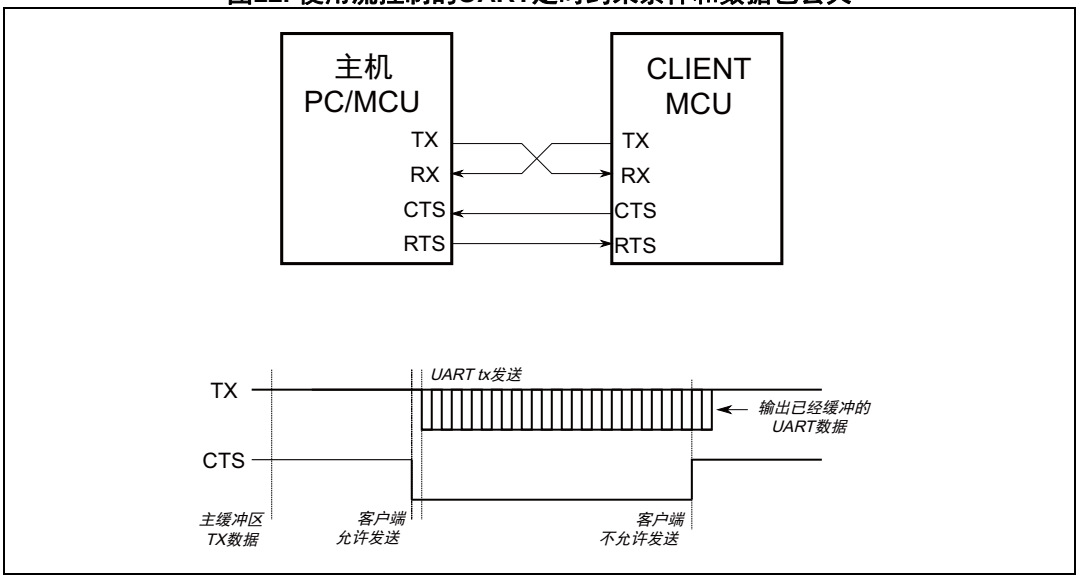
前面简单的测试平台特别适用于使用周期性正弦信号测量音频性能。它还可验证I²S仿真器的行为是否正确。标准音频应用程序的播放内容会有所不同；应在应用程序中添加一个输入接口，用于提供音频流。我们选择了非常通用的接口来传输音频内容，即，通用异步接收器发射器（UART）。大多数STM32微控制器系列都包含一个UART外设，可以按照本应用笔记描述的方法使用。

3.2.1 使用UART进行异步音频传输

该应用程序使用UART接口异步接收音频数字流。使用UART流控制来管理数据交换机制，避免任何数据包丢失。

UART接口（通用异步接收器发射器）是一个简单的2线标准接口。可额外使用两条线——RTS（请求发送）和CTS（清除发送）——来添加“流控制”特性，从而管理设备之间的数据包交换，避免任何数据丢失（不包括已缓冲的数据）。

