AN4229
Application note

How to implement a vocoder solution using STM32 microcontrollers

Introduction

This application note describes the vocal codecs (vocoder) available for STM32 32-bit ARM® Cortex® microcontrollers. The supported codecs are G.711, G.726, IMA-ADPCM and Speex.

It comes with examples of implementation on STM32F4DISCOVERY (referred as DISCOVERY board hereafter), STM322xG-EVAL and STM324xG-EVAL (referred as EVAL boards hereafter).

These examples are developed in such a way that they can be easily ported to any other STM32 microcontroller and board.

The vocoder firmware package is provided upon request, for more details please contact your local ST sales representative.

Table 1. Applicable products

Type	Applicable products
Microcontrollers	STM32 32-bit ARM Cortex MCU

Contents

List of tables

List of figures

1 Vocoder overview and comparison

A vocoder is a speech encoding, decoding and filtering application used to reproduce the human voice.

In this application note, four speech codecs are implemented: G.711, G.726, IMA-ADPCM and Speex. In the following sections, a short overview of these codecs is presented.

1.1 G.711

The G.711 codec is an ITU-T (telecommunication section of the international telecommunication union) standard for the compression of voice frequency signals. Its principal features are:

- Sampling frequency: 8 kHz
- 64 Kbit/s bitrate (8 kHz sampling frequency, 8-bit samples)
- Compression ratio: 1:2
- Used with ITU-T recommendations G.726

G.711 defines two main compression algorithms, the µ-law algorithm (used in North America and Japan) and A-law algorithm (used in Europe and the rest of the world). Both are logarithmic, but A-law was specifically designed to reduce computer processing requirements.

G.711 is primarily used in telephony, but it can also be used for fax communication over IP networks.

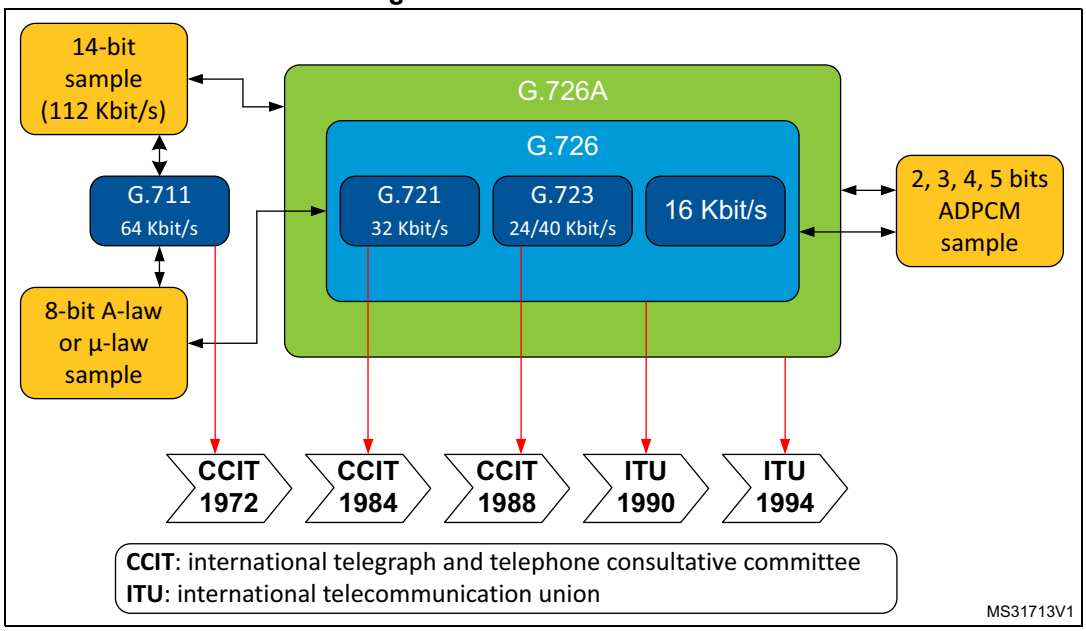
1.2 G.726

G.726 is an ITU-T ADPCM speech codec standard covering the transmission of voice at rates of 16, 24, 32, and 40 Kbit/s; actually, it supersedes both G.721 (ADPCM at 32 Kbits/s) and G.723 (ADPCM at 24 and 40 Kbits/s). It also introduces a new 16 Kbit/s rate (*Figure 1*). G.726 main features are:

- Sampling frequency: 8 kHz
- Bitrates: 16 Kbit/s, 24 Kbit/s, 32 Kbit/s and 40 Kbit/s
- Uses adaptive differential pulse code modulation (ADPCM)

In addition to DECT (digital enhanced cordless telecommunications) telephony, where G.726 is widely used, it is also used in several fields such as data and voice carrying (depending on the selected channel bitrate).

Figure 1. G.726 versus G.711



1.3 IMA-ADPCM

ADPCM (adaptive differential pulse code modulation) is a variant of DPCM that varies the size of the quantization step, to allow further reduction of the required bandwidth for a given signal-to-noise ratio.

IMA-ADPCM is a legacy audio codec developed by interactive multimedia association (IMA) with the following characteristics:

- Sampling frequencies: 8 (only frequency supported here), 11.025, 22.05, or 44.1 kHz.
- Compression ratio: 1:4

The main advantage of the IMA ADPCM compression algorithm reside in its simplicity. The algorithm is not limited to voice signals and can operate at any input sampling rate thus enabling compression of high quality audio as well.

1.4 Speex

The Speex codec is an open-source, patent-free and royalty-free software dedicated to speech compression and decompression.

Speex is based on CELP (code excited linear prediction) and designed to compress voice at bitrates ranging from 2 to 44 Kbit/s.

The features of Speex include:

- Three different sampling rates: narrowband (8 kHz), wideband (16 kHz) and ultra-wideband (32 kHz)
- Wide range of bitrates available (from 2 Kbit/s to 44 Kbit/s)
- Quality: ranges from 0 to 10
- Variable complexity: ranges from 1 to 10
- Variable bitrate (VBR)
- Voice activity detection (VAD)
- Perceptual enhancement…

Speex is targeted at voice over IP (VoIP) and file-based compression. The design goals have been to make a codec that would be optimized for high quality speech and low bit rate. Also, several other applications support the Speex codec such as streaming applications, videogames and audio processing applications.

1.5 Vocoder comparison

Table 2 summarizes and compares general and technical information for the previously presented codecs.

