Command line guide

Low power mobile data logger

ThingsLog LPMDL-110X Data logger

THINGSLOG

Table of Contents

Access to CLI console port

To access the command line interface you have to connect with an USBtoTTL cable to the logger console (UART) port.

To access the console port you have to connect to the 3 pins to the right of the input ports (in blue).
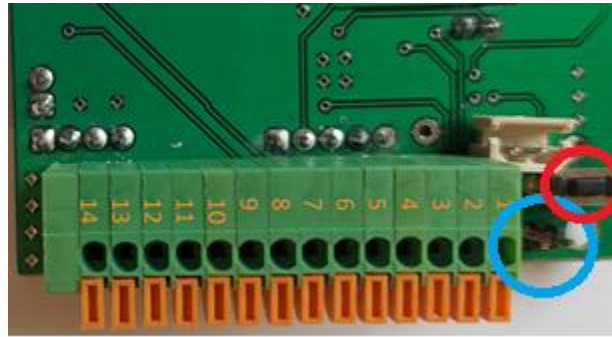
Figure 1 UART & service mode pins in LPMDL-110X

To put the device into service mode you need to press the button (red circle).

Console pins are left to right TXD RXD GND. If you are using ThingsLog console cable

- Black is GND
- Green is RXD
- White is TXD

Note: RX on the cable must be connected to TX on the board and RX on the cable must be connected with TX on the board.

Base service mode

There are different commands that can be used to configure the GPRS device when it is in service mode. In order to do this configuration it is necessary to put device in service mode using the service button.

Enter in Base service mode

1. Connect to the debug UART port using baud rate 57600
2. Press the service button
3. Verify that the following message appear in UART console: "service # ". This indicates that the device is in the base service mode

Example

```
service #
```

Store configuration

1. Verify you are in base service mode - the prompt is "service # "
2. Execute "store" command to save the configurations
3. Verify "service # " prompt appear

Example:

```
service # store
service #
```

Load configuration

1. Verify you are in base service mode - the prompt is "service # "
2. Execute `load` command to load the configurations
3. Verify "service # " prompt appear

Example:

```
service # load
service #
```

Get device number

1. Verify you are in base service mode - the prompt is "service # "
2. Execute `number` command to get the device number
3. Verify "number=00000001" hex device number appear

Example:

```
service # number
number=00000001
service #
```

Get firmware version

1. Verify you are in base service mode - the prompt is "service # "
2. Execute `version` command to get the firmware version
3. Verify "version=<some_version>" appear

Example:

```
service # version
version=0x0604bbc1
service #
```

Get date

1. Verify you are in base service mode - the prompt is "service # "
2. Execute `date` command to get the current date
3. Verify "date=<some_date>" appear

Example:

```
service # date
date=2017-04-01 12:40:00
service #
```

Set date

1. Verify you are in base service mode - the prompt is "service # "
2. Execute `set_d` command to set the device number
3. Verify "date=" prompt appear
4. Enter the new date in format YYYY-MM-dd HH:mm:ss
5. Verify "done" appear

Example:

```
service # set_d
Date=2017-04-01 12:40:00
Done
service #
```

Get current initial config

1. Verify you are in base service mode - the prompt is "service # "
2. Get current config by executing `configure`.

Example:

```
*** Config ***
2018-09-03 12:03:04
-----------
digits1=8
digits2=8
count_init1=1
count_init2=1
record_period=0
every=1
counts_threshold=300
sensors=01: CNT1
-----------
```

Description of configuration parameters

Name	Default Value	Possible values	Description
digits1	8	• Min value: 1 • Max value: 9	Total number of countable digits on the

