设计指南：*TIDEP-01018*
使用毫米波传感器的自动门参考设计

说明

TIDEP-01018 参考设计展示了 IWR6843 ISK 的使用方法，IWR6843 ISK 是 TI 推出的一款采用集成 DSP 的单芯片毫米波雷达传感器，适用于自动智能门应用。此参考设计采用了 IWR6843 ISK + ICB（工业载板），并在 IWR6843 器件中集成了完整的雷达处理链。本解决方案设计用于跟踪人员位置和速度，从而在人员靠近时模拟门的打开操作。

资源

TIDEP-01018	设计文件夹
IWR6843	产品文件夹
IWR6843ISK IWR6843 智能毫米波传感器标准天线插件模块	ISK/ICB 捆绑包工具文件夹

TI E2E™ Community

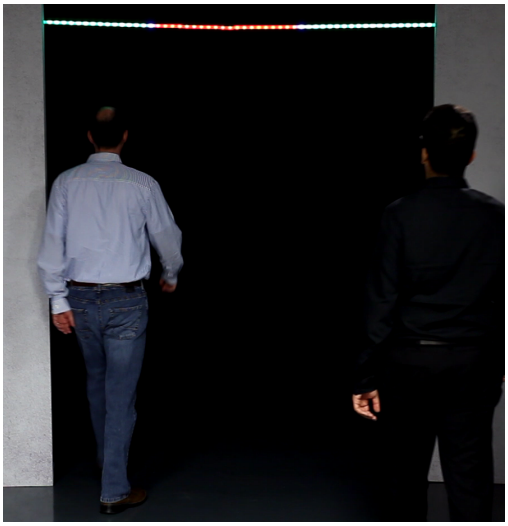
咨询我们的 E2E™ 专家

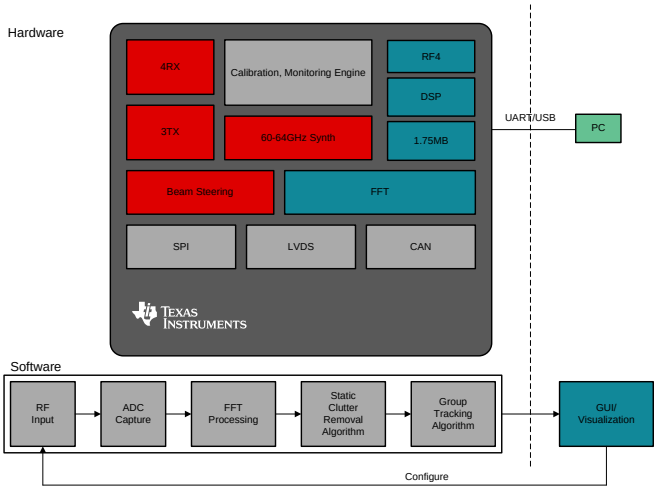
特性

- 使用用于入口系统的毫米波雷达传感器 IWR6843 展示相关硬件和软件
- 毫米波技术可提供非常适用于环境效应的距离、速度和角度信息
- 跟踪算法有助于确保门只在人员直接靠近时打开
- 在 5m 距离内方位视场角为 120 度
- 提供静态干扰消除和群组跟踪算法实现示例

应用

- 自动门
- 门和防护入口通道
- 旋转栅门
- 照明
- 区域扫描仪安全防护装置





该 TI 参考设计末尾的重要声明表述了授权使用、知识产权问题和其他重要的免责声明和信息。

1 System Description

The TIDEP-01018 provides a reference for an automated door application that can detect and track humans up to 5 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.

Entrance systems, such as sliding doors and turnstiles, should leverage sensor-driven intelligence to allow their systems to operate efficiently. The ability to continuously and consistently monitor human motion is an important function for this; moreover, while sensors such as passive infra-red (PIR) are in use, they suffer from limitations in accuracy, false alarms, and certain environmental changes. Radars allow for 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.

The design provided is an introductory-demo application developed on the IWR6843 ISK to simulate the opening of a door only when a person is directly approaching it. The radar data processing chain implements the static clutter removal, range azimuth heat map, and Doppler extraction algorithms for obtaining point cloud information. Furthermore the post-processed object data is used to determine the correct time a door should open such that it will open in time for its target. After viewing this design, the customer should be able to better understand the implementation of advanced algorithms on the IWR6843 device.