图12. 使用流控制的UART定时约束条件和数据包丢失



在本例中，从UART异步接收的音频数据将传输到SPI外设，其配置为连续发送I²S协议。应实施特定且精确的机制，避免DMA交换期间的任何数据丢失。

3.2.2 固件实现

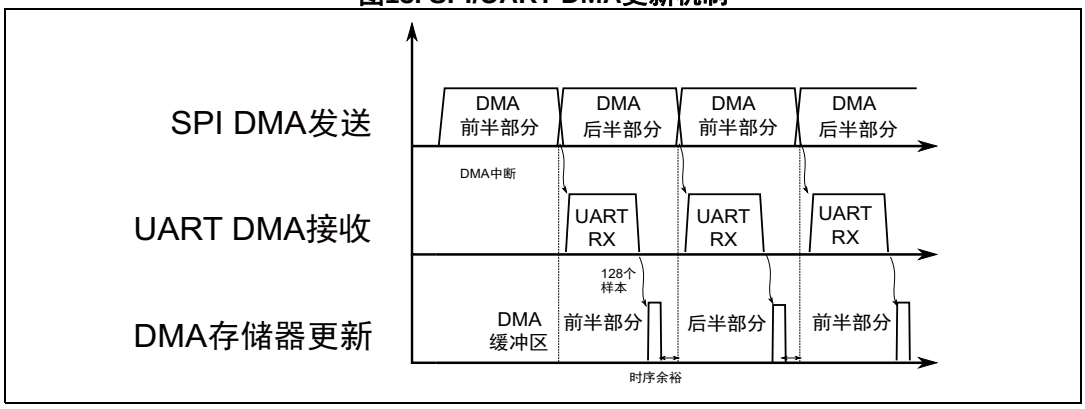
UART接收机制应根据以下原则交错执行DMA半块读取：

- 发生SPI DMA中断时，将调用UART接收函数。

在SPI DMA半缓冲区传输完成之前，应正确配置系统以检索所有数据（来自UART）。

此应用使用的比率为1.76：UART波特率为460800，SPI采样率为262.125 kHz

图13. SPI/UART DMA更新机制



此特定应用需要在SPI和UART缓冲区大小及其运行频率之间进行权衡。

使用大缓冲区：256个16位立体声样本（相当于1024个8位原始样本），提供足够的时间来调用、执行和完成UART接收调用功能。缺点

此外，本例基于STM32L031，允许最大系统时钟频率为32 MHz，SPI外设的最小倍频比为2。*表 3*提供最佳的系统配置。

表3. 带HSE时钟的I²S协议频率

I²S SCK频率（MHz）	SPI时钟预分频器	HSE频率（MHz）	注释
1.536	x16	24.576	最佳配置

在此应用中，必须在特定值24.576 MHz下使用来自HSE输入的外部时钟，使得I²S协议的工作频率为48 kHz。此外部振荡器可用作整个系统的主时钟源

表4. 带MSI时钟的I²S协议频率

MSI频率（kHz）	SPI时钟预分频器	I²S SCK频率（kHz）	注释
4194	x16	262.125	-

3.2.3 带流控制的UART外设参数

STM32 UART外设的硬件时钟SYSCLK具有优化的设置。微控制器时钟与UART外设的输出波特率之间的最小因子比为x8。

停止位 = 一个停止位

奇偶校验=无奇偶校验

波特率：460800波特

使能硬件流控（RTS和CTS信号）*/

UartHandle.Instance = USARTx;

UartHandle.Init.BaudRate = 460800;

UartHandle.Init.WordLength = UART_WORDLENGTH_8B;

UartHandle.Init.StopBits = UART_STOPBITS_1;

UartHandle.Init.Parity = UART_PARITY_NONE;

UartHandle.Init.HwFlowCtl = UART_HWCONTROL_RTS_CTS;

UartHandle.Init.Mode = UART_MODE_TX_RX;