			display of the meter for the first pulse input
digits2	8	• Min value: 1 • Max value: 9	Total number of countable digits on the display of the meter for the second pulse input
count_init1	1	Max value depends on the digits1 value	The value of the meter display of the first pulse input
count_init2	1	Max value depends on the digits1 value	The value of the meter display of the second pulse input
record_period	0	• 0 – MINUTES • 1 – HOURS	The value of the record period 0 mean minutes
every	1	• Min value: 1 • Max value: 127	The value in record period (1 - every 1 minute
counts_threshold	300	• Min value: 1 • Max value: 256	How many counters to keep in memory
sensors	01	Mask of 12 bits 0000000000000000 • 0 bit – Pulse sensor 1 • 1 bit – Pulse sensor 2 • 2 bit – analog sensor 1 • 3 bit – analog sensor 2 • 4 bit – On/Off sensor 1 • 5 bit – On/Off sensor 2 • 6 bit on/off output 1 • 7 bit on/off output 2 • 8 bit analog 3 • 9 bit analog 4 • 10 bit SEN5 – i2c sensor 1 • 11 bit SEN6 – i2c sensor 2 • 12 bit SEN7 – i2c sensor 3 • 13 bit SEN8 – i2c sensor 4 • 14 bit – future use • 16 bit – future use For example if you want to enable first pulse input, first analog and i2c sensor 1 and 2 that would resolve in: 0000 1100 0000 0101 Counting right to left (big endian stype): 0101 – first pulse and first analog 0000 – no on/offs are enabled 1100 – first 2 i2c are enabled 0000 - nothing else is enabled	Which input is active, 01 means the first pulse input

		So you have to set in set sensors: 0xC05 Hexadecimal number equal to the binary above!!!	

Set initial config

1. Verify you are in base service mode - the prompt is "service # "
2. Execute `set_conf` and enter the configuration menu

```
service # set_conf
config # help
commands:
 help
 set_count_init1
 set_count_init2
 set_digits1
 set_digits2
 set_rec_period
 set_every
 set_counts_threshold
 set_sensors
 save
 exit
```

3. Set inputs reading frequency. Execute `set_every`

```
config # set_every
set_every= 3
```

4. Enable sensor input port

```
config # set_sensors
sensors= 1
```

5. Set counts threshold (the number of counters to be gathered prior transmission attempt). Execute `set_counts_threshold` If the transmission is successful all counters will be deleted from memory. If transmission fails counters will be kept in memory and will be re-transmitted on next attempt. If memory got full the oldest counters will be deleted first.

```
config # set_counts_threshold
counts_threshold= 100
```

6. Set counter value. Has to be set all meaningful digits up to the magnet pointer. Execute `set_count_init1` for the first input and `set_count_init2` for the second.

```
config # set_count_init1
```

THINGSLOG

```
count_init1= 123
config # set_count_init2
count_init2= 234
```

7. Save and apply the configuration.

    Execute `save`

```
config # save
Save config
Applying config
Config Counters
rec_conf, size=102, rec_size=2, buff_size=206
Alarm enabled
service #
```

Setting up network identifiers

The logger could have different modem options: 2G, LoRa or NB-IoT.

Setting and troubleshooting 2G/GPRS service mode options

Set collecting server address and port
GSM data loggers are using by default TCP communication on port 4444.

1. Verify you are in gprs_service mode - the prompt is "gprs_service # "
2. Execute command `set_addr_port`
3. Verify "address=" prompt appear
4. Enter the new server address and press enter
5. Verify "port=" prompt appear
6. Enter the new server port and press enter
7. Execute `resolveIp` to set DNS name resolution 8 Verify that resolveIp= appears. Set it 1 to resolve address by DNS or 0 to disable resolving (address should be an IP address).
8. Verify "done" appear

Example:
```
service # extension
gprs_service # set_addr_port
address=10.10.10.10
port=4444
resolve_ip=1
done
gprs_service #
```

Get collecting server address and port

1. Verify you are in gprs_service mode - the prompt is "gprs_service # "
2. Execute command `get_addr_port`
3. Verify "address=<some_ip>" appear
4. Verify "port=<some_port>" appear

Example:

```
service # extension
gprs_service # get_addr_port
address=10.10.10.10
port=4444
done
gprs_service #
```

Set APN

1. Execute command `set_apn`
2. Verify "apn=" prompt appears
3. Enter the new APN and press enter
4. Verify "done" appear

Example:

```
service # extension
gprs_service # set_apn
apn=some.operator.apn
done
gprs_service #
```

- Get APN

1. Execute command `get_apn`
2. Verify "apn=some.operator.apn" appear

Example:

```
service # extension
gprs_service # get_apn
apn=some.operator.apn
done
gprs_service #
```

Set 2G network attachment timeout

Network attachment delay is the time for which the logger will wait to be recognized and paged by the mobile network.

