TI Designs: TIDEP-01000

People Tracking and Counting Reference Design Using mmWave Radar Sensor

Description

The TIDEP-01000 reference design demonstrates the use of the IWR6843 ISK, a single-chip, mmWave radar sensor from TI with integrated DSP, for an indoor and outdoor people-counting application. This reference design uses the IWR6843 ISK + ICB (Industrial Carrier Board) and integrates a complete radar-processing chain onto the IWR6843 device. This solution is shown to localize people out to 6 m, with an location accuracy greater than 90% and a counting density of 3 people per square meter.

Resources

TIDEP-01000	Design Folder
IWR6843	Product Folder
IWR6843ISK + ICB Bundle	Tool Folder for ISK/ICB Bundle

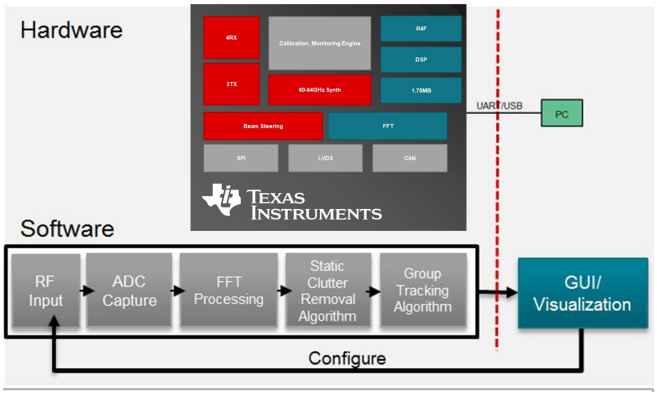
TI E2E™ Community — ASK Our E2E™ Experts

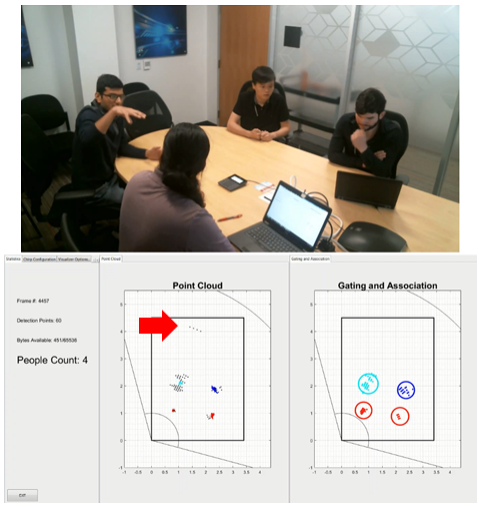
Features

- Demonstration Hardware and Software Using IWR6843, mmWave Radar Sensor for People Counting
- mmWave Technology Provides Range, Velocity, and Angle Information – Ideal for Environmental Effects
- All Detection and Signal Processing Handled Onboard IWR6843, Point Cloud and Object Information Streamed to PC for Visualization
- Azimuth Field of View of 120 Degrees Across Distance of 6 m, Which Can Extend to 14 m With Different Chirp Configuration
- Examples of Implementation of Static Clutter and Group Tracking Algorithms

Applications

- People Counting
- Motion Detector
- Automated Doors and Gates
- IP Network Camera
- Lighting Sensors
- Safety Guards





An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

NOTE: This design previously used the IWR1642. The source code for the IWR1642 People Counting Demo is available on TIREX.

1 System Description

The TIDEP-01000 provides a reference for building a people-counting application that can detect and track humans up to 6 meters away. This design is a custom data processing demo application built to work on the IWR6843, an integrated single-chip frequency modulated continuous wave (FMCW) radar sensor capable of operation in the 60 to 64 GHz band.

Human monitoring has become an important area of exploration, due to its potential for understanding people's count, activities, intents, and health issues. The ability to continuously and consistently monitor human motion is an important function in numerous applications, including surveillance, control, and analysis. Accuracy and precision plays an important role in these applications. While sensors such as passive infra-red (PIR) and time of flight (TOF) are in use, they suffer from limitations in accuracy, false alarms, and environmental changes such as darkness, brightness, and smoke.

Radars allow an accurate measurement of distances, relative velocities of people, and other objects. They are relatively immune to environmental conditions such as the effects of rain, dust, or smoke. Additionally, they can work in complete darkness or in bright daylight. They are therefore useful for building automation applications such as people counting, motion detection, IP network cameras, and safety guards.

The design provided is an introductory-demo application developed on the IWR6843 ISK to accurately count and track the number of people in the area of interest. The radar data processing chain implements the static clutter removal, range azimuth heat map, and Doppler extraction algorithms for obtaining point cloud information. This information is passed on to the group tracker for object localization. After viewing this design, the customer should be able to better understand the implementation of advanced algorithms on the IWR6843 device.

Table 1. Key Performance Specifications

PERFORMANCE PARAMETER	SHORT RANGE CONFIGURATION	LONG RANGE CONFIGURATION
Device	IWR6843BOOST	IWR6843BOOST
Field of view	120° horizontal, 30° vertical	120° horizontal, 30° vertical
Maximum range	6.3 meters	14 meters
Range resolution	5.5 cm	12 cm
Maximum velocity	23.6 Km/h	23.6 Km/h
Velocity resolution	0.36 Km/h	0.36 Km/h
Algorithms used	Static clutter removal, CFAR, group tracking	Static clutter removal, CFAR, group tracking
System power consumption	2 watts	2 watts

2 System Overview

2.1 Block Diagram

2.1.1 Hardware Block Diagram

The TIDEP-01000 is implemented on the IWR6843 ISK EVM(see Figure 1). The EVM is connected to a host PC through a universal asynchronous receiver-transmitter (UART) for visualization.

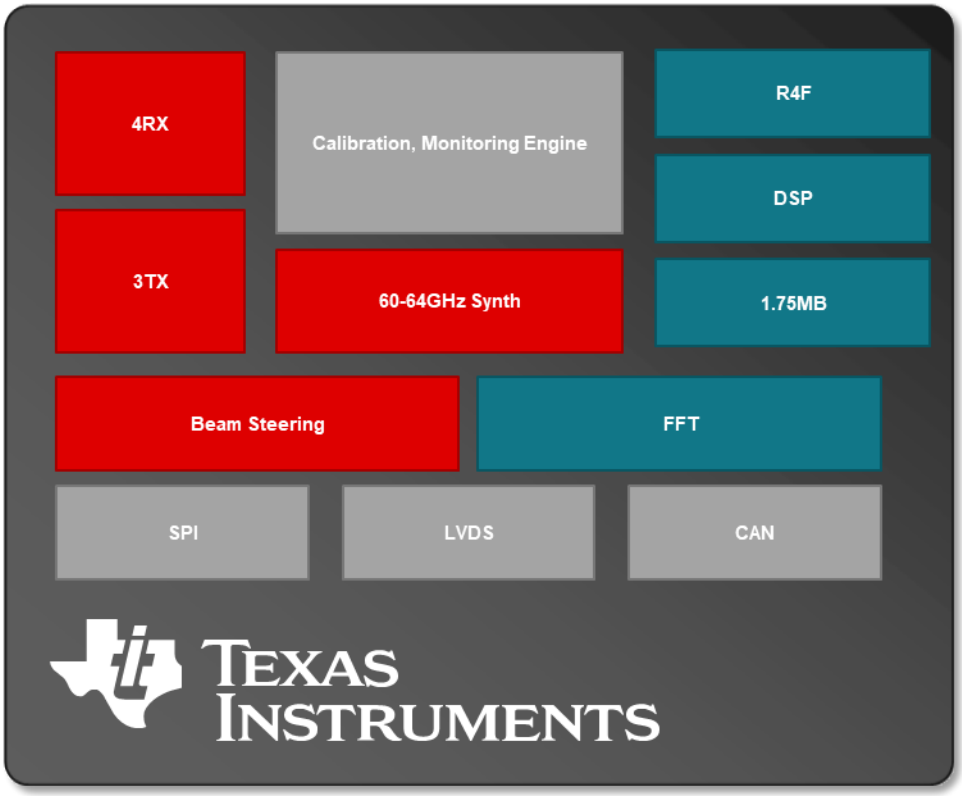


Figure 1. Hardware Block Diagram

2.1.2 Software Block Diagram

2.1.2.1 mmWave SDK Software Block Diagram

The mmWave software development kit (SDK) enables the development of mmWave sensor applications using the IWR6843 SOC and EVM (see Figure 2). The SDK provides foundational components that help end users focus on their applications. The SDK also provides several demonstration applications, which serve as a guide for integrating the SDK into end-user mmWave applications. This reference design is a separate package, installed on top of the SDK package.

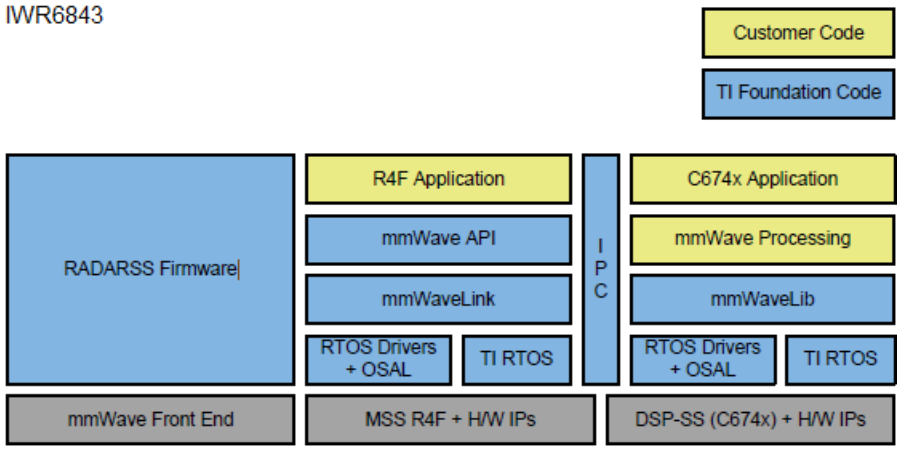


Figure 2. mmWave SDK Software Block Diagram

2.1.2.2 Software Block Diagram of People-Counting Application

As shown in Figure 3, the implementation of the people-counting application demo on the IWR6843 consists of a signal chain running on the C674x DSP, and the tracking module running on the ARM® Cortex®-R4F processor.

