Application Note: AN01027

Porting the XMOS USB 2.0 Audio Reference Software onto XU208 custom hardware

This application note shows you how to port the XMOS USB 2.0 audio reference software onto XU208 based custom hardware.

Required tools and libraries

- xTIMEcomposer Tools - Version 14.1.0 or newer
- USB Audio 2.0 Device Software - *Version 6.12.6 or newer*[1]

Required hardware

This application note describes how to to port the XMOS USB 2.0 audio device software to a board using the XU208 device.

Prerequisites

- This document assumes familiarity with the XMOS xCORE architecture, the XMOS tool chain and the xC language. Documentation related to these aspects which are not specific to this application note are linked to in the references appendix.
- For descriptions of XMOS related terms found in this document please see the *XMOS glossary*[2].
- For an overview of XMOS USB Audio 2.0 Device Software please see the *USB Audio Design Guide*[3] for reference.

[1] https://www.xmos.com/support/boards?product=18334&component=16275
[2] http://www.xmos.com/published/glossary
[3] https://www.xmos.com/support/boards?product=18334&component=14442

1 Overview

The XMOS USB 2.0 Audio Reference Software can be easily ported to to xCORE-200 XU208 devices. This application note details the steps required to adapt the USB Audio 2.0 Device Software to custom hardware that uses a XU208 device.

1.1 Port mapping of the custom XU208 board

When designing a custom XU208 based platform, use the following port mapping rules:

Ports	Port mapping rules
I2S ports including: MCLK (master clock), SDOs (serial data outputs), SDIs (serial data inputs), BCLK (bit clock) and LRCLK (word clock)	Assign to 1-bit ports.
I2C ports	Assign to either 1-bit or 4-bit ports. If I2C signals are assigned to a 4-bit port, the rest of the port bits should remain unused.
GPIO ports with low data rates including: MCLK_SEL (master clock 24.576MHz/22.768MHz selection), LEDs, buttons and any other general purpose I/O ports.	Assign to either 1-bit or multi-bit ports. Recommendation: Assign the GPIOs to multi-bit ports to reserve 1-bit ports for high speed signals that require 1-bit port assignment.

1.2 Software environment setup

The following steps describe how to setup the software environment:

1. Download the *XMOS Audio 2.0 Device Software - source code*[4] for reference.
2. Launch xTIMExompose version 14.1.0 or later and create a new workspace.
3. Select **File ▶ Import...**
4. Select **General ▶ Exisiting Projects into Workspace** and then click **Next**.
5. Select **Select archive file.**
6. Browse and select the downloaded archive, in step 1. Click **Open** and then click **Finish**.

The imported project can now be seen in the *Project Explorer* window in xTIMEcomposer.

[4] https://www.xmos.com/support/boards?product=18334&component=16275

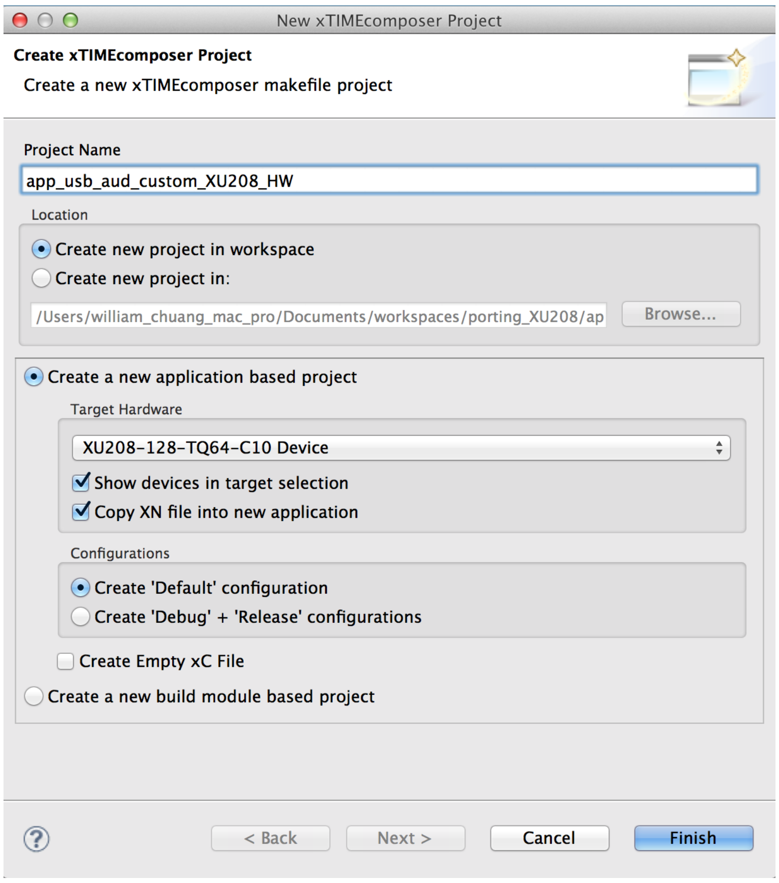
2 Porting the firmware to the XU208 custom hardware

