User's Guide

mmWave Studio GUI

This document outlines the mmWaveStudio Graphical User Interface details and instructions for software start up. Operating procedures for Radar API and Post-Processing utility is also explained briefly in this user's guide.

Contents

List of Figures 3

1. Introduction 6

2. mmWaveStudio Installation and Startup 6

2.1 Installation 6
2.2 Startup 7
2.3 USB Interface and Drivers 8

3. Using the DCA1000 EVM 11

3.1 Using DCA1000 CLI utility 11
3.2 Updating the DCA1000 FPGA Binary 11
3.3 Raw ADC data capture 11

4. mmWaveStudio User Interface 13

4.1 Menu Bar 13
4.1.1 File Menu 13
4.1.2 View Menu 14
4.1.3 Tools Menu 14
4.1.4 ToolBars Menu 14
4.1.5 Window Menu 16
4.1.6 Help Menu 16
4.2 Radar API Window 16
4.3 Output Window 17
4.4 LUA Shell 18

5. Automation/Scripting 19

6. Radar API Tab Operations 19

6.1 Connection Tab 19
6.1.1 Board Control Operations 21
6.1.2 RS232 Operations 22
6.1.3 Firmware Download 22
6.1.4 SPI Connection & RF Power Up 24

7. Static Config Tab Operations 24

8. Data Config Tab Operations 25

9. Sensor Config Tab Operations 26

9.1 Profile Config 26
9.1.1 Profile Manager 26

9.2 Chirp Config 27
9.2.1 Chirp manager 27
9.3 Frame Config 28
10. RegOp Tab 32
10.1 Register Operations Tab 33
10.2 Debug Signals 33
10.3 GPIO_0 33
11. Continuous Streaming Tab Operations 34
12. BPM Config Tab Operations 35
13. Event Monitor Tab 36
14. Advanced Frame Config Tab operations 37
15. Rampgen Timing Calculator Tab 38
16. Loopback Tab Operations 39
17. CalibConfig Tab Operations 40
17.1 Time Unit Config 40
17.2 RF Init Calibration Config 40
17.3 RunTime Calibration and Trigger Config 40
17.4 Time Unit Failure report 41
17.5 RF Init Calibration Summary Report 41
17.6 Runtime Calibration Report 41
18. Monitoring 41
18.1 Digital monitoring 41
18.2 Analog monitoring enables 42
18.3 TX monitoring 42
18.4 RX monitoring 43
18.5 DCBIST monitoring 44
18.6 Async Event Report Format 45
18.6.1 Logging of Async events 45
18.6.2 Decoding the Reports 47
18.6.3 Example of Decoding the Reports 48
19. RX and TX gain LUT 50
20. Import / Export Tab Operations 52
20.1 GUI (Graphical User Interface) 52
20.1.1 Capture Setup file 52
20.1.2 mmWave Configuration File 53
20.1.3 Types of Waveform 53
20.2 Capture Setup JSON Operations 53
20.2.1 Import 53
20.2.2 Setup 54
20.3 mmWave Device JSON Operations 54
20.3.1 Import 54
20.3.2 Load 55
20.3.3 Configure Device 55
20.4 Capturing Raw ADC Data 56
20.5 Export 56
20.6 LUA API's for JSON Parser 58
20.6.1 Capture Setup API's 58

20.6.2 mmWave Configuration API's 58

21. Radar post processing 59

21.1 List of plots 60
21.1.1 Basic plots 60
21.1.2 CQ plots (chirp quality metrics) 63
21.1.3 Characterization plots 63
21.2 Channel select 64
21.3 Frame /Profile /Chirp sliders 65
21.4 Information sidebar 65
21.4.1 Programmed parameters 66
21.4.2 Calculated parameters 66
21.4.3 Detection results 67
21.5 Play 67
21.6 Miscellaneous options 67
21.6.1 Detection 67
21.6.2 FFT processing settings 68
21.6.3 Miscellaneous 68
21.6.4 Time domain options 69
21.6.5 More miscellaneous options 69
21.7 Notes on Setting up PostProc for Characterization 69
21.8 Standalone processing (post processing done in a different session from configuration and capture) 70

22. Controlling mmWaveStudio from Matlab 71

23. Automation using LUA 73

23.1 Sample Lua script for automation 74

24. Format of the raw captured file 75

24.1 Preliminaries 75
24.2 Notation 75
24.3 Sample Format 76
24.4 TSW1400 EVM xWR16xx file formats 76
24.4.1 2 LVDS Lanes, complex data, variable number of channels, chirping mode 76
24.4.2 2 LVDS Lanes, real data, variable number of channels, chirping mode 76
24.4.3 2 LVDS Lanes, complex data, variable number of channels, continuous streaming mode 77
24.5 TSW1400 xWR12xx/xWR14xx file format 78
24.5.1 $\boldsymbol{n}$ LVDS Lanes, complex data, $\boldsymbol{n}$ channels, chirping/continuous streaming mode 78
24.5.2 $\boldsymbol{n}$ LVDS Lanes, real data, $\boldsymbol{n}$ channels, chirping/continuous streaming mode 79
24.6 *DCA1000 EVM capture format (xWR12xx/xWR14xx complex, 4 channel, 4 lanes [Interleaved])* 79
24.7 *DCA1000 EVM capture format (xWR12xx/xWR14xx real, 4 channel, 4 lanes [Interleaved])* ... 79
24.8 *DCA1000 EVM capture format (xWR16xx complex, 4 channel, 2 lanes [Non-Interleaved])* 80
24.9 *DCA1000 EVM capture format (xWR16xx real, 4 channel, 2 lanes [Non-Interleaved])* 80

List of Figures

Figure 2.1. mmWaveStudio startup error message 8
Figure 2.2. Device Manager (Other Devices) 9
Figure 2.3 FTDI driver update 10
Figure 2.4. Device Manager (USB Serial Port) 10
Figure 2.5. MMWAVE-DEVPACK or DCA1000 COM Ports 10
Figure 2.6. XWR1xx Dev Pack and EVM USB Enumeration 11

Figure 4.1. mmWaveStudio Main Window 13
Figure 4.2. Config ToolBar 14
Figure 4.3. Edit Button 15
Figure 4.4. Button Config 15
Figure 4.5. Edit User Defined Script 15
Figure 4.6. Radar API Window 17
Figure 4.7. Output Window Docking 17
Figure 4.8. Output Window Log 18
Figure 4.9. LUA Shell window 18
Figure 5.1. Command creation 19
Figure 5.2. LUA Script execution 19
Figure 6.1. Connect Tab 20
Figure 6.2. SOP Control 22
Figure 6.3. Serial Port Control and selecting the COM port for RS232 operations 22
Figure 6.4. Firmware Download 23
Figure 6.5. Firmware Download Progress 23
Figure 7.1. Static Config Tab 25
Figure 8.1. Data Config Tab 26
Figure 9.1. Profile Manager Tab 27
Figure 9.2. Chirp Manager Tab 28
Figure 9.3. Sensor Config Tab 29
Figure 9.4. IP address configuration in case of DCA1000 EVM 30
Figure 9.5. Setup DCA1000 configuration button from Connection tab 30
Figure 9.6. DCA1000 configuration window 31
Figure 9.7. Post Processing in Matlab 32
Figure 10.1. Register Read/Write 33
Figure 10.2. GPIO_0 option selection box 34
Figure 11.1. Continuous Streaming Tab 35
Figure 12.1. BPM Config Tab 36
Figure 13.1. Event Monitor Tab 37
Figure 14.1. Advanced frame config tab 38
Figure 15.1. Rampgen Timing Calculator 39
Figure 16.1. Loopback Config Tab 39
Figure 17.1. CalibConfig Tab 40
Figure 18.1. Digital monitoring configuration 42
Figure 18.2. RF Analog monitoring consolidated enables 42
Figure 18.3. Analog TX monitoring configuration 43
Figure 18.4. Analog RX monitoring configuration 44
Figure 18.5. DCBIST monitor 45
Figure 19.1. RX and TX LUT (read and write) 51
Figure 20.1 Import/Export GUI 52
Figure 20.2 Capture Import 54
Figure 20.3 mmWave Import 55
Figure 20.4 Capture and Post Processing 56
Figure 20.5 Export Capture File 57
Figure 20.6 Export mmWave File 57
Figure 21.1. PostProc Basic 59
Figure 21.2. 2D FFT Profile 60
Figure 21.3. Range Angle plot 61

Figure 21.4. Detection and Angle Estimation results 61
Figure 21.5. Chirp Config Picture 62
Figure 21.6. 1D FFT Profile 62
Figure 21.7. Time domain plot 63
Figure 21.8. Phase Stability 63
Figure 21.9. Amplitude stability 64
Figure 21.10. Zero-velocity bin vs High velocity bin. 64
Figure 21.11. Channel select 64
Figure 21.12. Sliders 65
Figure 21.13. Information sidebar 66
Figure 21.14. Play 67
Figure 21.15. Miscellaneous options 67

1. Introduction

The mmWaveStudio GUI is designed to characterize and evaluate the TI Radar devices. The mmWave device is configured and controlled from the mmWaveStudio by sending commands to the device over SPI. ADC data is captured using DCA1000 EVM or the TSW1400 EVM board and the data is processed in Matlab and the results are displayed in the GUI.

mmWaveStudio GUI utilizes C DLL and a set of API's to communicate from the GUI to the device through FTDI FT4232H device. The FT4232H is a USB 2.0 Hi-Speed (480 Mb/s) to UART IC. It has the capability of being configured in a variety of industry standard serial or parallel interfaces. The FT4232H features 4 UARTs. Two of these have an option to independently configure an MPSSE engine; this allows the FT4232H to operate as two UART/bit-bang ports plus two MPSSE engines used to emulate JTAG, SPI, I2C, bit-bang or other synchronous serial modes.

Key features of the mmWaveStudio GUI are

- Board Control (SOP Change, Reset Control)
- RS232 connection to device
- Firmware download over the RS232 interface
- Configuring the TI Radar device using the Radar API commands
- Interaction with DCA1000 EVM or TSW1400 EVM for raw ADC data capture
- Post-Processing of ADC data and visualization of the processed data

Refer to the TSW140x High Speed Data Capture/Pattern Generator Card User's Guide (SLWU079) for more information regarding the usage aspect of TSW1400 EVM and the related Software

Refer to DCA1000 EVM Capture Card User's Guide (SPRUIJ4) for more information regarding the usage aspect of DCA1000 EVM.

2. mmWaveStudio Installation and Startup

2.1 Installation

The following software should be installed before starting the mmWaveStudio

1. Install mmWaveStudio from the installer package
2. Install 32-bit Matlab Runtime Engine (Version 8.5.1): It is used to run the Post-Processing utility within mmWaveStudio.
   https://in.mathworks.com/supportfiles/downloads/R2015a/deployment_files/R2015aSP1/installers/win32/MCR_R2015aSP1_win32_installer.exe

> **NOTE:** Please make sure the Matlab Runtime Engine installed is exactly same as 32-bit Version 8.5.1

3. If required (read below note), install FTDI Drivers: FTDI USB Driver (mmwave_studio_<ver>\mmWaveStudio\ftdi) necessary to work with Radar device is installed. See section 2.3 for FTDI driver installation.

> **NOTE:** FTDI drivers will be installed automatically at the end of mmwavestudio installation. The step 3 is only required if the automatic FTDI installation fails.

4. Install Microsoft Visual C++ 2013 Redistributable package if using a Windows 10 machine from the link https://support.microsoft.com/en-us/help/3179560
5. Install following software if you are using TSW1400 EVM
   a. Install HSDC Pro Software (for TSW1400 EVM only): It is used to work with TSW1400 EVM http://www.ti.com/tool/dataconverterpro-sw

> **NOTE:** HSDC Pro Software should be installed in Administrator Mode

   b. Install HSDC Pro radar device specific files
      i. If you are working with a single XWR1243 device
         - Copy ADC_FIRMWARE.rbf from mmwave_studio_<ver>\ mmWaveStudio\HSDCProFiles to '1400 details\Firmware' folder in the HSDCPro installation. Don't worry if the existing files are overwritten.
         - Copy AWR12xx_lvds_4Channel_ddr_4bit_par_centre_xx_bit.ini files from mmwave_studio_<ver>\ mmWaveStudio\HSDCProFiles to '1400 details\ADC files\' folder in the HSDCPro installation
      ii. If you are working with XWR1642 device
         - Copy AWR1642_FIRMWARE.rbf from mmwave_studio_<ver>\ mmWaveStudio\HSDCProFiles to '1400 details\Firmware' folder in the HSDCPro installation. Don't worry if the existing files are overwritten.
         - Copy AWR16xx_lvds_4Channel_ddr_4bit_par_centre_xx_bit.ini files from mmwave_studio_<ver>\mmWaveStudio\HSDCProFiles to '1400 details\ADC files\' folder in the HSDCPro installation
   c. Copy TSW1400_IID_Lookup.csv from mmwave_studio_<ver>\ mmWaveStudio\HSDCProFiles to '1400 details\' folder in the HSDCPro installation

2.2 Startup

1. After the installation is complete, the GUI executable and associated files will reside in the following directory: C:\ti\mmwave_studio_<ver>\mmWaveStudio
2. Power up the xWR1xx DevPack (or DCA1000 EVM) and the xWR1xxx BOOST-EVM.

> **NOTE:** Make sure the above combination of DevPack/DCA1000 and AWR BOOST-EVM are

connected to the PC while opening mmWaveStudio for the first time.

3. To start the GUI, click on the file called “mmWaveStudio.exe", located under C:\ti\mmwave_studio_<ver>\mmWaveStudio\RunTime folder.

NOTE: mmWave Studio should to be started in Administrator Mode

NOTE: If mmWave Studio shortcut has been created already, update the shortcut to the new installation path.

NOTE: If you see the error message as shown in FIGURE 1 during opening mmWaveStudio in its output window, follow the steps mentioned below to solve this

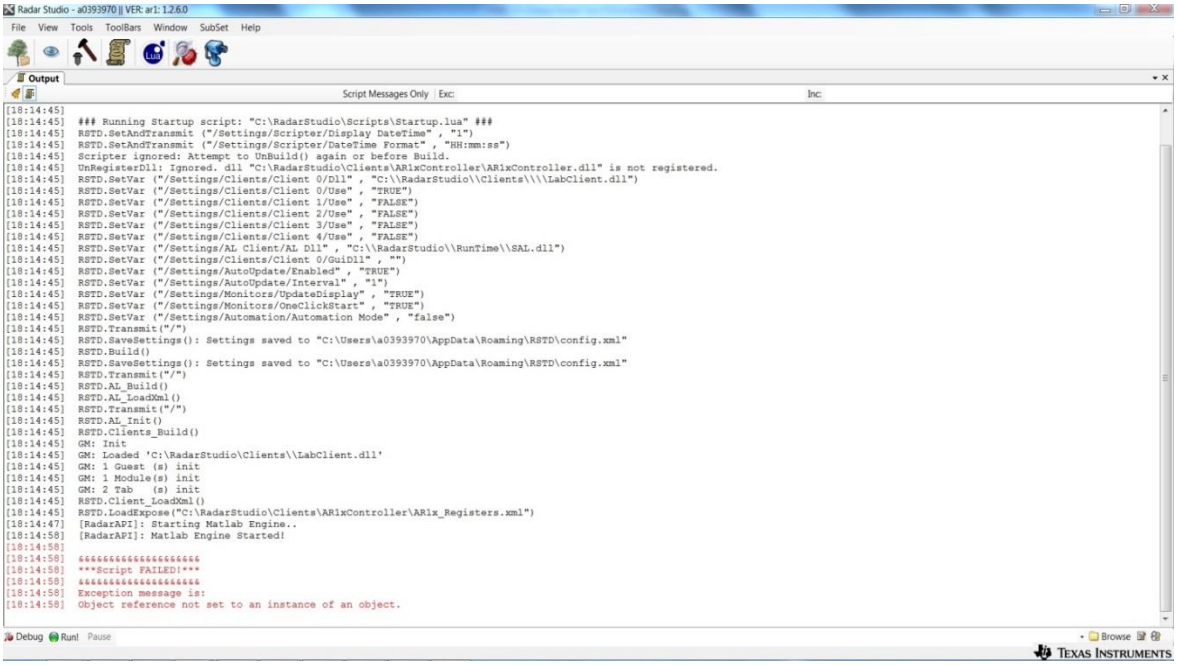


Figure 2.1. mmWaveStudio startup error message

a. Uninstall HSDC Pro

b. Install it again while TSW1400 board is connected through USB

c. Re-launch the mmWaveStudio GUI

2.3 USB Interface and Drivers

Once the USB is connected to the XWR1xx Dev Pack (or DCA1000 EVM), ensure to install the FTDI USB Drivers (mmwave_studio_<ver>\ftdi). After completion of the driver

installation, COM ports will be available in Windows Device Manager as shown in FIGURE 2.6.

Once xWR1xxx EVM is connected to the MMWAVE-DEVPACK or DCA1000 EVM, connect DevPack or DCA1000 to PC using the USB cable provided and connect the power cable. Once done, there should be 4 additional COM Ports as shown in Figure 2.5

When the DevPack or DCA1000 EVM is connected for the first time to the PC, Windows maybe not be able to recognize the device and would come up as 'Other devices' in device manager as shown in Figure 2.2

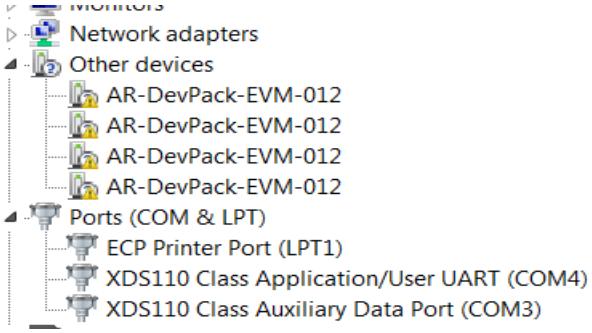


Figure 2.2. Device Manager (Other Devices)

In Windows device manager, right-click on these devices and update the drivers by pointing to the location of the FTDI driver as show in Figure 2.3

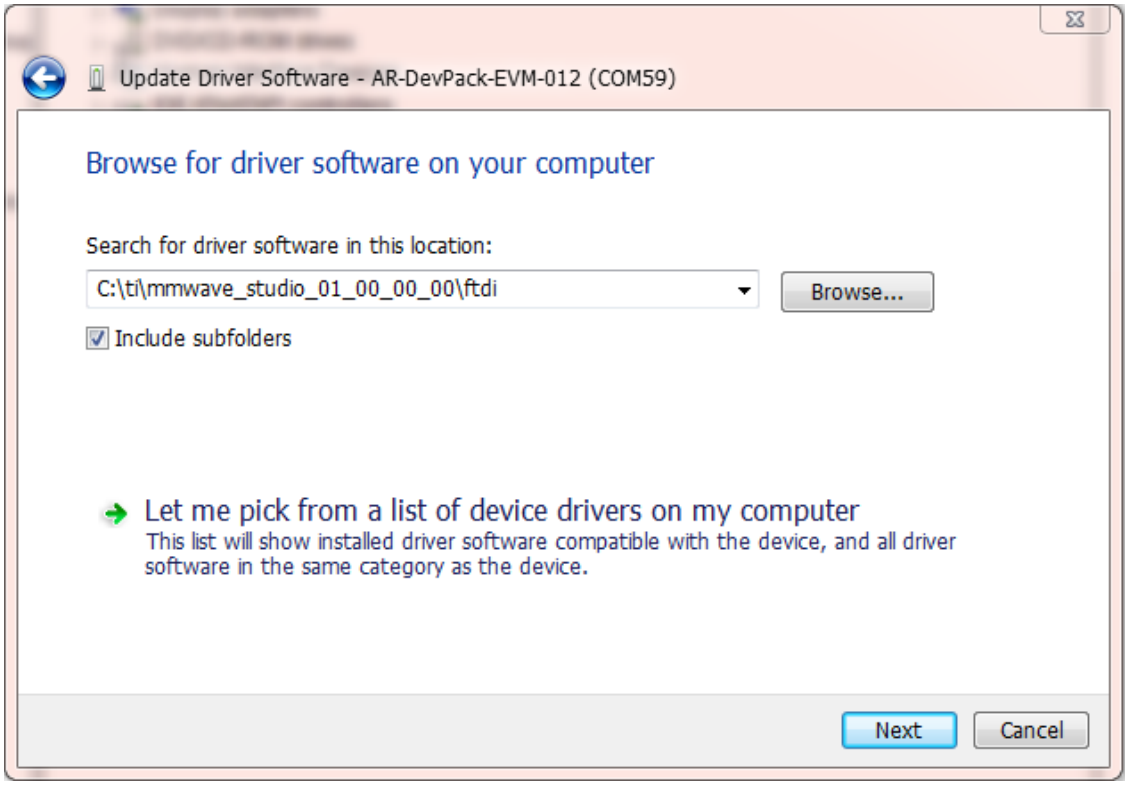


Figure 2.3 FTDI driver update

This must be done for all four COM ports. If after updating the FTDI driver, device manager still doesn't show 4 new COM Ports, as shown in Figure 2.4, you would need to update the FTDI driver once again.

Figure 2.4. Device Manager (USB Serial Port)

When all four COM ports are installed, the device manager recognizes these devices and indicates the COM port numbers, as shown in Figure 2.5

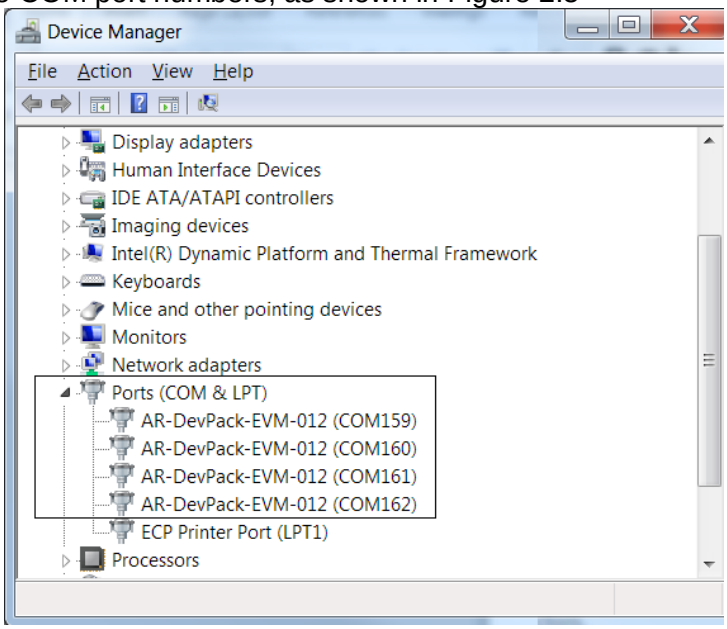


Figure 2.5. MMWAVE-DEVPACK or DCA1000 COM Ports

Next connect the USB cable from the XWR1xxx EVM to the PC. 2 COM ports will be enumerated with name XDS110. The mmWaveStudio should be connected to COM port numbered alongside the name XDS110 Class Application/User UART. In the following example, it is COM11. For more details, please refer to XWR12/XWR14/XWR16 EVM User Guide.

NOTE: To update XDS110 USB driver, download and install XDS emulator software from this link [http://processors.wiki.ti.com/index.php/XDS_Emulation_Software_Package]

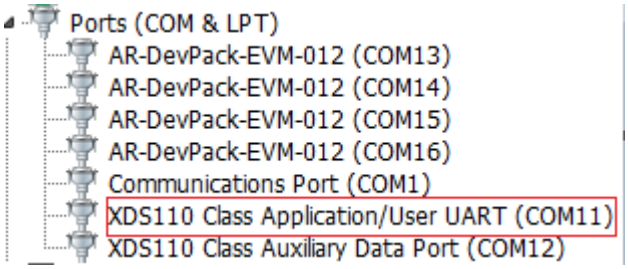


Figure 2.6. XWR1xx Dev Pack and EVM USB Enumeration

3. Using the DCA1000 EVM

The DCA1000 EVM receives the LVDS data from XWR1xxx device and transfers the raw ADC data to the target PC via the Ethernet interface. The default destination Ethernet port address configured in the DCA1000 EVM is 192.168.33.30 and socket number is 4098. User has to ensure that target PC is set with this IP address to receive the raw ADC data from the DCA1000 EVM.

NOTE: System Firewall on Host machine can sometime prevent the Ethernet connection to DCA1000 board. Ensure that the firewall allows the Ethernet port access to mmWavestudio.

3.1 Using DCA1000 CLI utility

The latest version of Studio contains a DCA1000 CLI utility, which allows controlling the DCA1000EVM over command line (without using the mmWaveStudio GUI). Internally, mmWaveStudio GUI uses the same utility for all communication with DCA1000EVM.

For instructions on using the DCA1000 CLI, refer Section 3 of the user guide present at *"ReferenceCode\DCA1000\Docs\TI_DCA1000EVM_CLI_Software_UserGuide.pdf"*

3.2 Updating the DCA1000 FPGA Binary

The latest version of Studio contains a FPGA binary (version 2.8) present at "*PlatformBinaries\DCA1000FPGA"* folder.

Instructions on flashing the FPGA binary onto the DCA1000 can be found at (Section 9 from the document https://www.ti.com/lit/ug/spruij4a/spruij4a.pdf)

3.3 Raw ADC data capture

The LVDS data captured by the DCA1000 EVM is packetized and transferred over the Ethernet interface as UDP datagrams. These UDP datagrams are received by the target PC and it is written into file on the target PC. These UDP data grams contains meta data like packet sequence number, number of data bytes received until now which helps in determining if there were any packets which were not received in order or if there were any packets which were dropped.

The file name to store the raw ADC data is given by the user in the SensorConfig Tab. The DCA1000 EVM appends “Raw_n” to the file name given by the user. After 1 GB of capture, the DCA1000 EVM will start capturing the data into another file. Each subsequent file will have the test “Raw_n” appended to the user given filename where n is a number starting from 0. For example, if the user given filename is adc_data.bin, after the capture, user will see adc_data_Raw_0.bin, adc_data_Raw_1.bin etc. files in the mmWaveStudio\PostProc (default location of the raw ADC data files) folder.

NOTE: In the previous versions of mmWaveStudio, it was necessary to run Packet Reorder utility once the raw data is captured from DCA1000 before post processing of the data using Matlab. In this version and in subsequent versions of mmWaveStudio, it is not necessary to run the Packet Reorder utility as the data obtained from the DCA1000 is already ordered and can directly be used for post processing.

4. mmWaveStudio User Interface

Invoke mmWaveStudio.exe from C:\ti\mmwave_studio_<ver>\ mmWaveStudio\RunTime\mmWaveStudio.exe. When the mmWaveStudio GUI software is started, the initial setup screen appears and the main window appears as shown in FIGURE 4.1.

The GUI version is reported in the upper left corner of the GUI.

