

# DAC39RF1x API Examples

# A Few Notes

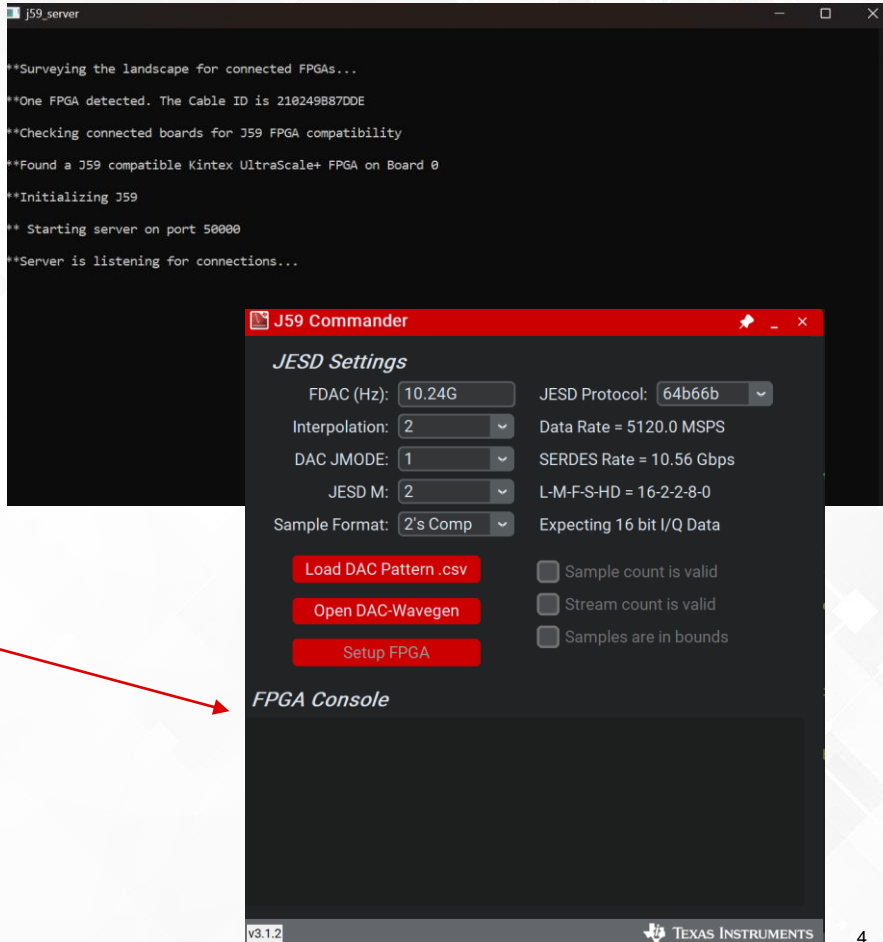
- These are instructions on how to use the example scripts provided with the API.
- The Example scripts are designed to completely bring up the DAC39RF10EVM or DAC39RF12EVM using the `dac39rf10_api`. In order to bring the DAC up, additional code is provided for things like setting up the clocks and handling SPI/I2C transactions.
- This API package comes with the following (All within **the <dac39rf10\_evm>** folder)
  - **<dac39rf10\_api>** The API for the DAC39RF10 and all related devices (DAC39RF12, DAC39RF10-EP, DAC39RFS10)
  - **<ftd2xx\_controller>** The package TI uses for SPI/I2C communication to our EVMs
  - **<lmk04828\_api>** An unofficial API for the LMK04828 (limited use cases used with our EVMs)
  - **<lmx1204\_api>** An unofficial API for the LMX1204 used on the DAC39RF10EVM
  - **<spi\_controller>** To assist with shadow registers
- The **<dac39rf1x.py>** file within the `dac39rf10_api` is a low level API with a header like file associated with it **<dac39rf1x\_constants.py>**. In addition to the **<dac39rf1x\_enums.py>** file, these three files are designed to be flat and easy to port over to a language like c.
- There is also a **<dac39rf1x\_wrapper.py>** file. This contains several helper and macro functions that do things like setup JESD links and help find valid combinations of JMODE, interpolation, DAC update Rates and stream count.

# Getting Things Setup

- Download the DAC39RF12EVM GUI. This will include the J59\_commander and J59\_server.exe.
  - This will also include instructions on how to setup Vivado.
  - This is optional but recommended to start.
- All necessary hardware connections for the DAC39RF10EVM or DAC39RF12EVM can be seen in the EVM user guides
- Make sure you have Python installed as the API is written in Python. (Python V3.11 or later is recommended)
- Its best to use a code editor like Visual Studio Code. The folder dac39rf1x\_top should be opened in the code editor.
- Within the dac39rf1x\_top directory there are four top level scripts.
  - **<dac\_top\_jesd\_helper.py>** This script will print valid JMODES to the screen based off inputs (similar to the way the GUI works). No hardware is required for this.
  - **<dac\_top\_dds.py>** This script sets the DAC up in DDS mode. No FPGA is required.
  - **<dac\_top\_real\_data.py>** This script sets the DAC up for real data modes (where no interpolation is needed).
  - **<dac\_top\_complex\_data.py>** This script sets the DAC up for IQ modes where digital up conversion is supported with the NCOs
- Control of the FPGA is done via. The J59\_server and J59\_commander.
- An example showing how to use the J59\_server and J59\_commander with the **<dac\_top\_complex\_data.py>** script.

