Evaluation Board User Guide

EVAL-ADAU1797Z

Evaluating the ADAU1797 High-Performance Audio Codec with Integrated HiFi 3z and FastDSP Cores

GENERAL DESCRIPTION

This user guide explains the setup of the EVAL-ADAU1797Z evaluation board.

EVALUATION BOARD

This evaluation board provides full access to all analog and digital inputs/outputs on the ADAU1797. The EVAL-ADAU1797Z can be powered by a single 3.8V to 5V supply. Once the main power is supplied, onboard regulators provide the voltages needed for the ADAU1797 and other devices. The printed circuit board (PCB) is an 8-layer design, with a ground plane and power plane on the inner layers. The EVAL-ADAU1797Z contains connectors for external microphones and speakers. The master clock can be provided externally or by the onboard 24.576MHz passive crystal or onboard 24.576MHz oscillator.

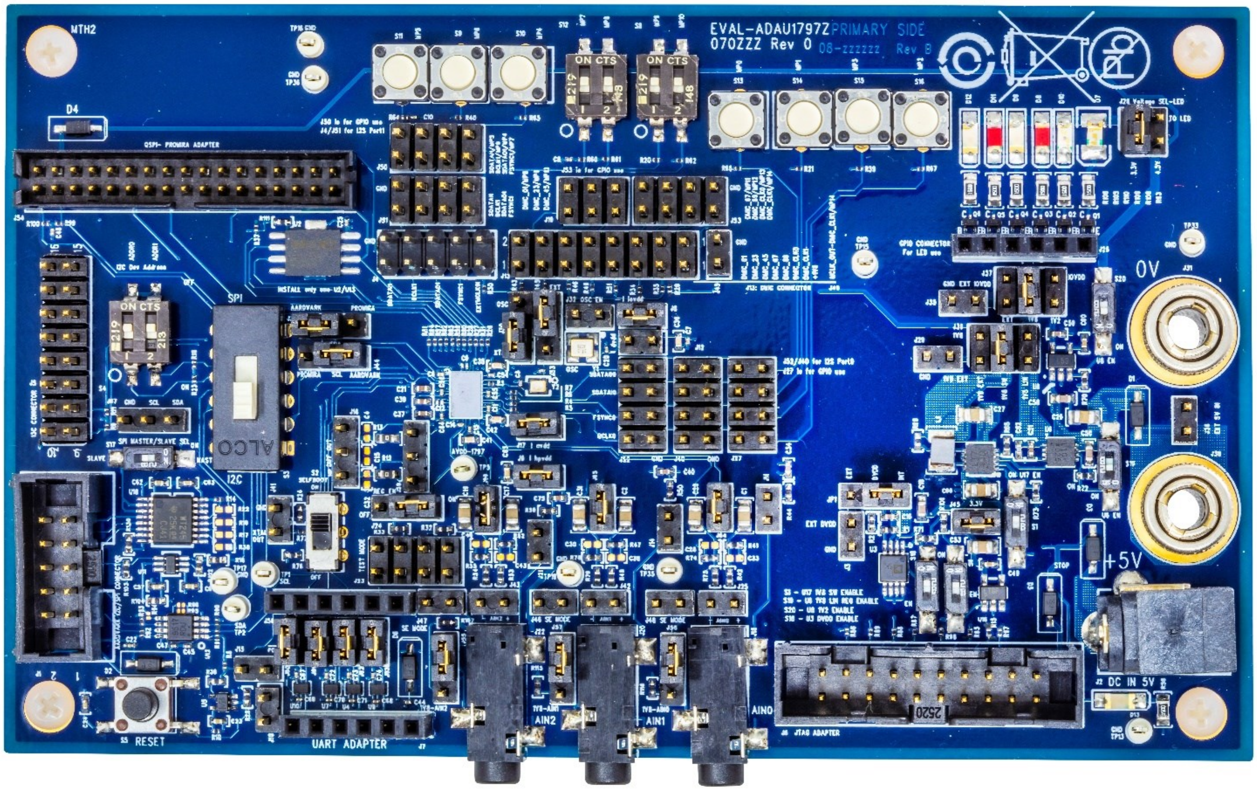


Figure 1. EVAL-ADAU1797Z Evaluation Board Photograph

One Analog Way, Wilmington, MA 01887-2356, U.S.A. | Tel: 781.329.4700

EVALUATION BOARD BLOCK DIAGRAM

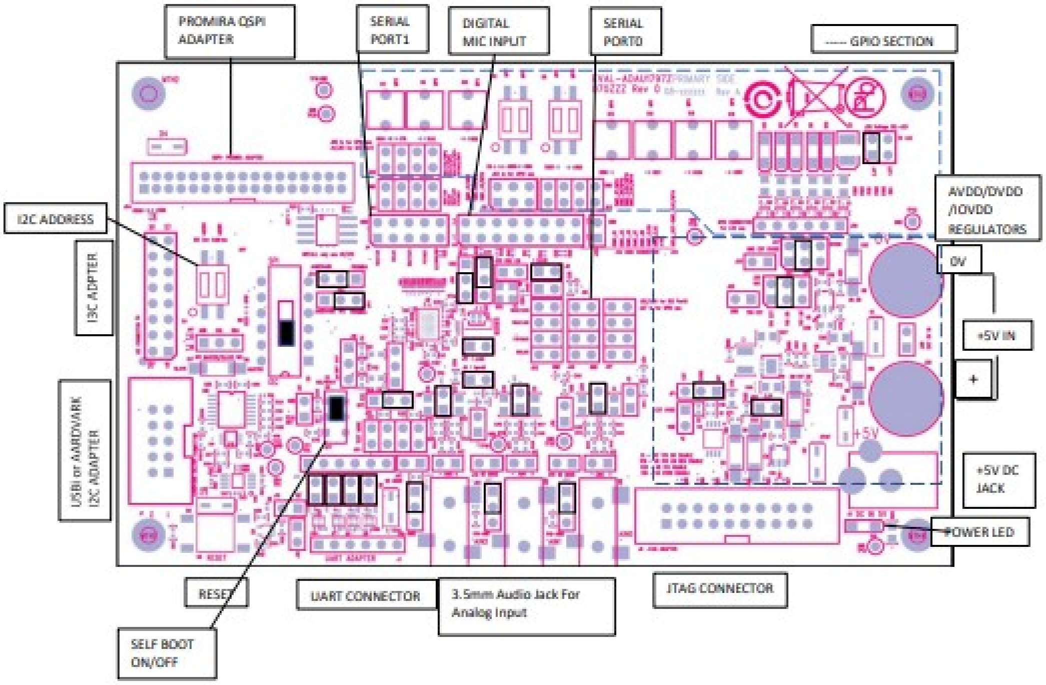


Figure 2. EVAL-ADAU1797Z Evaluation Board Block Diagram

QUICK START

Required Equipment

- 5V Power supply through lab supply, SEGGER J-Link adapter, or Aardvark adapter

HEADERS AND JUMPERS

The EV kit is fully assembled and tested. Follow the steps to make the required hardware connections and start operation of the kit.

1. To communicate with the part over I^{2}C/SPI, the USBi port (J1) provided on the evaluation board can be used with a USBi or Aardvark adapter.
2. The EEPROM Flash can be programmed by using the QSPI 34-pin header (J54) on the evaluation board. If using the Aardvark adapter to program with QSPI, a Level shifter board is required to convert 3.3V signals to 1.8V.
3. To communicate with the part through UART, the UART adapter port (J7) is provided on the evaluation board and can be used with a PMOD USB UART adapter.
4. The JTAG header (J6) on the evaluation board is available to connect to external JTAG adapters such as the Segger J-Link adapter. This enables C programming of the part using Xtensa Xplorer.
5. *Figure 3* shows the default jumper configurations for using the evaluation board. By default, the DVDD is generated internally by the ADAU1797. The part is in I^{2}C mode in this configuration.

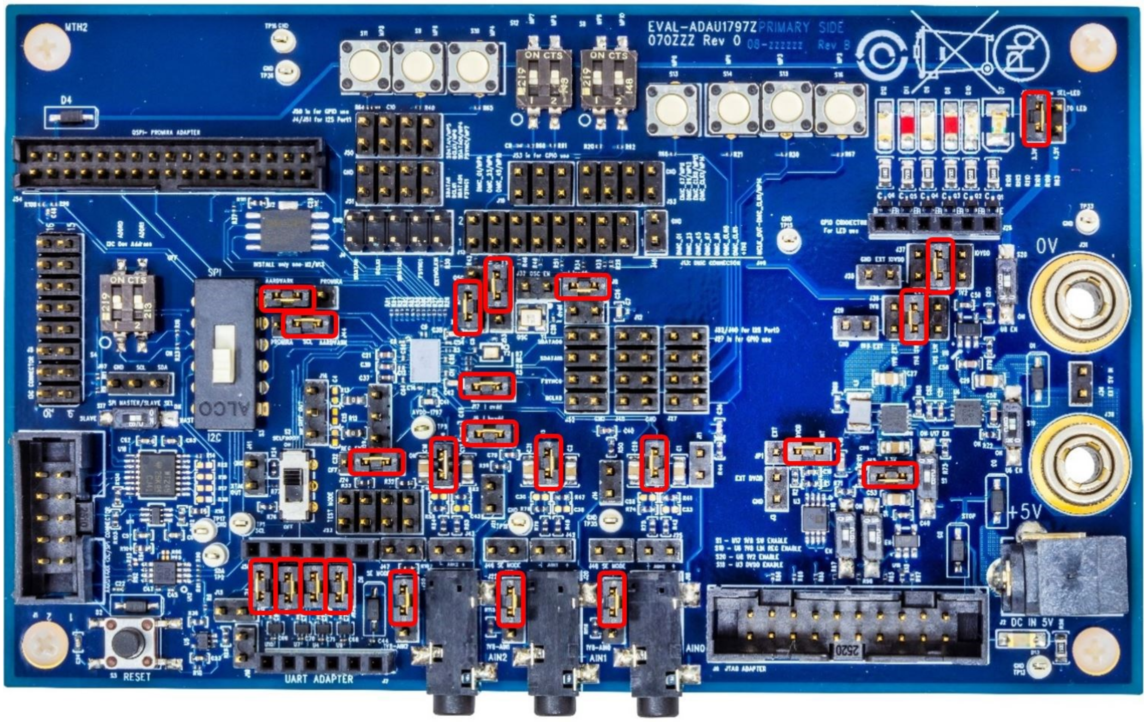


Figure 3. EVAL-ADAU1797Z Evaluation Board Default Jumper Connections

SETUP AND OPERATION

An example script is provided below. The script is executed in the Total Phase Aardvark I2C adapter. This script configures the following signal paths for quick hardware evaluation:

- ADC0/ADC1/ADC2 (Differential, no PGA) → FDSP (Pass-Through) → FDEC → ASRCO → SPT1
  - Note: ADCs run at 384kHz. FDSP runs at 384kHz. FDEC decimates down to 48kHz.
- SPT0 → ASRCI → FINT → FDSP (Pass-Through) → HPOUT
  - Note: SPT0 at 48kHz. FINT interpolates up to 384kHz. FDSP and HPOUT run at 384kHz.
- DMIC0/1 → PDMOUT
  - Note: DMIC0/1 run at 384kHz, DMIC_CLK = 6.144MHz. PDMOUT runs at 384kHz, PDM_CLK = 12.288MHz

```
<?xml version="1.0"?>

<aardvark>

<configure pullups="1" tpower="0" gpio="0" spi="0" i2c="1"/>

<i2c_bitrate khz="400"/>

<sleep ms="10"/>

<!-- Power Up -->

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 3C 01 00 00 00</i2c_write>   <!-- DVDD ON power_en=1 -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 2C 21 00 00 00</i2c_write>   <!-- master block en 1, CP_EN =1 -->

<sleep ms="10"/>

<!-- PLL  -->

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 40 08 07 00 00</i2c_write>   <!-- CLK_CTRL0    set MCLK pll source, INT mode, intern sync, XTAL MODE-->

<sleep ms="10"/>
```

```
<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 44 10 00 00 00</i2c_write>   <!-- CLK_CTRL1    set PLL_INPUT_PRESCALER = 16 -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 48 80 00 00 00</i2c_write>   <!-- CLK_CTRL2    set PLL_INTEGER_DIVIDER = 128  -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 54 01 00 00 00</i2c_write>   <!--  CLK_CTRL5   PLL_UPDATE = 1 -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 30 03 00 00 00</i2c_write>   <!--  PLL_MB_PGA_PWR   PLL_EN = 1 PGA_EN-->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 34 10 00 00 00</i2c_write>   <!-- PROC_EN = 1 -->

<sleep ms="10"/>

<!-- Read PLL lock in F0000408-->

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 04 08 </i2c_write>   <!-- PLL Lock status -->

<sleep ms="10"/>

<i2c_read addr="0x2B" count="4"/> <!-- check the value 0x00 00 01 00 if pll locked -->

<sleep ms="10"/>

<!-- check the Power_up_complete bit value -->

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 04 0C </i2c_write>

<sleep ms="10"/>

<i2c_read addr="0x2B" count="4"/> <!-- Read the Power_up_complete bit's value, bit 0 -->

<sleep ms="10"/>

<!—Master Block Enable -->

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 2C 23 00 00 00</i2c_write>   <!-- cm_startup_over=1, master_block_en=1, CP_EN =1 -->

<sleep ms="10"/>

<!-- ADC setup  -->

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 58 07 07 00 00</i2c_write>   <!-- 384KHz sample rate for ADC0-->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 5c 07 00 00 00</i2c_write>   <!--  ADC_CTRL2    ADC0/1/2 Differential mode -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 0C 17 00 00 00</i2c_write>   <!--  3 ADC enabled + PB0_EN -->

<sleep ms="10"/>

<!-- DMIC0/1 and DMIC_CLK0 SETUP  -->

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 28 F0 00 00 00</i2c_write>  <!-- SAI_CLK_PWR    DMIC0/1_CLK_EN = Enabled -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 10 03 00 00 00</i2c_write>  <!-- DMIC_PWR    DMIC0/1_EN = Enabled -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 A0 04 00 00 00</i2c_write>  <!-- DMIC_CTRL1    DMIC_CLK0_RATE = 6.144MHz -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 A4 27 00 00 00</i2c_write>  <!-- DMIC_CTRL2   DEC_ORDER = 5th, DMIC01_FS = 384kHz -->

<sleep ms="10"/>
```

<!-- PDM OUTPUT setup -->

```
<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 05 90 07 00 00 00</i2c_write>   <!--  PDM_CTRL1  12.288MHz,384kHz-->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 05 94 04 00 00 00</i2c_write>   <!--  PDM_CTRL2  PDM1_0_MUTE = 0, PDM_VOL_ZC = 1 -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 05 A4 3C 00 00 00</i2c_write>   <!--  PDM_ROUTE0    PDM_ROUTE0 = DMIC_0-->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 05 A8 3D 00 00 00</i2c_write>   <!--  PDM_ROUTE1    PDM_ROUTE1 = DMIC_1-->

<sleep ms="10"/>
```

<!-- Multi-Purpose PIN ROUTING -->

```
<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 02 5C 00 0D 00 00</i2c_write>   <!--  MP_CTRL2   MP5/SDATAI1 Mode = PDM Clock Output -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 02 58 0E 00 00 00</i2c_write>   <!--  MP_CTRL2   MP0/SDATAO_0 Mode = PDM Data Output -->

<sleep ms="10"/>
```

<!-- FDEC Setup -->

```
<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 18 47 00 00 00</i2c_write>   <!--  FDEC_CTRL1 fs = 384KHz and fs_out = 48KHz -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 28 01 00 00 00</i2c_write>   <!--  FDEC_ROUTE0 = FDSP_CH1  -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 2C 02 00 00 00</i2c_write>   <!--  FDEC_ROUTE0 = FDSP_CH2  -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 30 03 00 00 00</i2c_write>   <!--  FDEC_ROUTE0 = FDSP_CH3  -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 24 07 00 00 00</i2c_write>   <!--  FDEC_PWR 0/1/2 ON -->

<sleep ms="10"/>
```

<!-- ASRCO Setup -->

```
 <i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 84 14 00 00 00</i2c_write>   <!--  ASRCO0_CTRL  ASRCO_IN_FS = 48kHz -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 88 1E 00 00 00</i2c_write>   <!--  ASRCO0_ROUTE0  CH0 = FDEC CH0 -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 8C 1F 00 00 00</i2c_write>   <!--  ASRCO0_ROUTE0  CH0 = FDEC CH1 -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 90 20 00 00 00</i2c_write>   <!--  ASRCO0_ROUTE0  CH0 = FDEC CH2 -->

<sleep ms="10"/>
```

<!-- SPT0_INPUT Setup -->

```
<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 05 00 00 00 00 00</i2c_write>   <!-- SPT0_CTRL1, Stereo I2S-->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 05 04 00 00 00 00</i2c_write>   <!-- SPT0_CTRL2, SPT0 Slave mode 3.072MHz, 48kHz-->

<sleep ms="10"/>
```

```
<!-- SPT1 OUTPUT Setup-->

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 05 48 01 00 00 00</i2c_write>   <!-- SPT1_CTRL1, SPT1 TDM  -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 05 4C 00 00 00 00</i2c_write>   <!-- SPT1_CTRL2, SPT1 Slave mode 48k/6.144M-->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 05 50 10 00 00 00</i2c_write>   <!-- SPT1_ROUTE0 = ASRC0O_0-->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 05 54 11 00 00 00</i2c_write>   <!-- SPT1_ROUTE1 = ASRC0O_1-->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 05 58 12 00 00 00</i2c_write>   <!-- SPT1_ROUTE2 = ASRC0O_2-->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 05 5C 13 00 00 00</i2c_write>   <!-- SPT1_ROUTE3 = ASRC0O_3-->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 28 FF 00 00 00</i2c_write>    <!-- ALL ON -->

<sleep ms="10"/>

<!-- ASRCI Setup -->

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 18 71 00 00 00</i2c_write>   <!-- ASRC0_PWR,ASRCO_0/1/2_EN = 1 & ASRCI_0_EN = 1-->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 78 04 00 00 00</i2c_write>   <!-- ASRCI0_CTRL, 48KHzFS, SPT0 rate source-->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 7C 10 00 00 00</i2c_write>   <!-- ASRCI0_ROUTE01 = SPT_CH0,SPT_CH1 -->

<sleep ms="10"/>

<!-- FINT Setup -->

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 20 01 00 00 00</i2c_write>   <!-- FINT_PWR , FINT0_EN = 1 -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 48 54 00 00 00</i2c_write>   <!-- FINT_CTRL1, fs = 48KHz and fs_out = 384KHz -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 58 20 00 00 00</i2c_write>   <!-- FINT_ROUTE0 = ASRCI1_0 -->

<sleep ms="10"/>

<!-- DSP_PWR + FDSP ENABLE -->

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 34 01 01 00 00</i2c_write>   <!-- DSP_PWR,  FDSP ENABLE = 1 -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 FC 00 00 00 00</i2c_write>   <!-- HiFi Speed -->

<sleep ms="10"/>

<!-- FDSP  Control Setup-->

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 B8 00 00 00 00</i2c_write>   <!-- FDSP RUN=0 -->

<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 BC 01 00 00 00</i2c_write>   <!-- FDSP Speed -->

<sleep ms="10"/>
```

```
<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 CC 09 00 00 00</i2c_write>   <!-- FDSP_RATE_SOURCE = FINT 0 & 1 -->

<sleep ms="10"/>

<!-- FDSP PROGRAM & PARAMETER MEMORY: FINT0 to FDSP_OUT0 (PassThrough), ADC0/1/2 to FDSP_OUT1/2/3 (PassThrough) -->

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 04 00 00  00 00 00 12 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 04 00 04  00 04 00 12 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 04 00 08  00 08 00 12 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 04 00 0C  00 0C 00 12 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 04 00 10  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 04 00 14  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 00 00  00 00 00 70 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 00 04  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 00 08  00 00 00 01 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 00 0C  00 00 00 02 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 00 10  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 00 14  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 02 00  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 02 04  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 02 08  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 02 0C  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 02 10  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 02 14  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 04 00  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 04 04  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 04 08  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 04 0C  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 04 10  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 04 14  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 06 00  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 06 04  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 06 08  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 06 0C  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 06 10  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 06 14  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 08 00  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 08 04  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 08 08  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 08 0C  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 08 10  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 08 14  00 00 00 00 </i2c_write>
```

```
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0A 00  00 00 00 70 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0A 04  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0A 08  00 00 00 01 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0A 0C  00 00 00 02 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0A 10  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0A 14  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0C 00  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0C 04  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0C 08  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0C 0C  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0C 10  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0C 14  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0E 00  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0E 04  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0E 08  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0E 0C  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0E 10  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 0E 14  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 10 00  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 10 04  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 10 08  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 10 0C  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 10 10  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 10 14  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 12 00  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 12 04  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 12 08  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 12 0C  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 12 10  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 12 14  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 14 00  00 00 00 70 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 14 04  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 14 08  00 00 00 01 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 14 0C  00 00 00 02 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 14 10  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 14 14  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 16 00  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 16 04  00 00 00 00 </i2c_write>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 16 08  00 00 00 00 </i2c_write>
```

```
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 16 0C  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 16 10  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 16 14  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 18 00  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 18 04  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 18 08  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 18 0C  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 18 10  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 18 14  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 1A 00  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 1A 04  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 1A 08  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 1A 0C  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 1A 10  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 1A 14  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 1C 00  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 1C 04  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 1C 08  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 1C 0C  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 1C 10  00 00 00 00 </i2c_write>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B"> F0 02 1C 14  00 00 00 00 </i2c_write>
<!-- HPOUT Setup -->
<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 F4 47 00 00 00</i2c_write>   <!--DAC_CTRL1  set DAC FS 384kHz  -->
<sleep ms="10"/>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 08 20 00 00 00</i2c_write>   <!--DAC_ROUTE0  set DAC source to FDSP0   -->
<sleep ms="10"/>
<!-- FDSP RUN -->
<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 01 B8 01 00 00 00</i2c_write>   <!-- FDSP RUN=1 -->
<sleep ms="10"/>
<!-- UnMute ADC/DAC -->
<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 74 00 00 00 00</i2c_write>   <!-- adc's unmute -->
<sleep ms="10"/>
<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 F8 04 00 00 00</i2c_write>   <!--DAC_CTRL2  unmute DAC   -->
<sleep ms="10"/>
</aardvark>
```

Setup for Signal Path: DMIC→ PDMOUT

Connect the PDM Output to DMIC_01, and PDM Bit Clock_OUT to DMIC_CLK0 on the J13 connector.

