Distance estimation solution

using XENSIV™ KIT_CSK_BGT60TR13C 60 GHz radar

About this document

Scope and purpose

This application note demonstrates how to setup distance estimation from presence sensing algorithm using XENSIV™ KIT_CSK_BGT60TR13C kit, provided as part of the connected sensor kit (CSK) offering from Infineon.

Additionally, the document describes the required software and hardware, as well as how to set up and get started with distance estimation solution using XENSIV™ KIT_CSK_BGT60TR13C.

Intended audience

This document is intended for design engineers, technicians, and developers of electronic systems interested in building their own distance estimation solution for various consumer applications using the CYSBSYSKIT-DEV-01 kit and XENSIV™ BGT60TR13C radar sensor.

Table of contents

1 Introduction

1.1 FMCW radar systems

RADAR, which is an acronym for Radio Detection and Ranging, is a device that uses electromagnetic waves to obtain the range, angle or velocity of targets. Radar systems are used in a variety of applications. Some of them include satellite tracking, vital signs monitoring, presence detection, gestures recognition etc.

Modulated Radar which is also known as Frequency modulated continuous wave (FMCW) radar, is a type of Radar that can measure the range of the target using a technique known as Frequency Modulation. In Frequency Modulation, the frequency of the electromagnetic wave is linearly increased with time. This kind of signal whose frequency increases linearly with time is called a chirp. The FMCW system measures the instantaneous difference between the transmitted and reflected signal frequencies δf, which is directly proportional to the time difference δt of the reflected chirp. The time difference can be used to calculate the range of the target.

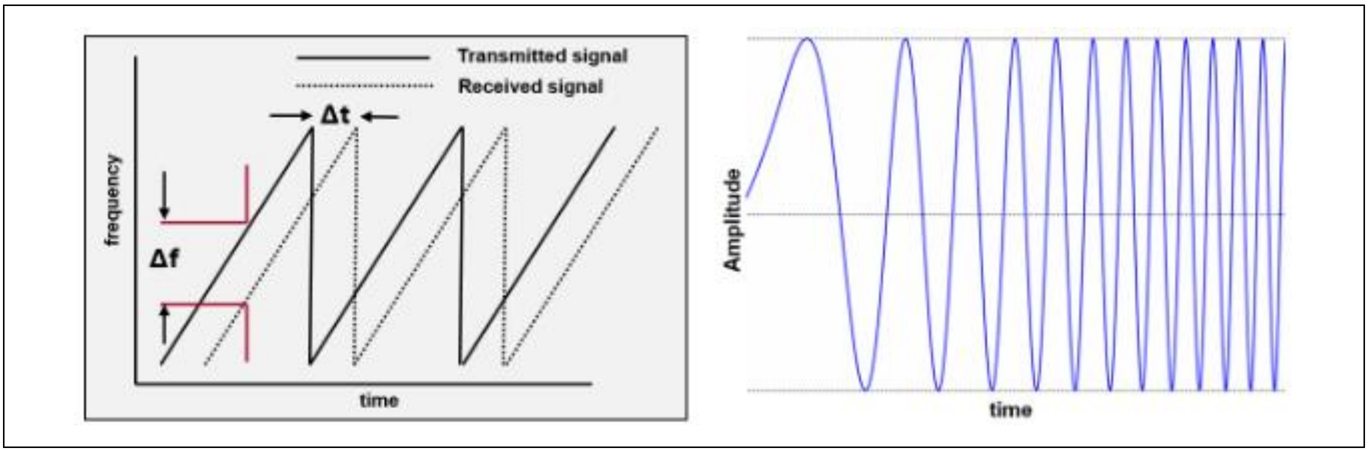

Figure 1 Chirp signal representation

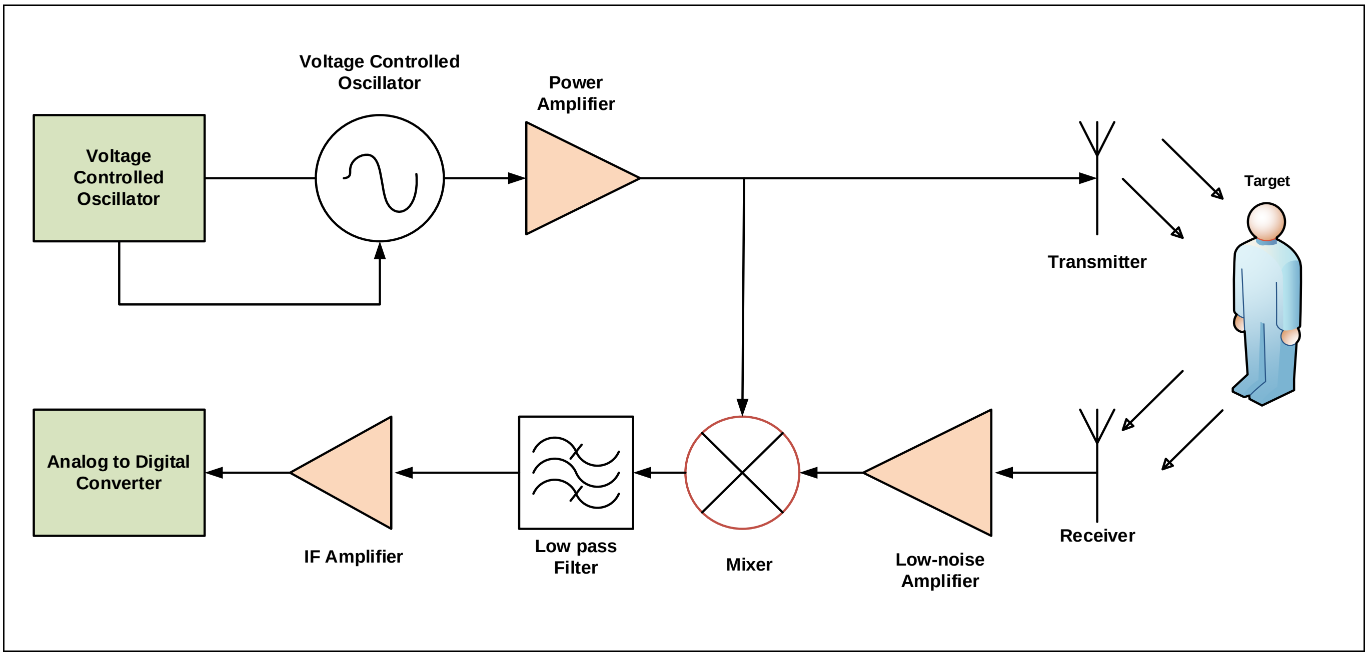

Figure 2 FMCW block diagram

1.2 Distance estimation in FMCW radar systems

A single chirp consists of a frequency ramp with a certain bandwidth and a certain chirp time. If a static single target reflects the radar signal back, the received signal will resemble the transmitted signal but will be weaker and time delayed.

