
**Interfacing PDM digital microphones
using STM32 32-bit Arm[®] Cortex[®] MCUs**

Introduction

Digital MEMS (microelectromechanical systems) microphones target all audio applications where small size, high sound quality, reliability and affordability are key requirements.

Their combination of small footprint and noise immunity allows the implementation of multiple microphones in a single device, creating an increasing growth of audio in industrial and consumer applications by offering a hands-free human machine interface, noise cancellation, and high quality audio capture.

The STM32 32-bit Arm[®] Cortex[®] MCUs offer a wide audio capability with a rich connectivity, including serial and enhanced voice-acquisition interfaces allowing the user to easily build solution for microphone-based applications.

This application note targets digital MEMS microphones having a pulse-density modulated (PDM) output, describing how to connect them in mono and stereo configurations to STM32 MCUs using the SPI/I2S, SAI and DFSDM peripherals. It provides guidelines along with examples based on CubeMX showing how to properly configure the STM32 to acquire and handle raw data from the microphones, to transform them into standard data for audio.

arm

Contents

1	PDM digital microphones overview	7
1.1	Sound acquisition overview	7
1.2	PDM digital microphone block diagram	8
1.3	Basic digital microphones connection	9
1.4	PDM and PCM signals	12
1.4.1	Pulse density modulation (PDM)	12
1.4.2	Pulse code modulation (PCM)	12
1.4.3	PDM to PCM conversion	13
1.5	Acoustic parameters	13
1.5.1	Sensitivity	13
1.5.2	Signal-to-noise ratio (SNR)	13
1.5.3	Acoustic overload point (AOP)	14
1.5.4	Power supply rejection ratio (PSRR)	14
1.6	Added value of digital microphones	14
1.7	Available ST digital microphones	15
2	Connecting PDM digital microphones to STM32 MCUs	16
2.1	Serial peripheral interface/Inter-IC sound (I2S)	16
2.1.1	Mono configuration	16
2.1.2	Stereo configuration	18
2.2	Serial Audio Interface (SAI)	20
2.2.1	Using a single sub-block	20
2.2.2	Using two synchronous SAI sub-blocks	23
2.2.3	Using PDM interface	24
2.3	Digital filter for sigma delta modulators (DFSDM)	27
2.3.1	Stereo configuration	28
2.4	Clocking considerations	28
2.4.1	The digital microphone clock	28
2.4.2	The Peripheral clocks	29
2.5	GPIOs number considerations	31
3	Digital signal processing	32

3.1	PDM audio software decoding Library	32
3.1.1	Overview	32
3.1.2	Digital data flow	32
3.1.3	Digital signal processing	33
3.2	DFSDM filters for digital signal processing	34
3.2.1	Digital data flow: acquisition and processing	34
4	Examples of configuration based on STM32CubeMX	35
4.1	Example 1: Interfacing digital microphones in mono or stereo mode with I2S, SPI or a single SAI sub-block	35
4.1.1	Hardware configuration using STM32CubeMX	35
4.1.2	Adding PDM software decoding library middleware files	47
4.2	Example 2: Interfacing digital microphones in stereo mode with SAI using two synchronous sub-blocks	48
4.2.1	SAI configuration using STM32CubeMX	48
4.2.2	Adding PDM software decoding library middleware files	53
4.3	Example 3: Interfacing digital microphones in stereo mode with SAI using PDM interface	53
4.3.1	SAI configuration using STM32CubeMX	53
4.3.2	Adding PDM software decoding library middleware files	57
4.4	Example 4: Interfacing digital microphones using DFSDM	58
4.4.1	DFSDM configuration using STM32CubeMX	58
5	Conclusion	65
6	Revision history	66

List of tables

Table 1.	DOUT signal pattern selection	9
Table 2.	Pin description	9
Table 3.	Added value of digital microphone	14
Table 4.	ST digital microphones	15
Table 5.	recommended IO lines versus the number digital microphones to be connected	24
Table 6.	Applications examples and the associated microphone clock frequency	29
Table 7.	Hardware used to connect one digital microphone.	31
Table 8.	Hardware used to connect two digital microphones	31
Table 9.	Hardware used to connect four digital microphones.	31
Table 10.	I2S2 clock configuration and accuracy	36
Table 11.	SPI clock configuration and accuracy	39
Table 12.	Clock configuration and accuracy	42
Table 13.	SAI clock configuration and accuracy	49
Table 14.	DFSDFM filter order values	60
Table 15.	DFSDM clock configuration accuracy values	64
Table 16.	Document revision history	66

List of figures

Figure 1.	Example of sound acquisition in audio application	7
Figure 2.	Typical PDM digital MEMS microphone block diagram	8
Figure 3.	Mono configuration - Generating data on right channel	9
Figure 4.	Right channel data pattern	10
Figure 5.	Mono configuration - Generating data on left channel	10
Figure 6.	Left channel data pattern	10
Figure 7.	Stereo configuration: Sharing one data line	11
Figure 8.	Stereo configuration data pattern	11
Figure 9.	PDM signal	12
Figure 10.	PCM signal	13
Figure 11.	Connecting one digital microphone to SPI or I2S in mono configuration	17
Figure 12.	Connecting two digital microphone to SPI block in stereo configuration	18
Figure 13.	Stereo mode timing diagram.	19
Figure 14.	Connecting one digital microphone to SAI in mono configuration	20
Figure 15.	Connecting two digital microphones to SAI in stereo configuration using a SAI sub-block and a timer	22
Figure 16.	Connecting two digital microphone to SAI in stereo configuration using two synchronous SAI sub-blocks	23
Figure 17.	PDM interface capability on interfacing up to four microphone pairs	25
Figure 18.	Data format when using the SAI PDM interface with a slot size of 32 bits, and 8 microphones	26
Figure 19.	Data format when using the SAI PDM interface with a slot size of 8 bits, and 8 microphones	26
Figure 20.	DFSDM capability to interface up to 4 digital microphones	27
Figure 21.	Stereo configuration	28
Figure 22.	Bus and kernel clock topology for SPI	30
Figure 23.	Bus and kernel clock topology for SAI and DFSDM	30
Figure 24.	Digital data acquisition and processing (block diagram).	33
Figure 25.	Digital signal processing	33
Figure 26.	Digital data acquisition and processing using DFSDM (block diagram)	34
Figure 27.	I2S GPIO pin configuration	36
Figure 28.	I2S2 clock configuration	37
Figure 29.	I2S configuration	37
Figure 30.	I2S parameter settings	38
Figure 31.	I2S DMA settings	38
Figure 32.	SPI GPIO and pin configuration	39
Figure 33.	SPI clock configuration	39
Figure 34.	SPI configuration	40
Figure 35.	SPI parameter setting	40
Figure 36.	SPI DMA settings	41
Figure 37.	SAI GPIO and pin configuration	41
Figure 38.	SAI clock configuration at 16 kHz for mono mode	42
Figure 39.	SAI configuration	43
Figure 40.	SAI parameter setting	44
Figure 41.	SAI DMA settings	45
Figure 42.	DMA request settings	45
Figure 43.	TIM GPIO and pin configuration	46
Figure 44.	TIM configuration	46

Figure 45.	TIM parameter settings	47
Figure 46.	SAI GPIO and pin configuration	48
Figure 47.	SAI clock configuration	49
Figure 48.	SAI parameter settings	51
Figure 49.	SAIB parameter settings	52
Figure 50.	SAI DMA settings	52
Figure 51.	SAI GPIO and pin configuration	53
Figure 52.	SAI clock configuration for 2 microphones at 16 kHz	54
Figure 53.	SAI configuration	54
Figure 54.	SAI parameter settings	55
Figure 55.	SAI DMA settings	56
Figure 56.	CORTEX_M7 configuration	56
Figure 57.	CORTEX_M7 parameter settings	57
Figure 58.	DFSDM GPIO configuration in stereo mode	58
Figure 59.	DFSDM pin configuration	59
Figure 60.	DFSDM Channel 1 configuration	59
Figure 61.	DFSDM Channel 0 configuration	60
Figure 62.	DFSDM Filter 0 configuration	61
Figure 63.	DFSDM Filter 1 configuration	61
Figure 64.	DFSDM output clock configuration	62
Figure 65.	DFSDM DMA setting	63
Figure 66.	DFSDM clock configuration	63

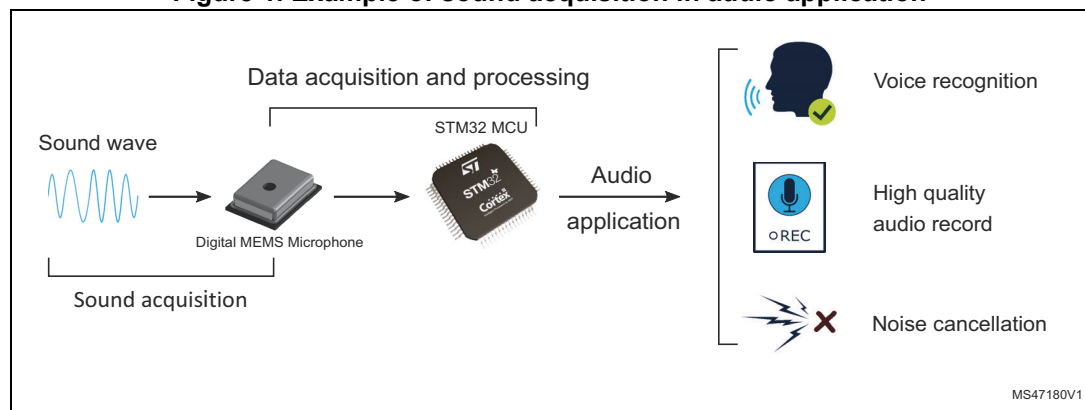
1 PDM digital microphones overview

This section provides a brief description of PDM digital microphones and gives a basic cases of interfacing them with STM32 MCUs

1.1 Sound acquisition overview

The digital MEMS microphone is a sensor that convert acoustic pressure waves into a digital signal. The STM32 microcontroller acquires digital data from the microphone(s) through particular peripherals to be processed and transformed into data standard for audio. The audio data is then handled by the microcontroller according to the targeted audio application.

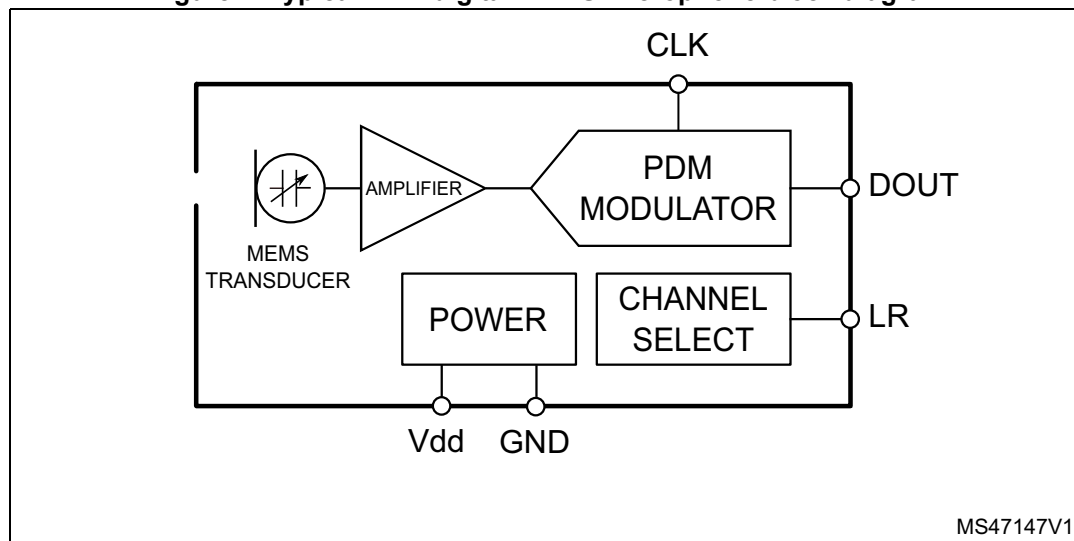
Figure 1. Example of sound acquisition in audio application



1.2 PDM digital microphone block diagram

The main parts in a digital microphone are a MEMS transducer, an amplifier and a PDM modulator.

Figure 2. Typical PDM digital MEMS microphone block diagram



MEMS TRANSDUCER

The MEMS TRANSDUCER is a variable capacitance that converts the change into air pressure caused by sound waves to a voltage.

AMPLIFIER

The AMPLIFIER buffers the voltage provided by the MEMS TRANSDUCER, and provides a sufficiently strong signal to the PDM MODULATOR.

PDM MODULATOR

PDM MODULATOR converts the buffered analog signal into a serial pulse density modulated signal. The clock input (CLK) is used to control the PDM modulator. The clock frequency range for ST digital microphones is from 1 MHz to 3.25 MHz. This frequency will define the sampling rate at which the amplifier's analog output signal is sampled to produce a discrete-time representation (PDM bitstream).

CHANNEL SELECT

The microphone's output is driven to the proper level on a selected clock edge and then goes into a high impedance state for the other half of the clock cycle. The CHANNEL SELECT defines the clock edge on which the digital microphone outputs valid data. The LR pin must be connected to Vdd or GND.

[Table 1](#) shows how to select the DOUT signal pattern.

Table 1. DOUT signal pattern selection

LR	DOUT	
	CLK low	CLK high
GND	Valid data	High impedance
Vdd	High impedance	Valid data

POWER

POWER delivers Vdd and GND supplies to the different digital microphone's components. The power supply should be properly provided to the microphone since any ripple can generate noise on the output.

Pin description

Table 2. Pin description

Pin name	Function	Direction
Vdd	3.3 V Power supply	Input
GND	0 V	Input
LR	Left/right selection	Input
CLK	Synchronization clock	Input
DOUT	Left/right PDM data output	Output

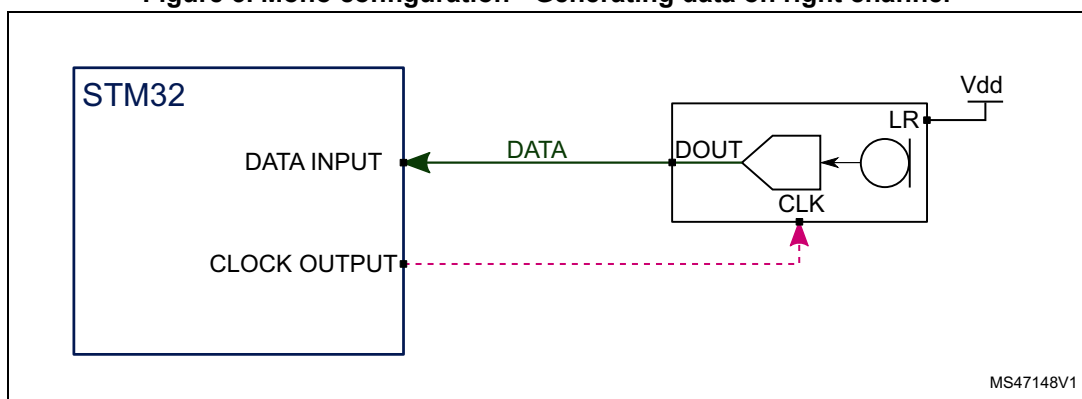
1.3 Basic digital microphones connection

Mono mode

In this mode LR pin can be either connected to Vdd or to GND.

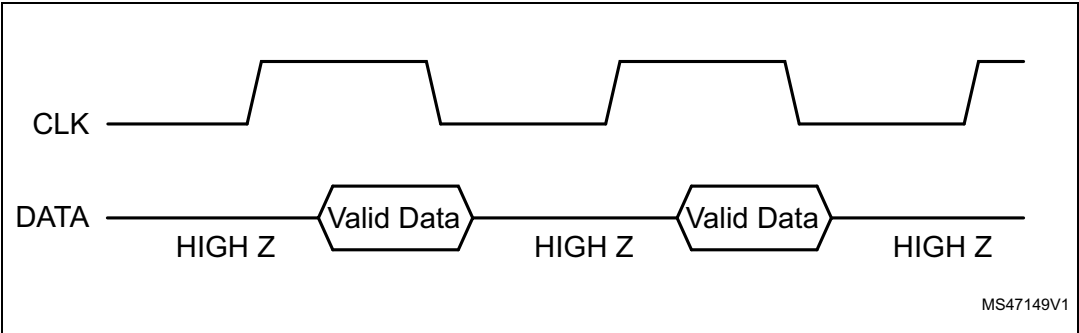
LR pin is connected to Vdd

Figure 3. Mono configuration - Generating data on right channel



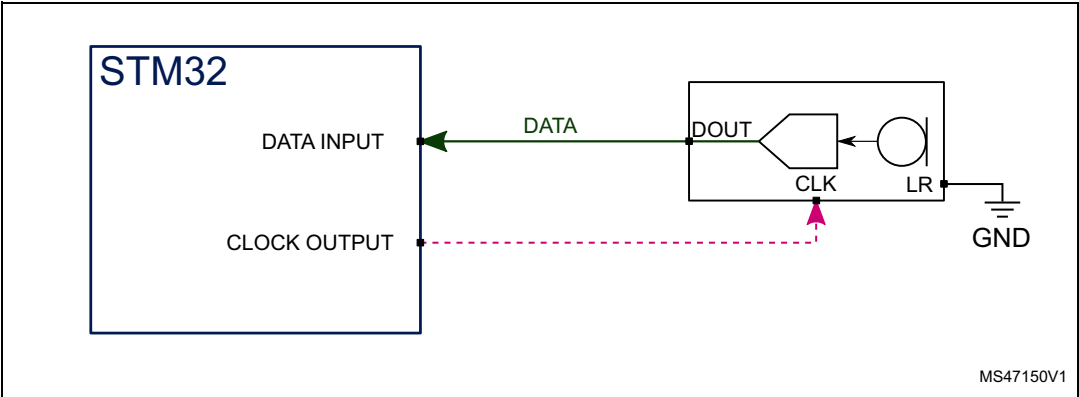
On the rising edge of the clock, the microphone will generate valid data for half of the clock period, then goes into a high impedance state for the other half.

Figure 4. Right channel data pattern



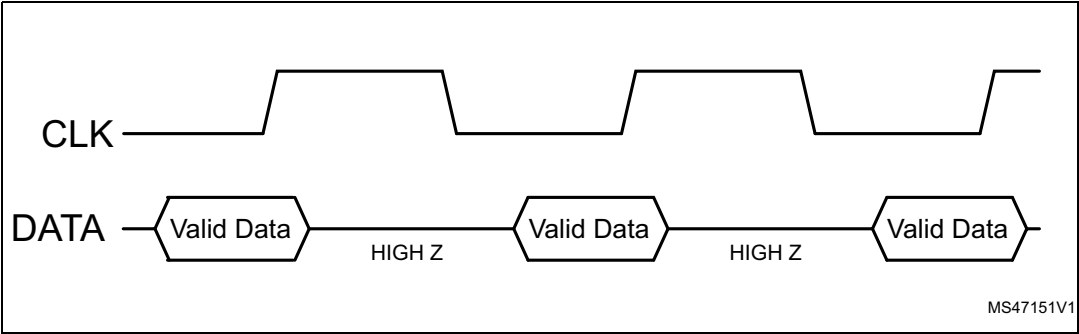
LR pin is connected to GND

Figure 5. Mono configuration - Generating data on left channel



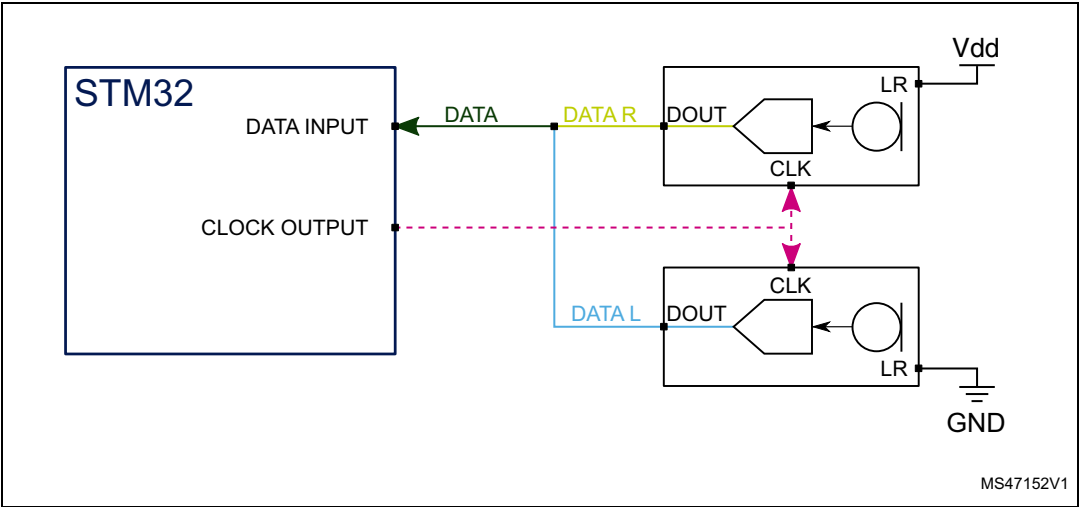
On the falling edge of the clock, the microphone will generate valid data for half of the clock period, then goes into a high impedance state for the other half.