- Range processing:
  - For each antenna, 1D windowing, and 1D fast Fourier transform (FFT)
  - Range processing is interleaved with the active chirp time of the frame
- Capon beam forming:
  - Static clutter removal
  - Covariance matrix generation, inverse-angle spectrum generation, and integration is performed
  - Outputs range-angle heat map
- CFAR detection algorithm:
  - Two-pass, constant false-alarm rate
  - First pass cell averaging smallest of CFAR-CASO in the range domain, confirmed by second pass cell averaging smallest of CFAR-CASO in the angle domain, to find detection points.
- Doppler estimation:
  - For each detected [range, azimuth] pair from the detection module, estimate the Doppler by filtering the range bin using Capon beam-weights, and then run a peak search over the FFT of the filtered range bin.
- Tracking:
  - Perform target localization, and report the results.
  - Output of the tracker is a set of trackable objects with certain properties like position, velocity, physical dimensions, and point density

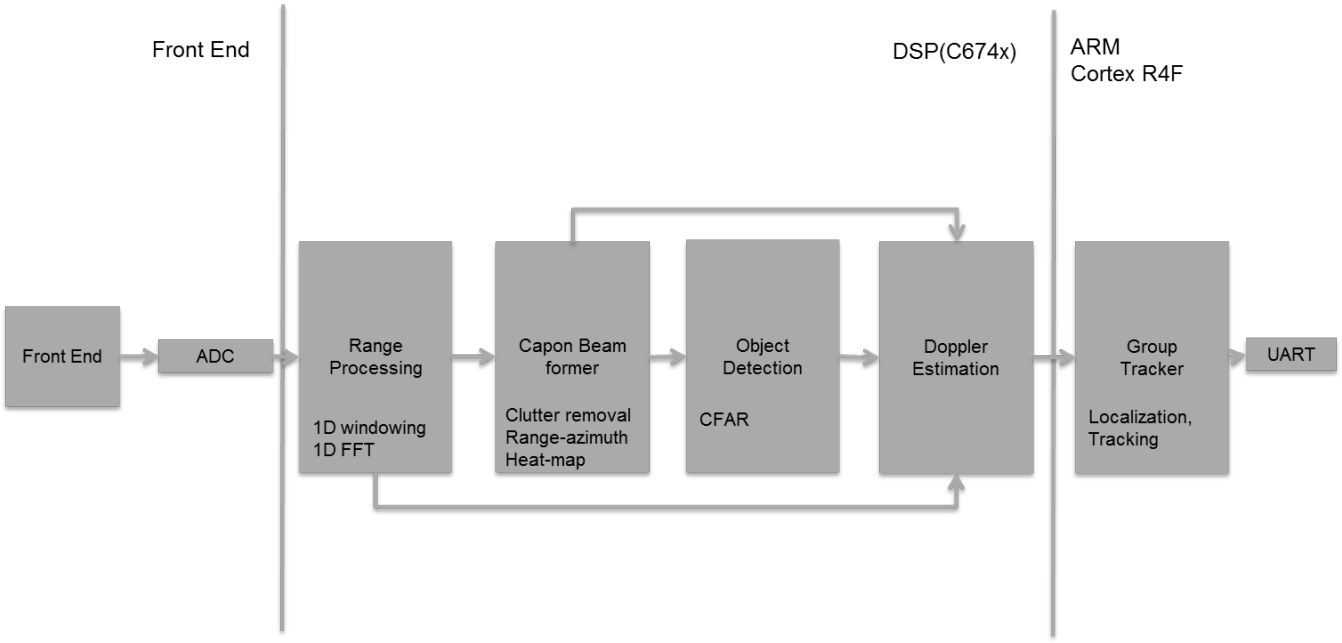


Figure 3. People-Counting Application Block Diagram

2.2 Highlighted Products

2.2.1 IWR6843

The IWR6843 is an integrated, single-chip, frequency modulated continuous wave (FMCW) sensor capable of operation in the 60- to 64-GHz band. The sensor is built with the low-power, 45-nm, RFCMOS process from TI and enables unprecedented levels of integration in an extremely small form factor. The IWR6843 is an ideal solution for low-power, self-monitored, ultra-accurate radar systems in the industrial space.

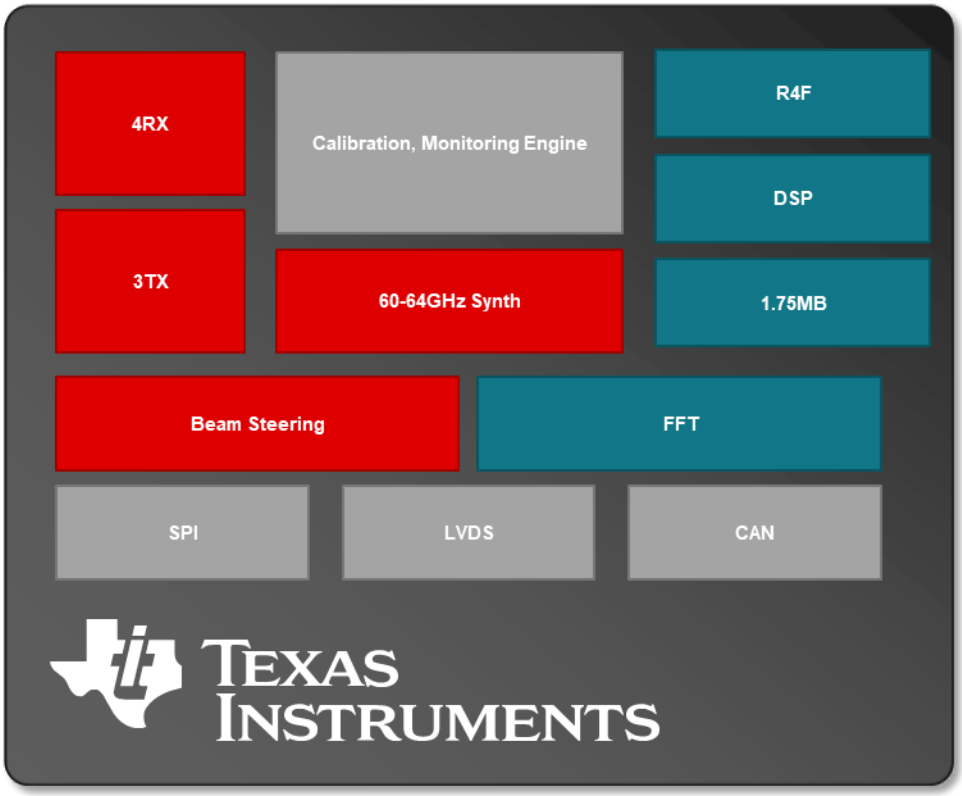


Figure 4. IWR6843 Block Diagram

The IWR6843 has the following features:

- FMCW transceiver
  - Integrated PLL, transmitter, receiver, baseband, and A2D
  - 60- to 64-GHz coverage, with 4-GHz available bandwidth
  - Four receive channels
  - Three transmit channels
  - Ultra-accurate chirp (timing) engine based on fractional-N PLL
  - TX power
    - 12 dBm
  - RX noise figure
    - 12 dB (60 to 64 GHz)
  - Phase noise at 1 MHz
    - –92 dBc/Hz (60 to 64 GHz)
- Built-in calibration and self-test (monitoring)
  - ARM Cortex-R4F-based radio control system

  - Built-in firmware (ROM)
  - Self-calibrating system across frequency and temperature
- C674x DSP for FMCW-signal processing
  - On-chip memory: 1.75MB
- Cortex-R4F MCU for object detection, and interface control
  - Supports autonomous mode (loading the user application from QSPI flash memory)

- Integrated peripherals
  - Internal memories with ECC
  - Up to six ADC Channels
  - Up to two SPI Channels
  - Up to two UARTs
  - CAN interface
  - I^2C
  - GPIOs
  - Two-lane LVDS interface for raw ADC data and debug instrumentation

2.2.2 mmWave SDK

The mmWave SDK is divided into two broad components: mmWave Suite and mmWave Demos.

2.2.2.1 mmWave Suite

The mmWave Suite is the foundational software part of the mmWave SDK and includes the following smaller components:

- Drivers
- OSAL
- mmWaveLink
- mmWaveLib
- mmWave API
- BSS firmware
- Board setup and flash utilities

For more information, see the mmWave SDK user's guide.

2.2.2.2 mmWave Demos

The SDK provides a suite of demonstrations that depict the various control and data-processing aspects of an mmWave application. Data visualization of the output of a demonstration on a PC is provided as part of these demonstrations.