3.2.4 主机侧：计算机接口使用Tera Term软件

主机配置应与STM32 UART外设设置保持一致。

图14. 主机串口配置

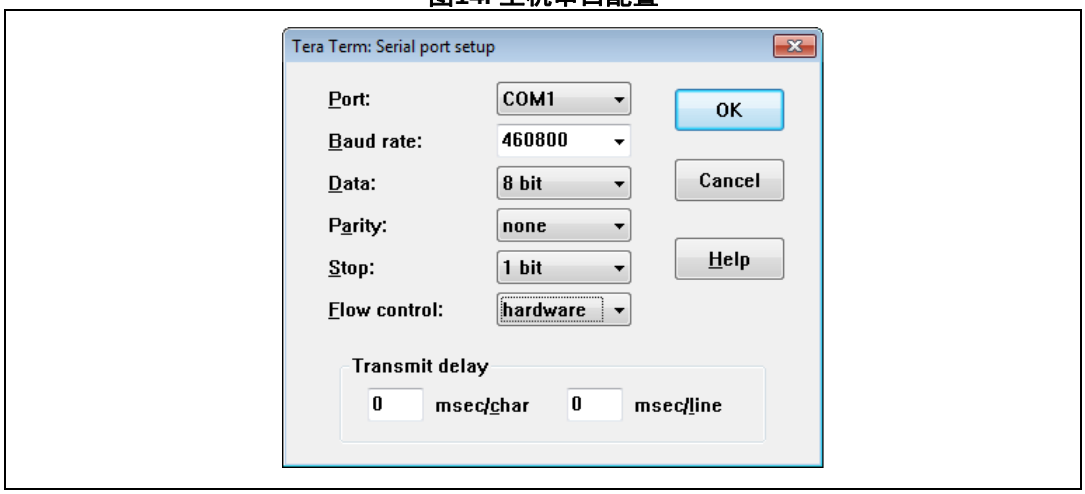
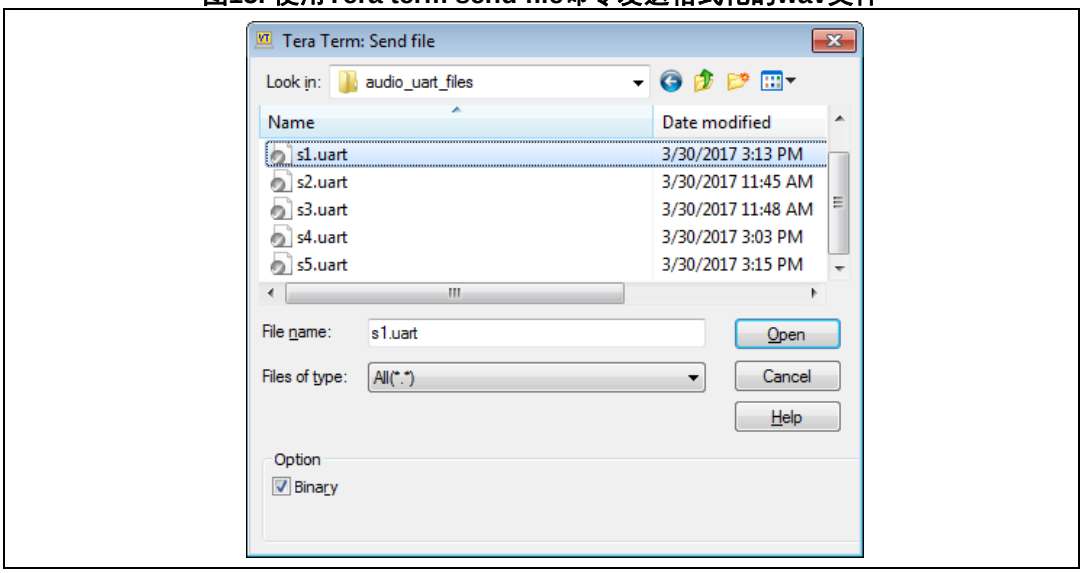


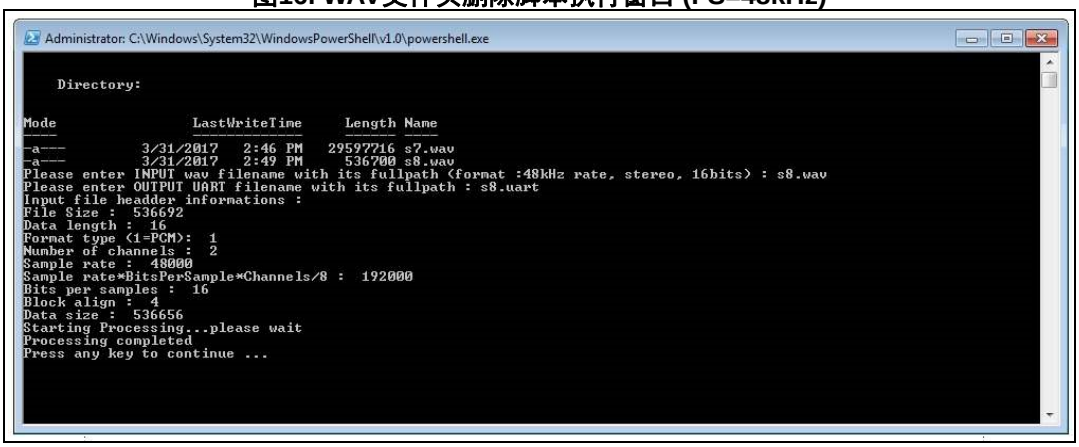
图15. 使用Tera term send file命令发送格式化的wav文件



3.2.5 WAV文件头删除脚本

文件标准波形音频文件格式（WAV）可以轻松地用作音频源。无论是用正弦音测试音频系统或是播放音乐，都很方便。WAV数据格式与I²S格式相同，唯一的缺点是需要删除不包含音频数据的文件头描述。可以使用一个简单的脚本完成此预处理，如*附录A：WAV格式至UART Powershell脚本*所示 。

