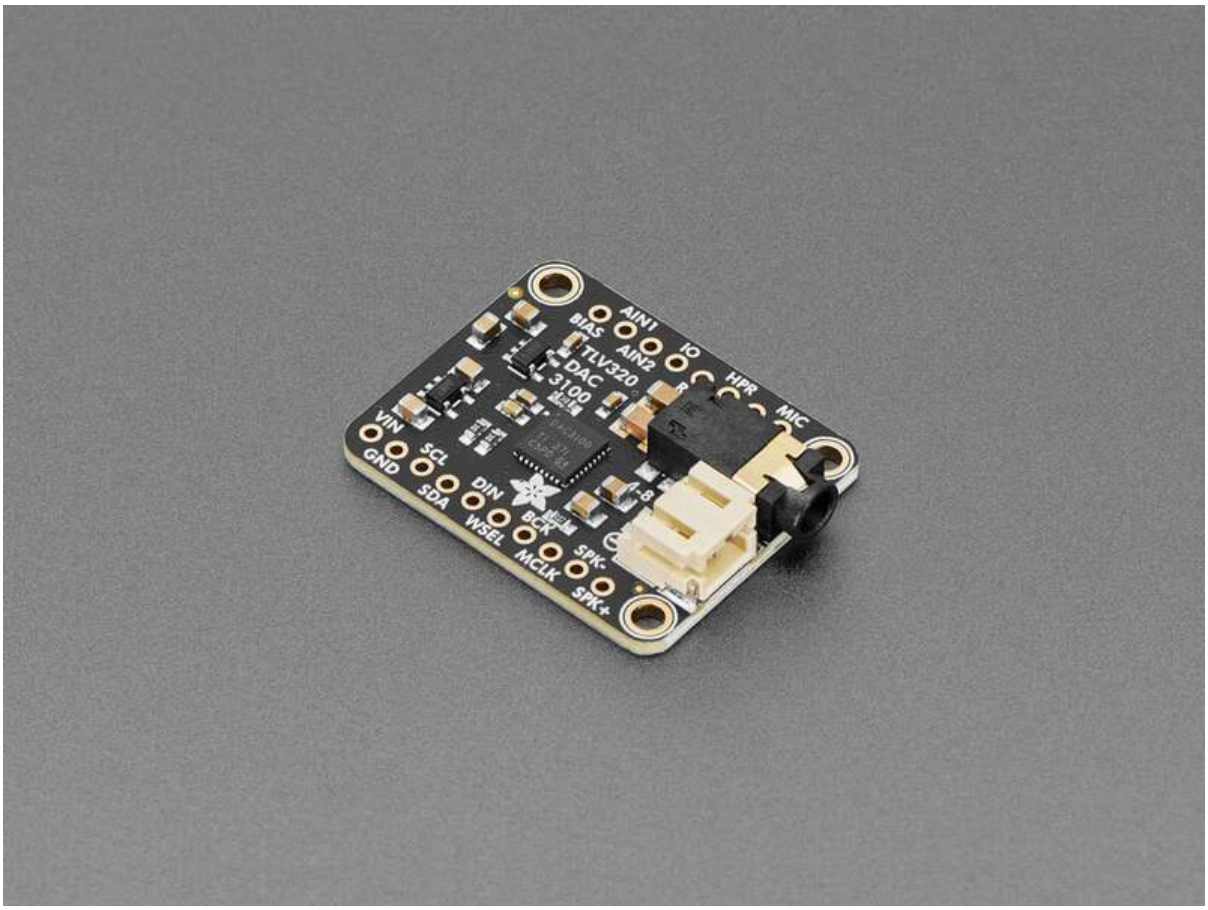
Adafruit TLV320DAC3100 I2S DAC

Created by Liz Clark

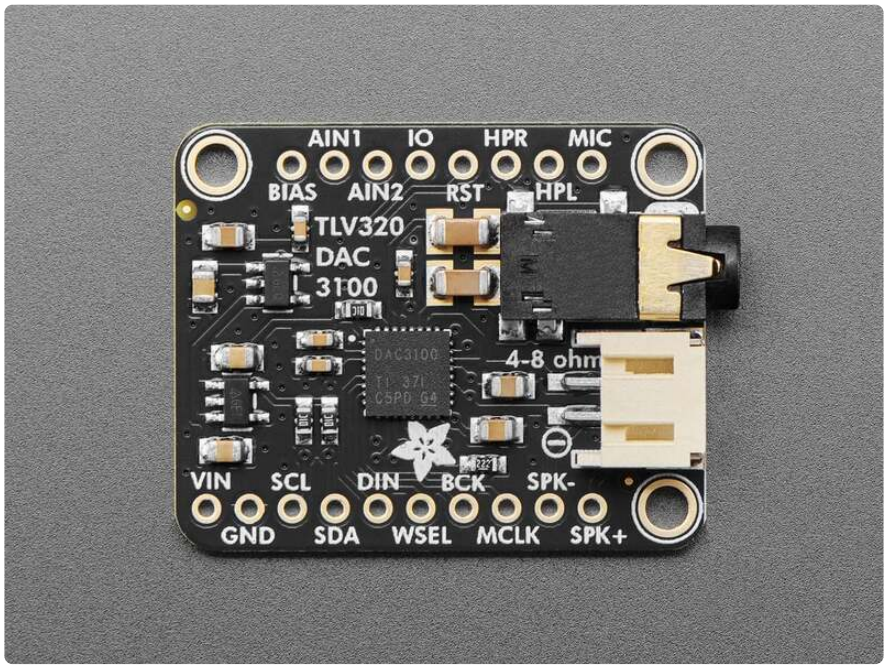


https://learn.adafruit.com/adafruit-tlv320dac3100-i2s-dac

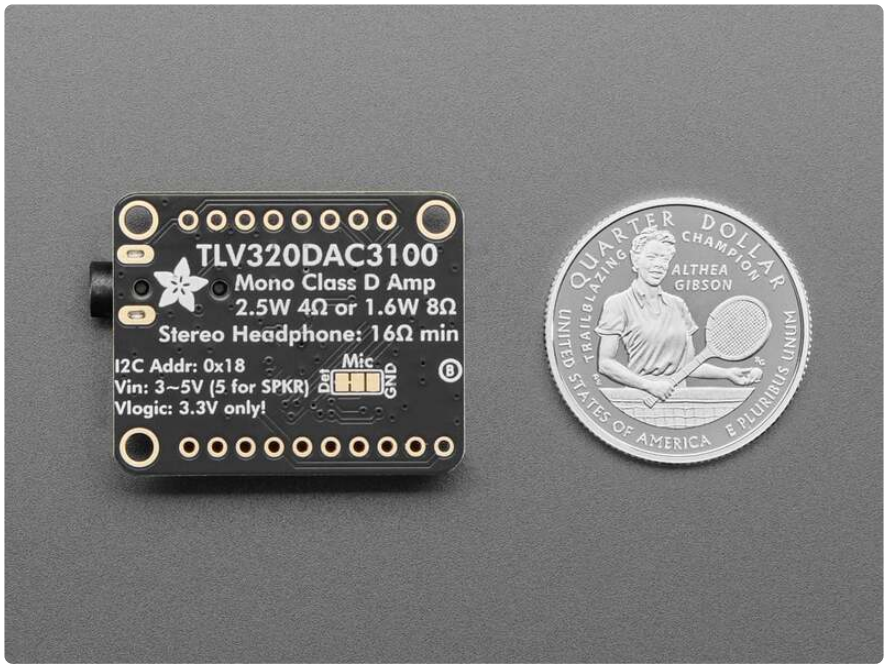
Last updated on 2025-09-25 09:44:45 AM EDT

Table of Contents

Overview

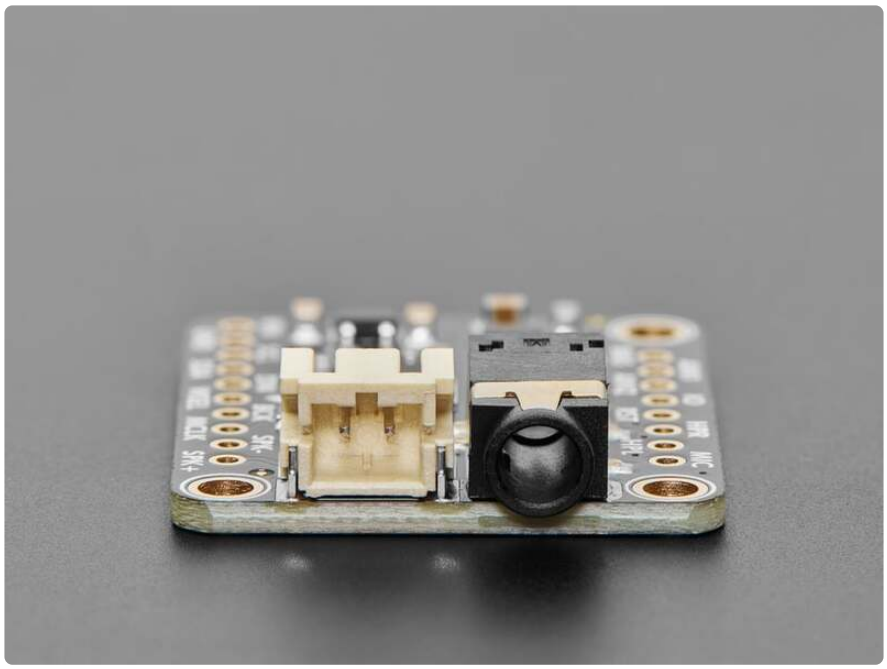


We stock a lot of chips and development boards that are able to do high quality digital I2S out, which makes for great quality audio playback. That's great when you have enough processing power to decode WAVs or MP3s in real time. However, most give you stereo line-out which cannot connect to headphones...until now!



