TX Phase Shifter Calibration Theory

For best performance of transmitter, beam-formed/steered, antenna arrays, the phase of the waveform incident to the antenna array must be well controlled. If the phase is not well controlled, performance compared to what is theoretically possible for the array will suffer, resulting in lower main-lobe beam power, less directional accuracy, and lower overall SNR. To ensure the best possible phase control a system-level calibration must be performed which incorporates the AWRx TX paths, TX phase-shifters and PCB structures.

This document describes an example procedure for gathering this calibration dataset, calculating the phase offset calibration matrix, and applying the calibration matrix to a TX beamforming example setup. All examples are run on the Cascade Radar RF (MMWCAS-RF-EVM) and Cascade Radar Host (MMWCAS-DSP-EVM) evaluation system and make use of the 12 TX azimuth antenna array. Automation scripts written in the mmWave Studio Lua environment and Matlab are used to execute each step of the calibration process.

Users of these scripts and calibration process are expected to be familiar with the basic operation of the MMWCAS-RF-EVM/MMWCAS-DSP-EVM and with the mmWave Studio environment. For a full description of the Cascade EVM mmWave Studio and Matlab post-processing example environment setup, please reference the *mmWave Studio Cascade User Guide* located in the mmWave Studio install directory under *\docs\mmwave_studio_cascade_user_guide.pdf*

Table of Contents

1 Summary

There are multiple components which contribute to phase offset error in the AWR based system. Phase offset is contributed by the AWR device package, PCB transmission-lines, and antenna. Phase offset will vary according to operating temperature and operating frequency. Closely matching the route delay from the AWR device package to the antenna can eliminate much of the PCB and package phase offset contribution, but some will still remain even with well length matched routing.

Additionally, phase offset is also contributed by the AWR transmit phase-shifters when being used to intentionally offset phase to each TX path. Ideally, each phase shifter contributes exactly 5.625 degree/LSB. However, the exact phase offset contributed by the TX phase-shifters will slightly vary, as a function of process corner, temperature and operating frequency.

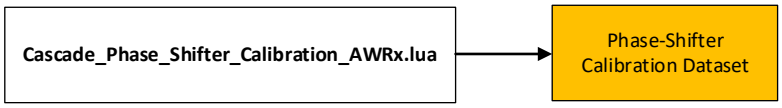
The combined phase offset error from phase-shifter, package, and PCB path should be calibrated to ensure best performance of the AWR transmitter antenna array for beam-forming/steering. The basic flow for calibration in these example scripts can be outlined as follows:

1. Measure the phase from the 2D-FFT (0-velocity bin) peak of a fixed calibration target for each TX channel, RX channel, (all virtual channels) and phase shifter programmable offset combination, across the AWR system.
    a. In the example scripts here, this calibration dataset is collected by taking 64 individual TDMA-MIMO data sets, with each dataset including all 12 TX (turned on one at a time, TDMA fashion). Each dataset has a different phase-shifter value applied to all 12 TX.
    b. This results in data from each virtual channel and phase-shifter combination
2. Calculate the phase offset, relative to the 0 phase shifter setting for each TX channel as observed by each RX channel.
    a. Absolute phase measured from the calibration target will appear somewhat random as is dependent on the target distance
    b. Relative phase offset to this a reference phase value will appear well controlled, with some excursions
    c. Having data from all RX channels may be redundant, but it is also possible to identify outlier virtual channels in this manner.
    d. Virtual channel gain/phase calibration for RX calibration can also be done with this same dataset (RX virtual channel gain/phase calibration is not covered in this document).
3. At this point, there exists a [*Number RX* x *Number TX* x *Number Phase Shifts*] matrix of phase offsets.
    a. The data could be averaged across RX, or a single reference RX could be chosen.
    b. In this example, a single reference RX channel is selected, and each -180 deg to +180 degree phase offset measurement in this RX channel is then mapped to a 0 to 360 degree range, to correspond to the controllable phase shifter values.
    c. This results in a reference phase offset calibration matrix of dimensions [*Number TX* x *TX* x *Number Phase Shifts*].

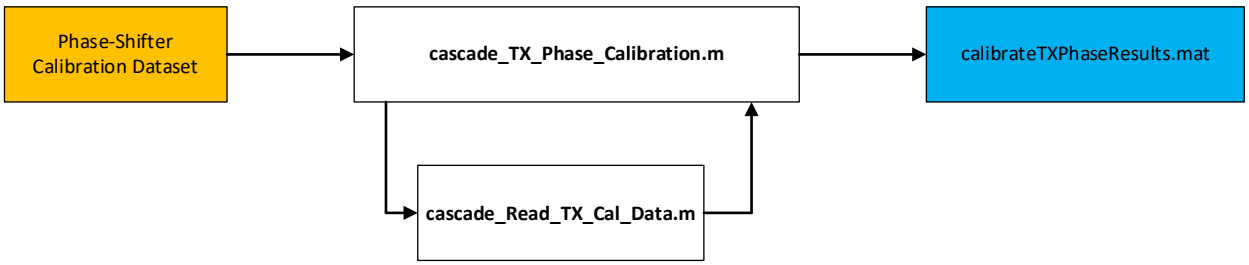
4. For a given phase-shifter enabled application, such as TX Beam-Steering, an ideal phase offset matrix can be produced as a function of the physical geometry, and antenna array theory (antenna spacing, wavelength, and desired beam-tilt angle).
5. This ideal phase offset matrix is then compared with the calibrated offset matrix and a mapping between the two matrices is created.
    a. This mapping, or look-up-table (LUT), allows the user to select a given state of the antenna phase offsets (for example a desired beam-tilt angle), and be presented with the minimal error phase shifter programming values to program for each AWR TX channel.

The process outlined above is accomplished in this example through a set of mmWave Studio Lua scripts and Matlab scripts and functions. This can be seen below:

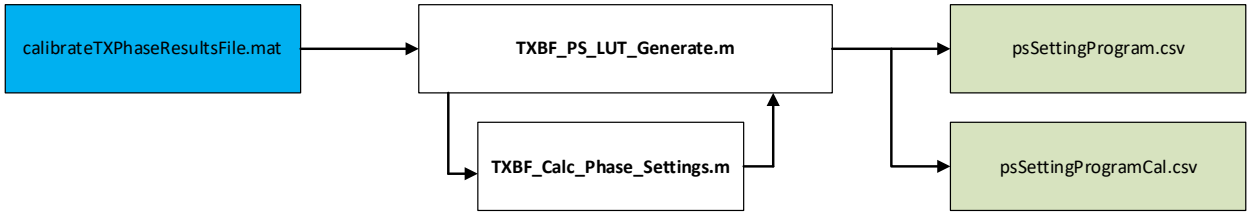
- First, a mmWave Studio Lua script, *Cascade_Phase_Shifter_Calibration_AWRx.lua*, is used for calibration data collection.



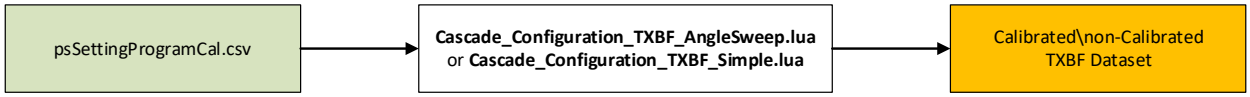
- A Matlab script, *cascade_TX_Phase_Calibration.m* is then used for post-processing of the calibration data set, creating a phase-offset matrix per virtual channel, and per TX phase shifter programmed value.



- Another Matlab script, *TXBF_LUT_Generate.m* is then used to create a phase shifter programming lookup table which translates beam-steering angle to calibrated phase shifter programming values.



- Next, two example mmWave Studio Lua scripts can be used to take TX beam-forming verification data on the same calibration target scene.
    - *Cascade_Configuration_TXBF_Simple.lua* sets up a fixed beam-steering angle using the azimuth antenna of the MMWCAS-RF-EVM and collects the resulting data.
    - *Cascade_Configuration_TXBF_AngleSweep.lua* sweeps through an arc of beam-steering angles, collecting data at each beam-steering offset angle. When the data is processed this results in a beam angle vs. RX power graph which can be compared to the non-calibrated or non-TXBF system performance.



