DCA1000EVM CLI Software User Guide

Revision 1.01
<10-Apr-2019>

Revision History

Version	Date	Author	Description
0.01	28-Feb-2019		Initial draft
0.02	29-Mar-2019		Sections 1.5, 2.3- 2.7, 4 updated
0.03	03-Apr-2019		Review comments incorporated
1.00	04-Apr-2019		Baselined
1.01	10-Apr-2019		Reordering config option added

Document License

Contributors to this document

Contents

1 Introduction

This document provides operational procedure and information of the DCA100EVM CLI application. This document explains the user interface level details of the DCA1000EVM CLI application.

1.1 Purpose

The purpose of this document is to present the user, a demo flow guide of the DCA1000EVM CLI software application and its commands and features.

1.2 Scope

The scope of this document is to briefly illustrate the user interface provided by the CLI tool for configuration and recording of the mmWave. The scope of this document is limited to using the pre-compiled tool for execution. The detailed information on the APIs and interfaces for extending/integrating this into another program is not included in this document. These subjects are discussed in a separate document.

1.3 Audience

Anyone interested in executing the DCA1000 EVM CLI.

1.4 Terms / Acronyms / Abbreviations

ACK	Acknowledgement
ADC	Analog-to-Digital Converter
API	Application Programming Interface
CLI	Command Line Interface
DDR	Double Data Rate
DLL	Dynamic Link Library
DOS	Disk Operating System
EEPROM	Electrically Erasable Programmable Read-Only Memory
EVM	Evaluation Module
FPGA	Field Programmable Gate Array
GB	Gigabit
GUI	Graphical User Interface
IP	Internet Protocol
IPC	Inter Process Communication
IPV4	Internet Protocol Version 4
JSON	JavaScript Object Notation

LIB	Library
LSB	Least Significant Bit
LTS	Long Term Support
LVDS	Low-Voltage Differential Signaling
MAC	Media Access Control
Mbps	Megabits per Second
MSB	Most Significant Bit
NACK	Negative Acknowledgement
PC	Personal Computer
OS	Operating System
OSAL	Operating System Abstraction Layer
UDP	User Datagram Protocol
UI	User Interface
us	Micro Second

1.5 CLI Overview

The DCA1000EVM CLI application is primarily a command line tool for configuration of FPGA and recording based on the user inputs. The DCA1000EVM CLI application connects to DCA1000EVM system through 1GB Ethernet for configuration and recording of data. RADAR EVM is connected to DCA1000EVM for data capture and connected to PC for configuration of data generation.

The DCA1000EVM CLI application has the following functionalities.

- CLI application parses the parameters in JSON formatted config file for the corresponding CLI Control commands.
- CLI application supports both Windows and Linux.
- Acknowledgement of commands and error codes are handled and returned by the CLI application as a signed 32-bit integer value to the calling application.
- The response status of each of the command is captured in a log file by the CLI application.
- CLI application supports running as a foreground as well as a background process.

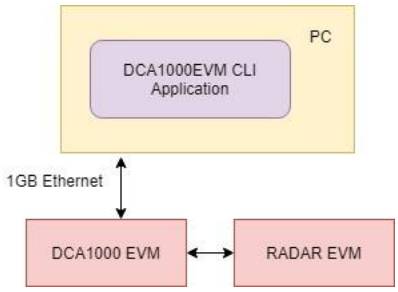


Figure 1 DCA1000EVM System overview

1.5.1 DCA1000EVM CLI Setup

- Copy the DCA1000EVM CLI binaries in the PC. Refer section 3.1.2 and 3.2.2 for the list of CLI binaries required for Windows and Linux platforms respectively.
- DCA1000EVM should be connected to Host PC via Ethernet cable to access the CLI and Data Transfer process.
- DCA1000EVM should be connected to PC via USB Cable (J1-Radar FTDI) for configuring the RADAR EVM by using on board FTDI chip.
- DCA1000EVM should be connected to TI Radar EVM via 60 pin HD Connector by using 60 pin Samtec ribbon cable.
- DCA1000EVM power input should be connected either from DC Jack or TI Radar EVM power output (from 60 pin HD connector) by selecting the switch SW3.
- RADAR EVM should be connected to ±5V power supply.
- Follow the mmWave Studio or mmWave SDK User Guide for additional RADAR EVM connectivity to PC and other pre-requisites.

1.5.2 DCA1000EVM Configuration Modes

DCA1000EVM recording can be tested either through Hardware switch configuration mode or Software CLI configuration mode.

1.5.2.1 Hardware switch configuration mode

DCA1000EVM switch settings for hardware configuration:

- Keep switch SW2.5 in HW_CONFIG position.

DCA1000EVM switch settings for FPGA configuration:

- Keep switch SW2.4 in RAW_MODE/MULTI_MODE
- Keep switch SW2.1 in LVDS_CAPTURE/DMM_PLAYBACK position.
- Keep switch SW2.2 in ETH_STREAM/SD_STORE position.
- Keep switch SW2.3 in AR1243_MODE for 4 Lane LVDS interfaces from xWR1243BOOST or in AR1642_MODE (Other side) for 2 Lane LVDS interfaces from xWR1642BOOST.
- Set SW1.1 to 12BIT_ON (or) SW1.2 to 14BIT_ON (or) SW1.3 to 16BIT_ON according to the data size selected in RADAR EVM.

DCA1000EVM switch settings for EEPROM configuration:

- Keep switch SW2 positioned at 11 (pin11) - DCA1000EVM will load default Ethernet configuration data from FPGA internal registers and the same will be used for Ethernet communication

1.5.2.2 Software CLI configuration mode

DCA1000EVM switch settings for software configuration:

- Keep switch SW2.5 in SW_CONFIG position.

DCA1000EVM JSON file settings for FPGA configuration:

- Logging Mode (dataLoggingMode) – RAW/MULTI
- LVDS Mode (lvdsMode) - 4 lane/2 lane
- Data Transfer Mode (dataTransferMode) – LVDS Capture/Playback
- Data Capture Mode (dataCaptureMode) – SD Card Storage/Ethernet Streaming
- Data Format Mode (dataFormatMode) – 12/14/16 bit

DCA1000EVM switch settings for EEPROM configuration:

- Keep switch SW2 positioned at 6 (pin6) - DCA1000EVM will load Ethernet configuration data from EEPROM.

1.5.2.3 Record process LED indications

- When data transfer starts **DATA_TRANS_PRG** LED (LD1) will start toggling and '**Record in progress**' message will be shown in CLI.
- When record is completed **DATA_TRANS_PRG** LED (LD1) will glow and '**Record is completed**' message will be shown in CLI.
- If RADAR EVM is not sending LVDS data, then **LVDS_PATH_ERR** LED (LD7) will glow and '**No LVDS data**' message will be shown in CLI.
- When DDR gets full **DDR_FULL** LED (LD6) will glow and '**DDR is full**' message will be shown in CLI.
- Whenever FPGA internal buffer gets full **FPGA_ERR** LED (LD5) will glow and '**LVDS buffer full**' message will be shown in CLI.

1.5.3 RADAR EVM configuration modes

Refer mmWave Studio or mmWave SDK for RADAR EVM setup details.

2 DCA1000EVM CLI Files and Status codes Information

2.1 JSON Config File

CLI tool accepts configuration parameter through JSON format config file. The config file is given as a command line argument to the CLI tool.

File extension : .json
Type : JSON formatted data in text format

Sample JSON File Data:

```
{
    "DCA1000Config": {
    "dataLoggingMode": "raw",
    "dataTransferMode": "LVDSCapture",
    "dataCaptureMode": "ethernetStream",
    "lvdsMode": 1,
    "dataFormatMode": 1,
    "packetDelay_us": 25,
    "ethernetConfig": {
            "DCA1000IPAddress": "192.168.33.180",
            "DCA1000ConfigPort": 4096,
            "DCA1000DataPort": 4098
    },
    "ethernetConfigUpdate": {
            "systemIPAddress": "192.168.33.30",
            "DCA1000IPAddress": "192.168.33.180",
            "DCA1000MACAddress": "12.34.56.78.90.12",
            "DCA1000ConfigPort": 4096,
            "DCA1000DataPort": 4098
    },
    "captureConfig": {
            "fileBasePath": "D:\\capture",
            "filePrefix": "outdoor_capture",
            "maxRecFileSize_MB": 1024,
            "sequenceNumberEnable": 1,
            "captureStopMode": "duration",
            "bytesToCapture": 50000,
            "durationToCapture_ms": 5000,
            "framesToCapture":10
    },
    "dataFormatConfig": {
            "MSBToggle": 0,
            "reorderEnable": 1,
```

```
                "laneFmtMap": 0,
                "dataPortConfig": [
                {
                        "portIdx": 0,
                        "dataType": "real"
                },
                {
                        "portIdx": 1,
                        "dataType": "complex"
                },
                {
                        "portIdx": 2,
                        "dataType": "real"
                },
                {
                        "portIdx": 3,
                        "dataType": "real"
                },
                {
                        "portIdx": 4,
                        "dataType": "complex"
                }
                ]
        }
        }
}
```

The min, max and default values for the JSON config file parameters are as follows:

Parameter	Min	Max	Default	Description
DCA1000Config	-	-	-	Object contains DCA1000EVM related configuration
dataLoggingMode	-	-	Raw	Data logging mode specifies the type of data being transferred in record mode through DCA1000EVM. This field is valid only when dataTransferMode is “LVDSCapture”. The valid options are • raw • multi
dataTransferMode	-	-	LVDSCapture	Data transfer mode specifies if DCA100EVM is in record mode or playback mode.

				The valid options are • LVDSCapture • LVDSPlayback
dataCaptureMode	-	-	ethernetStream	Data capture mode specifies the transport mechanism for getting data out of DCA1000EVM. This field is valid only when dataTransferMode is "LVDSCapture". The valid options are • ethernetStream • SDCardStorage
lvdsMode	1	2	1	LVDS mode specifies the lane config for LVDS. This field is valid only when dataTransferMode is "LVDSCapture". The valid options are • 1 (4lane) • 2 (2lane)
dataFormatMode	1	3	3	Data format mode specifies the bit-mode for the captured data. This field is valid only when dataTransferMode is "LVDSCapture". The valid options are • 1 (12 bit) • 2 (14 bit) • 3 (16 bit)
packetDelay_us	5	500	25	Value in usec to throttle the throughput of the Ethernet stream out of DCA1000EVM. Min and max values are dictated by the limits supported by DCA1000 H/W. This field is valid only when dataCaptureMode is "ethernetStream"
ethernetConfig	-	-	-	Config block for Ethernet stream. This block is valid only when dataCaptureMode is "ethernetStream"

DCA1000IPAddress	-	-	192.168.33.180	IP address of the DCA1000EVM
DCA1000ConfigPort	1	65535	4096	Config port number for config command communication between DCA1000EVM and PC
DCA1000DataPort	1	65535	4098	Data port number for data communication between DCA1000EVM and PC
ethernetConfigUpdate	-	-	-	Config block to reconfigure the Ethernet details in EEPROM of DCA1000EVM
systemIPAddress	-	-	192.168.33.30	To reconfigure the IP address of the PC in EEPPROM of DCA1000EVM
DCA1000IPAddress	-	-	192.168.33.180	To reconfigure the IP address of the DCA1000EVM in EEPPROM of DCA1000EVM
DCA1000MACAddress	-	-	12.34.56.78.90.12	To reconfigure the MAC address of the DCA1000EVM in EEPPROM of DCA1000EVM
DCA1000ConfigPort	1	65535	4096	To reconfigure the config port number in EEPPROM of DCA1000EVM for config command communication between DCA1000EVM and PC
DCA1000DataPort	1	65535	4098	To reconfigure the data port number in EEPPROM of DCA1000EVM for data communication between DCA1000EVM and PC
captureConfig	-	-	-	Config block for data capture
fileBasePath	-	-	D:\\capture	Valid file path on the PC where this CLI runs. This block is valid only when dataCaptureMode is “ethernetStream”
filePrefix	-	-	outdoor_capture	Filename conforming to host PC rules; CLI would append index numbers, etc to this filePrefix to derive the actual filename that contains recorded data
maxRecFileSize_MB	1	1024	1024	Record data file maximum size in MB
sequenceNumberEnable	0	1	0	This field controls whether the packet sequence number need to be stored in the data file or not.

