AUTO-DRIVING / TECHNICAL REPORTS / SENSORS

TR_Tin_TrafficMonitoringRadar_20211031

T Created by tin.giang.s2.tt
Oct 31, 2021 • 3 min read • 2 people viewed

Traffic Monitoring for 18xx or 68xx

Name	Date modified	Type	Size
.metadata	10/19/2021 3:39 AM	File folder	
area_scanner	10/19/2021 3:39 AM	File folder	
automated_doors_and_gates	10/19/2021 3:39 AM	File folder	
common	10/19/2021 3:39 AM	File folder	
gesture_recognition	10/19/2021 3:39 AM	File folder	
level_sensing	10/19/2021 3:39 AM	File folder	
long_range_people_detection	10/19/2021 3:39 AM	File folder	
out_of_box_demo	10/19/2021 3:39 AM	File folder	
parking_garage_sensor	10/19/2021 3:39 AM	File folder	
people_counting	10/19/2021 3:39 AM	File folder	
robotics	10/19/2021 3:39 AM	File folder	
traffic_monitoring	10/19/2021 3:39 AM	File folder	
vital_signs	10/19/2021 3:39 AM	File folder	

1. Requirements

Tool	Version	Download Link
TI mmWave SDK		MMWAVE-SDK 03_05_00_04 - TI.com
mmWave Industrial Toolbox		Industrial Toolbox (ti.com)
MATLAB Runtime	2019a (9.6)	MATLAB Runtime - MATLAB Compiler - MATLAB (mathworks.com)
Uniflash		UniFlash (ti.com)

2. Quickstart

a. Setup the EVM for Flashing Mode

1. Ensure that the SOP Switches (SW2) are set to Flash Programming Mode

2. Connect AoPCB Board to Host PC via USB cable.

3. Open Uniflash tool, select IWR1642 and click 'Start'.

4. Under Settings and Utilities tab, choose CFG_port as the COM port.

AoPCB Port	Host PC COM Ports	Port Function
CFG_port	Port: CP2105-Enhanced COM Port, Baud Rate:115200	AoPCB Programming & Configuration
DATA_port	Port: CP2105-Standard COM Port, Baud Rate:921600	AoPCB MSS Logger output

Table 2 : UART Port mapping

5. Under 'Program' tab, browse to the binary file to be programmed for 'Meta Image 1'. In the release zip package provided by Mistral, the binary file is 'MS_60GhzAoPCB_mmw_demo.bin'.

6. Click 'Load Image'.

7. Check logs in Console window to verify flashing status. After successful flashing, disconnect the board from PC.

8. Power OFF the Board.

9. Set SOP Switches back to Functional Mode for normal Radar Function

b. Flash the EVM using Uniflash

c. Setup the EVM for Functional Mode

d. Run the Lab

3. Data Formats

A TLV(type-length-value) encoding scheme is used with little endian byte order. For every frame, a packet is sent consisting of a fixed sized Frame Header and then a variable number of TLVs depending on what was detected in that scene. The TLVs can be of types representing the point cloud, target list object, and associated points.

Frame Header

Size: 52 bytes

```
frameHeaderStructType = struct(...
    'magicWord',                {'uint64', 8}, ... % syncPattern in hex is: '02 01 04 03 06 05 08 07'
    'version',                  {'uint32', 4}, ... % Software Version
    'platform',                 {'uint32', 4}, ... % A6843
    'timeStamp',                {'uint32', 4}, ... % Message create time in cycles
    'totalPacketLen',           {'uint32', 4}, ... % In bytes, including header
    'frameNumber',              {'uint32', 4}, ... % Frame Number
    'subFrameNumber',           {'uint32', 4}, ... % Sub-Frame number
    'chirpProcessingMargin',    {'uint32', 4}, ... % time left after chirp processing in cycles
    'frameProcessingMargin',    {'uint32', 4}, ... % time left after frame processing in cycles
    'trackingProcessingTime',   {'uint32', 4}, ... % time to run tracker
    'uartSendingTime',          {'uint32', 4}, ... % time to send uart message
    'numTLVs' ,                 {'uint16', 2}, ... % Number of TLVs in this frame
    'checksum',                 {'uint16', 2}); % Subframe number.
```
Frame Header Structure in MATLAB syntax for name, type, length

TLV Header

Size: 8 bytes

```
% TLV Type: 06 = DPIF Point cloud spherical, 07 = Target object list, 08 = Target index, 09 = DPIF Point Cloud Side In
fo
tlvHeaderStruct = struct(...
    'type',           {'uint32', 4}, ... % TLV object
    'length',         {'uint32', 4});    % TLV object Length, in bytes, including TLV header
```

TLV header in MATLAB syntax

Point Cloud TLV

Type: DPIF_POINT_CLOUD_SPHERICAL

Size: sizeof (DPIF_PointCloudSpherical) x numberOfPoints

Each Point Cloud TLV consists of an array of points. Each point is defined in 16 bytes.

```
DPIF_PointCloudSpherical = struct(...
    'range',           {'float', 4}, ... % Range, in m
    'azimuth',         {'float', 4}, ... % Azimuth angle, in rad
    'elevation',       {'float', 4}, ... % Elevation angle, in rad
    'doppler'          {'float', 4}, ... % Doppler, in m/s
```

Point Structure in MATLAB syntax

Point Cloud Side Info TLV

Type: DPIF_POINT_CLOUD_SIDE_INFO

Size: sizeof(DPIF_PointCloudSideInfo) x numberOfPoints

Each Point Cloud Side Info TLV consists of an array of point side info data. Each is 8 bytes.

```
DPIF_PointCloudSideInfo = struct(...
    'snr',             {'int16_t', 2}, ... % SNR, ratio
    'noise'            {'int16_t', 2}, ... % Noise
```

Point Side Info Structure in MATLAB syntax

Target Index TLV

Type: TARGET_INDEX

Size: numberOfPoints

Each Target List TLV consists of an array of target IDs. A targetID at index i is the target to which point i of the frame's point cloud was associated. Valid IDs range from 0-249.

```
targetIndex = struct(...
    'targetID',        {'uint8', 1});    % Track ID
```

Target ID Structure in MATLAB syntax

Other Target ID values:

Value	Meaning
253	Point not associated, SNR too weak
254	Point not associated, located outside boundary of interest
255	Point not associated, considered as noise

Exam Frame: 16xx - People Counting Demo

Example Parsing

Example UART stream with annotation of Frame Header and TLVs.

Frame Header
Point Cloud TLV
Target List TLV
Target Index TLV
Type Length Header

👍 Like   Be the first to like this   No labels 🏷