AN4967

Application note

Examples of AT commands on I-CUBE-LRWAN

Introduction

I-CUBE-LRWAN is a LoRaWAN® Expansion Package for STM32Cube, consisting in a set of libraries and application examples for microcontrollers of the STM32L0, STM32L1 and STM32L4 Series acting as end devices.

The I-CUBE-LRWAN main features are:

- Easy add-on of the low-power LoRa® solution
- Extremely low CPU load
- No latency requirements
- Small STM32L0 Series memory footprint

This application note describes the set of AT commands for the B-L072Z-LRWAN1 Discovery kit embedding the CMWX1ZZABZ-091 LoRa® module.

This document explains how to interface with the LoRaWAN® to manage the LoRa® wireless link using AT commands.

1 General information

The I-CUBE-LRWAN applies to the STM32 microcontrollers that are Arm® Cortex® core-based devices.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Table 1. **Acronyms**

Acronym	Definition
ABP	Activation by personalization
ETSI	European telecommunications standards institute
LoRa	Long range radio technology
LoRaWAN	LoRa wide-area network
OTAA	Over-the-air activation
RF	Radio frequency
RSSI	Received signal strength indicator
SNR	Signal-to-noise ratio

Reference documents

[1] LoRaWAN 1.0.3 Specification by LoRa Alliance® Specification Protocol– 2018, January

[2] User manual *STM32 LoRaWAN® Expansion Package for STM32Cube* (UM2073)

2 Overview

The B-L072Z-LRWAN1 Discovery kit embeds the CMWX1ZZABZ-091 LoRa firmware.

This firmware implements the AT_Slave module (see document [2]) that supports a set of AT commands to drive the LoRaWAN communications and the LoRa RF test. It applies to microcontrollers of the STM32L0, STM32L1 and STM32L4 Series.

The following sections contain the interface description, the AT commands definition, and the description of some use cases and of the embedded software.

3 AT commands

The AT command set is a standard developed by “Hayes” to control modems. AT stands for attention.

The command set consists of a series of short text strings for performing operations such as joining, data exchange and parameters setting.

In a context of LoRa modem, the Hayes command set is a variation of the standard AT Hayes commands.

The AT commands are used to drive the LoRa module and to send data (refer to document [1]). The AT commands are sent through the UART.

As described in document [2], the LoRa modem can be controlled either through a terminal emulation like Tera Term or PuTTY (see Figure 1), or through an embedded AT master module (see Figure 2).

Figure 1. Terminal emulation mode

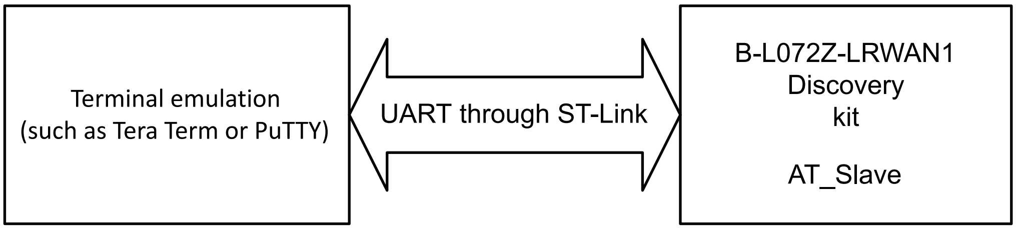
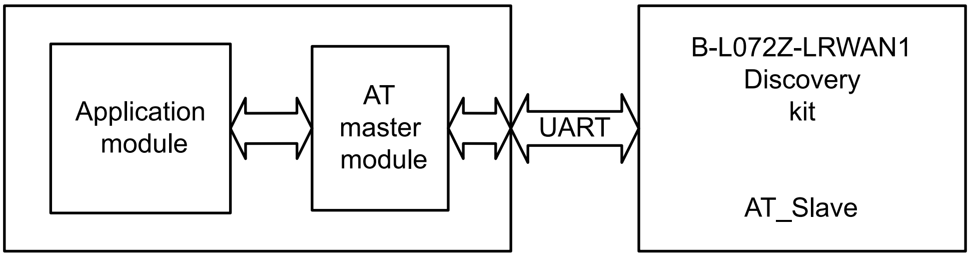


Figure 2. AT master mode



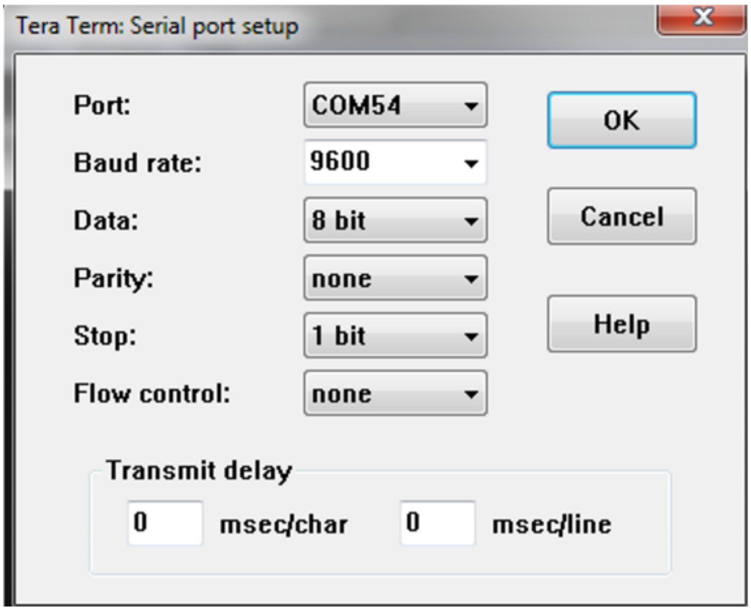
For illustration purposes, the rest of the document is based on the relation “terminal emulation” with the B-L072Z-LRWAN1 Discovery kit.

An UART over ST-LINK can then be used with standard Windows® software such as Tera Term or PuTTY. The chosen software has to be configured with the following parameters:

- Baud rate: 9600
- Data: 8 bits
- Parity: none
- Stop: 1 bit
- Flow control: none

The figures below show the standard configuration for Tera Term to use the UART over the ST-LINK.

Figure 3. **Tera Term serial port set up**



Tera Term: Serial port setup

Port: COM54

Baud rate: 9600

Data: 8 bit

Parity: none

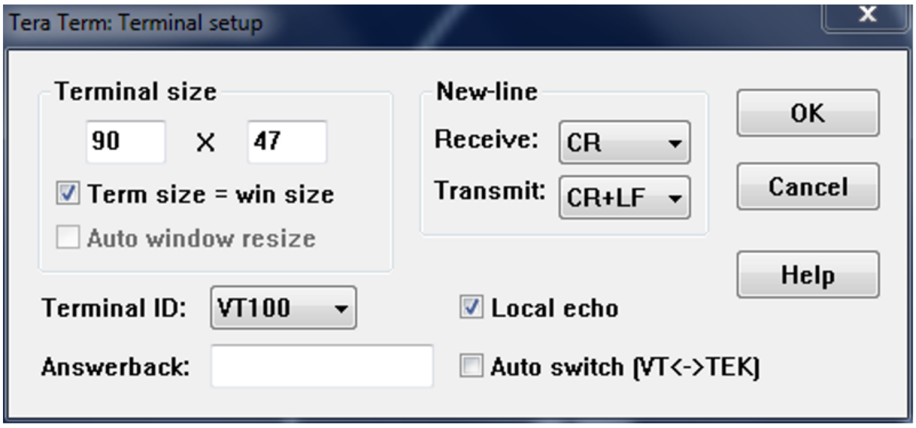
Stop: 1 bit

Flow control: none

Transmit delay

0 msec/char 0 msec/line

Figure 4. **Tera Term terminal setup**



Tera Term: Terminal setup

Terminal size

90 X 47

☑ Term size = win size

☐ Auto window resize

New-line

Receive: CR

Transmit: CR+LF

Terminal ID: VT100

Answerback:

☑ Local echo

☐ Auto switch (VT<->TEK)

All commands are of the form AT+XXX, with XXX denoting the command. The following command behaviors are available:

- AT+XXX? provides a short help of the given command (such as `AT+DEUI?`).
- AT+XXX is used to run a command (such as `AT+JOIN`).
- AT+XXX=? is used to get the value of a given command (such as `AT+CFS=?`).
- AT+XXX=<value> is used to provide a value to a command (such as `AT+SEND=2:Hello`).

