UG471: Flex SDK v3.x Range Test Demo User's Guide

This version of UG471 has been deprecated with the release of Simplicity SDK Suite 2025.6.1.

For the latest version, see docs.silabs.com.

****

This user's guide describes an easy way to evaluate the link budget of the Wireless Gecko EFR32 devices using Silicon Labs' Radio Abstraction Interface Layer (RAIL) by performing a range test between a transmitter and any number of receiver nodes. Range Test is a standalone test application that creates a radio link between the evaluation kits and sends a predefined number of packets from the transmitter side to the receiver. The Range Test demo implements packet error rate (PER) measurement. PER is a commonly-used technique for measuring the quality of RF links in wireless systems under particular conditions. Flex SDK 3.x contains multiple Range Test applications that enable the physical layer's (PHY) evaluation for a custom protocol or for standard protocols supported by RAIL (IEEE802.15.4g and Bluetooth LE).

Proprietary is supported on all EFR32FG devices. For others, check the device's data sheet under Ordering Information > Protocol Stack to see if Proprietary is supported. In Proprietary SDK version 2.7.n, Connect is not supported on EFR32xG22.

KEY POINTS

- Evaluate the link budget of Wireless Gecko EFR32 devices.
- Range Test is a standalone test application that creates a radio link between two evaluation kits.
- PER is a commonly-used technique for measuring the quality of RF links.
- EFR Connect smartphone application is now available for enhancing User Experience with the Dynamic Multiprotocol Range Test Applications.

1. Introduction

1.1 About the Range Test

The range test demo provides measurement results regarding the quality of the RF link. The demo uses at least two RF nodes. One node is used as the “transmitter” (TX) and the other(s) as the “receiver(s)” (RX). The transmitter sends packets to the receiver(s) repeatedly. The packet includes the self and remote IDs and the number of the sent packet. The packet number increments from packet to packet. The receiver receives the packet and checks the IDs. If they match, the packet number is stored and the received packet's RSSI displayed. Packet loss is therefore recognized by a packet number increase of greater than 1.

To conduct a range test, you will perform the following steps, described in more detail in later sections.

1. Connect the radio boards to the mainboards.
2. Connect the mainboards to the PC, select the **Flex (RAIL) - Range Test** example and configure your custom settings using Radio Configurator. The range test is performed as a one-way radio communication. Configure the devices so that one is in RX and the other is in TX mode. It is important to set the self- and remote-IDs of the participants correctly.
3. Build the project and flash the image to the device.
4. If UART communication is not used, unplug the devices from the PC, put batteries into the mainboards and switch them on.
5. Start the test on both devices.
6. Actual packet transmission can be started at the transmit side by pressing **START** again.
7. Follow the progress of the test on the LCDs.

1.2 The Range Test Menu System

The Range Test applications have a clean menu system for configuring the measurement parameters both for the transmit and the receive side. The configurable parameters and the functions of the menu differ by the applications, and are described in section 3.1.1 Configurable Parameters.

Any application created for a Silicon Labs Development Kit is configured to use the Wireless Starter Kit mainboard LCD panel and push buttons to navigate through the menu. This is the default user interface, but other options are available to configure the device.

A Command Line Interface (CLI) is also provided for each application. Therefore, the LCD and/or the push buttons are not needed to use these applications, making the examples more suitable for custom designs.

1.3 Selection Guide for the Range Test Applications

Flex SDK 3.x provides four types of Range Test application, which are classified in the following table

Table 1.1. Classification of the Range Test Applications

	Since Protocol	Dynamic Multiprotocol
Custom PHY	Range Test	Range Test DMP
Standard PHY	Range Test BLE and IEEE802.15.4	Range Test BLE and IEEE802.15.4 with DMP

1.3.1 Custom PHY vs Standard PHY Applications

Custom PHY Range Test applications are intended to evaluate the built-in or custom radio PHYs (defined in the *rail_config.c* file). Standard PHY means Bluetooth LE (BLE) or IEEE802.15.4g protocol PHYs configured with RAIL APIs. With these applications these two protocols' RF performance can be evaluated.

1.3.2 Single Protocol vs Dynamic Multiprotocol

The Single Protocol applications supports both the CLI and push button/LCD interfaces to the application's menu.

EFR32 devices with Bluetooth LE stack support (devices from the EFR32BG|MG families) programmed with the Dynamic Multiprotocol (DMP) Range Test applications can be connected to a Bluetooth-capable smartphone. Once the connection is established, the receiver/transmitter node can be controlled by the EFR Connect smartphone application, which virtualizes the default menu system on the smartphone's screen. This makes the DMP Range Test applications portable for custom designs even if those have no USART capabilities.

While the DMP applications can also be used like the single protocol applications, through the mainboard's push buttons/LCD or the CLI, the smartphone control and visualization of the results may have some advantages over the regular experience. However, advertising, collecting the measurement results, and maintaining the Bluetooth LE connection requires more power. Because of this, the device should periodically switch between the Bluetooth LE and the other (measurable) radio configuration.

Note: DMP applications require more RAM and flash memory than the single protocol applications.

Figure 1.1. EFR Connect Smartphone Application's logo in Google Play

2. Setting up Range Test Applications on the Development Kits

This chapter provides a quick summary on how to set up for range testing. Testing custom designs is outside the scope of this document. Therefore, the instructions assume the following:

- You have a complete development kit with two mainboardss and two radio boards.
- You have installed Simplicity Studio 5 and Flex SDK 3.x.
- You are familiar with using Simplicity Studio to configure, build, and flash applications. For more details, see QSG168: Proprietary Flex SDK v3.x Quick Start Guide or the online Simplicity Studio 5 User's Guide.

As noted above, running Range Test on a custom board is not extensively covered here. However, see section 2.4 A Note on Modifying the Application for Custom Boards for some guidance on this subject.

2.1 Prepare the Wireless Starter Kits

Every mainboard has a unique serial number that is displayed in Simplicity Studio's Launcher perspective once it is connected to the PC either with an ethernet or a USB cable.

