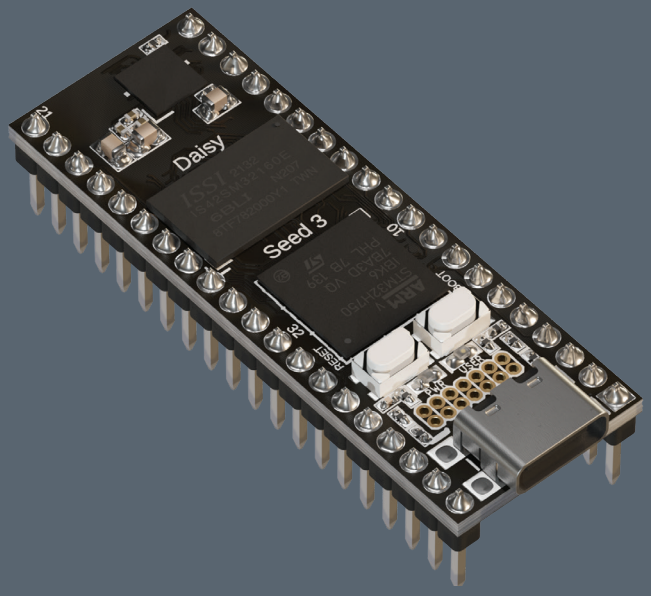
Seed3

Sound Computer



Features:

- Embedded platform for audio applications
- Up to192kHz / 32-bit audio hardware
- 64MB of SDRAM for up to 10 minute long audio buffers
- ARM Cortex-M7 MCU, running at 480MHz
- 31 total GPIO pins with configurable functionality
- 12-bit Digital to Analog Converters (x2)
- SD card interfaces
- PWM outputs
- Serial Protocols for connecting external sensors and devices (SPI, UART, I2s, I2C)
- Dedicated VIN pin for power
- USB-C port, and additional USB pins for full OTG-support as host and device
- Pin-to-pin compatible with the Seed pinout & footprint

Applications:

- Electronic Instruments (Eurorack modules, synthesizers, samplers, drum machines)
- Effects Units (Desktop Effects, Effects Pedals)
- Audio Playback (Sound Installations, Audio Feedback Devices)

Description:

Daisy is an embedded platform for audio. It features everything you need for creating high fidelity hardware devices. Just plug in a USB cable and start making sound!

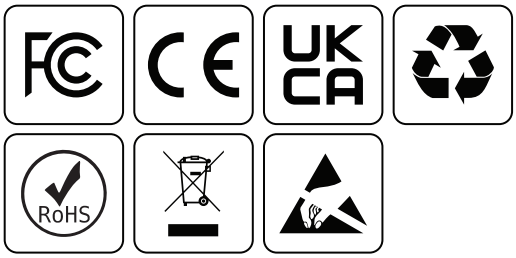
Programming the Daisy is a breeze with support for a number of languages including C++, Arduino, and Max/MSP Gen~. To get started, simply upload an example program over USB, and start tweaking!

Documentation, and examples are hosted on our GitHub repository for easy download. All firmware that we develop is released for free under a permissive open source license (MIT).

Table of Contents

Designed and Distributed by
Qu-Bit Electronix, Inc. DBA Daisy
1010 Calle Cordillera
Ste 103
San Clemente, CA 92673

daisy.audio



⚠ **Warning:**
Cancer and Reproductive Harm -
www.P65Warnings.ca.gov

FCC

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

1. This device may not cause harmful interference, and
2. this device must accept any interference received, including interference that may cause undesired operation.

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance of 20 cm between the radiator and your body.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

Reorient or relocate the receiving antenna. Increase the separation between the equipment and receiver. Connect the equipment into an outlet on a circuit different from that to which the receiver is connected. Consult the dealer or an experienced radio/TV technician for help.

FCC Certification.

See the complete FCC Test Report.

UKCA

This product conforms to the applicable requirements of the relevant UK Statutory Instruments and carries the UKCA mark accordingly.

CE

This product conforms to the essential requirements and other relevant provisions of the applicable European Union Directives and carries the CE marking accordingly.

Absolute Maximum Ratings - Table 1

PIN TYPE	MIN	MAX	UNIT
VIN Range	+4	+17	V
GPIO	0	+5	V
Audio Inputs	-1.8V	+1.8V	V

Audio inputs are AC coupled and 3.6Vpp, or approx. 1Vrms.

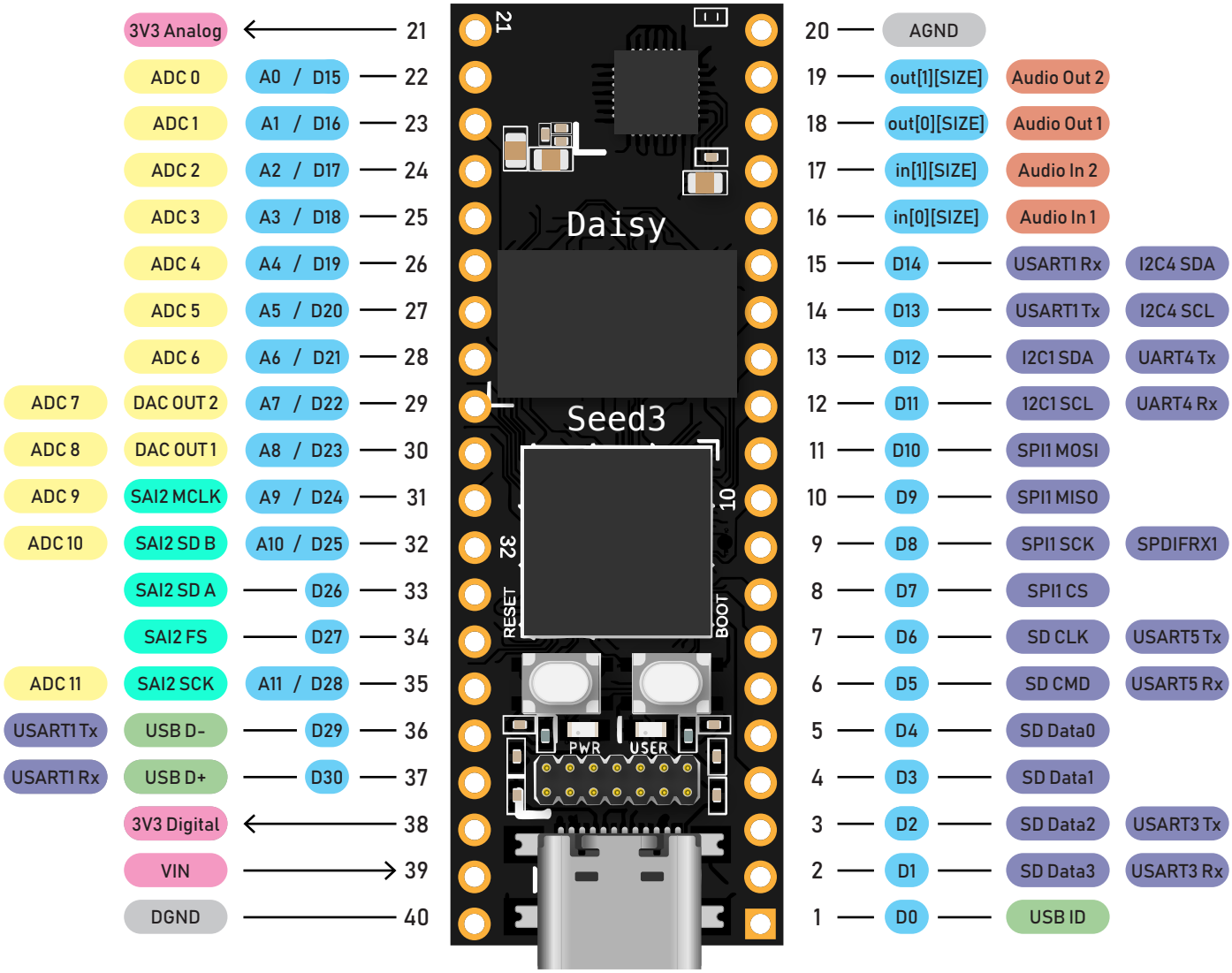
All GPIO Pins are 5V tolerant I/O except for the following pins which are 3.3V tolerant I/O:

Pin 24 - (A2/D17, PB1, ADC2)
Pin 25 - (A3/D17, PA7, ADC3)
Pin 28 - (A6/D21, PC4, ADC6)
Pin 29 - (A7/D22, PA5, ADC7)
Pin 30 - (A8/D23, PA4, ADC8)

Pinout

ANALOG AUDIO | ANALOG GPIO | DIGITAL AUDIO GPIO | PERIPHERAL GPIO

USB GPIO | GROUND | POWER | DAISY PIN NAMES*



* "D" for Digital GPIO or "A" for Analog I/O, depending on use case.

Pin Functions - Table 2

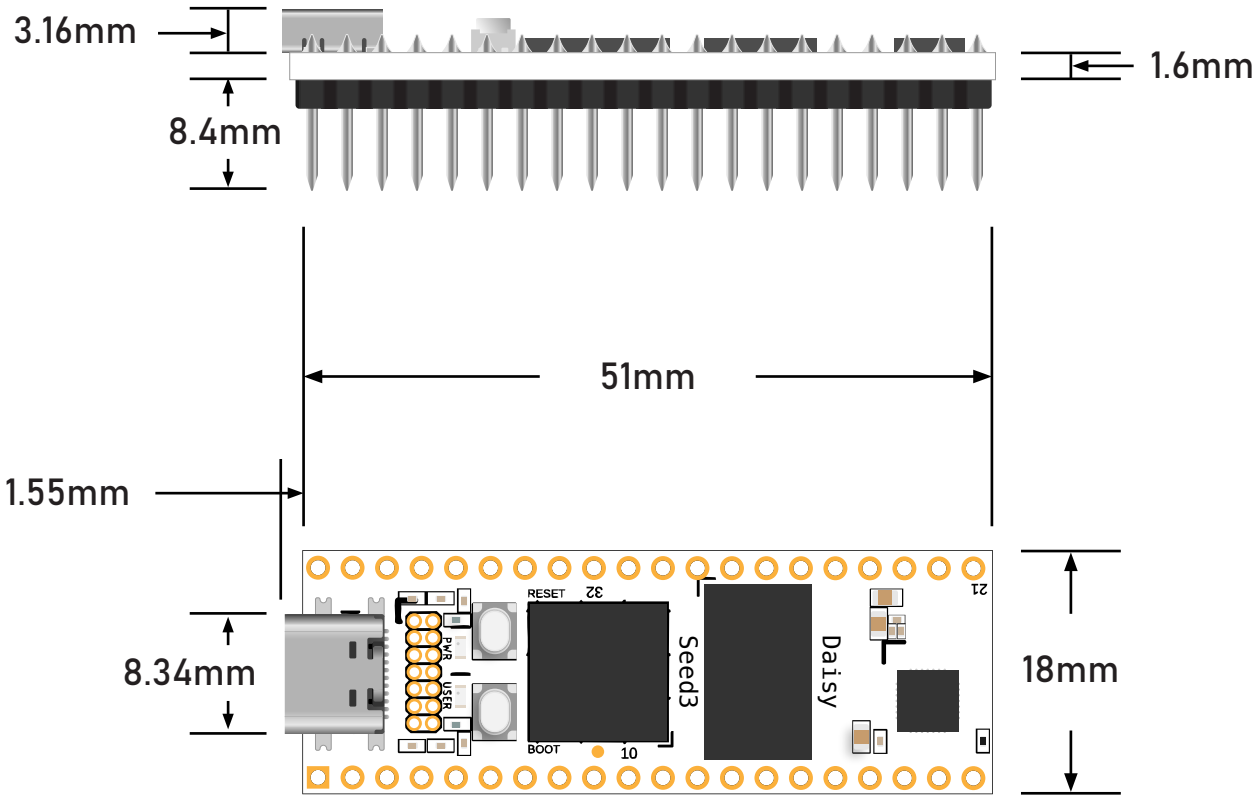
PINOUT	DAISY PIN NAME*	STM32 PIN NAME	PRIMARY FUNCTION	ALT. FUNCTIONS 1	ALT. FUNCTIONS 2
1	D0	PB12	GPIO	USB_HS_ID/UART5_RX/ USART3_CK	TIM1_BKIN
2	D1	PC11	GPIO	SDMMC1_D3/USART3_RX/UART4_RX	SPI3_MISO/ I2S3_SDI/HRTIM_FLT2
3	D2	PC10	GPIO	SDMMC1_D2/USART3_TX/UART4_TX	SPI3_SCK/I2S3_CK/HRTIM_EEV1
4	D3	PC9	GPIO	SDMMC1_D1/UART5_CTS	I2S_CKIN/MCO2
5	D4	PC8	GPIO	SDMMC1_D0/UART5_RTS	
6	D5	PD2	GPIO	SDMMC1_CMD/UART5_RX	
7	D6	PC12	GPIO	SDMMC1_CK/UART5_TX/ USART3_CK	SPI3_MOSI/I2S3_SDO
8	D7	PG10	GPIO	SPI1_NSS/I2S1_WS	HRTIM_FLT5
9	D8	PG11	GPIO	SPI1_SCK/I2S1_CK	LPTIM1_IN2/HRTIM_EEV4
10	D9	PB4	GPIO	SPI1_MISO/UART7_TX	SPI1_MISO/I2S1_SDI/SPI3_MISO/I2S3_SDI/ SPI6_MISO
11	D10	PB5	GPIO	SPI1_MOSI/UART5_RX	SPI1_MOSI/I2S1_SDO/SPI3_MOSI/I2S3_ SDO/SPI6_MOSI/I2C4_SMBA/ TIM17_BKIN
12	D11	PB8	GPIO	I2C1_SCL/UART4_RX	I2C4_SCL/TIM16_CH1/TIM4_CH3
13	D12	PB9	GPIO	I2C1_SDA/UART4_TX/ SPI2_NSS/ I2S2_WS	I2C4_SDA/I2C4_SMBA/TIM17_CH1/TIM4_ CH4
14	D13	PB6	GPIO	USART1_TX/LPUART1_TX/UART5_TX	I2C1_SCL/I2C4_SCL/ TIM16_CH1N/TIM4_ CH1
15	D14	PB7	GPIO	USART1_RX/LPUART1_RX	I2C1_SDA/I2C4_SDA/TIM17_CH1N/TIM4_ CH2
16	NC	x	AUDIO IN L		
17	NC	x	AUDIO IN R		
18	NC	x	AUDIO OUT L		
19	NC	x	AUDIO OUT R		
20	NC	x	AGND		
21	NC	x	+3V3A		
22	A0, D15	PC0	GPIO	ADC0/SAI2_FS_B	
23	A1, D16	PA3	GPIO	ADC1/USART2_RX	TIM2_CH4/TIM5_CH4
24	A2, D17	PB1	GPIO	ADC2	TIM1_CH3N/TIM3_CH4
25	A3, D18	PA7	GPIO	ADC3/SPI1_MOSI/I2S1_SDO/SPI6_MOSI	TIM1_CH1N/TIM3_CH2
26	A4, D19	PA6	GPIO	ADC4/SPI1_MISO/I2S1_SDI/SPI6_MISO	TIM1_BKIN/TIM3_CH1
27	A5, D20	PC1	GPIO	ADC5	
28	A6, D21	PC4	GPIO	ADC6/I2S1_MCK	
29	A7, D22	PA5	GPIO	ADC7/DAC1_OUT2	SPI1_SCK/I2S1_CK/SPI6_SCK/ D2PWREN/TIM2_CH1
30	A8, D23	PA4	GPIO	ADC8/DAC1_OUT1	SPI1_NSS/I2S1_WS/SPI3_NSS/I2S3_WS/ SPI6_NSS/D1PWREN
31	A9, D24	PA1	GPIO	ADC9/SAI2_MCLK_B	UART4_RX/TIM2_CH2/TIM5_CH2
32	A10, D25	PA0	GPIO	ADC10/SAI2_SD_B	UART4_TX/TIM2_CH1/TIM2_ETR/TIM5_CH1
33	D26	PD11	GPIO	SAI2_SD_A/I2C4_SMBA	LPTIM2_IN2
34	D27	PG9	GPIO	SAI2_FS_B/USART6_RX	SPI1_MISO/I2S1_SDI
35	A11, D28	PA2	GPIO	ADC11/SAI2_SCK_B	USART2_TX/TIM2_CH3/TIM5_CH3
36	D29	PB14	GPIO	USB_HS_D-/USART1_TX	TIM1_CH2N
37	D30	PB15	GPIO	USB_HS_D+/USART1_RX	
38	NC	x	+3V3D		
39	NC	x	VIN		
40	PG3	x	GND		

*Pin names are the same indices preceded by: “D” for Digital GPIO or “A” for Analog I/O

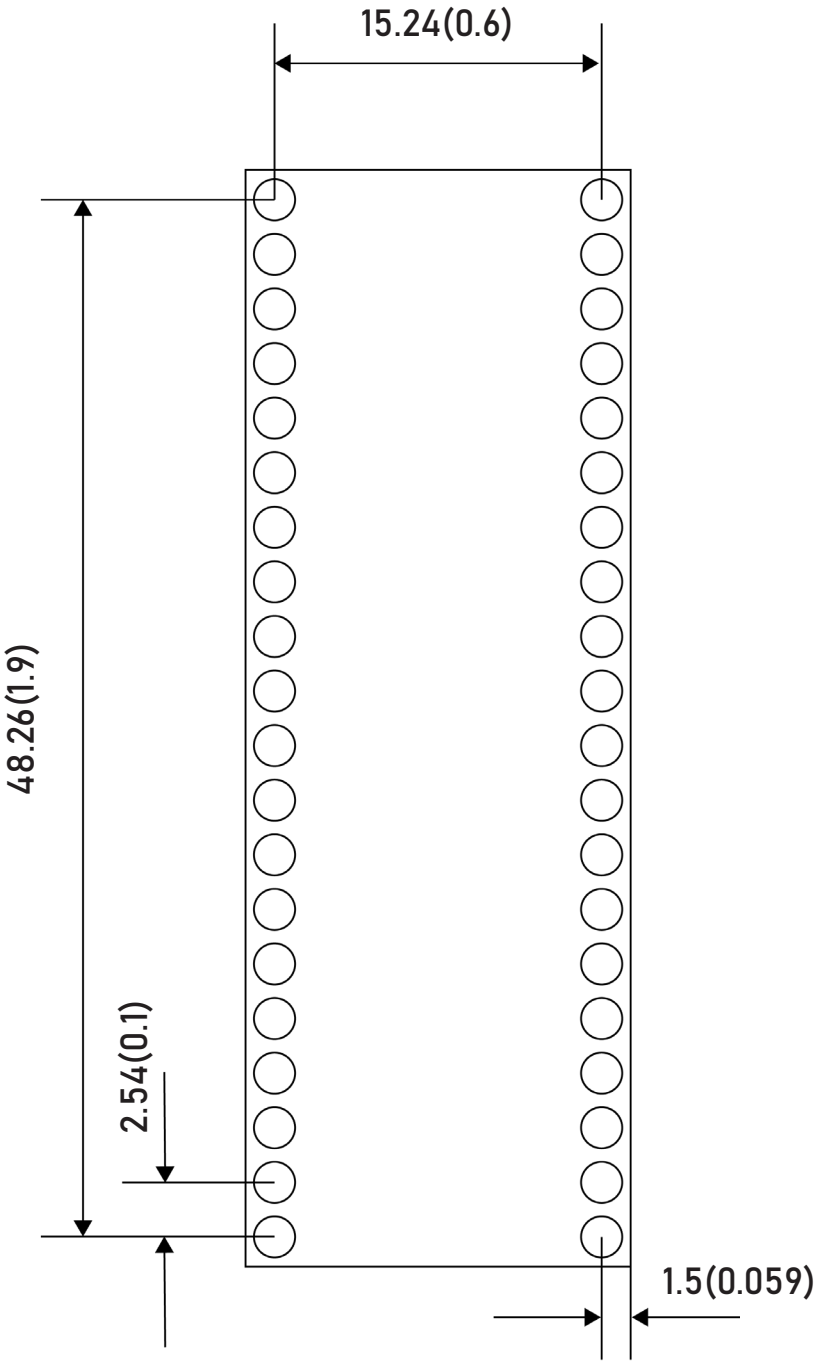
Electrical Characteristics - Table 3

* The min/max rating in this table represents the expected operating range for the device. Signals outside of this range will not necessarily damage the Daisy Seed. See Table 1 for Absolute min/max ratings.

