GigaDevice Semiconductor Inc.

GD32VW553 Basic Commands User Guide

Application Note
AN153

Revision 1.0

(Nov.2023)

Table of Contents

List of Figures

List of Tables

1. Basic user commands

Connect the test machine to the development board through a USB cable. Open the UART tool, and connect it to the correct COM port. After powering on and starting the development board correctly, issue a command through the UART tool, and the development board can complete the corresponding operation according to the command content.

In this manual, < > after the command indicates that the option is required, and [ ] indicates that the option is optional. Note that the commands are strictly case-sensitive.

1.1. help

The command has no option.

As shown in ***Figure 1-1. help command***, the help command lists all commands supported by the development board.

Note: View BLE related commands through the ble_help command.

Figure 1-1. help command

```
# help
==============================
      ble_help
==============================
      help
      reboot
      tasks
      free
      cpu_stats
      sys_ps
      ping
      join_group
      iperf
      iperf3
      wifi_debug
      wifi_open
      wifi_close
      wifi_mac_addr
      wifi_concurrent
      wifi_auto_conn
      wifi_scan
      wifi_connect
      wifi_connect_bssid
      wifi_disconnect
      wifi_status
      wifi_set_ip
      wifi_ps
      wifi_monitor
      wifi_ap
      wifi_ap_adv
      wifi_stop_ap
#
```

1.2. reboot

The command has no option.

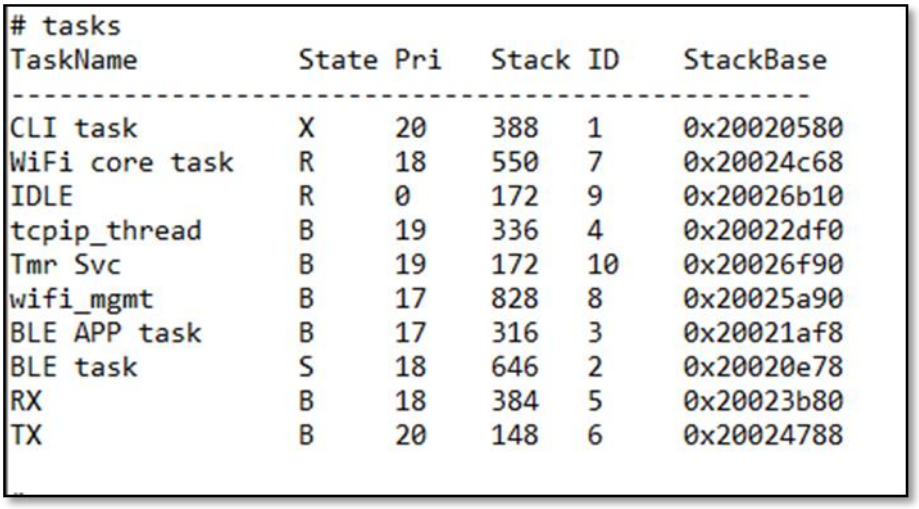
After the command is executed, the development board will restart and the serial port will print the start information. The command has a similar function as that of the reset button.

1.3. tasks

The command has no option.

After the command is executed, task-related information, including the status, priority, remaining minimum stack space for the task since task creation, task No., and the base address of the stack used by the task, will be printed, as shown in ***Figure 1-2. tasks command***.

Figure 1-2. tasks command



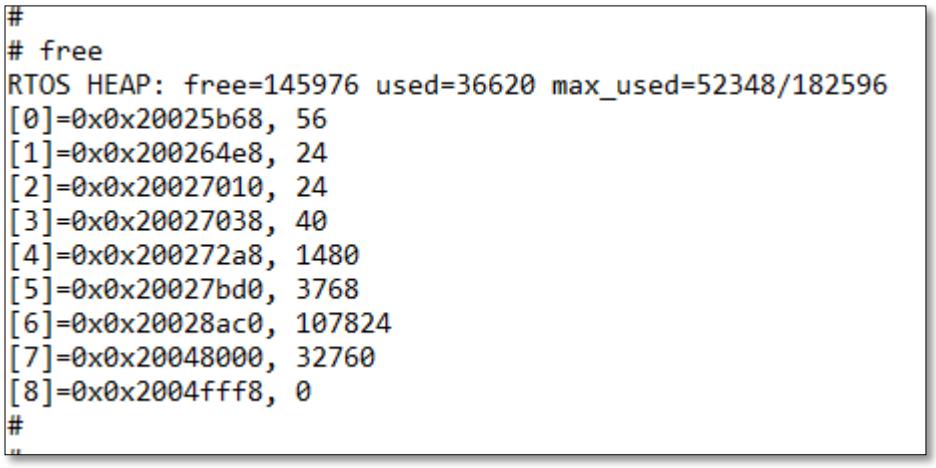
```
# tasks
TaskName          State Pri   Stack ID    StackBase
-----------------------------------------------
CLI task          X     20    388   1     0x20020580
WiFi core task    R     18    550   7     0x20024c68
IDLE              R     0     172   9     0x20026b10
tcpip_thread      B     19    336   4     0x20022df0
Tmr Svc           B     19    172   10    0x20026f90
wifi_mgmt         B     17    828   8     0x20025a90
BLE APP task      B     17    316   3     0x20021af8
BLE task          S     18    646   2     0x20020e78
RX                B     18    384   5     0x20023b80
TX                B     20    148   6     0x20024788
```

1.4. free

The command has no option.

After the command is executed, heap related information, including remaining heap, used heap, maximum used heap, maximum available heap, and the address and size of each available mem block, will be printed, as shown in ***Figure 1-3. free command***.

Figure 1-3. free command



1.5. sys_ps

Figure 1-4. sys_ps command

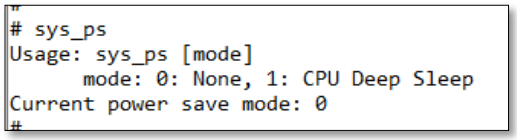


Figure 1-4. sys_ps command shows how to use the command. There are three modes:

Leave blank: No settings. Only print the current CPU power save mode;

0: Disable CPU power save.

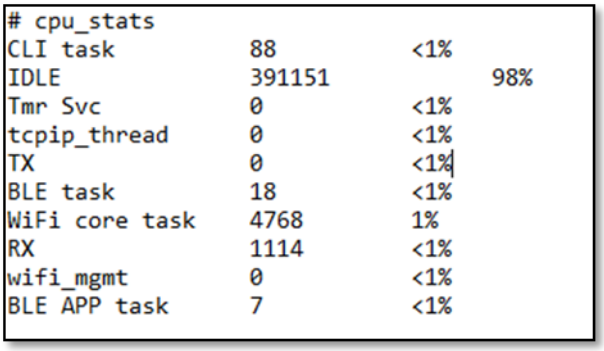
1: Enable CPU power save. The mode is deep sleep. When the CPU is idle, it automatically enters the deep sleep mode, and then it can be automatically woken up by wifi/ble or actively woken up by a uart rx event.

1.6. cpu_stats

The command has no option.

After this command is executed, the CPU usage of each task will be printed, including running time. as shown in Figure 1-5. cpu_stats command.

Figure 1-5. cpu_stats command



1.7. Wi-Fi

This section introduces Wi-Fi related commands.

1.7.1. wifi_open

The command has no option.

This command is used to enable WiFi functions. Other WiFi-related commands can be executed provided that WiFi is enabled. After the development board is started correctly, WiFi is enabled by default, so this command is unnecessarily executed to repeatedly enable WiFi. This command generally works with wifi_close by enabling WiFi after it is turned off by wifi_close. If WiFi is enabled, the serial port will give a prompt.

1.7.2. wifi_close

The command has no option.

wifi_close can turn off WiFi, but in this case, some commands, such as wifi_scan and wifi_connect, can not be executed.

The command has different execution results for different conditions of the development board:

- If the development board is connected to AP, the command will disconnect them and then turn off WiFi.
- If the development board is not connected to AP, the command will directly turn off WiFi.
- If the development board in softAP mode is connected to sta, the command will disconnect them and then turn off WiFi.
- If the development board in softAP mode is not connected to sta, the command will directly turn off WiFi.
- If WiFi is turned off, the serial port will give a prompt.

1.7.3. wifi_debug

- Usage: wifi_debug <0 or 1>

This command is used to control the printing of WiFi debug logs. 0 means printing is disabled, while 1 means printing is enabled.

1.7.4. wifi_scan

The command has no option. The development board cannot be in softap mode when the command is executed.

After the command is executed, AP information scanned by the development board, including RSSI, channel, BSSID, SSID, and encryption method, will be printed, as shown in ***Figure 1-6. wifi_scan command***.

Figure 1-6. wifi_scan command

```
# wifi_scan
# WIFI_SCAN: done
[0] (-34 dBm) CH=  1 BSSID=c4:70:ab:d9:bd:11 SSID=OpenWrt [OPEN]
[1] (-30 dBm) CH=  1 BSSID=1c:5f:2b:fd:be:60 SSID=D-Link_DIR-822  [RSN:WPA-PSK CCMP/CCMP]
[2] (-42 dBm) CH=  1 BSSID=86:e5:81:9b:d4:05 SSID=fly  [RSN:WPA-PSK CCMP/CCMP]
[3] (-47 dBm) CH=  1 BSSID=ba:fa:07:50:63:f6 SSID=Redmi K40  [RSN:WPA-PSK CCMP/CCMP]
[4] (-50 dBm) CH=  1 BSSID=08:3a:38:cc:2f:d0 SSID=GD-internet [OPEN]
[5] (-50 dBm) CH=  1 BSSID=08:3a:38:cc:2f:d1 SSID=GD-guest [OPEN]
[6] (-50 dBm) CH=  1 BSSID=08:3a:38:cc:2f:d2 SSID=GD-lan [OPEN]
[7] (-32 dBm) CH=  6 BSSID=88:c3:97:0d:c3:70 SSID=xiaomi_4a [OPEN]
[8] (-23 dBm) CH=  4 BSSID=68:77:24:bd:86:59 SSID=TP-LINK_8659  [RSN:WPA-PSK,SAE CCMP/CCMP][MFP:AES-128-CMAC]
[9] (-22 dBm) CH=  4 BSSID=72:77:24:bd:86:59 SSID=  [RSN:WPA-PSK CCMP/CCMP]
[10] (-22 dBm) CH=  5 BSSID=a2:aa:95:39:57:72 SSID=HUAWEI_AX3000  [RSN:WPA-PSK,SAE CCMP/CCMP][MFP:AES-128-CMAC]
[11] (-23 dBm) CH=  6 BSSID=60:3a:7c:26:f3:a0 SSID=tplink_8690 [OPEN]
[12] (-48 dBm) CH=  6 BSSID=08:3a:38:cc:2d:f1 SSID=GD-guest [OPEN]
[13] (-48 dBm) CH=  6 BSSID=08:3a:38:cc:2d:f2 SSID=GD-lan [OPEN]
[14] (-47 dBm) CH=  6 BSSID=08:3a:38:cc:2d:f0 SSID=GD-internet [OPEN]
[15] (-49 dBm) CH=  6 BSSID=0e:cc:cb:36:80:24 SSID=WuMingming  [RSN:WPA-PSK CCMP/CCMP]
[16] (-42 dBm) CH=  6 BSSID=ee:cb:9d:ce:33:ad SSID=yzq  [RSN:WPA-PSK,SAE CCMP/CCMP][MFP:AES-128-CMAC]
[17] (-41 dBm) CH=  6 BSSID=00:22:6b:60:0a:98 SSID=cisco  [RSN:WPA-PSK CCMP/CCMP]
[18] (-45 dBm) CH=  6 BSSID=82:8c:b8:9f:24:8b SSID=wlan_test  [RSN:WPA-PSK CCMP/CCMP]
[19] (-72 dBm) CH=  6 BSSID=08:3a:38:cc:0f:12 SSID=GD-lan [OPEN]
[20] (-55 dBm) CH= 11 BSSID=d6:4f:86:cb:c8:d0 SSID=iQOO Neo5  [RSN:WPA-PSK CCMP/CCMP]
[21] (-42 dBm) CH=  9 BSSID=50:eb:f6:06:8a:18 SSID=RT-AX56U [OPEN]
[22] (-69 dBm) CH= 11 BSSID=08:3a:38:cc:27:71 SSID=GD-guest [OPEN]
[23] (-22 dBm) CH= 11 BSSID=8c:53:c3:d8:0d:fd SSID=xiaomi_wifi6  [RSN:WPA-PSK CCMP/CCMP]
[24] (-69 dBm) CH= 11 BSSID=08:3a:38:cc:27:70 SSID=GD-internet [OPEN]
[25] (-69 dBm) CH= 11 BSSID=08:3a:38:cc:27:72 SSID=GD-lan [OPEN]
```

