UM1901
User manual

Getting started with the software package for digital MEMS microphones in X-CUBE-MEMSMIC1 expansion for STM32Cube

Introduction

This document describes how to get started with the X-CUBE-MEMSMIC1 software.

X-CUBE-MEMSMIC1 provides the complete STM32 middleware to build applications using digital MEMS microphones. It is easily ported across different MCU families, thanks to STM32Cube. This package contains sample applications for the acquisition of PDM signals from up to four digital MEMS microphones, PDM to PCM conversion and real time streaming of audio data to a PC via a standard USB audio-input driver.

The software provides implementation examples for STM32 Nucleo platforms equipped with the X-NUCLEO-CCA02M1 expansion board, featuring two on-board MEMS microphones (MP34DT01-M) as well as headers to connect additional microphones using ST coupon-based MEMS microphone boards [*4*].

The software is based on STM32Cube technology and expands the STM32Cube-based range of solutions.

Contents

1 STM32Cube

1.1 STM32Cube overview

The STMCube™ initiative was designed by STMicroelectronics to help developers reduce development effort, time and cost. STM32Cube covers the entire STM32 portfolio. Version 1.x includes:

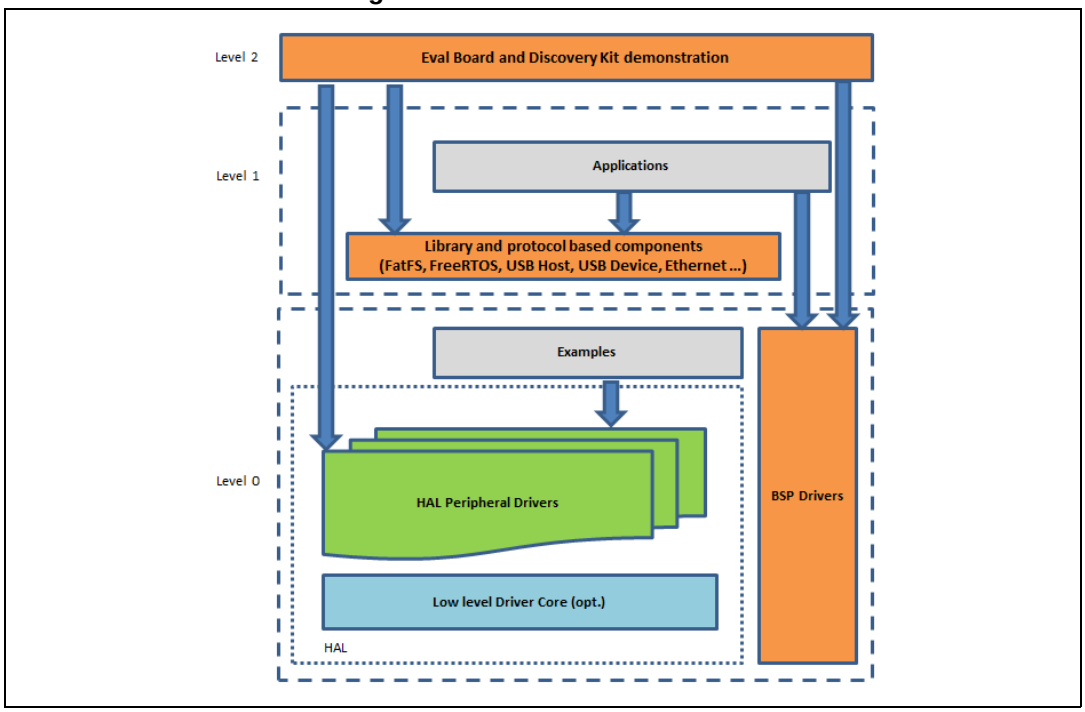
- STM32CubeMX, a GUI for generating C initialization code
- a comprehensive embedded software platform for each series (e.g., STM32CubeF4 for the STM32F4 series):
  - the STM32Cube HAL: STM32 abstraction layer embedded software for maximum portability across the STM32 portfolio
  - a consistent set of middleware components such as RTOS, USB, TCP/IP and graphics
  - all embedded software utilities with a full set of examples

Detailed information regarding STM32Cube is available on st.com at http://www.st.com/stm32cube.

1.2 STM32Cube architecture

The STM32Cube firmware is built around three independent levels that can easily interact with each other, as shown in the figure below.

Figure 1. Firmware architecture



Level 0 is divided into three sub-layers:

- **Board support package (BSP):** offers a set of APIs for the hardware components on the hardware boards (audio codec, IO expander, touchscreen, SRAM driver, LCD drivers, etc.) and consists of two parts:
  - **component**: the driver relative to the external device on the board and not related to the STM32, the component driver provides specific APIs to the BSP driver external components and may be ported to any other board
  - **BSP driver**: links the component driver to a specific board and provides a set of user-friendly APIs named BSP_*FUNCT_Action*() (e.g., BSP_LED_Init(), BSP_LED_On(), etc.)

  It is based on modular architecture, which is easily ported to any hardware by simply implementing the low level routines.
- **Hardware Abstraction Layer (HAL):** this layer provides the low level drivers and the hardware interfacing methods to interact with the upper layers (application, libraries and stacks).
  It provides generic, multi-instance and function-oriented APIs which render user applications unnecessary by providing ready to use processes. For example, for the communication peripherals (I²S, UART, et.), it provides APIs to initialize and configure the peripheral, manage data transfer based on polling, interrupts or DMA processes and manage any communication errors that may raise.
  The HAL Drivers APIs are split into two categories:
  - generic APIs with common and generic functions for the entire STM32 series
  - extension APIs with special functions for a specific family or part number
- **Basic peripheral usage examples:** this layer covers the examples for the STM32 peripheral using the HAL and BSP resources.

