環鴻科技股份有限公司
Universal Global Scientific Industrial Co., Ltd.

WM-SG-SM-42
AT Command Reference Manual

Version: 2.5

Amendment Records

Item:	Date:	Revision:	Page:	Change Description:	Changed by:
1	10/25/2016	1.0	All	Initial release	Andy Elise
2	10/26/2016	1.1	All	Correct total company name	Elise
3	11/30/2016	1.4	All	1. AT command modification * Removed Commands: ATP,AT+RSSI,AT+SNR, AT+RSTAT,AT+SF,AT+FRE,AT+TXP,AT_CR,AT+IQP,AT+PL,AT+TXON,AT+RXON,AT+STOP, AT+BW,AT+TONE,AT+RDCT,AT? * New Commands: AT+RF * Modify AT+DR 2. Correct command examples for radio tests	Andy Elise
4	12/16/2016	1.5	All	Add default channel	Elise
5	01/26/2017	1.6	All	1. AT command modification: *new commands: AT+VERB *removed command AT+JSTA AT+CER ATI (integrate to AT+VER) 2. Modified 1.1 and 1.2 for AT command console connection. 3. Add 3.3 event table	Andy Elise
6	02/03/2017	1.7	12,40,41	AT command modification: Re-add simplified AT+JSTA	Elise
7	10/27/2017	1.8	8,14,64,65	1. correct typo 2. add 4.11.4 switch to us915 example & 4.11.5 switch to eu868 example	Andy
8	12/08/2017	1.9		1. correct command example in sections: 3.4.3.3, 4.1, 4.3, 4.5, 4.6	Andy

9	03/05/2018	2.0		Modified the following sections for AS923 regions. 3.4.2.1 3.4.2.3 appendix 3	Andy
10	03/08/2018	2.1		1. update the following sections for AS923 Japan and KR920 region. 3.4.2.1 3.4.2.3 appendix 3 2. add RF test comment in sec. 4.1~4.7.	Andy
11	03/22/2018	2.2		1. add section 3.4.4.10 for CMD AT+CHMSK	Andy
12	07/01/2018	2.3		1. correct the description of command 'AT+PS' on page 15 and page 61	Andy
13	01/22/2019	2.4		1. Add appendix 4. Suspend & resume latency timing	Andy
14	04/08/2019	2.5		1. Modified the following sections for AU915 region (1) 3.4.2.1 Get/Set Device Band (AT+BAND) (2) Appendix 3. Data Rate Configurations in Device 2. Modified the following sections for command AT+VER (1) 3.2 Command Table (2) 3.4.2.2 Get/Set Device Band (AT+VER) 3. Add the following sections for firmware update function: (1) 3.4.8.1 Enable firmware update (AT+DFU) 4. Remove description in section 3.4.7.1	Andy

