Processor SDK – Radar

(v03.06)

User Guide

IMPORTANT NOTICE

TABLE OF CONTENTS

1 Introduction

Processor SDK Radar is a multi-processor software development package for TI's family of ADAS SoCs. The software framework allows users to create different Radar application data flows involving integration of FMCW transceiver, radar signal processing, and visualization on a display device. The framework has sample Radar processing data flows which exercises different CPUs and HW accelerators in the ADAS SoC and demonstrates how to effectively use different sub-systems within the SoC. Frame work is generic enough to plug in application specific algorithms in the system.

This document explains the HW/SW setup for TDA based Radar Platform.

1.1 References

Refer the below additional documents for more information about Processor SDK Radar

Document	Description
ProcessorSDKRadar_ReleaseNotes.pdf	Release specific information
VisionSDK_UserGuide_TDA3xx.pdf	This document contains install, build, execution information for TDA3xx ADAS SoC
VisionSDK_UserGuide_TDA2xx.pdf	This document contains install, build and execution information on the TDA2xx ADAS SoC.
VisionSDK_ApiGuide.CHM	User API interface details
VisionSDK_SW_Architecture.pdf	Overview of software architecture
VisionSDK_DevelopmentGuide.pdf	Details how to create data flow (s) & add new functionality
ProcessorSDKRadar_DevelopmentGuide.pdf	Details how to create data flow (s) & add new functionality specific to Radar data processing.
ProcessorSDKRadar_Datasheet.pdf	Performance and benchmark information for Radar Usecases.

1.2 Directory Structure

Once Processor SDK Radar is installed, two main directories are created namely vision_sdk and ti_components.

Directory	**Description**
vision_sdk	Root directory of Processor SDK Radar
vision_sdk/build	Support files for building entire release package
vision_sdk/build/rtos/makerules	Make rules for components and cores for RTOS applications
vision_sdk/build/rtos/tda3xx	Rules specifically for TDA3xx device
vision_sdk/build/rtos/tda2xx	Rules specifically for TDA3xx device
vision_sdk/apps/configs	Build configuration option folders and configuration files
vision_sdk/docs	Documentation for Processor SDK Radar. Use the Index.html file to navigate through all the documentation.
vision_sdk/sample_app	Sample Use case to get started
vision_sdk/apps	Applications and Usecase Examples, Use case for TDA2xxx and TDA3xx device
vision_sdk/apps/include	Include files needed for use case support infrastructure
vision_sdk/apps/src	Source files needed for use case support infrastructure
vision_sdk/apps/src/rtos/radar/include	**All include files related to radar framework**
vision_sdk/apps/src/rtos/radar/src	**All the source files for the radar framework**
vision_sdk/apps/src/rtos/radar/src/alg_plugins	**All the source files for the radar algorithm plugin**
vision_sdk/apps/src/rtos/radar/src/alg_plugins/alg_fxns	**All the source files for the radar algorithm functions**
vision_sdk/apps/src/rtos/radar/src/usecase	**Usecase folders for different Radar Data Processing flows.**
vision_sdk/links_fw/include	All the include files for the framework
vision_sdk/links_fw/include/link_api	Interface files for all the links
vision_sdk/links_fw/src	All the source files for the framework
vision_sdk/links_fw/src/links_common	Files which are for common across all links
vision_sdk/links_fw/src/links_dsp	Source files for individual links present on DSP
vision_sdk/links_fw/src/links_eve	Source files for individual links present on EVE
vision_sdk/links_fw/src/links_ipu	Source files for individual links present on IPU
vision_sdk/links_fw/src/main_app	Folder for main() functionality
vision_sdk/links_fw/src/utils_common	Common utilities used in framework
ti_components	Root directory of tools accompanying vision SDK
ti_components/algorithms/eve_sw_xx_xx_xx_xx	EVE Kernels Library
ti_components/cg_tools	All the code gen tools
ti_components/cg_tools/windows/arm_x_x_x	Tools needed for ARM CPU cores
ti_components/cg_tools/windows/arp32_x_x_x	Tools needed for ARP32 core

ti_components/cg_tools/windows/c6000_x.x.x	Tools needed for C66x DSP
ti_components/drivers	All the drivers used in Vision SDK
ti_components/drivers/pdk_xx_xx_xx_xx	Driver components for all the peripherals along with Lowest level SW interface for programming HW registers
ti_components/drivers/edma3_lld_xx_xx_xx_xx	Driver for system DMA usage
ti_components/networking	Networking related tools
ti_components/networking/ndk_x_xx_xx_xx	Network Development Kit
ti_components/networking/nsp_vayu_x_xx_xx_xx	Network Development Kit Support Package
ti_components/radar/mmwave_dfp_xx_xx_xx_xx	**mmWave device firmware package**
ti_components/os_tools	Operating System related tools
ti_components/os_tools/bios_x_xx_xx_xx	BIOS operating sytem used in Vision SDK
ti_components/os_tools/windows/xdctools_x_xx_xx_xx	XDC tools related files

2 System Requirements

This chapter provides a brief description of the system requirements (hardware and software) and instructions for installing Processor SDK Radar.

2.1 Windows Installation

2.1.1 PC Requirements

Installation of this release needs a windows machine with about 8GB of free disk space. Building of the SDK is supported on windows environment.

2.1.2 Software Requirements

All software packages required to build and run the Processor SDK Radar are included as part of the SDK release package.

2.1.3 A15 Compiler, Linker

The windows installer for the linaro tools should be downloaded from below link

https://launchpad.net/gcc-arm-embedded/+milestone/4.9-2015-q3-update

The tools need to be installed in "<install dir>/ti_components/cg_tools/windows/gcc-arm-none-eabi-4_9-2015q3" location.

IMPORTANT NOTE: A15 Compiler and linker MUST be installed before proceeding else compile will fail. Also make sure the compiler is installed at the exact path mentioned above

2.1.4 Code Composer Studio

CCS is needed to load, run and debug the software. CCS can be downloaded from the below link. CCS version CCS version 6.0.1.00040 or higher should be installed.

http://processors.wiki.ti.com/index.php/Download_CCS

2.2 Linux Installation

2.2.1 PC Requirements

Installation of this release needs a Linux Ubuntu 14.04 machine.

IMPORTANT NOTE: If you are installing Ubuntu on a virtual machine, ensure it is a 64 bit Ubuntu.

2.2.2 Software Requirements

All software packages required to build and run the Vision SDK are included as part of the SDK release package except for the ones mentioned below

2.2.2.1 A15 Compiler, Linker

The Linux installer should be downloaded from below link

https://launchpad.net/gcc-arm-embedded/+milestone/4.9-2015-q3-update

The tools need to be installed in $INSTALL_DIR/ti_components/os_tools/linux/ location.

IMPORTANT NOTE: A15 Compiler and linker MUST be installed before initiating the build else compilation will fail. Also make sure the compiler is installed at the exact path mentioned above after installation of vision sdk.

Use following steps to install the toolchain

2.2.3 Other software packages for build depending upon OS baseline

Ensure these packages/tools are installed on the installation machine

uname, sed, mkimage, dos2unix, dtrx, mono-complete, git, lib32z1 lib32ncurses5 lib32bz2-1.0 libc6:i386 libc6-i386 libstdc++6:i386 libncurses5:i386 libz1:i386 libc6-dev-i386 device-tree-compiler mono-complete

To install

```
$>sudo apt-get install <package_name>
```

2.3 Hardware Requirements

Hardware setup for different use-cases is described in this section

The typical connection between the TDA3xx and the AWR12xx is as shown below:

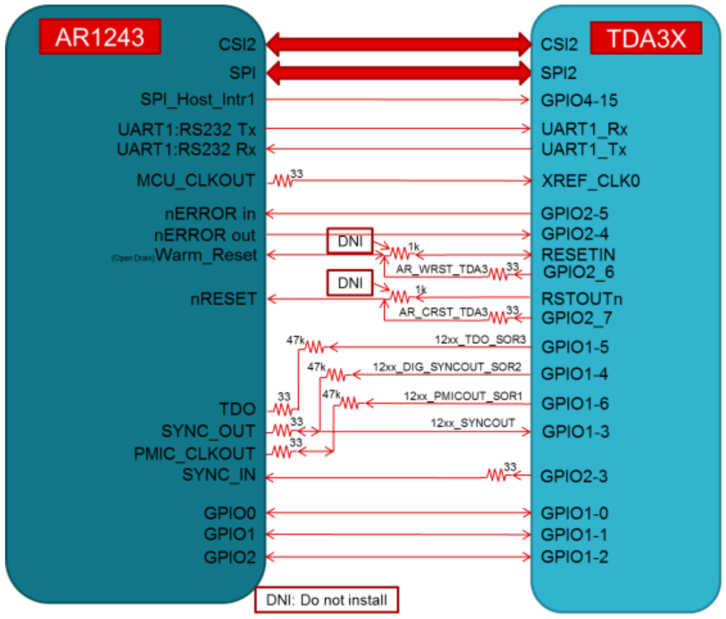


Figure 2.1 AWR12 + TDA3xx Connection

2.3.1 AWR1243 Satellite Radar Module + RVP-TDA3x via FPD-Link III

Satellite Radar use case is developed with the evaluation platform from D3 Engineering. The full platform includes the following components:

- **AWR1243 Satellite Radar Module (with Serializer UB953)**

AWR1243 Module

With Enclosure

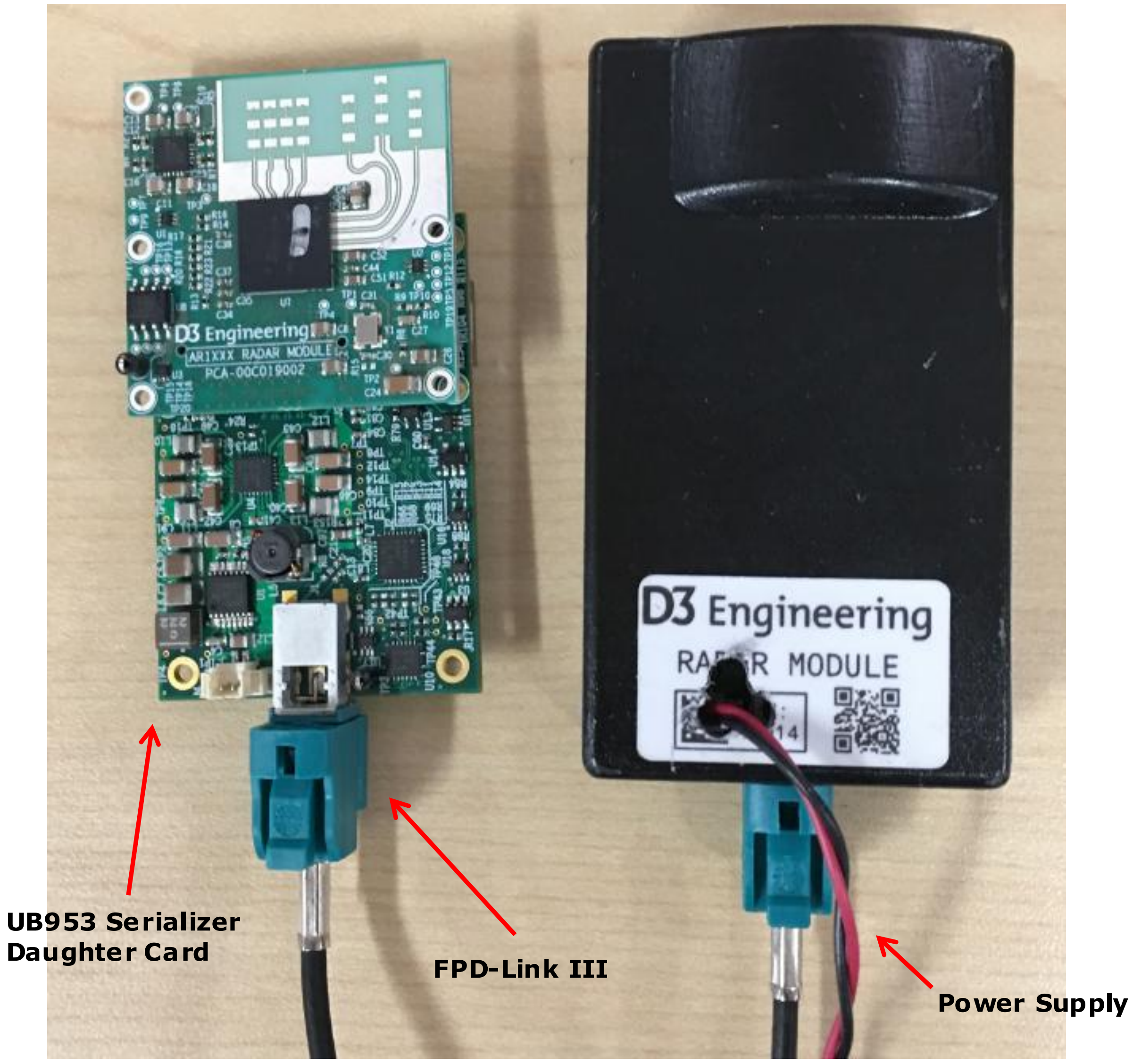


Figure 2.2 AWR1243 Satellite Radar Module

- **RVP-TDA3x (with Deserializer Hub UB960)**

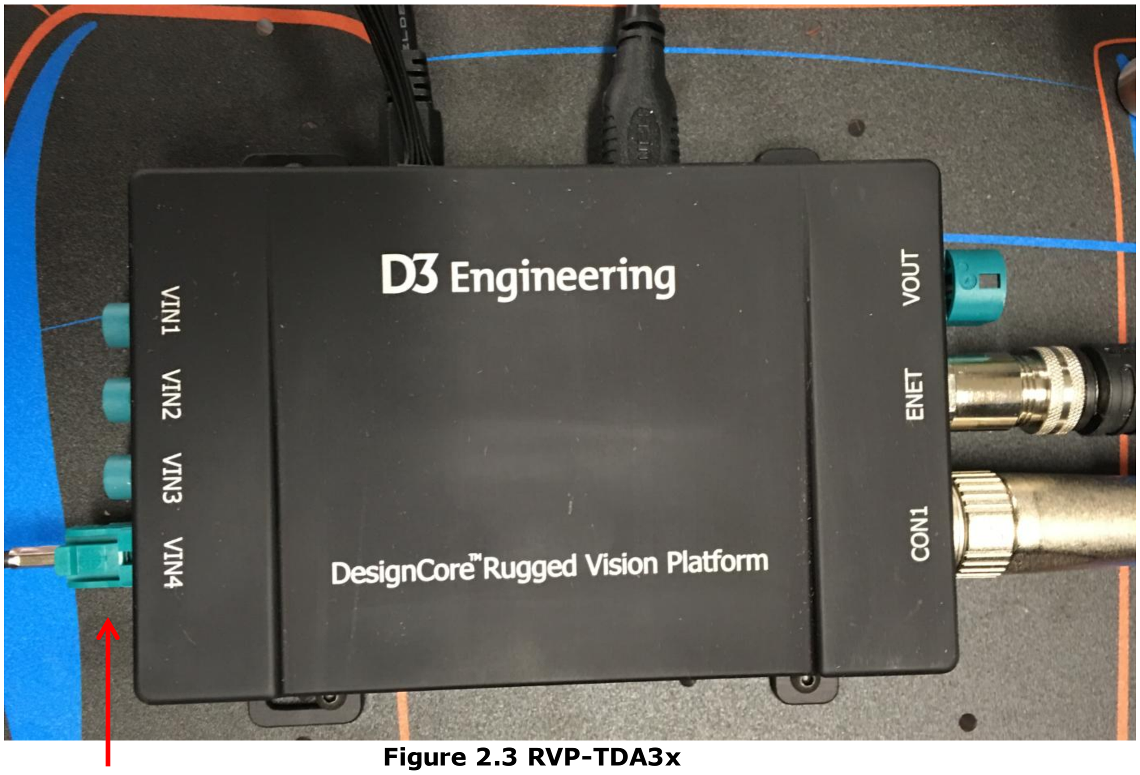


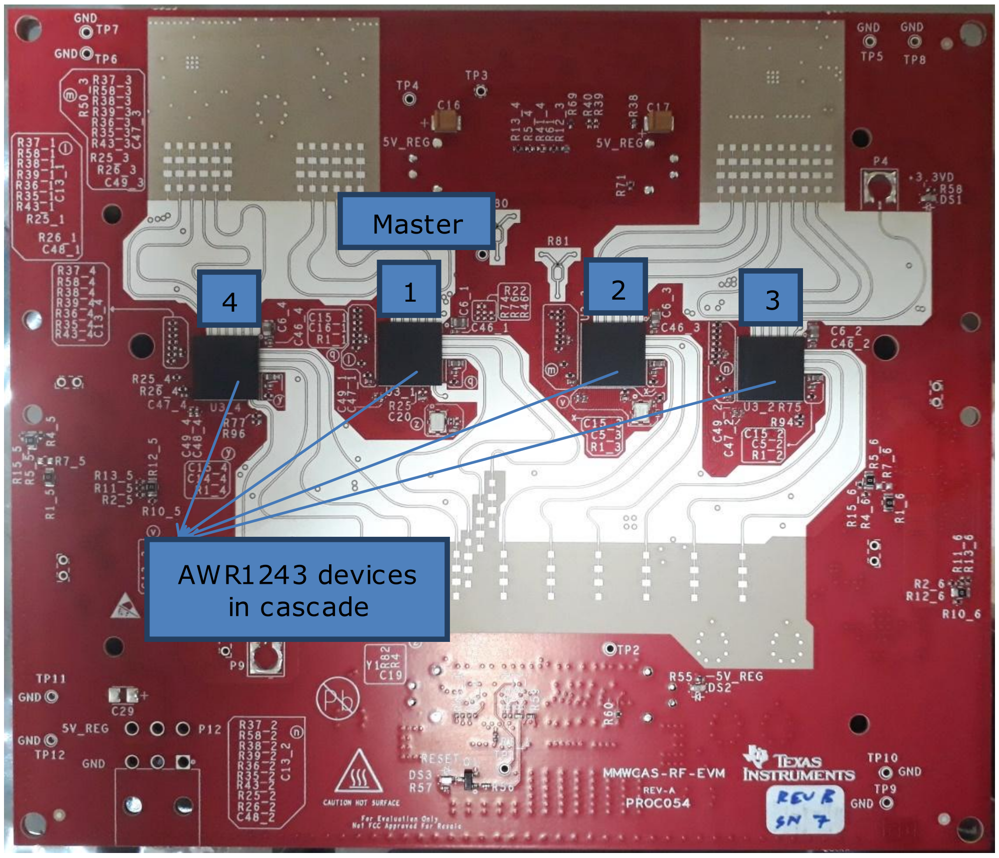
Figure 2.3 RVP-TDA3x

Satellite Radar Input

NOTE: Satellite Radar module should be connected to VIN4 on RVP-TDA3x as shown in Fig 2.9.

2.3.2 AWR1243 Cascade Radar Board and TDA2xx Processing board.

The AWR1243 cascade radar board and the TDA2xx board are as shown below:



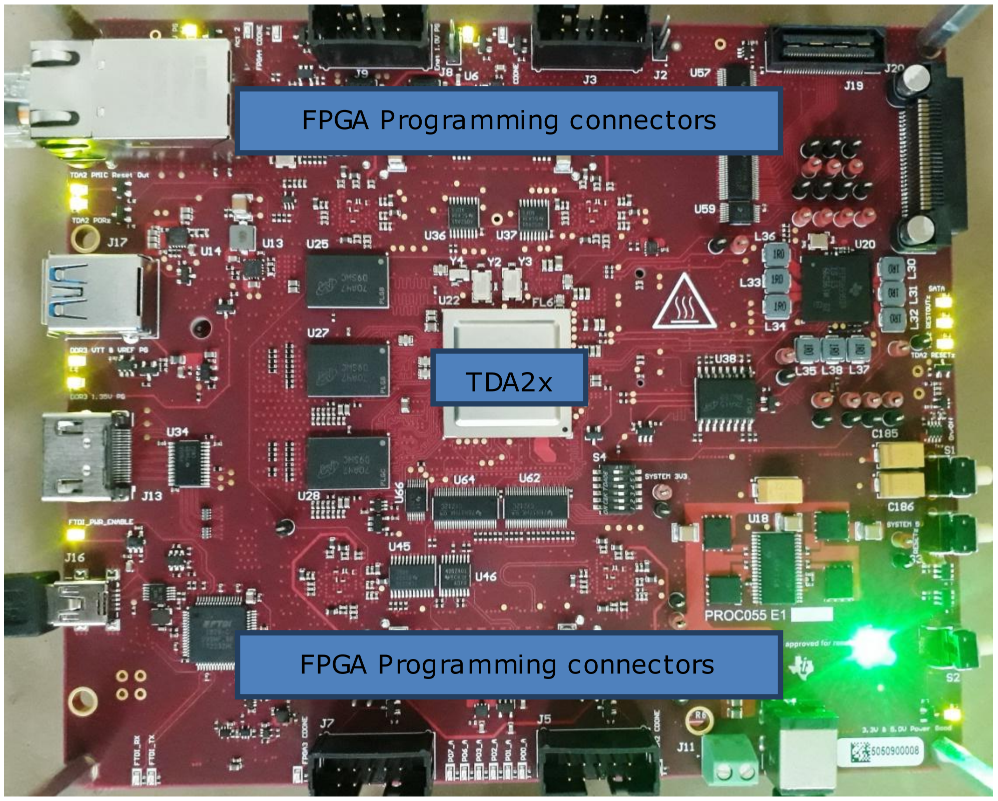
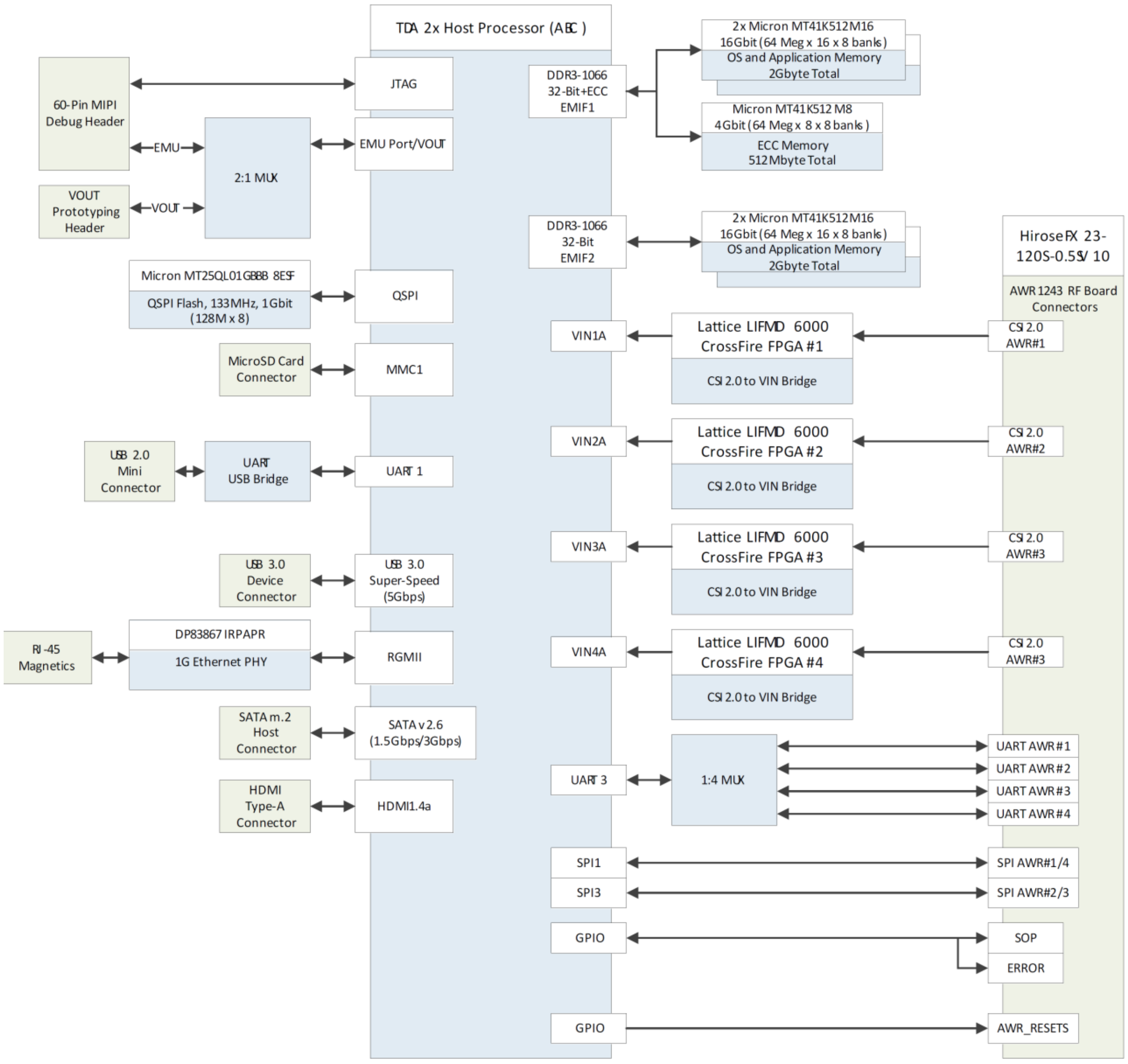
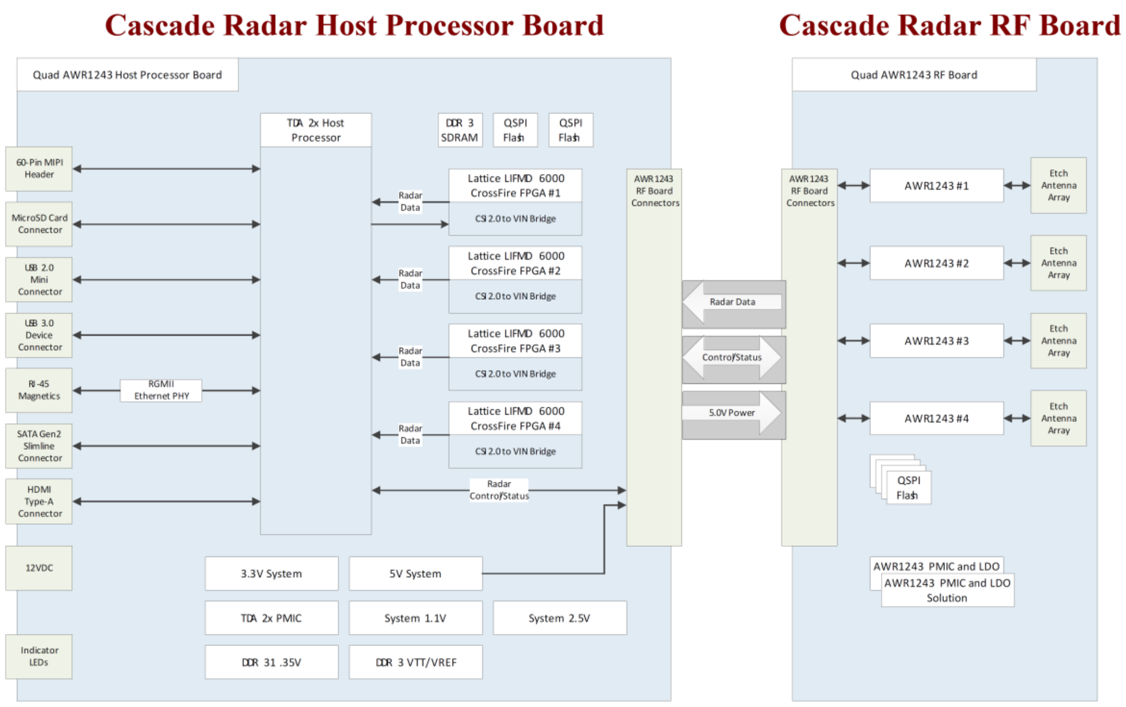