- Finally, another Matlab script, cascade_TXBF_Verification.m is used to plot the dataset collected by the Cascade_Configuration_TXBF_AngleSweep.lua.

2 Hardware and Software Description

The MMWCAS-RF-EVM and MMWCAS-DSP-EVM cascade radar evaluation system is utilized for these examples. Please reference the ***mmWave Studio Cascade User Guide*** for information on initial setup of the AWR cascade evaluation system.

2.1 Hardware and Software Needed

- MMWCAS-RF-EVM and MMWCAS-DSP-EVM assembly
- Corner-reflector and suitable test range
    - minimal clutter within FMCW range, to easily identify corner reflector target
    - minimal range to ensure plane-wave received back at AWR RX
- Host PC running either mmWave Studio 2.1 (or greater) for AWR1243P support, or mmWave Studio 3.1 (or greater) for AWR2243P support
    - Ensure that the example MIMO configuration and Matlab post-processing example has already been run and is functioning as expected.
    - This TX phase shifter calibration set builds on top of the Matlab TDMA MIMO processing chain examples

2.2 Software Script Description

The mmWave Studio Lua script system is used to automate all AWR device configuration and data collection in these examples. A set of Matlab scripts are used to post-process and graph the resulting datasets.

2.2.1 mmWave Studio Lua Configuration and Data Collection Scripts

The following three configuration and data capture Lua scripts are summarized here.

- *Cascade_Phase_Shifter_Calibration_AWRx.lua*
- *Cascade_Configuration_TXBF_Simple.lua*
- *Cascade_Configuration_TXBF_AngleSweep.lua*

2.2.1.1 Cascade_Phase_Shifter_Calibration_AWRx.lua

This is a mmWave Studio Lua automation script which gathers a TX phase shifter calibration dataset. This script handles configuration of the MMWCAS-RF-EVM AWRx devices and MMWCAS-DSP-EVM data capture.

- Using a TDMA-MIMO frame structure, this script takes data on all 4 AWRx devices, across all TX/RX virtual channels, sweeping the 6-bit phase-shifter offset through the 0 to 63 setting range
- Each dataset consists of a few TDMA-MIMO frames with the same phase shifter value applied to each TX channel.
- 64 datasets are captured with each dataset applying a different phase-shifter offset programing for all TX channels until all phase-shifter settings are covered
- Data is stored in the default */mmWaveStudio/Postproc/* folder location formatted as */TX_PS_CAL_phaseShiftValueX*
    - X is the phase-shifter offset (00 through 63)
    - This formatting is used later by the *cascade_TX_Phase_Calibration.m* Matlab script to read in the calibration dataset
- This calibration data collection script is based off the example TDMA-MIMO configuration and data capture Lua scripts. The same basic configuration settings must be setup. These are all located in lines 26 through 103 of the script:

    - RadarDevice
    - cascade_mode_list
    - metaImagePath
    - TDA_IPAddress
    - Profile Configuration
    - Frame Configuration
    - Data Capture Configuration

```
----------------------------------------User Constants--------------------------------------------------

dev_list                        =   {1, 2, 4, 8}                    -- Device map
RadarDevice                     =   {1, 1, 1, 1}                    -- {dev1, dev2, dev3, dev4}, 1: Enable, 0: Disable

cascade_mode_list               =   {1, 2, 2, 2}                    -- 0: Single chip, 1: Master, 2: Slave

-- F/W Download Path

-- Uncomment the next line if you wish to pop-up a dialog box to select the firmware image file
-- Otherwise, hardcode the path to the firmware metaimage below
-- By default, the firmware filename is: xwr12xx_metaImage.bin
--
-- metaImagePath                    =   RSTD.BrowseForFile(RSTD.GetSettingsPath(), "bin", "Browse to .bin file")

--metaImagePath                   =   "C:\\ti\\mmwave_dfp_02_00_08_02\\firmware\\xwr22xx_metaImage.bin"
metaImagePath                   =   "C:\\ti\\mmwave_dfp_02_01_05_00\\firmware\\xwr22xx_metaImage.bin"

-- IP Address for the TDA2 Host Board
-- Change this accordingly for your setup

--TDA_IPAddress                  =   "128.247.61.164"
TDA_IPAddress                   =   "192.168.33.180"

-- Device map of all the devices to be enabled by TDA
-- 1 - master ; 2- slave1 ; 4 - slave2 ; 8 - slave3

deviceMapOverall                =   RadarDevice[1] + (RadarDevice[2]*2) + (RadarDevice[3]*4) + (RadarDevice[4]*8)
deviceMapSlaves                 =   (RadarDevice[2]*2) + (RadarDevice[3]*4) + (RadarDevice[4]*8)
```

```
---------------------------------------- Sensor Configuration ----------------------------------------------

-- The sensor configuration consists of 3 sections:
--      1) Profile Configuration (common to all 4 AWR devices)
--      2) Chirp Configuration (unique for each AWR device - mainly because TXs to use are different for each chirp)
--      3) Frame Configuration (common to all 4 AWR devices, except for the trigger mode for the master)
-- Change the values below as needed.

-- Profile configuration
local profile_indx              =   0
local start_freq                =   77                          -- GHz
local slope                     =   30                          -- MHz/us
local idle_time                 =   100                         -- us
local adc_start_time            =   6                           -- us
local adc_samples               =   256                         -- Number of samples per chirp
local sample_freq               =   10000                       -- ksps
local ramp_end_time             =   60                          -- us
local rx_gain                   =   48                          -- dB
local tx0OutPowerBackoffCode    =   0
local tx1OutPowerBackoffCode    =   0
local tx2OutPowerBackoffCode    =   0
local tx0PhaseShifter           =   0
local tx1PhaseShifter           =   0
local tx2PhaseShifter           =   0
local txStartTimeUSec           =   0
local hpfCornerFreq1            =   0                           -- 0: 175KHz, 1: 235KHz, 2: 350KHz, 3: 700KHz
local hpfCornerFreq2            =   0                           -- 0: 350KHz, 1: 700KHz, 2: 1.4MHz, 3: 2.8MHz

-- Frame configuration
local start_chirp_tx            =   0
local end_chirp_tx              =   0
local nchirp_loops              =   128                         -- Number of chirps per frame
local nframes_master            =   3                           -- Number of Frames for Master
local nframes_slave             =   3                           -- Number of Frames for Slaves
local Inter_Frame_Interval      =   100                         -- ms
local trigger_delay             =   0                           -- us
local nDummy_chirp              =   0
local trig_list                 =   {1,2,2,2}                   -- 1: Software trigger, 2: Hardware trigger


-- Data capture
local capture_time              =   3000                            -- ms
local inter_loop_time           =   2000                            -- ms
local num_loops                 =   1
local n_files_allocation        =   0
local data_packaging            =   0                               -- 0: 16-bit, 1: 12-bit
local capture_directory         =   "" -- setup below in code
local num_frames_to_capture     =   0                               -- 0: default case; Any positive value - number of frames to capture
local framing_type              =   1                               -- 0: infinite, 1: finite
```

Figure 1 - Cascade_Phase_Shifter_Calibration_AWRx.lua script configuration variables

2.2.1.2 *Cascade_Configuration_TXBF_Simple.lua*

This is a mmWave Studio Lua automation script which sets up a static (single, fixed angle) TX beam-form example using basic chirp configuration. No AWR advanced frame configuration options are used. This script is meant to be used to validate the pre-calibration vs. post-calibration beam-forming performance in as simple of a setup as possible.

The script runs through a basic cascade setup with the requested chirp configuration and frame configuration settings applied.

The variable *psCalLUT* is a [9 TX] x [Number of Angles] phase-shifter calibration matrix which maps a specific beam-steering angle to a set of calibrated phase shifter programming values across the cascaded device array. This matrix is the output of the Matlab based calibration scripts. Each row is a specific steering angle and each column is the specific phase programming value for a specific transmitter channel.

