MT76 Test Mode Programming Guide

Version: **2.3**
Release date: **2024-12-03**

Version History

Version	Date	Author	Description
1.0	2022-03-10	Shayne Chen	Official release
1.1	2022-08-15	StanleyYP Wang	Add pre-cal
1.2	2022-08-30	StanleyYP Wang	Add an iwpriv command mapping table and some description
1.3	2022-09-08	StanleyYP Wang	Add iBF
1.4	2022-10-07	StanleyYP Wang	Add bridge name config for atenl
1.5	2022-11-08	StanleyYP Wang	Add ZWDFS
1.6	2022-12-21	StanleyYP Wang	Add eBF certification & iBF calibration with golden
1.7	2023-02-23	Hugo Yang	Apply AIP format
1.8	2023-03-17	StanleyYP Wang	Add per-chip status
1.9	2023-05-10	StanleyYP Wang	Fix typo
1.10	2023-07-26	StanleyYP Wang	Add atenl eeprom iBF calibration sync command
1.11	2023-10-27	StanleyYP Wang	Add single sku enable command in testmode
1.12	2023-11-05	StanleyYP Wang	Add mt7981 support
2.0	2023-11-15	StanleyYP Wang	Add Wi-Fi 7 chipset (mt7996) support
2.1	2024-03-28	StanleyYP Wang	Modify 3.7.1 DUT Command for Calibration with Instrument to add missing command for iBF instrument calibration
2.2	2024-11-29	StanleyYP Wang	• Add Wi-Fi 7 chipset mt7992 support • Modify 2.2 MT76 Test to add mt76-test interface add/delete commands and add single wiphy support • Modify 2.3.7 Write Back EEPROM Data to Flash to add atenl eMMC write back support
2.3	2024-12-03	StanleyYP Wang	• Add 6.4 Abbreviations • Modify 2.3.7 Write Back EEPROM Data to Flash to add eMMC write back example

Table of Contents

1 Introduction

MT76 is a SoftMAC MediaTek open-source Wi-Fi driver developed based on the Linux kernel wireless subsystem.

The test mode part is implemented with NL80211_CMD_TESTMODE, a standard nl80211 command, which is different from the usage of the wireless extension in proprietary drivers.

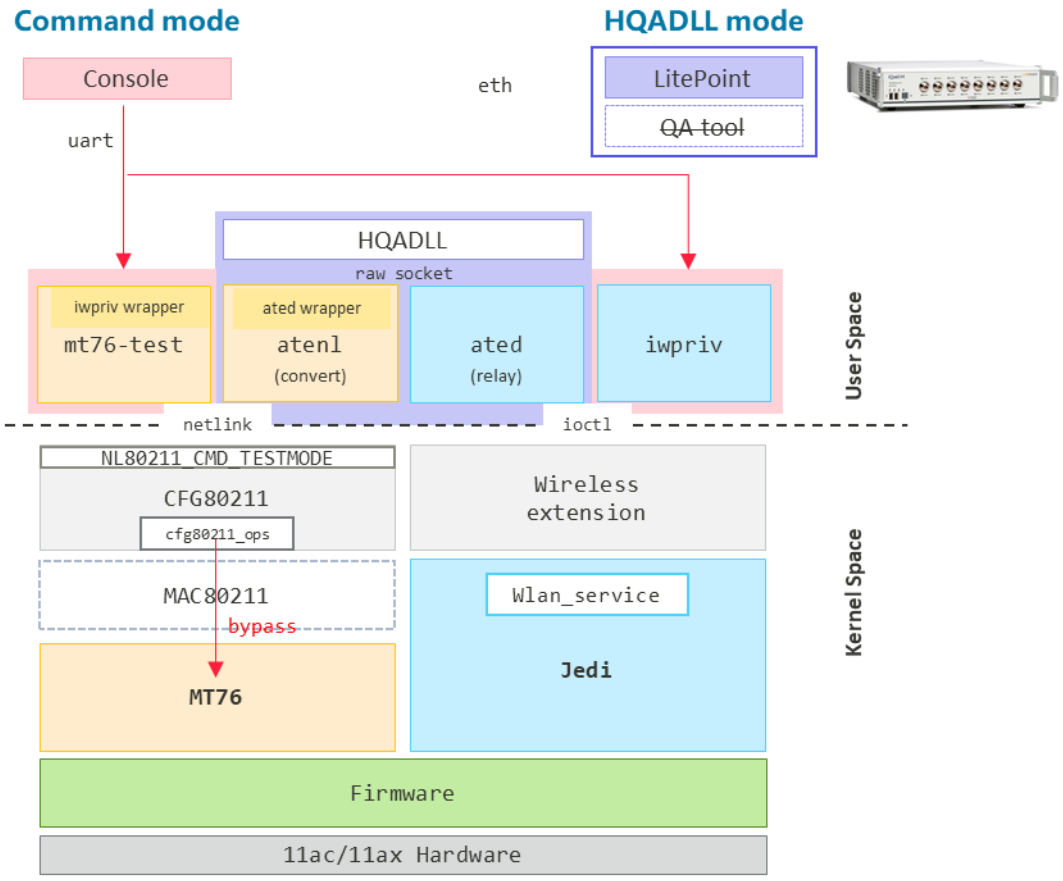
User application tools are provided to control testmode and get statistics. In Wi-Fi 6 chipsets, the tools for the proprietary driver communicate to the kernel stack with ioctl, while those for mt76 use generic netlink. For convenience and transparency, wrappers are provided to adapt to the original manual and HQADLL commands.

Note that the proprietary driver (logan) for Wi-Fi 7 chipsets also utilizes generic netlink (nl80211) instead of ioctl to communicate with the kernel stack. The iwpriv daemon is replaced by the mwctl daemon to handle user commands; however, iwpriv commands are also supported by utilizing symbolic links to link iwpriv and mwctl commands. **For Wi-Fi 7 mt76, we support both mwctl and iwpriv commands via our iwpriv wrapper.**

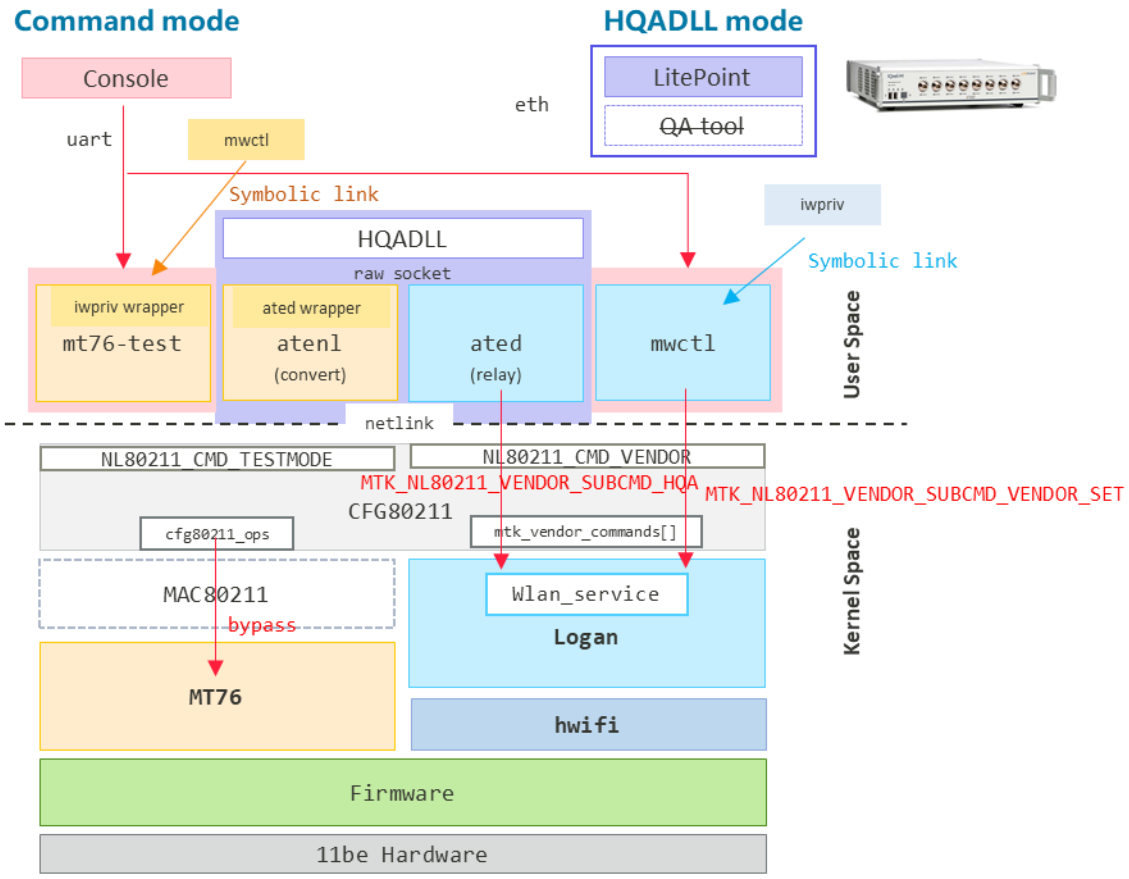
For QATool, please use the proprietary driver instead of MT76 upstream driver.

Below is an architectural overview for the comparison of mt76 and the proprietary driver.

Wi-Fi 6 Chipset:



Wi-Fi 7 Chipset:



2 Usage

2.1 Pre-setting

2.1.1 Set Country Code

If the test needs to bypass the DFS process or use some boundary channels, please switch to a customized regdomain before starting the test.

- Command

  `iw reg set VV`

- Example

  `iw reg get`

```
root@OpenWrt:/# iw reg set VV
root@OpenWrt:/# iw reg get
global
country VV: DFS-UNSET
        (2402 - 2494 @ 40), (N/A, 30), (N/A)
        (4910 - 4990 @ 80), (N/A, 30), (N/A)
        (5150 - 5875 @ 160), (N/A, 30), (N/A)
```

2.1.2 Check Firmware Mode (Mandatory for Wi-Fi 7)

In Wi-Fi 7 chipsets, due to the limitation of RAM size, we divide the WM firmware into two bins: the normal mode WM firmware bin and the test mode WM firmware bin. Therefore, before starting up, **please check that you are loading the test mode WM firmware bin via the following command or bootup log.**

- Command

  `cat /sys/kernel/debug/ieee80211/phy0/mt76/fw_version`

- Example

  (i) Debugfs command:

```
root@OpenWrt:/# cat sys/kernel/debug/ieee80211/phy0/mt76/fw_version
Version: 3.3.10.0
Rom Patch Build Time: 20230516165403a

WM Patch Build Time: 20230516165518, Mode: Testmode
WA Patch Build Time: 20230516165241
DSP Patch Build Time: 20230516165216
```

  (ii) Bootup log:

```
mt7996e 0000:01:00.0: WM_TM Firmware Version: ____000000, Build Time: 20230516165518
mt7996e 0000:01:00.0: DSP Firmware Version: ____000000, Build Time: 20230516165216
mt7996e 0000:01:00.0: WA Firmware Version: ____000000, Build Time: 20230516165241
```

2.1.3 Enter Test Mode Firmware Mode (Mandatory for Wi-Fi 7)

For Wi-Fi 7 chipsets, the driver will determine which WM firmware bin to load during bootup based on your EEPROM mode (flash, eFuse, binfile or default bin mode), EEPROM fields, and input module parameters.

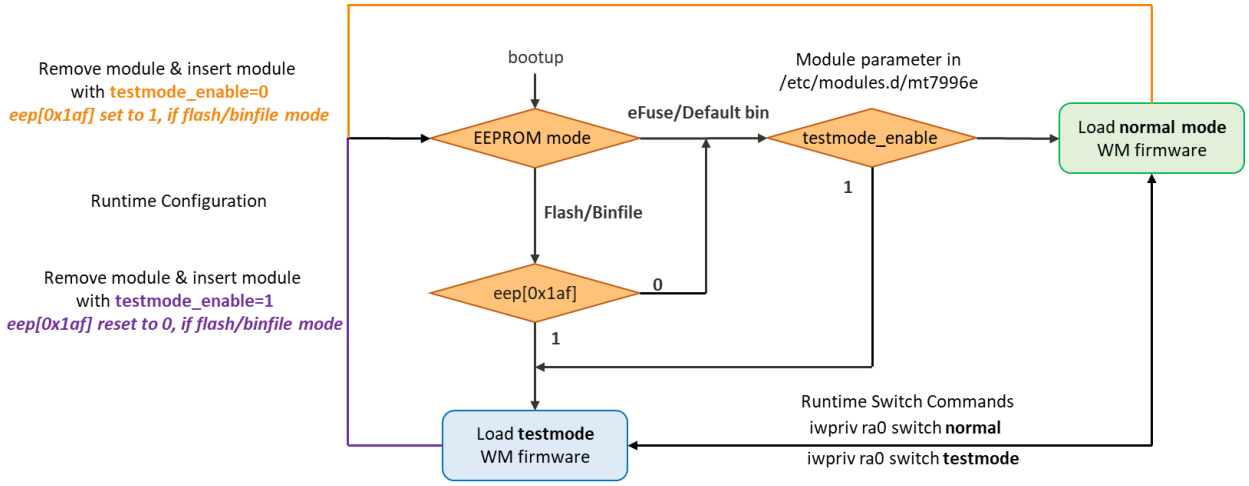
To check your current EEPROM mode, please enter the following commands:

`cat /sys/kernel/debug/ieee80211/phy0/mt76/eeprom_mode`

For more information about how to switch EEPROM modes, please refer to Section 6 of “MT76 Programming Guide.”

Please refer to the following descriptions or flowchart for the MT76 WM firmware loading flow.

- **Flash & binfile mode:**
  For flash and binfile mode, MT76 driver first checks eeprom field 0x1af (eep[0x1af]) to determine which bin to load. If eep[0x1af] = 1, then MT76 driver directly loads testmode firmware. Otherwise, MT76 driver further checks module parameter “testmode_enable” in /etc/module.d/mt7996e to decide which bin to load.
- **eFuse & default bin mode:**
  For eFuse and default bin mode, MT76 driver only checks module parameter “testmode_enable” in /etc/module.d/mt7996e to decide which bin to load.



There are two methods to **switch firmware mode at runtime**.

- Command

(i) Re-insert modules:

```
cat /sys/kernel/debug/ieee80211/phy0/mt76/eeprom_mode
// If eeprom mode == flash or binfile
atenl -i phy0 -c “eeprom set 0x1af=0x<val>”
atenl -i phy0 -c “eeprom precal sync”
atenl -i phy0 -c “sync eeprom all”

rmmod mt7996e
rmmod mt76-connac-lib
rmmod mt76
rmmod mac80211
rmmod cfg80211
rmmod compat
insmod compat
insmod cfg80211
insmod mac80211
insmod mt76
insmod mt76-connac-lib
insmod mt7996e testmode_enable=<val>
sleep 5
killall hostapd
```

```
killall netifd
```

(ii) iwpriv wrapper commands (wraps the above commands) **[Recommended]:**

```
iwpriv ra0 switch <testmode/normal>
```

- Example

```
root@OpenWrt:/# iwpriv ra0 switch testmode
set offset 0x1af[  126.979615] mt7996e 0000:01:00.0: Not pre-cal yet!
 to 0x1
[  126.984898] mt7996e 0000:01:00.0: Not pre-cal yet!
No Pre cal data or info!
No Pre cal data or info!
Unlocking Factory ...

Writing from /tmp/atenl-eeprom-phy0 to Factory ...
[  128.017576] Loading modules backported from Linux version v6.1.24-0-g0102425ac76b
[  128.025064] Backport generated by backports.git v5.15.92-1-44-gd6ea70fafd36
[  128.055082] mt7996e_hif 0001:01:00.0: assign IRQ: got 119
[  128.060545] mt7996e_hif 0001:01:00.0: enabling bus mastering
[  128.066297] mt7996e 0000:01:00.0: assign IRQ: got 120
[  128.071359] mt7996e 0000:01:00.0: enabling bus mastering
[  128.132470] mt7996e 0000:01:00.0: attaching wed device 0 version 3
[  128.170487] platform 15010000.wed: WO Firmware Version: ____000000, Build Time: 20230218204509
[  128.225165] mt7996e_hif 0001:01:00.0: attaching wed device 1 version 3
[  128.336273] mt7996e 0000:01:00.0: HW/SW Version: 0x8a108a10, Build Time: 20230516165403a
[  128.336273]
[  128.437691] mt7996e 0000:01:00.0: WM_TM Firmware Version: ____000000, Build Time: 20230516165518
[  128.475555] mt7996e 0000:01:00.0: DSP Firmware Version: ____000000, Build Time: 20230516165216
[  128.493237] mt7996e 0000:01:00.0: WA Firmware Version: ____000000, Build Time: 20230516165241
root@OpenWrt:/#
```

2.2 MT76 Test

mt76-test is a user application tool for test mode manual commands.

2.2.1 Start Test Mode

MT76 starts by adding a monitor interface.
Note that for **DBDC band1**, please change to **phy1/wlan1/phy1-ap0/mon1**.

Important Note:

- **Please delete all the non-monitor interfaces before starting up the test mode monitor interface.**
- **iwpriv ra0 set ATE=ATESTART will handle all the interface deletion in the iwpriv wrapper.**
- **mt76-test phyX add/del ${interface} is a newly added command to support test mode interface addition/deletion for Wi-Fi 7 single Wiphy model. It can also be used in Wi-Fi 6 projects. For the sake of convenience, this command sets the country code to VV, activates the monitor interface, and sets the test mode state to idle. Therefore, users can start the test mode right after entering this command.**

- Command

(i) Use the mt76-test command [Recommended for both non-single/single Wiphy]

```
iw dev ${non-monitor interface name} del
mt76-test phy0 add mon0
```

(ii) Use the iw command [non-single Wiphy]

```
iw phy phy0 interface add mon0 type monitor
iw dev ${non-monitor interface name} del
ifconfig mon0 up
```

(iii) Use the iw command [single Wiphy]

```
iw phy phy0 interface add mon0 type monitor radios <radio_idx>
# <radio_idx> = 0, 1, 2 for band 0, 1, 2
iw dev ${non-monitor interface name} del
ifconfig mon0 up
```

- Example

```
root@OpenWrt:/# ifconfig mon0
mon0      Link encap:UNSPEC  HWaddr 00-0C-43-2A-57-A9-01-00-00-00
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1768 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:566960 (553.6 KiB)  TX bytes:0 (0.0 B)
```

2.2.2 Stop Test Mode

Remove monitor interface.

Note that the mt76-test command sets the testmode state to “off” before deleting the interface. This help to avoid SIOCSIFFLAGS errors.

- Command

  (i) Use the mt76-test command [Recommended]

  ```
  mt76-test phy0 del mon0
  ```

  (ii) Use the iw command

  ```
  mt76-test phy0 set state=off
  iw dev mon0 del
  iw phy phy0 interface add phy0-ap0 type managed
  ```

2.2.3 Start and Stop TX

- Command
  - Start TX

    ```
    mt76-test phy0 set state=tx_frames
    ```

  - Stop TX

    ```
    mt76-test phy0 set state=idle
    ```

- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep state
state=tx_frames
root@OpenWrt:/# mt76-test phy0 dump | grep state
state=idle
```

3.1 TX VHT40 MCS9 ANT1 Band0

3.2 TX HE80 MCS11 ANT8 Band1

3.4 Duplicate TX

2.2.4 Start and Stop RX

- Command
  - Start RX

    ```
    mt76-test phy0 set state=rx_frames
    ```

  - Stop RX

    ```
    mt76-test phy0 set state=idle
    ```

- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep state
state=rx_frames
root@OpenWrt:/# mt76-test phy0 dump | grep state
state=idle
```

3.3 RX VHT20 ANT3 Band0

2.2.5 Start and Stop Continuous TX

- Command
  - Start continuous TX

    `mt76-test phy0 set state=tx_cont`
  - Stop continuous TX

    `mt76-test phy0 set state=idle`
- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep state
state=tx_cont
```

3.5 Continuous TX

2.2.6 Set Band Index

MT76 uses phyX to control testmode on each band. For example:

```
# band 0
iw dev phy0-ap0 del
mt76-test phy0 add mon0
or
iw phy phy0 interface add mon0 type monitor
iw dev phy0-ap0 del
ifconfig mon0 up

# band 1
iw dev phy1-ap0 del
mt76-test phy1 add mon1
or
iw phy phy1 interface add mon1 type monitor
iw dev phy1-ap0 del
ifconfig mon1 up
```

2.2.7 Set Channel and Bandwidth

MT76 utilizes iw commands to configure channels and bandwidth. Note that setting the 6G channel (BW 320) is currently not supported in the iw command[1]. We suggest using the first **iw set frequency command** in every case. Iwpriv wrapper handles the 6G case correctly, so feel free to use the iwpriv command to set channel and BW.

Note:

[1] iw version: 5.19 for Wi-Fi 6, 6.9 for Wi-Fi 7

1. Currently only the first command supports 6G BW320.
2. In the first command, please specify the center frequency to select BW320-1 or BW320-2.
3. **For HE/EHT & bandwidth > 20, LPDC is a must for transmitting packets.** If LPDC is not set, the firmware will block the TX.

- Command

  iw dev mon0 set freq <control freq> [5|10|20|40|80|80+80|160|320] [<center1_freq> [<center2_freq>]]

  iw dev mon0 set set freq <freq>
  [NOHT|HT20|HT40+|HT40-|5MHz|10MHz|80MHz|160MHz|320MHz]
  iw dev mon0 set channel <channel> [NOHT|HT20|HT40+|HT40-|5MHz|10MHz|80MHz|160MHz]

- Example

```
root@OpenWrt:/# iw dev mon0 set channel 7 HT20
root@OpenWrt:/# iw mon0 info
Interface mon0
        ifindex 19
        wdev 0x2
        addr 00:0c:43:2b:76:d7
        type monitor
        wiphy 0
        channel 7 (2442 MHz), width: 20 MHz, center1: 2442 MHz
        txpower 27.00 dBm
```

2.2.8 Set TX Count

Set the total number of TX packets.

Input Argument	Description	Value
tx_count	Set the total number of TX packets	[1, UINT_MAX]

- Command

  mt76-test phy0 set tx_count=10000000

- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep tx_count
tx_count=10000000
```

2.2.9 Set TX Length

Set the length of an MPDU. The maximum length is determined by the TX rate mode and the chip's capability.

Input Argument	Description	TX rate mode	Value
tx_length	Set the length of the MPDU	CCK, OFDM	[30, 2352]
		HT	[30, 7935]
		VHT, HE_SU, HE_EXT_SU, HE_TB, HE_MU	[30, 7991] (for mt7915)
			[30, 11454]
		EHT_SU, EHT_TB, EHT_MU	[30, **UINT_MAX**] (for Wi-Fi 7 only)

Note:

Description	Value
The length of the IEEE packet header	30
IEEE80211_MAX_FRAME_LEN	2352
IEEE80211_MAX_MPDU_LEN_HT	7935

Description	Value
IEEE80211_MAX_MPDU_LEN_VHT	7991 (for mt7915)
	11454

- Command

```
mt76-test phy0 set tx_length=1024
```

- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep tx_length
tx_length=1024
```

2.2.10 Set Antenna

Set antenna mask (bitmap representation) for both the TX and RX paths.

Note that bit 4 is used exclusively for the following chips.

1. mt7996 BE19000 6G band 4T5R
2. mt7992 BE7200 eFEM 5G band 4T5R, mt7992 BE7200 2i5e 5G band 4T5R
3. mt7992 BE7200 iFEM 5G band 5T5R

Input Argument	Description	Value
tx_antenna	Bit 0: enable TX/RX antenna 0 Bit 1: enable TX/RX antenna 1 Bit 2: enable TX/RX antenna 2 Bit 3: enable TX/RX antenna 3 **Bit 4: enable TX/RX antenna 4**	$[0, 2^{nss} - 1]$ nss = the number of tx/rx streams written in eeprom

- Command

```
# Enable antenna 1 & 2
mt76-test phy0 set tx_antenna=6
# Enable antenna 3 only
mt76-test phy0 set tx_antenna=8
```

- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep tx_antenna
tx_antenna=8
```

2.2.11 Set Spatial Extension Index

Set spatial extension index for TX. When not all antennas are enabled, the spatial extension index is used to prioritize which antenna to use for transmission. Please refer to the appendix Spatial Extension Index Table for more information.