Figure 2.10 TDA2xx Cascade Radar Antenna and CPU board.

The high level block diagram of the board is as below:

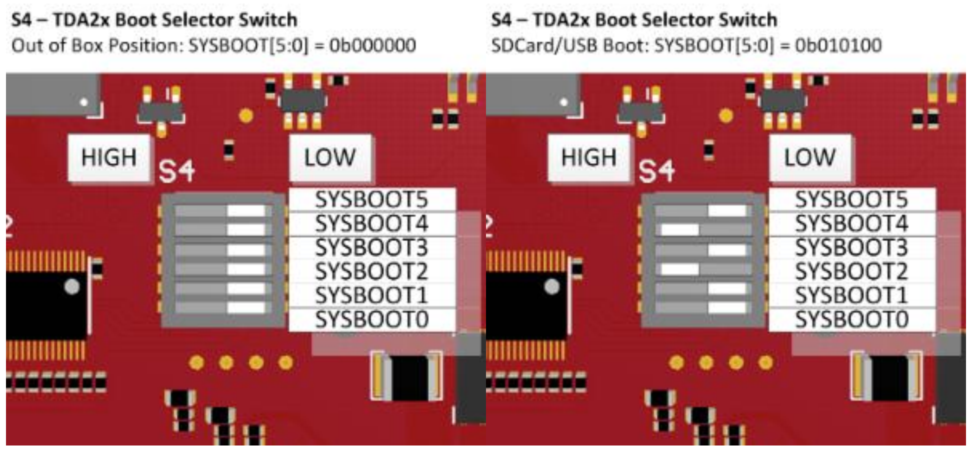

TDA 2x Host Processor (ABC)
60-Pin MIPI Debug Header
JTAG
EMU
2:1 MUX
EMU Port/VOUT
VOUT Prototyping Header
VOUT
Micron MT25QL01GBBB 8ESF
QSPI Flash, 133MHz, 1Gbit (128M x 8)
QSPI
MicroSD Card Connector
MMC1
USB 2.0 Mini Connector
UART USB Bridge
UART 1
USB 3.0 Device Connector
USB 3.0 Super-Speed (5Gbps)
RJ-45 Magnetics
DP83867 IRPAPR
1G Ethernet PHY
RGMII
SATA m.2 Host Connector
SATA v2.6 (1.5Gbps/3Gbps)
HDMI Type-A Connector
HDMI1.4a
DDR3-1066 32-Bit+ECC EMIF1
2x Micron MT41K512M16 16Gbit (64 Meg x 16 x 8 banks)
OS and Application Memory 2Gbyte Total
Micron MT41K512M8 4Gbit (64 Meg x 8 x 8 banks)
ECC Memory 512Mbyte Total
DDR3-1066 32-Bit EMIF2
2x Micron MT41K512M16 16Gbit (64 Meg x 16 x 8 banks)
OS and Application Memory 2Gbyte Total
Hirose FX 23-120S-0.5SV 10
AWR 1243 RF Board Connectors
VIN1A
Lattice LIFMD 6000 CrossFire FPGA #1
CSI 2.0 to VIN Bridge
CSI 2.0 AWR#1
VIN2A
Lattice LIFMD 6000 CrossFire FPGA #2
CSI 2.0 to VIN Bridge
CSI 2.0 AWR#2
VIN3A
Lattice LIFMD 6000 CrossFire FPGA #3
CSI 2.0 to VIN Bridge
CSI 2.0 AWR#3
VIN4A
Lattice LIFMD 6000 CrossFire FPGA #4
CSI 2.0 to VIN Bridge
CSI 2.0 AWR#3
UART 3
1:4 MUX
UART AWR#1
UART AWR#2
UART AWR#3
UART AWR#4
SPI1
SPI3
SPI AWR#1/4
SPI AWR#2/3
GPIO
SOP
ERROR
GPIO
AWR_RESETS

The connection between the AWR1243 board and the TDA2x board is as below:



2.3.2.1 *TDA2x Boot Selector Switch Settings*

Settings for the S4 switch to select the TDA2x boot mode.

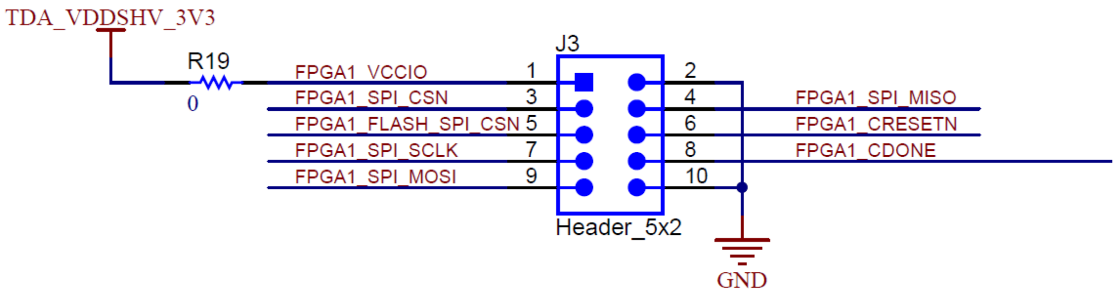

S4 – TDA2x Boot Selector Switch
Out of Box Position: SYSBOOT[5:0] = 0b000000
S4 – TDA2x Boot Selector Switch
SDCard/USB Boot: SYSBOOT[5:0] = 0b010100
HIGH
S4
LOW
SYSBOOT5
SYSBOOT4
SYSBOOT3
SYSBOOT2
SYSBOOT1
SYSBOOT0

2.3.2.2 *Lattice FPGA Flashing Steps*

The board has 4 connector ports which can be used to flash the Lattice LIFMD 6000 FPGA to program the SPI Flash which the FPGA can then load to convert CSI2 to Parallel. (J3, J9, J5, J7)

The steps to flash the FPGA SPI flash is as below. Note this is a one-time setup.

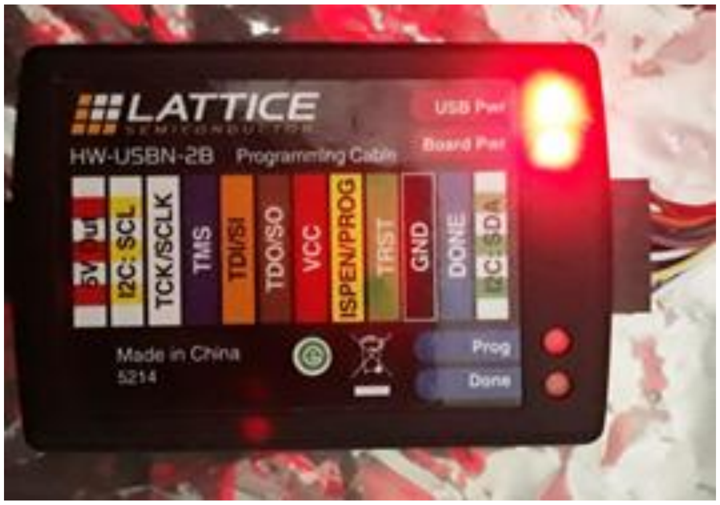
A Lattice USB2B Connector which looks like below should be connected to J3, J9, J5 and J7 in turns. All the 4 FPGA SPI flashes should be programmed.

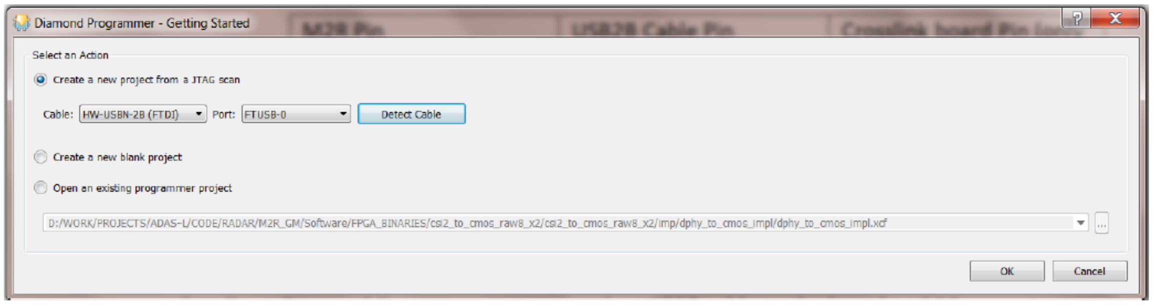


The connections are:

Connector Pin	**USB2B Cable Pin**
FPGA_CRESETn	TRST
FPGA1_MOSI	TDI
FPGA1_MISO	TDO
FPGA1_SCLK	TCK
FPGA1_CSn/FLASH1_CSn	ispEN
FPGA_VCCIO	VCC
GND	GND

1. Download Diamond Programmer: http://www.latticesemi.com/view_document?document_id=52058
2. Open Diamond Programmer; connect the USB2B cable to the board and PC, power on the board.

3. 

