



AN00248: Using lib_xua with lib_mic_array

Publication Date: 2025/11/13

Document Number: XM-012763-AN v1.0.1

IN THIS DOCUMENT

1	Introduction	1
2	Example application	1

1 Introduction

The XMOS USB Audio library [lib_xua](#) provides an implementation of USB Audio Class versions 1.0 and 2.0.

This application note demonstrates the implementation of a basic USB Audio Device with record functionality from PDM microphones on the [XK-EVK-XU316](#) board.

Core PDM microphone functionality is contained in [lib_mic_array](#). This library includes both the physical interfacing to the PDM microphones as well as efficient decimation to user configurable output sample rates - essentially providing PDM to PCM conversion.

I2S is also enabled as an output and the on board DAC is configured to play out the mic signals.

2 Example application

2.1 The CMakeLists.txt file

Add **lib_xua** to the dependent module list in the CMakeLists.txt file. Also add **lib_board_support** which contains code to setup the audio hardware:

```
set(APP_DEPENDENT_MODULES "lib_xua",
    "lib_board_support(1.1.1))
```

Compiler flags need to be set to tell **lib_board_support** which board is being used. For example:

```
set(APP_COMPILER_FLAGS = -DBOARD_SUPPORT_BOARD=XK_EVK_XU316
    ..)
```

2.2 Declarations

Allocating hardware resources for lib_xua

This implementation of a USB Audio device using **lib_xua** requires the following I/O pins:

- ▶ Audio Master clock (generated by the XCORE)
- ▶ Bit clock (generated by the XCORE)
- ▶ Word clock (generated by the XCORE)
- ▶ I2S data output pin (generated by the XCORE)



On an XCORE the pins are controlled by **ports**. The main application therefore declares a port for the master clock input signal. The **main()** function that is provided within **lib_xua/src/core/main.xc** itself has these ports defined. Look for **p_i2s_dac**, **p_lrclk**, **p_bclk** and **p_mclk_in** in **lib_xua/src/core/main.xc**.

lib_xua also requires two ports for internally calculating USB feedback. Please refer to the **lib_xua** library documentation for further details. In this example **XUA_Buffer()** and **XUA_AudioHub()** reside on the same tile and can therefore make use of the same master-clock port. These are defined as **p_for_mclk_count** and **p_mclk_in_usb** in the **main()** function in **lib_xua**.

In addition to **port** resources, some clock-block resources (**clk_audio_bclk**, **clk_audio_mclk** and **clk_audio_mclk_usb**), also defined in the **main()** function in **lib_xua**, are required.

Allocating hardware resources for lib_mic_array

In this example **lib_mic_array** requires a single 1-bit port for PDM data from two microphones. Each microphone is configured to produce a PDM sample on an opposite clock edge from the other. This means the the data is effectively double data rate (DDR) with respect to the PDM clock.

The microphones must be clocked by a clock synchronous to the audio application clock - typically 3.072MHz for 16, 32 or 48 kHz.

These ports, along with other **mic_array** required resources must be declared in **mic_array_conf.h**:

```
#define XUA_NUM_PDM_MICS          2
#define MIC_ARRAY_CONFIG_PORT_MCLK  PORT_MCLK_IN
#define MIC_ARRAY_CONFIG_PORT_PDM_CLK  PORT_PDM_CLK
#define MIC_ARRAY_CONFIG_PORT_PDM_DATA  PORT_PDM_DATA
#define MIC_ARRAY_CONFIG_CLOCK_BLOCK_A  XST_CLKBLK_1
#define MIC_ARRAY_CONFIG_CLOCK_BLOCK_B  XST_CLKBLK_2  /* Second clock needed as we use DDR */
```

The **XK-EVK-XU316** Board expects the XCORE to divide down the audio master clock (24.576MHz) and output the result to the microphones. This is done internally by **lib_mic_array** from the definitions **MIC_ARRAY_CONFIG_MCLK_FREQ** and **MIC_ARRAY_CONFIG_PDM_FREQ** which are left as defaults in **mic_array_conf.h** to generate the nominal 3.072 MHz PDM clock.

Please see the **lib_mic_array** library documentation for full details.

2.3 Configuring lib_xua

lib_xua must be configured to enable support for PDM microphones.

lib_xua has many parameters than can be configured at build time, some examples include:

- Supported sample-rates

Note

lib_mic_array does not currently support sample rate change after initialisation

- Channel counts
- Audio Class version
- Product/Vendor IDs
- Various product strings
- Master clock frequency



To enable PDM microphone support **XUA_NUM_PDM_MICS** must be set to a non-zero value. Setting this will cause the **XUA_AudioHub** task to forward sample rate information and receive samples from the relevant microphone related tasks.