1. Navigate to extension mode

2. Execute command `set_net_att_delay`
3. Verify "net_att_delay=some.delay " appears.
4. Enter the delay value in milliseconds (default is 35000 ms) e.g 35 seconds.
5. Don't forget to store the configuration

Example:

```
service # extension
gprs_service # set_net_att_delay
net_att_delay=some.delay
done
gprs_service #
```

Get 2G network attachment timeout

1. Navigate to extension mode
2. Execute command `get_net_att_delay`
3. Verify "net_att_delay=some.delay in ms" appears

Example:

```
service # extension
nbmodem_service # get_net_att_delay
net_att_delay=35000
done
```

Setting up LoRa/LoRaWAN service mode options

Set the logger to use ABP or OTAA

5. Verify you are in lora_service mode - the prompt is "lora_service # "
6. Execute command `set_use_abp`
7. Enter `1 or 0`, 1 – is ABP (app collect) and 0 is OTAA (app config)
8. Exit
9. Store the config

Get the logger to use ABP or OTAA

1. Verify you are in lora_service mode - the prompt is "lora_service # "
2. Execute command `get_use_abp`
3. 0, 1 – If setting is ABP and 0 is OTAA

Set the logger to send only the last full counter

10. Verify you are in lora_service mode - the prompt is "lora_service # "
11. Execute command `set_send_last_cnt`
12. Enter `1 or 0`, 1 – will send only the last one, 0 will send also delta 16 bit counters)
13. Exit
14. Store the config

Get the logger configuration to send only the last full counter

4. Verify you are in lora_service mode - the prompt is "lora_service # "
5. Execute command `set_send_last_cnt`
6. 0, 1 – If setting is 0 will send also differential counters, if 1 will send only the last 32 bit counter

Set configuration application

9. Verify you are in lora_service mode - the prompt is "lora_service # "
10. Execute command `set_app_config`
11. Verify " appkey=" prompt appear
12. Enter the new app key
13. Verify "appeui=" prompt appear
14. Enter the new appeui and press enter
15. Verify "deveui=" prompt appear
16. Enter the new deveui key
17. Verify "done" appear

Example:

```
lora_service # extension

set_app_config

lora_service # set_app_config

appkey=AAAAAAAAAAAAAAAAA

appeui=BBBBBBBBBBBBBBB

deveui=XXXXXXXXXXXXX
```

Get collecting config application

7. Verify you are in lora_service mode - the prompt is "lora_service # "
8. Execute command `get_app_config`
9. Verify "`appkey = AAAAAAAAAAAAAAAAA`" appears
10. Verify "`appeui = BBBBBBBBBBBBBBB`" appears
11. Verify "`deveui = XXXXXXXXXXXXX`" appears
12. Verify that done appears in the end

Example:

```
lora_service # get_app_config

appkey= AAAAAAAAAAAAAAAAA
```

```
appeui= BBBBBBBBBBBBBB

deveui= XXXXXXXXXXXXX

done
```

Set Collecting application

5. Execute command `set_app_collect`
6. Verify " appskey=" prompt appears
7. Enter the new APPKEY and press enter
8. Verify "nwkskey=" prompt appears
9. Enter the new NWKKEY and press enter
10. Verify that the "devaddr=" prompt appears
11. Enter the devaddr and press enter
12. Verify "done" appear

Example:

```
lora_service # set_app_collect
appskey=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
nwkskey=BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
devaddr=12345468
done
```

Get collecting app collect

3. Execute command `get_app_collect`
4. Verify "appkey= AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" appear
5. Verify "nwkskey= BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB" appear
6. Verify "devaddr=12345468" appear
7. Verify "done" appear

Example:

```
lora_service # get_app_collect[\n]

appskey= AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

nwkskey= BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB

devaddr=12345468

done
```

Setting up NB-IoT service mode options

Set collecting server address and port

1. Verify you are in gprs_service mode - the prompt is "gpr_service # "
2. Execute command `set_addr_port`
3. Verify "address=" prompt appear
4. Enter the new server address and press enter
5. Verify "port=" prompt appear
6. Enter the new server port and press enter
7. Execute `resolveIp` to set DNS name resolution 8 Verify that resolveIp= appears. Set it 1 to resolve address by DNS or 0 to disable resolving (address should be an IP address).
8. Verify "done" appear

Example:

```
service # extension
gprs_service # set_addr_port
address=10.10.10.10
port=4445
resolve_ip=1
done
gprs_service #
```

Get collecting server address and port

1. Verify you are in gprs_service mode - the prompt is "gpr_service # "
2. Execute command `get_addr_port`
3. Verify "address=<some_ip>" appear
4. Verify "port=<some_port>" appear