Output of the commands is provided on the UART. The output format is typically:

```
<value><CR><LF>
<CR><LF><Status<CR><LF>
```

Considering:

- `<value><CR><LF>` is returned when help AT+XXX? and get AT+XXX=? commands are run.
- `<CR>` and `<LF>` stands for the carriage return and line feed.
- When no value is returned, then `<value><CR><LF>` is not returned at all.
- Every command, except `ATZ` (MCU reset), returns a status string, that is preceded and followed by `<CR><LF>`. Possible status are:
  - `OK`: command run correctly without error.
  - `AT_ERROR`: generic error
  - `AT_PARAM_ERROR`: parameter of the command is wrong.
  - `AT_BUSY_ERROR`: LoRa network is busy, so the command could not complete.
  - `AT_TEST_PARAM_OVERFLOW`: parameter is too long.
  - `AT_NO_NETWORK_JOINED`: LoRa network is not joined.
  - `AT_RX_ERROR`: error detection during the reception of the command

More details on each command description and examples are given in the next sections. Each command preceded by # is provided by the host to the module, then the return of the module is printed.

3.1 AT_RX_ERROR

In case of `AT_RX_ERROR`, the command is corrupted when received in AT_Slave. Hence the command is not run.

However, in case of long commands, some spurious characters can still be in the queue, ready to be processed as a command. So, in case the user receives an `AT_RX_ERROR`, the user must first send `<CR><LF>` to purge the queue, and then send back the same command so that it is processed.

Example

```
# AT+APPKEY=2b:7e:15:16:28:ae:d2:a6:ab:f7:15:88:09:cf:4f:3c<CR><LF>
<CR><LF>AT_RX_ERROR<CR><LF>  /* a RX error has been encountered */
<CR><LF>AT_ERROR<CR><LF> /* after the command, AT_Slave have processed “something” which is
not a command - that could result in an error */
# <CR><LF>  /* newline to purge */
<CR><LF>AT_ERROR<CR><LF> /* purge could result in an error */
/* now it is ok to resend the command */
# AT+APPKEY=2b:7e:15:16:28:ae:d2:a6:ab:f7:15:88:09:cf:4f:3c<CR><LF>
```

3.2 AT command overview

Table 2. **AT commands**

Command	Parameters	Description
General Commands		
AT	None	Check if the interface is available.
AT	[?]	Help of all supported commands.
ATZ	None	Reset
AT+VL	[=verb_lvl], where verb_lvl = [0:3]	Sets/gets the verbose level.
AT+LTIME	[=?]	Gets the local time in UTC format.
Keys, IDs and EUIs management commands		
AT+APPEUI	[=01:02:03:04:05:06:07:08]	Sets/gets the application EUI.
AT+NWKKEY	[=2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C]	Sets/gets the network root key
AT+APPKEY	[=2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C]	Sets/gets the application root key.

Command	Parameters	Description
AT+APPSKEY	[=2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C]	Sets/gets the application session key.
AT+NWKSKEY	[=2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C]	Sets/gets the network session key.
AT+DADDR	[=01:02:0A:0B]	Sets/gets the device address.
AT+DEUI	[=01:23:45:67:89:AB:CD:EF]	Sets/gets the module unique ID.
AT+NWKID	[=127]	Sets/gets the network ID.
LoRa join and send data commands		
AT+JOIN	[=mode] where mode = 0 (ABP) or mode = 1 (OTAA)	Joins the network.
AT+LINKC	-	Piggyback link check MAC command request to the next uplink
AT+SEND	[=port_nb:confirmedmode:data] where confirmedmode = 0 or 1.	Sends packets to the network.
LoRa network management commands		
AT+VER	[=?]	Gets the LoRaWAN version.
AT+ADR	[=adr_enable] where adr_enable = 0 or 1	Sets/gets the adaptive data rate functionality.
AT+DR	[=datarate] where datarate = [0:7]	Sets/gets the data rate.
AT+BAND	[=region] where region = [0:9]	Sets/gets the active region
AT+CLASS	[=class] where class = [A, B or C]	Sets/gets the LoRa class.
AT+DCS	[=dutycycle] where dutycycle = 0 or 1	Sets/gets duty cycle settings.
AT+JN1DL	[=delay] where delay in ms	Sets/gets the join delay on Rx window 1.
AT+JN2DL		Sets/gets the join delay on Rx window 2.
AT+RX1DL		Sets/gets the delay of the Rx window 1.
AT+RX2DL		Sets/gets the delay of the Rx window 2.
AT+RX2DR	[=datarate] where X = [0:7]	Sets/gets data rate of the Rx window 2.
AT+RX2FQ	[=freq] where freq in Hz	Sets/gets the frequency of the Rx window 2.
AT+TXP	[=txpow] where txpow = [0:7]	Sets/gets the transmit power.
AT+PGSLOT	[=periodicity]	Sets/gets the ping slot.
Radio tests commands		
AT+TTONE	None	Sets the RF tone test.
AT+TRSSI		Sets the RF RSSI tone test.
AT+TCONF	[=freq:pow:bw:sf:cr:lna:pa :mod:paylen:freqdev :lowdropt:BT] [=868000000:14:125:12:4/5:0:0:1:255:0:0:0 for example	Sets/gets the config LoRa RF test.
AT+TTX	[=nb_packets_sent]	Sets the number of packets to be sent for PER RF Tx test.
AT+TRX	[=nb_packets_received]	Sets the number of packets to be received for PER RF Rx test.

Command	Parameters	Description
AT+CERTIF	[=mode] where mode = 0 (ABP) or mode = 1 (OTAA)	Sets the module in LoRaWAN certification with join mode.
AT+TTH	[=<Fstart>, <Fstop>, <FDelta>,<PacketNb>]	Starts RF Tx hopping test from Fstart to Fstop (in Hz or MHz), Fdelta in Hz
AT+TOFF	None	Stops RF tests.
Information command		
AT+BAT	None	Gets the battery level.

3.3 Event table

The table below details the events that the AT_Slave application sends as a notification to the host module.

Table 3. **Event table**

Event	Return value	Description
+EVT:JOINED	None	Notifies an host module has been join on the gateway by OTAA.
+EVT:JOIN FAILED	None	Notifies the host module has not completed the join transaction (ID/Keys error, Tx not received by the gateway, Rx not received or not decrypted). In this case, the AT+JOIN must be recalled.
+EVT:	:<port>:<size>:<payload>	Notifies the host module that an asynchronous frame has been received on a RX window with downlink frame.
+EVT:	RX_<slot>:<DR>:<RSSI>:<SNR>	Notifies the host module that an asynchronous frame has been received on a RX window with downlink parameters.
+EVT:	RX_<slot>:<DR>:<RSSI>:<SNR>:<DMODM>:<GWN>	Notifies the host module that an asynchronous frame has been received on a RX window with extended downlink parameters. This event replaces the previous event when at least one link check request (AT+LINKC) has been executed.
+EVT:SEND_CONFIRMED	None	Notifies the host module that a Tx frame has been acknowledge by the gateway.