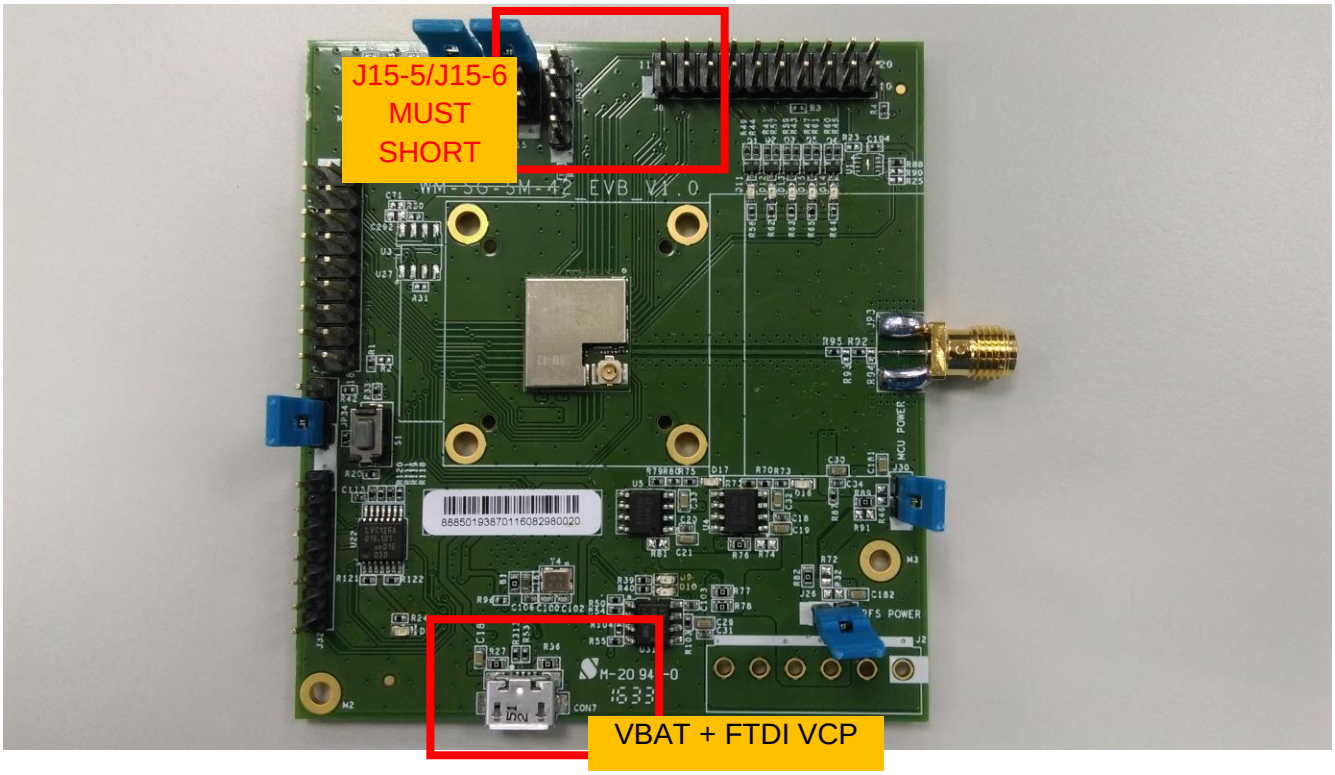
Contents

1. Overview

WM-SG-SM-42 IOT Module has a set of AT-Command for the LoRa RF Test and LoRaWAN Communications. This document briefly describes the usage and examples based on the AT Command.

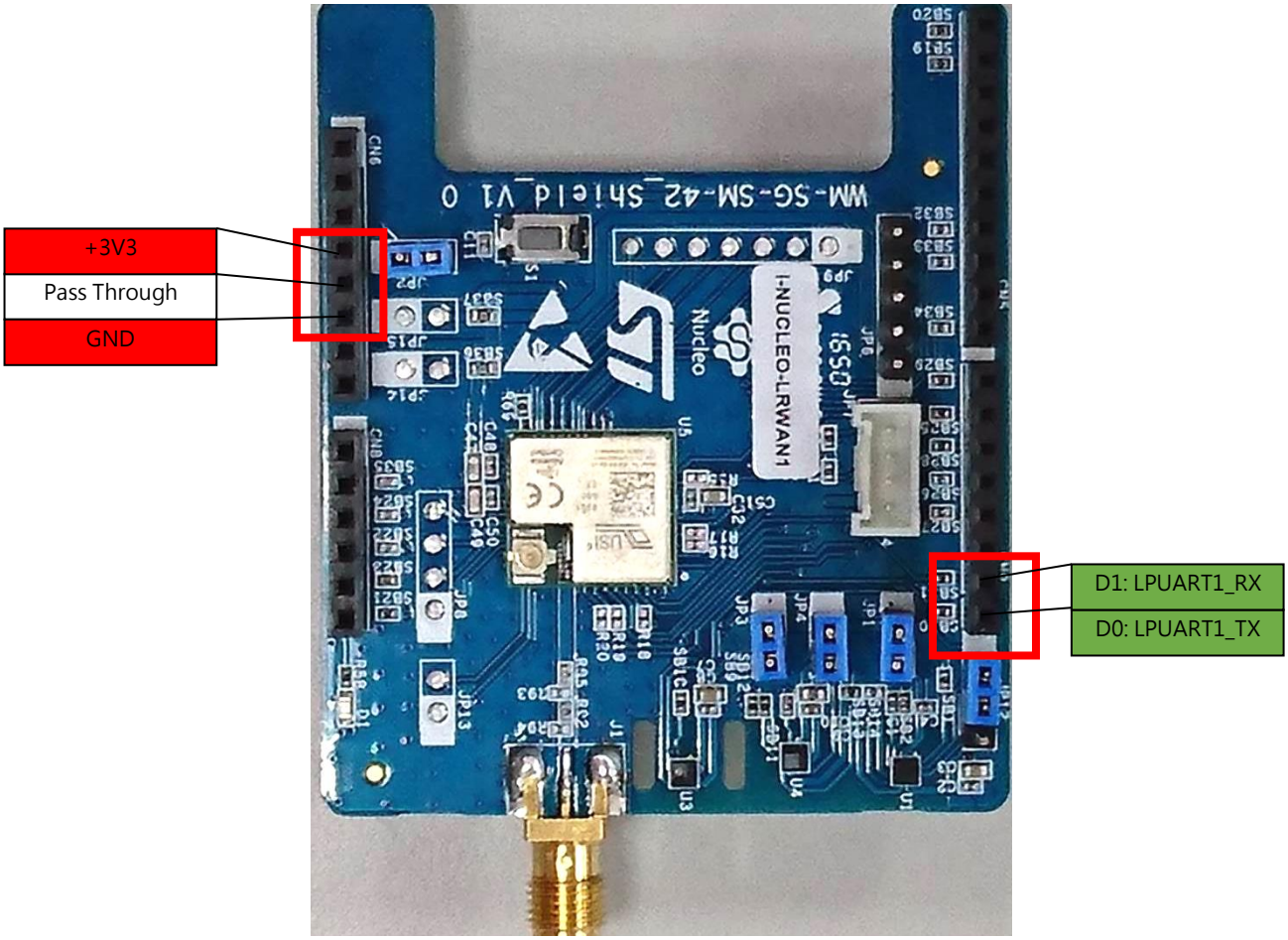
1.1 AT Command Console on WM-SG-SM-42 EVB

