Information Technology Institute (ITI)

Intake 39

Graduation project

Title:-

- An adaptable digital hearing aid.

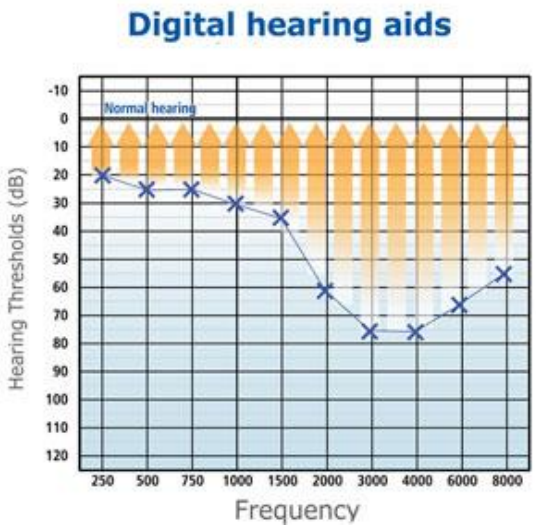
Team members:-

1) Hanna Nabil Abd El-shahid.
2) Bishoy Medhat Elraheb.

Under the supervision of:-

1) Eng.\ Ahmed Al-Ashmawy.
2) Eng.\ Yousef Nofel.

Description:-



- An adaptable digital hearing aid that allows the dynamic change of its amplification by the user. Moreover, it will be less costly than an analogue prefixed hearing aid that requires constant changing to adapt with the patient's changing state.
- The most basic function of a hearing aid is to amplify sound. Digital hearing aids do this in a rather sophisticated way.
- As sound enters the device, it is broken into multiple frequency bands. Each band is then amplified by the amount necessary to return the wearer's hearing to normal levels at that band.
- With digital technology, devices can now break sound into as many as 24 different bands.
- Given that every person has a unique pattern of hearing loss, the sound quality provided by a modern hearing aid is far better the previous analogue technologies that were restricted to two bands - base (low frequencies) and treble (high frequencies).

Problem Definition:-

- Hearing impairment is one of the commonest birth defects.
- It is the third leading chronic disability affecting nearly 250 million people in the world, and 75% of sufferers live in developing countries.
- The impact of hearing impairment on the individual and society is significant.
- Development of hearing loss leads to severe handicap that affects the sufferer's job, home and life with subsequent social and economic burden on the society.
- In children the problem is compounded since normal hearing is the primary source for acquisition of language, speech and cognitive skills.
- In a national household survey conducted to estimate the prevalence of hearing impairment in Egypt, it was found to be high in those aged 0-4 years (22.4%).

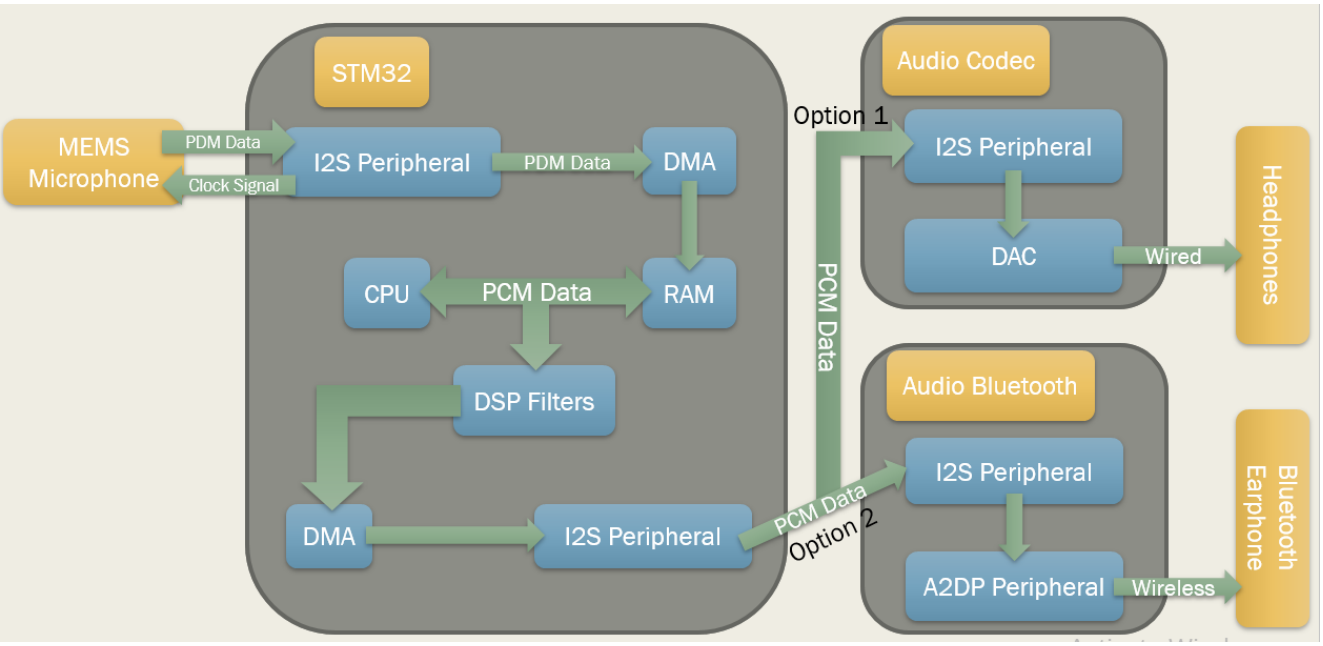
Our objective is a product with 2 options:-

- A complete standalone headset that works as a Digital Hearing Aid and controllable by a mobile app via Bluetooth **with:-**
- **1st option:** the output sound is send to an audio Bluetooth module then wirelessly to earphones.
- **2nd option:** the output sound is send to an audio codec then send by wire to earphones.

Our product advantage:-

- **Appearance**: Our product appears like mobile's earphones in shape so it will not be shame to anyone who have a hearing impairment problem to use our product.
- As we know many of friends have already this problem and they feel shy to wear a hearing aid because it has a unique shape.
- **Cost**: The range of hearing aids products is from 500£ to 2000£ but our product will be less than this price.
- **Controllable by mobile app:** you can control our product using mobile app by varying the gain of specific frequency bands using specific buttons on the mobile app.

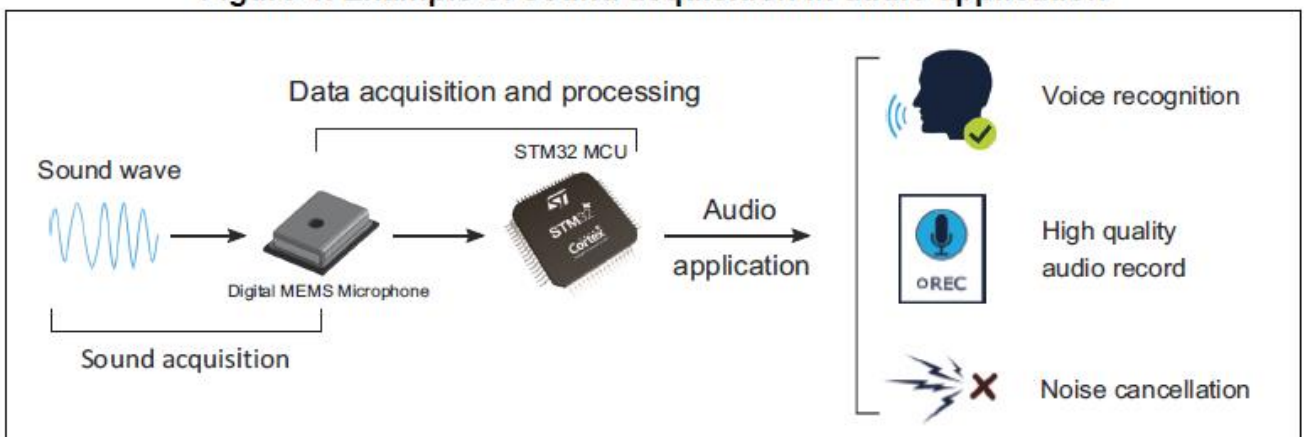
Block diagram:-



Mems microphones:-

- 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.

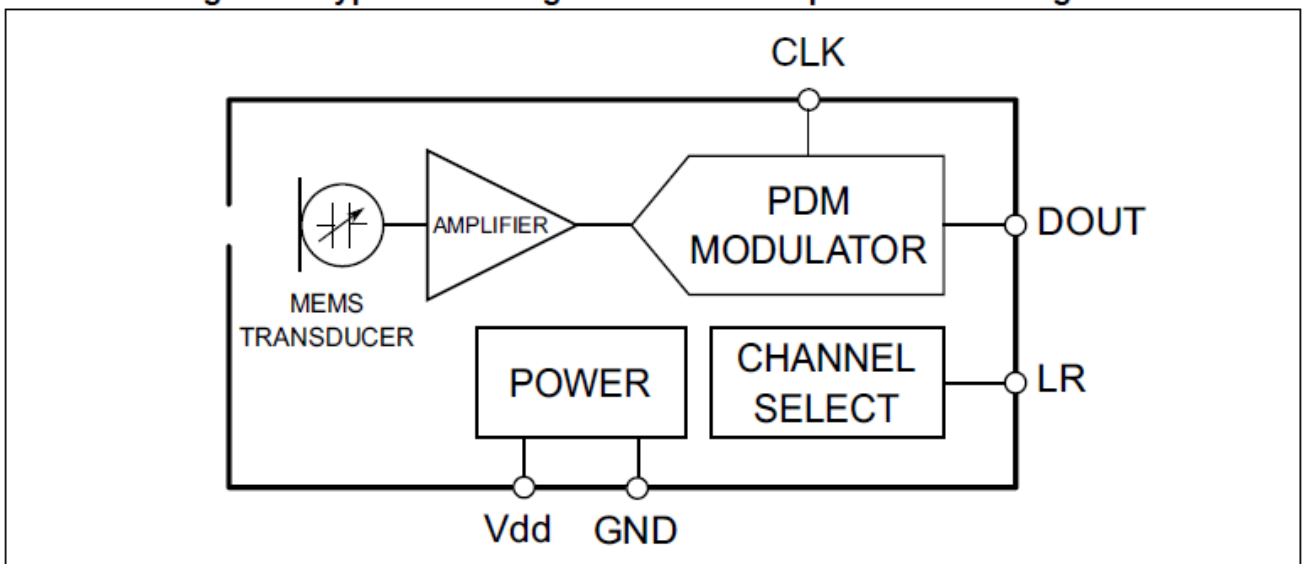
Figure 1. Example of sound acquisition in audio application



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 TRASDUCER, 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 bit stream).

