UM2372
用户手册

STM32F4/F7/H7系列的 STM32Cube PDM2PCM软件库

引言

PDM2PCM库将来自MEMS麦克风的PDM位流转换为PCM音频流。

本用户手册介绍了PDM2PCM库，它是STM32Cube固件包的一部分。它提供了有关接口参数和库配置的详细信息。它还显示了如何将此库集成到主程序中。

本文档适用于允许用户连接数字PDM麦克风的微控制器，即STM32F4、STM32F7和STM32H7系列。

目录

1 模块概述 5

1.1 算法功能 5

1.2 模块配置 5

1.3 资源总结 6

2 模块接口 7

2.1 API 7

2.1.1 PDM_FilterInit 函数 7

2.1.2 PDM_Filter_setConfig 函数 7

2.1.3 PDM_Filter_getConfig 函数 8

2.1.4 PDM_Filter_deInterleave 函数 8

2.1.5 PDM_Filter 函数 8

2.2 外部定义 9

2.2.1 返回的错误值 9

2.3 静态参数结构 9

2.4 动态参数结构 10

3 算法描述 11

3.1 处理步骤 11

3.1.1 高通滤波器 12

3.1.2 数字音量 13

3.2 数据格式 13

3.3 测量结果 13

3.3.1 失真测量 13

3.3.2 语音信号 16

4 应用描述 17

4.1 模块集成示例 17

4.1.1 库初始化 17

4.1.2 模块执行 17

4.1.3 模块API调用 18

5 版本历史 20

表格索引

表1. 封装 . . . 6
表2. PDM_FilterInit 函数 . . . 7
表3. PDM_Filter_setConfig 函数 . . . 8
表4. PDM_Filter_getConfig 函数 . . . 8
表5. PDM_Filter 函数 . . . 8
表6. 错误值 . . . 9
表7. 静态参数 . . . 9
表8. 动态参数 . . . 10
表9. 抽取因子和相应频率 . . . 11
表10. 300 Hz下的失真测量 . . . 14
表11. 500 Hz下的失真测量 . . . 14
表12. 1000 Hz下的失真测量 . . . 15
表13. PDM2PCM库输出的平均语音信号电平 . . . 16
表14. 文档版本历史 . . . 20
表15. 中文文档版本历史 . . . 20

图片索引

图1. 框图. 11
图2. 高通滤波器的频率响应 12
图3. 300 Hz下的失真测量. 13
图4. 500 Hz下的失真测量. 14
图5. 1000 Hz下的失真测量. 15
图6. 语音电平比较 16
图7. 模块流程图 19

1 模块概述

1.1 算法功能

PDM2PCM库具有从数字麦克风抽取和滤除脉冲密度调制（PDM）流的功能，用于将其转换为脉冲编码调制（PCM）信号输出流。

PCM输出流以16位分辨率实现。接口中未指定采样率，但本文档中约定使用16 kHz的PCM采样率。可配置各种抽取因子，以适应各种PDM时钟。

还推荐可配置的高通滤波器和数字音量。

1.2 模块配置

PDM2PCM库将1位数字样本的PDM信号（768 kHz至2.048 MHz）流作为输入。通过使用基于Arm®内核(a)的STM32微控制器同步串行端口（SPI或I2S）以8个样本的数据块为单位来采集该信号。

根据内核和使用的工具链，该模块有不同的可用版本。

arm

a. Arm是Arm Limited（或其子公司）在美国和/或其他地区的注册商标。

1.3 资源总结

*表 1*包含对存储器和频率的要求。

使用IAR Embedded Workbench® for Arm® v7.40 (IAR Embedded Workbench® common components v7.2)测量板上的占用量。

表1. 封装

PDM时钟	Flash代码 .text（字节）	Flash数据 .rodata（字节）	堆栈（字节）	RAM（字节）	频率（MHz）
2.048 MHz（抽取 = 128）	7020	789	50	1028	4.9
1.280 MHz（抽取 = 80）					3.4
1.024 MHz（抽取 = 64）					2.7
768 kHz（抽取 = 48）					2.3
512 kHz（抽取 = 32）					1.8

2 模块接口

需要使用两个文件来集成PDM2PCM库，pdm2pcm_glo.h头文件和正确的库文件（根据目标和工具链）。

它们包含所有要导出到软件集成框架的定义和结构。

注： *audio_fw_glo.h文件是通用头文件，它对于所有音频模块都通用且必须包含在音频框架中。*

2.1 API

五个函数具有主程序的软件接口：

- *PDM_FilterInit*
- *PDM_Filter_setConfig*
- *PDM_Filter_getConfig*
- *PDM_Filter_deInterleave*
- *PDM_Filter*