PINOUT	DAISY PIN NAME	MIN	MAX	TYPICAL
1	D0	0V	+3V3	0 to +3V3
2	D1	0V	+3V3	0 to +3V3
3	D2	0V	+3V3	0 to +3V3
4	D3	0V	+3V3	0 to +3V3
5	D4	0V	+3V3	0 to +3V3
6	D5	0V	+3V3	0 to +3V3
7	D6	0V	+3V3	0 to +3V3
8	D7	0V	+3V3	0 to +3V3
9	D8	0V	+3V3	0 to +3V3
10	D9	0V	+3V3	0 to +3V3
11	D10	0V	+3V3	0 to +3V3
12	D11	0	+3V3	0 to +3V3
13	D12	0	+3V3	0 to +3V3
14	D13	0	+3V3	0 to +3V3
15	D14	0	+3V3	0 to +3V3
16	NC	-1V8	+1V8	3.6Vpp
17	NC	-1V8	+1V8	3.6Vpp
18	NC			0dBFs @ 1Vrms
19	NC			0dBFs @ 1Vrms
20	NC			GND
21	NC			+3V3 (output only)
22	A0, D15	0V	+3V3	0 to +3V3
23	A1, D16	0V	+3V3	0 to +3V3
24	A2, D17	0V	+3V3	0 to +3V3
25	A3, D18	0V	+3V3	0 to +3V3
26	A4, D19	0V	+3V3	0 to +3V3
27	A5, D20	0V	+3V3	0 to +3V3
28	A6, D21	0V	+3V3	0 to +3V3
29	A7, D22	0V	+3V3	0 to +3V3
30	A8, D23	0V	+3V3	0 to +3V3
31	A9, D24	0V	+3V3	0 to +3V3
32	A10, D25	0V	+3V3	0 to +3V3
33	D26	0	+3V3	0 to +3V3
34	D27	0	+3V3	0 to +3V3
35	A11, D28	0	+3V3	0 to +3V3
36	D29	0	+3V3	0 to +3V3
37	D30	0	+3V3	0 to +3V3
38	NC			+3V3 (output only)
39	NC	+4V	+17V	+4V to +17V
40	PG3			GND


3.16mm
8.4mm
1.6mm
51mm
1.55mm
8.34mm
18mm
RESET
32
21
PWR
USER
Seed3
Daisy
BOOT
10

Dimensions in mm (inches)



Find the KiCad part here.

Find the EAGLE part here.

Onboard USB
The USB port onboard the Seed provides power to the board.
The USB-C is configured for typical 5V operation with the default 500mA power draw setting.

When power is applied this way:

- The PWR LED will illuminate
- Both the Digital and Analog +3.3V pins will output the voltage from their regulators.
- The firmware on the Seed will begin to run

This input is reverse protected, and can be connected even when power is being supplied through the Vin Pin.

The input power circuit for the Seed3 can be seen here:

Figure 1.0 - Onboard USB Power

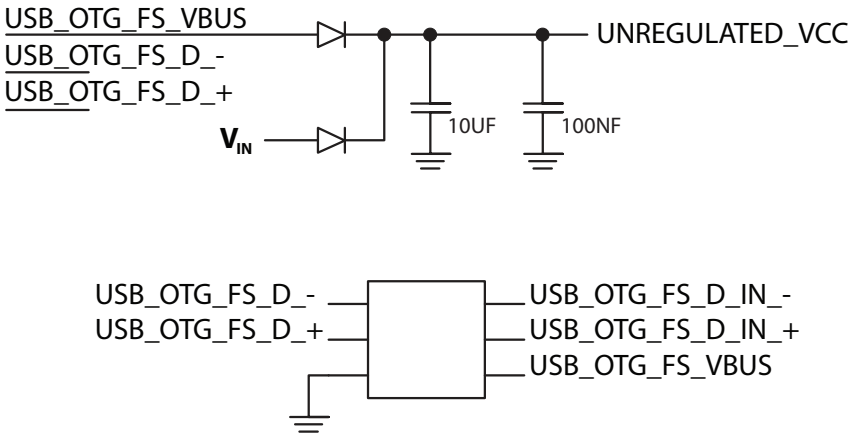
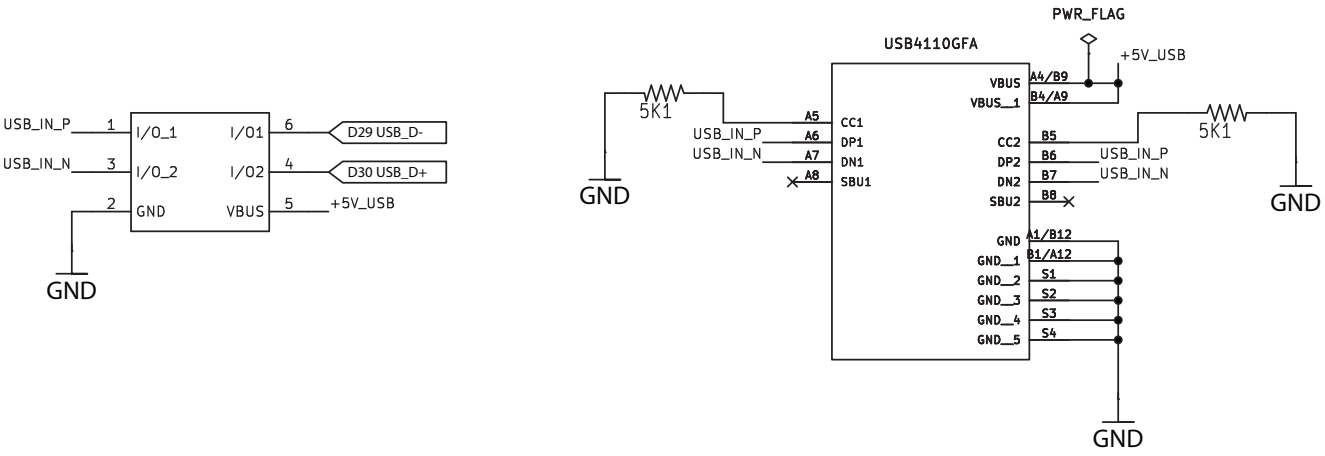


Figure 1.1 - External USB-C Power



Vin Pin

The Vin Pin provides an input to the Seed3 that can be used to supply the board with power.

This pin accepts a DC Voltage between 3.8 and 16V to power the Seed.

When power is applied this way:

- The PWR LED will illuminate
- Both the Digital and Analog +3.3V pins will output the voltage from their regulators
- The firmware on the Seed will begin to run

It is worth noting that higher input voltages can increase the heat generated by the Seed's voltage regulators.

As an example of a few input circuits:

Figure 1.2 - +9V Power Supply Input

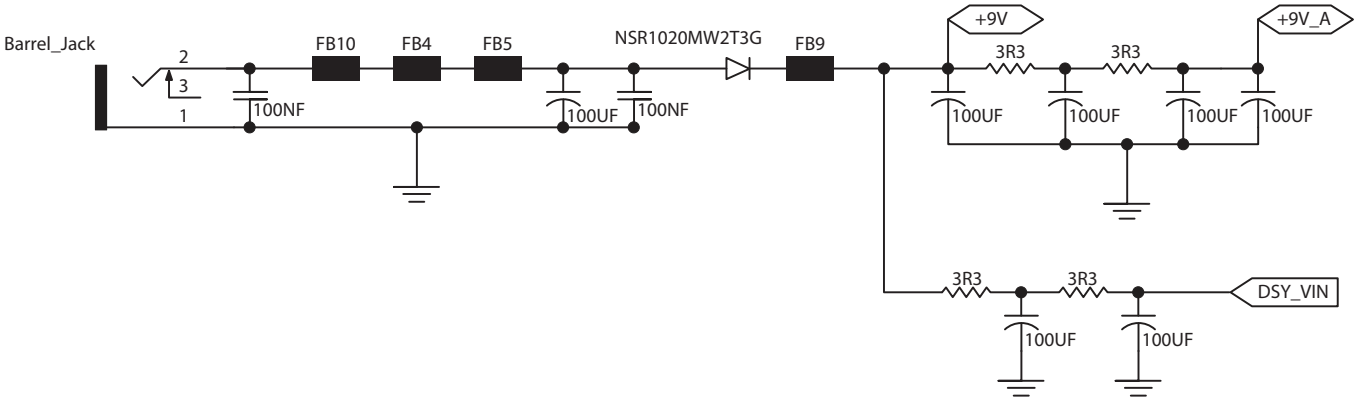
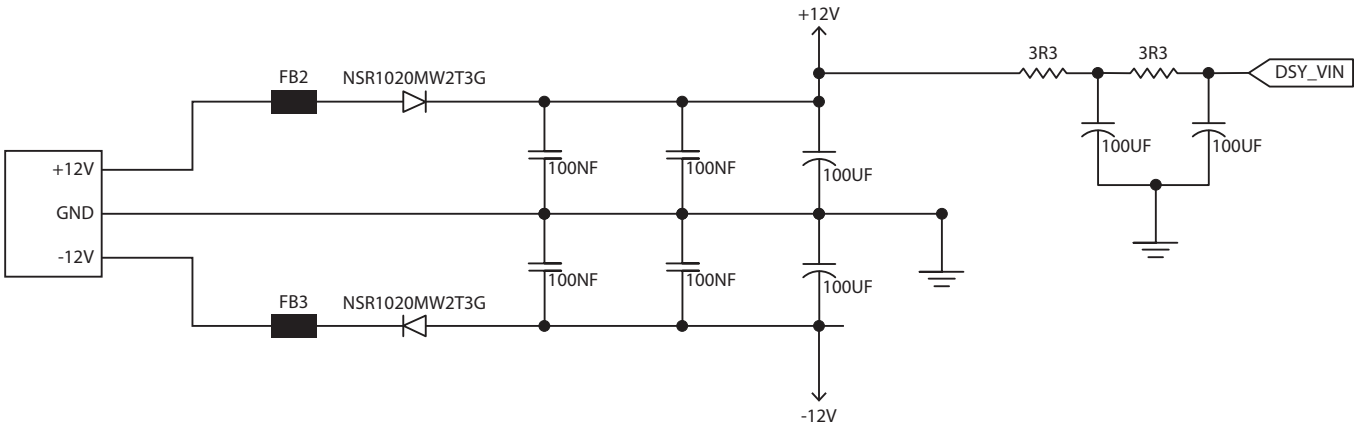


Figure 1.3 - Eurorack +/- 12V Power Supply Input



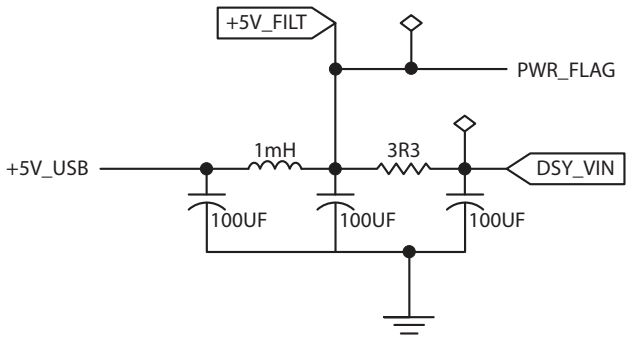
Additional Filtering

It is not required, but it is recommended, and often very helpful to have some additional filtering on whatever voltage is being used to power the Seed.

The examples above demonstrate some power supply filtering that can have a substantial effect on:

- Noise from the power supply affecting the performance of the Daisy
- Noise generated by the Daisy's CPU from getting back into the main power supply

Figure 1.4 - +5V USB VBus Power Supply Input



These filters both show using 100uF capacitors -- these are typically electrolytic, and can be quite large.
If you have enough room, these -- or even larger (220uF) -- values are suitable. Otherwise, you can reduce these to typically MLCC sizes (22uF C1206, 10uF C0805, etc.), but they will have a reduced effect.

A low value (3.3 ohm) resistor of suitable power rating (1/4 W) can also replace the inductor in the example below. The inductor is typically much larger, occupying more surface area on the PCB, and is not as strong of a filter as the resistor. However, the resistor will cause some voltage drop on the voltage, and generate some additional heat on the PCB.

Due to different voltage levels, you can see different combinations of these circuits demonstrated on each of the Seed3 Dev Kits. The figures above, demonstrate each of these techniques.

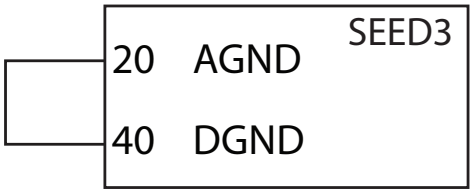
Ground

The Seed has two ground pins. AGND, and DGND.

These should be connected outside of the Daisy, and connected to whatever power supply ground is shared for the entire circuit. GND should always be connected prior to connecting anything to VIN.

Not following these recommendations can result in unwanted noise, or even damage to the Seed.

Figure 1.5 - Connecting AGND to DGND



Note: For all applications, AGND must be connected to DGND.

GPIO, or General Purpose I/O, make up most of the Seed's available Pins.

Even if a pin is labeled with some special, alternate function, all of the pins that are not related to power (VIN, 3v3 A, 3v3 D, DGND, AGND), or Audio (lin, rin, lout, rout) are GPIO.

These pins can be configured by software to serve a number of purposes, and have a few general functions:

Input

Configuring a GPIO as an input allows for buttons, toggles, or other "digital" inputs to be read by software. These inputs can only read the input in two states, "ON" or "OFF", which is why we call them, "Digital".

Each GPIO can configure a built-in pull-up or pull-down resistor for the pin.
This can help in reducing the number of external components necessary when interacting with various inputs.

For example, if we look at how one of the buttons and toggles on the Pedal DevKit is configured, we see that one side of the button is connected to GND, while the other is connected to the Seed.

Figure 2.1 - Tactile Switch

Available Pins: Any GPIO

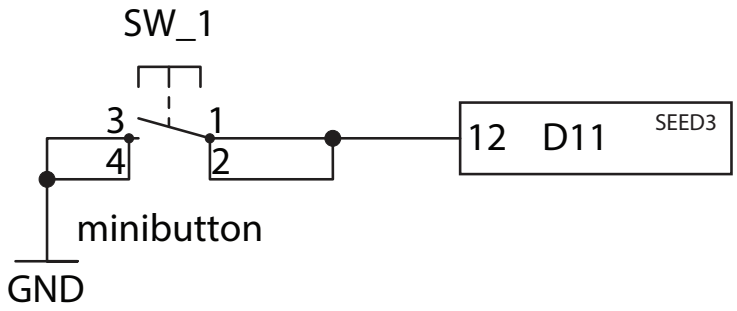
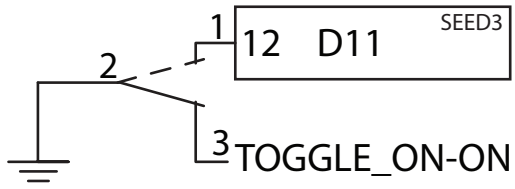


Figure 2.2 - Toggle Switch

Available Pins: Any GPIO



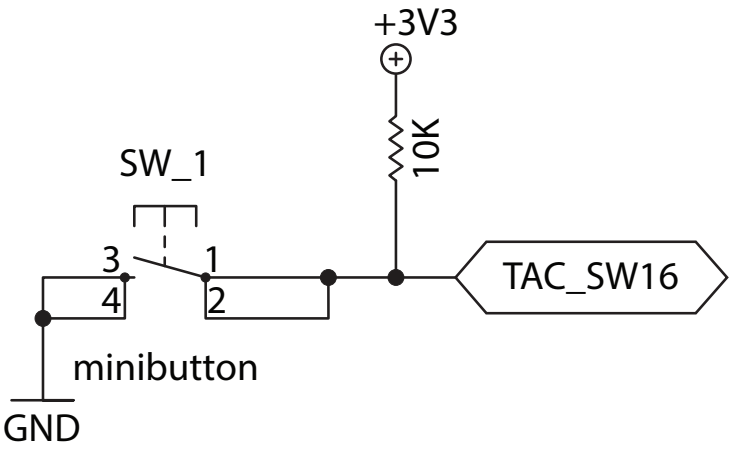
When we write the code to read from the pin we configure the pin as an input, with a pull up resistor.

```
GPIO my_button;
my_button.Init(seed::D0, GPIO::Mode::INPUT, GPIO::Pull::PULLUP);
```

If we didn't have the pull up resistors, as is often the case when adding several buttons with shift registers, we would need to include a pull-up resistor to +3.3V on each button trace going to its input.

You can see how these are configured when using a CD4021 shift register on the on the Desktop Dev Kit.

Figure 2.3 - Tactile Switch w/ pull-up resistor

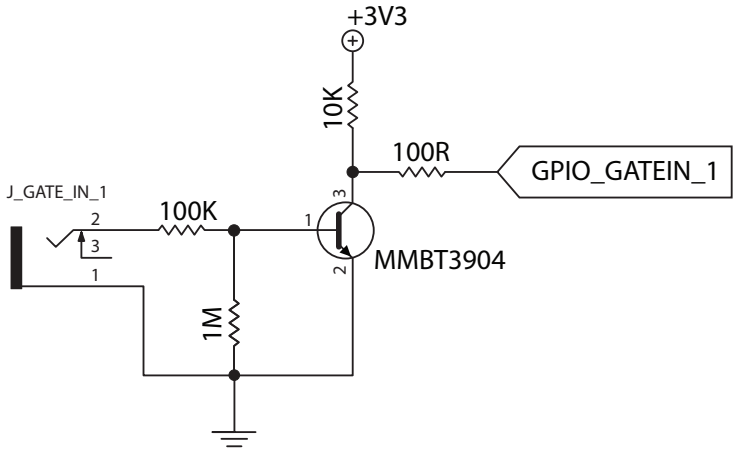


BJT Gate Input Circuit

Input signals aren't always going to be within an expected range. When this is the case, it can be helpful to have a circuit that will trigger at a sufficiently low voltage to create the expected signal, and protect from excessive voltages that might exceed the input of the Seed's GPIO.

In these cases we can use something like this BJT input circuit:

Figure 2.4 - Bipolar Junction Transistor Circuit



The GPIO_GATEIN_1 signal will pull down to GND whenever the signal on the input to the circuit exceeds about +0.3V, making this a reliable means of reading a signal that you're not sure will reach the normal threshold, or might exceed the GPIO's maximum input.

The input is inverted through this circuit, and only "digital" signals are generated (i.e. the output will always be either 0V or +3.3V.

Output

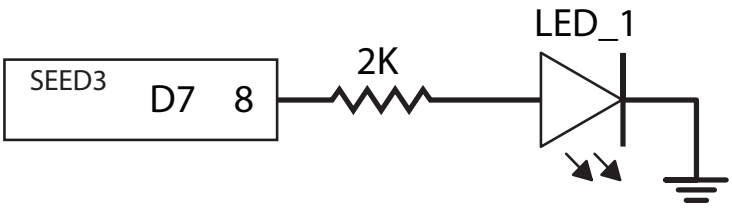
Configuring a GPIO as an output allows for software to control the state of external things like LEDs, relays, etc.

The Seed's GPIO can be set in either a push-pull, or an open-drain configuration.
Push-pull mode is the most commonly used configuration, and will drive the voltage of the output pin to 3.3V when written high, or 0V when written low.
On the other hand, open drain mode will drive the output to 0V when written low, but is otherwise in a Hi-Z state, and requires a pull-up resistor. This type of configuration can be common for sensors operating at different supply voltages, and for some communication protocols (like I2C).

It can be assumed that any examples discussed are operating with push-pull mode unless explicitly mentioned that they are using an open-drain configuration.

On the Pedal DevKit we can see some GPIO output's being used to drive LEDs.

Figure 2.5 - LED



In C++ we could set up, and control that LED with the following:

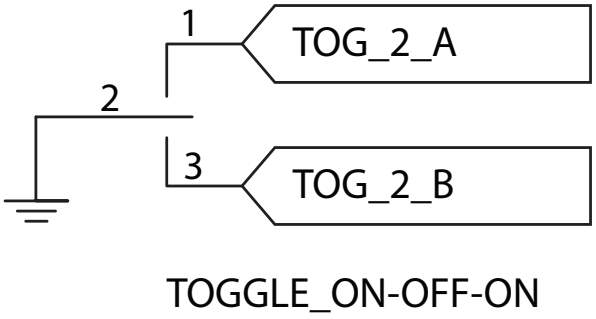
```
GPIO my_led;
my_led.Init(seed::D24, GPIO::Mode::Output);

my_led.Write(true); /**< Turn it on! */

my_led.Write(false); /**< Turn it off... */
```

Combining Multiple GPIO

While a number of pins can combine to use various peripherals within the STM32 such as SPI, I2C, UART, etc. there are some things that work using the GPIO in their basic modes.

A simple example of this would be a three-position toggle:

Figure 2.6 - Three-way toggle



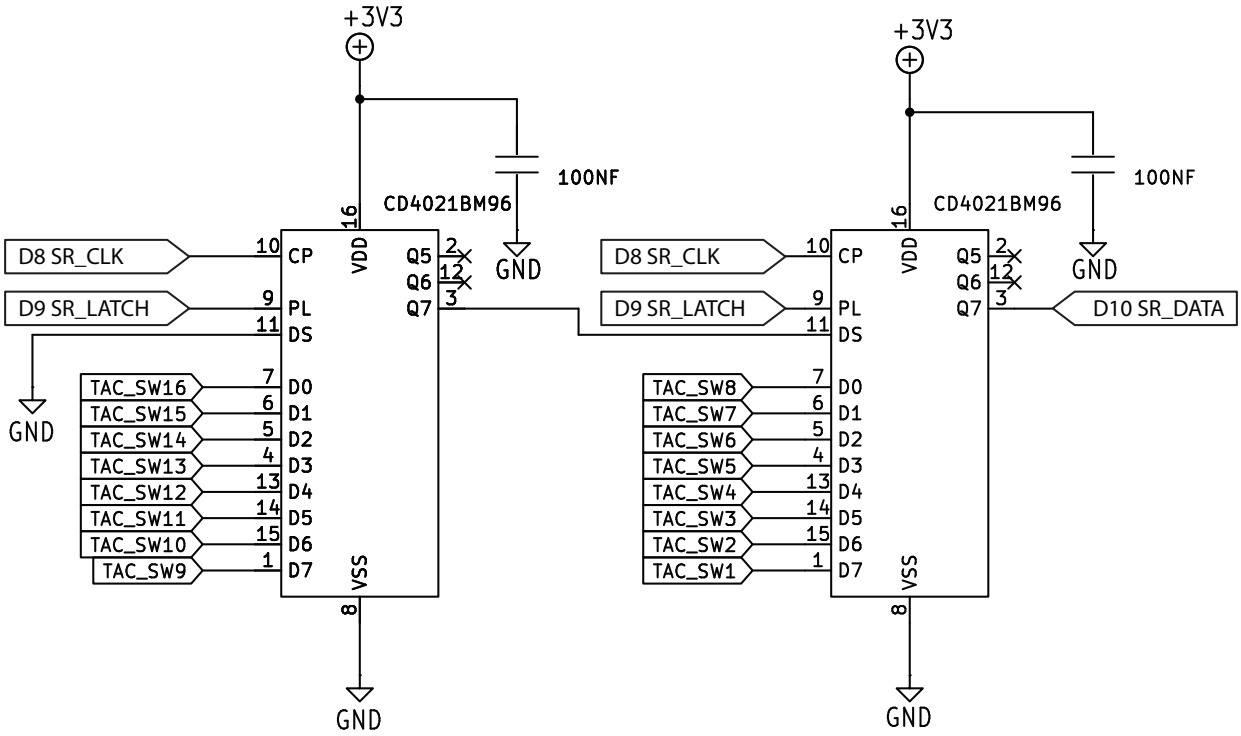
In this case, both signals will apply their pull ups, and the software can detect when the the toggle is in any of the three positions by looking at the combination of the state of each GPIO input.

