Evaluating the AD9084 Apollo MxFE Quad, 16-Bit 28GSPS RF DAC and Quad 12-Bit, 20GSPS RF ADC

FEATURES

- Fully functional evaluation board for the AD9084 direct sub8GHz and X-band transceiver
- PC software for control with **Analysis | Control | Evaluate (ACE) Software** or the Python Application (pyApp)

EVALUATION KIT CONTENTS

- AD9084-FMCA-EBZ evaluation board
- ADS10-V1EBZ data capture and transmit board
- Bootable microSD card

HARDWARE NEEDED

- Signal generators for the analog ADC inputs
- Band-pass filters for the analog ADC inputs
- A spectrum analyzer to measure the DAC outputs
- 2.92mm cables for clock inputs, ADC inputs, and DAC outputs
- A SMA cable for the FPGA REFCLK of the ADS10-V1EBZ
- Host PC running Windows 7 or later, equipped with a USB port (USB3.0 is preferred) and an Ethernet port
- 20GHz signal generator for off-board device clock generation (optional)
- <1GHz signal generator for off-board FPGA reference clock generation (optional)

SOFTWARE NEEDED

- **ACE Software**
- Python 3.8.6
- **VisualAnalog** to analyze captured ADC data when using the AD9084API directly (optional)

DOCUMENTS NEEDED

- AD9084 data sheet
- UG-2300 device user guide

GENERAL DESCRIPTION

This user guide describes the AD9084 evaluation platform and the necessary hardware and software configurations to evaluate the mixed-signal front-end RF transceiver (MxFE®).

The **ACE Software** allows the user to quickly evaluate the Apollo MxFE under different configurations, collectively known as use cases, to process and play an analog signal out of the digital-to-analog converter (DAC) or to capture a signal converted by the analog-to-digital converter (ADC). The signal samples are transferred across JESD204B or JESD204C links between the Apollo MxFE and the field-programmable gate array (FPGA) located on the ADS10-V1EBZ data capture and transmit board. The captured ADC samples can be downloaded to the host PC for display and analysis in the **ACE Software** or stored as vectors of the samples (a signal of a finite length stored as a list of samples) for processing with other software. The **ACE Software** also allows the user to generate common signal vectors or load custom vectors that can be uploaded to the ADS10-V1EBZ and then transmitted to the Apollo MxFE, processed by its digital signal processing (DSP), and reconstructed out of the DAC into a spectrum analyzer or other test equipment.

From a software perspective, the **ACE Software** interacts with the ADS10-V1EBZ across an remote procedure call (RPC) server hosted on the MicroZed mezzanine board mounted on the ADS10-V1EBZ. To control the AD9084 and other devices on the evaluation board, both ACE and the PyApp make requests to the RPC server. The server communicates with a precompiled version of the application programming interface (API) using a set of dynamic link libraries (DLLs) that are included with the software package. The MicroZed runs Ubuntu, a Linux distribution (distro), from a bootable microSD card supplied with the AD9084-FMCA-EBZ evaluation board.

Alternatively, users may download a specific version of the API to compile and execute the code directly on the MicroZed. To upload files and interact with the MicroZed, standard secure shell (SSH) clients, such as PuTTY or WinSCP, can be used. This approach allows modifying and compiling other versions of the API and executes the code directly on the MicroZed without using ACE, PyApp, or the RPC server interface. Signal vectors and captured ADC data is transferred between the ADS10-V1EBZ and the host PC across an Ethernet link and is stored as American Standard Code for Information Interchange (ASCII) files.

Various example C files are included with the Apollo MxFE API package to serve as a reference for developers to quickly configure the evaluation platform using the API or to port over any custom c-code from a prototype platform to compare the results. While the API specifically controls the Apollo MxFE installed on theAD9084-FMCA-EBZ evaluation board, the code included in the **Examples** folder also includes APIs for the ADS10-V1EBZ and other peripheral devices on the evaluation board, such as on-board clocking or other on-board devices that may be paired with the AD9084, to serve as a complete system reference design.

Note that certain Apollo MxFE features and DSP functionality may be initially available only in the API until support for these features is added to the Apollo MxFE PyApp and the **ACE Software**.

For full details on the AD9084, see the AD9084 data sheet, which must be consulted in conjunction with this user guide when using the AD9084-FMCA-EBZ.

TABLE OF CONTENTS

REVISION HISTORY

4/2025—Revision 0: Initial Version

EVALUATION BOARD PHOTOGRAPHS

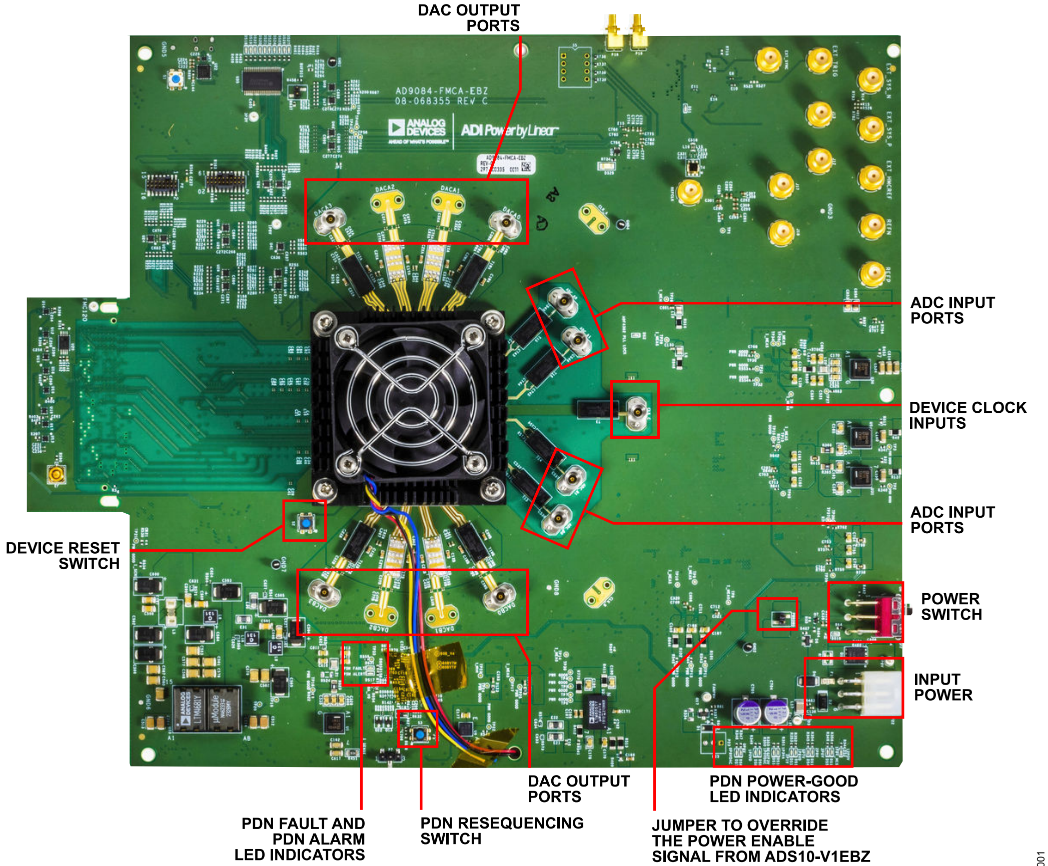


Figure 1. AD9084-FMCA-EBZ Evaluation Board, Top Image

EVALUATION BOARD PHOTOGRAPHS

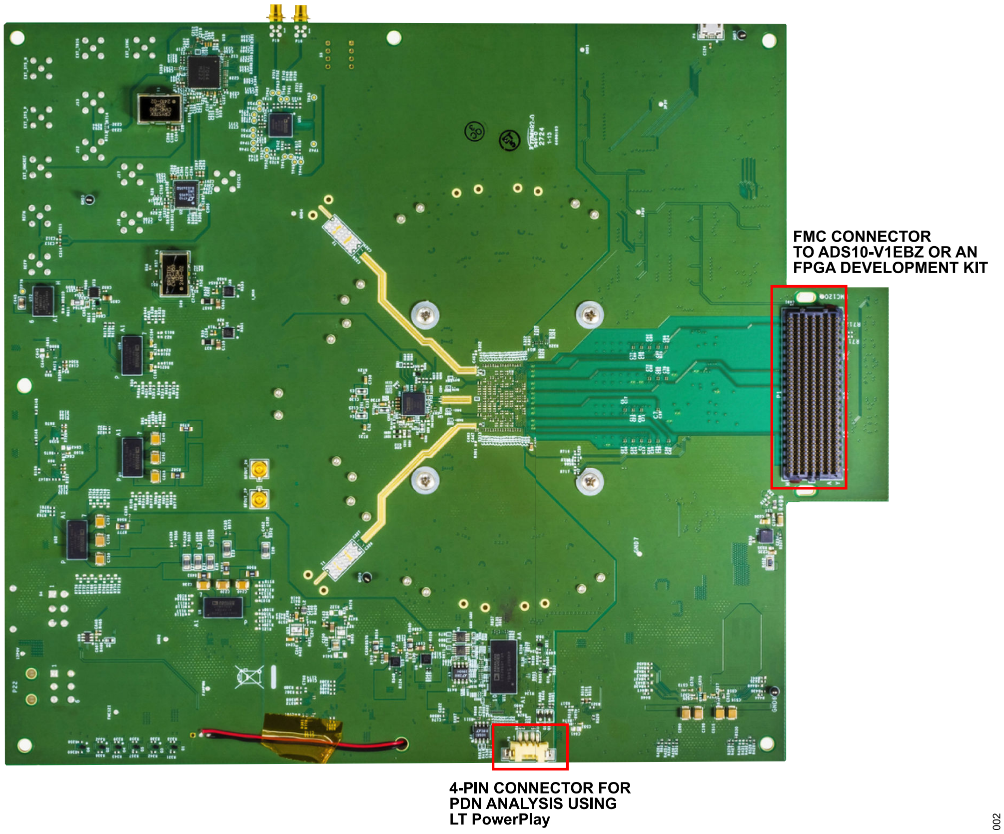


Figure 2. AD9084-FMCA-EBZ Evaluation Board, Bottom Image

EVALUATION BOARD OVERVIEW

Table 1. Evaluation Boards Available and/or Required

Evaluation Board	Description
Apollo MxFE Board	
AD9084-FMCA-EBZ	Apollo MxFE 4T4R differential evaluation board with on-board Marki baluns
FPGA Processor Board	
ADS10-V1EBZ	ADS10-V1EBZ data capture and transmit board

The standard Apollo MxFE evaluation board kit includes the AD9084-FMCA-EBZ and the ADS10-V1EBZ.

EVALUATION BOARD OVERVIEW

EVALUATION BOARD CONNECTION OVERVIEW

The hardware required to evaluate the AD9084, including the ADS10-V1EBZ and the AD9084-FMCA-EBZ evaluation boards, is shown in Figure 3.

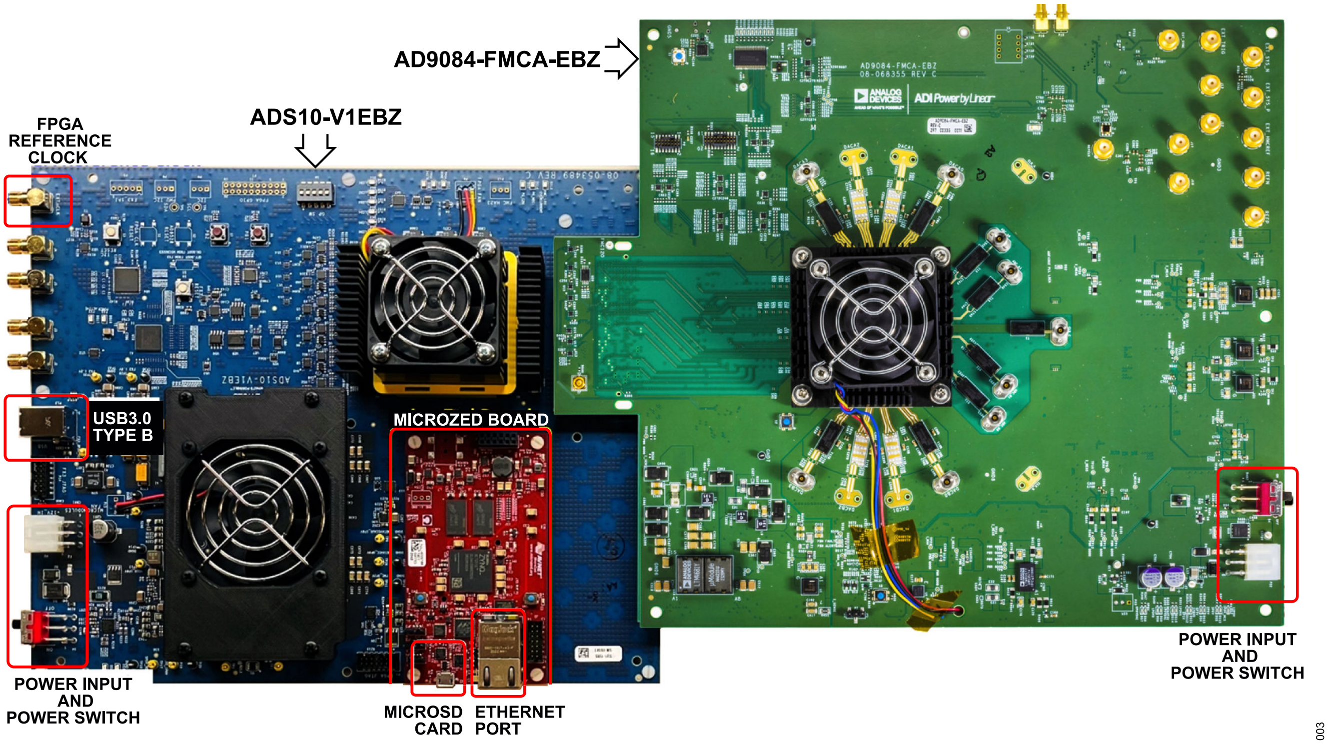


Figure 3. Bench Setup for Evaluating the AD9084

SETTING UP THE MICROZED

MicroZed is an embedded computer board that is installed as a mezzanine board on the ADS10-V1EBZ and that can initialize an operating system from bootable media, such as microSD cards. One of the supplied microSD cards, **Apollo v1.9** with the Apollo MxFE evaluation board contains a version of the Linux Ubuntu distro, a common operating system, onto which the user can upload C code and API examples to program the Apollo MxFE. Similarly, the **ACE Software** and other software from Analog Devices, Inc., can interface into the MicroZed board using a set of DLLs that communicate across an RPC server, locally hosted on the MicroZed board.

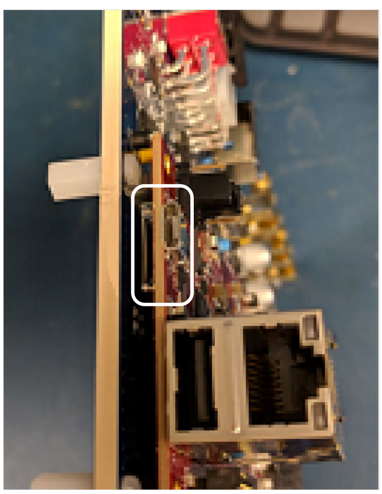
Jumpers on the MicroZed board are preconfigured to boot Linux from the microSD card.

Once the MicroZed board has finished booting, the host PC can establish an Ethernet link to the MicroZed board, access its memory and folder structure at the root level, and manipulate the various APIs to evaluate the Apollo MxFE along with its companion devices located on the AD9084-FMCA-EBZ evaluation board.

MICROSD CARD FOR THE MICROZED BOARD

To ensure proper connection between the microSD card and the MicroZed board, take the following steps:

1. Locate the **Apollo v1.9** microSD card included with the AD9084 evaluation board kit. It is important to note that the ADS10-V1EBZ ships with other microSD cards. These microSD are not related to the Apollo MxFE or its evaluation.
2. Insert the **Apollo v1.9** microSD card into the slot in the MicroZed board and ensure to face the contacts of the microSD card upwards. See Figure 4 for additional details.



004

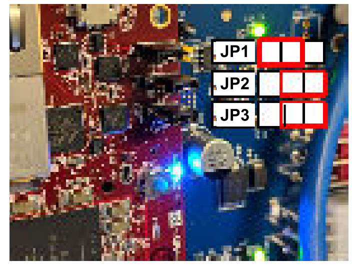
Figure 4. MicroSD Card Slot in MicroZed Board

As a precaution, ensure that the MicroZed board is seated properly on the ADS10-V1EBZ (see Figure 3). Note that only a visual inspection is needed.

BOOT INDICATOR JUMPERS

Ensure that the boot indicator jumpers on the MicroZed board are configured as shown in Figure 5 and as follows:

- Short Pin 2 to Pin 3 of **JP1**.
- Short Pin 1 to Pin 2 of **JP2**.
- Short Pin 1 to Pin 2 of **JP3**.



005

Figure 5. Boot Indicator Jumpers and their Connections on the MicroZed

The boot indicator jumpers (**JP1** to **JP3**), shown in Figure 5, instruct the MicroZed board to boot the image from the microSD card.

Once power is applied to the ADS10-V1EBZ, the MicroZed board automatically initiates the booting process. If the booting process is successful, the **D2** light-emitting diode (LED) illuminates blue. Note that the **D3** LED illuminates red if the MicroZed board cannot finish booting, and the **D5** LED Illuminates green if power is applied to the MicroZed board.

If the MicroZed board does not boot properly, check the boot indicator jumpers or try to reseat the microSD card while making sure that it clicks into place. As a last resort, consider reimaging the microSD card. Note that earlier versions of the microSD may not function with the latest API. Ensure that the latest microSD version (v1.9) is always used. Contact ADI Support to get more information on how to get the latest version of the microSD card image.

CONFIGURE THE NETWORK (ETHERNET) INTERFACE TO THE MICROZED BOARD

To configure the Ethernet controller on the host PC to communicate with the MicroZed board, take the following steps:

1. Ensure that the host PC is connected to the ADS10-V1EBZ using an Ethernet cable and that the MicroZed had finished booting. It is not necessary to connect the AD9084 evaluation board, but this board can remain connected.
2. Connect one end of the Ethernet cable directly to the PC Ethernet port or to a USB to Ethernet adapter and connect the other end of the Ethernet cable to the MicroZed board.
3. Power on the ADS10-V1EBZ board. Allow up to 10sec for the MicroZed board to boot up. Boot is complete when the D2 LED illuminates blue.
4. Open the local area connection settings. On Windows® 7, go to: **Start Menu**/**Control Panel**/**Network and Sharing Center**/

SETTING UP THE MICROZED

Change adapter settings. On Windows 10, go to **Start Menu/ Settings/Network & Internet/Change adapter options**.

