AN5481

Application note

LoRaWAN® AT commands for STM32CubeWL

Introduction

This application note explains how to interface with the LoRaWAN® to manage LoRa® wireless link by the way of AT commands .

This document lists the set of AT commands on the NUCLEO_WL55JC STM32WL Nucleo-73 boards (order codes NUCLEO-WL55JC1 for high-frequency band and NUCLEO-WL55JC2 for low-frequency band).

The firmware of the STM32CubeWL MCU Package is based on the STM32Cube HAL drivers.

1 General information

The STM32CubeWL runs on STM32WL Series microcontrollers based on the Arm®Cortex®-M processor.

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

arm

Table 1. Acronyms

Acronym	Definition
ABP	Activation by personalization
LoRa	Long range radio technology
LoRaWAN	LoRa wide-area network
OTAA	Over-the-air activation
RSSI	Received signal strength indicator
SNR	Signal/noise ratio

Reference documents

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

[2] Application note *How to build a LoRa application with STM32CubeWL* (AN5406)

[3] Application note *Migrating from STM32L0, STM32L1, STM32L4 Series to STM32WLEx microcontrollers* (AN5408)

2 Overview

The NUCLEO-WL55JC, STM32WL Nucleo-73 boards embed a set of AT commands for the LoRa RF test and LoRaWAN communications.

This application note details the interface, AT commands definition, some use cases and the embedded software description. For complete description of a LoRa application built with STM32CubeWL, refer to document [2].

3 AT commands

The AT commands are used to drive the LoRa module and to send data (see document [1] for more details). AT commands are sent through the UART peripheral.

In the demonstration below, a host (typically a Windows® host) can be connected to the module using ST-LINK. The UART over ST-LINK can then be used, with standard Windows software such as TeraTerm or PuTTY) with the following parameters:

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

Here is the typical configuration of Tera Term:

Figure 1. Tera Term serial port set up

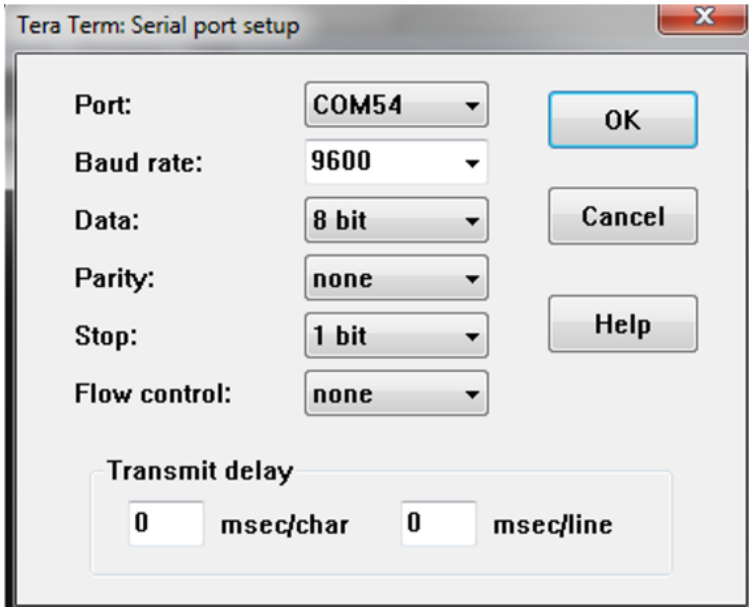
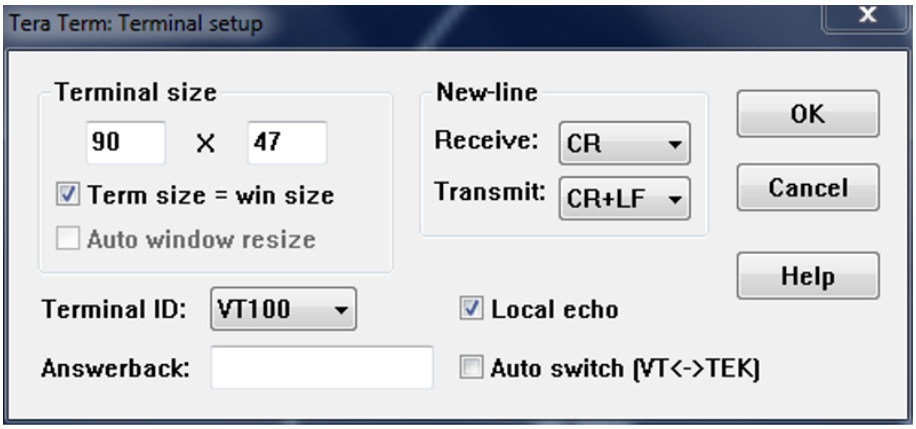