For a more complex example, the CD4021 shift register is controlled with two output pins and an input pin. A latch pin can be strobed to clear the values in the shift register, and then one output and one input can be strobed, and read respectively to read 8 parallel inputs from this set of three pins. These devices can be daisy chained to allow the connection of many inputs with only a few of the Daisy's GPIO pins.

If that sounds complicated, don't worry! A lot of the common devices have support built into our libraries. For example, this code will set up, and read the values from the sixteen buttons from the Desktop DevKit, and use one of the buttons to turn on/off the Daisy's LED.

```
// Initialization
ShiftRegister4021<2, 1> sr;
ShiftRegister4021<2, 1>::Config sr_config;
sr_config.clk = seed::D8;
sr_config.latch = seed::D9;
sr_config.data[0] = seed::D10;
sr.Init(sr_config);

// ... In the main while(1) {} loop
// Trigger SR to read all pins:
sr.Update();

// Read the value of SW_1 (flipped because of the pullup)
bool btn_1 = !sr.State(0);

// Light up the Daisy's LED when the button is presssed.
seed.SetLed(btn_1);
```

Figure 2.7 - Daisy-chained Shift Registers



Voltage Tolerance

While the Daisy is a 3.3V device, all of the GPIO pins are +5V Tolerant with the following exceptions:

Physical Pin Number	GPIO Name	STM32 Pin Name
Pin 24	D17	PB1
Pin 25	D18	PA7
Pin 28	D21	PC4
Pin 29	D22	PA5
Pin 30	D23	PA4

Figure 3.0 - Stereo Input Normalling

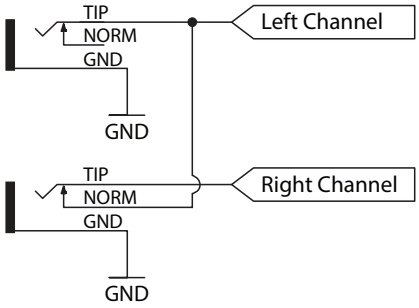


Figure 3.1 - Line Level Audio Input

Input Impedence: 20K Ω (typ.)

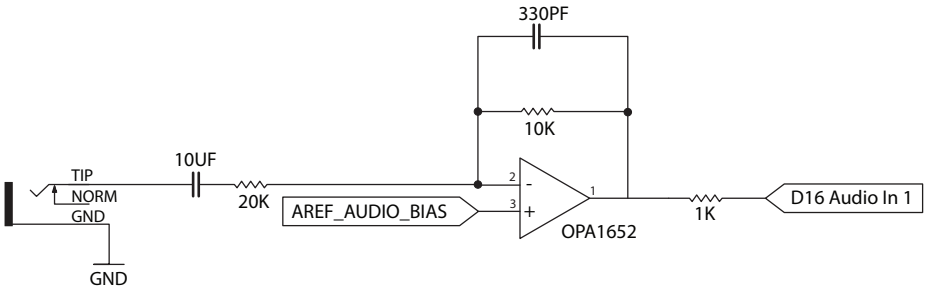


Figure 3.2 - Eurorack Level Audio Input

Input Impedence: 100K Ω (typ.)

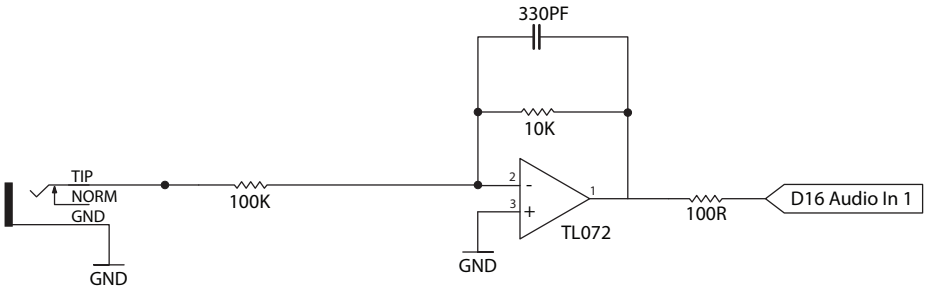


Figure 3.3 - Instrument Level Audio Input

Input Impedence: 1M Ω (typ.)

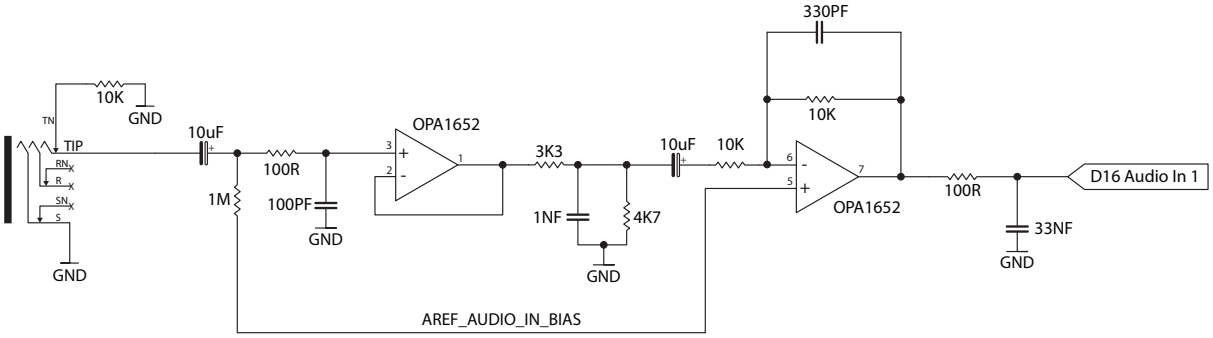


Figure 3.4 - Line Level Audio Output

Output Imepedence: 100R

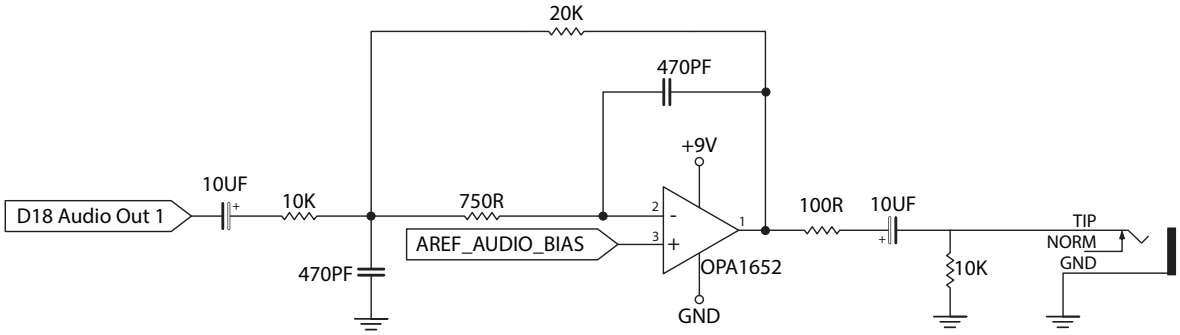


Figure 3.5 - Stereo Eurorack Level Audio Output

Output Imepedence: 100R

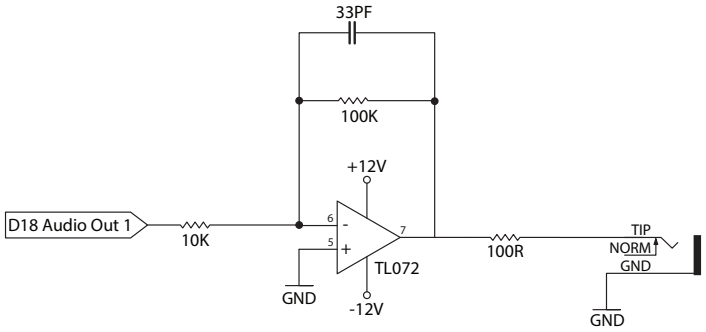


Figure 3.6 - Stereo Instrument Level Audio Output

Output Impedence: 100 Ω (typ.)

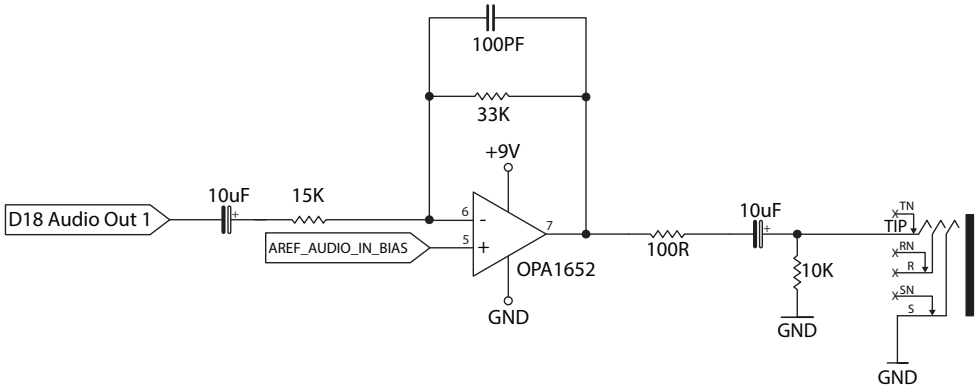
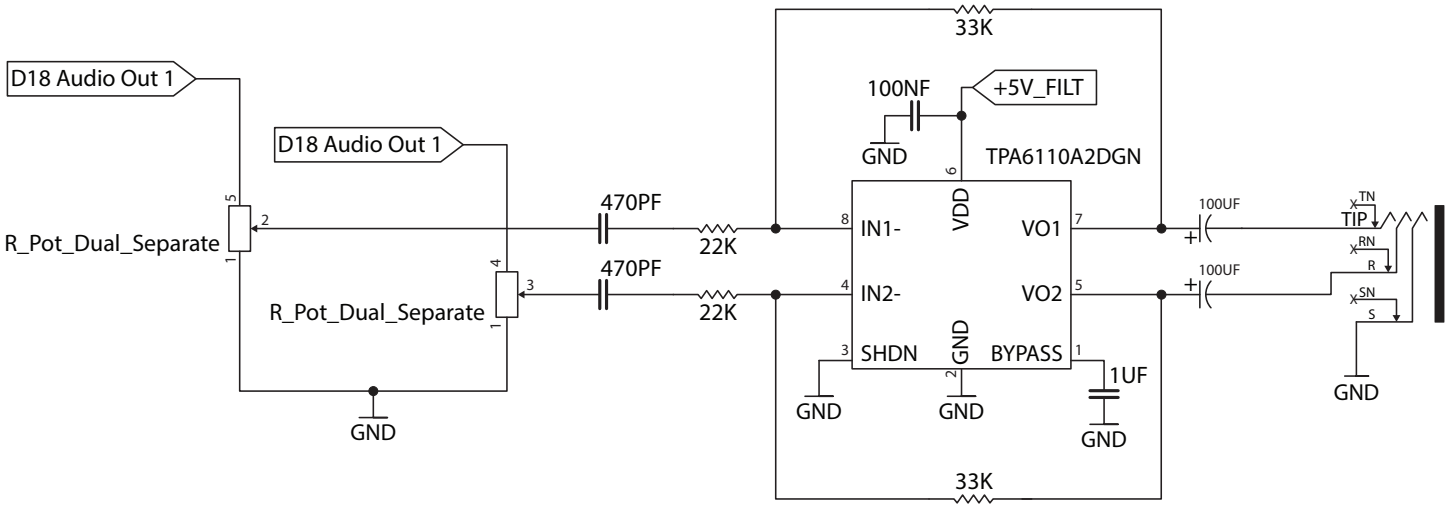


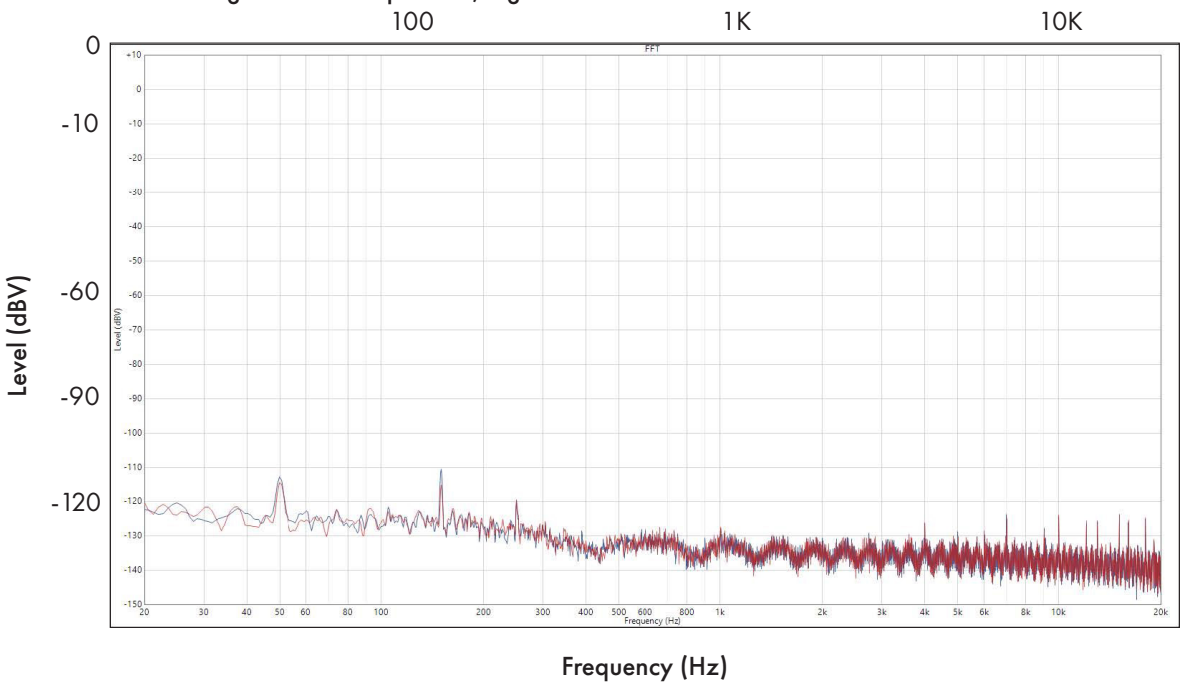
Figure 3.7 - Headphone Amp Output



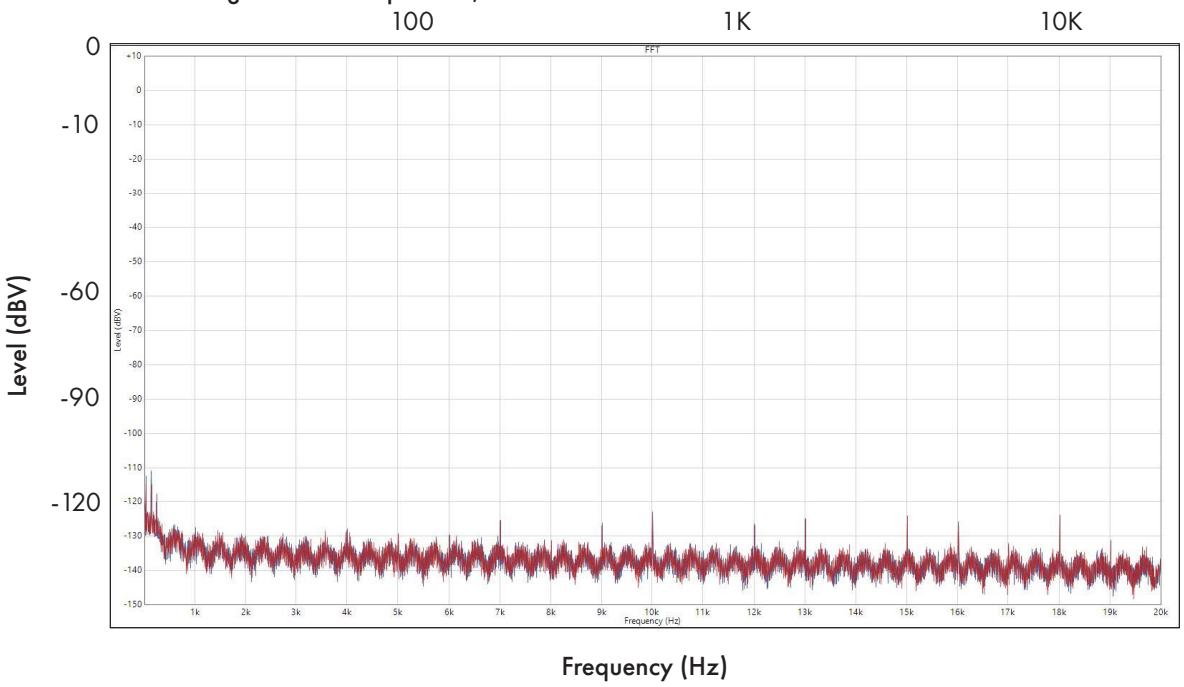
Line Level

Circuit: Typical app circuit on Seed3 Desktop Dev Kit
Measured with: Audio Precision APX555B Audio Analyzer

Audio Pass Through: Idle noise spectrum, logarithmic scale

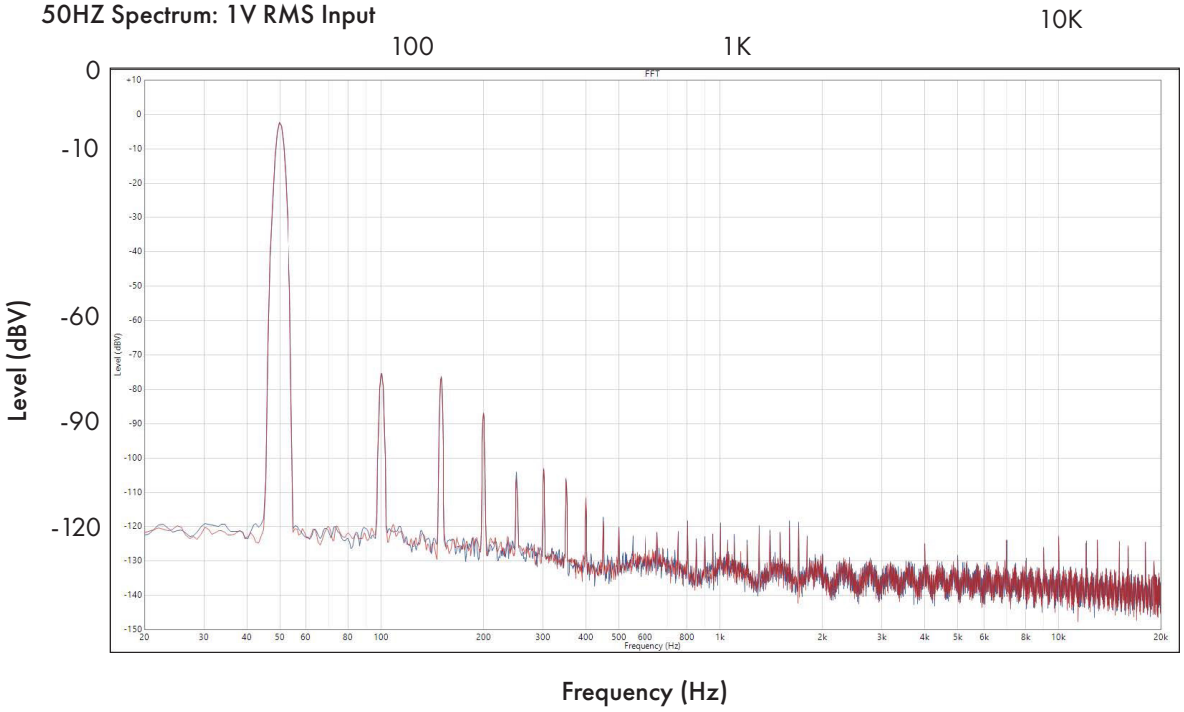


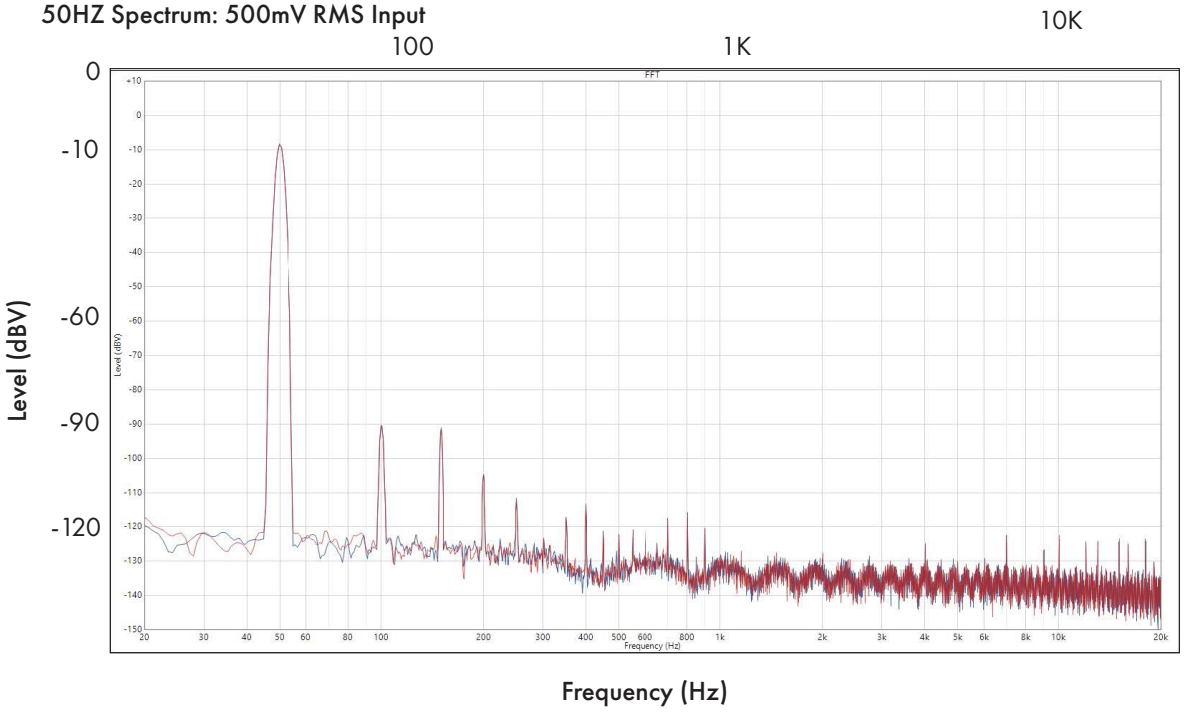
Audio Pass Through: Idle noise spectrum, linear scale