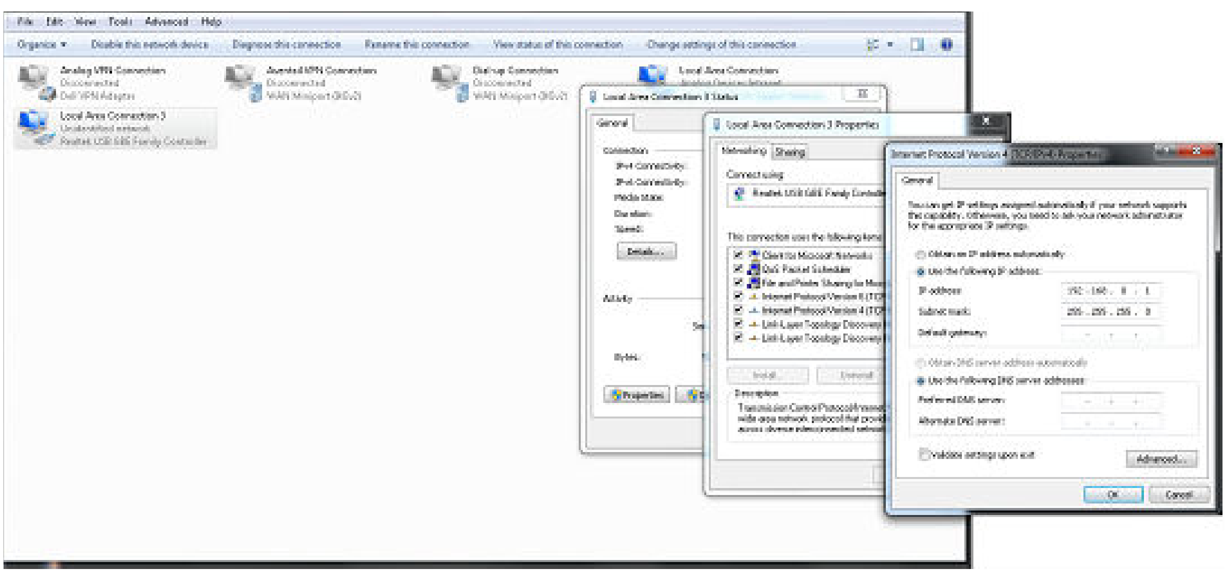
5. If the **Local Area Connection** icon does not appear in the **Network Connections** window, unplug the Ethernet connection from the MicroZed board and then reconnect it. In addition, ensure that the MicroZed board has completed the booting process.
6. Double-click the **Local Area Connection** icon that appears (Figure 6 shows **Local Area Connection 3** as an example).
7. Click **Properties**.
8. Select **Internet Protocol Version 4 (TCP/Ipv4)**.
9. Click **Properties**.
10. Enter **192.168.0.2** in the **IP address** field. You may also use 192.168.0.1; however, some network switches default to this address and may cause a contention.
11. Ensure that the **Subnet mask** field shows **255.255.255.0**.
12. Click **OK**.

CHECKING THE HOST PC CAN CONNECT TO THE MICROZED BOARD

To verify MicroZed board connection, establish a SSH connection to the MicroZed board from the host PC by taking the following steps:

1. Power on the ADS10-V1EBZ. Wait for the MicroZed board to boot up.
2. Open a Telnet software, such as WinSCP, PuTTY, or Tera Term. Windows 10 also has a simple SSH client integrated into the operating system.
3. Initiate an SSH session to **192.168.0.10**, which is also the IP address to the embedded RPC server on the MicroZed board.
4. Login with username: **root** and password: **analog**.
5. When the login is complete, the contents of the root folder display in the SSH terminal, which confirms that the host PC can communicate with the MicroZed board and the RPC server.

The user now navigates through the folder structure, manipulates and transfers files, compiles the API using the native Linux C Compiler (GCC), and uses any standard Linux commands available with root-level access. Take care when operating as a root user because changes in this bootable Linux media are persistent, and a reboot will not recover deleted files. The only way to recover the original Linux configuration and file structure is to reimage the microSD card with the original image from Analog Devices.



006

Figure 6. Internet Protocol Settings

EVALUATION BOARD HARDWARE SETUP

Typical laboratory setup to evaluate the AD9084 with a direct external clock input is shown in Figure 7. For a simplified setup, it is possible to loop the output from the DAC back into the ADC (that is, the external loop-back) and use the ADC to capture the signal generated by the DAC. In this case, the clock signals can be generated on-board, and, therefore, the signal generators and the spectrum analyzer are not necessary. However, the best device phase noise and dynamic range performance is realized when the AD9084 is evaluated using state-of-the-art laboratory equipment.

INSTRUMENTATION OVERVIEW

Figure 7 shows the hardware connections typically required when evaluating the AD9084 using the AD9084-FMCA-EBZ.

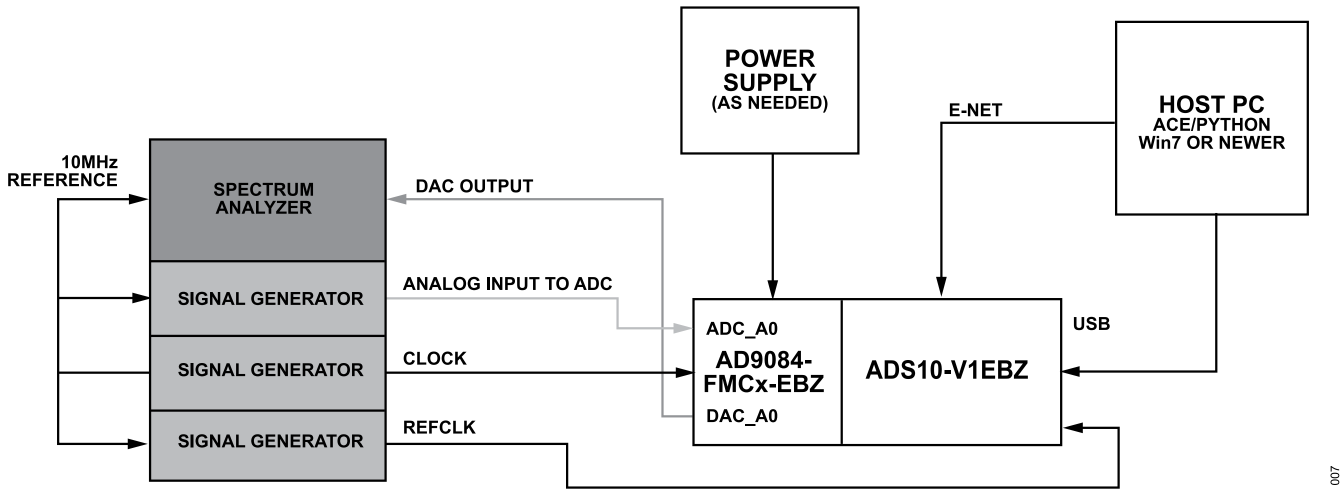


Figure 7. Instrumentation Setup for the Evaluation of the AD9084 Using An External Clock

EVALUATION BOARD HARDWARE SETUP

AD9084-FMCA-EBZ SETUP EXAMPLE

The ACE Software supports the AD9084-FMCA-EBZ. The AD9084-FMCA-EBZ uses wideband baluns to convert between the differential RF input and output ports of the Apollo MxFE (for example, one of the DAC output ports) and a single-ended 2.92mm connector on the board.

The **ACE Software** Apollo plug-in includes multiple, preset use cases. The sources for the clocks that the Apollo MxFE and the FPGA require are selectable to either use the on-board clock generator chips (ADF4382 and HMC7044) or to provide a clock from a source external to the board. When configured to receive an off-board clock, the on-board clock tree is disabled, and the clock is cabled from the source to the **CLK_C** 2.92mm connector. If using the on-board clock tree, no external clock source is needed. Other configurations are also available and detailed in the Clocking Schemes (Configurations) section.

Use Case 1 details follow:

- Direct clocking to the **CLK_C** input at 20GHz with 13dBm to 14dBm power at the connector.
  - A high-quality 2.92mm cable with lower insertion loss preferred.
- A FPGA REFCLK: 156.25MHz and 4dBm. It must be synchronized to the 20GHz clock source (typically using a common 10MHz reference).
- DACs setup with an interpolation of 16 (coarse digital upconversion (CDUC) × fine digital upconversion (FDUC) = 4 × 4).
  - A coarse numerically controlled oscillator (CNCO) set to 1.1GHz.
- ADCs setup with a decimation of 16 (coarse digital downconversion (CDDC) × fine digital downconversion (FDDC) = 4 × 4).
  - A CNCO set to 2.5GHz and a fine numerically controlled oscillator (FNCO) set to 5MHz.
- An ADC and DAC sampling rate of 20GSPS and 20GSPS, respectively
- A data rate of 1.25GSPS in-phase quadrature (I/Q) or 1GHz instantaneous bandwidth (iBW).
- The line rate is 10.3125Gbps per lane.
- In JESD204C mode, with the following setup per bank:
  - L = 8 (number of lanes)
  - M = 4 (number of converters)
  - F = 1 (octets per frame)
  - S = 1 (samples transmitted per single converter per frame cycle)
  - N' = 16 (total number of bits per sample)

Additional setups, including Use Case 1, shown in Table 2, are configured similarly, and the previous details can be inferred from the **ACE Software** GUI.

Table 2. Predefined Use Cases in the ACE Software for the AD9084-FMCA-EBZ Evaluation Board

Use Case Index Number	ADC Rate (GSPS)	DAC Rate (GSPS)	Coarse DEC in CDDC[1]	Fine DEC in FDDC	ADC Baseband Rate (GSPS)	Rx iBW (GHz)	Coarse INT[2] in CDUC	Fine INT in FDUC	DAC Baseband Rate (GSPS)	Tx iBW (GHz)	JESD204B or JESD204C	No. of L per Bank	Line Rate (Gbps)	Notes
1	20	20	4	4	1.25	1	4	4	1.25	1	C	8	10.3	
2	20	20	4	2	2.5	2	4	2	2.5	2	C	8	20.6	
3	10	10	4	2	1.25	1	4	2	1.25	1	C	8	10.3	
4	14	28	4	1	3.5	2.8	4	2	3.5	2.8	C	12	14.4	Not fully verified
5	14	28	4	2	1.75	1.4	8	2	1.75	1.4	C	8	14.4	Sequential ADC interleaving only

[1] DEC means decimation.

[2] INT means interpolation.

EVALUATION BOARD HARDWARE SETUP

SETTING UP OF THE INSTRUMENTATION

The following are the recommended instruments and connectors for the setup shown in Figure 7:

1. The AD9084-FMCA-EBZ, except for AD9084-FMCA-EBZ Re-vA, uses a 12V brick supply provided with the AD9084-FMCA-EBZ package.
2. An external direct clock is required for the Apollo MxFE. A low phase noise signal generator, such as the Rohde & Schwarz SMA100B or Keysight E8257D series, is recommended. Note that the performance listed in the AD9084 data sheet is from the Rohde & Schwarz SMA100B with the B-711 option.
    a. Set the device input clock to the desired frequency. The amplitude must be set to 15dBm, at the signal generator, excluding cable loss, so that the input clock has an optimal slew rate at the Apollo MxFE pins.
3. A signal generator is required at the ADC input. Set the frequency to 2.65GHz and the amplitude to ~3dBm, or, if using an in-line band-pass filter, to ~5dBm.
    a. Typically, a band-pass filter is recommended for the analog signal input to the ADC to filter out the harmonics generated by the signal source.
4. An FPGA reference clock is required. A third signal generator must be set at 156.25MHz, 4dBm for the REFCLK for the FPGA board (ADS10-V1EBZ).
    a. An alternative option is an auxiliary output, such as the Rohde & Schwarz SMA100B, on the back panel or front panel.
    b. This clock source must be locked to the same frequency reference as the device clock source.
5. A spectrum analyzer, such as the Keysight PXA/UXA or Rohde & Schwarz FSW/FSWP, is required to measure the DAC outputs.
6. Use a shielded 50Ω coaxial cable, such as the RG-58, with 2.92mm connectors. Subminiature Version A (SMA) cables can also mate to the 2.92mm connectors on the board: however, the SMA connectors are manufactured to a lower mechanical tolerance and may damage the 2.92mm connector permanently. Pay careful attention to the insertion loss of the cables. Typical 2 foot test cables are expected to be better than 2dB to 3dB at 20GHz.

EVALUATION BOARD HARDWARE SETUP

CLOCKING SCHEMES (CONFIGURATIONS)

At present, the default clock scheme for the AD9084-FMCA-EBZ is all on-board clocking where all the required clock signals are generated by components on the board. The reference on-board is generated by a 125MHz crystal; however, any of the on-board phase-locked loops (PLLs) can also receive references from external sources to allow more flexibility when selecting clock rates that are not multiples of the 125MHz on-board reference. Otherwise, the clocks can be provided externally through 2.92mm or SMA connectors. A detailed block diagram of the available schemes is shown in Figure 8.

Regardless of the clocking scheme, all clock sources must be generated from a common reference, whether external or on-board. For example, if using the ADF4382 to clock the Apollo MxFE **CLKC**, the HMC7044, which generates the other clocks (**SYSREF** and ADS10-V1EBZ **REFCLK**), must receive a reference that is derived from a common time base as the reference to the ADF4382. If an external source is used to drive **CLKC**, the sources for **SYSREF** and ADS10-V1EBZ **REFCLK** must be synchronized with the **CLKC** signal source.

The DAC sample rate can either be twice the ADC sample rate or equal to it, depending on the use case. In all cases, the clock frequency for the **CLKC** port of the Apollo MxFE must match the ADC sample clock rate.

To switch between schemes, the correct clock path (or paths) must be selected by soldering capacitors or resistors, and the on-board clocking ICs must be configured through software, using the API, PyApp, or ACE Software. For example, to change from all on-board clocking to all external clocking, the following changes must be made:

- Hardware changes, such as the following:
  - Around the ADF4382, as follows:
    - Depopulate C323/C325 to disconnected the ADF4382 path.
    - Populate C324/C330 to connect the **CLK_C** path.
  - Around VCX0, as follows:
    - Consider removing R51 to power down the on-board 125MHz reference crystal.
- Software changes. Disable the ADF4382 and HMC7044 as described in the Software Configuration of the ADF4382 and HMC7044 section.

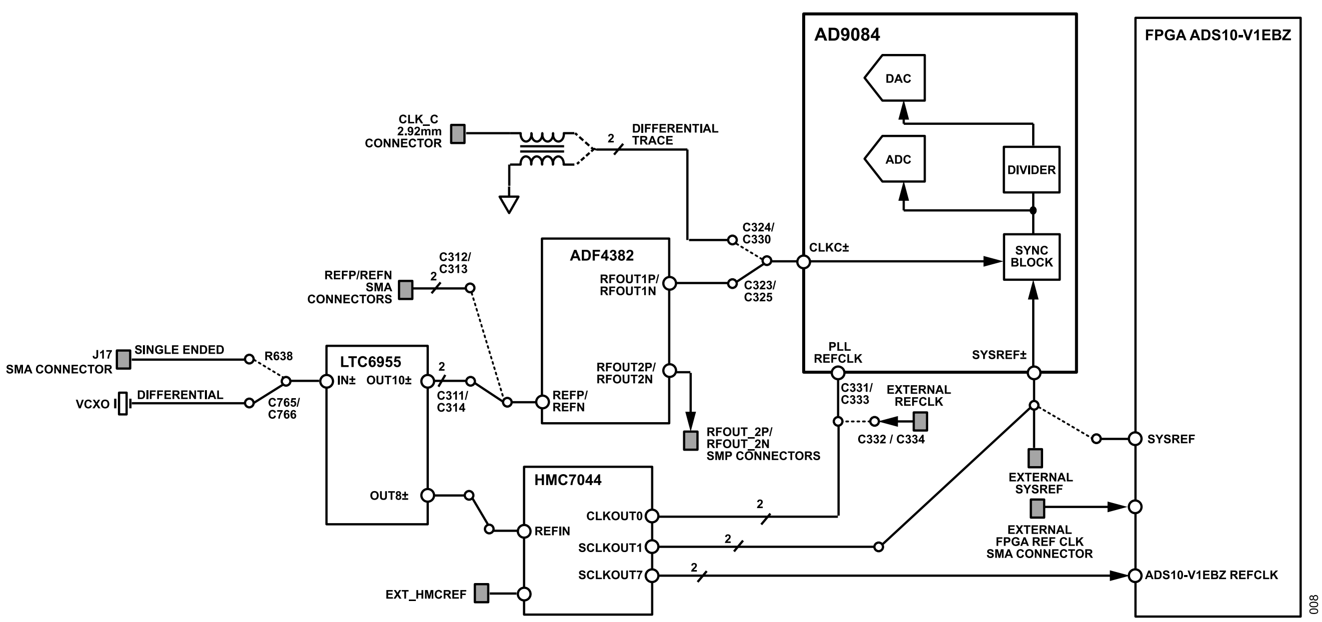


Figure 8. Clocking Scheme in the AD9084-FMCA-EBZ (Revision C)

EVALUATION BOARD HARDWARE SETUP

ADF4382 Loop Filter Optimization

By default, the loop filter of the ADF4382 is configured for a 250MHz phase and frequency detector (PFD) frequency. For optimal performance at 500MHz, use the component values and filter topology specified in Table 3.

Table 3. Loop Filter Values for Setting Up the ADF4382 with a 500MHz Reference Input Through J17

Reference Designator	Value	Part Number
R291 and R315	0Ω	ERJ-2GE0R00X or equivalent
C252 and C241	100pF	06035A101JAT2A or equivalent (choose C0G/NP0 type)
R13	110Ω	ERJ-2GEJ111X or equivalent
R289	250Ω	RT0603BRE07250RL or equivalent
C243	10.0nF	C2012C0G1H103J060AA or equivalent
R287 and R286	0Ω	YJP1608-R001 or equivalent
C242	Depopulate	Not applicable

EVALUATION BOARD HARDWARE SETUP

Software Configuration of the ADF4382 and HMC7044

API

The API examples run from the **~/src/examples/ads10_apollo_ex_main** directory on the MicroZed microSD card.

The API example run command includes the option to specify the clock source for the AD9084-FMCA-EBZ and the ADS10-V1EBZ FPGA reference clock. The general form of the example run command is as follows:

./debug/apollo_main EXAMPLE_TYPE DEVICE_PROFILE [-devclk=DEV_CLK_SRC] [-fpgaclk=FPGA_CLK_SRC]

If the evaluation board is externally clocked and the ADS10-V1EBZ reference clock is provided by an external source, the clocks do not need to be specified when running an API example. The API assumes external clock by default.

To run the full chip example with the device profile for **id00_uc06_F**, the command is as follows:

./debug/apollo_main fullchip id00_uc06_F

Note that no clocking arguments required with external clocking.

If the evaluation board has on-board clocking for the Apollo device clock (using the ADF4382) and for the ADS10-V1EBZ FPGA reference clock (through the FMC+ connector from the HMC7044), clocking parameters are passed on the command line.

To run the full chip example with the device profile for **id00_uc06_F**, the command is as follows:

./debug/apollo_main fullchip id00_uc06_F -devclk=adf4382 -fpgaclk=fmc

See the API Example: Ads10_apollo_ex_main Full Chip section for more information on running an API example.

ACE Software

From the **ACE Software** GUI either the ADF4382 or the ADF4382 and HMC7044 can be selected as the on-board clock source or sources. The device or devices is or will be configured according to the clock rates required by the use case loaded onto the **ACE Software**.

PyApp

See the Python Application (PyApp) section for information on the Python setup.

EVALUATION BOARD SOFTWARE SETUP

The following different methods are available to configure the AD9084-FMCA-EBZ by using software, where each method is designed to be a standalone and does not have a dependency on the other:

1. The ACE Software is a graphical user interface (GUI) that utilizes a precompiled API object, which is a convenient method for users who prefer not to write code to configure an evaluation board. Note that not all Apollo MxFE features may be supported, and that the API may be from an earlier version. The **ACE Software** may include applets and other software that can be useful to use together with the other configuration methods, such as the API.
2. The PyApp is a python wrapper that utilizes a precompiled API object, which is a convenient method for users who want to automate their evaluation process and have direct access to all the features available in the precompiled API object. All Apollo MxFE features are exposed; however, the API may be from an earlier version.
3. The API and the API examples give users direct access to the latest released API in C, with usage examples of the major features within the Apollo MxFE. All Apollo MxFE features are exposed as these features become available in the latest public API release.

