

# User Manual

## SmartBond™ Wireless Ranging SDK

UM-B-137

### Abstract

*This document provides basic information to help developers get familiar with the DA1469x Wireless Ranging application and modify or create a new Wireless Ranging application based on it.*

---

---

## SmartBond™ Wireless Ranging SDK

### Contents

<b>Abstract</b> .....	<b>1</b>
<b>Contents</b> .....	<b>2</b>
<b>Figures</b> .....	<b>3</b>
<b>Tables</b> .....	<b>3</b>
<b>1 Terms and Definitions</b> .....	<b>4</b>
<b>2 References</b> .....	<b>4</b>
<b>3 Introduction</b> .....	<b>5</b>
<b>4 Setup</b> .....	<b>5</b>
4.1 Hardware .....	5
4.1.1 Connecting Hardware and Powering On .....	6
4.1.2 Buttons and Switches .....	6
4.1.3 LCD Support .....	7
4.2 Development Environment .....	7
4.2.1 To Install the Development Environment .....	7
4.2.2 Import and Build the Wireless Ranging Project .....	7
4.2.3 Burning the Image into Flash Memory .....	10
4.2.4 Build Configurations .....	10
4.3 Running the Wireless Ranging Application .....	11
4.3.1 To Get Logging Information .....	11
4.3.2 Configuration Mode .....	12
4.3.3 CLI Mode .....	14
<b>5 Wireless Ranging Application</b> .....	<b>15</b>
5.1 Dialog Tone Exchange (DTE) Overview .....	15
5.1.1 Application Data Flow .....	16
5.1.2 Software Architecture .....	18
5.2 Dialog Tone Exchange .....	18
5.2.1 DTE Configuration .....	19
5.2.2 DTE Enable .....	19
5.3 Bluetooth LE Activity .....	20
5.3.1 Central Role .....	20
5.3.2 Peripheral Role .....	20
5.3.3 Bluetooth LE DTE Data Service .....	20
5.3.4 Measurement Cycle and Application State .....	20
5.3.5 Other Bluetooth LE Activity .....	21
5.3.6 Errors / Statistics .....	21
5.4 Distance Calculations (Phase/IFFT) .....	22
5.4.1 IFFT Based Distance Estimation (Default) .....	24
5.4.2 Phase Based Distance Estimation .....	25
5.5 Hardware Support .....	26
5.6 User Interface .....	26
5.7 CLI .....	26
5.7.1 CLI Commands .....	26
5.8 External Interface .....	27

**SmartBond™ Wireless Ranging SDK**

5.9	Configuration Options/Nonvolatile Parameters.....	28
5.9.1	Compile Time Configuration Switches.....	28
5.9.2	Non-Volatile Parameters.....	29
<b>6</b>	<b>Bluetooth LE IQ Data Applications.....</b>	<b>30</b>
6.1	Application Overview.....	30
6.2	RFMON Interface.....	30
6.2.1	RFMON Configuration.....	30
6.2.2	RFMON Enable.....	31
6.2.3	RFMON Filtering.....	31
6.3	External Interface.....	31
6.4	CLI Mode.....	32
<b>7</b>	<b>IQ Data Reports.....</b>	<b>32</b>
	<b>Revision History.....</b>	<b>34</b>

**Figures**

Figure 1:	DA14695 PRO DK.....	5
Figure 2:	DA14695 Wireless Ranging USB DK HW Components.....	6
Figure 3:	Import a Project into the Workspace.....	8
Figure 4:	Select the ble_range_dte Project.....	8
Figure 5:	Build Configurations for ble_range_dte Project.....	9
Figure 6:	Build Log.....	10
Figure 7:	Programming the Flash Image.....	10
Figure 8:	Programming Flash - Log Output.....	10
Figure 9:	Logging Information of Initiator after Reset.....	11
Figure 10:	Logging Information of Initiator after Being Connected to a Responder.....	12
Figure 11:	Configuration Mode.....	13
Figure 12:	CLI Mode for Initiator Device.....	14
Figure 13:	Distance Measurement Timeline.....	15
Figure 14:	Application Data Flow.....	17
Figure 15:	Wireless Ranging Application Software Modules.....	18
Figure 16:	Measurement Statistics Report.....	22
Figure 17:	Distance Calculations.....	23

**Tables**

Table 5-1:	Configuration Switches.....	28
Table 6-1:	RFMON_OUTPUT_MODE_IQ9 Format.....	30
Table 7-1:	Sampling Buffer Structure.....	33

---

**SmartBond™ Wireless Ranging SDK****1 Terms and Definitions**

ADC	Analog-to-Digital Converter
AGC	Automatic Gain Control
BLE	Bluetooth® Low Energy
CLI	Command-Line Interface
CMAC	Configurable Medium Access Controller
DK	Development Kit
DTE	Dialog Tone Exchange
GAP	Generic Access Profile
GATT	Generic Attribute Profile
IFFT	Inverse Fast Fourier Transform
IQ	In-phase and Quadrature
ISM	Industrial, Scientific, and Medical (radio band)
LCD	Liquid-Crystal Display
LE	Low Energy
MTU	Maximum Transmission Unit
NVM	Non-Volatile Memory
PDU	Protocol Data Unit
SDK	Software Development Kit
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.

SmartBond™ Wireless Ranging SDK

### 3 Introduction

This document describes the Wireless Ranging application on Dialog Semiconductor's DA1469x Bluetooth® Low Energy family products. It gives the basic mechanisms involved to help developers understand the source code.

Wireless 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 Setup

This section describes the supported hardware platforms and the necessary steps to build and run the Wireless Ranging application.

#### 4.1 Hardware

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

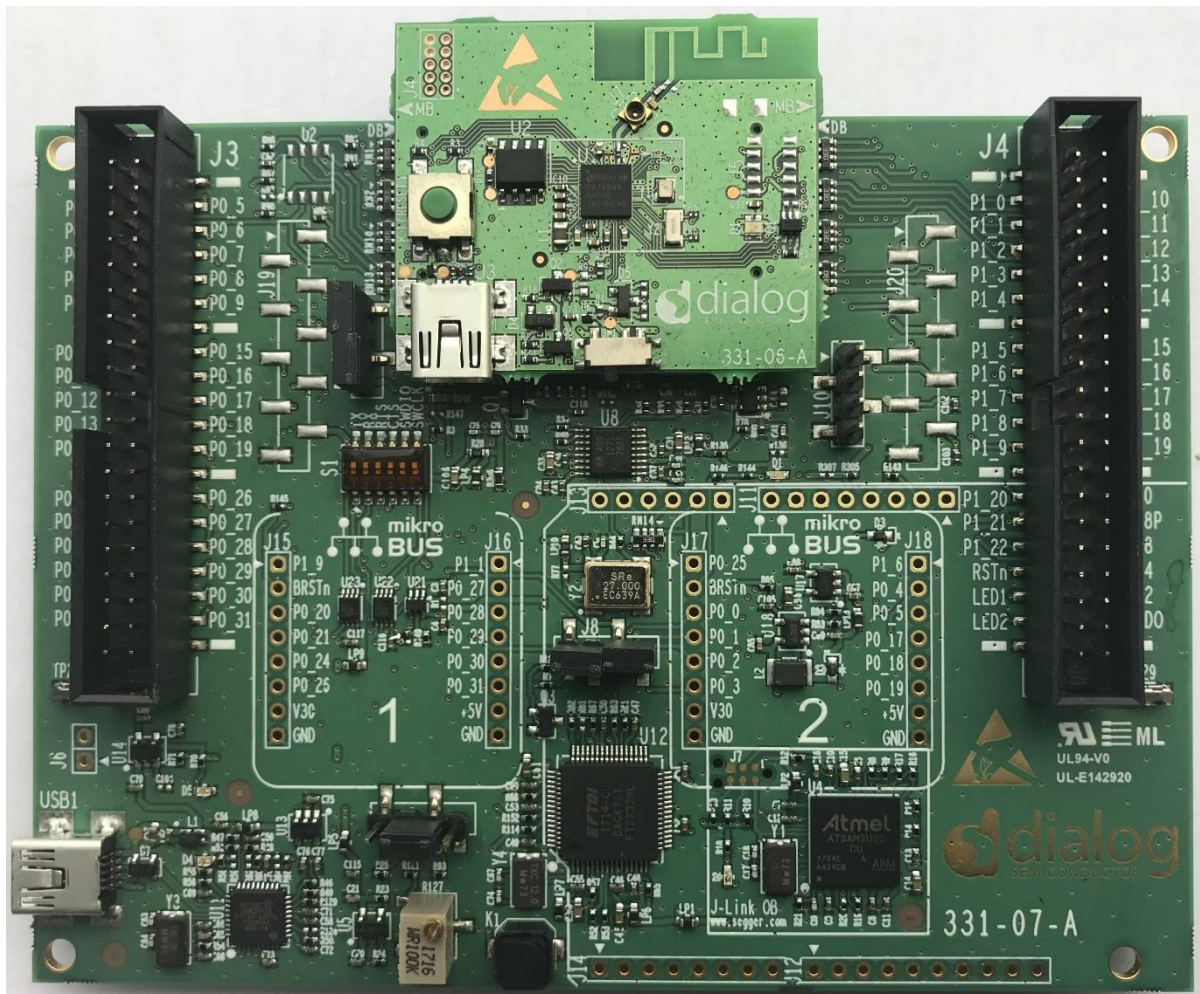


Figure 1: DA14695 PRO DK

## SmartBond™ Wireless Ranging SDK

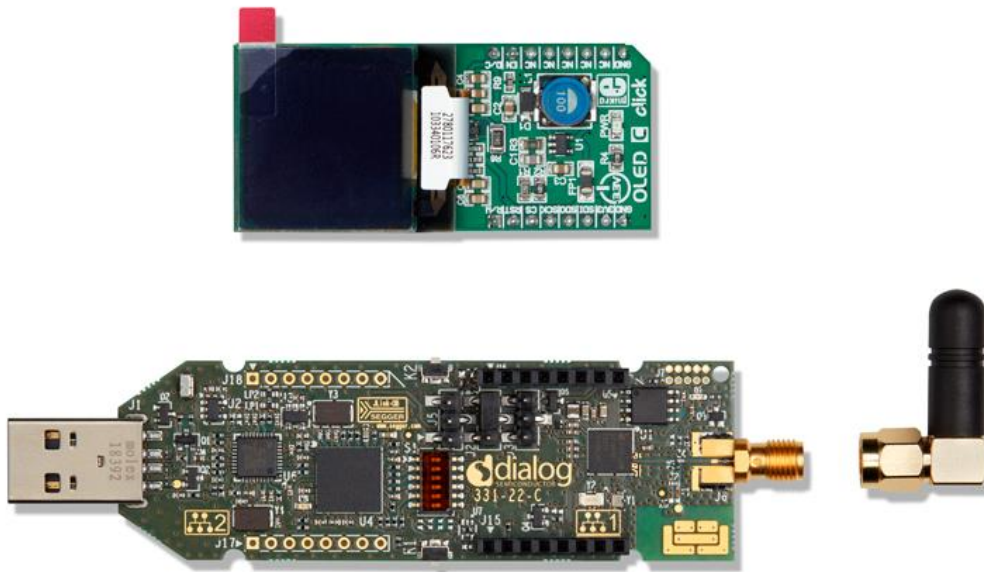


Figure 2: DA14695 Wireless Ranging USB DK HW Components

### 4.1.1 Connecting Hardware and Powering On

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

To flash the application FW image, please follow the instructions in sections 4.2.1 and 4.2.2 **first**.