The IP address and the serial number are displayed on the mainboard LCD during startup of the device, but may be lost when the app starts on the radio board. To see these again, disconnect and reconnect the mainboard's power. Once the information is displayed, press and hold the RESET button. The information remains displayed as long as the button is pressed

Note: The mainboard should be powered through its USB port, even if it is connected via Ethernet. Also, make sure that the 3-position power switch in the bottom left is set to AEM as shown in the following figure.

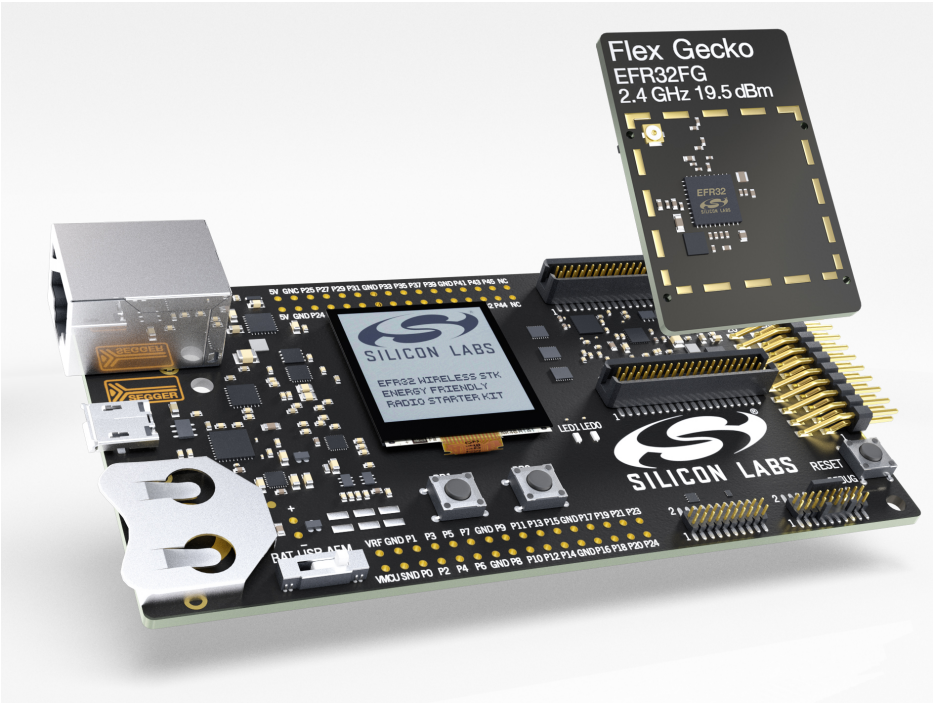


Figure 2.1. Wireless Starter Kit Mainboard with Radio Board

If you are going to use the device as a mobile device to run the range test, connecting an external AA battery pack or a USB power bank to the mainboard board is recommended. The coin cell battery will not have enough power to do long-term testing. If the development kit is powered from a battery, change the 3-position switch from AEM to VBAT.

2.2 Create a New Range Test Application

This section assumes that you have installed Simplicity Studio 5 and the Flex SDK, and have your device(s) connected to the PC.

Note: If you do not have a physical device handy, select the corresponding part in the **My Products** view to create a "virtual" device.

Select the physical or virtual device so that the **OVERVIEW** and other tabs are displayed in the **Launcher** perspective.

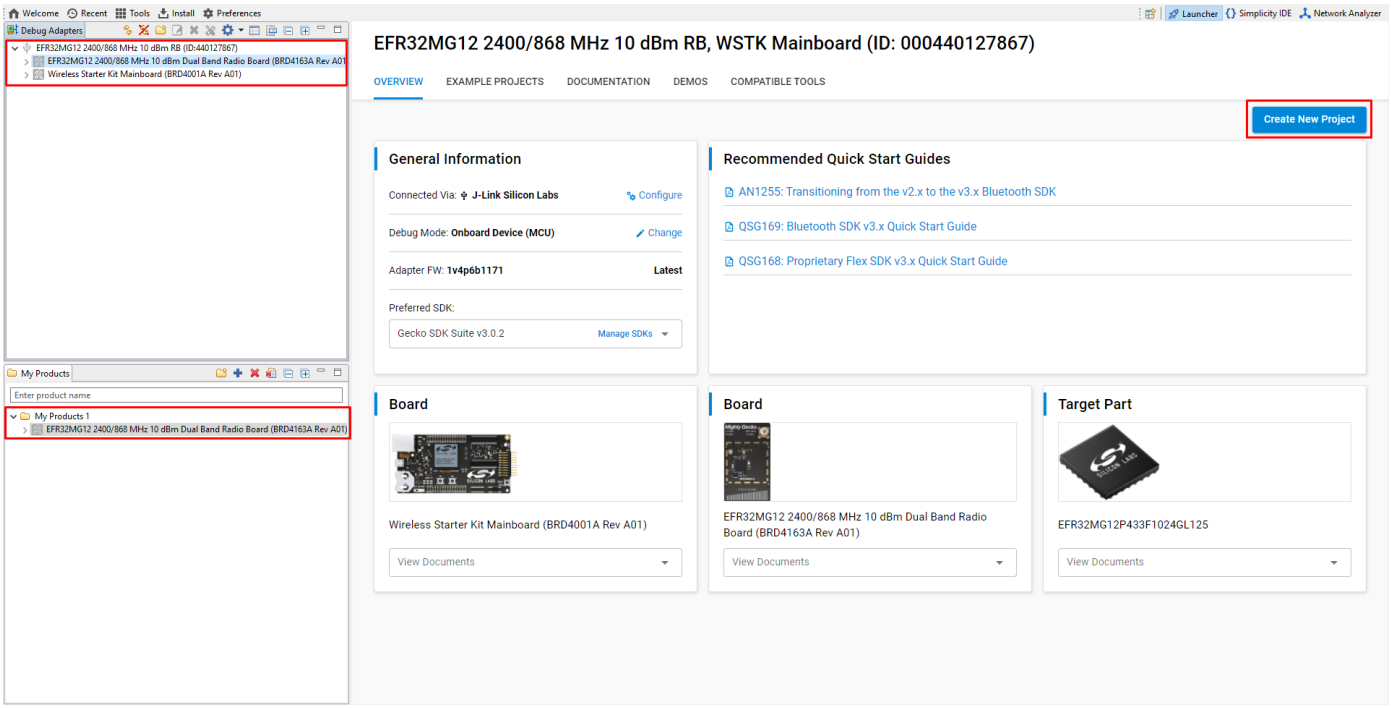


Figure 2.2. Simplicity Studio's Launcher Perspective

Alternatively, you can generate a project without a virtual or physical device by selecting **Project > New > Silicon Labs Project wizard**. Select the target board and the preferred SDK along with the toolchain, and click [**Next**].