Table 2. Codecs comparison

Codec name	G.711	G.726	IMA-ADPCM	Speex
Creator	ITU-T	ITU-T	IMA (interactive multimedia association)	Xiph.Org foundation
Algorithm	Companding A-law or µ-law, PCM, Lossy	ADPCM, Lossy	Adaptive differential pulse code modulation)	CELP (code excited linear prediction)
License	GPL	GPL	Free	Patent-free (BSD license)
Bitrate (Kbits/s)	64	16, 24, 32, 40	32	2.15 to 24.6 (NB) 4 to 44.2(WB)
Sampling rate (kHz)	8	8	8	8, 16, 32
Bits per input sample	16	13	16	16
Bits per output sample	8	2, 3, 4, 5	4	Latency: 30 ms (NB), 34 ms (WB)
Application	– Voice recording and playback – VoIP – Automated announcement systems – Intercom – Walkie-talkie	– Voice recording and playback – VoIP – Automated announcement systems	– Speech compression in telecommunications	– Voice recording and playback – VoIP – Streaming applications (teleconference)

Table 2. Codecs comparison (continued)

Codec name	G.711	G.726	IMA-ADPCM	Speex
Benefits	– Designed to deliver precise transmission of speech – Very low processing overheads	– Uses 32 Kbits which is half the rate of G.711 codec and hence increases the usable network capacity by 100% – Very much used on international trunks in the phone network	– Transmit only the difference between the real and the predicted value – Predict the current signal value from previous value	– Integration of NB and WB in the same bitstream – Wide range of bitrates – Dynamic bitrate switching and ariable bitrate (VBR) – Voice activity detection (VAD, integrated with VBR) – Variable complexity – Intensity stereo encoding option
Drawbacks	– Poor network efficiency – Lacks missing packet interpolation	– Not well-suited for sound effects	– Low compression rate (equal to ¼)	– Quality is permanently degraded to reduce file size – By only specifying quality parameter, there's no guaranty about the final average bitrate

1.6 Vocoder performance

Table 3 to *Table 6* show measurements for G.711, G.726, IMA-ADPCM and Speex codecs. Performance data are measured on STM32F4 device at System Clock equal to 168MHz with 5 wait states and compilation optimization parameter set to high. Speex data measurements are done on STM32F4DISCOVERY while all the other codecs are tested on STM3240G-EVAL board.

Table 3. G.711 performance

Parameter	G.711 encoder		G.711 decoder	
	µ-law	A-law	µ-law	A-law
Flash memory (bytes)	94	78	84	62
RAM (bytes)	0	0	0	0
Load (MHz)	0.38	0.3	0.22	0.2

Table 4. G.726 decoder performance

Parameter (1)	G.726 Decoder		
	μ-law	A-law	PCM
Flash memory (bytes)	2026	2018	1604
RAM (bytes)	144	144	144
Load (MHz)	18.06	17.55	11.76

1. Bitrate: 32 Kbits/s

Table 5. IMA-ADPCM performance

Parameter	IMA-ADPCM encoder	IMA-ADPCM decoder
Flash memory (bytes)	416	416
RAM (bytes)	16	16
Load (MHz)	0.62	0.51

Table 6. Speex encoder performance

Quality (1)	Load (MHz)	
	Narrowband	Wideband
1	40.8	53.4
2	33.6	46.2
3	37.2	49.8
4	37.2	64.2
5	52.2	56.4
6	52.2	70.2
7	44.4	98.4
8	44.4	114
9	72.6	100.8
10	59.4	-

1. Complexity = 1

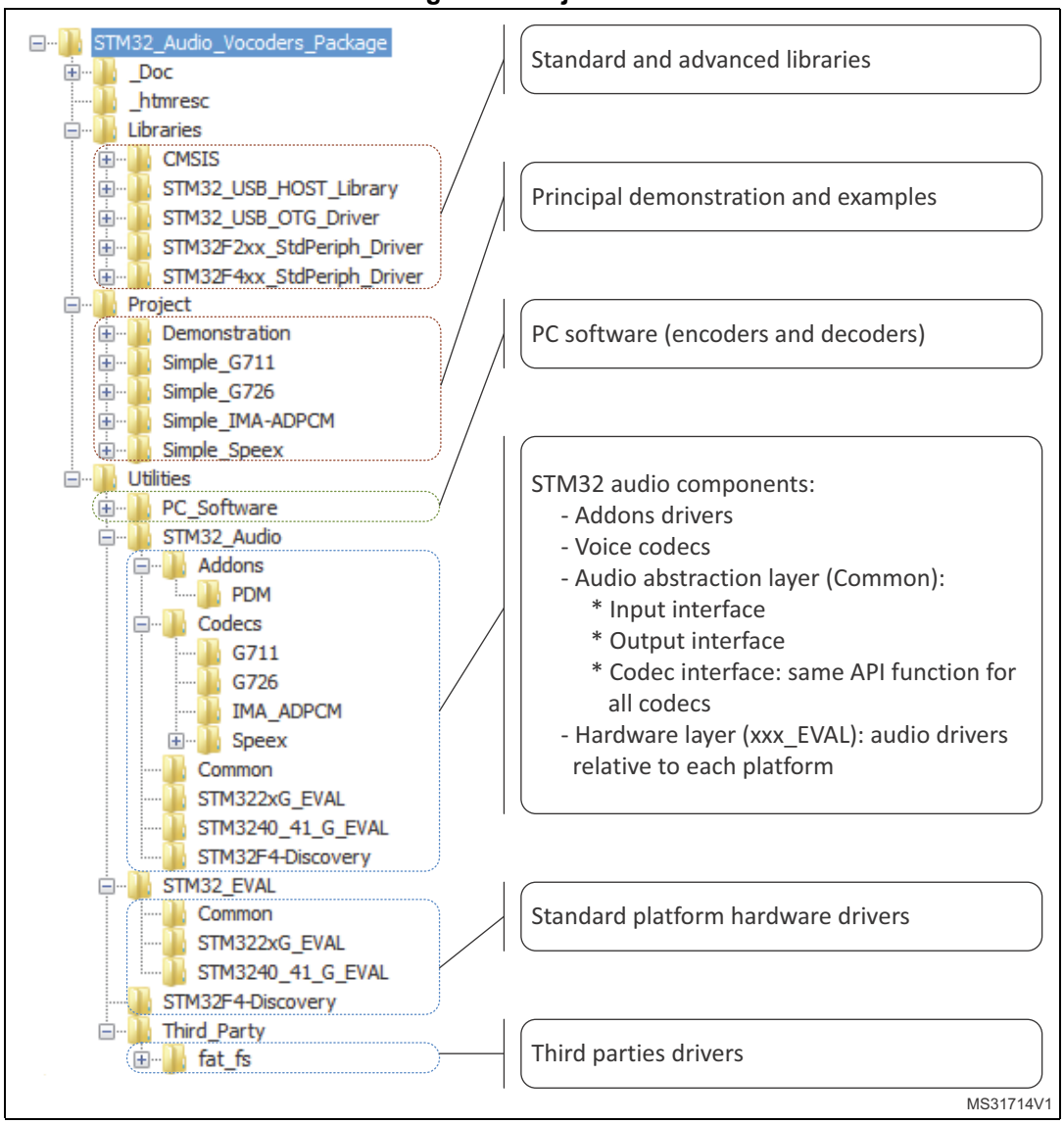
2 Package content and description

2.1 Firmware description

STM32 audio vocoders package includes a set of firmware libraries used to build speech-based audio applications. The firmware is made up of a main application (demonstration) and four simple examples (one for each codec).

