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 means of AT commands .

This document lists the set of AT commands on the NUCLEO_WL55JC STM32WL Nucleo 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 and terms

Acronym	Definition
ABP	Activation by personalization
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] Application note *How to build a LoRa® application with STM32CubeWL* (AN5406)

[3] User manual *Description of STM32WL HAL and low-layer drivers* (UM2642)

2 Overview

The NUCLEO-WL55JC, STM32WL Nucleo 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 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 send data. 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

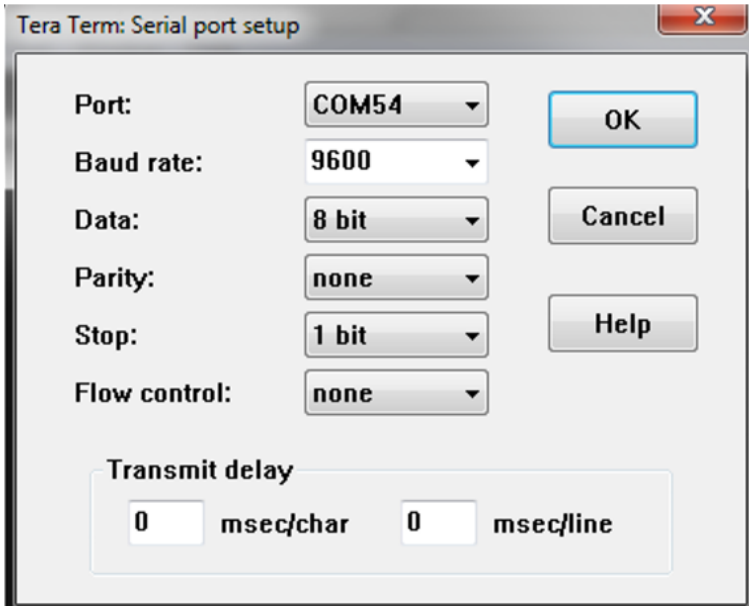
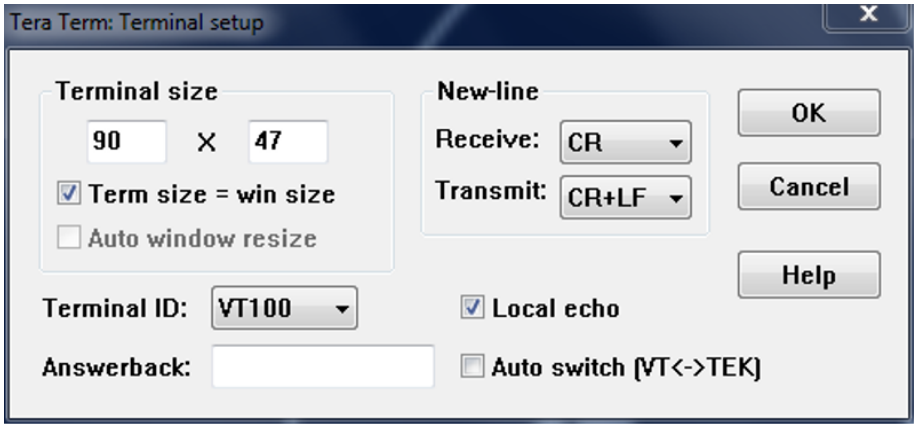


Figure 2. Tera Term terminal setup



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 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+RFS	None	Erase LoRaWAN context in flash memory and restore factory settings after a reset.
AT+CS	None	Store current LoRaWAN context to flash memory sector.
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.
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.

