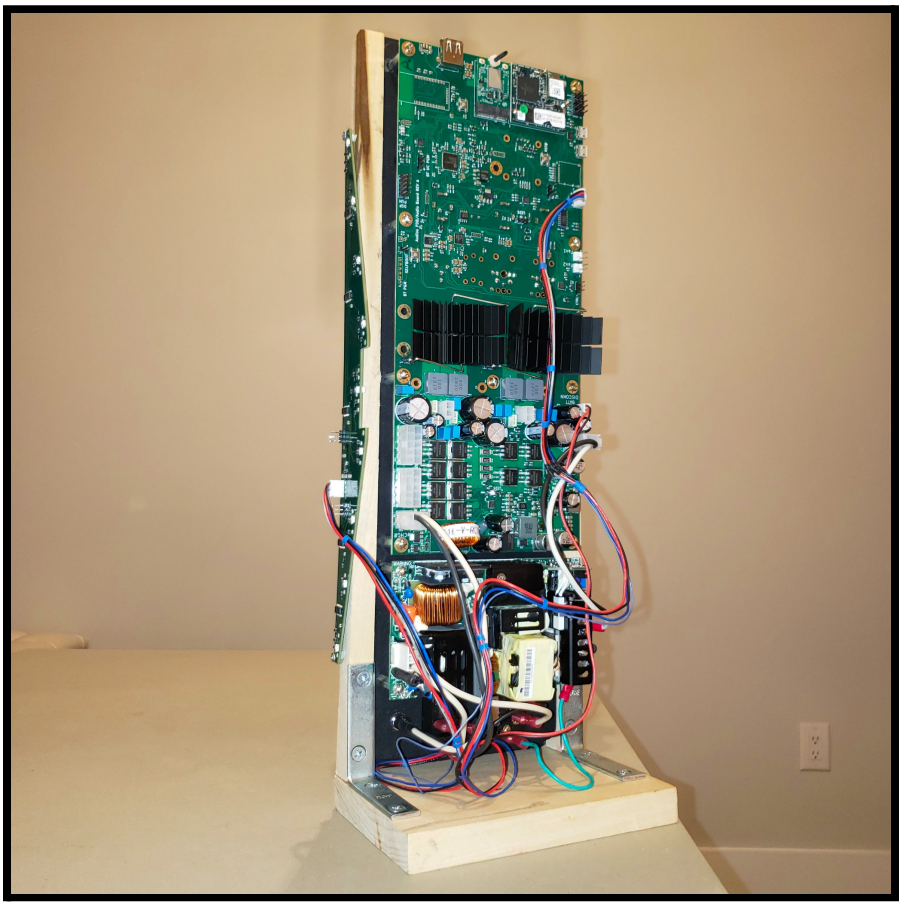
AUDIOS

TABLE OF CONTENTS

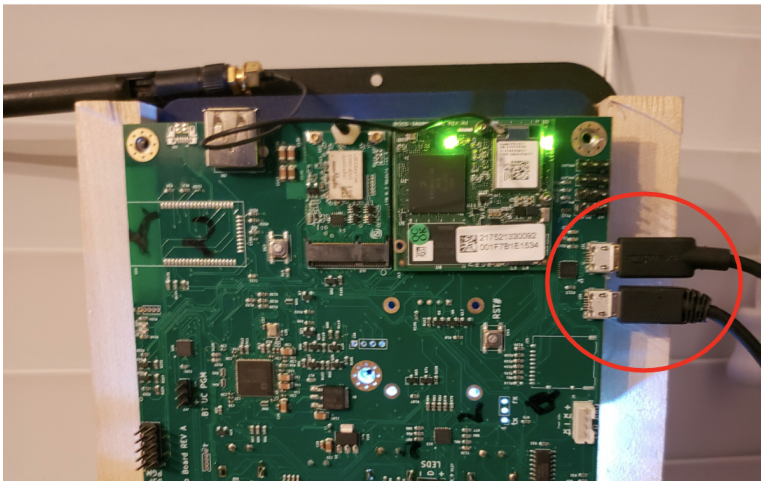


Drafted by Erik Young, erik.young@audios.com, Jan. 2022

[1] Connecting to the Audios Board via Serial

The NXP ARM module has two Micro USB output ports, one for serial, the other for mounting or flashing the 16GB eMMC. You'll need to use both ports to configure the Yocto Linux OS used by Audios.

