GR5xx DTM Test Application Note

Version: 3.5

Release Date: 2024-08-28

Shenzhen Goodix Technology Co., Ltd.

Shenzhen Goodix Technology Co., Ltd.

Headquarters: Floor 13, Phase B, Tengfei Industrial Building, Futian Free Trade Zone, Shenzhen, China

TEL: +86-755-33338828 Zip Code: 518000

Website: www.goodix.com

Preface

Purpose

This document introduces the test framework and test methods of Direct Test Mode (DTM) in Bluetooth Low Energy (Bluetooth LE) applications. It describes how to use a Bluetooth tester and DTMTool (an RF test tool) to test the radio frequency (RF) performances of GR5xx System-on-Chips (SoCs) and demonstrates a GR5xx DTM example, helping users quickly get started with secondary development.

Audience

This document is intended for:

- Device user
- Developer
- Test engineer
- Technical support engineer

Release Notes

This document is the seventh release of *GR5xx DTM Test Application Note*, corresponding to Bluetooth LE GR5xx SoC series.

Revision History

Version	Date	Description
1.0	2023-01-10	Initial release
3.0	2023-03-30	• Updated descriptions about GR5xx SoCs. • Updated chapters "DTM Tests with DTMTool (Manual Operation)" and "Appendix: Common HCI Commands in RF Performance Test".
3.1	2023-04-20	Updated the section "Configuring UART Ports".
3.2	2023-09-07	Added the section "GR533x DTM Project Configuration".
3.3	2023-11-06	Updated the approach for obtaining GProgrammer.
3.4	2024-08-06	• Added a note in the section "Project Directory". • Updated the section "Configuring UART Ports".
3.5	2024-08-28	• Updated the section "Environment Setup". • Updated the section "GR533x DTM Project Configurations". • Added a note about the SoC series to which the AoA/AoD functionality applies.

Contents

1 Introduction

The Direct Test Mode (DTM) is a standard mechanism defined by Bluetooth Special Interest Group (Bluetooth SIG) to test the radio frequency (RF) performance of Bluetooth devices for all Bluetooth channels. In general, DTM is used in application validation and product line tests, enabling RF engineers to effortlessly verify and optimize Bluetooth RF performances. Tested RF factors include:

- Transmission power and receiver sensitivity
- Frequency offset and drift
- Modulation characteristics
- Packet error rate

The DTM test on GR5xx System-on-Chips (SoCs) is based on UART physical interfaces and a Bluetooth tester. The Bluetooth tester controls GR5xx RF physical layer (PHY) by using Host Controller Interface commands (HCI CMD) to implement test procedures.

Before getting started, you can refer to the following documents.

Table 1-1 Reference documents

Name	Description
Developer guide of the specific GR5xx SoC	Introduces GR5xx Software Development Kit (SDK) and how to develop and debug applications based on the SDK.
J-Link/J-Trace User Guide	Provides J-Link operational instructions. Available at https://www.segger.com/downloads/jlink/UM08001_JLink.pdf.
Keil User Guide	Offers detailed Keil operational instructions. Available at https://www.keil.com/support/man/docs/uv4/.

2 DTM Test Framework and Procedures

This chapter introduces the test framework and fundamental test procedures of DTM.

2.1 DTM Test Framework

A DTM test involves three types of test devices:

- Device Under Test (DUT)
- Upper Tester (UT)
- Lower Tester (LT)

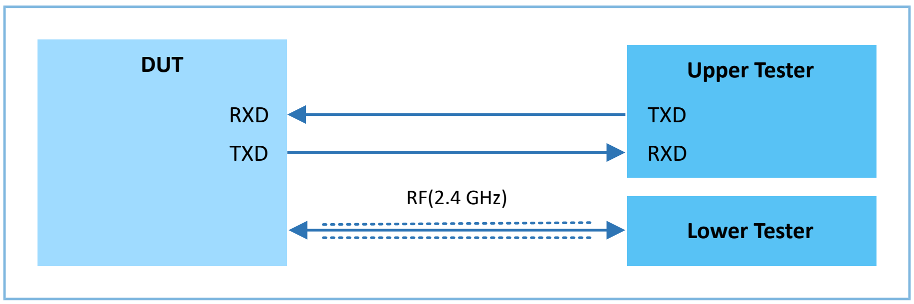


Figure 2-1 DTM test framework

DUT can work as a Transmitter Device (TXD) that transmits (TX) Bluetooth LE packets or a Receiver Device (RXD) that receives (RX) Bluetooth LE packets.

- TX test

  As a TXD, the DUT transmits test group sequences at a specified frequency to the LT that analyzes the transmission power, frequency offset, and modulation characteristics based on the received signals.

- RX test

  As an RXD, the DUT receives test group sequences transmitted by the LT at a specified frequency that is the same with that in LT transmission. After receiving, Bit Error Rate (BER) and other indicators received by the DUT are used to measure the RXD performance.

  UT communicates with DUT through UART, and LT communicates with DUT through Bluetooth LE links. The UART parameters of DUT are as follows:

Table 2-1 UART parameter setting in UT-DUT communications

Baud Rate (Recommended)	Bit Size	Checksum Bit	Stop Bit	Flow Control
1200, 2400, 9600, 14400, 19200, 38400, 57600, 115200, 230400, 460800, and 921600	8 bits	N/A	1 bit	Not supported

2.2 DTM Test Procedures

During DTM tests, the UT transmits an HCI test command through a UART interface to the DUT that returns the test Status Event or Packet Report Event to the UT. Before receiving a reply from the DUT, the UT does not deliver another test command. If the UT does not receive any reply from the DUT within a regulated time period, the UT delivers a reset command to the DUT.

The mapping between RF test commands/events and HCI commands/events is shown in Table 2-2.

Table 2-2 Mapping between RF test commands/events and HCI commands/events

RF Test Command/Event	HCI Command/Event
LE_TRANSMITTER_TEST	LE Transmitter Test Command
LE_RECEIVER_TEST	LE Receiver Test Command
LE_TEST_END	LE Test End Command
LE_STATUS	Command Complete Event
LE_PACKET_REPORT	Command Complete Event

Figure 2-2 illustrates the TX, RX, and End procedures of a DTM test.

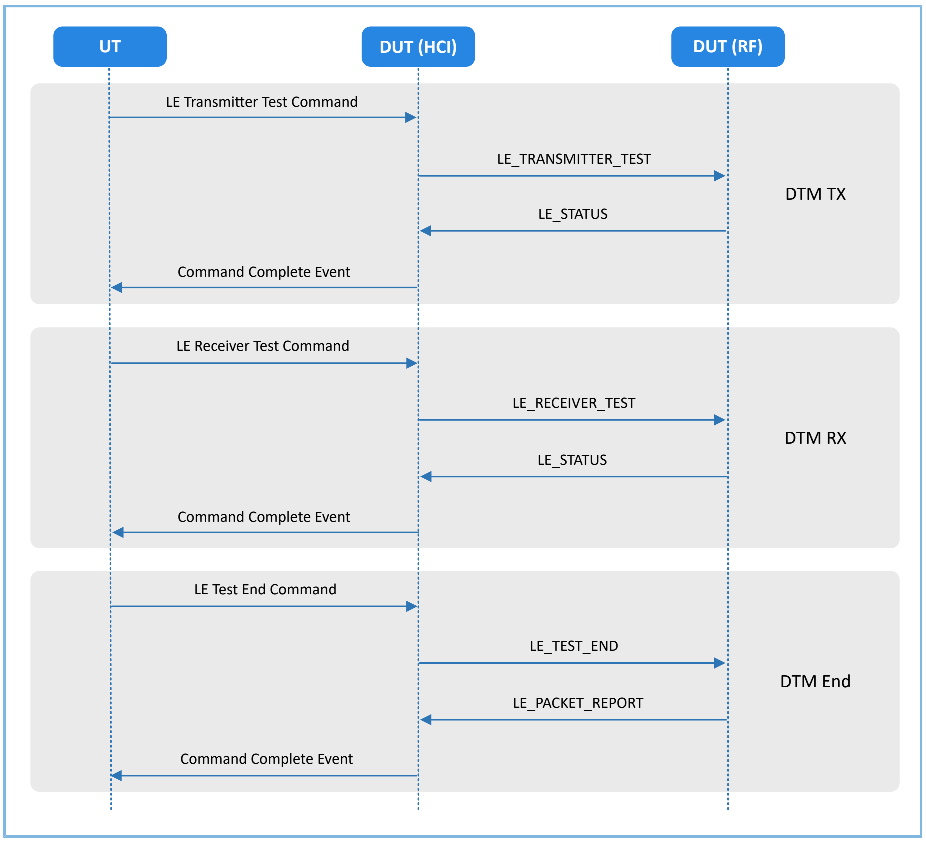


Figure 2-2 DTM test procedure example

3 DTM Test and Verification

GR5xx adopts two-wire UART to transmit HCI frame data to control RF PHY. Therefore, DTM test and verification can be performed by any Bluetooth tester that supports HCI format.

This chapter introduces how to quickly set up a DTM test environment and perform DTM test and verification with TLF3000/CMW500 as the Bluetooth tester and a development board as the DUT.

Note:

SDK_Folder is the root directory of the GR5xx SDK in use.

3.1 Preparation

You need the following hardware and software to set up a DTM test environment.

- **Hardware preparation**

Table 3-1 Hardware preparation

Name	Description
Development board	Starter Kit Board (SK Board) of the corresponding SoC
Connection cable	USB Type-C cable (Micro USB 2.0 cable for GR551x SoCs)
Bluetooth tester	CMW500/TLF3000

- **Software preparation**

Table 3-2 Software preparation

Name	Description
Windows	Windows 7/Windows 10
J-Link driver	A J-Link driver. Available at https://www.segger.com/downloads/jlink/.
Keil MDK5	An integrated development environment (IDE). MDK-ARM Version 5.20 or later is required. Available at https://www.keil.com/download/product/.
GProgrammer (Windows)	A programming tool. Available at https://www.goodix.com/en/software_tool/gprogrammer_ble.
TLF3000 RF-PHY Tester and driver	Downloaded at: https://www.fte.com/products/tlf3000.aspx.
R&S®CMW500 Tester and driver	Downloaded at: https://www.rohde-schwarz.com.

3.2 Firmware Programming

The source code of the DTM example is in `SDK_Folder\projects\ble\dtm\direct_test_mode`.

You can directly program the firmware file *direct_test_mode.bin*, which is located in `SDK_Folder\projects\ble\dtm\direct_test_mode\build\`, to the SK Board through GProgrammer. For details, see *GProgrammer User Manual*.

Note:

If the DUT is a GR533x device, you should modify the project configurations according to the specific application scenario of the RF power amplifier (PA). Once the necessary modifications have been completed, rebuild and program the new firmware. For detailed configurations, refer to "Section 6.2 GR533x DTM Project Configurations".

3.3 Environment Setup

- Environment setup for DTM tests with TLF3000:
  1. Install TLF3000 RF-PHY Tester and driver on a PC.
  2. Connect TLF3000 to the PC with a Micro USB cable, then connect the SK Board to the PC with a USB cable (Micro USB 2.0 cable for GR551x SoCs; USB Type-C cable for other GR5xx SoCs), and connect the antenna interface on the SK Board to the TX/RX interface on TLF3000 with an RF cable.

     The figure below shows the hardware connection.