> **NOTE:** If Matlab RunTime 8.5.1 is not installed prior to mmWave Studio invocation, error will be displayed saying Matlab Runtime engine is not installed

The mmWave Studio Main window has the following sections:

- Radar API Window
- Output Window

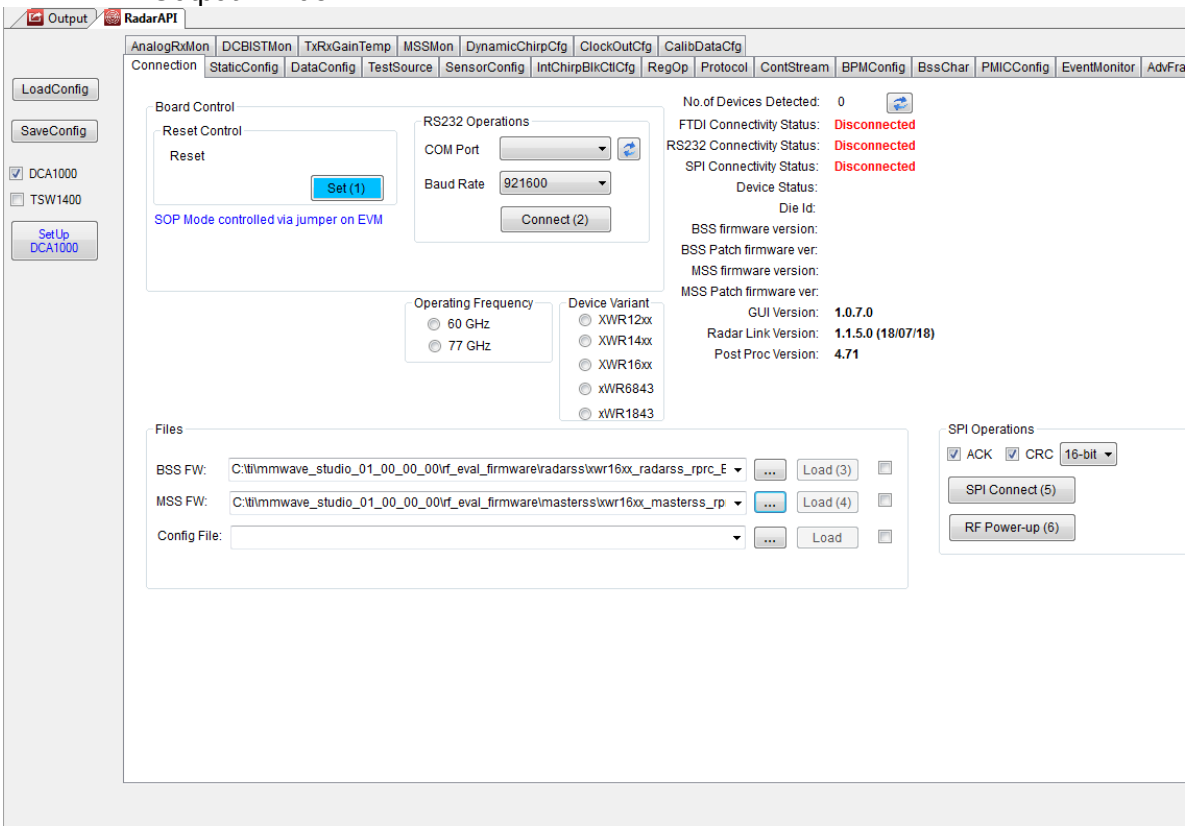


Figure 4.1. mmWaveStudio Main Window

Each of the above sections is described in detail in the forthcoming sections.

4.1 Menu Bar

The mmWaveStudio Main window has the following options in the Menu bar

4.1.1 File Menu

Exit: Exits the application and saves the setup (all current API configurations are stored locally in the PC which can be loaded back again).
Exit without saving layout: Exits without saving the current application setup.

4.1.2 View Menu

- The View menu opens the output window and LUA Shell window
- It also shows and hides toolbars and the status bar.

4.1.3 Tools Menu

Lock layout: Locks and unlocks the current layout. Once the layout is locked, the windows cannot be moved around.

Register DLLs to LUA: Using this option, any module developed in either C/LUA/C# can be loaded as a DLL. The APIs available (exposed) in the module can be called though LUA Shell. Refer to MSDN and LUA Help for creating libraries to be callable by LUA.

4.1.4 ToolBars Menu

Using this option, user can create shortcut buttons to associate frequent used actions in the form of LUA scripts. The below procedure details creating user defined button using the ToolBars menu.

1. Click *New.* The following dialog appears as shown in

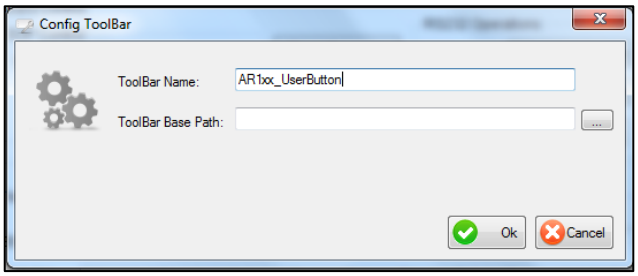


Figure 4.2. Config ToolBar

ToolBar Name – Sets the name for the ToolBar.

ToolBar Base Path – (optional) Defines a base path to search for the scripts which will be called via the toolbar's buttons.

> **NOTE:** Each Toolbar's configuration is saved in an XML file under %AppData%\RSTD\ToolBars folder

2. Set the ToolBar Name and click the OK button.
3. Click on the <new> button in the newly created toolbar.

Figure 4.3. Edit Button

4. This opens the User Button Configuration window

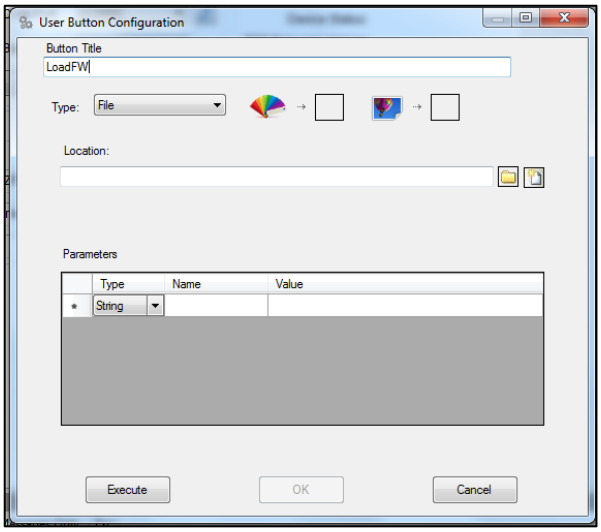


Figure 4.4. Button Config

5. Fill in the **Button Title** with a name for the button, and **Location** with the full path of the script you wish to run and click the **OK** button.
6. Clicking on the newly created button will execute the script it references.
7. Right-clicking on it will open a menu dialog with several options.

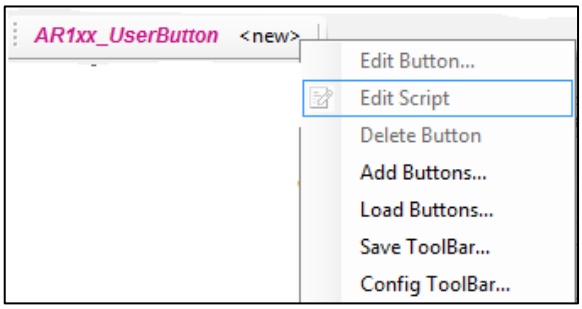


Figure 4.5. Edit User Defined Script

- **Edit Button** - Change the button's settings.
- **Edit Script** – Open the referenced script in your default text editor.
- **Debug Script** – Open the referenced script in the text editor.
- **Delete button** – Delete the user button.
- **Add Buttons** – Add buttons from another toolbar configuration file.

- **Load Buttons** – Load buttons from another toolbar configuration (clears existing buttons)
- **Save Toolbar** – Save the toolbar configuration to selected location (in xml format).
- **Config Toolbar** – Change the toolbar settings.

4.1.5 Window Menu

The Window menu shows open windows in the application.

4.1.6 Help Menu

The Help menu shows information about the current version of the application.

4.2 Radar API Window

As soon as the mmWaveStudio GUI is opened, the Radar API window appears as shown in FIGURE 4.6. Radar API Window is the primary tab through which the mmWave device's functionality can be verified. Also, the mmWave device is configured and controlled from the mmWaveStudio by sending commands to XWR1xxx device over SPI by interfacing through C DLL. Once the ADC data is captured using TSW1400 board, the post processing is done and plots are available in the GUI by interfacing with Matlab DLL.

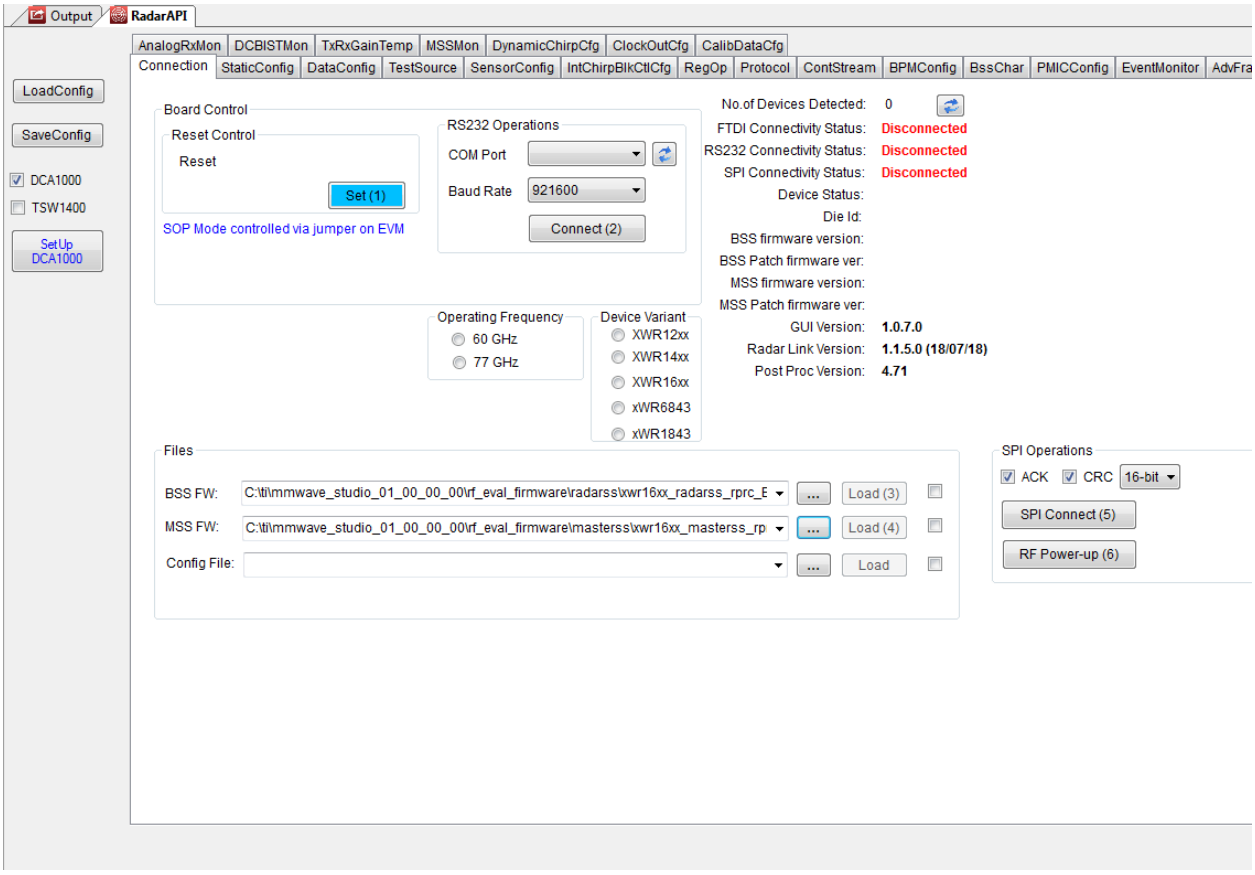


Figure 4.6. Radar API Window

In the Radar API window, the following tabs are available for communication with the Radar Device:

1. Connect
2. Static Config
3. Data Config
4. Sensor Config
5. RegOp
6. ContStream
7. BPM Config
8. AdvFrame Config
9. Calib Config
10. RampTimingCalculator
11. LoopBack
12. Calib Config

4.3 Output Window

The mmWaveStudio GUI components are dockable. Click View from the Menu bar, select output option. The output window also appears along with the main window. Select the output tab and drag to the section where you wish to dock. The output window docking is shown in the FIGURE 4.7.

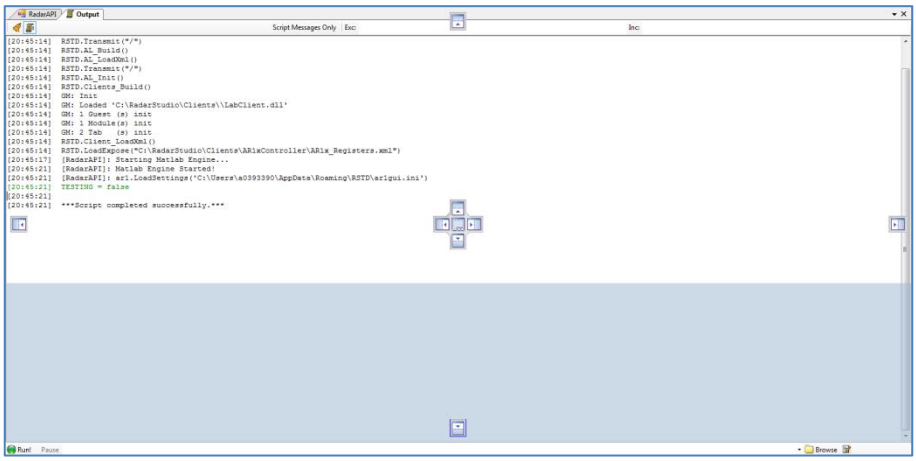


Figure 4.7. Output Window Docking

- The Output Window captures the user initiated actions along with timestamp.

- Both the logs and the script commands are available in the Output window.
- The script commands can be used to create custom LUA scripts and later run from the Run toolbar
- Also, upon an error or if the application is not responding, the error logs can be accessed by Right Clicking on the Output window.

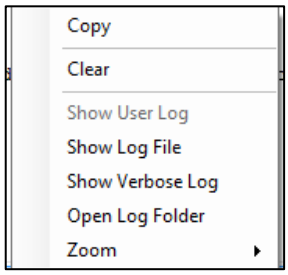


Figure 4.8. Output Window Log

4.4 LUA Shell

LUA Shell is used to execute LUA script commands, perform LUA operations. Click **View → LUA Shell** in the mmWaveStudio main window. The LUA Shell window appears as follows:

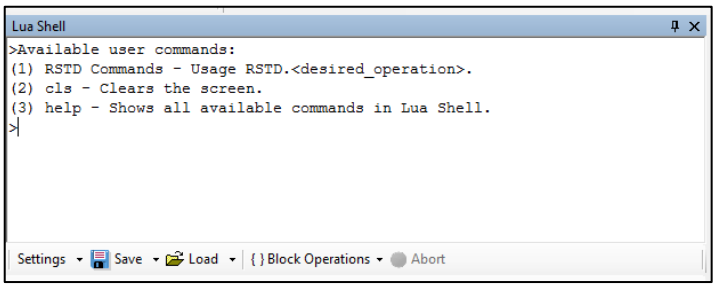


Figure 4.9. LUA Shell window

1. Running “help ar1” in the Lua Shell will list the functions contained under the ar1 module specific to Radar.
2. Running “help RSTD” in the Lua Shell will list the functions contained under the mmWaveStudio module.
3. Pressing Escape key - while the mouse cursor is on the last line deletes this line (last line only)
4. Running the ‘help’ command on each function name will display information on that function (e.g. “help ar1.Connect”)

NOTE: The LUA shell supports auto-completion with the tab key.

History command – Typing history command in the LUA Shell prompt shows all the commands being used by the user in current LUA Shell. By pressing Up-Arrow (↑) and Down-Arrow (↓), the recent commands can be accessed.

5. Automation/Scripting

The XWR1xx interface exposes functions to LUA, which can be used to test predefined sequences and automate functionality on the device. All these functions can be found under the ar1 module.

1. Whenever a command is issued in the GUI window, the equivalent script commands can be taken from the Output window and formed an automation script. For example, In the Static Config tab, when Channel and ADC Config command is sent, in the Output window, the script command is logged.

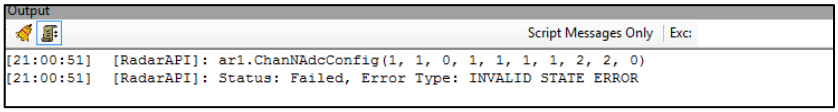


Figure 5.1. Command creation

2. The script commands start with ‘ar1’. The commands can be saved as a LUA file and used for automation.

3. To execute the LUA scripts, Browse and Select the script file and click Run

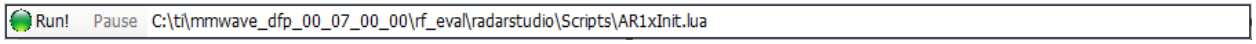


Figure 5.2. LUA Script execution

For more information on LUA Scripting language, please refer to:
http://lua-users.org/wiki/ - The Lua wiki (See the Lua Directory inside for Lua tutorials)
http://www.lua.org/manual/5.1/manual.html - Lua official Reference Manual

6. Radar API Tab Operations

The following sections briefly describe the operations in each tab of RadarAPI window

6.1 Connection Tab

When the mmWaveStudio is invoked, the Connect Tab appears as shown in FIGURE 6.1.

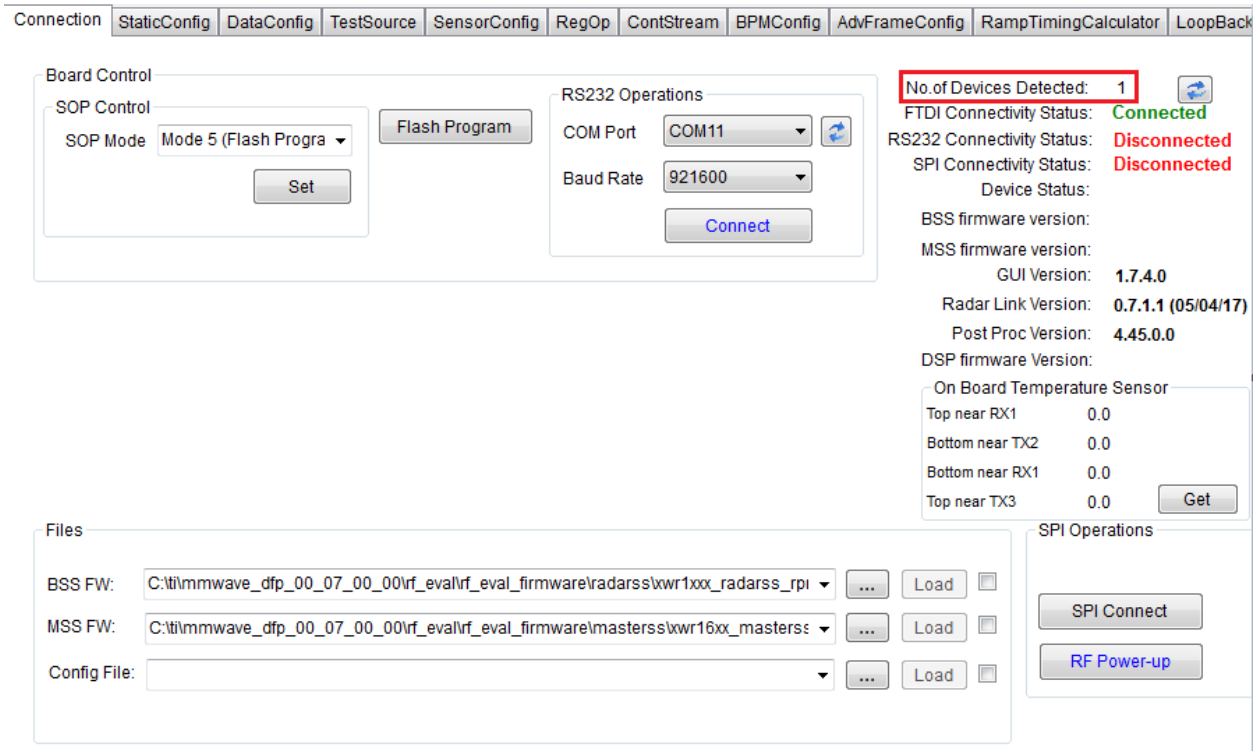


Figure 6.1. Connect Tab

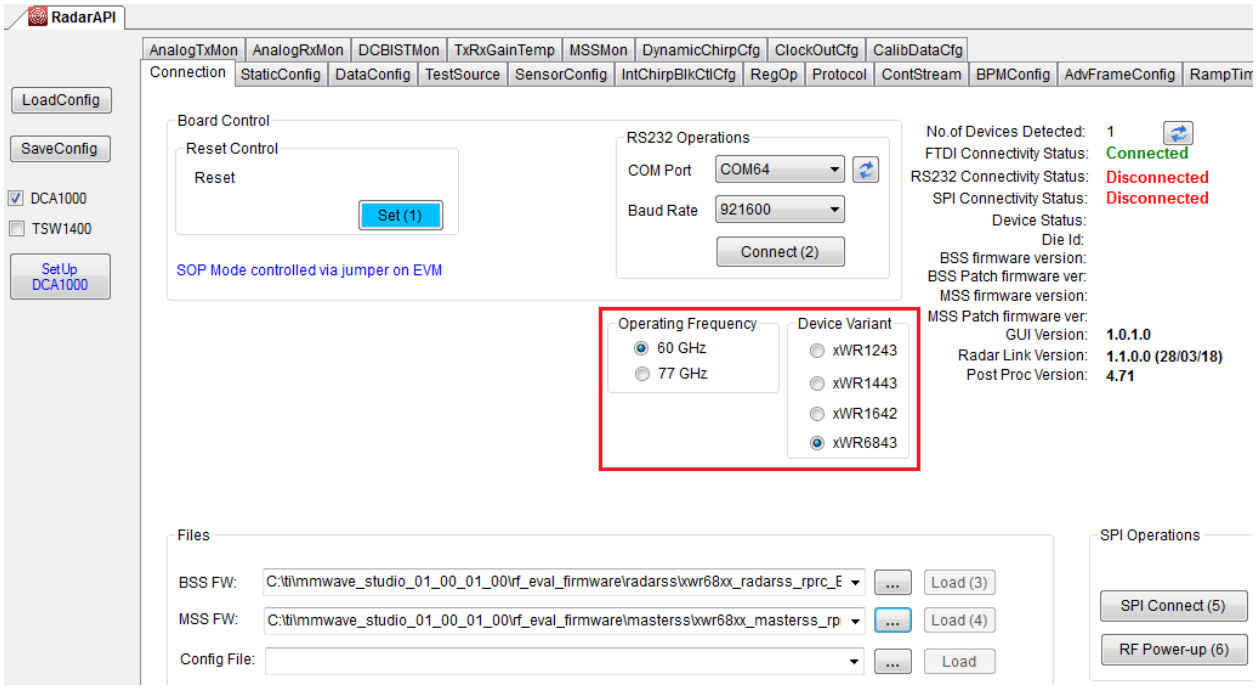
Using this tab, user can perform

- Board control operations
- RS232 Operations
- Connectivity Status and content versions
- Firmware Download
- SPI Connection

CAUTION:

Before starting with any action on mmWaveStudio please make sure that GUI detects the device connected to PC. Value of 'No of devices detected' must be 1 else click on till it detects the device.

For 60 GHz devices, the mmWaveStudio configuration should be as shown below. Select 60 GHz as operating frequency and select device variant as xWR6843.



6.1.1 Board Control Operations

mmWaveStudio does board control operation of XWR1xxx device through XDS110 port on the XWR1xxx EVM.

SOP Control

> **NOTE:** SOP control is only applicable when working with MMWAVE-DEVPACK. When using a DCA1000 EVM, use the jumpers on the XWR1xxx BOOST to control the SOP settings.

The XWR1xxx device implements a sense-on-power (SOP) scheme to determine the device operation mode. The device can be configured to power up in one of the three following modes:

- SOP2: Development Mode. This mode should be used for RF evaluation.
  - Set SOP Mode 2. For characterization and evaluation of XWR1xxx devices, always use SOP2 mode
  - Connect over RS232
  - Select the Operating frequency and Device variant if auto selection is incorrect
  - Perform MSS (masterss) & BSS (radarss) Firmware Download
  - Perform SPI Connect
  - Issue API to the device

Once the USB connections to the board is intact, Select the SOP Mode and Click Set button. For SOP setting to be effective, Reset is also performed by the GUI.

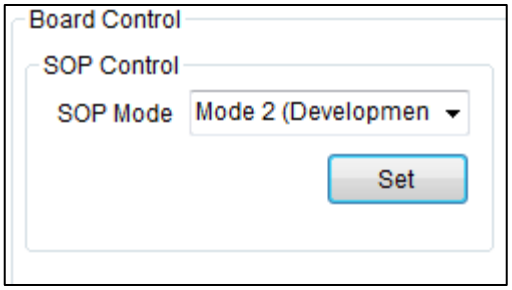


Figure 6.2. SOP Control

6.1.2 RS232 Operations

The serial port mode provides a direct PC connection of the XWR1xx device through the RS232 interface through the XDS110-USB interface. This mode allows firmware download and enables the device for characterization purposes. The serial port mode is also the default debug option on the Windows platform used to operate the device in TX and RX mode and using the read/write registers.

Once the SOP Mode is set, select the COM Port enumerated as 'XDS110 Class Application/User UART' for RS232 operations and click **Connect** button. The XDS110 port enumerates two ports as explained in USB interfaces and drivers section.

NOTE: Use the default Baud Rate of 921600 bps.

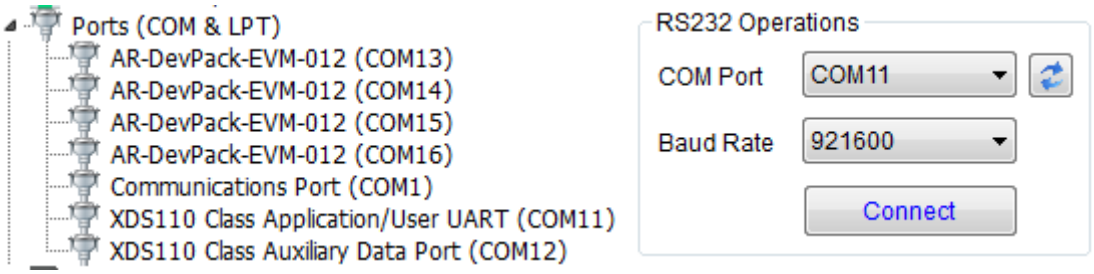


Figure 6.3. Serial Port Control and selecting the COM port for RS232 operations

Since the default baud rate configured in the device is 115200 bps, mmWaveStudio attempts to connect in 115200 baud rate, and then configures the RS232 module to re-establish connection in 921600 baud rate.

6.1.3 Firmware Download

The XWR1xx firmware is downloaded by the mmWaveStudio to the XWR1xx device while in Development mode (**SOP2**). The development mode provides a debug connection to the device using the dedicated RS232 interface such as writing and reading registers.