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

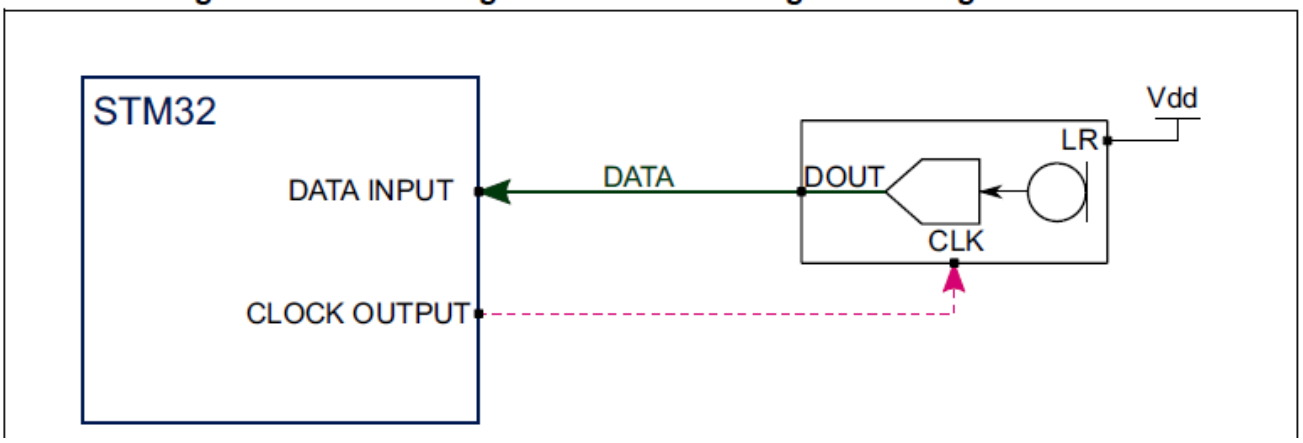
Pin name	Function	Direction
Vdd	3.3 V Power supply	Input
GND	0 V	Input
LR	Left/right selection	Input
CLK	Synchronization clock	Input
DOUT	Left/righ PDM data output	Output

Basic digital microphones connection:-

Mono mode:-

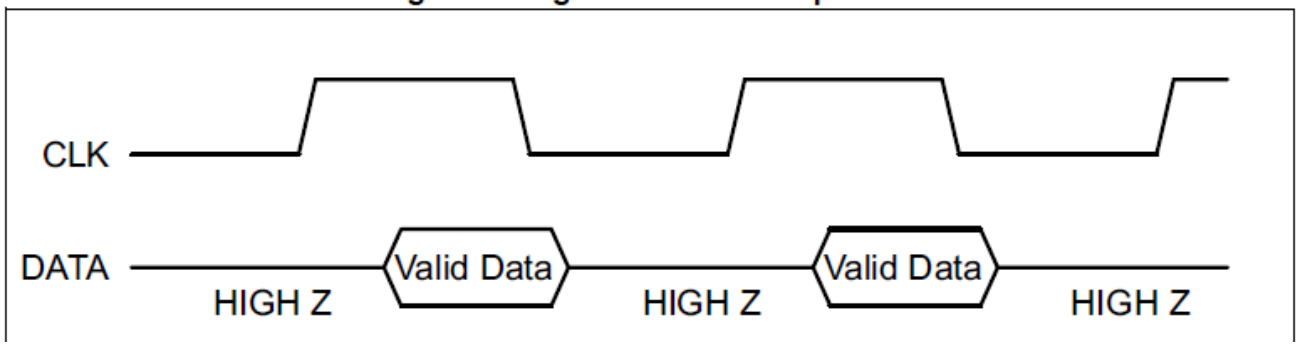
- In this mode LR pin can be either connected to Vdd or to GND.
- LR pin is connected to Vdd.

Figure 3. Mono configuration - Generating data on right channel



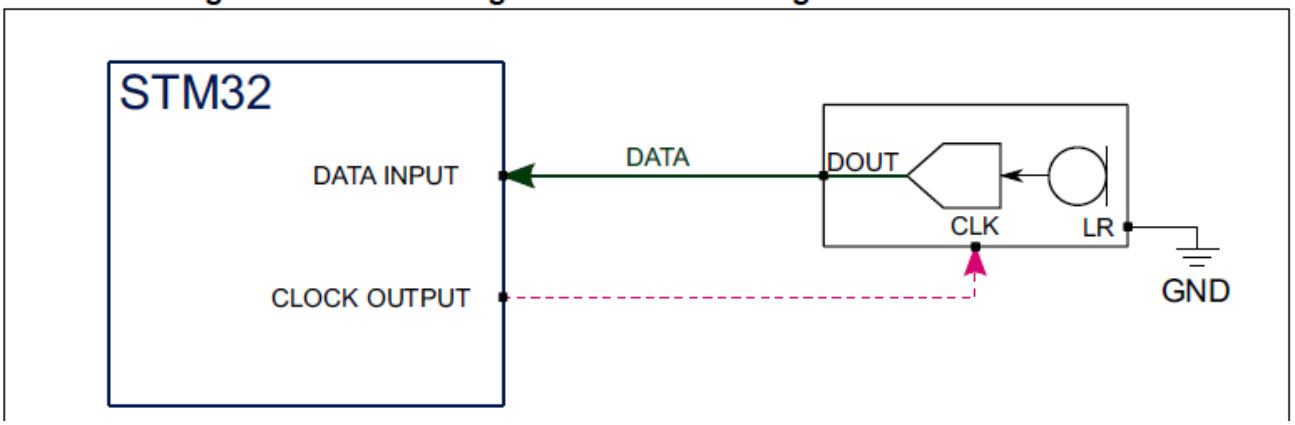
- On the rising edge of the clock, the microphone will generate valid data for half of the clock period, then goes into a high impedance state for the other half.

Figure 4. Right channel data pattern



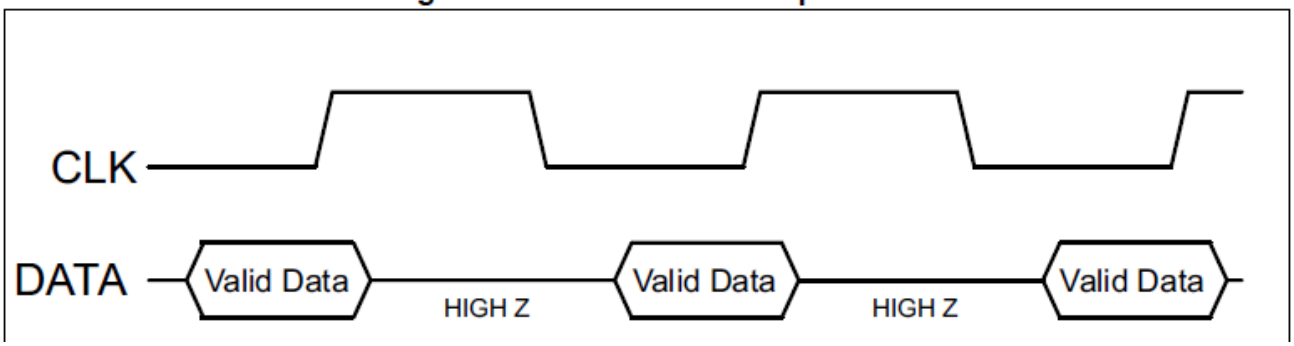
- LR pin is connected to GND.

Figure 5. Mono configuration - Generating data on left channel



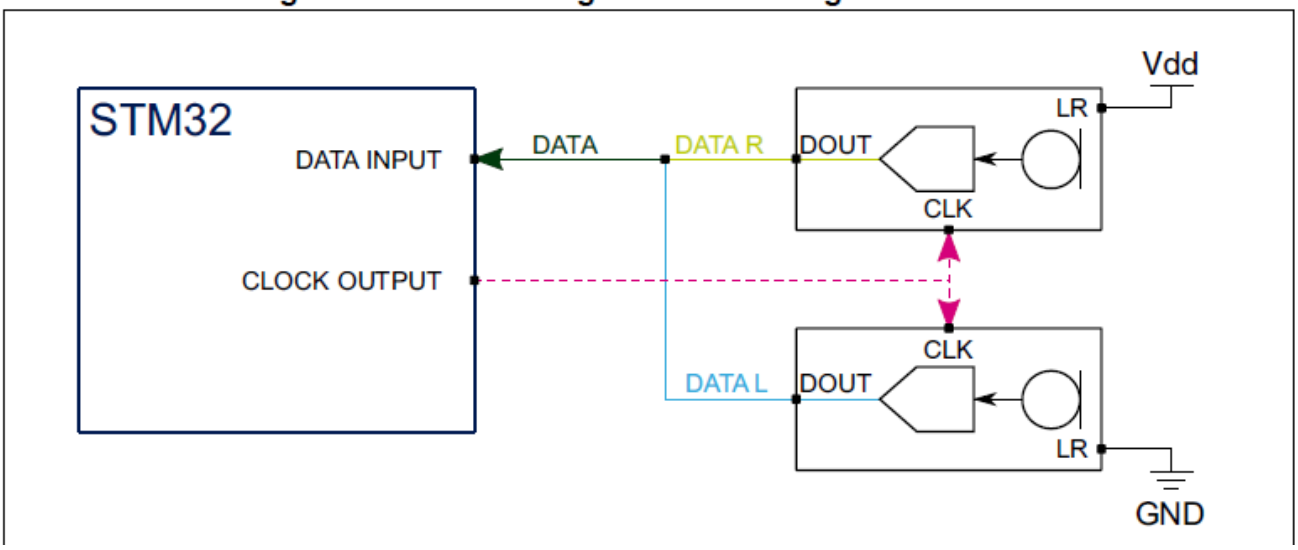
- On the falling edge of the clock, the microphone will generate valid data for half of the clock period, then goes into a high impedance state for the other half.

Figure 6. Left channel data pattern



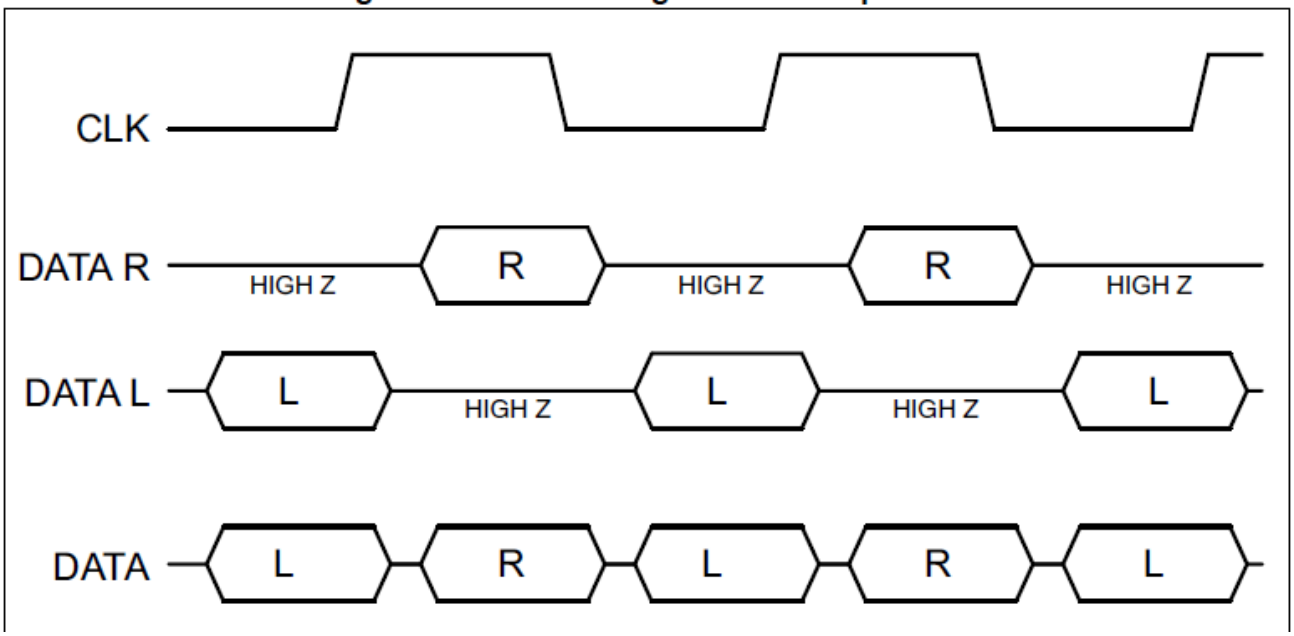
Stereo configuration:-

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.

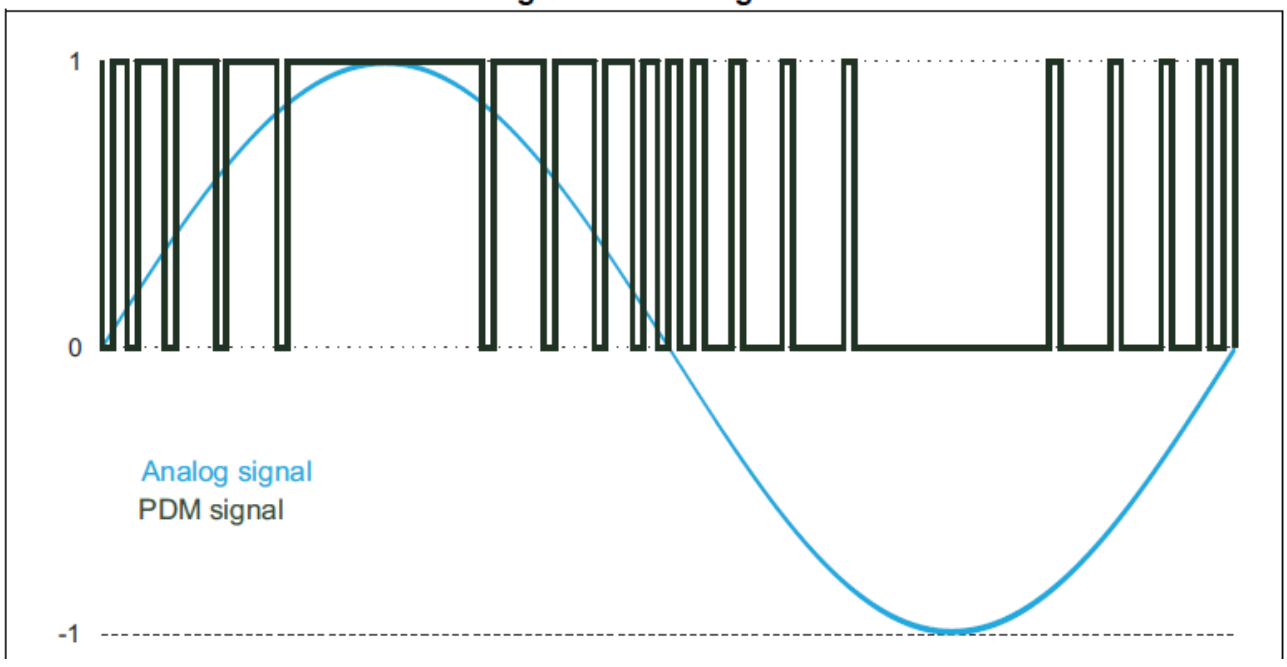
Figure 8. Stereo configuration data pattern



Pulse density modulation signal (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.