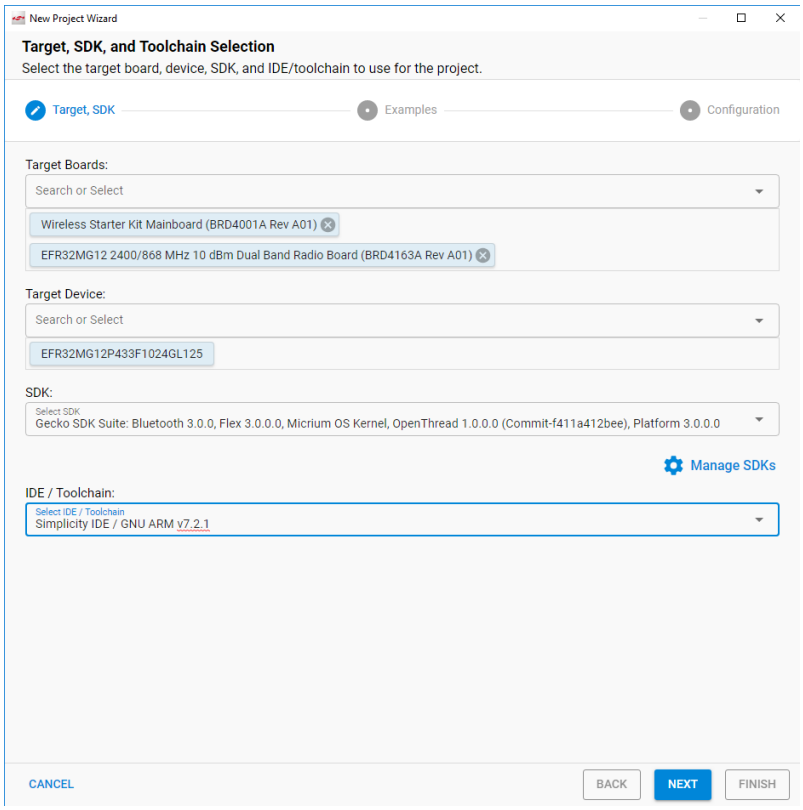


Figure 2.3. Alternative Project Creation Method

In the Example Project Selection dialog, you can browse through the supported applications for the given part. You may filter with the Range Test keyword. Select an application and click [**Next**].

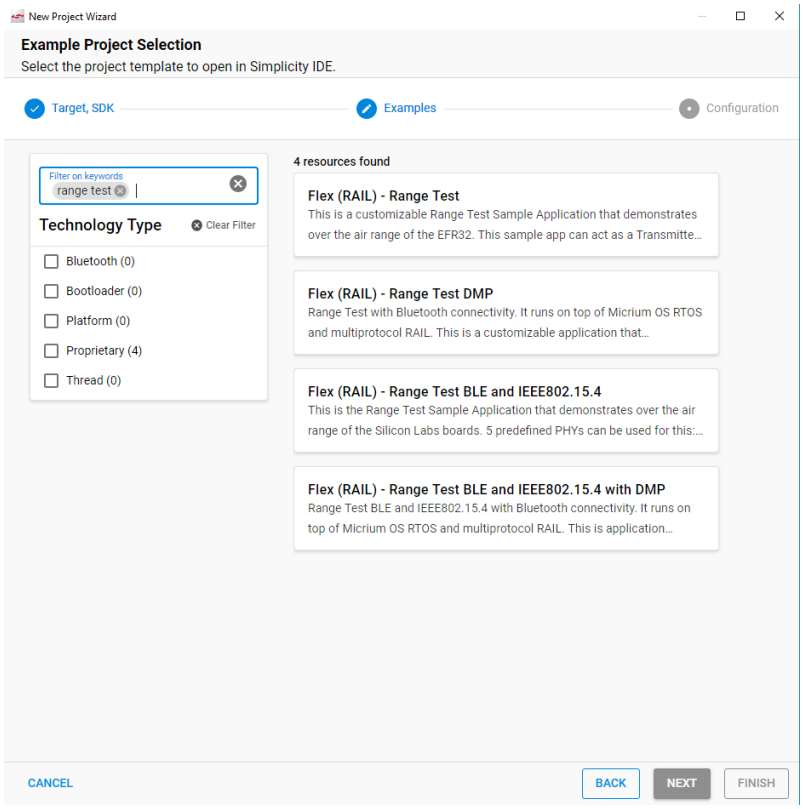


Figure 2.4. Example Browser in the New Project Wizard

Rename the project, if desired. The project is placed into your current workspace by default, but you can select a different location. Finally, either copy the sources or link to them. Click [**Finish**]. Project creation may take a short time.