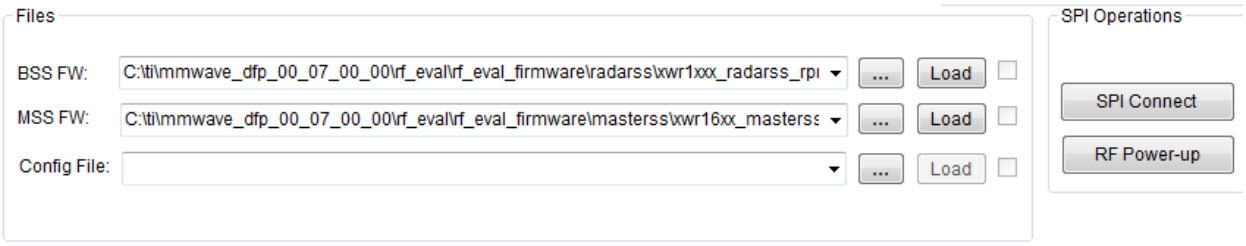


Figure 6.4. Firmware Download

The firmware download steps are as follows:

1. Select SOP Mode 2 (SOP selection is not needed when using DCA1000 EVM. User has to set the jumpers on xWR1xxx BOOST to set the device in SOP2 mode. Jumpers on SOP0 and SOP1 should be closed and SOP2 should be left open to set the device in SOP2 mode)
2. Connect over RS232
3. In the files area of the Connect tab, set the BSS and MSS firmware. Click the ... button and browse to the location of the file (e.g.

   mmwave_studio_<ver>\rf_eval_firmware\radarss\xwr16xx_radarss_rprc.bin for XWR16xx and

   mmwave_studio_<ver>\rf_eval_firmware\radarss\xwr12xx_xwr14xx_radarss_.bin for XWR12xx/XWR14xx devices
4. For MSS, load the binary file (e.g.

   mmwave_studio_<ver>\rf_eval_firmware\masterss\xwr16xx_masterss_rprc.bin for XWR16xx and

   mmwave_studio_<ver>\rf_eval_firmware\masterss\xwr12xx_xwr14xx_masterss.bin for XWR12xx/XWR14xx devices).
5. Click Load next to the BSS firmware. The mmWaveStudio application begins downloading the firmware files entered in the previous steps.

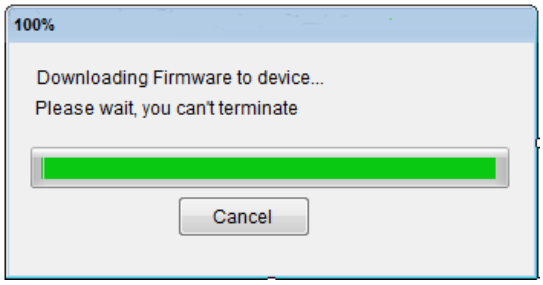


Figure 6.5. Firmware Download Progress

6. Once the BSS firmware download is complete, the Firmware version gets updated in the Status section

7. Click Load next to the MSS firmware update. The MSS Firmware version gets updated upon successful download.

6.1.4 SPI Connection & RF Power Up

The mmWave radar device communicates with the external host processor using the SPI interface. The mmWave device is configured and controlled from the external host processor by sending commands to mmWave device over SPI.

1. Once the MSS firmware boot up is complete, Click SPI Connect.
2. The SPI Connect button becomes SPI Disconnect indicating a success. If SPI connect does not succeed you may need to erase the serial flash and try again using the Uniflash tool.
3. Once SPI Connection is successful, Click RF PowerUp button. This command initiates BIST SS power up. Now, the user can issue commands to the device over SPI Communication interface. Go to section 7 to issue commands to the Radar device for RF evaluation.

7. Static Config Tab Operations

Using this tab, the user can set the following static device configurations:

1. RX and TX channel(s) needed for device operation
2. The data format of the ADC output (including the digital filtering)
3. RF LDO bypass option (Not used on EVMs)

> **NOTE:** RF LDO bypass option should not be enabled on EVMs. If RF LDOs are enabled, EVM may get damaged.

4. Low power options in the Sigma Delta ADC root sampling clock rate (reducing rate to half to save power in small IF bandwidth applications).
5. Frequency limits of operation
6. Trigger basic calibrations and RF initializations

After successful SPI Connection and RF Power Up, the Set buttons can be clicked in the order shown below:

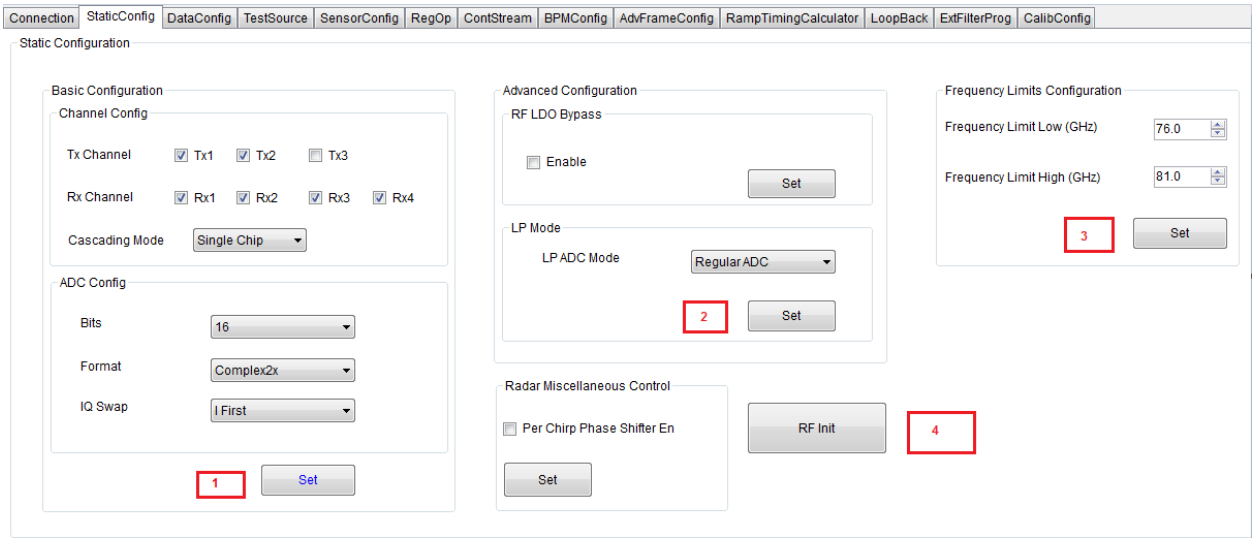


Figure 7.1. Static Config Tab

8. Data Config Tab Operations

Using this tab, the user can set the following Data Path configuration:

1. Data path format to transfer the captured ADC samples received over the receive chain to be transferred out to an external host.
2. Lane Configuration of the LVDS path to transfer Radar information to an external host.
3. Clock configurations of the LVDS lanes

Once the Static Configuration is complete, configure the Data Path by clicking Set button as follows

NOTE:

1. Matlab post processing supports data capture only in following modes
   a. 4 RX channel, 4 LVDS lanes
   b. 2 RX channels, 2 LVDS lanes
   c. 1 RX channel, 1 LVDS lane
2. Also, note that the RX channel number and LVDS lane number should match. For e.g. if user wants data from RX channels 1 and 3, then enabled LVDS lanes should be 1 and 3

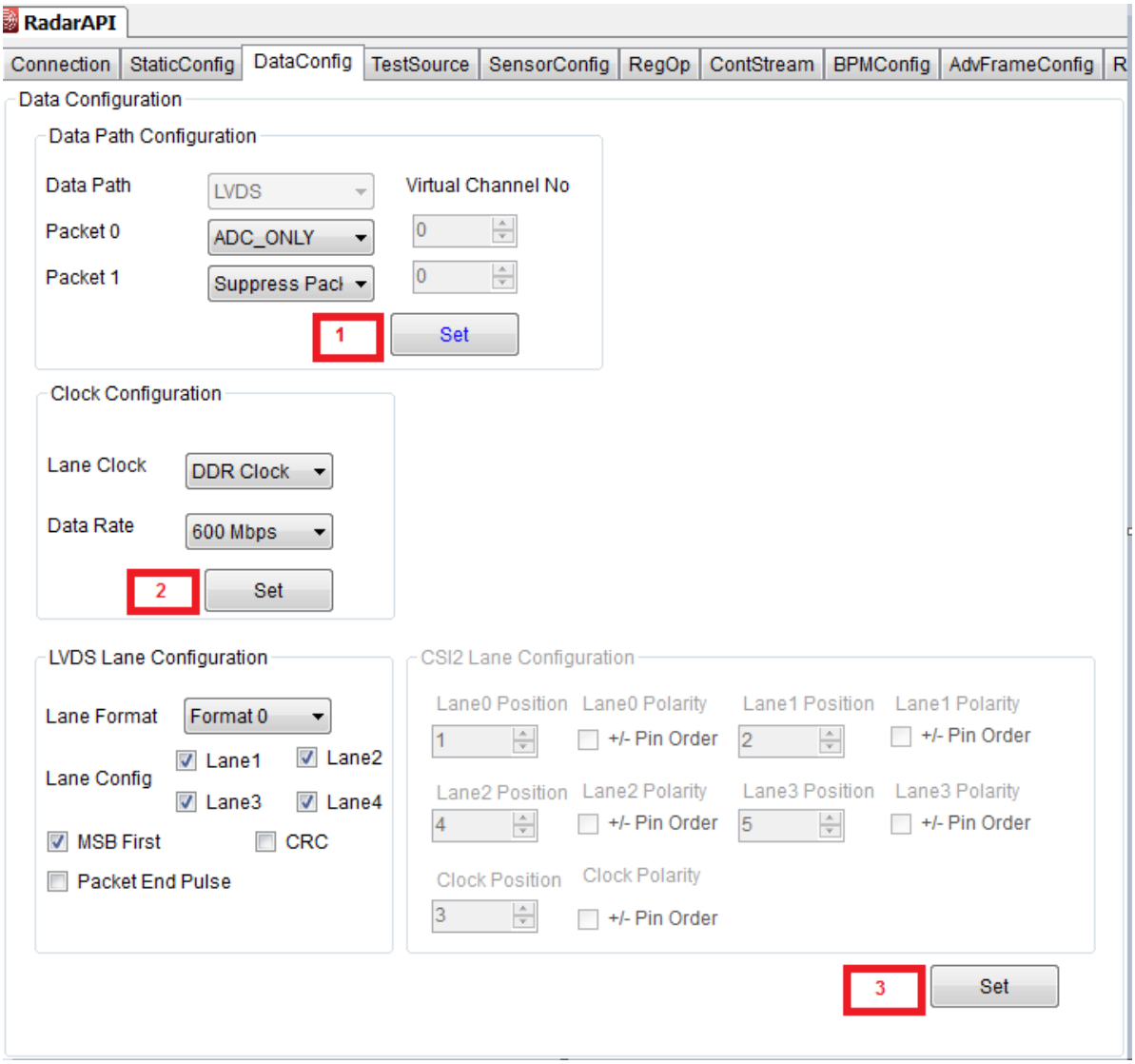


Figure 8.1. Data Config Tab

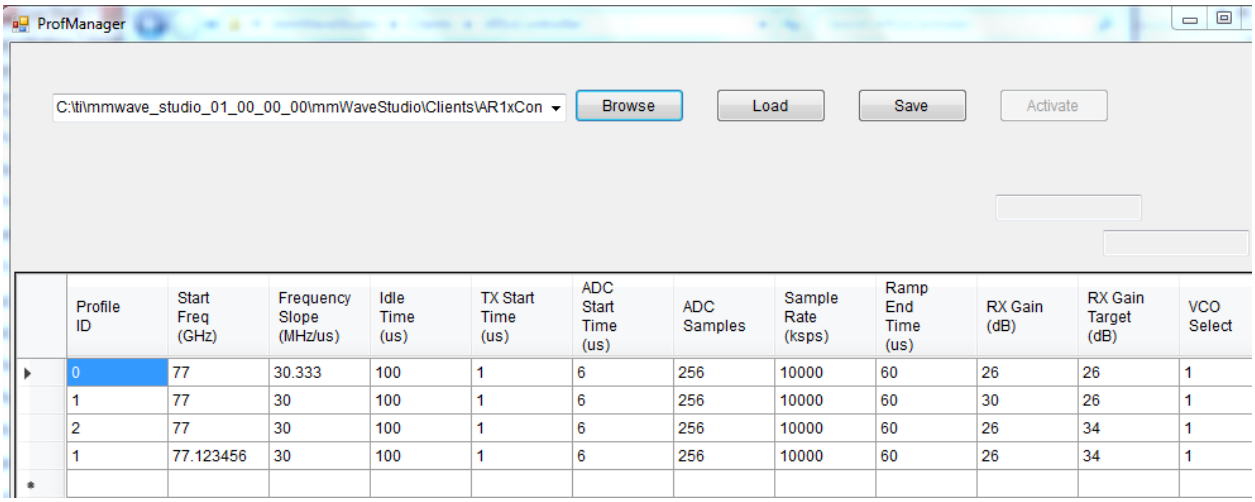
9. Sensor Config Tab Operations

Using this tab, the user can configure the chirp, profile parameter, associating defined profile to chirp index, frame configuration and other RF parameters.

9.1 Profile Config

Sets FMCW radar chirp profiles or properties (FMCW slope, chirp duration, TX power etc.). Since the device supports multiple profiles, each profile is defined in this sub block. Internal RF and analog calibrations may be triggered upon receiving this sub block and ASYNC_EVENT response sent once completed.

9.1.1 Profile Manager



Profile ID	Start Freq (GHz)	Frequency Slope (MHz/us)	Idle Time (us)	TX Start Time (us)	ADC Start Time (us)	ADC Samples	Sample Rate (ksps)	Ramp End Time (us)	RX Gain (dB)	RX Gain Target (dB)	VCO Select
0	77	30.333	100	1	6	256	10000	60	26	26	1
1	77	30	100	1	6	256	10000	60	30	26	1
2	77	30	100	1	6	256	10000	60	26	34	1
1	77.123456	30	100	1	6	256	10000	60	26	34	1

Figure 9.1. Profile Manager Tab

Users can use the Profile Manager to configure multiple profiles by clicking on Manage Profile button in the GUI. Each row corresponds to one profile. The sequence is as follows

Edit →Save → Activate

Or if the csv (ProfConfigData.csv) is modified externally, the device can be configured by using the following sequence

Load → Save → Activate

9.2 Chirp Config

This sub block contains chirp to chirp variations on top of the chirp profiles defined in the Profile config API. E.g. which profile is to be used for each chirp in a frame, and small dithers in FMCW start frequency and idle time for each chirp are possible and are defined here.

9.2.1 Chirp manager

Chirp Manager can be used to configure more chirps by storing all the configurations in csv file and loading it automatically. The Chirp Manager window is as shown as below.

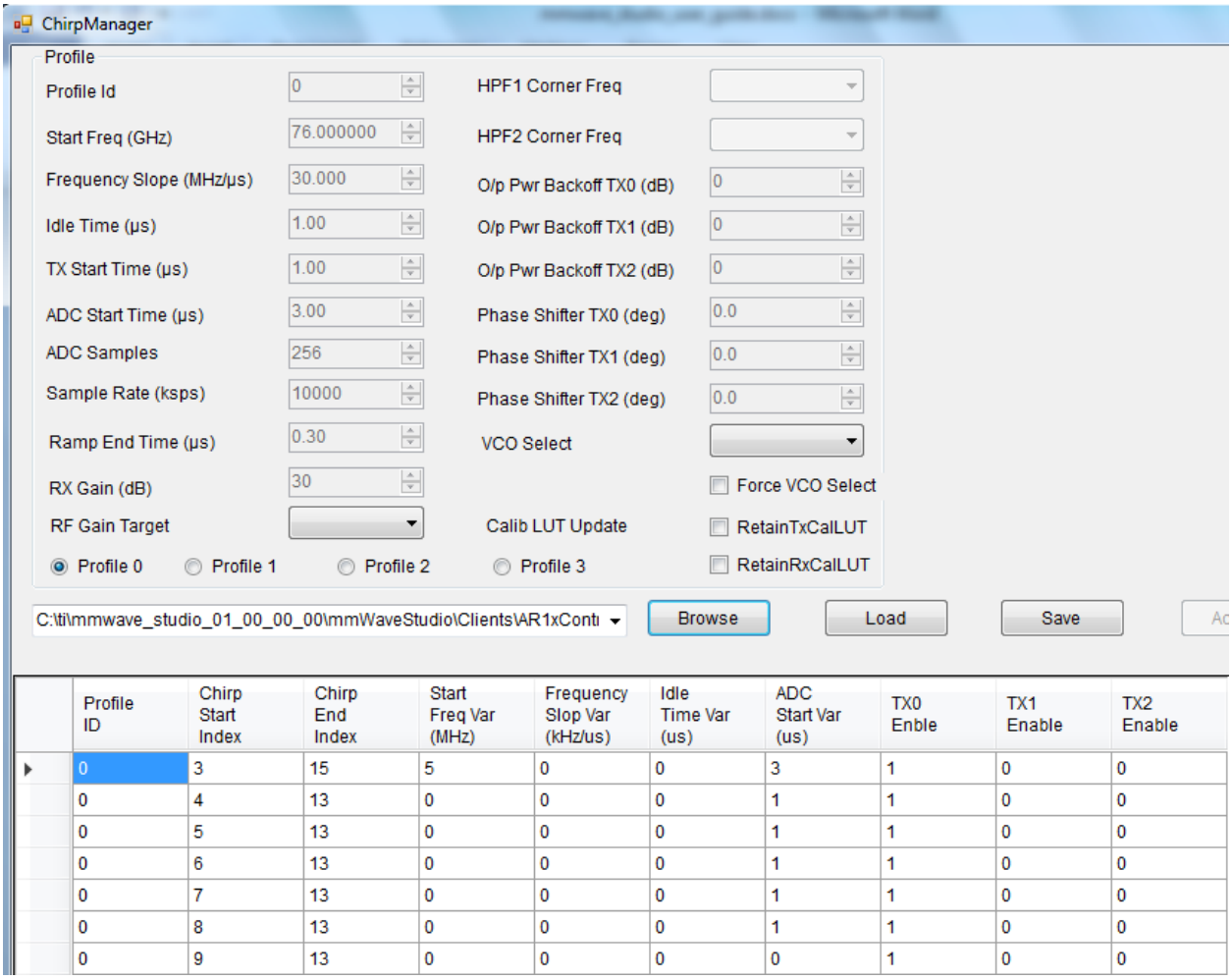
The sequence to configure chirps is as follows

Edit → Save → Activate

Or edit the csv file ChirpConfigData.csv externally and then

Load → Save → Activate

Note: The profile configuration as shown in Chirp Manager tab will be updated only if the Profiles are configured through the Profile Manager.



Profile ID	Chirp Start Index	Chirp End Index	Start Freq Var (MHz)	Frequency Slop Var (kHz/us)	Idle Time Var (us)	ADC Start Var (us)	TX0 Enble	TX1 Enable	TX2 Enable
0	3	15	5	0	0	3	1	0	0
0	4	13	0	0	0	1	1	0	0
0	5	13	0	0	0	1	1	0	0
0	6	13	0	0	0	1	1	0	0
0	7	13	0	0	0	1	1	0	0
0	8	13	0	0	0	1	1	0	0
0	9	13	0	0	0	0	1	0	0

Figure 9.2. Chirp Manager Tab

9.3 Frame Config

This sub block defines a frame, i.e. a sequence of chirps to be transmitted subsequently, the no. of frames to be transmitted, frame periodicity and how to trigger them.

NOTE:

1. There is a new option called “Dummy Chirps (End)” added in Frame Configuration. This denotes the number of dummy chirps which will run in the rampgen at the end of the frame. The ADC data, CP and CQ data for these chirps will not be sent out from the device.
2. In the case of non-zero dummy chirps being configured, the in-built Matlab post processing in Studio supports only when number of chirps loops is equal to 1. There is also a warning issued during frame configuration denoting “*Matlab Post Processing will not work properly if (chirp loops is not equal to 1 and dummy chirps is not equal to 0*”.

NOTE:

3. Ensure to open the HSDCPro software before issuing the Frame Configuration.
4. Matlab RunTime Engine (Version 8.5.1) and HSDCPro Software are pre-requisites for using the Post Processing utility available in mmWaveStudio GUI
5. Number of frames is zero for infinite samples

Follow the sequence as numbered in Sensor Configuration window as shown

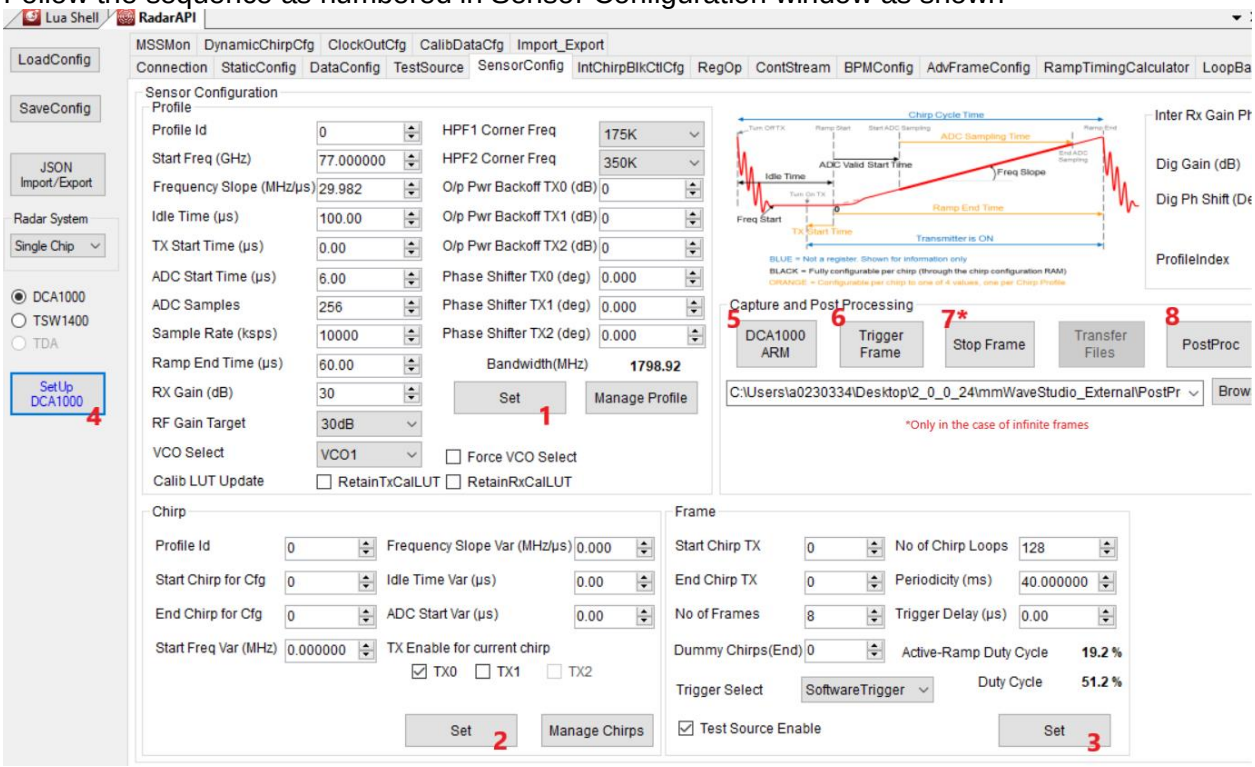


Figure 9.3. Sensor Config Tab

1. Open HSDCPro Software
2. Configure Profile
3. Configure Chirp
4. Configure Frame

NOTE:

1. To capture raw ADC data in TSW1400, do not enable infinite framing mode (no. of frames = 0)
2. TSW1400 captures frames only for 10 seconds. Hence, ensure that total period of data capture (= frame periodicity x No. of frames) < 10 seconds

5. Set up TSW 1400 (ADC capture card). Wait until the setup completes.

   In case of DCA1000, follow these additional steps as shown below

a. Set the PC IP address to 192.168.33.30
Set Sub Net Mask as 255.255.255.0

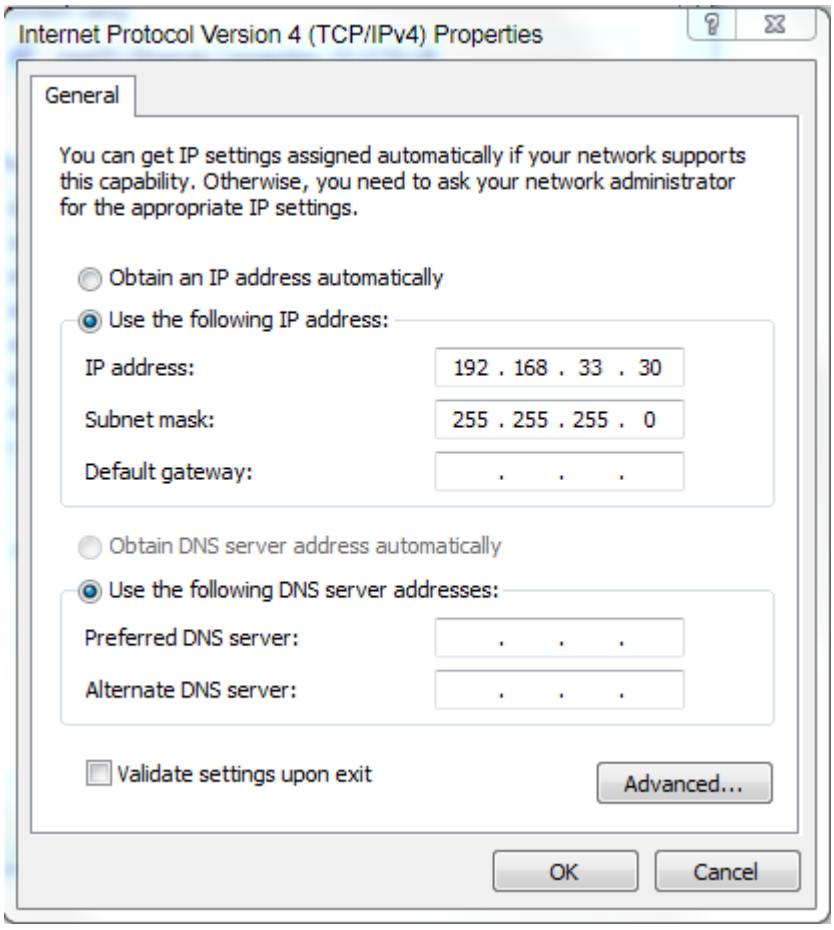


Figure 9.4. IP address configuration in case of DCA1000 EVM

b. Click on Setup DCA1000 from the Connection tab

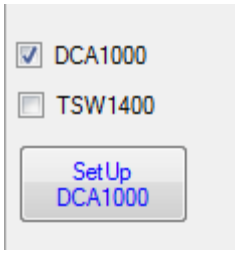


Figure 9.5. Setup DCA1000 configuration button from Connection tab

c. Ensure that the Output log shows success for all the commands sent during Setup DCA1000 like in the screenshot below

d. Ensure that FPGA version is read by the GUI as shown below

```
[16:21:43]  [RadarAPI]: arl.GetCaptureCardDllVersion()
[16:21:44]  [RadarAPI]: Sending dll_version command to DCA1000
[16:21:44]  [RadarAPI]:
[16:21:44]  DLL Version : 1.0
[16:21:44]  [RadarAPI]: arl.SelectCaptureDevice("DCA1000")
[16:21:44]  [RadarAPI]: Status: Passed
[16:21:46]  [RadarAPI]: arl.CaptureCardConfig_EthInit("192.168.33.30", "192.168.33.180",
"12:34:56:78:90:12", 4096, 4098)
[16:21:46]  [RadarAPI]: arl.CaptureCardConfig_Mode(1, 1, 1, 2, 3, 30)
[16:21:46]  [RadarAPI]: arl.CaptureCardConfig_PacketDelay(25)
[16:21:46]  [RadarAPI]: Sending fpga command to DCA1000
[16:21:46]  [RadarAPI]:
[16:21:46]  FPGA Configuration command : Success
[16:21:47]  [RadarAPI]: Sending record command to DCA1000
[16:21:47]  [RadarAPI]:
[16:21:47]  Configure Record command : Success
[16:21:47]  [RadarAPI]: arl.GetCaptureCardFPGAVersion()
[16:21:47]  [RadarAPI]: Sending fpga_version command to DCA1000
[16:21:48]  [RadarAPI]:
[16:21:48]
[16:21:48]  FPGA Version : 2.8 [Record]
[16:21:48]
```

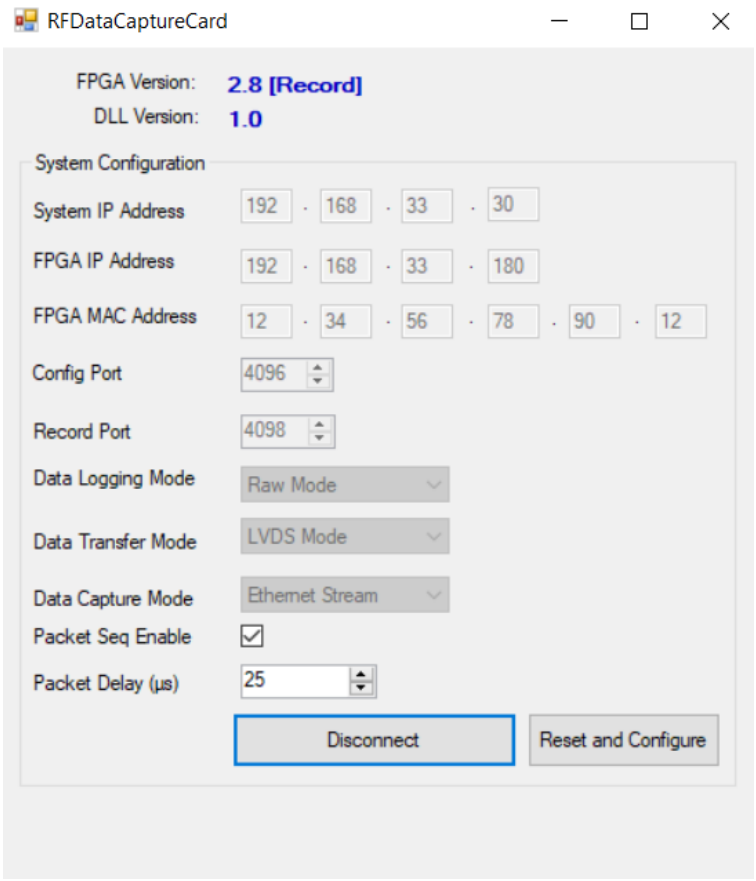


Figure 9.6. DCA1000 configuration window

6. Note that the text input field allows one to select the file to which captured ADC data is to be stored. Give an appropriate filename. ARM TSW1400 or DCA1000 ARM. Wait for two seconds.

