ICMS board v3.0: Point cloud generation for ICMS vital sign processing

using XENSIV™ BGT60ATR24AIP 60 GHz radar sensor and AURIX™ TC356/TC336DA

About this document

Scope and purpose

This document describes a software project for the ICMS board v3.0 for the application of in-cabin monitoring systems (ICMS). It demonstrates the main processing steps that need to be performed on AURIX™ TC336DA/356TA Signal Processing Unit (SPU) and Tricore™ CPUs to generate point cloud data from raw data recorded by the XENSIV™ BGT60 ATR24AIP MMIC and to calculate the breathing rate of a detected target.

Intended audience

This document is intended for design engineers, technicians, and developers of electronic systems, working with Infineon's 60 GHz radar sensors and AURIX™ TC336DA/356TA microcontroller on ICMS applications.

Keypoints

- Demonstrates a fundamental software building block for an in-cabin monitoring system (ICMS)
- Demonstrates how to establish SPI communication between the AURIX™ TC356 microcontroller and the XENSIV™ BGT60ATR24EAIP, enable the BGT60ATR24EAIP 's 40 MHz clock output, and switch the TC356 clock to this external source
- Details the specific signal processing steps for point cloud generation and breathing rate detection on AURIX™ TC3xx microcontrollers
- Shows how to utilize the Signal Processing Unit (SPU) and TriCore™ CPUs for efficient radar data processing

About this product family

Product family

The AURIX™ TC3xx family is a range of 32-bit multicore microcontrollers based on Infineon's TriCore™ architecture. These devices are optimized for real-time embedded systems, particularly for safety-critical automotive applications. The family integrates powerful processing capabilities, including multiple TriCore™ CPUs and a dedicated Signal Processing Unit (SPU), designed to accelerate complex algorithms like those required for radar signal processing.

Target applications

- In-cabin monitoring system (ICMS)

Table of contents

About this document **1**
About this product family **1**
Table of contents **2**
1 ICMS3.0 board overview and key features **3**
2 Software components **5**
2.1 Overview 5
2.2 Transferring radar data from BGT60ATR24EAIP to TC356TA: SPI and DMA setup 6
2.3 40 MHz clock supply 10
2.4 Radar and MMIC configuration 11
2.4.1 Overview 11
2.4.2 Changing MMIC parameters using XENSIV™ Radar Fusion GUI 12
2.5 Radar signal processing 13
2.5.1 Point-Cloud generation 13
2.5.1.1 Signal processing overview 13
2.5.1.2 Sorting ADC data 15
2.5.1.3 Range FFT and antenna calibration 15
2.5.1.4 Building a micro-Doppler frame from multiple radar frames 16
2.5.1.5 Static target removal 16
2.5.1.6 Capon spectrum: Creating a range angle map 19
2.5.1.7 Detect a target on the range angle map 22
2.5.2 Vital signs: Breathing rate detection 23
2.5.2.1 Signal processing overview 23
2.5.2.2 Beamforming 23
2.5.2.3 Phase extraction and phase unwrapping 23
2.5.2.4 Mean removal 24
2.5.2.5 Phase FFT 24
2.5.2.6 Spectrum filtering, peak selection, and breathing rate detection 25
2.6 Memory overview 26
2.7 Debugging 27
2.8 Adaptions required for using 60 GHz radar baseboard (TC336DA) 27
3 Related resources **29**
References **30**
Revision history **31**
Disclaimer **32**

1 ICMS3.0 board overview and key features

The ICMS board, illustrated in Figure 1, is a 2.5 mm x 5 mm printed circuit board (PCB). Its main purpose is to provide an evaluation platform for embedded radar applications using Infineon's XENSIV™ BGT60ATR24AIP radar sensor and the AURIX™ TC356TA microcontroller. Due to its compact size, this board is targeted at automotive in-cabin sensing applications such as child presence detection (CPD), seat occupancy detection (SOD), and vital signs monitoring. The board block diagram is illustrated in Figure 2.

Its key features are:

- **AURIX™ TC356TA –** The AURIX™ TC356TA is the computing device on the radar baseboard. It features three TriCore™ running at a clock frequency of 300 MHz and two dedicated radar signal processing units (SPU). Furthermore, it has 3.6 MB of SRAM and 4 MB of flash memory. For further information, refer to the AURIX™ TC356TA datasheet [1]
  Alternatively, AURIX™ TC336DA can be used on the radar baseboard (sales Product no.: SP006062532). Chapter 2.8 summarizes the adaptions that need to be considered to port the project. For further information, refer to the AURIX™ TC336DA datasheet [2]
- **XENSIV™ BGT60ATR24AIP –** The XENSIV™ BGT60ATR24AIP is an automotive 60 GHz frequency-modulated continuous wave (FMCW) radar MMIC. This sensor uses antenna-in-package (AIP) technology for its two transmit (Tx) and four receive (Rx) antennas. An SPI interface is used for chirp configuration and raw data transmission to the AURIX™ MCU. Its integrated oscillator has the option to output the clock signal, which can be used as input clock by the AURIX™ microcontroller. For further information, refer to the BGT60ATR24AIP datasheet [3]
- **USB-UART interface –** To communicate with a PC, the board uses the FTDI FT23HL-REEL chip. This allows to transmit data at a low rate, e.g., radar point cloud or application results. Radar raw data can only be transmitted with a low frame rate configuration
- **CAN-FD –** An OPTIREG™ TLE9251 high-speed CAN FD transceiver is featured on the ICMS board. It allows users to prototype automotive applications, where CAN communication is needed
- **DAP Interface –** The board includes a Device Access Port (DAP) connector (X402,) which is a standardized, high-performance debug and tool interface

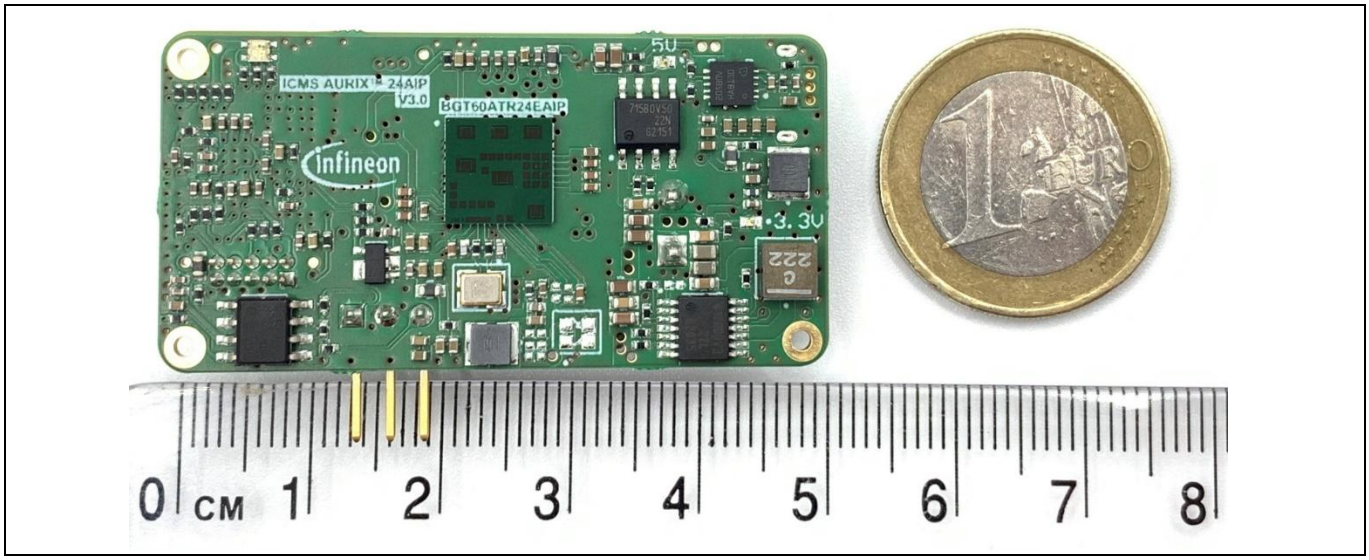

Figure 1 ICMS board v3.0

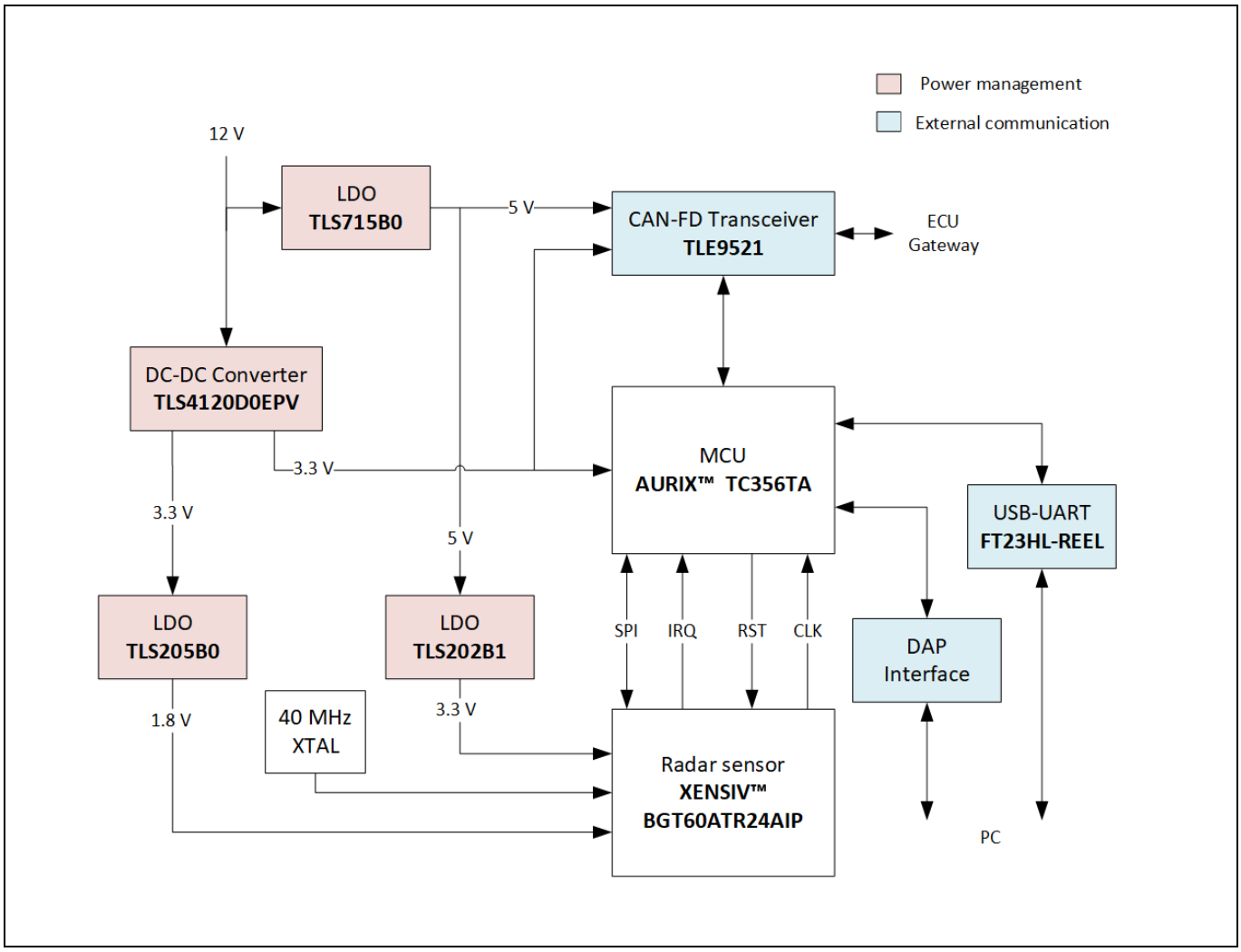

Figure 2 ICMS board v3.0 block diagram

2 Software components

2.1 Overview

The software described in this document is released as a code example in AURIX™ Development Studio as “iLLD_TC356TA_ICMS_BOARD_3_0_breathing_rate” and “iLLD_TC356TA_ICMS_BOARD_3_0_point_cloud.