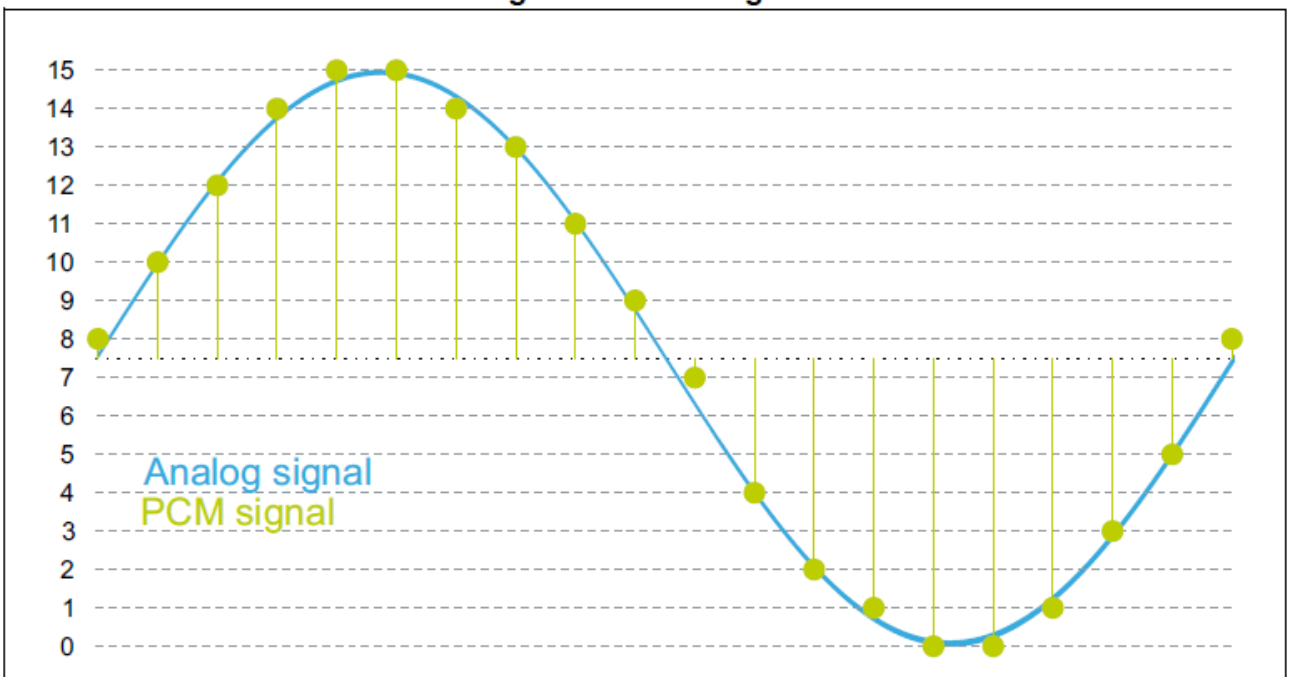
Figure 9. PDM signal



Pulse code modulation signal (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



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.

  - PDM Frequency = Audio Sampling Frequency * 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.

PDM2PCM software library:-

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

Algorithm functionality:-

- The PDM2PCM library has the function to decimate and filter out a Pulse Density Modulated (PDM) stream from a digital microphone, to convert it to a Pulse Code Modulated (PCM) signal output stream.
- The PCM output stream is implemented with 16-bit resolution.
- The sampling rate is not specified in the interface but it is agreed in this document that the PCM sampling rate used is 16 kHz.
- Various decimation factors can be configured, to adapt to various PDM clocks.
- A configurable high-pass filter and a digital volume are also proposed.

Module configuration:-

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

Module interfaces:-

- Two files are needed to integrate the PDM2PCM library, the pdm2pcm_glo.h header file and the right library file (according to target and tool chain).
- They contain all definitions and structures to be exported to the software integration framework.

APIs:-

- Five functions have a software interface to the main program:
- PDM_FilterInit.
- PDM_Filter_setConfig.
- PDM_Filter_getConfig.
- PDM_Filter_deInterleave.
- PDM_Filter.

PDM_FilterInit function:-

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

uint32_t PDM_FilterInit(PDM_Filter_Handler_t *pHandler);

Table 2. *PDM_FilterInit* function

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

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

PDM_Filter_setConfig function:-

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

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

Table 3. *PDM_Filter_setConfig* function

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

PDM_Filter_getConfig function:-

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

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

Table 4. *PDM_Filter_getConfig* function

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

PDM_Filter function:-

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

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

Table 5. *PDM_Filter* function

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

Static parameters structure:-

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

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

Table 7. Static parameters

Name	Description	Comment
bit_order	Specifies the bit order for input (MSB or LSB)	PDM_FILTER_BIT_ORDER_LSB (0x0000) PDM_FILTER_BIT_ORDER_MSB (0x0001)
endianness	Specifies if byte inversion is required	PDM_FILTER_ENDIANNESS_LE (0x0000) PDM_FILTER_ENDIANNESS_BE (0x0001)
high_pass_tap	Specifies the HP filter tap value	Coefficient value in Q31 format of the high-pass filter. If 0 the filter is not used.
in_ptr_channels	Specifies the number of channels in the input PDM stream	INTEGER NUMBER > 0 in_ptr_channels = 1 when used with one microphone.
out_ptr_channels	Specifies the number of channels in the output PCM stream.	INTEGER NUMBER > 0. out_ptr_channels=1 when used with one microphone.
pInternalMemory	Internal memory	Pointer to an array.

Dynamic parameters structure:-

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

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

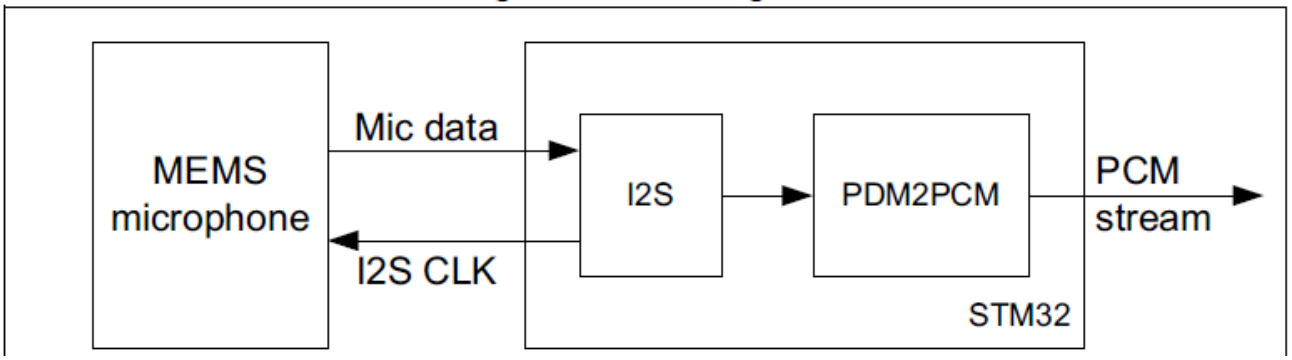
Table 8. Dynamic parameters

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

Processing steps:-

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

Figure 1. Block diagram



- The PDM data from the microphone are packed in 8-bit blocks, and then filtered and decimated.
- The frequency of the obtained PCM signal depends on the decimation factor configured before the library initialization.
- The decimation factors have been defined to get a PCM stream of the desired sampling frequency, depending on the PDM clock value.

Table 9. Decimation factors and corresponding frequencies

Decimation factor	PDM clock frequency	PCM sample rate
128	1.024 MHz	8 kHz
	2.048 MHz	16 kHz
	3.072 MHz	24 kHz
80	1.280 MHz	16 kHz
64	1.024 MHz	16 kHz
	2.048 MHz	32 kHz
	3.072 MHz	48 kHz
48	768 kHz	16 kHz
32	512 kHz	16 kHz
24	384 kHz	16 kHz
16	256 kHz	16 kHz

Library initialization:-

- Once the memory is allocated, some routines must be called to initialize the PDM2PCM library static memory:
- PDM_Filter_Init() has to be called each time the processing in the audio is stopped and started.
- PDM_Filter_setConfig() has to be called at least once before processing start, to set configurable parameter
- Furthermore, as the PDM2PCM library runs on STM32 devices, CRC HW block must be enabled and reset.
- The static and dynamic parameters structures must be allocated.
- Their types are defined in pdm2pcm_glo.h header. Example of allocation:

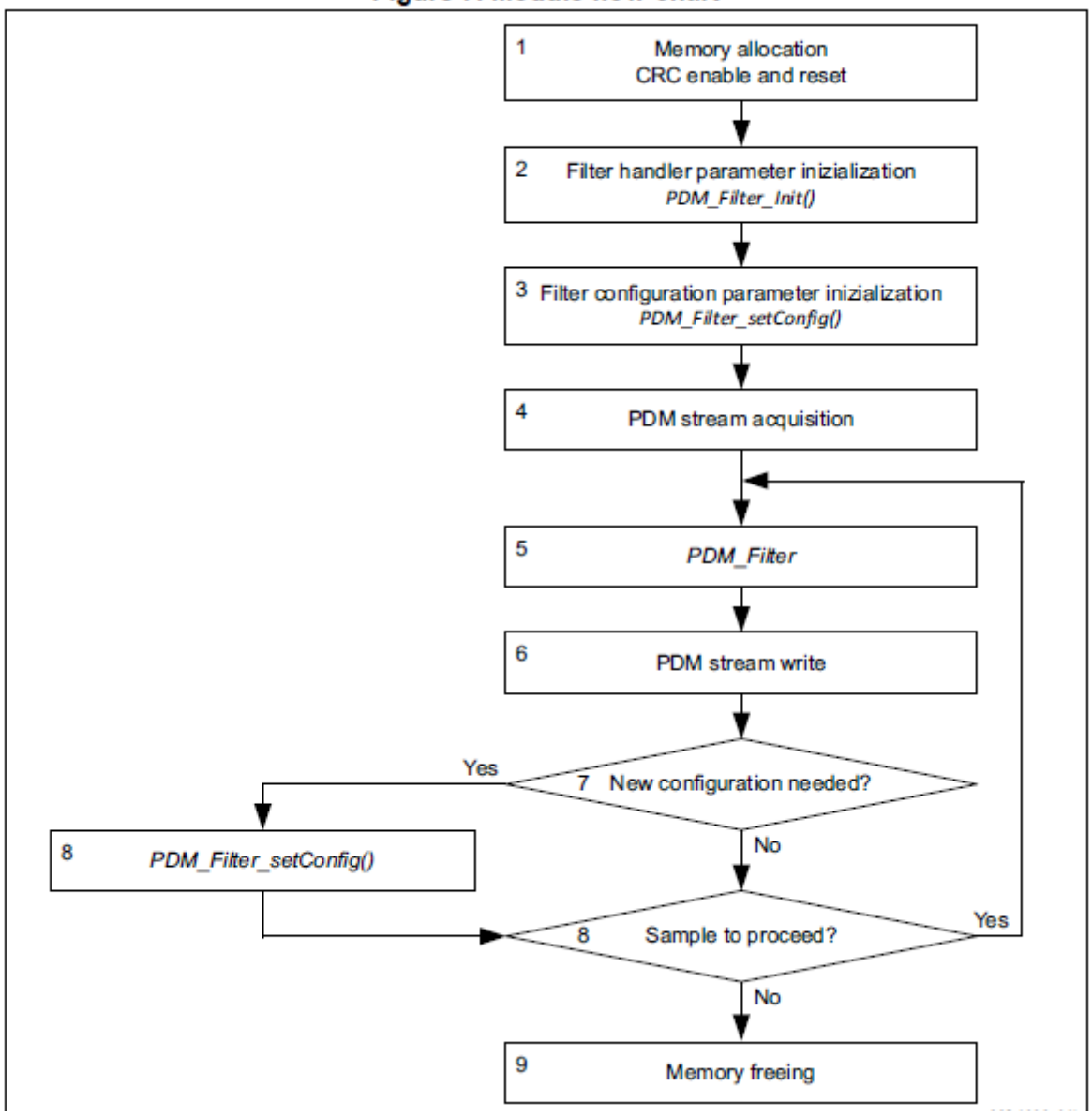
```
/*Enables and resets CRC-32 from STM32 HW */
__HAL_RCC_CRC_CLK_ENABLE();
CRC->CR = CRC_CR_RESET;

PDM_Filter_Handler_t PDM1_filter_handler;
PDM_Filter_Config_t PDM1_filter_config;

/* Initialize PDM Filter structure */
PDM1_filter_handler.bit_order = PDM_FILTER_BIT_ORDER_LSB;
PDM1_filter_handler.endianness = PDM_FILTER_ENDIANNESS_BE;
PDM1_filter_handler.high_pass_tap = 2122358088;
PDM1_filter_handler.out_ptr_channels = 1; PDM1_filter_handler.in_ptr_channels =
1;
PDM_Filter_Init((PDM_Filter_Handler_t *)(&PDM1_filter_handler));
```