3.4 General commands

3.4.1 AT

Description	Attention is used to check if the link is working properly.
Syntax	AT<CR>
Arguments	None
Response	None
Result code	<CR><LF>OK<CR><LF>

Example:

```
/* Example: check the AT link is working properly*/
# AT<CR>
<CR>
OK<CR>
```

3.4.2 AT?

Description	Provides the short help of all supported commands.
Syntax	`AT?<CR>`
Arguments	None
Response	None
Result code	`<CR><LF>OK<CR><LF>`

Example:

```
/* Example: Get the short help of ALL AT commands*/
# AT?<CR>
AT+<CMD>?
AT+<CMD>         : Run <CMD>
AT+<CMD>=<value> : Set the value
AT+<CMD>=?       : Get the value
<List of all commands help>
<CR>
OK<CR>
```

3.4.3 ATZ - MCU reset

Description	The command generates a NVIC reset: resets the whole system including radio and microprocessor.
Syntax	`ATZ<CR>`
Arguments	None
Response	None
Result code	None (NVIC_Reset action)

Example:

```
/* Example: set NVIC system reset */
# ATZ<CR>
APP_VERSION:          V1.1.0<CR>
MW_LORAWAN_VERSION:   V2.3.0<CR>
MW_RADIO_VERSION:    V1.1.0<CR>
###### DevEui:  AA:BB:CC:DD:EE:FF:00:11<CR>
###### AppEui:  01:02:03:04:05:06:07:08<CR>
###### DevAddr: 12:34:56:78<CR>
ATtention command interface<CR>
AT? to list all available functions<CR>
```

Note: *The displayed keys by command above after* `######` *(*`DevEUI`*,* `AppEui`*, and* `DevAddr`*) are just informative and not a command response.*

3.4.4 AT+VL - Verbose level

Description	Sets/gets the verbose level of the application.
Syntax	AT+VL=<verbose_leve><CR> AT+VL=?<CR>
Arguments	<verbose_level>, the default is 2 (VLEVEL_M) 0: VLEVEL_OFF 1: VLEVEL_L 2: VLEVEL_M 3: VLEVEL_H
Response	<verbose_level><CR><LF>
Result code	<CR><LF>OK<CR><LF> <CR><LF>AT_PARAM_ERROR<CR><LF>

Examples:

```
/* Example1: set verbose level */
# AT+VL=3<CR>
<CR>
OK<CR>

/* Example2: get verbose level */
# AT+VL =?<CR>
3<CR>
<CR>
OK<CR>
```

3.4.5 AT+LTIME - Local time in UTC format

Description	Gets the local time in UTC format.
Syntax	AT+LTIME=?<CR>
Arguments	None
Response	<local time><CR><LF>
Result code	<CR><LF>OK<CR><LF>

Example:

```
/* Example: Get the local time in UTC format */
#AT+ LTIME =?<CR>
LTIME:02h14m52s on 01/01/1970<CR>
<CR>
OK<CR> /* module returns the command error code */
```

3.5 Keys, IDs and EUIs management

3.5.1 AT+APPEUI - Application identifier

Description	Sets/gets the application EUI.
Syntax	AT+APPEUI=<id><CR> AT+APPEUI=?<CR>
Arguments	<id>, 8-byte value separated by ":" (hexadecimal format string)
Response	<id><CR><LF>
Result code	<CR><LF>OK<CR><LF> <CR><LF>AT_ERROR<CR><LF> <CR><LF>AT_PARAM_ERROR<CR><LF>

Examples:

```
/* Example1: set APP EUI */
# AT+APPEUI=01:02:03:04:05:06:07:08<CR>
<CR>
OK<CR>

/* Example2: get APP EUI */
# AT+APPEUI=?<CR>
01:02:03:04:05:06:07:08<CR>
<CR>
OK<CR>
```

3.5.2 AT+NWKKEY - Network root key

Description	Sets/gets the network root key. This key is used only in OTAA mode.
Syntax	AT+NWKKEY=<key><CR> AT+NWKKEY=?<CR>
Arguments	<id>, 4-byte value separated by ":" (hexadecimal format string)
Response	<key><CR><LF>
Result code	<CR><LF>OK<CR><LF> <CR><LF>AT_ERROR<CR><LF> <CR><LF>AT_PARAM_ERROR<CR><LF>

Examples:

```
/* Example1: set NWK Key */
# AT+NWKKEY=2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C<CR>
<CR>
OK<CR>

/* Example2: get NWK Key when #define KEY_EXTRACTABLE 1 */
# AT+NWKKEY=?<CR>
2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C<CR>
<CR>
OK<CR>

/* Example3: get NWK Key when #define KEY_EXTRACTABLE 0 */
# AT+NWKKEY=?<CR>
<CR>
AT_ERROR<CR>
```

3.5.3 AT+APPKEY - Application root key

Description	Sets/gets the application root key. This key is used only in OTAA mode.
Syntax	AT+APPKEY=<key><CR> AT+APPKEY=?<CR>
Arguments	<key>, 16-byte value separated by ”:” (hexadecimal format string)
Response	<key><CR><LF>
Result code	<CR><LF>OK<CR><LF> <CR><LF>AT_ERROR<CR><LF> <CR><LF>AT_PARAM_ERROR<CR><LF>

Examples:

```
/* Example1: set APP Key */
# AT+APPKEY=2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C<CR>
<CR>
OK<CR>

/* Example2: get APP Key when #define KEY_EXTRACTABLE 1 */
# AT+APPKEY=?<CR>
2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C<CR>
<CR>
OK<CR>

/* Example3: get APP Key when #define KEY_EXTRACTABLE 0 */
# AT+APPKEY=?<CR>
<CR>
AT_ERROR<CR>
```

3.5.4 AT+APPSKEY - Application session key

Description	Sets/gets the application session key. This key is used only in OTAA and APB modes. In OTAA mode, this key is replaced during the derivation process with the application root key and `JoinAccept` response information.

Syntax	`AT+APPSKEY=<key><CR>`
	`AT+APPSKEY=?<CR>`

Arguments	`<key>`, 16-byte value separated by ”:” (hexadecimal format string)
Response	`<key><CR><LF>`

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_ERROR<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`

Example:

```
/* Example1: set APP Session Key */
# AT+APPSKEY=2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C<CR>
<CR>
OK<CR>

/* Example2: get APP Session Key when #define KEY_EXTRACTABLE 1 */
# AT+APPSKEY=?<CR>
2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C<CR>
<CR>
OK<CR>

/* Example3: get APP Session Key when #define KEY_EXTRACTABLE 0 */
# AT+APPSKEY=?<CR>
<CR>
AT_ERROR<CR>
```

3.5.5 AT+NWKSKEY - Network session key

Description	Sets/gets the network session key. This key is used in OTAA and ABP modes. In OTAA mode, this key is replaced during the derivation process with the network's root key and `JoinAccept` response information.

Syntax	`AT+NWKSKEY=<key><CR>`
	`AT+NWKSEY=?<CR>`

Arguments	`<key>`, 16-byte value separated by ”:” (hexadecimal format string)
Response	`<key><CR><LF>`

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_ERROR<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`

Example:

```
/* Example1: set NWK Session Key */
# AT+NWKSKEY=2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C<CR>
<CR>
OK<CR>

/* Example2: get NWK Session Key when #define KEY_EXTRACTABLE 1 */
# AT+NWKSKEY=?<CR>
2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C<CR>
<CR>
OK<CR>

/* Example3: get NWK Session Key when #define KEY_EXTRACTABLE 0 */
# AT+NWKSKEY=?<CR>
<CR>
AT_ERROR<CR>
```

3.5.6 AT+DADDR - Device address

Description	Sets/gets the device address.

