DEMO POSITION2GO Software Guide

XENSIV™ 24 GHz radar

Board version V1.2

About this document

Scope and purpose

This user manual describes the firmware, supported algorithms and the development recommendations required to build an application around Infineon's DEMO POSITION2GO board.

It describes the demonstration applications of the Position2Go radar demo board based on the XENSIV™ 24 GHz BGT24MTR12 radar MMIC with details of package architecture and contents.

This document provides guidelines for novice users on how to build and run Position2Go radar applications such as outdoor ranging and target position estimation to support ease-of-use and faster-to-market integration.

Intended audience

This document is intended for users of the Position2Go demo board who want to get started with Infineon's Position2Go firmware solution, test several sensing demonstrations, and implement custom radar applications in the 24 GHz industrial, scientific and medical (ISM) band.

Table of contents

List of figures

List of tables

1 Introduction

The DEMO POSITION2GO board is a demonstration platform for Infineon's silicon-germanium (SiGe) based 24 GHz radar chipset, equipped with the 24 GHz transceiver chipset BGT24MTR12 and the 32-bit Arm® Cortex®-M4 based XMC4700 microcontroller.

The Position2Go demo board provides a complete evaluation platform for a radar system including demonstration firmware and a highly interactive Graphical User Interface (GUI).

The board is designed to evaluate the capabilities of the BGT24MTR12 MMIC, comprising one transmit and two receive channels with the XMC4700 microcontroller utilizing Infineon's powerful, free-of-charge toolchain DAVE™ for microcontroller programming.

It provides a complete evaluation platform for a radar system including demonstration firmware and a highly interactive Graphical User Interface (GUI).

Note: This Position2Go board does not serve as a reference design. The hardware and software provided with this board are for demonstration purposes only.

2 Basic radar concepts

Below, some basic radar concepts mentioned in this document are explained.

2.1 About Doppler radar

Doppler radar operates on the principle of sending a beam of electromagnetic radiation waves, tuned to a precise frequency, toward a moving object. When the electromagnetic radiation wave hits the moving object, it "bounces" back toward the source, which also contains a receiver. However, since the wave is reflected off a moving object, the wave is shifted as outlined by the Doppler effect.

The wave that is coming back toward the radar is treated as an entirely new wave, as if it were emitted by the target it bounced off. The target is acting like a new source for this new wave. When it is received at the radar, this wave has a frequency different from the frequency that was originally sent toward the target.

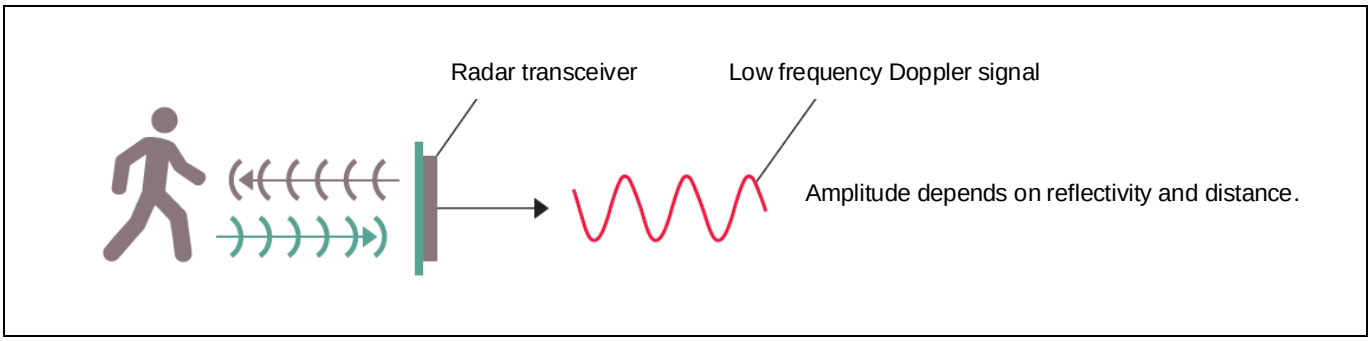

Figure 1 Doppler effect

Since the electromagnetic radiation was at a precise frequency when sent out and is at a new frequency on its return, this can be used to calculate the velocity v of the target.

The Doppler effect shifts the received frequency up or down based on the radial velocity of the target (closing or opening) in the beam, allowing for the direct and highly accurate measurement of target velocity.

Doppler shift f_d and velocity v are dependent on each other according to the following equations:

$$f_d = \frac{2 \cdot f_{Tx} \cdot v}{c} \cdot \cos \alpha$$

$$v = \frac{c \cdot f_d}{2 \cdot f_{Tx} \cdot \cos \alpha}$$

f_d : Doppler frequency [Hz]

f_{Tx} : Carrier frequency (24.0 x 10^9 Hz)

v : Object velocity [m/s]

c : Speed of light in vacuum (3 x 10^8 m/s)

α : Angle between beam center and target moving direction

2.2 About FMCW radar

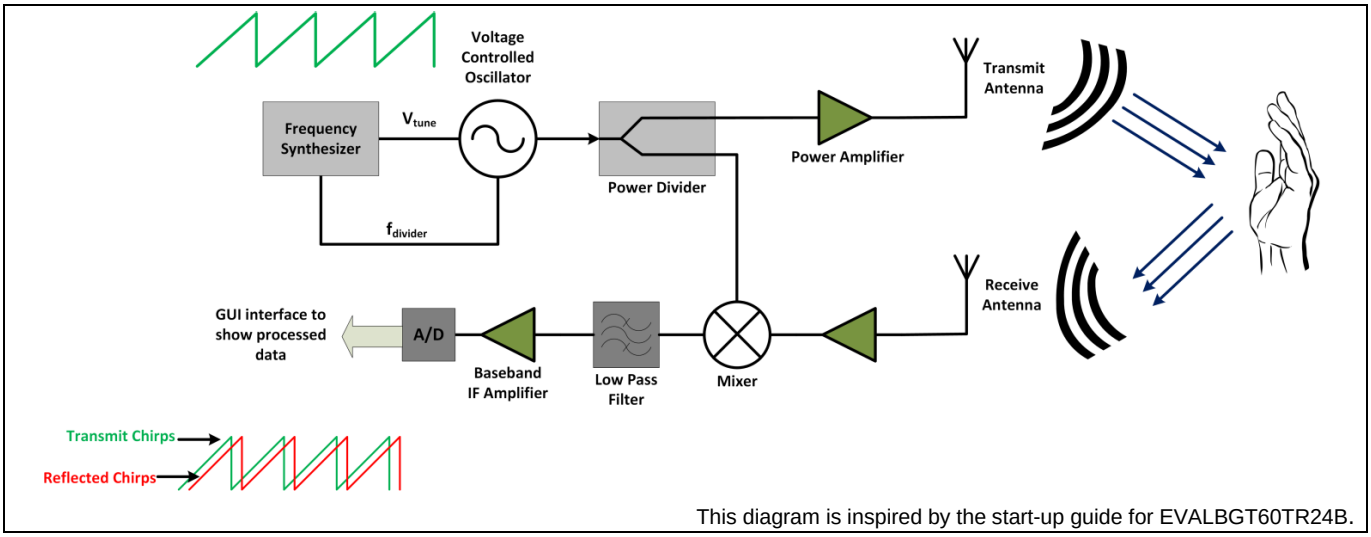

Figure 2 FMCW radar system block diagram

Figure 2 shows the flow of a typical Frequency Modulated Continuous Wave (FMCW) radar system. A frequency modulated transmitted signal is sent through the transmit antenna. The reflected signal from the target is obtained at the receive antenna. This signal is mixed with the transmitted signal at the mixer to obtain the Intermediate Frequency (IF) output. The IF signal is then digitized at the ADC to obtain the data samples of the received output.

2.2.1 Target range estimation from FMCW

Figure **3** shows the basic operation of the FMCW radar system. The transmitted signal is a frequency-modulated signal. The received signal at the receiver output is obtained after mixing the transmitted signal and the received signal. This generates an IF signal, which can be seen in the figure as a sinusoidal signal. The frequency of the IF signal corresponds to the beat frequency. This beat frequency is used to estimate the range of the target. In case of multiple targets, the IF signal comprises sinusoids with different frequencies corresponding to the target ranges.

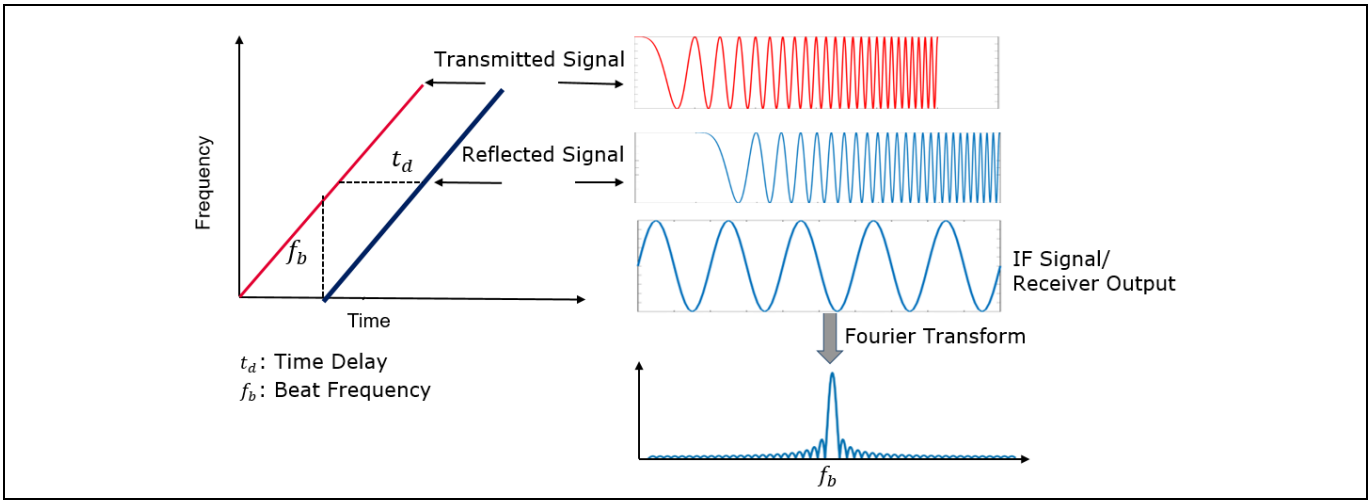

Figure 3 Basic principle of operation of the FMCW radar system

Figure 3 shows how to obtain the beat frequency by computing a Fast Fourier Transform (FFT) over the IF signal. The range of the target can be estimated from the beat frequency as follows:

$$R = \frac{C\,T_c\,f_b}{2\,B}$$

R : Target distance [m]

C : Speed of light in vacuum (3 x 10^8 m/s)

T_c : Up-chirp time [s]

f_b : Beat frequency corresponding to the target [Hz]

B : Bandwidth [Hz]

2.2.2 Doppler estimation with FMCW

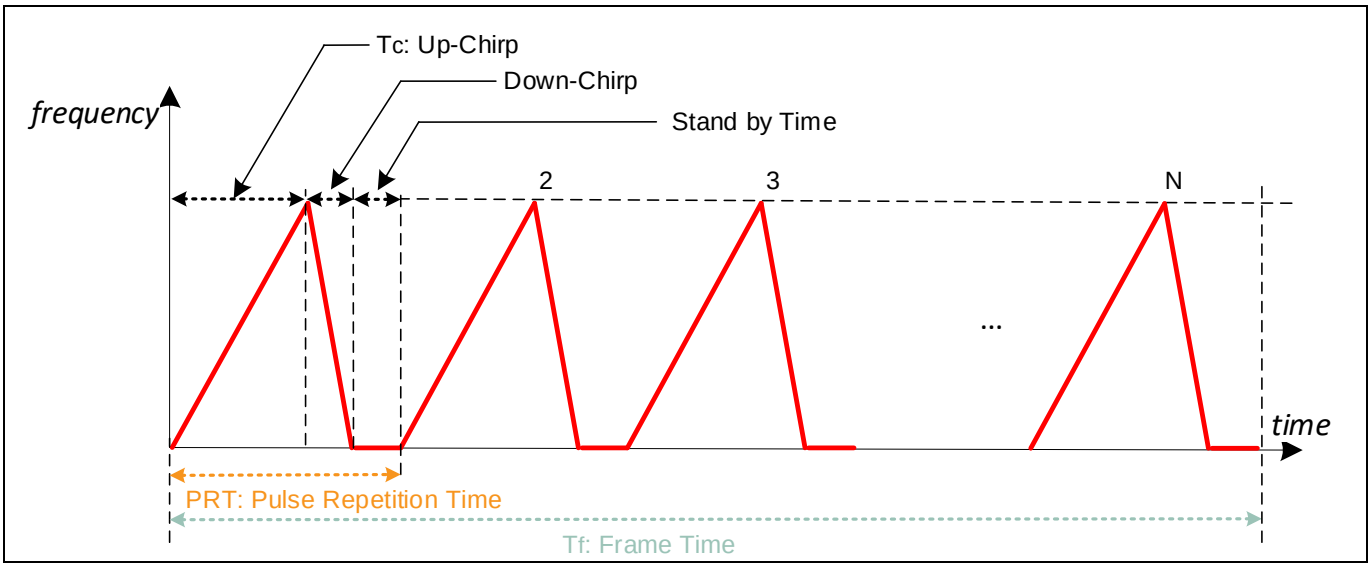

Figure 4 Multiple chirp configuration of FMCW for Doppler estimation

Motion in targets produces changes in the IF signal at the receiver. The frequency of the IF signal remains almost the same; however, the phase of the IF signal shows changes. This change in phase of the IF signal from one chirp to another is related to the radial velocity of the targets and can be given as:

$$\Delta\omega = \frac{4\,\pi v_r\,T_c}{\lambda}$$

$\Delta\omega$: Change in phase [rad]

v_r : Radial velocity of moving target [m/s]

T_c : Chirp duration [s]

λ : Wavelength [m]

The change of phase can be evaluated over multiple chirps to obtain the velocity of the moving target. This can be accomplished by computing FFT over the chirps.

- **Sample** – Each up-chirp is digitized by an ADC to 12-bit, complex (I/Q), and time-domain raw data stored as 2-byte-value. Currently, the maximum capturable number of samples per received chirp is defined by the SAMPLES_PER_CHIRP macro in the *config.h* header file.

- **Chirp** – In FMCW mode, the radar transmits a signal with linearly changing frequency over time. This signal is commonly referred to as a “chirp”. The frequency of a chirp signal can change from low to high (up-chirp) or from high to low (down-chirp).
  A chirp is characterized by its start frequency f_c, bandwidth B and duration T_c.

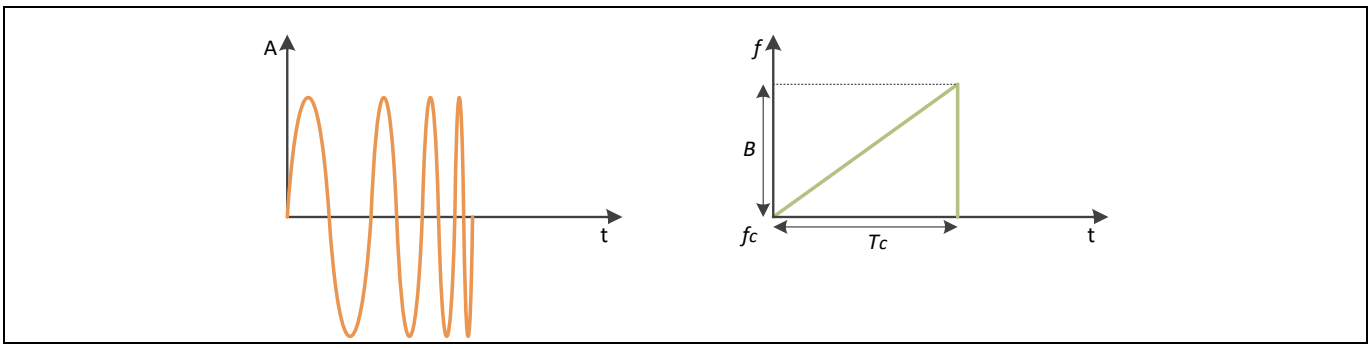

Figure 5 Chirp signal with amplitude, frequency as a function of time

- **Frame** – This is a sequence of one or N equally spaced chirps generated with the same set of parameters. All chirps are processed consecutively with minimum delay. Figure 4 shows N multi-chirps signal with frequency as a function of time. The steady time between two chirps, called Chirp Time Delay (CTD), consists of enabling the Phase Locked Loop (PLL) before the (first) up-chirp and settling time after the down-chirp.