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

Figure 7. Module flow-chart



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.

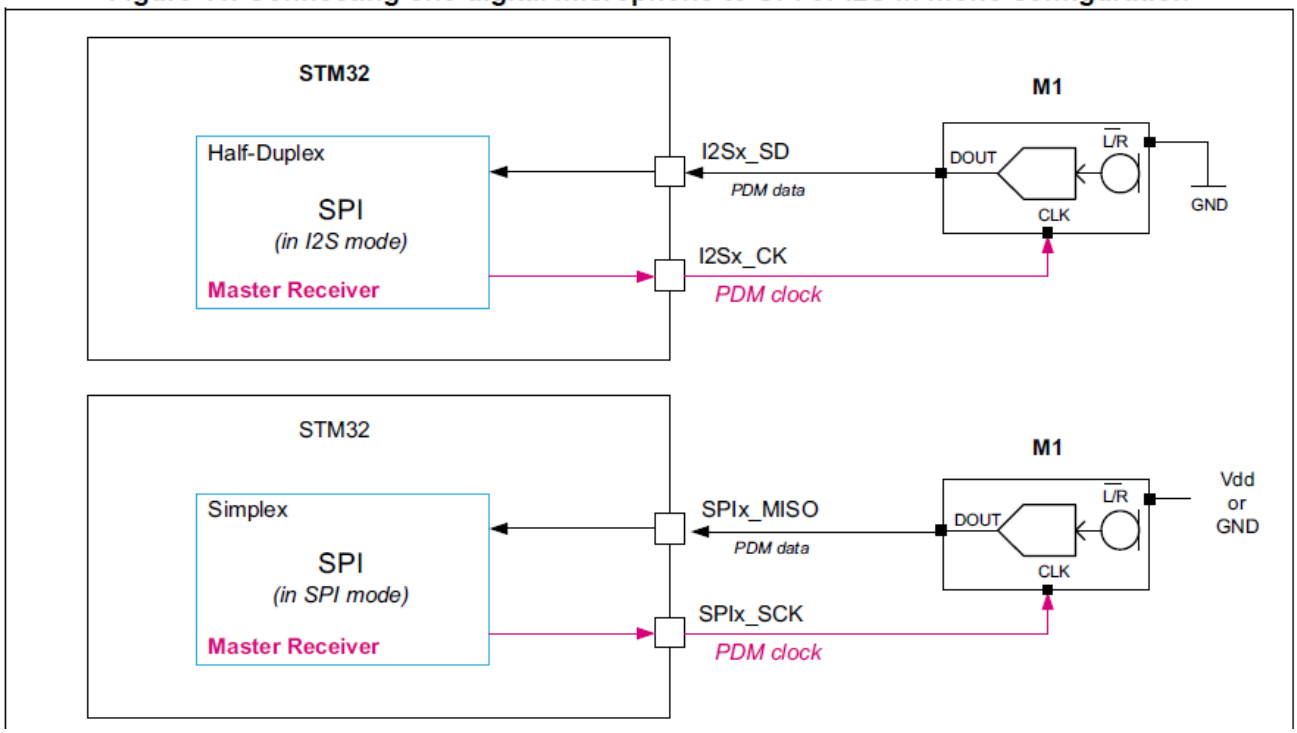
Serial peripheral interface/Inter-IC sound (I2S):-

- 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).
- 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.

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:

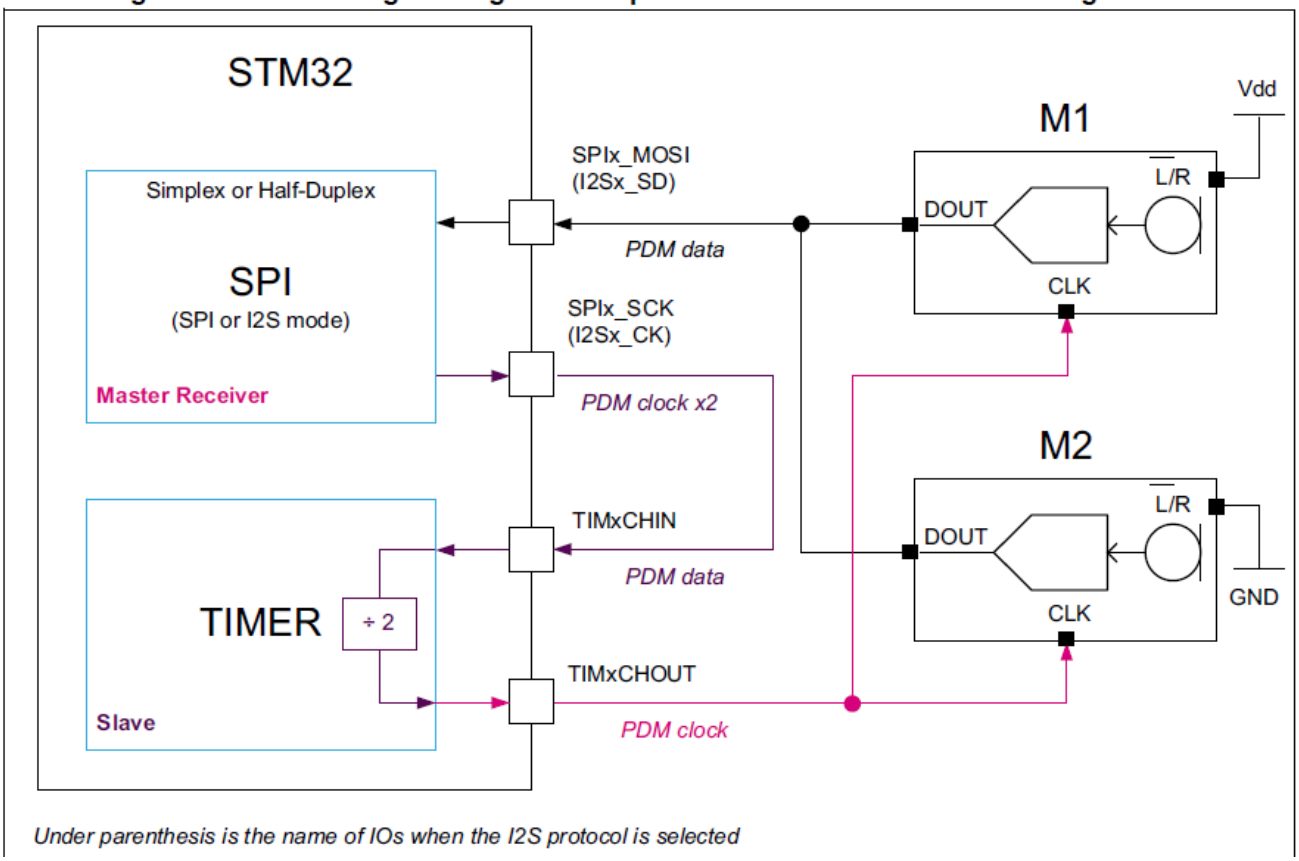
Bit:	15	14	13	12	11	...	0
Content:	$M1\_b_N$	$M1\_b_{N+1}$	$M1\_b_{N+2}$	$M1\_b_{N+3}$	$M1\_b_{N+4}$	...	$M1\_b_{N+15}$

$M1\_b_{xx}$ represents the data bits from the digital microphone 1, and $M1\_b_N$ is the older bit.

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.

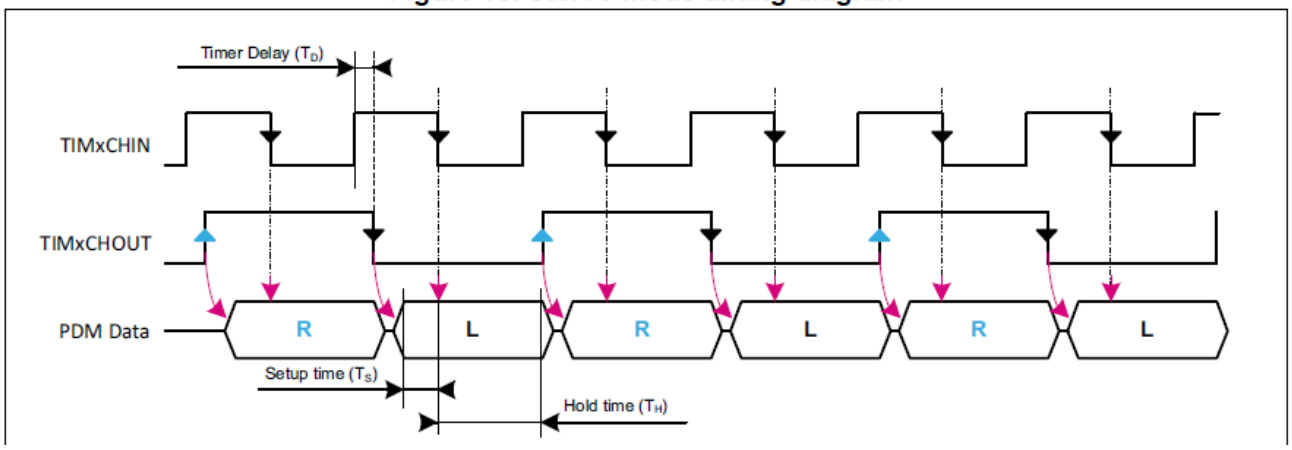
Figure 12. Connecting two digital microphone to SPI block in stereo configuration



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 (TS) of the samples provided by the microphones.
- For that purpose, the timer shall use a clock as fast as possible.
- The maximum delay (TD) 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_bxx represents the data bits from the digital microphone 1, and M1_bN is the older bit.
- M2_bxx represents the data bits from the digital microphone 2, and M2_bN is the older bit.

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.