Level 1 is divided into two sub-layers:

- **Middleware components:** set of libraries covering USB Host and Device libraries, STemWin, FreeRTOS, FatFS, LwIP and PolarSSL. Horizontal interaction between the components on this layer is performed directly by calling the feature APIs, while the vertical interaction with the low-level drivers is managed through specific callbacks and static macros implemented in the library system call interface.
  For example, the FatFs implements the disk I/O driver to access the microSD drive or the USB Mass Storage Class.
- **Examples based on the middleware components:** each middleware component comes with one or more examples (or ‘applications’) demonstrating how it is used. Integration examples that use several middleware components are also provided.

Level 2 provides comprehensive, real-time and graphical demonstrations of the middleware service layer, the low-level abstraction layer and basic peripheral usage applications for board-based functions.

2 X-CUBE-MEMSMIC1 software expansion for STM32Cube

2.1 Overview

The X-CUBE-MEMSMIC1 software package expands STM32Cube functionality with the following key features:

- comprehensive middleware to build applications using digital MEMS microphones (MP34DT01-M)
- easy portability across different MCU families thanks to STM32Cube
- a customized audio-input class USB driver so the device is recognized as a standard multichannel USB microphone
- PC-based streaming and recording using standard third-party audio editors
- free user-friendly license terms
- example applications available for the X-NUCLEO-CCA02M1 board connected on a NUCLEO-F401RE, NUCLEO- L053R8 or NUCLEO-F072RB board

This software enables the acquisition of up to four digital MEMS microphones through I²S and SPI peripherals and performs PDM to PCM format conversion, the main standard for audio communication and processing.

Exploiting the capabilities of the included audio-input USB driver, the device is recognized as a standard multichannel USB microphone by Windows™ or any Unix-like system; it can perform signal streaming to a host system for data recording and further processing using any standard audio editor, even if any software with a standard USB audio interface can be used to interact with the device.

2.2 Architecture

This software is fully compliant with and expands on the STM32Cube (see *Section 1: STM32Cube*) to enable development of applications using digital MEMS microphones.

The software is based on STM32CubeHAL, the hardware abstraction layer for the STM32 microcontroller. The package extends STM32Cube by providing a board support package (BSP) for the microphone expansion board and middleware components for audio processing and USB communication with a PC.

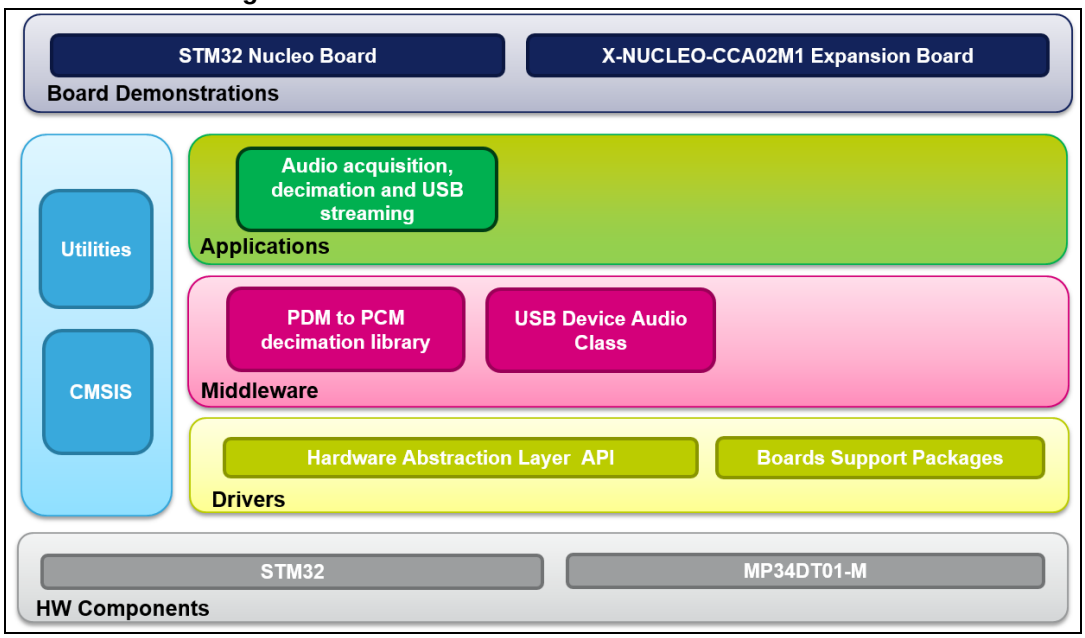
The software layers used by the application software to access and use the microphone expansion board are:

- STM32Cube HAL layer: provides a generic, multi-instance set of APIs to interact with the upper layers (the application, libraries and stacks). It consists of generic and extension APIs based on a common architecture which allows other layers like the middleware layer to function without specific Microcontroller Unit (MCU) hardware configurations.
  This structure improves library code reusability and guarantees easy device portability.
- Board Support Package (BSP) layer: is a set of APIs which provides a programming interface for certain board specific peripherals (LED, user button etc.). This interface also helps in identifying the specific board version and provides support for initializing required MCU peripherals and reading data.

For the microphone expansion board, it provides the interface for MP34DT01-M digital MEMS microphones.

