TI mmWave Labs

Vital Signs Measurement

(Version 1.4)

NOTE: ES3.0 devices only
This version of the mmWave Demo lab will work only with xWR1443BOOST ES3.0 EVMs, which require **mmWave SDK version 2.1 or above**. Please look at the next slide for information on identifying the version of the device on your EVM.

To download past versions of mmWave Industrial Toolbox which support ES2.0 EVMs, please follow the directions provided under **How to access previous Industrial Toolbox versions** at the bottom of the Industrial Toolbox landing page

Contents

- Overview
- Requirements
- Software setup
  - Pre-requisites
  - Downloading the Lab Project
  - Building the project
- Hardware setup
  - Preparing the EVM
  - Connecting the EVM
- Running the Lab
  - Loading Program
  - Running the GUI

Lab Overview

- This lab exercise demonstrates the ability of TI-IWR 14xx mmWave sensor to measure chest displacements due to breathing and heart beat
- Typical vital signs parameters for adults

Vital Signs	Amplitude	Frequency
Breathing Rate (Adults)	1- 12 mm	0.1 – 0.5 Hz
Heart Rate (Adults)	0.1 – 0.5 mm	0.8 – 2 Hz

- To measure these small scale vibrations/displacements, we measure the change in phase of the FMCW signal with time at the target range bin

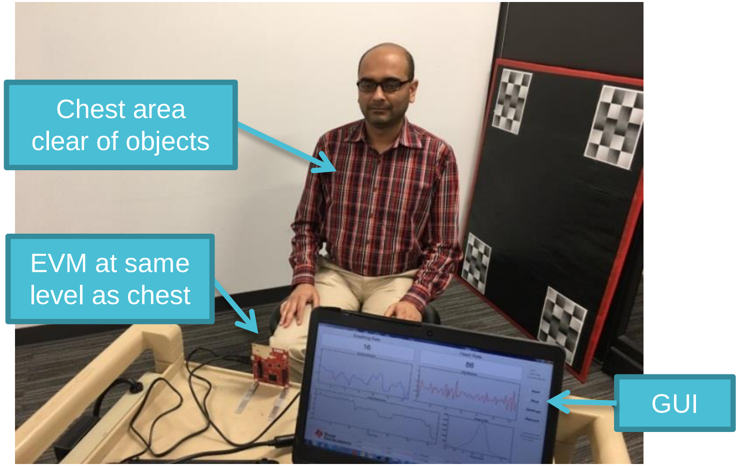
$$\Delta\phi_b = \frac{4\pi}{\lambda}\Delta R$$

- $\Delta\phi_b$ corresponds to the change in phase when the target moves a distance ΔR
- Note that a smaller wavelength λ will give better displacement sensitivity

- Code Composer Studio (CCS) project along with source code is provided for this lab

Lab Overview

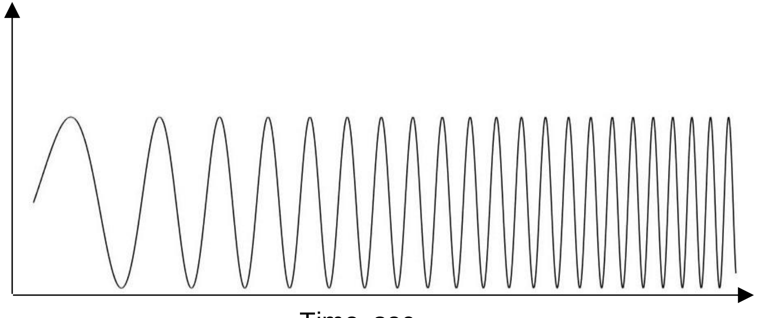
- TI mmWave sensor measures the chest displacement when it is pointed towards the chest of a person sitting in front of the sensor
- The onboard programmable processing cores on the TI IWR 14xx mmWave sensor are used to filter out the breathing and heart beat pattern from chest displacements and estimate the Breathing and Heart-rate
- GUI displays the chest displacements, filtered waveforms and the estimated Breathing and Heart-rate

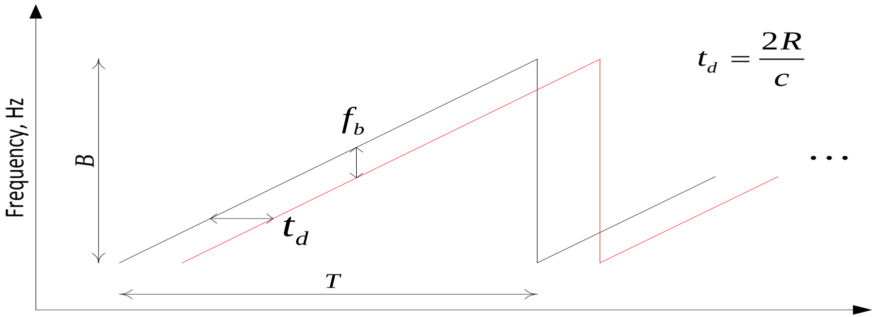


Note: Although not an absolute requirement, a lens in front of the sensor will improve the performance

FMCW Radar Basics

- Periodic linearly-increasing frequency chirps (known as **Frequency-Modulated Continuous Wave (FMCW))** are transmitted by radar towards the object





- Transmitted FMCW signal is given by $s(t) = e^{j\left(2\pi f_c t + \pi \frac{B}{T} t^2\right)}$
- Signal at the receiver is a delayed version of the transmitted signal $r(t) = e^{j\left(2\pi f_c (t - t_d) + \pi \frac{B}{T} (t - t_d)^2\right)}$
- The beat signal *b(t)* from an object at range R after mixing and filtering is given by

$$b(t) = s'(t) r(t) \approx e^{j\left(4\pi \frac{BR}{cT} t + \frac{4\pi}{\lambda} R\right)} = e^{j(f_b t + \phi_b)}$$

FMCW Radar – Vital signs Measurements

- Note that for a single object, the beat signal *b(t)* is a sinusoidal and has both frequency f_b and phase $\emptyset_b$

$$b(t) = e^{j\left(\underbrace{4\pi\frac{BR}{cT}}_{f_b}t + \underbrace{\frac{4\pi}{\lambda}R}_{\phi_b}\right)} = e^{j(f_b t + \phi_b)}$$

- To measure small scale vibrations, we measure the change in phase of the FMCW signal with time at the object range bin. If an object moves a distance ΔR then the change in phase between consecutive measurements is given by

$$\Delta\phi_b = \frac{4\pi}{\lambda}\Delta R$$

As an example at λ=4 mm when we have displacements as small as ΔR = 1mm, the corresponding phase change is $\Delta\phi_b = \pi$

- Phase can be measured by taking the FFT of the beat signal *b(t)* and computing the phase at the object range-bin.

