UM2372
User manual

STM32Cube PDM2PCM software library for the STM32F4/F7/H7 Series

Introduction

The PDM2PCM library converts a PDM bit stream from a MEMS microphone into a PCM audio stream.

This user manual describes the PDM2PCM library, which is part of the STM32Cube firmware package. It provides details about the interface parameters and the configuration of the library. It also shows how to integrate this library into a main program.

This document is applicable to the microcontrollers that allow the user to connect a digital PDM microphone, namely those of STM32F4, STM32F7 and STM32H7 Series.

Contents

List of tables

List of figures

1 Module overview

1.1 Algorithm functionality

The PDM2PCM library has the function to decimate and filter out a Pulse Density Modulated (PDM) stream from a digital microphone, to convert it to a Pulse Code Modulated (PCM) signal output stream.

The PCM output stream is implemented with 16-bit resolution. The sampling rate is not specified in the interface but it is agreed in this document that the PCM sampling rate used is 16 kHz. Various decimation factors can be configured, to adapt to various PDM clocks.

A configurable high-pass filter and a digital volume are also proposed.

1.2 Module configuration

PDM2PCM library takes as input a PDM signal (768 kHz to 2.048 MHz) stream of 1-bit digital samples. This signal is acquired in blocks of 8 samples by using a synchronous serial port (SPI or I2S) of the STM32 microcontroller, based on Arm® cores[a].

Different versions of the module are available, depending upon the core and the used tool chain.

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

1.3 Summary of resources

Table 1 contains the requirements for memories and frequency.

The footprints are measured on board, using IAR Embedded Workbench® for Arm® v7.40 (IAR Embedded Workbench® common components v7.2).

Table 1. Footprints

PDM clock	Flash code .text (bytes)	Flash data .rodata (bytes)	Stack (bytes)	RAM (bytes)	Frequency (MHz)
2.048 MHz (decimation = 128)	7020	789	50	1028	4.9
1.280 MHz (decimation = 80)					3.4
1.024 MHz (decimation = 64)					2.7
768 kHz (decimation = 48)					2.3
512 kHz (decimation = 32)					1.8

2 Module interfaces

Two files are needed to integrate the PDM2PCM library, the pdm2pcm_glo.h header file and the right library file (according to target and tool chain).

They contain all definitions and structures to be exported to the software integration framework.

Note: The audio_fw_glo.h file is a generic header file common to all audio modules and must be included in the audio framework.

2.1 APIs

Five functions have a software interface to the main program:

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

2.1.1 *PDM_FilterInit* function

This procedure initializes the static memory, sets default values and initializes lookup tables of the PDM2PCM library.

```
uint32_t PDM_FilterInit(PDM_Filter_Handler_t *pHandler);
```

Table 2. *PDM_FilterInit* function

I/O	Name	Type	Description
Input	*pHandler*	*PDM_Filter_Handler_t **	Pointer to internal static memory
Returned value	-	*uint32_t*	Error value

This routine must be called at least once at initialization time, when the real time processing has not started yet.

2.1.2 *PDM_Filter_setConfig* function

This procedure sets module dynamic parameters from the main framework to the module internal memory. It can be called at any time during processing.

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

Table 3. *PDM_Filter_setConfig* function

I/O	Name	Type	Description
Input	*pHandler*	*PDM_Filter_Handler_t **	Pointer to internal static memory
Input	pConfig	PDM_Filter_Config_t*	Pointer to dynamic parameters structure
Returned value	-	*uint32_t*	Error value

2.1.3 *PDM_Filter_getConfig* function

This procedure gets module dynamic parameters from internal static memory to the main framework. It can be called at any time during processing.

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

Table 4. *PDM_Filter_getConfig* function

I/O	Name	Type	Description
Input	*pHandler*	*PDM_Filter_Handler_t **	Pointer to internal static memory
Input	pConfig	PDM_Filter_Config_t*	Pointer to dynamic parameters structure
Returned value	-	*uint32_t*	Error value

2.1.4 *PDM_Filter_deInterleave* function

Not yet implemented.

2.1.5 *PDM_Filter* function

This procedure decodes an input PDM stream to an output PCM stream. It has to be called to process each frame.

