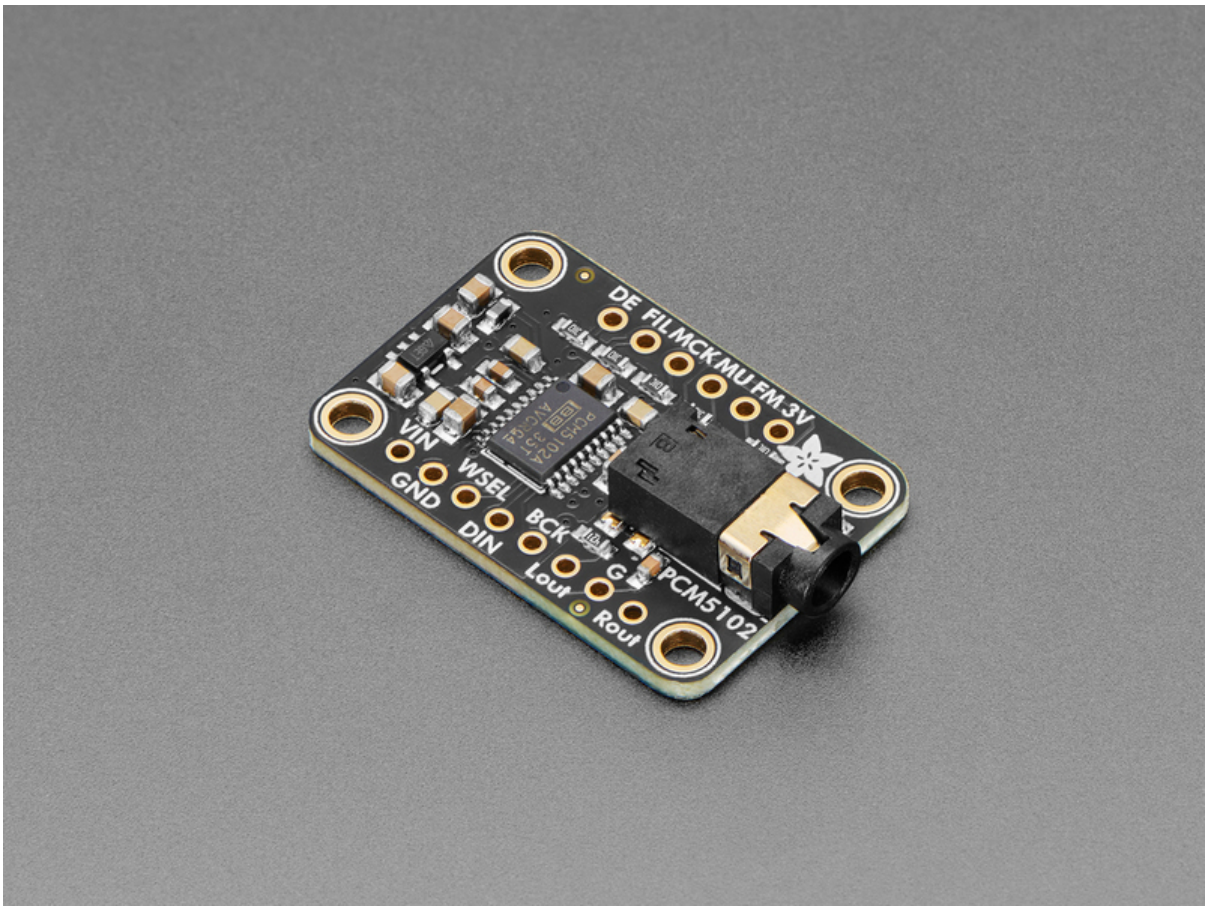
Adafruit PCM510x I2S DAC

Created by Liz Clark

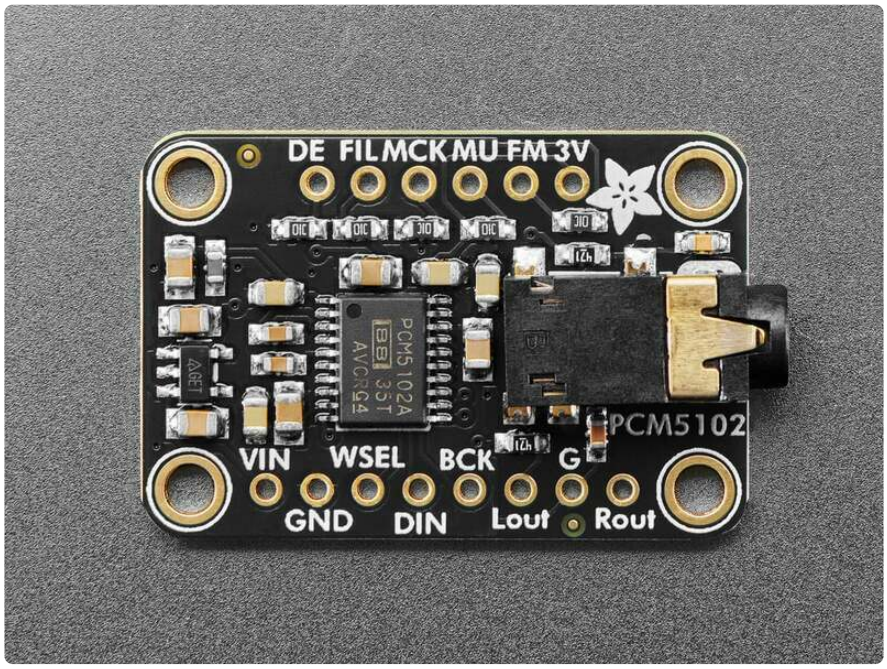


https://learn.adafruit.com/adafruit-pcm510x-i2s-dac

Last updated on 2025-04-02 12:01:01 PM EDT

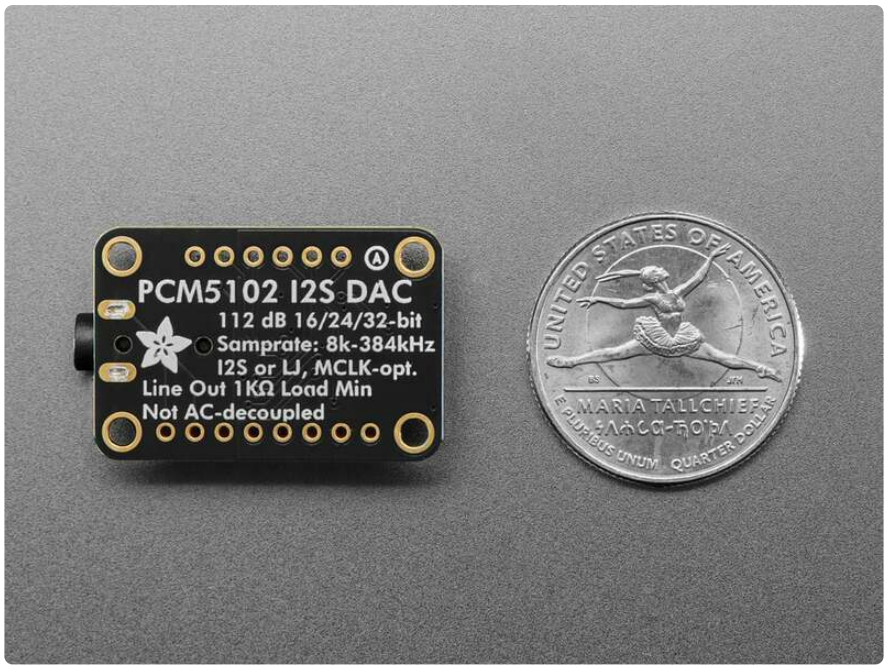
Table of Contents

Overview

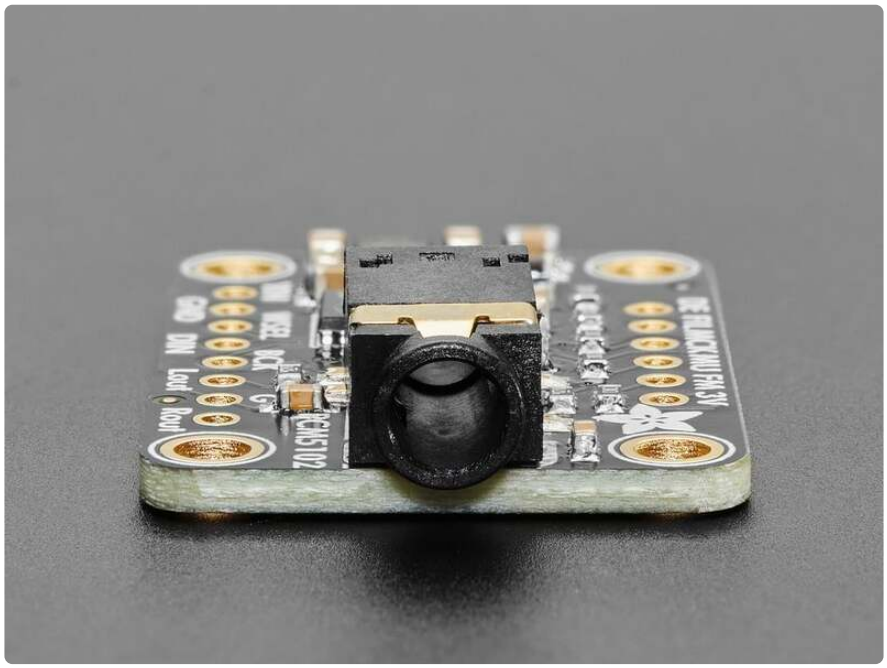