Figure 6. Left channel data pattern



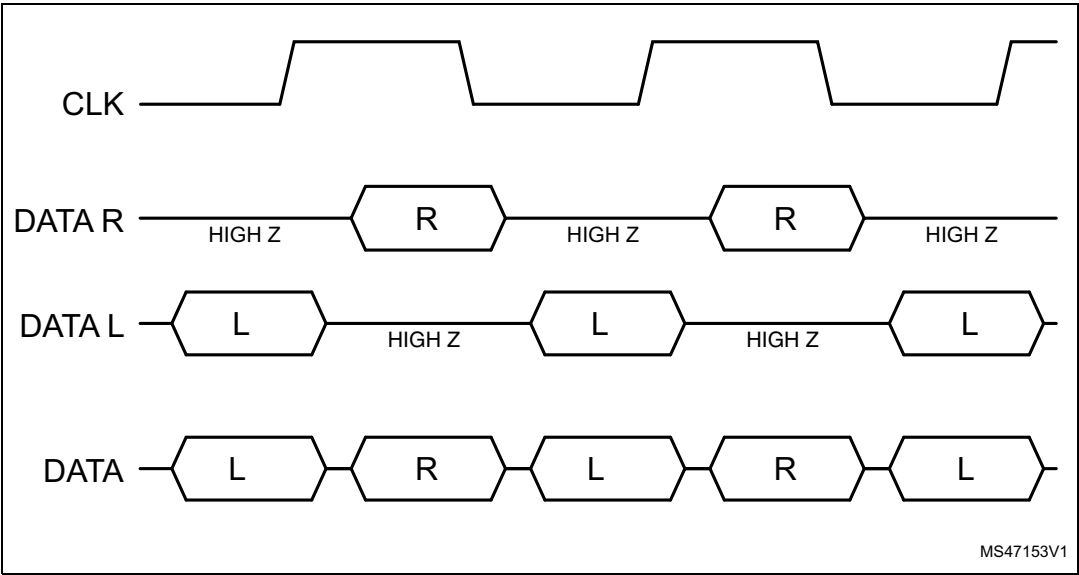
Stereo configuration

Figure 7. Stereo configuration: Sharing one data line



Two different digital MEMS microphones are connected on the same data line, configuring the first to generate valid data on the rising edge of the clock by setting the LR pin to Vdd and the other on the falling edge by setting the LR pin to GND.

Figure 8. Stereo configuration data pattern

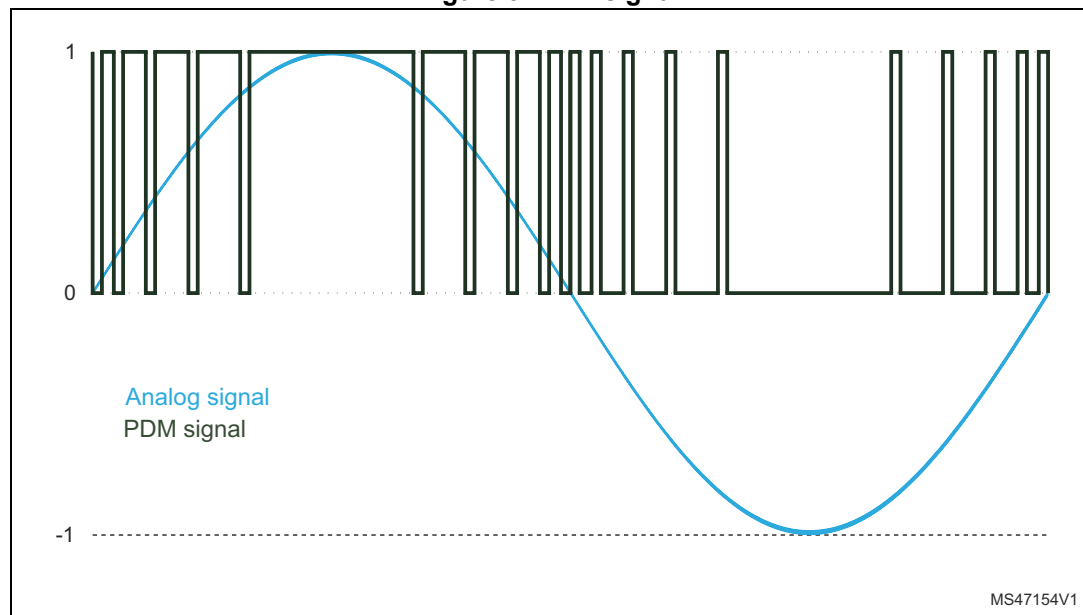


1.4 PDM and PCM signals

1.4.1 Pulse density modulation (PDM)

PDM is a form of modulation used to represent an analog signal in the digital domain. It is a high frequency stream of 1-bit digital samples. In a PDM signal, the relative density of the pulses corresponds to the analog signal's amplitude. A large cluster of 1s correspond to a high (positive) amplitude value, when a large cluster of 0s would correspond to a low (negative) amplitude value, and alternating 1s and 0s would correspond to a zero amplitude value.

Figure 9. PDM signal



1.4.2 Pulse code modulation (PCM)

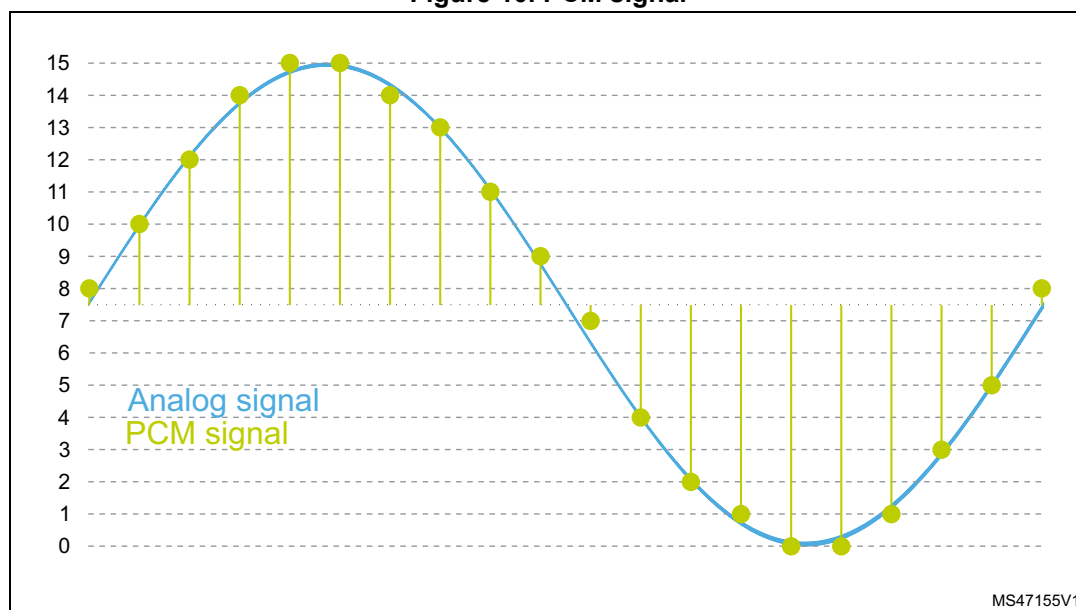
In the PCM signal, specific amplitude values are encoded into pulses.

A PCM stream has two basic properties that determine the stream's fidelity to the original analog signal:

- the sampling rate
- the bit depth

The sampling rate is the number of samples of a signal that are taken per second to represent it digitally. The bit depth determines the number of bits of information in each sample.

Figure 10. PCM signal



1.4.3 PDM to PCM conversion

In order to convert the PDM stream into PCM samples, the PDM stream needs to be filtered and decimated.

In the decimation stage, the sampling rate of the PDM signal is reduced to the targeted audio sampling rate (16 kHz for example). By selecting 1 of each M samples, the sample rate is reduced by a factor of M. Therefore, the PDM data frequency (which is the frequency of the microphone clock) is M times the target audio sampling frequency needed in an application, where M is the decimation factor.

$$\text{PDM Frequency} = \text{Audio Sampling Frequency} * \text{Decimation Factor}$$

The decimation factor is generally in the range of 48 to 128.

The decimation stage is preceded by a low-pass filter to avoid distortion from aliasing.

1.5 Acoustic parameters

1.5.1 Sensitivity

The sensitivity is the level of the electrical signal (expressed in dBFS) that the digital microphone outputs for a given acoustic reference signal.

Generally the sensitivity of a microphone is given using a tone of 1 kHz, at 1 Pa (or 94 dB SPL) as reference signal.

1.5.2 Signal-to-noise ratio (SNR)

The SNR specifies the ratio between the reference signal (94 dB SPL @ 1 kHz) and the amount of residual noise at the microphone output.

Higher SNR offers enhanced voice clarity as well as far-end (hands-free) intelligibility.

1.5.3 Acoustic overload point (AOP)

The AOP is the maximum acoustic signal, which the microphone can capture with acceptable distortion (some specifications allow up to 10% in terms of distortion at the acoustic overload point).

1.5.4 Power supply rejection ratio (PSRR)

The PSRR specification quantifies the capability of the microphone to reject noise relative to power supply changes.

1.6 Added value of digital microphones

Table 3. Added value of digital microphone

Feature	Added value
Immunity to RF noise and electromagnetic interference (EMI)	– Less integration effort
Analog signal conditioning not required	– Easier application design – Direct interface to codecs with digital microphones interface – Stereo mode needs only one data line – Significant saving on PCB area with more microphones in system – Flexibility to add additional microphones into the application
Robust digital transmission	– Easy MEMS positioning on application system – Standard digital conditioning – Allows audio enhancement integration for stereo capture, noise cancellation and beam forming

1.7 Available ST digital microphones

[Table 4](#) shows the available ST digital microphone.

Table 4. ST digital microphones

Part number	Top/bottom port	Supply voltage (V)	SNR (dB)	Sensitivity (dBFS)	AOP (dBSPL)
MP34DB02	Bottom	1.64 to 3.6	62.6	-26	120
MP34DT01-M	Top	1.64 to 3.6	61	-26	120
MP34DT02	Top	1.64 to 3.6	60	-26	120
MP34DT04	Top	1.6 to 3.6	64	-26	120
MP34DT04-C1	Top	1.6 to 3.6	64	-26	120
MP34DT05	Top	1.6 to 3.6	64	-26	122.5
MP45DT02-M	Top	1.64 to 3.6	61	-26	120

2 Connecting PDM digital microphones to STM32 MCUs

This section describes how to connect digital MEMS microphones to the SPI/ I²S, SAI and DFSDM peripherals embedded in STM32 microcontrollers in both mono and stereo configurations.

2.1 Serial peripheral interface/Inter-IC sound (I2S)

The STM32 microcontrollers offer a Serial Peripheral Interface block named SPI. Some of these SPI blocks also offer the possibility to use the Inter-IC Sound audio protocol (I2S). In addition, the STM32 series offer two versions of the SPI block, in this document, the oldest version is named SPI-V1, and latest version is named SPI-V2. When we refer to SPI block it means indifferently SPI-V1 or SPI-V2.

The SPI-V2 is currently available on STM32H7 family.

As we will in the next section, it is possible to connect one or two digital microphones to a SPI block by either using the SPI or I2S protocol.

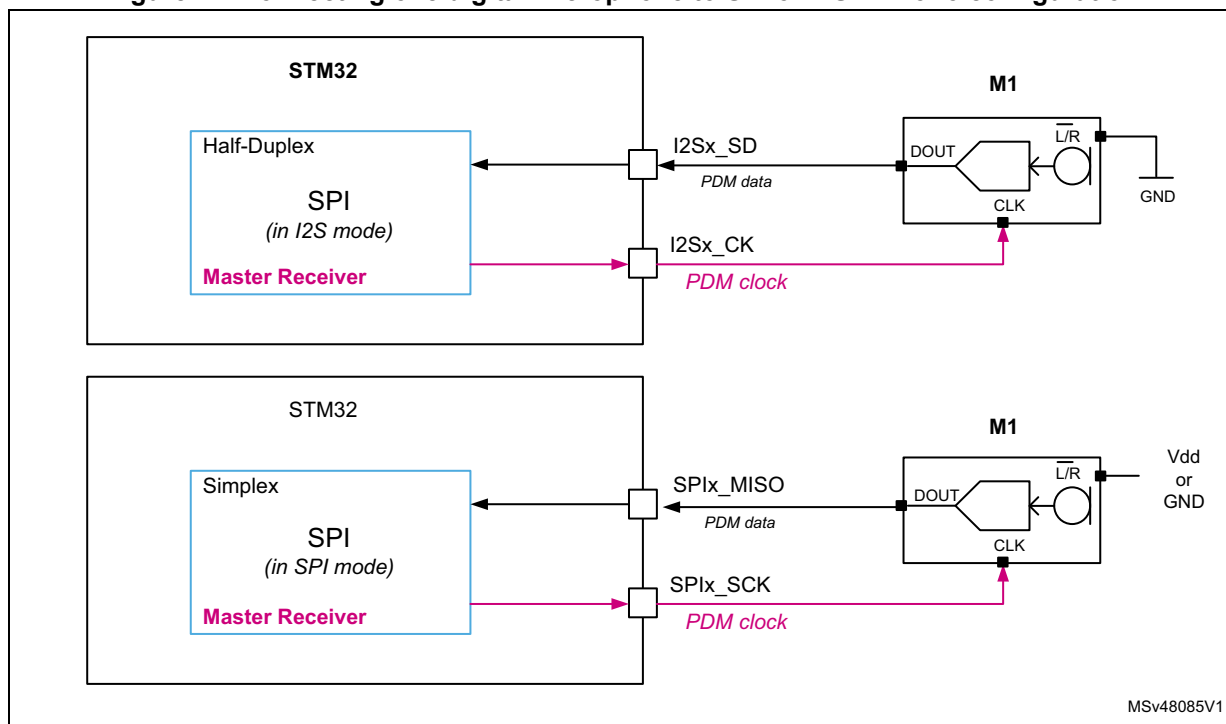
- The SPI protocol provides simple communication interface allowing the microcontrollers to communicate with external devices.
- The I2S protocol is widely used to transfer audio data from a microcontroller/DSP (Digital Signal Processor) to an audio codec, in order to play melodies or to capture sound from a microphone.

2.1.1 Mono configuration

A single digital microphone is connected to the SPI block. The SPI block can be configured either in SPI or in I2S mode.

In both cases, the SPI block is configured in master receiver mode. In this mode, the peripheral provides the clock to the digital microphone. The audio samples are acquired through the serial data pin.

Figure 11. Connecting one digital microphone to SPI or I2S in mono configuration



If the SPI protocol is used, the L/R channel selection (LR) pin of the microphone can be connected either to Vdd or to GND. The SPI clock polarity shall be aligned with the configuration of L/R input.

- If L/R = GND, then the SPI shall sample the incoming data using the rising edge of SPIx_SCK,
- If L/R = Vdd, then the SPI shall sample the incoming data using the falling edge of SPIx_SCK,

If the I2S protocol is used, it is recommended to set the L/R channel selection (LR) pin of the microphone to GND. By default the I2S protocol will sample the incoming data using the rising edge of I2Sx_CK. Note that the SPI-V2 block also offers the possibility to configure the sampling edge for the I2S protocol.

Data format

The samples acquired by the SPI block in I2S or SPI mode can be stored into the memory using either DMA or interrupt signaling.

The receive data register (SPIx_DR) will provide contiguous bits from the microphone like shown in the example hereafter for a 16-bit format:

Bit:	15	14	13	12	11	...	0
Content:	M1_b _N	M1_b _{N+1}	M1_b _{N+2}	M1_b _{N+3}	M1_b _{N+4}	...	M1_b _{N+15}

M1_b_{xx} represents the data bits from the digital microphone 1, and M1_b_N is the older bit.

Note: *The bit order of the received samples can be reversed if the interface is programmed in LSB first instead of MSB first. Peripherals generally support various data size which are not detailed here.*

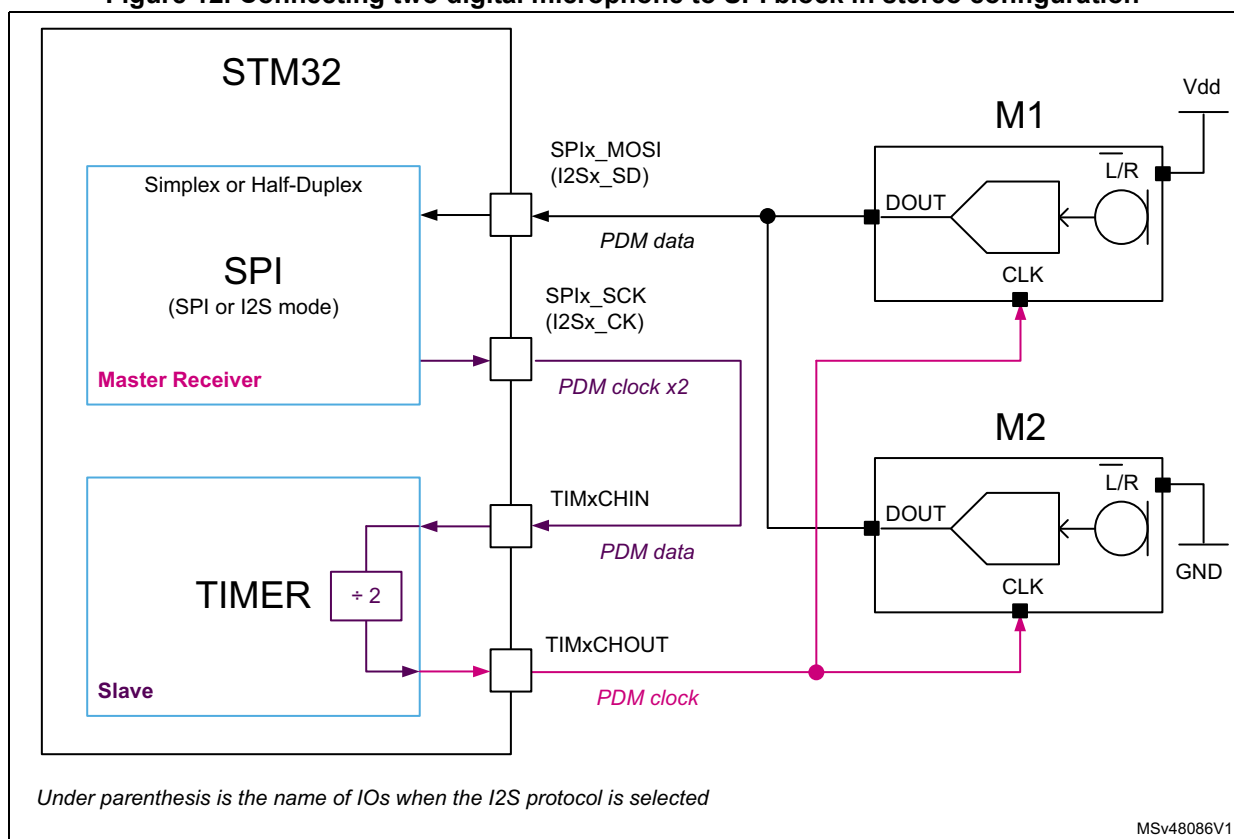
2.1.2 Stereo configuration

Two digital microphones can be connected to the SPI block using a timer. The SPI block can be configured either in SPI or in I2S mode.

In both cases, the SPI block is configured in master receiver mode. In this configuration, the SPI peripheral operates at twice the microphone frequency in order to read the data provided by both microphones, on the falling edge of its clock. This allows the two microphones to share a single data line.

The SPI block provides the clock to an embedded timer which divides the serial interface clock (SPIx_SCK or I2Sx_CK) by 2. The divided clock is delivered to the digital microphone. The audio samples are acquired by the I2S peripheral from the digital microphones data output pins.

Figure 12. Connecting two digital microphone to SPI block in stereo configuration



Using the timer as clock generator

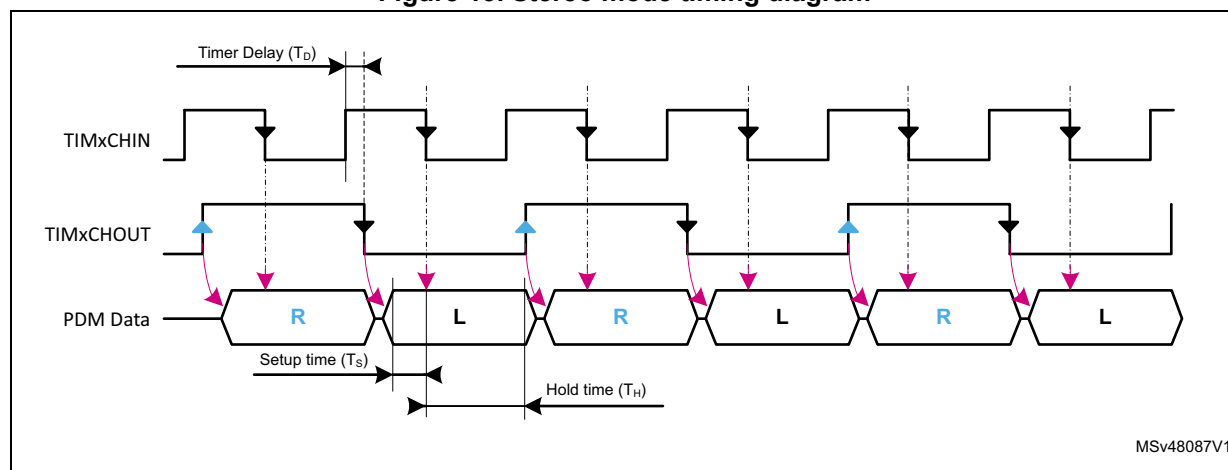
When the timer is used to generate the clock for the digital microphones, two points have to be taken into account:

- The application must insure that the delay introduced by the clock division performed by the timer still insures a margin in the setup time (T_S) of the samples provided by the microphones. For that purpose, the timer shall use a clock as fast as possible. The maximum delay (T_D) introduced by the timer between the input (TIMxCHIN) and the output clock (TIMxCHOUT) will be 5 clock cycles of the timer reference clock. The

timers generally use their APB clock or a multiple of their APB clock as reference. See [Figure 13](#).

- The application must insure that the peripheral providing the clock to TIMxCHIN input and the timer used for the division, are working with the same reference clock. If this rule is not respected, then from time to time the digital microphone will receive a clock having a longer or shorter period. This jitter may degrade the quality of the analog to digital conversion of the microphone.

Figure 13. Stereo mode timing diagram



Data format

The samples acquired by the SPI block in I2S or SPI mode can be stored into the memory using either DMA or interrupt signaling.

In this configuration, the data read from the microphones are interleaved bit per bit. The data stored into the SPIx_DR register will be interleaved as shown in the example hereafter for 16-bit format:

Bit:	15	14	13	12	11	...	0
Content:	M1_b _N	M2_b _N	M1_b _{N+1}	M2_b _{N+1}	M1_b _{N+2}	...	M2_b _{N+7}

M1_b_{xx} represents the data bits from the digital microphone 1, and M1_b_N is the older bit. M2_b_{xx} represents the data bits from the digital microphone 2, and M2_b_N is the older bit.