Figure 3 gives an overview of the *core0_main()* function. The main objective of this function blocks is to establish SPI communication between the AURIX™ TC356 microcontroller and the BGT60ATR24EAIP, enable the BGT60ATR24EAIP’s 40 MHz clock output, and switch the TC356 clock to this external source as described in Chapter 2.3. After switching to the external clock source, the SPI module needs to be re-initialized, and then also the STM, DMA, and RMEM modules are initialized. Upon successful initialization of these hardware modules, CPU0 enters the radar state machine, which handles the main radar application flow. The radar state machine is depicted in Figure 4.

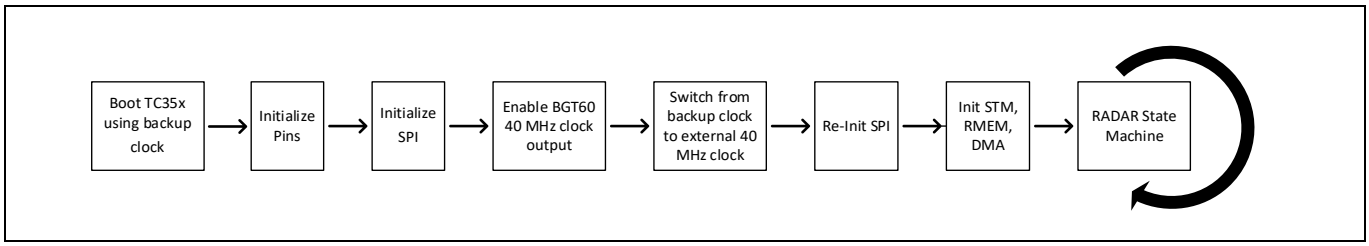

Figure 3 Application initialization code overview

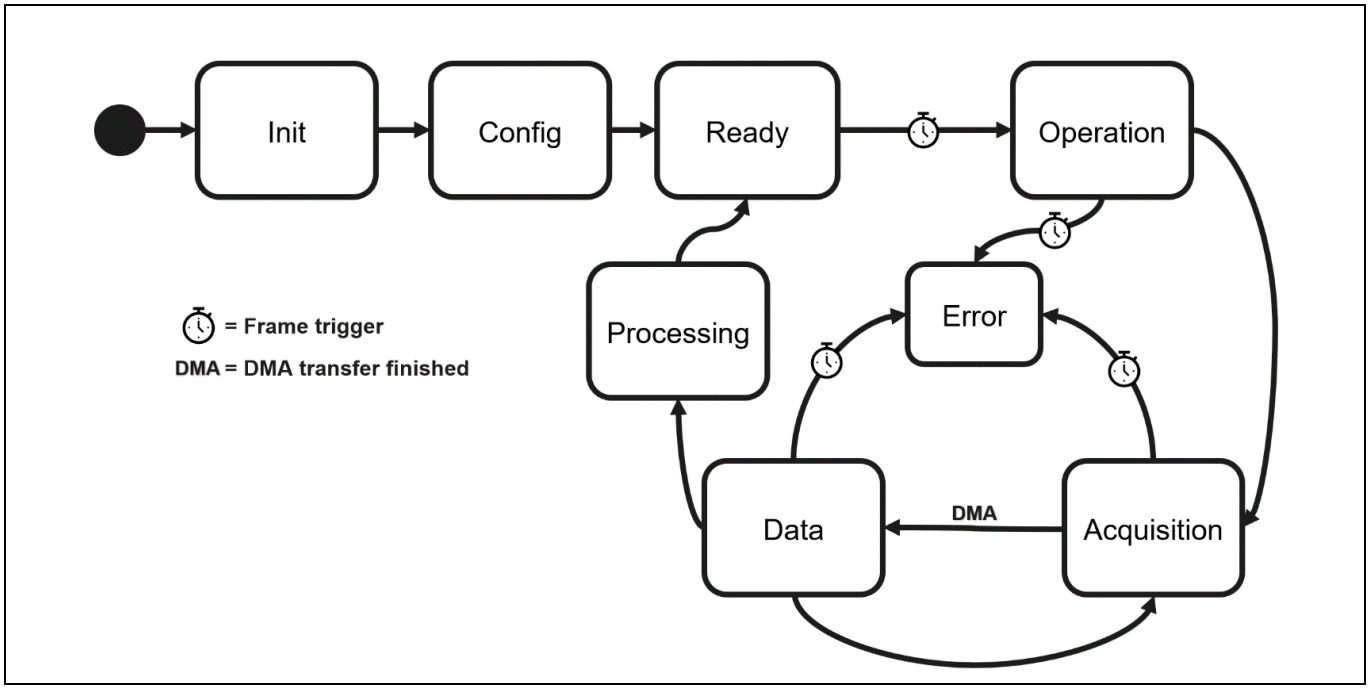

Figure 4 Radar state machine

While in the Config state, two major actions are performed: First, the configuration data for the BGT60ATR24EAIP is generated and transmitted to the BGT60ATR24EAIP via SPI. The configuration data defines the ramp scenario (number of ramps, samples per ramp, ramp shape, …) and other BGT60ATR24EAIP settings. Second, the Signal Processing Unit (SPU) is initialized. This step includes the basic SPU module initialization as well as the preparation of the linked list of SPU configurations which are executed during radar data processing. After successful completion of the *Config* state the state machine enters the Ready state.

The **Ready** state is a synchronization state. It is only left when a trigger from STM0 is received (“frame trigger”). The frame trigger is created with a period of 100 ms and marks the beginning of a radar frame.

Once the frame trigger is received, the **Operation** state is entered. In this state, the BGT60ATR24EAIP ramp generation is triggered via a SPI command. The state machine next enters the **Acquisition** state.

The **Acquisition** state is a wait state. While this state is active, the transfer of radar data from the BGT60ATR24EAIP to the TC356 is done. This transfer is done by the DMA module using the SPI interface as described in Chapter 2.2. Whenever data from a radar ramp has been transmitted, a DMA interrupt is triggered which then moves the state machine to the **Data** state.

In the **Data** state, the received radar data is processed. Each incoming ramp is pre-processed by the SPU and stored in the radar memory before the state machine returns to the **Acquisition** state. A ramp counter is used to track of the number of ramps in the current frame to allow the triggering of additional processing steps if enough data was received (e.g., after all ramps were received). As long as more data is expected in the current frame, the **Data** and **Acquisition** states will alternate. Only when all data has been received and processed, the **Data** state will exit back to the **Processing** state, where the radar state machine will wait for the next frame trigger.

In the **Processing** state, the SPU processing on each radar frame is performed and data from multiple radar frames is collected to build a micro-Doppler frame. After clutter removal (static target removel) Capon. beamforming is done to create a range angle matrix and to detect breathing targets in front of the ICMS board. For each of the detected targets, a breathing rate is calculated in the vital sign extraction processing steps. Only when all processing steps have been executed, the **Processing** state will exit back to the **Ready** state, where the radar state machine will wait for the next frame trigger.

The **Error** state is entered whenever a frame trigger is generated by STM0 while the state machine is not in the **Ready** state. This indicates that the 100 ms frame time was exceeded by the time necessary to acquire and process the radar data.

2.2 Transferring radar data from BGT60ATR24EAIP to TC356TA: SPI and DMA setup

The SPI interface is used to transfer the radar data from the XENSIV™ BGT60ATR24EAIP to the AURIX™ TC356TA microcontroller. The TC356TA acts as SPI bus master, the exchange of radar data is completely handled by the DMA module without any CPU intervention (non-radar data transmissions, e.g., BGT60ATR24EAIP register configurations or triggering the start of ramp scenarios, are still handled by the CPU). In general, the radar data is generated by the BGT60ATR24EAIP ADCs and written to a FIFO buffer, from where it is sent to the TC356TA via an SPI interface. On the TC356TA side, the DMA module is tasked with reading the radar data from the SPI input buffer and writing it to the radar memory (EMEM) (see Figure 5). Once in the EMEM the radar data can be accessed by the SPU.

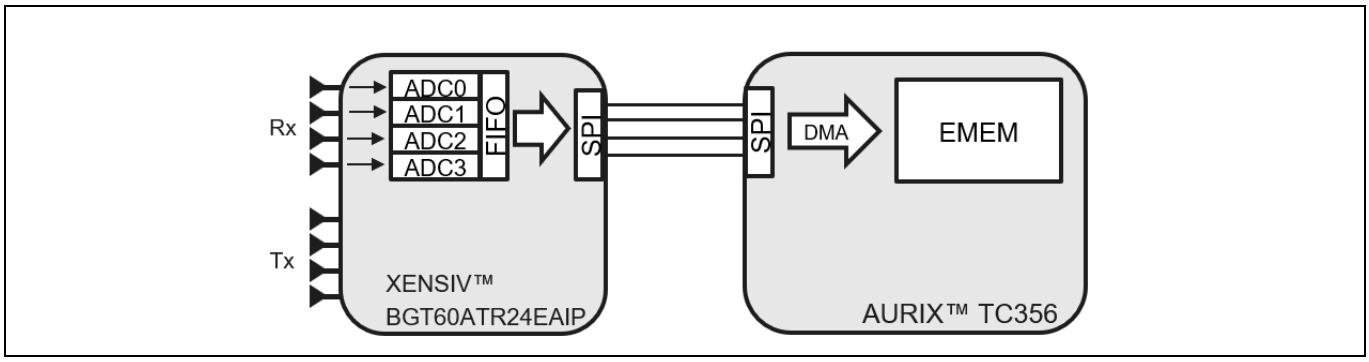

Figure 5 Radar data flow

A specific “burst transfer” mode is used to transfer the radar data via SPI. A burst transfer consists of a single 32-bit burst header packet followed by multiple 24-bit data blocks, which contain two 12-bit radar data packets each. Refer to the datasheet for the arrangement of the packets inside the data blocks. It depends on the number of activated ADCs which corresponds directly to the number of activated receive antennas. In this specific example, four receive antennas are used, leading to the data arrangement shown in Figure 6.

In the provided project, each burst transfer contains data from one complete ramp (all antennas).

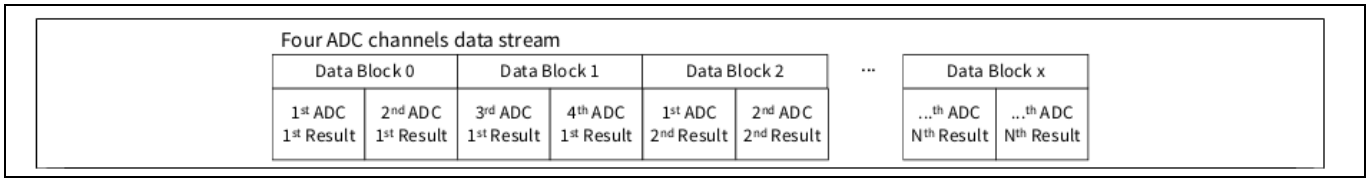

Figure 6 Burst Read data arrangement with four active ADCs

The SPI interface is set up as a full-duplex connection with a MOSI (master out, slave in), MISO (master in, slave out), and CLK (clock) line. Since the communication is full duplex, each bit transmitted by the master on the MOSI line is matched by a bit received simultaneously from the slave on the MISO line. Therefore, even if only read access is required, the master still needs to send dummy data on the MOSI line which will be discarded by the BGT60ATR24EAIP. In this example, dummy packets always have a value of 0x0.