2.1.1 *PDM_FilterInit* 函数

该程序初始化静态存储器，设置默认值并初始化PDM2PCM库的查找表。

uint32_t PDM_FilterInit(PDM_Filter_Handler_t *pHandler);

表2. *PDM_FilterInit* 函数

I/O	名称	类型	说明
输入	*pHandler*	*PDM_Filter_Handler_t **	指向内部静态存储器的指针
返回值	-	*uint32_t*	错误值

当实时处理尚未开始时，在初始化阶段，必须至少调用一次该例程。

2.1.2 PDM_Filter_setConfig 函数

该例程从主框架为模块内存设置模块动态参数。可以在处理期间的任何时间对其进行调用。

uint32_t PDM_Filter_setConfig(PDM_Filter_Handler_t *pHandler, PDM_Filter_Config_t *pConfig);

表3. *PDM_Filter_setConfig* 函数

I/O	名称	类型	说明
输入	*pHandler*	*PDM_Filter_Handler_t **	指向内部静态存储器的指针
输入	pConfig	PDM_Filter_Config_t*	指向动态参数结构的指针
返回值	-	*uint32_t*	错误值

2.1.3 *PDM_Filter_getConfig* 函数

该程序从主框架的内部静态存储器获取模块动态参数。可以在处理期间的任何时间对其进行调用。

uint32_t PDM_Filter_getConfig(PDM_Filter_Handler_t *pHandler, PDM_Filter_Config_t *pConfig);

表4. *PDM_Filter_getConfig* 函数

I/O	名称	类型	说明
输入	*pHandler*	*PDM_Filter_Handler_t **	指向内部静态存储器的指针
输入	pConfig	PDM_Filter_Config_t*	指向动态参数结构的指针
返回值	-	*uint32_t*	错误值

2.1.4 *PDM_Filter_deInterleave* 函数

尚未实现。

2.1.5 *PDM_Filter* 函数

该程序将输入PDM流解码为输出PCM流。必须通过调用它来处理每个帧。

uint32_t PDM_Filter(void *pDataIn, void *pDataOut, PDM_Filter_Handler_t * pHandler);

表5. *PDM_Filter* 函数

I/O	名称	类型	说明
输入	*pDataIn*	*void **	指向PDM输入数据的指针
输出	*pDataOut*	*void **	指向PCM输出数据的指针
输入	*pHandler*	*PDM_Filter_Handler_t**	指向内部静态存储器的指针
返回值	-	*uint32_t*	错误值

2.2 外部定义

2.2.1 返回的错误值

*表 6*列出了可能返回的错误值。每个错误都将专用位设为1，因此可以累积多个错误代码。

表6. 错误值

定义	值	说明
PDM_FILTER_NO_ERROR	0x0000	无错误
PDM_FILTER_ENDIANNESS_ERROR	0x0001	不支持的字节序
PDM_FILTER_BIT_ORDER_ERROR	0x0002	不支持的位序
PDM_FILTER_CRC_LOCK_ERROR	0x0004	目标不是STM32
PDM_FILTER_DECIMATION_ERROR	0x0008	不支持的抽取因子
PDM_FILTER_INIT_ERROR	0x0010	-
PDM_FILTER_CONFIG_ERROR	0x0020	-
PDM_FILTER_GAIN_ERROR	0x0040	不支持的麦克风增益
PDM_FILTER_SAMPLES_NUMBER_ERROR	0x0080	不支持的样本数

2.3 静态参数结构

在调用*PDM_Filter_setConfig()*函数之前，使用相应的静态参数结构设置PDM2PCM初始参数。

```
typedef struct {
  uint16_t bit_order;
  uint16_t endianness;
  uint32_t high_pass_tap;
  uint16_t in_ptr_channels;
  uint16_t out_ptr_channels;
  uint32_t pInternalMemory[INTERNAL_MEMORY_SIZE];
}PDM_Filter_Handler_t;
```

表7. 静态参数

名称	说明	备注
bit_order	指定输入的位序 （MSB或LSB）	PDM_FILTER_BIT_ORDER_LSB (0x0000) PDM_FILTER_BIT_ORDER_MSB (0x0001)
字节序	指定是否需要字节反转	PDM_FILTER_ENDIANNESS_LE (0x0000) PDM_FILTER_ENDIANNESS_BE (0x0001)
high_pass_tap	指定高通滤波器抽头值	高通滤波器的Q31格式系数值。 如果为0，则不使用滤波器。
in_ptr_channels	指定输入PDM流中的通道数	整数 > 0 与一个麦克风一起使用时，in_ptr_channels = 1。

表7. 静态参数（续）

名称	说明	备注
out_ptr_channels	指定输出PCM流中的通道数。	整数 > 0。 与一个麦克风一起使用时，out_ptr_channels=1。
pInternalMemory	内部存储器	指向阵列的指针。