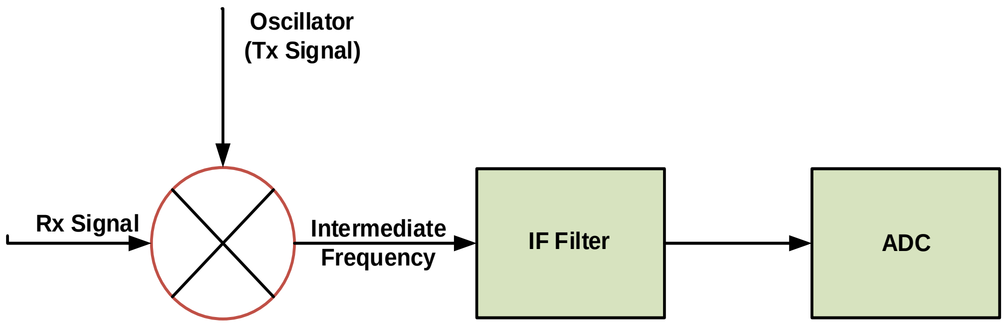

Figure 3 Radar demodulation scheme

- If the received signal is mixed with the local oscillator signal (Tx signal), the mixer output contains the sum frequency and the beat frequency of the two inputs. The sum frequency is typically filtered and only the intermediate frequency is processed further with intermediate frequency (IF) filters before it is digitized in an analog-to-digital converter (ADC).
- The propagation delay due to radar wave propagation with the speed of light c0 to a target at a range R (and back) is: $t_{delay} = \frac{2R}{c0}$
- Together with the linear chirp $df/dt = BW\ tchirp$, the resulting beat frequency can be calculated with the formula $fbeat = tdelay\ (df/dt) = 2R(BW)/\ c0 \cdot\ tchirp$
- The target range in dependence on the beat frequency is then $R = (c0 \cdot tchirp/2\ BW)\ fbeat.$ From the general Fourier transform (FT), it is known that the lowest beat frequency which can be detected is one over the measurement time. For short time delays, the lowest detectable beat frequency can be simplified to $1/tchirp$
- Thus, a single range bin of the FT is $c0/2BW$. The physical meaning of this is that it corresponds to the range resolution.

2 Hardware and software requirements

2.1 Hardware requirements

The application note is implemented and tested on below-mentioned boards. However, the code can be migrated to use other combinations.

- Rapid IoT Connect Developer (CYSBSYSKIT-DEV-01)
- XENSIV™ BGT60TR13C Wing (EVAL_BGT60TR13C_WING)

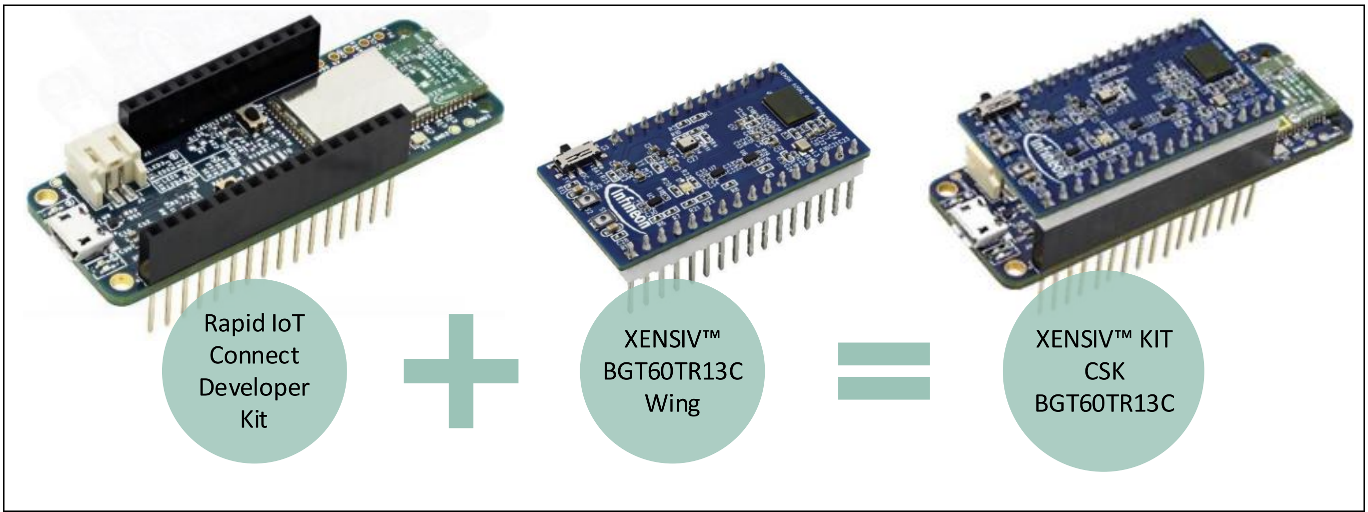

Figure 4 XENSIV™ KIT CSK BGT60TR13C

2.1.1 Rapid IoT Connect Developer Kit

The Rapid IoT Connect Developer Kit (CYSBSYSKIT-DEV-01) allows for evaluation of the Rapid IoT Connect module (CYSBSYS-RP01) on a standard Feather form factor. The CYSBSYS-RP01 Rapid IoT Connect module is a turnkey module that enables secure, scalable, and reliable compute and connect.