- mmWave processing demonstration

2.3 System Design Theory

2.3.1 Use-Case Geometry and Sensor Considerations

The IWR6843 is a radar-based sensor that integrates a fast, FMCW, radar front end, with both an integrated Arm Cortex-R4F MCU and a TI C674x DSP for advanced signal processing. The configuration of the IWR6843 radar front end depends on the configuration of the transmit signal and the configuration and performance of the RF transceiver, the design of the antenna array, and the available memory and processing power. This configuration influences key performance parameters of the system.

The key performance parameters at issue follow with brief descriptions:

- Maximum range
  - Range is estimated from a beat frequency in the de-chirped signal that is proportional to the round trip delay to the target. For a given chirp ramp slope, the maximum theoretical range is determined by the maximum beat frequency that can be detected in the RF transceiver. The maximum practical range is then determined by the SNR of the received signal and the SNR threshold of the detector.
- Range resolution
  - This is defined as the minimum range difference over which the detector can distinguish two individual point targets, determined by the bandwidth of the chirp frequency sweep. The higher the chirp bandwidth, the finer the range resolution.
- Range accuracy
  - This is often defined as a rule of thumb formula for the variance of the range estimation of a single point target as a function of the SNR.
- Maximum velocity
  - Radial velocity is directly measured in the low-level processing chain as a phase shift of the de-chirped signal across chirps within one frame. The maximum unambiguous velocity observable is then determined by the chirp repetition time within one frame. Typically this velocity is adjusted to be one-half to one-fourth of the desired velocity range to have better tradeoffs relative to the other parameters. Other processing techniques are then used to remove ambiguity in the velocity measurements, which experience aliasing.
- Velocity resolution
  - This is defined as the minimum velocity difference over which the detector can distinguish two individual point targets that also happen to be at the same range. This is determined by the total chirping time within one frame. The longer the chirping time, the finer the velocity resolution.
- Velocity accuracy
  - This is often defined as a rule of thumb formula for the variance of the velocity estimation of a single-point target as a function of the SNR.
- Field of view
  - This is the sweep of angles over which the radar transceiver can effectively detect targets. This is a function of the combined antenna gain of the transmit and receive antenna arrays as a function of angle, and can also be affected by the type of transmit or receive processing, which may affect the effective antenna gain as a function of angle. The field of view is typically specified separately for the azimuth and elevation.
- Angular resolution
  - This is defined as the minimum angular difference over which the detector can distinguish two individual point targets that also happened to have the same range and velocity. This is determined by the number and geometry of the antennas in the transmit and receive antenna arrays. This is typically specified separately for the azimuth and elevation.
- Angular accuracy
  - This is often defined as a rule of thumb formula for the variance of the angle estimation of a single point target as a function of SNR.

When designing the frame and chirp configuration for a people-counting use case, start by considering the geometry of the scene specifically the range of interest. Table 2 lists example chirp configurations with good range and Doppler resolution and memory for a 6-m and 14-m indoor people-counting application.

Table 2. Performance Parameters of Two Example Chirp Designs on the IWR6843

KEY INPUT PARAMETERS		
PERFORMANCE PARAMETER	6-METER RANGE	14-METER RANGE
Starting frequency (GHz)	60.6	60.6
Maximum range, R_{max} (m)	6.3	14
Range resolution (cm)	5.5	12
Maximum velocity (Km/h)	23.6	23.6
Velocity resolution (Km/h)	0.36	0.36
Idle time (µs)	30	30
ADC valid start time (µs)	10	10
Periodicity (ms)	50	50
Valid sweep bandwidth (MHz)	2713.66	1228
Ramp slope (MHz/µs)	53	24
Sampling frequency (Msps)	2.50	2.50
Number of samples per chirp	128	128
Ramp end time (µs)	62	62.20
Chirp repetition period (µs)	184.0	184.0
Number of chirps	128	128
Active frame time (ms)	23.55	23.55
Radar cube size (KB)	512	512

2.3.2 Low-Level Processing

An example of a processing chain for people counting is implemented on the IWR6843 EVM.

The main processing elements involved in the processing chain consist of the following components:

- Front end – Represents the antennas and the analog RF transceiver implementing the FMCW transmitter and receiver and various hardware-based signal conditioning operations. This must be properly configured for the chirp and frame settings of the usage case.
- ADC – Main element that interfaces to the DSP chain. The ADC output samples are buffered in ADC output buffers for access by the digital part of the processing chain.
- EDMA controller – User-programed DMA engine employed to move data from one memory location to another without using another processor. The EDMA can be programed to trigger automatically and can also be configured to reorder some of the data during the movement operations.
- C674 DSP – Digital signal-processing core that implements the configuration of the front end and executes the main signal processing operations on the data. This core has access to several memory resources as noted further in the design description.
- Arm R4F – Arm MCU that can execute application code including further signal processing operations and other higher level functions. In this application, the Arm Cortex-R4F primarily implements group tracker and relays target-list data over the UART interface. There is a shared memory visible to both the DSP and the Cortex-R4F.

The processing chain is implemented on the DSP and Cortex-R4F together. Table 3 lists the several physical memory resources used in the processing chain.

Table 3. Memory Configuration

SECTION NAME	SIZE (KB) AS CONFIGURED	MEMORY USED (KB)	DESCRIPTION
L1D SRAM	16	16	Layer one data static RAM is the fastest data access for DSP and is used for most time-critical DSP processing data that can fit in this section.
L1D cache	16	16	Layer one data cache caches data accesses to any other section configured as cacheable. LL2, L3, and HSRAM are configured as cache-able.
L1P SRAM	28	24	Layer one program static RAM is the fastest program access RAM for DSP and is used for most time-critical DSP program that can fit in this section.
L1P cache	4	4	Layer one cache caches program accesses to any other section configured as cacheble. LL2, L3, and HSRAM are configured as cache-able.
L2	256	176	Local layer two memory is lower latency than layer three for accessing and is visible only from the DSP. This memory is used for most of the program and data for the signal processing chain.
L3	640	600	Higher latency memory for DSP accesses primarily stores the radar cube and the range-Doppler power map. It is a less time-sensitive program. Data can also be stored here.
HSRAM	32	Currently unused	Shared memory buffer between the DSP and the R4F relays visualization data to the R4F for output over the UART in this design.

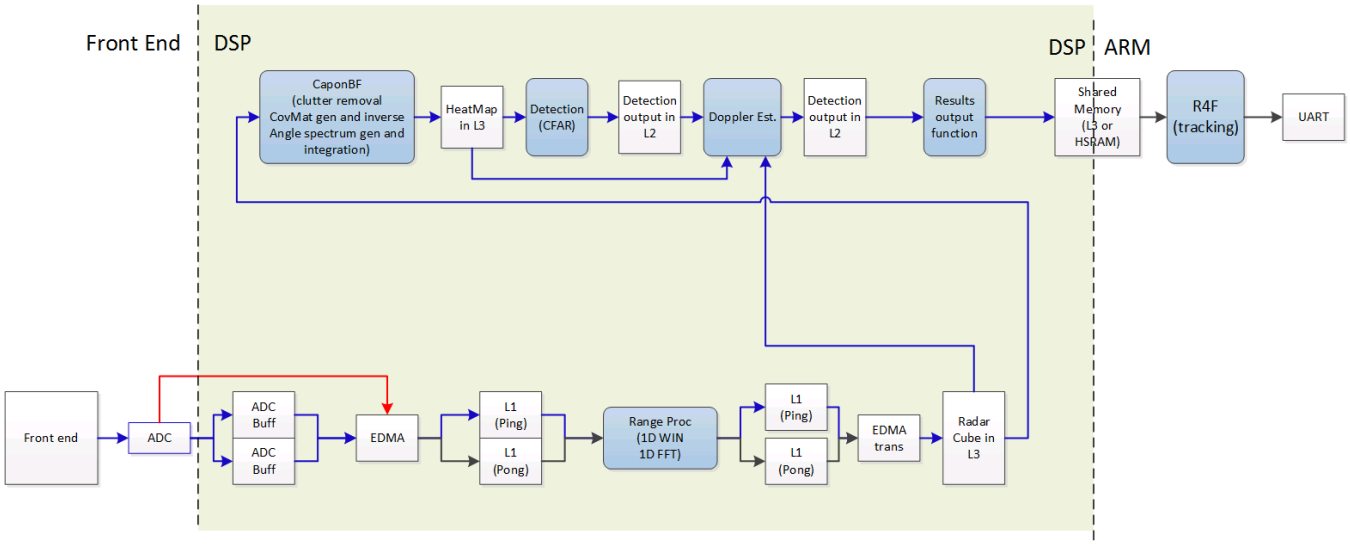


Figure 5. Processing Chain Flow: Detection-Tracking Visualization

As shown in Figure 5, the implementation of the people-counting example in the signal-processing chain consists of the following blocks implemented as DSP code executing on the C674x core in the IWR6843:

- Range processing
  - For each antenna, EDMA is used to move samples from the ADC output buffer to the local memory of the DSP. A 16-bit, fixed-point 1D windowing and 16-bit, fixed-point, 1D FFT are performed. EDMA is used to move output from the DSP local memory to the radar cube storage in layer three (L3) memory. Range processing is interleaved with active chirp time of the frame. All other processing occurs each frame, except where noted, during the idle time between the active chirp time and the end of the frame.
- Capon beam former
  - Let s(t) be the incoming waves after mixing to baseband. The sensor array signal to be processed is given by: $X(t) = A(\theta)s(t) + n(t)$. Where:
    - $A(\theta) = (a(\theta_1), \ldots a(\theta_M))$ is the steering matrix.
    - $a(\theta) = (e^{j2\pi y}1^{\sin(\theta)}, \ldots, e^{j2\pi y}N^{\sin(\theta)})$ is the steering vector.
    - M is the number of angle bins.
    - y_n is the sensor position normalized by wavelength.