Syntax	`AT+DADDR=<address><CR>`
	`AT+DADDR=?<CR>`

Arguments	`<address>`, 4-byte value separated by ”:” (hexadecimal format string)
Response	`<address><CR><LF>`

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_ERROR<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`

Examples:

```
/* Example1: set device address*/
# AT+DADDR=01:02:0A:0B<CR>
<CR>
OK<CR>

/* Example2: get device address*/
# AT+DADDR=?<CR>
01:02:0A:0B<CR>
<CR>
OK<CR>
```

3.5.7 AT+DEUI - Device EUI

Description	Sets/gets the device EUI.
Syntax	AT+DEUI=<EUI><CR> AT+DEUI=?<CR>
Arguments	<EUI>, 8-byte value separated by ”:” (hexadecimal format string)
Response	<EUI><CR><LF>
Result code	<CR><LF>OK<CR><LF> <CR><LF>AT_ERROR<CR><LF> <CR><LF>AT_PARAM_ERROR<CR><LF>

Examples:

```
/* Example1: set device EUI*/
# AT+DEUI=01:02:03:04:05:06:07:08<CR>
<CR>
OK<CR>

/* Example2: get device EUI */
# AT+DEUI=?<CR>
01:02:03:04:05:06:07:08<CR>
<CR>
OK<CR>
```

3.5.8 AT+NWKID - Network ID

Description	Sets/gets the network ID.
Syntax	AT+NWKID=<id><CR> AT+NWKID=?<CR>
Arguments	<id>, 1-byte decimal value from 0 to 127
Response	<id><CR><LF>
Result code	<CR><LF>OK<CR><LF> <CR><LF>AT_ERROR<CR><LF> <CR><LF>AT_PARAM_ERROR<CR><LF>

Examples:

```
/* Example1: set the network ID */
# AT+NWKID=127<CR>
<CR>
OK<CR>

/* Example2: get the network ID */
# AT+NWKID=?<CR>
127<CR>
<CR>
OK<CR>
```

3.6 Join and send data on LoRa network

3.6.1 AT+JOIN - Join LoRa network

Description	Join the LoRa network.
Syntax	`AT+JOIN=<mode><CR>`
Arguments	`<mode>` 0: join to a network by ABP 1: join to a network by OTAA
Response	`+EVT:JOINED` or `+EVT:JOIN_FAILED`

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`

Examples:

```
/* Example1: Join a network by ABP */
#AT+JOIN=0<CR>
+EVT:JOINED<CR>   /* event: ABP configuration done. Ready to start Tx */
<CR>
OK<CR>

/* Example2: Join a network by OTAA (Success result) */
#AT+JOIN=1<CR>
<CR>
OK<CR>

+EVT:JOINED<CR>   /* Event : OTAA join successful event */

/* Example3: Join a network by OTAA (Fail result) */
#AT+JOIN=1<CR>
<CR>
OK<CR>

+EVT:JOIN FAILED<CR> /* Event : OTAA join failed event. LoRaWAN network offline or keys not
                        aligned with the network configuration */
```

3.6.2 AT+LINKC - Link check request

Description	Piggyback link check MAC command request to the next uplink. The `DemodMargin` and `NbGateways` output information is provided into the extended Rx events +EVT:RX.
Syntax	`AT+LINKC<CR>`
Arguments	None
Response	None

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`

Examples:

```
/* Example: Piggyback Link Check Request to the next uplink */
#AT+LINKC<CR>
<CR>
OK<CR>
```

3.6.3 AT+SEND - Send data to LoRa network

Description	Sends application packets with specified and AppPort and payload to LoRaWAN network.
Syntax	`AT+SEND=<port>:<ack>:<payload><CR>`
Arguments	• `<port>`: application port to be transmitted • `<ack>` – 0: unconfirmed message – 1: confirmed message • `<payload>`: payload in hexadecimal format strings (maximum length is 242 bytes)
Response	`+EVT:SEND_CONFIRMED`

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`
	`<CR><LF>AT_DUTYCYLE_RESTRICTED<CR><LF>`
	`<CR><LF>AT_NO_NET_JOINED<CR><LF>`
	`<CR><LF>AT_BUSY_ERROR<CR><LF>`
	`<CR><LF>AT_CRYPTO_ERROR<CR><LF>`
	`<CR><LF>AT_ERROR<CR><LF>`

Examples:

```
/* Example1: Send a packet to the gateway in unconfirmed mode */
#AT+SEND=2:0:ABCD<CR>/* send a packet : "ABCD", with APP port is 2, unconfirmed message */
<CR>
OK<CR>

/* Example2: Send a packet to the gateway in confirmed mode */
# AT+SEND=10:1:7FFF<CR>/* send a packet : "7FFF", with APP port is 10, confirmed message */
<CR>
OK<CR>

+EVT:SEND_CONFIRMED
```

3.7 LoRa network management

3.7.1 AT+VER - Firmware version

Description	Gets the version of the AT_Slave firmware.

Syntax	`APP_VERSION: Vx.y.z<CR><LF>`
	`MW_LORAWAN_VERSION: Va.b.c<CR><LF>`
	`MW_RADION_VERSION: Vd.e.f<CR><LF>`

Arguments	None
Response	`<version><CR><LF>`
Result code	`<CR><LF>OK<CR><LF>`

Example:

```
/* Example: Get the Application and Middleware versions */
#AT+VER=?
APP_VERSION:          V1.1.0<CR>
MW_LORAWAN_VERSION:   V2.3.0<CR>
MW_RADION_VERSION:    V1.1.0<CR>
<CR>
OK<CR>
```

3.7.2 AT+ADR - Adaptive data rate functionality

Description	Sets/gets the adaptive data rate functionality.

Syntax	`AT+ADR=<enabled><CR>`
	`AT+ADR=?<CR>`

Arguments	`<enabled>` • 0: ADR disabled • 1: ADR enabled (default)
Response	`<enabled><CR><LF>`

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`

Examples:

```
/* Example1: Disable ADR */
#AT+ADR=0<CR>/* Disable ADR*/
<CR>
OK<CR> /* module returns the command error code */

/* Example2: Check ADR status */
# AT+ADR=?<CR>
0<CR>             /* module returns ADR status */
<CR>
OK<CR>/* module returns the command error code */
```

3.7.3 AT+DR - Data rate

Description	Sets/gets the Tx data rate.

Syntax	`AT+DR=<data rate><CR>`
	`AT+DR=?<CR>`

Arguments	`<data rate>` in the range [0,1,2,3,4,5,6,7]
Response	`<data rate><CR><LF>`

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_ERROR<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`

Note: *To be able to set data rate, the ADR must be disabled.*

Examples:

```
/* Example1: Set TX Data Rate */
#AT+DR=2<CR>/* Set TX Data Rate */
<CR>
OK<CR> /* module returns the command error code */

/* Example2: Get Data rate with Adaptive DataRate disabled */
#AT+ADR=?<CR>
0<CR>
<CR>
OK<CR>
# AT+DR=?<CR>
2<CR> /* module returns TX data rate */
<CR>
OK<CR>

/* Example3: Get Data rate with Adaptive DataRate enabled  */
#AT+ADR=?<CR>
1<CR>
<CR>
OK<CR>
# AT+DR=?<CR>
<CR>
AT_ERROR<CR>
```

3.7.4 AT+BAND - Active region

Description	Sets/gets the active region.
Syntax	AT+BAND=<band><CR> AT+BAND=?<CR>
Arguments	<band>: number corresponding to active regions 0: AS923 1: AU915 2: CN470 3: CN779 4: EU433 5: EU868 6: KR920 7: IN865 8: US915 9: RU864
Response	<band><CR><LF>
Result code	<CR><LF>OK<CR><LF> <CR><LF>AT_PARAM_ERROR<CR><LF>

Examples:

```
/* Example1: Set Active region */
#AT+BAND=0<CR>/* Set AS923 as active region*/
<CR>
OK<CR> /* module returns the command error code */

/* Example2: Get Active region */
# AT+BAND=?<CR>
5:EU868<CR>             /* module returns Active region */
<CR>
OK<CR> /* module returns the command error code */
```

3.7.5 AT+CLASS - LoRa class

Description	Sets/gets the LoRa class.