1.7.5. wifi_concurrent

- Usage: wifi_concurrent [0 or 1]

This command is used to control enabling of wifi concurrent mode. 0 means the mode is disabled, while 1 means the mode is enabled. When this option is not set, the current enabled state is printed only.

This command can not be executed until the macro CFG_WIFI_CONCURRENT in MSDK\macsw\export\ wlan_config.h file is opened.

1.7.6. wifi_connect

- Usage: wifi_connect <SSID> [PASSWORD]

The command is used to connect to an AP. The development board cannot be in softap mode when the command is executed.

- wifi_connect <SSID>

It is used to connect to an unencrypted AP.

- wifi_connect <SSID> <PASSWORD>

It is used to connect to an encrypted AP.

The connection process is as shown in ***Figure 1-7. Wifi_connect command***, and the serial port prints out the connection process information. If the wifi_connect command is executed after the AP is connected, the development board will first disconnect from the original AP and then connect to the new AP.

Figure 1-7. Wifi_connect command

```
# wifi_connect xiaomi_4a
[0] (-34 dBm) CH=  6 BSSID=88:c3:97:0d:c3:70 SSID=xiaomi_4a [OPEN]
MAC: auth req send
MAC: auth rsp received, status = 0
MAC: assoc req send
MAC: assoc rsp received, status = 0
WIFI_MGMT: DHCP got ip 192.168.3.127
#
# wifi_connect TP-LINK_8659 12345678
MAC: deauth send
[0] (-22 dBm) CH=  4 BSSID=68:77:24:bd:86:59 SSID=TP-LINK_8659  [RSN:WPA-PSK,SAE CCMP/CCMP][MFP:AES-128-CMAC]
SAE: commit send
SAE: commit received
SAE: confirm send, status_code = 0
SAE: confirm received, status_code = 0
MAC: assoc req send
MAC: assoc rsp received, status = 0
WPA: 4-1 received
WPA: 4-2 send
WPA: 4-3 received
WPA: 4-4 send
WIFI_MGMT: DHCP got ip 192.168.1.100
#
```

1.7.7. wifi_connect_bssid

- Usage: wifi_connect_bssid <BSSID> [PASSWORD]

This command is similar to the wifi_connect command, executed by using the same method. The only difference is that SSID in the option is modified to BSSID.

1.7.8. wifi_disconnect

The command has no option.

After the command is executed, the development board will disconnect from the AP. If the execution is successful, the serial port will print the following information:

MAC: deauth send

MGMT: disconnect complete

1.7.9. wifi_auto_conn

- Usage: wifi_auto_conn [0 or 1]

This command is used to set automatic connection to AP upon startup or not. 0 means no automatic connection, while 1 means automatic connection. When this option is not set, the current settings are printed only.

If automatic connection is set, when AP is successfully connected again, AP information will be saved in flash; however, if AP is repeatedly connected, the AP successfully connected last will be recoded as valid AP. Rebooted development board will be automatically connected to AP according to AP information in flash. If AP is not connected after automatic connection is set, rebooted development board will not be automatically connected to AP.

1.7.10. wifi_status

The command has no option.

After the command is executed, the serial port will print the Wi-Fi status of the current development board.

Wi-Fi currently has three modes: SoftAP, MONITOR, and STATION. The information printed by the command in different modes is different, as shown in ***Figure 1-8. wifi status command***.

Figure 1-8. wifi_status command

```
# wifi_status
WIFI Status:
============================
WiFi VIF[0]: 76:ba:ed:71:09:10
SoftAP
       Status: Started
       SSID: ap_test
       Channel: 6
       Security: WPA2
       IP: 192.168.237.1
       Client[0]:   76:ba:ed:ff:ff:02   192.168.237.150

# wifi_status
WIFI Status:
============================
WiFi VIF[0]: 76:ba:ed:71:09:10
Monitor

# wifi_status
WIFI Status:
============================
WiFi VIF[0]: 76:ba:ed:71:09:10
STA
       Status: Connected
       SSID: TP-LINK_8659
       BSSID: 68:77:24:bd:86:59
       Channel: 4
       Bandwidth: 0
       Security: WPA3
       RSSI: -22
       IP: 192.168.1.100

# wifi_status
WIFI Status:
============================
WiFi VIF[0]: 76:ba:ed:71:09:10
STA
       Status: Disconnected
```

This first line shows MAC address of the current Wi-Fi device; the second line shows the mode of the current Wi-Fi device, one of the above three modes.

In AP mode, the status, SSID, channel, encryption method, and IP address will be displayed. If any devices are connected to this AP, the information of these devices, including the MAC address and IP address, will also be displayed, and multiple devices will be sorted in sequence.

In STATION mode, WIFI Status indicates whether the current Wi-Fi device is connected to the AP; Connected means connected, and Disconnected means disconnected. If it is connected, the SSID, BSSID, and channel of the AP will be displayed.

1.7.11. wifi_monitor

- Usage: wifi_monitor stop | start <channel>

Figure 1-9. wifi_monitor command shows how to use the command. The wifi_monitor start <channel> command is used to start the monitor mode, and the monitoring channel needs to be specified; the wifi_monitor stop command is used to close the monitor mode and switch to the station mode.

Figure 1-9. wifi_monitor command

```
#
# wifi_monitor
Usage: wifi_monitor stop | start <channel>
start: start the monitor mode.
<channel>: 1~14.
stop: stop the monitor mode.
#
```

1.7.12. wifi_ps

- Usage: wifi_ps <mode>

Figure 1-10. wifi_ps command

```
# wifi_ps
Usage: wifi_ps <mode>
      mode: 0: off, 1: always on, 2: dynamic on
#
```

Figure 1-10. wifi_ps command shows how to use the command. There are three modes:

0: Disable power save;

1: Enable power save. The mode is Normal mode, and the Wi-Fi module will always be in power save mode;

2: Enable power save. The mode is Dynamic mode, and the Wi-Fi module will decide whether to enter or exit the power save mode based on the wifi TX/RX traffic;

1.7.13. wifi_ap

- Usage: wifi_ap <ssid> <password> <channel> [-a <akm>[,<akm 2>]] [-hide <hide_ap>]

The command is used to enable or disable the softap mode. ***Figure 1-11. wifi_ap command*** shows how to use the command.

Figure 1-11. wifi_ap command

```
#
# wifi_ap
Usage: wifi_ap <ssid> <password> <channel> [-a <akm>[,<akm 2>]] [-hide <hide_ap>]
<ssid>: The length should be between 1 and 32.
<password>: The length should be between 8 and 63, but can be "NULL" indicates open ap.
<channel>: 1~13.
[-a <akm>[,<akm 2>]]: only support following 5 AKM units: open; wpa2; wpa3; wpa2,wpa3 or wpa3,wpa2,
default wpa2.
[-hide <hide_ap>]: 0 means broadcast ssid or 1 means hidden ap, default 0.
for example:
    wifi_ap test_ap NULL 1 -a open -hide 0, means an open ap in channel 1 and can broadcast ssid.
    wifi_ap test_ap 12345678 1, means an WPA2 ap in channel 1.
#
```

ssid does not support Chinese characters. When "NULL" is filled in the password, it means that an open AP is enabled, and the -a configuration will be ignored. In addition, if an encrypted AP is enabled and the -a option is not configured to specify the encryption method, the default is wpa2 encryption.

1.7.14. wifi_ap_adv

This command is executed by using the same method as that for the wifi_ap command.

1.7.15. wifi_stop_ap

The command has no option. After the command is executed, the Softap mode will stop and switch to the station mode. If the execution is successful, the serial port will print the following information:

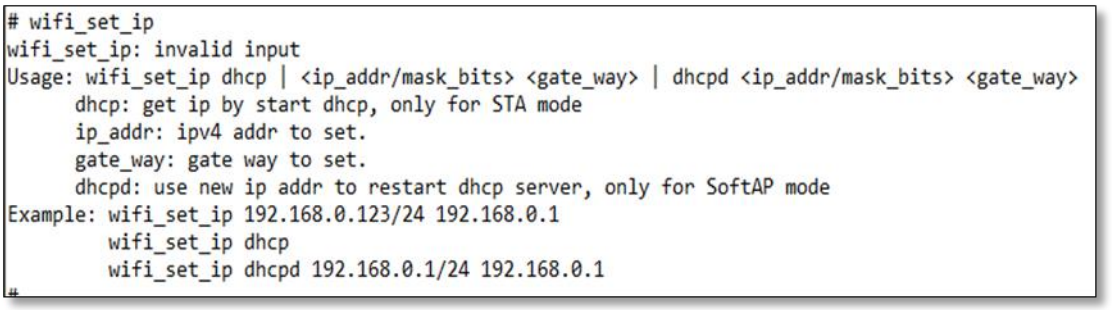
SoftAP successfully stoped!