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

Table 5. *PDM_Filter* function

I/O	Name	Type	Description
Input	*pDataIn*	*void **	Pointer to PDM input data
Output	*pDataOut*	*void **	Pointer to PCM output data
Input	*pHandler*	*PDM_Filter_Handler_t**	Pointer to internal static memory
Returned value	-	*uint32_t*	Error value

2.2 External definitions

2.2.1 Returned error values

Table 6 lists the possible returned error values. Each error sets a dedicated bit to 1, so more than one error code can be accumulated.

Table 6. Error values

Definition	Value	Description
PDM_FILTER_NO_ERROR	0x0000	No error
PDM_FILTER_ENDIANNESS_ERROR	0x0001	Unsupported endianness
PDM_FILTER_BIT_ORDER_ERROR	0x0002	Unsupported bit order
PDM_FILTER_CRC_LOCK_ERROR	0x0004	Target is not STM32
PDM_FILTER_DECIMATION_ERROR	0x0008	Unsupported decimation factor
PDM_FILTER_INIT_ERROR	0x0010	-
PDM_FILTER_CONFIG_ERROR	0x0020	-
PDM_FILTER_GAIN_ERROR	0x0040	Unsupported microphone gain
PDM_FILTER_SAMPLES_NUMBER_ERROR	0x0080	Unsupported number of samples

2.3 Static parameters structure

The PDM2PCM initial parameters are set using the corresponding static parameter structure before calling the *PDM_Filter_setConfig()* function.

```
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;
```

Table 7. Static parameters

Name	Description	Comment
bit_order	Specifies the bit order for input (MSB or LSB)	PDM_FILTER_BIT_ORDER_LSB (0x0000) PDM_FILTER_BIT_ORDER_MSB (0x0001)
endianness	Specifies if byte inversion is required	PDM_FILTER_ENDIANNESS_LE (0x0000) PDM_FILTER_ENDIANNESS_BE (0x0001)
high_pass_tap	Specifies the HP filter tap value	Coefficient value in Q31 format of the high-pass filter. If 0 the filter is not used.

Table 7. Static parameters (continued)

Name	Description	Comment
in_ptr_channels	Specifies the number of channels in the input PDM stream	INTEGER NUMBER > 0 in_ptr_channels = 1 when used with one microphone.
out_ptr_channels	Specifies the number of channels in the output PCM stream.	INTEGER NUMBER > 0. out_ptr_channels=1 when used with one microphone.
pInternalMemory	Internal memory	Pointer to an array.

2.4 Dynamic parameters structure

It is possible to change the PDM2PCM configuration by setting new values in the dynamic parameter structure before calling the *PDM_Filter_setConfig()* function.

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

Table 8. Dynamic parameters

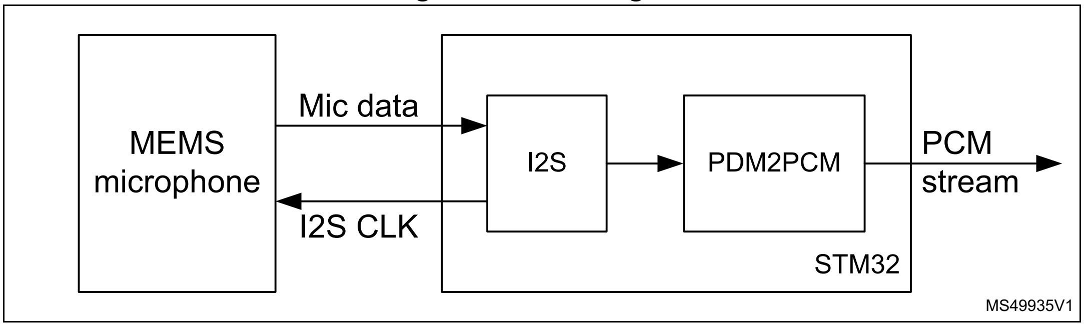
Name	Description	Comment
decimation_factor	Specifies the decimation factor.	PDM_FILTER_DEC_FACTOR_16 (0x0005) PDM_FILTER_DEC_FACTOR_24 (0x0006) PDM_FILTER_DEC_FACTOR_32 (0x0007) PDM_FILTER_DEC_FACTOR_48 (0x0001) PDM_FILTER_DEC_FACTOR_64 (0x0002) PDM_FILTER_DEC_FACTOR_80 (0x0003) PDM_FILTER_DEC_FACTOR_128 (0x0004)
output_samples_number	Specifies the number of PCM samples to be generated at each call of *PDM_Filter()* function	INTEGER NUMBER > 0
mic_gain	Specifies the microphone gain in dB	Gain is in the interval [-12 dB: +51 dB], with 1 dB steps.