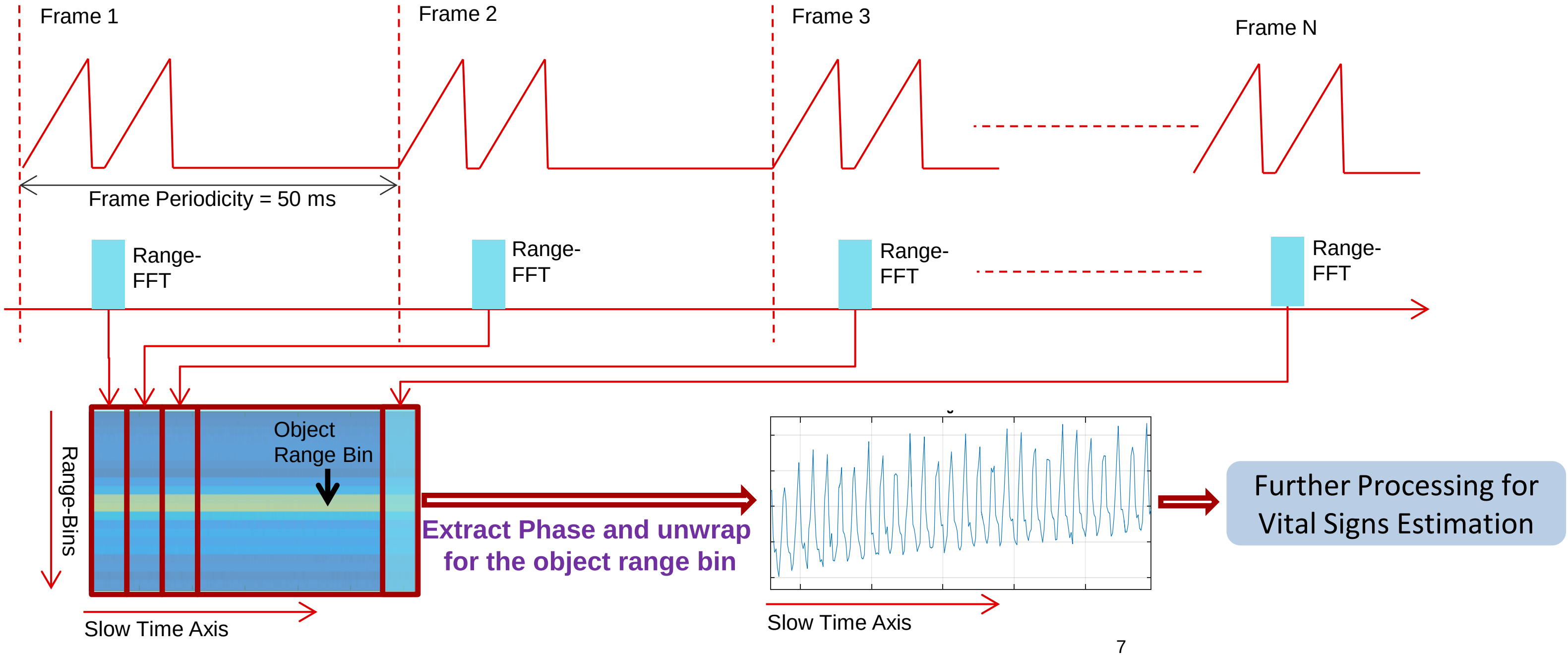
- Suppose we take the FFT and the object is at range-bin *m,* then the vibration signal *x(t)* can be extracted by measuring the phase at range-bin *m* at time indices nT_s , where *n* is the chirp index and T_s is the time between consecutive measurements

$$x(m, nT_s) = \frac{\lambda}{4\pi}\phi_b(m, nT_s)$$

Note that we are assuming that the vibrations *x(t)* are small so that the object remains in the same range-bin during the duration of the measurements

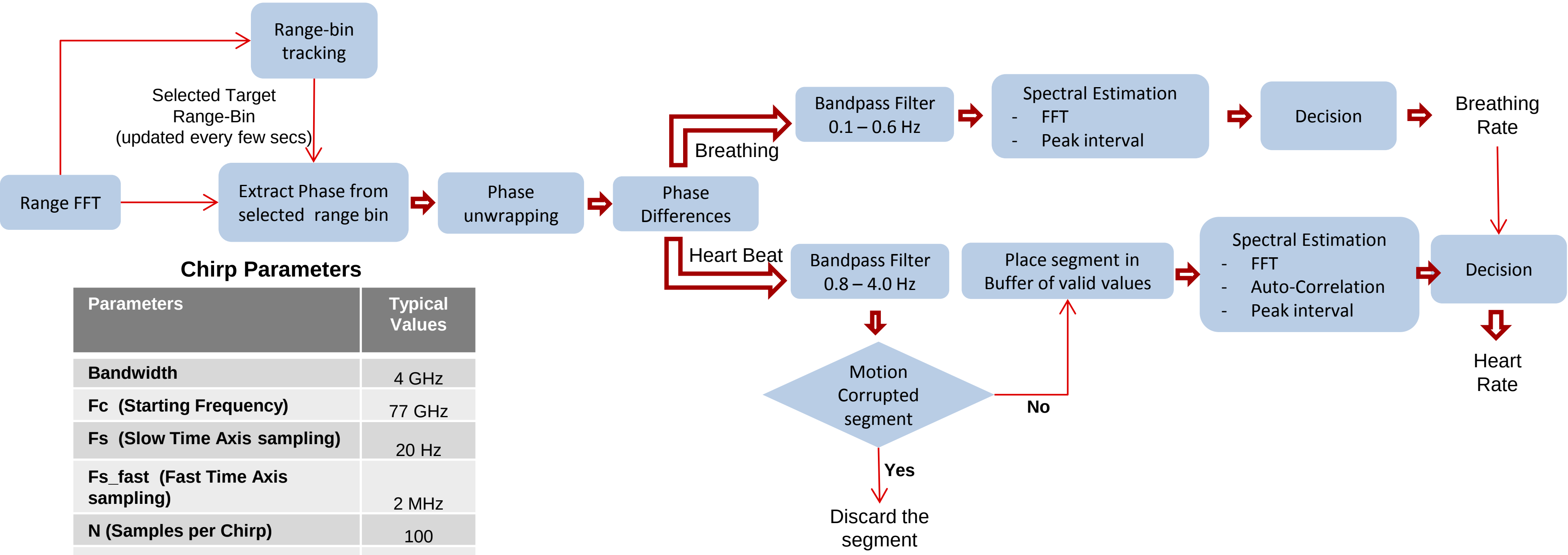
Chirp Configuration for Demo

- 100 ADC Samples per chirp. Chirp duration is 50 μs based on the IF sampling rate of 2 MHz
- Each chirp is configured to have 2 chirps. However only the 1st Chirp in the frame is used for processing
- A single TX-RX antenna pair is currently used for processing (Although all the RX antennas are enabled)
- Vital signs waveform is sampled along the “slow time axis” hence the vital signs sampling rate is equal to the Frame-rate of system



Implementation on the IWR-1443

- Real-time implementation (20 fps) on the C674x DSP Processing Core
- Processing done over a running window of *T* ~ 16 seconds. New estimates are updated every 1 second
- Memory Requirements ~ 16 kB, CPU Processing time for a single estimate ~ 4 ms



Chirp Parameters

Parameters	Typical Values
Bandwidth	4 GHz
Fc (Starting Frequency)	77 GHz
Fs (Slow Time Axis sampling)	20 Hz
Fs_fast (Fast Time Axis sampling)	2 MHz
N (Samples per Chirp)	100
T_c (Chirp Duration)	50 us