    The Capon BF approach is: $\theta_{capon} = \text{argmin}_{\theta}\{\text{trace}(A(\theta) \times R_n^{-1} \times A(\theta)^H\}$, where R_n is the spatial covariance matrix.
  - Static clutter removal is implemented by removing DC components per range bin. This removes the *static* object reflections like a chair or table in the area of interest. Then per range bin, spatial covariance matrix R_n is computed using multiple chirps within a frame. Then R_n is inverted and the upper diagonal of the R_n^{-1} is stored in memory for each range bin. Per range bin, capon beam former output is calculated and stores the angle spectrum in memory to construct the range-azimuth heat-map.
- Object detection
  - Two pass CFAR algorithms is used on the range azimuth heat map to perform the object detection. First pass is done per angle bin along the range domain. Second pass in the angle domain is used confirm the detection from the first pass. The output detected point list is stored in L2 memory.
- Doppler estimation
  - For each detected point in range and azimuth(angle) space, Doppler is estimated using the capon beam weights and Doppler FFT. The output is stored in the L2 memory.

  All the above processing except the range processing happens during inter-frame time. After DSP finishes frame processing, the results are written in shared memory (L3/HSRAM) for Cortex-R4F to input for the group tracker.
- Group tracker
  - The tracking algorithm implements the localization processing. Tracker works on the point cloud data from DSP, and provide localization information which can be used by classification layers (currently not implemented in this example of a people-counting application). Tracker inputs the point cloud data, performs target localization, and reports the results (a target list). Therefore, the output of the tracker is a set of trackable objects with certain properties (like position, velocity, physical dimensions, point density, and other features).

The output from the tracker is formatted and sent to the host using UART for visualization. The visualization update rate is slower than the actual processing rate due to the limited bandwidth of the UART interface.

Table 4 lists the results of benchmark data measuring the overall MIPS and memory consumption of the processing chain, up to and including the tracking.

Table 4. MIPS Use Summary

PARAMETER	AVAILABLE TIME	USED TIME	LOADING
Active chirp time	92 µs	14 µs	15%
Frame time	26.5 ms	7.2 ms	27%

2.3.3 High-Level Processing Details

2.3.3.1 Task Model

High-level processing is implemented with two tasks: higher priority mailbox task, and lower priority application task. When the system is configured, the mailbox task is pending on a semaphore, waiting for the frame ready message from DSP. When awakened, the mailbox task copies the relevant point cloud data from the shared memory into TCM, and posts the semaphore to an application task to run. It then creates the transport frame header, and initiates a DMA process for each part (TLV) of the frame. While DMA started sending data over UART, the mailbox task yields to the lower priority application task. When the DMA process completes, additional DMA can be scheduled (for example, TM2 and TM3). To achieve parallelism between the task processing and DMA, the transmit task sends the current (Nth) point cloud TLV with the previous (N-1)th target list and target index TLVs.

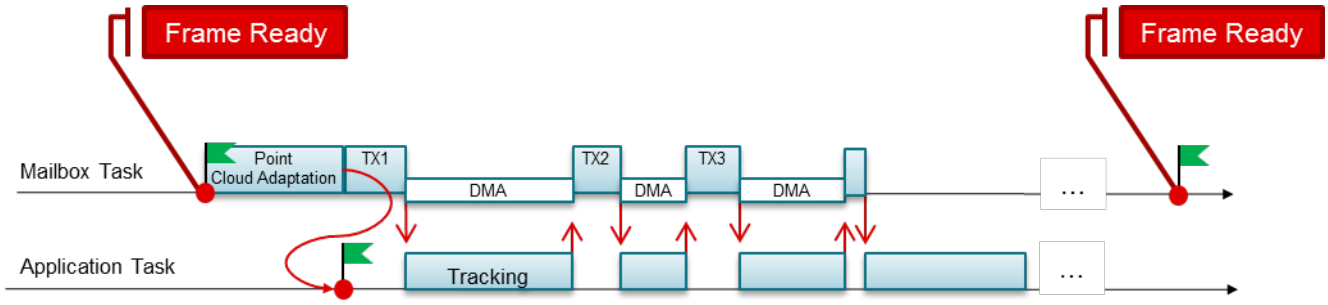


Figure 6. High-Level Processing Task Model

2.3.3.2 Group Tracker

The tracking algorithm is implemented as a library. The application task creates an algorithm instance with configuration parameters that describe sensor, scenery, and behavior of radar targets. The algorithm is called once per frame from the application task context. It is possible to create multiple instances of group tracker. Figure 7 shows the steps algorithm goes during each frame call. The algorithm inputs measurement data in polar coordinates (range, angle, Doppler), and tracks objects in Cartesian space. Therefore, use the extended Kalman filter (EKF) process.

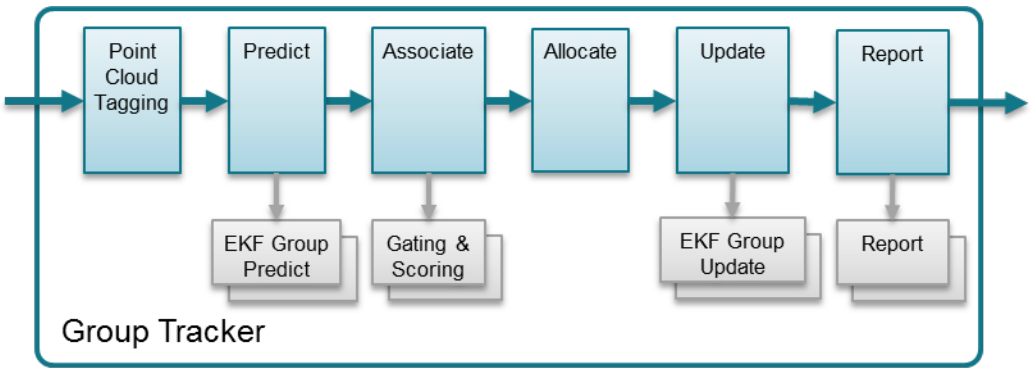


Figure 7. Group Tracking Algorithm

Point cloud input is first tagged based on scene boundaries. Some points may be tagged as *outside the boundaries*, and are ignored in association and allocation processes.

The predict function estimates the tracking group centroid for time n based on state and process covariance matrices, estimated at time n-1. Compute a-priori state and error covariance estimations for each trackable object. At this step, compute measurement vector estimations.

The association function allows each tracking unit to indicate whether each measurement point is *close enough* (gating), and if it is, to provide the bidding value (scoring). The point is assigned to a highest bidder. Points not assigned go through an allocate function. During the allocation process, points are first joined into a sets based on their proximity in measurement coordinates. Each set becomes a candidate for an allocation decision, and must pass multiple tests to become a new track. When passed, the new tracking unit is allocated. During the update step, tracks are updated based on the set of associated points. Compute the innovation, Kalman gain, and a-posteriori state vector and error covariance. In addition to classic EKF, the error covariance calculation includes group dispersion in a measurement noise covariance matrix.

The report function queries each tracking unit and produces the algorithm output.

2.3.3.3 Configuration Parameters

The configuration parameters are used to configure the tracking algorithm. They are adjusted to match the customer use case, based on particular scenery and target characteristics. Parameters are divided into mandatory, and optional (advanced). Mandatory parameters are described in Table 5.

Table 5. Mandatory Configuration Parameters

PARAMETER	DEFAULT	DIM	DESCRIPTION
maxNumPoints	250	—	Maximum number of detection points per frame
maxNumTracks	20	—	Maximum number of targets to track at any given time
stateTrackingVectorType	2DA	—	2DA = {x, y, vx, vy, ax, ay}. This is the only supported option
initialRadialVelocity	0	m/s	Expected target radial velocity at the moment of detection
maxRadialVelocity	N/A	m/s	Maximum absolute radial velocity reported by sensor
radialVelocityResolution	N/A	m/s	Minimal non-zero radial velocity reported by the sensor
maxAcceleration	2	m/s^2	Maximum target acceleration. Used to compute processing noise matrix
deltaT	50	ms	Frame rate
verbosityLevel	NONE	—	A bit mask representing levels of verbosity: NONE | WARNING | DEBUG | ASSOCIATION DEBUG | GATE_DEBUG | MATRIX DEBUG

2.3.3.3.1 Advanced Parameters

Advanced parameters are divided into a few sets. Each set can be omitted, and defaults are used by an algorithm. The customer must modify the necessary parameters to achieve better performance.

2.3.3.3.1.1 Scenery Parameters

This set of parameters describes the scene. It allows the user to configure the tracker with expected boundaries and scene entrances. These effect tracker behavior. The tracker does not track clusters outside of the boundaries set by the rightWall and leftWall parameters, and tracker behavior can be tuned differently for the areas defined by the lowerEntrance and upperEntrance parameters. Table 6 lists these parameters.

Table 6. Scenery Parameters

PARAMETER	DEFAULT	DIM	DESCRIPTION
lefWall	–1.5	m	Position of the left wall, in meters, set to -100 if no wall. Points behind the wall will be ignored
rightWall	1.5	m	Position of the right wall, in meters, set to 100 if no wall. Points behind the wall will be ignored
lowerEntrance	1	m	Entrance area lower boundary, in meters; set to 0 if not defined.
upperEntrance	4.5	m	Entrance area lower boundary, in meters; set to 100 if not defined.

2.3.3.3.1.2 Measurement Standard Deviation Parameters

This set of parameters is used to estimate standard deviation of the reflection point measurements. Table 7 lists these parameters.