3 Algorithm description

3.1 Processing steps

As shown in *Figure 1*, a MEMS microphone outputs a PDM stream, which is a high frequency stream of 1-bit digital samples. The library expects a stream made of 8-sample blocks (one byte), which will be acquired using a synchronous serial port (SPI or I2S) of the STM32 microcontroller. The microphone PDM output is synchronous with its input clock, therefore the used STM32 serial port generates a clock signal for the microphone.

Figure 1. Block diagram



The PDM data from the microphone are packed in 8-bit blocks, and then filtered and decimated. The frequency of the obtained PCM signal depends on the decimation factor configured before the library initialization.

The decimation factors have been defined to get a PCM stream of the desired sampling frequency, depending on the PDM clock value. Examples are given in *Table 9*.

Table 9. Decimation factors and corresponding frequencies

Decimation factor	PDM clock frequency	PCM sample rate
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
	2.048 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

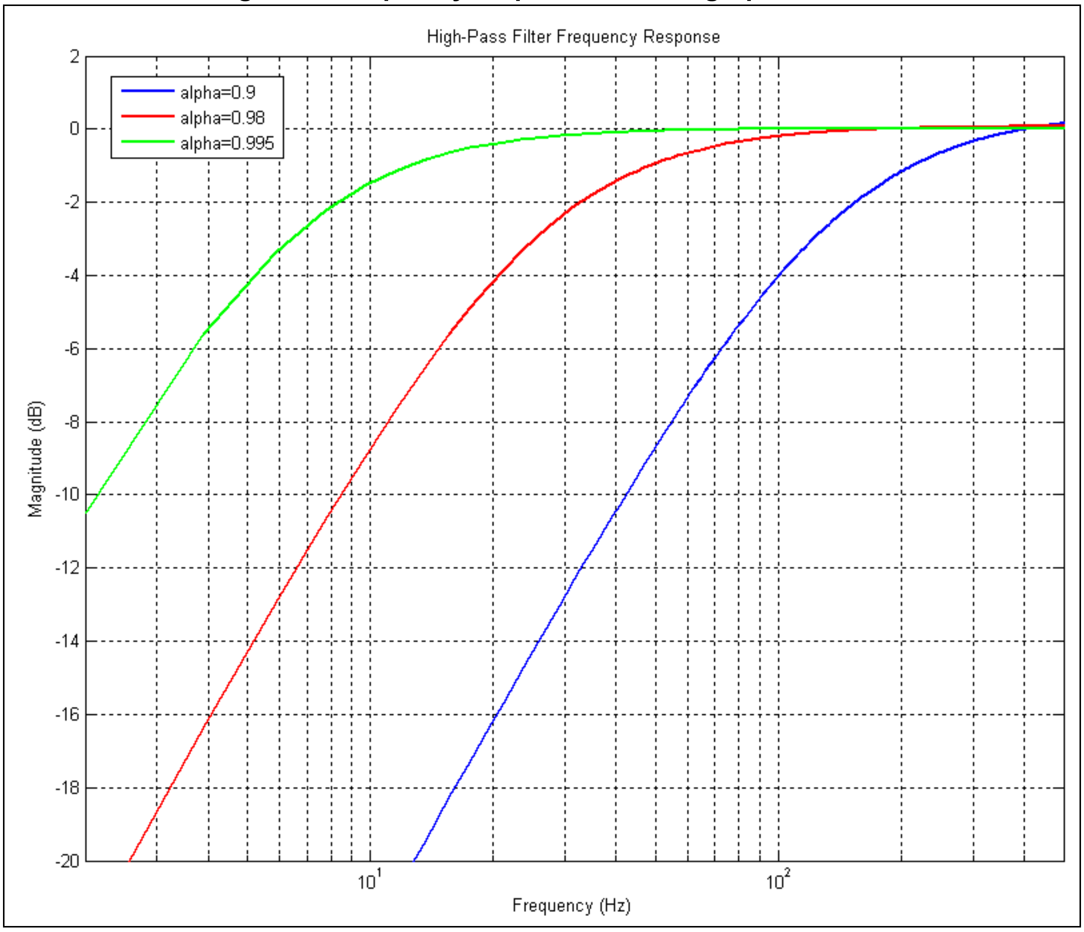
The digital signal resulting from the filter and decimator pipeline is then processed by a high-pass filter, to remove DC offset, and by a digital gain, to attenuate or amplify the PCM samples.