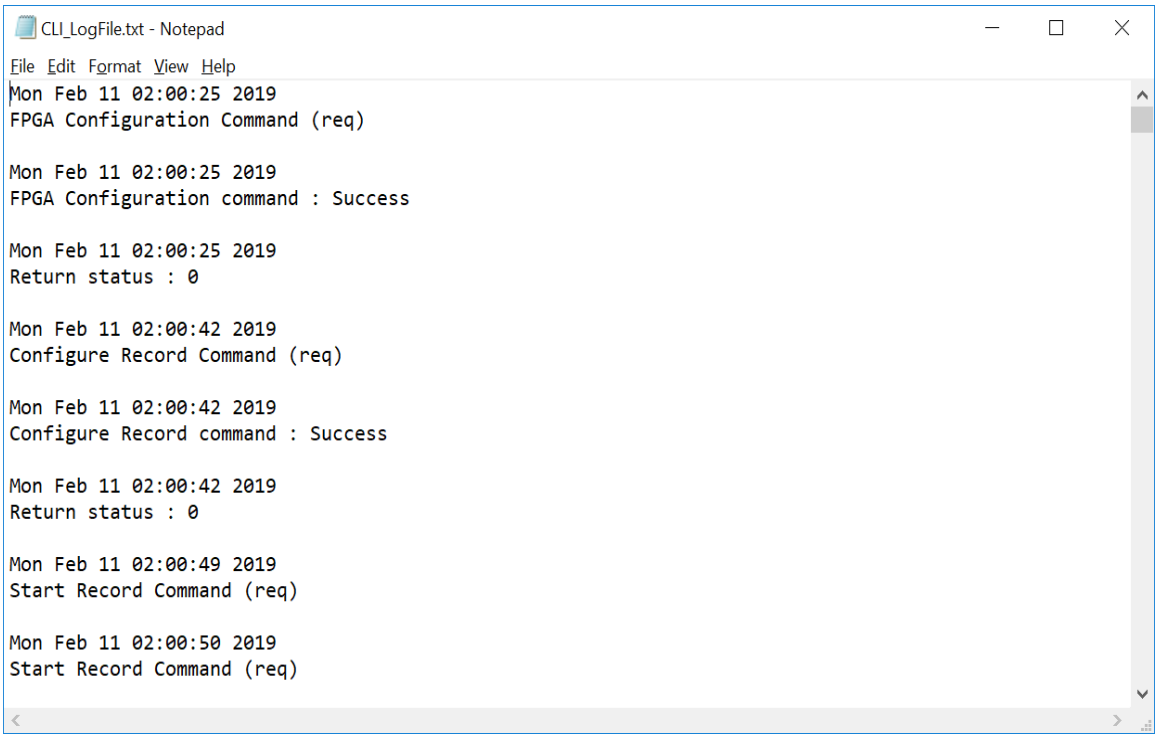
				This field is valid only when data captured using post processing method. The valid options are • 0 (Disable) • 1 (Enable)
captureStopMode	-	-	infinite	Stop mode for the capture. Based on this config, other fields should be specified to provide more config for that capture stop mode. The valid options are • bytes • frames • duration • infinite
bytesToCapture	128	0xFFFF FFFF	50000	Specifies the number of bytes to capture when captureStopMode is “bytes”
durationToCapture_ms	40	0xFFFF FFFF	5000	Specifies the capture duration in msec when captureStopMode is “duration”
framesToCapture	1	0xFFFF	10	Specifies the number of radar frames to capture when captureStopMode is “frames”
dataFormatConfig	-	-	-	Config block specifies the data format to assist in data formatting services of the CLI
MSBToggle	0	1	0	Specifies the MSB toggle in the captured data (16-bit value) to be enabled or not The valid options are • 0 (Disable) • 1 (Enable)
reorderEnable	0	1	1	Specifies the reordering of the captured data (bytes) to be enabled or not The valid options are • 0 (Disable) 1 (Enable)

Table 1 JSON File data description

2.2 Output Files

2.2.1 CLI logfile

File name : *CLI_LogFile.txt*
File extension : .txt
Type : Text
Description :
Contains all the commands execution information along with the timestamp. It can be viewed manually whenever required. The fill will be appended for new sessions and new command execution information.



2.2.2 Record process datafile

File name : <File_Prefix>_<Raw/Header Mode>_<iteration>.bin
File extension : .bin
Type : Binary
Description :

- In Raw Mode, the data streamed over ethernet would be captured in the filename provided by the user.

  For eg: If user provides the prefix filename as “RawModeCap”, then

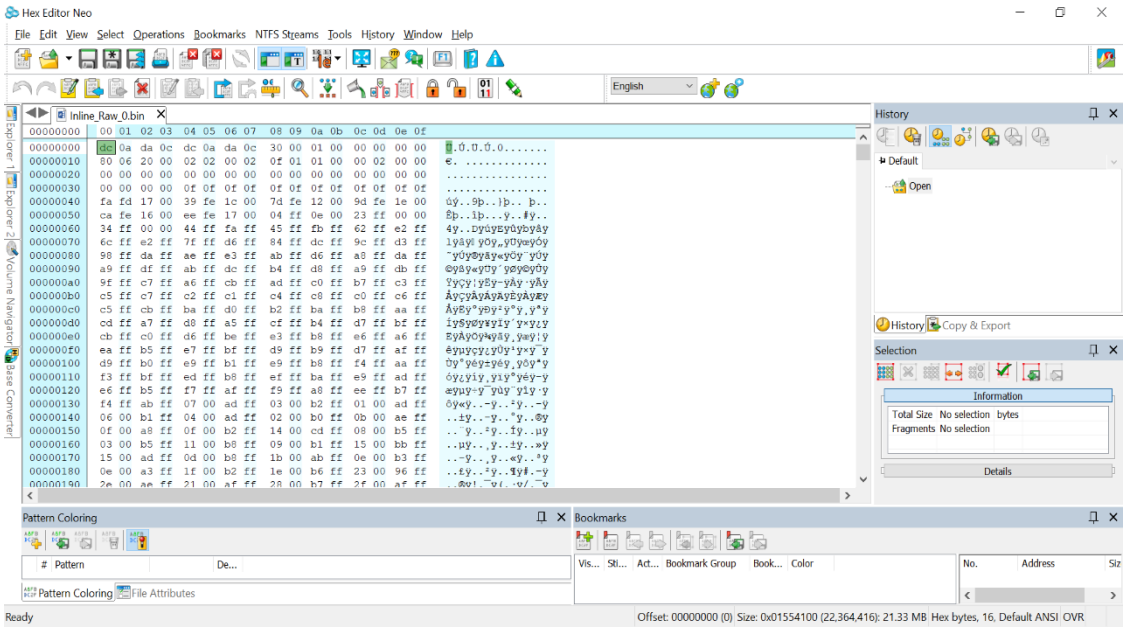
  “RawModeCap_Raw_0.bin” → Raw data would be captured

- In Multi-Mode, the data streamed over ethernet would be captured in the 5 different files based on the filename provided by the user

  For eg: If user provides the filename as “MultiModeCap”, then

"MultiModeCap_hdr_xxxx_0.bin" -> xxxx refers to 4 characters of the header received in the first packet of the respective 5 ports.

For ADC data, filename will be generated as "MultiModeCap_hdr_0ADC_0.bin" if the received header is '0x0CDA0ADC0CDA**0ADC**'

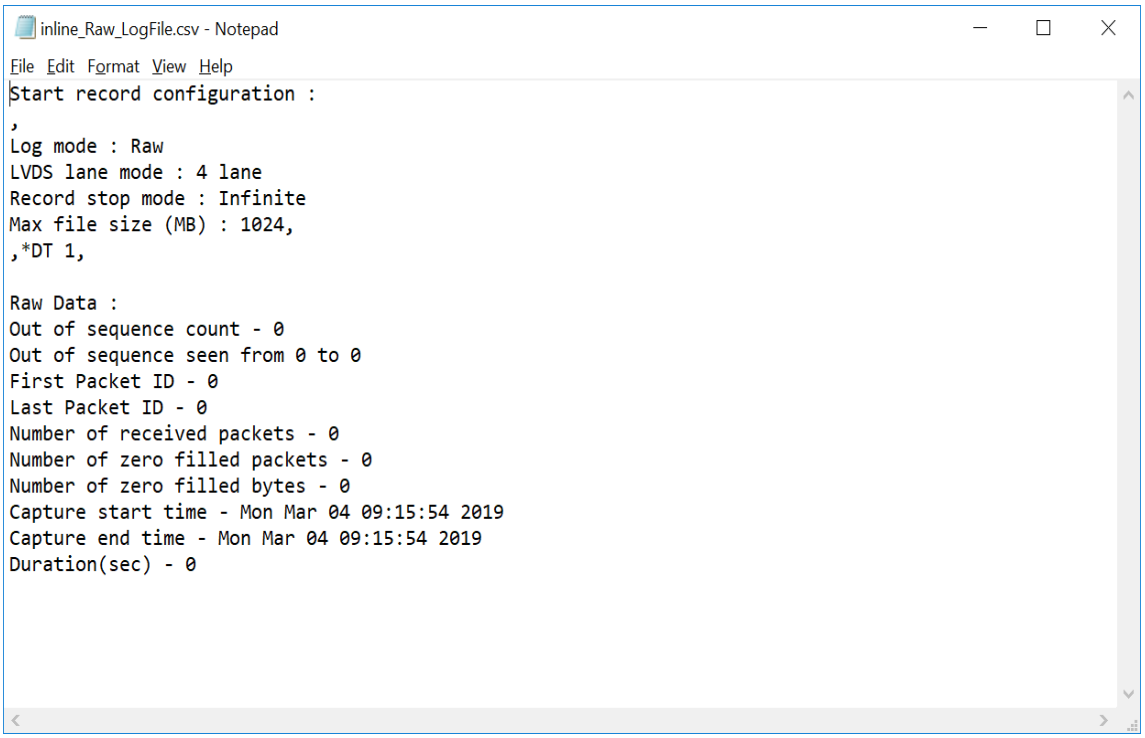


2.2.3 Record process logfile

File name : <File_Prefix>_<Raw/Header Mode>_LogFile.csv
File extension : .csv
Type : Comma separated text
Description : Contains the following information

- Record process configuration
  - Log mode
  - LVDS lane mode
  - Record stop mode
  - Maximum captured data file size
- If Post processing, record process summary
  - Out of sequence count
  - First packet ID
  - Last packet ID
  - Number of received packets
  - Capture start time
  - Capture end time
  - Capture duration
- If Inline processing, for each data port
  - Dropped packet offset (if any)
  - Number of dropped bytes at the offset (if any)

- If Inline processing, record process summary
  - Out of sequence count
  - Latest out of sequence between <seq num> and <seq num>
  - First packet ID
  - Last packet ID
  - Number of received packets
  - Number of zero filled packets
  - Number of zero filled bytes
  - Capture start time
  - Capture end time
  - Capture duration



2.3 System status codes

Following system status codes will be updated in the shared memory by CLI Record tool which can be read by '*query_status*' command by CLI Control tool.

DCA1000EVM System Status	Bit position	Status Type	Description
STS_NO_LVDS_DATA	0	Warning	If there is no LVDS data from RADAR EVM, after configured timeout seconds CLI would display "No LVDS data".

			User must ensure the proper DCA1000EVM FPGA configuration and can restart the capture.
STS_NO_HEADER	1	Warning	If data logging mode is configured to multi-mode and there is no LVDS data from RADAR EVM, after configured timeout seconds CLI would display as “No Header”. User must ensure the proper config file is executed and can restart the capture.
STS_EEPROM_FAILURE	2	Warning	If EEPROM failure happened, CLI would display “EEPROM Failure”. User can power cycle DCA1000EVM and can check EEPROM connectivity and address lines on hardware
STS_SD_CARD_DETECTED	3	NA	NA
STS_SD_CARD_REMOVED	4	NA	NA
STS_SD_CARD_FULL	5	NA	NA
STS_MODE_CONFIG_FAILURE	6	NA	NA
STS_DDR_FULL	7	Warning	If DDR is full, CLI would display “DDR full”. Once DDR is full, DCA1000EVM will truncate the overflown data from RADAR EVM. User can restart the capture with lesser LVDS rate config than the Ethernet data rate and/or lesser delay between record packets.
STS_REC_COMPLETED	8	Warning	If record is completed, CLI would display “Record is completed”. User can ensure the record is stopped after receiving record completion status.
STS_LVDS_BUFFER_FULL	9	Warning	If LVDS buffer is full, CLI would display “LVDS buffer full”. Once LVDS buffer is full, DCA1000EVM will truncate the overflown data from RADAR EVM. User can restart the capture with lesser LVDS rate config than the Ethernet data rate and/or lesser delay between record packets.
STS_PLAYBACK_COMPLETED	10	NA	NA
STS_PLAYBACK_OUT_OF_SEQ	11	NA	NA
Not used	12-15		Future Use
DLL Async Status	**Bit position**	**Status Type**	**Description**
STS_REC_PKT_OUT_OF_SEQ	16	Warning	If record packet is in out of sequence, CLI would display “Packet is in out of sequence”.

