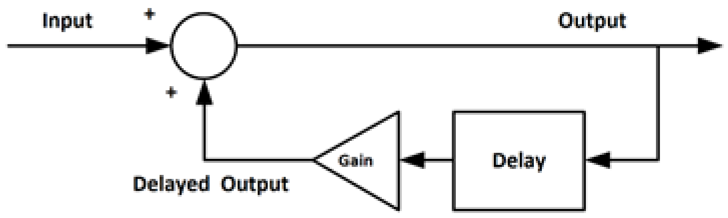
Workshop:
Control of Audio Weaver blocks on STM32F407 Discovery board

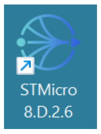
- Reverberation loop
- Designating control blocks in Audio Weaver Designer
- Reviewing the project in STM32CubeIDE
- STM32 configuration file
- Connecting ADC input and headphone output to Discovery board
- Running the reverberation project
- Variable voltage divider to control AWE variables

Reverberation loop

A simple reverberation loop uses positive feedback from output to input.



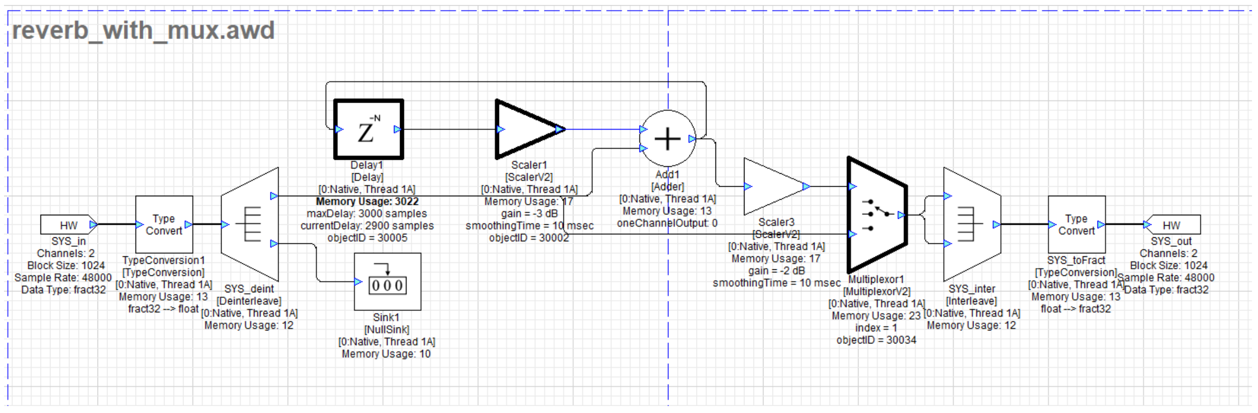
Open Audio Weaver Designer by double-clicking the icon:



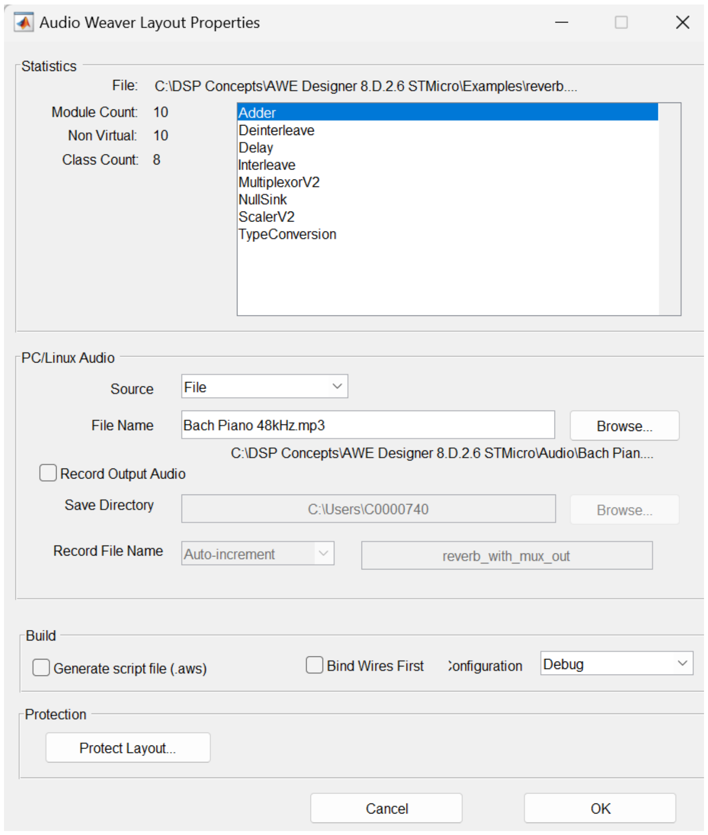
In AWE Server, ensure your Target connection is Native.

From the File menu, open the file:

```
C:\DSP Concepts\AWE Designer 8.D.2.6 STMicro\Examples\reverb_with_mux.awd
```



Under Layout Properties, verify that File is selected as the source. Also make sure that Debug is chosen for Configuration, at the bottom right. Click OK.


Audio Weaver Layout Properties

Statistics

File: C:\DSP Concepts\AWE Designer 8.D.2.6 STMicro\Examples\reverb....