The following sections describe how to port the USB Audio 2.0 Device Software to a XU208 custom hardware.

2.1 Create a project for the custom XU208 hardware

The first step is to create a new project in the workspace.

1. Select **File ▶ New ▶ xTIMEcomposer Project**.
2. In the **New xTIMEcomposer Project** dialog (see Figure 1), fill in the project name in the **Project Name** field (in this application note, it is named as “app_usb_aud_custom_XU208_HW”).
3. Select **Target Hardware** as “XU208-128-TQ64-C10 Device”, if this device is not shown there, be sure to check **Show device in target hardware** checkbox. Note : If the target hardware uses a different variant of the XU208 then ensure to select the appropriate device.
4. Check **Copy XN file into new application** checkbox.
5. Uncheck **Create Empty xC File** checkbox.
6. Click **Finish** button to create a new project.


New xTIMEcomposer Project

Create xTIMEcomposer Project

Create a new xTIMEcomposer makefile project

Project Name: app_usb_aud_custom_XU208_HW

Location
- [✓ Create new project in workspace]
- Create new project in: /Users/william_chuang_mac_pro/Documents/workspaces/porting_XU208/ap

[✓ Create a new application based project]

Target Hardware: XU208-128-TQ64-C10 Device
- [✓ Show devices in target selection]
- [✓ Copy XN file into new application]

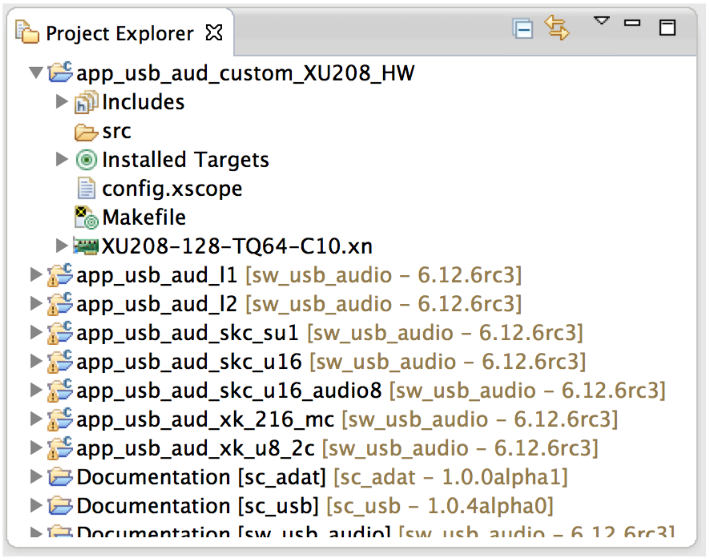
Configurations
- [✓ Create 'Default' configuration]
- Create 'Debug' + 'Release' configurations

- Create Empty xC File

Create a new build module based project

Figure 1: Example of creating a new project

The newly created project will be seen in the *Project Explorer* as shown in Figure 2.



```
▼ app_usb_aud_custom_XU208_HW
   ▶ Includes
     src
   ▶ Installed Targets
     config.xscope
     Makefile
   ▶ XU208-128-TQ64-C10.xn
▶ app_usb_aud_l1 [sw_usb_audio – 6.12.6rc3]
▶ app_usb_aud_l2 [sw_usb_audio – 6.12.6rc3]
▶ app_usb_aud_skc_su1 [sw_usb_audio – 6.12.6rc3]
▶ app_usb_aud_skc_u16 [sw_usb_audio – 6.12.6rc3]
▶ app_usb_aud_skc_u16_audio8 [sw_usb_audio – 6.12.6rc3]
▶ app_usb_aud_xk_216_mc [sw_usb_audio – 6.12.6rc3]
▶ app_usb_aud_xk_u8_2c [sw_usb_audio – 6.12.6rc3]
▶ Documentation [sc_adat] [sc_adat – 1.0.0alpha1]
▶ Documentation [sc_usb] [sc_usb – 1.0.4alpha0]
```