The evaluation platform is primarily controlled (programmed) using the **ACE Software** installed on a host PC. Other methods exist, such as direct manipulation using an API version loaded by the user onto the MicroZed board or by scripting from the host PC using common programming languages, such as Python and MATHLAB.

To quickly begin the evaluation of the Apollo MxFE, the only software required is the **ACE Software**.

For a more advanced user, such as a software developer interested in directly interfacing with the API, a simple SSH client may suffice, such as PuTTY or WinSCP. WinSCP is a good method for users accustomed to the Windows environment because it integrates a simple GUI to visualize the folder structure on the MicroZed board, along with the ability to initiate terminal sessions via PuTTY from within WinSCP. Both PuTTY and WinSCP are available for download online.

The Linux Ubuntu distro on the MicroZed board includes the GCC compiler, as well as a simple C debugger (GNU Project Debugger, GDB).

Note that Analog Devices distributes the Apollo MxFE evaluation software and its associated API through a secure software distribution (SSD) system. Users must create a **myAnalog** account on https://my.analog.com/ to request to be added to the distribution list for the latest ACE Software, PyApp, or API and must submit a software request form at https://form.analog.com/form_pages/software-modules/SRF.aspx. The download link becomes available through your **myAnalog** account.

ACE SOFTWARE

The **ACE Software** is a GUI that controls a precompiled API object through a server link, established with the microZed board. The API object and the server are updated to the latest supported version every time the **ACE Software** is restarted.

Generally, the **ACE Software** uses an earlier API version, typically a few versions behind the latest public API released.

Note that the A version silicon (for example, A0 or A2) is not supported in the current and future software releases.

Installing the ACE Software and Apollo Plug-In

The **Apollo_Installer** plug-in within the ACE Software requires a web connection to proceed with its installation. Note that the web connection is required if the **ACE Software** is being installed for the first time. If the installer detects that the correct ACE version is already installed on the host PC, it skips downloading the **ACE Software** and proceeds with the plug-in installation.

Proceed with the default selections to complete the installation.

In some cases, when the **ACE Sofware** encounters run-time errors on a particular host PC, a full reinstallation can be helpful as follows:

1. Scrub the uninstall **ACE Software** if it is already installed on the PC.
   a. Select to uninstall all the SDP drivers and LRF drivers.
   b. Select to uninstall the fx3 drivers.
2. Run the **Apollo_Installer.exe** file as an administrator. Allow the installation to complete.
   a. As part of the installation process, a **Choose Components** window appears that contains an **Apollo ACE Sandbox** check box. Confirm this check box is checked off before proceeding with the installation.
   b. Note that when the **Apollo ACE Sandbox** check box is checked three instances of the **ACE Software** install:
      1. **ACE**
      2. **ACE (Apollo)**
      3. **ACE (x86)**
   c. To use the **ACE Software** with the Apollo MxFE, use the **ACE (Apollo)** instance. Use one of the other two instances for all other **ACE Software** plug-ins from Analog Devices, Inc.
3. At this point, there is a known conflict if VisualAnalog is being used. **VisualAnalog** must be installed before the **Apollo_Installer.exe** file is run due to a conflict with system demonstration platform (SDP) driver version issues. If **VisualAnalog** was installed after running the **Apollo_Installer.exe** file, the Apollo_Installer.exe must be reinstalled to overwrite the SDP drivers with the latest versions.

EVALUATION BOARD SOFTWARE SETUP

The **ACE Software** Apollo installer bundles the **ACE Software** core with the latest Apollo MxFE plug-in. However, the **ACE Software** core may prompt the user to update to a newer ACE version. If this happens, cancel out of this prompt and uncheck the automatic update option. Do not update the **ACE Software** to a different version from what was included in the **ACE Software** Apollo installer.

The **ACE Software** automatically configures the Ethernet settings and IPC server settings. Figure 9 and Figure 10 show the correct configurations. These views are accessible by clicking the **Settings** link in the lower left corner of the **ACE Software** GUI.

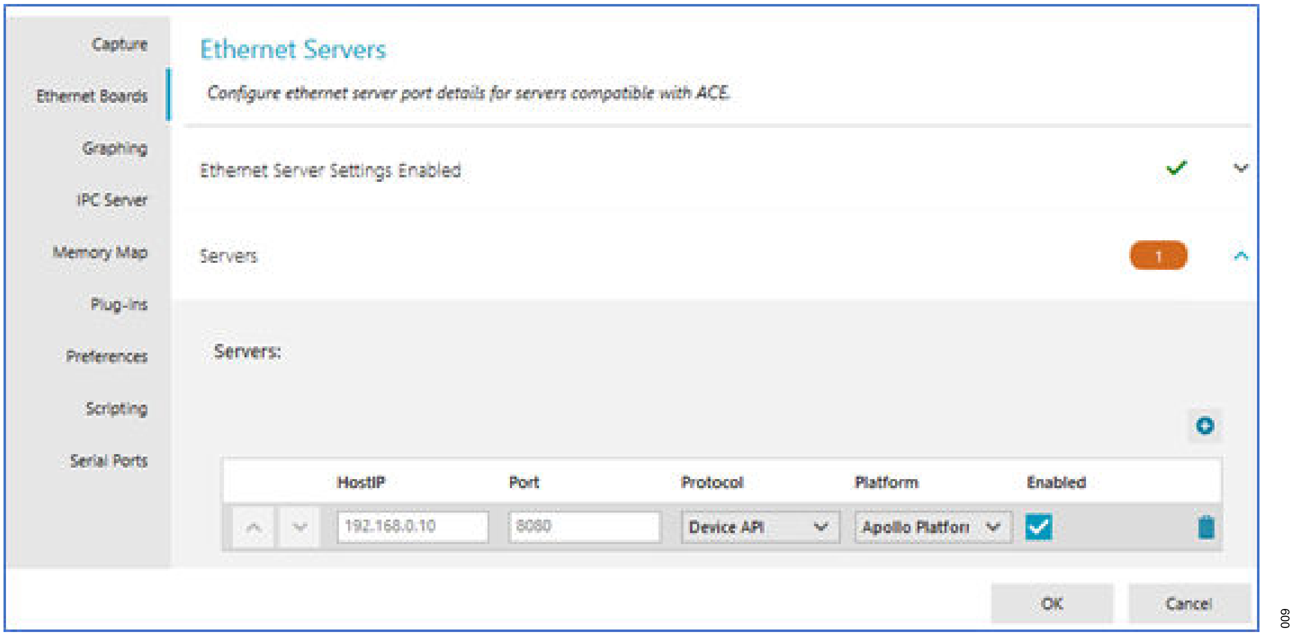


Figure 9. Ethernet Server Settings in ACE

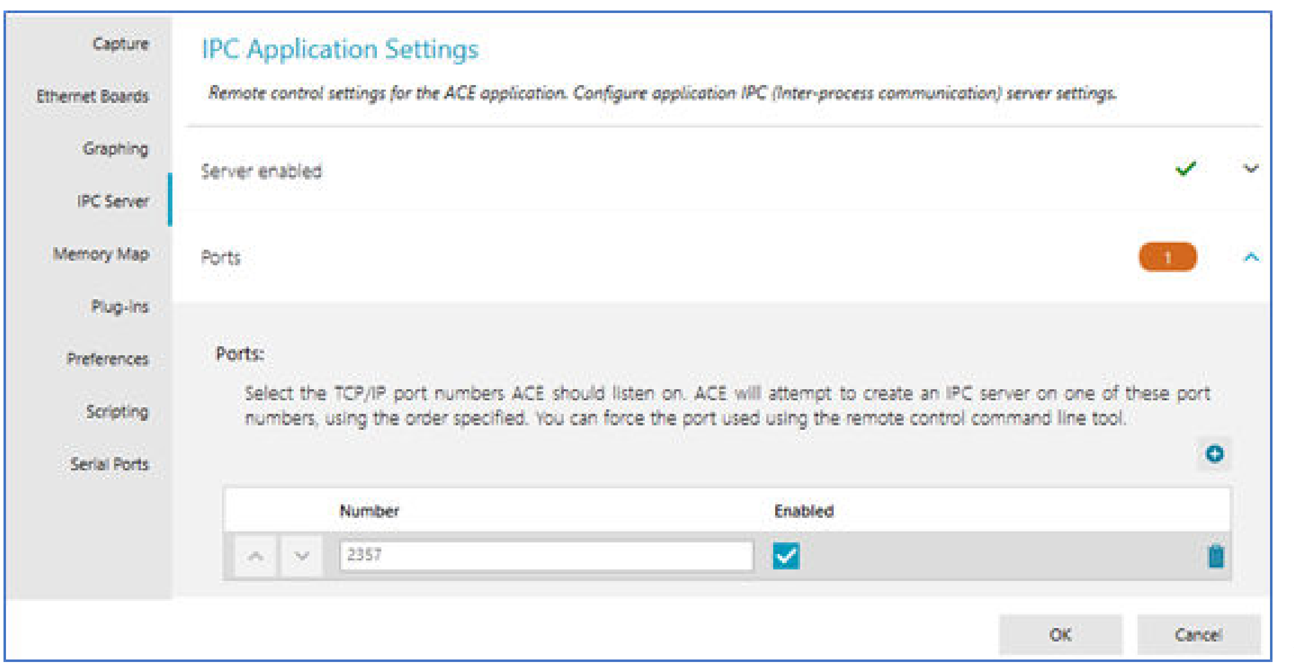


Figure 10. IPC Server Settings in ACE

EVALUATION BOARD SOFTWARE SETUP

Useful Hints When Debugging and Using the ACE Software

The **ACE Software** is constantly evolving to improve the experience of its users. Currently, a number of useful features exist that can simplify the evaluation of the AD9084 when using the **ACE Software**, which include the following:

- Be patient. There are multiple software layers that must operate together in order for the evaluation platform to function properly. For example, Ethernet links may take time to establish; the RPC server may be slow to boot due to other activity on the MicroZed board. If the **ACE Software** appears to **not respond** according to Windows, it does not mean it is frozen and needs to be restarted. Be patient and allow it to finish executing the command you previously requested to execute.
- Enable and always use the **Events** tab to get indications of what the **ACE Software** is currently doing, receive error messages, and so on (see Figure 11).

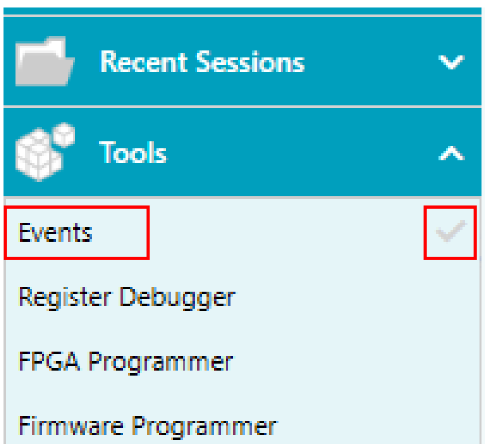


Figure 11. ACE Events Tool

- If the connection to the ADS10-V1EBZ is lost, it can be reset by disabling both the Ethernet boards server (that is, the RPC server) and the IPC server, and then re-enabling the RPC and IPC servers. The settings are accessed as shown in Figure 9 and Figure 10.
  - The hardware must be reconnected in the **ACE Software** within the **Connect Hardware** drop-down menu (see Figure 12). Do not use the **AD9084 Board (Local Only)** setting because it is in hardware emulation mode where the **ACE Software** does not communicate with the actual hardware. Use the **AD9084 Board** setting instead.

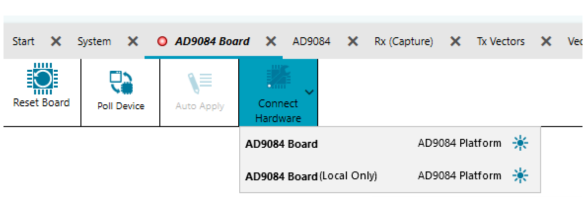


Figure 12. ACE Connect Hardware

- The **ACE Log** feature logs most of its activity in an ACE log, which is accessible by clicking on the folder icon in the top right corner of the **ACE Software** window (see Figure 13).

Figure 13. Folder Icon

  - However, this log only logs RPC commands and not the actual API calls.
  - Note that the error handling mechanism in the **ACE Software** does not have full awareness of the error handling and error messages generated by the API because there is a separate API log for that.
- The **API Log** feature is a useful tool for checking on the device status when debugging a set-up issue. It is accessible within the **ACE Software** through the Apollo MxFE plug-in. To download the **API Log**, click the **API Log File** button in **Tool Box** are of the **Dashboard** view (see Figure 23).
  - The log file is stored on the MicroZed board and can also be accessed manually without using the **ACE Software**.
  - Each click to download the **API Log FIle**, downloads a copy of the running API log file from the MicroZed board. A new file is created only when the MicroZed board is rebooted.
- The **ACE Macro** feature in the ACE Software can record all its activity as a macro that can then be re-executed.
- The **ACE and SPI Sequences** feature records all serial port interface (SPI) transactions to a particular device as a sequence that can then be executed by a memory device, such as an field-programmable gate array (FPGA) and a complex-programmable logic device (CPLD), without using API calls. The ACE Macro previously had awareness of the specific SPI transactions sent to the high-speed converter products. With the introduction of the API, the **ACE Software** no longer has awareness of the SPI commands executed by the API. Instead, the **ACE Software** sends requests to the RPC server on the MicroZed board that then execute one or more API commands. Therefore, if it is necessary to record the individual SPI transaction sent to the Apollo MxFE or another peripheral device on the evaluation board, it is optimal to use the API.

EVALUATION BOARD SOFTWARE SETUP

Programming the AD9084-FMCA-EBZ

The following steps explain how to setup the AD9084-FMCA-EBZ evaluation board within the **ACE Software** based on the setup explained in the Evaluation Board Software Setup section:.

1. Connect the AD9084 evaluation board and instrumentation as shown in Figure 3 and Figure 7.

    a. Ensure that the FMC+ connector on the AD9084-FMCA-EBZ is aligned with the FMC+ connector on the ADS10-V1EBZ before pressing these boards together. The FMC+ connectors cannot be replaced if a pin is damaged. Proper alignment and careful insertion avoids such problems.

    b. An FMC+ connector extender and saver (Samtec REF-212564-01) is a possible way to reduce the risk of pin damage to the FMC+ connector on the ADS10-V1EBZ. Mounting this on the ADS10-V1EBZ exposes the extender and saver to repeated insertions; therefore, protecting the connector on the ADS10-V1EBZ. If a pin is damaged on the extender and saver, the whole extender and saver can be easily replaced. Using this extender and saver may affect signal integrity; therefore, the user must evaluate the suitability of this alternative for the user application.

    c. FMC standoffs and jack screws (for example, Samtec ASP-199167-01 or Samtec ASP-199167-04) can be used to assist in the safe removal and insertion of the board FMC+ connectors.

    d. The options previously listed are not included with the evaluation kit and are the responsibility of the user.

2. Power on the ADS10-V1EBZ, and allow the MicroZed board to boot up.

    a. Wait until the **PG_C2M** and **PG_M2C** light emitting diodes (LEDs) are lit on the ADS10-V1EBZ data capture and transmit board.

3. Power on the AD9084-FMCA-EBZ evaluation board. A power supply that draws around 1.0A indicates it is a healthy board.

4. Ensure that the clock (if using external off-board clock sources) and analog signals to the AD9084-FMCA-EBZ and ADS10-V1EBZ are powered on and are set to the correct frequencies, if known in advance. See the Setting Up of the Instrumentation section for the signal power specifications.

    a. The device clock frequency and FPGA reference clock frequency are use case specific and are specified in the **ACE DEVICE CONFIG** window (see Figure 19). Frequencies can be adjusted after a use case is selected.

5. Click **ACE (Apollo)** to start the **ACE Software** when it appears in the Windows Analog Devices **Start** menu. If **ACE (Apollo)** does not appear, click **ACE** (see Figure 14).

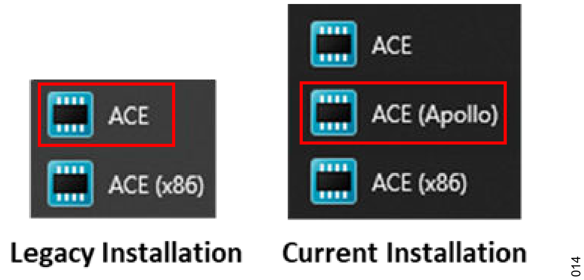


Figure 14. Starting the ACE Software

6. After several seconds, possibly 10 seconds or so, the RPC server completes initialization, and the plug-in appears in the **Attached Hardware** section within the **ACE Software**. Open the **AD9084 Board** plug-in from within the **Attached Hardware**" section (see Figure 16).

    a. The **ACE Software** does not detect hardware with the AD9084-FMCA-EBZ evaluation board powered down. This software loops until the evaluation board powers on, at which time, the hardware is detected, and the plug-in icon appears after several seconds.

    b. The **Events** tab can show useful status messages during initialization, which also indicate whether the RPC server is being initialized

    c. If within the **Attached Hardware** section an **Essential Plug-in Updates Available**" message appears, users should disregard this message (see Figure 15).

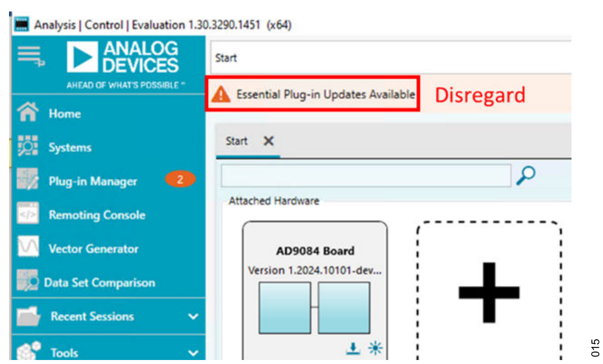


Figure 15. Essential Plug-in Updates Available Message

EVALUATION BOARD SOFTWARE SETUP

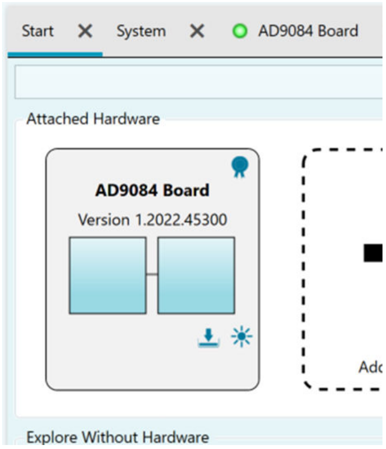


Figure 16. Opening AD9084 Board Plug-in in the ACE Software

7. The **ACE Software** version that comes with this revision of this user guide will prompt users to load the latest raw FPGA image to the microSD card in the microZed board on the ADS10-V1EBZ. If the user clicks **Yes** to load the new FPGA image, this prompt will no longer appear in subsequent runs of the **ACE Software** if the same microSD card is used. If the user clicks **No**, the **ACE Software** proceeds with using the existing FPGA image, and the **FPGA Update** prompt appears the next time the **ACE Software** starts. When this results, click **Yes** (see Figure 17).

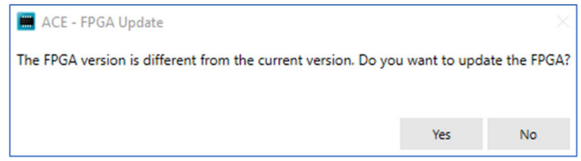


Figure 17. FPGA Update Prompt

After the new FPGA image is loaded, the **FPGA Update Done** window appears. Then, click **OK** (see Figure 18).