Command	Parameters	Description
AT+JN2DL	[=delay] where delay in ms	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 unicast ping slot periodicity. If the end device is joined, send a new PingSlotInfoReq.
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.
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	Description
+EVT:JOINED	Notifies the host module has been join on the gateway by OTAA.
+EVT:JOIN FAILED	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. Port: Rx port in the range of 1 to 199, used by the application server Size: number of bytes included in <payload>. in the range of 1 to LORAWAN_APP_DATA_BUFFER_MAX_SIZE Payload: received data as hexadecimal bytes

Event	Description
+EVT:RX_<slot>:<DR>:<RSSI>:<SNR>	Notifies the host module that an asynchronous frame has been received on a RX window with downlink parameters. RX_<slot> can be: • RX_1: received on first ClassA window • RX_2: received on second ClassA window • RX_C: received on continuous unicast ClassC window • RX_C_MC: received on continuous multicast ClassC window • RX_B: received on ping unicast ClassB window • RX_B_MC: received on ping multicast ClassB window DR: Rx datarate in the range of 0 to 15 (dependant of region definition) RSSI: Evaluated power signal strength (negative values possible) SNR: signal-to-noise ratio
+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. DMODM: Demodulation margin in the range of 0 to 254 indicating the link margin in dB of the last successfully received LinkCheckReq command. GWN: Number of gateways that successfully received the last LinkCheckReq command.
+EVT:SEND_CONFIRMED	Notifies the host module that a Tx frame has been acknowledge by the gateway.
+EVT:SWITCH_TO_CLASS_<Class>	Notifies the host module that the device has changed of class in the range of (A, B, C).
+EVT:RX_BC,<DR>,<RSSI>,<SNR>,<FQ>,<TIME>,<DESC>,<INFO>	Notifies the host that a new beacon (ClassB) has been received on the RX_BC window. DR: Rx datarate in the range of 0 to 15 (dependant of region definition). RSSI: Evaluated power signal strength (negative values possible). SNR: signal-to-noise ratio. FQ: beacon reception frequency in Hz. TIME: timestamp in seconds (GPS epoch). DESC: Info descriptor - describes how the information field must be interpreted. INFO: Info - more details in chapter "Beacon GwSpecific Field Format" of the document [1].
+EVT:BEACON_NOT_RECEIVED	Notifies the host that the expected RX_BC window does not received any beacon request.
+EVT:BEACON_LOST	Notifies the host that no more beacon have been received for 120 min. This event generates also a revert of class B to A and a new beacon acquisition process.
NVM DATA STORED NVM DATA RESTORED	Notifies the host that the requested store/restore has been successfully executed.

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>
APPLICATION_VERSION: V1.1.0<CR>
MW_LORAWAN_VERSION:  V2.4.0<CR>
MW_RADIO_VERSION:    V1.2.0<CR>
L2_SPEC_VERSION:     V1.0.4<CR>
RP_SPEC_VERSION:     V2-1.0.1<CR>
###### OTAA ######<CR>
###### AppKey:      2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C<CR>
###### NwkKey:      2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C<CR>
###### ABP  ######<CR>
###### AppSKey:     2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C<CR>
###### NwkSKey:     2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C<CR>
###### IDs  ######<CR>
###### DevEui:      00:80:E1:15:xx:xx:xx:xx<CR>
###### AppEui:      01:01:01:01:01:01:01:01<CR>
###### DevAddr:     xx:xx:xx:xx<CR>
ATtention command interface<CR>
AT? to list all available functions<CR>
```

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

3.4.4 AT+RFS - Restore factory settings

Description	Erases LoRaWAN context in flash memory and restore factory settings after a reset.
Syntax	AT+RFS<CR>
Arguments	None
Response	None
Results code	None (NVIC_Reset action)

Examples:

```
/* Example: Restore the default LoRaWAN context */
# AT+RFS<CR>
APPLICATION_VERSION: V1.2.0<CR>
MW_LORAWAN_VERSION:  V2.4.0<CR>
MW_RADIO_VERSION:    V1.2.0<CR>
L2_SPEC_VERSION:     V1.0.4<CR>
RP_SPEC_VERSION:     V2-1.0.1<CR>
###### OTAA ######<CR>
###### AppKey:      2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C<CR>
###### NwkKey:      2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C<CR>
###### ABP  ######<CR>
###### AppSKey:     2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C<CR>
###### NwkSKey:     2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C<CR>
###### IDs  ######<CR>
###### DevEui:      00:80:E1:15:xx:xx:xx:xx<CR>
###### AppEui:      01:01:01:01:01:01:01:01<CR>
###### DevAddr:     xx:xx:xx:xx<CR>
ATtention command interface<CR>
AT? to list all available functions<CR>
```

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

3.4.5 AT+CS - Context store

Description	Stores current LoRaWAN context to FLASH sector.
Syntax	AT+CS<CR>
Arguments	None
Response	None
Result code	<CR><LF>OK<CR><LF> <CR><LF>AT_ERROR<CR><LF>

Examples:

```
/* Example: Store the current LoRaWAN context to FLASH sector */
# AT+CS<CR>
NVM DATA STORED<CR>
<CR>
OK<CR>
```

3.4.6 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>
```

3.4.7 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	APPLICATION_VERSION: Vx.y.z<CR> MW_LORAWAN_VERSION: Va.b.c<CR> MW_RADION_VERSION: Vd.e.f<CR> L2_SPEC_VERSION: VA.B.C<CR> RP_SPEC_VERSION: Vx-D.E.F<CR
Arguments	None
Response	<version><CR><LF>
Result code	<CR><LF>OK<CR><LF>

Example:

```
/* Example: Get the Application and Middleware versions */
#AT+VER=?
APPLICATION_VERSION: V1.2.0<CR>
MW_LORAWAN_VERSION:  V2.4.0<CR>
MW_RADIO_VERSION:    V1.2.0<CR>
L2_SPEC_VERSION:     V1.0.4<CR>
RP_SPEC_VERSION:     V2-1.0.1<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. If the end device is joined, send a new PingSlotInfoReq.
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 $2^{<\text{periodicity}>}$, 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>:<freqdev>:<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 – Rx 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) – 3: MSK • `<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>

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: PA Boost State= 0<CR>
8: modulation LORA<CR>
9: Payload len= 16 Bytes<CR>
10: Frequency deviation not applicable<CR>
11: LowDRopt[0 to 2]= 2<CR>
12 BT product not applicable<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" */
<CR>
OK<CR>

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

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

/* Event: Beacon received -> Now the end-node is Class B "enable" */
+EVT+SWITCH_TO_CLASS_B<CR>

/* An automatic Send PingSlotInfoReq is generated with the default PingSlot periodicity
value */
/* --> MAC Ping Slot Info ANS is received by end-node in hidden way */
# AT+CLASS=?<CR>
B<CR> /*Class B "enable"*/
OK<CR>

...

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

/* example: Get the Local Time updated by the Last Beacon information */
#AT+LTIME=?<CR>
LTIME:01h01m01s on 01/01/2021<CR>
<CR>
OK<CR>
```

4.5 Class C enable request

The example below shows how to do a Class C 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 C mode can be made */
# AT+CLASS=C<CR> /* Request to switch to Class C "enable" */
<CR>
+EVT:SWITCH_TO_CLASS_C<CR>
<CR>
OK<CR>

OK
AT+SEND=10:0:1234<CR> /* This first uplink is required to confirm the synchro with the
Network Server. This uplink is forced at CONFIRMED */
<CR>
OK

