User's Guide for mmWaveStudio 4.3.1

mmWave Studio GUI User Guide

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.

Table of Contents

List of Figures **2**

1 Introduction **4**

2 mmWaveStudio Installation and Startup **4**

2.1 Installation 4
2.2 Startup 5
2.3 USB Interface and Drivers 6

3 Using the DCA1000 EVM **9**

3.1 Using DCA1000 CLI utility 9
3.2 Updating the DCA1000 FPGA Binary 9
3.3 Raw ADC data capture 9

4 mmWaveStudio User Interface **10**

4.1 Menu Bar 10
4.1.1 File Menu 10
4.1.2 View Menu 10
4.1.3 Tools Menu 11
4.1.4 ToolBars Menu 11
4.1.5 Window Menu 12
4.1.6 Help Menu 13
4.2 Radar API Window 13
4.2.1 Connection 13
4.2.2 Device Control & DeviceControl 1 14
4.2.3 Sensor Config & Sensor Config 1 15
4.2.4 Monitor Config 16
4.2.5 RFEval 17
4.2.6 JSON Import Export 17
4.2.7 Misc 17
4.3 Output Window 17
4.4 LUA Shell 18

5 Automation/Scripting **19**

6 Radar API Tab Operations **20**

6.1 Connection Tab 20
6.1.1 Board Control Operations 21
6.1.2 RS232 Operations 22

6.1.3 Firmware Download 23
6.1.4 SPI Connection & FEC Power Up 24
6.2 Device Control Tab 25
6.3 Sensor Config Tab 26
6.4 Monitor Config Tab 27
6.5 RFEval 28
6.6 JSON Import Export 28
6.7 Misc 28

7 APIs Programming sequence 30

8 Radar post processing 30

8.1 List of plots 31
8.1.1 Basic plots 31
8.1.2 CQ plots (chirp quality metrics) 34
8.1.3 Characterization plots 34
8.2 Channel select 36
8.3 Frame /Profile /Chirp sliders 36
8.4 Information sidebar 36
8.4.1 Programmed parameters 37
8.4.2 Calculated parameters 37
8.4.3 Detection results 38
8.5 Play 38
8.6 Miscellaneous options 38
8.6.1 Detection 38
8.6.2 FFT processing settings 39
8.6.3 Miscellaneous 40
8.6.4 Time domain options 40
8.6.5 More miscellaneous options 40
8.7 Notes on Setting up PostProc for Characterization 41
8.8 Standalone processing (post processing done in a different session from configuration and capture) 41

9 Controlling mmWaveStudio from Matlab 43

10 Automation using LUA 45

10.1 Sample Lua script for automation 45

List of Figures

Figure 2.1. mmWaveStudio startup error message 6
Figure 2.2. Device Manager (Other Devices) 7
Figure 2.3 FTDI driver update 7
Figure 2.4. Device Manager (USB Serial Port) 8
Figure 2.5. DCA1000 COM Ports 8
Figure 2.6. xWRL68xx EVM USB Enumeration 9
Figure 4.1. mmWaveStudio Main Window 10
Figure 4.2. Config ToolBar 11
Figure 4.3. Edit Button 11
Figure 4.4. Button Config 12
Figure 4.5. Edit User Defined Script 12
Figure 4.6. Radar API Window 13

Figure 4.7. Output Window Docking 18
Figure 4.8. Output Window Log 18
Figure 4.9. LUA Shell window 19
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 23
Figure 6.4. Firmware Download 23
Figure 6.5. Firmware Download Progress 24
Figure 8.1. PostProc Basic 30
Figure 8.2. 2D FFT Profile 32
Figure 8.3. Range Angle plot 32
Figure 8.4. Detection and Angle Estimation results 33
Figure 8.5. Chirp Config Picture 33
Figure 8.6. 1D FFT Profile 34
Figure 8.7. Time domain plot 34
Figure 8.8. Phase Stability 35
Figure 8.9. Amplitude stability 35
Figure 8.10. Zero-velocity bin vs High velocity bin. 36
Figure 8.11. Channel select 36
Figure 8.12. Sliders 36
Figure 8.13. Information sidebar 37
Figure 8.14. Play 38
Figure 8.15. Miscellaneous options 38

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 mmWave DFP API commands to the device over SPI. ADC data is captured using DCA1000 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.

This mmWave Studio is designed to support TI third generation low power and low cost radar devices like xWRLx432 and xWRL68xx. The mmWave DFP APIs for these devices are brand new and this version of Studio supports only these 3rd generation device APIs.

Key features of the mmWaveStudio GUI are

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

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

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 DCA1000 EVM and the xWRL68xx BOOST-EVM.

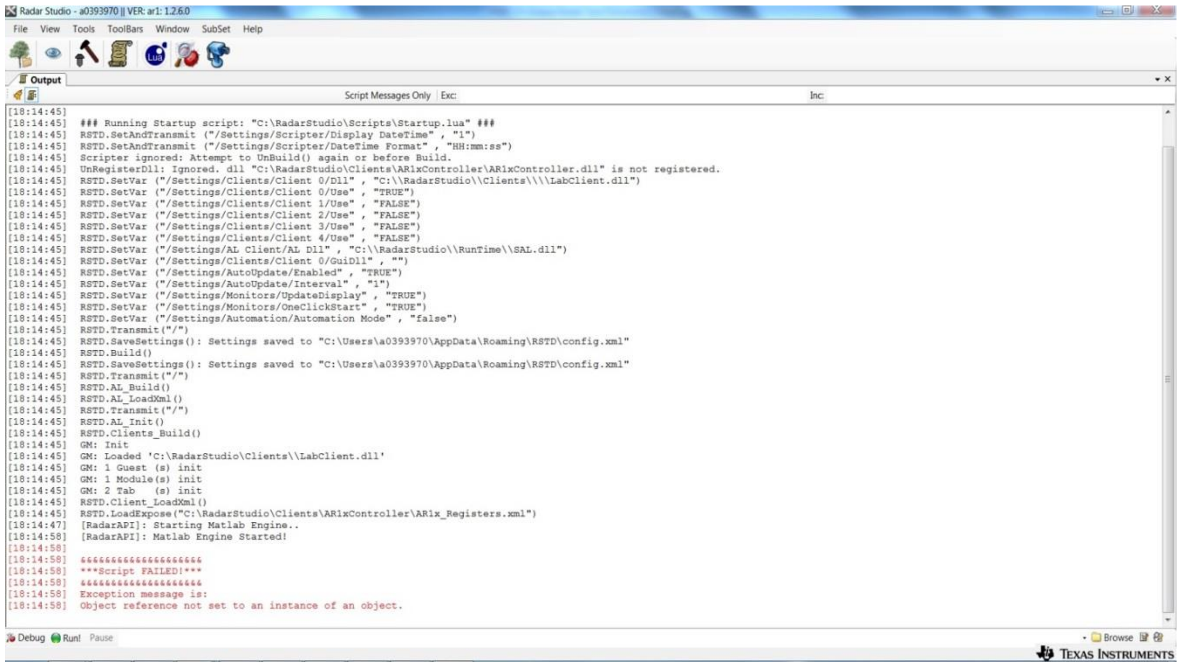
> **NOTE:** Make sure the above combination of DCA1000 and AWR BOOST-EVM are connected to the PC while opening mmWaveStudio for the first time.

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


Radar Studio - a0393970 || VER: ar1: 1.2.6.0
File View Tools ToolBars Window SubSet Help
Output
Script Messages Only Exc:
Inc:
Debug Run! Pause
Browse

```
[18:14:45]
[18:14:45]  ### Running Startup script: "C:\RadarStudio\Scripts\Startup.lua" ###
[18:14:45]  RSTD.SetAndTransmit ("/Settings/Scripter/Display DateTime" , "1")
[18:14:45]  RSTD.SetAndTransmit ("/Settings/Scripter/DateTime Format" , "HH:mm:ss")
[18:14:45]  Scripter ignored: Attempt to UnBuild() again or before Build.
[18:14:45]  UnRegisterDll: Ignored. dll "C:\RadarStudio\Clients\AR1xController\AR1xController.dll" is not registered.
[18:14:45]  RSTD.SetVar ("/Settings/Clients/Client 0/Dll" , "C:\\RadarStudio\\Clients\\\\LabClient.dll")
[18:14:45]  RSTD.SetVar ("/Settings/Clients/Client 0/Use" , "TRUE")
[18:14:45]  RSTD.SetVar ("/Settings/Clients/Client 1/Use" , "FALSE")
[18:14:45]  RSTD.SetVar ("/Settings/Clients/Client 2/Use" , "FALSE")
[18:14:45]  RSTD.SetVar ("/Settings/Clients/Client 3/Use" , "FALSE")
[18:14:45]  RSTD.SetVar ("/Settings/Clients/Client 4/Use" , "FALSE")
[18:14:45]  RSTD.SetVar ("/Settings/AL Client/AL Dll" , "C:\\RadarStudio\\RunTime\\SAL.dll")
[18:14:45]  RSTD.SetVar ("/Settings/Clients/Client 0/GuiDll" , "")
[18:14:45]  RSTD.SetVar ("/Settings/AutoUpdate/Enabled" , "TRUE")
[18:14:45]  RSTD.SetVar ("/Settings/AutoUpdate/Interval" , "1")
[18:14:45]  RSTD.SetVar ("/Settings/Monitors/UpdateDisplay" , "TRUE")
[18:14:45]  RSTD.SetVar ("/Settings/Monitors/OneClickStart" , "TRUE")
[18:14:45]  RSTD.SetVar ("/Settings/Automation/Automation Mode" , "false")
[18:14:45]  RSTD.Transmit("/")
[18:14:45]  RSTD.SaveSettings(): Settings saved to "C:\Users\a0393970\AppData\Roaming\RSTD\config.xml"
[18:14:45]  RSTD.Build()
[18:14:45]  RSTD.SaveSettings(): Settings saved to "C:\Users\a0393970\AppData\Roaming\RSTD\config.xml"
[18:14:45]  RSTD.Transmit("/")
[18:14:45]  RSTD.AL_Build()
[18:14:45]  RSTD.AL_LoadXml()
[18:14:45]  RSTD.Transmit("/")
[18:14:45]  RSTD.AL_Init()
[18:14:45]  RSTD.Clients_Build()
[18:14:45]  GM: Init
[18:14:45]  GM: Loaded 'C:\RadarStudio\Clients\\LabClient.dll'
[18:14:45]  GM: 1 Guest (s) init
[18:14:45]  GM: 1 Module(s) init
[18:14:45]  GM: 2 Tab   (s) init
[18:14:45]  RSTD.Client_LoadXml()
[18:14:45]  RSTD.LoadExpose("C:\RadarStudio\Clients\AR1xController\AR1x_Registers.xml")
[18:14:47]  [RadarAPI]: Starting Matlab Engine..
[18:14:58]  [RadarAPI]: Matlab Engine Started!
[18:14:58]
[18:14:58]  &&&&&&&&&&&&&&&&&&&&
[18:14:58]  ***Script FAILED!***
[18:14:58]  &&&&&&&&&&&&&&&&&&&&
[18:14:58]  Exception message is:
[18:14:58]  Object reference not set to an instance of an object.
```

Figure 2.1. mmWaveStudio startup error message

a. Install the FTDI drivers again while DCA1000 board is connected through USB

b. Re-launch the mmWaveStudio GUI

2.3 USB Interface and Drivers

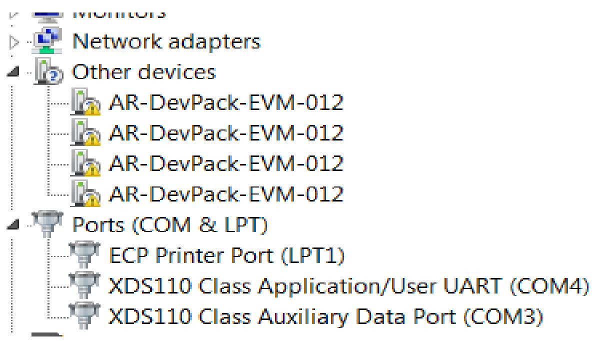
Once the USB is connected to the xWRL68xx DCA1000 EVM, ensure to install the FTDI USB Drivers (mmwave_studio_<ver>\ftdi). After completion of the driver installation, COM ports will be available in Windows Device Manager as shown in FIGURE 2.6.

In case ftdi device is not coming under COM Ports in device manager follow the steps below :

a. If the ftdi devices are coming under USB Devices:
    i. Right click on the device and do Uninstall device, then check the delete associated driver option and proceed. this should delete any ftdi related drivers (need to do this for each ports A,B,C,D)
    ii. Now go to Studio/ftdi/ and launch dpinst64.exe to install the ftdi drivers.
    iii. Disconnect and connect the device again, the ftdi ports should appear under COM Ports in device manager
b. If ftdi devices are coming under other devices it means there is no ftdi driver installed:
    i. Go to Studio/ftdi/ and launch dpinst64.exe to install the ftdi drivers.
    ii. Disconnect and connect the device again, the ftdi ports should appear under COM Ports in device manager

Once xWRL68xx EVM is connected to the DCA1000 EVM, connect 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

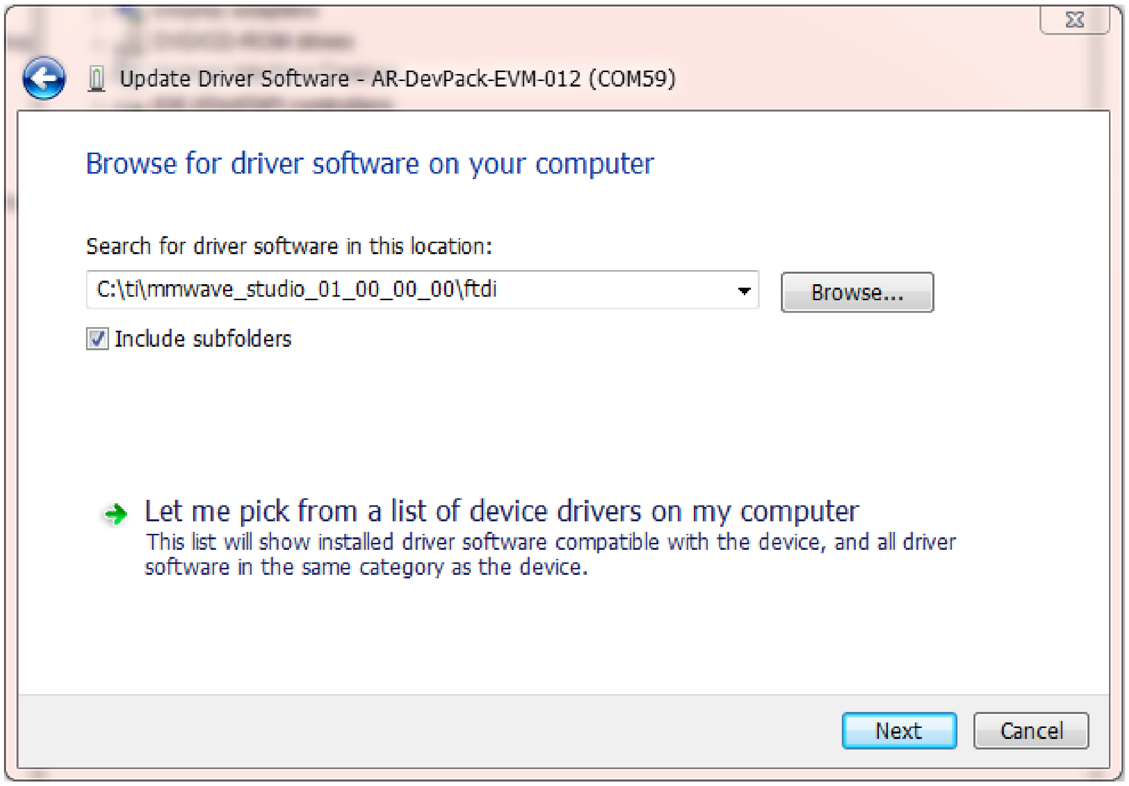
When the DCA1000 EVM is connected for the first time to the PC, Windows maybe not be able to recognize the device and would come up as 'Other devices' in device manager as shown in Figure 2.2



Network adapters
Other devices
AR-DevPack-EVM-012
AR-DevPack-EVM-012
AR-DevPack-EVM-012
AR-DevPack-EVM-012
Ports (COM & LPT)
ECP Printer Port (LPT1)
XDS110 Class Application/User UART (COM4)
XDS110 Class Auxiliary Data Port (COM3)

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



Update Driver Software - AR-DevPack-EVM-012 (COM59)

Browse for driver software on your computer

Search for driver software in this location:

C:\ti\mmwave_studio_01_00_00_00\ftdi

Browse...