Step 1: Connect both USB ports to a Linux PC or Laptop. We installed Ubuntu 20.04 on our Host PC. Power on the Audioboard.



Step 2: If you are running Ubuntu 20.04 the Silicon Labs CP210x USB to UART driver should already be installed and loaded. If not, then download and install it from here. After installation check to see if it is loaded by typing `lsmod` on the command line. You should see the driver is loaded and uses `usbserial`.

```
$ lsmod | grep cp210x
```

```
sasuke@sasuke-laptop:~$ lsmod | grep cp2
cp210x                 40960  0
usbserial              53248  1 cp210x
```

After verifying the driver is loaded, verify the serial device is available for use in the `/dev` directory. If it is present then the host PC is now able to connect to the ARM.

```
$ ls -lah /dev/ttyUSB0
```

```
sasuke@sasuke-laptop:~$ ls -lah /dev/ttyUSB0
crw-rw---- 1 root dialout 188, 0 Jan 10 17:08 /dev/ttyUSB0
```

Step 3: Add your `username` to the `dialout` and `tty` groups so that you don't need `root` or `sudo` to connect vial `ttyUSB0`.

```
$ sudo adduser <username> dialout
$ sudo adduser <username> tty
```

```
sasuke@sasuke-laptop:~$ sudo adduser sasuke dialout
Adding user `sasuke' to group `dialout' ...
Adding user sasuke to group dialout
Done.
```

Install `screen` and or `minicom`. We use both at different times depending on our development needs.

```
$ sudo apt install screen minicom
```

Start `screen` or `minicom` with the following. **NOTE**: With `minicom` you have to go to settings and turn Flow Control off.

```
$ [sudo] screen -AUh 10000 /dev/ttyUSB0 115200
```
Or
```
$ [sudo] minicom -D /dev/ttyUSB0
```

```
Welcome to minicom 2.7.1

OPTI+-----------------------------------------------------------------------+
Comp| A -    Serial Device      : /dev/ttyUSB0                              |
Port| B - Lockfile Location     : /var/lock                                 |
    | C -   Callin Program      :                                           |
Pres| D -  Callout Program      :                                           |
    | E -    Bps/Par/Bits       : 115200 8N1                                |
    | F - Hardware Flow Control : No                                        |
    | G - Software Flow Control : No                                        |
    |                                                                       |
    |    Change which setting?                                              |
    +-----------------------------------------------------------------------+
```

Once the serial program is started hit the `enter key`. For login name enter `root` and hit the `enter key` when asked for a password. There is no root user password.

```
pico-imx8mm login: root
Password:
[20418.764002] audit: type=1006
root@pico-imx8mm:~#
```

[2] Flashing the Audios Board with the raw image

The Linux image used to flash the audio board can be downloaded from this link imx8mm-pico-audios-base.tar.gz. It is a compressed image of the latest custom Yocto Audios recipe. With this version we have verified the WiFi, GPIO's, I2C, I2S and UART interfaces defined in the Audios device tree all work.

Unpack the image to your local directory

```
$ tar xvfz imx8mm-pico-audios-base.tar.gz
```

```
sasuke@sasuke-laptop:~/nxp/images$ tar xvfz imx8mm-pico-audios-base.tar.gz
./imx8mm-pico-audios-base.raw
```

Next must mount the eMMC as a USB drive. Connect to the serial port, then reset the Linux OS or power cycle the audio board. During the boot process, the `u-boot` bootloader will prompt you to `Hit any key to stop autoboot: 2`. Hit any key and the `u-boot-audio=>` will appear like below.

```
Normal Boot
Hit any key to stop autoboot:  0
u-boot-audios=>
```

Present the eMMC to the Linux OS with the following command

$ ums 0 mmc 2

```
u-boot-audios=>  ums 0 mmc 2
UMS: LUN 0, dev 2, hwpart 0, sector 0x0, count 0x1d34000
-
```

Open a terminal on your Linux host PC. On our host PC, both `/dev/sdb1` and `/dev/sdb2` automounted. Check to see if the eMMC automounted on your PC. It can take up to 30 secs.

