TI mmWave Labs

Medium Range Radar 2.0

Beam Steering with AWR1843

- Developer’s Guide -

Important Information

- **MRR Beamsteering Release v 2.0**
  - **Please read release notes for modes supported in this release**

Lab Overview

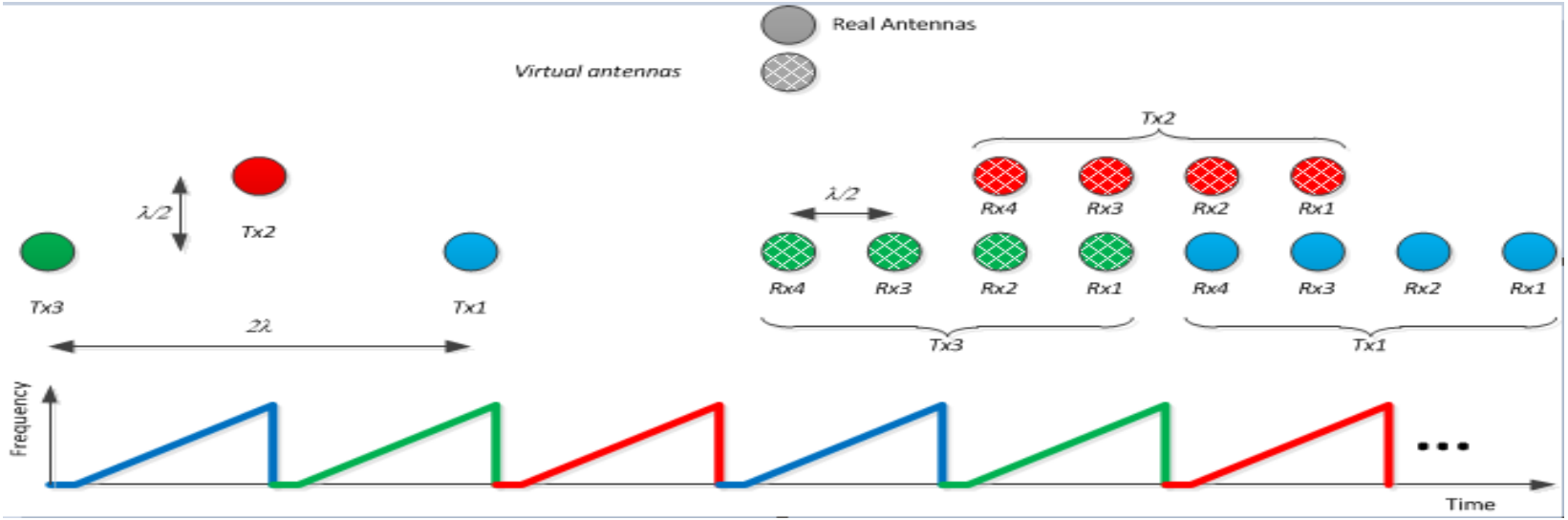
- This lab exercise demonstrates the use of 6-bit Phase Shifters available on AWR1843 mmWave Sensor to steer the beams at any angle with a granularity of 5.625 degrees.
- The detected objects are tracked (in the azimuthal and elevation plane) up to ~150m.
- Multi-mode capability is implemented in the current MRR Lab.
-

PARAMETER	MRR	USRR	DETAILS
Max Range	150m	30m	This represents the maximum distance that the radar can detect an object representing an RCS of approximately 10 m2.
Range Resolution	68.2 cm	4.3cm	Range resolution is the ability of a radar system to distinguish between two or more targets on the same bearing but at different ranges
Max Velocity	150 kmph	36 kmph	This is the native maximum velocity obtained using a two-dimensional FFT on the frame data. This specification will be improved over time by showing how higher-level algorithms can extend the maximum measurable velocity beyond this limit.
Velocity Resolution	0.11m/s	0.32m/s	This parameter represents the capability of the radar sensor to distinguish between two or more objects at the same range but moving with different velocities.

- Code Composer Studio (CCS) project along with source code is provided for this lab
- Pre-built binary files are also provided that can be loaded on to the AWR1843BOOST EVM

System Design

- **Antenna Configuration**
  - The AWR1843BOOST EVM uses four receivers and the three transmitters in two different chirp configurations. The first configuration (MRR) uses a TX beamforming with all the 3 TX transmitting simultaneously. The second configuration (USRR) uses a time division multiplexed MIMO configuration (that is, alternate chirps in a frame transmit on TX1, TX2 and TX3 respectively.). The MIMO configuration synthesizes an array of twelve virtual RX antennas(8 antennas in azimuthal plane and 4 antennas in the elevation plane), as shown in Figure 3. This technique improves the angle resolution by a factor of two (compared to a single TX configuration).



Chirp and System Performance Parameters

- **Chirp Configuration**

Parameter	MRR	USRR
Idle Time(us)	5	5
ADC start time (us)	4.8	3
Ramp end time(us)	60	44.0
Number of ADC samples	256	512
Frequency Slope(MHz/us)	4	56.25
ADC Sampling frequency(ksps)	4652	12500
MIMO	0	1
Number of chirps per profile	256	256
Effective chirp time(us)	53	42
Bandwidth(MHz)	240	3456
Frame Length(ms)	16.6	10.5
Memory requirement(KB)	1024	512

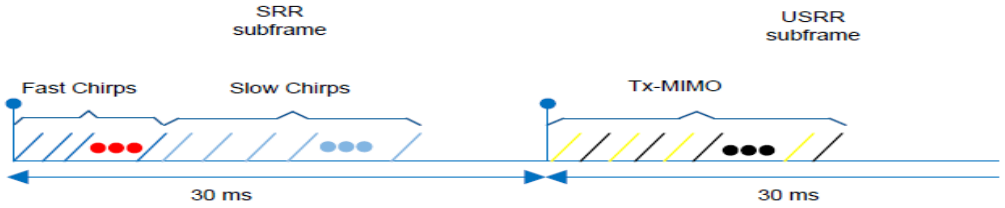
- **System Performance Parameters**

Parameter	MRR	USRR
Range Resolution(m)	0.70	0.043
Maximum Distance(m)	150	30
Maximum Velocity(kph)	150	36