Block Diagram Description

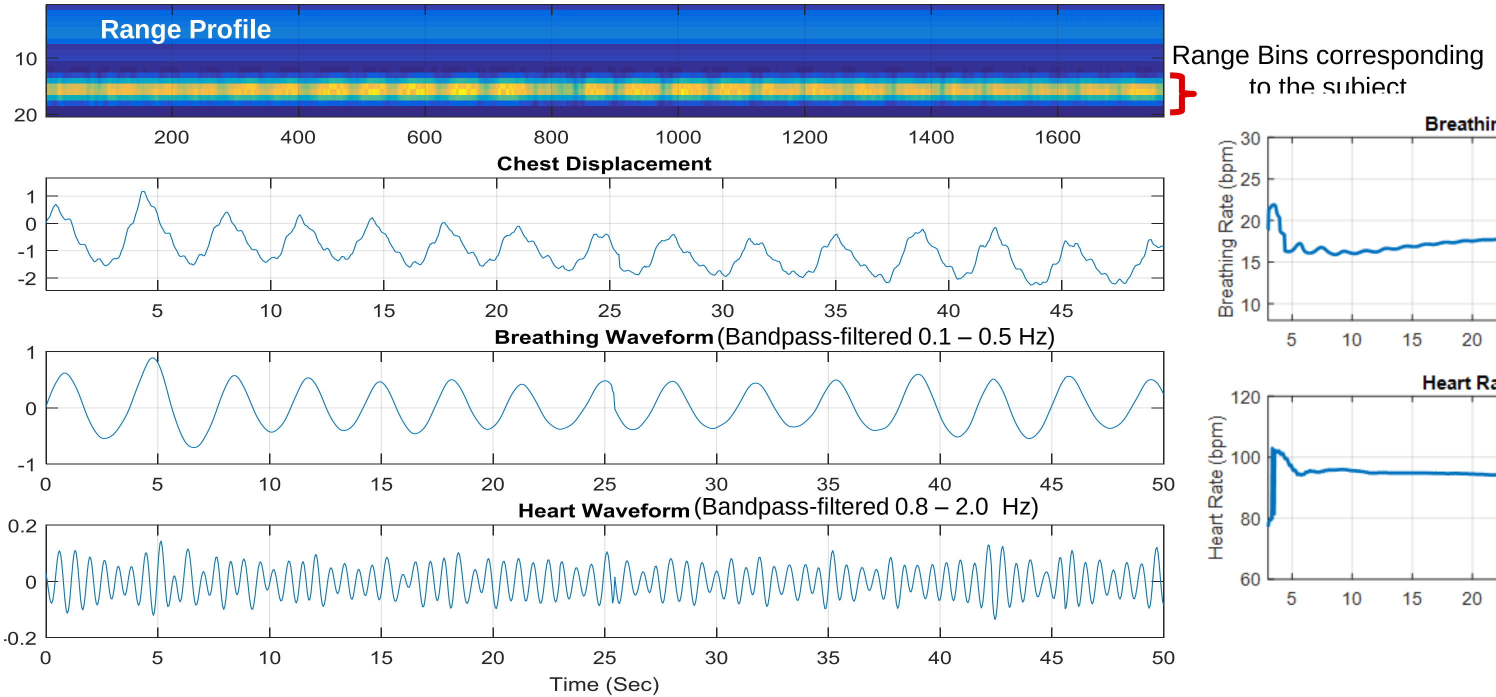
- **Range FFT :** A Fast Fourier Transform (FFT) is performed on the ADC data to obtain the range profile. The magnitude of the range-profile is displayed on the PC-GUI.
- **Target Range-bin :** The range-bin corresponding to the target is found by finding the max-value in the range profile within the user-specified range limits.
- **Phase Extraction :** The phase value of the selected range-bin is computed from the complex range profile data and these phase values are measured over time. The assumption is that the subject is in the same range-bin throughout these measurements. If the subject moves to a different range-bin then it will take a few seconds before the algorithm locks into the new target range-bin.
- **Phase Unwrapping :** Phase values are between [-π, π] and need to be unwrapped to obtain the actual displacement profiles. Phase unwrapping is performed by adding/subtracting 2π from the phase whenever the phase difference between consecutive values is greater/less than ± π. The unwrapped phase is displayed as the chest displacement is the PC-GUI.
- **Phase Difference :** The phase difference operation is performed on the unwrapped phase by subtracting successive phase values. This helps in enhancing the heart-beat signal and removing any phase drifts.
- **Impulsive Noise Removal :** The un-wrapped differential phase might be corrupted by several noise-induced phase wrapping errors. This impulse-like noise is removed by computing a forward a(m)–a(m+1) and backward a(m)-a(m-1) phase difference for each a(m) and if these exceed a certain threshold then a(m) is replaced by an interpolated value.

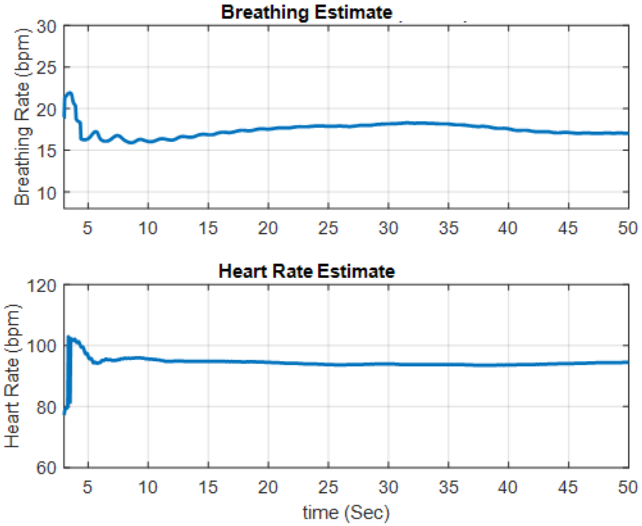
Block Diagram Description

- **Bandpass Filtering :** The phase values (after unwrapping and phase differences) are passed through two band-pass filters (serially-cascaded Bi-Quad IIR filter). These band-pass filters operate in real-time input data to generate a continuous stream of output data. The data after band-pass filtering is displayed as the breathing waveform and heart waveform in the PC-GUI.
- **Motion corrupted segment removal/Gain Control :** The purpose of this block is to reduce the impact of any large amplitude movements on the heart-rate estimates. The waveform is divided into segment of $L=20$ samples (corresponding to 1 sec). If the energy within this data segment exceeds a user-defined threshold ($E > E_{Th}$), then all the samples in that data segment/block are either scaled by $\sqrt{E_{Th}/E}$ or are alternatively discarded from the time-domain cardiac waveform.
- **Vital Signs Waveforms :** Band-pass filter outputs are stored in the breathing-waveform and cardiac-waveform buffer. Pre-processing steps such as windowing, gain control can be done on these prior to spectral estimation.
- **Spectral Estimation :** These buffers are passed on to the spectral estimation block. Several different types of spectral estimation techniques can be implemented. The current implementation provides a FFT, auto-correlation and an estimate based on the inter-peak distances in the time-domain waveforms to estimate the vital signs.
- **Vital Signs Decision :** The final heart-rate and breathing-rate decisions are made based on the confidence metric from different spectral estimation methods.

Example Measurements

- Subject seated at a distance of 1.5 meters form the Radar and was asked to remain stationary

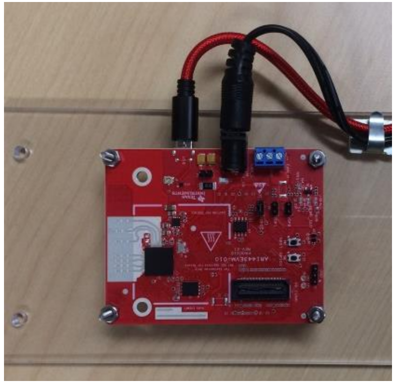




1. Requirements

- Software
  - **Pre-requisites**
    - Latest TI mmWave SDK and all related dependencies installed as mentioned in the mmWave SDK release notes.
  - Vital Signs Lab CCS Project
    - Download from TI Resource Explorer
  - UniFlash
    - For flashing firmware images onto
    - Download from TI.com/tool/uniflash
  - XDS110 Drivers
    - For EVM XDS device support
    - Included with CCS Installation, or standalone through TI XDS Emulation Software
  - MATLAB runtime R2016b (9.1)
    - For running the Vital Signs Lab GUI
    - Download from MATLAB website

- Hardware
  - AWR14xx/IWR14xx EVM ES3.0
  - Micro USB cable (included in the EVM package)
  - 5V/2.5A Power Supply
    - Purchase from Digikey
  - A lens/concentrator to direct the radar waves towards the chest



Steps

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

1. Pre-requisites

1. Install Pre-requisites | 2 | 3 | 4 | 5

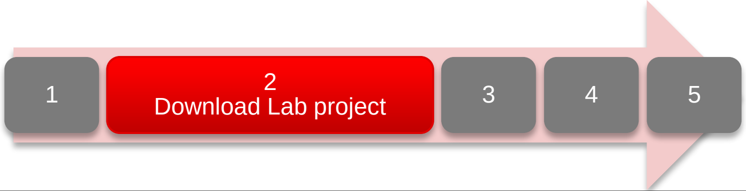
- It is assumed that you have the TI mmWave SDK 2.1.0.4 and all the related tools installed as mentioned in the mmWave SDK release notes.
  - The mmWave SDK release notes include the links for downloading the required versions of the above tools.
  - Helpful Tips
    - Please make sure that any existing PERL installations are removed from the PC before installing the version of PERL listed in the SDK release notes
    - After you've downloaded and saved **CRC.pm**, locate the saved file and remove the .txt extension if it is there. Please ensure that the file has a **.pm** extension and not a .txt extension at the end
    - XDC tools are provided as a zip file which needs to be extracted in the TI install directory (typically C:\ti)
- 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.