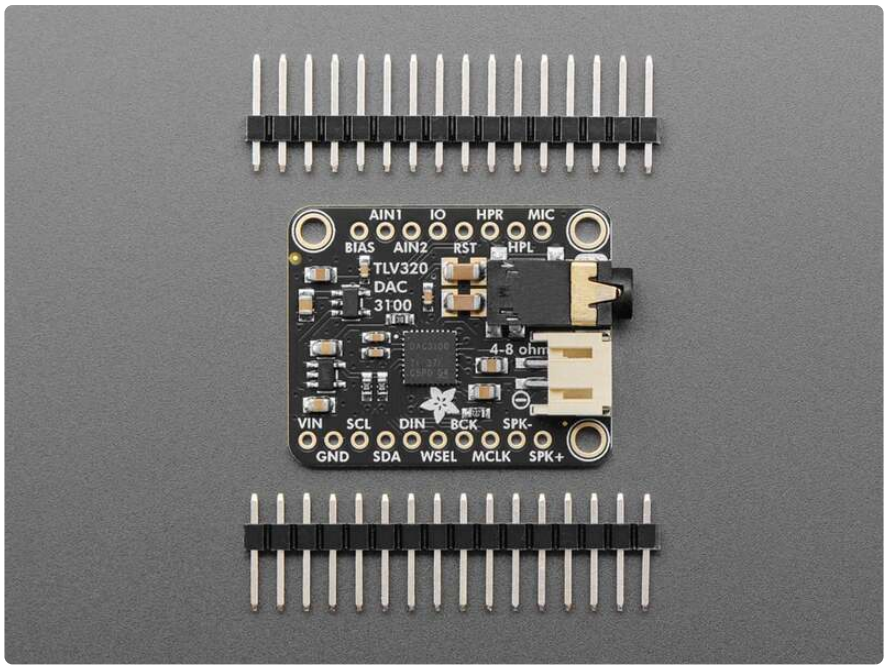
We really love the sounds coming out of the **Adafruit TLV320DAC3100 I2S DAC with Stereo Headphone and Mono Speaker output** - it's got clean, excellent-quality, stereo audio that can connect directly to your 16Ω headphones and/or a 4Ω-8Ω speaker. This makes it excellent for all-in-one audio projects without needing an external

amplifier. Please note that while it does not need a MCLK signal (you can configure it to use BCLK as the PLL input) it does require I2C configuration! You will need a microcontroller with our library (Arduino, CircuitPython or Python) to set up the board for audio playback.

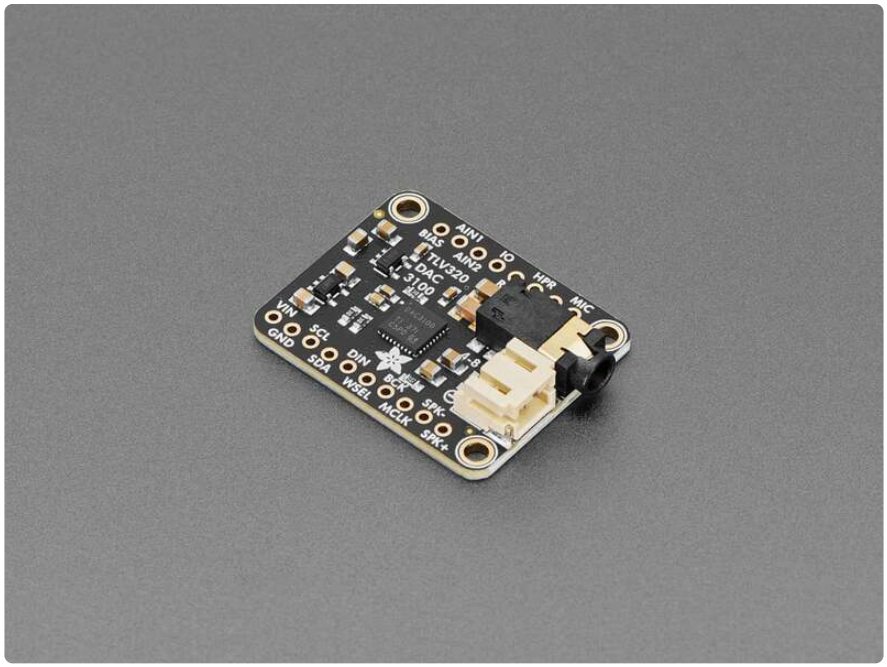


This breakout makes amplified I2S digital audio easy. You can power it with 3V (headphone only) or 5VDC (for speaker support) and provide BCLK (bit clock), WSEL (left/right word select), and DIN (data in). Then configure the board with I2C to determine the gain and which output you want activated. There's tons of configuration options available, but we've used it mostly for 16-bit I2S audio. There's a built-in PLL that will generate an MCLK signal from BCLK for you, so it can be used by any I2S source.

Audio output from the headphone is AC-coupled. Audio from the speaker is a class-D amplifier and must be connected to a speaker only. If you need an external amplifier, use the headphone jack.



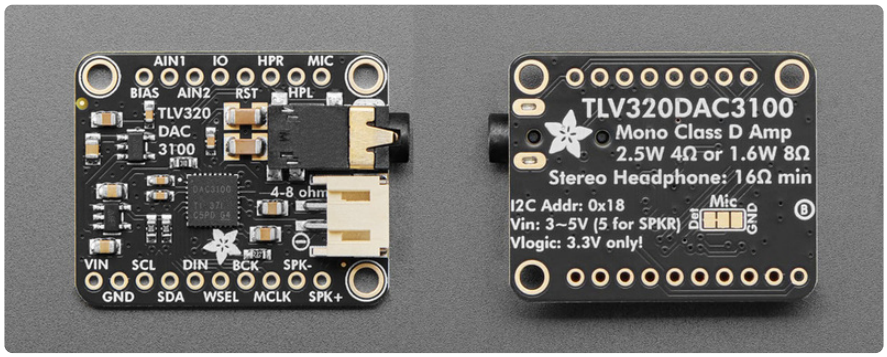
There's a few extra breakouts on this board: MIC and BIAS are connected to the 'fourth' contact on headsets that often have a microphone. You can configure the amp to provide a 2V bias voltage which will let you detect when a headset+mic is plugged in, and also detect when the headset button is pressed. There's also AIN1/AIN2 which are alternative mix-ins for the audio outputs, not I2S encoders. There is one 'GPIO' pin which can also be used as an IRQ line.