7. Trigger Frame
8. Stop Frame if infinite frames have been configured.
9. Post Processing. Opens a ‘radar post processing’ tool that reads the ‘dump file’ and processes it, and finally displays a series of useful plots. The basic post processing window appears as follows

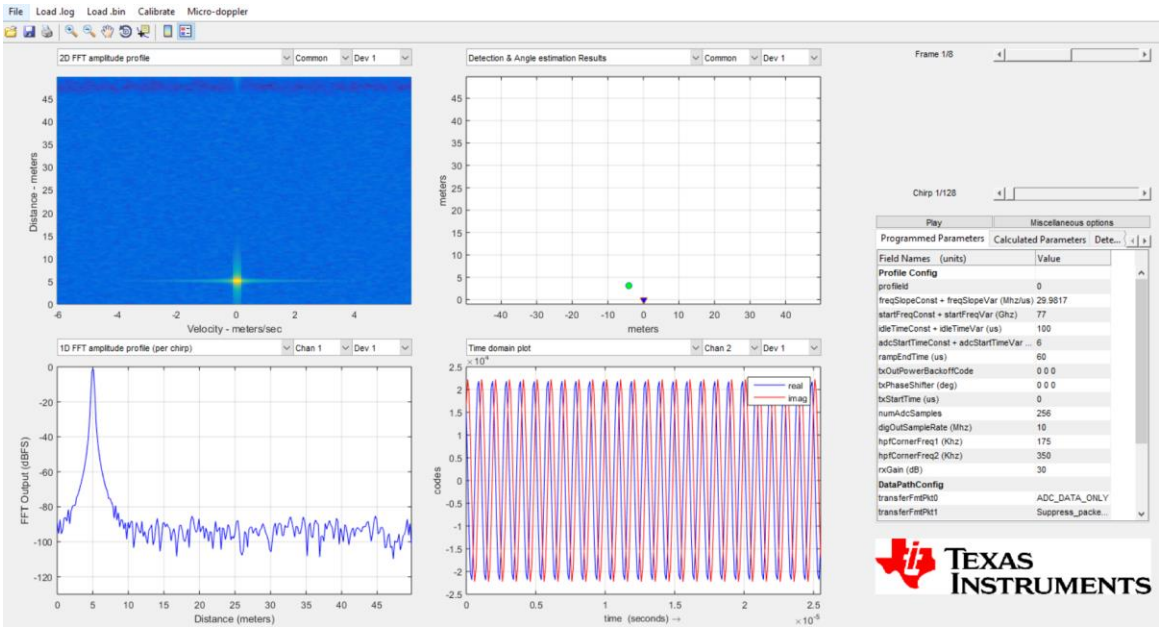


Figure 9.7. Post Processing in Matlab

NOTE:

1. It is recommended to perform “Setup DCA1000” just before DCA1000 ARM for the first time/session capture.
2. Make sure to provide at least 2 seconds gap between DCA1000 ARM and Trigger Frame. Else, no LVDS data would be captured and there would be a timeout for 30 secs.
3. On successful capture, “Record is completed” message would be displayed in the Output console window.
4. Steps 6, 7, 8 and 9 can be run multiple times in a loop.
5. If the number of samples to be collected changes (for e.g. if the number of frames is modified), then step 5 (setup TSW1400) needs to be retriggered.
6. The tool TSW1400 can only capture samples in multiples of 4096 (real samples per channel). If you have an odd number of samples to collect, please increase the number of frames, so that at least 4096 samples.

10. RegOp Tab

10.1 Register Operations Tab

Using RS232 communication interface, the device registers can be accessed using Register Operations tab. Also, the debug signals like APLL, synth can be configured using the Debug Signals section.

> **NOTE:** The Register Operation is enabled only upon successful connection over RS232 to XWR1xx device

Register Read/Write

Read/Write can be performed through bit fields and also 32-bit values.

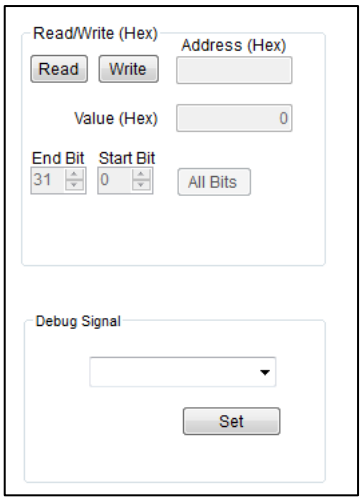


Figure 10.1. Register Read/Write

The figure shows the read/write registers, which is used to read and modify specific registers using their address and start/end bits for masking. Enter the address (in hex) and click the **Read** button to read it or set a value (in hex) and click **write** button to modify it. Click the All Bits button to mask the entire register.

10.2 Debug Signals

Select the debug signal and Click **Set** button. The debug signals available are:

1. NO MUX
2. APLL OUT
3. SYNTH OUT(2.5G)
4. SYNTH OUT(5G)

10.3 GPIO_0

The GPIO_0 can be used to bring out ADC valid signal which can be used to trigger the external instrument for synthesizer related measurements.

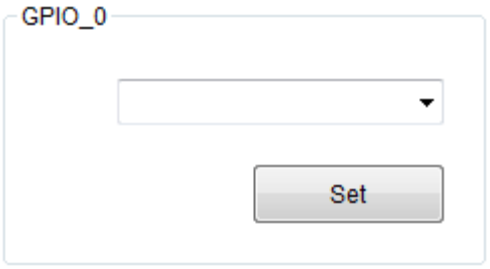


Figure 10.2. GPIO_0 option selection box

11. Continuous Streaming Tab Operations

The Continuous Streaming tab can be used for RF measurements. This tab contains the configurations of the data path to transfer the captured ADC samples continuously without missing any sample to an external host.

1. Set the StreamConfig parameters
2. Enable Continuous Streaming
3. Select Capture (to capture ADC samples) and then (optionally) select Display to process and display the captured ADC samples in the Radar Post Processing GUI.
4. Note: The number of samples to capture can be configured in the ‘Basic Configuration for Analysis tool’ Tab.
5. The remaining items in the ‘Basic Configuration for Analysis tool’ are not used by the ‘Post Processing tool’ but by the ‘Analysis tool’ which is a series of APIs used for ‘Signal Analysis’.

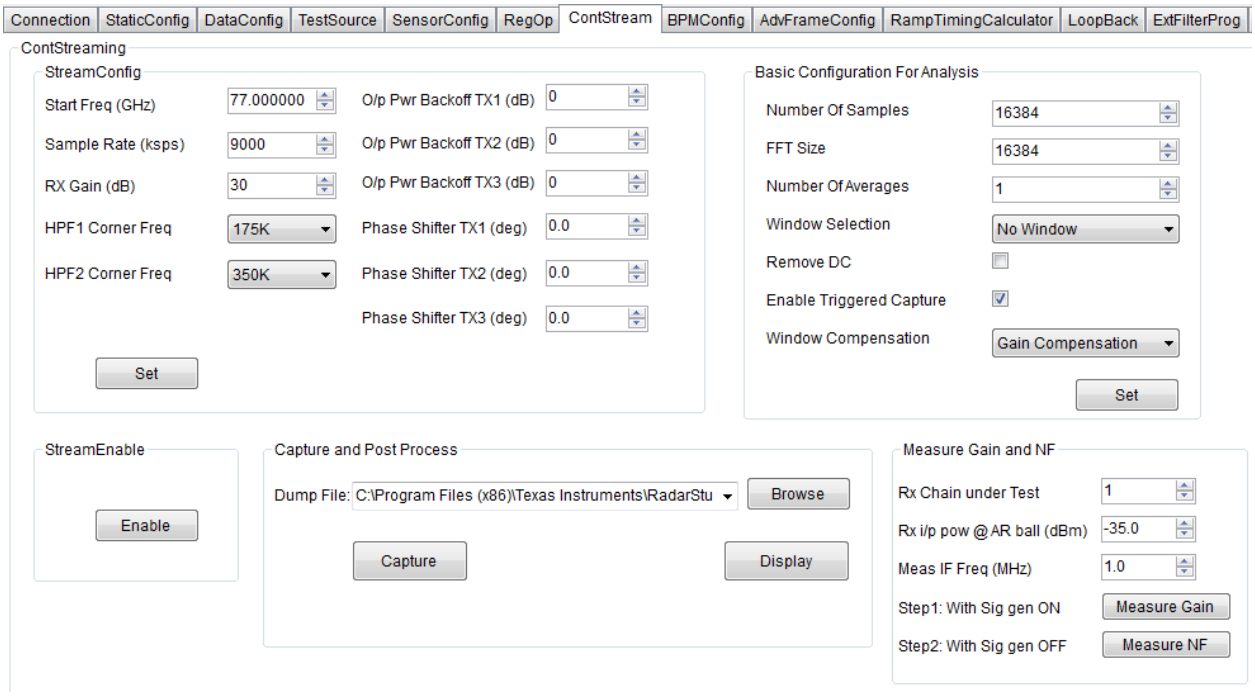


Figure 11.1. Continuous Streaming Tab

12. BPM Config Tab Operations

Using the BPM tab, static configurations related to BPM (Binary Phase Modulation) feature in each of the TXs is performed. Select the Start and End Index. Configure TX and Click **Set** button.

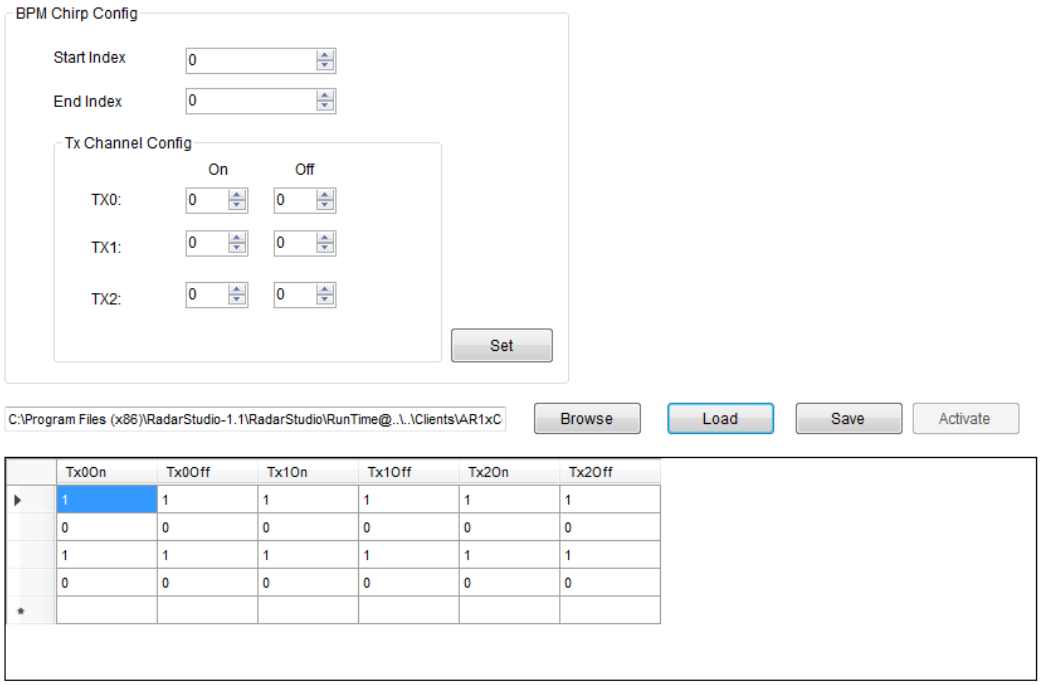


Figure 12.1. BPM Config Tab

13. Event Monitor Tab

This tab allows the users to read temperature sensor information and to read the external inputs fed to GPADC. Also, this tab allows reading of DFE statistics from the last frame.

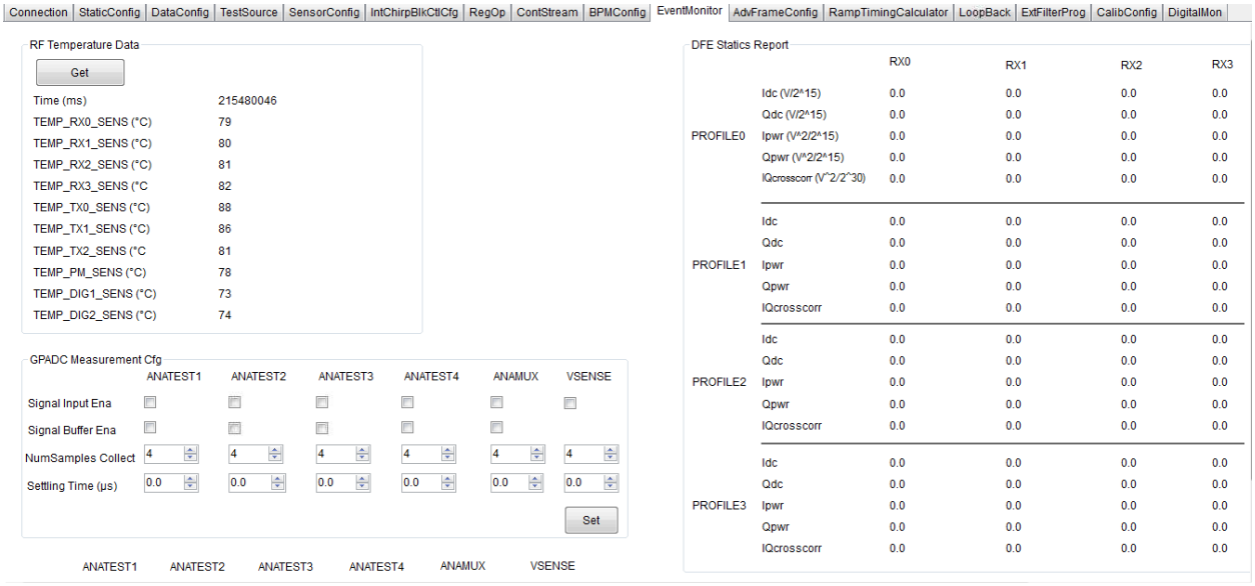


Figure 13.1. Event Monitor Tab

14. Advanced Frame Config Tab operations

This tab allows users with advanced frame configuration. Configuration options include sub-frames, bursts within each sub-frame, program burst loops, selection of chirp start index for the first chirp within the burst, configuration of sub-frame periodicity and burst periodicity. Also there are options to force a particular chirp profile within a sub-frame, selection of trigger source software/hardware and to program a trigger delay

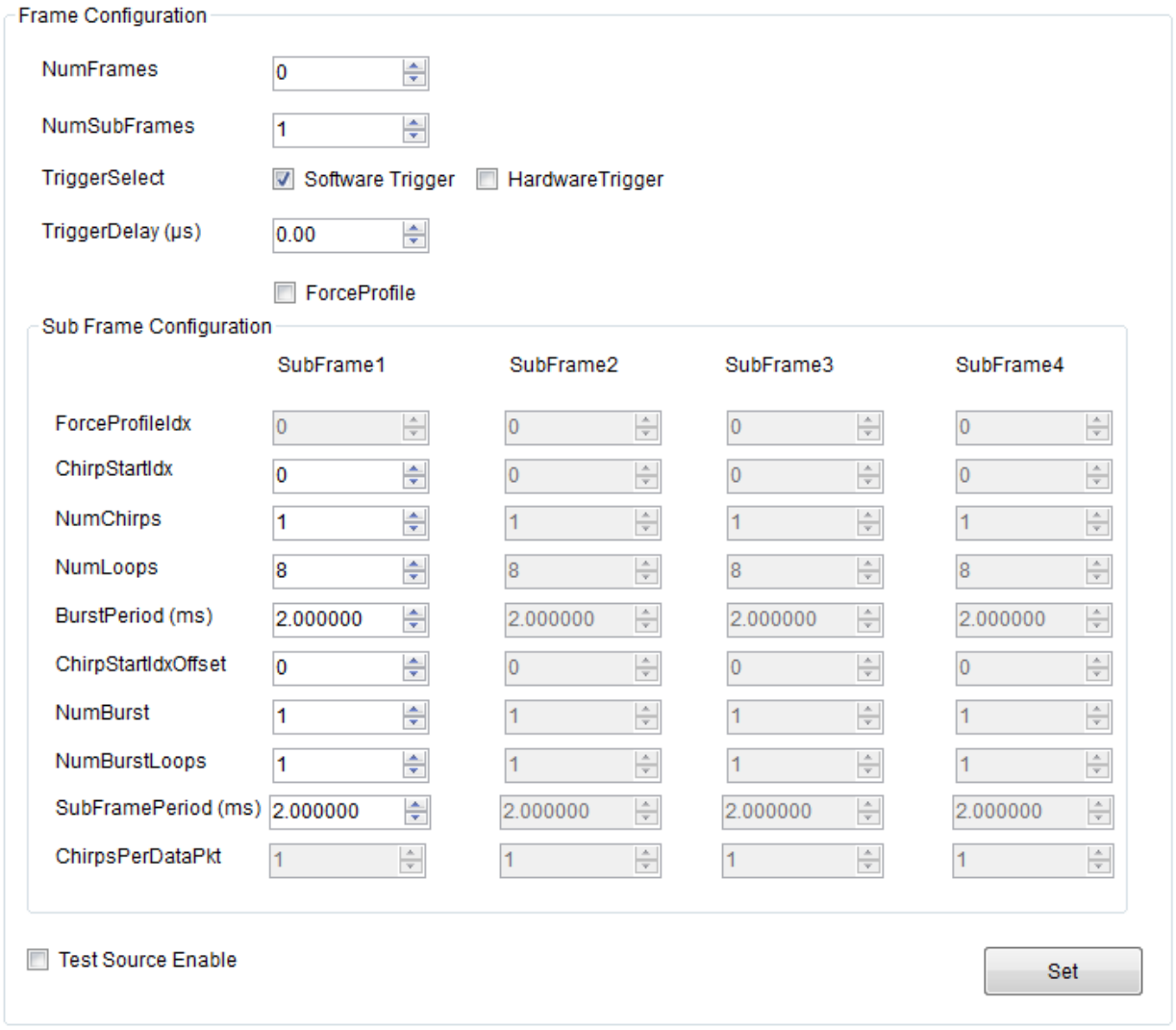


Figure 14.1. Advanced frame config tab

15. Rampgen Timing Calculator Tab

This tab gives the recommended timing parameters of the different chirp parameters like idle time, ADC start time, ramp end time and inter-chirp time etc. based on following inputs: ADC operating mode, DFE mode, slope, HPF corner frequencies and sampling rate. These calculations do not take into account the high speed transfer rate (CSI2/LVDS) requirements.

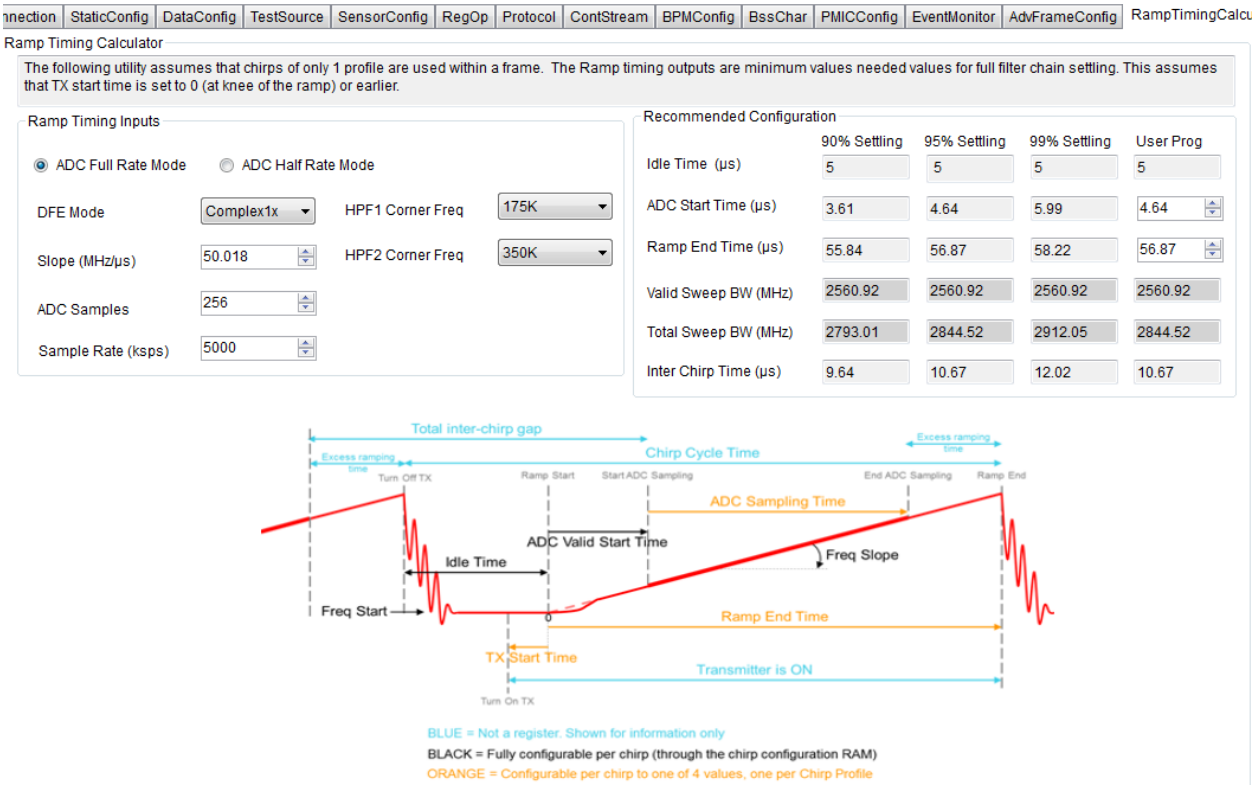


Figure 15.1. Rampgen Timing Calculator

16. Loopback Tab Operations

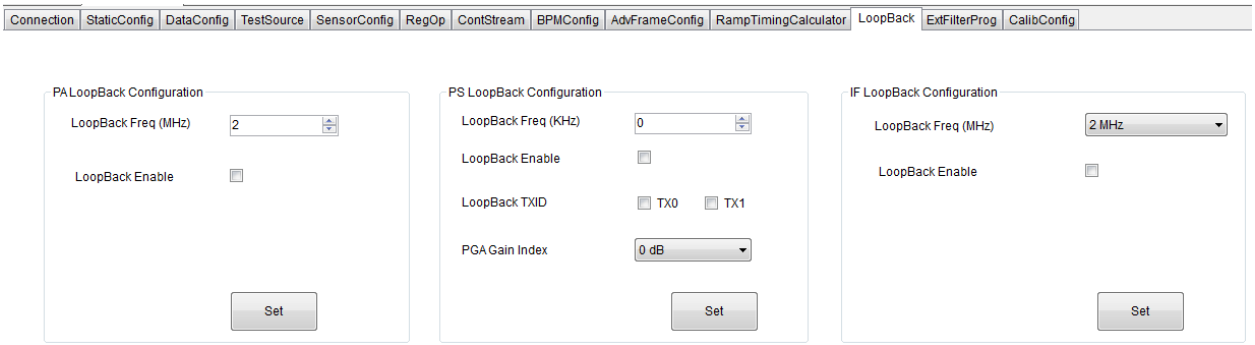


Figure 16.1. Loopback Config Tab

Loopback Tab supports following loopback options

1. PA Loopback
2. Phase Shifter loopback
3. IF loopback

The loopback APIs are intended to be issued after profile config and chirp config are issued to the device and before frame config API. The entire frame triggered from then

on will have loopback enabled. To disable loopback, the loopbacks can be disabled in this tab and frames triggered from then on will have loopbacks disabled.

> **NOTE:** All profiles in the frame will have loopback paths enabled as long as the loopback is enabled.