图16. WAV文件头删除脚本执行窗口 (FS=48kHz)



3.2.6 MCU的UART接口

由于UART是双向接口，我们建议使用ASCII字符，将简单的图形用户界面（GUI）集成到MCU固件中，以便管理主机和客户端之间的音频数据交换。*图 17*显示了上述简单的界面显示。固件检测UART输入缓冲区状态（空闲、正在接收、传输完成），以管理I²S协议。

图17. 主机/客户端UART交换机制

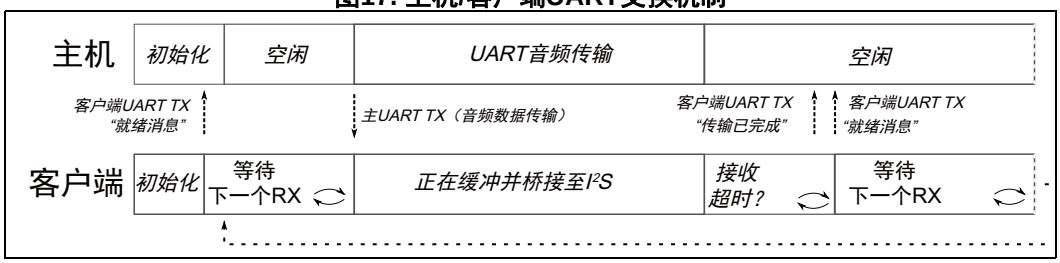
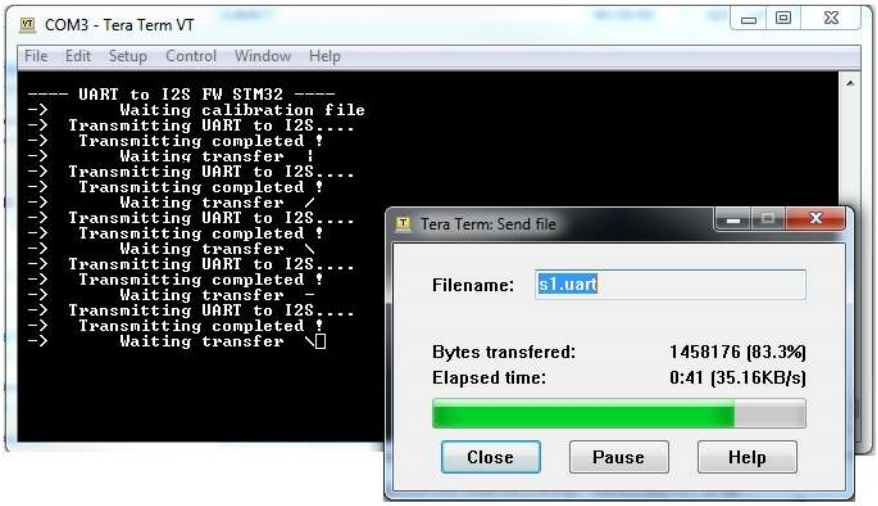