1.7.16. wifi_set_ip

- Usage: wifi_set_ip dhcp |<ip_addr> <gate_way>

The command is used to manually set a static IP address or automatically get an IP address through DHCP in station mode. It can also set the IP address and gateway in softap mode. ***Figure 1-12. wifi_set_ip command*** shows how to use the command.

Figure 1-12. wifi_set_ip command



1.7.17. wifi_mac_addr

- Usage: wifi_mac_addr [xx:xx:xx:xx:xx:xx]

This command is used to set temporary mac address of WiFi, which will become invalid after reboot or power-down reboot.

When this option is not set, the current mac address is printed only.

1.8. ping

- Usage: ping <target_ip | stop> [-n count] [-l size] [-i interval] [-t total time]

The command is used to perform the ping test.

The target_ip is a peer address. IPv4 is in a format of <ipv4_addr>, while IPv6 is in a format of <-6 ipv6_addr> (provided that Ipv6 is enabled).

In the parameters of the command, count indicates the number of ping packets; size indicates the packet length, in bytes; interval indicates the packet sending interval, in milliseconds; total time indicates the total running time, in seconds. By default, count is 5, size is 120, interval is 10, and total time is not used; if the total time option is used, the count and interval options do not work; interval defaults to 1000 ms, and count will be equal to the total time value.

Figure 1-13. ping command shows how to use the ping command.

Figure 1-13. ping command

```
16:04:22.596  # ping 192.168.1.1
16:04:22.599  # [ping_test] PING 192.168.1.1 120 bytes of data
16:04:22.647  [ping_test] 120 bytes from 192.168.1.1: icmp_seq=1 time=19 ms
16:04:22.648  [ping_test] 120 bytes from 192.168.1.1: icmp_seq=2 time=1 ms
16:04:22.649  [ping_test] 120 bytes from 192.168.1.1: icmp_seq=3 time=2 ms
16:04:22.698  [ping_test] 120 bytes from 192.168.1.1: icmp_seq=4 time=4 ms
16:04:22.700  [ping_test] 120 bytes from 192.168.1.1: icmp_seq=5 time=1 ms
16:04:22.702  [ping_test] 5 packets transmitted, 5 received, 0% packet loss
16:04:22.703  [ping_test] delay: min 1 ms, max 19 ms, avg 5 ms
16:04:23.769
16:04:31.693  # ping 192.168.1.1 -n 3
16:04:31.694  # [ping_test] PING 192.168.1.1 120 bytes of data
16:04:31.697  [ping_test] 120 bytes from 192.168.1.1: icmp_seq=1 time=1 ms
16:04:31.698  [ping_test] 120 bytes from 192.168.1.1: icmp_seq=2 time=1 ms
16:04:31.702  [ping_test] 120 bytes from 192.168.1.1: icmp_seq=3 time=1 ms
16:04:31.742  [ping_test] 3 packets transmitted, 3 received, 0% packet loss
16:04:31.743  [ping_test] delay: min 1 ms, max 1 ms, avg 1 ms
16:04:32.457
16:04:39.214  # ping 192.168.1.1 -n 3 -l 1000
16:04:39.217  # [ping_test] PING 192.168.1.1 1000 bytes of data
16:04:39.218  [ping_test] 1000 bytes from 192.168.1.1: icmp_seq=1 time=1 ms
16:04:39.265  [ping_test] 1000 bytes from 192.168.1.1: icmp_seq=2 time=1 ms
16:04:39.266  [ping_test] 1000 bytes from 192.168.1.1: icmp_seq=3 time=1 ms
16:04:39.270  [ping_test] 3 packets transmitted, 3 received, 0% packet loss
16:04:39.272  [ping_test] delay: min 1 ms, max 1 ms, avg 1 ms
16:04:39.826
16:05:02.193  # ping 192.168.1.1 -n 3 -l 500 -i 5000
16:05:02.194  # [ping_test] PING 192.168.1.1 500 bytes of data
16:05:02.196  [ping_test] 500 bytes from 192.168.1.1: icmp_seq=1 time=1 ms
16:05:07.231  [ping_test] 500 bytes from 192.168.1.1: icmp_seq=2 time=6 ms
16:05:12.209  [ping_test] 500 bytes from 192.168.1.1: icmp_seq=3 time=3 ms
16:05:12.211  [ping_test] 3 packets transmitted, 3 received, 0% packet loss
16:05:12.215  [ping_test] delay: min 1 ms, max 6 ms, avg 3 ms
16:05:15.208
16:11:03.842  # ping 192.168.1.1 -n 3 -l 500 -i 5000 -t 5
16:11:03.844  # [ping_test] PING 192.168.1.1 500 bytes of data
16:11:03.845  [ping_test] 500 bytes from 192.168.1.1: icmp_seq=1 time=8 ms
16:11:04.859  [ping_test] 500 bytes from 192.168.1.1: icmp_seq=2 time=3 ms
16:11:05.876  [ping_test] 500 bytes from 192.168.1.1: icmp_seq=3 time=1 ms
16:11:06.843  [ping_test] 500 bytes from 192.168.1.1: icmp_seq=4 time=1 ms
16:11:07.860  [ping_test] 500 bytes from 192.168.1.1: icmp_seq=5 time=1 ms
16:11:07.861  [ping_test] 5 packets transmitted, 5 received, 0% packet loss
16:11:07.867  [ping_test] delay: min 1 ms, max 8 ms, avg 2 ms
```

- ping stop

This command is used to stop the ping test, as shown in ***Figure 1-14. ping stop command***,

Figure 1-14. ping stop command

```
# ping 192.168.1.1 -n 3 -l 500 -i 5000 -t 50
# [ping_test] PING 192.168.1.1 500 bytes of data
[ping_test] 500 bytes from 192.168.1.1: icmp_seq=1 time=1 ms
[ping_test] 500 bytes from 192.168.1.1: icmp_seq=2 time=1 ms
[ping_test] 500 bytes from 192.168.1.1: icmp_seq=3 time=1 ms
[ping_test] 500 bytes from 192.168.1.1: icmp_seq=4 time=1 ms
ping stop
# [ping_test] 4 packets transmitted, 4 received, 0% packet loss
[ping_test] delay: min 1 ms, max 1 ms, avg 1 ms
```

1.9. join_group

- Usage: join_group <group ip eg:224.0.0.5>

The development board must be connected to the AP before the command is executed. After the command is executed, the development board will join a multicast group, such as:

- join_group 224.0.0.5

During this period, sniffer can be used to capture the IGMP protocol packet sent by the development board after the command is executed.

1.10. iperf3

The iperf3 command uses iperf3 for network speed test.

1.10.1. iperf3 -h

As shown in ***Figure 1-15. iperf3 -h command***, the serial port will print out the options related to the iperf3 command.

Figure 1-15. iperf3 -h command

```
#
# iperf3 -h
Usage:
    iperf3 <-s|-c hostip|stop|-h> [options]
Server or Client:
    -i #         seconds between periodic bandwidth reports
    -p #         server port to listen on/connect to
Server specific:
    -s           run in server mode
Client specific:
    -c <host>    run in client mode, connecting to <host>
    -u           use UDP rather than TCP
    -b #[KMG][/#] target bandwidth in bits/sec (0 for unlimited)
                 (default 1 Mbit/sec for UDP, unlimited for TCP)
                 (optional slash and packet count for burst mode)
    -t #         time in seconds to transmit for (default 10 secs)
    -l #[KMG]    length of buffer to read or write
    -S #         set the IP 'type of service'
#
```

1.10.2. iperf3 -s [options]

- iperf3 -s

Enable an iperf3 server to listen to TCP/UDP data on port 5201 by default. Other options are default values.

- -p <port>

Set the port that the server listens on. The port range is 0-65535, and the default is 5201.

For example: iperf3 -s -p 5003

The server listens on port 5003.

- -i <interval>

Set the period of the test results printed by the serial port (Interval column), in seconds, with a range of 0.1-60 and 0. When it is set to 0, it means that no periodic report is printed and only the final test results are output. The default is 4.

For example: iperf3 -s -i 0.5,

The period of the test results printed by the serial port is 0.5 s.

1.10.3. iperf3 -c <host> [options]

- iperf3 -c <host>

Enable an iperf3 client, and make a TCP connection with the server whose IP address is <host> on the default port 5201. All other options are default values.

- -u

Enable an iperf client, and make a UDP connection with the server whose IP address is <host> on the default port 5201. The -u option is usually used with the -b option to specify the data bandwidth to be sent.

- -p <port>

Set the client connection port, which must be the same as the port that the server listens on.

- -i <interval>

The -i option settings are the same as those on the server.

- -b <bandwidth/number>

The unit of bandwidth is bits/sec and the format is data[KMG]. For example, 50K, 50k or 50000 means that the bandwidth is set to 50 Kbits/sec; when bandwidth is 0, it means that there is no limit. The default in UDP mode is 1 Mbit/sec, and there is no limit for tcp connection.

When "/number" is not added after bandwidth, iperf3 will calculate the number of data packets that need to be sent per second to reach the specified bandwidth based on the length of each data packet, and then send each data packet at an average interval.

For example: iperf3 -c 192.168.3.132 -u -b 200k

When "/number" is added after bandwidth, the system enters the burst mode, and iperf3 will continuously send the specified number of data packets at one time without interval, but there is an even interval between the batches.

For example: iperf3 -c 192.168.3.132 -u -b 200k/60

- -t <time>

Set the data transmission time, in seconds. The default value is 10.

- -l <length>

Set the length of the read and write buffer, in bytes; the format is: data[KMG], the same as the -n option. It is recommended to set this value to 1472 in UDP mode and 1460 in TCP mode.

- -S <QOS value>

Set the QOS service type of the outstack packet. The number range is 0-255, and hexadecimal (0x prefix), octal (0 prefix), and decimal can be used, such as 0x16 == 026 == 22.

1.10.4. iperf3 stop

The command is used to stop the iperf3 test.

1.10.5. iperf3 test example

- Connect the development board and test machine to the same AP, and then view the IP address of the development board.
  - Use the wifi_connect command to connect the development board to the AP, and use the wifi_status command to view the IP address.
- The test machine opens the iperf3 command window and starts the test.
  - The server first executes the command: iperf3 -s -p <port> -i <interval>
  - The client then executes the command: iperf3 -c <host> -l <length> -p <port> -i <interval> -u -b <bandwidth/number> -t <time>
  - The -l, -p, -i, -u, -b, and -t options are optional. The -p option must be used by the server and client at the same time and have the same value; the -i option does not need to be used by the server and client at the same time and can have different values;
  - For example:
  - iperf3 -s -p 5004 -i 1
  - iperf3 -c 192.168.1.104 -l 1460 -p 5004 -i 2 -t 20 //TCP
  - iperf3 -c 192.168.1.104 -l 1472 -p 5004 -i 4 -t 30 -u -b 50M //UDP
- After the server executes the command, the print information is displayed in the window, telling us that the server is enabled and listening on the corresponding port. After the client executes the command, the test machine and development board will print the test information at the same time.

