User Manual

DA1469x ToF Demo

UM-B-106

Abstract

This document provides basic information to help developers get familiar with the DA1469x ToF Demo application and modify or create a ToF application based on it.

Contents

Figures

Tables

1 Terms and Definitions

ADC	Analog-to-Digital Converter
BLE	Bluetooth Low Energy
CLI	Command-Line Interface
CMAC	Configurable Medium Access Controller
DK	Development Kit
GATT	Generic Attribute Profile
IFFT	Inverse Fast Fourier Transform
IRQ	Interrupt Request
ISM	Industrial, Scientific, and Medical (radio band)
LCD	Liquid-Crystal Display
MTU	Maximum Transmission Unit
NVM	Non-Volatile Memory
SDK	Software Development Kit
TOF	Time of Flight
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
UUID	Universally Unique Identifier

2 References

[1] DA1469x, Datasheet, Dialog Semiconductor.
[2] UM-B-057, SmartSnippets™ Studio User Guide, User Manual, Dialog Semiconductor.
[3] UM-B-092, DA1469x Software Platform Reference, User Manual, Dialog Semiconductor.
[4] UM-B-093, DA1469x PRO Development Kit, User Manual, Dialog Semiconductor.
[5] UM-B-103, DA14695 USB Kit, User Manual, Dialog Semiconductor.

3 Introduction

This document describes the Time of Flight (ToF) ranging application on Dialog's DA1469x Bluetooth Low Energy family products. It describes the basic implementation details to help developers understand the source code.

ToF relates to the measuring of distances between nodes based on the travel time(s) of signals. ToF ranging can be applied to both localization, such as indoor positioning and asset tracking, and security-related use cases like distance bounding (key-less entry).

4 ToF General

The simplest method of measuring distance is by Receiver Signal Strength Indicator (RSSI), as RSSI is directly related to the distance between a transmitter and a receiver. The received signal strength, however, is also a function of the transmit power, antenna radiation profile, and non-line-of-sight attenuations. That is why the RSSI method suffers from a lot of spread and is not very accurate and reliable.

ToF on the other hand is in theory very accurate, as it is based on the constant speed with which signals travel. The ToF implementation is very similar to the well-known RADAR technology. However, as Bluetooth radios are not designed to simultaneously transmit and receive signals, the passive reflection is turned into an active one (Figure 1). Consequently, we need to deal with two independent nodes, each with their own independent timing, and perform individual measurements.

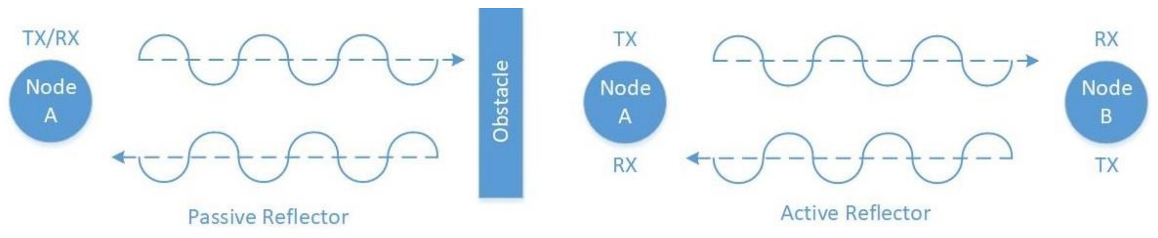


Figure 1: Active and Passive Reflector

Two basic methods exist in measuring the distance:

- The first is based on measuring the round-trip time of a signal initiated at A and echoed back at some later but fixed moment from B

  This approach relies on the fact that at both nodes time can be measured with a very high resolution because radio signals travel with the speed of light, that is, 3 ns per meter, and that the delayed echo from node B is very stable

  Standard BLE devices have no provision for measuring at such high time resolutions, nor methods to guarantee very accurate re-transmissions. This approach complicates implementation and is sensitive to process and voltage and temperature variations

- The second is based on measuring phases of incoming radio signals

  The phase of an unmodulated radio signal is a function of its frequency and the time (or distance) it travels

  Most radio architectures have a mixer in the receive path, bringing the received frequency down to a much lower Intermediate Frequency (IF ~ 1 MHz). The nice thing with mixers is that a 0.1 of a cycle at IF (1 MHz) is identical to 0.1 of a cycle at the received 2.4 GHz (representing a time of 0.04 ns), enabling accurate timing measurements. An additional benefit of this method is the fact that unmodulated carriers and a fixed IF frequency reduce most chip sensitivities, so that the complicated calibration methods can be avoided

The DA1469x ToF demo presented in this user manual is based on the second method using phase measurements. Dialog's DA1469x family is designed to measure the phase at high sampling rates (up to 16 MHz) of the IF signal with a 9-bit A-to-D converter.

To use phase information for measuring distances, the following points need to be considered:

- The first is that the phase of a signal wraps with every period of the signal, resulting in same output for every wavelength (~ 12 cm), which is not very useful when measuring larger distances of up to 100 m or so
- The second is that nodes A and B have their independent time/phase references which are not synchronized. If we would like these nodes to be synchronized, we need do that with the wanted measurement accuracy (1 ns for 30 cm accuracy). From a practical point of view, this leads to additional complexity and power. Luckily the synchronization issue can also be solved in a simpler way by transmitting the signal first from A to B and then from B to A.

These ToF basics for a 'Line-of-Sight' scenario are explained in Figure 2 and Figure 3. The symbol f stands for the radio frequency (2.4 GHz), r stands for the distance between the nodes, and $c0 = 3.10^8\ m/s$ is the speed of light.

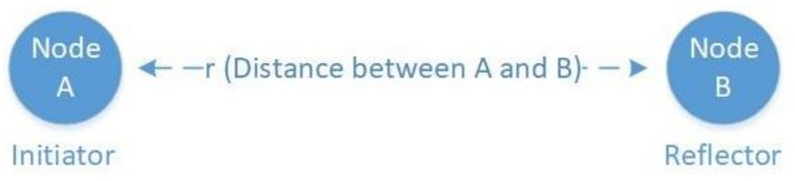
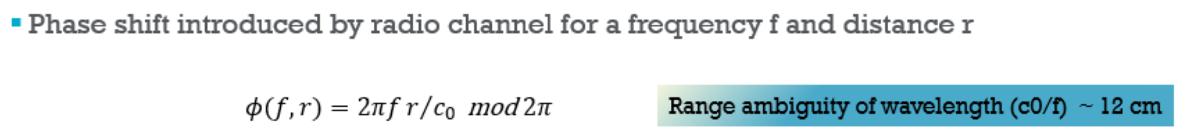


Figure 2: Distance Measurement between Nodes



▪ Phase shift introduced by radio channel for a frequency f and distance r

$$\phi(f,r) = 2\pi f\, r/c_0\ \ mod\, 2\pi$$

Range ambiguity of wavelength (c0/f) ~ 12 cm

Figure 3: Phase Shift Equation

Figure 3 shows that simply measuring the phase of the RF signal is not very useful as it repeats the same value every 12 cm of distance.

To solve the synchronization issue, measure the signals at both node A and node B with φf as the unknown phase offset between A and B during the time of measurement (Figure 4). Note that we assume the transmission AB and BA to be quick enough that this φf is constant.

- Initiator (A) and reflector (B) have their own clock references but need coarse synchronization only

$$\phi_{AB}(f,r) = 2\pi f \frac{r}{c_0} - \varphi_f \;\; mod\, 2\pi \qquad \phi_{BA}(f,r) = 2\pi f \frac{r}{c_0} + \varphi_f \;\; mod\, 2\pi$$

- Adding both phases cancel the unknown synchronisation (φ_f) between A, B

$$\phi_{2W}(f,r) = \phi_{AB} + \phi_{BA} = 4\pi f\, r/c_0 \;\; mod\, 2\pi$$

a range ambiguity ½ wavelength

Figure 4: Synchronization Cancelation Equation

From Figure 4 we can see that by adding both measured phases, the unknown offset (synchronization) between the nodes disappears. However, we still have an unpractical wrapping (ambiguity) in the 2-way phase signal.

To solve the ambiguity (wrapping), apply the above 2-way phase measuring at two different frequencies, $f0$ and $f1$, and evaluate the phase difference of these measurements.

- Resolve ambiguity by measuring at two frequencies (f_0 and f_1 with $\Delta_f = f_1 - f_0$) and looking at phase difference

$$\Delta_\phi = \phi_{2W}(f_1,r) - \phi_{2W}(f_0,r) = 4\pi\Delta_f\, r/c_0 \;\; mod\, 2\pi$$

- Such that

$$r = \frac{c_0}{4\pi\Delta_f}\Delta_\phi \;\; mod\, \frac{c_0}{2\Delta_f}$$

range ambiguity = ½ x <u>difference</u> of wavelength
Δ_f = 1MHz ~ 150 meter

Figure 5: Wrapping Removal Equation

By choosing the two frequencies, $f0$ and $f1$, relatively close to each other, we can push out the wrapping distance to practical values of > 100 m.

To conclude, we now have the scheme shown in Figure 6:

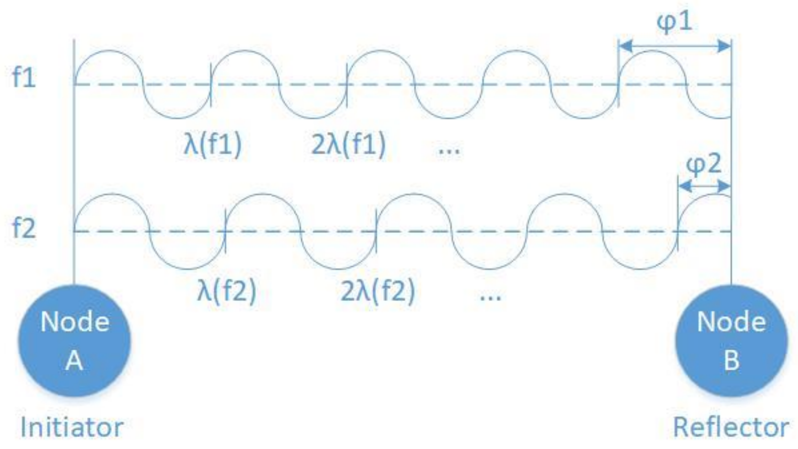


Figure 6: ToF Measurement Overview

Basically, we only need one frequency pair to calculate the correct distance. The reason for using more frequencies across the whole industrial, scientific, and medical (ISM) band (80 MHz wide) has to do with multipaths.

Radio waves travel in all directions and many signals arrive at the receiver after being reflected from indoor objects like walls or floors. For the correct ToF measurement, we need to separate these unwanted multipaths from the wanted direct path, which is also by definition the shortest.

The DA1469x demo offers two implementations to tackle the multipath issue, of which one is called "phase-averaging" and the other is called inverse fast Fourier transform (IFFT).