Line Level

Circuit: Typical app circuit on Seed3 Desktop Dev Kit
Measured with: Audio Precision APX555B Audio Analyzer

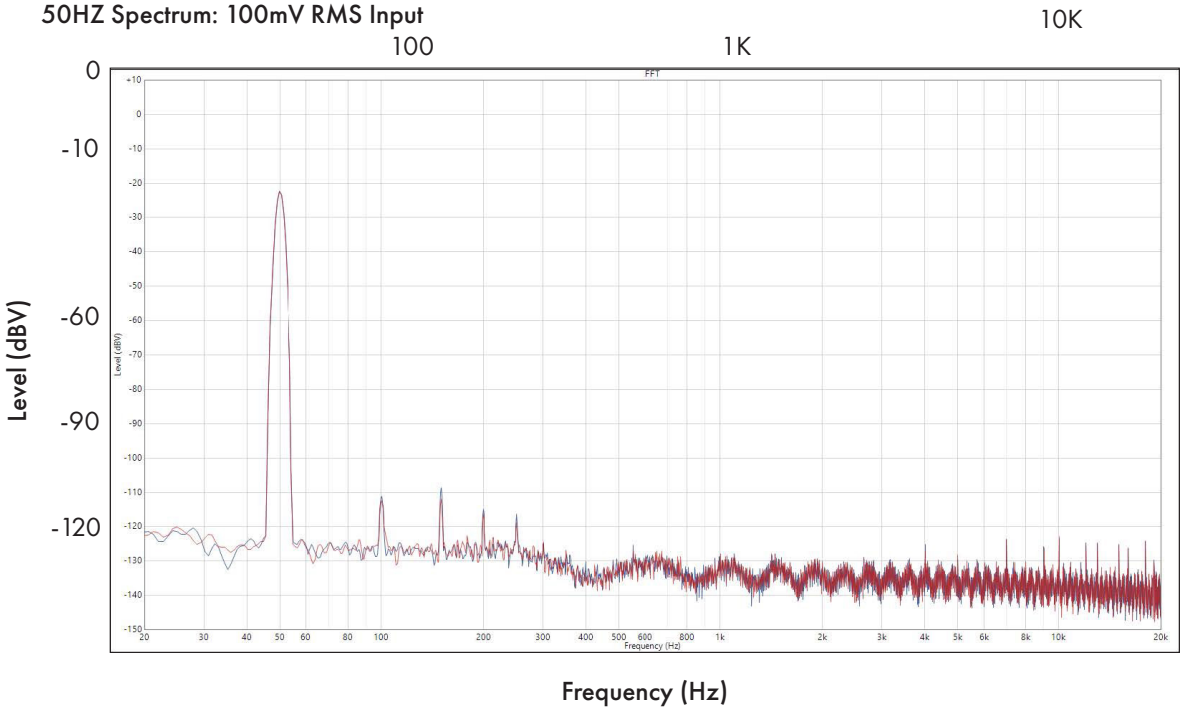




Line Level

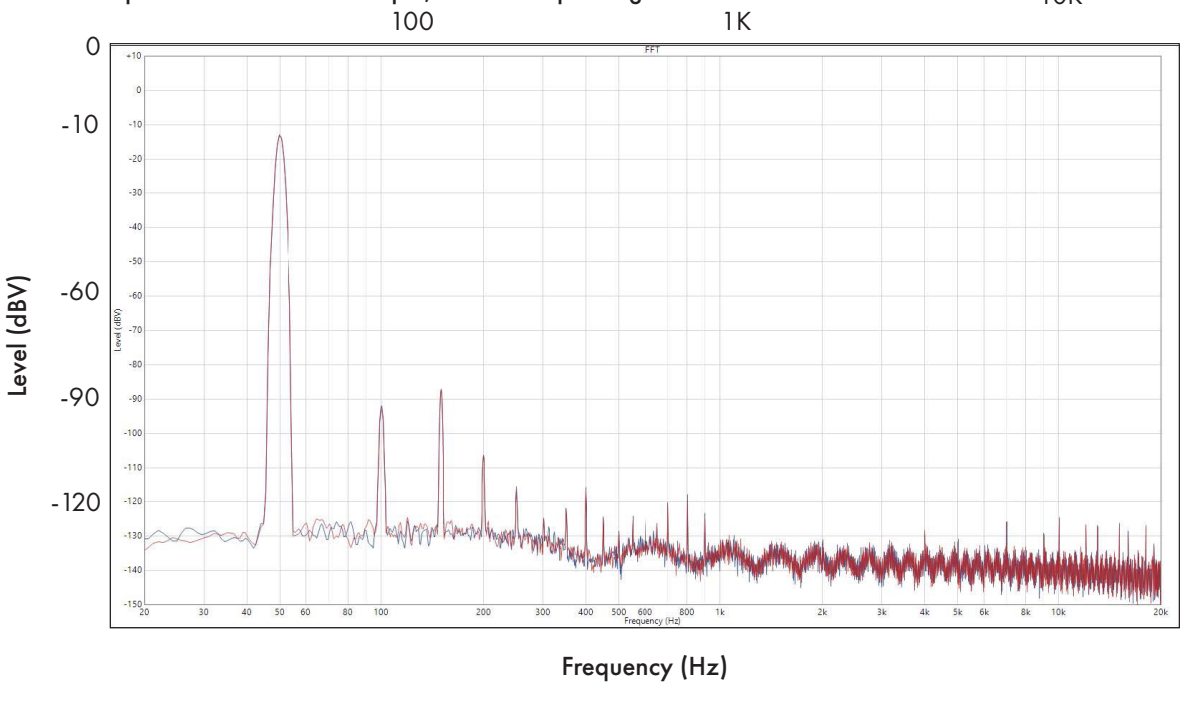
Circuit: Typical app circuit on Seed3 Desktop Dev Kit
Measured with: Audio Precision APX555B Audio Analyzer

50HZ Spectrum: 100mV RMS Input



Frequency (Hz)

50HZ Spectrum: 500mV RMS Input, 300ohm input stage

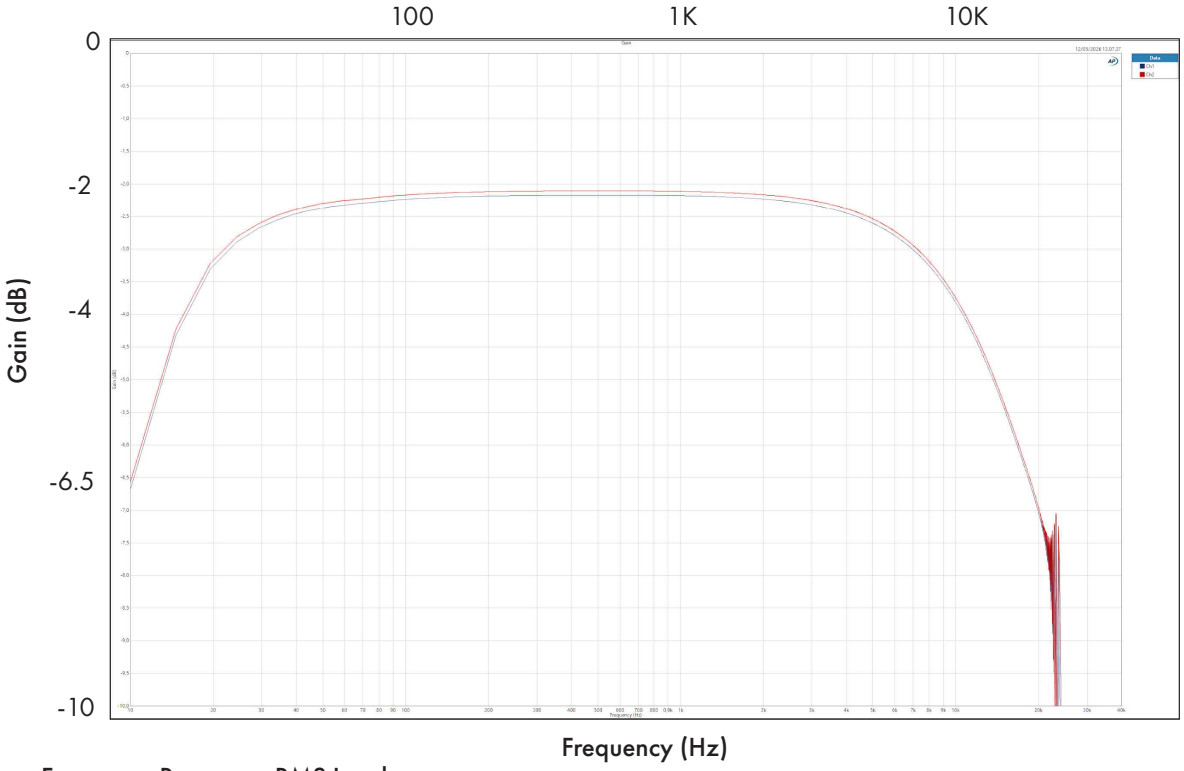


Frequency (Hz)

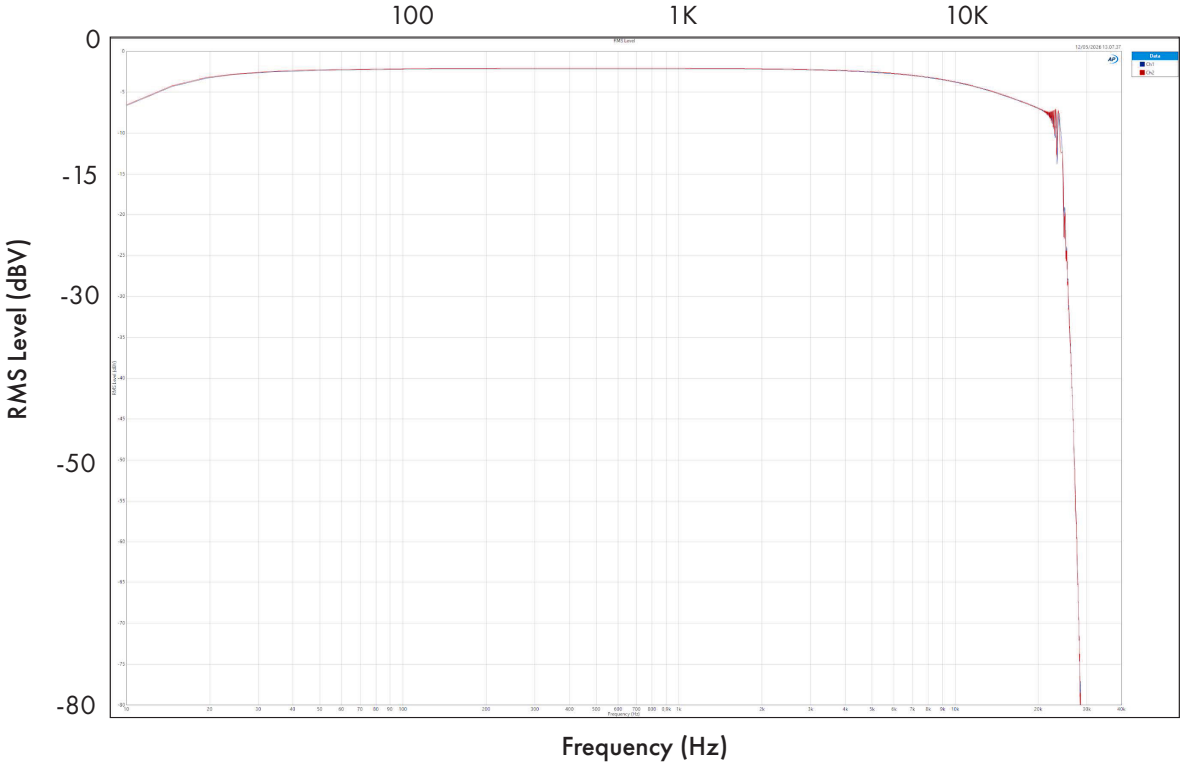
Line Level

Circuit: Typical app circuit on Seed3 Desktop Dev Kit
Measured with: Audio Precision APX555B Audio Analyzer

Frequency Response: Gain



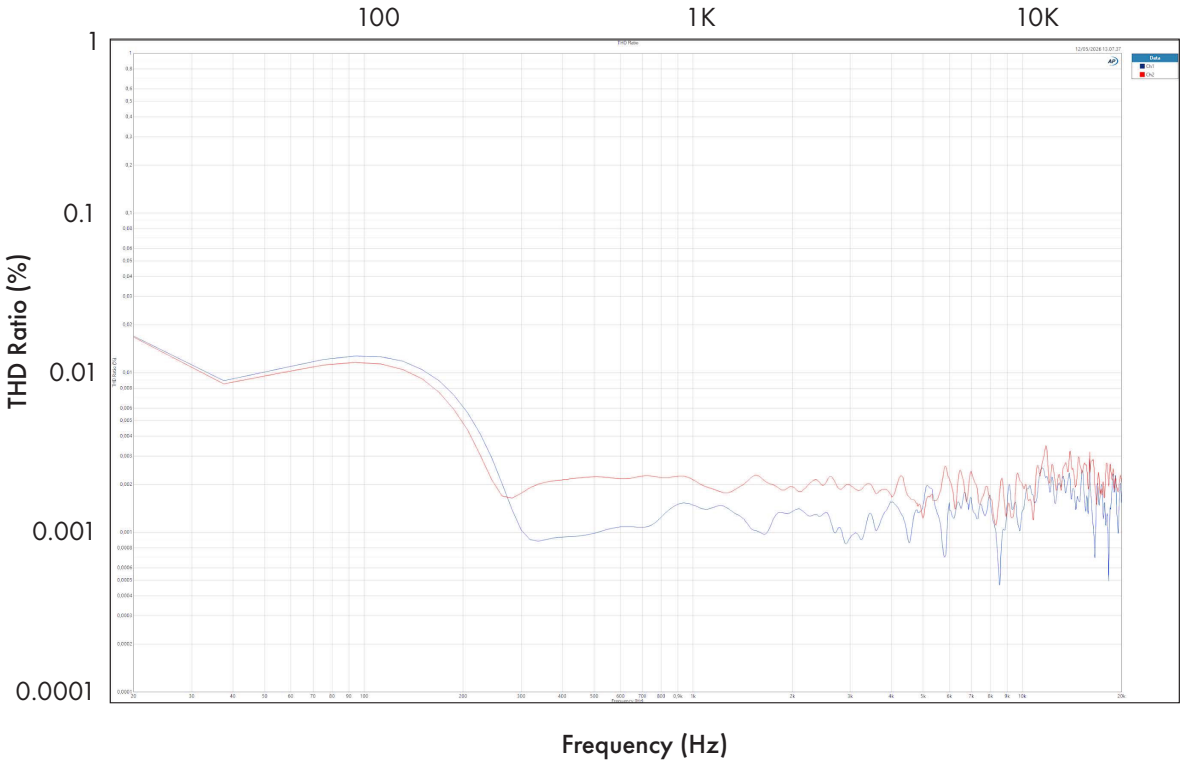
Frequency Response: RMS Level



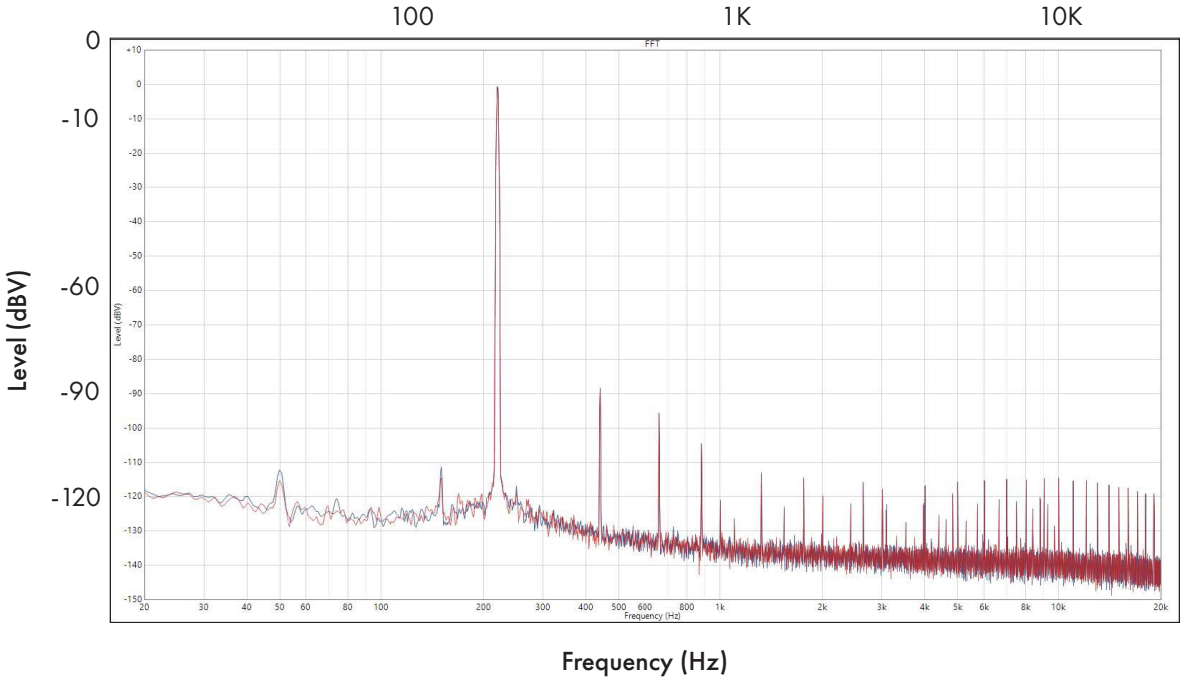
Line Level

Circuit: Typical app circuit on Seed3 Desktop Dev Kit
Measured with: Audio Precision APX555B Audio Analyzer