1.11. iperf

The iperf command calls iperf2 to perform network speed test. iperf runs in TCP mode by default, and the UDP mode must be specified through the -u option. The relevant options of the command (case-sensitive) are as follows.

1.11.1. iperf -h

As shown in ***Figure 1-16. iperf -h command***, the serial port will print out the options related to the iperf command.

Figure 1-16. iperf -h command

```
# iperf -h
Usage:
    iperf <-s|-c hostip|exit|-h> [options]
Client/Server:
    -u #      use UDP rather than TCP
    -i #      seconds between periodic bandwidth reports
    -l #      length of buffer to read or write (default 1460 Bytes)
    -p #      server port to listen on/connect to (default 5001)
Server specific:
    -s        run in server mode
Client specific:
    -b #      bandwidth to send at in bits/sec (default 1 Mbit/sec, implies -u)
    -S #      set the IP 'type of service'
    -c <host> run in client mode, connecting to <host>
    -t #      time in seconds to transmit for (default 10 secs)
#
```

1.11.2. iperf -s [options]

- iperf -s

Enable an iperf2 server in TCP mode, which listens on port 5001 by default, and other options are default values.

- iperf -s -u

Enable an iperf2 server in UDP mode, which listens on port 5001 by default, and other options are default values.

- -i <interval>

Set the period of the test results printed by the serial port (Interval column), in seconds, with a range of 1-3600 (which must be integer; non-integer should be rounded down). The default is 1.

- -l <length>

Set the length of the read and write buffer, in bytes. The default is 1460 bytes, the maximum value in UDP mode is 2380, and the maximum value in TCP mode is 4380. The recommended value is 1472 in UDP mode and 1460 in TCP mode.

- -p <port>

Set the port that the server listens on. The port range is 0-65535, and the default is 5001.

1.11.3. iperf -c <host> [options]

- iperf -c <host>

Enable an iperf2 client, and make a TCP connection with the server whose IP address is

<host> on the default port 5001. All other options are default values.

- iperf -c <host> -u

Enable an iperf3 client, and make a UDP connection with the server whose IP address is <host> on the default port 5001. All other options are default values.

- -i <interval>
- -l <length>

The -l and -i option settings are the same as those on the server.

- -p <port>

Set the client connection port, which must be the same as the port that the server listens on.

- -b <bandwidth>

The unit of bandwidth is bits/sec and the format is data[KMG]. For example, 50K, 50k or 50000 means that the bandwidth is 50 Kbits/sec; when bandwidth is 0, it means that there is no limit. The default is 1 Mbit/sec. Only used in UDP mode.

- -t <time>

Set the total transmission time. The default is 10 seconds.

- -S <QOS value>

Set the QOS service type of the IP packet. The number range is 0-255, and hexadecimal (0x prefix) or decimal can be used, such as 0x16 = 22.

1.11.4. iperf exit

The command is used to stop the iperf2 test.

1.11.5. iperf2 test example

- Connect the development board and test machine to the same AP, and then view the IP address of the development board.
  - Use the wifi_connect command to connect the development board to the AP, and use the wifi_status command to view the IP address.
  - The test machine opens the iperf2 command window and starts the test.
  - The server first executes the command:
  - iperf -s -p <port> -i <interval> -l <length> //TCP
  - iperf -s -p <port> -i <interval> -l <length> -u //UDP
  - The client then executes the command:
  - iperf -c <host> -l <length> -p <port> -i <interval> -b <bandwidth/number> -t <time> -S <number>//TCP
  - iperf -c <host> -l <length> -p <port> -i <interval> -u -b <bandwidth/number> -t <time> -S <number>//UDP
  - The -l, -p, -i, -u, -b, -t, and -S options are optional.
  - ! ! Note: The -p option must be used by the server and client at the same time and have the same value; the -i option does not need to be used by the server and client

at the same time and can have different values; the -u option must be used by the server and client at the same time.

- For example:
- iperf -s -p 5004 -i 1 //TCP
- iperf -s -p 5004 -i 1 –u //UDP
- iperf -c 192.168.1.104 -l 1460 -p 5004 -i 2 -t 20 -S 0xe0 //TCP
- iperf -c 192.168.1.104 -l 1472 -p 5004 -i 4 -t 30 -S 0xe0 -u -b 50M //UDP

- After the server executes the command, the print information is displayed in the window, telling us that the server is enabled and listening on the corresponding port. After the client executes the command, the test machine and development board will print the test information at the same time.

1.12. BLE

This section introduces BLE related commands.

1.12.1. ble_help

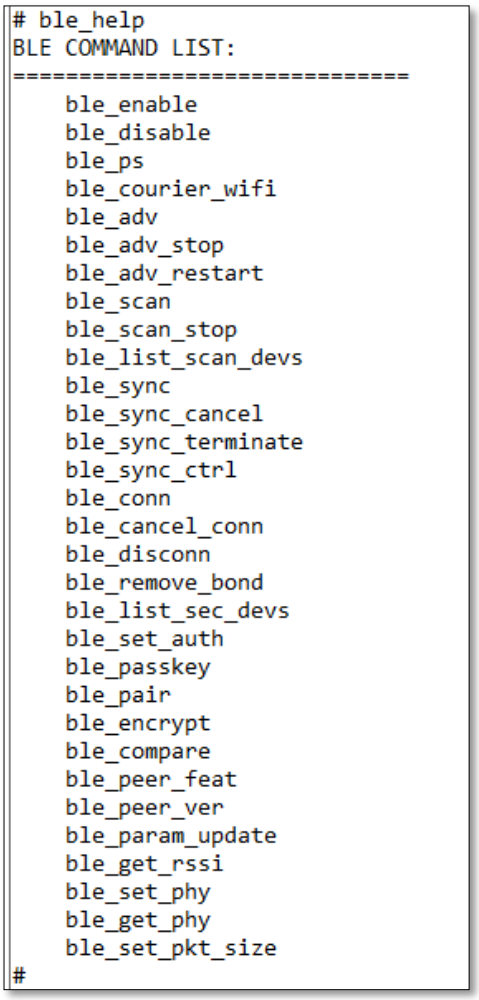
This command has no option.

As shown in ***Figure 1-17. ble_help command (msdk configuration)*** and ***Figure 1-18. ble_help command (msdk_ffd configuration)***, the ble_help command will list all commands of BLE. Different BLE commands can be executed in different configurations, so the ble_help command will list different BLE commands.

Figure 1-17. ble_help command (msdk configuration)

```
ble_help
BLE COMMAND LIST:
==============================
    ble_enable
    ble_disable
    ble_ps
    ble_courier_wifi
    ble_adv
    ble_adv_stop
    ble_adv_restart
    ble_disconn
    ble_remove_bond
    ble_list_sec_devs
    ble_set_auth
    ble_pair
    ble_encrypt
    ble_passkey
    ble_compare
    ble_peer_feat
    ble_peer_ver
    ble_param_update
    ble_get_rssi
#
```

Figure 1-18. ble_help command (msdk_ffd configuration)



1.12.2. ble_enable

This command has no option.

ble_enable is used to enable BLE. The execution of other BLE related commands takes effect only when BLE is enabled. After the development board is started correctly, BLE is enabled by default, so this command does not need to be executed. This command is usually used together with ble_disable. If the ble_enable command is used after BLE is disabled, BLE will enter the initial state and will not resume to the state before the ble_disable operation.

As shown in ***Figure 1-19. ble_enable command***, after BLE is disabled, executing ble_enable will enable BLE, and the serial port will display the reset log; if BLE is already enabled, the serial port will prompt that BLE is enabled.

Figure 1-19. ble_enable command

```
# ble_disable
ble disable success
# ble_enable
# BLE local addr: AB:89:67:45:23:01, type 0x0
=== BLE Adapter enable complete ===

# ble_enable
ble already enable
#
```

1.12.3. ble_disable

This command has no option.

ble_disable can be used to disable BLE. After BLE is disabled, some commands such as ble_adv, ble_scan, and ble_conn cannot be executed.

The BLE software and hardware will be reset after this command is executed, and then BLE will be disabled, so the command has slightly different execution results for different conditions of the development board, for example:

- If the development board does not enable any function of BLE, BLE will be disabled directly;
- If the development board has created a connection, the connection will be disconnected and then BLE will be disabled;
- If the development board has enabled advertising, it will stop advertising and then BLE will be disabled;
- If the development board has enabled scanning, it will stop scanning and then BLE will be disabled;
- If BLE is disabled, the serial port will prompt that BLE is disabled.

As shown in ***Figure 1-20. ble_disable command***, a prompt will be printed after ble_disable is executed.

Figure 1-20. ble_disable command

```
# ble_disable
ble disable success
#
# ble_adv 0
ble is disabled, please 'ble_enable' before
Error!
# ble_disable
ble is disabled, please 'ble_enable' before
Error!
#
```

1.12.4. ble_ps

- Usage: ble_ps <0 or 1>

This command is used to configure the power save function of BLE which is enabled by default. When ps mode is 1, the power save mode is enabled. When nothing of BLE is being processed or adv/scan/connection interval is greater than 5 ms, BLE core will be able to enter the sleep mode to save power. When ps mode is 0, the power save mode is disabled, and BLE core will not enter the sleep mode.

As shown in ***Figure 1-21. ble_ps command*** *错误!未找到引用源。*, a prompt will be printed after ble_ps is executed.

Figure 1-21. ble_ps command

```
# ble_ps
Current ps mode: 1
Usage: ble_ps <0 or 1>
    0: ble not deep sleep
    1: ble deep sleep
# ble_ps 0
ble_ps config complete. ps mode: 0
# ble_ps 1
ble_ps config complete. ps mode: 1
#
```

1.12.5. ble_courier_wifi

- Usage: ble_courier_wifi <0:disable or 1:enable>

This command is used to enable and disable the Bluetooth distribution network (Wi-Fi network configuration) function, which is disabled by default. After the function is enabled, the device will send advertising packets for the mobile phone to discover. The WeChat applet "GD Bluetooth distribution network" can be used to connect with the development board and continue the distribution operations. When the function is disabled, advertising will also be disabled.

As shown in ***Figure 1-22. ble_courier_wifi command*** *错误!未找到引用源。*, a prompt will be printed after ble_courier_wifi is executed.