Implementation on the AWR1843

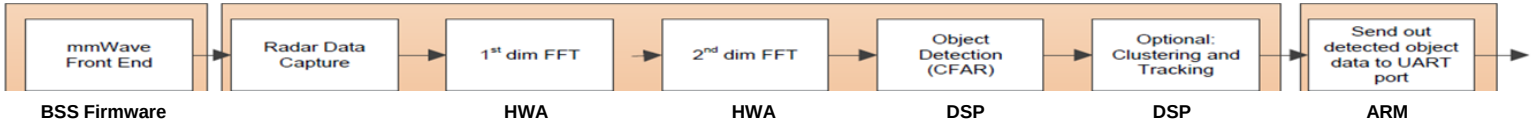
- **Modes Supported**
  - Three Modes
    - MRR subframe only
    - USRR subframe only
    - USRR/MRR subframes
- **Configuration Profile**
  - This lab makes use of “Advanced Frame Configuration” APIs to meet the requirements of the MRR and USRR use cases.
  - These APIs allow construction of frames consisting of multiple sub-frames, each one corresponding to particular use case.
  - 2 sub-frames are used in the USRR/MRR mode, 1 sub-frame for the MRR context and the other for USRR context.

Frame Construction



- **MRR subframe**
  - This subframe consists of two kinds of chirps (fast chirps and slow chirps). Both fast and slow chirps have the same slope; however, the slow chirp has a slightly higher ‘chirp repeat periodicity’ than the fast chirp. As a result, the slow chirps (when processed after 2D fast-Fourier transform (FFT)) have a lower maximum unambiguous velocity as compared to the fast chirps. Note that the fast and slow chirps do not alternate; instead, the fast chirp is repeated a certain number of times, followed by the slow chirp, which is again repeated an equal number of times. The purpose of this chirp design is to use the two separate estimations of target velocity from the ‘fast chirp’ and the ‘slow chirp’ along with the ‘Chinese remainder theorem’ to generate a consistent velocity estimate with a much higher max-velocity limit.
- **USRR subframe**
  - This subframe consists of three alternating chirps. Each chirp utilizes one of the threeTxs available on the AWR1843 device. Combined processing of this subframe allows the generation of a virtual Rx array of twelve Rx antennas, which consequently has better angular resolution (approximately 14.3°) than the 4 Rx antenna array.

Data Path – Processing Chain



- The RF front end is configured by the BIST subsystem (BSS). The raw data obtained from the various front end channels is taken by the HWA subsystem for processing.
- Processing during the chirps consists of:
  - 1D (range) FFT processing performed by the HWA that takes input from multiple receive antenna from the ADC buffer for every chirp (corresponding to the chirping pattern on the transmit antennae)
  - Transferring transposed output into the L3 RAM by enhanced direct memory access (eDMA)
- Processing during the idle or cool down period of the RF circuitry following the chirps until the next chirping period. This processing consists of:
  - 2D (velocity) FFT processing performed by the HWA that takes input from 1D output in L3 RAM and performs FFT to give a (range, velocity) matrix in the L3 RAM. The processing also includes the
  - CFAR detection in Doppler direction performed by the DSP. CFAR detection in range direction performed by the DSP and uses the mmWave library.
  - Peak grouping (for both Doppler and range) for the MRR subframe and Doppler for the USRR subframe
  - Direction of arrival (azimuth and elevation) estimation to map the X-Y-Z location of object
  - Additional pruning based on the SNR and the 2D-FFT magnitude of the object to avoid ground clutter

Clustering

- Clustering is performed using the dBscan algorithm .
- Algorithm is applied on the detected objects of both the MRR and the USRR subframes.
- The output of the clustering algorithm is the mean location of a cluster and its dimensions (assuming the cluster is a rectangle).
- For the USRR subframe, the clustering output is sent as is to the graphical user interface (GUI).
- USRR clusters allow the grouping of dense point clouds to rectangles. In cross-traffic scenarios, these clusters can be used to identify vehicles crossing the field of vision (FoV) of the radar.

Tracking

- Clustering output forms the input for the tracking algorithm.
- The strongest object in the cluster is provided as a representative object for the tracking algorithm.
- The tracker is a fairly standard Extended Kalman Filter (EKF with four states [x, y, vx, vv] and three inputs [r, v, sin(θ)] or range, relative velocity, and sine of the azimuth).
- Signal to noise ratio is taken as input and associated variance is computed using the Cramer-Rao Lower Bound formula for frequency variance .

Output Data Format

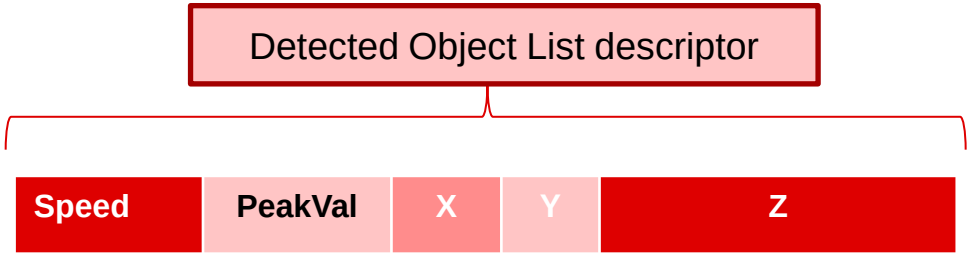
- Data Packet Format
  - A TLV(type-length-value) encoding scheme is used with little endian byte order. For every frame, a packet is sent consisting of a fixed sized **Frame Header** and then a variable number of TLVs .