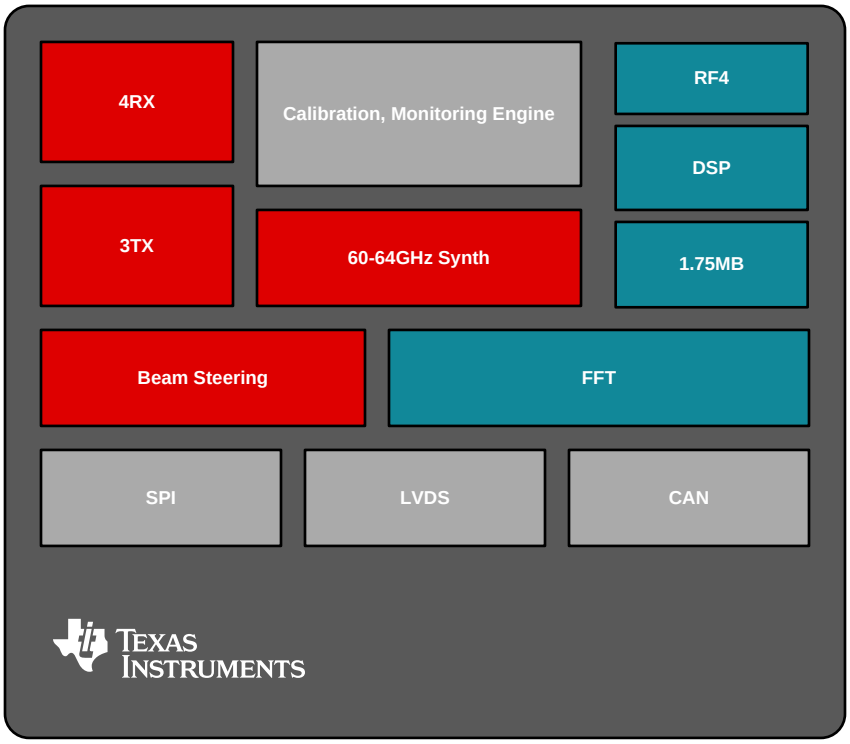
2 System Overview

2.1 Block Diagram

2.1.1 Hardware Block Diagram

The TIDEP-01018 is implemented on the IWR6843 ISK EVM as shown in 图 1. The EVM is connected to a host PC through a universal asynchronous receiver-transmitter (UART) for visualization.

图 1. Hardware Block Diagram

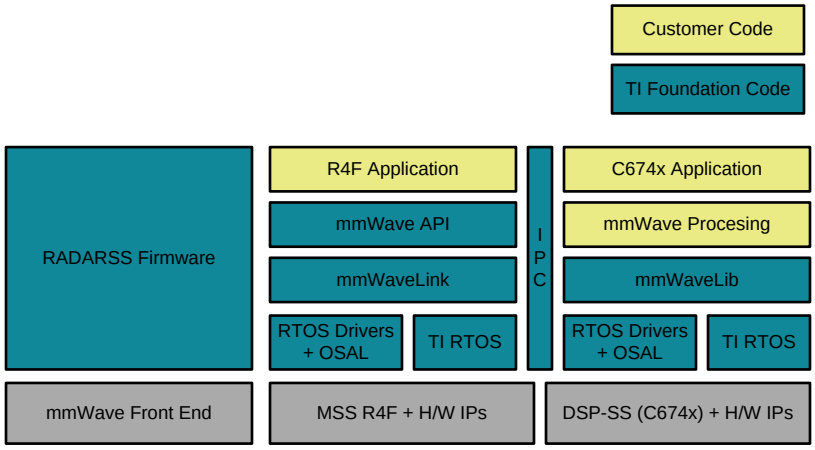


2.1.2 Software Block Diagram

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 as shown in 图 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.