Input Argument	Description	Value
tx_spe_idx	Determine the priority of each antenna	[0, 27]

- Command

```
mt76-test phy0 set tx_spe_idx=24
```

- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep tx_spe_idx
tx_spe_idx=24
```

2.2.12 Set Guard Interval and Long Training Field

Set the Guard Interval (GI) and Long Training Filed (LTF) based on the current rate mode.

Wi-Fi 6 chipset (mt7915, mt7916, mt7981, mt7986):

Input Argument	Description	Value
tx_rate_sgi	Long GI/Short GI *for HT/VHT* 0.8μs, 1.6μs, 3.2μs GI *for HE*	0, 1 0, 1, 2
tx_ltf	1x, 2x, 4x LTF	0, 1, 2

Wi-Fi 7 chipset (mt7996, mt7992):

Input Argument	Description	Value
tx_rate_sgi	In the Wi-Fi 7 chipset, tx_rate_sgi becomes an index to a specific GI & LTF combination based on your TX rate mode. Please refer to the table for more information.	[0, 4]
tx_ltf	Not used	-

Please follow the standard shown in the table below to set the correct GI and LTF for different modes. Cases not listed in the following table are not supported.

Note that the input argument tx_ltf is not used for the Wi-Fi 7 chipset, since testmode is offloaded to firmware. Therefore, please refer to the correct table based on your chip ID.

Wi-Fi 6 chipset (mt7915, mt7916, mt7981, mt7986):

TX Rate Mode	Type	Input Argument	Value
HT, VHT	Long GI	tx_rate_sgi	0
	Short GI		1
HE_SU HE_EXT_SU	1x LTF + 0.8μs GI	tx_ltf, tx_rate_sgi	0, 0
	2x LTF + 0.8μs GI		1, 0
	4x LTF + 0.8μs GI		2, 0
	2x LTF + 1.6μs GI		1, 1
	4x LTF + 3.2 μsGI		2, 1
HE_MU	2x LTF + 0.8μs GI		1, 0
	4x LTF + 0.8μs GI		2, 0
	2x LTF + 1.6μs GI		1, 1
	4x LTF + 3.2μs GI		2, 2
HE_TB	1x LTF + 1.6μs GI		0, 1
	2x LTF + 1.6μs GI		1, 1
	4x LTF + 3.2μs GI		2, 2

Wi-Fi 7 chipset (mt7996, mt7992):

Input Argument	TX Rate Mode	Type	Value
tx_rate_sgi	HT, VHT	Long GI	0
		Short GI	1
	HE_SU HE_EXT_SU	1x LTF + 0.8μs GI	0
		2x LTF + 0.8μs GI	1
		2x LTF + 1.6μs GI	2
		4x LTF + 3.2 μsGI	**3**
		4x LTF + 0.8μs GI	**4**

Input Argument	TX Rate Mode	Type	Value
	HE_MU	4x LTF + 0.8μs GI	0
		2x LTF + 0.8μs GI	1
		2x LTF + 1.6μs GI	2
		4x LTF + 3.2μs GI	3
	HE_TB	1x LTF + 1.6μs GI	0
		2x LTF + 1.6μs GI	1
		4x LTF + 3.2μs GI	2
	EHT_SU EHT_MU	2x LTF + 0.8μs GI	0
		2x LTF + 1.6μs GI	1
		4x LTF + 0.8μs GI	2
		4x LTF + 3.2μs GI	3
	EHT_TB	1x LTF + 1.6μs GI	0
		2x LTF + 1.6μs GI	1
		4x LTF + 3.2μs GI	2

- Command

```
# 2x LTF + 1.6μs GI for HE_SU
mt76-test phy0 set tx_rate_sgi=1 tx_ltf=1
```

- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep 'tx_ltf¥|tx_rate_sgi'
tx_rate_sgi=1
tx_ltf=1
```

2.2.13 Set TX Rate Mode

Set the rate mode for packet TX.

Input Argument	Description		Value
tx_rate_mode	802.11b		cck
	802.11g		ofdm
	802.11b/g/n		ht
	802.11ac		vht
	802.11ax	Single User	he_su
		Extended Range PPDU	he_ext_su
		Multiple User	he_tb
		Trigger-Based PPDU	he_mu
	802.11be	Single User	eht_su
		Multiple User	eht_mu
		Trigger-Based PPDU	eht_tb

Note:

- **Green field is not supported in mt76**

- Command

```
mt76-test phy0 set tx_rate_mode=he_su
```

- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep tx_rate_mode
tx_rate_mode=he_su
```

2.2.14 Set TX Rate Index (MCS)

Set the MCS value for packet TX.

Note:

- The data rate for VHT & HE is determined by NSS & MCS value

For more information, please refer to MCS table.

Input Argument	Description	TX Rate Mode	Value
tx_rate_idx	Set the MCS value	CCK, OFDM	[0, 3], 1SS only
		OFDM	[0, 7], 1SS only
		HT	[0, 31], 1 ~ 4SS
		VHT	[0, 9] ※
		HE	[0, 11] ※
		EHT	[0, 15]

- Command

  `mt76-test phy0 set tx_rate_idx=9`

- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep tx_rate_idx
tx_rate_idx=9
```

2.2.15 Set Spatial Stream Number

Set the number of spatial streams for packet TX in VHT/HE mode.

Input Argument	Description	Value
tx_rate_nss	Set the number of spatial streams	[1, 4]

- Command

  `mt76-test phy0 set tx_rate_nss=2`

- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep tx_rate_nss
tx_rate_nss=2
```

2.2.16 Set LDPC

Use low density parity check (LDPC) code for packet TX. **Note that LDPC is mandatary on HE mode with BW larger than 20 MHz.**

Input Argument	Description	Value
tx_rate_ldpc	Disable/Enable LDPC	0/1

- Command

  `mt76-test phy0 set tx_rate_ldpc=1`

- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep tx_rate_ldpc
tx_rate_ldpc=1
```

2.2.17 Set STBC

Use space time block coding (STBC) code for packet TX.

Input Argument	Description	Value
tx_rate_stbc	Disable/Enable STBC	0/1

- Command
  `mt76-test phy0 set tx_rate_stbc=1`
- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep tx_rate_stbc
tx_rate_stbc=1
```

2.2.18 Set TX Power

Set TX power of a single antenna. Note that mt7915/mt7916/mt7981/mt7986 only support setting antenna 0, and the TX power of antenna 1~3 will be the same as the one of antenna 0.

The TX power can be checked by

1. cat /sys/kernel/debug/ieee80211/**phyX**/mt76/txpower_sku, for Wi-Fi 6 & Wi-Fi 7 multi-Wiphy
2. cat /sys/kernel/debug/ieee80211/**phy0**/mt76/**bandX**/txpower_sku, for Wi-Fi 7 single Wiphy

Input Argument	Description	Value (Unit: 0.5dB)
tx_power	Set the power of antenna 0	[0, 63], decimal
	Set the power of antenna 1	
	Set the power of antenna 2	
	Set the power of antenna 3	

- Command
  `mt76-test phy0 set tx_power=38,0,0,0`
- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep tx_power
tx_power=38,0,0,0
```

2.2.19 Set Packet TX Time

Set the expected packet TX time (μsec). Note that if this value is set, **packet TX length will be recalculated** and ignore the original tx_length value.

Input Argument	Description	Value (Unit: µs)
tx_time	Set the TX frame transmission time	The range decided by rate mode/data rate, and BW when in VHT mode

- Command
  `mt76-test phy0 set tx_time=200`
- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep tx_time
tx_time=200
```

2.2.20 Set Inter-packet Gap

Set the inter-packet gap of TX frame (μsec). The minimum value of tx_ipg should be larger than the sum of default value of SIG_EXT, SIFS, and slot time listed in the table below. Otherwise, tx_ipg would be reset to 0.

Terminology	Description	Default Value (unit: µs)		
SIG_EXT	-	CCK		0
		Others		6
SIFS	A short interframe space defined in CSMA/CA to avoid confliction	10		
Slot time	The time that should elapse between a first electronic pulse being sent and a second one following it. In CSMA/CA, a slot time includes time for signal propagation in the air, clear channel assessment (CCA), and hardware to turn from RX to TX	Propagation	1	9
		CCA	4	
		RX/TX hardware turnaround	4	

Input Argument	Description	Value (unit: µs)
tx_ipg	Set the TX frame inter-packet gap	[25, 590000]

- Command
  `mt76-test phy0 set tx_ipg=50`
- Example
```
root@OpenWrt:/# mt76-test phy0 dump | grep tx_ipg
tx_ipg=50
```

2.2.21 Set Duty Cycle

Set the duty cycle of the TX frame, which can be calculated from tx_time and tx_ipg. Note that if both IPG and TX time are set, then the TX duty cycle will be determined.

$$tx\_duty\_cycle = \frac{tx\_time}{tx\_time + tx\_ipg}$$

Note that this command is not available in the Wi-Fi 7 chipset.

Input Argument	Description	Value
tx_duty_cycle	Set the duty cycle of TX frame	[0, 99], percentage

- Command
  `mt76-test phy0 set tx_duty_cycle=50`
- Example
```
root@OpenWrt:/# mt76-test phy0 dump | grep tx_duty_cycle
tx_duty_cycle=50
```

2.2.22 Set TX Frequency Offset

Set the RF frequency offset.

Input Argument	Description	Value
freq_offset	Set the offset of the TX RF frequency	[0, 127], decimal

- Command
  `mt76-test phy0 set freq_offset=42`

2.2.23 Set MAC Address

Set the MAC address of address 1, 2 and 3 in the MAC packet frame.

Input Argument		Description	Value
mac_addrs	DA (1st address)	Destination MAC address	xx:xx:xx:xx:xx:xx
	SA (2nd address)	Source MAC address	xx:xx:xx:xx:xx:xx
	BSSID (3rd address)	BSSID MAC address	xx:xx:xx:xx:xx:xx

- Command
  `mt76-test phy0 set mac_addrs=00:11:22:33:44:55,11:22:33:44:55:66,22:33:44:55:66:77`
- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep mac_addrs
mac_addrs=00:11:22:33:44:55,11:22:33:44:55:66,22:33:44:55:66:77
```

2.2.24 Set AID for Virtual WTBL

Set association ID (AID) for starting up virtual WTBL (Wireless Lan Table, storing capacity or information for the connected peer). **For mt7916, mt7981 and mt7986, WCID should not be 0 for TX**. Otherwise, the packet would be dropped by WA firmware.

Input Argument	Description	Value
aid	Set the association ID	[0, 16]

- Command
  `mt76-test phy0 set aid=1`
- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep aid
aid=1
```

2.2.25 Set RU Index

Set the RU index. Please refer to RU Index Table 9-29i or Data and Pilot Subcarrier Indices Table 27-7 to Table 27-9 for the indexing method. Note that the ru_idx here is not the RU index listed in the table of the link above. It is the “B7-B1 of the RU Allocation subfield”, which ranges from 0 to 68, in the table. The difference between the RU index and “B7-B1 of the RU Allocation subfield” listed in the table of the link above is that “B7-B1 of the RU Allocation subfield” is the accumulated index of RU index for each bandwidth.

Note that this command is not available in the Wi-Fi 7 chipset.

Input Argument	Description	Value
ru_idx	Set the RU index	[0, 68]

- Command
  `mt76-test phy0 set ru_idx=1`
- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep ru_idx
ru_idx=1
```

2.2.26 Set RU Allocation

Set the resource unit (RU) allocation subfield (8 bits) in HE-SIG-B (HE-MU PPDU for 802.11ax). The RU allocation subfield indicates the RU assignment, including the size of the RU(s) and their placement in the frequency domain.
Note that this command is not available in the Wi-Fi 7 chipset.

For more information, please refer to RU allocation Table 27-26.

Note:

- ru_idx is used to select a specific RU to TX/RX in 20/40/80 MHz bandwidth. Please refer to 2.2.25 Set RU Index for the indexing method.
- ru_alloc is used to specify the RU allocation in a 20 MHz bandwidth.

Input Argument	Description	Value
ru_alloc	Set the RU allocation subfield (8 bits)	[0, 255]

- Command
  `mt76-test phy0 set ru_alloc=1`
- Example

```
root@OpenWrt:/# mt76-test phy0 dump | grep ru_alloc
ru_alloc=1
```

2.2.27 Dump Settings

Dump the current configured settings.

- Command
  `mt76-test phy0 dump`
- Example

```
root@OpenWrt:/# mt76-test phy0 dump
state=off
tx_count=1
tx_length=1024
tx_rate_mode=ofdm
tx_rate_nss=1
tx_rate_idx=0
tx_rate_sgi=0
tx_rate_ldpc=0
tx_rate_stbc=1
tx_duty_cycle=50
tx_ipg=50
tx_time=200
```

2.2.28 Dump Statistics

Show current TX/RX status.

- Command
  `mt76-test phy0 dump stats`
- Example

```
root@OpenWrt:/# mt76-test phy0 dump stats
tx_pending=972233
tx_queued=27767
tx_done=22936
rx_packets=152
rx_fcs_error=152
last_freq_offset=0
last_rcpi=0,0,0,0
last_ib_rssi=0,0,0,0
last_wb_rssi=0,0,0,0
last_snr=0
rx_per=100.00%
```

2.2.29 Group Pre-calibration

Do group pre-calibration (including RX DCOC, RSSI DCOC, TX TSSI DCOC, TX LPFG, TX FDIQ, TX DCIQ, RX FDIQ, RX FIIQ, ADCDCOC) and save the result in mt76 driver. To write back the result to flash, please use atenl commands. For more information, please refer to Pre-cal example.

Note:

4. Please make sure you are in flash mode or bin file mode.
5. group_prek_clean will only clean the pre-cal data stored in the mt76 driver. If you want to clean the pre-cal data in flash memory, please use atenl commands.

- Command
  `mt76-test phy0 set state=group_prek`
  `mt76-test phy0 set state=group_prek_dump`
  `mt76-test phy0 set state=group_prek_clean`
- Example

```
root@OpenWrt:/# iwpriv phy0 set ATE=ATESTART
root@OpenWrt:/# mt76-test phy0 set state=group_prek
root@OpenWrt:/# mt76-test phy0 set state=group_prek_dump
[ 1067.338707] mt7915e 0000:01:00.0: Group Pre-Cal:
[ 1067.343342] mt7915e 0000:01:00.0: [0x00000000] 0x       0 0x       1 0xffffbfff 0x 10003ff
[ 1067.351598] mt7915e 0000:01:00.0: [0x00000010] 0x  300080 0x40023f90 0xfc000000 0xffffffff
[ 1067.359846] mt7915e 0000:01:00.0: [0x00000020] 0xf800000f 0x23f003ff 0x  3f9020 0xfffe05fd
[ 1067.368094] mt7915e 0000:01:00.0: [0x00000030] 0x       0 0x 3010100 0x408fe404 0xffbf8000
[ 1067.376344] mt7915e 0000:01:00.0: [0x00000040] 0x3fffc03f 0xc0c04000 0x  13fa80 0x ffff000
[ 1067.384592] mt7915e 0000:01:00.0: [0x00000050] 0x      10 0x40402010 0x 810fdd0 0xfbfbfc00
[ 1067.392840] mt7915e 0000:01:00.0: [0x00000060] 0x     3ff 0x     400 0xffff3fd8 0x       0
[ 1067.401088] mt7915e 0000:01:00.0: [0x00000070] 0x       0 0xff000000 0x3fff8ffe 0x       0
[ 1067.409338] mt7915e 0000:01:00.0: [0x00000080] 0x       0 0x       0 0x     400 0x       0
[ 1067.417585] mt7915e 0000:01:00.0: [0x00000090] 0x       0 0x       0 0x       0 0x       0
[ 1067.425834] mt7915e 0000:01:00.0: [0x000000a0] 0x       0 0x       0 0x       1 0xf0007ffe
[ 1067.434080] mt7915e 0000:01:00.0: [0x000000b0] 0x11fff3ff 0xffeffdc0 0x40173f93 0x13ff5006
[ 1067.442329] mt7915e 0000:01:00.0: [0x000000c0] 0xfe7fff00 0x13fc001f 0xcc37f800 0x b4007df
[ 1067.450575] mt7915e 0000:01:00.0: [0x000000d0] 0x   3f40f 0x 20001fc 0xfb08fe00 0xc34fefff
[ 1067.458822] mt7915e 0000:01:00.0: [0x000000e0] 0xc000ff03 0x  ff803e 0xfd83bf80 0x2103fbfe
[ 1067.467068] mt7915e 0000:01:00.0: [0x000000f0] 0xc0003f61 0x104ff01f 0x8f40efe0 0x4434ff3f
[ 1067.475314] mt7915e 0000:01:00.0: [0x00000100] 0xec0013d8 0x 827f80b 0xbba457f4 0x160f3ffb
[ 1067.483560] mt7915e 0000:01:00.0: [0x00000110] 0xfa0005ed 0x 207fe02 0xf3ed18fe 0x86048ff2
[ 1067.491806] mt7915e 0000:01:00.0: [0x00000120] 0xbec041fb 0x       0 0x       0 0x     400
[ 1067.500052] mt7915e 0000:01:00.0: [0x00000130] 0x       0 0x       0 0x       0 0x       0
```

3.6 Pre-calibration

2.2.30 DPD/Flatness Pre-calibration

Do DPD/Flatness pre-calibration (including TX DPD, TX Flatness) and save the result in mt76 driver. To write back the result to flash, please use atenl commands. For more information, please refer to Pre-cal example.

Note:

1. Please make sure you are in flash mode or bin file mode.
2. dpd_clean will only clean the pre-cal data stored in the mt76 driver. If you want to clean the pre-cal data in flash memory, please use atenl commands.
3. For DBDC chips, please **select the corresponding phy index**. Otherwise, the command will be blocked. For example, in order to perform 5G/6G DPD pre-calibration in AX7800, please enter:

```
mt76-test phy3 set state=dpd_5g
mt76-test phy1 set state=dpd_6g
```

- Command

```
mt76-test phy0 set state=dpd_2g
mt76-test phy0 set state=dpd_5g
mt76-test phy0 set state=dpd_6g
mt76-test phy0 set state=dpd_dump
mt76-test phy0 set state=dpd_clean
```

- Example

```
root@OpenWrt:/# mt76-test phy1 set state=dpd_6g
root@OpenWrt:/# mt76-test phy1 set state=dpd_dump
[ 2476.679980] mt7915e 0000:01:00.0: DPD Pre-Cal:
[ 2476.684482] mt7915e 0000:01:00.0: [0x00017810] 0x       0 0x       0 0x       0 0x       0
[ 2476.692734] mt7915e 0000:01:00.0: [0x00017820] 0x       0 0x       0 0x       0 0x       0
[ 2476.700982] mt7915e 0000:01:00.0: [0x00017830] 0x       0 0x       0 0x       0 0x       0
[ 2476.709232] mt7915e 0000:01:00.0: [0x00017840] 0x       0 0x       0 0x       0 0x       0
[ 2476.717478] mt7915e 0000:01:00.0: [0x00017850] 0x       0 0x       0 0x       0 0x       0
[ 2476.725728] mt7915e 0000:01:00.0: [0x00017860] 0x       0 0x       0 0x       0 0x       0
[ 2476.733977] mt7915e 0000:01:00.0: [0x00017870] 0x       0 0x       0 0x       0 0x       0
[ 2476.742228] mt7915e 0000:01:00.0: [0x00017880] 0x       0 0x       0 0x       0 0x       0
[ 2476.750475] mt7915e 0000:01:00.0: [0x00017890] 0x       0 0x       0 0x       0 0x       0
[ 2476.758739] mt7915e 0000:01:00.0: [0x000178a0] 0x       0 0x       0 0x       0 0x       0
[ 2476.766996] mt7915e 0000:01:00.0: [0x000178b0] 0x       0 0x       0 0x       0 0x       0
[ 2476.775262] mt7915e 0000:01:00.0: [0x000178c0] 0x       0 0x       0 0x       0 0x       0
[ 2476.783510] mt7915e 0000:01:00.0: [0x000178d0] 0x       0 0x       0 0x       0 0x       0
[ 2476.791758] mt7915e 0000:01:00.0: [0x000178e0] 0x       0 0x       0 0x       0 0x       0
[ 2476.800007] mt7915e 0000:01:00.0: [0x000178f0] 0x       0 0x       0 0x       0 0x       0
[ 2476.808256] mt7915e 0000:01:00.0: [0x00017900] 0x       0 0x       0 0x       0 0x       0
[ 2476.816504] mt7915e 0000:01:00.0: [0x00017910] 0x       0 0x       0 0x       0 0x       0
[ 2476.824752] mt7915e 0000:01:00.0: [0x00017920] 0x       0 0x       0 0x       0 0x       0
[ 2476.833000] mt7915e 0000:01:00.0: [0x00017930] 0x       0 0x       0 0x       0 0x       0
[ 2476.841248] mt7915e 0000:01:00.0: [0x00017940] 0x       0 0x       0 0x       0 0x       0
[ 2476.849496] mt7915e 0000:01:00.0: [0x00017950] 0x       0 0x       0 0x       0 0x       0
[ 2476.857743] mt7915e 0000:01:00.0: [0x00017960] 0x       0 0x       0 0x       0 0x       0
[ 2476.865990] mt7915e 0000:01:00.0: [0x00017970] 0x       0 0x       0 0x       0 0x       0
[ 2476.874238] mt7915e 0000:01:00.0: [0x00017980] 0x       0 0x       0 0x       0 0x       0
[ 2476.882499] mt7915e 0000:01:00.0: [0x00017990] 0x       0 0x       0 0x       0 0x       0
[ 2476.890758] mt7915e 0000:01:00.0: [0x000179a0] 0x       0 0x       0 0x       0 0x       0
[ 2476.899004] mt7915e 0000:01:00.0: [0x000179b0] 0x       0 0x       0 0x       0 0x       1
[ 2476.907252] mt7915e 0000:01:00.0: [0x000179c0] 0x       0 0x       0 0xffffffff 0xffff0000
[ 2476.915499] mt7915e 0000:01:00.0: [0x000179d0] 0x 3030303 0x 3030000 0x 3030303 0x 3030000
[ 2476.923746] mt7915e 0000:01:00.0: [0x000179e0] 0x 9090909 0x 9090000 0x 7070707 0x 7070000
[ 2476.931993] mt7915e 0000:01:00.0: [0x000179f0] 0x       0 0x       0 0x       0 0x       0
[ 2476.940240] mt7915e 0000:01:00.0: [0x00017a00] 0x       0 0x       1 0x       0 0x       0
[ 2476.948487] mt7915e 0000:01:00.0: [0x00017a10] 0x       0 0x       0 0x       0 0x       1
[ 2476.956735] mt7915e 0000:01:00.0: [0x00017a20] 0x      21a 0x      7ff 0x 7ff0001 0x 7fc07fc
[ 2476.964982] mt7915e 0000:01:00.0: [0x00017a30] 0x  120005 0x       0 0x       1 0x     234
[ 2476.973230] mt7915e 0000:01:00.0: [0x00017a40] 0x 7ff0000 0x 7ff07ff 0x 7fe07f8 0x    c0015
```

3.6 Pre-calibration

2.2.31 Implicit Beamforming (iBF)

This section covers the whole set of iBF commands, such as iBF setting initialization, iBF phase compensation, BF profile update, BF TX setting, iBF calibration, and saving calibrated iBF results to eeprom. Note that the argument of the iBF command must contain txbf_act and txbf_param at the same time. txbf_act sets the action TXBF and txbf_param sets the required parameters for the TXBF action. For more information, please refer to iBF example.

2.2.31.1 Init Setting for iBF

Init the setting of DUT for iBF including

1. Set the MAC address configuration
2. Set the PHY mode, MCS rate, bandwidth, guard interval and IPG

Input Argument	Description	Value
txbf_act	Init the setting for iBF DUT	init
	Init the setting for iBF golden unit	**Not supported** **Please use proprietary driver**
txbf_param	Do nothing/Enable	0/1

- Command

```
mt76-test phy0 set state=idle
# init for DUT
mt76-test phy0 set txbf_act=init txbf_param=1
```

- Example

```
root@OpenWrt:/# mt76-test phy2 set txbf_act=init txbf_param=1
[317018.374833] ibf cal process: act = 0, val = 1, 0, 0, 0, 0
```

3.7 Implicit Beamforming

2.2.31.2 Phase Compensation

Enable the iBF phase compensation or clear the compensated TX/RX phases.

Input Argument	Description		Value
txbf_act	Do phase compensation		phase_comp
txbf_param	Set BW	20 MHz (non-HT)	0
		20 MHz	1
		40 MHz	2
		80 MHz	3
		80 + 80 MHz	4
		160 MHz	5
		5 MHz OFDM	6
		10 MHz OFNM	7
	Set band index		0/1
	Is JP band		0/1
	Enable reading from eeprom		0/1
	Compensate TX/RX phases		0
	Clear compensated TX/RX phases		1

- Command

```
mt76-test phy0 set txbf_act=phase_comp txbf_param=0,0,0,0,1 aid=1
```

Note:

- If aid=1 is not entered before this command, it is required for adding virtual wtbl with wcid = aid. For more information, please refer to aid.

