



AIC8800D80X2系列射频测试说明

RF_TEST版本

版本号 v4.0

2025-7-28

爱科微半导体（上海）有限公司



公司	爱科微半导体（上海）有限公司 AIC Semiconductor (Shanghai) CO., Ltd.	
版本信息	日期	Release note
V1.0	2024 年 8 月 19 日	更新 pwrofst，添加 set_ant
V2.0	2024 年 12 月 23 日	更新 config 文件，tx 指令可设置功率
V3.0	2025 年 3 月 20 日	Flash 写保护
V4.0	2025 年 7 月 28 日	增加 Flash 写保护参数定义，更新 length 推荐值

AIC Semiconductor Confidential 20250728



目录

一. 工具介绍	3
二. WIFI_TEST 测试指令	4
2.1 WIFI 部分	4
2.1.1 WiFi 测试指令	4
2.1.2 单 TONE 测试指令	5
2.1.3 SRRC 指令	5
2.1.4 Flash 保护	6
2.1.5 晶体频偏校准指令	7
2.1.6 读写 mac 地址	8
2.1.7 TXpower 设置	9
2.1.8 信道功率补偿	10
2.1.9 config 文档使用	12
2.1.10 天线增益	17
2.1.11 物理层信息读取	18
2.1.12 产测底噪的读取	18
三. WIFI_TEST 编译说明	19
四. 测试示例:	21



一. 工具介绍

适用于 linux (ubuntu /android)

fmacfw.bin用于正常模式, fmacfw_rf.bin用于测试模式

以下以ubuntu为例, 用户界面输入测试命令: (以下命令均以 wlan0 为例, 实际以 ifconfig 显示为准) 格式
wifi_test if_name command parameters

COMMAND:

```
enum {  
    1. SET_TX,  
    2. SET_TXSTOP,  
    3. SET_RX,  
    4. SET_RXSTOP,  
    5. GET_RX_RESULT,  
    6. SET_XTAL_CAP,  
    7. SET_XTAL_CAP_FINE,  
    8. SET_FREQ_CAL,  
    9. SET_FREQ_CAL_FINE,  
    10. GET_FREQ_CAL,  
    11. SET_TXTONE  
    12. SET_SRRC  
    13. SET_MAC_ADDR,  
    14. GET_MAC_ADDR,  
    15. SET_BT_MAC_ADDR,  
    16. GET_BT_MAC_ADDR,  
    17. RDWR_PWRLVL,  
    18. RDWR_PWROFST,  
    19. RDWR_EFUSE_PWROFST,  
    20. AIC_USERCONFIG.TXT,  
};
```



二. WIFI_TEST测试指令

2.1 WIFI部分

2.1.1 WiFi测试指令

- wifi_test wlan0 set_tx chan bw mode rate length interval(可省掉) power(可省掉) \\ WiFi 发射测试开始

1-1-1: channel

	Chan num
2.4G	ch1-ch13
5G	Ch36-ch165

1-1-2: bandwidth

	bw
0	20M
1	40M
2	80M

1-1-3: mode 和 rate 对应关系

idx	mode		rate											
0	NON HT	idx	0	1	2	3	4	5	6	7	8	9	10	11
		rate	1M	2M	5.5M	11M	6M	9M	12M	18M	24M	36M	48M	54M
2	HT MF	idx	0-7											
		rate	MCS0-MCS7											
4	VHT	idx	0-9											
		rate	MCS0-MCS9											
5	HE SU	idx	0-11											
		rate	MCS0-MCS11											

idx	mode		rate											
2	HT MF	idx	8-15											
		rate	MIMO (MCS8-MCS15)											
4	VHT	idx	16-25											
		rate	MIMO (MCS0-MCS9)											
5	HE SU	idx	16-27											
		rate	MIMO (MCS0-MCS11)											

Length推荐值:

	20M	40M	80M
B/NON-HT	1000		
HT/VHT/HE	4000 (mcs7-11) 1000(mcs0)	8000 (mcs7-11) 1000(mcs0)	16000(mcs7-11) 1000(mcs0)

Interval: (Note: 此参数根据实际使用配置。对发包间隔无要求的此参数可不写, 使用默认值即可。)
发包间隔: 最小值50, 单位: μ s

Powr: 功率值, 此参数根据实际使用配置。对功率无要求的此参数可不写或者写255, 则使用默认值, (如果输入此参数前面的interval参数也需要输入)

SISO:

eg: wifi_test wlan0 set_tx 1 0 2 7 4000 \\ 2412MHz ,HT 20 MCS7 ,length4000 (发包间隔默认值)
eg: wifi_test wlan0 set_tx 1 0 2 7 4000 1000 \\ 2412MHz ,HT 20 MCS7 ,length4000 发包间1ms



eg: wifi_test wlan0 set_tx 1 0 2 7 4000 1000 15 \\ 2412MHz ,HT 20 MCS7 ,length4000 发包间1ms,功率15dBm