  Below are the possible TLVs

TLV Name	Type
Detected Points	1
Clusters	2
Tracked Objects	3

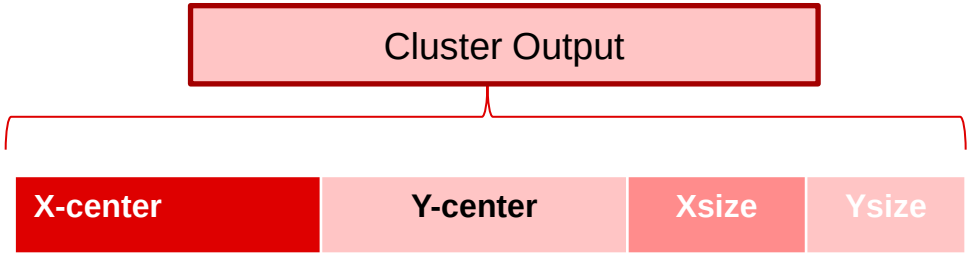
Detected Object List

Parameter	Size
Doppler Index	short(uint16)
Peak Vale	short(uint16)
X Co-ordinate in meters	short(uint16)
Y Co-ordinate in meters	short(uint16)
Z Co-ordinate	short(uint16)



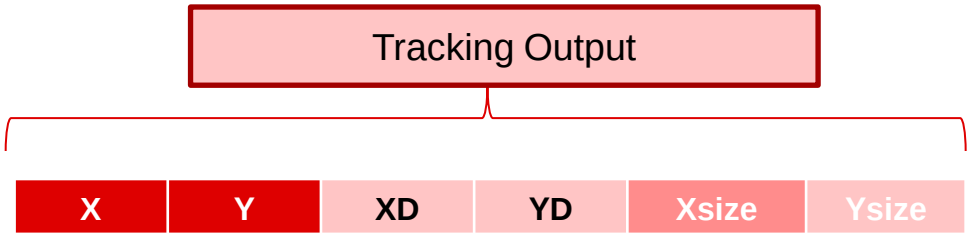
Cluster Output Format

Parameter	Size
Clustering center on X-direction	short(uint16)
Clustering center on Y-direction	short(uint16)
Clustering size on X-direction	short(uint16)
Clustering size on Y-direction	short(uint16)



Tracking Output Format

Parameter	Size
Tracking X co-ordinate	short(uint16)
Tracking Y co-ordinate	short(uint16)
Velocity in X direction	short(uint16)
Velocity in Y direction	short(uint16)
Cluster Size(x direction)	short(uint16)
Cluster Size(y direction)	short(uint16)



Demo Configuration: Operation Modes

- The demo supports 3 modes of operation
  - MRR only
  - USRR only
  - MRR/USRR
    - In this mode alternative MRR and USRR sub-frames are being generated
- The mode of operation is hard coded in the implementation
  - See “lab0011_mrr_beamsteering\src\common\mrr_config_consts.h”
  - In order to change the mode the demo must be re-build

```
/**! @brief The multi-mode Radar mode of operation. */
#define SUBFRAME_CONF_MRR_USRR                  /* Two subframes, MRR followed by USRR20. */
/**! @brief The USRR only mode of operation. */
//#define SUBFRAME_CONF_USRR                    /* One subframe USRR20. */
/**! @brief The MRR only mode of operation. */
// #define SUBFRAME_CONF_MRR                    /* One subframe MRR80. */
```

Demo Configuration: CLI configuration

- Most of the mmWave demos provided by our team on TI Resource Explorer support a CLI configuration from the PC host:
  - The RF and demo configuration parameters are usually provided in a file called profile.cfg
  - The content of this file is sent to the mmWave sensor target through the UART
  - The application running on the target uses this information to configure the RF front end and the demo.
- The MRR demo also supports a CLI, however the main difference with the other demos is that the RF configuration is not sent through the UART but hard coded in the application that runs on the target.
- The MRR demo supports only two commands which are sent by the Matlab GUI through UART to start the demo

Demo Configuration: CLI configuration (Cont.)

- Please see in following file “lab0011_mrr_beamsteering\gui\MRRvisualization\load_config.m”

```
cliCfg{1} = 'advFrameCfg';
cliCfg{2} = 'sensorStart';
```

- The MRR demo can be started using these commands using Tera Term for example.

Demo Configuration: RF configuration

- The RF configuration (profile, chirp, frame…) is performed using Advanced Frame Configuration. For more information about Advanced Frame Configuration please see following appnote: “Section 6 Advanced Chirp Configurations” in ***Programming Chirp Parameters in TI Radar Devices***
  - http://www.ti.com/lit/an/swra553/swra553.pdf
- As mentioned in previous section, the RF configuration is not sent through UART as in other demos.
- The RF configuration is hard coded in the code in the following file:
  - “lab0011_mrr_beamsteering\src\common\cfg.c”
  - “Cfg_AdvFrameCfgInitParams()”
- In order to modify the RF configuration one would need to re-build the demo.

Demo Configuration: Beam Steering

- This demo configures the beam steering on a per frame basis. This means that the beam steering angle is the same for all the chirps included in one frame.
- The beams steering configuration is hard coded.
- In the prebuild binaries, the beam steering is configured as follows
  - Max Angle = 60 deg
  - Min Angle = -60 deg
  - Angle Step Size = 20 deg
- The beams are alternating as follows:
  - -60deg; -40deg; -20deg; 0deg; 20deg; 40deg; 60deg; -60deg; -40deg….
- In the prebuild binaries, the frame duration is 60ms, so the beam steering cycling time is 7 x 60ms

Demo Configuration: Beam Steering (cont.)

- Steps to change the Beam Steering configuration:
  - In the file “lab0011_mrr_beamsteering\src\mss\mss_mrr_cli.c”, the initial value of the steering angle is defined as

```
gMrrMSSMCB.steer_Angle = -60;
```