Note that this board can be powered from 3~5VDC but all logic level is 3.3V only (it's quite rare for an I2S microcontroller/computer to have 5V logic!)

Each order comes with one Adafruit TLV320DAC3100 DAC breakout and some header you can solder on for breadboard usage.

Pinouts



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. If you want to use the speaker output, you need to provide 5V.
- **GND** - common ground for power and logic

The board needs to be powered with 5V if you are using the speaker output.

I2C Logic Pins

This I2S DAC **requires I2C configuration!** You will need a microcontroller with our library (Arduino, CircuitPython or Python) to set up the board for audio playback.

- **SCL** - I2C clock pin, connect to your microcontroller's I2C clock line. There's a **10K pullup** on this pin.
- **SDA** - I2C data pin, connect to your microcontroller's I2C data line. There's a **10K pullup** on this pin.

This I2S DAC requires I2C configuration! You will need a microcontroller with our library (Arduino, CircuitPython or Python) to set up the board for audio playback.

I2S Pins

- **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.
- **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.
- **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 TLV320DAC3100 because you can configure the I2C driver to use BCK as the PLL input.

Audio Output

You'll want to disable speaker output when using the headphone output for the best possible sound quality. The class-D amplifier for the speaker can cause some noise on the headphone output.

Speaker

- **SPK+** - This is the speaker positive output
- **SPK-** - This is the speaker negative output
- The speaker output is also available via the **JST-PH port**

Headphone

- **HPR** - This is the right channel headphone output
- **HPL** - This is the left channel headphone output
- **3.5mm output jack** - This is the onboard headphone jack. It can provide stereo audio that can connect directly to your 16Ω headphones.

Disable the speaker output when using the headphone output for the best possible sound quality.

Additional Pins

- **MIC** and **BIAS** - **MIC** is the mic detection pin and **BIAS** is the microphone bias pin. These pins are connected to the 'fourth' contact on the 3.5mm output jack. The fourth contact is for headsets that often have a microphone. You can configure the amp to provide a 2V bias voltage which will let you detect when a headset with a mic is plugged in, and also detect when the headset button is pressed.

- **AIN1** and **AIN2** - These pins are alternative mix-ins for the audio outputs (not I2S encoders).
- **IO** - This is an additional GPIO pin that can be used as an interrupt line.
- **RST** - This is the reset pin. The DAC needs to be reset before use by toggling this pin from low to high.

Mic Detect Jumper

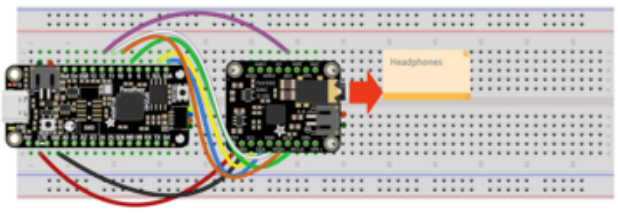
On the back of the board is the mic detect jumper. By default, the mic contact from the headphone jack is connected to the mic detect pin on the DAC. If you cut the jumper, you can disconnect the mic contact from mic detect and then solder the center pad to the GND pad.

CircuitPython

It's easy to use the **TLV320DAC3100** with CircuitPython, and the Adafruit_CircuitPython_TLV320 (https://adafru.it/1ahn) module. This module allows you to easily write Python code to configure this I2S DAC.

CircuitPython Microcontroller Wiring

First wire up the I2S DAC to your board exactly as follows. The following is the DAC wired to a Feather RP2040 with the headphone output:



Board 3.3V to **DAC VIN (red wire)**
Board GND to **DAC GND (black wire)**
Board SCL to **DAC SCL (yellow wire)**
Board SDA to **DAC SDA (blue wire)**
Board D9 to **DAC BCK (green wire)**
Board D10 to **DAC WSEL (orange wire)**
Board D11 to **DAC DIN (white wire)**
DAC 3.5mm output to **headphones**
Board D12 to **DAC RST (purple wire)**

If you want to use speaker output, you'll need to power the DAC with 5V.

CircuitPython Usage

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_TLV320** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folder and file:

- **adafruit_bus_device/**
- **adafruit_tlv320.mpy**

The DAC must be reset before initialization by toggling the RST pin to LOW and then back to HIGH.