- Example

```
root@OpenWrt:/# mt76-test phy2 set txbf_act=phase_comp txbf_param=0,0,0,0,1 aid=
1
[317110.517717] ibf cal process: act = 2, val = 0, 0, 0, 0, 1
[317111.546662] ibf cal process: phase comp info
[317111.551012] 0f 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00
[317111.556661] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[317111.562306] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

3.7 Implicit Beamforming

2.2.31.3 BF Profile Configuration Read and Write

2.2.31.3.1 eBF/iBF Profile Configuration Update

Configure the profile management unit (PFMU) tag and BF station record of eBF/iBF for updating the channel profile.

Input Argument	Description	Value
txbf_act	Configure the eBF/iBF profile	ebf_prof_update ibf_prof_update
txbf_param	PFMU index	[0, 63]
	Not used	-
	Nc (The number of columns in the compressed beamforming feedback matrix)	Default = 0

- Command

```
mt76-test phy0 set txbf_act=ebf_prof_update txbf_param=1,3,0
mt76-test phy0 set txbf_act=ibf_prof_update txbf_param=2,3,0
```

- Example

```
root@OpenWrt:/# mt76-test phy2 set txbf_act=ebf_prof_update txbf_param=1,3,0
[319904.570064] ibf cal process: act = 5, val = 1, 3, 0, 0, 0
root@OpenWrt:/# mt76-test phy2 set txbf_act=ibf_prof_update txbf_param=2,3,0
[319955.549101] ibf cal process: act = 4, val = 2, 3, 0, 0, 0
```

3.7 Implicit Beamforming

2.2.31.3.2 Channel Profile Configuration Update

Directly write the channel profile to the DUT.

Input Argument	Description	Value
txbf_act	Configure channel profile	prof_update_all
txbf_param	Start or stop to update	
	PFMU index	[0, 63]
	Starting sequence Ending sequence	f0 ff
	Update data	
	Subcarrier index	[00, 3F], hexadecimal
	Angle of H11	[0000, 0400], hexadecimal
	Angle of H21	[0000, 0400], hexadecimal
	Angle of H31	[0000, 0400], hexadecimal
	Angle of H41	[0000, 0400], hexadecimal
	Angle of H51 **(Only for mt7992 BE7200 iFEM 5G 5T5R)**	[0000, 0400], hexadecimal
	… (repeat the format above)	…

The angle of Hx1 originally ranges from $-\pi$ to π. We map it to $[-0.5, 0.5]$ and transform it through the formula listed below to hexadecimals.

$$y = \begin{cases} round\big((x+2)\times 512\big), & if\ x < 0 \\ round(x \times 512), & otherwise \end{cases}, \quad where\ x \in [-0.5, 0.5]$$

- Command

```
// Start profile update
mt76-test phy0 set txbf_act=prof_update_all txbf_param=01,f0
mt76-test phy0 set txbf_act=prof_update_all txbf_param=
           00,0000,0000,0000,0000,01,0000,0000,0000,0000,02,0000,0000,0000,0000,
           03,0000,0000,0000,0000,04,003c,039f,0055,0375,05,001c,0380,0035,0356,
           06,03fd,0360,0015,0336,07,03de,0340,03f6,0317
mt76-test phy0 set txbf_act=prof_update_all txbf_param=
           08,03be,0321,03d7,00f7,09,039e,0301,03b7,00d8,0a,037f,00e2,0398,00b8,
           0b,0000,0000,0000,0000,0c,0340,00a3,0359,0079,0d,0321,0084,033a,005a
           0e,0302,0064,031a,003a,0f,03de,0340,03f6,0317
                                    ⋮
// Stop profile update
mt76-test phy0 set txbf_act=prof_update_all txbf_param=01,ff
```

Note that you can input any length of data segment of a subcarrier (including subcarrier index, $\mathrm{Ang}_{\mathrm{H11}}$, $\mathrm{Ang}_{\mathrm{H21}}$, $\mathrm{Ang}_{\mathrm{H31}}$ and $\mathrm{Ang}_{\mathrm{H41}}$) and even in any order since it buffs the data until the data of subcarrier index = 3F (63 in decimal) is sent.

- Example

```
root@OpenWrt:/# mt76-test phy2 set txbf_act=prof_update_all txbf_param=01,f0
root@OpenWrt:/# mt76-test phy2 set txbf_act=prof_update_all txbf_param=00,0000,0
000,0000,0000,01,0000,0000,0000,0000,02,0000,0000,0000,0000,03,0000,0000,0000,00
00,04,003c,039f,0055,0375,05,001c,0380,0035,0356,06,03fd,0360,0015,0336,07,03de,
0340,03f6,0317
[ 1203.109344] ibf cal process: act = 9, val = 1, 0, 0, 0, 0, 0
[ 1203.115051] ibf cal process: act = 9, val = 1, 1, 0, 0, 0, 0
[ 1203.120709] ibf cal process: act = 9, val = 1, 2, 0, 0, 0, 0
[ 1203.126357] ibf cal process: act = 9, val = 1, 3, 0, 0, 0, 0
[ 1203.132004] ibf cal process: act = 9, val = 1, 4, 60, 927, 85, 885
[ 1203.138172] ibf cal process: act = 9, val = 1, 5, 28, 896, 53, 854
[ 1203.144338] ibf cal process: act = 9, val = 1, 6, 1021, 864, 21, 822
[ 1203.150685] ibf cal process: act = 9, val = 1, 7, 990, 832, 1014, 791
root@OpenWrt:/# mt76-test phy2 set txbf_act=prof_update_all txbf_param=01,FF
```

3.7 Implicit Beamforming

2.2.31.3.3 PFMU Tag Write

Configure the profile management unit (PFMU) tag of eBF/iBF for updating the channel profile. This command would use the PFMU tag data stored in driver to update it. If the PFMU tag data is not allocated in driver, it would not take any action.

Input Argument	Description	Value
txbf_act	Configure the eBF/iBF PFMU tag profile	pfmu_tag_write
txbf_param	Select which PFMU index to update	[0, 63]

- Command

```
mt76-test phy0 set txbf_act=pfmu_tag_write txbf_param=1
```

- Example

```
root@OpenWrt:/# mt76-test phy0 set txbf_act=pfmu_tag_write txbf_param=1
[  640.411588] mt7915e 0000:01:00.0: ibf cal process: act = 17, val = 1, 0, 0, 0, 0, 0
```

3.7 Implicit Beamforming

2.2.31.3.4 PFMU Tag Read

Read the current settings of the profile management unit (PFMU) tag of eBF/iBF.

Input Argument	Description	Value
txbf_act	Read the eBF/iBF PFMU tag profile	pfmu_tag_read
txbf_param	Select which PFMU index to read	[0, 63]
	Is beamformer or beamformee	Bfer: 1 Bfee: 0

- Command

  `mt76-test phy0 set txbf_act=pfmu_tag_read txbf_param=1,1`

- Example

```
root@OpenWrt:/# mt76-test phy0 set txbf_act=pfmu_tag_read txbf_param=2,1
[ 1109.886154] mt7915e 0000:01:00.0: ibf cal process: act = 16, val = 2, 1, 0, 0, 0, 0
[ 1109.893935] mt7915e 0000:01:00.0: ================== TXBf Profile Tag1 Info ==================
[ 1109.902619] mt7915e 0000:01:00.0: DW0 = 0x60030002, DW1 = 0x01400100, DW2 = 0x01c00180
[ 1109.910524] mt7915e 0000:01:00.0: DW4 = 0x00000000, DW5 = 0x000000cc, DW6 = 0x00000000
[ 1109.910524]
[ 1109.919899] mt7915e 0000:01:00.0: PFMU ID = 2              Invalid status = 0
[ 1109.927022] mt7915e 0000:01:00.0: iBf/eBf = 0
[ 1109.927022]
[ 1109.932844] mt7915e 0000:01:00.0: DBW    = 0
[ 1109.937020] mt7915e 0000:01:00.0: SU/MU  = 0
[ 1109.941195] mt7915e 0000:01:00.0: RMSD   = 3
[ 1109.945373] mt7915e 0000:01:00.0: nrow = 3, ncol = 0, ng = 0, LM = 0, CodeBook = 0 MobCalEn = 0
[ 1109.954055] mt7915e 0000:01:00.0: RU start = 0, RU end = 0
[ 1109.959531] mt7915e 0000:01:00.0: Mem Col1 = 0, Mem Row1 = 4, Mem Col2 = 0, Mem Row2 = 5
[ 1109.967607] mt7915e 0000:01:00.0: Mem Col3 = 0, Mem Row3 = 6, Mem Col4 = 0, Mem Row4 = 7
[ 1109.967607]
[ 1109.977158] mt7915e 0000:01:00.0: STS0_SNR = 0xcc, STS1_SNR = 0x00, STS2_SNR = 0x00, STS3_SNR = 0x00
[ 1109.986276] mt7915e 0000:01:00.0: STS4_SNR = 0x00, STS5_SNR = 0x00, STS6_SNR = 0x00, STS7_SNR = 0x00
[ 1109.995392] mt7915e 0000:01:00.0: ============================================================
[ 1110.004075] mt7915e 0000:01:00.0: ================== TXBf Profile Tag2 Info ==================
[ 1110.012758] mt7915e 0000:01:00.0: DW0 = 0x18000000, DW1 = 0x00000000, DW2 = 0x00600000
[ 1110.020661] mt7915e 0000:01:00.0: DW3 = 0x00000000, DW4 = 0x00000000, DW5 = 0xe0080000
[ 1110.020661]
[ 1110.030038] mt7915e 0000:01:00.0: Smart antenna ID = 0x0,  SE index = 24
[ 1110.036727] mt7915e 0000:01:00.0: RMSD threshold = 0
[ 1110.041684] mt7915e 0000:01:00.0: Timeout = 0x0
[ 1110.046207] mt7915e 0000:01:00.0: Desired BW = 0, Desired Ncol = 0, Desired Nrow = 3
[ 1110.053935] mt7915e 0000:01:00.0: Desired RU Allocation = 0
[ 1110.059498] mt7915e 0000:01:00.0: Mobility DeltaT = 0, Mobility LQ = 0
[ 1110.066014] mt7915e 0000:01:00.0: ============================================================
```

Note:

1. Tag 1 info:
   I. Invalid status: 0 for already updated, 1 for not updated.
   II. iBf/eBf: 0 for iBF mode, 1 for eBF mode.
   III. DBW: the meaning of the number is same as the BW parameter in phase compensation.
   IV. SU/MU: 0 for SU, 1 for MU
2. Tag 2 info:
   I. Timeout: 0xff for no timeout, 0 for already timeout.
   II. Desired Ncol: golden's antenna number – 1
   III. Desired Nrow: DUT's antenna number – 1

3.7 Implicit Beamforming

2.2.31.3.5 Set PFMU Tag Invalid Bit

Set the invalid bit of the PFMU tag. **Note that this command did not update the PFMU tag to firmware. If you want to validate or invalidate the PFMU tag, please use the PFMU tag write command above.**

Input Argument	Description	Value
txbf_act	Set the invalid bit of the PFMU tag	set_invalid_prof
txbf_param	The value of the invalid bit	Valid: 0 Invalid: 1

- Command

  `mt76-test phy0 set txbf_act=set_invalid_prof txbf_param=1`

- Example

```
root@OpenWrt:/# mt76-test phy0 set txbf_act=set_invalid_prof txbf_param=1
[ 1390.217962] mt7915e 0000:01:00.0: ibf cal process: act = 18, val = 1, 0, 0, 0, 0, 0
```

2.2.31.3.6 BF Station Record Read

Read the current settings of the station record of eBF/iBF.

Input Argument	Description	Value
txbf_act	Read eBF/iBF station record	sta_rec_read
txbf_param	Select which WLAN index to read *The WLAN index here is the wcid index set in Bfer's WTBL for the station you wish to send BF packet*	[0, 16]

- Command

  `mt76-test phy0 set txbf_act=sta_rec_read txbf_param=1`

- Example

```
root@OpenWrt:/# mt76-test phy0 set txbf_act=sta_rec_read txbf_param=1
[ 1496.033028] mt7915e 0000:01:00.0: ibf cal process: act = 19, val = 1, 0, 0, 0, 0, 0
[ 1496.040816] mt7915e 0000:01:00.0: ==================== BF Station Record ====================
[ 1496.049498] mt7915e 0000:01:00.0: pfmu          = 2
[ 1496.054455] mt7915e 0000:01:00.0: su_mu         = 0
[ 1496.059412] mt7915e 0000:01:00.0: bf_cap        = 0
[ 1496.064365] mt7915e 0000:01:00.0: sounding_phy  = 1
[ 1496.069320] mt7915e 0000:01:00.0: ndpa_rate     = 0
[ 1496.074276] mt7915e 0000:01:00.0: ndp_rate      = 24
[ 1496.079317] mt7915e 0000:01:00.0: rept_poll_rate = 0
[ 1496.084274] mt7915e 0000:01:00.0: tx_mode       = 2
[ 1496.089229] mt7915e 0000:01:00.0: ncol          = 0
[ 1496.094183] mt7915e 0000:01:00.0: nrow          = 3
[ 1496.099139] mt7915e 0000:01:00.0: bw            = 0
[ 1496.104096] mt7915e 0000:01:00.0: mem_total     = 0
[ 1496.109051] mt7915e 0000:01:00.0: mem_20m       = 0
[ 1496.114007] mt7915e 0000:01:00.0: mem_row0      = 4
[ 1496.118962] mt7915e 0000:01:00.0: mem_col0      = 0
[ 1496.123918] mt7915e 0000:01:00.0: mem_row1      = 5
[ 1496.128873] mt7915e 0000:01:00.0: mem_col1      = 0
[ 1496.133828] mt7915e 0000:01:00.0: mem_row2      = 6
[ 1496.138784] mt7915e 0000:01:00.0: mem_col2      = 0
[ 1496.143740] mt7915e 0000:01:00.0: mem_row3      = 7
[ 1496.148695] mt7915e 0000:01:00.0: mem_col3      = 0
[ 1496.153651] mt7915e 0000:01:00.0: smart_ant     = 0x0
[ 1496.158780] mt7915e 0000:01:00.0: se_idx        = 24
[ 1496.163823] mt7915e 0000:01:00.0: auto_sounding = 0
[ 1496.168779] mt7915e 0000:01:00.0: ibf_timeout   = 0xff
[ 1496.173993] mt7915e 0000:01:00.0: ibf_dbw       = 0
[ 1496.178949] mt7915e 0000:01:00.0: ibf_ncol      = 0
[ 1496.183905] mt7915e 0000:01:00.0: ibf_nrow      = 0
[ 1496.188860] mt7915e 0000:01:00.0: nrow_gt_bw80  = 0
[ 1496.193816] mt7915e 0000:01:00.0: ncol_gt_bw80  = 0
[ 1496.198771] mt7915e 0000:01:00.0: ru_start_idx  = 0
[ 1496.203727] mt7915e 0000:01:00.0: trigger_su    = 0
[ 1496.208682] mt7915e 0000:01:00.0: trigger_mu    = 0
[ 1496.213638] mt7915e 0000:01:00.0: ng16_su       = 0
[ 1496.218592] mt7915e 0000:01:00.0: ng16_mu       = 0
[ 1496.223548] mt7915e 0000:01:00.0: codebook42_su = 0
[ 1496.228503] mt7915e 0000:01:00.0: codebook75_mu = 0
[ 1496.233460] mt7915e 0000:01:00.0: he_ltf        = 0
[ 1496.238415] mt7915e 0000:01:00.0: ============================================================
```

2.2.31.4 BF TX Setting

2.2.31.4.1 Apply TXBF to WTBL

Update the WTBL (Wireless Lan TabLe, storing capacity or information for the connected peer) for applying TXBF.

Input Argument	Description	Value
txbf_act	Apply TXBF	apply_tx
txbf_param	Set WLAN index *The WLAN index here is the wcid index set in Bfer's WTBL for the station you wish to send BF packet*	Decimal
	Enable eBF TX	0/1
	Enable iBF TX	0/1
	Enable Mu BFTX	0 Enable is not supported yet
	Phase Calibration	1

- Command

```
mt76-test phy0 set txbf_act=apply_tx txbf_param=1,0,1,0,1
```

- Example

```
root@OpenWrt:/# mt76-test phy2 set txbf_act=apply_tx txbf_param=1,1,0,0,1
[320257.106366] ibf cal process: act = 6, val = 1, 1, 0, 0, 1
```

3.7 Implicit Beamforming

2.2.31.4.2 BF TX Preparation

Profile tag read/write for invalid iBF profile, set TXBF rate, TX length, TX count, and other TX settings.

Input Argument	Description	Value
txbf_act	Prepare to TX	tx_prep
txbf_param	Set BF on	0/1
	Set AID	1
	Set WLAN index *The WLAN index here is the wcid index set in Bfer's WTBL for the station you wish to send BF packet*	Decimal
	Set update	0/1

- Command

```
mt76-test phy0 set txbf_act=tx_prep txbf_param=0,1,1,0 aid=1 tx_count=10000000
tx_length=1024
mt76-test phy0 set state=tx_frames
```

Note:

1. tx_prep does not start TX in mt76, you must start it by setting state = tx_frames.
2. The TXBF parameter of setting AID and setting update (which triggers tx_apply with fixed parameter) is used for HQADLL mode. To simplify the format of driver code, we reserve those parameters. For command mode, just input the value in the table above.
3. tx_count is set to a large number for simulating continuous TX.

- Example

```
root@OpenWrt:/# mt76-test phy2 set txbf_act=tx_prep txbf_param=0,1,1,0 aid=1 tx_
count=10000000 tx_length=1024
[320319.342425] ibf cal process: act = 3, val = 0, 1, 1, 0, 0
```

3.7 Implicit Beamforming

2.2.31.4.3 BF TXCMD Configuration

Configure TXCMD BF bit manually.

Note that this command is available only in Wi-Fi 7 chipsets.

Input Argument	Description		Value
txbf_act	Read/Write TXCMD BF bit configuration		txcmd
txbf_param	Action	Read	0 (Not supported yet)
		Write	1
	Enable TXCMD BF bit manual control		0/1
	TXCMD BF bit		0/1

- Command

```
// Force TXCMD BF bit to 1 to TX BF
mt76-test phy0 set txbf_act=txcmd txbf_param=1,1,1
// Force TXCMD BF bit to 0 to disable TX BF
mt76-test phy0 set txbf_act=txcmd txbf_param=1,1,0
// Return to normal mode for TXCMD BF bit control
mt76-test phy0 set txbf_act=txcmd txbf_param=1,0,0
```

- Example

```
root@OpenWrt:/# mt76-test phy0 set txbf_act=txcmd txbf_param=1,1,1
[ 1127.294053] mt7996e 0000:01:00.0: ibf cal process: act = 20, val = 1, 1, 1, 0, 0, 0, 0, 0
```

3.7 Implicit Beamforming

2.2.31.5 TXBF Trigger/Stop Sounding

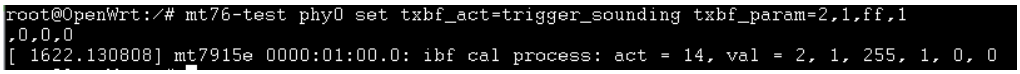
Trigger the firmware to start or stop sending sounding packets (including null data packets (NDP) and null data packet announcement (NDPA)). This command is used for iBF calibration and eBF certification with the golden device only.

Input Argument	Description	Value	
txbf_act	Trigger sounding Stop sounding	trigger_sounding stop_sounding	
txbf_param	Set sounding mode	SU sounding	0
		MU sounding	1
		SU periodic sounding	2
		MU periodic sounding	3
	Set the MU number (i.e. station number)	Decimal	
	Set the sounding interval for periodic sounding	Decimal (Unit: 4 ms)	
	Set WLAN index *The WLAN index here is the wcid index set in Bfer's WTBL for the station you wish to send sounding packet* *(4 u8 data forming a u32 data)*	Decimal	

- Command

```
// Starts sounding
mt76-test phy0 set txbf_act=trigger_sounding txbf_param=2,1,ff,1,0,0,0
// Stops sounding
mt76-test phy0 set txbf_act=stop_sounding txbf_param=0
```

- Example



3.7 Implicit Beamforming

2.2.31.6 iBF Calibration

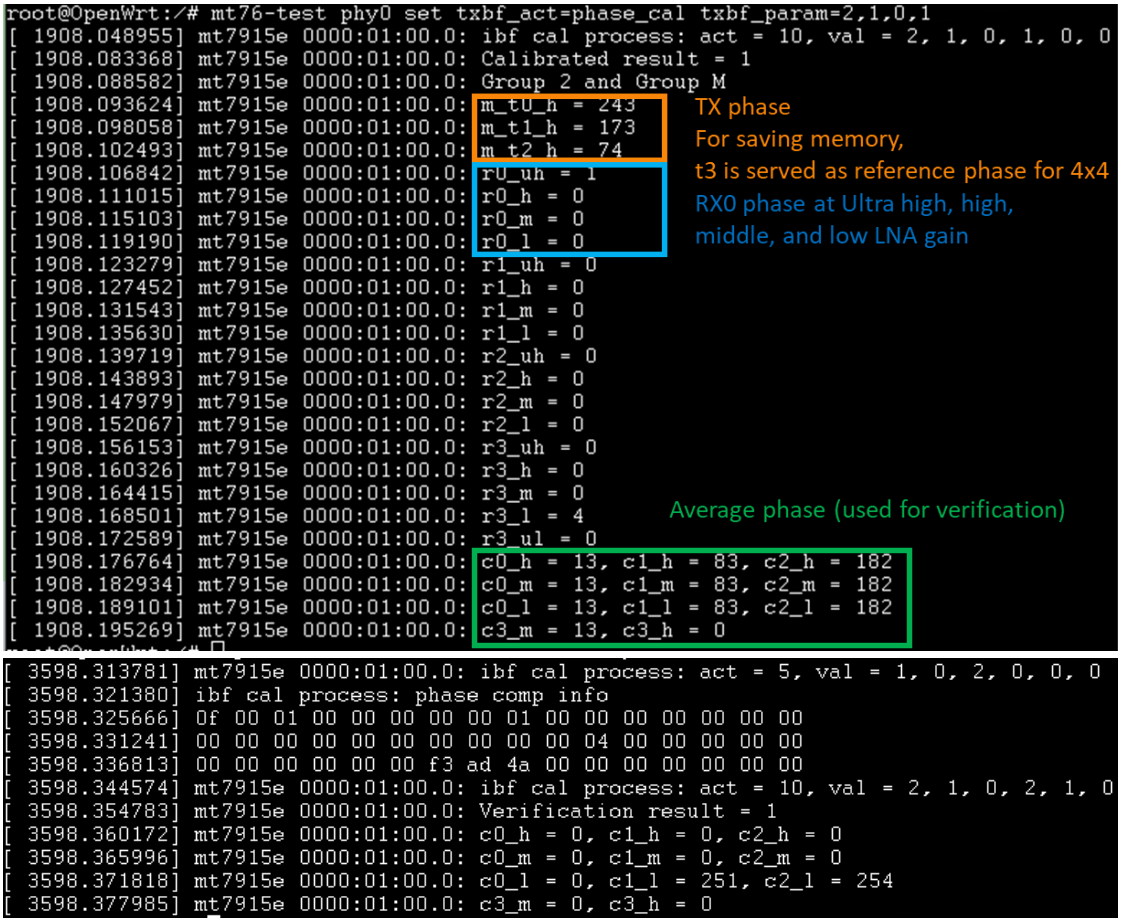
Do iBF calibration.

Input Argument	Description			Value
txbf_act	Start iBF calibration			phase_cal
txbf_param	Set group index			[0, 8], decimal
	Set the L/M/H channel in group (Use the lowest/middle/highest channel in each channel group to do phase calibration)	L		0
		M		1
		H		2
	Set SX2 (Band index)			0, for 2G band 0
				1, for 5G band 1
	Set calibration type	Do nothing		0
		Calibration		1
		Verification		2
		Calibration with instrument		3
	Set the low noise amplifier (LNA) gain level	2G	Low	0
			Middle	1
			High	2
			Ultra-high	3
		5G	Low	0
			Middle	1
			Middle high	2
			High	3
			Ultra-high	4
	iBF Version			0, for iBF 1.0
				3, for iBF 2.0

- Command

```
mt76-test phy0 set txbf_act=phase_cal txbf_param=1,1,1,3,1,0
```

- Example




```
[ 3598.313781] mt7915e 0000:01:00.0: ibf cal process: act = 5, val = 1, 0, 2, 0, 0, 0
[ 3598.321380] ibf cal process: phase comp info
[ 3598.325666] 0f 00 01 00 00 00 00 00 01 00 00 00 00 00 00 00
[ 3598.331241] 00 00 00 00 00 00 00 00 00 00 04 00 00 00 00 00
[ 3598.336813] 00 00 00 00 00 00 f3 ad 4a 00 00 00 00 00 00 00
[ 3598.344574] mt7915e 0000:01:00.0: ibf cal process: act = 10, val = 2, 1, 0, 2, 1, 0
[ 3598.354783] mt7915e 0000:01:00.0: Verification result = 1
[ 3598.360172] mt7915e 0000:01:00.0: c0_h = 0, c1_h = 0, c2_h = 0
[ 3598.365996] mt7915e 0000:01:00.0: c0_m = 0, c1_m = 0, c2_m = 0
[ 3598.371818] mt7915e 0000:01:00.0: c0_l = 0, c1_l = 251, c2_l = 254
[ 3598.377985] mt7915e 0000:01:00.0: c3_m = 0, c3_h = 0
```

Note:

1. If calibration is done successfully, it would return “Calibrated result = 1”.
2. This command will return the calibrated phase
   I. TX phase: the phase of the largest TX path (e.g. T3 for 4×4) would be the reference phase. That is, $m\_tn\_h = m\_tn\_h - m\_tN\_h, for \begin{cases} N = largest\ TX\ path \\ n = 0\ to\ N-1 \end{cases}$.
   II. RX phase: Record the N^{th} RX phase of ultra-high, high, middle, and low LNA gain. No ultra-low gain since iBF normally would not be applied in this case.
3. For verification:
   I. If verified success, it would return “Verification result = 1”.
   II. Please do phase compensation before phase verification.
   III. The verification here measures the degree of compensated phase. **The average phases must fall within the range of $[-15^{\circ}, 15^{\circ}]$ to pass.**

3.7 Implicit Beamforming

2.2.31.7 iBF Save to EEPROM

Write the calibrated phase into eeprom **in the driver**.

To sync the iBF cal result to atenl tmp file, please refer to the atenl eeprom ibf sync command.

Input Argument	Description	Value
txbf_act	Write the calibrated phase into EEPROM	e2p_update

Input Argument	Description		Value
txbf_param	Set the group index		[0, 8], decimal **[0, 12], decimal** **(for mt7992 BE7200 2i5i 5T5R)**
	Set the update band type	Update all	0
		Update BW 160	Not supported
		Update 2G only	
		Update 5G only	
	Set the update type	Update one group	Not supported yet
		Update all of the groups	1
		Erase all of the groups	Not supported
		Read calibrated phases from EEPROM	

- Command