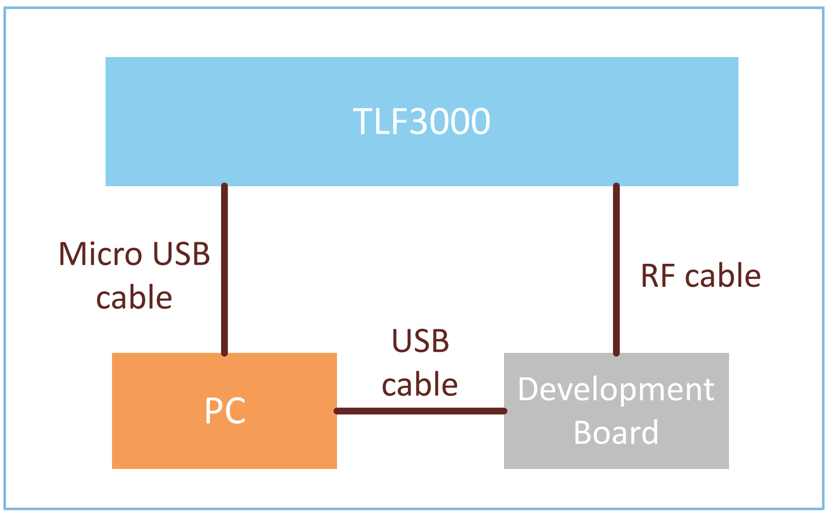


Figure 3-1 Hardware connection diagram (TLF3000)

- Environment setup for DTM tests with CMW500:
  1. Install the serial driver on CMW500. To use the CDC virtual serial port on the SK Board, you need to install the J-Link driver, and V7.96f or above is recommended.

Note:

To use the CDC virtual serial port, the transmission rate should not exceed 1 MHz. If a higher transmission rate is required, it is recommended to use the USB-to-UART connector on the SK Board, which can reach a maximum rate of 3 MHz. For details, refer to the corresponding Starter Kit user manual.

  2. Connect CMW500 to the UART port on the SK Board with a USB cable (Micro USB 2.0 cable for GR551x SoCs; USB Type-C cable for other GR5xx SoCs), and connect CMW500 to the RF port on the SK Board with an RF cable.

     The figure below shows the hardware connection.

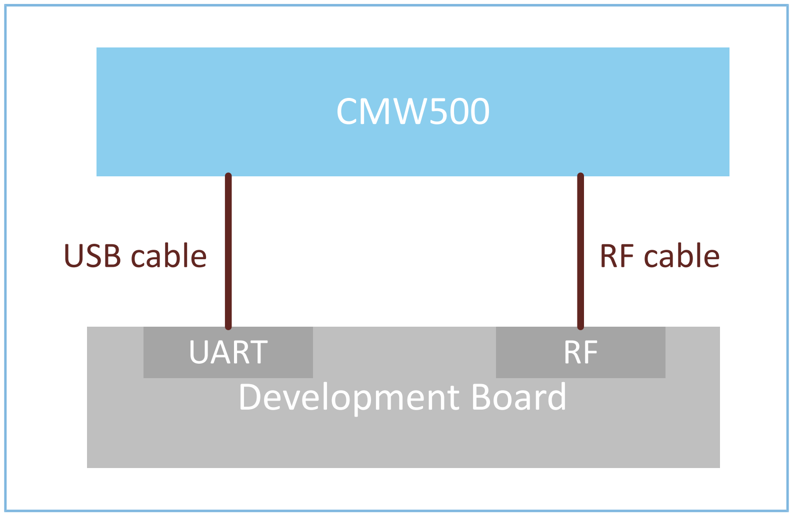


Figure 3-2 Hardware connection diagram (CMW500)

3.4 Test and Verification Procedures

This section introduces the DTM test and verification procedures by using TLF3000 and R&S® CMW500.

3.4.1 DTM Testing with TLF3000

DTM test procedures with TLF3000:

1. Power the TLF3000 hardware on, start the TLF3000 software, and select the TLF3000 model used in the DTM test. Click **RUN APP** to enter the TLF3000 software GUI.

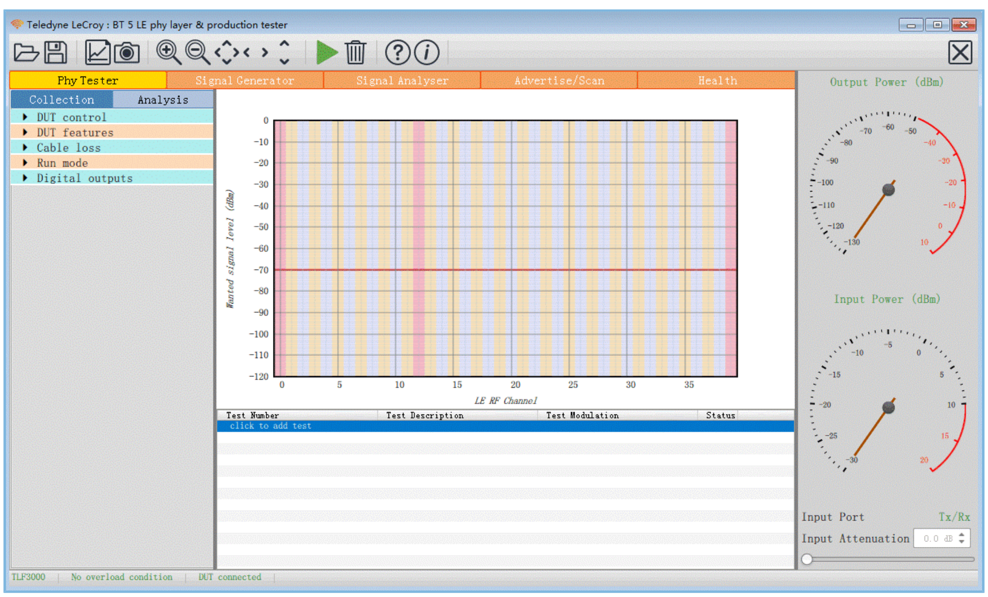


Figure 3-3 Hardware connection in a TLF3000 test environment

2. Configure **DUT control** parameters. Select a serial port from the **Comport** list. Select **H4** from **Interface** and **115200** from **Baud rate**. Select **None** from the **HW flow**, **SW flow**, and **Parity bits** lists. Select **One** from the **Stop bits** list.

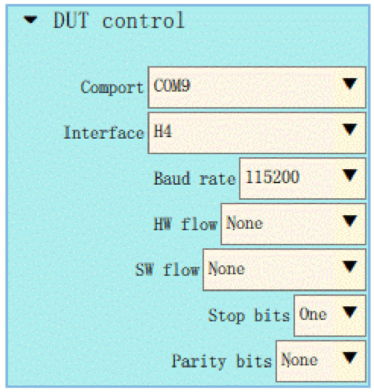


Figure 3-4 DUT control configuration

3. Open the **DUT features** pane, and click **Query DUT** to obtain the DUT-related features.

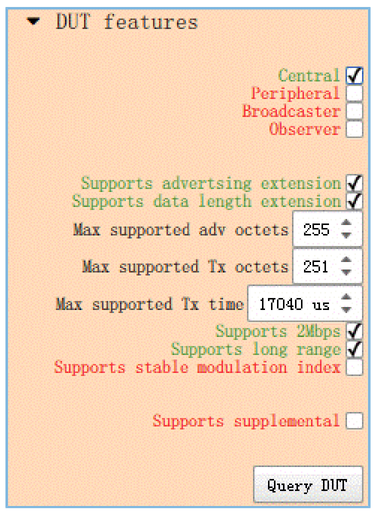


Figure 3-5 Obtaining DUT features

4. Select a running mode in the **Run mode** pane.

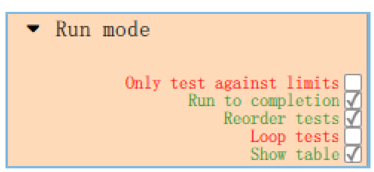


Figure 3-6 Configuring Run mode

5. Click **click to add test**, and the **Phy Level Test** window opens. Select test cases in the **TP/TRM-LE/CA/BV-12** tab. Configure the **Channels**, **Packet Lengths**, and **Number of packets** for the selected test cases. After configuration, click **Apply** to complete test adding.

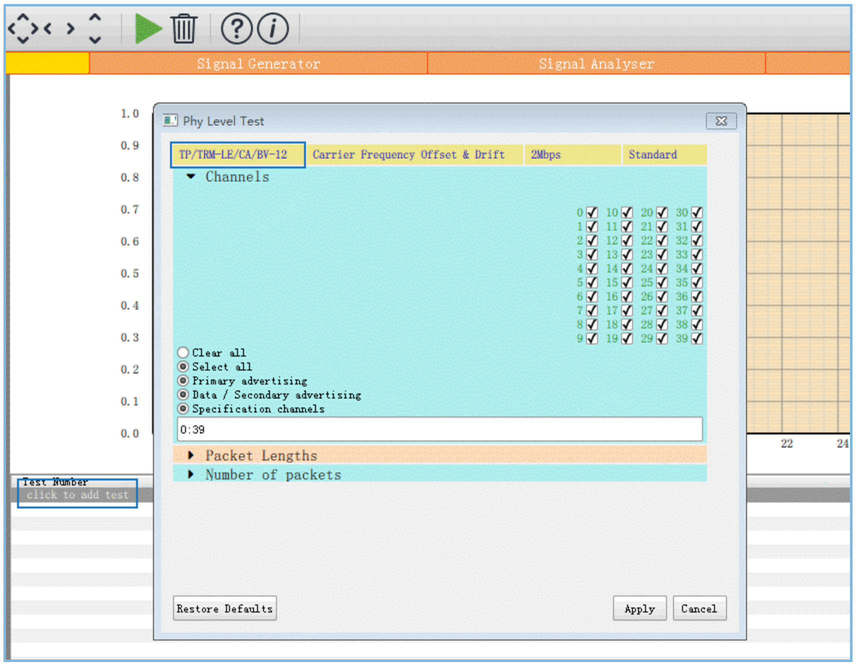


Figure 3-7 Selecting test cases

6. Click to start the DTM test. When the test completes, a window containing logs in HTML format automatically opens. Click to store the logs in an HTML file, or click to save the log screenshot.

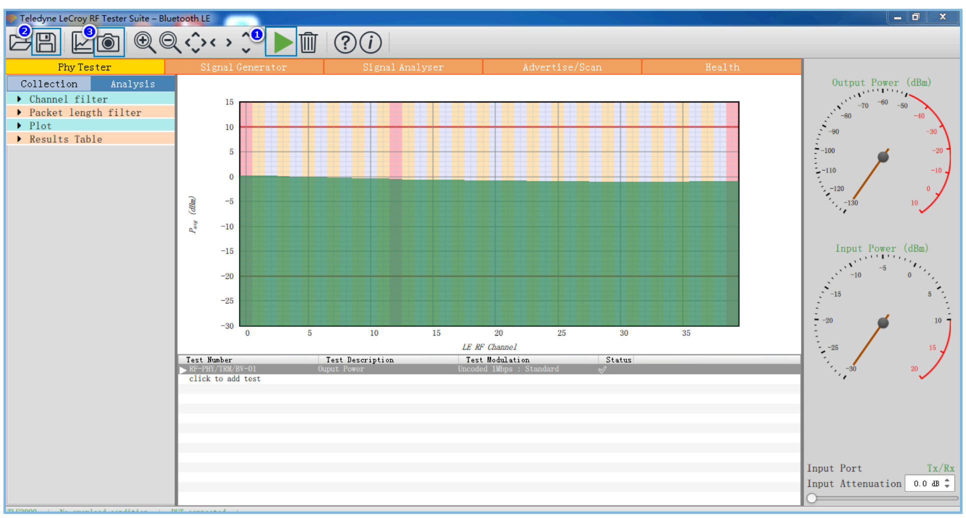


Figure 3-8 Running test cases

3.4.2 DTM Testing with R&S® CMW500

DTM test procedures with R&S® CMW500:

1. Configuring the Bluetooth signaling test mode on the R&S® CMW500. In this mode, the R&S® CMW500 automatically delivers HCI commands to interact with the SK Board, enabling automated DTM tests.

(1) On the R&S® CMW500, press **MEASURE** to open the **Measurement Controller** window, and select **Measurements** in **Bluetooth**.