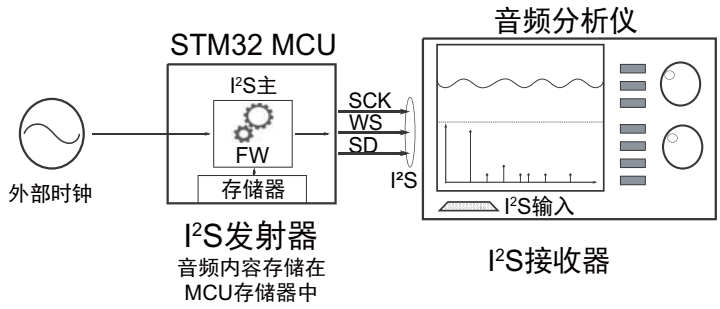
图18. 主机/客户端交换期间的客户端UART TX消息



4 使用I²S仿真外设的音频结果

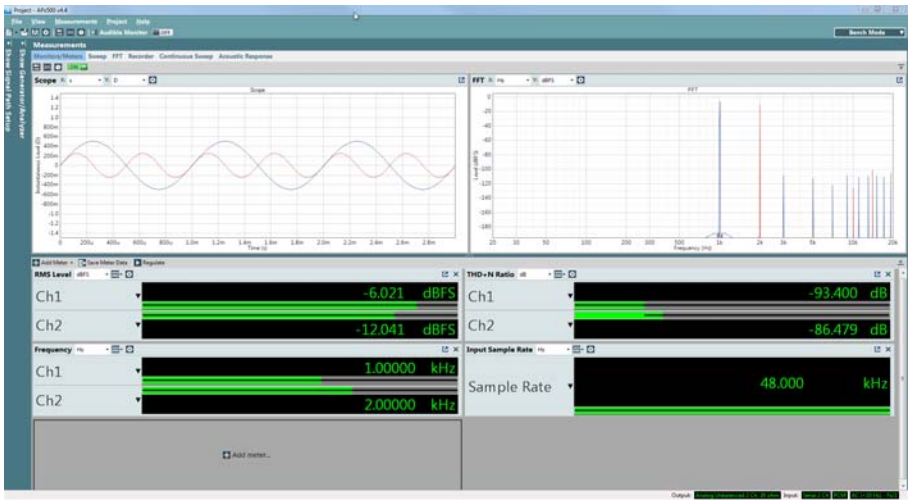
使用商用数字音频分析仪分析I²S音频内容，如*图 19* 所示。

图19. I²S测量工作台



使用音频分析仪及其接口执行音频测量，如*图 20* 所示。

图20. 音频测量接口（Audio Precision仪器）



4.1 音频常规测量

4.1.1 数字幅度（满刻度）

在音频数字域中，信号的幅度与满量程（FS）成比例，满量程（FS）代表最小和最大的信号幅度：在我们的应用中，分别对应0x0000（0 FS）和0x7FFF（1 FS）。如果我们以dBFS计算信号幅度，通常使用峰值幅度（而不是均方根）来计算dBFS等级。

公式1

- 数字正弦信号，峰值幅度等于16384 lsb
- 其在FS中的幅度为16384/32767 = 0.5 FS（其中，32767是最大信号等级）
- 我们可以将此值转换为分贝：20 x $\log_{10}$（0.5）= -6 dBFS

4.1.2 偏移

偏移测量值是输出信号的平均值（数字域）。通常，系统应在系统的中间量程范围内提供输出信号。在我们的应用程序中，I²S格式的中间量程值为0x0000（带符号的16位）。

4.1.3 频率响应

该测量定义了音频系统的频率范围。大多数情况下，1 kHz下的输出电平幅度被视作参照值，与测量幅度进行比较。我们用一系列频率测量输出信号的幅度（当FS = 48 kHz时，从20 Hz到20 kHz；或当FS = 8kHz时，从20 Hz到4 kHz），并与参照值进 行比较（1 kHz信号幅度），计算衰减值（单位dB）。具有平坦频率（+/- 1 dB）的音频系统不会变换音频信号及其内容，确保良好的音频呈现效果。

4.1.4 动态范围

根据最大输出电平与其噪声电平（最小输出信号）之间的比率（dB）定义动态范围测量。

4.1.5 总谐波失真

THD表示应用的线性性能。在此测试期间，对音频系统输入单频信号。从数学的角度而言，我们提取输入基波的谐波，计算主音与其失真（谐波）之间的比率（dB）或百分比。

16位舍去

由于使用了16位数据宽度，所生成的正弦信号含有不可忽略的失真水平，如*表 5*所示 。

表5. 理论正弦失真与数据位宽 (1 kHz 0 dBFS, FS = 48 kHz)