Syntax	`AT+CLASS=<class><CR>`
	`AT+CLASS=?<CR>`

Arguments	`<class>`: must be A, B or C.
Response	`<class><CR><LF>`

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_ERROR<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`
	`<CR><LF>AT_NO_CLASS_B_ENABLE<CR><LF>`
	`<CR><LF>AT_NO_NET_JOINED<CR><LF>`

Examples:

```
/* Example1: Set the LoRa Class */
#AT+CLASS=C<CR>/* Set Class C on device */
<CR>
OK<CR> /* module returns the command error code */

/* Example2: Get the LoRa Class */
# AT+CLASS=?<CR>
C<CR>            /* module returns Active Class */
<CR>
OK<CR> /* module returns the command error code */
```

3.7.6 AT+DCS - Duty cycle settings

Description	Sets/gets the duty cycle settings.

Syntax	`AT+DCS=<dutyCycleEnable><CR>`
	`AT+DCS=?<CR>`

Arguments	`<dutyCycleEnable>` 0: duty cycle disabled 1: duty cycle enabled
Response	`<dutyCycleEnable><CR><LF>`

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`

Examples:

```
/* Example1: Enable Duty cycle */
#AT+DCS=1<CR>
<CR>
OK<CR> /* module returns the command error code */

/* Example2: Get Duty cycle */
# AT+DCS=?<CR>
1<CR>             /* module returns Duty cycle */
<CR>
OK<CR> /* module returns the command error code */
```

3.7.7 AT+JN1DL - Join delay on Rx window 1

Description	Sets/gets the join accept delay between the end of the Tx and the join Rx window 1 (in ms).
Syntax	AT+JN1DL=<delay><CR> AT+JN1DL=?<CR>
Arguments	<delay>: value in ms
Response	<delay><CR><LF>
Result code	<CR><LF>OK<CR><LF> <CR><LF>AT_PARAM_ERROR<CR><LF>

Examples:

```
/* Example1: Set Join Delay on RX window 1*/
#AT+JN1DL=5000<CR>
<CR>
OK<CR> /* module returns the command error code */

/* Example2: Get Join Delay on RX window 1*/
# AT+JN1DL=?<CR>
5000<CR>        /* module returns Join Delay on RX window 1 in ms*/
<CR>
OK<CR> /* module returns the command error code */
```

3.7.8 AT+JN2DL - Join delay on Rx window 2

Description	Sets/gets the join accept delay between the end of the Tx and the join Rx window 2 (in ms).
Syntax	AT+JN2DL=<delay><CR> AT+JN2DL=?<CR>
Arguments	<delay>: value in ms
Response	<delay><CR><LF>
Result code	<CR><LF>OK<CR><LF> <CR><LF>AT_PARAM_ERROR<CR><LF>

Examples:

```
/* Example1: Set Join Delay on RX window 2*/
#AT+JN2DL=8000<CR>
<CR>
OK<CR> /* module returns the command error code */

/* Example2: Get Join Delay on RX window 2*/
# AT+JN2DL=?<CR>
8000<CR>        /* module returns Join Delay on RX window 2 in ms*/
<CR>
OK<CR> /* module returns the command error code */
```

3.7.9 AT+RX1DL - Delay of the Rx window 1

Description	Sets/gets the delay between the end of the Tx and the Rx window 1 (in ms).

Syntax	`AT+RX1DL=<delay><CR>`
	`AT+RX1DL=?<CR>`

Arguments	`<delay>`: value in ms
Response	`<delay><CR><LF>`

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`

Examples:

```
/* Example1: Set Delay on RX window 1*/
#AT+RX1DL=1500<CR>
<CR>
OK<CR> /* module returns the command error code */

/* Example2: Get Delay on RX window 1*/
# AT+RX1DL=?<CR>
1500<CR>        /* module returns Delay on RX window 1 in ms*/
<CR>
OK<CR> /* module returns the command error code */
```

3.7.10 AT+RX2DL - Delay of the Rx window 2

Description	Sets/gets the delay between the end of the Tx and the Rx window 2 (in ms).

Syntax	`AT+RX2DL=<delay><CR>`
	`AT+RX2DL=?<CR>`

Arguments	`<delay>`: value in ms
Response	`<delay><CR><LF>`

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`

Examples:

```
/* Example1: Set Delay on RX window 2*/
#AT+RX2DL=2500<CR>
<CR>
OK<CR> /* module returns the command error code */

/* Example2: Get delay on RX window 2*/
# AT+RX2DL=?<CR>
2500<CR>        /* module returns Delay on RX window 2 in ms*/
<CR>
OK<CR> /* module returns the command error code */
```

3.7.11 AT+RX2DR - Data rate of the Rx window 2

Description	Sets/gets the Rx window 2 data rate (0-7 corresponding to DR_X).
Syntax	AT+RX2DR=<datarate><CR> AT+RX2DR=?<CR>
Arguments	<datarate>: value in range [0:15]
Response	<datarate><CR><LF>
Result code	<CR><LF>OK<CR><LF> <CR><LF>AT_PARAM_ERROR<CR><LF>

Examples:

```
/* Example1: Set RX window 2 Data rate*/
#AT+RX2DR=5<CR>
<CR>
OK<CR> /* module returns the command error code */

/* Example2: Get RX window 2 Data rate */
# AT+RX2DR=?<CR>
5<CR>        /* module returns RX window 2 Data rate */
<CR>
OK<CR> /* module returns the command error code */
```

3.7.12 AT+RX2FQ - Frequency of the Rx window 2

Description	Sets/gets the Rx window 2 frequency.
Syntax	AT+RX2FQ=<freq><CR> AT+RX2FQ=?<CR>
Arguments	<freq>: value in Hz
Response	<freq><CR><LF>
Result code	<CR><LF>OK<CR><LF> <CR><LF>AT_PARAM_ERROR<CR><LF>

Examples:

```
/* Example1: Set RX window 2 Frequency */
#AT+RX2FQ=869535000<CR>
<CR>
OK<CR> /* module returns the command error code */

/* Example2: Get RX window 2 Frequency */
# AT+RX2FQ=?<CR>
869535000<CR>        /* module returns RX window 2 Frequency */
<CR>
OK<CR> /* module returns the command error code */
```

3.7.13 AT+TXP - Transmit power

Description	Sets/gets the transmit power.

Syntax	`AT+TXP=<TxPow><CR>`
	`AT+TXP=?<CR>`

Arguments	`<TxPow>`: must be in the range of the region activated in the range [0:15].
Response	`<TxPow><CR><LF>`

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`

Examples:

```
/* Example1: Set Transmit power */
#AT+TXP=3<CR>
<CR>
OK<CR> /* module returns the command error code */

/* Example2: Get Transmit power */
# AT+TXP=?<CR>
3<CR>             /* module returns Transmit power */
<CR>
OK<CR> /* module returns the command error code */
```

3.7.14 AT+PGSLOT - Ping slot

Description	Sets/gets the unicast ping slot periodicity.

Syntax	`AT+PGSLOT=<periodicity><CR>`
	`AT+PGSLOT=?<CR>`

Arguments	`<periodicity>`: periodicity to be transmitted, must be in the range [0:7] Ping slot periodicity is $2^{< periodicity >}$, in seconds.
Response	`<periodicity><CR><LF>`

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`

Example:

```
/* Example1: Set Ping Slot */
#AT+PGSLOT=4<CR>/* Set Ping Slot periodicity to 2^4= 16 seconds*/
<CR>
OK<CR> /* module returns the command error code */

/* Example2: Set Ping Slot */
#AT+PGSLOT=?<CR>
4<CR>
<CR>
OK<CR>/* module returns the command error code */
```

3.8 Radio test commands

3.8.1 AT+TTONE - RF tone test

Description	Starts a RF tone test.
Syntax	`AT+TTONE<CR>`
Arguments	None
Response	None

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_BUSY_ERROR<CR><LF>`

Example:

```
/* Example: starts a RF Tone test */
# AT+TTONE<CR>
[TimeDisplay]: Tx FSK Test<CR>
<CR>
OK<CR>
```

3.8.2 AT+TRSSI - RF RSSI tone test

Description	Starts a RF RSSI tone test.
Syntax	`AT+TRSSI<CR>`
Arguments	None
Response	`<rssi_lvl><CR><LF>`: value in dBm

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_BUSY_ERROR<CR><LF>`

Example:

```
/* Example: starts a RSSI tone test */
# AT+TRSSI<CR>
[TimeDisplay]: Rx FSK Test<CR>
[TimeDisplay]:>>> RSSI Value= -7 dBm<CR>
<CR>
OK<CR>
```

3.8.3 AT+TCONF - LoRa RF test configuration

Description	Sets/gets the LoRa RF test configuration.