Figure 2: Created project view in *Project Explorer*

2.2 Modify the XN file to fit the project to the custom hardware

A custom XN file is required to make the software fit the XU208 hardware design:

1. [Optional] Rename the XN file from "XU208-128-TQ64-C10.xn" to the required XN file name by right-click on the file name in *Project Explorer* and select *Rename* on the pop-up dialog (In this application note, it is renamed to "custom_XU208").
2. Open the XN file - *app_usb_aud_xk_216_mc\src\core\xk-audio-216-mc.xn* and copy all of the file content.
3. Open the XN file - *app_usb_aud_custom_XU208_HW\custom_XU208.xn* and select-all-paste the file content. After this step, the file content of this XN file will be the same as the *app_usb_aud_xk_216_mc application*.
4. [Optional] Rename the board name in the target XN file.
5. Modify "tileref tile[2]" to "tileref tile[1]", under *Declarations* in the target XN file.
6. Modify the package type from "XS2-UnA-512-FB236" to "XS2-UnA-128-TQ64", under *Packages>Package* in the target XN file.
7. Modify the node type from "XS2-L16A-512" to "XS2-L8A-128", under *Packages>Package>Nodes* in the target XN file.
8. [Optional] modify the node oscillator frequency to meet you custom hardware system clock frequency.
9. Delete all content of the tile 1 element.
10. Add the USB port mapping to the segment of the tile 0 as shown in the example XN file in Appendix §D
11. [Optional] Modify the Flash type in the segment of *ExternalDevices* in the target XN file to meet the Flash device type employed in the custom hardware design.
12. Reassign, add or remove the port mappings in the segment of the tile 0 in the target XN file to meet the port mapping in the custom hardware design.

The full code of the example of a modified XN file is listed in Appendix §D.

2.3 Modify the Makefile to fit to the custom project

The next step is to create a custom *Makefile* for the project.

1. Open the makefile - *app_usb_aud_custom_XU208_HW\Makefile*.
2. Modify the *TARGET* from "XU208-128-TQ64-C10" to the custom XN file name (in this application note, it is "custom_XU208").
3. Add and Modify the content of the *BUILD_FLAGS*, you can copy the content of the *BUILD_FLAGS* from the makefile - *app_usb_aud_xk_216_mc\Makefile* and (a) modify the definition of *-DUSB_TITLE=tile[1]* to *-DUSB_TITLE=tile[0]*, and (b) add the define *-DU208_ONLY* to active the S/W modification described later.
4. Modify the content of the "USED_MODULES" to be the same as the content in *app_usb_aud_xk_216_mc\Makefile*.
5. You can either add the XC compiler flags in *XCC_FLAGS* or optionally modify the pre-processor definitions in the customdefines.h later. If you choose to add the pre-processor flags in this *XCC_FLAGS* segment, then you can check the example Makefile in Appendix §D.
6. Add a new definition - "MODULE_LIBRARIES=xud_x200" in the target makefile.
7. Modify the make paths according to the example makefile in Appendix §D.

2.4 Port the software

This section describes the tasks required to port the software.

- First, create and modify configuration defines in customdefines.h.
- Second, implement the custom code for the audio hardware i.e. for initialization and/or handling audio stream format change.
- Third, migrate the software to the single-tile XU208 device.

The following steps show how to create and modify the custom definition file for the pre-processor definition employed in the USB Audio 2.0 Device Software.

1. Right-click on the *app_usb_aud_xk_216_mc\src\core\customdefines.h* file in *Project Explorer* and select *Copy* on the pop-up menu.
2. Right-click on the *app_usb_aud_custom_XU208_HW\src* in *Project Explorer* and select *Paste* on the pop-up menu.
3. Right-click on the *app_usb_aud_xk_216_mc\src\extensions* folder in *Project Explorer* and select *Copy* on the pop-up menu.
4. Right-click on the *app_usb_aud_custom_XU208_HW\src* in *Project Explorer* and select *Paste* on the pop-up menu.
5. Open *app_usb_aud_custom_XU208_HW\src\customdefines.h* and modify the definitions inside it to fit the project. The definition of *XUD_TILE*, *SPDIF_TX_TILE* and *MIDI_TILE* should be modified to 0 if these are not defined in *Makefile*. Also set *SPDIF_TX_INDEX (0)* to duplicate DAC channels 1/2.