Figure 2. Tera Term terminal setup



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

Next sections describe each command, including some examples. Each command preceded by # is the one provided by the host to the module. Then the return of the module is printed.

AT_ERROR is returned when a command is not recognized.

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 commands 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+APPKEY	[=01:02:0A:FB:A1:CD:4D:20:01:02:30:40:5A:6B:7F:88]	Sets/gets the application key.
AT+APPSKEY	[=02:03:0A:FB:A1:CD:4D:20:01:02:30:40:5A:6B:7F:88]	Sets the application key.
AT+NWKSKEY	[=02:03:0A:FB:A1:CD:4D:20:01:02:30:40:5A:6B:7F:88]	Sets 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	[=01:02:03:04]	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+SEND	[=port_nb:confirmedmode:data] where confirmedmode = 0 or 1.	Sends packets to the network.
AT+PGSLOT	[=periodicity]	Sets/gets the ping slot.
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.

Command	Parameters	Description
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.
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] [=868000000:14:125:12:4/5:0:0] for example	Sets/gets the config LoRa RF test.
AT+TTX	[=nb_packets_sent]	Sets the number of packets to be sent for PER Tx LoRa test.
AT+TRX	[=nb_packets_received]	Sets the number of packets to be received for PER RF RX test.
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 Rx frame has been received on a RX window with downlink frame.
+ EVT:	RX_<slot>:<DR>:<RSSI>:<SNR>	Notifies the host module that an asynchronous Rx frame has been received on a RX window with downlink parameters.
+ 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>
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>
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	<CR><LF> OK <CR><LF>

Example:

```
/* Example: set NVIC system reset */
# ATZ<CR>
ATtention command interface
0s000:MAC_VERSION= VX.Y.Z_rcA
###### DevEui:    AA-AA-AA-AA-AA-AA-AA-AA
###### AppEui:    01-02-03-04-05-06-07-08
###### AppKey:    CA FE CA FE CA FE CA FE CA FE CA FE CA FE CA FE
###### GenAppKey: BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
```

Note: The displayed keys by command above after ###### (DevEUI, AppEui, AppKey and GenAppKey) 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=<nb><CR> or 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>

Examples:

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

/* Example2: get verbose level */
# AT+VL =?<CR>
3
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
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> or AT+APPEUI=? <CR>
Arguments	<id>, 8-byte value separated by ":" (hexadecimal format string)
Response	<id><CR><LF>
Result code	<CR><LF> OK <CR><LF>

Examples:

```
/* Example1: set APP EUI */
# AT+APPEUI=01:02:03:04:05:06:07:08 <CR>
###### DevEui:    AA-AA-AA-AA-AA-AA-AA-AA
###### AppEui:    01-02-03-04-05-06-07-08
###### AppKey:    CA FE CA FE CA FE CA FE CA FE CA FE CA FE CA FE
###### GenAppKey: BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
OK<CR>

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

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

3.5.2 AT+APPKEY - Application key

Description	Sets/gets the application key.
Syntax	AT+APPKEY=<id><CR> or AT+APPKEY=? <CR>
Arguments	<key>, 16-byte value separated by ":" (hexadecimal format string)
Response	<key><CR><LF>
Result code	<CR><LF> OK <CR><LF>

Examples:

```
/* Example1: set APP Key */
# AT+APPKEY=01:02:0A:FB:A1:CD:4D:20:01:02:30:40:5A:6B:7F:88 <CR>
###### DevEui:    AA-AA-AA-AA-AA-AA-AA-AA
###### AppEui:    BB-BB-BB-BB-BB-BB-BB-BB
###### AppKey:    CA FE CA FE CA FE CA FE CA FE CA FE CA FE CA FE
###### GenAppKey: BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
OK <CR>