☑ Include subfolders

Let me pick from a list of device drivers on my computer

This list will show installed driver software compatible with the device, and all driver software in the same category as the device.

Next Cancel

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.

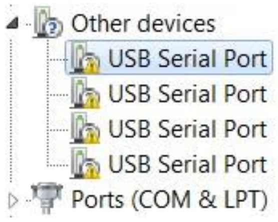


Figure 2.4. Device Manager (USB Serial Port)

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

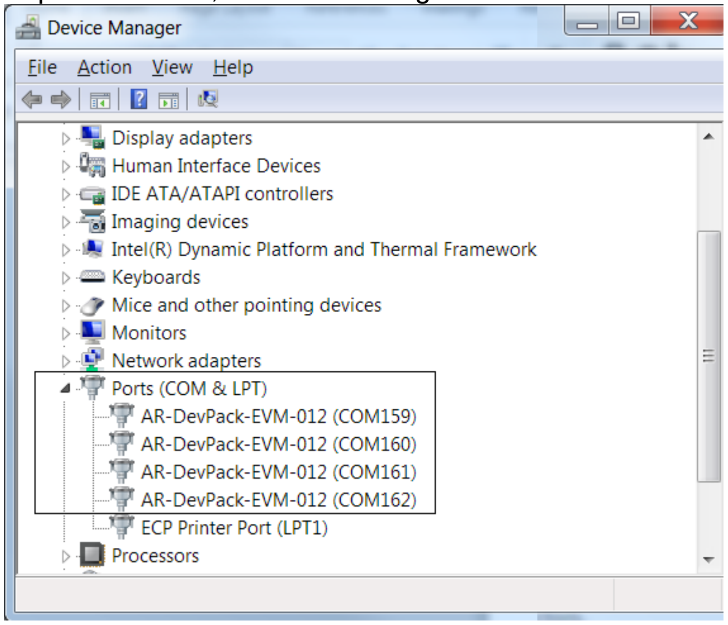

Device Manager
File Action View Help
Display adapters
Human Interface Devices
IDE ATA/ATAPI controllers
Imaging devices
Intel(R) Dynamic Platform and Thermal Framework
Keyboards
Mice and other pointing devices
Monitors
Network adapters
Ports (COM & LPT)
AR-DevPack-EVM-012 (COM159)
AR-DevPack-EVM-012 (COM160)
AR-DevPack-EVM-012 (COM161)
AR-DevPack-EVM-012 (COM162)
ECP Printer Port (LPT1)
Processors

Figure 2.5. DCA1000 COM Ports

Next connect the USB cable from the xWRL68xx 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 xWRL68xx 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]

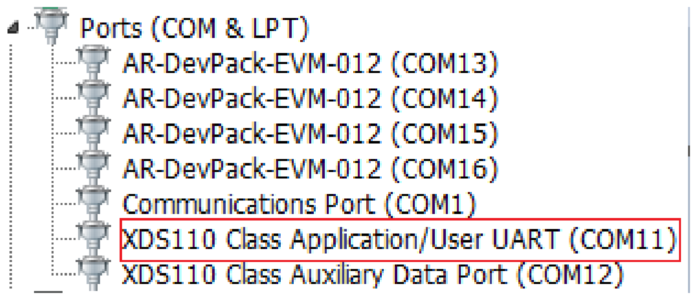

Ports (COM & LPT)
AR-DevPack-EVM-012 (COM13)
AR-DevPack-EVM-012 (COM14)
AR-DevPack-EVM-012 (COM15)
AR-DevPack-EVM-012 (COM16)
Communications Port (COM1)
XDS110 Class Application/User UART (COM11)
XDS110 Class Auxiliary Data Port (COM12)

Figure 2.6. xWRL68xx EVM USB Enumeration

3 Using the DCA1000 EVM

The DCA1000 EVM receives the LVDS data from xWRL68xx 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 2.9) 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.

The change is in the .dll file. No FPGA update is needed. Both adc_data_Raw_0.bin and adc_data.bin are reordered already and ready 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 and Lua Shell

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

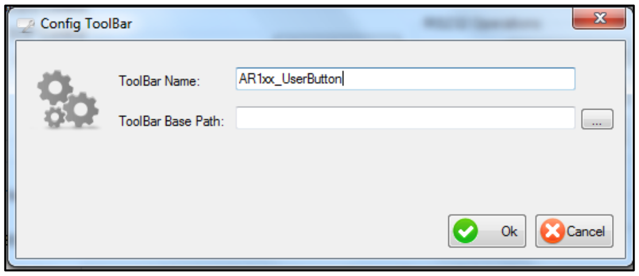
Lock layout: Locks and unlocks the current layout. Once the layout is locked, the windows cannot be moved around.

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



AR1xx_UserButton

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.

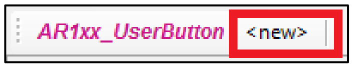


Figure 4.3. Edit Button

4. This opens the User Button Configuration window

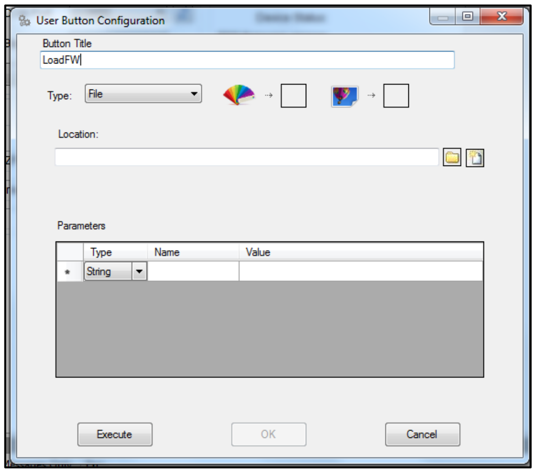


Figure 4.4. Button Config

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.

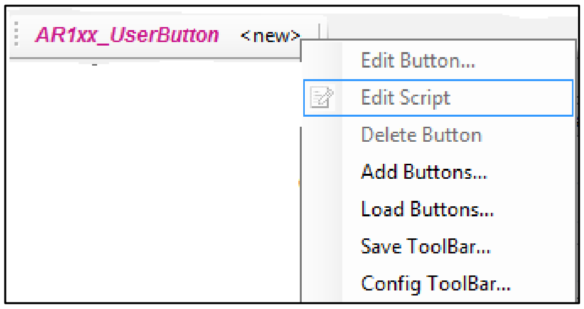


Figure 4.5. Edit User Defined Script

- **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

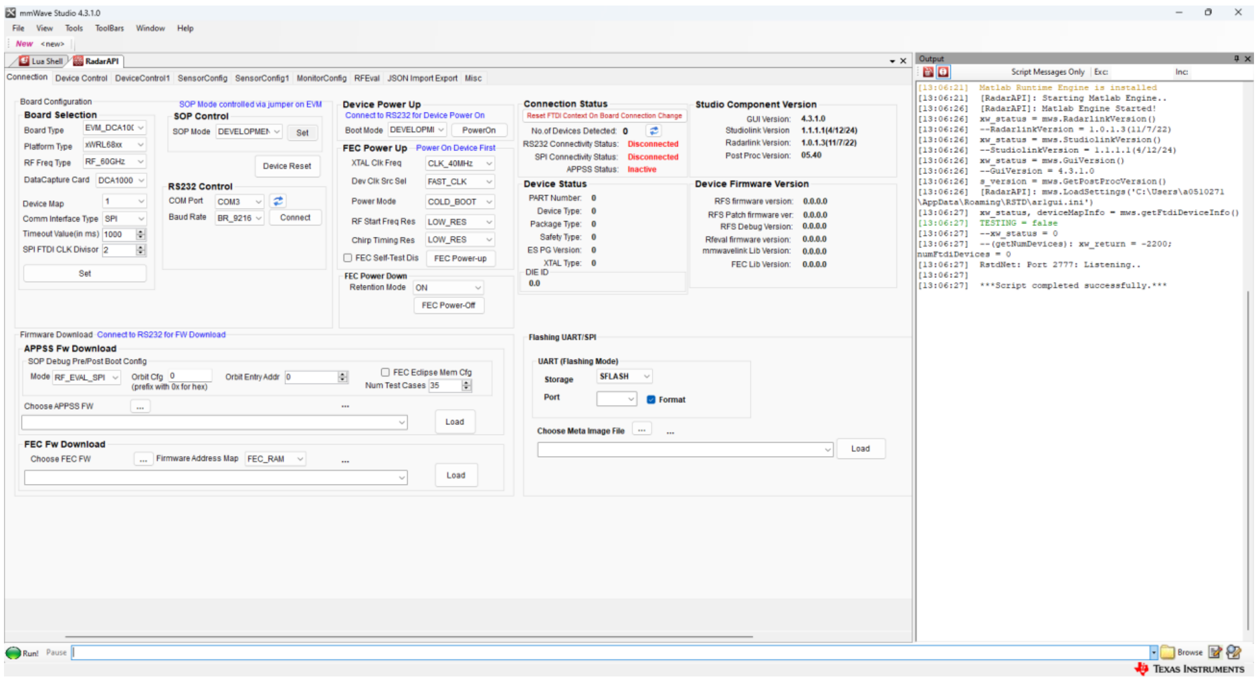
The Window menu shows open windows in the application.

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 xWRL68xx device over SPI by interfacing through C DLL. Once the ADC data is captured using DCA1000 board, the post processing is done and plots are available in the GUI by interfacing with Matlab DLL.



Board Configuration — SOP Mode controlled via jumper on EVM

Board Selection
Board Type: EVM_DCA10(
Platform Type: xWRL68xx
RF Freq Type: RF_60GHz
DataCapture Card: DCA1000
Device Map: 1
Comm Interface Type: SPI
Timeout Value(in ms): 1000
SPI FTDI CLK Divisor: 2

SOP Control
SOP Mode: DEVELOPMEN

RS232 Control
COM Port: COM3
Baud Rate: BR_9216

Device Power Up — Connect to RS232 for Device Power On
Boot Mode: DEVELOPMI

FEC Power Up — Power On Device First
XTAL Clk Freq: CLK_40MHz
Dev Clk Src Sel: FAST_CLK
Power Mode: COLD_BOOT
RF Start Freq Res: LOW_RES
Chirp Timing Res: LOW_RES
☐ FEC Self-Test Dis

FEC Power Down
Retention Mode: ON

Connection Status
Reset FTDI Context On Board Connection Change
No.of Devices Detected: 0
RS232 Connectivity Status: Disconnected
SPI Connectivity Status: Disconnected
APPSS Status: Inactive

Studio Component Version
GUI Version: 4.3.1.0
Studiolink Version: 1.1.1.1(4/12/24)
Radarlink Version: 1.0.1.3(11/7/22)
Post Proc Version: 05.40

Device Status
PART Number: 0
Device Type: 0
Package Type: 0
Safety Type: 0
ES PG Version: 0
XTAL Type: 0
DIE ID: 0.0

Device Firmware Version
RFS firmware version: 0.0.0.0
RFS Patch firmware ver: 0.0.0.0
RFS Debug Version: 0.0.0.0
Rfeval firmware version: 0.0.0.0
mmwavelink Lib Version: 0.0.0.0
FEC Lib Version: 0.0.0.0

Firmware Download — Connect to RS232 for FW Download

APPSS Fw Download
SOP Debug Pre/Post Boot Config
Mode: RF_EVAL_SPI
Orbit Cfg: 0 (prefix with 0x for hex)
Orbit Entry Addr: 0
☐ FEC Eclipse Mem Cfg
Num Test Cases: 35
Choose APPSS FW ...

FEC Fw Download
Choose FEC FW ...
Firmware Address Map: FEC_RAM

Flashing UART/SPI
UART (Flashing Mode)
Storage: SFLASH
Port:
☑ Format
Choose Meta Image File ...

```
[13:06:21]  Matlab Runtime Engine is installed
[13:06:21]  [RadarAPI]: Starting Matlab Engine..
[13:06:26]  [RadarAPI]: Matlab Engine Started!
[13:06:26]  xw_status = mws.RadarlinkVersion()
[13:06:26]  --RadarlinkVersion = 1.0.1.3(11/7/22)
[13:06:26]  xw_status = mws.StudiolinkVersion()
[13:06:26]  --StudiolinkVersion = 1.1.1.1(4/12/24)
[13:06:26]  xw_status = mws.GuiVersion()
[13:06:26]  --GuiVersion = 4.3.1.0
[13:06:26]  s_version = mws.GetPostProcVersion()
[13:06:26]  [RadarAPI]: mws.LoadSettings('C:\Users\a0510271
\AppData\Roaming\RSTD\arlgui.ini')
[13:06:27]  xw_status, deviceMapInfo = mws.getFtdiDeviceInfo()
[13:06:27]  TESTING = false
[13:06:27]  --xw_status = 0
[13:06:27]  --(getNumDevices): xw_return = -2200;
numFtdiDevices = 0
[13:06:27]  RstdNet: Port 2777: Listening..
[13:06:27]
[13:06:27]  ***Script completed successfully.***
```

Figure 4.6. Radar API Window

In the Radar API window, the following tabs are available for communication with the Radar Device. For more information on these APIs, refer mmwave_studio<version>\docs\mmwave_studio_lua_api_documentation.pdf

4.2.1 *Connection*

- mws.GuiVersion
- mws.StudiolinkVersion
- mws.RadarlinkVersion
- mws.selectBDType
- mws.selectComIfType

- mws.sopControl
- mws.nReset
- mws.GpioControl
- mws.rs232Config
- mws.sopDebugFwPreBootCfg
- mws.fwBuildImageDownload
- mws.sopDebugFwPostBootCfg
- mws.devicePowerup
- mws.fecDevicePowerOn
- mws.fecDevicePowerOff
- mws.fwMetaImageDownloadUart
- mws.fwMetaImageDownloadSpi
- mws.devicePowerup

4.2.2 Device Control & DeviceControl 1

- mws.fecDfpVerGet
- mws.fecRfPwrOnOff
- mws.fecGpadcMeasCfg
- mws.fecGpadcMeasTrig
- mws.fecDevClkctrl
- mws.fecRdifctrl
- mws.fecTempcMeasCfg
- mws.fecTempMeasTrig
- mws.fecRfsDbgCtrl
- mws.fecRfBootCal
- mws.fecRfRuntimeCal
- mws.fecRfCalDataGet
- mws.fecRfCalDataSet
- mws.fecRfClkBwCfg
- mws.fecDevStatusGet
- mws.fecRfStatusGet
- mws.fecRfFaultStatusGet

- mws.fecDieIdGet
- mws.fecRfRuntimeTxClpcCal
- mws.fecRxTxCalDataGet
- mws.fecRxTxCalDataSet

4.2.3 Sensor Config & Sensor Config 1

- mws.sensChirpPfComnCfg
- mws.sensChirpPfTimeCfg
- mws.sensFrameCfg
- mws.sensorStart
- mws.sensorStop
- mws.sensLoopBackCfg
- mws.sensLoopBackEna
- mws.monDbgPdMeas
- mws.monDbgTxPwrMeas
- mws.monLivSynthFreqCfg
- mws.monLivRxSatCfg
- mws.StartMatlabPostProc
- mws.sensChirpPfComnCfgGet
- mws.sensChirpPfTimeCfgGet
- mws.sensPerChirpLut
- mws.sensPerChirpLutGet
- mws.sensPerChirpCfg
- mws.sensPerChirpCfgGet
- mws.sensPerChirpCtrl
- mws.sensPerChirpCtrlGet
- mws.sensFrameCfgGet
- mws.sensorStsGet
- mws.sensDynPwrSaveDis
- mws.sensDynPwrSaveStsGet
- mws.rfevalDevRxDataStreamCfgSet
- mws.rfevalDevRxDataStreamAdvCfgSet

- mws.rfevalDevRxDataStreamCtrl
- mws.captureCardEthernetCfg
- mws.captureCardModeCfg
- mws.getCaptureCardFPGAVersion
- mws.captureCardCfgResetFPGA
- mws.captureCardStartRecord
- mws.captureCardStopRecord
- mws.captureCardDisconnect

4.2.4 Monitor Config

- mws.monEnableTrig
- mws.monDbgPdMeas
- mws.monDbgTxPwrMeas
- mws.monLivSynthFreqCfg
- mws.monLivRxSatCfg
- mws.monLivRxSatRspGet
- mws.monPllCtrlVltCfg
- mws.monTxNRxLbCfg
- mws.monTxNPwrCfg
- mws.monTxNBbCfg
- mws.monTxNDcSigCfg
- mws.monRxHpfDcSigCfg
- mws.monPmClkDcSigCfg
- mws.monLivSynthFreqRspGet
- mws.monPllCtrlVltRspGet
- mws.monTxNRxLbRspGet
- mws.monTxNPwrRspGet
- mws.monTxNBbRspGet
- mws.monTxNDcSigRspGet
- mws.monRxHpfDcSigRspGet
- mws.monPmClkDcSigRspGet

