

MMWAVE DFP User Guide



Product Release 1.2.6.3

Release Date: June 12, 2020

Contents

Abbreviations	3
1. System Overview	4
1.1 mmWave Firmware	4
1.2 mmWaveLink	4
1.3 mmWave RF Evaluation.....	5
2. AWR1243 - System Deployment	6
3. Getting started	7
4. PC Connection.....	7
4.1 Connection with AWR1243 BOOST.....	7
4.2 Connection with DCA1000-EVM.....	8
5. Demonstration Mode	11
6. Visual Studio Development Mode.....	13
7. Data capture using DCA1000 EVM.....	14
8. Introduction to DFP examples.....	15
8.1 mmwavelink_example	15
8.2 mmwavelink_monitoring.....	15
8.3 mmwavelink_example_nonos	16
9. API Sequence.....	17
10. Additional dependency	18

Abbreviations

Abbreviation	Description
AE	Asynchronous Event
DFP	Device Firmware Package
RadarSS	Radar Sub system
MasterSS	Master Sub system
SPI	Serial Peripheral Interface

1. System Overview

The mmWave device firmware package (DFP) is split in three broad components: mmWave Firmware, mmWaveLink and mmWave RF evaluation.

1.1 mmWave Firmware

mmWave Firmware is responsible for configuring RF/analog, digital front end in TI mmWave radar devices and consists of below components:

- Master SS firmware (Only AWR1243)
- Radar SS firmware (For all variants)

All the services of Master SS and Radar SS firmware are available to external processor using the APIs in mmWaveLink framework.

mmWave Firmware binaries are available at '*mmwave_dfp_<ver>firmware*'

1.2 mmWaveLink

1.2.1 mmWaveLink framework

mmWaveLink framework acts as driver for Master sub-system and Radar sub-system. It exposes a suite of low level APIs which allow application to enable, configure and control Master SS and Radar SS. It provides a well-defined platform and OS abstraction for the application to plug in the communication driver and OS routine callbacks to communicate with the TI mmWave devices. External Processor can use this framework to connect and configure AWR1243 device over SPI.

mmWaveLink source and library are available at '*mmwave_dfp_<ver>ti\control\mmwavelink*'

1.2.2 mmWaveLink examples

A visual studio based sample applications are included in DFP which emulates a host processor on windows PC and runs the mmWaveLink framework. SPI is emulated over USB using the FT4232H device. The platform library which implements the required SPI and OS callbacks is included in the mmWave DFP. Refer to section 7.2 for more details.

These examples are available under '*mmwave_dfp_<ver>ti\example*' path named as mmwavelink_example and mmwavelink_monitoring.

Figure-1 depicts high level modules of external processor and AWR1243 device.

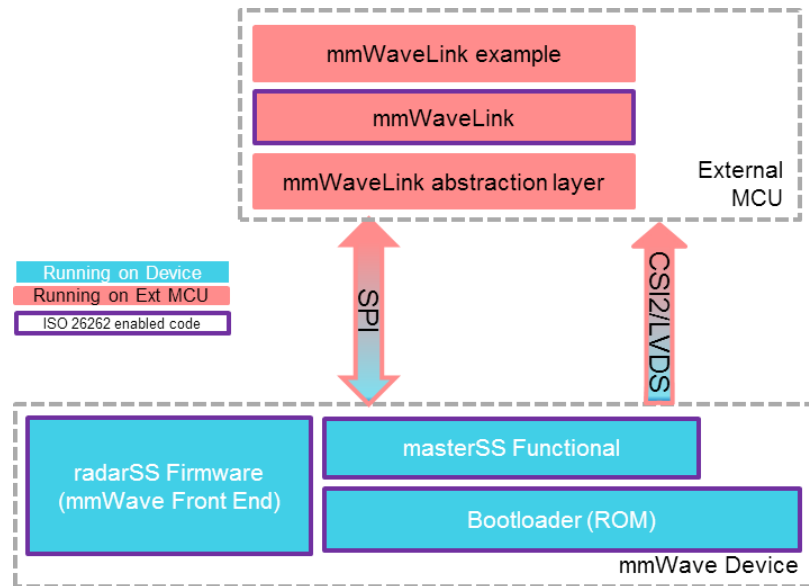


Figure 1. External Processor and AWR1243

1.3 mmWave RF Evaluation

For mmWave RF/System evaluation purpose, mmWave Studio/Radar studio Tool which is designed to communicate with all variants of TI mmWave devices for RF and system performance evaluation can be used; this tool is available at <http://www.ti.com/tool/MMWAVE-STUDIO>.