The boards are powered via USB. To power on, use the provided USB cable to connect the DA14695 PRO DK to a laptop or desktop.

When the DA14695 PRO DK is to be connected to a power adapter or power bank, you need to solder a header on J6 and install a jumper there to bypass the enumeration with a USB host.

If the DA14695 USB DK is to be connected to a power adapter or power bank, please remove D9, the tiny diode near K2<sup>1</sup>.

After power-on via USB, the DA14695 PRO or USB DK board programmed with the "range" application should then enter Scanning or Advertising mode based on the configuration of the non-volatile role parameter.

### 4.1.2 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
  - K2: HW resets the DA1469x
  - K1: Device enters configuration mode if button is pressed during reset

On both DKs, if you push the K1 button after reset, a console printout with the range measurement statistics collected so far is provided.

<sup>1</sup> HW modifications have already been applied in-house for all delivered DA14695 Wireless Ranging DKs

---

## SmartBond™ Wireless Ranging SDK

### 4.1.3 LCD Support

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

#### 4.1.3.1 OLED C Click Display

The following HW modification is 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<sup>2</sup>

## 4.2 Development Environment

This section describes how to set up the development environment of the DA1469x Wireless Ranging application.

### 4.2.1 To Install the Development Environment

1. Unzip the Wireless Ranging SDK release zip file (for example, WiRa\_10.440.8.4) into a known location on your drive.
2. All software development is based on [SmartSnippets™ Studio](#). Please see *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 Wireless Ranging SDK release.
4. Select the IDE from **Tools** section of the **Welcome** screen. The regular Eclipse environment view should now appear.

### 4.2.2 Import and Build the Wireless Ranging Project

1. To import the required projects, do the following steps:
  - a. Select **File > Import**.
  - b. Select **General > Existing Projects into Workspace** >click **Next** (Figure 3) > **Select root directory** (Figure 4).
  - c. Browse to the location of the release.
  - d. Deselect all projects.
  - e. From the project list select only the `ble_range_dte` and the `python_scripts` projects.
  - f. Click **Finish**.

<sup>2</sup> HW modifications have already been applied in-house for all delivered DA14695 Wireless Ranging DKs

## SmartBond™ Wireless Ranging SDK

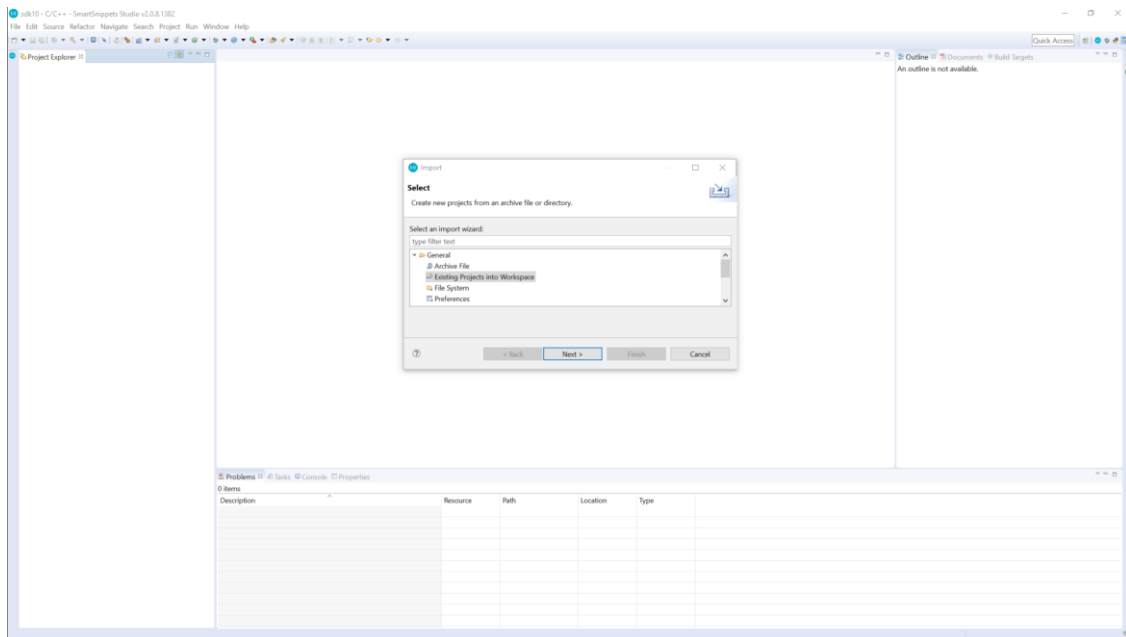


Figure 3: Import a Project into the Workspace

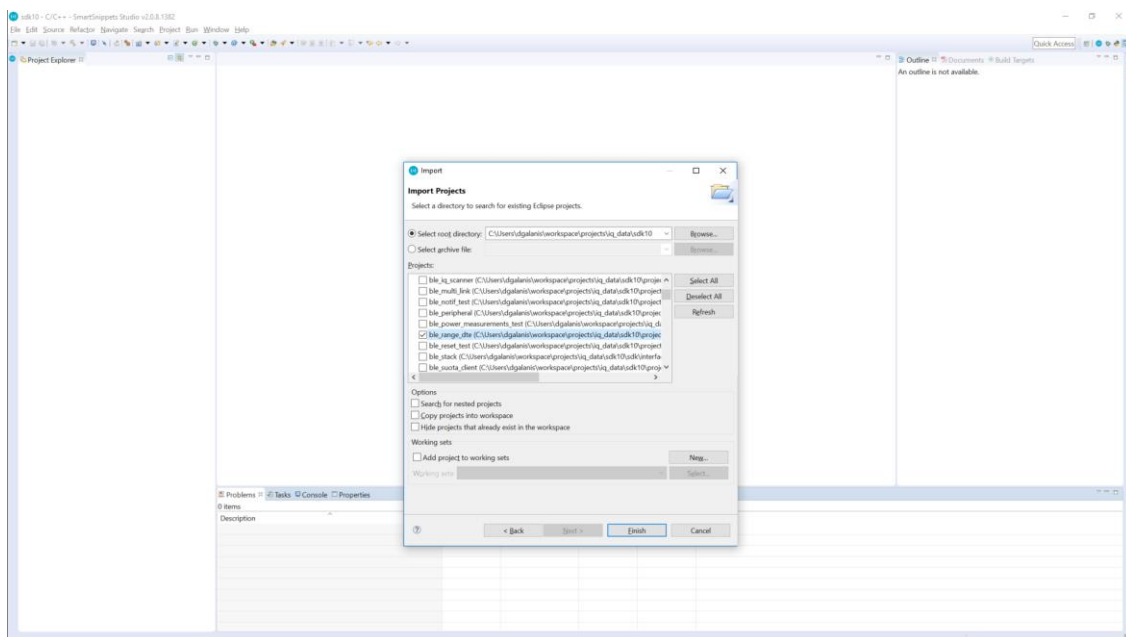


Figure 4: Select the ble\_range\_dte Project

2. Select a build configuration and build the ble\_range\_dte project (Figure 5). Right-click on the project name to get the project menu and select **Build Configurations > Set Active** to see configuration options.



SmartBond™ Wireless Ranging SDK

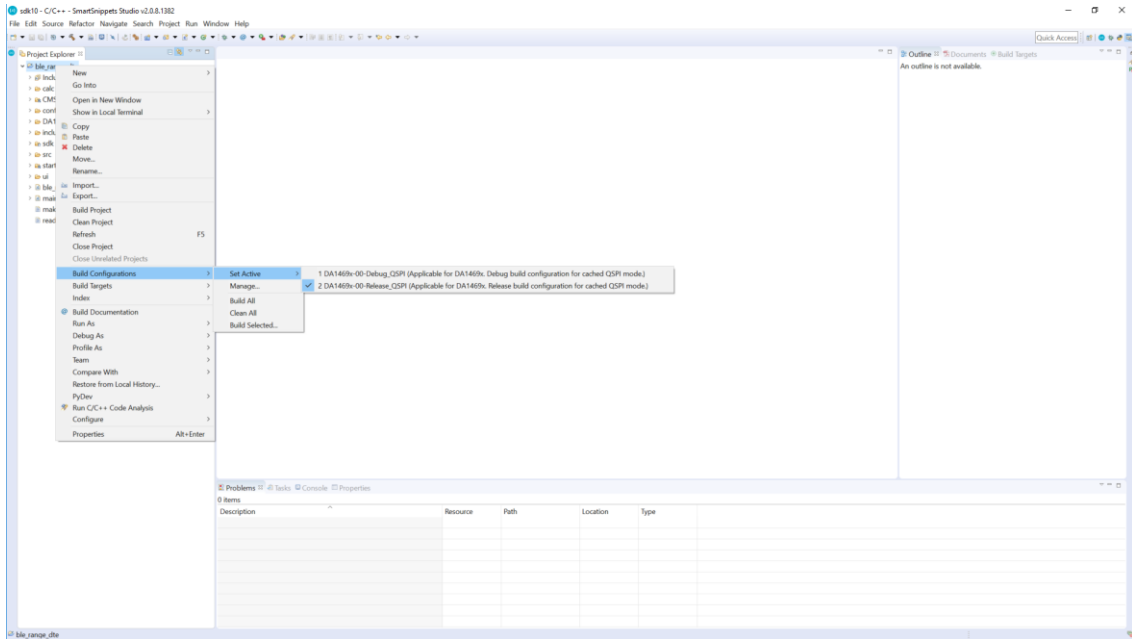


Figure 5: Build Configurations for b1e\_range\_dte Project

## SmartBond™ Wireless Ranging SDK

3. A successful build for the release configuration of the `ble_range_dte` project gives the result in the console window as shown in [Figure 6](#).



```

CDT Global Build Console

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

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

Invoking: Cross ARM GNU Print Size
text  data  bss  dac  hex  Filename
392340 884 247988 641132 9c8ec ble_range_dte.elf
Finished building: ble_range_dte.siz

17:09:14 Build Finished (took 36s.280ms)
    
```

Figure 6: Build Log

### 4.2.3 Burning the Image into Flash Memory

1. To write the generated image into the Flash memory, run the external script `program_qspi_jtag` ([Figure 7](#)).

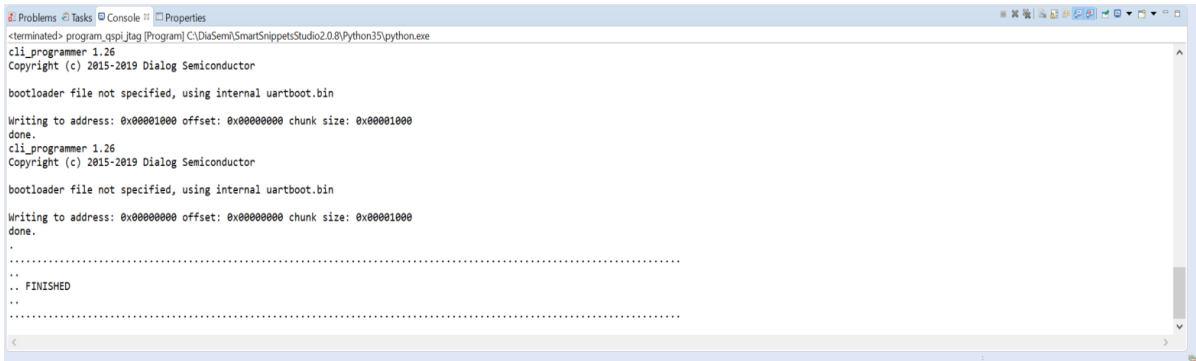


Figure 7: Programming the Flash Image

#### NOTE

Make sure to first click on the `ble_range_dte` 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 that `ble_range_dte` is selected and not any other project.