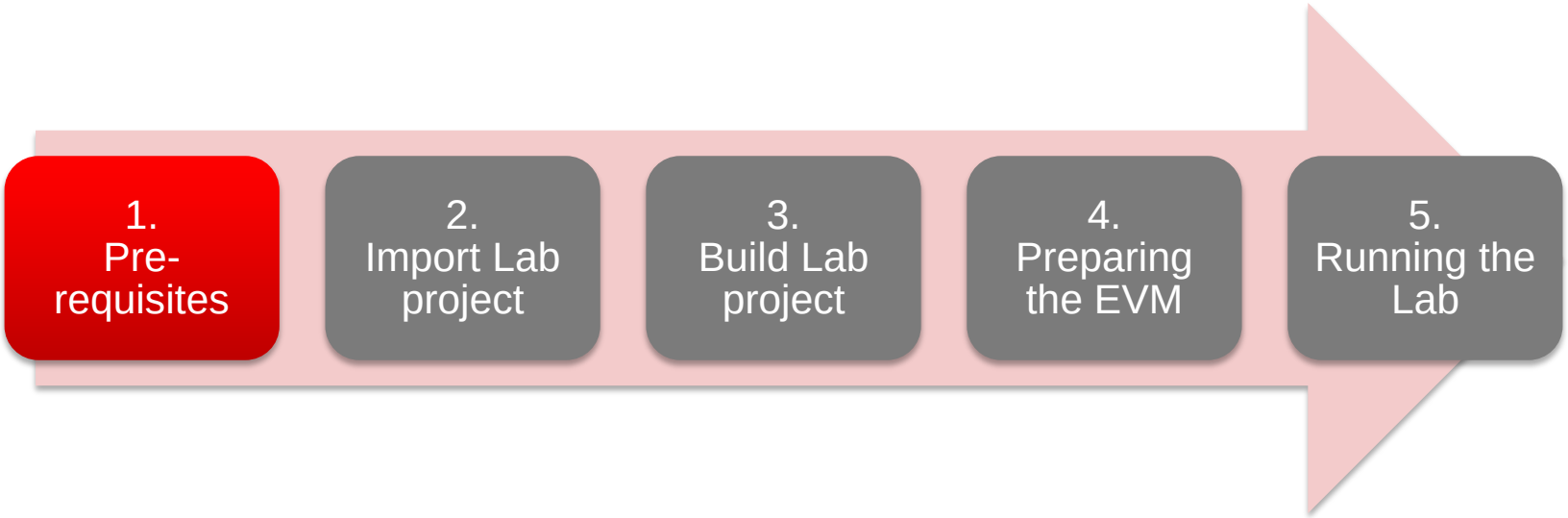
  - The value “-60” must be replaced with the new desired initial steering angle.

Demo Configuration: Beam Steering (cont.)

- Steps to change the Beam Steering configuration (cont.) :
  - In the file “lab0011_mrr_beamsteering\src\mss\mss_main.c” in the function “MRR_MSS_chirpIntCallback()”, the value “20” is the step size and the values “60” and “-60” are the maximum and minimum steering angles.
  - These values must be updated with the new desired values

```
gMrrMSSMCB.subframeId = 0;
Get_PhaseTxShifterCodeValue(gMrrMSSMCB.steer_Angle, &tx1phcode, &tx2phcode, &tx3phcode);
Cfg_TxPhaseShiftInitParams(&g_ptrtxPhaseShiftCfg, tx1phcode, tx2phcode, tx3phcode );
Semaphore_post (gMrrMSSMCB.phShftSemHandle);
gMrrMSSMCB.steer_Angle = gMrrMSSMCB.steer_Angle + 20;
if(gMrrMSSMCB.steer_Angle > 60)
{
    gMrrMSSMCB.steer_Angle = -60;
}
```

Steps for Building from the Source Code and Run

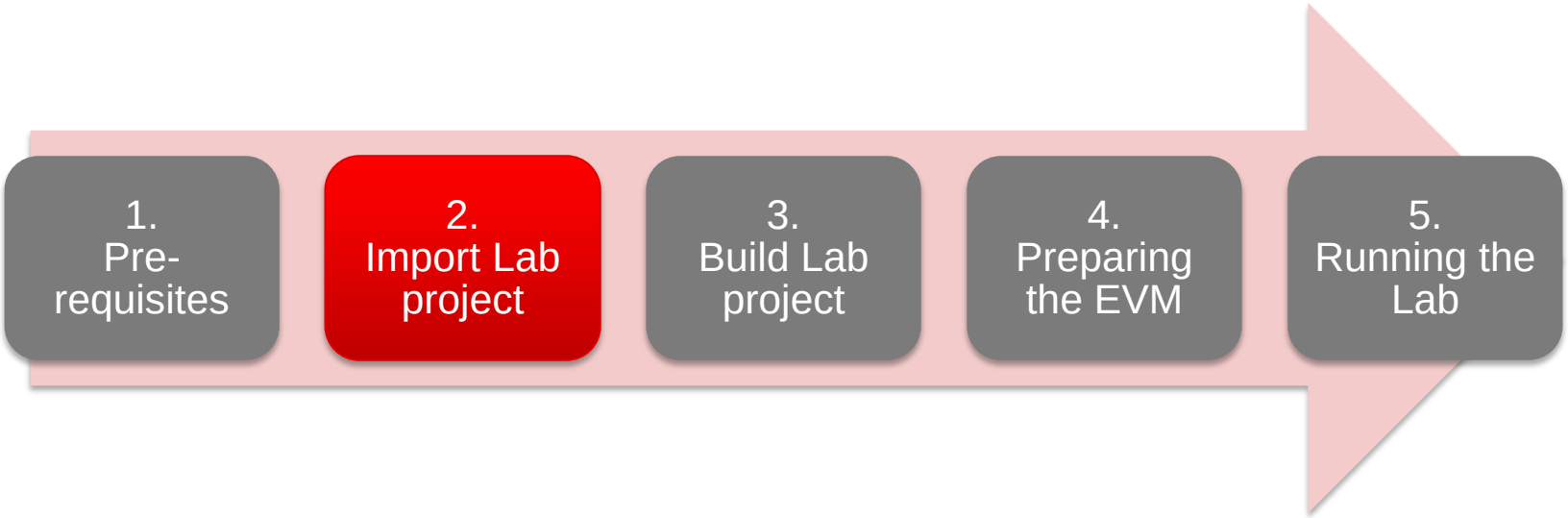


1. Pre-requisites

- Please read in the Lab Release Notes the version of the SDK required by this Lab
- It is assumed that you have the TI mmWave SDK and all the related tools installed as mentioned in the mmWave SDK release notes.
  - The mmWave SDK release notes include the links for downloading the required versions of the above tools.
- If you have already installed the mmWave SDK and all the required tools, you can move on to the next step i.e. downloading the lab on to your machine.

Note: Refer to the mmWave SDK for the dependent tools.