4.2.5 RFEval

- mws.rfEvalMemRead
- mws.rfEvalMemWrite
- mws.rfEvalRegRead
- mws.rfEvalRegWrite
- mws.rfEvalRfsCmd
- mws.rfEvalDevAppStsGet
- mws.rfEvalDevAppCfg
- mws.rfEvalAteInit
- mws.readRegisterField
- mws.writeRegisterField
- mws.readRegisterByName
- mws.writeRegisterByName
- mws.readRegister
- mws.writeRegister
- mws.memBlockRead
- mws.memBlockWrite

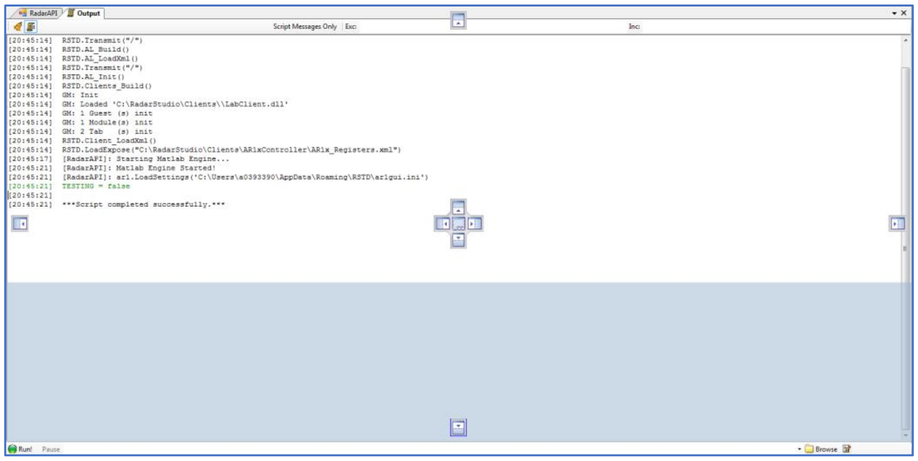
4.2.6 JSON Import Export

4.2.7 Misc

- mws.sensConvFreqToCode
- mws.sensConvTimeToCode
- mws.sensConvSlopeToCode
- mws.sensConvFtPeriodToCode
- mws.DisableConseLog

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.



```
[20:45:14]  RSTD.Transmit("/")
[20:45:14]  RSTD.AL_Build()
[20:45:14]  RSTD.AL_LoadXml()
[20:45:14]  RSTD.Transmit("/")
[20:45:14]  RSTD.AL_Init()
[20:45:14]  RSTD.Clients_Build()
[20:45:14]  GM: Init
[20:45:14]  GM: Loaded 'C:\RadarStudio\Clients\\LabClient.dll'
[20:45:14]  GM: 1 Guest (s) init
[20:45:14]  GM: 1 Module(s) init
[20:45:14]  GM: 2 Tab   (s) init
[20:45:14]  RSTD.Client_LoadXml()
[20:45:14]  RSTD.LoadExpose("C:\RadarStudio\Clients\AR1xController\AR1x_Registers.xml")
[20:45:17]  [RadarAPI]: Starting Matlab Engine...
[20:45:21]  [RadarAPI]: Matlab Engine Started!
[20:45:21]  [RadarAPI]: ar1.LoadSettings('C:\Users\a0393390\AppData\Roaming\RSTD\ar1gui.ini')
[20:45:21]  TESTING = false
[20:45:21]
[20:45:21]  ***Script completed successfully.***
```

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.

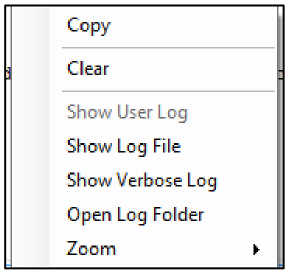
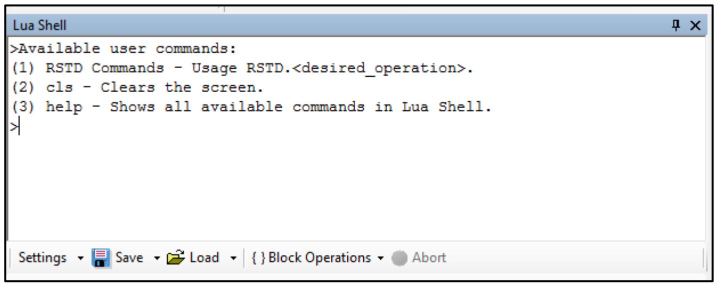


Figure 4.8. Output Window Log

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:



```
>Available user commands:
(1) RSTD Commands - Usage RSTD.<desired_operation>.
(2) cls - Clears the screen.
(3) help - Shows all available commands in Lua Shell.
>
```

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”)

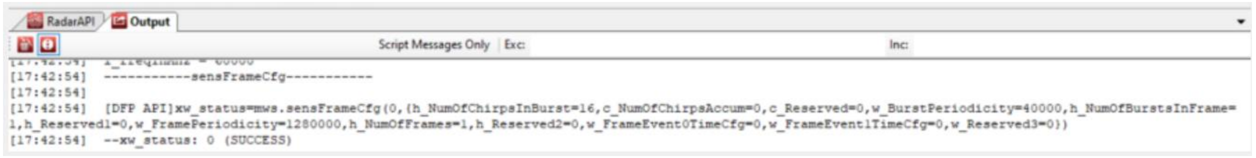
> **NOTE:** The LUA shell supports auto-completion with the tab key.

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 mmWave Studio exposes LUA API functions to interface with device, which can be used to test predefined sequences and automate functionality on the device. All these functions can be found under the mws module.

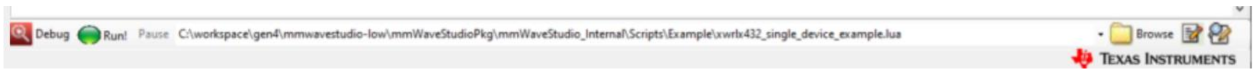
1. Whenever a command is issued in the GUI window, the equivalent script commands can be taken from the Output window and formed an automation script. For example, In the Static Config tab, when Channel and ADC Config command is sent, in the Output window, the script command is logged.



```
[17:42:54]  ------------sensFrameCfg------------
[17:42:54]
[17:42:54]  [DFP API]xw_status=mws.sensFrameCfg(0,{h_NumOfChirpsInBurst=16,c_NumOfChirpsAccum=0,c_Reserved=0,w_BurstPeriodicity=40000,h_NumOfBurstsInFrame=
1,h_Reserved1=0,w_FramePeriodicity=1280000,h_NumOfFrames=1,h_Reserved2=0,w_FrameEvent0TimeCfg=0,w_FrameEvent1TimeCfg=0,w_Reserved3=0})
[17:42:54]  --xw_status: 0 (SUCCESS)
```

Figure 5.1. Command creation

2. The script commands start with ‘mws’. 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



C:\workspace\gen4\mmwavestudio-low\mmWaveStudioPkg\mmWaveStudio_Internal\Scripts\Example\xwrlx432_single_device_example.lua

Figure 5.2. LUA Script execution

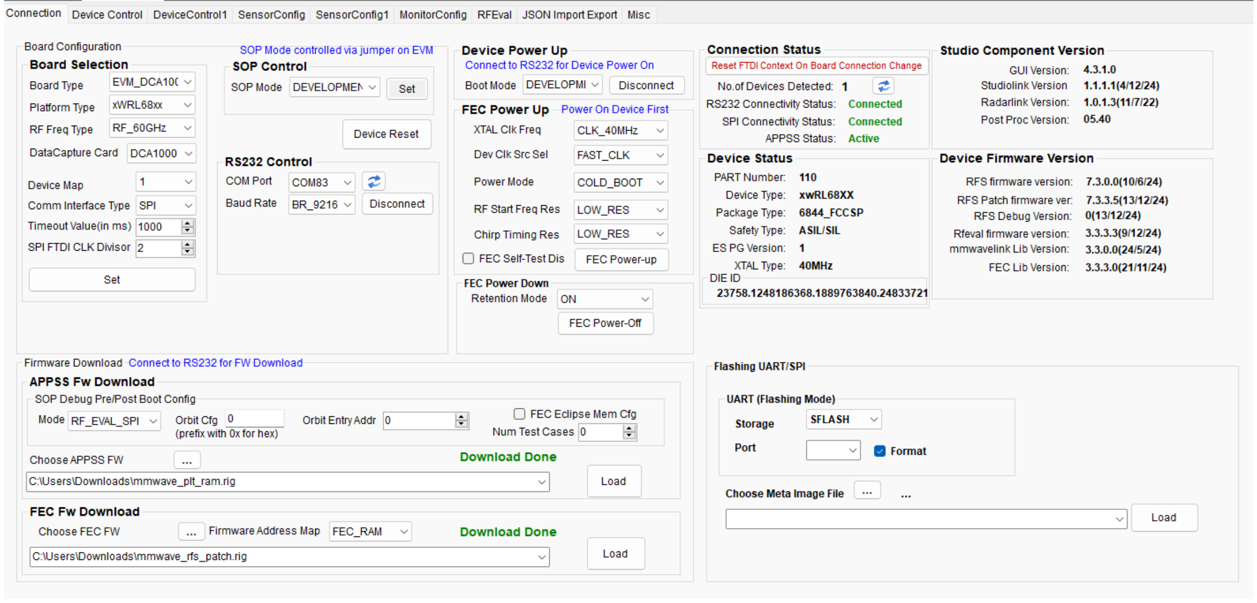
For more information on LUA Scripting language, please refer to: http://lua-users.org/wiki/ - The Lua wiki (See the Lua Directory inside for Lua tutorials) http://www.lua.org/manual/5.1/manual.html - Lua official Reference Manual

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.



Board Configuration
SOP Mode controlled via jumper on EVM

Board Selection
Board Type: EVM_DCA10(
Platform Type: xWRL68xx
RF Freq Type: RF_60GHz
DataCapture Card: DCA1000
Device Map: 1
Comm Interface Type: SPI
Timeout Value(in ms): 1000
SPI FTDI CLK Divisor: 2

SOP Control
SOP Mode: DEVELOPMEN

RS232 Control
COM Port: COM83
Baud Rate: BR_9216

Device Power Up
Connect to RS232 for Device Power On
Boot Mode: DEVELOPMI

FEC Power Up — Power On Device First
XTAL Clk Freq: CLK_40MHz
Dev Clk Src Sel: FAST_CLK
Power Mode: COLD_BOOT
RF Start Freq Res: LOW_RES
Chirp Timing Res: LOW_RES
☐ FEC Self-Test Dis

FEC Power Down
Retention Mode: ON

Connection Status
No.of Devices Detected: 1
RS232 Connectivity Status: Connected
SPI Connectivity Status: Connected
APPSS Status: Active

Studio Component Version
GUI Version: 4.3.1.0
Studiolink Version 1.1.1.1(4/12/24)
Radarlink Version: 1.0.1.3(11/7/22)
Post Proc Version: 05.40

Device Status
PART Number: 110
Device Type: xwRL68XX
Package Type: 6844_FCCSP
Safety Type: ASIL/SIL
ES PG Version: 1
XTAL Type: 40MHz
DIE ID
23758.1248186368.1889763840.24833721

Device Firmware Version
RFS firmware version: 7.3.0.0(10/6/24)
RFS Patch firmware ver: 7.3.3.5(13/12/24)
RFS Debug Version: 0(13/12/24)
Rfeval firmware version: 3.3.3.3(9/12/24)
mmwavelink Lib Version: 3.3.0.0(24/5/24)
FEC Lib Version: 3.3.3.0(21/11/24)

Firmware Download Connect to RS232 for FW Download

APPSS Fw Download
SOP Debug Pre/Post Boot Config
Mode: RF_EVAL_SPI
Orbit Cfg: 0 (prefix with 0x for hex)
Orbit Entry Addr: 0
☐ FEC Eclipse Mem Cfg
Num Test Cases: 0
Choose APPSS FW
Download Done
C:\Users\Downloads\mmwave_plt_ram.rig

FEC Fw Download
Choose FEC FW
Firmware Address Map: FEC_RAM
Download Done
C:\Users\Downloads\mmwave_rfs_patch.rig

Flashing UART/SPI
UART (Flashing Mode)
Storage: SFLASH
Port:
☑ Format
Choose Meta Image File
...

Figure 6.1. Connect Tab

Using this tab, user can perform

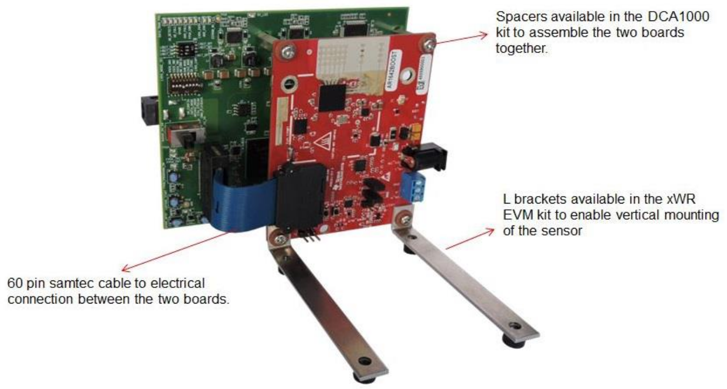
- Board control operations
- RS232 Operations
- Connectivity Status and content versions
- Firmware Download (RS232/SPI/UART)
- SPI Connection
- FEC Power On/Off

CAUTION:

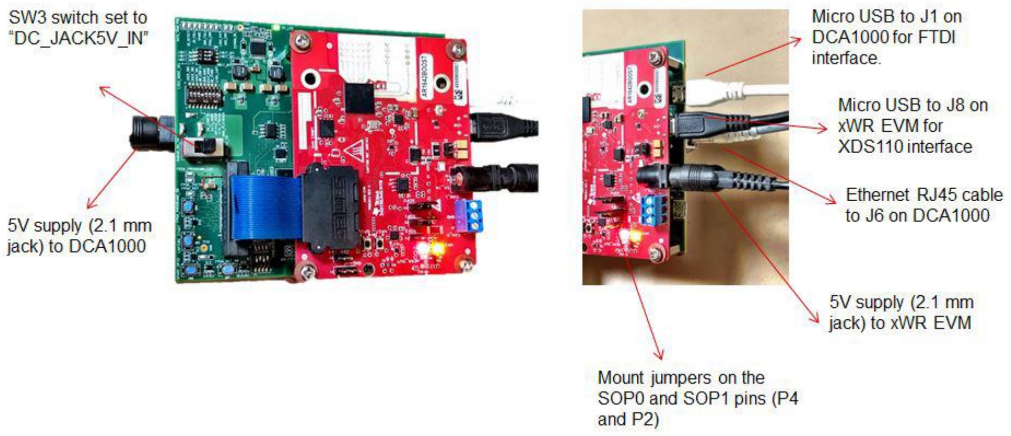
Before starting with any action on mmWaveStudio please make sure that GUI detects the device connected to PC. Value of 'No of devices detected' should not be 0 else click on to refresh. In case number of devices detected is 0 go through USB Interface and Drivers section to make sure correct drivers are installed and FTDI ports are coming under COM Ports in device manager.

6.1.1 Board Control Operations

mmWaveStudio does board control operation of L68xx device through FTDI ports on DCA100.


Spacers available in the DCA1000 kit to assemble the two boards together.
L brackets available in the xWR EVM kit to enable vertical mounting of the sensor
60 pin samtec cable to electrical connection between the two boards.

Hardware Connection: DCA1000 With xWR mmWave Sensor EVMs


SW3 switch set to "DC_JACK5V_IN"
5V supply (2.1 mm jack) to DCA1000
Micro USB to J1 on DCA1000 for FTDI interface.
Micro USB to J8 on xWR EVM for XDS110 interface
Ethernet RJ45 cable to J6 on DCA1000
5V supply (2.1 mm jack) to xWR EVM
Mount jumpers on the SOP0 and SOP1 pins (P4 and P2)

Data and Supply Cable Connections

(Refer to mmwave_studio<version>\docs\DCA1000_Quick_Start_Guide.pdf for more detail)

Board Selection

Configure the Board and Communication interface.

Refer: mws.selectBDType, mws.selectComIfType

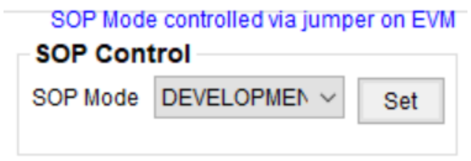
SOP Control

NOTE: When using a DCA1000 EVM, use the jumpers on the L68xx BOOST to control the SOP settings.