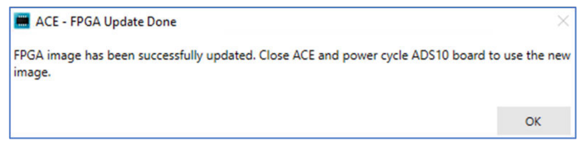


Figure 18. FPGA Update Done

8. Double-click on the **AD9084 Board** plug-in icon in Figure 16 opens the **AD9084 Configuration and Profile Generation** window in the **ACE Software** (see Figure 19).

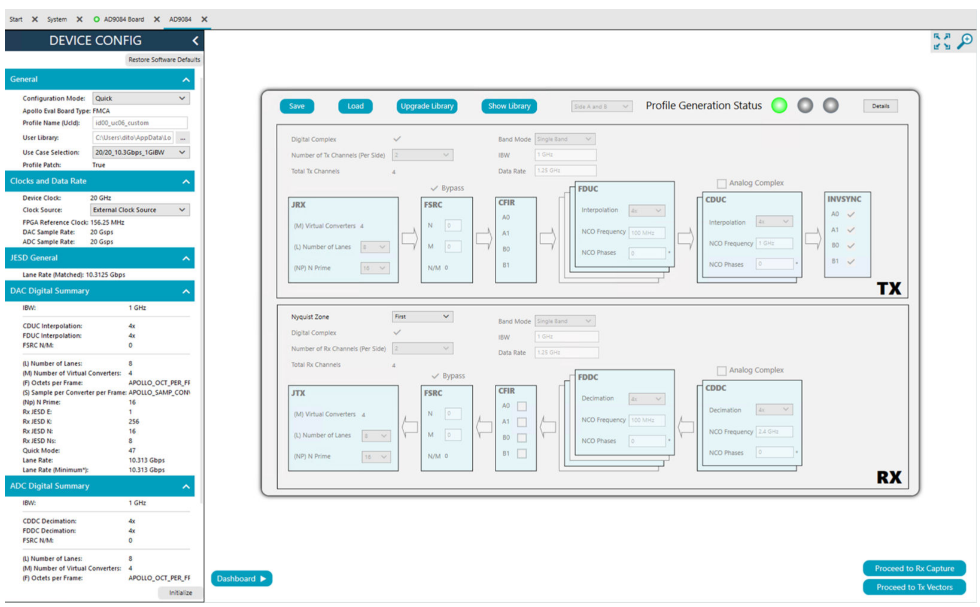


Figure 19. AD9084 Configuration and Profile Generation Window

EVALUATION BOARD SOFTWARE SETUP

ACE Software Device Profile Generator

The **ACE Software** plug-in **DEVICE CONFIG Profile Generation** user interface for the three available modes is shown in Figure 20, Figure 21, and Figure 22. A device profile contains the parameters and configuration details of an AD9084 use case. The **ACE Software** uses the device profile information to configure the device under test (DUT). The **ACE Software DEVICE CONFIG Profile Generator** can save the device profile as a file (**.blueprint**, **.h**, or **.json**) that can be saved, shared, and loaded back into the **ACE Software** (**<filename>.json** or **<filename>.blueprint**) or used by the API (**<filename>.h**). Different levels of device configuration profile generation are available in the three available configuration modes: quick, basic, and advanced.

Note that the initialization process typically takes 1 minute to 2 minutes to complete. If it runs successfully, users will not see any red error indicators anywhere on the screen. Sometimes, a transaction failed error might appear in the **ACE Software** window. If this happens, click the **Initialize** button.

1. A successful setup for the AD9084-FMCA-EBZ shows ~53W to 55W power drawn from the 12V DC power supply.
2. If a **-82** error appears, it possibly indicates that the **ACE Software** lost its connection to the RPC server. Try to restart the RPC and IPC servers. If this is not successful, reboot the MicroZed board and re-initialize the connection between the **ACE Software** and the ADS10-V1EBZ. In other words, the start-up sequence must be started again from power up.
3. A test signal can be applied to the ADC inputs at this point.

EVALUATION BOARD SOFTWARE SETUP

Quick Mode

The most simple **ACE Software** configuration mode is **Quick** mode, which is selected as shown in Figure 20. In **Quick** mode, the user can quickly make selections to configure the AD9084 to operate in one of several predetermined use cases, as follows (see Figure 20):

1. Select **Quick** mode in the **Configuration Mode** pull-down menu.
2. Select the board type as **FMCA** in the **Apollo Eval Board Type** pull-down menu (AD9084-FMCA-EBZ has on-board baluns for single-ended to differential conversion of the ADC and DAC analog signals).
3. If the user desires to save the device profile being used, the name of the device profile can be specified in the **Profile Name (Ucid)** field.
4. The **User Library** determines the location where the device profile will be saved to or loaded from.
5. Select from the **User Case Selections** pull-down menu one of the available predefined use cases (see Table 2).
6. Select from the **Clock Source** pull-down menu (see the Clocking Schemes (Configurations) section and the Clocking Options section for available clock options). By default, the AD9084-FMCA-EBZ is configured for **On Board Clock Source**.
7. The **ACE Software** calculates the **FPGA Reference Clock:** frequency. This clock must be supplied to the ASD10-V1EBZ by an external signal source if the **External Clock Source** option is being used.
8. In **Quick** mode, the only parameter that can be selected and/or changed in the block diagram view in Figure 19 is the ADC **Nyquist Zone** (**First** or **Second**) pull-down menu. This choice informs the **DEVICE CONFIG Profile Generator** regarding which Nyquist zone the signals for the target application reside in.
9. The **Save** button saves the configuration as a **.blueprint**, **.h**, or **.json** file to the location specified in the **User Library** field, using the name specified in the **Profile Name (Ucid)** field. Note that either the **.blueprint** or **.json** formats (not the **xx_summary.json** file) can be loaded back into the **ACE Software**.
10. The **Load** button opens a window displaying contents at the location specified in the **User Library** field. The user has the ability to navigate to a storage location of choice. Either the **.blueprint** file or **.json** file can be selected to be loaded into the **ACE Software** from this window. The path to an existing use case device profile can be specified if the desired use case is not among those available in the **Use Case Selections** drop-down menu. Note that files with the **xx_summary.json** name format contain summary profile information and are not intended for loading a use case device profile into the **ACE Software**.
11. The **Upgrade Library** button reads every blueprint from the **User Library** folder and recreates the device profiles from it in the device profile version of the **ACE Software** version being run. If there were updates to the blueprints from the **DEVICE CONFIG Profile Generator** core, these updates are made to match the new format. The **.blueprint**, **.json**, or **.h** files are also updated. This function is not needed unless there is a change to the device profile format and/or the blueprint format driven by a new software revision.
12. The **Show Library** button opens the Windows **File Explorer**, displaying files at the location specified in the **User Library**.
13. Click **Initialize** after the desired selections are made to initialize the AD9084.
14. Access to ADC Rx capture and DAC Tx vector functions is available when the **Proceed to Tx Vectors** button or **Proceed to Rx Capture** button is clicked. These functions are described in the DAC Vector Transmit section and the ADC Capture section.

To quickly select one of the predetermined use cases included with the **ACE Software** using the AD9084-FMCA-EBZ evaluation board with its default on-board clocking (ADF4382 and HMC7044), take the following steps:

1. Select the desired use case in the **User Case Selections** pull-down menu.
2. Click **Initialize**.
3. Click **Proceed to Tx Vectors** and **Proceed to Rx Capture**.

The **Apollo Eval Board Type** is **FMCA** and the **Clock Source** selected by default is ADF4382 and HMC7044 clocking that is why only the **User Case Selections** choice is needed.

EVALUATION BOARD SOFTWARE SETUP

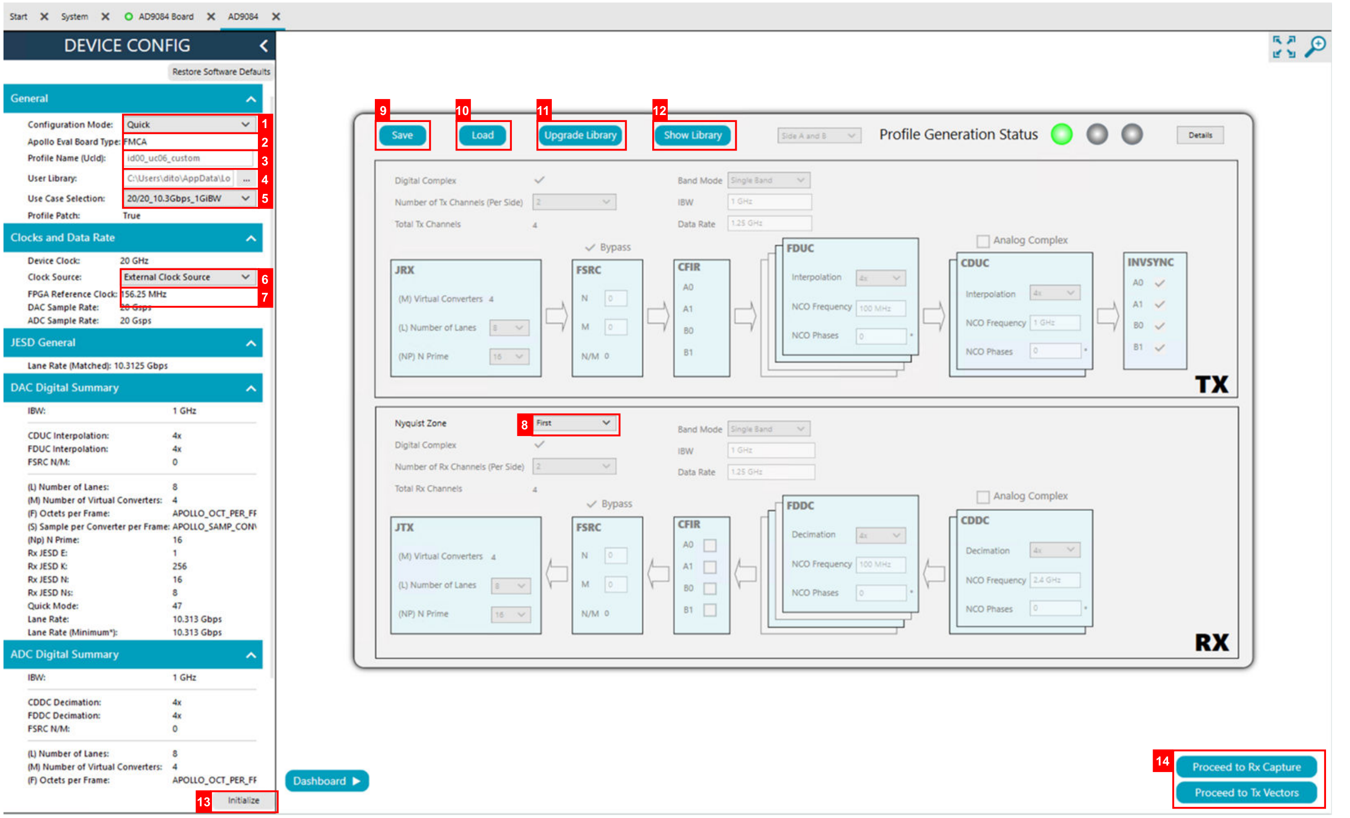


Figure 20. Device Configuration View, Quick Mode

EVALUATION BOARD SOFTWARE SETUP

Basic Mode

Basic mode provides the following additional capabilities over **Quick** mode (note that functions already described in the Quick Mode section are not described again in this section):

1. Select **Basic** mode in the **Configuration Mode** pull-down menu.
2. The **Device Clock** frequency can be specified in **Basic** mode (as well as in the **Advance** mode, see the Advanced Mode section for additional details). Changing the **Device Clock** frequency updates the **IBW** and **Data Rate** fields (Label 5). The maximum allowed clock frequency is dependent on the clock **DAC:ADC Ratio** parameter (**1:1** or **2:1**), which is described in the Advanced Mode section. A maximum of 20GHz is allowed for a **DAC:ADC Ratio** = 1:1, and up to a maximum of 14GHz is specified for a **DAC:ADC Ratio** = 2:1.
3. The number of received channels per side can be specified within the **Number of Rx Channels (Per Side)** pull-down menu, and the number of transmitted channels per side can be specified within the **Number of Tx Channels (Per Side)** pull-down menu.
4. The FSRC can be bypassed (**Bypass**) or allowed in Basic mode. This check-box is automatically checked as needed based on the **iBW** and **Data Rate** parameters. Unchecking the box allows FSRC to be used. When entering an **IBW** value, it may bypass FSRC again if there is a ratio that does not require FSRC. FSRC has an impact on the **IBW** value. For example, if an **IBW** or **Data Rate** value is set that is not possible without FSRC, and if FSRC is allowed, then the **ACE Software** will find a FSRC value to match that exact **IBW**. If FSRC is manually disabled, the **DEVICE CONFIG Profile Generator** finds the closest possible value to the user-provided **IBW**.
5. The **IBW** and/or the **Data Rate** can be specified. If the entered parameters are not achievable, the **DEVICE CONFIG Profile Generator** automatically fills with the closest compatible numbers. The user may need to iterate to find a valid configuration that is suitable for the target application.
6. Click **Initialize** after the desired selections are made to initialize the AD9084.

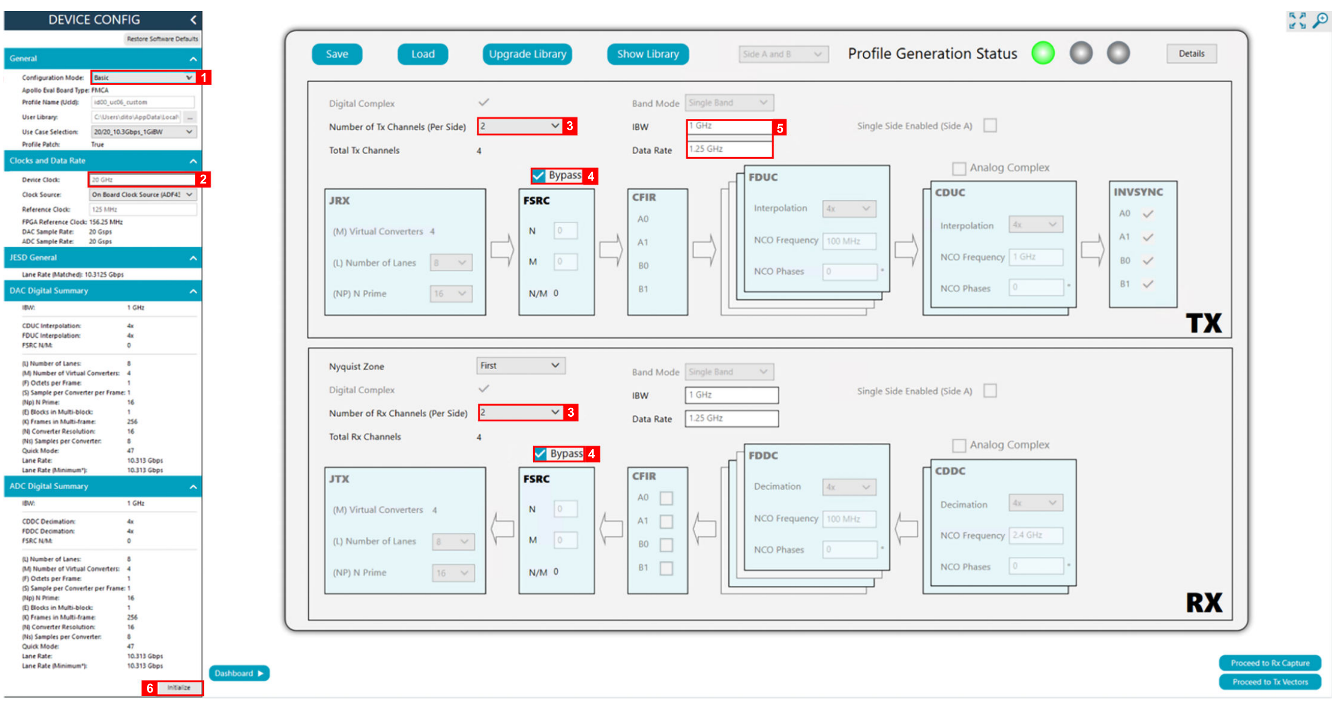


Figure 21. Basic Configuration Mode

EVALUATION BOARD SOFTWARE SETUP

Advanced Mode

Advanced mode adds the following significant **DEVICE CONFIG Profile Generator** functionality (note that information on the AD9084 configurations and terminology not defined in this user guide is available in the UG-2300 Device User Guide):

1. Select **Advanced** in the **Configuration Mode** pull-down menu.
2. Check off the **Enable DAC Shuffle** box DAC shuffle randomly selects (shuffles) MSB current sources.
3. Use the **ADC Random/Sequential** pull-down menu to select the ADC interleaving method.
4. Use the **DAC:ADC Ratio** pull-down box to select the DAC and ADC relative clock frequencies (**1:1** or **2:1**). Selecting **1:1** configures the DAC core sample rate and ADC core sample rate to be equal to each other (20GHz maximum), and selecting **2:1** configures the DAC core sample rate to be twice the ADC core sample rate. In this case, the maximum allowed clock frequency is 14GHz (applied to the clock pins), which results in a DAC sample rate of 28GSPS and an ADC sample rate of 14GSPS.
5. Use the **JESD Version** pull-down menu to select **JESD 204B** or **JESD 204C**.
6. Check the **Allow Lane Rate Adapt** check box to allow the **Lane Rate Strategy** to utilize **Lane Rate Adaptation**. Not checking this box means that only **Sample Repeat** or **Invalid Sample Insertion** can be used to rate match, if **RateMatch** is selected as the **Lane Rate Strategy** as follows. Select one of the following **Lane Rate Strategy** choices:

    a. **None**: Selecting **None** means that there is no lane rate matching performed. The user configuration determines the lane rate, and an error is reported if no matching JESD quick configuration modes are found.

    b. **Warning Only**: **Warning Only** is the same as **None** except that an additional warning is issued informing the user of a mismatch with the **Target Lane Rate**.

    c. **RateMatch**: **RateMatch** means that one of the lane rate strategies (**FSRC-Invalid-Sample-Insertion/Deletion**, **Sample Repeat** (software support pending), and **Lane Rate Adapt** (if allowed in the **Lane Rate Strategy**)) is used to attempt to match to the **Target Lane Tate**. FSRC is enabled in the digital data-path configuration pane (see Figure 22) and is the strategy utilized if FSRC is enabled. If FSRC is not enabled, **RateMatch** being checked enables the device to choose between **Lane Rate Adapt** (if allowed in the **Land Rate Strategy**) or sample repeat (software support pending). A warning is issued if the target is not met.
7. The **Use Target Lane Rate** check box targets the lane rate specified in the **Target Lane Rate** field.
8. The **Minimum Lane Rate** box is the theoretical minimum with FSRC enabled ignoring any clock constraints.
9. The J**Rx Lane Insertion Loss** pull-down menu default setting is the AD9084-FMCA-EBZ.
10. The **Lane Rate (Matched)** box is the rate that will be used for all links.
11. **Show Latency**: A preliminary latency calculator tool has been integrated into the **DEVICE CONFIG Profile Generator**. To enable this feature, check the **Show Latency** box. The estimated latency for both the receive and transmit paths will be displayed in the **DEVICE CONFIG** Wizard. This is a BETA version that has not been validated against hardware.
12. **Block Diagram View** as the following:

    a. Functional block parameters can be configured based on the needs of the application. Configure the digital datapaths based on system requirements by selecting options in this pane. This view represents one side of the device. Configurations made in this window are reflected in the **DEVICE CONFIG** Wizard on the left side of the screen. Note that M = 8 per side is not yet supported.

    b. The **Single Side Enabled (Side A)** check box is for 1T1R (1 DAC and 1 ADC) configuration. Note that single side 2T2R is not supported yet.

    c. The functional blocks in this diagram are described in the the UG-2300 Device User Guide.
13. Click **Initialize** after the desired selections are made to initialize the AD9084.
14. Click **Dashboard** to move to the **Dashboard** window (see the Dashboard View (Available in All Configuration Modes) section for additional information).