/* Example2: get APP Key */
# AT+APPKEY=? <CR>
01:02:0A:FB:A1:CD:4D:20:01:02:30:40:5A:6B:7F:88
OK <CR>
```

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

3.5.3 AT+APPSKEY - Application session key

Description	Sets/gets the application session key.
Syntax	AT+APPSKEY=<key><CR>
Arguments	<key>, 16-byte value separated by ":" (hexadecimal format string)
Response	<key><CR><LF>
Result code	<CR><LF> OK <CR><LF>

Example:

```
/* Example: set APP Session Key */
# AT+APPSKEY=02:03:0A:FB:A1:CD:4D:20:01:02:30:40:5A:6B:7F:88 <CR>
###### DevEui:    AA-AA-AA-AA-AA-AA-AA-AA
###### AppEui:    BB-BB-BB-BB-BB-BB-BB-BB
###### AppKey:    CA FE CA FE CA FE CA FE CA FE CA FE CA FE CA FE
###### GenAppKey: BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
OK<CR>
/* APP Session Key is not displayed, because confidential */
/* Not possible to get APP Session Key neither */
```

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

3.5.4 AT+NWKSKEY - Network session key

Description	Sets the network session key.
Syntax	AT+NWKSKEY=<key> <CR>
Arguments	<key>, 16-byte value separated by ":" (hexadecimal format string)
Response	None
Result code	<CR><LF> OK <CR><LF>

Example:

```
/* Example: set NWK Session Key */
# AT+NWKSKEY=02:03:0A:FB:A1:CD:4D:20:01:02:30:40:5A:6B:7F:88 <CR>
###### DevEui:    AA-AA-AA-AA-AA-AA-AA-AA
###### AppEui:    BB-BB-BB-BB-BB-BB-BB-BB
###### AppKey:    CA FE CA FE CA FE CA FE CA FE CA FE CA FE CA FE
###### GenAppKey: BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
OK <CR>
/* NWK Session Key is not displayed, because confidential */
/* Not possible to get NWK Session Key neither */
```

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

3.5.5 AT+DADDR - Device address

Description	Sets/gets the device address.
Syntax	AT+DADDR=<address><CR> or AT+DADDR=? <CR>
Arguments	<address>, 4-byte value separated by ":" (hexadecimal format string)
Response	< address ><CR><LF>
Result code	<CR><LF> OK <CR><LF>

Examples:

```
/* Example1: set device address*/
# AT+DADDR=01:02:0A:0B <CR>
###### DevEui:    AA-AA-AA-AA-AA-AA-AA-AA
###### AppEui:    BB-BB-BB-BB-BB-BB-BB-BB
###### AppKey:    CA FE CA FE CA FE CA FE CA FE CA FE CA FE CA FE
###### GenAppKey: BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
OK <CR>

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

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

3.5.6 AT+DEUI - Device EUI

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

Examples:

```
/* Example1: set device EUI*/
# AT+DEUI=01:02:03:04:05:06:07:08 <CR>
###### DevEui:    01-02-03-04-05-06-07-08
###### AppEui:    BB-BB-BB-BB-BB-BB-BB-BB
###### AppKey:    CA FE CA FE CA FE CA FE CA FE CA FE CA FE CA FE
###### GenAppKey: BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
OK <CR>

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

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

3.5.7 AT+NWKID - Network ID

Description	Sets/gets the network ID.
Syntax	AT+NWKID=<id> <CR> or AT+NWKID=? <CR>
Arguments	<id>, 4-byte value separated by ":" (hexadecimal format string)
Response	<id><CR><LF>
Result code	<CR><LF> OK <CR><LF>

Examples:

```
/* Example1: set device EUI*/
# AT+NWKID=01:02:03:04 <CR>
###### DevEui:    01-02-03-04-05-06-07-08
###### AppEui:    BB-BB-BB-BB-BB-BB-BB-BB
###### AppKey:    CA FE CA FE CA FE CA FE CA FE CA FE CA FE CA FE
###### GenAppKey: BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
OK <CR>

/* Example2: get device EUI */
# AT+NWKID=? <CR>
01:02:03:04
OK <CR>
```

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

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: indicates the join to a network by ABP. 1: indicates the join to a network by OTAA.
Response	+EVT:JOINED
Result code	<CR><LF> OK <CR><LF>

Examples:

```
/* Example1: Join a network by ABP */
#AT+JOIN=0  <CR>
OK <CR>  /* module returns the command error code */

/* Example2: Join a network by OTAA */
#AT+JOIN=1 <CR>
+EVT:JOINED   /* Event : OTAA join successful event */
OK   <CR>
```

3.6.2 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: indicates this is an unconfirmed message. – 1: indicates this is a confirmed message. • `<payload>`: payload in hexadecimal format strings (maximum length is 242 bytes)
Response	`+EVT:SEND_CONFIRMED`

Result code	`<CR><LF>AT_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 : “HELLO”, with APP port is 2, unconfirmed message */
OK <CR> /* module returns the command error code */

/* 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 */
OK <CR>/* module returns the command error code */
```

3.6.3 AT+PGSLOT - Ping slot

Description	Sets/gets the unicast ping slot periodicity.
Syntax	AT+PGSLOT=<periodicity> <CR> or AT+PGSLOT=? <CR>
Arguments	< periodicity >: periodicity to be transmitted, must be in the range [0:7] Ping slot periodicity is $2^{<\text{periodicity}>}$, in seconds.
Response	< periodicity > <CR><LF>
Result code	<CR><LF> OK <CR><LF>

Example:

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

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

3.7 LoRa network management

3.7.1 AT+VER - Firmware version

Description	Gets the version of the AT_Slave firmware.
Syntax	AT+VER=? <CR>
Arguments	None
Response	< version > <CR><LF>
Result code	<CR><LF> OK <CR><LF>

Example:

```
/* Example1: Get version of the AT Slave Firmware */
#AT+VER=? <CR>/* Disable ADR*/
APP_VERSION= VX.Y.Z           /* Application version */
MAC_VERSION= VA.B.C_rcD /*  MAC version */
OK <CR> /* module returns the command error code */
```

3.7.2 AT+ADR - Adaptive data rate functionality

Description	Sets/gets the adaptive data rate functionality.
Syntax	AT+ADR=<enabled> <CR> or AT+ADR=? <CR>
Arguments	< enabled >, with default 1 (enabled). • 0: indicates the ADR is disabled. • 1: indicates the ADR is enabled.
Response	< enabled ><CR><LF>
Result code	<CR><LF> OK <CR><LF>

Examples:

```
/* Example1: Disable ADR */
#AT+ADR=0 <CR>/* Disable ADR*/
###### DevEui:    AA-AA-AA-AA-AA-AA-AA-AA
###### AppEui:    BB-BB-BB-BB-BB-BB-BB-BB
###### AppKey:    CA FE CA FE CA FE CA FE CA FE CA FE CA FE CA FE
###### GenAppKey: BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
OK <CR> /* module returns the command error code */

/* Example2: Check ADR status */
# AT+ADR=? <CR>
0             /* module returns ADR status */
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=<range><CR> or 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>

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 */
OK <CR> /* module returns the command error code */

/* Example2: Get Data rate */
# AT+DR=? <CR>
2             /* module returns TX data rate */
OK <CR> /* module returns the command error code */
```

3.7.4 AT+BAND - Active region

Description	Sets/gets the active region.
Syntax	AT+BAND=<band> <CR> or 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>

Examples:

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

/* Example2: Get Active region */
# AT+BAND=? <CR>
5:EU868             /* module returns Active region */
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> or AT+CLASS=? <CR>
Arguments	< class >: must be A, B or C.
Response	< class > <CR><LF>
Result code	<CR><LF> OK <CR><LF>

Examples:

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

/* Example2: Get Active region */
# AT+CLASS=? <CR>
C             /* module returns Active Class */
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> or AT+DCS=? <CR>
Arguments	< dutyCycleEnable >: must be 0 or 1.
Response	< dutyCycleEnable ><CR><LF>
Result code	<CR><LF> OK <CR><LF>

Examples:

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

/* Example2: Get Duty cycle */
# AT+ DCS =? <CR>
1             /* module returns Duty cycle */
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> or AT+JN1DL=? <CR>
Arguments	< delay >: value in ms
Response	< delay > <CR><LF>
Result code	<CR><LF> OK <CR><LF>