```
mt76-test phy0 set txbf_act=e2p_update txbf_param=0,0,1
```

- Example

```
root@OpenWrt:/# mt76-test phy2 set txbf_act=e2p_update txbf_param=0,0,1
[320977.074582] ibf cal process: act = 10, val = 0, 0, 1, 0, 0
```

3.7 Implicit Beamforming

2.2.32 Explicit Beamforming (eBF)

This section covers the eBF certification commands. Some commands used in the eBF certification is already listed in iBF section. Note that the argument of the eBF command must contain txbf_act and txbf_param at the same time. txbf_act sets the action TXBF and txbf_param sets the required parameters for the TXBF action. For more information, please refer to eBF example.

2.2.32.1 Init Setting for eBF

Initialize the settings of DUT for eBF including

1. Set the MAC address configuration
2. Create second interface (broadcast and multicast (BMC) entry) in WTBL for using TXCMD to transmit sounding packet (**BF data should still use TXD for mt7915/mt7916/mt7981/mt7986**)
3. Enable ETXBF capability for the DUT
4. Set the tx_antenna_mask and the spe_idx

Input Argument	Description	Value
txbf_act	Init the setting for the eBF DUT	ebf_init
	Init the setting for the eBF golden device	**Not supported** **Please use Jedi/Logan as golden**
txbf_param	Do nothing/Enable	0/1

- Command

```
mt76-test phy0 set state=idle
# init for DUT
mt76-test phy0 set txbf_act=ebf_init txbf_param=1
```

- Example

```
root@OpenWrt:/# mt76-test phy0 set txbf_act=ebf_init txbf_param=1
[  308.826698] mt7915e 0000:01:00.0: ibf cal process: act = 3, val = 1, 0, 0, 0, 0, 0
```

```
root@OpenWrt:/# mt76-test phy0 set txbf_act=ebf_golden_init txbf_param=1
[  343.835718] mt7915e 0000:01:00.0: ibf cal process: act = 2, val = 1, 0, 0, 0, 0, 0
[  343.845190] mt7915e 0000:01:00.0: Set BBP RX CR = 5
```

3.8 Explicit Beamforming

2.2.33 Zero Wait DFS (ZWDFS)

These commands are used for testing the hardware functionality of background chain (i.e. check its Received Signal Strength Indication (RSSI) and Idle Power Indicator (IPI) counter (see 2.2.33.3 for IPI introduction)). Therefore, the channel selection could be non-DFS channels, and **the country code could remain at "VV"**.

2.2.33.1 Set channel and bandwidth of background chain

Enable and set the channel and bandwidth of the background chain (the RX chain used for ZWDFS) in test mode.

Input Argument	Description	Value
offchan_ch	Set the control channel of the background chain	[36, 196] Valid 5G channel
offchan_center_ch	Set the center channel of the background chain (*optional; if it is not set, the driver will calculate it based on the control channel and bw*)	[36, 196] Valid 5G channel
offchan_bw	Set the bandwidth of the background chain	NOHT
		20
		40
		80
		160

- Command

```
mt76-test phy0 set state=idle
mt76-test phy0 set offchan_ch=100 offchan_bw=80
```

Note:

- The state should be set to idle before enabling background chain.
- The background chain is enabled only when offchan ch and offchan bw are set.

- Example

```
root@OpenWrt:/# mt76-test phy1 dump | grep offchan
offchan_ch=100
offchan_center_ch=106
offchan_bw=80
```

3.9 Zero Wait DFS

2.2.33.2 Dump Channel and Bandwidth of Background Chain

Dump the current center channel and bandwidth if the background chain is enabled.

- Command

  `mt76-test phy0 dump`

- Example

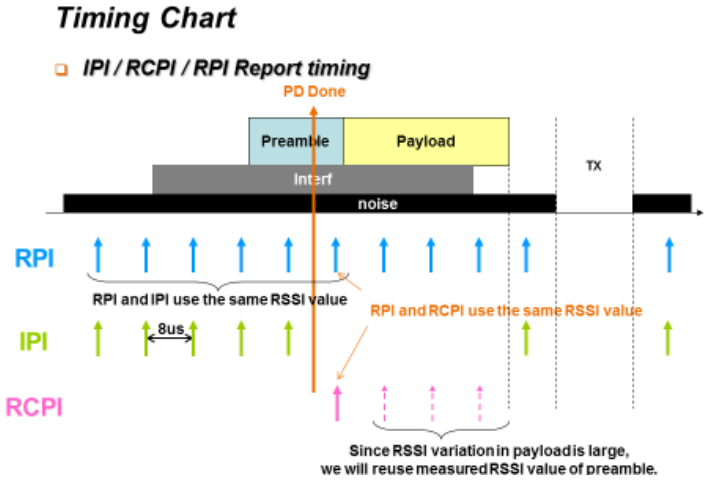
```
root@OpenWrt:/# mt76-test phy1 dump
state=idle
tx_count=1
tx_length=1024
tx_rate_mode=ofdm
tx_rate_nss=1
tx_rate_idx=0
tx_rate_sgi=0
tx_rate_ldpc=0
tx_rate_stbc=0
aid=0
ru_alloc=0
ru_idx=0
offchan_ch=100
offchan_center_ch=0
offchan_bw=80
```

3.9 Zero Wait DFS

2.2.33.3 Read Idle Power Indicator (IPI) Histogram of Background Chain

Read the IPI histogram of background chain. The IPI histogram shows the IPI count for each IPI index, where each index is defined as a specific power range as shown in the table below. Also, the IPI count of each IPI index could be read from the control register listed in the table below (via this command). Note that mt7986 has no dedicated RX, so there is no CR for IPI in mt7986.

The concept of IPI is shown in the following figure. The Idle Power Indicator, which is defined in 802.11K, is a counter used for idle power measurement. The IPI counter of the corresponding IPI index updates every 8 until a packet is detected (Packet Detection (PD) Done), based on the current RSSI value. Therefore, this indicator could be used to test out the hardware ability of the dedicated RX (background chain for ZWDFS).



IPI Index	Power Range [dBm]	CR Address of mt7915	CR Address of mt7916	CR Address of mt7996/mt7992
0	$(-\infty, -92)$	0x830AF0A8	0x83121000	0x83041000
1	$(-92, -89)$	0x830AF0AC	0x83121004	0x83041004
2	$(-89, -86)$	0x830AF0B0	0x83121008	0x83041008
3	$(-86, -83)$	0x830AF0B4	0x8312100C	0x8304100C
4	$(-83, -80)$	0x830AF0B8	0x83121010	0x83041010
5	$(-80, -75]$	0x830AF0BC	0x83121014	0x83041014
6	$(-75, -70]$	0x830AF0C0	0x83121018	0x83041018
7	$(-70, -65]$	0x830AF0C4	0x8312101C	0x8304101C
8	$(-65, -60]$	0x830AF0C8	0x83121020	0x83041020
9	$(-60, -55]$	0x830AF0CC	0x83121024	0x83041024
10	$(-55, \infty)$	0x830AF0D0	0x83121028	0x83041028

The format of the command is listed in the following table. This command also calculates the channel load, self-idle ratio, and IPI idle ratio according to the IPI threshold you set.

$$ipi\_hist\_count_i = ipi\ count\ for\ ipi\_index = i$$

$$ipi\_hist\_count\_th = \sum_{i=th}^{10} ipi\_hist\_count_i\,, where\ 0 \le th \le 10$$

$$ipi\_free\_count = \sum_{i=0}^{10} ipi\_hist\_count_i$$

$$ipi\_idle\_ratio\ = \frac{ipi\_free\_count - ipi\_hist\_count\_th}{ipi\_free\_count} \times 100\%$$

$$self\_idle\_ratio\ = \frac{ipi\_period - tx\_assert\_time}{ipi\_period} \times 100\%$$

$$channel\_load\ = \begin{cases} \frac{self\_idle\_ratio - ipi\_idle\_ratio}{self\_idle\_ratio} \times 100\%\,, & self\_idle\_ratio \ge ipi\_idle\_ratio \\ 0, & otherwise \end{cases}$$

Note that ipi_antenna_idx is not available in the Wi-Fi 7 chipset.

Input Argument	Description		Value
ipi_threshold	Set the IPI index threshold for calculating the channel load and idle ratio *When IPI threshold is set to K, then the total IPI count of IPI index ≥ K is used to calculate channel load and idle ratio*		[0, 10], decimal The range of the IPI index
ipi_period	Set the period for accumulating the IPI counter		[0, 10000], msec
ipi_antenna_idx	Specify to read the IPI histogram of which antenna	Antenna 0	0
		Antenna 1	1
		Antenna 2	2
		Antenna 3	3
		Use all antennas	4 (Default value)

- Command

```
mt76-test phy0 set ipi_threshold=8 ipi_period=100 ipi_antenna_idx=2
```

Note:

- The state should be set to idle and background chain should be enabled before reading IPI histogram.
- This command prints out all the IPI histogram of all the antenna if ipi_antenna_idx is not specified.
- This command automatically resets all the IPI counters to zero, and then it returns the accumulated IPI counter during the set IPI period.

- Example

```
root@OpenWrt:/# mt76-test phy1 set ipi_threshold=0 ipi_period=100 ipi_antenna_id
x=2
root@OpenWrt:/# [  770.249322] mt7915e 0000:01:00.0: Antenna index: 6
[  770.254118] mt7915e 0000:01:00.0: IPI 0 (power range: (-inf, -92] dBm): ipi count = 1456480
[  770.262459] mt7915e 0000:01:00.0: IPI 1 (power range: (-92, -89] dBm): ipi count = 5803663
[  770.270712] mt7915e 0000:01:00.0: IPI 2 (power range: (-89, -86] dBm): ipi count = 3166532
[  770.278964] mt7915e 0000:01:00.0: IPI 3 (power range: (-86, -83] dBm): ipi count = 3398024
[  770.287217] mt7915e 0000:01:00.0: IPI 4 (power range: (-83, -80] dBm): ipi count = 2707706
[  770.295468] mt7915e 0000:01:00.0: IPI 5 (power range: (-80, -75] dBm): ipi count = 1325184
[  770.303718] mt7915e 0000:01:00.0: IPI 6 (power range: (-75, -70] dBm): ipi count = 1172129
[  770.311970] mt7915e 0000:01:00.0: IPI 7 (power range: (-70, -65] dBm): ipi count = 696205
[  770.320134] mt7915e 0000:01:00.0: IPI 8 (power range: (-65, -60] dBm): ipi count = 716684
[  770.328299] mt7915e 0000:01:00.0: IPI 9 (power range: (-60, -55] dBm): ipi count = 574800
[  770.336464] mt7915e 0000:01:00.0: IPI 10 (power range: (-55, inf] dBm): ipi count = 4944143
[  770.344800] mt7915e 0000:01:00.0: IPI threshold 0: ipi_hist_count_th = 25961550, ipi_free_count = 25961550
[  770.354435] mt7915e 0000:01:00.0: TX assert time =  0 [ms]
[  770.359915] mt7915e 0000:01:00.0: band[1]: chan load = 100%, self idle ratio = 100%, idle ratio = 0%
```

Note that the "Antenna index: 6" in the above figure is because the antenna index starts from 4 for band index 1 of a DBDC chip. The IPI counter of each IPI index has a large number because of the noise in the environment I used. If the DUT is set in shielding box/room and the signal is transmitted via cable, then only the IPI counter of the IPI index would increase.

3.9 Zero Wait DFS

2.2.33.4 Reset Idle Power Indicator (IPI) Counter of Background Chain

Reset all the IPI counters to zero.

Input Argument	Description	Value
ipi_reset	Reset the IPI counter of the dedicated RX	1

You could also utilize the IPI reset CR to reset the IPI counter if the mt76 IPI reset command is not working.

Chip	IPI Reset CR Address	IPI Reset Method
mt7915	0x830AF070	Set the bit 27 of the CR to 1
mt7916	0x831A3008	Set the bit 2 of the CR to 1 *Note that the CR automatically resets bit 2 to 0 after setting it to 1.* *Don't worry if the value of bit 2 of CR remains 0.*
mt7981 mt7986	No dedicated RX	
mt7996	0x830A5DFC	Set the bit 28 of the CR to 1
mt7992		

- Command

```
mt76-test phy0 set ipi_reset=1
or
echo 0x831a3008 > /sys/kernel/debug/ieee80211/phy0/mt76/regidx/
echo 4 > /sys/kernel/debug/ieee80211/phy0/mt76/regval/
```

- Example

```
root@OpenWrt:/# iwpriv phy0 mac 830AF0B0
phy0        mac:[0x830AF0B0]:004e87f1
root@OpenWrt:/# mt76-test phy0 set ipi_reset=1
root@OpenWrt:/# iwpriv phy0 mac 830AF0B0
phy0        mac:[0x830AF0B0]:00001f9d
```

3.9 Zero Wait DFS

2.2.34 Enable TX Power Single SKU

Enable the single SKU table for verifying TX power in testmode.

Input Argument	Description	Value
sku_en	Enable/Disable single SKU in test mode	1/0

- Command

  `mt76-test phy0 set sku_en=1`

  Note:

  1. Please follow **MT76 Programming Guide Section 9** to add your single sku power limit table in the DTS file
  2. Please **set your country to "VV"** since our country would be set to VV in test mode.

```
&slot0 {
        mt7996@0,0 {
                reg = <0x0000 0 0 0 0>;
                device_type = "pci";
                mediatek,mtd-eeprom = <&factory 0x0>;
                power-limits {
                        r0 {
                                country = "VV";
                                txpower-2g {
                                        r0 {
                                                channels = <1 1>;
                                                txs_delta = <0 0 0>;
                                                rates-cck = <22 25 27 24>;
                                                rates-ofdm = <27 22 21 29 28 24 35 29>;
                                                rates-mcs =
                                                <1 22 26 20 22 32 30 30 30 28 30>,
                                                <1 26 30 34 30 20 22 40 22 32 28>,
                                                <2 126 126 126 126 126 126 126 126 126 126>;
```

  3. Please **do not use `tx_power` command** when sku_en is on.

- Example

  (i) `sku_en is off: TX power is decided by eeprom txpower field & tx_power`

```
root@OpenWrt:/# mt76-test phy0 dump | grep sku_en
sku_en=0
root@OpenWrt:/# cat /sys/kernel/debug/ieee80211/phy0/mt76/txpower_sku

Phy 0 TX Power Table (Channel 1)
                        1m     2m     5m    11m
CCK (TMAC)          :   48     48     48     48
                        6m     9m    12m    18m    24m    36m    48m    54m
OFDM (TMAC)         :   48     48     48     48     48     48     48     48
                    mcs0   mcs1   mcs2   mcs3   mcs4   mcs5   mcs6   mcs7
HT20 (TMAC)         :   48     48     48     48     48     48     48     47
                    mcs0   mcs1   mcs2   mcs3   mcs4   mcs5   mcs6   mcs7  mcs32
HT40 (TMAC)         :   48     48     48     48     48     48     47     48
                    mcs0   mcs1   mcs2   mcs3   mcs4   mcs5   mcs6   mcs7   mcs8   mcs9  mcs10  mcs11
VHT20 (TMAC)        :   48     48     48     48     48     48     48     47     46     46     45     45
VHT40 (TMAC)        :   48     48     48     48     48     48     48     47     46     46     45     45
VHT80 (TMAC)        :   48     48     48     48     48     48     48     48     48     48     48     48
VHT160 (TMAC)       :   48     48     48     48     48     48     48     48     48     48     48     48
HE26 (TMAC)         :   48     48     48     48     48     48     48     47     46     46     45     45
HE52 (TMAC)         :   48     48     48     48     48     48     48     47     46     46     45     45
HE106 (TMAC)        :   48     48     48     48     48     48     48     47     46     46     45     45
HE242 (TMAC)        :   48     48     48     48     48     48     48     47     46     46     45     45
HE484 (TMAC)        :   48     48     48     48     48     48     48     47     46     46     45     45
HE996 (TMAC)        :   48     48     48     48     48     48     48     48     48     48     48     48
HE2x996 (TMAC)      :   48     48     48     48     48     48     48     48     48     48     48     48
                    mcs0   mcs1   mcs2   mcs3   mcs4   mcs5   mcs6   mcs7   mcs8   mcs9  mcs10  mcs11  mcs12  mcs13  mcs14  mcs15
EHT26 (TMAC)        :   48     48     48     48     48     48     48     47     46     46     45     45     43     43     48     48
EHT52 (TMAC)        :   48     48     48     48     48     48     48     47     46     46     45     45     43     43     48     48
EHT106 (TMAC)       :   48     48     48     48     48     48     48     47     46     46     45     45     43     43     48     48
EHT242 (TMAC)       :   48     48     48     48     48     48     48     47     46     46     45     45     43     43     48     48
EHT484 (TMAC)       :   48     48     48     48     48     48     48     47     46     46     45     45     43     43     48     48
EHT996 (TMAC)       :   48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
EHT2x996 (TMAC)     :   48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
EHT4x996 (TMAC)     :   48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
EHT26_52 (TMAC)     :   48     48     48     48     48     48     48     47     46     46     45     45     43     43     48     48
EHT26_106 (TMAC)    :   48     48     48     48     48     48     48     47     46     46     45     45     43     43     48     48
EHT484_242 (TMAC)   :   48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
EHT996_484 (TMAC)   :   48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
EHT996_484_242 (TMAC):  48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
EHT2x996_484 (TMAC) :   48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
EHT3x996 (TMAC)     :   48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
EHT3x996_484 (TMAC) :   48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
```

(ii) sku_en is on: TX power is decided by the single SKU table

```
root@OpenWrt:/# mt76-test phy0 set sku_en=1
root@OpenWrt:/# mt76-test phy0 dump | grep sku_en
sku_en=1
root@OpenWrt:/# cat /sys/kernel/debug/ieee80211/phy0/mt76/txpower_sku

Phy 0 TX Power Table (Channel 1)
                        1m     2m     5m    11m
CCK (TMAC)          :   22     25     27     24
                        6m     9m    12m    18m    24m    36m    48m    54m
OFDM (TMAC)         :   27     22     21     29     28     24     35     29
                      mcs0   mcs1   mcs2   mcs3   mcs4   mcs5   mcs6   mcs7
HT20 (TMAC)         :   22     26     20     22     32     30     30     30
                      mcs0   mcs1   mcs2   mcs3   mcs4   mcs5   mcs6   mcs7  mcs32
HT40 (TMAC)         :   26     30     34     30     20     22     40     22     32
                      mcs0   mcs1   mcs2   mcs3   mcs4   mcs5   mcs6   mcs7   mcs8   mcs9  mcs10  mcs11
VHT20 (TMAC)        :   22     26     20     22     32     30     30     30     28     30      0      0
VHT40 (TMAC)        :   26     30     34     30     20     22     40     22     32     28      0      0
VHT80 (TMAC)        :   48     48     48     48     48     48     48     48     48     48      0      0
VHT160 (TMAC)       :   48     48     48     48     48     48     48     48     48     48      0      0
HE26 (TMAC)         :   20     36     32     30     32     38     40     32     20     32     28     30
HE52 (TMAC)         :   36     32     24     34     30     38     20     30     34     20     26     26
HE106 (TMAC)        :   30     22     38     32     34     34     38     20     24     22     26     24
HE242 (TMAC)        :   22     26     20     22     32     30     30     30     28     30     32     20
HE484 (TMAC)        :   26     30     34     30     20     22     40     22     32     28     36     34
HE996 (TMAC)        :   48     48     48     48     48     48     48     48     48     48     48     48
HE2x996 (TMAC)      :   48     48     48     48     48     48     48     48     48     48     48     48
                      mcs0   mcs1   mcs2   mcs3   mcs4   mcs5   mcs6   mcs7   mcs8   mcs9  mcs10  mcs11  mcs12  mcs13  mcs14  mcs15
EHT26 (TMAC)        :   48     48     48     48     48     48     48     47     46     46     45     45     43     43     48     48
EHT52 (TMAC)        :   48     48     48     48     48     48     48     47     46     46     45     45     43     43     48     48
EHT106 (TMAC)       :   48     48     48     48     48     48     48     47     46     46     45     45     43     43     48     48
EHT242 (TMAC)       :   48     48     48     48     48     48     48     47     46     46     45     45     43     43     48     48
EHT484 (TMAC)       :   48     48     48     48     48     48     48     47     46     46     45     45     43     43     48     48
EHT996 (TMAC)       :   48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
EHT2x996 (TMAC)     :   48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
EHT4x996 (TMAC)     :   48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
EHT26_52 (TMAC)     :   48     48     48     48     48     48     48     47     46     46     45     45     43     43     48     48
EHT26_106 (TMAC)    :   48     48     48     48     48     48     48     47     46     46     45     45     43     43     48     48
EHT484_242 (TMAC)   :   48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
EHT996_484 (TMAC)   :   48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
EHT996_484_242 (TMAC):  48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
EHT2x996_484 (TMAC) :   48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
EHT3x996 (TMAC)     :   48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
EHT3x996_484 (TMAC) :   48     48     48     48     48     48     48     48     48     48     48     48     48     48     48     48
```

2.3 atenl

atenl is a daemon for the communication between HQADLL command and driver. Moreover, it has command mode used for e2p operations. For specific e2p commands (for example, atenl -i phy0 -c "..."), the daemon bypasses HQADLL mode and terminates after the e2p command is done. atenl would run in the background in HQADLL mode.

Note:

- For the DBDC/TBTC device, please use the first phy index for the following commands.
- In HQADLL mode, do not start two atenl daemons with different interfaces. Please use “killall atenl” before starting another atenl process in HQADLL mode. If you use ated instead of atenl, you can skip this note because the ated wrapper handles the case correctly.
- In atenl, it will detect the SDK you use to assign the default bridge name. Currently, it supports OpenWrt (default bridge name = br-lan) and RDK-B (default bridge name = brlan0). For the other 3rd party SDK or using a non-default bridge, please specify the bridge name via the argument “-b” listed in the following table.
- To support writing back to flash memory without entering flash mode, please use -p to specify the partition name and offset. atenl supports MTD and eMMC flash write-back.

Argument	Description
-i	Specify interface
-u	Use unicast to send the response
-b	Specify bridge name
-c	Specific e2p command
-p	Specify flash partition name and offset for writing back
-h	Help

2.3.1 Start Daemon

Enter HQADLL mode

- Command

```
atenl -i phyX -u
```

2.3.2 Check EEPROM Mode

This command is used to:

1. Check the path of the temp EEPROM data for atenl
2. Indicate eFuse mode or flash mode (**here flash mode includes flash, binfile and default bin mode**)

- Command

```
atenl -i phy0 -c “eeprom file”
```

- Example

```
root@OpenWrt:/usr/sbin# atenl -i phy0 -c "eeprom file"
/tmp/atenl-eeprom
Flash mode: 1
```

2.3.3 Clear EEPROM Data Temp File

- Command

  `atenl -i phy0 -c "eeprom reset"`

- Example

```
root@OpenWrt:/usr/sbin# ls /tmp | grep atenl
atenl-eeprom
root@OpenWrt:/usr/sbin# atenl -i phy0 -c "eeprom reset"
root@OpenWrt:/usr/sbin# ls /tmp | grep atenl
root@OpenWrt:/usr/sbin#
```

2.3.4 Read EEPROM Data

- Command

```
# Read all
hexdump -C /tmp/atenl-eeprom
# Read specific offset
atenl -i phy0 -c "eeprom read 0x<offset>"
```

- Example

```
root@OpenWrt:/usr/sbin# hexdump -C /tmp/atenl-eeprom
00000000  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00005000  15 79 00 00 00 0c 43 2a  59 9f 00 0c 43 2a 59 a0  |.y....C*Y...C*Y.|
00005010  15 79 c3 14 00 80 02 00  15 79 c3 14 ed 9c 01 00  |.y.......y......|
00005020  16 79 c3 14 00 80 02 00  16 79 c3 14 00 00 01 00  |.y.......y......|
00005030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00005040  00 00 00 00 00 00 00 00  00 00 80 00 00 00 00 00  |................|
00005050  01 00 94 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00005060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00005070  30 1c 0f 00 00 00 07 b0  82 00 00 00 00 00 00 00  |0...............|
00005080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000050a0  00 00 00 00 00 00 00 00  ee 3c 0b d1 96 72 de d2  |.........<...r..|
000050b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00005190  24 24 08 00 28 00 00 15  00 00 00 00 00 00 00 00  |$$..(...........|
000051a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

2.3.5 Change Value to Specific Offset

This command only updates EEPROM data in the temp file. Writing back to flash or eFuse is required for the changes to take effect.