			User can restart the capture knowing that data loss might happen.
STS_REC_PROC_TIMEOUT	17	Fatal	If record process is timeout, CLI would display “Record process Timeout error”. User must ensure the system connectivity.
STS_REC_FILE_CREATION_ERR	18	Fatal	If record file creation is failed, CLI would display “Record file creation failed”. User must ensure the free disk space for data capture.
STS_REC_REORDERING_ERR	19	Fatal	If record reordering is failed in inline processing, CLI would display “Record process reordering failed”. User must restart the capture since the captured data is not proper for data alignment algorithm. For 4-lane, algorithm needs 16 bytes of data to do alignment. For 2-lane, algorithm needs 8 bytes of data to do alignment. If the number of bytes in the received packet is not the multiple of 16/8, then the data alignment will fail.
STS_INVALID_RESP_PKT_ERR	20	Warning	If the header and footer is not matching in the received packet over the config port, CLI would display as “Invalid packet received”. User can power cycle the DCA1000EVM and start sending commands.
STS_REC_INLINE_BUF_ALLOCATION_ERR	21	Fatal	If record inline buffer memory allocation is failed, CLI would display “Inline buffer allocation failed”. Received data will be discarded and not be processed. User must check the RAM size of the PC for data capture and can restart the capture.

Table 2 Asynchronous status codes

2.4 Record process states codes

Following record process states codes will be updated in the shared memory by CLI Record tool which can be read by ‘*query_status*’ command by CLI Control tool.

Record Process Status (*s32CommandStatus*)	Status	Description

STS_CLI_REC_PROC_IS_IN_PROG	-4029	When the record start command response is received by CLI Record tool, CLI Record tool will update the shared memory state as 'Record is in progress'
STS_CLI_REC_PROC_STOPPED	-4030	When the record stop command response is received by CLI Record tool, CLI Record tool will update the shared memory state as 'Record process is stopped'
STS_CLI_REC_PROC_START_INIT	-4031	When the record start command is executed by CLI Record tool, CLI Record tool will update the shared memory state as 'Record process is initiated'
STS_CLI_REC_PROC_START_FAILED	-4032	When the record start command response is not received by CLI Record tool, CLI Record tool will update the shared memory state as 'Start record process is failed'
STS_CLI_REC_PROC_STOP_INIT	-4033	When the record stop command is executed by CLI Record tool, CLI Record tool will update the shared memory state as 'Stop record process is initiated'
STS_CLI_REC_PROC_STOP_FAILED	-4069	When the record stop command response is not received by CLI Record tool, CLI Record tool will update the shared memory state as 'Stop record process is failed'

Table 3 Record process states

2.5 Capture stop modes

CLI application supports following four modes for stop record process.

- Bytes
  - CLI application will be configured with the *bytes* stop mode and number of bytes to capture, when start record command is sent to DCA1000EVM.
  - On receiving the data packets, the number of bytes will be counted.
  - If any one of the data ports receives the total bytes to be captured, DLL will send the record completed status to the CLI Record tool.
  - CLI Record tool will use the DLL to send the stop record command over the config port through Ethernet.

- Frames
  - DLL will be configured with the *frames* stop mode and number of frames to capture, when start record command is sent to DCA1000EVM.
  - This mode is valid only in multi-mode data capture.
  - On receiving the first packet, the Frame ID will be stored by DLL. The number of frames will be counted if the frame ID is present in the received packet (Every new frame will be received as a new packet and hence frame ID will always be the first 8 bytes of the data payload).
  - If any one of the data ports receives the total frames to be captured, DLL will send the record completed status to the CLI Record tool.
  - CLI Record tool will use the DLL to send the stop record command over the config port through Ethernet.

- Duration
  - CLI application will be configured with the *duration* stop mode and duration in millisecond to capture, when start record command is sent to DCA1000EVM.
  - On receiving the first packet, the timer will be started by DLL.
  - After the defined duration, DLL will send the record completed status to the CLI Record tool.
  - CLI Record tool will use the DLL to send the stop record command over the config port through Ethernet.

- Infinite
  - CLI application will be configured with the *infinite* stop mode when start record command is sent to DCA1000EVM.
  - Once the record completed status is received from the DCA1000EVM, DLL will send the record completed status to the CLI Record tool.
  - CLI Record tool will use the DLL to send the stop record command over the config port through Ethernet.

3 DCA1000EVM CLI Execution Instructions

3.1 Windows

3.1.1 System Requirements

The CLI application is tested on Windows 10 systems with below configurations.

- Operating System: Windows 10 64-bit OS
- RAM: 8 GB or above
- 1 GB-Ethernet port
- High speed storage for capture / store of the data files

DCA1000EVM CLI application to run in Windows platforms, needs the following files to be available at the same path.

3.1.2 Files Requirements To Execute CLI

Files required for CLI execution:

Files	Description
CLI Application Binary Files	
DCA1000EVM_CLI_Control.exe	Executable file that does validation of user inputs and execution of configuration commands
DCA1000EVM_CLI_Record.exe	Executable file that does validation of user inputs and execution of record commands
RF_API.dll	Dynamic library file that handles execution of configuration commands and recording of the captured data in files
configFile.json	JSON file for configuration from user. Refer section 2.1 for details on JSON file
Standard System Library Files	
libgcc_s_dw2-1.dll	System DLL required only if the CLI executables are built using dynamic linkage
libstdc++-6.dll	System DLL required only if the CLI executables are built using dynamic linkage
libgcc_s_sjlj-1.dll	System DLL required only if the CLI executables are built using dynamic linkage
libwinpthread-1.dll	System DLL required only if the CLI executables are built using dynamic linkage

- Open the command prompt and move to the directory where the above-mentioned files are downloaded.
- Start recording using command sequence mentioned in the following section 3.1.3

3.1.3 CLI Command Sequence

For successful recording of data from RADAR EVM sequence is given as follows

1. Configure FPGA
    - Ensure JSON config file (CLI) and Script config file (RADAR EVM) data format mode are in sync
    - Command - ***DCA1000EVM_CLI_Control.exe fpga configFile.json***
2. Configure record delay
    - Command - ***DCA1000EVM_CLI_Control.exe record configFile.json***
3. Start the record
    - Ensure JSON config file (CLI) and Script config file (RADAR EVM) data logging mode are in sync
    - Command - ***DCA1000EVM_CLI_Control.exe start_record configFile.json***
4. Stop the record after recording data
    - Command - ***DCA1000EVM_CLI_Control.exe stop_record configFile.json***

For successful record process, FPGA should be reconfigured in the following scenarios

- When the system is booted or rebooted
- When the FPGA or DCA1000EVM is reset
- On switching between multi-mode and raw mode

3.1.4 CLI Application Commands

CLI application suite includes two executables. They are: - DCA1000EVM_CLI_Control.exe and DCA1000EVM_CLI_Record.exe.

Of the two, the DCA1000EVM_CLI_Control.exe is the application providing options for the configuration commands and initiating the start and stop of the record process (DCA1000EVM_CLI_Record.exe). Following are some of the key aspects of the DCA100EVM_CLI_Control tool.

- CLI_Control tool supports blocking calls for configuration and Stop Record commands.
- CLI_Control tool supports non-blocking for the Start Record and record *query_status* commands.
- Multiple instances of the CLI_Control tool can be instantiated on the PC to control multiple EVMs while using unique UDP Ports for each EVM.
- It provides option to execute in “quiet” mode wherein there are no messages being displayed on the console.
- For all configuration commands, the configJsonFile should have valid DCA1000EVM system IP and config/data port numbers and the parameters corresponding to the executing commands.
- When CLI_Control is invoked to send continuous commands, CLI_Control will execute the command only when no other commands are in execution. If any other command is in progress, the CLI_Control will prompt the user to stop the already running process.

The generic execution flow of CLI_Control commands is as follows,

Calling Convention -

DCA1000EVM_CLI_Control.exe <command> [jsonCfgFile] [-q]

<command>: Supports following commands
[jsonCfgFile]: Json format input parameters text file path
[-q]: Quiet mode – No status display in the console

Commands supported by CLI_Control tool -

- fpga Configure FPGA
- eeprom Update EEPROM
- reset_fpga Reset FPGA
- reset_ar_device Reset AR Device
- start_record Start Record
- stop_record Stop Record
- record Configure Record delay
- dll_version Read DLL version
- fpga_version Read FPGA version
- cli_version Read CLI version
- query_status Read status of record process
- query_sys_status DCA1000EVM System aliveness
- -h List of commands supported

The generic execution flow of CLI_Control commands is as follows,

- User will initiate the control command through script or command line as *DCA1000EVM_CLI_Control.exe* <command> *configFile.json*
- CLI_Control tool reads the shared memory for checking whether any record process is running.
- If record process is running, CLI_Control tool will prompt the user to stop the already running record process. The CLI_Control tool logs the information in a log file with timestamp and will be terminated.
- If no process is running, CLI_Control tool validates the input parameters from the configJsonfile.
- CLI_Control initializes the config port ethernet connection and send <command> and wait for response packet till timeout.
- CLI_Control displays the command status to the user in the console. The CLI_Control tool also stores the configuration command status in a log file with timestamp. The exit code of the last executed process can also be read using OS system calls based on the calling application.

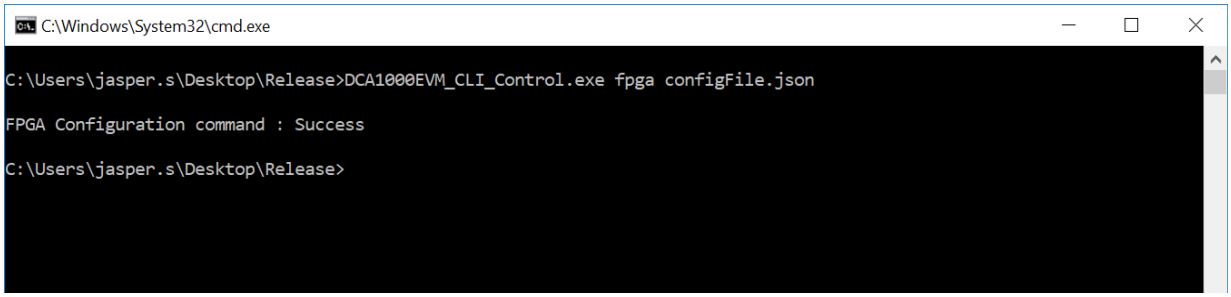
3.1.4.1 Configure FPGA

CLI_Control tool will execute this command to configure the DCA1000EVM with the following mode configuration

1. Logging Mode (*dataLoggingMode*) – RAW/MULTI
2. LVDS Mode (*lvdsMode*) - 4 lane/2 lane
3. Data Transfer Mode (*dataTransferMode*) – LVDS Capture/Playback
4. Data Capture Mode (*dataCaptureMode*) – SD Card Storage/Ethernet Streaming
5. Data Format Mode (*dataFormatMode*) – 12/14/16 bit
6. Timer (Not configurable using JSON file)

Command:

DCA1000EVM_CLI_Control.exe fpga configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention

Configure FPGA	"dataLoggingMode": "raw",
	"dataTransferMode": "LVDSCapture",
	"dataCaptureMode": "ethernetStream",
	"lvdsMode": 1,
	"dataFormatMode": 3,
	"ethernetConfig": {
	"DCA1000IPAddress": "192.168.33.180",
	"DCA1000ConfigPort": 4096,
	"DCA1000DataPort": 4098
	}
		DCA1000EVM_CLI_Control fpga configFile.json