The example customdefines.h is shown in Appendix §D for the reference.

The software functions for the custom audio hardware initialization and audio stream format change handling are shown in Appendix §D. There are two functions that should be implemented for the specific custom audio hardware. One is AudioHWInit which performs the audio hardware initialization during system boot and the other is AudioHwConfig which performs audio hardware reconfiguration when the audio stream sample rate changes. The audio hardware initialization and reconfiguration for the external PLL, ADC and DAC are implemented in the functions AuiodHWInit and AudioHwConfig.

Next modify the file *module_usb_audio\main.xc*, as shown below :

```
/* USB Interface Core */
{
#ifdef U208_ONLY
    set_core_high_priority_on();
#endif
#if (AUDIO_CLASS==2)
    XUD_Manager(c_xud_out, ENDPOINT_COUNT_OUT, c_xud_in, ENDPOINT_COUNT_IN,
            c_sof, epTypeTableOut, epTypeTableIn, p_usb_rst,
            clk, 1, XUD_SPEED_HS, XUD_PWR_CFG);
#else
    XUD_Manager(c_xud_out, ENDPOINT_COUNT_OUT, c_xud_in, ENDPOINT_COUNT_IN,
            c_sof, epTypeTableOut, epTypeTableIn, p_usb_rst,
            clk, 1, XUD_SPEED_FS, XUD_PWR_CFG);
#endif
}
```

This configures the core running the XUD to high priority to ensure that it has enough MIPS to execute the USB interface. Setting this will allocate 100MIPS to the USB task, leaving 400MIPS to be shared equally among the remaining tasks. It allows all remaining logical cores to be utilized. 400/7 = 57MIPS is sufficient for all other functions within the reference design when running at up to 384KHz, with stereo out.

APPENDIX A - USB audio firmware API

The complete details of the USB audio firmware API are provided in chapter 7 of the USB Audio Design Guide[5].

The chapter is split into the following sections:

Section	Contents
Configuration Defines	High level application configuration definitions (channel count, sample rate etc.), these defines are found in module_usb_audio/devicedefines.h.
Required User Function Definitions	Audio hardware initialization and configuration, HID configuration etc.
Component API	Functions can be called from the top level main of an application and and implement the various components, e.g. XUD, Endpoint0, mixer etc.

[5] https://www.xmos.com/support/boards?product=18334&component=14442

APPENDIX B - Building and launching the application

1. Right click on the project and select “Select As Current Project”
2. Right click on the project and select “Build Project”

Refer to the XMOS Tools User Guide in Appendix §C for information on building an executable binary and running, debugging or flashing it on the target hardware.

APPENDIX C - References

XMOS Tools User Guide

http://www.xmos.com/published/xtimecomposer-user-guide

XMOS xCORE Programming Guide

http://www.xmos.com/published/xmos-programming-guide

XMOS Platform Configuration and XN - Describe a Target Platform:

https://www.xmos.com/support/tools/documentation?subcategory=&component=14815

XMOS Platform Configuration and XN - XN Specification:

https://www.xmos.com/support/tools/documentation?subcategory=&component=14814

XMOS USB Audio Design Guide:

https://www.xmos.com/support/boards?product=18334&component=14442

APPENDIX D - Full source code listing

D.1 Source code for custom_XU208.xn

```
<?xml version="1.0" encoding="UTF-8"?>
<Network xmlns="http://www.xmos.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=
  ↪ "http://www.xmos.com http://www.xmos.com" ManuallySpecifiedRouting="true">
  <Type>Device</Type>
  <Name>custom XU208 hardware</Name>
  <Declarations>
    <Declaration>tileref tile[1]</Declaration>
    <Declaration>tileref usb_tile</Declaration>
  </Declarations>
  <Packages>
    <Package id="0" Type="XS2-UnA-128-TQ64">
      <Nodes>
        <Node Id="0" InPackageId="0" Type="XS2-L8A-128" Oscillator="24MHz" SystemFrequency="500MHz"
          ↪ referencefrequency="100MHz">
          <Boot>
            <Source Location="SPI:bootFlash"/>
          </Boot>
          <Tile Number="0" Reference="tile[0]">
            <Port Location="XS1_PORT_1B" Name="PORT_SQI_CS"/>
            <Port Location="XS1_PORT_1C" Name="PORT_SQI_SCLK"/>
            <Port Location="XS1_PORT_4B" Name="PORT_SQI_SIO"/>

            <!-- Audio Ports -->
            <Port Location="XS1_PORT_1N"  Name="PORT_MCLK_IN"/>
            <Port Location="XS1_PORT_1O"  Name="PORT_I2S_LRCLK"/>
            <Port Location="XS1_PORT_1P"  Name="PORT_I2S_BCLK"/>
            <Port Location="XS1_PORT_1M"  Name="PORT_I2S_DAC0"/>
            <Port Location="XS1_PORT_4E"  Name="PORT_I2C"/>
            <Port Location="XS1_PORT_1O"  Name="PORT_DSD_DAC0"/>
            <port Location="XS1_PORT_1M"  Name="PORT_DSD_DAC1"/>
            <Port Location="XS1_PORT_1P"  Name="PORT_DSD_CLK"/>
            <Port Location="XS1_PORT_1L"  Name="PORT_SPDIF_OUT"/>
            <Port Location="XS1_PORT_16B" Name="PORT_MCLK_COUNT"/>
            <!-- USB ports -->
            <Port Location="XS1_PORT_1H"  Name="PORT_USB_TX_READYIN"/>
            <Port Location="XS1_PORT_1J"  Name="PORT_USB_CLK"/>
            <Port Location="XS1_PORT_1K"  Name="PORT_USB_TX_READYOUT"/>
            <Port Location="XS1_PORT_1I"  Name="PORT_USB_RX_READY"/>
            <Port Location="XS1_PORT_1E"  Name="PORT_USB_FLAG0"/>
            <Port Location="XS1_PORT_1F"  Name="PORT_USB_FLAG1"/>
            <Port Location="XS1_PORT_1G"  Name="PORT_USB_FLAG2"/>
            <Port Location="XS1_PORT_8A"  Name="PORT_USB_TXD"/>
            <Port Location="XS1_PORT_8B"  Name="PORT_USB_RXD"/>
          </Tile>
        </Node>
        <Node Id="1" InPackageId="1" Type="periph:XS1-SU" Reference="usb_tile" Oscillator="24MHz">
        </Node>
      </Nodes>
      <Links>
        <Link Encoding="5wire">
          <LinkEndpoint NodeId="0" Link="8" Delays="52clk,52clk"/>
          <LinkEndpoint NodeId="1" Link="XL0" Delays="1clk,1clk"/>
        </Link>
      </Links>
    </Package>
  </Packages>
  <Nodes>
    <Node Id="2" Type="device:" RoutingId="0x8000">
      <Service Id="0" Proto="xscope_host_data(chanend c);">
        <Chanend Identifier="c" end="3"/>
      </Service>
    </Node>
  </Nodes>
  <Links>
    <Link Encoding="2wire" Delays="4,4" Flags="XSCOPE">
      <LinkEndpoint NodeId="0" Link="XL0"/>
      <LinkEndpoint NodeId="2" Chanend="1"/>
    </Link>
  </Links>
  <ExternalDevices>
    <Device NodeId="0" Tile="0" Class="SQIFlash" Name="bootFlash" Type="S25FL116K">
      <Attribute Name="PORT_SQI_CS" Value="PORT_SQI_CS"/>
      <Attribute Name="PORT_SQI_SCLK"   Value="PORT_SQI_SCLK"/>
      <Attribute Name="PORT_SQI_SIO"  Value="PORT_SQI_SIO"/>
    </Device>
  </ExternalDevices>

  <JTAGChain>
    <JTAGDevice NodeId="0"/>
    <JTAGDevice NodeId="1"/>
  </JTAGChain>
</Network>
```