-> LED Status with USB and Board power
4. Click "Detect Cable" to find the cable as below:
5. 
Diamond Programmer - Getting Started

Select an Action

(●) Create a new project from a JTAG scan

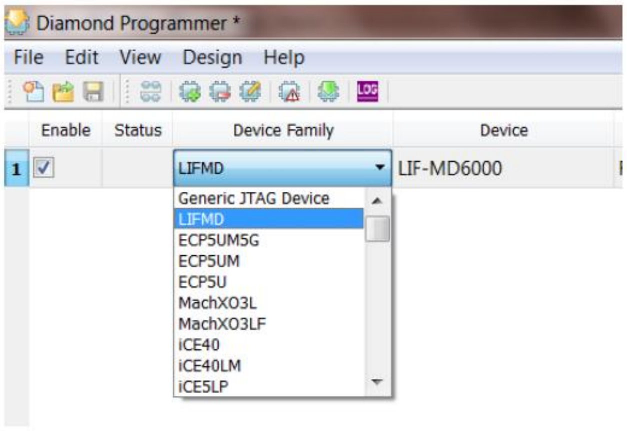
Cable: HW-USBN-2B (FTDI) Port: FTUSB-0 Detect Cable

( ) Create a new blank project

( ) Open an existing programmer project

D:/WORK/PROJECTS/ADAS-L/CODE/RADAR/M2R_GM/Software/FPGA_BINARIES/csi2_to_cmos_raw8_x2/csi2_to_cmos_raw8_x2/imp/dphy_to_cmos_impl/dphy_to_cmos_impl.xcf

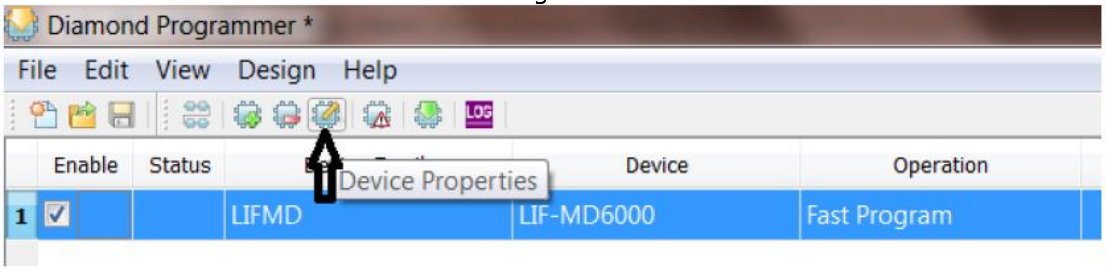
OK Cancel

6. Click "OK" to continue.

Diamond Programmer *

File Edit View Design Help

	Enable	Status	Device Family	Device
1	✓		LIFMD	LIF-MD6000

Generic JTAG Device
LIFMD
ECP5UM5G
ECP5UM
ECP5U
MachXO3L
MachXO3LF
iCE40
iCE40LM
iCE5LP

7. Select the device after the Diamond Programmer has launched:

Diamond Programmer *

File Edit View Design Help

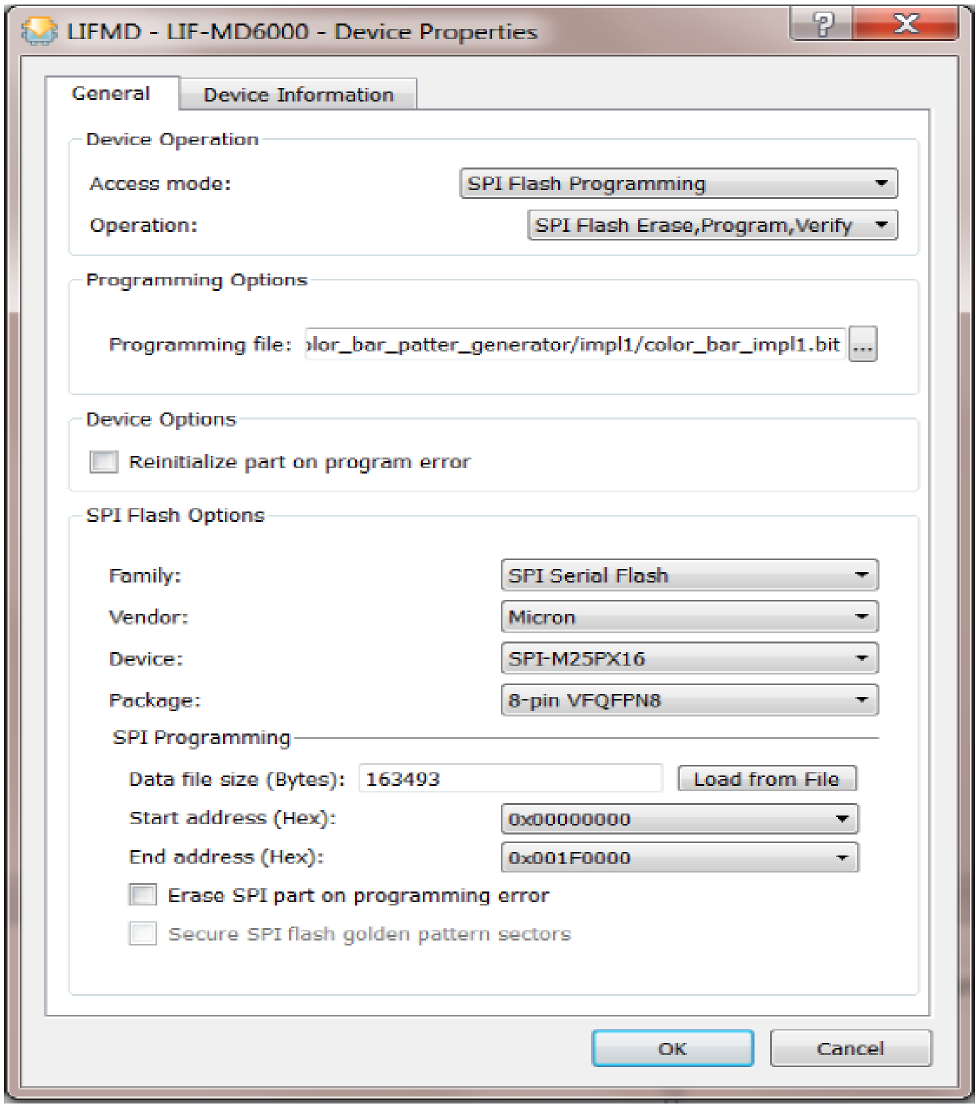
Device Properties

	Enable	Status	Device Family	Device	Operation
1	✓		LIFMD	LIF-MD6000	Fast Program

8. Choose the row and click on "Device Properties" Button.
9. Select the following properties and click "OK".

10. The FPGA image is present in (Note: The same binary is to be flashed to all the 4 FPGAs)

```
pdk/packages/ti/drv/vps/unit_test/cascadeRadarUt/fpgaimage/dphy_to_cmos
_raw8_2_dphy_to_cmos_impl.bit
```



LIFMD - LIF-MD6000 - Device Properties

General | Device Information

Device Operation

Access mode: SPI Flash Programming

Operation: SPI Flash Erase,Program,Verify

Programming Options

Programming file: olor_bar_patter_generator/impl1/color_bar_impl1.bit

Device Options

☐ Reinitialize part on program error

SPI Flash Options

Family: SPI Serial Flash

Vendor: Micron

Device: SPI-M25PX16

Package: 8-pin VFQFPN8

SPI Programming

Data file size (Bytes): 163493 | Load from File

Start address (Hex): 0x00000000

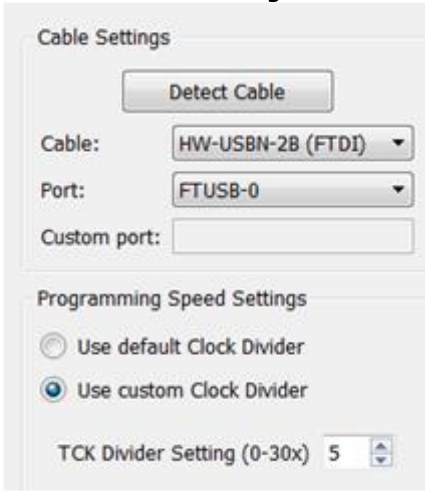
End address (Hex): 0x001F0000

☐ Erase SPI part on programming error

☐ Secure SPI flash golden pattern sectors

OK | Cancel

11. On the Cable Settings set the Clock Divider to "5" as below:



Cable Settings

Detect Cable

Cable: HW-USBN-2B (FTDI)

Port: FTUSB-0

Custom port:

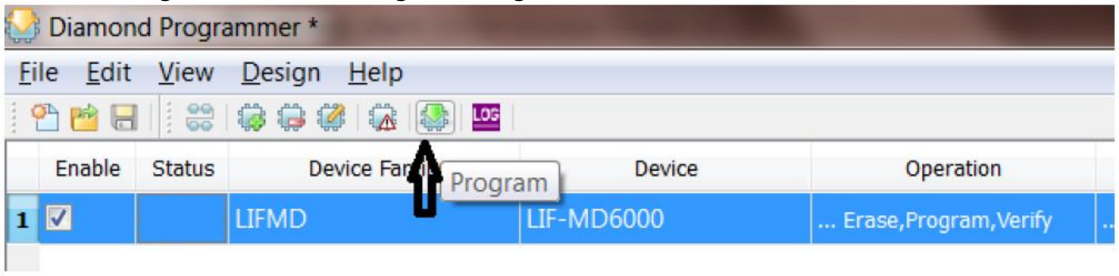
Programming Speed Settings

Use default Clock Divider

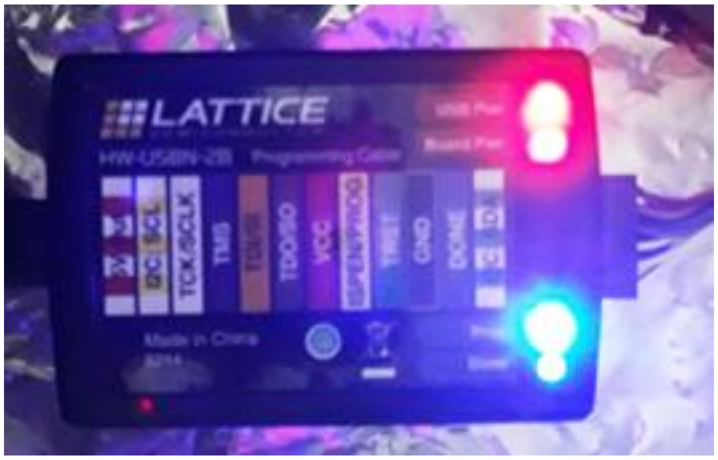
[✓ Use custom Clock Divider]

TCK Divider Setting (0-30x) 5

12. Click on Program to Start Programming.



	Enable	Status	Device Family	Device	Operation
1	✓		LIFMD	LIF-MD6000	... Erase,Program,Verify

13.  -> LED Status while Programming FPGA/SPI Flash.
14. The expected output on the console is as below:

```
INFO - Check configuration setup: Start.
INFO - Check configuration setup: Successful (Ignored JTAG Connection
Checking).
INFO - Device1 LIF-MD6000: SPI-M25PX16: SPI Flash Erase,Program,Verify
Initializing...
IDCode Checking...
Enabling...
Erasing...
Disabling...
Enabling...
Programming...
```

```
Disabling...
Verifying...
Finalizing...
INFO - Execution time: 00 min : 36 sec
INFO - Elapsed time: 00 min : 36 sec
INFO - Operation: successful.
```

2.4 Software Installation

PROCESSOR_SDK_RADAR_xx_xx_xx_xx_setupwin.exe is the SDK package installer.

Copy the installer to the path of your choice.

NOTE: For windows command prompt build install preferably at C: or D: top level folders to keep the overall path length small.

Double click the installer to begin the installation.

Follow the self-guided installer for installation.

IMPORTANT NOTE: On some computers running as administrator is needed. Right click on the installer and select option of "Run as administrator". If this is not done then you may see a message like "This program might not have installed correctly

On completion of installation a folder by name PROCESSOR_SDK_RADAR would have been created in the installation path.

2.4.1 Uninstall Procedure

To uninstall, double click on uninstall.exe created during installation in the folder PROCESSOR_SDK_RADAR.

At the end of uninstall, PROCESSOR_SDK_RADAR folder still remains. It is just an empty folder. It can be deleted manually.

3 Build and Run

This chapter provides a brief overview of the sample applications or use cases present in the SDK and procedure to build and run it.