PDM audio software decoding Library:-

Overview:-

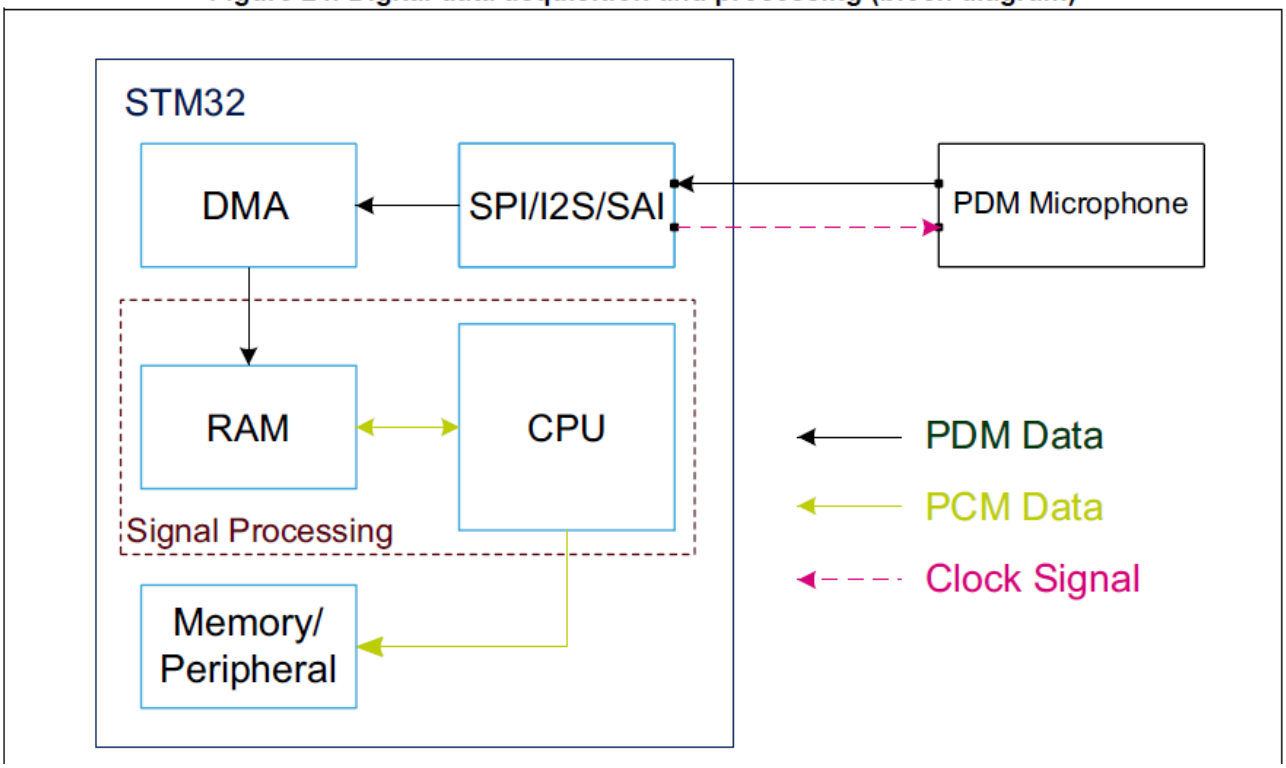
- PDM audio software decoding Library is an optimized software implementation for PDM signal decoding and audio signal reconstruction when connecting digital MEMS microphones with an STM32 microcontroller.
- This library implements several filters for the 1-bit PDM high frequency signal output from a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

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...).

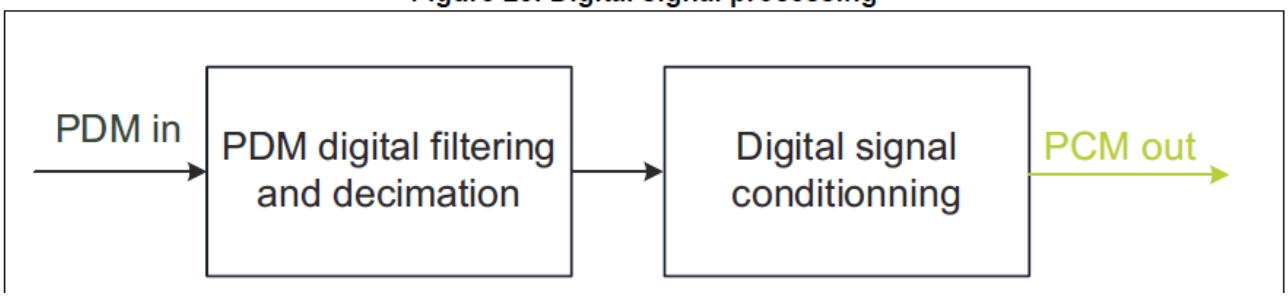
Figure 24. Digital data acquisition and processing (block diagram)



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

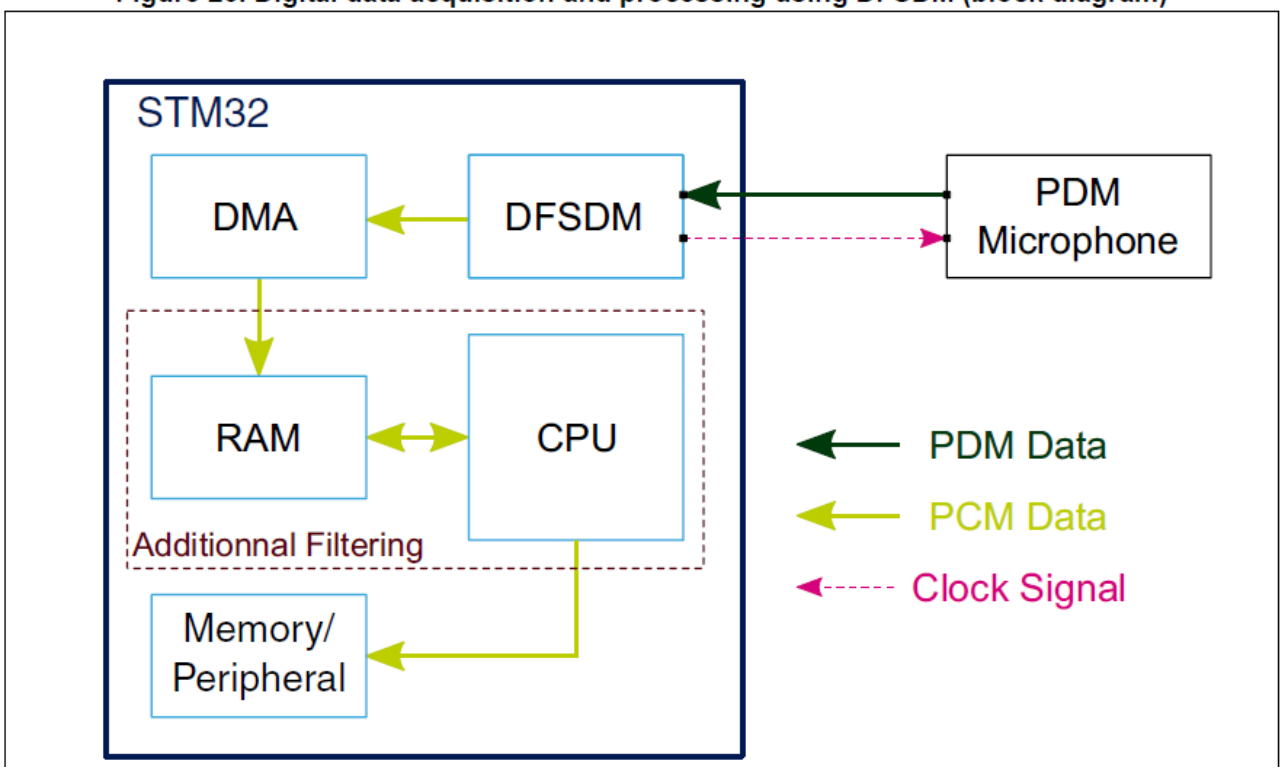
is further processed for proper signal conditioning implementing a low pass filter and a high pass filter.
- Both these filters can be enabled/disabled and configured (cut-off frequencies) by using the filter initialization function.

DFSDM filters for digital signal processing:-

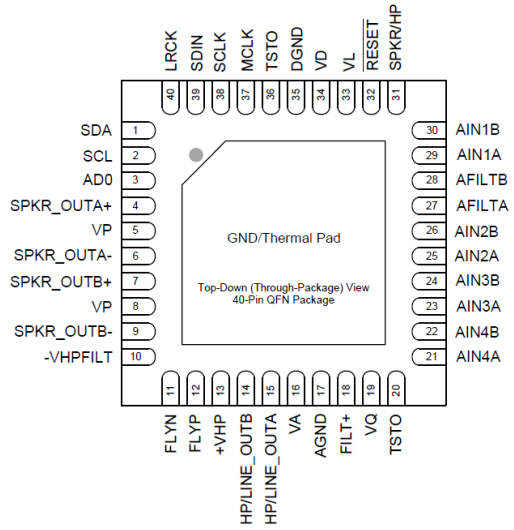
Digital data flow: acquisition and processing:-

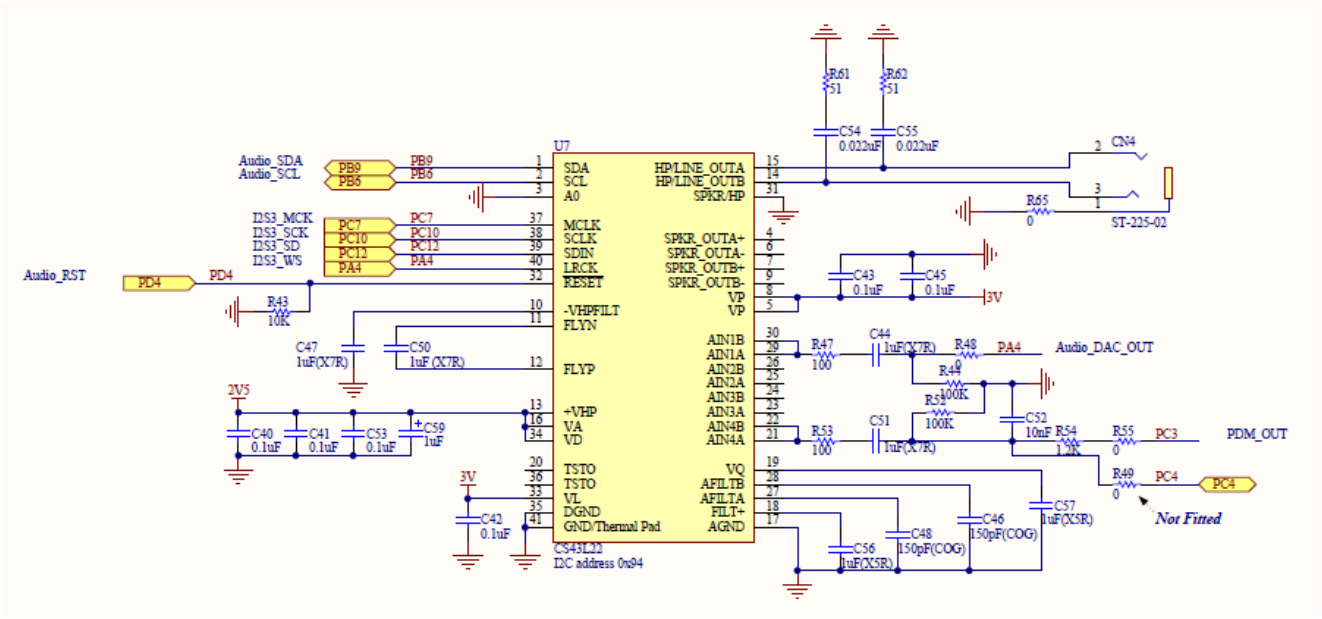
- The digital MEMS microphone outputs a PDM signal, which is a high frequency (1 to 3.25 MHz) stream of 1-bit digital samples.
- The data is acquired by the DFSDM serial transceiver that provides connection to the external Sigma-Delta modulator of the digital microphone.
- The digital filters perform 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)



Audio codec (CS43L22):-





Basic Architecture:-

- The CS43L22 is a highly integrated, low power, 24-bit audio DAC comprised of a Digital Signal Processing Engine, headphone amplifiers, a digital PWM modulator and two full-bridge power back-ends.
- Other features include battery level monitoring and compensation and temperature monitoring.
- The DAC is designed using multi-bit delta-sigma techniques and operates at an oversampling ratio of 128Fs, where Fs is equal to the system sample rate.
- The PWM modulator operates at a fixed frequency of 384 kHz.
- The power MOSFETs are configured for either stereo full-bridge or mono parallel full bridge output.
- The DAC operates in one of four sample rate speed modes: Quarter, Half, Single and Double.

- It accepts and is capable of generating serial port clocks (SCLK, LRCK) derived from an input Master Clock (MCLK).