- Command

  `atenl -i phy0 -c "eeprom set 0x<offset>=0x<val>"`

- Example

```
root@OpenWrt:/usr/sbin# atenl -i phy0 -c "eeprom set 0x670=0x10"
set offset 0x670 to 0x10
```

- Check

```
*
00005670  10 00 00 00 00 00 00 00  00 00 00 09 0b 00 00 00  |................|
00005680  06 09 00 00 00 08 0a 00  02 00 07 0a 00 00 00 00  |................|
```

2.3.6 Update Buffer Mode

This command will send an MCU command to trigger the buffer mode update.
This is same as jedi's "iwpriv ra0 set bufferMode=2".

- Command

```
atenl -i phy0 -c "eeprom update buffermode"
```

2.3.7 Write Back EEPROM Data to Flash

Write back values of EEPROM data (/tmp/atenl-eeprom) to flash (mtd/eMMC factory).

- Command

```
atenl -i phy0 -c "eeprom write flash"
or
atenl -i phy0 -c "sync eeprom all"
or
atenl -i phy0 -c "sync eeprom all" -p <partion name>:<offset>
```

- Example: If the driver is not in flash mode or cannot enter flash mode, the user should provide **the flash partition name** and **offset** to write back the EEPROM data to flash memory.

(i) For MTD

```
# Check MTD partition name (case insensitive)
# Usually, the eeprom data is stored in the factory partition
cat /proc/mtd
```

```
root@OpenWrt:/# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 08000000 00020000 "spi0.0"
mtd1: 00100000 00020000 "BL2"
mtd2: 00080000 00020000 "u-boot-env"
mtd3: 00400000 00020000 "Factory"
mtd4: 00200000 00020000 "FIP"
mtd5: 07080000 00020000 "ubi"
```

```
# Write back
atenl -i phy0 -c "sync eeprom all" -p Factory:0x0
```

(ii) eMMC

```
# Check MTD partition name (case insensitive)
# Usually, the eeprom data is stored in the factory partition
blkid
```

```
root@OpenWrt:/# blkid
/dev/mmcblk0p1: PARTLABEL="u-boot-env" PARTUUID="54e99ba2-aa8d-11ef-b58e-0242c0a80002"
/dev/mmcblk0p2: PARTLABEL="factory" PARTUUID="54e9c334-aa8d-11ef-b58e-0242c0a80002"
/dev/mmcblk0p3: PARTLABEL="fip" PARTUUID="54e9e468-aa8d-11ef-b58e-0242c0a80002"
/dev/mmcblk0p4: PARTLABEL="kernel" PARTUUID="54ea04b6-aa8d-11ef-b58e-0242c0a80002"
/dev/mmcblk0p5: TYPE="squashfs" PARTLABEL="rootfs" PARTUUID="54ea2112-aa8d-11ef-b58e-0242c0a80002"
```

```
# Write back
atenl -i phy0 -c "sync eeprom all" -p factory:0x0
```

2.3.8 Write Back EEPROM Data to eFuse

Write back values of EEPROM data (/tmp/atenl-eeprom) to eFuse. **Please check the correctness of EEPROM data before executing this command.**

To protect the FT calibrated values stored in eFuse, the MT76 driver restricts the offset that can be updated. For more information about the protected eFuse field, please refer to this table.

- Command

  `atenl -i phy0 -c “eeprom write to efuse”`

2.3.9 Write Back EEPROM Data to Bin File

Write back values of EEPROM data (/tmp/atenl-eeprom) to bin file mode bin.

- Command

  `dd if=/tmp/atenl-eeprom-phy0 of=/your/bin/file`

```
root@OpenWrt:/# dd -h
BusyBox v1.33.2 (2023-03-04 12:09:12 UTC) multi-call binary.

Usage: dd [if=FILE] [of=FILE] [ibs=N obs=N/bs=N] [count=N] [skip=N] [seek=N]
        [conv=notrunc|noerror|sync|fsync]
        [iflag=skip_bytes|count_bytes|fullblock|direct] [oflag=seek_bytes|append|direct]

Copy a file with converting and formatting

        if=FILE         Read from FILE instead of stdin
        of=FILE         Write to FILE instead of stdout
        bs=N            Read and write N bytes at a time
        ibs=N           Read N bytes at a time
        obs=N           Write N bytes at a time
        count=N         Copy only N input blocks
        skip=N          Skip N input blocks
        seek=N          Skip N output blocks
        conv=notrunc    Don't truncate output file
        conv=noerror    Continue after read errors
        conv=sync       Pad blocks with zeros
        conv=fsync      Physically write data out before finishing
        conv=swab       Swap every pair of bytes
        iflag=skip_bytes        skip=N is in bytes
        iflag=count_bytes       count=N is in bytes
        oflag=seek_bytes        seek=N is in bytes
        iflag=direct    O_DIRECT input
        oflag=direct    O_DIRECT output
        iflag=fullblock Read full blocks
        oflag=append    Open output in append mode

N may be suffixed by c (1), w (2), b (512), kB (1000), k (1024), MB, M, GB, G
```

2.3.10 Write Back Pre-cal Data to Flash

Write back values of pre-cal data (stored in the driver) to flash (mtd factory).

Note:

- The first command gets the pre-cal data from the driver and writes the result to"/tmp/atenl-eeprom".
- Therefore, we need the second command to write back the result to flash.

- Command

  `atenl -i phy0 -c "eeprom precal sync"`
  `atenl -i phy0 -c “sync eeprom all”`

2.3.11 Clean Pre-cal Data in Flash

Cleaning the values of pre-cal data stored in flash (mtd factory).

Note:

1. The first and second commands clean the Group and DPD pre-cal data in "/tmp/atenl-eeprom", respectively. Therefore, we need the third command to write back the result to flash.
2. To clean the pre-cal data stored in the mt76 driver, please refer to Group pre-calibration and DPD pre-calibration.

- Command

```
atenl -i phy0 -c "eeprom precal group clean"
atenl -i phy0 -c "eeprom precal dpd clean"
atenl -i phy0 -c "sync eeprom all"
```

2.3.12 Sync iBF Cal Data from Driver

Sync iBF cal data in driver eeprom array to eeprom data (/tmp/atenl-eeprom).

Note:

1. This command is required when using mt76 command mode to perform iBF cal. mt76-test txbf e2p_update command **only copies iBF cal data from pfmu_data to eeprom array in driver.** Therefore**,** atenl eeprom data (/tmp/atenl-eeprom) will not have iBF cal result after entering mt76-test txbf e2p_update command.
2. For mt76 HQADLL mode, this command is **not necessary**. HQA iBF phase eeprom update command would sync iBF cal data in driver eeprom array to atenl eeprom data (/tmp/atenl-eeprom). Therefore, just enter atenl -i phy0 -c "sync eeprom all" to write back to flash is fine.

- Command

```
atenl -i phy0 -c "eeprom ibf sync"
```

2.4 Wrapper

2.4.1 iwpriv Wrapper

The wrapper is a shell script placed in /usr/sbin/iwpriv. It is used to convert frequently used iwpriv ATE commands to mt76-test (or atenl for some cases). For more information, please refer to iwpriv mapping table.

Note that the iwpriv wrapper supports interface name conversion for AX8400, AX7800, AX6000, AX3000, BE19000 (mt7988a + mt7996), BE14000 (mt7988d + mt7996 2 adie TBTC), BE7200 (mt7988d + mt7992), and BE5040 (mt7988d + mt7992). Therefore, you can simply enter jedi's or logan's commands.

For Wi-Fi 7 chipsets, the iwpriv wrapper also supports translating mwctl commands to mt76-test commands.

For debugging purposes, iwpriv wrapper provides the following command to switch the work mode of the iwpriv wrapper.

- Command

```
iwpriv ra0 set WORKMODE=<RUN/PRINT/DEBUG>
```

Note:

1. The value is all caps.
2. This command is "iwpriv wrapper" specific, not jedi's iwpriv command.
3. The work mode is chip specific. That is, for example AX8400, if we only set the work mode of mt7986 to "DEBUG", then mt7915 would not be in DEBUG mode.

Argument	Description	Value
WORKMODE	Default mode Directly run the translated mt76-test commands	RUN
	Only print the translated mt76-test commands	PRINT
	Run and print the translated mt76-test commands	DEBUG

2.4.2 ated Wrapper

The wrapper is a shell script placed in /usr/sbin/ated. It is similar to setting a symbolic link, with interface name conversion (ra0 → phy0, rax0 → phy1) and additional clean up command. For more information, please refer to ated mapping table.

3 Example and Cross Reference

In this section, some testing examples and corresponding commands for our proprietary driver are provided.

3.1 TX VHT40 MCS9 ANT1 Band0

MT76	Jedi/Logan
# Start testmode by enabling monitor interface iw dev phy0-ap0 del mt76-test phy0 add mon0 iw dev mon0 set channel 36 HT40+ mt76-test phy0 set tx_ipg=50 tx_rate_mode=vht tx_rate_idx=9 tx_rate_sgi=0 tx_rate_ldpc=1 tx_length=1024 tx_antenna=1 tx_count=10000000 freq_offset=42 tx_power=40,0,0,0 # (optional) check parameters mt76-test phy0 dump # Start tx mt76-test phy0 set state=tx_frames # Dump tx status (tx_queued, tx_done) mt76-test phy0 dump stats	iwpriv ra0 set ATE=ATESTART iwpriv ra0 set ATECTRLBANDIDX=0 iwpriv ra0 set ATEIPG=50 iwpriv ra0 set ATETXMODE=4 iwpriv ra0 set ATETXMCS=9 iwpriv ra0 set ATETXBW=1:1 iwpriv ra0 set ATETXGI=0 iwpriv ra0 set ATETXLDPC=1 iwpriv ra0 set ATETXLEN=1024 iwpriv ra0 set ATETXANT=1 iwpriv ra0 set ATERXANT=1 iwpriv ra0 set ATETXCNT=10000000 iwpriv ra0 set ATETXFREQOFFSET=42 iwpriv ra0 set ATECHANNEL=36:1 iwpriv ra0 set ATETXPOW0=40 iwpriv ra0 set ATE=TXCOMMIT iwpriv ra0 set ATE=TXFRAME

3.2 TX HE80 MCS11 ANT8 Band1

MT76	Jedi/Logan
iw dev phy1-ap0 del mt76-test phy1 add mon1 iw dev mon1 set channel 40 80MHz mt76-test phy1 set tx_ipg=50 tx_rate_mode=he_su tx_rate_idx=11 tx_rate_sgi=0 tx_ltf=1 tx_rate_ldpc=1 tx_length=4096 tx_antenna=8 tx_count=10000000 freq_offset=42 tx_power=38,0,0,0 mt76-test phy1 set state=tx_frames mt76-test phy1 dump stats	iwpriv ra0 set ATE=ATESTART iwpriv ra0 set ATECTRLBANDIDX=1 iwpriv ra0 set ATEIPG=50 iwpriv ra0 set ATETXMODE=8 iwpriv ra0 set ATETXMCS=11 iwpriv ra0 set ATETXBW=2:2 iwpriv ra0 set ATETXGI=1 iwpriv ra0 set ATETXLDPC=1 iwpriv ra0 set ATETXLEN=4096 iwpriv ra0 set ATETXANT=8 iwpriv ra0 set ATERXANT=8 iwpriv ra0 set ATETXCNT=10000000 iwpriv ra0 set ATETXFREQOFFSET=42 iwpriv ra0 set ATECHANNEL=40:1 iwpriv ra0 set ATETXPOW0=38 iwpriv ra0 set ATE=TXCOMMIT iwpriv ra0 set ATE=TXFRAME

3.3 RX VHT20 ANT3 Band0

MT76	Jedi/Logan
iw dev phy0-ap0 del mt76-test phy0 add mon0 iw dev mon0 set channel 36 HT20 mt76-test phy0 set tx_rate_mode=vht tx_antenna=3 tx_rate_idx=0 # This command will also clean counters mt76-test phy0 set state=rx_frames # Dump RX status mt76-test phy0 dump stats	iwpriv ra0 set ATE=ATESTART iwpriv ra0 set ATECTRLBANDIDX=0 iwpriv ra0 set ATETXBW=0:0 iwpriv ra0 set ATETXANT=3 iwpriv ra0 set ATERXANT=3 iwpriv ra0 set ATETXMODE=4 iwpriv ra0 set ATETXMCS=0 iwpriv ra0 set ATECHANNEL=36:1 iwpriv ra0 set ATE=RXFRAME iwpriv ra0 set ATERXSTATRESET=0 iwpriv ra0 set ATERXSTAT=0

3.4 Duplicate TX

MT76	Jedi/Logan
iw dev phy0-ap0 del mt76-test phy0 add mon0 iw dev mon0 set channel 36 80MHz mt76-test phy0 set tx_ipg=50 tx_rate_mode=vht tx_rate_idx=8 tx_rate_sgi=1 tx_rate_ldpc=1 tx_rate_nss=2 tx_antenna=3 tx_length=4096 tx_count=1000000 tx_power=40,0,0,0 tx_spe_idx=24 mt76-test phy0 set state=tx_frames mt76-test phy0 dump stats	iwpriv ra0 set ATE=ATESTART iwpriv ra0 set ATECTRLBANDIDX=0 iwpriv ra0 set ATEIPG=50 iwpriv ra0 set ATETXMODE=4 iwpriv ra0 set ATETXMCS=8 iwpriv ra0 set ATETXBW=2:2 iwpriv ra0 set ATETXGI=1 iwpriv ra0 set ATETXLDPC=1 iwpriv ra0 set ATEVHTNSS=2 iwpriv ra0 set ATETXANT=3 iwpriv ra0 set ATERXANT=3 iwpriv ra0 set ATETXLEN=4096 iwpriv ra0 set ATETXCNT=1000000 iwpriv ra0 set ATETXFREQOFFSET=42 iwpriv ra0 set ATECHANNEL=36 iwpriv ra0 set ATETXPOW0=40 iwpriv ra0 set ATETXANT=1:24 iwpriv ra0 set ATE=TXCOMMIT iwpriv ra0 set ATE=TXFRAME

3.5 Continuous TX

This mode will **NOT** send normal packets, only has signal waveforms.

To leave continuous TX mode to send some normal packets again, we need to stop and start test mode.

MT76	Jedi/Logan
iw dev phy0-ap0 del mt76-test phy0 add mon0 iw dev mon0 set channel 6 HT20 mt76-test phy0 set tx_rate_mode=ofdm tx_rate_idx=7 tx_count=0 tx_power=40,0,0,0 tx_antenna=1 # For mt7916, mt7981, and mt7986 **mt76-test phy0 set aid=1** mt76-test phy0 set state=tx_cont mt76-test phy0 set state=idle mt76-test phy0 del mon0 iw phy phy0 interface add phy0-ap0 type managed ifconfig phy0-ap0 up	iwpriv ra0 set ATE=ATESTART iwpriv ra0 set ATECTRLBANDIDX=0 iwpriv ra0 set RBIST_SwitchMode=1 iwpriv ra0 set ATETXMODE=1 iwpriv ra0 set ATETXMCS=7 iwpriv ra0 set ATETXBW=0 iwpriv ra0 set ATETXCNT=0 iwpriv ra0 set ATETXPOW0=40 iwpriv ra0 set ATETXANT=1 iwpriv ra0 set ATECHANNEL=6:0 iwpriv ra0 set ATE=TXCONT iwpriv ra0 set ATE=TXCONTSTOP iwpriv ra0 set ATE=TXFRAME iwpriv ra0 set RBIST_SwitchMode=0 iwpriv ra0 set ATE=ATESTOP

3.6 Pre-calibration

Take **AX8400** for example. Assume both mt7915 & mt7986 are in flash mode.

Note that `atenl -i phy1 -c "eeprom precal sync" & atenl -i phy1 -c " sync eeprom all "` could be skipped since the buffer file in atenl would be the same for dbdc chip.

MT76	Jedi
# mt7915: 5G iw dev phy0-ap0 del mt76-test phy0 add mon0 mt76-test phy0 set state=group_prek mt76-test phy0 set state=dpd_5g atenl -i phy0 -c "eeprom precal sync" atenl -i phy0 -c "sync eeprom all" mt76-test phy0 del mon0 iw phy phy0 interface add phy0-ap0 type managed ifconfig phy0-ap0 up # mt7986: 2G iw dev phy1-ap0 del mt76-test phy1 add mon1 mt76-test phy1 set state=group_prek mt76-test phy1 set state=dpd_6g # Optional atenl -i phy1 -c "eeprom precal sync" atenl -i phy1 -c "sync eeprom all"	# mt7915: 5G iwpriv rai0 set ATE=ATESTART iwpriv rai0 set ATE=GROUPREK iwpriv rai0 set ATE=DPD5G ated -i rai0 -c "sync eeprom all" iwpriv rai0 set ATE=ATESTOP # mt7986: 2G iwpriv ra0 set ATE=ATESTART iwpriv ra0 set ATE=GROUPREK iwpriv ra0 set ATE=DPD2G ated -i ra0 -c "sync eeprom all" iwpriv ra0 set ATE=ATESTOP # mt7986: 6G iwpriv rax0 set ATE=ATESTART iwpriv rax0 set ATE=DPD6G ated -i rax0 -c "sync eeprom all" iwpriv rax0 set ATE=ATESTOP

MT76	Jedi
mt76-test phy1 del mon1 iw phy phy1 interface add phy1-ap0 type managed ifconfig phy1-ap0 up # mt7986: 6G iw dev phy2-ap0 del mt76-test phy2 add mon2 mt76-test phy2 set state=dpd_6g atenl -i phy2 -c "eeprom precal sync" atenl -i phy2 -c "sync eeprom all" mt76-test phy2 del mon2 iw phy phy2 interface add phy2-ap0 type managed ifconfig phy2-ap0 up	

Take BE19000 (mt7996) for **Wi-Fi 7 chipset** as an example. Assume it is in flash mode. **In Wi-Fi 7, we recommend setting up all the test mode monitor interfaces before taking any actions.**

MT76	Logan
# Testmode interface setup iw dev phy0-ap0 del iw dev phy1-ap0 del iw dev phy2-ap0 del ## mt7996: 2G mt76-test phy0 add mon0 ## mt7996: 5G mt76-test phy1 add mon1 ## mt7996: 6G mt76-test phy2 add mon2 # Start calibration mt76-test phy0 set state=group_prek mt76-test **phy0** set state=dpd_2g mt76-test **phy1** set state=dpd_5g mt76-test **phy2** set state=dpd_6g # Save calibration result to flash atenl -i phy0 -c "eeprom precal sync" atenl -i phy0 -c "sync eeprom all" # Testmode interface delete mt76-test phy0 del mon0 mt76-test phy1 del mon1 mt76-test phy2 del mon2	# Testmode interface setup ## mt7996: 2G, 5G & 6G at once iwpriv ra0 set ATE=ATESTART # Start calibration iwpriv ra0 set ATE=GROUPREK iwpriv ra0 set ATE=DPD2G iwpriv ra0 set ATE=DPD5G iwpriv ra0 set ATE=DPD6G # Save calibration result to flash ated -i ra0 -c "sync eeprom all" # Testmode interface delete iwpriv ra0 set ATE=ATESTOP

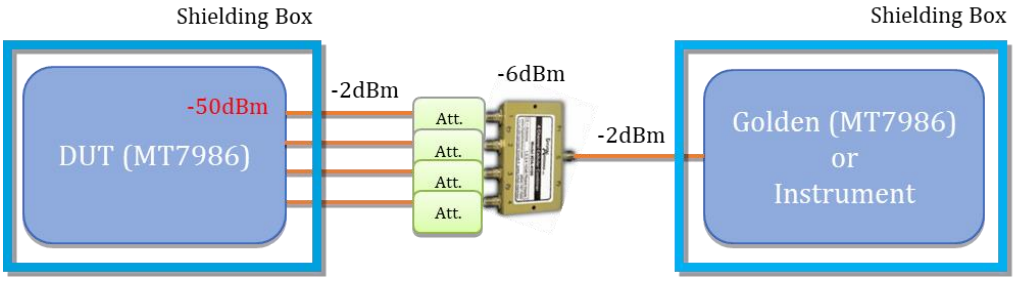
3.7 Implicit Beamforming

The testing environment is shown below. The iBF calibration and verification can be done by the golden device or instrument such as IQXel.

There are some prerequisites that should be satisfied.

- The RX target power should be **-50dBm** with 3dBm variation only. If it is not satisfied, then calibration would fail.
  - As shown in the figure, the combiner has 6dBm loss and each cable loss is 2dBm. The TX power of instrument is fixed at -40dBm.
  - Check the DUT's RX power by RX statistic dump.
  - If the RX target power is too large or too small, please add an attenuator or adjust the TX power of golden.
  - You could also check the TX power of the golden device via the following command.

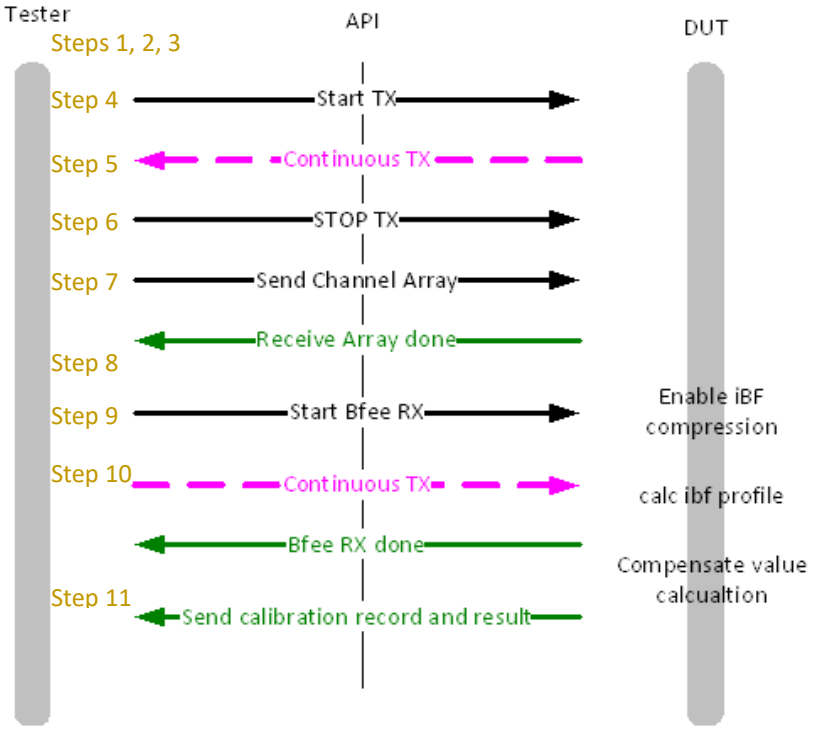
    `cat /sys/kernel/debug/ieee80211/phyX/mt76/txpower_sku,` for Wi-Fi 6, Wi-Fi 7 multi-Wiphy
    `cat /sys/kernel/debug/ieee80211/phy0/mt76/bandX/txpower_sku,` for Wi-Fi 7 single Wiphy
- iBF DUT and Golden **MUST** be done within separate shielding boxes.
- Conductive calibration is necessary. Over-the-air (OTA) is not recommended.



3.7.1 DUT Command for Calibration with Instrument

3.7.1.1 Calibration

Perform iBF calibration using the test instrument. The procedure is illustrated in the following figure.