The L68xx 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:

- SOP Development Mode. This mode should be used for RF evaluation.
  - Set SOP Development Mode. For characterization and evaluation of xWRL68xx devices, always use SOP Development mode
  - Connect over RS232
  - Perform APPSS & FEC Firmware Download
  - Perform SPI Connect
  - Issue API to the device

Once the USB connections to the board is intact, Select the SOP Mode(In case of evm use jumpers to set sop mode on board) and Click Set button. For SOP setting to be effective, reset the device.


SOP Mode controlled via jumper on EVM

SOP Control

SOP Mode: DEVELOPMEN

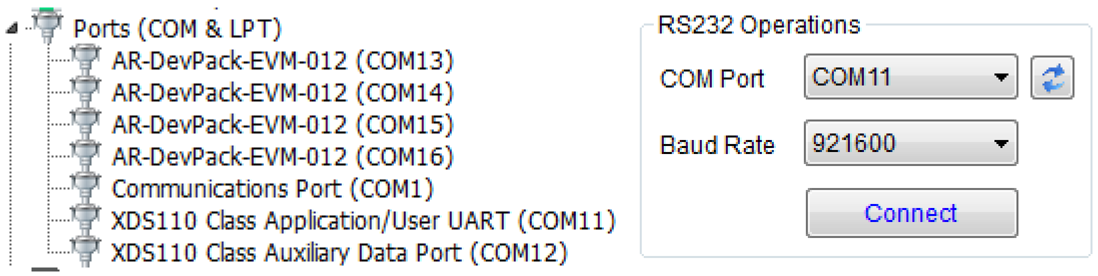
Figure 6.2. SOP Control

6.1.2 RS232 Operations

The serial port mode provides a direct PC connection of xWRL68xx 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.



Ports (COM & LPT)
- AR-DevPack-EVM-012 (COM13)
- AR-DevPack-EVM-012 (COM14)
- AR-DevPack-EVM-012 (COM15)
- AR-DevPack-EVM-012 (COM16)
- Communications Port (COM1)
- XDS110 Class Application/User UART (COM11)
- XDS110 Class Auxiliary Data Port (COM12)

RS232 Operations

COM Port: COM11

Baud Rate: 921600

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

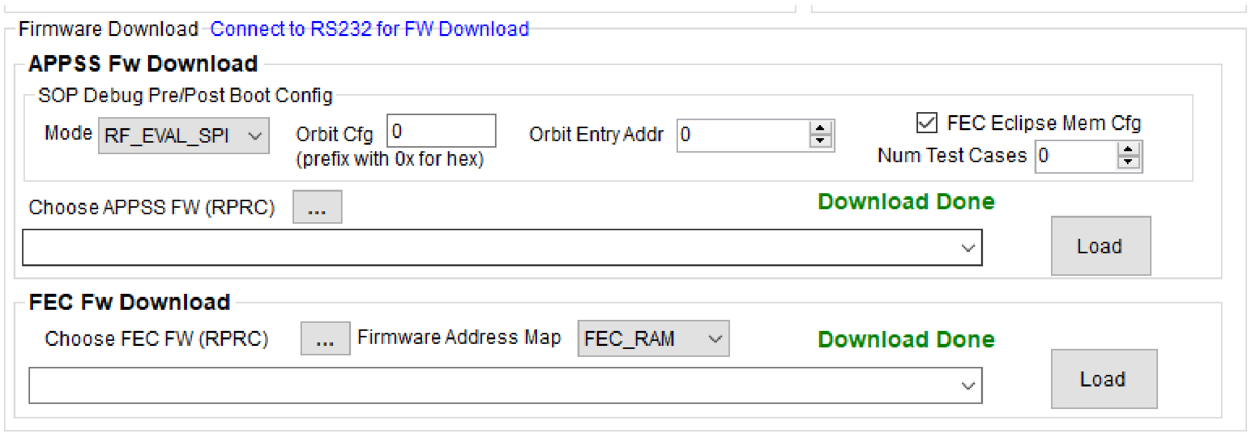
Since the default baud rate configured in the device is 115200 bps, mmWaveStudio attempts to connect in 115200 baud rate, and then configures the RS232 module to re-establish connection in 921600 baud rate.

Device status is get updated on same tab upon successful RS232 connection.

6.1.3 Firmware Download

The xWRL68xx firmware images are downloaded by the mmWaveStudio to the xWRL68xx device in SOP Development mode. The development mode provides a debug connection to the device using the dedicated RS232 interface such as writing and reading registers.

Refer mmWave DFP package for latest FEC and APPSS .rig image files. DFP is part of mmWave SDK package.



Firmware Download Connect to RS232 for FW Download

APPSS Fw Download

SOP Debug Pre/Post Boot Config

Mode: RF_EVAL_SPI

Orbit Cfg (prefix with 0x for hex): 0

Orbit Entry Addr: 0

☑ FEC Eclipse Mem Cfg

Num Test Cases: 0

Choose APPSS FW (RPRC)

Download Done

FEC Fw Download

Choose FEC FW (RPRC)

Firmware Address Map: FEC_RAM

Download Done

Figure 6.4. Firmware Download

The firmware download steps are as follows:

4. Select SOP Development Mode (SOP selection is not needed when using DCA1000 EVM. User has to set the jumpers on xWRL68xx BOOST to set the device in SOP mode.

5. Connect over RS232

6. Refer mws.sopDebugFwPreBootCfg and mws.sopDebugFwPostBootCfg for boot related configurations for SOP development mode

7. In the Firmware Download area of the Connect tab, set the FECSS and APPSS firmware. Click the ... button and browse to the location of the file

8. Click Load next to the APPSS & FEC firmware. The mmWaveStudio application begins downloading the firmware files entered in the previous steps.
   Refer mws.fwBuildImageDownload for download options configuration.

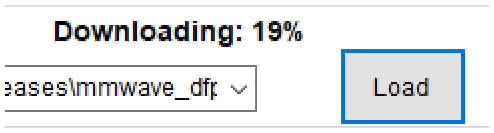


Figure 6.5. Firmware Download Progress

9. Once the FEC firmware download is complete, click Load next to the APPSS firmware update.

6.1.4 SPI Connection & FEC 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 APPSS firmware download is complete, Device Power On for SPI Connection.

2. The PowerOn button becomes Disconnect indicating a success. If Device Power On does not succeed you may need to erase the serial flash and try again using the Uniflash tool.

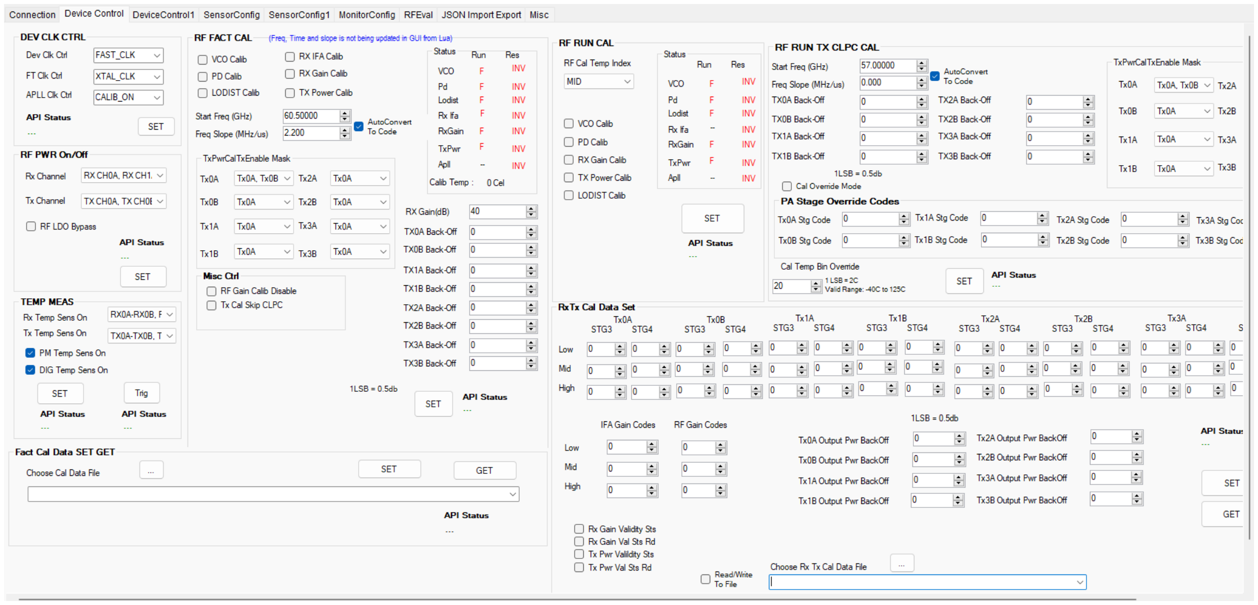
3. Once SPI Connection is successful, Click FEC PowerUp button. This command initiates FEC power up. Now, the user can issue commands to the device over SPI Communication interface.

4. On Successful device Power Up, Rfeval version will get updated in Firmware Status Section

5. On Successful FEC Power Up RFS, mmwavelink and fecsslib version will get updated in Firmware Status Section

6.2 Device Control Tab

These tabs contain Device Control DFP APIs. Refer to lua documentation mmwave_studio<version>\docs\mmwave_studio_lua_api_documentation.pdf for API parameter details.



DEV CLK CTRL
Dev Clk Ctrl: FAST_CLK
FT Clk Ctrl: XTAL_CLK
APLL Clk Ctrl: CALIB_ON

RF PWR On/Off
Rx Channel: RX CH0A, RX CH1.
Tx Channel: TX CH0A, TX CH0E
☐ RF LDO Bypass

TEMP MEAS
Rx Temp Sens On: RX0A-RX0B, F
Tx Temp Sens On: TX0A-TX0B, T
☑ PM Temp Sens On
☑ DIG Temp Sens On

RF FACT CAL (Freq, Time and slope is not being updated in GUI from Lua)
☐ VCO Calib ☐ RX IFA Calib
☐ PD Calib ☐ RX Gain Calib
☐ LODIST Calib ☐ TX Power Calib
Start Freq (GHz): 60.50000
Freq Slope (MHz/us): 2.200
☑ AutoConvert To Code

Status	Run	Res
VCO	F	INV
Pd	F	INV
Lodist	F	INV
Rx Ifa	F	INV
RxGain	F	INV
TxPwr	F	INV
Apll	--	INV

Calib Temp : 0 Cel

TxPwrCalTxEnable Mask
Tx0A: Tx0A, Tx0B Tx2A: Tx0A
Tx0B: Tx0A Tx2B: Tx0A
Tx1A: Tx0A Tx3A: Tx0A
Tx1B: Tx0A Tx3B: Tx0A

RX Gain(dB): 40
TX0A Back-Off: 0
TX0B Back-Off: 0
TX1A Back-Off: 0
TX1B Back-Off: 0
TX2A Back-Off: 0
TX2B Back-Off: 0
TX3A Back-Off: 0
TX3B Back-Off: 0
1LSB = 0.5db

Misc Ctrl
☐ RF Gain Calib Disable
☐ Tx Cal Skip CLPC

Fact Cal Data SET GET
Choose Cal Data File

RF RUN CAL
RF Cal Temp Index: MID
☐ VCO Calib
☐ PD Calib
☐ RX Gain Calib
☐ TX Power Calib
☐ LODIST Calib

Status	Run	Res
VCO	F	INV
Pd	F	INV
Lodist	F	INV
Rx Ifa	--	INV
RxGain	F	INV
TxPwr	F	INV
Apll	--	INV

RF RUN TX CLPC CAL
Start Freq (GHz): 57.00000
Freq Slope (MHz/us): 0.000
☑ AutoConvert To Code
TX0A Back-Off: 0 TX2A Back-Off: 0
TX0B Back-Off: 0 TX2B Back-Off: 0
TX1A Back-Off: 0 TX3A Back-Off: 0
TX1B Back-Off: 0 TX3B Back-Off: 0
1LSB = 0.5db
☐ Cal Override Mode

TxPwrCalTxEnable Mask
Tx0A: Tx0A, Tx0B Tx2A
Tx0B: Tx0A Tx2B
Tx1A: Tx0A Tx3A
Tx1B: Tx0A Tx3B

PA Stage Override Codes
Tx0A Stg Code: 0 Tx1A Stg Code: 0 Tx2A Stg Code: 0 Tx3A Stg Coc
Tx0B Stg Code: 0 Tx1B Stg Code: 0 Tx2B Stg Code: 0 Tx3B Stg Coc

Cal Temp Bin Override: 20
1 LSB = 2C
Valid Range: -40C to 125C

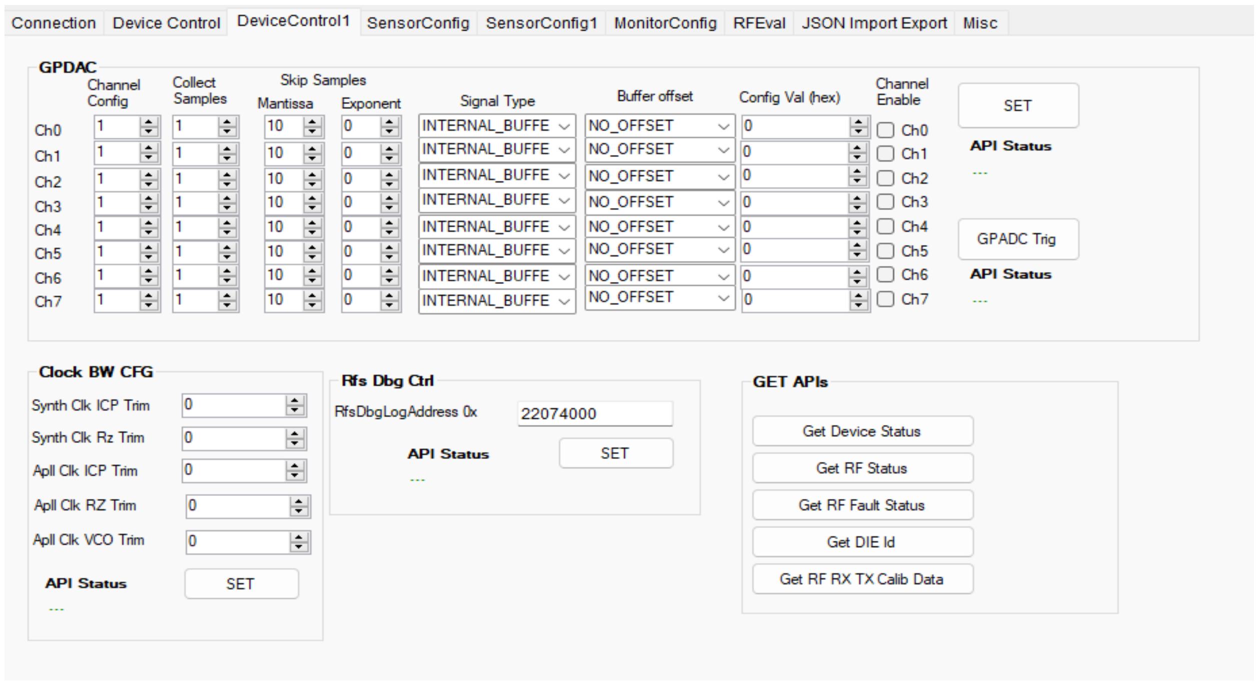
RxTx Cal Data Set

	Tx0A		Tx0B		Tx1A		Tx1B		Tx2A		Tx2B		Tx3A		
	STG3	STG4	STG3	STG4	STG3	STG4	STG3	STG4	STG3	STG4	STG3	STG4	STG3	STG4	S
Low	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mid	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
High	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	IFA Gain Codes	RF Gain Codes
Low	0	0
Mid	0	0
High	0	0

1LSB = 0.5db
Tx0A Output Pwr BackOff: 0 Tx2A Output Pwr BackOff: 0
Tx0B Output Pwr BackOff: 0 Tx2B Output Pwr BackOff: 0
Tx1A Output Pwr BackOff: 0 Tx3A Output Pwr BackOff: 0
Tx1B Output Pwr BackOff: 0 Tx3B Output Pwr BackOff: 0

☐ Rx Gain Validity Sts
☐ Rx Gain Val Sts Rd
☐ Tx Pwr Valildity Sts
☐ Tx Pwr Val Sts Rd
☐ Read/Write To File
Choose Rx Tx Cal Data File



Connection | Device Control | DeviceControl1 | SensorConfig | SensorConfig1 | MonitorConfig | RFEval | JSON Import Export | Misc