D.2 Source code for Makefile

```
# The TARGET variable determines what target system the application is
# compiled for. It either refers to an XN file in the source directories
# or a valid argument for the --target option when compiling
TARGET = custom_XU208

# The APP_NAME variable determines the name of the final .xe file. It should
# not include the .xe postfix. If left blank the name will default to
# the project name
APP_NAME = app_usb_aud_custom_XU208_HW

# The flags passed to xcc when building the application
BUILD_FLAGS     = -DFLASH_MAX_UPGRADE_SIZE=64*1024 -fcomment-asm -Xmapper --map -Xmapper MAPFILE -Wall -O3 -
   ↪ report -lquadflash -fsubword-select -save-temps -g -fxscope -DXSCOPE -DSDA_HIGH=2 -DSCL_HIGH=1 -
   ↪ DXUD_SERIES_SUPPORT=4 -march=xs2a -DUSB_TILE=tile[0] -DADAT_TX_USE_SHARED_BUFF=1 -DU208_ONLY

# The USED_MODULES variable lists other module used by the application. These
# modules will extend the SOURCE_DIRS, INCLUDE_DIRS and LIB_DIRS variables.
# Modules are expected to be in the directory above the BASE_DIR directory.
USED_MODULES = module_adat_rx module_adat_tx module_dfu module_i2c_shared module_i2c_single_port
   ↪ module_spdif_rx module_spdif_tx module_usb_audio module_usb_device module_usb_midi module_usb_shared
   ↪ module_xud

# Build config naming scheme:

# Audio Class:  1 or 2
# Input         enabled: i (channelcount)
# Output        enabled: o (channelcount)
# MIDI          enabled: m, disabled: x
# SPDIF out     enabled: s, disabled: x
# SPDIF in      enabled: s, disabled: x
# ADAT out      enabled: a, disabled: x
# ADAT in       enabled: a, disabled: x
# DSD out       enabled: d, disabled: x
# e.g. 2i10o10xsxxx: Audio class 2.0, input and output enabled (10 channels each), SPDIF output, no SPDIF
   ↪ input, no ADAT

# Test build configs (Note these make use of the defaults in customdefines.h)
XCC_FLAGS = $(BUILD_FLAGS) -DI2S_CHANS_DAC=2 -DI2S_CHANS_ADC=0 -DNUM_USB_CHAN_OUT=2 -DNUM_USB_CHAN_IN=0 -DMIDI
   ↪ =0 -DSPDIF_TX=1 -DSPDIF_RX=0 -DADAT_TX=0 -DADAT_RX=0 -DDSD_CHANS_DAC=2 -DMAX_FREQ=192000

ifeq ($(TEST_CONFIGS),1)
XCC_FLAGS_upgrade1 = $(BUILD_FLAGS) -DBCD_DEVICE_J=0x99 -DBCD_DEVICE_M=0x0 -DBCD_DEVICE_N=0x1
XCC_FLAGS_upgrade2 = $(BUILD_FLAGS) -DBCD_DEVICE_J=0x99 -DBCD_DEVICE_M=0x0 -DBCD_DEVICE_N=0x2
endif


MODULE_LIBRARIES = xud_x200

#=============================================================================
#=============================================================================
# The following part of the Makefile includes the common build infrastructure
# for compiling XMOS applications. You should not need to edit below here.

XMOS_MAKE_PATH ?= ../..
ifneq ($(wildcard $(XMOS_MAKE_PATH)/xcommon/module_xcommon/build/Makefile.common),)
include $(XMOS_MAKE_PATH)/xcommon/module_xcommon/build/Makefile.common
```

```
else
include ../module_xcommon/build/Makefile.common
endif
```