Connect the SDATAO0 to PDM Input on APx and SDATAI1 to PDM Bit Clock_IN on APx.

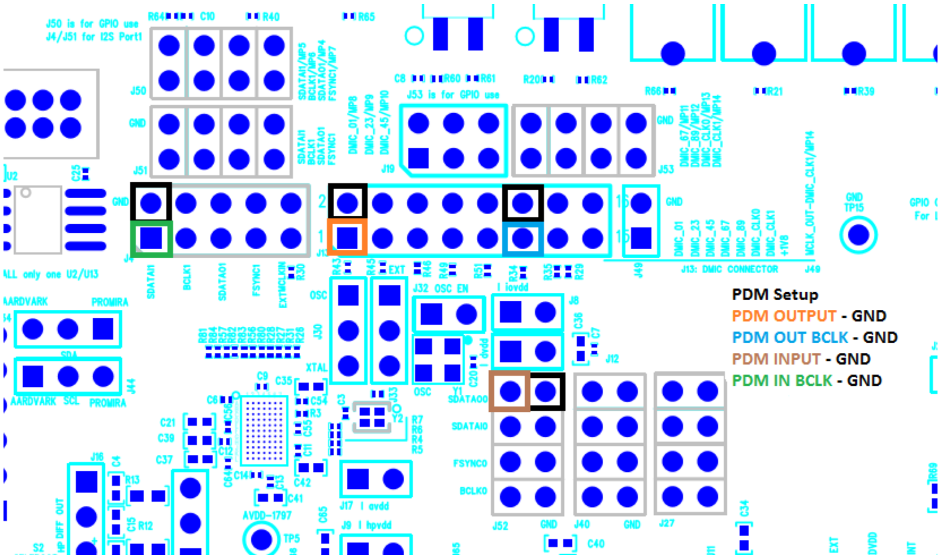


Figure 4. Pin Connections for DMIC to PDMOut

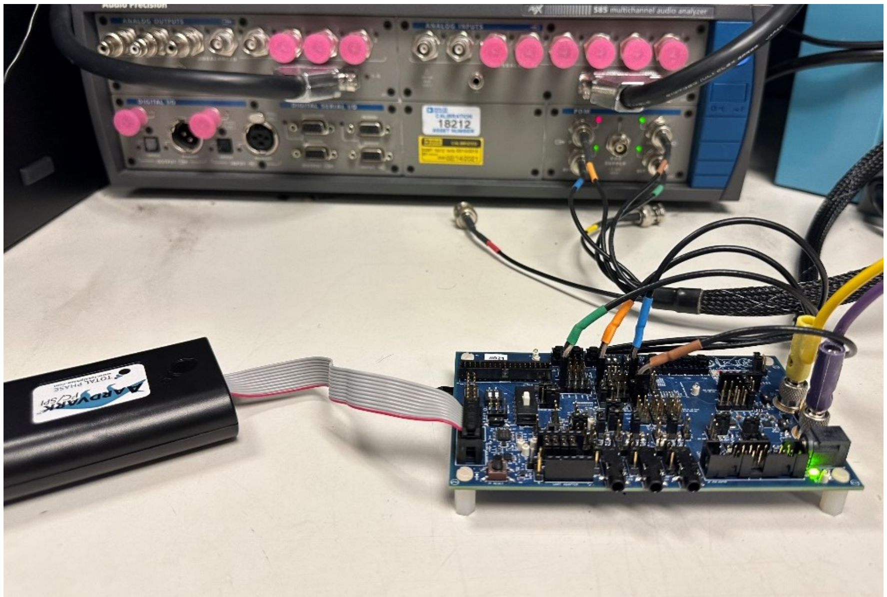


Figure 5. Physical Connections to the Evaluation Board and APx

Setup for Signal Path: ADC→FDSP→FDEC→ASRCO→SPT1

Now keep the connections for the above signal path as they are and add the following connections:

Connect 2 female XLR – to – 3.5mm cable to the AIN0 and AIN1 jacks. Connect the signal from AIN0 (J25) to AIN2 (J43) with a cable as shown in the figure if there are only two Analog Balanced Outputs on the APx.

Connect SPT1 signals to the Digital Serial Receiver of the APx. The three connections needed are BCLK1, SDATAO1, and FSYNC1 on the J4 header of the evaluation board.

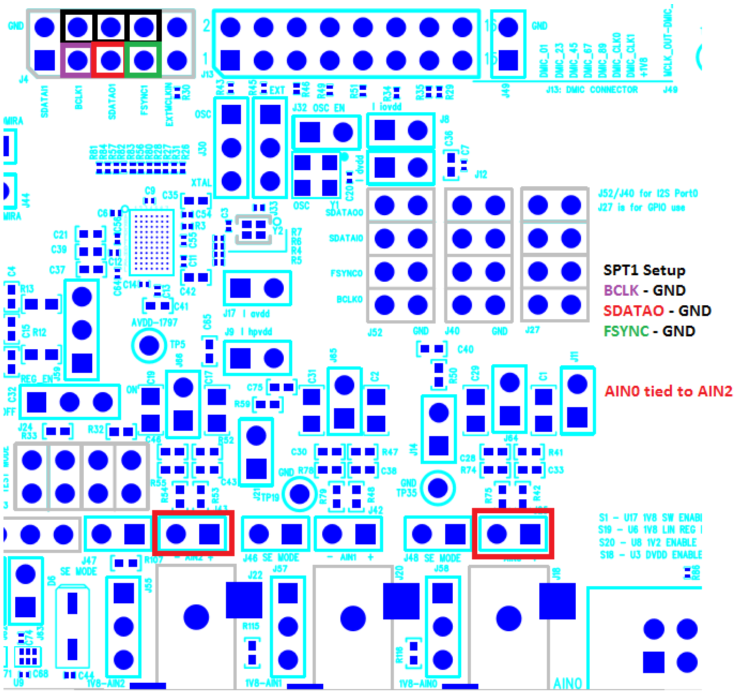


Figure 6. Pin Connections for ADC to SPT1

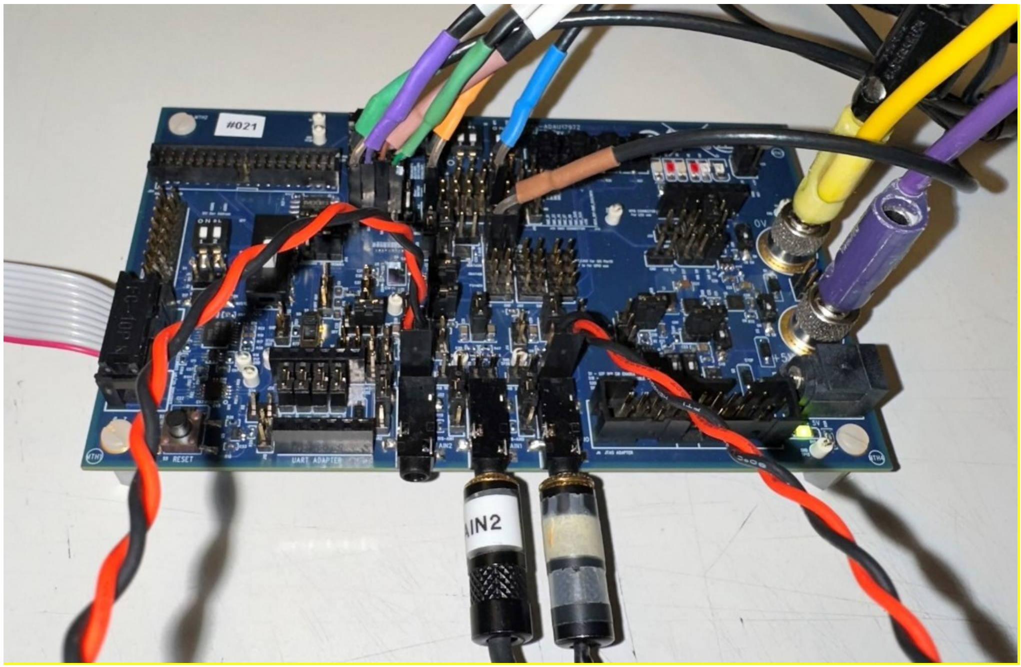


Figure 7. Evaluation Board with Cable Connections

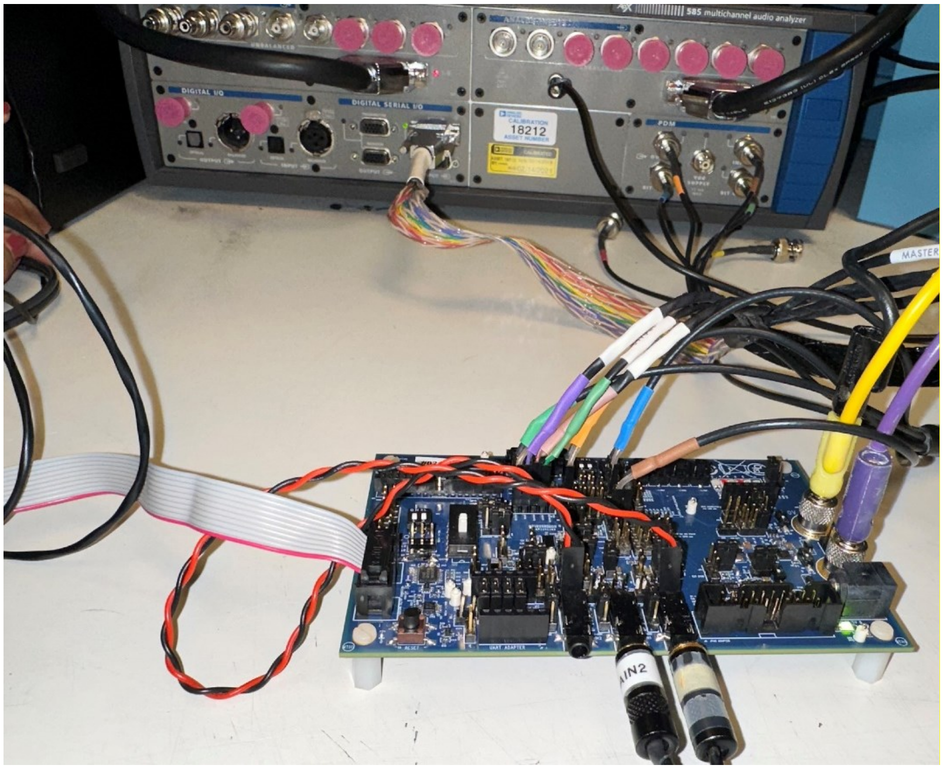


Figure 8. APx Cable Connections to the Evaluation Board

Setup for Signal Path: SPT0→ASRCI→FINT→FDSP→HPOUT

Again, keep the connections for the above signal paths intact as they are and add the following connections:

Connect the digital serial transmitter of APx to SPT0 signals on the evaluation board. The three connections needed are BCLK0, SDATAI0, and FSYNC0 on the J52 header of the evaluation board.

Connect a 16Ω + 33µH load to pins 1 and pins 3 of J16. Also, connect the 16Ω load to the AUX-0025 switching amplifier filter. Then connect the output of the switching amplifier filter to the analog balanced input of the audio precision.

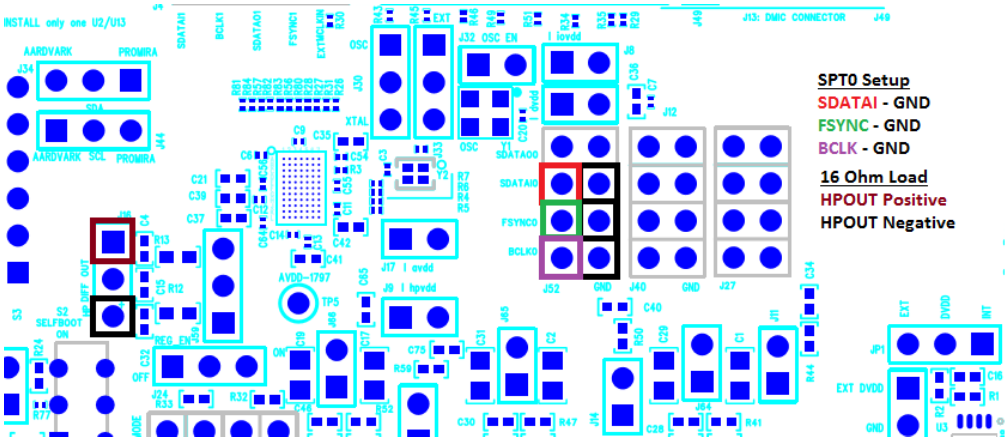


Figure 9. Pin Connections for SPT0 to HPOUT

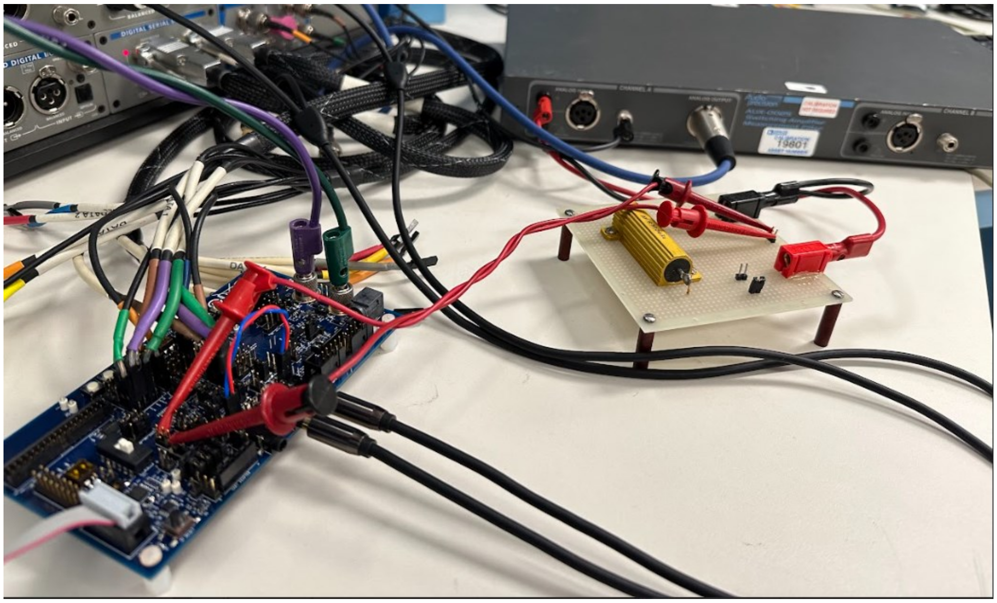


Figure 10. Evaluation Board Connected to the 16Ω Load which is Connected to the AUX0025 Filter

After completing the entire setup for all three signal paths, open the **Aardvark GUI** and configure the adapter. The example script can be copied into the **Batch Mode** window in the **Total Phase Control Center** software to execute the I2C instructions.

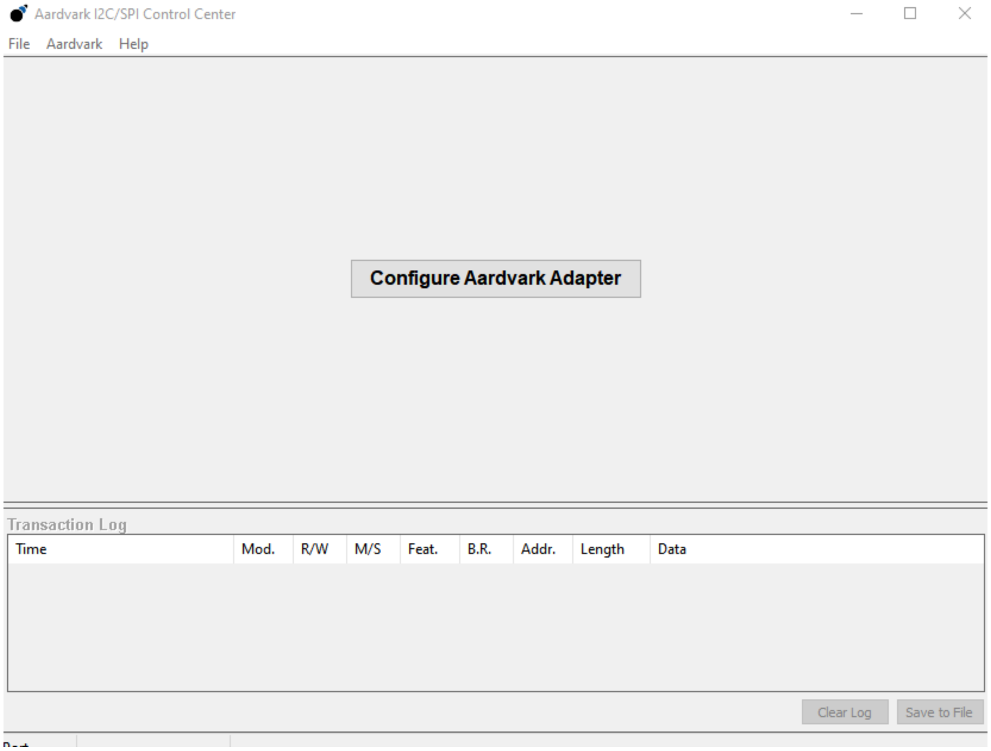
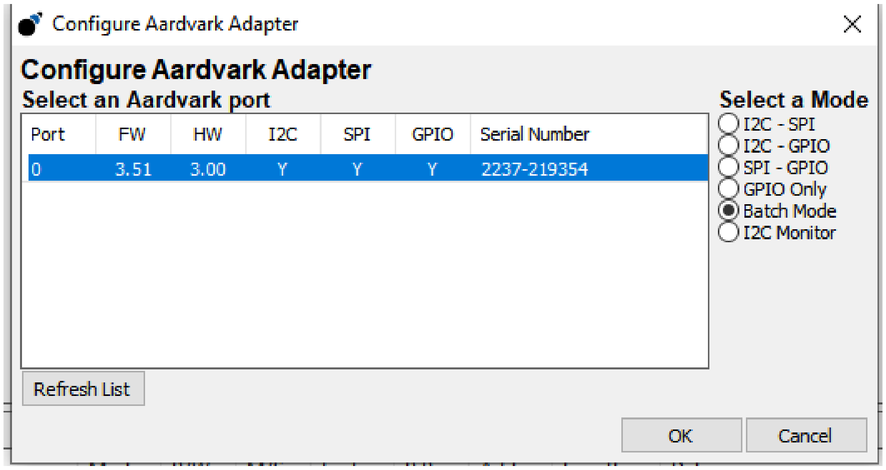