EVALUATION BOARD SOFTWARE SETUP

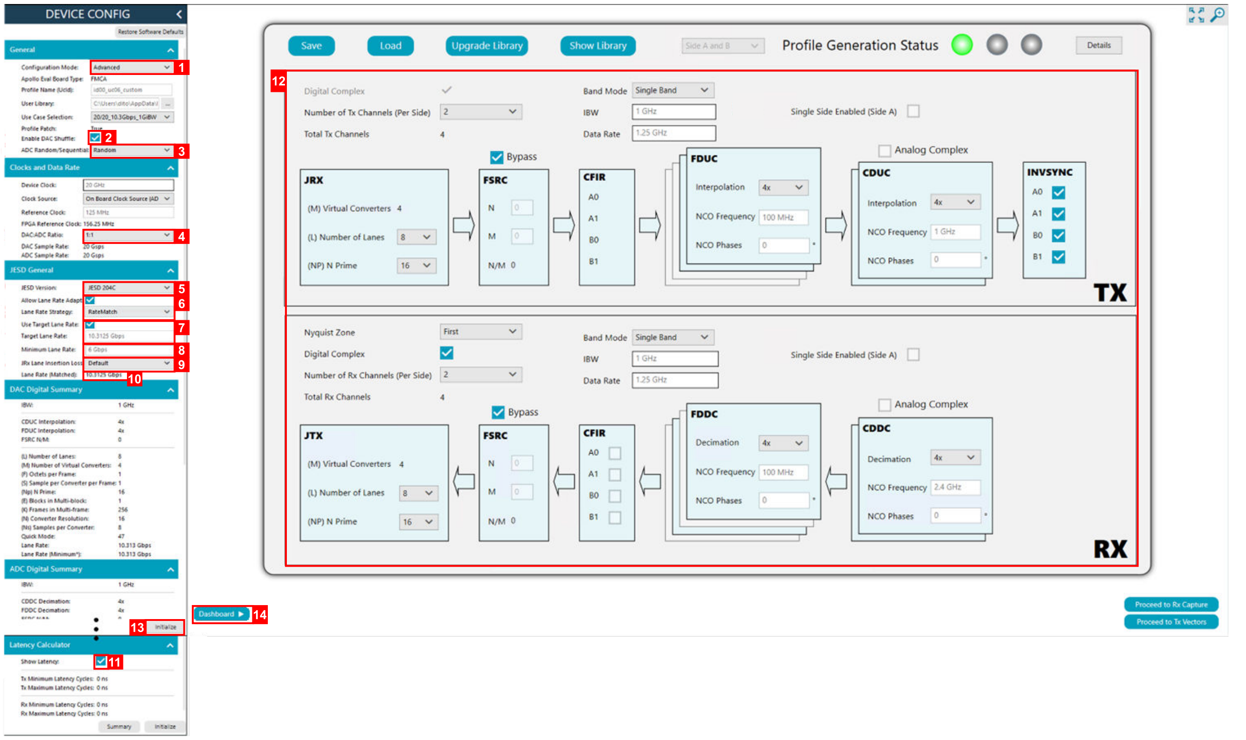
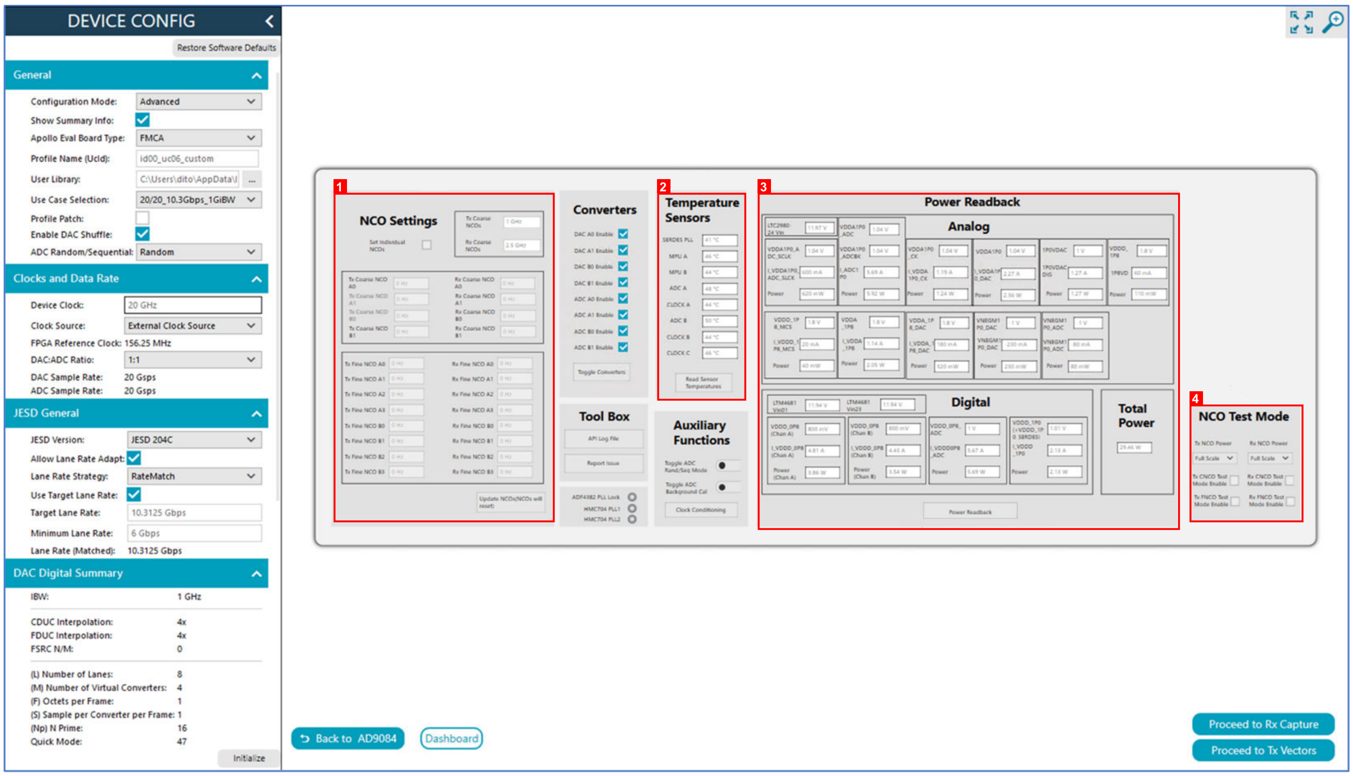


Figure 22. Advanced Configuration Mode

EVALUATION BOARD SOFTWARE SETUP

Dashboard View (Available in All Configuration Modes)

1. The NCO frequencies can be set any time after the **Initialize** setup process has completed. If 0Hz (no frequency translation) is desired, enter **0 Hz** into all of the NCO frequency fields. The individual NCO frequency fields are activated by clicking the **Set Individual NCOs** check box. The **Update NCOs(NCOs will reset)** button must be clicked for the desired NCO settings to take effect.
2. To read the temperature sensors on the AD9084 click **Read Sensor Temperatures**.
3. Click**Power Readback** to populate the **Power Readback** fields shown in Figure 23. The numbers shown in Figure 23 are for illustration only; real power numbers depend on the configuration. For board versions prior to the AD9084-FMCA-EBZ Revision C, several of the fields show as being zero, and the **Total Power** number does not include power from all domains.
4. **NCO Test Mode** is enabled and controlled from these fields. See the NCO Test Mode section for more information.



023

Figure 23. Dashboard

EVALUATION BOARD SOFTWARE SETUP

DAC Vector Transmit

Take the following steps to prepare and create vectors for DAC transmitting:

1. Click **Proceed to Tx Vectors** to prepare vectors for DAC transmit (see Figure 24).

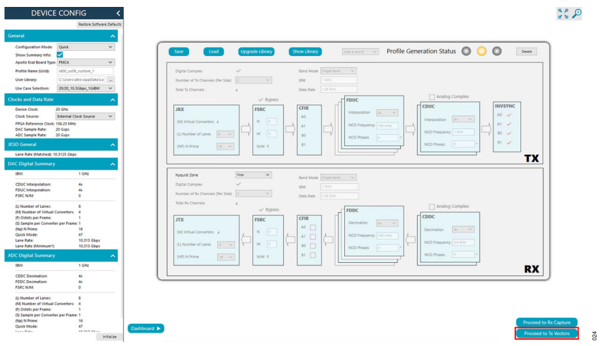


Figure 24. Tx Vectors Selection

2. Alternatively, enable **System Explorer** from the **Tools** tab, expand the **Run Time** section, and select **Tx Vectors** (see Figure 25).

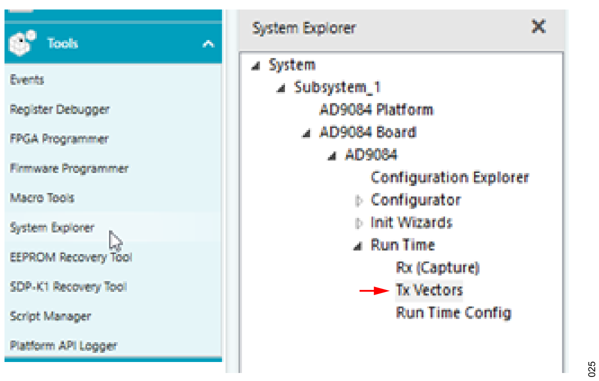


Figure 25. System Explorer

3. Under the **DATA TRANSMIT** are, click the **Vectors…** button (see Figure 26).

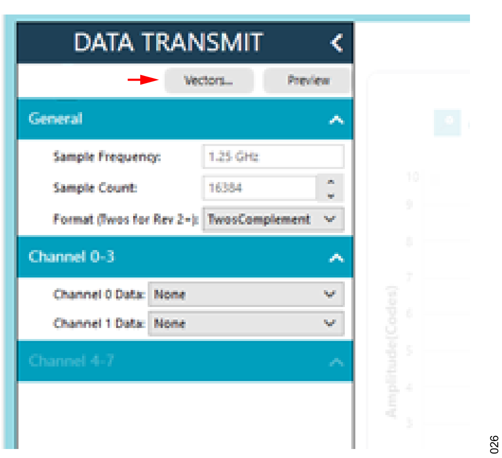


Figure 26. Data Transmit Vectors

4. Take the following steps to create a **Single Tone Vector** (see Figure 27):
   a. Under the **ADD** area, select **Single Tone Vector +** generation.
   b. Change the **Data Rate** box to the same value shown in the **Sample Frequency** box (see Figure 26).
   c. Select the **Desired Frequency** for the DAC. Note that this frequency is offset (shifted) by the numerically controlled oscillator (NCO) frequency set in the **Dashboard** (see Figure 23).
   d. The **Resolution** is the same as the **Np** (N') parameter for the use case running. **Np** (N') is shown in the **DEVICE CONFIG** section and block diagram in Figure 20, Figure 21, and Figure 22.
   e. Select the **Generate Complex Data** check box if the baseband signal is in quadrature (I/Q).
   f. Click **Preview** to generate a graphical display of the vector waveform.

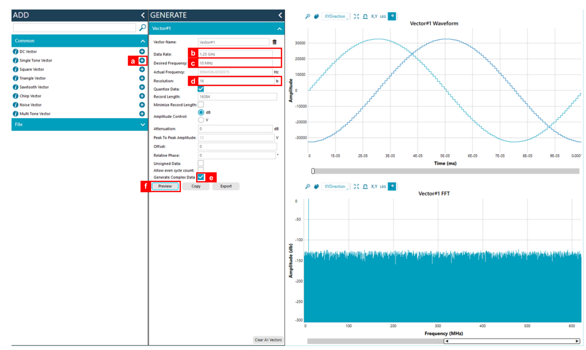


Figure 27. Generating and Previewing a Single Tone

5. Take the following steps to play the Tx vectors on the DACs (see Figure 28):
   a. Return to the **Tx Vectors** view.
   b. Select a vector for all DAC channels and sides, even for the DAC channels that are not used. Depending on the **ACE Software** revision, an error may result if this is not done.
   c. Click **Download** to download the vectors.
   d. Click **Play** to play the vectors.

Figure 28. Downloading the Vector to the DACs

EVALUATION BOARD SOFTWARE SETUP

6. From the **Dashboard** view (see Figure 23), set the NCO frequency within the **NCO Settings** area to the desired frequency to mix with the baseband signal.
7. At this point, the desired tone appears at the DAC outputs.

ADC Capture

Take the following steps to capture ADC outputs and vie time domain and fast Fourier transform (FFT) waveforms:

1. Click **Proceed to Rx Capture** to capture ADC outputs and view time domain and FFT waveforms (see Figure 29).

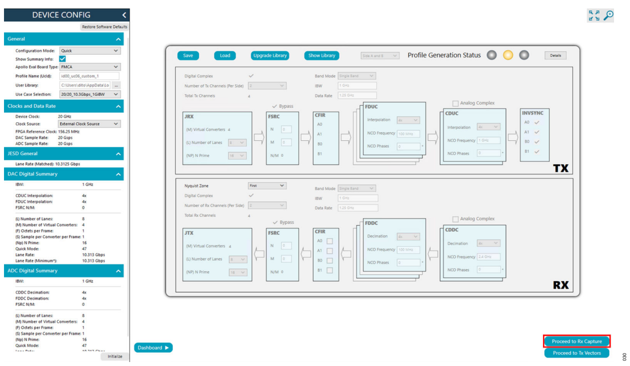


Figure 29. Rx Capture Selection

2. ADC **Rx Capture** can also be accessed through System Explorer (see Figure 30).

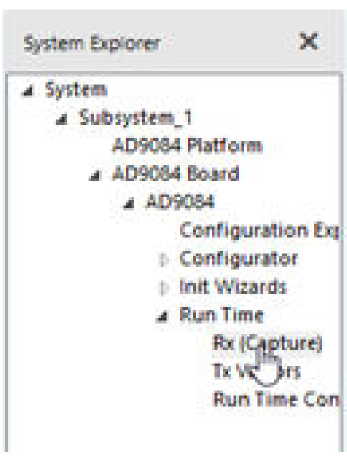


Figure 30. Rx Capture From System Explorer

3. The **FFT Analysis** window then opens. From the left-hand pane, select the **FFT** option (see Figure 31).

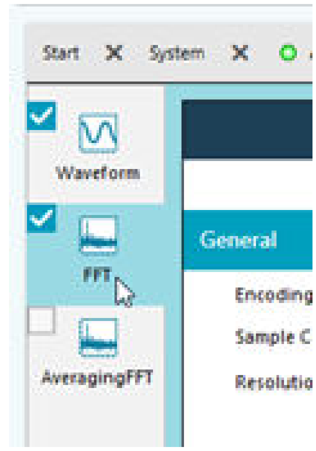


Figure 31. Selecting FFT in Analysis Window

4. Click **Run Once** to display the captured FFT (see Figure 32).
   a. Currently, **Run Continuously** sometimes leads to ADC capture errors; however, this is not related to the AD9084and does not indicate an issue with the AD9084.

Figure 32. Selecting Run Once or Run Continuously

5. An FFT from the captured data displays as shown in Figure 33. With a single tone sinusoid input at 2.65GHz −5dBFS amplitude, and the NCO set to 2.5GHz, the captured tone appears at ~150MHz, and the noise spectral density (NSD) value reads approximately –145dBFS/Hz or better. This value takes into account all the interleaving spurs that are showing up in the FFT and is expected to improve as the AD9084 development team makes improvements to the ADC calibration subroutines. The number of samples to capture for calculating the FFT is specified in the **Sample Count** pull-down menu. Note that higher sample counts result in a proportionally narrower FFT bin width, lower bin noise, and increased FFT frequency resolution.

EVALUATION BOARD SOFTWARE SETUP

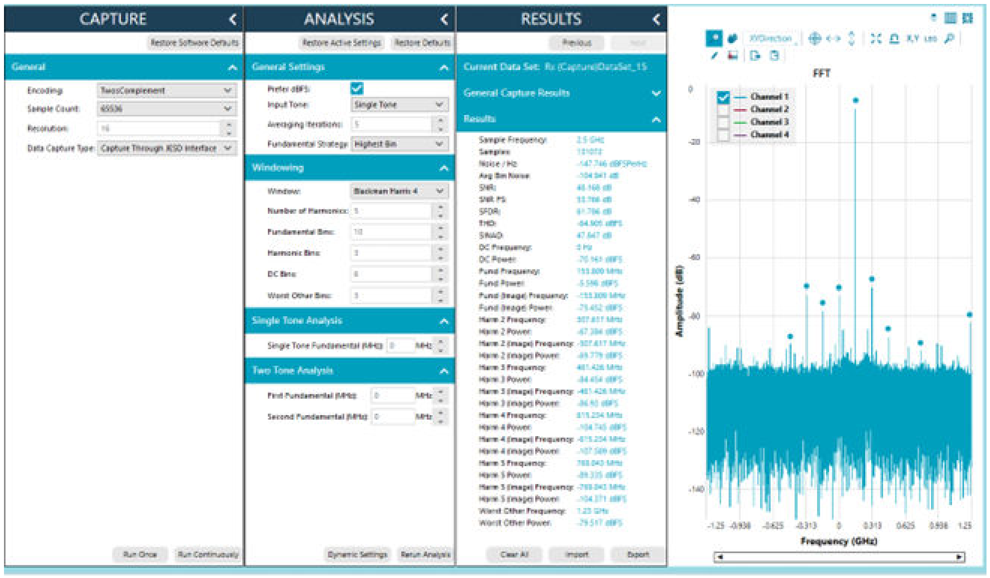


Figure 33. Single Tone FFT at 2.3GHz from ADC_A0

EVALUATION BOARD SOFTWARE SETUP

Apollo MxFE Features Currently Supported in the ACE Software

The following sections detail the general features supported within the ACE Software; however, it is not meant to be an exhaustive list of all the available features.

PFILT

The **PFILTER CONFIGURATION WIZARD** can be found in the **System Explorer** tab (see Figure 34). If a device profile is loaded with **PFilter** enabled, the **ACE Software** configures and loads the coefficients during initialization, and they then come up active. The PFILT block can also be enabled during run-time (see the **Enable PFilter** area of Figure 34). Coefficients can be loaded from the device profile or a text file on your local machine. Select **PFilter Mode** by going to the **Enable PFilter** section and using the **PFilter Mode** pull-down menu. Current support includes **Dual Real Mode**, **Half Complex 0**, and **Half Complex 1** (for more information on these modes see the AD9084 Design User Guide (UG-2300)). Once the mode and coefficients are selected, click **Apply** for them to take effect.