The SPI-V1 and SPI-V2 offer several data formats, for example the bit order of the received samples can be reversed if the interface is programmed in LSB first instead of MSB first, but it is important to notice that bits from microphone 1 (M1) and bits from microphone 2 (M2) will be always interleaved.

A software de-interleaving module is needed to separate the signal from the two microphones streams and to allow further processing like PDM-to-PCM conversion.

2.2 Serial Audio Interface (SAI)

The SAI integrated inside STM32 products, provides an interface allowing the microcontroller to communicate with external audio devices such as amplifiers, microphones, speakers, or audio processors. The SAI is composed of two independent sub-blocks that can operate synchronously or not. Each sub-block features its own audio clock generator.

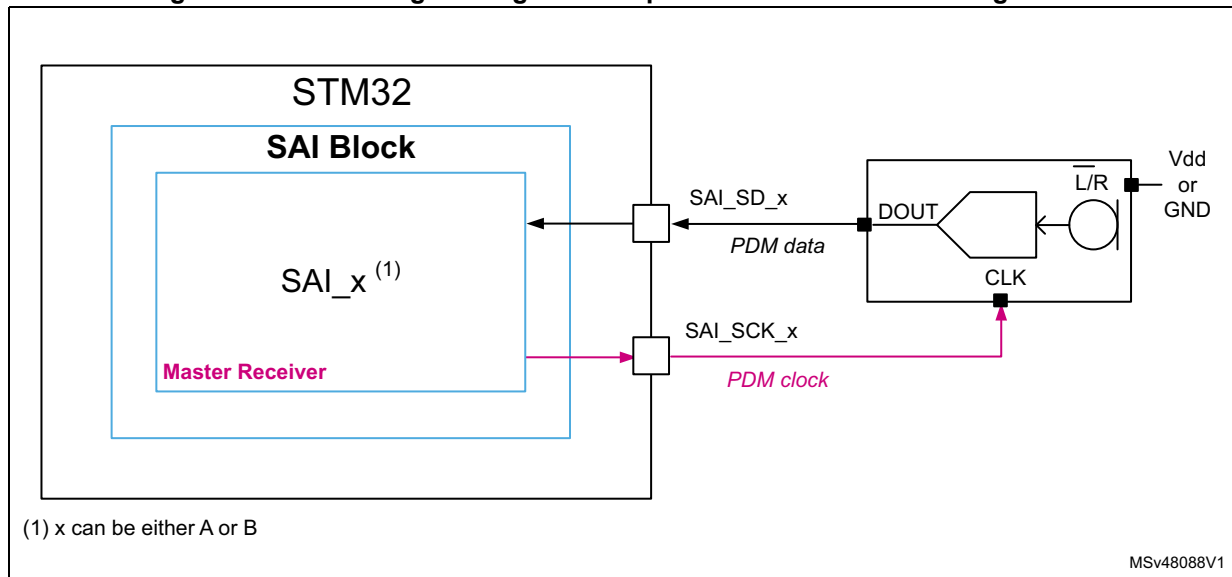
Some SAI also offer a dedicated PDM interface that can support up to 8 digital microphones.

2.2.1 Using a single sub-block

Mono configuration

The digital microphone is connected to one of the sub-blocks of the Serial Audio Interface (SAI) peripheral in mono configuration. The SAI sub-block is configured in master receive mode. In this configuration, the SAI sub-block provides the clock to the digital microphone. The audio samples are acquired by the SAI sub-block from the digital microphone data output (DOUT) pin through the Serial Data (SD) pin.

Figure 14. Connecting one digital microphone to SAI in mono configuration



The L/R channel selection (LR) pin of the microphone can be connected either to Vdd or to GND. The microphone outputs data on the rising or falling edges of the incoming clock signal depending on the selected channel. The sampling edge of the SAI clock shall be configured accordingly.

Note: *The other SAI sub-block is completely independent and can be used for another purpose: for example it can be connected to an external audio codec.*

Data format

The samples acquired by the SAI sub-block can be stored into the memory using either DMA or interrupt signaling.

The receive data register (SAI_ADR, SAI_BDR) will provide contiguous bits from the microphone like shown hereafter:

Bit:	k	k-1	k-2	k-3	k-4	...	0
Content:	M1_b _N	M1_b _{N+1}	M1_b _{N+2}	M1_b _{N+3}	M1_b _{N+4}	...	M1_b _{N+k}

M1_b_{xx} represents the data bits from the digital microphone 1, and M1_b_N is the older bit position inside SAI_ADR/BDR registers. For example if DS[2:0] is set to '010' (8 bits), then 'k' is equal to 7.

The size of data stored into the SAI_ADR/BDR registers depends on the programming of the data size (DS[2:0]), in addition SLOTSZ[1:0] must be forced to 0.

Note that the bit order of the received samples can be reversed if the interface is programmed in LSB first instead of MSB first.

Note: The data inside SAI_ADR/BDR registers are always right aligned.

Stereo configuration

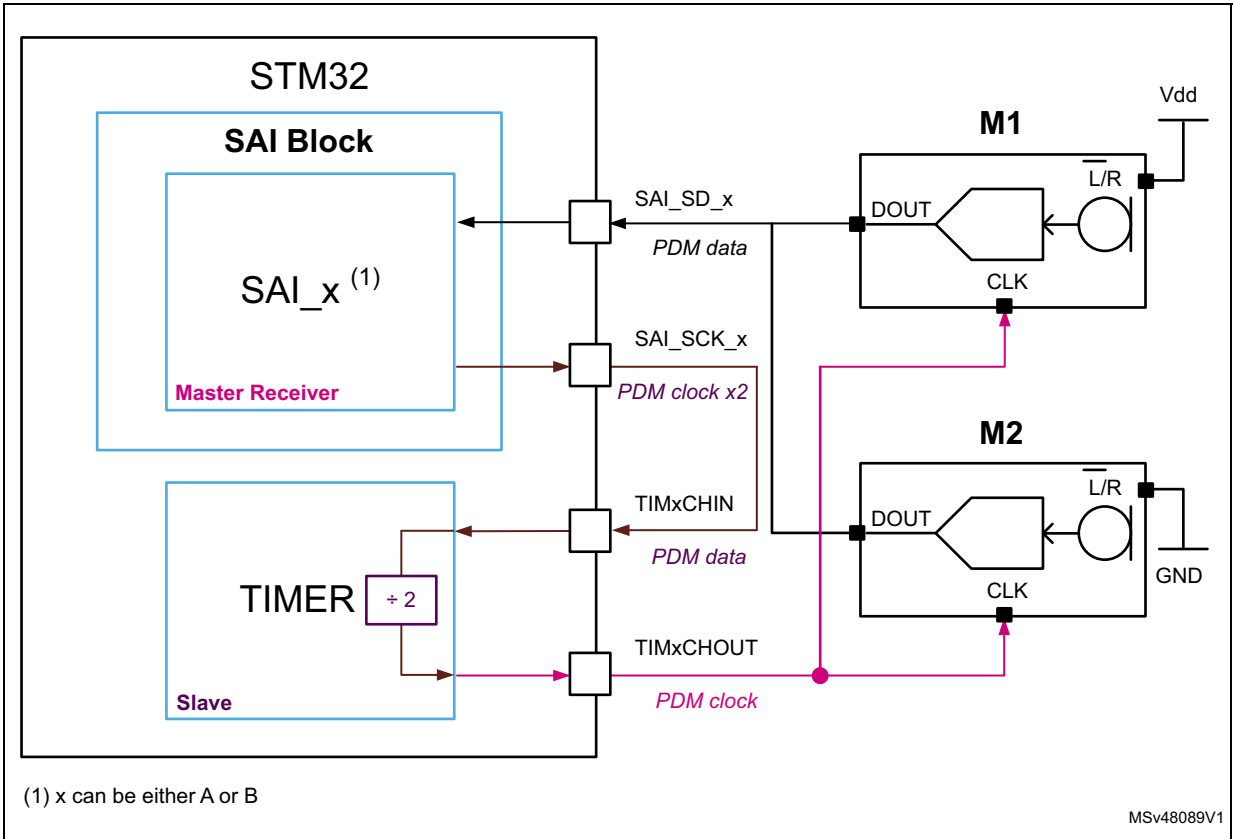
Two digital microphones can be connected to a single SAI sub-block using an internal timer. The SAI sub-block is configured in master receiver mode. In this configuration, the SAI sub-block operates at twice the microphone frequency in order to read the data provided by both microphones, on the same edge of its clock. This allows the two microphones to share a single data line (Refer to [Figure 15](#)).

The SAI sub-block provides the clock (SAI_SCK_x) to an embedded timer which performs a division by 2. The divided clock is delivered to the digital microphone.

Note: The other SAI sub-block is completely independent and can be used for another purpose: for example it can be connected to an external audio codec.

Timing diagram shown in [Figure 13](#) is also valid for this case. Refer also to section [Using the timer as clock generator](#) for recommendations concerning the use of the timer.

Figure 15. Connecting two digital microphones to SAI in stereo configuration using a SAI sub-block and a timer



Data format

The samples acquired by the SAI sub-block can be stored into the memory using either DMA or interrupt signaling.

In this configuration, the data read from the microphones are interleaved bit per bit. The data stored into the SAI_ADR/BD register will be interleaved as shown in the example hereafter:

Bit:	k	k-1	k-2	k-3	...	1	0
Content:	M1_b _N	M2_b _N	M1_b _{N+1}	M2_b _{N+1}	...	M1_b _{N+(K-1)/2}	M2_b _{N+(K-1)/2}

M1_b_{xx} represents the data bits from the digital microphone 1, and M1_b_N is the older bit. M2_b_{xx} represents data bits from the digital microphone 2, and M2_b_N is the older bit. 'k' is the bit position inside SAI_ADR/BDR registers, for example if DS[2:0] is set to '100' (16 bits), then 'k' is equal to 15.

The size of data stored into the SAI_ADR/BDR depends on the programming of the data size (DS[2:0]), in addition SLOTSZ[1:0] must be forced to 0.

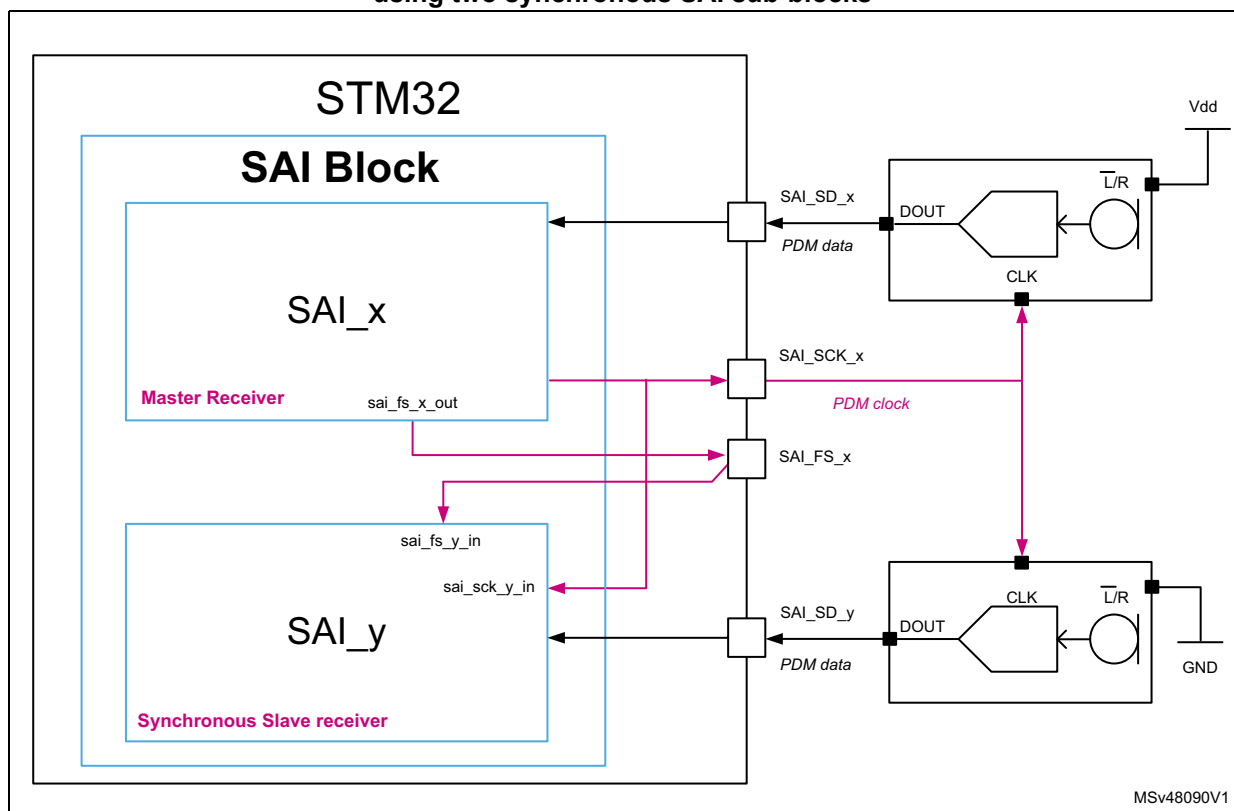
Note: The bit order of the received samples can be reversed if the interface is programmed in LSB first instead of MSB first, but it is important to notice that bits from microphone 1 (M1) and bits from microphone 2 (M2) will be always interleaved. The data inside SAI_ADR/BDR registers are always right aligned.

A software de-interleaving module is needed to separate the signal from the two microphones streams to allow further processing like PDM-to-PCM conversion.

2.2.2 Using two synchronous SAI sub-blocks

Two digital microphones can be connected to the SAI peripheral without using an embedded timer by synchronizing two SAI sub-blocks. Each Microphone is connected to a SAI sub-block. One of the SAI sub-block is configured in Master Receive mode while the other sub-block is configured in Synchronous Slave Receive mode. In this configuration, the SAI sub-block configured in Master mode will deliver the clock to the digital microphones and to the other SAI sub-block. The two SAI sub-blocks will read data synchronously from the microphones.

Figure 16. Connecting two digital microphone to SAI in stereo configuration using two synchronous SAI sub-blocks



Since the two microphones are not sharing one data line, the L/R channel selection (LR) pin of the microphones can be connected either to Vdd or to GND. The microphones output data on the rising or falling edges of the incoming clock signal depending on the selected channel. The clock polarity of each SAI sub-blocks should be configured accordingly.

Data format

The samples acquired by each SAI sub-block can be stored into the memory using either DMA or interrupt signaling. Up to 2 DMA channels are requested in that case.

For each SAI sub-block, the data format is similar to what is described for the SAI mono configuration.

2.2.3 Using PDM interface

The PDM interface is provided in order to support digital microphones. Up to 4 digital microphone pairs can be connected in parallel. The PDM interface also offers delay lines in order to perform micro-delays in each incoming bitstream, and thus making the beamforming application simpler. The depth of each delay cell is 8 bitstream samples.

The PDM function is intended to be used in conjunction with SAI_A sub-block, configured in Time Division Multiplexing (TDM) master mode. It cannot be used with SAI_B sub-block.

To reduce the memory footprint, the user can select the amount of microphones that the application needs. It is possible to select 2, 4, 6 or 8 microphones.

For example, if the application is using 3 microphones, the user has to select 4. In this case, the PDM data will be acquired through SAI_D1 and SAI_D2. SAI_D1 will receive the data from the first couple of microphones while SAI_D2 will receive data from the third microphone.

[Table 5](#) shows the recommended IO lines versus the number digital microphones to be connected.

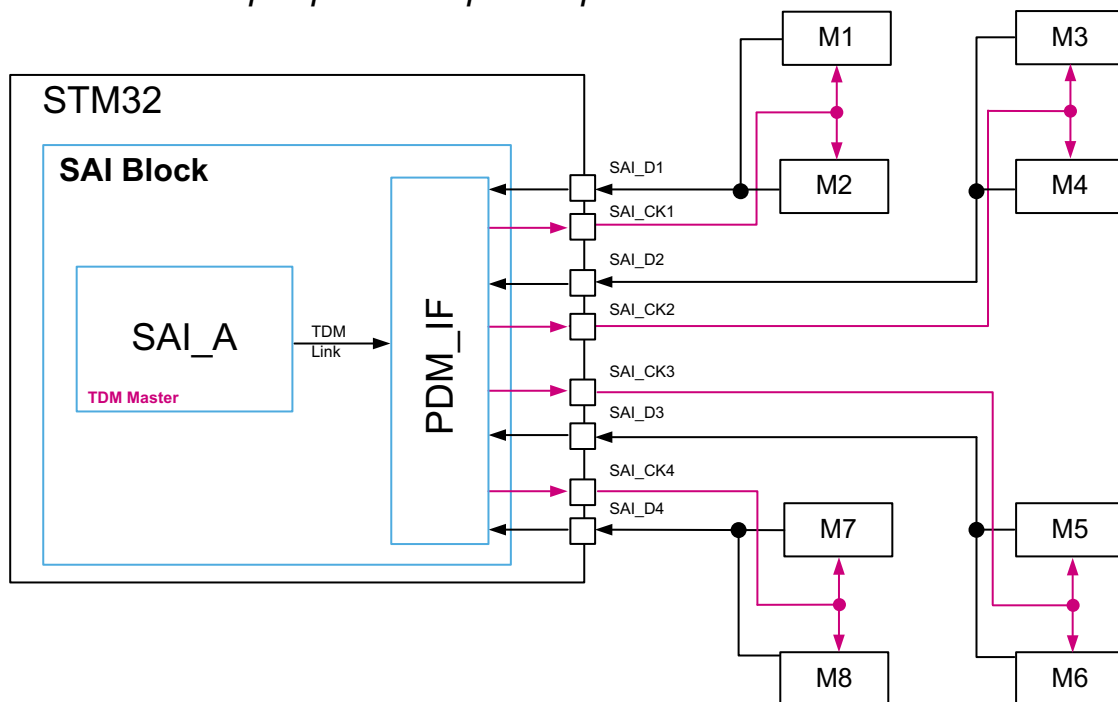
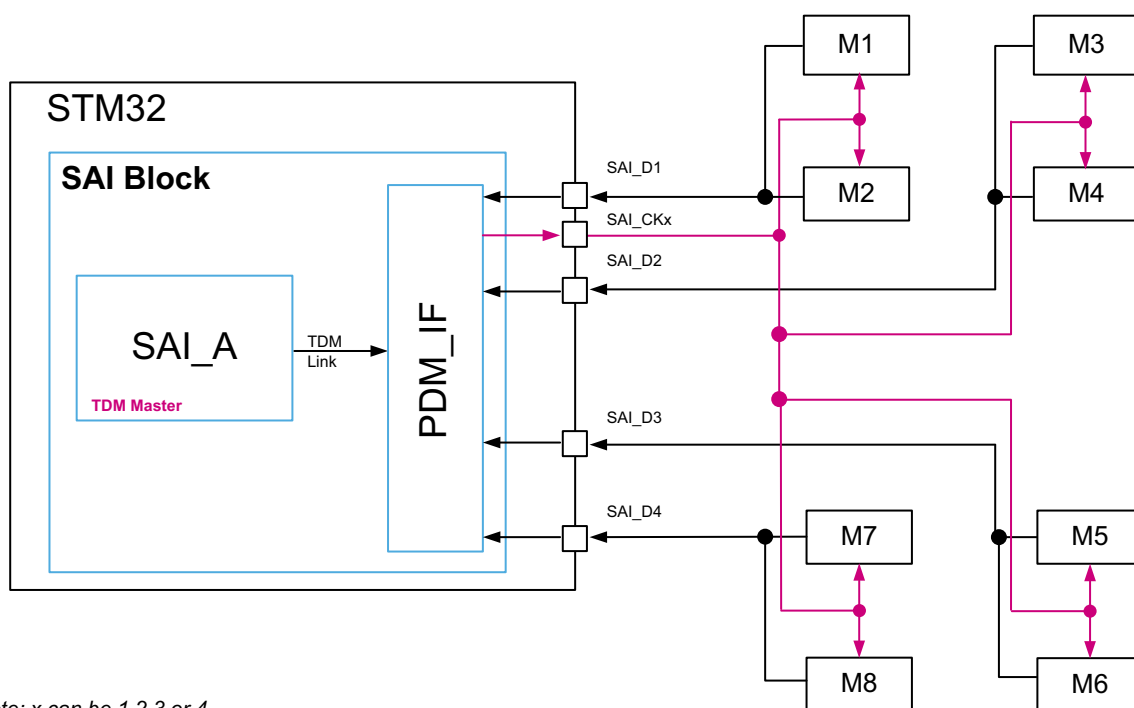
Table 5. recommended IO lines versus the number digital microphones to be connected

Number of Microphones	Recommended data IO lines	Recommended clock IO lines
1 or 2	SAI_D1	SAI_CK1,2,3 or 4
3 or 4	SAI_D1 and SAI_D2	
5 or 6	SAI_D1 and SAI_D2 and SAI_D3	
7 pr 8	SAI_D1 and SAI_D2 and SAI_D3 and SAI_D4	

The microphones can be clocked from the same SAI_CKx (x=0...3) or separately from different SAI_CK giving the user the flexibility to enable or disable the audio acquisition from particular microphones depending on the application.

The SAI operates at the number of microphones selected (2, 4, 6 or 8) times the microphone frequency to be able to read data from all the microphones in the application.