- Phase-averaging method:

  In this method, all phase differences across the ISM band (80 MHz wide) are averaged to result in a distance which is accordingly the average of all radio signal paths. This works well when the wanted direct path has a dominant signal power and the average of all paths comes close to the wanted direct path. It results in higher measured distances when the direct path is interrupted, for example, when a person stands between the two nodes, because the multipaths contributes more to the averaged radio signal
- IFFT method: it is best explained with Figure 7

  The collection of multipaths can be seen as a complex filter between node A and node B. By sweeping through the frequency band we measure the frequency transfer of this filter and by applying an inverse Fourier transform we get the impulse response of the filter. This impulse response yields various peaks, each representing a multipath. The DA1469x ToF demo does a peak search and selects the peak with lowest r value, which represents the direct path. This method is much less sensitive to the situation where the direct path gets interrupted

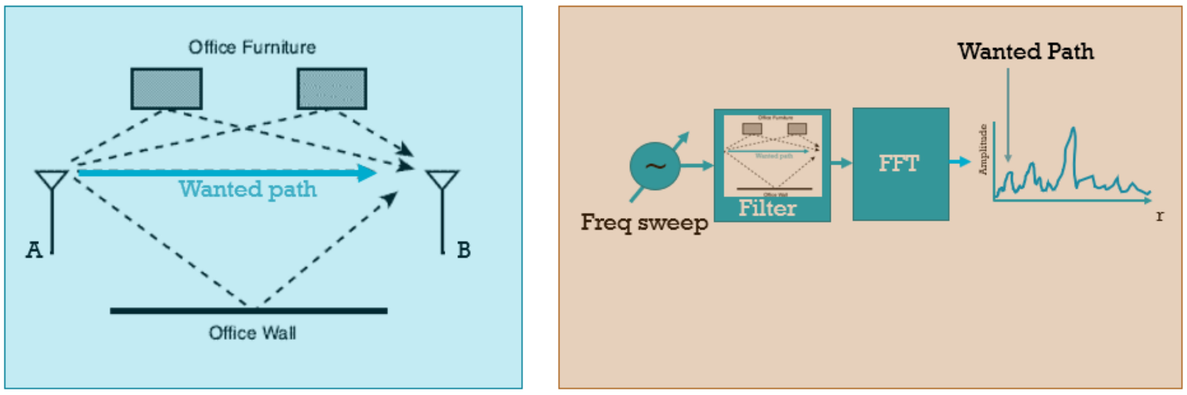


Figure 7: Explanation of IFFT Method

5 ToF Solution

The DA1469x ToF demo application calculates the distance between two devices by using the In-phase and Quadrature outputs of the RF-ADC (IQ data) from continuous wave signal receptions from the two devices. To achieve this, both devices need to synchronously transmit and receive continuous wave signals on a pre-defined set of frequencies for a pre-defined period. The device that receives the signals first is called the Initiator and the device that transmits the signals first is called the Responder. The signal magnitude A and phase angle Ø can be calculated as:

$$A^2 = I^2 + Q^2 \tag{1}$$

$$Ø = \arctan(Q/I) \tag{2}$$

Based on the IQ data, the DA1469x ToF demo application calculates the phase of the signal for each frequency that has been used for signal transmission between the two devices. The results of the phase calculation are exchanged, and each device calculates a distance measurement based on those results. If the IFFT calculation method is selected for the distance measurement, the amplitude values are also calculated and exchanged.

A BLE connection is used both for the synchronization and exchange of the preliminary results with phase and amplitude information. A custom BLE ToF service is implemented for the exchange of results and measurement status information. There are two types of synchronization:

- Radio measurement synchronization (section 5.1)
- Connection event synchronization (section 5.2)

5.1 Radio Measurement Synchronization

The radio measurement synchronization is based on TX and RX interrupts related to the first packet transmission of a BLE connection event. The DA1469x ToF demo application assumes that the radio is idle between connection events and available for the ToF measurement. The ToF signal transmissions start at a predefined time offset after the BLE packet transmission. That offset should be longer than the connection event duration so that the event and any BLE radio activity is completed before the ToF measurement starts. The measurement should complete before the next connection event starts. Since there is no arbitration, the radio availability is verified with a check that the radio is powered down and CMAC (Arm M0+ core/BLE Controller) in sleep mode.

5.2 Connection Event Synchronization

The DA1469x ToF demo application only runs the radio measurement on specific connection events. In order to achieve the event synchronization, the connection event counter is used. The Initiator device is the central and the Responder is the peripheral device in a BLE connection. The central (master) transmits to the peripheral (slave) a start request packet that contains an event counter value for a connection event instant in the future. That event will be used for the radio measurement, assuming the peripheral has enough time to receive this information and prepares for radio measurement before that instant passes.

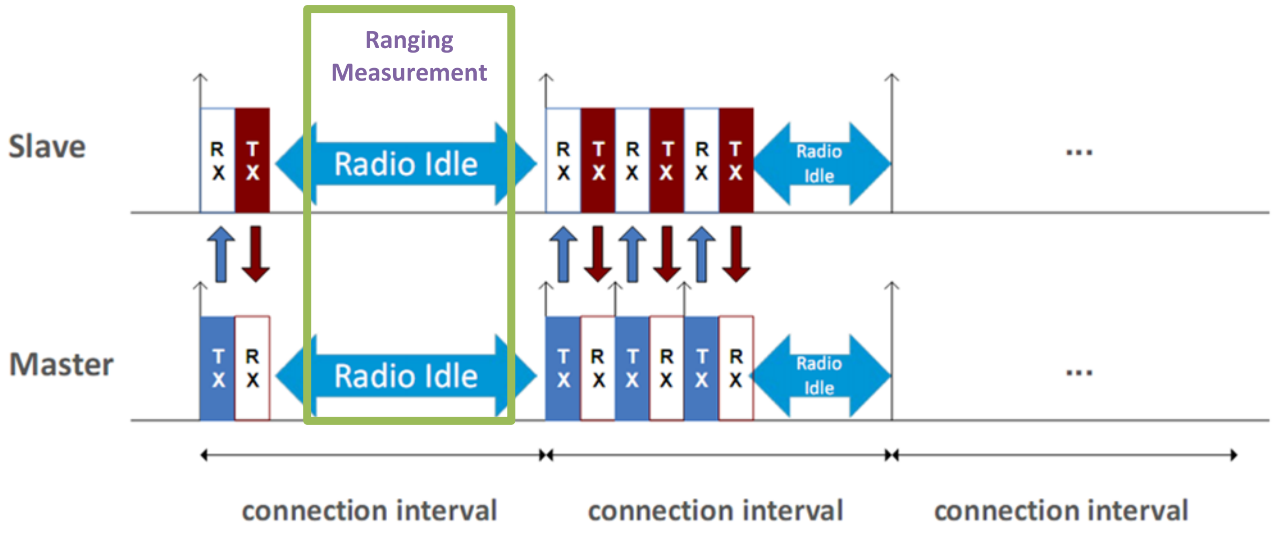


Figure 8: Radio Measurement Synchronization

5.3 ToF Application Data Flow

The ToF devices work in pairs with the BLE Central and Peripheral roles. Each pair has a unique set number. Initially the Peripheral device starts advertising with a custom ToF service UUID and the set number in the advertising data. The Central device starts scanning, looking for the ToF service UUID and a matching set number in the advertising reports it receives. If both the UUID and the set number match, the two devices get connected. The Central device has the Initiator role and the Peripheral device has the Responder role. The following sequence is then repeated until there is a disconnection:

1. The central sends a START measurement request with a connection event counter that refers to a future instant.
2. When that instant comes, the two devices synchronize and perform a radio measurement.
3. Both devices then perform phase calculations.
4. Both devices send the intermediate results to each other and then wait for the remote results to be received.
5. After the results are exchanged, the distance is calculated, and the measurement stops.

Another option is one-sided distance calculation: only one side sends the intermediate results and the other side combines the local and remote results, calculate the distance, and sends the distance measurement to the other side. This option is not available in the current application.

For the Central device, there is also a special command-line interface (CLI) mode where the Central device does not scan but receives CLI start/stop commands. The start command has a Bluetooth address parameter and the device attempts to connect to the remote device with the specific address and start measurements. The stop command stops the measurements and disconnects the remote device.

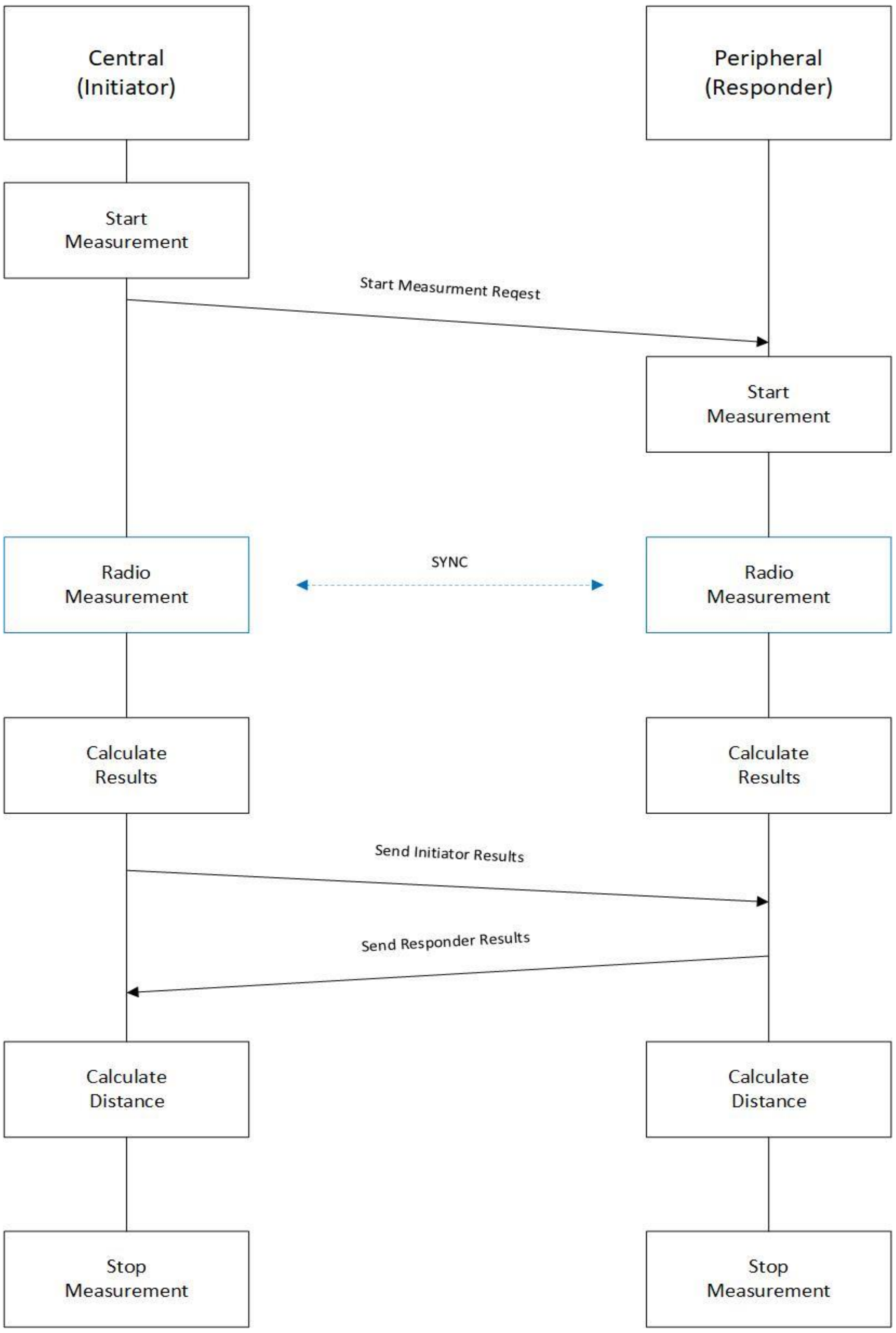


Figure 9: ToF Application Data Flow

6 Software Architecture

Figure 10 presents the different software blocks of the ToF demo application. Each rectangle indicates a module with the dashed ones representing optional features that can be enabled at compile time.