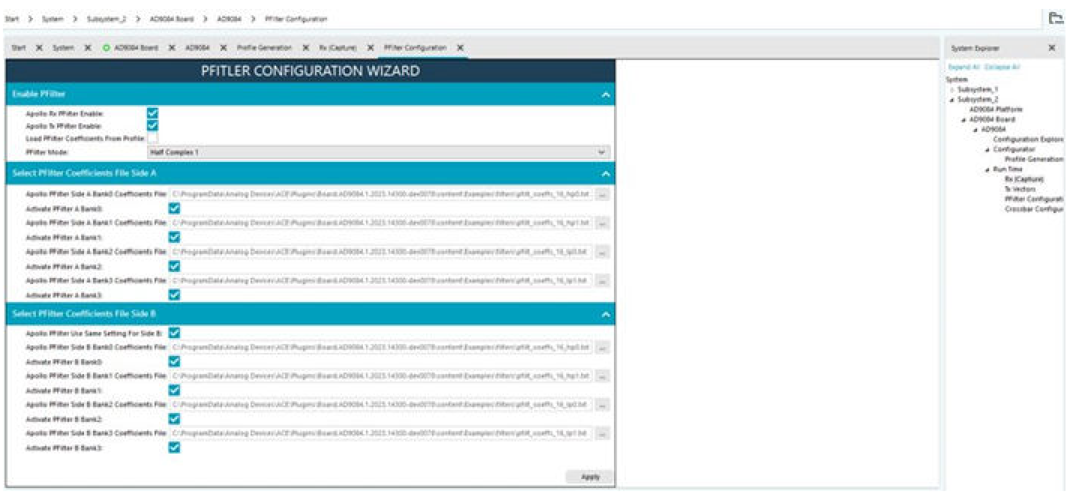


Figure 34. PFILT Configuration

EVALUATION BOARD SOFTWARE SETUP

Clocking Options

In addition to the on-board (Apollo and field-programmable gate array (FPGA)) clock configurations, the **ACE Software** supports two additional clocking options: **On Board Clock Source (ADF4382)** and **On Board Clock Source (ADF4382 & HMC7044)**. To select one of these additional options, use the **Clock Source** pull-down menu shown in the Apollo **DEVICE CONFIG** wizard section under the **Clocks and Data Rate** area (see Figure 35).

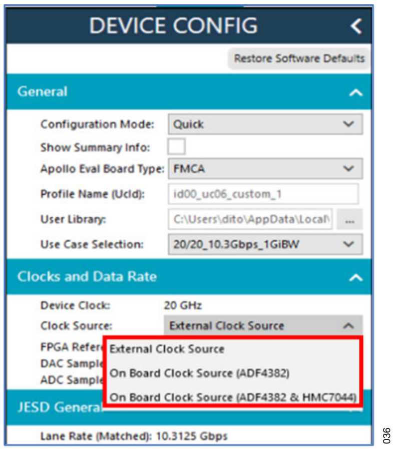


Figure 35. Apollo Clocking Options

See the Evaluation Board Connection Overview section for further details on hardware required to evaluate the AD9084, including the ADS10-V1EBZ and the AD9084-FMCA-EBZ evaluation boards. Note that the FPGA reference clock source must be synchronized to the Apollo clock source. In addition, hardware changes are required for the **External Clock Source** option. Contact ApolloSupport@analog.com for additional information.

Using the **On Board Clock Source (ADF4382)** option, the Apollo clock is supplied by an on-board ADF4382. In this configuration, a reference clock signal must be applied to J17 on the Apollo MxFE board. The FPGA reference still must be applied from an external source and synchronized to the J17 reference clock signal. When the **On Board Clock Source (ADF4382)** option is selected, an **ADF4382 Reference Clock** field appears in the Apollo **DEVICE CONFIG** wizard. This field automatically fills with the default reference frequency that works with the selected Use Case. The user can choose and enter an integer frequency from 10MHz to 500MHz, as required, for the application. Note that hardware changes are also required for this on-board clock source option. See the Clocking Schemes (Configurations) section for additional information. In addition, contact ApolloSupport@analog.com for additional information.

The **On Board Clock Source (ADF4382 & HMC7044)** option is the default configuration. In this option, both the Apollo clock and FPGA reference clock are supplied by an on-board ADF4382 and HMC7044. When the **On Board Clock Source (ADF4382 & HMC7044)** option is selected, an **External Reference Clock** field appears in the Apollo **DEVICE CONFIG** wizard. This field automatically fills with the default reference frequency that works with the selected Use Case. The user can choose and enter an integer frequency from 10MHz to 500MHz, as required, for the application.

CFIR

Cfir Configuration is available for Rx only in the **System Explorer** (see Figure 36).

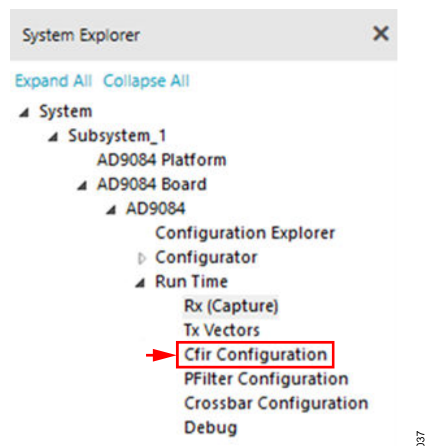


Figure 36. Cfir Configuration in System Explorer

Select **Cfir Configuration** in **System Explorer** to open the **CFIR CONFIGURATION WIZARD** shown in Figure 37.

EVALUATION BOARD SOFTWARE SETUP

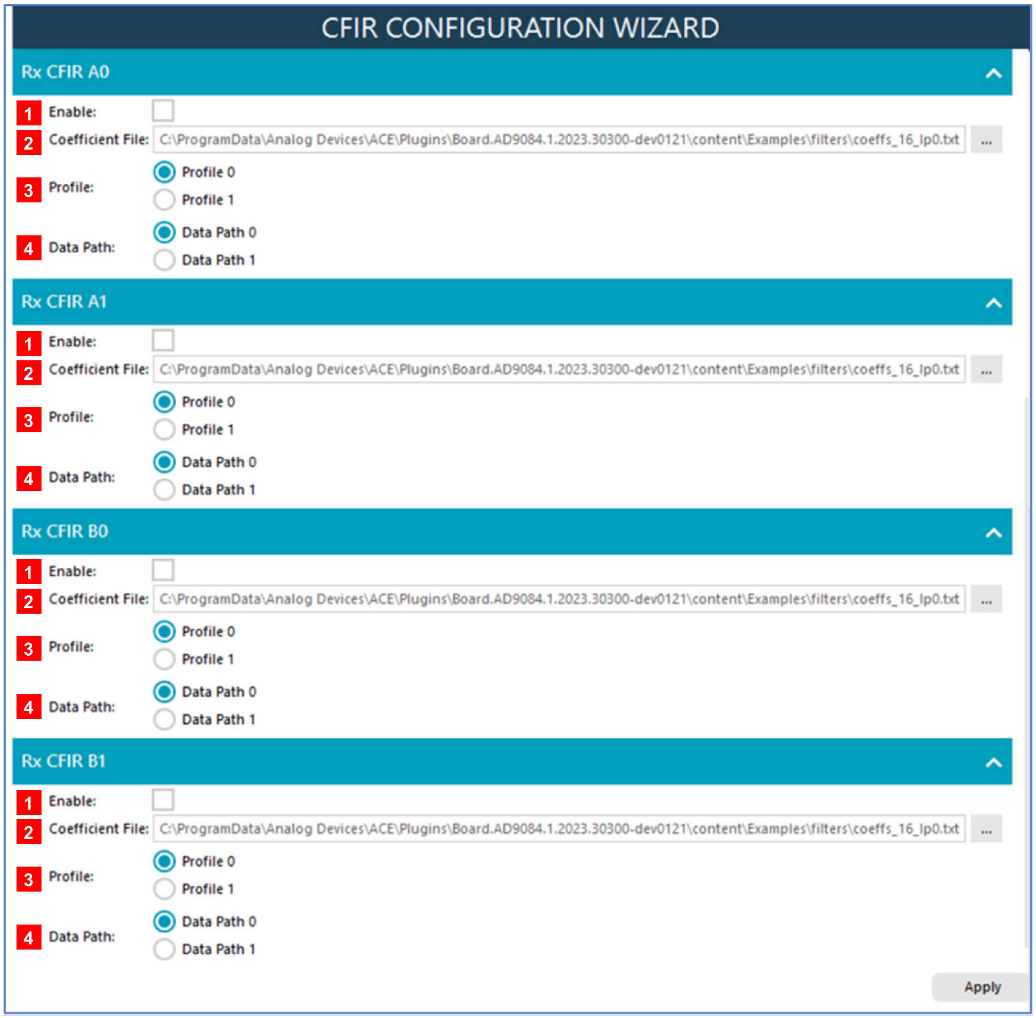


Figure 37. CFIR CONFIGURATION WIZARD

Take the following steps to configure Rx CFIR (also see Figure 37):

1. Check the **Enable:** check box to select which Rx channel has CFIR enabled.
2. Users can use the **Coefficient File:** box to enter the path to the desired CFIR coefficients. Five preloaded sample coefficient files (two low pass, two high pass, and one 0) are included and available by clicking the search icon at the right side of the ... field. A typical location for the preloaded files is: **C:\ProgramData\Analog Devices\ACE\Plugins\Board.AD9084.xxxx\content\Examples\filters**.
3. The CFIR block (**Profile:**) can store two hopping profiles (each containing filter coefficients) that the device can hop between. Use of this feature is not in the **ACE Software**. In the future, this field will allow the user to select which hopping profile to use.
4. In the **Data Path:** area, each Rx CFIR can send its data down one of two datapaths, which can be selected here.

Buffer Memory (BMEM) Capture and Arbitrary Waveform Generator (AWG)

The Rx BMEM feature has the ability to capture outputs from the ADC cores prior to any digital processing or to generate a stored digital waveform and send it down the datapath.

The BMEM capture feature is available in the **CAPTURE** pane of the **Rx Capture** window. To select this feature, select the **Capture Through RAM** in the **Data Capture Type** pull-down menu (see Figure 38).

EVALUATION BOARD SOFTWARE SETUP

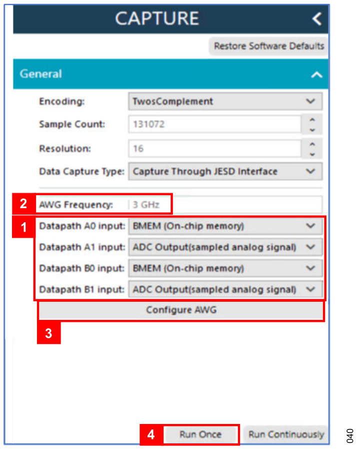


Figure 38. BMEM Capture Feature Selection

Click **Run Once** in the **Rx Capture** window with **Capture Through RAM** selected to produce a full bandwidth (no decimation) FFT. The number of samples is fixed at 64k per channel when using the BMEM capture feature.

The waveform generation capability of the Rx BMEM can be exercised with a single tone by using the controls shown in Figure 39 within in the **Rx Capture** window and by following these steps:

1. Select which Rx channel receives the BMEM signal.
2. Select the **AWG Frequency** by using the pull-down menu.
3. Click **Configure AWG** to load a sinusoid into the buffer of the BMEM and start a new AWG play.
4. Click **Run Once** to capture the waveforms sent from the BMEM feature.

Note the following:

- BMEM AWG and BMEM capture cannot both be selected simultaneously. One or the other must be chosen.
- The signal created by BMEM AWG is subject to any digital signal processing (DSP) functions configured in the Rx datapath.
- The signal captured by BMEM is prior to any DSP.

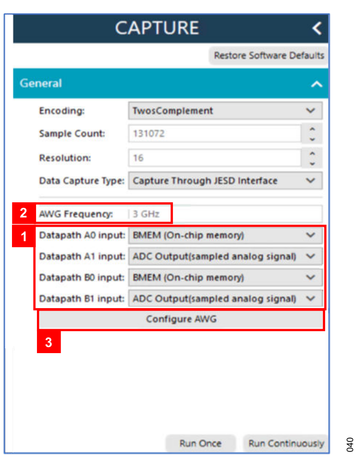


Figure 39. Waveform Generation of the Rx BMEM

NCO Test Mode

NCO Test Mode uses the NCOs in the digital datapath of both or either of the Rx and Tx paths to create a single tone at the frequency chosen by the user. Within the Tx path, the **NCO Test Mode** tone is fed to the DACs, and the resultant analog output can be observed on a spectrum analyzer, or the DAC output can be connected to the ADC inputs by a coax cable and observed on the ADC output FFT. Within the Rx path, the **NCO Test Mode** creates a digital test tone and sends it down the Rx datapath. This tone can be captured at the JESD interface and observed the same way any other ADC output is captured and observed.

The following **NCO Test Mode** controls are available in the **ACE Software** window (see Figure 40):

- Amplitude can be adjusted by using the **Tx NCO Power** and **Rx NCO Power** pull-down menus. The power options are **Full Scale** (Fs), **Fs/2**, and **Fs/4**.
- Both or either CNCO and/or FNCO can be enabled by checking the appropriate box in the NCO Test Mode area.
- When **Tx CNCO Test Mode Enable** or **Rx CNCO Test Mode Enable** is selected, the **ACE Software** uses coarse NCO setting values and ignores FNCO setting values.
- When **Tx FNCO Test Mode Enable** or **Rx FNCO Test Mode Enable is selected**, the **ACE Software** uses coarse NCO setting values in addition to the FNCO setting values.
- New updated NCOs setting values will not execute until the appropriate box in the **NCO Test Mode** area is unchecked and rechecked.

Figure 40. NCO Test Mode Controls Within the ACE Software

Rx NCO frequency guidelines include the following:

- Rx FNCO in **NCO Test Mode** only works at frequencies less than 1/2 the iBW.
- To avoid having the resultant spectrum appearing artificially noisy, the Rx CNCO and Rx FNCO values must be something other than SAMPLE_FREQUENCY/N, where SAMPLE_FRE-

EVALUATION BOARD SOFTWARE SETUP

QUENCY is the frequency of the ADC sampling clock, and N is an integer.

Debug

System Explorer contains a **Debug** tab that opens various inspection functions that are useful for status and debug purposes (see Figure 41).

Select the desired channel and Tx and/or Rx in the pull-down menus and then click the associated inspection button to produce the status and debug information (see Figure 41).

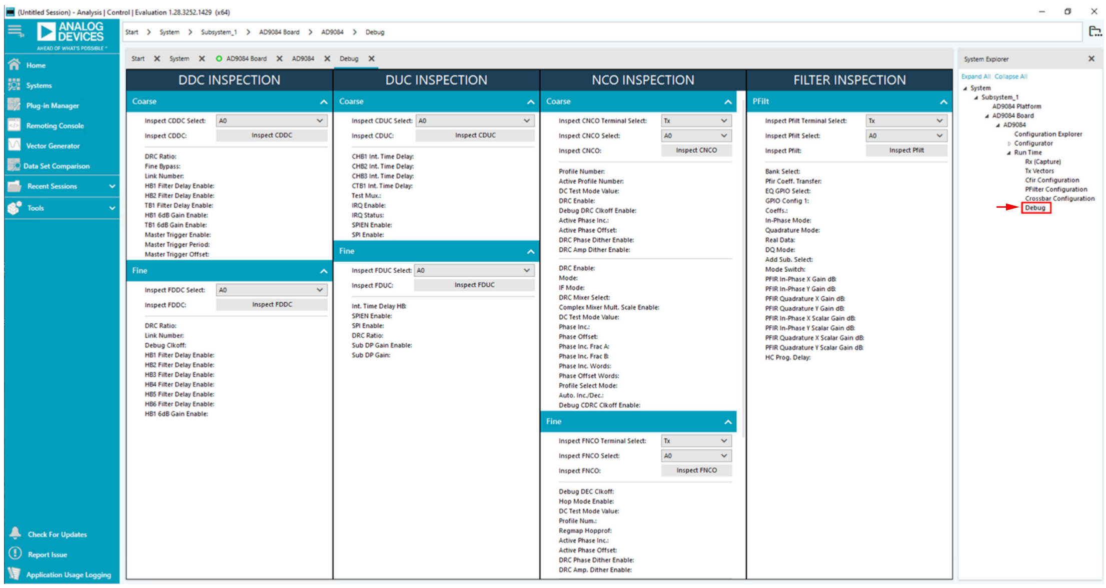


Figure 41. Debug Tab

EVALUATION BOARD SOFTWARE SETUP

Loopback 0

Loopback 0 can be found in **System Explorer** under the **Loopback0 Configuration**. Loopback 0 loops back an ADC input to a given DAC on the same side (A or B). The loopback feature is internal to the device. To turn on and off Loopback 0, select the **Enable Loopback0 Mode** check box (see Figure 42). Use the **Set Loopback Crossbar** drop-down menus within the **CROSSBAR CONFIGURATION WIZARD** to select which ADC input to route to which DAC (see Figure 42).

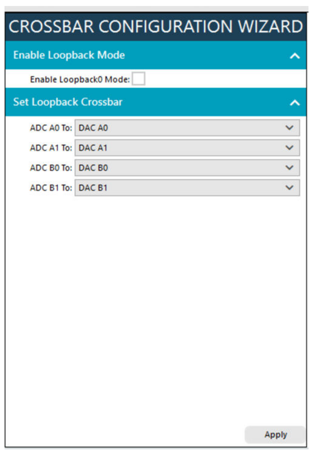


Figure 42. Enabling Loopback 0 Mode

Known Issues

Known issues for the Apollo MxFE evaluation board setup follow:

1. The **ACE Software** does not load the common vector options (see Figure 43), which is related to a bug in the **Core ACE Software**. The only known fix is to restart the **ACE Software**.

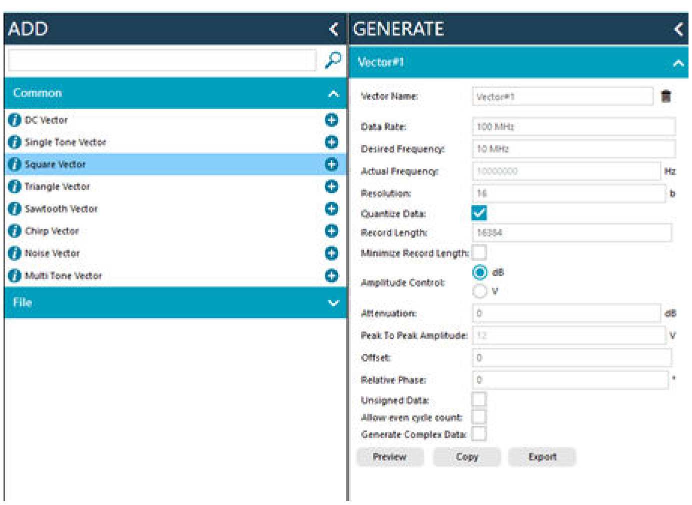


Figure 43. Common Vector Options

2. The MicroSD card image that is provided together with the Apollo MxFE evaluation board is designed to automatically program the FPGA on the ADS10-V1EBZ. However, occasionally, the FPGA loading does not fully complete and corrupts. When this happens, the **FPGA DONE** LED (DS11) and **FPGA INITB** LED (DS8) on the ADS10-V1EBZ data capture and transmit board repeatedly turn on and off. Power off the ADS10-V1EBZ and power it back to stops this. The normal expected sequence is as follows:
   a. Power on the ADS10-V1EBZ.
   b. The MicroZed™ board then loads its FPGA (Zynq) image from the microSD card and starts the embedded remote procedure call (eRPC) server.
   c. Then, the SSH connection to the MicroZed (192.168.0.10) is successful.
   d. The ADS10-V1EBZ FPGA is then programmed, and **FPGA_DONE** LED is lit.
   e. The ADS10-V1EBZ **PG_C2M** and **PG_M2C** LEDs are lit
   f. Switch on the power on to the AD9084-FMCA-EBZ.
   g. The ADS10-V1EBZ then signals the AD9084-FMCA-EBZ to power up. A string of LEDs by the power connector light up to indicate **power good**, and the fan starts rotating.
   h. Finally, the evaluation board can be programmed using the **ACE Software**, PyApp, or the API.
3. Occasionally, the **ACE Software** does not start the server automatically. If the **ACE Software** shows a starting server error, take the following steps:
   a. Open a command prompt and SSH into the MicroZed with the same login credentials as before. The full command is SSH **root@192.168.0.10** (**Password**: **analog**).
   b. Using standard Linux commands, navigate to the **home/analog** folder **root@ADS10Apollo:/# cd /home/analog**.
   c. Create the Apollo server manually: **root@ADS10Apollo:/home/analog# ./ApolloServer**.
4. A power delivery network (PDN) fault can occur due to the power transients when the Apollo MxFE is reinitialized into the same or a different configuration. It is advisable to push the PDN resequencing switch (see Evaluation Board Photographs) before reinitializing the Apollo MxFE.
5. If the evaluation setup has any issues after changing device configuration, changing software, changing hardware connections, and so on, it is recommended to power cycle the boards as previously described.