These parameters are set via defines in the **xua_conf.h** header file. For this simple application the complete contents of this file are as follows:

```
// Copyright 2017-2025 XMOS LIMITED.
// This Software is subject to the terms of the XMOS Public Licence: Version 1.

#ifndef _XUA_CONF_H_
#define _XUA_CONF_H_

#include <platform.h>

#define NUM_USB_CHAN_OUT      0          /* Number of channels from host to device */
#define NUM_USB_CHAN_IN      2          /* Number of channels from device to host */
#define I2S_CHANS_DAC        2          /* Number of I2S channels out of xCORE */
#define I2S_CHANS_ADC        0          /* Number of I2S channels in to xCORE */
#define MCLK_441             (512 * 44100) /* 44.1kHz family master clock frequency */
#define MCLK_48              (512 * 48000) /* 48kHz family master clock frequency */
#define XUA_PDM_MIC_FREQ     48000      /* Currently sample rate changes are not supported for
↳ PDM mics */
#define MIN_FREQ              XUA_PDM_MIC_FREQ /* Minimum sample rate */
#define MAX_FREQ              XUA_PDM_MIC_FREQ /* Maximum sample rate */

#define PORT_MCLK_IN_USB      XS1_PORT_1D
#define PORT_MCLK_COUNT      XS1_PORT_16B
#define PORT_I2S_DAC0        PORT_I2S_DAC_DATA

/* Since the PORT defines are done manually above, not in the XN file, lib_xua must be told
 * which tiles the ports are located */
#define XUA_AUDIO_IO_TILE_NUM 1

#define XUA_NUM_PDM_MICS      2
#define MIC_ARRAY_CONFIG_PORT_MCLK PORT_MCLK_IN
#define MIC_ARRAY_CONFIG_PORT_PDM_CLK PORT_PDM_CLK
#define MIC_ARRAY_CONFIG_PORT_PDM_DATA PORT_PDM_DATA
#define MIC_ARRAY_CONFIG_CLOCK_BLOCK_A XS1_CLKBLK_1
#define MIC_ARRAY_CONFIG_CLOCK_BLOCK_B XS1_CLKBLK_2 /* Second clock needed as we use DDR */

#define AUDIO_CLASS            1
#define VENDOR_STR             "XMOS"
#define VENDOR_ID              0x20B1
#define PRODUCT_STR_A2         "XUA PDM Example"
#define PRODUCT_STR_A1         "XUA PDM Example"
#define PID_AUDIO_1            1
#define PID_AUDIO_2            2
#define XUA_DFU_EN              0 /* Disable DFU for simplicity of example */

#endif
```

XUA_PDM_MIC_FREQ can be changed to 32000 or 16000 to reconfigure the system to lower sample rates than the default 48 kHz.

2.4 The application **main()** function

The **main()** function is provided within the **lib_xua/src/core/main.xc** file. It starts all of the tasks in parallel using the **xC par** construct.

Firstly the standard **lib_xua** USB side tasks are run on tile 0. This code starts the low-level USB task and an Endpoint 0 task. The Audio buffering task and a task to handle the audio I/O (**XUA_AudioHub**) is started on tile 1 where the I2S bus exists.

The microphone task **mic_array_task** spawns a single thread which handles PDM receive on the ports and the decimation filters to produce PCM. This is placed on tile 1 where the microphone hardware is connected. It connects directly to **XUA_AudioHub** and provides samples which are at the same rate as the audio I/O.

User callbacks

While the **main()** function is provided within **lib_xua** itself, there are a number of callbacks which allow for customisation of the application.

The DAC is configured by **AudioHwInit()** function in **hwsupport.xc**, which calls Audio hardware initialisation functions for the relevant hardware from **lib_board_support**



```

#include "xua.h"
#include "xk_evk_xu316/board.h"

void AudioHwInit()
{
    xk_evk_xu316_config_t hw_config = {MCLK_48};
    xk_evk_xu316_AudioHwInit(hw_config);

    const int samFreq = 48000; /* xk_evk_xu316_AudioHwConfig doesn't like rates below 22kHz so force to 48k
    ↪ which works OK */
    xk_evk_xu316_AudioHwConfig(samFreq, MCLK_48, 0, 32, 32);

    return;
}

```

Callback functions `UserBufferManagement()` and `user_pdm_process()` are defined in `user_callbacks.xc`. Both these functions are called from the main loop in `XUA_AudioHub()`. `user_pdm_process()` implements the PCM sample post processing, which for this example is a simple scaling by a factor of 64 to allow the mic captured audio to be heard easily. `UserBufferManagement()` routes the mic samples to the DAC.

```

// Copyright 2017-2025 XMOS LIMITED.
// This Software is subject to the terms of the XMOS Public Licence: Version 1.

#include <xs1.h>
#include <platform.h>
#include <string.h>

#include "xua.h"
#include "xud_device.h"

/* Copy mic samples inbound to the host to the DAC so they can be monitored there */
#pragma unsafe arrays
void UserBufferManagement(unsigned sampsFromUsbToAudio[], unsigned sampsFromAudioToUsb[])
{
    for(int i = 0; i < I2S_CHANS_DAC; i++)
    {
        sampsFromUsbToAudio[i] = sampsFromAudioToUsb[i];
    }
}

/* Apply some gain so we can hear the mics easily (non-saturating - will overflow) */
#pragma unsafe arrays
void user_pdm_process(int32_t mic_audio[MIC_ARRAY_CONFIG_MIC_COUNT])
{
    for(int i = 0; i < XUA_NUM_PDM_MICS; i++)
    {
        mic_audio[i] = mic_audio[i] << 6; /* x64 */
    }
}