The Rapid IoT Connect Developer Kit carries a CYSBSYS-RP01 Rapid IoT connect system-on-module (SoM). The Rapid IoT connect SoM includes a PSOC™ 6 MCU device, a CYW43012 single-chip radio, on-board crystals, oscillators, chip antenna, and passive components.

Figure 5 CYSBSYSKIT-DEV-01

Key features:

- CYSBSYS-RP01 module
- Support of up to 2 MB Flash and 1 MB SRAM
- 512 Mbit external Quad SPI NOR Flash that provides a fast, expandable memory for data and code
- KitProg3 on-board SWD programmer/debugger, USB-UART, and USB-I2C bridge functionality
- Battery connector, charging IC, and charging indicator LED
- One KitProg3 mode button, one KitProg3 status LED, and one KitProg3 power LED
- 16 KB of Emulated EEPROM
- Feather compatible pin header
- Delivers dual-cores, with a 150 MHz Arm® Cortex®-M4 as the primary application processor and a 100 MHz Arm® Cortex®-M0+ as the secondary processor for low-power operations.
- Supports Full-Speed USB, a Quad-SPI interface, 13 serial communication blocks, 7 programmable analog blocks, and 56 programmable digital blocks.

2.1.2 XENSIV™ BGT60TR13C Wing

The XENSIV™ BGT60TR13C MMIC is a 60 GHz radar sensor with integrated antennas and comes with one transmitting and three receiving antennas. BGT60TR13C MMIC enables ultra-wide bandwidth FMCW operation. It is equipped with an integrated Finite-State Machine (FSM). With the aid of the FSM, BGT60TR13C can perform FMCW frequency sweeps (so-called chirps), data acquisition as well as storing of samples into the internal FIFO memory autonomously.

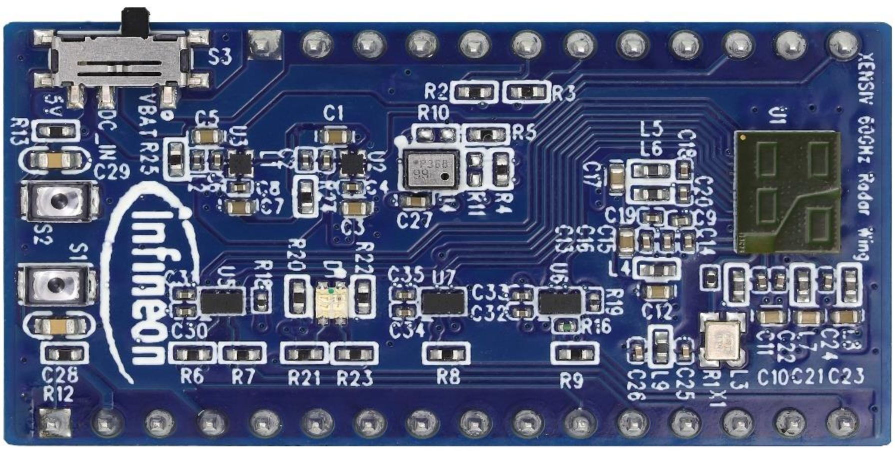

Figure 6 XENSIV™ BGT60TR13C Wing

BGT60TR13C features:

- It has ultra-wide bandwidth of 5.5 GHz a very low range resolution down to ~3 cm.
- Higher Doppler velocity achieved with a ramp-up speed of 400 MHz/µs.
- High Signal-to-Noise Ratio (SNR) ensures detection of people up to 10 m, front facing towards the sensor, while high sensitivity allows detection of movements down to sub-millimeter. Via the very commonly used SPI (Serial Peripheral Interface).
- 60 GHz radar sensor for FMCW operation.
- Antenna-in-package.
- Optimized power modes for low-power operation.

2.2 Software requirement

- ModusToolbox™
- Board support package
- C language
- Serial Terminal (Tera Term, ModusToolbox™ Terminal, etc.)

3 Getting Started

This section provides a walkthrough to create a presence detection application using the Eclipse IDE for ModusToolbox™, selecting a BSP. It also covers how to build and program using the IDE and to see the output in the serial terminal.

3.1 Creating presence detection application

To launch the Eclipse IDE:

- On Windows, select the Eclipse IDE for ModusToolbox™ item from the Start menu.
- For other operating systems, run the "modustoolbox" executable file.

Note: *When launching the Eclipse IDE, it provides an option to select the workspace location on your machine. This location is used by the IDE for creating and storing the files as part of application creation for a particular platform.*
The default workspace location is a folder called "mtw" in your home directory. You may add additional folders under the "mtw" folder or to choose any other location for each workspace.

- Click the New Application link in the Eclipse IDE Quick Panel as shown in the Figure 7.

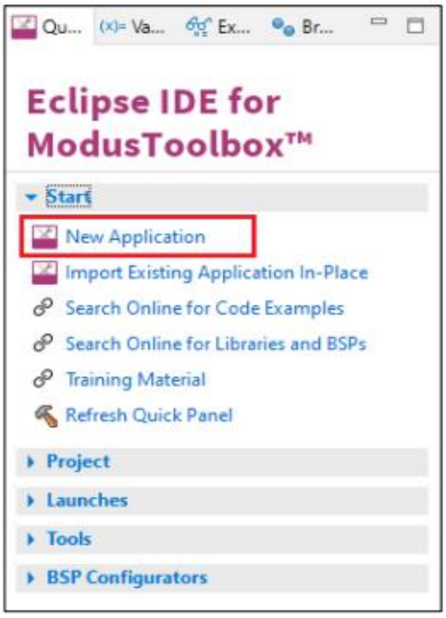

Figure 7 New application creation

- This launches the Project Creator tool to select the board support package (BSP) and the several applications available for a particular board.
- Select **CYSBSYSKIT-DEV-01** BSP from the **PSOC™ 6 BSPs** dropdown as shown in the Figure 8 and click next.