This guide is for both the PCM5100 and the PCM5102. Both boards have identical functionality, the only difference is the signal-to-noise/dynamic range (100dB for the PCM5100 and 112dB for the PCM5102).

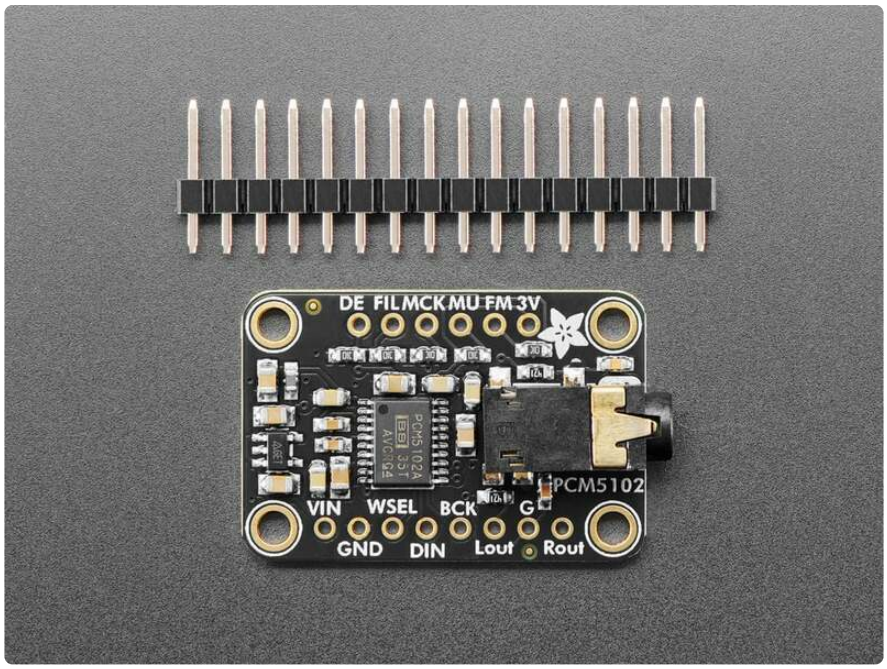
We stock a lot of chips and development boards that are able to do high quality digital I2S out, which makes for great quality audio playback. That's great when you have enough processing power to decode WAVs or MP3s in real time. However, we don't have a good selection of I2S DAC boards...until now! We really love the sounds coming out of the **Adafruit PCM5102 I2S DAC with Line Level Output** - it's got clean, excellent-quality stereo audio and does not need any MCLK or I2C configuration. Literally just pipe some I2S audio in and it will just work.