Any SPI data transfer using the QSPI module requires the use of three registers on the AURIX™ microcontroller: DATA, BACON, and RXEXIT.

- The DATA register contains the data to be transmitted on the MOSI line
- Data received on the MISO line is written to RXEXIT
- The BACON register defines basic configuration settings for the SPI transfer (e.g., clock polarity, number of bits in transferred data word, …)

Additionally, after completion of a SPI burst transfer, the TXC bit in the FLAGSCLEAR register is set to clear the SPI transmit interrupt request flag.

Three DMA channels are used to service these registers during SPI burst radar data transfers. Each DMA channel uses one or several transaction control sets (TCS) for different stages of the communication. The TCSs can be daisy-chained allowing automatic sequential execution. Each TCS defines the following parameters:

- DMA transfer source and destination addresses

- Address increment per transfer and optional cyclical buffer size for source and destination
- Transfer size (e.g., 16 bit or 32 bit)
- Number of transfers per TCS
- TCS trigger and successor TCS (if applicable)

Figure 7 gives an overview of which DMA channel/TCS is responsible for servicing a specific register during the burst transfer of a ramp of radar data.

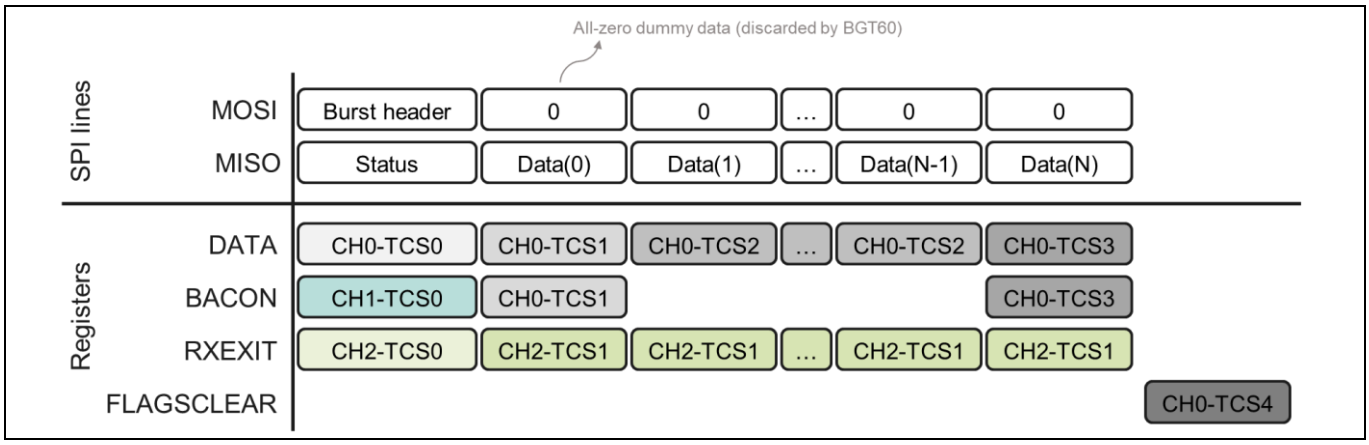

Figure 7 SPI registers to DMA channel/TCS mapping for burst read transfers

Table 1 gives a brief overview about the function of a specific DMA channel/TCS.

Table 1 Overview of DMA transaction control sets used for radar burst reads

DMA channel	TCS	Function of TCS	TCS trigger
0	0	Write burst header to DATA register	Triggered by software from BGT60 ISR
	1	Write dummy data (zero) to DATA register Write BACON register: Change SPI word length to 12 bits	Triggered by CH0-TCS0 done
	2	Write dummy data (zero) to DATA register N times (N = number of samples * number of ramps)-2	Triggered by CH0-TCS1 done
	3	Write dummy data (zero) to DATA register Write BACON register: Set LAST bit to indicate that the following packet is the last in the current frame	Triggered by CH0-TCS2 done
	4	Write 0x200 to FLAGSCLEAR register to clear SPI TX interrupt flag	Triggered by QSPI transmit interrupt
1	0	Write BACON configuring SPI for 32-bit word transfer	Triggered by software from BGT60 ISR
2	0	Read and discard 32-bit word from RXEXIT (This is the status information provided by BGT60ATR24EAIP on the MISO line during burst header transfer on MOSI)	Triggered by QSPI Receive interrupt
	1	Read 12-bit word from RXEXIT and write it to radar memory. Increment the destination address by 16 bits. (This aligns the 12-bit data received via SPI to 2-byte boundaries in the microcontroller memory)	Triggered by QSPI Receive interrupt

After completion of a burst transfer, ending with the execution of DMA Channel0-TCS4, one complete ramp of radar data has been transferred from the BGT60ATR24EAIP to the TC356TA radar memory (EMEM). This data block contains radar samples from all four receive antennas in an interleaved way, as shown in Figure 8.

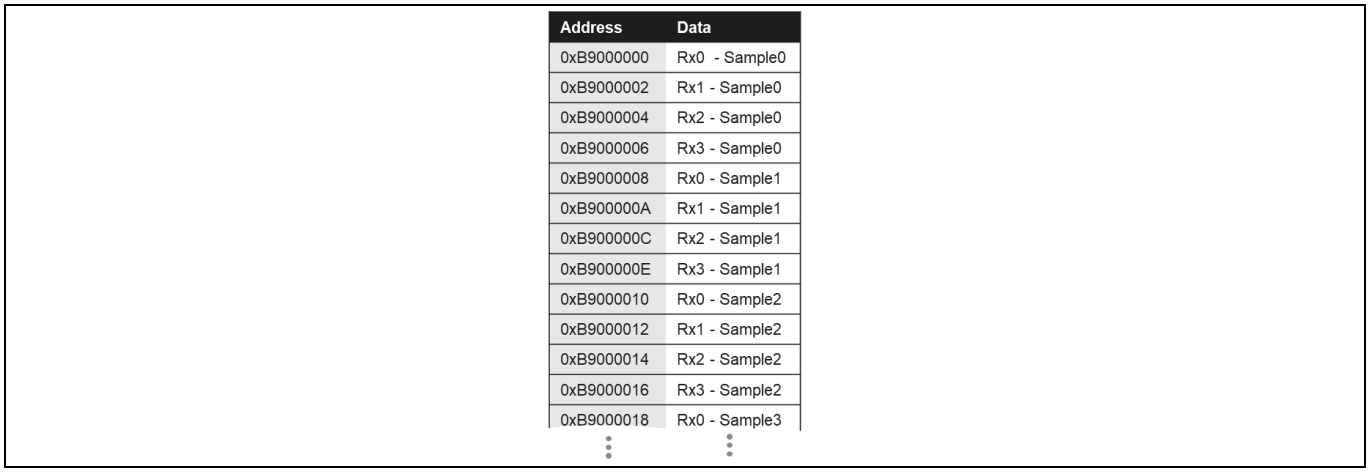

Figure 8 Interleaved data arrangement in EMEM after the DMA transfer

2.3 40 MHz clock supply

The AURIX™ TC356TA does not use a crystal oscillator as clock input. Instead, a 40 MHz clock signal is provided by the XENSIV™ BGT60ATR24EAIP. This 40 MHz clock must be enabled by sending a specific sequence of SPI commands to BGT60ATR24EAIP, which is described in [6].

Since the 40 MHz clock is not available when the AURIX™ is booted up, additional steps have to be taken to enable it:
First, the microcontroller boots using the built-in 100 MHz backup clock as clock source. After successful execution of the startup firmware, the QSPI module is initialized and the command sequence to enable the 40 MHz clock signal is sent to the BGT60ATR24EAIP via SPI.
Then, once the external clock is online, the AURIX™'s clock source is changed from the backup clock to the external 40 MHz clock and the internal PLLs are started, boosting the system clock frequency to a full 300 MHz.
Finally, the SPI module is re-initialized with respect to the new clock source and -frequency. The following code listing (an excerpt from the *core0_main( )* function) shows the functions relevant to the described process.

Code listing 1 Top-level functions for enabling and using 40MHz clock (Cpu0_main.c)

```
IfxPort_Init();      /* Init pins for LDO, MMIC Reset, Data IRQ, ... */
Qspi_Init(100e6);    /* Init QSPI module using backup clock */
Mmic_reset();        /* Hard-reset BGT60 via toggle of BGT60_RST_PIN*/
Mmic_disableSpiHighspeedMode(); /* set BGT60 SPI to HS mode (change clock edge)*/
Mmic_enable40MHzClock();        /* activate BGT60 40MHz clock output*/
IfxClock_switchTo40MHzClock();  /* Switch to 40MHz clock provided by BGT60 in CCU config PLL settings*/
Qspi_Init(200e6);    /* re-init QSPI with new clock settings */
```

Table 2 summarizes the steps performed by the respective functions.

Table 2 40 MHz external clock enabling functions

Function Name	Steps to configure MMIC oscillator as clock input for AURIX™
IfxPort_Init()	Configures the Reset Pin from MCU -> MMIC
Qspi_Init(100e6)	Enable QSPI module clock with 100 MHz QSPI module frequency, using backup clock as source Create SPI BACONs used for single and burst transfer modes Create command array utilized by DMA module during rada data transfer (contains data for DATA and BACON registers)
Mmic_reset()	Toggle BGT60ATR24EAIP reset pin
Mmic_disableSpiHighspeedMode()	Disables the SPI HighSpeed mode on the BGT60ATR24EAIP. (The maximum SPI clock speed for TC35 in full duplex configuration is 20 MHz, HighSpeed mode is only needed for > 25 MHz SPI communication)
Mmic_enable40MHzClock()	Enables the BGT60ATR24EAIP's 40 MHz clock output by configuring the BGT60ATR24EAIP's CLK_IN register as described in [6]
IfxClock_switchTo40MHzClock()	Reconfigure CCU to use 40 MHz clock Set input to external clock select Enable PLLs for 300 MHz clock generation
Qspi_Init(200e6)	Re-init QSPI with 200MHz module clock frequency, using PLL as source

2.4 Radar and MMIC configuration

2.4.1 Overview

The BGT60ATR24EAIP operates in Time Domain Multiplexing mode (TDM mode), where Tx1 and Tx2 alternatively transmit ramp signals, as Figure 9. Subsequent to data acquisition, radar signal processing is performed on AURIX™ Tricore™ CPUs and SPU.

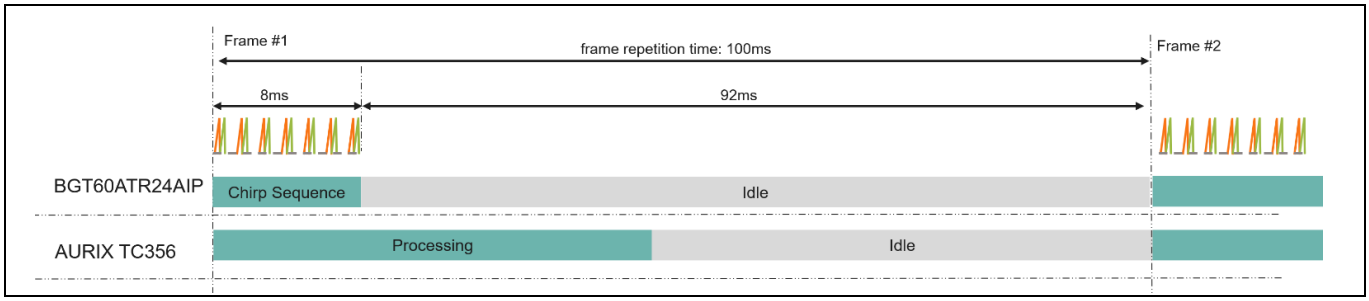

Figure 9 Radar configuration used with Tx1 and Tx2 sending ramps alternatively in TDM mode

The radar and MMIC parameters used are listed in Table 3, Table 4, and Table 5.