Module Count: 10
Non Virtual: 10
Class Count: 8

Adder
Deinterleave
Delay
Interleave
MultiplexorV2
NullSink
ScalerV2
TypeConversion

PC/Linux Audio

Source: File
File Name: Bach Piano 48kHz.mp3
C:\DSP Concepts\AWE Designer 8.D.2.6 STMicro\Audio\Bach Pian....
Record Output Audio: ☐
Save Directory: C:\Users\C0000740
Record File Name: Auto-increment, reverb_with_mux_out

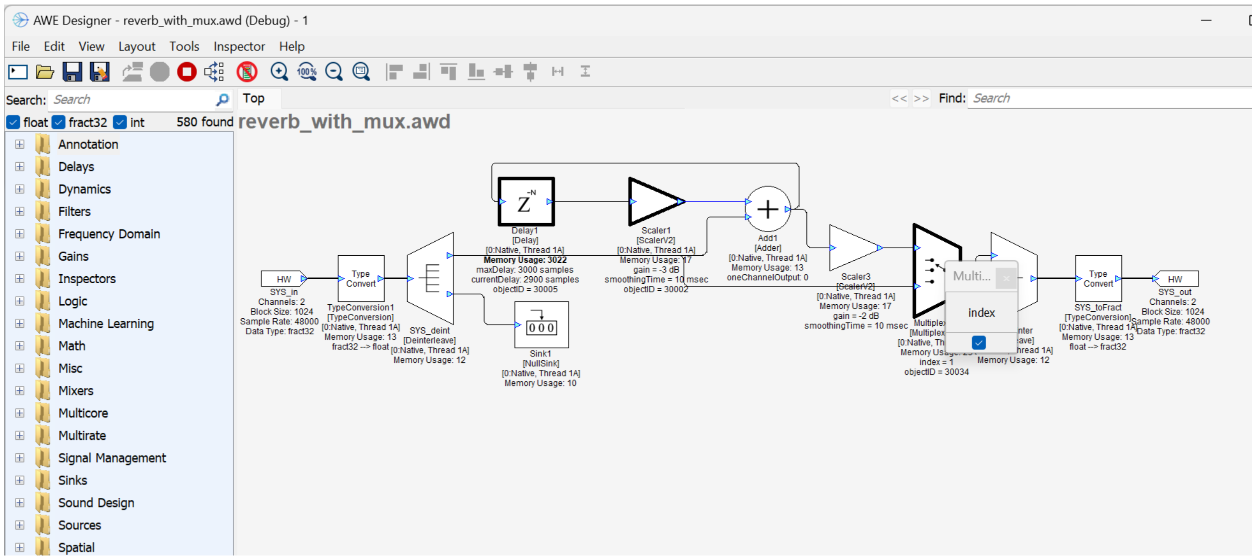
Build

Generate script file (.aws): ☐
Bind Wires First: ☐
Configuration: Debug

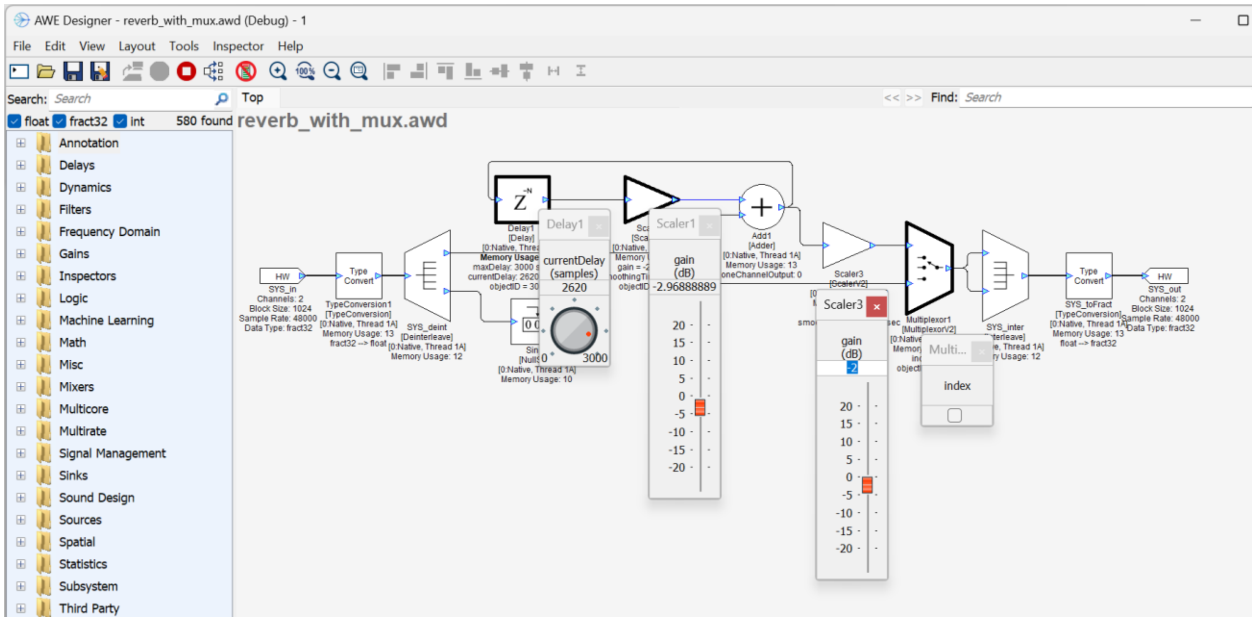
Protection

Click Play ▶.

While the Bach music plays, double-click on Multiplexor1. When the Index is checked off, you should hear the original music. When you uncheck the box you will hear the music with reverberation effects.



Double-click Delay1, Scaler1, and Scaler3 to open their controls. Notice how the effects change when you vary the delay and gain in the feedback loop. The overall volume should change with the gain settings for Scaler3. Check off the multiplexor whenever you want to hear the original music. Note that all of your changes are effective in real time.