Figure 17. PDM interface capability on interfacing up to four microphone pairs

One clock output per microphone pair*Common clock for all microphones*

Note: x can be 1,2,3 or 4

MSv48091V1

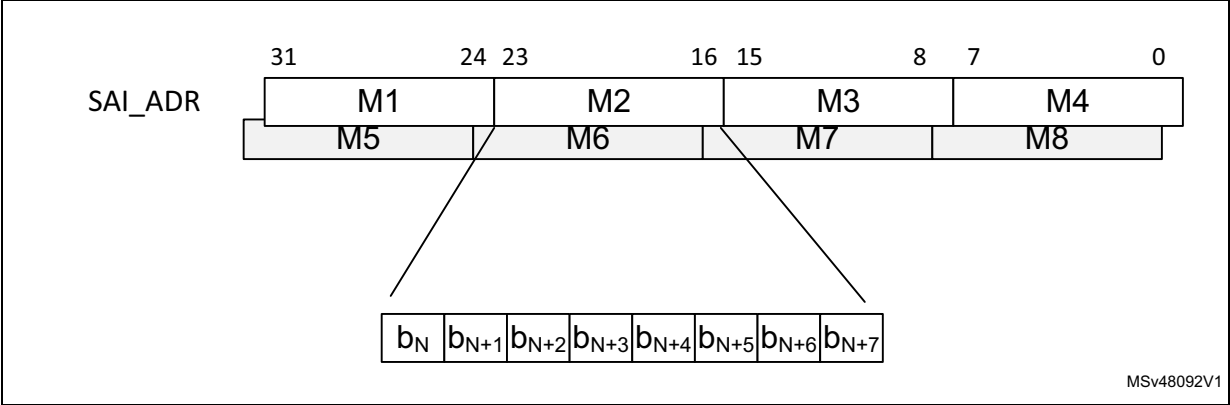
The PDM Interface of the SAI offers an optimal connection to the digital microphone, saving as much IOs as possible. In addition the PDM interfaces can separate the data of each microphone byte-wise, avoiding the de-interleaving operation.

Data format

The samples acquired by the SAI_A sub-block can be stored into the memory using a single DMA channel or interrupt signaling.

The receive data register (SAI_ADR) will provide 8 successive bits for each microphone as shown hereafter:

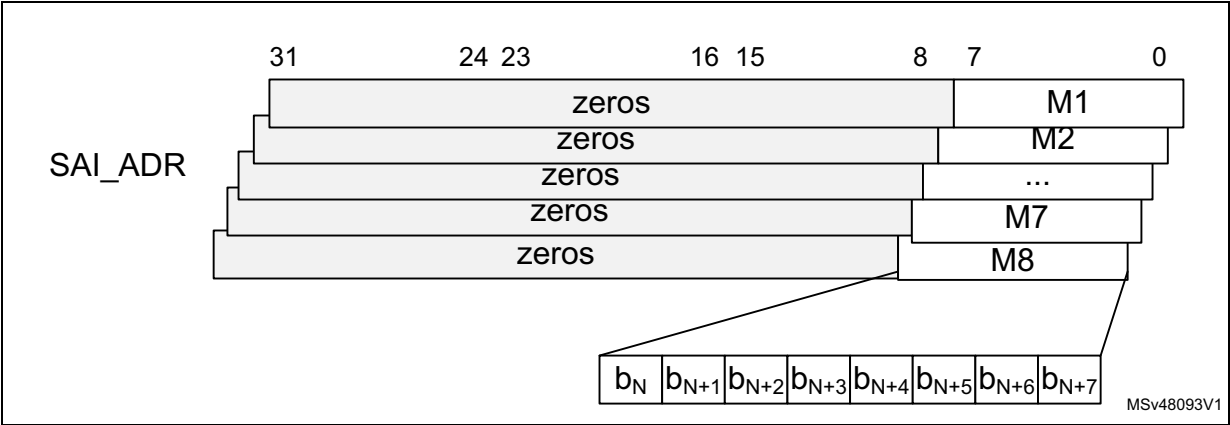
Figure 18. Data format when using the SAI PDM interface with a slot size of 32 bits, and 8 microphones



The size of data stored into the SAI_ADR depends on the programming of several SAI parameters, please refer to the SAI user specification on order to get more information about the data format, when the PDM interface is used.

Note that if the slot size is set to 8 bits, then the SAI_ADR will contain only on byte of data from one microphone. The SAI_ADR must be read 8 times to get one byte from the 8 microphones.

Figure 19. Data format when using the SAI PDM interface with a slot size of 8 bits, and 8 microphones



2.3 Digital filter for sigma delta modulators (DFSDM)

The DFSDM is a digital peripheral inside STM32 microcontrollers. It behaves like a standard ADC with a scalable speed/resolution and an external analog front-end.

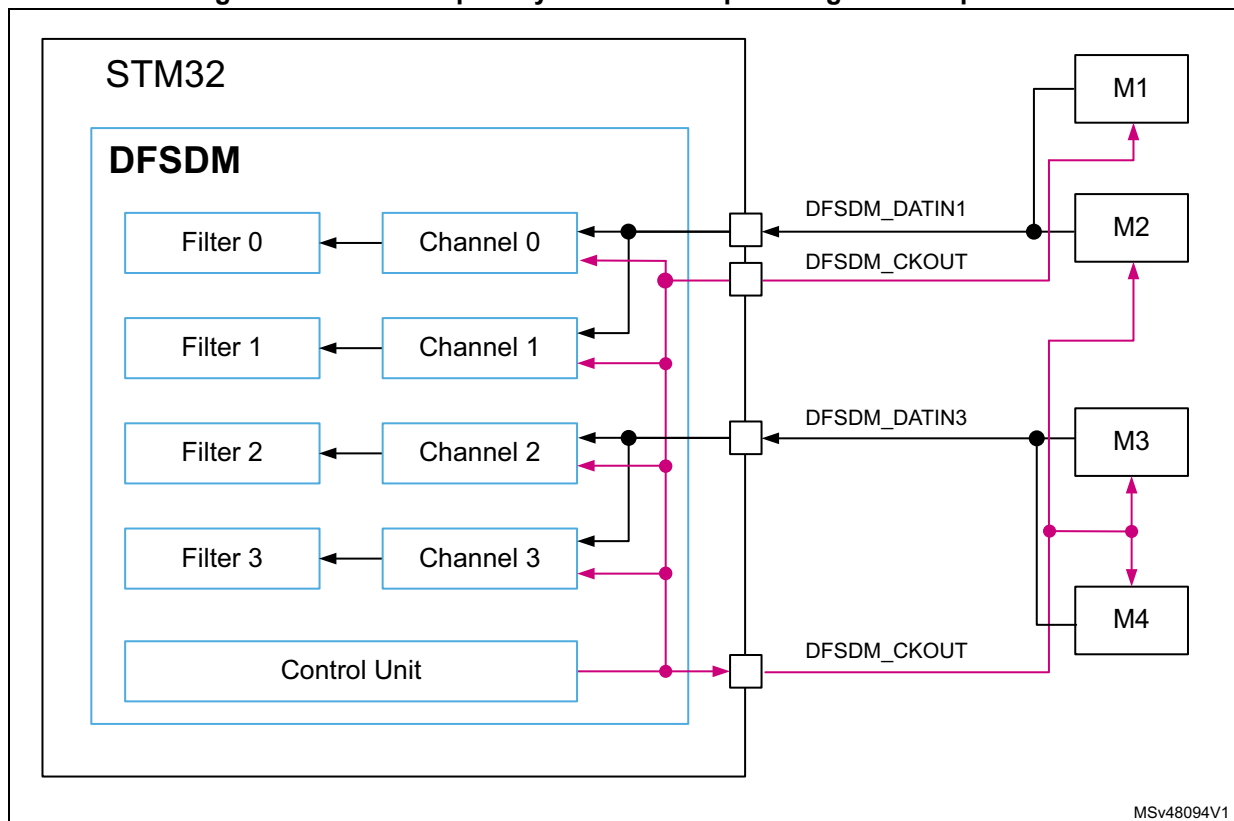
Digital MEMS microphones providing a PDM output data format can be directly connected to the DFSDM. The DFSDM provides filtered and decimated samples. Each filter has its own DMA channel as a consequence the samples of each microphone are separated. This allows the application to avoid heavy filtering and the de-interleaving operations. Finally some DFSDM blocks offer delay lines in order to perform micro-delays in each incoming bitstream, and thus making the beamforming application simpler. The depth of each delay line is at least equal to the decimation ratio.

It is possible to interface several digital microphones with a single DFSDM. It depends on the amount of filters integrated and the amount of interfaces.

The DFSDM features a clock output signal (DFSDM_CKOUT) to drive the digital microphones. The clock output has an adjustable division factor. The DFSDM_CKOUT can be the output on different IOs giving the user the flexibility to enable or disable the audio acquisition from particular microphones depending on the application. The configuration shown in [Figure 20: DFSDM capability to interface up to 4 digital microphones](#) is usually used in low power applications: the digital microphone M1 can work while the others switched in low-power mode as they are not clocked. It is also possible to power-off M3 and M4, to save more energy.

When all microphones need to be activated, then the same clock is provided to all microphones via two different PADs.

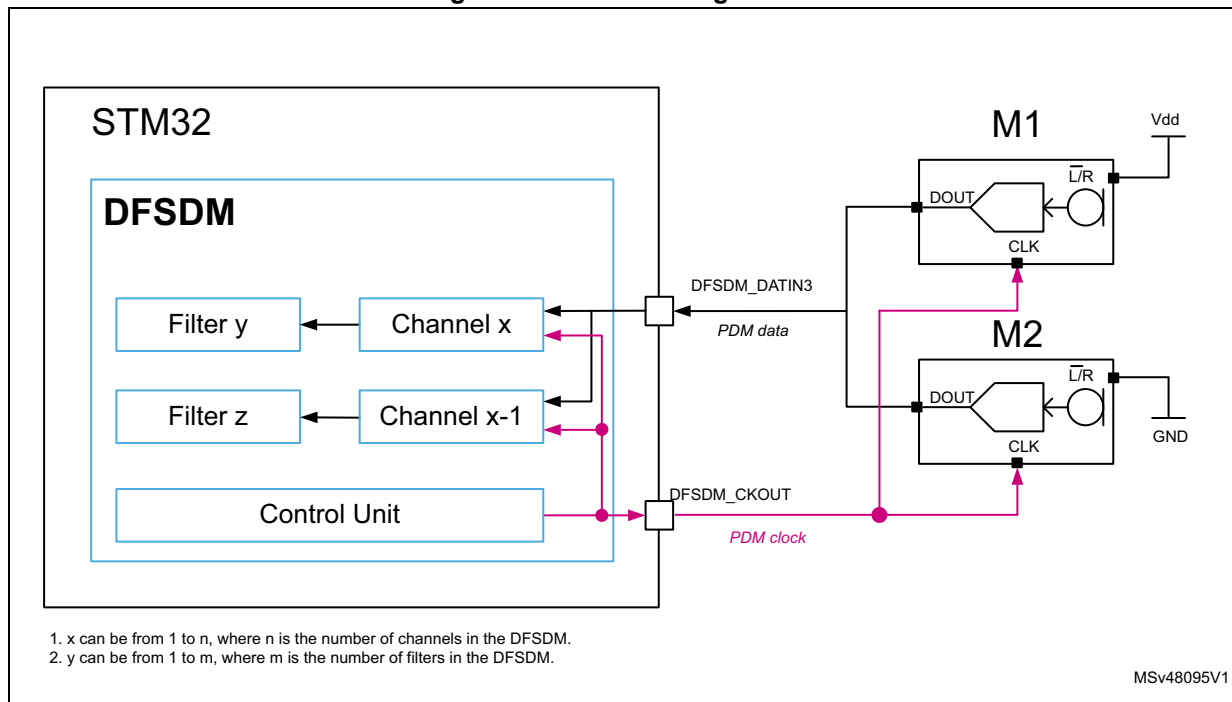
Figure 20. DFSDM capability to interface up to 4 digital microphones



2.3.1 Stereo configuration

Two digital microphones are connected to the DFSDM in stereo configuration. The DFSDM must enable two consecutive channels driven by an internal clock. The DFSDM peripheral provides an external clock (DFSDM_CKOUT) to drive the digital microphones. In this configuration, the DFSDM is programmed in order to allow the channels x and $x-1$ to receive the data from DFSDM_DATIN x pin. Each channel will be reading data on a different clock edge allowing the two microphones to share a single data line. Each channel will then redirect the acquired data to a different DFSDM filters to be processed.

Figure 21. Stereo configuration



Data format

The samples acquired by each digital filter can be stored into the memory using a dedicated DMA channel or interrupt signaling. There is no interleaving, each filter provides the converted samples of one microphone. The amount of requested DMA channels is equal to the amount of the digital filters activated.

2.4 Clocking considerations

2.4.1 The digital microphone clock

The clock provided to the digital microphone has several functions:

- When the clock is absent or at very low frequency (refer to product datasheet), then the digital microphones switch into low-power mode.
- When the clock frequency is low (usually between 400 and 800 kHz) the microphone is working in low-power mode. It means that there is a small degradation of its performance in order to reduce as much as possible the power consumption. This

feature is not available in all microphones. The clock shall be as clean as possible, with low-jitter.

- Finally, when the clock frequency is higher (about 1 to 4.8 MHz), the microphone is working in its nominal mode. The clock shall be as clean as possible, with low-jitter. Note that the accepted frequency range is dependent of the microphone, refer to the datasheet of the product for details.

Note as well that when a digital microphone just exits from low-power mode because its clock becomes active or in the detected frequency range, it will take several milliseconds before providing samples with the expected quality.

[Table 6](#) shows some applications examples and the associated microphone clock frequency.

Table 6. Applications examples and the associated microphone clock frequency

Use-cases	Clock frequency provided to the digital microphone
Sound capture	Between 400 and 800 kHz
Speech application	Between 1 and 1.5 MHz
High quality audio application	Between 2.4 and 4.8 MHz

2.4.2 The Peripheral clocks

In order to select an implementation, the application shall also consider the clocking possibilities that the microcontroller offers to the SPI, SAI or DFSDM. It is also important to check the capabilities of the clock generator embedded into the SPI, SAI and DFSDM.

Generally the peripheral audio blocks have two clock inputs:

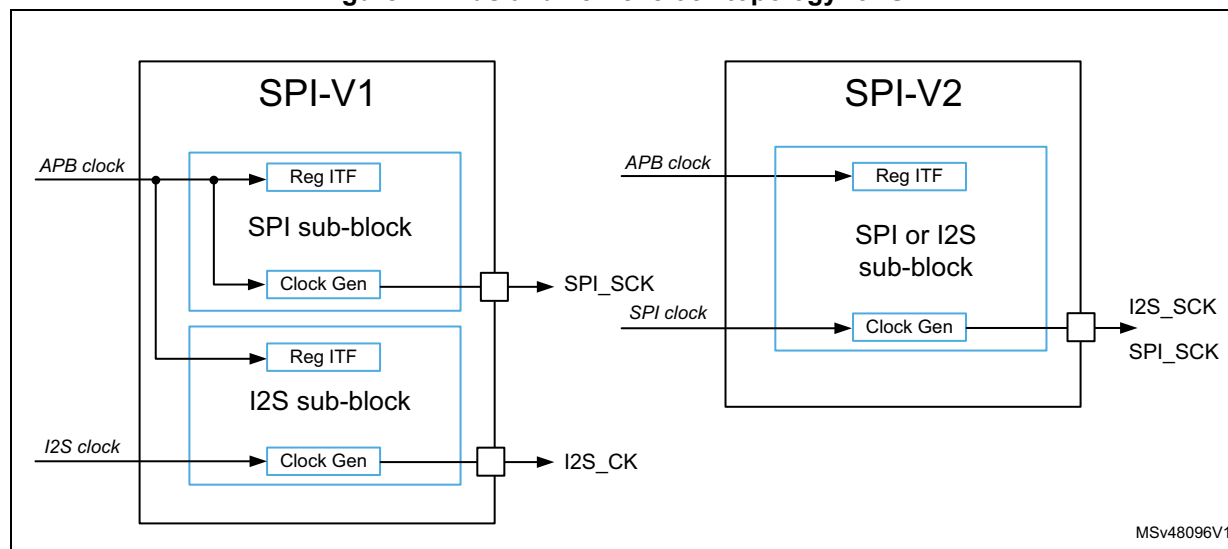
- A clock used for the register interface control (i.e. APB clock)
- A clock used for the timing generation of the serial interface, named kernel clock.

According to the peripheral and protocol selected, the peripheral clock generator will use the bus interface or a dedicated reference clock. For example the SPI-V1 will use the APB clock as reference clock if the SPI protocol is used, while the I2S protocol is using an I2S clock.

Controlling the frequency of the APB clock is generally less flexible than using a dedicated clock. One of the reason is that the APB clock impacts all the peripherals connected to this APB bus. However, in the case where a timer is used to support a stereo microphone configuration, using the APB (i.e. same clock than the timer) solves one of the issue listed in the [Using the timer as clock generator](#).

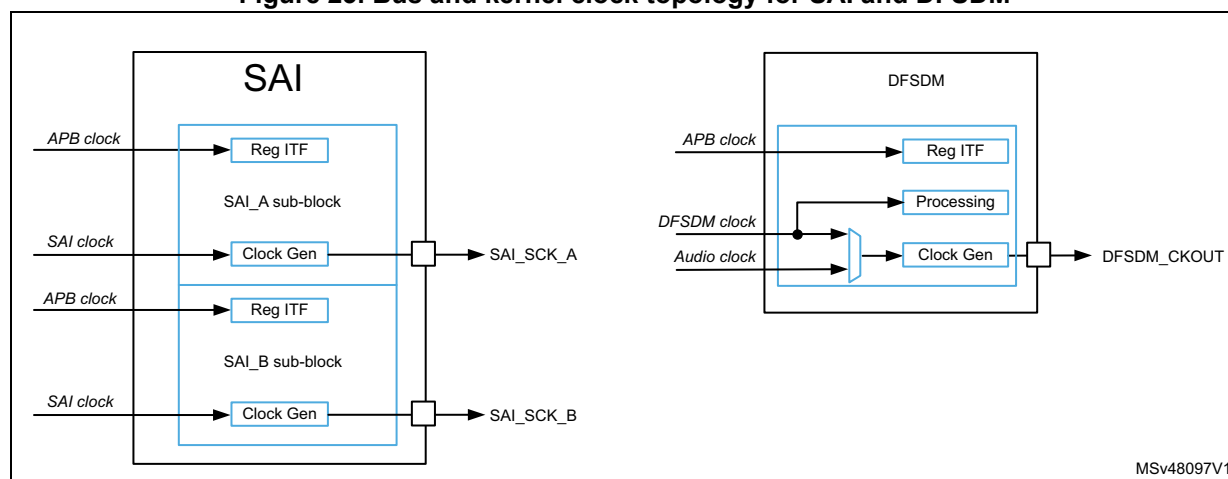
Some microcontrollers also offer the possibility to provide a copy of the APB clock as kernel clock.

Figure 22. Bus and kernel clock topology for SPI



Other peripherals (SPI-V2, SAI and DFSDM) always offer a dedicated kernel clock for the clock generation. This option is more flexible and makes the wanted clock frequency independent from the bus interface frequency.

Figure 23. Bus and kernel clock topology for SAI and DFSDM



Note: The DFSDM offers the possibility to select either an independent peripheral clock (audio clock) or to select the DFSDM clock which is synchronous of the APB clock.

2.5 GPIOs number considerations

This section helps the user to choose the most suited STM32 peripheral (SPI2S, SAI, DFSDM) to interface digital microphones in a particular application according to GPIOs number.

[Table 7](#), [Table 8](#) and [Table 9](#) shows the hardware used to connect respectively one, two and four digital microphones to the different audio and serial interfaces.

- The column Number of GPIOs indicates the number of GPIO necessary to connect digital microphones to an audio interface.
- The column Timer indicates whether the audio interface needs a Timer to deliver the appropriate clock to the microphones.

Table 7. Hardware used to connect one digital microphone

Interface	Number of GPIOs	Timer
SPI	2	No
I2S	2	No
SAI	2	No
DFSDM	2	No

Table 8. Hardware used to connect two digital microphones

Interface	Number of GPIOs	Timer
SPI	4	Yes
I2S	4	Yes
SAI	4	Yes
SAI (Two synchronous sub-blocks)	4	No
SAI with PDM interface	2	No
DFSDM	2	No

Table 9. Hardware used to connect four digital microphones

Interface	Number of GPIOs	Timer
SAI (Two synchronous sub-blocks)	6	Yes
SAI with PDM interface	3	No
DFSDM	3	No

3 Digital signal processing

This section presents two ways to convert PDM data into PCM data:

The first is a software solution which is the PDM audio software decoding library and the second is hardware solution using the DFSDM peripheral filters.