Figure 11. Aardvark Control Center GUI



Configure Aardvark Adapter

Configure Aardvark Adapter

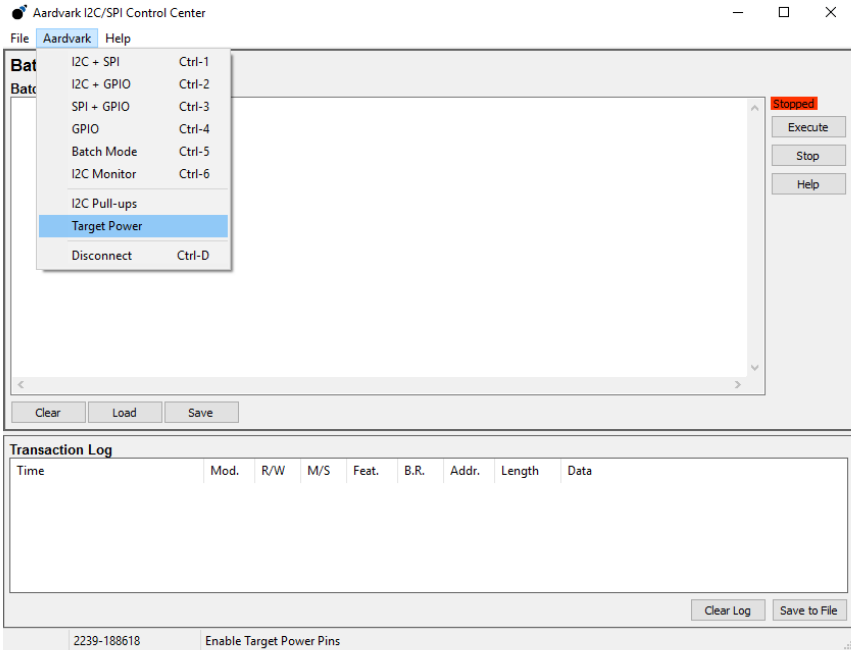
Select an Aardvark port

Port	FW	HW	I2C	SPI	GPIO	Serial Number
0	3.51	3.00	Y	Y	Y	2237-219354

Select a Mode

○ I2C - SPI
○ I2C - GPIO
○ SPI - GPIO
○ GPIO Only
● Batch Mode
○ I2C Monitor

Figure 12. Selecting the Adapter for Batch Mode Operation



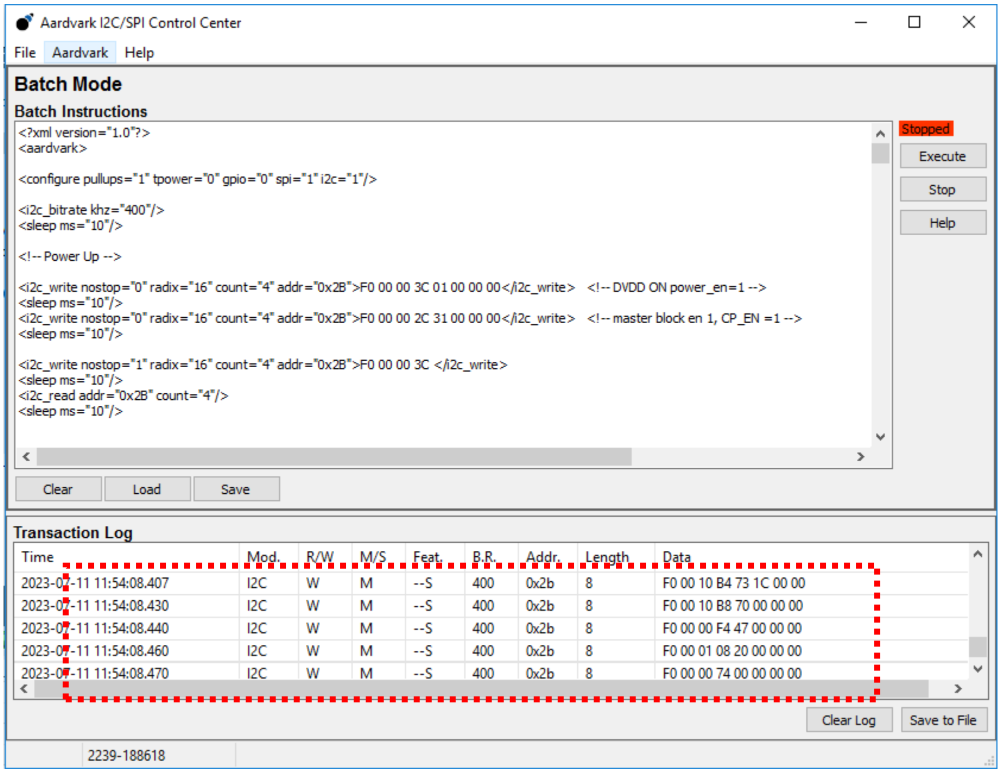
2239-188618 Enable Target Power Pins

Figure 13. Ensuring Target Power is OFF

Ensure that **Total Phase** has connected to the Aardvark in **Batch Mode** and that **Target Power** is OFF or unchecked under the **Aardvark** tab.

Once in **Batch Mode**, copy the script from this user guide and paste it into the window.

Once the script of I2C writes appears in the **Batch Mode** window, click **Execute**. The **Transaction Log** should display the I2C Reads/Writes, verifying the proper programming of the evaluation board.



```
<?xml version="1.0"?>
<aardvark>

<configure pullups="1" tpower="0" gpio="0" spi="1" i2c="1"/>

<i2c_bitrate khz="400"/>
<sleep ms="10"/>

<!-- Power Up -->

<i2c_write nostop="0" radix="16" count="4" addr="0x2B">F0 00 00 3C 01 00 00 00</i2c_write>   <!-- DVDD ON power_en=1 -->
<sleep ms="10"/>
<i2c_write nostop="0" radix="16" count="4" addr="0x2B">F0 00 00 2C 31 00 00 00</i2c_write>   <!-- master block en 1, CP_EN =1 -->
<sleep ms="10"/>

<i2c_write nostop="1" radix="16" count="4" addr="0x2B">F0 00 00 3C </i2c_write>
<sleep ms="10"/>
<i2c_read addr="0x2B" count="4"/>
<sleep ms="10"/>
```

Time	Mod.	R/W	M/S	Feat.	B.R.	Addr.	Length	Data
2023-07-11 11:54:08.407	I2C	W	M	--S	400	0x2b	8	F0 00 10 B4 73 1C 00 00
2023-07-11 11:54:08.430	I2C	W	M	--S	400	0x2b	8	F0 00 10 B8 70 00 00 00
2023-07-11 11:54:08.440	I2C	W	M	--S	400	0x2b	8	F0 00 00 F4 47 00 00 00
2023-07-11 11:54:08.460	I2C	W	M	--S	400	0x2b	8	F0 00 01 08 20 00 00 00
2023-07-11 11:54:08.470	I2C	W	M	--S	400	0x2b	8	F0 00 00 74 00 00 00 00

Figure 14. Verify Successful Load by Checking the Transaction Log

After this script has been successfully executed, the above signal paths can be verified. The Audio Precision can be used to create projects that provide the input and output signals. A USB-Streamer can also be used to provide I^{2}S/TDM input and receive I^{2}S/TDM output, so long as the voltage levels are shifted from 3.3V to 1.8V.

DETAILED DESCRIPTION OF HARDWARE

JUMPERS AND CONNECTORS

Table 1. Connector and Jack Descriptions

REFERENCE DESIGNATOR	TYPE	FUNCTIONAL NAME	DESCRIPTION
J1	2x5 Pin header 0.1” pitch	AARDVARK I^{2}C/SPI CONNECTOR	10-Pin header used to connect aardvark adapter.
J2	DC jack	5V DC INPUT	Barrel jack that provides external power to the board. J2 accepts a 3.8V DC to 6V DC input.
J3	2-Pin header 0.1” pitch	EXT DVDD	2-Pin header to connect external DVDD supply.
J4	2x5 Pin header 0.1” pitch	Serial Audio Port 1	10-Pin header to connect I^{2}S signals to Serial Port1.
J5	2x8 Pin header 0.1” pitch	I^{3}C CONNECTOR	16-Pin header used to connect I^{3}C adapter.
J6	2x10 Pin header 0.1” pitch	JTAG ADAPTER	20-Pin header used to connect JTAG adapter.
J7	6-Pin header 0.1” pitch	UART ADAPTER	6-Pin header used to connect 3.3V UART adapter.
J8	2-Pin header 0.1” pitch	IOVDD	Jumper connects power to the IOVDD supply of the ADAU1797 from the power supply section.

REFERENCE DESIGNATOR	TYPE	FUNCTIONAL NAME	DESCRIPTION
J9	2-Pin header 0.1" pitch	HPVDD	Jumper connects power to the HPVDD supply of the ADAU1797 from the power supply section.
J10	2-Pin header 0.1" pitch	RESET	Header used to generate the reset for ADAU1797.
J11, J14, J21	2-Pin header 0.1" pitch	ECM Micbias	Jumpers used to add a microphone bias to the analog microphone inputs AIN0, AIN1, and AIN2.
J12	2-Pin header 0.1" pitch	DVDD	Jumper connects power to the DVDD supply of the ADAU1797 from the power supply section.
J13	2x8 Pin header 0.1" pitch	DMIC CONNECTOR	Used to connect DMIC inputs to ADAU1797.
J15	2-Pin header 0.1" pitch	POWER DOWN	Jumper used to power down the ADAU1797 analog and digital circuits.
J16, J59	3-Pin header 0.1" pitch	OUTPUT	Jumper provides access to the left/right channel mono differential headphone output.
J17	2-Pin header 0.1" pitch	AVDD	Jumper connects power to the AVDD supply of the ADAU1797 from the power supply section.
J18, J20, J22	Stereo mini jack	AIN0, AIN1, AIN2	Analog Input channels.
J19	2x3 Pin header 0.1" pitch	GPIO	Male headers for connecting the ADAU1797 GPIO pins to switches.
J24	3-Pin header 0.1" pitch	INT DVDD REG_EN	Used to select between the external DVDD source or on-board regulator for DVDD.
J26	6-Way SIP socket	GPIO	Female header used for LEDs.
J27	2x4 Pin header 0.1" pitch	MPx pin jumpers	Jumper used to connect push-buttons on the board to the MPx pins on ADAU1797.
J28	2x2 Pin header 0.1" pitch	VOLTAGE SEL_LED	Jumper to select between 3.3V or 4.5V to LED.
J29	2-Pin header 0.1" pitch	EXTERNAL AVDD	Jumper used to connect external AVDD supply to the board.
J30	3-Pin header 0.1" pitch	EXTERNAL/CRYSTAL SELECT	Jumper used to select between routing the onboard crystal to the device or using an external clock signal to route to MCLK.
J31	Binding post	GND	Connect to GND or 0V of the power supply.
J32	2-Pin header 0.1" pitch	OSCILLATOR ENABLE	Jumper for enabling or disabling the on-board oscillator. Remove to enable the oscillator.
J33	3-Pin header 0.1" pitch	EXTERNAL/OSCILLATOR SELECT	Used to select between using the on-board oscillator or the external master clock to route to the ADAU1797.
J34	3-Pin header 0.1" pitch	AARDVARK OR PROMIRA I2C SELECT	Jumper to select between Aardvark or Promira for I^{2}C control.
J35	2-Pin header 0.1" pitch	EXTERNAL IOVDD	Jumper used to connect external IOVDD supply to the board.
J36	2-Pin header 0.1" pitch	+5V/GND	External header to connect 5V and GND to board.
J37	2x3 Pin header 0.1" pitch	IOVDD SELECT	Jumper used to select the IOVDD (1.8V/1.2V/EXT) to ADAU1797.
J38	Binding post	+5V	Binding Post is used to connect the external +5V supply to the board.
J39	2x3 Pin header 0.1" pitch	AVDD SELECT	Jumper used to select the AVDD 1.8V (EXT/Switching regulator/Linear regulator) to ADAU1797.
J40	2x5 Pin header 0.1" pitch	Serial Audio Port 0	10-Pin header to connect I^{2}S signal to Serial Port 0.
J41	2-Pin header 0.1" pitch	ADAU1797 XTAL OUT	ADAU1797 Crystal Oscillator Output
J44	3-Pin header 0.1" pitch	AARDVARK OR PROMIRA I2C SELECT	Jumper to select between Aardvark or Promira for I^{2}C control.

REFERENCE DESIGNATOR	TYPE	FUNCTIONAL NAME	DESCRIPTION
J45	2-Pin header 0.1” pitch	3.3V REG output	Jumper used to connect the 3.3V regulator output.
J46, J47, J48	2-Pin header 0.1” pitch	SE-MODE	Jumper to connect AIN1, AIN2, and AIN0 in Single-Ended Mode to ADAU1797.
J50	2x4 Pin header 0.1” pitch	GPIO	Male headers for connecting the ADAU1797 GPIO pins to switches.
J51	2x4 Pin header 0.1” pitch	Serial Audio Port 1	10-Pin header to connect I^{2}S signals to Serial Port1.
J52	2x4 Pin header 0.1” pitch	Serial Audio Port 0	10-Pin header to connect I2S signals to Serial Port0.
J53	2x4 Pin header 0.1” pitch	GPIO	Male header to connect the ADAU1797 GPIO for external use.
J54	2x17 Pin header 0.1” pitch	QSPI PORT- PROMIRA CONNECTOR	34-Pin header used to connect the Promira adapter.
J55, J57, J58	3-Pin header 0.1” pitch	DIFFERENTIAL MODE HEADERS	Headers when connected in position 1-2 provide AIN0, AIN1, and AIN2 in Differential Mode to ADAU1797, when connected in position 2-3 provide MICBias for MEMS Mic.
J56	6-Pin header 0.1” pitch	UART CONNECTOR	Used for 1.8V UART connection.
J60, J61, J62, J63	2-Pin headers 0.1” pitch	UART 3.3V I/O	Used to provide 3.3V level shifted signals to ADAU1797 UART Port.
J64, J65, J66	2-Pin header 0.1” pitch	SE/DIFF Input	Do not install for SE mode or install for DIFF mode.
J67	3-Pin header 0.1” pitch	SCL/SDA	Header used to monitor SCL, SDA signals to ADAU1797.

Table 2. Switch Descriptions

REFERENCE DESIGNATOR	TYPE	FUNCTIONAL NAME	DESCRIPTION
S1	SPST	SW REG ENABLE	Switch to enable the 1.8V switching regulator.
S2	SPDT	SELFBOOT	Switch to turn Selfboot mode ON/OFF.
S3	4PDT	I2C/SPI SELECT	Switch to select I^{2}C or SPI mode.
S4	2 Section SPST	I2C DEVICE ADDRESS	Switch to set I^{2}C device address.
S5	SPST-NO	RESET	Switch to provide a reset signal to ADAU1797.
S6	SPST	REG ENABLE	Switch to enable the 3.3V regulator.
S8, S12	2xSPST	GPIO	DIP switches for GPIO function.
S9, S10, S11, S13, S14, S15, S16	SPST-MOM	GPIO	Push switches for GPIO function.
S17	SPST	MASTER_SLAVE_SEL	Switch to select between SPI master or SPI slave mode for ADAU1797.
S18	SPST	DVDD REG ENABLE	Switch to enable 1.1V regulator.
S19	SPST	LIN REG ENABLE	Switch to enable 1.8V Linear Regulator.
S20	SPST	IOVDD REG ENABLE	Switch to enable 1.2V Linear Regulator.

DEFAULT SWITCH AND JUMPER SETTINGS

CLOCKING SETUP

The EVAL-ADAU1797Z provides the following multiple options for clocking the ADAU1797:

- Option # 1: Provide MCLK externally
  - In this option, the external master clock is connected to the "EXT MCLKIN" pin of J4. The top pins of J4 are GND.
  - J30 would be shunted toward "OSC." J33 would be shunted toward "EXT." J32 would not be important.
- Option # 2: Use the on-board 24.576MHz external oscillator
  - In this option, J30 would be shunted toward "OSC." J33 would be shunted toward the bottom. J32 would be left open.
- Option # 3: Use the ADAU1797 XTAL Oscillator
  - In this option, J30 would be shunted toward "XTAL." J33 and J32 would not be important.

The default jumper settings use Option #3 to use the ADAU1797 XTAL oscillator (see *Figure 15*).

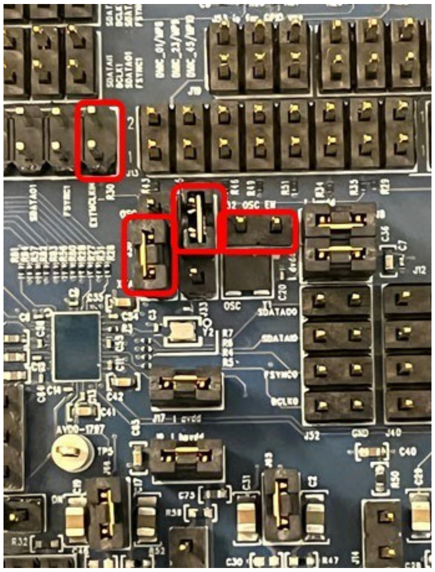


Figure 15. Clocking Setup, XTAL Mode

I2C SETUP

The ADAU1797 has two address pins, ADDR0 and ADDR1, that help to set the I2C device address. To set the ADDR0/ADDR1 pins to either IOVDD or GND, use S4 as shown in *Figure 16*. The boards as currently configured are set such that ADDR1 = ADDR0 = IOVDD, which results in a Device Address of 0x2B.

By default, jumpers J34 and J44 are set to connect the Aardvark I2C signals to the ADAU1797. This is also shown in *Figure 16*.

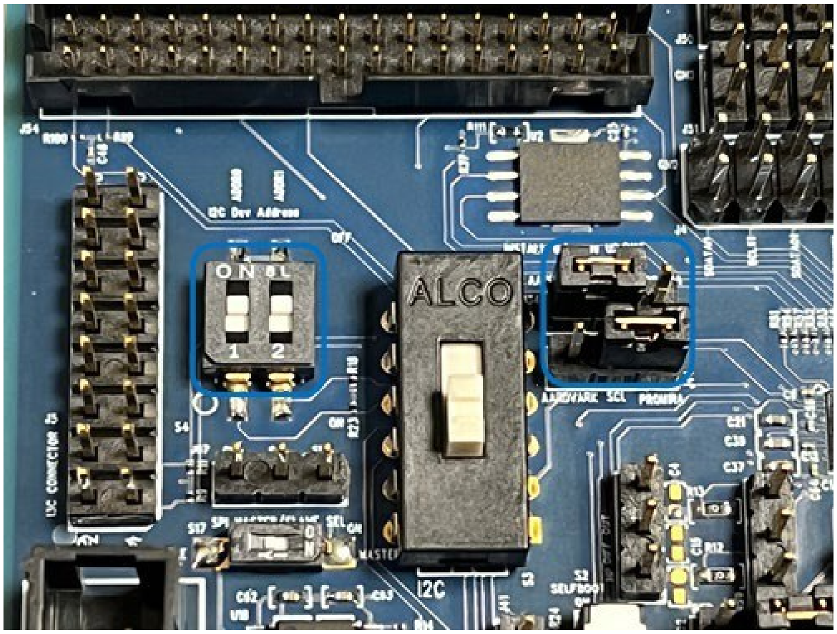


Figure 16. I^{2}C Configuration

POWER SETUP

The EVAL-ADAU1797Z must be powered with a +5V DC power supply. This can be provided through banana jacks J38 and J31, or by the wall-wart connection J2. Likewise, +5V power from the Aardvark I^{2}C/SPI or Promira I^{2}C/SPI adapter can be used instead. The +5V power from the JTAG or UART adapters can also be used. Diodes exist on the board to prevent issues if more than one power source is on at the same time.

Besides the +5V main board power supply, the EVAL-ADAU1797Z has onboard regulators to generate +1.8V, +1.2V, +1.1V, and +3.3V. Not all regulators must be enabled, but some must be enabled to ensure the ADAU1797 and some additional devices receive power.

Note that the +1.8V, +1.2V, and + 1.1V regulators are likely to be used for the ADAU1797 power supplies AVDD/HPVDD, IOVDD, and DVDD respectively. The EVAL-ADAU1797Z allows options for these supplies to be provided externally. Also, there are options to use the +1.8V linear regulator instead of the +1.8V switching regulator and to use the +1.8V regulator for IOVDD instead of the +1.2V regulator.

As the boards are currently configured, the +3.3V regulator (U16) is enabled. This means that S6 is switched to ON and that J45 is shunted. The +1.1V regulator (U3) is also enabled, meaning that S18 is switched to ON, and JP1 is set to "DVDD ◊ INT." The +1.2V regulator (U8) is also enabled, meaning that S20 is switched to ON, and J37 is set to "+1.2V" Lastly, the +1.8V switching regulator (U17) is enabled. This means that S1 is switched to ON, and J39 is set to "1.8V SW." These connections are shown in *Figure 17*.