3.1 Overview of application in release

The Processor SDK Radar supports the following use-cases

```
RADAR Usecases
--------------
TDA3xx EVM (no AWR12) or TDA2xx EVM (no AWR12):
6: Null Source (SD/Network) Input + Radar FFT (EVE1) + Null (SD/Network)

TDA3xx RVP
3: Radar (Single AR1243) Capture + Radar Object Detect (EVE1) + Display (TDA3xx Only)
7: Multi Radar (AR1243) Capture + Radar FFT (EVE1) + Display (TDA3xx Only)

TDA2xx Cascade Radar
9: Cascade Radar (4 AWR1243) Capture + Null (TDA2xx Only)
a: Cascade Radar (4 AWR1243) Capture + Radar Object Detect (DSP) + Null (TDA2xx Only)
```

Use option "s" on the main menu in UART to view PRCM Statistics and Bandwidth usage

3.2 Building the application

1. On windows command prompt, go inside the directory PROCESSOR_SDK_RADAR/vision_sdk/build.
2. Open file /vision_sdk/build/Rules.make and set

   MAKEAPPNAME=apps

   For TDA3xx RVP: **MAKECONFIG=tda3xx_rvp_bios_radar**

   For TDA2xx EVM (no AWR12): **MAKECONFIG= tda2xx_evm_bios_radar**

   For TDA2px EVM (no AWR12): **MAKECONFIG= tda2px_evm_bios_radar**

   For TDA2xx Cascade Radar: **MAKECONFIG= tda2xx_cascade_bios_radar**

   For TDA3xx EVM (no AWR12): **MAKECONFIG=tda3xx_evm_bios_radar** and modify configs/tda3xx_evm_bios_radar/cfg.mk with **RADAR_BOARD=none** (perform a clean build if already built for TDA3xx + DIB + VAB)
3. Build is done by executing gmake. "gmake" is present inside XDC package. For "gmake" to be available in windows command prompt, the XDC path must be set in the windows system path.

   IMPORTANT NOTE: xdc path is needed to be set in environment variables. If not, then set it using the set PATH = <Install_dir>/ti_components/os_tools/windows/xdctools_x_xx_xx_xx;%PATH% in command prompt

   IMPORTANT NOTE: If on Windows you are facing build issues, try by first setting the path to only the following:

set Path=C:\PROCESSOR_SDK_RADAR_<version>\ti_components\os_tools\windows\xdctools_<version>_core;C:\windows\system32;C:\windows;C:\windows\System32\Wbem;C:\windows\System32\WindowsPowerShell\v1.0\

IMPORTANT NOTE: If the installation folder depth is high then windows cmd prompt fails with error that it cannot find a file, even in file is present in mentioned path, this is because Windows has a limitation of 8191 characters for the commands that can execute. In such a situation as a workaround either restrict the folder depth to d:/ or if it cannot be restricted use git bash to build. Refer https://support.microsoft.com/en-in/kb/830473 **for more details**.

(**Always point to xdc path gmake only**)

4. Under vision_sdk directory
   a. When building first time run the below sequence of commands
      > gmake –s depend
      > gmake –s
   b. When building after the first time or incremental build, run the below command
      > gmake –s

   IMPORTANT NOTE: If you would like to speed up the compile time, you can use the option –j along with the gmake command:

   > gmake –s –j depend
   > gmake –s –j

   On windows machines it is recommended to use the –j<num of CPU cores of PC> instead of only –j. e.g. gmake –s –j2

Executing "**gmake –s –j depend**" will build all the necessary components (PDK drivers, EDMA drivers) and "**gmake –s –j**" will build the SDK framework and examples.

IMPORTANT NOTE: For incremental build, make sure to do "gmake -s –j depend" before "gmake –s -j" when below variables specified in /vision_sdk/apps/configs/$(MAKECONFIG)/*cfg.mk are changed

- when PROC_$(CPU)_INCLUDE is changed
- when DDR_MEM is changed
- when PROFILE is changed
- when ALG plugin or usecase is enabled or disabled in /vision_sdk/apps/configs/$(MAKECONFIG) /*_cfg.mk
- when any .h or .c file in TI component is installed in ti_components is changed
- when any new TI component is installed in ti_components

If "gmake -s –j depend" not done in these cases then build and/or execution may fail.

IMPORTANT NOTE: When options (other than those specified above) are changed in /vision_sdk/apps/configs/$(MAKECONFIG)/cfg.mk a clean build is recommended for the updated settings to take effect.

5. On a successful build completion, the executables will be generated in the below path

/vision_sdk/binaries/apps/$(MAKECONFIG)/vision_sdk/bin/tda3xx-evm

6. The build config that is selected in config file can be confirmed by doing below
   > **gmake –s showconfig**

7. Cleaning the build can be done by following command
   > **gmake –s clean**
   Alternatively, below folder can be deleted to delete all generated files
   > **gmake –j –s clean**
   > **rm –rf binaries**

3.3 Console Output

IMPORTANT NOTE:

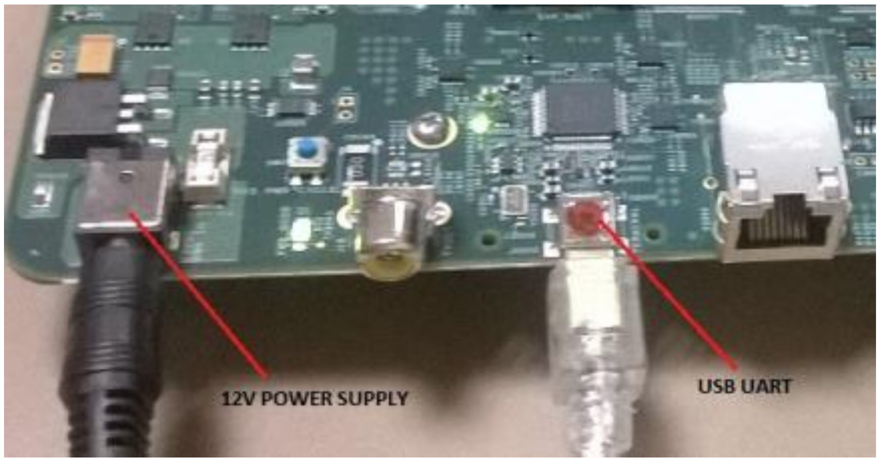
- **TDA3xx EVM Setup uses UART 2 and SBL uses UART 3. SDK logs appear on UART 2.**
- TDA2xx EVM Setup uses UART 1.

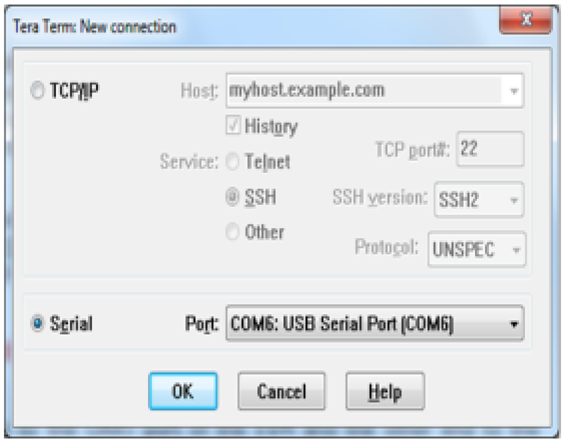
Connect a serial cable to the UART port of the EVM and the other end to the serial port of the PC (*configure the HyperTerminal at 115200 baud rate*) to obtain logs and select demo. EVM it detects 4 UART ports, you need to select the 2nd one.

IMPORTANT NOTE: On some EVMs we were observing that UART terminal does not work. Updating the USB to UART driver on PC made UART work on the failings PCs. You can download the drivers from the below links.

http://www.ftdichip.com/Drivers/VCP.htm

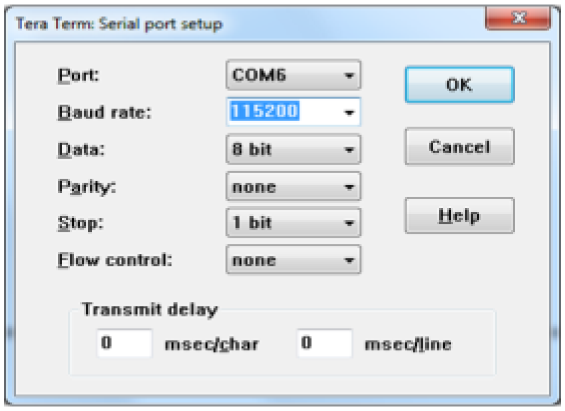
http://www.ftdichip.com/Drivers/CDM/CDM%20v2.10.00%20WHQL%20Certified.exe





Tera Term: New connection

- TCP/IP — Host: myhost.example.com; History; Service: Telnet / SSH / Other; TCP port#: 22; SSH version: SSH2; Protocol: UNSPEC
- Serial — Port: COM6: USB Serial Port (COM6)



Tera Term: Serial port setup

Port:	COM6
Baud rate:	115200
Data:	8 bit
Parity:	none
Stop:	1 bit
Flow control:	none

Transmit delay: 0 msec/char 0 msec/line

For RVP-TDA3x refer **3.3 UART settings** in VisionSDK_UserGuide_TDA3xx_RVP.pdf

3.4 Boot Modes

For TDA3xx refer **3.4 Boot Modes** in VisionSDK_UserGuide_TDA3xx.pdf

NOTE: On the RVP-TDA3x board the Boot Mode is always set to QSPI_SD.

For TDA2xx refer **3.4 Boot Modes** in VisionSDK_UserGuide_TDA2xx.pdf

3.5 Load using QSPI

For TDA3xx refer **3.5 Load using QSPI** in VisionSDK_UserGuide_TDA3xx.pdf

For TDA2xx refer **3.6 Load using QSPI** in VisionSDK_UserGuide_TDA2xx.pdf

3.6 Load using QSPI and SD boot

[Applicable only to TDA3xx]
Refer **3.6 Load using QSPI and SD boot** in VisionSDK_UserGuide_TDA3xx.pdf
NOTE: With AWR12 setup and TDA3x board modification this boot mode will not be functional

For RVP-TDA3x refer **3.4 Load using QSPI and SD boot** in VisionSDK_UserGuide_TDA3xx_RVP.pdf

3.7 Load using SD Card

[Applicable only to TDA2xx]
Refer **3.5 Load using SD card** in VisionSDK_UserGuide_TDA2xx.pdf

3.8 Load using CCS

For TDA3xx refer **3.7 Load using CCS** in VisionSDK_UserGuide_TDA3xx.pdf
For TDA2xx refer **3.8 Load using CCS** in VisionSDK_UserGuide_TDA2xx.pdf
For TDA3xx refer **3.5 Load using CCS** in VisionSDK_UserGuide_TDA3xx_RVP.pdf

IMPORTANT NOTE:

- **On the TDA2xx cascade Radar board, please make the following changes in the GEL files for the Ethernet to be functional.**

TDA2xx_pad_config.gel - Comment out the following lines:

```
    /* Set RGMII1_ID_MODE_N and RGMII2_ID_MODE_N in SMA_SW_1 for SR2.0 */
```

```
        WR_MEM_32(0x4A002534, (RD_MEM_32(0x4A002534) | (3 << 25)));
```

3.9 Run the demo

1. Power-on the Board after loading binaries by (SD, QSPI or CCS) and follow UART settings to setup the console for logs and selecting demo.
   Select demo required from the menu by keying in corresponding option from uart menu.

3.9.1 Usecase Specific Steps to run

3.9.1.1 Radar (Single AWR1243) Capture + Null (TDA3xx Only)

This usecase is used to capture from AWR12 hardware and Null.
User can dump the data from Null using CCS and feed the data to Radar Studio or any other PC tool for analysis.

Connect the AWR12xx hardware as per the guide (*EVM User's Guide V0.1 August 9, 2016*) provided along with hardware.

Select the usecase 1 (RADAR Use cases) and '2' (Capture Null) in usecase menu.

```
[IPU1-0]     87.683279 s:
[IPU1-0]     87.683370 s:  CHAINS: Init AR12xx ...
[IPU1-0]     87.683614 s:  UTILS_MCSPI: McSPI is configured in interrupt
mode!!
[IPU1-0]     89.040627 s:  AWR12XX: ES1.0 Device detected!!
[IPU1-0]     89.042670 s:  AWR12XX: Version Master : X.X.X.X
[IPU1-0]     89.042762 s:  AWR12XX: Version RF:X.X.X.X
[IPU1-0]     89.042823 s:  CHAINS: Config AR12xx ...
[IPU1-0]     89.042884 s:
[IPU1-0]
[IPU1-0]  ===========================
[IPU1-0]  Select Frame Configuration
[IPU1-0]  ===========================
[IPU1-0]
[IPU1-0]  1: Normal Frame
[IPU1-0]  2: Advanced Frame
[IPU1-0]
[IPU1-0]  Enter Choice:
```

Select '1' for normal frame configuration. This will allow you to first check if the board is operational with the normal frame capture.