MIMO:

eg: wifi_test wlan0 set_tx 1 0 2 8 4000 \\ MIMO 2412MHz ,HT20 MCS8 ,length4000
eg: wifi_test wlan0 set_tx 1 0 4 16 4000 \\ MIMO 2412MHz ,VHT20 MCS0 ,length4000
eg: wifi_test wlan0 set_tx 1 0 5 27 4000 \\ MIMO 2412MHz ,HE20 MCS11 ,length4000

2. wifi_test wlan0 set_txstop \\ WiFi发射测试停止
no parameter
3. wifi_test wlan0 set_rx chan_num bw \\ WiFi接收测试开始
chan_num (见1-1-1 **channel**)
bw (见1-1-2 **bandwidth**)
eg: wifi_test wlan0 set_rx 1 0 \\ 2412MHz, bandwidth 20M
4. wifi_test wlan0 set_rxstop \\ WiFi接收测试停止
no parameter
5. wifi_test wlan0 get_rx_result \\ WiFi 接收测试收到的包的个数
no parameter
返回参数: 从 SET_RX 到 SET_RXSTOP 这段时间内接收到总的数据包个数
6. wifi_test wlan0 set_ant val \\ 发单流时天线口选择设置
val:0 ANT0打开, ANT1打开
val:1 ANT0打开, ANT1关闭
val:2 ANT0关闭, ANT1打开

2.1.2 单TONE测试指令

wifi_test wlan0 set_txtone val \\ tx单tone
val: 0 关闭
val: 1 val 打开 (1后面的参数范围-20-19)

0 关闭	no parameter		
1 打开	-20 -1	0	1-19
	负偏	中心偏点	正偏

eg: wifi_test wlan0 set_txtone 1 1 \\打开正向偏1M
eg: wifi_test wlan0 set_txtone 0 \\ 关闭

2.1.3 SRRC指令

过SRRC杂散测试时, 打开SRRC指令

wifi_test wlan0 set_src val
val:0 关闭
val:1 打开



2.1.4 Flash保护

wifi_test wlan0 exec_flash_oper val

- val: 0 \\ check 读取当前 Flash 状态, 返回值-1:SR 状态异常; 0:写保护关闭; 1:写保护打开
1 \\ recover恢复Flash SR状态, 关闭写保护
2 \\ protect打开Flash写保护
3 \\read_wcr0读取上电校准结果flag, 返回非0表示有校准结果, 负值表示没有校准结果
4 \\ erase_wcr0擦除上电校准结果

Note: 适用于 AIC8800M80X2P,完成 Flash 开关写保护, 读状态, 读取/擦除上电校准结果。

AIC Semiconductor Confidential 20250728



2.1.5 晶体频偏校准指令

AIC8800D80X2 XTAL 电路内部提供了可变负载电容，支持负载电容为 9-11pF 的 crystal unit。

1. `wifi_test wlan0 set_xtal_cap val` \\晶体频偏粗调，默认值16(0x10)，
范围0-31(0x00~0x1F)

val: 十进制有符号数

eg: `wifi_test wlan0 set_xtal_cap -2` \\ 负向频偏，降低内部负载电容
2. `wifi_test wlan0 set_xtal_cap_fine val` 晶体频偏细调，默认值31(0x1F)，
范围0-63 (0x00~0x3F)

val: 十进制有符号数

eg: `wifi_test wlan0 set_xtal_cap_fine 10` \\ 正向频偏，提高内部负载电容
3. `wifi_test wlan0 set_freq_cal val` \\ 写晶体频偏校准粗调值到efuse\flash
val 十六进制绝对值

eg: `wifi_test wlan0 set_freq_cal 1a` \\ 写晶体频偏校准粗调值 0x1A 到 efuse\flash
4. `wifi_test wlan0 set_freq_cal_fine val` \\写晶体频偏校准细调值到efuse\flash
val: 十六进制绝对值

eg: `wifi_test wlan0 set_freq_cal_fine 16` \\ 写晶体频偏校准细调值0x16到efuse\flash
5. `wifi_test wlan0 get_freq_cal` \\ 读频偏值
no parameter

粗调校准流程:

- ①判断 frequency offset (Δf) 极性, $\Delta f > 0$, setxtalcap 4, 反之, setxtalcap -4;
- ②判断 frequency offset (Δf) 极性, $\Delta f > 0$, setxtalcap 2, 反之, setxtalcap -2;
- ③判断 frequency offset (Δf) 极性, $\Delta f > 0$, setxtalcap 1, 反之, setxtalcap -1;

细调校准流程:

- ①判断 frequency offset (Δf) 极性, $\Delta f > 0$, setxtalcapfine 16, 反之, setxtalcapfine -16;
- ②判断 frequency offset (Δf) 极性, $\Delta f > 0$, setxtalcapfine 8, 反之, setxtalcapfine -8;
- ③判断 frequency offset (Δf) 极性, $\Delta f > 0$, setxtalcapfine 4, 反之, setxtalcapfine -4;
- ④判断 frequency offset (Δf) 极性, $\Delta f > 0$, setxtalcapfine 2, 反之, setxtalcapfine -2;
- ⑤判断 frequency offset (Δf) 极性, $\Delta f > 0$, setxtalcapfine 1, 反之, setxtalcapfine -1;