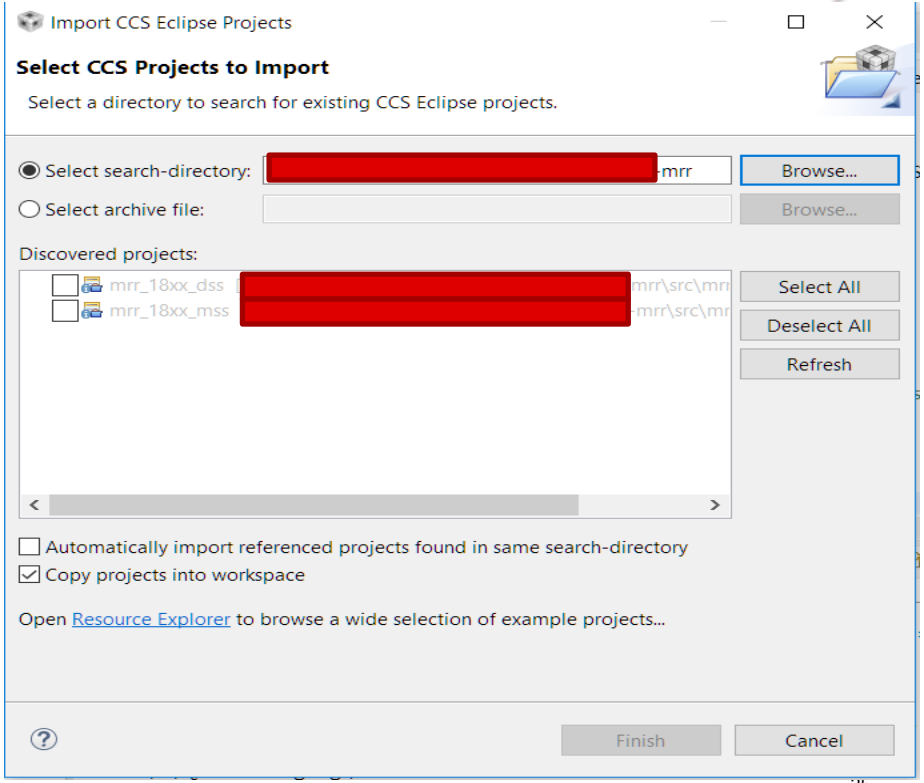
Steps



2. Import Lab Project

- The **Medium Range Radar** Lab consists of two CCS projects, one for the R4F core and one for the C674x DSP core
- The CCS projects are included in the Automotive Toolbox zip file installed as described in the Getting Started Guide.
- Open CCS. Select the “CCS Edit” perspective

- Select **Project ➔ Import CCS Projects.** Browse to select the lab folder
- The labs projects should be discovered by CCS. **Select All** and **Finish**
- This copies the projects in the user’s workspace and imports it into the CCS project explorer.
  - It is important to note that the copy created in the workspace is the one that gets imported in CCS. The original project downloaded in mmwave automotive toolbox is not touched.

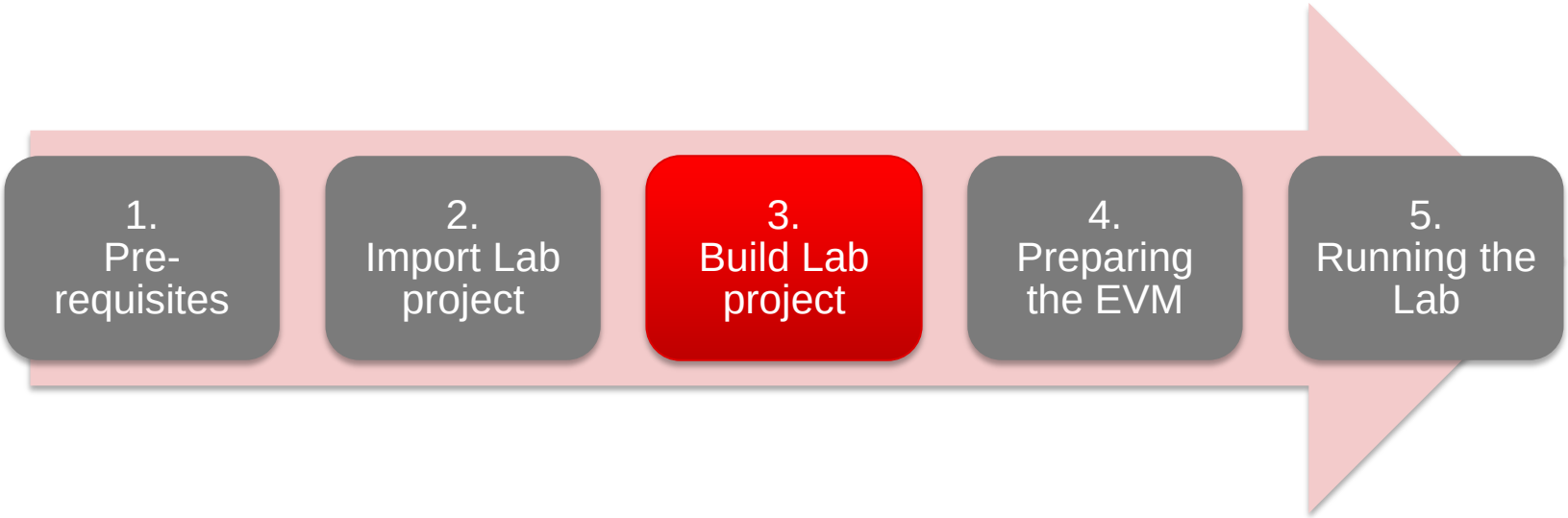


2. Import - continued

- The projects should be visible in CCS Project Explorer as shown here.

- We are ready to move on to the next step i.e. Building the project.

Steps


1.
Pre-
requisites
2.
Import Lab
project
3.
Build Lab
project
4.
Preparing
the EVM
5.
Running the
Lab

3. Build the Lab

- At this point we assume the two projects have been imported in CCS. If they have not please go to Step 2.

- With the **dss** project selected in Project Explorer, right click on the project and select **Rebuild Project**.
  - Selecting **Rebuild** instead of **Build** ensures that the project is always re-compiled. This is especially important in case the previous build failed with errors.