Tool	Version	Download link
CCS	7.1 or later	download link Please note that CCS v7.1 or later is mandatory. CCSv6.x cannot be used
TI SYS/BIOS	6.52.00.12	Included in mmwave sdk installer
TI ARM compiler	16.9.1.LTS	Included in mmwave sdk installer
TI CGT compiler	8.1.3	Included in mmwave sdk installer
XDC	3.50.00.10	Included in mmwave sdk installer
C64x+ DSPLIB	3.4.0.0	Included in mmwave sdk installer
C674x DSPLIB	3.4.0.0	Included in mmwave sdk installer
C674x MATHLIB (little-endian, elf/coff format)	3.1.2.1	Included in mmwave sdk installer
Mono JIT compiler	3.2.8	Only for Linux builds
mmwave device support packages	1.5.3 or later	Upgrade to the latest using CCS update process (see SDK user guide for more details)
TI Emulators package	6.0.0576.0 or later	Upgrade to the latest using CCS update process (see SDK user guide for more details)

Steps

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

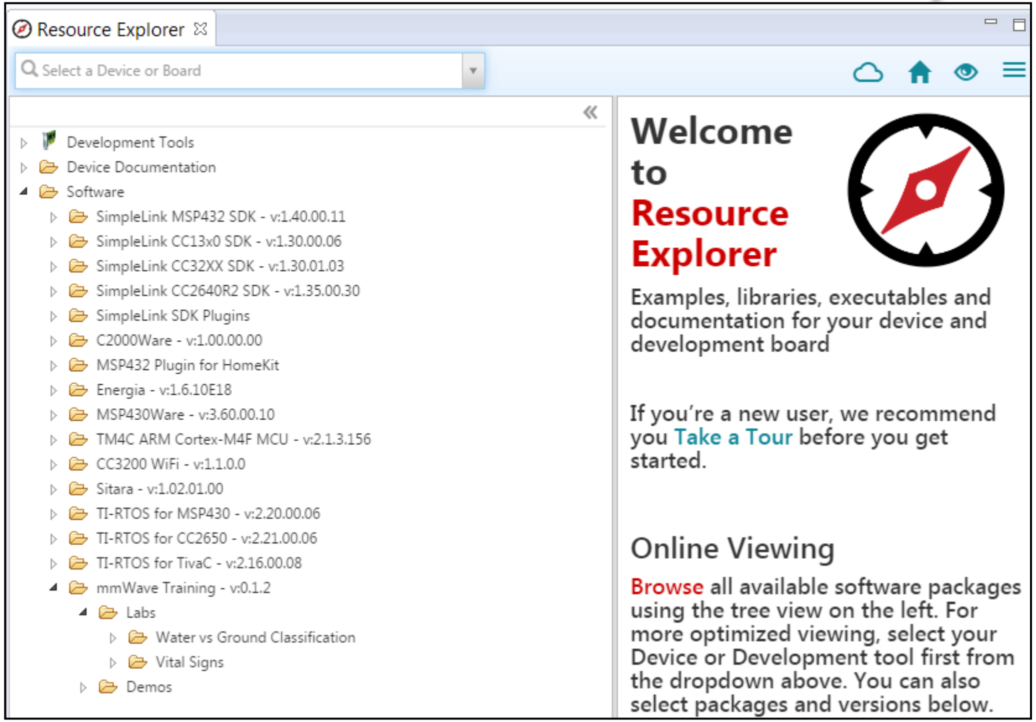
2. Import Lab project



- The mmWave Lab projects are available under **mmWave Training** in CCS Resource Explorer.

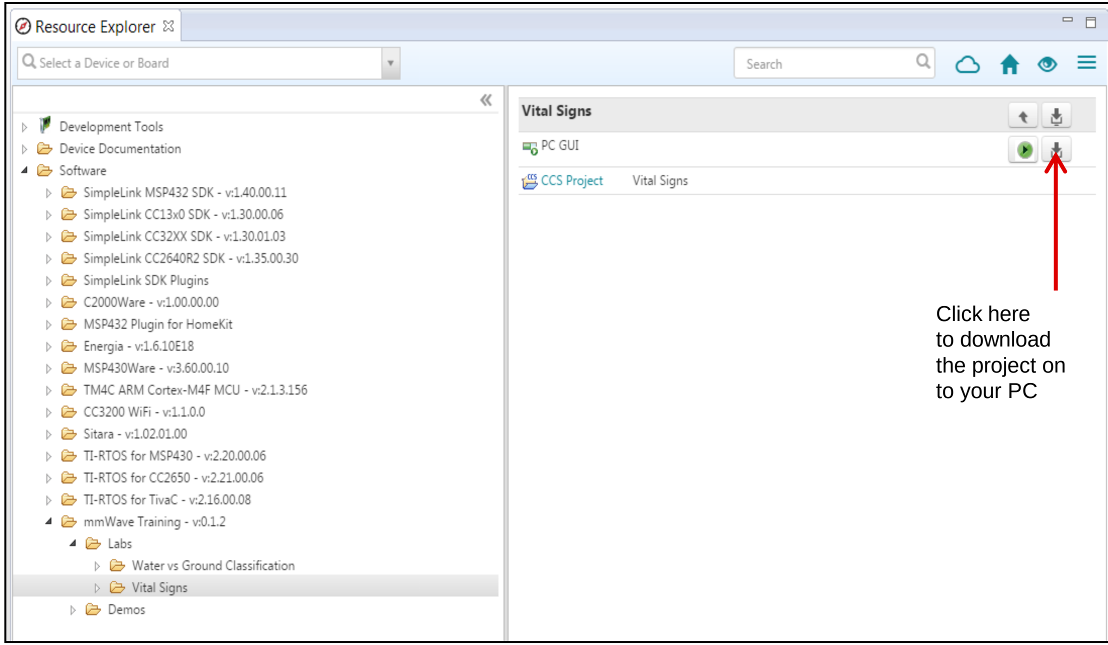
- To download the Vital Signs Lab, start CCS v7.1 (or later) and select **View ► Resource Explorer** to open the Resource Explorer.

- In the Resource Explorer Window, select **Software ► mmWave Training ► Labs**.



2. Import - continued

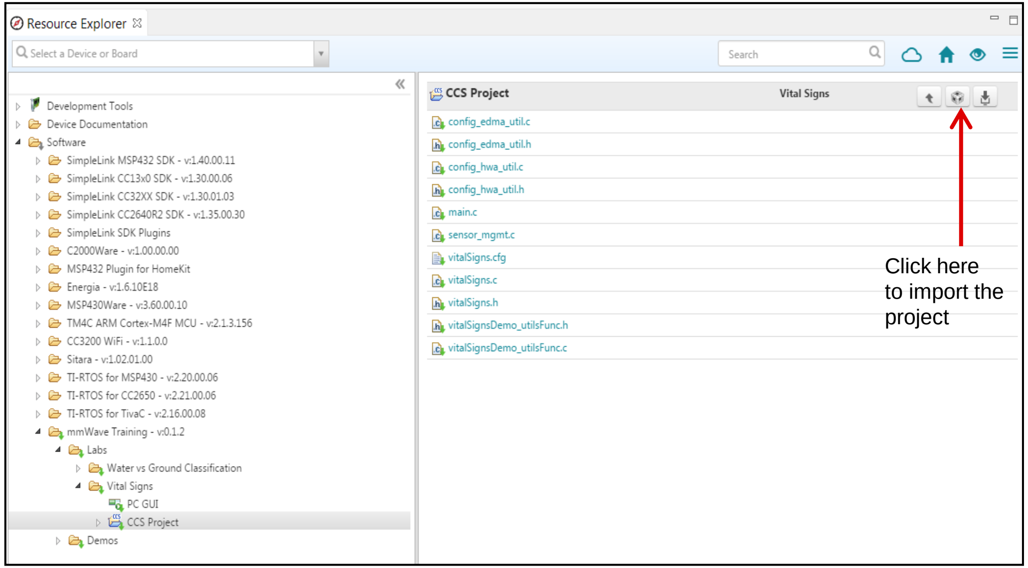
- Select the Vital Signs Lab in the left view.
- The right view shows the contents of the Lab which contains the CCS Project and the PC GUI.
- Click on the **Download and Install** button  in the top right corner as shown.
- Select the **Make Available Offline** option from the drop down to start downloading the Lab.