2. Compare your log output of the DA1469x Wireless Ranging application with that shown in [Figure 8](#) to check that your programming is successful.



```

<terminated> program_qspi_jtag [Program] C:\DiSemi\SmartSnippetsStudio2.0.8\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 8: Programming Flash - Log Output

### 4.2.4 Build Configurations

The DA1469x Wireless Ranging application is compatible with both the DA14695 PRO and USB board. It also supports the option to add an OLED on the board. The related configurations are controlled by define statements in file `custom_config_qspi.h` (to be found in folder `ble_range_dte/config/` in the workspace).

## SmartBond™ Wireless Ranging SDK

### 4.2.4.1 Boards

The default build configuration is for the DA1469x PRO development kit (DK) board. The USB DK board is also supported. But in order to build firmware for the USB board you need to set the related define from 1 (default) to 0:

```
#define dg_configUSE_ProDK ( 0 )
```

### 4.2.4.2 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. To enable, set the related LCD define for OLED display from 0 (default) to 1:

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

## 4.3 Running the Wireless Ranging Application

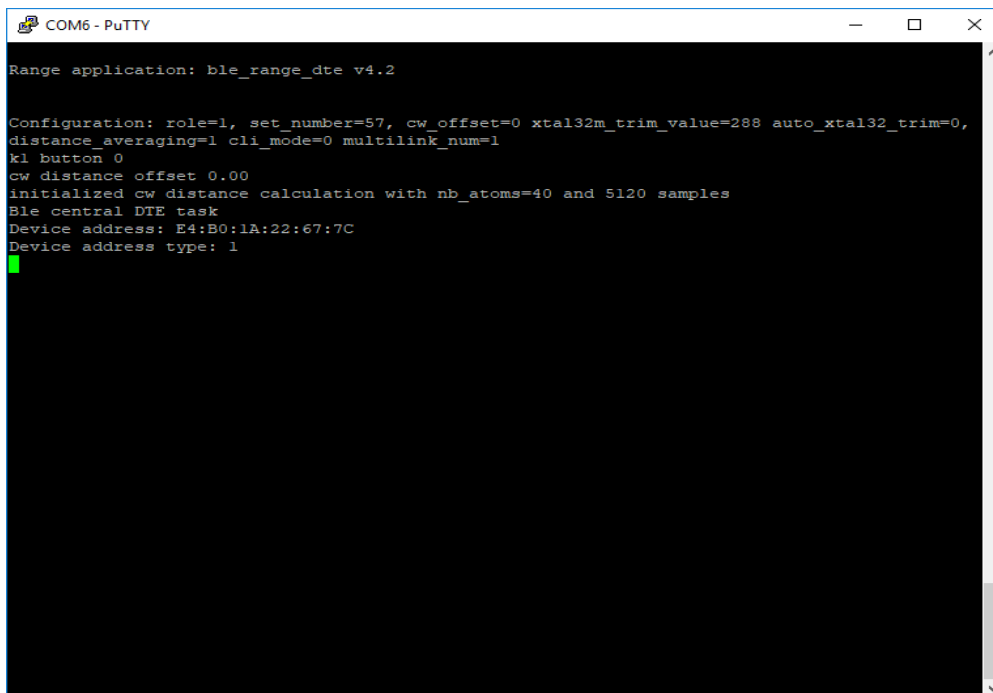
### 4.3.1 To Get Logging Information

The logging option is selected at compile time via file `custom_config_qspi.h` (located under folder `ble_range_dte/config/` in the workspace). For UART print, define statement `#define CONFIG_RETARGET` is used. 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

As described in 5.1, DTE ranging devices operate in pairs of nodes labelled as Initiator and Responder. In the following screenshots you can see the logging information of a device, which is configured as an Initiator after reset (Figure 9) and after being connected to a Responder (Figure 10).

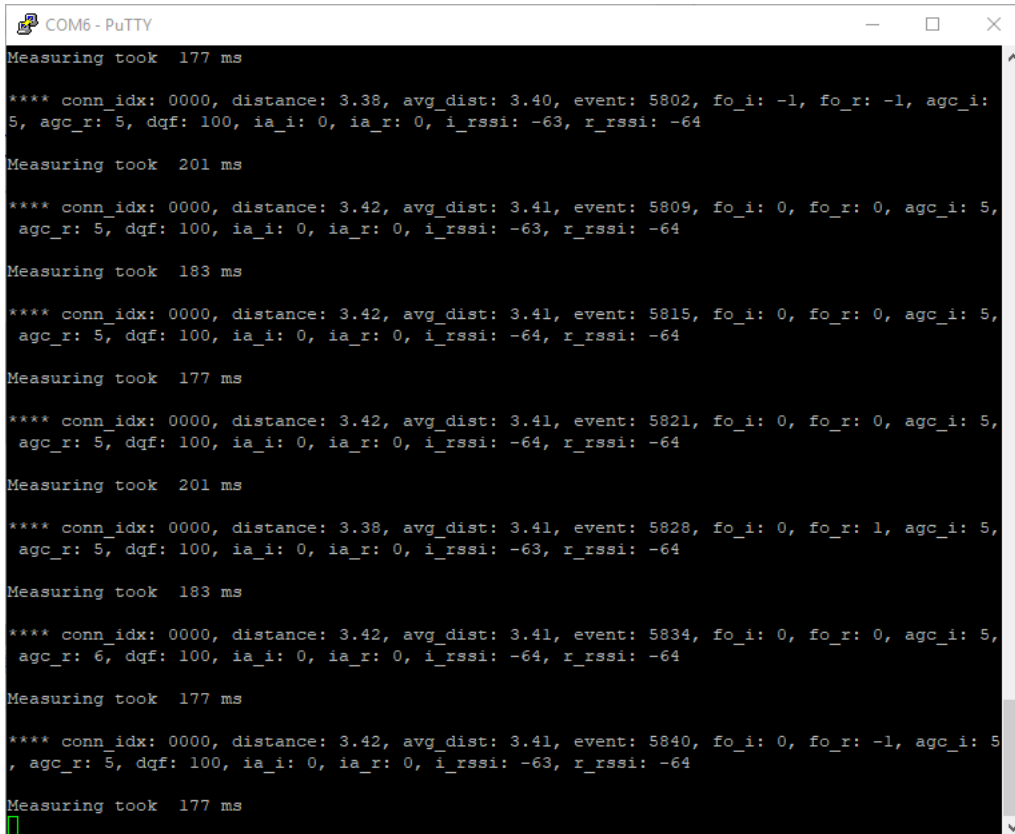


```
COM6 - PuTTY
Range application: ble_range_dte v4.2

Configuration: role=1, set_number=57, cw_offset=0 xtal32m_trim_value=288 auto_xtal32_trim=0,
distance_averaging=1 cli_mode=0 multilink_num=1
kl button 0
cw distance offset 0.00
initialized cw distance calculation with nb_atoms=40 and 5120 samples
Ble central DTE task
Device address: E4:B0:1A:22:67:7C
Device address type: 1
```

**Figure 9: Logging Information of Initiator after Reset**

## SmartBond™ Wireless Ranging SDK



```

COM6 - PuTTY
Measuring took 177 ms
**** conn_idx: 0000, distance: 3.38, avg_dist: 3.40, event: 5802, fo_i: -1, fo_r: -1, agc_i: 5, agc_r: 5, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -63, r_rssi: -64
Measuring took 201 ms
**** conn_idx: 0000, distance: 3.42, avg_dist: 3.41, event: 5809, fo_i: 0, fo_r: 0, agc_i: 5, agc_r: 5, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -63, r_rssi: -64
Measuring took 183 ms
**** conn_idx: 0000, distance: 3.42, avg_dist: 3.41, event: 5815, fo_i: 0, fo_r: 0, agc_i: 5, agc_r: 5, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -64, r_rssi: -64
Measuring took 177 ms
**** conn_idx: 0000, distance: 3.42, avg_dist: 3.41, event: 5821, fo_i: 0, fo_r: 0, agc_i: 5, agc_r: 5, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -64, r_rssi: -64
Measuring took 201 ms
**** conn_idx: 0000, distance: 3.38, avg_dist: 3.41, event: 5828, fo_i: 0, fo_r: 1, agc_i: 5, agc_r: 5, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -63, r_rssi: -64
Measuring took 183 ms
**** conn_idx: 0000, distance: 3.42, avg_dist: 3.41, event: 5834, fo_i: 0, fo_r: 0, agc_i: 5, agc_r: 6, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -64, r_rssi: -64
Measuring took 177 ms
**** conn_idx: 0000, distance: 3.42, avg_dist: 3.41, event: 5840, fo_i: 0, fo_r: -1, agc_i: 5, agc_r: 5, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -63, r_rssi: -64
Measuring took 177 ms

```

Figure 10: Logging Information of Initiator after Being Connected to a Responder

The output log contains the following information:

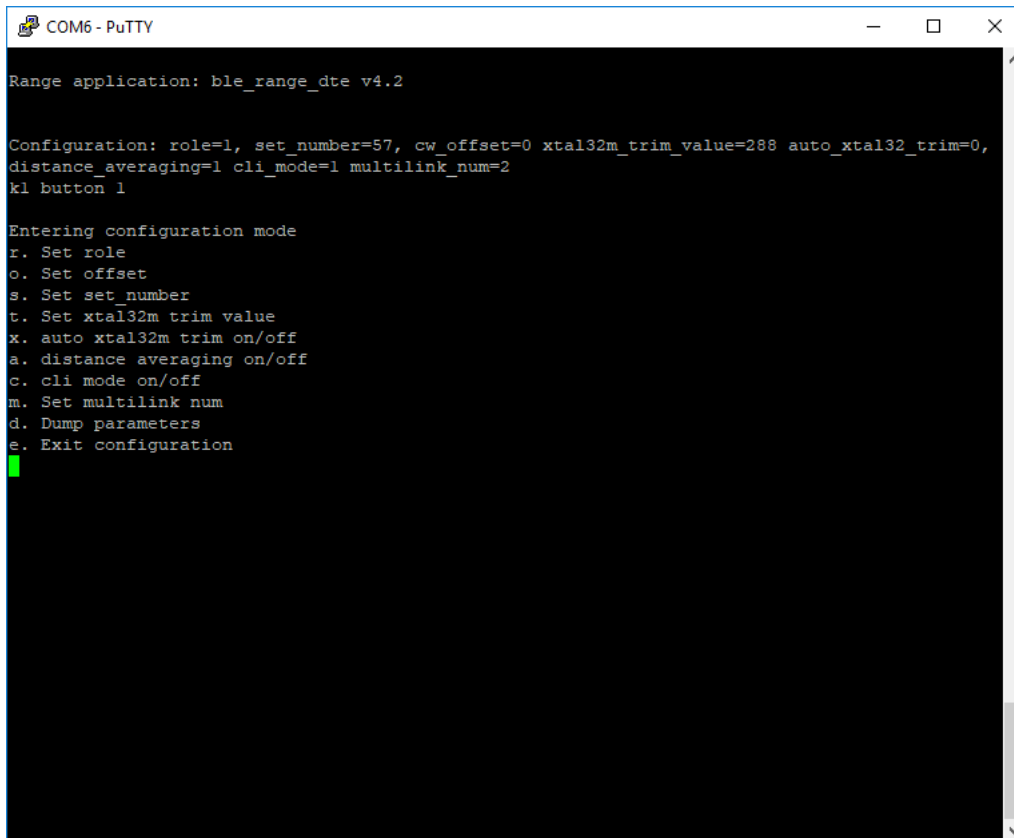
- `conn_idx` is the BLE connection that provides the anchor times for the ranging information related radio measurements
- `distance` is the actual distance measured in meters after the `cw_offset` is applied
- `avg_dist` is the average distance of the last four valid measurements (with `dqf:100`) when 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
- `ia_i` and `ia_r` are the numbers of Tone Exchange steps with invalid IQ data as identified during the IQ data acquisition phase. Currently, if there are any such steps, the whole measurement is dropped (i.e. `dqf = 0`)
- `i_rssi` and `r_rssi` are the RSSI values of the last BLE reception of the Initiator and the Responder respectively
- 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

### 4.3.2 Configuration Mode

If the K1 button is pressed during RESET, the device enters the configuration mode. The configuration menu is displayed (Figure 11), and users have the option to change various non-volatile parameters. To exit this mode, select the **'e. Exit configuration'** command. When this

## SmartBond™ Wireless Ranging SDK

command is selected, the device resets in normal mode with the new non-volatile configuration parameters. See section 5.9.2 for a description of these parameters.