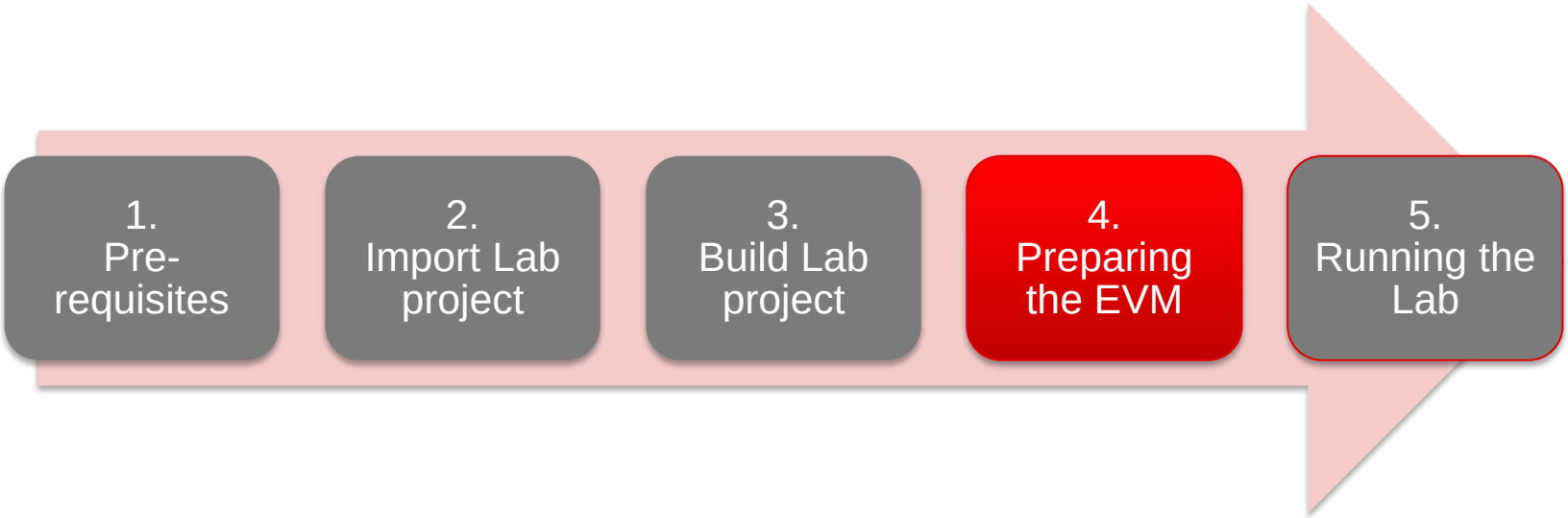
- On successful completion of the build, you should see the output in CCS console as shown here and the following two files should be produced in the project debug directory
  - mrr_18xx_dss.xe674
  - mrr_18xx_dss.bin

- If the build fails with errors, please ensure that all the pre-requisites are installed as mentioned in the mmWave SDK release notes.

- The **dss** project must be built BEFORE the **mss** project.
- With the **mss** project selected in Project Explorer, right click on the project and select **Rebuild Project**
- On successful completion of the build, you should see the output in CCS console as shown here and the following three files should be produced in the project debug directory
  - xwr18xx_mrr_demo.bin
  - mrr_18xx_mss.bin
  - mrr_18xx_mss.xer4f

- If the build fails with errors, please ensure that all the pre-requisites are installed as mentioned in the mmWave SDK release notes.