GPDAC

	Channel Config	Collect Samples	Skip Samples Mantissa	Skip Samples Exponent	Signal Type	Buffer offset	Config Val (hex)	Channel Enable
Ch0	1	1	10	0	INTERNAL_BUFFE	NO_OFFSET	0	☐ Ch0
Ch1	1	1	10	0	INTERNAL_BUFFE	NO_OFFSET	0	☐ Ch1
Ch2	1	1	10	0	INTERNAL_BUFFE	NO_OFFSET	0	☐ Ch2
Ch3	1	1	10	0	INTERNAL_BUFFE	NO_OFFSET	0	☐ Ch3
Ch4	1	1	10	0	INTERNAL_BUFFE	NO_OFFSET	0	☐ Ch4
Ch5	1	1	10	0	INTERNAL_BUFFE	NO_OFFSET	0	☐ Ch5
Ch6	1	1	10	0	INTERNAL_BUFFE	NO_OFFSET	0	☐ Ch6
Ch7	1	1	10	0	INTERNAL_BUFFE	NO_OFFSET	0	☐ Ch7

SET — API Status ---
GPADC Trig — API Status ---

Clock BW CFG

Synth Clk ICP Trim	0
Synth Clk Rz Trim	0
Apll Clk ICP Trim	0
Apll Clk RZ Trim	0
Apll Clk VCO Trim	0

API Status --- SET

Rfs Dbg Ctrl

RfsDbgLogAddress 0x 22074000

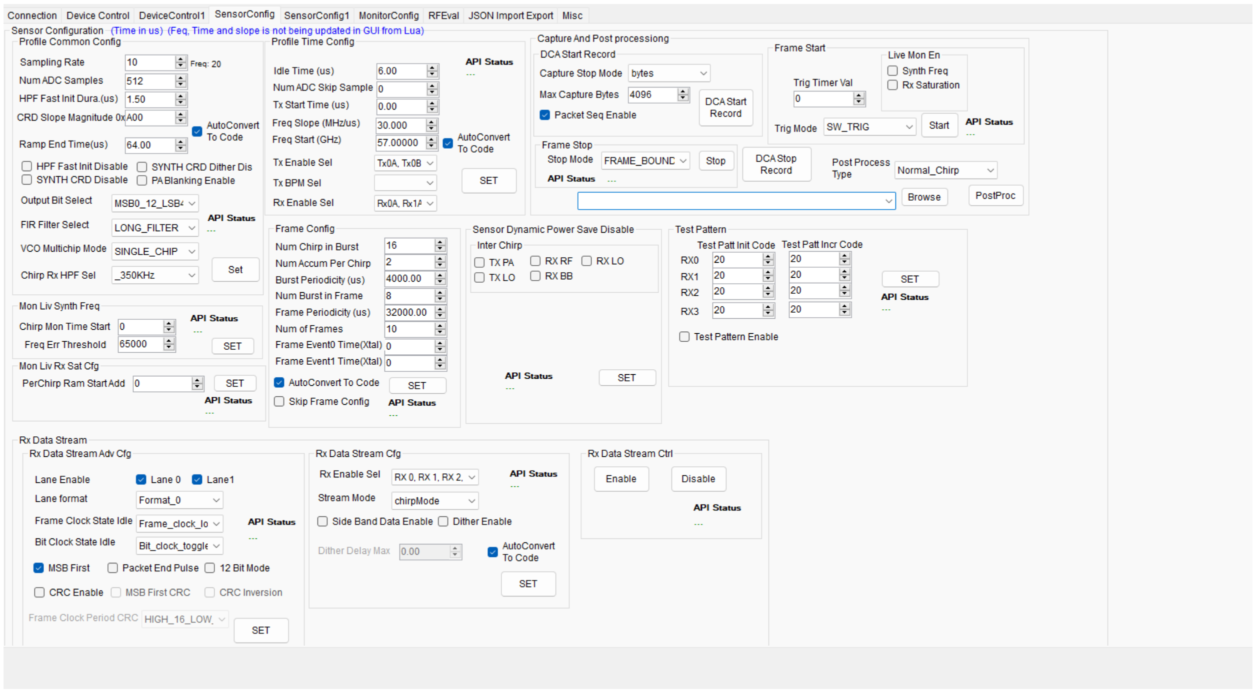
API Status --- SET

GET APIs

Get Device Status | Get RF Status | Get RF Fault Status | Get DIE Id | Get RF RX TX Calib Data

6.3 Sensor Config Tab

These tabs contain Sensor Control DFP APIs. Refer to lua documentation mmwave_studio<version>\docs\mmwave_studio_lua_api_documentation.pdf for API parameter details.



Connection | Device Control | DeviceControl1 | SensorConfig | SensorConfig1 | MonitorConfig | RFEval | JSON Import Export | Misc

Sensor Configuration (Time in us) (Feq, Time and slope is not being updated in GUI from Lua)

Profile Common Config

Sampling Rate	10 (Freq: 20)
Num ADC Samples	512
HPF Fast Init Dura.(us)	1.50
CRD Slope Magnitude 0x	A00
Ramp End Time(us)	64.00
Output Bit Select	MSB0_12_LSB4
FIR Filter Select	LONG_FILTER
VCO Multichip Mode	SINGLE_CHIP
Chirp Rx HPF Sel	_350KHz

✓ AutoConvert To Code
☐ HPF Fast Init Disable ☐ SYNTH CRD Dither Dis
☐ SYNTH CRD Disable ☐ PA Blanking Enable
API Status --- Set

Mon Liv Synth Freq

Chirp Mon Time Start 0
Freq Err Threshold 65000
API Status --- SET

Mon Liv Rx Sat Cfg

PerChirp Ram Start Add 0 SET
API Status ---

Profile Time Config

Idle Time (us)	6.00
Num ADC Skip Sample	0
Tx Start Time (us)	0.00
Freq Slope (MHz/us)	30.000
Freq Start (GHz)	57.00000
Tx Enable Sel	Tx0A, Tx0B
Tx BPM Sel	
Rx Enable Sel	Rx0A, Rx1A

✓ AutoConvert To Code
API Status --- SET

Frame Config

Num Chirp in Burst	16
Num Accum Per Chirp	2
Burst Periodicity (us)	4000.00
Num Burst in Frame	8
Frame Periodicity (us)	32000.00
Num of Frames	10
Frame Event0 Time(Xtal)	0
Frame Event1 Time(Xtal)	0

✓ AutoConvert To Code SET
☐ Skip Frame Config API Status ---

Capture And Post processiong

DCA Start Record: Capture Stop Mode bytes; Max Capture Bytes 4096; ✓ Packet Seq Enable; DCA Start Record

Frame Stop: Stop Mode FRAME_BOUND; Stop; API Status ---; DCA Stop Record

Post Process Type Normal_Chirp
Browse PostProc

Frame Start

Trig Timer Val 0
Trig Mode SW_TRIG Start
Live Mon En: ☐ Synth Freq ☐ Rx Saturation
API Status ---

Sensor Dynamic Power Save Disable

Inter Chirp: ☐ TX PA ☐ RX RF ☐ RX LO ☐ TX LO ☐ RX BB
API Status --- SET

Test Pattern

	Test Patt Init Code	Test Patt Incr Code
RX0	20	20
RX1	20	20
RX2	20	20
RX3	20	20

☐ Test Pattern Enable
SET API Status ---

Rx Data Stream

Rx Data Stream Adv Cfg

Lane Enable	✓ Lane 0 ✓ Lane1
Lane format	Format_0
Frame Clock State Idle	Frame_clock_lo
Bit Clock State Idle	Bit_clock_toggle
Frame Clock Period CRC	HIGH_16_LOW_

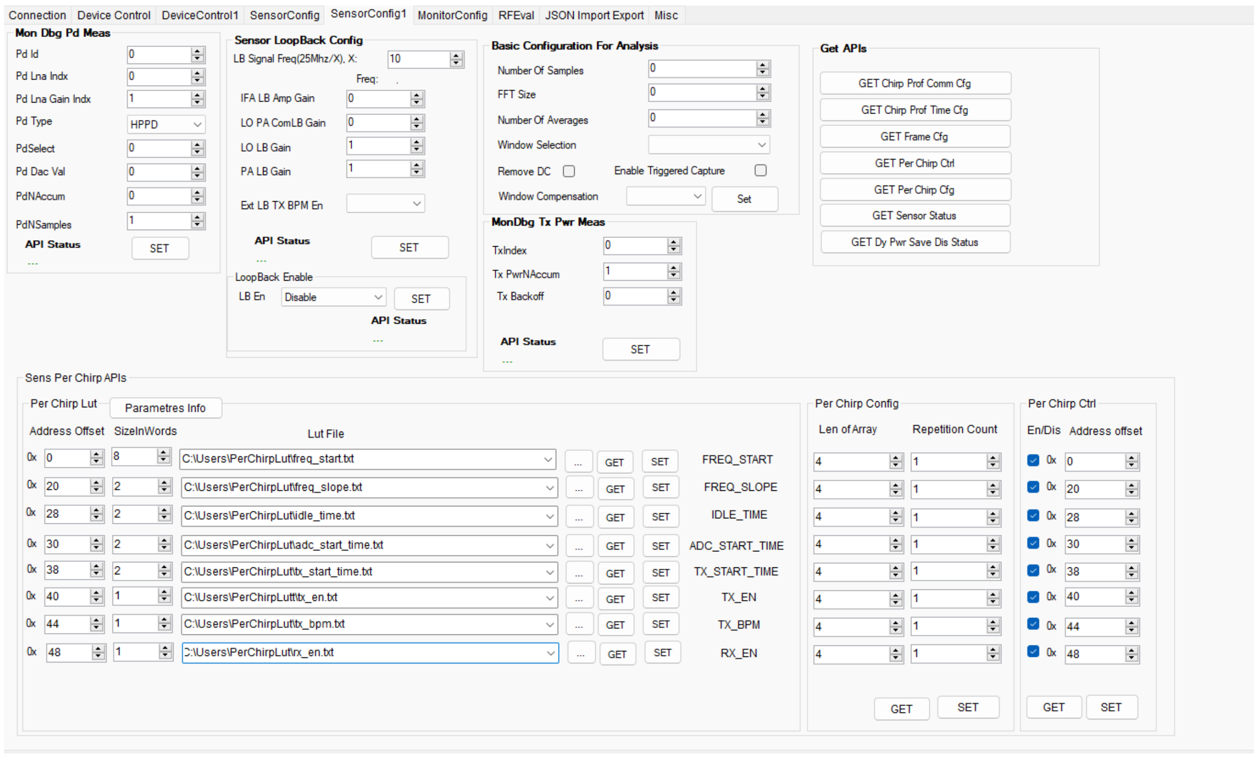
✓ MSB First ☐ Packet End Pulse ☐ 12 Bit Mode
☐ CRC Enable ☐ MSB First CRC ☐ CRC Inversion
API Status --- SET

Rx Data Stream Cfg

Rx Enable Sel RX 0, RX 1, RX 2,
Stream Mode chirpMode
☐ Side Band Data Enable ☐ Dither Enable
Dither Delay Max 0.00 ✓ AutoConvert To Code
API Status --- SET

Rx Data Stream Ctrl

Enable Disable
API Status ---



Mon Dbg Pd Meas
Pd Id 0
Pd Lna Indx 0
Pd Lna Gain Indx 1
Pd Type HPPD
PdSelect 0
Pd Dac Val 0
PdNAccum 0
PdNSamples 1
API Status SET
...

Sensor LoopBack Config
LB Signal Freq(25Mhz/X), X: 10
Freq: .
IFA LB Amp Gain 0
LO PA ComLB Gain 0
LO LB Gain 1
PA LB Gain 1
Ext LB TX BPM En
API Status SET
...
LoopBack Enable
LB En Disable SET
API Status
...

Basic Configuration For Analysis
Number Of Samples 0
FFT Size 0
Number Of Averages 0
Window Selection
Remove DC ☐ Enable Triggered Capture ☐
Window Compensation Set

MonDbg Tx Pwr Meas
TxIndex 0
Tx PwrNAccum 1
Tx Backoff 0
API Status SET
...

Get APIs
GET Chirp Prof Comm Cfg
GET Chirp Prof Time Cfg
GET Frame Cfg
GET Per Chirp Ctrl
GET Per Chirp Cfg
GET Sensor Status
GET Dy Pwr Save Dis Status

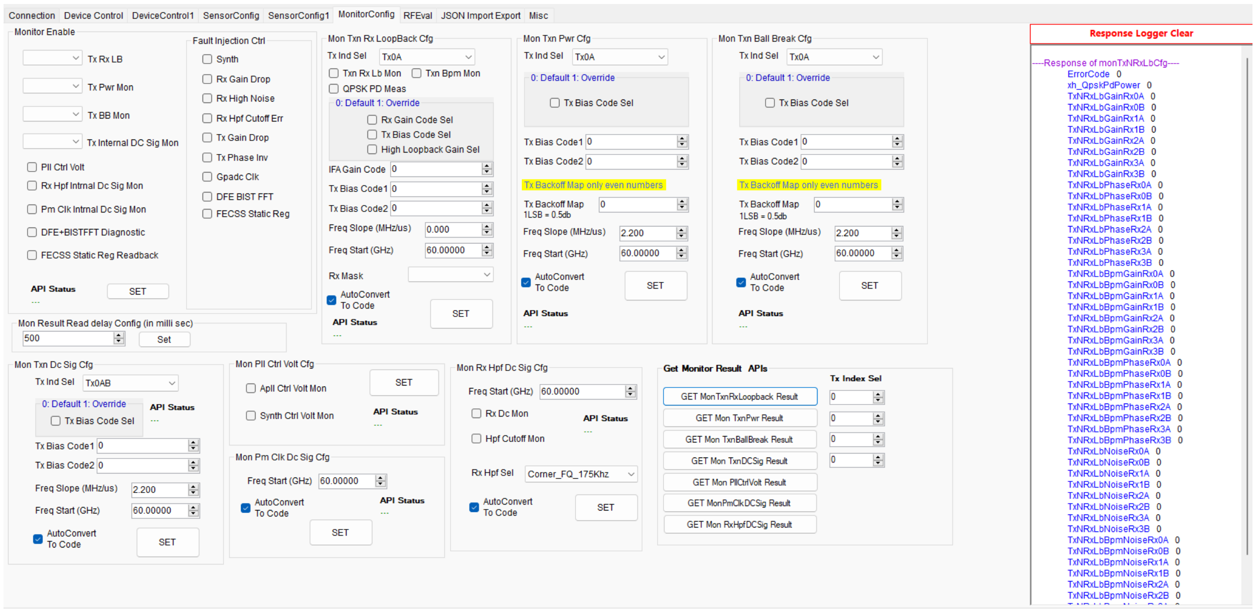
Sens Per Chirp APIs
Per Chirp Lut Parametres Info

Address Offset	SizeInWords	Lut File					Len of Array	Repetition Count	En/Dis	Address offset
0x 0	8	C:\Users\PerChirpLut\freq_start.txt	...	GET	SET	FREQ_START	4	1	✓	0x 0
0x 20	2	C:\Users\PerChirpLut\freq_slope.txt	...	GET	SET	FREQ_SLOPE	4	1	✓	0x 20
0x 28	2	C:\Users\PerChirpLut\idle_time.txt	...	GET	SET	IDLE_TIME	4	1	✓	0x 28
0x 30	2	C:\Users\PerChirpLut\adc_start_time.txt	...	GET	SET	ADC_START_TIME	4	1	✓	0x 30
0x 38	2	C:\Users\PerChirpLut\tx_start_time.txt	...	GET	SET	TX_START_TIME	4	1	✓	0x 38
0x 40	1	C:\Users\PerChirpLut\tx_en.txt	...	GET	SET	TX_EN	4	1	✓	0x 40
0x 44	1	C:\Users\PerChirpLut\tx_bpm.txt	...	GET	SET	TX_BPM	4	1	✓	0x 44
0x 48	1	C:\Users\PerChirpLut\rx_en.txt	...	GET	SET	RX_EN	4	1	✓	0x 48

Per Chirp Config GET SET
Per Chirp Ctrl GET SET

6.4 Monitor Config Tab

These tabs contain Monitor DFP APIs. Refer to lua documentation mmwave_studio<version>\docs\mmwave_studio_lua_api_documentation.pdf for API parameter details.



Monitor Enable
Tx Rx LB
Tx Pwr Mon
Tx BB Mon
Tx Internal DC Sig Mon
☐ Pll Ctrl Volt
☐ Rx Hpf Intrnal Dc Sig Mon
☐ Pm Clk Intrnal Dc Sig Mon
☐ DFE+BISTFFT Diagnostic
☐ FECSS Static Reg Readback
API Status SET
...