Note: 校准频偏指令对应参数均为十进制相对值，即相对默认值偏移值，输入指令后会返回配置后频偏实际参数，且以十六进制显示。写入efuse或flash的频偏校准值为十六进制绝对值

2.1.6 读写mac地址

- | | |
|---|---------------------------------|
| 1. wifi_test wlan0 set_mac_addr
(复) | \\写WiFi MAC地址到efuse(2次)或flash(重 |
|
eg: wifi_test wlan0 set_mac_addr 88 00 11 22 33 44 | \\写WiFi MAC地址 |
| 2. wifi_test wlan0 get_mac_addr
no parameter | \\ 读WiFi MAC地址 |
| 3. wifi_test wlan0 set_bt_mac_addr | \\写BT MAC地址到efuse(2次)或flash(重复) |
|
eg: wifi_test wlan0 set_bt_mac_addr 0A 1C 6B C6 96 7E | \\写BT MAC地址 |
| 4. wifi_test wlan0 get_bt_mac_addr
no parameter | \\ 读BT MAC地址 |

Note: 如果 wifi 还需要同时支持 p2p, softap, 两颗芯片的 mac 地址需要至少相差 4。



2.1.7 TXpower设置

1. `wifi_test wlan0 rdwr_pwrlvl band mod idx val` \\设置不同模式速率的功率
val: 十进制

4-1-1: band

	band		mod
2.4G	1	11b+11a/g	0
		11n/11ac	1
		11ax	2
5G	2	11a/g	0
		11n/11ac	1
		11ax	2

2.4G Rate Group

FmtIdx	0	1	2	3	4	5	6	7	8	9	10	11
11b+11a/g	1M	2M	5.5M	11M	6M	9M	12M	18M	24M	36M	48M	54M
11n/ac	MCS0	MCS1	MCS2	MCS3	MCS4	MCS5	MCS6	MCS7	MCS8	MCS9		
11ax	MCS0	MCS1	MCS2	MCS3	MCS4	MCS5	MCS6	MCS7	MCS8	MCS9	MCS10	MCS11

5G Rate Group

FmtIdx	0	1	2	3	4	5	6	7	8	9	10	11
11a/g	NA	NA	NA	NA	6M	9M	12M	18M	24M	36M	48M	54M
11n/ac	MCS0	MCS1	MCS2	MCS3	MCS4	MCS5	MCS6	MCS7	MCS8	MCS9		
11ax	MCS0	MCS1	MCS2	MCS3	MCS4	MCS5	MCS6	MCS7	MCS8	MCS9	MCS10	MCS11

Note: 5G 11a/g 比较特殊, 如果多个值同时写入, 前面4个写-128, 表示无效

`pwrlvl` 共有两种设置方法:

- 设置其中一个 Rate 的方法:

eg: `wifi_test wlan0 rdwr_pwrlvl 1 0 3 18` \\设置2.4G 11b+11a/g模式11M的TX power为18dBm

- 设置一组中多个 Rate 的方法:

eg. `wifi_test wlan0 rdwr_pwrlvl 1 1 15 15 15 15 15 14 14 14 13 13` \\设置2.4G 11n/ac模式下 MCS0-MCS9的发射功率分为15dBm 15 dBm 15 dBm 15 dBm 15 dBm 14 dBm 14 dBm 14 dBm 13 dBm 13 dBm

Note: 多个Rate的设置方法时需要将改模式下的所有速率都设置进去。

2. `wifi_test wlan0 rdwr_pwrlvl 0` \\读取功率增益档位, 写0或不写均实现读功能



2.1.8 信道功率补偿

1. awifi_test wlan0 rdwr_pwrofst band rate ch ofst

\\ 设置信道补偿

5-1-1: band\rate\ch\ofst 对应关系表

	band		Rate		ch	ofst
2.4G_ANT0	1	11b	0	CH1~CH4	0	-15~15
				CH5~CH9	1	-15~15
				CH10~CH13	2	-15~15
		ofdm_highrate	1	CH1~CH4	0	-15~15
				CH5~CH9	1	-15~15
				CH10~CH13	2	-15~15
2.4G_ANT1	2	11b	0	CH1~CH4	0	-15~15
				CH5~CH9	1	-15~15
				CH10~CH13	2	-15~15
		ofdm_highrate	1	CH1~CH4	0	-15~15
				CH5~CH9	1	-15~15
				CH10~CH13	2	-15~15
5.8G_ANT0	3	ofdm_highrate	0	CH36~CH50	0	-15~15
				CH51~CH64	1	-15~15
				CH98~CH114	2	-15~15
				CH115~CH130	3	-15~15
				CH131~CH146	4	-15~15
				CH147~CH166	5	-15~15
5.8G_ANT1	4	ofdm_highrate	0	CH36~CH50	0	-15~15
				CH51~CH64	1	-15~15
				CH98~CH114	2	-15~15
				CH115~CH130	3	-15~15
				CH131~CH146	4	-15~15
				CH147~CH166	5	-15~15

