AN655

RANGE TEST APPLICATION FOR EZRADIO® AND EZRADIOPRO®

1. Introduction

The range evaluation demo provides an easy way to evaluate the link budget of EZRadio® and EZRadioPRO® devices by performing a range test between two nodes. 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.

2. Supported Radio Types

The following RF ICs are supported by the range evaluation demo:

- Si4012 Transmitter
- Si4355 Receiver
- Si4455 Transceiver
- Si4060 Transmitter
- Si4063 Transmitter
- Si4362 Receiver
- Si4460 Transceiver
- Si4461 Transceiver
- Si4463 Transceiver
- Si4464 Transceiver
- Si4438 Transceiver
- Si4467 Transceiver
- Si4468 Transceiver

3. Development Kits

The range evaluation demo can run on the development boards described in the following subsections.

3.1. Wireless Mother Board Hardware Platform

The wireless motherboard platform is a demo, evaluation, and development platform for radio ICs. It consists of a wireless motherboard and interchangeable MCU and RF Pico boards.

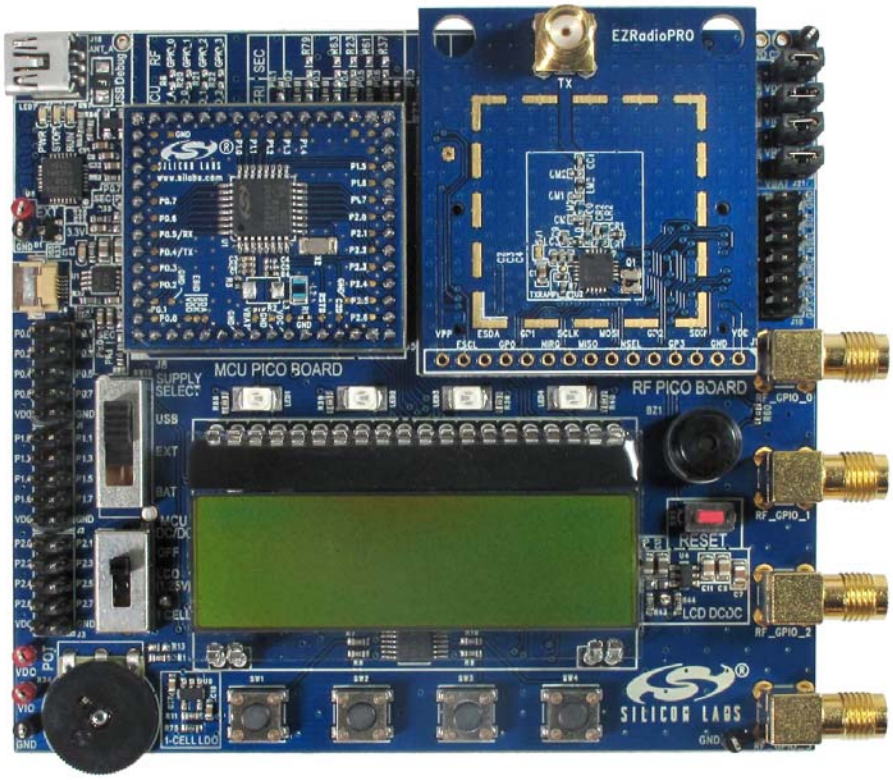


Figure 1. Wireless Motherboard Platform

3.2. LCD Base Board Platform

The LCD Base Board is a development board that can be used with a connected RF Pico board.

Figure 2. LCD Base Board Platform

4. Basics of the Packet Error Rate

The range test demo provides measured results regarding the quality of the RF link. The demo uses two RF nodes. In the one-way link range test, one node is used as the “transmitter” (TX) and the other as the “receiver” (RX). The transmitter sends packets to the receiver repeatedly. The packet includes the address of the transmitter and number of the sent packet. The packet number increments from packet to packet. The receiver receives the packet and checks its address. If their addresses match, the packet number is stored. In the two-way link range test, the receiver can send an acknowledge packet (ACK) back to the transmitter. The ACK packet also includes the address and packet number of the received packet.

Packet error rate can be calculated with the following equation:

$$\text{Packet Error Rate (\%)} = \frac{(P_{TX} - P_{RX})}{P_{TX}} \times 100$$

where P_{TX} is the number of sent packets and P_{RX} is the number of received packets.

4.1. Packet Structure

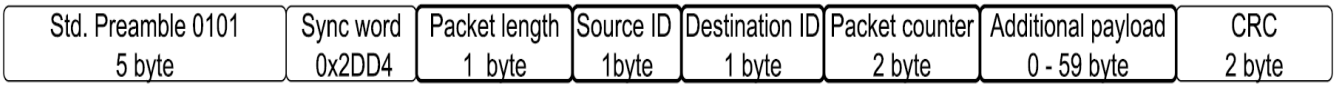


Figure 3. Packet Structure

5. Prerequisites for Code Development

The range test project has a unified structure and common driver set. This section provides a brief introduction of the structure of the range test projects. This structure may be familiar to customers who have already used the exportable example projects from WDS.