3.1 PDM audio software decoding Library

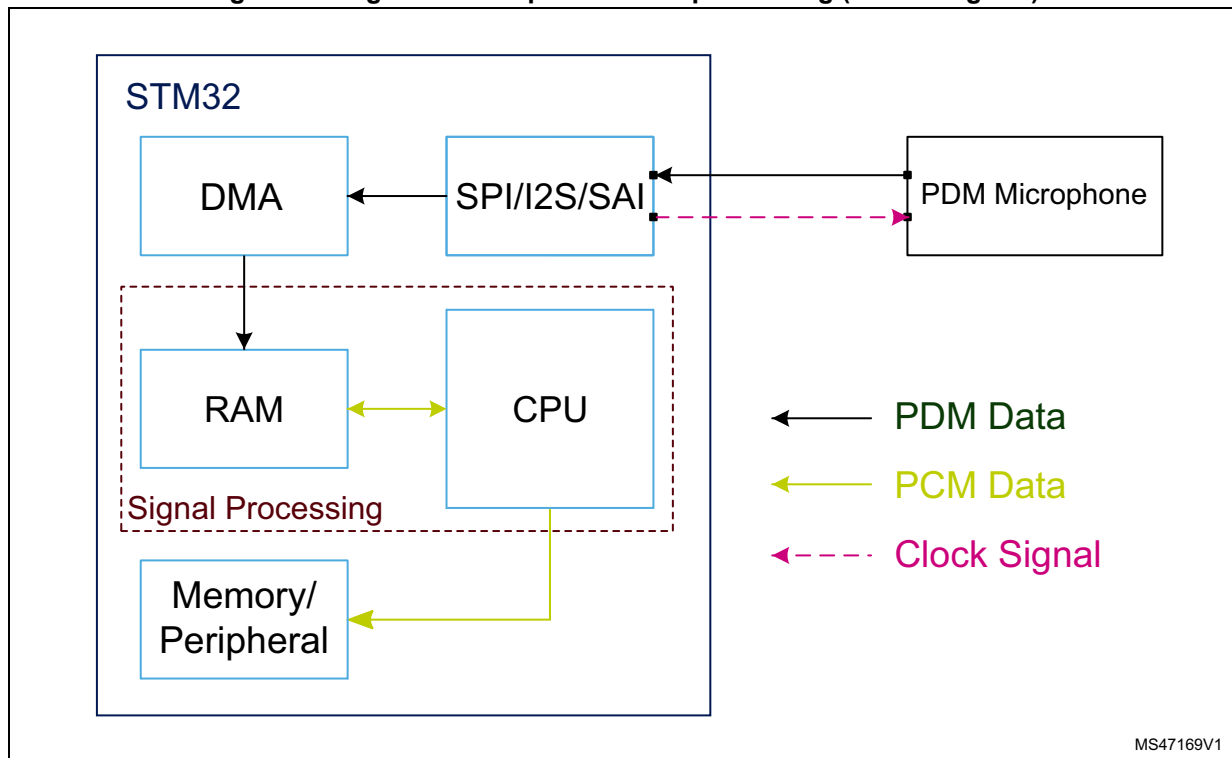
3.1.1 Overview

PDM audio software decoding Library is an optimized software implementation for PDM signal decoding and audio signal reconstruction when connecting digital MEMS microphones with an STM32 microcontroller. This library implements several filters for the 1-bit PDM high frequency signal output from a digital microphone and transforms it into a 16-bit PCM at a proper audio frequency.

3.1.2 Digital data flow

The digital MEMS microphone outputs a PDM signal, which is a high frequency (1 to 3.25 MHz) stream of 1-bit digital samples. The PDM data is acquired by a serial interface embedded in the STM32. This data is transferred through DMA (thus reducing the software overhead) to a system RAM buffer to be processed. After the conversion, the PCM raw data 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...)

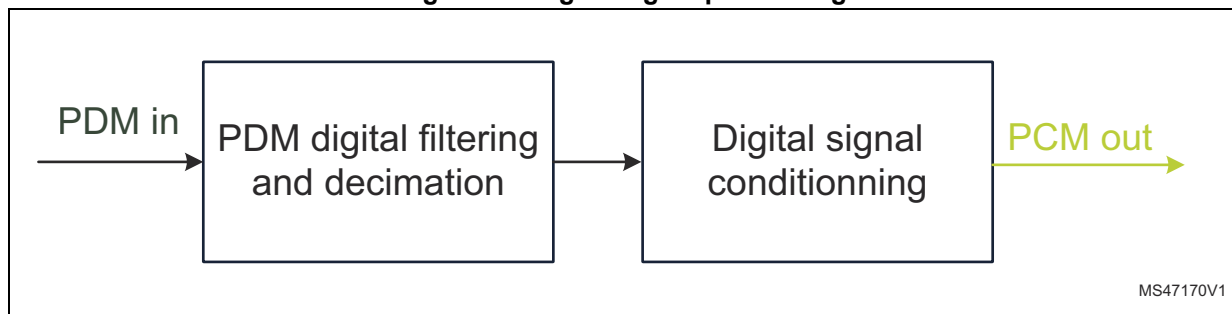
Figure 24. Digital data acquisition and processing (block diagram)



3.1.3 Digital signal processing

The PDM audio software decoding Library offers a two steps digital signal processing: PDM digital filtering and decimation and Digital signal conditioning.

Figure 25. Digital signal processing



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

On the second step, the digital audio signal resulting from the previous filter pipeline is further processed for proper signal conditioning implementing a low pass filter and a high pass filter. Both these filters can be enabled/disabled and configured (cut-off frequencies) by using the filter initialization function.

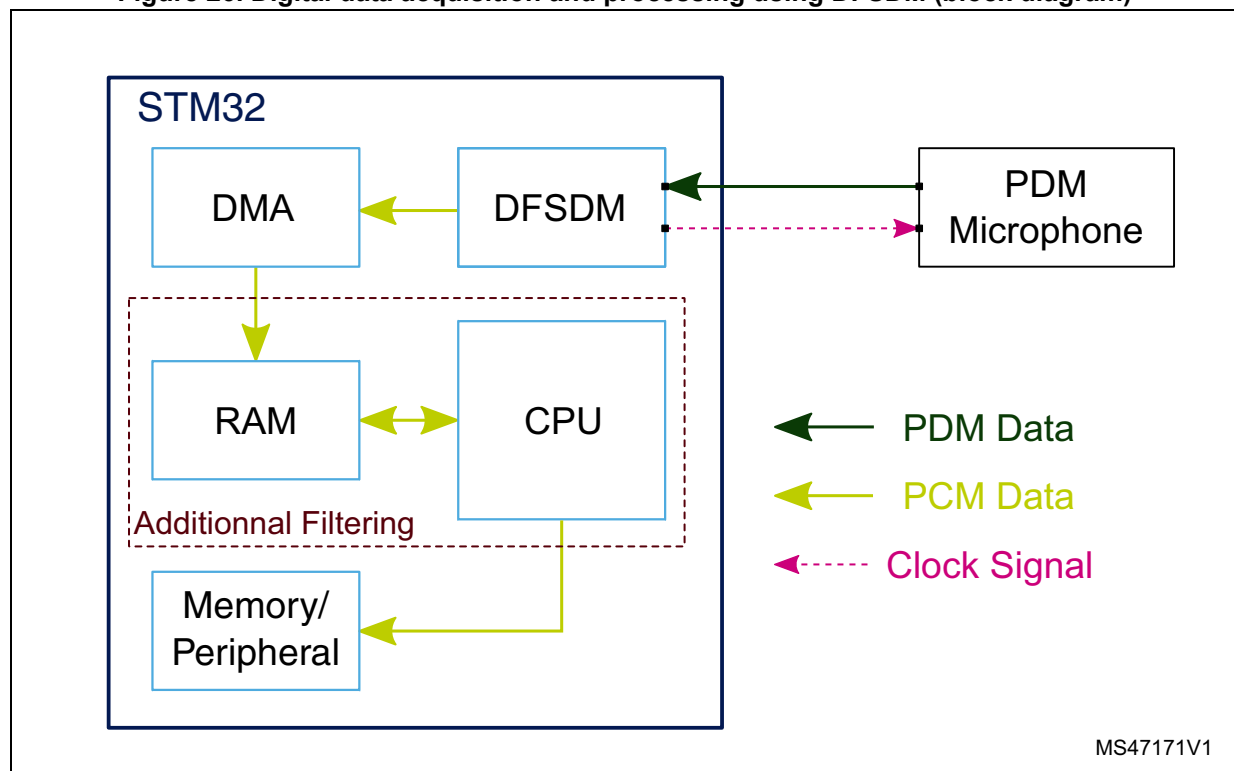
Note: *In stereo configuration, if two microphones are sharing one data line and the PDM data is interleaved bit per bit, a software de-interleaving step is needed to separate the signal from the two microphones before proceeding to the PDM to PCM conversion.*

3.2 DFSDM filters for digital signal processing

3.2.1 Digital data flow: acquisition and processing

The digital MEMS microphone outputs a PDM signal, which is a high frequency (1 to 3.25 MHz) stream of 1-bit digital samples. The data is acquired by the DFSDM serial transceiver that provides connection to the external Sigma-Delta modulator of the digital microphone. The digital filters perform CPU-free filtering that averages the 1-bit input data stream from the SD modulator into a higher resolution and a lower sample rate. This data is transferred through DMA (thus reducing the software overhead) to a system RAM buffer to be further filtered. After that, the PCM raw data 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...).

Figure 26. Digital data acquisition and processing using DFSDM (block diagram)



4 Examples of configuration based on STM32CubeMX

The following section guides the user through the different steps needed to create a basic audio application which acquires PDM data from digital microphones in mono or stereo modes then convert it into PCM data.

The main step consists in selecting the right hardware configuration and generate the C initialization code with the STM32CubeMX tool. In a second step, add the appropriate user code to the generated project.

Finally, user could refer to STM32Cube_FW packages audio examples and to "X-CUBE-MEMSMIC1" package to complete the needed user code for each example.

This section assumes that:

- The user wanted to get a PCM stream of 16 or 48 kHz
- The digital microphones receive a clock 64 times higher than the PCM stream frequency (oversampling by 64).

4.1 Example 1: Interfacing digital microphones in mono or stereo mode with I2S, SPI or a single SAI sub-block

This example is based on the NUCLEO-F413ZH board and using external digital microphones that will be connected to either I2S, SPI or SAI.

For 16 KHz sampling rate the bitclock generated by the interface must be 1.024 MHz for mono mode and 2.048MHz for stereo mode.

For 48 KHz sampling rate the bitclock generated by the interface must be 3.072 MHz for mono mode and 6.144MHz for stereo mode.

4.1.1 Hardware configuration using STM32CubeMX

I2S

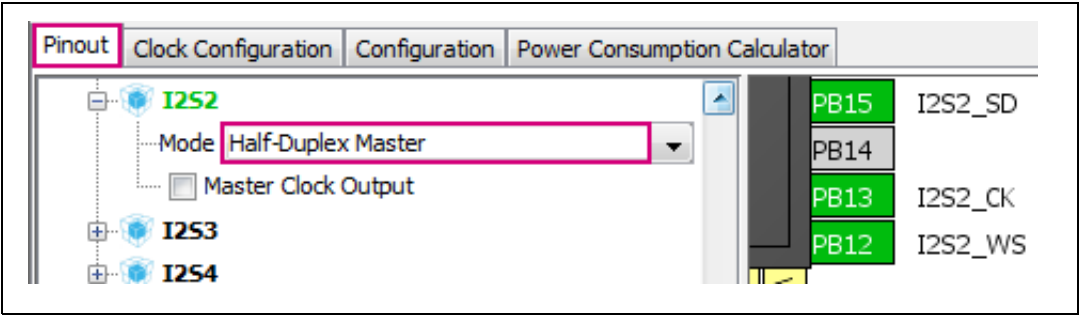
GPIO and Pin configuration

From the listed hardware in the Pinout tab, choose the I2S2 peripheral and configure it in Half-Duplex Master mode.

[Figure 27](#) shows how to enable the I2S2 in Half-Duplex Master mode.

The enabled pins, I2S2_SD, I2S2_CK and I2S2_WS, will be highlighted in green once the I2S peripheral GPIOs are correctly configured.

Figure 27. I2S GPIO pin configuration



Note: In this example the I2S2_WS pin won't be used. This pin can be released and brought back as a GPIO by applying a minor modification in code of the MSP Initialization file (stm32f4xx_hal_msp.c) after generating the project.

Clock configuration:

This section describes the different clock configuration for I2S in mono and stereo modes for 16 kHz and 48 kHz streams. HSE = 8 MHz is used as source clock.

In stereo mode, TIMER and I2S must use a clock coming from the same reference, for this reason the PLLR selected as clock source for I2S.

Note: The **accuracy** is the error between expected audio frequency and the real one.

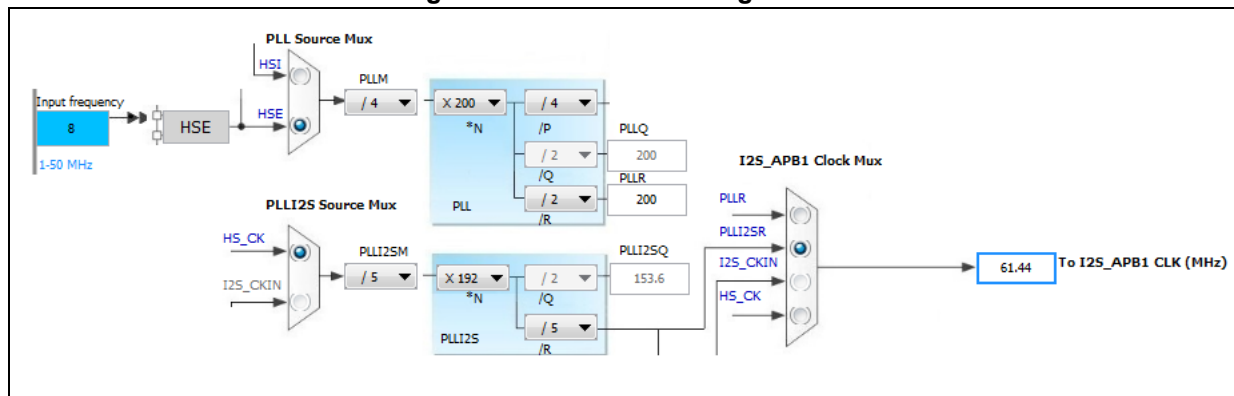
Table 10. I2S2 clock configuration and accuracy

Target Audio Frequency (kHz)	Microphone mode	I2S_APB1 Clock Mux source	DivM	PLLN	DivR	DivP	I2S clock (MHz) ⁽¹⁾	Accuracy (ppm)
16	Mono	PLLI2SR	5	192	2	-	153.60	0
		PLLI2SR	5	192	5	-	61.44	0
	Stereo (with timer)	PLLR	7	344	6	4	65.52	-186
48	Mono	PLLI2SR	5	192	2	-	153.60	0
		PLLI2SR	5	192	5	-	61.44	0
	Stereo (with timer)	PLLR	7	344	4	4	98.29	-186

1. In order to work properly, the I2S clock must be higher than its APB clock.

Figure 28 shows an example of I2S clock configuration for mono mode.

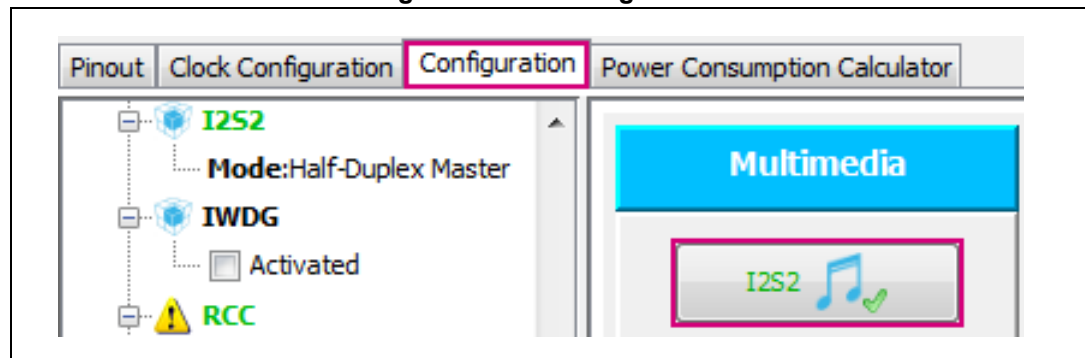
Figure 28. I2S2 clock configuration



I2S configuration

Select the configuration tab then click on the I2S2 button as shown in [Figure 29](#).

Figure 29. I2S configuration



a) I2S parameter settings:

In the I2S configuration window, select the parameter settings tab and then configure the parameters.

The I2S is configured according to the following conditions:

- Selected Audio Frequency = AUDIO_SAMPLING_FREQUENCY
- Communication Standard = MSB First (Left Justified): The I2S reads data on the falling edge of the clock. In mono configuration the LR pin of the microphone must be connected to GND.

[Figure 30](#) shows an example of I2S configuration for an Audio Sampling Frequency = 16 kHz in Mono mode.

Figure 30. I2S parameter settings

Configure the below parameters :

Search :

☒ Parameter Settings ☒ User Constants ☒ NVIC Settings ☒ GPIO Settings ☒ DMA Settings

☒ Generic Parameters

Transmission Mode	Mode Master Receive
Communication Standard	MSB First (Left Justified)
Data and Frame Format	16 Bits Data on 32 Bits Frame
Selected Audio Frequency	16 KHz
Real Audio Frequency	16.0 KHz
Error between Selected and Real	0.0 %

☒ Clock Parameters

Clock Source	I2S PLL Clock
Clock Polarity	Low

b) I2S DMA configuration

In this example the DMA handles the PDM data transfer from the I2S to the Memory.

In the I2S configuration window, select the DMA Settings Tab and add a DMA request.
[Figure 31](#) shows how to enable the DMA.

Figure 31. I2S DMA settings

☒ Parameter Settings ☒ User Constants ☒ NVIC Settings ☒ DMA Settings ☒ GPIO Settings

DMA Request	Stream	Direction	Priority
SPI2_RX	DMA 1 Stream 3	Peripheral To Memory	High

Click on the recently created DMA request and refer to the DMA request settings section to complete the DMA configuration.

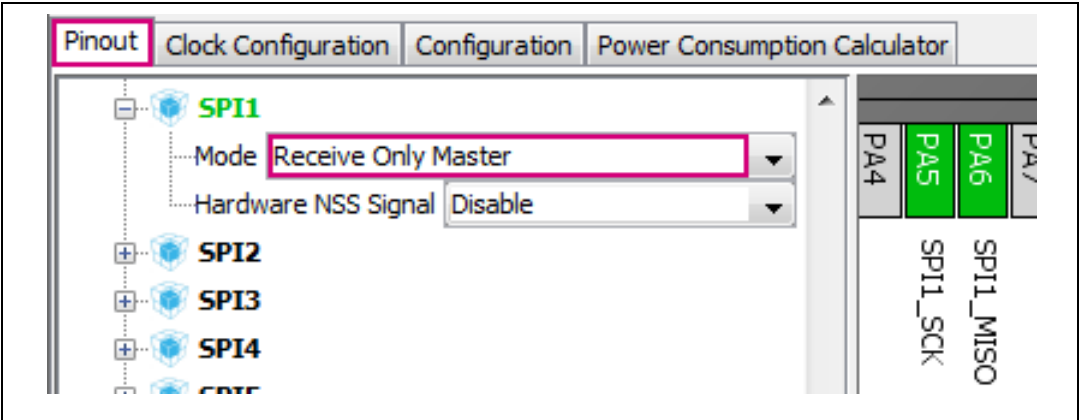
SPI

GPIO and pin configuration

From the listed hardware in the Pinout tab, choose the SPI1 peripheral and configure it in Receive Only Master mode.

[Figure 32](#) shows how to enable SPI1 in Receive Only Master mode. The enabled pins, SPI1_SCK and SPI1_MISO, are highlighted in green when the SPI peripheral GPIOs are correctly configured.

Figure 32. SPI GPIO and pin configuration



SPI clock configuration

This section describes the different clock configuration for SPI in mono and stereo modes for 16 KHz and 48 KHz streams. HSE = 8MHz is used as source clock.

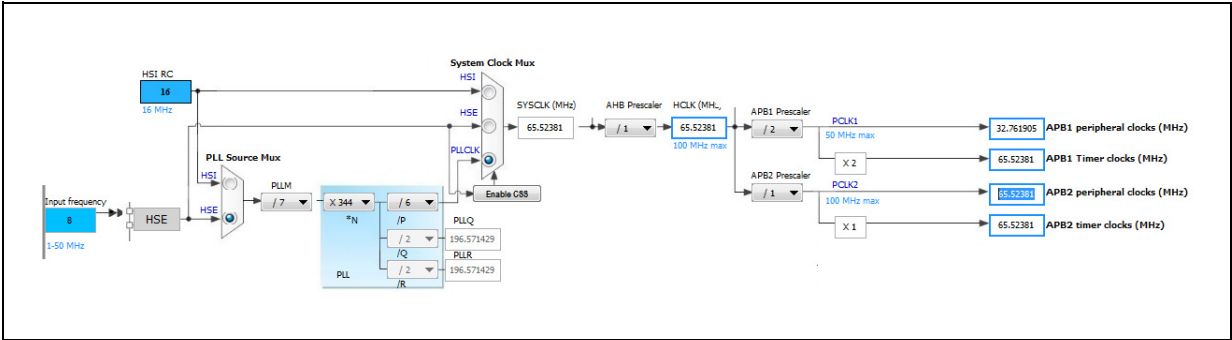
The APB2 clock is used as reference clock for the SPI.

Table 11. SPI clock configuration and accuracy

Target Audio Frequency (kHz)	Microphone mode	DivM	PLLN	DivR	DivP	SPI clock (MHz)	Accuracy (ppm)
16	Mono	7	344	-	6	65.52	-186
	Stereo (with timer)	7	344	-	6	65.52	-186
48	Mono	7	344	-	4	98.29	-186
	Stereo (with timer)	7	344	-	4	98.29	-186

Figure 33 shows an example of SPI clock configuration for mono mode.

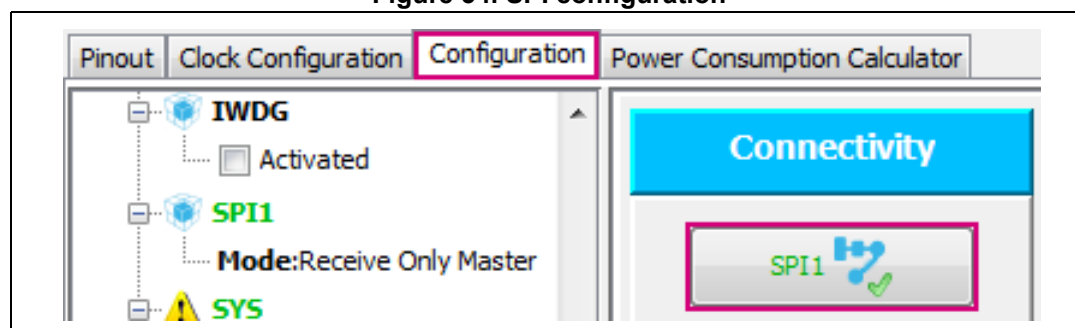
Figure 33. SPI clock configuration



SPI configuration

Select the configuration tab then click on the SPI button as shown in [Figure 34](#).