Fault Injection Ctrl
☐ Synth
☐ Rx Gain Drop
☐ Rx High Noise
☐ Rx Hpf Cutoff Err
☐ Tx Gain Drop
☐ Tx Phase Inv
☐ Gpadc Clk
☐ DFE BIST FFT
☐ FECSS Static Reg

Mon Result Read delay Config (in milli sec)
500 Set

Mon Txn Rx LoopBack Cfg
Tx Ind Sel Tx0A
☐ Txn Rx Lb Mon ☐ Txn Bpm Mon
☐ QPSK PD Meas
0: Default 1: Override
☐ Rx Gain Code Sel
☐ Tx Bias Code Sel
☐ High Loopback Gain Sel
IFA Gain Code 0
Tx Bias Code1 0
Tx Bias Code2 0
Freq Slope (MHz/us) 0.000
Freq Start (GHz) 60.00000
Rx Mask
✓ AutoConvert To Code SET
API Status
...

Mon Txn Pwr Cfg
Tx Ind Sel Tx0A
0: Default 1: Override
☐ Tx Bias Code Sel
Tx Bias Code1 0
Tx Bias Code2 0
Tx Backoff Map only even numbers
Tx Backoff Map 1LSB = 0.5db 0
Freq Slope (MHz/us) 2.200
Freq Start (GHz) 60.00000
✓ AutoConvert To Code SET
API Status
...

Mon Txn Ball Break Cfg
Tx Ind Sel Tx0A
0: Default 1: Override
☐ Tx Bias Code Sel
Tx Bias Code1 0
Tx Bias Code2 0
Tx Backoff Map only even numbers
Tx Backoff Map 1LSB = 0.5db 0
Freq Slope (MHz/us) 2.200
Freq Start (GHz) 60.00000
✓ AutoConvert To Code SET
API Status
...

Mon Txn Dc Sig Cfg
Tx Ind Sel Tx0AB
0: Default 1: Override
☐ Tx Bias Code Sel
API Status
...
Tx Bias Code1 0
Tx Bias Code2 0
Freq Slope (MHz/us) 2.200
Freq Start (GHz) 60.00000
✓ AutoConvert To Code SET

Mon Pll Ctrl Volt Cfg
☐ Apll Ctrl Volt Mon SET
☐ Synth Ctrl Volt Mon
API Status
...

Mon Pm Clk Dc Sig Cfg
Freq Start (GHz) 60.00000
✓ AutoConvert To Code
API Status
...
SET

Mon Rx Hpf Dc Sig Cfg
Freq Start (GHz) 60.00000
☐ Rx Dc Mon
API Status
...
☐ Hpf Cutoff Mon
Rx Hpf Sel Corner_FQ_175Khz
✓ AutoConvert To Code SET

Get Monitor Result APIs

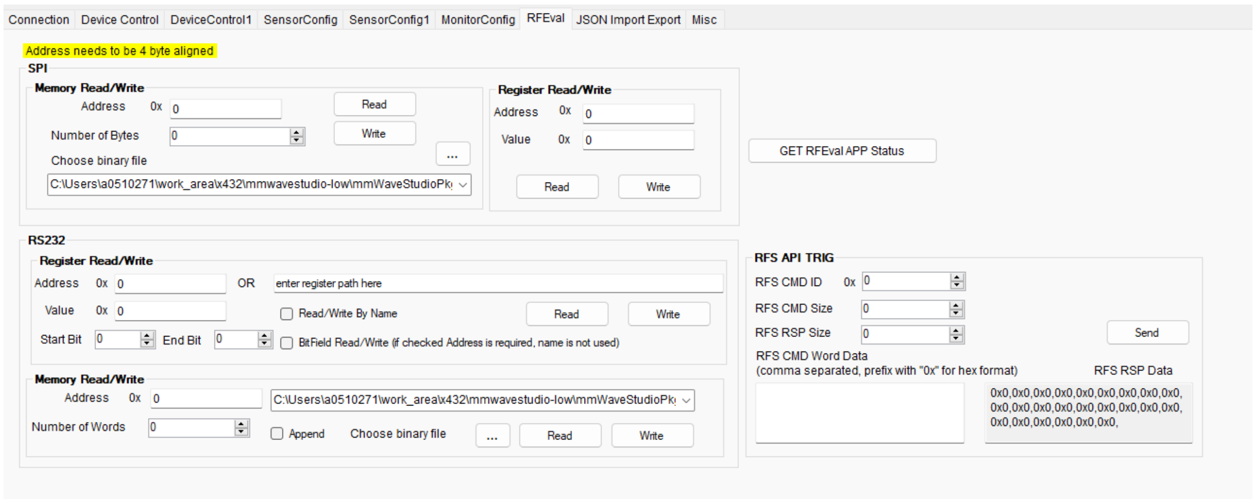
	Tx Index Sel
GET MonTxnRxLoopback Result	0
GET Mon TxnPwr Result	0
GET Mon TxnBallBreak Result	0
GET Mon TxnDCSig Result	0
GET Mon PllCtrlVolt Result	
GET MonPmClkDCSig Result	
GET Mon RxHpfDCSig Result	

Response Logger Clear

```
----Response of monTxNRxLbCfg----
        ErrorCode  0
        xh_QpskPdPower  0
        TxNRxLbGainRx0A  0
        TxNRxLbGainRx0B  0
        TxNRxLbGainRx1A  0
        TxNRxLbGainRx1B  0
        TxNRxLbGainRx2A  0
        TxNRxLbGainRx2B  0
        TxNRxLbGainRx3A  0
        TxNRxLbGainRx3B  0
        TxNRxLbPhaseRx0A  0
        TxNRxLbPhaseRx0B  0
        TxNRxLbPhaseRx1A  0
        TxNRxLbPhaseRx1B  0
        TxNRxLbPhaseRx2A  0
        TxNRxLbPhaseRx2B  0
        TxNRxLbPhaseRx3A  0
        TxNRxLbPhaseRx3B  0
        TxNRxLbBpmGainRx0A  0
        TxNRxLbBpmGainRx0B  0
        TxNRxLbBpmGainRx1A  0
        TxNRxLbBpmGainRx1B  0
        TxNRxLbBpmGainRx2A  0
        TxNRxLbBpmGainRx2B  0
        TxNRxLbBpmGainRx3A  0
        TxNRxLbBpmGainRx3B  0
        TxNRxLbBpmPhaseRx0A  0
        TxNRxLbBpmPhaseRx0B  0
        TxNRxLbBpmPhaseRx1A  0
        TxNRxLbBpmPhaseRx1B  0
        TxNRxLbBpmPhaseRx2A  0
        TxNRxLbBpmPhaseRx2B  0
        TxNRxLbBpmPhaseRx3A  0
        TxNRxLbBpmPhaseRx3B  0
        TxNRxLbNoiseRx0A  0
        TxNRxLbNoiseRx0B  0
        TxNRxLbNoiseRx1A  0
        TxNRxLbNoiseRx1B  0
        TxNRxLbNoiseRx2A  0
        TxNRxLbNoiseRx2B  0
        TxNRxLbNoiseRx3A  0
        TxNRxLbNoiseRx3B  0
        TxNRxLbBpmNoiseRx0A  0
        TxNRxLbBpmNoiseRx0B  0
        TxNRxLbBpmNoiseRx1A  0
        TxNRxLbBpmNoiseRx1B  0
        TxNRxLbBpmNoiseRx2A  0
        TxNRxLbBpmNoiseRx2B  0
```

6.5 RFEval

These tabs contain Register/Memory Read/Write APIs over RS232/SPI and RFEval Custom Commands. Refer to lua documentation mmwave_studio<version>\docs\mmwave_studio_lua_api_documentation.pdf for API parameter details.



Address needs to be 4 byte aligned

SPI – Memory Read/Write: Address 0x 0; Number of Bytes 0; Choose binary file: C:\Users\a0510271\work_area\x432\mmwavestudio-low\mmWaveStudioPk

SPI – Register Read/Write: Address 0x 0; Value 0x 0

RS232 – Register Read/Write: Address 0x 0; OR enter register path here; Value 0x 0; ☐ Read/Write By Name; Start Bit 0; End Bit 0; ☐ BitField Read/Write (if checked Address is required, name is not used)

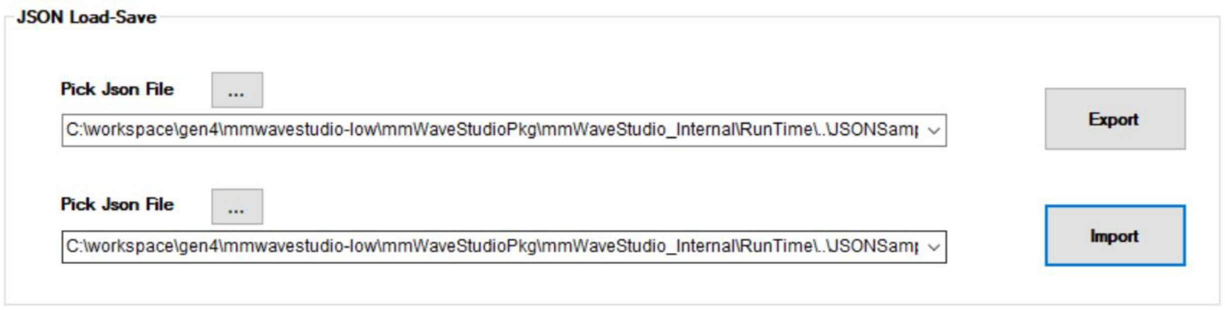
RS232 – Memory Read/Write: Address 0x 0; C:\Users\a0510271\work_area\x432\mmwavestudio-low\mmWaveStudioPk; Number of Words 0; ☐ Append

RFS API TRIG: RFS CMD ID 0x 0; RFS CMD Size 0; RFS RSP Size 0; RFS CMD Word Data (comma separated, prefix with "0x" for hex format)

RFS RSP Data: 0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0, 0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0, 0x0,0x0,0x0,0x0,0x0,0x0,0x0,

6.6 JSON Import Export

Use this tab to save GUI configuration into a json file. This file can be used to import configuration into GUI.



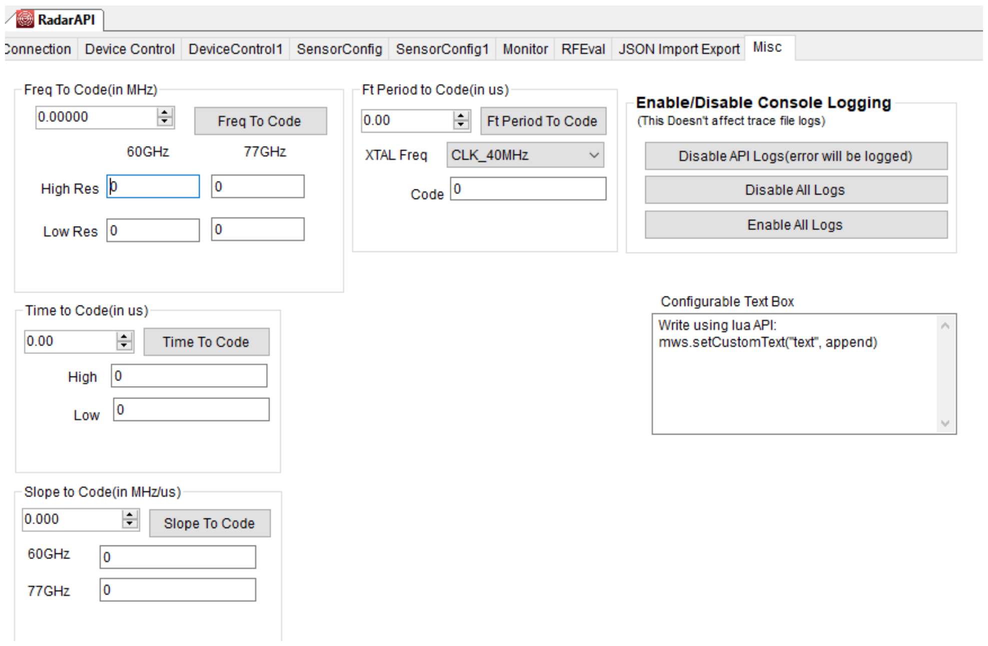
C:\workspace\gen4\mmwavestudio-low\mmWaveStudioPkg\mmWaveStudio_Internal\RunTime\..\JSONSam

C:\workspace\gen4\mmwavestudio-low\mmWaveStudioPkg\mmWaveStudio_Internal\RunTime\..\JSONSam

6.7 Misc

This Tab Contains Conversion APIs and some other debug features like:

1. Enable/Disable log print in console
2. Update a custom text box from lua script



Freq To Code(in MHz): 0.00000

	60GHz	77GHz
High Res	0	0
Low Res	0	0

Ft Period to Code(in us): 0.00

XTAL Freq: CLK_40MHz

Code: 0

Time to Code(in us): 0.00

High: 0

Low: 0

Configurable Text Box:

Write using lua API:
mws.setCustomText("text", append)

Slope to Code(in MHz/us): 0.000

60GHz: 0

77GHz: 0

7 APIs Programming sequence

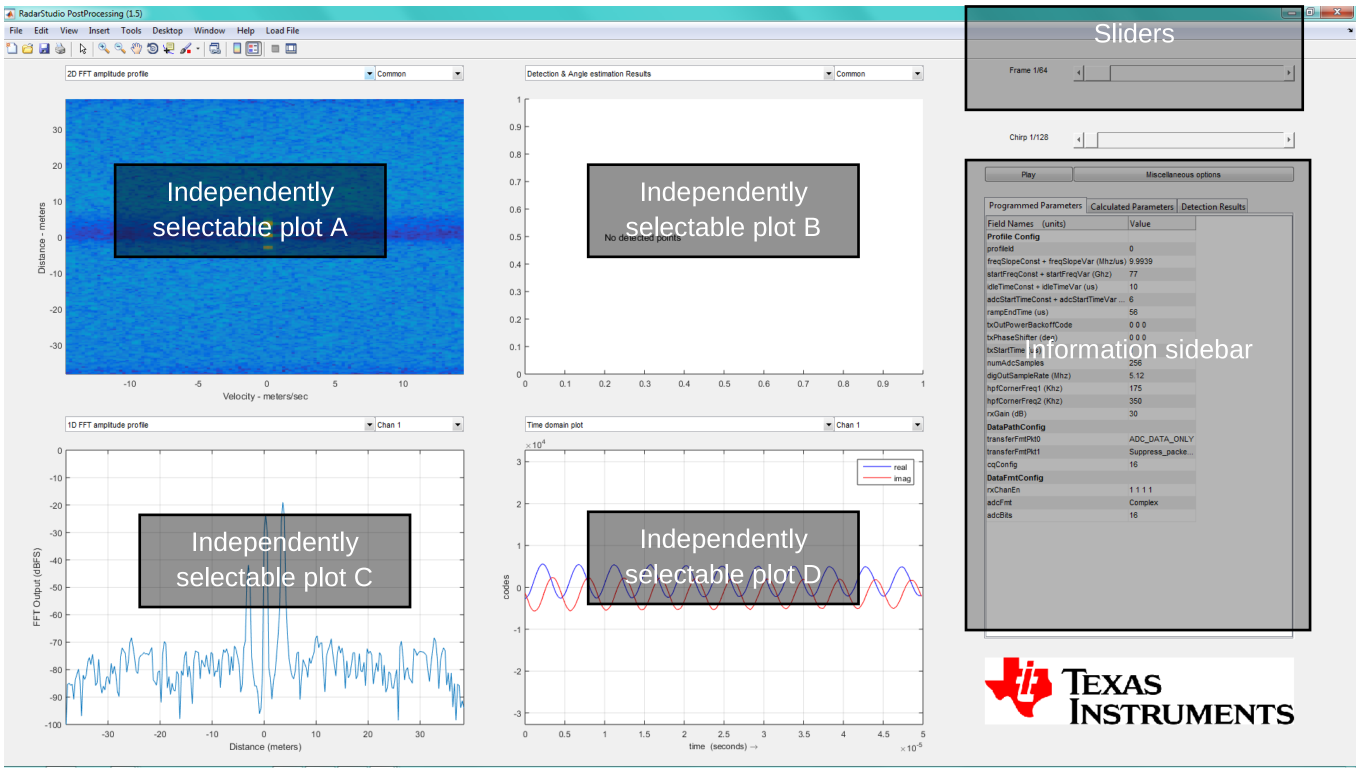
Refer the LUA documentation (mmwave_studio<version>\docs\mmwave_studio_lua_api_documentation.pdf) for API programming sequences under section **"API Programming Sequence".**

8 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:



Field Names (units)	Value
Profile Config	
profileId	0
freqSlopeConst + freqSlopeVar (Mhz/us)	9.9939
startFreqConst + startFreqVar (Ghz)	77
idleTimeConst + idleTimeVar (us)	10
adcStartTimeConst + adcStartTimeVar ...	6
rampEndTime (us)	56
txOutPowerBackoffCode	0 0 0
txPhaseShifter (deg)	0 0 0
txStartTime (us)	[illegible]
numAdcSamples	256
digOutSampleRate (Mhz)	5.12
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

Figure 8.1. 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.

