AN02016: Integrating Audio Weaver (AWE) Core into USB Audio

Publication Date: 2024/8/2
Document Number: XM-015091-AN v1.0.0

IN THIS DOCUMENT

1 Introduction

Audio Weaver (AWE) comprises GUI tools (Designer) and libraries (Core) for implementing audio Digital Signal Processing (DSP) algorithms. Developed by DSP Concepts (DSPC), it delivers signal processing building blocks, referred to as “modules”. Module capabilties range from simple filtering to data type conversions all the way to much more specialised processing. These can be assembled, deleted and rearranged in the Designer GUI and then executed on a device. A control library is available that enables on-line control of the blocks.

xcore.ai is a programmable multi-core device by XMOS, with flexible DSP and IO interfaces. The xcore.ai port of Audio Weaver Core is located in lib_awe.

2 Application Examples

The sample application provided alongside this application note is called `app_usb_audio_awe` and has multiple build configurations supporting different features. It is based on the XMOS USB Audio reference design which uses `lib_xua` and associated XK-AUDIO-316-MC hardware. It is very closely related to the standard USB Audio reference design provided by XMOS in `sw_usb_audio`. Documentation for this can be found in the USB Audio User Guide.

The thread diagram for the application example is shown in Fig. 1. Note how most of the design is either brought in from `lib_xua` or `lib_awe` with only the platform specific `lib_i2c` remote thread being part of the application. This is required for configuration of the audio hardware (i.e. ADCs and DACs) on the board.

DSP Concepts provide a helpful setup guide which can be found in the file `User_Guide_for_XMOS_EVK_with_AWE.pdf` provided in the `/doc` directory of the AN02016 download. This is designed to help you get up and running as quickly as possible and help you connect to the Audio Weaver Designer software. A sample design called `playBasic_3thread.awj` for use in the Audio Weaver Designer software may be found in the `examples/audioweaver` directory of the dependent lib_awe library.

The example application provide three build configurations, each one providing a different audio source/sink or tuning data path. These are descibed in Table 1.

Table 1: Example Application Builds

Build	Data path	Tuning path
UA	USB Audio to target, Line out from target	USB / HID and via firmware
UA_FFS	USB Audio to target, Line out from target	USB / HID and via firmware with FFS enabled
I2S	Line in to target, Line out from target	USB / HID and via firmware

The audio path is handled by the USB Audio callback to `UserBufferManagement(unsigned sampsFromUsbToAudio[], unsigned sampsFromAudioToUsb[])` which is called from the USB Audio Audio/I^{2}S thread.

```
void UserBufferManagement(unsigned sampsFromUsbToAudio[], unsigned
↪sampsFromAudioToUsb[])
{
#if(I2S_ONLY)
    /* Intercept samples from ADC (destined for the host) and process
↪them.
     * Send processed samples to the DAC
     */
    awe_offload_data_to_dsp_engine(g_c_to_dspc, sampsFromAudioToUsb,
↪sampsFromUsbToAudio);
#else
    /* Intercept samples from the host (destined for the DAC) and
↪process them.
     * Send processed samples to the DAC
     */
    awe_offload_data_to_dsp_engine(g_c_to_dspc, sampsFromUsbToAudio,
↪sampsFromUsbToAudio);
#endif // I2S_ONLY
} // end_marker_for_rst_literal_include
```

As per the comments in the above code snippet, samples are modified on their way from either the host (UA build) or an external ADC (I^{2}S build) before being passed on to their destination, an external DAC.

The control path in the example application is handled by both the `lib_awe` communications task in `awe_tuning_usb_hid.c` and the local control task in `awe_standalone_tuning.c`.