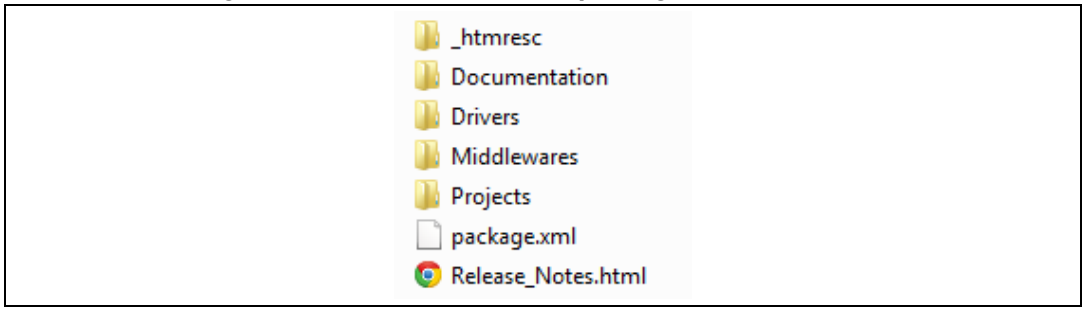
The figure below outlines the software architecture of the package.

Figure 2. X-CUBE-MEMSMIC1 software architecture



2.3 Folder structure

Figure 3. X-CUBE-MEMSMIC1 package folder structure



The following folders are included in the software package:

- 'Documentation': contains a compiled HTML file generated from the source code and detailed documentation of the software components and APIs
- 'Drivers': contains the HAL drivers and the board-specific drivers for supported board and hardware platforms, including those for the on-board components and the CMSIS vendor-independent hardware abstraction layer for the Cortex-M processor series
- 'Middlewares': contains libraries and protocols for the PDM to PCM conversion process and the audio-input USB driver
- 'Projects': contains a sample application for the NUCLEO-F401RE, NUCLEO-L053R8 or NUCLEO-F072RB platforms to access microphone data, with three development

environments (IAR Embedded Workbench for ARM, RealView Microcontroller Development Kit (MDK-ARM) and System Workbench for STM32 (SW4STM32)

2.4 APIs

Detailed descritions of all the functions and parameters of the user APIs user can be found in a compiled HTML file located inside the ‘Documentation’ folder.

2.5 Sample application description

An example application using the X-NUCLEO-CCA02M1 expansion board with NUCLEO-F401RE, NUCLEO-L053R8 or NUCLEO-F072RB boards is provided in the ‘Projects’ directory. Ready to be built projects are available for multiple IDEs.

2.5.1 Hardware-related acquisition processes

This section summarizes the digital MEMS microphone acquisition strategies and principles used in the application. It can simplify the comprehension of the firmware structure and utilization.

A digital MEMS microphone can be acquired through peripherals such as SPI, I²S or GPIO. It requires an input clock and it outputs a PDM stream at the same frequency of the input clock. This PDM stream has to be filtered and decimated in order to be converted in standard PCM format. Two different digital MEMS microphones can be connected on the same data line by configuring the first to generate valid data on the rising edge of the clock and the other on the falling edge. This is achieved by setting the L/R pin of each microphone.

On the X-NUCLEO-CCA02M1 expansion board, two microphones share the same data line and are routed to the Nucleo I²S peripheral (the first and the second microphone) and SPI peripheral (the third and the fourth).

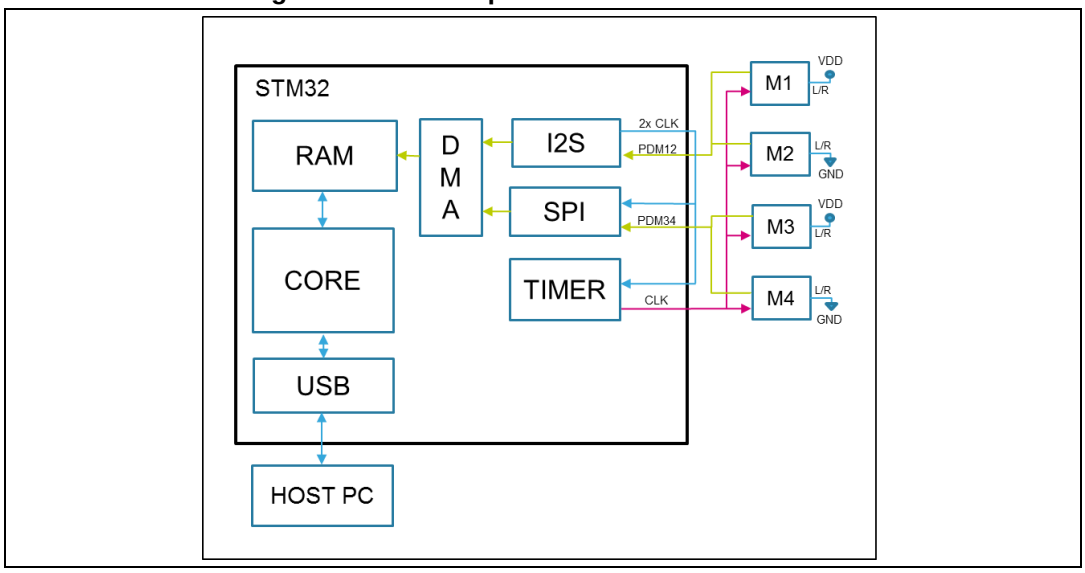
In this scenario, microphone acquisition works thus:

- a precise clock signal is generated by I²S peripheral while SPI is configured in slave mode and is fed by the same timing signal generated by I²S
- this clock is then halved by a timer and input to the microphones
- the SPI and I²S peripherals operate at twice the microphone frequency and can therefore read the data for two microphones each by reading the rising and falling edge of the microphone clock; acquisition is managed by the DMA
- a software demuxing step is required to separate the signal for each microphone to allow further processing like PDM to PCM conversion

Figure 4 shows a simplified diagram of the acquisition process. To acquire a single microphone only, I²S is used and the MEMS microphone is connected directly to its clock and data line. In this case, I²S and the microphone operate at the same frequency.