图 2. mmWave SDK Software Block Diagram

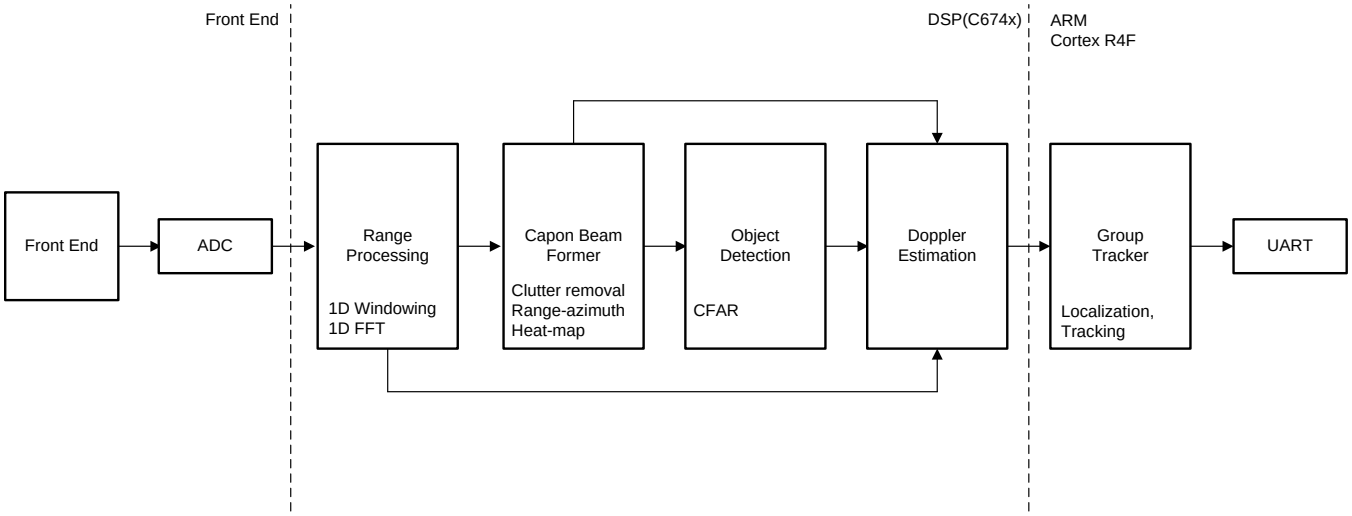


2.1.2.2 Software Block Diagram of Automated Doors Application

As shown in 图 3, the implementation of the Automated Doors 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
  - Tracking information used to trigger opening of door at specific time with regard to a person's position and velocity.

图 3. Automated Doors Application Block Diagram

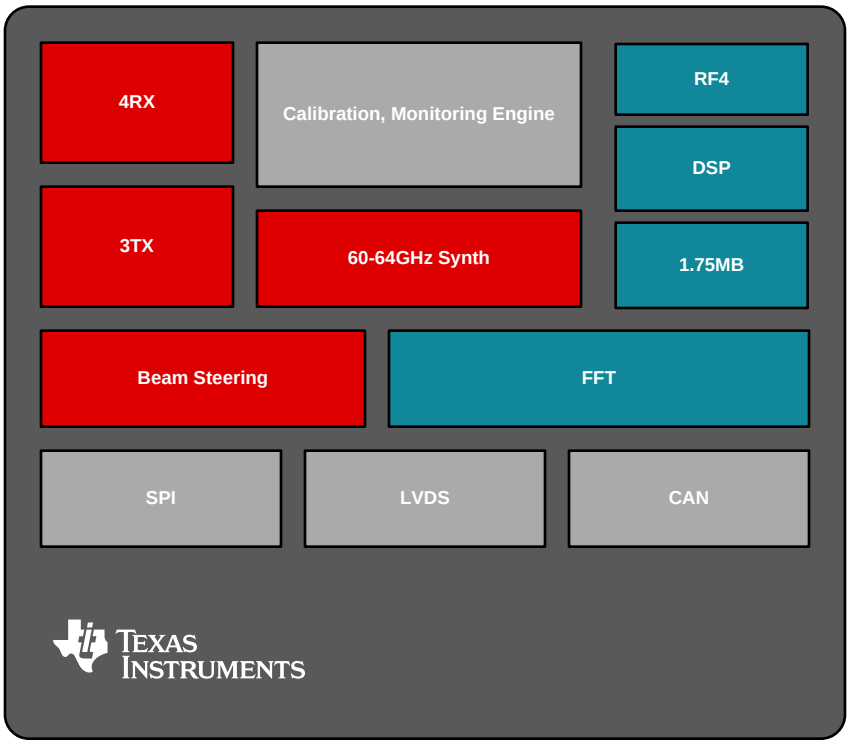


2.2 Highlighted Products

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 as shown in 图 4. 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.

图 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.

2.3.2 Low-Level Processing

An example of a processing chain for automated doors 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. 表 1 lists the several physical memory resources used in the processing chain.