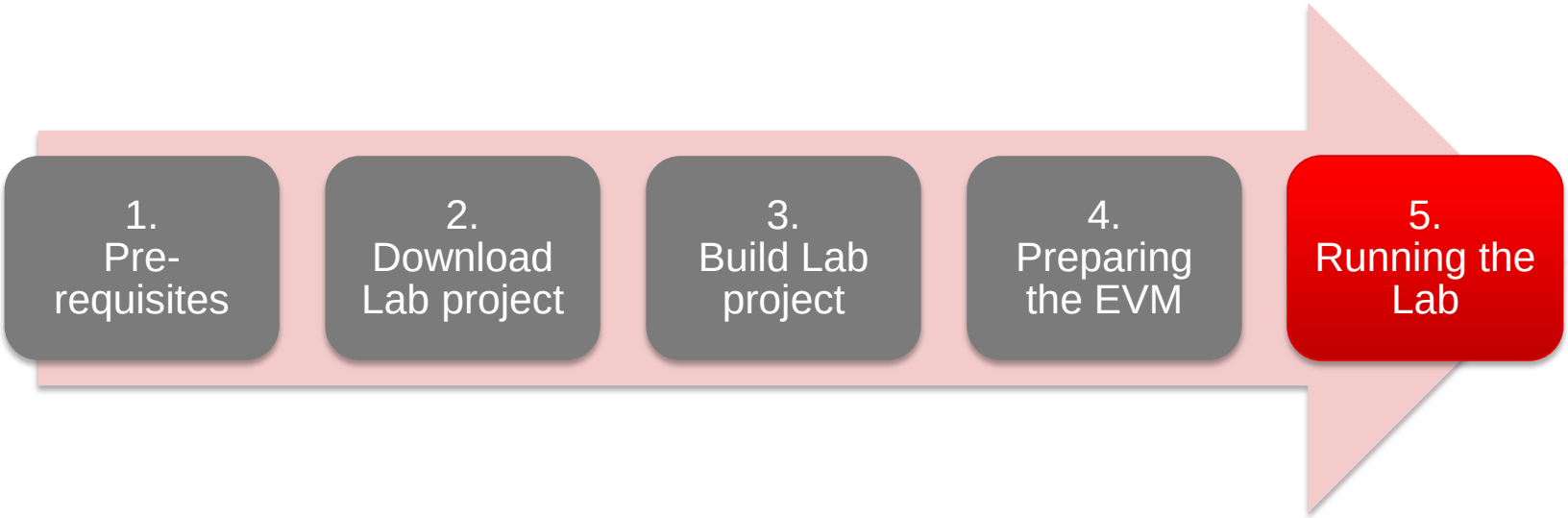
Steps


1.
Pre-
requisites
2.
Import Lab
project
3.
Build Lab
project
4.
Preparing
the EVM
5.
Running the
Lab

4. Preparing the EVM

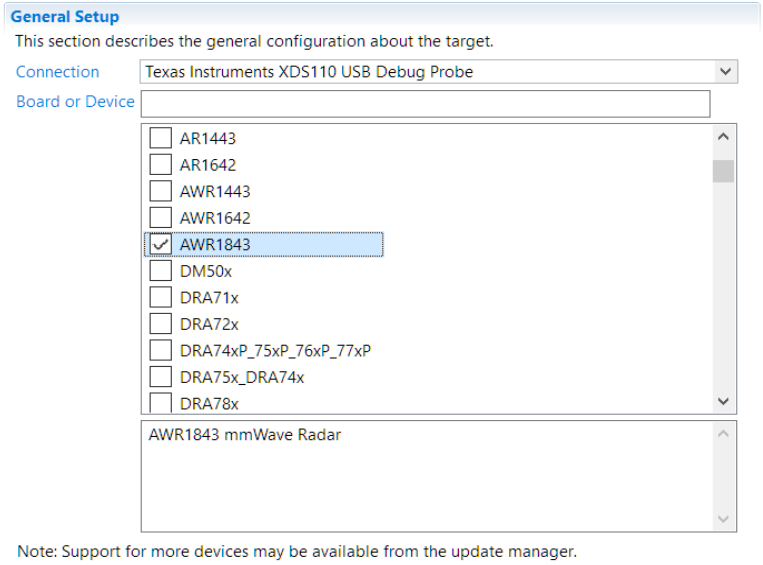
- There are two ways to execute the compiled code on the EVM:
  - Deployment mode : Flashing the binary (.bin image) on to the EVM serial flash
    - In this mode, the EVM boots autonomously from flash and starts running the bin image.
  - Debug mode: Downloading and running the executable (.xer4f image) from CCS.
    - You will need to flash a small CCS debug firmware on the EVM (one time) to allow connecting with CCS. This debug firmware image is provided with the mmWave SDK.
- The Getting Started Guide has demonstrated the deployment mode
- The following presentation explains the second method i.e. Debug mode (CCS).
  - To prepare the EVM for debug mode, we start with flashing the CCS debug firmware image.
  - Please use the flashing process described in the Getting Started Guide to flash the following binary to the EVM.
    - **C:\ti\mmwave_sdk_xx_xx_xx_xx\packages\ti\utils\ccsdebug\xwr18xx_ccsdebug.bin**

Steps



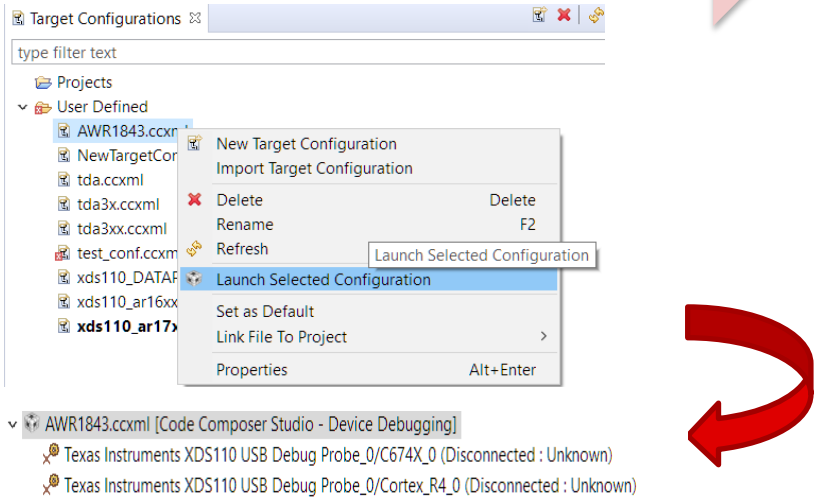
5. Connecting EVM to CCS

- It is assumed that you were able to download and build the Lab in CCS (completed steps 1, 2 and 3)
- To connect the Radar EVM to CCS, we need to create a target configuration
  - Go to File ► New ► New Target Configuration File
  - Name the target configuration accordingly and check the “Use shared location” checkbox. Press Finish
  - In the configuration editor window:
    - Select “Texas Instruments XDS110 USB Debug Probe” for **Connection**
    - Select AWR1843 device in the **Board or Device** text box.
    - Press the **Save** button to save the target configuration.
    - You can press the **Test Connection** button to check the connection with the board.



5. Connecting - continued

- Go to **View ► Target Configurations** to open the target configuration window.
- You should see your target configuration under **User Defined** configurations.
- Right click on the target configuration and select **Launch Select Configuration**.
- This will launch the target configuration in the debug window.
- Select the Texas Instruments XDS110 USB Debug probe/C674X_0 and press the **Connect Target** button 
- •Select the Texas Instruments XDS110 USB Debug probe/Cortex_R4_0and press the **Connect Target** button 



Click here to Connect to the target CPU

5. Loading the binary

- Once both targets are connected, select the C674X_0 target, and click on the **Load** button in the toolbar
- In the **Load Program** dialog, press the **Browse Project** button .
- Select the lab executable (.xe674) found in the project as shown, and press OK.
- Press OK again in the **Load Program** dialog.

5. Loading the binary

- Now select the Cortex_R4_0 target, and click on the **Load** button in the toolbar
- In the **Load Program** dialog, press the **Browse Project** button .
- Select the lab executable (.xer4f) found in the project and press OK.
- Press OK again in the **Load Program** dialog.

