AN1267: Radio Frequency Physical Layer Evaluation in *Bluetooth*® SDK v3.x and Higher

This application note provides an overview of how to perform Bluetooth-based radio frequency (RF) physical layer (PHY) evaluation with Bluetooth-enabled EFR32xG system-on-chips (SoCs) and BGM/MGM modules using Silicon Labs' software tools and dedicated firmware. A Bluetooth device's RF parameters are validated using a protocol called Direct Test Mode (DTM). DTM is described in the Bluetooth Core Specification versions 5.x, Volume 6, Part F.

KEY POINTS

- Basics of Direct Test Mode (DTM)
- Testing with NCP Commander
- Testing with DTM 2-wire firmware
- Test examples

Testing can be performed in three ways:

- Through DTM test commands issued by a host system over the SoC's or module's host interface.
- Through DTM test commands issued by a custom application running in the SoC or module itself. Such commands could be autonomously launched by the custom application, for example at boot or after a remote companion device writes a dedicated GATT characteristic over a Bluetooth LE connection.
- In a special test environment supported by dedicated firmware that allows a testing device to control the test target.

With these options, customers can fully evaluate transmit and receive performance, and test the RF functionality of their development kit hardware or custom hardware. Laboratory tests require RF test equipment, such as a spectrum analyzer and an RF signal generator, and/or a Bluetooth tester.

1. Basics of the Direct Test Mode (DTM)

The Bluetooth specification defines a mechanism, called Direct Test Mode (DTM), for testing the radio performance of Bluetooth low energy devices. This mechanism is described in the Bluetooth Core Specification, for example versions 4.2 or 5.2, Volume 6, Part F, which are available at https://www.bluetooth.com/specifications/bluetooth-core-specification. DTM is provided for the validation of a Bluetooth low energy device's radio-frequency (RF) physical layer (PHY), so as to ultimately guarantee an end product's interoperability and performance quality. RF testing is essential for a Bluetooth device, as with any device implementing a wireless standardized technology, since factors such as full compliance with the specification for interoperability, and conformance to communication regulations must be carefully assessed and validated before the product is launched. In addition, it may be desirable to evaluate the product's performance during production. The ability to easily accomplish RF testing in a standardized manner throughout the production cycle is useful.

DTM offers two approaches for RF PHY testing. In the first, an *Upper Tester* can enter special HCI (host control interface) commands over the standardized HCI interface of the Device Under Test (DUT) to start and stop the radio tests on the DUT. In the second, the Upper Tester has direct access to the DUT through a dedicated 2-wire connection and can autonomously start and stop the radio tests on the DUT in accordance with automated test routines.

The DTM protocol enabling the communication between the DUT and the Upper Tester also has provisions for feedback from the DUT, in the form of acknowledgements to the commands given, or in the form of Packet Count information being reported at the time a test is stopped.

A *Lower Tester*, or RF PHY Tester, is also part of the test setup, and is the actual lab equipment measuring the RF activity and performance. The RF PHY Tester can either be a separate device, like a spectrum analyzer, which is normally used with the first method where the HCI commands are issued by a generic host system, or it can be part of the same device. In the latter case the RF PHY Tester functions also as the Upper Tester, as for example commercial Bluetooth testers such as the one referenced later in this document.

The Bluetooth-enabled Silicon Labs EFR32xG SoCs and the BGM/MGM modules support both the approaches mentioned above. Special firmware can be loaded to enable the 2-wire DTM and allow the Upper Tester to take full control of the SoC or module (the device). The 2-wire link is a UART-like connection with no flow control operating at baud rates between 1200-115200, 8N1 (8 data bits, no parity, 1 stop bit). As one alternative, firmware that configures the device to operate in Network Co-Processor (NCP) mode can be used, since the host system can then issue the test commands included with the BGAPI Bluetooth API over the same host interface that is normally used to implement the BGAPI protocol for the control and configuration of the device's normal Bluetooth LE functionality. Note that in this case BGAPI-formatted test commands are sent and not the HCI commands defined in the standard. However, these BGAPI commands are then internally processed as HCI commands. The SDK does not contain any special firmware or configuration to disable the BGAPI DTM commands/responses and use the raw HCI command/responses instead.

.

2. Tests Enabled in the DTM Framework

DTM enables a set of RF PHY test cases, which are defined by the Bluetooth Special Interest Group (SIG) in the documents from the sections called “TCRL Release Table” and “Core - Test Requirements” found at the beginning of the https://www.bluetooth.com/specifications/qualification-test-requirements web page.

The Capability tests (as defined in the standard ISO subgroups) are organized in levels and groups representing protocol services, functional modules, and purposes, the latter being divided in operating conditions for the transmitter and the receiver. All the relevant RF PHY tests are in accordance to the test specifications RF-PHY.TS.5.1.1 or the updated RF-PHY.TS.p15 and are shown in the RF-P sheet of the Excel file called *Core.TCRL.2019-2.xlsx* found inside the *2019-2 TCRLs_2020-01-15_HFP1.8.zip*.

Below are examples of test cases for the Physical Layer Conformance. They are referred to by their identifiers, where RF-PHY stands for *RF-PHY Test Purpose* and TRM and RCV stand for Transmitter and Receiver test respectively.

- RF-PHY/TRM/BV-01-C [Output power]
- RF-PHY/TRM/BV-03-C [In-band emissions, uncoded data at 1 Ms/s]
- RF-PHY/TRM/BV-05-C [Modulation Characteristics, uncoded data at 1 Ms/s]
- RF-PHY/TRM/BV-06-C [Carrier frequency offset and drift, uncoded data at 1 Ms/s]
- RF-PHY/TRM/BV-08-C [In-band emissions at 2 Ms/s]
- RF-PHY/TRM/BV-10-C [Modulation Characteristics at 2 Ms/s]
- RF-PHY/TRM/BV-13-C [Modulation Characteristics, LE Coded (S=8)]

- RF-PHY/RCV/BV-01-C [Receiver sensitivity, uncoded data at 1 Ms/s]
- RF-PHY/RCV/BV-03-C [C/I and Receiver Selectivity Performance, uncoded data at 1 Ms/s]
- RF-PHY/RCV/BV-04-C [Blocking Performance, uncoded data at 1 Ms/s]
- RF-PHY/RCV/BV-05-C [Intermodulation Performance, uncoded data at 1 Ms/s]
- RF-PHY/RCV/BV-06-C [Maximum input signal level, uncoded data at 1 Ms/s]
- RF-PHY/RCV/BV-07-C [PER Report Integrity, uncoded data at 1 Ms/s]
- RF-PHY/RCV/BV-08-C [Receiver sensitivity at 2 Ms/s]
- RF-PHY/RCV/BV-10-C [Blocking performance at 2 Ms/s]
- RF-PHY/RCV/BV-27-C [Receiver sensitivity, LE Coded (S=8)]

All the above tests can be performed with the EFR32xG SoCs and the BGM/MGM modules, because they are supported by the DTM implementation built into the Silicon Labs Bluetooth stack.

Note: To accomplish the full set of tests in the specification, up to two external signal generators are needed to provide complete interference signals, in addition to a spectrum analyzer.

3. Testing with Bluetooth NCP Commander

An easy way to perform the RF PHY tests with the EFR32xG SoCs and the BGM/MGM modules is to use a PC running a Silicon Labs tool as the Upper Tester. The SoC or module must be configured to operate in NCP mode, and must have the **Bluetooth > Stack > DTM > Test** component installed to add the DTM commands.

Two tools are available:

- Bluetooth NCP Commander
- Bluetooth NCP Commander Standalone

AN1259: Using the v3.x Silicon Labs Bluetooth® Stack in Network Co-Processor Mode discusses the basics of the NCP firmware, how to load it to a device, and provides an example to help familiarize you with the NCP mode. It also includes instructions on how to get started controlling and configuring the Bluetooth functionality through Bluetooth NCP Commander and Bluetooth NCP Commander Standalone demo programs.

This chapter describes using Bluetooth NCP Commander or Bluetooth NCP Commander Standalone for RF PHY testing, where an SoC or module in a radio board attached to the Wireless Starter Kit (WSTK) is the example DUT, as shown in the following figure.

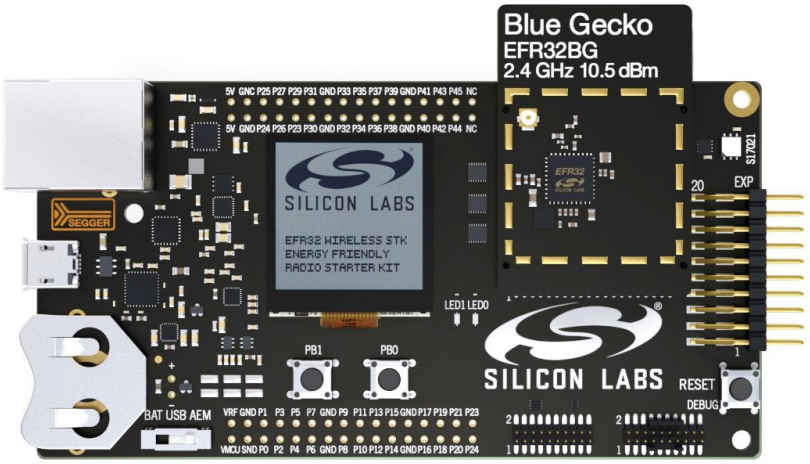
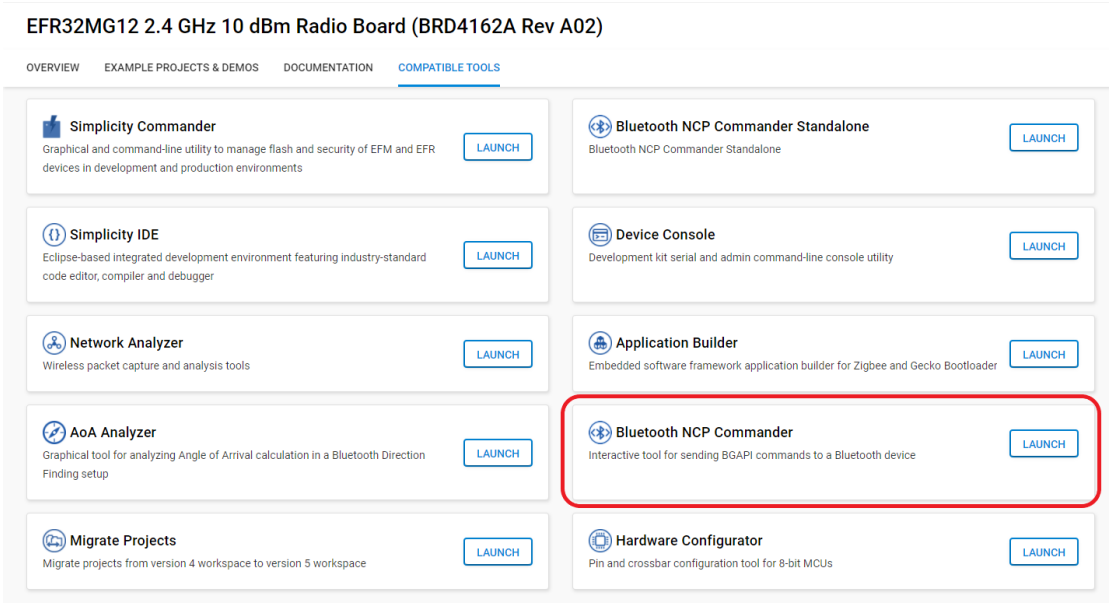