The PCM510x comes in a few different varieties: we've got the the good-quality **PCM5100** with 100dB signal-to-noise/dynamic range, and the excellent-quality **PCM5102** with 112dB. The '02 is a little more expensive but they are both quite good for any use and are pin-compatible so one may swap between the two.



This breakout makes I2S digital audio easy: all you need to do is power it with 3.3 to 5VDC, and provide BCLK (bit clock), WSEL (left/right word select), and DIN (data in). The data lines are 3.3V logic only. By default it's configured for I2S but you can also do Left-Justified by toggling the Format pin. Audio can be 16, 24 or 32-bit wide, the chip will automagically determine the right format from the WSEL / BCLK ratio. No MCLK pin is needed, the chip will auto-generate it internally from the bit clock - or you can provide it on the MCLK input if you want.

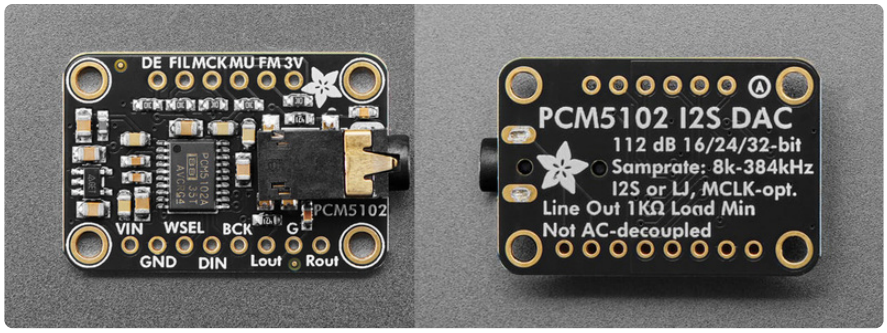


Other breakout pads provide: filtering (change from normal to low-latency by pulling high), mute (pull low to quickly set the outputs to ground), and de-emphasis for 44.1khz audio (default is off). The audio outputs are also available on breakout pads if you want to wire directly without using the 3.5mm jack.

Audio output is not AC-coupled because it is centered on ground: you can plug it into anything that is either AC coupled or has the same ground reference. Note that this is a line-level output, it cannot drive headphones - the output is for no less than 1K ohm loads!

Each order comes with one I2S Stereo DAC breakout and some header you can solder on for breadboard usage.

Pinouts



This guide is for both the PCM5100 and the PCM5102. Both boards have identical functionality, the only difference is the signal-to-noise/dynamic range (100dB for the PCM5100 and 112dB for the PCM5102).

Power Pins

- **VIN** - this is the power pin. It can be powered with 3.3 to 5VDC, however, the data lines are 3.3V logic only.
- **3V** - this is the 3.3V output from the onboard regulator
- **GND** - common ground for power and logic

The data lines are 3.3V logic level only

I2S Pins

- **WSEL** (Word Select or Left/Right Clock) - this is the pin that tells the DAC when the data is for the left channel and when it's for the right channel.
- **DIN** (Data In) - This is the pin that has the actual data coming in, both left and right data are sent on this pin, the WSEL pin indicates when left or right is being transmitted.
- **BCK** (Bit Clock) - This is the pin that tells the amplifier when to read data on the data pin.
- **MCK** (Main clock, optional) - This pin is optional for the PCM510x because it will auto-generate the main clock internally from BCK.