THD Ratio



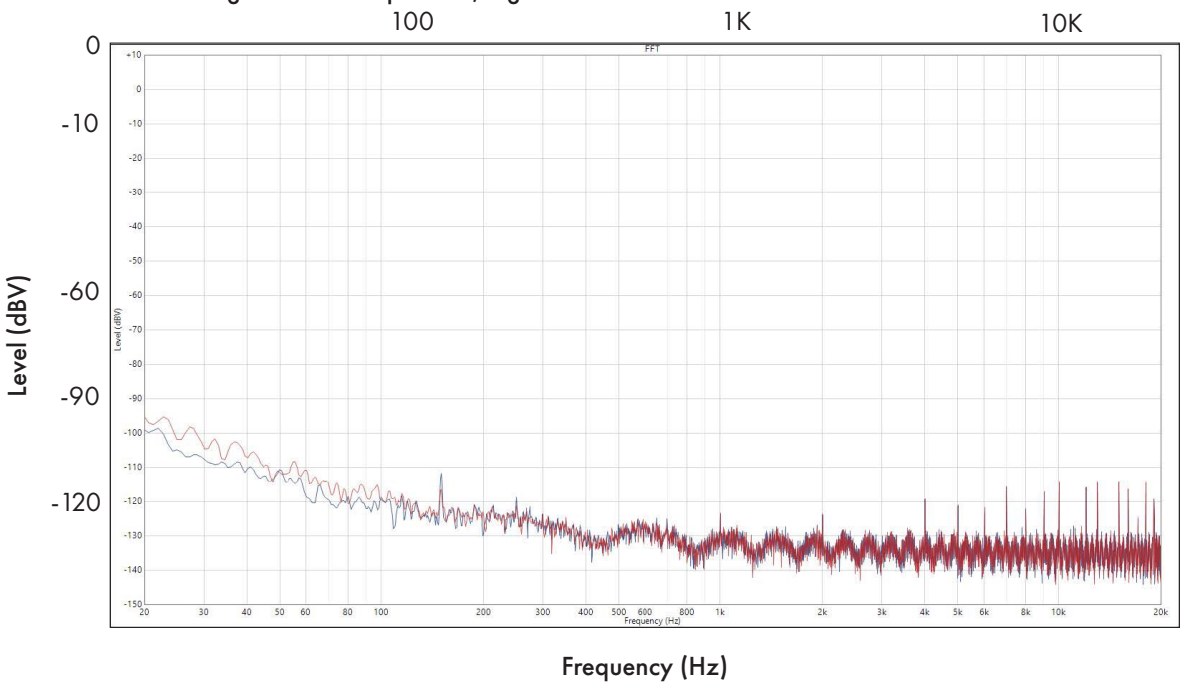
Audio Output: 220Hz



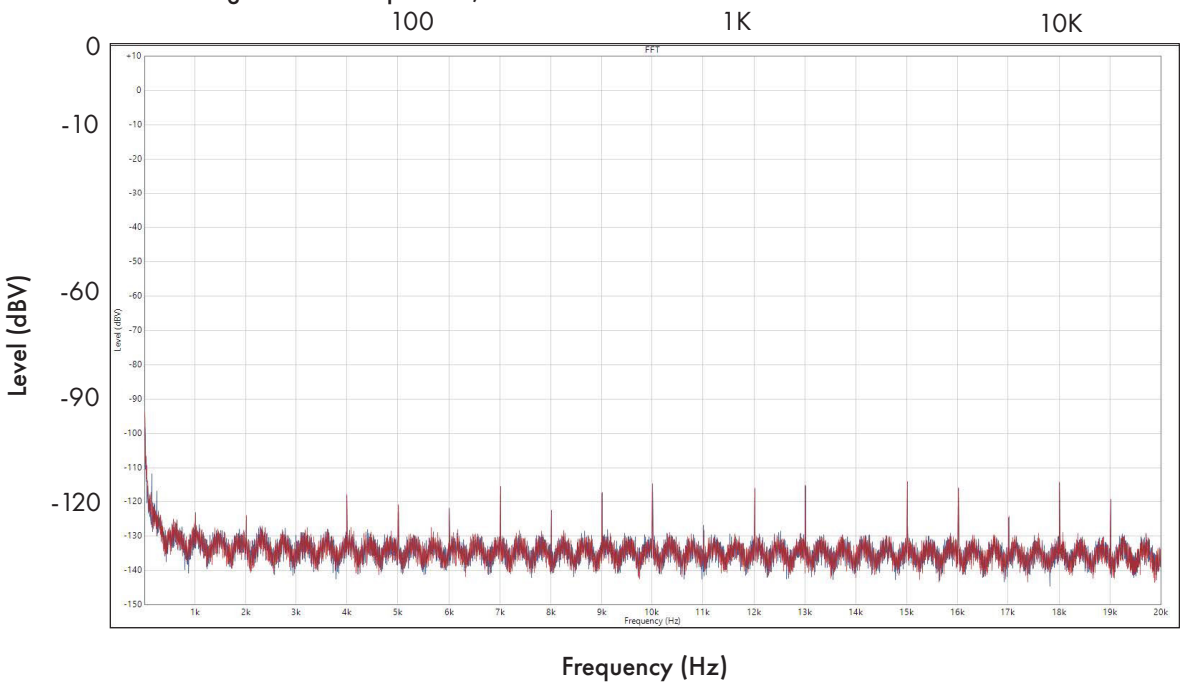
Instrument Level

Circuit: Typical app circuit on Seed3 Pedal Dev Kit
Measured with: Audio Precision APX555B Audio Analyzer

Audio Pass Through: Idle noise spectrum, logarithmic scale



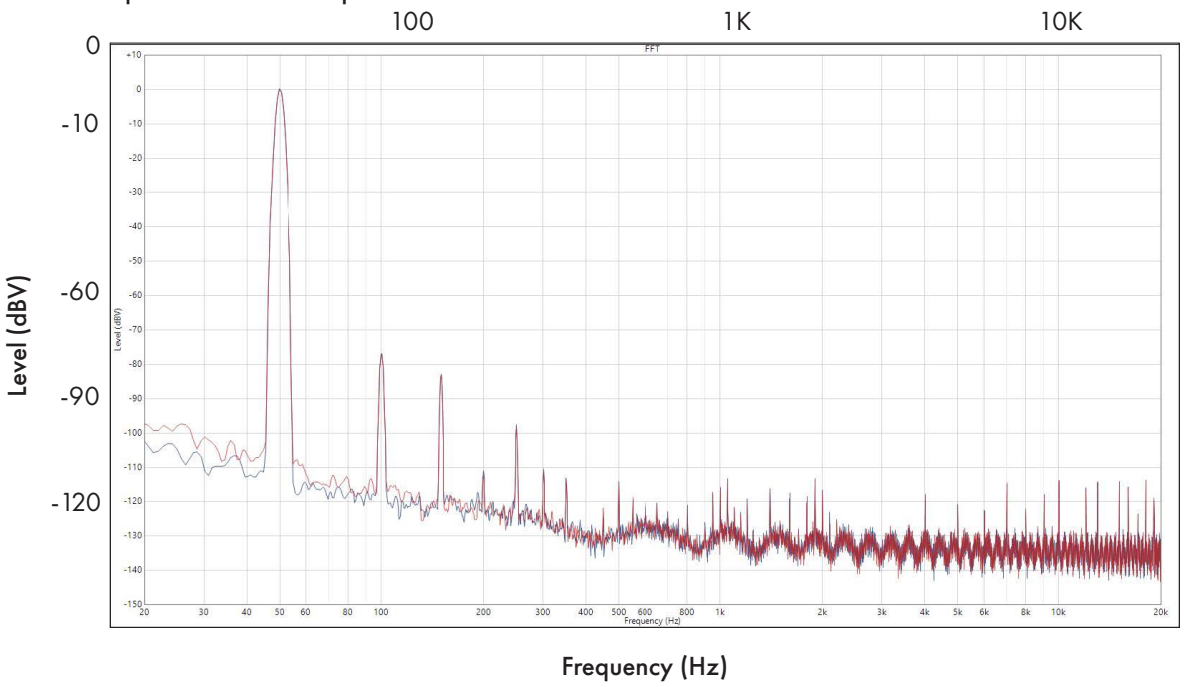
Audio Pass Through: Idle noise spectrum, linear scale



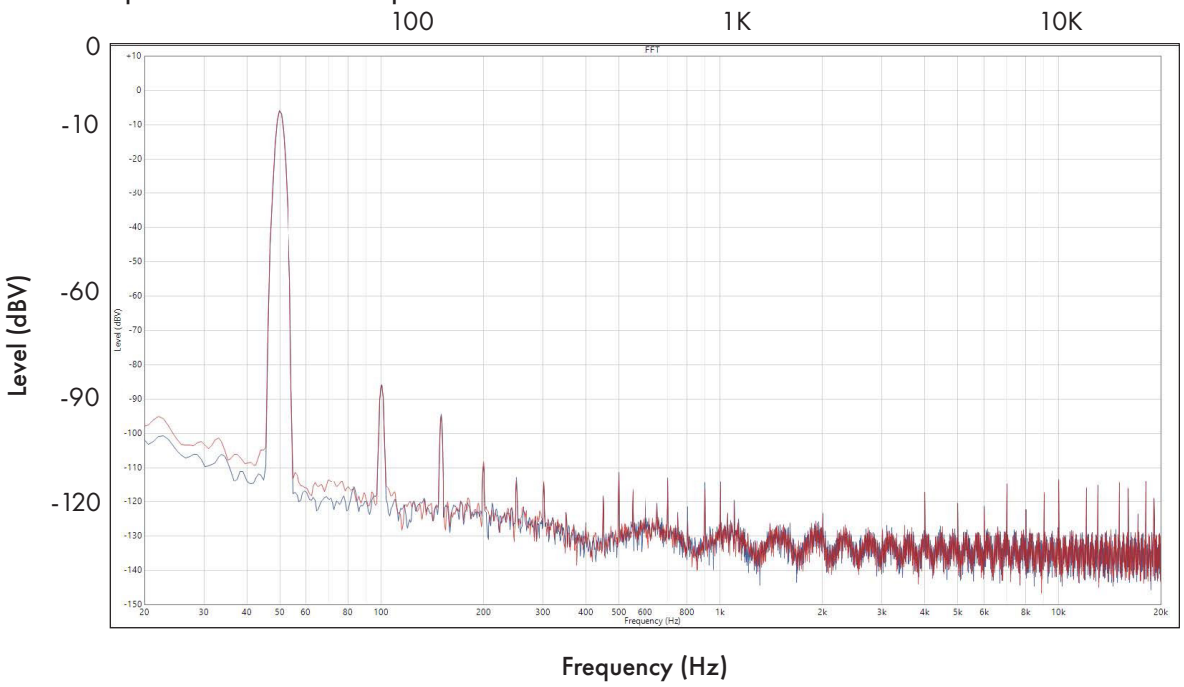
Instrument Level

Circuit: Typical app circuit on Seed3 Pedal Dev Kit
Measured with: Audio Precision APX555B Audio Analyzer

50HZ Spectrum: 1V RMS Input



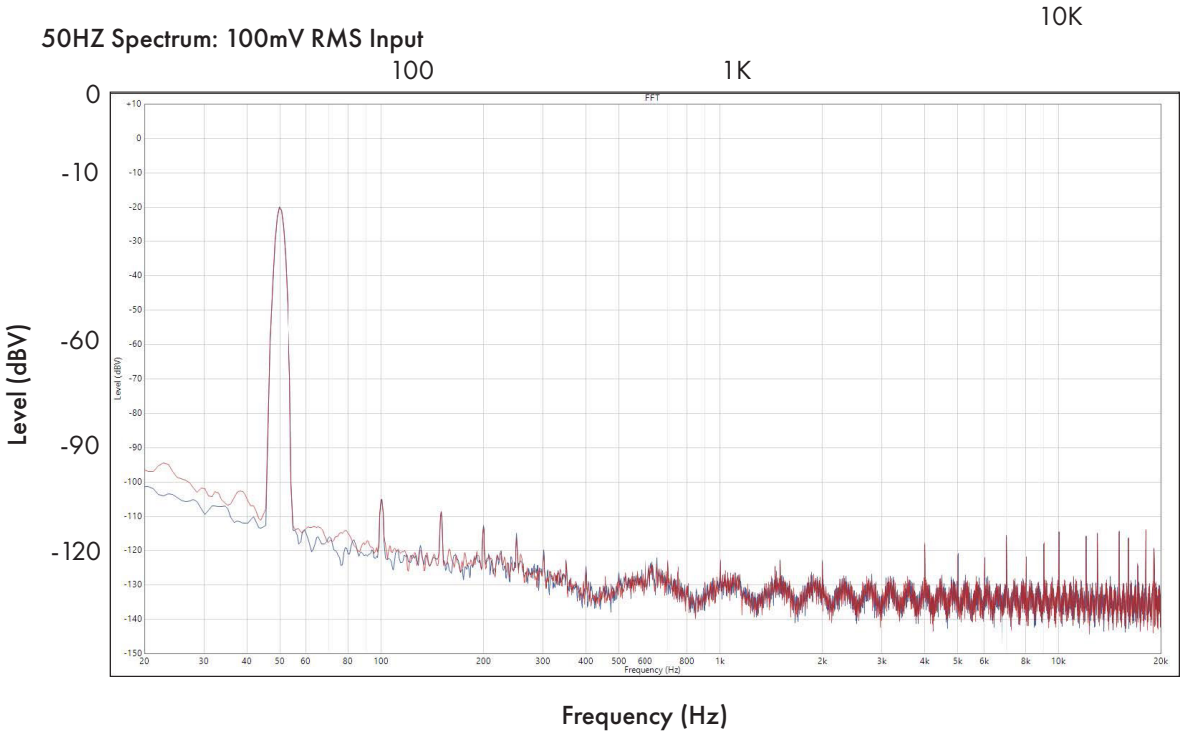
50HZ Spectrum: 500mV RMS Input



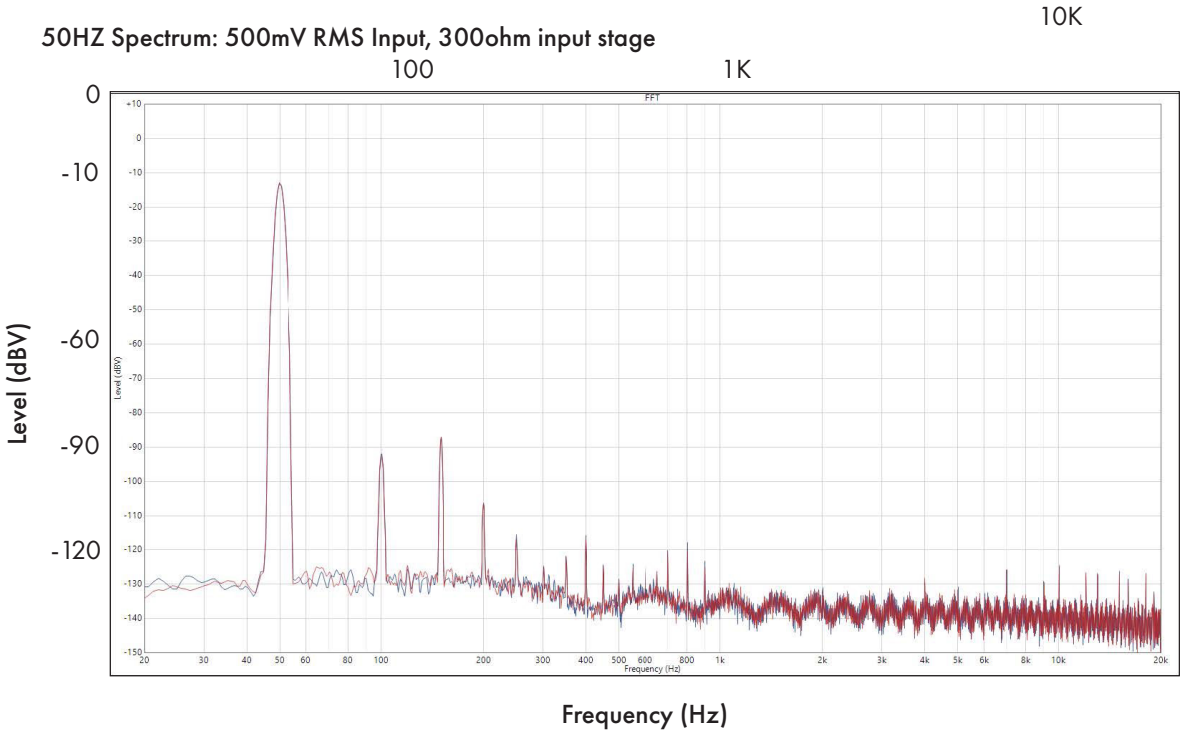
Instrument Level

Circuit: Typical app circuit on Seed3 Pedal Dev Kit
Measured with: Audio Precision APX555B Audio Analyzer

50HZ Spectrum: 100mV RMS Input



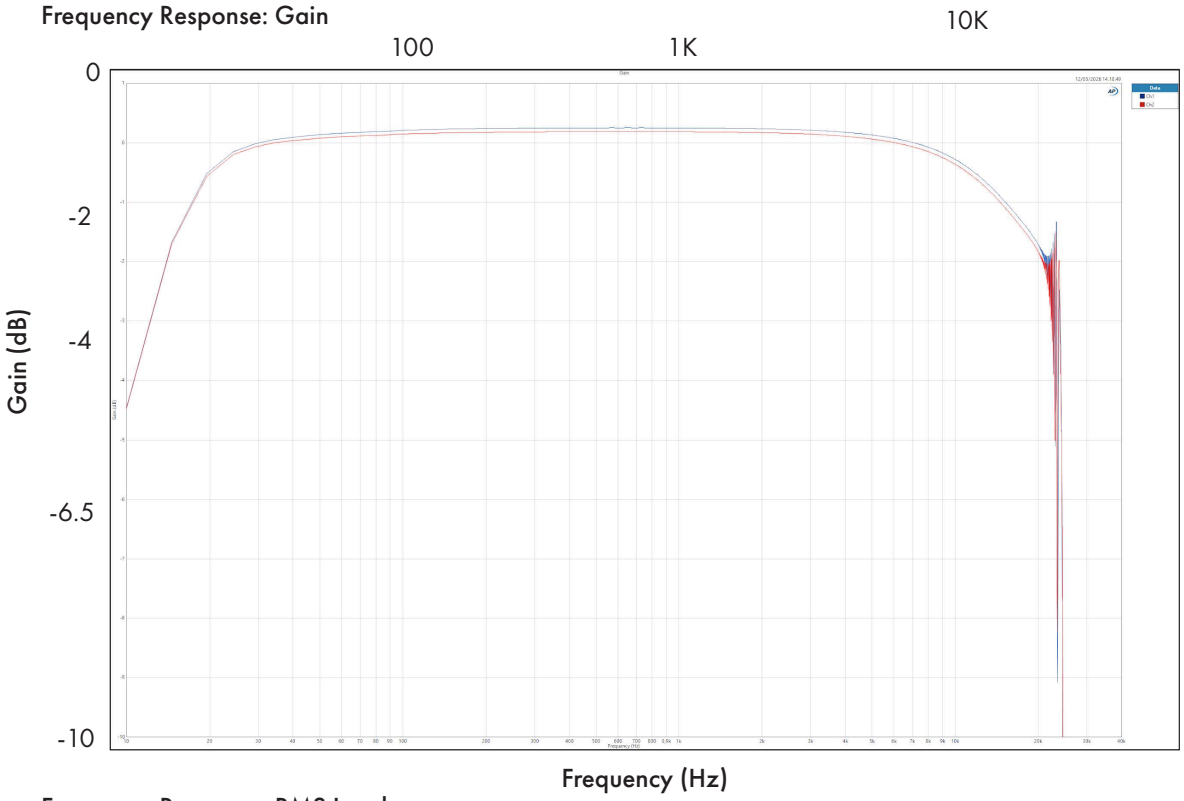
50HZ Spectrum: 500mV RMS Input, 300ohm input stage



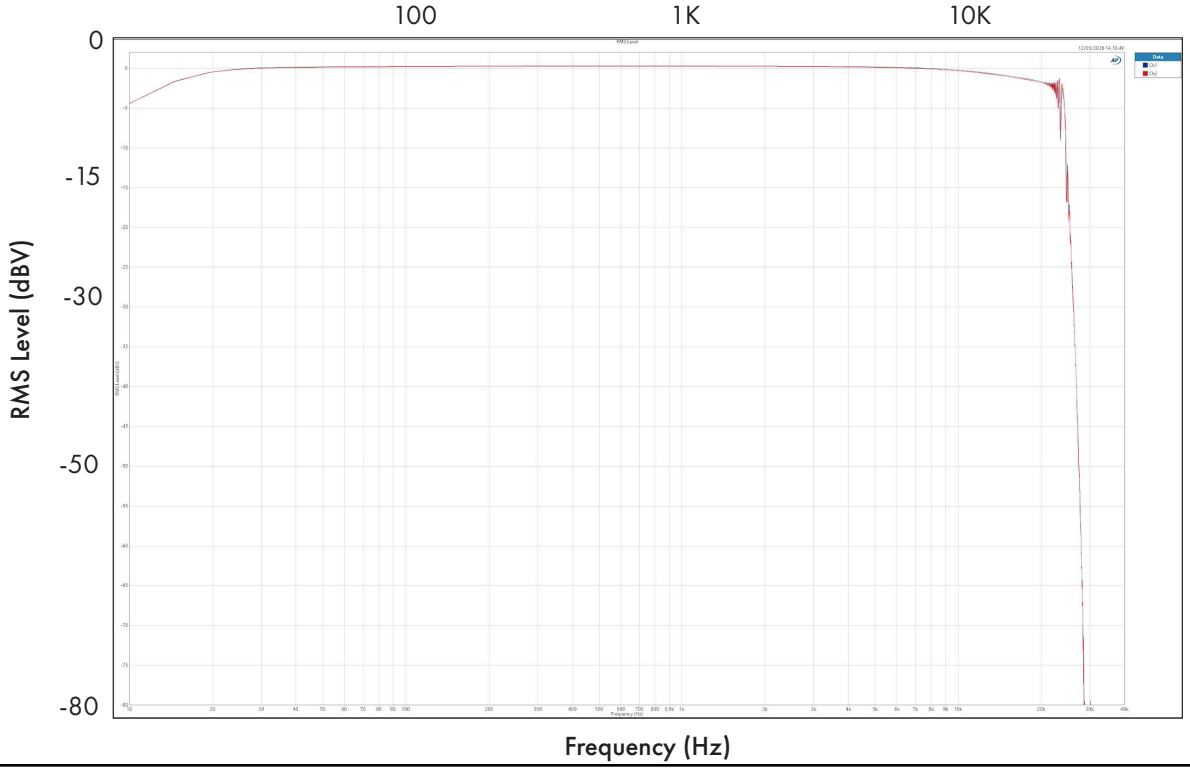
Instrument Level

Circuit: Typical app circuit on Seed3 Pedal Dev Kit
Measured with: Audio Precision APX555B Audio Analyzer