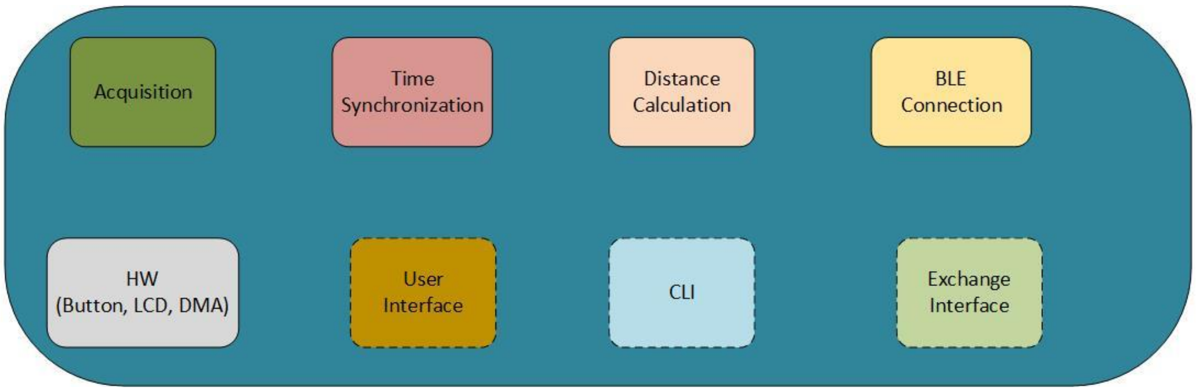


Figure 10: ToF Application Software Modules

6.1 Acquisition

The radio is equipped with a hardware monitoring block (RFMON) which is responsible for the acquisition of the data that are provided by the RF Unit, packing of the data in words of 32 bits, and storing them in system's memory. These data contain the output of the Demodulator and 9-bit RF-ADC samples (IQ data) sampled at 8 MHz sampling frequency (16 MHz sampling frequency is also possible with DA1469x).

The acquisition module is responsible for the collection of IQ data for different frequencies. For each frequency IQ data are collected through the RF monitor block to the RFMON buffer and a slice of the IQ data is copied to a separate acquisition data buffer. At the end of the acquisition the data buffer will contain the IQ data slices for all the different frequencies. Each acquisition on a single frequency is called an atom.

The RFMON buffer is statically allocated to a specific size and it is used with RFMON configured in cyclic mode. The acquisition buffer and length are set during the acquisition configuration.

There are two main activities related to the acquisition:

- Acquisition configuration (section 6.1.1)
- Acquisition cycle (section 6.1.2)

The configuration runs once during the application initialization. The acquisition cycle runs inside the synchronization timer callback when both devices are assumed to be synchronized. The acquisition cycle is executed only when the CMAC (Arm M0+ core/BLE Controller) is in the sleep state.

6.1.1 Acquisition Configuration

Configuration of acquisition parameters is done during the application initialization at function `ble_tof_cw_task()`. The structure with the configuration parameters is called `phase_meas_config_params_t` and is defined in `acquisition.h`. This structure is used to provide some configuration settings, thus not all members are writable.

```
typedef struct phase_meas_config_params_type
{
  ping_pong_usage_t role;
  uint8_t agc_freeze_lvl;      /* value to use during the AGC freeze (0 ... 9) */
  uint16_t meas_start_us;      /* start of data section to copy from each atom in us */
  uint16_t meas_length_us;     /* length of data section to copy from each atom in us
*/
  uint16_t f_start_mhz;        /* first measurement frequency in MHz */
  uint8_t f_step_mhz;          /* difference between two measurement frequencies in MHz
*/
  uint16_t nb_atoms;           /* number of atoms to measure */
  uint32_t *dst_buffer;        /* pointer to destination buffer */
  uint16_t dst_buffer_size;    /* length of destination buffer in Bytes, at least 16000
*/
  bool use_diversity;          /* increase PING and PONG length to support diversity */
  bool use_auto_xtal_trim;
  bool use_distance_average;
}
phase_meas_config_params_t;
```

In the code listed above, we can see the following information:

- Parameter role is related to Tx/Rx sequence. Two roles are defined:
  - Initiator which receives signals first (pong)
  - Responder which transmits signals first (ping)
- Parameters `f_start_mhz`, `f_step_mhz`, and `nb_atoms` can be modified to configure the frequency set that is used in the acquisition. The set is characterized with the first measurement frequency, the frequency step, and the number of frequencies to measure. Currently the maximum number is 40 frequencies (atoms)
- The `dst_buffer` and `dst_buffer_size` parameters can be modified to configure the acquisition data buffer that contains the IQ data slices from the different frequencies of the acquisition
- The `use_diversity` flag is experimental and should not be set
- The `meas_length_us` parameter configures the length of data section to be copied from each atom in µs and should not exceed the default value of `MEAS_LENGTH_US_NORMAL` (48 samples); a larger value will destroy the acquisition timing since the IQ data slice is not copied in time before the next atom acquisition starts
- Parameter `meas_start_us` cannot be written, and the values of `MEAS_START_INIT_US`/`MEAS_START_REFL_US` are the values used for each role
- The acquisition library is built with the `AGC_FREEZE_AUTO` define set. This value cannot change, thus automatic AGC freeze is enabled and the `agc_freeze_lv` parameter is always overwritten with the automatic freeze value
- The `use_auto_xtal_trim` and `use_distance_average` parameters are not directly related to the acquisition but rather to the measurement. The parameter `use_auto_xtal_trim` enables the auto-trimming of the XTAL32M, and the parameter `use_distance_average` includes a distance average functionality when set

6.1.2 Acquisition Cycle (Radio Measurement)

The radio measurement steps of the acquisition cycle are listed below:

1. Radio System Init
2. Prepare Acquisition
3. Run Acquisition for each atom
4. Radio System Stop

The radio measurement is triggered at both the Initiator and the Responder when the System Timer (SysTick) expires. It involves two parts:

- Acquisition initialization
- Actual acquisition

The acquisition initialization part is implemented by calling the following two functions, `radio_system_init()` that powers on the radio and the CMAC and enables the radio (step 1) and `prepare_acquisition()` that configures the RF monitor block and prepares the radio for the acquisition (step 2).

The first time the SysTick expires after the synchronization, the preparation for the measurement starts. The timer is set again to both devices for the start of the acquisition. The preparation timeout value is pre-calculated so that the preparation will take place before the start timer expires. Currently it is set to 1750 µs.

When SysTick expires for the second time, it enters free running mode which is set to be reset every 160 µs and the actual acquisition part starts (step 3). Based on the 160 µs interval, a new atom acquisition starts every 480 µs. When the acquisition is completed, the radio and the CMAC are powered down (step 4).

6.2 Time Synchronization Mechanism

In order to get reliable results from the ping pong sequence, the starting point and the atom changes should happen with an accuracy ≤ 2 µs. The mechanism that achieves the synchronization with this accuracy involves two interrupt request (IRQs), that is, the RF diagnostics IRQ and the SysTick IRQ.

6.2.1 IRQ Latency Improvement

Besides the two IRQs, another critical factor to ensure timing accuracy is the IRQ latency. In order to reduce the latency, the following steps are followed.

Both IRQs are configured with the highest priority (0) to avoid latency from FreeRTOS code that enables/disables interrupts with priority ≥1.

The system must always stay active while waiting the timers to expire. This way the execution path does not execute the code that implements the going to and waking up from sleep path that runs with interrupts disabled.

In addition, there are a few software development kit (SDK) modifications:

- Critical path functions `prvSystemSleep` and `eTaskConfirmSleepModeStatus` have been moved to RAM to avoid latency from cache misses
- A new `pm_active_wfi` function that minimizes the checks with IRQs disabled has been introduced since the system is not going to sleep

6.2.2 Synchronization Sequence

The synchronization is based on the first packet transmission of a BLE connection event and is implemented using the RF diagnostics IRQ and the SysTick IRQ.

The RF diagnostics IRQ introduces the first step of synchronization.

Based on that, the SysTick IRQ is used as the second step of synchronization that starts the radio measurement.

6.2.2.1 RF Diagnostics IRQ

The RF Diagnostics IRQ is used to get information regarding the packet transmission/reception. This IRQ is configured to be triggered when `TX_ENABLE` is set for the Initiator and when the `SYNC_WORD_FOUND` is set for the Responder. Since `TX_ENABLE` is set at a predefined time before the actual packet transmission, the difference between the `TX_ENABLE` of the Initiator and the `SYNC_WORD_FOUND` of the Responder is a constant offset calculated to be 121 µs for two DA1469x devices.

The RF Diagnostics IRQ handler starts the SysTick which programs the initiation of the acquisition cycle for the two devices. The timer value is adjusted with the previously mentioned constant offset so that both devices can wake up and start the radio measurement at the same time. The handler also disables the RF Diagnostics IRQ to prevent it from being triggered if there are more packets in the connection event or during the execution of the acquisition code. The RF Diagnostics IRQ will be re-enabled after the acquisition is completed.

There is an API with functions `ble_tof_sync_start()/ble_tof_sync_stop()` that can be used from the application to enable/disable the first part of synchronization with the RF Diagnostics IRQ.

6.2.2.2 SysTick IRQ

When the SysTick expires, both the Initiator and the Responder can wake up and start the acquisition.

The SysTick has the following states: PREPARE, START, RUN, and STOP.

The API functions `ble_tof_cw_start()/ble_tof_cw_stop()` can be used from the application to enable/disable the second step of synchronization that starts the acquisition cycle with the SysTick IRQ.

The SysTick Handler always reenables the RF Diagnostics IRQ even if the acquisition is not enabled. Also, it always sends a `SYNC_END` notification to the application. This is critical because the application depends on this notification for understanding that a connection event has been completed with or without the acquisition part following.

6.2.2.3 Notification Timer

`OS_TASK_NOTIFY_FROM_ISR` call is unsafe if SysTick has an IRQ priority of 0, so an extra timer (`HW_TIMER`) is used to pass FreeRTOS compatible notifications to the application task.

6.3 Distance Calculations (Phase/IFFT)

Once acquisition finishes, distance results are extracted out of the sampled measurement data. All operations specific to distance estimation are implemented inside directory “calc”. Figure 11 shows an overview of the related process.