For further information regarding MEMS microphone and PDM to PCM filtering and decimation, please refer to [*1*] and [*2*].

Figure 4. Audio acquisition hardware structure



2.5.2 Application description

The application example acquires the PDM outputs from the MEMS microphones, converts them into PCM format and streams the resulting signal to a host PC via USB. Synchronized acquisition of 1, 2 or 4 microphones is possible by configuring the solder bridges appropriately.

PDM and PCM data is available in the application before being sent to the USB to facilitate the development and testing of microphone-based audio processing algorithms.

In this application, STM32 I²S and SPI devices used for microphones acquisition are set up depending on the sampling frequency and number of channels. Data is acquired through I²S (up to 2 microphones) or both I²S and SPI peripherals (up to 4 microphones). Moreover USB driver is initialized and the relative descriptor is configured in order to fit the correct channels and sample frequency configuration.

After the configuration phase, the acquisition starts and the device streams the signals to a host device as a standard multichannel USB microphone.

Note: The positions of solder bridges and jumpers must be compliant with the desired microphone configuration; please refer to "Section 3.1.2: X-NUCLEO-CCA02M1 expansion board" for more hardware setup details.

In the firmware, audio-related parts of the application are collected mainly inside the audio_application.c file and usb_audio_if.c, that make use of the dedicated X-NUCLEO-CCA02M1 BSP layer and the PDM to PCM decimation library middleware. In this version, STM32 peripheral setup via the BSP layer is driven by the USB driver to allow total device control from the host PC. A different approach could be used to decouple USB and BSP functions by removing BSP functions from the USB interface file (usb_audio_if.c) and calling them in the application itself.

Follow these steps to set up the system for microphone acquisition and streaming (you can trace them in the application):

- Check the HW configuration for solder bridges and jumpers (see *Section 3.1.2: X-NUCLEO-CCA02M1 expansion board*).
- Initialize USB descriptor using USBD_AUDIO_Init_Microphone_Descriptor(…) according to the number of channels to be streamed.
- Initialize USB core and start USB functionalities with the functions: USBD_Init(…), USBD_RegisterClass(…), USBD_AUDIO_RegisterInterface(…), USBD_Start(…). This allows the device to be recognized as a standard USB microphone with the requested configuration.
- All the required peripherals and middleware (PDM to PCM library) must be configured depending on the number of channels to be streamed and the desired sampling frequency to be achieved; these steps are performed inside the usbd_audio_if.c file, whose functions are called in response to the USB operations that start when the device is connected to the PC (enumeration, dataInput…).
  The following BSP functions are used:
  - BSP Initialization: initialize the hardware peripherals and the PDM library depending on the desired acquisition settings
  - BSP Record: starts PDM acquisition; a double buffer mechanism is implemented and an interrupt is generated every millisecond to allow data processing
  - BSP Stop: stops data acquisition
- The interrupt service routine for audio data acquisition is implemented as a callback in the audio_application.c file (BSP_AUDIO_IN_Transfer_CallBack(…)) in which the PDM to PCM conversion is performed and data is sent to USB. To achieve this, the BSP layer provides a millisecond of PDM data ready to be processed; i.e., demuxed and arranged as required by the decimation library.
  The user can modify this function to add, for example, DSP functions to the audio before sending it to USB.
  Data is sent to USB using the data transfer function. For further information about USB APIs, refer the compiled HTML file located inside the ‘Documentation’ folder.

Alternatively, configuration and initialization of the audio peripherals can be managed independently of the USB flow by directly calling the audio-related BSP functions from the application space. In this case, the user could call the BSP Initialization, Record and Stop functions as needed.

Please note that, depending on the MCU used and the corresponding available resources, not all channels/sampling frequency combinations are allowed. These are the configurations supported by each Nucleo board to date:

- NUCLEO-F401RE: 1, 2 or 4 microphone acquisition and streaming at 16 kHz, 32 kHz or 48 kHz
- NUCLEO-L053R8 or NUCLEO-F072Rb: 1 or 2 microphone acquisition and streaming at 16 kHz or 1 microphone at 32 kHz

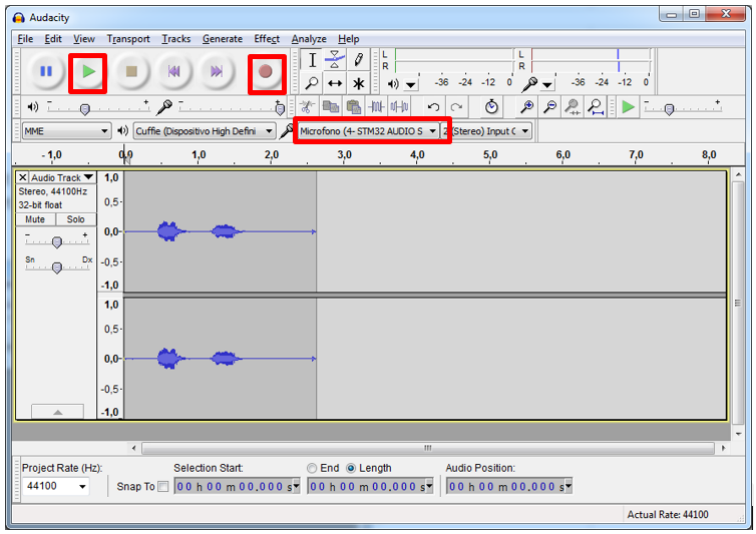
2.6 PC audio recording utility example: audacity