The RF evaluation firmware is available at '*mmwave_dfp<ver>rf_eval/rf_eval_firmware*' directory. Refer to mmWave Studio user guide for more details.

Figure-2 depicts the modules of mmWave Studio and connection with AWR1243 and TSW1400 EVM Board

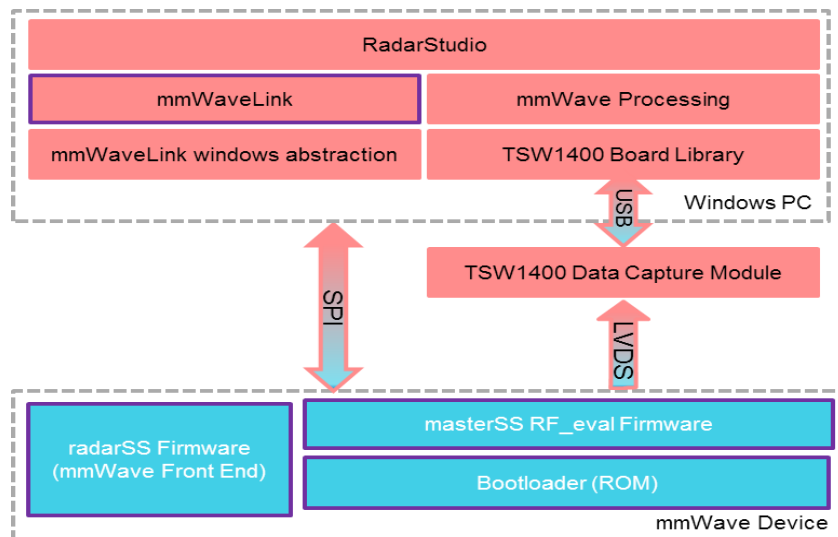
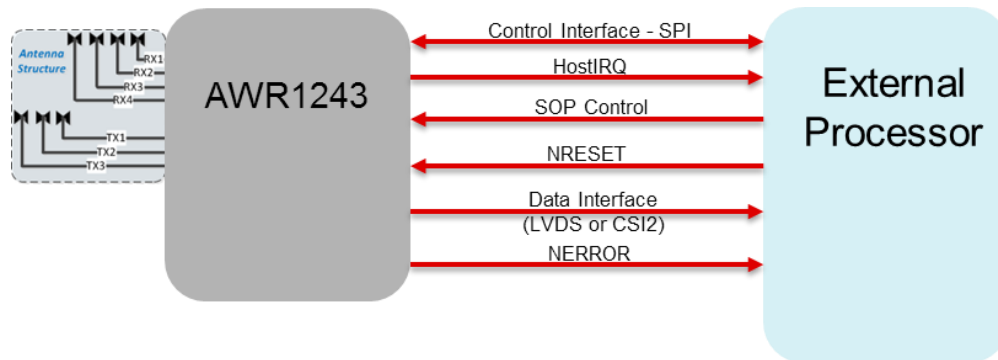


Figure 2. mmWave Studio and AWR1243

2. AWR1243 - System Deployment

User needs to connect an External Processor with AWR1243 device to configure mmWave front end (Radar SS) and process raw data captured over high speed interface. Figure-3 depicts the connection between external processor and AWR1243 device.


Figure 3. Connection between External Processor and AWR1243

Details of AWR1243 pins which needs to be connected to the external processor:

a. Control Interface- SPI

- SPI_CLK_1: Serial Clock by SPI master (external processor).
- SPI_CS_1: SPI chip select (active low), output from SPI master.
- MOSI_1: Master Output Slave Input (data output from master).
- MISO_1: Master Input Slave Output (data output from slave).

Note: External Processor should configure SPI with below configurations

- SPI mode 0 (Phase 0, Polarity 0)
- SPI word size = 16bit
- CS is toggled for each SPI word
- Minimum 2 SPI clock delay between CS active to rising edge of the first SPI clock
- SPI Clock is recommended to be less than 10MHz for SPI communication over FTDI chip of DCA1000 EVM from PC based application (mmwavelink_example application). AWR1243 device supports max 40MHz SPI clock.

- Host IRQ:** SPI_HOST1_INTR (SPI Host interrupt) line, output from AWR1243 to notify Host. It connects to one of the interrupt capable GPIO pin in external processor
- SOP Control:** TDO, SYNC_OUT, PMIC_CLKOUT. Output from external processor to set operating mode of AWR1243 device. For functional mode, It should be set as TDO = 1, SYNC_OUT = 0, PMIC_CLKOUT = 0
- NRESET:** Output from external processor to reset AWR1243 device. To power up AWR1243, NRESET = 1. To power Off, NRESET = 0

- e. **Data Interface:** These lines are used for High speed interface to send raw data over CSI2/LVDS from AWR1243 to external processor.