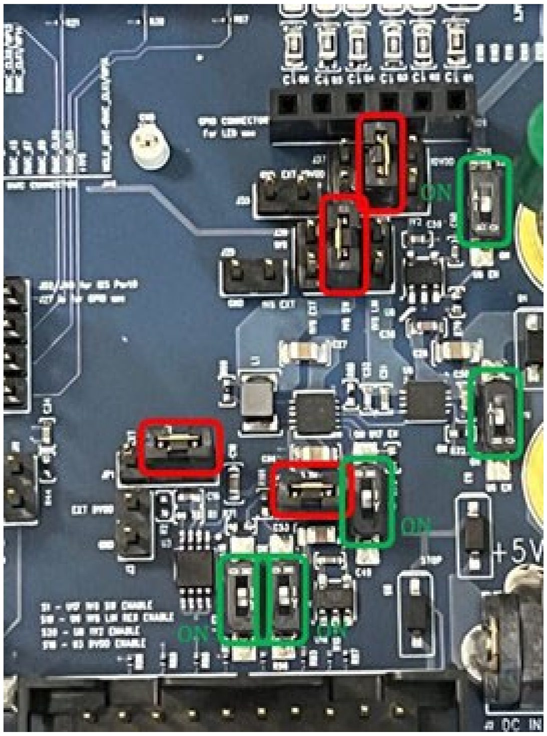


Figure 17. Power and Regulators

ADAU1797 POWER AND CURRENT MEASUREMENTS

The EVAL-ADAU1797Z has current measurement headers so that the AVDD, HPVDD, DVDD, or IOVDD current can be measured independently. In order to ensure the power supplies reach the ADAU1797, these jumpers should normally be shunted. J12 may be left open if REG_EN = ON. These headers and their locations are shown in *Figure 18*.

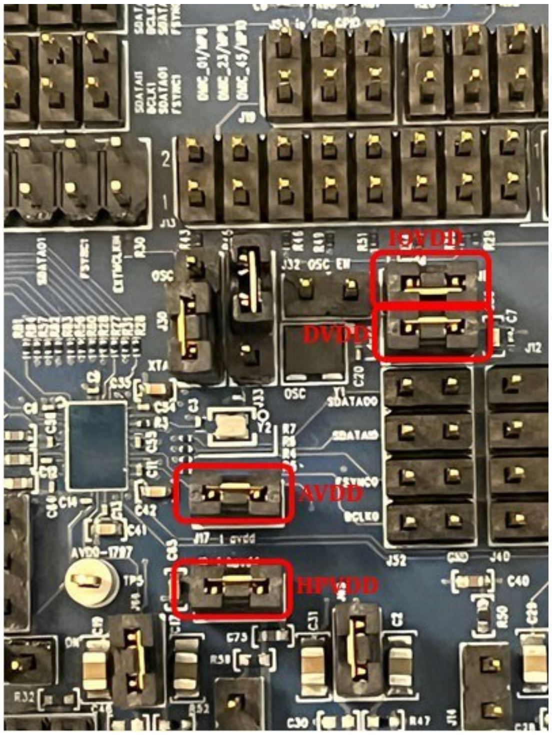


Figure 18. Current Measurement Headers

ANALOG INPUTS

The ADAU1797 provides three analog inputs. The analog inputs can be configured as Differential, Pseudo-differential, or Single Ended. The J18, J20, and J22 (3.5mm Stereo Jack) are provided to connect the external analog inputs to the board. For the ECM microphone, the 1.8V Micbias can be provided by installing jumpers for the headers J11/J14/J21. If using the Mems microphones, then install a jumper in position 2-3 for J55/J57/J58 to provide the 1.8V Micbias.

Table 3. Analog Inputs

HEADER	DIFFERENTIAL	PSEUDO-DIFFERENTIAL	SINGLE-ENDED
J64, J65, J66	Install	Install	OPEN
J46, J47, J48	OPEN	Install	Install
J55, J57, J58	Install 1-2 position	OPEN	OPEN

DIGITAL MICROPHONE INPUTS

The ADAU1797 provides ten digital microphone inputs. The inputs are provided in pairs e.g., DMIC_01, DMIC_23, DMIC_45, DMIC_67, and DIMC_89. The two Clock outputs are provided for connecting to the external digital microphones. The header J13 can be used to connect the external digital microphones. The digital microphones must be 1.8V logic. The J13 also provides 1.8V output to power the external digital microphones.

HPOUT HEADERS

The ADAU1797 has integrated Class-D headphone output. There are two headers available to find the Class-D headphone output: J59 and J16. J59 is placed before any filtering components. J16 is placed after the filtering component's placeholders. Note that currently, R13 and R12 are 0Ω resistors. C15, C4, and C32 are left open. But if desired, these placeholders can be replaced with EMI filtering components for example.

On both J59 and J16, the middle pin is GND. The outer pins are HPOUT+ and HPOUT-. This is shown in Figure 19.

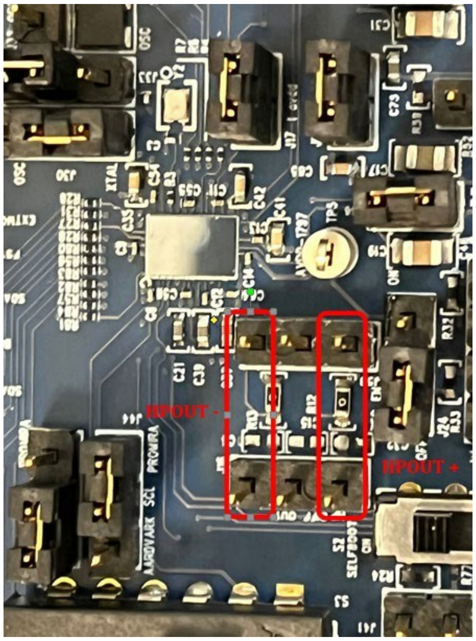


Figure 19. Class-D Headphone Output

SERIAL PORT HEADERS

The ADAU1797 has two serial ports, which are both accessible on the EVAL-ADAU1797Z. The header J4 has the connections for Serial Port 1. The header J51 has a second, redundant set of connections for Serial Port 1. The header J52 has the connections for Serial Port 0. The header J40 has a second, redundant set of connections for Serial Port 0.

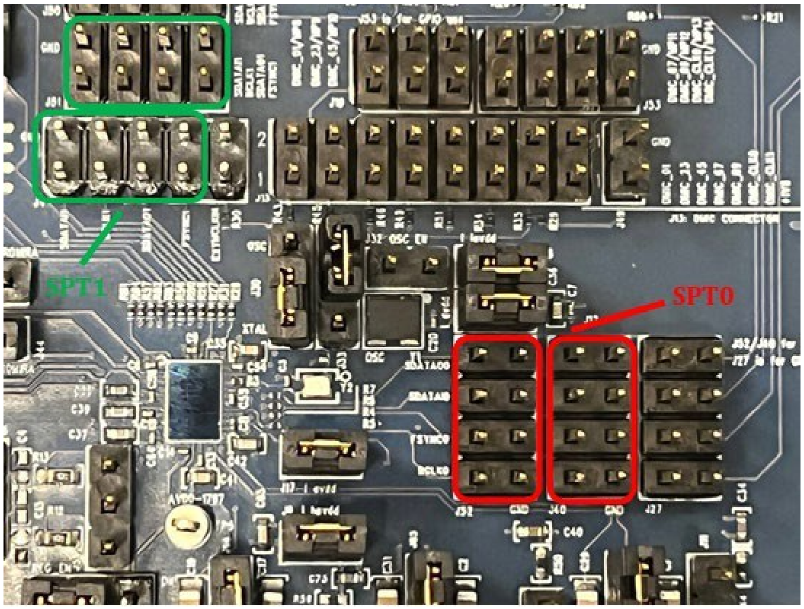


Figure 20. Serial Port Headers

GPIO

The ADAU1797 provides a total of 31 multipurpose pins that can be configured for GPIO use. 15 MPx pins are brought out to the headers J27, J50, J19, and J53. Some of these can be connected to onboard switches as well as for LED indication purposes. These MPx pins are compatible with 1.8V logic.

UART

The UART port connections for ADAU1797 are available at the header J56 which supports the 1.8V logic. In addition, the header J7 is provided to support the 3.3V logic signals. The UART provides RTS, CTS, TXD, and RXD signal pins. The UART port is tested with a PMOD adapter as shown in *Figure 21*.

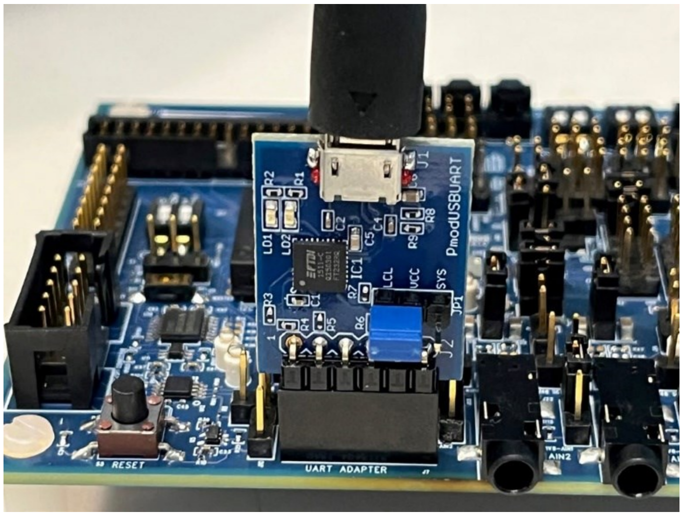
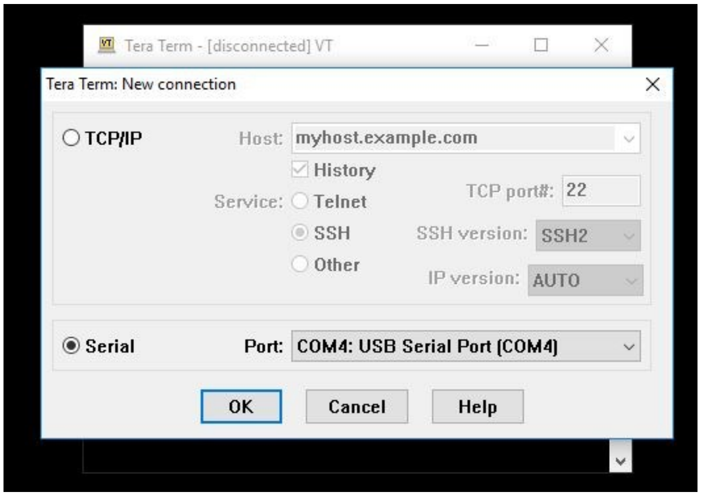


Figure 21. PMOD USB/UART Connection

TeraTerm software can be used to program the device using UART. The steps to set up TeraTerm are as follows:

Open TeraTerm and set up a serial connection through the USB serial port. Depending on the PC setup, the COM # could be different on the setup. Be sure to select the COM # that specifies the USB serial port as seen in *Figure 22*.


Tera Term - [disconnected] VT

Tera Term: New connection

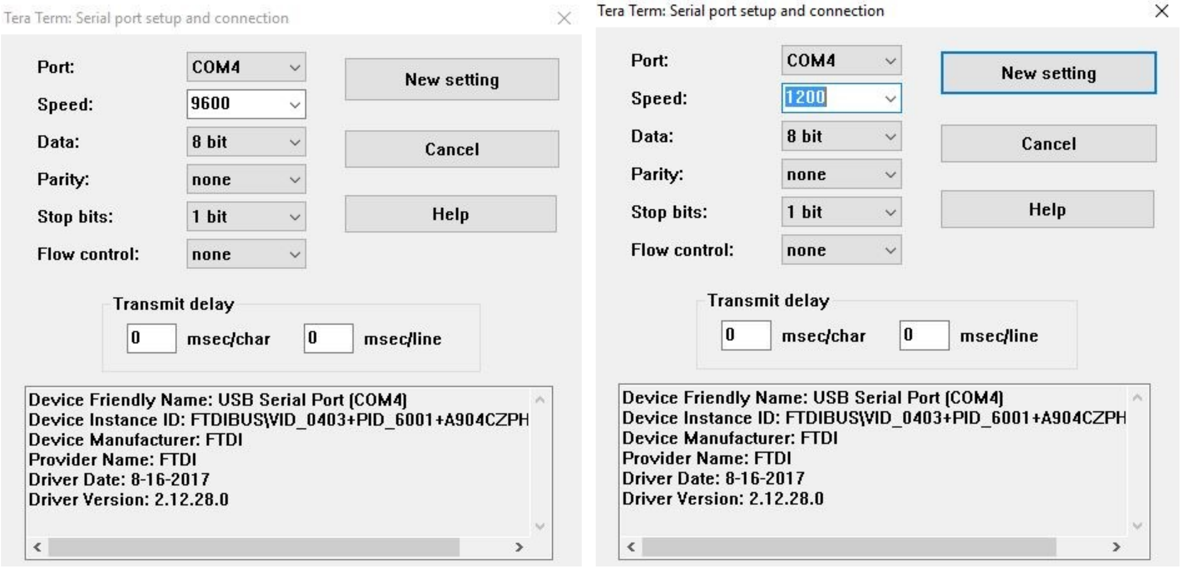
○ TCP/IP Host: myhost.example.com
☑ History
Service: ○ Telnet TCP port#: 22
● SSH SSH version: SSH2
○ Other IP version: AUTO

● Serial Port: COM4: USB Serial Port (COM4)

Figure 22. TeraTerm Setup

Click **OK** and a blank TeraTerm window appears. From here, Click **Setup → Serial port…**

The Connection Speed can be changed from the default of 9600 if needed, according to the script, and click **New setting**.


Tera Term: Serial port setup and connection
New setting
Cancel
Help
Transmit delay
Tera Term: Serial port setup and connection
New setting
Cancel
Help
Transmit delay

Port: COM4
Speed: 9600
Data: 8 bit
Parity: none
Stop bits: 1 bit
Flow control: none
0 msec/char 0 msec/line

```
Device Friendly Name: USB Serial Port (COM4)
Device Instance ID: FTDIBUS\VID_0403+PID_6001+A904CZPH
Device Manufacturer: FTDI
Provider Name: FTDI
Driver Date: 8-16-2017
Driver Version: 2.12.28.0
```

Port: COM4
Speed: 1200
Data: 8 bit
Parity: none
Stop bits: 1 bit
Flow control: none
0 msec/char 0 msec/line

```
Device Friendly Name: USB Serial Port (COM4)
Device Instance ID: FTDIBUS\VID_0403+PID_6001+A904CZPH
Device Manufacturer: FTDI
Provider Name: FTDI
Driver Date: 8-16-2017
Driver Version: 2.12.28.0
```

Figure 23. Setting Rate from 9600 to 1200

A blank terminal is now idling.

Next, open the **Aardvark GUI** and connect the Aardvark adapter for Batch Mode. Load the script which contains the code either for transmission or reception.

Transmitting data using UART:

Execute the script in the **Aardvark Control Center** and check that the **TeraTerm** window displays the expected output as per the script.

To clear the transmitted message on the **TeraTerm** window, go to **TeraTerm** and click **Edit → Clear Screen** to remove the message from the Terminal.

Receiving data using UART:

Here, the user can verify that the USB serial port can transmit data through the UART.

In the blank **TeraTerm** window, type any single alphabetical letter/message, and then click **Enter**. Switch back to the **Aardvark Control Center** and **Execute** the script for receiving data.

If the script has a read command, the user should be able to see the HEX equivalent of the received data in the **Transaction Log** of the **Aardvark Control Center**.

QSPI

The QSPI port connections are available for ADAU1797 at header J54. The Aardvark adapter can be used to program the device along with a Level Shifter which converts the output voltage from the Aardvark adapter to 1.8V which is desirable for ADAU1797 EVB.

To test the QSPI, connect the Aardvark adapter to the PC using the USB cable and connect the other end of the Aardvark adapter to the Level Shifter Board from Total Phase on the "ADAPTER" Port as shown in *Figure 24*. On the other end of the Level Shifter Board, connect the QSPI cable to the "TARGET1" port. Now connect the QSPI cable to J54 of the evaluation board (QSPI Promira Adapter) port. Turn the SELBOOT pin OFF by pushing S2 high (to the **OFF** silkscreen label).

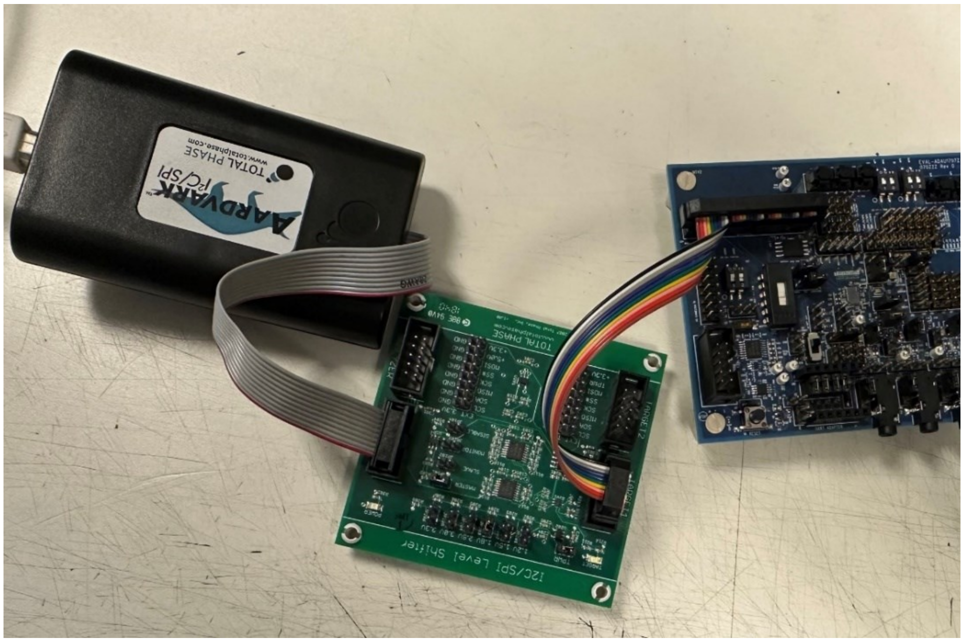


Figure 24. QSPI Connection

Now turn the +5V power supply to the evaluation board **ON** and open the **Aardvark GUI**. Click on **Configure Aardvark Adapter**.

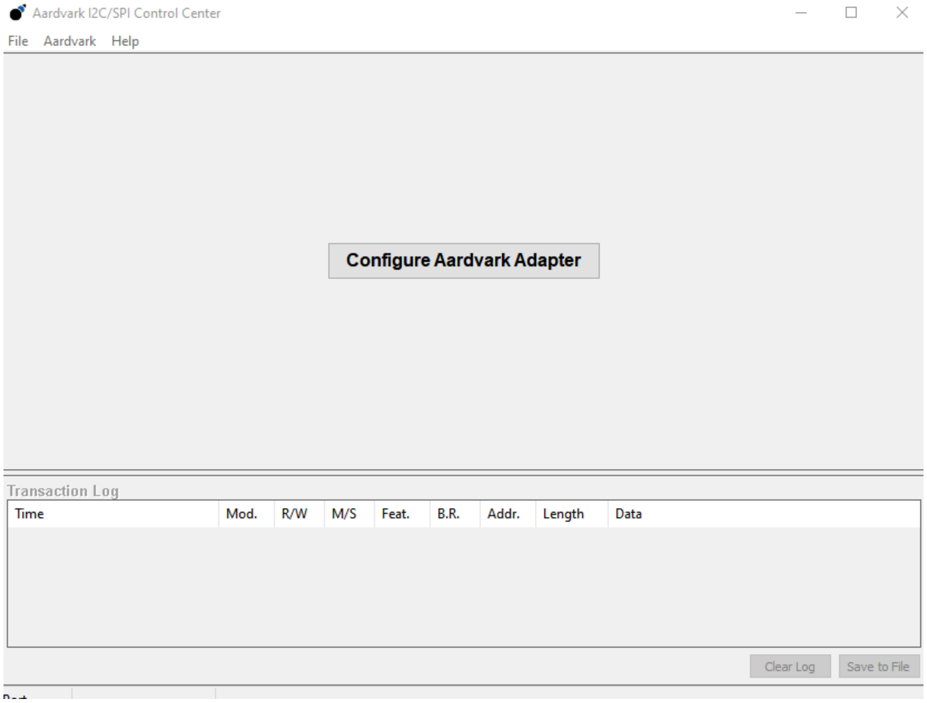
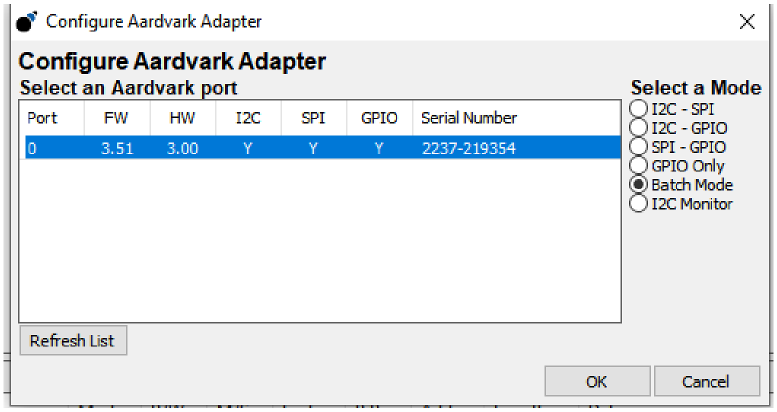

Aardvark I2C/SPI Control Center
File Aardvark Help
Configure Aardvark Adapter
Transaction Log
Time Mod. R/W M/S Feat. B.R. Addr. Length Data
Clear Log
Save to File

Figure 25. Aardvark Control Center

Select the adapter as seen in *Figure 26* and click **OK**.



