RF commands

3.1.1 rf tx <Data>

<Data>: a hexadecimal string representing data to be transmitted.

Response: there are two responses after entering this command. The first response used to indicate that whether command is valid or not, will be received after entering command. The second response will be received after transmitted.

Maximum transfer length: 255 Bytes.

First response: **Ok**, if <Data> string is valid.
Invalid, if <Data> string is not valid.

Second response: **radio_tx_ok**, if transmission is successful.
radio_err, if transmission is failed.

Example:

rf tx 5ab69f

>> Ok

>> radio_tx_ok

3.1.2 rf rx <RxWindowTime>

<RxWindowTime>: a decimal string representing receiving window in milliseconds, can be from **0** to **65535**. **0** means waiting until receiving a packet.

Response: there are two responses after entering this command. The first response, it used to indicate that whether command is valid or not, will be received after entering command. The second response will be received after received a packet or time out occurred.

First response: **Ok**, if <RxWindowTime> string is valid.
Invalid, if <RxWindowTime> string is not valid.

Second response: **radio_rx** <data> <rssi> <snr>, if reception is successful.
<data> - received data representing in hexadecimal.
<rssi> - received signal strength in decimal.
<snr> - received signal-to-noise value in decimal.
raido_err, if reception failed or time out occurred.

Example:

rf rx 1000

>> Ok

>> radio_rx 5432 -90 -50

3.1.3 rf set_freq <Frequency>

<Frequency>: a decimal string representing communication frequency in Hz, it can be values from **862000000** to **932000000** (S76S)**; 137000000** to **525000000** (S78S).

Response: **Ok**, if <Frequency> string is valid

Invalid, if <Frequency> string is not valid.

Set current communication frequency.

Example:

rf set_freq 915000000

>> Ok

3.1.4 rf set_pwr <Power>

<Power>: a decimal string representing transmitting power in dBm, it can be from **2** to **20**.

Response: **Ok**, if <Power> string is valid

Invalid, if <Power> string is not valid.

Set current transmitting power.

Example:

rf set_pwr 14

>> Ok

3.1.5 rf set_sf <SpreadingFactor>

<SpreadingFactor>: a string representing spreading factor used for communication, it can be: **7, 8, 9, 10, 11 and 12**.

Response: **Ok**, if <SpreadingFactor> string is valid

Invalid, if <SpreadingFactor> string is not valid.

Set current spreading factor.

Example:

rf set_sf 8

>> Ok

3.1.6 rf set_bw <BandWidth>

<BandWidth>: a string representing signal bandwidth in kHz, it can be: **125, 250, 500**.

Response: **Ok**, if <BandWidth> string is valid

Invalid, if <BandWidth> string is not valid.

Set current signal bandwidth.

Example:

rf set_bw 250

>> Ok

3.1.7 rf set_cr <CodingRate>

<CodingRate>: a string representing coding rate, can be: **4/5, 4/6, 4/7, 4/8**.

Response: **Ok**, if <CodingRate> string is valid

Invalid, if <CodingRate> string is not valid.

Set current coding rate used for communication.

Example:

rf set_cr 4/5

>> Ok

3.1.8 rf set_crc <State>

<State>: a string representing whether the CRC header is **on** or **off**.

Response: **Ok**, if <State> string is valid

Invalid, if <State> string is not valid.

Set current status of the CRC header.

Example:

rf set_crc on

>> Ok

3.1.9 rf set_sync <SyncWord>

<SyncWord>: a hexadecimal string representing sync word, it can be from **00** to **FF**.

Response: **Ok**, if <SyncWord> string is valid

Invalid, if <SyncWord> string is not valid.

Set the sync word used for communication.

Example:

rf set_sync 12

>> Ok

3.1.10 rf save

Response: **Ok**

Save p2p configuration parameters to EEPROM.

Example:

rf save

>> Ok

3.1.11 rf get_freq

Response: a decimal string representing communication frequency in Hz.

Return current communication frequency.

Returned string can be from **862000000** to **932000000** (S76S); **137000000 to 525000000** (S78S).

Example:

rf get_freq

>> 922500000

3.1.12 rf get_pwr

Response: a decimal string representing transmitting power in dBm.

Default: **14**

Return current transmitting power. Returned string can be from **2** to **20**.

Example:

rf get_pwr

>> 14

3.1.13 rf get_sf

Response: a string representing spreading factor used for communication.

Default: **7**

Return current spreading factor. Returned string can be: **7, 8, 9, 10, 11 and 12**.