WM-SG-SM-42 EVB uses a LPUART interface as the AT command console, the micro USB connector on the SM-SG-SM-42 EVB is the VBAT input and the virtual COM port of LPUART.
The default configuration of the AT command console is 115200,N,8,1, voltage level is 3.3V.
The J15-5/J15-6 and J16-5/J16/6 on the EVB must be short.



1.2 AT Command Console on WM-SG-SM-42 Shield

WM-SG-SM-42 Shield uses a LPUART interface as the AT command console, the D0 and D1 pin is the LPUART TX and LPUART RX, and the default configuration is 115200,N,8,1, CN6 pin 4 and pin 6 are the VBAT input and the voltage level is 3.3V.



2. Default Channel

This chapter lists the default channels for uplink message of LoRaWAN operation.

2.1 Country Code EU

BW:125KHz	
Rate: DR0 to DR5	
Duty cycle:<1%	
No. Ch.	Fre.
0	868.1
1	868.3
2	868.5

2.2 Country Code US

BW:125KHz		BW:125KHz		BW:125KHz		BW:125KHz	
Rate: DR0 to DR3		Rate: DR0 to DR3		Rate: DR0 to DR3		Rate: DR0 to DR3	
No. Ch.	Fre.	No. Ch.	Fre.	No. Ch.	Fre.	No. Ch.	No. Ch.
0	902.3	10	904.3	20	906.3	30	908.3
1	902.5	11	904.5	21	906.5	31	908.5
2	902.7	12	904.7	22	906.7	32	908.7
3	902.9	13	904.9	23	906.9	33	908.9
4	903.1	14	905.1	24	907.1	34	909.1
5	903.3	15	905.3	25	907.3	35	909.3
6	903.5	16	905.5	26	907.5	36	909.5
7	903.7	17	905.7	27	907.7	37	909.7
8	903.9	18	905.9	28	907.9	38	909.9
9	904.1	19	906.1	29	908.1	39	910.1

BW:125KHz		BW:125KHz		BW:125KHz		BW:125KHz	
Rate: DR0 to DR3		Rate: DR0 to DR3		Rate: DR0 to DR3		Rate: DR0 to DR3	
No. Ch.	Fre.	No. Ch.	Fre.	No. Ch.	Fre.	No. Ch.	No. Ch.
40	910.3	50	912.3	60	914.3	64	903
41	910.5	51	912.5	61	914.5	65	904.6
42	910.7	52	912.7	62	914.7	66	906.2
43	910.9	53	912.9	63	914.9	67	907.8
44	911.1	54	913.1			68	909.4
45	911.3	55	913.3			69	911
46	911.5	56	913.5			70	912.6
47	911.7	57	913.7			71	914.2
48	911.9	58	913.9				
49	912.1	59	914.1				

3. Command Reference

The following command reference is based on the firmware version 4.0.

3.1 Command Syntax

The symbol in the documents:

Symbol	Description
<CR>	This indicates a carriage return character, the value is 13 in decimal, 0x0D in hexadecimal.
<LF>	This indicates a linefeed character, the value is 10 in decimal, 0x0A in hexadecimal.
<…>	This indicates the description in this symbol is required in a command.
[…]	This indicates the description in this symbol is optional in a command.

3.2 Command Table

Command	Parameters	Description / Response
General Command		
AT		Check if interface is available to use.
ATE	[=<enabled>]	Enable / Disable local echo 0 = disable 1 = enable
ATZ	(none)	Reset
AT+VERB	[=<enabled>]	Enable/Disable Verbose Response 0 = disable 1 = enable
Command	**Parameters**	**Description / Response**
LoRa MAC Command		
AT+BAND	[=<country code>]	Set / Get device band by country +BAND=<country_code>
AT+VER	(none)	Get LoRaWAN version +VER=<lrwan_ver>,<fw_ver>,<hw>,<bootloader_ver>
AT+DR	[=<data rate>]	Set / Get data rate +DR=<data_rate>
AT+EUI	=<id>	Get / Set module unique ID +EUI=<id>
AT+APPEUI	[=<id>]	Set / Get application identifier (EUI) +APPEUI=<id>
AT+AK	[=<key>]	Set / Get application key +AK=<key>
AT+ADDR	=<address>	Set / Get device address (network_id + network_address) +ADDR=<address>
AT+NSK	[=<key>]	Set / Get network session key +NSK=<key>
AT+ASK	[=<key>]	Set / Get application session key +ASK=<key>
AT+CLASS	[=<class>]	Set / Get LoRaWAN class +CLASS=<class> ※ class B/C now is unsupported.
AT+DC	[=<enabled>]	Set / Get duty cycle +DC=<enabled> 0 = disable duty cycle 1 = enable duty cycle