$ df -th

```
sasuke@sasuke-laptop:~$ df -Th
df: /run/user/1000/doc: Operation not permitted
Filesystem     Type      Size  Used Avail Use% Mounted on
udev           devtmpfs  3.9G     0  3.9G   0% /dev
tmpfs          tmpfs     788M  1.7M  787M   1% /run
/dev/sda8      ext4       57G   12G   43G  21% /
tmpfs          tmpfs     3.9G   54M  3.8G   2% /dev/shm
tmpfs          tmpfs     5.0M  4.0K  5.0M   1% /run/lock
tmpfs          tmpfs     3.9G     0  3.9G   0% /sys/fs/cgroup
/dev/sda7      ext4      134G   79G   49G  63% /mnt/oldroot
/dev/sda1      vfat      120M  5.2M  114M   5% /boot/efi
tmpfs          tmpfs     788M   32K  788M   1% /run/user/1000
/dev/sdb2      ext4      1.1G  583M  387M  61% /media/sasuke/root
/dev/sdb1      vfat       84M   23M   61M  27% /media/sasuke/boot
```

If your eMMC drive also automounted, then you will see `/dev/sdb1` and `/dev/sdb2` associated with a mount point similar to the image above. Next, you must **unmount** `/dev/sdb1` and `/dev/sdb2` before flashing with the `dd` command below.

$ sudo umount /dev/sdb*

If the eMMC doesn't automount then simply verify it exists with the command `ls -lah /dev/sdb*`. After you've verified `/dev/sdb1` and `/dev/sdb2` exists, you can proceed to copy the image to the eMMC.

$ sudo dd bs=4M if=./imx8mm-pico-audios-base.raw of=/dev/sdb status=progress

```
sasuke@sasuke-laptop:~/nxp/images/audios$ sudo dd bs=4M if=imx8mm-pico-audios-base.raw of=/dev/sdb status=progress
1132462080 bytes (1.1 GB, 1.1 GiB) copied, 16 s, 70.5 MB/s
270+0 records in
270+0 records out
1132462080 bytes (1.1 GB, 1.1 GiB) copied, 16.063 s, 70.5 MB/s
```

Now return to the serial terminal (you have connect to the Yocto Audios Linux image). Disconnect the eMMC from the host by holding `CTRL+C`. Next, reset the ARM.

```
$ "hold CTRL then press C"
$ reset
```

Linux should boot Successfully.

[3] Compiling the Audios Kernel and Modules

Before compiling the kernel, download and install the toolchain by following the steps in this article:

Preparing a Toolchain

Download the Linux kernel found at this link Linux-5.4.70.tar.gz. It contains the base Audios `.config` settings used to compile the kernel and device tree. Unpack the image to your local directory and change to the extracted directory.

```
$ tar xvfz Linux-5.4.70.tar.gz
$ cd linux-5.4.70
```

Export the toolchain environment before compiling

```
$ export ARCH=arm64
$ export CROSS_COMPILE=/opt/gcc-linaro-6.4.1-2017.11-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-
$ export PATH=$PATH:/opt/gcc-linaro-6.4.1-2017.11-x86_64_aarch64-linux-gnu/bin/
```

Compile the kernel

```
$ make -j4
```

Compile the modules

```
$ mkdir modules_imx
$ make modules_install INSTALL_MOD_PATH=modules_imx
```

[4] Compiling the Audios Device Tree

The Audios device tree is included and compiled with the Linux kernel. To edit the Audios device tree, it can be found at the following location in the kernel directory.

```
$ linux-5.4.70/arch/arm64/boot/dts/freescale/imx8mm-pico-audios.dts
```

To compile the Audios device tree, simply compile the Audios kernel, and the `.dts` will compile as well.

[5] Installing the Kernel and Device Tree to the Audios Board

After compiling the kernel source, you have the option of copying the kernel and its modules over to the Yocto Audios Linux image. In the same way, if you updated the device tree, then it is necessary to copy it over to the Yocto Audios Linux image in order to use it.

Prior to copying either the kernel or device tree, you must mount the eMMC as a USB drive first. If the eMMC automounts as `/dev/sdb1` and `/dev/sdb2`, you can proceed to the respective copy instruction below. If not then you'll have to mount the eMMC with the commands below.

```
$ sudo mkdir /mnt/boot
$ sudo mkdir /mnt/root
$ sudo mount /dev/sdb1 /mnt/boot
$ sudo mount /dev/sdb2 /mnt/root
```

To copy the kernel over to the boot partition. First, rename the original kernel so that you can use it as a backup if necessary.

```
$ sudo mv /mnt/boot/Image /mnt/boot/Image.backup
```

Copy the new kernel to the boot partition

```
$ sudo cp arch/arm64/boot/Image /media/boot/
```