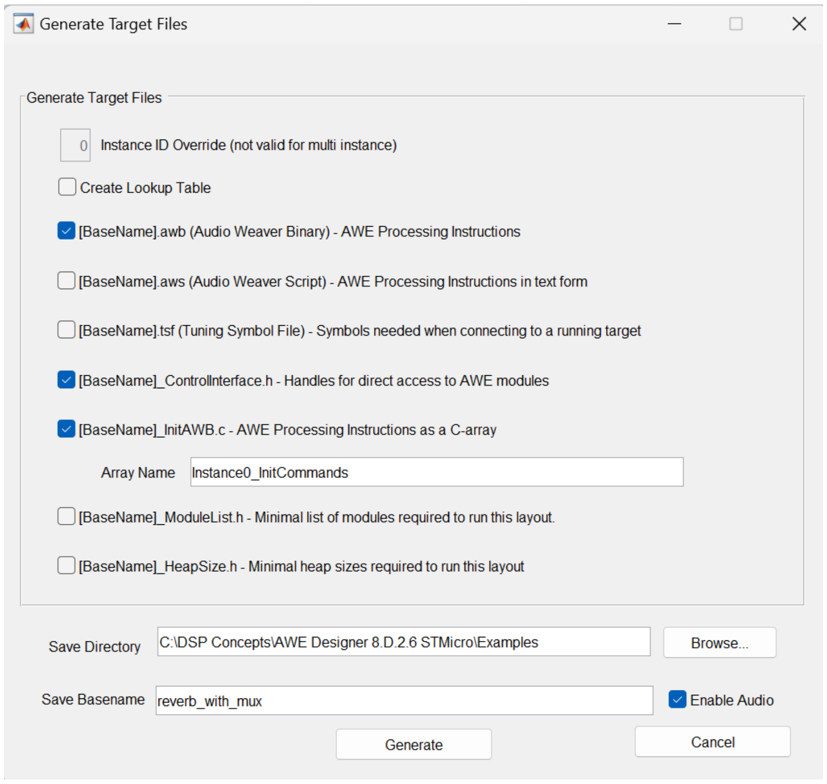
Designating control blocks in Audio Weaver Designer

All blocks are controllable in real time when a layout is running in Native mode on your PC. However, not all blocks can be controlled once the code is exported to the STM32F407 Discovery board. The three blocks that are outlined in black are special. It is these blocks that will be accessible through a control interface when code is exported. You can view the properties of any module by right-clicking on the block and choosing View Properties. Locate the object ID for the blocks in the layout. If an object ID between 30000 and 32767 is assigned for any block, that block becomes controllable following export.

Choose settings for Delay1 and Scaler1 that produce a noticeable reverb effect. If you cannot find a set you are satisfied with, you can choose a delay of 2900 samples, and a gain of -3 dB.

It will be helpful to set Scaler3 to -2 dB. This block will not be controllable after export.

Choose Tools – Generate Target Files. Three boxes should be checked off, and you should choose a Basename for the generated files. Note that the generated files will appear in `C:\DSP Concepts\AWE Designer 8.D.2.6 STMicro\Examples`.



Generate Target Files

Generate Target Files

0 Instance ID Override (not valid for multi instance)

[ ] Create Lookup Table

[✓] [BaseName].awb (Audio Weaver Binary) - AWE Processing Instructions

[ ] [BaseName].aws (Audio Weaver Script) - AWE Processing Instructions in text form

[ ] [BaseName].tsf (Tuning Symbol File) - Symbols needed when connecting to a running target

[✓] [BaseName]_ControlInterface.h - Handles for direct access to AWE modules

[✓] [BaseName]_InitAWB.c - AWE Processing Instructions as a C-array

Array Name: Instance0_InitCommands

[ ] [BaseName]_ModuleList.h - Minimal list of modules required to run this layout.

[ ] [BaseName]_HeapSize.h - Minimal heap sizes required to run this layout

Save Directory: C:\DSP Concepts\AWE Designer 8.D.2.6 STMicro\Examples

Save Basename: reverb_with_mux

[✓] Enable Audio

Go to `C:\DSP Concepts\AWE Designer 8.D.2.6 STMicro\Examples`. You should expect to see six files with your Basename:

- reverb_with_mux.awb
- reverb_with_mux.awd
- reverb_with_mux_ControlInterface.h
- reverb_with_mux_ControlInterface_AWE6.h
- reverb_with_mux_InitAWB.c
- reverb_with_mux_InitAWB.h

The file `reverb_with_mux.awd` is the layout for Audio Weaver Designer and does not need to be included in your STM32 project.

Reviewing the project in STM32CubeIDE

Open STM32CubeIDE by clicking on the STM32CubeIDE 1.17.0 icon.