AT+NTYP	[=<type>]	Set / Get network type +NTYP=<type> 0 = private network type 1 = public network type
AT+ADR	[=<enabled>]	Enable / Disable ADR +ADR=<enabled> 0 = enable 1 = disable
AT+JOIN	=<mode>	Join a network 0 = join gateway with ABP mode 1 = join gateway with OTAA mode
AT+SEND	=<port>,<data>,<ack>	Send LoRaWAN packet port = the port number to the application data = the data with hexadecimal string format ack = 1, confirmed message ack = 0, unconfirmed message
AT+RX1DT	[=<time>]	Set / Get the delay time between the end of the TX and the RX window 1 +RX1DT=<time> ※ The default delay time is 1s.
AT+RX2DT	[=<time>]	Set / Get Rx2 delay time +RX2DT=<time> ※ The default delay time is 2s.
AT+JRX1DT	[=<time>]	Set / Get Join accept RX1 Delay +JRX1DT=<time> ※ The default delay time is 5s.
AT+JRX2DT	[=<time>]	Set / Get Join accept RX2 Delay +JRX2DT=<rx2_delay> ※ The default delay time is 6s.
AT+RX2DR	[=<data_rate>]	Set / Get the data rate for RX window 2 +RX2DR=<rx2_dr> ※ data rate = 0 to 15
AT+JSTA	[=<joined_status>]	Get the joined status +JSTA=<joined_status> 0 = non-joined 1 = joined a network
AT+CHMSK	[=<channel_mask>]	Disable channel number by chMask defined in the each region spec.

		※ data rate = 0 to 15
Command	**Parameters**	**Description / Response**
Radio Test Command		
AT+RF	[<pwr>][,<freq>][,<sf>][,<bw>][,<cr>][,<crcOn>][,<preamble>][,<iq_inverted>][,<fix_len][,<hopOn>][,<hopPeriod>]	Set radio settings
AT+TXT	=<count>[,<payload>]	transmit modulation packet with payload count = 1 to 65536 payload = the payload in hexadecimal, max. length is 64 bytes.
AT+STAT	[=<n><CR>]	Get/Reset packet statistics n = 0 reset packet statistics if no parameter specified: [+STAT=<tx_good_cnt>,<tx_err_cnt>,<rx_good_cnt>,<rx_miss_cnt>,<rx_err_cnt>,<rssi>,<snr>,<ack_cnt>,<uplink_cnt><CR>]
Command	**Parameters**	**Description / Response**
Peripheral Command		
AT+RREG	=<addr>[,<len>]	Read SPI register from sx127x chip addr = register address len = read length (default is 1)
AT+WREG	=<addr>,<data>	Write SPI register to sx127x chip addr = register address data = the data to write
AT+GPIO	=<gpio>[,<level>]	read / set the voltage level on a GPIO +GPIO=<gpio>,<level> gpio = the name of GPIO level = volate level; 0 is low, 1 is high
AT+UART	=<baud>[,<parity>][,<data_bits>][,<stop_bits>][,<flow_ctrl>]	read / set the configuration of UART baud = 2400 to 115200 parity = 0 to 3 (None. Odd, Even) data_bits = 5 to 9 (5 bits to 9 bits) stop_bits = 0 (1 bits) or 1 (2 bits) flow_ctrl = must be 0.

AT+SIIC	=[<speed>][,<mode>][,<addr_size>]	initialize the I2C bus speed = 100000 or 400000 mode = must be 0 addr_size = 0 (7-bits) or 1 (10-bits)
AT+RIIC	=<dev_addr>,<data_addr>[,<len>]	read data from I2C bus +RIIC=<data_in_hex> dev_addr = the device address data_addr = the register address for read len = the data length to be read
AT+WIIC	=<dev_addr>,<data_addr>,<data>	write data to I2C bud dev_addr = the device address data_addr = the register address for read data = a string with hexadecimal format string
AT+RADC	=<channel>	read analog value on specified GPIO +ADC=<channel>,<value> channel = 7 to 9
AT+BAT	(none)	read / set the voltage level on a GPIO +BAT=<bat_level> bat_level = 0 to 254
Command	**Parameters**	**Description / Response**
DCT Command		
AT+WDCT	[=<type>]	update last setting to DCT type = omit or 0 to restore DCT with default.
AT+DEFMODE	[=<op_mode>]	read / set the current operation mode +MODE=<op_mode> 0 = Idle 1 = continue single tone test (for freq-error test) 2 = continue tx test (for tx quality test) 3 = continue rx test (for sensitivity test) 4 = packet tx test (for sensitivity test) 5 = listen mode 6 = LoraWAN mode
Command	**Parameters**	**Description / Response**
Power Control Command		
AT+PS	[=<set_type>,<value>][,<auto_sleep_time>]	read / set the MCU power control set_type = 0 to 2 value = depended on the <set_type> auto_sleep_time = the time to sleep in millisecond
AT+SLEEP	(none)	enter sleep mode +PS=<sleep_mode>,<clock_type>