```
COM6 - PuTTY
Range application: ble_range_dte v4.2

Configuration: role=1, set_number=57, cw_offset=0 xtal32m_trim_value=288 auto_xtal32_trim=0,
distance_averaging=1 cli_mode=1 multilink_num=2
kl button 1

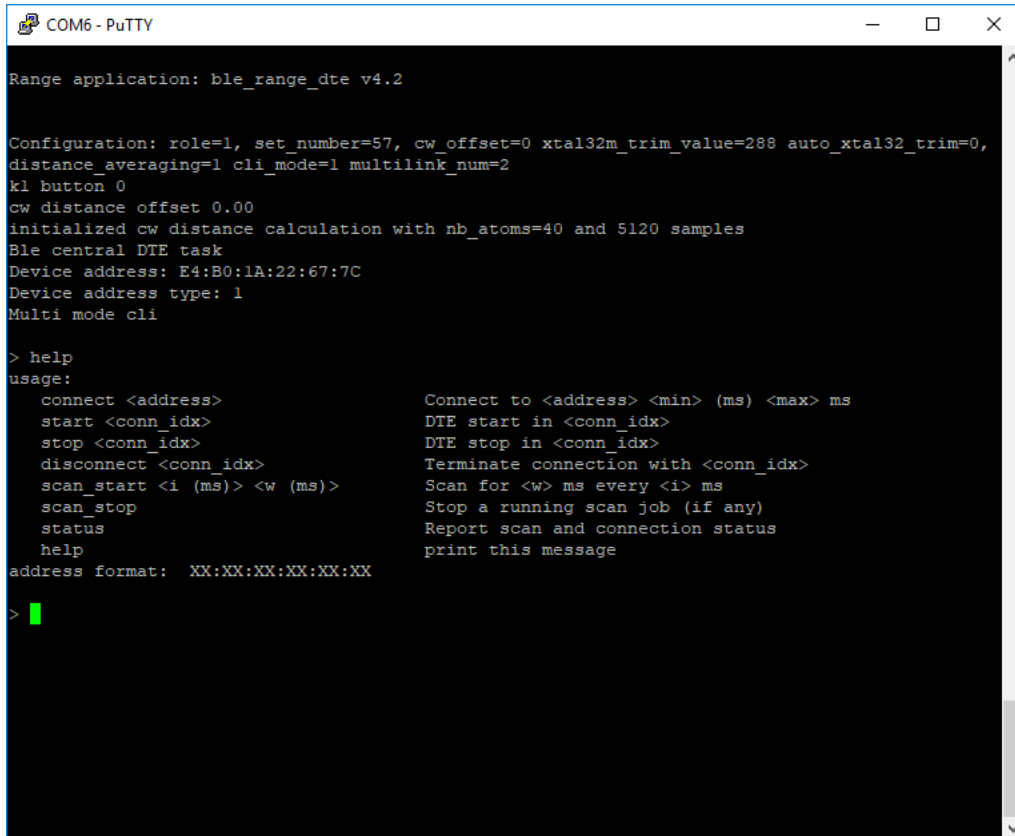
Entering configuration mode
r. Set role
o. Set offset
s. Set set_number
t. Set xtal32m trim value
x. auto xtal32m trim on/off
a. distance averaging on/off
c. cli mode on/off
m. Set multilink num
d. Dump parameters
e. Exit configuration
█
```

Figure 11: Configuration Mode

## SmartBond™ Wireless Ranging SDK

### 4.3.3 CLI Mode

The CLI is a special mode for the Initiator device, which may be enabled in the configuration mode, as described in section 4.3.2 above, where the setting for `cli_mode` is set to `on`. When operating in CLI mode, the Initiator waits for the user to issue any of the set of available `cli` commands (Figure 12) whose detailed description and functionality is given in 5.7.



```
COM6 - PuTTY
Range application: ble_range_dte v4.2

Configuration: role=1, set_number=57, cw_offset=0 xtal32m_trim_value=288 auto_xtal32_trim=0,
distance_averaging=1 cli_mode=1 multilink_num=2
kl button 0
cw distance offset 0.00
initialized cw distance calculation with nb_atoms=40 and 5120 samples
Ble central DTE task
Device address: E4:B0:1A:22:67:7C
Device address type: 1
Multi mode cli

> help
usage:
  connect <address>           Connect to <address> <min> (ms) <max> ms
  start <conn_idx>           DTE start in <conn_idx>
  stop <conn_idx>            DTE stop in <conn_idx>
  disconnect <conn_idx>      Terminate connection with <conn_idx>
  scan_start <i (ms)> <w (ms)> Scan for <w> ms every <i> ms
  scan_stop                  Stop a running scan job (if any)
  status                     Report scan and connection status
  help                       print this message
address format: XX:XX:XX:XX:XX:XX