Example:

```
service # extension
gprs_service # get_addr_port
address=10.10.10.10
port=4445
done
gprs_service #
```

Set APN

1. Execute command `set_apn`
2. Verify "apn=" prompt appears
3. Enter the new APN and press enter
4. Verify "done" appear

Example:

```
service # extension
gprs_service # set_apn
apn=some.operator.apn
done
gprs_service #
```

Get APN

1. Execute command `get_apn`
2. Verify "apn=some.operator.apn" appear

Example:

```
service # extension
gprs_service # get_apn
apn=some.operator.apn
done
gprs_service #
```

Set NB-IoT network bands
The modem supports the following network bands

B1 @H-FDD: 2100MHz
B3 @H-FDD: 1800MHz
B8 @H-FDD: 900MHz
B5 @H-FDD: 850MHz
B20 @H-FDD: 800MHz
B28 @H-FDD: 700MHz

You can configure the network bands with the following CLI command. Note that you may configure more than one network band.

Example:

```
service # modem
nbmodem_service # set_bands
bands=8,28
nbmodem_service # exit
```

Get NB-IoT network bands

1. Navigate to modem
2. Execute command `get_bands`
3. Verify "bands=some.band" appears

Example:

```
nbmodem_service # get_bands
bands=28
done
nbmodem_service # exit
```

Set NB-IoT network attachment timeout

Network attachment delay is the time for which the logger will wait to be recognized and paged by the mobile network.

6. Navigate to extension mode
7. Execute command `set_net_att_delay`
8. Verify "net_att_delay=some.delay " appears.
9. Enter the delay value in milliseconds (default is 15000 ms) e.g 15 seconds.

Example:

```
service # extension
gprs_service # set_net_att_delay
net_att_delay=some.delay
done
gprs_service #
```

Get NB-IoT network attachment timeout

10. Navigate to extension mode
11. Execute command `get_net_att_delay`
12. Verify "net_att_delay=some.delay in ms" appears

Example:
```
service # extension
nbmodem_service # get_net_att_delay
net_att_delay=15000
done
```

Set MQTT parameters

Note: MQTT is available only in loggers with MQTT firmware!!!

To configure the mqtt username

1. Navigate to extension mode
2. Execute command set_mqtt
3. Verify that the username/password and root topic prompts appear.
4. All strings parameters be < 10 symbols in length

```
service # extension
```

```
nbmodem_service #set_mqtt
username=mqtt_user
password=mqtt_pass
root_topic=mqtt_topic
```

GET MQTT parameters

1. Navigate to extension mode
2. Execute command get_mqtt

```
service # extension
nbmodem_service #get_mqtt
username=mqtt_user
password=mqtt_pass
```
root_topic=mqtt_topic

Setup analog sensor settling time

Analog sensors values are obtained by 16 bit Analog to digital convertor (ADC).ADC reading time is a parameter that is valid only if you have a logger with analog ports. It should be equal or bigger than the analog sensor setting time. Setting time is the time required by the sensor placed in its particular environment to produce a correct measurement and output it on its analog port. The adc reading time is in ms. Sensors suitable for low power measurements have short adc read times (10-50ms). Setting ADC reading time to 0 means that the logger will not switch off the power supply to the sensor and will not measure its values in low power mode. This is the ideal mode for sensors on power.

1. Execute command `set_adc_read_time`
2. Verify "adc_read_time=" appears

```
ST=2018-09-03 11:50:56
WKUP 2018-09-03 11:50:56
Service mode triggered
Service
Type help for help
service # adc_read_time
adc_read_time: 400
service # set_adc_read_time
adc_read_time=2000
Done
service # adc_read_time
adc_read_time: 2000
service #
```

Note: The default ADC read time for LPMDL-110X is 50 ms. If you want to disable the logger to switch on/off the power supply of the logger you have to set the ADC read time to 0.