Command	Parameters	Description / Response
Watchdog Command		
AT+WDG	[=<enabled>]	disable / enable the watchdog +WDG=<enabled> 0 = disable, 1 = enable

3.3 Event Table

The following is the possible event sent from the module to host serial port actively. Host parser may need to handle this event at any time.

Event	Parameters	Description
Radio Test Event		
+TX	=<status>	Notify TX status. 0: TX done without error. Otherwise: error code
LoRa MAC Command		
+JoinAccepted	(none)	Notify host module has been joined on the gateway by OTAA.
+RCV	=<port>,<len>,<payload>	Notify data received. port = application port len = received length payload = received payload
Command	**Parameters**	**Description / Response**
Power Control Event		
+PS	=<sleep_mode>,<clock_type>	Notify power saving mode has been changed sleep_mode = current power saving mode clock_type = current clock setting

3.4 AT Commands

3.4.1 General Commands

3.4.1.1 Check Connection Status (AT)

Description	The command is for checking if the interface between the module and host is ready to use.
Syntax	AT<CR>
Arguments	None
Response	None
Result Code	OK<CR>

Examples:

```
/* Example1: check if interface is ready to use */
# AT<CR>
OK<CR>
#
```

3.4.1.2 Enable/Disable Local Echo (ATE)

Description	The command is for enabling / disabling to transmit the received characters back on the UART interface, or inquiring if local echo is enabled or disabled.
Syntax	ATE[=<n>]<CR>
Arguments	<n> if n is 0, disable the local echo; if n is 1, enable the local echo. default local echo is enabled.
Response	[<n><CR>]
Result Code	OK<CR>

Examples:

```
/* Example1: Inquiring the status of local echo.*/

# ATE<CR>        /* Query status of local echo */
1<CR>            /* module returns 1 that indicates local echo is enabled */
OK<CR>           /* module returns the command error code */
#                /* ready for next command */

/* Example2: Disabling the local echo */
# ATE=0<CR>   /* disable local echo */
OK<CR>           /* module returns the command error code */
#                /* ready for next command */
```

3.4.1.3 Enable/Disable Verbose Response (AT+VERB)

Description	The command is for enabling / disabling the verbose response
Syntax	AT+VERB[=<enabled>]<CR>
Arguments	<enabled> 0 disable 1 enable
Response	[+VERB=<enabled><CR>] (while verbose = 1) [<n><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Query if verbose response is enabled  */

# AT+VERB<CR>        /* Query if verbose response is enabled */
1<CR>                /* module returns 1 that indicates verbose is enabled */
```

```
OK<CR>          /* module returns the command error code */
#               /* ready for next command */

/* Example2:  Disable the verbose */
# AT+VERB=0<CR>   /* disable verbose */
OK<CR>          /* module returns the command error code */
#               /* ready for next command */
```

Examples with brief response:

```
/* Example1: Inquiring the status of local echo.*/

# ATE<CR>       /* Query status of local echo */
+ATE:1<CR>      /* module returns 1 that indicates local echo is enabled */
OK<CR>          /* module returns the command error code */
#               /* ready for next command */
```

3.4.1.4 System Reset (ATZ)

Description	The command is for reset the whole system including radio and microprocessor.
Syntax	ATZ<CR>
Arguments	None
Response	None
Result Code	OK<CR>

Examples:

```
/* Example1: take system reboot */
# ATZ<CR>
```

```
OK<CR>
#
```

3.4.2 MAC Commands

Please make sure the DEFMODE is LoraWAN mode (6) before executing the MAC commands.

3.4.2.1 Get/Set Device Band (AT+BAND)

Description	The command is for set band by country code for LoRaWAN. Note: Need to write to DCT and reset module to enable this setting.
Syntax	AT+BAND[=<country_code>]<CR>
Arguments	<country_code> the country code index, the default is 0 (EU868). 0 = EU868 Band 1 = US915 Band 2 = IN865 Band 3 = AS923 Band 4 = KR920 Band 5 = AU915 Band 64 = TH923 Band (Thailand 923..925) 66 = JP923 Band (Japan 920-923)
Response	[+BAND=<country_code><CR>] (while verbose = 1) [<country_code><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring current country code index for band.*/
# AT+BAND<CR>     /* Query current country code index */
+BAND=0           /* The module returns country code index is 0:EU (868 Band) */
```

```
OK<CR>                  /* module returns the command error code */
#                       /* ready for next command */

/* Example2: Set band by country code index */
/* Note: Need to write to DCT and reset module to enable this settings*/
# AT+BAND=1<CR>         /* set band at US (Band 915) */
OK<CR>                  /* module returns the command error code */
# AT+WDCT<CR>           /* update current setting to DCT */
OK<CR>                  /* module returns the command error code */
# ATZ<CR>               /* reset module */
#
```