> █
```

Figure 12: CLI Mode for Initiator Device

## 5 Wireless Ranging Application

This section describes the software modules of the application and how Bluetooth LE and Dialog Tone Exchange (DTE) are used in the application to get distance measurements. The application is in folder `projects\dk_apps\range`.

### 5.1 Dialog Tone Exchange (DTE) Overview

The DA1469x Wireless Ranging application calculates the distance between two devices with the In-phase and Quadrature outputs of the RF-ADC (IQ data) from the continuous wave signals received from the two devices. To achieve this, both devices need to exchange continuous wave signals on a pre-defined set of frequencies for a pre-defined time (DTE). The device that initiates this procedure is called the Initiator and the other device is the Responder. The signal magnitude  $A$  and phase angle  $\emptyset$  is calculated as:

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

$$\emptyset = \arctan(Q/I) \tag{2}$$

Based on the IQ data, the DA1469x Wireless Ranging application calculates the phase of the signal for each frequency that is 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 for both the DTE synchronization and the exchange of the preliminary results with phase and amplitude information. A custom BLE DTE Data service (5.3.3) is implemented for the exchange of the intermediate results.

The tone exchange synchronization is based on TX and RX timestamps related to the last packet transmission of a BLE connection event. The DTE signal transmissions start at a predefined time offset after the last BLE packet transmission. The connection interval needs to be greater than the tone preparation and exchange duration (~17 ms) so that the tone exchange does not overlap with the next connection event.

This exchange takes place in specific connection events. In order to achieve event synchronization, the BLE connection event counter is used. In a BLE connection, the role of the Initiator device is assigned to the central node and the Responder is assigned to the peripheral. The Central device transmits to the peripheral device a 'start request' control packet that contains an event counter value for a connection event instance in the future. At the end of the event that matches that counter, the tone exchange will take place; assuming that the peripheral has enough time to receive this information and prepare for the tone exchange before that instance passes.

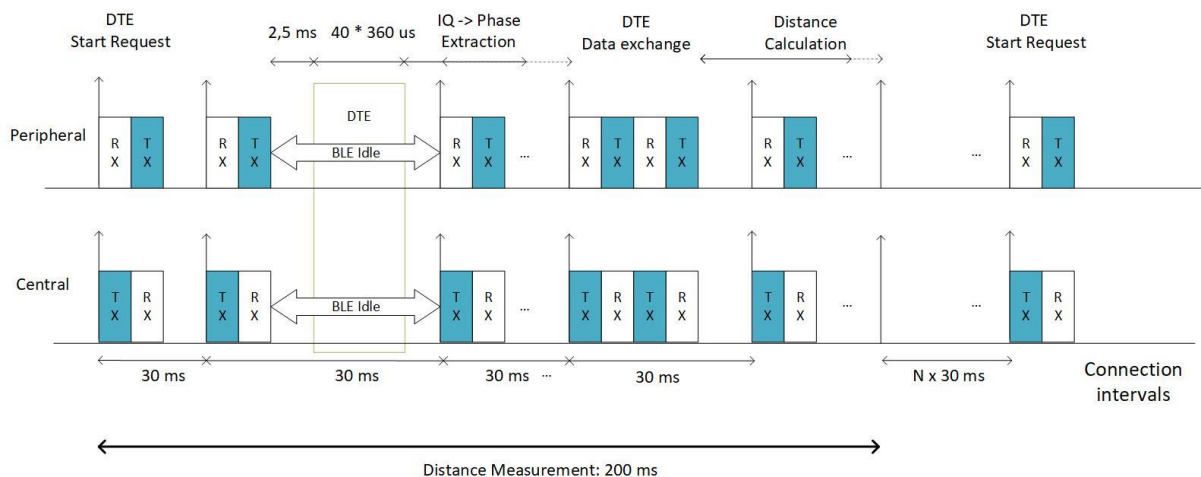


Figure 13: Distance Measurement Timeline

---

## SmartBond™ Wireless Ranging SDK

### 5.1.1 Application Data Flow

The Wireless Ranging 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 DTE Data service UUID and the set number in the advertising data. The Central device starts scanning, looking for the DTE Data 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 device sends a Link Layer START measurement request with a connection event counter that refers to a future instance.
2. When that instance comes, the two devices synchronize and carry out the tone exchange.
3. Both devices then perform the phase calculations.
4. Both devices send the intermediate results to each other using the BLE DTE Data service 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 possibility is one-sided distance calculation: only one side sends the intermediate results and the other side combines the local and remote results, calculates the distance, and sends the distance measurement to the other side. This option is however 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 commands for DTE start/ stop and BLE connect/disconnect scan start/stop. After a device has been connected, the start command can be used to start measurements using the BLE connection index as a parameter.



SmartBond™ Wireless Ranging SDK

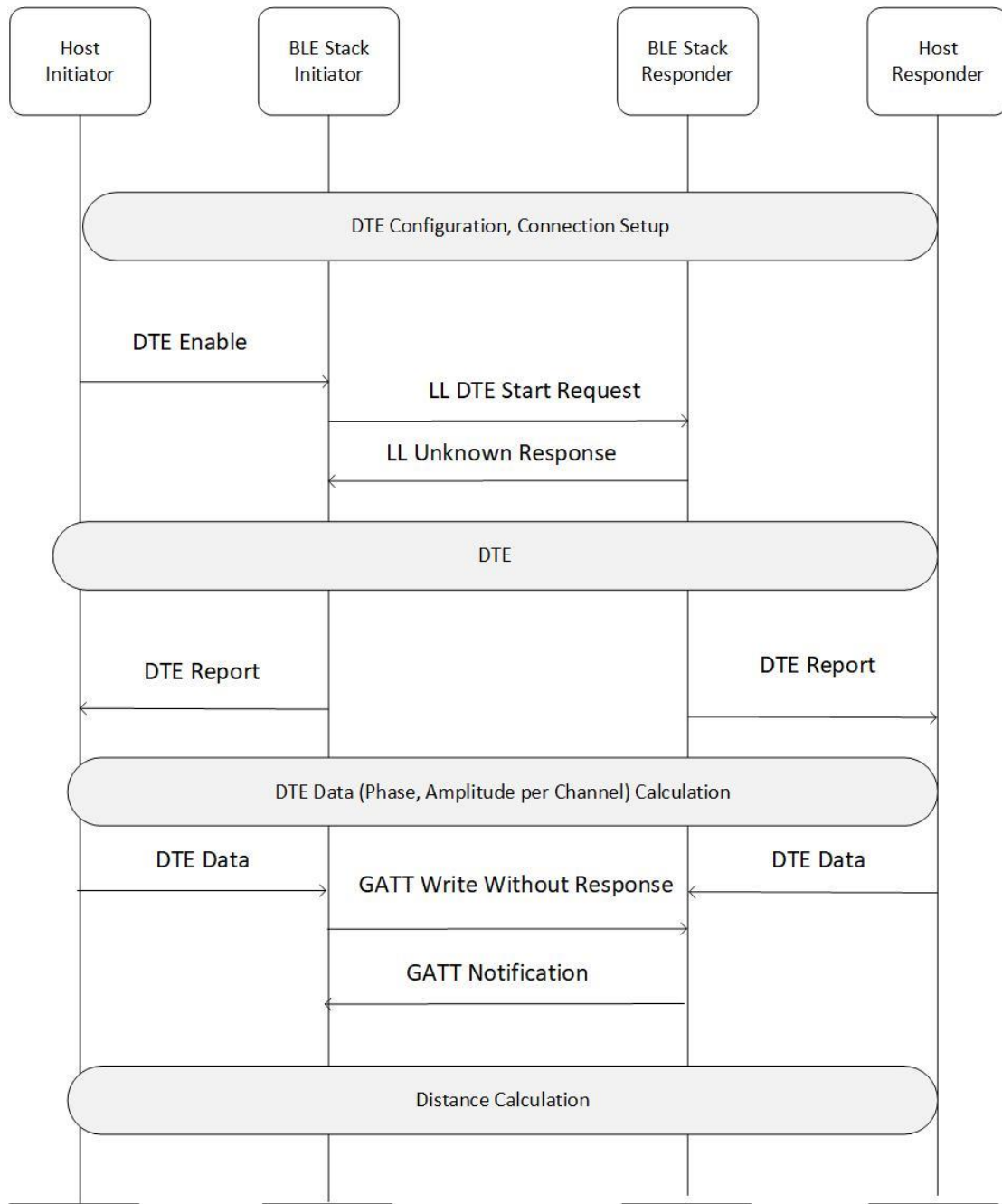


Figure 14: Application Data Flow

## SmartBond™ Wireless Ranging SDK

### 5.1.2 Software Architecture

Figure 15 presents the different software blocks of the Wireless Ranging application. Each rectangle indicates a module. The dashed blocks represent optional features that can be enabled at compile time.

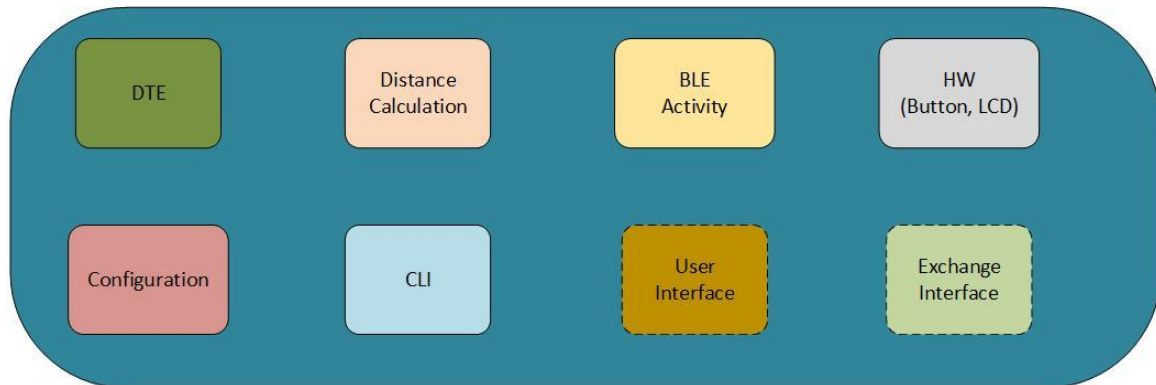


Figure 15: Wireless Ranging Application Software Modules

## 5.2 Dialog Tone Exchange

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

The DTE 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 IQ data buffer. At the end of the DTE, the data buffer will contain the IQ data slices for all the different frequencies. Each Tone Exchange step in a single frequency is called an atom.

The RFMON buffer is statically allocated to a specific size and is used with RFMON configured in cyclic mode. The number of IQ data buffers and the length is predefined at compile time. Each time before DTE starts, the first available IQ data buffer is selected.

There are two main activities related to the DTE:

- DTE configuration (section 5.2.1)
- DTE enable (section 5.2.2)

The configuration runs once when the application is initialized.

**SmartBond™ Wireless Ranging SDK**
**5.2.1 DTE Configuration**

Configuration of DTE parameters is done when the application initializes at function `ble_range_dte_task()`. The structure with the configuration parameters is called `gap_dte_params_t` and is defined in `ble_gap.h`.

```
typedef struct {
    bool    is_initiator;           /* acquisition role */
    uint8_t nb_atoms;              /* number of atoms to measure */
    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 */
    uint8_t  agc_freeze_lvl;       /* value to use during the AGC freeze (0 ...
9) */
    bool    agc_freeze_auto;       /* use automatic AGC freeze */
    uint8_t  tx_power;             /* tx power index to use */
} gap_dte_params_t;
```

The code listed above has the following information:

- Parameter `is_initiator` refers to Tx/Rx sequence. Two roles are defined:
  - Initiator
  - Responder
- Parameters `f_start_mhz`, `f_step_mhz`, and `nb_atoms` 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 `meas_length_us` parameter configures the length of the data section to be copied from each atom in  $\mu\text{s}$  with a default value of 16  $\mu\text{s}$  (128 samples)
- Parameters `agc_freeze_lvl` and `agc_freeze_auto` control the AGC gain. If `agc_freeze_auto` is set to true, the AGC value to be used is estimated via a Tx/Rx sequence that takes place before DTE. Otherwise `agc_freeze_lvl` represents the fixed AGC value that will be used for DTE
- The `tx_power` parameter controls the TX power during DTE. The default value is the index for max TX power (+6 dBm)

The related command to configure DTE is:

```
ble_error_t ble_gap_dte_params_set(gap_dte_params_t *dte_params);
```

**5.2.2 DTE Enable**

The DTE is initiated from the initiator with the following command:

```
ble_error_t ble_gap_dte_enable(uint16_t conn_idx, bool enable);
```

This function must be called from the Initiator application at the beginning of each measurement cycle. Each call triggers one DTE, after that DTE is disabled again so there is no need to explicitly call this function with the flag set to false.

When the controller of the Initiator receives this command, it sends a Link Layer control PDU to the Responder controller that contains the BLE instance (connection event) after which DTE will take place.

## SmartBond™ Wireless Ranging SDK

### 5.3 Bluetooth LE Activity

A BLE connection is used for both the DTE synchronization and the exchange of the intermediate DTE Data. 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 (see 5.1).

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

#### 5.3.1 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 DTE Data service's UUID and a matching set number in the received advertising reports. 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 the 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 in which an attempt is made to connect to a remote device with a specific Bluetooth® address that is a parameter of the connect command.

#### 5.3.2 Peripheral Role

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

1. Advertising State:  
The device starts advertising with the DTE Data service's UUID and the device set number in the advertising data. Then the peripheral waits for connection requests from the central device.
2. Connected State:  
After the two devices connect, the peripheral replies to the discovery and MTU requests and waits for a START request to start the measurement sequence.

#### 5.3.3 Bluetooth LE DTE Data Service

A custom BLE service is implemented for the exchange of intermediate DTE results. The service attribute database contains the following attributes:

- 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

#### 5.3.4 Measurement Cycle and Application State

After both devices are connected, they follow a similar measurement cycle until one device gets disconnected. The related application data flow of the measurement cycle is described in 5.1.1. During the measurement cycle there is a sequence of state changes. The initial state is STOP.

**SmartBond™ Wireless Ranging SDK**

1. **START:**
  - The central device enters the START state, starts the measurement timer and enables DTE.
  - The peripheral device enters the START state when a DTE report is received and starts the measurement timer.
2. **CALC:**
  - Both devices switch to CALC state after the DTE report is received. They calculate phases and amplitudes from the IQ data of the report, send the results to each other (the central uses a BLE GATT write without a response command and the peripheral uses a BLE GATT notification), wait for remote results, calculate the distance, and switch to the STOP state. Both devices also switch to the STOP state if a timeout occurs.
3. **STOP:**
  - The Central and Peripheral devices print log information. The Central device switches to the START state.

**5.3.5 Other Bluetooth LE Activity**

Multiple - but not in parallel - DTE ranging measurements over an equal number of BLE connections are also supported. This is made possible by the non-volatile configuration parameter `multilink_num` and its value (see 5.9.2).

By design, in the default application mode, the central device goes through steps 1-4 listed in 5.3.1 repeatedly until the number of connections value set for `multilink_num` is reached. Unless modified by the user (see 4.3.2), the default value of this parameter is 1, thus enabling only a single DTE capable BLE connection to be established. Once connections reach the value set for `multilink_num`, the DTE range measurement sequence described in 5.3.4 is triggered for each one of them in a Round-Robin fashion. If any of the connections is dropped, the central device restarts scanning, in an effort to restore the connections number to `multilink_num` again. So, several DTE range measurements and scan jobs may be active at the same time.

Things are slightly different when the central device operates in CLI mode. In this case the value set for `multilink_num` is ignored and the user may request for each DTE capable connection to take part or not to the Round-Robin scheduling via the cli commands `start` and `stop` respectively, which are described in detail in 5.7.1.

**5.3.6 Errors / Statistics**

As described in 5.3.4, after entering START state, both devices will activate a measurement timer to track local state and detect any errors. The measurement timeout is currently configured to **500 ms** so that it covers the duration of the entire measurement sequence, corresponding to the application's default connection interval of **30 ms**, plus an additional timeout to account for possible latencies, packet retransmissions etc. Depending on the local measurement state, errors are identified and recorded with respect to the currently DTE active connection index.

The existing DTE measurement error codes are defined in file `dte_common.h`.