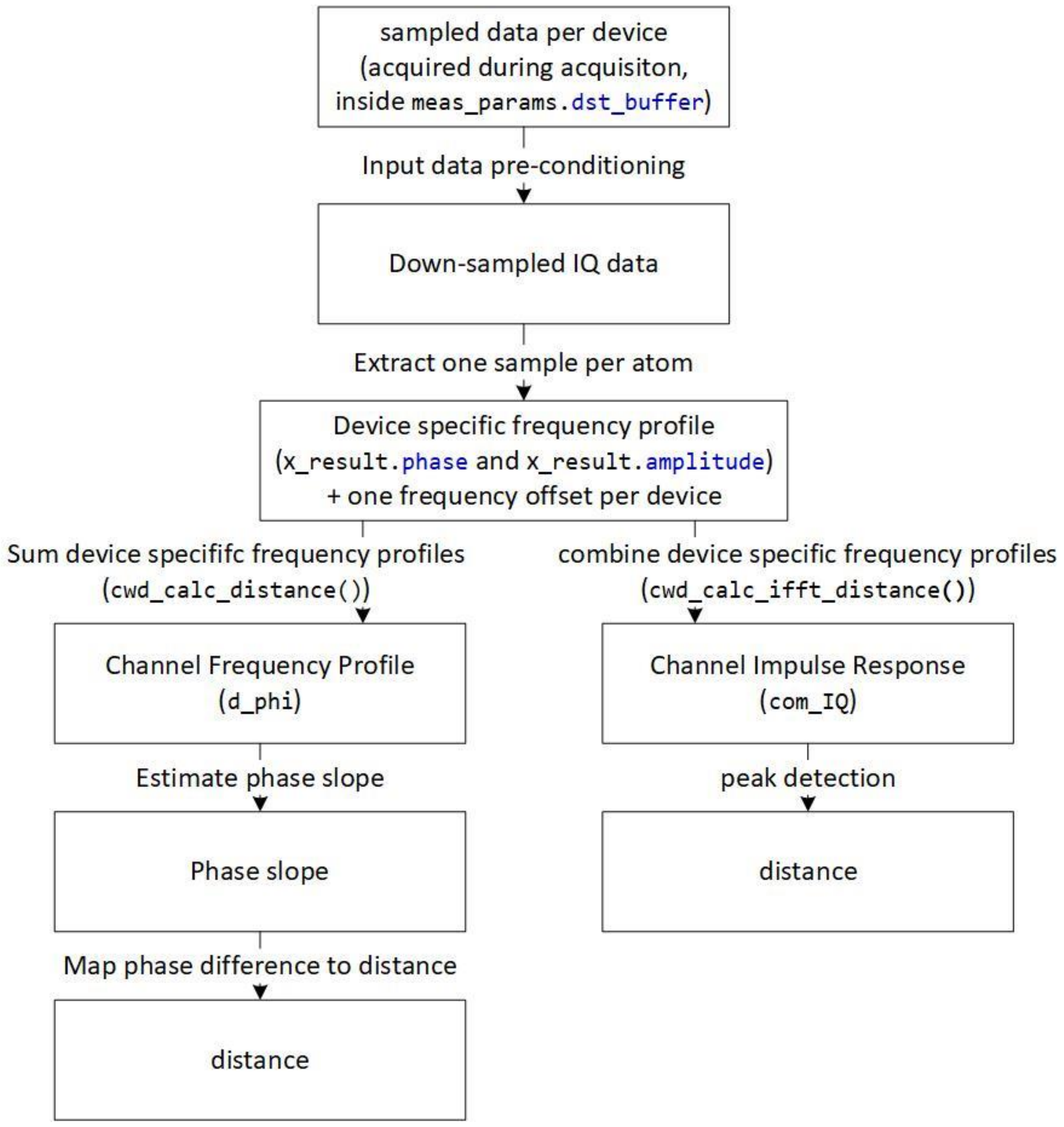

sampled data per device (acquired during acquisiton, inside meas_params.dst_buffer)
Input data pre-conditioning
Down-sampled IQ data
Extract one sample per atom
Device specific frequency profile (x_result.phase and x_result.amplitude) + one frequency offset per device
Sum device specififc frequency profiles (cwd_calc_distance())
combine device specific frequency profiles (cwd_calc_ifft_distance())
Channel Frequency Profile (d_phi)
Channel Impulse Response (com_IQ)
Estimate phase slope
peak detection
Phase slope
distance
Map phase difference to distance
distance

Figure 11: Distance Calculations

There are two basic ways of distance estimation:

- IFFT based distance calculation (right hand side in Figure 11, and section 6.3.1)
- Phase based distance calculation (left hand side in Figure 11, and section 6.3.2)

The following algorithm steps are common to both ways:

1. Input data pre-conditioning:
   a. Extract IQ data out of acquired test bus data samples (sampling rate of 8 MS/s)
   b. Sample down by factor of two to a sampling rate of 4 MS/s
   c. number of samples to be processed is drastically reduced but the data samples still yield similar results
2. Extract one representative sample per atom/measurement frequency (output data of this step is called “device specific frequency profile” here):
   a. Compensate DC offset per atom (can be deactivated by setting parameter “`cwd_parm.comp_dc_offset`” to “false”)
   b. Go over from IQ domain into polar domain (extract phases and magnitudes)
   c. Mix phases down by the intermediate frequency `f_if`, ±1 MHz, depending on role
   d. Initiator has its PLL 1 MHz above the Responder, so it will receive 1 MHz below its frequency and has to mix it 1 MHz up to get it to 0 IF. Responder has to mix it down by 1 MHz
   e. Apply a linear fit on the phase data of each atom to get one frequency offset and one phase value per atom
   f. Average magnitude values to get one magnitude per atom
3. Average frequency offset for all atoms
4. Exchange frequency profiles between devices.

6.3.1 IFFT Based Distance Estimation (Default)

If "TOF_ENABLE_IFFT_DIST_CALC" is defined, the distance is determined as follows:

1. Combine device specific frequency profiles into one single channel profile and go over to time domain:
   a. Go over from polar domain into IQ domain (generate signal vectors out of phase and frequency data)
   b. Multiply device specific frequency profiles to get the channel frequency response
   c. Go over into time domain via IFFT
   d. Generate the Channel Impulse Response (Channel Transfer Function) by using absolute values only
   e. Normalize the Channel Impulse Response
2. Detect the most reasonable peak of the Channel Impulse Response and map the related time delay to a distance:
   a. Detect the first 20 maxima of the Channel Impulse Response
   b. Filter the peak list by removing all peaks below a configurable threshold
   c. Calculate corresponding distance to each peak above the threshold
   d. Find the peak with the shortest distance and take it as the estimated result

6.3.2 Phase Based Distance Estimation

If "TOF_ENABLE_IFFT_DIST_CALC" is not defined, the distance is determined as follows:

1. Sum phase data of both devices per atom to get a Channel Frequency Profile
2. Estimate the slope of the Channel Frequency Profile (phases)
    a. Build phase differences between subsequent phase values
    b. Average phase differences
3. Map the estimated phase difference to a distance and take it as the estimated result

The underlying relation between distance D, phase difference $\Delta\varphi$, frequency difference Δf, and speed of light c is:

$$D = \frac{c}{4\pi} \cdot \frac{-1 \cdot \sum_{N-1} \Delta\varphi_n}{(N-1) \cdot \Delta f} \quad (3)$$

D	Distance in m
c	Speed of light in m/s
$\Delta\varphi$	Phase difference in radians
Δf	Frequency difference in Hz
N	Number of atoms

6.4 BLE Connection

There are two configuration parameters related to the BLE connection: the role and the set number.

A device is configured to have the role of an Initiator or a Responder. An Initiator is the BLE central device while a Responder is the BLE peripheral device.

The set number is used to distinguish two different ToF sets that are simultaneously operated in each other's vicinity (within BLE range).

6.4.1 Central Role

The central device goes through the following sequence of states before the measurement cycle starts:

1. Scanning State:

   The device starts scanning, looking for the ToF service UUID and a matching set number in the advertising reports it receives. If both the UUID and the set number match, the device attempts to connect with the remote device.
2. Connected State:

   The connection attempt is successful.
3. Maximum Transmission Unit (MTU) Exchange State:

   The central device requests the maximum allowed MTU to increase results throughput.

   Data packet length extension is by default enabled on the DA1469x devices so the PDU Payload Length in the Maximum Transmit Data Channel is 251 octets.
4. Discovery State: the device discovers BLE services and related attributes.

After the discovery is completed, the device starts the measurement cycle.

In CLI mode the scanning is bypassed, and the device enters a Connecting State, attempting to connect to a remote device with a specific Bluetooth address that is a parameter of the start measurement command.

6.4.2 Peripheral Role

The peripheral device goes through following sequence of states before the measurement cycle starts:

1. Advertising State:

   The device starts advertising with the ToF service UUID and the device set number in the advertising data. Then it waits for connection requests from the central device.
2. Connected State:

   After the two devices connect, the peripheral replies to discovery and MTU requests and waits for START request to start measurement sequence.

6.4.3 BLE ToF Service

A custom BLE ToF service is implemented for the exchange of results and measurement status information. The service attribute database contains the following attributes:

- Status: information from START/STOP requests. START request includes the BLE event counter for the next measurement. STOP request includes the reason the measurement has failed
- I_Result: result information from the initiator that contains the synchronization event counter, AGC gain, frequency offset, and the phase and amplitude calculations from the frequencies set that have been used in the acquisition
- R_Result: similar result information from the responder

- Value: the distance calculation result, currently not used since both devices receive the results of the other side and calculate the distance

6.4.4 ToF Measurement Cycle

After both devices are connected, they follow a similar measurement cycle until one device gets disconnected:

1. START:

   The central device enables acquisition, sets the synchronization event counter, and sends a START request to the peripheral device using the BLE GATT write without response command

   The peripheral device waits for the reception of the START command from the central device

   Both devices switch to SYNC state once the peripheral device receives the START command
2. SYNC:

   Both the central and peripheral devices disable acquisition, switch to CALC state, and set timeout for remote side results.
3. CALC:

   Both the central and peripheral devices calculate phases and amplitudes, send results to each other (the central uses BLE GATT write without response command and the peripheral uses a BLE GATT notification), wait for remote results, calculate distance, and switch to the STOP state if timeout happens or remote stop notification is received.
4. STOP:

   The Central and Peripheral devices print log information and start again.

6.5 Hardware Support

The follow hardware features are used to support the ToF demonstration in a user-friendly way:

- Buttons: the function `tof_hw_k1_button_pressed` checks whether K1 button is pressed. It is used during reset to decide whether to enter the configuration mode
- DMA: the function `init_cw_dma` initializes and the function `start_cw_dma` starts the DMA transfer of the IQ data slices to the acquisition destination buffer
- LCD: the high-level interface function `tof_lcd_init` initializes display and the function `tof_lcd_draw_string` draw a spring with the distance measurement value in a display

6.6 User Interface

This module contains different fonts implementations and basic graphic functions for text drawing in an LCD display.

6.7 CLI

The CLI is a special mode for the Initiator device where the device does not start scanning after reset but waits for CLI commands to set address, start, stop, and acts accordingly. See section 9.6 for how to enable this mode.

The CLI functionality is implemented in the file `tof_cli.c`. The function `tof_cli_init` initializes the CLI task. This task receives input strings and notifies the waiting BLE central task that a string has been read from CLI. Function `tof_cli_get_clistr` returns the current input string that has been read from CLI. Function `readline` reads a line from CLI and `parseline` parses the line to get a command. In the '`start`' command, the line should also contain the address parameter.

6.8 External Interface

The external interface module can be used to support communication with an external host through RAM using the SEGGER JTAG. To enable this external interface functionality, in the file `custom_config_qspi.h`, change the line `#undef MATLAB_MODE` to `#define MATLAB_MODE`.

Initially this mode was used to support MATLAB interaction through SEGGER JTAG. Similar functionality can also be implemented now with Python using PyMon. PyMon is a python package that enables the control of a Dialog Semiconductor Bluetooth SoC through a SEGGER J-Link debugger probe.

The header file `tof_interface.h` describes a sample set of parameters for data exchange between the application and MATLAB. There are three different types of parameters:

- Bidirectional flags used for handshake between the ToF and the MATLAB application
- Configuration parameters set from MATLAB
- Information parameters sent to MATLAB

These parameters are stored in a special "`.tof_section`" so that they can be located at a fixed predefined address that MATLAB can access. For this reason, the original `sections.ld.h` SDK file has been modified and the "`.tof_section`" has been added.

There are three main functions in the "`.tof_section`":

- `tof_interface_init` for parameters initialization from the application side
- `tof_interface_params_exchange` for setting parameters from MATALB, and
- `tof_interface_data_exchange` for sending information to MATLAB

The last two functions implement a blocking handshake where they set a ready flag and then wait for MATLAB to set parameters/copy information and then clear a ready flag.

7 Configuration Options/Nonvolatile Parameters

There are two different options for configuring theDA1469x ToF demo application:

- The static switches that need to be configured during compilation time (section 7.1)
- The dynamic parameters that are stored in nonvolatile memory and can be changed dynamically at run time (section 7.2)

7.1 Compile Time Configuration Switches

The configuration switches for the DA1469x ToF demo application listed in Table 1 are defined in file `ble_tof_cw_config.h`.