Example output

Successful data transmission – NB-IoT

```
>>> Modem ON[\r][\n]
>>> wait modem to start up[\r][\n]
>>> To Modem: AT[\r][\n]
>>> From Modem: [\xFFFFFFFF][\xFFFFFFFF][\r][\n]
>>> Boot: Unsigned[\r][\n]
>>> Security [\r][\n]
>>> OK[\r][\n]
>>> To Modem: AT+CMEE=1[\r][\n]
>>> From Modem: [\r][\n]
>>> OK[\r][\n]
>>> To Modem: ATE0[\r][\n]
>>> From Modem: [\r][\n]
>>> OK[\r][\n]
>>> To Modem:AT+NCONFIG=AUTOCONNECT,FALSE[\r][\n]
>>> From Modem: [\r][\n]
>>> OK[\r][\n]
>>> To Modem:AT+CPSMS=0[\r][\n]
>>> From Modem: [\r][\n]
>>> OK[\r][\n]
>>> To Modem: AT+CFUN=0[\r][\n]
>>> From Modem: [\r][\n]
>>> OK[\r][\n]
>>> To Modem: AT+CFUN?[\r][\n]
>>> Sys: CFUN->[\r][\n]
>>> +CFUN:0[\r][\n]
>>> [\r][\n]
>>> OK[\r][\n]
>>> [\r][\n]
>>> To Modem: AT+QCHIPINFO=VBAT[\r][\n]
>>> Sys: Voltage->[\r][\n]
>>> +QCHIPINFO:VBAT,2556[\r][\n]
>>> [\r][\n]
>>> To Modem: AT+CFUN=1[\r][\n]
>>> From Modem: [\r][\n]
>>> OK[\r][\n]
>>> To Modem: AT+CFUN?[\r][\n]
>>> Exp:*+CFUN:*[\r][\n]
>>> [\r][\n]
>>> OK[\r][\n]
>>> *[\r][\n]
>>> Buf:*[\r][\n]
```

```
>>> +CME ERROR: 51[\r][\n]
>>> [\r]*[\r][\n]
>>> To Modem: AT+CFUN?[\r][\n]
>>> Sys: CFUN->[\r][\n]
>>> +CFUN:1[\r][\n]
>>> [\r][\n]
>>> OK[\r][\n]
>>> [\r][\n]
>>> To Modem: AT+CGDCONT=1,"IP","iot-test"[\r][\n]
>>> From Modem: [\r][\n]
>>> OK[\r][\n]
>>> To Modem: AT+QIDNSCFG=8.8.8.8,4.2.2.2[\r][\n]
>>> From Modem: [\r][\n]
>>> OK[\r][\n]
>>> mdm: AT+COPS=0[\r][\n]
>>> From Modem: [\r][\n]
>>> OK[\r][\n]
>>> AT+CEREG=1[\r][\n]
>>> From Modem: [\r][\n]
>>> OK[\r][\n]
>>> AT+CEREG?[\r][\n]
>>> BUFF [\r][\n]
>>> +CEREG:1,2[\r][\n]
>>> [\r][\n]
>>> OK[\r][\n]
>>> [\r][\n]
>>> reg den check: 1[\r][\n]
>>> Reg trying[\r][\n]
>>> AT+CEREG?[\r][\n]
>>> BUFF [\r][\n]
>>> +CEREG:1[\r][\n]
>>> [\r][\n]
>>> +CEREG:1,1[\r][\n]
>>> [\r][\n]
>>> O[\r][\n]
>>> reg den check: 0[\r][\n]
>>> Reg ok[\r][\n]
>>> To Modem: AT+CGATT?[\r][\n]
>>> From Modem: [\r][\n]
>>> +CGATT:1[\r][\n]
>>> [\r][\n]
>>> OK[\r][\n]
>>> To Modem: AT+CSQ[\r][\n]
>>> Sys: Signal->[\r][\n]
>>> +CSQ:1,99[\r][\n]
>>> [\r][\n]
>>> OK[\r][\n]
>>> [\r][\n]
>>> rssi=1,ber=99[\r][\n]
>>> To Modem: AT+QDNS=0,iot.thingslog.com[\r][\n]
>>> ip_addr: 78.128.42.106[\r][\n]
>>> To Modem: AT+NSOCR=DGRAM,17,4587,1[\r][\n]
>>> Socket: 1[\r][\n]
>>> Sending cnt.len=30[\r][\n]
>>> To Modem: AT+NSOST=1,78.128.42.106,4445,56,FFFFFF2F040031[\r][\n]
>>> 1A06160108173500[\r][\n]
```

```
>>> 2A020163[\r][\n]
>>> 2B0209FC[\r][\n]
>>> 3F001EFrom Modem: [\r][\n]
>>> 1,56[\r][\n]
>>> [\r][\n]
>>> OK[\r][\n]
>>> From Modem: [\r][\n]
>>> +NSONMI:1,12[\r][\n]
>>> [\r][\n]
>>> To Modem: AT+NSORF=1,12[\r][\n]
>>> From Modem: [\r][\n]
>>> 1,78.128.42.106,4445,12,0000091A0716010817351C06,0[\r][\n]
>>> [\r][\n]
>>> OK[\r][\n]
>>> [\r][\n]
>>> Parsing data, rx_size: 12[\r][\n]
>>> Data: 00 00 09 1A 07 16 01 08 17 35 1C 06 [\r][\n]
>>> Abs time  : 2022-01-08 23:53:28[\r][\n]
>>> Curr time : 2022-01-08 23:53:28[\r][\n]
>>> Delta time: 0[\r][\n]
>>> To Modem: AT+NSOCL=1[\r][\n]
>>> From Modem: [\r][\n]
>>> OK[\r][\n]
>>> Modem OFF[\r][\n]
>>> Post counters success[\r][\n]
>>> ST=2022-01-08 23:53:28[\r][\n]
```