Configure Aardvark Adapter

Select an Aardvark port

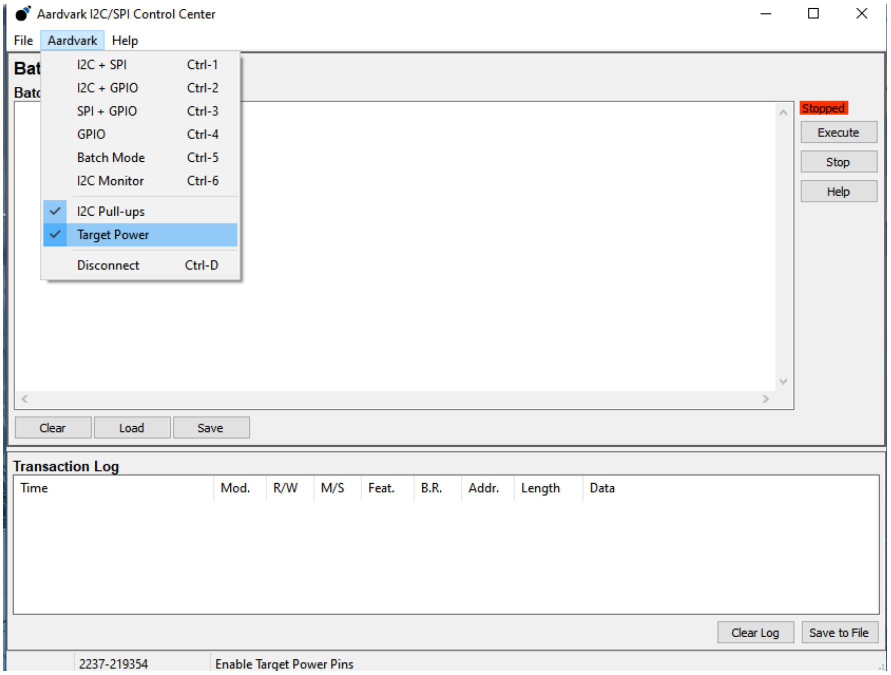
Port	FW	HW	I2C	SPI	GPIO	Serial Number
0	3.51	3.00	Y	Y	Y	2237-219354

Select a Mode

○ I2C - SPI
○ I2C - GPIO
○ SPI - GPIO
○ GPIO Only
● Batch Mode
○ I2C Monitor

Figure 26. Selecting Adapter for Batch Mode

Now click on the **Adapter** tab and click **Target Power** in the drop-down menu. If an option for 5.5V is visible select that, otherwise, turn ON **Target Power**. Target Power should have a checkmark on the left. After this step, the user should see a green LED glowing on the I2C/SPI Level Shifter board.



✓ I2C Pull-ups
✓ Target Power

Transaction Log

Time	Mod.	R/W	M/S	Feat.	B.R.	Addr.	Length	Data

2237-219354 Enable Target Power Pins

Figure 27. Target Power ON

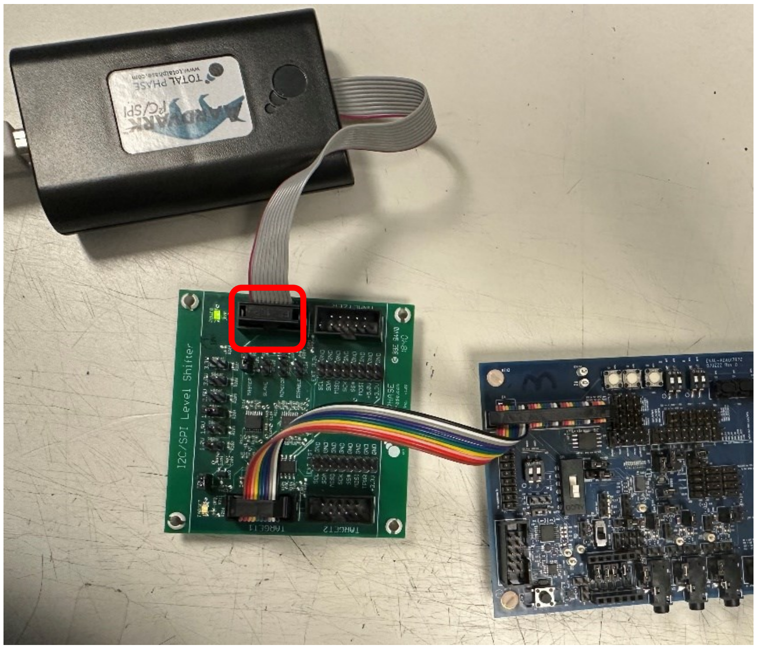
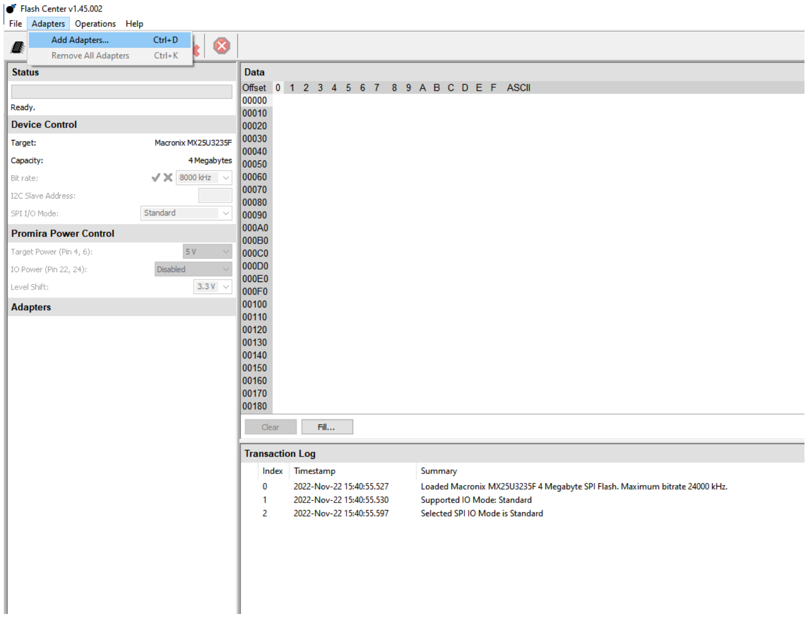


Figure 28. Verify Level Shifter LED

After this step, program the Flash using the Flash Center software. Make sure to close the Aardvark Control Center software before opening the Flash Center software, otherwise, the Aardvark adapter is unavailable to use.

Open the **Total Phase Flash Center** software. From the tabs at the top, click on **Adapters → Add Adapters…** and the software should show the connected Aardvark adapter. Click on it and click **Add**.



Status

Ready.

Device Control

Target: Macronix MX25U3235F

Capacity: 4 Megabytes

Bit rate: 8000 kHz

I2C Slave Address:

SPI I/O Mode: Standard

Promira Power Control

Target Power (Pin 4, 6): 5 V

IO Power (Pin 22, 24): Disabled

Level Shift: 3.3 V

Adapters

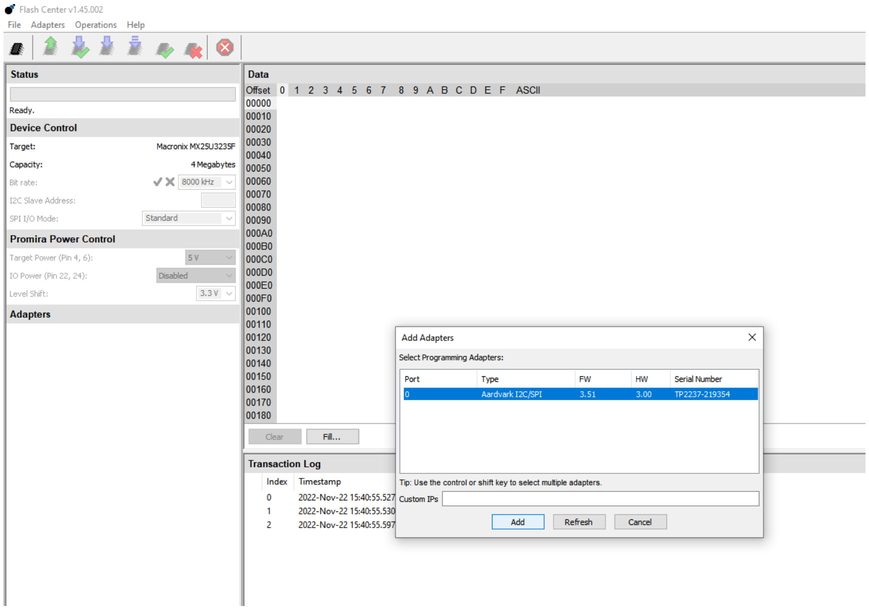
Data

```
Offset 0 1 2 3 4 5 6 7 8 9 A B C D E F ASCII
00000
00010
00020
00030
00040
00050
00060
00070
00080
00090
000A0
000B0
000C0
000D0
000E0
000F0
00100
00110
00120
00130
00140
00150
00160
00170
00180
```

Transaction Log

```
Index  Timestamp                  Summary
0      2022-Nov-22 15:40:55.527   Loaded Macronix MX25U3235F 4 Megabyte SPI Flash. Maximum bitrate 24000 kHz.
1      2022-Nov-22 15:40:55.530   Supported IO Mode: Standard
2      2022-Nov-22 15:40:55.597   Selected SPI IO Mode is Standard
```

Figure 29. Total Phase Control Center



Ready.

Device Control

Target: Macronix MX25U3235F
Capacity: 4 Megabytes
Bit rate: 8000 kHz
I2C Slave Address:
SPI I/O Mode: Standard

Promira Power Control

Target Power (Pin 4, 6): 5 V
IO Power (Pin 22, 24): Disabled
Level Shift: 3.3 V

Adapters

```
Offset 0 1 2 3 4 5 6 7 8 9 A B C D E F ASCII
00000
00010
00020
00030
00040
00050
00060
00070
00080
00090
000A0
000B0
000C0
000D0
000E0
000F0
00100
00110
00120
00130
00140
00150
00160
00170
00180
```

Add Adapters

Select Programming Adapters:

Port	Type	FW	HW	Serial Number
0	Aardvark I2C/SPI	3.51	3.00	TP2237-219354

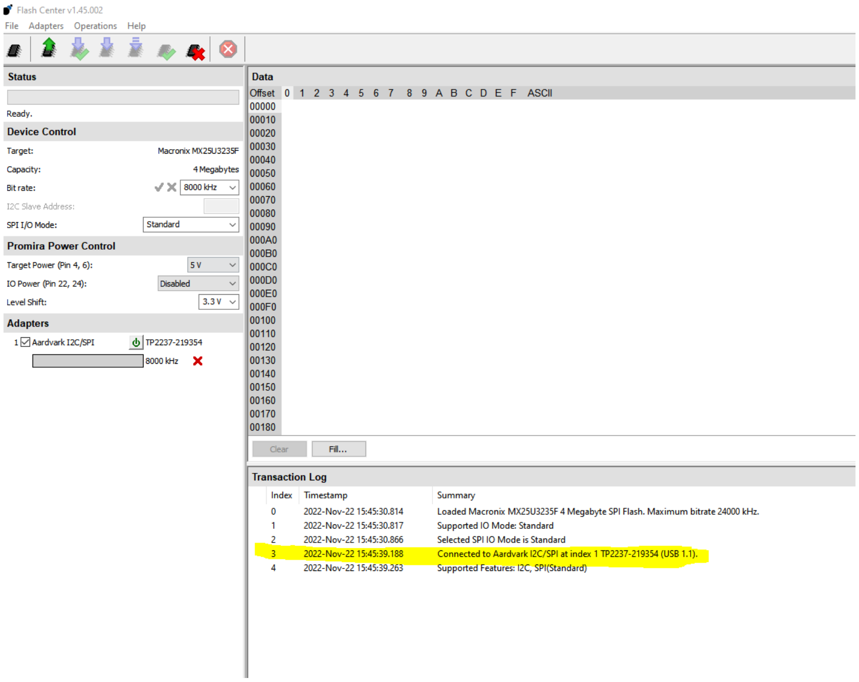
Tip: Use the control or shift key to select multiple adapters.

Custom IPs

```
Index Timestamp
0     2022-Nov-22 15:40:55.527
1     2022-Nov-22 15:40:55.530
2     2022-Nov-22 15:40:55.597
```

Figure 30. Select Aardvark Adapter

Once the user clicks on **Add**, the console shows a message saying connected to the Aardvark adapter.



Ready.

Device Control

Target: Macronix MX25U3235F
Capacity: 4 Megabytes
Bit rate: 8000 kHz
I2C Slave Address:
SPI I/O Mode: Standard

Promira Power Control

Target Power (Pin 4, 6): 5 V
IO Power (Pin 22, 24): Disabled
Level Shift: 3.3 V

Adapters

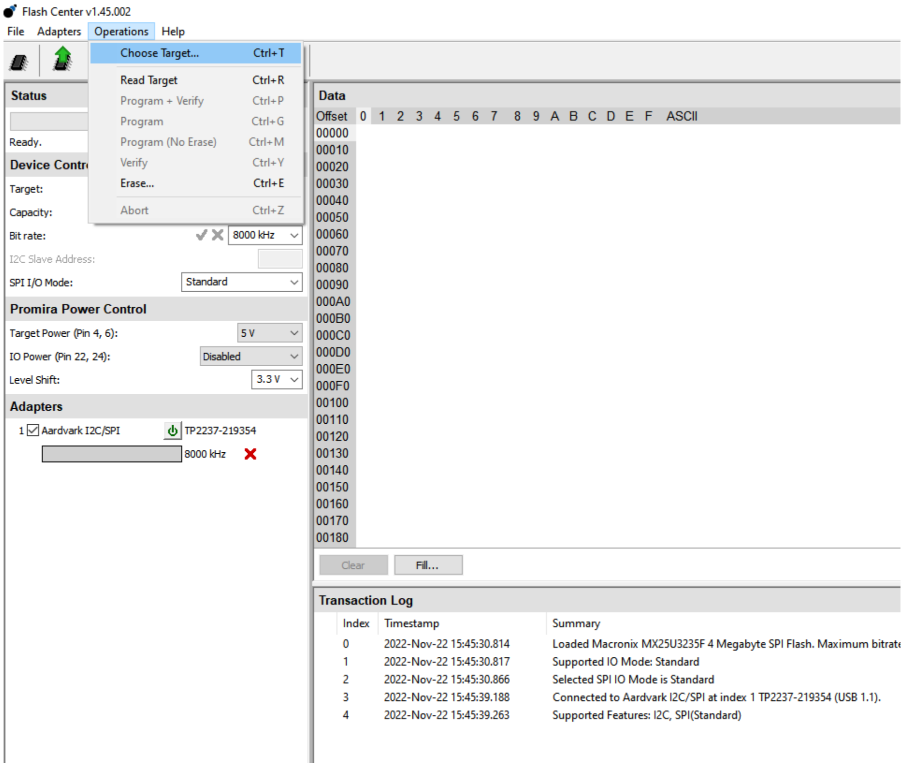
1 ☑ Aardvark I2C/SPI TP2237-219354
8000 kHz

```
Offset 0 1 2 3 4 5 6 7 8 9 A B C D E F ASCII
00000
00010
00020
00030
00040
00050
00060
00070
00080
00090
000A0
000B0
000C0
000D0
000E0
000F0
00100
00110
00120
00130
00140
00150
00160
00170
00180
```

```
Index Timestamp                Summary
0     2022-Nov-22 15:45:30.814 Loaded Macronix MX25U3235F 4 Megabyte SPI Flash. Maximum bitrate 24000 kHz.
1     2022-Nov-22 15:45:30.817 Supported IO Mode: Standard
2     2022-Nov-22 15:45:30.866 Selected SPI IO Mode is Standard
3     2022-Nov-22 15:45:39.188 Connected to Aardvark I2C/SPI at index 1 TP2237-219354 (USB 1.1).
4     2022-Nov-22 15:45:39.263 Supported Features: I2C, SPI(Standard)
```

Figure 31. After Connecting Adapter

Next, go to **Operations** in the menu bar and select **Choose Target**.



Bit rate: 8000 kHz

I2C Slave Address:

SPI I/O Mode: Standard

Target Power (Pin 4, 6): 5 V

IO Power (Pin 22, 24): Disabled

Level Shift: 3.3 V

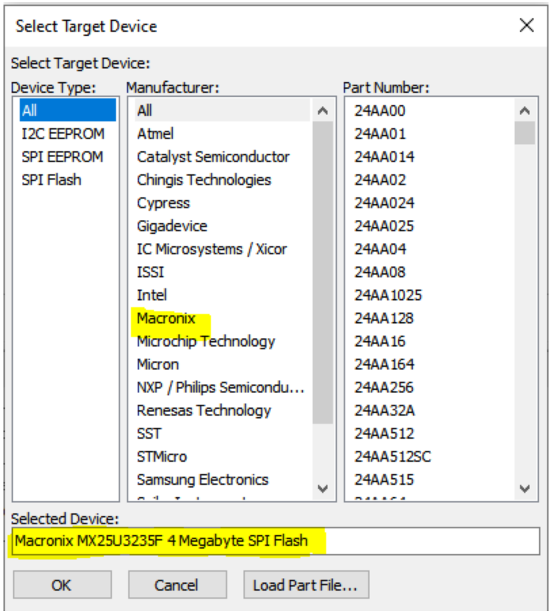
1 ☑ Aardvark I2C/SPI TP2237-219354

8000 kHz

Index	Timestamp	Summary
0	2022-Nov-22 15:45:30.814	Loaded Macronix MX25U3235F 4 Megabyte SPI Flash. Maximum bitrate
1	2022-Nov-22 15:45:30.817	Supported IO Mode: Standard
2	2022-Nov-22 15:45:30.866	Selected SPI IO Mode is Standard
3	2022-Nov-22 15:45:39.188	Connected to Aardvark I2C/SPI at index 1 TP2237-219354 (USB 1.1).
4	2022-Nov-22 15:45:39.263	Supported Features: I2C, SPI(Standard)

Figure 32. Choose Target

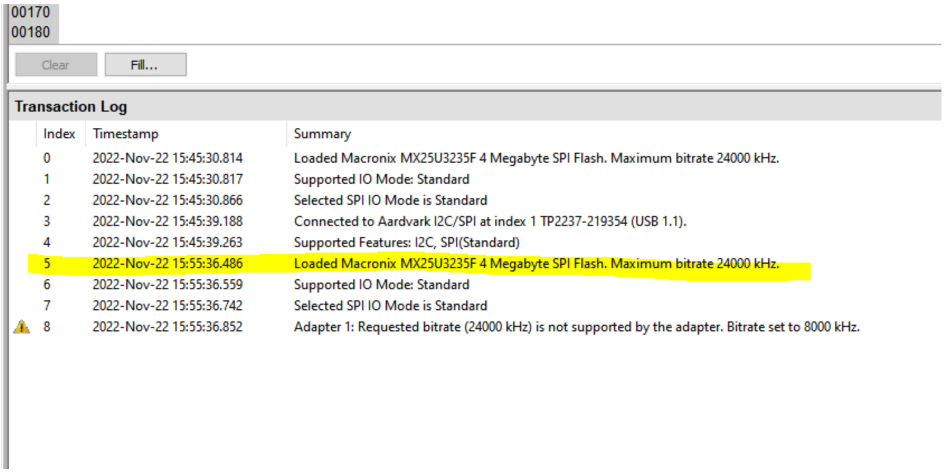
Choose the target as per the name shown in *Figure 33*. This is the name for the QSPI Flash device on ADAU1797. Note, this can be typed in under the **Selected Device** menu bar. Once the device is found, the user does not need to re-find it when opening this window for subsequent tests. Simply verify the text is already in the **Selected Device** section and then click **OK**.