Table 1: Configuration Switches

Switch	Default	Effect
TOF_ENABLE_CW_DIST_CALC	enabled	It enables distance calculation in firmware
TOF_ENABLE_IFFT_DIST_CALC	enabled	If it is enabled, the distance is calculated by Channel Impulse Response If it is disabled, the distance is calculated by Channel Frequency Profile
TOF_ENABLE_JITTER_STATS	disabled	Pins P1_02 and P1_03 get toggled during synchronization and can be used to measure synchronization jitter when this switch is enabled
TOF_ENABLE_AUTO_XTAL_TRIM	enabled	If it is enabled, the XTAL32M_TRIM register is automatically tuned so that the frequency offset is minimized (Note 1)
TOF_ENABLE_DISTANCE_AVG	enabled	If it is enabled, a 4-value sliding average is applied to the raw distance value
CLI_MODE	disabled	If it is enabled, the central device does not scan but waits for commands from CLI
MATLAB_MODE	undefined	If it is defined, data exchange with external host after measurement completion is supported
USE_RANDOM_BD_ADDRESS	enabled	If it is enabled and the current address is equal to the SDK default, a random BD address can be used

Note 1 See section 7.2 for how to enable XTAL auto trimming and distance average functionality

7.2 Nonvolatile Parameters

Configuration of nonvolatile parameters is done during the application initialization by function `ble_tof_cw_task()`. The structure with the configuration parameters is called `tof_config_params_t` and is defined in file `ble_tof_cw_task.c`.

```
typedef struct {
        tof_role_t role;
        uint8_t set_number;
        int16_t cw_offset;
        uint16  xtal32m_trim_value;
        uint8_t flag_auto_xtal32_trim:1;
        uint8_t flag_distance_averaging:1;

} tof_config_params_t;
```

- Parameter `role` is related to the acquisition Tx/Rx sequence. Two roles are defined:
  - Initiator, which receives signals first (pong), and
  - Responder, which transmits signals first (ping)
- `set_number` is a number less or equal to 255. It is used under normal operation to uniquely identify an Initiator/Responder set. The Initiator (central) device uses this number to identify the Responder (peripheral) to which it will connect. It is not meaningful if CLI mode is enabled
- `cw_offset` is a fixed offset in centimeters that adjusts the distance measurement so that the correct distance result is produced. Initially we need to calibrate the distance measurements at fixed distances and calculate the `cw_offset`
- `xtal32m_trim_value` is used to set the XTAL frequency trimming register. This value may change if auto trimming is enabled
- If `flag_auto_xtal32_trim` is set and the related compile time flag `TOF_ENABLE_AUTO_XTAL_TRIM` is enabled, the application auto-tunes the XTAL32M_TRIM register in order to minimize the frequency offset at the end of each measurement
- If `flag_distance_averaging` is set and the related compile time flag `TOF_ENABLE_DISTANCE_AVG` is enabled, a 4-value sliding average is applied to the raw distance value
- If K1 button is pressed, the application enters a configuration mode where all these parameters can be changed from CLI

8 Development Environment

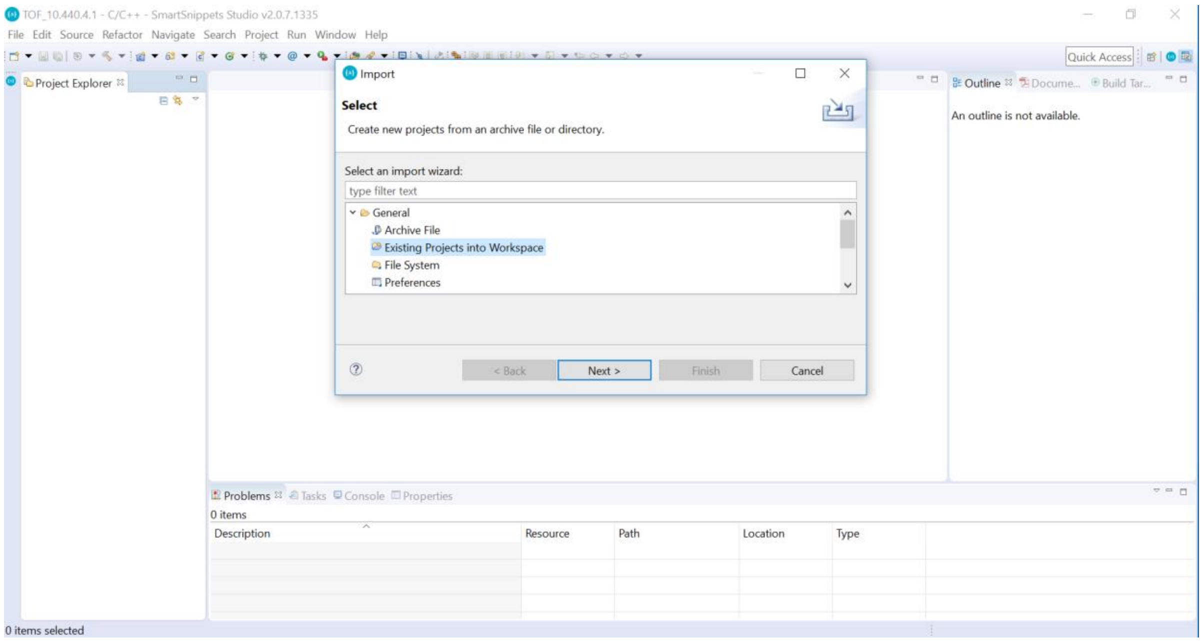
This section describes how to set up the development environment of the DA1469x ToF demo application.

8.1 Installing the Environment

1. Unzip the ToF engineering release zip file (for example, TOF_10.440.4.1) into a known location on your drive.
2. All software development is based on SmartSnippets™ Studio. Please refer to *UM-B-057* [2] for its user manual. Follow the setup instructions of *chapter 1* in [2] for a fresh installation.
3. In the **Welcome** screen of SmartSnippets™ Studio, enter the location of your unzipped ToF engineering release.
4. Select IDE from **Tools** section of the **Welcome** screen. The regular Eclipse environment view should now appear.

8.2 Importing and Building the ToF Project

1. To import the required projects, follow the steps below:
   a. Select **File** > **Import**.
   b. Select **General** > **Existing Projects into Workspace** >click **Next** (Figure 12) > **Select root directory** (Figure 13).
   c. Browse to the location of the release.
   d. Deselect all projects.
   e. From the project list select only the `ble_tof_cw` and the `python_scripts` projects.
   f. Click **Finish**.


TOF_10.440.4.1 - C/C++ - SmartSnippets Studio v2.0.7.1335

Import

Select

Create new projects from an archive file or directory.

Select an import wizard:

type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences

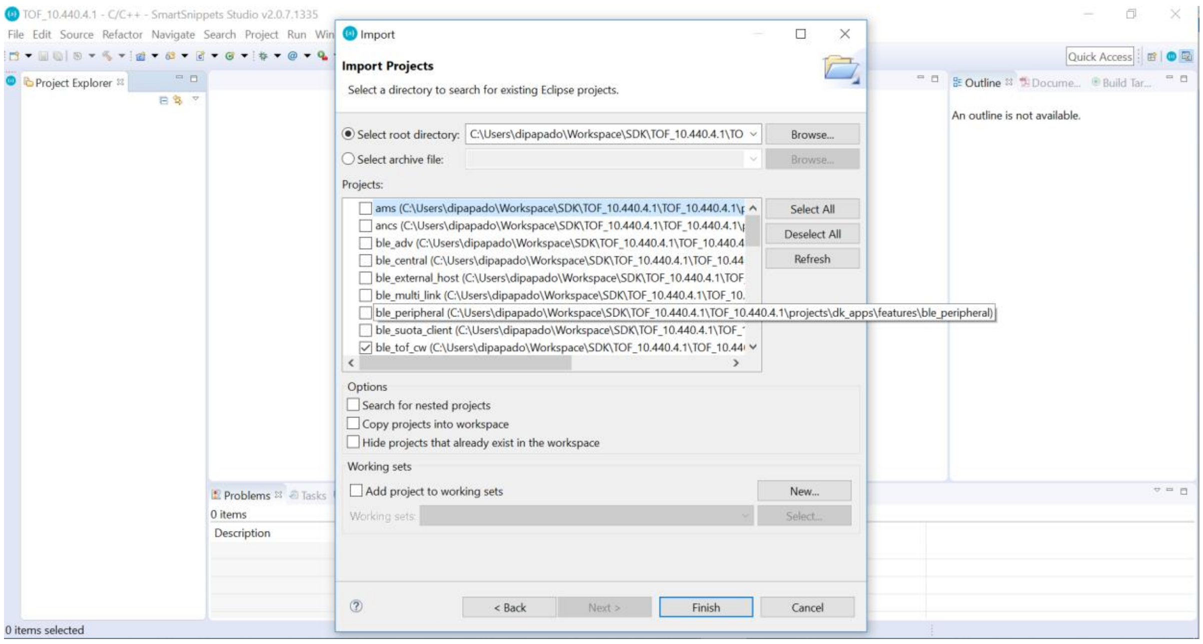
< Back | Next > | Finish | Cancel

An outline is not available.

0 items

0 items selected

Figure 12: Importing a Project into the Workspace



Import Projects

Select a directory to search for existing Eclipse projects.

(●) Select root directory: C:\Users\dipapado\Workspace\SDK\TOF_10.440.4.1\TO

( ) Select archive file:

Projects:

- [ ] ams (C:\Users\dipapado\Workspace\SDK\TOF_10.440.4.1\TOF_10.440.4.1\p
- [ ] ancs (C:\Users\dipapado\Workspace\SDK\TOF_10.440.4.1\TOF_10.440.4.1\p
- [ ] ble_adv (C:\Users\dipapado\Workspace\SDK\TOF_10.440.4.1\TOF_10.440.4
- [ ] ble_central (C:\Users\dipapado\Workspace\SDK\TOF_10.440.4.1\TOF_10.44
- [ ] ble_external_host (C:\Users\dipapado\Workspace\SDK\TOF_10.440.4.1\TOF
- [ ] ble_multi_link (C:\Users\dipapado\Workspace\SDK\TOF_10.440.4.1\TOF_10.
- [ ] ble_peripheral (C:\Users\dipapado\Workspace\SDK\TOF_10.440.4.1\TOF_10.440.4.1\projects\dk_apps\features\ble_peripheral)
- [ ] ble_suota_client (C:\Users\dipapado\Workspace\SDK\TOF_10.440.4.1\TOF_
- [✓] ble_tof_cw (C:\Users\dipapado\Workspace\SDK\TOF_10.440.4.1\TOF_10.44

Options

- [ ] Search for nested projects
- [ ] Copy projects into workspace
- [ ] Hide projects that already exist in the workspace