CircuitPython Tone Playback

```
# SPDX-FileCopyrightText: 2025 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import array
import math
import time

import audiobusio
import audiocore
import board
import digitalio

import adafruit_tlv320
```

```
# Reset the DAC before use
reset_pin = digitalio.DigitalInOut(board.D12)
reset_pin.direction = digitalio.Direction.OUTPUT
reset_pin.value = False  # Set low to reset
time.sleep(0.1)  # Pause 100ms
reset_pin.value = True  # Set high to release from reset

i2c = board.I2C()
dac = adafruit_tlv320.TLV320DAC3100(i2c)

# set sample rate & bit depth, use bclk
dac.configure_clocks(sample_rate=44100, bit_depth=16)

# use headphones
dac.headphone_output = True
dac.dac_volume = -10  # dB
# or use speaker
# dac.speaker_output = True
# dac.speaker_volume = -20 # dB

if "I2S_BCLK" and "I2S_WS" in dir(board):
    audio = audiobusio.I2SOut(board.I2S_BCLK, board.I2S_WS, board.I2S_DIN)
else:
    audio = audiobusio.I2SOut(board.D9, board.D10, board.D11)
# generate a sine wave
tone_volume = 0.5
frequency = 440
sample_rate = dac.sample_rate
length = sample_rate // 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, sample_rate=sample_rate)

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

Once the code starts running, you'll begin hearing a one second 440Hz tone, every other second.

WAV Playback

Download and copy this WAV audio file to your **CIRCUITPY** drive.

StreetChicken.wav

https://adafru.it/1aho

Then, update the **code.py** file with the code below:

```
# SPDX-FileCopyrightText: 2025 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import audiobusio
import audiocore
import board
```

```
import adafruit_tlv320

i2c = board.I2C()
dac = adafruit_tlv320.TLV320DAC3100(i2c)

# set sample rate & bit depth, use bclk
dac.configure_clocks(sample_rate=44100, bit_depth=16)

# use headphones
dac.headphone_output = True
dac.dac_volume = -15  # dB
# or use speaker
# dac.speaker_output = True
# dac.speaker_volume = -10 # dB

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 the code starts running, you'll hear the Street Chicken WAV file play once.

Python Docs

Python Docs (https://adafru.it/1agr)

Arduino

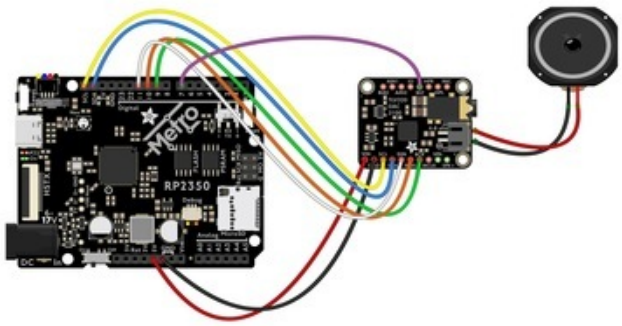
Using the TLV320DAC3100 breakout with Arduino involves wiring up the breakout to your Arduino-compatible microcontroller, installing the Adafruit_TLV320_I2S (https://adafru.it/1ahp) library, and running the provided example code.

Wiring

You can power the I2S 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.

Additionally, if you are using the speaker output, you'll want to power the board with 5V.

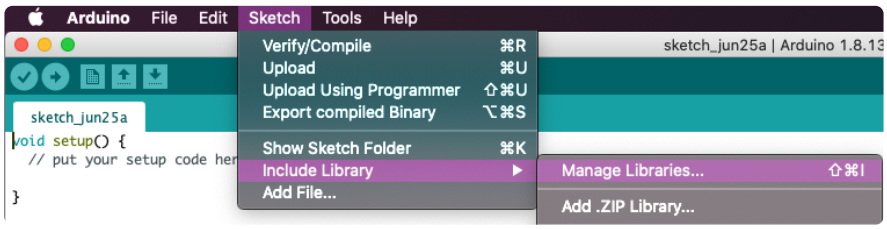
Here is an Adafruit Metro RP2040 wired up to the DAC for speaker output:



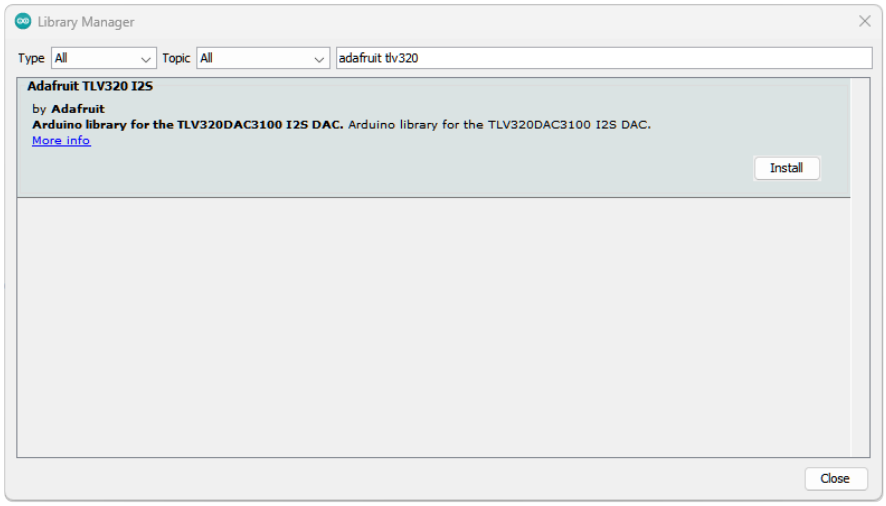
Board 5V to **DAC VIN (red wire)**
Board GND to **DAC GND (black wire)**
Board SCL to **DAC SCL (yellow wire)**
Board SDA to **DAC SDA (blue wire)**
Board D9 to **DAC BCK (green wire)**
Board D10 to **DAC WSEL (orange wire)**
Board D11 to **DAC DIN (white wire)**
Board D7 to **DAC RST (purple wire)**
DAC JST-PH to **speaker (red and black wires)**