Table 3 Frame parameter overview

Parameter	Value
Number of Tx used	2
Number of Rx used	4
Number of samples per ramp	256
Number of ramps per Tx	8
Sampling frequency	2.22 MHz
Start frequency	58 GHz
Stop frequency	62 GHz
Valid sweep bandwidth	4 GHz
Range resolution	3.75 cm
Maximum range	4.76 m
Minimum range (first 8 range bins not evaluated)	0.3 m
Frame active time	1.89 ms
Frame repetition time	100 ms

Table 4 Azimuth power spectrum parameter overview

Parameter	Value
Number of angle bins used	32
Number of virtual receivers in azimuth used	4

Table 5 Micro-Doppler frame parameter overview

Parameter	Value
Number of frames used	100
Time span covered	6.4 s

2.4.2 Changing MMIC parameters using XENSIV™ Radar Fusion GUI

1. To adapt MMIC configuration parameters like bandwidth and time delays, download the XENSIV™ Radar Fusion GUI available from Infineon Developer Center [4]
2. Navigate to **Device** > **Dummy Device** and select a dummy device from the XENSIV™ Radar Fusion GUI menu, see Figure 10

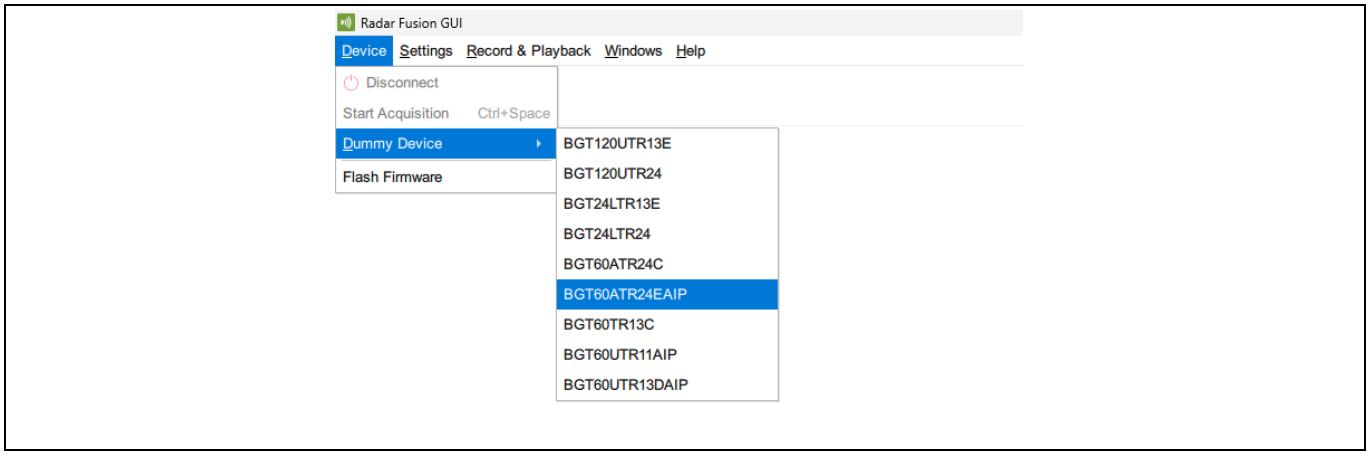

Figure 10 Device selection in GUI

3. Navigate to **File** > **Import Registers (.txt)** and **select ..\MMIC_Configs\active_config_icms3board.txt**
4. Adapt configuration as required

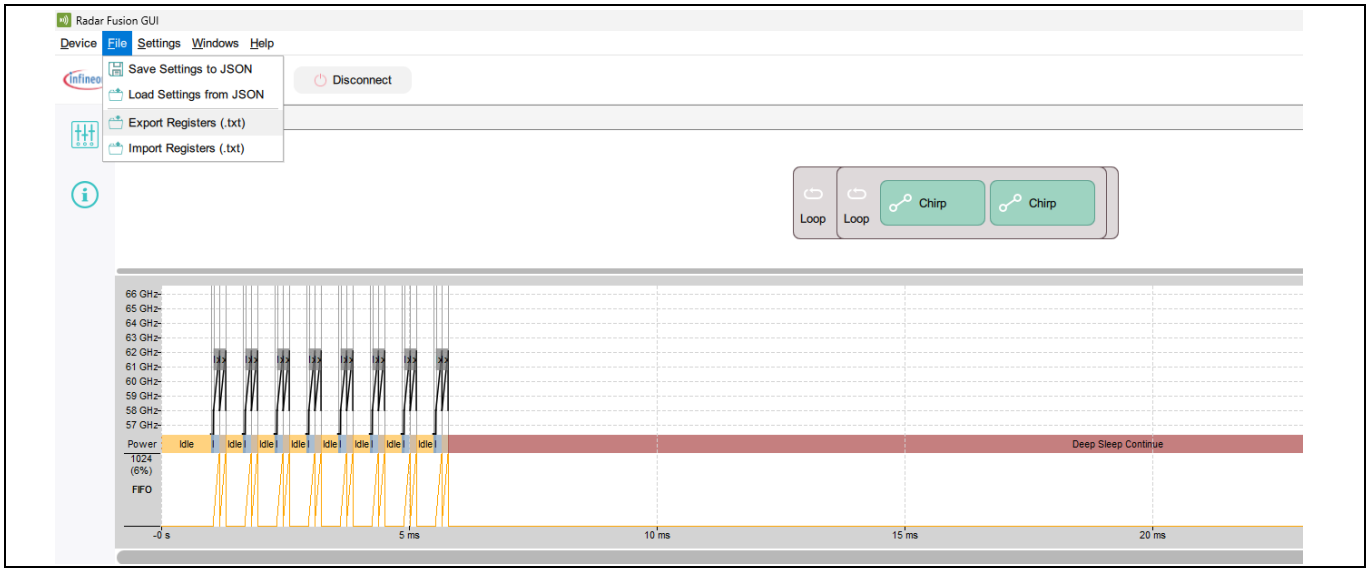

Figure 11 Exporting a MMIC configuration from the GUI

5. Navigate to **File** > **Export Registers (.txt)** to export the register
   The register settings then need to be copied over to the respective section of IfxBgt60ATR24E_init.c to be used in the c project

2.5 Radar signal processing

Following the initialization of the hardware components, the radar processing flow starts. Figure 12 provides a comprehensive overview of the signal processing steps executed on SPU and Tricore™.

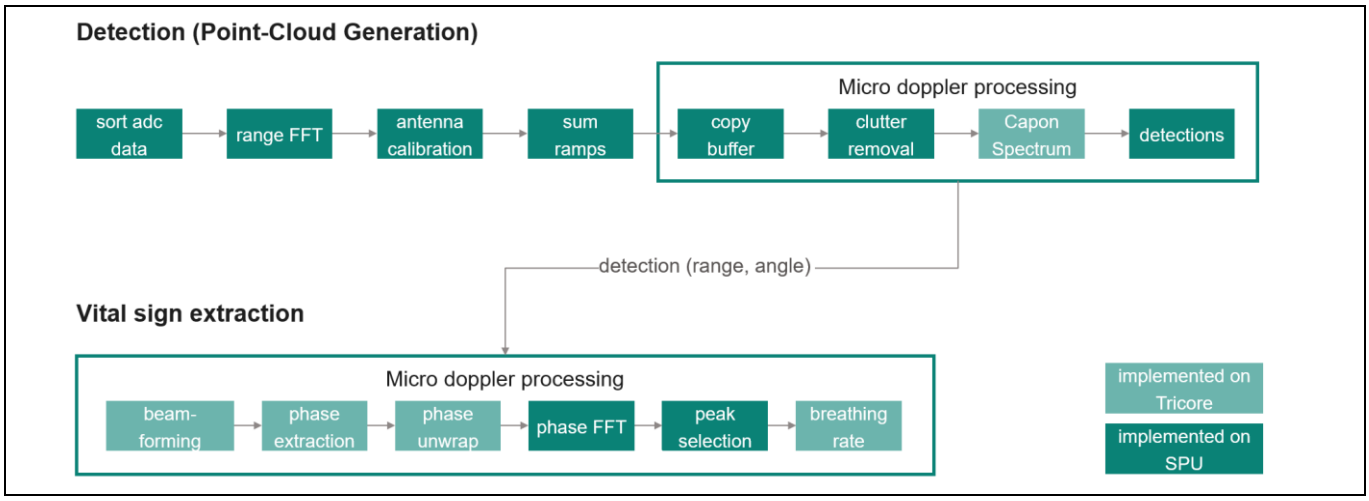

Figure 12 Processing steps from ADC data to creating detections based on a micro-Doppler frame and breathing rate calculation

On SPU, the desired processing flow is broken down into several "configurations". Each "configuration" represents a specific set of SPU functionality ", each representing a distinct set of SPU functionalities that are enabled during a single pass of data through the SPU. The configurations are stored in the SPU config RAM, and linked lists are generated that automatically execute these configurations in a predetermined sequences (see Table 7 and Table 8 for linked list examples). The SPU configurations are set up in *Spu0_cfg_init()*.

2.5.1 Point-Cloud generation

2.5.1.1 Signal processing overview

To detect breathing targets in front of the radar sensor, a series of signal processing steps are applied to each radar frame, and a micro-Doppler frame is constructed from multiple individual radar frames (Table 6). Subsequently, static targets (clutter) are are removed from this micro-Doppler frame (Table 7) and a range angle map is calculated using Capon beamforming. Target detection is then performed using a combination of Constant False Alarm Rate (CA-CFAR) processing, local maxumium detection and a comparison against a fixed threshold along range and angle dimensions (Table 8).

