AN3998
Application note

PDM audio software decoding on STM32 microcontrollers

1 Introduction

This application note presents the algorithms and architecture of an optimized software implementation for PDM signal decoding and audio signal reconstruction when connecting an ST MP45DT02 MEMS microphone with an STM32 microcontroller. It can directly take the Pulse Density Modulated (PDM) data output from the microphone and convert it to 16-bit pulse-code modulation (PCM) format.

This document also provides quick start information describing how to implement the PDM Library for single microphone acquisition via I2S based on the STM32F4 microcontroller and STM32F4DISCOVERY board.

For more details about this implementation, please refer to AN3997 *Audio playback and recording using the STM32F4DISCOVERY*.

Contents

List of figures

2 PDM signal introduction

Pulse density modulation, or PDM, is a form of modulation used to represent an analog signal in the digital domain.

In a PDM signal, specific amplitude values are not encoded into pulses as they would be in PCM. Instead it is the relative density of the pulses that corresponds to the analog signal's amplitude.

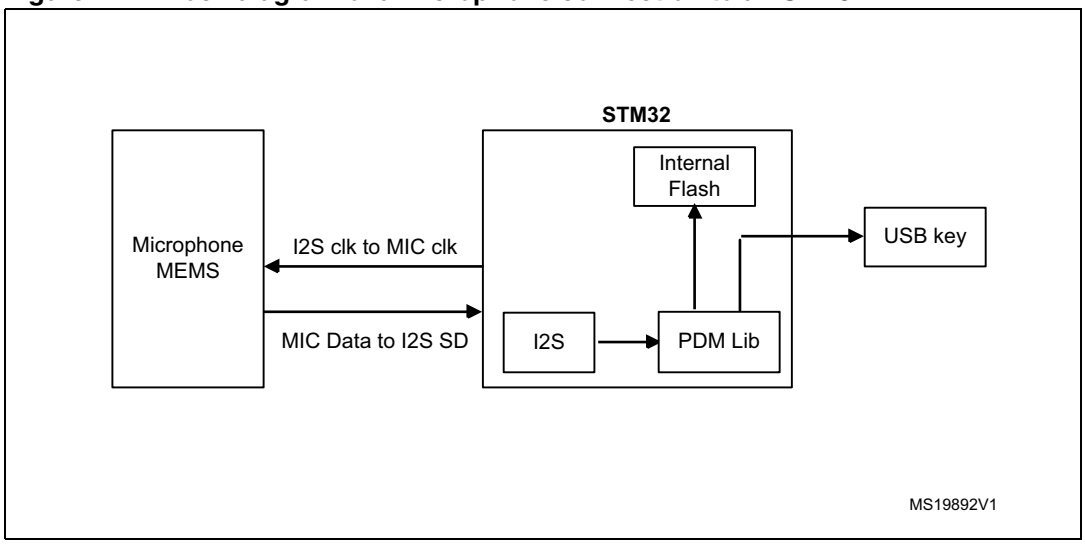
To get the framed data from the PDM bit stream, decimation filters are usually used. The first stage of decimation is used to reduce the sampling frequency, followed by a high pass filter to remove the signal DC offset.

3 Hardware interface: microphone connection and acquisition

The MP45DT02 MEMS microphone outputs a PDM signal, which is a high frequency (1 to 3.25 MHz) stream of 1-bit digital samples.

This output is acquired in blocks of 8 samples by using a synchronous serial port (SPI or I2S) of the STM32 microcontroller. The microphone's PDM output is synchronous with its input clock; therefore an STM32 SPI/ I2S peripheral generates a clock signal for the microphone.

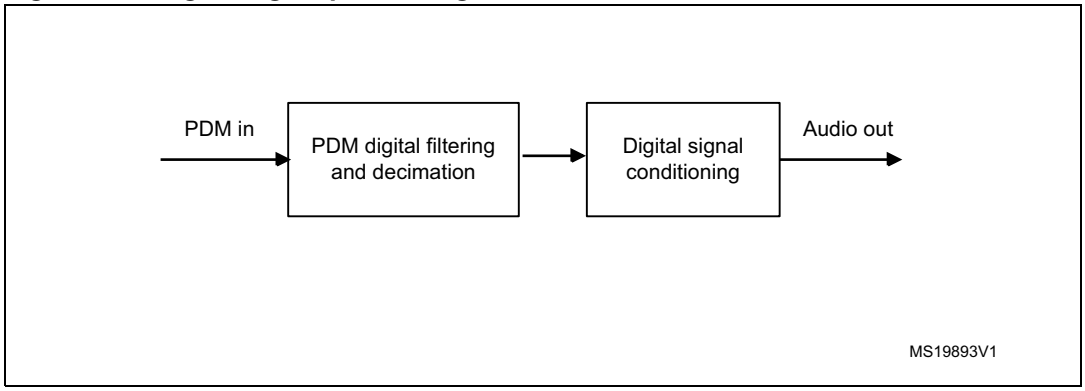
Figure 1. Block diagram of a microphone connection to an STM32.



4 Software interface: digital signal processing

The data coming from the microphone is sent to the decimation process, which consists of two parts: a decimation filter converting 1-bit PDM data to PCM data, followed by two individually configurable IIR filters (low pass and high pass). The reconstructed audio is in 16-bit pulse-code modulation (PCM) format. After the conversion, it produces raw data that can be handled depending on the application implementation (stored as wave/compressed data in a mass storage media, transferred to an external audio codec DAC through I2S peripheral...).