Figure 3.1. WSTK with the Blue Gecko EFR32BG SoC

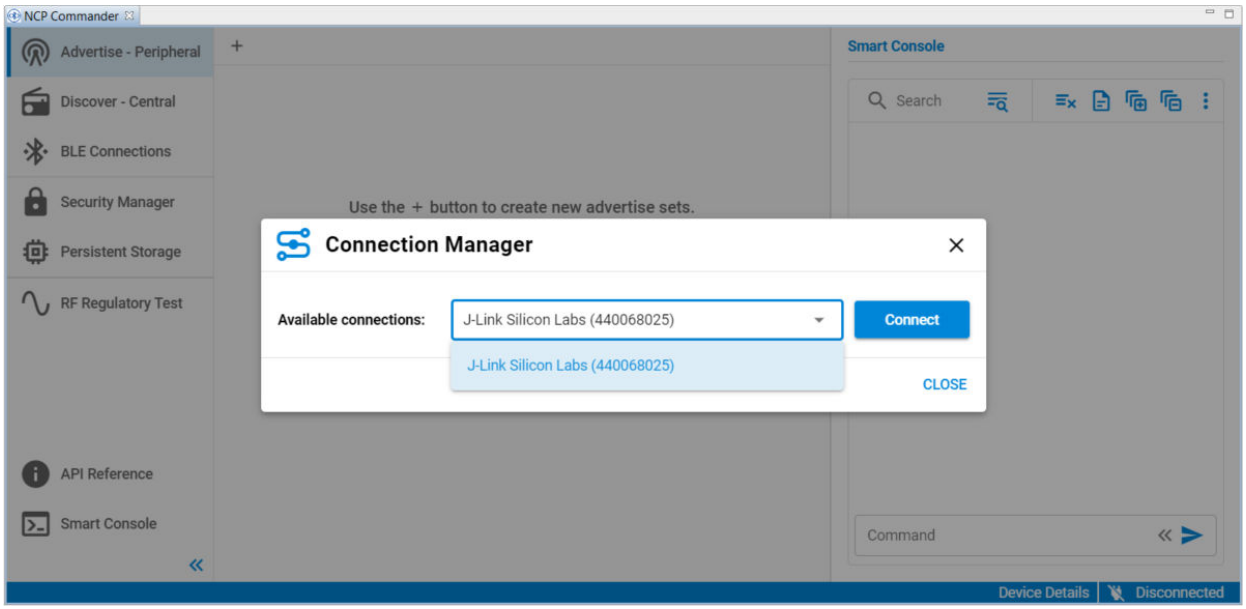
3.1 Using Bluetooth NCP Commander

Bluetooth NCP Commander is an easy-to-use tool that can be used for testing different stack features, by sending BGAPI commands to the target device. The tool has two versions: a version integrated in Simplicity Studio, which makes it easy to connect to your development kit and start testing, and a standalone version to test a board in an environment where Simplicity Studio cannot be installed, or if you want to test a custom board that can be accessed on UART interface, but not through a Simplicity Studio supported debug adapter using VCOM.

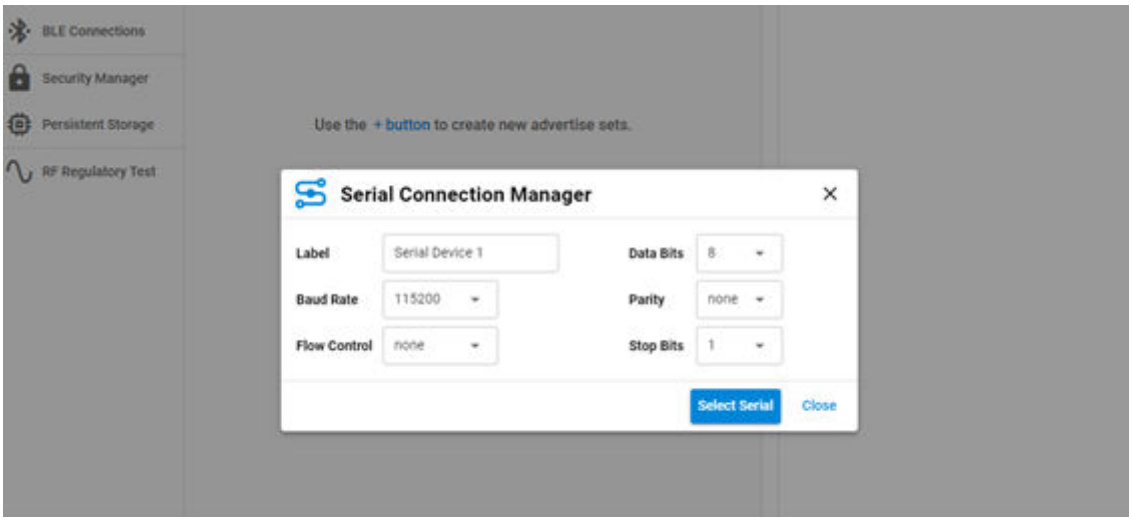
1. To open the integrated Bluetooth NCP Commander, select the target board in the **Debug Adapters** view, and check that the preferred SDK is set to **Gecko SDK Suite: Bluetooth**. Select the **Compatible Tools** tab, and click **[Launch]** next to Bluetooth NCP Commander.



2. To open the standalone tool, either navigate to C:\SiliconLabs\SimplicityStudio\v5\developer\adapter_packs\ncp_commander, and start NcpCommander.exe, or find the tool on the **Compatible Tools** tab or in the **Tools** menu.
3. If you use the integrated version, select the target device, and click **[Connect]**.



4. If you use the standalone tool, provide the UART interface settings, and then select the COM port on which the device can be accessed.



Next, change to the **RF regulatory Test** view. The control dialog shown in the following figure opens. Depending on the SDK in use, the NCP Commander might present different views and contents.

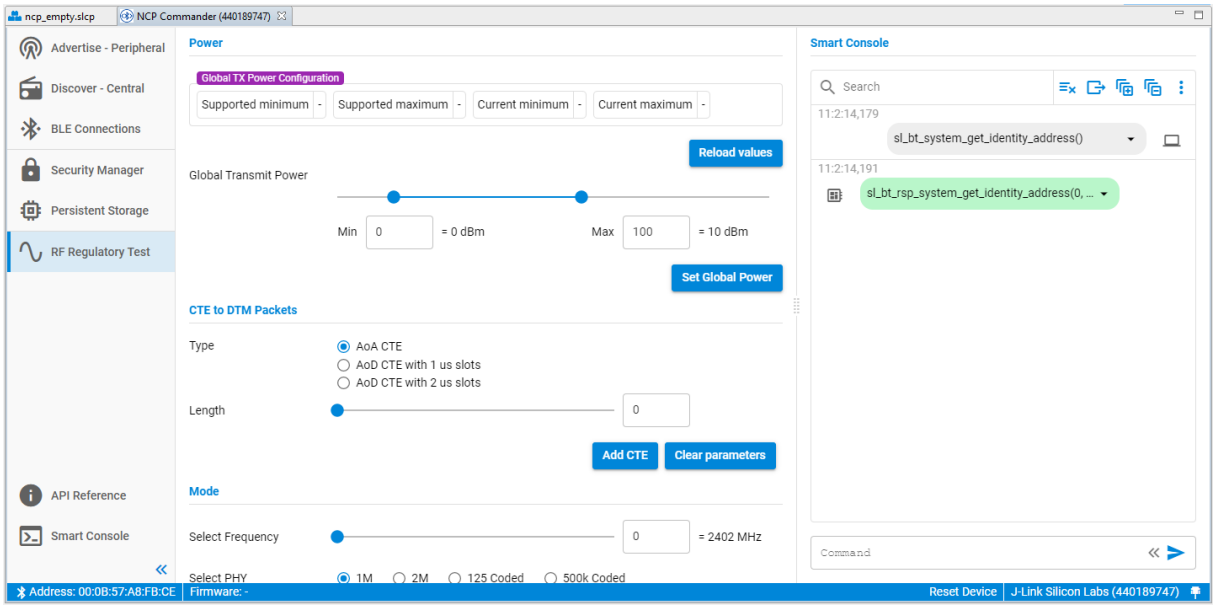


Figure 3.5. RF Regulatory Test Control Dialog

In this dialog you can interact with the device and set the RF test parameters using the sliders and the radio buttons. All the necessary configuration options are provided, namely transmission power level, operational frequency, packet type and length, and whether to perform a transmit or receive test.

If you intend to perform transmit tests using anything other than the default transmit power, change the power and click **Set** next to the Transmit power slider before launching the next test. Normally the default value is the maximum allowed by the firmware for a particular SoC or module. Clicking **Set** executes the BGAPI's `system_set_tx_power` command, which changes the power value.

When the configuration parameters for the test are complete, click **Start test**. The test runs until you click **Stop test**.

Depending on the SDK in use, Bluetooth NCP Commander might present different views and contents, such as in the following where no firmware uploading section exists.