Line Inputs:-

- 4 pairs of stereo analog inputs are provided for applications that require analog pass-through directly to the HP/Line amplifiers.
- This analog input portion allows selection from and configuration of multiple combinations of these stereo sources.

Line & Headphone Outputs:-

- The analog output portion of the CS43L22 includes a headphone amplifier capable of driving headphone and line-level loads.
- An on-chip charge pump creates a negative headphone supply allowing a full-scale output swing centered around ground.
- This eliminates the need for large DC-Blocking capacitors and allows the amplifier to deliver more power to headphone loads at lower supply voltages.

Speaker Driver Outputs:-

- The Class D power amplifiers drive 8 Ω (stereo) and 4 Ω (mono) speakers directly, without the need for an external filter.
- The power MOSFETS are powered directly from a battery eliminating the efficiency loss associated with an external regulator.
- Battery level monitoring and compensation maintains a steady output as battery levels fall.
- A temperature monitor continually measures the die temperature and registers when predefined thresholds are exceeded.

Fixed Function DSP Engine:-

- The fixed-function digital signal processing engine processes the PCM serial input data.
- Independent volume control, left/right channel swaps, mono mixes, tone control and limiting functions also comprise the DSP engine.

Digital Interface Formats:-

- The serial port operates in standard I²S or DSP Mode digital interface formats with varying bit depths from 16 to 24. Data is clocked into the DAC on the rising edge of SCLK.

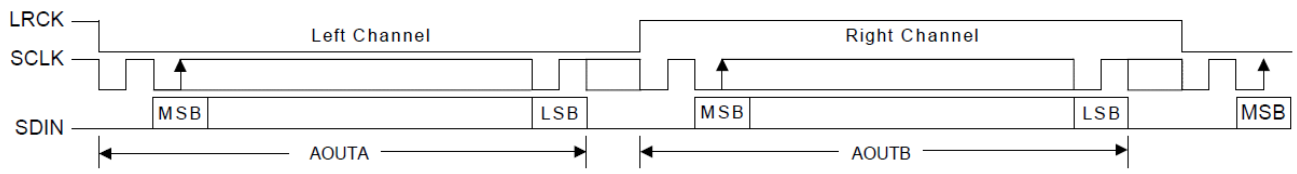


Figure 12. I²S Format

DSP Mode:-

- In DSP Mode, the LRCK acts as a frame sync for 2 data-packed words (left and right channel) input on SDIN.
- The MSB is input on the first SCLK rising edge after the frame sync rising edge.
- The right channel immediately follows the left channel.

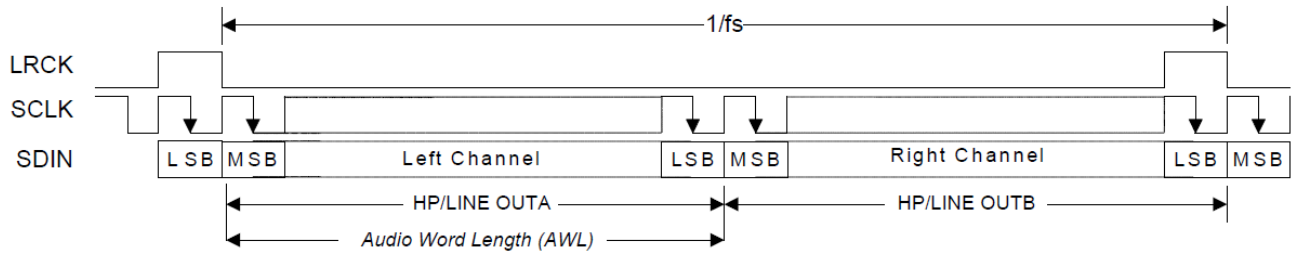


Figure 15. DSP Mode Format)

Initialization:-

- The CS43L22 enters a Power-Down state upon initial power-up. The interpolation and decimation filters, delta-sigma and PWM modulators and control port registers are reset. The internal voltage reference, and switched-capacitor low-pass filters are powered down.
- The device will remain in the Power-Down state until the RESET pin is brought high. The control port is accessible once RESET is high and the desired register settings can be loaded per the interface descriptions.
- Once MCLK is valid, the quiescent voltage, VQ, and the internal voltage reference, FILT+, will begin powering up to normal operation.
- The charge pump slowly powers up and charges the capacitors. Power is then applied to the headphone amplifiers and switched-capacitor filters, and the analog/digital outputs enter a muted state.
- Once LRCK is valid, MCLK occurrences are counted over one LRCK period to determine the MCLK/LRCK frequency ratio and normal operation begins.

Recommended Power-Up Sequence:-

1. Hold RESET low until the power supplies are stable.
2. Bring RESET high.
3. The default state of the “Power Ctl. 1” register (0x02) is 0x01. Load the desired register settings while keeping the “Power Ctl 1” register set to 0x01.
4. Load the required initialization settings.
5. Apply MCLK at the appropriate frequency. SCLK may be applied or set to master at any time; LRCK may only be applied or set to master while the PDN bit is set to 1.
6. Set the “Power Ctl 1” register (0x02) to 0x9E.
7. Bring RESET low if the analog or digital supplies drop below the recommended operating condition to prevent power glitch related issues.

Recommended Power-Down Sequence:-

To minimize audible pops when turning off or placing the DAC in standby,

1. Mute the DAC's and PWM outputs.
2. Disable soft ramp and zero cross volume transitions.
3. Set the "Power Ctl 1" register (0x02) to 0x9F.
4. Wait at least 100 µs.

- The device will be fully powered down after this 100 µs delay.
- Prior to the removal of the master clock (MCLK), this delay of at least 100 µs must be implemented after step 3 to avoid premature disruption of the DAC's power down sequence.
- A disruption in the device's power down sequence (i.e. removing the MCLK signal before this 100 µs delay) has consequences on both the headphone and PWM speaker amplifiers: The charge pump may stop abruptly, causing the headphone amplifiers to drive the outputs up to the +VHP supply.
- Also, the last state of each '+' and '-' PWM output terminal before the premature removal of MCLK could randomly be held at either VP or AGND.
- When this event occurs, it is possible for each PWM terminal to output opposing potentials, creating a DC source into the speaker voice coil.
- The disruption of the device's power down sequence may also cause clicks and pops on the output of the DAC's as the modulator holds the last output level before the MCLK signal was removed.

5. MCLK may be removed at this time.
6. To achieve the lowest operating quiescent current, bring RESET low. All control port registers will be reset to their default state.

Required Initialization Settings:-

- Various sections in the device must be adjusted by implementing the initialization settings shown below after power-up sequence step 3. All performance and power consumption measurements were taken with the following settings:

1. Write 0x99 to register 0x00.
2. Write 0x80 to register 0x47.
3. Write '1'b to bit 7 in register 0x32.
4. Write '0'b to bit 7 in register 0x32.
5. Write 0x00 to register 0x00.

CONTROL PORT OPERATION:-

- The control port operates using an I^2C interface with the CS43L22 acting as a slave device.

I²C Control:-

- SDA is a bidirectional data line. Data is clocked into and out of the device by the clock, SCL.
- The AD0 pin sets the LSB of the chip address; '0' when connected to DGND, '1' when connected to VL.
- This pin may be driven by a host controller or directly connected to VL or DGND.
- The AD0 pin state is sensed and the LSB of the chip address is set upon the release of the RESET signal (a low-to-high transition).
- The signal timings for a read and write cycle are shown in Figures.
- A Start condition is defined as a falling transition of SDA while the clock is high.
- A Stop condition is defined as a rising transition of SDA while the clock is high.
- All other transitions of SDA occur while the clock is low.
- The first byte sent to the CS43L22 after a Start condition consists of a 7-bit chip address field and a R/W bit (high for a read, low for a write).
- The upper 6 bits of the address field are fixed at 100101.
- To communicate with the CS43L22, the chip address field, which is the first byte sent to the CS43L22, should match 100101 followed by the setting of the AD0 pin.
- The eighth bit of the address is the R/W bit. If the operation is a write, the next byte is the Memory Address Pointer (MAP), which selects the register to be read or written. If the operation is a read, the contents of the register pointed to by the MAP will be output.
- Setting the auto-increment bit in MAP allows successive reads or writes of consecutive registers. Each byte is separated by an acknowledge bit.
- The ACK bit is output from the CS43L22 after each input byte is read and is input to the CS43L22 from the microcontroller after each transmitted byte.

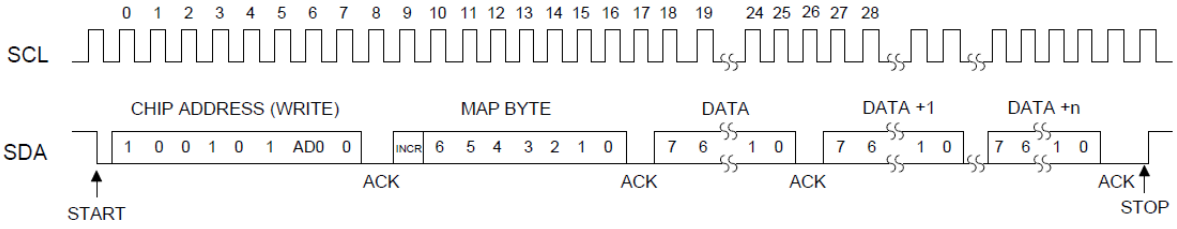


Figure 16. Control Port Timing, I²C Write

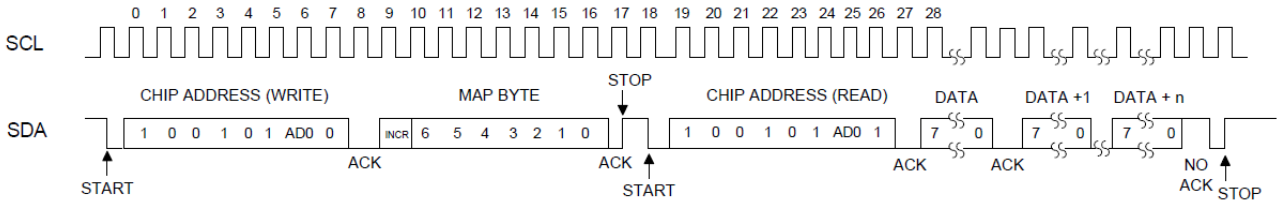


Figure 17. Control Port Timing, I²C Read

- Since the read operation cannot set the MAP, an aborted write operation is used as a preamble. As shown in Figure, the write operation is aborted after the acknowledge for the MAP byte by sending a stop condition.
- The following pseudocode illustrates an aborted write operation followed by a read operation.
- Send start condition.
- Send 10010100 (chip address & write operation).
- Receive acknowledge bit.
- Send MAP byte, auto-increment off.
- Receive acknowledge bit.
- Send stop condition, aborting write.
- Send start condition.
- Send 10010101 (chip address & read operation).
- Receive acknowledge bit.
- Receive byte, contents of selected register.
- Send acknowledge bit.
- Send stop condition.
- Setting the auto-increment bit in the MAP allows successive reads or writes of consecutive registers.
- Each byte is separated by an acknowledge bit.