Table 7. Measurements Standard Deviation Parameters

PARAMETER	DEFAULT	DIM	DESCRIPTION
LengthStd	1/3.46	m	Expected standard deviation of measurements in target length dimension
WidthStd	1/3.46	m	Expected standard deviation of measurements in target width dimension
DopplerStd	1.0f	m/s	Expected standard deviation of measurements of target radial velocity

Typically, the uniform distribution of reflection points across target dimensions can be assumed. In such cases, standard deviation can be computed as: $\sigma = \frac{b-a}{\sqrt{12}}$.

For example, for the targets that are 1 m wide, standard deviation can be configured as: $\frac{1}{\sqrt{12}}$.

2.3.3.3.1.3 Allocation Parameters

The reflection points reported in the point cloud are associated with existing tracking instances. Points that are not associated are subjects for the allocation decision. Each candidate point is clustered into an allocation set. To join the set, each point must be within maxDistance and maxVelThre from the set's centroid. When the set is formed, it must have more than setPointsThre members, and pass the minimal velocity and SNR thresholds. Table 8 lists these parameters.

Table 8. Allocation Parameters

PARAMETER	DEFAULT	DIM	DESCRIPTION
SNR threshold	100	—	Minimum total SNR for the allocation set, linear sum of power ratios
Velocity threshold	0.1	m/s	Minimum radial velocity of the allocation set centroid
Points threshold	5	—	Minimum number of points in the allocation set
maxDistanceThre	1	m^2	Maximum squared distance between candidate and centroid to be part of the allocation set
maxVelThre	2	m/s	Maximum velocity difference between candidate and centroid to be part of the allocation set

2.3.3.3.1.4 State Transition Parameters

Each tracking instance can be in either FREE, DETECT, or ACTIVE state. Once per frame, the instance can get HIT (have non-zero points associated to a target instance) or MISS (no points associated) event.

When in FREE state, the transition to DETECT state is made by the allocation decision. See Section 2.3.3.3.1.3 for the allocation decision configuration parameters. When in DETECT state, use the det2active threshold for the number of consecutive hits to transition to ACTIVE state, or det2free threshold of number of consecutive misses to transition back to FREE state. When in ACTIVE state, the handling of the MISS (no points associated) is as follows:

- If the target is in the static zone *and* the target motion model is close to static, then assume that the reason for no detection is because they were removed as static clutter. In this case, increment the miss count, and use the static2free threshold to extend the life expectation of the static targets.
- If the target is outside the static zone, then no points were associated because that target is exiting. In this case, use the exit2free threshold to quickly free the exiting targets.
- If the target is in the static zone, but has non-zero motion in radial projection, then the lack of detections occurs when the target is obscured by other targets. In this case, continue target motion according to the model, and use the active2free threshold.

Table 9 lists the parameters used to set this behavior.

Table 9. State Transition Parameters

PARAMETER	DEFAULT	DIM	DESCRIPTION
det2activeThre	10	—	In DETECT state; how many consecutive HIT events needed to transition to ACTIVE state
det2freeThre	5	—	In DETECT state; how many consecutive MISS events needed to transition to FREE state
active2freeThre	10	—	In ACTIVE state and NORMAL condition; how many consecutive MISS events needed to transition to FREE state
static2freeThre	100	—	In ACTIVE state and STATIC condition; how many consecutive MISS events needed to transition to FREE state
exit2freeThre	5	—	In ACTIVE state and EXIT condition; how many consecutive MISS events needed to transition to FREE state

2.3.3.3.1.5 Gating Parameters

The gating parameters set is used in the association process to provide a boundary for the points that can be associated with a given track. These parameters are target-specific. Table 10 lists each of these parameters.

Table 10. Gating Function Parameters

PARAMETER	DEFAULT	DIM	DESCRIPTION
Volume	4	—	Gating volume
LengthLimit	3	m	Gating limit in length
WidthLimit	2	m	Gating limit in width
VelocityLimit	0	m/s	Gating limit in velocity (0 – no limit)

The gating volume can be estimated as the volume of the ellipsoid, computed as $V = \frac{4\pi}{3}abc$, where a, b, and c are the expected target dimensions in range (m), angle (rad), and doppler (m/s).

For example, consider a person as a radar target. For the target center, we could want to reach ±0.45 m in range (a = 0.9), ±3 degree in azimuth (b = 6π / 180), and ±5.18 m/s in radial velocity (c = 10.36), resulting in a volume of approximately 4.

In addition to setting the volume of the gating ellipsoid, the limits can be imposed to protect the ellipsoid from overstretching. The limits are the function of the geometry and motion of the expected targets. For example, setting the WidthLimit to 8 m does not allow the gating function to stretch beyond 8 m in width.

2.3.3.4 Memory Use

The Cortex-R4F uses tightly-coupled memories (256KB of TCMA and 192KB of TCMB). TCMA is used for program and constants (PROG), while TCMB is used for RW data (DATA). Memory use at the Cortex-R4F is summarized in the following tables. Table 11 lists the total memory footprint, indicating memory use.

Table 11. Cortex-R4F Memory Use

MEMORY	AVAILABLE (BYTES)	USED (BYTES)	USE (PERCENTAGE)
PROGRAM	261888	103170	39%
DATA	196608	171370	87%

Table 12 lists the memory used by the tracking algorithm in percentages to a total memory footprint. The tracking algorithm is instantiated with 250 maximum measurements in point-cloud input, and a maximum of 20 tracks to maintain at any given time.

Table 12. Group Tracking Algorithm Memory Use

MEMORY	AVAILABLE (BYTES)	USED BY GTRACK (BYTES)	USE (PERCENTAGE)
PROGRAM	103710	12609	12%
DATA	92056	14650	16%

2.3.4 Output Through UART

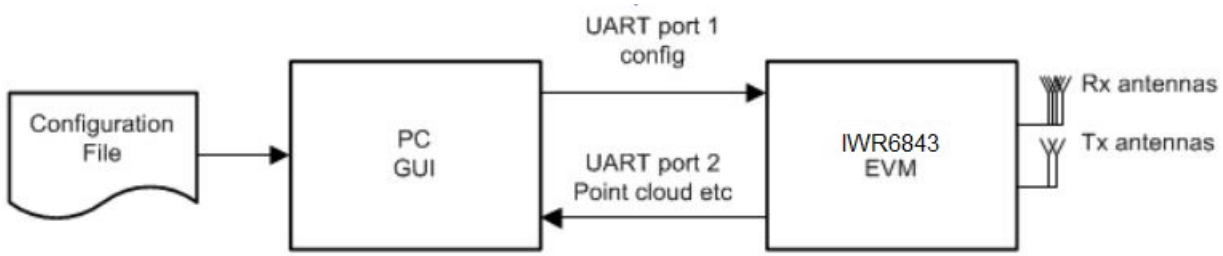


Figure 8. IWR6843 UART Communication

As shown in Figure 8, the example processing chain uses one UART port to receive input configuration from the front end and signal-processing chain, and uses the second UART port to send out processing results for display. See the information included in the software package for details on the format of the input configuration and output results.

2.3.4.1 Output Results Format

The transport process at the R4F outputs one frame every frame period. The frame has a fixed header, followed by a variable number of segments in tag/length/value (TLV) format. Each TLV has a fixed header, followed by a variable size payload. Byte order is little endian. Figure 9 shows a visualization of the data output.

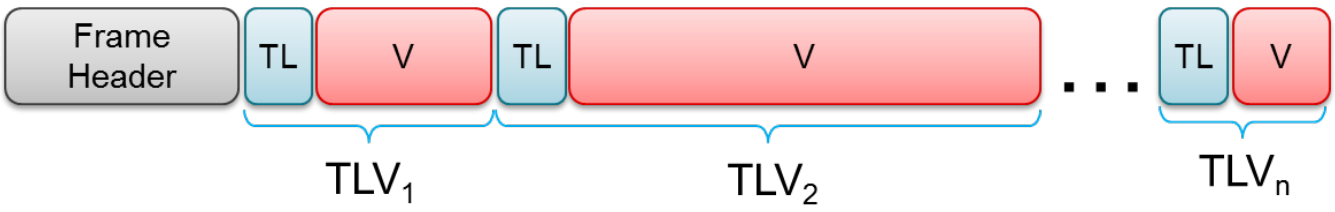


Figure 9. Output Frame Format

2.3.4.2 Frame Header

The frame header is a fixed size (52 bytes) and has following structure (using MATLAB® notation, with name, type, and length in bytes). The header is designed to self-describe the content, and allow the user application to operate in a lossy environment. The header fields are protected with the checksum, see Figure 10.

```
frameHeaderStructType = struct(...
    'sync',              {'uint64', 8}, ... % Sync Pattern
    'version',           {'uint32', 4}, ... % mmWaveSDK version
    'platform',          {'uint32', 4}, ... % IWR6843
    'timestamp',         {'uint32', 4}, ... % 600MHz free running clocks
    'packetLength',      {'uint32', 4}, ... % In bytes, including header
    'frameNumber',       {'uint32', 4}, ... % Starting from 1
    'subframeNumber',    {'uint32', 4}, ...
    'chirpMargin',       {'uint32', 4}, ... % Chirp Processing margin, in us
    'frameMargin',       {'uint32', 4}, ... % Frame Processing margin, in us
    'uartSentTime' ,     {'uint32', 4}, ... % Time spent to send data, in us
    'trackProcessTime', {'uint32', 4}, ... % Tracking Processing time, in us
    'numTLVs' ,          {'uint16', 2}, ... % Number of TLVs in thins frame
    'checksum',          {'uint16', 2});    % Header checksum
```