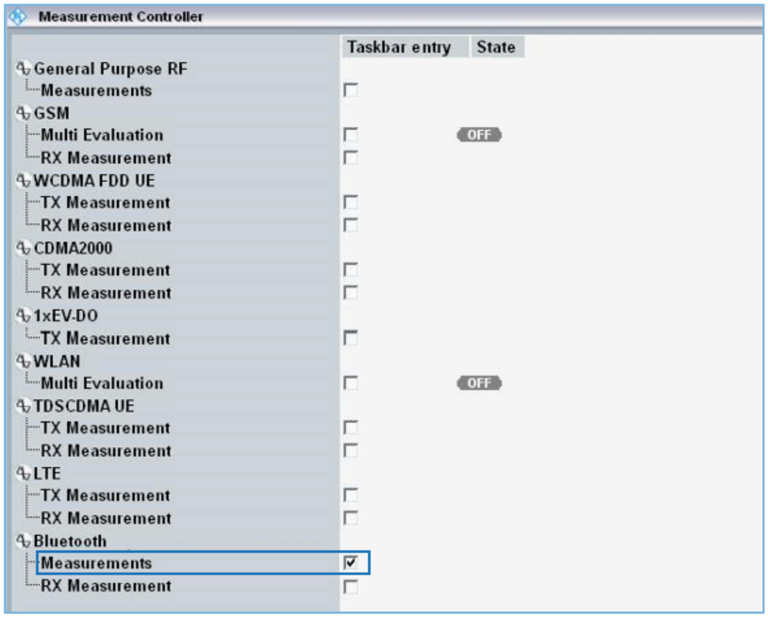


Figure 3-9 Configuring Bluetooth measurements for R&S® CMW500

(2) On the R&S® CMW500, press **SIGNALGEN** to open the **Generator/Signaling Controller** window, and select **Signaling** in **Bluetooth**.

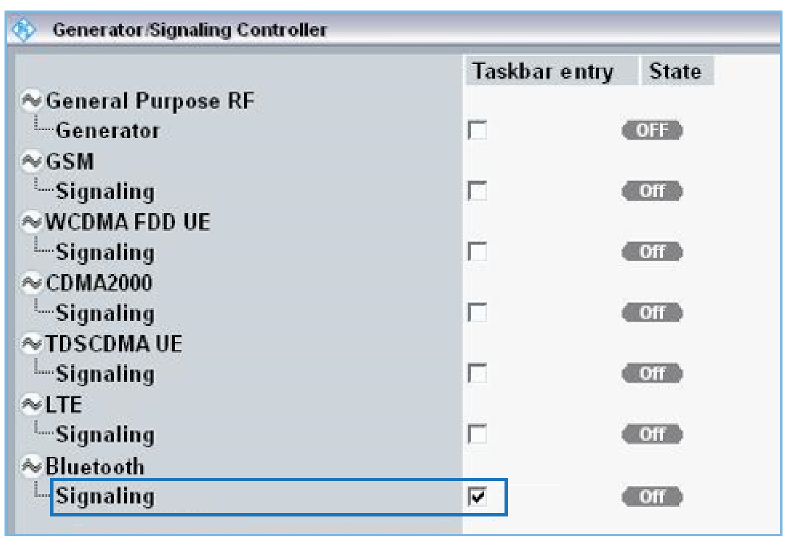


Figure 3-10 Configuring Bluetooth signaling mode for R&S® CMW500

2. Connecting to UART

(1) Confirm that the R&S® CMW500 serial drivers have been properly installed on the PC. Make sure that the R&S® CMW500 tester has been correctly connected to the UART port of the SK Board.

(2) The R&S® CMW500 tester automatically identifies the COM port used by the UART, and developers only need to configure the baud rate, odd-even parity, and other parameters related to the DTM test firmware. After setting these parameters, click **Connection Check** to connect the R&S® CMW500 tester to the UART. You can implement the DTM test after successful connection to UART.

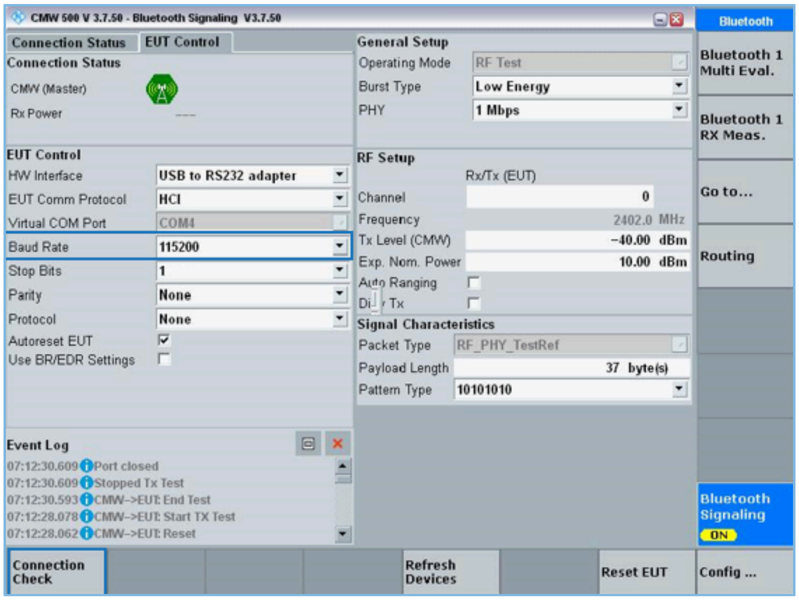


Figure 3-11 Connecting to UART

3. Running DTM automation test

Start R&S® CMWrun, and select **CMW RUN**. Load the Bluetooth LE test program, BLE_PHY_5_0_0, and click to run the DTM test.

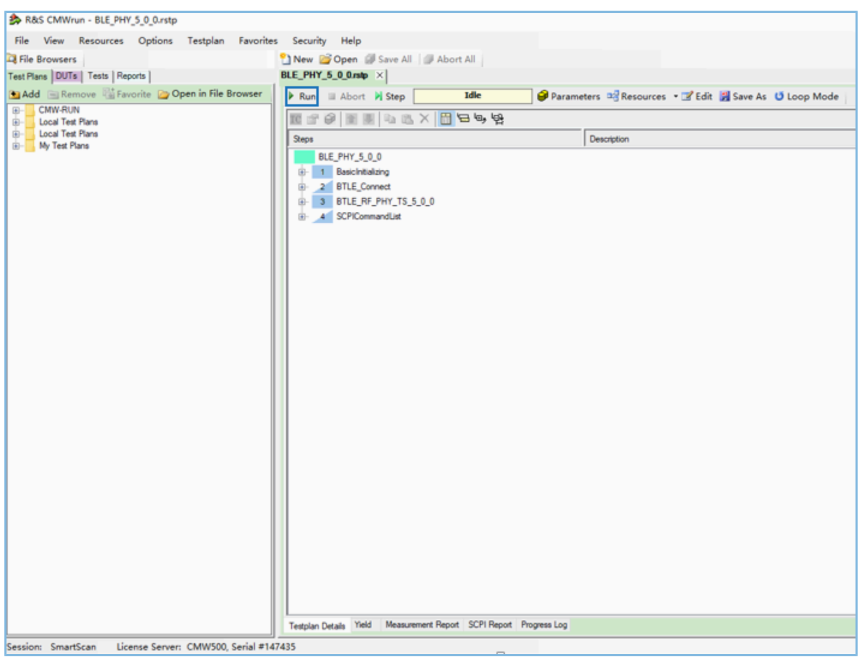
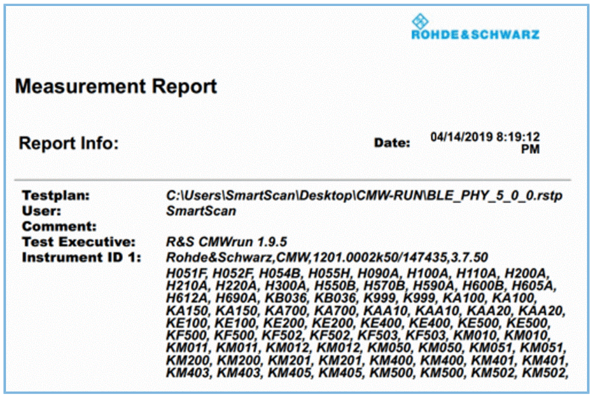


Figure 3-12 R&S®CMW500 DTM test interface

4. Generating a test report.

   After the test completes, the R&S® CMW500 software tool automatically generates a test report, as shown in Figure 3-13.


ROHDE&SCHWARZ

Measurement Report

Report Info: Date: 04/14/2019 8:19:12 PM

Testplan:	C:\Users\SmartScan\Desktop\CMW-RUN\BLE_PHY_5_0_0.rstp
User:	SmartScan
Comment:	
Test Executive:	R&S CMWrun 1.9.5
Instrument ID 1:	Rohde&Schwarz,CMW,1201.0002k50/147435,3.7.50
	H051F, H052F, H054B, H055H, H090A, H100A, H110A, H200A, H210A, H220A, H300A, H550B, H570B, H590A, H600B, H605A, H612A, H690A, KB036, KB036, K999, K999, KA100, KA100, KA150, KA150, KA700, KA700, KAA10, KAA10, KAA20, KAA20, KE100, KE100, KE200, KE200, KE400, KE400, KE500, KE500, KF500, KF500, KF502, KF502, KF503, KF503, KM010, KM010, KM011, KM011, KM012, KM012, KM050, KM050, KM051, KM051, KM200, KM200, KM201, KM201, KM400, KM400, KM401, KM401, KM403, KM403, KM405, KM405, KM500, KM500, KM502, KM502,

Figure 3-13 CMW500 DTM test report

Note:

The R&S® CMW500 automation software tool (R&S® CMWrun) should be purchased by users.

---

4 Application Details

Settings for serial ports vary, depending on the DUT. Users can modify the configurations on the serial ports of the direct_test_mode project to meet the DTM test requirements from the DUT.

4.1 Project Directory

The direct_test_mode project is in `SDK_Folder\projects\ble\dtm\direct_test_mode`, and the project file is located in the Keil_5 folder.

Note:

The direct_test_mode project for GR5405 SDK is in `SDK_Folder\projects\ble\dtm\dtm`.

Double-click the project file, *direct_test_mode.uvprojx*, to view the direct_test_mode project directory structure of the DTM application in Keil. For related files, see Table 4-1.

Table 4-1 Description of direct_test_mode files

Group	File	Description
gr_libraries	hci_uart.c	This file adapts to the UART drivers of Bluetooth LE Stack DTM test.
user_app	main.c	This file contains the main() function, enabling registration of HCI UART interfaces.

4.2 Configuring UART Ports

UART ports of GR5xx SoCs are listed below. In the direct_test_mode example project, an SK Board is used as the DUT. By default, UART0 (UART3 for GR5526) serves as the HCI communication interface.

Table 4-2 UART ports of GR5xx SoCs

GR5xx SoC	UART Port
GR551x	UART0 and UART1
GR5526	UART0 to UART5
GR5525	UART0 to UART3
GR533x	UART0 and UART1
GR5405	UART0 and UART1

According to the I/O mapping of UART ports on the SK Board, the configuration information (in `platform\boards\board_SK.h` under the project directory) of GR5xx SoCs is shown as follows.

- GR551x

```
/*******HCI UART IO CONFIG***********************/
#define HCI_UART_ID                         APP_UART_ID_0
#define HCI_UART_FLOW_ON                    0
#define HCI_UART_BAUDRATE                   115200
#define HCI_UART_TRN_PORT                   APP_IO_TYPE_NORMAL
#define HCI_UART_FLOW_PORT                  APP_IO_TYPE_NORMAL
#define HCI_UART_TX_PIN                     APP_IO_PIN_10
```

```
#define HCI_UART_RX_PIN                        APP_IO_PIN_11
#define HCI_UART_CTS_PIN                       APP_IO_PIN_2
#define HCI_UART_RTS_PIN                       APP_IO_PIN_5
#define HCI_UART_TX_PINMUX                     APP_IO_MUX_2
#define HCI_UART_RX_PINMUX                     APP_IO_MUX_2
#define HCI_UART_CTS_PINMUX                    APP_IO_MUX_0
#define HCI_UART_RTS_PINMUX                    APP_IO_MUX_0
```

- GR5526

```
/*******HCI UART IO CONFIG***********************/
#define APP_HCI_UART_ID                            APP_UART_ID_3
#define APP_HCI_UART_FLOW_ON                       0
#define APP_HCI_UART_BAUDRATE                      115200
#define APP_HCI_UART_TRN_PORT                      APP_IO_TYPE_GPIOA
#define APP_HCI_UART_FLOW_PORT                     APP_IO_TYPE_GPIOA
#define APP_HCI_UART_TX_PIN                        APP_IO_PIN_4
#define APP_HCI_UART_RX_PIN                        APP_IO_PIN_5
#define APP_HCI_UART_CTS_PIN                       APP_IO_PIN_8
#define APP_HCI_UART_RTS_PIN                       APP_IO_PIN_9
#define APP_HCI_UART_TX_PINMUX                     APP_IO_MUX_3
#define APP_HCI_UART_RX_PINMUX                     APP_IO_MUX_3
#define APP_HCI_UART_CTS_PINMUX                    APP_IO_MUX_4
#define APP_HCI_UART_RTS_PINMUX                    APP_IO_MUX_4
#define APP_HCI_UART_TRIGGER_PIN                   AON_GPIO_PIN_1
```

- GR5525

```
/*******HCI UART IO CONFIG***********************/
#define APP_HCI_UART_ID                           APP_UART_ID_0
#define APP_HCI_UART_FLOW_ON                      0
#define APP_HCI_UART_BAUDRATE                     115200
#define APP_HCI_UART_TRN_PORT                     APP_IO_TYPE_GPIOA
#define APP_HCI_UART_FLOW_PORT                    APP_IO_TYPE_GPIOA
#define APP_HCI_UART_TX_PIN                       APP_IO_PIN_10
#define APP_HCI_UART_RX_PIN                       APP_IO_PIN_11
#define APP_HCI_UART_CTS_PIN                      APP_IO_PIN_2
#define APP_HCI_UART_RTS_PIN                      APP_IO_PIN_5
#define APP_HCI_UART_TX_PINMUX                    APP_IO_MUX_2
#define APP_HCI_UART_RX_PINMUX                    APP_IO_MUX_2
#define APP_HCI_UART_CTS_PINMUX                   APP_IO_MUX_0
#define APP_HCI_UART_RTS_PINMUX                   APP_IO_MUX_0
#define APP_HCI_UART_TRIGGER_PIN                  AON_GPIO_PIN_1
```

- GR533x

```
/*******HCI UART IO CONFIG***********************/
#define APP_HCI_UART_ID                           APP_UART_ID_0
#define APP_HCI_UART_FLOW_ON                      0
#define APP_HCI_UART_BAUDRATE                     115200
#define APP_HCI_UART_TRN_PORT                     APP_IO_TYPE_AON
#define APP_HCI_UART_FLOW_PORT                    APP_IO_TYPE_AON
#define APP_HCI_UART_TX_PIN                       APP_IO_PIN_0
#define APP_HCI_UART_RX_PIN                       APP_IO_PIN_1
#define APP_HCI_UART_TX_PINMUX                    APP_IO_MUX_7
#define APP_HCI_UART_RX_PINMUX                    APP_IO_MUX_8
#define APP_HCI_UART_CTS_PINMUX                   APP_IO_MUX_5
#define APP_HCI_UART_RTS_PINMUX                   APP_IO_MUX_6
```


- GR5405

```
/*******HCI UART IO CONFIG***********************/
#define APP_HCI_UART_ID APP_UART_ID_0
#define APP_HCI_UART_FLOW_ON 0
#define APP_HCI_UART_BAUDRATE 115200
#define APP_HCI_UART_TRN_PORT APP_IO_TYPE_AON
#define APP_HCI_UART_FLOW_PORT APP_IO_TYPE_AON
#define APP_HCI_UART_TX_PIN APP_IO_PIN_0
#define APP_HCI_UART_RX_PIN APP_IO_PIN_1
#define APP_HCI_UART_TX_PINMUX APP_IO_MUX_7
#define APP_HCI_UART_RX_PINMUX APP_IO_MUX_8
#define APP_HCI_UART_CTS_PINMUX APP_IO_MUX_5
#define APP_HCI_UART_RTS_PINMUX APP_IO_MUX_6
```

Note:

- To use other UART ports on the SK Board as the HCI communication interface, modify the APP_HCI_UART_ID macro in *board_SK.h*, and then modify the UART pin macro based on the I/O mapping of UART ports on the SK Board. For more information about I/O multiplexing on GR5xx SoCs, refer to the corresponding datasheet.
- If you do not use an SK Board as the DUT, you need to modify the settings of the UART ports according to the hardware layout of the DUT in use.

4.3 Transplanting DTM Functionality

This section elaborates on how to transplant the DTM functionality to a Bluetooth LE application project.

To do that, follow the steps below:

1. Add the *hci_uart.c* file to the application project, and refer to *hci_uart.h* in the *main.c* file.
2. Call the ble_stack_init() function to initialize the Bluetooth Protocol Stack.
3. During initialization, call the ble_hci_uart_init() function to initialize the UART port and communications protocol stack required in a DTM test.

Path: `user_app\main.c` under *direct_test_mode.uvprojx*

Name: main()

```
#include "hci_uart.h"

int main (void)
{
    ble_hci_uart_init();
    ble_stack_init(&ble_evt_handler, &heaps_table);/*< init ble stack*/


    //loop
    while(1)
    {

    }
}
```

5 DTM Tests with DTMTool (Manual Operation)

In general, DTM tests are performed by dedicated instruments automatically. The DUT communicates with a test instrument through HCI commands. You can also use GRDirect Test Mode Tool (hereinafter referred to as "DTMTool") to simulate a test instrument to perform tests by sending DTM test commands.

This chapter introduces how to use DTMTool to control the DUT to enter the DTM TX or RX mode, as well as DTMTool functionalities. Operations of UT are customized according to the Core-Test Requirements, and are not described in this document.

Note:

DTM tests shall be performed in shielding boxes.

5.1 Introduction to DTMTool

As an RF test tool, DTMTool controls the DUT to perform DTM tests by delivering HCI commands.

DTMTool is portable. It is ready for use by extracting the .zip folder for installation and then double-clicking *GRDirect Test Mode Tool.exe* in the folder.

The main interface of DTMTool is shown below:

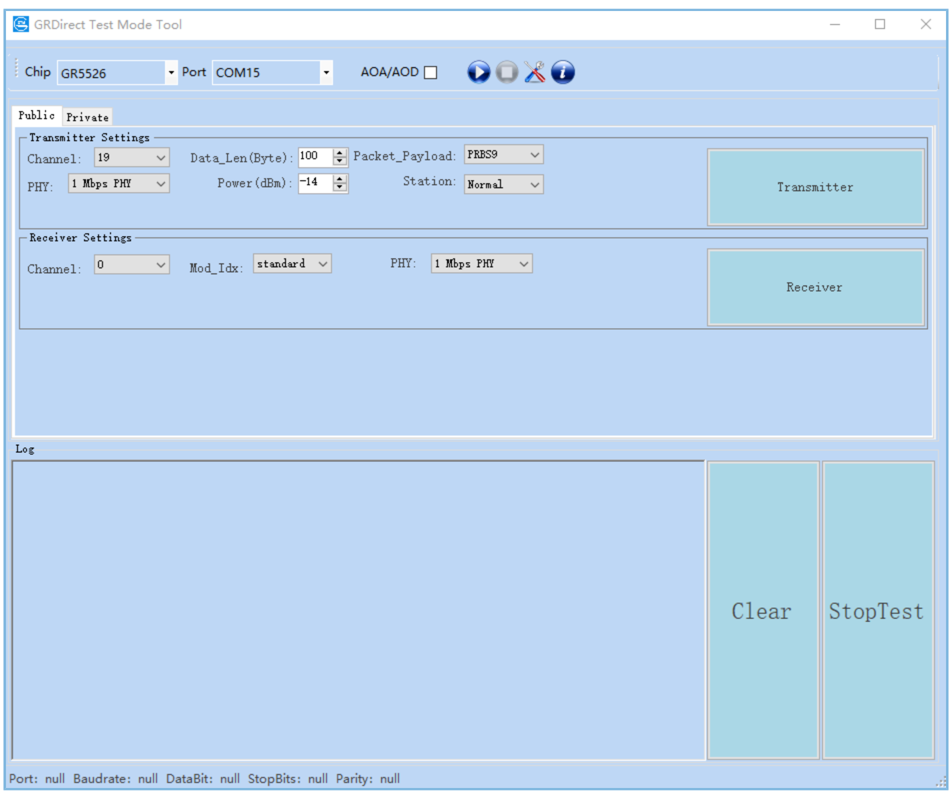


Figure 5-1 Main interface of DTMTool

Note:

DTMTool screenshots in this document are provided by taking a GR5526 SoC for example.

- **Chip**: GR5xx SoC part number
- **Port**: port No.
- **AOA/AOD**: Configure whether to enable the AoA/AoD functionality during testing. If it is selected, Constant Tone Extension (CTE)-related parameters need to be configured.
- : Establish connection with the selected port to get ready for DTM testing.
- : Break the connection with the selected port.
- : Configure port parameters, such as **BaudRate**.
- **Public** tab: Configure TX and RX parameters. For details, refer to "Section 5.2 DTM TX Test (Manual Operation)" and "Section 5.3 DTM RX Test (Manual Operation)".
- **Private** tab: Configure parameters of a custom test item, including those for single carrier wave and reading/writing XO values. For details, refer to "Section 5.4 Private Commands".
- **Log** pane: Show logs about DUT connection state, DTM test process, and DTM test results.
  - **Clear**: Clear the data in the **Log** pane.
  - **StopTest**: Stop a DTM test.

Note:

The AoA/AoD functionality in DTMTool is for GR5526 series only, which also applies to the content below.

5.2 DTM TX Test (Manual Operation)

5.2.1 Initiating a DTM TX Test

1. After hardware connection completes, start DTMTool; then select the SoC series to be tested and the corresponding port No., check **AOA/AOD** if needed, and click to establish connection with the DUT.
2. After successful connection, click the **Public** tab, set parameters in the **Transmitter Settings** pane, and click **Transmitter**, to start a DTM TX test. Then, GR5xx transmits test data packets of a particular type on the specified channel at a fixed interval.
3. View corresponding test process and results in the **Log** pane.

- When **AOA/AOD** is not checked:

  The corresponding DTM TX test interface showing TX parameter setting, TX command, and response is as follows:

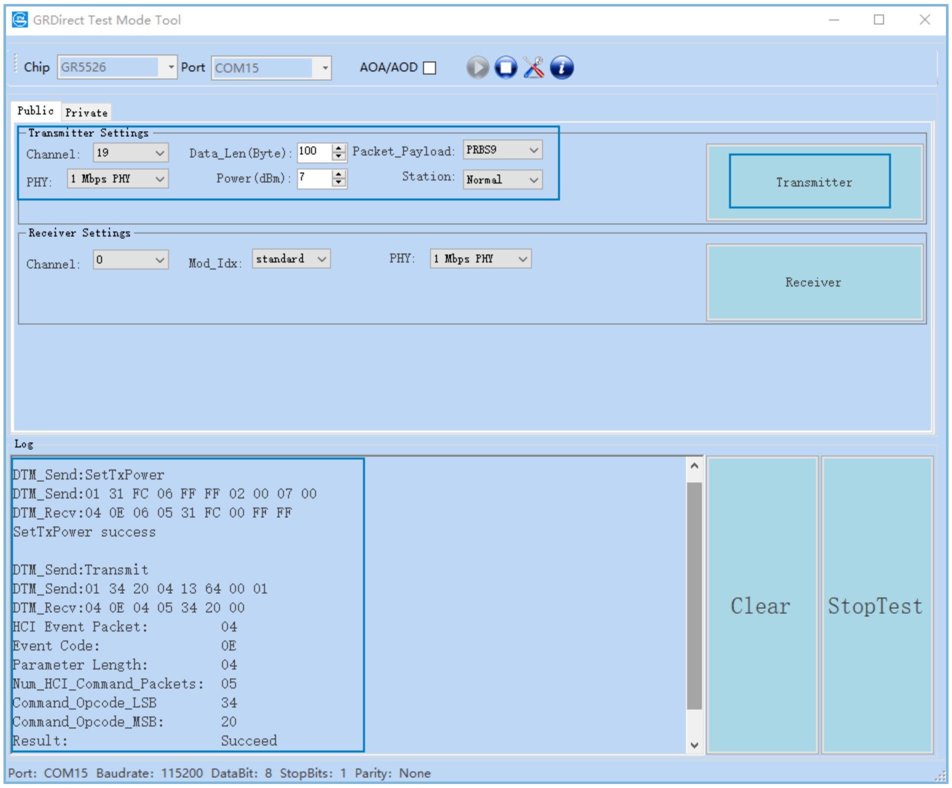


Figure 5-2 DTM TX test interface of DTMTool with **AOA/AOD** unchecked

- **Channel**: channel No.; options: Channel 0–Channel 39; packet transmission frequency: 2402 + Channel x 2 (MHz)
- **Station**: duty ratio for transmitting data; options: **Normal**, **High**, and **Low**
    - **Normal**: Use the standard packet transmission mode specified by *Bluetooth Core Spec*. This mode applies to normal tests (excluding sideband radiation and harmonic radiation tests).
    - **High**: Use the standard packet transmission mode specified by *Bluetooth Core Spec*; **Data_Len(Byte)** is invalid and the data packet is at a fixed length; **PHY** options 1M, 2M, S2 (500k), and S8 (125k) correspond to 193 bytes, 239 bytes, 229 bytes, and 229 bytes, respectively. This mode applies to sideband radiation tests.
    - **Low**: Use the Goodix duty ratio mode; **Data_Len(Byte)** is invalid and the data packet length is fixed at 1 byte; DTM packet transmission period is fixed at 5 x 312.5 µs. This mode applies to harmonic radiation tests to reduce the duty cycle.

You can initiate a DTM TX test without AoA/AoD by using the LE Transmitter Test Command V2 and the LE Modify TX Power Command (a private HCI command). For command details, see "Section 8.2 LE Transmitter Test Command V2" and "Section 8.9 LE Modify TX Power Command".

- When **AOA/AOD** is checked:

  The corresponding DTM TX test interface showing TX parameter setting, TX command, and response is as follows:

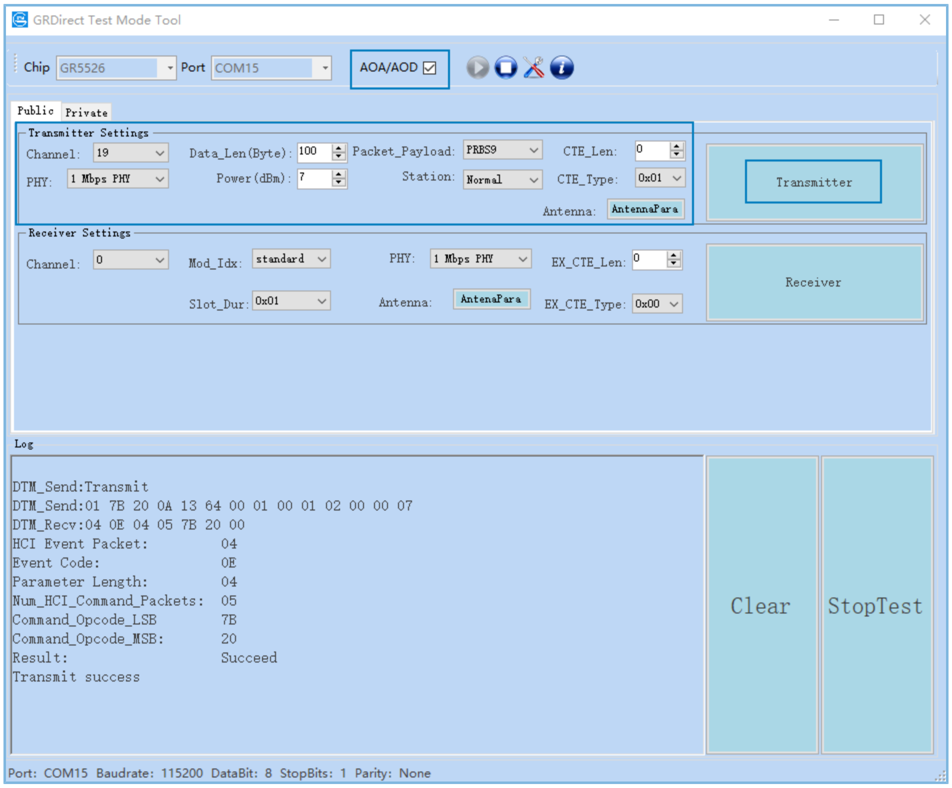


Figure 5-3 DTM TX test interface of DTMTool with **AOA/AOD** checked

- **CTE_Len**: CTE length
- **CTE_Type**: CTE type
- **Antenna**: antenna-related parameters. Click **AntennaPara** to configure related parameters in the pop-up window.

You can initiate a DTM TX test with AoA/AoD by using the LE Transmitter Test Command V4. For command details, see "Section 8.3 LE Transmitter Test Command V4".

5.2.2 Ending a DTM TX Test

After a DTM TX test completes, click **StopTest** to end the test.

Note:

To start a new DTM test, end the ongoing DTM test first.

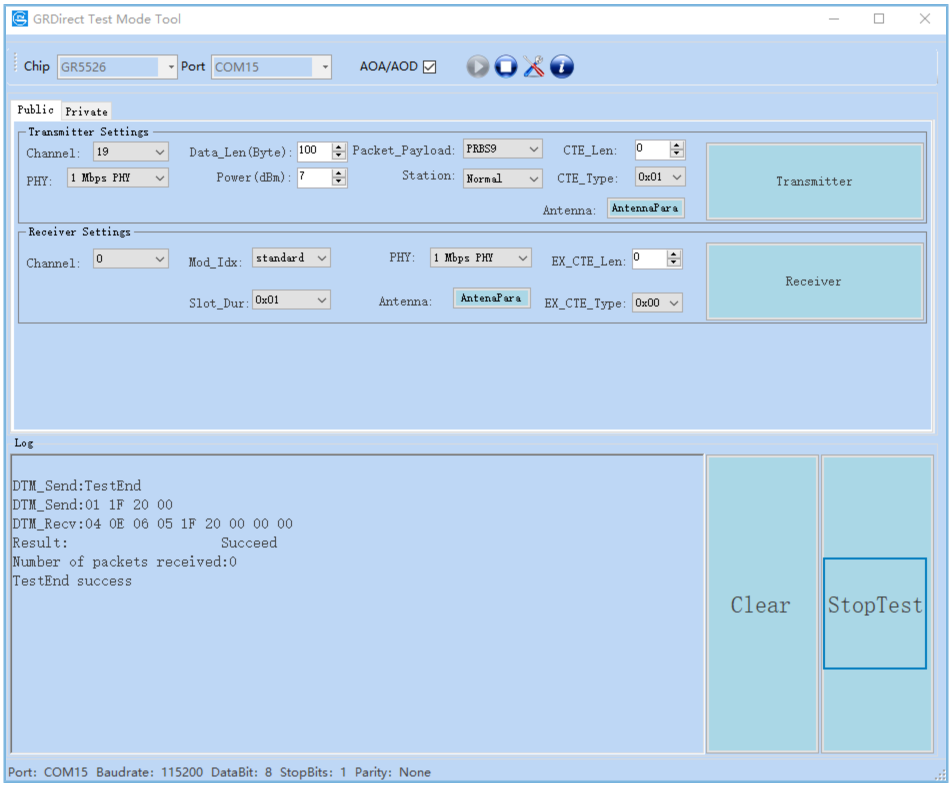


Figure 5-4 To end a DTM TX test

You can end a DTM TX test by using the LE Test End Command. For command details, see "Section 8.6 LE Test End Command".

5.2.3 Initiating a DTM TX Loop Test

DTMTool supports automatic DTM TX testing from Channel 0 to Channel 39 at a fixed interval.

1. Click **Private** on the main interface of DTMTool, and then click **Transmitter Loop Test** in the **Hopping settings** pane.

2. Set **Interval (ms)** for TX testing between channels in the pop-up window, then select **Loop**, and finally click **Start**, to initiate a DTM TX loop test from Channel 0 to Channel 39.

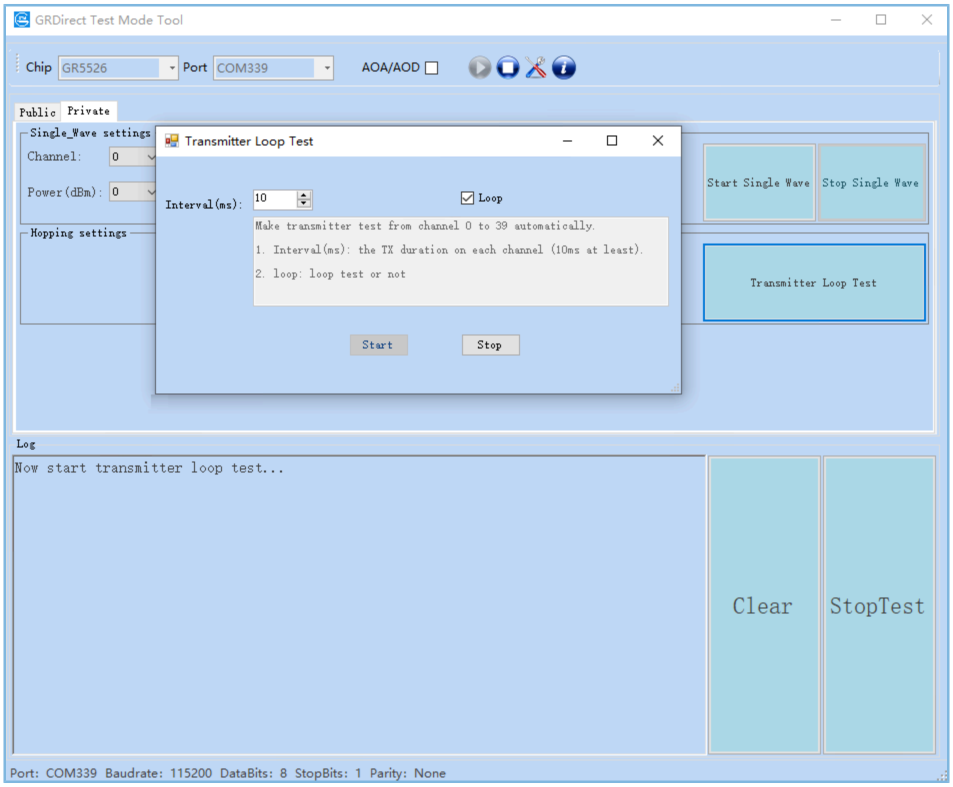


Figure 5-5 To initiate a DTM TX loop test

5.3 DTM RX Test (Manual Operation)

5.3.1 Initiating a DTM RX Test

1. After hardware connection completes, start DTMTool; then select the SoC series to be tested and the corresponding port No., check **AOA/AOD** if needed, and click to establish connection with the DUT.
2. After successful connection, click the **Public** tab, set parameters in the **Receiver Settings** pane, and click **Receiver**, to start a DTM RX test. Then, GR5xx will monitor data packets at a fixed frequency and record the number of packets received.
3. View corresponding test process and results in the **Log** pane.

- When **AOA/AOD** is not checked:

  The corresponding DTM RX test interface showing RX parameter setting, RX command, and response is as follows:

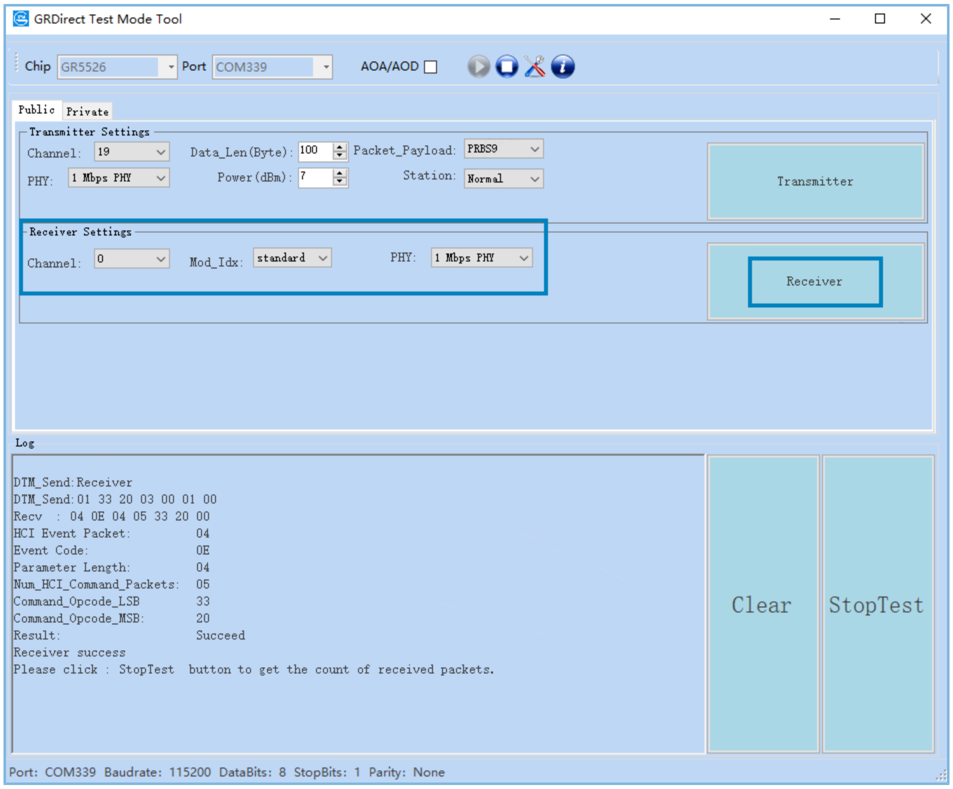


Figure 5-6 DTM RX test interface of DTMTool with **AOA/AOD** unchecked

- ◦ **Channel**: channel No.; options: Channel 0–Channel 39; packet reception frequency: 2402 + Channel x 2 (MHz)
- ◦ **Mod_Idx**: modulation index. Only standard modulation is supported.

You can initiate a DTM RX test without AoA/AoD by using the LE Receiver Test Command V2. For command details, see "Section 8.4 LE Receiver Test Command V2".

- When **AOA/AOD** is checked:

  The corresponding DTM RX test interface showing RX parameter setting, RX command, and response is as follows:

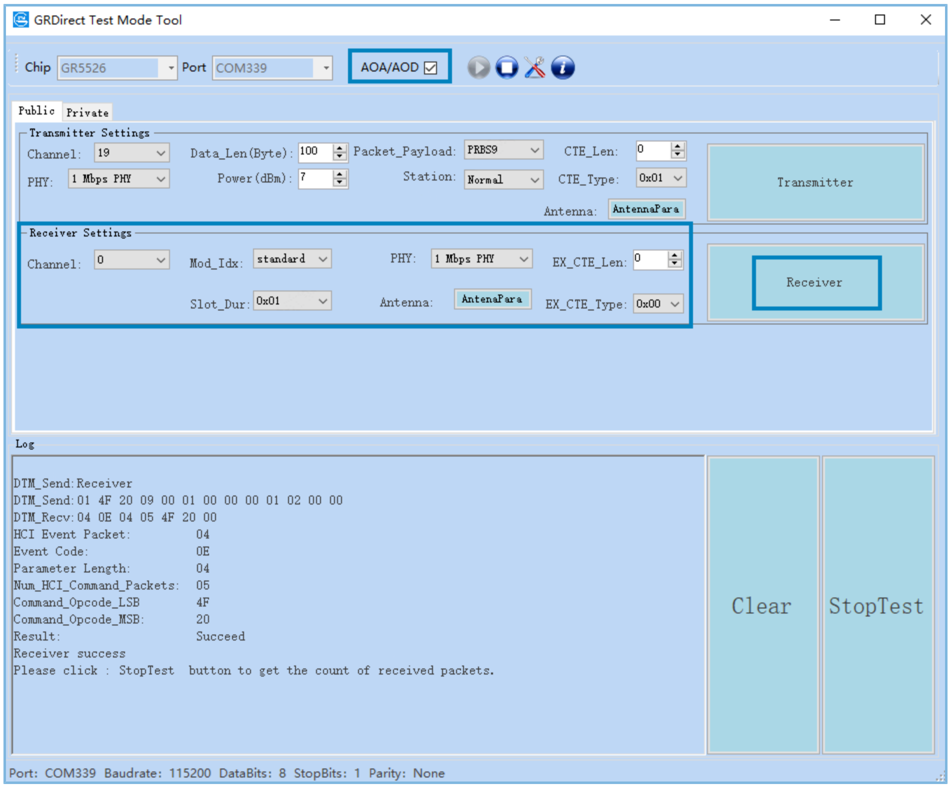


Figure 5-7 DTM RX test interface of DTMTool with **AOA/AOD** checked

You can initiate a DTM RX test with AoA/AoD by using the LE Receiver Test Command V3. For command details, see "Section 8.5 LE Receiver Test Command V3".

5.3.2 Ending a DTM RX Test

After a DTM RX test completes, click **StopTest** to end the test.

In RX test mode, the number of packets received in the return event is equal to the actual number of packets received.

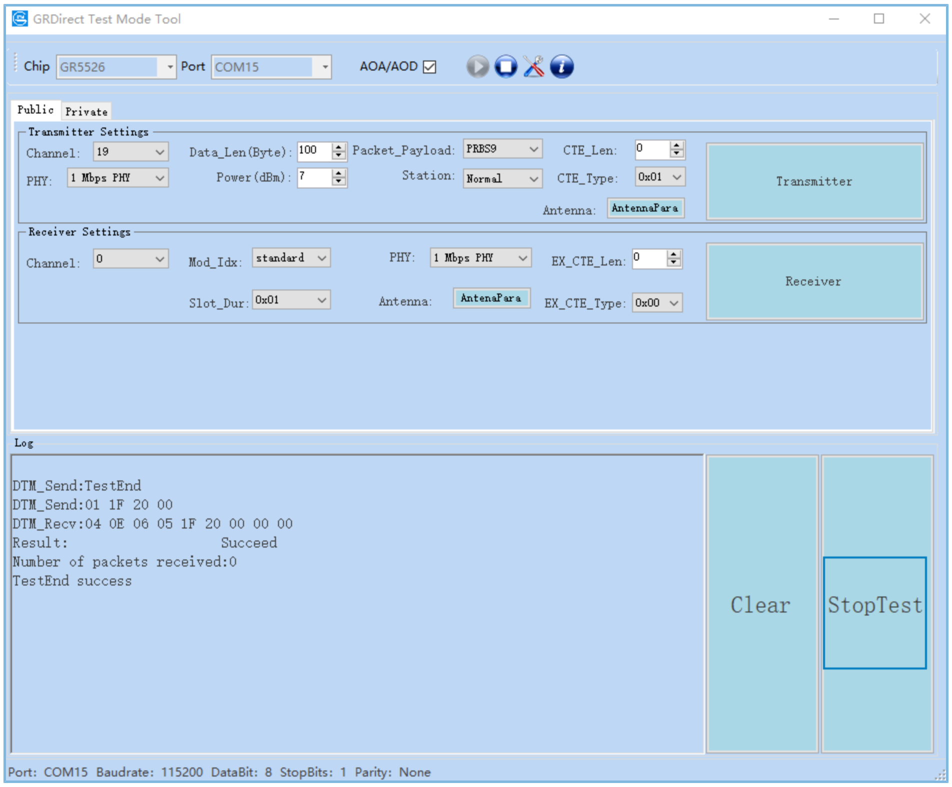


Figure 5-8 To end a DTM RX test

You can end a DTM RX test by using the LE Test End Command. For command details, see "Section 8.6 LE Test End Command".

5.4 Private Commands

To simplify DTM test for users, Goodix provides customized private HCI commands in the GR5xx DMT project. These customized commands are not standard HCI commands defined in *Bluetooth Core Spec*.

5.4.1 Single Carrier Wave Test

In the **Single_Wave settings** area under the **Private** tab, set **Channel** and **Power (dBm)**, and click **Start Single Wave**, to start a single carrier wave test. Then GR5xx will generate single carrier wave data packets for the specified channel.

Click **Stop Single Wave** to stop the test.

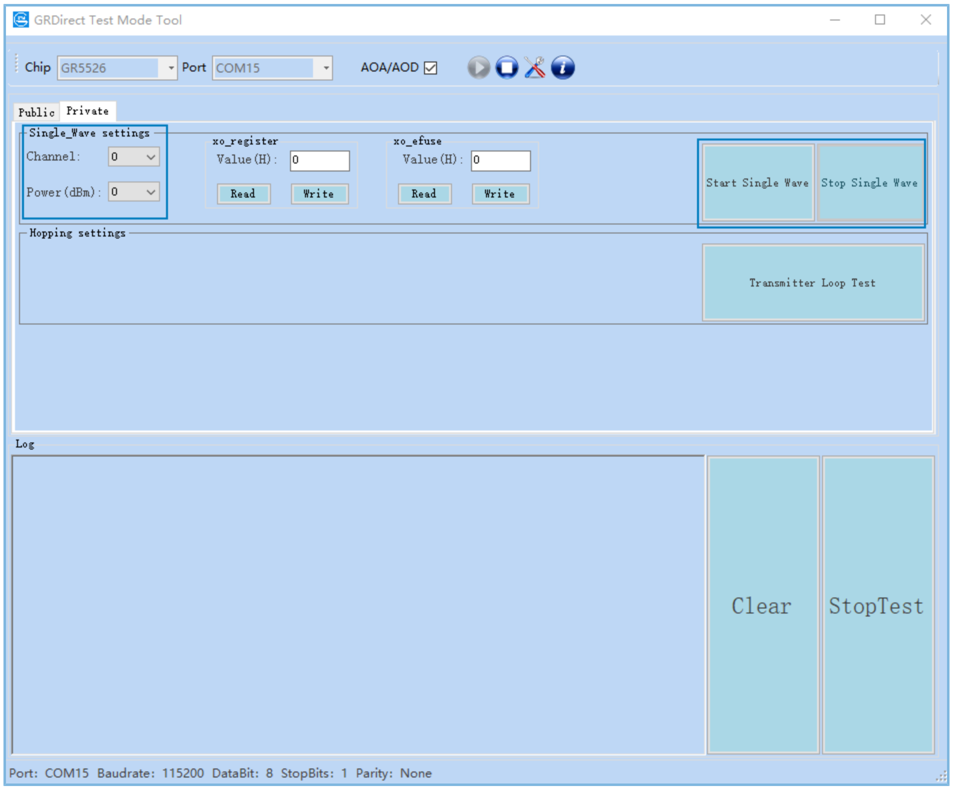


Figure 5-9 Single carrier wave test

You can start/stop a single carrier wave test by using the LE Start/Stop Signal Carrier Wave Command. For command details, see "Section 8.7 LE Start Signal Carrier Wave Command".

Note:

End an ongoing DTM test task (if any) before starting a single carrier wave test.

5.4.2 Writing and Reading a Register XO Value

To write an XO value to the corresponding register of a specific GR5xx SoC, input an XO value in the field of **Value (H)** in the **xo_register** area under the **Private** tab, and click **Write**.

To read the XO value from the register, click **Read**. The read value then will be displayed in the field of **Value (H)**.

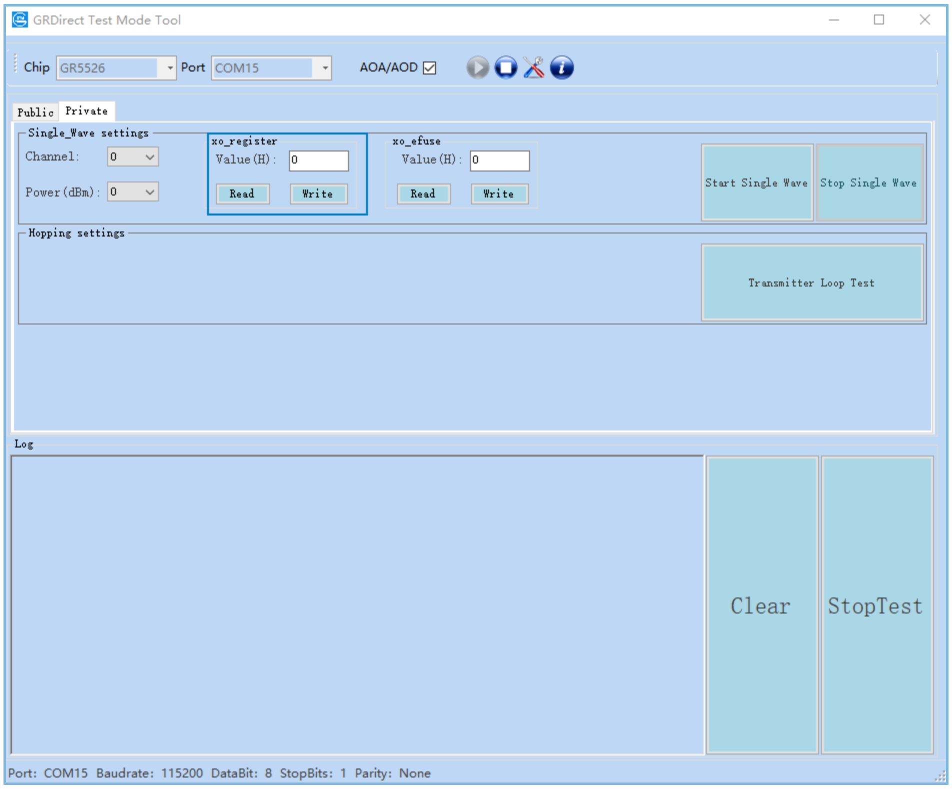


Figure 5-10 To write/read a register XO value

You can write/read a register XO value by using the Write/Read Register XO Command. For command details, see "Section 8.10 Write Register XO Command" and "Section 8.11 Read Register XO Command".

5.4.3 Writing and Reading an eFuse XO Value

To write an XO value in eFuse of a specific GR5xx SoC, input an XO value in the field of **Value (H)** in the **xo_efuse** area under the **Private** tab, and click **Write**.

Note:

eFuse can only be written once and cannot be erased. The settings will take effect only after system reboot.

To read the XO value from eFuse, click **Read**. The read value then will be displayed in the field of **Value (H)**.

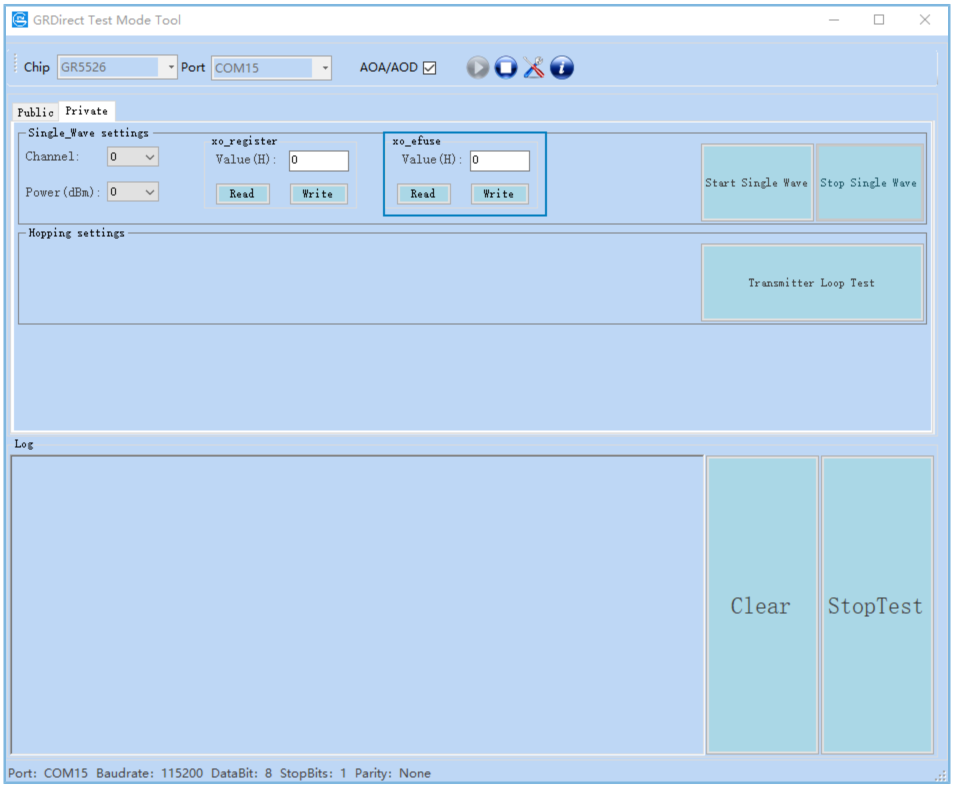


Figure 5-11 To write/read an eFuse XO value

You can write/read an eFuse XO value by using the Write/Read eFuse XO Command. For command details, see "Section 8.12 Write eFuse XO Command" and "Section 8.13 Read eFuse XO Command".

6 Test Considerations

6.1 General Considerations

1. See *RF-PHY.TS.pdf* (available at www.bluetooth.com) for test standards.
2. Take cable loss into account in results of a transmission power test.

6.2 GR533x DTM Project Configurations

To perform DTM testing on the GR533x device, it is necessary to configure the relevant parameters in the application project to align with the specific RF PA application scenario. This involves selecting the appropriate chip series, PA type, and power mode.

Table 6-1 Application project configuration parameters

PA Type	HPA	SPA		UPA
SoC Series	GR5332	GR5332/ GR5331/GR5330		GR5331/GR5330
Power Mode	SYS_LDO	DC-DC/SYS_LDO		DC-DC/SYS_LDO
Supply Voltage (V)	1.15	1.05	1.15	1.05
Power Range (dBm)	[-10, 15]	GR5332: [-20, 4] GR5331/GR5330: [-20, 5]	GR5332: [-20, 5] GR5331/GR5330: [-20, 6]	[-15, 2]

Tip:

You can configure related parameters in *custom_config.h* of the application project.

- **SoC series configuration**

  Configure the **CHIP_TYPE** macro according to the specific SoC in use.

```
// <o> Chip version
#ifndef SOC_GR533X
#define SOC_GR533X
#endif
// <o> Select chip type
// <0=> GR5330AENI
// <1=> GR5331AENI
// <2=> GR5331CENI
// <3=> GR5332AENE
// <4=> GR5332CENE
#ifndef CHIP_TYPE
#define CHIP_TYPE                        2
#endif
```

- **PA configuration**

  Configure the **RF_TX_PA_SELECT** macro to select an appropriate PA according to the TX output power requirements in applications.

```
// <o> RF TX PA select
```



```
// <1=> BLE_RF_TX_MODE_SPA_MODE  (-20~6 dBm TX power for GR5331/GR5330, -20~5 dBm TX power
 for GR5332)
// <2=> BLE_RF_TX_MODE_UPA_MODE  (-15~2 dBm TX power for GR5331/GR5330)
// <3=> BLE_RF_TX_MODE_HPA_MODE  (-10~15 dBm TX power for GR5332)
#ifndef RF_TX_PA_SELECT
#define RF_TX_PA_SELECT                  1
#endif
```

- **System power configuration**

  Configure the **SYSTEM_POWER_MODE** macro according to the specific system power mode in use.

```
// <o1.0..2> System power mode
// <0=> DCDC MODE
// <1=> SYSLDO MODE
#ifndef SYSTEM_POWER_MODE
#define SYSTEM_POWER_MODE                0
#endif
```

Note:

When an **HPA** is adopted in applications, the system power mode must be switched to **SYS_LDO**.

---

7 FAQ

This chapter describes possible problems, reasons, and solutions during the DTM test.

7.1 Why Do I Fail to Deliver Commands and Encounter Invalid Commands?

- Description

  Developers fail to deliver commands when using the GRUart serial port debugging tool or testers. Or the delivered commands are invalid.

- Analysis

  An error occurs in configurations of the GRUart serial port debugging tool, testers, or the serial cable.

- Solution

  Check whether the UART port on the DUT matches to the UART pin in the DTM firmware. Fore details, see "Section 4.2 Configuring UART Ports".

  If the UART pin is correctly used, check the configurations (baud rate, stop bit, and data bit) on GRUart and testers. Check the RX and TX pins are correctly connected to each other.

7.2 Why Do I Fail the Test with DTMTool?

- Description

  The test with DTMTool fails.

- Analysis

  The failure is caused by executing commands in a wrong order.

- Solution

  You can check the specific command that failed to execute in the **Log** pane and the reason for the failure in the return event of that command.

Note:

Before executing a new command, you need to end the ongoing command (if any).

---

8 Appendix: Common HCI Commands in RF Performance Test

8.1 LE Reset Command

Table 8-1 Command format

Byte Description	Value
HCI Command Packet	0x01
Command Opcode LSB	0x03
Command Opcode MSB	0x0C
Parameter Length	0x00

Table 8-2 Returned events

Byte Description	Value
HCI Event Packet	0x04
Event Code	0x0E
Parameter Length	0x04
Num HCI Command Packets	0x05
Command Opcode LSB	0x03
Command Opcode MSB	0x0C
Status	• 0x00: Command succeeded. • 0x01 to 0xFF: Command failed. See "Volume 2, Part D: Error Codes" in *Bluetooth Core Spec* for a list of error code and descriptions.

8.2 LE Transmitter Test Command V2

Table 8-3 Command format

Byte Description	Value
HCI Command Packet	0x01
Command Opcode LSB	0x34
Command Opcode MSB	0x20
Parameter Length	0x04
Channel	= (F – 2402)/2, where F ranges from 2402 MHz to 2480 MHz. Range: 0x00 to 0x27
Data Length	0x01 to 0xFF: Length in bytes of payload data in each packet.
Packet Payload	• 0x00: Pseudorandom binary sequence 9 • 0x01: Pattern of alternating bits '11110000' • 0x02: Pattern of alternating bits '10101010'

Byte Description	Value
	• 0x03: Pseudorandom binary sequence 15 • 0x04: Pattern of All '1' bits • 0x05: Pattern of All '0' bits • 0x06: Pattern of alternating bits '00001111' • 0x07: Pattern of alternating bits '0101'
PHY	• 0x01: Transmitter adopts the LE 1M PHY. • 0x02: Transmitter adopts the LE 2M PHY. • 0x03: Transmitter adopts the LE Coded PHY with S = 8 data coding. • 0x04: Transmitter adopts the LE Coded PHY with S = 2 data coding.

Table 8-4 Returned events

Byte Description	Value
HCI Event Packet	0x04
Event Code	0x0E
Parameter Length	0x04
Num HCI Command Packets	0x05
Command Opcode LSB	0x34
Command Opcode MSB	0x20
Status	• 0x00: Command succeeded. • 0x01 to 0xFF: Command failed. See "Volume 2, Part D: Error Codes" in *Bluetooth Core Spec* for a list of error code and descriptions.

8.3 LE Transmitter Test Command V4

Table 8-5 Command format

Byte Description	Value
HCI Command Packet	0x01
Command Opcode LSB	0x7B
Command Opcode MSB	0x20
Parameter Length	0x0A
Channel	= (F – 2402)/2, where F ranges from 2402 MHz to 2480 MHz. Range: 0x00 to 0x27
Data Length	0x01 to 0xFF: Length in bytes of payload data in each packet.
Packet Payload	• 0x00: Pesudo-Random bit sequence 9 • 0x01: Pattern of alternating bits '11110000' • 0x02: Pattern of alternating bits '10101010'

Byte Description	Value
	• 0x03: Pesudo-Random bit sequence 15 • 0x04: Pattern of All '1' bits • 0x05: Pattern of All '0' bits • 0x06: Pattern of alternating bits '00001111' • 0x07: Pattern of alternating bits '0101'
PHY	• 0x01: Transmitter adopts the LE 1M PHY. • 0x02: Transmitter adopts the LE 2M PHY. • 0x03: Transmitter adopts the LE Coded PHY with S = 8 data coding. • 0x04: Transmitter adopts the LE Coded PHY with S = 2 data coding.
CTE Length	• 0x00: Do not transmit a Constant Tone Extension. • 0x02 to 0x14: Length of the Constant Tone Extension in units of 8 µs • All other values: Reserved for future use
CTE Type	• 0x00: AoA Constant Tone Extension • 0x01: AoD Constant Tone Extension with 1 µs slots • 0x02: AoD Constant Tone Extension with 2 µs slots • All other values: Reserved for future use
Switching Pattern Length	• 0x02 to 0x4B: The number of antenna IDs in the pattern • All other values: Reserved for future use
Antenna IDs[i]	0xXX: Antenna ID in the pattern
TX Power Level	• 0xXX: Set the transmitter to the specified or the nearest transmit power level. Range: -127 to +20; unit: dBm • 0x7E: Set the transmitter to the minimum transmit power level. • 0x7F: Set the transmitter to the maximum transmit power level.

Table 8-6 Returned events

Byte Description	Value
HCI Event Packet	0x04
Event Code	0x0E
Parameter Length	0x04
Num HCI Command Packets	0x05
Command Opcode LSB	0x7B
Command Opcode MSB	0x20
Status	• 0x00: Command succeeded. • 0x01 to 0xFF: Command failed. See "Volume 2, Part D: Error Codes" in *Bluetooth Core Spec* for a list of error code and descriptions.

8.4 LE Receiver Test Command V2

Table 8-7 Command format

Byte Description	Value
HCI Command Packet	0x01
Command Opcode LSB	0x33
Command Opcode MSB	0x20
Parameter Length	0x03
Channel	= (F – 2402)/2, where F ranges from 2402 MHz to 2480 MHz. Range: 0x00 to 0x27
PHY	• 0x01: Receiver adopts the LE 1M PHY. • 0x02: Receiver adopts the LE 2M PHY. • 0x03: Receiver adopts the LE Coded PHY.
Modulation Index	• 0x00: Assume the transmitter will have a standard modulation index. • 0x01: Assume the transmitter will have a stable modulation index.

Table 8-8 Returned events

Byte Description	Value
HCI Event Packet	0x04
Event Code	0x0E
Parameter Length	0x04
Num HCI Command Packets	0x05
Command Opcode LSB	0x33
Command Opcode MSB	0x20
Status	• 0x00: Command succeeded. • 0x01 to 0xFF: Command failed. See "Volume 2, Part D: Error Codes" in *Bluetooth Core Spec* for a list of error code and descriptions.

8.5 LE Receiver Test Command V3

Table 8-9 Command format

Byte Description	Value
HCI Command Packet	0x01
Command Opcode LSB	0x4F
Command Opcode MSB	0x20
Parameter Length	0x09
Channel	= (F – 2402)/2, where F ranges from 2402 MHz to 2480 MHz.

Byte Description	Value
	Range: 0x00 to 0x27
PHY	• 0x01: Receiver adopts the LE 1M PHY. • 0x02: Receiver adopts the LE 2M PHY. • 0x03: Receiver adopts the LE Coded PHY.
Modulation Index	• 0x00: Assume the transmitter will have a standard modulation index. • 0x01: Assume the transmitter will have a stable modulation index.
Expected CTE Length	• 0x00: No Constant Tone Extension expected (default) • 0x02 to 0x14: Expected length of the Constant Tone Extension in units of 8 µs • All other values: Reserved for future use
Expected CTE Type	• 0x00: Expected AoA Constant Tone Extension • 0x01: Expected AoD Constant Tone Extension with 1 µs slots • 0x02: Expected AoD Constant Tone Extension with 2 µs slots • All other values: Reserved for future use
Slot Durations	• 0x01: Switching and sampling slots are 1 µs each. • 0x02: Switching and sampling slots are 2 µs each. • All other values: Reserved for future use
Switching_Pattern_Length	• 0x02 to 0x4B: The number of antenna IDs in the pattern • All other values: Reserved for future use
Antenna IDs[i]	0xXX: Antenna ID in the pattern

Table 8-10 Returned events

Byte Description	Value
HCI Event Packet	0x04
Event Code	0x0E
Parameter Length	0x04
Num HCI Command Packets	0x05
Command Opcode LSB	0x4F
Command Opcode MSB	0x20
Status	• 0x00: Command succeeded. • 0x01 to 0xFF: Command failed. See "Volume 2, Part D: Error Codes" in *Bluetooth Core Spec* for a list of error code and descriptions.

8.6 LE Test End Command

Table 8-11 Command format

Byte Description	Value
HCI Command Packet	0x01
Command Opcode LSB	0x1F
Command Opcode MSB	0x20
Parameter Length	0x00

Table 8-12 Returned events

Byte Description	Value
HCI Event Packet	0x04
Event Code	0x0E
Parameter Length	0x06
Num HCI Command Packets	0x05
Command Opcode LSB	0x1F
Command Opcode MSB	0x20
Status	• 0x00: Command succeeded. • 0x01 to 0xFF: Command failed. See “Volume 2, Part D: Error Codes” in *Bluetooth Core Spec* for a list of error code and descriptions.
Number of Packets Received (LSB)	0xXX
Number of Packets Received (MSB)	0xXX

8.7 LE Start Signal Carrier Wave Command

Table 8-13 Command format

Byte Description	Value
HCI Command Packet	0x01
Command Opcode LSB	0x31
Command Opcode MSB	0xFC
Parameter Length	0x06
Reserved	0xFFFF
Private Command	0x0001: Start a single carrier wave test.
Frequency	= (F – 2402)/2, where F ranges from 2402 MHz to 2480 MHz. Range: 0x0000–0x0027
TX Power	Range: 0x000 dBm–0x07 dBm (GR533x supports 0x0000 dBm–0x000f dBm.)

Table 8-14 Returned events

Byte Description	Value
HCI Event Packet	0x04
Event Code	0x0E
Parameter Length	0x06
Num HCI Command Packets	0x05
Command Opcode LSB	0x31
Command Opcode MSB	0xFC
Status	• 0x00: Command succeeded. • 0x01 to 0xFF: Command failed. See "Volume 2, Part D: Error Codes" in *Bluetooth Core Spec* for a list of error code and descriptions.
Reserved	0xFFFF

8.8 LE Stop Signal Carrier Wave Command

Table 8-15 Command format

Byte Description	Value
HCI Command Packet	0x01
Command Opcode LSB	0x31
Command Opcode MSB	0xFC
Parameter Length	0x06
Reserved	0xFFFF
Private Command	0x0009: Stop a single carrier wave test.
Reserved	0x0000

Table 8-16 Returned events

Byte Description	Value
HCI Event Packet	0x04
Event Code	0x0E
Parameter Length	0x06
Num HCI Command Packets	0x05
Command Opcode LSB	0x31
Command Opcode MSB	0xFC
Status	• 0x00: Command succeeded. • 0x01 to 0xFF: Command failed. See "Volume 2, Part D: Error Codes" in *Bluetooth Core Spec* for a list of error code and descriptions.

Byte Description	Value
Reserved	0xFFFF

8.9 LE Modify TX Power Command

Table 8-17 Command format

Byte Description	Value
HCI Command Packet	0x01
Command Opcode LSB	0x31
Command Opcode MSB	0xFC
Parameter Length	0x06
Reserved	0xFFFF
Private Command	0x0002: Modify TX power.
TX Power	Range: 0x0000 dBm–0x0007 dBm (GR533x supports 0x0000 dBm–0x000f dBm.)

Table 8-18 Returned events

Byte Description	Value
HCI Event Packet	0x04
Event Code	0x0E
Parameter Length	0x06
Num HCI Command Packets	0x05
Command Opcode LSB	0x31
Command Opcode MSB	0xFC
Status	• 0x00: Command succeeded. • 0x01 to 0xFF: Command failed. See "Volume 2, Part D: Error Codes" in *Bluetooth Core Spec* for a list of error codes and descriptions.
Reserved	0xFFFF

8.10 Write Register XO Command

Table 8-19 Command format

Byte Description	Value
HCI Command Packet	0x01
Command Opcode LSB	0x31
Command Opcode MSB	0xFC
Parameter Length	0x06
Reserved	0xFFFF

Byte Description	Value
Private Command	0x0003: Write the register XO.
XO	0xXXXX: XO value

Table 8-20 Returned events

Byte Description	Value
HCI Event Packet	0x04
Event Code	0x0E
Parameter Length	0x06
Num HCI Command Packets	0x05
Command Opcode LSB	0x31
Command Opcode MSB	0xFC
Status	• 0x00: Command succeeded. • 0x01 to 0xFF: Command failed. See "Volume 2, Part D: Error Codes" in *Bluetooth Core Spec* for a list of error code and descriptions.
Reserved	0xFFFF

8.11 Read Register XO Command

Table 8-21 Command format

Byte Description	Value
HCI Command Packet	0x01
Command Opcode LSB	0x31
Command Opcode MSB	0xFC
Parameter Length	0x06
Reserved	0xFFFF
Private Command	0x0004: Read the register XO
Reserved	0xFFFF

Table 8-22 Returned events

Byte Description	Value
HCI Event Packet	0x04
Event Code	0x0E
Parameter Length	0x06
Num HCI Command Packets	0x05
Command Opcode LSB	0x31

Byte Description	Value
Command Opcode MSB	0xFC
Status	• 0x00: Command succeeded. • 0x01 to 0xFF: Command failed. See "Volume 2, Part D: Error Codes" in *Bluetooth Core Spec* for a list of error code and descriptions.
XO Value	0xXXXX: Returned XO value in the register

8.12 Write eFuse XO Command

Table 8-23 Command format

Byte Description	Value
HCI Command Packet	0x01
Command Opcode LSB	0x31
Command Opcode MSB	0xFC
Parameter Length	0x06
Reserved	0xFFFF
Private Command	0x0005: Write the eFuse XO.
XO	0xXXXX: XO value

Table 8-24 Returned events

Byte Description	Value
HCI Event Packet	0x04
Event Code	0x0E
Parameter Length	0x06
Num HCI Command Packets	0x05
Command Opcode LSB	0x31
Command Opcode MSB	0xFC
Status	• 0x00: Command succeeded. • 0x01 to 0xFF: Command failed. See "Volume 2, Part D: Error Codes" in *Bluetooth Core Spec* for a list of error code and descriptions.
Reserved	0xFFFF

8.13 Read eFuse XO Command

Table 8-25 Command format

Byte Description	Value
HCI Command Packet	0x01

Byte Description	Value
Command Opcode LSB	0x31
Command Opcode MSB	0xFC
Parameter Length	0x06
Reserved	0xFFFF
Private Command	0x0006: Read the eFuse XO.
Reserved	0xFFFF

Table 8-26 Returned events

Byte Description	Value
HCI Event Packet	0x04
Event Code	0x0E
Parameter Length	0x06
Num HCI Command Packets	0x05
Command Opcode LSB	0x31
Command Opcode MSB	0xFC
Status	• 0x00: Command succeeded. • 0x01 to 0xFF: Command failed. See "Volume 2, Part D: Error Codes" in *Bluetooth Core Spec* for a list of error code and descriptions.
XO value	0xXXXX: Returned XO value in the eFuse