Examples:

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

/* Example2: Get Join Delay on RX window 1*/
# AT+JN1DL=? <CR>
5000        /* module returns Join Delay on RX window 1 in ms*/
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> or AT+JN2DL=? <CR>
Arguments	< delay >: value in ms
Response	< delay > <CR><LF>
Result code	<CR><LF> OK <CR><LF>

Examples:

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

/* Example2: Get Join Delay on RX window 2*/
# AT+JN2DL=? <CR>
8000        /* module returns Join Delay on RX window 2 in ms*/
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> or AT+RX1DL=? <CR>
Arguments	< delay >: value in ms
Response	< delay > <CR><LF>
Result code	<CR><LF> OK <CR><LF>

Examples:

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

/* Example2: Get Delay on RX window 1*/
# AT+RX1DL=? <CR>
1500        /* module returns Delay on RX window 1 in ms*/
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> or AT+RX2DL=? <CR>
Arguments	< delay >: value in ms
Response	< delay > <CR><LF>
Result code	<CR><LF> OK <CR><LF>

Examples:

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

/* Example2: Get delay on RX window 2*/
# AT+RX2DL=? <CR>
2500        /* module returns Delay on RX window 2 in ms*/
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> or AT+RX2DR=? <CR>
Arguments	< datarate >: value in range [0:15]
Response	< datarate > <CR><LF>
Result code	<CR><LF> OK <CR><LF>

Examples:

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

/* Example2: Get RX window 2 Data rate */
# AT+RX2DR=? <CR>
5        /* module returns RX window 2 Data rate */
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> or AT+RX2FQ=? <CR>
Arguments	< freq >: value in Hz
Response	< freq > <CR><LF>
Result code	<CR><LF> OK <CR><LF>

Examples:

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

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

3.7.13 AT+TXP - Transmit power

Description	Sets/gets the transmit power.
Syntax	AT+TXP=< TxPw > <CR> or AT+TXP=? <CR>
Arguments	< TxPw >: must be in the range of the region activated in the range [0:15].
Response	< TxPw > <CR><LF>
Result code	<CR><LF> OK <CR><LF>

Examples:

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

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

3.8 Radio test commands

3.8.1 AT+TTONE - RF tone test

Description	Sets a RF tone test.
Syntax	AT+TTONE<CR>
Arguments	None
Response	None
Result code	<CR><LF> OK <CR><LF>

Example:

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

3.8.2 AT+TRSSI - RF RSSI tone test

Description	Sets 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>

Example:

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

3.8.3 AT+TCONF - LoRa RF test configuration

Description	Sets/gets the LoRa RF test configuration.
Syntax	AT+TCONF =< config > <CR> or AT+TCONF =? <CR>
Arguments	< config >
Response	< config > <CR><LF>
Result code	<CR><LF> OK <CR><LF>

Examples:

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

/* Example2: Get LoRa RF test configuration */
# AT+TCONF=? <CR>
freq:  in Hz
power  :[-9 :22]dBm
bandwidth : Lora [0:7.8125, 1: 15.625, 2: 31.25, 3: 62.5, 4: 125, 5: 250, 6: 500]kHz
FSK : [4800Hz :467000 Hz]
loraSf_datarate : Lora[SF5..SF12] FSK [600..300000 bits/s]
codingRate: Lora Only [1: 4/5, 2: 4/6, 3: 4/7, 4: 4/8]
lna: 0:off 1:On
paBoost_state: 0:off 1:On
modulation: 0: FSK, 1: Lora, 2:BPSK(Tx)
payloadLen: [1:256]
fskdDeviation: FSK only [4800:467000]
Note: no check applied wrt bandwidth. Common practice is to have bandwidth>1,5*fskDev
lowDrOpt: Lora Only 0: off, 1:On, 2: Auto (1 when SF11 or SF12, 0 otherwise
BTproduct: FSK only [0 no Gaussian Filter Applied, 1:  BT=0,3, 2: BT=0,5, 3: BT=0,7, 4: BT=1]
can be copy/paste in set cmd: 868000000:14:4:12:4/5:0:0:1:16:25000:2:3
/* config sum up  for set command*/
OK <CR> /* module returns the command error code */
```

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

Description	Sets the number of packets to be sent for a PER RF TX test.
Syntax	AT+ TTX =<nb_packets> <CR>
Arguments	<nb_packets>
Response	None
Result code	<CR><LF> OK <CR><LF>

Example:

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

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

Description	Sets the number of packets to be received for a PER RF RX test.
Syntax	AT+ TRX =<nb_packets> <CR>
Arguments	<nb_packets>
Response	None
Result code	<CR><LF> OK <CR><LF>

Example:

```
/* Example: set number of packets to be received for PER RF RX test */
# AT+TRLRA=4 <CR>
[TimeDisplay]:PRE OK
[TimeDisplay]:HDR OK
[TimeDisplay]:OnRxDone
[TimeDisplay]:RssiValue=-7 dBm, SnrValue=7
[TimeDisplay]:Rx: 1 of 4  >>> PER= 0 %  /* PER percentage is updated/displayed after each
reception*/
[TimeDisplay]:PRE OK
[TimeDisplay]:HDR OK
[TimeDisplay]:OnRxDone
[TimeDisplay]:RssiValue=-7 dBm, SnrValue=6
[TimeDisplay]:Rx: 2 of 4  >>> PER= 0 % /* PER percentage is updated/displayed after each
reception*/
[TimeDisplay]:PRE OK
[TimeDisplay]:HDR OK
[TimeDisplay]:OnRxDone
[TimeDisplay]:RssiValue=-7 dBm, SnrValue=5
[TimeDisplay]:Rx: 3 of 4  >>> PER= 0 % /* PER percentage is updated/displayed after each
reception*/
[TimeDisplay]:PRE OK
[TimeDisplay]:HDR OK
[TimeDisplay]:OnRxDone
[TimeDisplay]:RssiValue=-7 dBm, SnrValue=6
[TimeDisplay]:Rx: 4 of 4  >>> PER= 0 % /* PER percentage is updated/displayed after each
reception*/
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>,<PacketNb><CR>
Arguments	<Fstart>, <Fstop>, <FDelta>,<PacketNb>
Response	None
Result code	<CR><LF> OK <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
[TimeDisplay]: Tx Hop at 868000000Hz. 0 of 6
[TimeDisplay]:Tx LoRa Test
[TimeDisplay]:Tx 1 of 1
[TimeDisplay]:OnTxDone
[TimeDisplay]:Tx Hop at 868100000Hz. 1 of 6
[TimeDisplay]:Tx LoRa Test
[TimeDisplay]:Tx 1 of 1
[TimeDisplay]:OnTxDone
[TimeDisplay]:Tx Hop at 868200000Hz. 2 of 6
[TimeDisplay]:Tx LoRa Test
[TimeDisplay]:Tx 1 of 1
[TimeDisplay]:OnTxDone
[TimeDisplay]:Tx Hop at 868300000Hz. 3 of 6
[TimeDisplay]:Tx LoRa Test
[TimeDisplay]:Tx 1 of 1
[TimeDisplay]:OnTxDone
[TimeDisplay]:Tx Hop at 868400000Hz. 4 of 6
[TimeDisplay]:Tx LoRa Test
[TimeDisplay]:Tx 1 of 1
[TimeDisplay]:OnTxDone
[TimeDisplay]:Tx Hop at 868500000Hz. 5 of 6
[TimeDisplay]:Tx LoRa Test
[TimeDisplay]:Tx 1 of 1
[TimeDisplay]:OnTxDone

OK
```

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

Description	Sets the module in LoRaWAN certification and with the choice of join mode.
Syntax	AT+CERTIF =<mode > <CR>
Arguments	< mode > 0: indicates the join to a network by ABP. 1: indicates the join to a network by OTAA.
Response	+EVT:JOINED
Result code	<CR><LF> OK <CR><LF>

Examples:

```
/* Example1: Set the module in LoRaWAN certification and Join network by ABP */
#AT+CERTIF=0 <CR>
###### DevAddr:   01020304
###### NwkSKey:   CA FE CA FE CA FE CA FE CA FE CA FE CA FE CA FE
###### AppSKey:   CA FE CA FE CA FE CA FE CA FE CA FE CA FE CA FE
+EVT:JOINED
[TimeDisplay]: TX on freq 868100000 Hz at DR 0
OK <CR> /* module returns the command error code */