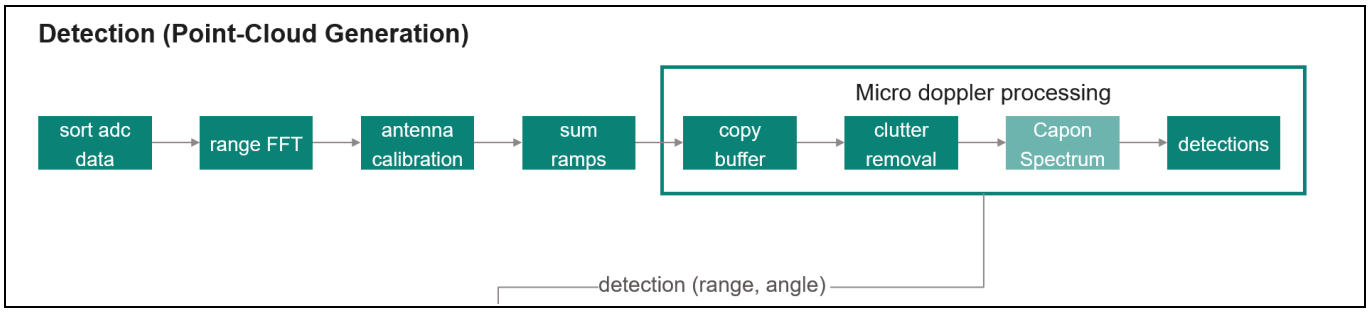

Figure 13 Processing steps from ADC data to creating detections based on a micro-Doppler frame

Table 6 SPU signal processing steps performed on each radar frame (create micro-Doppler frame)

Task description	SPU configuration
Apply range FFT along samples, windowing, calibration and reject symmetric half of range spectrum	IfxSpu_cfg2_rFFt.c
Copy micro-Doppler frame and write with address offset so that a circular buffer is set up	IfxSpu_cfg3_microDopplerBuffer.c
Calculate sum across ramps of radar frame and write result to micro-Doppler frame	IfxSpu_cfg4_sum_ramps.c

Table 7 SPU signal processing steps performed on micro-Doppler frame (static target removal)

Task description	SPU configuration
Calculate average across ramps in micro-Doppler frame	IfxSpu_cfg5_avg.c
Change format of average from 32 bit to 16 bit	IfxSpu_cfg6_avg32to16.c
Extend 16bit average result in ramp dimension	IfxSpu_cfg7_reshape.c
Subtract average from micro-Doppler frame	IfxSpu_cfg8_subinplace.c

Table 8 SPU signal processing steps performed on capon range angle map (detections)

Task description	SPU configuration
Perform CFAR along angle dimension	IfxSpu_cfg9_aCFAR
Perform CFAR along range dimension	IfxSpu_cfg10_rCFAR
Look for local maxima and apply absolute threshold along range dimension	IfxSpu_cfg11_rThreshLM
Look for local maxima and apply absolute threshold along angle dimension	IfxSpu_cfg12_aThreshLM

2.5.1.2 Sorting ADC data

Upon reception from the MMIC, the samples of a single ramp are stored in the adc_raw buffer within the radar memory, in the order shown in Figure 14 and Figure 15.

To facilitate efficient processing, the data is then reordered by the SPU (see configuration *IfxSpu_cfg1_sortadc.c* to store samples from a single receiver in consecutive memory locations. The resulting sorted data is stored in the adc_sort buffer within the radar memory, as depicted in Figure 15.

To accommodate the sorting process, the SPU configuration slot 0 (where *IfxSpu_cfg1_sortadc.c* is written to) is updated with a new output target address corresponding to the address offset of the current ramp with respect to the start of radar memory.

The sorting of a single radar frame is considered complete as soon as ADC data from all 16 ramps is sorted and written to radar memory.

Ramp1	Tx1	Sample 1	Rx1	Rx2	Rx3	Rx4
		Sample 2	Rx1	Rx2	Rx3	Rx4
		Sample 3	Rx1	Rx2	Rx3	Rx4
		....				
		Sample256	Rx1	Rx2	Rx3	Rx4

Figure 14 adc_raw MMIC samples of one ramp stored in radar memory

Ramp 1	Tx1	Rx1	Sample1	Sample2	...	Sample256
		Rx2	Sample1	Sample2	...	Sample256
		Rx3	Sample1	Sample2	...	Sample256
		Rx4	Sample1	Sample2	...	Sample256
Ramp 2	Tx2	Rx1	Sample1	Sample2	...	Sample256
		Rx2	Sample1	Sample2	...	Sample256
		Rx3	Sample1	Sample2	...	Sample256
		Rx4	Sample1	Sample2	...	Sample256
Ramp 3	Tx1	Rx1	Sample1	Sample2	...	Sample256
		Rx2	Sample1	Sample2	...	Sample256
		Rx3	Sample1	Sample2	...	Sample256
		Rx4	Sample1	Sample2	...	Sample256
...						

Figure 15 adc_sort MMIC samples of first three ramps stored in radar memory after sorting with SPU

2.5.1.3 Range FFT and antenna calibration

A range FFT is performed across samples by running SPU configuration *IfxSpu_cfg2_rFFt.c*. Hanning window and Tx – Rx-specific calibration constants (obtained from 0deg calibration) are combined by defining eight Tx – Rx-specific windows in CMEM, denoted as **calibWndw**. Each of these windows is calculated as the product of a standard Hanning window and the Tx – Rx calibration value. Subsequently, the symmetric half of the FFT output is discarded.

2.5.1.4 Building a micro-Doppler frame from multiple radar frames

To construct a circular micro-Doppler frame in radar memory, which is comprising data from 100 individual radar frames (compare sum ramps and copy buffer in Figure 12), two SPU configurations are implemented. The first configuration, *IfxSpu_cfg3_microDopplerBuffer.c*, is used to copy and shift the previous data of the micro-Doppler frame discarding the data from the oldest radar frame.

Next, the current radar frame data is written to the other end of the micro-Doppler frame using SPU configuration IfxSpu_cfg4_sum_ramps.c. To mitigate the effects of fast-moving targets, the coherent integration, e.g., sum across all ramps is calculated and written to the micro-Doppler buffer. The sequence of processing steps is shown in Figure 16.

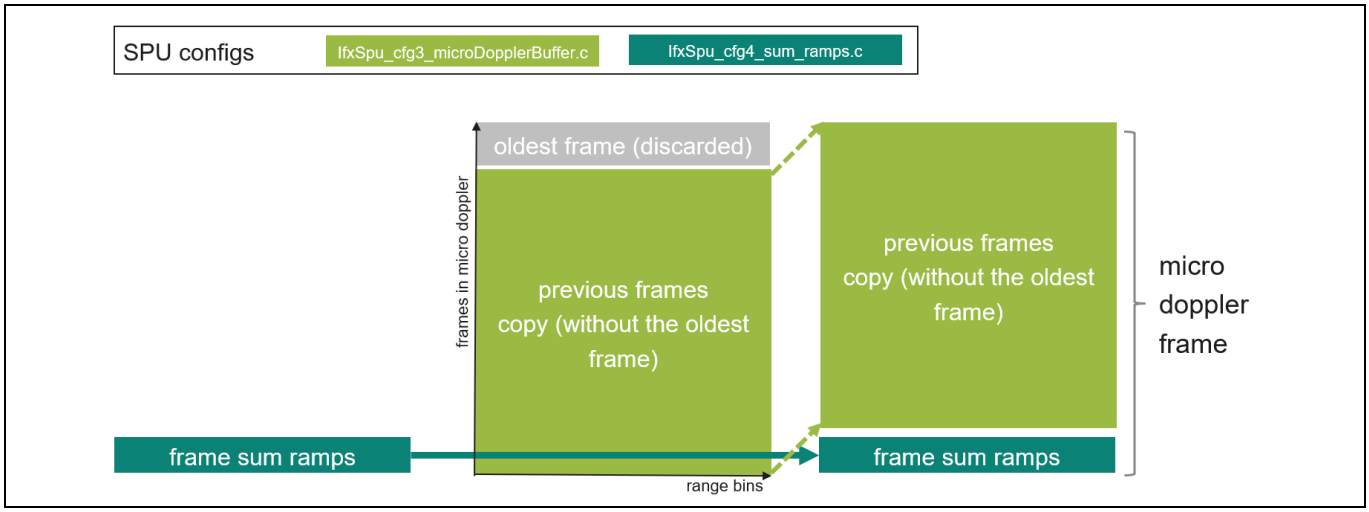

Figure 16 Building a circular micro-Doppler buffer in radar memory for subsequent detections and vital sign detections

2.5.1.5 Static target removal

To detect breathing persons in the micro-Doppler frame, it is essential to remove the signal originating from static targets.

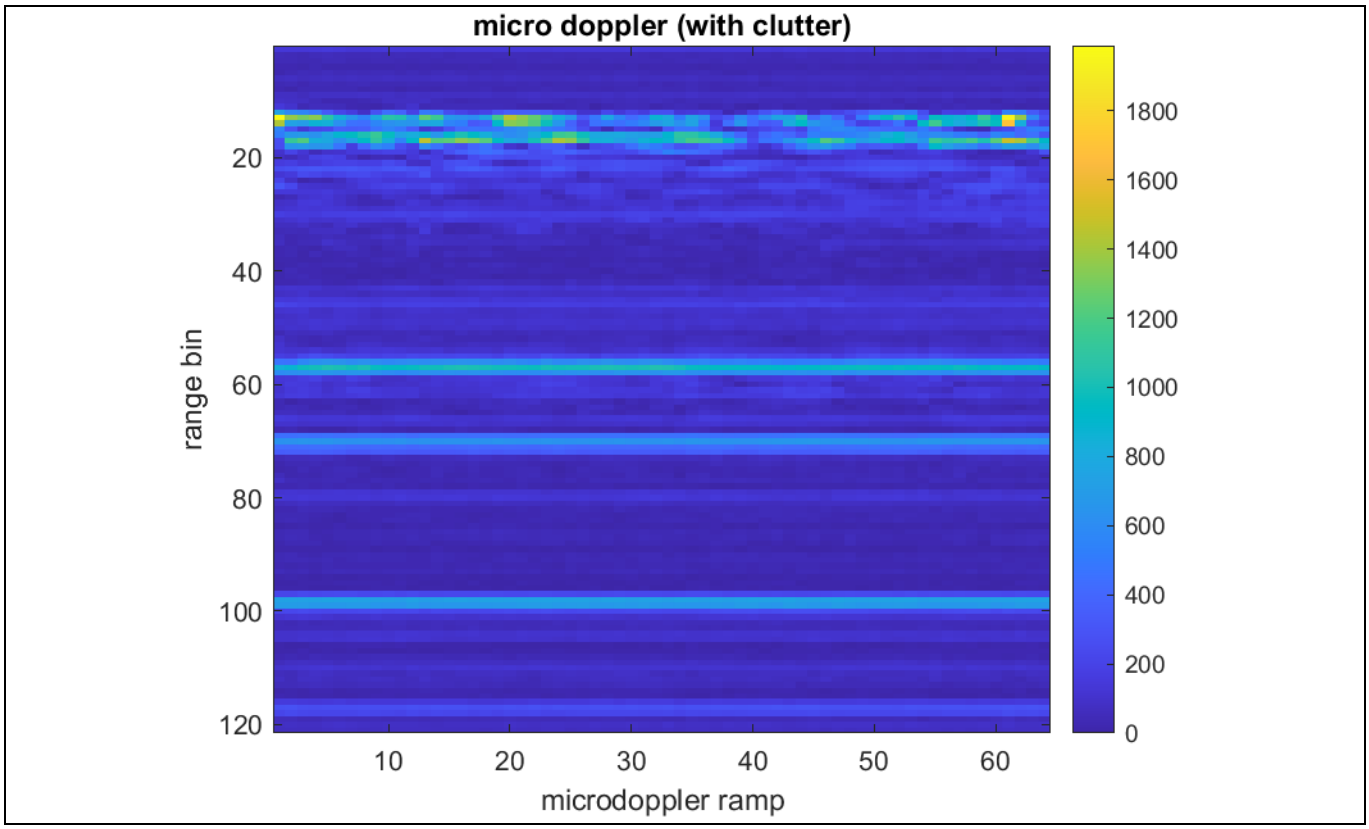

Figure 17 Micro-Doppler frame with a breating person in front of the sensor and static targets present as horizontal lines. The plot shows the aAbsolute value for one virtual antenna

The micro-Doppler frame prior to clutter removal (Figure 17) exhibits a large number of lines in range dimension (static targets). These lines are effectively removed in the micro-Doppler frame after static target removal (Figure 19).

The process of removing static targets (clutter) from the micro-Doppler cube involves two major steps:
First, the mean of the range FFT results across ramps is calculated for each range bin and each antenna.

In the second step, the calculated mean value is subtracted from the radar data cube for each ramp, as shown in Figure 18.