3.1.1 High-pass filter

The high-pass filter is a one-pole recursive filter. The cut-off frequency is configured by modifying the parameter *high_pass_tap* from the *PDM_Filter_Handler_t*. This coefficient value must be in the range [0 : 1]. The format used is Q0.31, meaning that 1 corresponds to the maximum integer value obtainable with 31-bit resolution. For example, configuring the *high_pass_tap* parameter to 0.98 corresponds to $0.98*(2^{31}-1) = 2104533974$.

Figure 2 is a plot of the frequency response of this filter for three different values of the *high_pass_tap* parameter, for a PCM sampling rate of 16 kHz.

Figure 2. Frequency response of the high-pass filter



When the coefficient is set to 0, the high-pass filter is bypassed.

3.1.2 Digital volume

The digital volume attenuates or amplifies the samples before saturating them to a signed 16-bit value. The *mic_gain* parameter is the gain value (in dB) to apply to the PCM stream. The minimum value is -12 dB, the maximum is 51 dB, with 1 dB steps.

3.2 Data formats

The input of PDM2PCM library is expected to be a PDM stream, byte-packed, at the MEMS microphone clock frequency. It can be a single or double data stream. The output is a PCM stream.

3.3 Results of measurements

All measurements have been made using an STM32F469 board, with a MEMS microphone MP34DT01 mounted on it. They have been made in an anechoic environment, using a professional monitoring system as acoustic source.

3.3.1 Distortion measurements

The distortion measurements are made with a test signal at a nominal acoustic level of about 90 dB SPL at the source point. The microphone is placed at a distance of 10 cm, and *mic_gain* is equal to 0 dB.

These data take into account all system noise, including noise floors brought back by PCB and power supplies.