This section describes the use of the Audacity® application for multi-channel audio recording. Audacity® is a free, open source, cross-platform software package for recording and editing sounds and is available for Windows®, Mac®, GNU/Linux® other operating systems as a freeware audio editing environment (http://sourceforge.net/projects/audacity/).

In Windows 7, the released Audacity version is capable of recording up to 2 microphones; the proprietary ASIO driver is usually the best way to perform multi-channel recordings on Windows, but licensing restrictions prevent including ASIO support in released versions of Audacity. However, it can be compiled with ASIO support for private, non-distributable use. For more information, refer to [3].

To start audio recording, ensure that STM32 AUDIO Streaming in FS mode is the selected audio input device and proceed to record and play audio using from the interface shown below.

Figure 5. Audacity for windows



3 System setup guide

3.1 Hardware description

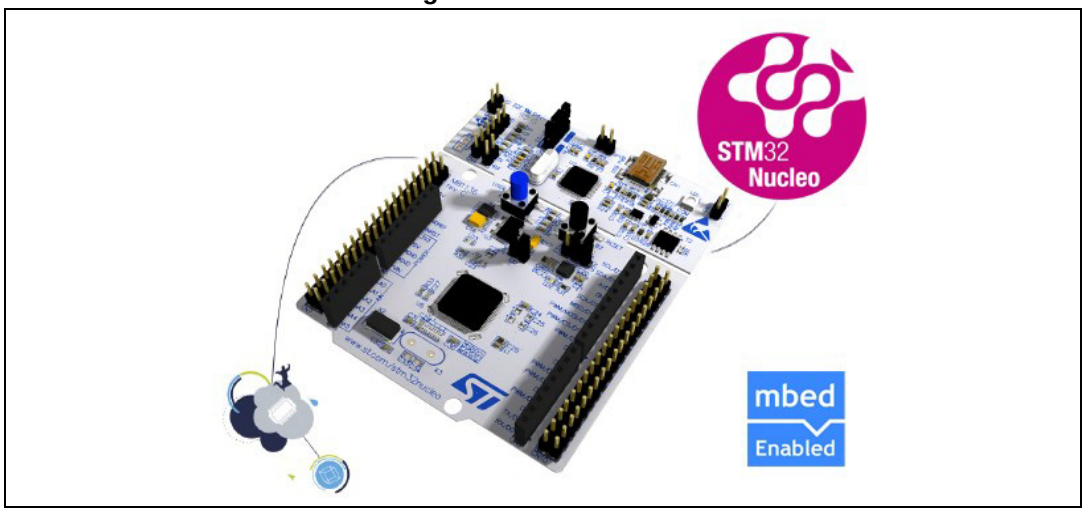
This section describes the hardware components needed for developing applications based on digital MEMS microhones. The following sub-sections describe the individual components.

3.1.1 STM32 Nucleo platform

The STM32 Nucleo boards provide an affordable and flexible way for users to try out new ideas and build prototypes with any STM32 microcontroller lines. The Arduino™ connectivity support and ST Morpho headers make it easy to expand the functionality of the STM32 Nucleo open development platform with a wide choice of specialized expansion boards. The STM32 Nucleo board does not require any separate probes as it integrates the ST-LINK/V2-1 debugger/programmer. The STM32 Nucleo board comes with the STM32 comprehensive software HAL library together with various packaged software examples.

Information about the STM32 Nucleo boards is available on st.com at http://www.st.com/stm32nucleo.