The following figure shows the PHY selection option in NCP Commander, together with the option to select the PN9 continuously modulated carrier, and the possibility to extend the payload to up to 255 bytes:

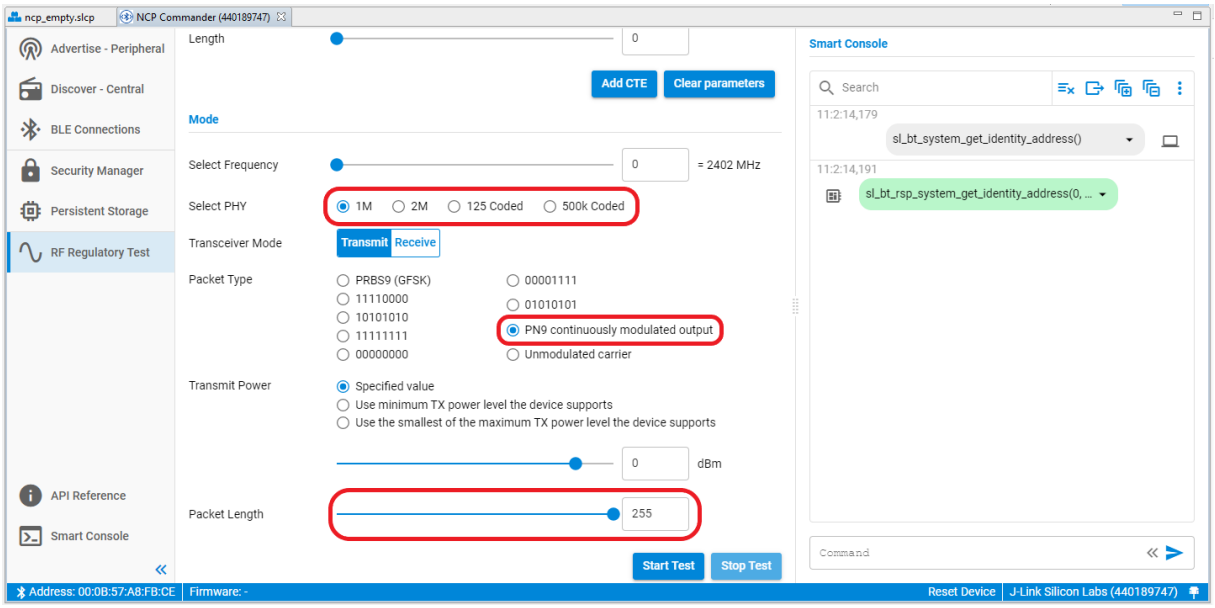


Figure 3.6. Selecting the PHY and Other Parameters with Bluetooth NCP Commander

The following figure shows an example of the commands and responses and events in an actual test:

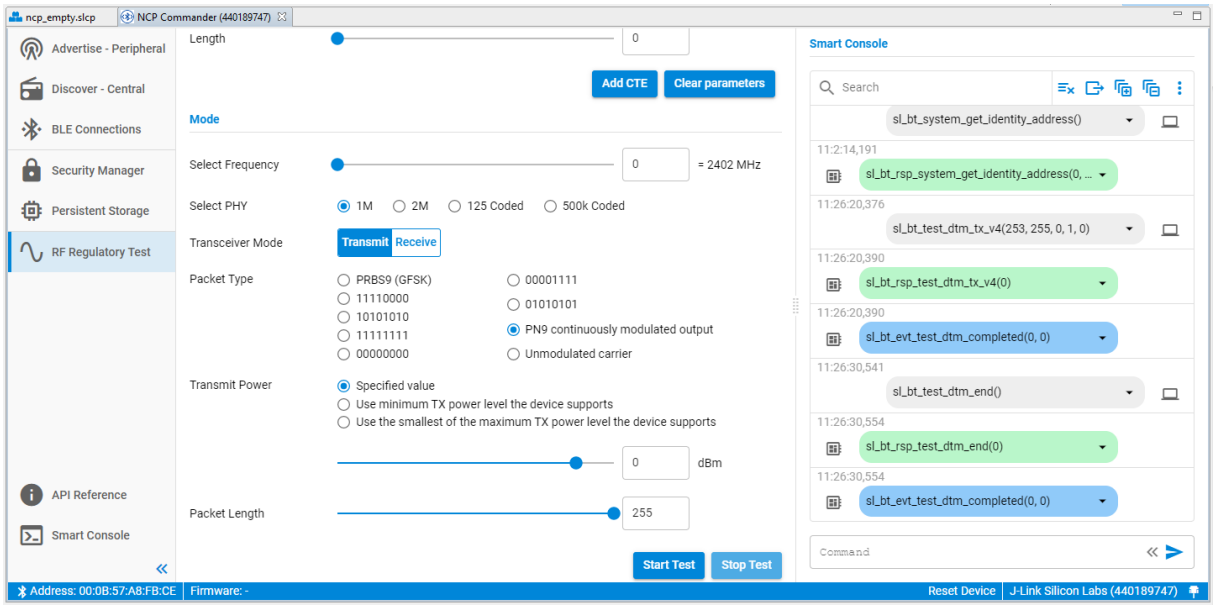


Figure 3.7. Starting and Stopping a Transmission Test

In this example, the test is started by clicking **Start test**. After a DTM test is started, the device will only accept the `test_dtm_end` command, which is entered by clicking **Stop test**. The only other way to stop a test are by resetting the device or through a power-cycle.

Note: Once a test is started the normal Bluetooth LE functionality is not available, in other words, advertising and scanning are not possible while a test is running, and similarly no connection can exist.

Another way to start and stop tests with NCP Commander is by entering the commands with the desired parameters in the command field of the Smart Console window, labeled **Command**. Such commands can be copy-pasted from the API reference, for example:

```
sl_bt_test_dtm_tx_v4(0,37,19,1,0)
```

See section 3.2 DTM Commands for more information.

3.2 DTM Commands

When you click **Start test** in either Bluetooth NCP Commander or BGTool, the tool sends one of the BGAPI's DTM commands over the host interface of the device. The command parameters correspond to the slider and radio button selections. The following summarizes the API's DTM commands. The commands with their related responses and the event mentioned below are found in the *Bluetooth Software API v3 Reference*.

`test_dtm_tx(packet_type, length, channel, phy)`

`test_dtm_tx_v4(packet_type, length, channel, phy, power_level)`

Starts a transmitter test. The DUT returns a response indicating that the command was received successfully. Shortly after this, a `test_dtm_completed` event is triggered, indicating that the command was processed by the radio and the actual test mode is started. At this point, the device is sending Bluetooth LE packets continuously at a fixed interval defined in the specification. The test is stopped using the `test_dtm_end` command, which is also followed by a `test_dtm_completed` event.

Only when a `test_dtm_completed` event follows the `test_dtm_end` command, the event's `number_of_packets` field carries the actual number of packets sent during the test.

The type and length of each packet is set by the `packet_type` and `length` parameters. The newest firmware versions add a parameter called `phy` that allows the selection of the PHY among 1M, 2M, 125k Coded, and 500k Coded, when supported by the DUT. The power_level parameter is the TX power level in dBm, with a range of -127 to +20. This parameter also includes options to use the minimum TX power level that the device supports, or the smallest of the maximum TX power level that the device supports and the global maximum TX power setting in the stack.

Note that a special packet type named `test_pkt_carrier` exists that can be used to transmit a continuous unmodulated carrier. The length field is ignored in this mode.

Another special packet type named `test_pkt_pn9` can be used to transmit a continuous modulated carrier instead, by means of a PN9 stream offering a 100% duty cycle.

In general, for the regulatory testing PRBS9 and/or the continuous modulated carrier are used, whereas the other two packet payloads are used when testing for the Bluetooth qualification.

`test_dtm_rx(channel, phy)`

Starts a receiver test. The procedure is similar to the transmitter test described above. When ending the test with the `test_dtm_end` command, the expected `test_dtm_completed` event from the device carries in its `number_of_packets` field the actual number of packets received during the test.

`test_dtm_end()`

Can be issued at any time to end a transmitter or a receiver test. When the command is processed by the radio and the test has ended, a `test_dtm_completed` event is triggered.

3.3 Using Other Methods

Using either Bluetooth NCP Commander or Bluetooth NCP Commander Standalone is just one of the ways to perform the RF PHY tests in the lab. In fact, the DTM commands from the BGAPI protocol are also available to any host MCU capable of issuing the BGAPI commands over the device's host interface, for example when implementing a host program based on the BGLib library.

The commands can also be launched by a DUT operating in standalone mode, by means of a custom C program loaded onto the SoC/module. Users could start with the **SOC-Empty** example project, and customize it to simply have a test command autonomously launched at boot and terminated by a hard reset or after a timer has expired. Alternatively, the program could be designed to allow an incoming Bluetooth LE connection over which a remote device could write a dedicated GATT characteristic. The characteristic's content would define the test command the application should launch and how long this should run until a terminating event such as allowing the next connection or a reset.

Using this programming strategy, customers often prepare multiple firmware images for their test house, each containing the standalone functionality to launch a specific test at bootup, and then simply provide instructions to the test house on how to change firmware images.

Notes and Limitations:

1. The DTM tests discussed in this section are meant for regulatory testing as well as for Bluetooth qualification. All tests are designed to satisfy the guidelines normally set by the test houses that will conduct the regulatory evaluation.

2. During a 1M PHY transmit test where the typical 37 bytes of payload are used, subsequent packet transmissions are started at an interval of 625 µs. In this case, the packet transmission itself lasts 376 µs, resulting in a duty cycle of 60.2%. This comes from the fact that the packet used in the test is made of a preamble (8 bit) plus a sync word (32 bit) plus a packet type field (16 bit) plus the payload (296 bit) plus the CRC (24 bit), while the time to transmit one bit is 1 µs given the air-interface baud rate of 1 Mbit/s.

   The max payload size that can be configured with the `test_dtm_tx_v4` command can be up to 255 bytes, according to the LE Data Packet Length Extension introduced in the Core Specification version 4.2. Given all the new possible combinations of PHYs and packet lengths in the newest firmware versions, the duty cycle will actually depend now also on the interval between packets which is not fixed but calculated based on the information found in the Core Specification version 4.2 and 5.x, in chapter "*4.1.6 LE Test Packet Interval*" of Volume 6, Part F.
3. As of this document version, no configuration option exists to define a fixed number of packets that the device would transmit after issuing the `test_dtm_tx_v4` command. One useful use case would be to estimate packets lost: the test setup would have a transmitter sending a number of packets known in advance and a receiver reporting the number of packets received (via the `test_dtm_completed` event).

   This could evaluate performance changes due to distance or other conditions, or be used during production to validate the quality of a product just manufactured. The workaround for the lack of such a configuration option is to let the unit transmit only for a fixed amount of time, calculated by taking into account the interval at which the packets are transmitted: in the example case of a 1M PHY and 37 bytes payload, having the interval between packets set to 625 µs, the packet rate is 1600 [packets/s].

   Note: Currently the `test_dtm_completed` event reports the actual packets sent during the test when the test mode launched with the `test_dtm_tx_v4` command is stopped. This is useful for a precise estimation of the PER when comparing the exact number of packets sent by the stack with the actual packets received by the receiver. The DTM communication response in 2-wire firmware does not report the sent packet count.

4. Testing with the DTM 2-Wire Firmware

If a Bluetooth tester device is available, it should be used to evaluate the RF performance of the DUT, since such dedicated equipment is faster and offers automated and accurate testing. These benefits among others are achieved because the tester can control and configure the DUT in all the tests required for complete RF PHY evaluation. Such control is possible thanks to the DTM 2-wire capability and its related protocol, which are part of the Bluetooth specification.