NOTE:

1. The Studio PostProc visualizer only supports data captured using RDIF Data Swizzling Mode 2 (“Pin0_bit0_Cycle3” in GUI).

2. If capture size exceeds 1GB, DCA Capture CLI splits the captured data into multiple files of size <= 1GB. PostProc only processes first file from these generated files. Hence only maximum of 1 GB data can be used for radar Post Processing.

8.1 List of plots

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

8.1.1 Basic plots

8.1.1.1 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.

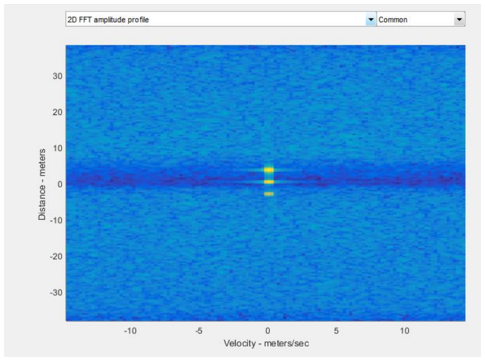


Figure 8.2. 2D FFT Profile

8.1.1.2 *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.

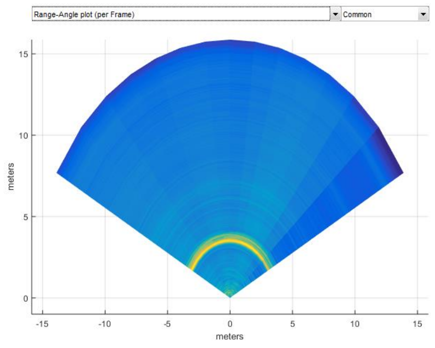


Figure 8.3. Range Angle plot

8.1.1.3 *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.

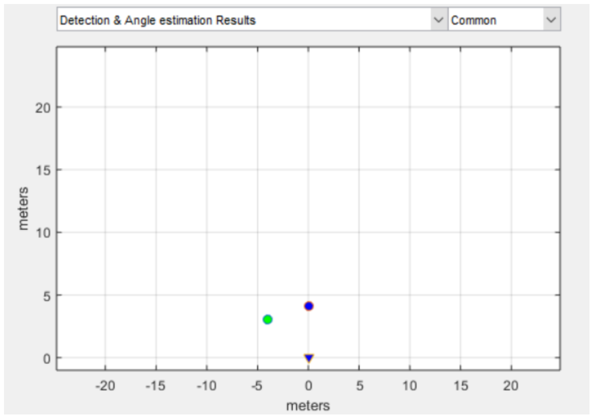


Figure 8.4. Detection and Angle Estimation results

8.1.1.4 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.

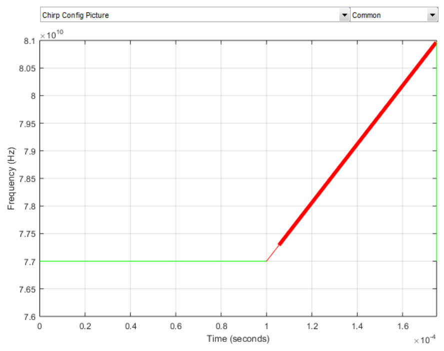


Figure 8.5. Chirp Config Picture

8.1.1.5 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).

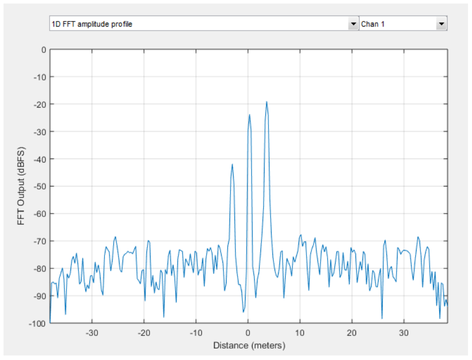


Figure 8.6. 1D FFT Profile

8.1.1.6 *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).

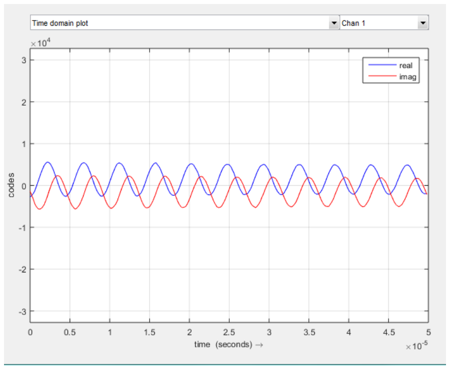


Figure 8.7. Time domain plot

8.1.2 *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. ADC/IF Saturation Indicator

8.1.3 *Characterization plots*

8.1.3.1 *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).

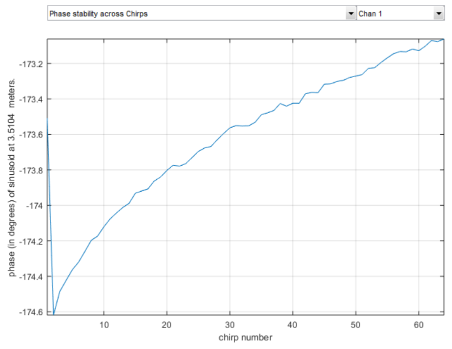


Figure 8.8. Phase Stability

8.1.3.2 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.

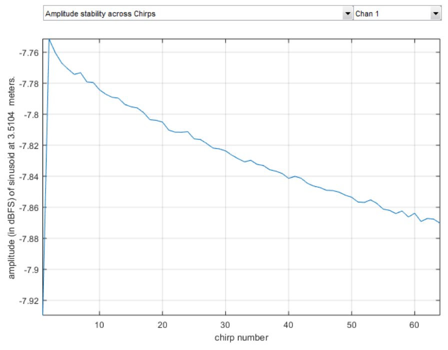


Figure 8.9. Amplitude stability

8.1.3.3 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 '-$max_{velocity}'$.

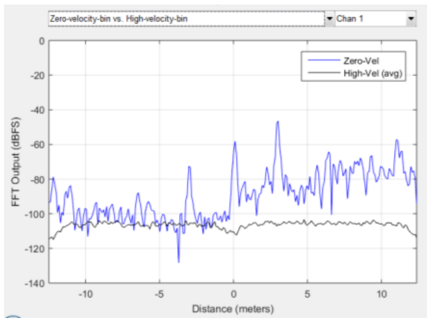


Figure 8.10. Zero-velocity bin vs High velocity bin.

8.2 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.

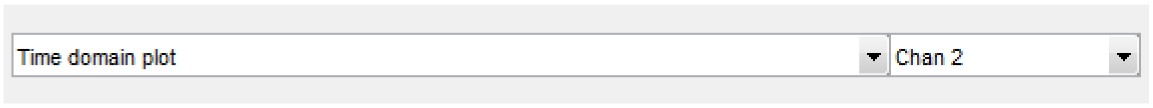


Figure 8.11. Channel select

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

8.3 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.

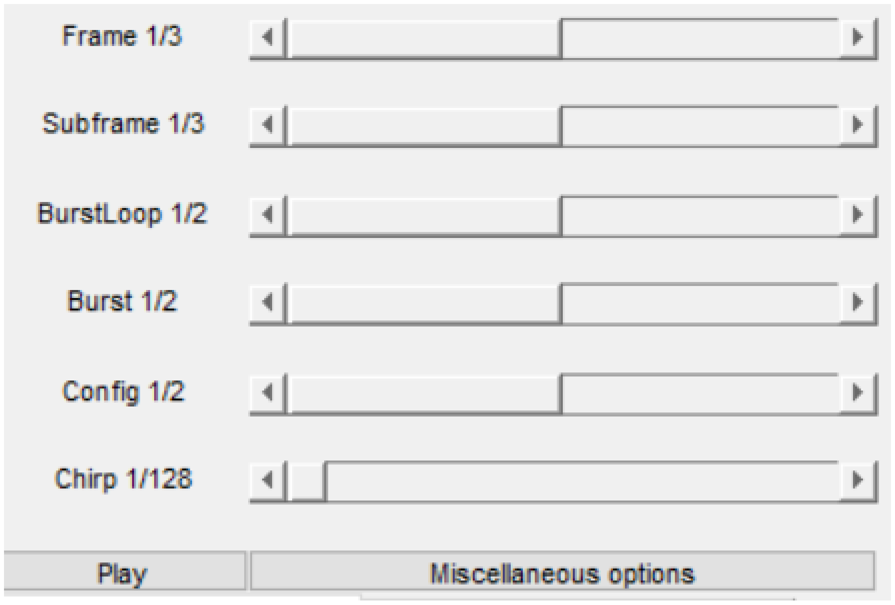
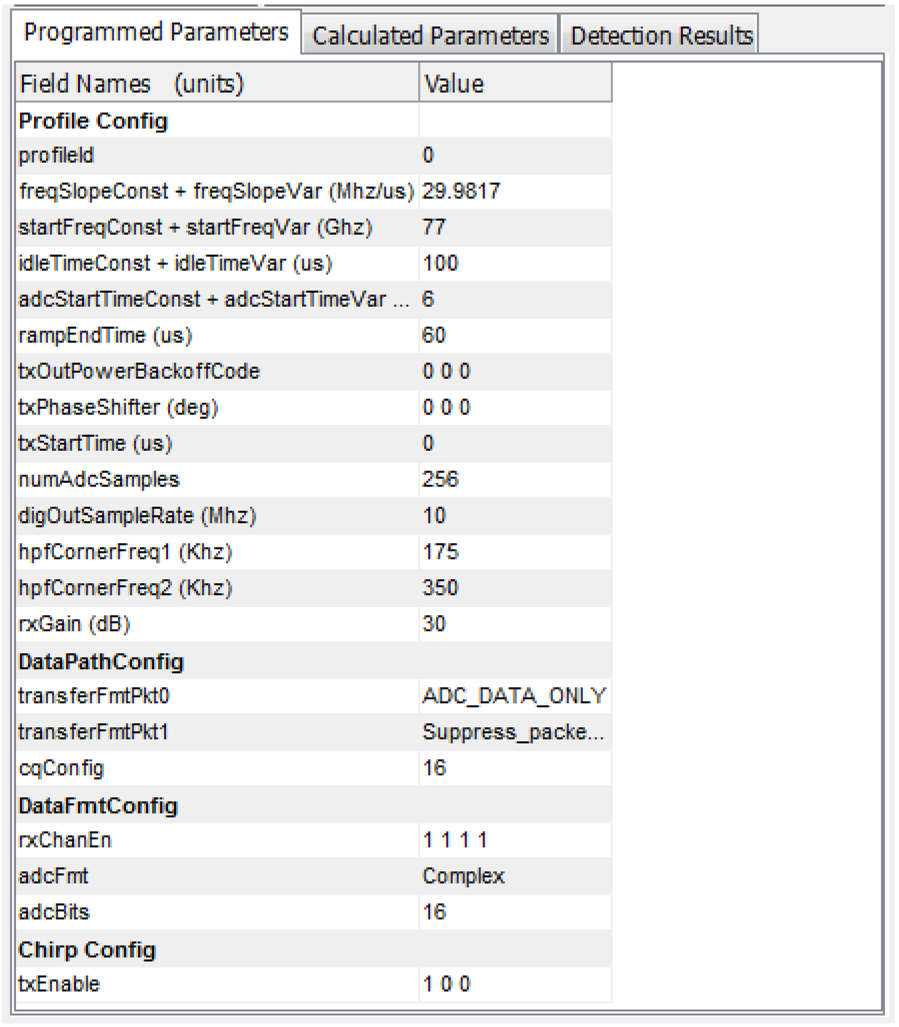


Figure 8.12. Sliders

8.4 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 8.13. Information sidebar

These tabs are explained in the subsequent subsections.

NOTE: Chirp parameters programmed via the Sensor Per Chirp APIs are not reflected in the Studio PostProc visualizer information sidebar.

8.4.1 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.

8.4.2 Calculated parameters

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

a. 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.

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

c. Range resolution

d. Velocity resolution

8.4.3 Detection results

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

8.5 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.

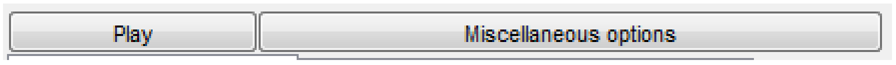
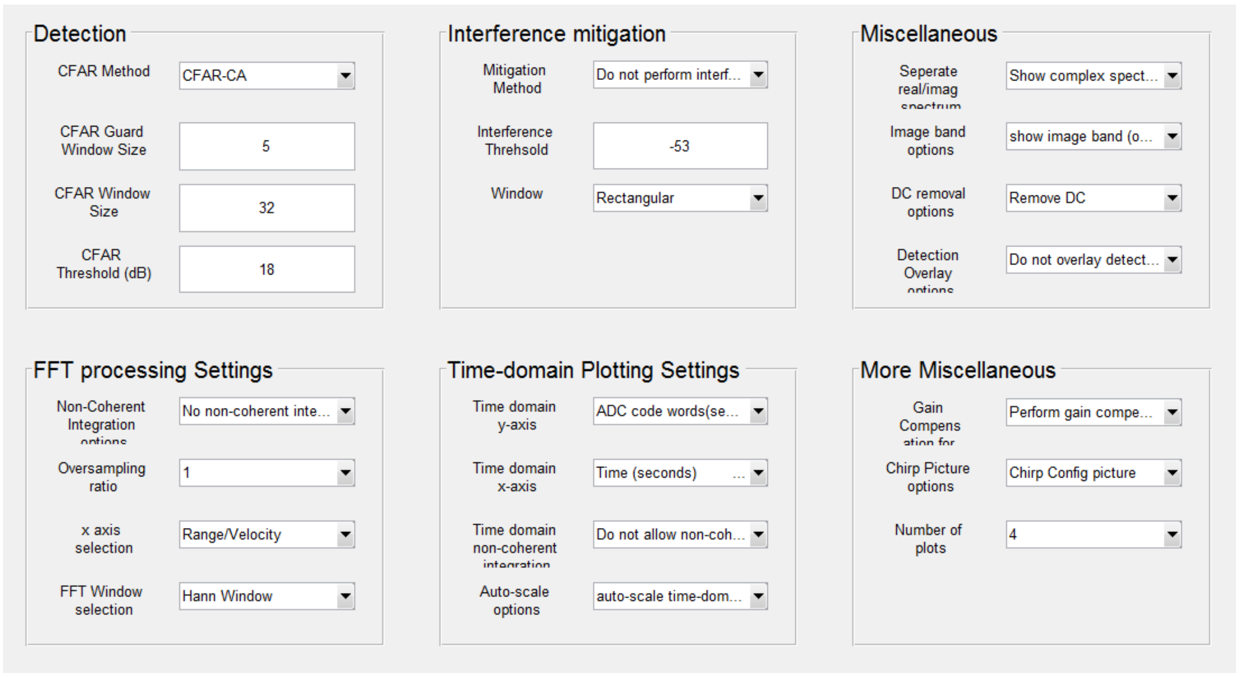


Figure 8.14. Play

8.6 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.



Detection
CFAR Method: CFAR-CA
CFAR Guard Window Size: 5
CFAR Window Size: 32
CFAR Threshold (dB): 18

Interference mitigation
Mitigation Method: Do not perform interf...
Interference Threhsold: -53
Window: Rectangular

Miscellaneous
Seperate real/imag spectrum: Show complex spect...
Image band options: show image band (o...
DC removal options: Remove DC
Detection Overlay options: Do not overlay detect...

FFT processing Settings
Non-Coherent Integration options: No non-coherent inte...
Oversampling ratio: 1
x axis selection: Range/Velocity
FFT Window selection: Hann Window

Time-domain Plotting Settings
Time domain y-axis: ADC code words(se...
Time domain x-axis: Time (seconds) ...
Time domain non-coherent integration: Do not allow non-coh...
Auto-scale options: auto-scale time-dom...

More Miscellaneous
Gain Compensation for: Perform gain compe...
Chirp Picture options: Chirp Config picture
Number of plots: 4

Figure 8.15. Miscellaneous options

8.6.1 Detection

The detection algorithm is 2D-CFAR-CA.