Figure 2. Digital signal processing



4.1 PDM digital filtering and decimation

The PDM signal from the microphone is filtered and decimated in order to obtain a sound signal at the required frequency and resolution.

The frequency of the PDM data output from the microphone (which is the clock input to the microphone) must be a multiple of the final audio output needed from the system. For example, to perform a decimation of 80, for the output rate of 30 kHz, we need to provide a clock frequency 2.4MHz to the microphone.

The output of the filter pipeline is a 16-bit value, we consider [-32768, 32767] as the output range for a unitary gain (0 dB).

4.2 Digital signal conditioning

The digital audio signal resulting from the previous filter pipeline is further processed for proper signal conditioning.

The first stage is a high pass filter designed mainly to remove the signal DC offset. It has been implemented via an IIR filter with a cut-off frequency below the audible frequency range in order to preserve signal quality.

The second stage is a low pass filter implemented using an IIR filter.

Both filters can be enabled/disabled and configured (cut-off frequencies) by using the filter initialization function.Gain can be controlled by an external integer variable (MicGain) as shown in the following equation: G = MicGain/64.

5 PDM audio software decoding library description

The PDM library is composed of a structure and the implementation of four PDM filter functions. The library uses two buffers, the PDM Input buffer and the PCM Output buffer; the application must define these buffers in the main program.

- Input buffer (data) is a uint8 variable with a length equal to (Output frequency / 1000 * decimation factor * Input Microphone Channels / 8) at least.
- Output buffer (dataOut) is a uint16 variable with a length equal to (Output frequency / 1000 * Output Microphone Channels) at least.

The structure is defined in the pdm_filter.h file and is used to configure the filter; it is composed as follows:

```
typedef struct {
uint16_t Fs;
float LP_HZ;
float HP_HZ;
uint16_t Out_MicChannels;
char InternalFilter[34];
} PDMFilter_InitStruct;
```

- **Fs:** Defines the frequency output of the filter in Hz.
- **LP_HZ:** Defines the low pass filter cut-off frequency. If 0, the low pass filter is disabled.
- **HP_HZ:** Defines the high pass filter cut frequency. If 0, the high pass filter is disabled.
- **In_MicChannels:** Define the number of microphones in the input stream. This parameter is used to specify the interlacing of microphones in the input buffer. The PDM samples are grouped eight by eight in u8 format (8-bit).
- **Out_MicChannels:** Defines the number of microphones in the output stream; this parameter is used to interlace different microphones in the output buffer. Each sample is a 16-bit value.
- **InternalFilter:** Defines a 34 Byte memory used internally by the library during the PDM decimation step.

The PDM audio software decoding library includes one header file pdm_filter.h and binary/object codes for the following platforms:

- libPDMFilter_IAR.a : for IAR compiler
- libPDMFilter_Keil.lib : for ARM compiler
- libPDMFilter_GCC.a : for GNU compiler

This Library is provided in the "STM32F4-DISCOVERY board firmware package" (V1.10 and later) under the following path 'Utilities\STM32F4-Discovery'

5.1 PDM_Filter_Init

This function is used to initialize the filter of a single output channel. It is defined as follows:

```
Void PDM_Filter_Init (PDMFilter_InitStruct * Filter)
```

It initializes the PDM Filter, according to the parameters specified in the PDMFilter_InitStruct.

The following is an example of filter initialization for a single microphone with a PDM at 16 kHz:

```
PDMFilter_InitStruct Filter;

Filter.LP_HZ = 8000;

Filter.HP_HZ = 10;

Filter.Fs = 16000;

Filter.Out_MicChannels = 1;

Filter.In_MicChannels = 1;


PDM_Filter_Init ((PDMFilter_InitStruct *) Filter);
```

Before calling the PDM_Filter_Init() function, the application code must enable the clock of the STM32 microcontroller CRC peripheral (configuration done in RCC register).

Otherwise, the PDM_Filter_Init() function will fail and go into an infinite loop.

5.2 PDM_Filter_xx_xx

These functions are used to process a millisecond of PDM data from a single microphone. They return a number of PCM samples equal to the frequency defined in the filter initialization, divided by 1000. In case of frequencies that are not multiple of 1000 (like 44100 Hz or 22050 Hz), the samples returned by the function are equal to the floor division of the frequencies (44 and 22).

The functions are defined as follows:

```
PDM_Filter_XX_XX(uint8_t* data, uint16_t* dataOut, uint16_t MicGain,
PDMFilter_InitStruct * Filter)
```

- **data:** is the input buffer containing the PDM; the application must pass to the function the pointer to the first input sample of the microphone that must be processed.
- **dataOut:** is the output buffer processed by the PDM_Filter function. The application must pass to the function the pointer to the first sample of the channel to be obtained.
- **Volume:** is a value between 0 and 64 used to specify the microphone gain.
- **Filter:** is the structure containing all the filter parameters specified by the user using the PDM_Filter_Init function.

There are four different implementations of this function depending on the decimation factor (64 or 80) and on the LSB or MSB representation of the input buffer.

6 Revision history

Table 1. Document revision history

Date	Revision	Changes
27-Oct-2011	1	Initial release.