



AIC8800D80 射频测试说明

RF_TEST版本

版本号 v3.0

2023-2-2

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	2023 年 2 月 2 日	
V2.0	2023 年 7 月 19 日	更新信道补偿方式
V3.0	2024 年 1 月 24 日	发包间隔配置/增加 MAC 地址写入 Note



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 晶体频偏校准指令	6
2.1.3 读写 mac 地址.....	7
2.1.4 TX power 设置	8
2.1.5 信道功率补偿.....	9
2.1.6 userconfig 使用	11
三. WIFI_TEST 编译说明.....	13



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_MAC_ADDR,
 13. GET_MAC_ADDR,
 14. SET_BT_MAC_ADDR,
 15. GET_BT_MAC_ADDR,
 16. SET_POWER
 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) \\ 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

	mode	rate											
0	NON HT	0	1	2	3	4	5	6	7	8	9	10	11
		1M	2M	5.5M	11M	6M	9M	12M	18M	24M	36M	48M	54M
2	HT MF	0-7											
		mcs0-7											
4	VHT	0-9											
		mcs0-9											
5	HE SU	0-11											
		mcs0-11											

Length推荐值:

	20M	40M	80M
B/NON-HT	1024		
HT/VHT/HE	4096	8192	16384

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: μs

eg: wifi_test wlan0 set_tx 1 0 2 7 4096 \\ 2412MHz, HT 20 MCS7, length4096 (发包间隔默认值)

eg: wifi_test wlan0 set_tx 1 0 2 7 4096 1000 \\ 2412MHz, HT 20 MCS7, length4096 发包间1ms

- wifi_test wlan0 set_txstop \\ WiFi发射测试停止
no parameter

- 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的个数
6. 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



2.1.2 晶体频偏校准指令

AIC8800D80 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

2.1.3 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。



2.1.4 TX power设置

1. `wifi_test wlan0 rdwr_pwrmm val` \\切换功率设置模式
val: 0: `rdwr_pwrlvl`设置模式, 1: `set_power`设置模式
2. `wifi_test wlan0 set_power val` \\功率设置
val: 十进制
eg: `wifi_test wlan0 set_power 16` \\设置WiFi所有Rate的TX power 为16dBm

Note: 在evm达标的范围内设置power。

3. `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

Fmt\Idx	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

Fmt\Idx	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。

4. `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



2.1.5 信道功率补偿

1. wifi_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	1	11b	0	CH1~CH4	0	-7~7
				CH5~CH9	1	-7~7
				CH10~CH13	2	-7~7
		OFDM_highrate	1	CH1~CH4	0	-7~7
				CH5~CH9	1	-7~7
				CH10~CH13	2	-7~7
		OFDM_lowrate	2	CH1~CH4	0	-7~7
				CH5~CH9	1	-7~7
				CH10~CH13	2	-7~7
5G	2	OFDM_lowrate	0	CH36~CH50	0	-7~7
				CH51~CH64	1	-7~7
				CH98~CH114	2	-7~7
				CH115~CH130	3	-7~7
				CH131~CH146	4	-7~7
				CH147~CH166	5	-7~7
		OFDM_highrate	1	CH36~CH50	0	-7~7
				CH51~CH64	1	-7~7
				CH98~CH114	2	-7~7
				CH115~CH130	3	-7~7
				CH131~CH146	4	-7~7
				CH147~CH166	5	-7~7
		OFDM_midrate	2	CH36~CH50	0	-7~7
				CH51~CH64	1	-7~7
				CH98~CH114	2	-7~7
				CH115~CH130	3	-7~7
				CH131~CH146	4	-7~7
				CH147~CH166	5	-7~7

eg. wifi_test wlan0 rdwr_pwrofst 1 1 1 2 \\设置2.4G, 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 7, 最小-7, 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_6M, 11g_54M 校准 11b, ofdm_lowrate, ofdm_highrate 速率划分区间。在 ch1, ch7, ch13 校准信道划分区间。