Click here to download the project on to your PC

2. Import - continued

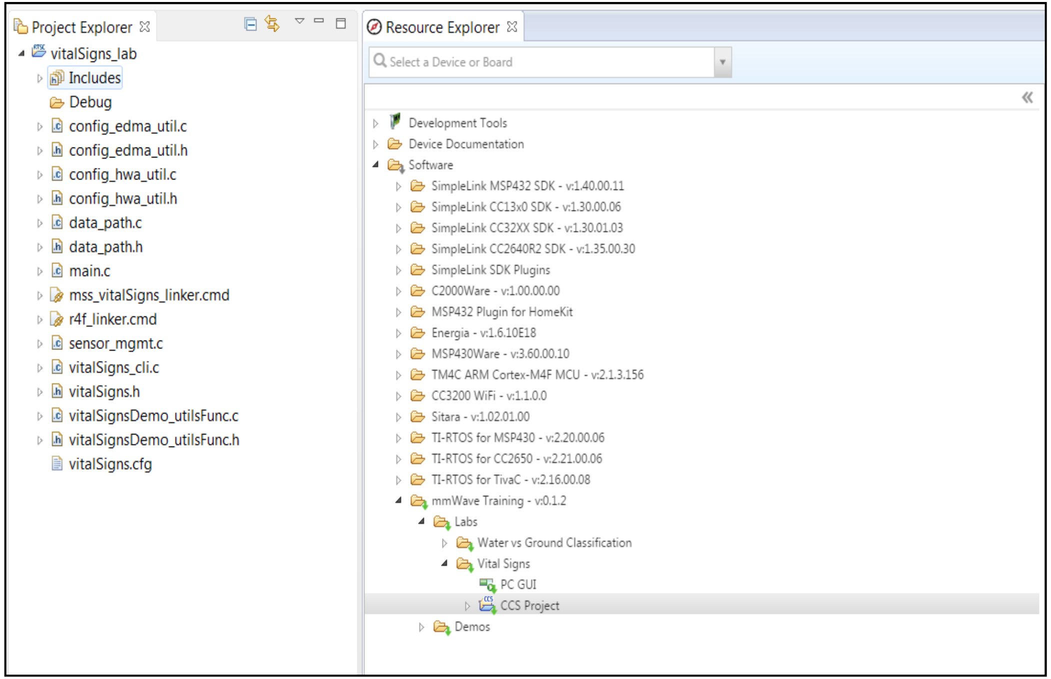
- The project will be downloaded in C:\ti\mmwave_training
- Select the CCS project in the left view as shown.
- Click on the **Import to IDE** button which should be visible in the right side view after a successful download.
- This copies the project in the user's workspace and imports it into the CCS 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_training is not touched.



2. Download - continued

1 | 2 Download Lab project | 3 | 4 | 5

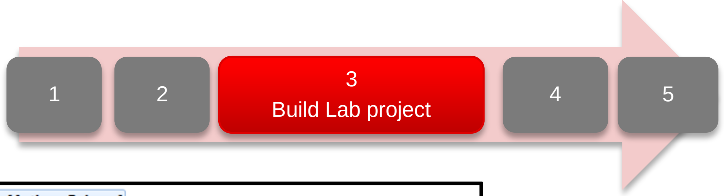
- After successfully completing the **Import to IDE** operation, the project should be visible in CCS Project Explorer as shown here.
- At this point, we have successfully downloaded the Vital Signs Lab and imported it in CCS.
- We are ready to move on to the next step i.e. Building the project.



Steps

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

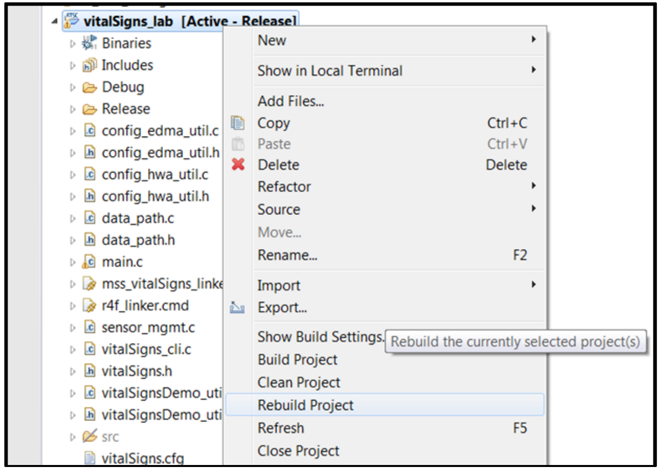
3. Build the Lab



- With the vitalSigns_lab 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
  - xwr14xx_vitalSigns_lab_mss.xer4f
  - xwr14xx_vitalSigns_lab_mss.bin

- If the build fails with errors, please ensure that all the pre-requisites are installed as mentioned in the mmWave SDK release notes.
  - Please not the the pre-built binary files , both .xer4f and .bin, are provided with the lab in the prebuilt_binaries directory
  - Look under C:\ti\mmwave_training_<version>\labs\lab0002-vital-signs\lab0002_vital_signs_pjt\Prebuilt_binaries



```
C:/ti/mmwave_sdk_02_01_00_04/packages/scripts/ImageCreator/crc_multicore_image/crc_multicore_image.exe xwr14xx_vitalSigns_lab_mss.bin
xwr14xx_vitalSigns_lab_mss.tmp
size of App Image is 128320 bytes
cur_crc_read_addr 128
cur_crc_read_addr 256
cur_crc_read_addr 96064
Failed to remove CRC temp file

C:/ti/mmwave_sdk_02_01_00_04/packages/scripts/ImageCreator/append_bin_crc/gen_bincrc32.exe xwr14xx_vitalSigns_lab_mss.bin
>>>> Binary CRC32 = 940ac973 <<<<
>>>> Total bytes in binary file 128324 <<<<



**** Build Finished ****
```

Steps

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

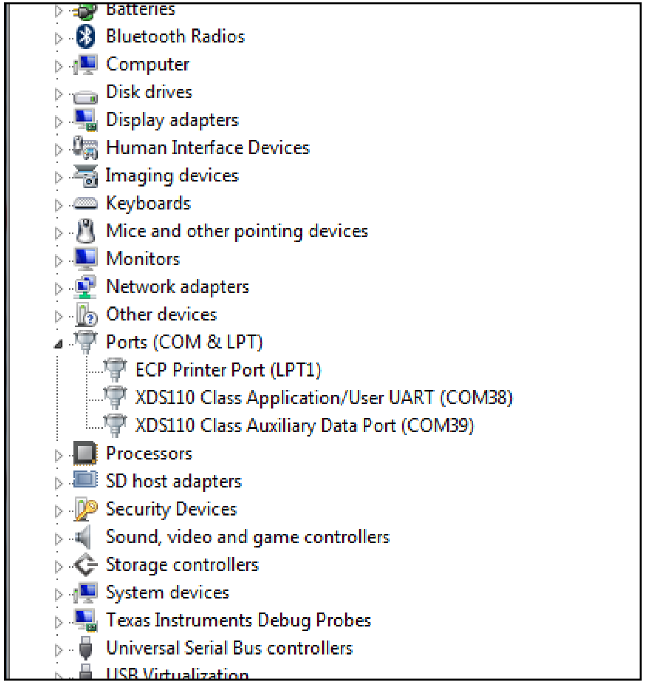
4.1 Preparing the EVM

- 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.
  - As a recap, the build process in Step 3 produces both the .bin and .xer4f images.