+EVT:RX_1, PORT 0, DR 0, RSSI -49, SNR 5<CR> /* Empty downlink with Ack bit enabled */
```

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.	`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). See 'PWR Low Power Mode Selection' in document [3].*

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:

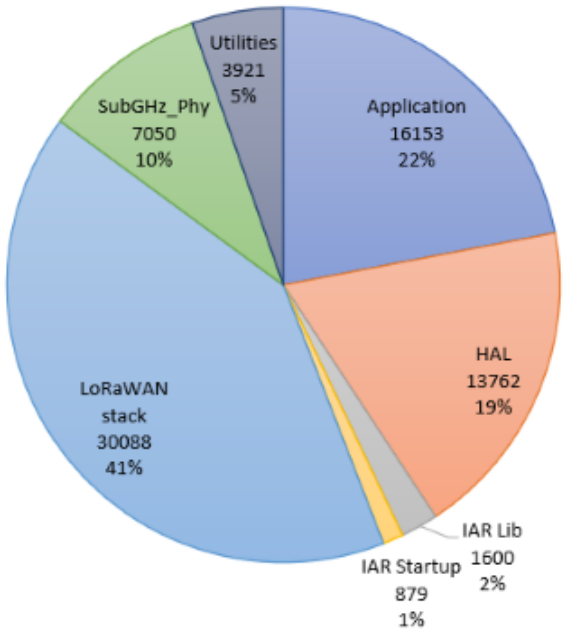
IAR Compiler: EWARM 8.30.1

- Optimization: level 3 for size
- Debug option: off
- Trace option: VLEVEL_M: debug traces enabled
- Target: NUCLEO-WL55JC1
- LoRaMAC Class A
- LoRaMAC region EU868 and US915

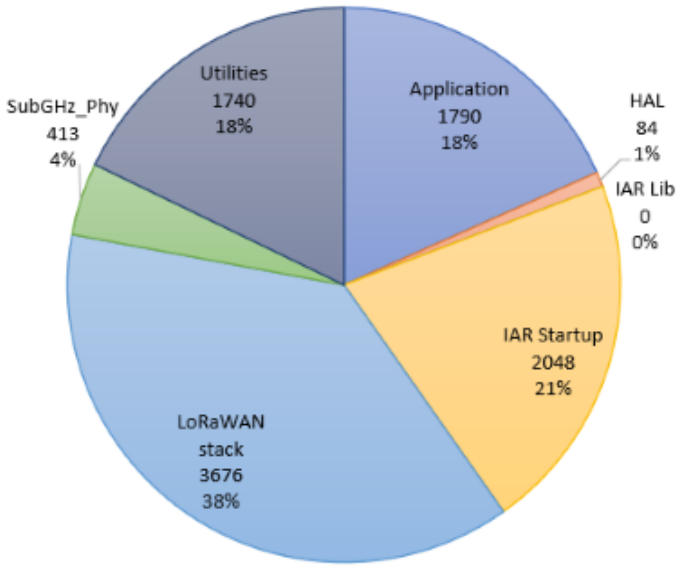
Table 5. Memory footprint detail

Label	RO data (FLASH, bytes)	RW data (RAM, bytes)	Description
Application	16153	1790	User application including `lora_at.c`, `lora_command.c`, and `test_rf.c`
LoRaWAN stack	30088	3676	Middleware LmHandler interface, MAC and Region
SubGHz_Phy	7050	413	Middleware radio interface
Utilities	3921	1740	All STM32 services (sequencer, time server, low-power mgr, trace, and mem)
HAL	13762	84	STM32WL HAL and LL drivers
IAR startup	879	2048	int vector, CSTACK, HEAP and init table
IAR library	1600	0	IAR proprietary libraries
Total memory	**73453**	**9751**	-

Figure 3. **LoRaWAN_AT_Slave memory footprint**



[FLASH] LoRaWAN_AT_Slave



[RAM] LoRaWAN_AT_Slave

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
6-Jul-2021	3	Updated: • Reference documents • Table 2. AT commands • Table 3. Event table • All command descriptions and examples in Section 3.4 to Section 3.9 • Section 3.7.14 AT+PGSLOT - Ping slot moved • Section 5.3 Compilation switches Added: • Section 3.5.2 AT+NWKKEY - Network root key • Section 3.6.2 AT+LINKC - Link check request
21-Feb-2022	4	Updated: • Table 1. Acronyms and terms • Section 3 AT commands • Section 3.2 AT command overview • Section 3.3 Event table • Section 3.4.3 ATZ - MCU reset Added: • Section 3.4.4 AT+RFS - Restore factory settings • Section 3.4.5 AT+CS - Context store • Section 4.4 Class B enable request • Section 4.5 Class C enable request

Contents

1 General information 2
2 Overview 3
3 AT commands 4
3.1 AT_RX_ERROR 5
3.2 AT command overview 6
3.3 Event table 7
3.4 General commands 9
3.4.1 AT 9
3.4.2 AT? 9
3.4.3 ATZ - MCU reset 10
3.4.4 AT+RFS - Restore factory settings 11
3.4.5 AT+CS - Context store 11
3.4.6 AT+VL - Verbose level 12
3.4.7 AT+LTIME - Local time in UTC format 12
3.5 Keys, IDs and EUIs management 13
3.5.1 AT+APPEUI - Application identifier 13