The settings in the sample project files assume that some Silicon Labs or third-party software tools are already installed on the PC on which the sample project will be compiled. The tools that need to be installed depend on the functionality to be used. The following list contains a complete set of such programs:

- **Silicon Laboratories IDE**
  Used to open the preconfigured project files and manage the build process.
- **Keil C51 v9.0 (or higher)**
  Compilers to use with the Silicon Laboratories IDE to manage build process.
- **Silicon Labs Flash Programming Utility (optional)**
  Needed only if programming outside the Silicon Labs IDE is necessary.
- **Make (optional)**
  This tool is needed in case another compiler is used or the build process takes place outside of the SiLabs IDE. The “Makefile” required by this tool is already included in the project. However, it is only recommended for advanced users since it may require manual editing.

6. Deploying Code to the Silicon Labs IDE

The Silicon Labs integrated development environment (IDE) is a standard tool for program development for any Silicon Labs 8-bit MCUs, including the C8051F93x and C8051F91x that are used on the hardware platforms. The Silicon Laboratories IDE integrates a project manager, source-code editor, source-level debugger, and an in-system flash programmer. The IDE interfaces to third-party development tool chains to provide system designers a complete embedded software development environment.

The Range Test application is intended to be built via the Keil C51 toolchain. The project deployed from WDS comes with a predefined project file that is already configured to use the Keil C51. The Keil Demonstration Toolset includes a compiler, linker, and assembler and easily integrates into the IDE.

6.1. Workflow for Downloading and Running a Project

Perform the following steps to download and run a project:

1. Connect the hardware platform to the PC according to the description of the platform used.
2. Start Silicon Labs IDE (IDE 4.40 or higher required) on your computer.
3. Select *Project→Open Project...* to open the Range Test project file.
4. Before connecting to the target device, several connection options may need to be set.
   a. Open the *Connection Options* window by selecting *Options→Connection Options...* in the IDE menu.
   b. Select *USB Debug Adapter* in the *Serial Adapter* section.
   c. If more than one adapter is connected, choose the appropriate serial number from the drop-down list.
   d. Check *Power target after disconnect* if the target board is currently being powered by the USB Debug Adapter. The board will remain powered after the software is disconnected by the IDE.
   e. Next, the correct *Debug Interface* must be selected. Check the C2 Debug Interface.Once all the selections are made, click the “OK” button to close the window.
5. Click the “Connect” button in the toolbar or select *Debug→Connect* from the menu to connect to the MCU of the platform.
6. Erase the flash of the MCU in the *Debug→Download object code→Erase all code space* menu item.
7. Download the compiled HEX file either by hitting the “Download code” (Alt+D) toolbar button or from the *Debug→Download object code* menu item.

7. Directory Structure of the Range Test Code

The range test code has a common directory structure with separate source and project files to ease understanding of the individual modules. For every sample project, the following directories and files can be found in the main directory:

- **bin**
  Contains the SiLabs project files for Keil compiler and the Makefile if the make tool is used instead.
- **doc**
  Doxygen-generated documentation based on comments inside the source files in html format.
- **out**
  The outputs of the compilation process are sent to this folder. After a successful compilation, this directory contains files, such as the hex file, the linker output, and the OMF file.
- **src**
  Directories containing the source files.
  a. Application
  b. Common
  c. Drivers
- **Doxyfile**
  This file contains the Doxygen documentation generator settings.
- **Cleanup.bat**
  This is the batch file used to delete all files generated during the build process.

The individual software modules are organized into several source files. The sample projects contain one header file (bsp.h) that is included in the source files and collects the individual headers that need to be included. Under the “src” folder, the “application” folder contains application-related sources. The “common” directory contains software modules that are intended to separate the hardware and radio-dependent drivers and low-level modules and provide an independent API to the application layer. These middle-layer modules are dependent on the services provided by the low-level drivers, and their compiled object files are highly dependent on the actual defined platform and radio. The low-level driver modules (e.g., handlers, drivers) are located under the “driver” directory.

8. Software Layers

Like all of our sample projects, the layered software approach is followed. There is a distinct scope for each software module, and all modules can communicate through each other's API functions. The software modules are separated and focused to cover one specific task. The following figure shows the software layers and its relations.