Table 4 Configure FPGA calling convention

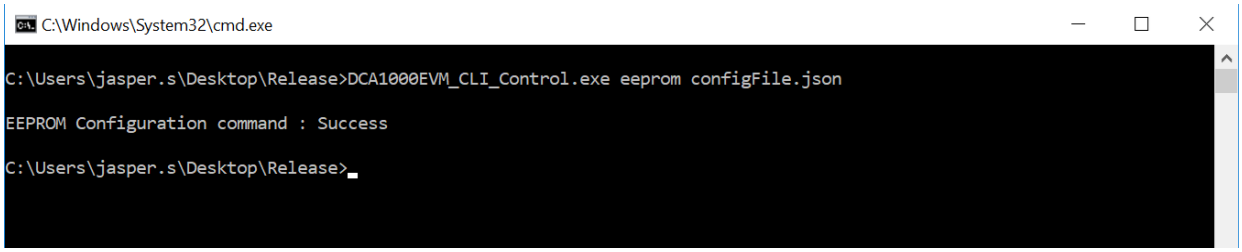
3.1.4.2 Configure EEPROM

CLI_Control tool will execute this command to reconfigure the IP address of the DCA1000EVM with the following configuration

1. MAC ID (*ethernetConfigUpdate -> DCA1000MACAddress*)
2. PC IP Address (*ethernetConfigUpdate -> systemIPAddress*)
3. Board IP Address (*ethernetConfigUpdate -> DCA1000IPAddress*)
4. Record port number (*ethernetConfigUpdate -> DCA1000DataPort*)
5. Configuration port number (*ethernetConfigUpdate -> DCA1000ConfigPort*)

Command:

DCA1000EVM_CLI_Control.exe eeprom configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention

Configure EEPROM	"ethernetConfig": {
	"DCA1000IPAddress": "192.168.33.180",
	"DCA1000ConfigPort": 4096,
	"DCA1000DataPort": 4098
	},
	"ethernetConfigUpdate": {
	"systemIPAddress": "192.168.33.30",
	"DCA1000IPAddress": "192.168.33.180",
	"DCA1000MACAddress": "12.34.56.78.90.12",
	"DCA1000ConfigPort": 4096,
	"DCA1000DataPort": 4098
	}
		DCA1000EVM_CLI_ Control eeprom configFile.json

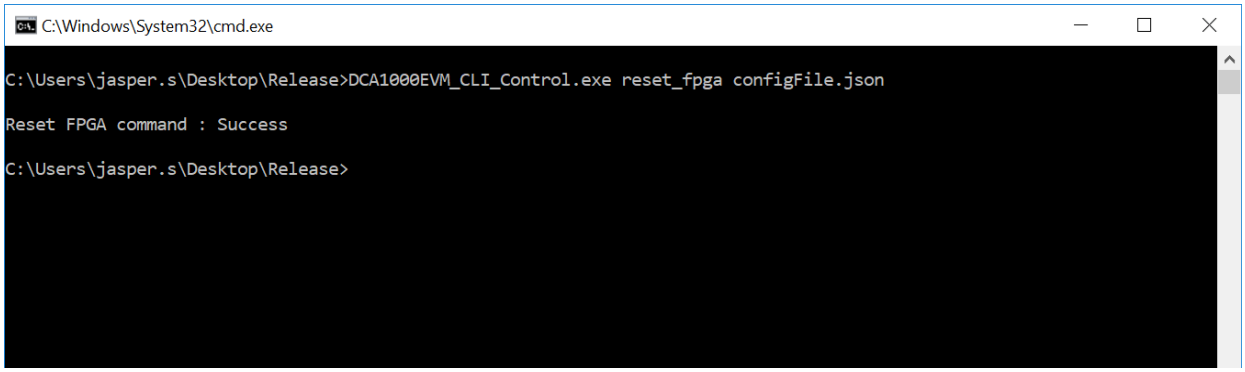
Table 5 Configure EEPROM calling convention

3.1.4.3 Reset FPGA

CLI_Control tool will execute this command to reset the DCA1000EVM FPGA.

Command:

DCA1000EVM_CLI_Control.exe resert_fpga configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention
Reset FPGA	"ethernetConfig": { "DCA1000IPAddress": "192.168.33.180", "DCA1000ConfigPort": 4096, "DCA1000DataPort": 4098 }	DCA1000EVM_CLI_Control reset_fpga configFile.json

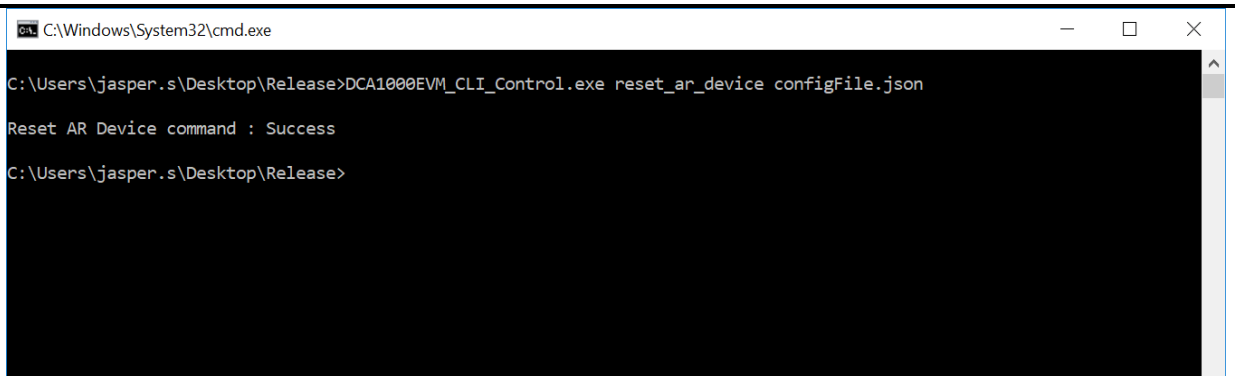
Table 6 Reset FPGA calling convention

3.1.4.4 Reset RADAR EVM

CLI_Control tool will execute this command to reset RADAR EVM.

Command:

DCA1000EVM_CLI_Control.exe reset_ar_device configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention
Reset RADAR EVM	"ethernetConfig": { "DCA1000IPAddress": "192.168.33.180", "DCA1000ConfigPort": 4096, "DCA1000DataPort": 4098 }	DCA1000EVM_CLI_Control reset_ar_device configFile.json

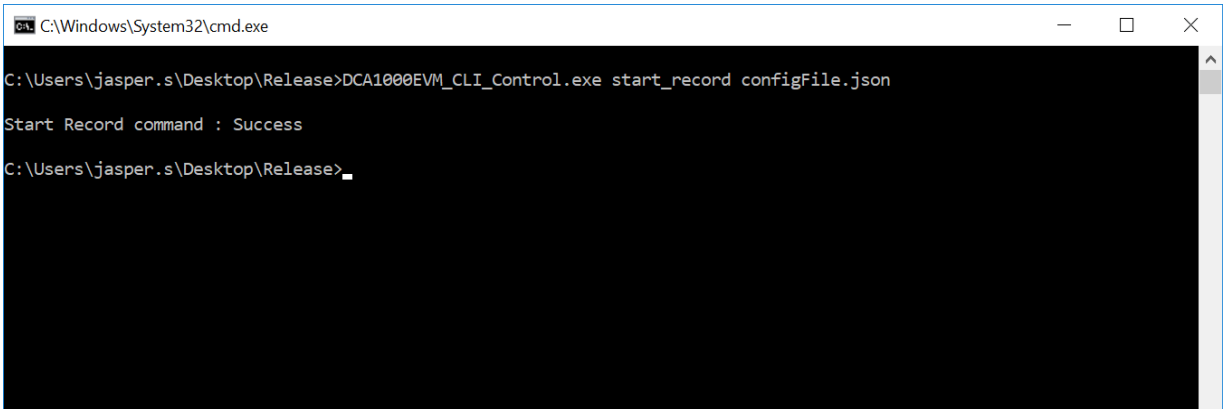
Table 7 Reset RADAR EVM calling convention

3.1.4.5 Start record

CLI_Control tool will execute the command to verify DCA1000EVM connectivity and invoke the CLI Record tool to start the recording.

Command:

DCA1000EVM_CLI_Control.exe start_record configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention

Start Record	"dataLoggingMode": "raw",
	"ethernetConfig": {
	"DCA1000IPAddress": "192.168.33.180",
	"DCA1000ConfigPort": 4096,
	"DCA1000DataPort": 4098
	},
	"captureConfig": {
	"fileBasePath": "C:\\Users\\CLI_Inline",
	"filePrefix": "inline",
	"maxRecFileSize_MB": 1024,
	"sequenceNumberEnable": 1,
	"captureStopMode": "infinite",
	"bytesToCapture": 4000,
	"durationToCapture_ms": 4000,
	"framesToCapture": 40
	},
	"dataFormatConfig": {
	"MSBToggle": 0,
	"reorderEnable": 1,
		DCA1000EVM_CLI_Control start_record configFile.json

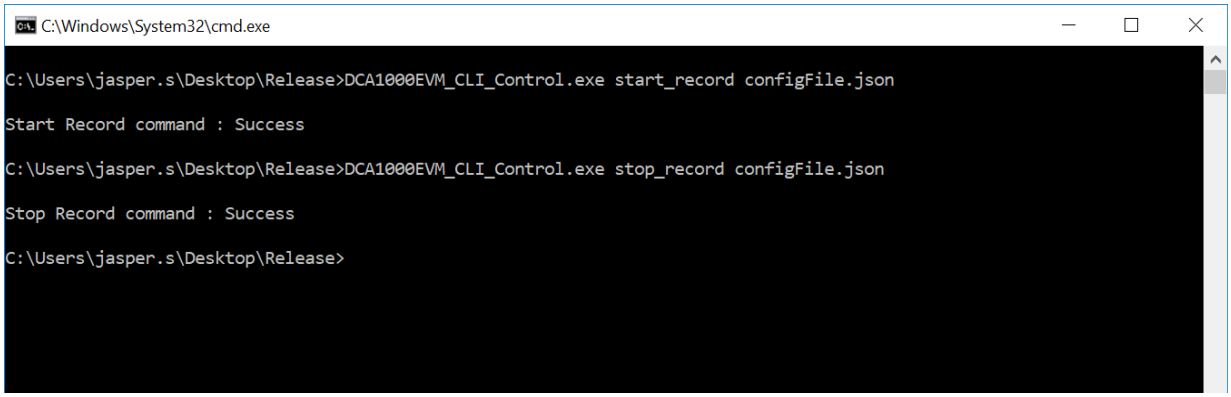
Table 8 Start Record calling convention

3.1.4.6 Stop record

CLI_Control tool will execute this command to stop the recording.

Command:

DCA1000EVM_CLI_Control.exe stop_record configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention

Stop Record (Based on user stop)	"ethernetConfig": {
	"DCA1000IPAddress": "192.168.33.180",
	"DCA1000ConfigPort": 4096,
	"DCA1000DataPort": 4098
	}
		DCA1000EVM_CLI_Control stop_record configFile.json

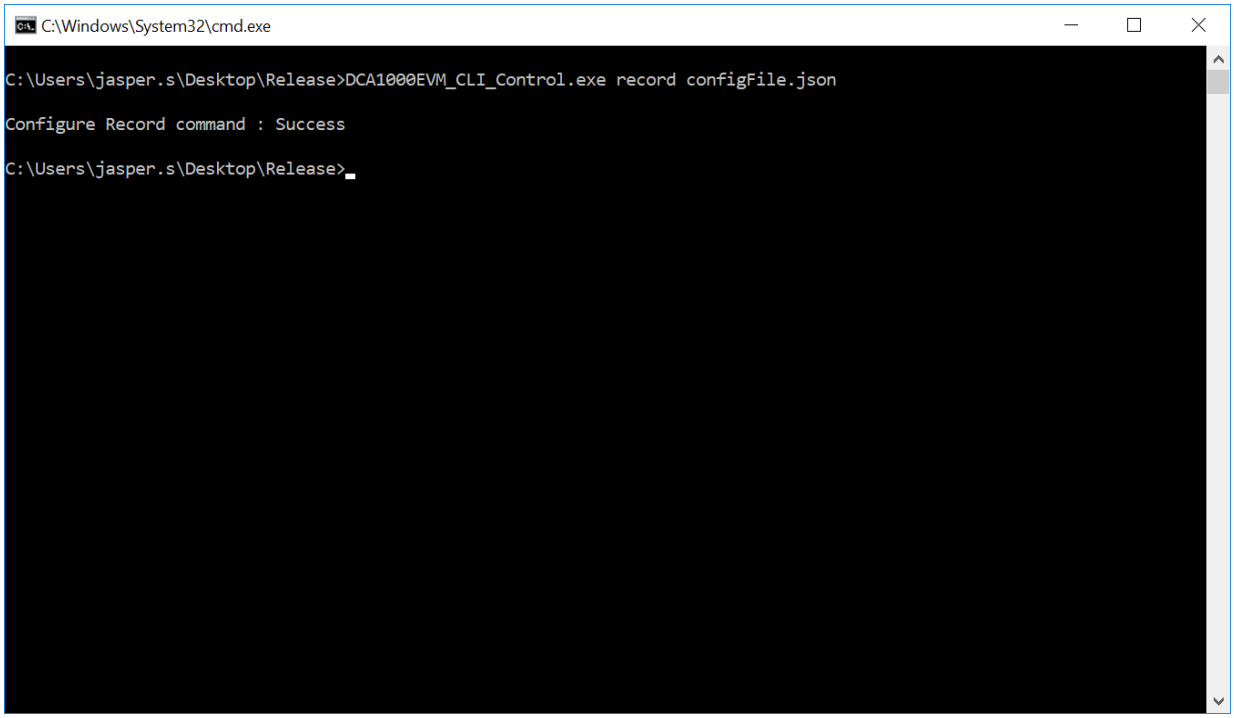
Table 9 Stop Record calling convention

3.1.4.7 Configure record delay

CLI_Control tool will execute this command to configure delay between record packets.