2.4 动态参数结构

在调用*PDM_Filter_setConfig()*函数之前，通过在动态参数结构中设置新值，可更改PDM2PCM配置。

```
typedef struct {
  uint16_t decimation_factor;
  uint16_t output_samples_number;
  int16_t  mic_gain;
}PDM_Filter_Config_t;
```

表8. 动态参数

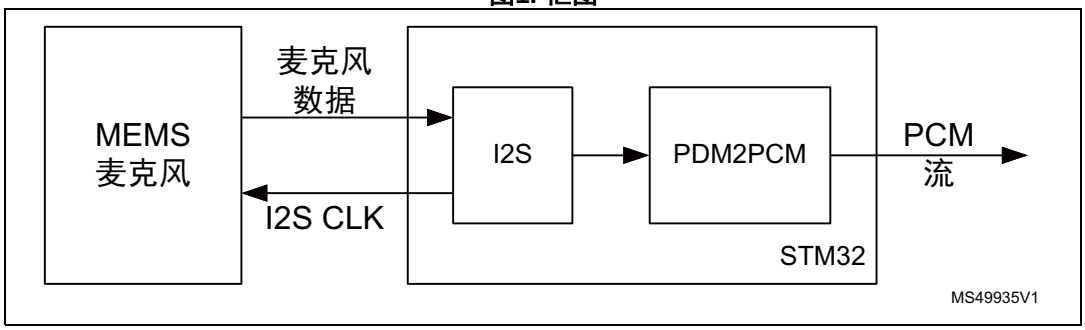
名称	说明	备注
decimation_factor	指定抽取因子。	PDM_FILTER_DEC_FACTOR_16 (0x0005) PDM_FILTER_DEC_FACTOR_24 (0x0006) PDM_FILTER_DEC_FACTOR_32 (0x0007) PDM_FILTER_DEC_FACTOR_48 (0x0001) PDM_FILTER_DEC_FACTOR_64 (0x0002) PDM_FILTER_DEC_FACTOR_80 (0x0003) PDM_FILTER_DEC_FACTOR_128 (0x0004)
output_samples_number	指定每次调用*PDM_Filter()*函数时要生成的PCM样本数量	整数 > 0
mic_gain	指定麦克风增益（以dB为单位）	增益在[-12 dB: +51 dB]的区间内，步长为1 dB。

3 算法描述

3.1 处理步骤

如*图 1*中所示，MEMS麦克风输出PDM流，它是1位数字采样的高频数据流。该库需要一个由8个样本块（一个字节）组成的数据流，将使用STM32微控制器的同步串行端口（SPI或I2S）获取该数据流。麦克风PDM输出与其输入时钟同步，因此使用的STM32串行端口产生麦克风的时钟信号。

图1. 框图



麦克风的PDM数据以8位块打包，然后进行滤波和抽取。获取的PCM信号的频率取决于库初始化之前配置的抽取因子。

已定义抽取因子，以获取所需采样频率的PCM流，该频率取决于PDM时钟值。*表 9*中给出了示例。

表9. 抽取因子和相应频率

抽取因子	PDM 时钟频率	PCM采样率
128	1.024 MHz	8 kHz
	2.048 MHz	16 kHz
	3.072 MHz	24 kHz
80	1.280 MHz	16 kHz
64	1.024 MHz	16 kHz
	1 MHz	32 kHz
	3.072 MHz	48 kHz
48	768 kHz	16 kHz
32	512 kHz	16 kHz
24	384 kHz	16 kHz
16	256 kHz	16 kHz