CSI2_TXP[0]	Differential data Out – Lane 0
CSI2_TXM[0]	
CSI2_CLKP	Differential clock Out
CSI2_CLKM	
CSI2_TXP[1]	Differential data Out – Lane 1
CSI2_TXM[1]	
CSI2_TXP[2]	Differential data Out – Lane 2
CSI2_TXM[2]	
CSI2_TXP[3]	Differential data Out – Lane 3
CSI2_TXM[3]	

- f. **NERROR:** NERROR_OUT line is used by AWR1243 to notify external processor when fatal error occurs in device.

3. Getting started

The best way to get started with the mmWave DFP is to start running mmWaveLink examples that are provided as part of the package. `mmwavelink_example` and `mmwavelink_monitoring` are the two applications available at '`mmwave_dfp_<ver>\tilexample`' directory.

These applications demonstrate firmware download on AWR1243 and configuration of mmWave front end using mmWaveLink framework over SPI interface. It allows user to use difference chirp parameters and configure mmWave device using application executable.

Note1: To receive the CSI2/LVDS, Connect the CSI-2/LVDS Interface of DCA1000 EVM to an external device which has a compatible CSI-2/LVDS. The detail of the external device is out-of-scope for this document.

Note2: Following sections describe the general procedure for connecting the EVM and running the example. Refer to mmWave Studio user guide for steps required to run mmWave Studio GUI.

4. PC Connection

4.1 Connection with AWR1243 BOOST

When the EVM is powered on and connected to Windows PC via the supplied USB cable, there should be two additional COM Ports in Device Manager as shown in Figure-4. See EVM User Guide for details on the COM port.

If below COM ports doesn't show up in the Device Manager install the emupack available [here](#).

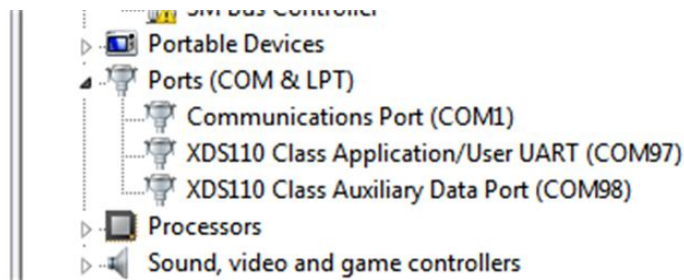


Figure 4. AWR1243-BOOST COM Ports

The USB connection provides below functionalities.

- JTAG for CCS connectivity – Only required for debug purpose. More detail on this is not in the scope of this document.
- UART for downloading firmware in serial data flash using mmWave Studio GUI or UniFlash. Refer to respective user guides for more details.
- UART for downloading Firmware using mmWave Studio GUI. Refer to mmWave Studio user guide for more details.

Note: This step can be skipped if user wants to run mmWaveLink_example only. It is primarily intended for mmWave Studio GUI and debug purpose.

4.2 Connection with DCA1000-EVM

For running mmWaveLink_example and mmWave Studio GUI tool, User also needs to connect the DCA1000 along with the EVM. Stack the DCA1000 with the EVM as shown in Figure-5

For more details on DCA1000 connection, see the [DCA1000 User Guide](#).