Example:

rf get_sf

>> 7

3.1.14 rf get_bw

Response: a string representing signal bandwidth in kHz.

Default: **125**

Return current signal bandwidth. Returned string can be: **125, 250 and 500**.

Example:

rf get_bw

>> 125

3.1.15 rf get_prlen

Response: a decimal string representing preamble length.

Default: **12**

Return current preamble length. Returned strings can be from **0** to **65535**.

Example:

rf get_prlen

>> 12

3.1.16 rf get_cr

Response: a string representing current coding rate.

Default: **4/6**

Return current coding rate used for communication. Returned string can be: **4/5, 4/6, 4/7, 4/8**.

Example:

rf get_cr

>> 4/6

3.1.17 rf get_sync

Response: a hexadecimal string representing current sync word.

Default: **12**

Return current sync word used for communication.

Example:

rf get_sync

>> 12

3.1.18 rf rx_con <Continuous>

<Continuous>: a string representing whether Rx continuous mode is **on** or **off**.

Response: **Ok**, if < Continuous > string is valid

Invalid, if < Continuous > string is not valid.

Set Rx continuous mode can be **on** or **off**.

Example:

rf rx_con on

>> Ok

3.1.19 rf lora_tx_start <Times> <Interval> <Data>

<Times>: a decimal string representing how many time of TX counts, it can be values from **0** to **100000**, "0" means TX would not stop until "*rf lora_tx_stop*" send.

<Interval>: a decimal string representing LoRa TX interval in **ms**, it can be values from **3** to **300000**.

<Data>: a hexadecimal string representing data to be transmitted. The maximum transfer length: **255** bytes

Response: there are two responses after entering this command. The first response used to indicate that whether command is valid or not, it will be received after entering command. The second response will be received after every successful LoRa TX.

First response: **Ok**, if <Data> string is valid.
Invalid, if <Data> string is not valid.

Second response: **rf lora_tx(N)**, if transmission is successful and N means how many time TX already emits successfully.

Start LoRa TX after executing this command.

Example: (TX times is 100, TX interval is 200ms, TX data is 0xaa, 0xaa, 0x55, 0x55 that is 4 bytes)

rf lora_tx_start 100 200 aaaa5555

>> Ok

>> rf lora_tx(1)

>> rf lora_tx(10)

>> rf lora_tx(20)

…

Note: (rf lora_tx(n) would be only shown when n is 1, 10, 20, 30, …)

3.1.20 rf lora_tx_stop

First response: **rf lora_tx=N**, N means how many times TX had already emitted by using "*rf lora_tx_start*" command. If N is 0, it means no any successful TX or no any "*rf lora_tx_start*" ever sent.

Second Response: **Ok**, if command is correct and no any argument sees.
Invalid, if any argument is given.

Stop LoRa TX which started from "rf lora_tx_start".

Example:

rf lora_tx_stop

>> rf lora_tx=10

>> Ok

3.1.21 rf lora_rx_start <Data>

<Data>: a hexadecimal string representing that demands to be matched. Max length limitation is 255 bytes.

Response: there are two responses after entering this command. The first response, it used to indicate that whether command is valid or not, will be received after entering command. The second response will be received after received a packet.

First response: **Ok**, if <RxWindowTime> string is valid.
Invalid, if <RxWindowTime> string is not valid.
Second response: **rf lora_rx_start**(<num>) **rssi**(<rssi>) **snr**(<snr>), if reception is successful.
<num> - received data representing in hexadecimal.
<rssi> - received signal strength in decimal.
<snr> - received signal-to-noise value in decimal.

Example:
rf lora_rx_start aaaa5555
>> Ok
>> rf lora_rx(1) rssi(-96) snr(32)

>> rf lora_rx(10) rssi(-96) snr(30)

>> rf lora_rx(20) rssi(-97) snr(30)

Note: (rf lora_rx(n) would be only shown when n is 1, 10, 20, 30, …)

3.1.22 rf lora_rx_stop

First response: **rf lora_rx=N**, N means how many times RX had already received by using "*rf lora_rx_start*" command. If N is 0, it means no any successful RX (or has RX but payload data dis-matched) or no any "*rf lora_rx_start*" ever sent.
Second Response: **Ok**, if command is correct and no any argument sees.
Invalid, if any argument is given.

Stop LoRa RX which started from "rf lora_rx_start".

Example:

```
rf lora_rx_stop
>> rf lora_rx=51


>> Ok
```