D.3 Source code for customdefines.h

```
// Copyright (c) 2016, XMOS Ltd, All rights reserved

#ifndef _CUSTOMDEFINES_H_
#define _CUSTOMDEFINES_H_

//#include "user_main.h"

/*
 * Device configuration option defines to override default defines found devicedefines.h
 *
 * Build can be customised but changing and adding defines here
 *
 * Note, we check if they are already defined in Makefile
 */

/* Tile defines */
#define AUDIO_IO_TILE      0
#define XUD_TILE           0
#define SPDIF_TX_TILE      0
#define MIDI_TILE          0

/* Mixer core enabled by default */
#ifndef MIXER
#define MIXER              1
#endif

/* Mixing disabled by default */
#ifndef MAX_MIX_COUNT
#define MAX_MIX_COUNT      2
#endif

/* Board is self-powered i.e. not USB bus-powered */
#define SELF_POWERED       1

/* Enable/Disable MIDI - Default is MIDI off */
#ifndef MIDI
#define MIDI                   0
#endif

/* Enable/Disable SPDIF output - Default is S/PDIF on */
#ifndef SPDIF_TX
#define SPDIF_TX               1
#endif

/* Defines relating to channel count and channel arrangement (Set to 0 for disable) */
//:audio_defs
/* Number of USB streaming channels - Default is 4 in 4 out */
#ifndef NUM_USB_CHAN_IN
#define NUM_USB_CHAN_IN    (0)          /* Device to Host */
#endif
#ifndef NUM_USB_CHAN_OUT
#define NUM_USB_CHAN_OUT   (2)          /* Host to Device */
#endif

/* Number of IS2 chans to DAC..*/
#ifndef I2S_CHANS_DAC
#define I2S_CHANS_DAC      (2)
#endif

/* Number of I2S chans from ADC */
#ifndef I2S_CHANS_ADC
#define I2S_CHANS_ADC      (0)
#endif

/* Number of DSD chans to DAC..*/
#ifndef DSD_CHANS_DAC
#define DSD_CHANS_DAC      (2)
```

```
#endif

/* Channel index of SPDIF Rx channels (duplicated DAC channels 1/2 when index is 0) */
#define SPDIF_TX_INDEX     (0)

/* Channel index of SPDIF Rx channels */
#define SPDIF_RX_INDEX     (0)

/* Channel index of ADAT Tx channels */
#if defined(SPDIF_TX) && (SPDIF_TX==1)
#define ADAT_TX_INDEX      (SPDIF_TX_INDEX + 2)
#else
#define ADAT_TX_INDEX      (I2S_CHANS_DAC)
#endif

/* Channel index of ADAT Rx channels */
#if defined(SPDIF_RX) && (SPDIF_RX==1)
#define ADAT_RX_INDEX      (SPDIF_RX_INDEX + 2)
#else
#define ADAT_RX_INDEX      (I2S_CHANS_ADC)
#endif

/* Master clock defines (in Hz) */
#define MCLK_441           (512*44100)   /* 44.1, 88.2 etc */
#define MCLK_48            (512*48000)   /* 48, 96 etc */

/* Maximum frequency device runs at */
#ifndef MAX_FREQ
#define MAX_FREQ           (192000)
#endif

//:
/***** Defines relating to USB descriptors etc *****/
//:usb_defs
#define VENDOR_ID          (0x20B1) /* XMOS VID */
#define PID_AUDIO_2        (0x0008) /* SKC_SU1 USB Audio Reference Design PID */
#define PID_AUDIO_1        (0x0009) /* SKC_SU1 Audio Reference Design PID */
//:

/* Enable/Disable example HID code */
#ifndef HID_CONTROLS
#define HID_CONTROLS       1
#endif

#endif
```

D.4 Source code for audiohw.xc

```
// Copyright (c) 2016, XMOS Ltd, All rights reserved

#include <xs1.h>
#include <assert.h>
#include "devicedefines.h"
#include <platform.h>
#include "i2c_shared.h"
#include "print.h"
#include "dsd_support.h"

#define DAC_I2C_ADDR 0x48
#define ADC_I2C_ADDR 0x5A

#ifndef IAP
/* If IAP not enabled, i2c ports not declared - still needs for DAC config */
on tile [0] : struct r_i2c r_i2c = {XS1_PORT_4A};
#else
extern struct r_i2c r_i2c;
#endif

#define DAC_REGWRITE(reg, val) {data[0] = val; i2c_shared_master_write_reg(r_i2c, DAC_I2C_ADDR, reg, data, 1)
    ↪ ;}
#define DAC_REGREAD(reg, val)  {i2c_shared_master_read_reg(r_i2c, DAC_I2C_ADDR, reg, val, 1);}
#define ADC_REGWRITE(reg, val) {data[0] = val; i2c_shared_master_write_reg(r_i2c, ADC_I2C_ADDR, reg, data, 1)
    ↪ ;}
```

```
void AudioHwInit(chanend ?c_codec)
{
    unsigned char data[2] = {0, 0};

    /* Init the i2c module */
    i2c_shared_master_init(r_i2c);

    // To do: initialise the audio hardware here including PLL selection, ADC (if any) and DAC (if any).
}

/* Configures the external audio hardware for the required sample frequency.*/
void AudioHwConfig(unsigned samFreq, unsigned mClk, chanend ?c_codec, unsigned dsdMode,
    unsigned sampRes_DAC, unsigned sampRes_ADC)
{
        unsigned char data[1] = {0};

    // To do: Handle the events listed below and reconfigure the PLL, ADC (if any) and DAC (if any)
    // (1) sample rate change
    // (2) PCM/DSD mode switching


   return;
}
//:
```