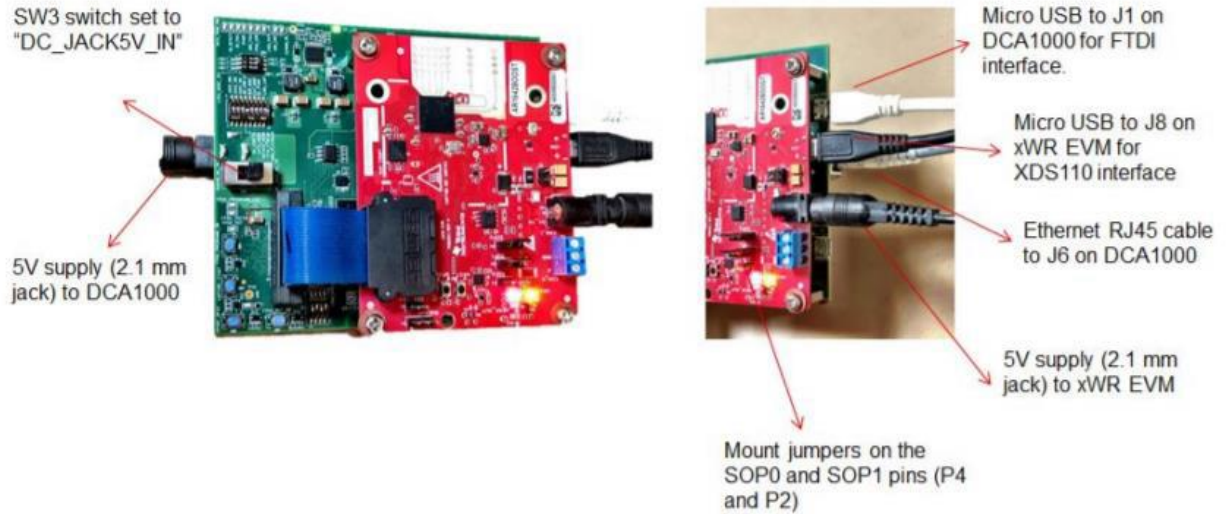


Figure 5. AWR1243 BOOST – DCA1000 Connection

Once stacking is done, 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-9

When the DCA1000 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-6.



Figure 6. 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 (part of [mmWaveStudio](#) installer) as show in Figure-7

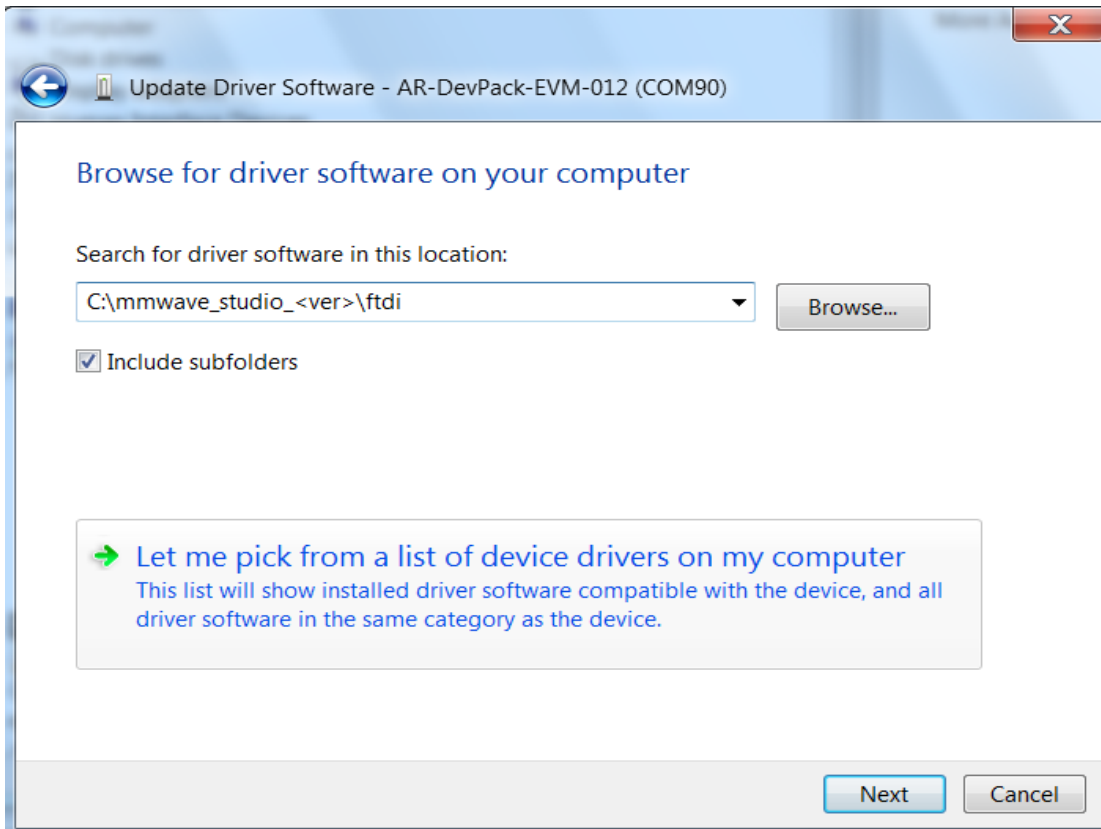


Figure 7. FTDI driver update

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-8, you would need to update the FTDI driver once again.



Figure 8. 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-9.