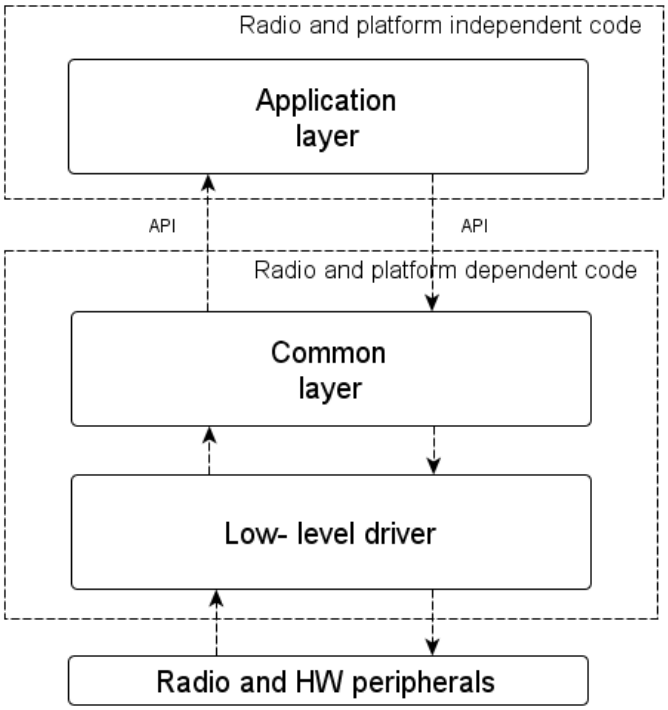


Figure 4. Software Layers of the Range Test

8.1. Application Layer

The application layer is located at the top of the modules hierarchy. It logically controls the whole range test demo. Functionally, it is independent of the hardware peripherals and the radio, and its logical operation can remain unchanged even if the layers below have been modified later. It can be adapted to any device without difficulty. Generally, it can control the radio to work as a transmitter, a receiver, or a transceiver. It calculates the moving average packet error rate based on the transmitted and received packets. It maintains the graphical menu on the LCD screen and supervises the interaction of the user. In the menu, the preloaded radio settings generated by the WDS can be configured. The radio settings are located in the sRadioConfig.h header. This header file contains the custom and preloaded radio configurations according to the actual RF Pico board and the host platform. This file has been automatically generated and is not allowed to be modified manually.

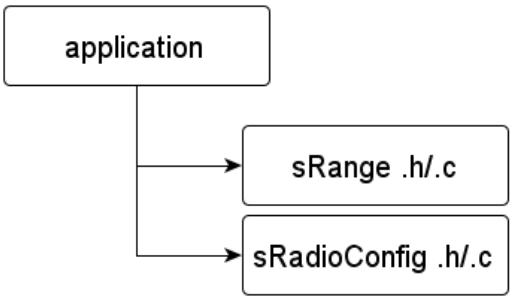


Figure 5. Application Layer Related Files

8.2. Common Layer

In the modules hierarchy, the common layer works between the low-level driver and the application layer. Functionally, it is dependent on the hardware peripherals and the radio. It is responsible for showing the dynamically changing menu that is updated on the LCD screen. It can store and load the actual configuration from and to the FLASH memory. It can send log information about packet error rate to the UART interface that can be processed by a monitoring computer. If any errors are encountered during the range test demo, it can provide helpful information about the sources of these errors. It can control the radio RFIC to send and receive packets. It can request information from the radio about its internal state, such as the current RSSI value during reception, and it can check whether or not a packet transmission has successfully occurred.