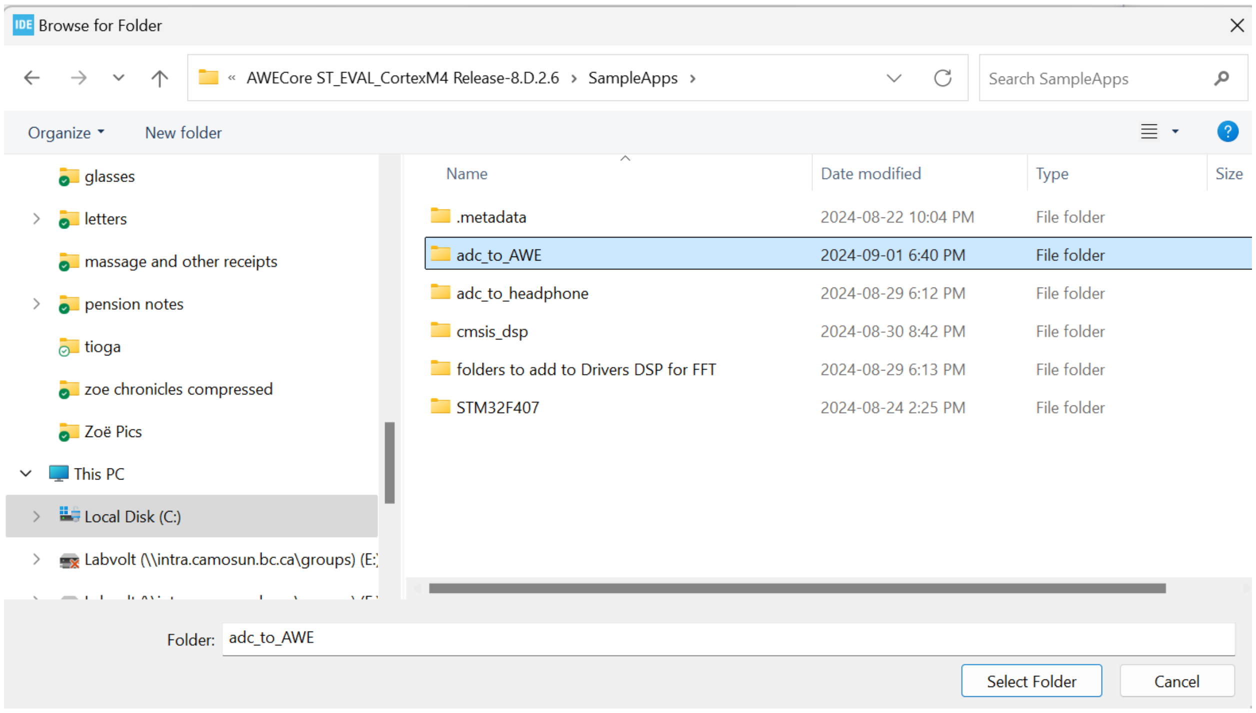
For the workspace directory, enter:

```
C:\DSP Concepts\AWECore ST_EVAL_CortexM4 Release-8.D.2.7\SampleApps
```

Click Launch.

Choose File – Open Projects from File System. For Import Source, choose:

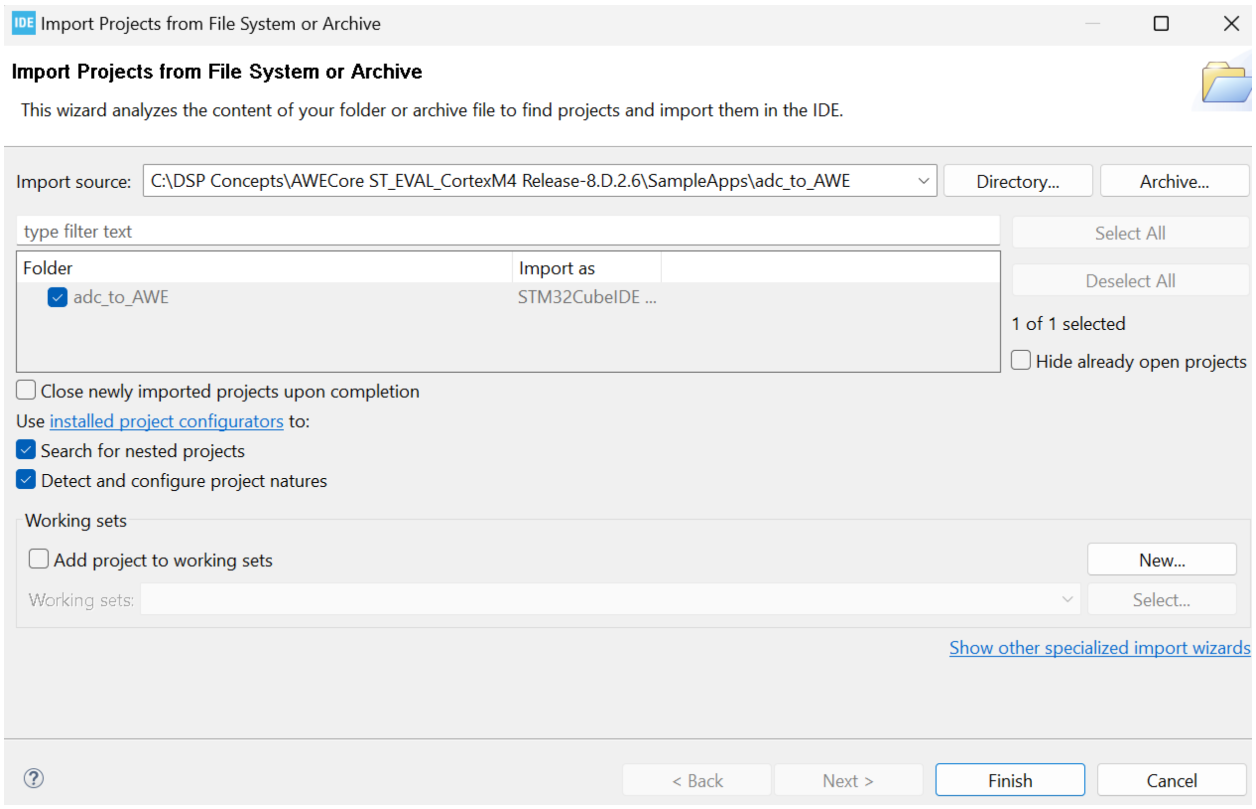
```
C:\DSP Concepts\AWECore ST_EVAL_CortexM4 Release-
8.D.2.7\SampleApps\adc_to_AWE
```