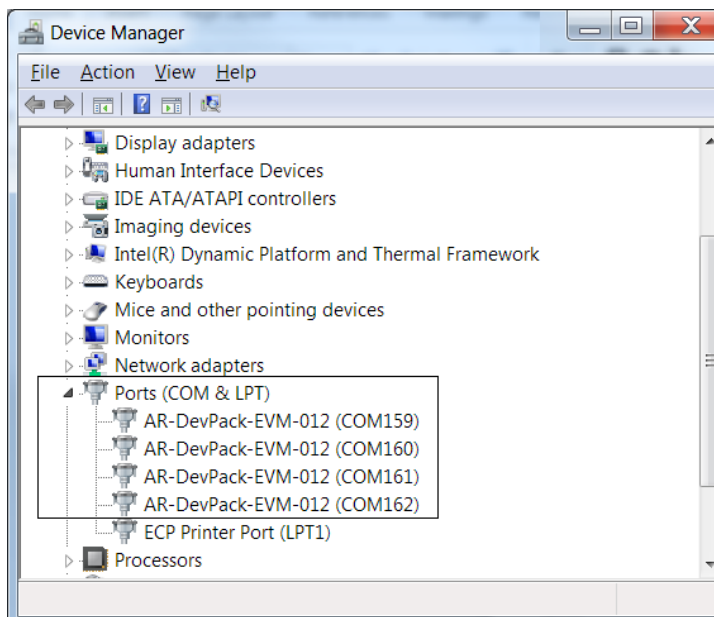


Figure 9. DCA1000 COM Ports

Note: To run mmWaveLink example user must connect AWR1243 EVM and DCA1000 as shown in Figure 5

5. Demonstration Mode

This mode should be used with prebuild executable provided in DFP.

1. Connect AWR1243 EVM and DCA1000 setup to Windows PC as mentioned in section 4.2
2. Open windows command prompt and goto '`mmwave_dfp_<ver>\ti\example\`' path.
3. To run `mmwavelink_example` application goto to '`mmwavelink_example`' directory and run `mmwavelink_example.exe`
4. To run `mmwavelink_monitoring` application goto to '`mmwavelink_monitoring`' directory and run `mmwavelink_monitoring.exe`

Once executed the application reads mmWave radar configuration from `mmwaveconfig.txt` file and sends it to AWR1243 device over SPI. If you prefer to change the configuration, open `mmwaveconfig.txt` in text editor and update the parameters. For more details on these parameters, refer to doxygen HTML files in '`mmwave_dfp_<ver>\ti\mmwavelink\docs`' directory.

Figure 10 and figure 11 show output screen of this `mmwavelink_example` (default setting).

```

===== mmWaveLink Example Application =====
Device map 1 : SOP 4 mode successful

Device map 1 : Device reset successful

rComIfOpen Callback Called for Device Index [0]
rDeviceEnable Callback is called by mmWaveLink for Device Index [0]
mmWave Device Power on success for deviceMap 1

=====Firmware Download=====
Meta Image download started for deviceMap 1
Download in Progress: 0%..10%..20%..30%..40%..50%..60%..70%..80%..90%..Done!

Meta Image download complete ret = 0

Waiting for firmware update response from mmWave device
Firmware update successful for deviceMap 1
=====
Debug: Finished rDeviceSetMiscConfig
CRC Type set for MasterSS success for deviceMap 1

RF Version [ 2. 0. 0. 1]
MSS version [ 1.10. 0.20]
mmWaveLink version [ 1. 2. 6. 1]

RF Patch Version [ 1. 2. 6.11]
MSS Patch version [ 1. 2. 6.12]
Radar/RF subsystem Power up successful for deviceMap 1

```

Figure 10. Output Window 1

```

=====
Calling rSetFrameConfig with
Start Idx[0]
End Idx[127]
Loops[1]
Periodicity[100]ms
Frame Configuration success for deviceMap 1
=====

Async event: Frame trigger
Sensor Start successful for deviceMap 1

=====

Async event: Frame stopped
Sensor Stop successful for deviceMap 1

GPAdc measurement API success for deviceMap 1

rDeviceDisable Callback is called by mmWaveLink for Device Index [0]
Device power off success for deviceMap 1

===== mmWaveLink Example Application execution Successful=====

```

Figure 11. Output Window 2

6. Visual Studio Development Mode

This mode should be used when debugging with Visual Studio is involved and/or developing mmWaveLink application.

1. Install Microsoft Visual Studio 2017 express edition or newer version on windows computer.
2. Connect AWR1243 EVM and DCA1000 setup to Windows PC as mentioned in section 4.2
3. Open Visual Studio IDE and then go for menu option 'File->Open->Project\Solution'.
4. Select 'mmwavelink_example.sln' file from **mmwave_dfp_<ver>ti\example\mmwavelink_example'** directory.
5. It will open mmwavelink_example solution containing two projects mmwavelink and mmwavelink_example as given in below picture.