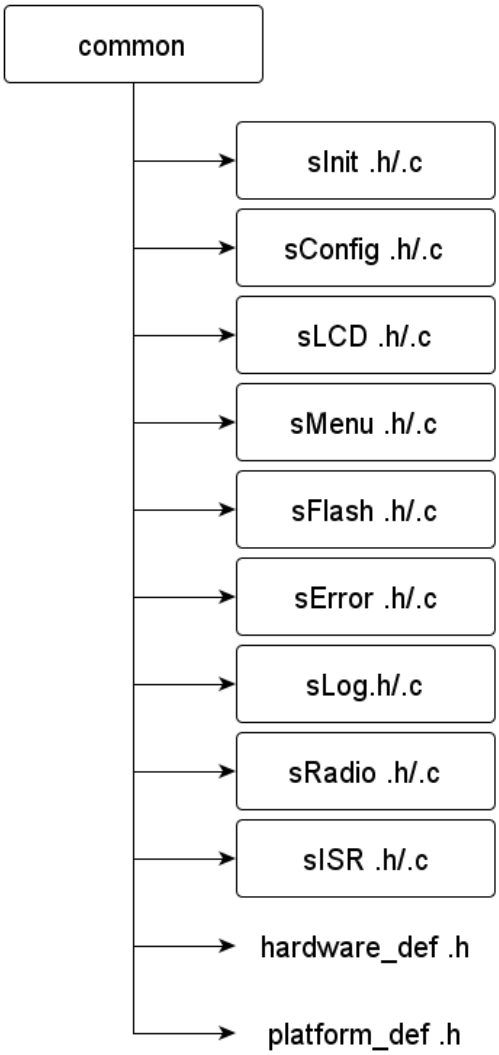


Figure 6. Common Layer Related Files

8.3. Low-Level Driver

In the modules hierarchy, the low-level driver is located between the common layer and the hardware. It consists of a set of interfaces that provide possible options for controlling various peripherals on modular HW platforms. Registers can be initialized with preconfigured settings, and peripherals can be enabled to start/stop their own processing. The major tasks of these software modules are to initialize the hardware elements and control their behaviors. The principle of their installation is to provide a façade for the upper layers. Functionally, the application layer at the top of the hierarchy can be independent of the hardware, and its logical operation can remain unchanged even if the hardware has been modified later. All the modules are primarily responsible for handling the dedicated internal peripherals, such as the IO, timers, SPI and PCA, SMBUS, UART, and LCD.

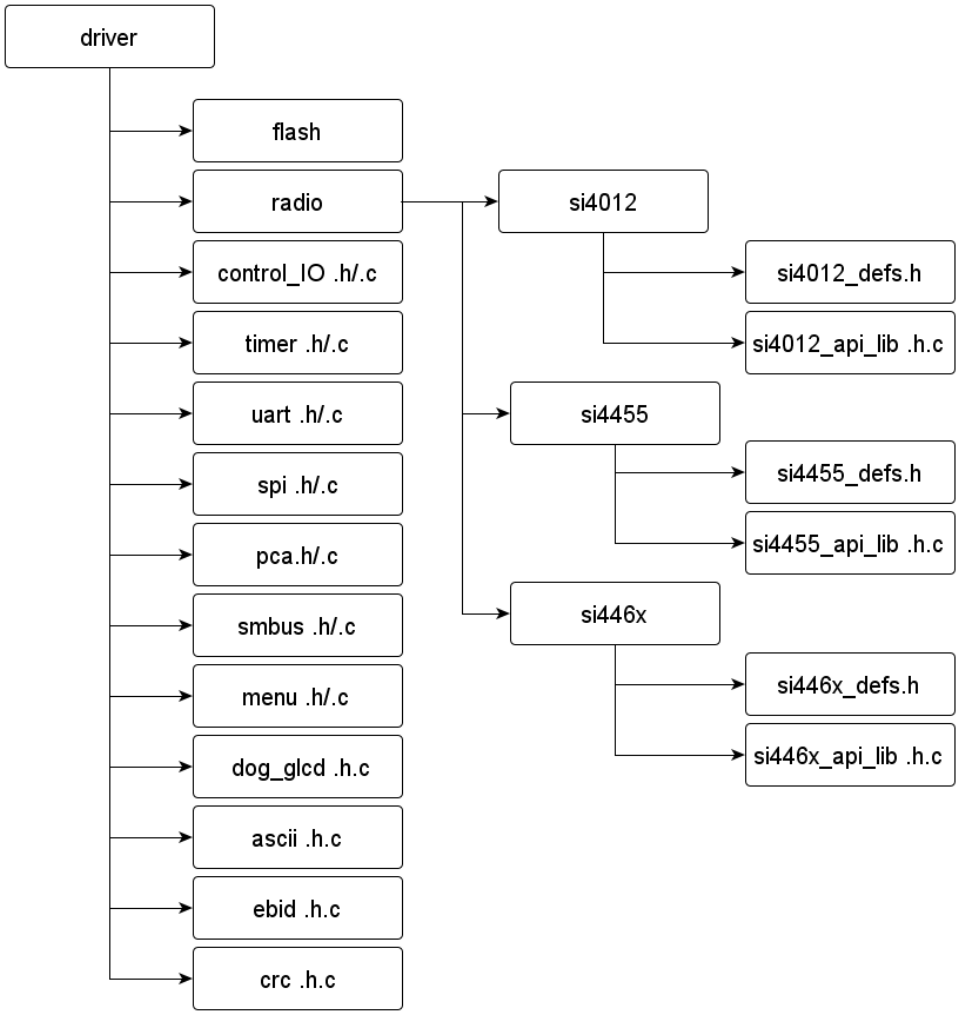


Figure 7. Low Level Driver Related Files

More detailed information about the radio drivers can be found in the following application notes:

- AN692: Si4355/Si4455 programming guide
- AN633: Programming Guide for EZRadioPRO® Devices.

9. Menu System

After running the demo, the first screen is the welcome screen. It shows the Silicon Laboratories logo as long as any of the buttons are pressed. If the motherboard cannot recognize the type of RF Pico board, then it shows an error message. In this case, it is recommended that the user follow the instruction displayed on the screen.

