Hi3861 V100 / Hi3861L V100 NV

User Guide

Issue	**01**
Date	**2020-04-30**

Trademarks and Permissions

Hi, HISILICON, and other HiSilicon icons are trademarks of HiSilicon Technologies Co., Ltd. All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

HiSilicon (Shanghai) Technologies Co., Ltd.

Address: New R&D Center, 49 Wuhe Road,
Bantian, Longgang District,
Shenzhen 518129 P. R. China

Website: https://www.hisilicon.com/en/

Email: support@hisilicon.com

About This Document

Purpose

This document describes the Hi3861 V100/Hi3861L V100 NV file directory structure, NV partitions, operations in the factory partition and non-factory partition, frequently asked questions (FAQs), and troubleshooting methods.

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
- 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.

Symbol	Description
⚠ 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. NOTICE is used to address practices not related to personal injury.
📖 NOTE	Supplements the important information in the main text. 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. • Updated **Figure 2-1** in **2 File Directories**. Added the description of **nv_builder**. Updated **Table 2-1**. • Updated the NV range reserved for the non-factory partition user in **3 NV Partitions**. Updated the division details of the factory partition and non-factory partition. • Updated the code sample for NV interface initialization in **4.2 API Usage**. • Updated the code sample for NV initialization in **4.3 Programming Sample**. • Updated the code sample for NV interface initialization in **5.2 API Usage**. • Updated the code sample for NV initialization in **5.3 Programming Sample**. • Updated the description of **menuconfig** in CE and FCC versions in **6.1 Overview**. • Updated **Figure 6-2** in **6.2 Configuring the Power**. Added **Figure 6-3**. Updated the sample description about the assumption that the power corresponding to 11g 24 Mbps needs to be increased to 18 dBm. • Added **Figure 6-4** in **6.3 Configuring the Frequency Offset and Band Power Offset**. Added the description that negative numbers are configured in two's complement mode. • Added **6.4 Configuring RF PLL Parameters**.

Issue	Date	Change Description
00B03	2020-04-21	Added **6 RF PLL Parameter Configuration**.
00B02	2020-03-06	Deleted the description of the ID range in **7.1 Precautions**.
00B01	2020-01-15	This issue is the first draft release.

Contents

1 NV Overview

The non-volatile memory (NV) module manages critical system configurations. The NV is stored in the flash memory and is divided into the following partitions:

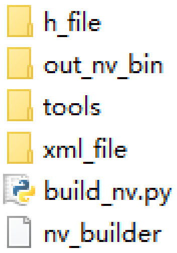
- Factory partition: used only at factory
- Non-factory partition:
  - Keep partition: NV items keep their original values after upgrade.
  - Modem partition: The values of the NV items are changed after upgrade.

The following describes how to use the factory partition and non-factory partition of NV.

2 File Directories

The NV items are configured in the **\tools\nvtool** directory. Figure 2-1 shows the file directories.

Figure 2-1 NV file directories



h_file
out_nv_bin
tools
xml_file
build_nv.py
nv_builder

The NV file directories are described as follows:

- **h_file**: NV structure storage files
- **out_nv_bin**: generated NV .bin files
- **tools**: NV tool files
- **xml_file**: configuration files for NV items of the non-factory partition and the factory partition
- **build_nv.py**: NV build script
- **nv_builder**: executable file generated by **build_nv.py**, which is used during **make build**

Table 2-1 describes the files to be configured.

Table 2-1 NV configuration description

List	File Path	Description
mss_nvi_db.xml	\tools\nvtool\xml_file	Defines the configuration file for NV items in use and NV items of the factory partition (CE certification).

List	File Path	Description
mss_nvi_db_fcc.xml	\tools\nvtool\xml_file	Defines the configuration file for NV items in use and NV items of the factory partition (FCC certification).
nv_factory_struct_def.txt	\tools\nvtool\h_file\nv	Defines the NV structure names in the factory partition.
nv_modem_struct_def.txt	\tools\nvtool\h_file\nv	Defines the common NV structure names.