Figure 34. SPI configuration



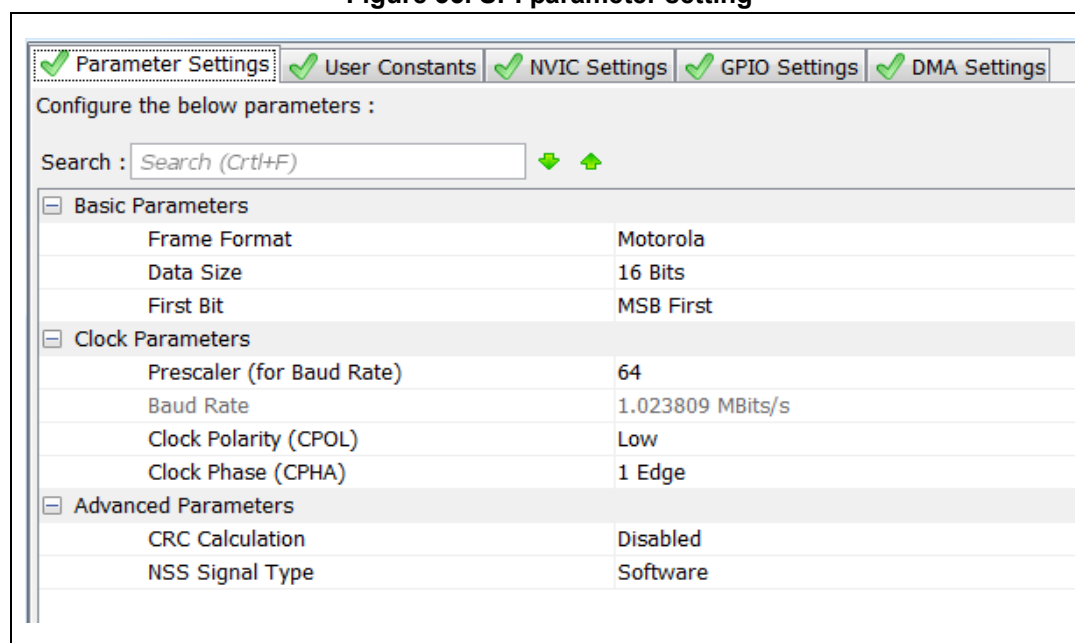
a) SPI parameter settings

In the SPI configuration window, select the parameter Settings and tab then configure the parameters.

In mono mode the Clock Polarity and the Clock Phase are chosen according to the state of L/R pin of the microphone

[Figure 35](#) shows an example of SPI configuration with an Audio Frequency = 16 kHz in Mono mode.

Figure 35. SPI parameter setting

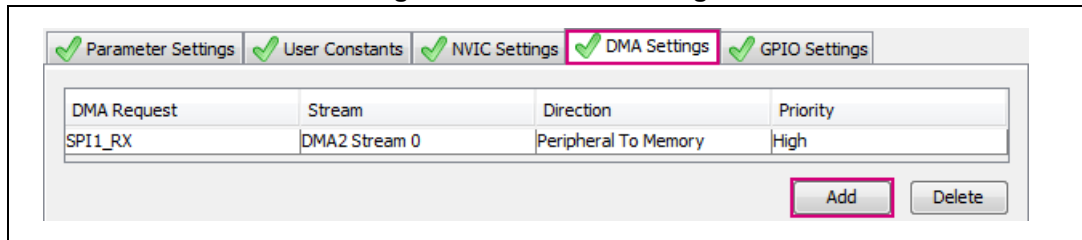


b) SPI DMA settings

In this example the DMA handles the PDM data transfer from the SPI to the Memory.

Open the SPI configuration window, select the DMA Settings Tab and add a DMA request. [Figure 36](#) shows how to enable the DMA.

Figure 36. SPI DMA settings



Click on the DMA request and refer to the DMA request settings section to complete the DMA configuration.

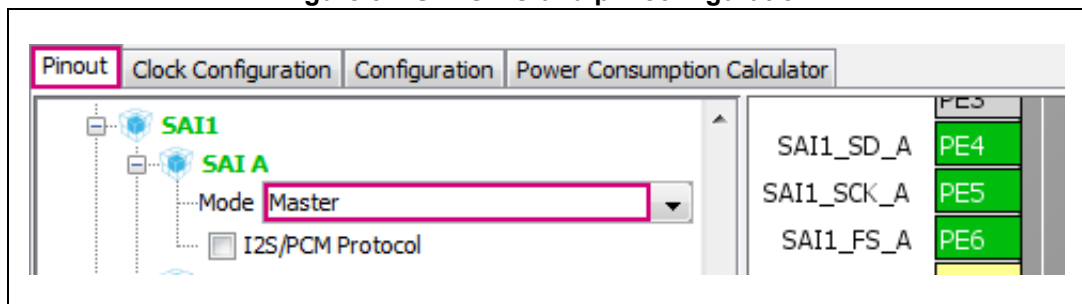
SAI

GPIO and pin configuration

From the listed hardware in the Pinout tab, choose the SAI1 peripheral and enable its sub-block A in Master mode.

[Figure 37](#) shows how to enable the sub-block A of the SAI in Master mode. The enabled pins, SAI1_SD_A, SAI1_SCK_A and SAI1_FS_A, is highlighted in green once the SAI interface GPIOs are correctly configured.

Figure 37. SAI GPIO and pin configuration



Note:

In this example the SAI1FS_A pin cannot be used. This pin can be released and brought back as a GPIO by applying a minor modification in the MSP Initialization file code (stm32f4xx_hal_msp.c), after generating the project.

Clock configuration

In the Clock Configuration tab set the SAI clock.

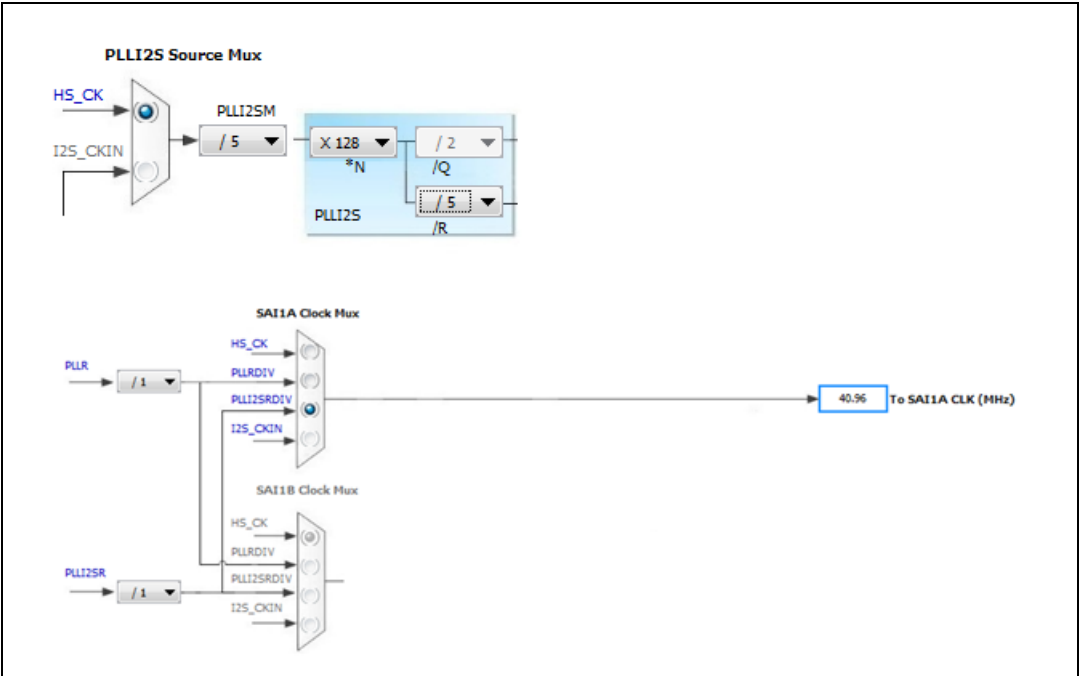
[Table 12](#) provides the audio sampling frequency accuracy values for the clock configuration. The PLLI2SR is used as source clock.

Table 12. Clock configuration and accuracy

Target Audio Frequency (kHz)	Microphone mode	DivM	PLLN	DivR	SAI clock (MHz)	Accuracy (ppm)
16	Mono	5	128	5	40.96	0
	Stereo (with timer)	5	128	5	40.96	0
48	Mono	5	192	25	12.29	0
	Stereo (with timer)	5	192	25	12.29	0

Figure 38 shows an example of SAI clock configuration.

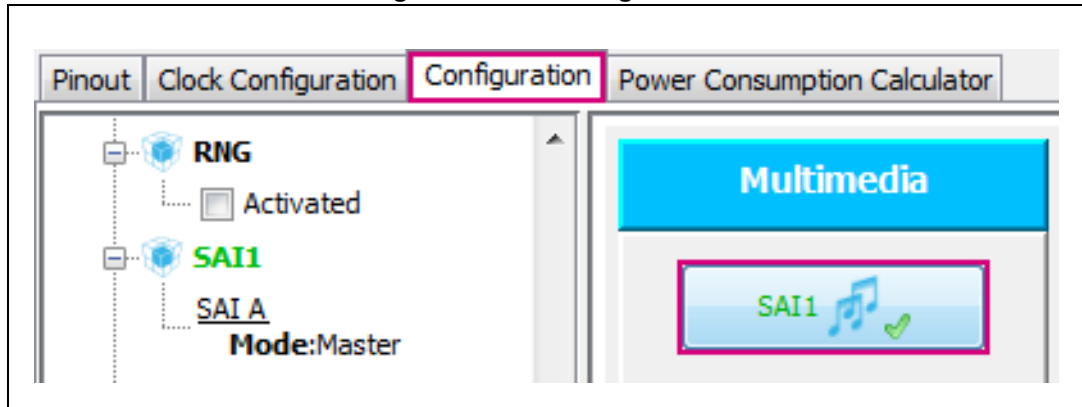
Figure 38. SAI clock configuration at 16 kHz for mono mode



SAI configuration

Select the Configuration tab then click on the SAI1 button as shown in [Figure 39](#).

Figure 39. SAI configuration



a) SAI parameter settings

In the SAI configuration window, select the parameter Settings tab and then configure the parameters. The enabled SAI sub-block is configured according to the following conditions:



- Data Size = 16 Bits
- Output Mode = Stereo (This parameter doesn't change whether one or two microphones are connected to SAI)
- Companding Mode = No companding mode
- Frame Synchro definition = Channel Identification
- Number of Slots = 4
- Slot Active = All
- Frame Synchro Active Length = Frame Length / 2
- Master Clock Divider = Enabled (other possible solution with Master Clock Divider disabled)
- Audio Frequency = Audio Sampling Frequency
- In mono mode the Clock Strobing is chosen according to the state of L/R pin of the microphone.

[Figure 40](#) shows an example of SAI configuration for an Audio Frequency = 16 kHz in Mono mode.

Figure 40. SAI parameter setting

☒ Parameter Settings
 ☒ User Constants
 ☒ NVIC Settings
 ☒ GPIO Settings
 ☒ DMA Settings

Configure the below parameters :

Search :  

☒ SAI A

Basic Parameters	
Protocol	Free
Audio Mode	Master Transmit
Frame Length	64 bits
Data Size	16 Bits
Slot Size	DataSize
Output Mode	Stereo
Companding Mode	No companding mode
SAI SD Line Output Mode	Driven
Frame Parameters	
First Bit	MSB First
Frame Synchro Active Level Length	32
Frame Synchro Definition	Channel Identification
Frame Synchro Polarity	Active Low
Frame Synchro Offset	First Bit
Slot Parameters	
First Bit Offset	0
* Number of Slots (only Even Values)	4
Slot Active Final Value	0x0000FFFF
Slot Active	All
Clock Parameters	
Clock Source	I2S PLL Clock
Master Clock Divider	Enabled
Audio Frequency	16 KHz
Real Audio Frequency	16.0 KHz
Error between Selected	0.0 %
Clock Strobing	Falling Edge
Advanced Parameters	
Fifo Threshold	Empty
Output Drive	Disabled
Synchronization External	Disabled

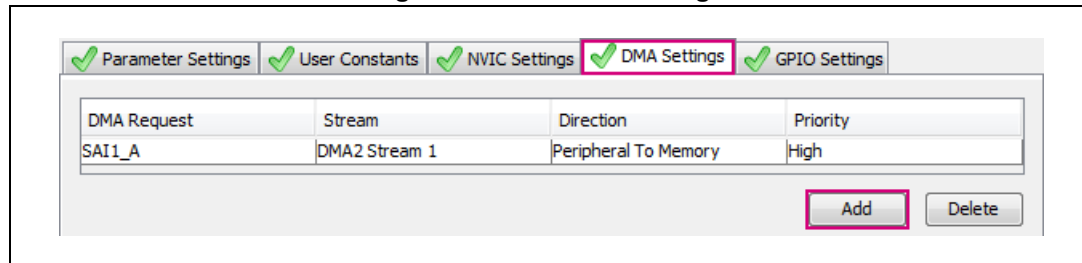
b) SAI DMA settings

In this example the DMA will handle the PDM data transfer from the SAI to the Memory.

In the SAI configuration window, select the DMA Settings Tab and add a DMA request.

[Figure 41](#) shows how to enable the DMA.

Figure 41. SAI DMA settings

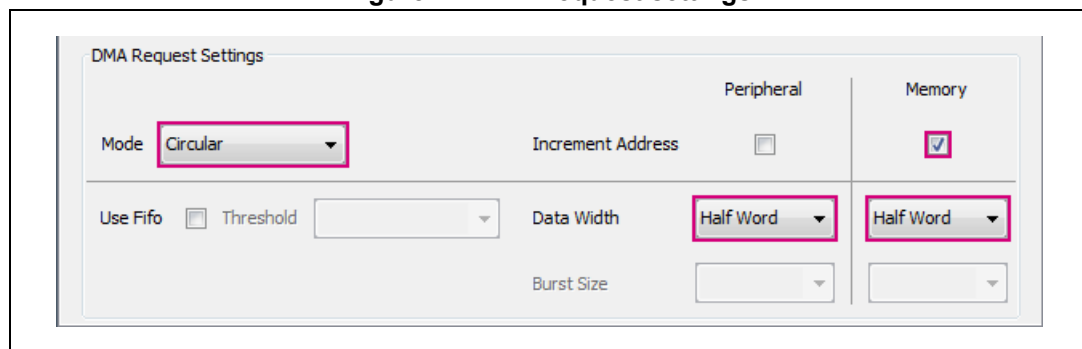


Click on the recently created DMA request and refer to the DMA request settings section to complete the DMA configuration.

DMA request settings

[Figure 39](#) shows the DMA configuration request for each audio interface.

Figure 42. DMA request settings



Timer

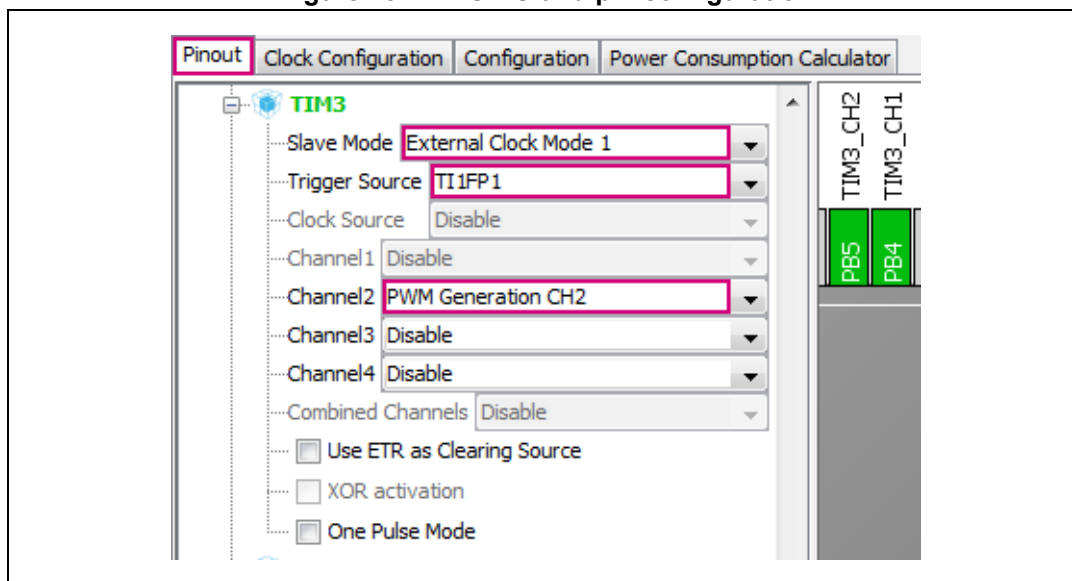
In stereo mode, a timer is used to divide the clock frequency generated by the audio interface and deliver the divided clock to the digital microphones.

GPIO and pin configuration

In the Pinout tab, choose Timer 3 from the listed hardware and enable the slave mode by selecting External Clock Mode 1. Select TI1FP1 as a Trigger Source and enable Channel 2 in PWM Generation CH2.

[Figure 39](#) shows how to enable TIM3. The used pins, TIM3_CH1 and TIM3_CH2, is highlighted in green once the TIM3 GPIOs are correctly configured.

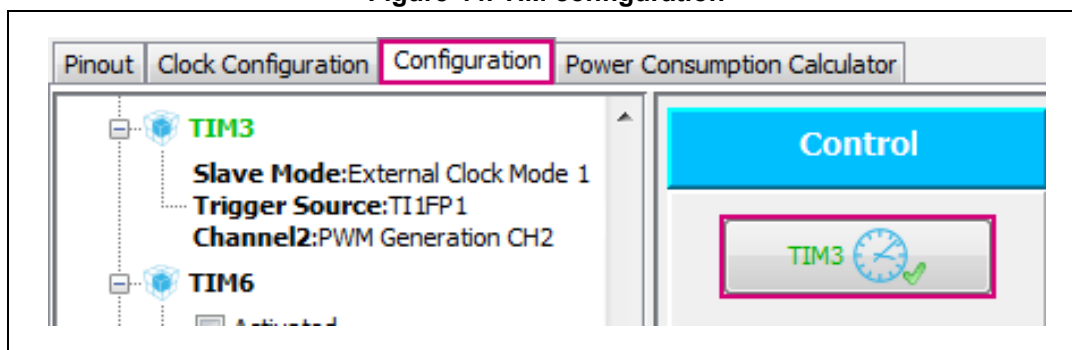
Figure 43. TIM GPIO and pin configuration



Timer configuration

Select the Configuration tab then click on the TIM3 button as shown in [Figure 44](#).

Figure 44. TIM configuration



TIM parameter settings

In the TIM3 configuration window, select the Parameter Settings tab.

Figure 44 shows how to configure TIM3 in order to divide the Trigger Source clock frequency by 2.

Figure 45. TIM parameter settings

The screenshot shows the 'Parameter Settings' tab for a TIM3 configuration. The interface includes a search bar and several expandable sections. The 'Counter Settings' section is expanded, showing parameters like Prescaler (PSC - 16 bits value) set to 0, Counter Mode set to Up, Counter Period (AutoReload Register - 16 bits val...) set to 1, Internal Clock Division (CKD) set to No Division, and Slave Mode Controller set to ETR mode 1. The 'Trigger Output (TRGO) Parameters' section is also expanded, showing Master/Slave Mode set to Disable (no sync between this TIM (Master) and its Slaves) and Trigger Event Selection set to Reset (UG bit from TIMx_EGR). The 'Trigger' section is expanded, showing Trigger Polarity set to Rising Edge and Trigger Filter (4 bits value) set to 0. The 'PWM Generation Channel 2' section is expanded, showing Mode set to PWM mode 1, Pulse (16 bits value) set to 1, Fast Mode set to Disable, and CH Polarity set to High.

Section	Parameter	Value
Counter Settings	Prescaler (PSC - 16 bits value)	0
	Counter Mode	Up
	Counter Period (AutoReload Register - 16 bits val...)	1
	Internal Clock Division (CKD)	No Division
	Slave Mode Controller	ETR mode 1
Trigger Output (TRGO) Parameters	Master/Slave Mode	Disable (no sync between this TIM (Master) and its Slaves)
	Trigger Event Selection	Reset (UG bit from TIMx_EGR)
Trigger	Trigger Polarity	Rising Edge
	Trigger Filter (4 bits value)	0
PWM Generation Channel 2	Mode	PWM mode 1
	Pulse (16 bits value)	1
	Fast Mode	Disable
	CH Polarity	High

4.1.2 Adding PDM software decoding library middleware files

Choose the favorite toolchain, generate the project with CubeMX and open the generated project.

For the STM32F413xx MCU, the PDM audio software decoding library includes one header file `pdm_filter.h` and binary/object codes for the following platforms:

- `libPDMFilter_CM4F_IAR.a`: for IAR compiler
- `libPDMFilter_CM4F_Keil.lib`: for ARM compiler
- `libPDMFilter_CM4F_GCC.a`: for GNU compiler

This Library is provided in the "STM32Cube_FW_F4" firmware package (V1.16 and later) under the following path "Middlewares\ST\STM32_Audio\Addons\PDM"

Make sure to add header file `pdm_filter.h` path to the project Include Paths and the appropriate binary/object file to the project source files.

4.2 Example 2: Interfacing digital microphones in stereo mode with SAI using two synchronous sub-blocks

This example is based on the NUCLEO-F413ZH board and using two external digital microphones that will be connected each one to an SAI sub-block. It is applicable to the hardware connection described in the [Section 2.2.2: Using two synchronous SAI sub-blocks](#)

4.2.1 SAI configuration using STM32CubeMX

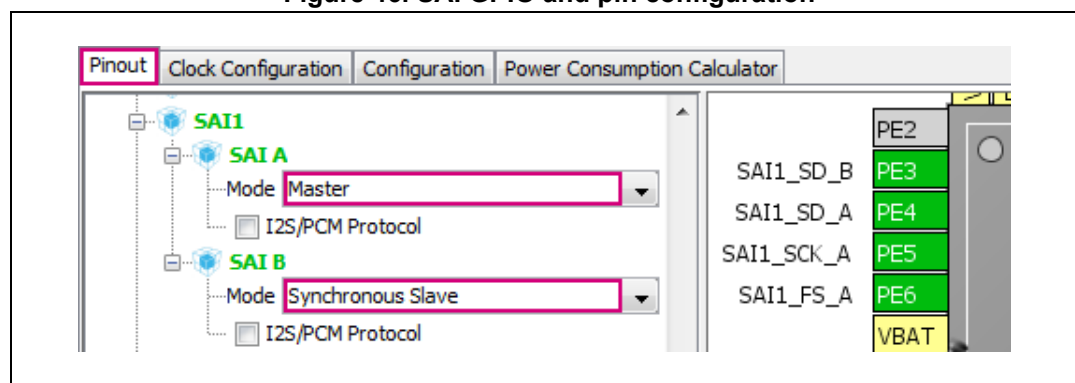
GPIO and Pin configuration

After creating the STM32CubeMX project, choose the SAI1 peripherals from the listed hardware in the Pinout tab, configure its sub-block A in Master mode and the sub-block B in Synchronous Slave mode.