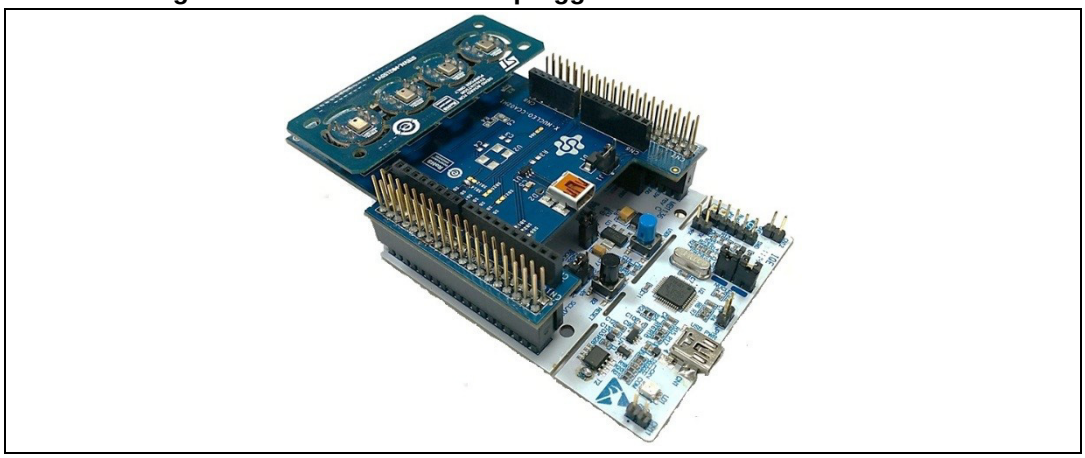
Figure 6. Nucleo board



3.1.2 X-NUCLEO-CCA02M1 expansion board

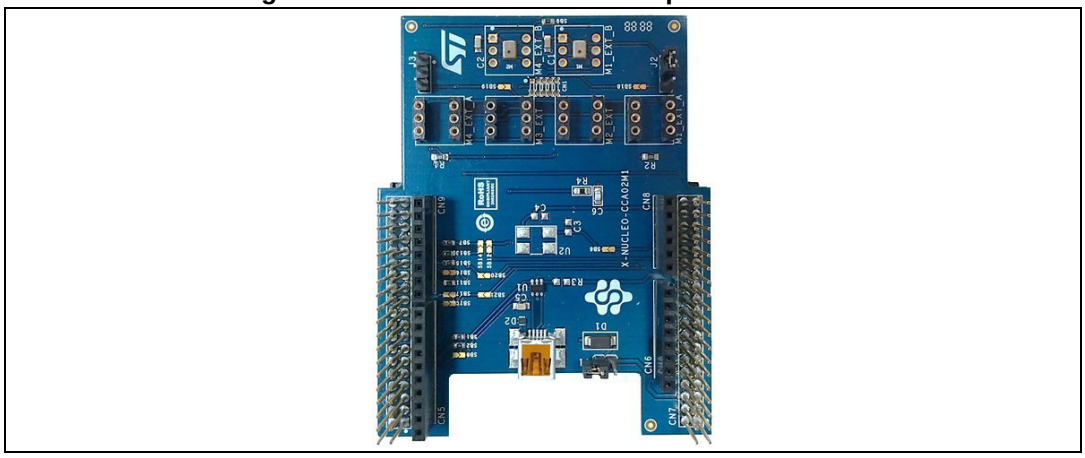
X-NUCLEO-CCA02M1 is an evaluation board based on digital MEMS microphones. It is compatible with the Morpho connector layout and is designed around STMicroelectronics MP34DT01-M digital microphones. It has two microphones soldered on the board and it offers the possibility of connecting additional microphones using MP32DT01-based coupon evaluation boards (STEVAL-MKI129Vx or STEVAL-MKI155Vx), as shows in *Figure 7*.

Figure 7. STEVAL-MKI155V1 plugged on X-NUCLEO-CCA02M1



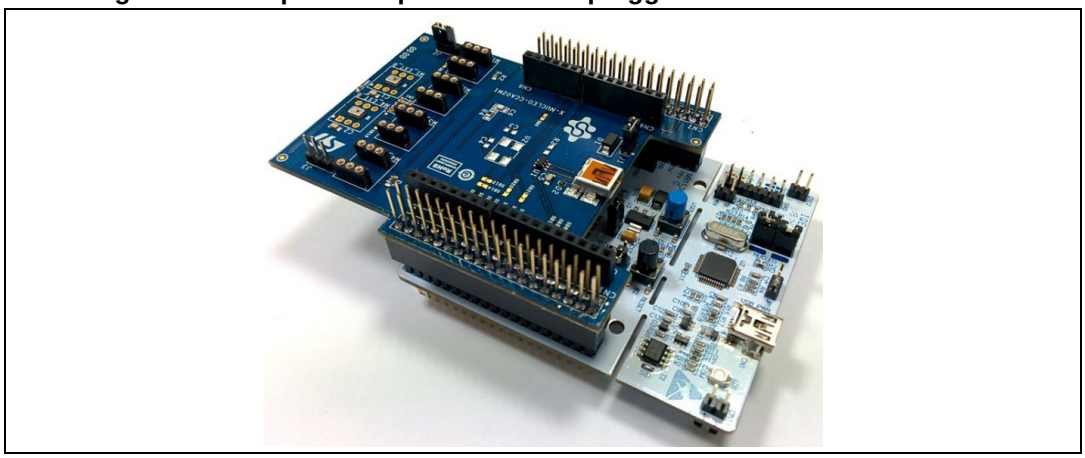
X-NUCLEO-CCA02M1 allows the acquisition and streaming of up to two microphones using the I²S bus and up four coupon microphones using I²S and SPI together. It represents an easy and fast solution for the development of microphone-based applications, as well as a starting point for audio algorithm implementation.

Figure 8. X-NUCLEO-CCA02M1 expansion board



Information about the X-NUCLEO- CCA02M1 expansion board is available on www.st.com at http://www.st.com/x-nucleo.

Figure 9. Microphone expansion board plugged to STM32 Nucleo board



3.2 Software description

The following software components are needed in order to set up the development environment for creating applications for the STM32 Nucleo equipped with a MEMS microphone expansion board:

- X-CUBE-MEMSMIC1: an STM32Cube expansion for audio application development; The X-CUBE-MEMSMIC1 firmware and related documentation is available on st.com
- development tool-chain and compiler: the STM32Cube expansion software supports the following environments:
  - IAR Embedded Workbench for ARM® (EWARM) toolchain + ST-Link
  - RealView Microcontroller Development Kit (MDK-ARM) toolchain + ST-LINK
  - System Workbench for STM32 (SW4STM32) + ST-LINK

3.3 Hardware and software setup

This section describes the hardware and software setup procedures, and the corresponding system setup.