To ensure demo functional
Print Performance Statistics and verify the capture fps is 15+

```
[IPU1-0]  [ ISSCAPTURE ] Link Statistics,
[IPU1-0]  ******************************
[IPU1-0]
[IPU1-0]  Elapsed time       = 11243 msec
[IPU1-0]
[IPU1-0]  Get Full Buf Cb    =  15.15 fps
[IPU1-0]  Put Empty Buf Cb   =  15.15 fps
[IPU1-0]  Driver/Notify Cb   =  15.15 fps
[IPU1-0]
[IPU1-0]  Input Statistics,
[IPU1-0]
```

```
[IPU1-0]  CH | In Recv | In Drop | In User Drop | In Process
[IPU1-0]     | FPS     | FPS     | FPS          | FPS
[IPU1-0]  ---------------------------------------------------
[IPU1-0]   0 |  15.15      0. 0      0. 0          15.15
[IPU1-0]
[IPU1-0]  Output Statistics,
[IPU1-0]
[IPU1-0]  CH | Out | Out     | Out Drop | Out User Drop
[IPU1-0]     | ID  | FPS     | FPS      | FPS
[IPU1-0]  -------------------------------------------
[IPU1-0]   0 |  0     15.15     0. 0      0. 0
```

If you are able to see 15 FPS capture for the normal frame configuration, you can confirm this demo is working correctly.
In order to save the data from CCS, you can follow the steps:

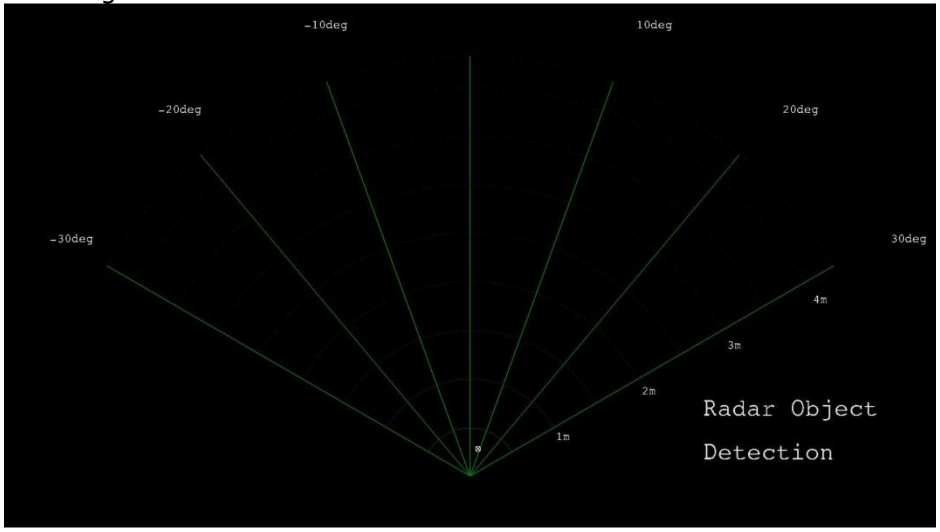
1. Place a break point at NullLink_drvDumpFrames in IPU1_0.
2. Once you hit the breakpoint, place the expression in CCS Expressions window – "((System_VideoFrameBuffer *)pBuf->payload)->bufAddr[0]". This contains the address of the buffer where the captured data is shown.

3.9.1.2 *Radar (Single AR1243) Capture + Radar Object Detect (EVE1) + Display (TDA3xx Only)*

This usecase is used to capture from AWR12 hardware Radar FFT Algo + Peak detection + Beam Forming on EVE, Radar Draw Objects on DSP and Display.

Connect the AWR12xx hardware as per the guide (*EVM User's Guide V0.1 August 9, 2016*) provided along with hardware.

Select the usecase 3 (RADAR Use cases) in usecase menu
Output is as shown similar to below based on the objects placed in-front of the radar and the current radar parameters. The maximum range and velocity vary based on the radar parameters. The latest SDK supports maximum range of 4 m for in room demonstrations and ultra-short range radar.



3.9.1.3 Radar (Single AWR1243) Capture + Radar FFT (EVE1) + DSP (FFT Heat Map) + Null (TDA3xx Only)

This usecase is used to capture from AWR12 hardware Radar FFT + Peak Detection + Beam Forming Algo on EVE, Radar Object Draw on DSP and Send out the data using Ethernet.

To flash the AWR12 firmware is flashed by selecting '1' (Radar Usecases), '1' AWR12 Firmware Flash, '2' Flash AWR12xx Firmware. Note this is one time operation and need not be done unless you plan to upgrade to a different AWR12 firmware.

```
[IPU1-0]  AWR12XX Flash Programming Menu
[IPU1-0]  -----------------------------
[IPU1-0]
[IPU1-0]  1: Erase AWR12xx Flash.
[IPU1-0]
[IPU1-0]  2: Flash AWR12xx Firmware.
[IPU1-0]
[IPU1-0]  x: Exit
```

Wait for the flashing to complete. This may take over 2-3 mins:

```
[IPU1-0]      34.812755 s:
[IPU1-0]      34.812785 s:  CHAINS: Started erasing the AWR12 flash memory....
[IPU1-0]      36.139083 s:  AWR12XX: ES1.0 Device detected!!
[IPU1-0]      49.296895 s:  CHAINS: Erasing AWR12 flash memory completed.
[IPU1-0]      49.296986 s:  CHAINS: Started flashing the AWR12 firmware....
[IPU1-0]      49.297078 s:  AWR12XX: Flashing BSS...
[IPU1-0]      95.136033 s:  AWR12XX: Flashing BSS finished.
[IPU1-0]      95.136125 s:  AWR12XX: Flashing MSS...
[IPU1-0]     117.495180 s:  AWR12XX: Flashing MSS finished.
[IPU1-0]     117.495272 s:  AWR12XX: Flashing Config...
[IPU1-0]     117.933660 s:  AWR12XX: Flashing Config finished.
[IPU1-0]     117.933752 s:  CHAINS: Finished flashing the AWR12 firmware.
[IPU1-0]     117.933813 s:
```

Exit out of the flashing menu and Select Usecase '4' for the usecase.
You would see the print below:

```
[IPU1-0]      87.683279 s:
[IPU1-0]      87.683370 s:  CHAINS: Init AWR12xx ...
[IPU1-0]      87.683614 s:  UTILS_MCSPI: McSPI is configured in interrupt
mode!!
[IPU1-0]      89.040627 s:  AWR12XX: ES1.0 Device detected!!
[IPU1-0]      89.042670 s:  AWR12XX: Version Master : X.X.X.X
[IPU1-0]      89.042762 s:  AWR12XX: Version RF:X.X.X.X
[IPU1-0]      89.042823 s:  CHAINS: Config AWR12xx ...
[IPU1-0]      89.042884 s:
[IPU1-0]
[IPU1-0]  ===========================
[IPU1-0]  Select Frame Configuration
[IPU1-0]  ===========================
[IPU1-0]
[IPU1-0]  1: Normal Frame
[IPU1-0]  2: Advanced Frame
```

Select '1' for normal frame configuration or '2' for the advanced frame.

```
[IPU1-0]  Select Network Mode,
[IPU1-0]  --------------------
[IPU1-0]  1: TFDTP
[IPU1-0]  2: TCP/IP
[IPU1-0]
[IPU1-0]  Enter Choice:
```

Select TFDTP or TCP/IP for the network transmit of the Object Detection output.

Once the usecase starts running you can use the following menu options to interact with the usecase:

```
[IPU1-0]  ====================
[IPU1-0]  Chains Run-time Menu
[IPU1-0]  ====================
[IPU1-0]
[IPU1-0]  0: Stop Chain
[IPU1-0]
[IPU1-0]  c: Read-back and Check AR params
[IPU1-0]
[IPU1-0]  Change display:
[IPU1-0]               1: Profile 0
[IPU1-0]               2: Profile 1
[IPU1-0]               3: Profile 2
[IPU1-0]               4: Profile 3
[IPU1-0]
[IPU1-0]  d: Dynamically change slope
[IPU1-0]
[IPU1-0]  p: Print Performance Statistics
```

First use 'p' to check the capture FPS is as expected (15 FPS for normal frame), (15 FPS for advanced frame).

Use 'c' to check if the AWR12 parameters are applied as expected. Note this works only for AWR12 ES2.0 samples.

```
[IPU1-0] 116194.933233 s:  CHAINS: Parameters have matched with programmed!!
```

Use 'd' to dynamically change the Profile parameter slope:

```
[IPU1-0]
[IPU1-0] 115768.469438 s:  CHAINS: AWR12xx Stopping Radar Sensor ...
[IPU1-0] 115768.471542 s:  CHAINS: AWR12xx Radar Stopped ...
[IPU1-0] 115768.503965 s:  CHAINS: Reconfiguring Parameters ...
[IPU1-0] 115768.508113 s:  CHAINS: Reconfigured Parameters ...
[IPU1-0] 115768.508296 s:  CHAINS: AWR12xx Re-starting Radar Sensor ...
[IPU1-0] 115768.512017 s:  CHAINS: AWR12xx Radar Started ...
[IPU1-0] 115768.512078 s:
```

In order to see the output from the network, in the PC command prompt type:

For TFDTP : `network_rx --host_ip <PC ipaddr> --target_ip <Board ipaddr> --usetfdtp  --files <file>`

For TCP/IP : `network_rx --host_ip <PC ipaddr> --target_ip <Board ipaddr> --files <file>`

This will save an RGB 565 image each of size 1920x1080. You can use an RGB viewer to visualize the output.

3.9.1.4 Null Source (SD/Network) Input + Radar FFT (EVE1) + Null (SD/Network)

Note: This usecase doesn't require AWR12xx Hardware and any board Modification When using SD card, the card needs to be in FAT32 file system

Input Clips can be found at
vision_sdk/apps/src/rtos/radar/src/usecases/radar_read_fft_write/Input_512x128_4Rx_1Tx_1TS_10Frm.bin
Details regarding this input file are captured in
vision_sdk/apps/src/rtos/radar/src/usecases/radar_read_fft_write/Input_512x128_4Rx_1Tx_1TS_10Frm_cfg.txt
This usecase does file read and write from either network or SD Card.

The user can select the mode of operation at run time using the following Menu option:

Note: On TDA3x EVM Network and FATFS cannot be enabled at the same time. At compile time you can select one of the two. Ensure your choice is consistent with the build time option. On TDA2xx EVM there is no compile time restriction. You can choose the option at run time.

```
Select Data Read/Write Mode,
----------------------------
1: SD CARD
2: NETWORK

Enter Choice:
```

Once the mode has been selected, based on the selection the source and destination of the FFT processing would be derived either from network or SD card.

3.9.1.4.1 SD Card

If SD Card is selected, ensure that the file Input_512x128_4Rx_1Tx_1TS_10Frm.bin is present on the SD Card.
The following Menu option is then displayed.

```
[IPU1-0]  ====================
[IPU1-0]  Chains Run-time Menu
[IPU1-0]  ====================
[IPU1-0]  0: Stop Chain
[IPU1-0]  p: Print Performance Statistics
[IPU1-0]  f: File IO Menu
[IPU1-0]  Enter Choice:
```

For further SD File read and write options Enter 'f':

```
[IPU1-0]  ====================
[IPU1-0]  FILE IO Run-time Menu
[IPU1-0]  ====================
[IPU1-0]
[IPU1-0]  a: Write Start
[IPU1-0]  b: Write Pause
[IPU1-0]  c: Write Resume
[IPU1-0]  d: Write Stop
[IPU1-0]  e: Write One Frame
```
Select the desired write option.
File write will create a file in sd card.