- **Single-chirp vs. multi-chirp** – A radar system must configure chirp parameters and allow for multiple chirp configurations in a single frame. So that it transmits more than two chirps, in order to measure the speed of a target, a simple phase comparison technique is insufficient.

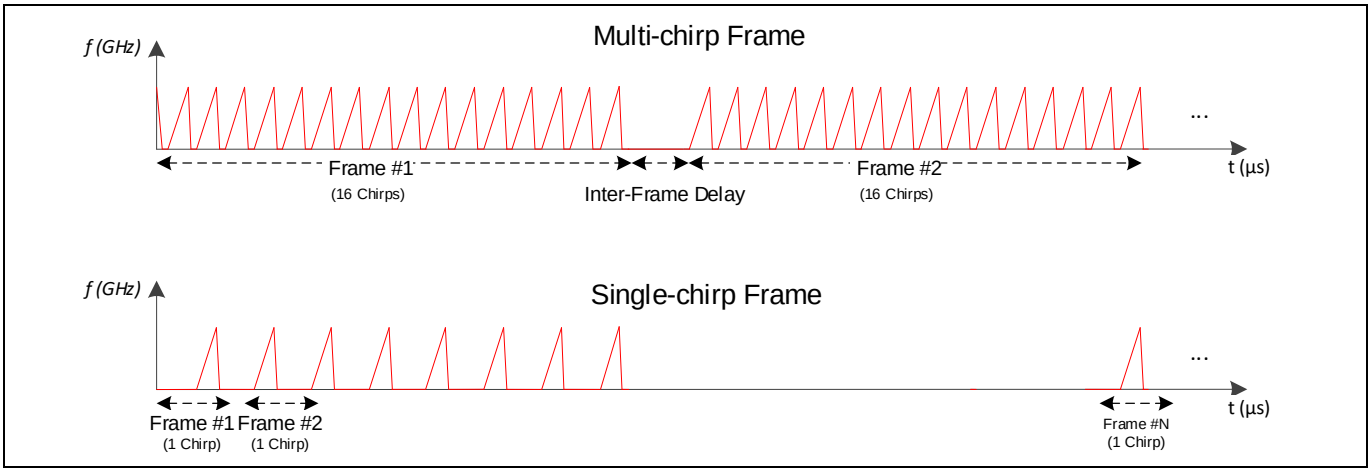

Figure 6 Single-chirp vs. multi-chirp signal

3 Hardware overview

This section gives a detailed overview of the Position2Go hardware platform. The firmware running in the XMC4700 MCU configures the following peripherals:

- Analog-to-Digital Converter (ADC)
- Direct Memory Access (DMA)
- General-Purpose Input Output (GPIO) – for SPI protocol
- Capture and Compare Unit (CCU) – for firmware timings
- Hardware/software interrupts
- USB interface – for host communication

This is in order to communicate with the components embedded on the Position2Go board shown in **Figure** 7.

- BGT24MTR12 – Infineon highly integrated 24 GHz transceiver IC
- LMX2491 – Phased Locked Loop (PLL)
- PGA112 – Programmable Gain Amplifier (PGA)

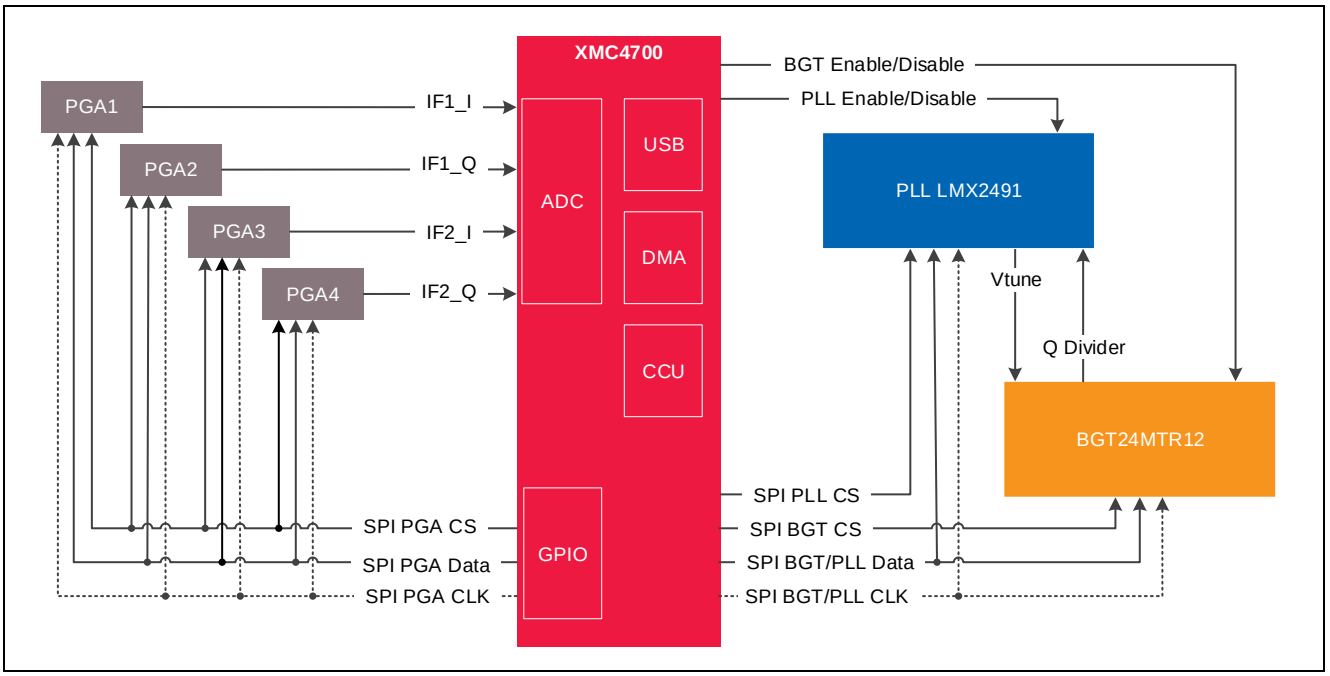

Figure 7 Position2Go baseband module

4 Firmware description

4.1 Overview

Firmware (FW) is a piece of software written in C language to control different ICs and peripherals via the host processor, which is the XMC4700 Cortex M4 MCU in the Position2Go Kit.

The Position2Go firmware is developed with Infineon's DAVE™4 (Digital Application Virtual Engineer), Infineon's free development toolchain. It is a C/C++-language software development and code generation tool for XMC™ microcontroller applications using DAVE™ APPs to configure the MCU peripherals (ADC, DMA, CCU4…), which reduces development time and allows for quick porting of the firmware across XMC™-series MCUs.

The Position2Go firmware includes various radar demonstration applications to demonstrate the Position2Go board's capabilities and facilitate the development of user applications, and that can be used to detect:

- Position of multiple targets
- Distance of multiple targets
- Motion, speed, and direction of movement of multiple targets (approaching or retreating)

4.2 Global architecture

This section describes the software components of the Position2Go firmware illustrated in Figure 8.

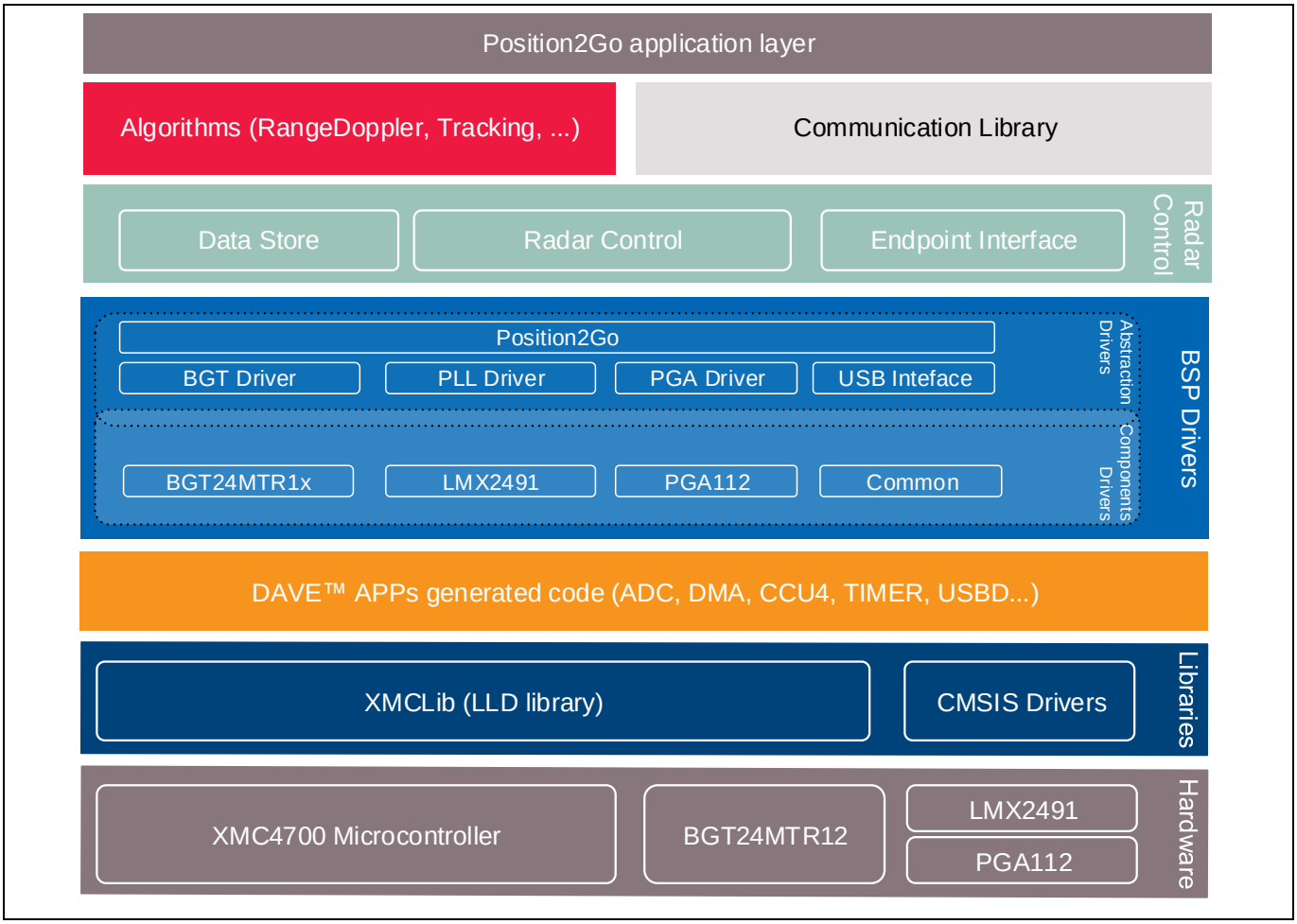

Figure 8 Firmware architecture

- **Position2Go application layer** – is a customer-specific layer that defines the entry point of the Position2Go demonstration platform and contains:
    - The initialization functions for XMC™ peripherals, host communication library and radar control layer
    - The main application state machine
    - Function template for the user to add their own code (data acquisition callbooks, algorithms process…).
- **Algorithms** – contains the supported algorithms by the Position2Go demonstration platform implementations, e.g. range Doppler, tracking… used for processing and calculating information out of radar raw data in order to detect stationary and moving objects (refer to the algorithms section for more details).
- **Communication library** – contains a set of functions to ensure USB data communication between the Position2Go board and the Radar GUI tool:
    - Defines all communication endpoints
    - Contains communication endpoints settings and configuration
    - Contains protocol communication layer.
- **Radar control layer** – contains high-level functions that can be used to set the specific mode for the Position2Go board, basically classified into three categories:
    - **Radar control** – offers high-level radar services to the user application and host communication library layers, e.g. radar device initialization, radar start, radar stop, set/get calibration…
    - **Data store** – contains global structures for hardware settings and algorithms configuration
    - **Endpoint interface** – ensures communication between the host communication library and radar control layer.
- **BSP driver** – The Board Support Package (BSP) driver is a set of functions that can be used to control and manage all components embedded in the Position2Go board, and it contains:
    - Low layer functions to initialize and control the specific board features (BGT24MTR12, PGA112, LMX2491…)
    - Functions to control power-up and power-down sequences for all hardware components
    - Functions to manage the data acquisition process from BGT to XMC™ microcontroller internal RAM memory (DMA, timer, ADC)
    - Specific driver for each component (BGT, PGA, PLL).
- **DAVE™ APPs generated code** – contains the generated library sources from DAVE™-configured building block APPs for XMC4700 MCU peripherals. It contains Application Program Interfaces (APIs) and data structures meant to be used in application code.
- **Libraries** – contains the following libraries:
    - **CMSIS** – Cortex Microcontroller Software Interface Standard (CMSIS) is a vendor-independent hardware abstraction layer for the Cortex-M processor series and defines generic tool

interfaces. The CMSIS enables consistent device support and establishes simple software interfaces to the processor and the peripherals, simplifying software reuse, reducing the learning curve for microcontroller developers, and reducing the time-to-market for new devices.

- **XMC-Lib** – consists of various low-level drivers for the XMC™ microcontrollers' family peripherals. Each driver consists of a set of routines and data structures covering all peripheral functionalities. Built on top of the CMSIS, it provides access to all XMC4000 peripheral features.

4.3 Firmware concept

Position2Go firmware can be divided into repetitive and non-repetitive tasks. After device initialization, control resides in the main loop, where two possible events can trigger further processing:

- Frame timer interrupt
- GUI request by the host communication protocol

A task-level flow diagram of the firmware is shown in Figure 9.

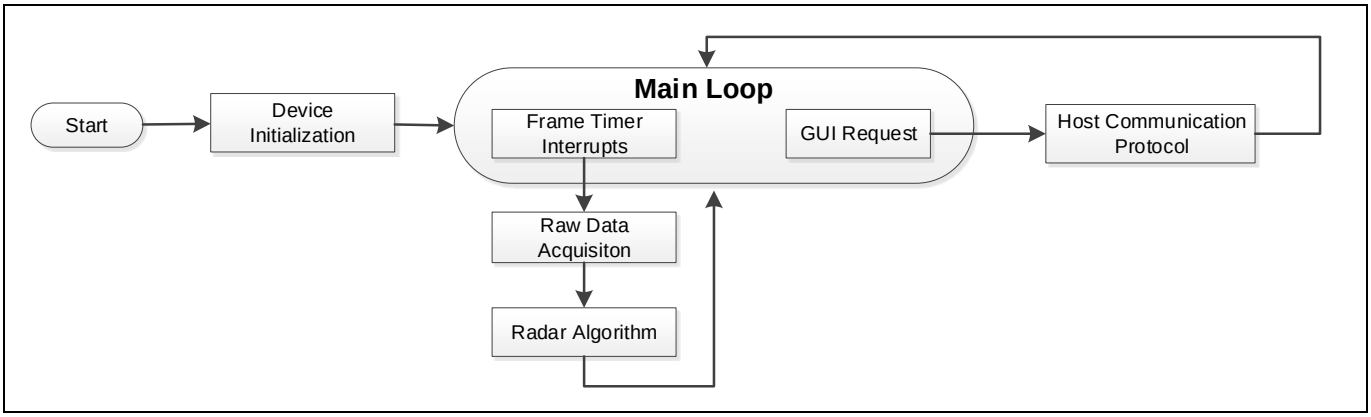

Figure 9 Firmware flow diagram

- **Device initialization** – This is the first task to be executed, only once when the firmware is started. During this task, the firmware initializes XMC™ peripherals and the Position2Go radar device, then registers the used endpoints for the host communication. Program control then goes to the main loop.
- **Frame timer interrupts** – These are the internal trigger of the chirp generation phase over PLL, in order to start the raw data acquisition process.
- **Raw data acquisition** – The BSP layer is configured to start collecting raw data from the radar device.
- **Radar algorithm** – Once raw data is collected, run the registered algorithm process in order to compute the parameters of the target's distance, angle and velocity.