For TX antenna ≤ 4: (Wi-Fi 6 chipsets, mt7996, mt7992 (except BE7200 iFEM 5G))

Step	MT76	Jedi/Logan
0	# Start the first channel group's group M iBF calibration & verification ## For saving testing time, we only do calibration & verification on group M <group>=$\begin{cases} 0, \ for\ 2G \\ 1 \sim 8, \ for\ 5G \\ 1 \sim 12, \ for\ \boldsymbol{mt7992\ 5G} \end{cases}$ <band_idx>=$\begin{cases} 0, \ for\ 2G \\ 1, \ for\ 5G \end{cases}$ <channel>=group_to_chan(<group>) <MCS>=$\begin{cases} 15 + 8 * (n_{ss} - 2), \ 1 < n_{ss} \leq 4 \\ 31, \ otherwise \end{cases}$ <wlan_idx>=$\begin{cases} 1, \ otherwise \\ \boldsymbol{2, \ mt7996 \ / \ mt7992\ 5G} \end{cases}$ <ver>=$\begin{cases} 0, \ otherwise\ iBF\ 1.0 \\ \boldsymbol{3, \ mt7992\ iBF\ 2.0} \end{cases}$	
1	# Start iBF TX Calibration iw dev phy0-ap0 del mt76-test phy0 add mon0 mt76-test phy0 set txbf_act=init txbf_param=1	# Start iBF TX Calibration iwpriv ra0 set ATE=ATESTART ## Skip it for Wi-Fi 7 iwpriv ra0 set ATECTRLBANDIDX=<band_idx> iwpriv ra0 set ATETxBfInit=<band_idx>
2	# Group 0 ~ 8 iw dev mon0 set channel <channel> # Group 9 ~ 12 (only for iBF 2.0) ## Use CBW 160 DBW 20 mt76-test phy1 set tx_pkt_bw=20 tx_pri_sel=4 iw dev mon1 set channel <channel> 160MHz	# Group 0 ~ 8 iwpriv ra0 set ATECHANNEL=<channel>:<band_idx> # Group 9 ~ 12 (only for iBF 2.0) ## Use CBW 160 DBW 20 iwpriv rai0 set ATETXBW=5:0 iwpriv rai0 set ATECHANNEL=<channel>:<band_idx>:4:0
3	# Clear compensated TX/RX phases mt76-test phy0 set txbf_act=phase_comp txbf_param=0,<band_idx>,<group>,0,1 aid=1	# Clear compensated TX/RX phases iwpriv ra0 set ATEIBFPhaseComp=00:<band_idx>:<group>:00:01

Step	MT76	Jedi/Logan
4	# Configure eBF PFMU profie for channel update mt76-test phy0 set txbf_act=ebf_prof_update txbf_param=1,3,0 # Prepare for TX mt76-test phy0 set tx_rate_idx=<MCS> mt76-test phy0 set txbf_act=apply_tx txbf_param=<wlan_idx>,1,0,0,1 # Configure BF TXCMD (**only Wi-Fi 7 required**) **mt76-test phy0 set txbf_act=txcmd txbf_param=1,1,1** mt76-test phy0 set txbf_act=tx_prep txbf_param=0,1,<wlan_idx>,0 mt76-test phy0 set aid=1 tx_count=10000000 tx_length=1024 # Start TX mt76-test phy0 set state=tx_frames	# Configure eBF PFMU profie for channel update iwpriv ra0 set ATEEBfProfileConfig=01:03:00 # Prepare and start TX iwpriv ra0 set ATETXMCS=<MCS> iwpriv ra0 set TxBfTxApply=<wlan_idx>:01:00:00:01 # Configure BF TXCMD (**only Wi-Fi 7 required**) **iwpriv ra0 set TxBfTxCmd=1:1:1** iwpriv ra0 set ATETxPacketWithBf=00:<wlan_idx>:00
5	# Instrument start to calculate channel profile	
6	# Stop TX mt76-test phy0 set state=idle	# Stop TX iwpriv ra0 set ATETxPacketWithBf=00:<wlan_idx>:01
7	# Instrument writes channel profile to DUT mt76-test phy0 set txbf_act=prof_update_all txbf_param=01,f0 mt76-test phy0 set txbf_act=prof_update_all txbf_param=00,0000,0000,0000,0000,… mt76-test phy0 set txbf_act=prof_update_all txbf_param=01,ff	# Instrument writes channel profile to DUT iwpriv ra0 set TxBfProfileData20MAllWrite=01:F0 iwpriv ra0 set TxBfProfileData20MAllWrite= 00:0000:0000:0000:0000: 01:0000:0000:0000:0000… iwpriv ra0 set TxBfProfileData20MAllWrite= 08:035f:0322:0031:00f2:09:0337:0312:007c:00af… … iwpriv ra0 set TxBfProfileData20MAllWrite=01:FF
8	# Configure iBF PFMU profie for channel update mt76-test phy0 set txbf_act=ibf_prof_update txbf_param=2,3,0	# Configure iBF PFMU profie for channel update iwpriv ra0 set ATEIBfProfileConfig=02:03:00
9	# Enable DUT’s RX mt76-test phy0 set state=rx_frames	# Enable DUT’s RX iwpriv ra0 set ATE=RXFRAME
	# Instrument start to TX packet to DUT (OFDM 54M)	
10	# Enable iBF RX mt76-test phy0 set txbf_act=apply_tx txbf_param=<wlan_idx>,0,1,0,1 # Start to do the iBF phase calibration mt76-test phy0 set txbf_act=phase_cal txbf_param=<group>,1,<band_idx>,3,1,<ver>	# Enable iBF RX iwpriv ra0 set TxBfTxApply=<wlan_idx>:00:01:00:01 # Start to do the iBF phase calibration iwpriv ra0 set ATEIBfInstCal=<group>:01:<band_idx>:03:01:<ver>

For 5T5R 4SS (mt7992 BE7200 iFEM 5G band):

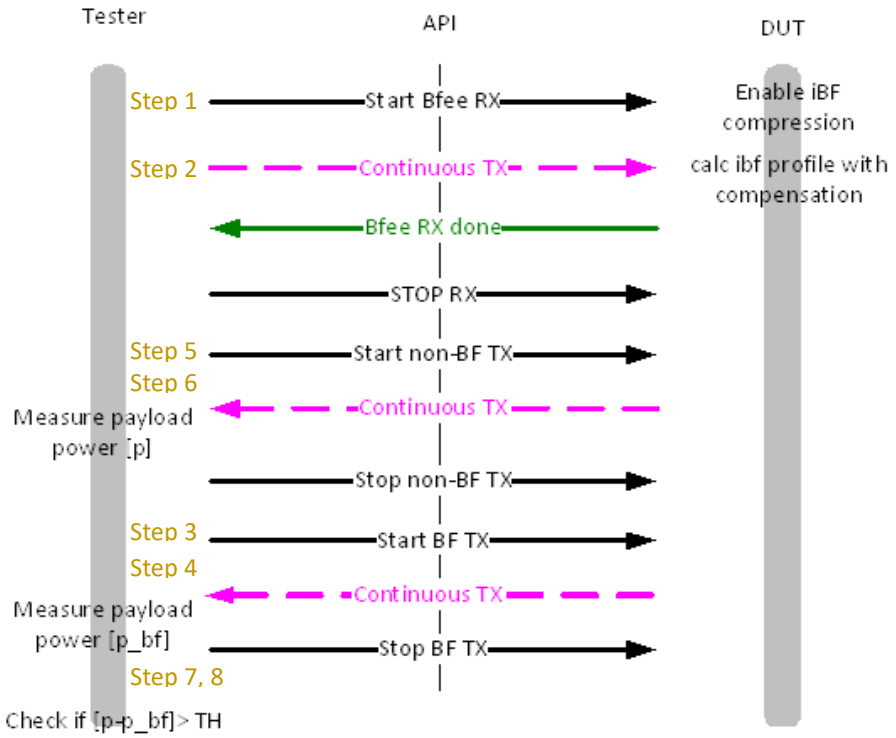
To support 5T5R iBF, the DUT is required to transmit a 5x5 iBF profile to the instrument for calibration. However, the chip only supports 4 spatial streams (SS), so the packet needs to be transmitted twice with different TX antenna settings to enable the instrument to combine these two 4x4 iBF profiles into a 5x5 iBF profile. As a result, the **iBF commands for chips supporting 5T5R 4SS differ from those for other chips**.

Step	MT76	Logan
0	# Start the first channel group's group M iBF calibration & verification ## For saving testing time, we only do calibration & verification on group M <group>=1 ~ 12 <band_idx>=1 <channel>=group_to_chan(<group>) <wlan_idx>=2 <ver>=3	
1	# Start iBF TX Calibration iw dev phy1-ap0 del mt76-test phy1 add mon1 mt76-test phy1 set txbf_act=init txbf_param=1	# Start iBF TX Calibration iwpriv rai0 set ATE=ATESTART iwpriv rai0 set ATETxBfInit=<band_idx>
2	mt76-test phy1 set tx_antenna=31 tx_rate_nss=4 # For group 1 ~ 8 iw dev mon1 set channel <channel> # For group 9 ~ 12 ## Use CBW 160 DBW 20 mt76-test phy1 set tx_pkt_bw=20 tx_pri_sel=4 iw dev mon1 set channel <channel> 160MHz	iwpriv rai0 set ATETXNSS=4 iwpriv rai0 set ATETXANT=31 # For group 1 ~ 8 iwpriv rai0 set ATECHANNEL=<channel>:<band_idx> # For group 9 ~ 12 ## Use CBW 160 DBW 20 iwpriv rai0 set ATETXBW=5:0 iwpriv rai0 set ATECHANNEL=<channel>:<band_idx>:4:0
3	# Unstable Tx power caused by GT switching can be avoided by fixing pre-GT (**only required for 7977 Adie**) echo 0x8318FD30 > /sys/kernel/debug/ieee80211/phy0/mt76/regidx echo 0 > /sys/kernel/debug/ieee80211/phy0/mt76/regval # Clear compensated TX/RX phases mt76-test phy1 set txbf_act=phase_comp txbf_param=0,<band_idx>,<group>,0,1 aid=1	# Unstable Tx power caused by GT switching can be avoided by fixing pre-GT (**only required for 7977 Adie**) mwctl phy phy0 mac 8318FD30=0 # Clear compensated TX/RX phases iwpriv rai0 set ATEIBFPhaseComp=00:<band_idx>:<group>:00:01
4	# Configure eBF PFMU profie for channel update mt76-test phy1 set txbf_act=ebf_prof_update txbf_param=1,4,0 # Prepare for the first TX mt76-test phy1 set tx_antenna=23 tx_rate_nss=4 mt76-test phy1 set txbf_act=apply_tx txbf_param=<wlan_idx>,1,0,0,1 # Configure BF TXCMD mt76-test phy1 set txbf_act=txcmd txbf_param=1,1,1 mt76-test phy1 set txbf_act=tx_prep txbf_param=0,1,<wlan_idx>,0 mt76-test phy1 set aid=1 tx_count=10000000 tx_length=1024 # Start TX mt76-test phy1 set state=tx_frames	# Configure eBF PFMU profie for channel update iwpriv rai0 set ATEEBfProfileConfig=01:04:00 # Prepare and start the first TX iwpriv rai0 set ATETXNSS=4 iwpriv rai0 set ATETXANT=23 ## Start TX commit, not necessary mwctl rai0 set HQA=SetTestEng=0x1,18 iwpriv rai0 set TxBfTxApply=<wlan_idx>:01:00:00:01 # Configure BF TXCMD iwpriv rai0 set TxBfTxCmd=1:1:1 iwpriv rai0 set ATETxPacketWithBf=00:<wlan_idx>:00

Step	MT76	Logan
5	# Unstable Tx power caused by GT switching can be avoided by fixing pre-GT (**only required for 7977 Adie**) echo 0x8318FD30 > /sys/kernel/debug/ieee80211/phy0/mt76/regidx echo 0 > /sys/kernel/debug/ieee80211/phy0/mt76/regval # Prepare for the second TX mt76-test phy1 set tx_antenna=27 tx_rate_nss=4 mt76-test phy1 set txbf_act=apply_tx txbf_param=<wlan_idx>,1,0,0,1 mt76-test phy1 set txbf_act=txcmd txbf_param=1,1,1 mt76-test phy1 set txbf_act=tx_prep txbf_param=0,1,<wlan_idx>,0 mt76-test phy1 set aid=1 tx_count=10000000 tx_length=1024 # Start TX mt76-test phy1 set state=tx_frames	# Unstable Tx power caused by GT switching can be avoided by fixing pre-GT (**only required for 7977 Adie**) mwctl phy phy0 mac 8318FD30=0 # Prepare for the second TX iwpriv rai0 set ATETXNSS=4 iwpriv rai0 set ATETXANT=27 # Start the second TX iwpriv rai0 set TxBfTxApply=<wlan_idx>:01:00:00:01 # Configure BF TXCMD iwpriv rai0 set TxBfTxCmd=1:1:1 iwpriv rai0 set ATETxPacketWithBf=00:<wlan_idx>:00
6	# Instrument start to calculate channel profile	
7	# Stop TX mt76-test phy1 set state=idle	# Stop TX iwpriv rai0 set ATETxPacketWithBf=00:<wlan_idx>:01
8	# Instrument writes channel profile to DUT ## 5 angle tuple for each subcarrier mt76-test phy1 set txbf_act=prof_update_all txbf_param=01,f0 mt76-test phy1 set txbf_act=prof_update_all txbf_param=00,0000,0000,0000,0000,0000,01,… mt76-test phy1 set txbf_act=prof_update_all txbf_param=01,ff	# Instrument writes channel profile to DUT ## 5 angle tuple for each subcarrier iwpriv rai0 set TxBfProfileData20MAllWrite=01:F0 iwpriv rai0 set TxBfProfileData20MAllWrite= 00:0000:0000:0000:0000:0000: 01:0000:0000:0000:0000… iwpriv rai0 set TxBfProfileData20MAllWrite= 08:00ac:03b0:00a5:068b:033d:09:0094:0397:008d:0272:0324… iwpriv rai0 set TxBfProfileData20MAllWrite=01:FF
9	# Enable DUT’s RX mt76-test phy1 set tx_antenna=31 mt76-test phy1 set state=rx_frames	# Enable DUT’s RX iwpriv rai0 set ATETXANT=31 iwpriv rai0 set ATE=RXFRAME
10	# Configure iBF PFMU profie for channel update mt76-test phy1 set txbf_act=ibf_prof_update txbf_param=2,4,0	# Configure iBF PFMU profie for channel update iwpriv rai0 set ATEIBfProfileConfig=02:04:00
	# Instrument start to TX packet to DUT	
11	# Enable iBF RX mt76-test phy1 set txbf_act=apply_tx txbf_param=<wlan_idx>,0,1,0,1 # Start to do the iBF phase calibration mt76-test phy1 set txbf_act=phase_cal txbf_param=<group>,1,<band_idx>,3,1,<ver>	# Enable iBF RX iwpriv rai0 set TxBfTxApply=<wlan_idx>:00:01:00:01 # Start to do the iBF phase calibration iwpriv rai0 set ATEIBfInstCal=<group>:01:<band_idx>:03:01:<ver>

3.7.1.2 Verification

Verify the calibrated iBF data through iBF gain with the test instrument, and the procedure is shown in the following figure. Note that the verification result is unaffected by the order of non-BF TX and BF-TX.



For TX antenna ≤ 4: (Wi-Fi 6 chipsets, mt7996, mt7992 (except BE7200 iFEM 5G))

Step	MT76	Jedi/Logan
1	# Start iBF TX Verification # Compensate TX / RX phase # <bf_on>=$\begin{cases}0, & for\ group\ 0 \sim 8\\3, & for\ group\ 9 \sim 12\end{cases}$ mt76-test phy0 set txbf_act=phase_comp txbf_param=<bf_on>,<band_idx>,<group>,0,0 aid=1	# Start iBF TX Verification # Compensate TX / RX phase # <bf_on>=$\begin{cases}0, & for\ group\ 0 \sim 8\\3, & for\ group\ 9 \sim 12\end{cases}$ iwpriv ra0 set ATEIBFPhaseComp=<bf_on>:<band_idx>:<group>:00:00
2	# Enable DUT's RX mt76-test phy0 set state=rx_frames # Instrument start to TX packet (OFDM 54M)	# Enable DUT's RX iwpriv ra0 set ATE=RXFRAME # Instrument start to TX packet (OFDM 54M)
3	# Set MCS rate mt76-test phy0 set tx_rate_idx=4 # Start BF TX mt76-test phy0 set txbf_act=tx_prep txbf_param=1,1,<wlan_idx>,0 aid=1 tx_count=10000000 tx_length=1024 mt76-test phy0 set state=tx_frames	# Set MCS rate iwpriv ra0 set ATETXMCS=4 # Start BF TX iwpriv ra0 set ATETxPacketWithBf=01:<wlan_idx>:00
4	# Instrument start to measure the averaged power of BF TX (P_BF)	
5	# Start non-BF TX mt76-test phy0 set txbf_act=tx_prep txbf_param=0,1,<wlan_idx>,0 aid=1 tx_count=10000000 tx_length=1024 mt76-test phy0 set state=tx_frames	# Start non-BF TX iwpriv ra0 set ATETxPacketWithBf=00:<wlan_idx>:00

Step	MT76	Jedi/Logan
6	# Instrument start to measure the averaged power of non-BF TX (P_NBF)	
7	# Stop DUT's TX mt76-test phy0 set state=idle	# Stop DUT's TX iwpriv ra0 set ATE=TXSTOP
	# Instrument calculates iBF gain via P_BF – P_NBF, it should be > 10 dB for 4x4 AP to 1x1 STA	
8	# Write the calibrated phases of groups into eeprom mt76-test phy0 set txbf_act=e2p_update txbf_param=0,0,1 atenl -i phy0 -c "eeprom ibf sync" # Write the result to flash atenl -i phy0 -c "sync eeprom all"	# Write the calibrated phases of groups into eeprom iwpriv ra0 set ATEIBFPhaseE2pUpdate=00:00:01 # Write the result to flash ated -i phy0 -c "sync eeprom all"

For 5T5R 4SS (mt7992 BE7200 iFEM 5G band):

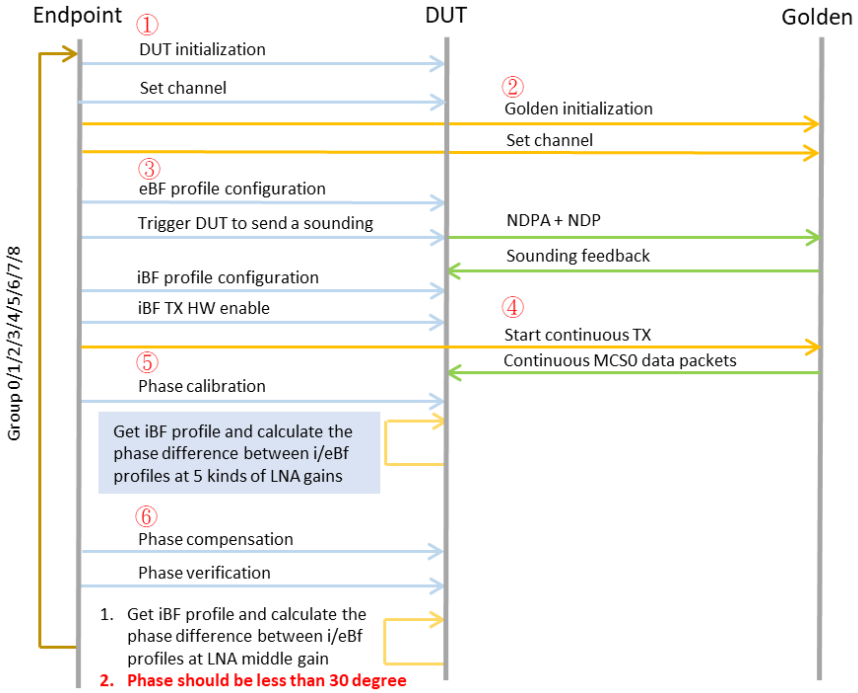
Step	MT76	Jedi/Logan
1	# Start iBF TX Verification # Compensate TX / RX phase # <bf_on>=$\begin{cases}0, & for\ group\ 0 \sim 8\\3, & for\ group\ 9 \sim 12\end{cases}$ mt76-test phy1 set txbf_act=phase_comp txbf_param=<bf_on>,<band_idx>,<group>,0,0 aid=1	# Start iBF TX Verification # Compensate TX / RX phase # <bf_on>=$\begin{cases}0, & for\ group\ 0 \sim 8\\3, & for\ group\ 9 \sim 12\end{cases}$ iwpriv rai0 set ATEIBFPhaseComp=<bf_on>:<band_idx>:<group>:00:00
2	# Enable DUT's RX mt76-test phy1 set state=rx_frames # Instrument start to TX packet (OFDM 54M)	# Enable DUT's RX iwpriv rai0 set ATE=RXFRAME # Instrument start to TX packet (OFDM 54M)
3	# Set MCS rate & NSS mt76-test phy1 set tx_rate_idx=4 tx_rate_nss=1 # Start BF TX mt76-test phy1 set txbf_act=tx_prep txbf_param=1,1,<wlan_idx>,0 aid=1 tx_count=10000000 tx_length=1024 mt76-test phy1 set state=tx_frames # Stop BF TX mt76-test phy1 set txbf_act=tx_prep txbf_param=1,1,<wlan_idx>,0 aid=1 tx_count=1 tx_length=1024 mt76-test phy1 set state=tx_frames	# Set MCS rate & NSS iwpriv rai0 set ATETXMCS=4 iwpriv rai0 set ATETXNSS=1 # Start BF TX iwpriv rai0 set ATETxPacketWithBf=01:<wlan_idx>:00 # Stop BF TX iwpriv rai0 set ATETxPacketWithBf=00:<wlan_idx>:01
4	# Instrument start to measure the averaged power of BF TX (P_BF)	
5	# Start non-BF TX mt76-test phy1 set tx_antenna=1 mt76-test phy1 set txbf_act=tx_prep txbf_param=0,1,<wlan_idx>,0 aid=1 tx_count=10000000 tx_length=1024 mt76-test phy1 set state=tx_frames # Stop non-BF TX mt76-test phy1 set txbf_act=tx_prep txbf_param=0,1,<wlan_idx>,0 aid=1 tx_count=1 tx_length=1024 mt76-test phy1 set state=tx_frames	# Start non-BF TX iwpriv rai0 set ATETXANT=1 iwpriv rai0 set ATETxPacketWithBf=00:<wlan_idx>:00 # Stop non-BF TX iwpriv rai0 set ATETxPacketWithBf=00:<wlan_idx>:01
6	# Instrument start to measure the averaged power of non-BF TX (P_NBF)	
7	# Stop DUT's TX mt76-test phy1 set state=idle	# Stop DUT's TX iwpriv rai0 set ATE=TXSTOP
	# Instrument calculates iBF gain via P_BF – P_NBF, it should be > 10 dB for 4x4 AP to 1x1 STA	

Step	MT76	Jedi/Logan
8	# Write the calibrated phases of groups into eeprom mt76-test phy1 set txbf_act=e2p_update txbf_param=0,0,1 atenl -i phy0 -c “eeprom ibf sync” # Write the result to flash atenl -i phy0 -c “sync eeprom all”	# Write the calibrated phases of groups into eeprom iwpriv rai0 set ATEIBFPhaseE2pUpdate=00:00:01 # Write the result to flash ated -i phy0 -c “sync eeprom all”

3.7.2 DUT Command for Calibration with Golden

3.7.2.1 Calibration and Verification

Do iBF calibration with the golden device, and the procedure is shown in the following figure (orange for Golden, blue for DUT, and white for common). This test procedure is a template suitable for different channel bands and chips. **Note that please use the proprietary driver for the golden device.**



Step	MT76	Jedi/Logan
0	# Start the first channel group’s group M iBF calibration & verification ## For saving testing time, we only do calibration & verification on group M <group>=$\begin{cases} 0, & for\ 2G \\ 1 \sim 8, & for\ 5G \end{cases}$ <band_idx>=$\begin{cases} 0, & for\ 2G \\ 1, & for\ 5G \end{cases}$ <channel>=group_to_chan(<group>) <wlan_idx>=$\begin{cases} 1, & otherwise \\ \mathbf{2}, & \boldsymbol{Eagle\ 5G} \end{cases}$	

Step	MT76	Jedi/Logan
1	# DUT init iw dev phy0-ap0 del mt76-test phy0 add mon0 mt76-test phy0 set txbf_act=init txbf_param=1 mt76-test phy0 set aid=1 ## Set channel iw dev mon0 set channel <channel> HT20	# DUT init iwpriv ra0 set ATE=ATESTART ## This CR is chip dependent, we handle it in mt76 and iwpriv wrapper. Not required in mt76 anymore. ~~iwpriv ra0 mac 820E3030=301~~ ## DUT init iwpriv ra0 set ATETxBfInit=<band_idx> ## Set channel iwpriv ra0 set ATECHANNEL=<channel>:<band_idx>
2	# Please use proprietary driver	# Golden device init iwpriv ra0 set ATE=ATESTART iwpriv ra0 set ATETxBfGdInit=<band_idx> ## Set channel iwpriv ra0 set ATECHANNEL=<channel>:<band> iwpriv ra0 set ATE=RXFRAME ## Not required in mt76 ~~iwpriv ra0 ATETXCNT=1~~ ~~iwpriv ra0 set ATE=TXFRAME~~
3	# Clear compensated TX/RX phases mt76-test phy0 set txbf_act=phase_comp txbf_param=0,<band_idx>,<group>,0,1 # Enable DUT's RX mt76-test phy0 set state=rx_frames # eBF profile configuration mt76-test phy0 set txbf_act=ebf_prof_update txbf_param=1,3,0 # Apply eBF profile mt76-test phy0 set txbf_act=apply_tx txbf_param=<wlan_idx>,1,0,0,1 # Configure BF TXCMD (**only Wi-Fi 7 is required**) **mt76-test phy0 set txbf_act=txcmd txbf_param=1,1,1** # Trigger sounding mt76-test phy0 set txbf_act=trigger_sounding txbf_param=2,1,0c,<wlan_idx>,0,0,0 # Stop sounding after 1 sec mt76-test phy0 set txbf_act=stop_sounding txbf_param=0 # iBF profile configuration mt76-test phy0 set txbf_act=ibf_prof_update txbf_param=2,3,0 # Apply iBF profile mt76-test phy0 set txbf_act=apply_tx txbf_param=<wlan_idx>,0,1,0,1	# Clear compensated TX/RX phases iwpriv ra0 set ATEIBFPhaseComp=00:<band_idx>:<group>:00:01 # Enable DUT's RX iwpriv ra0 set ATE=RXFRAME # eBF profile configuration iwpriv ra0 set ATEEBfProfileConfig=01:03:00 # Apply eBF profile iwpriv ra0 set TxBfTxApply=<wlan_idx>:01:00:00:01 # Configure BF TXCMD (**only Wi-Fi 7 is required**) **iwpriv ra0 set TxBfTxCmd=1:1:1** # Trigger sounding iwpriv ra0 set TriggerSounding=02:01:0C:<wlan_idx>:00:00:00 # Stop sounding after 1 sec iwpriv ra0 set StopSounding=1 # iBF profile configuration iwpriv ra0 set ATEIBfProfileConfig=02:03:00 # Apply iBF profile iwpriv ra0 set TxBfTxApply=<wlan_idx>:00:01:00:01
3-1 Sounding golden debug step	# Please use proprietary driver	# Check Bfee RX NDP & trigger BFRP (for debugging) iwpriv ra0 mac ${BFEE_RX_NDP_ADDR} # Check Bfee RX NDPA & TX feedback report (for debugging) iwpriv ra0 mac ${BFEE_RX_NDPA_ADDR}
3-2 Sounding DUT debug step	# Check Bfer RX feedback report counter (for debugging) echo ${BFER_RX_FBK_ADDR} > /sys/kernel/debug/ieee80211/phy0/mt76/regidx cat /sys/kernel/debug/ieee80211/phy0/mt76/regval # Check eBF profile if not receiving feedback mt76-test phy0 set txbf_act=pfmu_tag_read txbf_param=1,1	# Check Bfer RX feedback report counter (for debugging) iwpriv ra0 mac ${BFER_RX_FBK_ADDR} # Check eBF profile if not receiving feedback iwpriv ra0 set TxBfProfileTagRead=01:01 # Check iBF profile if not receiving feedback iwpriv ra0 set TxBfProfileTagRead=02:01 # Check BF station record if not receiving feedback