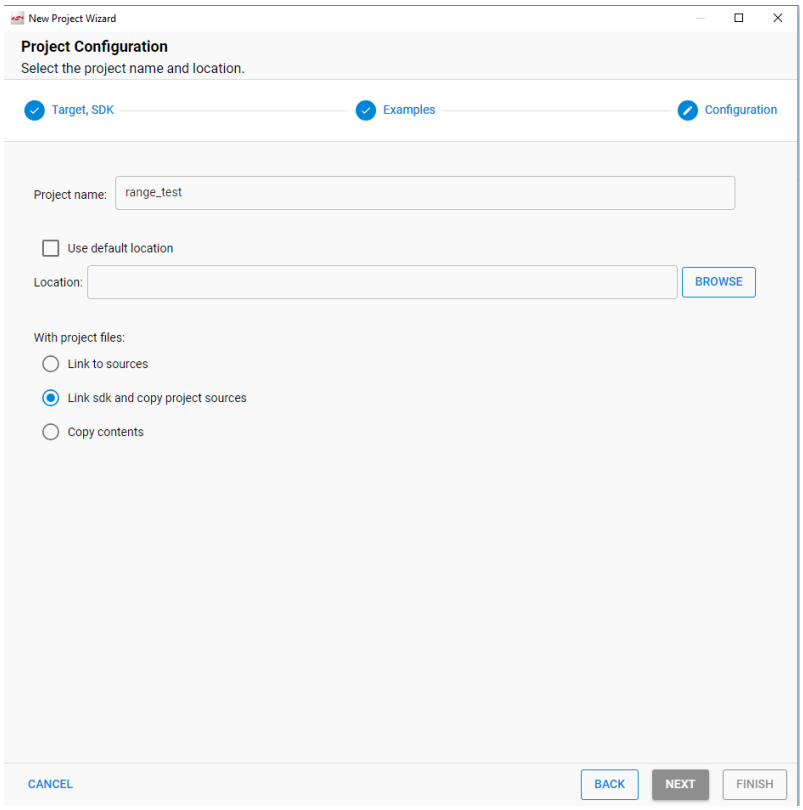


Figure 2.5. Configuration in the New Project Wizard

Note: Creating the project means copying or linking all the required source code into your project's folder, setting up project configurations based on the installed components and generating all the project customizations (basically through header files) according to the configurable components' default settings.

2.3 Customizing the Application

If you have created a custom PHY application, the Radio Configurator interface (stored in the .radioconf file) is displayed. Here you can set up the desired PHY. You can browse through the built-in PHYs or create a custom configuration (recommended only for experienced users). Saving this file generates a new *radio_config.c* file. For more information on how to set the modem parameters, refer to AN1253: EFR32 Radio Configurator Guide for Simplicity Studio 5.

Otherwise, for a Standard PHY application, the project's, the *.slcp* file is opened in the project editor tool called the Project Configurator. Generally, and if you are working with development kits, you should not modify this file. The project is ready to build.

Note: If you closed the Radio Configurator, you can re-open it from the Project Configurator's Configuration Tools tab.

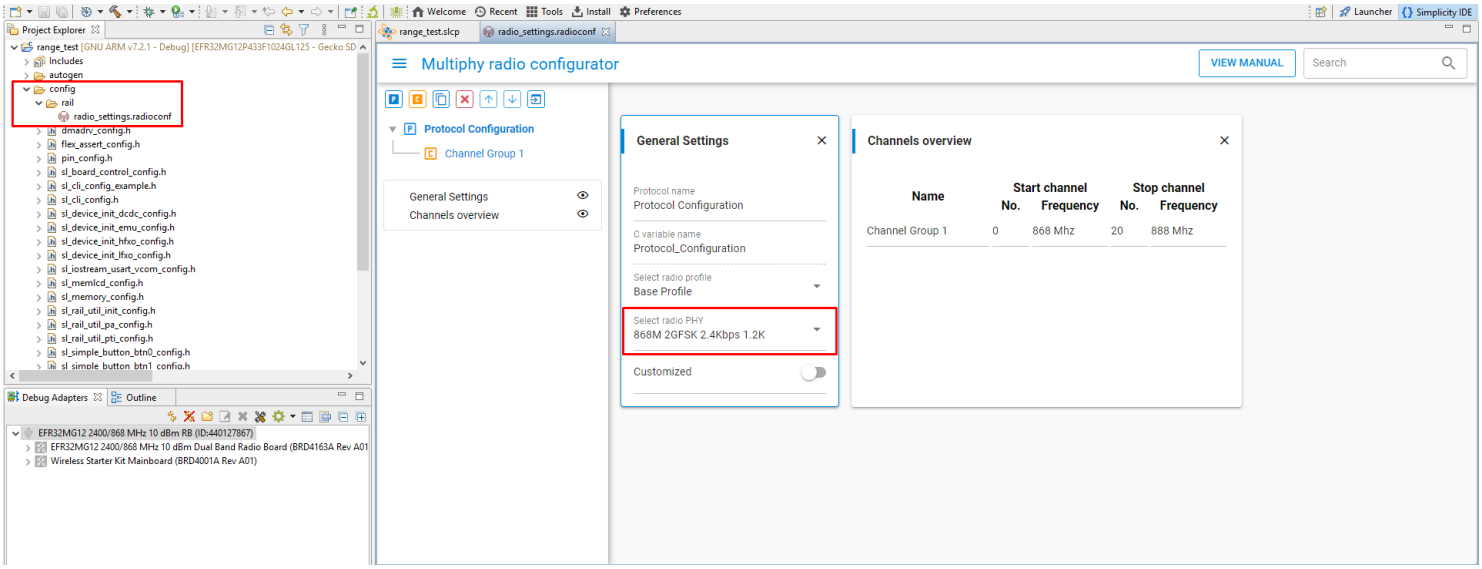


Figure 2.6. The Radio Configurator Interface

2.4 A Note on Modifying the Application for Custom Boards

Though the default Range Test user interface is the mainboard's LCD and push buttons, the application can be driven exclusively by CLI if desired. Hence, with some modifications, Range Test applications can run on custom designs that lack LCD and push buttons.

The standard single protocol version of Range Test simplifies this task, reducing the effort required to purge LCD and push button support from the application to two simple steps. In the Component Editor, uninstall the following components (search for the component name to locate):

- **Simple Button**
- **GLIB Graphics Library**

Doing so removes all dependencies but retains the **Sleeptimer** component. Macros in the application source code will then exclude from the build sections related to the LCD and push buttons, enabling the project to compile with only CLI support. Porting this to a custom board may require further modifications to translate the mainboard default settings (for example, the USART configuration) to the custom design, but the steps above will produce a viable CLI-only application that can be demonstrated on a mainboard before transitioning to your custom target.