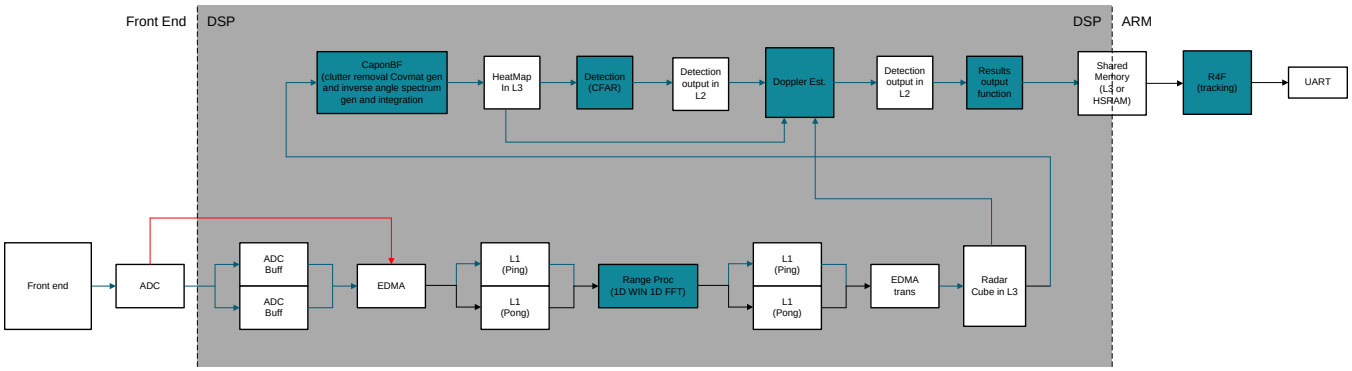
表 1. Memory Configuration

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 cacheable. 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.

表 1. Memory Configuration (continued)

SECTION NAME	SIZE (KB) AS CONFIGURED	MEMORY USED (KB)	DESCRIPTION
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.

图 5. Processing Chain Flow: Detection-Tracking Visualization



As shown in 图 5, the implementation of the Automated Doors 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(θ)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 Automated Doors 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.

表 2 lists the results of benchmark data measuring the overall MIPS and memory consumption of the processing chain, up to and including the tracking.

表 2. MIPS Use Summary

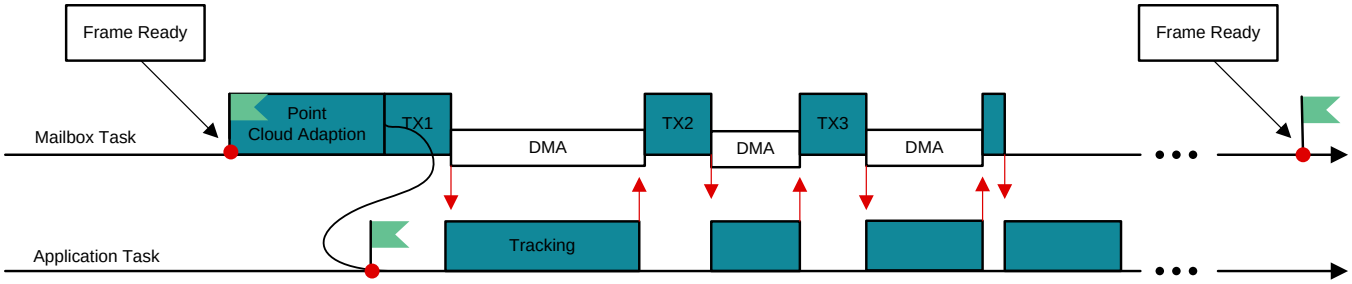
PARAMETER	AVAILABLE TIME	USED TIME	LOADING
Active chirp time	92 µs	14 µs	15%
Frame time	26.5 ms	7.2 ms	27%

2.3.3 High-Level Processing Details

2.3.3.1 Task Model

As shown in 图 6 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.