The project tree is presented in *Figure 2*.

Figure 2. Project tree



There are multiple abstraction layers for the audio components. The main abstraction layer is located in "STM32_Audio/Common" (manages output interfaces, decoders and mass storage control), while all the encoding/decoding operations are called by the abstraction

layer interfaces. *Figure 3* summarizes the different firmware layers in the vocoders demonstration. *Figure 4* shows the whole demonstration architecture.

Figure 3. Firmware layers

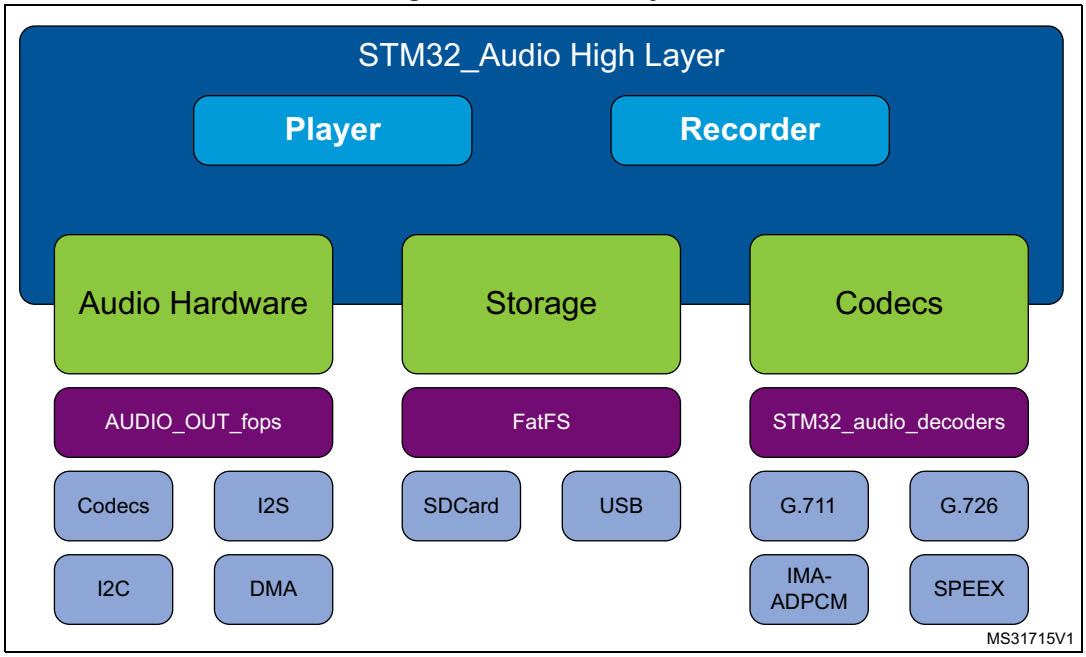
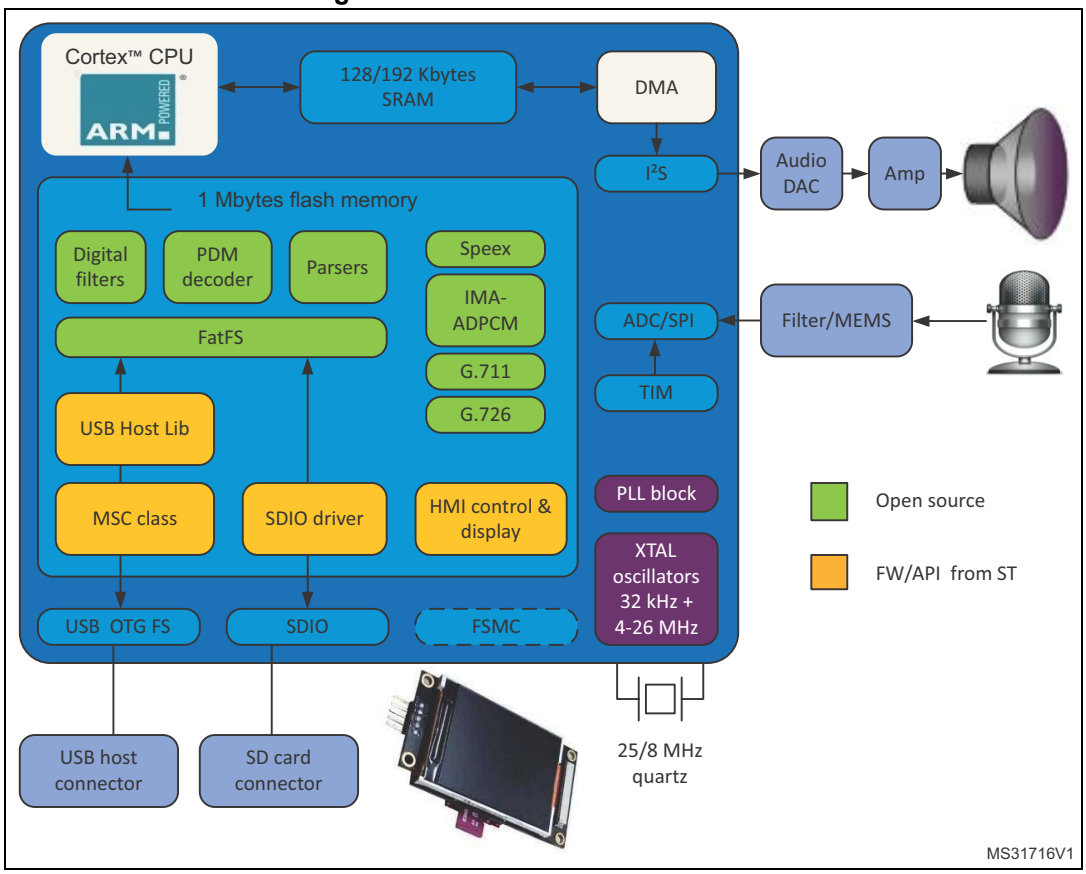


Figure 4. Demonstration architecture



The STM322xG-EVAL and STM324xG-EVAL evaluation boards (referred as “EVAL boards” hereafter) demonstrate the following main features:

- G.711, G.726, IMA-ADPCM and Speex codecs support
- Advanced display-based user interface
- Play and record on SD card and USB
- Playback features: play, stop, next and previous
- File information display (title, format, record/play status)
- Control via joystick (navigation and selection)

and for the STM32F4DISCOVERY board (referred as “DISCOVERY board” hereafter):