3.3.1 Hardware setup

The following hardware components are needed:

1. an STM32 Nucleo development platform (suggested order code: NUCLEO-F401RE)
2. a microphone expansion board (order code: X-NUCLEO-CCA02M1)
3. a USB type A to Mini-B USB cable to connect the STM32 Nucleo to the PC
4. a USB type A to Mini-B USB cable to connect the X-NUCLEO-CCA02M1 to the PC for USB streaming

3.3.2 External microphone connection

The X-NUCLEO-CCA02M1 expansion board is designed to support up to four external digital microphones based on the ST coupon daughterboard concept (part number: STEVAL-MKI129Vx or STEVAL-MKI155Vx , refer to [4] for further details).

For this purpose, four headers (M1_EXT_A, M2_EXT, M3_EXT, M4 EXT_A) allow operation along a linear array concept. Footprints for additional headers (M1_EXT_B and M4_EXT_B) can be used as an alternative to M1_EXT_A and M4_EXT_A to create a square-shaped microphone array.

Note: M1_EXT_A, M1_EXT_B and M4_EXT_A, M4_EXT_B share the same signals, so be careful not to plug both M4_EXT_A and M4_EXT_B or M1_EXT_A and M1_EXT_B at the same time to avoid possible microphones damage.

3.3.3 Jumper configuration

The X-NUCLEO-CCA02M1 expansion board offers various solutions for microphone acquisition and USB streaming. Depending on your needs, you can choose to acquire on-board microphones or external microphones from one to four channels.

Each configuration requires the correct solder-bridge setup. Different scenarios with corresponding hardware configurations are described below.

Microphone acquisition solutions and relative jumper status

Different acquisition scenarios can be achieved depending on the number of microphones and configuration. JP2 and JP3 are used to choose between external and internal microphones, while solder jumpers are used to connect the right signals to the MCU devices (SPI, Timer, I²S).

Note: For each hardware configuration, the correct firmware initialization parameters must be used.

Acquisition of one microphone

The I²S Peripheral is used directly to give the right clock to the microphone and then to acquire it.

For this scenario, apply the configuration in *Table 1*.

Table 1. Solder bridge configuration for one microphone

SB	Status
SB7	Open
SB8	Open
SB9	Open
SB10	Open
SB11	Close
SB12	Open
SB13	Close
SB14	Close
SB15	Open
SB16	Open
SB17	Open
SB18	Open
SB19	Open
SB20	Open
SB21	Open

In addition, J2 must be placed in position 1-2 for on-board microphone acquisition or 2-3 for an external microphone, while J3 must be left open. If using external microphones, do not plug anything in M2_EXT header.

Acquisition of two microphone

In this scenario, the I²S peripheral is used to generate twice the frequency needed by the microphones (see *Section 2.5.1*). The clock is then halved by the timer and routed to the microphones to give them the right clock. In this way, the I²S reads values from both edges of the merged PDM lines.

For this scenario, apply the configuration in *Table 2*.

Table 2. Solder bridge configuration for two microphones

SB	Status
SB7	Close
SB8	Open
SB9	Open /Close
SB10	Open
SB11	Close
SB12	Open
SB13	Close
SB14	Open
SB15	Close
SB16	Open
SB17	Open
SB18	Open
SB19	Open
SB20	Open
SB21	Open

In addition, J2 must be placed in position 1-2 for on-board microphone acquisition or 2-3 for using external microphones, while J3 must be left open. When acquiring on-board microphones, close SB9 to acquire both of them.

Acquisition of four external microphones

In this case, the I²S peripheral is used to generate a clock frequency at twice the frequency needed by the microphones, and SPI is configured in slave mode in order to use the timing signal from the I²S. As in the previous case, the clock is then halved by the timer and routed to the microphones to give the right clock. I²S and SPI read values from both the edges of the merged PDM lines.

For this scenario, apply the configuration in *Table 3*.

Table 3. Solder bridge configuration for four microphones

SB	Status
SB7	Close
SB8	Close
SB9	Open
SB10	Close
SB11	Close
SB12	Open
SB13	Close
SB14	Open
SB15	Close
SB16	Open
SB17	Open
SB18	Open
SB19	Open
SB20	Open
SB21	Open

In addition, J2 and J3 must be placed in position 2-3 for external microphone acquisition.

Board power supply jumper

X-NUCLEO-CCA02M1 expansion board has an on-board USB connector to power the underlying STM32 Nucleo board. For this purpose, close JP1 on the X-NUCLEO-CCA02M1 expansion board and configure jumper JP5 on the STM32 Nucleo board on the E5V side.

USB connection solder bridges

X-NUCLEO-CCA02M1 USB connector is connected to DM and DP pins of the STM32 Nucleo board through two solder jumpers: SB1 and SB2. These solder bridges must be closed if you want to use USB communication, otherwise you can leave them open.

3.3.4 Software setup

This section lists the minimum requirements for the developer to setup the SDK, run the sample testing scenario based on previous descriptions and customize applications.