When configuring **mss_nvi_db.xml**, you need to configure the NV group and NV items, as shown in **Table 2-2** and **Table 2-3**.

Table 2-2 Description of NV group configuration in mss_nvi_db.xml

Name	Meaning	Notes
NAME	NV group	Factory, Keep, and Modem can be configured.
ID	NV group ID	Each group ID must be unique.
FEATURE	NV group features	Not configured or used
USEDMODE	Usage mode of an NV group	Not configured or used
PARAM_DEF_FILE	NV group parameter file	Path of the structure for configuring NV parameters

Table 2-3 Description of the NV item configuration in mss_nvi_db.xml

Name	Meaning	Notes
ID	NV item ID	0x0–0xFF are valid. Each ID must be unique.
NAME	NV item name	-
PARAM_NAME	NV structure name	-
PARAM_VALUE	Initial value of the structure corresponding to the NV item	Set this parameter to the actual initial value.
DEV	Type of the network device on which the NV takes effect	The value can be **CCO**, **STA**, or **NDM**.

Name	Meaning	Notes
CATEGORY, **BROADCAST**, and **DESCRIPTION** are not configured or used.		

3 NV Partitions

With the range of [0x0, 0xFF], the NV is divided into the factory partition and non-factory partition:

- Factory partition: [0x0, 0x1F], where [0x16, 0x1F] is reserved for the user
- Non-factory partition: [0x20, 0xFF], where [0x98, 0xFF] is reserved for the user

Table 3-1 shows the division details.

Table 3-1 Division between the factory partition and non-factory partition

Category	Start Address	End Address	Description
Factory partition	HI_NV_FACTORY_ID_START (0x0)	HI_NV_FACTORY_ID_END (0x16)	Excluding 0x16
Reserved area for the factory partition user	HI_NV_FACTORY_USR_ID_START (0x16)	HI_NV_FACTORY_USR_ID_END (0x20)	Excluding 0x20
Non-factory partition	HI_NV_NORMAL_ID_START (0x20)	HI_NV_NORMAL_ID_END (0x80)	Excluding 0x80
Non-factory partition with unchanged data after upgrade	HI_NV_STABLE_ID_START (0x80)	HI_NV_STABLE_ID_START (0x98)	Excluding 0x98

Category	Start Address	End Address	Description
Reserved area for the user of the non-factory partition with unchanged data after upgrade	HI_NV_STABLE_USR_ID_START (0x98)	HI_NV_STABLE_USR_ID_START (0xA0)	Excluding 0xA0
Reserved area for the non-factory partition user	HI_NV_NORMAL_USR_ID_START (0xA0)	HI_NV_NORMAL_USR_ID_END (0xFF)	Including 0xFF

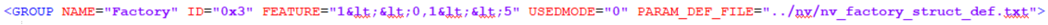
4 Operation Methods of NV Factory Partition

4.1 Procedure

Step 1 Set an NV group.

Open the **mss_nvi_db.xml** file in **\tools\nvtool\xml_file** of the SDK, set **GROUP NAME** to **Factory**, and set **PARAM_DEF_FILE** to **../nv/nv_factory_struct_def.txt**, as shown in **Figure 4-1**.

Figure 4-1 NV group configuration for the factory partition

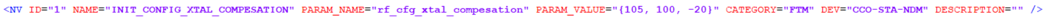


```
<GROUP NAME="Factory" ID="0x3" FEATURE="1&lt;&lt;0,1&lt;&lt;5" USEDMODE="0" PARAM_DEF_FILE="../nv/nv_factory_struct_def.txt">
```

Step 2 Set an NV item.

Add an NV item of the factory partition under **GROUP NAME="Factory"**, as shown in **Figure 4-2**.