Figure 3. Distortion measurements at 300 Hz

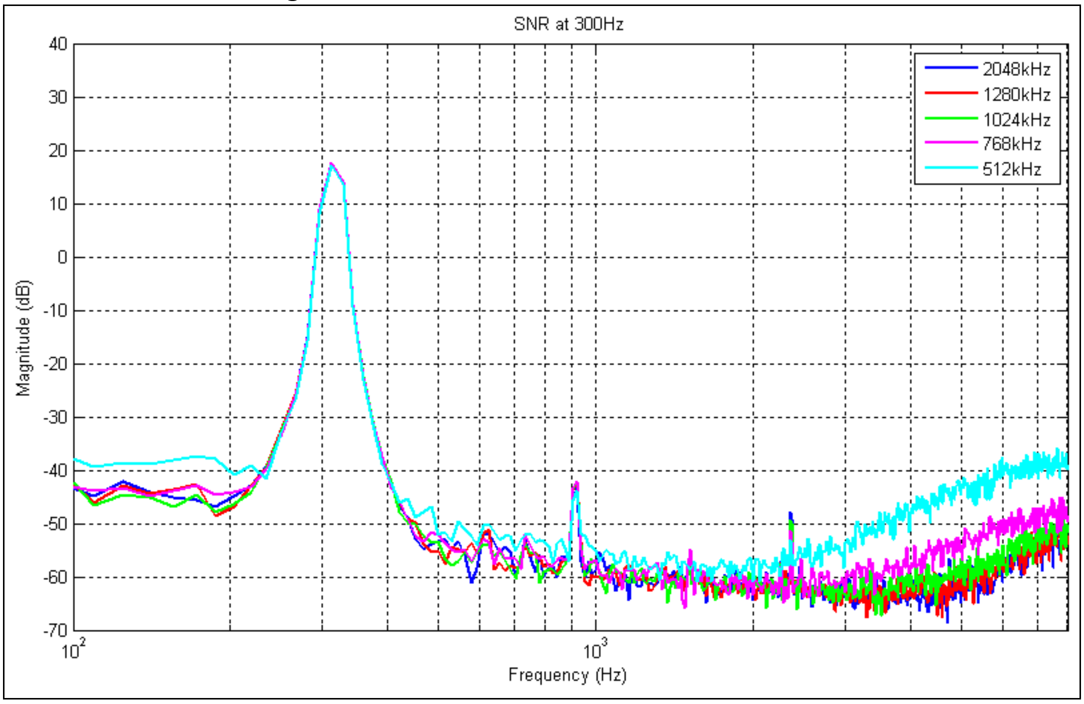


Table 10. Distortion measurements at 300 Hz

PDM clock	SNR at -31 dBFS	SNR at 0 dBFS (extrapolation)
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

Figure 4. Distortion measurements at 500 Hz

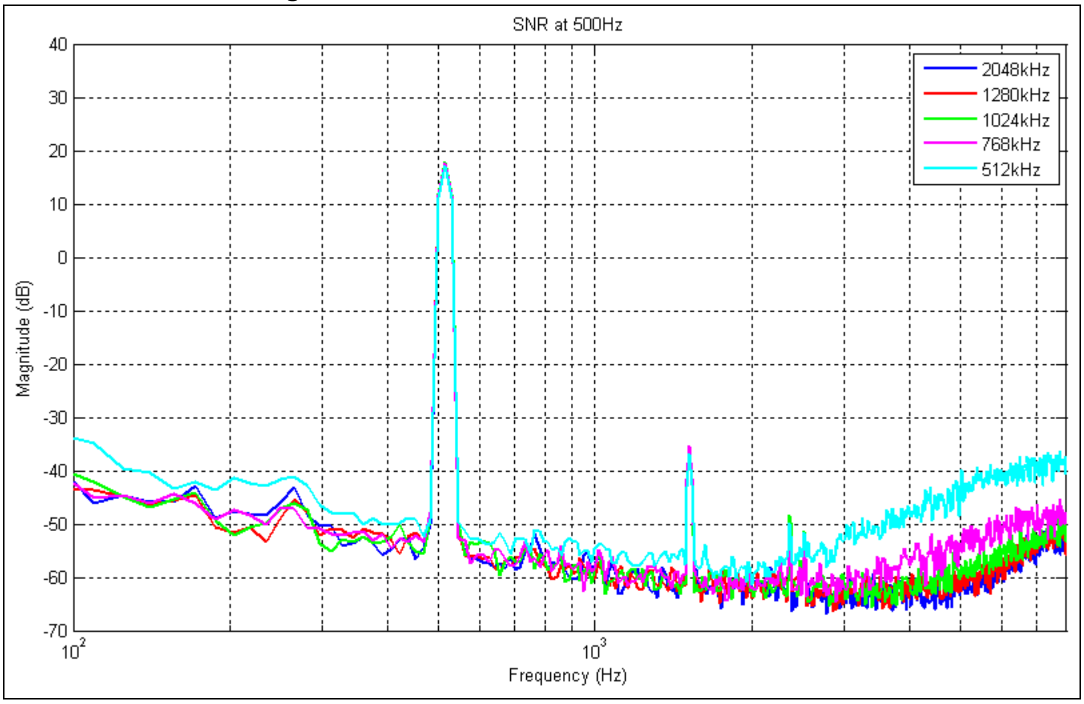


Table 11. Distortion measurements at 500 Hz

PDM clock	SNR at -31 dBFS	SNR at 0 dBFS (extrapolation)
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