Browse for Folder
« AWECore ST_EVAL_CortexM4 Release-8.D.2.6 › SampleApps ›
Search SampleApps
Organize
New folder
glasses
letters
massage and other receipts
pension notes
tioga
zoe chronicles compressed
Zoë Pics
This PC
Local Disk (C:)
Labvolt (\\intra.camosun.bc.ca\groups) (E:)

Name	Date modified	Type	Size
.metadata	2024-08-22 10:04 PM	File folder	
adc_to_AWE	2024-09-01 6:40 PM	File folder	
adc_to_headphone	2024-08-29 6:12 PM	File folder	
cmsis_dsp	2024-08-30 8:42 PM	File folder	
folders to add to Drivers DSP for FFT	2024-08-29 6:13 PM	File folder	
STM32F407	2024-08-24 2:25 PM	File folder	

Folder: adc_to_AWE

In the Import window, click Finish.



Import Projects from File System or Archive

Import Projects from File System or Archive

This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.

Import source: C:\DSP Concepts\AWECore ST_EVAL_CortexM4 Release-8.D.2.6\SampleApps\adc_to_AWE

Folder	Import as
☑ adc_to_AWE	STM32CubeIDE ...

1 of 1 selected

☐ Hide already open projects

☐ Close newly imported projects upon completion

Use installed project configurators to:

☑ Search for nested projects

☑ Detect and configure project natures

Working sets

☐ Add project to working sets

Working sets:

The five generated files have already been added to `Core/Src` and `Core/Inc` in the project directory `C:\DSP Concepts\AWECore ST_EVAL_CortexM4 Release-8.D.2.7\SampleApps\adc_to_AWE`. The two files:

```
reverb_with_mux.awb
reverb_with_mux_InitAWB.c
```

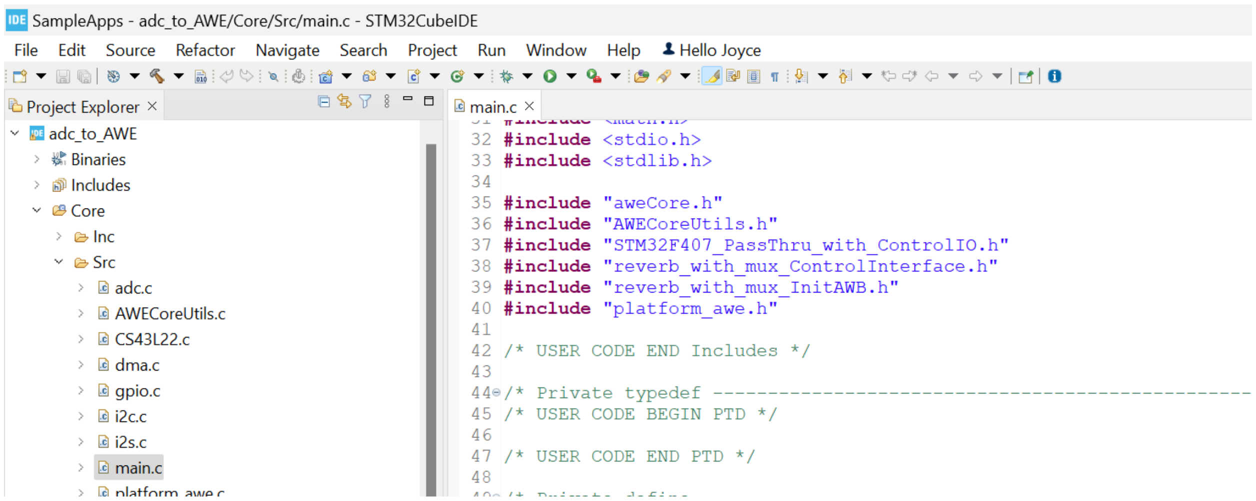
belong in `Core/Src`, and the three files:

```
reverb_with_mux_ControlInterface.h
reverb_with_mux_ControlInterface_AWE6.h
reverb_with_mux_InitAWB.h
```

belong in `Core/Inc`.

You can use this procedure to replace the five `reverb_with_mux` files with the Basenamed files for any Audio Weaver Designer layout you create yourself.

Open `main.c`. When you create your own Audio Weaver Designer layouts, you must make one other modification in the code. Where you see **#include** "reverb_with_mux_ControlInterface.h" and **#include** "reverb_with_mux_InitAWB.h", you must change to your basename: **#include** "*yourbasename*_ControlInterface.h" and **#include** " *yourbasename* _InitAWB.h".


SampleApps - adc_to_AWE/Core/Src/main.c - STM32CubeIDE

```
#include <stdio.h>
#include <stdlib.h>

#include "aweCore.h"
#include "AWECoreUtils.h"
#include "STM32F407_PassThru_with_ControlIO.h"
#include "reverb_with_mux_ControlInterface.h"
#include "reverb_with_mux_InitAWB.h"
#include "platform_awe.h"

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */
```

From `Core/Inc`, open `reverb_with_mux_ControlInterface.h`. You should find references to `currentDelay` for `Delay1`, `gain` for `Scaler1`, and `index` for `Multiplexor1`, the three blocks that were designated in Audio Weaver Designer. For each, you can note the type for the variable, as well as acceptable range of values for the variable.