[Figure 46](#) shows how to enable the sub-block A of SAI1 in Master mode and the sub-block B in Synchronous Slave mode. In this case, the bit clock and the frame synchronization signals are shared between the two sub-blocks to reduce the number of external pins used for the communication.

The enabled pins, SAI1_SD_A, SAI1_SD_B, SAI1_SCK_A and SAI1_FS_A, will be highlighted in green once the SAI peripheral GPIOs are correctly configured.

Figure 46. SAI GPIO and pin configuration



Clock configuration using STM32CubeMx

This section describes the clock configuration for SAI using two synchronous sub-blocks for 16 KHz and 48 KHz streams.

For 16KHz sampling rate the bitclock generated by SAI should be 1.024MHz .

For 48KHz sampling rate the bitclock generated by SAI should be 3.072MHz .

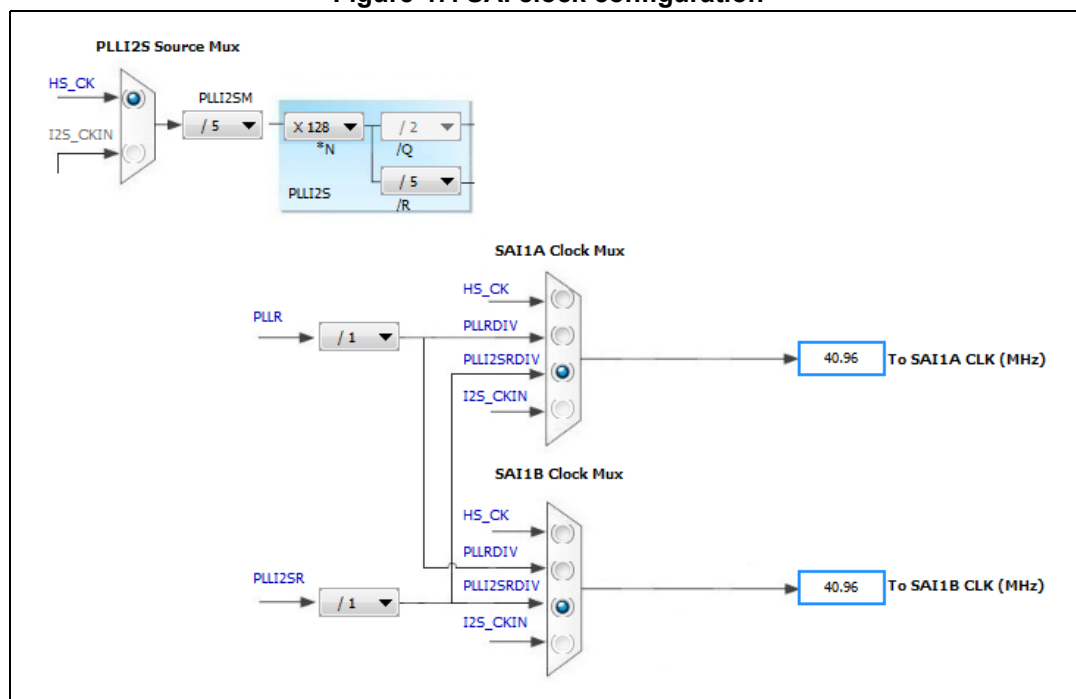
The PLLI2SR is used as source clock. To get the accurate bitclock frequency, the user must configure the adequate MCKIDV.

Table 13. SAI clock configuration and accuracy

Target Audio Freq (kHz)	DivM	PLLN	DivR	SAI clock (MHz)	Accuracy (ppm)
16	5	128	5	40.96	0
48	5	192	25	12.29	0

Figure 47 provides the SAI clock configuration used in this example.

Figure 47. SAI clock configuration



Parameter settings

In the SAI configuration window, select the parameter Settings tab then configure the parameters. The SAI sub-blocks should be configured according to the following conditions:

- Data Size = 16 Bits
- Frame Length = 4 x Data Size
- Output Mode = Stereo
- Companding Mode = No companding mode
- Frame Synchro definition = Channel Identification
- Number of Slots = 4
- Slot Active = All
- Frame Synchro Active Length = Frame Length / 2
- Master Clock Divider = Enabled
- Audio Frequency = AUDIO_SAMPLING_FREQUENCY



The Clock Strobing is chosen according to the state of L/R pin of the microphone connected to every sub-block.

Figure 48 and *Figure 49* show respectively how to configure SAIA and SAIB for an audio sampling frequency = 16 kHz.

Figure 48. SAI parameter settings

☒ Parameter Settings
 ☒ User Constants
 ☒ NVIC Settings
 ☒ GPIO Settings
 ☒ DMA Settings

Configure the below parameters :

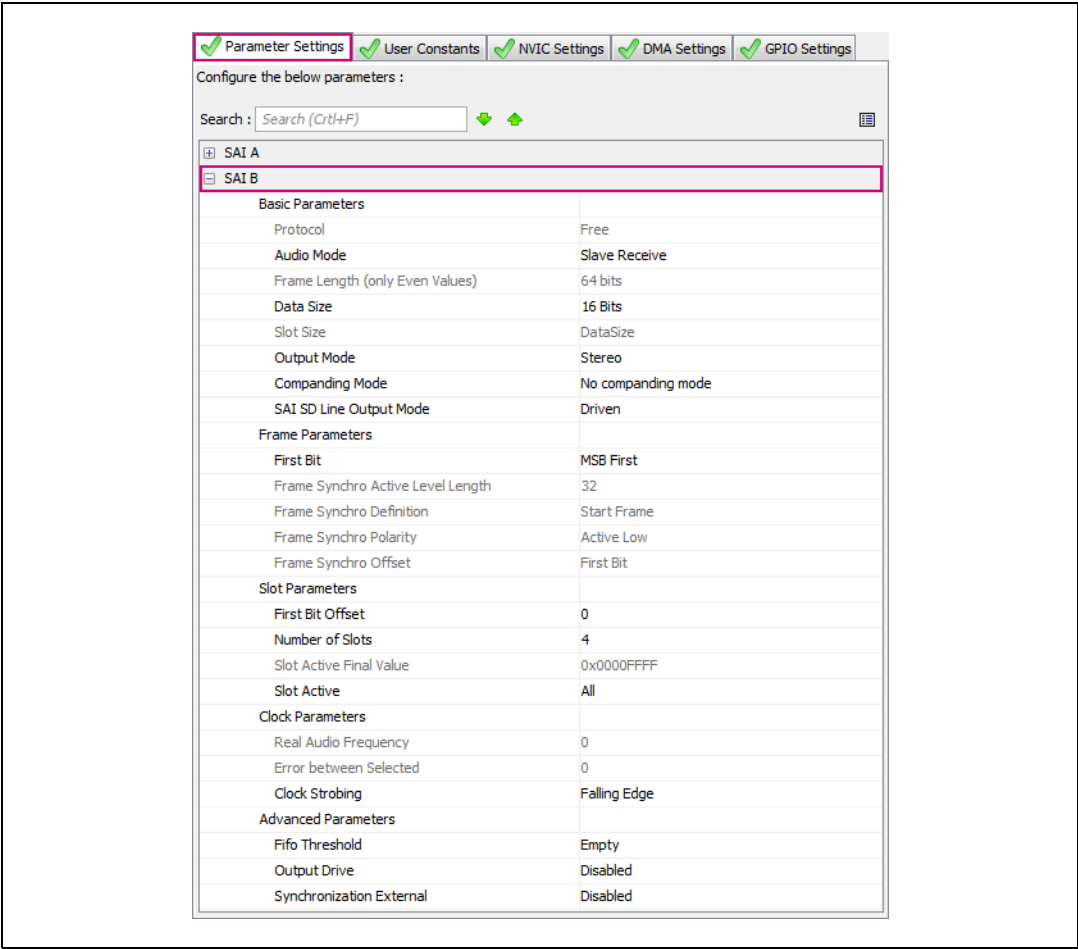
Search :  

☒ SAI A

Basic Parameters	
Protocol	Free
Audio Mode	Master Receive
Frame Length	64 bits
Data Size	16 Bits
Slot Size	DataSize
Output Mode	Stereo
Companding Mode	No companding mode
Frame Parameters	
First Bit	MSB First
Frame Synchro Active Level Length	32
Frame Synchro Definition	Channel Identification
Frame Synchro Polarity	Active Low
Frame Synchro Offset	First Bit
Slot Parameters	
First Bit Offset	0
Number of Slots (only Even Values)	4
Slot Active Final Value	0x0000FFFF
Slot Active	All
Clock Parameters	
Clock Source	I2S PLL Clock
Master Clock Divider	Enabled
Audio Frequency	16 KHz
Real Audio Frequency	16.0 KHz
Error between Selected	0.0 %
Clock Strobing	Rising Edge
Advanced Parameters	
Fifo Threshold	Empty
Output Drive	Disabled
Synchronization External	Disabled

☒ SAI B

Figure 49. SAIB parameter settings

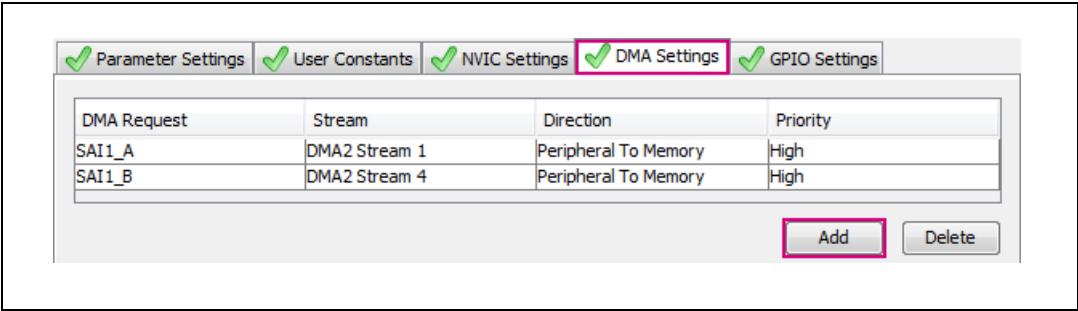


SAI DMA settings

In this example the DMA handles the PDM data transfer from each SAI sub-block to the Memory.

In the SAI configuration window, select the DMA Settings Tab and add a DMA request for each sub-block. [Figure 50](#) shows how to enable the DMA.

Figure 50. SAI DMA settings



Click on the recently created DMA requests and refer to the DMA request settings section to complete the DMA configuration.

4.2.2 Adding PDM software decoding library middleware files

Refer to [Section 4.1.2: Adding PDM software decoding library middleware files](#).

4.3 Example 3: Interfacing digital microphones in stereo mode with SAI using PDM interface

This example is based on the NUCLEO-H743ZI board and using two external digital microphones that will be connected in Stereo mode to SAI through the PDM interface.

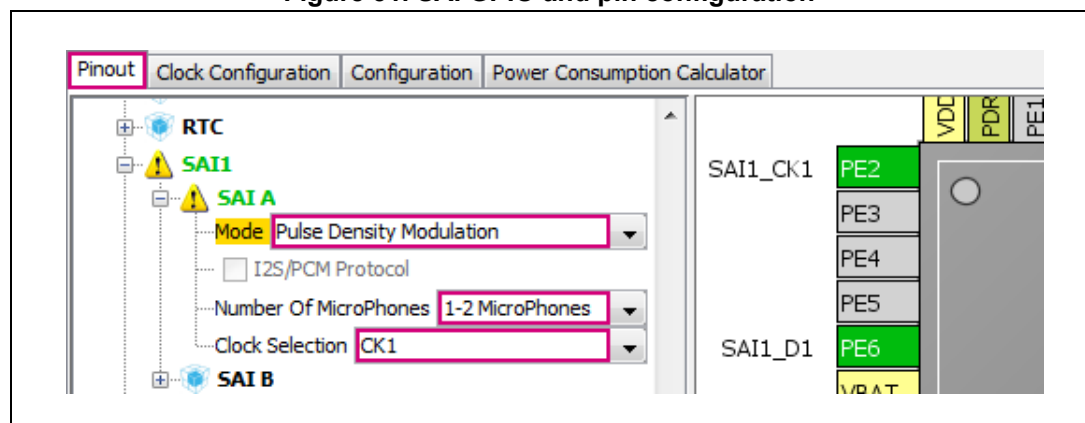
4.3.1 SAI configuration using STM32CubeMX

GPIO and Pin configuration

After creating the STM32CubeMX project, choose the SAI1 peripherals from the listed hardware in the Pinout tab and configure it in Pulse Density Modulation mode. Select the number of microphones as well as the output clock.

[Figure 51](#) shows how to enable the sub-block A of SAI1 in Pulse Density Modulation mode, supporting of two microphones and having CK1 as an output clock. The enabled pins, SAI1_CK1 and SAI1_D1, will be highlighted in green once the SAI peripheral GPIOs are correctly configured.

Figure 51. SAI GPIO and pin configuration

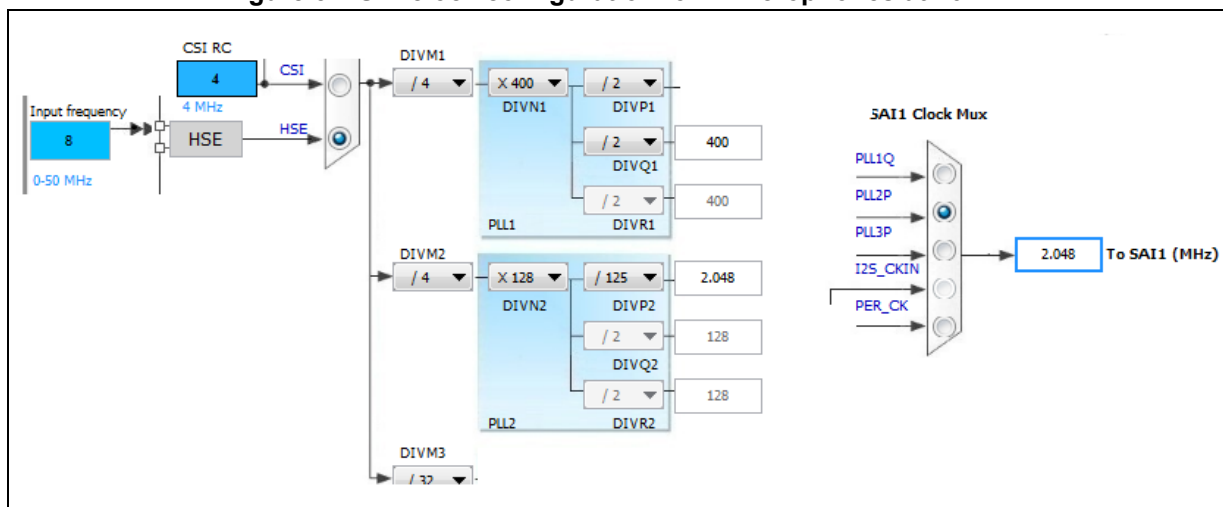


Clock configuration

[Figure 52](#) provides the SAI clock configuration used in this example.

For more details about clock configuration, refer to “Allowed TDM frame configuration” table in reference manual RM0433.

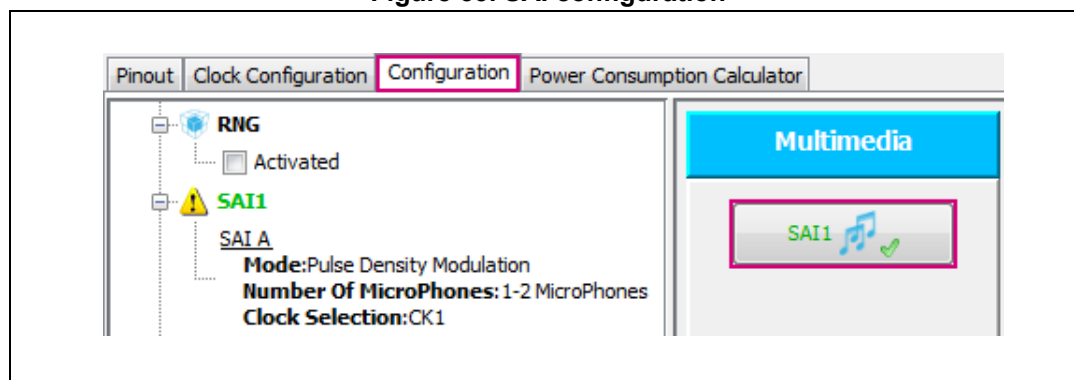
Figure 52. SAI clock configuration for 2 microphones at 16 kHz



SAI configuration

Select the configuration tab then click on the SAI1 button as shown in [Figure 53](#).

Figure 53. SAI configuration





Parameter settings

In the SAI configuration window, select the parameter settings tab and then configure the parameters. The SAI sub-block A should be configured according to the following conditions:

- Data Size = 16 Bits
- Frame Length = 16
- Output Mode = Stereo
- Companding Mode = No companding mode
- Frame Synchro definition = Start Frame
- Number of Slots = 1
- Slot Active = All
- Frame Synchro Active Length = 1
- Frame Synchro Polarity = Active High
- Master Clock Divider = Disabled

Figure 54 shows how to configure SAI for an audio sampling frequency= 16 kHz.

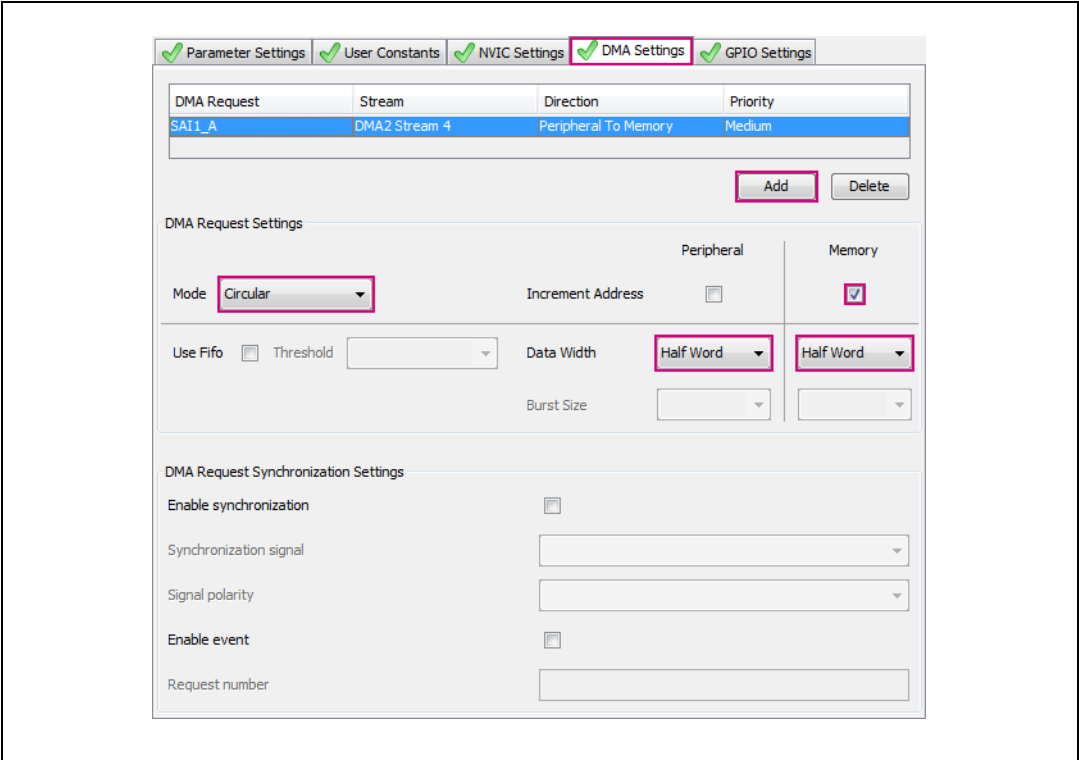
Figure 54. SAI parameter settings

<input checked="" type="checkbox"/> Parameter Settings <input checked="" type="checkbox"/> User Constants <input checked="" type="checkbox"/> NVIC Settings <input checked="" type="checkbox"/> GPIO Settings <input checked="" type="checkbox"/> DMA Settings	
Configure the below parameters :	
Search : <input type="text" value="Search (Ctrl+F)"/>  	
Basic Parameters	
Protocol	Free
Audio Mode	Master Receive
Data Size	16 Bits
Output Mode	Stereo
Companding Mode	No companding mode
First Bit	LSB First
Data Frame Parameters	
Frame Length (only Even Values)	16 bits
Frame Synchro Active Level Length	1
Frame Synchro Definition	Start Frame
Frame Synchro Polarity	Active High
Frame Synchro Offset	First Bit
Block Slot Parameters	
Slot Size	DataSize
First Bit Offset	0
Number of Slots	1
Slot Active Final Value	0x0000FFFF
Slot Active	All
Clock Parameters	
Clock Source	SAI PLL Clock
Master Clock Divider	Disabled
Real Audio Frequency	128.0 KHz
Clock Strobing	Falling Edge
Advanced Parameters	
Fifo Threshold	Empty
Output Drive	Disabled
Synchronization External	Disabled

DMA Settings

In this example the DMA will handle the PDM data transfer from the SAI to the Memory.
In the SAI configuration window, select the DMA Settings Tab and add a DMA request.
[Figure 55](#) shows how to configure the DMA.