eg. wifi_test wlan0 rdwr_pwrofst 1 1 1 2 \\ 设置 2.4G ANT0, OFDM_highrate,CH5~CH9 信道补偿为 2

ofst 为带符号偏移值，步进为 1，对应功率变化 0.5dbm，最大 15，最小-15，可通过调整响应信道补偿值来优化信道功率差异。

Note: pwrofst 后面不带参数可直接显示当前发射功率增益档位配置信息。

Note:2.4G 分别在 11b_1M, 11g_54M 校准 11b, ofdm_highrate 速率。在 ch1, ch7, ch13 校准信道。

5G 分别在 11a_54M 校准 ofdm_highrate 速率。在 ch42, ch58, ch106, ch122, ch138, ch155 校准信道。

2. wifi_test wlan0 rdwr_efuse_pwrofst band rate ch ofst \\ 写信道补偿值到efuse（2次）或flash（重复）

eg. wifi_test wlan0 rdwr_efuse_pwrofst 1 1 1 2 \\写 2.4G, OFDM_highrate,CH5~CH9 校准值到 efuse

Note: efpwrofst 0 或者后不加参数能读取 efuse 中信道功率补偿值。

OFDM Rate 分类



2.4G

	OFDM												
	BPSK 1/2	BPSK 3/4	QPSK 1/2	QPSK 3/4	16QAM 1/2	16QAM 3/4	64QAM 2/3	64QAM 3/4	64QAM 5/6	256QAM 3/4	256QAM 5/6	1024QAM 3/4	1024QAM 5/6
NON-HT	6M	9M	12M	18M	24M	36M	48M	54M					
HT	MCS0		MCS1	MCS2	MCS3	MCS4	MCS5	MCS6	MCS7				
VHT	MCS0		MCS1	MCS2	MCS3	MCS4	MCS5	MCS6	MCS7	MCS8	MCS9		
HE	MCS0		MCS1	MCS2	MCS3	MCS4	MCS5	MCS6	MCS7	MCS8	MCS9	MCS10	MCS11

5G

	OFDM												
	BPSK 1/2	BPSK 3/4	QPSK 1/2	QPSK 3/4	16QAM 1/2	16QAM 3/4	64QAM 2/3	64QAM 3/4	64QAM 5/6	256QA M 3/4	256QA M 5/6	1024QA 3/4	1024QAM 5/6
NON-HT	6M	9M	12M	18M	24M	36M	48M	54M					
HT	MCS0		MCS1	MCS2	MCS3	MCS4	MCS5	MCS6	MCS7				
VHT	MCS0		MCS1	MCS2	MCS3	MCS4	MCS5	MCS6	MCS7	MCS8	MCS9		
HE	MCS0		MCS1	MCS2	MCS3	MCS4	MCS5	MCS6	MCS7	MCS8	MCS9	MCS10	MCS11



2.1.9 config文档使用

1.aic_userconfig.txt 文档使用:

随固件一起 cp 到 /lib/firmware/下, 更改文档内参数后掉电重新上电生效

enable = 0 文档不生效, enable = 1 文档生效, 默认为 1

(参数意义可以详见上述 2.1.4、2.1.5)

```
# txpwr_lvl
enable=1
lvl_11b_1lag_1m_2g4=18
lvl_11b_1lag_2m_2g4=18
lvl_11b_1lag_5m5_2g4=18
lvl_11b_1lag_11m_2g4=18
lvl_11b_1lag_6m_2g4=18
lvl_11b_1lag_9m_2g4=18
lvl_11b_1lag_12m_2g4=18
lvl_11b_1lag_18m_2g4=18
lvl_11b_1lag_24m_2g4=16
lvl_11b_1lag_36m_2g4=16
lvl_11b_1lag_48m_2g4=15
lvl_11b_1lag_54m_2g4=15
lvl_11n_1lac_mcs0_2g4=18
lvl_11n_1lac_mcs1_2g4=18
lvl_11n_1lac_mcs2_2g4=18
lvl_11n_1lac_mcs3_2g4=18
lvl_11n_1lac_mcs4_2g4=16
lvl_11n_1lac_mcs5_2g4=16
lvl_11n_1lac_mcs6_2g4=15
lvl_11n_1lac_mcs7_2g4=15
lvl_11n_1lac_mcs8_2g4=14
lvl_11n_1lac_mcs9_2g4=14
lvl_11ax_mcs0_2g4=18
lvl_11ax_mcs1_2g4=18
lvl_11ax_mcs2_2g4=18
lvl_11ax_mcs3_2g4=18
lvl_11ax_mcs4_2g4=16
lvl_11ax_mcs5_2g4=16
lvl_11ax_mcs6_2g4=15
lvl_11ax_mcs7_2g4=15
lvl_11ax_mcs8_2g4=14
lvl_11ax_mcs9_2g4=14
lvl_11ax_mcs10_2g4=13
lvl_11ax_mcs11_2g4=13
lvl_11a_6m_5g=18
lvl_11a_9m_5g=18
lvl_11a_12m_5g=18
lvl_11a_18m_5g=18
lvl_11a_24m_5g=16
lvl_11a_36m_5g=16
lvl_11a_48m_5g=15
lvl_11a_54m_5g=15
lvl_11n_1lac_mcs0_5g=18
lvl_11n_1lac_mcs1_5g=18
lvl_11n_1lac_mcs2_5g=18
lvl_11n_1lac_mcs3_5g=18
lvl_11n_1lac_mcs4_5g=16
```



```
lvl_11n_11ac_mcs5_5g=16
lvl_11n_11ac_mcs6_5g=15
lvl_11n_11ac_mcs7_5g=15
lvl_11n_11ac_mcs8_5g=14
lvl_11n_11ac_mcs9_5g=14
lvl_11ax_mcs0_5g=18
lvl_11ax_mcs1_5g=18
lvl_11ax_mcs2_5g=18
lvl_11ax_mcs3_5g=18
lvl_11ax_mcs4_5g=16
lvl_11ax_mcs5_5g=16
lvl_11ax_mcs6_5g=14
lvl_11ax_mcs7_5g=14
lvl_11ax_mcs8_5g=13
lvl_11ax_mcs9_5g=13
lvl_11ax_mcs10_5g=12
lvl_11ax_mcs11_5g=12
```

```
# txpwr_loss
```

```
loss_enable_2g4=0
```

```
loss_value_2g4=2 // 此值如果是需要配置天线增益请按正值配置， power loss 请按负值配置
```

```
loss_enable_5g=0
```

```
loss_value_5g=5 // 此值如果是需要配置天线增益请按正值配置， power loss 请按负值配置
```

```
# txpwr_ofst
```

```
ofst_enable=0
```

```
ofst_2g4_ant0_11b_chan_1_4=0
```

```
ofst_2g4_ant0_11b_chan_5_9=0
```

```
ofst_2g4_ant0_11b_chan_10_13=0
```

```
ofst_2g4_ant0_ofdm_highrate_chan_1_4=0
```

```
ofst_2g4_ant0_ofdm_highrate_chan_5_9=0
```

```
ofst_2g4_ant0_ofdm_highrate_chan_10_13=0
```

```
ofst_2g4_ant1_11b_chan_1_4=0
```

```
ofst_2g4_ant1_11b_chan_5_9=0
```

```
ofst_2g4_ant1_11b_chan_10_13=0
```

```
ofst_2g4_ant1_ofdm_highrate_chan_1_4=0
```

```
ofst_2g4_ant1_ofdm_highrate_chan_5_9=0
```

```
ofst_2g4_ant1_ofdm_highrate_chan_10_13=0
```

```
ofst_5g_ant0_ofdm_highrate_chan_42=0
```

```
ofst_5g_ant0_ofdm_highrate_chan_58=0
```

```
ofst_5g_ant0_ofdm_highrate_chan_106=0
```

```
ofst_5g_ant0_ofdm_highrate_chan_122=0
```

```
ofst_5g_ant0_ofdm_highrate_chan_138=0
```

```
ofst_5g_ant0_ofdm_highrate_chan_155=0
```

```
ofst_5g_ant1_ofdm_highrate_chan_42=0
```

```
ofst_5g_ant1_ofdm_highrate_chan_58=0
```

```
ofst_5g_ant1_ofdm_highrate_chan_106=0
```

```
ofst_5g_ant1_ofdm_highrate_chan_122=0
```

```
ofst_5g_ant1_ofdm_highrate_chan_138=0
```

```
ofst_5g_ant1_ofdm_highrate_chan_155=0
```

```
# xtal_cap
```

```
xtal_enable=0
```

```
xtal_cap=24
```

```
xtal_cap_fine=31
```



2.aic_powerlimit.txt 文档使用

用于单独限制信道功率，与上述 aic_userconfig.txt 放置路径相同，驱动打开 CONFIG_POWER_LIMIT 使能，默认地区有”SRRC FCC ETSI JP UNSET”，其中”UNSET”表示”未定义国家码所属地区”。

例如：Table 1 内的“CH01”后的 5 个”15”分别表示 5 个地区的信道”1”限制为“15dBm”。

执行“wifi_test wlan0 country_set ***”切换国家码指令后，根据驱动内的国家码地区映射表对应到”FCC”，然后与上述的 userconfig 文件内的功率值进行比较，取较小值作为发射功率值（txpwr_loss 使能后，userconfig.txt 的功率值和 power_limit.txt 内的功率值都会在设定值的基础上减取 loss_value 值）。