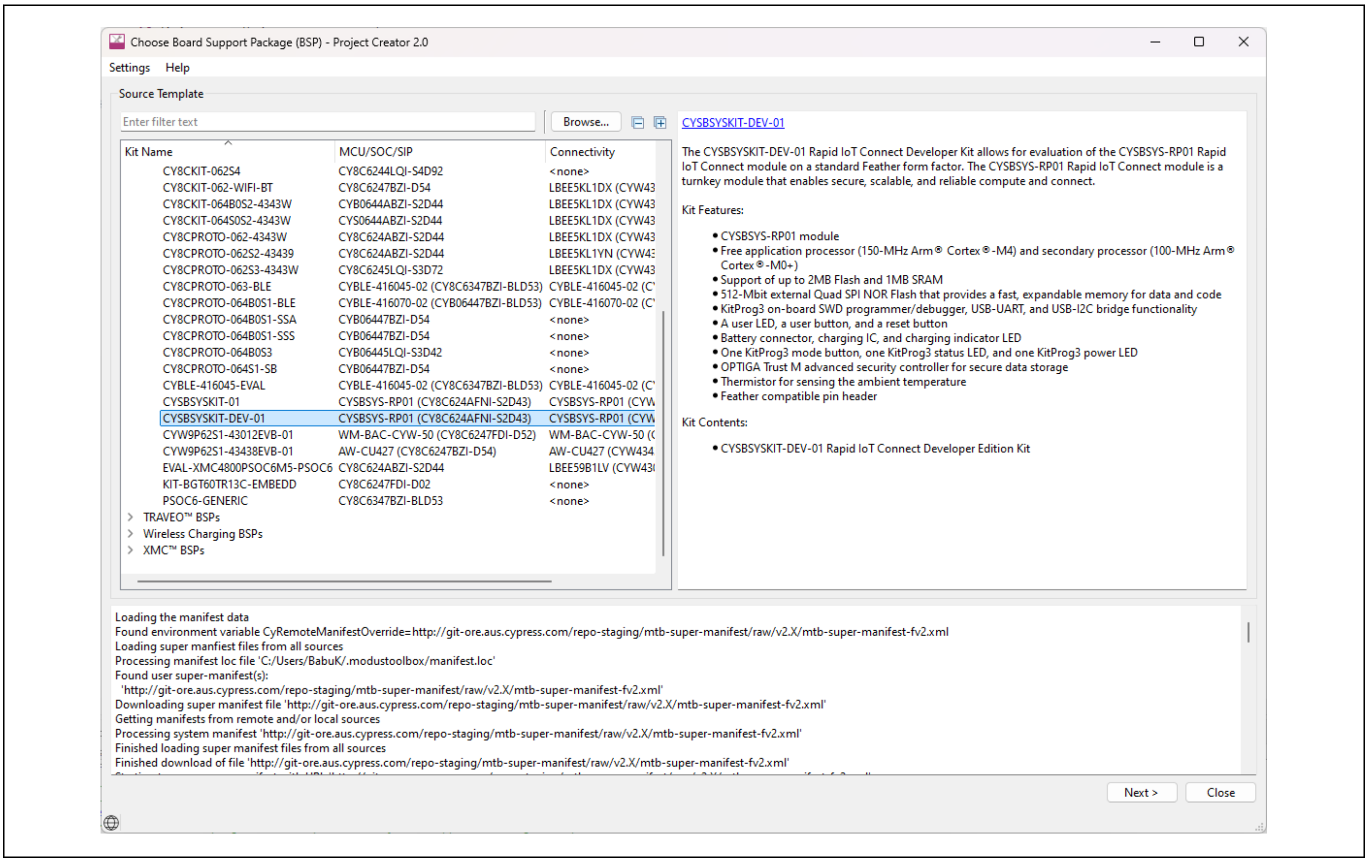

Figure 8 Selecting the BSP

- Now select the **Human Presence Detection** application from **Sensing** drop down as shown in the Figure 9 and click **Create**.

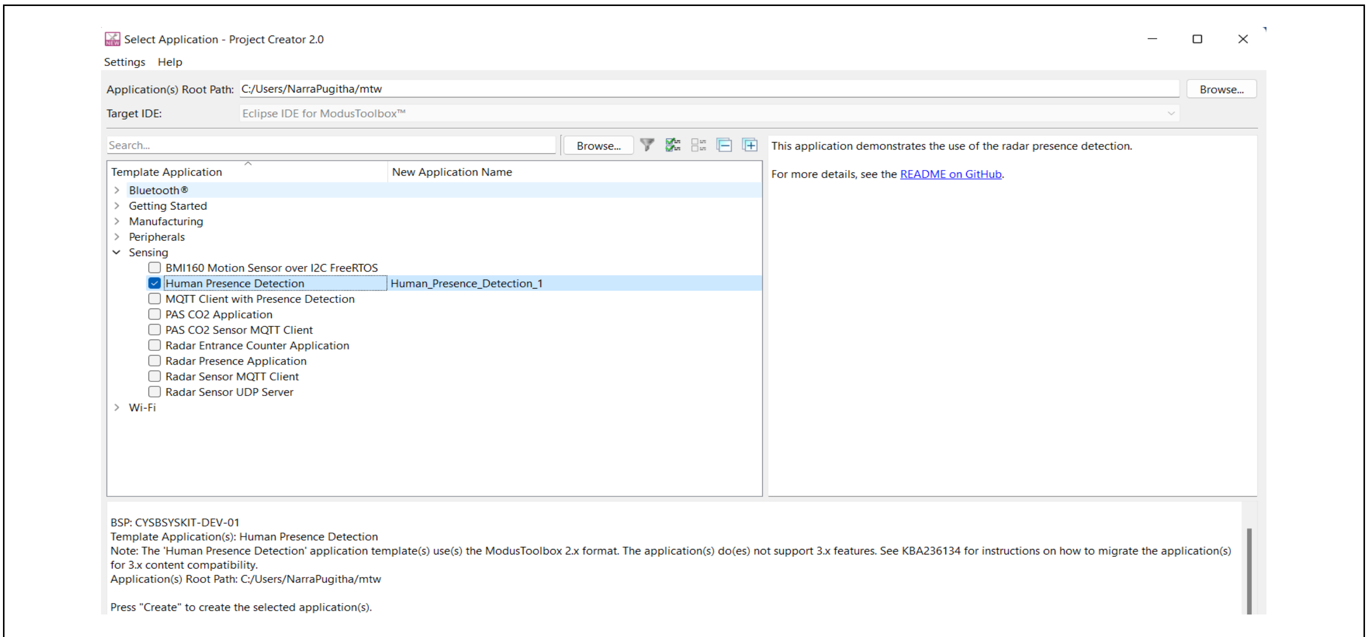

Figure 9 Selecting the application

When complete, the Project Creator tool closes automatically. In the IDE, a message displays about importing the project.