3.9.1.4.2 Network

If the network option is selected, based on the compile options if the NSP_TFDTP_INCLUDE = yes, the following menu is displayed.

```
Select Network Mode,
--------------------
1: TFDTP
2: TCP/IP

Enter Choice:
```

Enter the choice of TFDTP or TCP/IP.
On the PC side run the following applications for TCP/IP:

CONSOLE 1:
```
vision_sdk/apps/tools/network_tools/bin>
network_rx.exe   --target_ip   <BOARD_IP>   --host_ip      <PC_IP>   --files
../output_file.bin
```

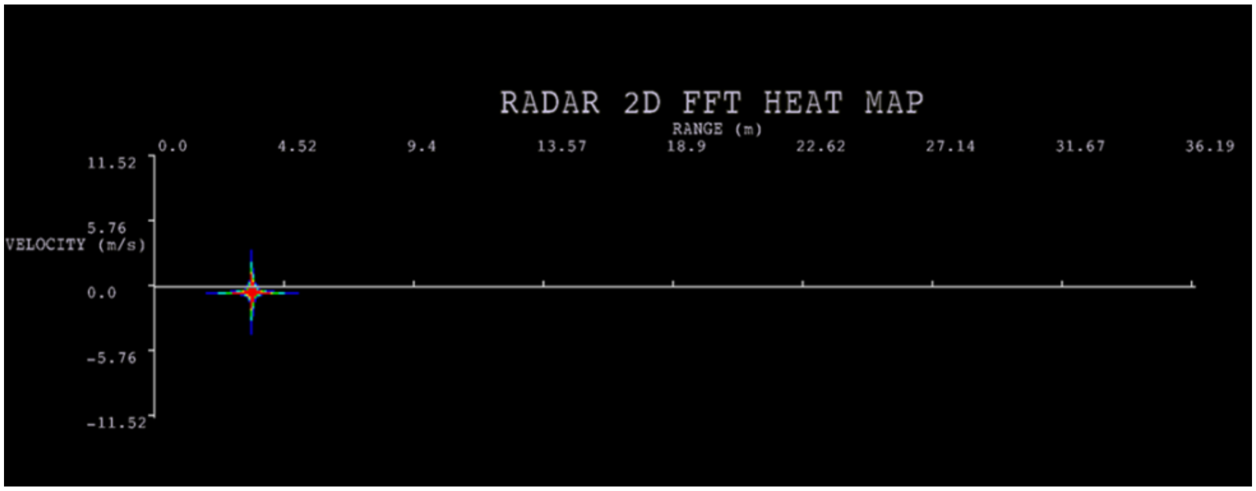
CONSOLE 2:
```
vision_sdk/apps/tools/network_tools/bin>
network_rx.exe  --target_ip  <BOARD_IP>  --host_ip   <PC_IP>  --files  <Install
Path>/
vision_sdk/apps/src/rtos/radar/src/usecases/radar_read_fft_write/Input_512x12
8_4Rx_1Tx_1TS_10Frm.bin
```

On the PC side run the following applications for TFDFTP:

CONSOLE 1:
```
vision_sdk/apps/tools/network_tools/bin >
network_rx.exe --target_ip <BOARD_IP> --host_ip   <PC_IP>  --usetfdtp  --files
../output_file.bin
```

CONSOLE 2:
```
vision_sdk/apps/tools/network_tools/bin >
network_rx.exe --target_ip <BOARD_IP> --host_ip   <PC_IP>  --usetfdtp  --files
<Install                                                                 Path>/
vision_sdk/apps/src/rtos/radar/src/usecases/radar_read_fft_write/Input_512x12
8_4Rx_1Tx_1TS_10Frm.bin
```

For both SD Card and network, the display shows the output on FFT Heat Map layout for the input.

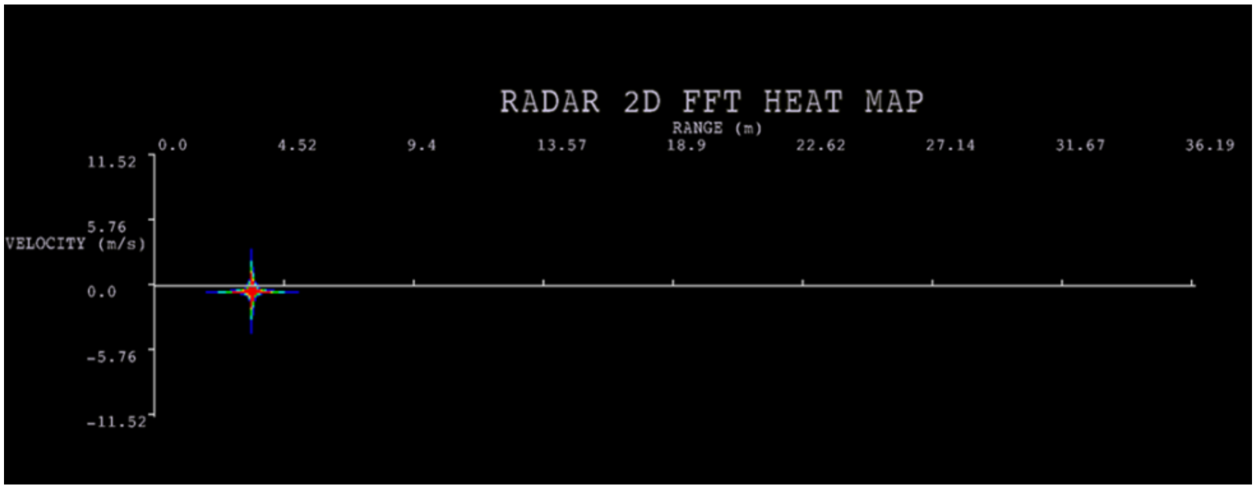


3.9.1.5 Satellite Radar (Single AWR1243) Capture + Radar FFT (EVE1) + DSP (FFT Heat Map) + Display (TDA3xx Only)

This usecase is used to capture from AWR12 hardware Radar FFT Algo on EVE, Radar DrawFFTHeatMap on DSP and Display.

Connect the AWR1243 satellite radar module to RVP-TDA3x VIN4 as per Chapter 2.3.3 in this guide.

Select the usecase 7 (RADAR Use cases) in usecase menu
Output is as shown similar to below based on the objects placed in-front of the radar and the current radar parameters. The maximum range and velocity vary based on the radar parameters.



Note: Intensity of detected object may vary depending on draw parameters and actual object reflection (Maximum Range and velocity and resolution subject to change)

3.9.1.6 Cascade Radar (4 AWR1243) Capture + Radar Object Detect (DSP) + Null

This usecase is used to capture the raw ADC data from the 4 AWR and then perform FFT on the EVE cores followed by Object detection on the DSP1.

Make sure the FPGA is flashed using the steps mentioned in Section 2.3.2.2.
Connect the Ethernet cable from the board to a router. The default build is for dynamic IP configuration.

Select the usecase 'a' in the menu options.

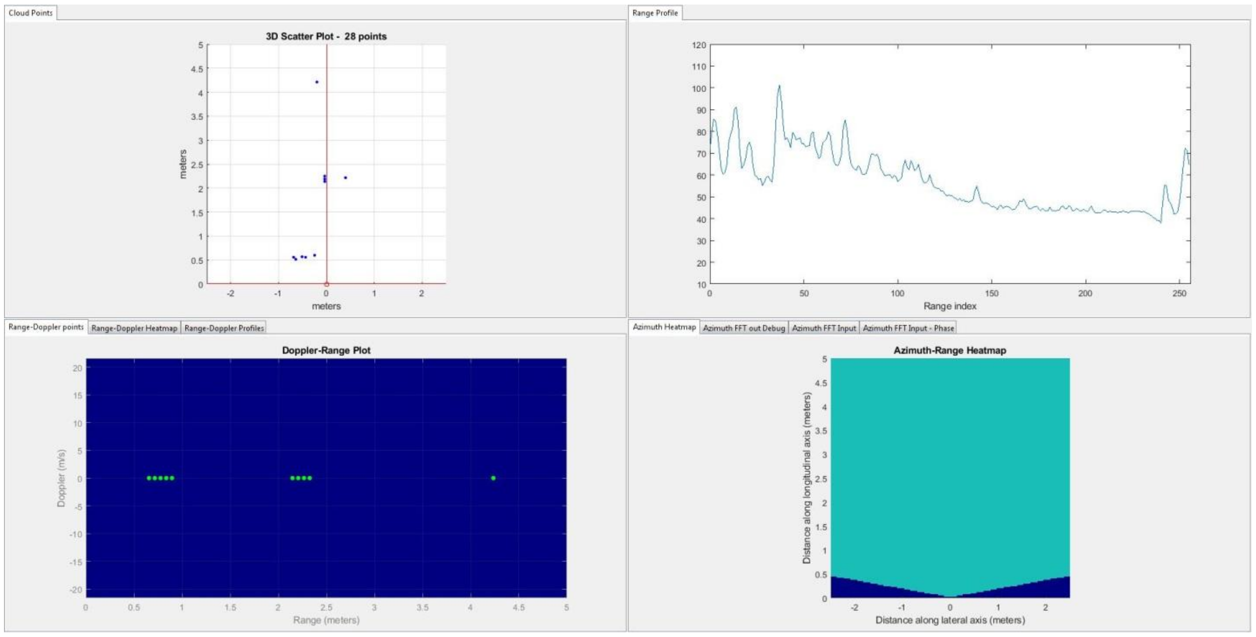
Once the usecase is running, on the PC connected to the network router open Matlab and change directory to vision_sdk\apps\src\rtos\radar\src\usecases\cascade_radar_object_detect

Run the following matlab script:
radar_cascade_demo(Target IP, Host IP, Host Mac Address, config, range_depth, profileNum);

For example:
radar_cascade_demo('172.24.190.64', '172.24.190.43', '5C-F9-DD-76-0B-76', 'TI', 5, 1);

The config can be 'TI', 'TIBF' etc. Please check the matlab script for the full list of out of box supported configurations.

The MATLAB GUI looks like below:



The usecase provides run time menu options to "Control GUI output" and "Control Calibration"

While entering the options for the GUI control of (1 1 0 0 1) one should type "1<space>1<space>0<space>0<space>1<enter>"

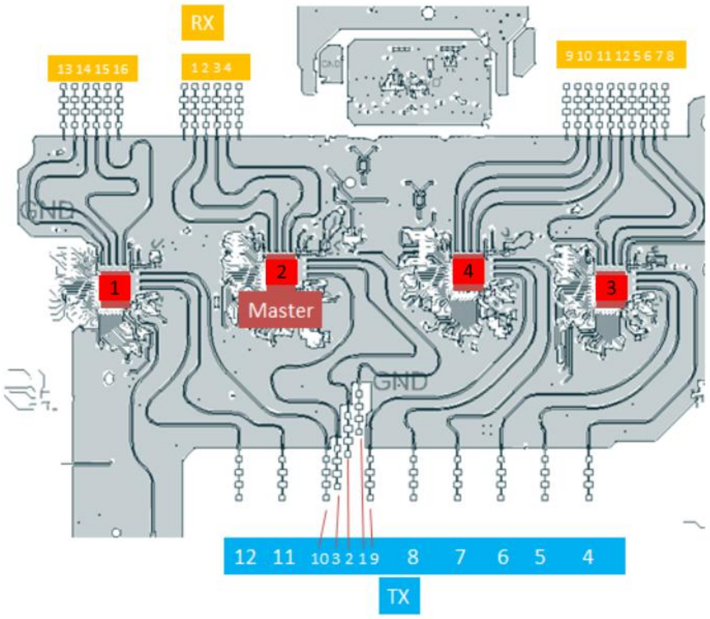
An example is shown below.

```
[IPU1-0]  ====================
[IPU1-0]  Chains Run-time Menu
[IPU1-0]  ====================
[IPU1-0]
[IPU1-0]  0: Stop Chain
[IPU1-0]
[IPU1-0]  p: Print Performance Statistics
[IPU1-0]
[IPU1-0]  g: Control GUI output
[IPU1-0]
[IPU1-0]  c: Control Calibration
[IPU1-0]
[IPU1-0]  Enter Choice:
[IPU1-0]
[IPU1-0]       234.222015  s:  GUI  Monitor  (0/1)  (<pointCloud>  <rangeProfile>
<noiseProfile> <rangeDopHeatMap> <azimuthHeatMap>):
[IPU1-0]    237.502135 s: 1
[IPU1-0]    238.813792 s:
[IPU1-0]    243.127127 s: 1
[IPU1-0]    244.174556 s:
[IPU1-0]    245.342005 s: 0
[IPU1-0]    246.247056 s:
[IPU1-0]    247.990392 s: 0
[IPU1-0]    248.871591 s:
[IPU1-0]    249.351064 s: 1
[IPU1-0]    250.502531 s:
[IPU1-0]    250.539681 s:
[IPU1-0]
[IPU1-0]  ====================
[IPU1-0]  Chains Run-time Menu
[IPU1-0]  ====================
[IPU1-0]
[IPU1-0]  0: Stop Chain
[IPU1-0]
[IPU1-0]  p: Print Performance Statistics
[IPU1-0]
[IPU1-0]  g: Control GUI output
[IPU1-0]
[IPU1-0]  c: Control Calibration
[IPU1-0]
[IPU1-0]  Enter Choice:
[IPU1-0]
[DSP1  ]    250.539376 s:  RADAR DSP PROCESS: Changing the GUI parameters
```

For Control Calibration, the Calibration is sent out over Ethernet to the Matlab tool.
This is saved as a CC.mat file.
Once this file is generated you need to run vision_sdk\apps\src\rtos\radar\src\usecases\cascade_radar_object_detect\create_calib_coefs.m to finally create the SD card based coefficients and then copy this over to the SD card for the next run.