Successful configuration and data transmission – LoRa/LoRaWAN

In this scenario in the backend has been pressed "Start setup" the timer start ticking and the logger obtained its configuration successfully.

In **Bold** are key messages with **#** we put some comments.

```
>> WKUP 2020-04-19 06:17:00[\r][\n]
>>> [2]=0 0 0 0 [\r][\n]
>>> FIFO: f=0, r=7, r_size=8, size=288[\r][\n]
>>> FIFO: num=3, size=2, max_size=36[\r][\n]
>>> Lora send counters[\r][\n]
>>> Mdm start[\r][\n]
>>> From Modem: RN2903 1.0.5 Nov 06 2018 10:45:27[\r][\n]
>>> sys get vdd[\r][\n]
>>> Bat: 2778[\r][\r][\n]
>>> mac set appkey *[\r][\n]
>>> From Modem: ok[\r][\n]
>>> mac set appeui *[\r][\n]
>>> From Modem: ok[\r][\n]
>>> mac set deveui *[\r][\n]
>>> From Modem: ok[\r][\n]
>>> mac set linkchk 44[\r][\n]
>>> From Modem: ok[\r][\n]
>>> mac save[\r][\n]
>>> From Modem: ok[\r][\n]
>>> OTAA init[\r][\n]
>>> mac get status[\r][\n]
>>> buff=00000440[\r][\n]
```

```
>>> [\r][\n]
>>> mac join otaa[\r][\n]
>>> From Modem: ok[\r][\n]
>>> From Modem: accepted[\r][\n]
>>> mac tx uncnf 13 031504130611000ADA[\r][\n]
>>> From Modem: ok[\r][\n]
>>> Buff: mac_tx_ok[\r][\n]
>>>  Parsing data, rx_size: 0[\r][\n]
>>> Data: [\r][\n]
>>> mac tx uncnf 11
0315041306110001000100302031203120312031203120312031203120312031203120312
03120312000000000[\r][\n]
>>> From Modem: ok[\r][\n]
>>> Buff: mac_tx_ok[\r][\n]
>>>  Parsing data, rx_size: 0[\r][\n]
>>> Data: [\r][\n]
>>> mac tx uncnf 11
0315041306110002000100000000010001000100010001000100010001000100010001000
10001000100000000[\r][\n]
>>> From Modem: ok[\r][\n]
>>> Buff: mac_tx_ok[\r][\n]
>>>  Parsing data, rx_size: 0[\r][\n]
>>> Data: [\r][\n]
>>> mac tx uncnf 11
0315041306110003000101000000000000000000000000000000000000000000000000000
000000000[\r][\n]
>>> From Modem: ok[\r][\n]
>>> Buff: mac_tx_ok[\r][\n]
>>>  Parsing data, rx_size: 0[\r][\n]
>>> Data: [\r][\n]
>>> mac tx uncnf 11
0315041306110004000101000000000000000000000000000000000000000000000000000
000000000[\r][\n]
>>> From Modem: ok[\r][\n]
>>> Buff: mac_tx_ok[\r][\n]
>>>  Parsing data, rx_size: 0[\r][\n]
>>> Data: [\r][\n]
>>> Error 0[\r][\n]
>>> Lora tear down[\r][\n]
>>> Mdm off[\r][\n]
>>> Error=0[\r][\n]
```