EVALUATION BOARD SOFTWARE SETUP

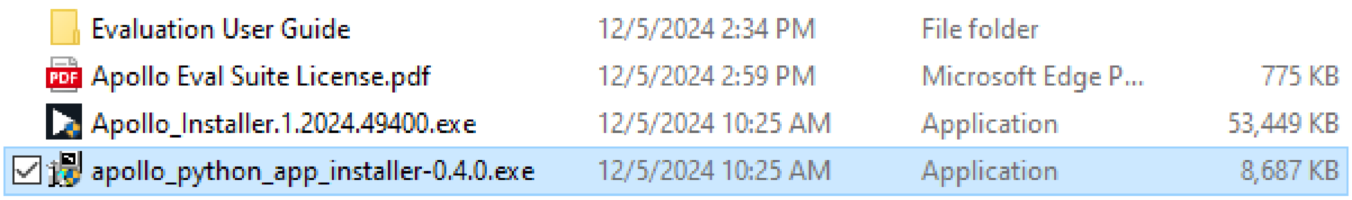
PYTHON APPLICATION (PYAPP)

The Apollo MxFE PyApp is an alternate method for interfacing to the Apollo SPI register map similar to using the API and is included in the distribution package together with the **ACE Software** and can be installed using the included executable (see Figure 44). Python 3.8.x interpreter must be installed separately with its installer from the Python website.

Additional installation steps are described in the Additional Python Installation Steps and Configuration section.

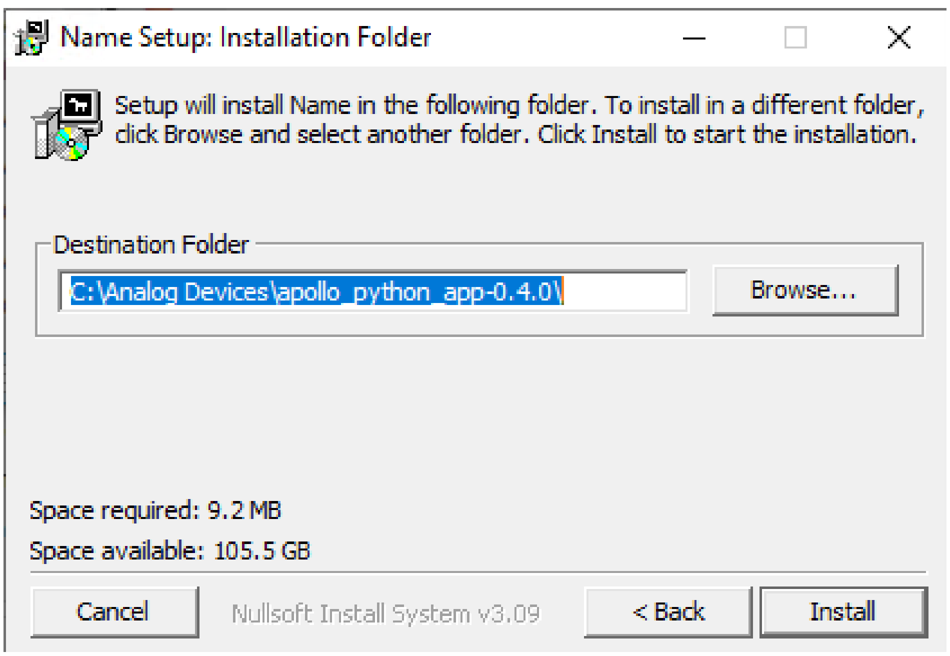
An Integrated Development Environment (IDE) software, such as Visual Studio Code (VS Code) or PyCharm, is generally required to view and compile the resulting Python code.

The PyApp installer will ask for a location to install the package. Select a location that is easy to access because you must execute the code from that location and use it as your working directory (see Figure 45 and Figure 46). To configure the PyApp, follow the instructions given in **getting_started.md** file in the top level of the installation directory. This file subsequently points you to an explanation of the package contents in the **apollo_app/README.md** file.



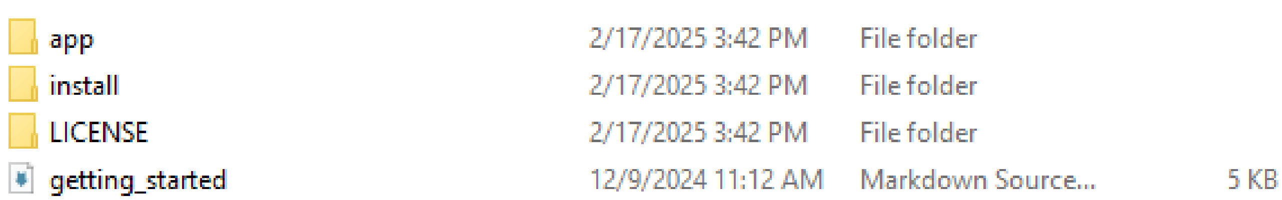
046

Figure 44. Python Installer in Evaluation Software Folder



047

Figure 45. Install Location Selection



048

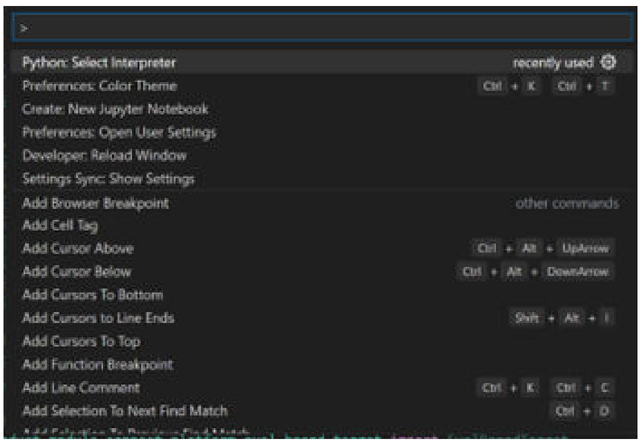
Figure 46. Package Contents Post Installation

EVALUATION BOARD SOFTWARE SETUP

Configuring the IDE: VS Code

Once the Python package is installed, open the **src** folder (rather than the **main.py** file) using VS Code. Ensure installation of the Python extensions VS Code requires to interpret Python. To load the Apollo virtual environment and its interpreter in VS Code, press **Ctrl** + **Shift** + **P**, and type **python select interpreter**. Select **enter interpreter path** and then **find**. Do not pick the native Python 3.8.x interpreter. Navigate to the virtual environment you created and select **Scripts/python.exe**. Reload the terminal (**Ctrl** + **~**), and the name of the virtual environment appears before the terminal prompt in parenthesis.

To run code in the IDE, press **Play** in the top right or use the debug feature, which is also in the top right.



049

Figure 47. VS Code Configuration

General Use

In general, using the PyApp has a similar feel to the API. APIs, structs, and enums are all accessible in a python syntax (see the Using Enums section and Using Structs section for additional information). Examples in the C API can be used as reference, and the same APIs can be called in the PyApp. The API can also be opened with VS Code and used as a reference for finding new APIs to integrate into Python examples. Python and API logs are generated into a logs folder and are encouraged to be used for debugging. Running the C examples also generates logs that can be compared to those generated in Python.

Defining Hardware Using system_config.py

As described in **apollo_app/README.md** installation file, the user starts with editing the **system_config.py** file located in the **config** folder. This file contains hardware revision selections, clocking options, and default device profile selections. Comment or uncomment to select the appropriate default test profile for your board. This line can be changed anytime to point to your own custom profile **.json** path and file. The vector data is the default file location to save files to, which can be changed; however, note that an absolute path is best. The information in this file is used for all examples programs in the directory (including the **main.py** file).

Figure 48. config/system_config.py

EVALUATION BOARD SOFTWARE SETUP

startup.py

The **startup()** method in the **startup.py** file initializes the Apollo MxFE and any board attach based on inputs from the **system_config.py** file. All standalone example files begin by running the **startup()** file.

```
if __name__ == "__main__":
    hf.clear_scr()
    from config.config_parser import ConfigAssigningParser

    parser = ConfigAssigningParser()
    args = parser.parse_args()

    profile = Profile.load(system_config.apollo_profile_json)
    assert isinstance(profile, Apollo_top_t), log.exception("Profile instance not valid")

    assert profile.profile_cfg.is_8t8r == False, (
        "Got 8t8r profile in fmca_4t4r_fullchip.py example. Make sure the filepath `apollo
    )

    fourT4R_config = FourT4RFullChipConfig()

    with StartUp(profile) as platform:
        run(platform, fourT4R_config)
```

051

Figure 49. Startup() Running for the 4T4R Full Chip Standalone Example

The initialization flow follows the start-up diagram in the UG-2300 Device User Guide. This flow must be followed in the customer application. A dictionary of the Apollo platform is returned from this function, which must be utilized with the **Context Manager**.

Example Files for User Implementation

The example files contain examples that can serve as guides for user implementation. These examples can run standalone or be imported into the higher level **main.py** program, where several examples can run in succession. The current example files include the following:

- **Tmu_read.py** starts up the Apollo MxFE and prints the time measurement unit (TMU) data.
- **Adc_bmem_capture.py** starts up the Apollo MxFE and does a BMEM capture through the ADC. The specified ADCs BMEMs are returned in a dictionary and can be saved to files for plotting. NOTE that these captures are unsigned. **VisualAnalog** can handle this; however, some other plotting tools may need the twos complement to plot correctly.
- **Tx_nco_test_mode.py** starts up the Apollo MxFE and then sets the coarse NCOs to test mode (no JESD data). It does a frequency sweep with the NCO before finishing.
- **Power_readback.py** configures the Apollo MxFE and then reads the power back. It uses a utility in the **power_module** to calculate the total dissipated power as well as to give individual rail information. Note that two of the rails (**I_1P3V_ADC1P0** and **I_VDDA1P0_ADC_SCLK**) are uncorrected. These are corrected and reported as **I_ADC1P0** and **I_ADC_SCLK**. They are left in for reference.
- **4t4r_fullchip.py** is the full chip example for the AD9084. Note that a device profile supporting 4T4R must be selected to use this. The transmit portion of this example assumes that M = 4 JESD mode with one complex vector per DAC.
- **Adc_mode_switch.py** helps the ADCs of the Apollo MxFE support two slice modes: random and sequential. This example swaps between the two, calibrates, and captures to demonstrate the feature. The example uses three different methods to switch the mode: via firmware, the register map, and the general-purpose input and output (GPIO).
- **Fft_sniffer.py** is the FFT sniffer base functionality example.
- **Plotter_example.py** puts the DACs into test mode (no JESD) and requires the DAC output to be looped externally to the ADC input. The code opens a browser window to an FFT plotter and sweeps the receiver NCO in 1GHz steps from 1GHz to 8GHz, while sweeping the transmit NCO ±200MHz across the current receiver NCO value (for example, −800MHz to +1200MHz, then −1800MHz to +2200MHz, and so on). The current NCO value is shown in the console output, and the plotted FFT appears in the browser window.
- **Tx_nco_ffh.py** loads 16 different CNCO tuning word profiles and hops between them using direct hopping with both the GPIO and register map, as well as triggered hopping where the next profile is selected via the GPIO or register map, and then activated via a SPI trigger.

More examples will be added in future releases.

If any example cannot find the logger utility, import the operating system and edit the root directory as shown in Figure 50.

```
# Run ADC BG Calibration
log.info("Present -7dBFS tone to ADCs, Press enter to run BG Cal for 60s")
hf.pressEnter()
adc_cal_chans = Apollo_adc_select_e.APOLLO_ADC_A0 | Apollo_adc_select_e.APOLLO_ADC_A1 | Apollo_adc_select_e.APOLLO_ADC_B0 | Apollo_adc_select_e.APOLLO_ADC_B1
adi_apollo.adc.bgcal_unfreeze(adc_cal_chans)
time.sleep(60)
```

053

Figure 50. Appending the Root Directory

EVALUATION BOARD SOFTWARE SETUP

Fmca_4t4r_fullchip.py

The **fmca_4t4r_fullchip.py** example file runs the JESD transmit and capture. After startup, ADC background calibration runs, which is a 60sec calibration for ADC performance optimization (see Figure 51). For optimal performance, this calibration must be done with a −7dBFS signal going to each ADC that is used for testing.

```
# Run ADC BG Calibration
log.info("Present -7dBFS tone to ADCs, Press enter to run BG Cal for 60s")
hf.pressEnter()
adc_cal_chans = Apollo_adc_select_e.APOLLO_ADC_A0 | Apollo_adc_select_e.APOLLO_ADC_A1 | Apollo_adc_select_e.APOLLO_ADC_B0 | Apollo_adc_select_e.APOLLO_ADC_B1
adi_apollo.adc.bgcal_unfreeze(adc_cal_chans)
time.sleep(60)
```

053

Figure 51. ADC Background Calibration

Transmit vectors are generated as 10% and 20% of the baseband data rate and 6dB backoff using the transmit helper functions located in the **apollo_app/product_module/common/transmit** folder. These vectors are loaded to the FPGA to be sent to Side A and Side B, respectively. Once transmit starts, serialization and deserialization (serdes) calibration runs if the lane rate is greater than 16Gbps.

After transmitting, a capture is collected with the ADCs and saved to the **rx_capture_filepath** folder located in the **system_config.py** file. The capture size can be modified using the **capture_size** file variable. Note that too large of a capture may crash the example due to a timeout between the FPGA and PC.

```
# samples per virtual converter.
capture_size = 128 * 1024

cap_data = rx.jesd_capture(adi_fpga_apollo, profile, adcs, capture_size)

# # Save captured IQ Data in txt files.
for i in range(0, len(cap_data)):
    np.savetxt(system_config.rx_capture_filepath + f"apollo_adc_cap_iq_data16_conv_{i}.txt", cap_data[i], fmt="%i")
```

054

Figure 52. ADC Capture

Runnig Main.py

Each of the examples previously listed can be run directly from the example file, or these examples can be imported into the **main.py** file and run together. The **main.py, startup()** file runs once at the beginning of the program regardless of the number of examples imported. If the **adc_mode_switch** example file runs inside of the **main.py** file, the **adc_mode_sw_enable** argument must be set to **True**, as is shown in Figure 53. Note that the default value is **False**.

```
with StartUp(profile, adc_mode_sw_enable=True) as platform:
    board: ApolloBoard = platform.board_ex
    adi_apollo: ApolloDevice = platform.adi_apollo_ex
    adi_adf4382: Adf4382_ex = platform.adi_adf4382_ex
    adi_fpga_apollo: FpgaApollo_ex = platform.adi_fpga_apollo_ex
    adi_hmc7044: Hmc7044_ex = platform.adi_hmc7044_ex
    adi_ltc2977: Ltc2977Device = platform.adi_ltc2977_ex
    adi_ltc2980: Ltc2980Device = platform.adi_ltc2980_ex
    adi_ltm4681: Ltm4681Device = platform.adi_ltm4681_ex
    adi_adl6331: Adl6331_ex = platform.adi_adl6331_ex
    adi_adl6332: Adl6332_ex = platform.adi_adl6332_ex
    adi_adf4030: Adf4030_ex = platform.adi_adf4030_ex
```

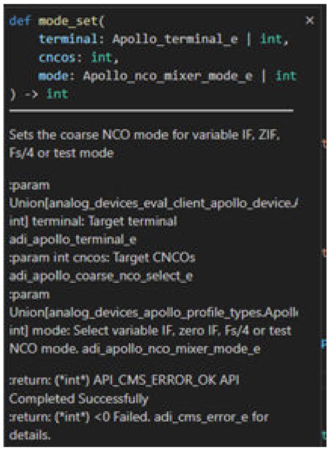
055

Figure 53. Start-Up Object Instantiation

EVALUATION BOARD SOFTWARE SETUP

Using Enums

To use enums in the PyApp, start with typing an API and use IntelliSense for VS Code to get the description of the API. The image shown in Figure 54 appears after typing **adi_apollo.cnco.mode_set**, showing the required arguments and their types.



056

Figure 54. IntelliSense Function Description

057

Figure 55. Use Example of All Enums

Note that in the stop section, the **Apollo_terminal_e** and **Apollo_nco_mixer_mode_e** enums are listed for the first and third arguments; however, for the CNCOs argument, it is looking for an **int**. Looking at the actual docstring description in the lower section, the enum for CNCOs is listed as **adi_apollo_coarse_nco_select_e**, rather than int. However, this is the API enum it is expecting because the PyApp is a Python mirror of the API. To access this enum though Python, it must change to **Apollo_coarse_nco_select_e**; however, either way works. Figure 55 shows a call using all enums.

EVALUATION BOARD SOFTWARE SETUP

Using Structs

In the PyApp, structs are treated as classes, which is demonstrated at start-up and in the TMU example. The user instantiates an object of whatever struct that is required by the API.

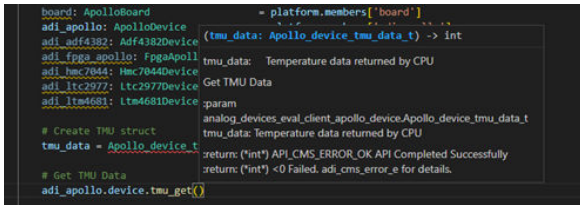


Figure 56. API Requiring a Struct

Once the struct is identified, **Apollo_device_tmu_data_t**, which in this example, creates a variable of that type.

Figure 57. TMU Struct Example

This variable gets passed into the **tmu_get()** API call, and the TMU data can be accessed through the struct. Other structs may have information that must be filled in before passing it into the function. In this case, the struct is created the same way; however, the user see the structs members using IntelliSense and then fill these structs in before passing them into the API. Figure 58 and Figure 59 show that the **adi_apollo.pfilt.mode_pgm()** call expects an **Apollo_pfilt_mode_pgm_t** struct. This struct is created in Figure 59 and can then be given as the argument to the **mode_pgm** function.