- G.711, G.726, IMA-ADPCM and Speex codecs support
- LED-based user interface
- Play and record on USB flash drive only
- Playback features: play, stop, next and previous
- Control via MEMS accelerometer and "user" push button

While simple examples features are:

- Only one codec (amongst G.711, G.726, IMA-ADPCM and Speex) is supported in each example
- Minimalist user interface
- Play and record on internal flash memory
- Simple commands using tamper (play/stop) and key (record/stop) push buttons

2.2 Audio API description

There are multiple abstraction layers for the audio components:

- Decoders and mass storage operation management
- Output interface (codec, I2S,DMA…) management, independently from FatFS operations: it allows managing all output interface without caring about hardware layer
- Input Interface (ADC, MEMS PDM, microphones…): it allows managing all input interfaces without caring about hardware layer.
- High abstraction layer (AudioPlayer_xxx and AudioRecorder_xxx)

All decoding/encoding operations are called by the abstraction layer interfaces (AudioPlayer and AudioRecorder) as well as the read/write functions using FatFS library or equivalent for read/write operations from internal STM32 flash memory. So when the application calls this level of API, it doesn't need to call decoders/encoders or mass storage open/write/read/close functions. *Figure 5* and *Figure 6* explain the structure of the high layer for the audio player and recorder respectively.

Figure 5. High layer audio player API

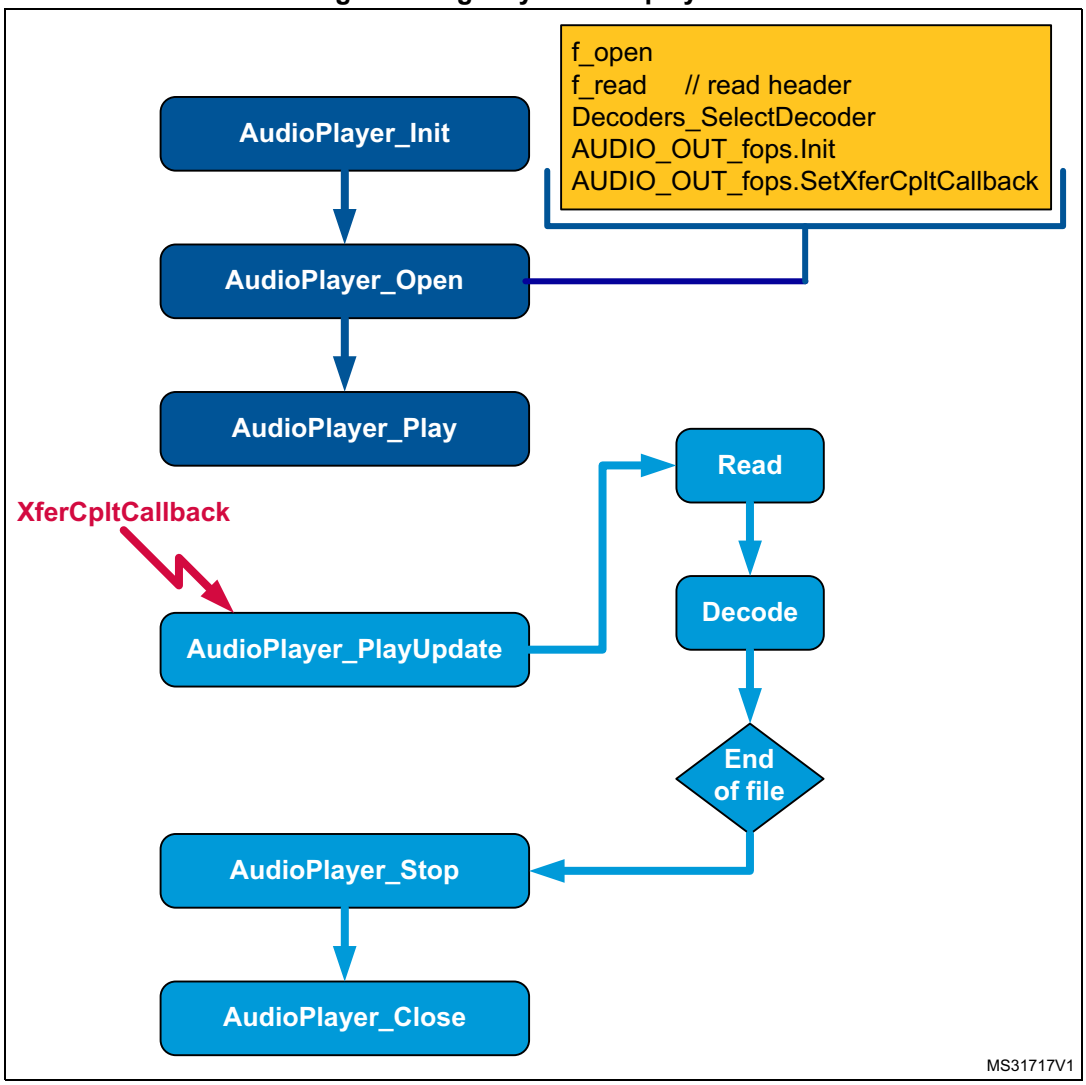
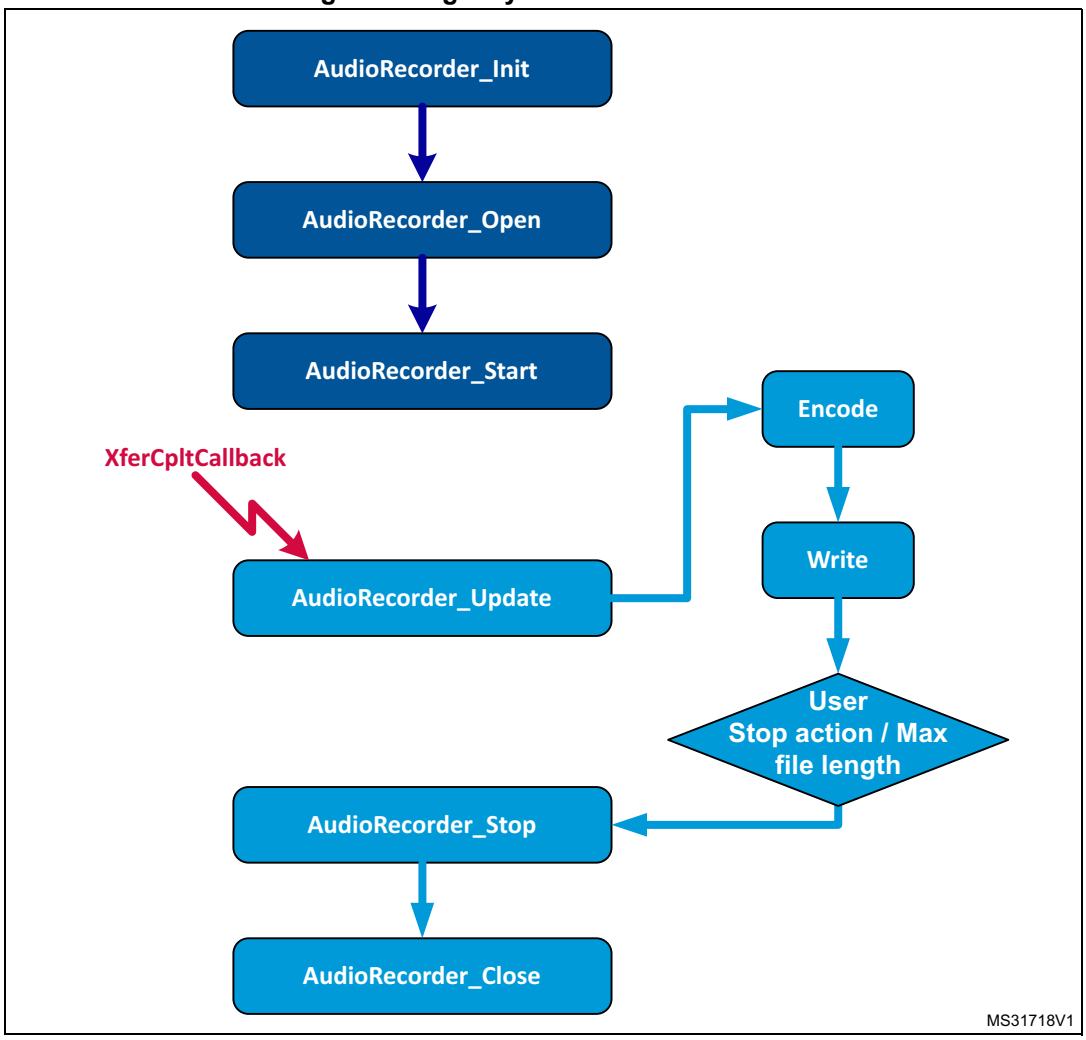