2.5 PHY Limitations

Range Test applications work with a fixed packet length by default. Some profiles (Long Range, Connect, wM-Bus) define variable length packet configurations in their PHYs, but length configuration will be overridden with the fixed length setup configured on the application's menu. This limitation should be considered when designing the protocol for evaluation.

2.6 Build the Application

When you have finished configuring the project, click **Build** (hammer icon) on the toolbar.

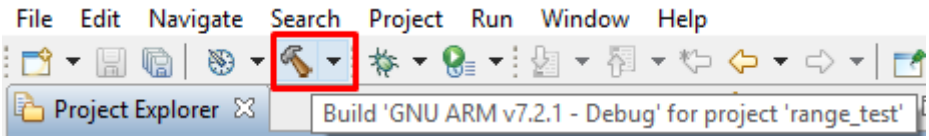


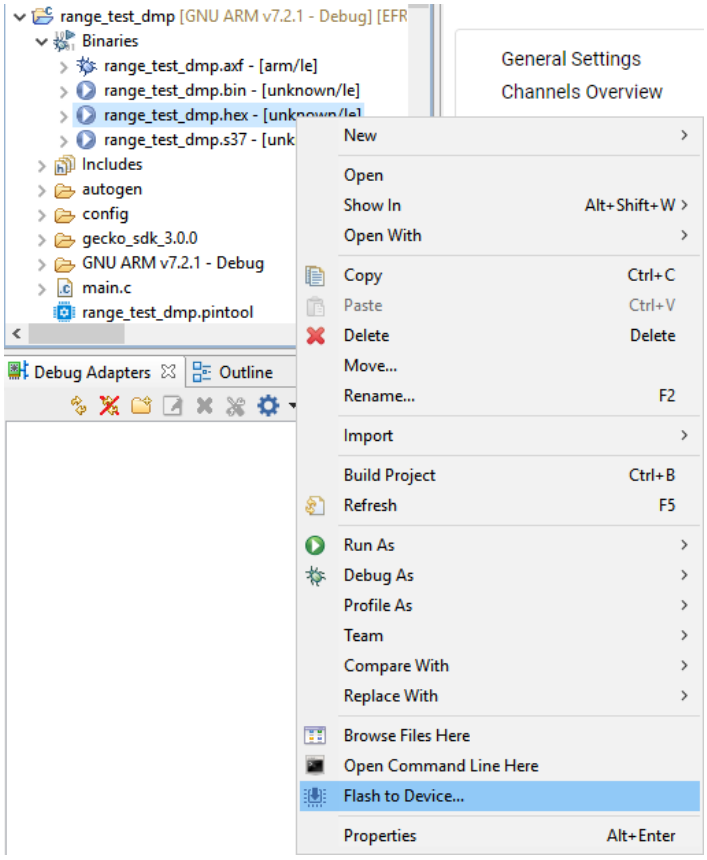
Figure 2.7. Build Icon

Your sample application will be compiled based on its build configuration.

2.7 Load the Application onto a Device

Once the binary file has been generated you can flash it onto a connected device with the following options:

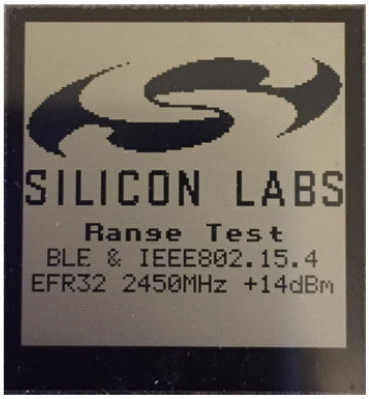
1. Use **Flash to Device**: Right-click the image file in Project Explorer view and select **Flash to Device** to open the Flash Programmer.



2. Use Simplicity Commander: Run the executable file located under Simplicity Studio's installation folder in the `developer/adapter_packs/commander` folder, or open it through the **Tools** button on the toolbar or the Project COMPATIBLE TOOLS tab.
3. Use the debug functionality: Click **Debug** (bug icon) on the menu to (incrementally) build the project if any modifications have been made since the last build. With this mode all the debug functionality will be available.

3. Using the Range Test Applications

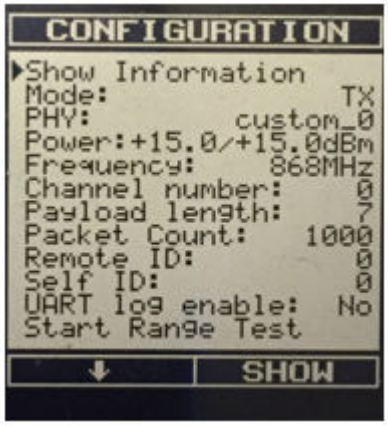
On startup, the Welcome Screen is shown on the LCD. This includes the Silicon Laboratories logo, the carrier frequency and the RF power level. The Welcome Screen is shown for up to three seconds, or as long as any push button is held down. The following figures show the Welcome screen for Custom PHY applications and Standard PHY applications, respectively.

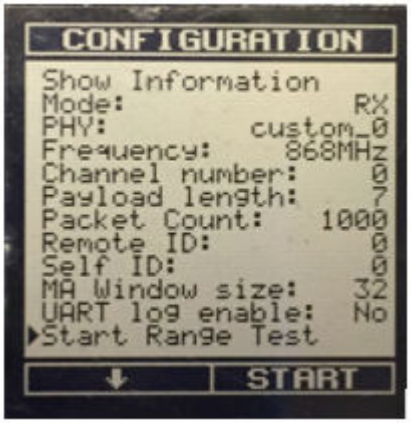


3.1 Configuration

3.1.1 Configurable Parameters

Next, the LCD displays the menu, on which the (mostly configurable) parameters and functions are displayed. The menu differs slightly, depending on the selected mode. The **Show Information** and the **Start Range Test** functions are also displayed. Selecting **Show Information** refreshes the screen to show the Welcome Screen again. The following figures show the menu for TX and RX.