3.9.1.6.1 Understanding Antenna Offsets in the board

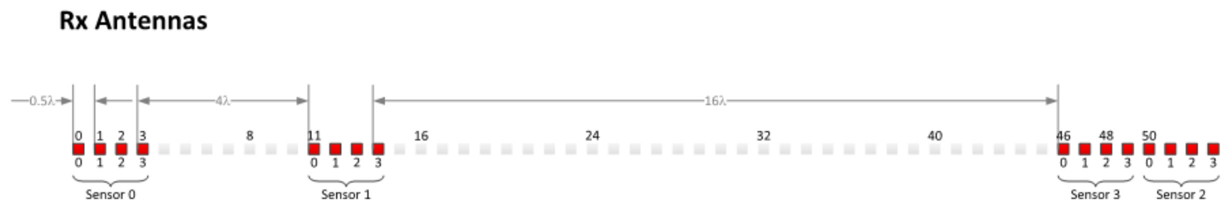
The antennas on the TI board are as shown below. The Rx antenna and the Tx antenna numbering convention is also given below.



The structures which need an update for a customer specific antenna configuration is as shown below:

```
aoa_rxAntOffset_t
gAoa_sensorRxOffset[RADAR_CASCADE_MAX_NUM_SENSORS] =
{
        {
                .sensorAzimOffs = 0,
                .antAzimOffs = {0, 1, 2, 3},
                .antElevOffs = {0, 0, 0, 0},
                .antRowOffs = {0, 0, 0, 0}
        },
        {
                .sensorAzimOffs = 11,
                .antAzimOffs = {0, 1, 2, 3},
                .antElevOffs = {0, 0, 0, 0},
                .antRowOffs = {0, 0, 0, 0}
        },
        {
                .sensorAzimOffs = 50,
                .antAzimOffs = {0, 1, 2, 3},
                .antElevOffs = {0, 0, 0, 0},
                .antRowOffs = {0, 0, 0, 0}
        },
        {
                .sensorAzimOffs = 46,
                .antAzimOffs = {0, 1, 2, 3},
                .antElevOffs = {0, 0, 0, 0},
                .antRowOffs = {0, 0, 0, 0}
        },
};
```

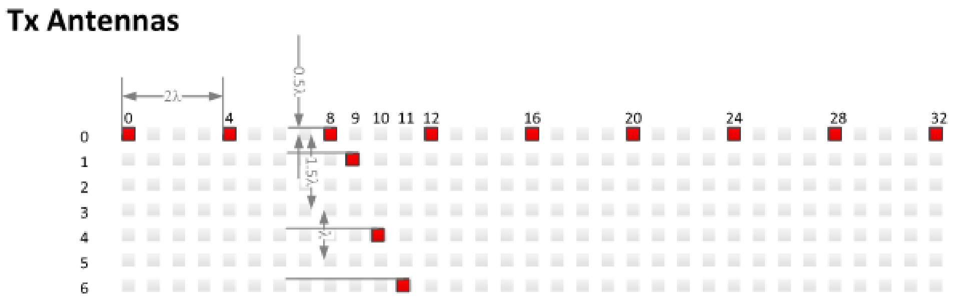
The way to update this is to look at each Rx antenna separated by half Lambda distance and then determine the sensorAzimOffs value. For example, the red boxes below show the antenna offsets corresponding to the TI board and the sensor numbering convention.



Similarly the Tx antenna configuration is given by the structure below:

```
aoa_txAntOffset_t gAoa_txAntOffset =
{
      .antAzimOffs = {0, 8, 16, 24, 32, 9, 10, 11},
      .antElevOffs = {0, 0,  0,  0,  0, 1,  4,  6},
      .antRowOffs = {0, 0,  0,  0,  0, 1,  2,  3},
};
```

Here the antAzimOffs and the antElevOffs are given in half lambda spacing convention. antRowOffs is given by an index number in the array which determines how the data should be arranged in memory for the elevation angle estimation. The order is determined by the chirp configuration which determines which Tx antenna fires after the other.



3.9.1.6.2 Steps to update the Matlab script

In order to change the profile parameters used in the MATLAB script, update the following entries:

```
% chirpPeriod = <Idle time in seconds> <ramp end time in seconds>
Params.chirpPeriod = 5e-6 + 40e-6;
% Fsamp = sampling rate in samples per second
Params.Fsamp = 8e6;
% freqSlope = Frequency slope in Hz per second.
Params.freqSlope = 79e12;
```

For your custom board, update the following values:

```
% Params
Params.numSensor = 4;
Params.numRxAntPerSensor = 4;
Params.numTxAzimAnt = 5;
Params.numTxElevAnt = 3;
Params.numTxAnt = Params.numTxAzimAnt + Params.numTxElevAnt;
Params.numVirtualAnt = (Params.numTxAnt )...
```

```
                                            * Params.numRxAntPerSensor *
Params.numSensor;
Params.numRangeBins = 256;
Params.numDopplerBins = 64;
Params.numAzimuthBins = 128;
Params.numElevBins = 64;
Params.azimuthRowLength = 86;
```

Also update the aoaInitLookupTable function by updating the following variables:

```
% sensorRxOffset[Number of Sensors]
%      -> sensorAzimOffs
%      -> antAzimOffs[ numRxAntPerSensor ]
%      -> antElevOffs[ numRxAntPerSensor ] --> Not used
%      -> antRowOffs [ numRxAntPerSensor ]
%
% txAntOffset
%      -> antAzimOffs [ numTxAnt ] (numTxAnt = numTxAzimAnt +
%      numTxElevAnt)
%      -> antElevOffs [ numTxAnt ]
%      -> antRowOffs  [ numTxAnt ]
% rowToElevationIdx
```

3.9.1.6.3 File-based Use Case Configuration Support

Starting from the SDK 3.6 release, a new feature to support reading cascade radar use case configurations from the file system is introduced. This support includes reading Radar front end and algorithms configurations. Please note that if the configuration file is not presented in the SD card, the use case will still run with the built-in configurations defined in the use case source code. To use the example configuration files, please copy radar_test_vector folder under ~\vision_sdk\apps\src\rtos\radar\src\usecases\cascade_radar_object_detect\ onto SD card. Please refer to the following instructions to customize the configurations.

- **Configuration File Path**
  By default, configuration files are expected to be found under "***radar_test_vector/***" folder on SD card. The default path can be changed by updating the macro, **RADAR_CONFIG_PARAM_DIR_NAME**, in ~/vision_sdk/apps/src/rtos/radar/include/common/chains_radar_util_read_params.h.
- **Configuration File Hierarchy**
  By default, the first file, "***sensor_master_config.txt***", to be read is an index file which points to the locations of the radar front end and algorithms master configuration files respectively. The file name is defined by the macro, **MASTER_SENSOR_CONFIG_FILENAME**, in ~/vision_sdk/apps/src/rtos/radar/include/common/chains_radar_util_read_params.h.
- **Master Configuration Files**
  For each master configuration file listed in "***sensor_master_config.txt***", it lists the individual configuration path for the corresponding instance. For examples, the cascade radar use case has 4 radar and 1 algorithm instances.
- **Radar Front End Configuration Files**
  For the cascade radar front end configuration file, the first instance (index 0) is assumed to be the master radar so its configuration file should contain all the fields which are configurable from the file input. The master radar configurations will be applied to other slave radars as the baseline values. The slave radar configuration files are expected to contain the different chirp

configurations. The chirp configurations of each slave will be updated using the input values read from the corresponding file.

- **Radar Algorithms Configuration Files**
  The cascade radar processing algorithms include 2D-FFT, CFAR, and AoA. The Algorithm Link Create Params of these algorithms can be read from the files. Please note that, for AoA, TX antenna offsets are required. This will be automatically calculated based on the radar front end chirp configurations and the pre-defined TX Antenna Array map as shown below. When using different cascade RF board than TI EVM, this map should be updated based on the physical layout.

```
/**< \brief This structure contains the Transmit antenna offsets in the elevation
 *          and azimuth direction in multiples of lambda by 2.
 */
Chanis_RadarAntOffset
gCascadeRadarTxAntArray[CHAINS_RADAR_CASCADE_NUMSENSORS][CHAINS_RADAR_CASCADE_MAX_TX_
ANTENNA_PERSENSOR] =
{
    /* Master */
    {   /* TX1, TX2, TX3 - {Azimuth offset, Elevation offset (in lambda/2)} */
        {11U, 6U}, {10U, 4U}, {9U, 1U}
    },
    /* Slave 1 */
    {   /* TX1, TX2, TX3 */
        {32U, 0U}, {28U, 0U}, {24U, 0U}
    },
    /* Slave 2 */
    {   /* TX1, TX2, TX3 */
        {20U, 0U}, {16U, 0U}, {12U, 0U}
    },
    /* Slave 1 */
    {   /* TX1, TX2, TX3 */
        {8U, 0U}, {4U, 0U}, {0U, 0U}
    }
};
```

4 Frequently Asked Questions

Q. Always see the error as sd card read error please retry

SD card must be in FAT32 format and SD card is not accessible if HW modification done for radar setup.

Q. While building SBL, path not found errors appear….. will this be an issue ?

This can be ignored as the following are not used for SBL build

Q. While building we find error as file not present, when checked the file is present at the location is there something to be set?

If the folder depth increases then windows cmd prompt fails with error cannot find file even in file is present in mentioned path, this is because of windows 8191 char limitation. Refer https://support.microsoft.com/en-in/kb/830473

Q. We see build error as

Interrupt/Exception caught (code = xxxxxxxx, addr = xxxxxxxxx)

This is because the gmake/make path is wrong. Set xpc_path as mentioned above and retry

Q. I get the following warnings while building:

```
vision_sdk/apps/configs/autorules_footer_cfg.mk:78:  ipc_PATH  does  not
exist! ()
vision_sdk/apps/configs/autorules_footer_cfg.mk:78: avbtp_PATH does not
exist! (ti_components/networking/avbtp_0_10_00_00)
vision_sdk/apps/configs/autorules_footer_cfg.mk:78: hdvicplib_PATH does
not                                                              exist!
(ti_components/codecs/ivahd_hdvicp20api_01_00_00_23_production)
vision_sdk/apps/configs/autorules_footer_cfg.mk:78:  jpegvdec_PATH  does
not exist! (ti_components/codecs/ivahd_jpegvdec_01_00_13_01_production)
vision_sdk/apps/configs/autorules_footer_cfg.mk:78:  jpegvenc_PATH  does
not exist! (ti_components/codecs/ivahd_jpegvenc_01_00_16_01_production)
vision_sdk/apps/configs/autorules_footer_cfg.mk:78:  vlib_PATH  does  not
exist! (ti_components/algorithms/vlib_c66x_3_3_0_3)
vision_sdk/apps/configs/autorules_footer_cfg.mk:78:   fc_PATH   does   not
exist! (ti_components/codecs/framework_components_3_40_02_07)
```

This is harmless. These are optional components which are not used by Processor SDK Radar.

Also refer VisionSDK_FAQs.pdf

5 Revision History

Version	Date	Revision History
1.0	29th Sept 2016	Initial Version
1.1	30th Sept 2016	Addressed Review comments
1.2	26th Oct 2016	Updates for Release 2.11
1.3	30th Jan 2017	Updates for Release 2.12
1.4	26th June 2017	Updates for Release 3.00
1.5	28th June 2017	Updated linux installation steps
1.6	14th Oct 2017	Updates for Release 3.01
1.7	20th Dec 2017	Updates for Release 3.02
1.8	4th April 2018	Updates for Release 3.03
1.9	29th June 2018	Updates for Release 3.04
1.10	26th Sept 2018	Updates for Release 3.05
1.11	24th Dec 2018	Updates for Release 3.06. Added details on the calibration process and antenna offset calculation to TI cascade board.

««« § »»»