In order to enable the DTM 2-wire communication between a commercial Bluetooth tester device (that is, the Upper Tester with the included RF PHY measurement capability) and the DUT, the latter needs to run a special firmware where the protocol is included. This special DTM 2-wire-capable firmware is available as an example application project in Bluetooth SDK v3x through Simplicity Studio® 5 (SSv5). This section provides a summary of the project creation and use process. It assumes you are familiar with building and flashing applications using the Bluetooth SDK v3.x with SSv5. For more information about these processes, see the online SSv5 User's Guide, available through the SSv5 help menu, or *QSG169: Bluetooth® SDK v3.x Quick Start Guide*, installed with the SDK

To create the project in SSv5, in the Launcher-perspective, select the correct DUT in the Debug Adapter view. This correctly prepopulates the Target Board and Target Device (SoC or module) settings. In the File menu, select **New > Simplicity Studio Project Wizard**. The Target, SDK and Toolchain Selection dialog opens. Verify the target hardware, SDK version, and toolchain are correct. Click [**NEXT**].

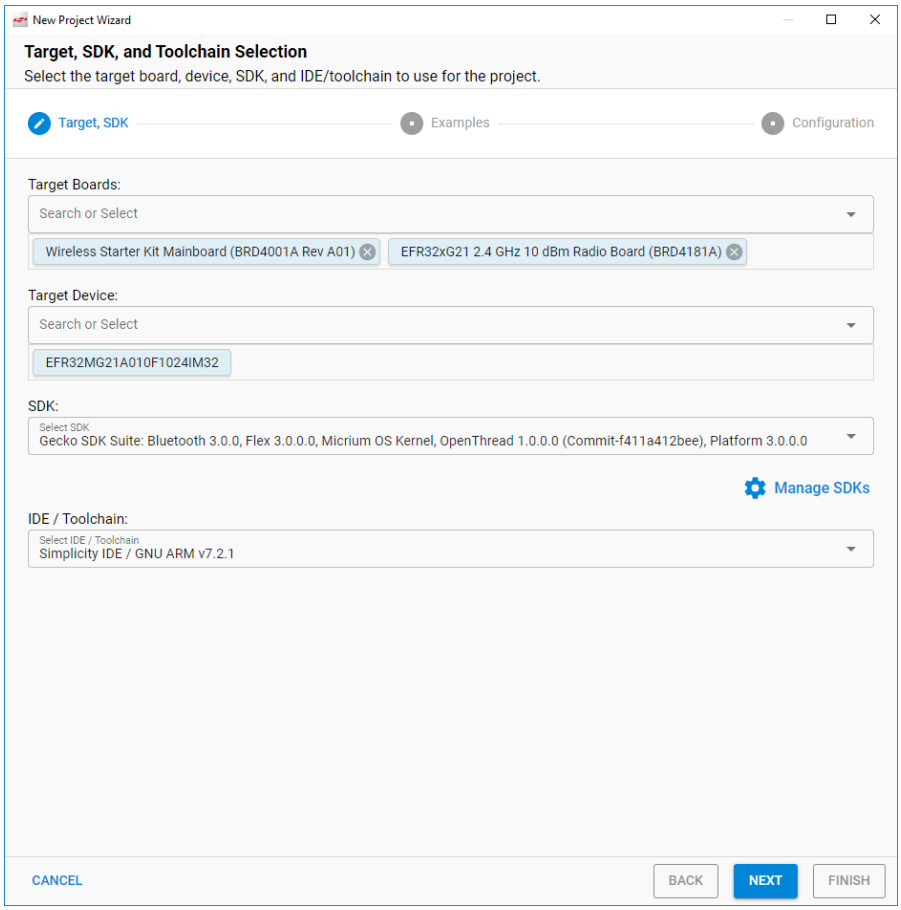


Figure 4.1. Simplicity Studio 5 Target, SDK and Toolchain Selection Dialog

The Example Project Selection dialog opens. Use the Technology Type and Keyword filters to search for a specific example, in this case **Bluetooth - SoC DTM**. Select it and click [**NEXT**].

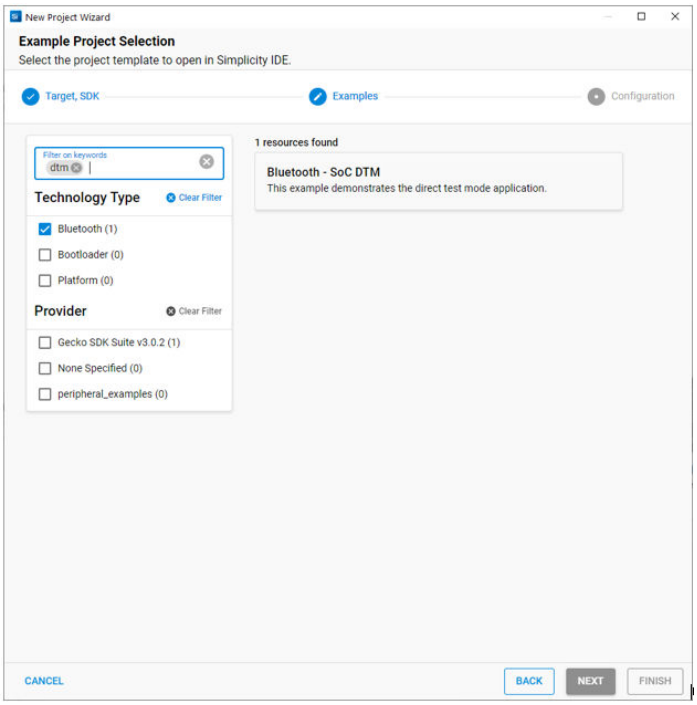


Figure 4.2. Simplicity Studio 5 Example Project Selection Dialog

The Project Configuration dialog opens. You do not need to change any of the default values. Click [**FINISH**] to create the DTM example project.

The example application configuration is as follows:

- The hardware link to the Bluetooth tester is enabled over the device's pins that, in its evaluation radio board, are mapped to the WSTK expansion header's UART TX (pin 12) and UART RX (pin14).
- The link uses UART parameters of 115200, 8N1 with no hardware flow control.

You can use the example application as is if these settings are correct for your DUT. Refer to the documentation of a particular SoC or module and of its evaluation radio board to determine which pins are in use in order to have the DTM signaling routed to the intended expansion header pins. See 4.3 Customizing the SoC DTM Application for more information on application customization.

After changing the configuration (if necessary), build the application image and flash it to the DUT. If the application is not working and cannot even be debugged, the bootloader might be missing from the DUT. The easiest way to program the bootloader is to flash a pre-compiled demo application (with bootloader) such as Empty SoC from the Launcher-perspective DEMOS-tab, then re-flash the Soc Dtm image.
.

4.1 Connecting and Testing with the Bluetooth tester

With the DTM firmware installed, the DUT is ready to be wired to the Bluetooth tester, as shown in the following figure.

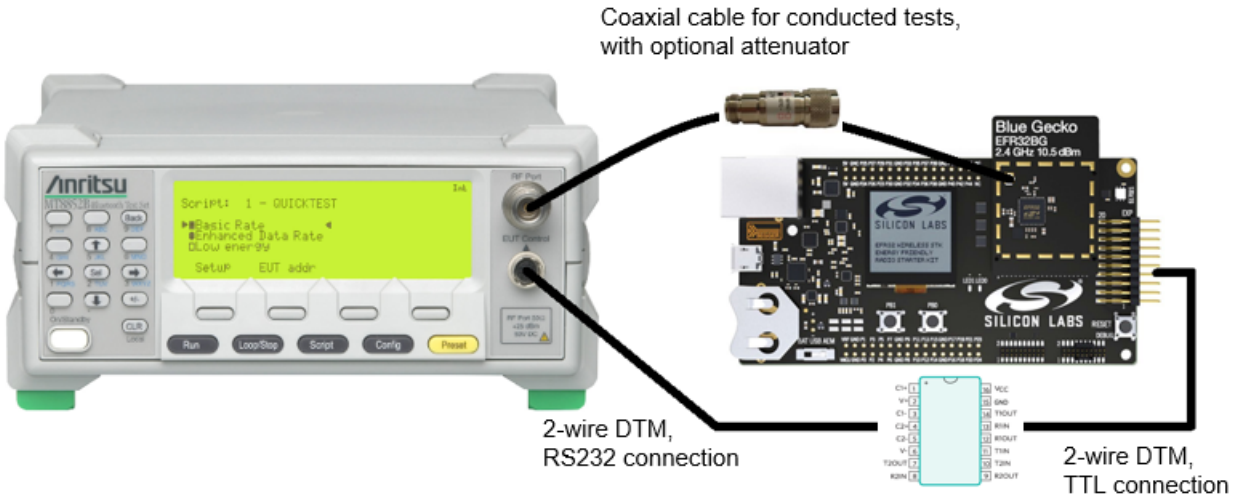


Figure 4.3. Connecting the Bluetooth Tester to the WSTK

Testing can be performed at the Upper Tester interface, normally through PC software, as shown in the example using Anritsu MT8852B as the Bluetooth tester.

First, establish the connection to the DUT with the correct parameters, as highlighted in the following figure.