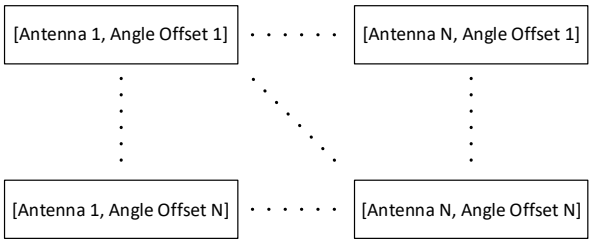


Figure 2 – *psCalLUT* matrix format

```
psCalLUT = {
{0,  30,  56,  22,  56, 25,   55,  23,  54}, --> angle = 75 deg
{0,  32,  60,  30,  4,  36,    6,  38,  8 },
{0,  34,  63,  32,  12, 46,   19,  53,  25},
{0,  36,   6,  33,  20, 56,   31,   6,  42},
{0,  39,  10,  39,  28,  6,   44,  20,  59},
{0,  41,  14,  45,  37, 16,   57,  35,  14},
{0,  43,  19,  51,  45, 27,    8,  50,  31},
{0,  45,  23,  6,   53, 37,   21,   3,  48},
{0,  47,  27,  12,  62, 48,   33,  18,  3 },
{0,  49,  32,  18,  10, 60,   46,  33,  21},
{0,  52,  36,  24,  18,  8,   59,  48,  38},
{0,  54,  40,  31,  26, 19,   10,   1,  55},
{0,  56,  45,  32,  35, 29,   23,  17,  10},
{0,  58,  49,  35,  44, 40,   36,  32,  27},
{0,  60,  53,  41,  52, 51,   49,  47,  45},
{0,   0,   0,   0,  0,   0,    0,   0,  0 }, --> angle = 90 deg
{0,   2,  62,  54,  9,  11,   14,  15,  18},
{0,   4,  4,   60,  17, 21,   26,  30,  35},
{0,   7,  8,   14,  25, 32,   39,  45,  52},
{0,   9,  12,  21,  34, 43,   52,  61,  8 },
{0,  11,  17,  27,  42, 54,    3,  14,  25},
{0,  13,  21,  32,  50,  3,   16,  29,  42},
{0,  15,  26,  33,  59, 14,   29,  44,  59},
{0,  17,  30,  38,  7,  24,   42,  59,  14},
{0,  19,  34,  44,  15, 35,   54,  12,  31},
{0,  21,  39,  50,  23, 45,    5,  27,  49},
{0,  24,  43,  56,  32, 56,   18,  42,  3 },
{0,  26,  47,  62,  40,  5,   31,  56,  20},
{0,  28,  52,  17,  48, 15,   43,   9,  37},
{0,  30,  56,  23,  56, 26,   56,  24,  54},
{0,  32,  60,  30,  4,  36,    7,  39,  9 }, --> angle = 105 deg
}
```

Figure 3 –Example psCalLUT matrix (calibration setup for beam-steering between +/- 15 degrees at 1 deg resolution)

Only the 9 azimuth TX antenna are used in this example setup. These map to the AWR #4, AWR #3 and AWR #2 devices as shown in the following figure (excerpted from the MMWCAS-RF-EVM User Guide). AWR #1 (master) TX antennas mapped to elevation and are not used in this example.

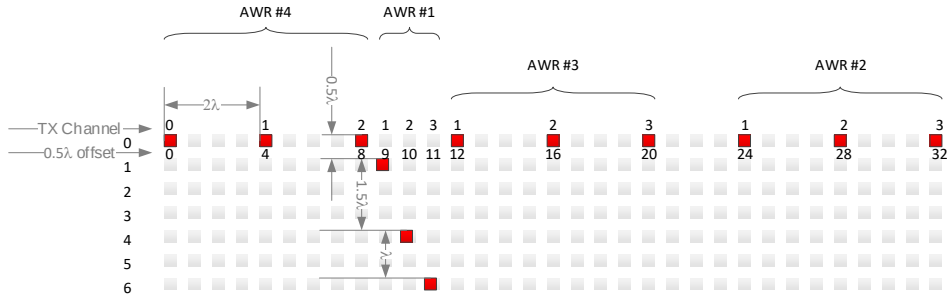


Figure 4 - MMWCAS-RF-EVM TX Antenna Mapping

The selected beam-steering angle (*psCalLUT* row index) is selected with the ***angleIdx*** variable.

Based on the *angleIdx* value, the *psCalLUT* values are mapped to the ***psSettings*** variable which is then used to program each individual device using the ar1.ProfileConfig_mult() lua API command.

After configuration the selected number of frames are captured and the data can be subsequently post-processed in the mmWave Studio post processing GUI, or any other external post processing environment.

2.2.1.3 Cascade_Configuration_TXBF_AngleSweep.lua

This is a mmWave Studio Lua automation script which sets up a **sweep** of static (single, fixed angle) TX beam-form example using basic chirp configuration. The ***angleIdx*** variable is swept through the ***psCalLUT*** matrix, with a dataset captured each angle offset.

This script is mostly identical to ***Cascade_Configuration_TXBF_Simple.lua*****,** with the addition of a sweep of the *psCalLUT* matrix. This script is meant to capture the full range of beam-steering offsets from the antenna system. When the data is post-processed, this can be used to create an RX SNR vs. beam angle graph.

After data collection is completed, the data can be post processed in the provided ***cascade_TXBF_Verification.m*** script.

2.2.2 Matlab Post-Processing Scripts

The following Matlab post-processing scripts are summarized here.

- cascade_TX_Phase_Calibration.m
  - cascade_Read_TX_Cal_Data.m
- TXBF_PS_LUT_Generate.m
  - TXBF_Calc_Phase_Settings.m
  - WrapTo360.m
- cascade_TXBF_Verification.m
- TXBF_Create_PSCal_Advanced_Frame_Config.m

2.2.2.1 cascade_TX_Phase_Calibration.m

This is a Matlab post-processing script which reads the captured calibration dataset and generates a *calibrateTXPhaseResults.mat* file that can be used to generate a look-up table for phase-shifter offsets.

The *calibrateTXPhaseResults.mat* file includes the following variables:

- *phaseValues*: Transmitters/System x Receivers/System x Phase-Shifter Offsets (3-dimensional) matrix which includes the per transmitter, receiver, phase-shifter setting, target measurement phase values used to compute the *phaseOffsetValues* variable

- *phaseOffsetValues*: Transmitters/System x Receivers/System x Phase-Shifter Offsets (3-dimensional) matrix. This matrix stores the per transmitter, receiver, phase-shifter setting, phase-offsets referenced to the zero-phase shifter setting offset of a selected TX channel.

The script works by iterating through the TX phase shifter calibration dataset and finding the peak of the 2D-FFT (0-velocity bin). This peak corresponds to the fixed calibration target IF return. The phase data of this 2D-FFT bin is recorded. By comparing each of these target phase values to the target phase seen on a reference TX/RX channel (virtual channel), the phase offsets to each TX can be computed. Phase shifter offsets are found from the calibration target phase data using the following equation:

$$\begin{aligned} phaseOffsetValues(TX\_X, RX\_Y, Phase\ Shifter\ Setting\ Z) \\ = phaseValues(TX\_X, RX\_Y, Phase\ Shifter\ Setting\ Z) \\ - \ phaseValues(TX\_ref, RX\_Y, Phase\ Shifter\ Setting\ 0) \end{aligned}$$

Multiple variables should be modified to match the calibration data environment before the script is executed. These include the following variables:

```
dataFolder_calib_data_path =
'C:\Radar_Data\20200721_phase_shifter_cal_testing\20200722_testdata_SN5733600
018_outdoor_cal2\'; % folder holding all of the calibration datasets
```

```
DEBUG_PLOTS = 1;     % optionally display debug plots while calibration data
is being processed
numDevices = 4;      % number of AWRx devices being processed, set to 4, for
the full MMWCAS-RF-EVM device array
numTX = 3;           % number of AWRx TX channels being processed, set to 3,
for the full AWRx device channels
numRX = 16;          % number of MMWCAS-RF-EVM RX channels being processed,
set to 16, for the full MMWCAS-RF-EVM RX channels
numPhaseShifterOffsets = 64; % number of phase-shifter offset increments
being processed, set to 64 for full phase-shifter range

% select reference TX/Device channel for computing offsets
refDevice = 4;
refTX = 1;
```

Figure 5 - cascade_TX_Phase_Calibration.m script variable options

The below graphs are example outputs (enabled with the *DEBUG_PLOTS*) option.