Figure 6. High layer audio recorder API



2.3 Player and recorder processes

2.3.1 Audio player

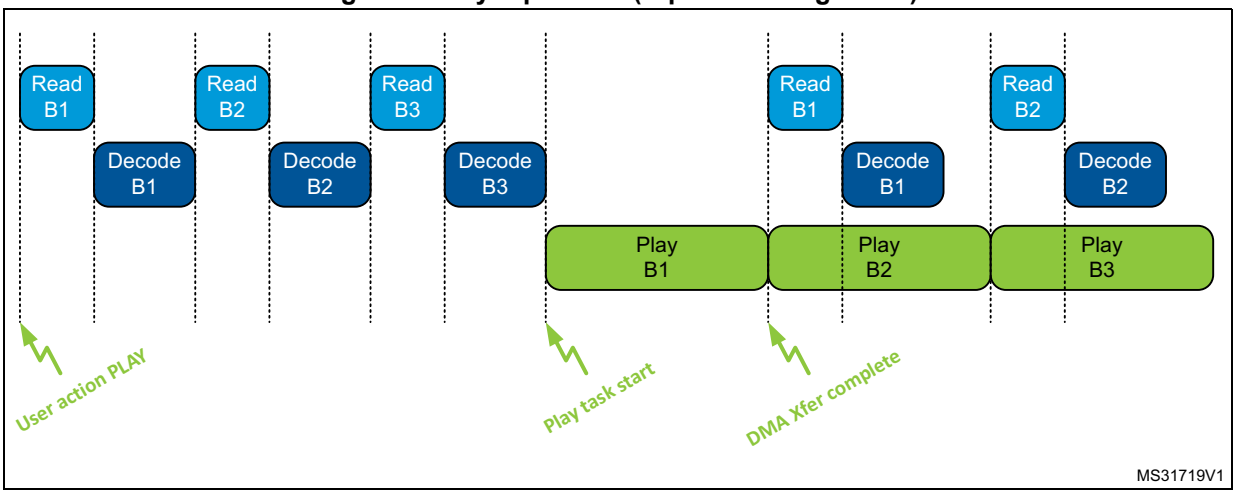
The audio player uses the triple buffering method to perform the following steps:

1. read frames from the mass storage device.
2. decode frames with the selected codec
3. transfer decoded frames to the audio jack through the I2S interface. This transfer is performed by DMA, so CPU is free to perform other tasks while I2S is getting data from decoded buffers.

The DMA transfer complete interrupt informs the player task that a new buffer needs to be read and decoded.

While the DMA outputs a buffer, the application reads and decodes other buffers. *Figure 7* describes this triple-buffering process.

Figure 7. Player process (triple-buffering mode)

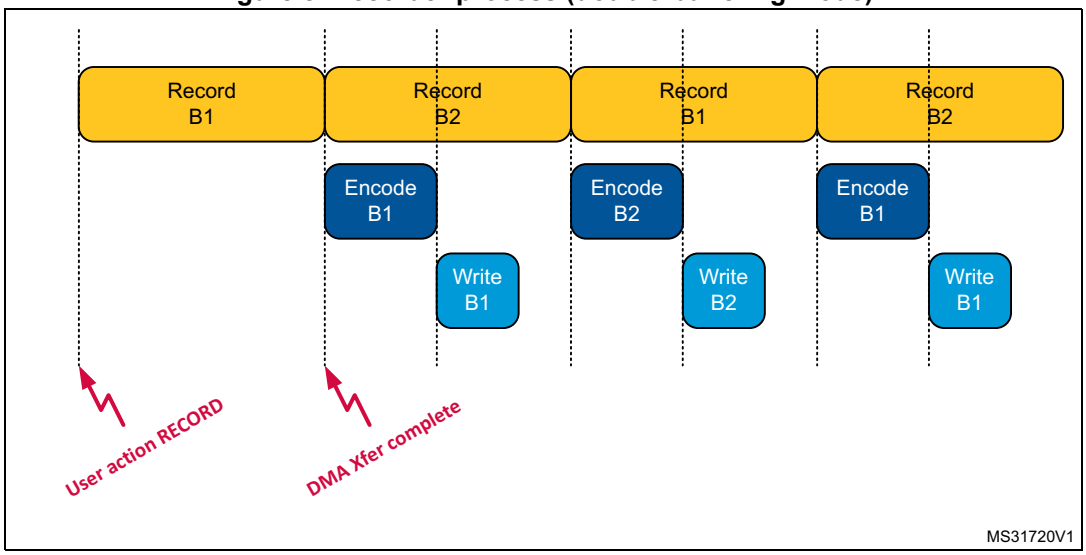


2.3.2 Audio recorder

Audio Recorder Buffering task consists on getting the recorded frame from the ADC peripheral then encode it and write it to mass storage device. ADC conversion is triggered by a timer. Once a buffer is ready, the recorder task is notified and gets the ready buffer for encoding and writing.