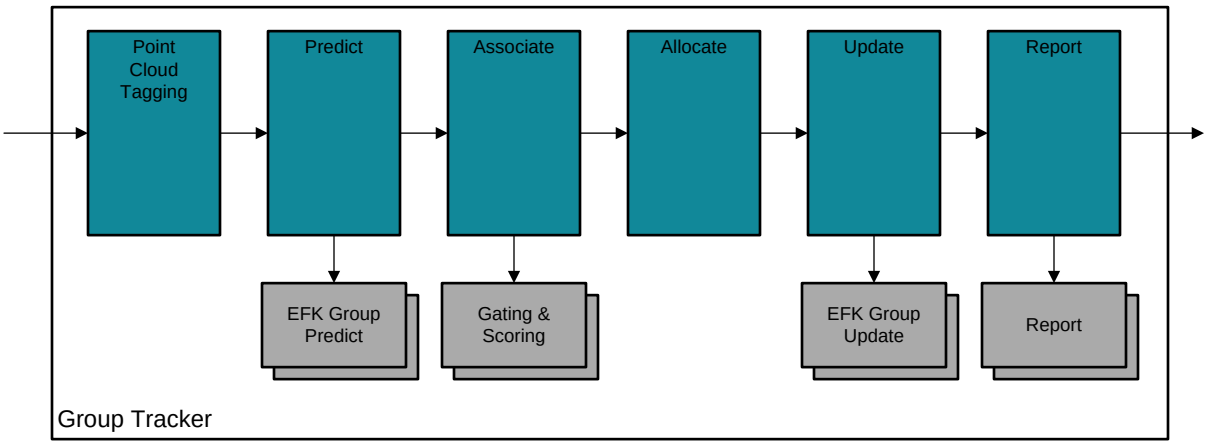
图 6. High-Level Processing Task Model



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. 图 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.

图 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 –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 表 3.

表 3. 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. 表 4 lists these parameters.

表 4. Scenery Parameters

PARAMETER	DEFAULT	DIM	DESCRIPTION
leftWall	–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. 表 5 lists these parameters.

表 5. 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. 表 6 lists these parameters.

表 6. 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 节 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.

表 7 lists the parameters used to set this behavior.

表 7. 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. 表 8 lists each of these parameters.

表 8. 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 © = 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. 表 9 lists the total memory footprint, indicating memory use.

表 9. Cortex-R4F Memory Use

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

表 10 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.

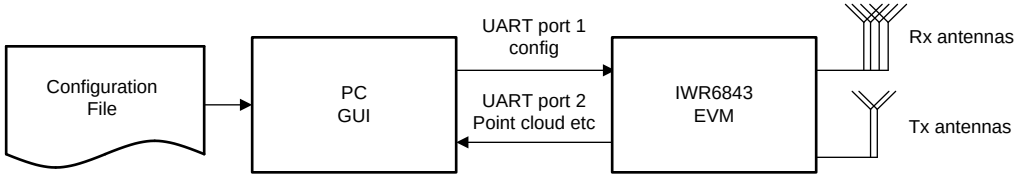
表 10. Group Tracking Algorithm Memory Use

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

2.3.4 Output Through UART

As shown in 图 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.

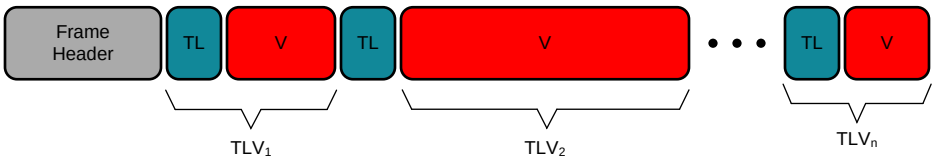
图 8. IWR6843 UART Communication



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. 图 9 shows a visualization of the data output.

图 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 图 10.

图 10. Frame Header

```
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
```

2.3.4.3 TLV Elements

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

图 11. TLV Header

```
tlvHeaderStruct = struct(...
    'type',             {'uint32', 4}, ... % TLV object Type
    'length',           {'uint32', 4});    % TLV object Length, in bytes, including 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 图 12.

图 12. Point Cloud TLV

```
% 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
```

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

图 13. Target List TLV