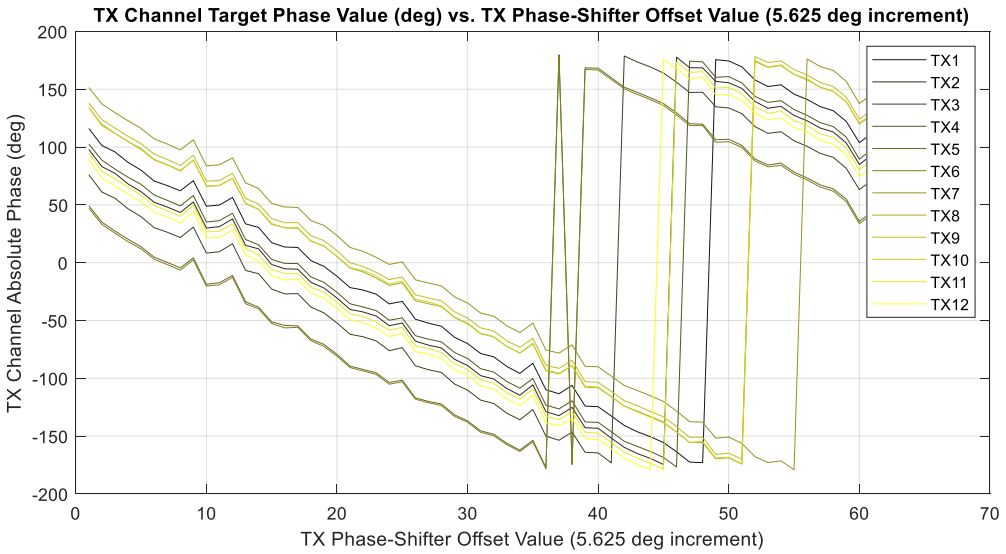


Figure 6 – Example Raw Target Phase Values

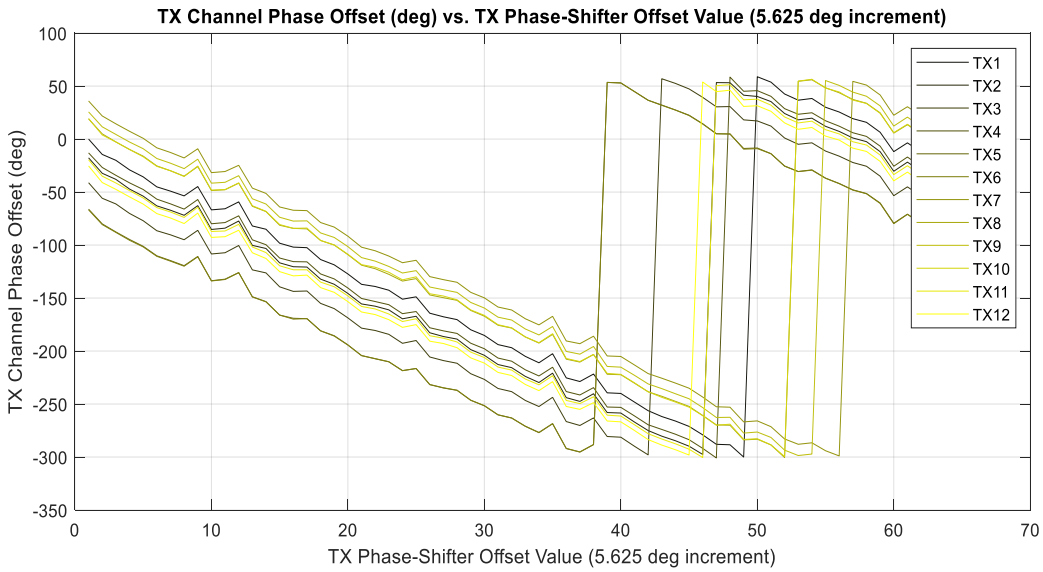


Figure 7 - Example Phase Offset Values (values, relative to reference channel)

2.2.2.2 *cascade_Read_TX_Cal_Data.m*

This is a dependency for *cascade_TX_Phase_Calibration.m* which reads a specific frame of the calibration data.

2.2.2.3 *TXBF_PS_LUT_Generate.m*

This script reads in the *calibrateTXPhaseResults.mat* generated by the *cascade_TX_Phase_Calibration.m* script and generates the TX beam forming calibrated look up table (LUT). This LUT transforms a select beam-steering angle to a set of calibrated phase-shifter programmable values.

This script takes in the matrix ***phaseOffsetValues*** and extracts out a single RX dataset of phase offset values. These values are then processed in the ***TXBF_Calc_Phase_Settings()*** function to calculate a specific beam-tilt angle to calibrated phase shifter programming LUT. This LUT is exported as a CSV file. Both a calibrated and ideal LUT are generated to compare against. The uncalibrated version consists of the ideal phase shifter values to program if there were no additional phase offset errors present in the system. The following CSV are generated.

psSettingProgram.csv – This is the calculated ideal phase shifter programming matrix which maps azimuth beam-steering angle to phase shifter program value.

psSettingProgramCal.csv – This script is the calibrated phase shifter programming matrix which maps azimuth beam-steering angle to phase shifter program value, which takes into effect the system phase offset errors.

The following option variables should be setup appropriately before running this script.

```
%% setup linear array variables
lambda = 3.900424 * 10 ^ -3;     % lambda in meters
d = 2 * lambda;                  % array separation in lambda
numAnts = 9;                     % number of antenna elements in the azimuth
array
deltaPhi = 360/64;               % ideal AWRx phase shifter discrete value in
degrees

deltaTheta = 1.0;                % beam-steering angle offset
%theta = 80:deltaTheta:100;      % beam-steering angle array -- defined
theta = -15:deltaTheta:15;       % beam-steering angle array
numPsOffsets = 64;


%phase Shifter calibration file
phaseShiftCalFile = '<example path>\calibrateTXPhaseResults.mat';
load(phaseShiftCalFile);
```

Figure 8 – TXBF_PS_LUT_Generate.m setup option

The phase offsets are unwrapped into a 0 to 360 degree space so that they can be compared to the phase-shifter programmable settings.

