MiWi RN Command Set

For SAR30/R21 MiWi P2P and Star Protocol

Table of Contents

1. INTRODUCTION 3

2. PROTOCOL BASIC 4

3. COMMAND LIST 6

4. Command DETAILS 8

5. Command EXAMPLE 20

6. VERSION HISTORY 22

1. INTRODUCTION

This document is to describe MiWi ASC command set, which is also called as RN command set.

This command set only works with MiWi P2P or MiWi Star protocol, MiMi mesh is not supported.

The purpose of AT/RN command is to provide simple and readable command controlling and returning for host MCU or PC terminal.

If user uses PC terminal to talk with MiWi by RN command set, send command “echo” to enable echo will get his input echo back, this will be good for user to know his input.

If user uses host MCU to control MiWi by RN command set, don’t enable echo, as this will add extra cost for MCU to decoding data.

In the following, will call either PC terminal or host MCU as just host. And will call the slave unit controlled by host as just MiWi.

2. PROTOCOL BASIC

UART Configuration

Command uses UART as communication channel, baudrate is 115200bps, 8bit, no parity, and 1 stop bit. There is no flow control.

Command Format

To make command readable, all field of one command are just ASCII strings.

And both command sent from host and returned to host are ended by character '\r', which value s 0x0D. MiWi will detect '/r/ as the end of one complete command sent by host. On the contrary, host also need to detect '/r/ as the end of one complete command returned by MiWi.

Configure Mode and Action Mode

Here it is defined two command mode applied for this communication protocol. First is Configure Mode. In this mode, user can use the command to configure MiWi protocol level behaviors like channel, PAN ID. Usually these command are started by tag "cfg" and followed with parameters.

After execute command "~cfg", MiWi will exit from Configure Mode and enter Action Mode.

Those command started with "cfg" can only be accepted in Confgure mode, and it will be rejected in Action Mode.

Meanwhile, those command meaning MiWi action can only be accepted in Action mode, but will be rejected in Configure mode.

Only a few commands which don't make configuration or start action will be allowed in both Configure mode and Action mode.

After MiWi exit from Configure Mode, it will stay at Action Mode. There is no way to enter Configure mode again unless a reset.

Reboot

After MiWi is hardware or software reset, it will return a "Reboot" to host, host can only send his command after "Reboot" is received.

Echo

If host is a MCU, then don't need to enable echo.

But if host is a PC terminal, use typing in characters to control MiWi, it is better that use type in "echo" to enable echo mode. Under echo mode, any characters user input will be echoed back in PC terminal and displayed out, so user can clearly know what he has inputted.

Acknowledgement

With this communication protocol, any command is sent by host, there will be acknowledgement returned to host. Acknowledgement can be either AOK command, ERR command, or execution information. But there is only one command sent by MiWi after it gets command from host. As a result, host must wait for its acknowledgement before it sends another command.

On the contrary, in some situation, MiWi will initially send a command to host, host also need to give its acknowledgement as AOK or ERR back to MiWi.

3. COMMAND LIST

Host send to MiWi

- echo
- ~echo
- cfg reconn r1
- cfg pan r1
- cfg channel r1
- cfg phymod r1
- cfg txpower r1
- ~cfg
- get ver
- get addr
- get channel
- get pan
- get role
- get consize
- get conn r1
- get edsize
- get myindex
- get eds r1 r2
- start
- join
- remove r1
- send r1 r2 r3
- reset

MiWi report to host

AOK

ERR

Reboot

ver r1

addr r1

channel r1

pan r1

role r1

consize r1

conn r1 r2 r3

recv r1 r2 r3 r4

status r1

error r1

4. COMMAND DETAILS

echo

Enable MiWi echo back what user input. Send command to enable echo when using PC terminal to control, and don't send command to enable echo if using host MCU to control.

In default, echo mode is disabled.

Example: echo

Response: AOK

~echo

Disable MiWi echo back what user input.

Example: ~echo

Response: AOK

cfg reconn r1

Configure reconnection behavior. This command is only accepted in configuration mode.

r1: 8bits hex value, in range of 0~2, representing the reconnection behavior after system is rebooted, as shown in below table:

Reconnection setting	
0	Network freezer is disabled, it will create full new network, and automatically run network creation.
1	Network freezer is enabled, it will use old parameters stored in PDS to restore the network.
2	Network freezer is disabled, it will wait for "start" and "join" command to create network.