Figure 5. Distortion measurements at 1000 Hz

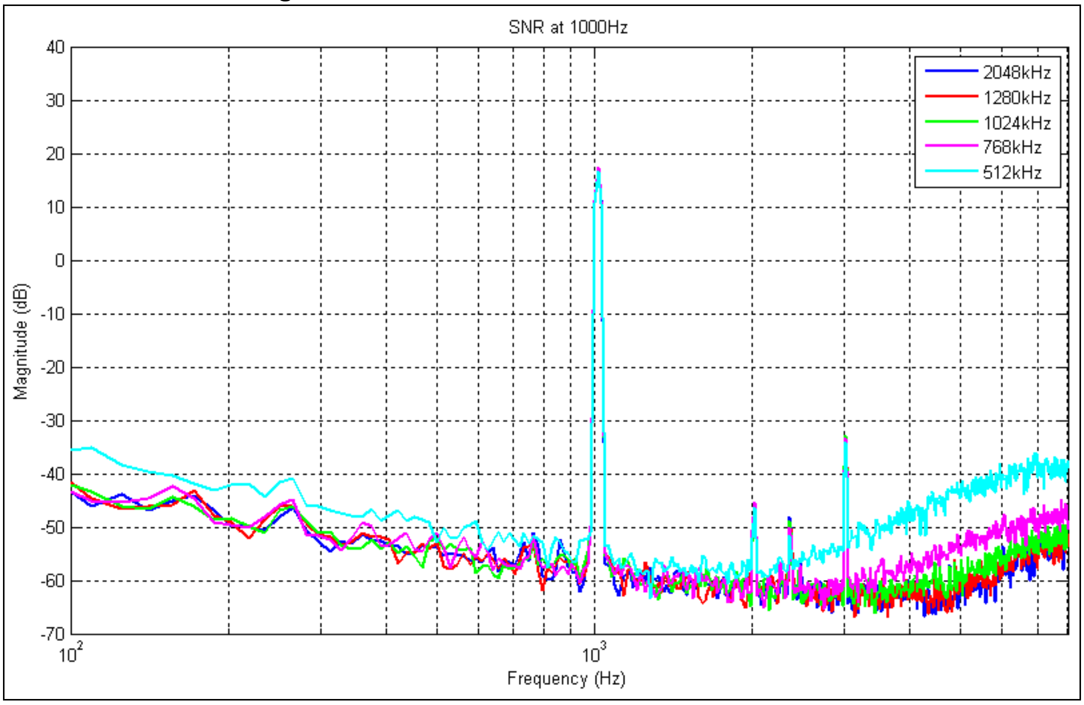


Table 12. Distortion measurements at 1000 Hz

PDM clock	SNR at -31 dBFS	SNR at 0 dBFS (extrapolation)
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 Speech signal

The test signal is a speech sequence, played at a nominal level of 90 dB SPL. The MEMS microphone is placed at 30 cm from the acoustic source. The digital signal captured at the output of the PDM2PCM library for different *mic_gain* values is shown in *Figure 6*.