Figure 10. Frame Header

2.3.4.3 TLV Elements

Each TLV has a fixed header (8 bytes) followed by a TLV-specific payload. Figure 11 shows the TLV header.

```
tlvHeaderStruct = struct(...
    'type',              {'uint32', 4}, ... % TLV object Type
    'length',            {'uint32', 4});    % TLV object Length, in bytes, including TLV header
```

Figure 11. TLV Header

Three TLVs are supported at this time, as follows:

- Point cloud TLV
  - Type = POINT_CLOUD_2D
  - Length = sizeof (tlvHeaderStruct) + sizeof (pointStruct2D) × numberOfPoints
  - Each detection point is defined as in Figure 12.

```
% Point Cloud TLV object consists of an array of points.
% Each point has a structure defined below
pointStruct2D = struct(...
    'range',             {'float', 4}, ... % Range, in m
    'azimuth',           {'float', 4}, ... % Angle, in rad
    'doppler',           {'float', 4}, ... % Doppler, in m/s
    'snr',               {'float', 4});    % SNR, ratio
```

Figure 12. Point Cloud TLV

- Target list TLV
  - Type = TARGET_LIST
  - Length = sizeof (tlvHeaderStruct) + sizeof (targetStruct) × numberOfTargets
  - Each target is defined as in Figure 13.

```
% Target List TLV object consists of an array of targets.
% Each target has a structure define below
targetStruct2D = struct(...
    'tid',              {'uint32', 4}, ... % Track ID
    'posX',             {'float', 4}, ... % Target position in X dimension, m
    'posY',             {'float', 4}, ... % Target position in Y dimension, m
    'velX',             {'float', 4}, ... % Target velocity in X dimension, m/s
    'velY',             {'float', 4}, ... % Target velocity in Y dimension, m/s
    'accX',             {'float', 4}, ... % Target acceleration in X dimension, m/s2
    'accY',             {'float', 4}, ... % Target acceleration in Y dimension, m/s
    'EC',               {'float', 9*4}, ... % Tracking error covariance matrix, [3x3], in
range/angle/doppler coordinates
    'G',                {'float', 4});    % Gating function gain
```

Figure 13. Target List TLV

- Target Index TLV
  - Type = TARGET_INDEX
  - Length = sizeof (tlvHeaderStruct) + numberOfPoints.
  - Payload is a byte array, each byte represents a tracking ID.

NOTE: The target index TLV received in the N-th frame indices the point cloud in (N-1)-th frame.

NOTE: The track ID is a byte. Values 0 to 249 are supported. Values 250 to 255 are reserved.

2.4 Implementation Considerations

2.4.1 Floating-Point Versus Fixed-Point Implementation

The C674x DSP integrated in the IWR6843 offers a rich set of fixed-point and floating-point instructions. The floating-point instruction set can accomplish addition, subtraction, multiplication, and conversion between a 32-bit fixed point and floating point – in a single cycle for a single-precision floating point, and in one to two cycles for a double-precision floating point. The majority of the single-precision, floating-point instructions are at the same speed as the 32-bit fixed-point instructions (the single-precision, floating-point FFT is almost as efficient as a 32-bit, fixed-point FFT). There are also fast instructions to calculate the reciprocal and reciprocal square root in a single cycle with 8-bit precision. With one or more iterations of Newton-Raphson interpolation, the user can achieve higher precision in a few tens of cycles. Another advantage of using floating-point arithmetic is that the user can maintain both the precision and dynamic range of the input and output signal, without spending CPU cycles checking the dynamic range of the signal or rescaling intermediate computation results to prevent overflow or underflow. These enable the user to skip or do less requalification of the fixed-point implementation of an algorithm, making algorithm porting simpler and faster.

With the above, the 16-bit, fixed-point operations are two to four times faster than the corresponding single-precision, floating-point instructions. Trade-offs between precision, dynamic range, and cycles cost must be carefully examined to select suitable implementation schemes.

Using the example of 1D FFT, because the maximum effective ADC bit per sample is about 10 bits, under the noise and cluttered condition the output peak to average ratio is limited (not a delta function in the ideal case), data size does not expand between input and output. Because the deadline requirement for chirp processing is generally tight, a 16-bit, fixed-point FFT is used for the balanced dynamic range and SNR performance, memory consumption, and cycle consumption.

Using the example of 2D FFT, there is additional signal accumulation in the Doppler domain; thus, the output signal peak tends to be very big. A single-precision, floating-point FFT is used, so adjusting the input signal level (which may cause SNR loss) or having a special FFT to have dynamic scaling for each butterfly is not required—both could have much higher cycle cost. In addition, because there is a 2D windowing function before FFT, the data reformatting from 16-bit IQ to single-precision, floating-point and 2D windowing can be combined without additional cycle cost. The drawback of the floating-point FFT is that the output data size is doubled from the input data size. The 2D FFT results cannot be stored back to the radar cube. For DoA detection, reconstructing the 2D FFT results per detected object is required at additional cycle cost.

For the example of clustering, the 16-bit fixed-point can safely cover the dynamic range and precision requirements of the maximum range and range resolution. The arithmetic involved is the distance between the two points and decision logic, which can be easily implemented using 16-bit, fixed-point multiplications instructions and 32-bit fixed-point condition check instructions. Thus, a fixed-point implementation is used, which is approximately two times the cycle improvement of the floating-point implementation.

2.4.2 EDMA Versus DSP Core Memory Access

Enhanced direct memory access (EDMA) provides an efficient data transfer between various memories with minimum DSP core intervention and cost. In general, data movement in the radar processing chain is regular and ordered, whereas data from lower-level, slow memory is moved to higher-level faster memory for DSP processing, then transferred back to lower-level memory for storage. Thus, EDMA is the preferred way to accomplish most of these data movements. Specifically, the ping-pong scheme can be used for EDMA to parallelize data transfer and signal processing, so that at a steady state, there is no overhead for data movement.

There are a few scenarios that must consider the trade-offs between using EDMA and direct core access.

First, if there is irregular data access pattern for a processing module, using EDMA would be very cumbersome and sometimes impractical. For the example of the two-pass CFAR algorithm used in the example signal processing chain, a CFAR-CASO search must be conducted in the range domain; then, immediately conduct a CFAR-CA search in a doppler domain to confirm the results of the first pass. For this 2D alike search, using EDMA for data movement could be cumbersome. Thus, DSP is used to access the L3 memory directly with an L1D cache with L3 memory turned on. Cycle performance degrades to

1.8× to 2× of the entire power heat map stored in L2 memory (thus no EDMA involved). Flexibility is gained, if it is required to change the search order or do other algorithm tuning because no hardcoded EDMA is tied with this implementation, and there is no requirement to use any local buffer in the L2 memory to store the power heat map. With the current memory usage and cycle cost, it is a good design choice.

Secondly, when the size of the data transfer is small, the EDMA overhead (setting up PaRamSet, triggering the EDMA, and checking the finish of EDMA) compared to the signal processing cost itself increases, and it might be more cycle-efficient to use direct DSP access to L3 with the L1 cache on. It has been observed for small 2D FFT size of 32, direct core access costs less cycles than using EDMA. In addition, code is simpler without the ping-pong scheme and EDMA.

2.4.3 DSP Memory Optimization

To optimize the DSP memory, portions of the L1D and L1P are configured as SRAM.

There are 32KB of L1D and 32KB of L1P in C674x. Typically, memory is configured as L1D cache and L1P cache as a whole, but for the radar processing chain, the data and program memory footprint is relatively small, which makes it possible to carve out a portion of L1D and L1P and use them as SRAM, without any cycle performance impact.

In this implementation, 16KB of L1D are configured as the L1 data cache. The remaining 16KB are configured as data SRAM. The EDMA input and output ping-pong buffers are allocated in this fast memory and shared between range processing and Doppler processing.

4KB of L1P is configured as L1 program cache. The remaining 28KB is configured as program SRAM, which holds the majority of the real-time frame-work code and all algorithm kernels except tracking.

With this implementation, 40KB of L2 memory was saved, which can be used for adding new algorithms or for other optimizations. There was approximately a 5% to 10% cycle improvement for range processing, while no cycle penalty for other modules with data buffers in L2 or L3 memory was observed. Specific to range processing, because all functions are in L1P SRAM, all input and output buffers are in L1D SRAM and only FFT twiddle factors are in L2, but the FFT will be fetched to the L1D cache and stay there for the all antennas and all chirps. There is very small cycle fluctuation because there is fewer L1 cache operations in the background.

3 Hardware, Software, Testing Requirements, and Test Results

3.1 Required Hardware and Software

3.1.1 Hardware

The IWR6843 ISK + ICB bundle is required to get the demonstration running (see Figure 14).