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

   i. 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.

   ii. 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.

   iii. 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.

8.6.2 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).

8.6.3 Miscellaneous

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

b. **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.

c. **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.

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

8.6.4 Time domain options

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

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

g. **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 $\sqrt{I^2+Q^2}$

h. **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.

8.6.5 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.

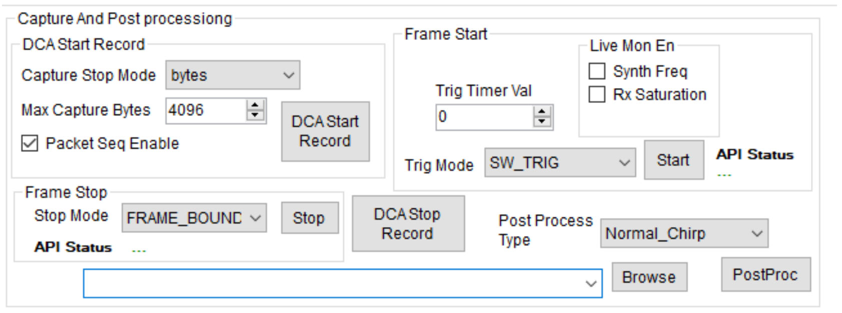
c. **Number of plots.** By default 4 plots are shown on the main window. This dropdown can change that to 1, 2, 4, or 9 independently configurable plots.

8.7 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.

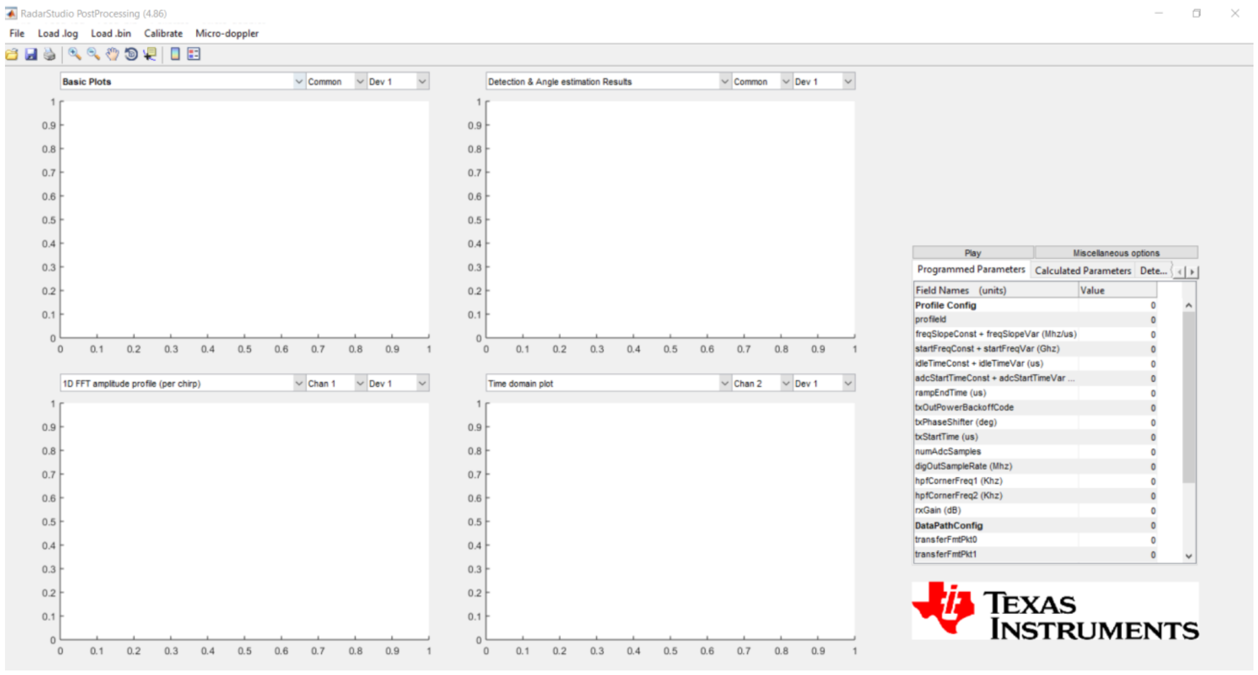
8.8 Standalone processing (post processing done in a different session from configuration and capture)

- 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.



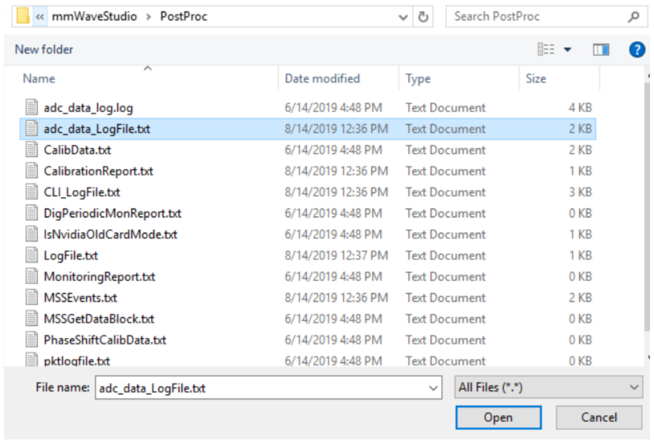
Capture Stop Mode: bytes
Max Capture Bytes: 4096
☑ Packet Seq Enable
Trig Timer Val: 0
Trig Mode: SW_TRIG
Stop Mode: FRAME_BOUND
API Status: ---
Post Process Type: Normal_Chirp

- An empty Post processing window will be opened



Field Names (units)	Value
Profile Config	0
profileId	0
freqSlopeConst + freqSlopeVar (Mhz/us)	0
startFreqConst + startFreqVar (Ghz)	0
idleTimeConst + idleTimeVar (us)	0
adcStartTimeConst + adcStartTimeVar ...	0
rampEndTime (us)	0
txOutPowerBackoffCode	0
txPhaseShifter (deg)	0
txStartTime (us)	0
numAdcSamples	0
digOutSampleRate (Mhz)	0
hpfCornerFreq1 (Khz)	0
hpfCornerFreq2 (Khz)	0
rxGain (dB)	0
DataPathConfig	0
transferFmtPkt0	0
transferFmtPkt1	0

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



« mmWaveStudio › PostProc

Search PostProc

New folder

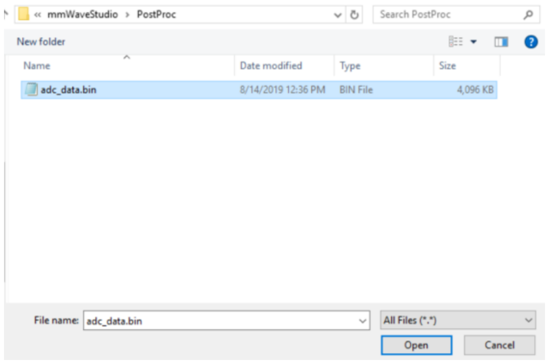
Name	Date modified	Type	Size
adc_data_log.log	6/14/2019 4:48 PM	Text Document	4 KB
adc_data_LogFile.txt	8/14/2019 12:36 PM	Text Document	2 KB
CalibData.txt	6/14/2019 4:48 PM	Text Document	2 KB
CalibrationReport.txt	8/14/2019 12:36 PM	Text Document	1 KB
CLI_LogFile.txt	8/14/2019 12:36 PM	Text Document	3 KB
DigPeriodicMonReport.txt	6/14/2019 4:48 PM	Text Document	0 KB
IsNvidiaOldCardMode.txt	6/14/2019 4:48 PM	Text Document	1 KB
LogFile.txt	8/14/2019 12:37 PM	Text Document	1 KB
MonitoringReport.txt	6/14/2019 4:48 PM	Text Document	0 KB
MSSEvents.txt	8/14/2019 12:36 PM	Text Document	2 KB
MSSGetDataBlock.txt	6/14/2019 4:48 PM	Text Document	0 KB
PhaseShiftCalibData.txt	6/14/2019 4:48 PM	Text Document	0 KB
pktlogfile.txt	6/14/2019 4:48 PM	Text Document	0 KB

File name: adc_data_LogFile.txt

All Files (*.*)

Open Cancel

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



« mmWaveStudio › PostProc

Search PostProc

New folder

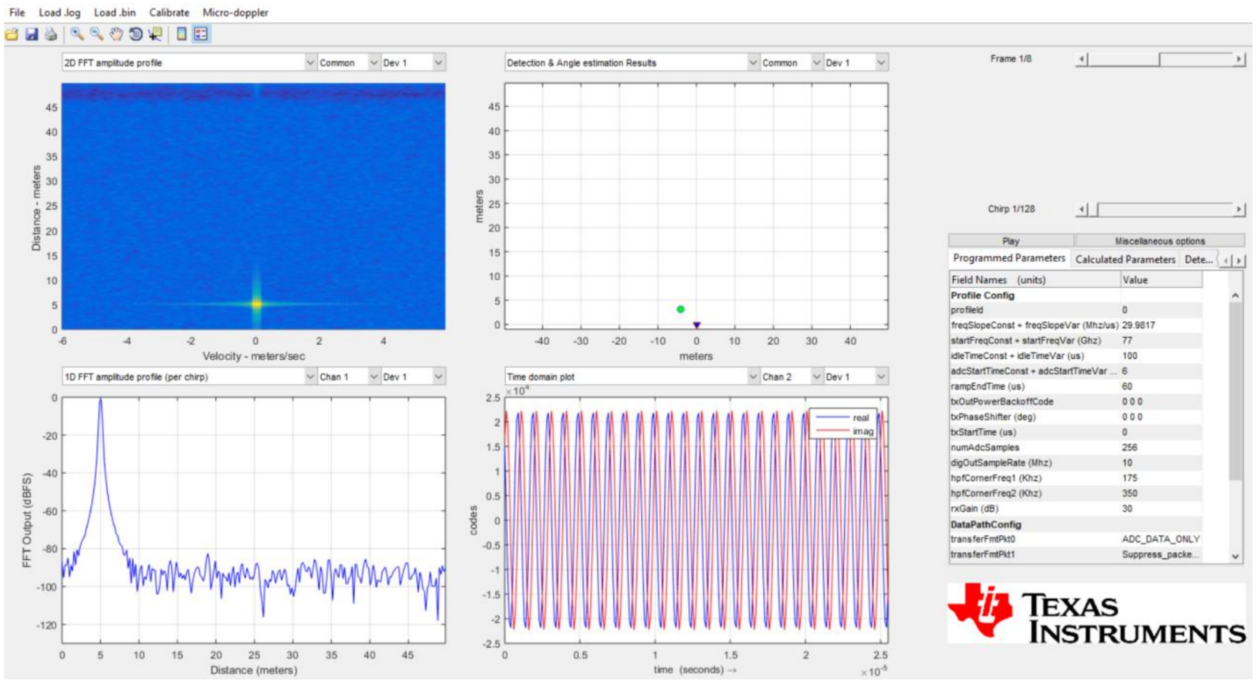
Name	Date modified	Type	Size
adc_data.bin	8/14/2019 12:36 PM	BIN File	4,096 KB

File name: adc_data.bin

All Files (*.*)

Open Cancel

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



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...

9 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
1   addpath(genpath('.\'))
2
3   % Initialize mmWaveStudio .NET connection
4   RSTD_DLL_Path =
    'C:\ti\mmwave_studio_01_00_00_01\mmWaveStudio\Clients\RtttNetClientContr
    oller\RtttNetClientAPI.dll';
5
6   ErrStatus = Init_RSTD_Connection(RSTD_DLL_Path);
7   if (ErrStatus ~= 30000)
8       disp('Error inside Init_RSTD_Connection');
9       return;
10  end
11
12  %Example Lua Command
13  %strFilename =
    'C:\\ti\\mmwave_studio_01_00_00_01\\mmWaveStudio\\Scripts\\Example_scrip
    t_AllDevices.lua';
14  %Lua_String = sprintf('dofile("%s")',strFilename);
15  %ErrStatus =RtttNetClientAPI.RtttNetClient.SendCommand(Lua_String);
```

Init_RSTD_Connection.m

```
    %% Init_RSTD_Connection.m
1   function ErrStatus = Init_RSTD_Connection(RSTD_DLL_Path)
2   %This script establishes the connection with mmWaveStudio software
3   %   Pre-requisites:
4   %   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
```

10 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.

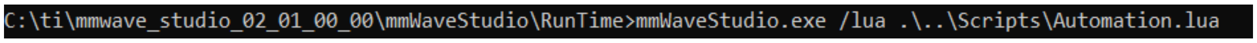
2. Open the command prompt and type

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

10.1 Sample Lua script for automation

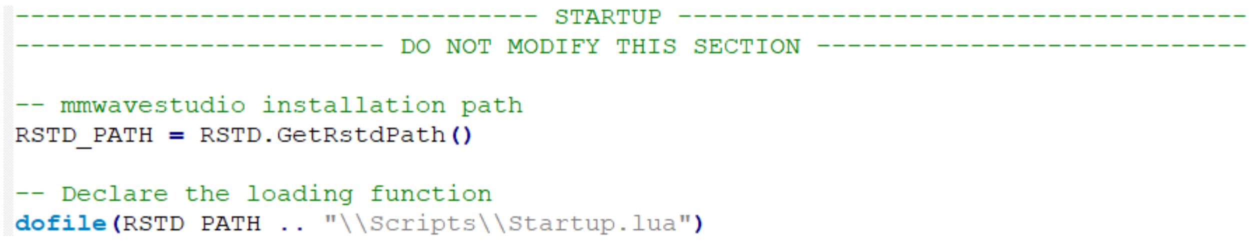
A sample Lua script named "*Automation.lua*" is present at the Scripts directory for reference.

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



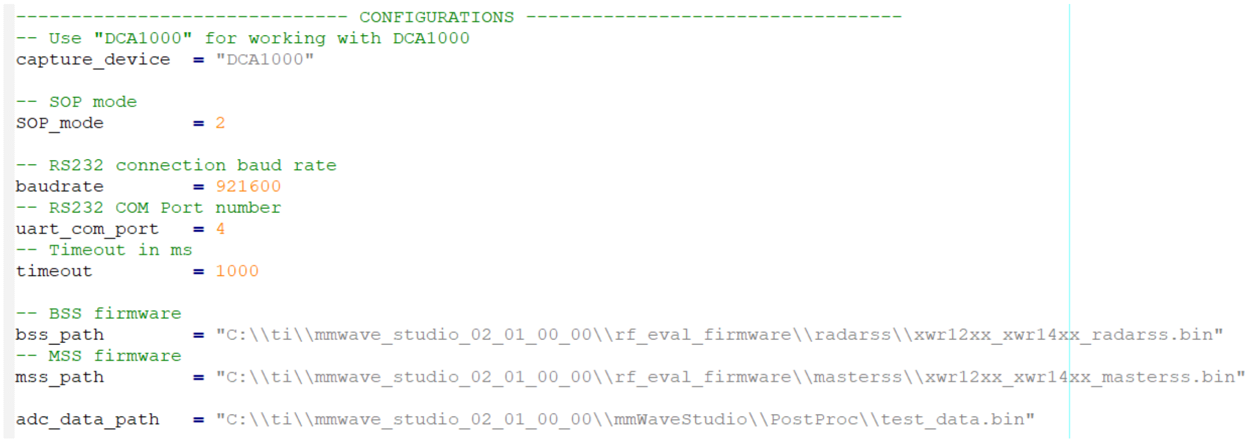
```
C:\ti\mmwave_studio_02_01_00_00\mmWaveStudio\RunTime>mmWaveStudio.exe /lua .\..\Scripts\Automation.lua
```

- 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.

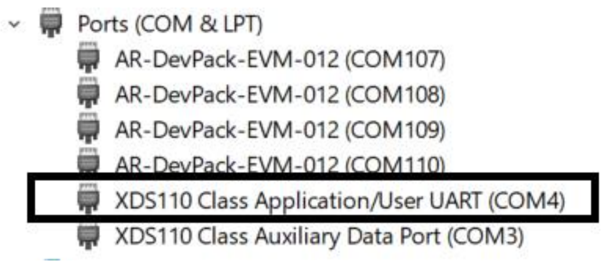


```
------------------------------ CONFIGURATIONS ----------------------------------
-- Use "DCA1000" for working with DCA1000
capture_device  = "DCA1000"

-- 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 "DCA1000".

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.



Ports (COM & LPT)
AR-DevPack-EVM-012 (COM107)
AR-DevPack-EVM-012 (COM108)
AR-DevPack-EVM-012 (COM109)
AR-DevPack-EVM-012 (COM110)
XDS110 Class Application/User UART (COM4)
XDS110 Class Auxiliary Data Port (COM3)

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()
```