8 位	16 位	24 位	32 位
THD = -53 dB	THD = -99 dB	THD = -127 dB	THD = -200 dB

4.2 恒定音频内容

我们选择了以下参数来验证I²S仿真接口的完整性。存储器内容设置为常量/ DC值，以验证：

- 信号时序与I²S规范
- 毛刺检测：长期采集，用于检测信号
- 测量偏移量和满刻度

4.3 正弦音频内容

我们选择了以下参数来验证I²S仿真接口的完整性。

- 电平幅度（dBFS）
- 频率响应（Hz）
- 失真/THD（dB）

在分析通过I²S协议传输的音频信号之前，必须评估预期的性能。

利用内置数学函数计算正弦波形。得到一个双精度浮点数的数组。由于使用16位格式，因此必须执行舍去操作，降低所表示的正弦曲线的精度。此操作会导致结果信号出现失真（可见于*表 6* ）。

通过数学方法分析信号，可知，信号电平在-6dBFS 时，预期的失真水平约为-93 dB

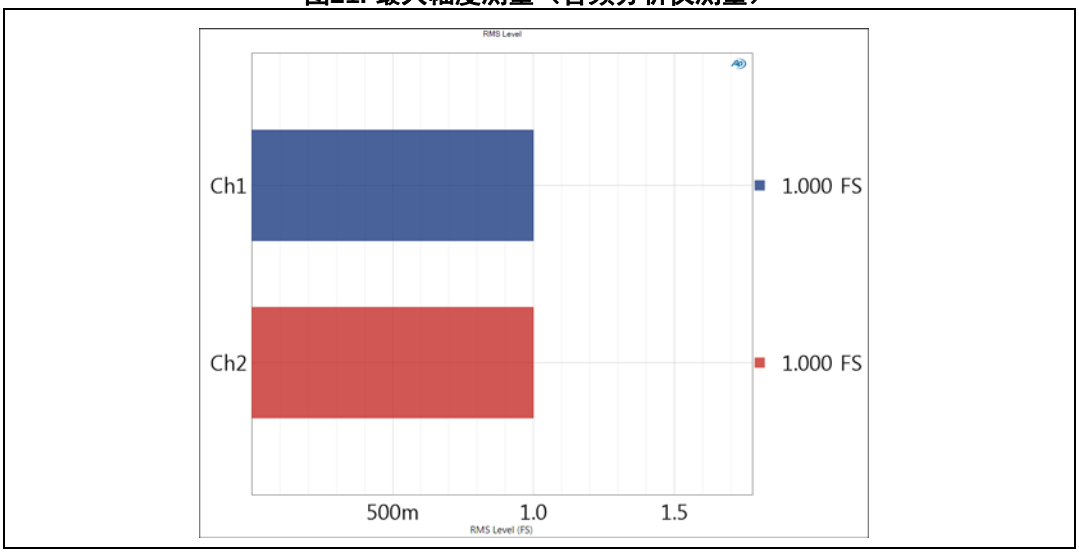