Figure 4-2 NV item configuration for the factory partition



```
<NV ID="1" NAME="INIT_CONFIG_XTAL_COMPESATION" PARAM_NAME="rf_cfg_xtal_compesation" PARAM_VALUE="{105, 100, -20}" CATEGORY="FTM" DEV="CCO-STA-NDM" DESCRIPTION="" />
```

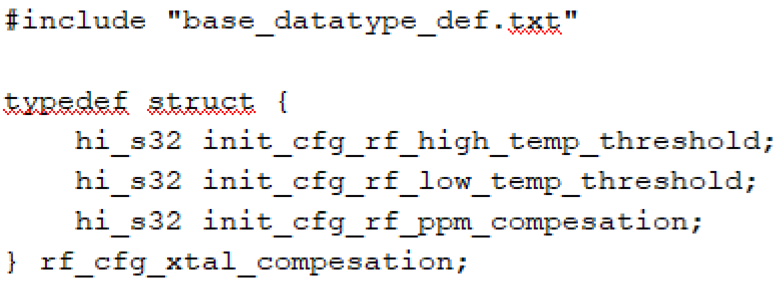
- **ID**: NV ID
- **NAME**: NV name
- **PARAM_NAME**: NV structure name
- **PARAM_VALUE**: initial value of the NV structure The initialization format requirements are as follows:
  - Members are separated by commas (,) despite their data types.
  - A basic data type array is managed by **[ ]**. If multiple members are assigned the same value, abbreviation is supported, for example, **[0,]**.

  - A structure array is enclosed by **{}**.
  - A structure is enclosed by **{}**.
  - A bit is enclosed by **{}**.
- **DEV**: type of the network device on which the NV takes effect. The options are **CCO**, **STA**, and **NDM**. For example, **CCO-STA** indicates that this NV item is generated by both the CCO and STA.
- **CATEGORY** and **DESCRIPTION**: reserved

Step 3 Add an NV structure.

After setting the NV value, open the **nv_factory_struct_def.txt** file in the **\tools\nvtool\h_file\nv** directory and add the **rf_cfg_xtal_compesation** structure corresponding to the NV value, as shown in **Figure 4-3**.

Figure 4-3 Example of adding an NV structure



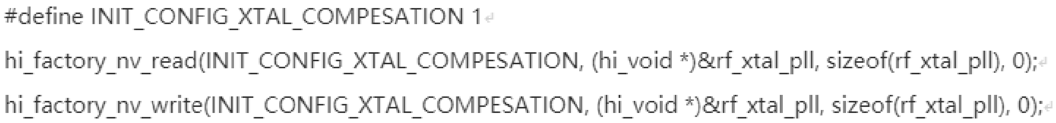
```
#include "base_datatype_def.txt"

typedef struct {
    hi_s32 init_cfg_rf_high_temp_threshold;
    hi_s32 init_cfg_rf_low_temp_threshold;
    hi_s32 init_cfg_rf_ppm_compesation;
} rf_cfg_xtal_compesation;
```

Step 4 Read or write the NV.

After the NV structure is added, call **hi_factory_nv_read** or **hi_factory_nv_write** to read or write the NV value for the factory partition, as shown in **Figure 4-4**.

Figure 4-4 Read and write in the NV factory partition



```
#define INIT_CONFIG_XTAL_COMPESATION 1
hi_factory_nv_read(INIT_CONFIG_XTAL_COMPESATION, (hi_void *)&rf_xtal_pll, sizeof(rf_xtal_pll), 0);
hi_factory_nv_write(INIT_CONFIG_XTAL_COMPESATION, (hi_void *)&rf_xtal_pll, sizeof(rf_xtal_pll), 0);
```

----End

4.2 API Usage

- Initialize the NV APIs before reading or writing data of the NV factory partition. Example:

```
/* Call before use, hi_nv_init(0x8000, 0x2000, 0x1000); */
hi_u32 hi_factory_nv_init(hi_u32 addr, hi_u32 total_size, hi_u32 block_size);
```

- Read data of the NV factory partition. Example:

```
/* The ID corresponds to the definition in the XML file. pdata indicates the data buffer, and len indicates the data length (in bytes). Generally, the value is sizeof (corresponding structure in the XML file). The flag is to be determined and reserved for future performance optimization. */
hi_u32 hi_factory_nv_read(hi_u8 id, hi_pvoid pdata, hi_u8 len, hi_u32 flag);
```

- Write data of the NV factory partition. Example:

```
/* The ID corresponds to the definition in the XML file. pdata indicates the data buffer, and
len indicates the data length (in bytes). Generally, the value is sizeof (corresponding
structure in the XML file). The flag is to be determined and reserved for future
performance optimization. */
hi_u32 hi_factory_nv_write(hi_u8 id, hi_pvoid pdata, hi_u8 len, hi_u32 flag);
```

4.3 Programming Sample

Programming sample for the NV factory partition:

```
/* Add the following information to GROUP NAME="Factory" in nv mss_nvi_db.xml: */
<NV ID="0x02" NAME="NV_ID" PARAM_NAME="test_nv" PARAM_VALUE="{0}"
CATEGORY="TEST" DEV="CCO-STA-NDM" BROADCAST="1" DESCRIPTION="" />

/* NV structure */
typedef struct {
    hi_u32 param;
} test_nv;

#define NV_ID 0x02

hi_void nv_test_main(void)
{
    hi_u32 ret;
    test_nv nv;
    hi_u32 err_info = 0;

    hi_flash_deinit();
    ret = hi_flash_init();
    if (ret != HI_ERR_SUCCESS) {
        err_info |= 1 << 0x6;
    }

/* Initialize the NV. */
    ret = hi_factory_nv_init(0x8000, 0x2000, 0x1000);
    if (ret != HI_SUCCESS) {
        printf("nv init fail\r\n");
    }
/* Read the NV value. */
    ret = hi_factory_nv_read(NV_ID, &nv, sizeof(test_nv), 0);
    if (ret != HI_ERR_SUCCESS) {
        printf("%x\n", ret);
    }
    printf("%d, %d\n", ret, nv.param);

/* Assign a value to the NV structure parameter. */
    nv.param = 2;
/* Write the NV value. */
    ret = hi_factory_nv_write(NV_ID, &nv, sizeof(test_nv), 0);
    if (ret != HI_ERR_SUCCESS) {
        printf("%x\n", ret);
    }
/* Read the written NV value again. */
    ret = hi_factory_nv_read(NV_ID, &nv, sizeof(test_nv), 0);
    if (ret != HI_ERR_SUCCESS) {
        printf("%x\n", ret);
    }
    printf("%d, %d\n",ret, nv.param);
}
```

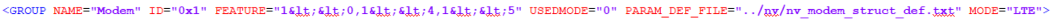
5 Operation Methods of NV Non-Factory Partition

5.1 Procedure

Step 1 Set an NV group.

Open the **mss_nvi_db.xml** file in the **\tools\nvtool\xml_file** directory of the SDK, set **GROUP NAME** to **Modem** or **Keep** for the non-factory partition group, and set **PARAM_DEF_FILE** to **../nv/nv_modem_struct_def.txt**, as shown in **Figure 5-1**.

Figure 5-1 NV group configuration for the non-factory partition

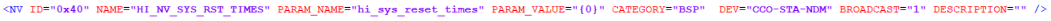


```
<GROUP NAME="Modem" ID="0x1" FEATURE="1&lt;&lt;0,1&lt;&lt;4,1&lt;&lt;5" USEDMODE="0" PARAM_DEF_FILE="../nv/nv_modem_struct_def.txt" MODE="LTE">
```

Step 2 Set an NV item.

Add an NV item to the corresponding NV group, as shown in **Figure 5-2**.

Figure 5-2 NV item configuration for the non-factory partition



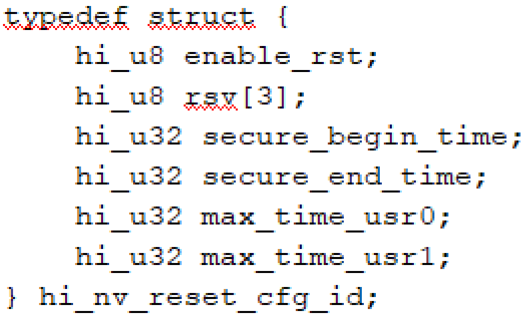
```
<NV ID="0x40" NAME="HI_NV_SYS_RST_TIMES" PARAM_NAME="hi_sys_reset_times" PARAM_VALUE="{0}" CATEGORY="BSP"  DEV="CCO-STA-NDM" BROADCAST="1" DESCRIPTION="" />
```

- **ID**: NV ID
- **NAME**: NV name
- **PARAM_NAME**: NV structure name
- **PARAM_VALUE**: initial value of the NV structure The initialization format requirements are as follows:
  - Members are separated by commas (,) despite their data types.
  - A basic data type array is managed by **[ ]**. If multiple members are assigned the same value, abbreviation is supported, for example, **[0,]**.

  - A structure array is enclosed by **{}**.
  - A structure is enclosed by **{}**.
  - A bit is enclosed by **{}**.
- **DEV**: type of the network device on which the NV takes effect. The options are **CCO**, **STA**, and **NDM**. For example, **CCO-STA** indicates that this NV item is generated by both the CCO and STA.
- **BROADCAST**, **CATEGORY** and **DESCRIPTION**: reserved

Step 3 After setting the NV value, open the **nv_modem_struct_def.txt** file in the **\tools\nvtool\h_file\nv** directory and add the **hi_sys_reset_times** structure corresponding to **Step 2**, as shown in **Figure 5-3**.

Figure 5-3 Example of adding an NV structure

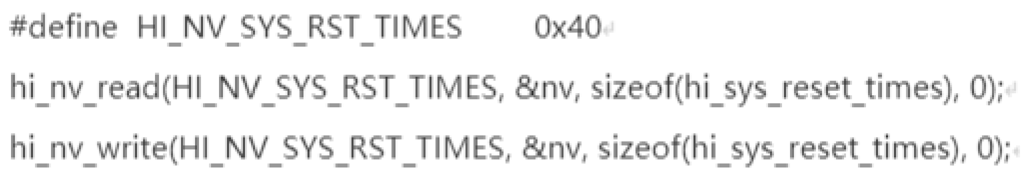


```
typedef struct {
    hi_u8 enable_rst;
    hi_u8 rsv[3];
    hi_u32 secure_begin_time;
    hi_u32 secure_end_time;
    hi_u32 max_time_usr0;
    hi_u32 max_time_usr1;
} hi_nv_reset_cfg_id;
```

Step 4 Read or write the NV.

After the NV structure is added, call **hi_nv_read** or **hi_nv_write** to read or write the NV value for the non-factory partition, as shown in **Figure 5-4**.

Figure 5-4 Read and write in the NV non-factory partition



```
#define  HI_NV_SYS_RST_TIMES        0x40
hi_nv_read(HI_NV_SYS_RST_TIMES, &nv, sizeof(hi_sys_reset_times), 0);
hi_nv_write(HI_NV_SYS_RST_TIMES, &nv, sizeof(hi_sys_reset_times), 0);
```

----End

5.2 API Usage

- Initialize the NV APIs before reading or writing data of the NV non-factory partition. Example:

```
/* Call before use, hi_nv_init(0xA000, 0x2000, 0x1000); */
hi_u32 hi_nv_init(hi_u32 addr, hi_u32 total_size, hi_u32 block_size);
```

- Read data of the NV non-factory partition. Example:

```
/* The ID corresponds to the definition in the XML file. pdata indicates the data buffer, and
len indicates the data length (in bytes). Generally, the value is sizeof (corresponding
structure in the XML file). The flag is to be determined and reserved for future
performance optimization. */
hi_u32 hi_nv_read(hi_u8 id, HI_CONST hi_pvoid pdata, hi_u8 len, hi_u32 flag);
```

- Write data of the NV non-factory partition. Example:

  ```
  /* The ID corresponds to the definition in the XML file. pdata indicates the data buffer, and len indicates the data length (in bytes). Generally, the value is sizeof (corresponding structure in the XML file). The flag is to be determined and reserved for future performance optimization. */
  hi_u32 hi_nv_write(hi_u8 id, HI_CONST hi_pvoid pdata, hi_u8 len, hi_u32 flag);
  ```

5.3 Programming Sample

Programming sample for the NV non-factory partition:

```
/* Add the following information to the nv mss_nvi_db.xml file: */
<NV ID="0x61" NAME="NV_ID" PARAM_NAME="test_nv" PARAM_VALUE="{0}"
CATEGORY="TEST" DEV="CCO-STA-NDM" BROADCAST="1" DESCRIPTION="" />

/* NV structure */
typedef struct {
    hi_u32 param;
} test_nv;

#define NV_ID 0x61

hi_void nv_test_main(void)
{
    hi_u32 ret;
    test_nv nv;
    hi_u32 err_info = 0;

    hi_flash_deinit();
    ret = hi_flash_init();
    if (ret != HI_ERR_SUCCESS) {
        err_info |= 1 << 0x6;
    }

/* Initialize the NV. */
    ret = hi_nv_init(0xA000, 0x2000, 0x1000);
    if (ret != HI_SUCCESS) {
        printf("nv init fail\r\n");
    }
/* Read the NV value. */
    ret = hi_nv_read(NV_ID, &nv, sizeof(test_nv), 0);
    if (ret != HI_ERR_SUCCESS) {
        printf("%x\n", ret);
    }
    printf("%d, %d\n", ret, nv.param);

/* Assign a value to the NV structure parameter. */
    nv.param = 3;
/* Write the NV value. */
    ret = hi_nv_write(NV_ID, &nv, sizeof(test_nv), 0);
    if (ret != HI_ERR_SUCCESS) {
        printf("%x\n", ret);
    }
/* Read the written NV value again. */
    ret = hi_nv_read(NV_ID, &nv, sizeof(test_nv), 0);
    if (ret != HI_ERR_SUCCESS) {
        printf("%x\n", ret);
    }
    printf("%d, %d\n",ret, nv.param);
}
```

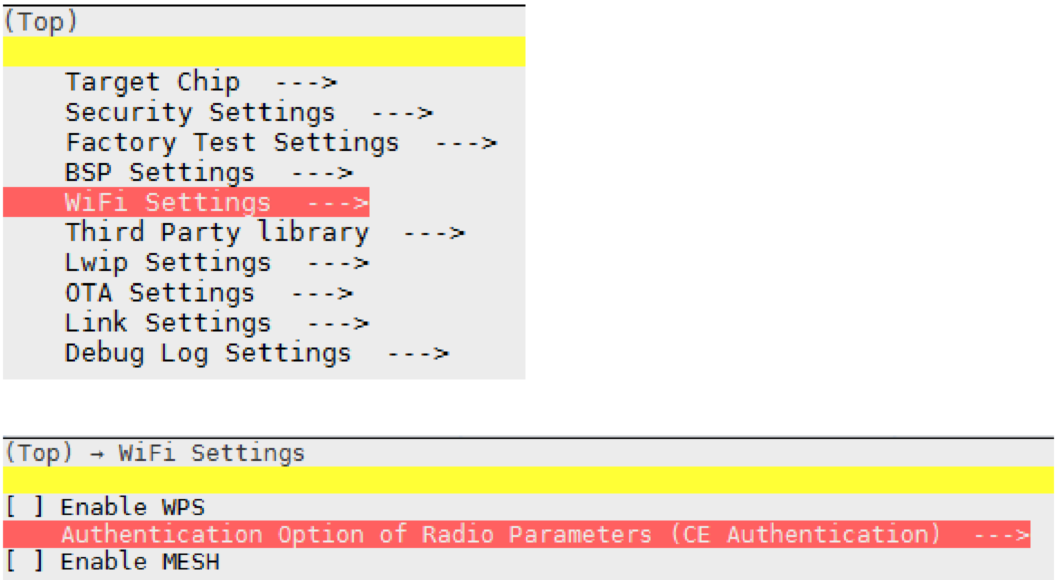
6 RF PLL Parameter Configuration

6.1 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 6-1 Configuration options for compilation

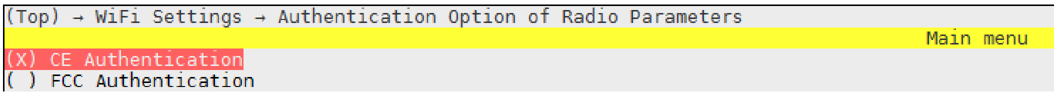


```
(Top)

    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
```

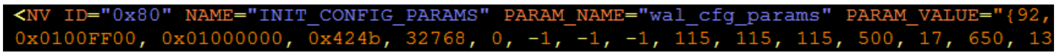
NOTE

This chapter uses the CE version as an example.

6.2 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 6-2**.

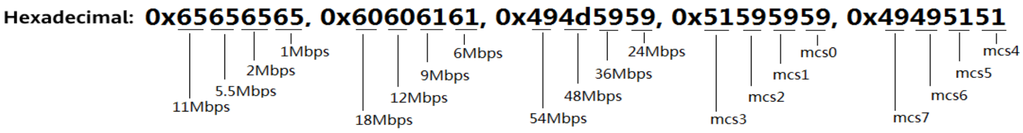
Figure 6-2 Example of the configuration item whose NV ID is 0x80 in the mss_nvi_db.xml file



```
<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 6-3**.

Figure 6-3 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 6-1**.

Table 6-1 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

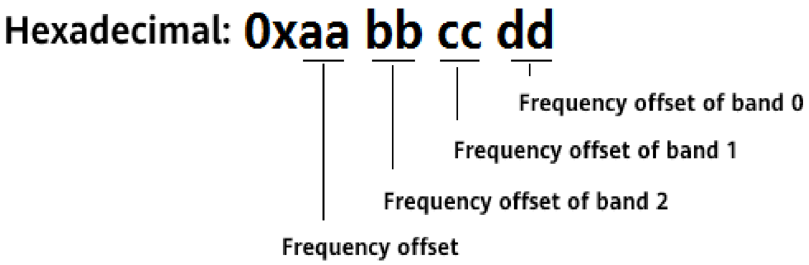
New dbb_scale = (10^((New target power – Old target power)/20)) x Old dbb_scale. **^** indicates the power operation, * indicates the multiplication operation, and **/** indicates the division operation.

Assume that you need to increase the power corresponding to 11g 24 Mbps to 18 dBm. The target power queried by **Table 6-1** 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.

6.3 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 6-4 Example of the 13th element in the NV ID="0x80" configuration item in the mss_nvi_db.xml file


Hexadecimal: 0xaa bb cc dd
Frequency offset of band 0
Frequency offset of band 1
Frequency offset of band 2
Frequency offset

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.

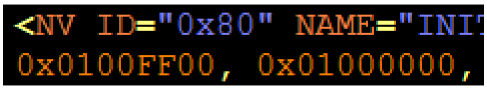
The negative number is configured in the form of two's complement (Two's complement of the configured value = 0x100 – Absolute value of the negative offset). For example, **0xf6000000** indicates that the frequency offset is –10.

6.4 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.

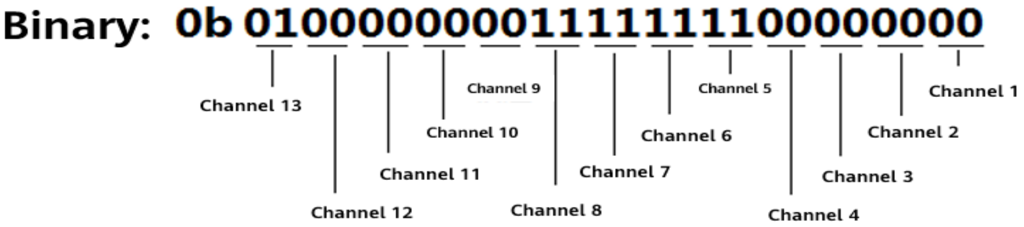
In the **tools\nvtool\xml_file\mss_nvi_db.xml** file of the SDK, the 14th and 15th elements of the PARAM_VALUE configuration item (**NV ID = "0x80"**) are used to adjust the RF PLL parameters of Hi3861 and Hi3861L, respectively. The value of each element starts from bit[0]. See **Figure 6-5**. Every two bits correspond to a channel (configurable range: 0x0–0x3).

Figure 6-5 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 6-6 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 6-5** 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**.

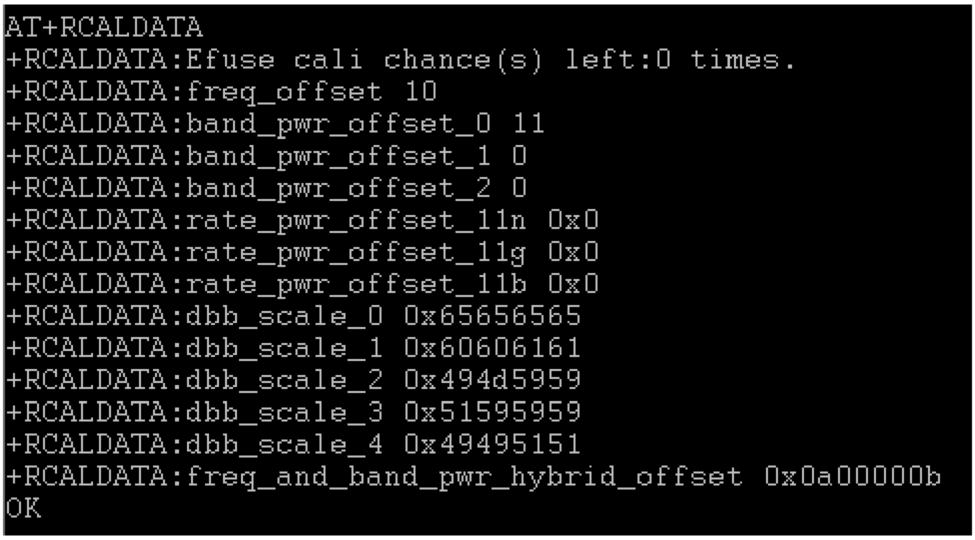
6.5 Obtaining the Factory Compensation Value

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 6-7**.

Figure 6-7 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

7 Precautions and Common Errors

7.1 Precautions

- The ID must be unique. Otherwise, an error is reported.
- The space of the current NV area is limited to 4 KB.
- The maximum size of each NV member is 252 bytes (256 – 4 bytes), where 4 bytes are used for cyclic redundancy check (CRC).

7.2 Common Errors

Mismatch Between the NV Structure and the Initial Value

An error would result if the **PARAM_VALUE** structure does not match the initial value structure or the format symbols are incorrectly used. Check whether a compilation alarm is reported during version compilation.

Too Frequent NV Write

The NV items are saved in the flash memory. Minimize the NV write times to prolong the lifetime of the flash memory.

The module APIs apply only to small data storage with limited write times.