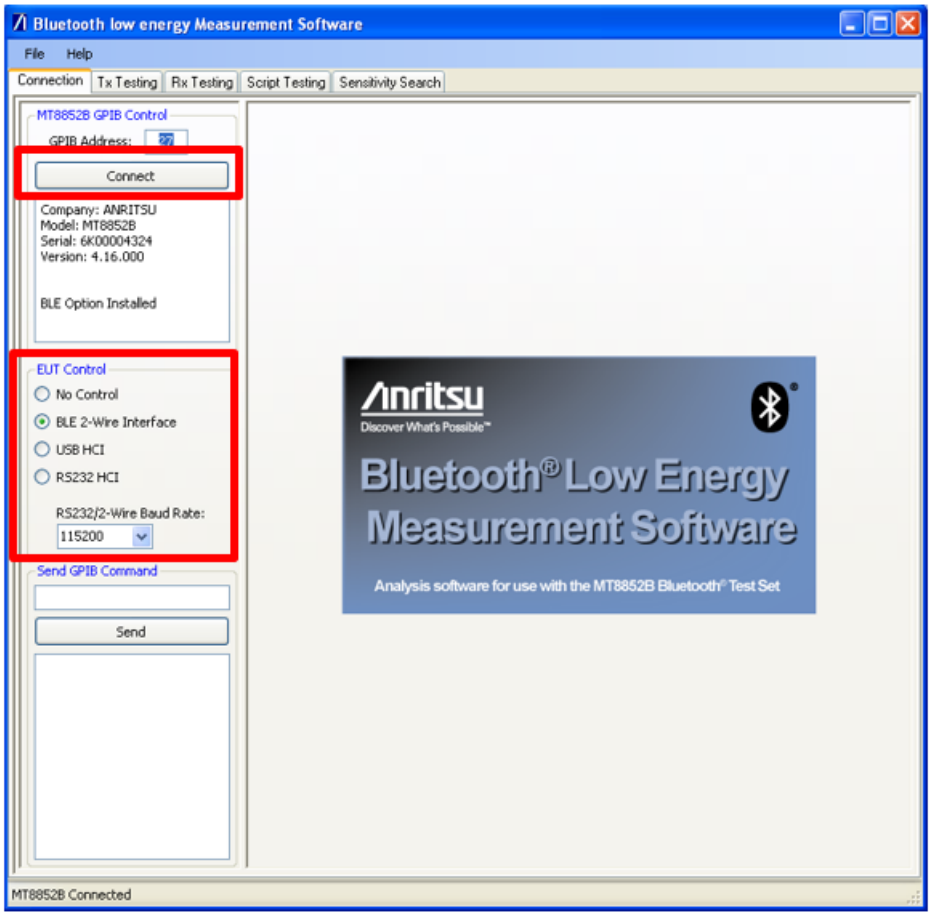


Figure 4.4. Connecting the Upper Tester to the DUT

Next, pre-existing or custom scripts can be used to start a suite of tests, as shown in the following figure. In the same figure the test report is displayed by the tester PC program, after the automated tests have completed, with an overall result of “Passed”. Notice also

the **Fixed Offset** field in the **Script Setup** box, which can be used to compensate for power loss, for example over the coaxial antenna cable in the case of conducted RF tests.

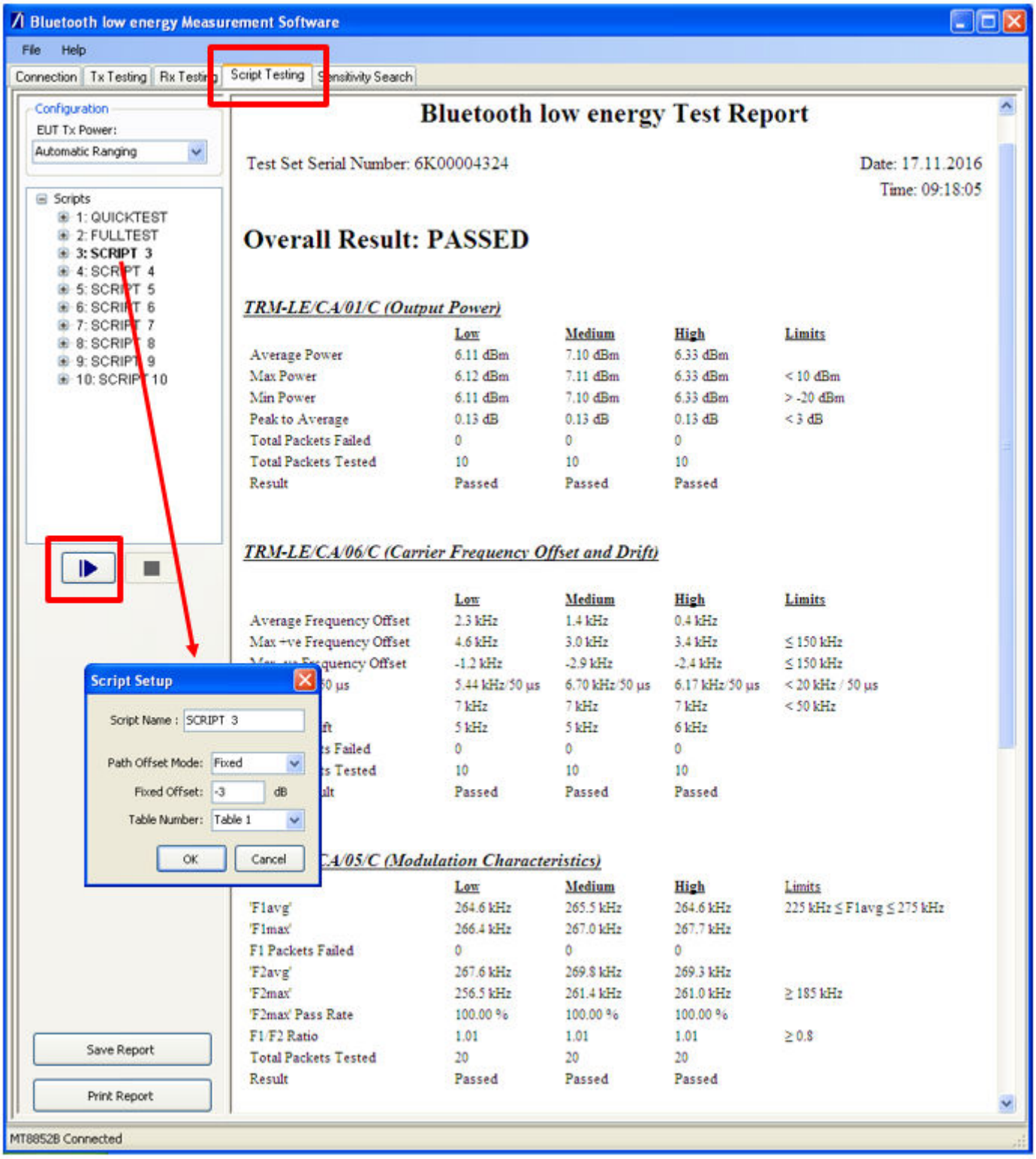


Figure 4.5. Launching a Suite of Tests / Test Results

Any commercial Bluetooth tester should be capable of testing among others the Packet Error Rate (PER). In this test configuration, the Upper Tester first configures the DUT into receiving mode using the appropriate DTM 2-wire command (step 2 in the figure below after the test configuration by the user in step 1), and then the Upper Tester starts sending a defined number of packets at the RF TX power level configured by the user (step 3). When the test is ended by the user (step 4), the corresponding DTM 2-wire command is also sent to the DUT and, much as in the BGAPI `test_dtm_completed` event discussed earlier, the DUT reports over the 2-wire link the number of received packets so that the Upper Tester can calculate the PER, all according to the specification.

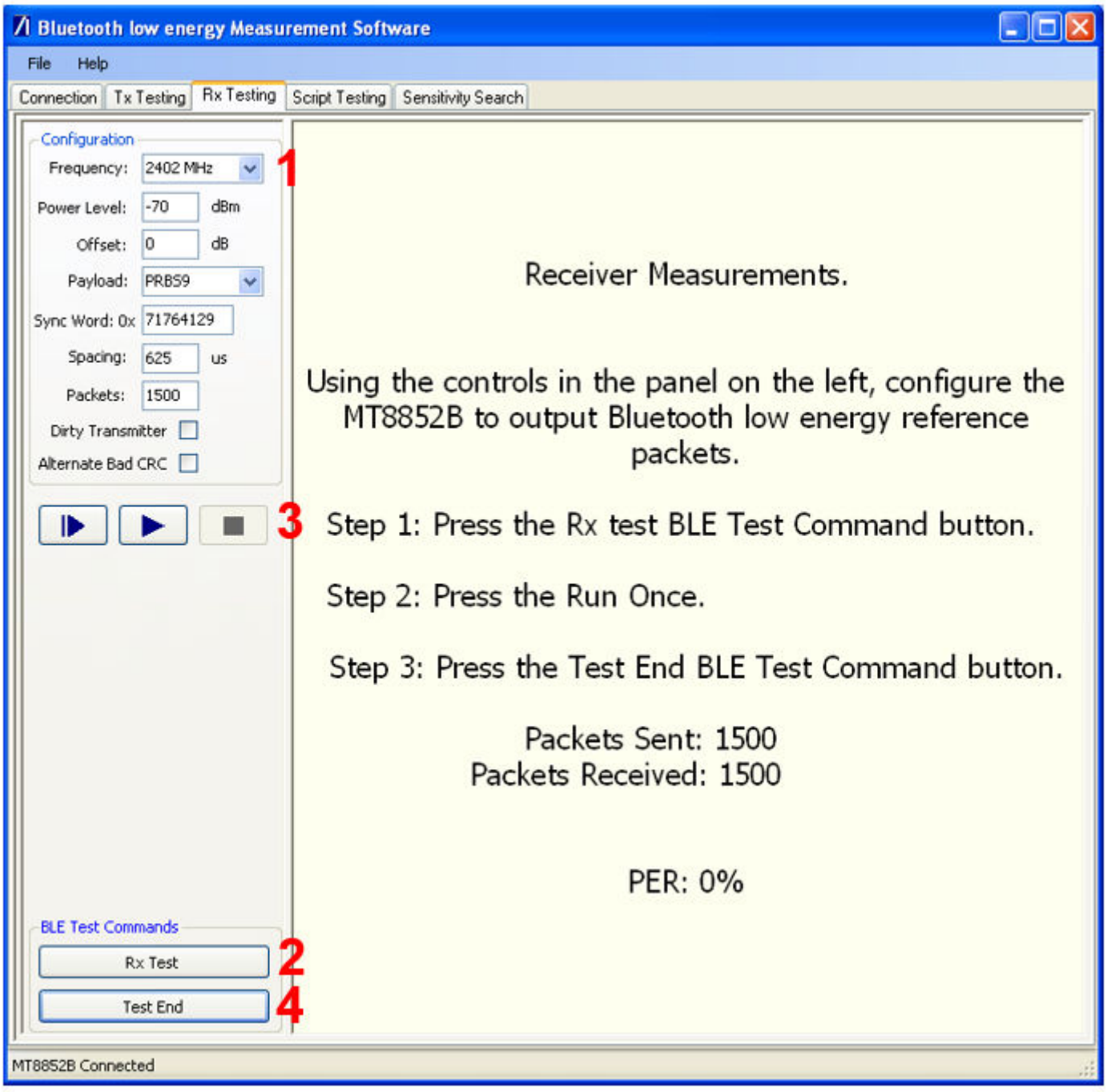


Figure 4.6. Receiver Measurements (PER)

4.2 Testing Equipment Considerations

Certain Bluetooth testers, like the Anritsu MT8852B mentioned in this application note, use RS232 interfaces to connect to a DUT for the DTM 2-wire, meaning that, when testing with the DTM firmware and a WSTK, for example, a level shifter is needed in order to interface to the device's TTL logic.

Certain Bluetooth testers have no capability to generate RF signals with a power lower than a certain value, for example -90 dBm in the case of the Anritsu MT8852B mentioned in this application note. Given that modern SoCs/modules have a receiver sensitivity going well below that, an attenuator might be required for appropriate testing, to further decrease the power of a signal directed to the device in receiving test mode.