17. CalibConfig Tab Operations

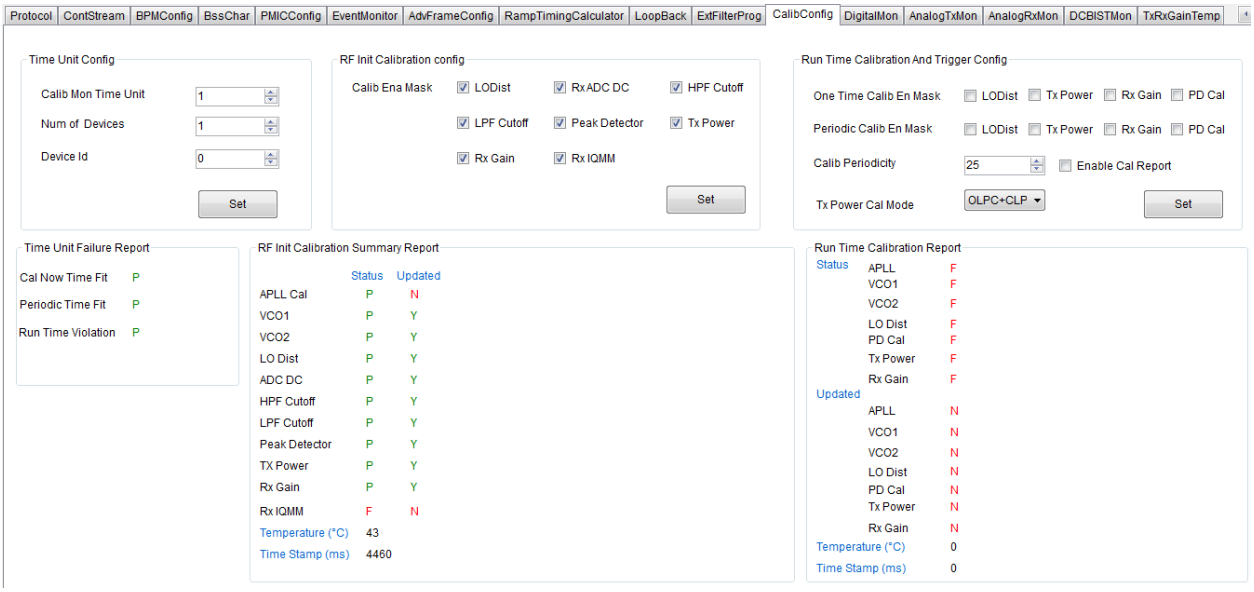


Figure 17.1. CalibConfig Tab

The CalibConfig Tab allows users to control the RF Init calibrations and Runtime calibrations which are scheduled by the firmware.

17.1 Time Unit Config

This allows configuration of the Calibration/Monitoring Time Unit in units of frames. No. of devices should be set to 1 if only 1 device is used

17.2 RF Init Calibration Config

This allows control of calibrations which will be triggered at RF Init. Default operation is will all calibrations enabled

17.3 RunTime Calibration and Trigger Config

This allows either ‘One Time calibration’ which will be triggered instantaneously or ‘Periodic calibrations based on a Calibration Periodicity (in units of CalibMonTimeUnits) which will be triggered periodically when the frames are transmitted. Periodic calibrations are not triggered are frames are not ongoing.

17.4 Time Unit Failure report

This indicates a failure if the inter-frame/inter-burst idle time is not sufficient to trigger periodic calibrations/monitoring as specified by the user. The failure message is sent after frame config message is sent to the device.

17.5 RF Init Calibration Summary Report

This indicates the status of RF Init calibrations and if the results were applied in hardware. Along with this report, the boot time temperature and the time stamp is also indicated.

17.6 Runtime Calibration Report

This indicates the status of run time calibrations and if their results were applied in the hardware. Along with this information, the current device temperature and the timestamp at which the calibration results were applied is also indicated. If the user does not enable 'Cal Report' in the 'Run Time Calibration and Trigger Config', then the periodic Run Time calibration reports will not be sent by the device.

18. Monitoring

18.1 Digital monitoring

The entire digital monitoring boot up test status, latent fault tests and periodic tests can be triggered through this tab. This is not functional in the firmware now, but will be made available in future releases.

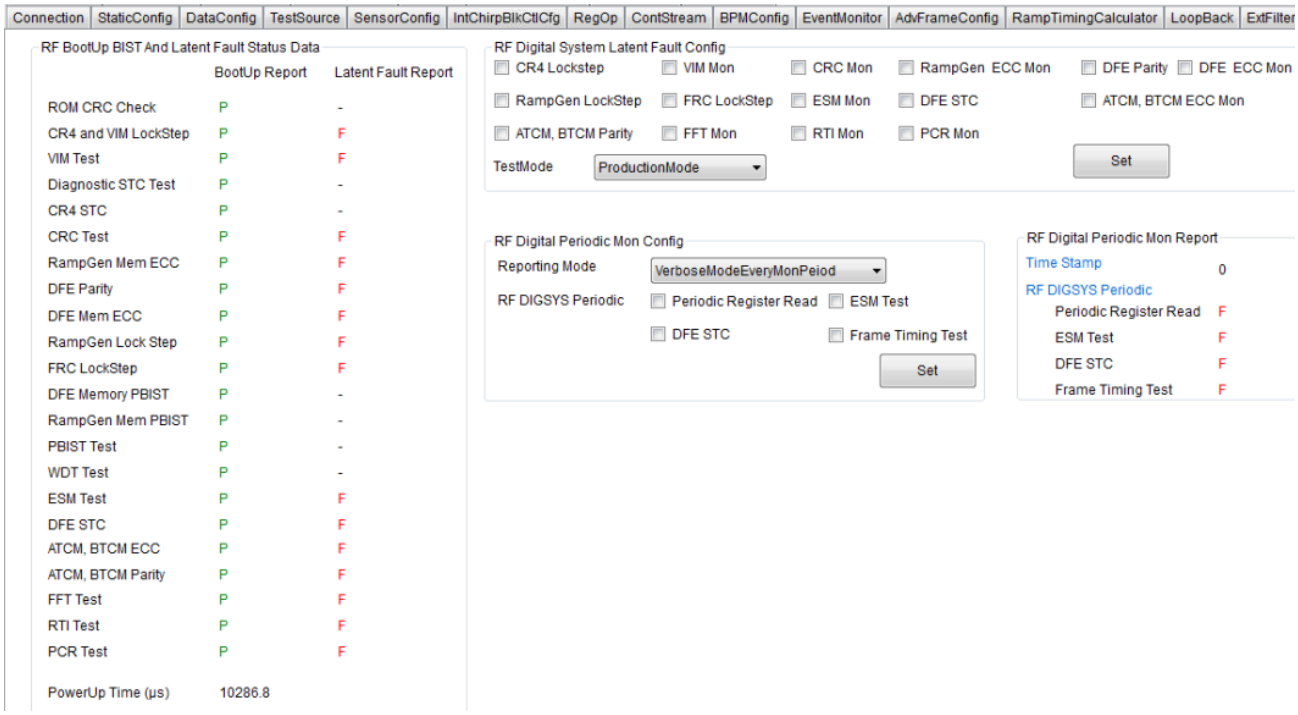


Figure 18.1. Digital monitoring configuration

18.2 Analog monitoring enables

This is included in AnalogTxMon tab. There is a checkbox for each analog monitoring feature

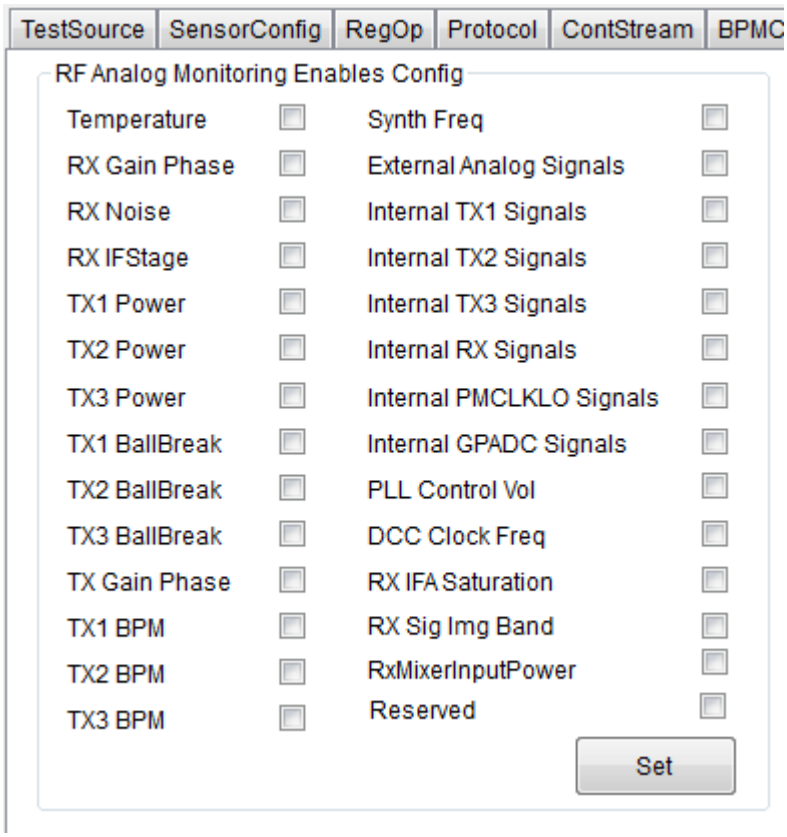


Figure 18.2. RF Analog monitoring consolidated enables

18.3 TX monitoring

The entire analog TX monitoring configurations are available in AnalogTxMon tab. This includes

1. TX power monitor
2. TX ball break monitor
3. TX BPM monitor
4. TX gain and phase monitor

The details of these monitor configurations are given in the Radar Interface Control Document.

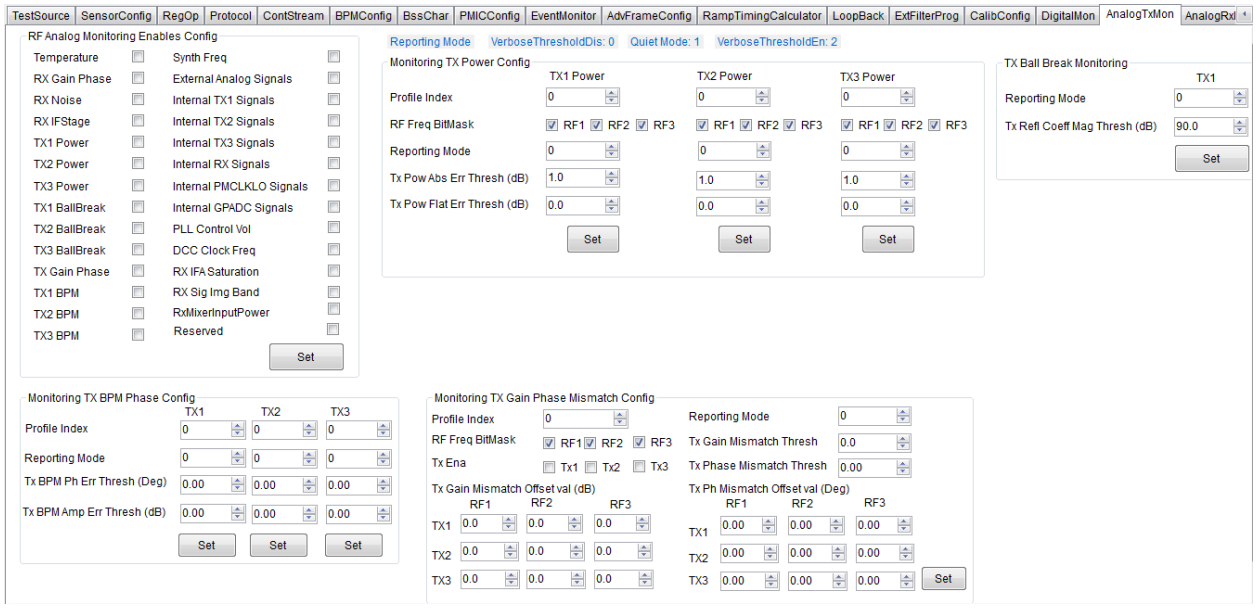


Figure 18.3. Analog TX monitoring configuration

18.4 RX monitoring

The entire analog RX monitoring configuration is available in the AnalogRxMon tab. This includes the configurations for following monitors

1. RX gain phase monitor
2. RX noise figure monitor
3. RX IF stage monitor
4. RX saturation detector monitor
5. RX signal and image band monitor

This tab also includes the temperature monitor and synthesizer frequency error monitor configurations.

The details of these monitor configurations are given in the Radar Interface Control Document.

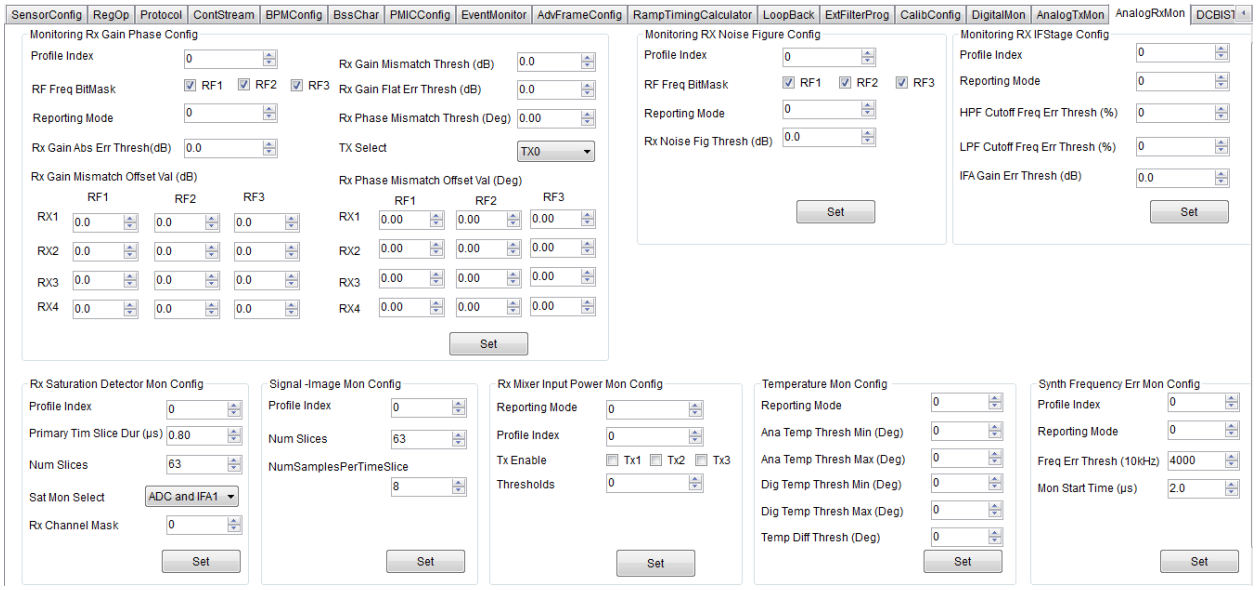


Figure 18.4. Analog RX monitoring configuration

18.5 DCBIST monitoring

The entire DCBIST monitor configurations are available in DCBISTmon tab. This includes the configurations for following monitors

1. External analog signals monitor
2. Internal TX signals monitor
3. Internal RX signals monitor
4. Internal PMCLKLO signals monitor
5. GPADC monitor
6. PLL control voltage monitor
7. DCC based clock monitor

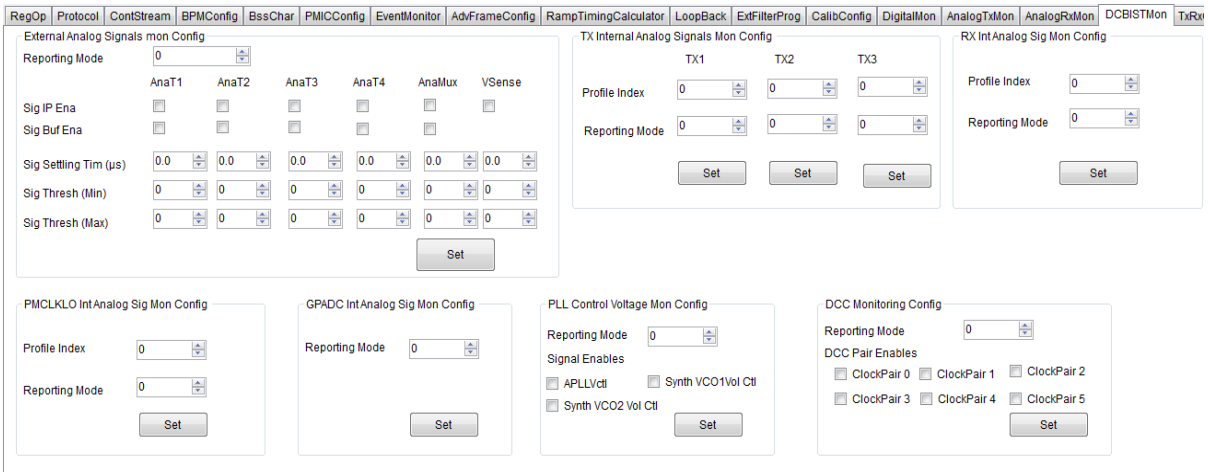


Figure 18.5. DCBIST monitor

The details of these monitor configurations are given in the Radar Interface Control Document.

Note that all configurations need to be done after profile configuration has been defined and before starting the frames. After triggering the frames, the monitoring reports are stored in the MonitoringReport.txt file under the mmwavestudio\PostProc folder.

Note: Since mmWaveStudio uses FTDI for SPI emulation, the SPI speed is limited to 3.75Mbps. Hence the throughput on data transfer is limited by this interface. To avoid any loss of monitoring data, keep the inter-frame idle time as high as 200 ms. This limitation does not apply to the silicon where SPI can be used at a higher clock rate.

18.6 Async Event Report Format

18.6.1 Logging of Async events

The async events sent by the device and received at mmWaveStudio are categorized and logged into 4 different files present under the PostProc folder

1. ***CalibrationReport.txt***

   This report logs all the async events related to the calibrations.

   - RF Init Calibration status
   - Run time Calibration report

2. ***BSSEvents.txt***

   This report logs all the async events related to the BSS.

   - BSS CPU Fault status
   - BSS ESM Fault status
   - RF GPADC measurement data
   - Calibration Monitoring Timing Fail report

3. ***MSSEvents.txt***

   This report logs all the async events related to the MSS

   - MSS Power up async event
   - BSS Power up async event
   - MSS CPU Fault status
   - MSS ESM Fault status
   - MSS Boot Error status
   - MSS Latent Fault monitoring
   - MSS Periodic Test monitoring
   - MSS RF Error status
   - MSS Voltage monitoring Error status

4. ***MonitoringReport.txt***

   This report logs all other async events coming from the device specifically the analog and the digital monitors.

> **NOTE:** In the case of cascade system, the reports for each device are logged into separate files and the name of the file is suffixed with the device index.
>
> For example, on receiving a runtime calibration report from slave 2 (device index 2), it would be logged into a file named CalibrationReport_2.txt

18.6.2 Decoding the Reports

The reports logged into the above files follow the below process:

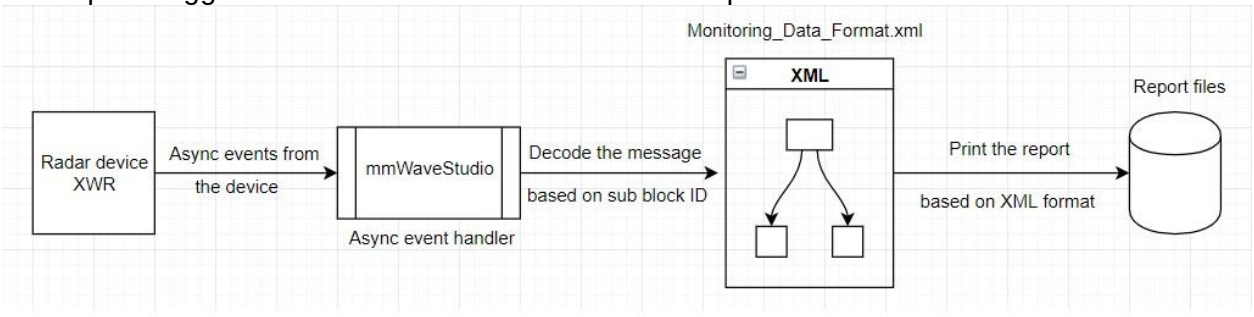


Figure 18.6. Monitoring Report Flow

The async events sent from the device will be received at the async event handler of mmWaveStudio. Based on the sub block ID received, the report can be identified. This async event data is logged into the respective files based on the format present in the Monitoring_Data_Format.xml.

This XML is present at the PostProc folder and by default makes use of the format mentioned in the ICD for each report.

The format for each report in the XML starts with a <*MonReport*> tag.

Each <*MonReport*> tag has the following branches

- <*name*> tag: The name of the report.
- <*size*> tag: The size of the report excluding the sub block ID and sub block length.
- <*field*> tag: Each field in the report.

Each <*field*> tag has the following branches

- <*field_name*> tag: The name of the corresponding field.
- <*offset*> tag: The byte offset from the start of the report of a particular field.
- <*bytes*> tag: The number of bytes corresponding to a particular field.
- <*split*> tag: This tag gives information on whether the following field needs to be printed as a whole (even though it might be a bitmap) or printed separately into sub-fields. If no split is required, it is provided as zero.

If split of the <*field*> is required, then the corresponding sub-fields are represented through the <*bitinfo*> tag. Each <*bitinfo*> tag contains the following fields

- <*bitname*> tag: The name of the corresponding sub-field
- <*bitstart*> tag: the bit offset from the start of the field of a particular sub-field.
- <*bits*> tag: The number of bits corresponding to a particular sub-field.

- <*mult_fact*> tag: This tag provides the multiplication factor on which the corresponding sub-field obtained needs to be tweaked.
- <*signed*> tag: This tag provides the limits similar to a signed value representation.

18.6.3 Example of Decoding the Reports

A sample report of RX Gain Phase Monitoring logged into MonitoringReport.txt file is given below

RXGainPhaseMonitoring: 0, 0, 0, 0, 0, 0, 0, -38, 0, -36, 0, -34, 0, 48.20, 48.40, 47.50, 48.00, 51.80, 51.80, 51.70, 51.00, 52.40, 52.70, 52.20, 50.80, 186.71, 183.48, 186.49, 182.64, 280.90, 284.87, 281.29, 275.81, 14.98, 16.91, 13.14, 8.44, -42, -42, -42, -40, -36, -36, 0, -36, -36, -36, -34, -36, -38, 0, 42902

The fields are decoded as follows

- STATUS_RX_GAIN_ABS 0
- STATUS_RX_GAIN_MISMATCH 0
- STATUS_RX_GAIN_FLATNESS 0
- STATUS_RX_PHASE_MISMATCH 0
- RESERVED 0
- ERROR_CODE 0
- PROFILE_INDX 0
- LOOPBACK_POWER_RF1 -38
- RESERVED 0
- LOOPBACK_POWER_RF2 -36
- RESERVED 0
- LOOPBACK_POWER_RF3 -34
- RESERVED 0
- RX_GAIN_VALUE_RX0_RF1 48.20
- RX_GAIN_VALUE_RX1_RF1 48.40
- RX_GAIN_VALUE_RX2_RF1 47.50
- RX_GAIN_VALUE_RX3_RF1 48.00
- RX_GAIN_VALUE_RX0_RF2 51.80
- RX_GAIN_VALUE_RX1_RF2 51.80
- RX_GAIN_VALUE_RX2_RF2 51.70

- RX_GAIN_VALUE_RX3_RF2 51.00
- RX_GAIN_VALUE_RX0_RF3 52.40
- RX_GAIN_VALUE_RX1_RF3 52.70
- RX_GAIN_VALUE_RX2_RF3 52.20
- RX_GAIN_VALUE_RX3_RF3 50.80
- RX_PHASE_VALUE_RX0_RF1 186.71
- RX_PHASE_VALUE_RX1_RF1 183.48
- RX_PHASE_VALUE_RX2_RF1 186.49
- RX_PHASE_VALUE_RX3_RF1 182.64
- RX_PHASE_VALUE_RX0_RF2 280.90
- RX_PHASE_VALUE_RX1_RF2 284.87
- RX_PHASE_VALUE_RX2_RF2 281.29
- RX_PHASE_VALUE_RX3_RF2 275.81
- RX_PHASE_VALUE_RX0_RF3 14.98
- RX_PHASE_VALUE_RX1_RF3 16.91
- RX_PHASE_VALUE_RX2_RF3 13.14
- RX_PHASE_VALUE_RX3_RF3 8.44
- RX_NOISE_POWER1_RX0_RF1 -42
- RX_NOISE_POWER1_RX1_RF1 -42
- RX_NOISE_POWER1_RX2_RF1 -42
- RX_NOISE_POWER1_RX3_RF1 -40
- RX_NOISE_POWER1_RX0_RF2 -36
- RX_NOISE_POWER1_RX1_RF2 -36
- RESERVED 0
- RX_NOISE_POWER2_RX2_RF2 -36
- RX_NOISE_POWER2_RX3_RF2 -36
- RX_NOISE_POWER2_RX0_RF3 -36
- RX_NOISE_POWER2_RX1_RF3 -34
- RX_NOISE_POWER2_RX2_RF3 -36
- RX_NOISE_POWER2_RX3_RF3 -38
- RESERVED 0

- TIME_STAMP 42902

NOTE: There is another XML placed at the PostProc folder named Monitoring_Data_Format_Legacy.xml which contains the format in which the reports were being printed in the earlier versions.

The user can tweak the XML according to requirement based on the instructions/information provided in the previous section.

If reports are to be generated in a particular format according to an XML, the user needs to rename that particular XML file to Monitoring_Data_Format.xml and place it in the PostProc folder.

For the XML changes to take place, mmWaveStudio needs to be restarted as the XML information/format for each report is loaded at the startup of mmWaveStudio.

19. RX and TX gain LUT

Users have the option to read the RX and TX gain LUTs (for a given profile) used in the firmware to correct for temperature variations. If the user wishes to override the LUT used by the firmware, he/she can modify these LUTs and inject them back into the device. The firmware will then use the user injected LUT instead of its internally generated LUT.

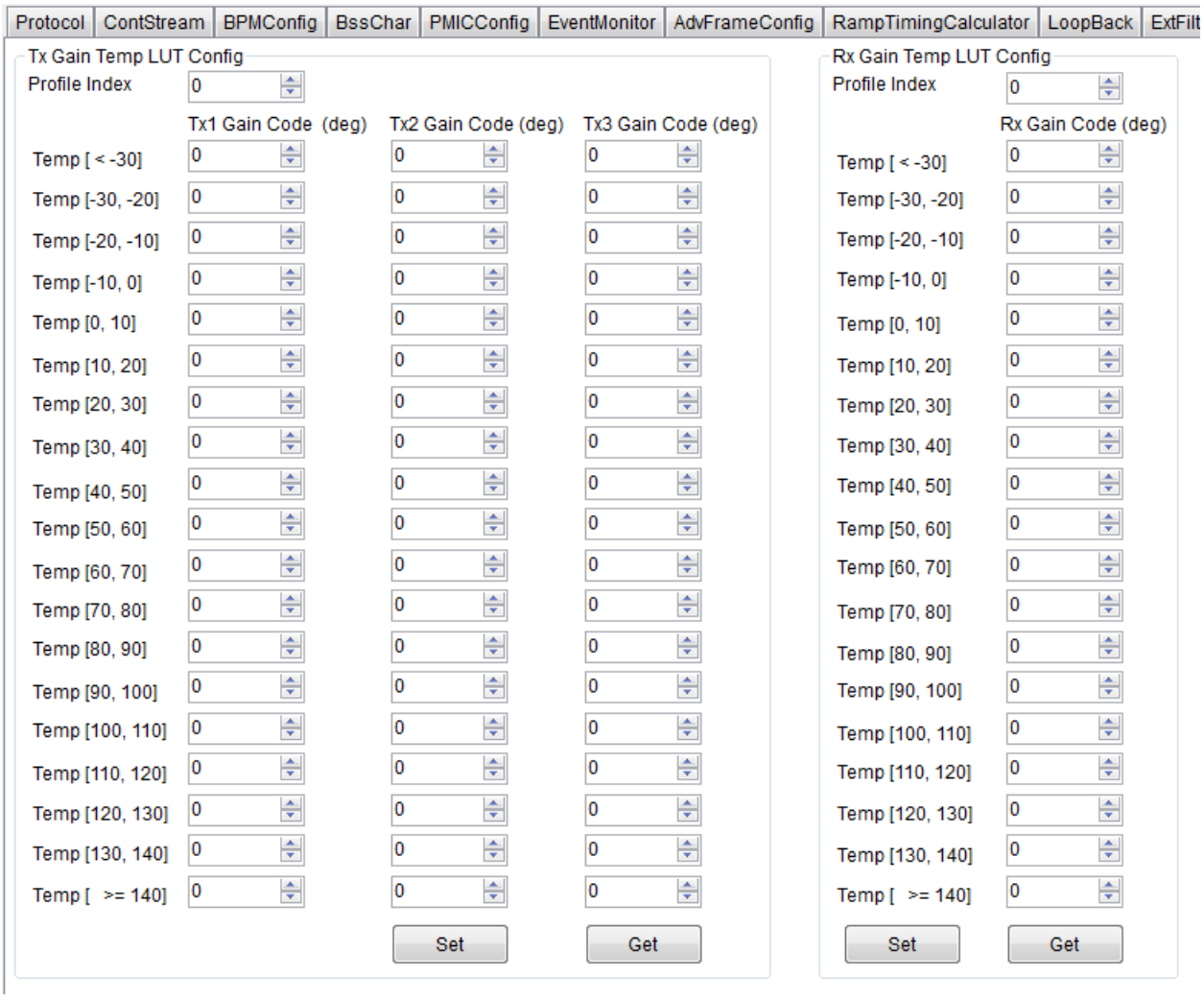


Figure 19.1. RX and TX LUT (read and write)

20. Import / Export Tab Operations

For Data capture, there are certain set of configuration that needs to be performed. These configurations can be classified into following 3 categories:

1. Connection (to the mmWave device)
2. Capture Card Configuration(configuration of the capture card)
3. mmWave Device Configuration

Import/Export tab enables the user to specify all these configurations via JSON format configuration performed from the GUI. The above mentioned configurations are split between 2 JSON files:

1. **Capture Setup File:** contains Connection + Capture Card Configuration
2. **mmWave Configuration File:** contains mmWave Device Configuration.