Command:

DCA1000EVM_CLI_Control.exe record configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention
Configure Record delay	"packetDelay_us": 5, "ethernetConfig": { "DCA1000IPAddress": "192.168.33.180",	DCA1000EVM_CLI_Control record configFile.json

	"DCA1000ConfigPort": 4096, "DCA1000DataPort": 4098 }	

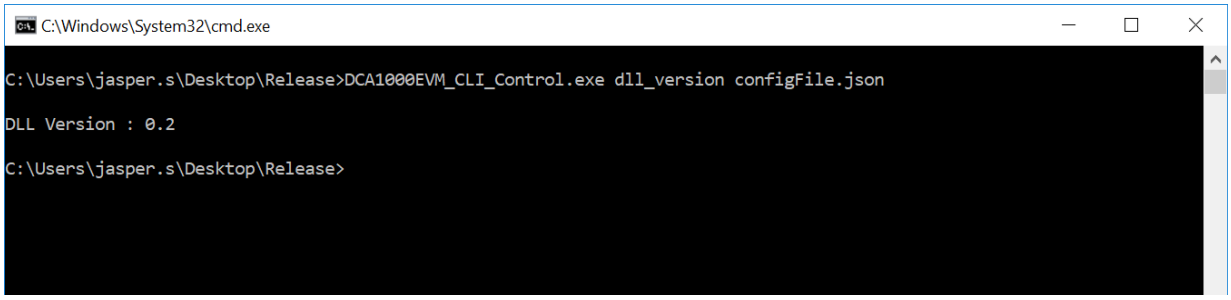
Table 10 Configure record delay calling convention

3.1.4.8 Read DLL version

CLI_Control tool will execute this command to read DLL version.

Command:

DCA1000EVM_CLI_Control.exe dll_version configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention
Read DLL version	NA	DCA1000EVM_CLI_Control dll_version

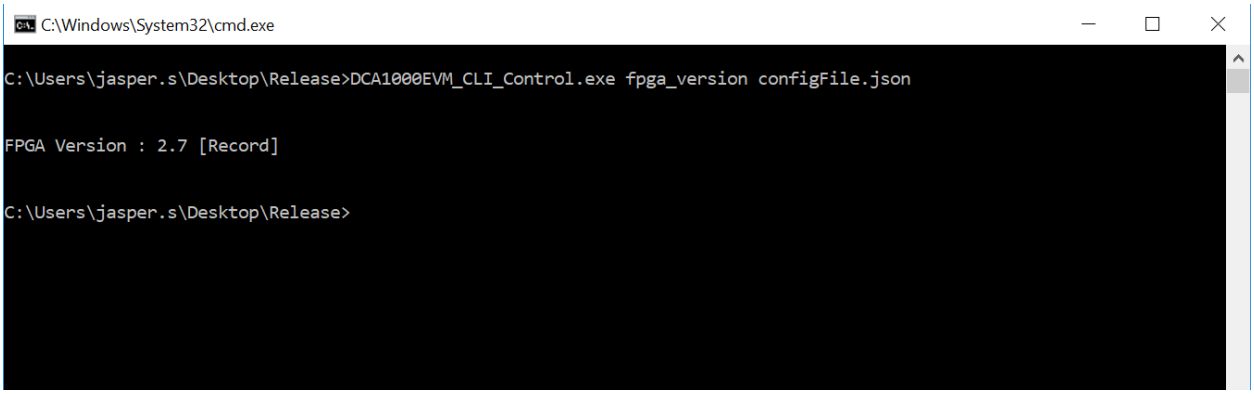
Table 11 Read DLL version calling convention

3.1.4.9 Read FPGA version

CLI_Control tool will execute this command to read FPGA version of DCA1000EVM.

Command:

DCA1000EVM_CLI_Control.exe fpga_version configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention
Read FPGA version	"ethernetConfig": { "DCA1000IPAddress": "192.168.33.180", "DCA1000ConfigPort": 4096, "DCA1000DataPort": 4098 }	DCA1000EVM_CLI_Control fpga_version configFile.json

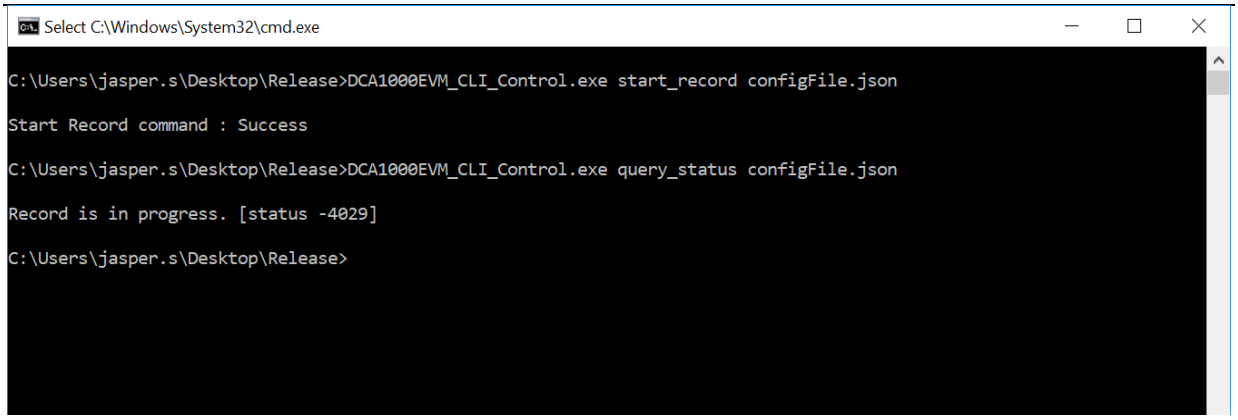
Table 12 Read FPGA version calling convention

3.1.4.10 Query Record Process Status

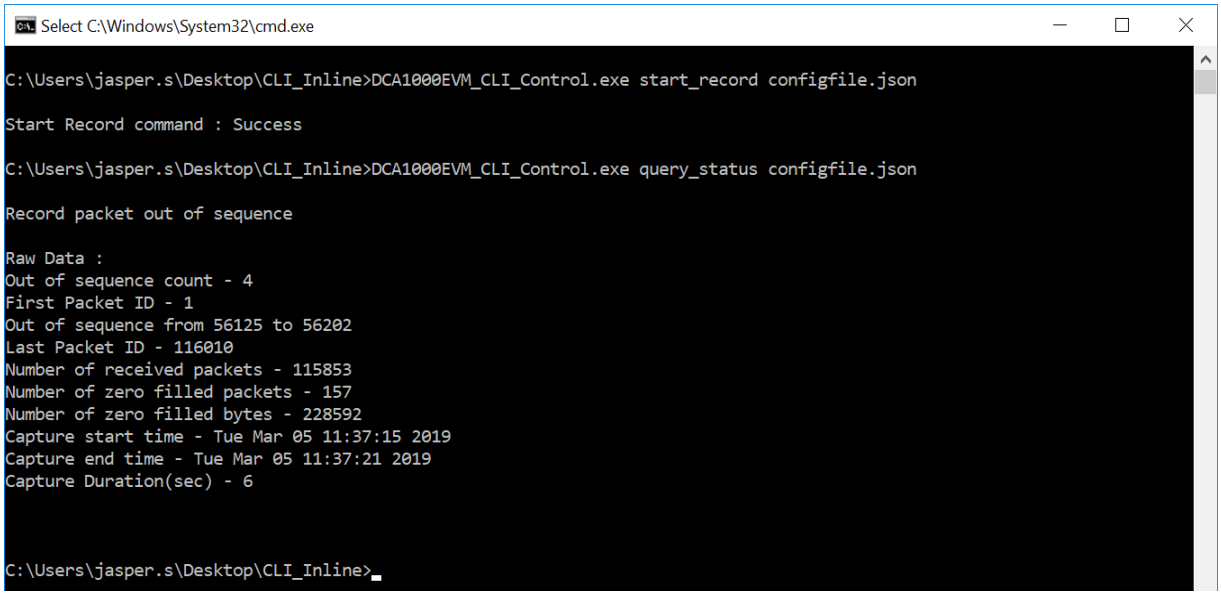
CLI_Control tool will execute this command to read the state of the record process and summary of record processing at the command execution point of time. This command can be executed when the CLI record process is running.

Command:

DCA1000EVM_CLI_Control.exe query_status configFile.json



This command will also display in the console and log in the logfile when any of the async status is received while recording is in progress.
Refer section 2.4 for record process states and 2.3 for system async status details.



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention
Query record process status	"ethernetConfig": { "DCA1000IPAddress": "192.168.33.180", "DCA1000ConfigPort": 4096, "DCA1000DataPort": 4098	DCA1000EVM_CLI_Control query_status configFile.json

	}	

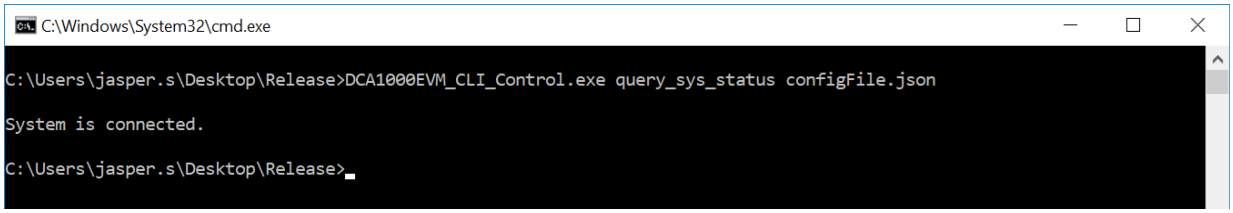
Table 13 Query record status calling convention

3.1.4.11 Query System Aliveness Status

CLI_Control tool will execute this command to verify DCA1000EVM system connectivity.

Command:

DCA1000EVM_CLI_Control.exe query_sys_status configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention
System connectivity	"ethernetConfig": { "DCA1000IPAddress": "192.168.33.180", "DCA1000ConfigPort": 4096, "DCA1000DataPort": 4098 }	DCA1000EVM_CLI_Control query_sys_status configFile.json

Table 14 Query DCA1000EVM system status calling convention

3.2 Linux

3.2.1 System Requirements

The CLI application is tested on Linux systems with below configurations.

- Operating System: Ubuntu 16xx LTS 64-bit OS
- RAM: 8 GB or above
- 1 GB-Ethernet port
- High speed storage for capture / store of the data files

DCA1000EVM CLI application to run in Linux platforms, needs the following files to be available at the same path.