The kernel needs the modules (i.e. drivers) in order to work. So copy over the modules to the root partition

```
$ sudo cp -Rv modules_imx/lib/modules/5.4.47-audios-gc97e635ed13b-dirty /media/root/lib/modules/
```

After copy finishes, unmount the eMMC and reset the ARM from the serial terminal

```
$ sudo umount /dev/sdb*
```

[6] Sending I2C commands to the DSP via Command Line

The ADAU1451 DSP I2C pins are predefined and tested in the `imx8mm-pico-audios.dtb` device tree file as an example/reference. We tested the I2C ARM to DSP interface so that the audio driver team can easily integrate it with the sound driver. Feel free to modify the ADAU1451 I2C device tree entry if the sound driver needs further customization.

Before sending test commands, first, verify the DSP is present. NOTE: The I2C address `0x76` defined in the DSP firmware uses 8-bit address decoding, while Yocto Linux I2C address `0x3b` uses 7-bit decoding and defines the software address of the DSP.

List and detect the available I2C devices with `i2cdetect -y 2`. NOTE: Linux I2C Bus 2 maps to the device tree I2C3

$ i2cdetect -y 2

```
root@pico-imx8mm:~# i2cdetect -y 2
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- 3b 3c -- -- --
40: 40 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

The DSP is located at I2C bus 0x3b. Write to a register with the sample command below where 0xF1 0x00 is the 2-byte ADAU1451 0xF100 register address.

$ i2cset 2 0x3b 0xF1 0x00 0x00 0x02 i

```
root@pico-imx8mm:~# i2cset 2 0x3b 0xF1 0x00 0x00 0x02 i
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will write to device file /dev/i2c-2, chip address 0x3b, data address
0xf1, data 0x00 0x00 0x02, mode i2c block.
Continue? [Y/n] y
```

Read the DSP register to verify the write worked. This is done in two steps. First, set the DSP internal pointer register to the corresponding 2-byte register `0xF100` we wrote to.

$ i2cset -y 2 0x3b 0xF1 0x00

Now read the register value back

$ i2cget 2 0x3b 0xF1 w