3.1.1 Build application

After loading the application, build it to generate the necessary files. Select a project. Then, in the Quick Panel, click the Build Application link shown in the Figure 10.

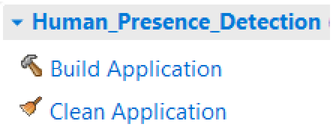

Figure 10 Build application

Messages display in the console, indicating whether the build was successful or not.

3.1.2 Program application

In the Project Explorer, select the desired project. Then, in the Quick Panel, click the Program (KitProg3_MiniProg4) link as shown in the Figure 11.

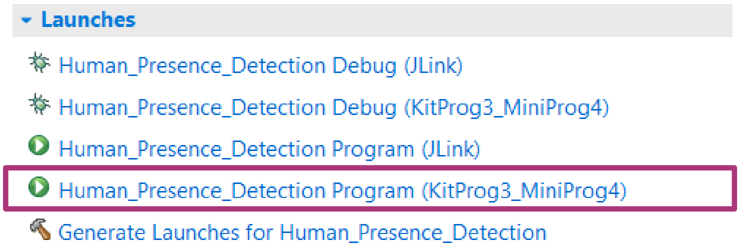

Figure 11 Program application

Open ModuxToolbox™ terminal from toolbar (Ctrl + Alt + Shift + T) and select a COM port where the board is connected (not the MiniProg4 port). Set the serial port parameters to 8N1 and 115200 baud. After programming, the application starts automatically. Confirm that "presence detection using XENSIV™ 60-GHz radar" is displayed on the UART terminal.

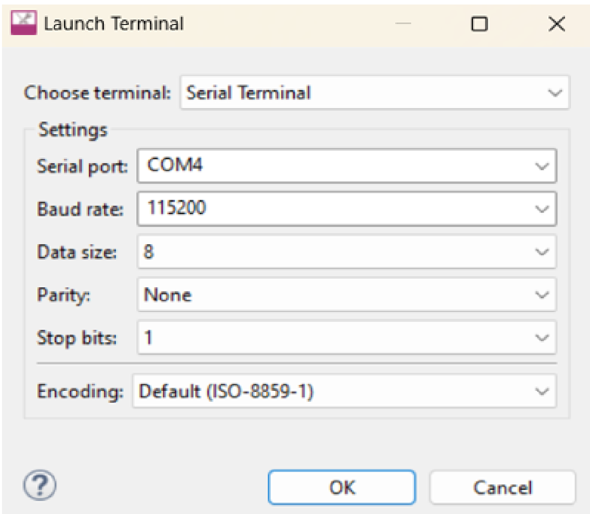

Figure 12 Terminal settings

```
COM4    <Closed> Human_Presence_Detection ModusShell
[INFO] macro presence 4 1.303445
[INFO] macro presence 2 0.651723
[INFO] macro presence 1 0.325861
[INFO] macro presence 3 0.977584
[INFO] macro presence 1 0.325861
[INFO] macro presence 2 0.651723
[INFO] macro presence 1 0.325861
[INFO] micro presence 1 835339
[INFO] macro presence 1 0.325861
[INFO] macro presence 5 1.629307
[INFO] macro presence 1 0.325861
[INFO] macro presence 4 1.303445
[INFO] macro presence 1 0.325861
[INFO] macro presence 4 1.303445
[INFO] macro presence 1 0.325861
```

Figure 13 Output in ModusToolbox™ Terminal

4 Parameters configuration

User can configure the application parameters using the options provided on the terminal as follows:

- Press 'enter' key to switch from work to settings mode.
- Type help and press 'enter' key to see a list of configurable parameters as shown in the Figure 14.

```
[INFO] macro presence 1 0.325861
[INFO] macro presence 4 1.303445
[INFO] macro presence 1 0.325861
[INFO] macro presence 4 1.303445
[INFO] micro presence 1 3421145
[INFO] macro presence 4 1.303445
[INFO] macro presence 1 0.325861

Enter setting mode and stop processing
> help

help - Lists all the registered commands
set_max_range <value> - Sets the max range for presence algorithm in meters. Range <0.66-5.0>
set_macro_threshold <value> - Sets macro threshold for presence algorithm. Range <0.5-2.0>
set_micro_threshold <value> - Sets micro threshold for presence algorithm. Range <0.2-50.0>
bandpass_filter <enable|disable> - Enabling/disabling bandpass filter
decimation_filter <enable|disable> - Enabling/disabling decimation filter
set mode <macro only|micro only|micro if macro|micro and macro> - Chooses work mode
```

Figure 14 Terminal output for parameters configuration

The user can configure the following parameters:

- Macro only
- Micro only
- Macro and micro
- Micro if macro
- The minimum and maximum range detection can be configured
- The sensitivity level of each mode can be adjusted by changing the thresholds levels.
- Enabling/ Disabling bandpass filter and decimation filter

Table 1 Console output description

Parameters	Event type	Description
Radar state	Macro or Micro Presence	Presence event detected
Range bin	2	Maximum range bin
Time stamp	‘4298’	Relative time in ms

Note: Time stamp is relative to the boot time. This means when application first boot, the time counting starts from 0 ms.

Table 2 Parameters configuration

Key	Default value	Valid values
Set_max_range (m)	2.0	0.66 – 5.0
Set_max_threshold	0.5	0.5 – 2.0
Set_micro_threshold	12.5	0.2 – 99.0
bandpass_filter	disable	enable/disable
Decimation_filter	disable	enable/disable
set_mode	MICRO_IF_MACRO	MACRO_ONLY, MICRO_ONLY, MICRO_IF_MACRO, MICRO_AND_MACRO

Note: *Macro and Micro threshold parameters can be adjusted to achieve different levels of sensitivity. Table 3 summarizes three different levels (for instance high means solution being more sensitive to stationary peoples).*

Table 3 Sensitivity level with the corresponding threshold settings

Sensitivity	Macro_threshold_value	Micro_threshold_value
High	0.5	12.5
Medium	1.0	25
Low	2.0	50

5 Operation

5.1 Architecture of the application

This application uses a modular approach to build a presence application combining radar driver and presence algorithm library. The components used in this application are shown in Figure 15.