```
typedef enum {
    DTE_NO_ERROR,
    DTE_ERROR_INSTANT_PASSED,
    DTE_ERROR_SYNC_FAILED,
    DTE_ERROR_RESULTS_TO,
    DTE_ERROR_DISCONNECTED,
    DTE_ERROR_LAST,
} dte_error_t;
```

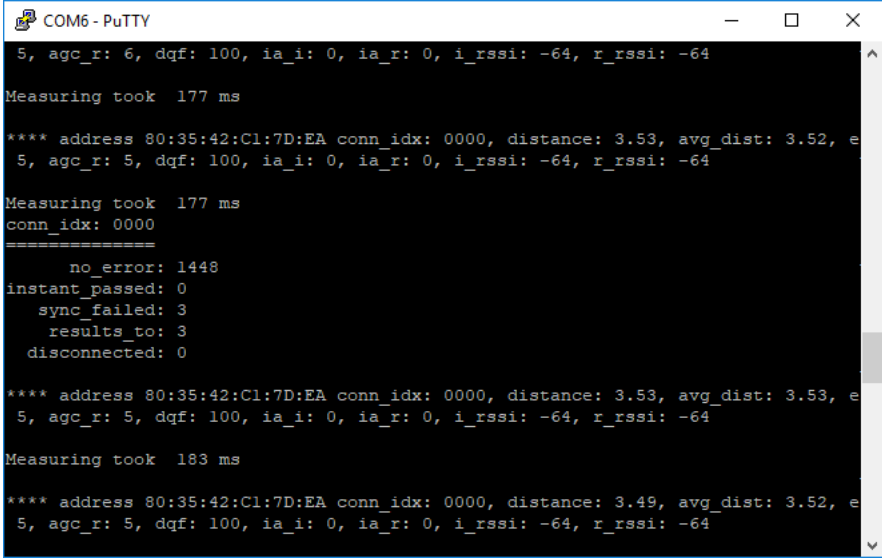
The information in the enumeration above has the following meaning:

- **DTE\_NO\_ERROR:** The measurement sequence has finished without any error
- **DTE\_ERROR\_INSTANT\_PASSED:** Failed to start measurement because the requested connection event counter has passed

## SmartBond™ Wireless Ranging SDK

- DTE\_ERROR\_SYNC\_FAILED: Failed to synchronize with partner device
- DTE\_ERROR\_RESULTS\_TO: Failed to receive either the report with the acquisition data from the RF Unit, or the intermediate results from the remote device
- DTE\_ERROR\_DISCONNECTED: Failed because the partner device has been disconnected

Every device maintains a table with measurement statistics for each DTE capable connection. When the K1 button is pressed during normal operation, the user may request for the statistics table to be printed to the console output as shown in [Figure 16](#).



```

COM6 - PuTTY
5, agc_r: 6, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -64, r_rssi: -64
Measuring took 177 ms
**** address 80:35:42:C1:7D:EA conn_idx: 0000, distance: 3.53, avg_dist: 3.52, e
5, agc_r: 5, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -64, r_rssi: -64
Measuring took 177 ms
conn_idx: 0000
=====
no_error: 1448
instant_passed: 0
sync_failed: 3
results_to: 3
disconnected: 0
**** address 80:35:42:C1:7D:EA conn_idx: 0000, distance: 3.53, avg_dist: 3.53, e
5, agc_r: 5, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -64, r_rssi: -64
Measuring took 183 ms
**** address 80:35:42:C1:7D:EA conn_idx: 0000, distance: 3.49, avg_dist: 3.52, e
5, agc_r: 5, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -64, r_rssi: -64

```

Figure 16 Measurement Statistics Report

### 5.4 Distance Calculations (Phase/IFFT)

Once acquisition finishes, distance results are extracted from the sampled measurement data. All operations specific to distance estimation are implemented inside directory “calc”. [Figure 17](#) shows an overview of the related process.

SmartBond™ Wireless Ranging SDK

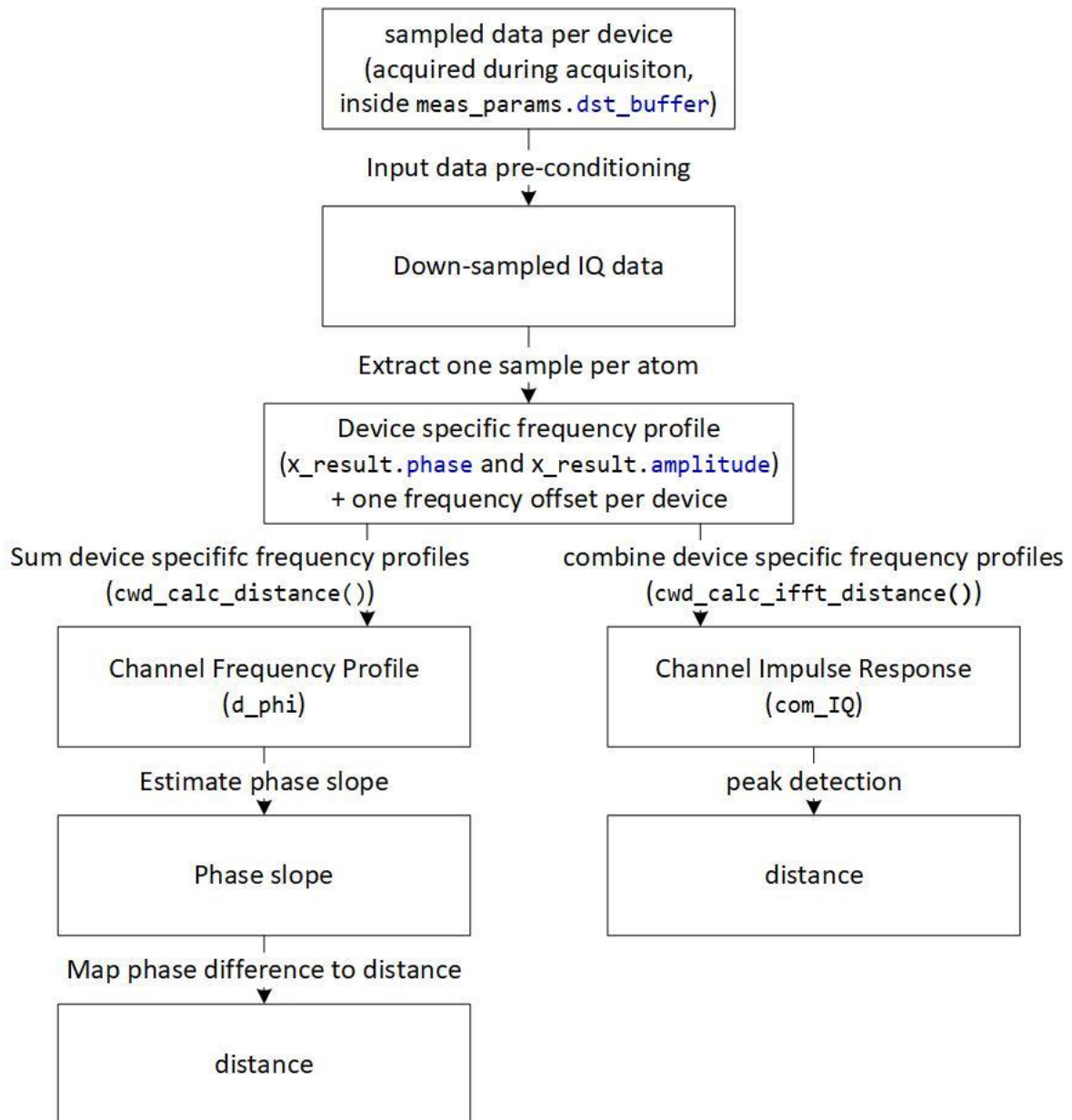


Figure 17: Distance Calculations

There are two basic ways of distance estimation:

- IFFT-based distance calculation (right-hand side in Figure 17, and section 5.4.1)
- Phase-based distance calculation (left-hand side in Figure 17, and section 5.4.2)

## SmartBond™ Wireless Ranging SDK

The following algorithm steps are common to both ways:

1. Input data pre-conditioning:
  - a. Extract IQ data from acquired test bus data samples (sampling rate of 8 MS/s).
  - b. Sample down by a 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  $f_{if}$ ,  $\pm 1$  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.

### 5.4.1 IFFT Based Distance Estimation (Default)

If **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 the 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 the 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. Remove all peaks below a configurable threshold to filter the peak list.
  - c. Calculate the corresponding distance to each peak above the threshold.
  - d. Find the peak with the shortest distance and take that as the estimated result.



## SmartBond™ Wireless Ranging SDK

### 5.4.2 Phase Based Distance Estimation

If **ENABLE\_IFFT\_DIST\_CALC** is not defined, the distance is determined as follows:

1. Retrieve the 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 that as the estimated result.

The underlying relation between distance  $D$ , phase difference  $\Delta\varphi$ , frequency difference  $\Delta f$ , 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
$\Delta f$	Frequency difference in Hz
$N$	Number of atoms

## SmartBond™ Wireless Ranging SDK

### 5.5 Hardware Support

The following hardware features are used to support the Wireless Ranging application in a user-friendly way:

- **Buttons:** the function `hw_k1_button_pressed` checks if the K1 button is pressed. It is used during reset to decide whether to enter the configuration mode. If configuration mode is not entered, a function is registered on the K1 button push to report DTE ranging statistics for every existing connection.
- **LCD:** the high-level interface function `lcd_init` initializes the display, and the function `lcd_draw_string` draws a string with the distance measurement value in a display.

### 5.6 User Interface

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

### 5.7 CLI

The CLI is a special mode only for the Initiator, where after reset the device waits for the user to issue the appropriate commands to do certain operations like connect, range measurement, etc. See section 4.3.3 for how to enable this mode.

The CLI functionality is implemented in file `cli.c`. The function `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 the CLI. Function `cli_get_clistr` returns the current input string that has been read from the CLI. Function `readline` reads a line from the CLI and `parseline` parses the line to get a command.

#### 5.7.1 CLI Commands

The following is a list of available cli commands, their format and functional description.

- `connect <address> <min> <max>`: This command takes one mandatory and two optional input arguments. The first argument should be the address of the remote device to try to connect to and should be in the form `<xx:xx:xx:xx:xx:xx>`. Additionally, up to two input arguments may be used to specify the minimum and maximum values for the connection interval. If any of the optional arguments is not given, the default value of 30ms is used. *NOTE: Any user-scheduled scan needs to be stopped before attempting to make a connection. See command `scan_stop` for more details.*
- `disconnect <conn_idx>`: If a valid connection index is provided as an input argument, the Initiator will try to terminate the corresponding connection.
- `start <conn_idx>`: With this command the Initiator is instructed to trigger DTE range measurement for the specified connection index.
- `stop <conn_idx>`: The device will stop triggering any further DTE range measurements for this connection index.
- `status`: Helper command to display information about the available connections. Upon execution, entries of the format `<conn_idx> <address> <DTE status>` will be printed on the console output. DTE status indicates whether range measurement for this connection is unsupported, disabled or enabled. If DTE ranging is supported, then the start and stop commands described earlier can be used to enable or disable it. Besides the list, command output also provides information regarding the number of connections, active scan jobs and ongoing advertising.
- `scan_start <interval> <>window>`: The Initiator will try to schedule a scan job of `<window>` ms every `<interval>` ms. The scan interval needs to be greater than the scan window otherwise the command will fail. The `status` command described above can be used to check whether the user has a scan job already scheduled in the background or not. Currently the command provides no console output and its usage is intended for testing purposes only.

## SmartBond™ Wireless Ranging SDK

- `scan_stop`: The device will try to remove any scan job the user has scheduled.

### 5.8 External Interface

The external interface module can be used to support communication with an external host through RAM using the SEGGER JTAG. To enable the external interface functionality, set the define `EXCHANGE_MODE` to 1 in file `ble_range_dte.h`.

This mode supports Python interaction through SEGGER JTAG using PyMon. PyMon is a Dialog python package that enables the control of a Dialog Semiconductor Bluetooth® SoC through a SEGGER J-Link debugger probe.

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

- Bidirectional flags used for handshake between the DTE application and Python scripts
- Configuration parameters set from Python
- Data parameters sent to Python

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

There are four main functions for the exchange mode support:

- `exchange_mem_init` for handshake parameters initialization from the application side
- `exchange_mem_params_exchange` for setting the `is_initiator`, `f_start_mhz`, `f_step_mhz`, `nb_atoms` DTE configuration parameters from Python
- `exchange_mem_iq_data_set` and `exchange_mem_phase_data_set` for sending iq data and phase / magnitude information to Python

The exchange functions implement a blocking handshake where they set a ready flag and then wait for Python code to set parameters/copy information and then clear the flag.

There is an initial handshake between the application and python in function `exchange_mem_params_exchange`.

Then at every measurement there are two handshakes, one for IQ data `exchange_mem_iq_data_set` and one for phase and magnitude data exchange `exchange_mem_phase_data_set`.

A python script `dte_iq_data_analyze.py` to test the iq data and phase exchange and the PyMon module is available in `projects\host_apps\python_iqdata_tools`.

The script can be tested with the Spyder python environment that is included in Anaconda, the Python Data Science Platform.

To show the plots in Spyder in their own interactive window, go to **Tools > Preferences > Ipython console > Graphics** tab > **Graphics backend > Backend** and set to **Automatic**.

## SmartBond™ Wireless Ranging SDK

### 5.9 Configuration Options/Nonvolatile Parameters

There are two different options to configure the DA1469x Wireless Ranging application:

- The static switches that need to be configured during software compilation (section 5.9.1)
- The dynamic parameters that are stored in non-volatile memory and can be changed dynamically at run time (section 5.9.2)

#### 5.9.1 Compile Time Configuration Switches

The configuration switches for the DA1469x Wireless Ranging application listed in Table 5-1 are defined in file `ble_range_dte_config.h`.

**Table 5-1: Configuration Switches**

Switch	Default	Effect
ENABLE_CW_DIST_CALC	enabled	Enables distance calculation in firmware
ENABLE_IFFT_DIST_CALC	enabled	If enabled, the distance is calculated by Channel Impulse Response, otherwise the distance is calculated by Channel Frequency Profile
EXCHANGE_MODE	disabled	If enabled, data exchange with external host after measurement completion is supported
USE_RANDOM_BD_ADDRESS	enabled	If enabled and the current address is equal to the SDK default, a random BD address will be generated

**Note 1** See section 5.9.2 on how to enable XTAL auto trimming and distance average functionality

## SmartBond™ Wireless Ranging SDK

### 5.9.2 Non-Volatile Parameters

Configuration of non-volatile parameters is done when the application initializes by function `ble_range_dte_task()`. The structure with the configuration parameters is called `config_params_t` and is defined in file `ble_range_dte_task.c`.