The menu options are:

Common

- **Mode** (RX or TX): The applications start in RX mode by default, but the mode can be changed any time, before or between measurements. In addition, the same device can act as a transmitter and then as a receiver without resetting the device.
- **PHY:** Only configurable in a Standard PHY application (BLE 1/2Mbps or 802.15.4) and if you have a Multi-PHY radio configuration in a Custom PHY application.
- **Frequency:** Not a configurable parameter, but shows the selected PHY's base channel frequency. Therefore it is not updated when the Channel number is increased.
- **Channel number:** Selects the channel on which the RX node(s) will listen or the TX node will transmit.
- **Payload length:** Configurable between 7-64 in a Custom PHY application. In a Standard PHY application it can be configured to between 5-24 with a BLE PHY or 5-116 with an 802.15.4 PHY.
- **Packet count:** Ranges from 500 to infinite in both Modes, and sets the transmitted/expected number of packets on the participants.
- **Remote/Self ID:** In RX Mode, the radio listens on the given channel and inspects the packets received. Only packets that are sent with matching **Self ID** and **Remote ID** will be processed. Therefore the IDs should be configured on the TX accordingly.

RX/TX only Parameters

- **MA Window size (RX only):** Moving Average or MA window size configures how many packets will be counted in a shorter PER measurement during the test. The moving average PER is useful when trying to position the devices until you get a specific PER, for example to find the maximum distance between the units for 1% PER. The moving average in this case shows an almost instantaneous PER. When the positions are fixed you can re-run the test and the total PER for many packets will provide more accurate result.
- **Power (TX only)**: Output power can be set in the LCD menu in 0.5 dBm steps (power setpoint), between -15 and +20 dBm. Actual minimum and maximum power may vary on different frequencies and different parts. Also, the target power may differ from the power that is set by RAIL. The LCD menu informs the user about the setpoint and the actual power in that order.
- **UART log enable (TX only):** If on, a status message can be observed on the UART TX line for each radio packet, formatted in human readable format. The default pin assignment is the standard VCOM port available on the mainboard.

  Note: This mode will not interfere with the command line interface (CLI). The CLI will be available regardless of whether or not the UART log is enabled.

Many of these options are also available in the EFR Connect smartphone app. After the menu items are configured, the range test can be started. During the test, all of the measured information can be observed on the LCD or the smartphone's screen.

3.1.2 Configure on the LCD Menu Using Push Buttons

The two push buttons on the mainboard are used to navigate through the menu system; soft labels on the bottom of the screen describe the current function of each button. In general, button 1 is used to navigate down through the menu items. Button 0 is used to configure the menu item selected by the pointer. Items with a **+** sign in the right bottom label are configurable.

3.1.3 Configure Using the Command Line

You can access all the configurable parameters (not depending on the Mode) through the CLI. The parameters can be written or read by commands starting with `set` or `get`, respectively.

To see the available commands and the available arguments issue, the `help` command.

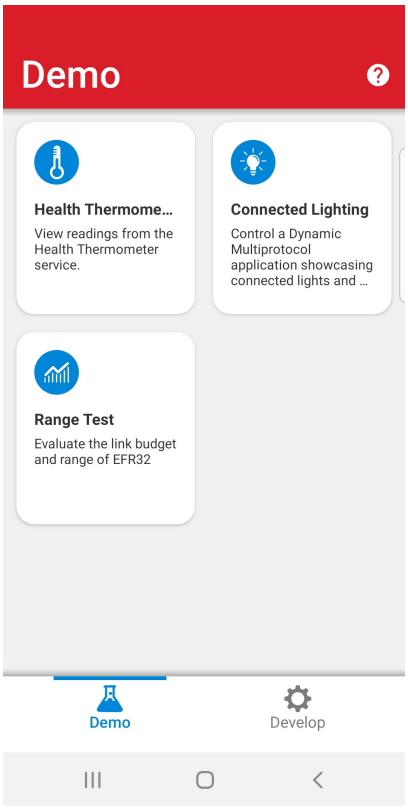
3.1.4 Configure Through the EFR Connect Smartphone Application

Many configurable parameters can be set through the EFR Connect smartphone application, once a connection has been established. See section 3.1.1 Configurable Parameters for more information

3.2 Using the EFR Connect Smartphone Application

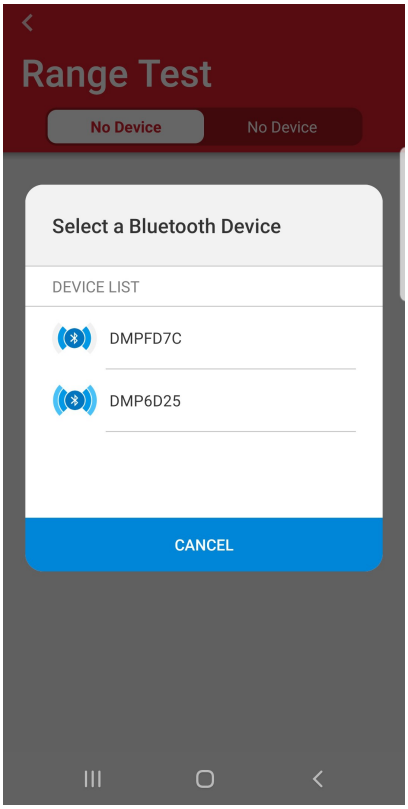
The Silicon Labs EFR Connect app uses the Bluetooth adapter on your phone/tablet to scan, advertise, connect, and interact with Bluetooth LE devices.

Documentation for EFR Connect is available through docs.silabs.com. When the EFR Connect smartphone app is opened, the following screen is displayed.