Frequency Response: Gain

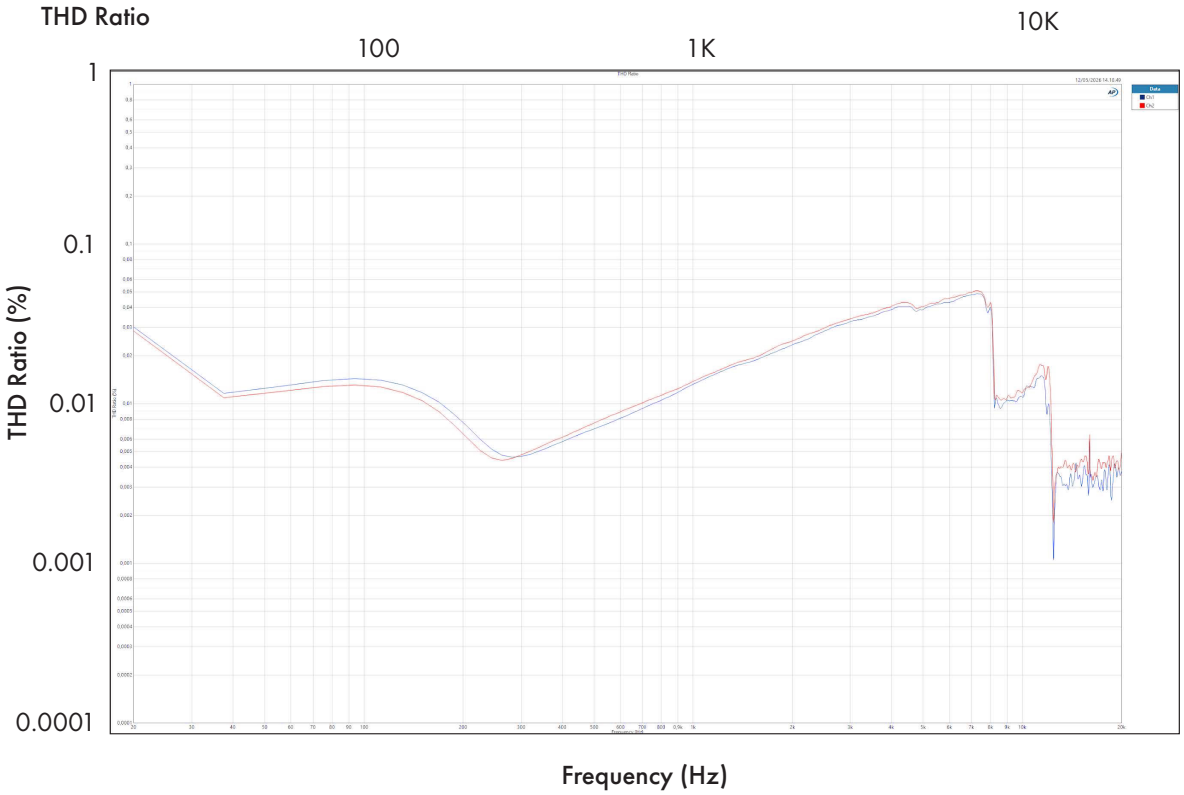


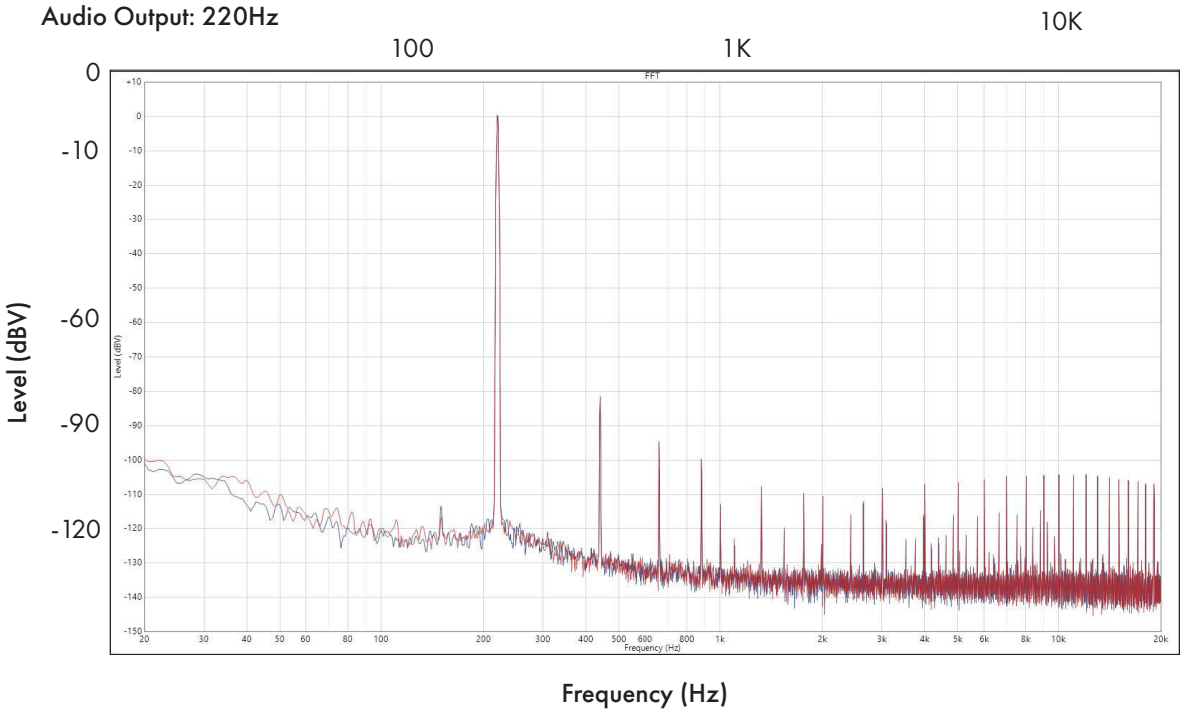
Frequency Response: RMS Level



Instrument Level

Circuit: Typical app circuit on Seed3 Pedal Dev Kit
Measured with: Audio Precision APX555B Audio Analyzer

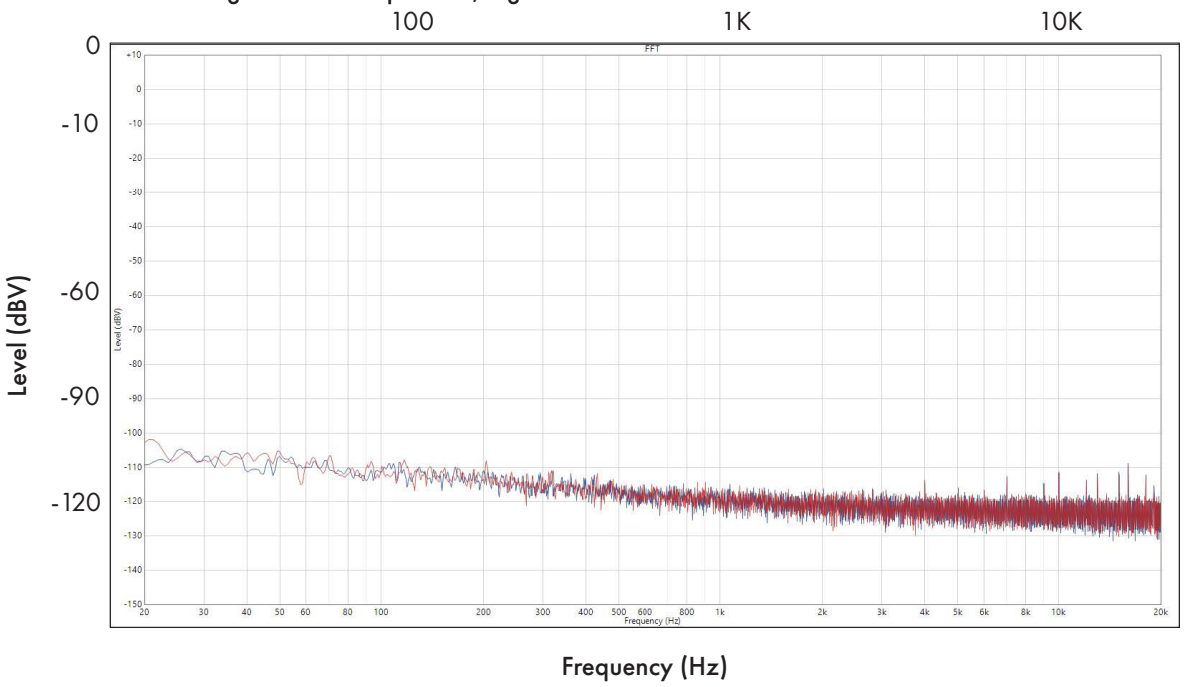




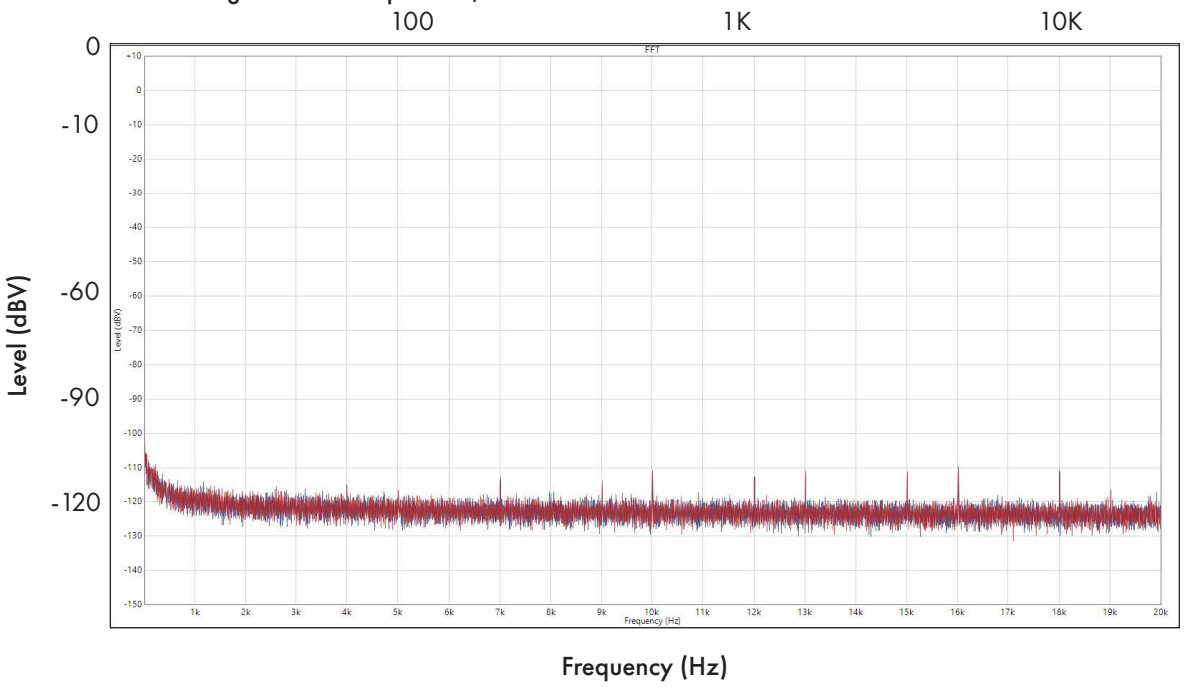
Modular Level

Circuit: Typical app circuit on Seed3 Eurorack Dev Kit
Measured with: Audio Precision APX555B Audio Analyzer

Audio Pass Through: Idle noise spectrum, logarithmic scale

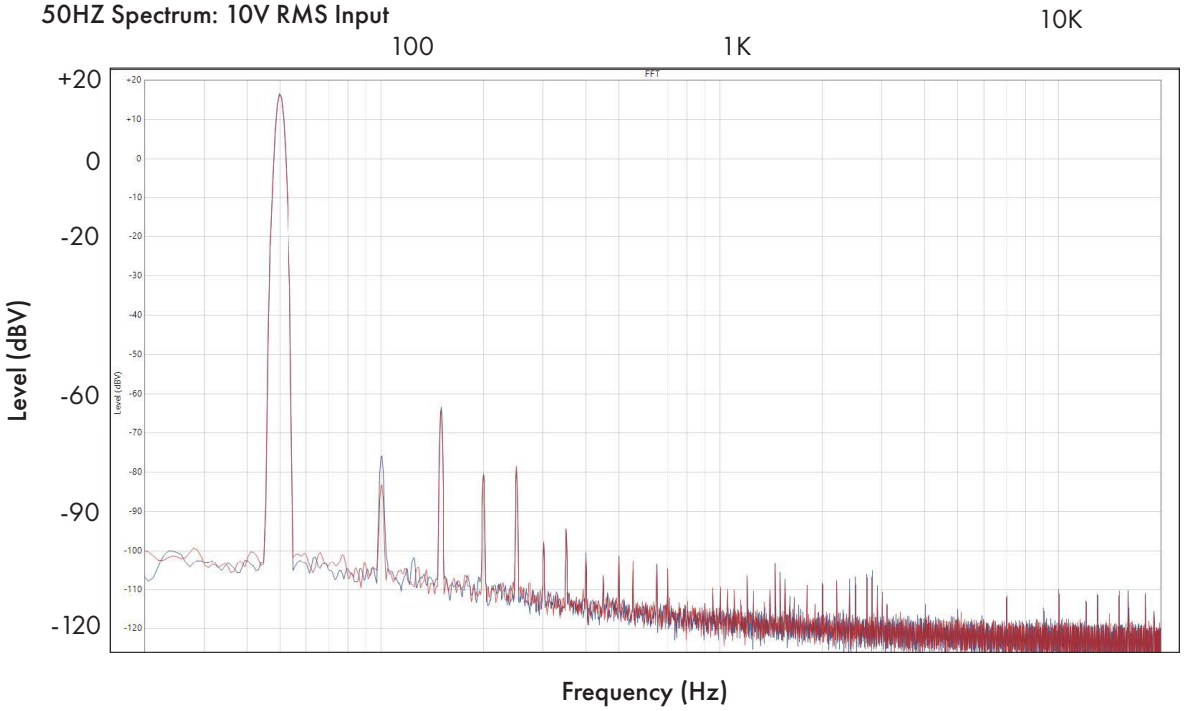


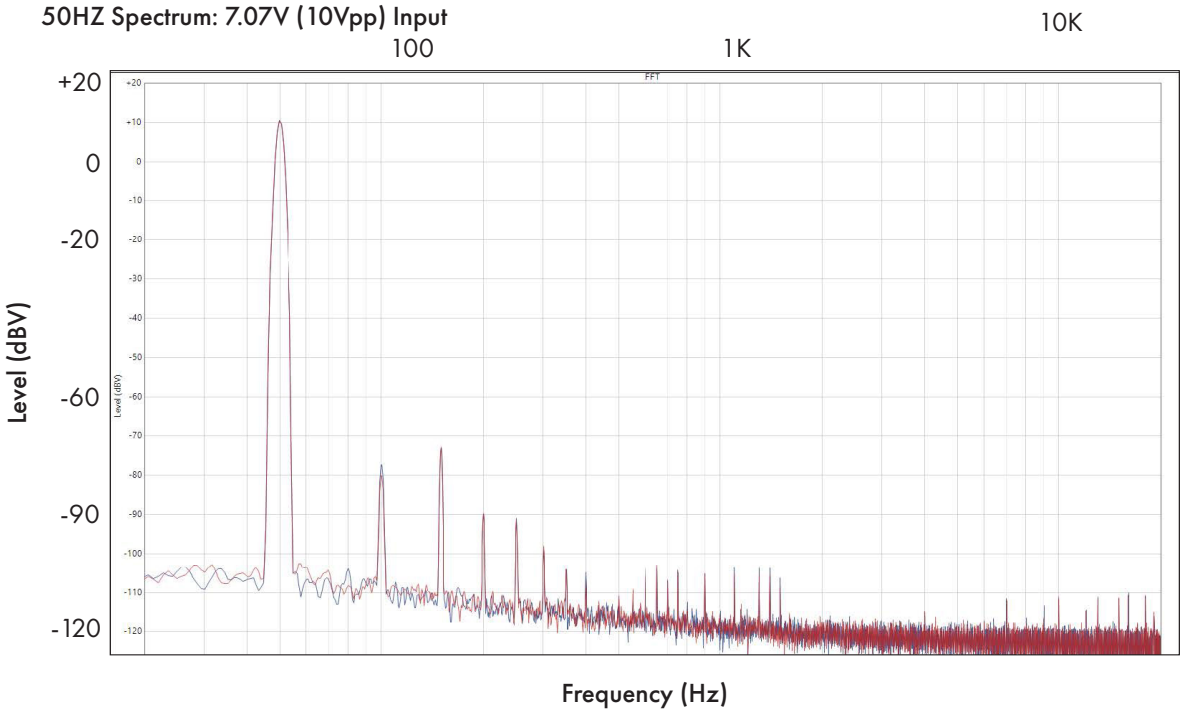
Audio Pass Through: Idle noise spectrum, linear scale



Modular Level

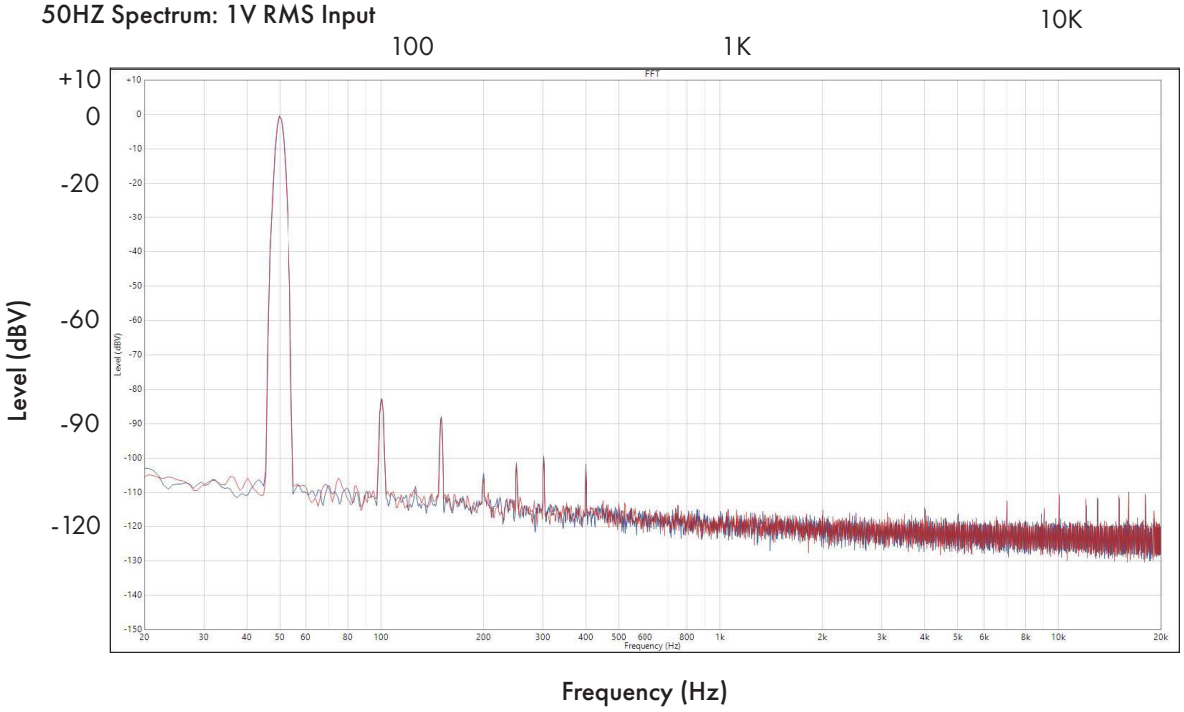
Circuit: Typical app circuit on Seed3 Eurorack Dev Kit
Measured with: Audio Precision APX555B Audio Analyzer

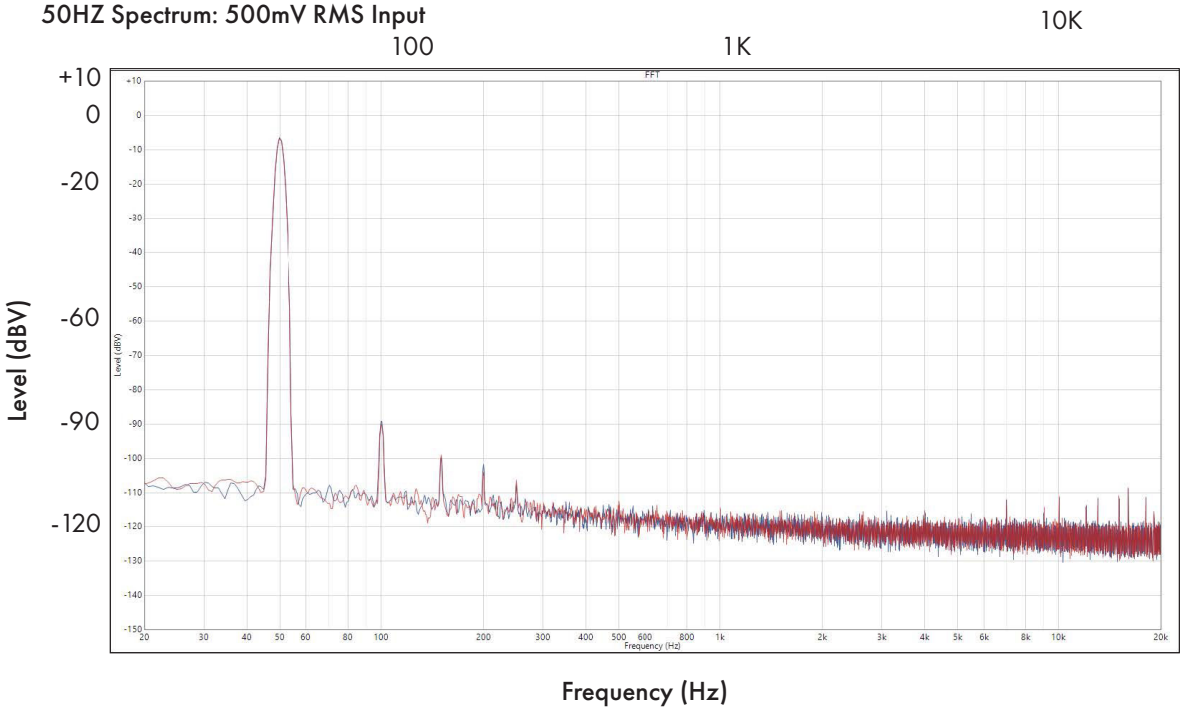




Modular Level

Circuit: Typical app circuit on Seed3 Eurorack Dev Kit
Measured with: Audio Precision APX555B Audio Analyzer