Presence Detection library detects both macro and micro movements in a configurable range using the data acquired by radar sensor. It uses the Sensor-DSP library that provides signal processing functions required to support the implementation of presence detection algorithm.

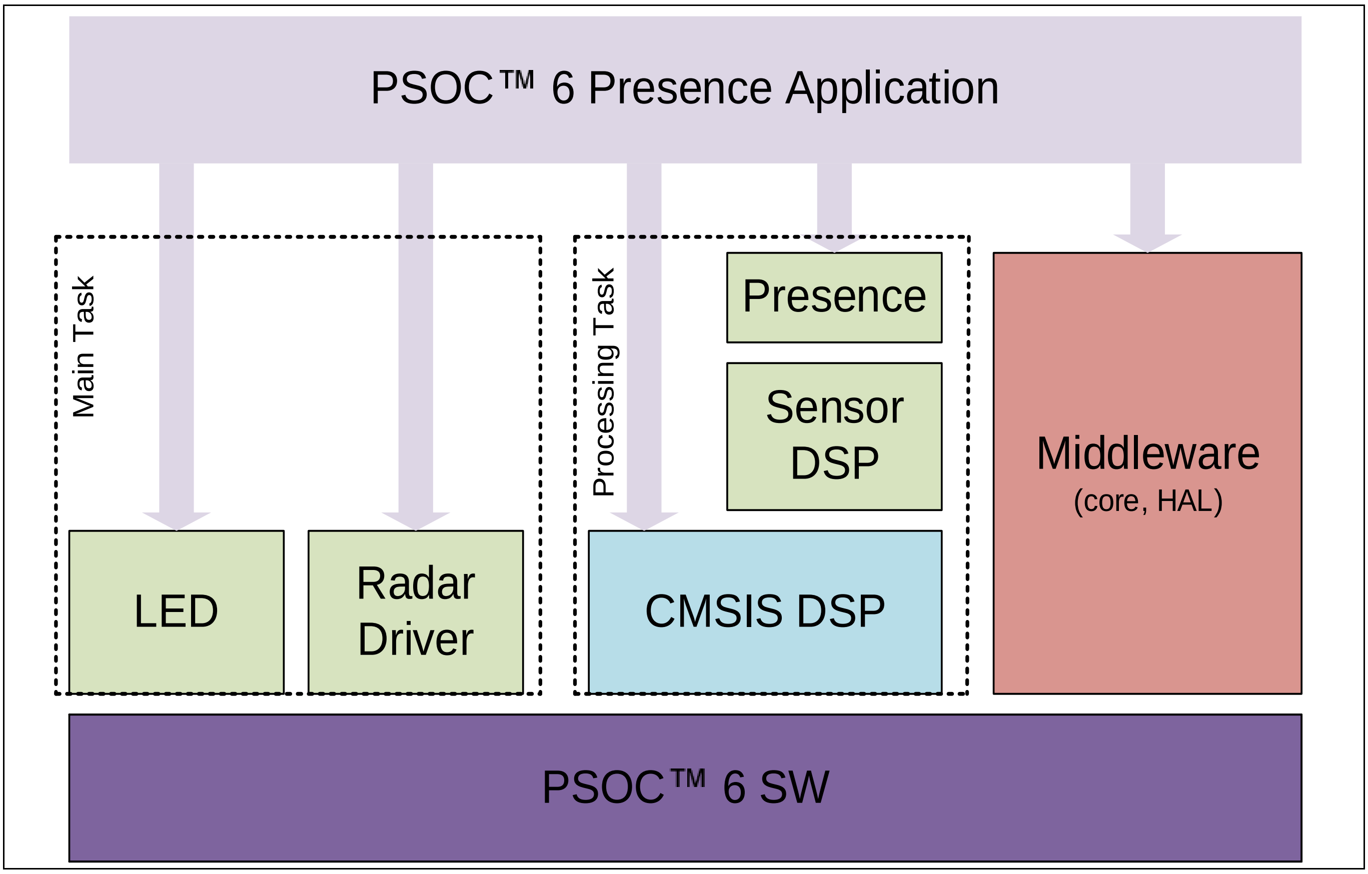

Figure 15 Application architecture

The **Sensor DSP** library provides signal processing functions required to support the implementation of different sensor applications, i.e., radar gesture recognition, vital sensing and presence detection. The Sensor-DSP library builds on top of the standard ARM CMSIS library. It supports Range and Doppler FFT, FFT windowing related functions, Detection related functions, Angle estimation related functions.