4.3.1 幅度和偏移测量

本节显示使用1 kHz频率的正弦信号，偏移和最大幅度的结果。

表6. DC偏移和最大幅度测量

DC偏移（FS）CH1和CH2	最大幅度（FS）CH1和CH2
0FS	1FS

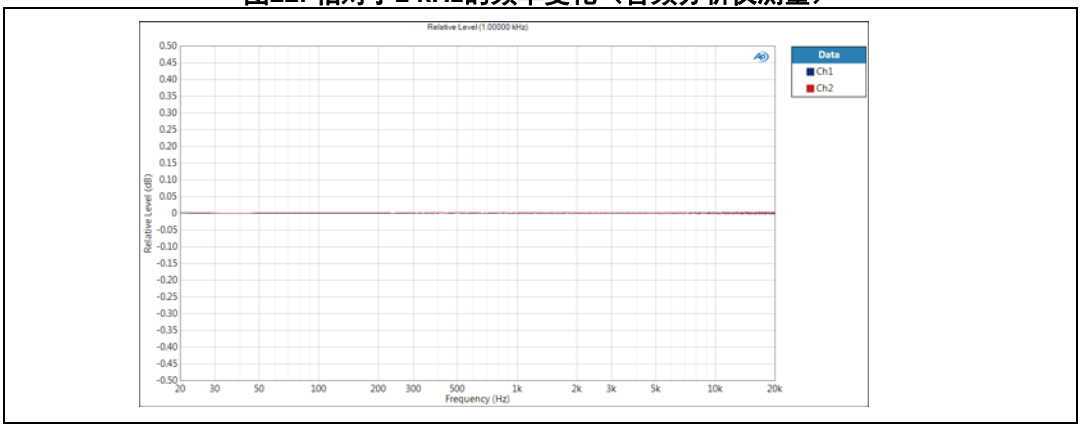
图21. 最大幅度测量（音频分析仪测量）



4.3.2 频率响应

本节显示在20 Hz至20 kHz的范围内，相对于1 kHz频率幅度的频率响应结果。

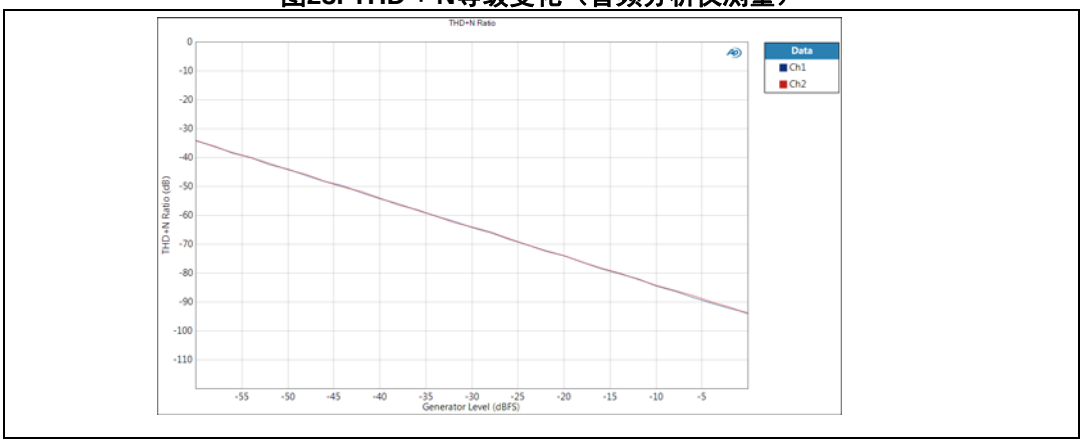
图22. 相对于1 kHz的频率变化（音频分析仪测量）



4.3.3 动态范围 (THD+N)

本节显示使用1 kHz正弦信号进行总谐波失真+噪声测量的结果，其中声音电平变化范围为-60 dBFS至0 dBFS。

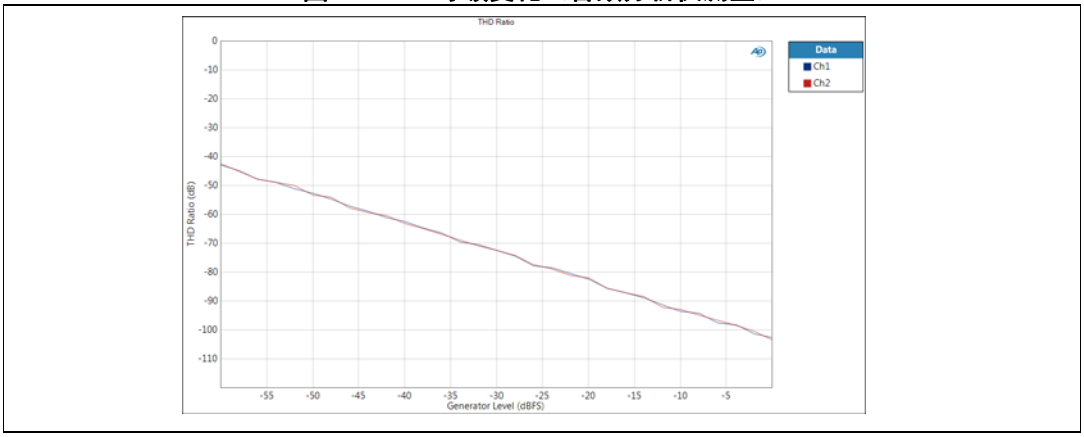
图23. THD + N等级变化（音频分析仪测量）



4.3.4 总谐波失真 (THD)