3.2.2 Files Requirements To Execute CLI

Files required for CLI execution:

Files	Description
DCA1000EVM_CLI_Control	Executable file that does validation of user inputs and execution of configuration commands
DCA1000EVM_CLI_Record	Executable file that does validation of user inputs and execution of record commands
libRF_API.so	Dynamic library file that handles execution of configuration commands and recording of the captured data in files
configFile.json	JSON file for configuration from user. Refer section 2.1 for details on JSON file

- Open the command prompt and move to the directory where the above-mentioned files are downloaded.
- Make the files DCA1000EVM_CLI_Control and DCA1000EVM_CLI_Record as executables using '*sudo chmod +x DCA1000EVM_CLI_Control*' and '*sudo chmod +x DCA1000EVM_CLI_Record*' commands (if the files are not already in the executable mode).
- Update LD_LIBRARY_PATH using '*export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$pwd*' command.
- Start recording using command sequence mentioned in the following section 3.2.3

3.2.3 CLI Command Sequence

For successful recording of data from RADAR EVM sequence is given as follows

1. Configure FPGA
   - Ensure JSON config file (CLI) and Script config file (RADAR EVM) data format mode are in sync
   - Command - ***./DCA1000EVM_CLI_Control fpga configFile.json***
2. Configure record delay
   - Command - ***./DCA1000EVM_CLI_Control record configFile.json***
3. Start the record
   - Ensure JSON config file (CLI) and Script config file (RADAR EVM) data logging mode are in sync
   - Command - ***./DCA1000EVM_CLI_Control start_record configFile.json***

4. Stop the record after recording data
    - Command - ***./DCA1000EVM_CLI_Control stop_record configFile.json***

For successful record process, FPGA should be reconfigured in the following scenarios

- When the system is booted or rebooted
- When the FPGA or DCA1000EVM is reset
- On switching between multi-mode and raw mode

3.2.4 CLI Application Commands

CLI application suite includes two executables. They are: - DCA1000EVM_CLI_Control.exe and DCA1000EVM_CLI_Record.exe.

Of the two, the DCA1000EVM_CLI_Control.exe is the application providing options for the configuration commands and initiating the start and stop of the record process (DCA1000EVM_CLI_Record.exe). Following are some of the key aspects of the DCA100EVM_CLI_Control tool.

- CLI_Control tool supports blocking calls for configuration and Stop Record commands.
- CLI_Control tool supports non-blocking for the Start Record and record *query_status* commands.
- Multiple instances of the CLI_Control tool can be instantiated on the PC to control multiple EVMs while using unique UDP Ports for each EVM.
- It provides option to execute in “quiet” mode wherein there are no messages being displayed on the console.
- For all configuration commands, the configJsonFile should have valid DCA1000EVM system IP and config/record port numbers and the parameters corresponding to the executing commands.
- When CLI_Control is invoked to send continuous commands, CLI_Control will execute the command only when no other commands are in execution. If any other command is in progress, the CLI_Control will prompt the user to stop the already running process.

The generic execution flow of CLI_Control commands is as follows,

Calling Convention -
./DCA1000EVM_CLI_Control <command> [jsonCfgFile] [-q]

<command>: Supports following commands
[jsonCfgFile]: Json format input parameters text file path
[-q]: Quiet mode – No status display in the console

Commands supported by CLI_Control tool -

- fpga Configure FPGA
- eeprom Update EEPROM
- reset_fpga Reset FPGA

- reset_ar_device Reset AR Device
- start_record Start Record
- stop_record Stop Record
- record Configure Record delay
- dll_version Read DLL version
- fpga_version Read FPGA version
- cli_version Read CLI version
- query_status Read status of record process
- query_sys_status DCA1000EVM System aliveness
- -h List of commands supported

The generic execution flow of CLI_Control commands is as follows,

- User will initiate the control command through script or command line as *./DCA1000EVM_CLI_Control* <command> *configFile.json*
- CLI_Control tool reads the shared memory for checking whether any record process is running.
- If record process is running, CLI_Control tool will prompt the user to stop the already running record process. The CLI_Control tool logs the information in a log file with timestamp and will be terminated.
- If no process is running, CLI_Control tool validates the input parameters from the configJsonfile.
- CLI_Control initializes the config port ethernet connection and send <command> and wait for response packet till timeout.
- CLI_Control displays the command status to the user in the console. The CLI_Control tool also stores the configuration command status in a log file with timestamp. The exit code of the last executed process can also be read using OS system calls based on the calling application.

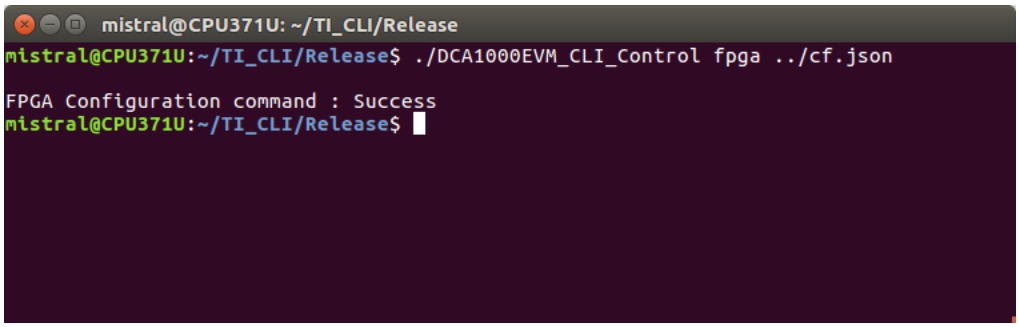
3.2.4.1 Configure FPGA

CLI_Control tool will execute this command to configure the DCA1000EVM with the following mode configuration

1. Logging Mode (*dataLoggingMode*) – RAW/MULTI
2. LVDS Mode (*lvdsMode*) - 4 lane/2 lane
3. Data Transfer Mode (*dataTransferMode*) – LVDS Capture/Playback
4. Data Capture Mode (*dataCaptureMode*) – SD Card Storage/Ethernet Streaming
5. Data Format Mode (*dataFormatMode*) – 12/14/16 bit
6. Timer (Not configurable using JSON file)

Command:

./DCA1000EVM_CLI_Control fpga configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention

Configure FPGA	"dataLoggingMode": "raw",
	"dataTransferMode": "LVDSCapture",
	"dataCaptureMode": "ethernetStream",
	"lvdsMode": 1,
	"dataFormatMode": 3,
	"ethernetConfig": {
	"DCA1000IPAddress": "192.168.33.180",
	"DCA1000ConfigPort": 4096,
	"DCA1000DataPort": 4098
	}
		DCA1000EVM_CLI_Control fpga configFile.json

Table 15 Configure FPGA calling convention

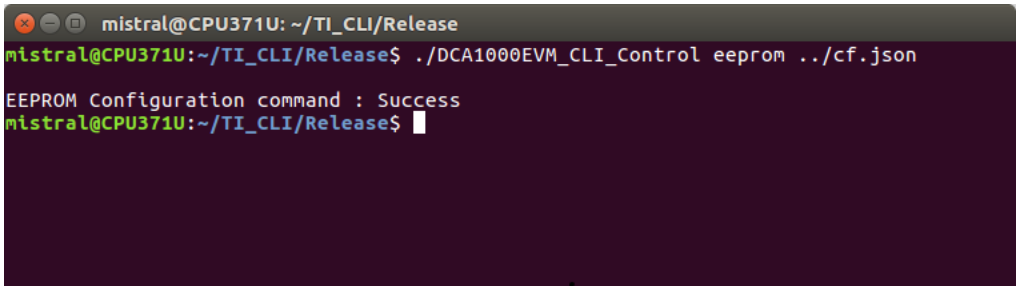
3.2.4.2 Configure EEPROM

CLI_Control tool will execute this command to reconfigure the IP address of the DCA1000EVM with the following configuration

1. MAC ID (*ethernetConfigUpdate -> DCA1000MACAddress*)
2. PC IP Address (*ethernetConfigUpdate -> systemIPAddress*)
3. Board IP Address (*ethernetConfigUpdate -> DCA1000IPAddress*)
4. Record port number (*ethernetConfigUpdate -> DCA1000DataPort*)
5. Configuration port number (*ethernetConfigUpdate -> DCA1000ConfigPort*)

Command:

./DCA1000EVM_CLI_Control eeprom configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention

Configure EEPROM	"ethernetConfig": {
	"DCA1000IPAddress": "192.168.33.180",
	"DCA1000ConfigPort": 4096,
	"DCA1000DataPort": 4098
	},
	"ethernetConfigUpdate": {
	"systemIPAddress": "192.168.33.30",
	"DCA1000IPAddress": "192.168.33.180",
	"DCA1000MACAddress": "12.34.56.78.90.12",
	"DCA1000ConfigPort": 4096,
	"DCA1000DataPort": 4098
	}
		DCA1000EVM_CLI_Control eeprom configFile.json

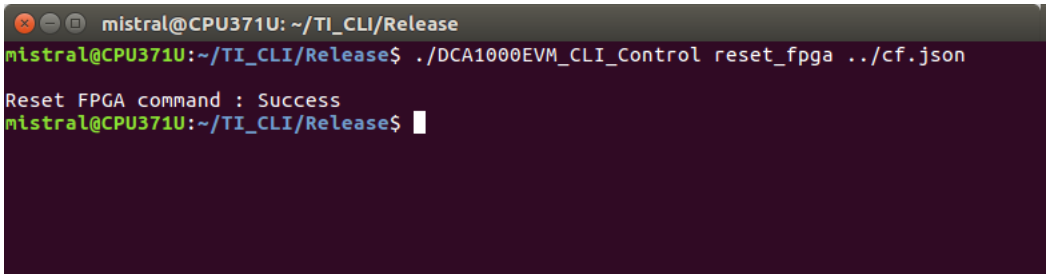
Table 16 Configure EEPROM calling convention

3.2.4.3 Reset FPGA

CLI_Control tool will execute this command to reset the DCA1000EVM FPGA.

Command:

./DCA1000EVM_CLI_Control resert_fpga configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention
Reset FPGA	"ethernetConfig": { "DCA1000IPAddress": "192.168.33.180", "DCA1000ConfigPort": 4096, "DCA1000DataPort": 4098 }	DCA1000EVM_CLI_Control reset_fpga configFile.json

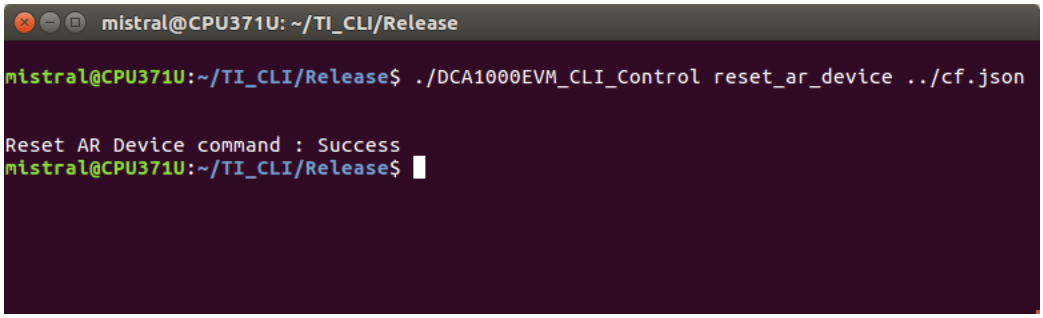
Table 17 Reset FPGA calling convention

3.2.4.4 Reset RADAR EVM

CLI_Control tool will execute this command to reset RADAR EVM.

Command:

./DCA1000EVM_CLI_Control reset_ar_device configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention
Reset RADAR EVM	"ethernetConfig": { "DCA1000IPAddress": "192.168.33.180", "DCA1000ConfigPort": 4096, "DCA1000DataPort": 4098 }	DCA1000EVM_CLI_Control reset_ar_device configFile.json

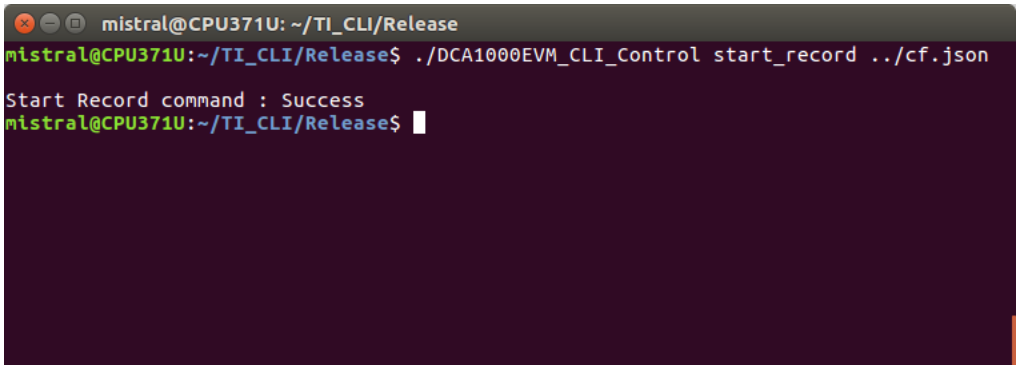
Table 18 Reset RADAR EVM calling convention

3.2.4.5 Start record

CLI_Control tool will execute the command to verify DCA1000EVM connectivity and invoke the CLI Record tool to start the recording.