Examples with brief response:

```
/* Example1: Inquiring current country code index for band.*/
# AT+BAND<CR>
0
OK<CR>
#
```

3.4.2.2 Get LoRaWAN version (AT+VER)

Description	The command is for read the LoRaWAN version and firmware version.	
Syntax	AT+VER<CR>	
Arguments	None	
Response	[+VER=<lrwan_ver>,<fw_ver>,<hw>,<bootloader_ver><CR>]	(while verbose = 1)
	[lrwan_ver>,<fw_ver>,<hw>,<bootloader_ver><CR>]	(while verbose = 0)
Result Code	OK<CR>	

Examples with verbose response:

```
# AT+VER<CR>
+VER=1.0.1,4.0,SM42,1.0
OK<CR>
#
```

Examples with brief response:

```
# AT+VER<CR>
1.0.1,4.0,SM42,1.0
OK<CR>
#
```

3.4.2.3 Get/Set Default Data Rate (AT+DR)

Description	The command is for set default operating data rate for LoRaWAN.
Syntax	AT+DR[=<data_rate>]<CR>
Arguments	<data_rate> 0..15 (DR0..DR15) Note. The data rate configuration is different on each band, please refer to the Appendix 3 for correct data rate id based on the band that you want to use: Appendix 3.1 EU868 Data Rate Table Appendix 3.2 US915 Data Rate Table Appendix 3.3 IN865 Data Rate Table Appendix 3.4 AS923 Data Rate Table Appendix 3.5 TH923 Data Rate Table Appendix 3.5 JP923 Data Rate Table Appendix 3.5 KR920 Data Rate Table
Response	[+DR=<data_rate><CR>] (while verbose = 1)

	[<data_rate><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring current default data rate.*/
# AT+DR<CR>          /* Query current country code index */
+DR=0                /* The module returns default data rate is DR_0 */
OK<CR>               /* module returns the command error code */
#                    /* ready for next command */

/* Example2: Set band by country code index */
# AT+DR=1<CR>        /* set default data rate is DR_1 */
OK<CR>               /* module returns the command error code */
#                    /* ready for next command */
```

Examples with brief response:

```
/* Example1: Inquiring current default data rate.*/
# AT+DR<CR>
0       /
OK<CR>
#                    /
```

3.4.2.4 Get Device EUI (AT+EUI)

Description	The command is for read end-device identifier (DevEUI) for LoRaWAN. Note: USI will burn the unique IEEE EUI64 at factory.
Syntax	AT+EUI<CR>
Arguments	None

Response	[+EUI=<id><CR>] (while verbose = 1) [<id><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
# AT+EUI<CR>
+EUI=00,11,22,33,44,55,66,77      /* The module returns 8 bytes DevEUI */
OK<CR>                            /* module returns the command error code */
#                                 /* ready for next command */
```

Examples with brief response:

```
# AT+EUI<CR>
00,11,22,33,44,55,66,77
OK<CR>
#
```

3.4.2.5 Get/Set Application EUI (AT+APPEUI)

Description	The command is for set application identifier (AppEUI) for LoRaWAN.
Syntax	AT+APPEUI[=<id>]<CR>
Arguments	<id> It is a 8 bytes value encoded in hexadecimal format string
Response	[+APPEUI=<id><CR>] (while verbose = 1) [<id><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring application identifier.*/
# AT+APPEUI<CR>
+APPEUI=00,11,22,33,44,55,66,77    /* The module returns 8 bytes AppEUI */
OK<CR>                             /* module returns the command error code */
#                                  /* ready for next command */

/* Example2: Set AppEUI */
# AT+APPEUI=1122334455667788<CR>   /* set application identifier value in hexadecimal format strings (8
                                     bytes)*/
OK<CR>                             /* module returns the command error code */
#                                  /* ready for next command */
```

Examples with brief response:

```
/* Example1: Inquiring application identifier.*/
# AT+APPEUI<CR>
00,11,22,33,44,55,66,77
OK<CR>
```

3.4.2.6 Get/Set Application Key (AT+AK)

Description	The command is for set application key (AppKey) for LoRaWAN.
Syntax	AT+AK[=<key>]<CR>
Arguments	<key> It is a 16 bytes value encoded in hexadecimal format string
Response	[+AK=<key><CR>] (while verbose = 1) [<key><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring application key.*/
# AT+AK<CR>
+AK=11,22,33,44,55,66,77,88,99,aa,bb,cc,dd,ee,ff,00      /* The module returns 16 bytes AppKey */
OK<CR>                          /* module returns the command error code */
#                               /* ready for next command */

/* Example2: Set AppKey */
# AT+AKI=00112233445566778899aabbccddeeff<CR>   /* set application key value in hexadecimal format
                                                   strings (16 bytes)*/
OK<CR>                  /* module returns the command error code */
#                       /* ready for next command */
```

Examples with brief response:

```
/* Example1: Inquiring application key.*/
# AT+AK<CR>
11,22,33,44,55,66,77,88,99,aa,bb,cc,dd,ee,ff,00
OK<CR>
#
```

3.4.2.7 Get/Set Device Address (AT+ ADDR)

Description	The command is for set end-device address (DevAddr) for LoRaWAN.
Syntax	AT+ADDR[=<address>]<CR>
Arguments	<address> It is a 4 bytes value encoded in hexadecimal format string
Response	[+ADDR=<address><CR>] (while verbose = 1) [<address ><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring end-device address.*/
# AT+ADDR<CR>
+ADDR=00,11,22,33            /* The module returns 4 bytes DevAddr */
OK<CR>                       /* module returns the command error code */
#                            /* ready for next command */

/* Example2: Set DevAddr */
# AT+ADDR=11223344<CR>   /* set end-device address value in hexadecimal format strings (4 bytes)*/
OK<CR>                       /* module returns the command error code */
#                            /* ready for next command */
```

Examples with brief response:

```
/* Example1: Inquiring end-device address.*/
# AT+ADDR<CR>
00,11,22,33
OK<CR>
#
```

3.4.2.8 Get/Set Network Session Key (AT+NSK)

Description	The command is for set network session key (NwkSKey) for LoRaWAN.
Syntax	AT+NSK[=<key>]<CR>
Arguments	<key> It is a 16 bytes value encoded in hexadecimal format string
Response	[+NSK=<key><CR>] (while verbose = 1) [<key><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring network session key.*/
# AT+NSK<CR>
+NSK=11,22,33,44,55,66,77,88,99,aa,bb,cc,dd,ee,ff,00  /* The module returns 16 bytes NwkSKey */
OK<CR>                          /* module returns the command error code */
#                               /* ready for next command */

/* Example2: Set NwkSKey */
# AT+NSK=00112233445566778899aabbccddeeff<CR>   /* set network session key value in hexadecimal
                                                   format strings (16 bytes)*/
OK<CR>                  /* module returns the command error code */
#                       /* ready for next command */
```

Examples with brief response:

```
/* Example1: Inquiring network session key.*/
# AT+NSK<CR>
11,22,33,44,55,66,77,88,99,aa,bb,cc,dd,ee,ff,00
OK<CR>
#
```

3.4.2.9 Get/Set Application Session Key (AT+ASK)

Description	The command is for set applicant session key (AppSKey) for LoRaWAN.
Syntax	AT+ASK[=<key>]<CR>
Arguments	<key> It is a 16 bytes value encoded in hexadecimal format string
Response	[+ASK=<key><CR>] (while verbose = 1) [<key><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring application session key.*/
# AT+ASK<CR>
+ASK=11,22,33,44,55,66,77,88,99,aa,bb,cc,dd,ee,ff,00      /* The module returns 16 bytes AppSKey */
OK<CR>                          /* module returns the command error code */
#                               /* ready for next command */

/* Example2: Set AppSKey */
# AT+ASK=00112233445566778899aabbccddeeff<CR>   /* set application session key value in hexadecimal
                                                   format strings (16 bytes)*/
OK<CR>                  /* module returns the command error code */
#                       /* ready for next command */
```

Examples with brief response:

```
/* Example1: Inquiring network session key.*/
# AT+ASK<CR>
11,22,33,44,55,66,77,88,99,aa,bb,cc,dd,ee,ff,00
OK<CR>
#
```

3.4.2.10 Get/Set LoRaWAN class (AT+CLASS)

Description	The command is for set the operation mode for LoRaWAN.
Syntax	AT+CLASS[=<class>]<CR>
Arguments	<class> It is a class index for operation mode. The firmware only supports Class A so far. 0 = Class A (baseline) 1 = Class B (beacon) 2 = Class C (continuous)
Response	[+CLASS =<class><CR>] (while verbose = 1)

	[<class><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring the operation mode.*/

# AT+CLASS<CR>
+CLASS=0                /* The module returns in which class */
OK<CR>                  /* module returns the command error code */
#                       /* ready for next command */

/* Set operation mode (Not support yet) */
```

