# A brief description of the serial API of a Lora end device in UNSW’s Lora chain network

The MERE MIOT team provides a Lora chain network demo which includes a Lora repeater and a few sample end-devices. The end-devices can be controlled by a host via UART port, so that the host can communicate with other connected devices as well as a MQTT broker via the Lora chain network.

The baud rate of the UART port is 115200.

# Commands:

The supported commands are all in text format which include:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Command | Input para | Return Value | Return Code | Note |
| AT+SET\_ID | <Device\_ID>,<PAN\_ID> |  | OK | PANID is reserved\*\* |
| AT+RESET |  |  | OK |  |
| AT+JOIN |  |  | OK |  |
| AT+STATUS=? |  | IDLE/SEARCHING/TXRX/BUZY | OK |  |
| **AT+REJOIN** |  |  | OK |  |
| **AT+QUIT** |  |  | OK |  |
| AT+SEND= | <Destination>,<isBlockMode>,<Data>,<QoS > |  | OK or AT\_ERROR or AT\_ TIMEOUT or AT\_BUZY |  |
| AT+CHAN\_INFO= | <isBlockMode> | <<id>,<idx>,<rss>,<snr>,>x8 | OK or AT\_ERROR or AT\_ TIMEOUT or AT\_BUZY |  |
| +RECEIVE= |  | <Source>,<SourceRepeater>,<MessageType>,<Payload> |  | In coming message |
| **+EVT** |  | “CONNECTED”or “LOST CONNECTION” |  |  |
|  |  |  |  |  |

All commands and messages come with “\r\n”

\*The Destination , Source, Data and QoS&MessageType are all in hex string format.

\*\* The Device ID 0000,FFFF,FFFE,FFFD are reserved.

## 1.1 AT+SET\_ID=<Device\_ID>,<PAN\_ID>

This command is to set the device ID and PAN ID of the end device.

**Input：**

Device\_ID is the end device’s identifier, and it must be unique in the whole lora network.

PAN\_ID is reserved as the unique identifier of the network. Currently can be any two bytes.

**Returned value**: ”+RES= OK”

Note: these values are stored in flash and only need to be set only once.

**Example:**

AT+SET\_ID=0001,0000\r\n

\* From firmware V\_1\_2\_0, frequently setting a same ID to the device became safe and will not decrease the flash lifespan.

\* From firmware V\_1\_2\_0, when an end device is connected to a repeater, the connection remains even after a new ID is set.

## 1.2 AT+JOIN

This command is to activate the end device and the device will search for the existing repeaters and try to join the closest one.

**Returned value:** ”+RES= OK”

The returned value only means it start to join. It does not guarantee the joining. The joining process can take a few seconds or the device will keep searching if not a repeater is detected. You can check the status with command “AT+STATUS=?”

**\* From firmware V\_1\_2\_0 onward, the end device would keep the current connection and do nothing if it is connected to a repeater.**

**\* From firmware V\_1\_2\_0 onward, when the end device joins a repeater, it will print an event like ”+EVT= CONNECTED X” (X represents the index of the joined repeater). The host then does not have to repeatedly query the status. The end device will also report ”+EVT= LOST\_CONNECTION” when the connection is down (currently it means the end device has not received R2N message for a while).**

## 1.3 AT+REJOIN

This command would force the end device to search all the R2N channels and join the repeater with the strongest RSS.

**Returned value:** ”+RES= OK”

The returned value only means it start to join. It does not guarantee the joining. You can check the status with command “AT+STATUS=?”

## 1.4 AT+QUIT

This command would command the end device to quit from the joined repeater. It then cannot receive or transmit messages.

**Returned value:** ”+RES= OK”

## 1.5 AT+RESET

This command is to reset the device.

**Returned value**: ”+RES= OK”

Example:

A black text on a white background

Description automatically generated

## 1.6 AT+CHAN\_INFO=<isBlockMode>

This command is to request the channel information of all repeaters.

**Input:**

isBlockMode indicates whether this command will be executed in block mode or unblock mode.

Since the network can be busy sometimes, in block mode, the device will wait for 20s and try to obtain the channel information when the network is free. Otherwise, it will return ERROR immediately if the network is busy.

**Returned value:**

There are two returned values. The first returned value can be:

“+RES= OK”(which indicate the command is received)

or “+RES= COMMAND\_FORMAT\_ERROR” (which indicate the command format error)

If the first return value is “+RES= OK”, the second value will be returned in 0-20sec, which can be

“+RES= BUSY”

“+RES= ERROR”

“+RES= TIMEOUT”

Or "+RES=<id1>,<idx1>,<rss1>,<snr1>,<id2>,<idx2>,<rss2>,<snr2>,…<idn>,<idxn>,<rssn>,<snrn>”

There can be up to 8 sets of channel information.

In block mode, the end-devices will **stop responding** to new commands (including the reset command) until the second value returned.

Example:

A close-up of a computer screen

Description automatically generated

## 1.7 AT+SEND=<Destination>,<isBlockMode>,<Data>,<QoS >

This message is used to send data via the Lora chain network.

**Input:**

Destination is the device id of the message destination. The destination can be an End device a repeater or the server. If the destination is the server, set destination to FFFE.

isBlockMode indicates whether this command will be executed in block mode or unblock mode.

Data is the payload to send.