5.2 Application execution

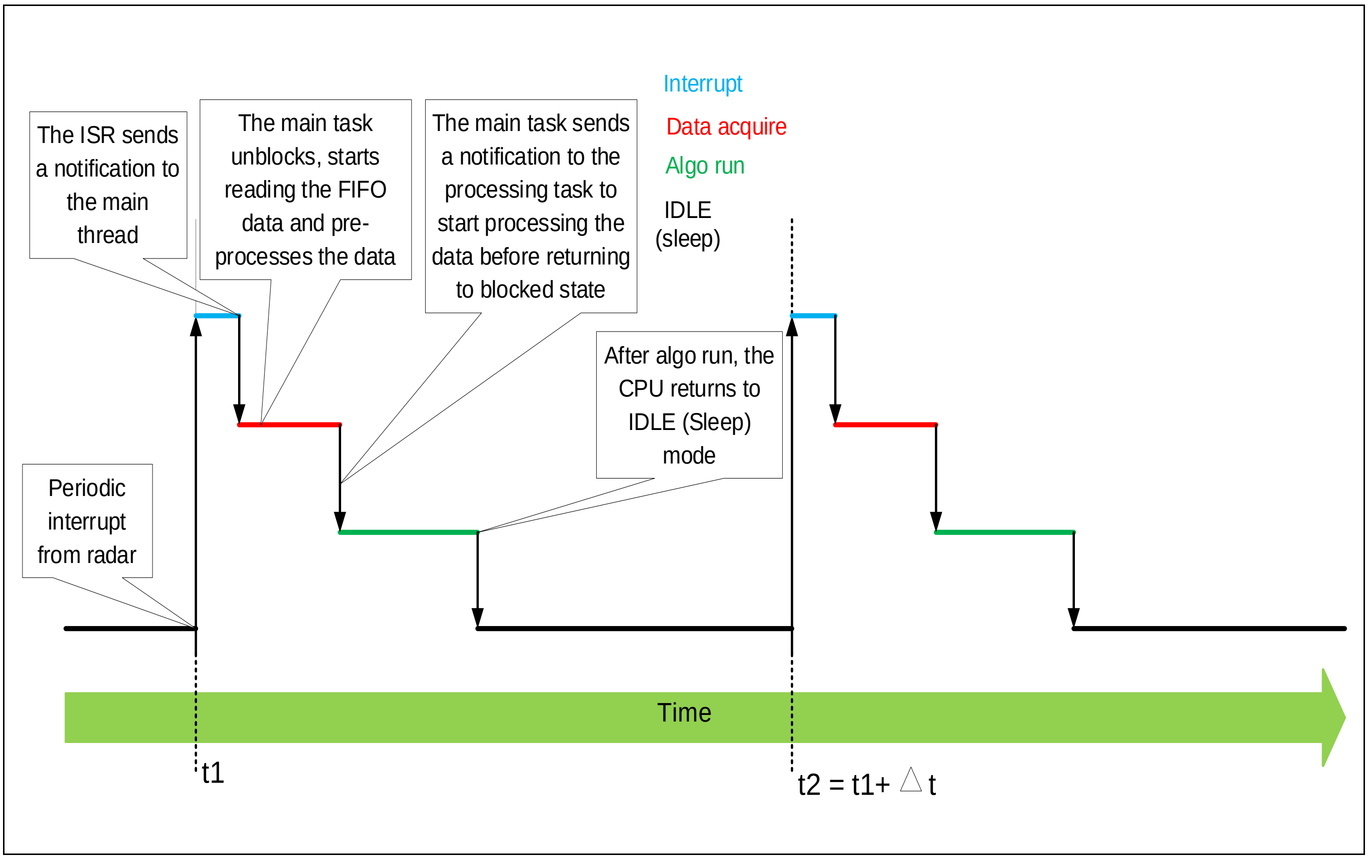

Figure 16 Application execution

- The application starts with the initialization of BSP, creates a FreeRTOS™ main task, initialize retarget-IO library to use the debug UART port and starts the FreeRTOS™ scheduler.
- The main task Creates a timer to toggle user LED, create the processing RTOS task, initializes the hardware interface to the sensor and LEDs, initializes the radar device, waits for interrupt from radar device indicating availability of data, reads the data, converts it to floating point and notifies the processing task.
- The data processing task initializes the presence sensing library and register an event call-back.
- It creates a console task to handle parameter configuration for the library, receives notification from main task and executes the presence algorithm and provides the result on terminal and LEDs
- For CYSBSYSKIT-DEV-01, the radar task is suspended if the radar wing board is not connected to the feather kit. The sensor initialization process is indicated by blinking the red LED (CYBSP_USER_LED). The red LED (CYBSP_USER_LED) on CYSBSYSKIT-DEV-01 keeps blinking when the system is operational (ready state).

6 Important Terms and Concepts

- Chirp – A chirp is a signal whose frequency changes (i.e., increases or decreases) with time at a constant rate df/dt. During the transmission of a chirp, the receiving antenna signal is actively sampled into a digital waveform by an ADC (analog-to-digital converter) for further processing
- Number of samples per chirp – Nsamples represents the number of samples acquired during each chirp of a frame. Nsamples is obtained from maximum range and range resolution, but also depends on pulse repetition time and consequently the maximum speed
- Maximum range (in meters) – The bins of the range transform represent the range between 0 and this value.
- Range resolution (in meters) – Minimum range separation needed to distinguish two targets of same velocity
- Total bandwidth – Total bandwidth of the chirp or df is obtained from range resolution. If ΔR is the desired range resolution and c is the speed of light, then: $df=c/2\Delta R$, where c=299792458m/s denotes the speed of light.
- Start frequency Hz – represents the chirp start frequency
- End frequency Hz – represents the chirp end frequency.
- Sample rate Hz – The ADC sampling rate, defines how fast the samples are acquired
- Chirp repetition time (in seconds) – The PRT or chirp repetition time is the time interval between two consecutive chirps
- Number chirps per frame or Nc – A consecutive number of chirps composes a measurement frame together with a frame end delay.
- Frame repetition time (in seconds) – The period also called frame time is the time interval between the beginnings of two consecutive frames
- Micro-motions – Detecting small movements like gestures or small head movements in a typical smart home environment for instance while working on laptop/keyboard. Micro-motion also includes detection of Stationary humans (normally breathing and blinking eyes) in sitting or standing positions (in line of sight).
- Macro-motions – Detecting major movements into or through the field of view (Motion Detection).

7 Distance estimation in presence detection application

By default, the application will only print the parameters stated in Table 1, but the user can output the estimated distance at which the target is present by following the steps mentioned below:

- Declare a range variable of type float say float R in presence_detection_cb () function present in main.c file.
- float R (in meters) = ((xensiv_radar_presence_get_bin_length(handle)) *(event->range_bin));
- The above-mentioned API can be used to print the range in human presence detection application.
- Theoretically xensiv_radar_presence_get_bin_length () function gets the range resolution i.e., (c) / (2 * bandwidth_hz * fft_size /num_samples_per_chirp) which is approximately equal to the value 0.325 and range_bin is the value at which the target is present.

  Please note that:

  - Speed of light in meters per second C: 299792458
  - bandwidth_hz: 460000000
  - fft_size (human presence detection application): 128
  - num_samples_per_chirp: 128

Example
The value of range bin is 2, means the radar detected the target in that particular range bin then range = 0.325*2= 0.66 m. The target is at the range 0.66 m.