```
typedef struct {
    dte_role_t role;
    uint8_t set_number;
    int16_t cw_offset;
    uint16_t xtal32m_trim_value;
    uint8_t flag_auto_xtal32_trim:1;
    uint8_t flag_distance_averaging:1;
    uint8_t cli_mode:1;
    uint8_t multilink_num;
} config_params_t;
```

- Parameter `role` is related to the acquisition Tx/Rx sequence. Two roles are defined, the Initiator and the Responder.
- `set_number` is a number less or equal to 254. It is used at 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 reported. 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, the application auto-tunes the XTAL32M\_TRIM register in order to minimize the frequency offset at the end of each measurement. To achieve this and have the Initiator's frequency as a reference, both devices perform trimming towards opposite directions. Coarse tuning is stopped earlier for the Responder node so that oscillation around the optimum value is avoided. Fine tuning is finally performed on the Initiator node.
- Setting `flag_distance_averaging`, enables a  $2^N$  sliding window filter to apply on the raw values for average distance reporting. The sample window size is controlled by the compile time switch `DIST_MA_WINDOW`.
- If `cli_mode` is set, the Initiator will operate in manual mode leveraging a cli for assisted user setup. This setting does not apply to Responder devices and is therefore ignored.
- In the default application mode, the Initiator device continues to scan for more connections after a connection is established, until `multilink_num` is reached. At this time the DTE range measurement is triggered for every connection in a Round-Robin fashion as described in [5.3.5](#). The value of `multilink_num` is ignored if the device is (a) just a Responder or (b) an Initiator that operates in CLI mode.
- If the K1 button is pressed during device reset, the application enters into configuration mode where all these parameters can be changed from the CLI.

## 6 Bluetooth LE IQ Data Applications

Besides DTE, IQ data can also be received from BLE packet receptions. In folder `projects\dk_apps\iq_data` there are two applications `ble_iq_scanner` and `ble_iq_advertiser` that can be used as examples of how to use BLE IQ data in scanning, advertising or connected state.

### 6.1 Application Overview

By default, the applications support simple scan and advertising scenarios. There is also a special CLI mode that allows to test more advanced scenarios. The supported test cases are described in function `rfmon_test_config`. Each test case has a unique identification number.

In function `rfmon_test_run` the configuration of RFMON takes place and the related BLE activity starts.

### 6.2 RFMON Interface

The following commands are supported to initialize and control the RFMON functionality:

#### 6.2.1 RFMON Configuration

- `void rfmon_init(bool send_gapm_evt, rfmon_report_cb cb);`

Configures how the IQ data reports will be send to the application. If parameter `send_gapm_evt` `send` is set to true, the report is sent as a GAP event, else the GAP layer is bypassed and a direct system call through `sys_rfmon_report_process` is used. In the latter case, a special 'rfmon' task is responsible to receive the report and call the registered report callback that handles the report. The IQ data reports are described in section 7.

- `ble_error_t rfmon_params_set(uint8_t output_mode, uint8_t trigger, uint8_t report_errors);`

This command configures the controller hardware that is responsible to capture IQ samples from the received packets. It can be issued only when sampling is not enabled in the controller.

`RFMON_OUTPUT_MODE_IQ9` is the only available setting for `output_mode` with the following format: `adcout_i & adcout_q & agc_setting & vga3_out_i & vga3_out_q` (shown in Table 6-1).

- Parameter `trigger` can have the following values:
  - `RFMON_TRIGGER_DEM_EN`: The capturing starts with the rising edge of `DEM_EN` and stops with the falling edge of this signal
  - `RFMON_TRIGGER_SIGNAL_DETECT`: The capturing starts when a signal is detected and stops with the falling edge of `DEM_EN`
  - `RFMON_TRIGGER_UNTIL_SYNC`: The capturing starts with the rising edge of `DEM_EN` and stops with the rising edge of `SYNC_FOUND_P`
  - `RFMON_TRIGGER_FROM_SYNC`: The capturing starts with the rising edge of `SYNC_FOUND_P` and stops with the falling edge of `DEM_EN`
- Parameter `report_errors` is set to value `RFMON_REPORT_ERRORS` if samples from packets are reported with CRC errors, else the value `RFMON_REPORT_NO_ERRORS` is used

**Table 6-1: RFMON\_OUTPUT\_MODE\_IQ9 Format**

Signal	Bit width	Format	Description
<code>adcout_i</code>	9	signed	RF ADC I output
<code>adcout_q</code>	9	signed	RF ADC Q output
<code>agc_setting</code>	4	unsigned	RX gain. Value 0 means max gain. 6dB gain steps

vga3_out_i	5	signed	RF ADC I output after digital HPF and normalization
vga3_out_q	5	signed	RF ADC I output after digital HPF and normalization

### 6.2.2 RFMON Enable

- `ble_error_t rfmon_enable(uint8_t enable, uint8_t scan_filter_enable, uint8_t adv_filter_enable, uint8_t conn_filter_enable);`

Enables RFMON sampling and the relevant filtering.

- If the `enable` parameter is set to `RFMON_SAMPLING_ENABLE`, then RFMON is enabled. If set to `RFMON_SAMPLING_DISABLE`, then it will be disabled. And if set to `RFMON_SAMPLING_UNCHANGED`, then it will keep the current state
- The following three parameters, that is the `scan_filter_enable`, the `adv_filter_enable` and the `conn_filter_enable` will enable, disable or leave the relevant filters unchanged

If this command results in enabling sampling, an IQ data report will be generated whenever the sampling of a packet is completed, and the received packet passes through the selected filtering criteria.

### 6.2.3 RFMON Filtering

- `ble_error_t rfmon_filter_manage_device(uint8_t type, uint8_t enable, uint8_t address_or_handle[BD_ADDR_LEN]);`

This command is used to request the controller to add or remove entries from the RFMON IQ filters. There are three filters indicated, one for the scanner (`ADV_IND` and `SCAN_RSP`), one for the advertiser (`SCAN_REQ`) and one for the connections filtering specific connection handles.

- The `Type` parameter specifies the filter on which the device entry will be added or removed. If it has the value of `RFMON_TYPE_SCAN`, it concerns the scanning filter. If it has the value of `RFMON_TYPE_ADV`, it concerns the advertising filter and finally, if it has the value of `RFMON_TYPE_CONN`, it refers to the connection filter
- If the `enable` parameter is set to `RFMON_DEVICE_ENABLE`, the controller will add the specific entry in the corresponding filter as defined by the `Type` parameter. If the parameter is set to `RFMON_DEVICE_DISABLE`, the Controller will remove the specific entry from the corresponding filter
- The `address_or_handle` parameter is used as an identifier for the device. For advertise or scanning, the `address_or_handle` parameter specifies the BD address of the device. For connections the `address_or_handle` parameter specifies the connection handle of the connection that is selected
- `ble_error_t rfmon_filter_clear(uint8_t type);`  
This command will request the controller to clear all entries of a specific RFMON Filter.
  - The RFMON filter type is indicated by the `type` parameter and can have the following values: `RFMON_TYPE_SCAN`, `RFMON_TYPE_ADV`, `RFMON_TYPE_CONN`

## 6.3 External Interface

The external interface module is used to support communication with an external host through RAM using the SEGGER JTAG. To enable the external interface functionality, set the define statement `EXCHANGE_MODE` to 1 in the application configuration file, e.g. `ble_iq_advertiser_config.h`.

The external interface is similar to that described for the `ble_range_dte` application in 5.8.

The main differences are:

- Struct `exchange_params_t` in header file `exchange_mem.h` describes a different set of parameters for data exchange between the application and Python

## SmartBond™ Wireless Ranging SDK

- There are no parameters set from Python
- There is only one data exchange function `exchange_mem_data_set` for IQ data exchange every time an IQ data report is received in the application
- A python script `iq_data_analyzer.py` to test the IQ data exchange and the PyMon module can be found in `projects\host_apps\python_iqdata_tools`

### 6.4 CLI Mode

The CLI mode supports 2 commands: 'r' to start running a test case and 's' to stop it.

Also, if a valid test case identification number is entered as input, the application changes the current RFMON configuration to the configuration for that test case.

## 7 IQ Data Reports

The RFMON Report event is used by the controller to report that the IQ information of a received BLE packet or a DTE has been stored in the respective buffer. The event contains two parameters: the report status and the sampling buffer address.

This event is also used by the controller to report that it has insufficient resources to store the IQ samples and had failed to sample at least once. In this case, the status will be set to 0x01 and the sampling buffer address will be set to 0.

The buffer space must be statically allocated during application image creation. The defines `RFMON_NUM_OF_BUFFERS`, `RFMON_PACKET_BUFFER_SIZE` and `RFMON_IQ_DATA_BUFFER_SIZE` in file `custom_config.qspi.h` control the number of sampling buffers, the max size of the BLE packet data and the size of the IQ data that can be received in each sampling buffer.

Each buffer consists of three parts: a control part ("header"), a part where the data of the received packet is stored (enough to hold the maximum possible packet), and a part where the IQ samples of the packet are stored. This last buffer operates in circular mode. Each sample is 32-bits. BLE reception is a 16 MSPS operation, and gets 16 samples per bit for LE 1M PHY. So, for an advertising packet (47 bytes), there is 24064 bytes needed for the IQ data, while for a data packet (41 bytes, no Data Length Extension) 20992 bytes is needed. DTE is an 8 MSPS operation so we get 8 samples per us. Since the IQ samples buffer operates in circular mode, the reception of a large BLE packet will not crash the system, although the samples in the buffer will be overridden. No memory overflow will occur though.

The structure with the sampling buffer parameters is called `sampling_buffer_header_t` and is defined in `user_config_defs.h`.