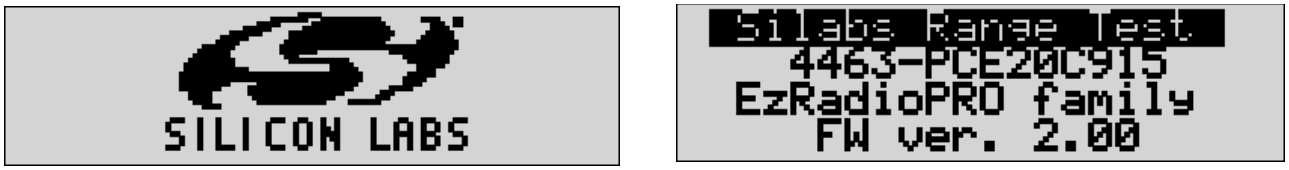


Figure 8. RF Pico Board Identification

The on-screen menu system is designed for easy configuration. For accurate range testing, the demo measures the actual packet error rate (PER) of the radio link. The RF settings of the radio can be configured based on data rate, modulation, and frequency settings, output power, packet count, and payload length. It is also possible to change the self- and remote-IDs of the participants. The menu also provides an option to select how the strength of the received signal (RSSI) is shown on the screen. It can be represented as a moving graph if “Icon” is selected, a decimal number if “Number” is selected, or a decibel value if “dBm” is selected. The signal strength of the incoming packet is measured during packet reception. If the RSSI is presented as a graph on the screen, the dynamic change in RSSI value can be observed accurately by configuring the moving average window size. The actual RSSI value is latched and shown when the sync word of the packet is received. The RSSI is typically used to qualify the link: a higher number or greater level shows a better link quality. For more information about how the decimal number relates to actual received strength value, refer to the EZRadio/EZRadioPRO data sheet. The log functionality can be enabled to provide information about the transmitted and received packets.