Library Installation

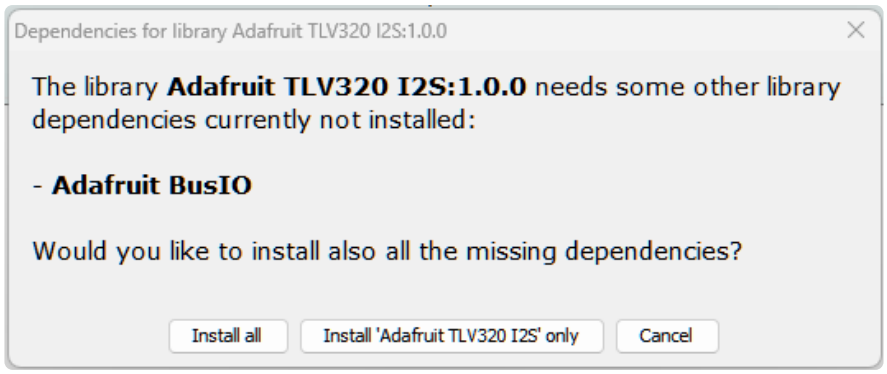
You can install the **Adafruit_TLV320_I2S** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **Adafruit_TLV320_I2S**, and select the **Adafruit TLV320 I2S** library:



If asked about dependencies, click "Install all".



If the "Dependencies" window does not come up, then you already have the dependencies installed.

If the dependencies are already installed, you must make sure you update them through the Arduino Library Manager before loading the example!

Sine Tone Example

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

#include <Adafruit_TLV320DAC3100.h>
#include <I2S.h>
#include <math.h>

Adafruit_TLV320DAC3100 codec;  // Create codec object

#define pBCLK D9   // BITCLOCK - I2S clock
#define pWS   D10  // LRCLOCK - Word select
#define pDOUT D11  // DATA - I2S data

// Create I2S port
I2S i2s(OUTPUT);

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("\n\nTLV320DAC3100 Sine Tone Test");
  // Start I2C communication with codec
  Serial.println("Initializing codec...");
  if (!codec.begin()) {
```

```
    Serial.println("Failed to initialize codec!");
  }
  codec.reset();
  // Step 1: Set codec interface to I2S with 16-bit data
  Serial.println("Configuring codec interface...");
  if (!codec.setCodecInterface(TLV320DAC3100_FORMAT_I2S, TLV320DAC3100_DATA_LEN_16))
{
    Serial.println("Failed to configure codec interface!");
  }

  // Step 2: Configure clock - using PLL with BCLK as input
  Serial.println("Configuring codec clocks...");
  if (!codec.setCodecClockInput(TLV320DAC3100_CODEC_CLKIN_PLL) ||
      !codec.setPLLClockInput(TLV320DAC3100_PLL_CLKIN_BCLK)) {
    Serial.println("Failed to configure codec clocks!");
  }

  // Step 3: Set up PLL - these values work well for 44.1kHz sample rate
  if (!codec.setPLLValues(1, 1, 8, 0)) {
    Serial.println("Failed to configure PLL values!");
  }

  // Step 4: Configure DAC dividers
  if (!codec.setNDAC(true, 8) ||
      !codec.setMDAC(true, 2) ||
      !codec.setDOSR(128)) {
    Serial.println("Failed to configure DAC dividers!");
  }

  // Step 5: Power up PLL
  if (!codec.powerPLL(true)) {
    Serial.println("Failed to power up PLL!");
  }

  // Step 6: Configure DAC path - power up both left and right DACs
  Serial.println("Configuring DAC path...");
  if (!codec.setDACDataPath(true, true,
                            TLV320_DAC_PATH_NORMAL,
                            TLV320_DAC_PATH_NORMAL,
                            TLV320_VOLUME_STEP_1SAMPLE)) {
    Serial.println("Failed to configure DAC data path!");
  }

  // Step 7: Route DAC output to headphone
  if (!codec.configureAnalogInputs(TLV320_DAC_ROUTE_MIXER, // Left DAC to mixer
                                   TLV320_DAC_ROUTE_MIXER, // Right DAC to mixer
                                   false, false, false,    // No AIN routing
                                   false)) {               // No HPL->HPR
    Serial.println("Failed to configure DAC routing!");
  }

  // Step 8: Unmute DAC and set volume (higher for testing)
  Serial.println("Setting DAC volume...");
  if (!codec.setDACVolumeControl(
          false, false, TLV320_VOL_INDEPENDENT) || // Unmute both channels
      !codec.setChannelVolume(false, 18) ||        // Left DAC +0dB
      !codec.setChannelVolume(true, 18)) {         // Right DAC +0dB
    Serial.println("Failed to configure DAC volume control!");
  }


  if (!codec.setChannelVolume(false, 12.0) ||
      !codec.setChannelVolume(true, 12.0)) {
    Serial.println("Failed to set DAC channel volumes!");
  }

  if (!codec.enableSpeaker(true) ||                // Dis/Enable speaker amp
      !codec.configureSPK_PGA(TLV320_SPK_GAIN_6DB, // Set gain to 6dB
                              true) ||             // Unmute
```

```
      !codec.setSPKVolume(true, 0)) { // Enable and set volume to 0dB
    Serial.println("Failed to configure speaker output!");
  }

  // Initialize I2S peripheral
  Serial.println("Initializing I2S...");
  i2s.setBCLK(pBCLK);
  i2s.setDATA(pDOUT);
  i2s.setBitsPerSample(16);

  // Start I2S at the sample rate
  if (!i2s.begin(sampleRate)) {
    Serial.println("Failed to initialize I2S!");
  }

}

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++;
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the TLV320DAC3100 recognized over I2C. Then, you'll hear a sine wave play through the headphone or speaker output.