Audio Output

The audio output from this breakout is line-level. It is not AC-coupled because it is centered on ground. You can plug it into anything that is either AC coupled or has the same ground reference.

- **Lout** - This is the left channel audio output
- **Rout** - This is the right channel audio output
- **G** - This is a clean analog ground signal for the audio output
- **3.5mm output jack** - This is the onboard output audio jack. Note that it cannot drive headphones - the output is for no less than 1K ohm loads!

Note that it cannot drive headphones - the output is for no less than 1K ohm loads!

Control Pins

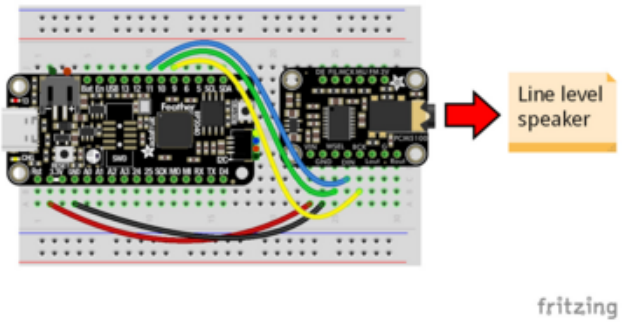
- **DE** - This is the de-emphasis pin for 44.1khz audio. By default it is off.
- **FIL** - This is the filter pin. You can change the filter from normal to low-latency by pulling the pin high.
- **MU** - This is the mute pin. You can pull this pin low to set the outputs to ground.
- **FM** - This is the format pin. You can change the format from I2S to Left justified by pulling the pin high.

CircuitPython

It's easy to use the **PCM510x I2S DAC** with CircuitPython and the builtin `audiobusio` module. This module allows you to easily write Python code that lets you play audio.

CircuitPython Microcontroller Wiring

First wire up the DAC to your board exactly as follows. The following is the DAC wired to a Feather RP2040:



Board 3.3V to **DAC VIN (red wire)**
Board GND to **DAC GND (black wire)**
Board D9 to **DAC BCK (yellow wire)**
Board D10 to **DAC WSEL (green wire)**
Board D11 to **DAC DIN (blue wire)**
DAC 3.5mm output to **line-level speaker**

CircuitPython Tone Playback

Click the **Download Project Bundle** button below to download the **code.py** file in a zip file. Extract the contents of the zip, and copy the **code.py** file to your **CIRCUITPY** drive.

```
# SPDX-FileCopyrightText: 2021 Kattni Rembor for Adafruit Industries
# SPDX-License-Identifier: MIT
"""
CircuitPython I2S Tone playback example.
Plays a tone for one second on, one
second off, in a loop.
"""
import time
import array
import math
import audiocore
import board
import audiobusio

audio = audiobusio.I2SOut(board.D9, board.D10, board.D11)

tone_volume = 0.1  # Increase this to increase the volume of the tone.
frequency = 440  # Set this to the Hz of the tone you want to generate.
length = 8000 // frequency
sine_wave = array.array("h", [0] * length)
for i in range(length):
    sine_wave[i] = int((math.sin(math.pi * 2 * i / length)) * tone_volume * (2 ** 15
- 1))
sine_wave_sample = audiocore.RawSample(sine_wave)

while True:
    audio.play(sine_wave_sample, loop=True)
    time.sleep(1)
    audio.stop()
    time.sleep(1)
```

Once successfully copied, you'll begin hearing a one second 440Hz tone, every other second.

WAV Playback

Click the **Download Project Bundle** button below to download the **code.py** file and the **StreetChicken.wav** file in a zip file. Extract the contents of the zip, and copy the **code.py** and **StreetChicken.wav** files to your **CIRCUITPY** drive.