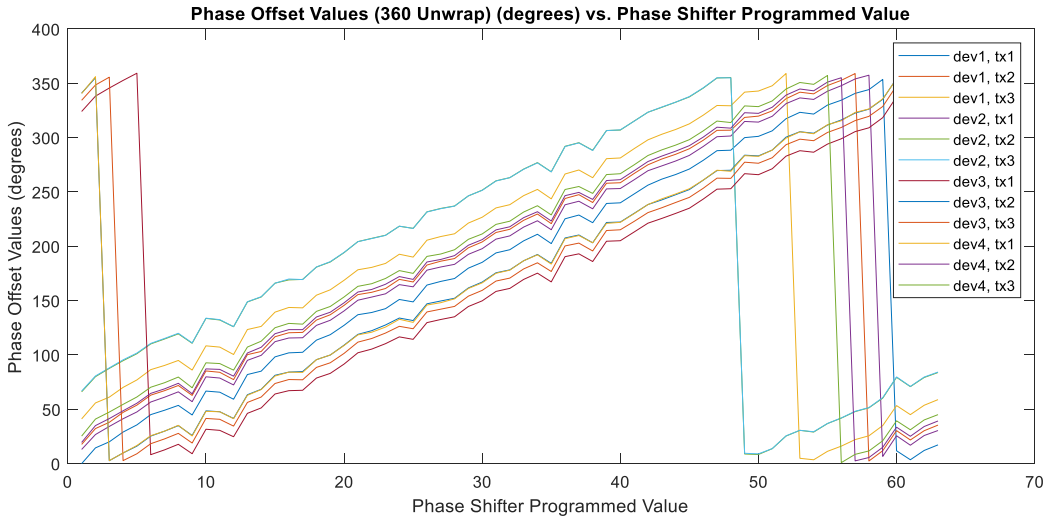


Figure 9 - Example phase shifter offset unwrapping

An ideal phase shifter offset matrix is constructed based on the array geometry. This contains ideal phase shifter values for each antenna channel. Then, the ideal and unwrapped calibration tables are compared by searching for the closest match to the ideal phase offset value for each TX channel and each beam-steering angle. The result is a set of calibrated, programmable offsets as shown in the figure below.

```
psCalLUT = {
{0,  30,  56,  22,  56, 25,   55,  23,  54}, --> angle = 75 deg
{0,  32,  60,  30,  4,  36,    6,  38,  8 },
{0,  34,  63,  32,  12, 46,   19,  53,  25},
{0,  36,   6,  33,  20, 56,   31,   6,  42},
{0,  39,  10,  39,  28,  6,   44,  20,  59},
{0,  41,  14,  45,  37, 16,   57,  35,  14},
{0,  43,  19,  51,  45, 27,    8,  50,  31},
{0,  45,  23,  6,   53, 37,   21,   3,  48},
{0,  47,  27,  12,  62, 48,   33,  18,  3 },
{0,  49,  32,  18,  10, 60,   46,  33,  21},
{0,  52,  36,  24,  18,  8,   59,  48,  38},
{0,  54,  40,  31,  26, 19,   10,   1,  55},
{0,  56,  45,  32,  35, 29,   23,  17,  10},
{0,  58,  49,  35,  44, 40,   36,  32,  27},
{0,  60,  53,  41,  52, 51,   49,  47,  45},
{0,   0,   0,  0,   0,   0,    0,   0,  0 }, --> angle = 90 deg
{0,   2,  62,  54,  9,  11,   14,  15,  18},
{0,   4,  4,   60,  17, 21,   26,  30,  35},
{0,   7,  8,   14,  25, 32,   39,  45,  52},
{0,   9,  12,  21,  34, 43,   52,  61,  8 },
{0,  11,  17,  27,  42, 54,    3,  14,  25},
{0,  13,  21,  32,  50,  3,   16,  29,  42},
{0,  15,  26,  33,  59, 14,   29,  44,  59},
{0,  17,  30,  38,  7,  24,   42,  59,  14},
{0,  19,  34,  44,  15, 35,   54,  12,  31},
{0,  21,  39,  50,  23, 45,    5,  27,  49},
{0,  24,  43,  56,  32, 56,   18,  42,  3 },
{0,  26,  47,  62,  40,  5,   31,  56,  20},
{0,  28,  52,  17,  48, 15,   43,   9,  37},
{0,  30,  56,  23,  56, 26,   56,  24,  54},
{0,  32,  60,  30,  4,  36,    7,  39,  9 }, --> angle = 105 deg
}
```

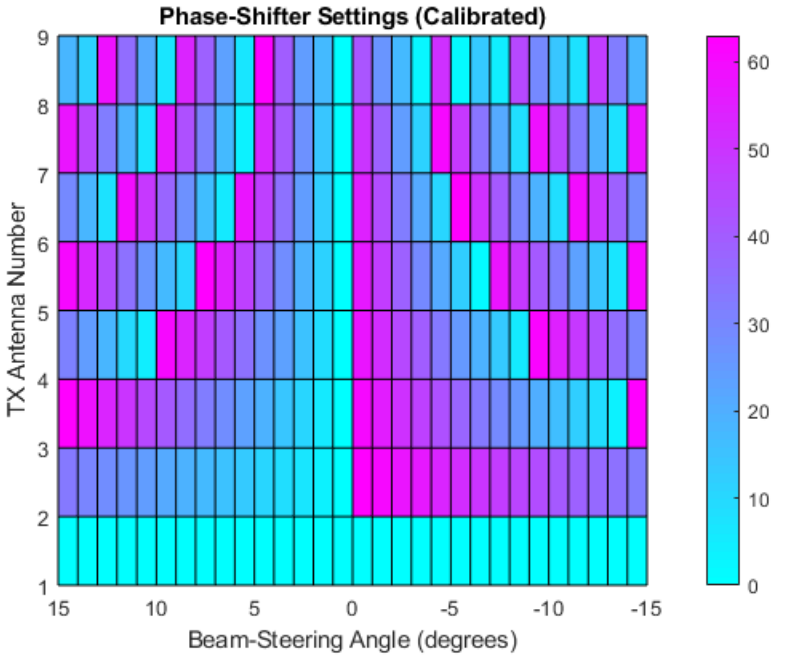


Figure 10 - Example phase shifter lookup table matrix (values and heat-map shown).

The beam-steering coordinate system has been chosen such that 0-degrees points in the boresight direction with +/- extents to either side as seen in the figure below.

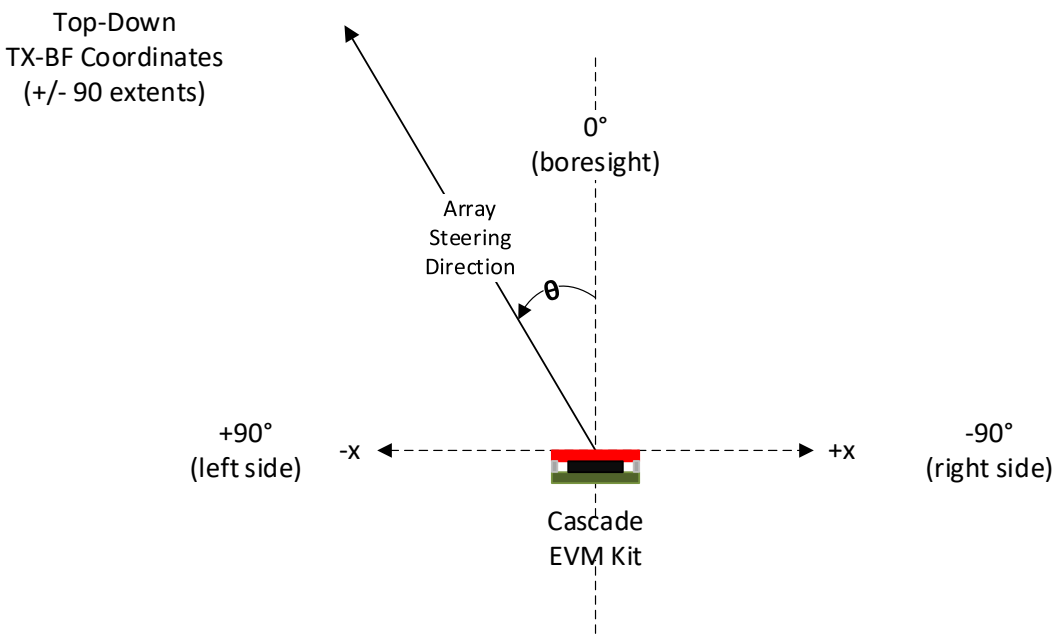


Figure 11 – Cascade EVM Beam-Steering Angle Coordinate System (top-down view of Cascade EVM kit)

This orientation maps to a AWRx device and TX azimuth antenna array coordinate system as seen below. AWRx device 4, TX1, is the phase reference channel used in this example. All other TX channel phase shifter offsets are chosen relative to this channel.