2.1 USB Audio (UA) Build

The feature set of this build configuration is as follows:

- USB Audio Class 2.0 (High Speed)
- Stereo input to DSP from the host
- Stereo output from DSP on the `OUT 1/2` 3.5mm analog jack
- Audio from the host is pumped through the AWE framework before being played on the output jack
- Asynchronous clocking (local audio clock to hardware)
- 24 bit Sample resolution
- 48 kHz sample rate

- Tuning to AWE provided over USB HID with VID 0x20b1 and PID 0x0018 supporting live tuning from the Audio Weaver software. Additionally, AWB files may be loaded (and in one case controlled) using the buttons on the board.

Note: Any commercial designs based on this app note should use a valid VID/PID pair as supplied by the USB IF. The XMOS VID should not be used in any circumstance.

The thread diagram for the UA application example is shown in Fig. 1. In addition to the I^{2}C remote master a new application thread has been added `awe_standalone_tuning` which handles loading of the AWB image and volume control via the buttons. The volume control provides an example of tuning the DSP blocks from firmware.

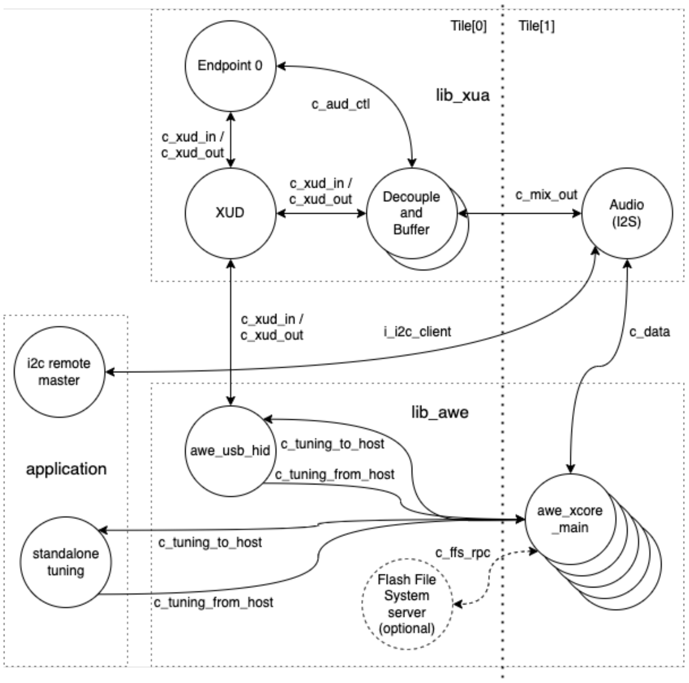

Fig. 1: Application thread diagram for `app_usb_audio_awe`

The button control is decribed in Table 2.

Table 2: `UA_STANDALONE` control functions

Button	Function
2	Load the `PlayBasic_3thread` AWB file which contains the multi-band compressor example
1	Load the `simple_volume` AWB file which contains a pass-through with volume control
0	When the `simple_volume` AWB is selected, it controls the volume in 10 dB decrements. No function for other designs.

Note: When the firmware boots, there is no design loaded so you will not hear any sound played from the host. Please either load a design via Audio Weaver Designer or press button 2 or 1 to load an AWB and enable audio processing.

2.2 USB Audio with Flash File System (UA_FFS) Build

The `UA_FFS` build configuration is a superset of the `UA` configuration. In addition, it has the internal Flash File System (FFS) enabled. This means, via the Audio Weaver GUI, you may add files to a file system that is stored in external flash memory. The files may include `.awb` compiled design images which can be loaded or even booted from so that the AWE system comes up pre-configured with a particular design.

The FFS is stored in the data partition of the flash memory and the boot partition (used for the boot image(s)) is protected from accidental overwriting.

An additional thread is used on Tile[0] which acts as the FFS server and handles accesses to the external QSPI Flash via requests from AWE core.

For more details on using the FFS from Audio Weaver please consult the DSP Concepts documentation.

Note: When using the FFS ensure that the timeout setting in the AWE Server "Change Connection" dialogue is increased to 5000 ms. This is because some of the low-level flash operations may exceed the 1500 ms default timeout setting which will cause communications errors.