Step	MT76	Jedi/Logan
	# Check iBF profile if not receiving feedback mt76-test phy0 set txbf_act=pfmu_tag_read txbf_param=2,1 # Check BF station record if not receiving feedback mt76-test phy0 set txbf_act=sta_rec_read txbf_param=1	iwpriv ra0 set StaRecBfRead=01
4	# Please use proprietary driver	# Golden start continuous TX ## Set TX count to a large number iwpriv ra0 set ATETXCNT=0 ## Set golden's TX power if required (iwpriv ra0 set ATETXPOW0=<val>) iwpriv ra0 set ATE=TXFRAME
4-1 debug step	# Check Bfer TX packet applied iBF counter echo ${BF_APP_CNT_ADDR} > /sys/kernel/debug/ieee80211/phy0/mt76/regidx cat /sys/kernel/debug/ieee80211/phy0/mt76/regval # If counter is counting, then try ${BF_NOT_APP_REASON}	# Check Bfer TX packet applied iBF counter iwpriv ra0 mac ${BF_APP_CNT_ADDR} # If counter is counting, then try ${BF_NOT_APP_REASON}
5	# Phase calibration mt76-test phy0 set txbf_act=phase_cal txbf_param=<group>,1,<band_idx>,1	# Phase calibration iwpriv ra0 set ATEIBfGdCal=<group>:01:<band_idx>:01
6	# Phase verification ## Phase compensation mt76-test phy0 set txbf_act=phase_comp txbf_param=1,<band_idx>,<group>,0,0 ## Phase verification mt76-test phy0 set txbf_act=phase_cal txbf_param=<group>,1,<band_idx>,2,1	# Phase verification iwpriv ra0 set ATEIBFPhaseVerify=<group>:01:<band_idx>:02:01:00
7	# Please use proprietary driver	# Stop golden's TX iwpriv ra0 set ATE=TXSTOP
8	# Move on to the iBF calibration for the next group or stop the procedure if all groups are done <group>=<group>+1 <channel>=group_to_chan(<group>) **# Back to the channel setting part in step 1 & 2**	
9	# Write the calibrated phases of groups into eeprom mt76-test phy0 set txbf_act=e2p_update txbf_param=0,0,1 atenl -i phy0 -c "eeprom ibf sync" # Write the result to flash atenl -i phy0 -c "sync eeprom all"	# Write the calibrated phases of groups into eeprom iwpriv ra0 set ATEIBFPhaseE2pUpdate=00:00:01 # Write the result to flash ated -i phy0 -c "sync eeprom all"

3.7.3 iBF Calibration Channel Group

In the iBF calibration and verification test, we would not calibrate and verify all channels due to testing time and cost constraints. Instead, we classify the channel into several channel groups and select the **median channel** (this can be changed by setting the group L, M, and H in phase calibration) to represent the entire group. According to the RF test, the BF gain loss of the boundary channel in each channel group is no more than 0.5 dBm.

Note that the 5G channel group differs for Wi-Fi 6 and Wi-Fi 7 chipsets.

Wi-Fi 6 chipset (mt7915, mt7916, mt7981, mt7986):

Channel Group		Group Range	Calibrated Channel	Frequency (Unit: MHz)	EEPROM Address	
2G	0	Channel: [1, 14] Frequency: [2412, 2484]	8	2447	iPA + iLNA (Adie: 7975)	[0x651, 0x678]
					ePA + eLNA (Adie: 7976)	[0x60A, 0x631]
5G	1	Channel: [184, 196] & [8,16] Frequency: [4920, 5080]	196	4980	iPA + iLNA (Adie: 7975)	[0x679, 0x6A0]
					ePA + eLNA (Adie: 7976)	[0x632, 0x659]
	2	Channel: [36, 48] Frequency: [5180, 5240]	44	5220	iPA + iLNA (Adie: 7975)	[0x6A1, 0x6C8]
					ePA + eLNA (Adie: 7976)	[0x65A, 0x681]
	3	Channel: [52, 68] Frequency: [5260, 5340]	60	5300	iPA + iLNA (Adie: 7975)	[0x6C9, 0x6F0]
					ePA + eLNA (Adie: 7976)	[0x682, 0x6A9]
	4	Channel: [72, 92] Frequency: [5360, 5460]	84	5420	iPA + iLNA (Adie: 7975)	[0x6F1, 0x718]
					ePA + eLNA (Adie: 7976)	[0x6AA, 0x6D1]
	5	Channel: [96, 112] Frequency: [5480, 5560]	104	5520	iPA + iLNA (Adie: 7975)	[0x719, 0x740]
					ePA + eLNA (Adie: 7976)	[0x6D2, 0x6F9]
	6	Channel: [116, 136] Frequency: [5580, 5680]	124	5620	iPA + iLNA (Adie: 7975)	[0x741, 0x768]
					ePA + eLNA (Adie: 7976)	[0x6FA, 0x721]
	7	Channel: [140, 157] Frequency: [5700, 5785]	149	5745	iPA + iLNA (Adie: 7975)	[0x769, 0x790]
					ePA + eLNA (Adie: 7976)	[0x722, 0x749]
	8	Channel: [161, 181] Frequency: [5805, 5905]	173	5865	iPA + iLNA (Adie: 7975)	[0x791, 0x7B8]
					ePA + eLNA (Adie: 7976)	[0x74A, 0x771]

Wi-Fi 7 iBF 1.0 (mt7996):

Channel Group		Group Range	Calibrated Channel	Frequency (Unit: MHz)	EEPROM Address
2G	0	Channel: [1, 14] Frequency: [2412, 2484]	8	2447	[0xC00, 0xC2D]
5G	1	Channel: [36, 48] Frequency: [5180, 5240]	44	5220	[0xC2E, 0xC5B]
	2	Channel: [52, 64] Frequency: [5260, 5320]	60	5300	[0xC5C, 0xC89]
	3	Channel: [68, 96]	84	5420	[0xC8A, 0xCB7]

Channel Group		Group Range	Calibrated Channel	Frequency (Unit: MHz)	EEPROM Address
		Frequency: [5340, 5480]			
	4	Channel: [100, 112] Frequency: [5500, 5560]	104	5520	[0xCB8, 0xCE5]
	5	Channel: [116, 128] Frequency: [5580, 5640]	124	5620	[0xCE6, 0xD13]
	6	Channel: [132, 144] Frequency: [5660, 5720]	136	5680	[0xD14, 0xD41]
	7	Channel: [149, 161] Frequency: [5745, 5805]	153	5765	[0xD42, 0xD6F]
	8	Channel: [165, 181] Frequency: [5825, 5905]	173	5865	[0xD70, 0xD9D]

Wi-Fi 7 iBF 2.0 (mt7992):

Channel Group		Group Range	Calibrated Channel	Frequency (Unit: MHz)	EEPROM Address
2G	0	Channel: [1, 14] Frequency: [2412, 2484]	8	2447	[0xC00, 0xC1C]
5G	1	Channel: [36, 48] Frequency: [5180, 5240]	44	5220	[0xC1D, 0xC3E]
	2	Channel: [52, 64] Frequency: [5260, 5320]	60	5300	[0xC3F, 0xC60]
	3	Channel: [68, 96] Frequency: [5340, 5480]	84	5420	[0xC61, 0xC82]
	4	Channel: [100, 112] Frequency: [5500, 5560]	104	5520	[0xC83, 0xCA4]
	5	Channel: [116, 128] Frequency: [5580, 5640]	124	5620	[0xCA5, 0xCC6]
	6	Channel: [132, 144] Frequency: [5660, 5720]	136	5680	[0xCC7, 0xCE8]
	7	Channel: [149, 161] Frequency: [5745, 5805]	153	5765	[0xCE9, 0xD0A]
	8	Channel: [165, 181] Frequency: [5825, 5905]	173	5865	[0xD0B, 0xD2C]
	9	Channel: [36, 64] Frequency: [5180, 5240] **CBW: 160 MHz, DBW: 20 MHz**	52	5260	[0xD2D, 0xD4E]
	10	Channel: [68, 96] Frequency: [5340, 5480] **CBW: 160 MHz, DBW: 20 MHz**	84	5420	[0xD4F, 0xD70]
	11	Channel: [100, 128] Frequency: [5500, 5640] **CBW: 160 MHz, DBW: 20 MHz**	116	5580	[0xD71, 0xD92]
	12	Channel: [149, 181] Frequency: [5745, 5905] **CBW: 160 MHz, DBW: 20 MHz**	165	5825	[0xD93, 0xDB4]

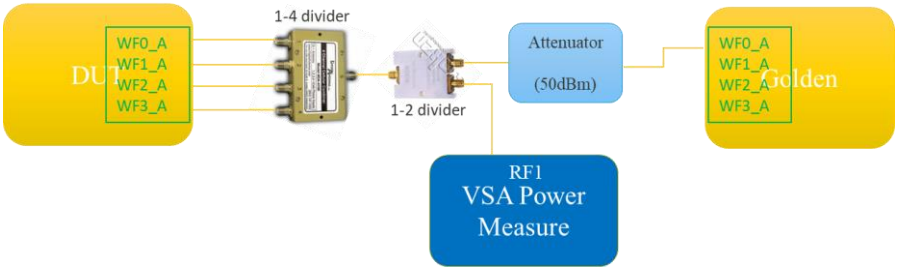
3.8 Explicit Beamforming

3.8.1 Certification

As we all know, beamforming mechanism will get power gain due to in-phase signal by different TX antenna. When applying eBF in normal mode TX, we need to compensate the BF power gain to meet the power constraint. As a result, the goal of this certification is to create the eBF power gain table based on the measured power gain, also known as the TXBF power backoff table, and to test the functionality of eBF.

There are some certification criteria, such as continuous TX, duty cycle > 98%, and manually controlled BF on/off. We simulate the continuous TX by setting a large TX count number, and BF on/off could be triggered by the command listed below. The eBF certification laboratory's duty cycle > 98% criterion for measuring TX power is out of date. They can now accept package mode and measure power gain without the requirement of duty cycle > 98%.

This eBF test needs a "DUT" and a "Golden Unit" to realize gain boosting, where the DUT and the Golden Unit play the roles of beamformer and beamformee, respectively. The testing environment is shown in the figure below. **For DBDC mt7915, the beamformer (DUT) would be 2T instead of 4T, as shown in the following figure.**



The following is an example of eBF certification for **mt7986 2G HT20 (AX6000)**. For other testcases, please modify the interface (phy0), channel, BW, and TX rate mode. The commands of DUT and golden need to **be entered in sequence** during the test (orange for Golden, blue for DUT, and white for common). **Note that please use the proprietary driver for the golden device.**

Step	MT76	Jedi
1	# Start eBF certification ## Do nothing	# Start eBF certification iwpriv ra0 set ATEEBFCE=1 iwpriv ra0 set ATECTRLBANDIDX=0
2	# Please use proprietary driver	# Golden device init iwpriv ra0 set ATEConTxETxBfGdProc=02:00:00:006:000:0
3	# DUT init ## Start testmode for DUT iw dev phy0-ap0 del mt76-test phy0 add mon0 ## Switch channel iw dev mon0 set channel 6 HT20 ## Set TX parameters (tx_count: use a large num to simulate cont. TX) mt76-test phy0 set tx_rate_mode=ht tx_rate_idx=0 tx_rate_nss=1 tx_rate_sgi=0	# DUT init iwpriv ra0 set ATEConTxETxBfInitProc=02:00:00:01:04:18:006:000:0:04000

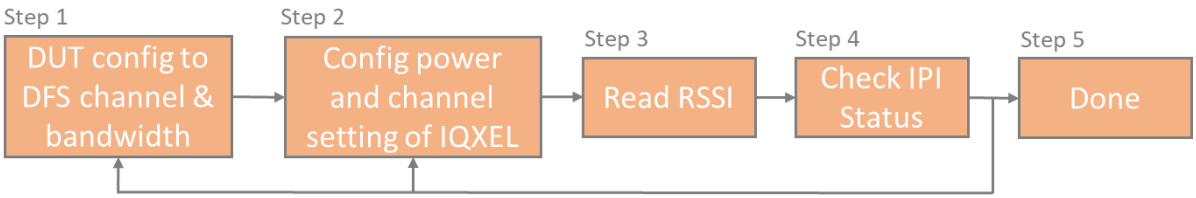
Step	MT76	Jedi
	tx_rate_ldpc=1 tx_power=18,0,0,0 tx_count=10000000 tx_length=4000 tx_ipg=4 ## eBF DUT init mt76-test phy0 set txbf_act=ebf_init txbf_param=1 ## Create virtual WTBL entry mt76-test phy0 set aid=1 ## Stop auto sounding mt76-test phy0 set txbf_act=stop_sounding txbf_param=1 ## Update channel information mt76-test phy0 set txbf_act=update_ch txbf_param=1 ## Update eBF profile (PFMU tag & STA record) mt76-test phy0 set txbf_act=ebf_prof_update txbf_param=0,0,0 ## Apply eBF TX settings (PFMU tag & STA record) mt76-test phy0 set txbf_act=apply_tx txbf_param=1,1,0,0,0 ## Trigger one-shot sounding packet mt76-test phy0 set txbf_act=trigger_sounding txbf_param=0,1,0,1,0,0,0 ## Trigger periodic sounding packet mt76-test phy0 set txbf_act=trigger_sounding txbf_param=2,1,ff,1,0,0,0 ## Start RX (for receiving BF sounding feedback packet) mt76-test phy0 set state=rx_frames	
4 Sounding golden debug step	# Please use proprietary driver	# Check Bfee RX NDP & trigger BFRP (for debugging) iwpriv ra0 mac ${BFEE_RX_NDP_ADDR} # Check Bfee RX NDPA & TX feedback report (for debugging) iwpriv ra0 mac ${BFEE_RX_NDPA_ADDR}
5 Sounding DUT debug step	# Check Bfer RX feedback report counter (for debugging) echo ${BFER_RX_FBK_ADDR} > /sys/kernel/debug/ieee80211/phy0/mt76/regidx cat /sys/kernel/debug/ieee80211/phy0/mt76/regval	# Check Bfer RX feedback report counter (for debugging) iwpriv ra0 mac ${BFER_RX_FBK_ADDR}
6	# Start BF TX mt76-test phy0 set state=tx_frames	# Start BF TX iwpriv ra0 set ATE=TXFRAME
7	# Check Bfer TX packet applied eBF counter echo ${BF_APP_CNT_ADDR} > /sys/kernel/debug/ieee80211/phy0/mt76/regidx cat /sys/kernel/debug/ieee80211/phy0/mt76/regval # If counter is counting, then try ${BF_NOT_APP_REASON}	# Check Bfer TX packet applied eBF counter iwpriv ra0 mac ${BF_APP_CNT_ADDR} # If counter is counting, then try ${BF_NOT_APP_REASON}
8	# Use the instrument IQxel to check BF ON waveform	
9	# Changing TX power ## This step could be skipped if it is not required # Stop TX first	# Changing TX power ## This step could be skipped if it is not required # Stop TX first iwpriv ra0 set ATE=TXSTOP

Step	MT76	Jedi
	mt76-test phy0 set state=idle # Re-configure TX power mt76-test phy0 set tx_power=30,0,0,0 Go back to step 6 to re-trigger continuous TX	# Re-configure TX power iwpriv ra0 set ATETXPOW0=30 Go back to step 6 to re-trigger continuous TX
10	# Turn off BF ## Stop sounding ## BF Off and it takes a few seconds to take effect mt76-test phy0 set txbf_act=stop_sounding txbf_param=1 ## Set PFMU tag invalid bit to true mt76-test phy0 set txbf_act=set_invalid_prof txbf_param=1 ## Update PFMU tag mt76-test phy0 set txbf_act=pfmu_tag_write txbf_param=0 ## Read PFMU tag to check if it is updated successfully mt76-test phy0 set txbf_act=pfmu_tag_read txbf_param=0,1	# Turn off BF ## Stop sounding ## BF Off and it takes a few seconds to take effect iwpriv ra0 set StopSounding=1 ## Set PFMU tag invalid bit to true iwpriv ra0 set TxBfProfileTagInValid=1 ## Update PFMU tag iwpriv ra0 set TxBfProfileTagWrite=0 ## Read PFMU tag to check if it is updated successfully iwpriv ra0 set TxBfProfileTagRead=00:01
11	# Check Bfer TX packet applied eBF counter ## It should be 0 after turning off eBF echo ${BF_APP_CNT_ADDR} > /sys/kernel/debug/ieee80211/phy0/mt76/regidx cat /sys/kernel/debug/ieee80211/phy0/mt76/regval	# Check Bfer TX packet applied eBF counter ## It should be 0 after turning off eBF iwpriv ra0 mac ${BF_APP_CNT_ADDR}
12	# Use the instrument IQxel to check BF OFF waveform	
13	# Resume BF On ## The way of MT76 resume BF on is different from jedi **MT76 should go back to step 3 & 6 to resume BF on** ## Therefore, if you are using iwpriv commands in MT76 (based on iwpriv wrapper), you should enter the ATEConTxETxBfInitProc and TXFRAME command instead of TriggerSounding listed in the right table	# Resume BF On iwpriv ra0 set TriggerSounding=02:01:FF:01:00:00:00

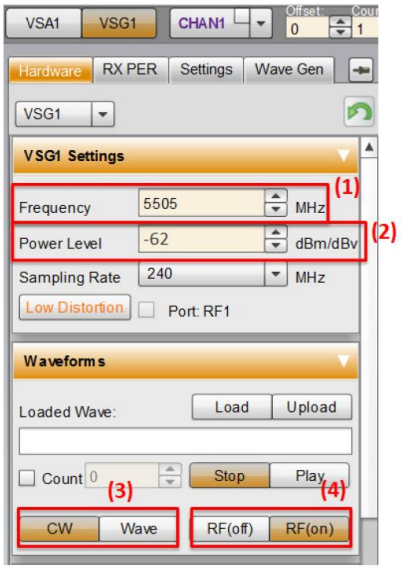
3.9 Zero Wait DFS

Verify the hardware capability of the dedicated RX via LitePoint instrument (IQXEL). The testing procedure is illustrated in the following figure.

The test method here is to utilize a simple Continuous Wave (CW) tone to trigger the update of the IPI counter (see here for the introduction of IPI). Since CW tone is a signal with stable power and is also not a packet (no PD done), only the IPI counter within the power range of CW tone would increase endlessly. We use this indicator to check if the hardware of the dedicated RX could receive the signal correctly.



1. Enable the dedicated RX of DUT via this command.
2. Set the VSG pattern and power of IQXEL. Please follow the steps to configure your instrument.
   (1) Set the center frequency you want to test.
   Note that, due to physical reasons, the frequency should have 5 MHz offset to avoid the DC tone. If CW hits on the DC tone, then the instrument cannot detect the signal properly.
   (2) Set the power level you want to test. The recommended test range is -62dBm.
   (3) Select CW tone.
   (4) Click the “RF(on)” button.



3. Get the RSSI value via reading control register (CR). The address of the CR of RSSI is listed in the following table. If the read RSSI value has a 1~2dBm offset due to cable loss, then it is acceptable. Otherwise, the test failed. The IPI counter would not be correct anyway.

Chip	RSSI CR Address	RSSI Reading Method [dBm]
mt7915	0x830AF0E0	ADC RSSI = CR[15:8] – 256 (in decimal) **IB RSSI = CR[7:0] – 256 (in decimal)** ***IB RSSI is the RSSI in which we are interested***
mt7916	0x831203E0	RSSI = CR[31:24] – 256 (in decimal)
mt7981 mt7986	-	-
mt7996 mt7992	0x830403E0	RSSI = CR[31:24] – 256 (in decimal)

4. Check the IPI status via reading IPI CR or IPI show histogram command. The IPI counter of the corresponding IPI index should increase faster than the other IPI indexes.
5. If you are going to test another power range or channel, please reset the IPI counter. And then, go back to step 1 or 2 according to your test requirements. Otherwise, the test is done.

Wi-Fi 6 chipset (mt7915, mt7916):

Step	MT76	Jedi
1	# Start ZWDFS Verification iw dev phy0-ap0 del mt76-test phy0 add mon0 # Enable the dedicated RX mt76-test phy0 set offchan_ch=100 offchan_bw=80	# Start ZWDFS Verification iwpriv ra0 set ATE=ATESTART # Enable the dedicated RX iwpriv ra0 set DfsRxCtrl=100:2
2	# Setup Instrument	
3	# Read RSSI CR echo ${RSSI_ADDR} > /sys/kernel/debug/ieee80211/phy0/mt76/regidx cat /sys/kernel/debug/ieee80211/phy0/mt76/regval # Calculate RSSI = CR[31:24] - 256	# Read RSSI CR iwpriv ra0 mac ${RSSI_ADDR} # Calculate RSSI = CR[31:24] - 256
4	# Read IPI status (e.g. IPI index 8) # Method 1: IPI histogram show command mt76-test phy0 set ipi_threshold=8 ipi_period=100 # Method 2: read IPI histogram CR echo ${IPI_ADDR} > /sys/kernel/debug/ieee80211/phy0/mt76/regidx cat /sys/kernel/debug/ieee80211/phy0/mt76/regval	# Read IPI status (e.g. IPI index 8) # Method 1: IPI histogram show command iwpriv ra0 set DfsRxHist=8:100 # Method 2: read IPI histogram CR iwpriv ra0 mac ${IPI_ADDR}
5	# Reset the IPI counter or finish test mt76-test phy0 set ipi_reset=1 or echo ${IPI_RESET_ADDR} > /sys/kernel/debug/ieee80211/phy0/mt76/regidx echo ${RESET_VALUE} > /sys/kernel/debug/ieee80211/phy0/mt76/regval # Back to step 1 or 2	# Reset the IPI counter or finish test iwpriv ra0 mac ${IPI_RESET_ADDR}=${RESET_VALUE} # Back to step 1 or 2

Wi-Fi 7 chipset (mt7996, mt7992):

For the Wi-Fi 7 chipset, the HW module of the dedicated RX (ZWDFS) lies in band 0, so band 0 should also be enabled while using ZWDFS.

Step	MT76	Logan
1	# Start ZWDFS Verification mt76-test phy0 add mon0 mt76-test phy1 add mon1 # Enable band 0 RX mt76-test phy0 set state=rx_frames # Enable the dedicated RX mt76-test phy1 set offchan_ch=100 offchan_bw=80	# Start ZWDFS Verification iwpriv ra0 set ATE=ATESTART iwpriv rai0 set ATE=ATESTART # Enable band 0 RX iwpriv ra0 set ATE=RXFRAME # Enable the dedicated RX iwpriv rai0 set DfsRxCtrl=100:2
2	# Setup Instrument	
3	# Read RSSI CR echo ${RSSI_ADDR} > /sys/kernel/debug/ieee80211/phy0/mt76/regidx cat /sys/kernel/debug/ieee80211/phy0/mt76/regval # Calculate RSSI = CR[31:24] - 256	# Read RSSI CR iwpriv ra0 mac ${RSSI_ADDR} # Calculate RSSI = CR[31:24] - 256

Step	MT76	Logan
4	# Read IPI status (e.g. IPI index 8) # Method 1: IPI histogram show command mt76-test phy0 set ipi_threshold=8 ipi_period=100 # Method 2: read IPI histogram CR echo ${IPI_ADDR} > /sys/kernel/debug/ieee80211/phy0/mt76/regidx cat /sys/kernel/debug/ieee80211/phy0/mt76/regval	# Read IPI status (e.g. IPI index 8) # Method 1: IPI histogram show command iwpriv ra0 set DfsRxHist=8:100 # Method 2: read IPI histogram CR iwpriv ra0 mac ${IPI_ADDR}
5	# Reset the IPI counter or finish test mt76-test phy0 set ipi_reset=1 or echo ${IPI_RESET_ADDR} > /sys/kernel/debug/ieee80211/phy0/mt76/regidx echo ${RESET_VALUE} > /sys/kernel/debug/ieee80211/phy0/mt76/regval # Back to step 1 or 2	# Reset the IPI counter or finish test iwpriv ra0 mac ${IPI_RESET_ADDR}=${RESET_VALUE} # Back to step 1 or 2

4 iwpriv (mwctl) and ated Command Mapping Table

The mapping between iwpriv/ated commands and mt76-test/atenl commands is shown in the table below. **It should be noted that iwpriv/ated commands that are not listed here are not currently supported by MT76.** To check how the iwpriv commands are mapped by the mt76-test commands, please enable iwpriv wrapper debug mode.

