# Hi3861 V100 / Hi3861L V100 Production Line Equipment Test

# User Guide

**Issue** 01

**Date** 2020-04-30

## HiSilicon (Shanghai) Technologies Co., Ltd.

| | |
|---|---|
| Address: | New R&D Center, 49 Wuhe Road,<br>Bantian, Longgang District,<br>Shenzhen 518129 P. R. China |
| Website: | https://www.hisilicon.com/en/ |
| Email: | support@hisilicon.com |

# About This Document

## Purpose

This document describes the board manufacturing test solution of Hi3861/Hi3861L, including software loading, eFUSE data loading, test items, and test methods. The test methods are closely related to the signal strength. Some general test items are not described in detail.

This document provides guide on how to design equipment tests during the production line and at factory.

## Related Versions

The following table lists the product versions related to this document.

| Product Name | Version |
| --- | --- |
| Hi3861 | V100 |
| Hi3861L | V100 |

## Intended Audience

The document is intended for:

- Technical support engineers
- PCB hardware development engineers
- Software development engineers

## Symbol Conventions

The symbols that may be found in this document are defined as follows.

| Symbol | Description |
| --- | --- |
| DANGER | Indicates a hazard with a high level of risk which, if not avoided, will result in death or serious injury. |
| WARNING | Indicates a hazard with a medium level of risk which, if not avoided, could result in death or serious injury. |
| CAUTION | Indicates a hazard with a low level of risk which, if not avoided, could result in minor or moderate injury. |
| NOTICE | Indicates a potentially hazardous situation which, if not avoided, could result in equipment damage, data loss, performance deterioration, or unanticipated results.<br>NOTICE is used to address practices not related to personal injury. |
| NOTE | Supplements the important information in the main text.<br>NOTE is used to address information not related to personal injury, equipment damage, and environment deterioration. |

# Change History

| Issue | Date | Change Description |
|---|---|---|
| 01 | 2020-04-30 | This issue is the first official release.<br>● Updated the description of **menuconfig** in CE and FCC versions in **1.1 Overview**.<br>● Updated the NV ID in the **mss_nvi_db.xml** file in **1.2 Configuring the Power**. Update **Figure 1-2**. Added **Figure 1-3**. Updated the example description about the assumption that the power corresponding to 11g 24 Mbps needs to be increased to 18 dBm.<br>● Updated the NV ID in the **mss_nvi_db.xml** file in **1.3 Configuring the Frequency Offset and Band Power Offset**. Added **Figure 1-4**. Added the description that negative numbers are configured by using two's complement.<br>● Added **1.4 Configuring RF PLL Parameters**.<br>● Updated the NV ID in the **mss_nvi_db.xml** file in **Step 3** of **1.5 Obtaining the Factory Compensation Value**.<br>● Updated steps 3 and 4 in section 2.1.<br>● Updated **Software Preparation** in **2.2.1 Preparing Software/Hardware**.<br>● Added the description of the directory for generating the **efuse_cfg.bin** file to **Step 2** in **2.2.2 Making an eFUSE Image**. Added a NOTICE.<br>● Updated **2.2.3 Creating a Factory Test Image**.<br>● Updated the description of **2.2.4 Burning Images by Using HiBurn.exe** and **Table 2-1**. Added a NOITCE.<br>● Updated the description of writing the eFUSE command in **Step 8** of **3.2.2 Process Details**. Deleted the description of step 10 for ending the factory test mode.<br>● Added the description of the MAC addresses of the STA and SoftAP in **3.2.4 MAC Address Write**.<br>● Added examples in the description of test commands in **3.2.5 Test Commands**. Updated the command description for frequency offset compensation at room temperature. Deleted the description of the command for ending the factory test mode.<br>● Added the description of the RX performance test scheme in **3.3.1 Test Items**.<br>● Updated the always TX commands in **Step 1** of **3.3.3 Signal Strength Test**.<br>● Added **3.3.4 RX Performance Test**.<br>● Updated the description of test commands related to the signal strength test in **3.3.6 Test Commands**. Added the |

| Issue | Date | Change Description |
| --- | --- | --- |
| | | description of the test commands related to the RX performance test.<br>• Updated the command for switching the service mode in **3.4 Switch a Factory Test BIN to a Service BIN**. Added the description of deleting the factory test binary file. Added a NOICTE that the binary file must be erased during product commercial use. |
| 00B08 | 2020-04-21 | • Added **1 Software Version Configuration Preparation**.<br>• Added section 2.1.<br>• Added **3.4 Switch a Factory Test BIN to a Service BIN**. |
| 00B07 | 2020-04-07 | • Updated the description of the 2048-bit eFUSE in the board test in **2.2.2 Making an eFUSE Image**.<br>• Updated the enumerated values in step 2 of method 1 in **2.2.3 Creating a Factory Test Image**. Added the description of method 2.<br>• Updated **Figure 3-3** in **3.2.1 Flowchart**.<br>• Added the description of step 10 for ending the factory test mode in **3.2.2 Process Details**.<br>• Updated the AT command for writing a MAC value into the eFUSE in **3.2.5 Test Commands**. Added the description of the AT command for stopping the factory test mode.<br>• Deleted the description of the test item for querying the version number from **3.3.1 Test Items**.<br>• Deleted the procedure for reading the module version number from **3.3.2 Process Details**.<br>• Deleted the description of the test command for reading the version number from **3.3.6 Test Commands**. |
| 00B06 | 2020-03-25 | In **3.2.5 Test Commands**, updated the description of the **<control>** parameter in the test command for enabling always TX; updated the description of the test command for disabling always TX; updated the **type** parameter in the test command for writing a calibration value to the eFUSE; updated the **mac** and **type** parameters in the test command for writing a calibration value to the eFUSE. |

| Issue | Date | Change Description |
|---|---|---|
| 00B05 | 2020-03-19 | • In **2.2.4 Burning Images by Using HiBurn.exe**, added the description of the **-forceread:** command in **Table 2-1**.<br>• In **3.2.5 Test Commands**, added the description of the command and factory test result; updated the description of the **<control>** parameter in the test command for enabling always TX; updated the description of the test command for writing a calibration value to the eFUSE; updated the description of the test command for writing a MAC value to the eFUSE; updated the description of the test command for querying the factory test compensation data.<br>• In **3.3.6 Test Commands**, added the command description. |
| 00B04 | 2020-03-06 | In **3.2.5 Test Commands**, updated the description of the AT commands for the power compensation offset and power compensation at each rate. |
| 00B03 | 2020-02-26 | In **2.2.3 Creating a Factory Test Image**, updated the code sample of step 2. |
| 00B02 | 2020-02-12 | • In **2.2.2 Making an eFUSE Image**, updated the procedure for configuring the eFUSE area of the second type based on the board.<br>• In **Table 2-1** of **2.2.4 Burning Images by Using HiBurn.exe**, deleted the description of the **-loader** command and added the description of the **-2ms** command.<br>• Updated **Figure 3-3** in **3.2.1 Flowchart**.<br>• Added **Step 9** to **3.2.2 Process Details**.<br>• In **3.2.5 Test Commands**, updated the description of the commands for compensating the frequency offset at ambient temperature and writing a MAC value to the eFUSE; added the description of the commands for switching the AT command UART port and resetting the board.<br>• In **3.3.6 Test Commands**, updated the description of the command for reading the MAC address. |
| 00B01 | 2020-01-15 | This issue is the first draft release. |

# Contents



# 1 Software Version Configuration Preparation



## 1.1 Overview

The SDK supports the menuconfig of the CE and FCC versions. The configuration files of these versions are stored in the **tools\nvtool\xml_file** directory in the SDK.

- **mss_nvi_db.xml**: CE version (default)
- **mss_nvi_db_fcc.xml**: FCC version

**Figure 1-1** Configuration options for compilation

```
(Top)

    Target Chip  --->
    Security Settings  --->
    Factory Test Settings  --->
    BSP Settings  --->
    WiFi Settings  --->
    Third Party library  --->
    Lwip Settings  --->
    OTA Settings  --->
    Link Settings  --->
    Debug Log Settings  --->
```

```
(Top) → WiFi Settings

[ ] Enable WPS
    Authentication Option of Radio Parameters (CE Authentication)  --->
[ ] Enable MESH
```

```
(Top) → WiFi Settings → Authentication Option of Radio Parameters
                                                                  Main menu
(X) CE Authentication
( ) FCC Authentication
```

**NOTE**

This chapter uses the CE version as an example.

## 1.2 Configuring the Power

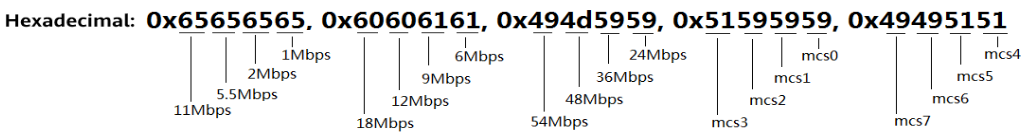
Open **tools\nvtool\xml_file\mss_nvi_db.xml** in the SDK and locate the configuration item whose NV ID is **0x80**, as shown in **Figure 1-2**.

**Figure 1-2** Example of the configuration item whose NV ID is 0x80 in the mss_nvi_db.xml file

```
<NV ID="0x80" NAME="INIT_CONFIG_PARAMS" PARAM_NAME="wal_cfg_params" PARAM_VALUE="{92,
0x0100FF00, 0x01000000, 0x424b, 32768, 0, -1, -1, -1, 115, 115, 115, 500, 17, 650, 13
```

The 8th to 12th elements in **PARAM_VALUE** are the configured power values of each rate, as shown in **Figure 1-3**.

**Figure 1-3** Example of the 8th to 12th elements in PARAM_VALUE




These bits correspond to the dbb_scale power configurations of [11b 1–11Mbps], [11g 6–18Mbps], [11g 24–54Mbps], [11n mcs0–3], and [11n mcs4–7], respectively. Each byte of each element value corresponds to a power configuration of a rate. For example, the first byte 0x59 of the tenth element value **0x494d5959** of **PARAM_VALUE** corresponds to the power configuration of 11g 24 Mbps, and the fourth byte 0x49 corresponds to the power configuration of 11g 54 Mbps.

The default RF power settings are used. For example, the default RF power of the current CE version is shown in **Table 1-1**.

**Table 1-1** Default RF power of CE version

| Protocol | Speed | 20 MHz Bandwidth |
|---|---|---|
| 802.11b | 1 Mbps | 16 |
| | 2 Mbps | 16 |
| | 5.5 Mbps | 16 |
| | 11 Mbps | 16 |
| 802.11g | 6 Mbps | 17 |
| | 9 Mbps | 17 |

| Protocol | Speed | 20 MHz Bandwidth |
|---|---|---|
| | 12 Mbps | 17 |
| | 18 Mbps | 17 |
| | 24 Mbps | 17 |
| | 36 Mbps | 17 |
| | 48 Mbps | 16 |
| | 54 Mbps | 16 |
| 802.11n | mcs0 | 16.5 |
| | mcs1 | 16.5 |
| | mcs2 | 16.5 |
| | mcs3 | 16.5 |
| | mcs4 | 16.5 |
| | mcs5 | 16.5 |
| | mcs6 | 16 |
| | mcs7 | 16 |

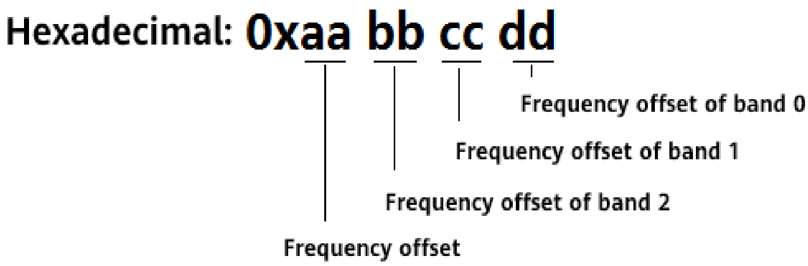
New dbb_scale = (10^((New target power – Old target power)/20)) x Old dbb_scale. **^** indicates the power operation, ***** indicates the multiplication operation, and **/** indicates the division operation.

Assume that you need to increase the power corresponding to 11g 24 Mbps to 18 dBm. The target power queried by **Table 1-1** is 17 dBm. Query the value of the tenth element of the NV item whose ID is **0x80** in the **mss_nvi_db.xml** file. If the value of dbb_scale is **0x59**, new dbb_scale = (10^((18 – 17)/20)) * 0x59, that is, **0x64**. The value of the tenth element corresponding to **PARAM_VALUE** is changed to **0x494d5964**. The method for modifying the power of other rates is similar. Save the modification, recompile the SDK, and reload the binary file to the board to make the configuration take effect.

## 1.3 Configuring the Frequency Offset and Band Power Offset

In the **tools\nvtool\xml_file\mss_nvi_db.xml** file of the SDK version, the 13th element in **NV ID="0x80"** corresponds to the frequency offset and the power offset of bands 0–2 (unit: 0.1 dB).

**Figure 1-4** Example of the 13th element in the NV ID="0x80" configuration item in the mss_nvi_db.xml file




In the figure:

- Bytes 1–3 correspond to the power offsets of bands 0–2 (band 0 corresponds to channels 1–4, band 1 corresponds to channels 5–9, and band 2 corresponds to channels 10–13 or 14).
- Byte 4: frequency offset

For example, the element value **0x0a00000b** indicates that the power offset of band 0 is 1.1 dB, the power offset of band 1 and band 2 is 0 dB, and the frequency offset is 10.

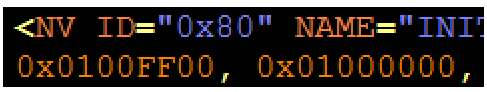
The negative number is configured in the form of two's complement (Two's complement of the configured value = 0x100 – Absolute value of the negative offset). For example, **0xf6000000** indicates that the frequency offset is –10.

# 1.4 Configuring RF PLL Parameters

The RF PLL may be affected by the crystal clock in some channels, causing EVM deterioration. In this case, you can adjust the register value to reduce the impact.

In the **tools\nvtool\xml_file\mss_nvi_db.xml** file of the SDK, the 14th and 15th elements of the PARAM_VALUE configuration item (**NV ID = "0x80"**) are used to adjust the RF PLL parameters of Hi3861 and Hi3861L, respectively. The value of each element starts from bit[0]. See **Figure 1-5**. Every two bits correspond to a channel (configurable range: 0x0–0x3).

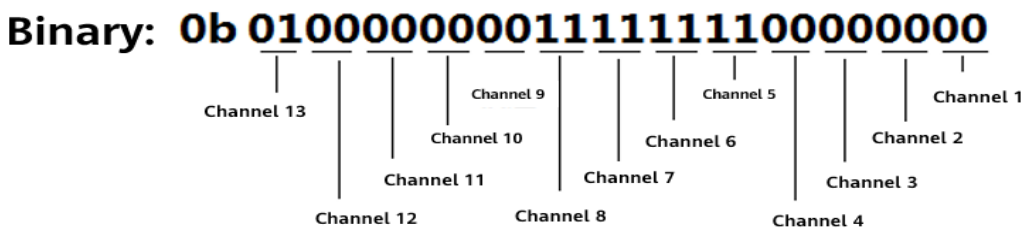
**Figure 1-5** Example of the values of the 14th and 15th elements in the PARAM_VALUE configuration item whose NV ID is 0x80 in the mss_nvi_db.xml file



```
<NV ID="0x80" NAME="INI
0x0100FF00, 0x01000000,
```

**Figure 1-6** Example of the values of the 14th and 15th elements in the PARAM_VALUE configuration item whose NV ID is 0x80 in the mss_nvi_db.xml file




The values of the 14th and 15th elements of **PARAM_VALUE** in **Figure 1-5** are described as follows:

- **0x0100FF00**: RF PLL parameter of Hi3861. For channels 5–8, the configured value is **0x3**; for channel 13, the configured value is **0x1**; for other channels, the configured value is **0x0**.
- **0x01000000**: RF PLL parameter of Hi3861L. For channel 13, the configured value is **0x1**; for other channels, the configured value is **0x0**.

Adjustment description: This parameter is optimized based on the HiSilicon version debugging. If the actual performance of the user board is slightly worse than that of other channels (for example, the EVM of channels ch5, ch6, ch7, chn8, and ch13 is worse than that of channel chn1) with the same target power, you can fine-tune this parameter to optimize the EVM. The adjustment method is to traverse the configured values from 0 to 3 to obtain the optimal configuration value of each channel and update the value to **mss_nvi_db.xml**.

## 1.5 Obtaining the Factory Compensation Value

For module users, to reconfigure **mss_nvi_db.xml** based on the module factory compensation value and compile the version, perform the following steps:

**Step 1** Obtain the module factory information from the module vendor. If the information is successfully obtained, go to **Step 3**.

**Step 2** Power on the module and run the **AT+RCALDATA** command to obtain the factory calibration parameters, as shown in **Figure 1-7**.

**Figure 1-7** Example of running the AT+RCALDATA command

```
AT+RCALDATA
+RCALDATA:Efuse cali chance(s) left:0 times.
+RCALDATA:freq_offset 10
+RCALDATA:band_pwr_offset_0 11
+RCALDATA:band_pwr_offset_1 0
+RCALDATA:band_pwr_offset_2 0
+RCALDATA:rate_pwr_offset_11n 0x0
+RCALDATA:rate_pwr_offset_11g 0x0
+RCALDATA:rate_pwr_offset_11b 0x0
+RCALDATA:dbb_scale_0 0x65656565
+RCALDATA:dbb_scale_1 0x60606161
+RCALDATA:dbb_scale_2 0x494d5959
+RCALDATA:dbb_scale_3 0x51595959
+RCALDATA:dbb_scale_4 0x49495151
+RCALDATA:freq_and_band_pwr_hybrid_offset 0x0a00000b
OK
```

- **dbb_scale_0** to **dbb_scale_4**: corresponds to the values of 8th to 12th elements in **PARAM_VALUE**.
- **freq_and_band_pwr_hybrid_offset**: corresponds to the value of the 13th element in **PARAM_VALUE**.

**Step 3** Set the factory calibration parameters to the configuration item whose NV ID is **0x80** in the **mss_nvi_db.xml** file and save the settings.

**Step 4** Recompile the SDK.

**----End**

# 2 Burning Solution for Software and Fixed eFuse



## 2.1 Software Loading Principle

Hi3861/Hi3861L has a built-in ROMBoot program. After the chip is powered on, the software program is loaded to the NOR flash zone over the UART*x* port with the support of ROMBoot.

Hardware requirements:

- UART*x* port for debugging
- Baud rate of the UART port: 115200–921600 bit/s. You are advised to set the baud rate to 921600 bit/s to improve efficiency.
- One-to-many burning tool
- Involved pins:
  - UARTx_TXD: output, TX end of the UART port
  - UARTx_RXD: input, RX end of the UART port

## 2.2 Software Burning Procedure

### 2.2.1 Preparing Software/Hardware

#### Software Preparation

- Obtain the **HiBurn.exe** burning software for the equipment test.
- Generate the **efuse_cfg.bin** file corresponding to the eFUSE area of the second type by referring to **2.2.2 Making an eFUSE Image**.
- Make the **Hixx_allinone.bin** image by referring to **2.2.3 Creating a Factory Test Image**.

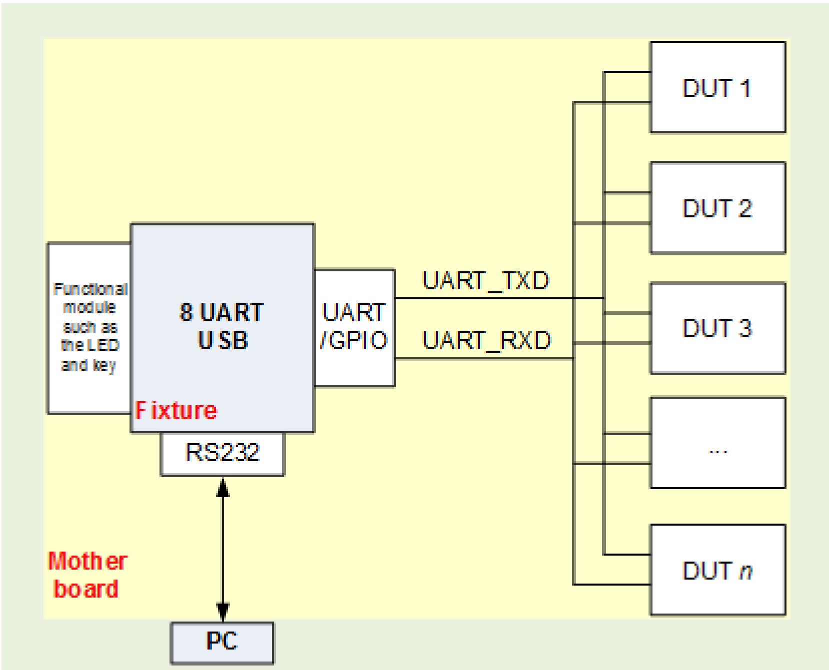
- Place **HiBurn.exe** and **Hixx_allinone.bin** in the corresponding folders and burn the images in HiBurn GUI or CLI mode.

### Hardware Preparation

If no one-to-many burning tool is available, connect the UART port of the PC to the equipment test board, power on the equipment test board, and determine the UART*x* port of the PC.

If a one-to-many burning tool is used (to burn multiple boards at the same time), connect the USB port to the main control PC, connect the UART port of the board to that of the one-to-many burning tool, and confirm the serial number of each UART port on the PC. After the connection is complete, power on the board.

**Figure 2-1** Hardware block diagram of the burning system




## 2.2.2 Making an eFUSE Image

NOTE

For details about the eFUSE configuration area, see the *Hi3861 V100/Hi3861L V100 eFUSE User Guide*.

The 2048-bit eFUSE is configured in the Hi3861/Hi3861L to store related configuration information. The 2048 bits are classified into three types: The first type is configured by the chip vendor before delivery, the second and third types are configured by the module vendor, and the third type is configured on the PC by running AT commands during the board test.

The eFUSE area of the second type needs to be configured based on the board to make the corresponding **efuse_cfg.bin** file.

**Step 1** Modify the content of the **efuse.csv** file in **tools\efuse_tool**, including:

- The first column "burn": **0** indicates data needs to be written, and **1** indicates no data needs to be written.
- The second column "name": indicates the eFUSE name.
- The third column "start_bit": indicates the start bit index of an eFUSE object.
- The fourth column "bit_width": indicates the length of an eFUSE object.
- The fifth column "value": indicates the value of eFUSE. The written data must be 32-bit aligned and in hexadecimal little-endian format.
- The sixth column "lock": indicates the name of a lock bit.

**Step 2** Run the **tools\efuse_tool\efuse_cfg_gen.py** script to generate the **efuse_cfg.bin** file in **tools\efuse_tool** as a part of the image to be burnt.

**----End**

**NOTICE**

When the functions that need the public key or private key to be written, such as secure boot, flash encryption and decryption, and TEE HUKS, are enabled, ensure that the eFUSE data is correctly written and the lock bit is configured. For example, when configuring root_key, you need to configure PG4. The eFUSE cannot be modified. The eFUSE lock is checked during flash encryption and decryption or in the TEE HUKS. If the eFUSE is not locked, the configuration is invalid.

## 2.2.3 Creating a Factory Test Image

**Step 1** Create the **basebin** folder in the **build** directory, and place the **efuse_cfg.bin** file created in **2.2.2 Making an eFUSE Image** to the **build/basebin** directory.

**Step 2** Run the **./build.sh menuconfig** command to open the build menu and locate **Factory Test Settings**.

```
    Target Chip  --->
    Security Settings  --->
    Factory Test Settings  --->
    BSP Settings  --->
    WiFi Settings  --->
    Third Party library  --->
    Lwip Settings  --->
    OTA Settings  --->
    Link Settings  --->
    Debug Log Settings  --->
```

**Step 3** Select **factory test enable** and press **S** to save the settings and exit.

```
(Top) → Factory Test Settings

[*] factory test enable
```

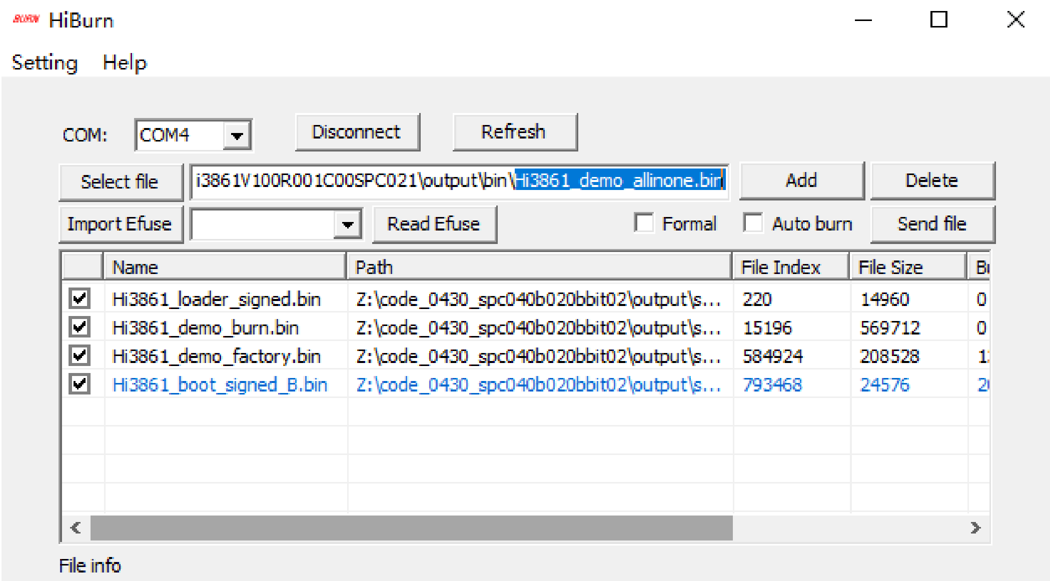
**Step 4** Run the **./build.sh all** command to build other items.

If you use **Makefile** for building, run the following command:

```
make clean;make factory;make clean_factory;make all
```

**Step 5** After the building is complete, open the generated **hixx_allinone.bin** file in the **output/bin/** directory of the SDK in HiBurn GUI mode. If **xxx_factory.bin** is displayed, the .bin file is compiled successfully. When the .bin file is burnt in HiBurn GUI mode, select all .bin files generated during building and burn them.




NOTE

- For details about the default flash partition locations of the factory test image and service image, see the description of the partition table in section "Flash Partitions and Protection" in the *Hi3861/Hi3861L V100 SDK Development Guide*.
- The content of **xx_burn.bin** can be customized by modifying the SDK script based on the application scenario. For example, you can specify that the content of **xx_burn.bin** contains only the image program but not the NV parameter. You can specify the burning address and size of **xx_burn.bin**. For details about SDK scripts, see **burn_bin_builder** (Sconstruct), **make_hbin** (make_upg_file.py), and **__main__** (pkt_builder.py). If the preceding files are modified by building **Makefile**, you need to set **PYTHON_SCRIPTS** to **n** in **build/sdk/make_scripts/config.mk**.

**----End**

## 2.2.4 Burning Images by Using HiBurn.exe

In addition to burning images on the GUI, **HiBurn.exe** can be invoked in CLI mode in the Windows environment. It can be integrated into the existing burning program of the factory production line. The invocation command is as follows:

```
HiBurn.exe params
```

Commands are separated by spaces. If a command contains parameters, the command is separated from the parameters by colons (:), for example:

```
HiBurn.exe -com:31 -bin:C:\test_bin\wifi\hixx_allinone.bin -signalbaud:921600
```

**Table 2-1** describes the parameters that must be configured in the **HiBurn.exe** burning command.

NOTICE

When the firmware encryption function is enabled (that is, **FLASH_ENCPY_CFG** in the eFUSE is set to **1**), the image program can be burnt for a maximum of six times.

**Table 2-1** Parameters of the HiBurn.exe burning command

| Command | Parameter | Description |
| --- | --- | --- |
| -com: | x | UART port number of the PC, for example, 0 |
| -bin: | path\hixx_allinone.bin | Absolute path of the **hixx_allinone.bin** file generated during software preparation. |
| -signalbaud: | 115200 | Baud rate of the serial port when **hixx_allinone.bin** is transferred in ROMBoot. The default value is 115200 bit/s. You are advised to set the baud rate to 921600 bit/s or higher based on the hardware support to improve the burning efficiency. |
| -2ms | None | Interruption packets sending at an interval of 2 ms. This parameter is used in fastboot scenarios. If this parameter is not specified, interruption packets are sent at an interval of 10 ms. |
| -forceread: | 10 | If this parameter is contained, the data reading of the UART port at an interval of 10 ms is enabled. Generally, this function does not need to be enabled. If HiBurn cannot be used in some PC environments, set this parameter. |

# 3 Board Test Solution



## 3.1 Test System Hardware

The tests are classified into the performance test during the production line and the functional test at factory, which have the same key equipment (**Table 3-1**) but different hardware diagrams of the test system (**Figure 3-1** and **Figure 3-2**).

**Table 3-1** Key equipment of a test system

| Name | Description and Requirements |
|---|---|
| PC | Controls the test process. It connects to the Wi-Fi communication tester through the Ethernet, connects to the test fixture through the RS232 port, and runs the test software on the PC to implement the overall equipment test function. |
| Barcode scanner | Each DUT has an independent MAC address. The unique MAC address code on a DUT is scanned to implement the addressing function of the DUT. (This is one of the methods for generating the MAC address code, and may also be implemented by other methods, for example, writing the unique address code in the eFUSE.) |

| Name | Description and Requirements |
| --- | --- |
| Test fixture (including the mother board and test fixture) | Carries power supplies for supply power to DUTs. The RS232 port connects the DUTs and the PC, and the RF cable connects the DUTs. |
| DUT | Indicates a device under test, which is connected to the equipment test system through the fixture for performing various specification tests. |
| Wi-Fi communication tester | Calibrates the power and frequency offset of a DUT. |

**Figure 3-1** Hardware diagram of a test system during the production line

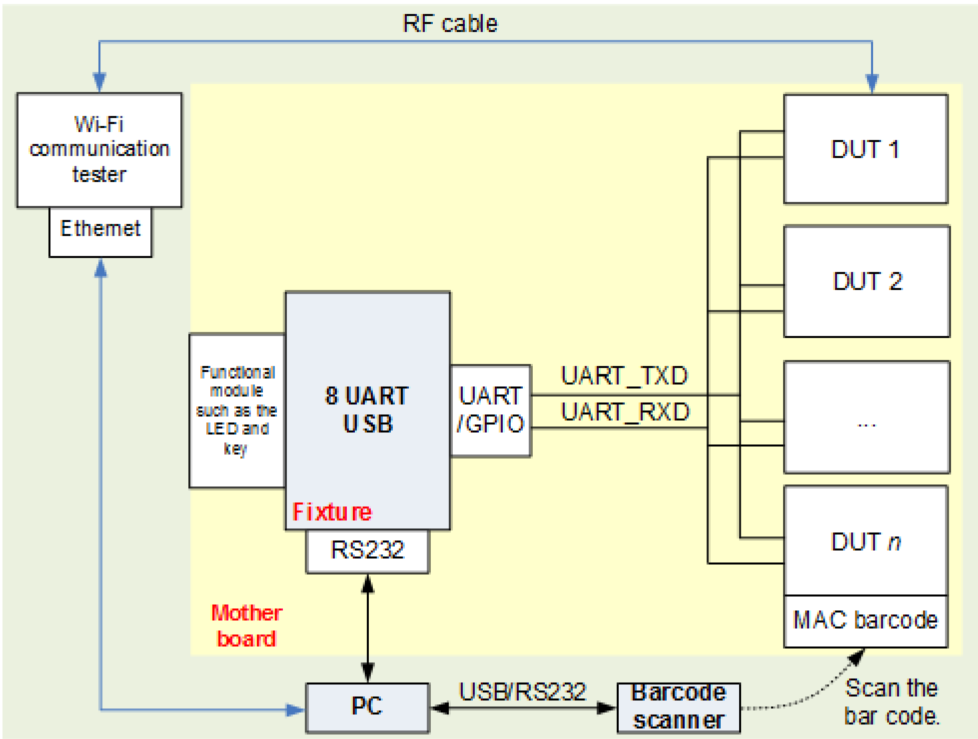
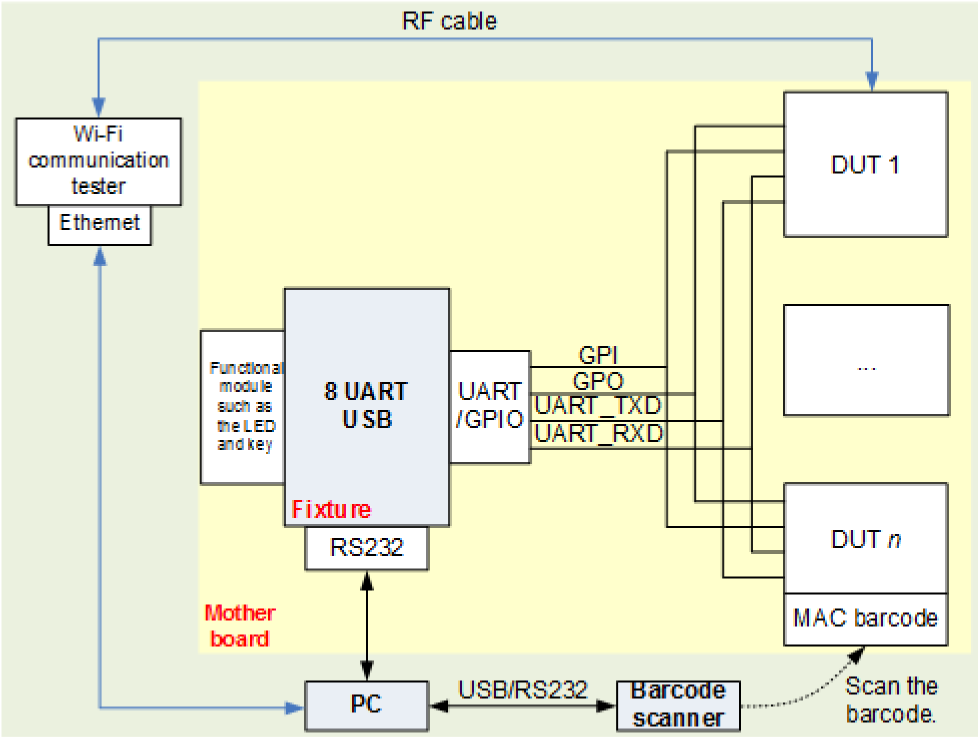


**Figure 3-2** Hardware diagram of a test system at factory

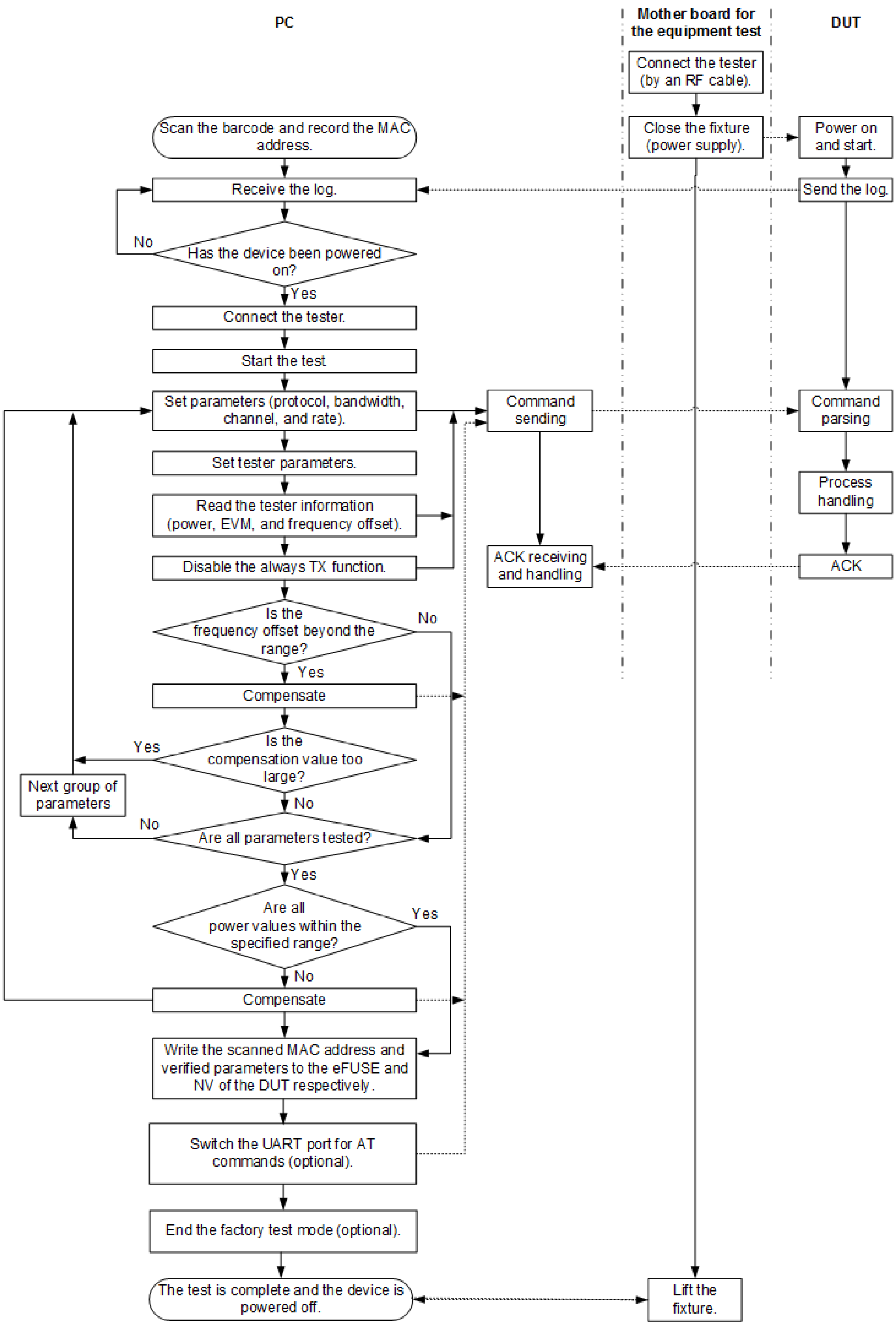



# 3.2 Performance Test

## 3.2.1 Flowchart

Figure 3-3 shows an example of the performance test process.

**Figure 3-3** Example of the performance test process




## 3.2.2 Process Details

**Step 1** The PC scans the MAC address of a DUT.

**Step 2** Connect the RF port of the fixture to the Wi-Fi communication tester.

**Step 3** Place the module in the fixture and power on the module.

**Step 4** The PC receives the DUT log and checks whether the DUT is powered on. If the device is powered on, the PC starts to connect to the Wi-Fi communication tester. Otherwise, wait for the device to power on.

**Step 5** Start the test:

1. Run the following commands to set the parameters such as the protocol, bandwidth, channel, and rate.
```
AT+STARTSTA
AT+IFCFG=wlan0,down
AT+ALTX=1, protocol mode, bandwidth, channel, and rate
AT+IFCFG=wlan0,up
```
2. Set the tester parameters, and read and record the information such as the power, EVM, and frequency offset.
3. Disable the always TX function by running **AT+ALTX=0**.
4. Check the frequency offset. If the frequency offset exceeds the range, compensate the frequency offset until it falls within the range of the indicator. If the value is too large, perform the test in **Step 5.1** again after compensation.
5. Sets the next group of parameters to perform the test in **Step 5**.

**Step 6** According to the test result in **Step 5**, ensure that all the power values are within the specified range. Otherwise, perform the following steps to calibrate the power:

1. If the power of only a few protocol rates exceeds the range, adjust the corresponding dBB scale.
2. (Optional) If the average power difference between different bands is large (for example, greater than 0.2 dB), adjust each band.
3. (Optional) If the total power difference is large, adjust all bands at the same time.

Note:

- Average power difference of a band = Average power of all channels in the band at each protocol rate – Typical power
- Total power difference = Average power of all channels at each protocol rate – Typical power
- If the indicator value still exceeds the threshold after compensation, the module is faulty.

**Step 7** After the compensation, perform **Step 5** again for verification. If the power error distribution has high requirements, multiple iterations may be performed.

**Step 8** After the verification is successful, write the calibration parameters and MAC address to the eFUSE by using the command. For each board, the calibration parameters and MAC address can be written for a maximum of three times. The last written valid value (not all 0s) takes effect.

For details about how to write data to the eFUSE, see **3.2.5 Test Commands**.

**Step 9** (Optional) Switch the UART port for AT commands. If you need to switch the UART port after the performance test is complete, run the **AT+SETUART** command. The setting takes effect after the system is restarted.

**Step 10** (Optional) Reset the module and verify it based on sampling.

**Step 11** After the test is complete, power off the device.

**----End**

## 3.2.3 RF Calibration

Adjust the DUT parameters by sending AT commands. Calculate whether each value of the DUT needs to be compensated by reading the data of the Wi-Fi communication tester. After the compensation is performed by running the AT command and the verification is successful again, calibration parameters are written to the corresponding eFuse areas of the DUT.

**NOTE**

For details about the test procedure, see **Step 5** to **Step 10**.

For details about AT commands, see **3.2.5 Test Commands**.

## 3.2.4 MAC Address Write

After the DUT calibration test is passed, the MAC address scanned and recorded by the PC is sent to the DUT through the AT command and written to the corresponding eFUSE area of the DUT.

The written MAC address is the MAC address of the STA. Except that the last field of the SoftAP MAC address increases by 1, other fields are the same as the written MAC address.

**NOTE**

For details about AT commands, see **3.2.5 Test Commands**.

## 3.2.5 Test Commands

**NOTE**

- The commands are for reference only. For details, see the *Hi3861 V100/Hi3861L V100 AT Commands User Guide*.
- The factory test result must be saved to the eFUSE. Otherwise, the test result will be overwritten after the software is reloaded.

| No. | Test Command | Description |
|---|---|---|
| 1 | Enables always TX. | **AT +ALTX=<control>,<protocol_mode>,<bw>,<chn>,<rate>**<br>Parameter description:<br>**<control>**:<br>**1**: The modulation signal is always transmitted.<br>**2**: The DC signal is always transmitted (used for frequency offset measurement during CE certification).<br>**<protocol_mode>**: protocol type<br>**0**: 802.11n<br>**1**: 802.11g<br>**2**: 802.11b<br>**<bw>**: bandwidth<br>**5**: 5 MHz bandwidth<br>**10**: 10 MHz bandwidth<br>**20**: 20 MHz bandwidth<br>**<chn>**: channel ID. Value range: 1–14<br>**<rate>**: transmission rate, in Mbit/s<br>802.11b supports 1, 2, 5.5, and 11.<br>802.11g supports 6, 9, 12, 18, 24, 36, 48, and 54.<br>802.11n supports 0, 1, 2, 3, 4, 5, 6, and 7, indicating MCS0–7.<br>Example:<br>AT+ALTX=1,0,20,1,7<br>OK |
| 2 | Disables always TX. | AT+ALTX=0 |
| 3 | Compensates the power offset. | AT+CALBPWR=<band num>,<offset><br>Parameter description:<br>**band num**: 0, 1, or 2<br>**offset**: –60 to +60 (unit: 0.1 dBm)<br>Example:<br>AT+CALBPWR=0,10<br>OK |
| 4 | Compensates the power for each rate at 802.11b. | AT+CALRPWR=<protocol>,<rate>,<val><br>Parameter description:<br>**protocol**: 0 (11n), 1 (11g), and 2 (11b)<br>**rate**: 0–7 (11n), 0–7 (11g), 0–3 (11b)<br>**val**: –8 to +7 (unit: 0.1 dBm) |

| No. | Test Command | Description |
| --- | --- | --- |
| | Compensates the power for each rate at 802.11g. | Example:<br>AT+CALRPWR=0,1,7<br>OK |
| | Compensates the power for each rate at 802.11n. | |
| 5 | Compensates the frequency offset at room temperature. | AT+CALFREQ=<offset><br>Parameter description:<br>**offset**: frequency offset compensation value. The recommended value range is –60 to +60 (the value range that can be delivered is –128 to +127. The recommended step is less than 10. If the value exceeds the recommended range, the tester may fail to demodulate signals, or commands fail to be delivered.) The value is not in 1:1 proportional to the frequency offset (but about 1:3 ppm).<br>Example:<br>AT+CALFREQ=10<br>OK |
| 6 | Writes the calibration value to the eFUSE. | AT+WCALDATA[=type]<br>Parameter description:<br>**type**: write type. This parameter is optional. The default value is **0**. The value **0** indicates that data is written to the eFUSE, and the value **1** indicates that data is written to the NVRAM<br>**Note**: When the write type is **0** (eFUSE), each board has three write opportunities. If all compensation parameters are set to **0**, the write operation does not take effect. The last written non-all-0 value is valid. The two write types cannot be used together. The **freq** and **band pwr** compensation values of the eFUSE overwrite the corresponding values in the NV. The **rate pwr** compensation value written to the NV is added to the compensation value corresponding to the eFUSE.<br>Example:<br>AT+WCALDATA=1<br>OK |

| No. | Test Command | Description |
| --- | --- | --- |
| 7 | Writes the MAC address to the eFUSE. | AT+EFUSEMAC=<mac>[,type]<br>**AT+EFUSEMAC?**<br>Parameter description:<br>MAC address: for example, **3A:13:24:33:25:c3**<br>**type**: write type. This parameter is optional. The default value is **0**. 0: Data is written to the eFUSE; 1: Data is written to the NVRAM.<br>The read command is described as follows:<br>The MAC address is preferentially read from the NVRAM instead of the eFUSE.<br>**Note**: One each board, data can be written to the eFUSE for only three times. It is recommended that the number of times that data is written to the NVRAM be less than or equal to 20. The MAC address of the NVRAM has a higher priority, and therefore may overwrite that of the eFUSE.<br>Example:<br>AT+EFUSEMAC? # Query<br>+EFUSEMAC:00:00:00:00:00:00 # No valid MAC address has been written to the eFUSE and NV.<br>+EFUSEMAC:Efuse mac chance(s) left:3 times. # Number of times that the MAC address can be written to the eFUSE. This field is displayed only when no valid MAC address is configured for the NV.<br>OK<br>AT+EFUSEMAC=50:21:00:33:02:49,1 # The MAC address is written to the NV.<br>OK<br>AT+EFUSEMAC? # Readback query<br>+EFUSEMAC:50:21:00:33:02:49 # The NV has a valid MAC address, which can be used preferentially.<br>OK |

| No. | Test Command | Description |
| --- | --- | --- |
| 8 | Switches the UART port for ATcommands. | **AT+SETUART=***<UART port for AT commands, debugging, or sigma tests>*<br>Parameter description:<br>UART port for AT commands in the SDK<br>UART port for SDK debugging, such as shell command or diag diagnosis<br>UART port for sigma tests in the SDK<br>For example, **AT+SETUART=1,0,2** indicates that the AT command uses UART1.<br>**Note:**<br>If the three UART ports need to be used at the same time, the UART port numbers must be unique.<br>After the UART port is switched, the board can be invalid only after being restarted.<br>During SDK compilation, the default configuration can be written by using the NV item (ID: 0x42).<br>UART0 is used by default during program burning and is not affected by this configuration. |

| No. | Test Command | Description |
| --- | --- | --- |
| 9 | Queries the factory test compensation data. | AT+RCALDATA<br>Example:<br>AT+RCALDATA<br>+RCALDATA:Efuse cali chance(s) left:1 times. #Number of remaining times that the calibration value can be written<br>+RCALDATA:freq_offset 5 #Compensation value of the current frequency offset<br>+RCALDATA:band_pwr_offset_0 0 #Power compensation value of the current bandwidth<br>+RCALDATA:band_pwr_offset_1 -1<br>+RCALDATA:band_pwr_offset_2 0<br>+RCALDATA:rate_pwr_offset_11n 0x0 #Power compensation value of each 802.11n rate. Each byte indicates a rate.<br>+RCALDATA:rate_pwr_offset_11g 0x0<br>+RCALDATA:rate_pwr_offset_11b 0x0<br>+RCALDATA:dbb_scale_0 0x65656565 #DBB scale value after bandwidth power compensation<br>+RCALDATA:dbb_scale_1 0x60606161<br>+RCALDATA:dbb_scale_2 0x494d5959<br>+RCALDATA:dbb_scale_3 0x51595959<br>+RCALDATA:dbb_scale_4 0x49495151<br>+RCALDATA:freq_and_band_pwr_hybrid_offset 0x0500ff00 #Combination value by byte<br>OK |
| 10 | Resets the board. | *AT+RST=<delay_us>*<br>**<delay_us>**: restart delay, in μs<br>Example: **AT+RST=1000000** |

# 3.3 Functional Test

## 3.3.1 Test Items

| Test Item | Description | Test Solution |
| --- | --- | --- |
| MAC address query | Reads the MAC address. | The PC issues a command to read the MAC address through the UART port. |

| Test Item | Description | Test Solution |
| --- | --- | --- |
| GPIO test | Tests the hardware interface connectivity. | The AT command loopback is used to read the GPIO status to obtain the connectivity of the I/O soldering. |
| Signal strength test | Tests the RF signal quality. | The **AT+ALTX** command is used to set the channel and rate, and a radio communication tester is used to obtain the corresponding RF signal parameters, which you can use to check whether the parameters meet the specifications. |
| RX performance test | Tests the RX sensitivity. | A tester is used to transmit packets at a specified rate, protocol, power, and quantity, query the number of received packets, and check whether the number of received packets meets the specifications. |

## 3.3.2 Process Details

**Step 1** Scan the MAC code on the module.

**Step 2** Place the module in the fixture and power on the module (connect the RF port first and then power on the module).

**Step 3** Use the test software to check the print information of the UART port to determine whether the module is powered on. After the module is powered on, connect the module to the Wi-Fi communication tester.

**Step 4** Read the MAC address in the eFUSE and verifies whether it matches the scanned MAC address.

**Step 5** Test the signal strength.

**Step 6** Perform a GPIO test.

**Step 7** Finish the test.

**----End**

## 3.3.3 Signal Strength Test

Switch the 802.11b, 802.11g, and 802.11n protocol modes, and test the signal quality of channels 1, 7, and 13 in each mode, bandwidth, and rate.

**Step 1** Run the following commands:

```
AT+STARTSTA (issued only when AT+ALTX is run for the first time)
AT+IFCFG=wlan0,down
AT+ALTX=1, protocol, bandwidth, channel, and rate
AT+IFCFG=wlan0,up
```

**Step 2** Set the parameters (such as channel and protocol mode) of the corresponding tester, read the information such as power, EVM, and frequency offset, and check whether the specifications are met (see the RF TRX-DR specifications).

**Step 3** Disable the always TX function by running **AT+ALTX=0**.

**Step 4** Repeat **Step 1** to **Step 3** to test the next group until all groups are tested.

**----End**

## 3.3.4 RX Performance Test

Switch the 802.11b, 802.11g, and 802.11n protocol modes, and test the receiver sensitivity of channels 1, 7, and 13 in each mode, bandwidth, and rate.

**Step 1** Deliver the always RX command:

```
AT+STARTSTA (delivered upon the first execution)
AT+IFCFG=wlan0,down
AT+ALRX=1, protocol, bandwidth, channel, 0
AT+IFCFG=wlan0,up
```

**Step 2** Set the related parameters (channel, protocol mode, transmit power, and number of transmitted packets) of the tester, wait until the packet transmission is complete, read the number of packets that are successfully received, and check whether the specifications are met (for details, see the RF TRX-DR specifications).

The command for querying the number of successfully received packets is as follows:

```
AT+RXINFO #Query the number of received packets.
```

**Step 3** Disable the always RX function by running **AT+ALTX=0**.

**Step 4** Repeat **Step 1** to **Step 3** to test the next group until all groups are tested.

**----End**

**NOTE**

It is recommended that you use unicast packets for the test.

## 3.3.5 GPIO Test

A GPIO test is performed to check the electrical conductivity of the module soldering.

Use cables to lead all I/Os to the test mother board through the tooling base and perform tests on the mother board. Generally, you should test I/Os in pairs. Set I/O_X to output state A and the connected I/O_Y to input state. The conductivity is determined by reading the status of I/O_Y. If I/O_X is in state A, the two I/Os are connected properly; otherwise, the two I/O pins are not properly soldered.

## 3.3.6 Test Commands

**NOTE**

The commands are for reference only. For details, see the *Hi3861 V100/Hi3861L V100 AT Commands User Guide*.

| No. | Test Command | Description |
| --- | --- | --- |
| 1 | MAC address query | **AT+EFUSEMAC?**<br>Parameter description:<br>MAC address: for example, **3A:13:24:33:25:c3** |
| 2 | Related to GPIO tests | **AT+SETIOMODE**: Sets the I/O working mode.<br>**AT+GETIOMODE**: Queries the I/O working mode.<br>**AT+GPIODIR**: Sets the GPIO to input or output. Or **AT +WTGPIO**: Sets the GPIO output level.<br>**AT+RDGPIO**: Reads the GPIO level status. |
| 3 | Related to signal strength tests | **AT+STARTSTA**<br>Enables always TX.<br>**AT+IFCFG=wlan0,down**<br>**AT+ALTX=1**, protocol, bandwidth, channel, and rate<br>**AT+IFCFG=wlan0,up**<br>Disables always TX.<br>**AT+ALTX=0** |
| 4 | RX performance test | Enables always RX.<br>**AT+STARTSTA** (delivered upon the first execution)<br>**AT+IFCFG=wlan0,down**<br>**AT+ALRX=1**, protocol, bandwidth, channel, 0<br>**AT+IFCFG=wlan0,up**<br>Queries the number of received packets.<br>**AT+RXINFO**<br>Disables always RX.<br>**AT+ALRX=0** |

# 3.4 Switch a Factory Test BIN to a Service BIN

After the factory test is complete, run the **AT+FTM=0** command in factory test mode. After the module is restarted, the module switches from factory test mode to service mode. (If you need to switch from service mode to factory test mode during debugging, run the **AT+FTM=1** command and restart the module.)

After the factory test is complete, switch to the service mode (the factory test binary file can be deleted only in service mode), and run the **AT+FTMERASE** command to delete the factory test binary file. This prevents the factory test calibration parameters from being modified after delivery. After the factory test binary file is deleted, the factory test function command cannot be used. If the factory test needs to be performed again, start from software burning. After switching to the service mode, run the **AT+RST** command to reset the module and check whether the module can start properly.

**NOTICE**

Before commercial use, the binary file for the factory test must be erased. Otherwise, the RF parameters may be modified.