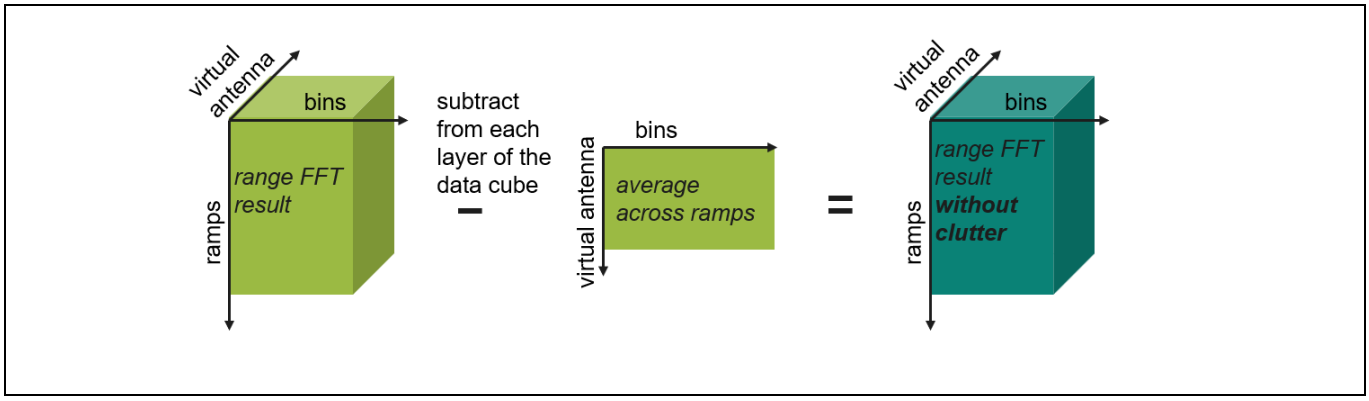

Figure 18 Static targets are removed from micro-Doppler data cube by first calculating the mean across ramps and then subtracting the result from the micro-Doppler data cube

The following SPU configurations are used to perform static target removal after every update of the micro-Doppler frame circular buffer (e.g., with every radar frame recorded):

IfxSpu_cfg5_avg.c

- Load micro-Doppler frame from radar memory (data is transposed during load to load it in the correct order for further processing)
- Apply 180 deg phase shift to data (to subtract via addition in cfg8)
- Calculate mean across ramps for each range bin and each virtual antenna
- Store average output to radar memory with 32-bit precision (average32)

IfxSpu_cfg6_avg32to16.c

- Load average data (32-bit precision) (average32)
- Store average output to EMEM with 16-bit precision (average16)

IfxSpu_cfg7_reshape.c

- Load average data (16-bit precision) (average16)
- Repeat for RADAR_NUM_MICRO_DOPPLER to extend average data cube to original data cube size
- Store average output to RADAR memory with 16-bit precision (average16)

IfxSpu_cfg8_subinplace.c

- Load average data (16-bit precision) (<-average16) and micro-Doppler frame (micro-Doppler)
- Add average data to range FFT data for each ramp (subtraction due to 180deg phase shift in cfg5)
- Repeat for RADAR_NUM_MICRO_DOPPLER
- Store clutter removal output to radar memory with 16-bit precision (micro-Doppler)

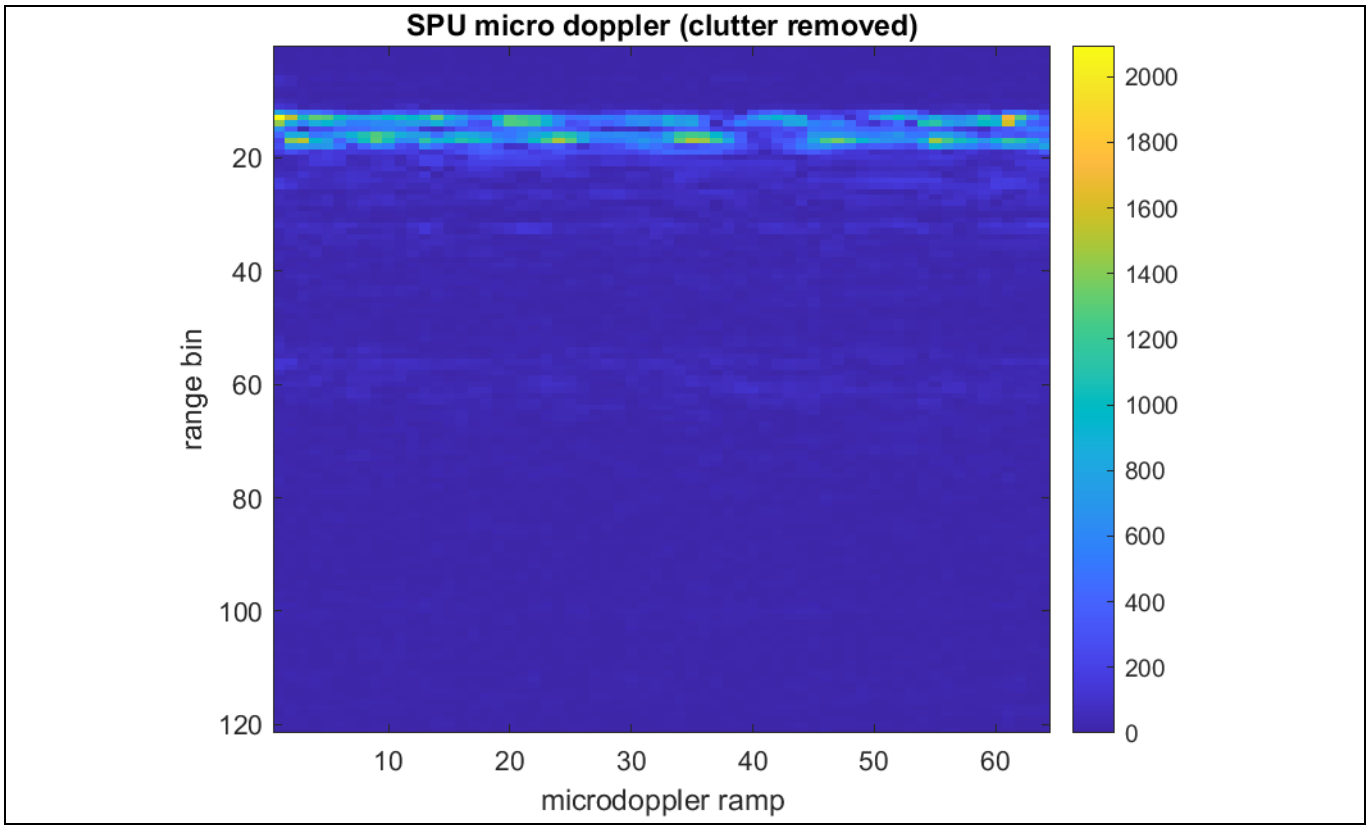

Figure 19 Micro-Doppler frame with a breating person in front of the sensor and static targets removed. The plot shows the absolute values for one virtual antenna

2.5.1.6 Capon spectrum: Creating a range angle map

The range angle map is calculated using Capon beamforming applied on the micro-Doppler frame after static target removal. Capon beamforming is selected to increase angular resolution as compared to using an angle FFT. The calculation is performed after every update of the micro-Doppler circular buffer, e.g., when a new radar frame is recorded. The Capon spectrum P_{nz} is calculated from the azimuth steering vector $a(\theta_z)$ and the inverse of the spatial covariance matrix ${R_n}^{-1}$:

$$P_{nz} = \frac{1}{a(\theta_z)^H {R_n}^{-1} a(\theta_z)}$$

Equation 1

Where, n is the range bin and z is the azimuth angle bin. Capon beamforming was selected instead of a standard Angle FFT as it provides higher spatial resolution.

The following steps are implemented on Tricore™ in *SCM_generator* to calculate the range-angle map:

1. Extracting the virtual azimuth antenna array from a micro-Doppler frame (SCM_generator)

Figure 20 shows the antenna layout and data structure of BGT60ATR24AIP used in TDM MIMO mode. The virtual antenna array in azimuth is created from Tx1Rx3, Tx2Rx1, Tx2Rx2, Tx2Rx4 marked with an ellipse. Note that an extension to using all receivers and to an evaluation of elevation is possible. For the sake of simplicity this example is focused on evaluating azimuth using only four virtual antennas.

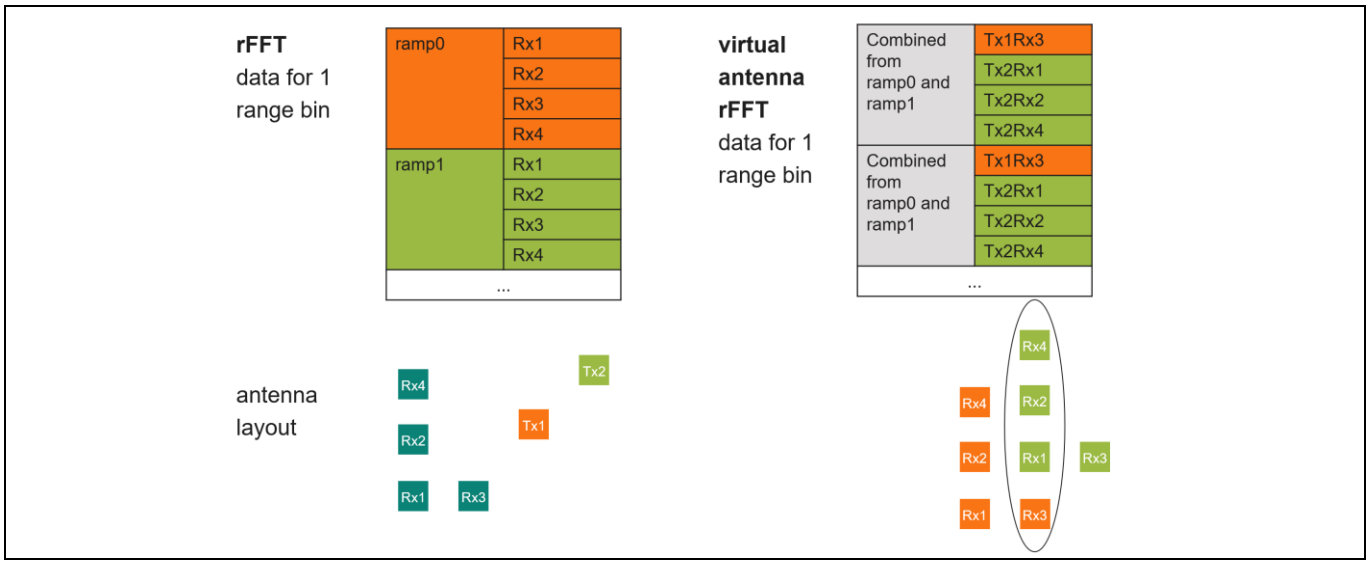

Figure 20 Creating a virtual antenna array in azimuth direction from 2Tx and 4Rx used in time domain multiplex (TDM) mode

2. Generating the spatial covariance matrix (*SCM_generator*) for each range bin

The data from this virtual antenna array is used as input data for the (estimation of) spatial covariance matrix calculation:

$$R_n = X_n * X_n^T,$$

Equation 2

Where, n is the range bin and X_n is a vector of virtual antenna values recorded for this range bin for the first ramp in the micro-Doppler frame. The result is a 4x4 estimation of the covariance matrix for each range bin stored in *Rn*.