5G 分别在 11a_6M, 11a_54M, 11ax_mcs11 校准 ofdm_lowrate, ofdm_midrate, ofdm_highrate 速率划分区间。在 ch42, ch58, ch106, ch122, ch138, ch155 校准信道划分区间。



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-LowRate						OFDM-highRate							
	BPSK	BPSK	QPSK	QPSK	16QAM	16QAM	64QAM	64QAM	64QAM	256QAM	256QAM	1024QAM	1024QAM	
	1/2	3/4	1/2	3/4	1/2	3/4	2/3	3/4	5/6	3/4	5/6	3/4	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-LowRate				OFDM-midRate					OFDM-highRate			
	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	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



2.1.6 userconfig 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_11ag_1m_2g4=18
lvl_11b_11ag_2m_2g4=18
lvl_11b_11ag_5m5_2g4=18
lvl_11b_11ag_11m_2g4=18
lvl_11b_11ag_6m_2g4=18
lvl_11b_11ag_9m_2g4=18
lvl_11b_11ag_12m_2g4=18
lvl_11b_11ag_18m_2g4=18
lvl_11b_11ag_24m_2g4=16
lvl_11b_11ag_36m_2g4=16
lvl_11b_11ag_48m_2g4=15
lvl_11b_11ag_54m_2g4=15
lvl_11n_11ac_mcs0_2g4=18
lvl_11n_11ac_mcs1_2g4=18
lvl_11n_11ac_mcs2_2g4=18
lvl_11n_11ac_mcs3_2g4=18
lvl_11n_11ac_mcs4_2g4=16
lvl_11n_11ac_mcs5_2g4=16
lvl_11n_11ac_mcs6_2g4=15
lvl_11n_11ac_mcs7_2g4=15
lvl_11n_11ac_mcs8_2g4=14
lvl_11n_11ac_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_11ac_mcs0_5g=18
lvl_11n_11ac_mcs1_5g=18
lvl_11n_11ac_mcs2_5g=18
lvl_11n_11ac_mcs3_5g=18
```



```
lvl_11n_11ac_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=0  
loss_value=2
```

```
# txpwr_ofst  
ofst_enable=0  
ofst_chan_1_4=0  
ofst_chan_5_9=0  
ofst_chan_10_13=0  
ofst_chan_36_64=0  
ofst_chan_100_120=0  
ofst_chan_122_140=0  
ofst_chan_142_165=0
```

```
# xtal cap  
xtal_enable=0  
xtal_cap=24  
xtal_cap_fine=31
```



三. WIFI_TEST编译说明

1. `sudo cp aic8800D80 /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 a69c:8d80 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. 运行 `wifi_test`

例子 1: 可以连上 cable 测试

`set_tx 1 1 2 7 4096` // chan:1 bw:20m mode:2 rate:mcs7 length:4096byte

```
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:

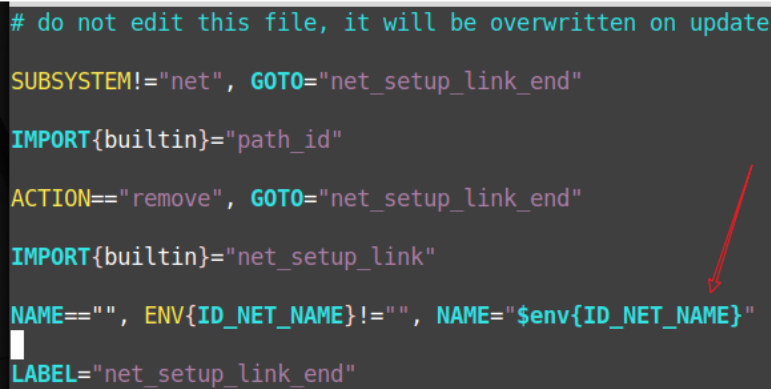
Ubuntu 平台建议做一下网络重命名规则, 这样子 lsusb 后 aic8800 的芯片会显示成 wlan0, 否则会用 mac 地址进行了重命名。

```
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"
```