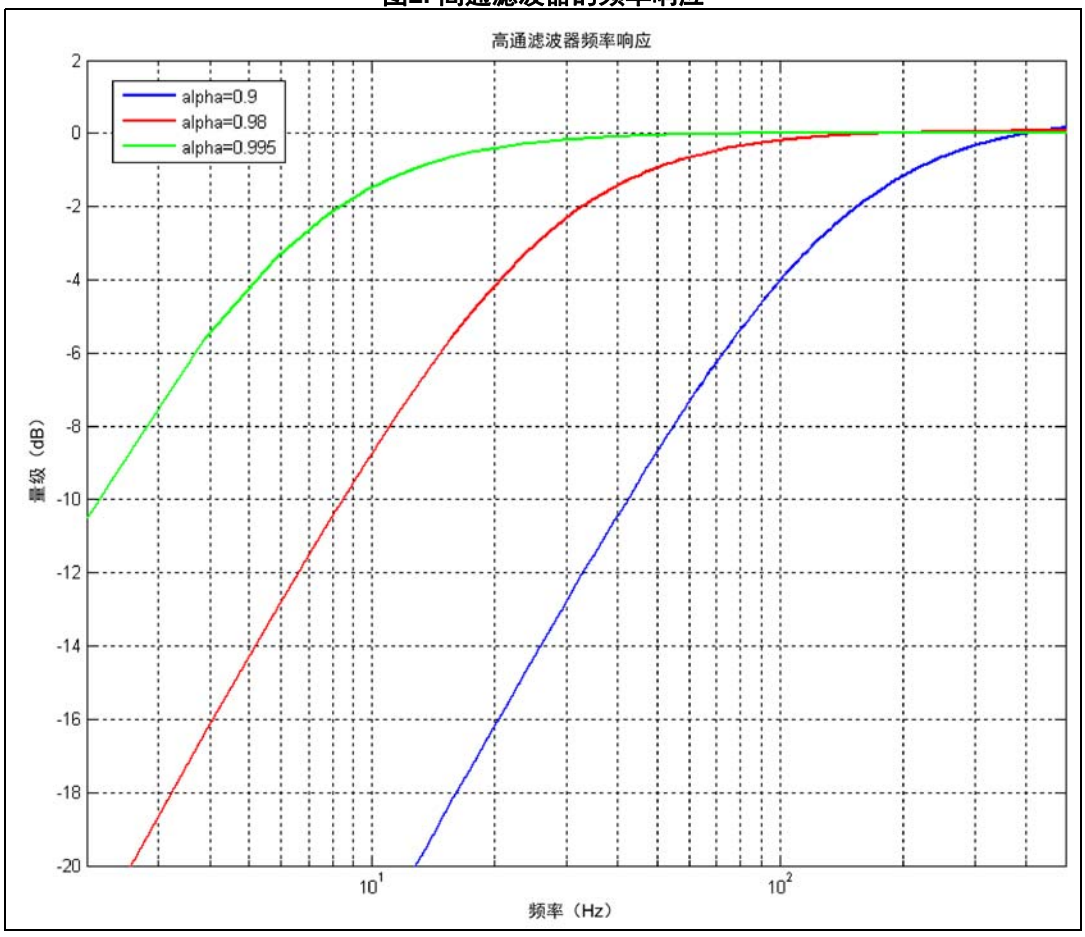
然后，滤波器和抽取器流水线所产生的数字信号经高通滤波器处理，以消除DC偏移，随后经数字增益处理，以衰减或放大PCM采样值。

3.1.1 高通滤波器

高通滤波器是单极递归滤波器。通过修改*PDM_Filter_Handler_t*的*high_pass_tap*参数来配置截止频率。系数值必须在[0 : 1]的范围内。使用的格式为Q0.31，意味着1对应于可通过31位分辨率获得的最大整数值。例如，将*high_pass_tap*参数配置为0.98对应于$0.98*(2^{31}-1) = 2104533974$。

*图 2*是在16 kHz的PCM采样率下，该滤波器三个不同的*high_pass_tap*参数值的频率响应图。

图2. 高通滤波器的频率响应



当将系数设为0时，旁路高通滤波器。

3.1.2 数字音量

在将其饱和为带符号的16位值之前，数字音量衰减或放大样本。*mic_gain*参数是应用于PCM流的增益值（以dB为单位）。最小值为-12 dB，最大值为51 dB，步长为1 dB。