Example: cfg reconn 0 //create new network, and firmware run network creation automatically.

Response: AOK

ERR

cfg pan r1

Configure PAN ID for the MiWi network. This command is only accepted in configuration mode.

r1: 16bits hex value , representing the PAND ID to be configured.

Default value is 0x1234.

Example: cfg pan 7890 //configure PAND ID to 0x7890

Response: AOK

ERR

PAN ID will be stored into PDS, and can be used for network restore at next time restart.

cfg channel r1

Configure channel number for the MiWi network. This command is only accepted in configuration mode.

r1: 8bits hex value , representing the channel number to be configured. Valid value is in the range of 0 to 26. Otherwise it will report ERR.

Default value is channel 8 for SAMR30 devices, and channel 26 for SAMR21 devices.

Example: cfg channel 6 //configure to channel 6

Response: AOK

Notes about channel and band frequency supported in this firmware:

For SAMR21, only band 0 is supported by this firmware, and this band comply with IEEE 802.15.4. In this band, channel 0x0B to channel 0x1A can be selected, these channels are described in below table:

0x0B	2405MHz
0x0C	2410MHz
0x0D	2415MHz
0x0E	2420MHz
0x0F	2425MHz
0x10	2430MHz
0x11	2435MHz
0x12	2440MHz
0x13	2445MHz
0x14	2450MHz
0x15	2455MHz
0x16	2460MHz
0x17	2465MHz
0x18	2470MHz
0x19	2475MHz
0x1A	2480MHz

For SAMR30, also only band 0 is supported by this firmware, this band comply with IEEE 802.15.4, 868.3MHZ of EU band, and 906MHZ to 924MHZ of North American band. Below table describes the details:

0x00	868.3MHz
0x01	906MHz
0x02	908MHz
0x03	910MHz
0x04	912MHz
0x05	914MHz
0x06	916MHz
0x07	918MHz
0x08	920MHz
0x09	922MHz
0x0A	924MHz

Channel number will be stored into PDS, and can be used for network restore at next time restart.

cfg phymod r1

Configure PHY modulation for SAMR30 or SAMR21. This command is only accepted in configuration mode.

r1: 8bits hex value, while least 5bits will be accepted.

For SAMR30, this value will be set into register TRX_CTRL_2, bit5 OQPSK_SCRAM_EN, bit4 ALT_SPECTRUM, bit3 BPSK_OQPSK, bit2 SUB_MODE, bit1~0 OQPSK_DATA_RATE. Values in below table will be accepted, otherwise it will report ERR.

PHY Mode	7	6	5	4	3	2	1	0	Compliance
BPSK-20	-	-	-	0	0	0	0	0	IEEE 802.15.4-2003/2006/2011: channel page 0, channel 0
BPSK-40	-	-	-	0	0	1	0	0	IEEE 802.15.4-2003/2006/2011: channel page 0, channel 1 to 10
BPSK-40-ALT	-	-	-	1	0	1	0	0	Proprietary, alternative spreading code
OQPSK-SIN-RC-100	-	-	-	0	1	0	0	0	IEEE 802.15.4-2006/2011: channel page 2, channel 0
OQPSK-SIN-RC-200	-	-	-	0	1	0	0	1	Proprietary
OQPSK-SIN-RC-400-SCR-ON	-	-	1	0	1	0	1	0	Proprietary, scrambler on
OQPSK-SIN-RC-400-SCR-OFF	-	-	0	0	1	0	1	0	Proprietary, scrambler off
OQPSK-RC-100	-	-	-	1	1	0	0	0	Proprietary
OQPSK-RC-200	-	-	-	1	1	0	0	1	Proprietary
OQPSK-RC-400-SCR-ON	-	-	1	1	1	0	1	0	Proprietary, scrambler on
OQPSK-RC-400-SCR-OFF	-	-	0	1	1	0	1	0	Proprietary, scrambler off
OQPSK-SIN-250	-	-	-	0	1	1	0	0	IEEE 802.15.4-2006/2011: channel page 2, channel 1 to 10
OQPSK-SIN-500	-	-	-	0	1	1	0	1	Proprietary
OQPSK-SIN-500-ALT	-	-	-	0	1	1	1	1	Proprietary, alternative spreading code
OQPSK-SIN-1000-SCR-ON	-	-	1	0	1	1	1	0	Proprietary, scrambler on
OQPSK-SIN-1000-SCR-OFF	-	-	0	0	1	1	1	0	Proprietary, scrambler off
OQPSK-RC-250	-	-	-	1	1	1	0	0	IEEE 802.15.4-2011: channel page 5, channel 0 to 3
OQPSK-RC-500	-	-	-	1	1	1	0	1	Proprietary
OQPSK-RC-500-ALT	-	-	-	1	1	1	1	1	Proprietary, alternative spreading code
OQPSK-RC-1000-SCR-ON	-	-	1	1	1	1	1	0	Proprietary, scrambler on
OQPSK-RC-1000-SCR-OFF	-	-	0	1	1	1	1	0	Proprietary, scrambler off