4.3 Customizing the SoC DTM Application

Configuring the device to use other pins and UART settings for the 2-wire UART than the defaults is done by configuring the Software Component.

When you create a new project, the GATT Configurator is opened by default. To go to the Project Configurator, click the <project>.slcp tab, or double-click the <project>.slcp file in the Simplicity IDE Project Explorer view. Click the Software Components tab. Check the Configurable Components and Installed Components filters. Type "stream" in the search box. You should see only one IO Stream implementation named **exp**. Select **exp**.

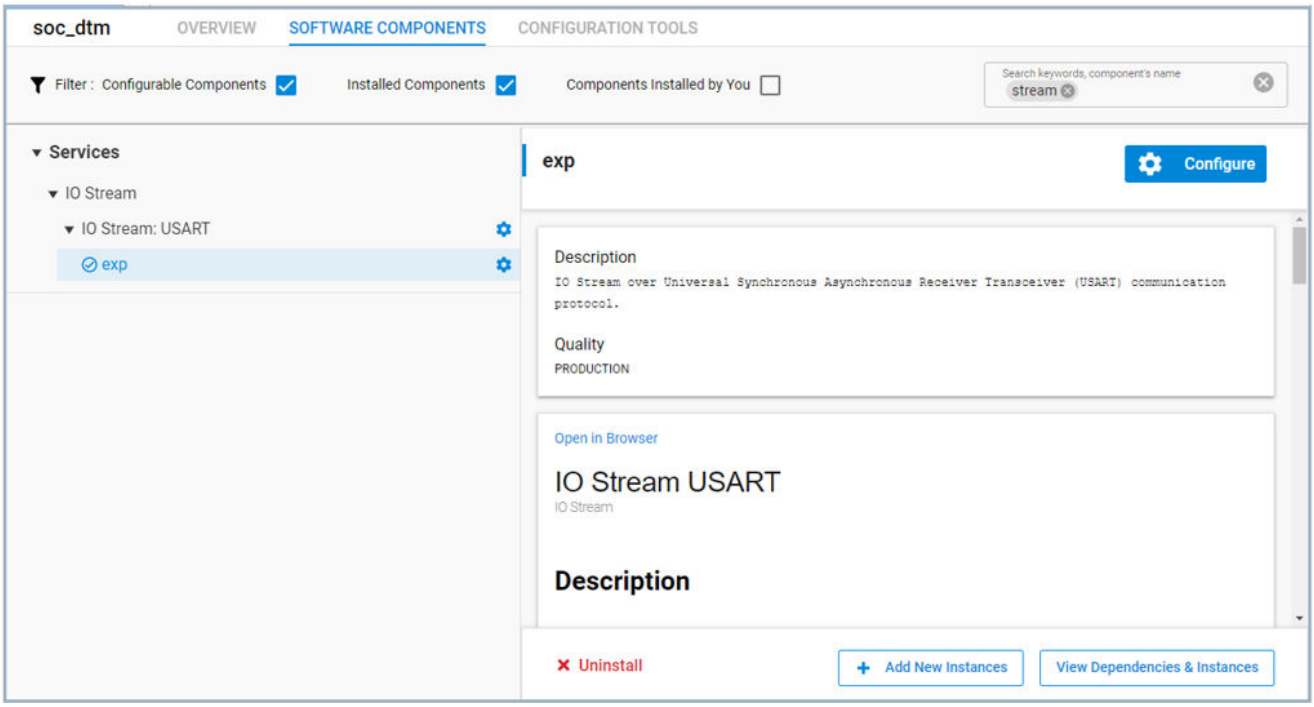


Figure 4.7. IO Stream Component Using UART

Click [**Configure**] to open the Component Editor. Here you can configure the baud rate, pins, and other parameters. Make your change and close the Component Editor. Changes are autosaved.

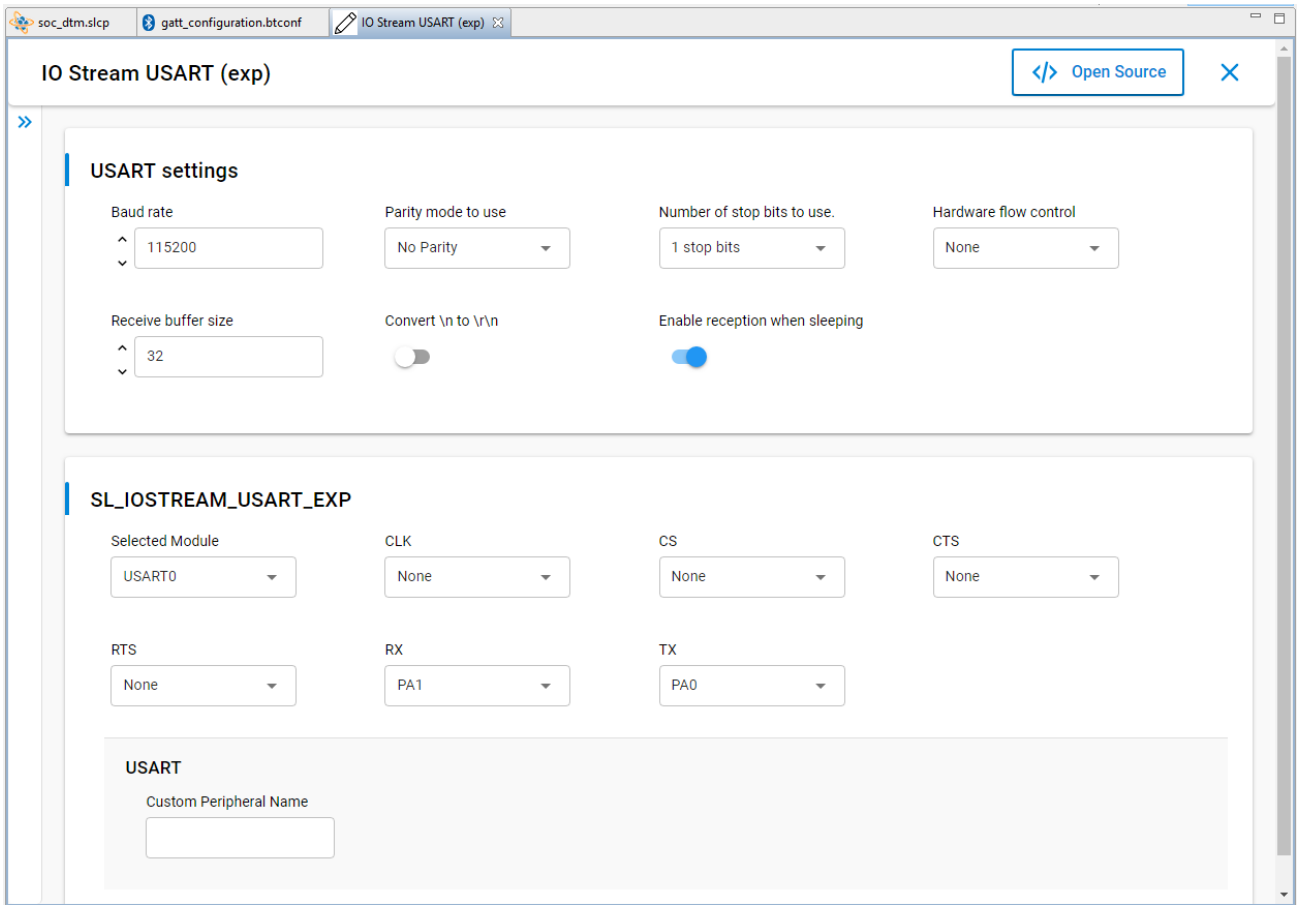


Figure 4.8. IO Stream UART and Pin Settings

Instead of configuring the component, you can change configurations in the Pin Tool. Double click the file <project name>.pintool to open it.

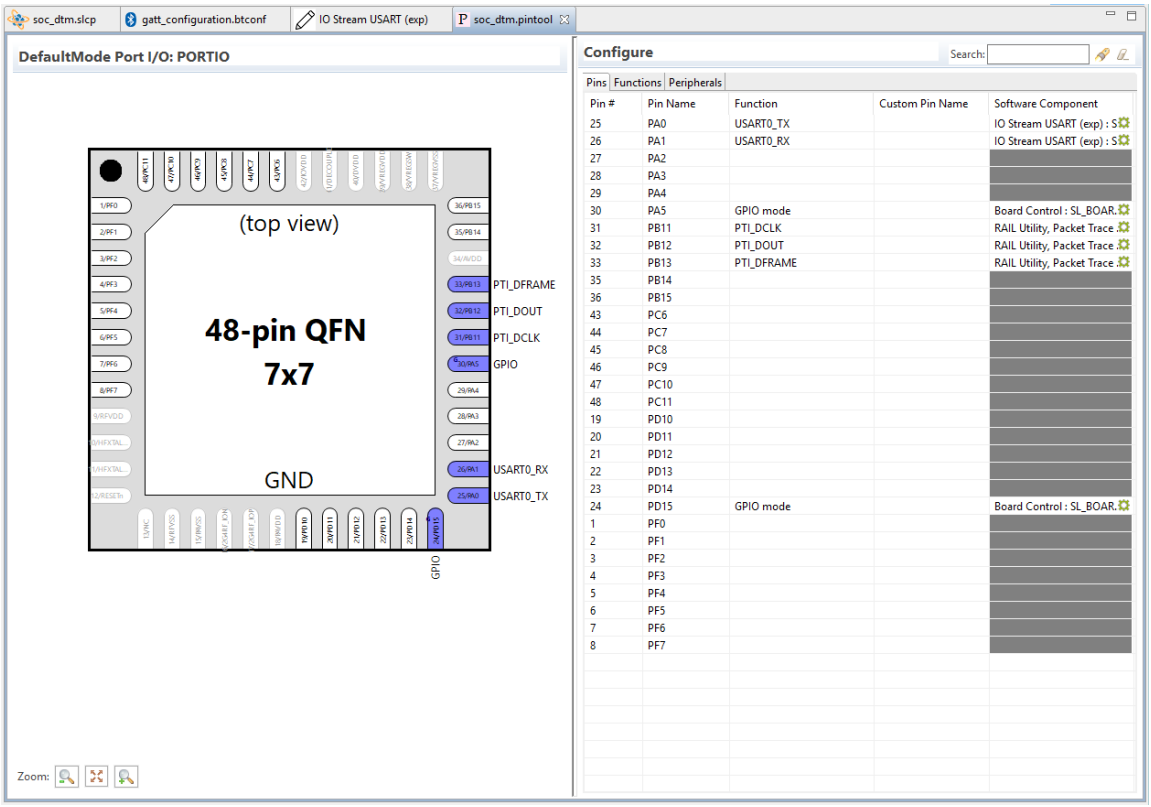
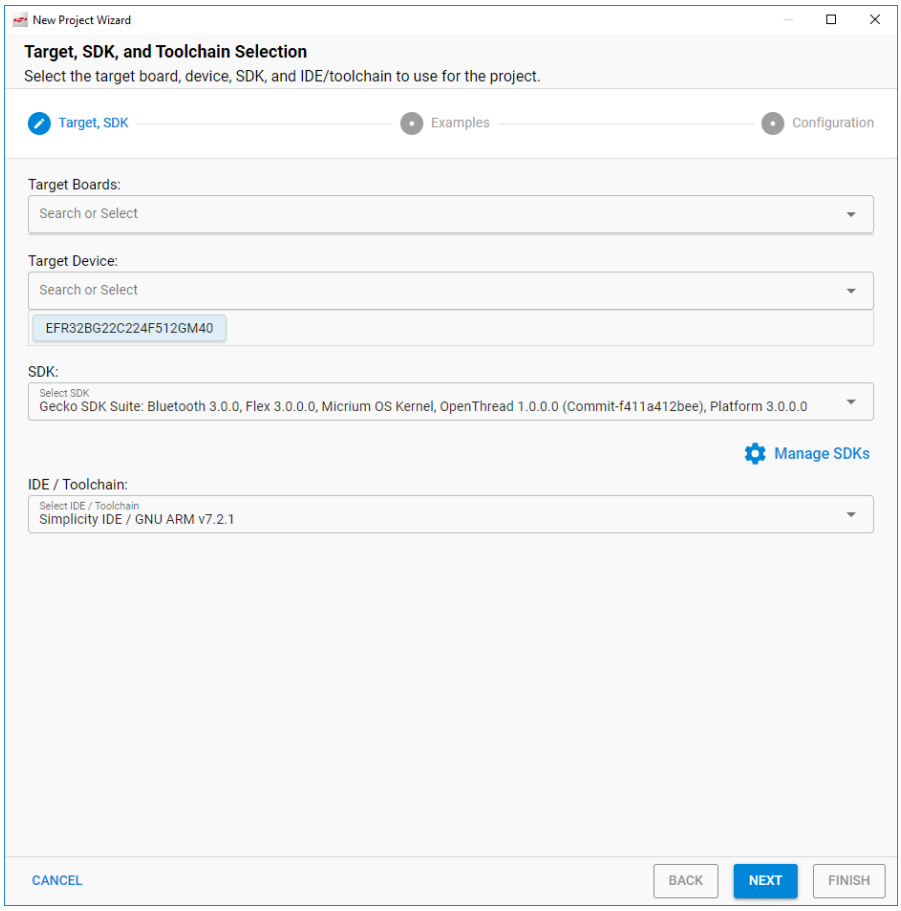