Selected Device: Macronix MX25U3235F 4 Megabyte SPI Flash

Figure 33. Select Macronix MX25U3235F 4MB SPI Flash

After clicking **OK**, the console should show that the QSPI was loaded.



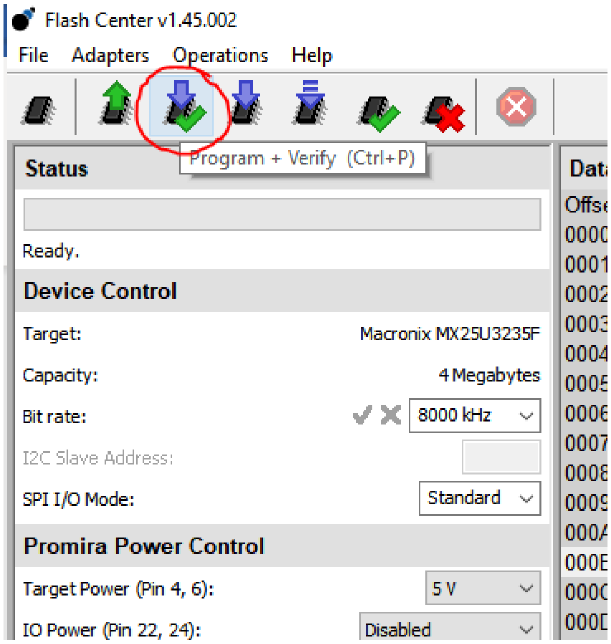
```
Index  Timestamp                 Summary
0      2022-Nov-22 15:45:30.814  Loaded Macronix MX25U3235F 4 Megabyte SPI Flash. Maximum bitrate 24000 kHz.
1      2022-Nov-22 15:45:30.817  Supported IO Mode: Standard
2      2022-Nov-22 15:45:30.866  Selected SPI IO Mode is Standard
3      2022-Nov-22 15:45:39.188  Connected to Aardvark I2C/SPI at index 1 TP2237-219354 (USB 1.1).
4      2022-Nov-22 15:45:39.263  Supported Features: I2C, SPI(Standard)
5      2022-Nov-22 15:55:36.486  Loaded Macronix MX25U3235F 4 Megabyte SPI Flash. Maximum bitrate 24000 kHz.
6      2022-Nov-22 15:55:36.559  Supported IO Mode: Standard
7      2022-Nov-22 15:55:36.742  Selected SPI IO Mode is Standard
⚠ 8    2022-Nov-22 15:55:36.852  Adapter 1: Requested bitrate (24000 kHz) is not supported by the adapter. Bitrate set to 8000 kHz.
```

Figure 34. Confirm Target Loaded

Now go to **File → Load File…**, change the file type to **Binary files,** and load the .bin file. Click **Open**.

The Data should display on the right side of the screen.

Now click on **Program + Verify**.



Ready.

Target: Macronix MX25U3235F

Capacity: 4 Megabytes

Bit rate: 8000 kHz

I2C Slave Address:

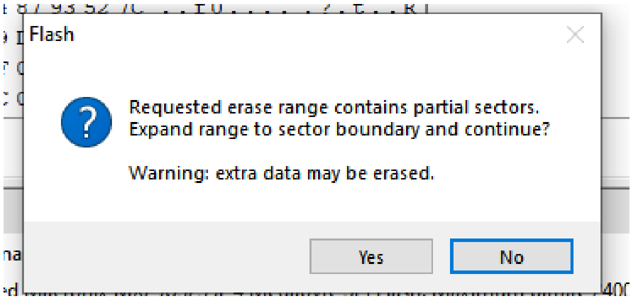
SPI I/O Mode: Standard

Target Power (Pin 4, 6): 5 V

IO Power (Pin 22, 24): Disabled

Figure 35. Program + Verify

The user might see a pop-up from **Flash** as follows, say **yes**:



Requested erase range contains partial sectors.
Expand range to sector boundary and continue?

Warning: extra data may be erased.

Figure 36. Erase Data Prompt

It should take a few seconds to load the data in the Flash and the progress is shown in a progress bar on the left. After successful programming and verification, the user sees the status **Verify xxx bytes Succeeded**.

Now the Flash is programmed. Next, turn OFF the power supply to the evaluation board. Switch the SELFBOOT back to ON, and then turn the power supply ON to power the evaluation board again before moving on.

Next, disconnect the flash center adapter cable, close the Flash Center software, and power down the board. Turn the SELFBOOT switch (S2) back to ON. Power the board back on, and now the part self-boots with the loaded program.

I^{3}C

The J5 header is available for connecting to an I^{3}C adapter or external I^{3}C devices. By connecting the I^{3}C adapter to J5, communication with the device can be established.