```
% 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
```

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

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

注: 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.

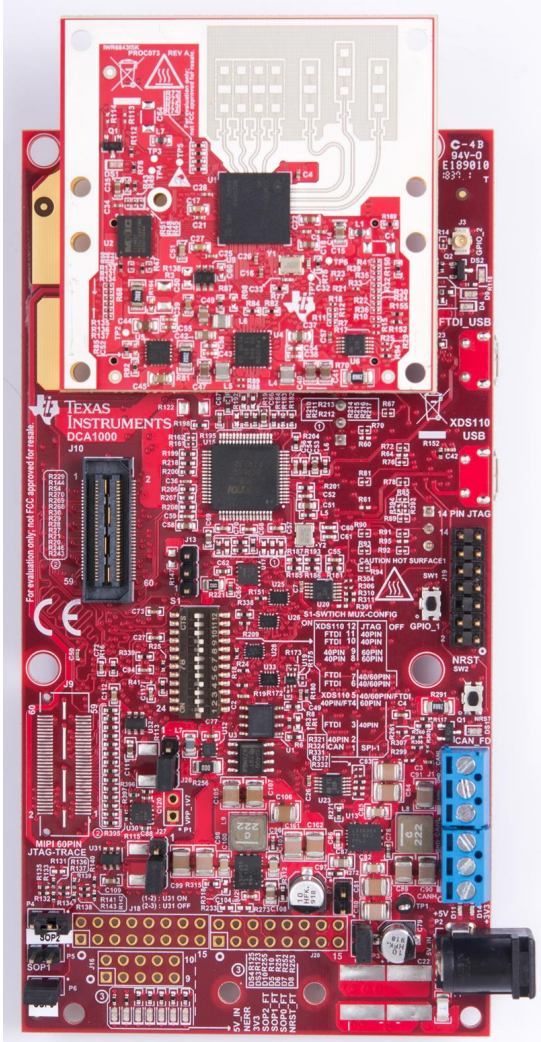
3 Hardware, Software, Testing Requirements, and Test Results

3.1 Required Hardware and Software

3.1.1 Hardware

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

图 14. IWR6843 EVM



3.1.2 Software

- mmWave software development kit (SDK)
- Automated Doors User's Guide

3.2 Testing and Results

3.2.1 Test Results

3.2.1.1 Test Scenario 1: Single Person, Single Lane

This fundamental scenario was conducted to test basic functionality. The IWR6843 should detect if a person is walking away from a door or towards it. If a person is walking away from the door, the door should stay entirely closed as shown in 图 15. If a person is walking towards a door, as shown in 图 16, the door should open when a person is within three seconds of reaching it.

The GUI uses three lane indicators to show if a door is closed (red) or open (green), the open state will persist for fifty frames if more people are walking through the door. The GUI also features a plot of a scene which tracks people in a scene with an added vector to indicate velocity and direction. Please note that for the below tests, the GUI has been inverted to reflect the point-of-view of the mmWave Sensor.