/* Example2: Set the module in LoRaWAN certification and Join network by OTAA */
#AT+CERTIF=1 <CR>
[TimeDisplay]: TX on freq 868100000 Hz at DR 0
 +EVT:JOINED     /* Event: OTAA Join successful event*/
OK   <CR> /* module returns the command error code */
```

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
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 /* battery level in mV */
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>
OK<CR>
/* Join in OTAA mode */
# AT+JOIN = 1 <CR>
 +EVT:JOINED <CR> /* Event: OTAA join successful event */
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 */
OK<CR>
```

4.2 Join and send in confirmed mode

```
/* Check AT Link is OK */
# AT <CR>
OK<CR>
/* Join in OTAA mode */
# AT+JOIN = 1 <CR>
 +EVT:JOINED <CR> /* Event: OTAA join successful event */
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>
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>
OK<CR>
/* Join in OTAA mode */
# AT+JOIN = 1 <CR>
JOINED <CR> /* Event: OTAA join successful event */
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 */
OK<CR>
+EVT:50:4:ABCD <CR> /*Receive downlink frame */
+EVT:RX_1, DR 0, RSSI -49, SNR 5 <CR> /*Receive downlink parameters */
```

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 [2].

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.	`AT_Slave\LoRaWAN\Target\lorawan_conf.h`
	REGION_EU433	Enables the EU low-band selection.	
	REGION_US915	Enables the US band selection.	
Debug	DEBUGGER_ON	Enables the debugger and debug pins.	`AT_Slave \LoRaWAN\Core\Inc\sys_conf.h`
	APP_LOG_ENABLED	Enables trace mode.	
	VERBOSE_LEVEL	Trace level	
Command	NO_HELP	Enables short help on AT commands when using `AT+<CMD>?`.	`AT_Slave \LoRaWAN\App\lora_command.c`

5.3.1 Debug switch

Debug and trace modes can be enables by setting:

```
#define DEBUGGER_ON  1
#define APP_LOG_ENABLED 1
```

in the `AT_Slave \LoRaWAN\Core\Inc\sys_conf.h` file.

The debug mode enables the `DBG_GPIO_SET` and `DBG_GPIO_RST` macros. The debug mode also enables the debugger mode even when MCU goes in low power.

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

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_ON` must be set to 0.

5.3.2 Footprint

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

IAR Compiler: EWARM 8.30.1

- Optimization: level 3 for size
- Debug option: off
- Trace option: VLEVEL_L (minimal traces)
- Target: NUCLEO-WL55JC1
- LoRaMAC Class A
- LoRaMAC region EU868 only

Table 5. Memory footprint detail

Memory location	FLASH (bytes)	RAM (bytes)	Description
Application layer	9016	1053	lora_app.c, lora_at.c, lora_command.c, test_rf.c
LoRa stack	32869	3405	Includes MAC + RF driver.
HAL	12942	84	-
Utilities	3848	972	Includes services like sequencer, timer server, low-power manager.
Others	2472	437	Includes other peripherals such as USART, DMA or ADC.
Total memory	**61147**	**5939**	**-**

Revision history

Table 6. **Document revision history**

Date	Version	Changes
3-Jun-2020	1	Initial release.
20-Nov-2020	2	Updated: • Table 2. AT commands • Section 3.4.1 AT • Note added in Section 3.4.3 and in all sub-sections of Section 3.5 • Section 3.6.1 AT+JOIN - Join LoRa network • Section 3.6.2 AT+SEND - Send data to LoRa network • Section 3.8.3 AT+TCONF - LoRa RF test configuration • Section 3.8.4 AT+TTX - Packets to be sent for PER RF TX test • Section 3.8.5 AT+TRX - Packets to be received for PER RF RX test • Section 3.8.7 AT+CERTIF - Module in LoRaWAN certification with join mode • Section 4 Examples • Table 4. Main options for application configuration • Table 5. Memory footprint detail Added: • Section 3.3 Event table • Section 3.4.2 AT? • Section 3.8.6 AT+TTH - RF Tx hopping test • Section 4.3 Rx received data

Contents

List of tables

List of figures