注：目前只有非信令模式支持不同带宽功率限制。

使用示例：

信令示例 1：

“wifi_test wlan0 country_set US”切换国家，假如 table 1 内的 FCC 的信道 1 值为 8，仪表端目标功率会是 8 左右。

非信令示例 1：

“wifi_test wlan0 country_set CN”切换国家，然后执行前文对应测试指令，假如 table 4 (40M)内的 SRRC 的信道 38(36, 以中心频点为准),值为 8，仪表端目标功率会是 8 左右。

powerlimit.txt 默认配置：

```
# Table 1:
## 2.4G, 20M,#5#
## START
## SRRC FCC ETSI JP UNSET
CH01 15 15 15 15 15
CH02 16 16 16 16 16
CH03 16 16 16 16 16
CH04 16 16 16 16 16
CH05 16 16 16 16 16
CH06 16 16 16 16 16
CH07 16 16 16 16 16
CH08 16 16 16 16 16
CH09 16 16 16 16 16
CH10 16 16 16 16 16
CH11 16 16 16 16 16
CH12 12 12 12 12 12
CH13 12 12 12 12 12
CH14 NA NA NA 12 NA
## END
```

```
# Table 2:
## 2.4G, 40M,#5#
## START
## SRRC FCC ETSI JP UNSET
CH01 NA NA NA NA NA
CH02 NA NA NA NA NA
CH03 16 16 16 16 16
CH04 16 16 16 16 16
CH05 16 16 16 16 16
CH06 16 16 16 16 16
CH07 16 16 16 16 16
```



```
CH08 16 16 16 16 16
CH09 16 16 16 16 16
CH10 16 16 16 16 16
CH11 16 16 16 16 16
CH12 NA NA NA NA NA
CH13 NA NA NA NA NA
CH14 NA NA NA NA NA
## END
```

Table 3:

5G, 20M, #5#

START

SRRC FCC ETSI JP UNSET

5G Band 1

```
CH36 15 16 16 16 15
CH40 15 16 16 16 15
CH44 15 16 16 16 15
CH48 15 16 16 16 15
```

5G Band 2

```
CH52 15 16 16 16 15
CH56 15 16 16 16 15
CH60 15 16 16 16 15
CH64 15 16 16 16 15
```

5G Band 3

```
CH100 NA 16 16 16 15
CH104 NA 16 16 16 15
CH108 NA 16 16 16 15
CH112 NA 16 16 16 15
CH116 NA 16 16 16 15
CH120 NA 16 16 16 15
CH124 NA 16 16 16 15
CH128 NA 16 16 16 15
CH132 NA 16 16 16 15
CH136 NA 16 16 16 15
CH140 NA 16 16 16 15
CH144 NA NA 16 16 15
```

5G Band 4

```
CH149 16 16 11 NA 11
CH153 16 16 11 NA 11
CH157 16 16 11 NA 11
CH161 16 16 11 NA 11
CH165 16 16 11 NA 11
```

END

Table 4:

5G, 40M, #5#

START

SRRC FCC ETSI JP UNSET

5G Band 1

```
CH38 15 16 16 16 15
CH46 15 16 16 16 15
```

5G Band 2

```
CH54 15 16 16 16 15
CH62 15 16 16 16 15
```

5G Band 3

```
CH102 NA 16 16 16 15
CH110 NA 16 16 16 15
```




```
CH118  NA  16  16  16  15
CH126  NA  16  16  16  15
CH134  NA  16  16  16  15
CH142  NA  16  NA  16  15
# 5G Band 4
CH151  16  16  11  NA  11
CH159  16  16  11  NA  11
##  END
```

Table 5:

5G, 80M, #5#

START

SRRC FCC ETSI JP UNSET

5G Band 1

CH42 15 16 16 16 15

5G Band 2

CH58 15 16 16 16 15

5G Band 3

CH106 NA 16 16 16 15

CH122 NA 16 16 16 15

CH138 NA 16 NA NA 15

5G Band 4

CH155 16 16 11 NA 11

END



2.1.10 天线增益

输出功率=目标功率 (pwr_lvl) -天线增益值

设置的天线增益值将降低目标功率，目标功率本身取决于国家/地区设置，请参阅国家/地区代码设置。天线增益值并非适用于所有国家/地区。

若要设置天线增益值则需要在 /firmware/aic8800/aic8800d80X2/aic_userconfig_8800d80X2.txt 文件设置 txpwr_loss

```
“# txpwr_loss
  loss_enable_2g4=0
  loss_value=2
  loss_enable_5g=0
  loss_value=2 “
```

loss_enable=1 天线增益设置使能

loss_enable=0 天线增益设置不使能

loss_value 天线增益设置的值

Eg: 若需要 2.4G 11b 1M 的输出功率为 10db

aic_userconfig_8800d80.txt 文件里面 txpwr_lvl 11b 1M 配置如下:

```
“# txpwr_lvl
  enable=1
  lvl_11b_11ag_1m_2g4=18“
```

txpwr_loss 配置应改为:

```
“# txpwr_loss
  loss_enable_2g4=1
  loss_value=8
  loss_enable_5g=0
  loss_value=2 “
```

$$\text{Output Power} = (\text{txpwr_lvl}) - (\text{txpwr_loss})$$

Note: In the 802.11ax specification, a device classified as Class A must achieve a transmit power accuracy of +/- 3 dB. Therefore, a transmitter with a 1 dB step size meets this requirement.