This threshold of three seconds is established at code level and is configurable for a particular system; moreover, the time is calculated as a function of a person's both range and velocity with respect to the door.

图 15. Person Moving Away From Sensor. All Lanes Stay Red Indicating a Closed State

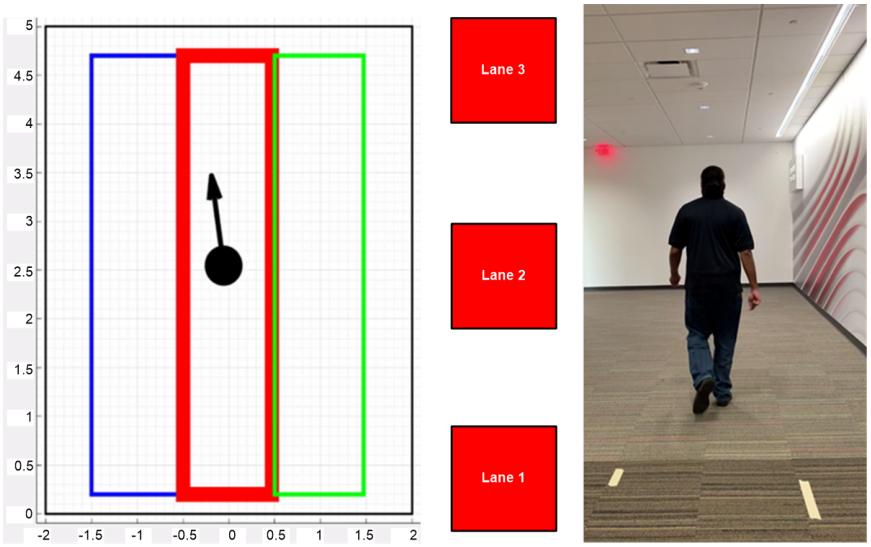
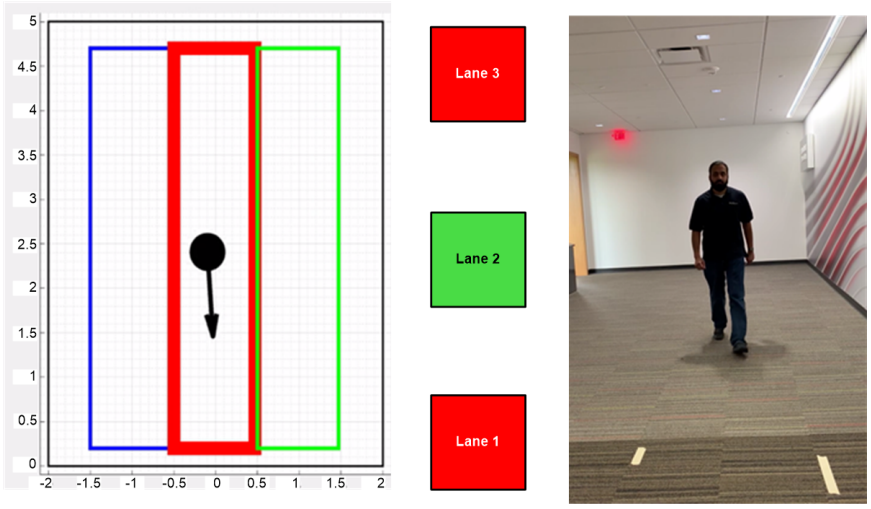


图 16. Person Moving Toward the Sensor. Middle Indicator Turned Green



3.2.1.2 Test Scenario 2: Walking Past a Door

One condition for this design was that the door should not open unless a person is walking towards it which is what this test aimed to show. This was clearly observed as the door stayed in a closed state. This is accomplished through the use of range, angle, and velocity information in order to plot and track a persons position and direction of movement.

图 17 shows a single person walking past the door while all three lanes remain closed. 图 18 shows two people walking past the door while all three lanes also remain closed. This specific scenario is important because it shows the sensor can resolve the position and direction multiple people in a scene.

图 17. One Person Walking Past But Not Towards Door

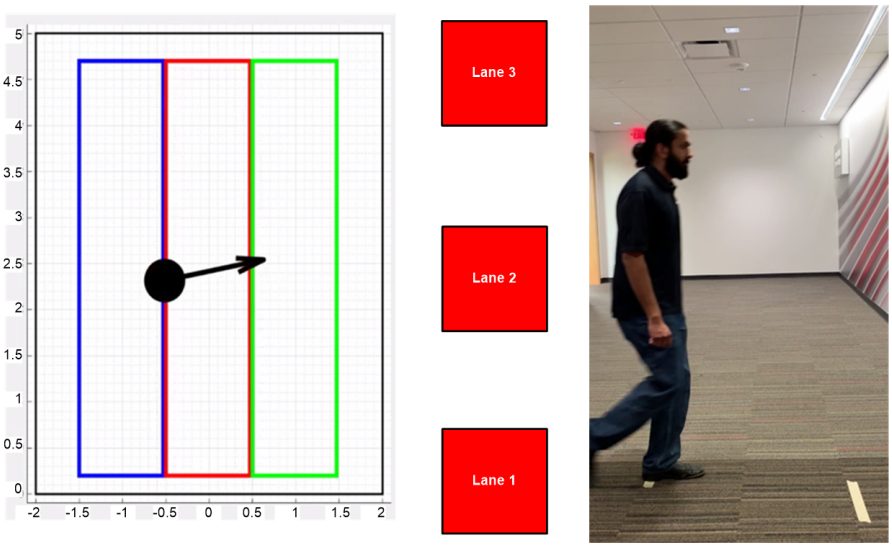
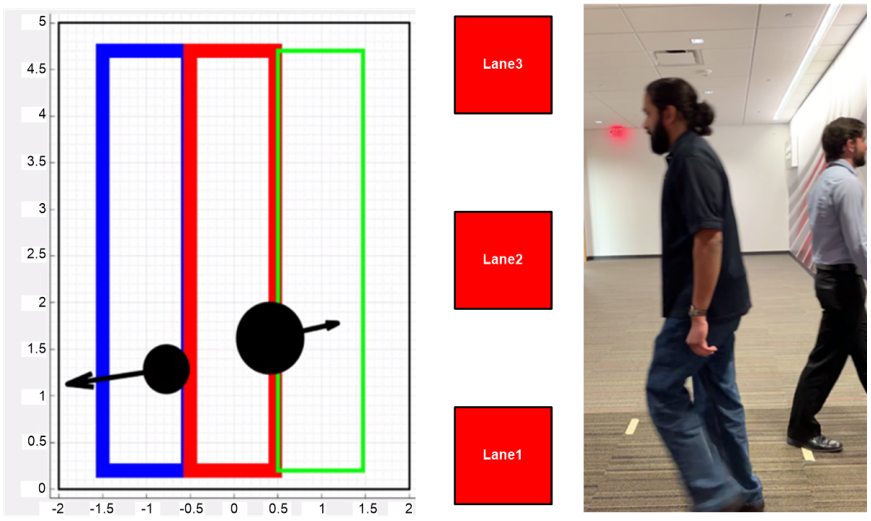


图 18. Two People Walking Past Door

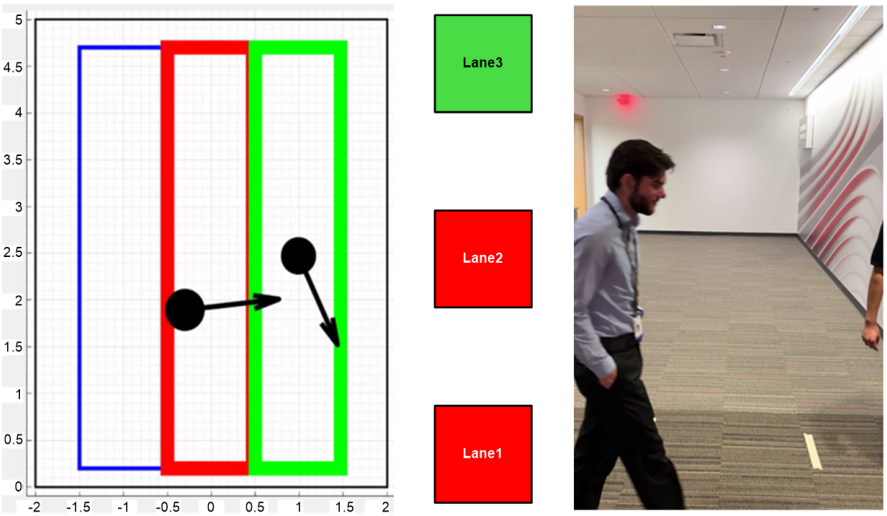


3.2.1.3 Test Scenario 3: Walking Past and Towards