3. Applying spatial smoothing and forward backward filtering on spatial covariance matrix(*SCM_generator*)

To improve the final range angle spectrum result two filtering steps are performed on the spatial covariance matrix prior to inversion: Spatial smoothing and forward-backward filtering. Where the forward backward filtered covariance matrix is calculated as:

$$R_n{}^{fb} = \frac{1}{2}(R_n + R_{back})$$

Equation 3

Where, R_{back} is R_n rotated by 180° and matrix dimension reduced to 3x3.

Spatial smoothing calculates the sum across subarrays in the matrix and reduces the size of the covariance matrix by 1 providing a 3x3 covariance result $R_n{}^{SS}$ as depicted in Figure 21.

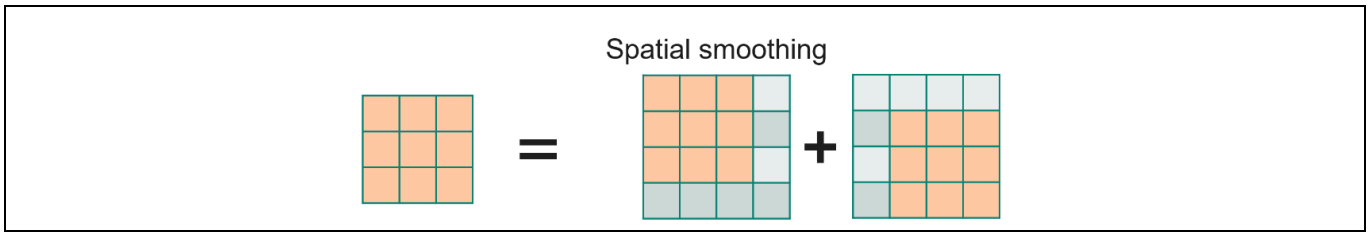

Figure 21 Principle of spatial smoothing depicted for a 4x4 covariance Matrix

Forward backward filtering and spatial smoothing are combined to on filtered covariance matrix result ${R_n}^{ssfb}$:

$$ {R_n}^{ssfb} = \frac{1}{2}({R_n}^{ss} + {R_n}^{fb}) $$

Equation 4

The filtered covariance matrix is calculated for each range bin a stored in *Rn_filtered*.

To ensure numerical stability in the subsequent inversion step a regularization term on the order of machine precision (+10*FLT_EPSILON) is added to the filtered result.

4. Generating the azimuth steering vector (*ASM_generator*) for the desired angular range

The azimuth steering matrix is generated using the following equation:

$$ A(\theta) = (a(\theta_1), a(\theta_2) \ldots\ldots a(\theta_m)), $$

Equation 5

Where, m is the number of angular bins used (32 in this example) and each steering vector is calculated via

$$ a(\theta_z) = \left[1,\ e^{j\mu_a},\ \ldots\ldots, e^{j(N_r-1)\mu_a}\right]^T, $$

Equation 6

Where, $\mu_a = 2\pi\frac{d}{\lambda}\sin(\theta_z)$ and d being distance between two virtual antennae and λ being the wavelength used.

5. Inverting the filtered spatial covariance matrix (*ComplexMatrixInversion*)

To invert the complex valued 3x3 spatial covariance matrix, real and imaginary parts of the matrix are written to a 6x6 real valued matrix as described in [7]. This real valued 6x6 matrix is inverted using AURIX™ DSPlib [5]. The real and imaginary part of inverted 3x3 can then be extracted from the 6x6 inversion result.

6. Calculating the capon range angle spectrum (*Power_Spectrum*)

The capon power spectrum is calculated using:

$$P_{nz} = \frac{1}{a(\theta_z)^H R_n^{-1} a(\theta_z)}$$

Equation 7

Where the Hermitian of $a(\theta_z)^H$ and the matrix products are calculated using DSPlib functions *Ifx_mtxHrmComplexF32* and *Ifx_mtxMulComplexF32* respectively.

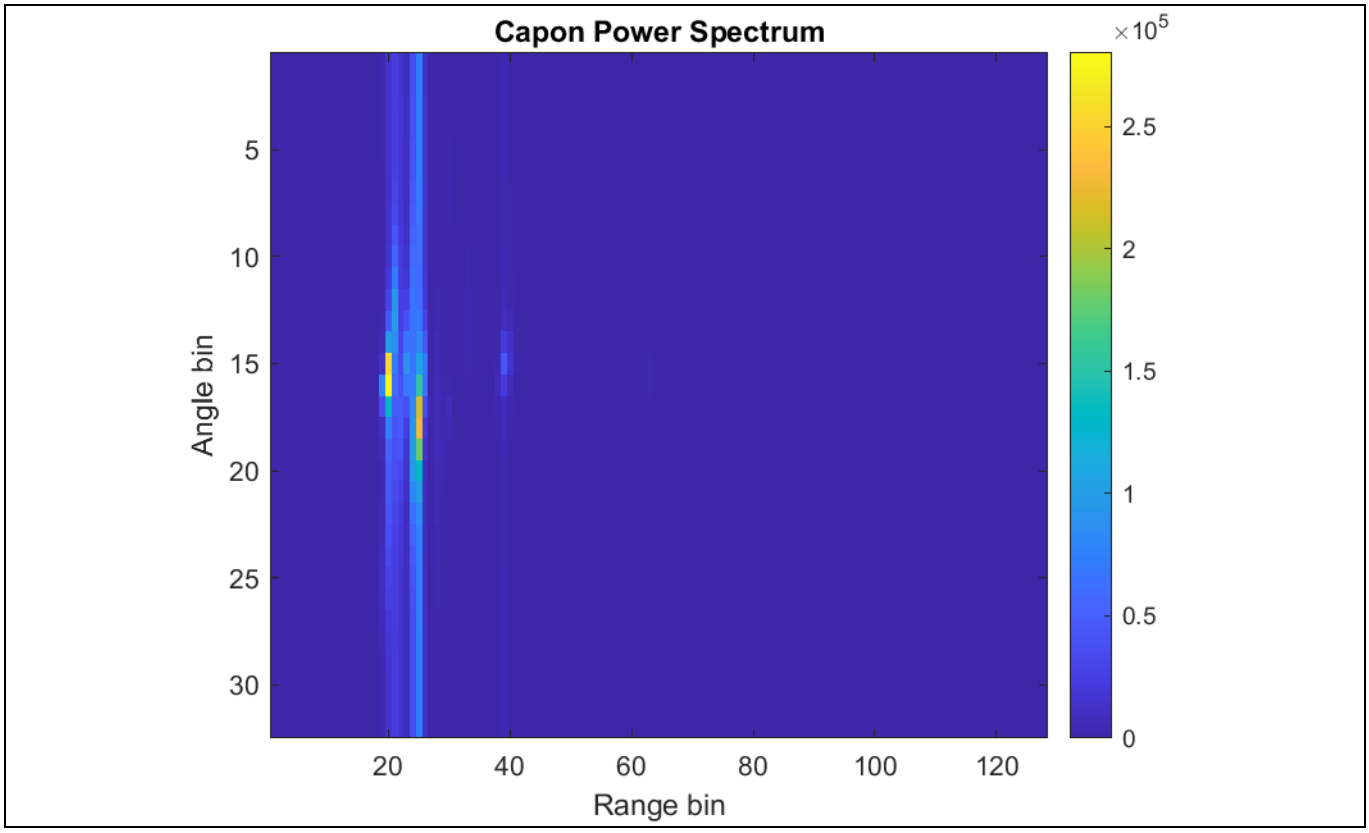

Figure 22 Capon range angle spectrum showing a single target (still sitting breathing person) in front of the sensor

2.5.1.7 Detect a target on the range angle map

To extract the range bin and angle bin of targets present in the range-angle map, the detection mechanisms offered by the SPU are utilized. The following SPU configurations are run to perform 2D CFAR and apply local maximum search in two dimensions. The SPU configurations used to implement these steps are summarized in Table 8.

The resulting bitmaps are then combined (*IfxDetectionMapCombine*) and transposed (*IfxDetectionMapTranspose*) to calculate the final detection bitmap DetMap. This bitmap is then processed to check for detections and write detection range and angle bins to det_targets variable.

2.5.2 Vital signs: Breathing rate detection

Following the detection of targets described in the previous chapter (Point-Cloud generation), the breathing rate of detected targets is calculated using the processing steps depicted in Figure 23. The targets range and angle bin and the micro-Doppler frame are used as input data to the vital sign processing block.

The breathing of a target causes chest displacements that are detected by the radar, even if they are inside of a range bin. The movement modulates the phase of the received signal. The breathing rate can be calculated by extracting the phase of the signal and performing an FFT.

2.5.2.1 Signal processing overview

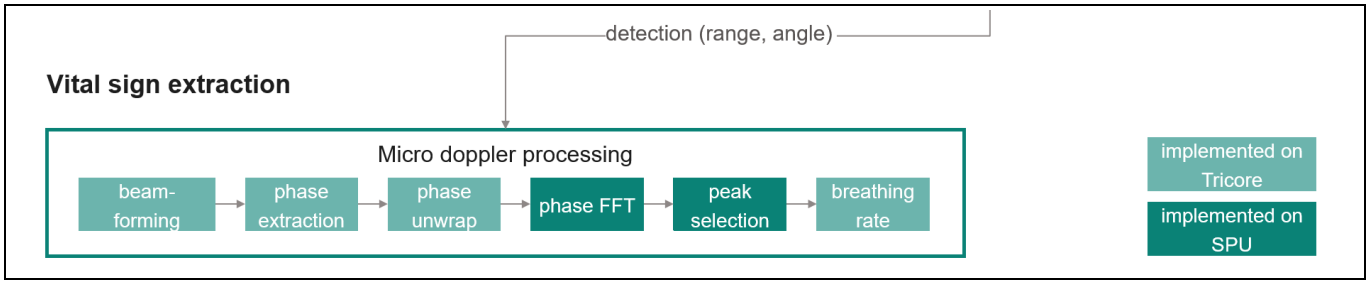

Figure 23 Processing steps from detections to calculating a breathing rate

2.5.2.2 Beamforming

In order to combine the signal from all virtual antennas and to enhance the signal coming from the direction of the target detected, a beamforming step is performed prior to phase extracted. The micro-Doppler data for one range bin is multiplied with a steering vector in the **beamforming** function.

2.5.2.3 Phase extraction and phase unwrapping

The complex values of the beamformed micro-Doppler frames are extracted and converted from floating point to 16-bit precision integer using the conversion function DSPlib *Ifx_convertRealF32Q15.* They are then used in the DSPlib function *Ifx_atan2Q15* to calculate the phase of every selected micro-Doppler frame. The phase values vector $\varphi_{n,}$ is stored in *phaseData*.

Phase unwrapping is necessary to remove jumps caused by the *Ifx_atan2Q15* function. Results prior to and after phase unwrapping are depicted in Figure 24.

The following algorithm is implemented and performed on every phase vector φ_n

1. Calculating the difference between every element and the next:

$$\Delta = \varphi_{n,i+1} - \varphi_{n,i},$$

Equation 8

Where, n is the range bin and i the corresponding micro-Doppler frame.

2. Comparing the absolute value of the difference and a set decision threshold
3. Deciding if it is necessary to add or subtract 2π from every element after the current position

The decision threshold and π have to be scaled with 2^{13} to match the format of phaseData.