Syntax	`AT+TCONF=<freq>:<pow>:<bw>:<sf>:<cr>:<lna>:<pa>:<mod>:<paylen>:<freq dev>:<lowdropt>:<BT><CR>`
	`AT+TCONF=?<CR>`

Arguments	• <freq>: frequency in Hz • <pow>: Tx power in range [-9:22] dBm • <bw>: – LoRa (in kHz) ◦ 0: 7.8125 ◦ 1: 15.625 ◦ 2: 31.25 ◦ 3: 62.5 ◦ 4: 125 ◦ 5: 250 ◦ 6: 500 – FSK: 4800 to 467000 Hz • <sf>: – LoRa: SF5 to SF12 bit/s – FSK: 600 to 300000 bit/s • <cr>: LoRa only – 1: 4/5 – 2: 4/6 – 3: 4/7 – 4: 4/8 • <lna>: low-noise amplifier – 0: Off – 1: On • <pa>: PA boost – 0: Off – 1: On • <mod>: modulation – [0: FSK – 1: LoRa – 2: BPSK(Tx) • <paylen>: payload length 1 to 256 • <freqdev>: FSK only 4800 to 467000 • <lowdropt>: low DR optimization, LoRa only – 0: Off – 1: On – 2: Auto (1 when SF11 or SF12, 0 otherwise) • <BT>: FSK only – 0: no Gaussian filter applied – 1: BT = 0,3 – 2: BT = 0,5 – 3: BT = 0,7 – 4: BT = 1

Response	• Freq= `<freq> Hz<CR>` • Power= `<pow> dBm<CR>` • Bandwidth= `<bw> (=125000 Hz)<CR>` • SF= `<sf><CR>` • CR= `<cr> (=4/5)<CR>` • LNA State= `<lna><CR>` • PA Boost State= `<pa><CR>` • Modulation `<mod><CR>` • Payload len= `<paylen> Bytes<CR>` • `<freqdev><CR>` • `LowDRopt[0 to 2]= <lowdropt><CR>` • `<BT><CR>`

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`

Note: <pa>, <freqdev>, <lowdropt> and <BT> arguments are required by the command syntax but are not used on the B-L072Z-LRWAN1 platform.

Examples:

```
/* Example1: Set LoRa RF test configuration */
#AT+TCONF=868000000:14:4:12:4/5:0:0:1:16:25000:2:3<CR>
<CR>
OK<CR> /* module returns the command error code */

/* Example2: Get LoRa RF test configuration */
# AT+TCONF=?<CR>
1: Freq= 868000000 Hz<CR>
2: Power= 14 dBm<CR>
3: Bandwidth= 4 (=125000 Hz)<CR>
4: SF= 12<CR>
5: CR= 1 (=4/5)<CR>
6: LNA State= 0<CR>
7: modulation LORA<CR>
8: Payload len= 16 Bytes<CR>
can be copy/paste in set cmd: AT+TCONF=868000000:14:4:12:4/5:0:0:1:16:25000:2:3<CR>
<CR>
OK<CR>
```

3.8.4 AT+TTX - Packets to be sent for PER RF TX test

Description	Starts a PER RF TX test with the number of packets to be sent.
Syntax	`AT+TTX=<nb_packets><CR>`
Arguments	`<nb_packets>`
Response	None

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`
	`<CR><LF>AT_BUSY_ERROR<CR><LF>`

Example:

```
/* Example: Starts a PER RF TX test with the number of packets to be sent. */
# AT+TTX=4<CR>
[TimeDisplay]:Tx Test<CR>
[TimeDisplay]:Tx Test: Packet 1 of 4<CR>
[TimeDisplay]:OnTxDone<CR>
[TimeDisplay]:Tx Test: Packet 2 of 4<CR>
[TimeDisplay]:OnTxDone<CR>
[TimeDisplay]:Tx Test: Packet 3 of 4<CR>
[TimeDisplay]:OnTxDone<CR>
[TimeDisplay]:Tx Test: Packet 4 of 4<CR>
[TimeDisplay]:OnTxDone<CR>
<CR>
OK<CR>
```

3.8.5 AT+TRX - Packets to be received for PER RF RX test

Description	Starts a PER RF RX test with the number of packets to be received.
Syntax	`AT+TRX=<nb_packets><CR>`
Arguments	`<nb_packets>`
Response	None

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`
	`<CR><LF>AT_BUSY_ERROR<CR><LF>`

Example:

```
/* Example: Starts a PER RF RX test with the number of packets to be received. */
# AT+TRLRA=4<CR>
[TimeDisplay]:PRE OK<CR>
[TimeDisplay]:HDR OK<CR>
[TimeDisplay]:OnRxDone<CR>
[TimeDisplay]:RssiValue=-7 dBm, SnrValue=7<CR>
[TimeDisplay]:Rx: 1 of 4 >>> PER= 0 %<CR> /* PER percentage is updated/displayed after each
reception*/
[TimeDisplay]:PRE OK<CR>
[TimeDisplay]:HDR OK<CR>
[TimeDisplay]:OnRxDone<CR>
[TimeDisplay]:RssiValue=-7 dBm, SnrValue=6<CR>
[TimeDisplay]:Rx: 2 of 4 >>> PER= 0 %<CR> /* PER percentage is updated/displayed after each
reception*/
[TimeDisplay]:PRE OK<CR>
[TimeDisplay]:HDR OK<CR>
[TimeDisplay]:OnRxDone<CR>
[TimeDisplay]:RssiValue=-7 dBm, SnrValue=5<CR>
[TimeDisplay]:Rx: 3 of 4 >>> PER= 0 %<CR> /* PER percentage is updated/displayed after each
reception*/
[TimeDisplay]:PRE OK<CR>
[TimeDisplay]:HDR OK<CR>
[TimeDisplay]:OnRxDone<CR>
[TimeDisplay]:RssiValue=-7 dBm, SnrValue=6<CR>
[TimeDisplay]:Rx: 4 of 4 >>> PER= 0 %<CR> /* PER percentage is updated/displayed after each
reception*/
<CR>
OK<CR>
```

3.8.6 AT+TTH - RF Tx hopping test

Description	Starts RF Tx hopping test from Fstart to Fstop, with Fdelta steps.
Syntax	`AT+TTH=<Fstart>,<Fstop>,<FDelta>,<nb_packets><CR>`
Arguments	• `<Fstart>`: frequency start (in Hz or MHz) • `<Fstop>`: frequency stop (in Hz or MHz) • `<FDelta>`: frequency bandwidth (in Hz) • `<nb_packets>`: number of packets to be sent
Response	None

Result code	`<CR><LF> OK <CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`
	`<CR><LF>AT_BUSY_ERROR<CR><LF>`