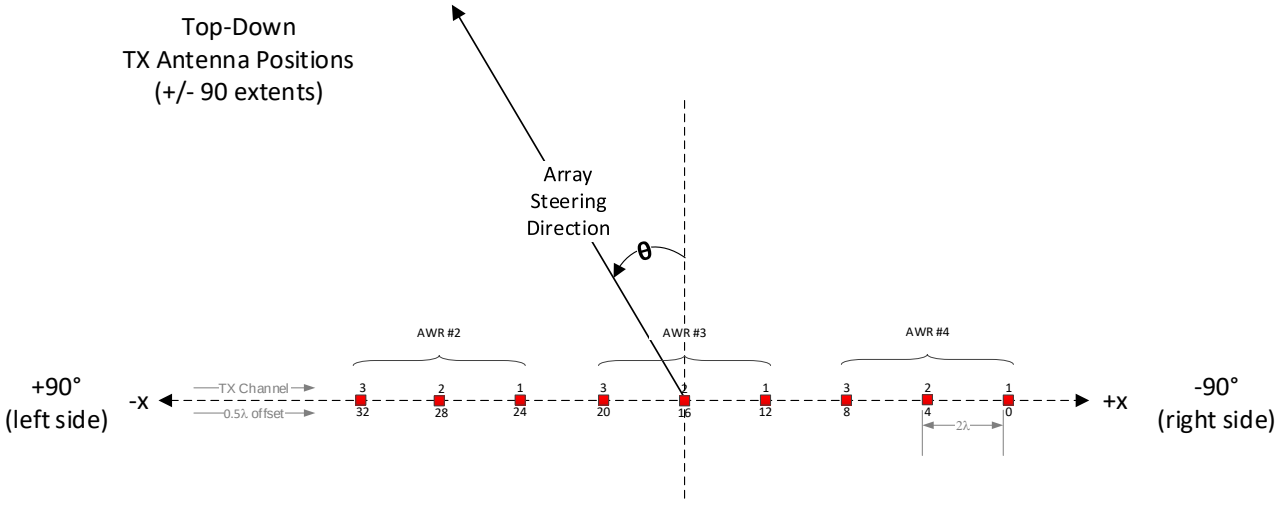


Figure 12 - Cascade EVM Beam-Steering Angle Coordinate System and Antenna and AWRx Device Positions

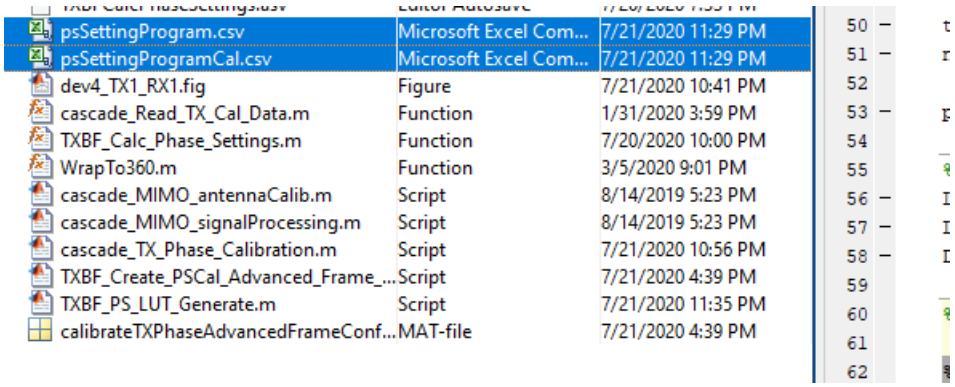


Figure 13 – Example CSV calibration matrix output

2.2.2.4 TXBF_Calc_Phase_Settings.m

This script is a dependency for *TXBF_PS_LUT_Generate.m* which generates the calibrated phase offsets for a single beam-steering angle.

2.2.2.5 cascade_TXBF_Verification.m

This script reads in a dataset gathered by the Cascade_Configuration_TXBF_AngleSweep.lua script. The goal is to find the peak magnitude vs. beam-steering angle. If the same boresight reflector scene used for the initial calibration dataset is used for this dataset, then this angle sweep will result in a measured target magnitude vs. angle graph, which should follow the phased array gain. See the below **section 5** comparing a calibrated vs. uncalibrated TX beamforming dataset.

2.2.2.6 TXBF_Create_PSCal_Advanced_Frame_Config.m

This script reads in the *calibrateTXPhaseResults.mat* file and generates a new phase shifter calibration matrix ***Ph*** which is in a compatible with the existing Advanced Frame configuration based TX Beam-Steering examples found in *\mmWaveStudio\MatlabExamples\4chip_cascade_TxBF_example*. This script saves the **Ph** matrix in the ***phaseShifterCalibration.mat***.

Users can then replace the existing *\mmWaveStudio\MatlabExamples\4chip_cascade_TxBF_example\Calibrations\phaseShifterCalibration.mat* with this new *phaseShifterCalibration.mat*

2.3 Software Script Installation

The supplied Lua and Matlab scripts are designed to operate within the mmWave Studio and Matlab post-processing example library environments released with the mmWave Studio installer.

If these scripts do not already exist in your installation of the 4chip_cascade_MIMO_example, copy the following files to the below locations:

\mmWaveStudio*MatlabExamples**4chip_cascade_MIMO_example**main**cascade*\

- cascade_TX_Phase_Calibration.m
- cascade_Read_TX_Cal_Data.m
- TXBF_PS_LUT_Generate.m
- TXBF_Calc_Phase_Settings.m
- WrapTo360.m

\mmWaveStudio\MatlabExamples\4chip_cascade_MIMO_example\main\cascade\paramGen

- parameter_file_gen_TXPS_VerificationCalib_json.m

The Matlab scripts described here must be executed from *MatlabExamples**4chip_cascade_MIMO_example**main**cascade*\ directory as they are dependent on the MIMO example source code for reading the calibration data binary files and JSON formatted description files. The example Matlab script environment setup covered in the mmWave Studio Cascade User Guide should be followed to enable these scripts to run as well.

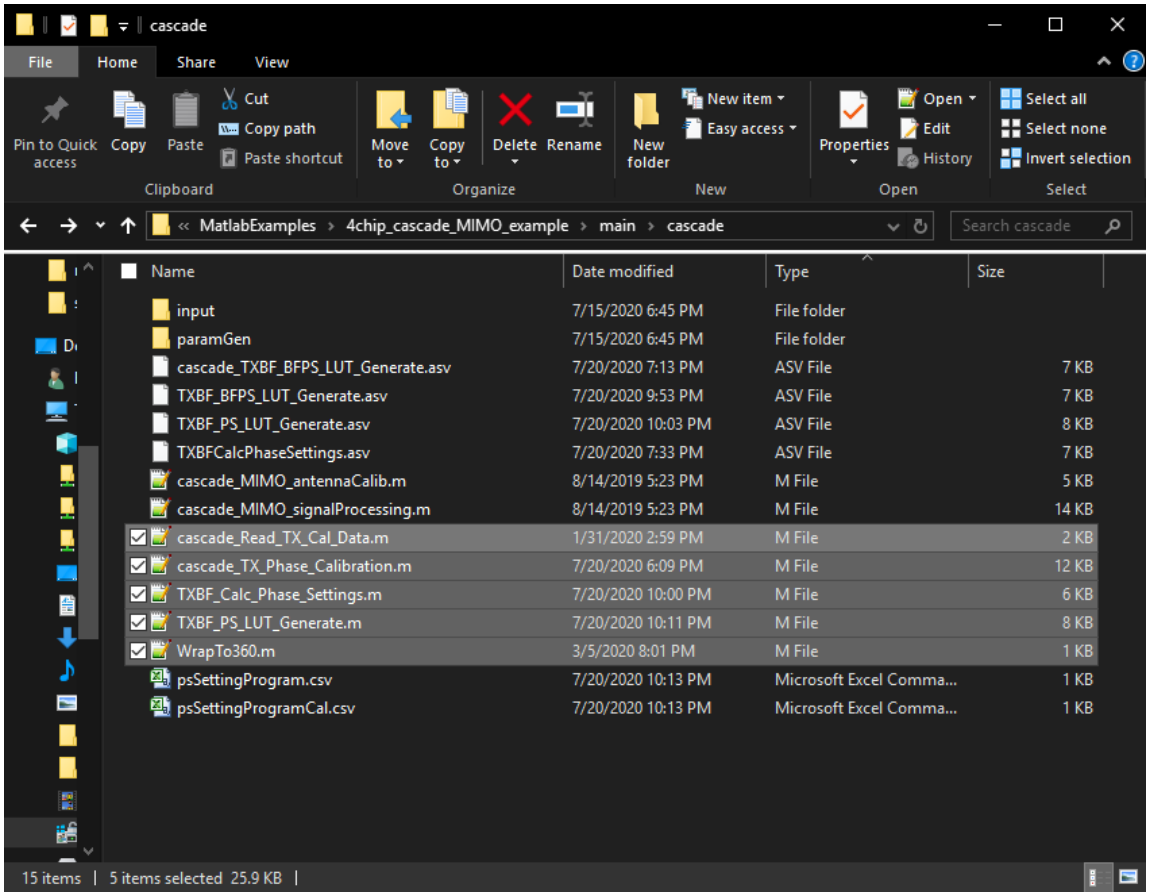


Figure 14 - Matlab script directory with TX phase shifter calibration files highlighted

The associated Lua scripts can be stored anywhere convenient. The script will be loaded in mmWave Studio to run as needed.