Figure 6. Comparison of speech levels

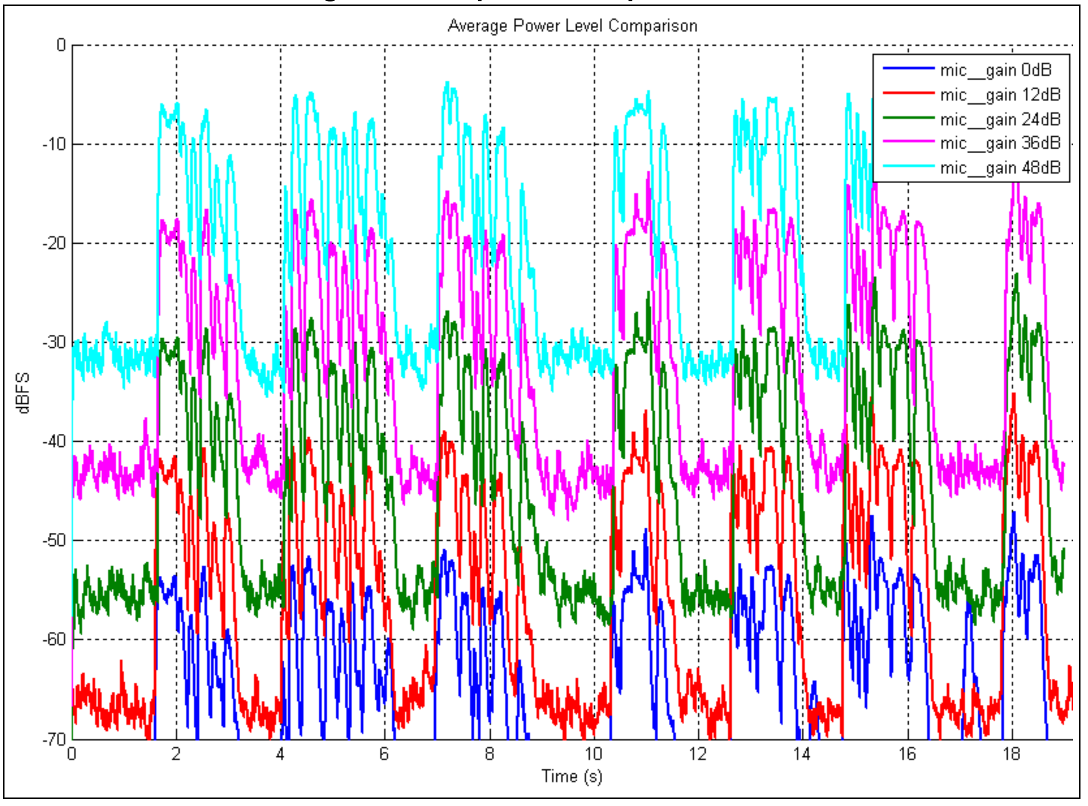


Table 13. Average speech signal level at PDM2PCM library output

***mic_gain* parameter value**	**Digital speech level**
0 dB	-54 dBFS
12 dB	-42 dBFS
24 dB	-30 dBFS
36 dB	-18 dBFS
48 dB	-8 dBFS

4 Application description

4.1 Module integration example

4.1.1 Library initialization

Once the memory is allocated, some routines must be called to initialize the PDM2PCM library static memory:

- *PDM_Filter_Init()* has to be called each time the processing in the audio is stopped and started.
- *PDM_Filter_setConfig()* has to be called at least once before processing start, to set configurable parameter

Furthermore, as the PDM2PCM library runs on STM32 devices, CRC HW block must be enabled and reset.