4.1 Iwpriv (mwctl)

4.1.1 Set State

Jedi/Logan	MT76
Basic commands	
iwpriv ra0 set ATE=ATESTART	Enter monitor mode
iwpriv ra0 set ATE=ATESTOP	state=off and leave monitor mode
iwpriv ra0 set ATE=TXFRAME	state=tx_frames
iwpriv ra0 set ATE=TXSTOP	state=idle
iwpriv ra0 set ATE=TXCOMMIT	aid=1
iwpriv ra0 set ATE=TXREVERT	aid=0
iwpriv ra0 set ATE= RXFRAME	state=rx_frames
iwpriv ra0 set ATE=RXSTOP	state=idle
iwpriv ra0 set ATE=TXCONT	state=tx_cont
Pre-cal commands	
iwpriv ra0 set ATE=GROUPREK	state=group_prek Write Back Pre-cal Data to Flash
iwpriv ra0 set ATE=GROUPREKDump	state=group_prek_dump
iwpriv ra0 set ATE=GROUPREKClean	state=group_prek_clean
iwpriv ra0 set ATE=DPD2G	state=dpd_2g Write Back Pre-cal Data to Flash
iwpriv ra0 set ATE=DPD5G	state=dpd_5g Write Back Pre-cal Data to Flash
iwpriv ra0 set ATE=DPD6G	state=dpd_6g Write Back Pre-cal Data to Flash
iwpriv ra0 set ATE=DPDDump	state=dpd_dump
iwpriv ra0 set ATE=DPDClean	state=dpd_clean

4.1.2 Set Configs

Jedi/Logan	MT76
Basic commands	
iwpriv ra0 set ATECTRLBANDIDX=<val>	Use phyX to control
iwpriv ra0 set ATETXBW=<val>(:<val>) iwpriv ra0 set ATECHANNEL=<val>(:<val>…)	iw set channel
iwpriv ra0 set ATETXCNT=<val>	tx_count
iwpriv ra0 set ATETXLEN=<val>	tx_length
iwpriv ra0 set ATETXANT=<val>	tx_antenna and tx_spe_idx

Jedi/Logan	MT76	
iwpriv ra0 set ATERXANT=<val>		
iwpriv ra0 set ATETXGI=<val>	tx_rate_sgi & tx_ltf	
iwpriv ra0 set ATETXMODE=<val>	tx_rate_mode	
iwpriv ra0 set ATETXMCS=<val>	tx_rate_idx	
iwpriv ra0 set ATEVHTNSS=<val> iwpriv ra0 set ATETXNSS=<val>	tx_rate_nss	
iwpriv ra0 set ATETXLDPC=<val>	tx_rate_ldpc	
iwpriv ra0 set ATETXSTBC=<val>	tx_rate_stbc	
iwpriv ra0 set ATETXPOW0=<val> iwpriv ra0 set ATETXPOW1=<val> iwpriv ra0 set ATETXPOW2=<val> iwpriv ra0 set ATETXPOW3=<val>	tx_power	
iwpriv ra0 set ATEPKTTXTIME =<val>	tx_time	
iwpriv ra0 set ATEIPG=<val>	tx_ipg	
iwpriv ra0 set ATEDUTYCYCLE=<val>	tx_duty_cycle	
iwpriv ra0 set ATETXFREQOFFSET=<val>	freq_offset	
iwpriv ra0 set ATEDA=xx:xx:xx:xx:xx:xx iwpriv ra0 set ATESA= xx:xx:xx:xx:xx:xx iwpriv ra0 set ATEBSSID= xx:xx:xx:xx:xx:xx	mac_addr	
iBF commands		
iwpriv ra0 set ATETxBfInit=<val>	txbf_act=init	txbf_param=<val>
iwpriv ra0 set ATETxBfGdInit=<val>	txbf_act=init	txbf_param=<val>
iwpriv ra0 set ATEIBFPhaseComp=<val>:<val>:..	txbf_act=phase_comp	txbf_param=<val>,...
iwpriv ra0 set ATEEBfProfileConfig=<val>:<val>:..	txbf_act=ebf_prof_update	txbf_param=<val>,...
iwpriv ra0 set ATEIBfProfileConfig=<val>:<val>:..	txbf_act=ibf_prof_update	txbf_param=<val>,...
iwpriv ra0 set TxBfTxApply=<val>:<val>:..	txbf_act=apply_tx	txbf_param=<val>,...
iwpriv ra0 set ATETxPacketWithBf=<val>:<val>:..	txbf_act=tx_prep	txbf_param=<val>,...
iwpriv ra0 set TxBfTxCmd=<val>:<val>:..	txbf_act=txcmd	txbf_param=<val>,...
iwpriv ra0 set TxBfProfileData20MAllWrite=<val>:<val>:..	txbf_act=prof_update_all	txbf_param=<val>,...
iwpriv ra0 set ATEIBfInstCal=<val>:<val>:..	txbf_act=phase_cal	txbf_param=<val>,...
iwpriv ra0 set ATEIBfGdCal=<val>:<val>:..		
iwpriv ra0 set ATEIBFPhaseVerify=<val>:<val>:...	txbf_act=phase_comp txbf_act=phase_cal	txbf_param=<val>,...
iwpriv ra0 set ATEIBFPhaseE2pUpdate=<val>:<val>:..	txbf_act=e2p_update atenl “eeprom ibf sync”	txbf_param=<val>,...
iwpriv ra0 set TriggerSounding=<val>:<val>:...	txbf_act=trigger_sounding	txbf_param=<val>,...
iwpriv ra0 set StopSounding=<val>	txbf_act=stop_sounding	txbf_param=<val>,...
iwpriv ra0 set TxBfProfileTagWrite=<val>	txbf_act=pfmu_tag_write	txbf_param=<val>,...
iwpriv ra0 set TxBfProfileTagRead=<val>:<val>	txbf_act=pfmu_tag_read	txbf_param=<val>,...
iwpriv ra0 set TxBfProfileTagInValid=<val>	txbf_act=set_invalid_prof	txbf_param=<val>,...
iwpriv ra0 set StaRecBfRead=<val>	txbf_act=sta_rec_read	txbf_param=<val>,...
eBF commands		
iwpriv ra0 set ATEConTxETxBfGdProc=<val>:...	This command is translated to a lot of mt76’s command. See eBF example.	
iwpriv ra0 set ATEConTxETxBfInitProc=<val>:...	This command is translated to a lot of mt76’s command. See eBF example.	
ZWDFS commands		
iwpriv ra0 set DfsRxCtrl=<chan>:<bw>	offchan_ch=<chan> offchan_bw=<bw>	

Jedi/Logan	MT76
iwpriv ra0 set DfsRxHist=<threshold>:<period>(:<antenna_idx>)	ipi_threshold=<threshold> ipi_period=<period> ipi_antenna_idx=< antenna_idx>

4.1.3 Statistic

Jedi/Logan	MT76
iwpriv ra0 set ATESHOW=1	dump dump stats
iwpriv ra0 set ATERXSTAT=<val>	
iwpriv ra0 set ResetCounter=1	Auto reset on tx_frame or rx_frame
iwpriv ra0 set ATERXSTATRESET=<val>	
iwpriv ra0 show wtbl=<wlan_idx>	echo <wlan_idx> > /sys/kernel/debug/ieee80211/phy0/mt76/wlan_idx cat /sys/kernel/debug/ieee80211/phy0/mt76/wtbl_info

4.1.4 MISC

Jedi/Logan	MT76
Read/Write EEPROM	
iwpriv ra0 e2p <offset>	Read EEPROM Data
iwpriv ra0 e2p <offset>=<val>	Change Value to Specific offset
Read/Write Control Register	
iwpriv ra0 mac <offset>	Section 7.7 Set/Dump Control Register in the MT76 Programming Guide
iwpriv ra0 mac <offset>=<val>	
Buffer Mode	
iwpriv ra0 set bufferMode=2 ※only support bufferMode=2	Update buffer mode

4.2 ated

Jedi/Logan	MT76
ated -i ra0 -c “sync eeprom all”	sync eeprom

5 Testmode Overall Status (Per-chip)

Chip		Test Item	Command Mode (iwpriv/mt76-test)	ATENL/HQADLL (Litepoint)
MT7915 A		Basic TX/RX	Support	Support
		iBF		
MT7916	2/5G 2T2R + 1R	Basic TX/RX	No RFB	
		iBF		
	2/5G 3T3R	Basic TX/RX	Support	Support
		iBF	Fail	
	2/6G 2T2R + 1R	Basic TX/RX	-	-
		iBF		
	2/6G 3T3R	Basic TX/RX	Support	Support
		iBF	Fail	
MT7981	2/5G 3T3R	Basic TX/RX	Support	Support
		iBF		
MT7986	MT7975	Basic TX/RX	Support	Support
		iBF		
	MT7976 2/5G	Basic TX/RX	Support	Support
		iBF		
	MT7976 2/5G	Basic TX/RX	Support	Support
		iBF		
MT7996	MT7976+MT7977+MT7977 *BE19000 eFEM*	Basic TX/RX	Support	T.B.D
		iBF		
	MT7975+MT7977+MT7977 *BE19000 iFEM*	Basic TX/RX	Support	T.B.D
		iBF		
	MT7976DA+MT7977 *BE14000 eFEM*	Basic TX/RX	Support	T.B.D
		iBF		
	MT7976C+MT7977 *BE14000 iFEM*	Basic TX/RX	Support	T.B.D
		iBF		
MT7992	MT7976G+MT7977 (2G 4T4R+ZWDFS, 5G 4T5R) *BE7200 eFEM*	Basic TX/RX	Support	T.B.D
		iBF		
	MT7975+MT7977 (2G 4T4R+ZWDFS, 5G 4T5R) *BE7200 2i5e*	Basic TX/RX	Support	T.B.D
		iBF		
	MT7975+MT7979 (2G 4T4R+ZWDFS, 5G 5T5R) *BE7200 iFEM*	Basic TX/RX	Support	T.B.D
		iBF		
	MT7976C+MT7977 (2G 2T2R, 5G 3T3R) *BE5040 eFEM*	Basic TX/RX	Support	T.B.D
		iBF		
	MT7976C+MT7977 (2G 2T2R, 5G 3T3R) *BE5040 iFEM*	Basic TX/RX	Support	T.B.D
		iBF		

6 Appendix

6.1 Spatial Extension Index Table

		Antenna Indexing				SW Valid Settings			
SPE index	Class	priority 1	priority 2	priority 3	priority 4	4T	3T	2T	1T
0	default	0	1	2	3	O	O	O	O
1	default	1	0	2	3	O	O	O	X
2	default	0	2	1	3	O	O	X	X
3	default	2	0	1	3	O	O	X	X
4	default	1	2	0	3	O	O	X	X
5	default	2	1	0	3	O	O	X	X
6	default	1	3	0	2	O	X	X	X
7	default	3	1	0	2	O	X	X	X
8	default	0	3	1	2	O	X	X	X
9	default	3	0	1	2	O	X	X	X
10	default	0	1	3	2	O	X	X	X
11	default	1	0	3	2	O	X	X	X
12	default	0	2	3	1	O	X	X	X
13	default	2	0	3	1	O	X	X	X
14	default	0	3	2	1	O	X	X	X
15	default	3	0	2	1	O	X	X	X
16	default	2	3	0	1	O	X	X	X
17	default	3	2	0	1	O	X	X	X
18	default	1	2	3	0	O	X	X	X
19	default	2	1	3	0	O	X	X	X
20	default	1	3	2	0	O	X	X	X
21	default	3	1	2	0	O	X	X	X
22	default	2	3	1	0	O	X	X	X
23	default	3	2	1	0	O	X	X	X
24	SE_SET1 from CR	*	*	*	*	O	O	O	X
25	SE_SET2 from CR	*	*	*	*	O	O	O	X
26	SE_SET3 from CR	*	*	*	*	O	O	O	X
27	SE_SET4 from CR	*	*	*	*	O	O	O	X
28	Reserved	*	*	*	*	X	X	X	X
29	Reserved	*	*	*	*	X	X	X	X
30	Reserved	*	*	*	*	X	X	X	X
31	Reserved	*	*	*	*	X	X	X	X

6.2 Beamforming Debug CR

The following tables record the most used CR for debugging or checking BF. Note that the CR presented here is mostly **read clear**.

Chip		Bfee RX NDP Count CR Address	Description (Unit: MPDU)
mt7915	Band 0	0x820EA044	[31:24] Number of TXBF feedback CQI Report count [23:16] Number of TXBF feedback aborted due to WTBL response control [15:8] Number of HE Trigger Frame BRP packet received [7:0] Number of NDP packet received
	Band 1	0x820FA044	
mt7916 mt7981 mt7986	Band 0	0x820ED7BC	[31:16] Number of TXBF feedback CQI report count [15:0] Number of NDP packet received
	Band 1	0x820FD7BC	
mt7996	Band 0	0x820ED9F0	[31:0] Number of NDP packet received
	Band 1	0x820FD9F0	
	Band 2	0x830ED9F0	
mt7992	Band 0	0x820EDAEC	
	Band 1	0x820FDAEC	

Chip		Bfee RX NDPA Count & TX Feedback Report CR Address	Description (Unit: MPDU)
mt7915	Band 0	0x820EA040	[31:16] Number of successfully transmitted TXBF feedback [15:0] Number of TXBF feedback triggered (NDPA Count)
	Band 1	0x820FA040	
mt7916 mt7981 mt7986	Band 0	0x820ED7B8	[31:16] Number of HE Trigger Frame BRP packet received [15:0] Number of TXBF feedback triggered (NDPA Count)
	Band 1	0x820FD7B8	
mt7996	Band 0	0x820ED9E8	[31:0] Number of TXBF feedback triggered (NDPA Count)
	Band 1	0x820FD9E8	
	Band 2	0x830ED9E8	
mt7992	Band 0	0x820EDAE4	
	Band 1	0x820FDAE4	

Chip		Bfer RX Feedback Count CR Address	Description (Unit: MPDU)
mt7915	Band 0	0x820EA0F8	[31:24] Number of Bfer received feedback for **Total** [23:16] Number of Bfer received feedback for **HE** [15:8] Number of Bfer received feedback for **VHT** [7:0] Number of Bfer received feedback for **HT**
	Band 1	0x820FA0F8	
mt7916 mt7981 mt7986	Band 0	0x820ED7B0, for **HT** & **VHT**	[31:16] Number of Bfer received feedback for **VHT** [15:0] Number of Bfer received feedback for **HT**
	Band 1	0x820FD7B0, for **HT** & **VHT**	
	Band 0	0x820ED7B4, for **HE**	[15:0] Number of Bfer received feedback for **HE**
	Band 1	0x820FD7B4, for **HE**	
mt7996	Band 0	0x820ED9D8 + <offs>	[31:0] Number of Bfer received feedback <offs>= $\begin{cases} 0x0, & \textbf{HT} \\ 0x4, & \textbf{VHT} \\ 0x8, & \textbf{HE} \\ 0xC, & \textbf{EHT} \end{cases}$
	Band 1	0x820FD9D8 + <offs>	
	Band 2	0x830ED9D8 + <offs>	
mt7992	Band 0	0x820EDAD4 + <offs>	
	Band 1	0x820FDAD4 + <offs>	

Chip		Bfer TXBF Applied Count CR Address	Description (Unit: PPDU)
mt7915	Band 0	0x820EA0F0	[31:16] Number of TX **iBF** applied count [15:0] Number of TX **eBF** applied count
	Band 1	0x820FA0F0	
mt7916 mt7981 mt7986	Band 0	0x820ED7A8	[31:16] Number of TX **iBF** applied count [15:0] Number of TX **eBF** applied count
	Band 1	0x820FD7A8	
mt7996	Band 0	0x820ED9CC	[31:0] Number of TX **eBF** applied count
	Band 1	0x820FD9CC	
	Band 2	0x830ED9CC	
	Band 0	0x820ED9D0	[31:0] Number of TX **iBF** applied count
	Band 1	0x820FD9D0	
	Band 2	0x830ED9D0	
mt7992	Band 0	0x820EDAC8	[31:0] Number of TX **eBF** applied count
	Band 1	0x820FDAC8	
	Band 0	0x820EDACC	[31:0] Number of TX **iBF** applied count
	Band 1	0x820FDACC	

6.3 Protected FT Field

The following table lists the protected FT fields in eFuse for each chip variant. The offsets listed below means their value **CANNOT be overwritten**.

Chip	eFuse Protected Region	
mt7996 BE19000 eFEM (7976, 7977, 7977)	**Ddie** [0x0, 0x3FF]	[0x10, 0x18F] [0x1B0, 0x2BF] [0x2C6, 0x2FF] [0x300, 0x30F] [0x311, 0x3FF]
	Adie 0 (MT7976) [0x400, 0x11FF]	[0x400, 0x47f] [0xB90, 0xB98] [0xB9A, 0xB9F] 0xBA6, 0xBA8, 0xBAA [0xBB0, 0xBBF]
	Adie 1 (MT7977) [0x1E00, 0x2A00]	[0x1E00, 0x1E6E] [0x1E70, 0x1E7C] [0x1E7E, 0x1F0F]
	Adie 2 (MT7977) [0x1200, 0x1E00]	[0x1200, 0x126E] [0x1270, 0x127C] [0x127E, 0x130F]

Chip	eFuse Protected Region	
mt7996 BE19000 iFEM (7975, 7977, 7977)	**Ddie** [0x0, 0x3FF]	Same as mt7996 BE19000 eFEM Ddie
	Adie 0 (MT7976) [0x400, 0x11FF]	[0x9C0, 0xA2F] [0xAC0, 0xAFF] 0xBA1, 0xBA9 [0xBB0, 0xBBF]
	Adie 1 (MT7977) [0x1E00, 0x2A00]	Same as mt7996 BE19000 eFEM Adie1
	Adie 2 (MT7977) [0x1200, 0x1E00]	Same as mt7996 BE19000 eFEM Adie 2
mt7996 BE14000 2adie eFEM/iFEM (7976, 7977)	**Ddie** [0x0, 0x3FF]	Same as mt7996 BE19000 eFEM Ddie
	Adie 0 (MT7976) [0x400, 0x11FF]	Same as mt7996 BE19000 eFEM Adie 0
	Adie 1 (MT7977) [0x1200, 0x1E00]	Same as mt7996 BE19000 eFEM Adie 2
mt7992	**Ddie** [0x0, 0x3FF]	[0x10, 0x18f] [0x1b0, 0x1e00]
	Adie 0, 1	**All protected**

6.4 Abbreviations

Abbreviation	Full Form
ADC	Analog-to-Digital Converter
ADCDCOC	Analog-to-Digital Converter Direct Current Offset Correction
Adie	Analog Die
AID	Association Identifier
ANT	Antenna
AP	Access Point
ATE	Automated Test Equipment
BF	Beamforming
Bfee	Beamformee
Bfer	Beamformer
BMC	Broadcast and Multicast
BRP	Beam Refinement Process
BSSID	Basic Service Set Identifier
BW	Bandwidth
Cal	Calibration
CBW	Channel Bandwidth
CCA	Clear Channel Assessment
CCK	Complementary Code Keying
CQI	Channel Quality Indicator
CR	Control Register
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CW	Continuous Wave
DA	Destination Address
dB	Decibel

Abbreviation	Full Form
DBDC	Dual Band Dual Concurrent
DBW	Data Bandwidth
DC	Direct Current
DCIQ	Direct Current In-phase and Quadrature
DCOC	Direct Current Offset Correction
Ddie	Digital Die
DFS	Dynamic Frequency Selection
DPD	Digital Pre-Distortion
DTS	Device Tree Source
DUT	Device Under Test
eBF	Explicit Beamforming
EEPROM/E2P	Electrically Erasable Programmable Read-Only Memory
eFEM	External Front-End Module
eFuse	Electronic Fuse
EHT	Extremely High Throughput
eLNA	External Low Noise Amplifier
eMMC	Embedded Multimedia Card
ePA	External Power Amplifier
FDIQ	Frequency Domain In-phase and Quadrature
FIIQ	Frequency Independent In-phase and Quadrature
FT	Final Test
FW	Firmware
GI	Guard Interval
GT	Gain Table
HE	High Efficiency
HQADLL	Hardware Quality Assurance Dynamic-link Library
HT	High Throughput
IB RSSI	Intermediate Frequency/Baseband Received Signal Strength Indicator
iBF	Implicit Beamforming
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
iFEM	Internal Front-End Module
iLNA	Internal Low Noise Amplifier
iPA	Internal Power Amplifier
IPG	Inter-Packet Gap
IPI	Idle Power Indicator
JP	Japan
LDPC	Low-Density Parity-Check
LNA	Low Noise Amplifier
LPFG	Low-Pass Filter Gain
LTF	Long Training Field
MAC	Media Access Control
MCS	Modulation and Coding Scheme
MHz	Megahertz
MPDU	MAC Protocol Data Unit
MTD	Memory Technology Device
MU	Multi-User

Abbreviation	Full Form
NDP	Null Data Packet
NDPA	Null Data Packet Announcement
NSS	Number of Spatial Streams
OFDM	Orthogonal Frequency Division Multiplexing
OTA	Over-The-Air
PD	Packet Detection
PFMU	Profile Management Unit
PHY	Physical Layer
PPDU	Physical Protocol Data Unit
Pre-cal	Pre-calibration
RAM	Random Access Memory
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
RU	Resource Unit
RX	Receiver/Receive
SA	Source Address
SDK	Software Development Kit
SIFS	Short Interframe Space
SKU	Stock Keeping Unit
SPE	Spatial Extension
SS	Spatial Streams
STA	Station
STBC	Space-Time Block Coding
SU	Single User
SW	Software
Sync	Synchronization/Synchronize
TB	Trigger-based
TBTC	Triple-Band Triple Concurrent
TSSI	Transmit Signal Strength Indicator
TX	Transmitter/Transmit
TXBF	Transmit Beamforming
TXCMD	TX command
TXD	Transmit Descriptor
VHT	Very High Throughput
WA (-CPU)	Wi-Fi Acceleration/Application CPU
WCID	Wireless Client Identifier
Wi-Fi	Wireless Fidelity
Wiphy	Wireless Physical Layer
WLAN	Wireless Local Area Network
WM (-CPU)	Wi-Fi MAC CPU
WTBL	Wireless Lan Table
ZWDFS	Zero Wait Dynamic Frequency Selection

Exhibit 1 Terms and Conditions

Your access to and use of this document and the information contained herein (collectively this “Document”) is subject to your (including the corporation or other legal entity you represent, collectively “You”) acceptance of the terms and conditions set forth below (“T&C”). By using, accessing or downloading this Document, You are accepting the T&C and agree to be bound by the T&C. If You don’t agree to the T&C, You may not use this Document and shall immediately destroy any copy thereof.

This Document contains information that is confidential and proprietary to MediaTek Inc. and/or its affiliates (collectively “MediaTek”) or its licensors and is provided solely for Your internal use with MediaTek’s chipset(s) described in this Document and shall not be used for any other purposes (including but not limited to identifying or providing evidence to support any potential patent infringement claim against MediaTek or any of MediaTek’s suppliers and/or direct or indirect customers). Unauthorized use or disclosure of the information contained herein is prohibited. You agree to indemnify MediaTek for any loss or damages suffered by MediaTek for Your unauthorized use or disclosure of this Document, in whole or in part.

MediaTek and its licensors retain titles and all ownership rights in and to this Document and no license (express or implied, by estoppels or otherwise) to any intellectual propriety rights is granted hereunder. This Document is subject to change without further notification. MediaTek does not assume any responsibility arising out of or in connection with any use of, or reliance on, this Document, and specifically disclaims any and all liability, including, without limitation, consequential or incidental damages.

THIS DOCUMENT AND ANY OTHER MATERIALS OR TECHNICAL SUPPORT PROVIDED BY MEDIATEK IN CONNECTION WITH THIS DOCUMENT, IF ANY, ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. MEDIATEK SPECIFICALLY DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, COMPLETENESS OR ACCURACY AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MEDIATEK SHALL NOT BE RESPONSIBLE FOR ANY MEDIATEK DELIVERABLES MADE TO MEET YOUR SPECIFICATIONS OR TO CONFORM TO A PARTICULAR STANDARD OR OPEN FORUM.

Without limiting the generality of the foregoing, MediaTek makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does MediaTek assume any liability arising out of the application or use of any product, circuit or software. You agree that You are solely responsible for the designing, validating and testing Your product incorporating MediaTek’s product and ensure such product meets applicable standards and any safety, security or other requirements.

The above T&C and all acts in connection with the T&C or this Document shall be governed, construed and interpreted in accordance with the laws of Taiwan, without giving effect to the principles of conflicts of law.