本节显示使用1 kHz正弦信号进行总谐波失真测量的结果，其中声音电平变化范围为-60 dBFS至0 dBFS。

图24. THD 等级变化（音频分析仪测量）



5 结论

本应用笔记介绍了将STM32外设组合在一起，进而创建新特性或高级特性的方法。如此一来，可以轻松地使用SPI和TIMER外设来执行I²S外设仿真。使用相同的方法，还可以仿真I²S从设备。

此外，STM32DMA外设可以确保外设之间实现高效的数据交换，可轻松在应用环境中创建桥接，如UART-to-I²S示例所示。

您可以借助本文档中共享的解决方案，开发或改进基于STM32微控制器系列的自有解决方案。

附录A WAV格式至UART Powershell脚本

```
dir *.wav 'Displaying the wav file list inside the folder'
$input_file = Read-Host 'Please enter INPUT wav filename with its fullpath (format: 48kHz rate, stereo, 16bits) '
$output_file = Read-Host 'Please enter OUTPUT UART filename with its fullpath '
$input_buffer = [System.IO.File]::ReadAllBytes($input_file)

$input_buffer_length=[math]::floor($input_buffer.length/256)
$output_buffer_length=($input_buffer_length)*(256+2);
$input_buffer_length=($input_buffer_length-1)*256;
Write-Host "Input file headder informations:"
Write-Host "File Size: "
($input_buffer[7]*16777216+$input_buffer[6]*65536+$input_buffer[5]*256+$input_buffer[4])
Write-Host "Data length: "
($input_buffer[19]*16777216+$input_buffer[18]*65536+$input_buffer[17]*256+$input_buffer[16])
Write-Host "Format type (1=PCM): " ($input_buffer[21]*256+$input_buffer[20])
Write-Host "Number of channels: " ($input_buffer[23]*256+$input_buffer[22])
Write-Host "Sample rate: "
($input_buffer[27]*16777216+$input_buffer[26]*65536+$input_buffer[25]*256+$input_buffer[24])
Write-Host "Sample rate*BitsPerSample*Channels/8: "
($input_buffer[31]*16777216+$input_buffer[30]*65536+$input_buffer[29]*256+$input_buffer[28])
Write-Host "Bits per samples: " ($input_buffer[35]*256+$input_buffer[34])
Write-Host "Block align: " ($input_buffer[33]*256+$input_buffer[32])
Write-Host "Data size: "
($input_buffer[43]*16777216+$input_buffer[42]*65536+$input_buffer[41]*256+$input_buffer[40])
$output_buffer=@(0) * ($output_buffer_length)
$j=1;$uart_frame_cnt=511;
$riff_header=44;
Write-Host "Starting Processing...please wait"
for($i=0; $i -le ($input_buffer_length); $i++)
{
 $output_buffer[$i+$j]= $input_buffer[$i+$riff_header];
  if($uart_frame_cnt -eq 0)
 {
 $uart_frame_cnt=511;
 $j=$j+2;
 }
 else
 {$uart_frame_cnt=$uart_frame_cnt-1;}
```

```
}
Write-Host "Processing completed"
[io.file]::WriteAllBytes($output_file,$output_buffer) 'Output buffer writing
Write-Host "Press any key to continue ..."
$x = $host.UI.RawUI.ReadKey("NoEcho,IncludeKeyDown")
```

6 版本历史

表7. 文档版本历史

日期	版本	变更
2017年12月11日	1	初始版本。

表8. 中文文档版本历史

日期	版本	变更
2019年3月12日	1	中文初始版本。

重要通知 - 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。本文档的中文版本为英文版本的翻译件，仅供参考之用；若中文版本与英文版本有任何冲突或不一致，则以英文版本为准。