The static and dynamic parameters structures must be allocated. Their types are defined in pdm2pcm_glo.h header. Example of allocation:

```
/*Enables and resets CRC-32 from STM32 HW */
    __HAL_RCC_CRC_CLK_ENABLE();
    CRC->CR = CRC_CR_RESET;

    PDM_Filter_Handler_t  PDM1_filter_handler;
    PDM_Filter_Config_t   PDM1_filter_config;

    /* Initialize PDM Filter structure */
    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 Module execution

The run time process can start when the hardware is configured and the PDM2PCM library is initialized and configured.

At each new interrupt, when enough bits have been buffered, the PDM2PCM filter routine can be called. Between two consecutive calls to this filter routine, the dynamic parameters can be changed.

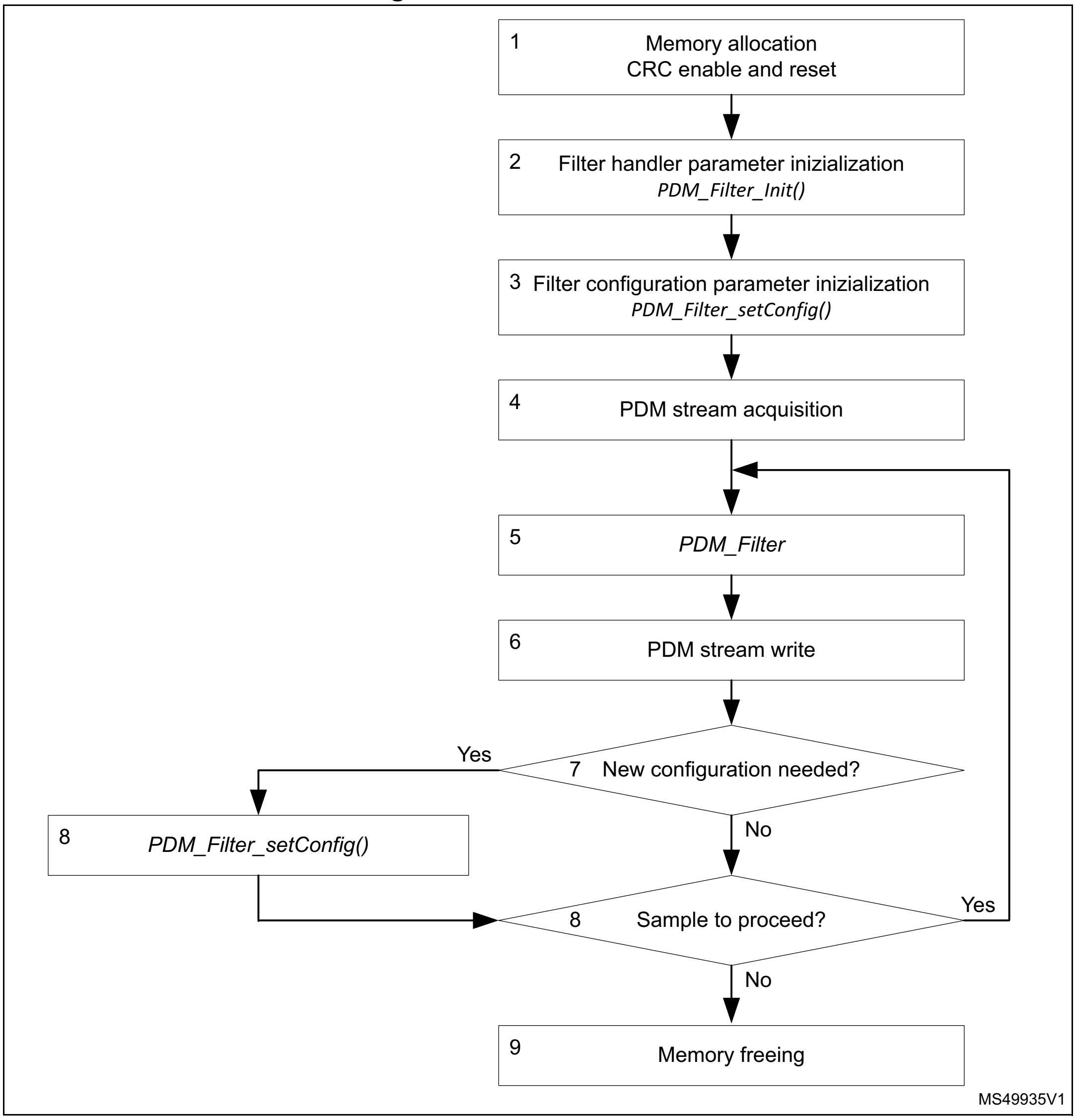
```
do
{
    /* process current frame */
    PDM_Filter(&pdm_buffer[0], &pcm_buffer[0], &PDM1_filter_handler);

                        :
                        :
    /* change volume setting */
    PDM1_filter_config.mic_gain = 12;
    PDM_Filter_setConfig((PDM_Filter_Handler_t *)&PDM1_filter_handler,
&PDM1_filter_config);
}
```

4.1.3 Module APIs calls

The flow is shown in *Figure 7*, and the steps listed below:

1. As explained above, PDM2PCM handler and configuration structures have to be allocated, as well as PDM input and PCM output buffers. CRC must be enabled in order to unlock the library.
2. The PDM filter handler parameters must now be set to the desired values, and *PDM_Filter_Init()* function called.
3. The PDM filter configuration parameters must now be set to the desired values, and *PDM_Filter_setConfig()* function called.
4. The PDM input bit stream is read from the proper interface, packed byte by byte.
5. Call to *PDM_Filter() function* will execute the PDM2PCM algorithm.
6. The PCM output audio stream can now be written in the proper interface.
7. If needed, the user can change configuration parameters and call the *PDM_Filter_setConfig()* function to update the library configuration.
8. If the application and the PDM input stream are still running, the processing loop goes back to step 5, otherwise it ends.
9. Once the processing loop is over, allocated memory has to be freed.

Figure 7. Module flow-chart



5 Revision history

Table 14. Document revision history

Date	Revision	Description of changes
06-Jul-2018	1	Initial release