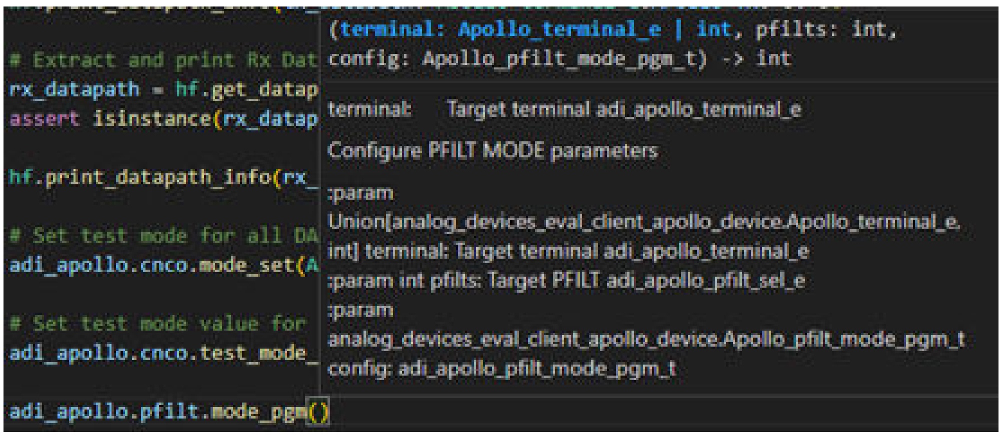


Figure 58. Another Struct Example

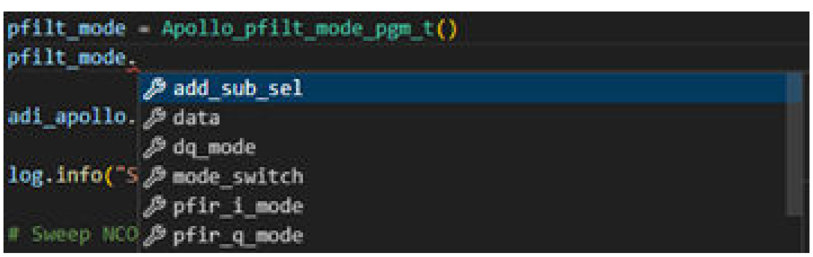


Figure 59. Members of pfilt_mode Struct

EVALUATION BOARD SOFTWARE SETUP

API AND THE API EXAMPLES

The MicroZed board on the ADS10-V1EBZ runs an Ubuntu Linux distro, which already includes a precompiled version of the API with which the **ACE Software** and the eRPC server communicate to issue hardware commands. Alternatively, the user can upload a particular API source C code that can be compiled directly on the target evaluation platform and executed to configure the evaluation board. The API package includes many example C files that allow the user to quickly execute code to confirm the hardware setup and to isolate potential issues related to the **ACE Software** or Python across its eRPC server.

Uploading the API to the Target (MicroZed Board)

Take the following steps to upload the API to the MicroZed board:

1. Download the most recent version of the API available to you from your **myAnalog** account. As of 12/23/2024, the most recent release is 0.4.58.
2. Initiate a new SSH session. The following example uses WinSCP because it offers a convenient GUI and includes a built-in SSH terminal:
   a. **Host Name: 192.168.0.10**
   b. **User name: root**
   c. **Password: analog**

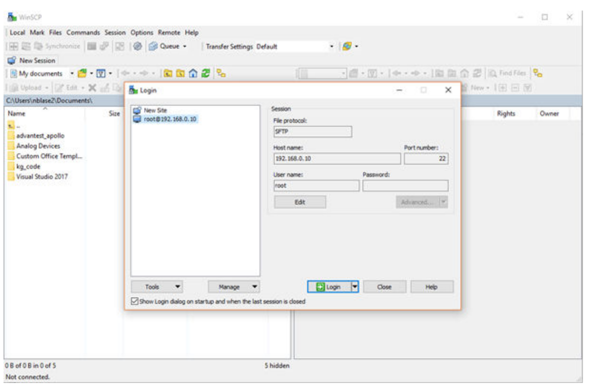


Figure 60. SSH Session Example

3. Once logged in, copy the API onto the target by dragging it over from your local machine.

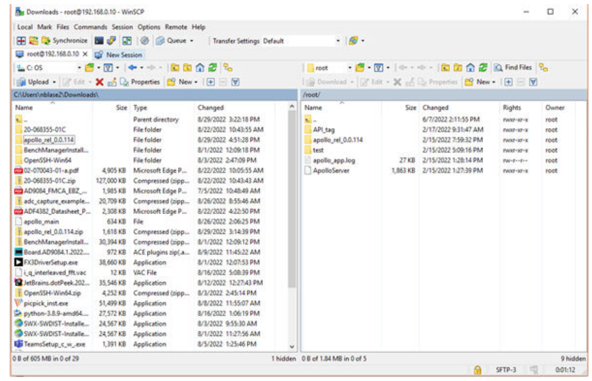


Figure 61. API Folder Transfer

4. Once copied, open a command prompt and SSH into the MicroZed board with the same login credentials as used previously. Note that the full command is **ssh root@192.168.0.10** (see Figure 62).

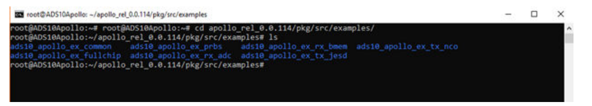


Figure 62. Command Prompt

5. Using standard Linux commands, navigate to the example code that you want to execute and compile the code using the **make clean all** command. The directory in which the code was compiled is the working directory from which the code can be compiled and executed.

Note that the Version A silicon (for example, A0 or A2) is not supported in the current and future software releases.

EVALUATION BOARD SOFTWARE SETUP

API Example: Ads10_apollo_ex_main Full Chip

Take the following steps to execute the **ads10_apollo_ex_main fullchip** API example from the **Examples** folder within the directory structure of the API package, which is loaded by using the steps described in the Uploading the API to the Target (MicroZed Board) section:

1. Navigate to the folder where the **main.c** file of the **ads10_apollo_ex_main** example is located and compile the example code by executing the Linux command **make clean all (B silicon is assumed in the compile, A silicon is no longer supported)**. For example, **cd~ /src/examples/ads10_apollo_ex_main** (or similar path).

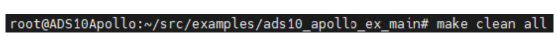


Figure 63. Linux Command to Make Clean All for the main.c File of the ads10_apollo_ex_main Example

2. Once the code is compiled, a debug folder is automatically created within the working directory, which includes the **apollo_main** executable. To run the **fullchip** example, use the **./debug/apollo_main fullchip id00_uc06_F** command, passing the example name (in this case, **fullchip**) and using the case name (in this case, **id00_uc06_F**) in the command line.

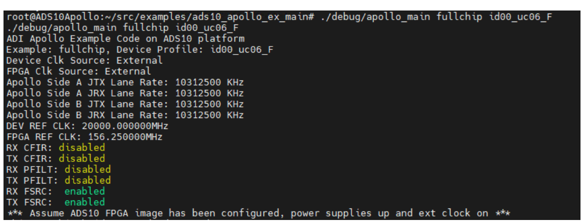


Figure 64. Example Command

Note that the **ads10_apollo_ex_main** directory contains many other examples and use cases. Typing the command **./debug/apollo_main fullchip** alone without passing any other parameters produces a list of available examples and use cases, along with command line options for specifying the evaluation board clock configuration options.

3. The full chip example establishes both the transmit and receive links to transmit a signal out of the DACs and to capture signals through the ADC. The signal captured by the ADC is saved as a list of integers (samples) in a **.txt** file, located in /**home/analog/Apollo**. If this directory is not already present on the MicroZed board microSD card, create it manually before running the API examples that store ADC samples. To transfer the captured samples to your PC for further analysis, open WinSCP and drag the files over (see Figure 65).

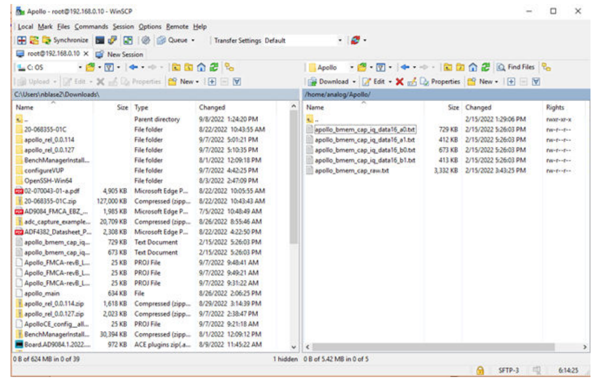


Figure 65. WinSCP Files Storage

4. At this point, the captured samples can be analyzed on the host PC, using the Visual Analog software or other software tools, such as MATLAB. A typical analysis step is to process the captured time-domain data using a FFT algorithm that converts the signal to its frequency-domain representation.

EVALUATION BOARD SOFTWARE SETUP

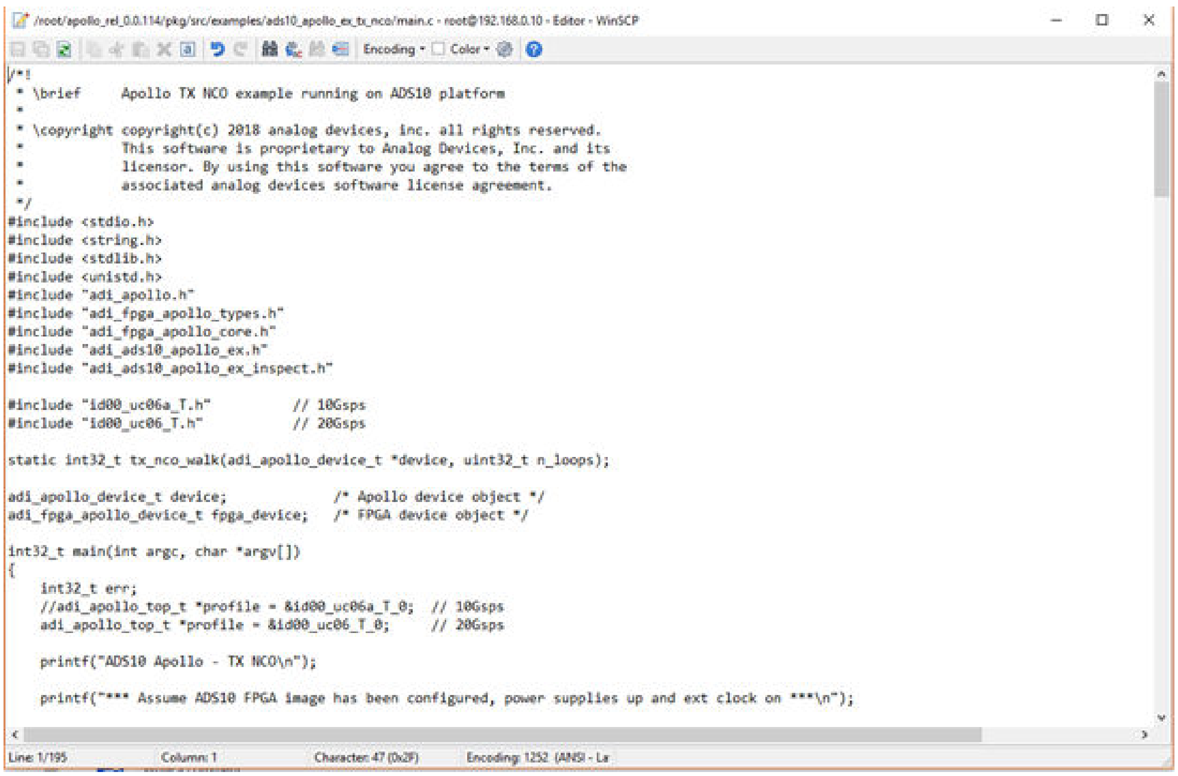
Modifying the API Examples

The API examples are a collection of **.c** files that can be modified using a simple text editor. There are multiple ways to make an edit to the code that can be grouped to two main methods: modifying the files directly on the MicroZed board using an SSH terminal or an SFTP client (such as WinSCP) or modifying the files on the local host and reloading the modified version to the MicroZed board. Either method works well, and the choice depends on the personal preference of the user.

While logged in to the MicroZed board, you can use the Vim text editor or another built-in Linux tool. You can also edit the files through WinSCP by right clicking the example code you want to edit and **selecting open with notepad** (or any other text editor), as shown in Figure 66.

After making changes, you must always recompile the code with the **make clean all** command, as describe in the API Example: Ads10_apollo_ex_main Full Chip section.

Note that it is convenient to open two SSH sessions with the MicroZed board: one session through WinSCP, to modify the C files and another session through a terminal to compile and execute the resulting code.



```
/*!
 * \brief     Apollo TX NCO example running on ADS10 platform
 *
 * \copyright copyright(c) 2018 analog devices, inc. all rights reserved.
 *            This software is proprietary to Analog Devices, Inc. and its
 *            licensor. By using this software you agree to the terms of the
 *            associated analog devices software license agreement.
 */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include "adi_apollo.h"
#include "adi_fpga_apollo_types.h"
#include "adi_fpga_apollo_core.h"
#include "adi_ads10_apollo_ex.h"
#include "adi_ads10_apollo_ex_inspect.h"

#include "id00_uc06a_T.h"         // 10Gsps
#include "id00_uc06_T.h"          // 20Gsps

static int32_t tx_nco_walk(adi_apollo_device_t *device, uint32_t n_loops);

adi_apollo_device_t device;             /* Apollo device object */
adi_fpga_apollo_device_t fpga_device;   /* FPGA device object */

int32_t main(int argc, char *argv[])
{
    int32_t err;
    //adi_apollo_top_t *profile = &id00_uc06a_T_0;  // 10Gsps
    adi_apollo_top_t *profile = &id00_uc06_T_0;     // 20Gsps

    printf("ADS10 Apollo - TX NCO\n");

    printf("*** Assume ADS10 FPGA image has been configured, power supplies up and ext clock on ***\n");
```

068

Figure 66. Opening the main.c File of the ads10_apollo_ex_tx_nco API Example Using WinSCP and the Native Text Editor on the Host PC

EVALUATION BOARD SOFTWARE SETUP

Notes on Recompiling (Optional Procedure)

Note that running the **make clean all** command for every recompile takes several minutes. To reduce the time required to recompile after making a localized API change, the time setting on the MicroZed board clock must be changed to be close to the time setting on the PC clock. To accomplish this, take the following these steps:

1. On the PC running the Apollo MxFE evaluation software, open a **cmd** window and type **echo %date% %time**, which results in the following format: Mon 03/04/2024 17:41:53.09.
2. Open an SSH session on the MicroZed board and login as described in the Checking the Host PC Can Connect to the MicroZed Board section. As an alternative to Step 1, the date can be typed in the SSH window, which produces a result in the following format: **Mon Mar 4 17:41:53 EDT 2024**. This result will likely show an incorrect time and date because the MicroZed board clock is not yet synchronized.
3. To set the MicroZed board clock, do the following:
    a. If using the Step 1 format, first, type **timedatectl set-timezone America/New_York** to set the MicroZed board clock timezone to United States Eastern time. Note that a list of available timezones can be obtained by typing **timedatectl list-timezones**. Then, type **date -s "Mon 03/04/2024 17:41:53.09"**" using the actual time produced in Step 1 to set the MicroZed board clock assuming the timezone is on the previous line.
    b. If using the format from Step 2, type **date -s "Mon Mar 4 17:41:53 EDT 2024**" using the actual time and time zone, which sets the MicroZed board clock including the time zone.
4. Compile a clean version, date stamped with the new date from Step 3 (the GCC compiler uses this date to track whether files were modified and need to be recompiled) by using the **make clean all** command.
5. When modifying the example code, GCC can recompile only files that must be recompiled by using the **make all** command.

The MicroZed board clock information is erased at power down; therefore, this optional procedure must be redone after every power cycle of the MicroZed board if a faster API compile capability is required.

EVALUATION BOARD SOFTWARE SETUP

Post Processing Captured ADC Samples in VisualAnalog

While the AD9084 ADC calibration algorithm evaluation is in progress, the ADC FFT shown in the **ACE Software** can include all interleaving spurs in its noise calculation, and therefore, misrepresent the true NSD performance of the AD9084. Because the **ACE Software** is continuously improving and the device calibration algorithms are optimized, it may be required by the user to post process the ADC data outside of the **ACE Software**.

Whether the ADC samples are captured using the **ACE Software** or directly through the API, the **VisualAnalog** software can run an FFT on the ADC samples to accurately represent the actual ADC performance of the AD9084.

Note that the SDP drivers in **VisualAnalog** overwrite the SDP drivers installed by the **ACE Software**. If installing Visual Analog for the first time, reinstall the AD9084 **ACE Software** to overwrite the SDP drivers as follows:

- If captured from the **ACE Software**, **Export** the data and store it in the following location: **C:\Users\adiguest\Desktop\saved_data**. Ensure that you note the file name under which the data is saved (see Figure 67).

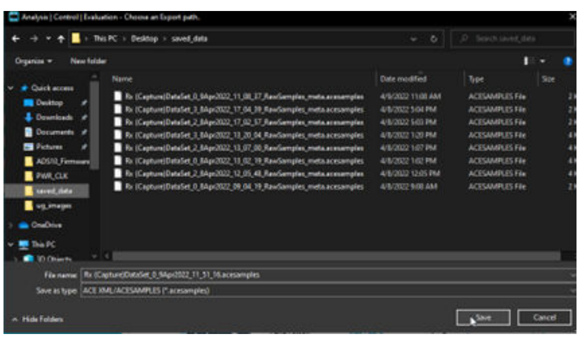


Figure 67. Selecting FFT in Analysis Window

- If captured directly by the API, copy the data file from the MicroZed board to the local host, as described in the final steps of the procedure in the API Example: Ads10_apollo_ex_main Full Chip section.
- Open **VisualAnalog** (see Figure 68).

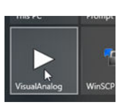


Figure 68. Opening VisualAnalog

- Open the **fft_analysis.vac** canvas from the Desktop. The canvas may also be opened from the **Recent** tab in the **New Canvas** window when **VisualAnalog** opens (see Figure 69).

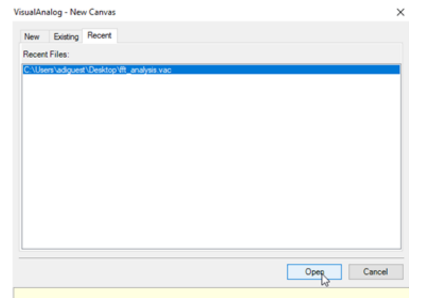


Figure 69. Opening the VisualAnalog - New Canvas

- Select the file previously saved in the **Pattern Loader** in the **VisualAnalog** canvas (see Figure 70).

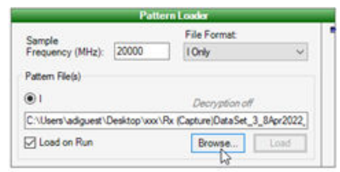


Figure 70. Loading the Saved Data in Pattern Loader in the VisualAnalog Canvas

- Ensure that **All files (*.*)** is selected in the open file dialog and select the file in **.csv** format that was saved from the **ACE Software**.
- Click **Open**.
- Click **Run** in **VisualAnalog** (see Figure 71).

Figure 71. Running the VisualAnalog Canvas

- The captured data post processes and displays in the **VisualAnalog** canvas as shown in Figure 72.

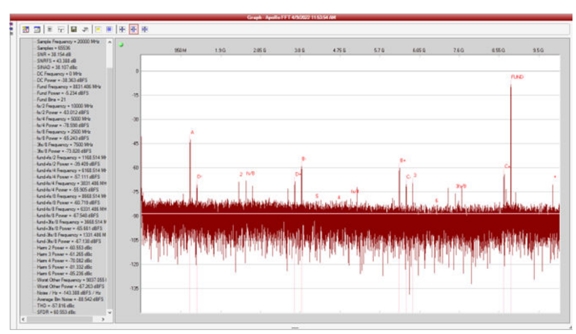


Figure 72. Selecting the Saved Data in Pattern Loader in the VisualAnalog Canvas

NOTES

ESD Caution

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.