Figure 1-22. ble_courier_wifi command

```
# ble_courier_wifi
Usage: ble_courier_wifi <0:disable; 1:enable>
#
# ble_courier_wifi 1
bcwl_adv_mgr_evt_hdlr adv[255] state change 0x0 ==> 0x1, reason 0x0
ble_courier_wifi ret:0
# bcwl_adv_mgr_evt_hdlr adv[1] state change 0x1 ==> 0x2, reason 0x0
bcwl_adv_mgr_evt_hdlr adv[1] state change 0x2 ==> 0x3, reason 0x0
bcwl_adv_mgr_evt_hdlr adv[1] state change 0x3 ==> 0x4, reason 0x0
bcwl_adv_mgr_evt_hdlr adv[1] state change 0x4 ==> 0x6, reason 0x0

# ble_courier_wifi 0
ble_courier_wifi ret:0
# bcwl_adv_mgr_evt_hdlr adv[1] state change 0x6 ==> 0x2, reason 0x0
bcwl_adv_mgr_evt_hdlr adv[255] state change 0x2 ==> 0x0, reason 0x0
```

1.12.6. ble_adv

- Usage: ble_adv <adv type>

This command is used to enable advertising so that local device can be found and connected by other BLE devices. The advertising type can be set to legacy advertising (scannable connectable undirected), extended advertising (connectable undirected), or periodic advertising (undirected periodic) through adv type parameter. Up to 2 advertising sets are supported at the same time.

If the device is connected by other devices, advertising will be stopped but will not be removed.

As shown in ***Figure 1-23. ble_adv command*** *错误!未找到引用源。*, a prompt will be printed after ble_adv is executed. When adv state is 0x6, it means success; otherwise, the execution fails. Adv index will also be prompted and can be used for the ble_adv_stop or ble_adv_restart command. For example, adv idx in the figure below is 0.

Figure 1-23. ble_adv command

```
# ble_adv
Usage: ble_adv <adv type>
<adv type>: advertising type, value 0 ~ 2
      0: legacy advertising, 1: extended advertising, 2: periodic advertising
      support 2 advertising sets at the same time
#
# ble_adv 0
adv state change 0x0 ==> 0x1, reason 0x0
adv index 0
# adv state change 0x1 ==> 0x2, reason 0x0
adv state change 0x2 ==> 0x3, reason 0x0
adv state change 0x3 ==> 0x4, reason 0x0
adv state change 0x4 ==> 0x6, reason 0x0
```

1.12.7. ble_adv_stop

- Usage: ble_adv_stop <adv idx> [remove]
- adv idx: advertising index, which can be obtained from the log of executing the ble_adv command
- remove: whether advertising needs to be removed after it is stopped. The default value is 1, and in this case, advertising will be removed after it is stopped. If the value is set to 0, advertising will not be removed, and can be enabled again through ble_adv_restart. This operation skips the creation procedure compared with the operation of enabling advertising through ble_adv.

This command is used to disable advertising.

As shown in ***Figure 1-24. ble_adv_stop command*** *错误!未找到引用源。*, a prompt will be printed after ble_adv_stop is executed. When an invalid adv idx is used, a fail prompt and the non-zero status will be displayed.

Figure 1-24. ble_adv_stop command

```
# ble_adv_stop 0
# adv state change 0x6 ==> 0x2, reason 0x0
adv stopped, remove 1
adv state change 0x2 ==> 0x0, reason 0x0

# ble_adv_stop 1 0
# adv state change 0x6 ==> 0x2, reason 0x0
adv stopped, remove 0

# ble_adv_stop 0
stop adv fail status 0x40
#
```

1.12.8. ble_adv_restart

- Usage: ble_adv_restart <adv idx>
- adv idx: advertising index, which can be obtained from the log of executing the ble_adv command

This command is used to restart advertising which is in stopped state. In the following two scenarios, the advertising can be restarted through ble_adv_restart : 1. after ble_adv enables advertising and a connection is established, advertising will be stopped; 2. after 'ble_adv_stop idx 0' is executed, advertising will be stopped but not removed.

As shown in ***Figure 1-25. ble_adv_restart command*** *错误!未找到引用源。*, a prompt will be printed after ble_adv_restart is executed. When adv state is 0x6, it means restart success; otherwise, it means failure; if adv idx is an illegal index, a failure log will be printed.

Figure 1-25. ble_adv_restart command

```
# ble_adv 0
adv state change 0x0 ==> 0x1, reason 0x0
adv index 0
# adv state change 0x1 ==> 0x2, reason 0x0
adv state change 0x2 ==> 0x3, reason 0x0
adv state change 0x3 ==> 0x4, reason 0x0
adv state change 0x4 ==> 0x6, reason 0x0
ble_adv_stop 0 0
# adv state change 0x6 ==> 0x2, reason 0x0
adv stopped, remove 0

# ble_adv_restart 0
# adv state change 0x2 ==> 0x6, reason 0x0

# ble_adv_restart 1
restart adv fail 0x40
#
```

1.12.9. ble_scan

This command has no option.

This command can only be used in msdk_ffd configuration.

This command is used to enable BLE scan procedure, which will remove the information scanned last time and print out information of the devices scanned this time, including the device address, device address type, rssi, name, dev idx, etc. Of which, dev idx can be used to connect or sync. ble_scan_stop can be used to stop the scan procedure.

As shown in ***Figure 1-26. ble_scan command*** *错误!未找到引用源。*, a prompt will be printed after ble_scan is executed.

Figure 1-26. ble_scan command

```
# ble_scan
# Ble Scan enabled status 0x0
new device addr A0:0B:16:90:45:D4, addr type 0x0, rssi -93, sid 0xff, dev idx 0, peri_adv_int 0, name
new device addr B8:7C:6F:A9:80:91, addr type 0x0, rssi -76, sid 0xff, dev idx 1, peri_adv_int 0, name
new device addr 61:A2:D2:6C:AB:32, addr type 0x1, rssi -91, sid 0xff, dev idx 2, peri_adv_int 0, name
new device addr 7D:F5:F7:70:77:8C, addr type 0x1, rssi -65, sid 0xff, dev idx 3, peri_adv_int 0, name
new device addr 79:C8:B9:04:03:AA, addr type 0x1, rssi -62, sid 0xff, dev idx 4, peri_adv_int 0, name
new device addr 05:55:95:51:C4:D7, addr type 0x1, rssi -81, sid 0xff, dev idx 5, peri_adv_int 0, name
ble_scan_stop
# Ble Scan disabled  status 0x0
```

1.12.10. ble_scan_stop

This command has no option.

This command can only be used in msdk_ffd configuration.

This command is used to disable the scan procedure. The status is 0 after success; otherwise, it fails.

As shown in ***Figure 1-27. ble_scan_stop command*** *错误!未找到引用源。*, a prompt will be printed after ble_scan_stop is executed.

Figure 1-27. ble_scan_stop command

```
# ble_scan
# Ble Scan enabled status 0x0
new device addr A0:0B:16:90:45:D4, addr type 0x0, rssi -93, sid 0xff, dev idx 0, peri_adv_int 0, name
new device addr B8:7C:6F:A9:80:91, addr type 0x0, rssi -76, sid 0xff, dev idx 1, peri_adv_int 0, name
new device addr 61:A2:D2:6C:AB:32, addr type 0x1, rssi -91, sid 0xff, dev idx 2, peri_adv_int 0, name
new device addr 7D:F5:F7:70:77:8C, addr type 0x1, rssi -65, sid 0xff, dev idx 3, peri_adv_int 0, name
new device addr 79:C8:B9:04:03:AA, addr type 0x1, rssi -62, sid 0xff, dev idx 4, peri_adv_int 0, name
new device addr 05:55:95:51:C4:D7, addr type 0x1, rssi -81, sid 0xff, dev idx 5, peri_adv_int 0, name
ble_scan_stop
# Ble Scan disabled  status 0x0
```

1.12.11. ble_list_scan_devs

This command has no option.

This command can only be used in msdk_ffd configuration.

This command is used to query the latest scanned devices and device index and device address will be displayed.

As shown in ***Figure 1-28. ble_list_scan_devs command*** *错误!未找到引用源。*, a prompt will be printed after ble_list_scan_devs is executed.

Figure 1-28. ble_list_scan_devs command

```
# ble_scan
# Ble Scan enabled status 0x0
new device addr A0:0B:16:90:45:D4, addr type 0x0, rssi -93, sid 0xff, dev idx 0, peri_adv_int 0, name
new device addr B8:7C:6F:A9:80:91, addr type 0x0, rssi -76, sid 0xff, dev idx 1, peri_adv_int 0, name
new device addr 61:A2:D2:6C:AB:32, addr type 0x1, rssi -91, sid 0xff, dev idx 2, peri_adv_int 0, name
new device addr 7D:F5:F7:70:77:8C, addr type 0x1, rssi -65, sid 0xff, dev idx 3, peri_adv_int 0, name
new device addr 79:C8:B9:04:03:AA, addr type 0x1, rssi -62, sid 0xff, dev idx 4, peri_adv_int 0, name
new device addr 05:55:95:51:C4:D7, addr type 0x1, rssi -81, sid 0xff, dev idx 5, peri_adv_int 0, name
ble_scan_stop
# Ble Scan disabled  status 0x0

# ble_list_scan_devs
dev idx: 0, device addr: A0:0B:16:90:45:D4
dev idx: 1, device addr: B8:7C:6F:A9:80:91
dev idx: 2, device addr: 61:A2:D2:6C:AB:32
dev idx: 3, device addr: 7D:F5:F7:70:77:8C
dev idx: 4, device addr: 79:C8:B9:04:03:AA
dev idx: 5, device addr: 05:55:95:51:C4:D7
```

1.12.12. ble_sync

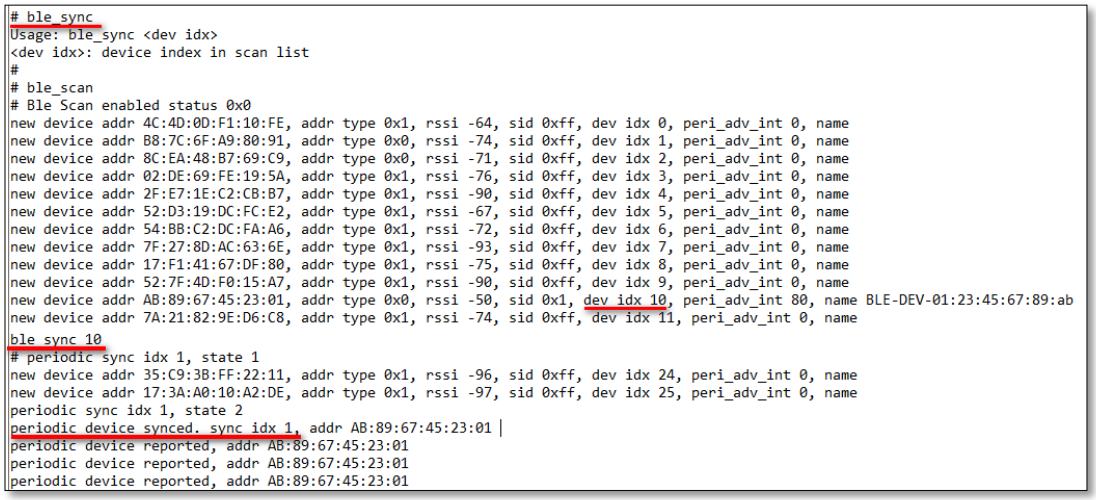
- Usage: ble_sync <dev idx>
- dev idx needs to be obtained from the scan list.