```
root@pico-imx8mm:~# i2cget 2 0x3b 0xF1 w
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will read from device file /dev/i2c-2, chip address 0x3b, data address
0xf1, using read word data.
Continue? [Y/n] y
0x0200
```

[7] Programming the DSP

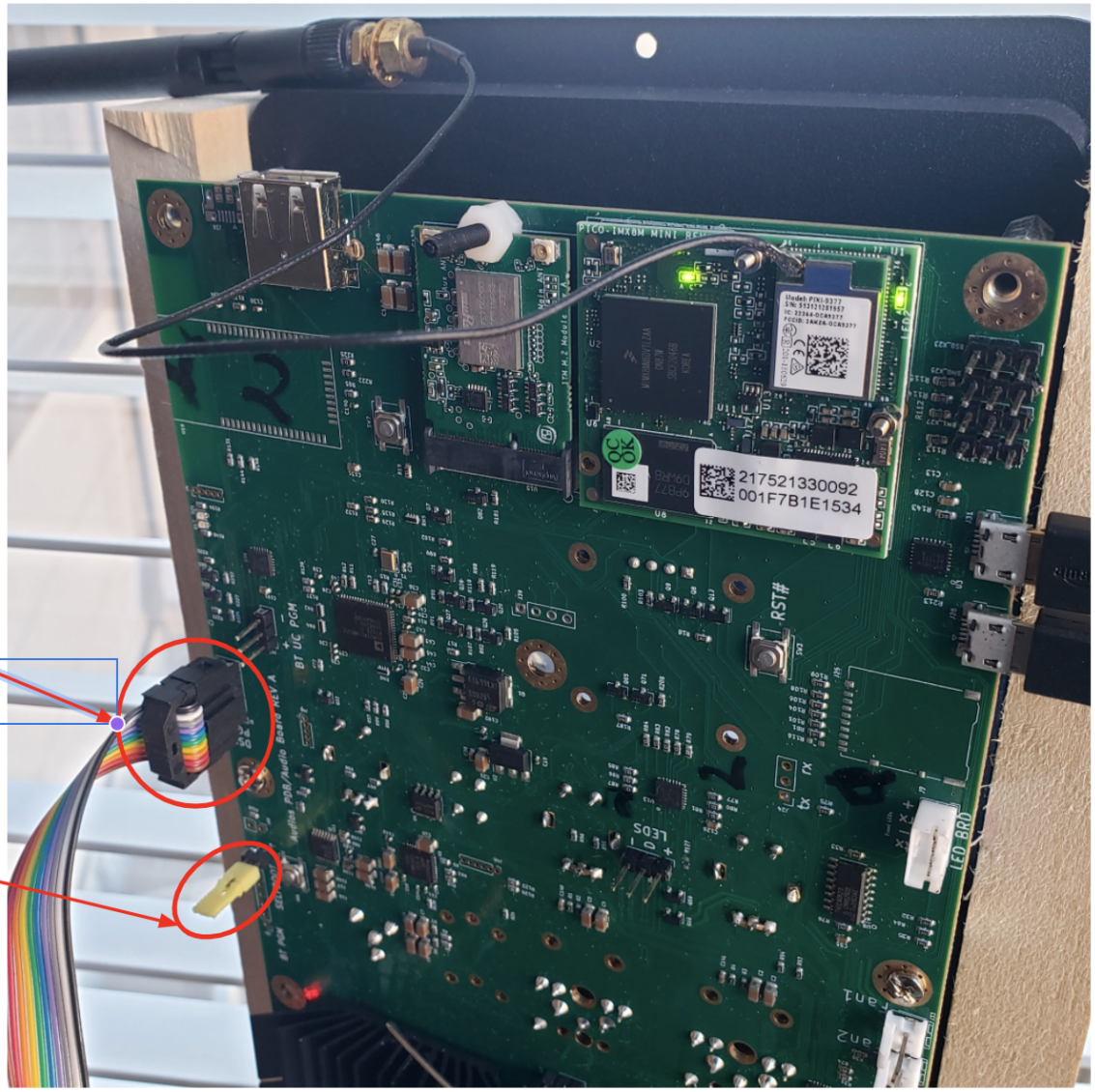
In order to save firmware to the DSP EEPROM, you must first connect the jumper like in the image below. After the programming is finished, remove the jumper so that the DSP loads its boot image from the EEPROM instead of the connected Windows PC or Laptop.