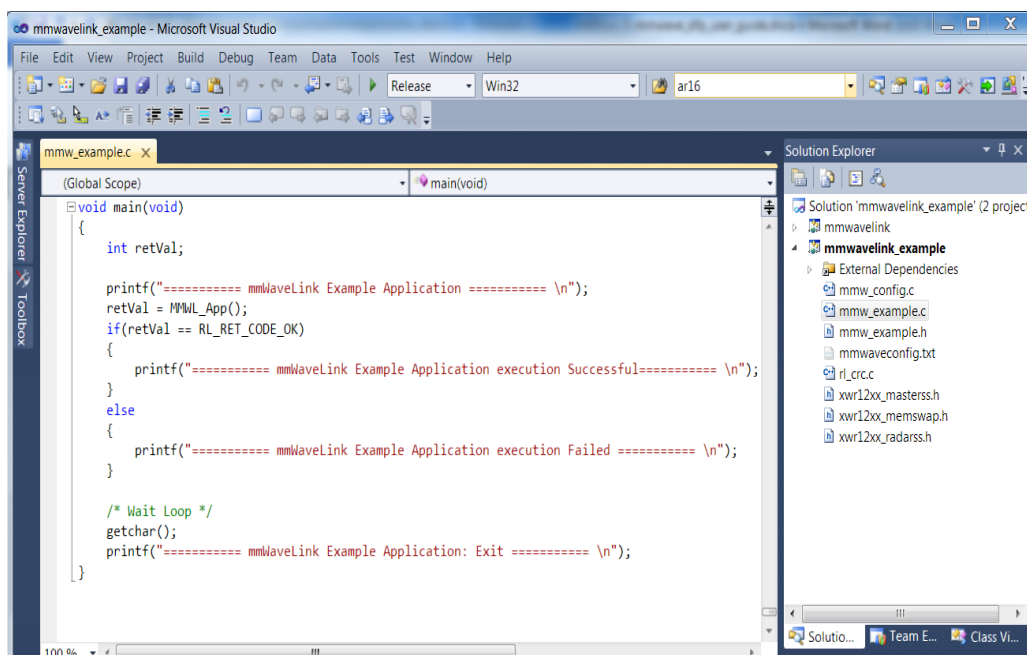


Figure 12. Visual Studio Development mode

6. Click menu option 'Build->Build Solution' which will build both projects.
7. To run the application in debugging mode press 'F5' key.
8. To debug, put breakpoints in the application source code as required.

Note - Please follow the same steps as mentioned above to open and run mmwavelink_monitoring Visual Studio project.

7. Data capture using DCA1000 EVM

As the DFP examples are being executed, we can capture the data from the AWR1243 EVM using the DCA1000 capture card via the DCA1000EVM CLI application.

The DCA1000EVM CLI application is primarily a command line tool for configuration of DCA1000 FPGA and recording based on the user inputs (from a json file). The DCA1000EVM CLI application connects to DCA1000EVM system through Ethernet for configuration and recording of data.

The CLI application (DCA1000EVM_CLI_Control.exe) comes as a part of the mmWaveStudio package (C:\ti\mmwave_studio_02_01_00_00\mmWaveStudio\PostProc). The CLI application makes use of the “cf.json” file present at the PostProc folder. This json file contains information regarding the configuration and capture of AWR1243 EVM. For more information on using the CLI extensively, please refer to the CLI user guide present in the mmWaveStudio package (C:\ti\mmwave_studio_02_01_00_00\mmWaveStudio\ReferenceCode\DCA1000\Docs) folder.

For getting started, the following commands are issued

1. **fpga** (for connection and configuration of DCA1000 with the PC)

```
C:\ti\mmwave_studio_03_00_00_07\mmWaveStudio\PostProc>DCA1000EVM_CLI_Control.exe fpga cf.json
FPGA Configuration command : Success
```

2. **record** (for providing information on packet delay)

```
C:\ti\mmwave_studio_03_00_00_07\mmWaveStudio\PostProc>DCA1000EVM_CLI_Control.exe record cf.json
Configure Record command : Success
```




The commands 1 and 2 can be given even before the execution of the DFP example.

For starting the capture, the command 3 needs to be given during the execution of the DFP example.