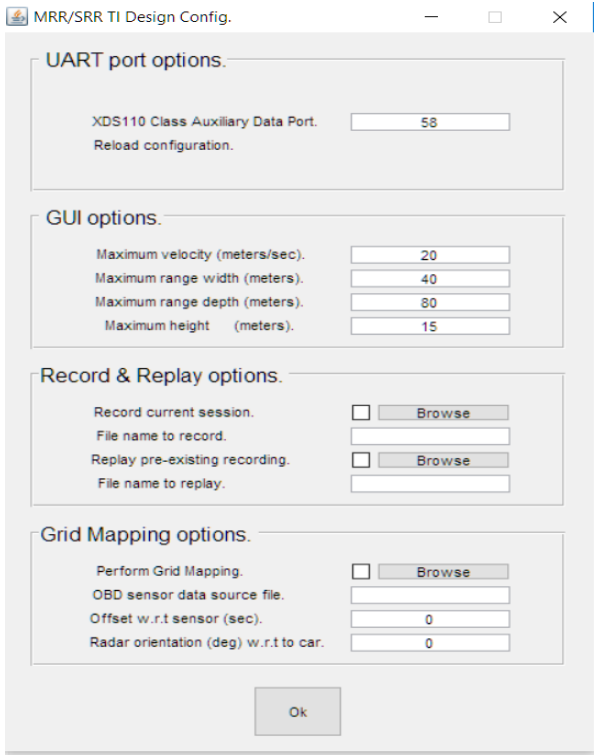
5. Running the binary

- With both executables loaded, group in CCS the two cores, ARM and DSP.
- Select the group and press the Run/Resume button
- The program should start executing and generate console output.
- If everything goes fine, you should see the
- “Debug: MMWave has been configured for MRR.”
- “Debug: Sensor will start momentarily.”

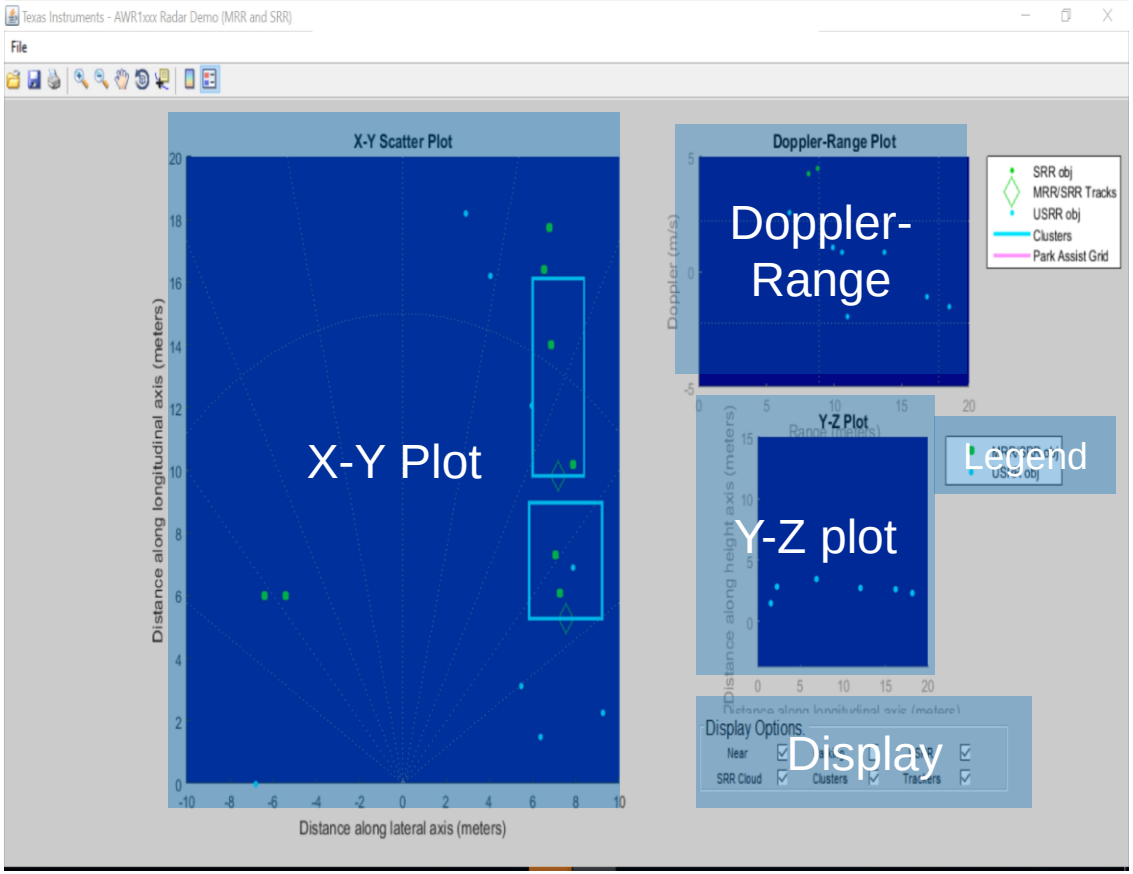
- Start the GUI as described on next slide

Running the Lab PC-GUI

1. Navigate to the folder **gui ► MRRvisualization** and click on MRRVISUALIZER**.exe**
2. Windows should open i.e. a Display prompt window and a GUI window. If the EVM is connected to the PC, then the display prompt window should successfully open the COM ports (to double check, make sure they match with the port numbers on the Device Manager).
3. In the GUI window, fill in the **Data COM Port** field. (Make sure that no other EVM is connected to the USB ports of the PC)
4. After filling all the options. Click “Ok”



Running the Lab PC-GUI



The Matlab GUI consists of 5 components

- **X-Y scatter plot** – Displays the positions of the point clouds, the tracks, and the cluster.
- **Y-Z scatter plot** – Displays the positions of the point clouds in elevation.
- **Doppler range plot** – Displays the Doppler-range coordinates of the point cloud and the clusters.
- **Legend** – Description of the different kinds of points being displayed on the screen.
- **Display options** - different types of the point cloud can be enabled and disabled at the user's discretion during demonstration.

Learn more about TI mmWave Sensors

- Learn more about xWR1x devices, please visit the product pages
  - AWR1443: http://www.ti.com/product/AWR1443
  - AWR1642: http://www.ti.com/product/AWR1642
  - AWR1843: http://www.ti.com/product/AWR1843
- Get started evaluating the platform with xWR1x EVMs, purchase EVM at
  - AWR1443 EVM: http://www.ti.com/tool/AWR1443BOOST
  - AWR1642 EVM: http://www.ti.com/tool/AWR1642BOOST
  - AWR1843 EVM: http://www.ti.com/tool/AWR1843BOOST
- Download mmWave SDK @ http://www.ti.com/tool/MMWAVE-SDK
- Ask question on TI's E2E forum @ http://e2e.ti.com