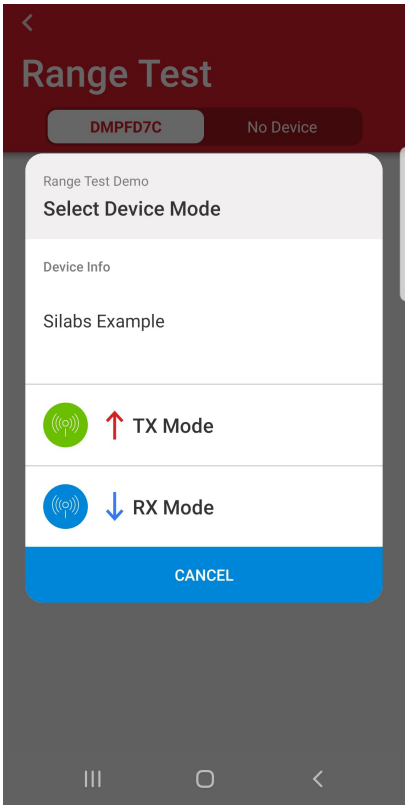


Select **Range Test Demo** from the list of available applications.

Select the kit to which you would like to connect. The IDs are the last 4 bytes by of the MCU's Unique ID.



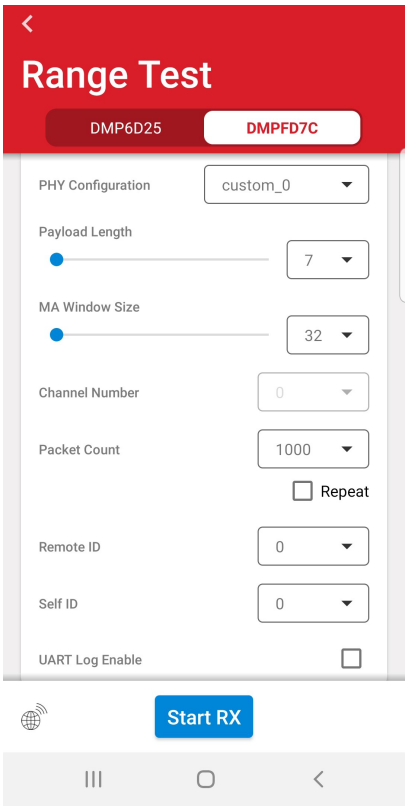
Select the application's mode: TX or RX.

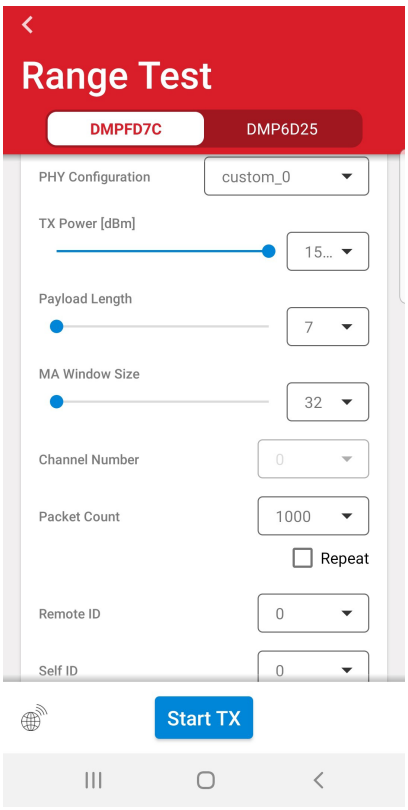


Selecting a mode establish the connection, and the device stops advertising as long as it is connected to the smartphone. Therefore you should reconnect to the device to change its mode.

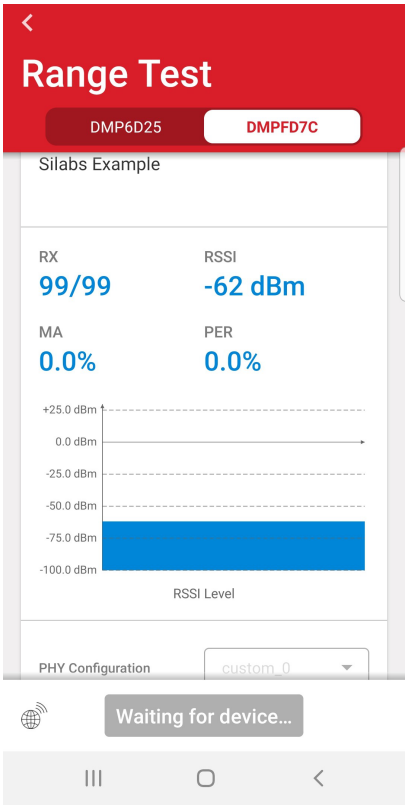
Note: An ongoing RX measurement will block re-connection, since the device will not advertise while it is waiting for the TX node's packets. To reconnect, first terminate measurement either by using the push buttons or by a CLI command.

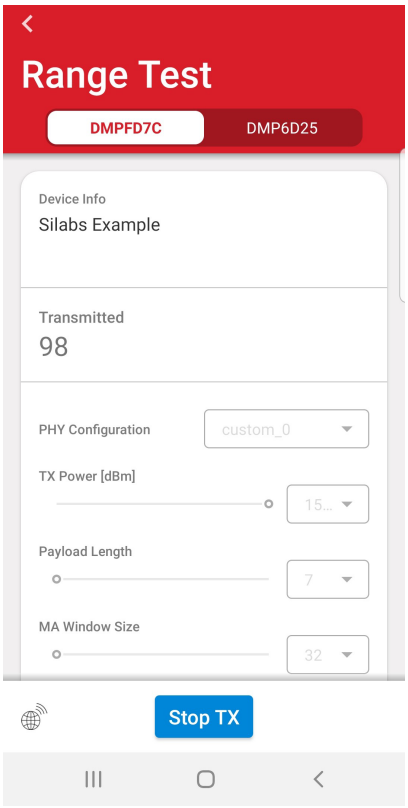
The following figures show the starting screen for RX and TX.





The following screens shows RX and TX mode with ongoing measurements.





The same configuration parameters are available as on the LCD.

Notes:

- Do not change the Mode using the push buttons or a CLI command while the connection is established between the device and your smartphone.

- If you measure Bluetooth LE PHY in a Standard PHY application, the connection will be created through a separate Bluetooth LE protocol, with the Bluetooth LE stack, and will not interfere with the measurement.
- If you start an RX measurement, stop it before trying to reconnect to the device. Otherwise the device will not advertise itself. However, during TX measurement the device continues to send advertising packets.
- The smartphone application has a much longer RSSI history than is available on the mainboard's LCD.