Figure 4.9. Pin Tool

You can change the pin position from the Configure editor Pins tab. Select a pin, click the fuction to display the Function dropdown list, and select a function. Click the Software Component symbol to open the Component Editor. Changes made in the Pin Tool are not autosaved.

Once your changes are complete, compile and flash to test them.

In the Simplicity Studio screenshot shown in Figure 4.1 Simplicity Studio 5 Target, SDK and Toolchain Selection Dialog on page 10, a WSTK with a radio board carrying an EFR32BG21 was attached to the PC. More often a custom design is used instead, and this custom board is obviously not recognized by the Silicon Labs' software. In this case, start the DTM firmware creation as described in the beginning of the chapter but select the correct Target Device in the Target, SDK and Toolchain Selection dialog. The Target Board is left empty with a custom design. Check the SDK as it might not be correct by default and also check the Toolchain.



From there the process continues as previously described. Most likely the UART pins will need some configuration, as described above.

5. Test Examples with the BGAPI Commands test_dtm_tx_v4 and test_dtm_rx

5.1 Output Power—RF-PHY/TRM/BV-01-C

The power basic measurement is used to make all of the output power measurements.

In this example, the `test_dtm_tx_v4` command is used.

In this test mode, only the frequency and the output power are configurable parameters.

A spectrum analyzer is used to measure the power of the radio and must be connected through a compatible RF cable to the RF connector of the radio board.

It is also possible to test the output power with a separate Bluetooth tester device when the DTM firmware is installed in the DUT.

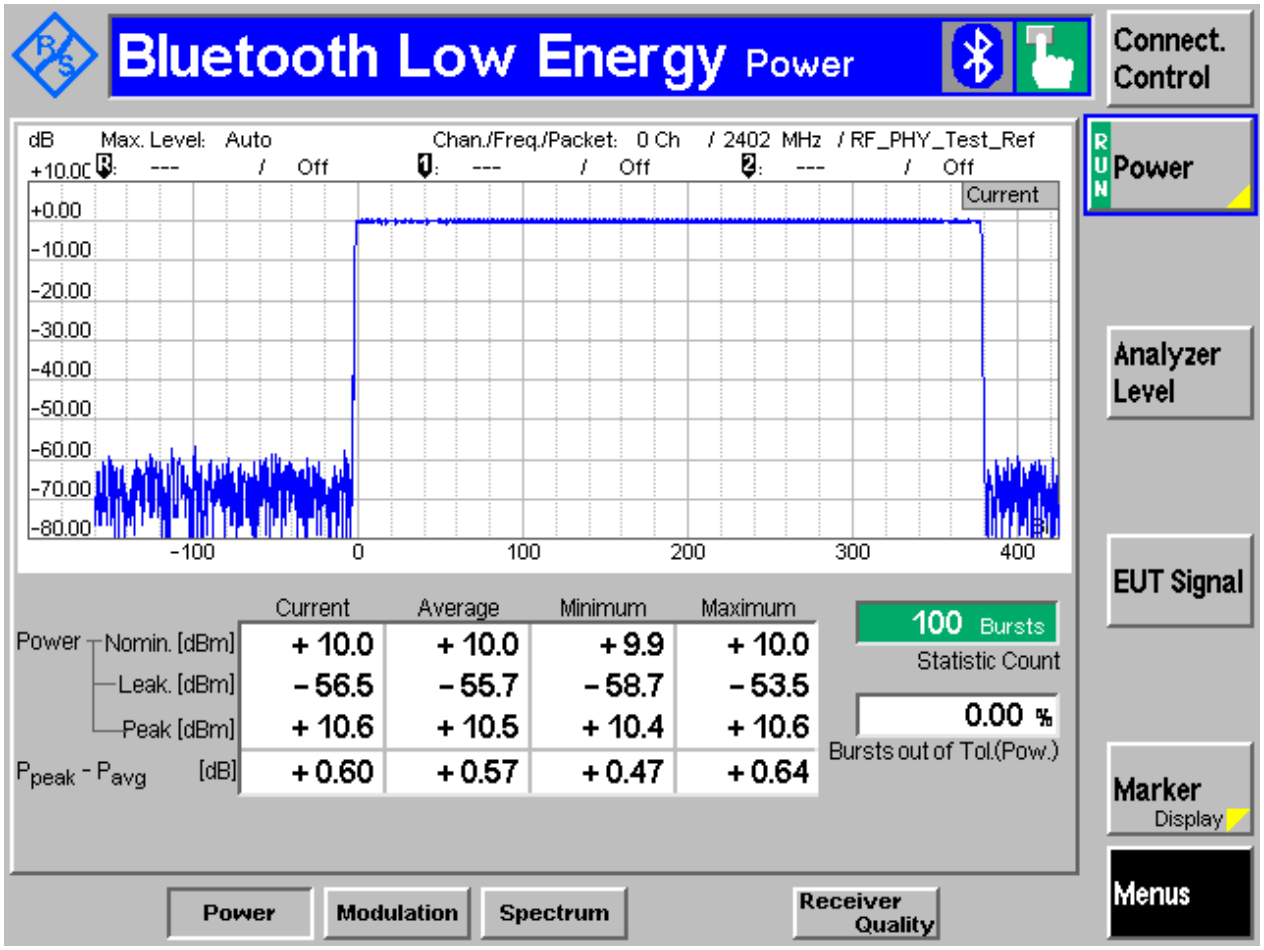


Figure 5.1. Output Power Measurement with the Rohde & Schwarz CBT Bluetooth Tester

5.2 Modulation—RF-PHY/TRM/BV-05-C and RF-PHY/TRM/BV-06-C

Used to make the modulation, and frequency offset and drift measurements.

In this example, the `test_dtm_tx_v4` command is used.

To follow the test specification, you need to be able to change your payload data pattern (either 1010 pattern or 11110000 pattern).

A spectrum analyzer is used to measure the modulation, frequency offset, and drift measurements of the radio and must be connected through a compatible RF cable to the RF connector of the radio board.

It is also possible to test by using a separate Bluetooth tester device provided that the DTM firmware is installed in the DUT.