This command can only be used in msdk_ffd configuration.

This command is used to synchronize with a periodic advertising. BLE scan must be kept enabled until synchronization is established successfully. If sync is successful, the sync idx will be printed for the ble_sync_terminate or ble_syc_ctrl command to use. The periodic advertising report function is enabled by default, so the application will periodically print periodic advertising report logs. The ble_sync_ctrl command can be used to disable the report function.

As shown in ***Figure 1-29. ble_sync command*** *错误!未找到引用源。*, a prompt will be printed after ble_sync is executed.

Figure 1-29. ble_sync command



1.12.13. ble_sync_cancel

This command has no option.

This command can only be used in msdk_ffd configuration.

When sync is created but unestablished, this command can be used to cancel the operation.

As shown in ***Figure 1-30. ble_sync_cancel command*** *错误!未找到引用源。*, a prompt will be printed after ble_sync_cancel is executed.

Figure 1-30. ble_sync_cancel command

```
# ble_sync 7
# periodic sync idx 1, state 1

# ble_sync_cancel
per sync cancel success
# periodic sync idx 1, state 3
periodic sync idx 1, state 0
```

1.12.14. ble_sync_terminate

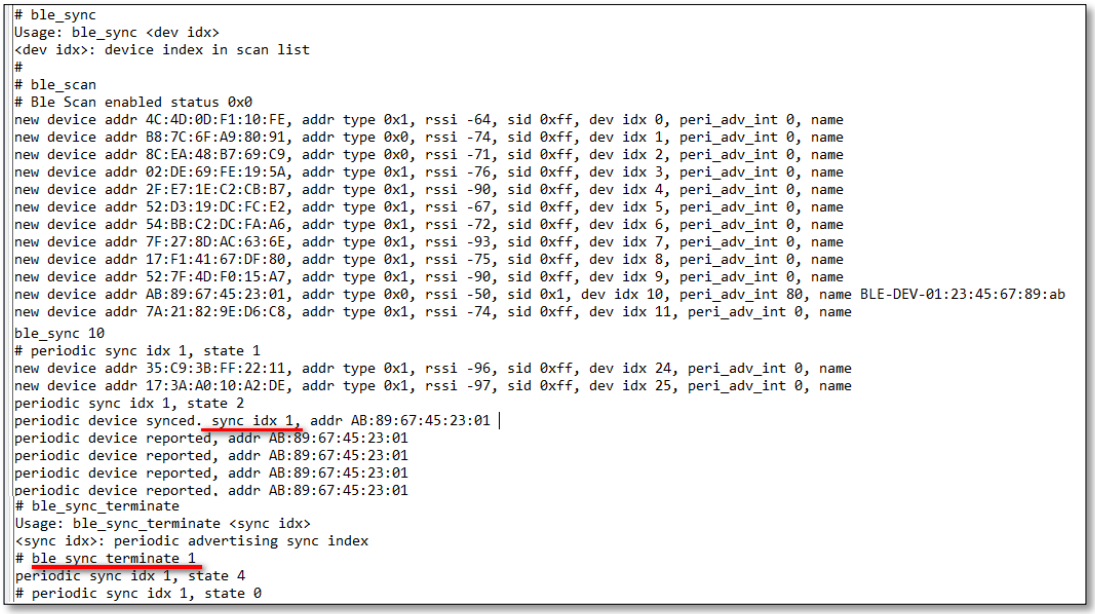
- Usage: ble_sync_terminate <sync idx>
- sync idx needs to be obtained from the log of successful establishment of sync through the ble_sync command.

This command is used to terminate the specified sync link.

This command can only be used in msdk_ffd configuration.

As shown in ***Figure 1-31. ble_sync_terminate command*** *错误!未找到引用源。*, a prompt will be printed after ble_sync_terminate is executed.

Figure 1-31. ble_sync_terminate command



1.12.15. ble_sync_ctrl

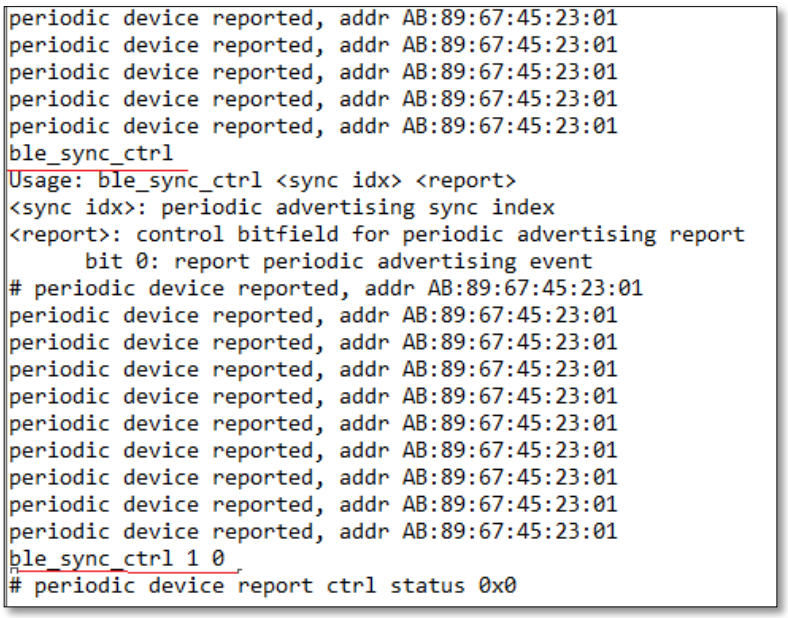
- Usage: ble_sync_ctrl <sync idx> <report>
- sync idx needs to be obtained from the log of successful establishment of sync through the ble_sync command.

This command can only be used in msdk_ffd configuration.

This command is used to enable or disable the periodic advertising report function, which is enabled by default. When enabled, every time a synchronized message is received, it will be reported to APP.

As shown in ***Figure 1-32. ble_sync_ctrl command*** *错误!未找到引用源。*, a prompt will be printed after ble_sync_ctrl is executed.

Figure 1-32. ble_sync_ctrl command



1.12.16. ble_conn

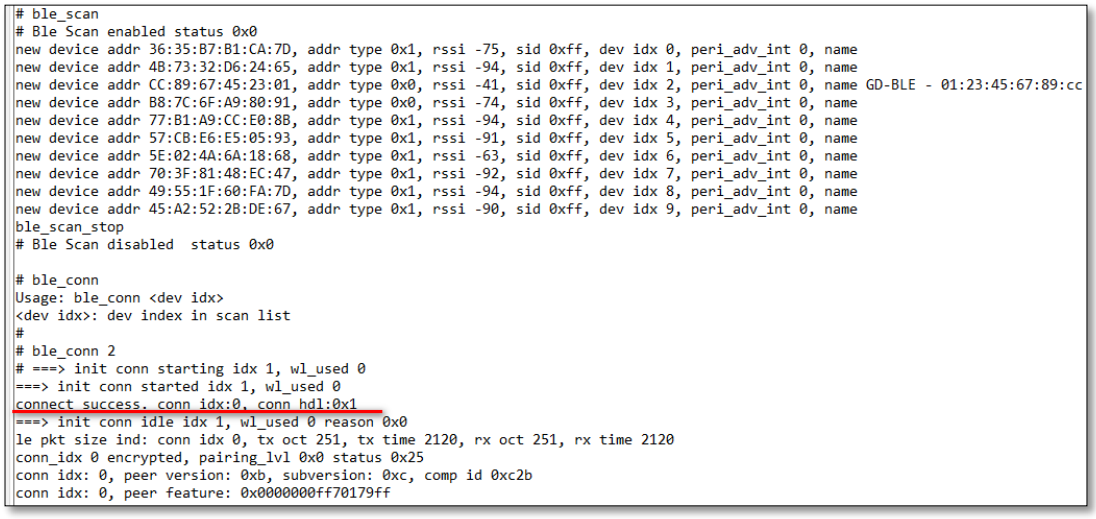
- Usage: ble_conn <dev idx>
- dev idx needs to be obtained from the scan list.

This command can only be used in msdk_ffd configuration.

This command is used to initiate a connection. Before executing this command, ble_scan should be used to get dev idx in the scanned list. If the target device is not scanned, the connection cannot be established.

As shown in ***Figure 1-33. ble_conn command*** *错误!未找到引用源。*, a prompt will be printed after ble_conn is executed. If the connection is successfully established, the log with a red underline in ***Figure 1-33. ble_conn command*** will be printed in which conn idx can be used in commands such as ble_disconn, ble_pair, and ble_encrypt.

Figure 1-33. ble_conn command



```
# ble_scan
# Ble Scan enabled status 0x0
new device addr 36:35:B7:B1:CA:7D, addr type 0x1, rssi -75, sid 0xff, dev idx 0, peri_adv_int 0, name
new device addr 4B:73:32:D6:24:65, addr type 0x1, rssi -94, sid 0xff, dev idx 1, peri_adv_int 0, name
new device addr CC:89:67:45:23:01, addr type 0x0, rssi -41, sid 0xff, dev idx 2, peri_adv_int 0, name GD-BLE - 01:23:45:67:89:cc
new device addr B8:7C:6F:A9:80:91, addr type 0x0, rssi -74, sid 0xff, dev idx 3, peri_adv_int 0, name
new device addr 77:B1:A9:CC:E0:8B, addr type 0x1, rssi -94, sid 0xff, dev idx 4, peri_adv_int 0, name
new device addr 57:CB:E6:E5:05:93, addr type 0x1, rssi -91, sid 0xff, dev idx 5, peri_adv_int 0, name
new device addr 5E:02:4A:6A:18:68, addr type 0x1, rssi -63, sid 0xff, dev idx 6, peri_adv_int 0, name
new device addr 70:3F:81:48:EC:47, addr type 0x1, rssi -92, sid 0xff, dev idx 7, peri_adv_int 0, name
new device addr 49:55:1F:60:FA:7D, addr type 0x1, rssi -94, sid 0xff, dev idx 8, peri_adv_int 0, name
new device addr 45:A2:52:2B:DE:67, addr type 0x1, rssi -90, sid 0xff, dev idx 9, peri_adv_int 0, name
ble_scan_stop
# Ble Scan disabled  status 0x0

# ble_conn
Usage: ble_conn <dev idx>
<dev idx>: dev index in scan list
#
# ble_conn 2
# ===> init conn starting idx 1, wl_used 0
===> init conn started idx 1, wl_used 0
connect success. conn idx:0, conn hdl:0x1
===> init conn idle idx 1, wl_used 0 reason 0x0
le pkt size ind: conn idx 0, tx oct 251, tx time 2120, rx oct 251, rx time 2120
conn_idx 0 encrypted, pairing_lvl 0x0 status 0x25
conn idx: 0, peer version: 0xb, subversion: 0xc, comp id 0xc2b
conn idx: 0, peer feature: 0x0000000ff70179ff
```

1.12.17. ble_cancel_conn

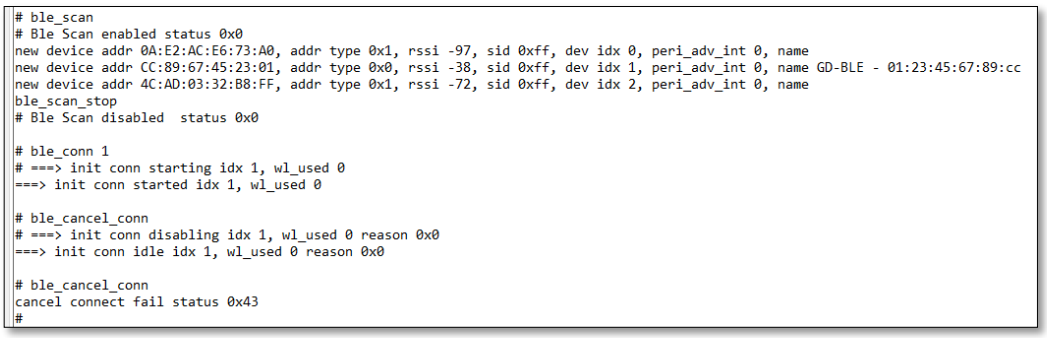
This command has no option.