QoS indicates the how important is this message

“00” = Low priority with no remote ACK required

“02” = Low priority with first repeater’s ACK required but not no remote ACK

“04”= Medium priority with first repeater’s ACK required but not no remote ACK

“06” = Medium priority with both ACK required

“08” = High priority with both ACK required

“10” = Highest priority with both ACK required

Since the network can be busy sometimes, in block mode, the device will wait for 20s and try to obtain the channel information when the network is free. Otherwise, it will return ERROR immediately if the network is busy.

**Returned value:**

There are two returned values. The first returned value can be:

“+RES= OK”(which indicate the command is received)

or “+RES= COMMAND\_FORMAT\_ERROR” (which indicate the command format error)

If the first return value is “+RES= OK”, the second value will be returned in 0-20sec, which can be

“+RES= BUSY”

“+RES= ERROR”

“+RES= TIMEOUT”

Or “+STATUS= OK,<SEQ>” (Indicating the message has been sent to the repeater)

If the remote ACK is requested in the QoS setting, an asynchronized ACK is expected to be received in 0-5s. There can be multiple messages waiting ACK. The returned SEQ is used as the identifier of the message. The remote ACK will also carry the SEQ of the sent message so that the host would know which message is received.

In block mode, the end-devices will **stop responding** to new commands (including the reset command) until the second value returned.

Example:

A white background with black text

Description automatically generatedA screenshot of a computer

Description automatically generated

## +RECEIVE=<Source>,< SourceRepeater >,<MessageType>,<Payload>

**Input:**

Source is the device id of the message source.

SourceRepeater is the id of the first repeater that forward this message. (not valid right now)

Payload is the received payload or the sequence number of an ACK(used to indicate which sent message is received).

MessageType can be:

“1X” indicates this message is a common message that contains payload.

“5X” indicates this message is an ACK of a previous message and data is the sequence number.

Example:

A number on a white background

Description automatically generated

1. New features V\_1\_2\_0

**From firmware v\_1\_2\_0 onward, the end device will automatically try to join a repeater after powered up without a “AT+JOIN” command. It only stops when “AT+QUIT” received.**

**From firmware v\_1\_2\_0 onward, the end device could remember the last joined repeater index and perform a quick joining after hard/soft reset if that repeater exists. However, this only works if power supply is not interrupted. If the power is off, this information will be lost.**

**From firmware V\_1\_2\_0 onward, when the end device joins a repeater, it will print an event like ”+EVT= CONNECTED X” (X represents the index of the joined repeater). The host then does not have to repeatedly query the status. The end device will also report ”+EVT= LOST\_CONNECTION” when the connection is down (currently it means the end device has not received R2N message for a while).**

1. MQTT Lora message forwarder

The MQTT Lora message forwarder will work on a Lora repeater which may have Internet access. It is designed to forward messages between the assigned MQTT server and the Lora chain network.

It always converts MQTT Json messages to Lora messages (in bytes) for downstream data flow (server to end-devices) and converts lora messages to MQTT Json messages for upstream data flow (end-devices to server).

The Lora forwarder automatically subscribes to the downstream topic and publishes messages to the upstream topic.

Message transceiving at MQTT broker

The broker shall send message in the following json format:

It must contain “Destination”, "LoraQoS", "SessionId", "Payload" these four fields.

{

    "Destination": "0102",

    "LoraQoS": "15",

    "SessionId": "0405",

    "Payload": "01AA030405060708090A0B0C0D0E0F00"

}

Data in “Destination” must be a two bytes hex string indicating the ID of the destination device.

Data in "LoraQoS" is a one-byte hex string, the first half can only be “1” now. The second half can be  0 or 2 or 4 or 6 or 8 or 10 indicating how important this message is and whether an ACK is expected. (when >=4, ACK is expected)

LCN\_QOS\_LOW = 0,

LCN\_QOS\_LOW\_ACK = 2,

LCN\_QOS\_MED = 4,

LCN\_QOS\_MED\_2ACK = 6,

LCN\_QOS\_HIGH = 8,

LCN\_QOS\_ULTRA\_HIGH = 10,

Data in " SessionId" is a unique sequence number of the message which can prevent the message from being repeatedly sent. It can be generated based on an internal uint16 message counter of the broker. (The server can simply generate sessionId with an incremental counter.)

Data in "Payload" is the payload in hex string.

The broker may receive message in the following json format:

It must contain “Source”, “SrcRepeater”, "LoraQoS", "SessionId", "Payload" these five fields

{

"Source": "0102",

"SrcRepeater": "2000",

"LoraQos": "16",

"SessionId": "AABB",

"Payload": "1D1A1D1A"

}

Data in “Source” must be a two bytes hex string indicating the ID of the source device.

Data in “SrcRepeater” must be a two bytes hex string indicating the ID of the first repeater that forwards this message (source repeater), if this message is sent by a Repeater, source repeater would be itself.

Data in "LoraQoS" is a one-byte hex string, the first half indicates the message type. It can be “1” for common upstream message or “5” for an ACK of a previous downstream message. The second half currently can be ignored.

Data in " SessionId" can help broker to identify the echo messages. The message from the same source, with the same sessionId within a short period of time (say 1 min) can be ignored.

The repeater shall filter out echo messages. The broker may receive echo messages only when there are multiple repeaters in the same network have access to the internet.

Data in "Payload" is the payload in hex string.

The following figure shows an example of a downstream message with ACK required.



The following figure shows an example of a upstream message with ACK required. The MQTT broker do not need to take care of the ACK.