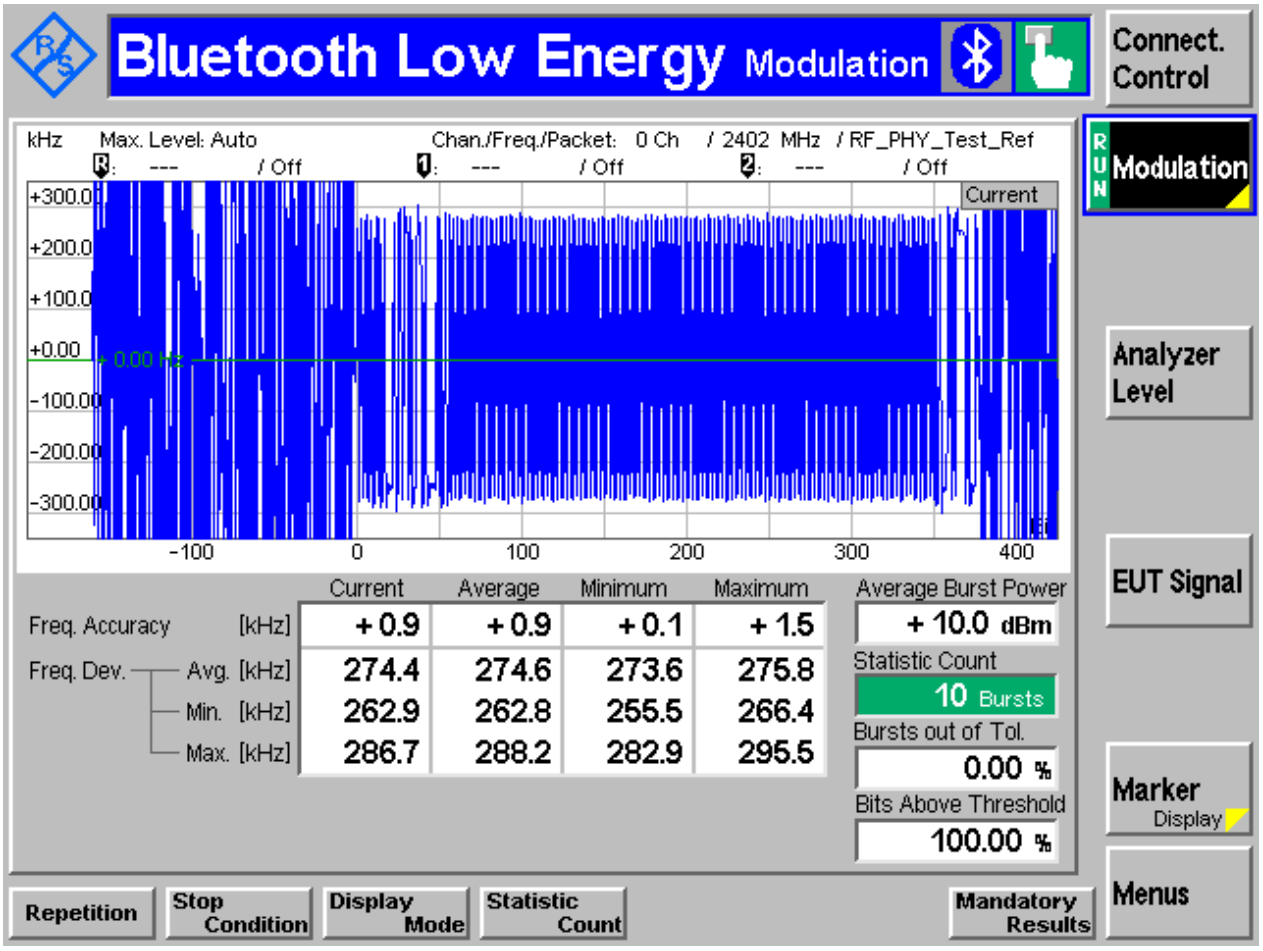


Figure 5.2. Modulation Characteristics

5.3 Spectrum—RF-PHY/TRM/BV-03-C

Used for in-band emissions measurements.

In this example, the `test_dtm_tx_v4` command is used.

A spectrum analyzer is used to measure the spectrum of the radio and must be connected through a compatible RF cable to the RF connector of the radio board.

It is also possible to use a separate Bluetooth tester device provided that the DTM firmware is installed in the DUT.

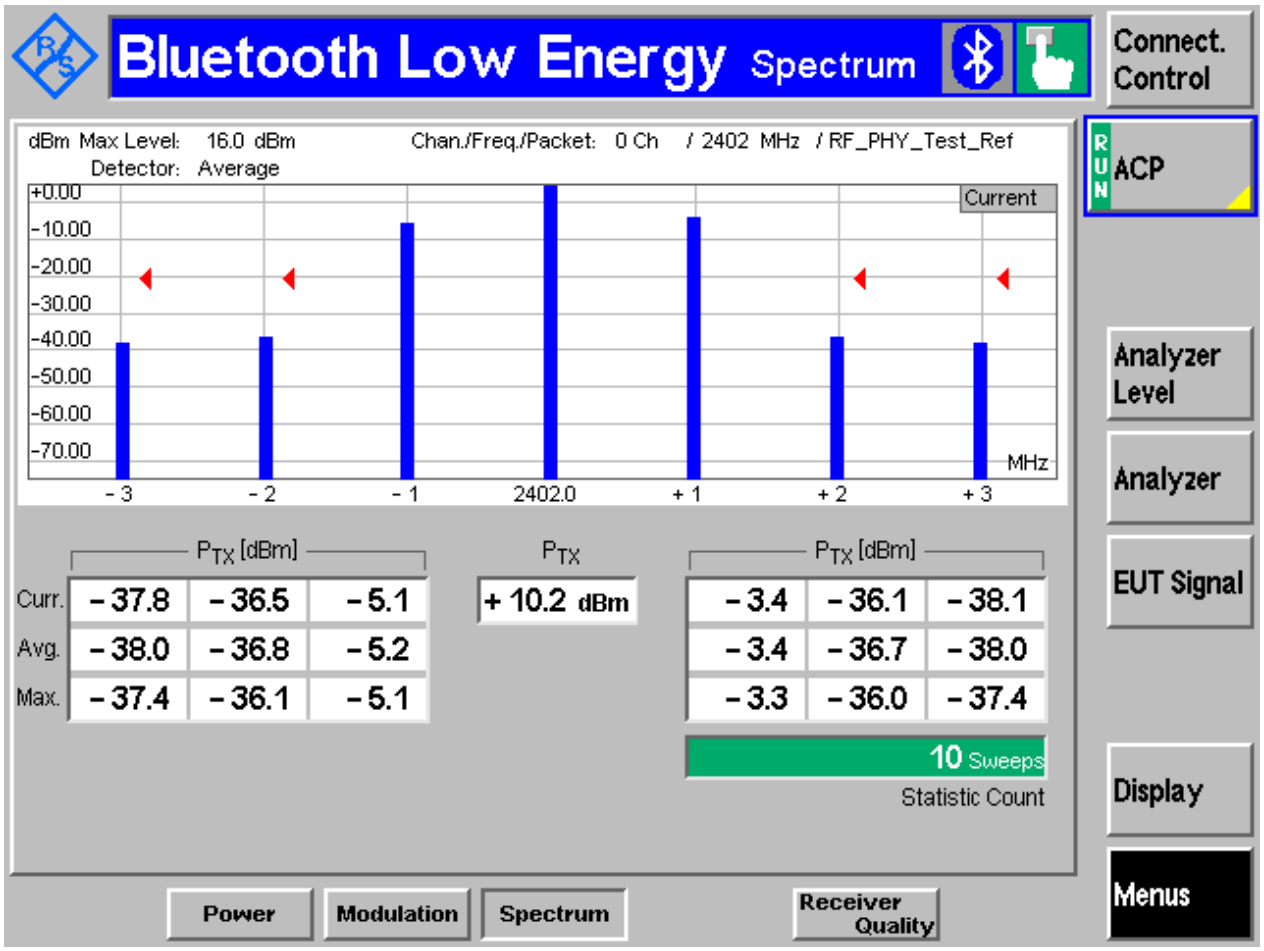


Figure 5.3. Spectrum Test

5.4 Receiver Tests

Used to make all of the sensitivity-based measurements, as well as blocker measurements.

In this example, the `test_dtm_rx` command is used. In Bluetooth NCP Commander Standalone, the test is started by selecting **Low energy receive** and then pressing the **Start test** button.

A spectrum analyzer and two RF generators are used to validate the receiver tests and must be connected through a compatible RF cable to the RF connector of the radio board.

A separate Bluetooth tester device can be used, but you might have to consider adding to it an external signal generator to provide the second interferer or blocker signal.

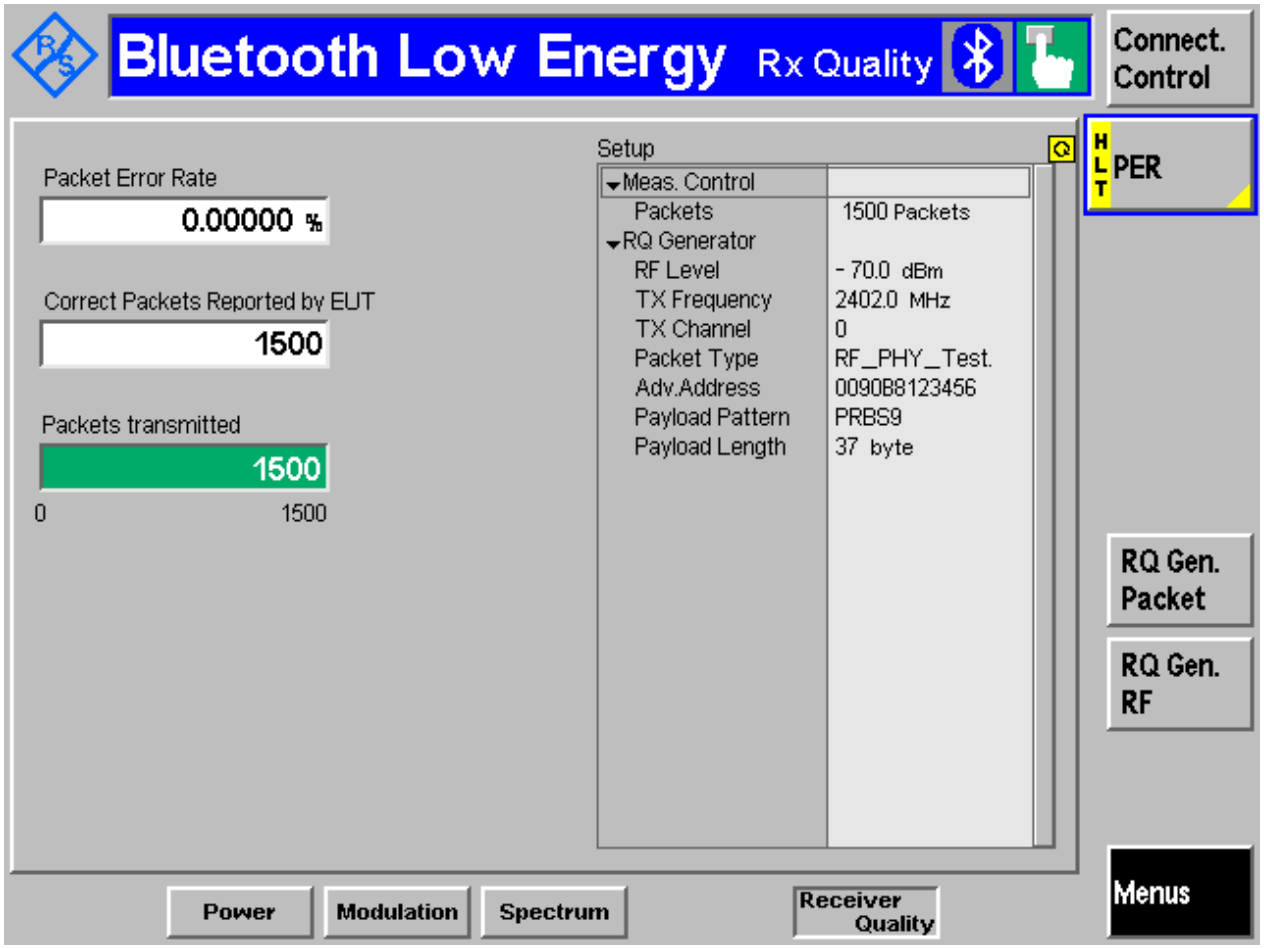


Figure 5.4. Receiver Test

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