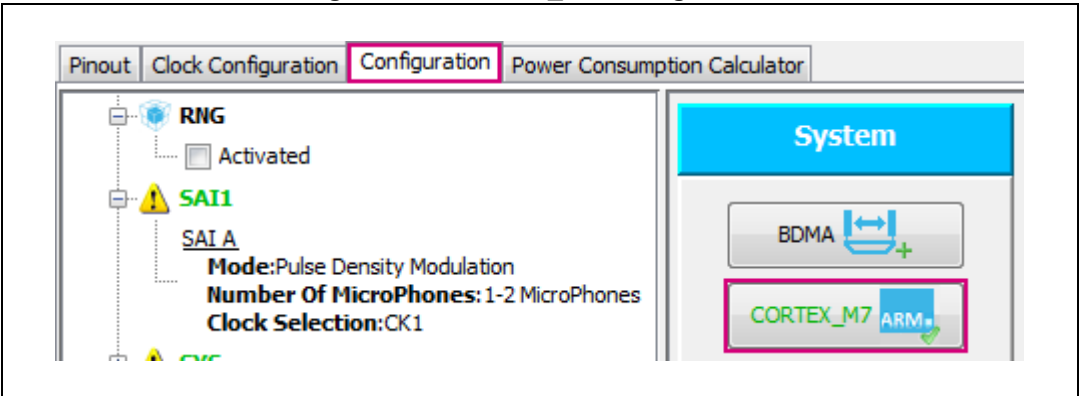
Figure 55. SAI DMA settings



Cortex_M7 configuration

Select the configuration tab then click on the CORTEX_M7 button as shown in [Figure 56](#).

Figure 56. CORTEX_M7 configuration



Parameter settings

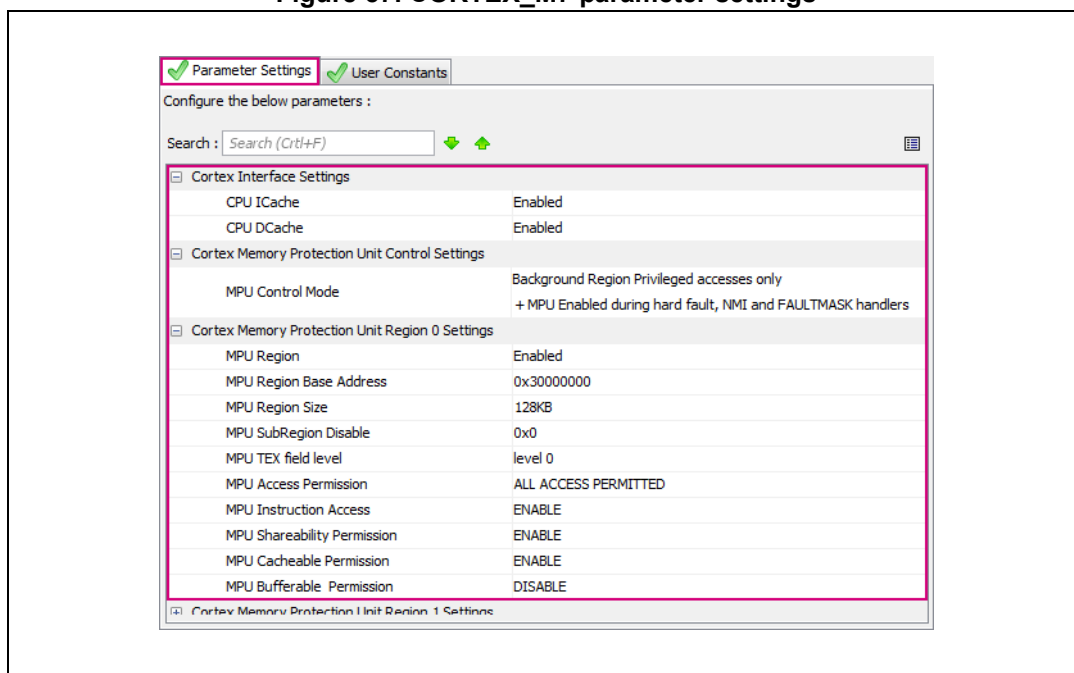
In the CORTEX_M7 configuration window, select the parameter settings tab to configure the Cache and the MPU. This allows to enhance the performance in case of use of AXI interface with several masters.

Refer to the Template project in the STM32Cube_FW_H7_V1.0.0 firmware package (V1.0.0 and later) under the following path “Projects\STM32H743ZI-Nucleo\Templates” for a typical MPU configuration.

For more details about the MPU configuration and use, please refer to AN4838 “Managing memory protection unit (MPU) in STM32 MCUs”.

Figure 57 shows how to configure the Cache and the MPU for this example.

Figure 57. CORTEX_M7 parameter settings



4.3.2 Adding PDM software decoding library middleware files

Choose your favorite toolchain, generate the project with CubeMX and open the generated project.

For the STM32H7 series, the PDM audio software decoding library includes one header file “pdm2pcm_glo.h” and binary/object codes for the following platforms:

- libPDMFilter_CM7_IAR.a: for IAR compiler
- libPDMFilter_CM7_Keil.lib: for ARM compiler
- libPDMFilter_CM7_GCC.a: for GNU compiler

This Library is provided in the “STM32Cube_FW_H7” firmware package (V1.0.0 and later) under the following path “Middlewares\ST\STM32_Audio\Addons\PDM”

Make sure to add header file pdm2pcm_glo.h path to the project Include Paths and the appropriate binary/object file to the project source files.

4.4 Example 4: Interfacing digital microphones using DFSDM

This example is based on the NUCLEO-F413ZH board and using external digital microphones that will be connected to DFSDM in stereo mode. It is applicable to the hardware connection described in the [Section 2.3: Digital filter for sigma delta modulators \(DFSDM\)](#).

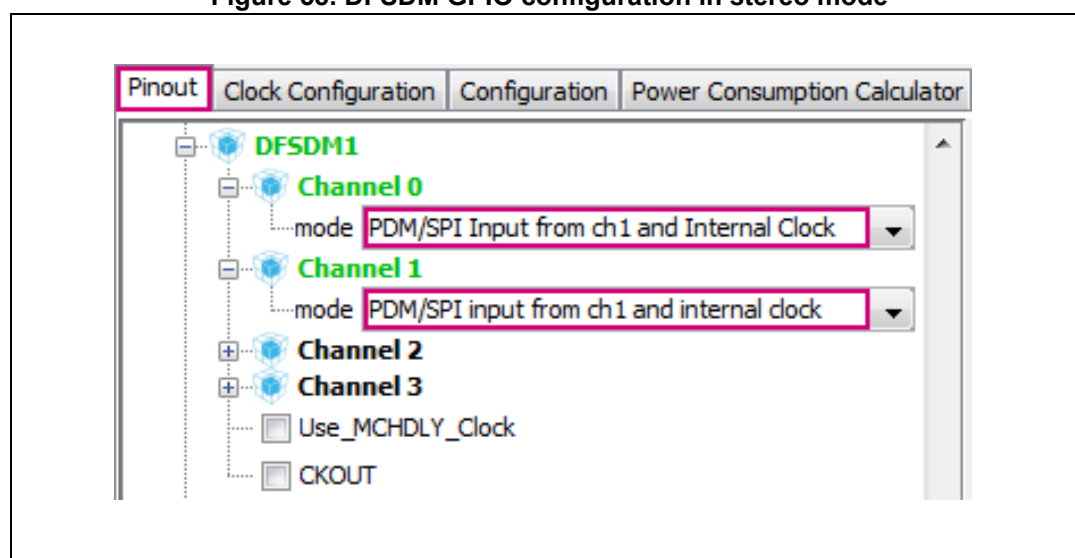
4.4.1 DFSDM configuration using STM32CubeMX

GPIO and Pin configuration

After creating the STM32CubeMX project, choose the DFSDM1 peripheral from the listed hardware in the Pinout tab and enable the needed DFSDM channels as shown in figures below.

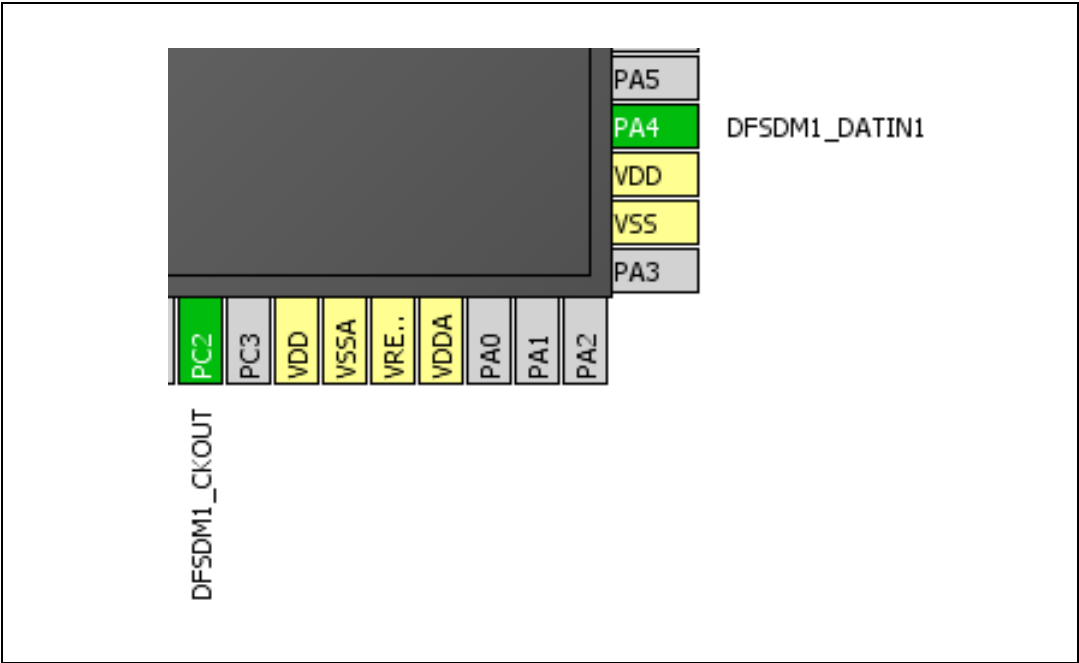
[Figure 58](#) shows how to enable DFSDM channels in stereo mode. In this mode the enabled channels must be successive.

Figure 58. DFSDM GPIO configuration in stereo mode



The used pins, DFSDM1_DATIN1 and DFSDM1_CKOUT, will be highlighted in green once the DFSDM peripheral GPIOs are correctly configured as shown in [Figure 59](#).

Figure 59. DFSDM pin configuration



Channel configuration

In the DFSDM configuration window, select the parameter settings tab then configure the parameters.

The value of the field Type is chosen according to the state of L/R pin of the microphones connected DFSDM channels. In stereo mode each channel must read data on a different clock edge to allow the microphones to share one data line.

Figure 60 and *Figure 61* show how to configure respectively Channel 1 and Channel 0 for an Audio Frequency = 16 kHz.

Figure 60. DFSDM Channel 1 configuration

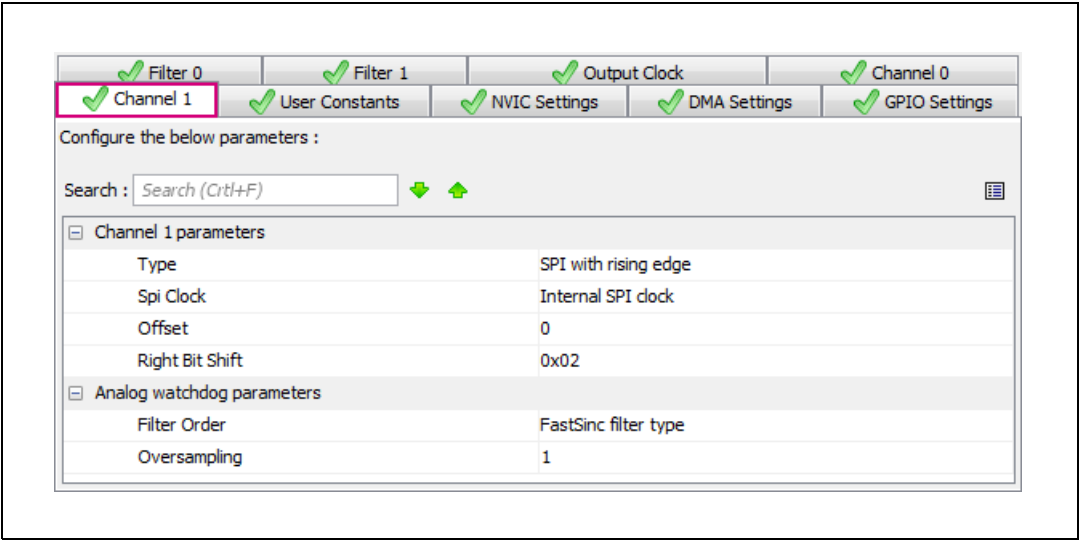
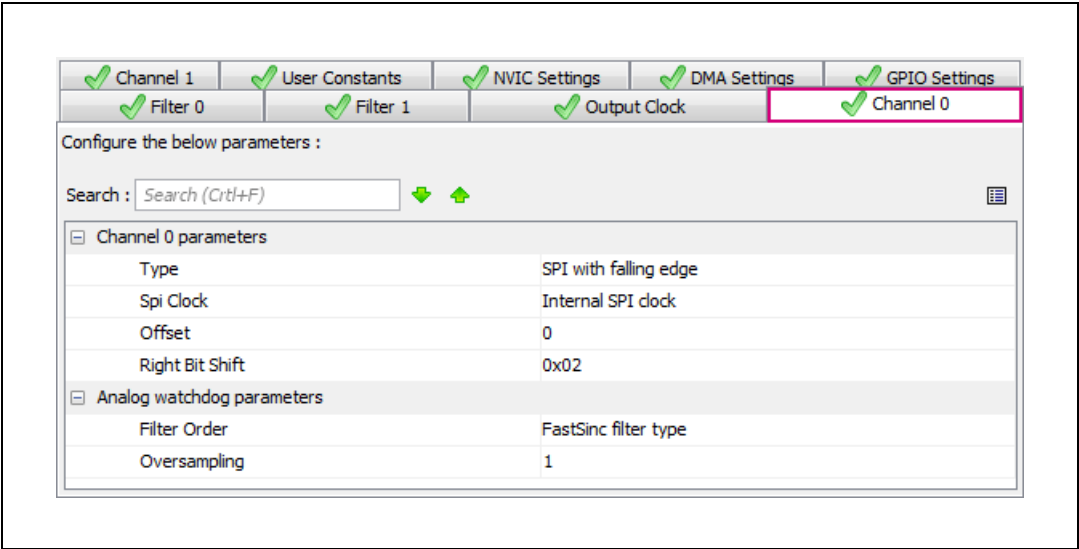


Figure 61. DFSDM Channel 0 configuration



Filter configuration

In this example, Filter 0 and Filter 1 are linked respectively to Channel 1 and Channel 0 as regular channels.

The Sinc Order is configured according to the audio sampling frequency. Table 11 shows its possible values.

Table 14. DFSDFM filter order values

Audio Sampling Frequency (kHz)	Filter order
16	Sinc 4 or Sinc 5 filter type
48	Sinc 4 or Sinc 5 filter type

The field Fosr contains the value of the decimation factor.

[Figure 62](#) and [Figure 63](#) show a configuration example of Filter 1 and Filter 0 for an Audio Sampling Frequency = 16 kHz.

Figure 62. DFSDM Filter 0 configuration

Configure the below parameters :

Search :

regular channel selection	
regular channel selection	Channel 1
Continuous Mode	Continuous Mode
Trigger to start regular conversion	Software trigger
Fast Mode	Enable
Dma Mode	Enable
injected channel selection	
Channel0 as injected channel	Disable
Channel1 as injected channel	Disable
Channel2 as injected channel	Disable
Channel3 as injected channel	Disable
Filter parameters	
Sinc Order	Sinc 4 filter type
Fosr	64
Iosr	1

Figure 63. DFSDM Filter 1 configuration

Configure the below parameters :

Search :

regular channel selection	
regular channel selection	Channel 0
Continuous Mode	Continuous Mode
Trigger to start regular conversion	Software trigger
Fast Mode	Enable
Dma Mode	Enable
injected channel selection	
Channel0 as injected channel	Disable
Channel1 as injected channel	Disable
Channel2 as injected channel	Disable
Channel3 as injected channel	Disable
Filter parameters	
Sinc Order	Sinc 4 filter type
* Fosr	64
Iosr	1

Output Clock configuration

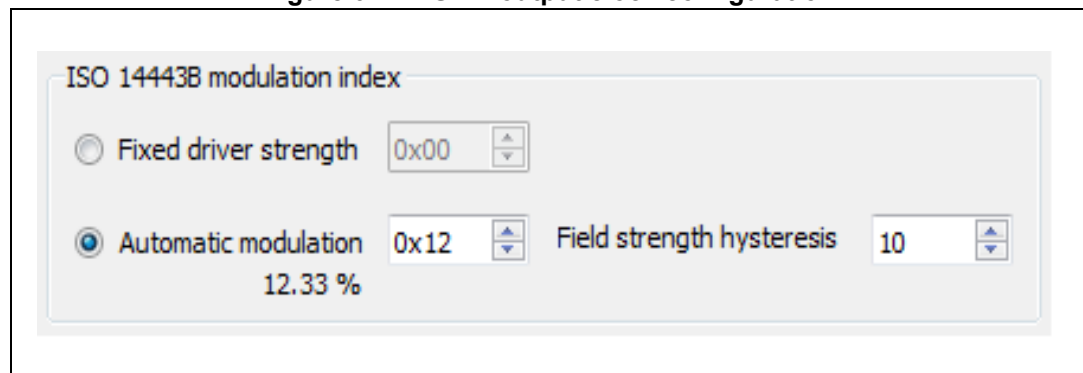
In this example, Audio Clock is selected as a source for DFSDM output clock which will give the user more flexibility when configuring the clock.

The clock divider value must respect the following formula:

Divider = DFSDM Clock Source / (AUDIO_SAMPLING_FREQUENCY × DECIAMTION_FACTOR)

[Figure 64](#) shows a configuration example of a DFSDM Clock Source = 48 MHz, an Audio Sampling Frequency = 16 kHz and a Decimation Factor = 64.

Figure 64. DFSDM output clock configuration



DFSDM DMA Settings

In this example the data transfer from DFSDM channels to the memory will be handled by the DMA thus reducing the software overhead.

In the DFSDM configuration window, select the DMA Settings Tab and add a DMA request for each Filter.

Click on the recently created DMA requests and configure the settings as shown in [Figure 65](#).

Figure 65. DFSDM DMA setting

DMA Request	Stream	Direction	Priority
DFSDM1_FLT0	DMA2 Stream 0	Peripheral To Memory	High
DFSDM1_FLT1	DMA2 Stream 1	Peripheral To Memory	High

Mode: **Circular** Increment Address: ☐ Memory: ☒

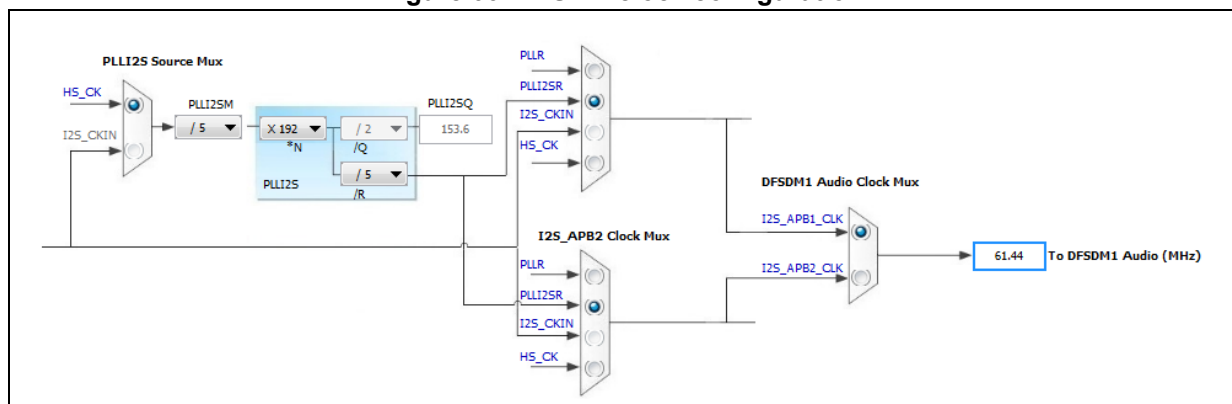
Use Fifo: ☐ Threshold: Data Width: **Word** **Word**

Burst Size:

DFSDM clock configuration

In this example, the DFSDM output clock is derived from audio clock delivering a 61.44 MHz clock to DFSDM as shown in [Figure 66](#).

Figure 66. DFSDM clock configuration



[Table 15](#) provides audio sampling frequency accuracy values for this clock configurations.

- For 16 kHz stream, the CKOUT frequency must be 1.024 MHz.
- For 48 kHz stream, the CKOUT frequency must be 3.072 MHz.
- The oversampling ratio is 64.

Table 15. DFSDM clock configuration accuracy values

Target Audio Frequency (kHz)	DivM	DivN	DivR	Output Clock Divider	DFSDM clock (MHz)	Accuracy (ppm)
16	5	192	5	60	61.44	0
48	5	192	5	20	61.44	0

5 Conclusion

The STM32 microcontrollers provide a variety of audio and serial interfaces to connect digital microphones giving the flexibility to choose the most suited solution for each application.

This application note demonstrates the STM32 microcontroller audio capabilities to interface digital MEMS microphones by describing the different ways to connect these microphones to the STM32 peripherals and providing guidelines to use and properly configure these peripherals to acquire and handle the raw data from the microphones.

6 **Revision history**

Table 16. Document revision history

Date	Revision	Changes
14-Dec-2017	1	Initial release.



IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2017 STMicroelectronics – All rights reserved