4.3.1 Raw data acquisition

Figure 10 shows the main blocks of the Position2Go raw data acquisition phase in detail, before going through the radar algorithm processing phase. There are more details about the chirp generation and data sampling phases in the next sections.

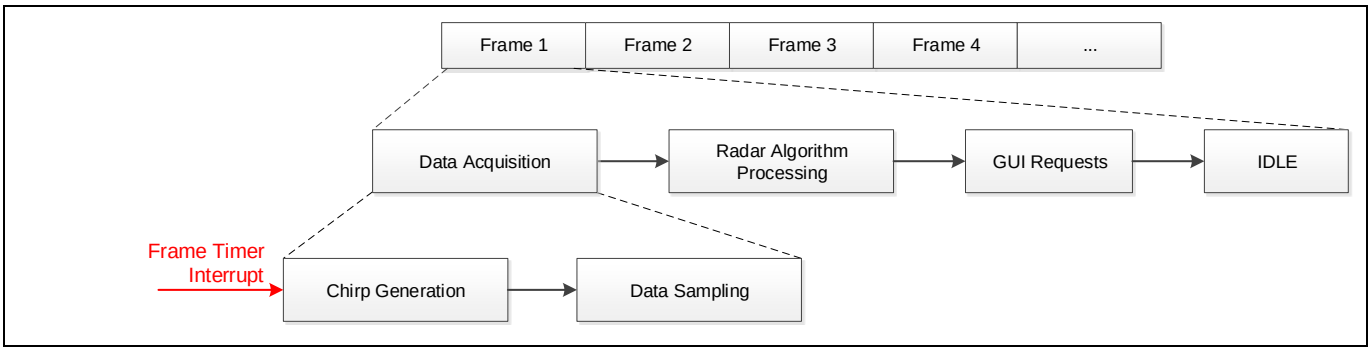

Figure 10 Raw data acquisition flow diagram

4.3.2 Chirp generation

Chirp generation is the first part of the data acquisition process; it is triggered by an internal frame timer, which will start generating chirps over PLL. For Position2Go, the firmware communicates with the PLL module via SPI to configure the ramp generation patterns, their direction and their order of generation. Bandwidth and chirp time parameters are taken into consideration to configure ramps.

Figure 11 depicts the flow diagram for FMCW chirp signal generation.

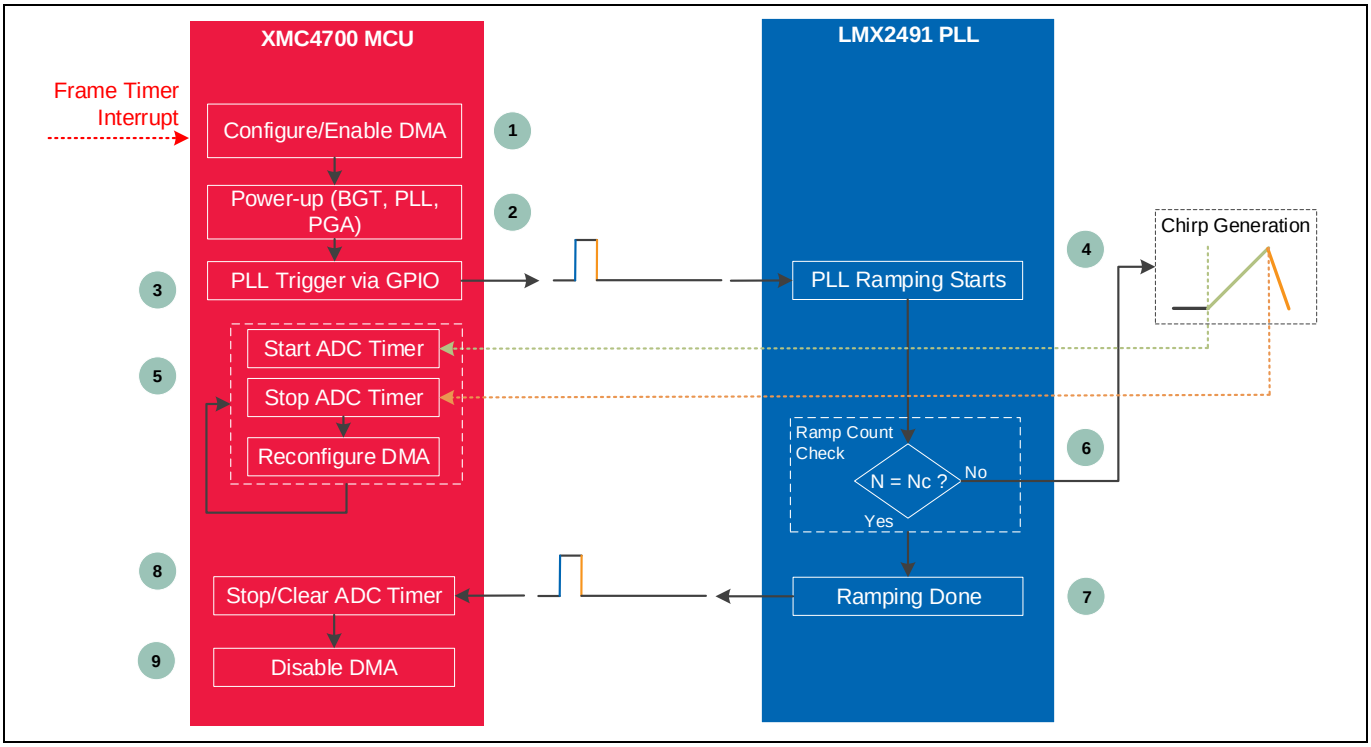

Figure 11 FMCW chirp generation flow diagram

Here is a short description of each of the above blocks:

- **Configure and enable DMA** – DMA source/destination address set-up; four DMA channels are configured for the two RX complex (I/Q) data samples (Q1, I1, Q2 and I2)
- **Trigger PLL high** – rising edge, the signal controlling the PLL via GPIO in order to start PLL ramping, and to start the ADC timer once the ramp starts
- **Start ADC timer** – CCU4 is set as ADC timer and started in order to trigger equidistant ADC samples
- **Trigger PLL low** – falling edge, the signal controlling the PLL via GPIO in order to stop PLL ramping
- **Disable DMA** – disables the DMA peripheral
- **Stop and clear ADC timer** – ADC TRIG timer has to be controlled

The generated chirp signal for the Position2Go FMCW application is shown in Figure **12**.

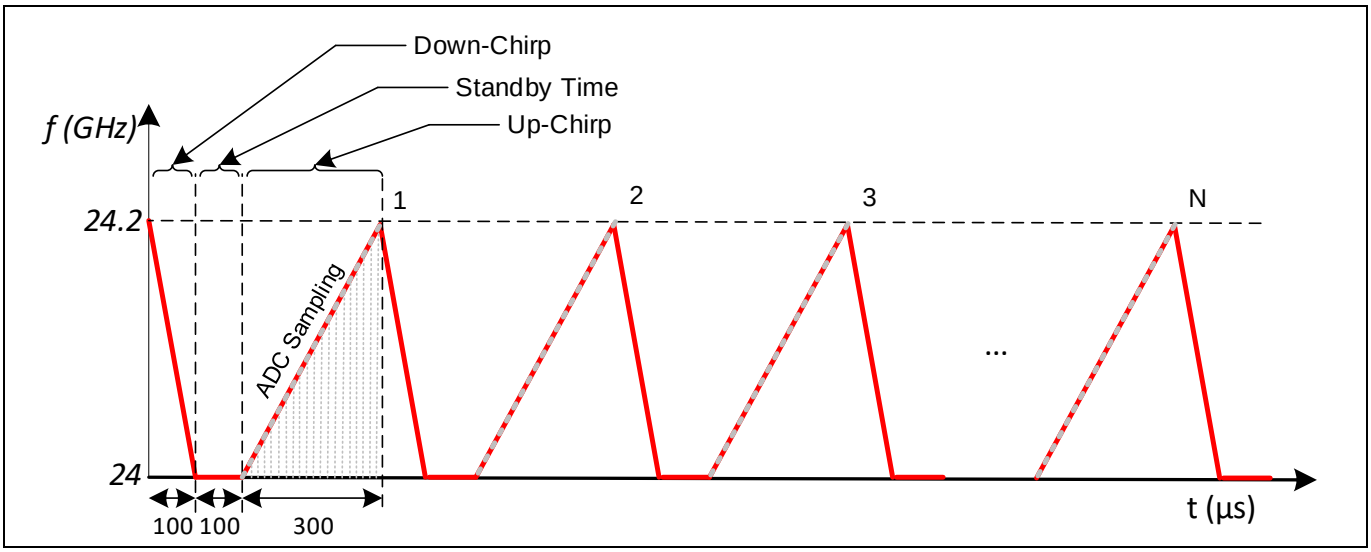

Figure 12 Position2Go generated FMCW multi-chirp signal

The up-chirp is sampled and extracted raw data are used by the radar algorithms, and the down-chirp is only utilized to minimize the frequency over-shots after up-chirp.

Changing the chirp time (up-chirp time) also changes the ADC timer period. The chirp time setting is limited because of MCU processing capabilities and PLL configuration limitations. Minimum chirp time is dependent on the desired number of range bins.

Note: Only the up-chirp time is configurable by the user in the config.h file within a range of [50, 2500 µs] and the down-chirp time as well as the standby time have fixed values of 100 µs.

Currently, for Position2Go radar firmware:

- Up to 16 chirps per single frame are generated for FMCW radar
- Chirp time must be at least 300 µs for 256 samples per chirp (Ns), 200 µs for 128(Ns) and 100 µs for 64(Ns), to ensure a supported sampling rate; valid values are in the range of [50 to 3000] µs.
- A settling time of 50 µs after each chirp is required to avoid clipping at the beginning of the subsequent chirp

4.3.3 Data sampling

The start of the data acquisition process will be triggered by an internal frame timer, which will start chirp generation over PLL. When the PLL launches chirp generation, the sampling timer is triggered at the same time, which triggers the ADC to start sampling data from RX antennas.

Once an ADC sample is ready to be moved to the acquisition buffer, the DMA transfer starts. This task is repetitive; it will end once the number of required samples per chirp (Ns) is reached. At that moment, the DMA will stop the transfer process by raising a transfer complete interrupt.

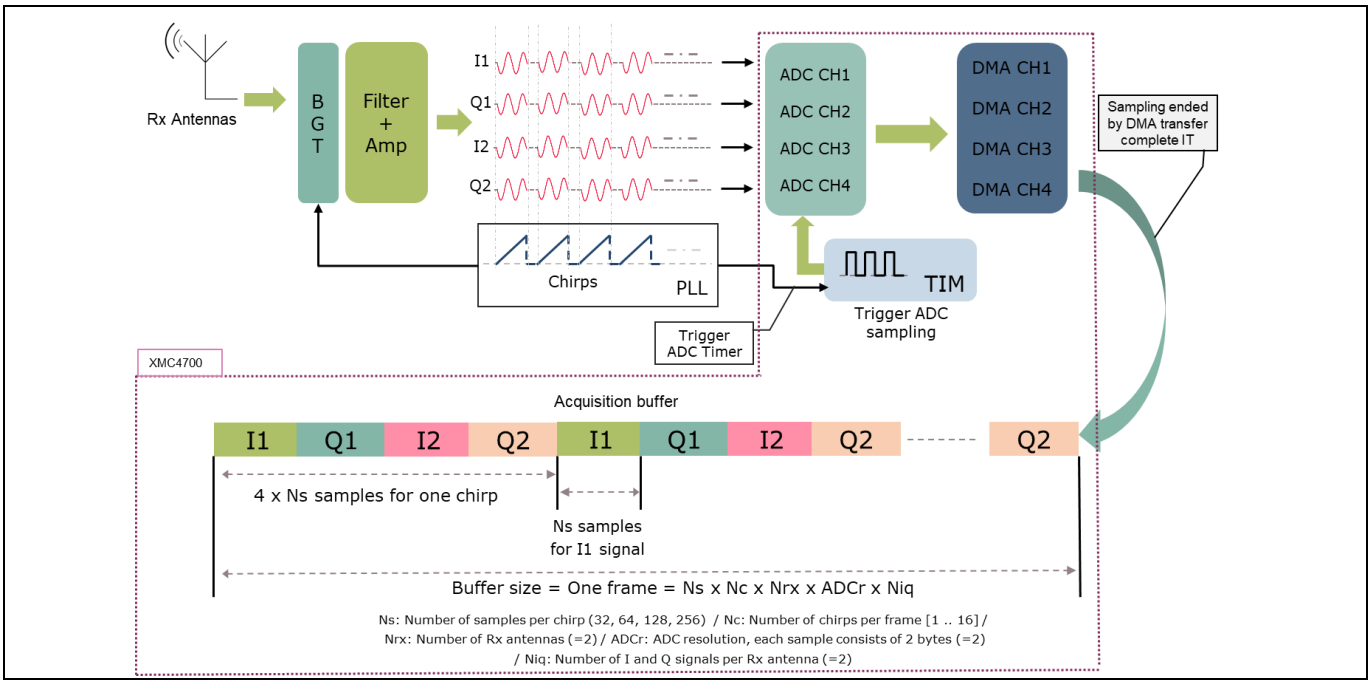

Figure 13 Position2Go data acquisition and sampling flow

Raw signal acquisition is performed through the Event Request Unit (ERU), where I/Q signals are acquired via ADC, based on the ERU configured action, without any ongoing SW control. **Figure** 14 depicts how the PLL and the XMC™ MCU are interconnected.