This command can only be used in msdk_ffd configuration.

This command is used to cancel an unestablished connection after the ble_conn command is executed. If the connection is successfully established and needs to be disconnected, execute the ble_disconn command.

As shown in ***Figure 1-34. ble_cancel_conn command*** *错误!未找到引用源。*, a prompt will be printed after ble_cancel_conn is executed. When init conn enters the idle state, it means that the execution is successful.

Figure 1-34. ble_cancel_conn command



```
# ble_scan
# Ble Scan enabled status 0x0
new device addr 0A:E2:AC:E6:73:A0, addr type 0x1, rssi -97, sid 0xff, dev idx 0, peri_adv_int 0, name
new device addr CC:89:67:45:23:01, addr type 0x0, rssi -38, sid 0xff, dev idx 1, peri_adv_int 0, name GD-BLE - 01:23:45:67:89:cc
new device addr 4C:AD:03:32:B8:FF, addr type 0x1, rssi -72, sid 0xff, dev idx 2, peri_adv_int 0, name
ble_scan_stop
# Ble Scan disabled  status 0x0

# ble_conn 1
# ===> init conn starting idx 1, wl_used 0
===> init conn started idx 1, wl_used 0

# ble_cancel_conn
# ===> init conn disabling idx 1, wl_used 0 reason 0x0
===> init conn idle idx 1, wl_used 0 reason 0x0

# ble_cancel_conn
cancel connect fail status 0x43
#
```

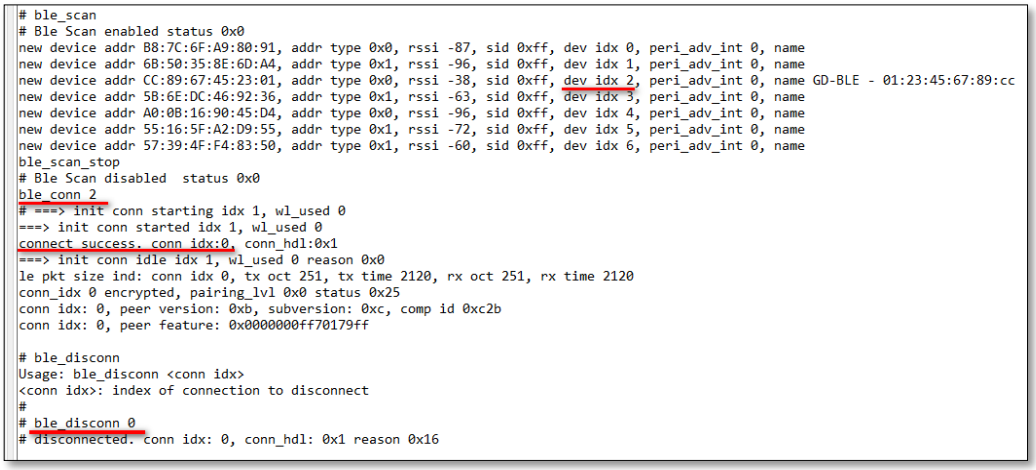
1.12.18. ble_disconn

- Usage: ble_disconn <conn idx>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command is used by the device to disconnect the established connection.

As shown in ***Figure 1-35. ble_disconn command*** *错误!未找到引用源。*, a prompt will be printed after ble_disconn is executed.

Figure 1-35. ble_disconn command



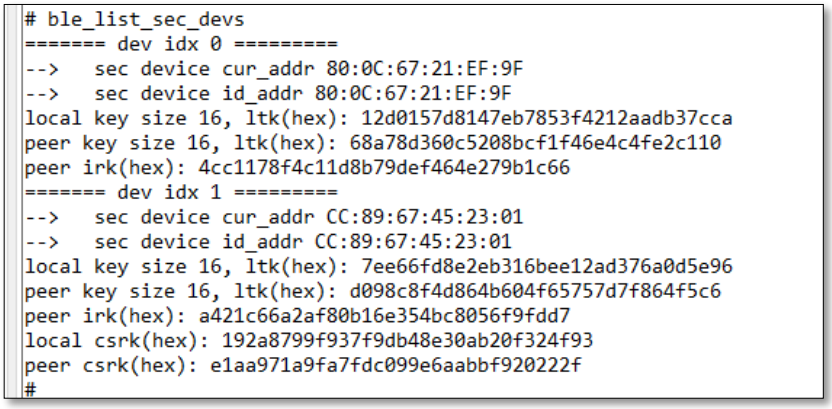
1.12.19. ble_list_sec_devs

This command has no option.

This command is used to query the bonded device information stored in flash and the peer device information currently connected, including dev idx, id_addr, LTK, IRK, etc.

As shown in ***Figure 1-36. ble_list_sec_devs command*** *错误!未找到引用源。*, a prompt will be printed after ble_list_sec_devs is executed.

Figure 1-36. ble_list_sec_devs command



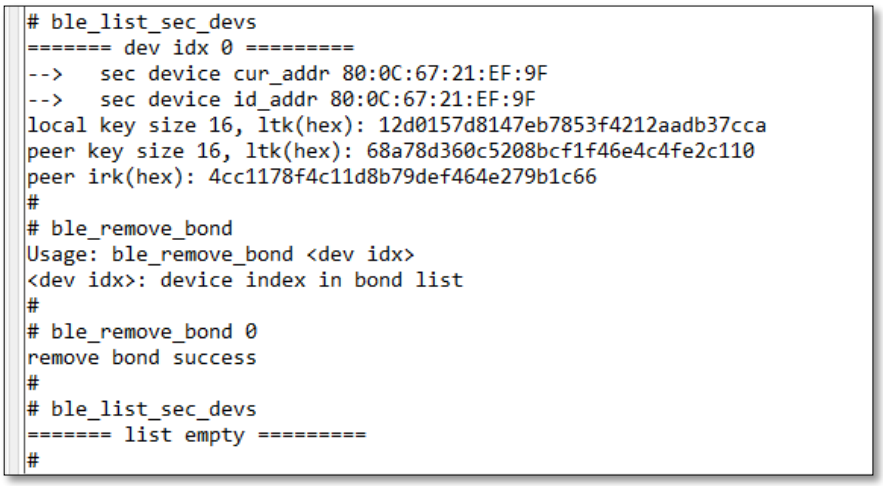
1.12.20. ble_remove_bond

- Usage: ble_remove_bond <dev idx>
- dev idx needs to be obtained from the ble_list_sec_devs command.

This command is used to remove the bond information of the device. If the device is in the connected state, the connection will be disconnected, and then the bond information will be removed. The corresponding content in flash will also be removed.

As shown in ***Figure 1-37. ble_remove_bond command*** *错误!未找到引用源。*, a prompt will be printed after ble_remove_bond is executed.

Figure 1-37. ble_remove_bond command



1.12.21. ble_set_auth

- Usage: ble_set_auth <bond> <mitm> <sc> <iocap>

This command is used to configure device security strategies: whether to save pairing information after pairing, whether to support man-in-the-middle attack protection, whether to support secure connection and IO capabilities, etc.

If bond flag is configured, the LTK, IRK, CSRK, and other information of the peer device will be saved to flash after the device is paired successfully. The configuration of mitm flag means that man-in-the-middle attack protection is supported. If the peer device also supports it, different pairing methods can be selected according to IO capabilities. The configuration of sc flag means that the device supports secure connection. If the peer device also supports it, a long-term key can be generated through the ECDH key exchange algorithm. The configuration of iocap allows the selection of IO capacities to be used during pairing, which support display only, display yes no, keyboard only, no input no output, keyboard display, etc.

As shown in ***Figure 1-38. ble_set_auth command*** *错误!未找到引用源。*, a prompt will be printed after ble_set_auth is executed.

Figure 1-38. ble_set_auth command

```
# ble_set_auth
Usage: ble_set_auth <bond> <mitm> <sc> <iocap>
<bond>: bonding flag for authentication
      0x00: no bonding
      0x01: bonding
<mitm>: mitm flag for authentication
      0x00: mitm protection not required
      0x01: mitm protection required
<sc>: secure connections flag for authention
      0x00: secure connections pairing is not supported
      0x00: secure connections pairing is supported
<iocap>: io capability to set
      0x00: display only
      0x01: display yes no
      0x02: keyboard only
      0x03: no input no output
      0x04: keyboard display
#
# ble_set_auth 1 0 0 2
ble set auth success.
```

1.12.22. ble_pair

- Usage: ble_pair <conn idx>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command is used to start pairing with the specified device connected. The pairing operation is used to create a key that can be used to encrypt the link.

As shown in ***Figure 1-39. ble_pair command*** *错误!未找到引用源。*, a prompt will be printed after ble_pair is executed.

Figure 1-39. ble_pair command

```
# ble_pair
Usage: ble_pair <conn idx>
<conn idx>: index of the connection to pair
#
# ble_pair 0
# bond ind, key size 16, ltk: 0xbf528921c3f9e555e3b71972b0951ca7
rcv remote irk: 0x4cc1178f4c11d8b79def464e279b1c66
rcv remote identity addr: 0x80:0xc:0x67:0x21:0xef:0x9f, type 0
conn_idx 0 pairing success, level 0x1 ltk_present 1 sc 0
local key size 16, ltk(hex): 6d99cb37930a4a239034ac67dc32a7f9
peer key size 16, ltk(hex): bf528921c3f9e555e3b71972b0951ca7
peer irk(hex): 4cc1178f4c11d8b79def464e279b1c66
bond data ind: gatt_start_hdl 0, gatt_end_hdl 0, svc_chg_hdl 0, cli_info 1, cli_feat 0, srv_feat 0
```

1.12.23. ble_passkey

- Usage: ble_passkey <conn idx> <passkey>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command is used to input the passkey (6-digit number) during pairing procedure. The passkey must be the same with the one in the peer device to make sure pairing is successful.