EVALUATION BOARD SCHEMATICS AND ARTWORK

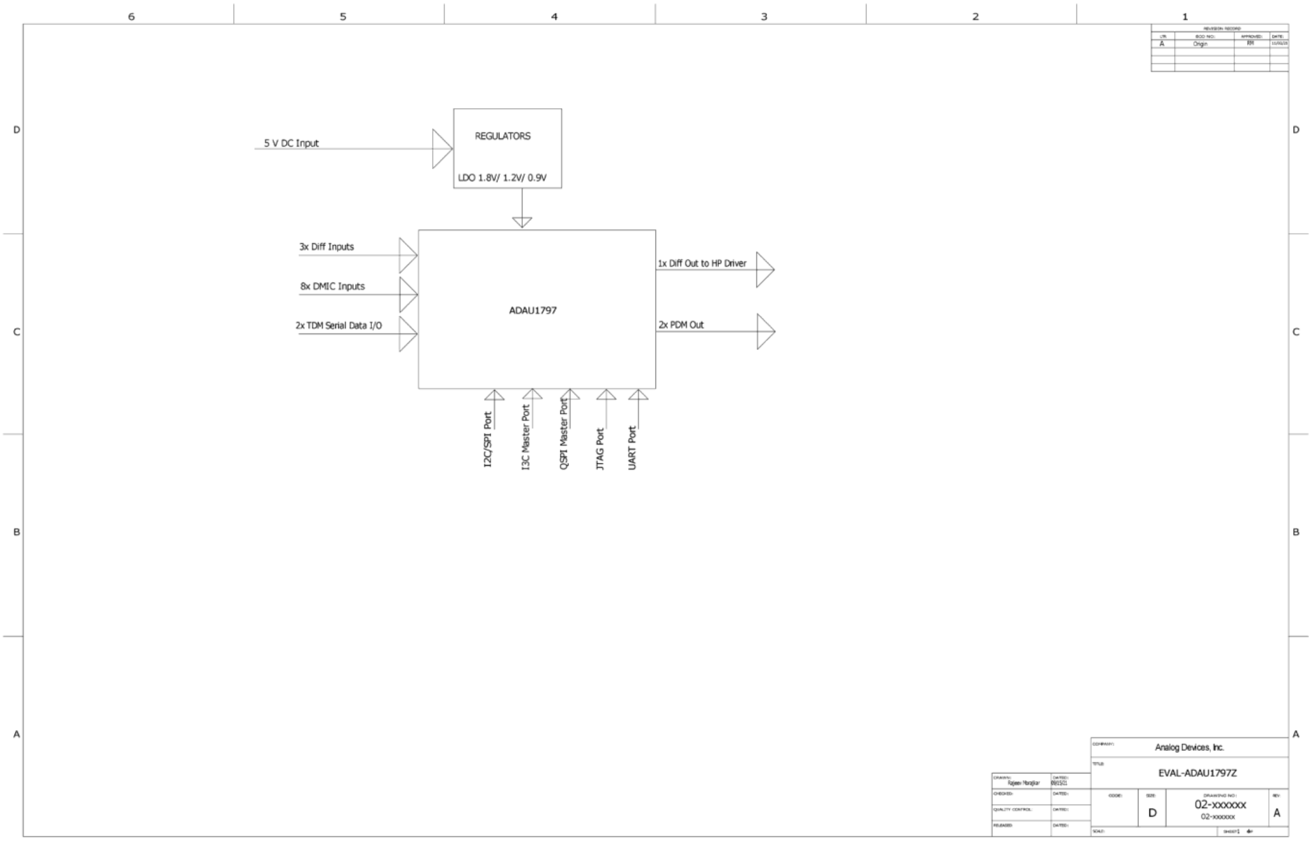


Figure 37. EVAL-ADAU1797Z Schematic Block Diagram

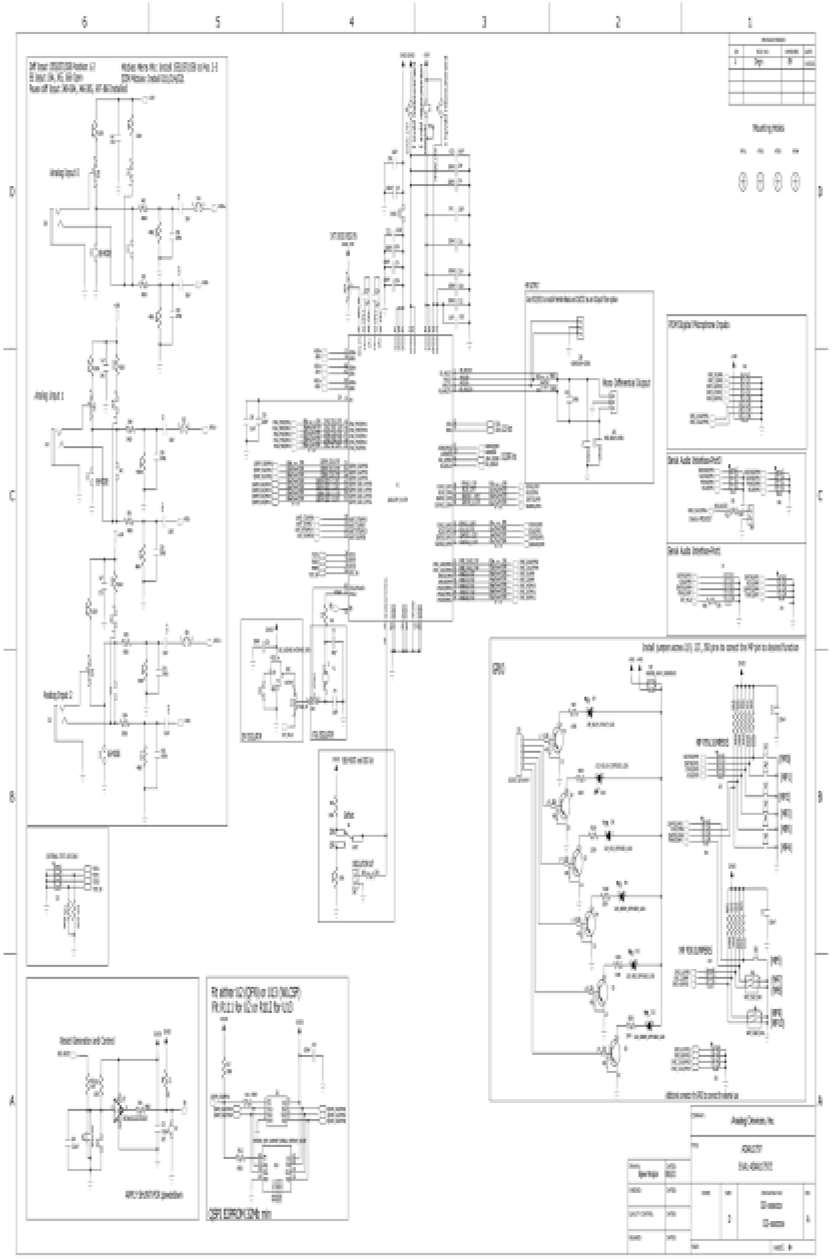


Figure 38. EVAL-ADAU1797Z Schematic

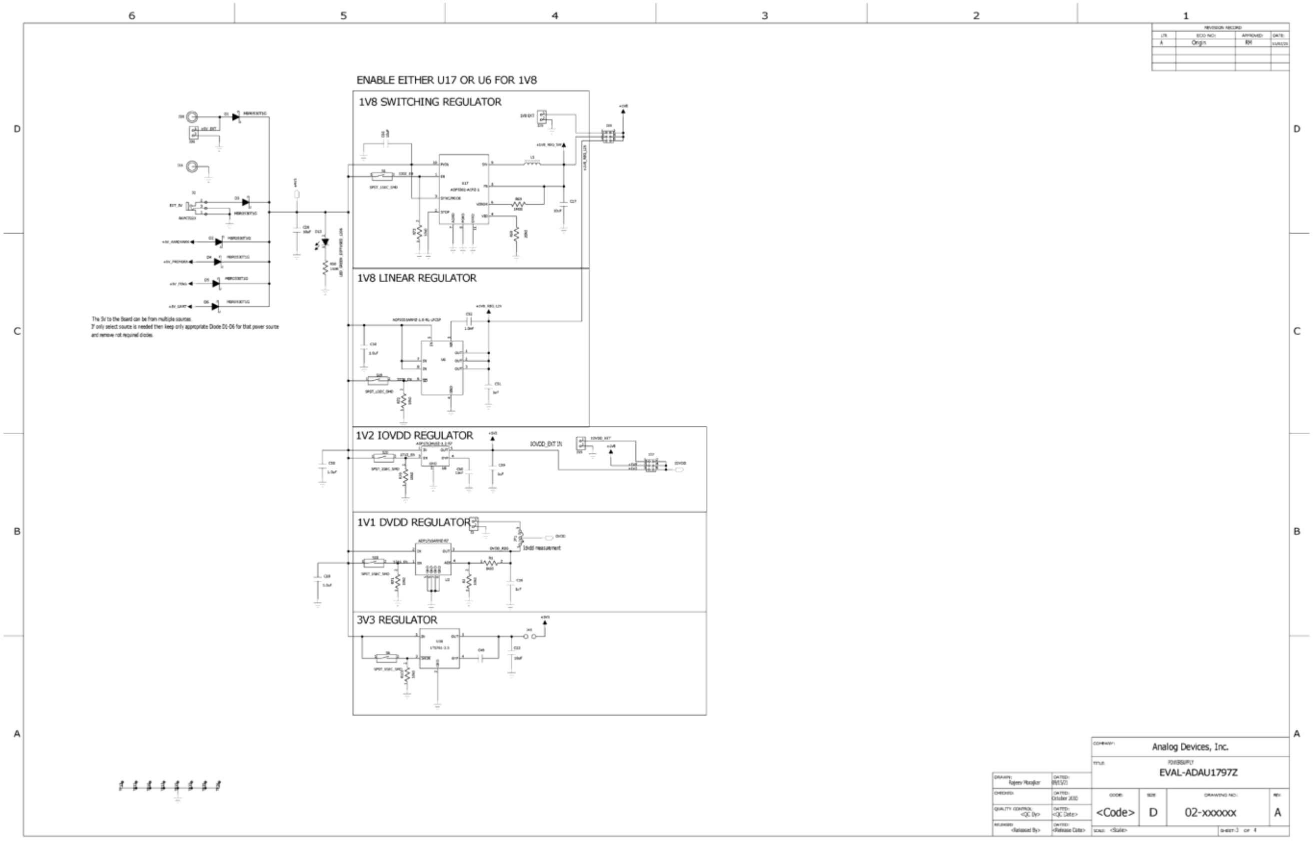


Figure 39. EVAL-ADAU1797Z Schematic Power Supply

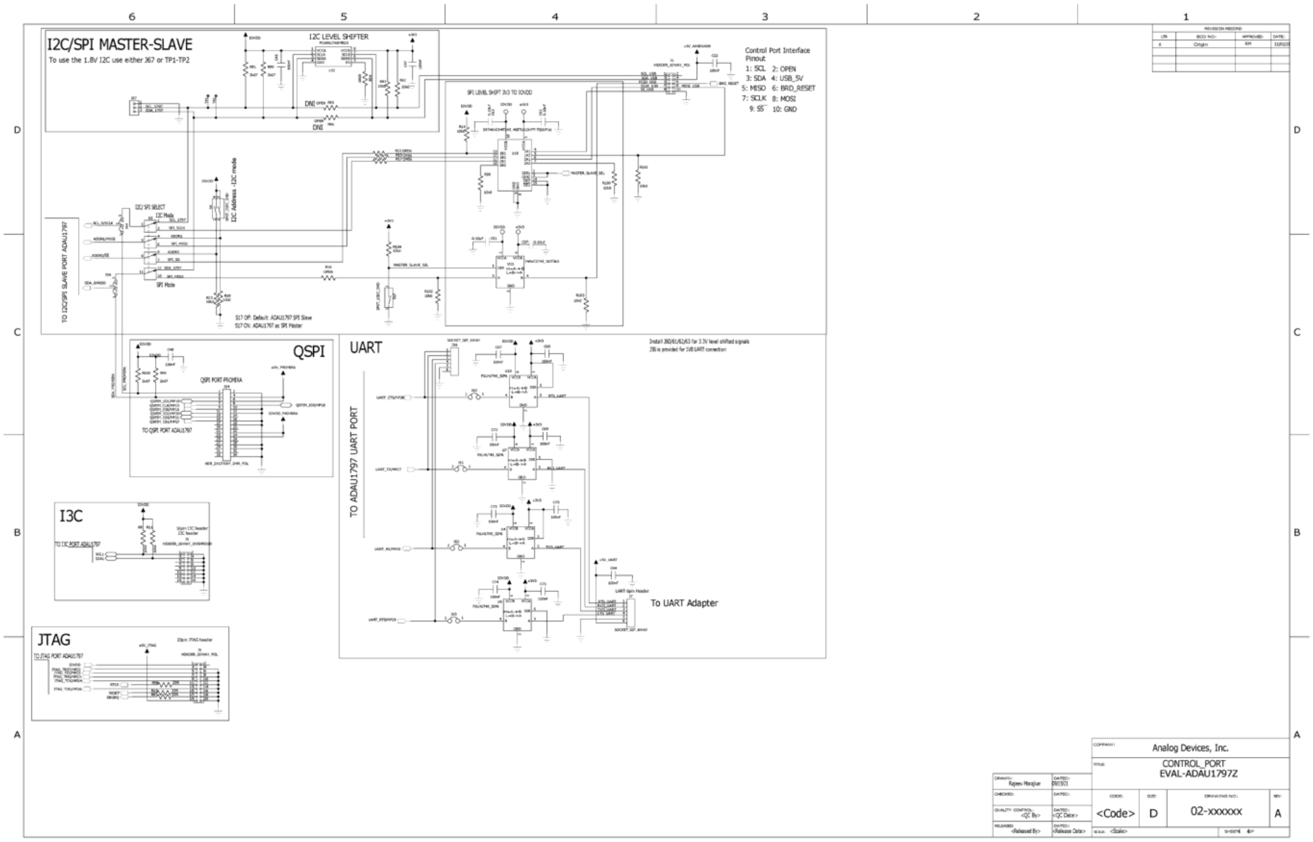


Figure 40. EVAL-ADAU1797Z Schematic Control Port

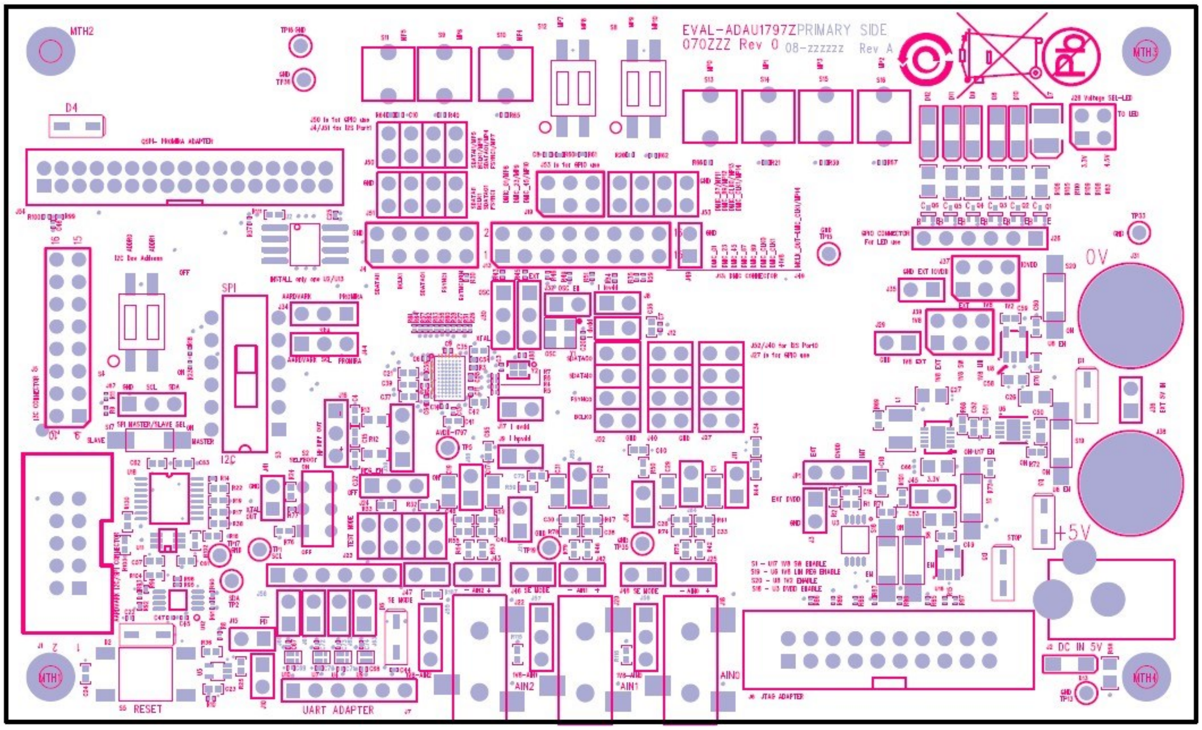


Figure 41. EVAL-ADAU1797Z Layout: Top Assembly

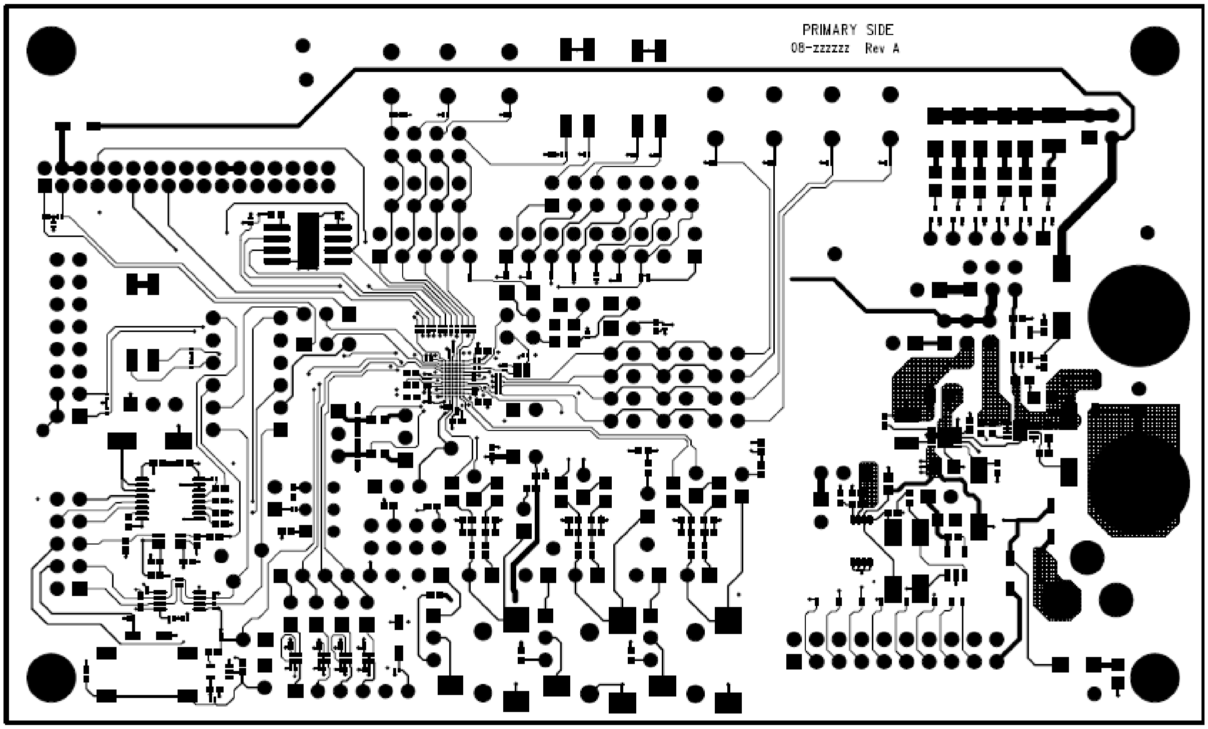


Figure 42. EVAL-ADAU1797Z Layout: Top Copper

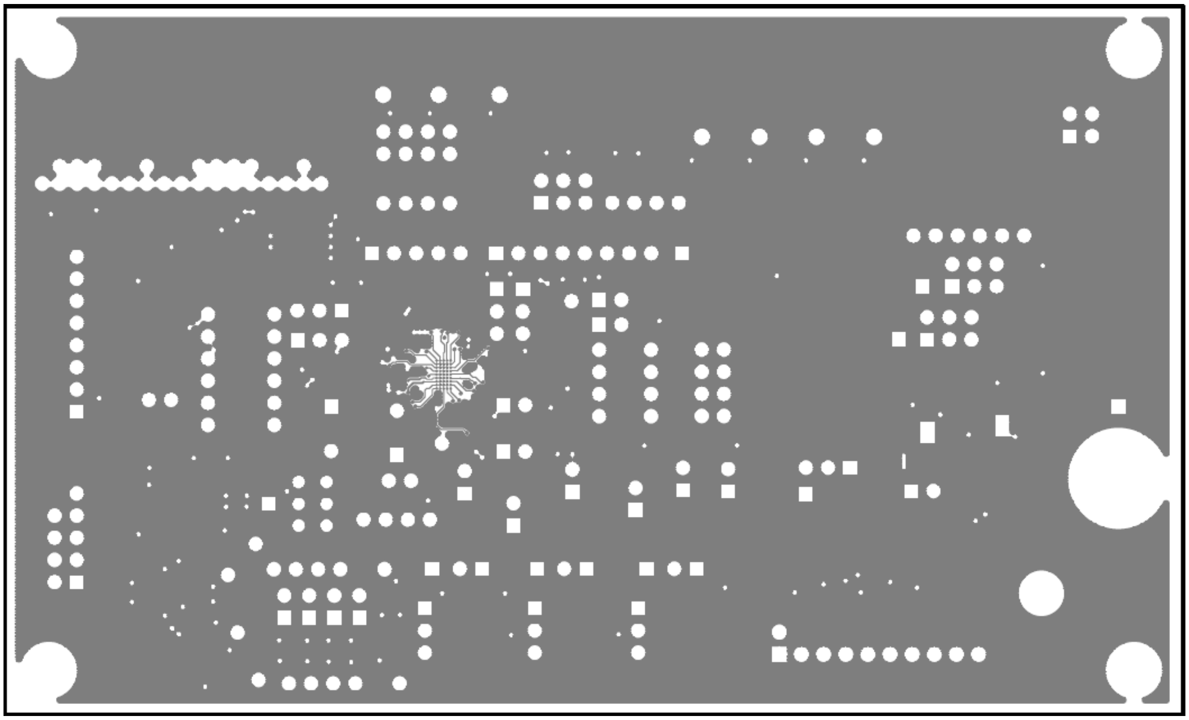


Figure 43. EVAL-ADAU1797Z Layout: Layer2 Ground Plane

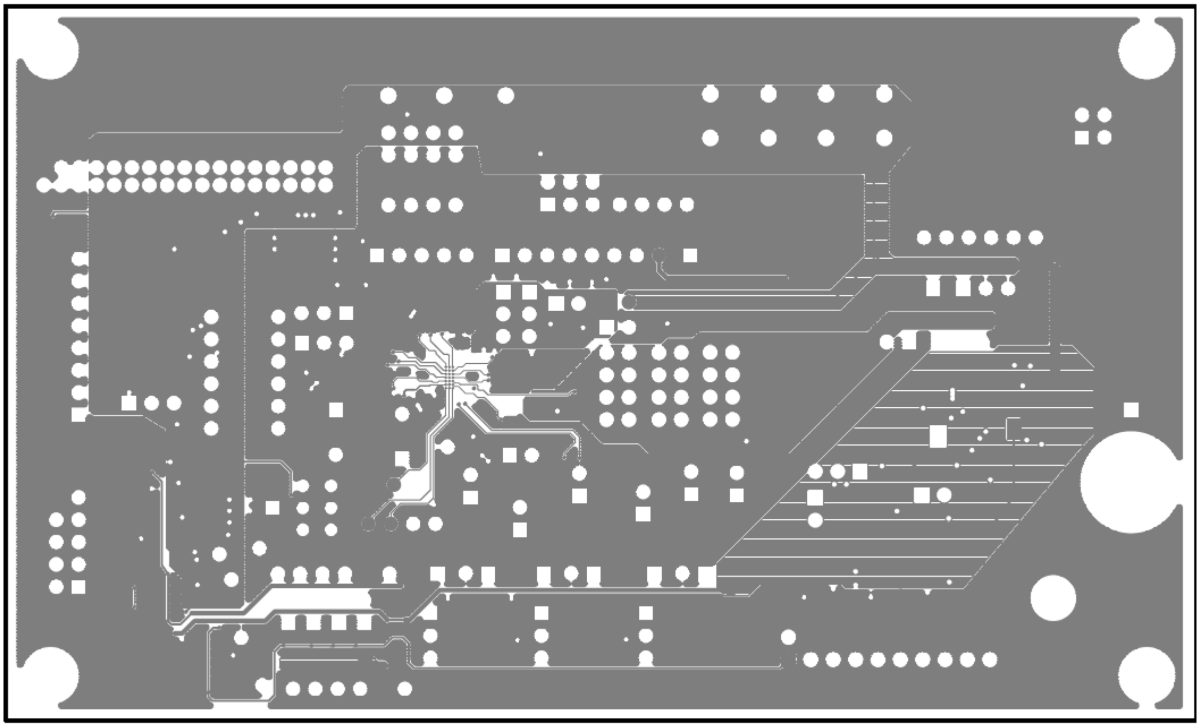


Figure 44. EVAL-ADAU1797Z Layout: Layer3 Power Plane

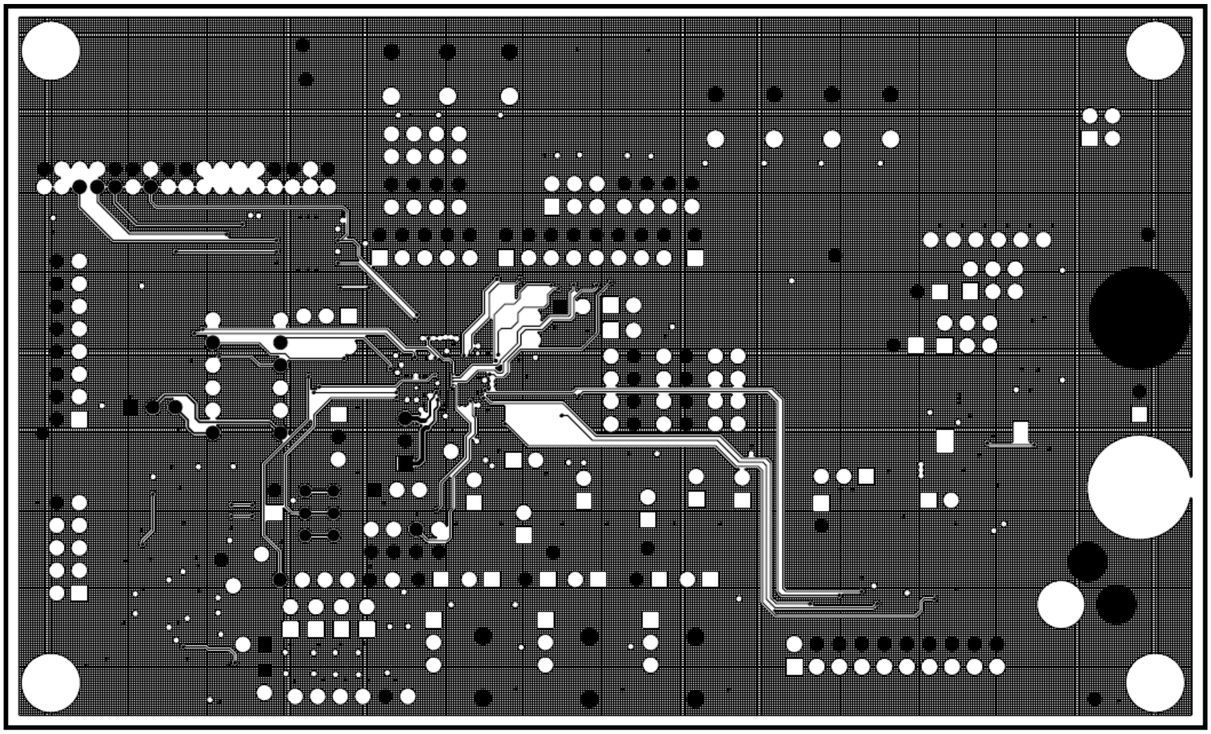


Figure 45. EVAL-ADAU1797Z Layout: Layer4

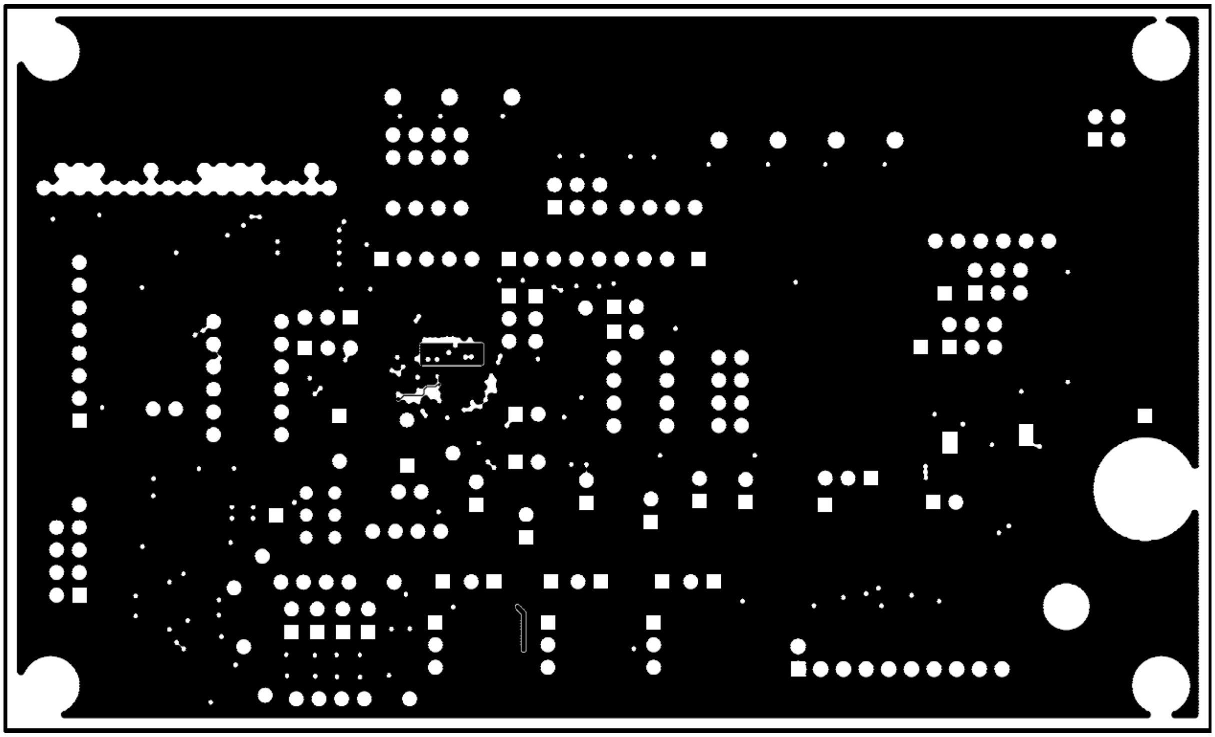


Figure 46. EVAL-ADAU1797Z Layout: Layer5

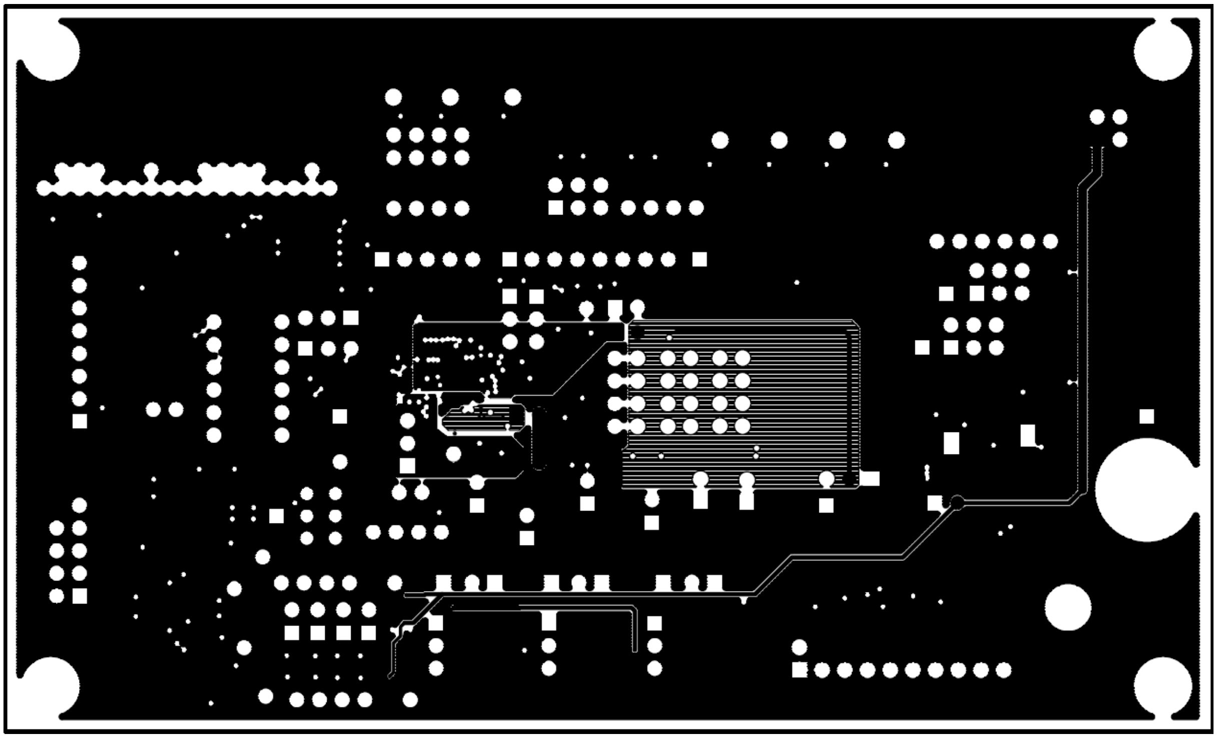


Figure 47. EVAL-ADAU1797Z Layout: Layer6

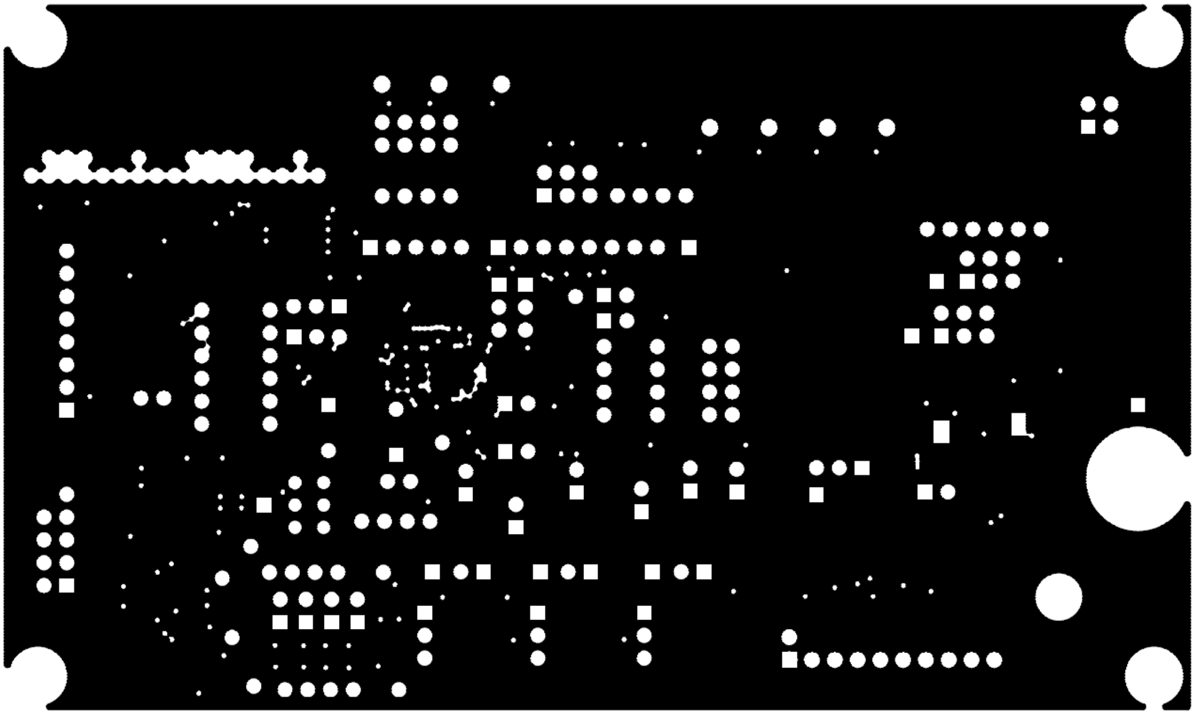


Figure 48. EVAL-ADAU1797Z Layout: Layer7

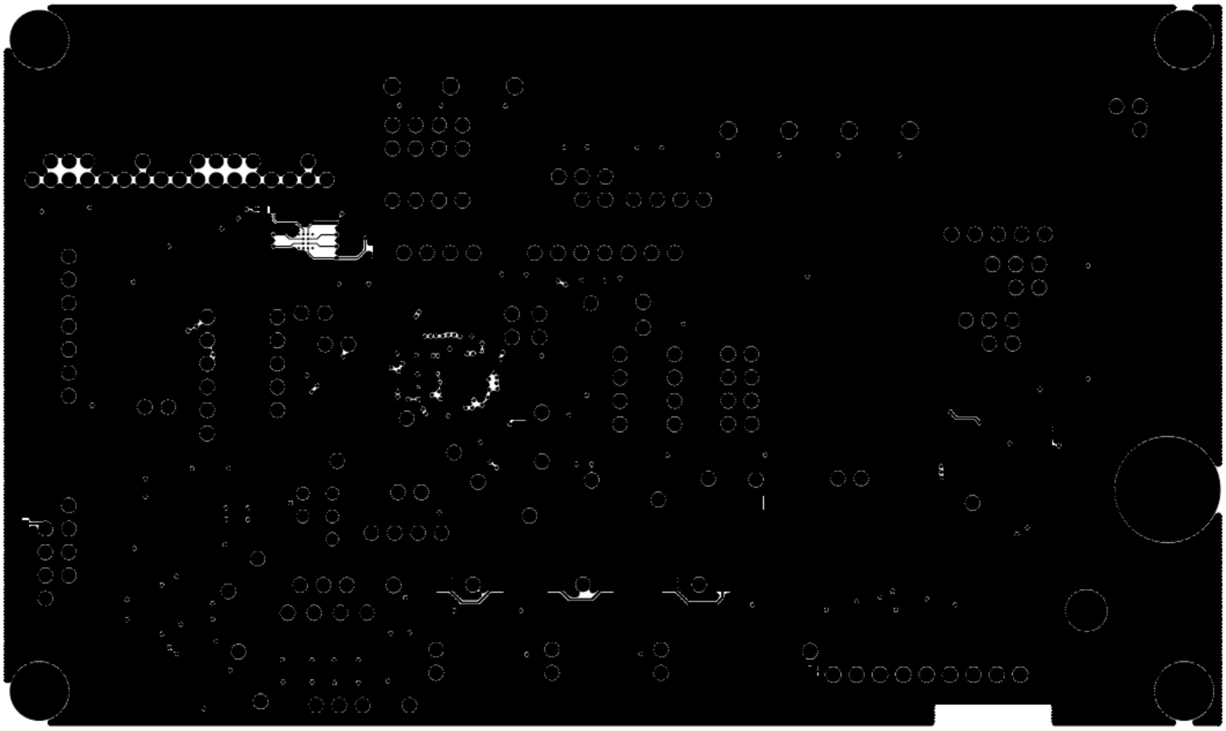


Figure 49. EVAL-ADAU1797Z Layout: Bottom Copper

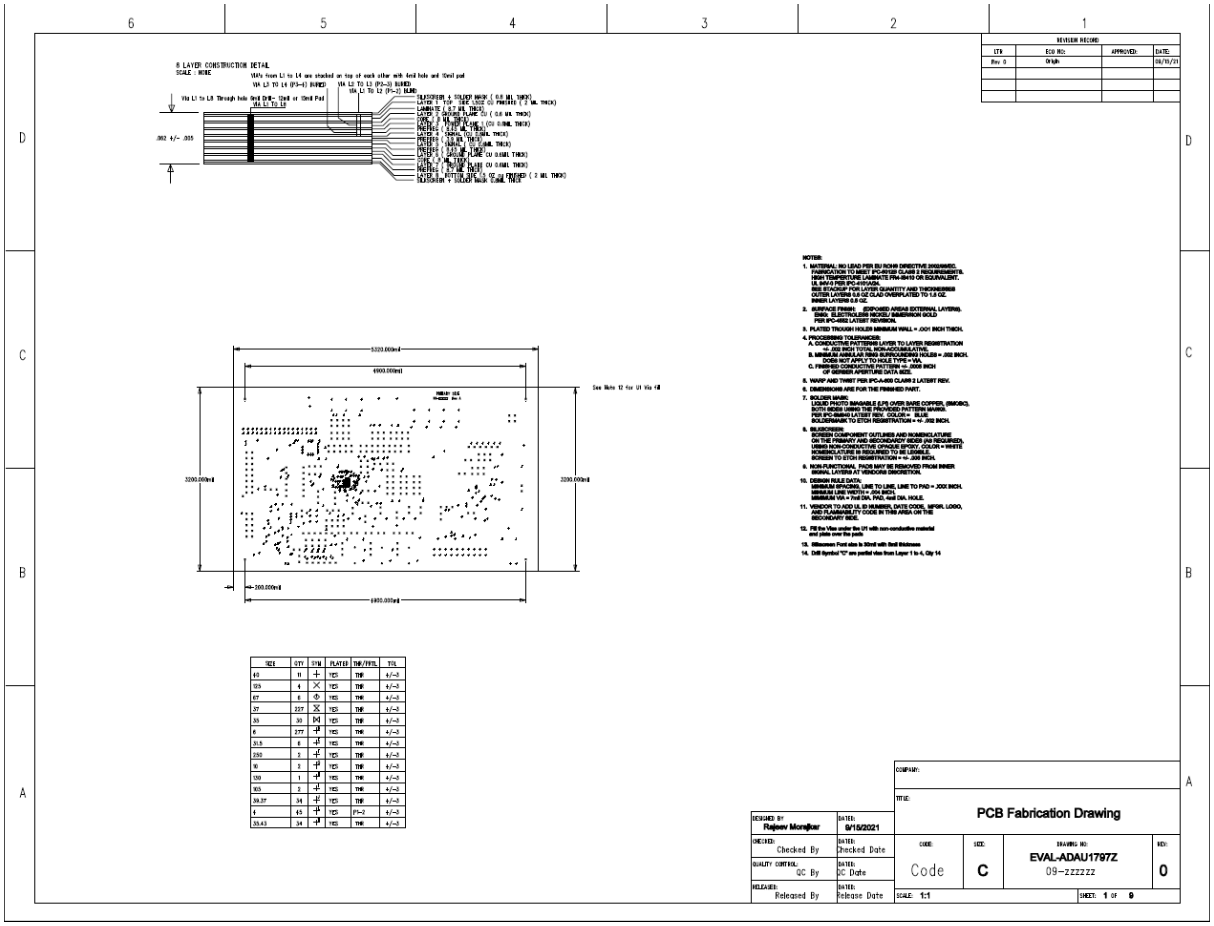


Figure 50. EVAL-ADAU1797Z Layout: Fab Drawing

ORDERING INFORMATION

BILL OF MATERIALS

Table 4. Bill of Materials

QTY	REFERENCE DESIGNATOR	DESCRIPTION	VALUE	SUGGESTED MANUFACTURER	SUGGESTED PART NUMBER
1	C21	Multilayer Ceramic 6.3V X5R (0402)	2.2μF	Digi-Key	587-JMK105BJ225KVHFCT-ND
1	C52	Multilayer Ceramic 50V NP0 (0402)	1.0nF	Digi-Key	490-3244-1-ND
10	C1-2 C17 C19 C26-27 C29 C31 C53 C66	Multilayer Ceramic 10V X7R (0805)	10μF	Digi-Key	490-3905-1-ND
8	C16 C34 C37 C39-40 C51 C59 C75	Multilayer Ceramic 6.3V X7R (0402)	1μF	Digi-Key	311-1702-1-ND
3	C18 C50 C58	Multilayer Ceramic 16V X7R (0603)	1.0μF	Digi-Key	490-3900-1-ND
6	C23-24 C57 C61-63	Multilayer Ceramic 16V X7R (0402)	0.10μF	Digi-Key	490-3261-1-ND
2	C3 C5	Multilayer Ceramic 50V NPO (0201)	16pF	Digi-Key	490-17812-1-ND
6	C30 C28 C33 C38 C43 C46	Do Not Stuff	OPEN	OPEN	OPEN
5	C35-36 C41-42 C65	Multilayer Ceramic 6.3V X5R (0402)	10μF	Digi-Key	490-13211-1-ND

3	C4 C15 C32	Do Not Stuff	OPEN	OPEN	OPEN
2	C49 C60	Multilayer Ceramic 25V X7R (0402)	10nF	Digi-Key	490-6340-1-ND
28	C6-14 C20 C22 C25 C44-45 C47-48 C54-56 C64 C67-74	Multilayer Ceramic 6.3V X5R (0201)	100nF	Digi-Key	490-3167-1-ND
1	D7	Blue 25millicandela 470nm 1210	Blue Clear	Digi-Key	67-1871-1-ND
1	D10	Yellow Diffused 4.0millicandela 585nm 1206	Yellow Diffused	Digi-Key	L62307CT-ND
6	D1-6	Schottky 30V 0.5A SOD123 Diode	MBR0530T1G	Digi-Key	MBR0530T1GOSCT-ND
2	D8 D11	Red Diffused 6.0millicandela 635nm 1206	Red Diffused	Digi-Key	67-1003-1-ND
3	D9 D12-13	Green Diffused 10millicandela 565nm 1206	Green Diffused	Digi-Key	67-1002-1-ND
1	J1	10-Way Shroud Polarized Header	2x5	Digi-Key	MHC10K-ND
1	J2	Mini Power Jack 0.08" R/A TH	RAPC722X	Digi-Key	SC1313-ND
1	J4	10-Way (2X5) UnShroud 0.1" Header	2x5	Digi-Key	S2011EC-05-ND
1	J6	20-way Shroud Polarized	2x10	Digi-Key	MHC20K-ND
1	J28	4-Way Unshrouded Header	2x2	Digi-Key	S2011E-02-ND
1	J54	Header 2x17Way, 2mm Pitch, Shrouded, Polarized	2x17 Pin, 2mm Pitch	Digikey	WM6581-ND
3	J16 J59 J67	3-Pin Header Unshrouded 0.10"	3 Pin Header	Digi-Key	S1011E-03-ND
3	J18 J20 J22	Stereo Mini Jack, SMT	SJ-3523-SMT	Digi-Key	CP-3523SJCT-ND
3	J19 J37 J39	6-Way Unshrouded Header	2x3	Digi-Key	S2011E-03-ND
7	J23 J27 J40 J50-53	8-Way Unshrouded Header Dual Row	2x4	DigiKey	S2011E-04-ND; or cut S2011E-36-ND
9	J24 J30 J33-34 J44 J55 J57-58 JP1	3-Pos SIP Header	3-Jumper	Digi-Key	S1011E-03-ND
7	J25 J42-43 J45-48	2-Pin Header Unshrouded Jumper 0.10"	2-Jumper	Digi-Key	S1011E-02-ND
6	J3 J29 J35-36 J41 J49	2-Pin Header Unshrouded Jumper 0.10"; use Shunt Tyco 881545-2	2-Jumper	Digi-Key	S1011E-02-ND
2	J31 J38	Binding Post Mini Uninsulated Base TH	Nickel Binding Post	Digikey	J587-ND
2	J5 J13	16-Way Unshrouded	2x8	Digi-Key	S2011E-08-ND
3	J7 J26 J56	6-Way Female Socket, 0.1"	1x6	Digi-Key	S7039-ND
17	J8-12 J14-15 J17 J21 J32 J60-66	2-Pin Header Unshrouded Jumper 0.10"; use Shunt Tyco 881545-2	2-Jumper	Digi-Key	S1011E-02-ND
1	L1	2.2µH Inductor	2.2µH	Digi-Key	490-5336-1-ND
4	MTH1-4	HEX Standoff 4-40 Nylon 1/2" Standoff RND 4-40THR .500"L ALIM	4-40 Mounting Hole for Standoff	Digikey	1902CK-ND
4	MTH1-4	Nylon Screw Pan Phillips 4-40	4-40 Panhead Screw	Digikey	H542-ND

6	Q1-6	Pre-Biased Transistor NPN-DTC114YET1G	DTC114 YET1G	Digi-Key	DTC114YET1GO SCT-ND