```



2.5 Hardware setup

Required hardware

- ▶ 1x **XK-EVK-XU316** board
- ▶ 2x Micro USB cable (For USB and XTAG)
- ▶ 3.5mm audio cable to connect to the line out connector on the **XK-EVK-XU316** board
- ▶ Host computer

Setup procedure

1. Connect a pair of PDM microphones to the microphone connector on the **XK-EVK-XU316** via a ribbon cable. Refer to the [xcore.ai Explorer hardware manual](#) for more details.
2. Connect one Micro USB cable between the host computer and the **XK-EVK-XU316** **DEBUG** port.
3. Connect the other Micro USB cable between the host computer and the **XK-EVK-XU316** **USB** port.

Fig. 1 shows the **XK-EVK-XU316** board with all the cables connected.

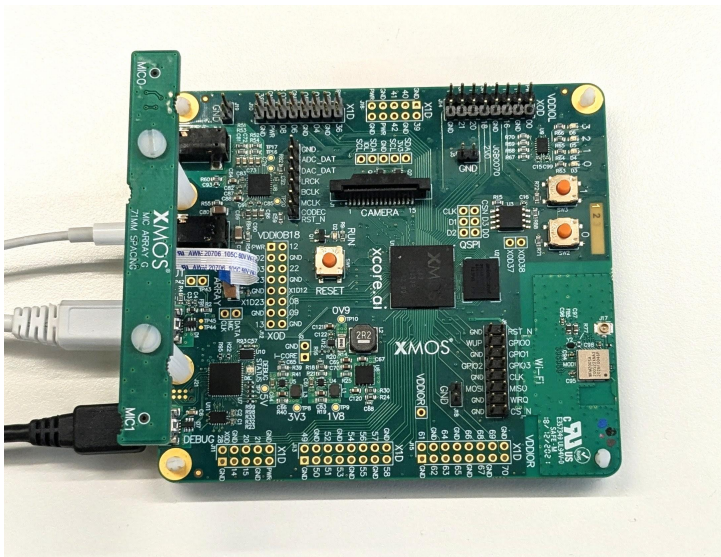


Fig. 1: Hardware setup



2.6 Building the example

This section assumes that the [XMOSE XTC Tools](#) have been downloaded and installed. The required version is specified in the accompanying **README**.

Installation instructions can be found [here](#).

Special attention should be paid to the section on [Installation of Required Third-Party Tools](#).

The application is built using the [xcommon-cmake](#) build system, which is provided with the XTC tools and is based on [CMake](#).

The **an00248** software ZIP package should be downloaded and extracted to a chosen working directory.

To configure the build, the following commands should be run from an XTC command prompt:

```
cd an00248
cd app_an00248
cmake -G "Unix Makefiles" -B build
```

All required dependencies are included in the software package. If any dependencies are missing, they will be retrieved automatically during this step.

The application binaries should then be built using **xmake**:

```
xmake -j -C build
```

Binary artifacts (.xe files) will be generated under the appropriate subdirectories of the **app_an00248/bin** directory – one for each supported build configuration.

For subsequent builds, the **cmake** step may be omitted. If **CMakeLists.txt** or other build files are modified, **cmake** will be re-run automatically by **xmake** as needed.

The application uses approximately 49 kB on tile 0 and 17 kB on tile 1 (of 512 kB on each).

2.7 Running the example

From an XTC command prompt, the following command should be run from the **an00248/app_an00248** directory:

```
xrun ./bin/app_an00248.xe
```

Alternatively, the application can be programmed into flash memory for standalone execution:

```
xf1ash ./bin/app_an00248.xe
```

A USB audio device called **XUA PDM Example** should now be seen enumerated and the stereo microphone audio input stream can be recorded on the USB host or listened to by connecting to the analog output jacks.



Copyright © 2025, All Rights Reserved.

XMOSE Ltd. is the owner or licensee of this design, code, or Information (collectively, the "Information") and is providing it to you "AS IS" with no warranty of any kind, express or implied and shall have no liability in relation to its use. XMOSE Ltd makes no representation that the Information, or any particular implementation thereof, is or will be free from any claims of infringement and again, shall have no liability in relation to any such claims.

XMOSE, XCORE, VocalFusion and the XMOSE logo are registered trademarks of XMOSE Ltd. in the United Kingdom and other countries and may not be used without written permission. Company and product names mentioned in this document are the trademarks or registered trademarks of their respective owners.