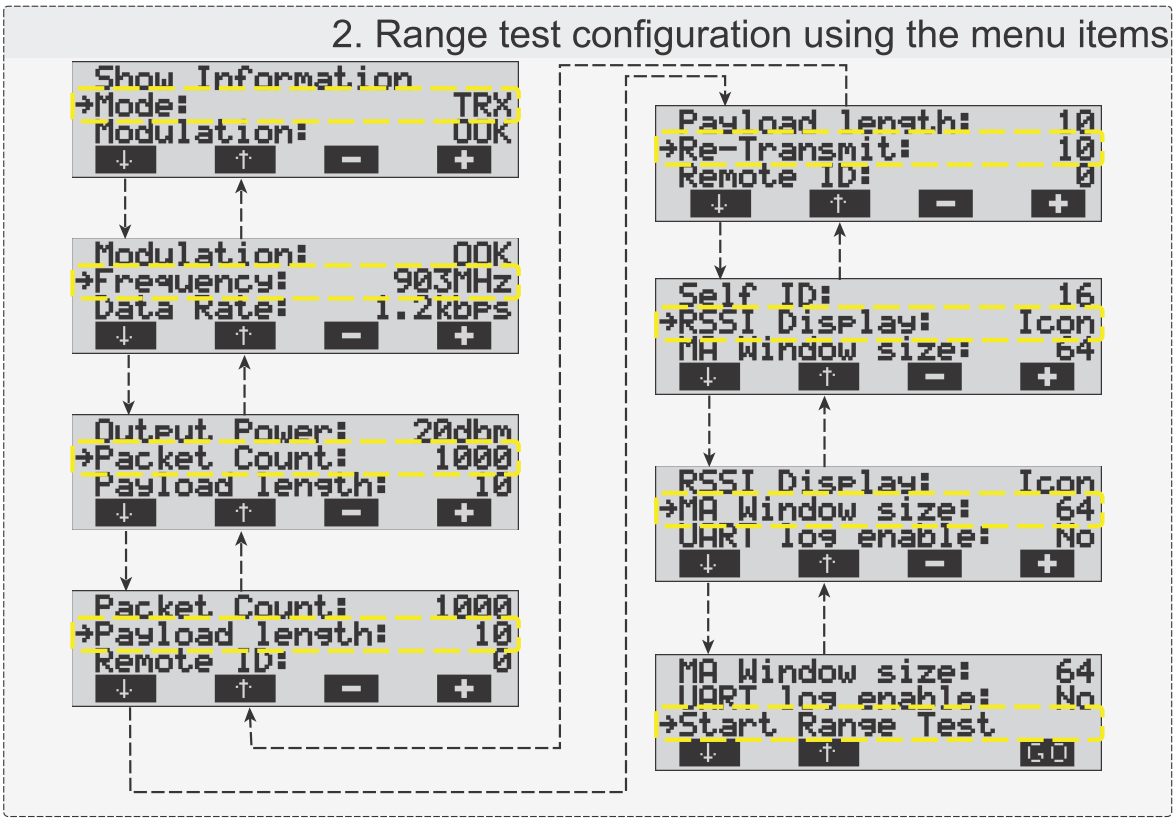


Figure 9. Configuration Menu

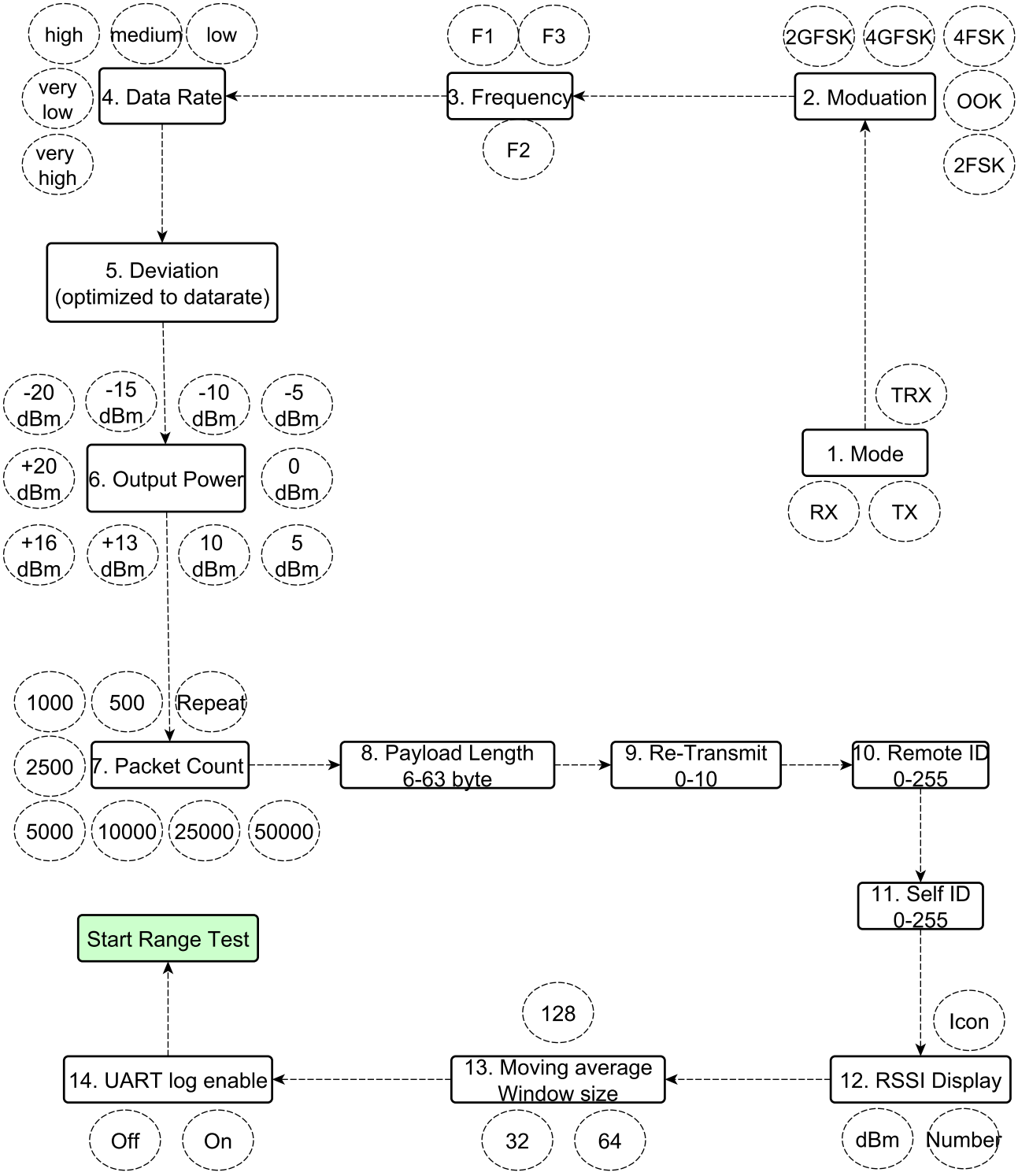


Figure 10. Menu System

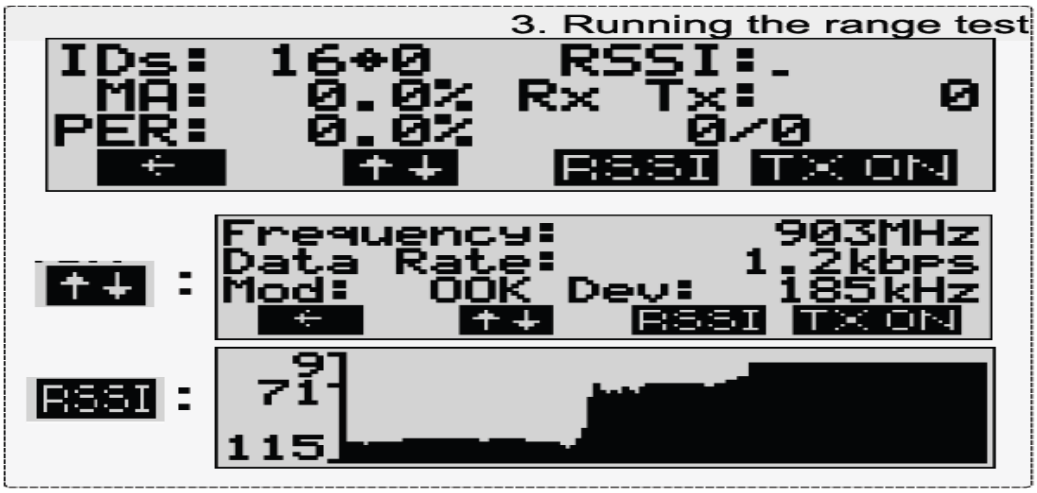


Figure 11. Range Test Information

Four push buttons are used to navigate the menu system; soft labels describe the current function of the given buttons. In general, push buttons 1 and 2 are used to navigate down and up through the menu items. Push buttons 3 and 4 are used to configure the menu item selected by the pointer. The demo can be configured through several menu items. 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 screen. It is also possible to switch to the “RSSI graph” and the “runtime info” screens.

10. Range Test

Perform the following steps to conduct the range test:

1. Connect RF Pico boards to the two hardware platforms.
2. Put batteries into the devices and switch the devices on.
3. Connect the hardware platforms to the Wireless Development Suite.
4. Select the Range Test project and configure your custom settings through the “Frequency and Power”, “RF parameters” (etc…) tabs. Nonetheless, WDS also appends additional RF “canned settings” that are optimized to the RF Pico board. Deploy project either by “Download Project” or “Generate Source”.
5. Configure both devices to be in TRX mode or one of them to be RX and the other to be in TX mode. Note that Si4355-based devices can only be receivers, and Si4012-based devices can only be transmitters.