- This 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 note that the same flashing process can be used to flash the Lab binary to run it in deployment mode.

4.2 Connecting to the EVM

- Power on the EVM using a 5V/2.5A power supply.
- Connect the EVM to your PC and check the COM ports in Windows Device Manager
- The EVM exports two virtual COM ports as shown below:
  - XDS110 Class Application/User UART (COM_{UART}):
    - Used for passing configuration data and firmware to the EVM
  - XDS110 Class Auxiliary Data Port (COM_{AUX})
    - Used to send processed radar data output
- Note the COM_{UART} and COM_{AUX} port numbers, as they will be used later for flashing and running the Lab.

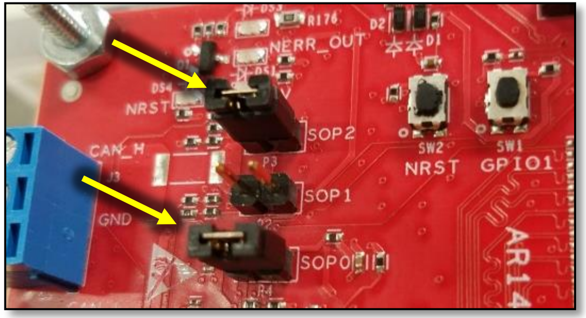


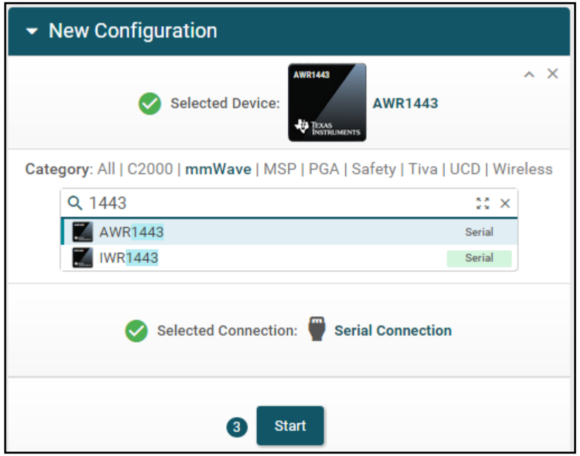
COM_{UART}: COM38 **COM_{AUX}**: COM39

- The actual port numbers on your machine may be different

4.3 Flashing CCS debug firmware

1. Put the EVM in flashing mode by connecting jumpers on SOP0 and SOP2 as shown in the image.
2. Open the **UniFlash** tool
3. In the **New Configuration** section, locate and select the appropriate device (AWR1443 or IWR1443)
4. Click **Start** to proceed





4.3 Flashing CCS debug firmware

1. In the **Program** tab, browse and locate the MSS imageshown below:

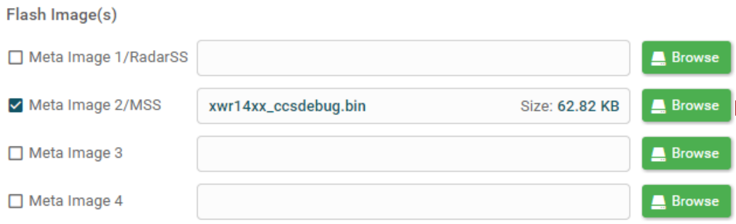


Image	Location
Meta Image 2/MSS	C:\ti\mmwave_sdk_<ver>\packages\ti\utils\ccsdebug**xwr14xx_ccsdebug_mss.bin**

2. In the **Settings & Utilities** tab, fill the **COM Port** text box with the Application/User UART COM port number (**$\mathbf{COM_{UART}}$**) noted earlier

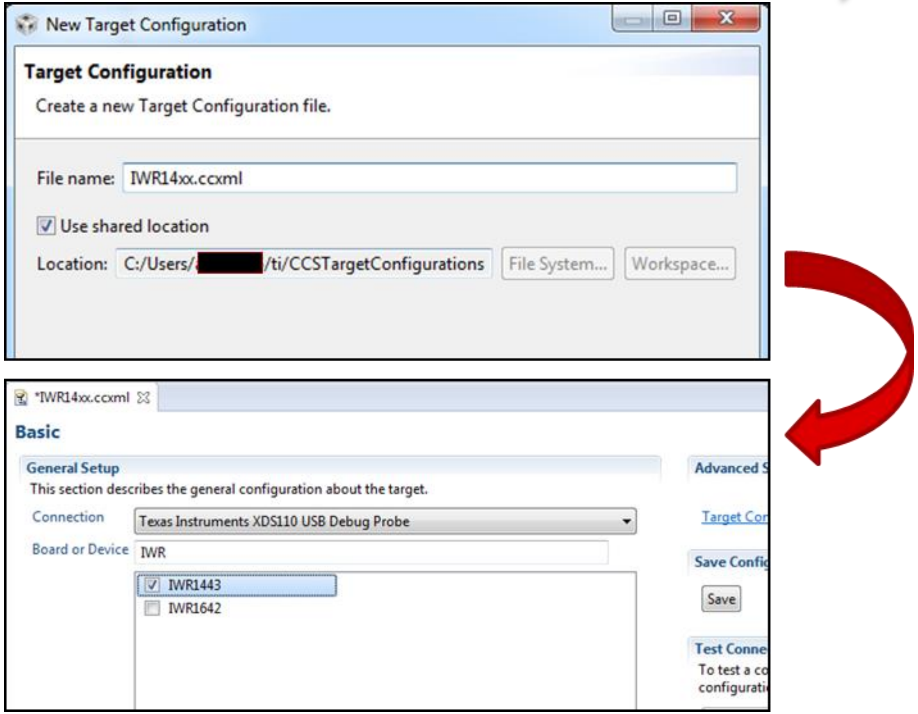
3. Return to the **Program** tab, power cycle the device and click on **Load Images**
4. When the flash procedure completes, UniFlash's console should indicate: [SUCCESS] Program Load completed successfully
8. Power off the board and remove the jumper from only header **SOP2** (this puts the board back in functional mode)

Steps

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

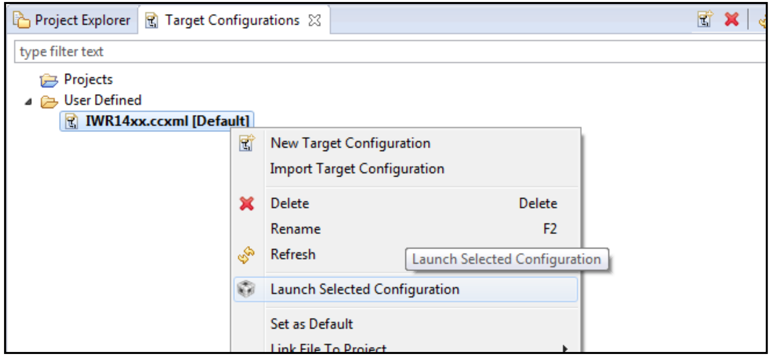
5.1 Connecting EVM to CCS

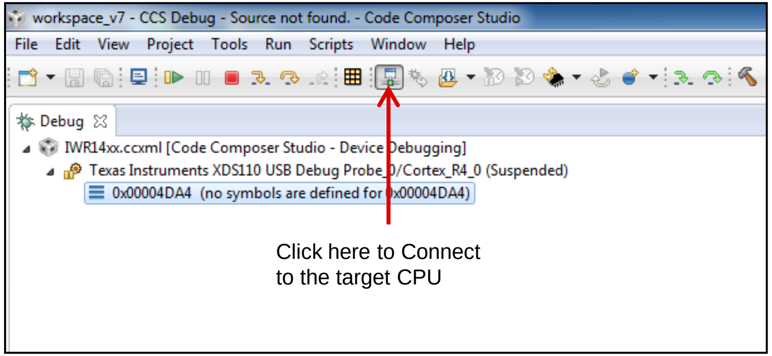
- It is assumed that you were able to download and build the Lab in CCS (completed steps 1, 2 and 3)
- To connect the Radar EVM to CCS, we need to create a target configuration
  - Go to File ► New ► New Target Configuration File
  - Name the target configuration accordingly and check the “Use shared location” checkbox. Press Finish
  - In the configuration editor window:
    - Select “Texas Instruments XDS110 USB Debug Probe” for **Connection**
    - Type **IWR** in the **Board or Device** text box and select **IWR1443** device.
    - Press the **Save** button to save the target configuration.
    - You can press the **Test Connection** button to check the connection with the board.