Working sets

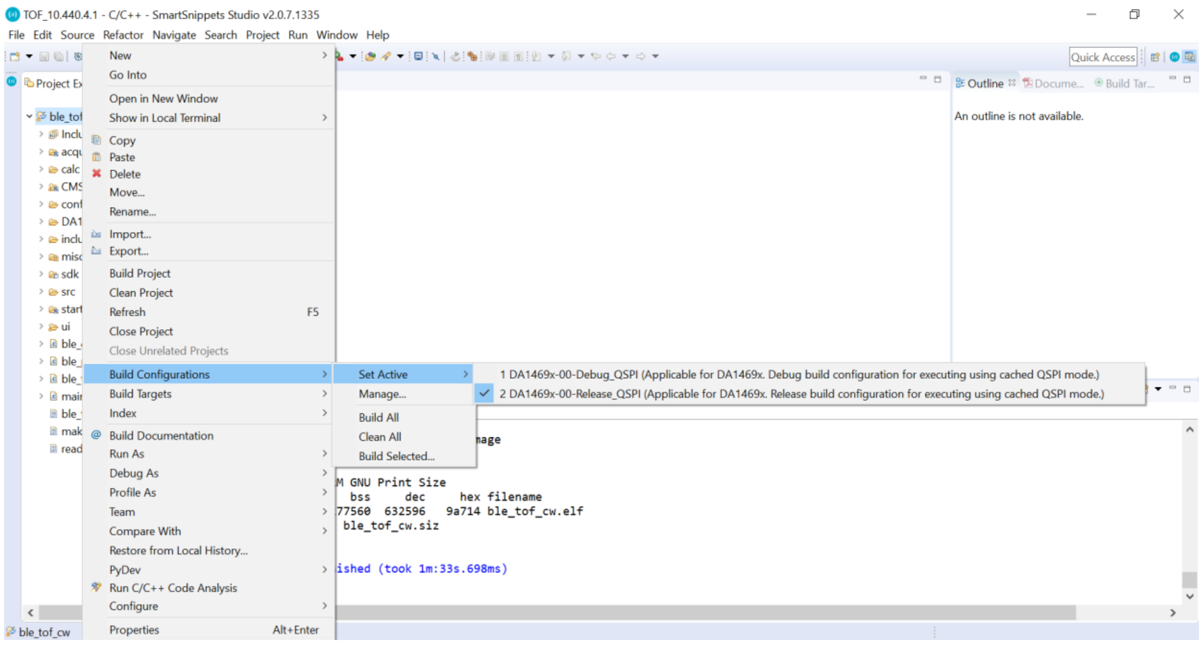
- [ ] Add project to working sets

Working sets:

An outline is not available.

Figure 13: Selecting the `ble_tof_cw` Project

2. Select a build configuration and build the `ble_tof_cw` project (Figure 14). Right click on the project name to get the project menu and select **Build Configurations** > **Set Active** to see configuration options.



1 DA1469x-00-Debug_QSPI (Applicable for DA1469x. Debug build configuration for executing using cached QSPI mode.)

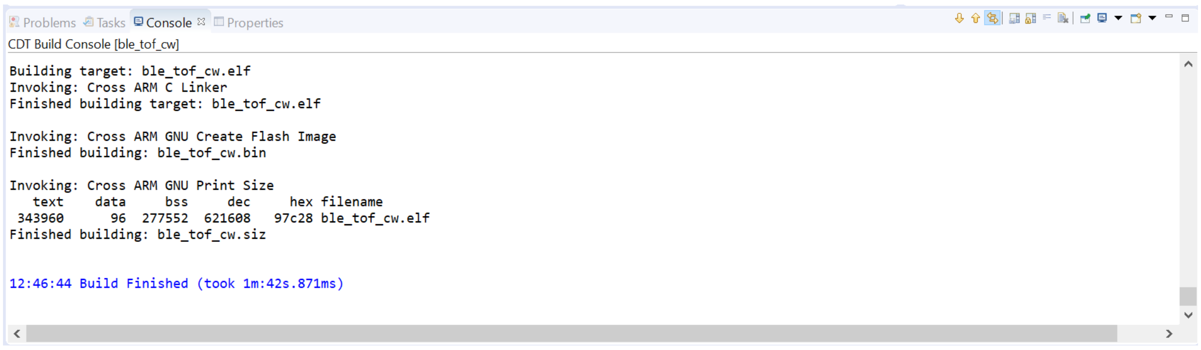
✓ 2 DA1469x-00-Release_QSPI (Applicable for DA1469x. Release build configuration for executing using cached QSPI mode.)

An outline is not available.

```
age

M GNU Print Size
 bss     dec     hex filename
77560  632596   9a714 ble_tof_cw.elf
 ble_tof_cw.siz

ished (took 1m:33s.698ms)
```

Figure 14: Build Configurations for `ble_tof_cw` Project

3. A successful build for the release configuration of `ble_tof_cw` project gives the result in the console window shown in Figure 15.



```
Building target: ble_tof_cw.elf
Invoking: Cross ARM C Linker
Finished building target: ble_tof_cw.elf

Invoking: Cross ARM GNU Create Flash Image
Finished building: ble_tof_cw.bin

Invoking: Cross ARM GNU Print Size
   text    data     bss     dec     hex filename
 343960      96  277552  621608   97c28 ble_tof_cw.elf
Finished building: ble_tof_cw.siz

12:46:44 Build Finished (took 1m:42s.871ms)
```

Figure 15: Build Log

8.3 Burning the Image into Flash Memory

1. To write the generated image into the Flash memory, run the external script `program_qspi_jtag` (Figure 16):

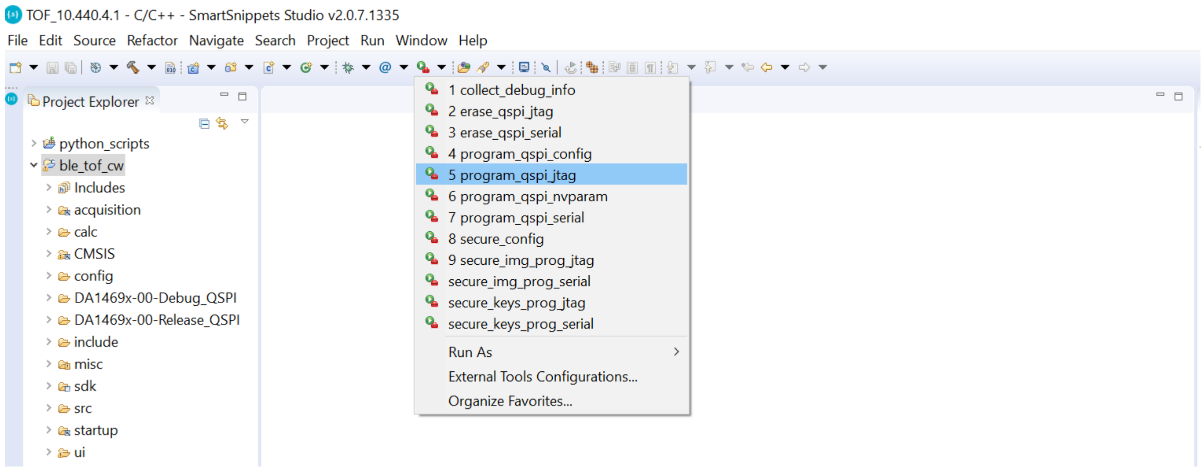
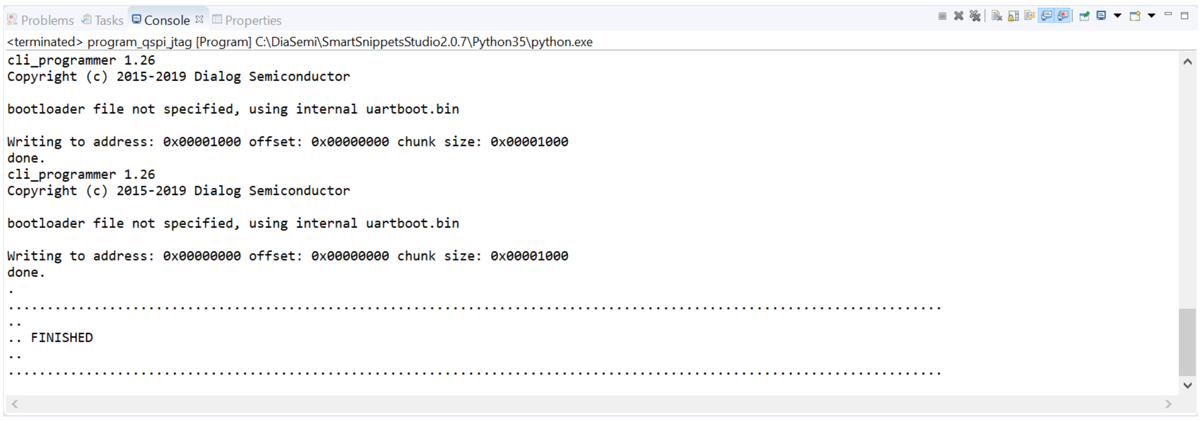


Figure 16: Programming the Flash Image

NOTE
Make sure to first click on the `ble_tof_cw` in the **Project Explorer** tree on the left before triggering the programming script. Based on the selected project before triggering the script, it will flash the last build configuration that has been compiled successfully. Therefore, you need to make sure the `ble_tof_cw` is selected and not any other project

2. Successful programming of the DA1469x ToF demo application should look like the log output shown in Figure 17:



```
<terminated> program_qspi_jtag [Program] C:\DiaSemi\SmartSnippetsStudio2.0.7\Python35\python.exe
cli_programmer 1.26
Copyright (c) 2015-2019 Dialog Semiconductor

bootloader file not specified, using internal uartboot.bin

Writing to address: 0x00001000 offset: 0x00000000 chunk size: 0x00001000
done.
cli_programmer 1.26
Copyright (c) 2015-2019 Dialog Semiconductor

bootloader file not specified, using internal uartboot.bin

Writing to address: 0x00000000 offset: 0x00000000 chunk size: 0x00001000
done.
.
.......................................................................................................................
..
.. FINISHED
..
.......................................................................................................................
```

Figure 17: Programming Flash - Log Output

8.4 Build Configurations

The DA1469x ToF demo application is compatible with both the DA14695 PRO and USB board. It also supports the option to add an OLED or an Adafruit display depending on the board. The related configurations are controlled by defines in file `custom_config_qspi.h` (located under folder `ble_tof_cw/config/` in the workspace).

8.4.1 Boards

The default build configuration is for the DA1469x PRO development kit (DK) board. The USB DK board is also supported. In order to build firmware for the USB board you need to uncomment the line:

```
#define dg_configUSE_Board        "boards/brd_usb_kit_da1469x.h"
```

8.4.2 LCDs

By default, LCD support is disabled. Both the DA1469x PRO and USB boards can support the mikroBUS ClickBoard OLED display (part number PSP27801) at mikroBUS slot 1. Only the DA1469x PRO board can support the Adafruit display on the Arduino shield interface (part number DT280QV10CT). To enable it you need to:

1. Enable the LCD adapter:

   ```
   #define dg_configLCDC_ADAPTER              ( 1 )
   ```

2. Enable the specific LCD configuration for OLED display:

   ```
   #define dg_configUSE_PSP27801              ( 1 )
   ```

   or for AdaFruit display:

   ```
   #define dg_configUSE_DT280QV10CT           ( 1 )
   ```

9 Setup/Platforms

The DA1469x ToF demo application is compatible with both the DA14695 PRO DK and USB DK. The HW boards are described in references [4] and [5].