6. Configure the range test through the menu items. Select either from the “canned settings” or the custom setting.
7. Confirm that the transmitter sends packets and that the receiver answers. The range test can be performed inside a building if indoor propagation is tested. However, it is advised to perform the test outside the building, line-of-sight, to get the best possible range result.
8. If PER>1%, reset the PER on the transmitter and try to walk further in the area. Propagation conditions usually improve if the user distances himself from a possibly shadowed area.

11. Radio Configuration Settings to Test

The Wireless Development Suite provides several optimized radio configurations and one additional custom configuration to test. Basically, the modulation type can be OOK, 2FSK, 2GFSK, 4FSK, or 4GFSK. According to the RF matching network of the connected RF Pico board, WDS determines the frequency band and offers three different frequency options in the given band by default. Given the modulation type and the concrete center frequency, WDS offers four different data rate options. As a result, the maximum combination of automatically-generated configurations can be up to 60. They are stored in a linked list. The custom radio configuration is appended to the end of the list. WDS automatically generates the list and puts it into the range test project.

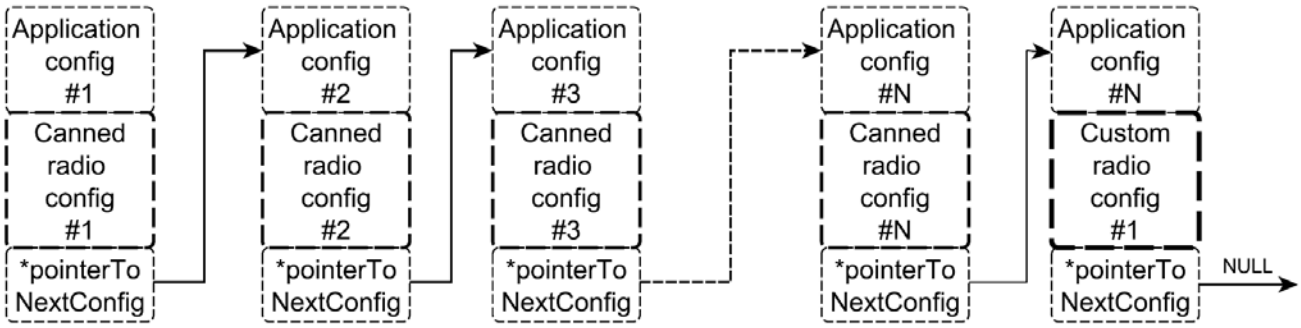


Figure 12. Predefined Configurations and the Custom Setting Provided by the WDS as a Linked List