As shown in ***Figure 1-40. ble_passkey command*** *错误!未找到引用源。*, a prompt will be printed after ble_passkey is executed.

Figure 1-40. ble_passkey command

```
# ble_set_auth 1 1 0 2
ble set auth success.
# ble_pair 0
# conn_idx 0 waiting for user to input key ......
ble_passkey
Usage: ble_passkey <conn idx> <passkey>
<conn idx>: index of connection to input passkey
<passkey>: passkey value to input, should be 6-digit value between 000000 and 999999
#
# ble_passkey 0 366279
input passkey0: 366279 passkey1: 0
# bond ind, key size 16, ltk: 0xe7b672e24a20a327567cc89d208c2f04
rcv remote irk: 0x4cc1178f4c11d8b79def464e279b1c66
rcv remote identity addr: 0x80:0xc:0x67:0x21:0xef:0x9f, type 0
conn_idx 0 pairing success, level 0x5 ltk_present 1 sc 0
local key size 16, ltk(hex): 9957c1d5710148fdf36cdbc7eb4cf8f3
peer key size 16, ltk(hex): e7b672e24a20a327567cc89d208c2f04
peer irk(hex): 4cc1178f4c11d8b79def464e279b1c66
bond data ind: gatt_start_hdl 0, gatt_end_hdl 0, svc_chg_hdl 0, cli_info 1, cli_feat 0, srv_feat 0
```

1.12.24. ble_encrypt

- Usage: ble_encrypt <conn idx>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

The command is used to encrypt the specified connection. If the link is already encrypted, the encryption key will be refreshed.

As shown in ***Figure 1-41. ble_encrypt command*** *错误!未找到引用源。*, a prompt will be printed after ble_encrypt is executed.

Figure 1-41. ble_encrypt command

```
# ble_encrypt
Usage: ble_encrypt <conn idx>
<conn idx>: index of the connection to start encryption
#
# ble_encrypt 0
# conn_idx 0 encrypted, pairing_lvl 0x5 status 0x0
conn idx 0 ping timeout set status 0x0
```

1.12.25. ble_compare

- Usage: ble_compare <conn idx> <result>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command is used to determine whether the temporary keys generated at both sides are the same during the pairing with the specified device connected.

As shown in ***Figure 1-42. ble_compare command*** *错误!未找到引用源。*, a prompt will be printed after ble_compare is executed.

Figure 1-42. ble_compare command

```
ble_conn 13
# ===> init conn starting idx 1, wl_used 0
===> init conn started idx 1, wl_used 0
connect success. conn idx:0, conn_hdl:0x1
===> init conn idle idx 1, wl_used 0 reason 0x0
le pkt size ind: conn idx 0, tx oct 251, tx time 2120, rx oct 251, rx time 2120
conn idx: 0, peer version: 0xb, subversion: 0xc, comp id 0xc2b
conn idx: 0, peer feature: 0x0000000ff70179ff
conn_idx 0 num val: 365294
waiting for user to compare......

# ble_compare
Usage: ble_compare <conn idx> <result>
<conn idx> index of connection
<result>: numeric comparison result, 0 for fail and 1 for success
#
# ble_compare 0 1
compare result: 1
# bond ind, key size 16, ltk: 0x1316d3d3bdb200f9bb006e9c9a663480
rcv remote irk: 0x9db73b59862a11c553732ca71f6e894
rcv remote identity addr: 0xab:0x89:0x67:0x45:0x23:0x1, type 0
bond ind csrk: e4 63 4c 41 7c 0d 04 57 fa c1 3e ca 38 8f 13 27
conn_idx 0 pairing success, level 0xd ltk_present 1 sc 1
local key size 16, ltk(hex): 1316d3d3bdb200f9bb006e9c9a663480
peer key size 16, ltk(hex): 1316d3d3bdb200f9bb006e9c9a663480
peer irk(hex): 9db73b59862a11c5530732ca71f6e894
local csrk(hex): 2e43fe4c2eda3d9ce2d5eedd8995d0dc
peer csrk(hex): e4634c417c0d0457fac13eca388f1327
```

1.12.26. ble_peer_feat

- Usage: ble_peer_feat <conn idx>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command is used to get the supported features of the specified device connected. For the meaning of each bit, refer to the FEATURE SUPPORT in the BLE Core Spec.

As shown in ***Figure 1-43. ble_peer_feat command*** *错误!未找到引用源。*, a prompt will be printed after ble_peer_feat is executed.

Figure 1-43. ble_peer_feat command

```
# ble_peer_feat
Usage: ble_peer_feat <conn idx>
<conn idx>: index of connection
#
# ble_peer_feat 0
# conn idx: 0, peer feature: 0x0000000ff70179ff
```

1.12.27. ble_peer_ver

- Usage: ble_peer_ver <conn idx>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command is used to get the version information of the specified device connected, including Bluetooth version information (0xb: BT5.2), subversion information, and company identifier (GigaDevice: 0x0C2B).

As shown in ***Figure 1-44. ble_peer_ver command*** *错误!未找到引用源。*, a prompt will be printed after ble_peer_ver is executed.

Figure 1-44. ble_peer_ver command

```
# ble_peer_ver
Usage: ble_peer_ver <conn idx>
<conn idx>: index of connection
#
# ble_peer_ver 0
# conn idx: 0, peer version: 0xb, subversion: 0xc, comp id 0xc2b
```

1.12.28. ble_get_rssi

- Usage: ble_get_rssi <conn idx>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command is used to get RSSI information of the latest packet received from the specified device connected.

As shown in ***Figure 1-45. ble_get_rssi command*** *错误!未找到引用源。*, a prompt will be printed after ble_get_rssi is executed.

Figure 1-45. ble_get_rssi command

```
# ble_get_rssi
Usage: ble_get_rssi <conn idx>
<conn idx>: index of connection
#
# ble_get_rssi 0
# conn idx 0 rssi: -42
ble_get_rssi 0
# conn idx 0 rssi: -55
```

1.12.29. ble_param_update

- Usage: ble_param_update <conn idx> <interval> <latency> <supv tout> <ce len>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

The command is used to update the parameters of the specified connection, such as connection interval, latency and supervision timeout.

As shown in ***Figure 1-46. ble_param_update command*** *错误!未找到引用源。*, a prompt will be printed after ble_param_update is executed.

Figure 1-46. ble_param_update command

```
# ble_param_update
Usage: ble_param_update <conn idx> <interval> <latency> <supv tout> <ce len>
<conn idx>: index of connection
<interval>: connection interval in unit of 1.25ms, range from 0x0006 to 0x0C80 in hex value
<latency>: connection latency to update in hex value
<supv tout>: supervision timeout in unit of 10ms, range from 0x000A to 0x0C80 in hex value
<ce len>: connection event length in unit of 0.625 ms in hex value
#
# ble_param_update 0 6 0 a 0
# conn idx 0, param update ind: interval 6, latency 0, sup to 10
conn idx 0, param update result status: 0x0
```

1.12.30. ble_set_phy

- Usage: ble_set_phy <conn idx> <tx phy> <rx phy> <phy opt>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command can only be used in msdk_ffd configuration.

This command is used to set TX/RX PHY for the specified connection. If TX/RX PHY is set to 0, it means that all PHYs are supported; otherwise, the meaning of each bit is shown in the figure below.

As shown in ***Figure 1-47. ble_set_phy command*** *错误!未找到引用源。*, a prompt will be printed after ble_set_phy is executed.

Figure 1-47. ble_set_phy command

```
# ble_set_phy
Usage: ble_set_phy <conn idx> <tx phy> <rx phy> <phy opt>
<conn idx>: index of connection
<tx phy>: transmit phy to set
      bit 0: 1M phy, bit 1: 2M phy, bit 2: coded phy
<rx phy>: receive phy to set
      bit 0: 1M phy, bit 1: 2M phy, bit 2: coded phy
<phy opt>: phy options for codec phy
      0x00: no prefer coding
      0x01: prefer S=2 coding be used
      0x02: prefer S=8 coding be used
#
# ble_set_phy 0 2 2 0
# le phy ind conn idx 0: tx phy 0x2, rx phy 0x2
conn idx 0 le phy set status 0x0
```

1.12.31. ble_get_phy

- Usage: ble_get_phy <conn idx>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command can only be used in msdk_ffd configuration.

This command is used to get the current TX/RX PHY of the specified connection.

As shown in ***Figure 1-48. ble_get_phy command*** *错误!未找到引用源。*, a prompt will be printed after ble_get_phy is executed, the result value meaning is 0x1: 1M; 0x2: 2M; 0x3: coded.

Figure 1-48. ble_get_phy command

```
# ble_get_phy
Usage: ble_get_phy <conn idx>
<conn idx>: index of connection
#
# ble_get_phy 0
# le phy ind conn idx 0: tx phy 0x1, rx phy 0x1
conn idx 0 le phy get status 0x0
```

1.12.32. ble_set_pkt_size

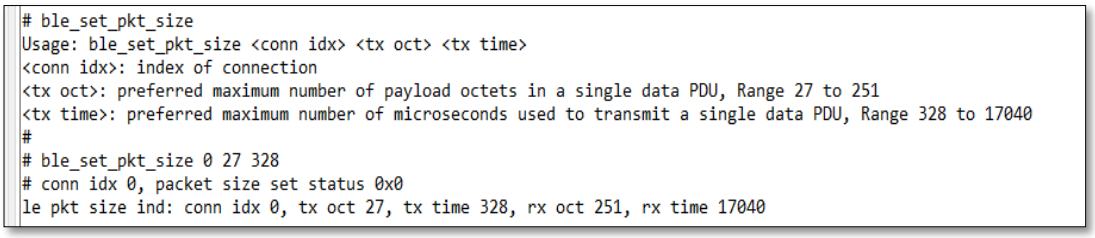
- Usage: ble_set_pkt_size <conn idx> <tx oct> <tx time>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command can only be used in msdk_ffd configuration.

This command is used to set the maximum number of bytes and time for sending a PDU on the specified connection.

As shown in ***Figure 1-49. ble_set_pkt_size command*** *错误!未找到引用源。*, a prompt will be printed after ble_set_pkt_size is executed.

Figure 1-49. ble_set_pkt_size command



2. Revision history

Table 2-1. Revision history

Revision No.	Description	Date
1.0	Initial release	Dec.5, 2023

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which have been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as its suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as its suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.