3.3 Application Execution Tips

The test on the TX side runs as long as the number of transmitted packets reaches the predefined number or until the test is terminated by pressing Button 0. You can follow the number of transmitted packets on the transmit-side's LCD.

The RX side restarts whenever it receives a lower-indexed packet than the last one with the expected **Remote** and **Self ID**. The signal strength of the incoming packet is measured during packet reception, and the actual RSSI value is shown on the LCD. The RSSI values are also presented as a graph.

The number of lost packets and the packet error rate are defined only at the receive side and are based on the first and last received packet numbers, regardless of the **Packet count** parameter.

The RSSI is typically used to qualify the link: a higher level might show a better link quality. The actual RSSI value is measured when the sync word of the packet is received (for more details see `RAIL_RxPacketDetails_t.rssi`).

It is not necessary to start both sides synchronously as well as to receive the first **N** packets.

The range test can be performed inside a building if indoor propagation is tested. However, line-of-sight testing outside the building is recommended to get the best possible range result, as well as the best comparable results from different settings. It is also recommended that the antennas be located at least 1.5 m above the ground.

If PER < 1%, reset the current measurements on the boards and try moving the two devices closer. Propagation conditions usually improve if you distance yourself from a possibly faded area.

4. Range Test Applications' Metrics

Range Test creates a radio link between the evaluation kits and sends a predefined number of packets from the transmitter to the receiver node. The receiver node continually recalculates and reports PER during the entire measurement. The application displays the current Received Signal Strength Indicator (RSSI) level in dBm units and draws a chart of the RSSI historical data. For long tests the transmitter can be set up to transmit infinitely.

4.1 Range Test's Packet Assembly

Simple PHY Range Test applications' packet structure includes a packet counter, a destination and a source id, as shown in the source code below.

```
typedef struct range_test_packet_t{
  uint16_t packet_counter;    ///> Value showing the number of this packet.
  uint8_t  destination_id;    ///> Destination device ID this packet was sent to.
  uint8_t  source_id;         ///> Device ID which shows which device sent this packet.
  uint8_t  repeat;            ///> Unused.
} range_test_packet_t;
```

In 802.15.4 mode the application sets up a similar packet payload, immediately after setting up the MAC header (MHR) field. Both destination address and PAN ID are set to broadcast (0xFFFF), while the source address is 0x0000.

```
  data_frame->payload.source_id = range_test_settings.source_id;
  data_frame->payload.destination_id = range_test_settings.destination_id;
  data_frame->payload.packet_counter = packet_number;
  data_frame->payload.repeat = 0x00;
```

In BLE mode the application sends non-connectable undirected advertisement packets. The packet number and the IDs are configured as shown in the following code snippet:

```
  ble_tx_pdu->manufactSpec.payload.packet_counter = packet_number;
  ble_tx_pdu->manufactSpec.payload.destination_id = range_test_settings.destination_id;
  ble_tx_pdu->manufactSpec.payload.source_id = range_test_settings.source_id;
  ble_tx_pdu->manufactSpec.payload.repeat = 0xFF;
```

In each cases the remaining part of the packet/payload is padded with `0x55` and `0xAA` at the odd and even byte indexes, respectively.

4.2 Definition of Packet Error Rate

Packet error rate is calculated according to the following equation:

$$Packet\ Error\ Rate(PER)[\%] = \frac{(P_{Tx} - P_{Rx})}{P_{Tx}} * 100$$

Where *PTX* is the number of sent packets and *PRX* is the number of received packets.

5. Expected Results

The following table contains expected range test results for the EFR32FG1 series boards under conditions recommended in section 3.3 Application Execution Tips.

Table 5.1. Test Measurements with EFR32FG1 Series Boards

Configuration	Tx Power (dBm)	Rx Sensitivity (dBm)	PCB Antenna Gain (dB)	Range in Dry Weather Condition (m)	Range in Wet Weather Condition (m)	Link Budget (dB)
169 MHz 2GFSK 2.4 kbps 1.2 kHz	19.5	-124	-12.5	3000	480	118.5
169 MHz 2GFSK 38.4 kbps 20 kHz	19.5	-112	-12.5	1200	240	106.5
169 MHz 2GFSK 500 kbps 125 kHz	19.5	-97	-12.5	380	100	91.5
433 MHz 2GFSK 2.4 kbps 1.2 kHz	10	-122.5	-0.5	4400	640	131.
433 MHz 2GFSK 50 kbps 25 kHz	10	-110	-0.5	1600	310	119
433 MHz 2GFSK 100 kbps 50 kHz	10	-107.5	-0.5	1300	270	116.5
490 MHz 2GFSK 2.4 kbps 1.2 kHz	19.5	-123	0	9400	1100	142.5
490 MHz 2GFSK 10 kbps 5 kHz	19.5	-118	0	6400	850	137.5
490 MHz 2GFSK 38.4 kbps 20 kHz	19.5	-111.6	0	3900	590	131.1
490 MHz 2GFSK 100 kbps 50 kHz	19.5	-107.8	0	2900	470	127.3
868 MHz 2GFSK 2.4 kbps 1.2 kHz	13	-118.2	0	2700	440	131.2
868 MHz 2GFSK 38.4 kbps 20 kHz	13	-109.1	0	1300	260	122.1
868 MHz 2GFSK 500 kbps 125 kHz	13	-100	0	670	150	113
915 MHz 2GFSK 0.6 kbps 0.3 kHz	19.5	-125	0.5	7800	990	145.5
915 MHz 2GFSK 50 kbps 25 kHz	19.5	-107.9	0.5	2100	370	128.4
915 MHz 2GFSK 100 kbps 50 kHz	19.5	-105.5	0.5	1700	300	126
915 MHz 2GFSK 500 kbps 175 kHz	19.5	-98	0.5	980	200	118.5
2.4 GHz 2GFSK 250 kbps	19.5	-99	1-3	280	105	122.5
2.4 GHz 2GFSK 1 Mbps	19.5	-93.1	1-3	180	75	116.5

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com