3.5.2 AT+NWKKEY - Network root key 13
3.5.3 AT+APPKEY - Application root key 14
3.5.4 AT+APPSKEY - Application session key 15
3.5.5 AT+NWKSKEY - Network session key 16
3.5.6 AT+DADDR - Device address 16
3.5.7 AT+DEUI - Device EUI 17
3.5.8 AT+NWKID - Network ID 17
3.6 Join and send data on LoRa network 18
3.6.1 AT+JOIN - Join LoRa network 18
3.6.2 AT+LINKC - Link check request 18
3.6.3 AT+SEND - Send data to LoRa network 19
3.7 LoRa network management 20
3.7.1 AT+VER - Firmware version 20
3.7.2 AT+ADR - Adaptive data rate functionality 20
3.7.3 AT+DR - Data rate 21
3.7.4 AT+BAND - Active region 22
3.7.5 AT+CLASS - LoRa class 23
3.7.6 AT+DCS - Duty cycle settings 23
3.7.7 AT+JN1DL - Join delay on Rx window 1 24
3.7.8 AT+JN2DL - Join delay on Rx window 2 24

3.7.9 AT+RX1DL - Delay of the Rx window 1 . . . 25
3.7.10 AT+RX2DL - Delay of the Rx window 2 . . . 25
3.7.11 AT+RX2DR - Data rate of the Rx window 2 . . . 26
3.7.12 AT+RX2FQ - Frequency of the Rx window 2 . . . 26
3.7.13 AT+TXP - Transmit power . . . 27
3.7.14 AT+PGSLOT - Ping slot . . . 27
3.8 Radio test commands . . . 28
3.8.1 AT+TTONE - RF tone test . . . 28
3.8.2 AT+TRSSI - RF RSSI tone test . . . 28
3.8.3 AT+TCONF - LoRa RF test configuration . . . 29
3.8.4 AT+TTX - Packets to be sent for PER RF TX test . . . 31
3.8.5 AT+TRX - Packets to be received for PER RF RX test . . . 32
3.8.6 AT+TTH - RF Tx hopping test . . . 33
3.8.7 AT+CERTIF - Module in LoRaWAN certification with join mode . . . 34
3.8.8 AT+TOFF - RF test . . . 34
3.9 Information . . . 35
3.9.1 AT+BAT - Battery level . . . 35
4 Examples . . . 36
4.1 Join and send in unconfirmed mode . . . 36
4.2 Join and send in confirmed mode . . . 36
4.3 Rx received data . . . 36
4.4 Class B enable request . . . 37
4.5 Class C enable request . . . 38
5 Embedded software description . . . 39
5.1 Firmware overview . . . 39
5.2 LPUART . . . 39
5.3 Compilation switches . . . 39
5.3.1 Debug switch . . . 40
5.3.2 Footprint . . . 40
Revision history . . . 42
List of tables . . . 45
List of figures . . . 46

List of tables

Table 1. Acronyms and terms . . . 2
Table 2. AT commands . . . 6
Table 3. Event table . . . 7
Table 4. Main options for application configuration . . . 39
Table 5. Memory footprint detail . . . 40
Table 6. Document revision history . . . 42

List of figures

Figure 1. Tera Term serial port set up 4
Figure 2. Tera Term terminal setup 4
Figure 3. LoRaWAN_AT_Slave memory footprint 41