3.2 数据格式

PDM2PCM库的输入预计为在MEMS麦克风时钟频率下以字节打包的PDM流。它可能为单数据流或双数据流。输出为PCM流。

3.3 测量结果

所有测量均使用STM32F469板进行，其上面安装有MEMS麦克风MP34DT01。它们在消声环境中进行，并将专业的监控系统作为声源。

3.3.1 失真测量

在原点处以大约90 dB SPL的标称声级测试信号，以进行失真测量。麦克风布置在相距10 cm的位置，*mic_gain*等于0 dB。

这些数据考虑了所有系统噪声，包括PCB和电源带回的底噪。

图3. 300 Hz下的失真测量

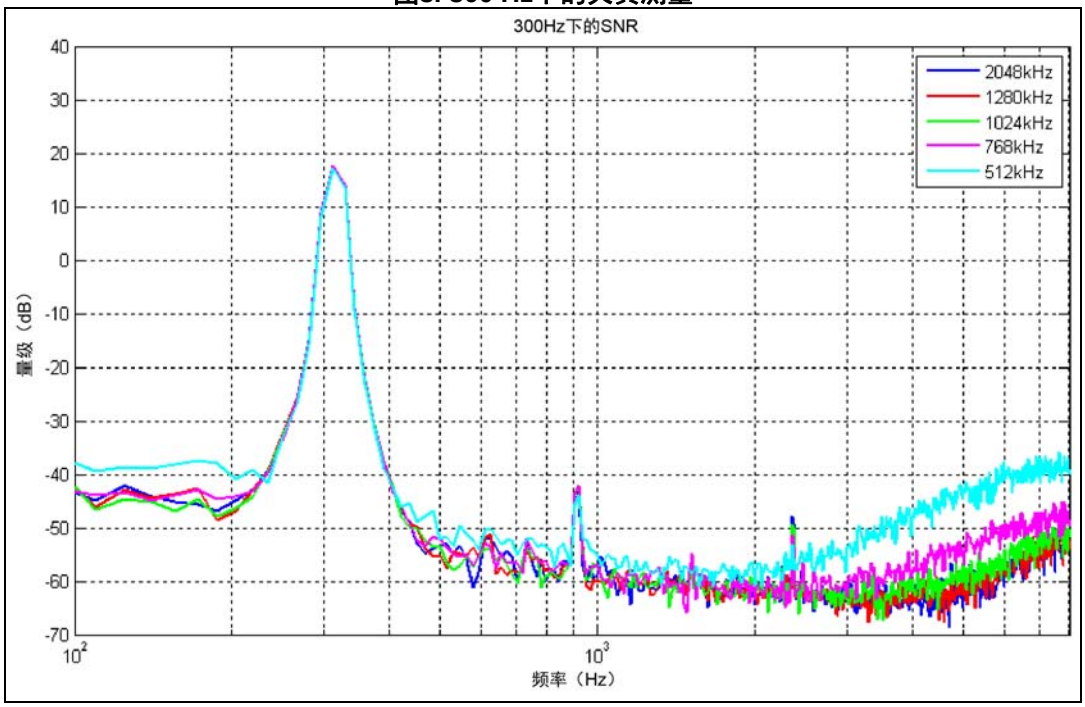


表10. 300 Hz下的失真测量

PDM时钟	-31 dBFS下的SNR	0 dBFS下的SNR（推断）
2048 kHz	42.8	73.8
1280 kHz	42.7	73.7
1024 kHz	42.0	73.0
768 kHz	39.8	72.8
512 kHz	32.1	63.1

图4. 500 Hz下的失真测量

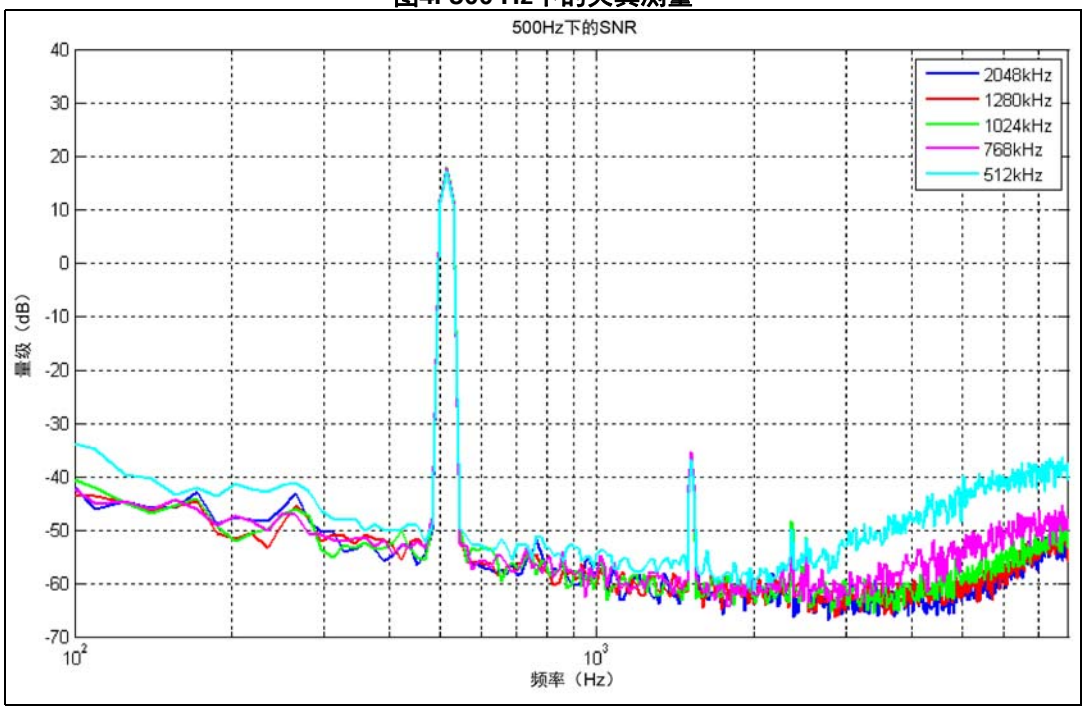


表11. 500 Hz下的失真测量

PDM时钟	-31 dBFS下的SNR	0 dBFS下的SNR（推断）
2048 kHz	43.1	74.1
1280 kHz	42.1	73.1
1024 kHz	42.1	73.1
768 kHz	40.1	71.1
512 kHz	29.8	60.8