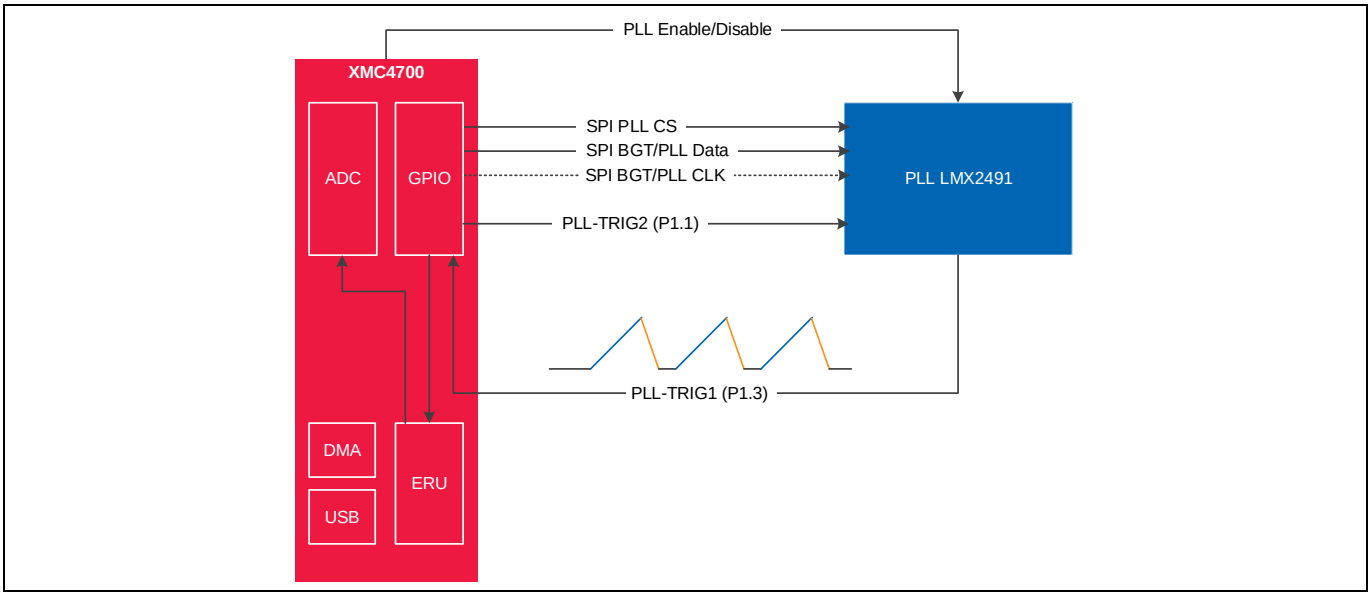

Figure 14 PLL and XMC™ ERU interconnection

The ERU is used to start ADC conversions based on the port pin state following these steps:

- The XMC4700 MCU triggers PLL to start ramp via the PLL-TRIG2 (P1.1) pin.
- PLL generates the ramp signal, which is fed into MCU via PLL-TRIG1 (P1.3) pin.
- The ERU1 PLL-TRIG1 (P1.3) pin is used to trigger ADC acquisition, based on the detected edges of PLL ramp signal. Detected up-chirps are considered as trigger signal to start the ADC timer and down-chirps to stop it. Timer interval is deduced through the CCU4 slice, which also defines the ADC sampling ratio.
- ADC data sampling and DMA transfers are performed within the measurement time window.

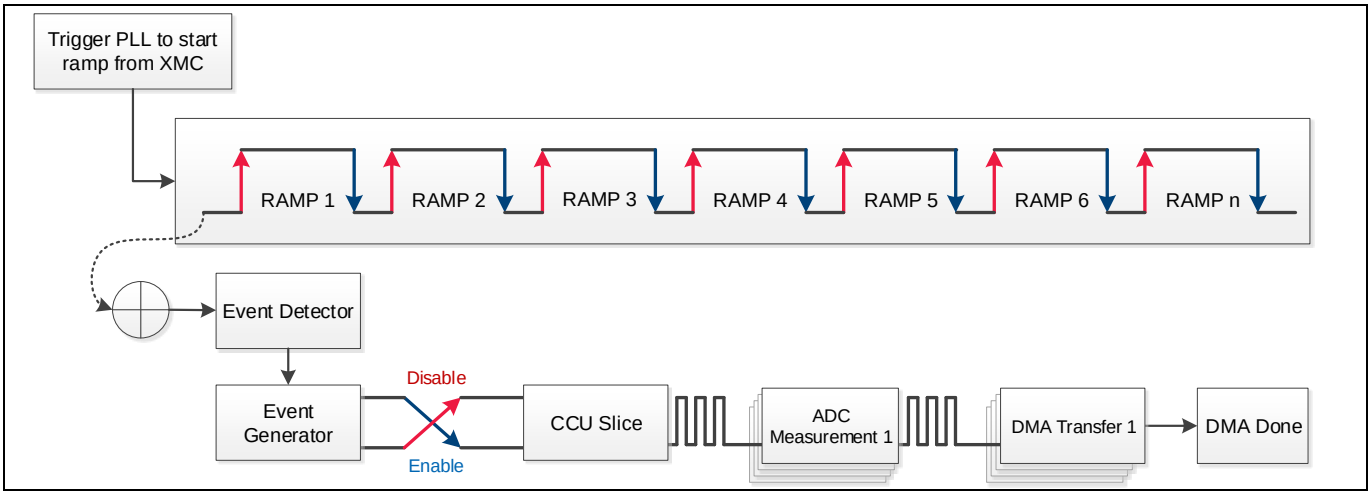

Figure 15 ADC acquisition via ERU

Note: During acquisition, the XMC4700 MCU can enter into SLEEP power-optimized mode to reduce power consumption. The current firmware does not include the settings to put the microcontroller into SLEEP mode, since in this mode the MCU will not be able to service GUI requests; it is completely up to the end user to define such settings.

Please refer to Figure 16, which gives the detailed flow diagram of the data acquisition phase.

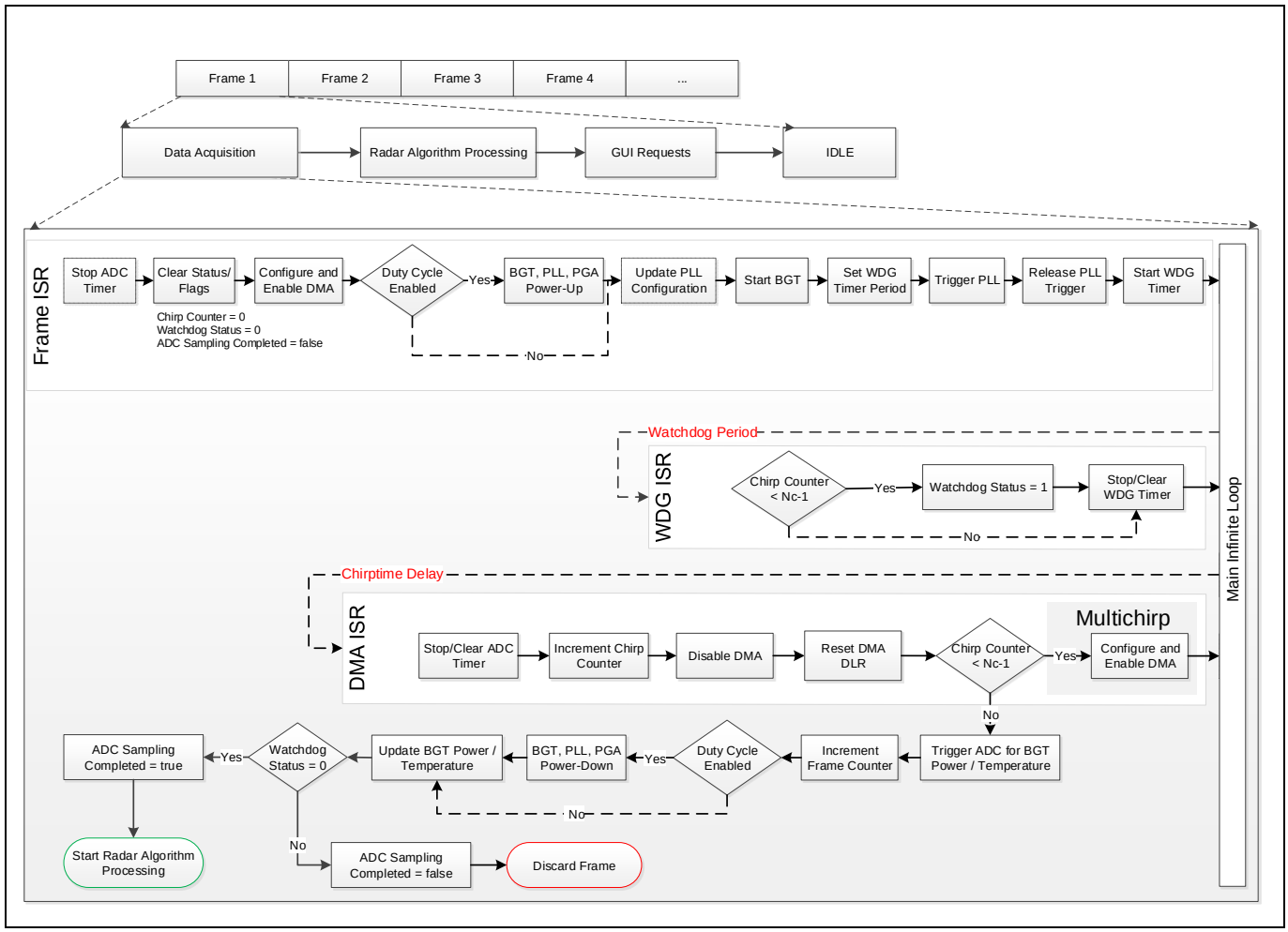

Figure 16 Data acquisition flow diagram

4.4 Power saving

The Position2Go module offers the possibility to operate the BGT24MTR12 in a duty-cycle mode. Enabling duty cycling keeps the overall power consumption and thermal dissipation very low. Enabling/disabling the duty-cycle mode is performed by turning PLL/BGT on/off. By disabling the duty-cycling, the board is kept constantly on in order to carry out lab testing and measurements.

The duty-cycle mode is composed of the following stages:

- **BGT PLL power-up** – The BGT module is enabled by setting the DIGITAL_IO_BGT_POWER_ENABLE pin to low, followed by a delay inserted for BGT power-up. Afterwards, the DIGITAL_IO_PLL_CE pin is set to high to enable the CE of the PLL.
  If the PLL configuration is changed, the PLL ramps need to be disabled in order to update the PLL configuration and subsequently enabled again via the SPI protocol. A delay for PLL lock is inserted before configuring and starting the BGT via SPI. A delay is needed for BGT Q2 divider to be zero and VCO to be settled after the SPI setting is transmitted.
  The *bsp_components_power_up ()* method is used to define the BGT and PLL power-up protocol with the appropriate delays during duty-cycling mode.

- **BGT PLL power-down** – PLL ramps are disabled, the BGT module is stopped to avoid out-of-band spurs, and a delay is inserted for BGT power-down SPI settings. Then, the BGT and PLL modules are disabled by setting the DIGITAL_IO_BGT_POWER_ENABLE pin to high and the DIGITAL_IO_PLL_CE pin to low, respectively.
  The method *bsp_components_power_down ()* is used to define the BGT and PLL power-up protocol with their respective delays during duty-cycle mode.

Figure 17 shows the duty-cycle mode sequence diagram.

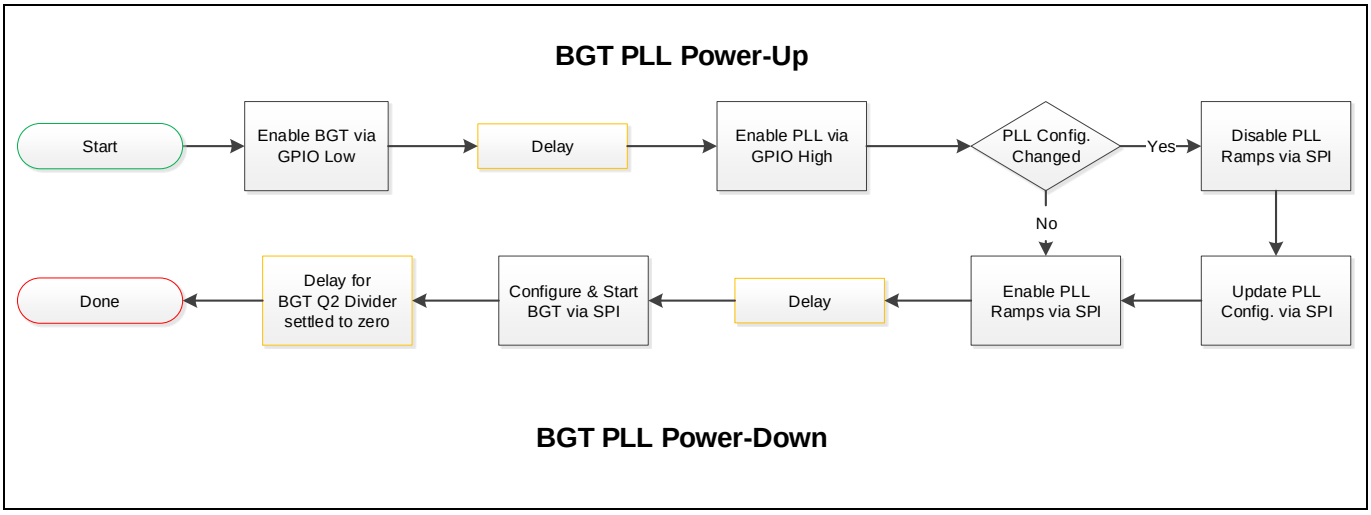

Figure 17 Duty-cycling flow diagram

The duty-cycle mode feature can be enabled/disabled through the DUTY_CYCLE_ENABLE defined under the configuration's project file *config.h* and via the Radar GUI application.

4.5 Radar control layer

The Position2Go software package provides a simple interface to the radar kit through the radar control layer, which offers full flexibility to configure all radar parameters based on the application end requirements.

4.5.1 Radar control API

The APIs define the high-level interface used to configure the radar RF parameters, the behavior and capabilities of the component, and its inputs and outputs, and provide a set of firmware methods to manage radar functionalities. Radar control APIs are called from the application layer and are prefixed by “radar_”.

Detailed technical information about the radar APIs available to the user is provided in a compiled HTML file in the Firmware_Software/Documentation/FW_API folder, where all the functions and parameters are described.

4.5.2 Data store module

The data store module is apart from the radar control layer; it mainly contains the hardware device settings and the algorithm settings structures, as shown in Figure **18**.

```
/*
 * Device Settings Structure
 */
typedef struct device_settings_TAG {
    /* PLL Settings */
    uint32_t        pll_chirp_time_usec;
    float           pll_frequency_KHz;
    float           pll_lower_frequency_KHz;
    float           pll_upper_frequency_KHz;
    float           pll_bandwidth_MHz;
    uint32_t        pll_num_of_chirps_per_frame;

    /* ADC Settings */
    uint32_t        adc_sampling_freq_Hz;
    uint8_t         adc_resolution;
    uint8_t         adc_use_post_calibration;

    /* BGT and PGA Settings */
    uint16_t        pga_rx_gain_level;
    uint8_t         power_duty_cycle_enable_flag;
    uint8_t         bgt_tx_power_level;
    uint8_t         bgt_rx_lna_gain_enable_flag;

    /* Frame Settings */
    uint32_t        frame_period_usec;
    uint32_t        num_samples_per_chirp;
    uint8_t         rx_antenna_mask;

    uint32_t        isGainlevelUpdated;
    uint32_t        is_duty_cycle_enable_updated;
    uint32_t        isUpdated_fmcw_config;
    uint32_t        isUpdated_doppler_config;

    uint32_t        pll_modulation_mode;

} device_settings_t;
```

```
/*
 * Algorithm Settings Structure
 */
typedef struct algo_settings_TAG {
    uint8_t     isUpdated;
    uint8_t     isChecked;

    uint32_t    max_number_of_targets;
    uint32_t    max_number_of_tracks;

    uint32_t    tracking_enable;
    uint32_t    num_of_tracks;
    uint32_t    mvg_avg_len;
    uint32_t    median_filter_len;
    uint32_t    mti_filter_len;
    uint32_t    mti_filter_enable;
    uint32_t    range_thresh_type;
    uint32_t    adaptive_thresh_offset;
    uint32_t    range_offset_cm;
    int16_t     angle_offset_deg;

    uint32_t    min_distance_cm;
    uint32_t    max_distance_cm;
    uint32_t    range_detection_threshold;

    uint32_t    speed_detection_threshold;
    uint32_t    min_speed_kmh;
    uint32_t    max_speed_kmh;
    float       wave_length_ant_spacing_ratio;
    float       min_angle_for_track_assignment;

} algo_settings_t;
```

Figure 18 Data store hardware device and algorithm settings’ structures

These two structures are shared between all firmware modules. If a firmware module, e.g. the host communication library or algorithm, requires one or more parameters from the settings structures, a ***fetch operation*** is performed to get the up-to-date value from the data store, as shown in Figure **19**.

On the other hand, if there is an update for one or many parameters in the settings structures, a ***push/store operation*** is performed to update the data store structures with the new changed value.

A shadow (or a copy) of the hardware settings structure is maintained in the data store, and it always contains the old settings parameters. In case of unsupported parameters (e.g. the value is out of range), the current hardware structure is discarded and overwritten by the shadow copy and the new required change will be discarded.