Example:

```
 /* Example: set TX hopping test  from 868 to 868,5 MHz  with 6 steps of 100 kHz */

# AT+TTH=868000000,868500000,100000,6<CR>
[TimeDisplay]: Tx Hop at 868000000Hz. 0 of 6<CR>
[TimeDisplay]:Tx LoRa Test<CR>
[TimeDisplay]:Tx 1 of 1<CR>
[TimeDisplay]:OnTxDone<CR>
[TimeDisplay]:Tx Hop at 868100000Hz. 1 of 6<CR>
[TimeDisplay]:Tx LoRa Test<CR>
[TimeDisplay]:Tx 1 of 1<CR>
[TimeDisplay]:OnTxDone<CR>
[TimeDisplay]:Tx Hop at 868200000Hz. 2 of 6<CR>
[TimeDisplay]:Tx LoRa Test<CR>
[TimeDisplay]:Tx 1 of 1<CR>
[TimeDisplay]:OnTxDone<CR>
[TimeDisplay]:Tx Hop at 868300000Hz. 3 of 6<CR>
[TimeDisplay]:Tx LoRa Test<CR>
[TimeDisplay]:Tx 1 of 1<CR>
[TimeDisplay]:OnTxDone<CR>
[TimeDisplay]:Tx Hop at 868400000Hz. 4 of 6<CR>
[TimeDisplay]:Tx LoRa Test<CR>
[TimeDisplay]:Tx 1 of 1<CR>
[TimeDisplay]:OnTxDone<CR>
[TimeDisplay]:Tx Hop at 868500000Hz. 5 of 6<CR>
[TimeDisplay]:Tx LoRa Test<CR>
[TimeDisplay]:Tx 1 of 1<CR>
[TimeDisplay]:OnTxDone<CR>
<CR>
OK<CR>
```

3.8.7 AT+CERTIF - Module in LoRaWAN certification with join mode

Description	Starts the module in LoRaWAN certification and with the choice of join mode.
Syntax	`AT+CERTIF=<mode><CR>`
Arguments	`<mode>` 0: join to a network by ABP 1: join to a network by OTAA
Response	`+EVT:JOINED` `+EVT:JOIN_FAILED`

Result code	`<CR><LF>OK<CR><LF>`
	`<CR><LF>AT_PARAM_ERROR<CR><LF>`

Examples:

```
/* Example1: Set the module in LoRaWAN certification and Join network by ABP */
#AT+CERTIF=0<CR>
+EVT:JOINED<CR> /* event: ABP configuration done. Ready to start Tx */
<CR>
OK<CR>

/* Example2: Set the module in LoRaWAN certification and Join network by OTAA */
#AT+CERTIF=1<CR>
<CR>
OK<CR>

+EVT:JOINED<CR> /* Event : OTAA join successful event */
```

3.8.8 AT+TOFF - RF test

Description	Stops the RF test.
Syntax	`AT+TOFF<CR>`
Arguments	None
Response	None
Result code	`<CR><LF>OK<CR><LF>`

Example:

```
/* Example: stops RF test */
# AT+TOFF<CR>
Test Stop<CR>
<CR>
OK<CR> /* module returns the command error code */
```

3.9 Information

3.9.1 AT+BAT - Battery level

Description	Gets the battery level (in mV).
Syntax	`AT+BAT=?<CR>`
Arguments	None
Response	`<level><CR><LF>`: value is in mV
Result code	`<CR><LF>OK<CR><LF>`

Example:

```
/* Example: Get the battery level in mV */
#AT+ BAT=?<CR>
3300<CR> /* battery level in mV */
<CR>
OK<CR> /* module returns the command error code */
```

4 Examples

Here are some basic examples that describe how to use the AT commands. In the following sections, commands provided by the host are preceded by `#`, and comments are embraced with `/* */`.

4.1 Join and send in unconfirmed mode

```
/* Check AT Link is OK */
#AT<CR>
<CR>
OK<CR>
/* Join in OTAA mode */
#AT+JOIN=1<CR>
 +EVT:JOINED<CR> /* Event: OTAA join successful event */
<CR>
OK<CR>
/* Network is joined, now data can be sent */
#AT+SEND=50:0:01234ABCD<CR>/* Send hexadecimal values in unconfirmed mode to port 50 */
<CR>
OK<CR>
```

4.2 Join and send in confirmed mode

```
/* Check AT Link is OK */
#AT<CR>
<CR>
OK<CR>
/* Join in OTAA mode */
#AT+JOIN=1<CR>
 +EVT:JOINED<CR> /* Event: OTAA join successful event */
<CR>
OK<CR>
/* Network is joined, now data can be sent */
#AT+SEND=50:1:01234ABCD<CR>/* Send hexadecimal values in confirmed mode to port 50 */
+EVT:SEND_CONFIRMED<CR>
<CR>
OK<CR>
```

4.3 Rx received data

It is possible to retrieve data sent from a specified port, when `+EVT:RX` is received.

```
/* Check AT Link is OK */
#AT<CR>
<CR>
OK<CR>
/* Join in OTAA mode */
#AT+JOIN=1<CR>
JOINED<CR> /* Event: OTAA join successful event */
<CR>
OK<CR>
/* Network is joined, now data can be sent */
#AT+SEND=50:0:01234ABCD<CR>/* Send hexadecimal values in unconfirmed mode to port 50 */
<CR>
OK<CR>
+EVT:50:4:ABCD<CR> /*Receive downlink frame */
+EVT:RX_1, DR 0, RSSI -49, SNR 5 <CR> /*Receive downlink parameters */
```

4.4 Class B enable request

The example below shows how to do a Class B request through an AT command sequence.

```
/* Join request in OTAA mode */
#AT+JOIN=1<CR>
<CR>
OK<CR>
/* wait for few seconds to wait for join to complete */
+EVT:JOINED<CR> /* end-device has joined the network */

/* now the network is joined, a request to enter into a Class B mode can be made */
#AT+CLASS=B<CR> /* Request to switch to Class B "enable" */
OK<CR>

/* A built-in MAC message is sent to the network to acquire the system time "Device Time
Req" */
#AT+SEND=50:0:0123<CR> /* Send data will allow piggybacking the MAC Device Time Req - could
be a dummy message */
OK<CR>

/* --> MAC Ping Device Time ANS is received by end-node in hidden way */
#AT+CLASS=?<CR>
B, S0<CR> /* Beacon Acquisition on-going */
OK<CR>
/* Loop on AT+CLASS=? until Beacon Acquisition on-going */

#AT+CLASS=?<CR>
B, S1<CR> /* Beacon Acquisition locked */
OK<CR>

#AT+PGSLOT=4<CR>/* Set Ping Slot periodicity to 2^4= 16 seconds and Send PingSlotInfoReq */
OK<CR>
/* --> MAC Ping Slot Info ANS is received by end-node in hidden way */

/* now the end-node is Class B "enable" */
#AT+CLASS=?<CR>
B<CR> /*Class B "enable"*/
OK<CR>

/* example: Local Time can be requested */
#AT+LTIME=?<CR>
LTIME:01h01m01s on 01/01/2021<CR>
<CR>
OK<CR>
```

5 Embedded software description

5.1 Firmware overview

This overview does not consider LoRa technology and implementation itself as it shares the implementation with the class A application. Readers interested by LoRa implementation details can refer to class A documentation.

The AT command processing can be found in the following source files:

- `lora_command.c`: contains all commands definition and handlers.
- `lora_at.c`: contains basic action to provide.

A command is processed whenever it ends with `<CR>` or `<LF>`.

5.2 LPUART

The AT-Slave module executes the two following task types:

- LoRa tasks: the AT-Slave module manages the received windows and sends data.
- the AT-Slave module receives commands from the master that schedules LoRa tasks and then sends back the requested value and the status of the command.

This means that the MCU does nothing most of the time, waiting for a command from the master or a LoRa task schedule.

So it is important to be in Stop mode in order to optimize low-level power of the MCU. As commands are received through the UART, the LPUART (low-power UART) is used, explaining why communication transfer rate is limited to 9600 bauds.

LPUART is initialized so that it is enabled in Stop mode, and wake-up from Stop mode is performed on Start bit detection. The LPUART handler `LPUART1_IRQHandler()` calls `HAL_UART_IRQHandler()` that, when RXNE flag is raised, triggers RxISR interrupt to transfer, via DMA, the input character that is stored in an internal circular buffer.