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) (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.

TLV320_Audio_Playback_Arduino.zip

https://adafru.it/1ahq

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 <Adafruit_TLV320DAC3100.h>
#include <I2S.h>
#include <math.h>

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

Adafruit_TLV320DAC3100 codec;  // Create codec object

// 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() {
  Serial.begin(115200);

  while (!Serial) delay(10);
  Serial.println("\n\nTLV320DAC3100 Sine Tone Test");
  // Start I2C communication with codec
  Serial.println("Initializing codec...");
  if (!codec.begin()) {
    Serial.println("Failed to initialize codec!");
  }
  codec.reset();
  // Step 1: Set codec interface to I2S with 16-bit data
  Serial.println("Configuring codec interface...");
  if (!codec.setCodecInterface(TLV320DAC3100_FORMAT_I2S, TLV320DAC3100_DATA_LEN_16))
{
    Serial.println("Failed to configure codec interface!");
  }
```

```
// Step 2: Configure clock - using PLL with BCLK as input
Serial.println("Configuring codec clocks...");
if (!codec.setCodecClockInput(TLV320DAC3100_CODEC_CLKIN_PLL) ||
    !codec.setPLLClockInput(TLV320DAC3100_PLL_CLKIN_BCLK)) {
  Serial.println("Failed to configure codec clocks!");
}

// Step 3: Set up PLL - these values work well for 44.1kHz sample rate
if (!codec.setPLLValues(1, 1, 8, 0)) {
  Serial.println("Failed to configure PLL values!");
}

// Step 4: Configure DAC dividers
if (!codec.setNDAC(true, 8) ||
    !codec.setMDAC(true, 2) ||
    !codec.setDOSR(128)) {
  Serial.println("Failed to configure DAC dividers!");
}

// Step 5: Power up PLL
if (!codec.powerPLL(true)) {
  Serial.println("Failed to power up PLL!");
}

// Step 6: Configure DAC path - power up both left and right DACs
Serial.println("Configuring DAC path...");
if (!codec.setDACDataPath(true, true,
                          TLV320_DAC_PATH_NORMAL,
                          TLV320_DAC_PATH_NORMAL,
                          TLV320_VOLUME_STEP_1SAMPLE)) {
  Serial.println("Failed to configure DAC data path!");
}

// Step 7: Route DAC output to headphone
if (!codec.configureAnalogInputs(TLV320_DAC_ROUTE_MIXER, // Left DAC to mixer
                                 TLV320_DAC_ROUTE_MIXER, // Right DAC to mixer
                                 false, false, false,    // No AIN routing
                                 false)) {               // No HPL->HPR
  Serial.println("Failed to configure DAC routing!");
}

// Step 8: Unmute DAC and set volume (higher for testing)
Serial.println("Setting DAC volume...");
if (!codec.setDACVolumeControl(
        false, false, TLV320_VOL_INDEPENDENT) || // Unmute both channels
    !codec.setChannelVolume(false, 18) ||        // Left DAC +0dB
    !codec.setChannelVolume(true, 18)) {         // Right DAC +0dB
  Serial.println("Failed to configure DAC volume control!");
}


if (!codec.setChannelVolume(false, 12.0) ||
    !codec.setChannelVolume(true, 12.0)) {
  Serial.println("Failed to set DAC channel volumes!");
}

if (!codec.enableSpeaker(true) ||                // Dis/Enable speaker amp
    !codec.configureSPK_PGA(TLV320_SPK_GAIN_6DB, // Set gain to 6dB
                            true) ||             // Unmute
    !codec.setSPKVolume(true, 0)) { // Enable and set volume to 0dB
  Serial.println("Failed to configure speaker output!");
}