The modem features two gain adjustment blocks: coarse and fine. The fine gain adjustment step is 1 dB. The hardware design incorporates a 1 dB DSP module, which the digital front end reuses for the transmitter's fine gain adjustment. Consequently, the fine gain step remains at 1 dB.

2.1.11 物理层信息读取

对 sta 和 ap 的 phymode, bandwidth, freq, rssi, noise, txpower, chanutil(chan time & busy time), country code, 以及 tx/rx 物理层速率相关的信息读取, sta 模式不需要带 mac 地址, ap 模式的时候需要带对端 sta mac 地址, 不带有些项目会缺失比如 tx/rxrate=0:

```
./wifl_test wlp3s0 GET_CS_INFO 82:7B:19:02:DC:50
GET_CS_INFO:
phymode=4(0:B 1:G 2:A 3:N 4:AC 5:AX)
bandwidth=0(0:20 1:40 2:80)
freq=5180
rssi=-50
snr=49
noise=-99
txpwr=14
chan busy times=82/102(ms)
country code =00
rx nss=2, MCS=8
tx nss=2, MCS=9
```

2.1.12 产测底噪的读取

需要先发起一次 scan, 在 scan 的时候切换到每个信道上, 信道上没有 rx 的时候读取的底噪值, 需要有接收模式的判断, 可能稍有 delay, 正常的模式不打开, 需要的时候打开 Makefile 里的 CONFIG_READ_NOISE=y 去读取, 读取到的两列值分别是天线 0 和 1 的可参考的底噪值:

```
./wifl_test wlp3s0 READ_NOISE
```



三. WIFI_TEST编译说明

1. `sudo cp aic8800D80X2 /lib/firmware/ -r`
2. `make` 编译驱动
3. 插入 usb 板子, 按下 pwrkey
4. 输入 `lsusb`, 在 ubuntu 上能看到 ID 为 368b:8d90 的设备
5. `sudo insmod aic_load_fw.ko testmode=1`, `sudo insmod aic8800_fdrv.ko`(如果要从测试模式切换回正常模式, 请 `rmmmod wifi` 驱动后重新上电执行 `sudo insmod aic_load_fw.ko testmode=0`)
6. 加载完驱动输入 `lsusb`, 在 ubuntu 上能看到 ID 为 368b:8d99 的设备
7. 运行 `wifi_test`

例子 1: 可以连上 cable 测试

`set_tx 1 1 2 7 4000 // chan:1 bw:20m mode:2 rate:mcs7 length:4000byte`

```
liruizhe@aic:~/android_driver/USB/driver_fw/drivers/aic8800$ sudo wifi_test wlan0 set_tx 1 0 2 7 4096
set_tx:
done
liruizhe@aic:~/android_driver/USB/driver_fw/drivers/aic8800$
```

例子 2: 可以连上 cable 测试

`set_rx 14 1 // chan:14 bw:40m 开始接收`

`set_rxstop //停止接收`

`get_rx_result: // 1 秒内收到314个包,183 个正确`

```
liruizhe@aic:~/android_driver/USB/driver_fw/drivers/aic8800$ sudo wifi_test wlan0 set_rx 14 1
set_rx:
done
liruizhe@aic:~/android_driver/USB/driver_fw/drivers/aic8800$ sudo wifi_test wlan0 set_rxstop
set_rxstop:
done
liruizhe@aic:~/android_driver/USB/driver_fw/drivers/aic8800$ sudo wifi_test wlan0 get_rx_result
get_rx_result:
done: getrx fcsok=183, total=314
```

例子 3:

设置频偏校准:

`set_xtal_cap 6` 后晶体的寄存器值为 0x16, 设置为 1 后晶体的值为 0x18, 经过校准后, 最后一次显示的值就是校准完后需要配置的值。

```
liruizhe@aic:~/android_driver/USB/driver_fw/drivers/aic8800$ sudo wifi_test wlan0 set_xtal_cap 0
set_xtal_cap:
done:xtal cap: 0x10
liruizhe@aic:~/android_driver/USB/driver_fw/drivers/aic8800$ sudo wifi_test wlan0 set_xtal_cap 6
set_xtal_cap:
done:xtal cap: 0x16
liruizhe@aic:~/android_driver/USB/driver_fw/drivers/aic8800$ sudo wifi_test wlan0 set_xtal_cap 1
set_xtal_cap:
done:xtal cap: 0x17
liruizhe@aic:~/android_driver/USB/driver_fw/drivers/aic8800$
```