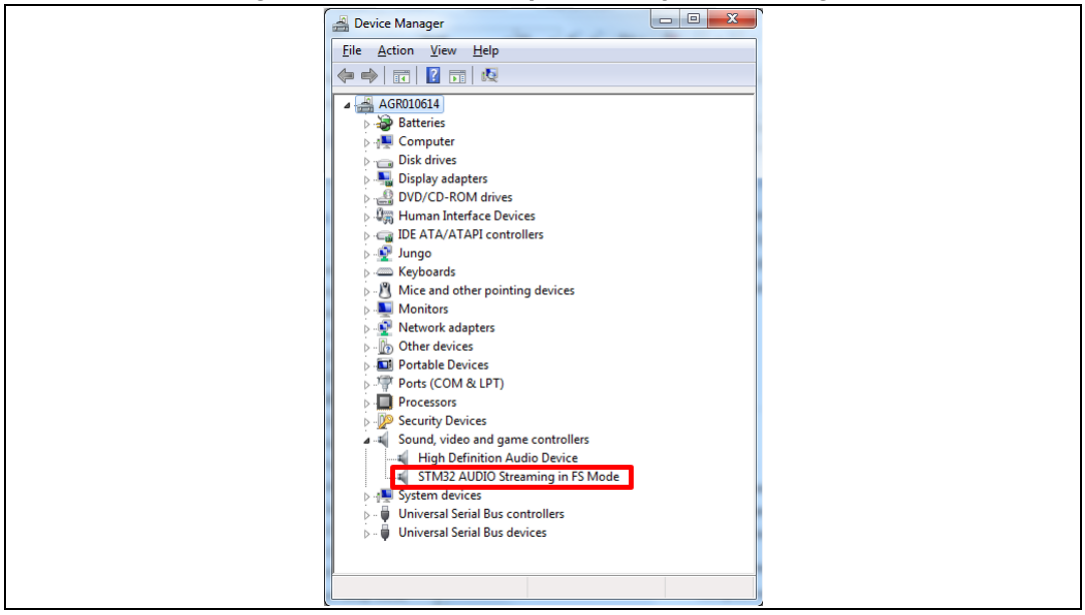
Development tool-chains and compilers

Select one of the integrated development environments supported by the STM32Cube expansion software and read the system requirements and setup information for the selected IDE provider.

Recognition of the device as a standard USB microphone in Windows 7

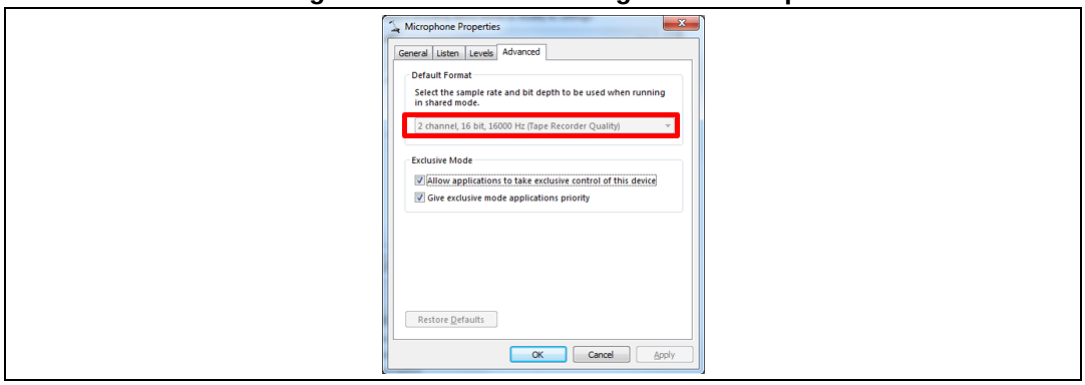
The sample application for audio acquisition and streaming includes an audio input USB driver that allows the device to be recognized as a standard USB microphone. After firmware download to MCU flash, jumper setup and X-NUCLEO-CCA02M1 connection to PC via USB, check the device manager to ensure it has been recognized correctly, as shown in the figure below.

Figure 10. STM32 microphone in system manager



The final step is to right-click on the volume icon in the Windows task bar and choose ‘Recording device’. Now select STM32 microphone and click on ‘Properties’. In the ‘Advanced’ tab, there is a summary of the current device setup in terms of sampling frequency and number of channels. Select the right configuration in order to be able to record an save audio (*Figure 11*).

Figure 11. Advanced configuration setup



3.3.5 System setup guide

This section describes how to setup different hardware components before writing and executing an application on the STM32 Nucleo board with the MEMS microphone expansion board.

STM32 Nucleo and microphone expansion board setup

The STM32 Nucleo board integrates the ST-LINK/V2-1 debugger/programmer. The developer can download the relevant version of the ST-LINK/V2-1 USB driver by searching STSW-LINK008 or STSW-LINK009 on www.st.com.

The microphones expansion board X-NUCLEO-CCA02M1 can be easily connected to the STM32 Nucleo motherboard through the Morpho extension connector, see *Figure 9*. The microphone expansion board is capable of interfacing with the external STM32 microcontroller on STM32 Nucleo using I²C, I²S and USB.

4 Acronyms and abbreviations

Table 4. Acronyms and abbreviations

Acronym	Description
PDM	Pulse density modulation
PCM	Pulse code modulation
USB	Universal serial bus
SPI	Serial peripheral interface
I²S	Integrated interchip sound
BSP	Board support package
HAL	Hardware abstraction layer
IDE	Integrated development environment

5 References

1. MP34DT01-M MEMS audio sensor omnidirectional digital microphone, data sheet.
2. PDM audio software decoding on STM32 microcontrollers, Application note AN3998.
3. Audacity ASIO Audio interface wiki.
4. STEVAL-MKI129Vx, STEVAL-MKI155Vx: Microphone coupon board based on the MP34DT01 digital MEMS microphone, Data brief.

6 Revision history

Table 5. Document revision history

Date	Revision	Changes
09-June-2015	1	Initial release.