图5. 1000 Hz下的失真测量

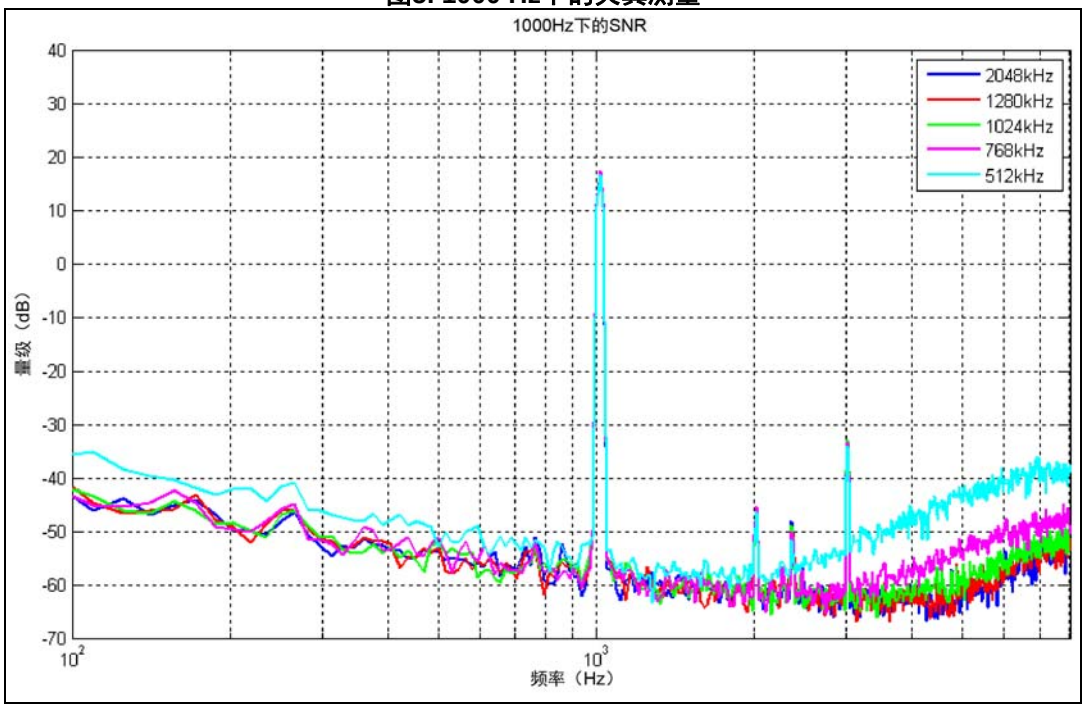


表12. 1000 Hz下的失真测量

PDM时钟	-31 dBFS下的SNR	0 dBFS下的SNR（推断）
2048 kHz	38.7	69.7
1280 kHz	38.5	69.5
1024 kHz	38.4	69.4
768 kHz	37.1	68.1
512 kHz	31.2	62.2

3.3.2 语音信号

测试信号是以90 dB SPL的标称电平播放的语音序列。MEMS麦克风放在离声源30 cm远的地方。在PDM2PCM库的输出端捕获的不同*mic_gain*值下的数字信号如*图 6*中所示。