20.1 GUI (Graphical User Interface)

For ease of location, a JSON Import/Export button has been added on the left side of the GUI, which leads to Import/ Export Tab. The various features on this tab are explained in the subsequent sub sections

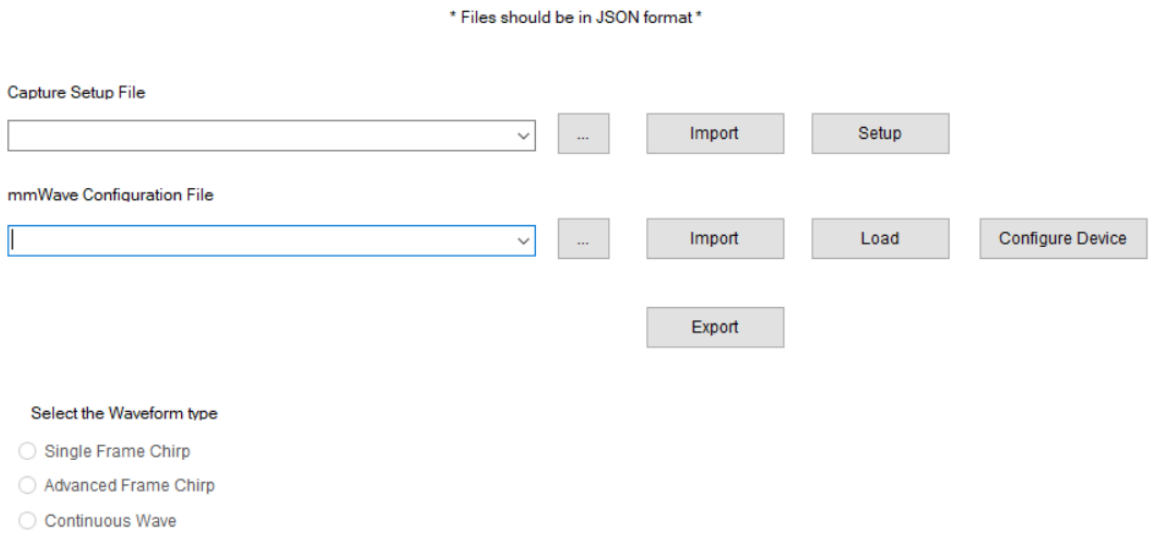


Figure 20.1 Import/Export GUI

20.1.1 Capture Setup file

The capture setup file will contain configuration related to connection tab and Data Capture configuration. Further details are as below:

1. Connection Tab:
    a. COM port settings

b. Baud rate

c. BSS Firmware path

d. MSS Firmware path

e. Operating Frequency [New]

f. Device Variant [New]

2. Capture Card Configuration (for DCA1000)

a. Data logging Mode

b. Data transfer Mode

c. Data Capture Mode

d. Packet sequence enable

e. Packet delay

3. Post proc Info

a. Path and name of raw data capture file

b. Path for name of processed raw data capture file

20.1.2 mmWave Configuration File

This file contains the configuration for the mmWave device. Most of the configurations specified in ICD document are supported.

Note: This file can be generated using the mmWave Sensing Estimator tool.

20.1.3 Types of Waveform

There are 3 waveform types available in the GUI.

1. Single Frame Chirp (Legacy Frame)
2. Advanced Frame Chirp
3. Continuous Wave

The input mmWave configuration file might contain configuration for more than one waveform type. This radio button selection decides which configuration (only one) will be executed for the device.

20.2 Capture Setup JSON Operations

The following features are supported by the Capture Setup section of JSON Parser in mmWaveStudio.

20.2.1 Import

It loads the Setup and Capture specific configuration in the appropriate tabs.

Specifically, this operation loads the Connection Tab related information and RF Capture Card related information.

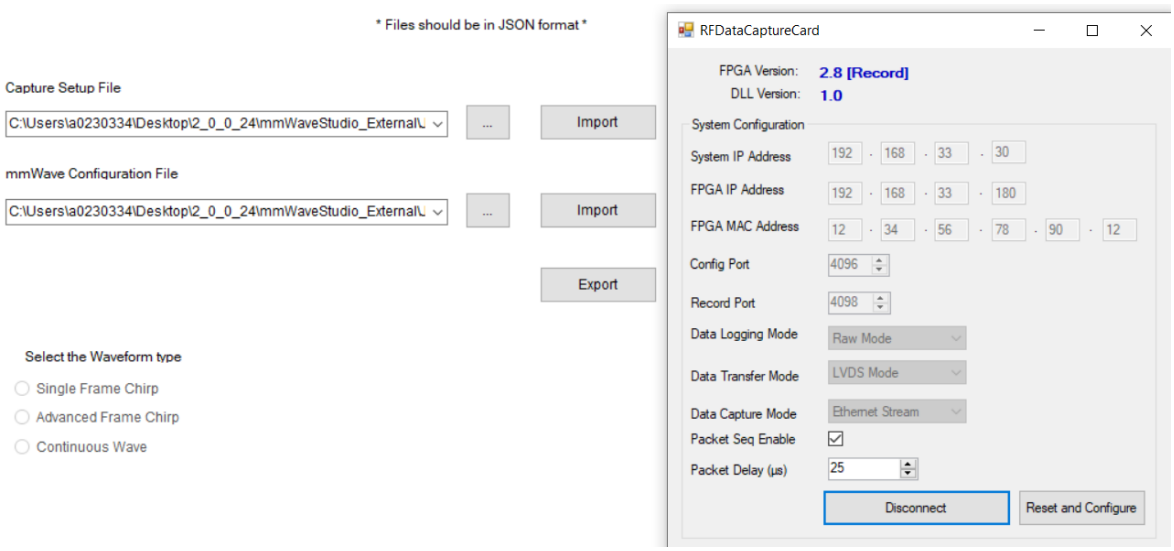


Figure 20.2 Capture Import

In the above snapshot, the paths for the Capture setup file and mmWave configuration file are updated.

20.2.2 Setup

It configures the Setup and Capture fields with the loaded configuration. It does the following configurations in sequence.

1. Reset Control
2. RS232 Operations
3. BSS Firmware download
4. MSS Firmware download
5. SPI Connect
6. RF Power Up
7. DCA1000 Capture Card configuration setup. (if DCA1000 is used).

20.3 mmWave Device JSON Operations

The following features are supported by the mmWave Device section of JSON Parser in mmWaveStudio.

20.3.1 Import

It performs various checks before populating the configuration in the appropriate fields.

At the trigger of Import, the deviceID and the corresponding waveform type will be displayed based on the input mmwave configured used.

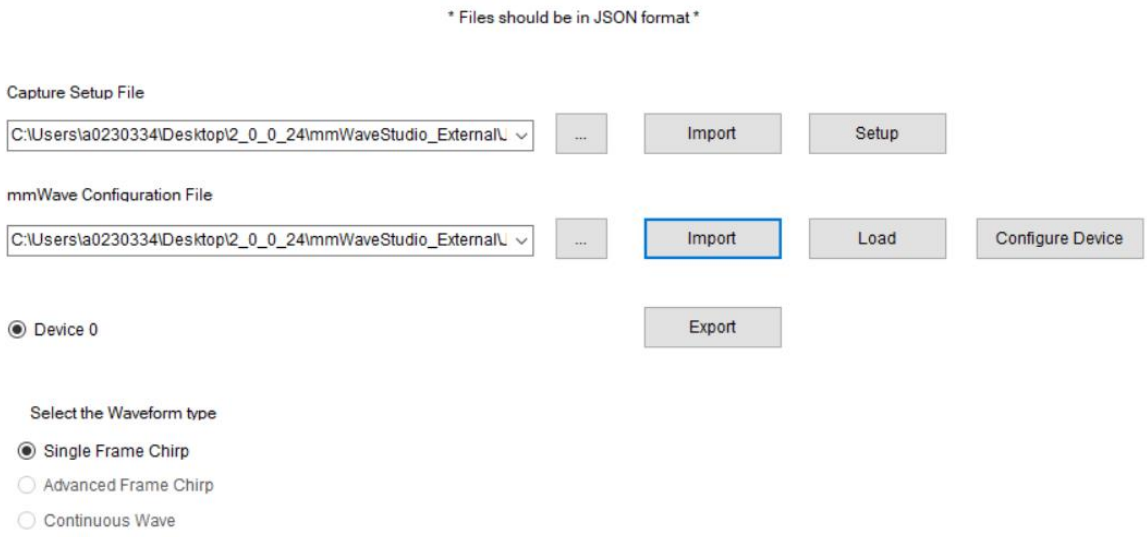


Figure 20.3 mmWave Import

A message will be displayed in the Output log after successful completion of the import operation.

20.3.2 Load

It loads the mmWave Device specific configuration in the appropriate tabs.

After loading the configuration, the user can either use “Configure Device” to execute all of the configurations present in input mmwave configuration file at one click, or choose to manually execute selective configurations in various tabs.

A message will be displayed in the Output log after successful completion of the Load operation.

20.3.3 Configure Device

It configures the mmWave Device with the loaded configuration.

Note: It executes only those configurations that are present in the input mmwave configuration file.

A message will be displayed in the Output log after successful completion of the Configure Device operation.

20.4 Capturing Raw ADC Data

After successful execution of “Configure Device”, there will be a popup display instructing the user to choose the Raw ADC Output bin file in the Sensor Configuration Tab.

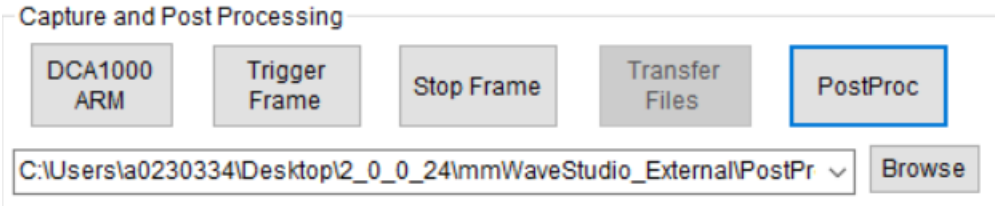


Figure 20.4 Capture and Post Processing

The sequence of steps for proper execution is as follows:

1. Select the name and path of the Raw ADC Output bin file.
2. Click on DCA1000 ARM configuration.

Note: On clicking DCA1000 ARM, the name and path of the Raw ADC Output bin file will be updated automatically if it is explicitly provided in the input Capture Setup JSON file.

3. Trigger the frame.
4. Stop Frame (required only in the case of infinite frames)
5. Post Process the captured raw ADC data. After a while, the post processed output will be displayed.
6. For subsequent capture of ADC data, go to step 1. It is advised to provide different file names for subsequent captures as it prevents the loss of existing ADC data.

20.5 Export

It exports the Capture Setup and mmWave Device specific configuration that is either performed from the GUI or is a part of the input JSON file. The user can export the configuration at any instant.

On clicking Export, it displays two “Save As” pop-up windows in a sequence.

1. The first window is used to export the Capture Setup Configuration with .setup.json as the extension.

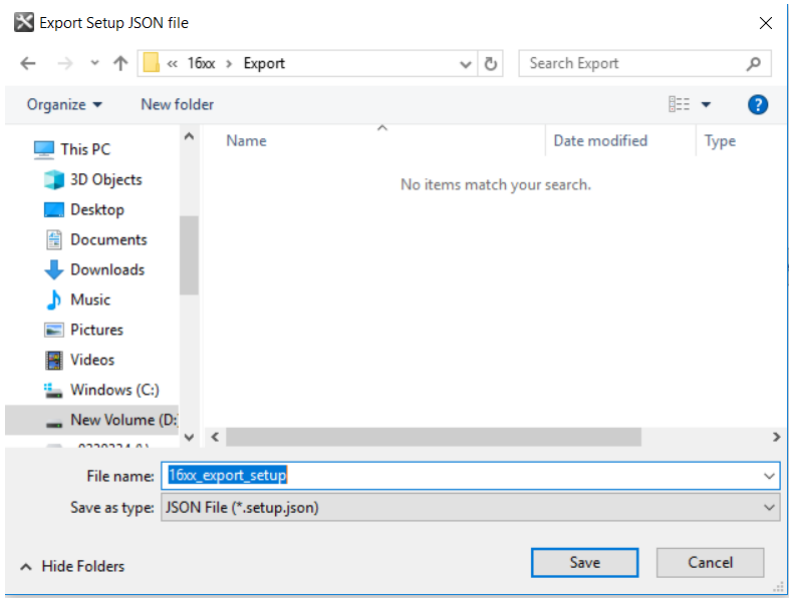


Figure 20.5 Export Capture File

2. The second window is used to export the mmWave Device Configuration with .mmwave.json as the extension.

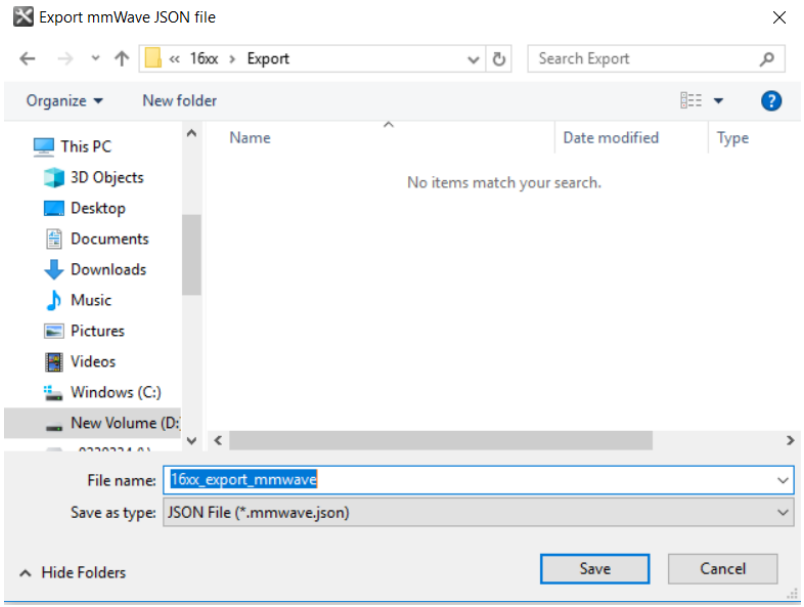


Figure 20.6 Export mmWave File

20.6 LUA API's for JSON Parser

20.6.1 Capture Setup API's

- **Int32 ar1.CaptureImport(**String capturePath**)**

  String capturePath: absolute path of the input setup configuration file.

  It is used for importing the Capture Setup JSON Configuration file.

- **Int32 ar1.ConfigureSetup()**

  It is used for configuring the Capture Card and Setup related fields.

20.6.2 mmWave Configuration API's

- **Int32 ar1.JsonImport(**String jsonFilePath**)**

  String jsonFilePath: absolute path of the input mmWave configuration file.

  It is used for importing the mmWave JSON Configuration file.

- **Int32 ar1.JsonLoad(**Int32 devId**)**

  Int32 devId: Device ID for which the configuration is to be loaded.

  It is used to populate device specific configuration from the JSON file onto various tabs for the device specified.

- **Int32 ar1.JsonExecute(**Int32 devId**)**

  Int32 devId: ID for the device to be configured.

  It is used to configure a device with the loaded configuration from JSON file.

- **Int32 ar1.JsonExport(**String jsonFilePath_Capture,String jsonFilePath_mmwave**)**

  String jsonFilePath_Capture: absolute path of the output setup configuration file.

  String jsonFilePath_mmwave: absolute path of the output mmWave configuration file.

  It exports the configuration present in the input mmWave configuration file and the configuration corresponding to commands executed by the user.

21. Radar post processing

The radar post processing (or PostProc) tool is used to visualize the adc data collected on the PC. It works using two inputs - the first is the raw LVDS capture (stored as .bin file) and the second is the sequence of APIs that were used to program the radar device (stored as .log file). Using these two sources of information, PostProc is able to interpret the data that the radar device collects and provide meaningful plots.

Note that mmWaveStudio automatically provides the dump file, and the sequence of APIs to the Matlab PostProc tool. So there is no need to manually provide these two inputs.

The following image shows how PostProc looks (with annotations) when you launch it first:

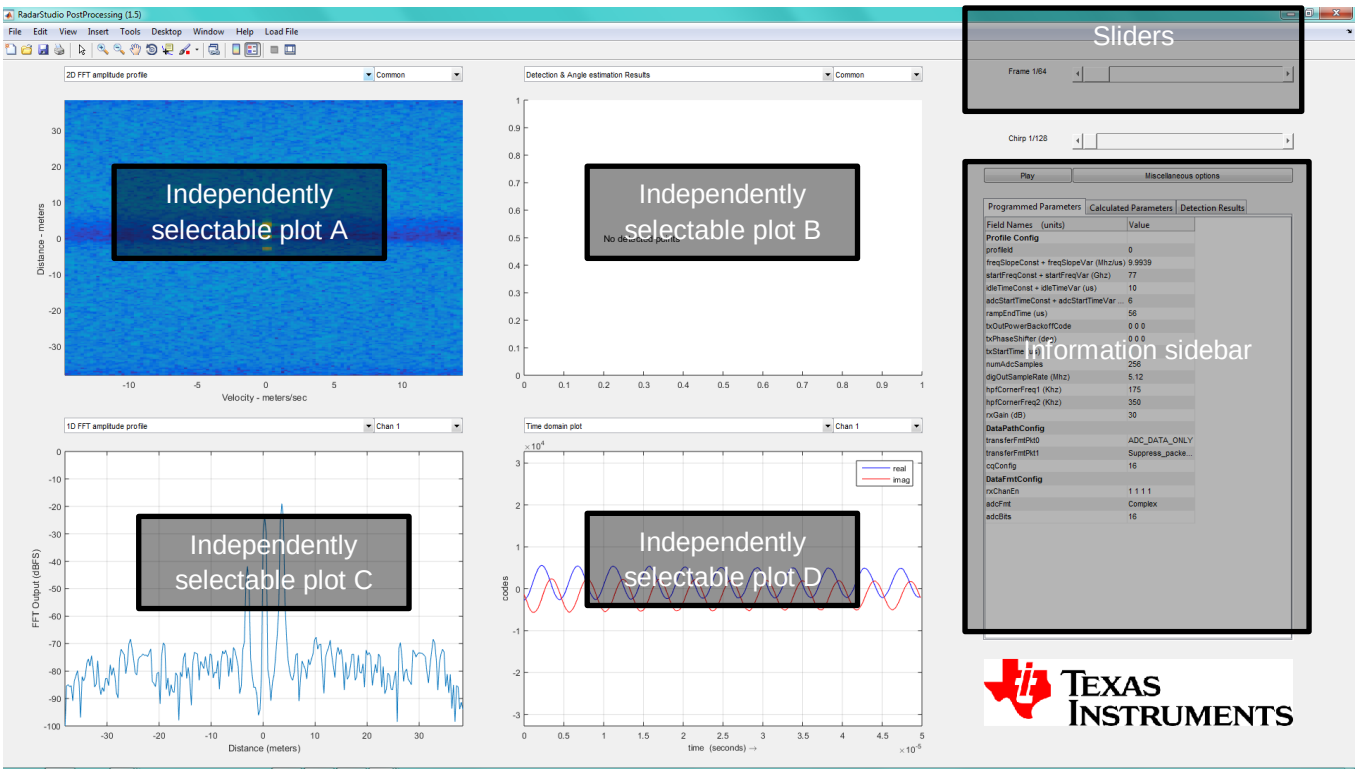


Figure 21.1. PostProc Basic

There are four different independently selectable plots. Each of these plots can be configured to different channels and different kinds of plots. Some of these plots are valid for all channels and some are simply a common measurement for all channels. Some plots are per-chirp and others are per frame. Each optionally is described later in the ‘List of plots’ section.

There is a set of sliders (on the top-right) that allow one to move around in the captured scenario, and display any frame, as well any of the chirps of the frame. If you are using the advanced frame configuration, the sub-frames, the bursts, the burst-loops are also individually selectable using the sliders.

Below the sliders, are two buttons, the ‘play’ button ‘plays’ the captured scenario, moving through each frame and updating the plots. ‘Playing’ through a captured scenario is a good way to see and correlate the captured scenario with what the different plots show, and see if there are issues.

While the default plots are a useful way to understand the radar data, some modification (to the plots) is sometimes required. So a number of ‘less-common’ options to modify the different plots are provided in a separate menu. This menu can be accessed using the ‘Miscellaneous options’ button.

Finally, below the ‘Play’ and ‘Miscellaneous options’ buttons is series of spreadsheets which provide the important parameters that were used to plot these graphs.

21.1 List of plots

Note: This list will be updated as more capability is added to PostProc.

21.1.1 Basic plots

21.1.1.1 2D FFT profile

The Range-velocity ‘mesh’ graph computed by performing a 2D-FFT of one frame of the captured scenario is shown in this plot. The x-axis shows the velocity (in meters/second) and the y-axis shows the range (in meters). Strong-reflectors are shown in brighter colours, and the noise floor is shown in dark blue.

The 2D output of each channel can be independently selected using the channel selector. If the ‘common’ option is selected in the channel selector, then the non-coherent-sum of the 2D FFT output across channels is plotted.

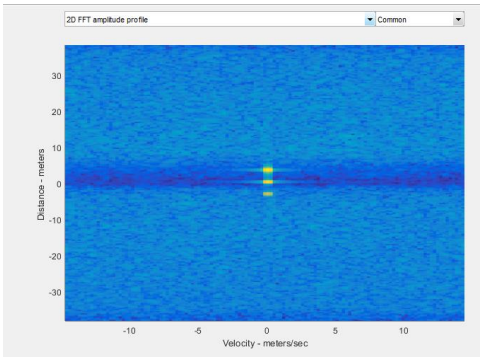


Figure 21.2. 2D FFT Profile

21.1.1.2 Range Angle plot

This plot shows a top-down view of the range-angle mesh. Targets are shown in brighter colours. The processing assumes a 1-Tx 4-Rx antenna configuration, with all 4 Rx antennas lying on the same plane (separated by $\lambda/2$, where λ is the starting wavelength of the FMCW ramp).

No calibration is performed in this step. Note that since only 4 antennas are used in the 3rd dimension processing, only a crude estimation of the angles are possible.

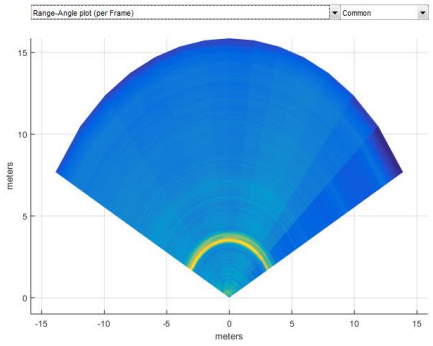


Figure 21.3. Range Angle plot

21.1.1.3 Detection and Angle estimation Results

A simple 2D-CFAR-CA (Constant false alarm – cell averaging) algorithm is used to detect targets in the scene. The parameters of the CFAR can be modified in the 'Miscellaneous Options' menu.

Targets are shown as coloured dots.

- Red dots indicate that the target has negative velocity,
- Green dots imply zero velocity
- Blue dots indicate positive velocity.

As in the range-angle plot, the antenna configuration assumes 4-Rx antennas separated by $\lambda/2$ organised as a linear array, the third dimension processing is simply a 3D-FFT followed by a peak search. Hence only a single target is detected per range-velocity bin.

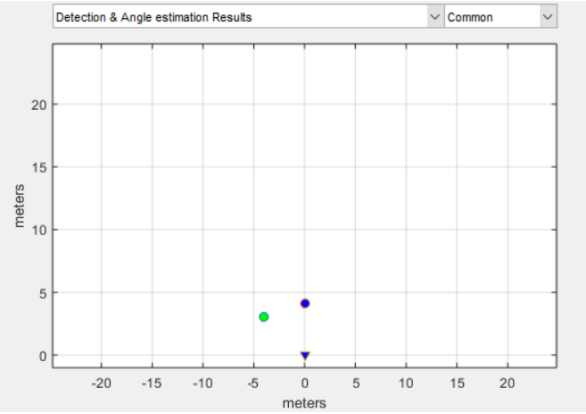


Figure 21.4. Detection and Angle Estimation results

21.1.1.4 Chirp Config Picture

This is a graphical depiction of the chirps that have been configured on the device. The x-axis is time (in seconds), and the y-axis is the ramp frequency. Using the 'miscellaneous options' menu, the plot can be modified to show all the chirps of a complete frame.

Since the chirp is common to all the rx channels on the device, only ‘common’ option is allowed on the channel selection dropdown.

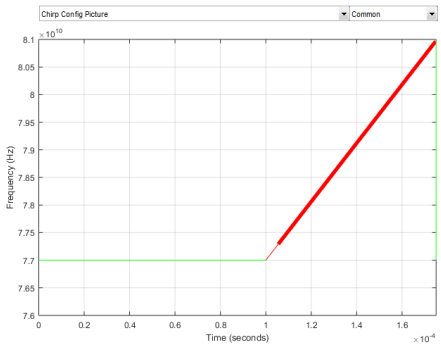


Figure 21.5. Chirp Config Picture

21.1.1.5 1D FFT amplitude profile

This is the ‘1D-FFT’ amplitude profile. The x-axis can be configured (using the ‘miscellaneous options’ menu) to be in meters or in Hz (IF frequency) or in samples. The y-axis is given in dBFS.

Also, in the ‘miscellaneous options’ menu are options to do ‘non-coherent’ sum across chirps (per frame) and across antennas, as well as options to select the window (by default the Hann window is used).