Command:

./DCA1000EVM_CLI_Control start_record configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention
Start Record	"dataLoggingMode": "raw", "ethernetConfig": { "DCA1000IPAddress": "192.168.33.180", "DCA1000ConfigPort": 4096, "DCA1000DataPort": 4098 }, "captureConfig": { "fileBasePath": "C:\\Users\\CLI_Inline", "filePrefix": "inline",	DCA1000EVM_CLI_Control start_record configFile.json

	"maxRecFileSize_MB": 1024, "sequenceNumberEnable": 1, "captureStopMode": "infinite", "bytesToCapture": 4000, "durationToCapture_ms": 4000, "framesToCapture": 40 }, "dataFormatConfig": { "MSBToggle": 0, "reorderEnable": 1,	

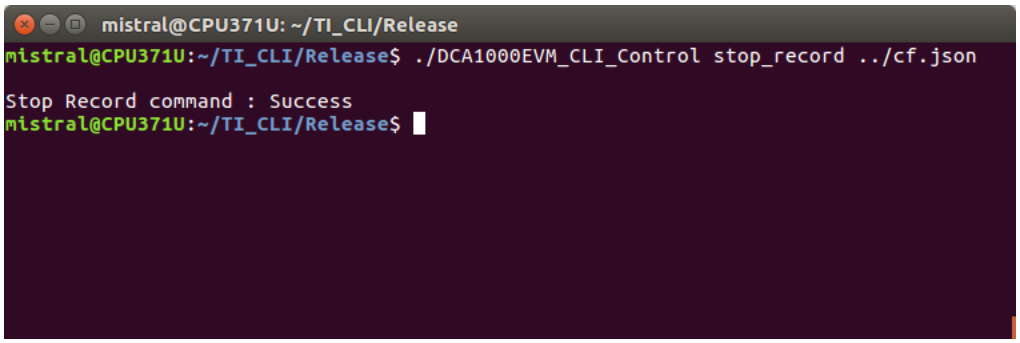
Table 19 Start Record calling convention

3.2.4.6 Stop record

CLI_Control tool will execute this command to stop the recording.

Command:

./DCA1000EVM_CLI_Control stop_record configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention

Stop Record (Based on user stop)	"ethernetConfig": {
	"DCA1000IPAddress": "192.168.33.180",
	"DCA1000ConfigPort": 4096,
	"DCA1000DataPort": 4098
	}
		DCA1000EVM_CLI_Control stop_record configFile.json

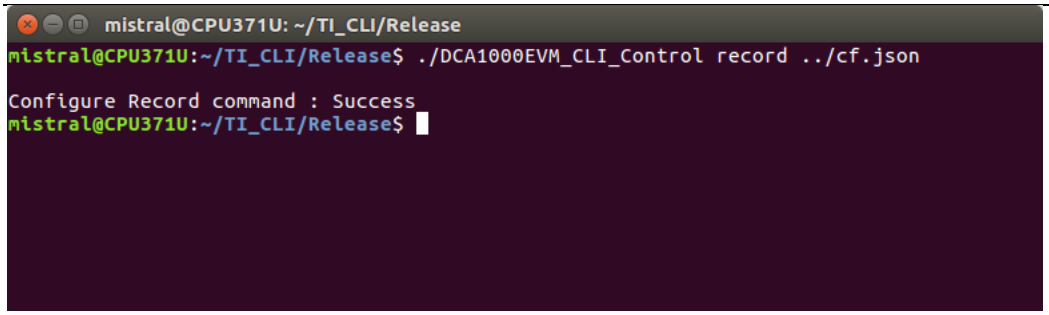
Table 20 Stop Record calling convention

3.2.4.7 Configure record delay

CLI_Control tool will execute this command to configure delay between record packets.

Command:

./DCA1000EVM_CLI_Control record configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention

Configure Record delay	"packetDelay_us": 5,
	"ethernetConfig": {
	"DCA1000IPAddress": "192.168.33.180",
	"DCA1000ConfigPort": 4096,
	"DCA1000DataPort": 4098
	}
		DCA1000EVM_CLI_Control record configFile.json

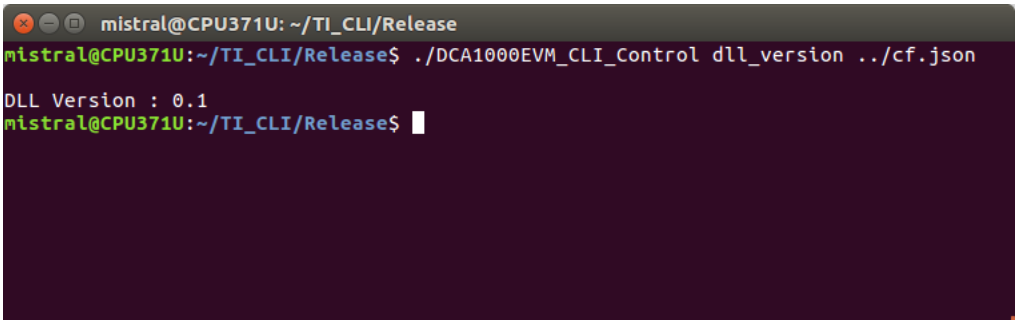
Table 21 Configure record delay calling convention

3.2.4.8 Read DLL version

CLI_Control tool will execute this command to read DLL version.

Command:

./DCA1000EVM_CLI_Control dll_version configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention
Read DLL version	NA	DCA1000EVM_CLI_Control dll_version

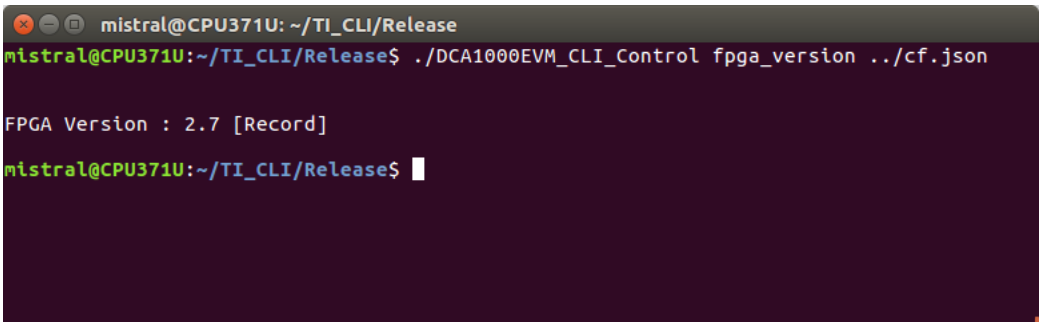
Table 22 Read DLL version calling convention

3.2.4.9 Read FPGA version

CLI_Control tool will execute this command to read FPGA version of DCA1000EVM.

Command:

./DCA1000EVM_CLI_Control fpga_version configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention
Read FPGA version	"ethernetConfig": { "DCA1000IPAddress": "192.168.33.180", "DCA1000ConfigPort": 4096, "DCA1000DataPort": 4098 }	DCA1000EVM_CLI_Control fpga_version configFile.json

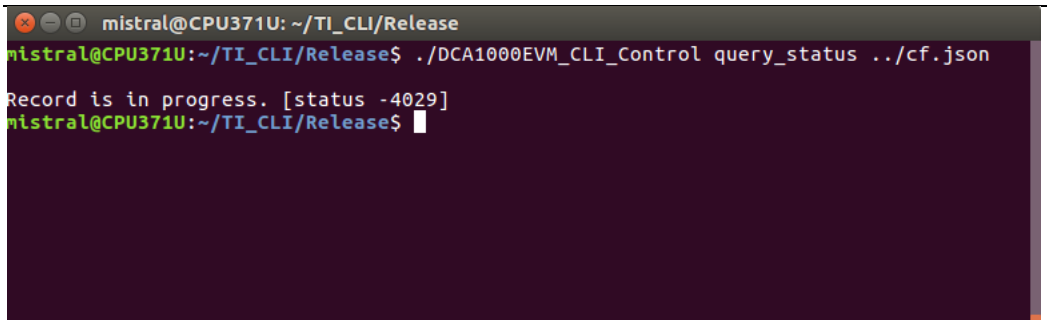
Table 23 Read FPGA version calling convention

3.2.4.10 Query Record Process Status

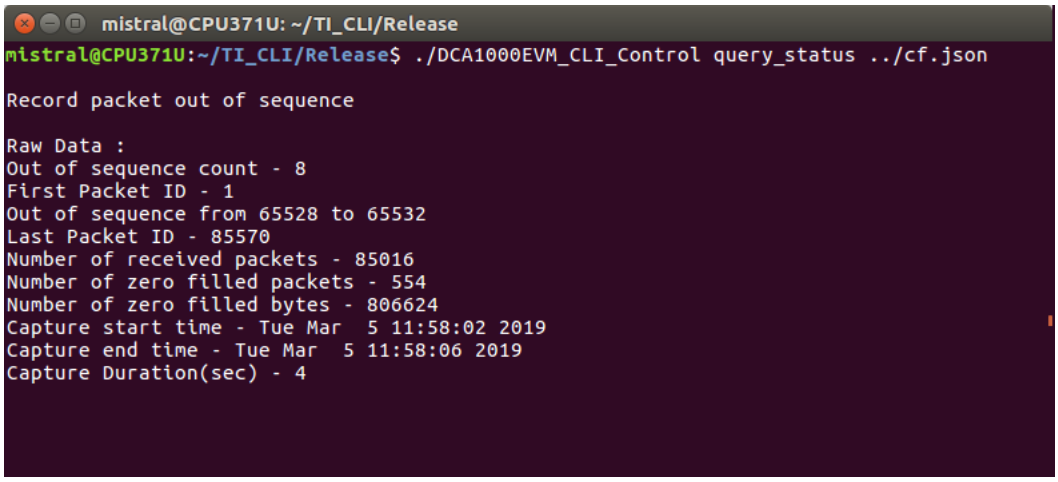
CLI_Control tool will execute this command to read the state of the record process and summary of record processing at the command execution point of time. This command can be executed when the CLI record process is running.

Command:

./DCA1000EVM_CLI_Control query_status configFile.json



This command will also display in the console and log in the logfile when any of the async status is received while recording is in progress.
Refer section 2.4 for record process states and 2.3 for system async status details.



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention
Query record process status	"ethernetConfig": { "DCA1000IPAddress": "192.168.33.180", "DCA1000ConfigPort": 4096, "DCA1000DataPort": 4098 }	DCA1000EVM_CLI_Control query_status configFile.json

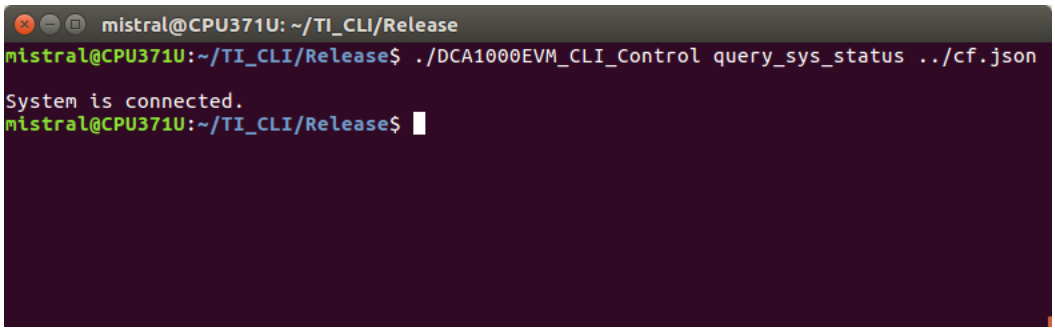
Table 24 Query record status calling convention

3.2.4.11 Query System Aliveness Status

CLI_Control tool will execute this command to verify DCA1000EVM system connectivity.