5.1 Connecting - continued

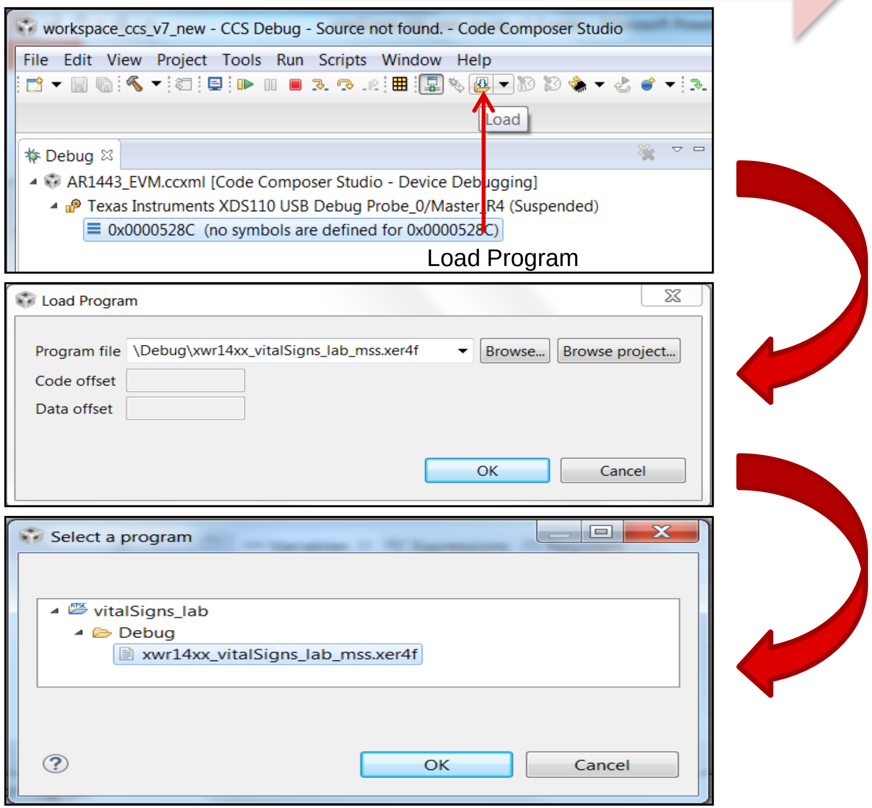
- Go to **View ► Target Configurations** to open the target configuration window.
- You should see your target configuration under **User Defined** configurations.
- Right click on the target configuration and select **Launch Selected Configuration**.
- This will launch the target configuration in the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





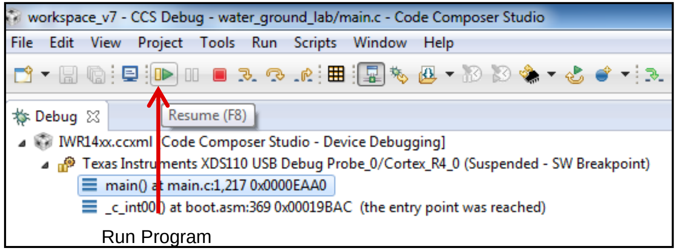
5.2 Loading the binary

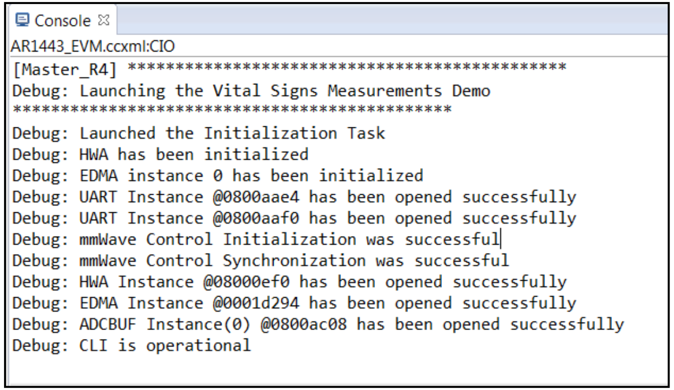
- With the target connected, click on the **Load** button in the toolbar.
- In the **Load Program** dialog, press the **Browse Project** button .
- Select the lab executable (.xer4f) as shown and press OK.
- Press OK again in the **Load Program** dialog.



5.3 Running the binary

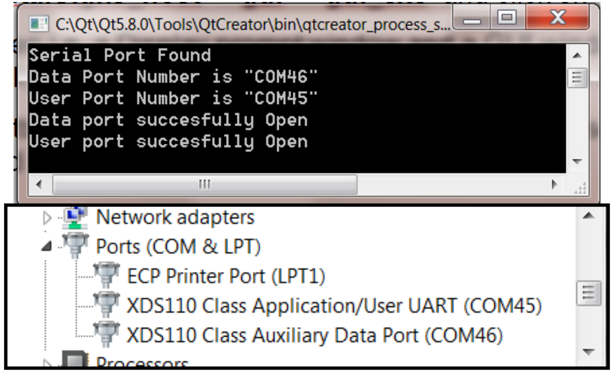
- With the executable loaded, press the Run/Resume button
- The program should start executing and generate console output as shown.
- If everything goes fine, you should see the “CLI is operational” message which indicates that the program is ready and waiting for the sensor configuration.
- The sensor configuration is sent using the Lab GUI which is based on Matlab.
  - Note: Please ensure that MATLAB runtime R2016b (9.1) is installed as mentioned in the pre-requisites section.





5.4 Running the Lab GUI

1. Navigate to the folder **vitalSigns_host ► gui ► gui_exe** and click on **VitalSignsRadar_Demo.exe**
2. Two 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, the **User UART COM Port** and **Data COM Port** fields should automatically be filled with the correct port numbers (Make sure that no other EVM is connected to the USB ports of the PC)



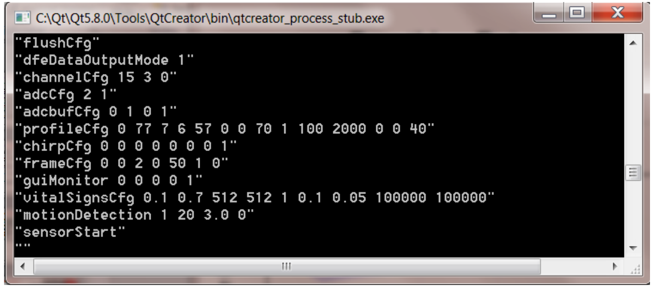
COM ports (The GUI should automatically fill these fields)

If the GUI does not open you might need the vc runtime which can be downloaded from the link below
https://www.microsoft.com/en-us/download/details.aspx?id=48145

Texas Instruments

5.4 Running the Lab PC-GUI

1. Press the **Start** Push button in the GUI. In the Display Prompt window you should see the configuration settings being read from the configuration text file and sent through the UART to the EVM
2. As soon as the **sensorStart** command is sent, the GUI should start displaying the data
3. Please follow the Getting Started Guide for more details on placing the sensor and starting the GUI.