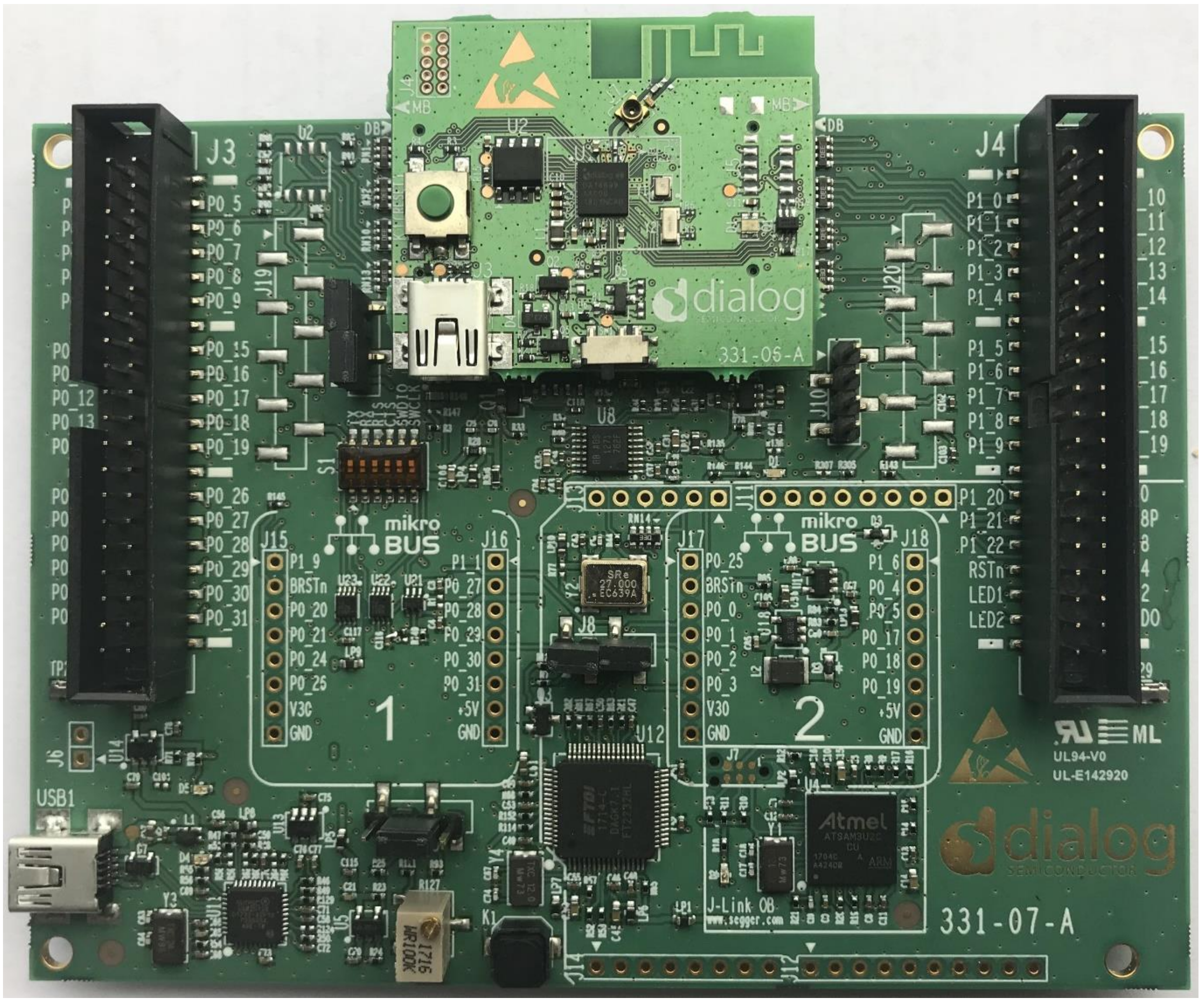


Figure 18: DA14695 PRO DK

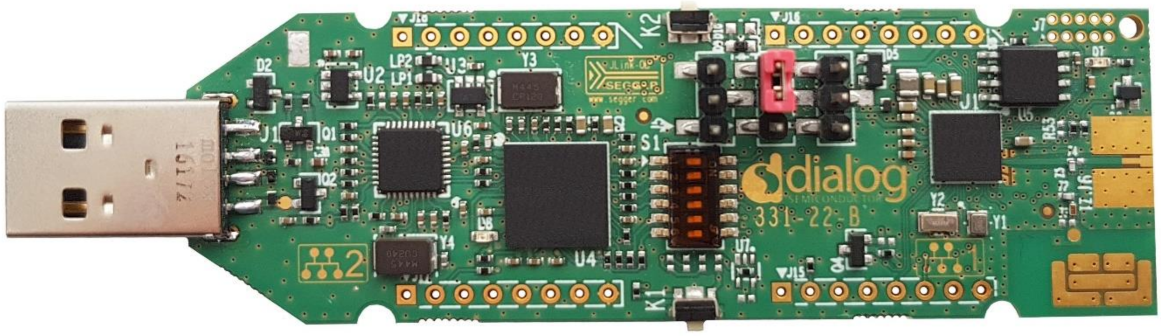


Figure 19: DA14695 USB DK

9.1 Connecting Hardware and Powering On

Do **NOT** change any jumper positions or remove/misalign the daughterboard.

To flash the demo image please first follow the instructions in sections 8.1and 8.2.

The boards are powered via USB. Connect the DA14695 PRO DK to a laptop or desktop by the provided USB cable to power it on. To connect the DA14695 PRO DK to a power adapter, please solder a header on J6 and place a jumper there to bypass enumeration with a USB host. To power on the DA14695 USB DK via a power bank, please remove D9, the tiny diode near K2.

After being powered on via USB, the DA14695 PRO or USB DK board should then enter Scanning or Advertising mode based on the configuration of the nonvolatile role parameter.

9.2 Buttons and Switches

Both DKs have buttons/switches with similar functionality:

- DA14695 PRO DK
  - KRESET: HW resets the DA1469x (daughter board)
  - K1: Device enters configuration mode if button is pressed during reset (mother board)
- DA14695 USB DK
  a. K2: HW resets the DA1469x
  b. K1: Device enters configuration mode if button is pressed during reset

9.3 LCD Support

Both DKs support the OLED C click display connected to mikroBUS 1 sockets.

The DA14695 PRO DK also supports the Adafruit's LCD display connected to the Arduino sockets.

9.3.1 OLED C Click Display

The following HW modifications are required on the DA14695 PRO mother board/USB DK board:

- Add sockets to J15 and J16 for OLED C Click Display to be clicked on
- To control reset of the LCD, remove R309 and add R308 (1 KΩ)

9.3.2 Adafruit's 2.8" Thin Film Transistor LCD

The following HW modifications are required on the DA14695 PRO DK motherboard:

- Add sockets to J11, J12, J13, and J14 for Adafruit's LCD to be clicked on
- Remove R305 and add R307(0 Ω) from DA14695 PRO DK motherboard to provide power to the LCD from the motherboard's USB

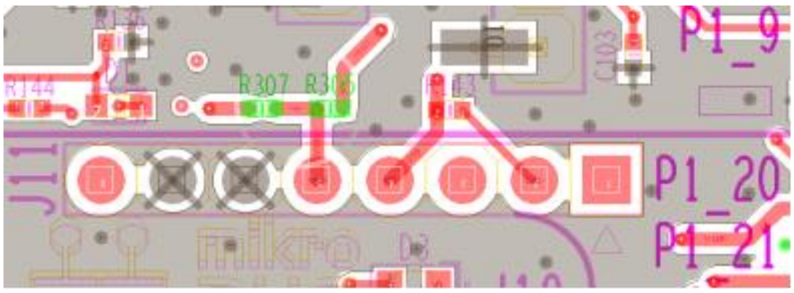


Figure 20: Remove R305 and R307 from DA14695 PRO DK Motherboard

- Connect TE pin (39) of the Adafruit LCD to P1_22 of the DA14695 PRO DK motherboard through cable
- To control reset of the LCD, remove R309 and add R308 (1 KΩ)

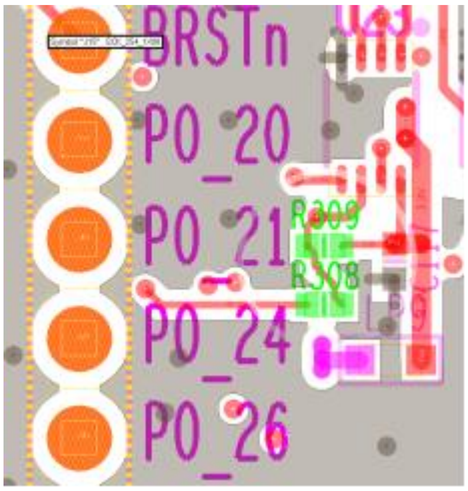


Figure 21: Remove R309 and R308 from DA14695 PRO DK Motherboard

9.4 Obtaining Logging Information

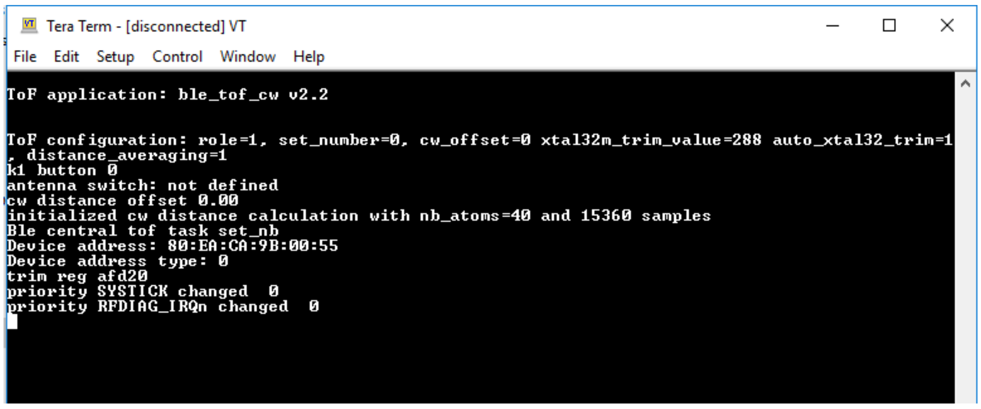
The logging option is selected at compile time via file `custom_config_qspi.h` (located under folder `ble_tof_cw/config/` in the workspace). For UART print enable: `#define CONFIG_RETARGET`. By default, UART printing is enabled.

Use your favorite UART terminal client (such as Putty, TeraTerm, RealTerm, Minicom, or others) to get connected to the first COM port assigned to the development board with the following settings:

- Baud rate: 115200
- Bits: 8
- Parity: None
- Stop Bits: 1

In the following screens you can see the logging information of a device which is configured as an Initiator:

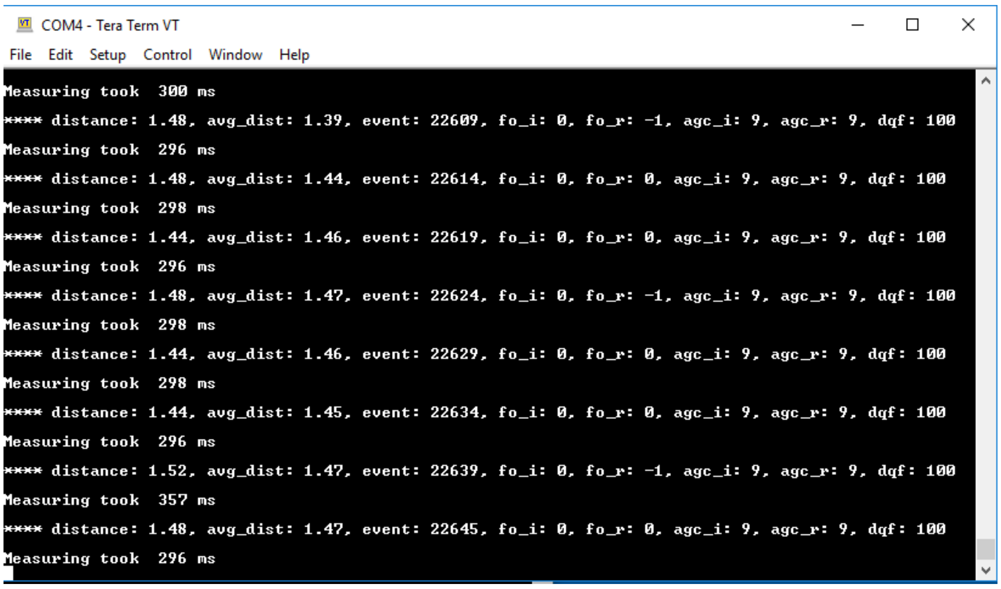
- After reset:


Tera Term - [disconnected] VT
File Edit Setup Control Window Help

```
ToF application: ble_tof_cw v2.2


ToF configuration: role=1, set_number=0, cw_offset=0 xtal32m_trim_value=288 auto_xtal32_trim=1
, distance_averaging=1
k1 button 0
antenna switch: not defined
cw distance offset 0.00
initialized cw distance calculation with nb_atoms=40 and 15360 samples
Ble central tof task set_nb
Device address: 80:EA:CA:9B:00:55
Device address type: 0
trim reg afd20
priority SYSTICK changed  0
priority RFDIAG_IRQn changed  0
```

Figure 22: Logging Information of Initiator after Reset

- After the device is connected to the Responder:



```
Measuring took  300 ms
**** distance: 1.48, avg_dist: 1.39, event: 22609, fo_i: 0, fo_r: -1, agc_i: 9, agc_r: 9, dqf: 100
Measuring took  296 ms
**** distance: 1.48, avg_dist: 1.44, event: 22614, fo_i: 0, fo_r: 0, agc_i: 9, agc_r: 9, dqf: 100
Measuring took  298 ms
**** distance: 1.44, avg_dist: 1.46, event: 22619, fo_i: 0, fo_r: 0, agc_i: 9, agc_r: 9, dqf: 100
Measuring took  296 ms
**** distance: 1.48, avg_dist: 1.47, event: 22624, fo_i: 0, fo_r: -1, agc_i: 9, agc_r: 9, dqf: 100
Measuring took  298 ms
**** distance: 1.44, avg_dist: 1.46, event: 22629, fo_i: 0, fo_r: 0, agc_i: 9, agc_r: 9, dqf: 100
Measuring took  298 ms
**** distance: 1.44, avg_dist: 1.45, event: 22634, fo_i: 0, fo_r: 0, agc_i: 9, agc_r: 9, dqf: 100
Measuring took  296 ms
**** distance: 1.52, avg_dist: 1.47, event: 22639, fo_i: 0, fo_r: -1, agc_i: 9, agc_r: 9, dqf: 100
Measuring took  357 ms
**** distance: 1.48, avg_dist: 1.47, event: 22645, fo_i: 0, fo_r: 0, agc_i: 9, agc_r: 9, dqf: 100
Measuring took  296 ms
```

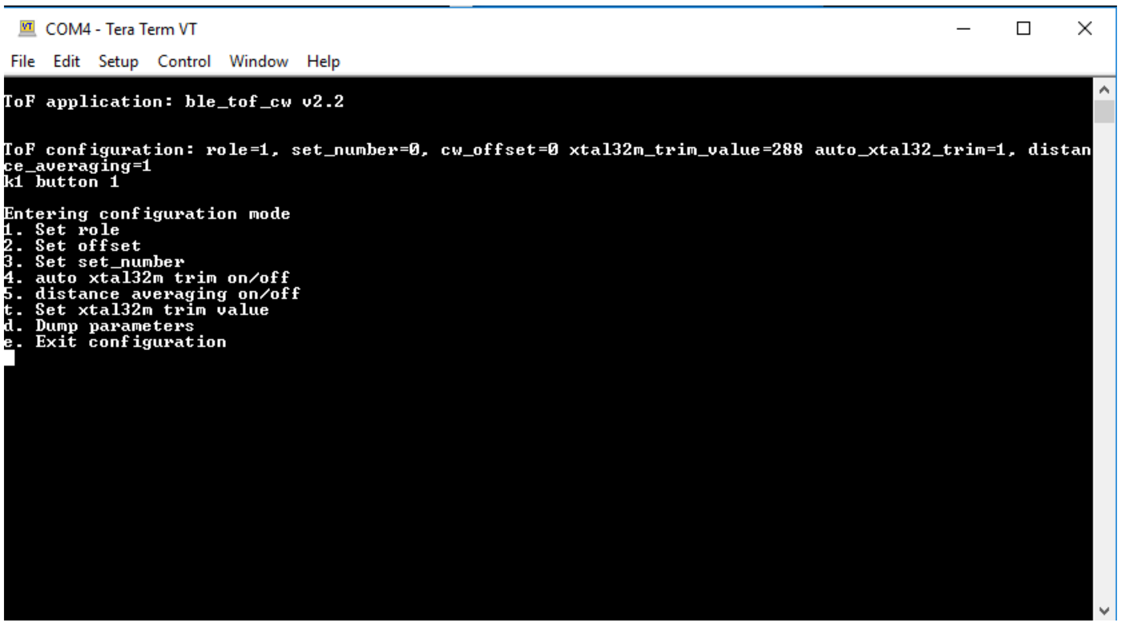
Figure 23: Logging Information of Initiator after Being Connected to Responder

The logging output contains the following information:

- `distance` is the actual distance measured in meters after the `cw_offset` has been applied
- `avg_dist` is the average distance of the last four valid measurements (with `dqf` :100) when distance averaging is enabled
- `event` is the connection event counter of the event that a radio measurement has taken place
- `fo_i` and `fo_r` are the frequency offsets in kHz for the Initiator and the Responder
- `agc_i` and `agc_r` are the AGC gain values of the Initiator and the Responder, respectively, used during the data acquisition
- `dqf` is a distance quality factor based on the frequency offset values of both sides. It should be 100 for a good measurement
- Measuring time is only printed in the logging information of the Initiator and is the time from sending the start command until the measurement is stopped

9.5 Configuration Mode

If K1 button is pressed during RESET, the device enters the configuration mode. The configuration menu is displayed (Figure 24), and users have the option to change various nonvolatile parameters. To exit this mode, select the '**e. Exit configuration**' command. When this command is selected, the device resets in normal mode with the new nonvolatile configuration parameters. See section 7.2 for a description of these parameters.



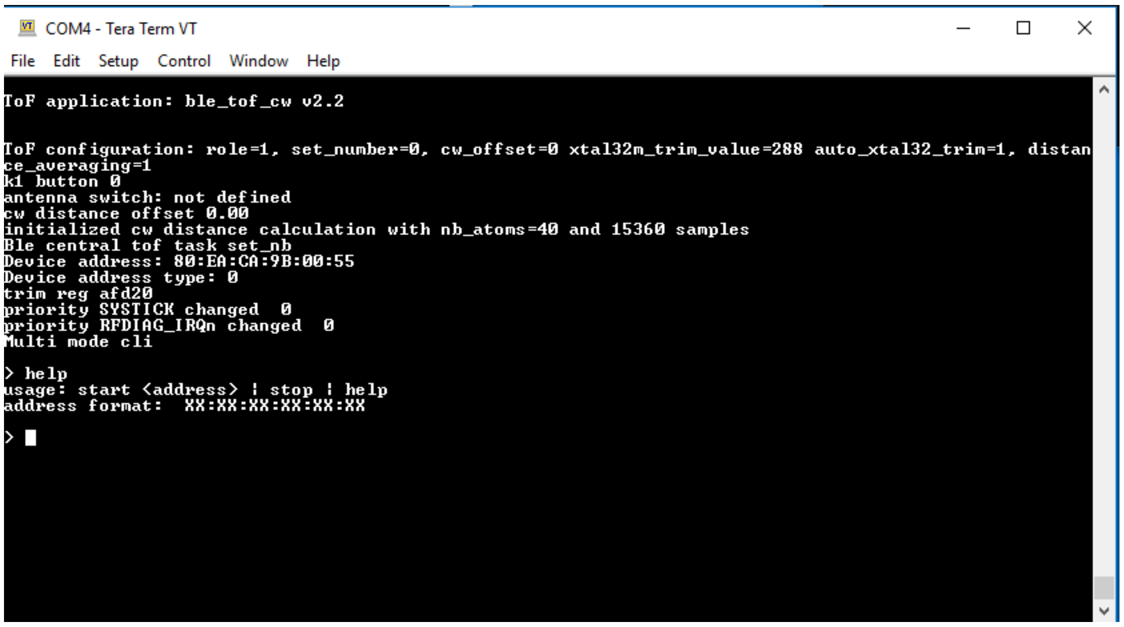
```
ToF application: ble_tof_cw v2.2

ToF configuration: role=1, set_number=0, cw_offset=0 xtal32m_trim_value=288 auto_xtal32_trim=1, distan
ce_averaging=1
k1 button 1

Entering configuration mode
1. Set role
2. Set offset
3. Set set_number
4. auto xtal32m trim on/off
5. distance averaging on/off
t. Set xtal32m trim value
d. Dump parameters
e. Exit configuration
```

Figure 24: Configuration Mode

9.6 CLI Mode

The CLI is a special mode for the Initiator device. This mode is enabled if the code is built with `#define CLI_MODE (1)` in file `custom_config_qspi.h`.

After reset, the Initiator does not scan but waits for cli commands set address, start, stop, and acts accordingly.



```
ToF application: ble_tof_cw v2.2

ToF configuration: role=1, set_number=0, cw_offset=0 xtal32m_trim_value=288 auto_xtal32_trim=1, distan
ce_averaging=1
k1 button 0
antenna switch: not defined
cw distance offset 0.00
initialized cw distance calculation with nb_atoms=40 and 15360 samples
Ble central tof task set_nb
Device address: 80:EA:CA:9B:00:55
Device address type: 0
trim reg afd20
priority SYSTICK changed  0
priority RFDIAG_IRQn changed  0
Multi mode cli

> help
usage: start <address> | stop | help
address format:  XX:XX:XX:XX:XX:XX

>
```

Figure 25: CLI Mode for Initiator Device

10 Limitations

The following limitations apply to the current implementation:

- The system must always stay active in both Initiator and Responder devices in order to reduce the IRQ latency that is important for the radio measurement synchronization
- There is no BLE activity between connection events. This way we know that the triggering of RF Diagnostics IRQ by `TX_ENABLE`/ `SYNC_WORD_FOUND` in Initiator/Responder is related to the connection event and that the CMAC sleeps between the connection events. Because of this restriction only one connection with no synchronous advertising/scanning can be supported during measurements
- Raw DMA: In the current implementation, the DMA SDK code has been bypassed because DMA takes place at a critical time during data acquisition and we would like to avoid the latencies imposed by the DMA SDK code. In the future the DMA SDK code will be integrated
- BLE traffic: The transmission of the START request is time critical, so if the transmission is delayed because of the existing BLE traffic, the distance measurement will fail due to the instant passed message.

Revision History

Revision	Date	Description
1.0	20-Mar-2019	Initial version.

Status Definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

Disclaimer

Contacting Dialog Semiconductor

United Kingdom (Headquarters)
Dialog Semiconductor (UK) LTD
Phone: +44 1793 757700

Germany
Dialog Semiconductor GmbH
Phone: +49 7021 805-0

The Netherlands
Dialog Semiconductor B.V.
Phone: +31 73 640 8822

Email:
enquiry@diasemi.com

North America
Dialog Semiconductor Inc.
Phone: +1 408 845 8500

Japan
Dialog Semiconductor K. K.
Phone: +81 3 5769 5100

Taiwan
Dialog Semiconductor Taiwan
Phone: +886 281 786 222

Web site:
www.dialog-semiconductor.com

Hong Kong
Dialog Semiconductor Hong Kong
Phone: +852 2607 4271

Korea
Dialog Semiconductor Korea
Phone: +82 2 3469 8200

China (Shenzhen)
Dialog Semiconductor China
Phone: +86 755 2981 3669

China (Shanghai)
Dialog Semiconductor China
Phone: +86 21 5424 9058