At the beginning, the shadow and current settings structures parameters are set to the default settings values from the *config.h* file. The user can change these default settings by updating the *config.h* file or through the Radar GUI tool interface.

In case of a valid hardware change request received from the GUI, the hardware structure will be updated with the new value in the data store. The ***Radar Control*** submodule will apply this change on the BSP driver before the start of acquisition of the next new frame.

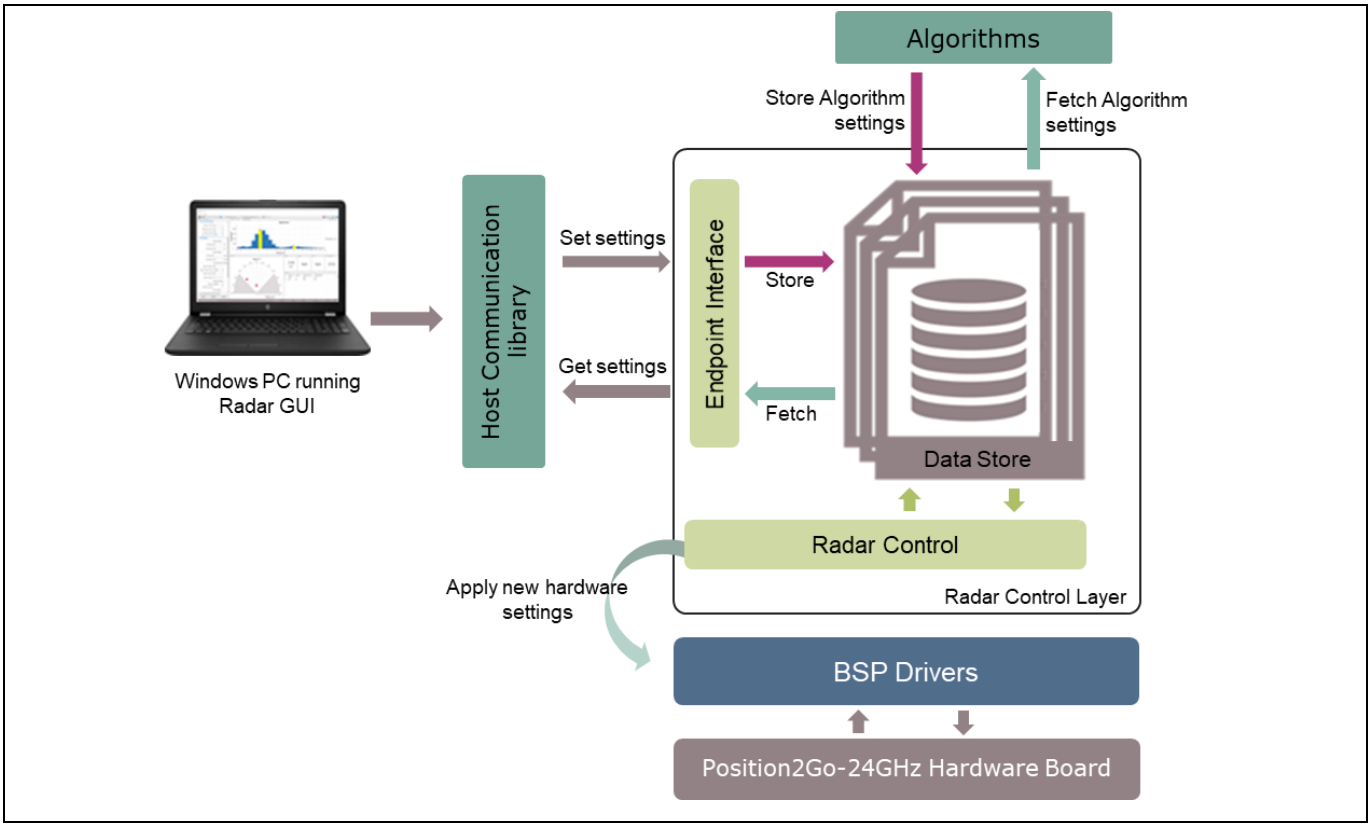

Figure 19 Interconnection of the data store module with other firmware modules

4.6 DAVE™ project overview

The Position2Go firmware is released as a ready-to-run DAVE™4 project, where source files are generated based on the DAVE™ APPs used, which are graphical-configurable application-oriented software components, used to enable users' quick reuse and customization.

Table 1 lists the DAVE™ APPs used, based on the Position2Go board, to generate the appropriate firmware source code.

Table 1 DAVE™ project APPs used

DAVE™ app	Number of instances	App description
ADC	4	Allows for digitizing analog signals using ADC via queue and scan request sources with advanced features
CMSIS_DSP	1	Provides the CMSIS DSP software library, a suite of common signal processing functions to apply on Cortex-M processor-based devices
DIGITAL_IO	13	Used to configure a port pin as digital input/output
DMA_CH	4	Used to perform single and multi-block data transfer using the General Purpose Direct Memory Access (GPDMA) module on the XMC4000
E_EEPROM_XMC4	1	Emulates a part of program Flash as EEPROM for permanent data storage
INTERRUPT	1	Enables overwriting of the Interrupt Service Routine (ISR) provided in the system file and sets the interrupt priority
SYSTIMER	1	Uses the SysTick interrupt to call user functions periodically at a specified rate after a given time period expires
TIMER	4	Provides an accurate timer by using the hardware CCU timer; this can be used as a trigger input to other peripherals or to create an event
USBD_VCOM	1	USB virtual COM port application. This App implements the VCOM over USB CDC class driver

DAVE™ APPs used are configured to address the XMC4700 peripherals to ensure communication, data processing and result visualization. The fundamental functionalities are SPI, ADC and DMA:

- **GPIO for SPI protocol** – The XMC4700 GPIO peripheral is configured to emulate the SPI protocol interface, mainly used to configure the BGT24MTR12 radar chip, in order to:
  - Set transmitter output power (TX output power adjustable range in dB)
  - Set receiver LNA gain (RF front-end LNA gain reduction in dB)
  - Control (turn on/off) the different internal building blocks.

  In addition, the SPI interface can be used to:
  - Read out the different sensors' outputs
  - Enable/disable the LMX2491 PLL.

The GPIO pin configurations of the Position2Go board used for SPI interface communication are listed in **Table 2**, with the standard input values in brackets.

Table 2 GPIO pin configurations for SPI protocol

Peripheral pin	Pin direction	Description
		BGT configuration
BGT	Output (1)	P2.3 pin: power enable Push-pull mode, output level low HW control disabled
BGT_Data pin	Output	P8.7 pin: BGT slave data input line, used in push-pull pull-down mode
BGT_CLK pin	Output	P8.5 pin: PLL slave data input line, used in push-pull pull-down mode HW control disabled
BGT_CS pin	Output	P8.6 pin: slave chip select pin used in push-pull pull-up mode HW control disabled
		PGA configuration
PGA	Enabled	
PGA_Data pin	Output	P5.1 pin: PGA slave data input line, used in push-pull pull-down mode
PGA_CLK pin	Output	P8.5 pin: slave clock line, used in push-pull pull-down mode
PGA_CS pin	Output	P8.4 pin: slave chip select pin used in push-pull pull- down mode
		PLL configuration
PLL	Enabled	P2.2 pin: enable Push-pull mode, output level low Output strength medium edge HW control disabled
PLL_Data pin	Output	P8.7 pin PLL slave data input line, used in push-pull pull-down mode
PLL_CLK pin	Output	P8.5 pin: slave clock line, used in push-pull pull-down mode
PLL_CS pin	Output	P1.8 pin: slave chip select pin used in push-pull pull-down mode

- **ADC configuration** – The XMC4700 integrated 12-bit ADC is used to sample and process the analog down-converted signals in the baseband. The ADC peripheral configuration is set to:
  - Conversion mode: 12-bit resolution
  - Four channels, two channels for each IF (IF1_I, IF1_Q, IF2_I, IF2_ Q), with two bytes per sample for each ADC channel
  - Sample time [ns]: 75

  The ADC conversion mode has an impact on the maximum possible sample rate. Higher resolution reduces the maximum sample rate. Figure 20 shows the configuration of the ADC DAVE™ app:

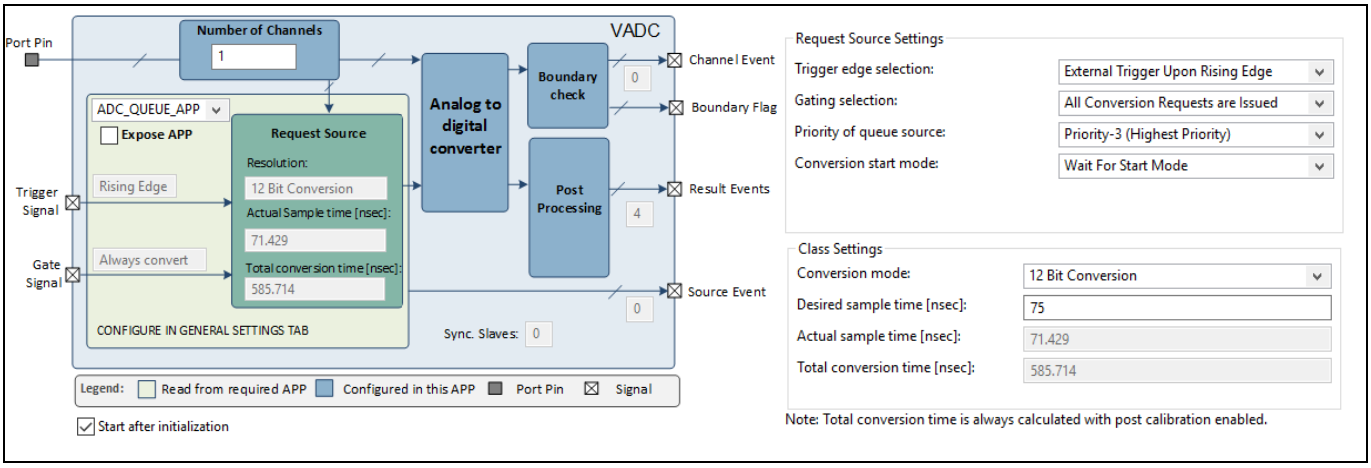

Figure 20 ADC DAVE™ app configuration

- **DMA_CH** – The XMC4700 GPDMA peripheral is configured to transfer data from the ADC peripheral to XMC™ memory. Four DMA channels are configured for the two complex RX data measurements (Q1, I1, Q2 and I2), as follows:
  - Transfer type: single block
  - Transfer flow: peripheral to memory (DMA flow controller)
  - Block size: 250
  - Transfer width: 16 bits
  - Burst width: 1 word
  - Handshaking: hardware

Figure 21 shows the configuration of the DMA_CH DAVE™ app.

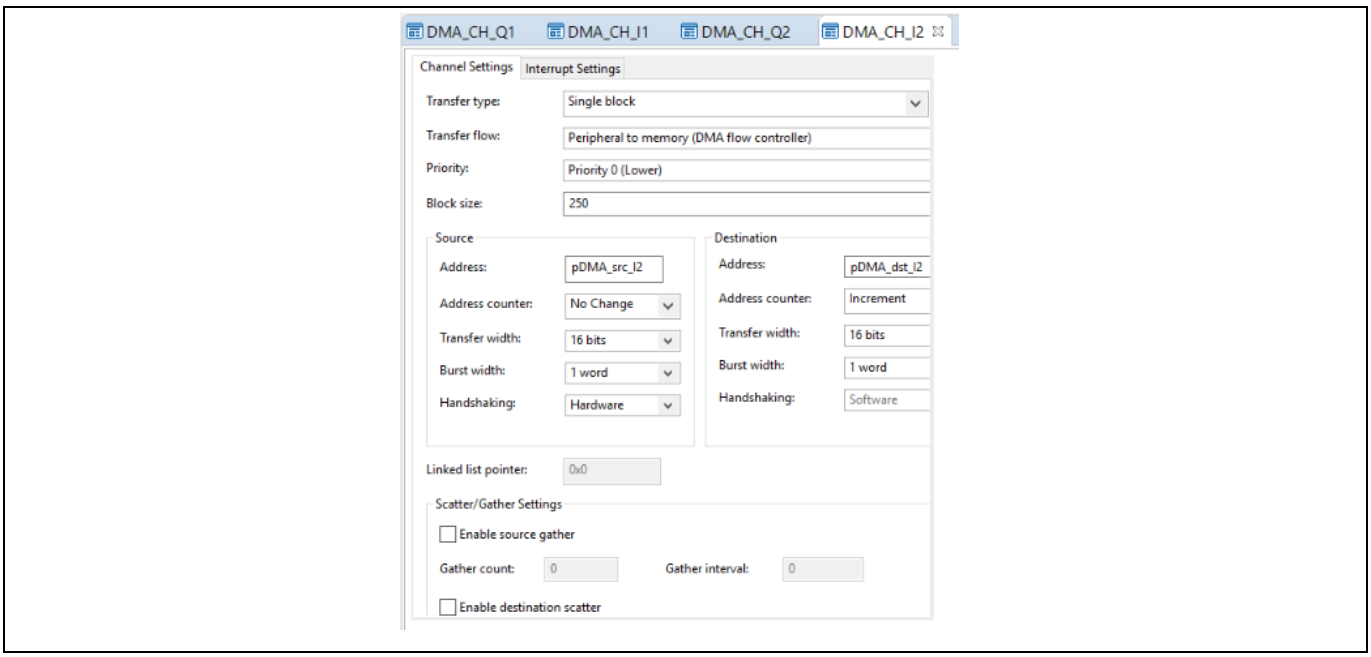

Figure 21 DMA DAVE™ app configuration

4.7 Firmware package overview

Position2Go is a firmware package for XMC™ microcontrollers and BGT24MTR12 radar chips. It provides a complete solution to build radar applications in a single package containing the source code for various exemplary applications facilitating the development of user applications. Figure 22 shows a top-level view of the Position2Go package file structure.