As shown in *Figure 8*, double buffering is used for this module.

Figure 8. Recorder process (double-buffering mode)



2.4 Software

In addition to the firmware described above, the vocoders package includes also four software applications (PC applets) that allow the user to encode and decode audio files on the PC with the different voice codecs.

3 How to use the demonstration

3.1 Hardware configuration

Supported boards are: STM32F4DISCOVERY (referred as DISCOVERY board hereafter), STM322xG-EVAL and STM324xG-EVAL (referred as EVAL boards hereafter).

EVAL boards settings for the "demonstration":

- Connect a USB flash drive to the USB OTG FS connector (CN8).
- Connect an SD card to the MSC connector (CN6), also jumper JP16 must be set to position 1-2 (MSD) to be able to use SD card.
- Connect a headset (headphone + microphone) to the audio jack connector (CN11).

EVAL boards settings for the simple examples (such as "Simple_G711"):

- Connect a headset (headphone + microphone) to the audio jack connector (CN11).

DISCOVERY board settings:

- Connect a USB flash drive to the USB OTG FS connector (CN5).
- Connect a headphone to the audio jack connector (CN4); no external microphone is needed as the MEMS integrated microphone (U9) is used.

3.2 Firmware configuration

All the settings that a typical user needs to adjust are regrouped into one single file: *Demonstration\inc\audio_app_conf.h.*

For example PLAYLIST_DIRECTORY and RECORD_DIRECTORY permit to set the directory paths where the player/recorder will read/write audio files. Note that "0:/" corresponds to USB flash drive and "1:/" corresponds to SD card.

In addition to general audio settings, such as volume (DEFAULT_VOLUME) and sampling frequency (SAMPLE_RATE_FREQ), all the codecs parameters can also be set into this file (audio_app_conf.h). There is only one exception: the Speex mode selection (narrowband or wideband) is set directly into the preprocessor defined symbols.

Users should take in consideration the following correspondence table (*Table 7*) when setting Speex parameters:

Table 7. Speex encoded frame size

SPX_ENCODER_QUALITY	ENCODED_FRAME_SIZE	
	Narrowband	Wideband
1	10	15
2	15	20
3	20	25
4	20	32
5	28	42
6	28	52

Table 7. Speex encoded frame size (continued)

SPX_ENCODER_QUALITY	ENCODED_FRAME_SIZE	
	Narrowband	Wideband
7	38	60
8	38	70
9	46	86
10	62	106

Note as well that all the codecs parameters can also be set into the codec process header files:

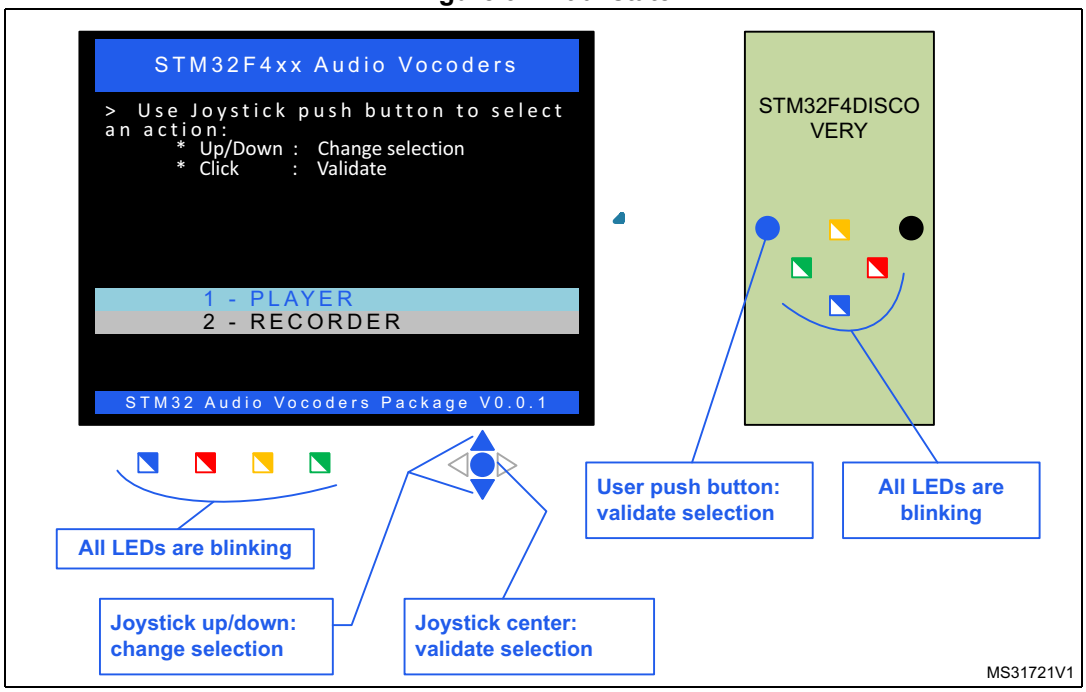
- G711process.h
- G726process.h
- adpcmprocess.h
- speexprocess.h

But if the same parameter is set into anyone of these files as well as in the general setting file (audio_app_conf.h), only this last one is taken into account.

3.3 Main application steps

In the initial state (Main Menu), all LEDs are blinking (*Figure 9*). The user can then select the player or recorder.