When configure PHY modulation, need to consider channel number setting altogethor.

For SAMR21, this value is used to set register TRX_CTRL_2, bit2~0 OQPSK_DATA_RATE. Value only from 0 ~ 3 are accepted, other value will get ERR response.

Value	Description
0x0	250kbps
0x1	500kbps
0x2	1000kbps
0x3	2000kbps
0x4 ~ 0x7	All other values are reserved.

Default setting is BPSK-20 for channel 0 and BPSK-40-ALT for other channel for SAMR30, or 250kbps for SAMR21.

Example: cfg phymod 4 //configure to BPSK-40 for SAMR30, 40kbps, BPSK modulation.

Response: AOK

Example: cfg phymod 3 //configure to 2000kbps for SAMR21

Response: AOK

PHY modulation or data rate setting will not be stored into PDS, user need to configure them every time when system restarts.

cfg txpower r1

Configure PHY transmitter power for SAMR30 or SAMR21. This command is only accepted in configuration mode.

r1: 8bits hex value.

For SAMR30, firmware will not check the value is proper or not, while user need to set the correct value according to table below.

TX Power [dBm]	915MHz North American Band PHY Modes: BPSK-40 (GC_TX_OFFS=3), BPSK-40-ALT (GC_TX_OFFS=3), OQPSK-SIN-{250,500,1000} (GC_TX_OFFS=2)	868.3MHz European Band PHY Modes: BPSK-20 (GC_TX_OFFS=3), OQPSK-SIN-RC-{100,200,400} (GC_TX_OFFS=2) OQPSK-RC-{100,200,400} (GC_TX_OFFS=3)	780MHz Chinese Band PHY Modes: OQPSK-RC-{250,500,1000} (GC_TX_OFFS=2)
11	0xC0	0xA0	0xC1
10	0xC1	0x80	0xE3
9	0x80	0xE4	0xE4
8	0x82	0xE6	0xC5
7	0x83	0xE7	0xE7
6	0x84	0xE8	0xE8
5	0x40	0xE9	0xE9
4	0x86	0xEA	0xEA
3	0x00	0xCB	0xCB
2	0x01	0xCC	0xCC
1	0x02	0xCC	0xCD
0	0x03	0xAD	0xCE
-1	0x04	0x47	0xCF
-2	0x27	0x48	0xAF
-3	0x05	0x49	0x26
-4	0x07	0x29	0x27
-5	0x08	0x90	0x28
-6	0x91	0x91	0x29
-7	0x09	0x93	0x07
-8	0x0B	0x94	0x08
-9	0x0C	0x2F	0x09
-10	0x0D	0x30	0x0A
-11	0x0E	0x31	0x0B
-12	0x0F	0x0F	0x0C
-13	0x10	0x10	0x0D
-14	0x11	0x11	0x0E
-15	0x12	0x12	0x0F
-16	0x13	0x13	0x10
-17	0x14	0x14	0x11
-18	0x15	0x15	0x13
-19	0x16	0x17	0x14
-20	0x17	0x18	0x15
-21	0x19	0x19	0x16
-22	0x1A	0x1A	0x17
-23	0x1B	0x1B	0x18
-24	0x1C	0x1C	0x19
-25	0x1D	0x1D	0x1A

The firmware will configure GC_TX_OFFS, no need user to configure it.

For SAMR21, firmware only accept value in 0x0~0xf, means power from +4dbm to -17dbm, referring to below table. For other values input, it will report ERR.