Your **CIRCUITPY** drive contents should resemble the following.


```
# SPDX-FileCopyrightText: 2021 Kattni Rembor for Adafruit Industries
# SPDX-License-Identifier: MIT
"""
CircuitPython I2S WAV file playback.
Plays a WAV file once.
"""
```

```
import audiocore
import board
import audiobusio

audio = audiobusio.I2SOut(board.D9, board.D10, board.D11)

with open("StreetChicken.wav", "rb") as wave_file:
    wav = audiocore.WaveFile(wave_file)

    print("Playing wav file!")
    audio.play(wav)
    while audio.playing:
        pass

print("Done!")
```

Once successfully copied, you'll hear the Street Chicken WAV file play once.

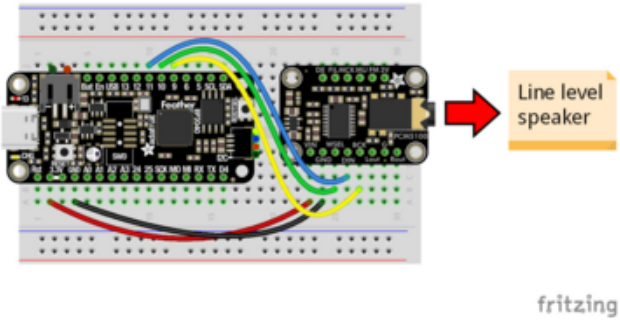
Arduino

Using the PCM510x I2S DAC with Arduino involves wiring up the DAC and running the provided example code.

Wiring

You can power the DAC with 3.3 to 5VDC, however, the data lines are 3.3V logic only. You'll want to use a 3.3V logic level board.

Here is an Adafruit Feather RP2040 wired up to the DAC:



Board 3.3V to **DAC VIN (red wire)**
Board GND to **DAC GND (black wire)**
Board D9 to **DAC BCK (yellow wire)**
Board D10 to **DAC WSEL (green wire)**
Board D11 to **DAC DIN (blue wire)**
DAC 3.5mm output to **line-level speaker**

No additional libraries are required for these examples.

Tone Example Code

```
// SPDX-FileCopyrightText: 2016 Sandeep Mistry
// SPDX-FileCopyrightText: 2022 Earle F. Philhower, III
// SPDX-FileCopyrightText: 2023 Ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/*
  This example generates a square wave based tone at a specified frequency
  and sample rate. Then outputs the data using the I2S interface to a
  MAX08357 I2S Amp Breakout board.

  created 17 November 2016
  by Sandeep Mistry
  modified for RP2040 by Earle F. Philhower, III <earlephilhower@yahoo.com>


    bool setBCLK(pin_size_t pin);
    - This assigns two adjacent pins - the pin after this one (one greater)
      is the WS (word select) signal, which toggles before the sample for
      each channel is sent

    bool setDATA(pin_size_t pin);
    - Sets the DOUT pin, can be any valid GPIO pin
*/

#include <I2S.h>

// Create the I2S port using a PIO state machine
I2S i2s(OUTPUT);

// GPIO pin numbers on Feather RP2040
#define pBCLK D9 // BITCLOCK
#define pWS   D10 // LRCLOCK
#define pDOUT D11 // DATA

const int frequency = 440; // frequency of square wave in Hz
const int amplitude = 500; // amplitude of square wave
const int sampleRate = 16000; // 16 KHz is a good quality

const int halfWavelength = (sampleRate / frequency); // half wavelength of square
wave

int16_t sample = amplitude; // current sample value
int count = 0;

void setup() {
  Serial.begin(115200);
  while (!Serial) delay(10);
  Serial.println("I2S simple tone");

  i2s.setBCLK(pBCLK);
  i2s.setDATA(pDOUT);
  i2s.setBitsPerSample(16);

  // start I2S at the sample rate with 16-bits per sample
  if (!i2s.begin(sampleRate)) {
    Serial.println("Failed to initialize I2S!");
    while (1); // do nothing
  }

}

void loop() {
  if (count % halfWavelength == 0) {
    // invert the sample every half wavelength count multiple to generate square
wave
```

```
    sample = -1 * sample;
  }

  // write the same sample twice, once for left and once for the right channel
  i2s.write(sample);
  i2s.write(sample);

  // increment the counter for the next sample
  count++;
}
```

Once you have uploaded the sketch to your board, you'll hear a tone.