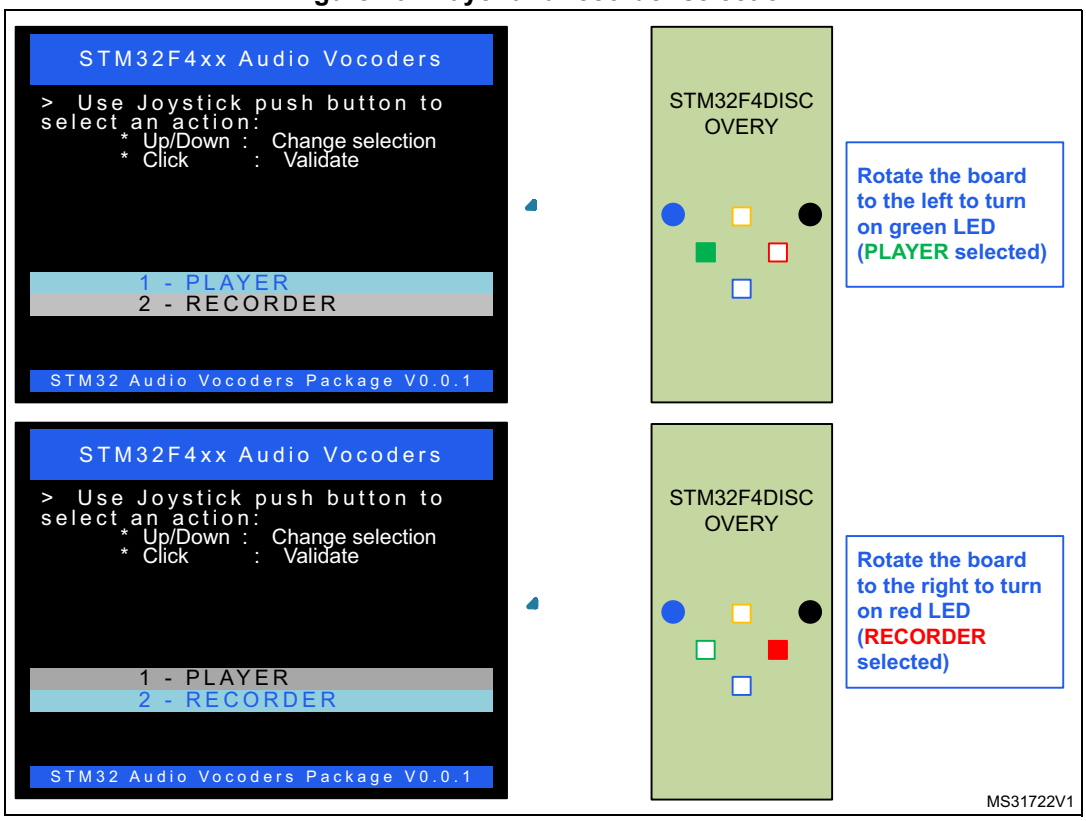
Figure 9. Initial state



On EVAL boards the selected module is highlighted. Use joystick up and down to change selection and Joystick Center to validate the selection and navigate to next menu.

On DISCOVERY board, rotate the board to the left or right to change selection and push user button to validate it. In the main menu, if the green LED is on then the player module is selected while the red LED corresponds to the recorder module (*Figure 10*).

Figure 10. Player and recorder selection



3.3.1 Player menu

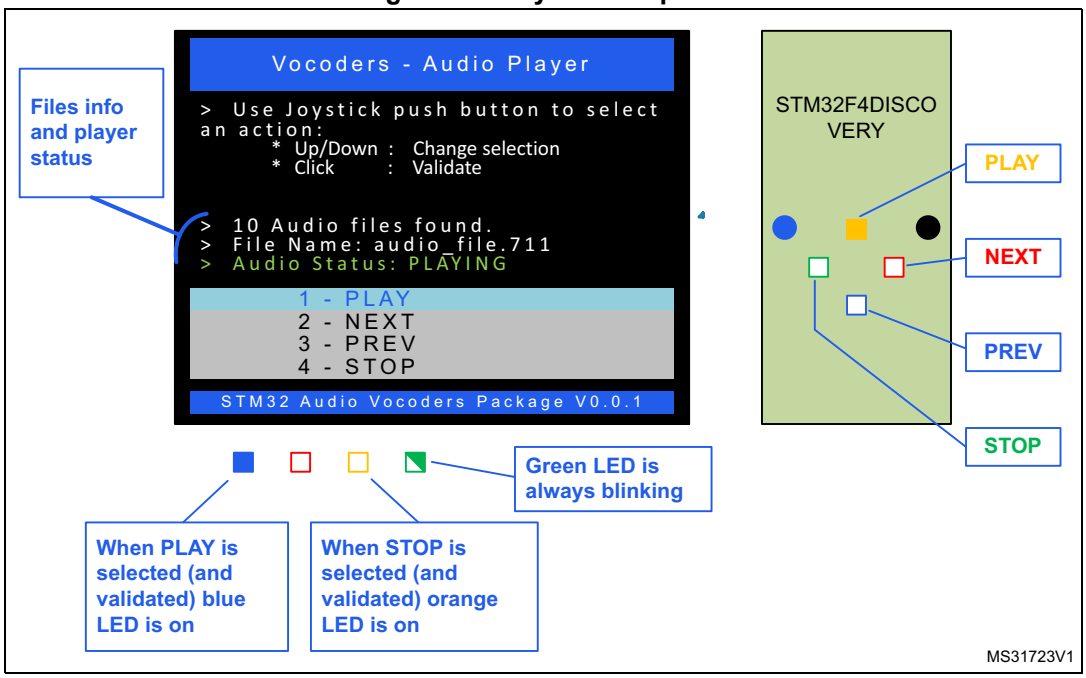
In the audio player menu there are four possible actions: play, next, previous and stop.

On DISCOVERY board, here are the correspondences between LEDs indications and selected actions:

- Orange LED is on: PLAY action selected
- Red LED is on: NEXT action selected
- Blue LED is on: PREV action selected
- Green LED is on: STOP action selected

On EVAL board display the selected action is clearly highlighted together with some useful information: number of supported files found on media storage, currently selected file name, player status. Additionally, the LEDs provide an indication on the audio player status (see *Figure 11*).

Figure 11. Player description

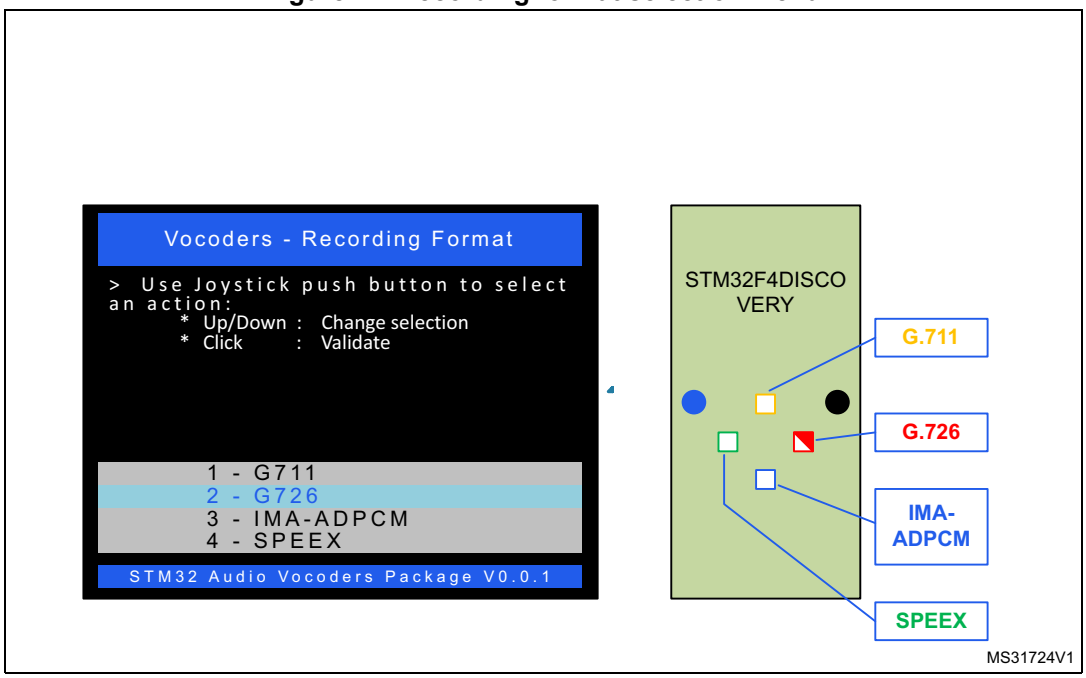


3.3.2 Recording format menu

Before accessing the recorder menu, user should first select a recording format in the intermediate menu (*Figure 12*).