Input for this program comes from an ADC, connected between pin PA1 and GND. The output is connected to the headphone jack of the Discovery board. Both input and output have DMA: one half of a DMA buffer is filling, while the other half is processing or transmitting.

From `Core/Src`, open `main.c`. In the `while(1)` loop, ADC DMA callbacks move input signal samples to a `send_to_AWE_buffer` array. These values are imported to the AWE layout now stored on the STM32F407, and pumped through the layout. I2S DMA callbacks move the outputs from the layout, stored in a `receive_from_AWE_buffer` array, to the headphone output. The information about the Audio Weaver Designer layout is contained in the target files you generated earlier.

The gain for `Scaler1` is handled by this `if` statement:

```
if (awe_ctrlGetModuleClass(&g_AWEInstance, AWE_Scaler1_gain_HANDLE, &classID) ==
OBJECT_FOUND)
{
   // Check that module assigned this object ID is of module class Scaler1
   if (classID == AWE_Scaler1_classID)
   {
      // convert ADC value to a scaler1Gain in dB between -24 and 24 as per
ControlInterface.h
//       scaler1Gain=potValue/2400.0*48.0-24.0;
//       if(scaler1Gain < -24.0) scaler1Gain = -24.0;
//       if(scaler1Gain > 24.0)scaler1Gain = 24.0;
      scaler1Gain = -3.0;
      awe_ctrlSetValue(&g_AWEInstance, AWE_Scaler1_gain_HANDLE, (void *)&scaler1Gain,
0, 1);
   }
}
```

At the moment, `scaler1Gain` is set to -3.0. The lines currently commented out will permit `scaler1Gain` to be computed from an ADC input value.

The delay for `Delay1` is handled by this `if` statement:

```
if (awe_ctrlGetModuleClass(&g_AWEInstance, AWE_Delay1_currentDelay_HANDLE, &classID)
== OBJECT_FOUND)
{
   // Check that module assigned this object ID is of module class SinkInt
   if (classID == AWE_Delay1_classID)
   {
      // convert ADC value to a Delay1Value between 0 and 3000  per ControlInterface.h
//       delay1Value=(int)(potValue2/2400.0*3000.0);
//       if(delay1Value < 0) delay1Value = 0;
//       if(delay1Value > 3000) delay1Value = 3000;
      delay1Value = 2900;
      awe_ctrlSetValue(&g_AWEInstance, AWE_Delay1_currentDelay_HANDLE, (void
*)&delay1Value, 0, 1);
   }
}
```

At the moment, `delay1Value` is set to 2900. The lines currently commented out will permit `delay1Value` to be computed from an ADC input value.

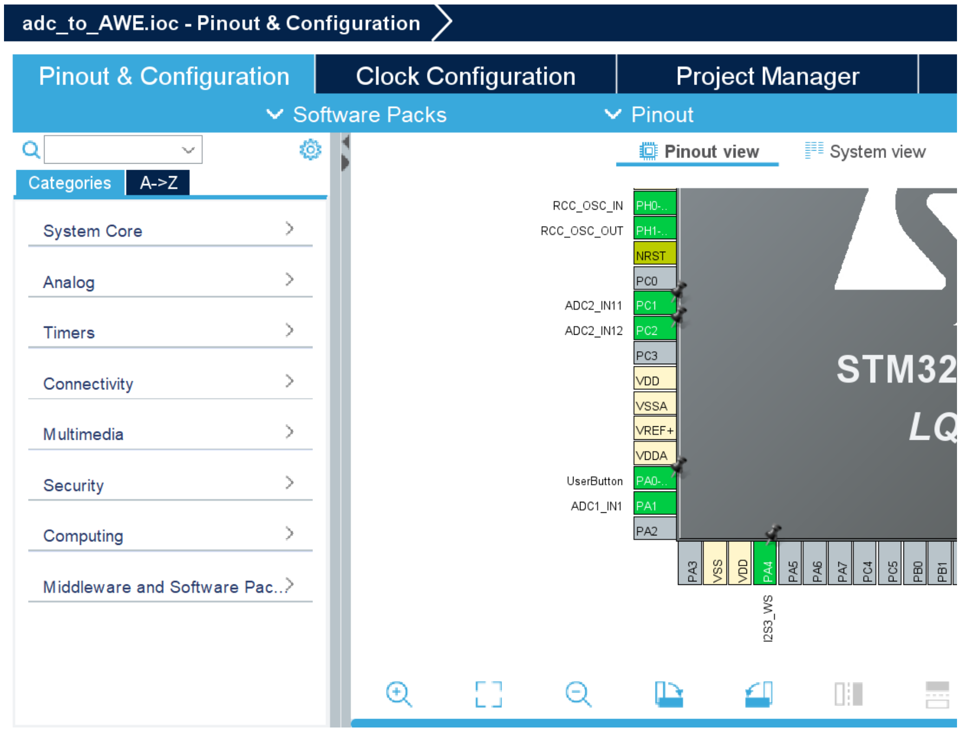
The index for `Multiplexor1` is handled by this `if` statement:

```
if (awe_ctrlGetModuleClass(&g_AWEInstance, AWE_Multiplexor1_index_HANDLE, &classID) ==
OBJECT_FOUND)
{
   // Check that module assigned this object ID is of module class SinkInt
   if (classID == AWE_Multiplexor1_classID)
   {
      // use this method to provide momentary change to no reverb
      // pushed button = 1, unpushed button = 0
      if(HAL_GPIO_ReadPin(UserButton_GPIO_Port, UserButton_Pin)==GPIO_PIN_SET)
      {
         user_button = 1;
      }
      else
      {
         user_button = 0;
      }
      awe_ctrlSetValue(&g_AWEInstance, AWE_Multiplexor1_index_HANDLE, (void
*)&user_button, 0, 1);
   }
}
```

