

Example to enable raw data capture through DCA1000 CLI interface

Mmwave SDK and radar studio together provide a method to capture raw ADC data through LVDS without radar studio. The basic idea is to run SDK out of box demo on the target EVM. The out of box demo will detect point cloud and at the same time send out ADC raw data through LVDS.

At the same time users will use DCA1000 CLI control interface to control the DCA1000 to get data captured through LVDS lanes. Users can find some document on the SDK users guide on DCA1000 CLI control interface on section 3. 3. 2. mmWave demo with LVDS-based. The SDK users guide is located at (replace mmwave_sdk_xx_xx_xx_xx with the latest SDK version): C:\ti\mmwave_sdk_xx_xx_xx_xx\docs

Users can also find some guide on DCA1000 CLI interface radar studio package located at (replacing mmwave_studio_xx_xx_xx_xx with the latest radar studio version):

C:\ti\mmwave_studio_xx_xx_xx_xx\mmWaveStudio\ReferenceCode\DCA1000\Docs

The DCA1000 CLI interface is provided for windows, and can be recompiled for other platform. Advance users can also modify it to get more advance features.

In this package, we provide an example (through MATLAB) to use SDK out-of-box demo and DCA1000 CLI control interface to get the raw data captured through LVDS streaming. An MATLAB script controls (configure/start/stop) the sensor, communicate with DCA1000 board and parse the captured ADC raw data to plot the 2D FFT output. A sensor configuration file chirp1_60G.cfg is provided as an example to configure the sensor.

Here list the procedure to use this example.

1. Flash the out of box binary to the target.

Users can find two binaries here, one for xwr6843 ES1 device and one for xwr6843 ES2 device.

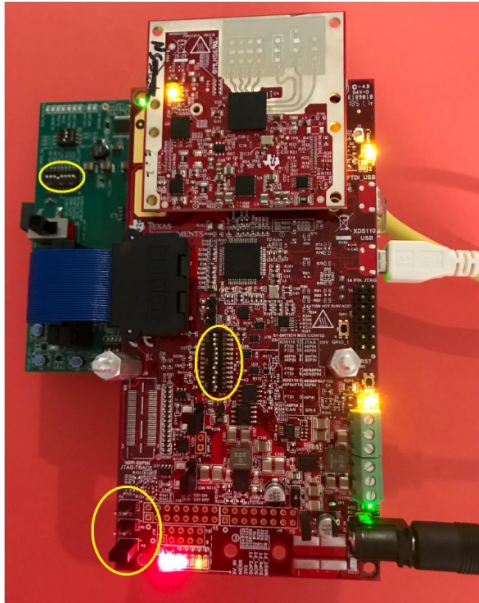
. \prebuild_binary\ xwr68xx_mmw_demo_ES1.bin

. \prebuild_binary\ xwr68xx_mmw_demo_ES2.bin

Same binary can be used for different antenna module of xwr6843 ES2 devices, such as ODS, ISK and AOP boards. These two binary files are copied from SDK out of box demo. Note that different version of binary also requires slightly different radar configuration file. Two examples are given accordingly by “chirp1_60G_ES1.cfg” and “chirp1_60G_ES2.cfg” under .\config\ directory.

2. Hardware Setup

The picture is attached below to show the hardware setup with DCA1000 + mmwave IC booster + xwr6843 EVM. Please pay close attention to the switch settings in yellow circles. Similar as the data capture using radar studio, users have to set PC to static IP, disable WIFI and firewall to be able to communicate with the DCA1000 board.



3. Create a directory C:\myCliSavedData

This directory is used to save the original captured raw data. The directory name is listed in the “.\config\datacard_config.json”, as well as hard coded inside MATLAB script .\matlab\CLI_OneTimeDataCapture_function.m

4. Run the MATLAB data capture script

MATLAB script uses some of the DCA1000 CLI executable and DLL library which is inside the radar studio directory. Therefore, the MATLAB capture script has to run under radar studio PostProc directory. Then, we will add the script directory by using “addpath”. After that, we can run capture_example. An example is given below (Please replace mmwave_studio_xx_xx_xx_xx with the latest radar studio version name):

```
>> cd C:\ti\mmwave_studio_xx_xx_xx_xx\mmWaveStudio\PostProc
```

```
>> addpath C:\DCA1000CLI_dataCapture_git\matlab
```

```
>> capture_example
```

Inside the “capture_example.m”, modified the “homeDir” as needed.

5. Main features in this example

The MATLAB script includes following main features:

- Reset radar board
- Configure the DCA1000 board through `.\config\datacard_config.json`
- Configure the radar sensor through serial port and configuration file `.\config\chirp1_60G_ES2.cfg`
- Parameter parsing: get/derived the system parameter from the sensor configuration file `.\config\chirp1_60G_ES2.cfg`, such as number of ADC samples, range resolution and etc.
- Stop sensor
- Start the DCA1000 data capture
- Restart sensor
- Stop DCA1000 data capture after pause for 3 second (users can change the capture file size by changing this pause time before sending stop command).
- Parse the first frame of captured data and apply range and Doppler FFT. Plot the Doppler FFT output to quickly validate the data. Note that this parsing function has some limitations; it matches with the following settings in configuration.
 - `reorderEnable` is set to 1 inside `.\config\datacard_config.json`
 - `lvdsStreamCfg -1 0 1 0` inside radar configuration file to disable header bytes and disable point cloud to transfer through LVDS.
 - `adcbufCfg -1 0 1 1 1` inside radar configuration file, only have one option in ADC buffer that Q sample before I sample.

IF any of these settings change, the parsing function has to be modified.

- Save the capture data into a unique file name based on test name and configuration file name.

6. Easy ways to adapt to the new use case

- Come up with your own chirp configuration file (instead of using `chirp1_60G_ES2.cfg` or `chirp1_60G_ES1.cfg`). Users are encouraged to change the profile/chirp/frame configuration, but keep the rest of the settings.

7. MATLAB licence issue

To run the data capture, we need to disable WIFI and other network, and only connect with the DCA1000 board. If users only have floating MATLAB LICENCE, then you may have trouble running capture script for a long time. In that case, users can generate a MATLAB executable as below:

```
>> mcc -m CLI_OneTimeDataCapture_function.m
```

Then save the execution function under `C:\ti\mmwave_studio_xx_xx_xx_xx\mmWaveStudio\PostProc.`