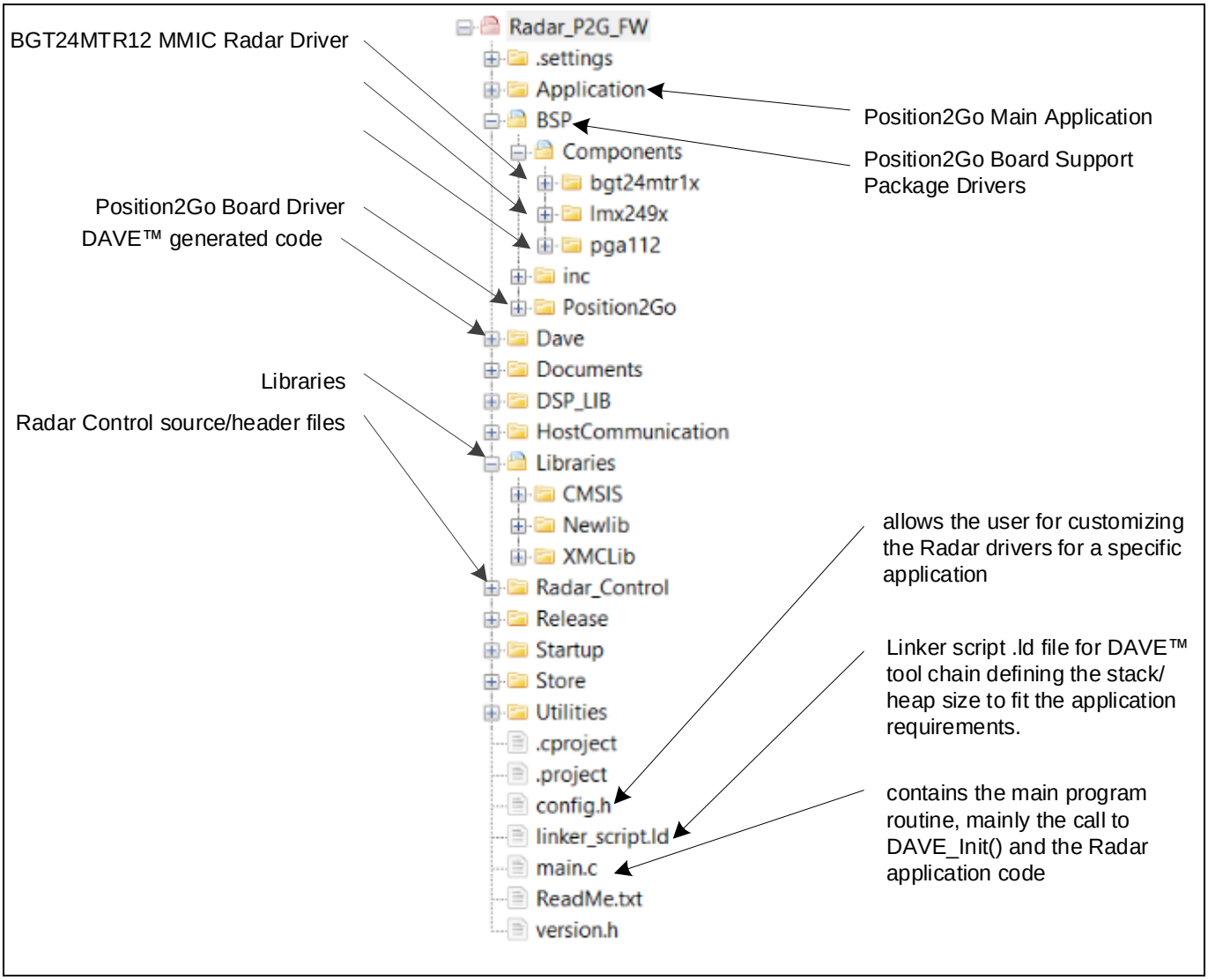

Figure 22 Package folder structure

4.8 Footprint

The purpose of the following sections is to provide the memory requirements for all the Position2Go firmware modules, including devices' drivers, algorithms, and main radar applications. The aim is to have an estimation of fixed and customizable memory requirements in case of removal or addition of a module or feature. The footprint data are provided for the following environment:

- **Board** – DEMO_POSITIION2GO v1.2
- **Firmware** – P2G_FW (V1.0.x)
- **Toolchain** – DAVE™ v4.4.2

After building a project, the build result is displayed in the console window, where the code size figures are listed. The values are organized according to memory areas, arranged by the linker file (*.ld) into the text, data and bss sections. Table 3 shows the Position2Go build memory utilization for the radar firmware configurations, main modules, and algorithms. The information has been gathered by analyzing the corresponding (*.elf) file.

Table 3 Position2Go firmware footprint

Firmware			Footprint			
Version	**DAVE project**	**Optimization**	**text[(1)] [byte]**	**data [byte]**	**bss[(2)] [byte]**	**Total [byte]**
v1.0.0	Radar_P2G_FW	None (-O0)	70220	1072	80272	151564 byte (0x2500c)
	Radar_P2G_FMCW	None (-O0)	100424	3840	154180	258444 byte (0x3f18c)
v1.0.1	P2G_FW	Optimize most (-O3)	45416	992	80272	126680 byte (0x1eed8)
	P2G_FMCW	Optimize most (-O3)	66408	3752	154180	224340 byte (0x36c54)

(1) text: code.

(2) bss: statically allocated variables that are not explicitly initialized to any value.

4.9 Firmware timings

This section presents the typical timings of the Position2Go firmware that should be used to ensure correct execution of the important radar application sequences.

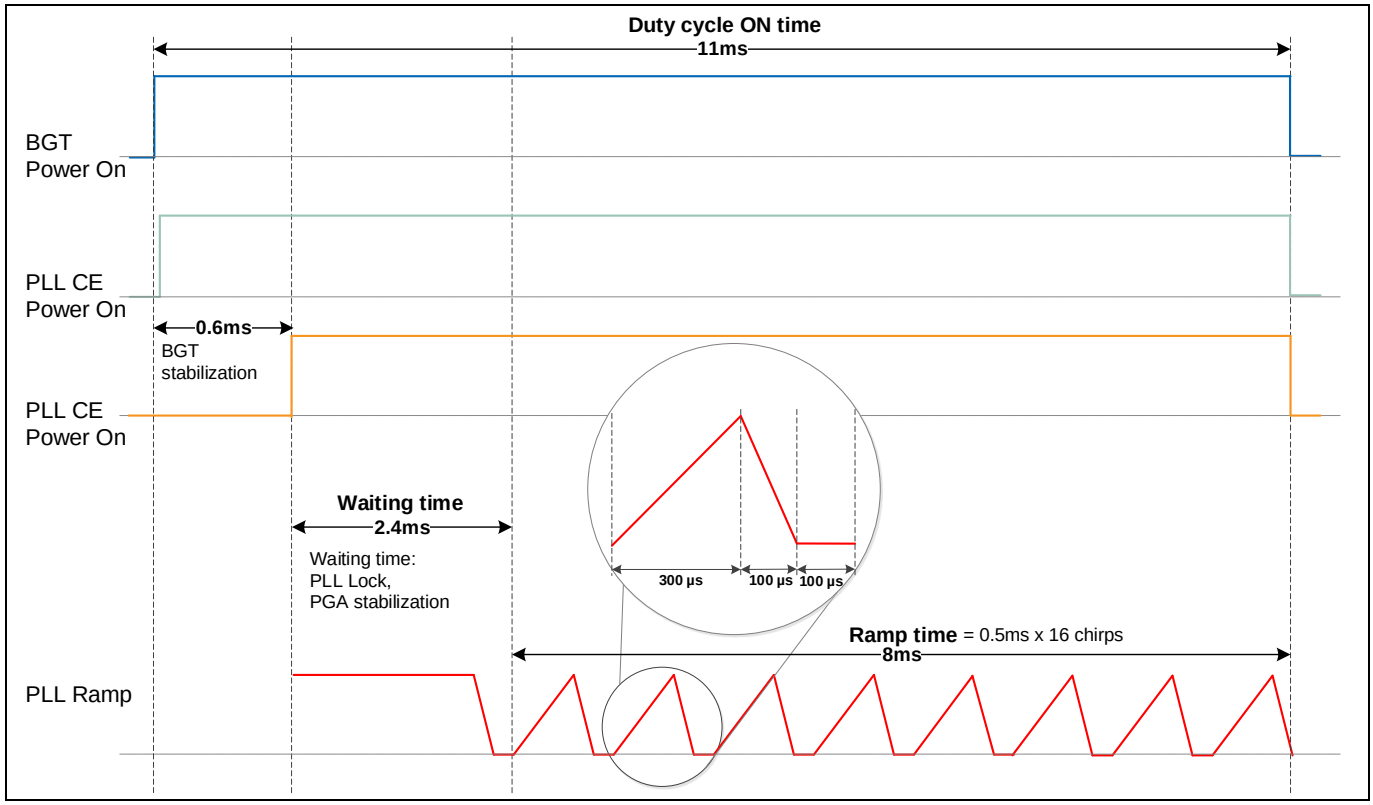

Figure 23 Raw data acquisition timings

4.10 Firmware customization and configuration

The configuration file *config.h* allows for customizing the drivers and algorithms for the Position2Go radar application. The following parameters can be configured: enable/disable or modify some options by uncommenting/commenting or modifying the values of the related define statements, as described in Table 4.

Table 4 Define statements used for radar firmware configuration

Parameter	Description	Default	Valid range
General configurations			
FW_MODULATION_TYPE	Firmware modulation type, Doppler [0] or FMCW [1]	1	[0 to 1]
NUM_OF_CHIRPS	Valid range of chirps in relation to the DATA_SIZE	16	[1 to 64]
SAMPLES_PER_CHIRP	Size of IQ raw ADC buffer	64	[32 to 256]
FRAME_PERIOD_MSEC	Time period of one frame to capture data (units in ms)	150	[50 to ...]
BGT/PGA configurations			
DUTY_CYCLE_ENABLE	Enable [1] or disable [0] duty cycling of Position2Go via BGT and PLL on/off */	1	[0 to 1]
BGT_TX_POWER	BGT TX power levels: min. [1] to max. [7]	7	[1 to 7]
LNA_GAIN_ENABLE	Enable [1] or disable [0] LNA gain in BGT RX	1	[0 to 1]
PGA_GAIN	PGA112 gain value	4	[0 to 7]
RADAR_SIGNAL_PART	Only I signal, [1] only Q signal, [2] both IQ (complex) signals are captured during radar data frame acquisition	2	[0 to 2]
DSP configurations			
RANGE_ FFT_SIZE	FFT length for FMCW mode, two times zero padding	128	2 x SAMPLES_PER_CHIRP
DOPPLER_FFT_SIZE	FFT length for FMCW mode, two times zero padding	32	2 x NUM_OF_CHIRPS
FFT_INPUT_TYPE	FFT input type: real input I [0], real input Q [1], complex input IQ [2]	2	[0 to 2]
MAX_NUM_OF_TARGETS	Maximum number of targets to be detected	5	[1 to 5]
RX_ANTENNA_SELECTION	Receiver antenna: RX1 [1] or RX2 [2]	1	[1 to 2]
ALGO_PROCESS_TIME_USEC	Minimum time (units in µs) used to process raw data for 16 chirps and 64 samples.	12000	
Tracking and filter configurations			
TRACKING_ENABLE	Enable [1] or disable [0] tracking	0	[0 to 1]
CURRENT_NUM_OF_TRACKS	Current maximum configured number detected targets to be tracked	5	[1 to 5]
MVG_AVG_LEN[(1)]	Moving average window length	2	[1 to 8]

Parameter	Description	Default	Valid range
MEDIAN_FILTER_LEN	Length of the median filter for angle, must be odd number	5	[0 to 13]
ANGLE_QUANTIZATION	Disable [0]; else number of degrees to be quantized	1	[0 to ...]
MTI_FILTER_ENABLE	Enable [1] or disable [0] MTI filter	0	[0 to 1]
MTI_FILTER_LEN	MTI filter weight	100	[1 to 1000]
MIN_ANGLE_FOR_ASSIGNMENT	Minimum angle to assign a track to a target (estimated to 50)	50	[50 to …]
FMCW configurations			
FMCW_SUPPORTED	Comment/uncomment this macro, to enable/disable FMCW support	-	N.A.
BANDWIDTH_MHZ	The bandwidth range is defined as the difference between the minimum and maximum RF frequencies of a device (units in MHz)	200	[1 to 200]
CHIRP_TIME_USEC	Up-chirp time (units in µs) Note: Down-chirp time (100 µs) and chirp time delay (100 µs) are fixed	300	[50 to 3000]
MINIMUM_RANGE_CM	Exclude targets below this distance (units in cm)	90	[0 to 49900]
MAXIMUM_ RANGE_CM	Exclude targets beyond this distance (units in cm)	10000	[10 to 50000]
RANGE_DETECTION_THRESHOLD	FFT spectrum threshold to detect a target in FMCW mode	100	[0 to 1000]
Doppler configurations			
DOPPLER_SUPPORTED	Comment/uncomment this macro, to enable/disable FMCW support	-	N.A.
DOPPLER_SAMPLING_FREQ_HZ	Sampling frequency (units in Hz)	20000	[0 to 100000]
MINIMUM_SPEED_KMH	Filter out targets below this speed (units in km/h)	0	[0 to 3.5]
MAXIMUM_SPEED_KMH	Filter out targets above this speed (units in km/h)	4	[0.1 to 20]
SPEED_DETECTION_THRESHOLD	FFT spectrum threshold to detect a target in Doppler	50	[0 to 1000]

5 Algorithm description

The key feature of the Position2Go is to track human targets in real time. In this section, the main principles of the implemented algorithms for Position2Go – e.g., range Doppler, tracking – are briefly described.

5.1 Algorithm overview

The algorithm running on the XMC™ microcontroller computes the targets' parameters and extracts the required information from the targets: range, velocity and angle from the digitized raw data provided by the BGT radar sensor. A tracking algorithm is also implemented, which runs on the XMC™ MCU and enables continuous tracking of the targets. Figure 24 gives an overview of the Position2Go basic algorithm flow.

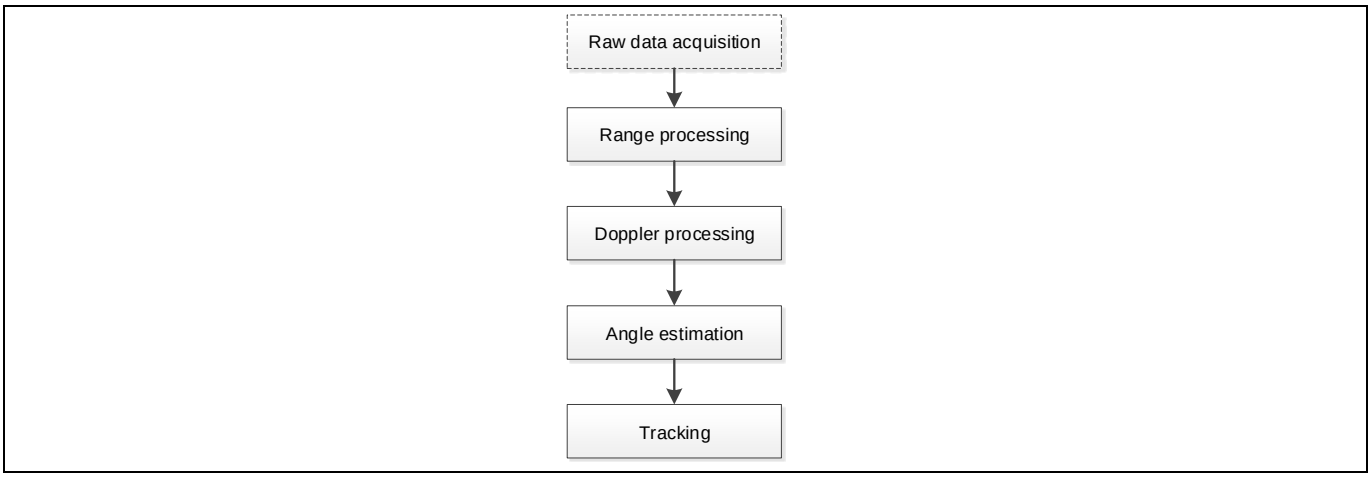

Figure 24 Position2Go basic algorithm flow

5.2 Radar algorithm

In this sub-section, the major blocks used in the implemented algorithm are described, as shown in Figure 25.