This test served to combine the factors in the previous two test scenarios and the specific conditions of those scenarios. A person would walk past the door which should not trigger any door to open but another person would walk toward the door which should trigger the door to open. This scenario also shows the sensor's capability to resolve multiple people appropriately.

图 19 shows this specific case where one person that has walked past the left and middle lanes, which remain closed, while the person walking toward the sensor in the right hand lane has triggered the door in an open state.

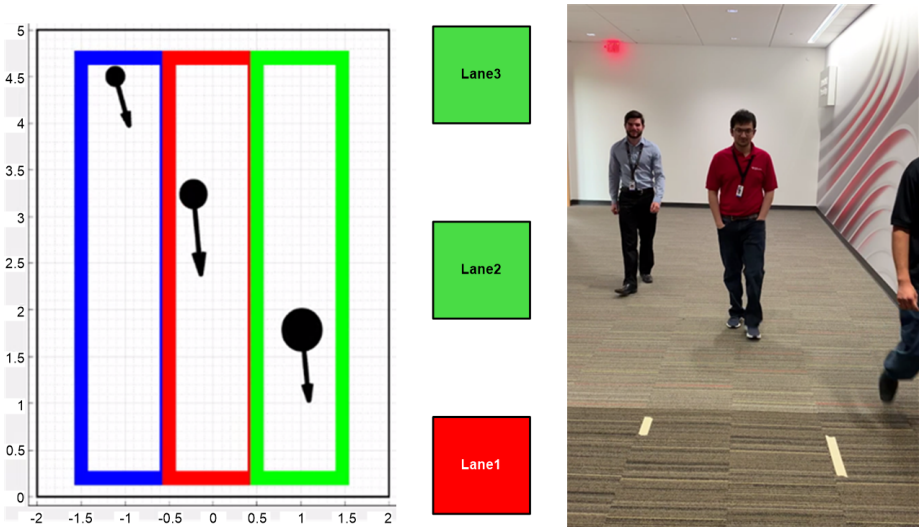
图 19. One Person Walking Past and Another Person Walking Toward Door



3.2.1.4 Test Scenario 4: People in Three Lanes

The final test conducted was to have three people walking down each of the lanes in order to observe independent functionality. The plot in 图 20 accurately shows each person in a separate lane walking toward the door. What is important to note is that while the person in the left lane is present in the plot, their lane is indicated as being in a closed state. This is due to the timing threshold set on each lane. The person in left lane has been clearly detected but the sensor has computed that he is not within three seconds of reaching the door, leaving it in a closed state.

图 **20. Three People Walking Towards Door**



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 商标

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265