3. **start_record** (for starting the data capture. It closes on its own after a timeout of 2 secs when there is no further data being pumped out from the AWR1243)

```
C:\ti\mmwave_studio_03_00_00_07\mmWaveStudio\PostProc>DCA1000EVM_CLI_Control.exe start_record cf.json
Start Record command : Success
```

This command can be given roughly while the firmware is being downloaded in the DFP example application. From the time the command is issued, the DCA1000 will be actively looking for data over the LVDS lanes. Once the AWR1243 has stopped framing, and roughly after a timeout of 2 secs, the DCA1000 CLI application closes and the captured data will be available at the location mentioned in the cf.json file. The following picture describes the raw ADC data being captured, a capture log file indicating the statistics of the capture and a CLI log file showing the commands being executed.

 adc_data_check_Raw_0.bin	2/5/2020 11:45 AM	BIN File	5,120 KB
 adc_data_check_Raw_LogFile.csv	2/5/2020 11:45 AM	Microsoft Excel Co...	1 KB
 CLI_LogFile.txt	2/5/2020 11:45 AM	Text Document	14 KB

8. Introduction to DFP examples

DFP package contains two different examples to demonstrate miscellaneous mmWave sensor features.

8.1 mmwavelink_example

Along with basic mmwave features, this application showcases some of advanced features such as advanced frame configuration, continuous mode, dynamic chirp and dynamic profile configuration. User can enable or disable these advanced features using global TAGs.

```

----- mmw_example.c -----
/* Enable Advanced frame config test in the application
TRUE -> config AdvFrame, FALSE -> config Legacy Frame */
unsigned char gLinkAdvanceFrameTest = FALSE;

/* Enable/Disable continuous mode config test in the application */
unsigned char gLinkContModeTest = FALSE;

/* Enable/Disable Dynamic Chirp config & Dynamic Chirp Enable test in application */
unsigned char gLinkDynChirpTest = TRUE;

/* Enable/Disable Dynamic Profile config test in application */
unsigned char gLinkDynProfileTest = TRUE;

/* TRUE -> Enable LDO bypass. Support only on AWR1243 Rev B EVMs
FALSE -> Disable LDO bypass.
CAUTION : DON'T ENABLE LDO BYPASS ON UNSUPPORTED DEVICES. IT MAY DAMAGE EVM. */
unsigned char gLinkBypassLdo = TRUE;

```

After setting required TAGs to TRUE/FALSE, build the application using visual studio project provided in DFP and run mmwavelink_example.exe.

8.2 mmwavelink_monitoring

This application demonstrates different types of monitoring feature of mmwave sensor device. User can enable or disable legacy/advanced frame configuration based on global TAG.

```

----- mmw_monitoring.c -----
/* Enable Advanced frame config test in the application
TRUE -> config AdvFrame, FALSE -> config Legacy Frame */

```

```
unsigned char gLinkAdvanceFrameTest = FALSE;

/* TRUE -> Enable LDO bypass. Support only on AWR1243 Rev B EVMs
   FALSE -> Disable LDO bypass.
   CAUTION : DON'T ENABLE LDO BYPASS ON UNSUPPORTED DEVICES. IT MAY DAMAGE EVM. */
unsigned char gLinkBypassLdo = TRUE;
```

All monitoring reports are received as asynchronous event from the device and handled under MMWL_asyncEventHandler function.

NOTE – Solely purpose of these examples provided in DFP package is to showcase the usage of different mmWaveLink APIs which will further help user to port it on any external processor.

8.3 mmwavelink_example_nonos

This application shows to run mmwavelink on non-os platforms like embedded hosts, where os routines are not supported. User can enable or disable these advanced features using global TAGs.

```
----- mmw_example_nonos.c -----
/* Enable Advanced frame config test in the application
TRUE -> config AdvFrame
FALSE -> config Legacy Frame */
unsigned char gLinkAdvanceFrameTest = FALSE;

/* Enable/Disable continuous mode config test in the application */
unsigned char gLinkContModeTest = FALSE;

/* TRUE -> Enable LDO bypass. Support only on AWR1243 Rev B EVMs
   FALSE -> Disable LDO bypass.
   CAUTION : DON'T ENABLE LDO BYPASS ON UNSUPPORTED DEVICES. IT MAY DAMAGE EVM. */
unsigned char gLinkBypassLdo = TRUE;
```

After setting required TAGs to TRUE/FALSE, build the application using visual studio project provided in DFP and run mmwavelink_example_nonos.exe.

9. API Sequence

Below Figure shows the API sequence in mmwavelink_example (default setting). For more details on these APIs, refer to mmWaveLink framework doxygen HTML files in '*mmwave_dfp_<ver>ti\mmwavelink\docs*' directory.