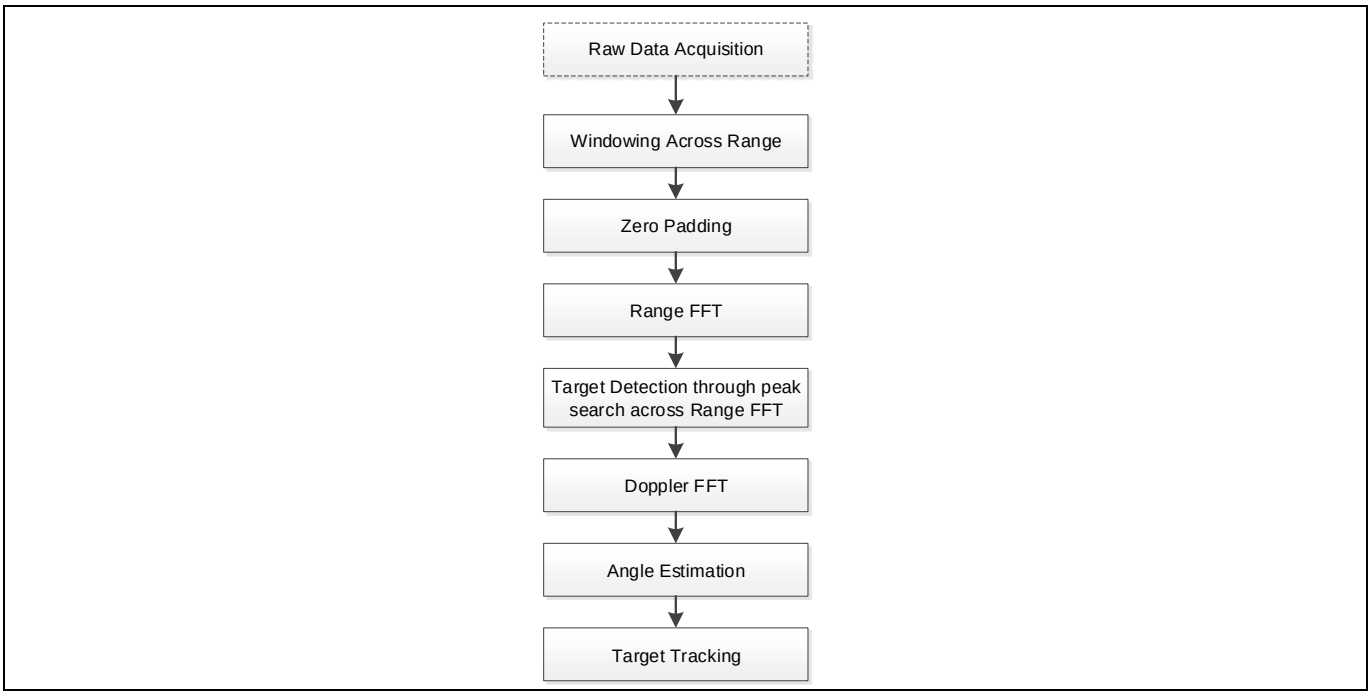

Figure 25 Block diagram of the radar algorithm implementation

- **Raw data acquisition** – The raw data is collected for every frame. The raw data is a 2D matrix comprising slow and fast time samples for every receiver.

- **Range processing** – After the pre-processed raw data for every receiver is available, the processing over fast time samples is carried out to extract the information about the target ranges. A Blackman window is applied across the range dimension, which helps in enhancing the signal SNR and suppressing the side lobes. Zero padding has been used to improve the received signal characteristic. After windowing and zero padding, a FFT is carried out over the range dimension. The range processing generates a range FFT image.

- **Target detection** – The range FFT is used to search for targets. A peak search is carried out over the range FFT and is used for detection of target ranges.

- **Doppler processing** – Range processing provides range FFT data over all the chirps. This range FFT data is processed over the slow time (chirps) to extract information about target velocity. A Chebyshev window is applied across the Doppler dimension. After zero padding across Doppler, a FFT is computed over the slow time to obtain a range Doppler map.

- **Angle estimation** – The Postion2Go sensor comprises 1 TX and 2 RX antennas. This configuration helps with estimation of the Angle of Arrival (AoA) of the targets. A phase monopulse angle estimation method is used to compute the AoA of the targets from the range Doppler map. Figure 26 gives more details about the monopulse estimation method used for AoA estimation.

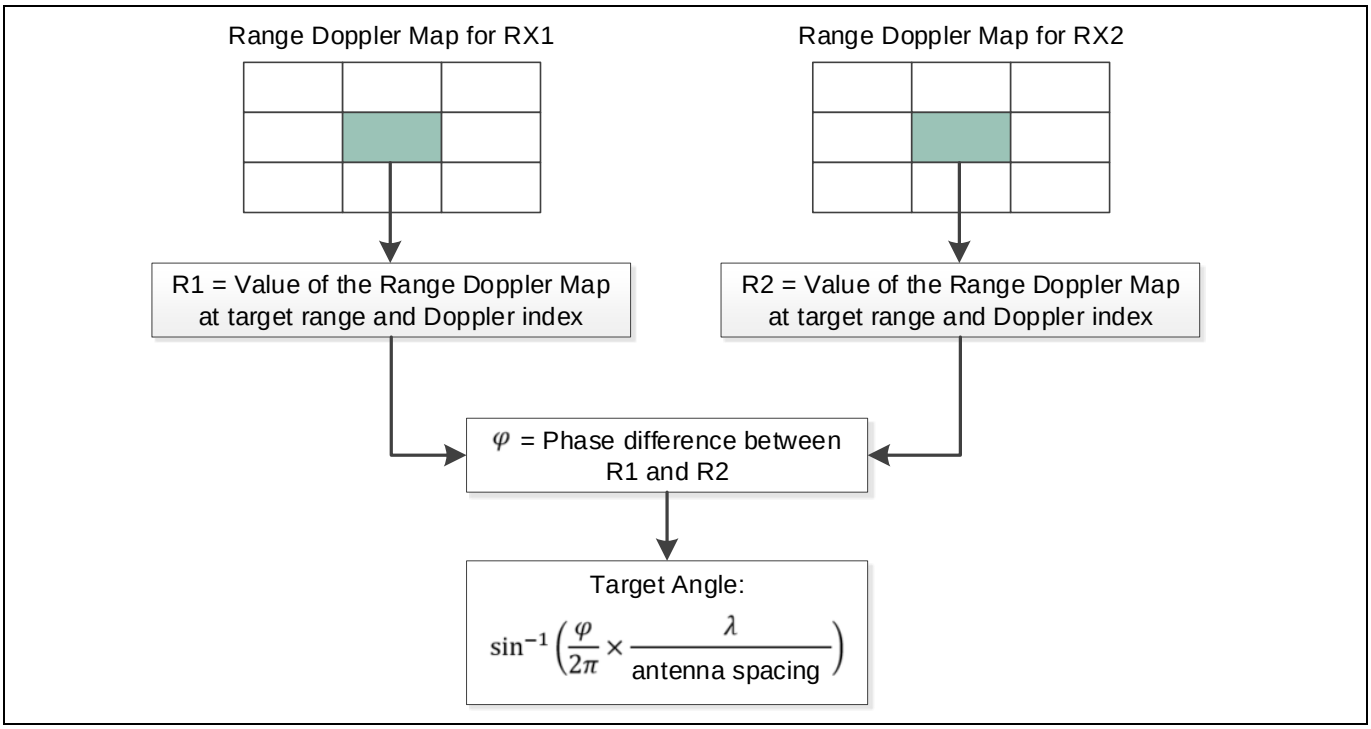

Figure 26 Monopulse angle estimation method flow

- **Tracking** – The parameters of the targets, namely range Doppler and angle, are obtained at the end of each processed frame. A Strongest Nearest Neighbor (SNN) approach is used to associate the target measurements with the corresponding tracks. An Alpha-Beta tracker is implemented to continuously track the moving targets.

6 Radar calibration

Calibration refers to the act of evaluating the radar as measurement equipment, before starting measurements. The Position2Go calibration data are temperature-dependent, so that re-calibration is required for (environment) temperature changes and therefore, the user should consider not calibrating but setting the minimum distance to 0.9 m.

6.1 Calibration data

For the Position2Go demo board, there are two types of calibration data:

- **ADC calibration data** – consists of raw ADC data for RX1 (I1 and Q1) and RX2 (I2 and Q2) calibration contents. ADC calibration is only applied for the first chirp. Each RX channel (I1, Q1, I2, Q2) requires 256 bytes as calibration data, independently of the used number of samples per chirp [32, 64, 128, 256]. For less than 256 bytes, the calibration data are completed with zeros, as shown in Figure 27.

- **Algorithm calibration data** – consists of the algorithm's offset calibration data, as follows:
  - Angle offset data – to manage RX1 vs. RX2 amplitude difference (units in degrees), coded in 2 bytes
  - Range offset data – to manage the TX vs. RX delay (units in cm), coded in 2 bytes.

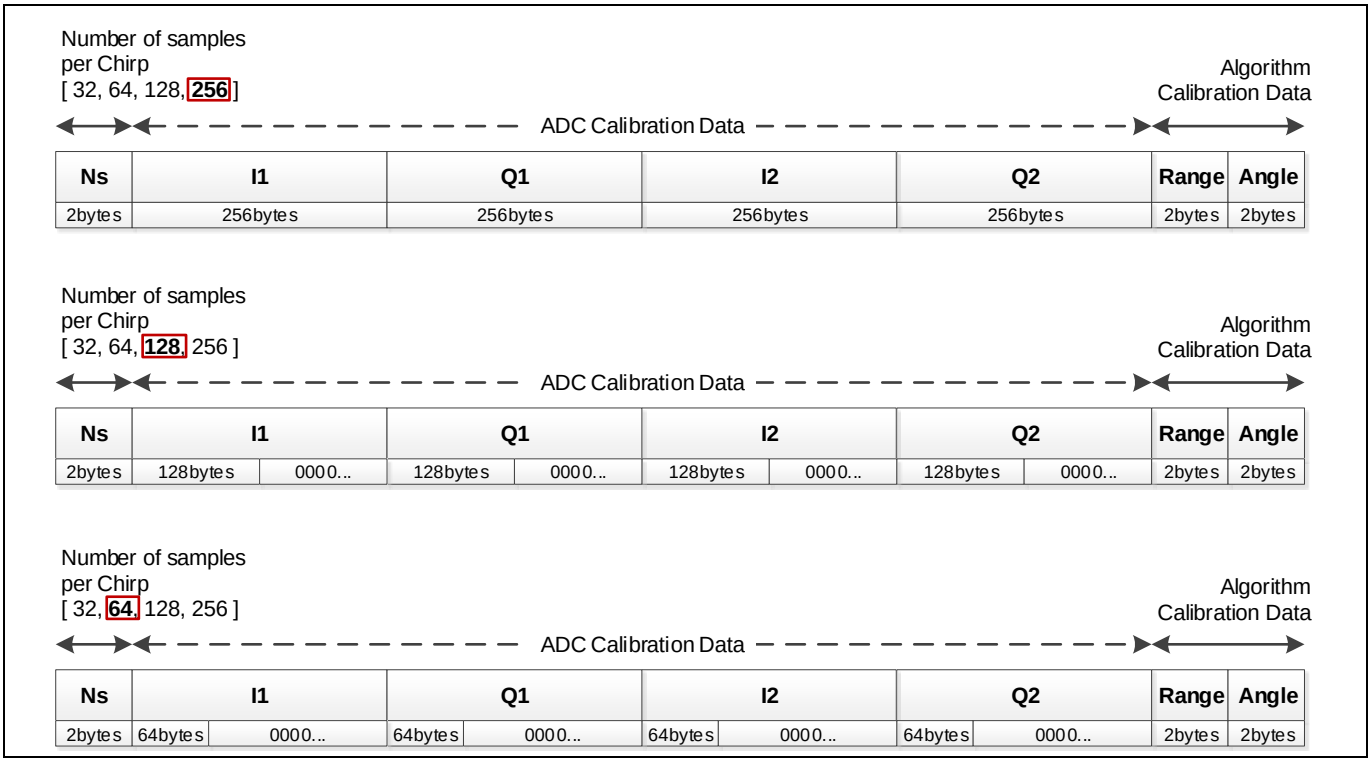

Figure 27 Calibration data format

6.2 Calibration routines

Two ADC calibration routines are described below:

- **Routine 1:** Start the Radar GUI, hold a 24 GHz absorber in front of the antennas and press the **Set (SRAM)** control button in the calibration section.

  Result: The reflections of the transmitting wave at the antenna's feeding point, the TX-to-RX leakage, and the BGT's VCO leakage are canceled. The unwanted environmental reflections that can interfere with the Position2Go remain. Calibration data are saved in the XMC4700 SRAM area.

- **Routine 2:** Remove any targets in front of the Position2Go antenna. Only the reflection source cannot be moved, as with a building. Then, use the Radar GUI and press the **Set (SRAM)** control in the calibration section.

  Result: Next to the RF impairments canceled in Routine 1, environmental reflections are also considered here. Calibration data are saved in the XMC4700 SRAM area.

For algorithm (range and angle) calibration, follow the routines detailed below:

- **Routine 1:** Measure range by another instrument, e.g. laser rangefinder. Then, fine-tune the measured range by Position2Go, by introducing a range offset through the Radar GUI, to have the estimated measured range. Finally, press the **Set (SRAM)** control in the calibration section to apply the required offset.

  Result: The introduced offset is correcting the measured value, and calibrating the Position2Go device to get accurate range values.

- **Routine 2:** Put the radar sensor under a flat surface with an estimated angle of 0 degrees, in the direction of the roof. Then, fine-tune the measured range by Position2Go, by introducing a range offset through the Radar GUI to have a measured angle of 0 degrees. Finally, press the **Set (SRAM)** control in the calibration section to apply the required offset.

  Result: Introduced offset corrects the measured value, and calibrates the Position2Go device to get accurate angle values.

6.3 Calibration target memories

Two types of target memory are used to manage (read, clear or set) calibration data for both calibration types (ADC, Algo):

- **FLASH** – A portion of Flash area emulated as an EEPROM memory is used as one calibration target memory. An area size of 2048 bytes is allocated at the bottom of the Flash for EEPROM emulation. Flash calibration data save and clear operations are permanent. Flash calibration read operation loads data into the SRAM area. All Position2Go boards are pre-calibrated with initial calibration data, and stored in Flash-emulated EEPROM (TX power level, RX LNA gain, chirp-time, bandwidth, etc...) defined in the *config.h* file.

- **SRAM** – This is used as another calibration target to save calibration temporarily. Setting and clearing SRAM calibration changes calibration data in SRAM only. It has no effect on the calibration data saved in Flash and is lost after powering down the device. An SRAM memory section of 2048 bytes in size contains the current calibration contents of the EEPROM emulated on the Flash memory.

6.4 Using Radar GUI for calibration

Calibration operations can be executed by the Radar GUI tool, referring to the **General Settings > Calibration** section. The Radar GUI offers two user modes: Regular User and Expert. User mode can be changed using the Settings menu as shown in Figure 28.

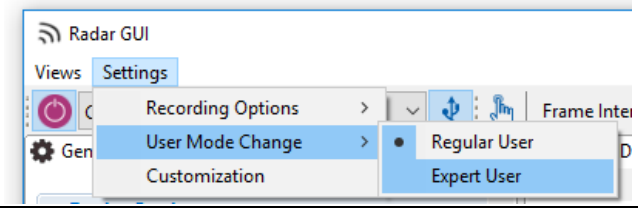

Figure 28 Regular vs. Expert User mode switch

Available control buttons in the calibration section depend on the Radar GUI mode. Figure 29 illustrates the calibration process through the Radar GUI control buttons, and the executed operations behind.

Each operation is detailed and explained in sections 6.4.1 and 6.4.2.