Step 1: Connect the jumper and programming cable to the board like below, then power on the Audio board.

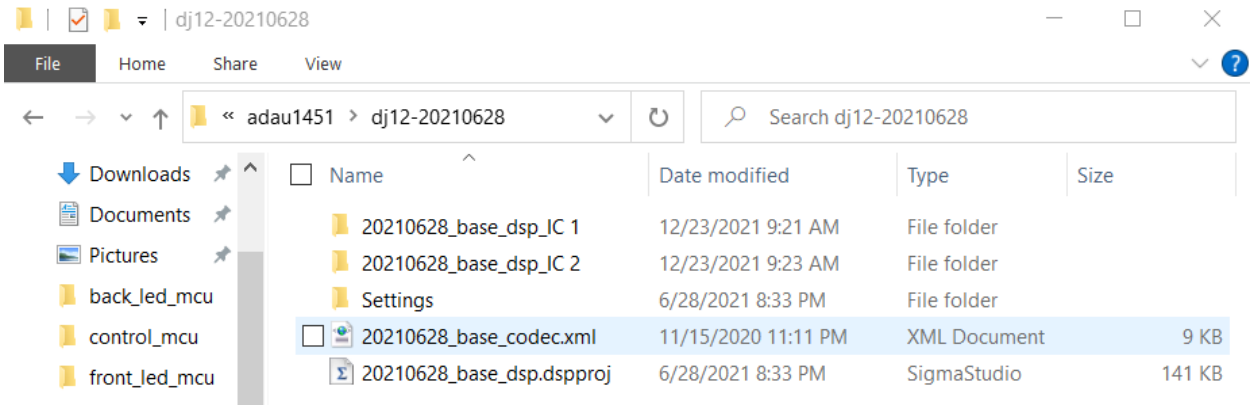
DSP Programmer
We use the Sigma programmer cable

DSP Jumper
The DSP must have both pins jumped in order to save the program to the DSP Memory.

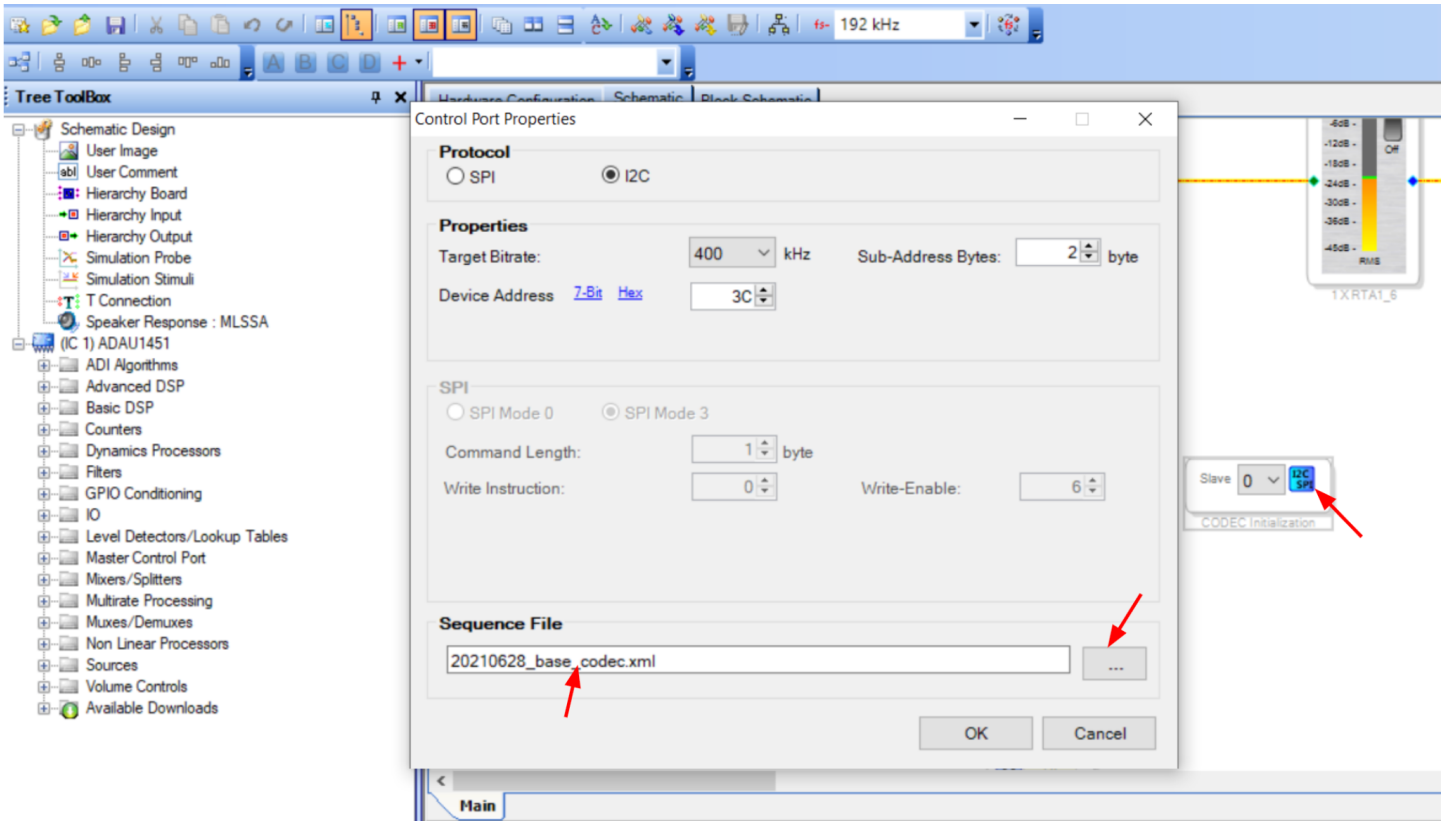
Remove the jumpers for normal boots



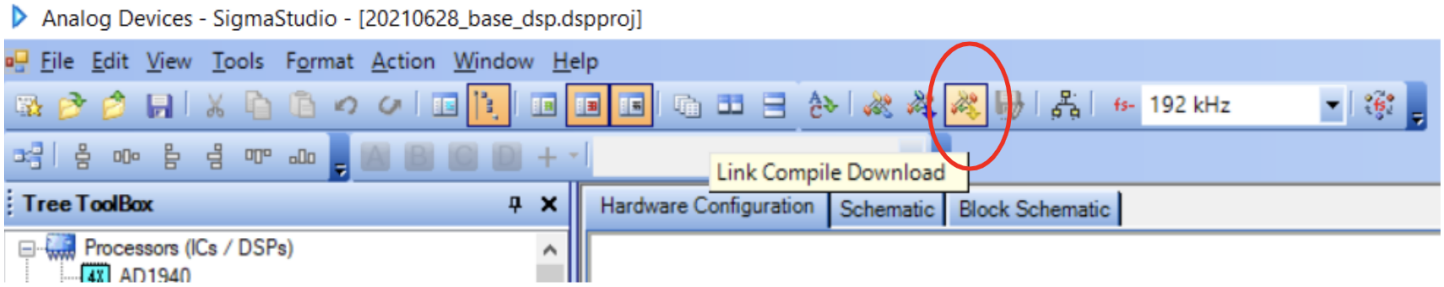
Step 2: Download the dj12-20210628.zip file and extract the folder. Enter the `dj12-20210628` directory, right-click the file `20210628_base_dsp.dspproj`, and select open with Sigma Studios



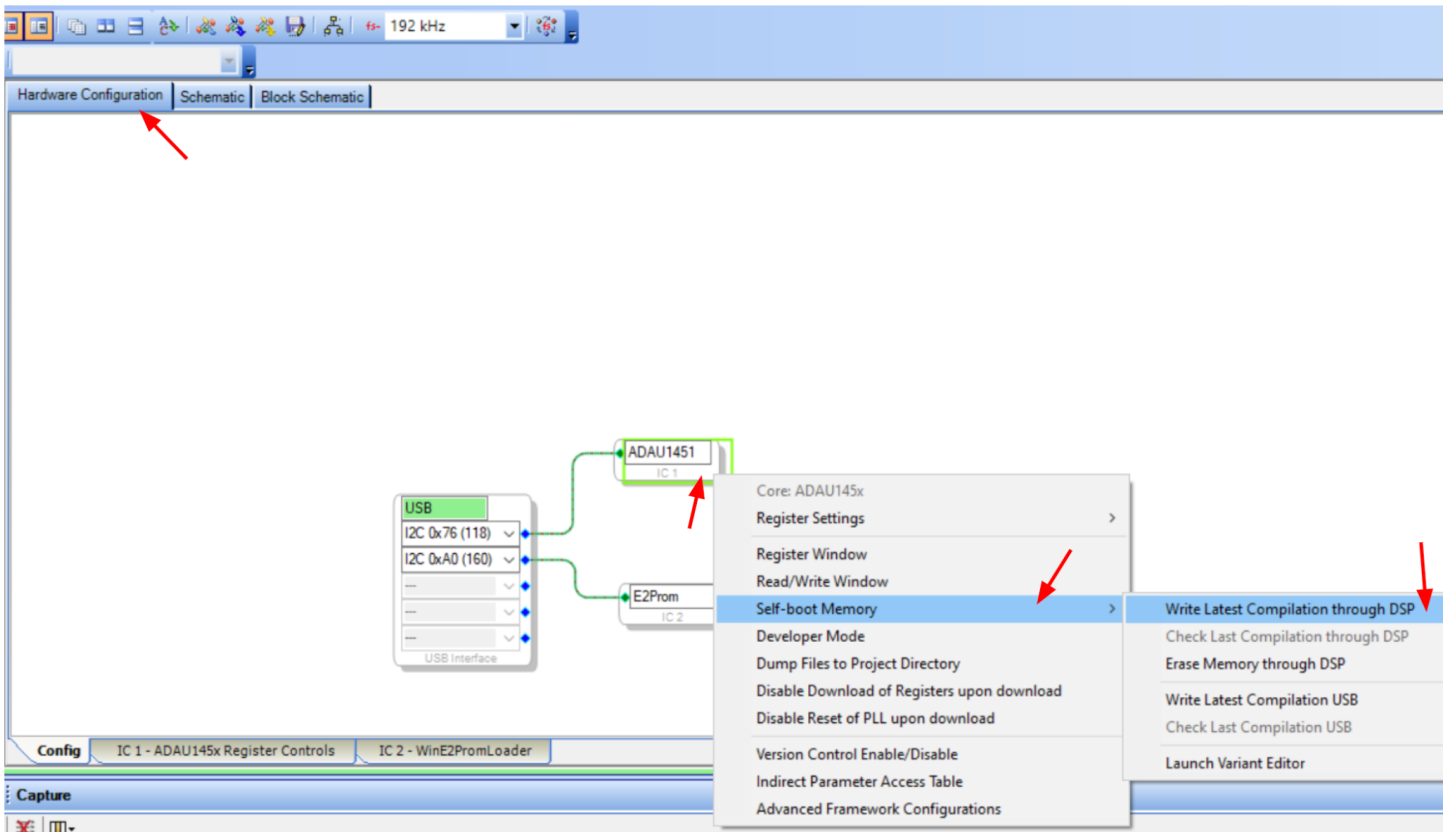