将校准后的值设置到硬件 efuse 里去:

```
liruizhe@aic:~/android_driver/USB/driver_fw/drivers/aic8800$ sudo wifi_test wlan0 set_freq_cal 17
set_freq_cal:
done: freq_cal: 0x17 (remain:0)
liruizhe@aic:~/android_driver/USB/driver_fw/drivers/aic8800$
```

例子 4: mac 地址的 efuse 写, 写完后读取一下:

```
liruizhe@aic:~/android_driver/USB/driver_fw/drivers/aic8800$ sudo wifi_test wlan0 get_mac_addr
get_mac_addr:
done: get macaddr = 00 : 00 : 00 : 00 : 00 : 00
      (remain:0)
liruizhe@aic:~/android_driver/USB/driver_fw/drivers/aic8800$
```

注 1:

以上是以 usb 平台为例，sdio 平台也类似，需要将 driver/rwnx_drv/fullmac/Makefile 的 CONFIG_USB_SUPPORT=n, CONFIG_SDIO_SUPPORT=y。用户空间的 aicrf_test 在客户平台上运行即可。

注 2:

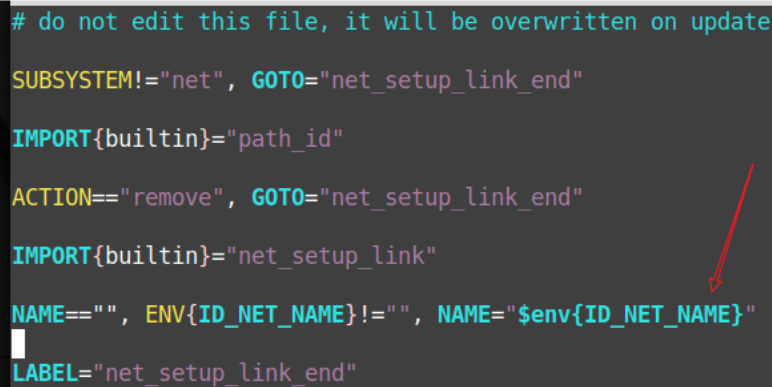
Ubuntu 平台建议做一下网络重命名规则，这样子 lsusb 后 aic8800 的芯片会显示成 wlan0，否则会用 mac 地址进行了重命名。

```
1 | cp /lib/udev/rules.d/80-net-setup-link.rules /etc/udev/rules.d/
```

然后执行如下命令，修改刚才复制过来的80-net-setup-link.rules文件：

```
1 | sudo vim /etc/udev/rules.d/80-net-setup-link.rules
```

如下图所示，将箭头所指的ID_NET_NAME改成ID_NET_SLOT即可。



```
# do not edit this file, it will be overwritten on update
SUBSYSTEM!="net", GOTO="net_setup_link_end"
IMPORT{builtin}="path_id"
ACTION=="remove", GOTO="net_setup_link_end"
IMPORT{builtin}="net_setup_link"
NAME="", ENV{ID_NET_NAME}!="", NAME="$env{ID_NET_NAME}"
LABEL="net_setup_link_end"
```

AIC Semiconductor



四. 测试示例:

WIFI TX SISO

11b-ANT0

wifi_test wlan0 set_tx 1 0 0 0 1000

wifi_test wlan0 set_ant 1

11b-ANT1

wifi_test wlan0 set_tx 1 0 0 0 1000

wifi_test wlan0 set_ant 2

11n-HT20 mcs7 ANT0

wifi_test wlan0 set_tx 1 0 2 7 4000

wifi_test wlan0 set_ant 1

11n-HT40 mcs7 ANT0

wifi_test wlan0 set_tx 1 1 2 7 4000

wifi_test wlan0 set_ant 1

11ac-VHT40 mcs9 ANT0

wifi_test wlan0 set_tx 1 1 4 9 8000

wifi_test wlan0 set_ant 1

11ax-HE80 mcs11 ANT0

wifi_test wlan0 set_tx 1 1 5 11 16000

wifi_test wlan0 set_ant 1

WIFI TX MIMO

11n-HT20 mcs15 ANT0

wifi_test wlan0 set_tx 1 0 2 15 4000

11ac-VHT40 mcs9 ANT0

wifi_test wlan0 set_tx 1 1 4 25 8000

11ax-HE80 mcs11 ANT0

wifi_test wlan0 set_tx 1 1 5 27 16000



WIFI RX

20M

```
wifi_test wlan0 set_rx 1 0  
wifi_test wlan0 set_rxstop  
wifi_test wlan0 get_rx_result
```

40M

```
wifi_test wlan0 set_rx 1 1  
wifi_test wlan0 set_rxstop  
wifi_test wlan0 get_rx_result
```

80M

```
wifi_test wlan0 set_rx 1 2  
wifi_test wlan0 set_rxstop  
wifi_test wlan0 get_rx_result
```

AIC Semiconductor Confidential 20250728