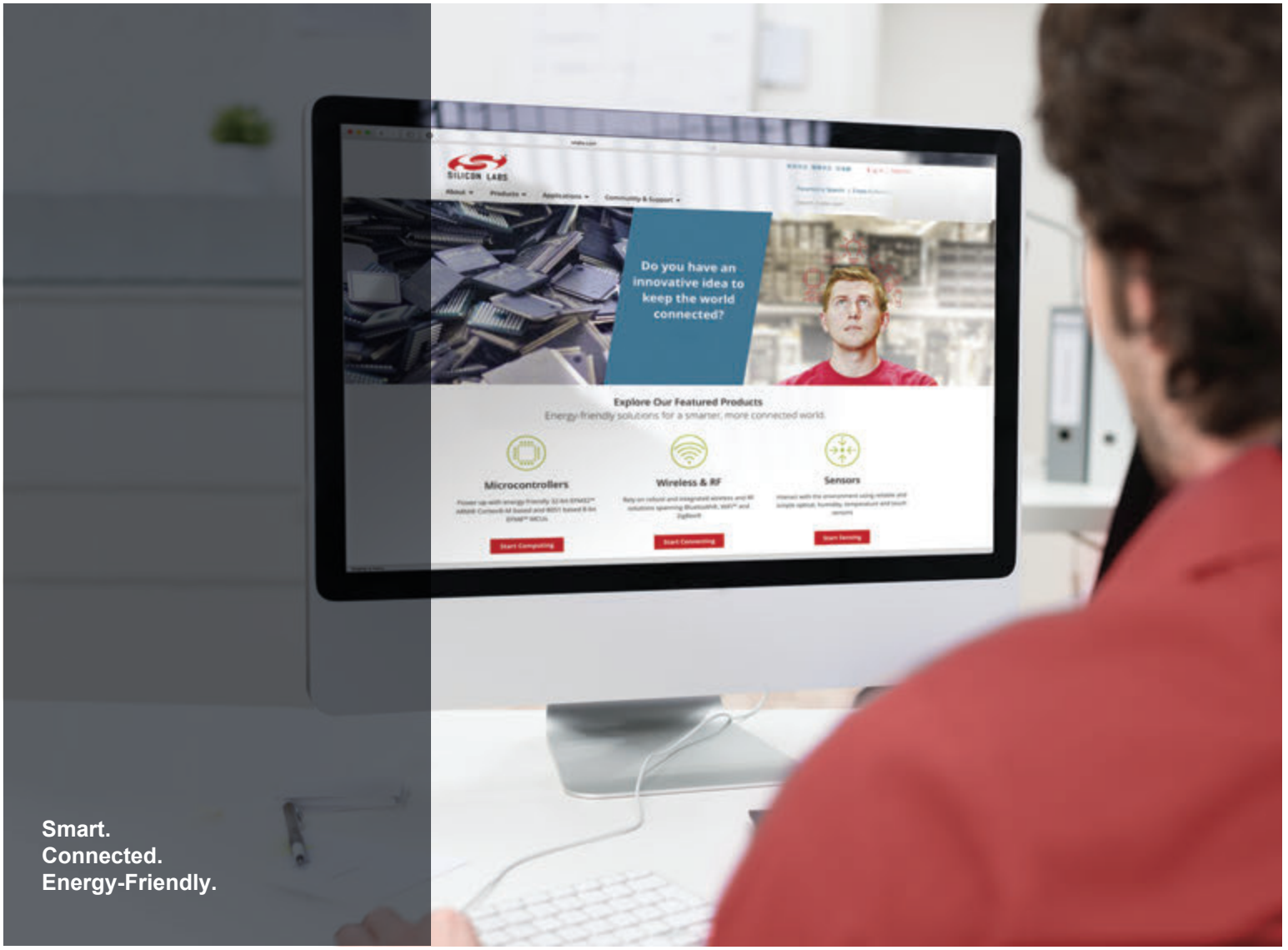
12. Test Modes

12.1. Bidirectional Communication

The range test can also be performed by means of two-way radio communication. Transmission can be initiated by selecting “TRX” in the “MODE” menu item. The originator transmits a ping packet for the other node. If it receives the packet correctly, it transmits back an acknowledgment packet. Each ping packet has a serial number increased by the originator after every packet transmission, which is transmitted back by the acknowledgment packet. If the originator receives the acknowledgment within a predefined timeout, then it considers the link functional; otherwise, it increases the number of missed packets by one. The originator also stores the number of transmitted ping packets so the demo can calculate the Packet Error Rate based on this information.

12.2. One-Way Communication

The range test can also be performed by means of one-way radio communication. Transmission can be initiated by selecting “TX” and “RX” in the “MODE” menu item. In this case, one end of the link must be set up as a transmitter (this will be the originator as described in the bidirectional link), and the other end of the link must be a receiver. The test needs to be started at the transmit side by pressing “TX ON”. The demo runs as long as the number of transmitted or received packets reaches the predefined number or until the demo is interrupted by Button 1. The user can follow the number of transmitted packets on the LCD screen. The one-way range test demo works the same as the bidirectional range test; however, 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 IDs.



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS®

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

http://www.silabs.com