图6. 语音电平比较

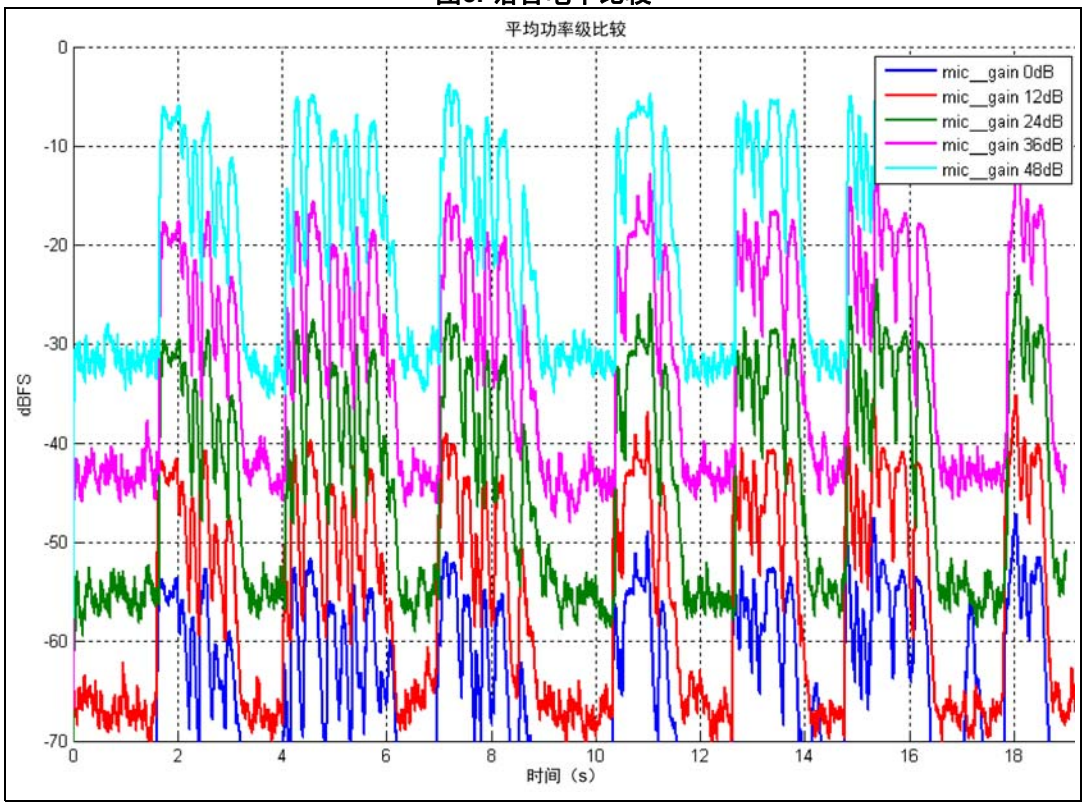


表13. PDM2PCM库输出的平均语音信号电平

*mic_gain*参数值	数字语音电平
0 dB	-54 dBFS
12 dB	-42 dBFS
24 dB	-30 dBFS
36 dB	-18 dBFS
48 dB	-8 dBFS

4 应用描述

4.1 模块集成示例

4.1.1 库初始化

在分配内存后，必须通过调用一些程序来初始化PDM2PCM库静态存储器：

- 每次停止和启动音频处理时，必须调用*PDM_Filter_Init()*。
- 在开始处理前必须至少调用一次*PDM_Filter_setConfig()*，以设置可配置参数

此外，由于PDM2PCM库在STM32设备上运行，因此必须使能并重置CRC HW块。

必须分配静态和动态参数结构。在pdm2pcm_glo.h头文件中定义其类型。分配示例：

```
/*从STM32 HW使能和重置CRC-32 */
  __HAL_RCC_CRC_CLK_ENABLE();
  CRC->CR = CRC_CR_RESET;

  PDM_Filter_Handler_t  PDM1_filter_handler;
  PDM_Filter_Config_t   PDM1_filter_config;

  /* 初始化PDM滤波器结构 */
  PDM1_filter_handler.bit_order = PDM_FILTER_BIT_ORDER_LSB;
  PDM1_filter_handler.endianness = PDM_FILTER_ENDIANNESS_BE;
  PDM1_filter_handler.high_pass_tap = 2122358088;
  PDM1_filter_handler.out_ptr_channels = 1;
  PDM1_filter_handler.in_ptr_channels  = 1;
  PDM_Filter_Init((PDM_Filter_Handler_t *)(&PDM1_filter_handler));

  PDM1_filter_config.output_samples_number = 16;
  PDM1_filter_config.mic_gain = 24;
  PDM1_filter_config.decimation_factor = PDM_FILTER_DEC_FACTOR_64;
  PDM_Filter_setConfig((PDM_Filter_Handler_t *)&PDM1_filter_handler, &PDM1_filter_config);
```