Value	TX Output Power [dBm]
0x0	+4
0x1	+3.7
0x2	+3.4
0x3	+3
0x4	+2.5
0x5	+2
0x6	+1
0x7	0
0x8	-1
0x9	-2
0xA	-3
0xB	-4
0xC	-6
0xD	-8
0xE	-12
0xF	-17

Default setting is +3dbm for SAMR30, and +4dbm for SAMR21.

Example: cfg txpower 3 //configure to +3dbm for SAMR21

Response: AOK

PHY modulation or data rate setting will not be stored into PDS, user need to configure them every time when system restarts.

~cfg

Exit from configuration mode, thus it will enter action mode.

If reconnection setting is 0 or 1, it will automatically create or restore MiWi network, will not wait for user command "start" or join.

Example: ~cfg

Response: AOK

get ver

This command is to get firmware version. Firmware version will be returned as format that "cmd01fw01a". In this example, it means command version is 01 and firmware version is 01a. For the firmware version, the last character represents MiWi protocol, 'a' means SAMR30 P2P protocol firmware, 'b' means SAMR30 Star protocol firmware, 'c' means SAMR21 P2P protocol firmware, 'd' means SAMR21 Star protocol firmware.

Return format: **ver r1**

Example: get ver

Response: ver cmd01fw01a //command version 01, firmware version 01a

get addr

This command is to get device its own 64bits IEEE long address.

Return format: **addr r1**

Example: get addr

Response: addr b42aafd993ba01485 //device own address is 0xb42aafd993ba01485

get channel

This command is to get channel number configured for MiWi. The channel returned is an 8bit hex value, in range of 0 to 26.

Return format: channel r1

Default channel number is 8 for SAMR30 device and 26 for SAMR21.

Example: get channel

Response: channel 6 //return channel = 6

get pan

This command is to get PAND ID configured for MiWi. The PAND ID returned is a 16bits hex value.

Return format: pan r1

Default PAND ID is 0x1234.

Example: get pan

Response: pan 1234 //return PAN ID = 0x1234

get role

This command is to get role of this device in the MiWi network. This command is only accepted in action mode.

Returned is 2bits value. Bit 0 means who started the network, bit 1 means who is the PAN. Bit 1 is only valid in Star protocol. Details is shown below:

Bit 0: 0: joining network 1: starting network

Bit 1: 0: end device in Star network 1: PAN coordinator in Star network.

Return format: role r1

Example: get role

Response: role 01 //this device is the first device, who is starting the network.

get consize

This command is to get total of connection or saying how many devices are connected. This command is only accepted in action mode.

Returned is hex value, representing how many devices is connected.

Return format: consize r1

Use this command to get total of connection before using get conn r1 command to get connection detailed information.

Example: get consize

Response: consize 01 //1 peer device is connected.

get conn r1

This command is to get the connection status and peer device 64bits IEEE long address. This command is only accepted in action mode.

r1 is index of connection. This index number cannot be equal to or greater than the total connections returned by **get consize** command.

Return format: **conn r1 r2 r3**

r1 is the index of connection, it is equal to r1 of get command get conn r1

r2 is a boolean value. True means the connection is valid, false means the connection is invalid.

r3 is 64bits hex value representing peer device long address.

Example: get conn 0

Response: conn 01 9fc65cf9e2450591 //the connection is valid, and peer device long address is 0x9fc65cf9e2450591.

Note that whenever connection table is changed, device will initiatively send conn r1 r2 r3 to host MCU, without get conn r1 command necessarily sent.

get edsize

This command is to get total of end device table, which is periodically shared by PAN device. This command is only for MiWi Star end device, not available for Star PAN or P2P. This command is only accepted in action mode.

Returned is hex value, representing how many end devices is in the Star network, note that this number includes the device itself.

Return format: edsize r1

Before using get eds r1 r2 command to get end devices table information, need to use get edsize to get total of end device first.

Example: get edsize

Response: edsize 2 //two end device is in the network, including the device itself.

get myindex

This command is to get index number of itself. This command is only for MiWi Star end device, not available for Star PAN or P2P. The device itself is an end device in current Star network, get edsize command will also count itself, therefore, usually need to use get myindex to identify the index number belonging to itself. If host MCU want to send command to others by index, it will not send command to itself. This command is only accepted in action mode.

Returned is hex value, representing index number of itself in current Star network.

Return format: myindex r1

Use this command to get total of connection before using get conn r1 command to get connection detailed information.