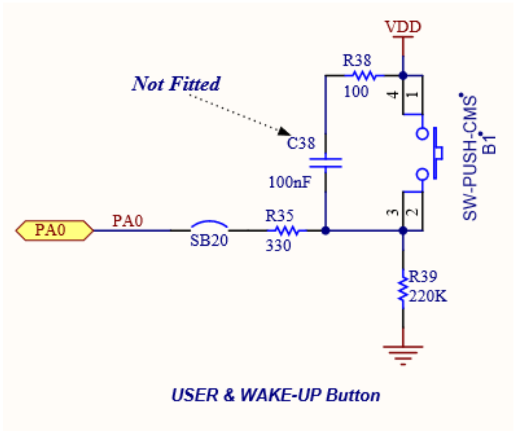
The user button on the Discovery board is bright blue. Pressing this button will be equivalent to checking off the multiplexor index in Audio Weaver Designer. This means that when the button is pressed, you will hear original sound; when the button is not pressed, you will hear reverberating sound.

STM32 configuration file

From your Project Explorer, double-click on `adc_to_AWE.ioc`.



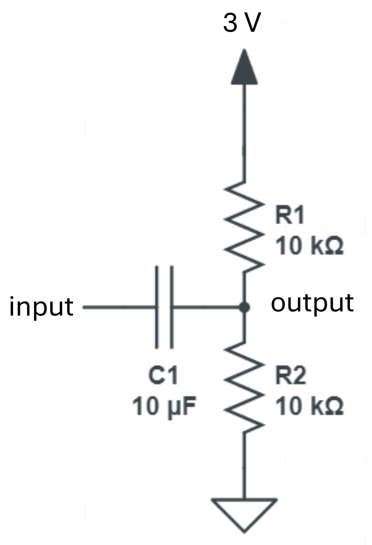
Three ADC channels are in use. ADC1 Channel 1, on pin PA1, is the ADC that accepts the input signal. ADC2 Channel 11 and ADC2 Channel 12, on pins PC1 and PC2, will provide voltages to control the gain and delay in the reverberation positive feedback loop. The user button is connected to pin PA0, as confirmed in the schematic for the board.



Connecting ADC input and headphone output to Discovery board

Plug a micro USB cable into one end of the Discovery board and a mini USB cable into the other end. Plug both USB cables into your PC.

The analog-to-digital converters on the Discovery board expect non-negative voltage inputs between 0 and 3 V. These voltages are mapped to 12-bit values from 0 to 4095. The audio from your PC will have a zero mean voltage, and the negative-going voltages will be chopped off by the ADC. In this workshop, you will use music from YouTube as the input signal, so it will be important to use a circuit that shifts the DC bias from 0 V to 1.5 V. Build this circuit on a breadboard. Use the top bar of the breadboard as the power rail, and the bottom bar of the breadboard as a ground rail.



Connect a stereo mini to flying leads cable to the headphone jack of your PC. Using a bit of wire inserted in the black ground jumper, connect the ground jumper to the ground rail of your breadboard. Using another bit of wire inserted in one of the signal jumpers (left or right), connect the signal to the input point of your circuit.

Using a piece of wire inserted into a jumper, connect the ground rail of your breadboard to a GND pin on the Discovery board.

Use the same method to connect the power rail of your breadboard to a 3 V pin on the Discovery board, and to connect the output from your circuit to pin PA1.

Once you have done this, you have connected input from your PC to the Discovery board. Choose some music from YouTube. If you have an oscilloscope, you should be able to see that the signal at the input point of your breadboard has mean 0 V, and the signal at the output point of your breadboard has mean 1.5 V.

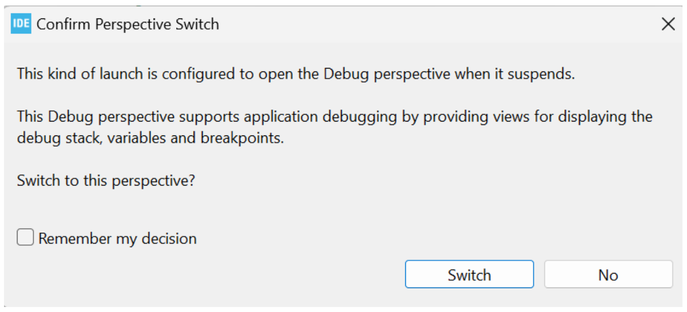
Note that in this project just one stereo channel is connected (at pin PA1). To introduce full stereo input, another ADC pin would need to be configured. The second stereo input would also require a DC offset circuit.

Connect a headphone to the headphone jack of the Discovery board.

Running the reverberation project

Click Run – Debug (or click on the picture of a bug) to load your program onto the Discovery board.

Agree to Perspective Switch.



Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective supports application debugging by providing views for displaying the debug stack, variables and breakpoints.

Switch to this perspective?

☐ Remember my decision

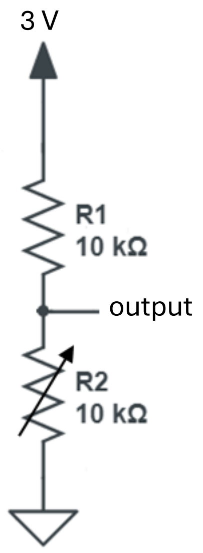
Once the download has been verified successfully, click Play/Resume:

On your headphone you should hear reverberating music from YouTube. Whenever you are pressing the blue user button, you should hear the original version of the music.