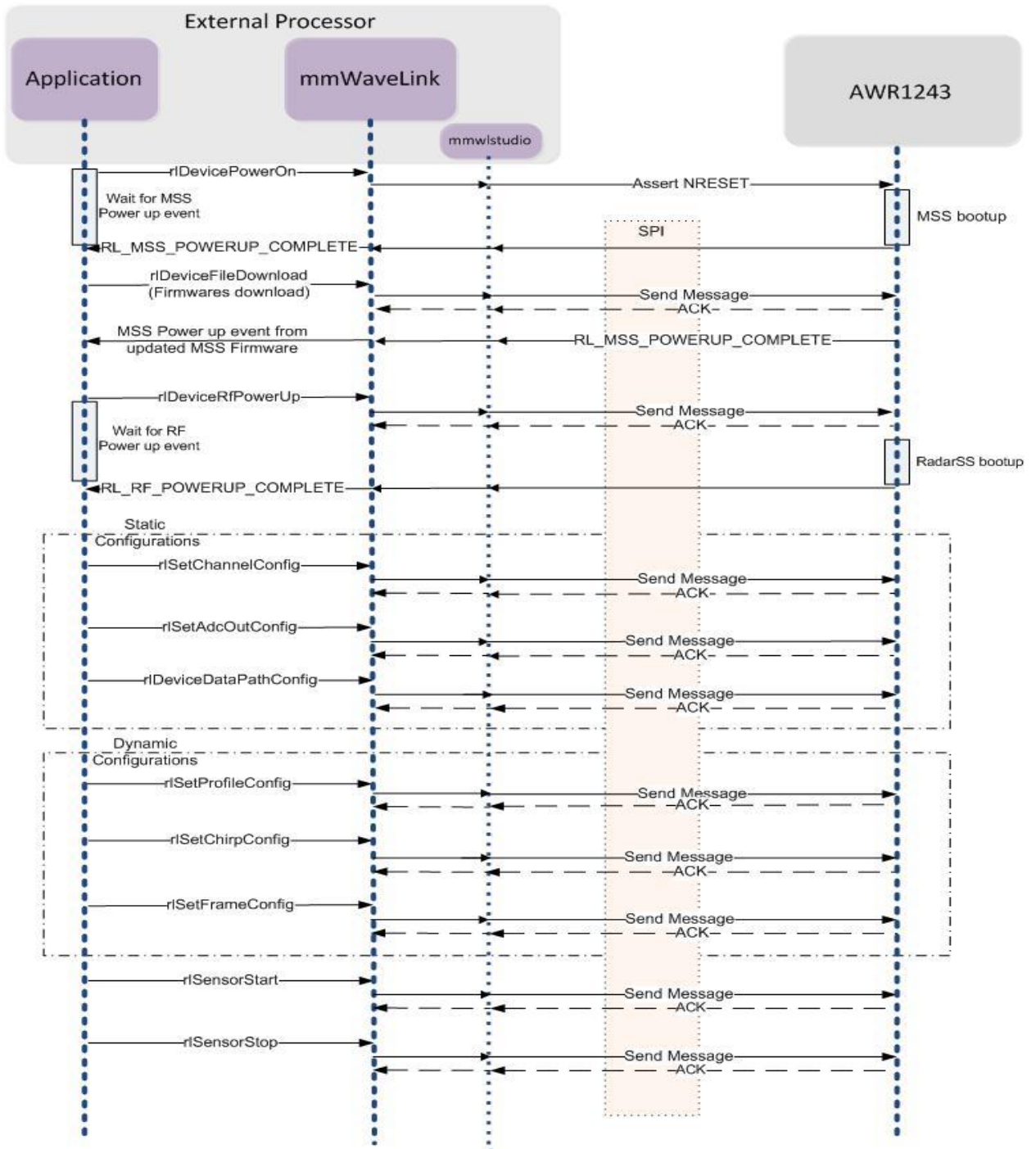


Figure 13. API Sequence

10. Additional dependency

This mmWaveLink example application on windows PC uses below library to communicate with AWR1243.

mmwlstudio.lib: This library provides implementation of all the platform dependent routines required by TI mmWaveLink framework. Broadly it implements below platform interfaces as mentioned in mmWaveLink porting guide.

- Communication Interface
- Device Control Interface
- OS Interface

This library is available at '*mmwave_dfp_<ver>\ti\example\platform\mmwlstudio*' directory.

To port mmWaveLink framework to a new processor, developer need to provide similar interfaces as implemented on a windows platform using mmwkstudio library.

For more information on porting steps, refer to mmWaveLink framework doxygen HTML files in '*mmwave_dfp_<ver>\ti\mmwavelink\docs*' directory