The thread diagram in Fig. 1 depicts the addition of the optional FFS flash server thread which manages the low-level flash accesses.

2.3 I^{2}S Build

This build configuration uses I^{2}S for the source of audio data, rather than USB. The feature set of this build configuration is as follows:

- Stereo input from the IN 1/2 3.5 mm analog jack (line level)
- Stereo output on the OUT 1/2 3.5 mm analog jack (line level)
- Audio from the host is pumped through the AWE framework before being played on the output jack
- Tuning to AWE provided over USB HID with VID `0x20b1` and PID `0x0018` supporting live tuning from the Audio Weaver software
- DFU is available via USB

Note: When the firmware boots, there is no design loaded so you will not hear any sound played from the host. Please load an AWB from the host using the Audio Weaver software.

2.4 Building the Examples

The following section assumes you have downloaded and installed the XMOS XTC Tools. See the README for required version.

The application uses the `xcommon-cmake` build system as bundled with the XTC Tools. This requires CMake to be installed.

You will first need to download the `an02016` sofware zip-file and unzip it to a chosen directory. Next, open an XTC command prompt.

To configure the build using xcommon-cmake:

```
cd an02016
cd app_usb_audio_awe
cmake -G "Unix Makefiles" -B build
```

All required dependencies are included in the software download, however, if any are missing it is at this point that will be downloaded by the build system.

Ensure you have the libAWECore.a file placed in the `lib_awe/lib/xs3a` directory. This is the core archive file containing the AWE library:

```
cp libAWECore.a lib_awe/lib_awe/lib/xs3a
```

Note: The `libAWECore.a` file is not provided as part of the lib_awe repository for commercial reasons. This should be obtained from your XMOS or DSPC contact directly.

Finally, build the application binaries using `xmake`:

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

This will build both the `UA` (USB Audio), `I2S` (I^2S only for data transport but with USB/HID enabled for control) and `UA_FFS` binaries.

The application uses approximately 50-64 kB of RAM on Tile[0], depending on build configuration, and 504 kB on Tile[1] when allocating a 50k long-words for `AWE_HEAP_SIZE_LONG_WORDS`. Each tile of an xcore.ai device has 512kB of RAM.

Note: It is possible to trade-off the number of AWE modules available versus the available heap size. This is beyond the scope of this application note - please contact your XMOS or DSPC support contact for details.

2.5 Running the Examples

To run the application use the following command from the lib_awe/app_usb_audio_awe directory where <build> should be one of `UA`, `I2S` or `UA_FFS`:

```
xrun bin/<build>/app_usb_audio_awe_<build>.xe
```

Alternatively to make the design non-volatile by programming in to flash memory use the following command:

```
# UA, I2S
xflash bin/<build>/app_usb_audio_awe_<build>.xe
```

```
# UA_FFS
xflash --factory bin/UA_FFS/app_usb_audio_awe_UA_FFS.xe --boot-
↪partition-size 0x80000 --data ../audioweaver/awb_files/data_
↪partition_ffs.bin
```

The figure `0x80000` equates to 512 kB which is the amount of space reserved for the boot partition. For this example, the required storage in flash for the application is in the order of 348 kB leaving around 164 kB of space for the application to grow if needed. A simple way to determine the required boot partition size if to run the following command and then inspect the file size of `flash.bin`:

```
xflash -o flash.bin app_usb_audio_awe/bin/UA_FFS/app_usb_audio_awe_UA_
↪FFS.xe
```

In this case the rest of the flash beyond the boot partition (for this target 3.5 MB) is available for the FFS.

Once flashed or run, the USB audio device should appear in your host OS's audio settings window (except for the I^{2}S build configuration).

Note: No audio will be passed through from the host to the 3.5 mm jack until an AWE design is loaded.

For designs which are tuned via USB/HID you may connect the Audio Weaver designer software via USB/HID according to the documentation in the `User_Guide_for_XMOS_EVK_with_AWE.pdf` file which is supplied alongside this application note.