1	R3	Chip Resistor 1% 50mW Thick Film 0201	1k0	Digi-Key	YAG3431CT-ND
1	R8	Chip Resistor 1% 50mW Thick Film 0201	1K	Digi-Key	YAG3431CT-ND
1	R68	Chip Resistor 1% 63mW Thick Film 0402	20k0	Digi-Key	311-20.0KLRCT-ND
1	R69	Chip Resistor 1% 63mW Thick Film 0402	1M00	Digi-Key	311-1.00MLRCT-ND
1	R94	Do Not Stuff	OPEN	OPEN	OPEN
1	R1	RES 3.74kΩ 1% 1/16W 0402	3k74	Digi-Key	311-3.74KLRCT-ND
1	R25	Chip Resistor 1% 63mW Thick Film 0402	1k00	Digi-Key	311-1.00KLRCT-ND
4	R10 R14 R18 R23	Chip Resistor 1% 50mW Thick Film 0201	10k0	Digi-Key	311-10KNCT-ND
2	R92-93	Chip Resistor 1% 1/20W Thick Film 0201	2k67	Digi-Key	541-CRCW02012K67FNEDCT-ND
3	R107 R115-116	Chip Resistor 1% 100mW Thick Film 0402	100R	Digi-Key	P100LCT-ND
2	R12-13	Chip Resistor 5% 125mW Thick Film 0603	0R00	Digi-Key	P0.0GCT-ND
4	R16-17 R19 R22	Do Not Stuff	OPEN	OPEN	OPEN
12	R2 R38 R70-73 R101-104 R130 R132	Chip Resistor 1% 100mW Thick Film 0402	10k0	Digi-Key	P10.0KLCT-ND
2	R24 R76	Chip Resistor 1% 100mW Thick Film 0402	100k	Digi-Key	P100KLCT-ND
2	R32-33	Chip Resistor 5% 100mW Thick Film 0402	0R00	Digi-Key	P0.0JCT-ND
9	R36 R42 R48 R53-54 R75 R79 R111-112	Chip Resistor 5% 63mW Thick Film 0402	0R00	Digi-Key	P0.0JCT-ND
33	R4-7 R15 R26-31 R34-35 R43 R45-46 R49 R51 R56-57 R77 R80-89 R97-98	RES 33R Ω 1/20W 5% 0201 SMD	33R	DIGIKEY	311-33NCT-ND
6	R41 R47 R52 R55 R74 R78	Chip Resistor 1% 63mW Thick Film 0402	49k9	Digi-Key	541-49.9KLCT-ND
3	R44 R50 R59	Chip Resistor 1% 63mW Thick Film 0402	2k00	Digi-Key	P2.00KLCT-ND
7	R58 R63 R105-106 R108-110	Chip Resistor 1% 125mW Thick Film 0805	150R	Digi-Key	P150CCT-ND
14	R9 R11 R20-21 R37 R39-40 R60-62 R64-67	Chip Resistor 1% 50mW Thick Film 0201	10k0	Digi-Key	311-10KNCT-ND
4	R90-91 R99-100	Chip Resistor 1% 1/20W Thick Film 0201	2k67	Digi-Key	541-CRCW02012K67FNEDCT-ND
2	R95-96	Chip Resistor 5% 50mW Thick Film 0201	OPEN	Digi-Key	P0.0JCT-ND
1	S2	SPDT Slide Switch PC Mount	SPDT	Digi-Key	EG1918-ND
1	S3	4PDT Slide Switch Vertical Break-Before-Make	4PDT Slide	Digi-Key	450-1633-ND
1	S5	Tact Switch 6mm Gull Wing	SPST-NO	Digi-Key	450-1133-ND
6	S1 S6 S17-20	1 Section SPST SMD	1x SPST	Digi-Key	563-1003-1-ND

3	S4 S8 S12	2 Section SPST SMD Switch Raised Act	2x SPST	Digi-Key	CT2192LPST-ND
7	S9-11 S13-16	Tact Switch Long Stroke (Normally open)	SPST-MOM	Digi-Key	SW426-ND
1	TP5	Mini Test Point White .1" OD	5002	Digi-Key	5002K-ND
10	TP1-2 TP13 TP15-17 TP19 TP33 TP35-36	Mini Test Point White .1" OD	5002	Digi-Key	5002K-ND
1	U1	Low Latency Audio CODEC with DSP ADAU1797	ADAU1797	Analog Devices Inc.	ADAU1797BCBZ-RL
1	U2	32Mbit QSPI CSerial EEPROM-WLCSP-12	MX25U3232FM2I02	Digi-Key	1092-MX25U3232FM2I02-ND
1	U3	Adjustable Low-Dropout Voltage Regulator	ADP1715ARM Z-R7	Digikey	ADP1715ARMZ-R7CT-ND
1	U5	SNGL BUS BUFF NON-INV GATE OPEN DRAIN SC70-5	SN74AUC1G07DCKR	DigiKey	296-12464-1-ND
1	U6	High-Accuracy, Ultralow IQ, 500mA, anyCAP Low Dropout Regulator	ADP3335ACPZ-1.8-RL	Digikey	ADP3335ACPZ-1.8-R7CT-ND
1	U8	Fixed Low-Dropout Voltage Regulator 1.2V	ADP1713AUJZ-1.2-R7	Digikey	ADP1713AUJZ-1.2-R7CT-ND
1	U11	IC TXRX DUAL 3ST 6TSSOP	74AVC1T45	Digikey	568-9222-1-ND
1	U12	IC I2C BUS REPEATER 8-TSSOP PCA9617A	PCA9617ADMR2G	DigiKey	2156-PCA9617ADMR2G-OS-ND
1	U13	32Mbit QSPI CSerial EEPROM-WLCSP-12	MX25U3232FM2I02	Digi-Key	1092-MX25U3232FM2I02-ND
1	U16	Linear Regulator 3.3V, 100mA TSOT-23-5	LT1761-3.3	Digikey	LT1761IS5-3.3#TRMPBFCT-ND
1	U17	Ultra Low Power Step-Down Regulator ADP5302-ACPZ-1	ADP5302ACPZ-1	DIGIKEY	ADP5302ACPZ-1-R7CT-ND
1	U18	IC 4BIT DUAL BUS TXRX TSSOP16	SN74AVC4T245PWR	DigiKey	296-18056-1-ND
4	U4 U7 U9-10	IC TRANSLATOR BIDIR SIP6	FXLH1T45L6X	Digikey	FXLH1T45L6XFSCT-ND
1	Y1	24.576MHz Fixed SMD Oscillator 1.8 - 3.3VDC	24.576MHz	Digi-Key	535-11729-1-ND
1	Y2	Crystal 24.576MHz SM1612	24.576MHz	Digi-key	

REVISION HISTORY

9/2023—Revision 0: Initial Version

ASSUMED BY ANALOG DEVICES FOR ITS USE, NOR FOR ANY INFRINGEMENTS OF PATENTS OR OTHER RIGHTS OF THIRD PARTIES THAT MAY RESULT FROM ITS USE. SPECIFICATIONS ARE SUBJECT TO CHANGE WITHOUT NOTICE. NO LICENCE, EITHER EXPRESSED OR IMPLIED, IS GRANTED UNDER ANY ADI PATENT RIGHT, COPYRIGHT, MASK WORK RIGHT, OR ANY OTHER ADI INTELLECTUAL PROPERTY RIGHT RELATING TO ANY COMBINATION, MACHINE, OR PROCESS WHICH ADI PRODUCTS ALL INFORMATION CONTAINED HEREIN IS PROVIDED “AS IS” WITHOUT REPRESENTATION OR WARRANTY. NO RESPONSIBILITY IS OR SERVICES ARE USED. TRADEMARKS AND REGISTERED TRADEMARKS ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS.