



Hi3861V100 / Hi3861LV100 NV

使用指南

文档版本 01

发布日期 2020-04-30

版权所有 © 上海海思技术有限公司2020。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HISILICON、海思和其他海思商标均为海思技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

上海海思技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： <https://www.hisilicon.com/cn/>

客户服务邮箱：support@hisilicon.com



前言

概述

本文档详细的介绍了Hi3861V100/Hi3861LV100 NV文件的目录结构、NV分区、工厂区与非工厂区操作方法与常见的问题解答及故障处理方法。

产品版本

与本文档相对应的产品版本如下。

| 产品名称 | 产品版本 |
|---------|------|
| Hi3861 | V100 |
| Hi3861L | V100 |



读者对象

本文档主要适用于以下工程师：




- 技术支持工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

| 符号 | 说明 |
|-----------------------------------------------------------------------------------------------|---------------------------------|
|  危险 | 表示如不可避免则将会导致死亡或严重伤害的具有高等级风险的危害。 |
|  警告 | 表示如不可避免则可能导致死亡或严重伤害的具有中等级风险的危害。 |



| 符号 | 说明 |
|---------------------------------------------------------------------------------------------|------------------------------------------------------------------------|
|  注意 | 表示如不可避免则可能导致轻微或中度伤害的具有低等级风险的危害。 |
|  须知 | 用于传递设备或环境安全警示信息。如不可避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。 |
|  说明 | 对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。 |

修改记录

| 文档版本 | 发布日期 | 修改说明 |
|-------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 01 | 2020-04-30 | 第一次正式版本发布。 <ul style="list-style-type: none">在“2 文件目录”中更新图2-1；新增关于nv_builder的说明；更新表2-1。在“3 NV分区”中更新非工厂区中预留给用户使用的NV范围；更新工厂区与非工厂区具体划分的说明。更新“4.2 工厂区接口使用”中对NV接口初始化的代码示例。更新“4.3 编程示例”中NV初始化的代码示例。更新“5.2 非工厂区接口使用”中对NV接口初始化的代码示例。更新“5.3 编程示例”中NV初始化的代码示例。在“6.1 概述”中更新关于CE和FCC版本的menuconfig配置的描述。在“6.2 功率配置”中更新图6-2；新增图6-3；更新关于假设需要将11g 24Mbps对应的功率提高到18dBm的示例说明。在“6.3 频率偏移和band功率偏移配置”中新增图6-4；新增关于负数按补码形式配置的说明。新增“6.4 RF PLL参数配置”小节。 |
| 00B03 | 2020-04-21 | 新增“ 6 RF PLL参数配置 ”章节。 |
| 00B02 | 2020-03-06 | 删除“ 7.1 注意事项说明 ”关于ID范围的说明。 |
| 00B01 | 2020-01-15 | 第一次临时版本发布。 |



目录

| | |
|----------------------------|----|
| 前言..... | i |
| 1 NV 概述..... | 1 |
| 2 文件目录..... | 2 |
| 3 NV 分区..... | 4 |
| 4 工厂区 NV 操作方法..... | 5 |
| 4.1 工厂区操作步骤..... | 5 |
| 4.2 工厂区接口使用..... | 6 |
| 4.3 编程示例..... | 7 |
| 5 非工厂区 NV 操作方法..... | 8 |
| 5.1 非工厂区操作步骤..... | 8 |
| 5.2 非工厂区接口使用..... | 9 |
| 5.3 编程示例..... | 10 |
| 6 RF PLL 参数配置..... | 11 |
| 6.1 概述..... | 11 |
| 6.2 功率配置..... | 12 |
| 6.3 频率偏移和 band 功率偏移配置..... | 13 |
| 6.4 RF PLL 参数配置..... | 14 |
| 6.5 获取产测出厂补偿值..... | 15 |
| 7 注意事项及常见错误说明..... | 16 |
| 7.1 注意事项说明..... | 16 |
| 7.2 常见错误说明..... | 16 |



1 NV 概述

NV模块用于管理系统关键配置信息。NV存储于Flash上，分为以下2个区：

- 工厂区：仅在工厂时使用。
- 非工厂区：分为以下2个区：
 - Keep区：NV项在升级后保留原值
 - Modem区：NV项在升级后被新版本值替换。

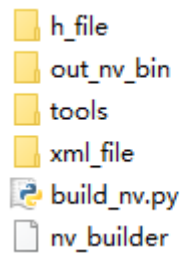
下面介绍NV工厂区与非工厂区的使用方式。



2 文件目录

NV项配置位于“\tools\nvtool”目录，具体文件目录说明如[图2-1](#)所示。

图 2-1 NV 文件目录



NV文件目录说明如下：

- h_file：NV结构体存放位置
- out_nv_bin：生成的NV bin文件
- tools：NV工具文件
- xml_file：NV项和工厂区NV项配置文件
- build_nv.py：NV的编译脚本
- nv_builder：build_nv.py生成的可执行文件，make编译时使用

在配置过程中，需要配置的具体文件说明如[表2-1](#)所示。

表 2-1 NV 配置说明

| 列表 | 文件路径 | 文件说明 |
|---------------------------|-----------------------------|-------------------------|
| mss_nvi_db.xml | \tools\nvtool \xml_file | 使用NV项和工厂区NV项配置文件（CE认证） |
| mss_nvi_db_fcc.xml | \tools\nvtool \xml_file | 使用NV项和工厂区NV项配置文件（FCC认证） |
| nv_factory_struct_def.txt | \tools\nvtool \h_file\nv | 定义工厂区NV结构体名称位置 |



| 列表 | 文件路径 | 文件说明 |
|-------------------------|-----------------------------|---------------|
| nv_modem_struct_def.txt | \tools\nvtool \h_file\nv | 定义普通NV结构体名称位置 |

在配置mss_nvi_db.xml过程中，需要对NV组以及具体NV项进行配置，NV组配置如表2-2所示，具体NV项配置如表2-3所示。

表 2-2 NV 组配置 mss_nvi_db.xml 说明

| 名称 | 含义 | 说明 |
|----------------|---------|-----------------------|
| NAME | NV组 | 可配置Factory、Keep、Modem |
| ID | NV组ID | 各组ID不能重复 |
| FEATURE | NV组特征 | 暂未配置使用 |
| USEDMODE | NV组使用模式 | 暂未配置使用 |
| PARAM_DEF_FILE | NV组参数文件 | 配置NV参数结构体路径 |

表 2-3 NV 项配置 mss_nvi_db.xml 说明

| 名称 | 含义 | 说明 |
|---------------------------------------|--------------|-----------------------|
| ID | NV项ID号 | 0x0 ~ 0xFF有效，各项ID不能重复 |
| NAME | NV项名字说明 | - |
| PARAM_NAME | NV结构体名称 | - |
| PARAM_VALUE | NV项对应结构体的初始值 | 按实际初始值配置 |
| DEV | NV生效的网络设备类型 | 选取值可为CCO、STA、NDM |
| CATEGORY、BROADCAST、DESCRIPTION暂未配置使用。 | | |



3 NV 分区

NV区域范围为[0x0, 0xFF]，工厂区与非工厂区区域划分如下：

- 工厂区：NV范围为[0x0, 0x1F]，其中[0x16, 0x1F]预留给用户使用。
- 非工厂区：NV范围为[0x20, 0xFF]，其中[0x98, 0xFF]预留给用户使用。

工厂区与非工厂区具体划分如表3-1所示。

表 3-1 工厂区与非工厂区具体划分说明

| 类别 | 起始地址 | 结束地址 | 备注 |
|--------------|-----------------------------------|----------------------------------|----------|
| 工厂区 | HI_NV_FACTORY_ID_START (0x0) | HI_NV_FACTORY_ID_END (0x16) | 不包括 0x16 |
| 工厂区用户预留区 | HI_NV_FACTORY_USR_ID_START (0x16) | HI_NV_FACTORY_USR_ID_END (0x20) | 不包括 0x20 |
| 非工厂区 | HI_NV_NORMAL_ID_START (0x20) | HI_NV_NORMAL_ID_END (0x80) | 不包括 0x80 |
| 非工厂升级保留区 | HI_NV_STABLE_ID_START (0x80) | HI_NV_STABLE_ID_START (0x98) | 不包括 0x98 |
| 非工厂升级保留用户预留区 | HI_NV_STABLE_USR_ID_START (0x98) | HI_NV_STABLE_USR_ID_START (0xA0) | 不包括 0xA0 |
| 非工厂用户预留区 | HI_NV_NORMAL_USR_ID_START (0xA0) | HI_NV_NORMAL_USR_ID_END (0xFF) | 包括 0xFF |



4 工厂区 NV 操作方法

4.1 工厂区操作步骤

4.2 工厂区接口使用

4.3 编程示例

4.1 工厂区操作步骤

步骤1 设置NV组。

打开SDK目录“\tools\nvtool\xml_file”中的“mss_nvi_db.xml”文件，选择 GROUP NAME="Factory" 组别，PARAM_DEF_FILE选择工厂区NV结构体 PARAM_DEF_FILE="..\nv\nv_factory_struct_def.txt"（如图4-1所示）。

图 4-1 工厂区 NV 组配置

```
<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">
```

步骤2 设置NV项。

在GROUP NAME="Factory"下增加工厂区NV配置项（如图4-2所示）。

图 4-2 工厂区 NV 项配置

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

- ID: NV项ID号。
- NAME: NV项名字说明。
- PARAM_NAME: NV结构体名称。
- PARAM_VALUE: NV项对应结构体的初始值。初始化格式要求如下:
 - 成员不区分数据类型，使用逗号进行分隔。
 - 基本数据类型数组使用 “[]” 管理，多个成员赋值为相同值可以缩写（例如 “[0,]”）。
 - 结构体数组使用 “{ }” 表示。
 - 结构体使用 “{ }” 表示。



- 比特位使用“{ }”表示。
- DEV: 本NV生效的网络设备类型, 选取值可为CCO、STA、NDM (例如: CCO-STA表示CCO和STA均生成该NV)。
- CATEGORY、DESCRIPTION: 暂未使用。

步骤3 添加NV结构体。

设置完NV值后, 打开目录“\tools\nvtool\h_file\nv”中的“nv_factory_struct_def.txt”文件, 添加设置NV值对应的结构体“rf_cfg_xtal_compesation”(如图4-3所示)。

图 4-3 NV 添加结构体示例

```
#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;
```

步骤4 NV读写操作。

添加NV结构体后, 调用hi_factory_nv_read接口读取设定初始NV值或调用hi_factory_nv_write接口写入工厂区NV值(如图4-4所示)。

图 4-4 工厂区 NV 读写操作

```
#define INIT_CONFIG_XTAL_COMPESTION 1

hi_factory_nv_read(INIT_CONFIG_XTAL_COMPESTION, (hi_void *)&rf_xtal_pll, sizeof(rf_xtal_pll), 0);
hi_factory_nv_write(INIT_CONFIG_XTAL_COMPESTION, (hi_void *)&rf_xtal_pll, sizeof(rf_xtal_pll), 0);
```

----结束

4.2 工厂区接口使用

- 在对工厂区NV数据读写前, 需要对NV的接口进行初始化。示例:
/* 使用前调用, hi_nv_init(0x8000, 0x2000, 0x1000); */
hi_u32 hi_factory_nv_init(hi_u32 addr, hi_u32 total_size, hi_u32 block_size);
- 读工厂区NV数据操作。示例:
/* id与xml中定义对应, pdata为数据buffer, len为数据长度(字节为单位), 一般为sizeof(xml中对应结构体), flag待定, 为后续性能优化预留 */
hi_u32 hi_factory_nv_read(hi_u8 id, hi_pvoid pdata, hi_u8 len, hi_u32 flag);
- 写工厂区NV数据操作。示例:
/* id与xml中定义对应, pdata为数据buffer, len为数据长度(字节为单位), 一般为sizeof(xml中对应结构体), flag待定, 为后续性能优化预留 */
hi_u32 hi_factory_nv_write(hi_u8 id, hi_pvoid pdata, hi_u8 len, hi_u32 flag);



4.3 编程示例

工厂区NV编程示例：

```
/* nv mss_nvi_db.xml 中选择 GROUP NAME="Factory" 组别下添加 */  
<NV ID="0x02" NAME="NV_ID" PARAM_NAME="test_nv" PARAM_VALUE="{0}"  
CATEGORY="TEST" DEV="CCO-STA-NDM" BROADCAST="1" DESCRIPTION="" />
```

```
/* NV结构体 */  
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;  
    }  
  
    /* NV初始化 */  
    ret = hi_factory_nv_init(0x8000, 0x2000, 0x1000);  
    if (ret != HI_SUCCESS) {  
        printf("nv init fail\r\n");  
    }  
    /* NV值读取 */  
    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);  
  
    /* 给NV结构体参数赋值 */  
    nv.param = 2;  
    /* NV值写入 */  
    ret = hi_factory_nv_write(NV_ID, &nv, sizeof(test_nv), 0);  
    if (ret != HI_ERR_SUCCESS) {  
        printf("%x\n", ret);  
    }  
    /* 再次读取写入的NV值 */  
    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 非工厂区 NV 操作方法

5.1 非工厂区操作步骤

5.2 非工厂区接口使用

5.3 编程示例

5.1 非工厂区操作步骤

步骤1 设置NV组。

打开SDK目录“\tools\nvtool\xml_file”中的“mss_nvi_db.xml”文件，选择非工厂区的GROUP NAME="Modem"或GROUP NAME="Keep" 组别，且PARAM_DEF_FILE选择非工厂区NV结构体PARAM_DEF_FILE="../nv/nv_modem_struct_def.txt"（如图5-1所示）。

图 5-1 非工厂区 NV 组配置

```
<GROUP NAME="Modem" ID="0x1" FEATURE="1&lt;t;6&lt;t;0,1&lt;t;6&lt;t;4,1&lt;t;6&lt;t;5" USEDMODE="0" PARAM_DEF_FILE="../nv/nv_modem_struct_def.txt" MODE="LTE">
```

步骤2 设置NV项。

在对应NV组中设置的组别增加NV配置项（如图5-2所示）。

图 5-2 非工厂区 NV 项配置

```
<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项ID号。
- NAME：NV项名字说明。
- PARAM_NAME：NV结构体名称。
- PARAM_VALUE：NV项对应结构体的初始值。初始化格式要求如下：
 - 成员不区分数据类型，使用逗号进行分隔。
 - 基本数据类型数组使用“[]”管理，多个成员赋值为相同值可以缩写（例如“[0,]”）。
 - 结构体数组使用“{ }”表示。



- 结构体使用 “{ }” 表示。
- 比特位使用 “{ }” 表示。
- DEV: 本NV生效的网络设备类型, 选取值可为CCO、STA、NDM (例如: CCO-STA表示CCO和STA均生成该NV)。
- BROADCAST、CATEGORY、DESCRIPTION: 暂未使用。

步骤3 设置完NV值后, 打开目录 “\tools\nvtool\h_file\nv” 中的 “nv_modem_struct_def.txt” 文件, 添加**步骤2**中对应的结构体 “hi_sys_reset_times” (如**图5-3**所示)。

图 5-3 NV 添加结构体示例

```
typedef struct {
    hi_u8 enable_rst;
    hi_u8 rstv[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;
```

步骤4 NV读写操作。

添加NV结构体后, 调用hi_nv_read接口读取设定初始NV值或调用hi_nv_write接口写入工厂区NV值 (如**图5-4**所示)。

图 5-4 非工厂区 NV 读写操作

```
#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);

----结束
```

5.2 非工厂区接口使用

- 在对非工厂区NV数据读写前, 需要对NV的接口进行初始化。示例:
/* 使用前调用, hi_nv_init(0xA000, 0x2000, 0x1000); */
hi_u32 hi_nv_init(hi_u32 addr, hi_u32 total_size, hi_u32 block_size);
- 读非工厂区NV数据操作。示例:
/* id与xml中定义对应, pdata为数据buffer, len为数据长度 (字节为单位), 一般为sizeof(xml中对应结构体), flag待定, 为后续性能优化预留 */
hi_u32 hi_nv_read(hi_u8 id, HI_CONST hi_pvoid pdata, hi_u8 len, hi_u32 flag);
- 写非工厂区NV数据操作。示例:
/* id与xml中定义对应, pdata为数据buffer, len为数据长度 (字节为单位), 一般为sizeof(xml中对应结构体), flag待定, 为后续性能优化预留 */
hi_u32 hi_nv_write(hi_u8 id, HI_CONST hi_pvoid pdata, hi_u8 len, hi_u32 flag);



5.3 编程示例

非工厂区NV编程示例:

```
/* nv mss_nvi_db.xml 中添加 */
<NV ID="0x61" NAME="NV_ID" PARAM_NAME="test_nv" PARAM_VALUE="{0}"
CATEGORY="TEST" DEV="CCO-STA-NDM" BROADCAST="1" DESCRIPTION="" />

/* NV结构体 */
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;
    }

    /* NV初始化 */
    ret = hi_nv_init(0xA000, 0x2000, 0x1000);
    if (ret != HI_SUCCESS) {
        printf("nv init fail\r\n");
    }
    /* NV值读取 */
    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);

    /* 给NV结构体参数赋值 */
    nv.param = 3;
    /* NV值写入 */
    ret = hi_nv_write(NV_ID, &nv, sizeof(test_nv), 0);
    if (ret != HI_ERR_SUCCESS) {
        printf("%x\n", ret);
    }
    /* 再次读取写入的NV值 */
    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 参数配置

- 6.1 概述
- 6.2 功率配置
- 6.3 频率偏移和band功率偏移配置
- 6.4 RF PLL参数配置
- 6.5 获取产测出厂补偿值

6.1 概述

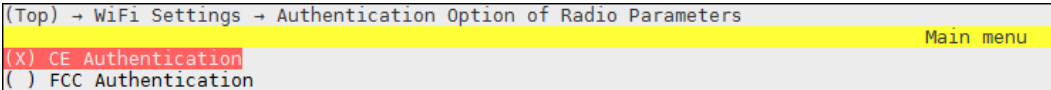
SDK支持CE和FCC版本的menuconfig配置，CE和FCC版本配置文件位于SDK版本目录的“tools\nvtool\xml_file”文件夹下，其中：

- mss_nvi_db.xml：对应CE版本（默认配置）
- mss_nvi_db_fcc.xml：对应FCC版本

图 6-1 编译配置选择示例

```
(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
```

说明

本章以CE版本为例进行介绍。

6.2 功率配置

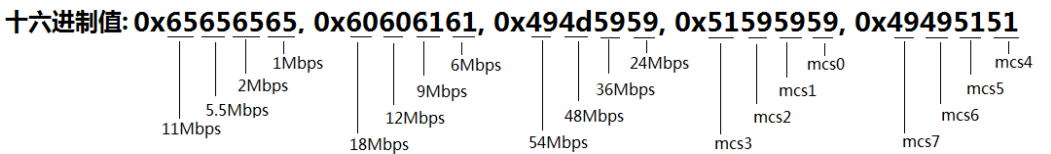
打开在SDK版本目录的“tools\nvtool\xml_file\mss_nvi_db.xml”，定位到NV ID="0x80"的配置项，如图6-2所示。

图 6-2 mss_nvi_db.xml 文件的 NV ID="0x80"的配置项示例

```
<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
```

PARAM_VALUE的第8~12元素值为各速率功率的配置值如图6-3所示。

图 6-3 PARAM_VALUE 的第 8~12 元素值含义示例



其含义分别为[11b 1~11Mbps]、[11g 6~18Mbps]、[11g 24~54Mbps]、[11n mcs0~3]、[11n mcs4~7]对应的dbb scale功率配置。每个元素值的每byte对应一种速率的功率配置，例如：PARAM_VALUE的第10个元素值0x494d5959的第一个字节0x59对应11g 24Mbps的功率配置，第4个字节0x49对应11g 54Mbps的功率配置。

默认值按出厂默认射频功率调节设置，例如：当前CE版本出厂默认射频功率如表6-1所示。

表 6-1 CE 版本出厂默认射频功率

| 协议 | 速率 | 20M带宽 |
|-----|---------|-------|
| 11b | 1Mbps | 16 |
| | 2Mbps | 16 |
| | 5.5Mbps | 16 |
| | 11Mbps | 16 |
| 11g | 6Mbps | 17 |
| | 9Mbps | 17 |
| | 12Mbps | 17 |
| | 18Mbps | 17 |



| 协议 | 速率 | 20M带宽 |
|-----|--------|-------|
| | 24Mbps | 17 |
| | 36Mbps | 17 |
| | 48Mbps | 16 |
| | 54Mbps | 16 |
| 11n | mcs0 | 16.5 |
| | mcs1 | 16.5 |
| | mcs2 | 16.5 |
| | mcs3 | 16.5 |
| | mcs4 | 16.5 |
| | mcs5 | 16.5 |
| | mcs6 | 16 |
| | mcs7 | 16 |

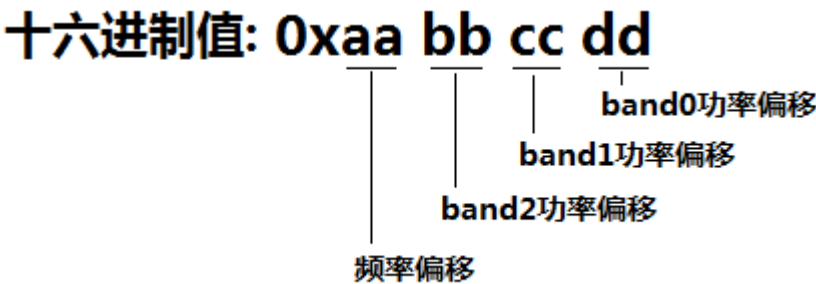
新dbb_scale=(10^((新目标功率 - 老目标功率)/20)) * 旧dbb_scale，其中“^”表示幂运算，“*”表示乘法运算，“/”表示除法运算。

假设需要将11g 24Mbps对应的功率提高到18dBm，通过表6-1查询到之前的目标功率为17dBm，从mss_nvi_db.xml中NV ID="0x80"的配置项的第10个元素值查询到之前dbb_scale为0x59，则新的dbb_scale=(10^((18 - 17)/20)) * 0x59=0x64，对应PARAM_VALUE的第10个元素值更新为0x494d5964（其他速率功率修改方法相类似）。修改保存后，重新编译SDK，重新加载bin到单板即可令配置生效。

6.3 频率偏移和 band 功率偏移配置

SDK版本“tools\nvtool\xml_file\mss_nvi_db.xml”文件的NV ID="0x80"配置项的第13个元素值对应频偏和band0~2的功率偏移（单位：0.1dB）。

图 6-4 mss_nvi_db.xml 文件的 NV ID="0x80"配置项的第 13 个元素值含义示例



其中：

- 第1~3byte：分别对应band0~2的功率偏移（band0对应信道1~4，band1对应信道5~9，band2对应信道10~13或14）。
- 第4byte：对应频率偏移。

例如：元素值0x0a00000b表示band0功率偏移为1.1dB，band1和2功率偏移为0dB，频率偏移值为10。

负数按补码形式配置（配置值补码 = 0x100 - 负偏移绝对值），例如：0xf6000000表示频率偏移值为-10。

6.4 RF PLL 参数配置

RF PLL在某些信道可能受晶体时钟影响，导致EVM劣化，此时可以通过调节寄存器值降低此影响。

SDK版本“tools\nvtool\xml_file\mss_nvi_db.xml”文件的NV ID="0x80"配置项PARAM_VALUE的第14和第15个元素值分别调节Hi3861和Hi3861L芯片的RF PLL参数（如图6-5所示），每个元素值从bit[0]开始，每2bit对应一个信道（可配置范围0x0~0x3）。

图 6-5 mss_nvi_db.xml 文件的 NV ID="0x80"配置项 PARAM_VALUE 的第 14 和第 15 个元素值示例

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

图 6-6 mss_nvi_db.xml 文件的 NV ID="0x80"配置项 PARAM_VALUE 的第 14 和第 15 个元素值含义示例

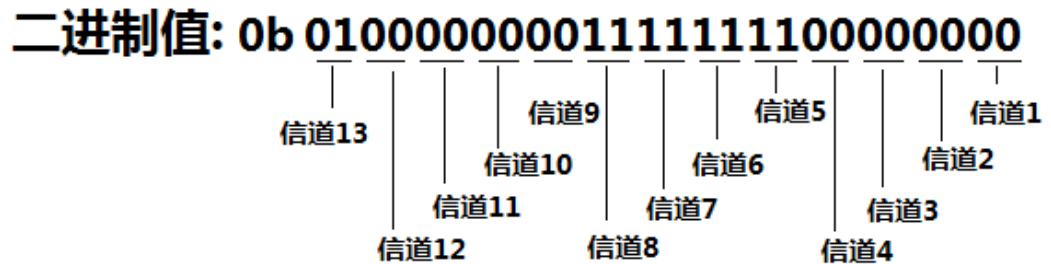


图6-5中PARAM_VALUE的第14和第15个元素值含义说明：

- 0x0100FF00：Hi3861芯片的RF PLL参数。信道5~8配置值为0x3，信道13配置值为0x1，其他信道配置值为0x0。
- 0x01000000：Hi3861L芯片的RF PLL参数。信道13配置值为0x1，其他信道配置值为0x0。

调节说明：该参数基于海思公版调试优化，用户单板的实际性能表现如果存在部分信道在同样目标功率下略差于其他信道（例如：ch5/6/7/8、ch13比ch1等信道EVM差），则可以微调该参数来优化EVM，调节方式为将配置值从0~3遍历，得到每个信道最佳的配置值，刷新至mss_nvi_db.xml中。



6.5 获取产测出厂补偿值

对于模组用户，如果需要基于模组出厂补偿值重新配置mss_nvi_db.xml，编译版本，则：

步骤1 从模组厂家获取模组产测信息，如果成功获取则直接执行**步骤3**。

步骤2 将模组上电，下发命令“AT+RCALDATA”获取出厂校准参数（如图6-7所示）。

图 6-7 下发命令“AT+RCALDATA”示例

```
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_1ln 0x0
+RCALDATA:rate_pwr_offset_1lg 0x0
+RCALDATA:rate_pwr_offset_1lb 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~dbb_scale_4：分别对应PARAM_VALUE的第8~12元素值。
- freq_and_band_pwr_hybrid_offset：对应PARAM_VALUE的第13个元素值。

步骤3 将获取的出厂校准参数配置到mss_nvi_db.xml中NV ID="0x80"的配置项，保存。

步骤4 重新编译SDK。

----结束



7 注意事项及常见错误说明

7.1 注意事项说明

7.2 常见错误说明

7.1 注意事项说明

- ID不可重复，否则将报错。
- 当前NV区域空间限制为4KB。
- NV每个成员限制为252byte（即（256-4）byte，其中4byte为CRC校验）。

7.2 常见错误说明

NV 结构体和初始值不匹配

PARAM_VALUE结构体与初值结构不匹配，或格式对应的符号不正确，都可能会导致问题，请编译版本时仔细核对是否存在编译告警。

写 NV 过于频繁

NV实际保存在Flash中，请尽量减少写NV的次数，确保不超过Flash寿命上限。

本模块接口只适用于写入次数有限的小数据存储。