You can change the frequency (pitch) of the tone by updating the `frequency` variable.

```
const int frequency = 440; // frequency of square wave in Hz
```

Audio Playback Example Code

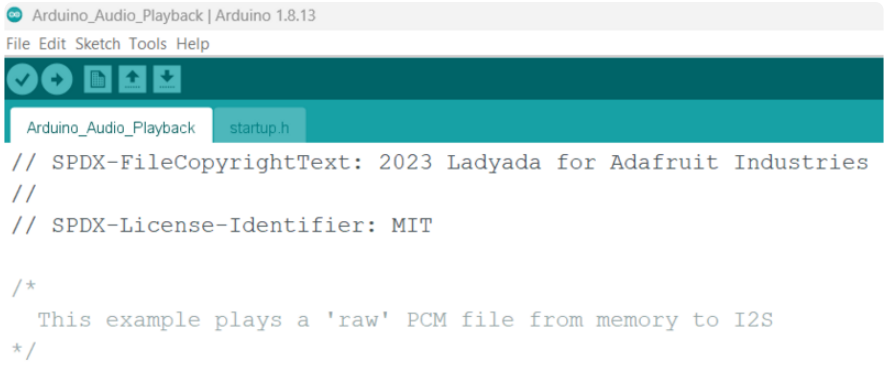
This example plays a PCM audio file when the Boot button is pressed. Check out this Python script to convert WAV files to PCM files (https://adafru.it/XmC).

Click the button below to download the source code and header file. Unzip it, and open it with the Arduino IDE.

Arduino_Audio_Playback.zip

https://adafru.it/1afA

When you open the code in the Arduino IDE, you will see the Arduino sketch code in one tab, and the header file in a second tab.



Once you've uploaded the sketch to your board, you'll hear the startup tone play in a loop with a short pause.

```
// SPDX-FileCopyrightText: 2023 Ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/*
```

```
  This example plays a 'raw' PCM file from memory to I2S
*/

#include <I2S.h>

#include "startup.h" // audio file in flash

// Create the I2S port using a PIO state machine
I2S i2s(OUTPUT);

// GPIO pin numbers on Feather RP2040
#define pBCLK D9 // BITCLOCK
#define pWS   D10 // LRCLOCK
#define pDOUT D11 // DATA

void setup() {
  Serial.begin(115200);
  while (!Serial) delay(10);
  Serial.println("I2S playback demo");
}

void loop() {
}

void setup1() {
  i2s.setBCLK(pBCLK);
  i2s.setDATA(pDOUT);
  i2s.setBitsPerSample(16);
}

void loop1() {
  // the main loop will tell us when it wants us to play!
    play_i2s(startupAudioData, sizeof(startupAudioData), startupSampleRate);
    delay(1000);
}

void play_i2s(const uint8_t *data, uint32_t len, uint32_t rate) {
  // start I2S at the sample rate with 16-bits per sample
  if (!i2s.begin(rate)) {
    Serial.println("Failed to initialize I2S!");
    delay(500);
    i2s.end();
    return;
  }

  for(uint32_t i=0; i<len; i++) {
    uint16_t sample = (uint16_t)data[i] << 6; // our data is 10 bit but we want 16
bit so we add some gain
    // write the same sample twice, once for left and once for the right channel
    i2s.write(sample);
    i2s.write(sample);
  }
  i2s.end();
}
```

Downloads

Files

- PCM510x Datasheet (https://adafru.it/1afB)
- EagleCAD PCB Files on GitHub (https://adafru.it/1afC)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/1afD)

Schematic and Fab Print