4.1.2 模块执行

运行时进程可在已配置硬件且已初始化和配置PDM2PCM库时启动。

在每个新的中断中，当缓冲了足够的位时，可以调用PDM2PCM滤波器例程。在该滤波器程序的两次连续调用期间，可更改动态参数。

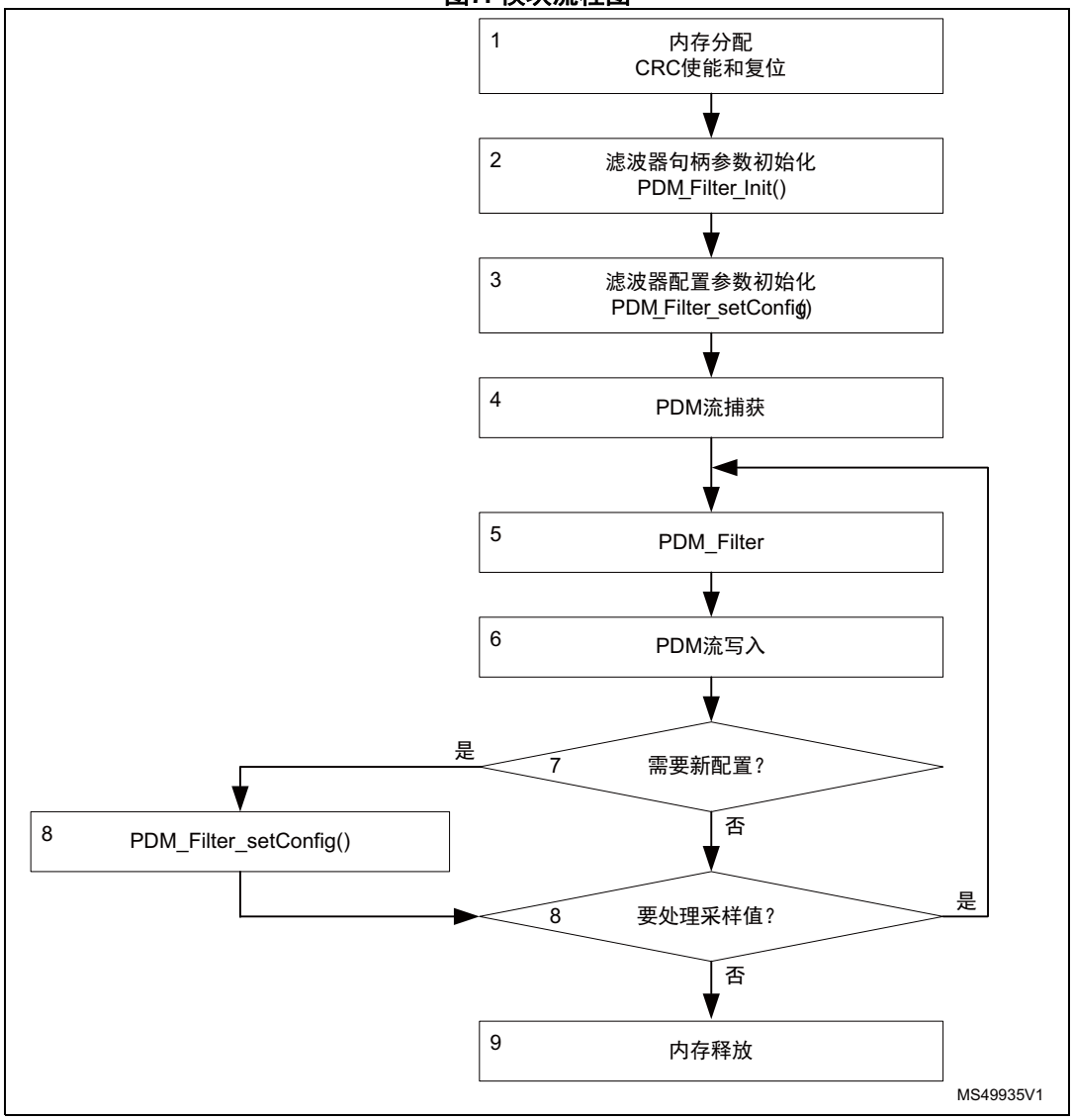
```
do
{
  /* 处理当前帧 */
  PDM_Filter(&pdm_buffer[0], &pcm_buffer[0], &PDM1_filter_handler);

                :
                :
  /* 改变音量设置 */
  PDM1_filter_config.mic_gain = 12;
  PDM_Filter_setConfig((PDM_Filter_Handler_t *)&PDM1_filter_handler, &PDM1_filter_config);
}
```

4.1.3 模块API调用

流程如*图 7*中所示，步骤如下：

1. 如上文所述，必须分配PDM2PCM句柄和配置结构以及PDM输入和PCM输出缓冲区。必须使能CRC才能解锁库。
2. 现在必须将PDM滤波器句柄参数设为所需值，并调用*PDM_Filter_Init()*函数。
3. 现在必须将PDM滤波器配置参数设为所需值，并调用*PDM_Filter_setConfig()*函数。
4. 从合适的接口读取逐字节打包的PDM输入位流。
5. 调用*PDM_Filter() function*函数将执行PDM2PCM算法。
6. 此时，可以在合适的接口中写入PCM输出音频流。
7. 如果需要，用户可以更改配置参数并调用*PDM_Filter_setConfig()*函数，以更新库配置。
8. 如果应用程序和PDM输入流仍在运行，则处理循环返回到第5步，否则将结束。
9. 一旦处理循环结束，必须释放分配的存储空间。

图7. 模块流程图


1
内存分配
CRC使能和复位
2
滤波器句柄参数初始化
PDM_Filter_Init()
3
滤波器配置参数初始化
PDM_Filter_setConfig()
4
PDM流捕获
5
PDM_Filter
6
PDM流写入
7
需要新配置？
是
否
8
PDM_Filter_setConfig()
8
要处理采样值？
是
否
9
内存释放
MS49935V1

5 版本历史

表14. 文档版本历史

日期	版本	变化说明
2018年7月6日	1	初始版本

表15. 中文文档版本历史

日期	版本	变化说明
2019年3月19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。