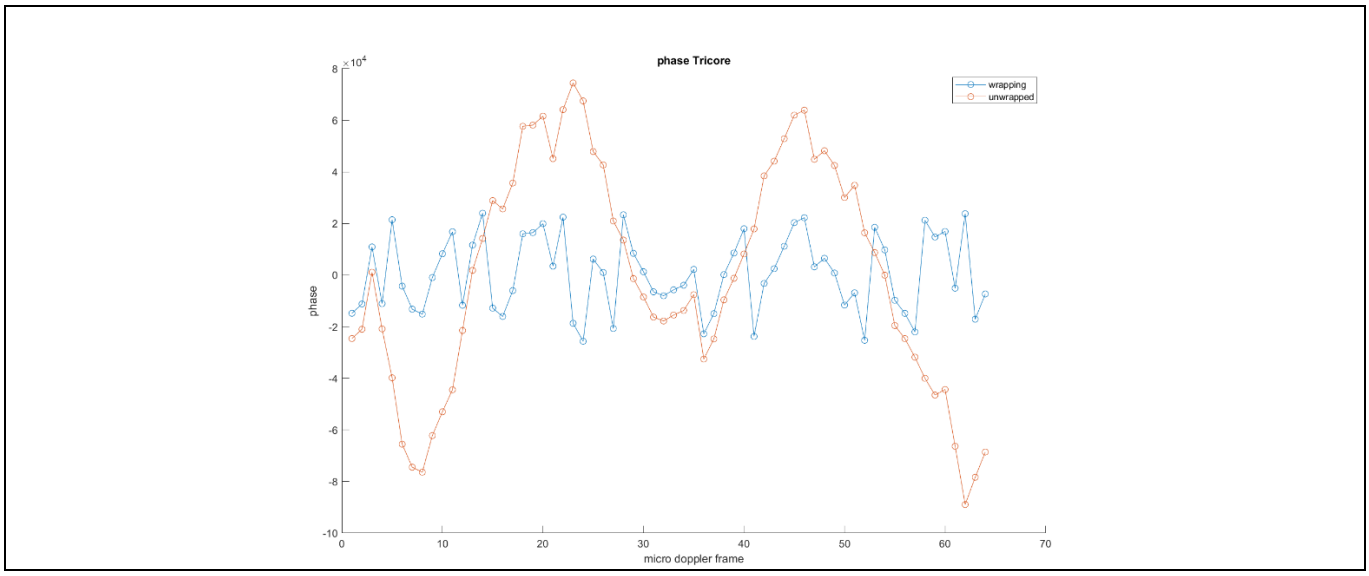

Figure 24 Phase prior to and after phase unwrapping

2.5.2.4 Mean removal

Prior to the phase FFT, the mean of every phaseData vector is calculated and subtracted from every element in the vector. This removes the DC offset of the signal.

2.5.2.5 Phase FFT

A phase FFT is performed across every phase vector by running SPU configuration *IfxSpu_cfg13_pFFT.c* to obtain the frequency of the chest displacements. The input vector is automatically padded with zeros to get the desired FFT length of 1024. The symmetric half of the FFT is rejected.

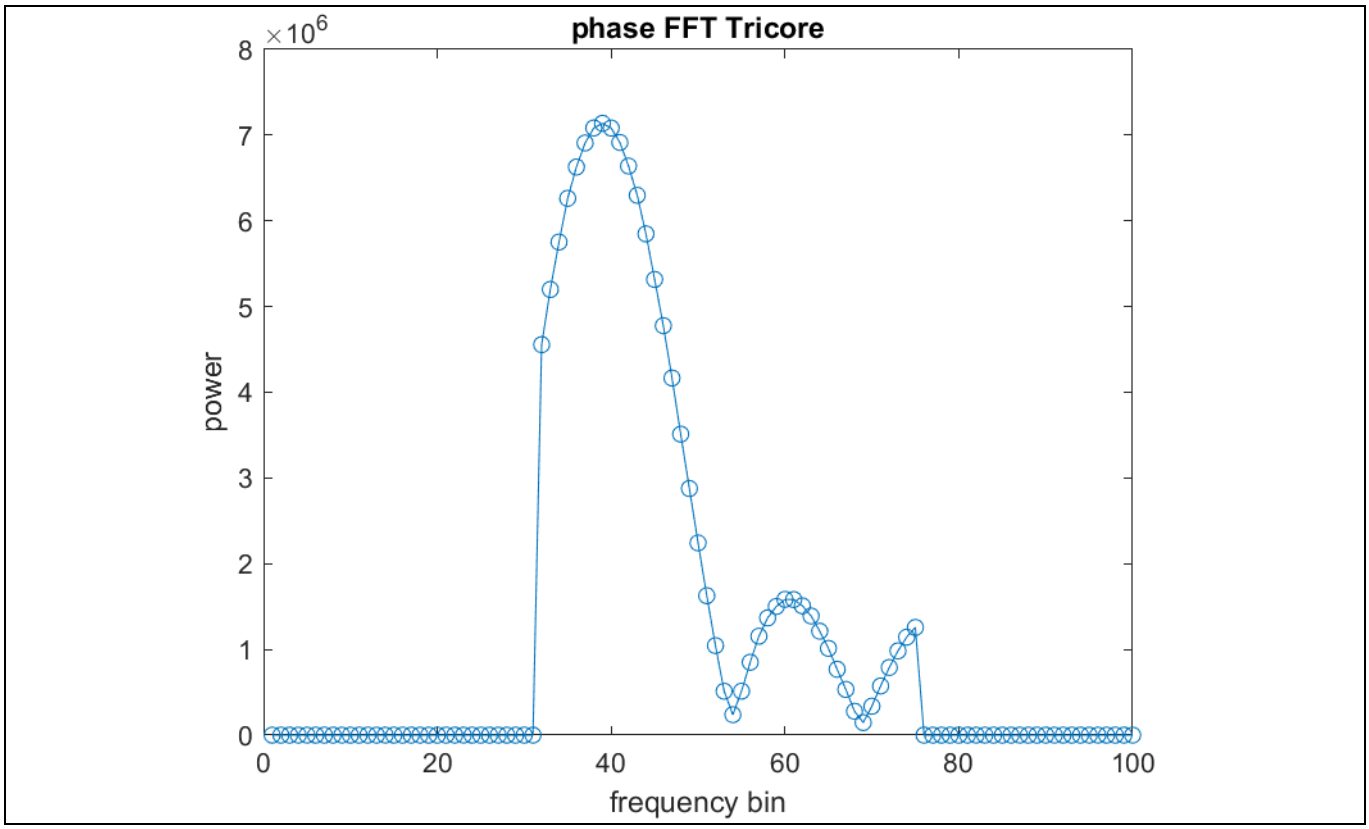

Figure 25 Result of phase FFT with FFT length 1024 (only first 110 frequency bins are plotted). The maximum peak within the frequency filter is detected with SPU statistics unit

2.5.2.6 Spectrum filtering, peak selection, and breathing rate detection

To select the correct frequency peak, the spectrum is filtered by applying a window function on SPU (*IfxSpu_cfg14_stats.c*). As a result, frequency bins outside of the filter window are set to zero as displayed in Figure 25.

Peak selection is done on the SPU with sideband statistics. The index where the global maximum is located is stored and used in the following formula to calculate breathing rate.

$$B_n = 60(I_n + 1)f_s l^{-1},$$

Equation 9

Where, I_n is the index of the global maximum of the FFT for the range bin n, f_S is the micro-Doppler frame sampling frequency and l is the length of the phase FFT performed on the SPU. To get the desired unit (BPM: breaths per minute) the result is multiplied by 60.

Table 9 shows an overview of the time required to execute the processing blocks on CPU and SPU. Note that this can further be optimized. Currently, the power spectrum and detections are performed after every radar frame collection, e.g., every 100 ms. One could change this to performing detections only every few seconds.

The breathing rate of a target is calculated once the micro-Doppler buffer is available. Thus, the breathing rate of a target in front of the sensor is available within 64 * 100 ms = 6.4s.

Table 9 Time required to execute the processing blocks on AURIX™ CPUs and SPU.

Processing block	Time [ms]	Comment
Data acquisition and sorting	8.00	256 samples, 16 ramps every 50 ms
Frame processing	0.32	Range FFT, calibration, coherent integration across ramps, update circular buffer
Micro-Doppler processing	1.92	100 frames, Static target removal
Power spectrum calculation	23.17	120 range bins, 32 angle bins, Capon beamforming, spatial smoothing and forward – backward filtering
Detections	0.69	2D CA-CFAR, local maximum and fixed therashold
Beamforming	0.58	–
Breathing rate extraction	0.40	Phase extraction, Phase unwrapping, mean removal, phase FFT
Free processing budget (assuming 100 ms frame repetition rate)	65.9	–

2.6 Memory overview

Table 10 lists all variables in radar memory and the memory space occupied. Note that of the available 2 MB of extension or radar memory (EMEM) available in TC356TA only 1 MB is used in this example.

Table 10 Radar memory overview

Name	Number of elements	Data type	Size [kBytes]
adc_raw	1024	sint16	2.048
adc_sort	16384	sint16	32.768
rmapcalibrated	8192	csint16	32.768
microdopplerclutterremoved	102400	csint16	262.144
average16	102400	csint16	262.144
average32even	1024	csint32	8.192
Rn	2048	csint32	16.384
Rn_filtered	1152	csint32	9.216
Rnn	1152	cfloat32	9.216
PS	4096	sint32	16.384
CFARAMap	128	uint32	0.512
CFARLMRMap	128	uint32	0.512
CFARRMap	128	uint32	0.512
CFARLMAMap	128	uint32	0.512
Amap_combined	128	uint32	0.512
Rmap_combined	128	uint32	0.512
Rmap_combined_transpose	128	uint32	0.512
DetMap	128	uint32	0.512
phaseData	320	sint16	0.640
PHI	320	sint32	1.280
rr_sectrum	2560	csint32	20.480
rr_sectrum_filtered	2560	csint32	20.480
microdopplerTargets	2560	cfloat32	20.480
microdopplerBeamformed	320	cfloat32	2.560
Total			983.424

2.7 Debugging

All processing results are written to the global variable g_rdr. This can be used to aceess the final target list (*det_targets)* as well as intermediate results like the capon power spectrum (*PS)*.

2.8 Adaptions required for using 60 GHz radar baseboard (TC336DA)

The software and processing steps can easily be ported to the radar baseboard with AURIX™ TC336DA (sales Product no.: SP006062532). As the memory consumed fits within the 1 MB of radar memory offered by AURIX™ TC336DA, the radar configuration and processing steps do not need to be adapted.

Note that with the 60 GHz radar baseboard using AURIX™ TC336DA the following adaptions are required:

- External clock: there is no need to configure clock output on BGT60ATR24AIP and external clock input on AURIX™ as an external oscillator is available on the board which is directly connected to AURIX™
- Account for different pin connections in IfxPort_init.c and IfxPort_init.h
- Check SPI clock setting and SPI pins in IfxQspi_init.c and IfxQspi_init.h

3 Related resources

- 32-bit TriCore™ AURIX™ – TC3xx
- TC33x/TC32x 32-Bit Single-Chip microcontroller datasheet
- Developer Community
- AURIX™ Digital collaterals
- Design resources

References

[1] Infineon Technologies AG: TC35x datasheet; Available online

[2] Infineon Technologies AG: TC33xEXT datasheet; Available online

[3] Infineon Technologies AG: BGT60ATR24AIP MMIC datasheet

[4] Infineon Technologies AG: Infineon Developer Center; Available online

[5] Infineon Technologies AG: Getting started with AURIX DSPlib AP32528; Available online

[6] Infineon Technologies AG: BGT60ATR24AIP clock configuration

[7] IEEE Xplore: A note on the inversion of complex matrices; Available online

Revision history

Document revision	Date	Description of changes
V 1.0	2025-08-15	Initial release
V 1.1	2025-09-16	Change of document classification

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2025-09-16

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2025 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

AN0047

Important notice

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.