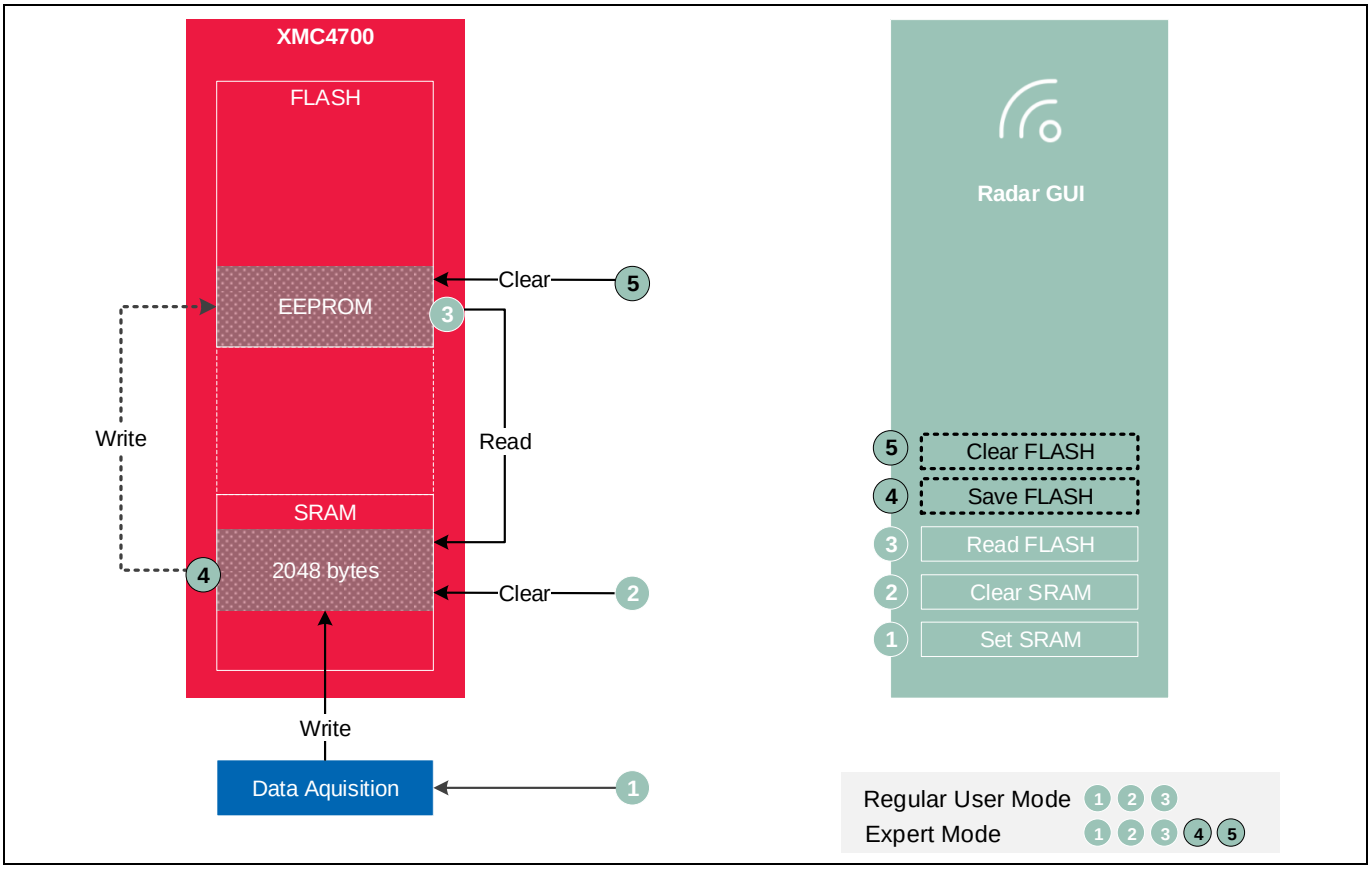

Figure 29 Calibration process through Radar GUI

6.4.1 Regular User mode calibration

The control buttons of the calibration section in Regular User mode are shown in Figure **30**.

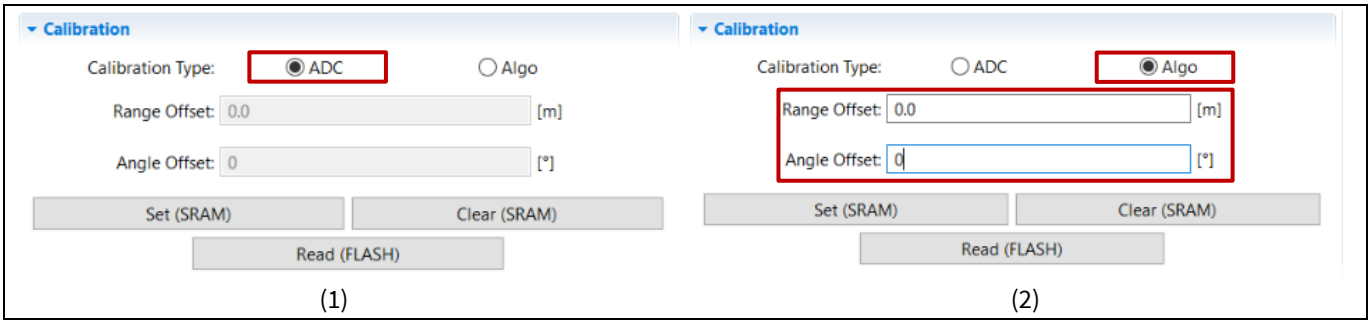

Figure 30 Regular User mode calibration options

In Regular User mode, the calibration process can be performed by **Set (SRAM)** for both calibration types (ADC, Algo). Additionally, the calibration data in the SRAM can be cleared by **Clear (SRAM)** and the initial calibration data stored in the Flash can be read into the SRAM by **Read (FLASH)**. Write to the Flash memory target is not allowed in this mode.

Default GUI calibration type value is set to ADC, where the calibration operation is applied to ADC raw data before starting data processing through the algorithm. The user can also switch the calibration type to Algo, using the available radio button under the calibration section, as shown in Figure **30**.

For algorithm calibration, two possible options are offered, which the user can manipulate:

- **Range Offset** – Measured range value can be fine-tuned by adding or subtracting a range offset, units in meters, based on another range reference, e.g. measured by laser rangefinder instrument.
- **Angle Offset** – Measured angle value can be fine-tuned by adding or subtracting an angle offset, units in degrees, based on another angle reference. e.g. 0 degree angle.

Once done, the user can execute the chosen calibration operation by clicking on the appropriate control button, depending on the selected user mode (Regular, Expert…).

6.4.2 Expert User mode calibration

The additional control buttons in Expert User mode, for both calibration types, are highlighted in **Figure** 31.

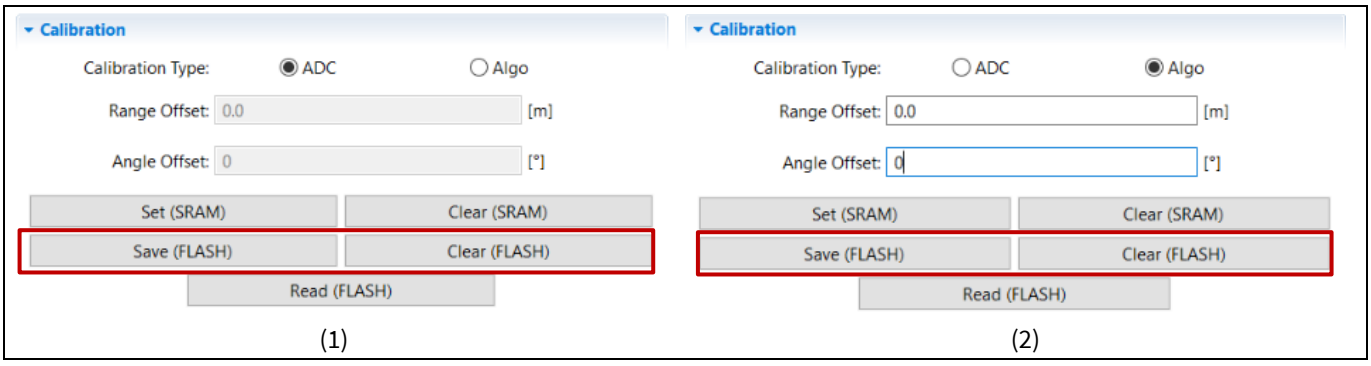

Figure 31 Expert User mode calibration options

Expert User mode enables all configurable options. Additional **Save (FLASH)** and **Clear (FLASH)** control buttons are only visible in Expert mode.

*Note: **Save (FLASH)** and **Clear (FLASH)** control buttons operations require password authentication every time the Radar GUI is started. Once erased or overwritten, initial Flash calibration data is lost with no possibility to back up.*

6.4.3 Change calibration default values

Radar GUI offers the possibility to change various device settings and chirp parameters, respecting their valid range values. Changing such values may affect the calibration process, for which reason the calibration status is either “Calibration Valid” or “Calibration NOT Valid”, as shown in **Figure** 32.

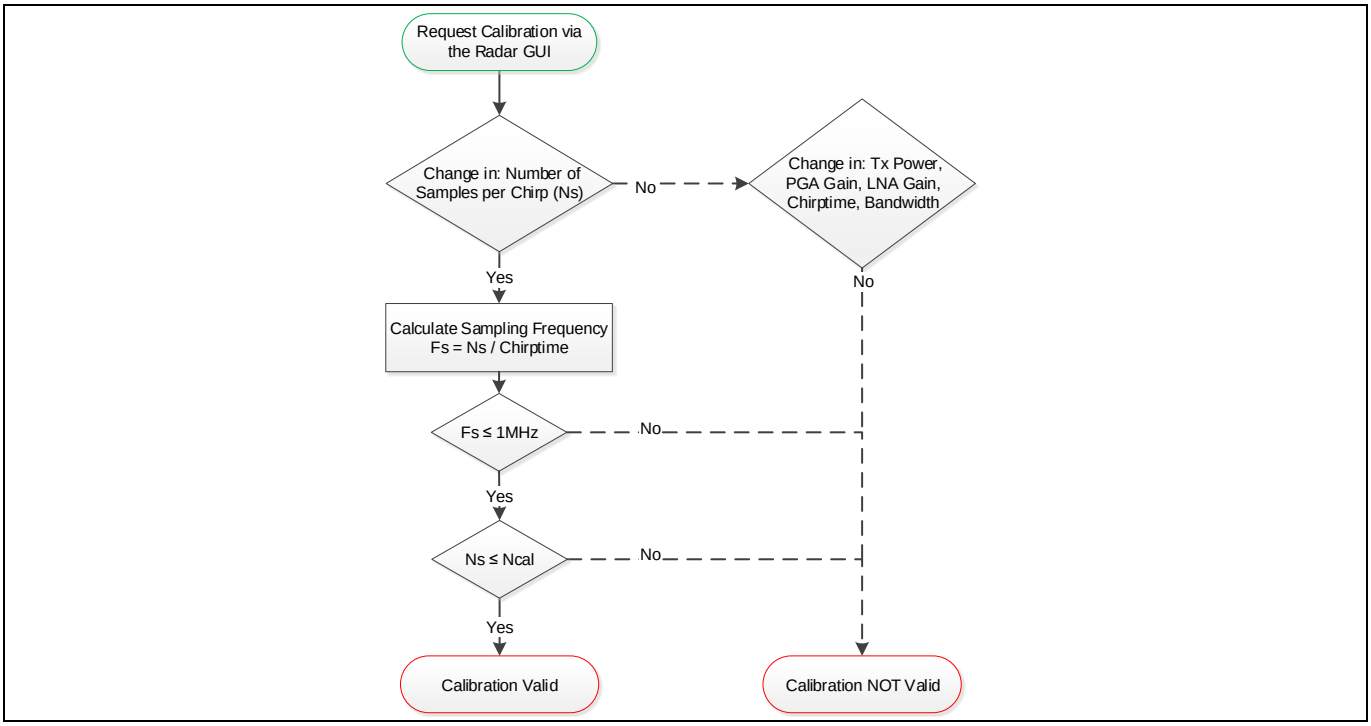

Figure 32 Change calibration default values

If the user changes the default values of the parameters TX power, PGA gain, LNA gain, chirp time or bandwidth, calibration is no longer valid and the status is changed to “Calibration NOT Valid”.

If user changes the “Samples per chirp (Ns)” value in the valid range of [32 to 256], the truth table in Table 5 must be considered to guarantee “Calibration Valid” status.

Table 5 Calibration truth table

Ns \ Ncal	256	128	64	32
256	Y	N	N	N
128	Y(1)	Y	N	N
64	Y(1)	Y(1)	Y	N
32	Y(1)	Y(1)	Y(1)	Y

N: Not valid
Y: Valid (without subsampling)

Y[(1)]: Valid (with subsampling)

Flashing radar application firmware does not overwrite the Flash calibration, which is preserved. **Figure** 33 shows the layout of XMC4700 memory mapping for Flash and SRAM.

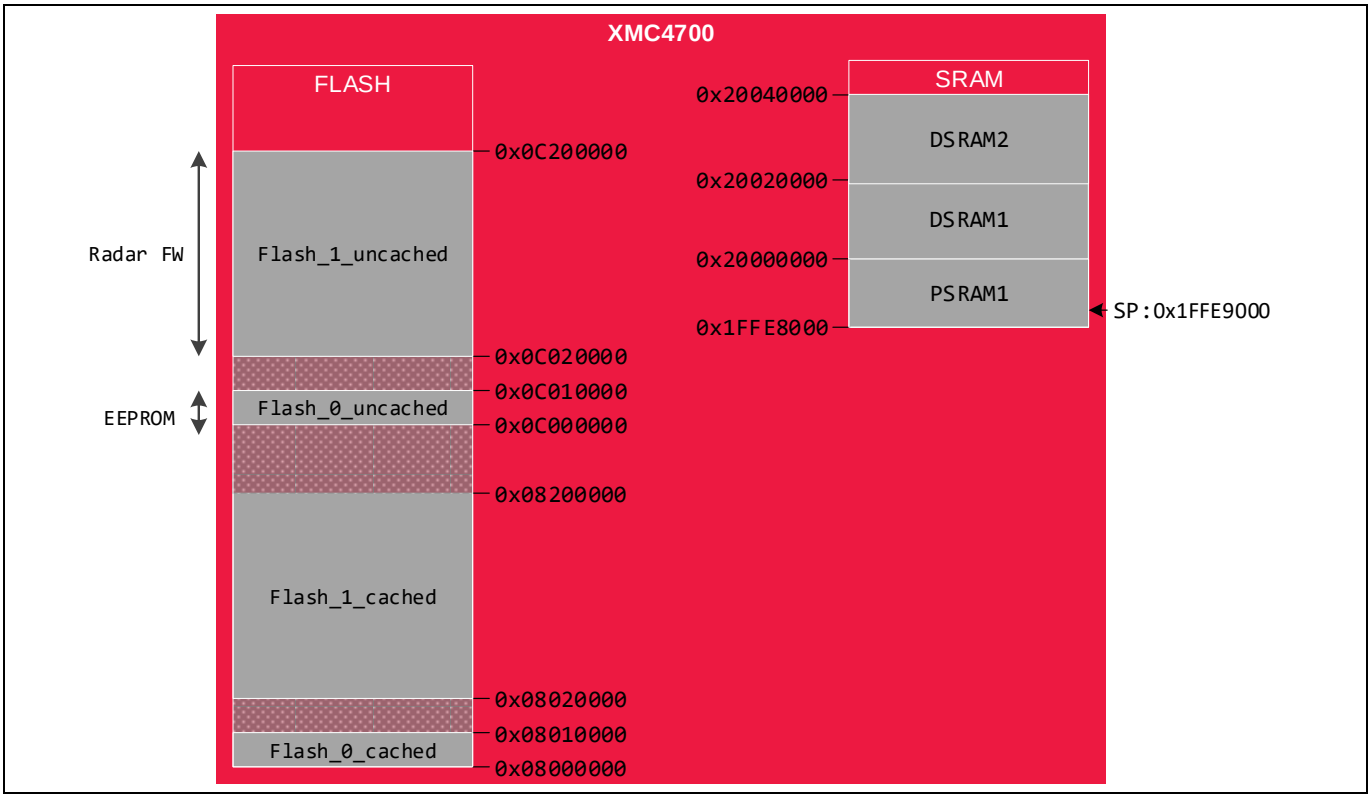

Figure 33 XMC4700 memory mapping

7 Authors

Radar Application Engineering Team, Business Line “Radio Frequency and Sensors”

8 References

[1] Infineon BGT24MTR12 – 24 GHz Radar MMIC – datasheet

[2] Infineon XMC4700 32-bit Arm® Cortex®-M4 Microcontroller – datasheet

[3] Infineon Application Note – AN305 – User's guide to BGT24MTR11

[4] Infineon Application Note – AN553 – Demo Position2Go

Revision history

Document revision	Date	Description of changes
1.00	2018-12-14	Initial version
1.10	2019-06-14	Moved Host Communication Library, Running Position2Go Application and Radar GUI section to 24GHz Radar Tools and Development Environment User Manual Removed Known Limitations section
1.20	2023-02-14	Miscellaneous document cleanup updates

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

Edition 2023-02-14

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2023 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

UM_1905_PL32_1905_111857

IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie") .

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, 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.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

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.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

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.