The buffer of read characters is then processed in the normal thread (not in the interrupt thread). A command is recognized when the new character received is `<CR>` or `<LF>`.

5.3 Compilation switches

The table below includes the main options for the application configuration.

Table 4. Main options for application configuration

Option type	Switch option	Definition	Location
LoRa band selection	REGION_EU868	Enables the EU high-band selection.	`LoRaWAN_AT_Slave\LoRaWAN\Target\lorawan_conf.h`
	REGION_EU433	Enables the EU low-band selection.	
	REGION_US915	Enables the US band selection.	
Debug	DEBUGGER_ENABLED	Enables the debugger and debug pins.	`LoRaWAN_AT_Slave\Core\Inc\sys_conf.h`
	APP_LOG_ENABLED	Enables trace mode.	
	VERBOSE_LEVEL	Trace level	
	PROBE_PINS_ENABLED	Enables four pins usable as probe signals by MW radio layer	
	LOW_POWER_DISABLE	Enables/disables the low-power mode: • 0: MCU enters Stop 2 mode[1]. • 1: MCU enters Sleep mode.	
Command	NO_HELP	Enables short help on AT commands when using `AT+<CMD>?`.	`LoRaWAN_AT_Slave\LoRaWAN\App\lora_command.c`

1. Stop 2 is a Stop mode with low-power regulator and VDD12I interruptible digital core domain supply OFF (less peripherals activated than in Stop 1 to reduce power consumption).

5.3.1 Debug switch

Debug and trace modes can be enabled by setting:

```
#define DEBUGGER_ENABLED  1
#define APP_LOG_ENABLED 1
#define PROBE_PINS_ENABLED 1
```

in the `LoRaWAN_AT_Slave\Core\Inc\sys_conf.h` file.

The debug mode `(DEBUGGER_ENABLED)` enables the SWD pins even when the MCU goes in low-power mode.

The probe pin mode (`PROBE_PINS_ENABLED`) enables `PROBE_GPIO_WRITE`, `PROBE_GPIO_SET_LINE`, and `PROBE_GPIO_RST_LINE` macros, as well as the debugger mode, even when the MCU goes in low-power mode.

The trace mode enables the `APP_LOG ()` macro that refers to the `UTIL_ADV_TRACE_COND_FSend()` function defined in `Utilities\trace\adv_trace\stm32_adv_trace.c`.

The trace level can be set with

```
#define VERBOSE_LEVEL VLEVEL_M
```

with four levels proposed:

- VLEVEL_OFF: traces disabled
- VLEVEL_L: functional traces enabled
- VLEVEL_M: debug traces enabled
- VLEVEL_H: all traces enabled

Note: To reach a true low power, `DEBUGGER_ENABLED` must be set to 0.

5.3.2 Footprint

Values given in the below table, have been measured for the following configuration:

IAR Compiler: IAR Embedded Workbench® 8.32.4

- Optimization: level 3 for size
- Debug option: off
- Trace option: VLEVEL_M: debug traces enabled
- Target: B-L072Z-LRWAN1
- LoRaMAC Class A
- LoRaMAC region EU868 and US915

Table 5. **Memory footprint detail**

Label	RO data (FLASH, bytes)	RW data (RAM, bytes)	Description
Application	17119	1775	User application including `lora_at.c`, `lora_command.c`, and `test_rf.c`
LoRaWAN stack	27863	2924	Middleware LmHandler interface, MAC and Region
SubGHz_Phy	5496	612	Middleware radio interface
Utilities	4158	1732	All STM32 services (sequencer, time server, low-power mgr, trace, and mem)
HAL	11820	0	STM32L0 HAL and LL drivers
IAR startup	530	1537	int vector, CSTACK, HEAP and init table
IAR library	988	0	IAR proprietary libraries
Total memory	**67974**	**8580**	**-**

Revision history

Table 6. **Document revision history**

Date	Version	Changes
10-Jan-2017	1	Initial release.
25-Aug-2017	2	Updated document title and Section 3: Overview. Added Section 4.8: RF tests and its subsections. Updated Figure 1: Terminal emulation mode. Updated Table 1: List of acronyms. Minor text edits across the whole document.
14-Dec-2017	3	Updated Section 2: Reference documents and Section 3: Overview.
11-Jul-2018	4	Updated Section 2: Reference documents, Section 6.3: Compilation switches, Section 6.3.1: Debug switches and Section 6.4: Footprint. Minor text edits across the whole document. Updated Table 13: LoRa® class command and its footnote 1, Table 37: Compilation switch options and Table 38: AT_Slave footprint. Removed former Section 4.4.5: AT+FCD: frame counter downlink, former Section 4.4.6: AT+FCU: frame counter uplink, former Section 6.2: Low layer driver and former Note in Section 4.3.4: AT+NJM: LoRa® network join mode.
17-Dec-2018	5	Updated Section 4: AT commands, Section 4.2.3: AT+APPSKEY: Application session key and Section 4.2.7: AT+NWKSKEY: Network session key. Added Section 4.5: Class B mode, Section 4.6: Asynchronous events with their subsections and tables, and Section 5.4: Class B enable request. Minor text edits across the whole document. Updated Figure 1: Terminal emulation mode. Updated Table 8: Application session key command, Table 12: Network session key command, Table 13: LoRa® class command and its footnotes.
08-Jul-2019	6	Updated Section 4.8.3: AT+TTX: Start RF Tx test and Section 4.8.3: AT+TTX: Start RF Tx test. Updated Table 31: Start RF Tx test command and Table 32: Start RF Rx test command.
15-Feb-2021	7	Updated Section 4: AT commands, Section 4.3.2: AT+SEND: Send binary data, Section 4.8.3: AT+TTX: Start RF Tx test, Section 4.8.4: AT+TRX: Start RF Rx test, Section 4.8.5: AT+TCONF: Configure LoRa RF test, Section 4.8.7: AT+CERTIF: Set the module in LoRaWAN certification mode, Section 5.1: Join and send, Section 5.2: Confirmation, Section 5.3: Receiving data, Section 5.4: Class B enable request, Section 6.3.1: Debug switches and Section 6.4: Footprint. Added Section 4.1.4: AT+VL: Verbose level, Section 4.4.5: AT+BAND: Active region and Section 4.8.8: AT+TTH: RF Tx hopping test. Removed former Section 4.2.2: AT+APPKEY: Application key, Section 4.2.3: AT+APPSKEY: Application session key, Section 4.2.7: AT+NWKSKEY: Network session key, Section 4.3.1: AT+CFM: confirm mode, Section 4.3.2: AT+CFS: confirm status, Section 4.3.4: AT+NJM: LoRa® network join mode, Section 4.3.5: AT+NJS: LoRa® network join status, Section 4.3.6: AT+RECV: last received text data, Section 4.3.7: AT+RECVB: last received binary data, Section 4.3.8: AT+SEND: send text data, Section 4.4.7: AT+PNM: Public network mode, Section 4.5.2: AT+BFREQ, Section 4.5.3: AT+BTIME, Section 4.5.4: AT+BGW, Section 4.7.2: AT+RSSI: RSSI on reception and Section 4.7.3: AT+SNR: Signal to noise ratio. Updated Table 6: Application identifier command, Table 7: Device address command, Table 9: Network ID command, Table 10: Join LoRa network command, Table 11: Send data command and its footnotes, Table 13: LoRa® class command, Table 14: Duty cycle settings command, Table 26: Asynchronous events, Table 28: Version of the firmware command, Table 31: Start RF Tx test command, Table 32: Start RF Rx test command, Table 33: Configure LoRa RF test command,Table 35: Set the module in LoRaWAN® Certification mode command, Table 37: Compilation switch options and Table 38: AT_Slave footprint. Minor text edits across the whole document.

Date	Version	Changes
5-Oct-2021	8	Updated: • Section 3 AT commands • Section 4 Examples • Section 5 Embedded software description

Contents

List of tables

List of figures