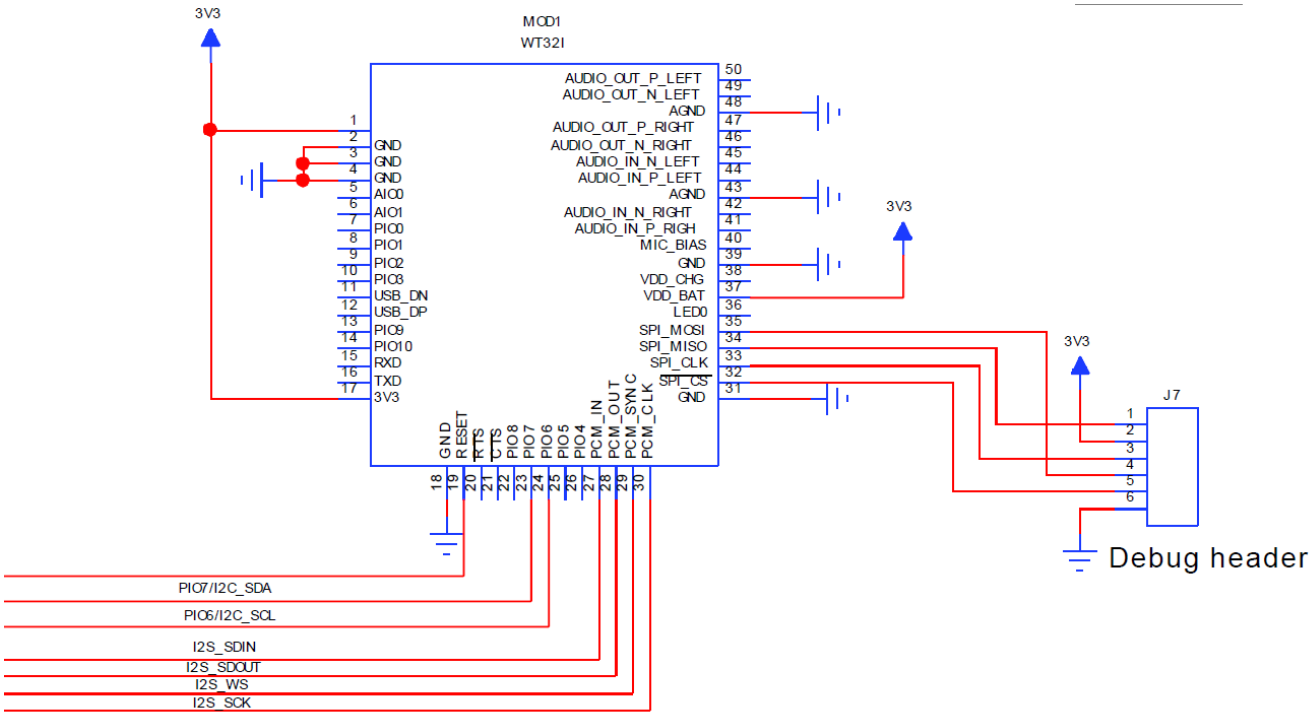
Memory Address Pointer (MAP):-

- The MAP byte comes after the address byte and selects the register to be read or written.
- Refer to the pseudo code above for implementation details.

Map Increment (INCR):-

- The device has MAP auto-increment capability enabled by the INCR bit (the MSB) of the MAP.
- If INCR is set to 0, MAP will stay constant for successive I^2C writes or reads.
- If INCR is set to 1, MAP will auto-increment after each byte is read or written, allowing block reads or writes of successive registers.

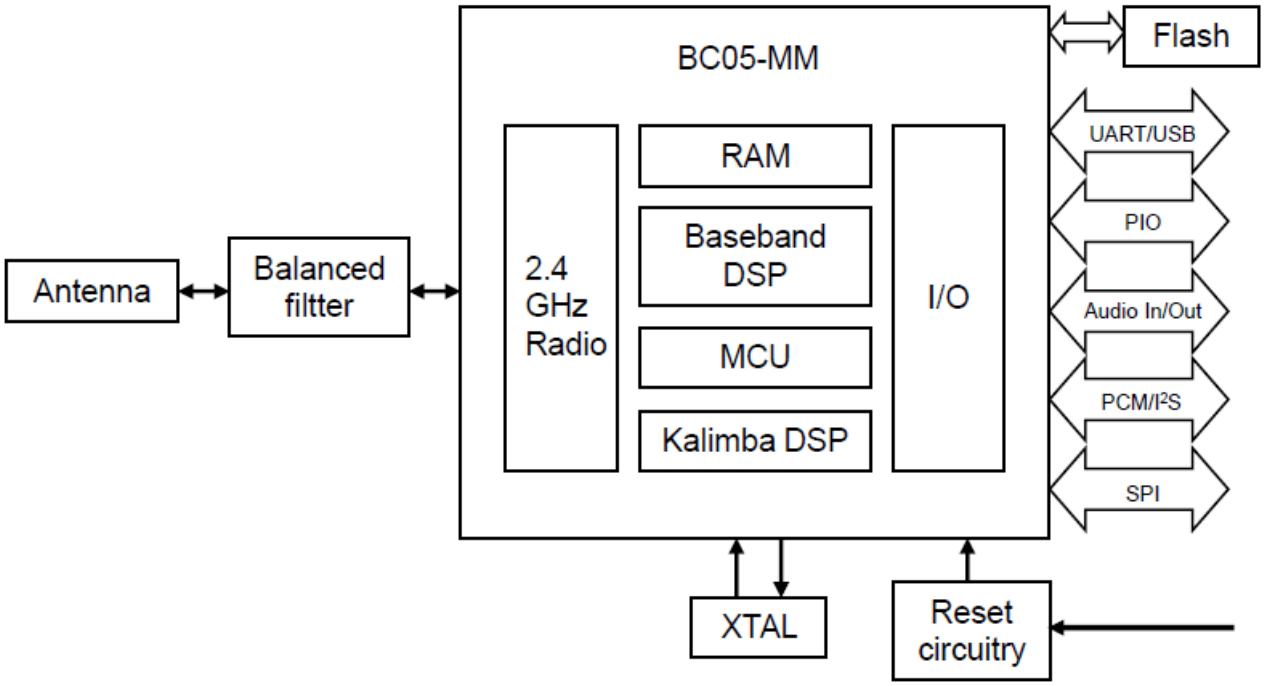
Audio Bluetooth module (WT32I-A-AI6):-



Description:-

- WT32i is an audio specific Bluetooth 3.0 module with excellent radio frequency performance and enhanced audio features, enabling a best in class Bluetooth audio experience.
- In addition to a certified Bluetooth radio and software stack, WT32i also contains a DSP, stereo audio codec, and battery charger making it ideal for fixed and portable audio applications.
- WT32i includes Bluegiga's iWRAP6 Bluetooth stack software which implements A2DP, AVRCP v.1.5 profiles and supports aptX® and AAC audio codecs for stereo audio applications.
- For hands-free applications iWRAP6 software also supports HFP v.1.6, HSP, MAP and PBAP and CVC® echo cancellation software.
- For data communications to Android and iOS applications iWRAP6 also implements Bluetooth Serial Port Profile (SPP) and Apple iAP profiles.
- WT32i is an ideal solution for developers who want to quickly integrate the latest Bluetooth audio technologies without the time and costs typically involved with a Bluetooth audio chipset design.

Block diagram:-



BC05-MM

- The BlueCore®5-Multimedia External is a single-chip radio and baseband IC for Bluetooth 2.4GHz systems.
- It provides a fully compliant Bluetooth v3.0 specification system for data and voice. BlueCore5-Multimedia External contains the Kalimba DSP coprocessor with double the MIPS of BlueCore3-Multimedia External, supporting enhanced audio applications.

XTAL

- The reference clock of WT32i is generated with 26 MHz crystal.
- All BC05-MM internal digital clocks are generated using a phase locked loop, which is locked to the frequency of either the 26 MHz crystal or an internally generated watchdog clock frequency of 1 kHz.

RESET CIRCUITRY

- The internal reset circuitry keeps BC05-MM in reset during boot in order for the supply voltages to stabilize.
- This is to prevent corruption of the flash memory during booting.

BALANCED FILTER

- The internal balanced filter provides optimal impedance matching and band pass filtering in order to achieve lowest possible in-band and out-of-band emissions.

ANTENNA

- The antenna is a ceramic chip antenna with high efficiency.
- The antenna is insensitive to surrounding dielectric materials and requires only a small clearance underneath which makes it compatible with previous WT32I designs

and well suitable for designs with high density.

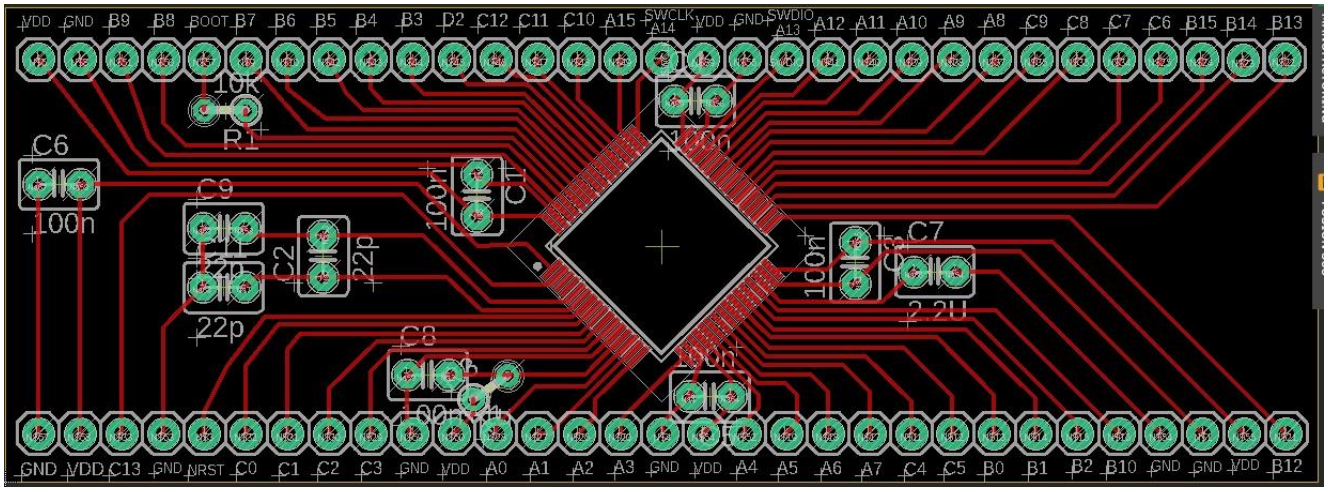
FLASH

- 16 Mbit flash memory is used for storing the Bluetooth protocol stack and Virtual Machine applications.
- It can also be used as an optional external RAM for memory-intensive applications.

PCB Designs:-

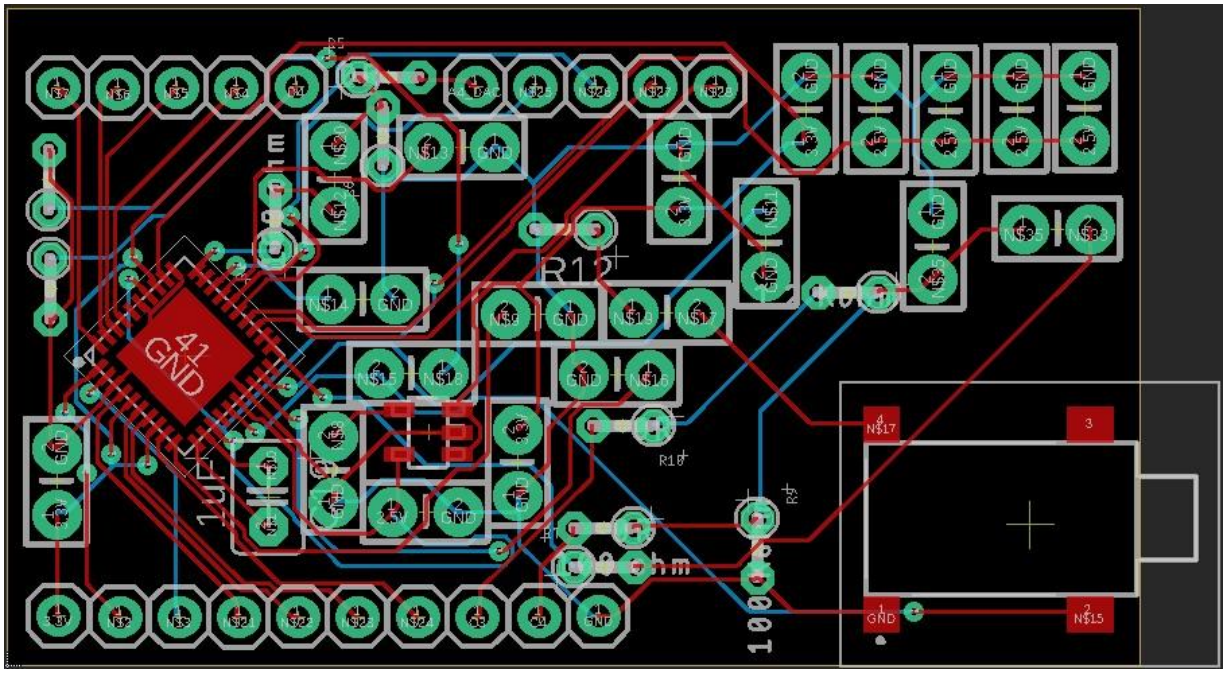
STM32F446RC (Version 1):-

- First of all we design a PCB that have STM32F446RC with all its 64 pins.