```
typedef struct sampling_buffer_header_ {
    bool buffer_status;
    uint8_t sampling_rate;
    uint8_t agc_value;
    bool packet_status;
    uint8_t access_address[4];
    uint8_t channel_idx;
    uint8_t crc_seed[3];
    uint8_t channel_map[5];
    uint8_t hop_interval;
    uint16_t rssi;
    uint32_t sampling_start;
    uint32_t sampling_end;
}
```



SmartBond™ Wireless Ranging SDK

```
uint8_t oflow;
uint8_t reserved[3];
} sampling_buffer_header_t;
```

The parameters in the sampling buffer header are analyzed below:

- `buffer_status` if false the buffer is free, else the buffer is used and contains data to be processed by the application
- `sampling_rate` currently refers to the physical BLE symbol rate, 0 for 1 Mb, 1 for a 2 MB rate
- `agc_value` for BLE packet or DTE
- `packet_status` for BLE packets set to 0 if CRC was correct, otherwise 1
- `access_address` for the access address of a BLE packet. Set to 0 for DTE, can be used to separate BLE and DTE RFMON reports since an all zero-access address is not a valid BLE access address
- `channel_idx` BLE packet channel index, for DTE channel index of last BLE packet reception before DTE
- `crc_seed` initialization value for the CRC calculation of BLE packets
- `channel_map` the channel map used (only for BLE data packets)
- `hop_interval` the `hopIncrement` value used in the data channel selection algorithm. It has a random value in the range from 5 to 16. This parameter is meaningful only for BLE data packets
- `rssi` value of BLE packet, for DTE RSSI value of last BLE packet reception before DTE
- `sampling_start` location of IQ data buffer start. If IQ data buffer is in shared memory it will be after the end of the section for the BLE packet data
- `sampling_end` location of last sample in IQ data buffer
- `oflow` is the RFMON overflow status after BLE packet reception or DTE
- `reserved[3]` added for word alignment of sampling buffer structure

By default, the sampling buffer area is in shared memory right after the end of the sampling buffer area header `cmac_sampling_table` in CMAC, accessed through the `cmac_sampling_table_ptr` in the host.

The structure of a sampling buffer is shown in [Table 7-1](#).

**Table 7-1: Sampling Buffer Structure**

Header
BLE Packet Data
BLE / DTE IQ Data

The report can be received in the application as a GAP event (`BLE_EVT_GAP_RFMON_REPORT`) or as a direct system call that bypasses the GAP layer. In the latter case a special 'rfmon' task is responsible to receive the report and call a registered callback that handles the report. DTE Reports use the GAP event method while the BLE IQ data application uses the system call method. This can be defined during runtime at the RFMON initialization, see [6.2](#).

For BLE IQ reports only, there is an option to use Host memory for IQ data samples. This option is disabled by default but can be enabled if define statement `RFMON_HOST_IQ_DATA_BUFFER` is added in file `custom_config_qspi.h` with a value of 1. This is possible because the IQ data buffer is used as the RFMON buffer and RFMON can do direct DMA transfers without controller involvement. It will not work for DTE reports since in that case there is an intermediate step in the controller that copies IQ data samples from the RFMON buffer to the IQ data buffer, so the IQ data buffer needs to be in shared memory.

## Revision History

Revision	Date	Description
1.0	07-May-2020	Initial version.

## SmartBond™ Wireless Ranging SDK

### 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

Unless otherwise agreed in writing, the Dialog Semiconductor products (and any associated software) referred to in this document are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of a Dialog Semiconductor product (or associated software) can reasonably be expected to result in personal injury, death or severe property or environmental damage. Dialog Semiconductor and its suppliers accept no liability for inclusion and/or use of Dialog Semiconductor products (and any associated software) in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Information in this document is believed to be accurate and reliable. However, Dialog Semiconductor does not give any representations or warranties, express or implied, as to the accuracy or completeness of such information. Dialog Semiconductor furthermore takes no responsibility whatsoever for the content in this document if provided by any information source outside of Dialog Semiconductor.

Dialog Semiconductor reserves the right to change without notice the information published in this document, including, without limitation, the specification and the design of the related semiconductor products, software and applications. Notwithstanding the foregoing, for any automotive grade version of the device, Dialog Semiconductor reserves the right to change the information published in this document, including, without limitation, the specification and the design of the related semiconductor products, software and applications, in accordance with its standard automotive change notification process.

Applications, software, and semiconductor products described in this document are for illustrative purposes only. Dialog Semiconductor makes no representation or warranty that such applications, software and semiconductor products will be suitable for the specified use without further testing or modification. Unless otherwise agreed in writing, such testing or modification is the sole responsibility of the customer and Dialog Semiconductor excludes all liability in this respect.

Nothing in this document may be construed as a license for customer to use the Dialog Semiconductor products, software and applications referred to in this document. Such license must be separately sought by customer with Dialog Semiconductor.

All use of Dialog Semiconductor products, software and applications referred to in this document is subject to Dialog Semiconductor's [Standard Terms and Conditions of Sale](http://www.dialog-semiconductor.com), available on the company website ([www.dialog-semiconductor.com](http://www.dialog-semiconductor.com)) unless otherwise stated.

Dialog, Dialog Semiconductor and the Dialog logo are trademarks of Dialog Semiconductor Plc or its subsidiaries. All other product or service names and marks are the property of their respective owners.

© 2020 Dialog Semiconductor. All rights reserved.

### RoHS Compliance

Dialog Semiconductor's suppliers certify that its products are in compliance with the requirements of Directive 2011/65/EU of the European Parliament on the restriction of the use of certain hazardous substances in electrical and electronic equipment. RoHS certificates from our suppliers are available on request.

## 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](http://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

## X-ON Electronics

Largest Supplier of Electrical and Electronic Components

*Click to view similar products for* [Bluetooth Development Tools - 802.15.1 category:](#)

*Click to view products by* [Dialog Semiconductor manufacturer:](#)

Other Similar products are found below :

[DA14580PRODTLKT](#) [1628](#) [MBH7BLZ02-EF-KIT](#) [CYBLE-014008-PROG](#) [FWM7BLZ20-EB-KIT](#) [ATSAMB11ZR-XPRO](#) [SKY66111-21EK1](#) [SECO-RSL10-TAG-GEVB](#) [3026](#) [MIKROE-2471](#) [MOD-NRF8001](#) [BLE-IOT-GEVB](#) [450-0184](#) [MIKROE-2399](#) [EKSHCNZXZ](#) [EVAL\\_PAN1026](#) [EVAL\\_PAN1720](#) [EVAL\\_PAN1740](#) [2267](#) [2479](#) [2487](#) [2633](#) [STEVAL-IDB005V1D](#) [STEVAL-IDB001V1](#) [MIKROE-2545](#) [SIPKITSLF001](#) [2995](#) [STEVAL-IDB007V1M](#) [2829](#) [DFR0267](#) [DFR0296](#) [DFR0492](#) [TEL0073](#) [BM-70-CDB](#) [WSM-BL241-ADA-008DK](#) [STEVAL-BTDP1](#) [ACD52832](#) [TEL0095](#) [ISP1507-AX-TB](#) [RN-4871-PICTAIL](#) [DA14695-00HQDEVKT-P](#) [DA14695-00HQDEVKT-U](#) [EVK-NINA-B112](#) [EBSHJNZXZ](#) [EKSHJNZXZ](#) [BMD-200-EVAL-S](#) [ACN BREAKOUT BOARD](#) [ACN SKETCH](#) [2269](#) [2746](#)