At this moment, you have no way to control the gain or the delay in your program in real time. You can only edit the values of `scaler1Gain` and `delay1Value` and rerun your program. You are welcome to try this.

Variable voltage divider to control AWE variables

A variable voltage divider circuit will provide control of gain and/or delay. As a potentiometer varies between 0 Ω and 10 kΩ, the voltage at the output varies between 0 V and 1.5 V. A 12-bit ADC will interpret this range of voltages as numbers in the approximate range 0 to 2047. Because the power supply voltage and the resistor values are not precise, the range may vary somewhat.



On your breadboard, build a variable voltage divider circuit. Use the top rail of the breadboard for the power supply and the bottom rail for ground. Your DC bias circuit already has power and ground connections that can be shared with your variable voltage divider circuit.

Connect the output of your variable voltage divider circuit to pin PC2. This pin connects to ADC2 Channel 12, which is used to set the delay value.

Go back to `main.c`. Within the `while(1)` loop, a loop counter is used to blink an orange LED about every 1 second. During this 1 second interval, 100 samples are taken from ADC2 Channel 11 and 100 samples are taken from ADC2 Channel 12. The file `adc.c` provides the ADC configuration details. At the same time as the orange LED is toggled, the ADC readings are averaged. Averaging is a good way to overcome noise. Averaged numbers for ADC2 Channels 11 and 12 are stored in `potValue` and `potValue2`. These numbers are proportional to the voltage across the potentiometers.

Modify the `if` statement for `Delay1` as follows:

```
if (awe_ctrlGetModuleClass(&g_AWEInstance, AWE_Delay1_currentDelay_HANDLE, &classID)
== OBJECT_FOUND)
{
   // Check that module assigned this object ID is of module class SinkInt
   if (classID == AWE_Delay1_classID)
   {
      // convert ADC value to a Delay1Value between 0 and 3000  per ControlInterface.h
      delay1Value=(int)(potValue2/2400.0*3000.0);
      if(delay1Value < 0) delay1Value = 0;
      if(delay1Value > 3000) delay1Value = 3000;
//      delay1Value = 2900;
      awe_ctrlSetValue(&g_AWEInstance, AWE_Delay1_currentDelay_HANDLE, (void
*)&delay1Value, 0, 1);
   }
}
```

Now the hard-coded value for `delay1Value` is commented out. Instead, `potValue2` is converted to a delay between 0 and 3000 samples.

Run – Debug your project and Play/Resume. Make sure music is still playing on YouTube. You should hear reverberating music, and when you press the user button you should hear unaltered music, as before.

Adjust the potentiometer value while your project is running. Enter `potValue2` and `delay1Value` in Live Expressions. You should see the values changing as you adjust the resistance of your potentiometer, and you should hear changes in the reverberating sound.

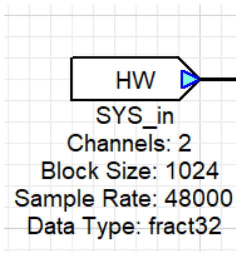
At this point your Discovery board project is running standalone, and you are able to control variables in the project in real time. Your PC is supplying power and music, but is not doing any processing. Now imagine the Discovery board is powered by a power pack and connected to an external microphone. Through this process, you could create an interesting layout in Audio Weaver Designer and build a standalone microcontroller product with controls to manage the variables in your software in real time. The potentiometers could be knobs or sliders on the outside of a box.

If you like, you may build a second variable voltage divider to control `scaler1Gain`, connected to pin PC1 on the Discovery board. In this case, the `if` statement for `Scaler1` will be modified in this way:

```
if (awe_ctrlGetModuleClass(&g_AWEInstance, AWE_Scaler1_gain_HANDLE, &classID) ==
OBJECT_FOUND)
{
   // Check that module assigned this object ID is of module class Scaler1
   if (classID == AWE_Scaler1_classID)
   {
      // convert ADC value to a scaler1Gain in dB between -24 and 24 as per
ControlInterface.h
      scaler1Gain=potValue/2400.0*48.0-24.0;
      if(scaler1Gain < -24.0) scaler1Gain = -24.0;
      if(scaler1Gain > 24.0)scaler1Gain = 24.0;
//      scaler1Gain = -3.0;
      awe_ctrlSetValue(&g_AWEInstance, AWE_Scaler1_gain_HANDLE, (void *)&scaler1Gain,
0, 1);
   }
}
```

If you are feeling ambitious and time allows, you can create a simple layout in Audio Weaver that involves a gain or delay block. Listen to the effects on your PC and generate target files when you are satisfied with operation of your layout. Follow the process outlined in this workshop to create your own standalone product.

Sampling frequency and block size are properties of the HW input block in Audio Weaver Designer, and may be edited by right-clicking on the block and choosing View Properties. The sampling frequency for your Audio Weaver layout should be 48 kHz, to match the `adc_to_AWE` project settings.



The block size from your Audio Weaver layout determines the size of half a DMA buffer through the line:

```
#define N  AUDIO_BLOCK_SIZE
```

The processing required during each loop of your software determines the block size needed. Processing must be completed within the time it takes for one half of a DMA buffer to fill. You can certainly reduce the block size if your processing demands are low. If you begin to hear glitches in your output, you may need to increase block size.