Command:

./DCA1000EVM_CLI_Control query_sys_status configFile.json



JSON File data and command calling conventions:

Command	JSON File data	CLI calling convention
System connectivity	"ethernetConfig": { "DCA1000IPAddress": "192.168.33.180", "DCA1000ConfigPort": 4096, "DCA1000DataPort": 4098 }	DCA1000EVM_CLI_Control query_sys_status configFile.json

Table 25 Query DCA1000EVM system status calling convention

4 DCA1000EVM CLI Building Instructions

The source code folder structure and building steps for Windows and Linux platforms are explained in this section.

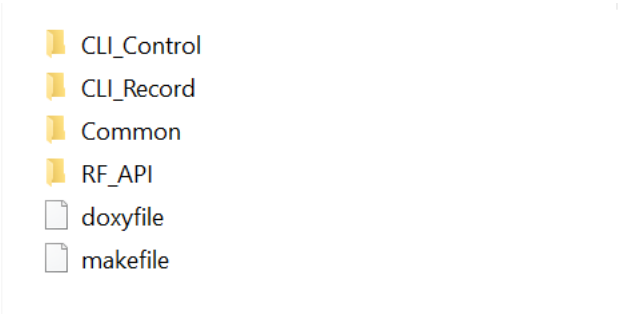
4.1 Windows

4.1.1 Files to build Windows binaries

Following are the files required to build DCA1000EVM CLI application

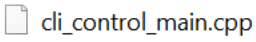
- CLI source codes
- Make compiler tool or IDEs

DCA1000EVM CLI application source code folder structure



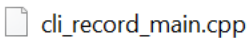
- CLI_Control Folder – Files to handle the user inputs for configuration and start and stop of the record process
- CLI_Record Folder – Files to handle the user inputs for record process
- Common folder – Files to handle OSAL services like shared memory and timer event, validation of JSON config file and common definitions
- RF_API folder – Files to handle UDP socket communication through ethernet and generation of captured data files
- doxyfile – Script to build and generate doxygen format documentation
- makefile – Script to build and generate CLI binares

CLI Control Tool source code folder structure



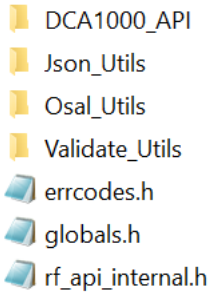
- cli_control_main.cpp – File to handle the user inputs for configuration and start and stop of the record process

CLI Record Tool source code folder structure



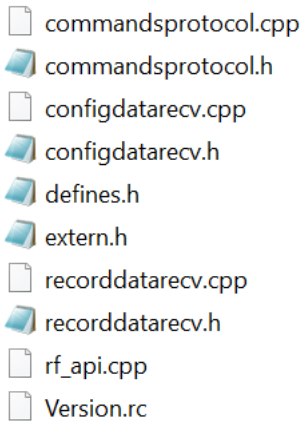
- cli_record_main.cpp – File to handle the user inputs for record process

Common source code folder structure



- DCA1000_API folder – Contains declarations of exported APIs and defines, which other applications can use it for integration.
- Json_Utils folder – Utility files for handling, parsing JSON data (user input)
- OSAL_Utils folder – Utility files for handling shared memory, timer event and socket APIs which is specific to the operating system
- Validate_Utils folder – Utility files for validation of input parameters for a particular command
- errcodes.h – Contains error code definitions
- rf_api_internal.h – Contains API decalararions and defines specific to the DCA1000EVM CLI application

API DLL Source code folder structure



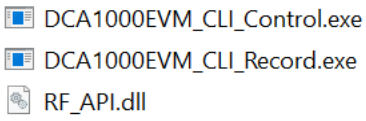
4.1.2 Steps to build Windows binaries

- CLI application is a light weight utility which can be built using compiler make tools like mingw / gnuwin32 or IDE tools like Visual Studio / Qt framework.

Build Instructions for CLI executables

- Open the command prompt and set the directory in which all the DCA1000EVM files are installed.
- Run the makefile script using '*make*' or '*mingw32-make.exe*' commands.
- *Release* folder will be created in the current directory and following listed binaries will be generated inside the *Release* folder.
- To clean up the binaries, run '*make clean*' or '*mingw32-make.exe clean*' command to remove the files. This command will retain the *Release* folder and will delete only CLI_Control exe, CLI_Record exe and RF_API.dll binary files.

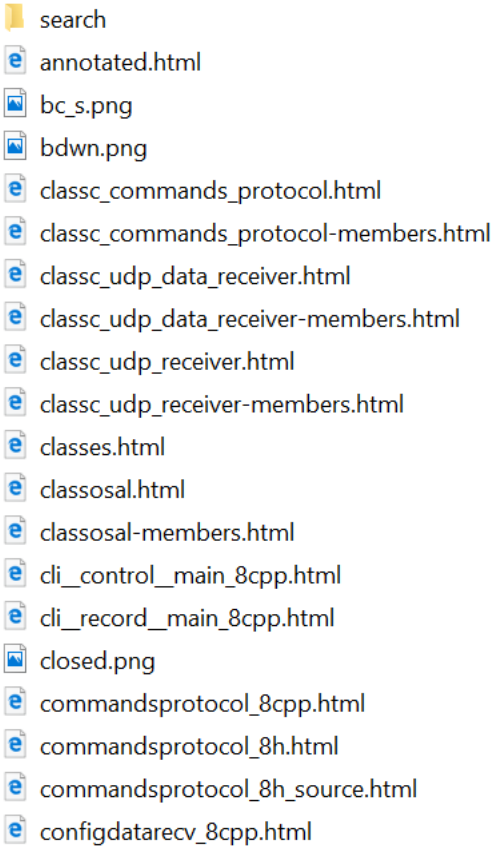
Release folder structure



Build Instructions for Doxygen documentation

- Open the command prompt and set the directory in which all the DCA100EVM source files are installed.
- Run the doxyfile script using '*Doxygen'* command.
- *Doxygen_CLI* folder will be created in the current directory and html files will be generated inside the *Doxygen_CLI* folder.

Doxygen_CLI folder structure



4.1.3 Troubleshooting steps

TroubleShooting steps to manually kill record process:

- Run task manager and kill the record process.

- Run the command '*DCA1000EVM_CLI_Control.exe stop_record configFile.json*' using command prompt to update the record process status so that subsequent commands will be allowed to execute.

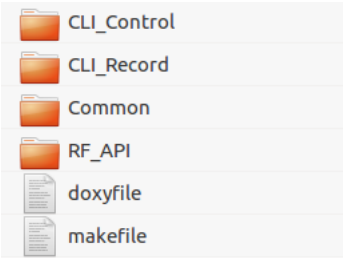
4.2 Linux

4.2.1 Files to build Linux binaries

Following are the files required to build DCA1000EVM CLI application

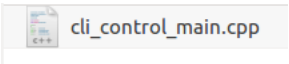
- CLI source codes
- Make compiler tool or IDEs

DCA1000EVM CLI application source code folder structure



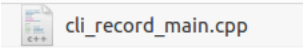
- CLI_Control Folder – Files to handle the user inputs for configuration and start and stop of the record process
- CLI_Record Folder – Files to handle the user inputs for record process
- Common folder – Files to handle OSAL services like shared memory and timer event, validation of JSON config file and common definitions
- RF_API folder – Files to handle UDP socket communication through ethernet and generation of captured data files
- doxyfile – Script to build and generate doxygen format documentation
- makefile – Script to build and generate CLI binares

CLI_Control Tool source code folder structure



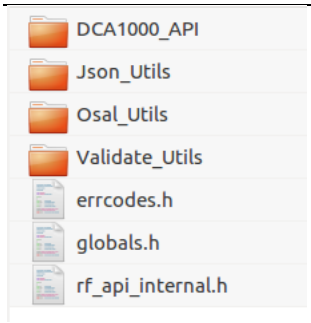
- cli_control_main.cpp – File to handle the user inputs for configuration and start and stop of the record process

CLI Record Tool source code folder structure



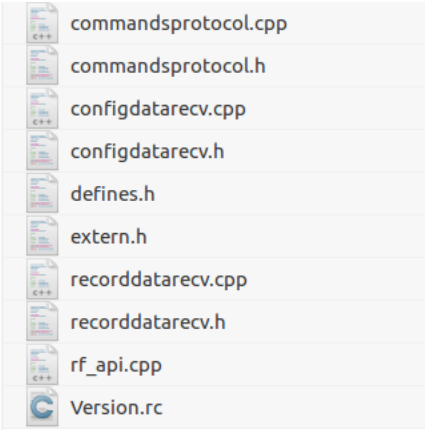
- cli_record_main.cpp – File to handle the user inputs for record process

Common source code folder structure



- DCA1000_API folder – Contains declarations of exported APIs and defines, which other applications can use it for integration.
- Json_Utils folder – Utility files for handling, parsing JSON data (user input)
- OSAL_Utils folder – Utility files for handling shared memory, timer event and socket APIs which is specific to the operating system
- Validate_Utils folder – Utility files for validation of input parameters for a particular command
- errcodes.h – Contains error code definitions
- rf_api_internal.h – Contains API decalararions and defines specific to the DCA1000EVM CLI application

API DLL Source code folder structure



4.2.2 Steps to build Linux binaries

- CLI application is a light weight utility which can be built using compiler make tools or IDE tools like Qt framework.

Build Instructions for CLI executables

- Open the command prompt and set the directory in which all the DCA1000EVM files are installed.
- Run the makefile script using '*make*' command.
- *Release* folder will be created in the current directory and following listed binaries will be generated inside the *Release* folder.

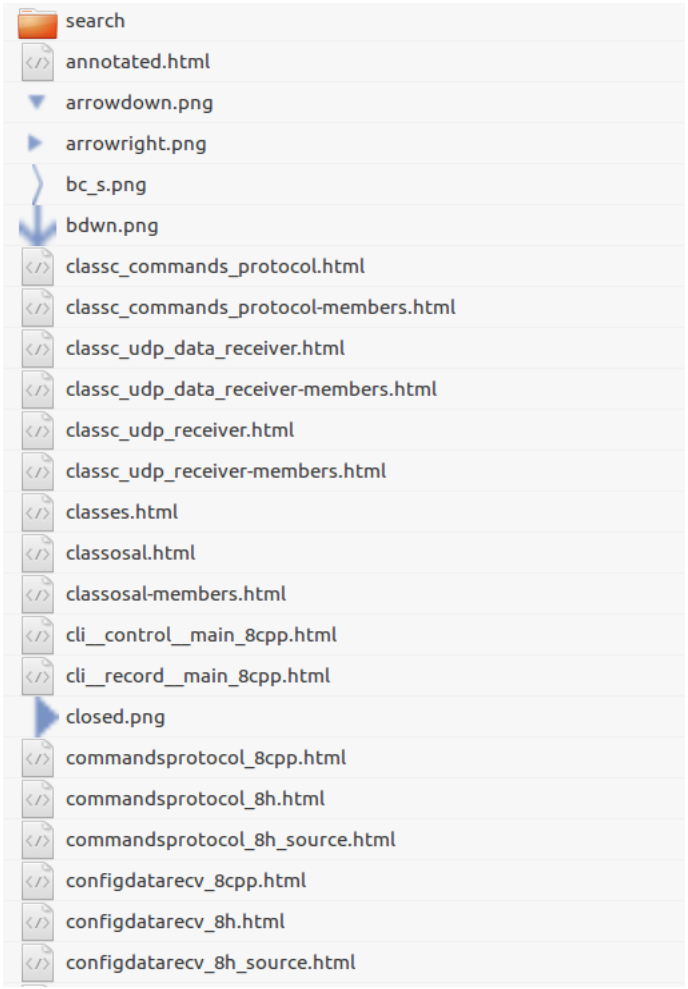
- To clean up the binaries, run '*make clean*' command to delete the files. This command will retain the *Release* folder and will delete only CLI_Control exe, CLI_Record exe and libRF_API.so binary files.

Release folder structure

Build Instructions for Doxygen documentation

- Open the command prompt and set the directory in which all the DCA100EVM files are installed.
- Run the doxyfile script using '*doxygen*' command.
- *Doxygen_CLI* folder will be created in the current directory and html files will be generated inside the *Doxygen_CLI* folder.

Doxygen_CLI folder structure



4.2.3 Troubleshooting steps

TroubleShooting steps to manually kill record process:

- Run '*ps -ae*' to find the *DCA1000EVM_CLI_Record* process ID.
- Run '*sudo kill -9 <process ID>*' to kill the record process.
- Run the command '*./DCA1000EVM_CLI_Control stop_record configFile.json*' using command prompt to update the record process status so that subsequent commands will be allowed to execute.