# Getting a link up

- This script is setup to bring the DAC up at 10.24GSPS with 4x interpolation in JMODE2. This will show how to get this script to work with the J59\_commander and J59\_server
- Run j59\_server.exe.
  - If the FPGA is connected and powered up and Xilinx drivers are installed, this should be what j59\_server shows.
- Start j59\_commander.exe



# Getting a link up

- Run the <dac\_top\_complex\_data.py> script.
  - This should be the output of the script.
- Once the script has run, the clocks will be setup and the DAC will be waiting for the FPGA to setup the link.
- In the J59\_Commander, make it match the setup of the DAC.
- Press <Open DAC-Wavegen> in the J59 Commander

```
JESD Lane Rate = 10.56 Gbps  
JESD Lane Count = 16  
jesd L-M-F-S-HD = 16-4-2-4-0
```



Note the interpolation, JMODE and JESD\_M match the default state of the <dac\_top\_complex\_data.py> script

# Getting a link up

**DAC-Wavegen**

**Global Settings**

Sample Count: 122880 Signal Format: Complex  
DAC Input Rate: 2560.0M Channel Count: 2 IQ  
DAC Resolution: 16 RBW: 20833.33333333333 Hz

**Channel 1** Signal Type: Single Tone  
Frequency (Hz): 1.000354G Current Bin Number: 48017  
Level (dBFS): -1.0 Bin Adjust: - +  
Initial Phase (\*): 0.0 ☒ Prime Bins

**Channel 2** Signal Type: Chirp  
Center Freq (Hz): 0.0 Signal BW (Hz): 400.0M  
Level (dBFS): 0.0 Chirp Type: Linear  
Initial Phase (\*): 0.0 Period Count: 1

**Channel 3** Signal Type: Single Tone  
Frequency (Hz): Current Bin Number:  
Level (dBFS): Bin Adjust: - +  
Initial Phase (\*): ☐ Prime Bins

**Channel 4** Signal Type: Single Tone  
Frequency (Hz): Current Bin Number:  
Level (dBFS): Bin Adjust: - +  
Initial Phase (\*): ☐ Prime Bins

**Buttons:** Load Tones to FPGA, Generate Tones to .csv

1. Once all entries have been filled out on active channels, press <Load Tones to FPGA>

2. Now Tones are loaded into the J59 Commander. The three indicators should go green and Setup FPGA should be selectable.

**J59 Commander**

**JESD Settings**

FDAC (Hz): 10.24G JESD Protocol: 64b66b  
Interpolation: 4 Data Rate = 2560.0 MSPS  
DAC JMODE: 2 SERDES Rate = 10.56 Gbps  
JESD M: 4 L-M-F-S-HD = 16-4-2-4-0  
Sample Format: 2's Comp Expecting 16 bit I/Q Data

**Buttons:** Load DAC Pattern .csv, Open DAC-Wavegen, Setup FPGA

**Status Indicators:** ☒ Sample count is valid, ☒ Stream count is valid, ☒ Samples are in bounds

**FPGA Console**

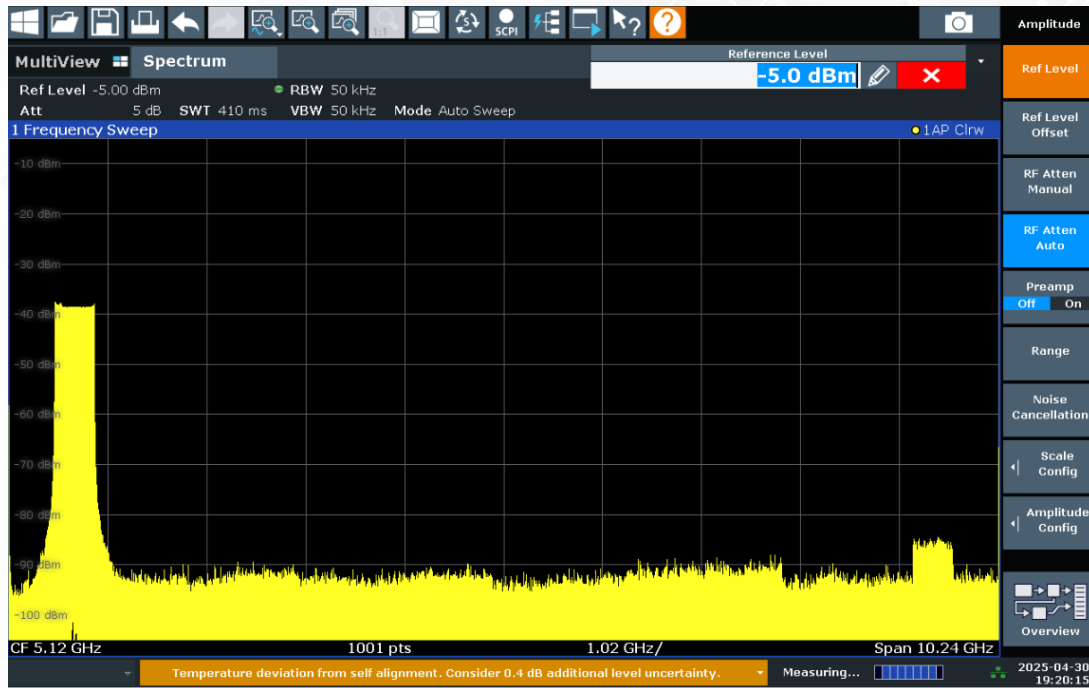
v3.1.2 TEXAS INSTRUMENTS

Example Tones Used. JESD M = 4 so 2 IQ streams required

# Getting a link up



The J59 Commander Console after a link is brought up



The output of channel B with the 400MHz wide chirp. The NCO was enabled and set to 684MHz