Step 3: Prior to flashing the DSP EEPROM, Sigma Studios needs to point to the configuration `20210628.xml` located in the `dj12-20210628` directory. On the Schematic tab, click the I2C/SPI CODEC Initialization module. Set the Sequence File to `20210628.xml` located in the project directory.



Step 4: After modifying the DSP file, but before flashing the EEPROM, you must load the program into DSP runtime memory. This also allows you to test your program effects in real-time. Select the `Link Compile Download` button on the menu bar.

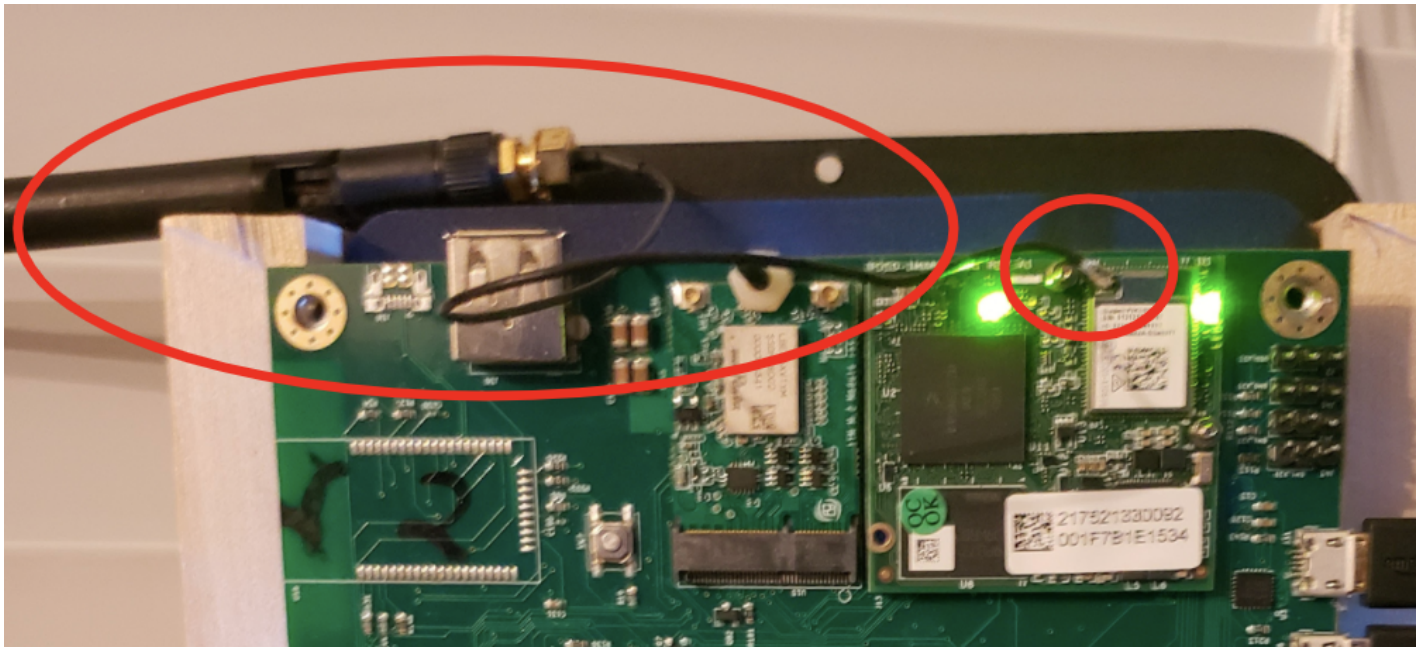


Step 5: To permanently write your changes to EEPROM, select the `Hardware Configuration` tab, right-click the ADAU1451 IC1 module, then select `Self-boot Memory` **->** `Write Latest Compilation through DSP`. After the write completes, you should be able to power cycle the board and the changes will persist.



[8] Connecting to WiFi

Connecting to WiFi is straightforward, however, it will not work at all if you don't have a physical antenna connected like in the picture below.



To connect to WiFi, after booting and connecting via serial, simply follow this guide here