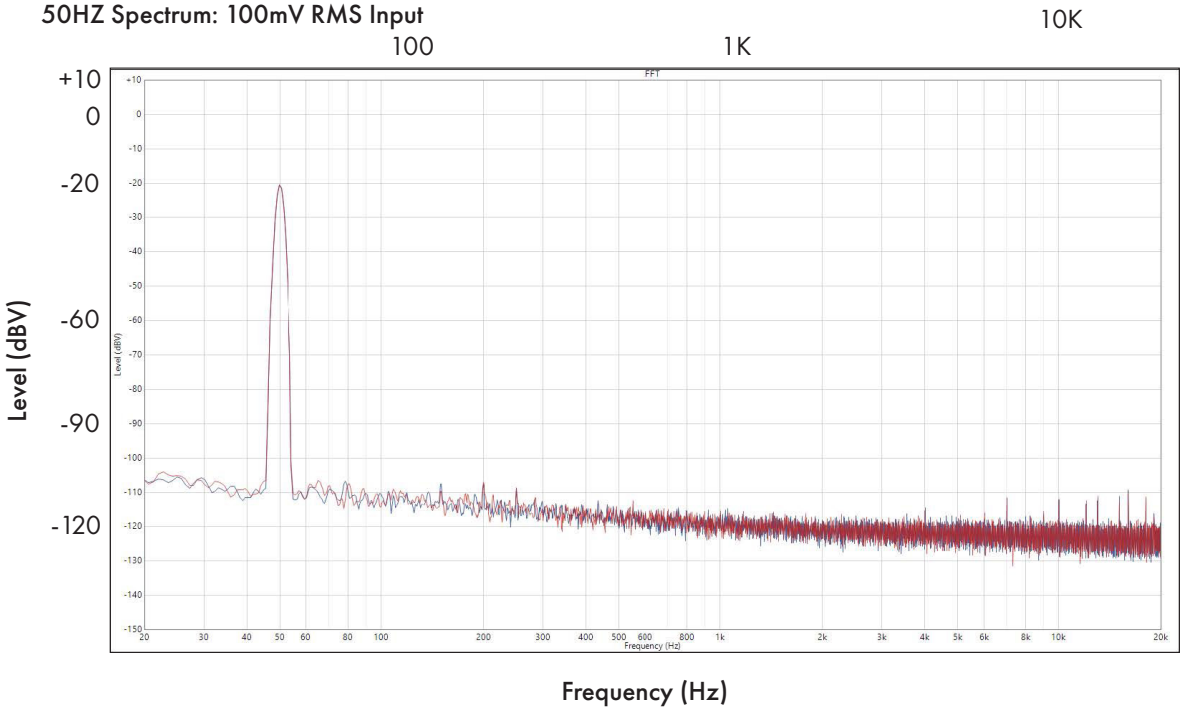




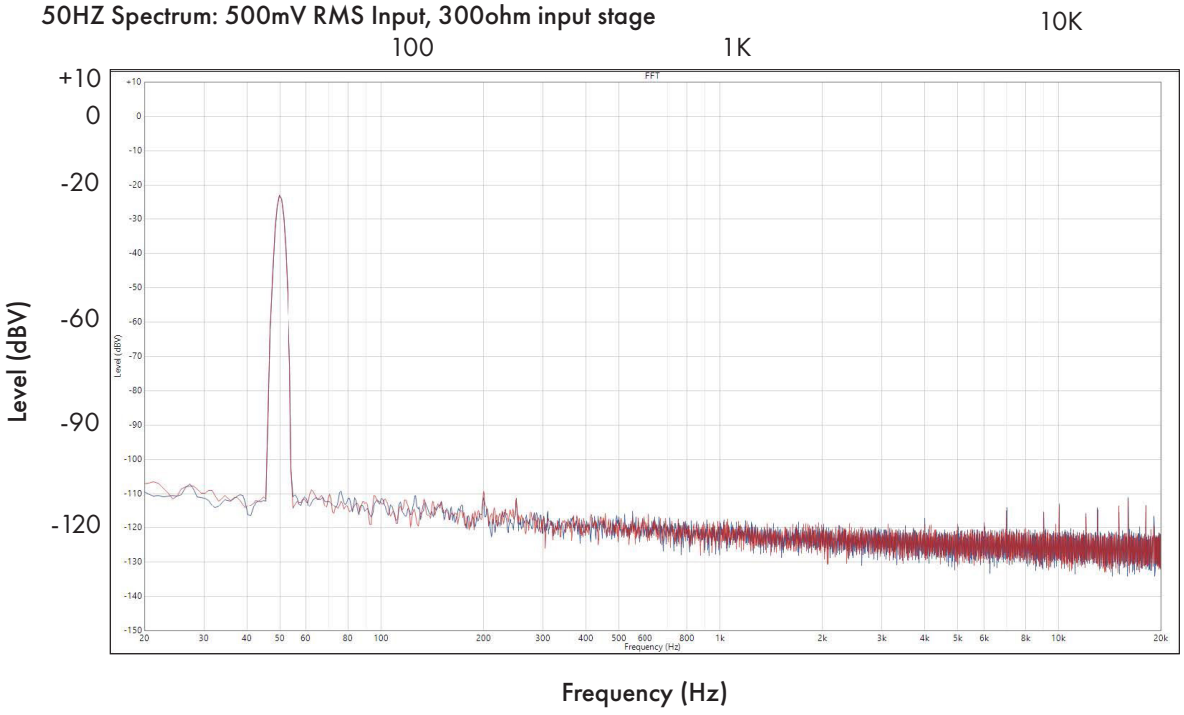
Modular Level

Circuit: Typical app circuit on Seed3 Eurorack Dev Kit
Measured with: Audio Precision APX555B Audio Analyzer

50HZ Spectrum: 100mV RMS Input



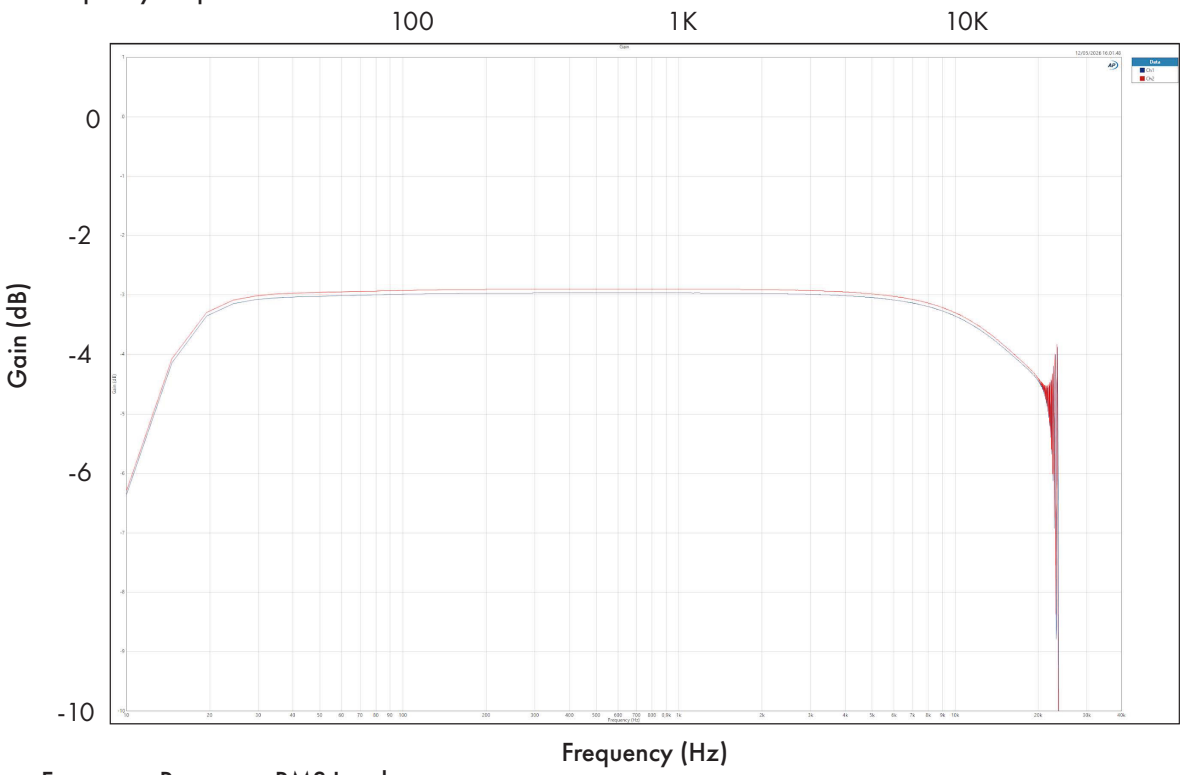
50HZ Spectrum: 500mV RMS Input, 300ohm input stage



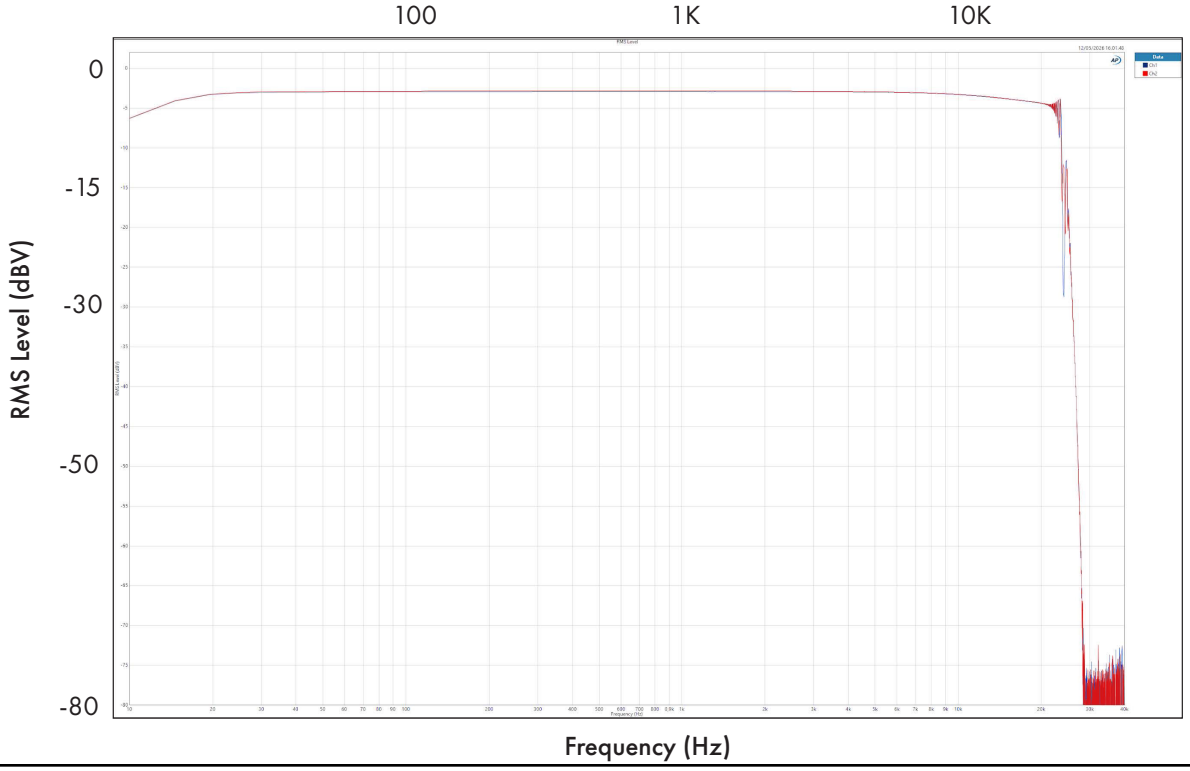
Modular Level

Circuit: Typical app circuit on Seed3 Eurorack Dev Kit
Measured with: Audio Precision APX555B Audio Analyzer

Frequency Response: Gain



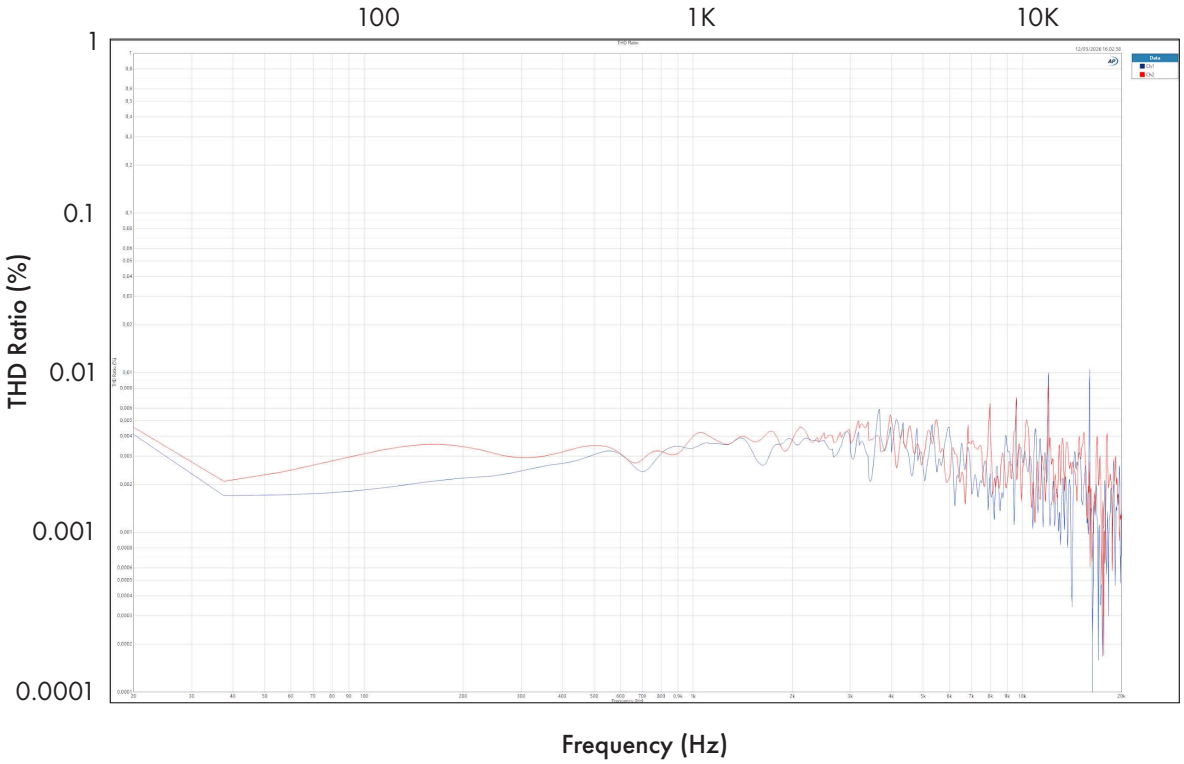
Frequency Response: RMS Level



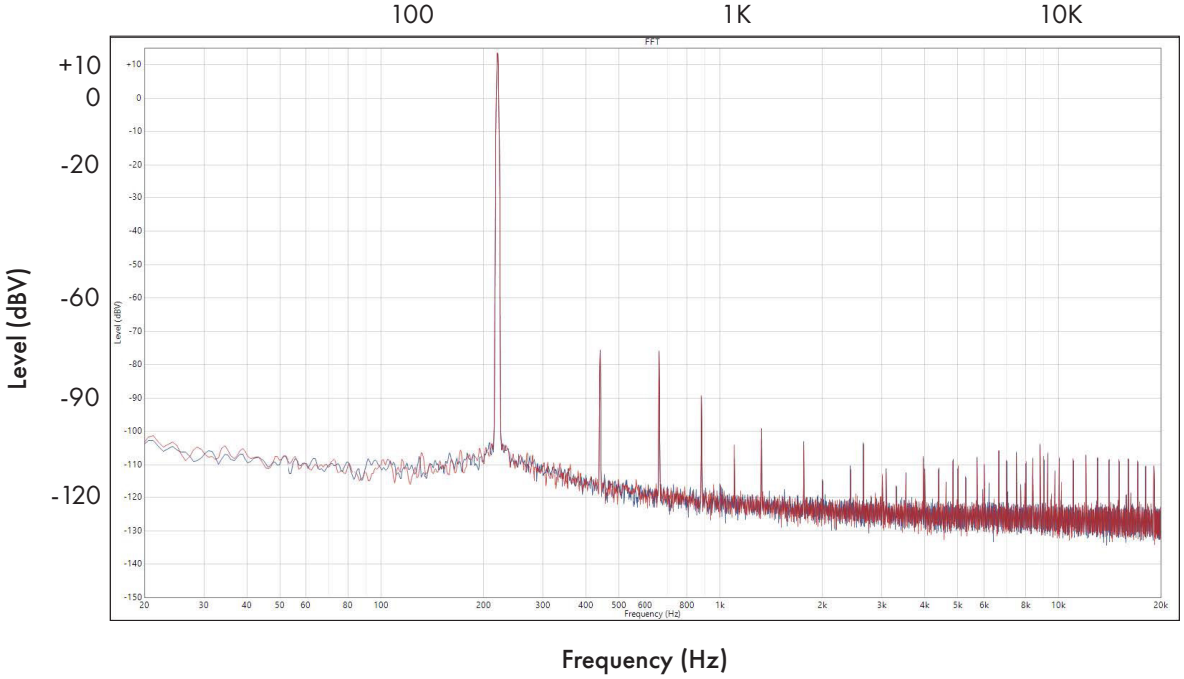
Modular Level

Circuit: Typical app circuit on Seed3 Eurorack Dev Kit
Measured with: Audio Precision APX555B Audio Analyzer

THD Ratio



Audio Output: 220Hz



An ADC is an Analog to Digital Converter. This allows outside voltages to be measured, and read by software.

Most commonly, this could be potentiometers controlling parameters of some synthesizer, or a control voltage input from another synthesizer used for modulation.

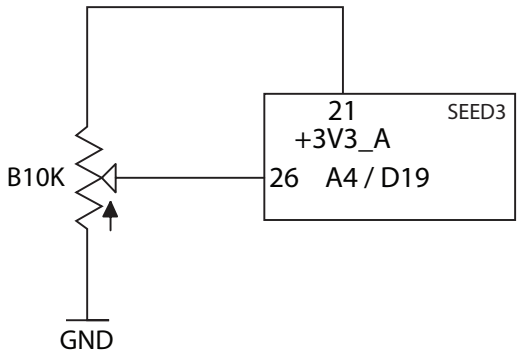
12 Pins on the Seed are connected to 16-bit ADCs.
The read speeds can be configured individually for each pin, and there is an ADC-wide oversampling mechanism that can be applied to all of the reads across the configured pins.

Below, we'll get into some of the common circuits used to read from various devices. But it's worth noting that in all cases we are taking some external signal with a known range, and converting that to a signal that will appear at the Seed's pin between 0V and 3.3V. Once that signal is attached to the pin, and the ADC is enabled, the value can be read in software with a sixteen bit value of 0 corresponding to 0V, and 65535 (2^{16}) corresponding to +3.3V. The exact range, accuracy, and response will depend on the ADC configuration, the number of pins used, and the electrical characteristics of whatever analog signal is connected to the pin.

As a minimal example, we can take a look at this circuit from the Pedal DevKit where there are a few pots connected directly to the ADCs.

Here is a more minimal view of what's going on there:

Figure 4.1 - Potentiometer Available Pins: Any ADC



Once this is wired up, we can read from them in C++ like this:

```
hw.Init();

// Configure the ADC
AdcChannelConfig adc_cfg;
adc_cfg.InitSingle(seed::D15);
hw.adc.Init(&adc_cfg, 1);
hw.adc.Start();

uint16_t integer_read = hw.adc.Get(0); //< 0 to 65535
float decimal_read = hw.adc.GetFloat(0); //< 0.0 to 1.0
```

Typical Applications

Below you'll find a several typical application circuits. Unless otherwise specified, the +3v3A signal is provided from the 3v3A pin of the Daisy, and all MCP600x op-amps are powered from that voltage as well.