5.4 Running GUI - continued

- Have the subject sit comfortably on the chair. As sub-mm chest displacements are being measured the subject is required to be **very still** for accurate measurements
- Make sure that a peak corresponding to the subject can be seen in the Range Profile plot.
- Now, chest displacements due to breathing should be clearly visible in the **Breathing Waveform** plot
- Once a few chest displacements have been seen, ask the subject to hold their breath. The breathing-rate should go to zero and turn red, the breathing waveform plot should be more or less constant and the heart rate waveforms should still be **visible**. If the breathing rate does not go to zero OR the heart rate waveform is not visible then either the subject is not properly aligned with the radar or there is interference coming from other moving objects within the Radar field-of-view.
- Wait 20-30 seconds so that enough data frames are received for an accurate estimate of the vital signs

Normal Breathing

Holding Breath

5.4 Running GUI – (Configuration)

- **vitalSignsCfg** can be modified to change the algorithm parameters

Sample Configuration File

```
dfeDataOutputMode 1
channelCfg 15 1 0
adcCfg 2 1
adcbufCfg 0 1 0 1
profileCfg 0 77 7 6 57 0 0 70 1 100 2000 0 0 40
chirpCfg 0 0 0 0 0 0 0 1
frameCfg 0 0 2 0 50 1 0
guiMonitor 0 0 0 1
vitalSignsCfg 0.3 1.0 256 512 0.1 0.1 0.05 300000 300000
sensorStart
```

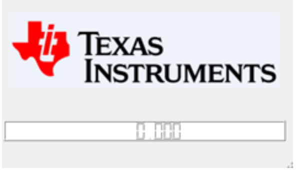
Configuration	Parameters	Values	Comments
vitalSignsCfg	Start Range (meters)	0.3	The subject/person is expected to be within the Start Range and End Ranges. The program searches for the maximum peak within these ranges and assumes that peak corresponds to the subject
	End Range (meters)	1.0	
	Breathing Waveform Size	256	Specifies the number of points within the waveforms. As an example, given a frame-rate of 20 Hz and 256 number of samples in the waveform then the time duration of the waveform would be = 256/20 ~ 12.8 seconds. In general, larger the time duration, better the frequency resolution after the FFT and higher the FFT processing gain. However, due to the inherent time-frequency resolution tradeoff, we lose the ability to measure instantaneous changes in the heart-rate and breathing-rate if large waveform sizes are used.
	Heart-rate Waveform Size	512	
	Threshold for Gain Control	0.1	Only applies to the cardiac waveform in the current version. If the energy of a segment of the cardiac waveform exceeds this threshold, then that segment is appropriately scaled to decrease the energy level
	Alpha filter value for Breathing waveform energy computation	0.1	Alpha filter values for recursive averaging of the waveform energies based on the equation below where x(n) is the current waveform value while E(n) is the energy. $E(n) = \alpha x^2(n) + (1-\alpha)E(n-1)$
	Alpha filter value for heart-beat waveform energy computation	0.05	
	Scale Factor for breathing waveform	300000	Scaling factors to convert waveform values in floating points to 32 bit integers required by the FFT accelerator. Typical values when measuring vital signs from the front would be 10000 while higher values (e.g. 300000) might be required when measuring the vital signs from the back of a person due to much smaller displacement amplitudes.
	Scale Factor for heart-beat waveform	300000	

5.4 Running GUI – (Configuration)

- **motionDetection**

The purpose of this block is to discard the data segments that might be corrupted by large amplitude movements. The heart waveform is divided into segment of L samples. If the energy within this data segment exceeds a user-defined threshold E_{Th} then all the samples are discarded from the time-domain heart waveform.

Configuration	Parameters	Values	Comments
motionDetection	Enable	1	0: Disable the Block 1: Enable the Block
	Data segments Length (L)	20	Data segment over which the energy is computed
	Threshold (E_{TH})	0.04	Energy threshold value. If the energy in the data segment length exceeds this value then the data segment is discarded
	Gain Control	0	0: Disable Gain Control 1: Enable Gain Control



Data segment valid

Data segment discarded

Note : The display panel below the TEXAS INSTRUMENTS icon will turn RED if the current data segment is discarded due to motion corruption.

Limitations of the Demo

- The user has to be relatively still for the demo to effectively work
- The EVM must be level with the subject’s chest. Use an adjustable table or adjustable chair in order to have the EVM and chest at the same height.
- Any objects within the same radial distance, plus ~1m, as the subject can impact the measurements
- Any objects in front of the chest (badges, lanyards, necklaces, etc.) can interfere with the data
- The breathing and heart-rate band-pass filter have hard-coded lower and higher cut-off frequencies. For the breathing it is 6 – 30 beats per minute while for the heart-rate it is 48 – 120 beats per minutes. If any user has vital signs rate outside these ranges, the demo with not show correct results
- Bandpass-filter coefficients have been hard-coded based on a frame-rate of 20 fps. If a different frame-rate is specified than the band-pass filter performance will not be as required
- The heart-rate value might jump during measurements. This can be due to several reasons (e.g. noise, alignment issues, interference from other objects, breathing harmonics overlapping with the heart rate frequency etc.). If the subject stays stationary, the heart-rate values ultimately should converge to the correct value
- One reason the heart rate might display a wrong value is the presence of breathing harmonic overlapping the heart-rate spectrum region i.e. [0.8 – 2.0] Hz. In the current demo the 2nd breathing harmonic is cancelled. For example if the person has a breathing rate of 26 bpm and the heart rate happens to be ~ 52 bpm it will be discarded as the algorithm will interpret this as a breathing harmonic rather than a correct heart-rate

Output packet on UART

Packet Structure

Total packet size is multiple of 32 bytes

Header	40 Bytes
Message TLV Header	8 Bytes
Vital Signs Output Stats	88 Bytes
Message TLV Header	8 Bytes
Range Profile (Complex)	Number of Processed range bins *4 Bytes
Extra Padding	Extra padding packet size multiple of 32 bytes

Header

MagicWord (0x0102, 0x0304, 0x0506, 0x0708)	8 Bytes
version	4 Bytes
totalPacketLen	4 Bytes
platform	4 Bytes
frameNumber	4 Bytes
timeCpuCycles	4 Bytes
numDetectedObj	4 Bytes
numTLVs	4 Bytes
reserved	4 Bytes

Message TLV Header

Type	4 Bytes
Length	4 Bytes

Vital Signs Output Stats

Parameter	Type
rangeBinIndexMax	uint16
rangeBinIndexPhase	uint16
maxVal	float
processingCyclesOut	uint32
rangeBinStartIndex	uint16
rangeBinEndIndex	uint16
unwrapPhasePeak_mm	float
outputFilterBreathOut	float
outputFilterHeartOut	float
heartRateEst_FFT	float
heartRateEst_FFT_4Hz	float
heartRateEst_xCorr	float
breathingRateEst_FFT	float
breathingRateEst_peakCount	float
heartRateEst_peakCount_filtered	float
confidenceMetricBreathOut	float
confidenceMetricHeartOut	float
confidenceMetricHeartOut_4Hz	float
sumEnergyBreathWfm	float
sumEnergyHeartWfm	float
confidenceMetricHeartOut_xCorr	float
reserved	float
reserved	float
reserved	float

Learn more about TI mmWave Sensors

- Learn more about xWR1x devices, please visit the product pages
  - IWR1443: http://www.ti.com/product/IWR1443
  - IWR1642: http://www.ti.com/product/IWR1642
  - AWR1443: http://www.ti.com/product/AWR1443
  - AWR1642: http://www.ti.com/product/AWR1642
- Get started evaluating the platform with xWR1x EVMs, purchase EVM at
  - IWR1443 EVM: http://www.ti.com/tool/IWR1443BOOST
  - IWR1642 EVM: http://www.ti.com/tool/IWR1642BOOST
  - AWR1443 EVM: http://www.ti.com/tool/AWR1443BOOST
  - AWR1642 EVM: http://www.ti.com/tool/AWR1642BOOST
- Download mmWave SDK @ http://www.ti.com/tool/MMWAVE-SDK
- Ask question on TI’s E2E forum @ http://e2e.ti.com