Example: get myindex

Response: myindex 1 //index of itself is 1, index is starting from 0

get eds r1 r2

This command is to get end device table information, which is shared by its PAN device. This command is only for MiWi Star end device, not available for Star PAN or P2P. This command is only accepted in action mode.

r1 is the start index, device table information will be read and returned starting from this index.

r2 is the end index, device table information will be read and returned ending by this index.

Return format: **eds r1**

r1 is composed of a number of end device information unit, starting from specified starting index and end by specified ending index. Every information unit is composed by 4 bytes, it has this format as this aabbccdd, in which aabbcc is the higher 3 bytes of target 8bytes of IEEE long address, dd is the index number in the end device table.

aabbcc 3 bytes of address will be used in send command for end device X sending data to end device Y.

Example: get eds 0 1

Response: eds 45e680001be68001 //includes two end device information. First is indexed by 00, it has 3bytes higher address 0x45e680. Another is indexed by 01, it has 3bytes higher address 1be680.

Note: in above Response example, IEEE address of two end devices used in this command test is 0x45e680feff629106 and 0x1be680feff629106

start

This command is to call MiAPP API to start MiWi network. In Star protocol, who starts the network, he will be the PAN coordinator, and other devices will join his network.

Example: start

Response: AOK

ERR

join

This command is to call MiAPP API to establish network, which means join a exist network. In Star protocol, this device will be end device.

Example: join

Response: AOK

ERR

remove r1

This command is to call MiAPP API to remove one connection in the network. This command is only accepted in action mode.

r1 is index of connection. This index number cannot be equal to or greater than the total connections returned by **get consize** command.

Example: remove 0 //remove connection with index = 0

Response: AOK

ERR

send r1 r2 r3

This command is to send transparent data to peer device. It has many kinds of usage depending on r1 and r2.

r1: it can be a connection index value, or 0xffff as broadcast address, or 64bits IEEE long address indicating that who will receive the data.

If r1 is index value, it should be small than total connection number received by command **get consize**. In this case, data transmission type is unicast.

If r1 is “ffff” or “FFFF”, it will broadcast data to all connected device.

Send command also support data transmission from end device X to end device Y, which means the sender is end device in a Star network. In this transmission type, r1 is the first 3 bytes of end device Y’s long address. It’s first 3 bytes can be returned by **get eds r1 r2** command

If r1 is a 64bits hex value, it means data is transmitted to the device who has long address as this 64bits hex value. Data transmission type is unicast.

r2: it can be 0 or other non-zero value.

If r2 is 0, firmware will auto count the actual size of r3 string and send data out.

If r2 is non-zero, firmware will only send r2 size of data.

r3: the data to be sent out. With default code, r3 as user data has length limitation. Below shows user data length in 3 different transmission type:

For broadcast type, user data length limitation is 56 bytes, excluding ‘r’ character at command tail.

For PAN unicast to end devices, user data length limitation is 51 bytes, excluding ‘r’ character at command tail.

For end device X sending data to end device Y under Star network, user data length limitation is 47 bytes, excluding ‘r’ at command tail.

Example: send 0 0 hello //unicast send “hello” string to device whose connection index = 0

Response: AOK

ERR

Example: send 0 5 hello /unicast send 5bytes of “hello” string, to device whose connection index = 0

Response: AOK

ERR

Example: send b42aafd993ba0148 0 hello //unicast send “hello” string to device whose long address = b42aafd993ba0148

Response: AOK

ERR

Example: send ffff 0 hello //broadcast send “hello” string to all connected device

Response: AOK

ERR

Example: send 45e680 0 hello //end device X send data to end device Y, “45e680” is the first 3 bytes of end device Y’s long address, which is gotten from get eds r1 r2 command

Response: AOK

ERR

recv r1 r2 r3 r4

When MiWi device receives any data from peer device, it reports received data to host by this command. Other than received data, relative information including data security, broadcast or unicast type, RSSI value, source address will also be reported by this command.

r1: 2bits hex value, representing data type shown in below table.

r1	Meaning
0	Unicast type and unsecured.
1	Unicast type and secured.
2	Broadcast type and unsecured.
3	Broadcast type and secured.

r2: 8bits hex value representing RSS value.

r3: 64bits hex value representing source device 64bits IEEE long address. For unicast from PAN or broadcast, r3 simply means the source device’s address. However, for data receiving type that end device X sending data to end device Y, r3 doesn’t represents end device’s Y’s address, but represents its PAN’s address, because this case is forwarded from PAN.

r4: received data. For unicast from PAN or broadcast, r4 simply represents full data input at peer device’s send command. However, for data receiving type that end device X sending data to end device Y, r4 doesn’t purely represents user data which is input at send command. Under this receiving type, first 3 bytes of r4 represents end device X’s long address’s first 3 bytes.