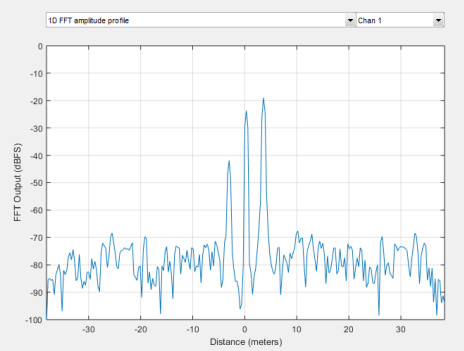


Figure 21.6. 1D FFT Profile

21.1.1.6 Time domain plot

This plot has the ‘Raw ADC data’ per chirp is plotted. The x-axis can be either time or ‘instantaneous ramp frequency’ or ‘sample number’. The y-axis is in either ADC codes or in (1/full-scale).

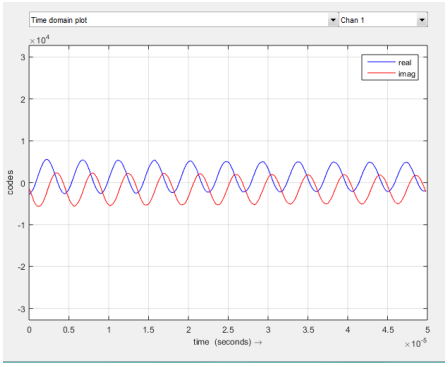


Figure 21.7. Time domain plot

21.1.2 CQ plots (chirp quality metrics)

Chirp quality metrics are used to identify regions of a chirp affected by interference. It is still under development. The metrics used are namely

1. Wide band energy monitor
2. ADC/IF Saturation Indicator
3. DFE Energy monitor.

21.1.3 Characterization plots

21.1.3.1 Phase stability across chirps

Phase stability is an important concern when measuring vibration frequency, and also in low velocity measurement. This plot shows the phase (of the strongest reflector) as a function of the chirps of a frame. The x-axis is in 'chirp number' and the y-axis is in degrees. Note that the y-axis label also tells where the strongest reflector is present (in meters).

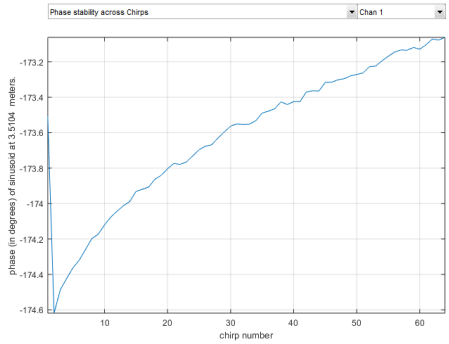


Figure 21.8. Phase Stability

21.1.3.2 Amplitude stability across chirps

Amplitude stability is a good measure of how stable the transmit power is across a Frame. This plot shows the amplitude of the strongest tone in the chirp. The position of the strongest reflector (in meters) is shown in the y-axis label.

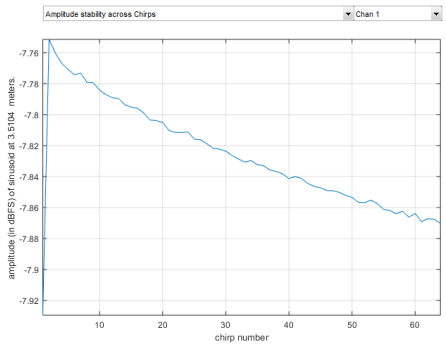


Figure 21.9. Amplitude stability

21.1.3.3 Zero-velocity bin vs High velocity bin

In a stationary scene, all energy from reflectors will be concentrated on the zero-velocity range bins. High-velocity range-bins will essentially be showing the thermal noise floor. This plot displays the Zero-velocity and the High velocity bins for an easy estimate of the SNR.

The 'high velocity bin' graph is generated from the '2D-FFT' by averaging the velocity bins from '$max_{velocity}/2$' to '$max_{velocity}$' and '$-max_{velocity}/2$' to '$-max_{velocity}$'.

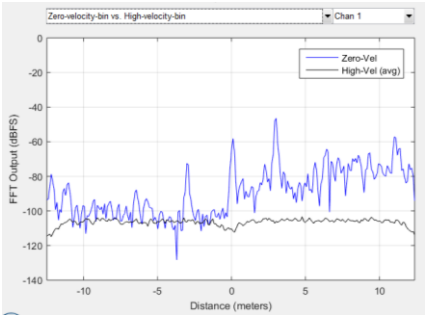


Figure 21.10. Zero-velocity bin vs High velocity bin.

21.2 Channel select

Some of these plots use all the Rx antennas to create the plot (1a, 1b, 1c, 2a, 2b, 2c), some have a per Rx measurement (1e, 1f, 3a, 3b), some aren't associated with the Rx plot (1d).

For measurements that are associated with a particular Rx, the channel select option can be used to select the channel whose measurement is to be shown.

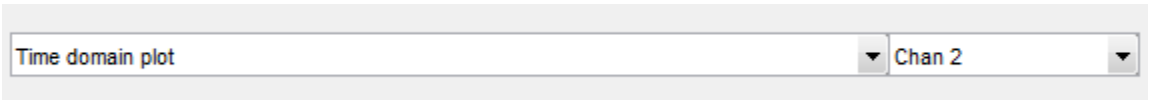


Figure 21.11. Channel select

PostProc will automatically select the first channel when changing from a common measurement to a per-channel measurement and vice-versa.

21.3 Frame /Profile /Chirp sliders

These slides allow exploring the different frames of the ADC dump. Within a frame, the different chirps can also be selected. If multiple ‘chirp config’ are selected as part of the frame, each is shown separately in the profile slider. Note that if the Advanced Frame Config is used, additional sliders will appear to allow the selection of ‘sub-frames’, ‘bursts’, ‘burst-loops’ which are configurable only through the Advanced Frame Config.

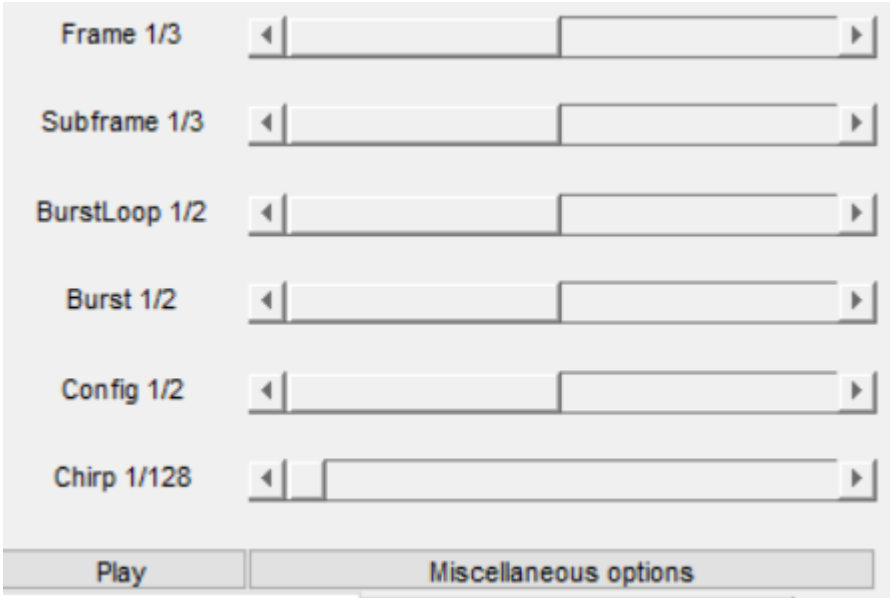
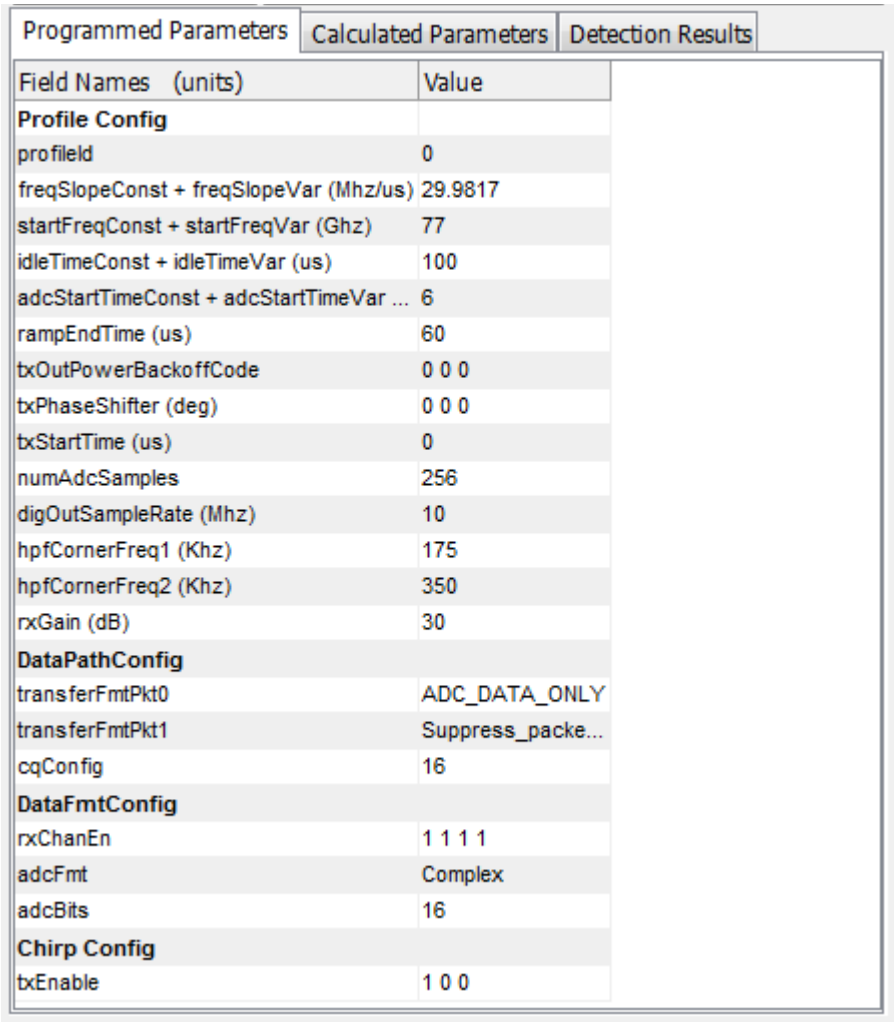


Figure 21.12. Sliders

21.4 Information sidebar

The sidebar has (as of now) three separate tabs.



Field Names (units)	Value
Profile Config	
profileId	0
freqSlopeConst + freqSlopeVar (Mhz/us)	29.9817
startFreqConst + startFreqVar (Ghz)	77
idleTimeConst + idleTimeVar (us)	100
adcStartTimeConst + adcStartTimeVar ...	6
rampEndTime (us)	60
txOutPowerBackoffCode	0 0 0
txPhaseShifter (deg)	0 0 0
txStartTime (us)	0
numAdcSamples	256
digOutSampleRate (Mhz)	10
hpfCornerFreq1 (Khz)	175
hpfCornerFreq2 (Khz)	350
rxGain (dB)	30
DataPathConfig	
transferFmtPkt0	ADC_DATA_ONLY
transferFmtPkt1	Suppress_packe...
cqConfig	16
DataFmtConfig	
rxChanEn	1 1 1 1
adcFmt	Complex
adcBits	16
Chirp Config	
txEnable	1 0 0

Figure 21.13. Information sidebar

These tabs are explained in the subsequent subsections.

21.4.1 Programmed parameters

This gives a short list of the parameters that were programmed using the APIs. This is the data that the post-proc tool uses to analyse the data.

21.4.2 Calculated parameters

This is a useful list of parameters that are calculated using the programmed parameters. They include

d. True start frequency ramp. The programmed start frequency doesn't take into account the ADC start time. When the ADC start time is taken into consideration, the true start of the ramp happens a few microseconds after the programmed start.

e. Bandwidth. The bandwidth provided here correctly takes into account the ADC start time and the number of samples to be collected.

f. Range resolution

g. Velocity resolution

21.4.3 Detection results

Up to five detected measurements are shown, along with their SNR, velocities and ranges.

21.5 Play

If the ADC data has multiple frames, the play button plays through the frames one by one, allowing one to see the evolution of the scene.

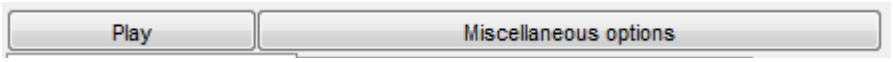


Figure 21.14. Play

21.6 Miscellaneous options

This window holds a list of options to modify and alter the plots. Options are organised in different ‘boxes’, with each box corresponding to a particular type of plot. The current ‘boxes’ are.

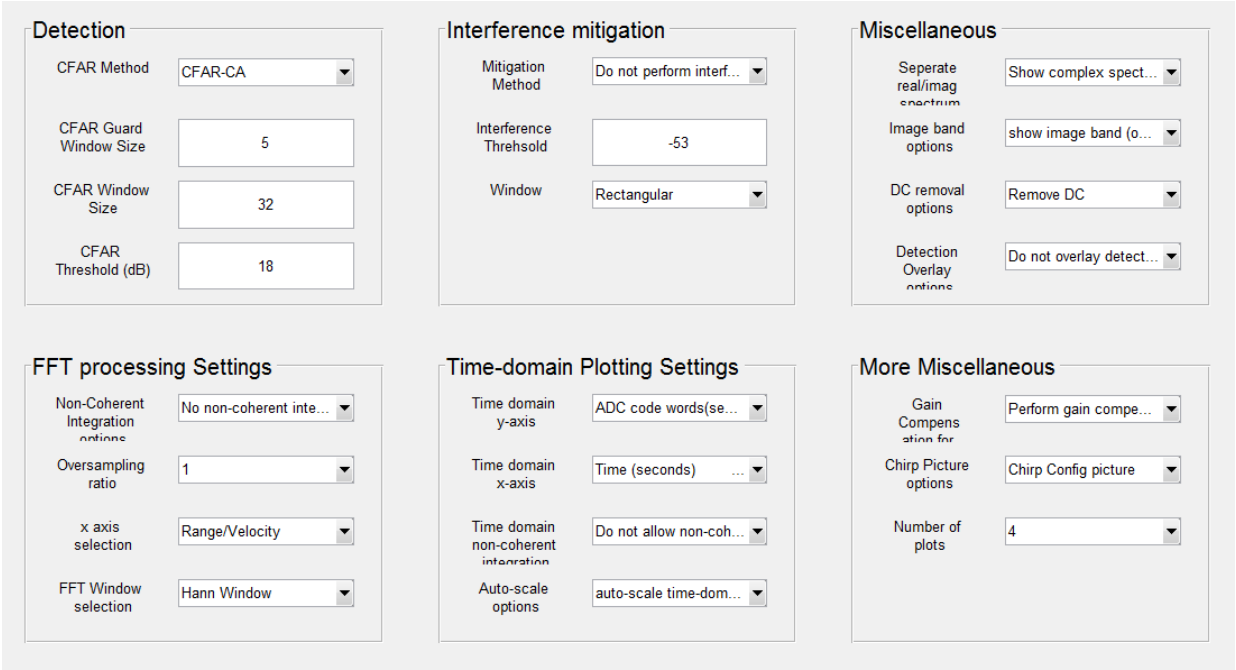


Figure 21.15. Miscellaneous options

21.6.1 Detection

The detection algorithm is 2D-CFAR-CA.

a. **CFAR method.** There are three different variants of CFAR that can be selected.

iii. CFAR-CA computes the noise floor considering samples that lie in a window that spans both sides of the CUT (cell under test). The noise floor is simply the mean of the energy in that window.

iv. CFAR-CA-SO selects the 'smaller of' (hence -SO) the noise-floor computed using samples that lie on left-side of the CUT and the noise floor computed using samples that lie on the right of the CUT.

v. CFAR-CA-GO selects the 'greater of' (hence -GO) the noise-floor computed using samples that lie on left-side of the CUT and the noise floor computed using samples that lie on the right of the CUT.

b. **CFAR guard-window size.** The guard window is used to reduce the effect of adjacent targets (or leakage) when measuring the noise floor. Essentially the samples that lie within the guard-window are omitted from the computation of the noise-floor.

c. **CFAR window size.** The size of the window determines the number of samples that are used to compute the noise-floor. The larger the window size, the better the noise floor estimate. However, the larger the window, the more probable that multiple targets will occur in the same window and result in a raised noise floor.

d. **CFAR Threshold.** Once the noise floor has been computed, it is compared against the CUT. If the CUT is greater than the noise-floor by the CFAR threshold, then CUT is declared as a valid target. Reducing the detection threshold will help increase the number of detected targets.

21.6.2 FFT processing settings

a. **Non-coherent Integration options**. Allow non-coherent integration across chirps and across chirps and antennas for the '1D FFT amplitude plots' and the 'time domain plots'. Allowing non-coherent integration improves the 'look' of the noise floor (in the '1D FFT amplitude profile'), and shows the envelope of complex signals in the time domain plot.

b. **Oversampling ratios**. Setting the 'oversampling ratio' higher than one creates zero-padded FFTs which improve bin-resolution (Note : zero-padding is done only in range not in velocity). Be aware that setting larger oversampling ratios, require proportionally more RAM to compute the FFT and also to display it.

c. **x-axis selection**. Changes the x-axis from meters (default) to hz. This is useful when doing looking at the IF spectrum, as opposed to an actual target.

d. **FFT-window selection**. Different FFT windows have different uses. By default we use the Hann window, which has a reasonable tradeoff between main lobe expansion and suppression of side-lobes. There are three other options, no-window (no energy loss, least main lobe expansion, no suppression of side lobes), and Blackman-Harris (high main lobe expansion, better suppression of side lobes (as compared to Hann)) and flat-top window (primarily used in characterization).

21.6.3 Miscellaneous

e. **Separate real/imag**. In the '1-D FFT amplitude profile' plot optionally separate the imaginary and real parts of the FFT.

f. **Image band options**. In the 'complex 2X' mode, the image band (i.e. negative frequencies) is visible. However, since no target can lie 'behind' the radar, the image band is of no interest in field tests. It is only useful in interference detection, and in monitoring.

g. **DC removal option.** This option can remove DC on a per-frame basis. By default DC is removed as it will result in cleaner images.

h. **Detection overlay options**. Allow the detection results to be displayed over the '2D FFT profile' plot, and the range-angle plot.

21.6.4 Time domain options

a. **Time domain x-axis**. The x-axis can be changed from time (seconds) to 'instantaneous ramp frequency' (Hz) or 'sample number'.

b. **Time domain y-axis**. The y-axis can be changed from ADC codewords to 1/fullscale.

c. **Time domain non-coherent integration options.** In conjunction with non-coherent integration options can construct the envelope of a complex time domain signal by computing $\sqrt{I^2 + Q^2}$

d. **Auto scale options**. The y-axis can be optionally allowed to auto-scale. This is useful for cases where the ADC swing is very low.

21.6.5 More miscellaneous options

a. **Window compensation options**. Applying a window prior to FFT causes a change in the absolute output level of the FFT (This change is called the windowing loss). These set of options allow two different methods of compensating for the windowing loss – Gain-compensation or Energy-compensation.

   Both methods differ only in the compensation factor. When using Gain-compensation the each fft output is divided by a factor given by $\sum_{i=0}^{N-1} w_i/N$.

   When using Energy-compensation the factor is given by $rms([w_0\ w_1\ \ldots w_{N-1}])$.

b. **Chirp picture options**. This option selects between different plots in the 'Chirp Picture'. For example a 'single chirp's picture' can be plotted or a 'complete frame's worth of chirps' can be plotted.

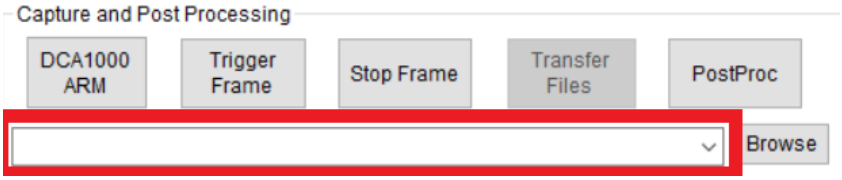
c. **Number of plots.** By default 4 plots are shown on the main window. This dropdown can change that to 1, 2, 4, or 9 independently configurable plots.

21.7 Notes on Setting up PostProc for Characterization

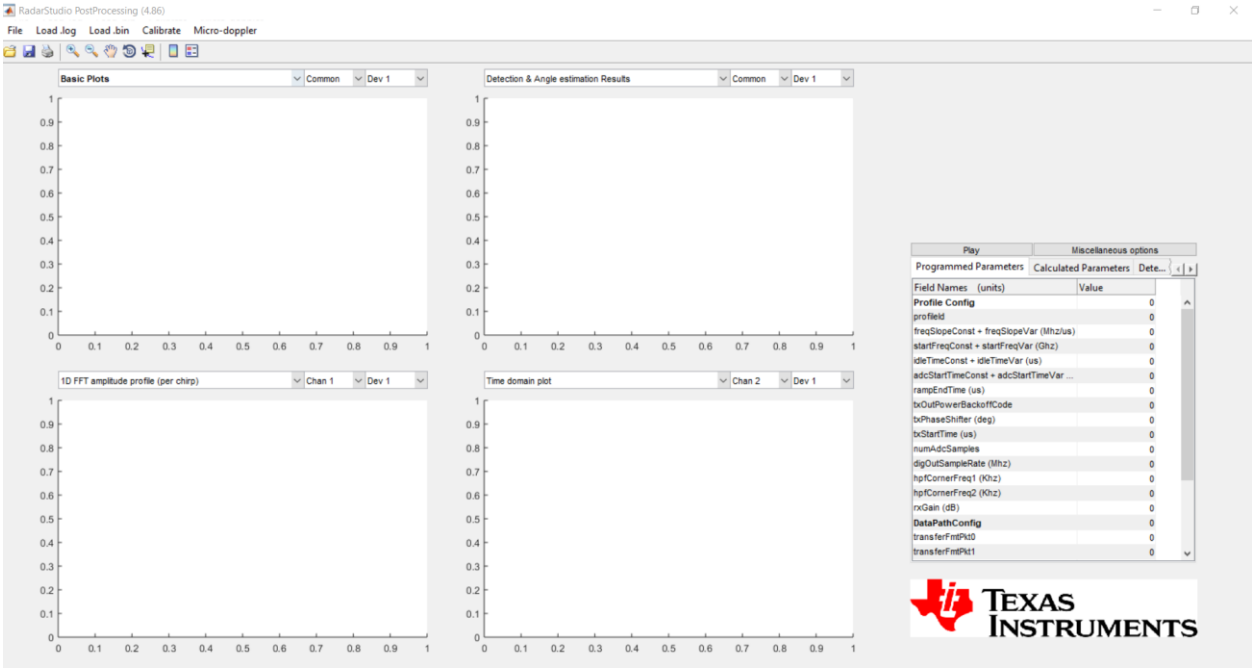
The tool is 'by default' configured for characterization so no changes need to be done. There is no reason to access the 'miscellaneous options' menu.

21.8 Standalone processing (post processing done in a different session from configuration and capture)

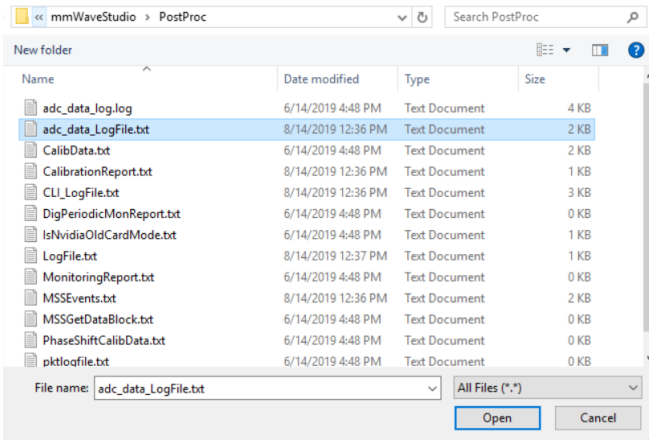
- This is particularly useful when the data is already captured from a previous session and the user just needs to post process the data using Studio.
- Go to the “Sensor Config” tab directly once Studio is opened.
- Provide an empty path (like in the screenshot below) and click on “PostProc” button.



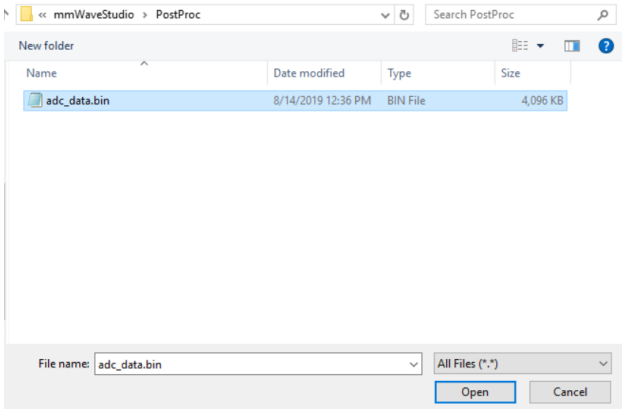
- An empty Post processing window will be opened



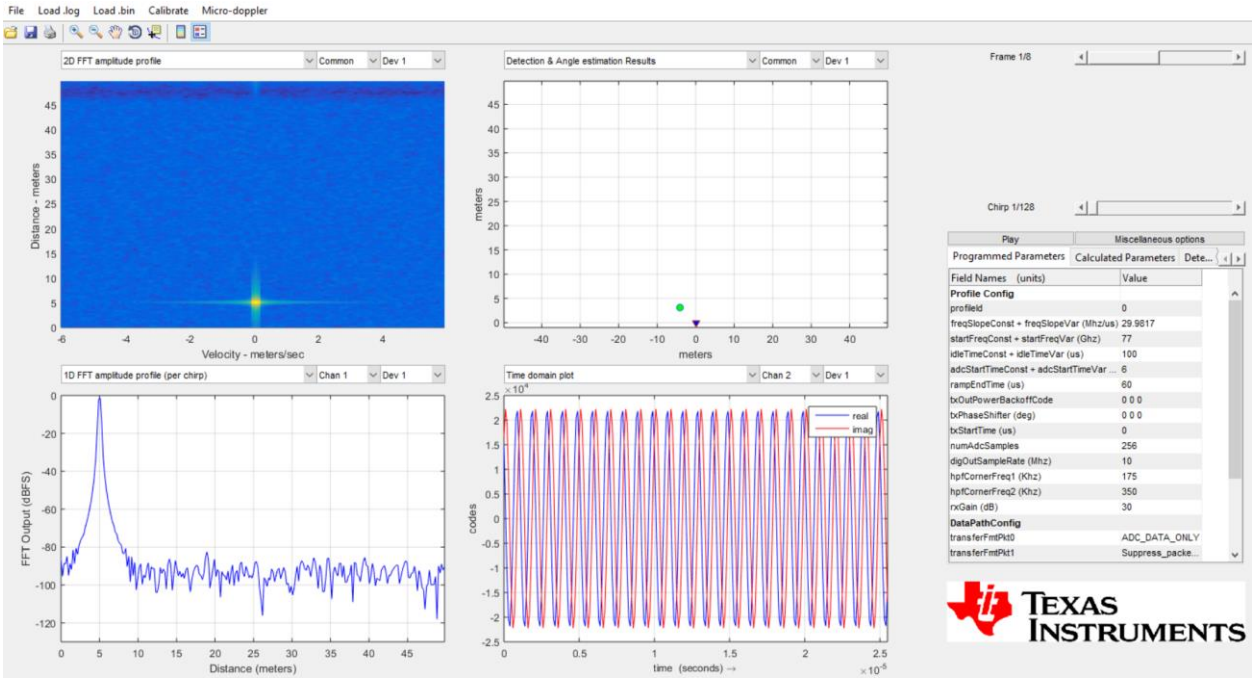
- Click on “Load .log” on the toolbar and select the corresponding log file.