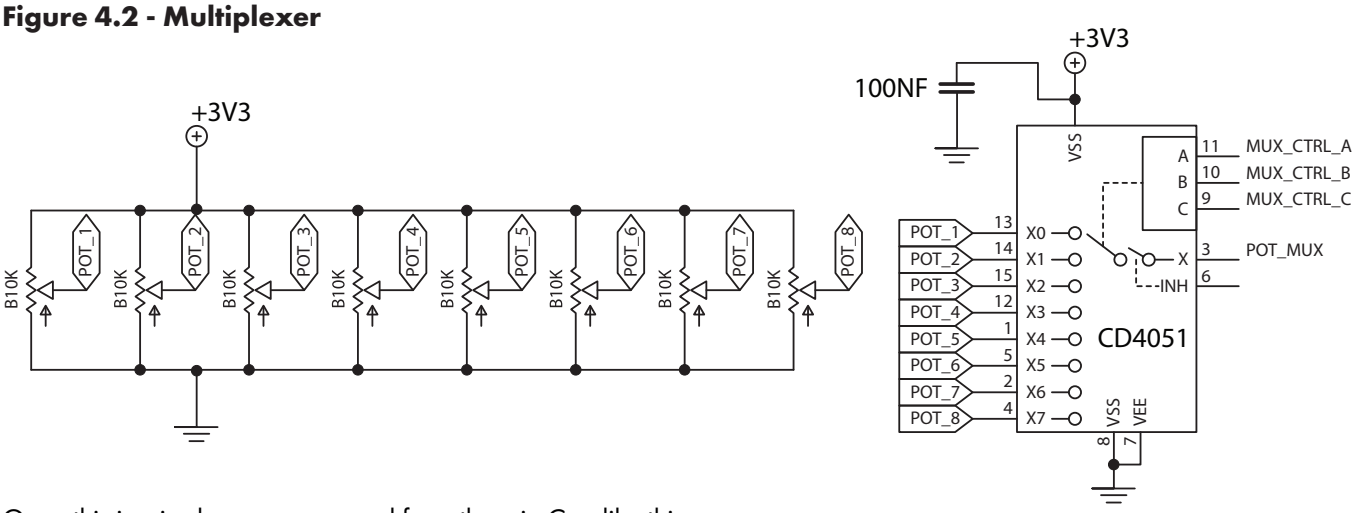
Multiplexed Pots

Pin Requirements:

- 3x GPIO pins
- 1x ADC pins
- CD4051 multiplexor

We wire the X output of the 4051 multiplexor to the Seed's ADC input, and wire the three GPIO pins to the A, B, and C pins so the Daisy can control which of the 8 inputs to read from.

Figure 4.2 - Multiplexer



Once this is wired up, we can read from them in C++ like this:

```
AdcChannelConfig adc_cfg;
adc_cfg.InitMux(seed::D15, seed::D24, seed::D25, seed::D26);
hw.adc.Init(&adc_cfg, 1);
hw.adc.Start();

// first pot connected to mux (as integer)
uint16_t integer_read = hw.adc.GetMux(0, 0);

// second pot connected to mux (as float)
float float_read = hw.adc.GetMuxFloat(0, 1);
```

This can be both reduced to require fewer pins, or expanded to support many more inputs with fewer pins:

- Using only two of the GPIO pins instead would still allow for four inputs, while requiring three pins.
- Multiple Mux chips can be controlled by the same 3 GPIO pins, which means you can add another eight inputs with one more ADC pin, and another CD4051.

It is worth mentioning that because the 8 inputs are connected a single ADC pin, each of those inputs will be read 8 times slower than another pin connected directly to the ADC.
For this reason, it is typical to directly connect control voltage signals that may need a higher input frequency response, while potentiometers, and other "slow" inputs can be connected to multiplexors as needed.

This is how the ADC in the Seed works as well, all twelve inputs are connected to the same peripheral, which is why the configuration of which channels, and their read speeds can affect the overall sample rate of all inputs that are enabled.

Bipolar CV Inputs (dual-supply)

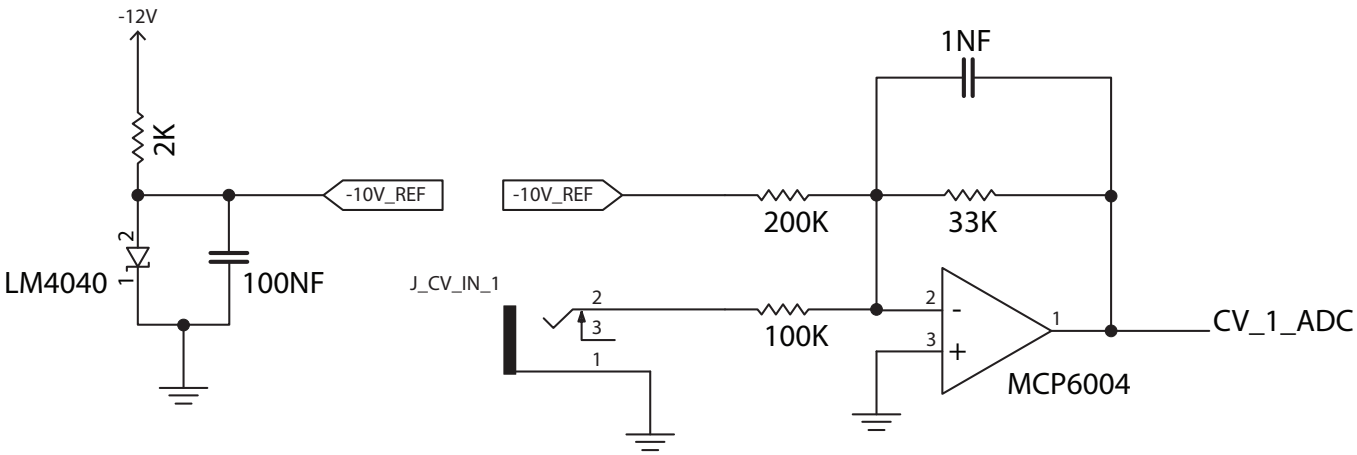
This circuit demonstrates a circuit configured for an input between the range of -5V to +5V, and converts that to +3.3V to 0V
You'll notice that the output is inverted, and the "center" point of 0V on the input ends up being around 1.6V

This circuit requires a negative reference voltage. Here we're using -10V, and we're generating it from an LM4040-10V voltage shunt. This is a constant current device that will generate a high precision, fixed voltage.

This circuit uses an MCP6004 powered at +3.3V. This opamp has a few unique characteristics that make it suitable for rail-to-rail inputs like this, and allow it to act as a clamp on the voltage that ensures the output won't exceed the 0V to +3v3 input range on the ADC pins.

When using an op-amp powered on a voltage above +3.3V, or a different type of op amp, you will want to handle clamping of both the input to the op-amp (to protect the op-amp), and the output (to protect the Daisy).

Figure 4.3 - Bipolar CV input / dual-supply

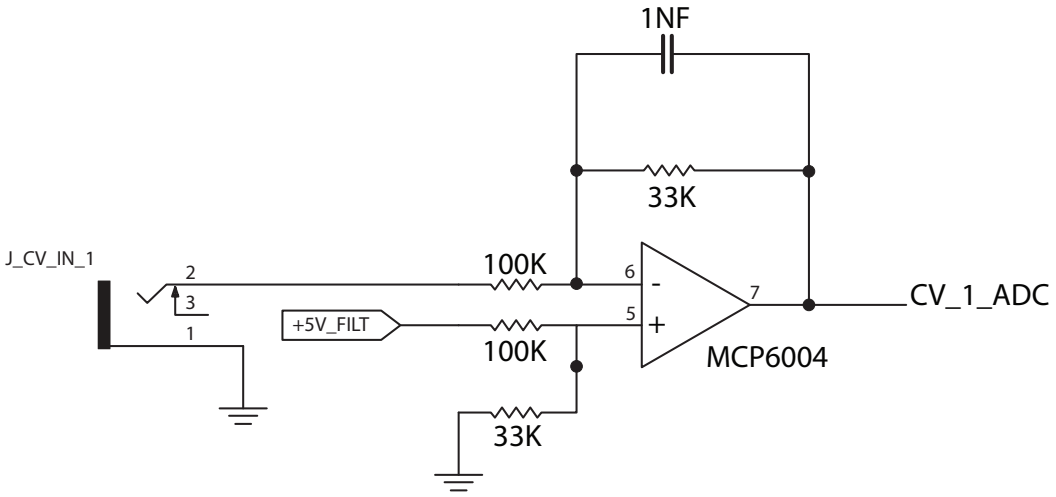


Bipolar CV Inputs (single-supply)

This circuit accomplishes the same goal as the dual-supply version above, but does not require a negative voltage to operate.

This configuration has slightly increased sensitivity to power supply noise due to the 3.3V bias being connected to the + input of the op-amp, and requires an additional passive compared to the other circuit.

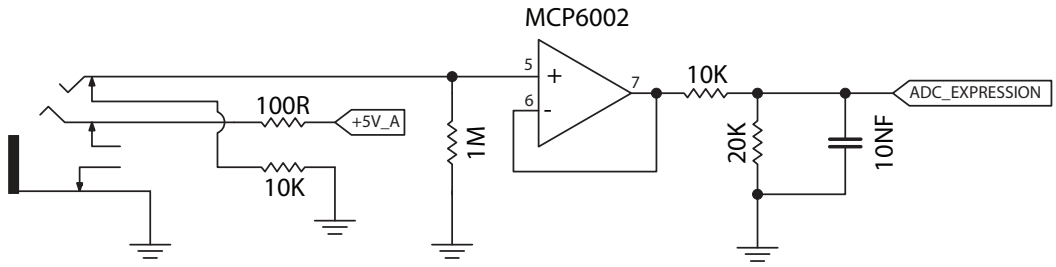
Figure 4.4 - Bipolar CV input / single-supply



Expression Input

Expression is an analog input commonly available on effects pedals. Here we provide the signals to the external pot on the foot pedal, and can then read that into Daisy

Figure 4.5 - Expression Input



The Seed has an SDMMC controller that can be used to connect to SD or microSD cards with up to 4 parallel data lines, and run at speeds up to 150MHz.

To achieve reliable 4-bit I/O at higher speeds, the layout of the traces between the Seed and the SD Card become very important.

Below you will find some guidelines that can help ensure the SD Card works as expected with 4-bit parallel I/O.

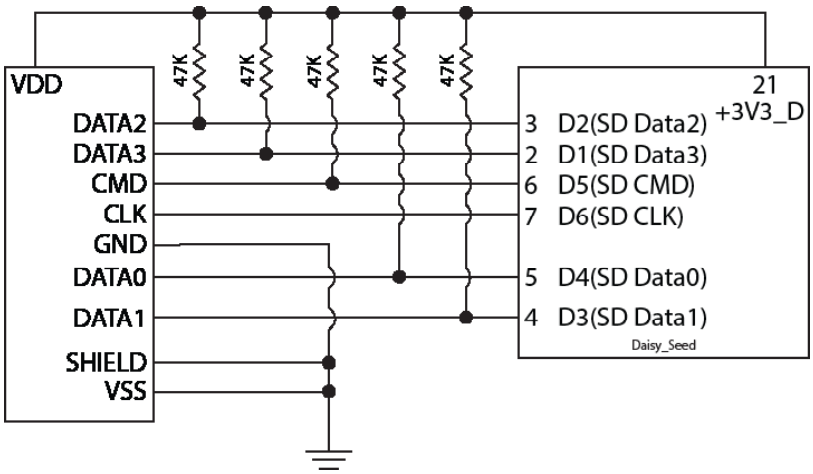
- Maximum difference in trace length should be kept below 10mm
- Maximum trace length of any signal should be kept below insert or termination resistors should be used.
- The number of vias between layers should be the same for each of the DATA and CMD lines.

The following table contains the relevant data from the Seed3 PCB layout necessary for achieving the guidelines laid out above.

HW Pin	SD Card Function	Trace Length (mm)	Via Quantity
7	SDMMC1_CK	4.853	0
6	SDMMC1_CMD	9.767	1
2	SDMMC1_D0	13.94	2
3	SDMMC1_D1	14.942	2
4	SDMMC1_D2	14.224	1
5	SDMMC1_D3	17.586	1

Figure 5.1 - Micro SD

47K pullup resistors necessary, except for Pin 7.



For a 1-bit SDMMC configuration, disconnect D1, D2, and D3 from the Seed3 in the above circuit.

The Seed has a 12-bit DAC that can be used to generate an analog voltage between 3.3V.
This DAC has two channels that are connected to pins D22, and D23.

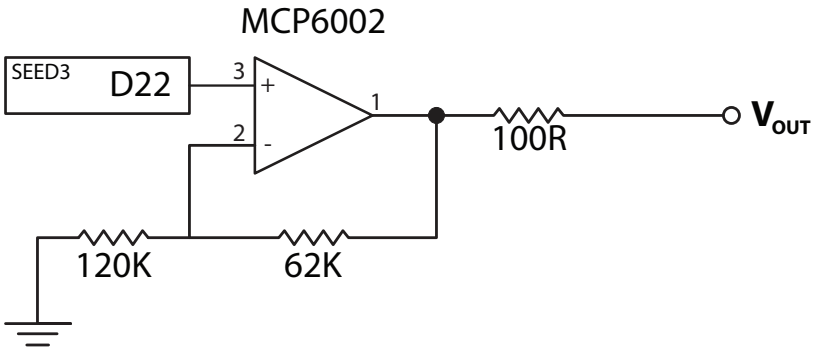
Typical Application
It is quite common to amplify the DAC output. Most commonly, this is typically boosting from +3v3 to +5V.

This is demonstrated below, but this is a generic non-inverting op-amp circuit, and the gain can be adjusted by recalculating the resistors with this formula:

$$G = 1 + \left(\frac{Rf}{R2} \right)$$

Figure 6.1 - DAC

Available Pins: D22, D23



MIDI is available on the Seed through either of the USB ports, or any of its UART ports.

If you have a functional USB port wired up, or are using the built-in USB, then MIDI is purely handled in software, and your hardware is already set up.

On the other hand, UART MIDI can take a few different forms, and requires a pretty specific circuit.

Typical Application

UART MIDI Input

This circuit, connected with an optocoupler, receives the MIDI from a transmitting device, and creates a valid UART signal for the Seed to process.

In both examples, the input names correspond to the original 5-pin DIN pin number connections to illustrate that only the connector can change while the rest of the circuit can remain the same.

Figure 7.1 - MIDI input circuit for +3.3V supply

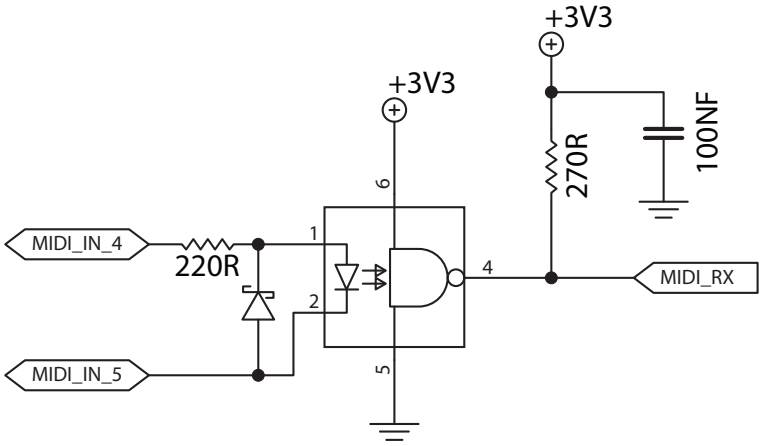


Figure 7.2 - 5-pin MIDI input

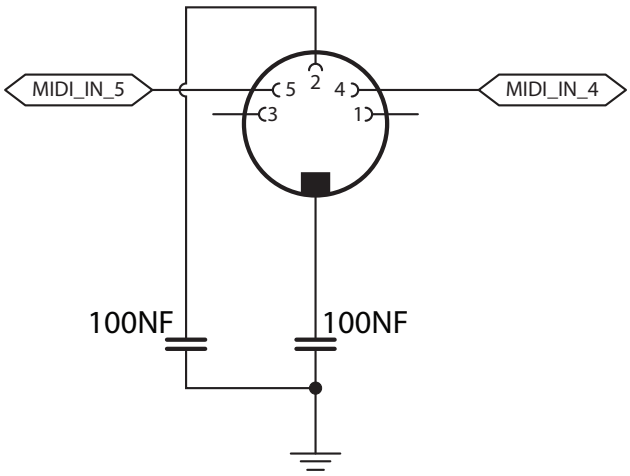
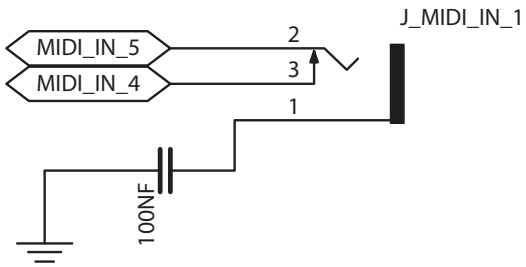


Figure 7.3 - TRS MIDI input



UART MIDI Output

MIDI output is surprisingly simple. We connect one pin to a UART output on the Seed, through a 10 ohm resistor, and then connect the other pin to 3.3V through a 10 ohm resistor.

Show both the DIN and TRS connections.

Figure 7.4 - 5-pin MIDI output

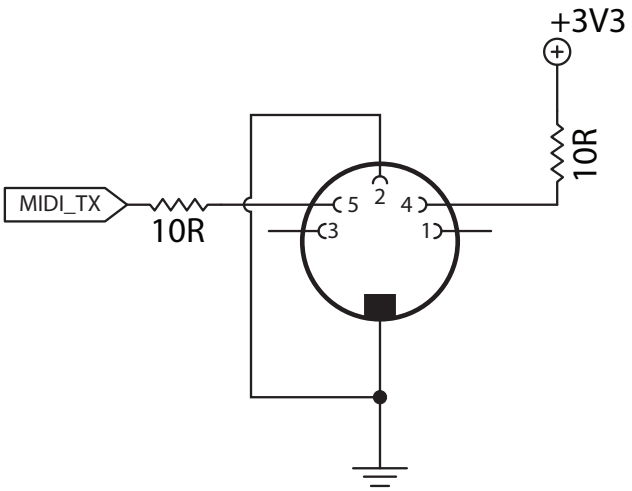
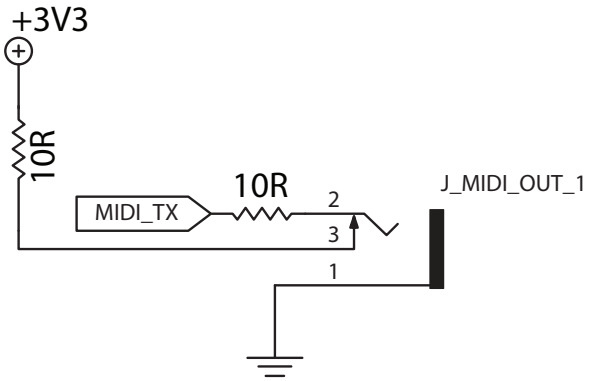


Figure 7.5 - TRS MIDI output



MIDI Thru

Since the MIDI Input is a current loop between devices it can't be passively split like some other signals. Thankfully, the actual digital signal produced by the circuit can be.

For that reason, the MIDI Thru circuit will look nearly identical to the MIDI Output circuit, except that we are using the MIDI Rx signal produced by the MIDI Input circuit instead of the Tx signal produced by Daisy.

Figure 7.6 - 5-pin MIDI thru

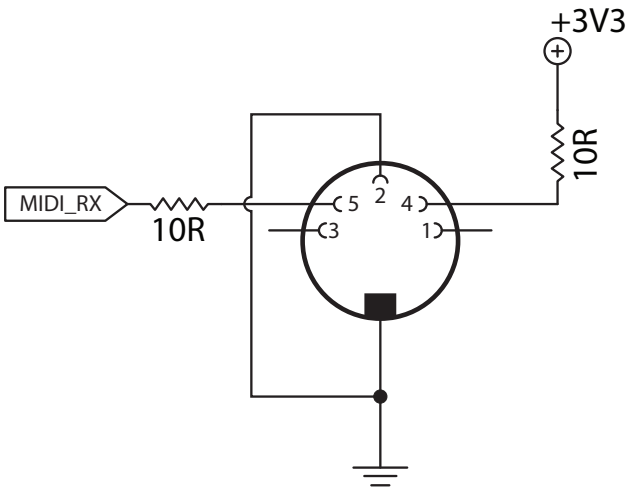
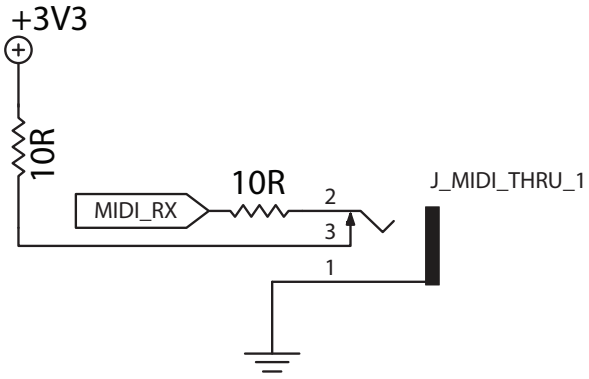


Figure 7.7 - TRS MIDI output

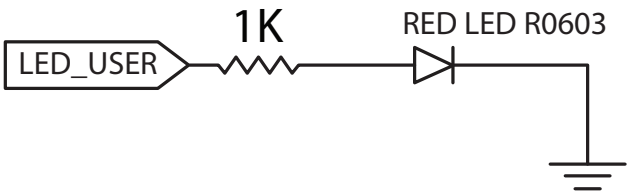


The Seed has an LED by the USB connector that is accessible to applications as a GPIO.

This LED is connected to the STM32's pin: PC7

It is connected through a 1K resistor as shown here:

Figure 8.1 - LED



In most integrations for creating firmware for the Seed, this LED will already be configured, and initialized. For example, in C++ this brief example will blink the LED once per second:

```
#include "daisy_seed.h"

using namespace daisy;

DaisySeed seed;

int main(void) {
    seed.Init();
    while(true) {
        seed.SetLed(true);
        System::Delay(500);
        seed.SetLed(false);
        System::Delay(500);
        }
    }
```

RELEASE	DATE	DESCRIPTION
v1.0	23/July/2026	• release version of datasheet