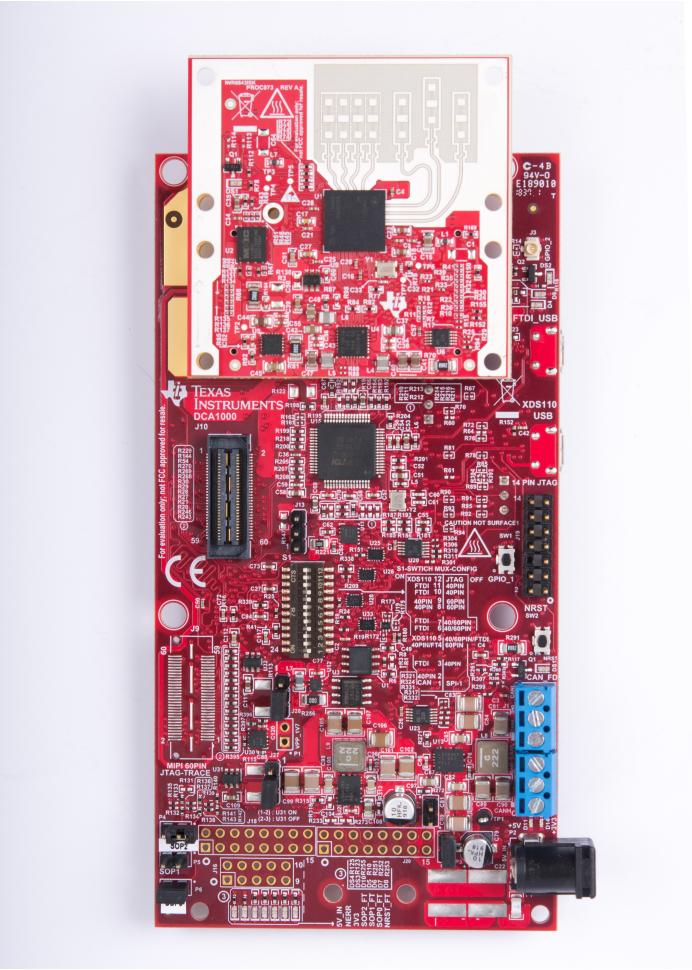


Figure 14. IWR6843 EVM

3.1.2 Software

- mmWave SDK
- People-Counting Application Project

3.2 Testing and Results

3.2.1 Test Setup

The people-counting system was set up for five different test cases, to characterize and demonstrate the capabilities of the system.

The test cases were:

1. Waypoint localization
2. People Counting Density
3. People Counting Accuracy
4. People counting in a conference room
5. Extending detection range

For each of the test cases, the sensor was mounted to a tripod and elevated to a height of 1.8 to 2.2 m. The sensor was positioned in the environment so that the field of view encompassed the area of interest, and was oriented towards the direction in which people would enter the scene. The downtilt of the sensor was adjusted from 10 to 30 degrees, depending on the test case, to achieve a balance between the resulting signal strength and noise floor.

3.2.2 Test Results

3.2.2.1 Test Scenario 1: Waypoint Localization

In this test scenario, the radar sensor was placed in a room oriented with no azimuth tilt. Defining the location of the radar in physical environments with Cartesian coordinate locations of (0,0), a line was measured and taped out extending from (0,0) to (0,6). Along this line, waypoint markers were placed every meter. Figure 15 shows this setup. The test subject then walked to each of the waypoint markers and stood at the marker. The localization accuracy of the tracker was computed by comparing the difference between the centroid coordinates of the test subject returned by the tracker, versus the known physical location of the test subject.

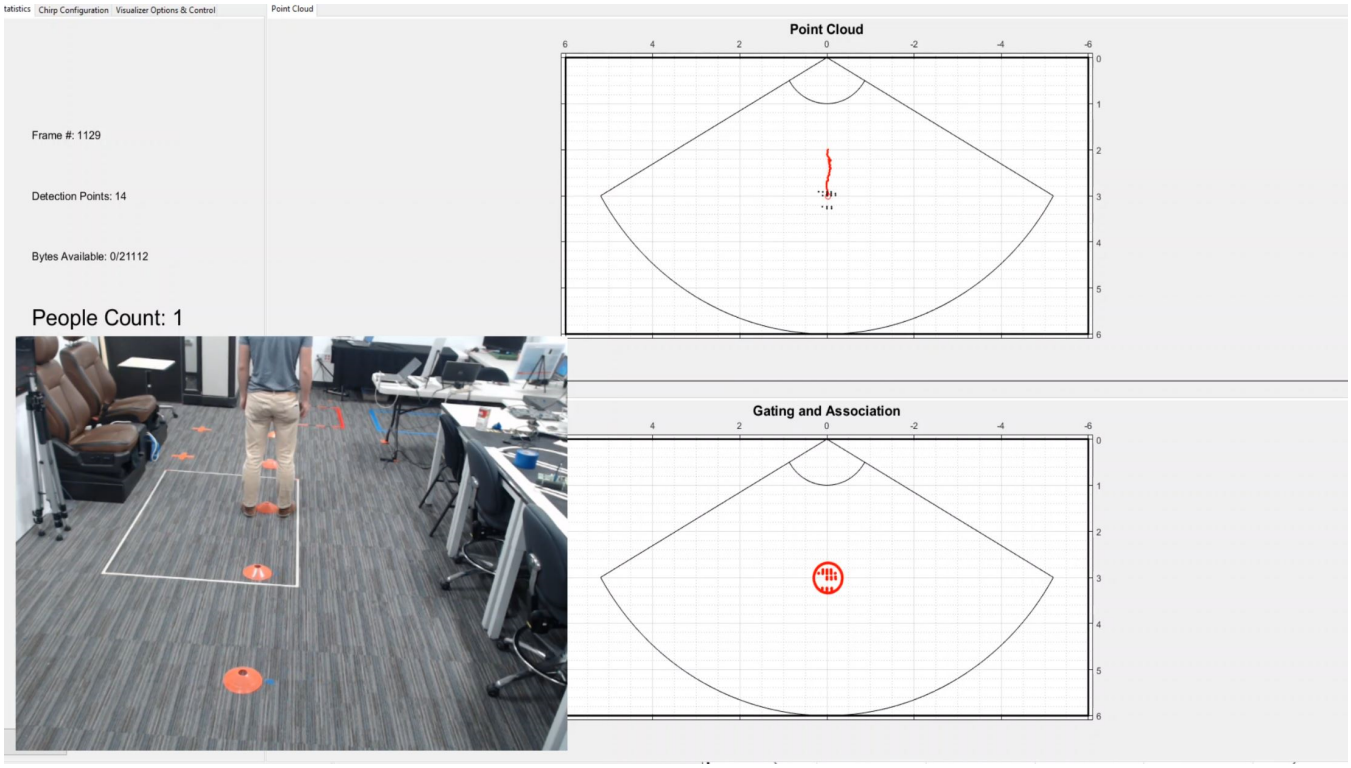


Figure 15. Ground Truth and Visualizer Snapshot With Person at Waypoint 4 m

Test Scenario 1 Observations

The results of the scenario show the localization capabilities of the people-counting system (see Table 13). The centroid of the point clouds were on average at least within 0.05 meters of the point where the person was aiming to stand.

Table 13. Tracker Accuracy

WAYPOINT	OBJECT CENTROID COORDINATES REPORTED BY TRACKER		LOCALIZATION ACCURACY (Raw Error meters)	
	X [m]	Y [m[	X	Y
(0,5)	0.0176	5.0046	0.0176	0.0046
(0,4)	0.03	4.0125	0.03	0.0125
(0,3)	0.0246	2.9981	0.0246	0.0019
(0,2)	0.0615	1.9897	0.0615	0.0103
(0,1)	0.1655	1.0165	0.1655	0.0165
—	—	Average	0.05984	0.00916

3.2.2.2 Test Scenario 2: People Counting Density

In this test scenario, the radar sensor was placed against the wall of a room. One subzone was defined in the software, and this was manifested in the physical world by marking the floor with tape to indicate the corners. The box was 1x1 m in size. In this scene, three people walked into the scene, and then, one by one, walked into the box, as shown in Figure 16. Figure 17 shows how all three people were counted as they stood near each other.

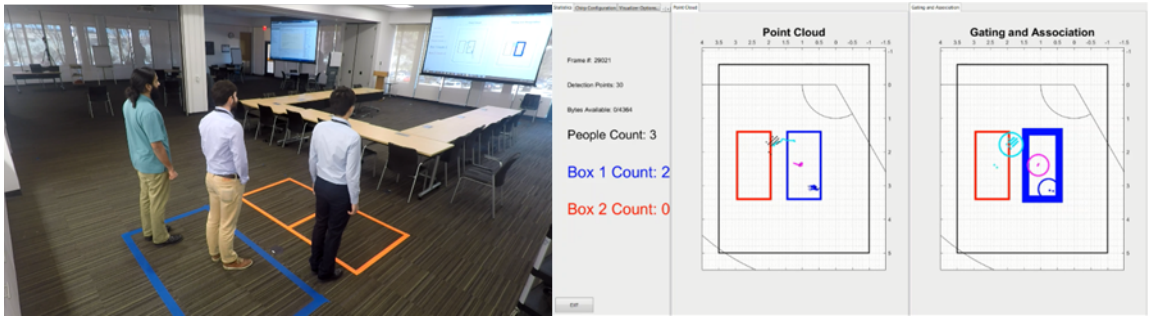


Figure 16. Test Subjects Tracked Entering and Exiting Subzones

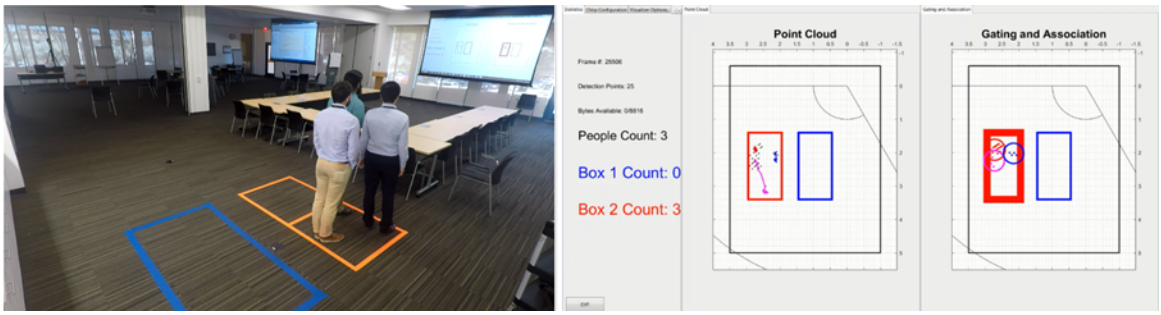


Figure 17. Tracker Successfully Detects Three Test Subjects in 1 m²

Test Scenario 2 Observations

In applications where people can be standing in close proximity to each other, performance with high crowd densities can be of concern as the point clouds merge together. It is shown that even with the default tracker settings provided for the people-counting system, a density of three people per square meter could be achieved.