- Click on “Load .bin” on the toolbar and select the corresponding bin file.



- The basic plots will be obtained for the bin file and the log file selected



22. Controlling mmWaveStudio from Matlab

1. From mmWaveStudio's Lua Shell, call the RSTD.NetStart() command

   This command starts a listening server on port 2777 by default.

   (you can also add this command to "C:\ti\mmwave_studio_01_00_00_01\mmWaveStudio\Scripts\Startup.lua" so that it is called automatically)

2. There are two Matlab scripts shown below. Copy them and create scripts with names RSTD_Interface_Example.m and Init_RSTD_Connection.m. You can modify the path to the RtttNetClientAPI.dll in RSTD_Interface_Example.m, and then call the RSTD_Interface_Example from Matlab. This script demonstrates how to connect to mmWaveStudio from Matlab and allow Lua commands to be sent to mmWaveStudio. The script internally calls Init_RSTD_Connection.m to establish

the connection to port 2777, and then displays a green message in the mmWaveStudio Output window.

See lines 41-47 in Init_RSTD_Connection.m for an example of sending a single Lua Command. Basically, you construct the Lua command as a string in Matlab and pass it to the RtttNetClientAPI.RtttNetClient.SendCommand API. Also see commented lines 12-15 in RSTD_Interface_Example.m for an example on how to get mmWaveStudio to run an external Lua Script.

RSTD_Interface_Example.m

```
%% RSTD_Interface_Example.m
addpath(genpath('.\'))

% Initialize mmWaveStudio .NET connection
RSTD_DLL_Path =
'C:\ti\mmwave_studio_01_00_00_01\mmWaveStudio\Clients\RtttNetClientControl
ler\RtttNetClientAPI.dll';

ErrStatus = Init_RSTD_Connection(RSTD_DLL_Path);
if (ErrStatus ~= 30000)
    disp('Error inside Init_RSTD_Connection');
    return;
end

%Example Lua Command
%strFilename =
'C:\\ti\\mmwave_studio_01_00_00_01\\mmWaveStudio\\Scripts\\Example_script_
AllDevices.lua';
%Lua_String = sprintf('dofile("%s")',strFilename);
%ErrStatus =RtttNetClientAPI.RtttNetClient.SendCommand(Lua_String);
```

Init_RSTD_Connection.m

```
%% Init_RSTD_Connection.m
function ErrStatus = Init_RSTD_Connection(RSTD_DLL_Path)
%This script establishes the connection with mmWaveStudio software
%   Pre-requisites:
%   Type RSTD.NetStart() in mmWaveStudio Luashell before running the script.
This would open port 2777
%   Returns 30000 if no error.
if (strcmp(which('RtttNetClientAPI.RtttNetClient.IsConnected'),'')) %First
time the code is run after opening MATLAB
    disp('Adding RSTD Assembly');
    RSTD_Assembly = NET.addAssembly(RSTD_DLL_Path);
    if ~strcmp(RSTD_Assembly.Classes{1},'RtttNetClientAPI.RtttClient')
        disp('RSTD Assembly not loaded correctly. Check DLL path');
        ErrStatus = -10;
        return
    end
    Init_RSTD_Connection = 1;
elseif ~RtttNetClientAPI.RtttNetClient.IsConnected() %Not the first time but
port is disconnected
% Reason:
```

```
% Init will reset the value of Isconnected. Hence Isconnected should be checked before Init
% However, Isconnected returns null for the 1st time after opening MATLAB (since init was never called before)
    Init_RSTD_Connection = 1;
else
    Init_RSTD_Connection = 0;
end

if Init_RSTD_Connection
    disp('Initializing RSTD client');
    ErrStatus = RtttNetClientAPI.RtttNetClient.Init();
    if (ErrStatus ~= 0)
        disp('Unable to initialize NetClient DLL');
        return;
    end
    disp('Connecting to RSTD client');
    ErrStatus = RtttNetClientAPI.RtttNetClient.Connect('127.0.0.1',2777);
    if (ErrStatus ~= 0)
        disp('Unable to connect to mmWaveStudio');
        disp('Reopen port in mmWaveStudio. Type   RSTD.NetClose() followed by RSTD.NetStart()')
        return;
    end
    pause(1);%Wait for 1sec. NOT a MUST have.
end

disp('Sending test message to RSTD');
Lua_String = 'WriteToLog("Running script from MATLAB\n", "green")';
ErrStatus = RtttNetClientAPI.RtttNetClient.SendCommand(Lua_String);
if (ErrStatus ~= 30000)
    disp('mmWaveStudio Connection Failed');
end
disp('Test message success');
end
```

23. Automation using LUA

mmWaveStudio allows a Lua script to be executed through command line (it will open the GUI). Below steps explain the procedure

1. Navigate to the Runtime folder where mmWaveStudio executable is present.
2. Open the command prompt and type

mmWaveStudio.exe /lua <path to the Lua script>

23.1 Sample Lua script for automation

A sample Lua script named "*Automation.lua*" is present at the Scripts directory for reference.

In order to execute the script, the command to be provided is as follows:

```
C:\ti\mmwave_studio_02_01_00_00\mmWaveStudio\RunTime>mmWaveStudio.exe /lua .\..\Scripts\Automation.lua
```

- Every Lua script that needs to be executed should have the startup portion as shown below

```
----------------------------------- STARTUP ---------------------------------------
------------------------ DO NOT MODIFY THIS SECTION ----------------------------

-- mmwavestudio installation path
RSTD_PATH = RSTD.GetRstdPath()

-- Declare the loading function
dofile(RSTD_PATH .. "\\Scripts\\Startup.lua")
```

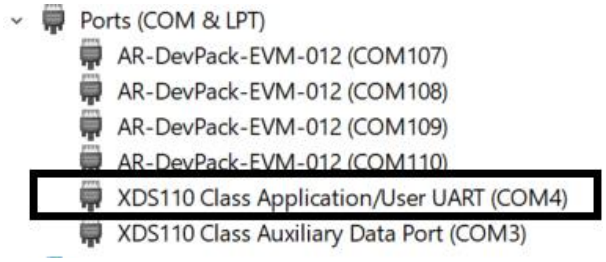
- The sample script shows how to do a basic capture using the DCA1000 EVM. The script can be extended for custom usage.

```
------------------------------ CONFIGURATIONS --------------------------------
-- Use "DCA1000" for working with DCA1000
capture_device  = "DCA1000"

-- SOP mode
SOP_mode        = 2

-- RS232 connection baud rate
baudrate        = 921600
-- RS232 COM Port number
uart_com_port   = 4
-- Timeout in ms
timeout         = 1000

-- BSS firmware
bss_path        = "C:\\ti\\mmwave_studio_02_01_00_00\\rf_eval_firmware\\radarss\\xwr12xx_xwr14xx_radarss.bin"
-- MSS firmware
mss_path        = "C:\\ti\\mmwave_studio_02_01_00_00\\rf_eval_firmware\\masterss\\xwr12xx_xwr14xx_masterss.bin"

adc_data_path   = "C:\\ti\\mmwave_studio_02_01_00_00\\mmWaveStudio\\PostProc\\test_data.bin"
```

a. The capture device should be specified as either "DCA1000" or "TSW1400" based on the capture device present.

b. The SOP mode has to be chosen appropriately. In this example, SOP2 is chosen.

c. The COM port and the baud rate should be chosen appropriately. COM port can be obtained through device manager.



d. The BSS and the MSS firmware paths.

e. ADC file path for the capture.

- Once the basic connection tab configuration is done, the device is configured done until the frame configuration.
- The DCA1000 capture card is setup and the device starts framing.
- The captured data is stored at the mentioned ADC file path.
- Once the API's are executed, the script exits with the closure of mmWaveStudio GUI.

```
------------------------ Exit MMwave Studio ---------------------------------
os.exit()
```

24. Format of the raw captured file

This section provides a quick description of the format of the binary files collected by TSW1400 EVM and DCA1000 EVM from the xWR12xx, xWR14xx and the xWR16xx devices. There are subtle differences between the data generated by xWR16xx and xWR12xx/xWR14xx, and also differences when the number of channels is changed, as well as when the number of LVDS lanes is changed, and when the type of data (from complex to real) is changed.

Each combination therefore needs to be separately described. Since the number of combinations is very large, only the most useful combinations are described.

Next, we describe the notation used in the rest of the section. We then describe the 'Sample Format' – how to interpret each sample. We then describe the arrangement of data for xWR16xx's capture first, followed by xWR12xx/xWR14xx's capture.

24.1 Preliminaries

24.2 Notation

- Rx***k***I***n*** : The **n**th in-phase sample corresponding to **k**th receive channel
- Rx***k***Q***n*** : The **n**th quadrature-phase sample corresponding to **k**th receive channel
- **N**: The number of samples per chirp

24.3 Sample Format

Every sample captured by TSW1400 is **2 bytes** long and in the 'offset binary format'. In other words, every sample has an extra 2^{15} added to it. To get the true '2s complement' number, perform the following operation on every sample 'x' read from the file.

$$x = x - 2^{15}$$

Data captured from DCA1000 does issue. Each sample is in 2s complement format.

24.4 TSW1400 EVM xWR16xx file formats

24.4.1 2 LVDS Lanes, complex data, variable number of channels, chirping mode

Chirp 1	Rx*0*I*0*	Rx*0*Q*0*	Rx*0*I*1*	Rx*0*Q*1*	…	Rx*0*I*N-1*	Rx*0*Q*N-1*
	Rx*1*I*0*	Rx*1*Q*0*	Rx*1*I*1*	Rx*1*Q*1*	…	Rx*1*I*N-1*	Rx*1*Q*N-1*
	Rx*2*I*0*	Rx*2*Q*0*	Rx*2*I*1*	Rx*2*Q*1*	…	Rx*2*I*N-1*	Rx*2*Q*N-1*
	Rx*3*I*0*	Rx*3*Q*0*	Rx*3*I*1*	Rx*3*Q*1*	…	Rx*3*I*N-1*	Rx*3*Q*N-1*
Chirp 2	Rx*0*I*0*	Rx*0*Q*0*	Rx*0*I*1*	Rx*0*Q*1*	…	Rx*0*I*N-1*	Rx*0*Q*N-1*
	Rx*1*I*0*	Rx*1*Q*0*	Rx*1*I*1*	Rx*1*Q*1*	…	Rx*1*I*N-1*	Rx*1*Q*N-1*
	Rx*2*I*0*	Rx*2*Q*0*	Rx*2*I*1*	Rx*2*Q*1*	…	Rx*2*I*N-1*	Rx*2*Q*N-1*
	Rx*3*I*0*	Rx*3*Q*0*	Rx*3*I*1*	Rx*3*Q*1*	…	Rx*3*I*N-1*	Rx*3*Q*N-1*

In the above diagram, each green block refers to one 16-bit number. Data is arranged in **row-major-order**. Note also, that each chirp consists of ***N*** complex samples.

If the number of enabled channels is two or one, then the rows corresponding to the disabled channels are removed in the above diagram. There is no support for three channels on 2 lanes. As an example with two channels (say, 1 and 3), then the 'data format' will look as follows.

Chirp 1	Rx*1*I*0*	Rx*1*Q*0*	Rx*1*I*1*	Rx*1*Q*1*	…	Rx*1*I*N-1*	Rx*1*Q*N-1*
	Rx*3*I*0*	Rx*3*Q*0*	Rx*3*I*1*	Rx*3*Q*1*	…	Rx*3*I*N-1*	Rx*3*Q*N-1*
Chirp 2	Rx*1*I*0*	Rx*1*Q*0*	Rx*1*I*1*	Rx*1*Q*1*	…	Rx*1*I*N-1*	Rx*1*Q*N-1*
	Rx*3*I*0*	Rx*3*Q*0*	Rx*3*I*1*	Rx*3*Q*1*	…	Rx*3*I*N-1*	Rx*3*Q*N-1*

24.4.2 2 LVDS Lanes, real data, variable number of channels, chirping mode

Chirp 1	Rx*0*I*0*	Rx*0*I*2*	Rx*0*I*1*	Rx*0*I*3*	…	Rx*0*I*N-2*	Rx*0*I*N-1*
	Rx*1*I*0*	Rx*1*I*2*	Rx*1*I*1*	Rx*1*I*3*	…	Rx*1*I*N-2*	Rx*1*I*N-1*
	Rx*2*I*0*	Rx*2*I*2*	Rx*2*I*1*	Rx*2*I*3*	…	Rx*2*I*N-2*	Rx*2*I*N-1*

	Rx*3I0*	Rx*3I2*	Rx*3I1*	Rx*3I3*	…	Rx*3IN-2*	Rx*3IN-1*
Chirp 2	Rx*0I0*	Rx*0I2*	Rx*0I1*	Rx*0I3*	…	Rx*0IN-2*	Rx*0IN-1*
	Rx*1I0*	Rx*1I2*	Rx*1I1*	Rx*1I3*	…	Rx*1IN-2*	Rx*1IN-1*
	Rx*2I0*	Rx*2I2*	Rx*2I1*	Rx*2I3*	…	Rx*2IN-2*	Rx*2IN-1*
	Rx*3I0*	Rx*3I2*	Rx*3I1*	Rx*3I3*	…	Rx*3IN-2*	Rx*3IN-1*

24.4.3 2 LVDS Lanes, complex data, variable number of channels, continuous streaming mode

Random number of samples	x	x	x	x	…	x	x	
	x	x	x	x	…	x	x	
	x	x	x	x	…	x	x	
	x	x	x	x	…	x	x	
Sync Pattern	$2047+2^{15}$	$2048+2^{15}$	$2047+2^{15}$	$2048+2^{15}$	$2047+2^{15}$	$2048+2^{15}$	$2047+2^{15}$	$2048+2^{15}$
Data Packet 1	Rx*0I0*	Rx*0Q0*	Rx*0I1*	Rx*0Q1*	…	Rx*0I1023*	Rx*0Q1023*	
	Rx*1I0*	Rx*1Q0*	Rx*1I1*	Rx*1Q1*	…	Rx*1I1023*	Rx*1Q1023*	
	Rx*2I0*	Rx*2Q0*	Rx*2I1*	Rx*2Q1*	…	Rx*2I1023*	Rx*2Q1023*	
	Rx*3I0*	Rx*3Q0*	Rx*3I1*	Rx*3Q1*	…	Rx*3I1023*	Rx*3Q1023*	
Sync Pattern	$2047+2^{15}$	$2048+2^{15}$	$2047+2^{15}$	$2048+2^{15}$	$2047+2^{15}$	$2048+2^{15}$	$2047+2^{15}$	$2048+2^{15}$
Data Packet 2	Rx*0I0*	Rx*0Q0*	Rx*0I1*	Rx*0Q1*	…	Rx*0I1023*	Rx*0Q1023*	
	Rx*1I0*	Rx*1Q0*	Rx*1I1*	Rx*1Q1*	…	Rx*1I1023*	Rx*1Q1023*	
	Rx*2I0*	Rx*2Q0*	Rx*2I1*	Rx*2Q1*	…	Rx*2I1023*	Rx*2Q1023*	
	Rx*3I0*	Rx*3Q0*	Rx*3I1*	Rx*3Q1*	…	Rx*3I1023*	Rx*3Q1023*	

In the continuous streaming mode, a sync packet precedes every LVDS data packet. In order to make sense of the data, first, find the sync packet and then collect $(n4)$kB, (where n is the number of channels). The next data packet begins after $(n4)$ kB. It will also have a sync pattern in its start.

The sync packet is actually just the following numbers with each number being 2 bytes long, and the entire packet being 128 bits.

2047	2048	2047	2048	2047	2048	2047	2048

However, since HSDC Pro converts the samples to ‘offset binary, we add 2^{15} to the sync pattern, and then search for it.

If the number of enabled channels is two or one, then the rows corresponding to the disabled channels are removed in the above diagram. There is no support for three channels on 2 lanes. As an example, the following diagram shows the data format when only two channels (1 and 3) are enabled.

Random number of samples	x	x	x	x	…	x	x	
	x	x	x	x	…	x	x	
	x	x	x	x	…	x	x	
	x	x	x	x	…	x	x	
Sync Pattern	2047 +2^{15}	2048+2^{15}	2047+2^{15}	2048+2^{15}	2047+2^{15}	2048+2^{15}	2047+2^{15}	2048+2^{15}
Data Packet 1	Rx*1*I*0*	Rx*1*Q*0*	Rx*1*I*1*	Rx*1*Q*1*	…	Rx*1*I*1023*	Rx*1*Q*1023*	
	Rx*3*I*0*	Rx*3*Q*0*	Rx*3*I*1*	Rx*3*Q*1*	…	Rx*3*I*1023*	Rx*3*Q*1023*	
Sync Pattern	2047 +2^{15}	2048+2^{15}	2047+2^{15}	2048+2^{15}	2047+2^{15}	2048+2^{15}	2047+2^{15}	2048+2^{15}
Data Packet 1	Rx*1*I*0*	Rx*1*Q*0*	Rx*1*I*1*	Rx*1*Q*1*	…	Rx*1*I*1023*	Rx*1*Q*1023*	
	Rx*3*I*0*	Rx*3*Q*0*	Rx*3*I*1*	Rx*3*Q*1*	…	Rx*3*I*1023*	Rx*3*Q*1023*	

24.5 TSW1400 xWR12xx/xWR14xx file format

24.5.1 n LVDS Lanes, complex data, n channels, chirping/continuous streaming mode

Chirp 1	Rx*0*I*0*	Rx*0*Q*0*	Rx*1*I*0*	Rx*1*Q*0*	Rx2I*0*	Rx*2*Q*0*	Rx*3*I*0*	Rx*3*Q*0*
	Rx*0*I*1*	Rx*0*Q*1*	Rx*1*I*1*	Rx*1*Q*1*	Rx*2*I*1*	Rx*2*Q*1*	Rx*3*I*1*	Rx*3*Q*1*
	…		…		…		…	…
	Rx*0*I*N-1*	Rx*0*Q*N-1*	Rx*1*I*N-1*	Rx*1*Q*N-1*	Rx2I*N-1*	Rx*2*Q*N-1*	Rx*3*I*N-1*	Rx*3*Q*N-1*
Chirp 2	Rx*0*I*0*	Rx*0*Q*0*	Rx*1*I*0*	Rx*1*Q*0*	Rx2I*0*	Rx*2*Q*0*	Rx*3*I*0*	Rx*3*Q*0*
	Rx*0*I*1*	Rx*0*Q*1*	Rx*1*I*1*	Rx*1*Q*1*	Rx*2*I*1*	Rx*2*Q*1*	Rx*3*I*1*	Rx*3*Q*1*
	…		…		…		…	…
	Rx*0*I*N-1*	Rx*0*Q*N-1*	Rx*1*I*N-1*	Rx*1*Q*N-1*	Rx*2*I*N-1*	Rx*2*Q*N-1*	Rx*3*I*N-1*	Rx*3*Q*N-1*

In the above diagram, each green block refers to one 16-bit number. Data is arranged in **row-major-order**. Note that the data format is a *transpose* of the xWR16xx format.

The data format remains unchanged in the 'continuous streaming' mode where one can think of the data collected as belonging to a single large chirp. There is no sync packet at the start of an LVDS packet, as there is one lane for each channel, so we will always know which lane has which channel.

If the number of enabled channels is less than four, then the columns corresponding to the disabled channels will have zeroes in it. For example, if only channels 1 and 3 are enabled, then the data format will look as follows.

Chirp 1	0	0	Rx*1*I*0*	Rx*1*Q*0*	0	0	Rx*3*I*0*	Rx*3*Q*0*
	0	0	Rx*1*I*1*	Rx*1*Q*1*	0	0	Rx*3*I*1*	Rx*3*Q*1*
	…		…		…		…	…
	0	0	Rx*1*I*N*	Rx*1*Q*N*	0	0	Rx*3*I*N*	Rx*3*Q*N*
Chirp 2	0	0	Rx*1*I*0*	Rx*1*Q*0*	0	0	Rx*3*I*0*	Rx*3*Q*0*
	0	0	Rx*1*I*1*	Rx*1*Q*1*	0	0	Rx*3*I*1*	Rx*3*Q*1*
	…		…		…		…	…
	0	0	Rx*1*I*N*	Rx*1*Q*N*	0	0	Rx*3*I*N*	Rx*3*Q*N*

24.5.2 *n* LVDS Lanes, real data, *n* channels, chirping/continuous streaming mode

Chirp 1	Rx*0*I*0*	Rx*0*I*1*	Rx*1*I*0*	Rx*1*I*1*	Rx2I*0*	Rx*2*I*1*	Rx*3*I*0*	Rx*3*I*1*
	Rx*0*I*2*	Rx*0*I*3*	Rx*1*I*2*	Rx*1*I*3*	Rx2I*2*	Rx*2*I*3*	Rx*3*I*2*	Rx*3*I*3*
	…		…		…		…	…
	Rx*0*I*N-2*	Rx*0*I*N-1*	Rx*1*I*N-2*	Rx*1*I*N-1*	Rx2I*N-2*	Rx*2*I*N-1*	Rx*3*I*N-2*	Rx*3*I*N-1*
Chirp 2	Rx*0*I*0*	Rx*0*I*1*	Rx*1*I*0*	Rx*1*I*1*	Rx2I*0*	Rx*2*I*1*	Rx*3*I*0*	Rx*3*I*1*
	Rx*0*I*2*	Rx*0*I*3*	Rx*1*I*2*	Rx*1*I*3*	Rx2I*2*	Rx*2*I*3*	Rx*3*I*2*	Rx*3*I*3*
	…		…		…		…	…
	Rx*0*I*N-2*	Rx*0*I*N-1*	Rx*1*I*N-2*	Rx*1*I*N-1*	Rx*2*I*N-2*	Rx*2*I*N-1*	Rx*3*I*N-2*	Rx*3*I*N-1*

24.6 *DCA1000 EVM capture format (xWR12xx/xWR14xx complex, 4 channel, 4 lanes [Interleaved])*

Chirp 1	Rx**0**I**0**	Rx**1**I**0**	Rx**2**I**0**	Rx**3**I**0**	Rx**0**Q**0**	Rx**1**Q**0**	Rx**2**Q**0**	Rx**3**Q**0**
	Rx**0**I**1**	Rx**1**I**1**	Rx**2**I**1**	Rx**3**I**1**	Rx**0**Q**1**	Rx**1**Q**1**	Rx**2**Q**1**	Rx**3**Q**1**
	…		…		…		…	…
	Rx**0**I**N-1**	Rx**1**I**N-1**	Rx**2**I**N-1**	Rx**3**I**N-1**	Rx**0**Q**N-1**	Rx**1**Q**N-1**	Rx**2**Q**N-1**	Rx**3**Q**N-1**
Chirp 2	Rx**0**I**0**	Rx**1**I**0**	Rx**2**I**0**	Rx**3**I**0**	Rx**0**Q**0**	Rx**1**Q**0**	Rx**2**Q**0**	Rx**3**Q**0**
	Rx**0**I**1**	Rx**1**I**1**	Rx**2**I**1**	Rx**3**I**1**	Rx**0**Q**1**	Rx**1**Q**1**	Rx**2**Q**1**	Rx**3**Q**1**
	…		…		…		…	…
	Rx**0**I**N-1**	Rx**1**I**N-1**	Rx**2**I**N-1**	Rx**3**I**N-1**	Rx**0**Q**N-1**	Rx**1**Q**N-1**	Rx**2**Q**N-1**	Rx**3**Q**N-1**

S

24.7 *DCA1000 EVM capture format (xWR12xx/xWR14xx real, 4 channel, 4 lanes [Interleaved])*

Chirp 1	Rx**0**I**0**	Rx**1**I**0**	Rx**2**I**0**	Rx**3**I**0**	Rx**0**I**1**	Rx**1**I**1**	Rx**2**I**1**	Rx**3**I**1**
	Rx**0**I**2**	Rx**1**I**2**	Rx**2**I**2**	Rx**3**I**2**	Rx**0**I**3**	Rx**1**I**3**	Rx**2**I**3**	Rx**3**I**3**

	…		…		…		…	…
	Rx0IN-2	Rx1IN-2	Rx2IN-2	Rx3IN-2	Rx0IN-1	Rx1IN-1	Rx2IN-1	Rx3IN-1
Chirp 2	Rx0I0	Rx1I0	Rx2I0	Rx3I0	Rx0I1	Rx1I1	Rx2I1	Rx3I1
	Rx0I2	Rx1I2	Rx2I2	Rx3I2	Rx0I3	Rx1I3	Rx2I3	Rx3I3
	…		…		…		…	…
	Rx0IN-2	Rx1IN-2	Rx2IN-2	Rx3IN-2	Rx0IN-1	Rx1IN-1	Rx2IN-1	Rx3IN-1

24.8 *DCA1000 EVM capture format (xWR16xx complex, 4 channel, 2 lanes [Non-Interleaved])*

Chirp 1	Rx0I0	Rx0I1	Rx0Q0	Rx0Q1	Rx0I2	Rx0I3	Rx0Q2	Rx0Q3
	Rx0I4	Rx0I5	Rx0Q4	Rx0Q5	Rx0I6	Rx0I7	Rx0Q6	Rx0Q7
	…		…		…		…	…
	Rx3IN-4	Rx3IN-3	Rx3QN-4	Rx3QN-3	Rx3IN-2	Rx3IN-1	Rx3QN-2	Rx3QN-1
Chirp 2	Rx0I0	Rx0I1	Rx0Q0	Rx0Q1	Rx0I2	Rx0I3	Rx0Q1	Rx0Q3
	Rx0I4	Rx0I5	Rx0Q4	Rx0Q5	Rx0I6	Rx0I7	Rx0Q6	Rx0Q7
	…		…		…		…	…
	Rx3IN-4	Rx3IN-3	Rx3QN-4	Rx3QN-3	Rx3IN-2	Rx3IN-1	Rx3QN-2	Rx3QN-1

24.9 *DCA1000 EVM capture format (xWR16xx real, 4 channel, 2 lanes [Non-Interleaved])*

Chirp 1	Rx0I0	Rx0I1	Rx0I2	Rx0I3	Rx0I4	Rx0I5	Rx0I6	Rx0I7
	Rx0I8	Rx0I9	Rx0I10	Rx0I11	Rx0I12	Rx0I13	Rx0I14	Rx0I15
	…		…		…		…	…
	Rx3IN-8	Rx3IN-7	Rx3IN-6	Rx3IN-5	Rx3IN-4	Rx3IN-3	Rx3IN-2	Rx3IN-1
Chirp 2	Rx0I0	Rx0I1	Rx0I2	Rx0I3	Rx0I4	Rx0I5	Rx0I6	Rx0I7
	Rx*0I8*	Rx*0I9*	Rx*0I10*	Rx*0I11*	Rx0I*12*	Rx*0I13*	Rx*0I14*	Rx*0I15*
	…		…		…		…	…
	Rx3I*N-8*	Rx3I*N-7*	Rx3I*N-6*	Rx3I*N-5*	Rx3I*N-4*	Rx3I*N-3*	Rx3I*N-2*	Rx3I*N-1*

IMPORTANT NOTICE

Texas Instruments Incorporated ("TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your noncompliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265