3 Calibration Procedure

The following sections describe the physical target range setup, data collection and post processing steps to obtain TX phase shifter calibration, apply the calibration, and verify the calibrated TXBF configuration.

3.1 Calibration Physical Setup

This example TX phase shifter calibration and verification is performed with a single corner reflector target placed at boresight relative to the MMWCAS-RF-EVM antenna array. TI recommends at least 5 meters distance to cascade radar EVM such that target is sufficiently in far-field of radiation pattern and a plane wave reflection is presented to the RX antenna array. The range should be relatively clear of clutter (radially) for at least the distance to the corner reflector so no other targets are accidently used for calibration.

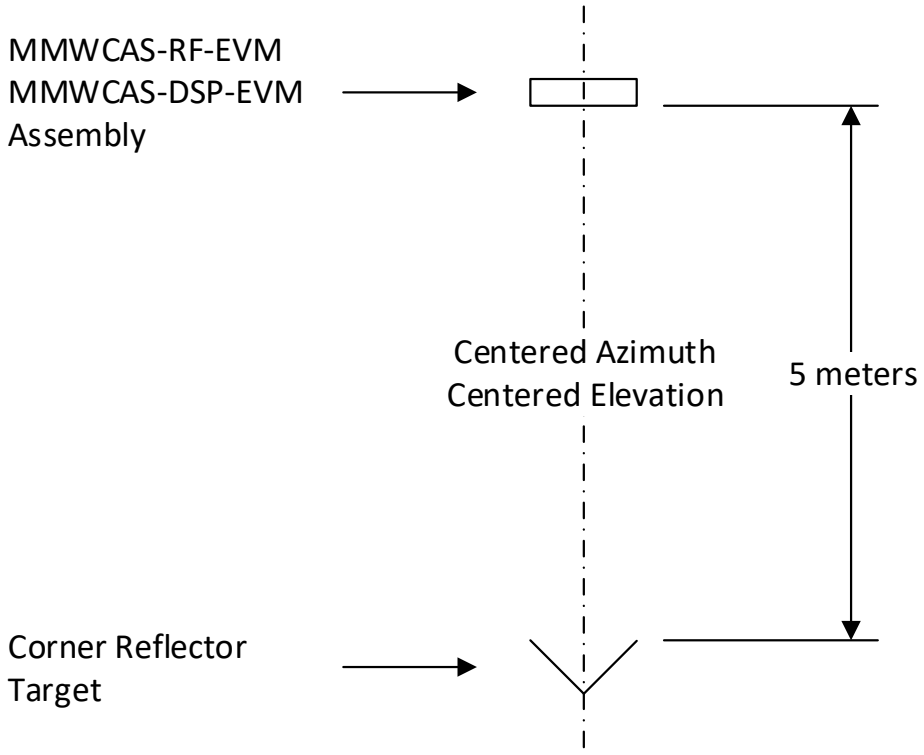


Figure 15 - Calibration Target and EVM Relative Positioning

3.2 Calibration Script Process

The following sections describe the calibration script execution flow. The below diagram outlines this flow, starting with running the Cascade_Phase_Shifter_Calibration_AWRx.lua script in mmWave Studio, and ending with the collection of TX phase shifter calibrated TX Beam-formed data. The below diagram shows how the different datasets are generated and used for the different calibration processing scripts.

3.2.1 Calibration Data Collection

Calibration Data Collection steps:

- Setup and un the Cascade_Phase_Shifter_Calibration_AWRx.lua script in mmWave Studio
    - User should modify script for required chirp configuration to match the desired chirp operating conditions that the calibration is intended for
    - User should modify script to select specific chirp profile, number of devices/TX-channels/TX phase-shifter offset values desired in sweep
- Script will run through a sweep of devices/TX-channels and phase-shifter values
- A full, 4-device, 12-TX, 64 phase-shifter offset run may take 30 minutes to iterate through. The time required is mostly limited by data transfer from SSD over Ethernet.

Once the calibration data is collected, the user should have a directory with multiple data captures present. An example directory listing is shown below:

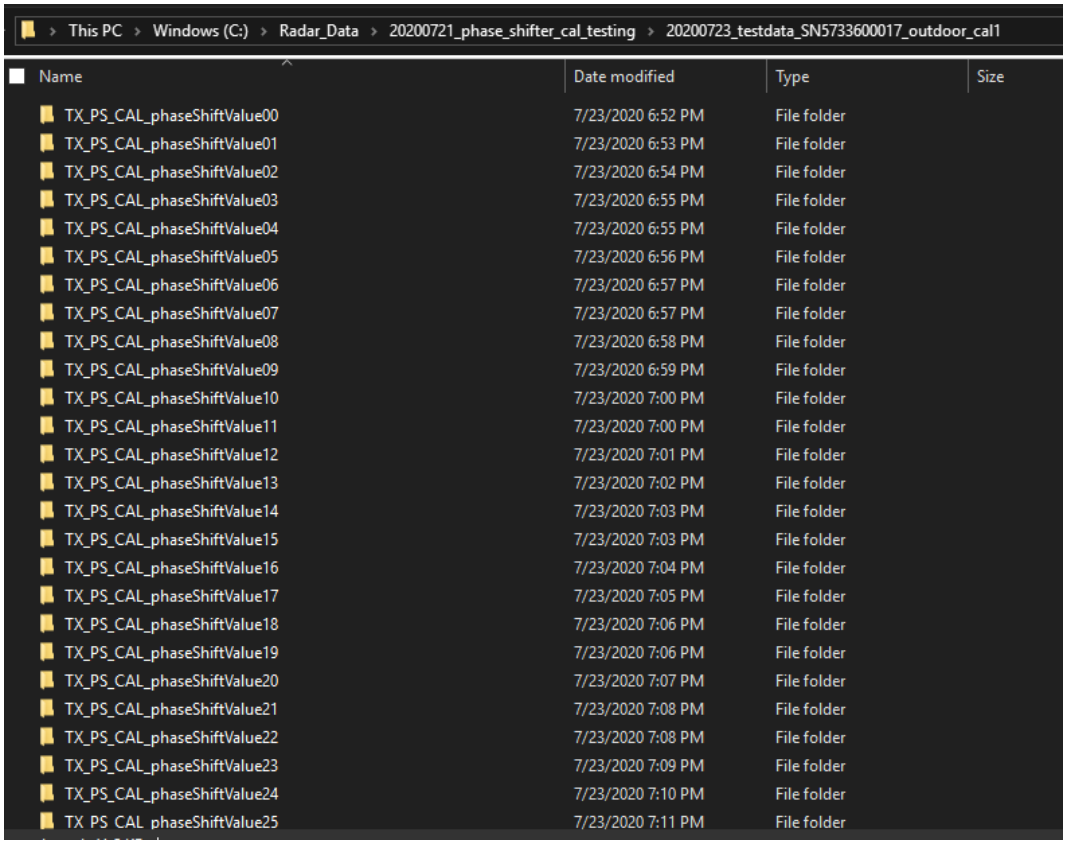


Figure 16 - Example calibration data folder listing

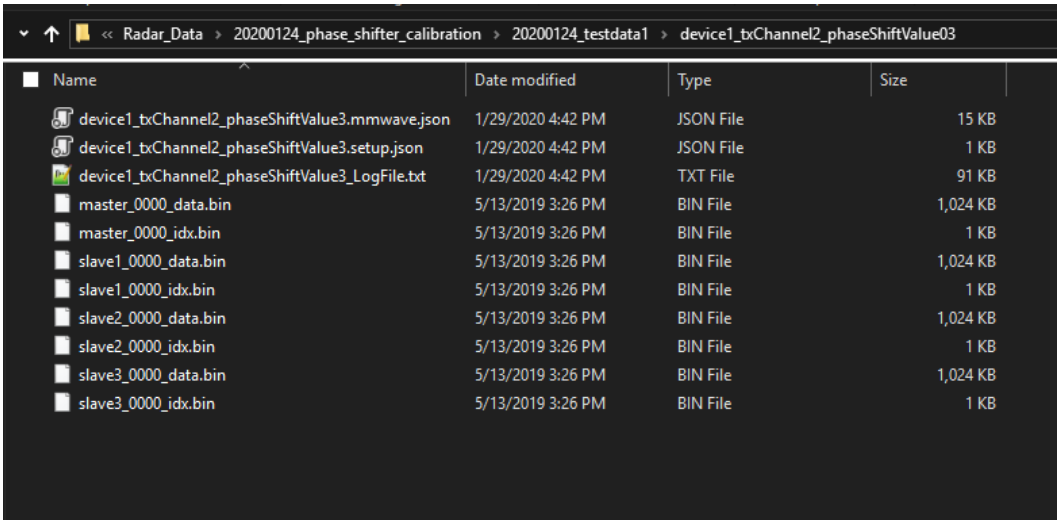


Figure 17 – Example calibration data folder listing, for a single device, TX channel and phase shifter offset iteration

3.2.2 Calibration Data Processing

First, setup the Matlab environment for the *cascade_TxBF_Calibration.m* script. Just as with the Matlab MIMO example, the user needs to run the *add_paths.m* script run prior to execution to set environment path variables for dependencies. Full description of the Matlab post-processing environment setup can be found in the mmWave Studio install directory under *\docs\mmwave_studio_cascade_user_guide.pdf*

The user needs to modify the user control variables before starting the script. See above script description in **section 2.2.2.1**.

Once script execution is completed, the raw phase offsets for each TX/RX channel can be found in the ***phaseValues*** variable. This is the input into the next script set which will create the beam angle to phase shifter setting lookup table. This matrix along with other resulting data are all saved off to a .mat file in the calibration data directory under as ***calibrateTXPhaseResultsFileFull.mat***

At the end of execution the calibration target phase and phase offset matrix are exported to a matlab .mat file for future import.

```
231      % save calibration data to .mat file
232 -    calibrateTXPhaseResultsFile = [dataFolder_calib_data_path '\calibrateTXPhaseResults.mat'];
233 -    save(calibrateTXPhaseResultsFile, 'phaseOffsetValues', 'phaseValues', 'phaseError');
```

3.2.3 Beam Angle to Phase Shifter Lookup Table Generation

The user should then run the *TXBF_PS_LUT_Generate.m* script to generate both a calibrated and uncalibrated beam steering angle to phase shifter programming lookup table.

This script assumes the above shown **Figure 11** and **Figure 12** azimuth linear array coordinate systems when computing the phase angles. Antenna 1 is the AWR #4, TX1 starting on the -90 deg (right) side of the beam angle.

Open the ***TXBF_PS_LUT_Generate.m*** and edit the following variables to match the calibration dataset and setup the desired lookup table parameters. Modify the ***phaseShiftCalFile*** variable to point to the calibration dataset generated *calibrateTXPhaseResults.mat*.

Once the variables for creating the lookup table and the input calibration data have been set, run the script. This script references the 4D matrix variable ***phaseOffsetValues*** from the calibration *calibrateTXPhaseResults.mat* file to create a beam steering angle to programmable phase shifter offset value.

The script will output a few debug graphs, and two CSV (comma separated variable) files will be written to disk in the current working directory as shown below. Each CSV file will contain a matrix of beam angle offsets and phase shifter programming values. These can be applied to the example

Cascade_Configuration_TXBF_Simple.lua, *Cascade_Configuration_TXBF_AngleSweep.lua* TXBF examples as well as the Matlab advanced frame configuration TXBF example.

3.2.4 Validating Phase Shifter Calibration Results

As explained in the above description of the *Cascade_Configuration_TXBF_Simple.lua***,** the resulting CSV files can be imported into the Lua example code, allowing users to setup a calibrated, static azimuth beam-steering example.

The values are copied into the TX Beam-Forming example .lua script and formatted into a Lua 2 dimensional list. As described above, this list *psCalLUT* is then used to generate appropriate TX phase offset values for TX beam-formed use-cases. See Figure 3 and Figure 4 above and associated text.

3.2.5 Updating Calibration in Matlab TXBF Example

Run the TXBF_Create_PSCal_Advanced_Frame_Config.m script with the *calibrateTXPhaseResults.mat* input path set appropriately. The resulting *phaseShifterCalibration.mat can then be applied to the Advanced Frame Configuration TXBF example.*

4 Example Calibrated vs. Uncalibrated TX-BF Datasets

To validate the resulting TX phase shifter lookup table a data capture against a static boresight target can be done for each of the beam steering angles. The above described *Cascade_Configuration_TXBF_AngleSweep.lua* script and *cascade_TXBF_Verification.m* file capture and process data in this manner.

The received power from this boresight target vs. the beam steering angle is plotted in the *cascade_TXBF_Verification.m* to reveal the resulting array gain pattern. One example dataset is shown below. In this case the calibrated vs. uncalibrated dataset is shown together.

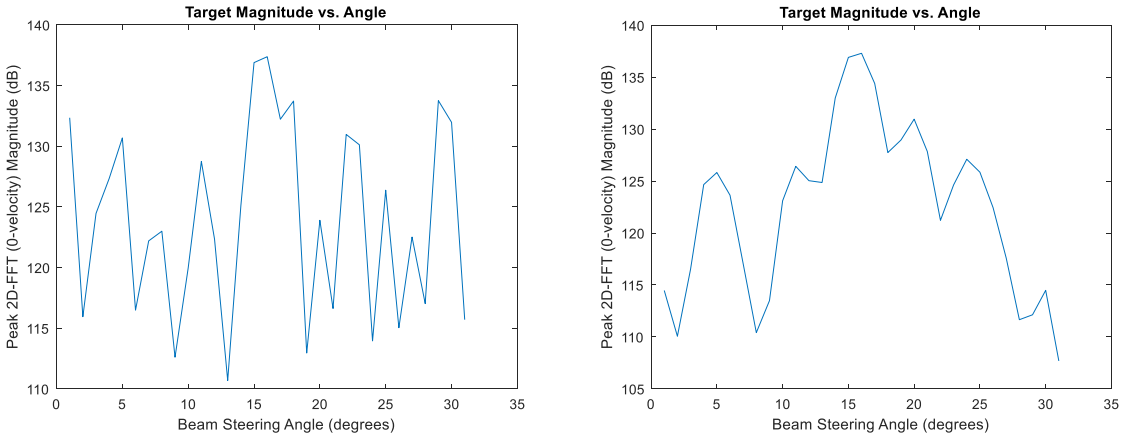


Figure 18 - Example Uncalibrated (left) and Calibrated (right) Beam Angle vs. Target Received Power

5 Revision History

- 2020/02/19 – rosales.r@ti.com
    - Second revision, fixed some typos.

- 2020/03/09 – rosales.r@ti.com
    - Added description of the TX phase calibration matrix generation matlab scripts
    - Showed resulting calibrated vs. uncalibrated settings matrix
    - Added an example calibrated vs. uncalibrated peak detection vs. beam-steering angle graphs generated with the MMWCAS-RF-EVM and a boresight corner reflector target

- 2020/07/21 – rosales.r@ti.com
    - Additional updates for clarity prior to release of examples in mmWave Studio package

- 2020/08/11 – rosales.r@ti.com
    - Updated figure 21 – title was incorrect
    - Updating documentation to deal only with the script remove redundant explainations.
    - Added Section 2.2.2.6 describing the TXBF_Create_PSCal_Advanced_Frame_Config.m