3.2.2.3 Test Scenario 3: People Counting Accuracy

In this test scenario, the radar sensor was placed against the wall of a room. An area of interest was defined in front of the sensor. People walked into the area of interest one at a time, until the desired number of people was reached. Then 3000 frames of data, which corresponds to 2 minutes and 30 seconds, were captured. This exercise was repeated with 1, 3 , 5, 7, 9, and 11 people. The table below shows accuracy when counting each number of people. Accuracy is defined as the percentage of frames where the count is correct, or within a specific range, e.g. plus or minus 1 of true value.

Table 14. People Count Accuracy

	People Count Percent Accuracy		
Number of People in Scene	+- 0 People	+- 1 Person	+-2 People
1	99%	100%	100%
3	95%	100%	100%
5	51%	85%	100%
7	59%	85%	98%

Table 14. People Count Accuracy (continued)

	People Count Percent Accuracy		
9	14%	43%	84%
11	6%	45%	51%

3.2.2.4 Test Scenario 4: People Counting in a Conference Room

The radar sensor was placed in the corner of a 3.5 × 4.5-m conference room that contained furniture, including a table and chairs. In this scene, four people entered the room and sat in the chairs at the table. Figure 18 shows a snapshot of the output of the sensor and the scene in the room. The static objects, tables, and chairs are not counted as objects. The arrow indicates multipath reflections from the walls that are not considered objects as a result of the algorithm specifications.

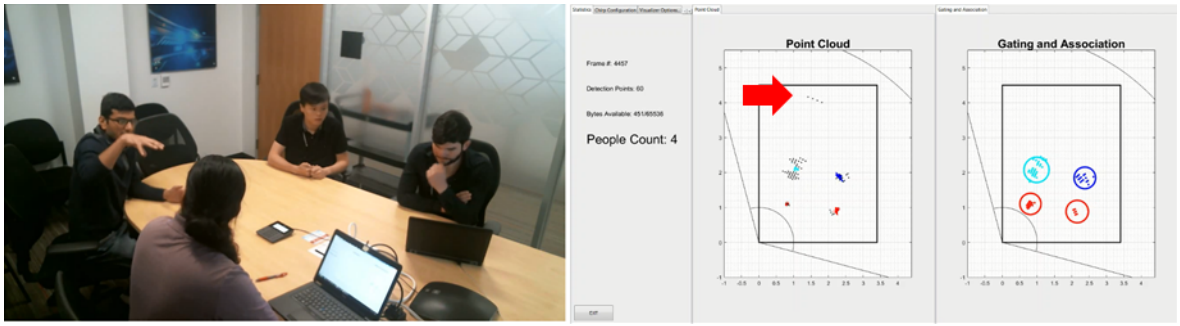


Figure 18. Ground Truth and Visualizer Snapshots of Sensor Mounted in Conference Room

Test Scenario 3 Observations

The results from the conference room test scenario demonstrate the ability of the people-counting system to count people who are sitting, in addition to walking or standing as in the previous test scenarios. Furthermore, it is shown to function in a radar-challenging environment with clutter and nearby walls that can introduce multipath reflections.

3.2.2.5 Test Scenario 4: Extending Detection Range

The design goal for the chirp for the people counting system was 6m. Using this same chirp, the frequency slope was decreased from 60 to 24 to enable a maximum range of 14 m. The details of this chirp are in Table 2, and the key tradeoff for increasing the maximum range was a decrease in range resolution. This chirp was used with the sensor mounted in a hallway. As shown in Figure 19, the system detected three total people in the scene, as the third person at 14 m approaches the sensor. The sensor continues to track the person from when he entered the scene to when he existed the sensor's field of view, as shown in Figure 20.

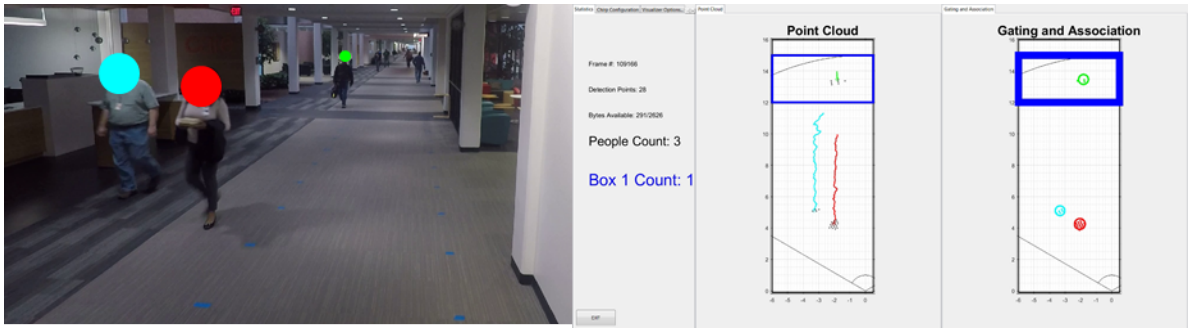


Figure 19. Ground Truth and Visualizer Snapshot as Person Enters, Tracked at 14 m While Two Other People are Counted in Scene

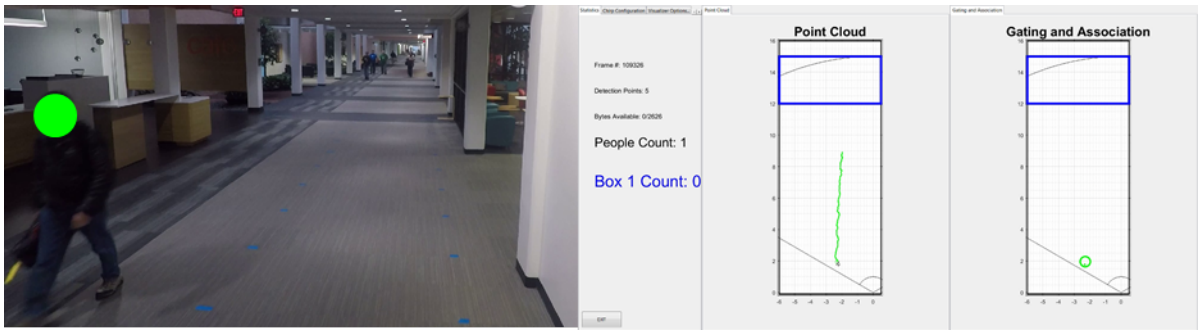


Figure 20. Ground Truth and Visualizer Snapshot as Person Exits the Sensor Field of View

Test Scenario 4 Observations

This test scenario shows the ability to change the detection and tracking range of the people-counting system by modifying the chirp configuration used. A person is shown to be detected at 14 m, and the history of the person's movement is tracked as he moves through the scene.

4 Design Files

4.1 Schematics

To download the schematics, see the design files at IWR6843.

4.2 Bill of Materials

To download the bill of materials (BOM), see the design files at IWR6843.

4.3 Altium Project

To download the Altium Designer® project files, see the design files at IWR6843.

5 Software Files

To download the software files, see the design files at IWR6843.

6 Related Documentation

1. Texas Instruments, *IWR6843 Data Sheet*
2. Texas Instruments, *IWR68xx/16xx/14xx Industrial Radar Family*, technical reference manual
3. Texas Instruments, *mmWave SDK*, tools folder

6.1 Trademarks

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from B Revision (November 2018) to C Revision **Page**

- Changed 57 GHz to 60 GHz in Section 1 2
- Changed 57 GHz to 60 GHz throughout Section 2.2.1 6
- Changed 7-GHz available bandwidth to 4-GHz available bandwidth in Section 2.2.1 6
- Changed 84 GHz to 64 GHz in Section 2.2.1 6

Changes from A Revision (April 2018) to B Revision **Page**

- Changed all instances of IWR1642 to IWR6843 1
- Changed all instances of 77 GHz to 60 GHz in reference to the IWR6843 1
- Changed information in Description 1
- Added note regarding IWR1642 2
- Changed information in Section 1 2
- Changed data in Table 1 2
- Changed information in Section 2.1.2.2 4
- Changed information in Section 2.2.1 6
- Deleted *ADC data capture demonstration* and *ADC data streaming demonstration* from Section 2.2.2.2 8
- Changed data in Table 2 10
- Deleted *Defining subzones for people counting* in Section 3.2.1 23
- Added *People Counting Density* in Section 3.2.1 23
- Added *People Counting Accuracy* in Section 3.2.1 23
- Changed 2 m to 2.2 m in Section 3.2.1 23
- Changed information in Section 3.2.2.1 24
- Deleted *Ground Truth and Visualizer Snapshot With Person at Waypoint 1 m* image from Section 3.2.2.1 24
- Changed information in Section 3.2.2.2 25
- Added Section 3.2.2.3 25

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265