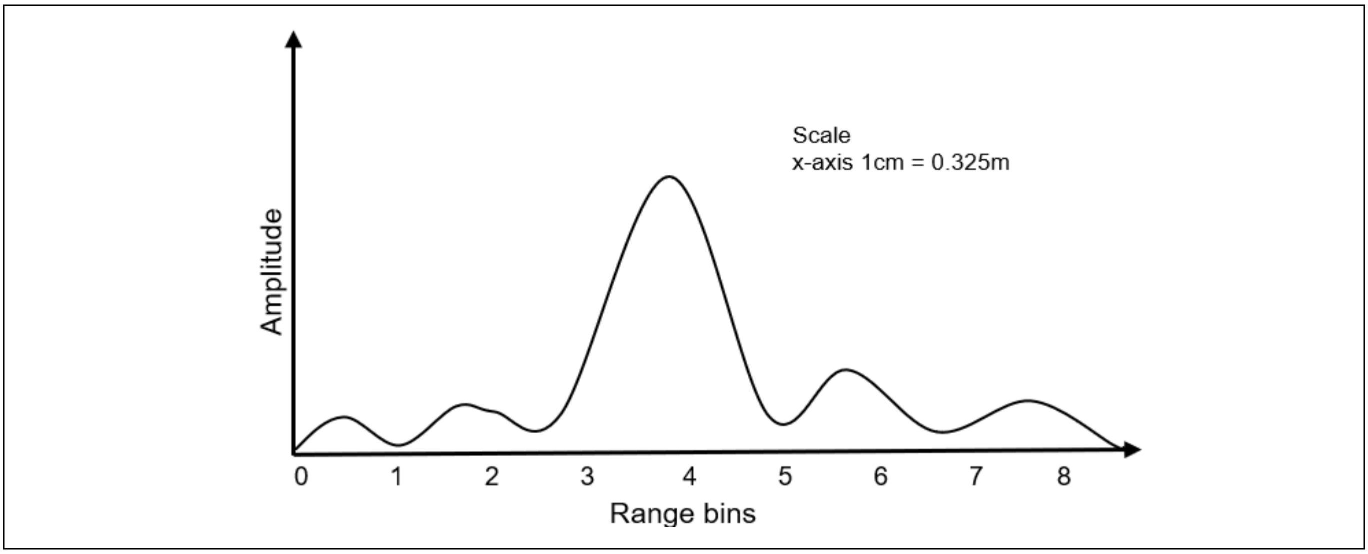

Figure 17 Range bins

Note: The distance is computed by the existing presence detection algorithm.

Code Listing 1 Presence detection callback function

```
void presence_detection_cb(xensiv_radar_presence_handle_t handle,
                           const xensiv_radar_presence_event_t *event,
                           void *data)
{
    (void)handle;
    (void)data;

    float R = ((xensiv_radar_presence_get_bin_length(handle)) * (event->range_bin)); // Range in meters

    switch (event->state)
    {
    case XENSIV_RADAR_PRESENCE_STATE_MACRO_PRESENCE:
        cyhal_gpio_write(LED_RGB_RED, true);
        cyhal_gpio_write(LED_RGB_GREEN, false);
        printf("[INFO] macro presence % " PRIi32 " %f  %" PRIi32 "\n",
               event->range_bin,
               R,
               event->timestamp);
        break;

    case XENSIV_RADAR_PRESENCE_STATE_MICRO_PRESENCE:
        cyhal_gpio_write(LED_RGB_RED, true);
        cyhal_gpio_write(LED_RGB_GREEN, false);
        printf("[INFO] micro presence %" PRIi32 " %f  %" PRIi32 "\n",
               event->range_bin,
               R,
               event->timestamp);
        break;

    case XENSIV_RADAR_PRESENCE_STATE_ABSENCE:
        printf("[INFO] absence %" PRIu32 "\n", event->timestamp);
        cyhal_gpio_write(LED_RGB_RED, false);
        cyhal_gpio_write(LED_RGB_GREEN, true);
        break;

    default:
        printf("[WARN]: Unknown reported state in event handling\n");
        break;
    }
}
```

Compile the code as mentioned in the section 3.1.1 after editing. Follow the steps mentioned in the section 3.1.2 to program the board and to see results on the terminal.

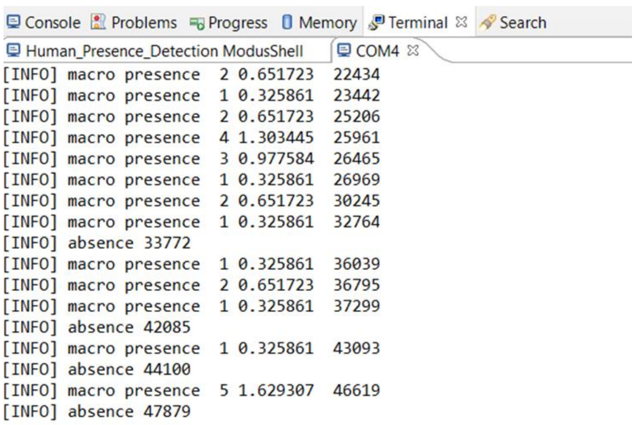

Figure 18 ModusToolbox™ Terminal

Note: *Terminal output will be in format in case of target present < Presence State, Range-Bin, Range, Timestamp>. In case of target absence <Absence, Timestamp>.*

References

[1] Infineon Technologies AG. *XENSIV™ KIT_CSK_BGT60TR13C board*; Available online

[2] Infineon Technologies AG. *XENSIV™ KIT_CSK_BGT60TR13C user guide*; Available online

[3] Infineon Technologies AG: *AN228571 – Getting started with PSOC™ 6 MCU on ModusToolbox™ software*; Available online

[4] Infineon Technologies AG: *Using ModusToolbox™ software*; Available online

Revision history

Document revision	Date	Description of changes
1.00	2023-06-06	Initial release
1.10	2024-10-01	Updated document ID Miscellaneous document cleanup updates

Trademarks

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

Edition 2024-10-01

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

AN080949