// Initialize I2S peripheral
Serial.println("Initializing I2S...");
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();
}
```

Arduino Docs

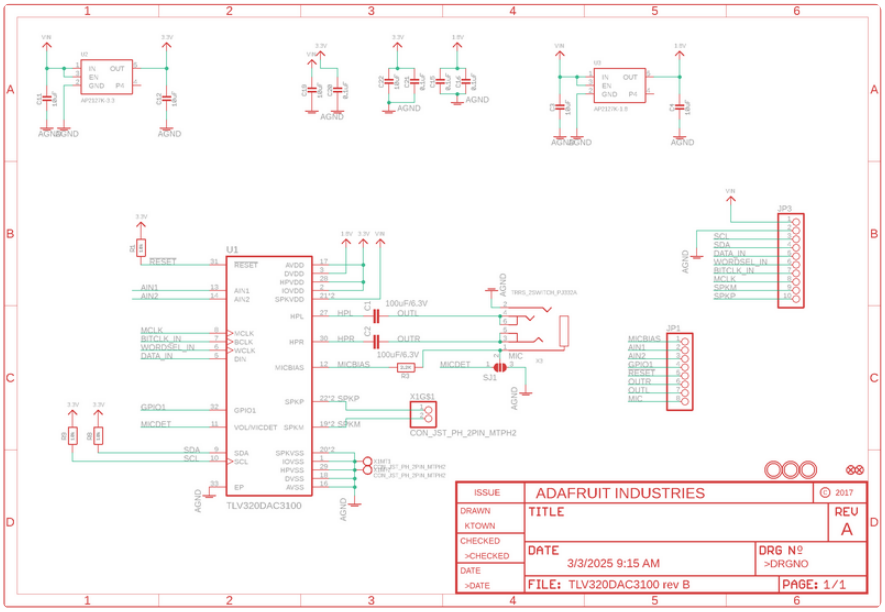
Arduino Docs (https://adafru.it/1ags)

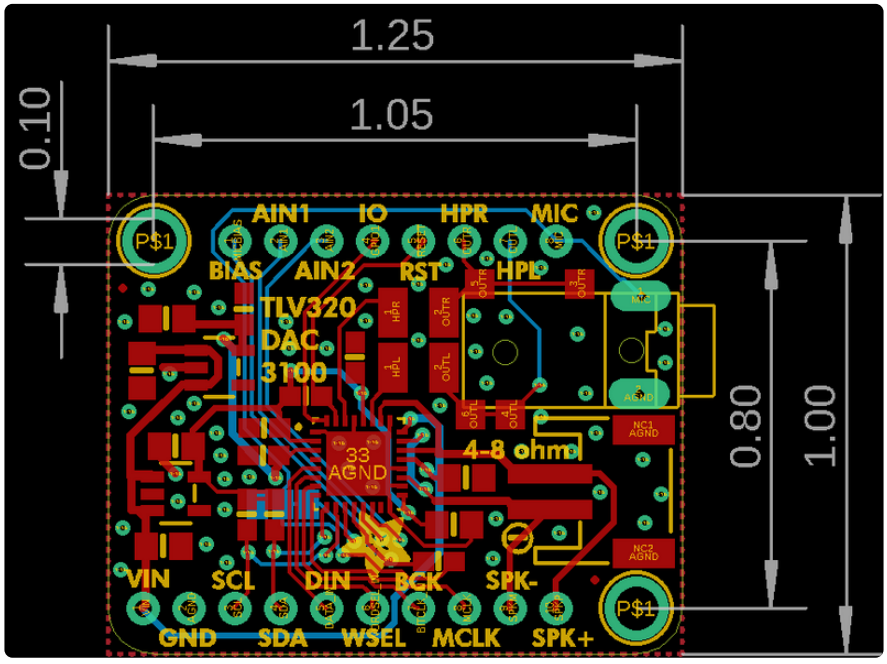
Downloads

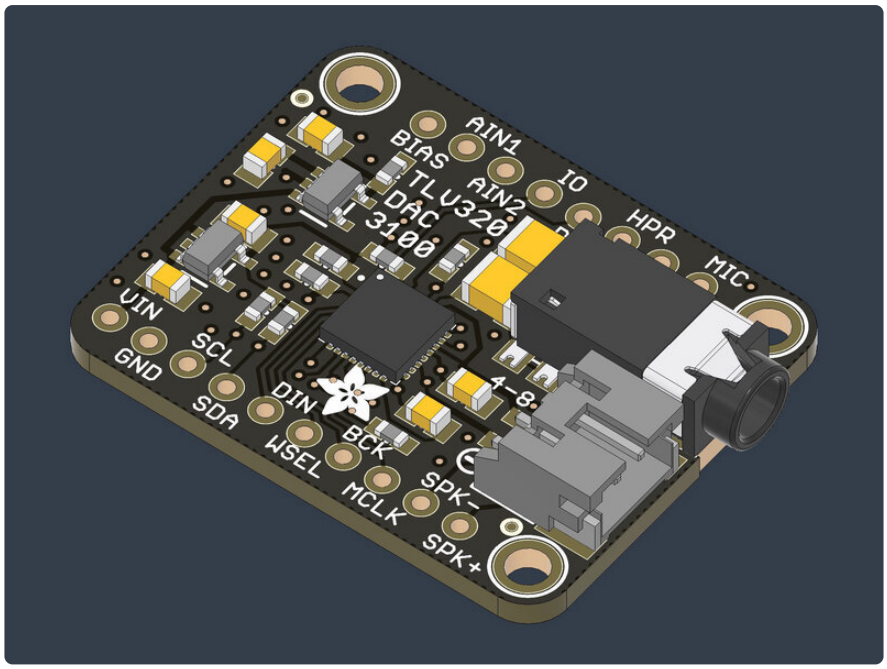
Files

- TLV320DAC3100 Datasheet (https://adafru.it/1ahr)
- EagleCAD PCB Files on GitHub (https://adafru.it/1ahs)
- 3D models on GitHub (https://adafru.it/1ahT)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/1aht)

Schematic and Fab Print






BIAS
AIN1
AIN2
IO
HPR
MIC
TLV320
DAC
3100
VIN
GND
SCL
SDA
DIN
WSEL
BCK
MCLK
SPK-
SPK+
4-8