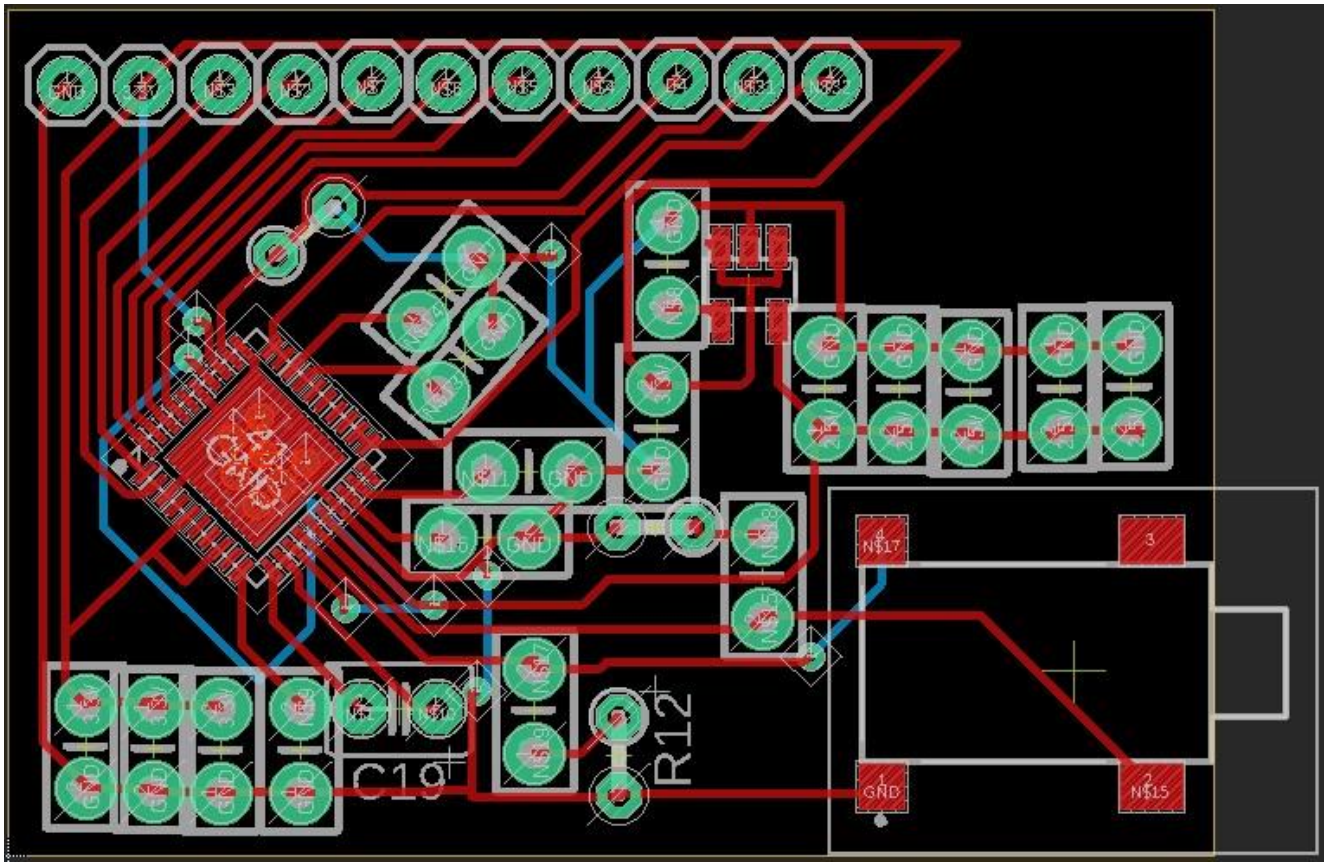


CS43L22-CNZ with Analog Pins:-

- After that we design a PCB that have the IC of the audio codec and all its needed components to work well.

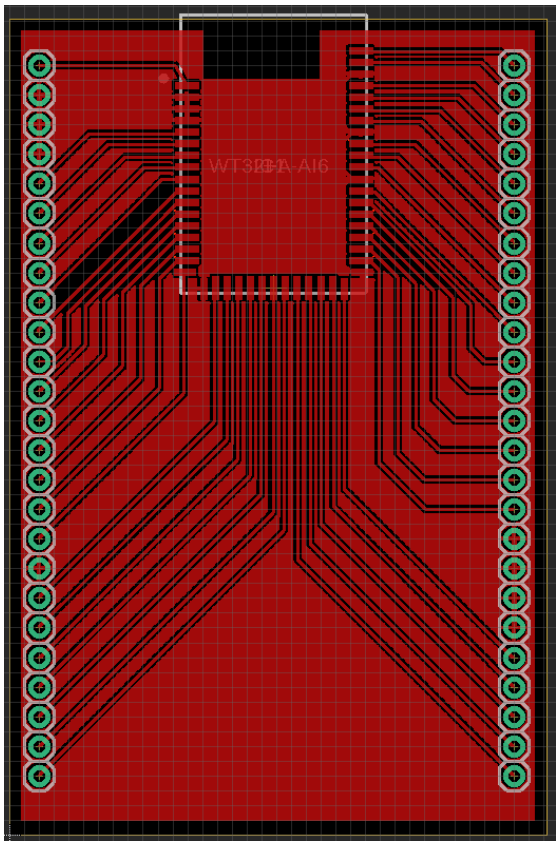


CS43L22-CNZ without Analog Pins:-



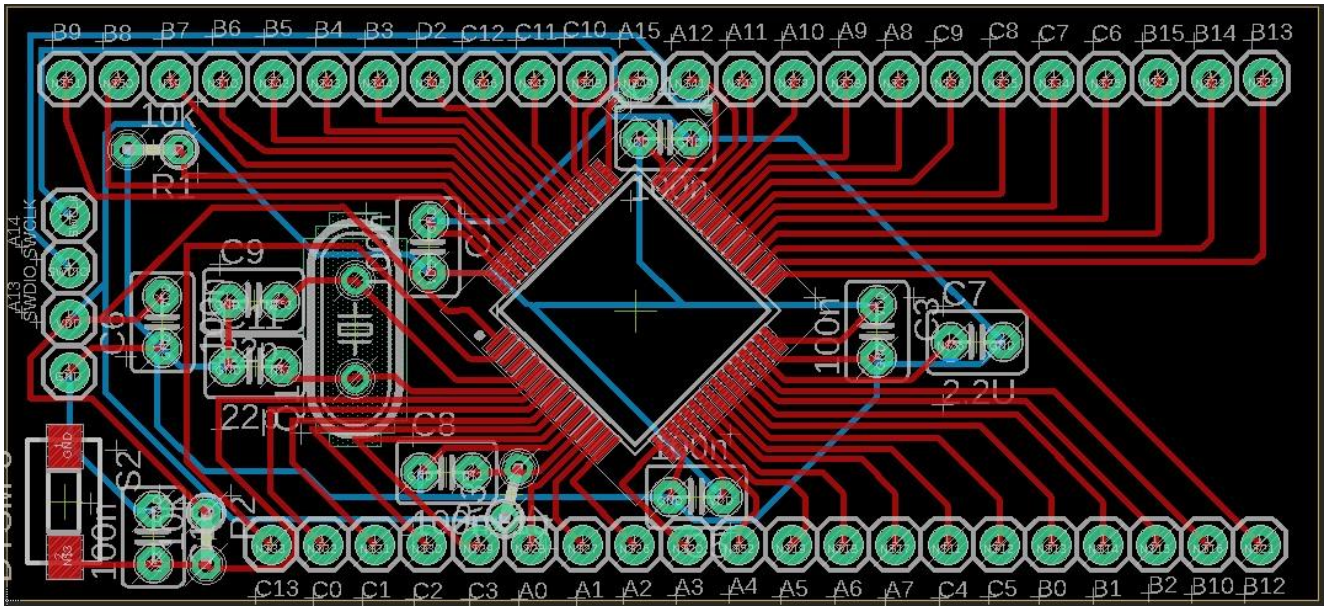
Audio Bluetooth Module (WT32I-A-AI6):-

- After that we design a PCB that have the audio Bluetooth module.



STM32F446RC (Version 2):-

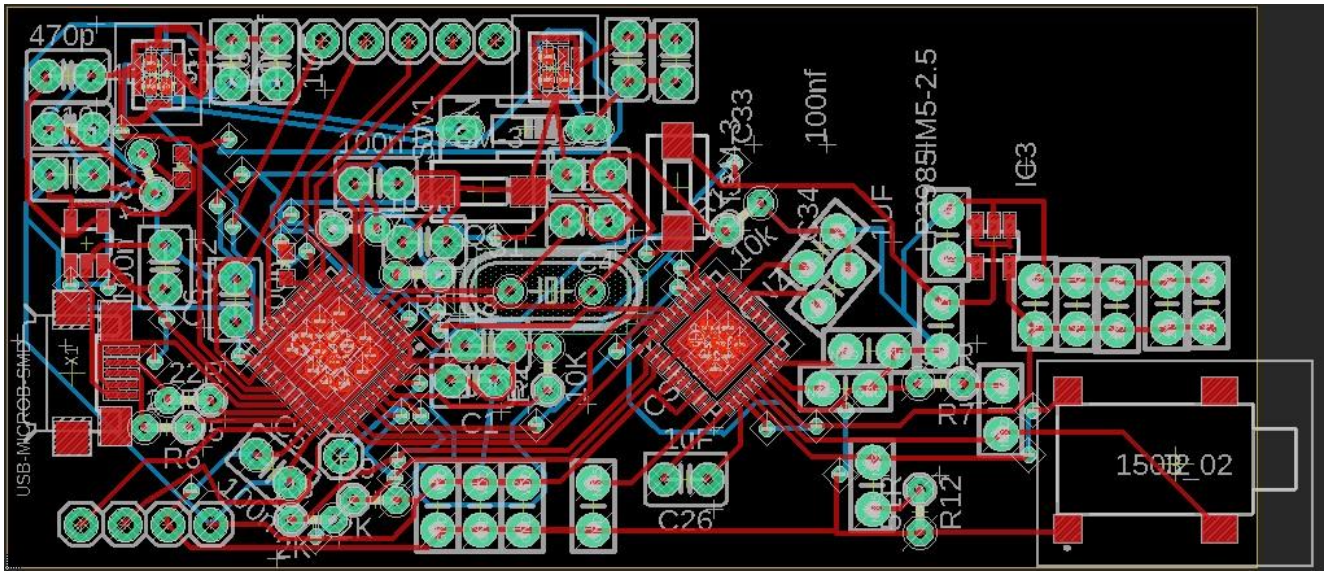
- After that we design PCB that have STM32F446RC but with only used pins in our project to reduce the size of the PCB and of course the size of the project.



Our future work:-

- We chose STM32F413CG instead of STM32F446RC because the new one has DFSDM peripheral that will help us in our DSP work as we described before.
- And here is the last design of our PCB with all the components (STM32F413CG, audio codec IC, 2 MEMs microphones and all other needed components).

STM32F413CG (Version 1) with Audio Codec:-



References used in this project:-

- AN3126:- Audio and waveform generation using the DAC in STM32 microcontrollers.
- AN3997:- Audio playback and recording using the STM32F4DISCOVERY.
- AN3998:- PDM audio software decoding on STM32 microcontrollers.
- AN4031:- Using the STM32F2, STM32F4 and STM32F7 Series DMA controller.
- AN4566:- Extending the DAC performance of STM32 microcontrollers.
- AN4739:- STM32Cube firmware examples for STM32F4 Series.
- AN5027:- Interfacing PDM digital microphones using STM32 32-bit Arm® Cortex® MCUs.
- AN5073:- Receiving S/PDIF audio stream with the STM32F4/F7/H7 Series.

- UM2372:- STM32Cube PDM2PCM software library for the STM32F4/F7/H7 Series.

- RM0390:- Reference manual STM32F446xx advanced Arm®-based 32-bit MCUs.
- PM0214:- Programming manual STM32 Cortex®-M4 MCUs and MPUs programming Manual.

- STM32F446xC/E:- datasheet.
- STM32F446xC/xE:- Errata sheet.
- STM32F446xC/xE:- device limitations.