Note that there is no connection index or received data size reported.

This command is initiated by MiWi device.

Example: recv 00 c4 b42aafd993ba01485 hello //received unicast&unsecured data, RSSI = 0xb4, source device address = 0x b42aafd993ba01485, data received is “hello”

status r1

This command is for MiWi device report some typical status change to host MCU.

r1: below table shows r1 value and its representation.

r1 value	Representation
1	Reconnection is successful, this is using old parameters stored in PDS memory to restore the network.
2	Reconnection is failed, or there is no reconnection enabled.
other	Invalid value or reserved for future.

error r1

This command is for MiWi device report some error code to host MCU.

r1: below is the table which shows r1 value and its representation, however, not all of values will be used, only several value is reported under certain situation, these situation majorly includes data transmission and start/join a network.

r1 value	Representation
1	Failure.
2	Channel access failure.
3	No ACK from receiver.
4	Time expired.
5	Transaction is expired.
6	End device is already in the network.
7	No route information.
8	Scanned but no beacon found.
9	Scanned and reached the maximum times.
10	Memory is not available.
11	Error with TX failure.
12	Error with TRX failure.
13	Error with invalid input.
14	Reconnection is in progress.
15	Reconnection is finished.
16	ADDR_NOT_FOUND_IN_SCANNED_LIST.
0xf0	ENTRY_NOT_EXIST
0xf1	No enough space available.
0xf2	Not the same PAN.
0xf3	NOT_PERMITTED
0xf4	ACTIVE_SCAN

reset

This command will erase PDS memory of SAMR30 or SAMR21, therefore those network parameters stored previously will be full erased. Next time after system is restarted, need to create full new network.

Example: reset

Response: AOK

Reboot

When system is restarted, firmware will first do some initialization work. After that, MiWi device will report “Reboot” to host. Host only can send command after “Reboot” is received.

Considering to have a good display when using PC terminal, a character ‘/r’ is attached after “Reboot” string.

AOK

MiWi device use this command to acknowledge any command sent by host, and it means the command sent by host is fine to be executed.

ERR

MiWi device use this command to acknowledge any command sent by host, and it means the command sent by host is not proper, maybe because of wrong command or wrong parameter.

5. COMMAND EXAMPLE

On one device side:

Return: Reboot

Type in: echo //enable echo

Return: AOK

Type in: cfg pan 5678 //configure PAN ID = 0x5678

Return: AOK

Type in: cfg channel 6 //configure channel = 6

Return: AOK

Type in: cfg reconn 0 //configure reconnection setting = 0

Return: AOK

Type in: ~cfg //exit from configuration mode, enter action mode

Type in: get addr //get local address

Return: addr b42aafd993ba0148 //return local address

Type in: get role //get the role

Return: role 01 //return role is starter

Type in: get consize //get total connection number

Return: consize 01 //return total connection number is 1

Type in: get conn 0 //get connection information by index = 0

Return: conn 1 9fc65cf9e2450591 //return connection information, including valid/invalid and peer address

Type in: send 9fc65cf9e2450591 0 hello //send data by unicast and by long address

Return: AOK

Type in: send 0 0 apple //send data by unicast and by index

Return: AOK

Type in: send ffff 0 how are you ? //send data by unicast and by broadcast

Return: AOK

On another device side:

Return: Reboot

Type in: echo//enable echo

Return: AOK

Type in: cfg pan 5678 //configure PAN ID = 0x5678

Return: AOK

Type in: cfg channel 6 //configure channel = 6

Return: AOK

Type in: cfg reconn 0 //configure reconnection setting = 0

Return: AOK

Type in: ~cfg //exit from configuration mode, enter action mode

Type in: get addr//get local address

Return: addr 9fc65cf9e2450591//return local address

Type in: get role //get the role

Return: role 00 //return role is joiner

Type in: get consize //get total connection number

Return: consize 01 //return total connection number is 1

Type in: get conn 0 //get connection information by index = 0

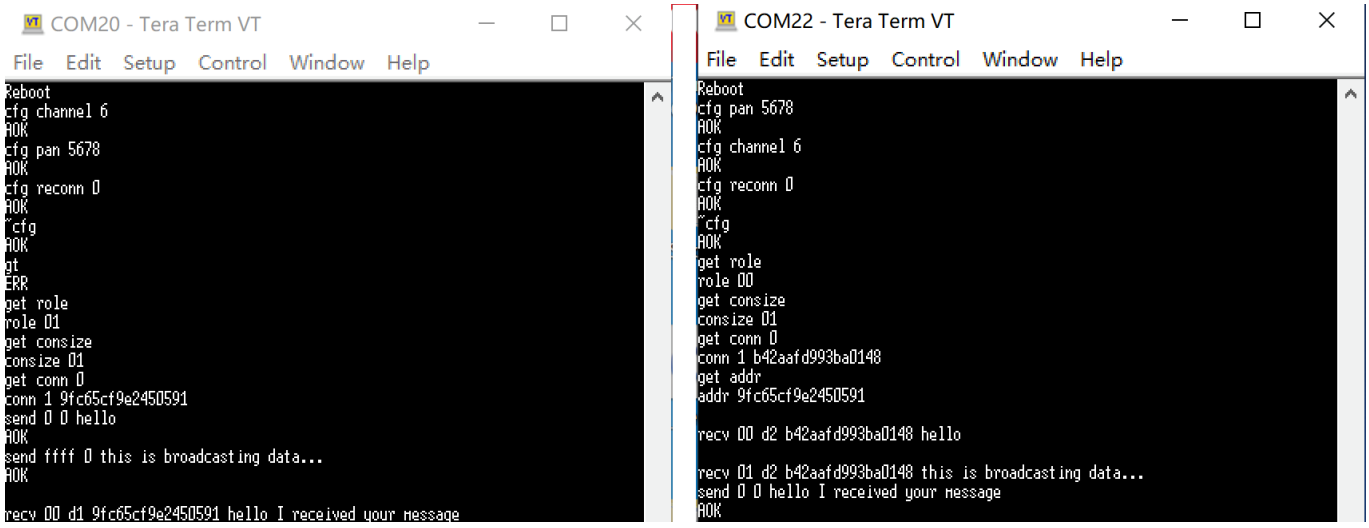
Return: conn 1 b42aafd993ba0148//return connection information, including valid/invalid and peer address

Return: recv 00 c4 b42aafd993ba01485 hello //received unicast data

Return: recv 00 c4 b42aafd993ba01485 apple //received unicast data

Return: recv 01 c4 b42aafd993ba01485 how are you //received broadcast data

This is a screenshot for actual command running on PC terminal.



6. VERSION HISTORY

VERSION	DATE	AUTHOR	COMMENT
V0.1	2021-2-1	Diffin Yang	Initial draft.
V0.2	2021-2-26	Diffin Yang	Simplified command, kept 17 command/events, removed all rest.
V0.3	2021-3-9	Diffin Yang	Modified some command to readable name, and created a command sequence example.
V0.4	2021-4-7 ~ 2021-4-13	Diffin Yang	Command change: 1. Added “echo”, “~echo” command 2. Modified “set channel” command to “cfg channel” 3. Removed “ret” tag from all receiving command 4. Renamed “network” command to “start” 5. Added “get channel”, “get pan” and their return command 6. Added “get consize”, “get role” and their return command 7. Modified “recv” command format to carry more useful information. 8. Modified “send” command to be more powerful and smart. Documents: 1. Added more detailed description in protocol basic chapter. 2. New added command details chapter. 3. Modified command example according to command change.
V0.5	2021-4-14	Diffin Yang	Command change: 1. New added “remove” command to work with “start” and “join”.
V0.6	2021-7-23 ~ 2021-8-9	Diffin Yang	Command change: 1. New added “get edsize”, “get myindex”, “get eds r1 r2” for end device to get shared connection table from PAN. 2. New added “cfg phymod”, “cfg txpower”, added them for configure SAMR30/SAMR21 PHY modulation or data rate and transmission power. 3. New added “status r1”, “error r1” to report error state and status change to host side. 4. Modified “conn r1 r2” to “conn r1 r2 r3”, added one more parameter for telling index, added to return this command runtime when connection is added or removed from connection table. 5. Added more description on “send” and “recv” to explain transmission type that end device X sending data to end device Y.

.