On DISCOVERY board, the LEDs are blinking during this step in order to differentiate the format selection menu from the player menu.

Figure 12. Recording format selection menu



3.3.3 Recorder menu

In the recorder menu (*Figure 13*) there are three possible actions: record, stop and play.

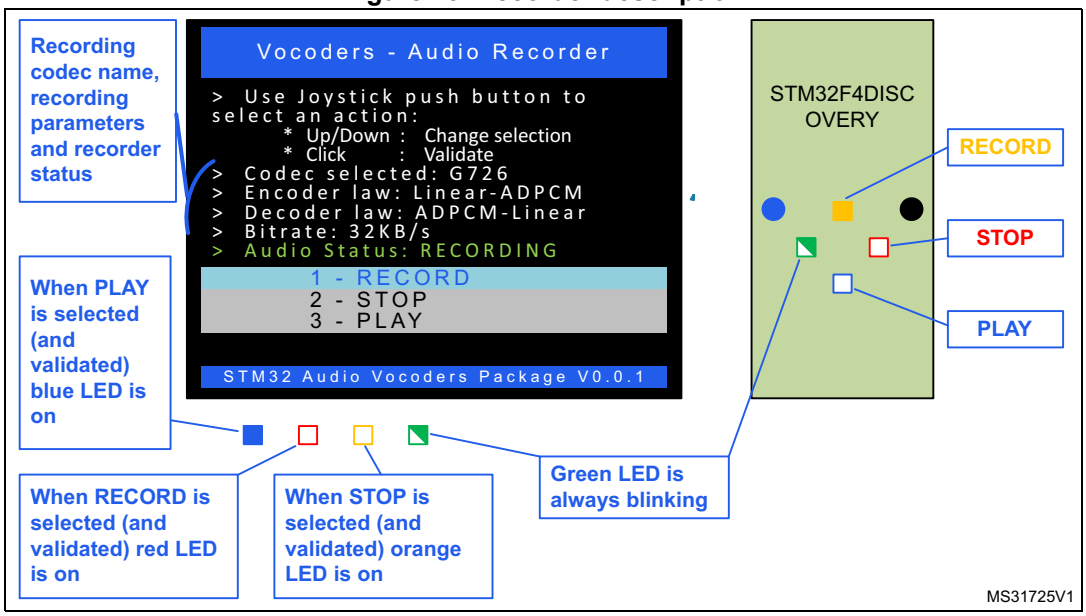
On DISCOVERY board, here are the correspondences between LEDs indications and selected actions:

- Orange LED is on: RECORD action selected
- Red LED is on: STOP action selected
- Blue LED is on: PLAY action selected
- Green LED is always blinking

On EVAL board, the following items are displayed (*Figure 13*):

- actions menu
- recording codec name (selected from previous menu)
- recording parameters
- recorder status.

Figure 13. Recorder description



3.4 Simple examples

Simple examples are provided only on EVAL boards. Encoding/decoding is performed on/from internal flash memory so no external media storage is needed.

There are four simple examples (one for each codec):

- Simple_G711
- Simple_G726
- Simple_IMA-ADPCM
- Simple_Speex

The display is minimalist and there is no menu selection. The control is done by key and tamper push button to respectively record/stop and play/stop.

4 Memory footprint

The footprint results are computed using EWARM v6.40 compiler with high size optimization level.

Table 8. Project footprints

Project	Flash memory (bytes)		SRAM (bytes)
	ro code	ro data	rw data
Demonstration on EVAL board	76 992	18 069	82 602
Demonstration on F4-Discovery	70 056	9 436	83 163
Simple_G711	26 692	9 451	10 660
Simple_G726	23 808	9 677	10 512
Simple_IMA-ADPCM	22 372	9 441	10 160
Simple_Speex	54 584	19 892	75 912

5 FAQ

This section gathers some of the most frequent questions vocoders package users may ask. It gives some solutions and tips.

Table 9. FAQ

No.	Question	Answer/Solution
1	How to select media storage (USB flash drive or SD card)?	In file “audio_app_conf.h”, set PLAYLIST_DIRECTORY and/or RECORD_DIRECTORY to “0:/” to select USB flash drive and “1:/” to select SD card. By default these two defines are set to “0:/USER”.
2	How to configure Codecs parameters?	Almost all codec parameters can be set into “audio_app_conf.h” file. See *Section 3.2* for other alternatives.
3	Is there any parameter to configure when implementing the application in EVAL boards or DISCOVERY board?	All necessary configurations are already done and managed by these three defines: USE_STM322xG_EVAL, USE_STM324xG_EVAL and USE_STM3F4XX_DISCOVERY. As reference, main differences are: – HSE_VALUE (equal to 8 MHz on the DISCOVERY board and 25 MHz on the EVAL board) – LCD support (only on EVAL) – SD card support (only on EVAL) – MEMS accelerometer support (only on DISCOVERY board) – USB power switch port configuration (HOST_POWERSW_PORT_RCC, HOST_POWERSW_PORT and HOST_POWERSW_VBUS)
4	How to add a new codec?	To allow the application to support new codec, the user should: – Add codec routines under “Codecs/” – Add the codec functions (callbacks) layer regrouped into a new file called “xxxprocess.c” (under Common/). User can take “wavprocess.c” as reference. – Update file “Common/stm32_audio_decoders.c” to support new function associations. Normally, the other layers (player, recorder, audio_in and audio_out Interfaces) do not have to be modified.
5	How to disable LCD display on EVAL board?	Un-define (comment) LCD_ENABLE, into “audio_app_conf.h”.

6 References

- Speex: http://www.xiph.org
- ITU-T Recommendation G.711: http://www.itu.int
- ITU-T Recommendation G.726: http://www.itu.int

7 Revision history

Table 10. Document revision history

Date	Revision	Changes
27-May-2014	1	Initial release.