Examples with brief response:

```
# AT+CLASS<CR>
0
OK<CR>
#
```

3.4.2.11 Enable/Disable Duty Cycle (AT+DC)

Description	The command is for set the duty cycle as enabled / disabled. This command is for EU (Band 868) only since the duty cycle is 100% in US (Band 915). Note: Disabling duty cycle for testing only. It should be enabled for shipping
Syntax	AT+DC[=<enabled>]<CR>
Arguments	<enabled> The default is 1 (On). 0 : Indicates the duty cycle is off 1 : Indicates the duty cycle is on

Response	[+DC =<enabled><CR>] (while verbose = 1) [<enabled><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring the duty cycle.*/
# AT+DC<CR>
+DC=1          /* The module returns duty cycle' s status */
OK<CR>              /* module returns the command error code */
#                   /* ready for next command */

/* Example2: Set duty cycle as disabled for EU (Band 868) */
# AT+DC=0<CR>   /*Off duty cycle*/
OK<CR>             /* module returns the command error code */
#                 /* ready for next command */
```

Examples with brief response:

```
/* Example1: Inquiring the duty cycle.*/
# AT+DC<CR>
1
OK<CR>
#
```

3.4.2.12 Get/Set Network Type (AT+NTYP)

Description	The command is for set network type for LoRaWAN
Syntax	AT+NTYP[=<type>]<CR>
Arguments	<type> The default is 1 (Public network).

	0 : Indicates private network type 1 : Indicates public network type
Response	[+NTYP =<type><CR>] (while verbose = 1) [<type><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring the network type.*/
# AT+NTYP<CR>
+NTYP=1                    /* The module returns network type */
OK<CR>                     /* module returns the command error code */
#                          /* ready for next command */

/* Example2: Set network type */
# AT+NTYP=0<CR>    /* Set network type as private*/
OK<CR>                /* module returns the command error code */
#                     /* ready for next command */
```

Examples with brief response:

```
/* Example1: Inquiring the network type.*/
# AT+NTYP<CR>
1
OK<CR>
```

3.4.2.13 Enable/Disable Adaptive Data Rate (AT+ADR)

Description	The command is for set the adaptive data rate as enabled / disabled for LoRaWAN.
Syntax	AT+ADR[=<enabled>]<CR>

Arguments	<enabled> The default is 0 (disabled). 0 : Indicates the ADR is disabled 1 : Indicates the ADR is enable
Response	[+ADR =<enabled><CR>] (while verbose = 1) [<enabled><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring the ADR.*/
# AT+ADR<CR>
+ADR=0              /* The module returns ADR' s status */
OK<CR>              /* module returns the command error code */
#                   /* ready for next command */

/* Example2: Set ADR */
# AT+ADR=1<CR>   /*Enable ADR*/
OK<CR>             /* module returns the command error code */
#                  /* ready for next command */
```

Examples with brief response:

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

3.4.2.14 Join Network (AT+JOIN)

Description	The command is for join network by ABP / OTAA for LoRaWAN.
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	None
Result Code	OK<CR>

Examples

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

/* Example1: Join a network by OTAA */
# AT+JOIN=1<CR>     /* OTAA */
OK<CR>                /* module returns the command error code */
#                     /* ready for next command */
# +JoinAccepted       /* Event : OTAA join successful event */
```

3.4.2.15 Send Packet (AT+SEND)

Description	The command is for transmit application packets with specified and AppPort and payload to the air for LoRaWAN.
Syntax	AT+SEND=<port>,<payload>,<ack><CR>
Arguments	<port> the application port to be transmitted <payload> the payload in hexadecimal format strings, the maximum length is 64 bytes. <ack> 0: Indicates this is a unconfirmed message

	1: Indicates this is a confirmed message.
Response	None
Result Code	OK<CR>

Examples

```
/* Example1: Send a packet  to gateway*/
# AT+SEND=2,000000000000007F0000000000000000,0<CR>  /* send a packet with APP port is 2, battery
                                        level is 50%, unconfirmed message */
OK<CR>                 /* module returns the command error code */
#                      /* ready for next command */

/* Example2: Send a packet  to gateway and received packet from gateway*/
# AT+SEND=2,000000000000007F0000000000000000,0<CR>  /* send a packet with APP port is 2, battery
                                        level is 50%, unconfirmed message */
OK<CR>                   /* module returns the command error code */
#
+RCV=2,1,1             /* Event :Gateway send packet to module for APP port 2 */
                       /* Event :The packet' s payload size is 1 */
                       /* Event : The payload data in hexadecimal format strings */
```

3.4.2.16 Get/Set RX1 Delay Time (AT+ RX1DT)

Description	The command is for set the delay time of RX window 1 for LoRaWAN.
Syntax	AT+RX1DT[=<time>]<CR>
Arguments	<time> It is time value in millisecond, the default is 1000 ms.
Response	[+RX1DT=<time><CR>] (while verbose = 1) [<time><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring the delay time for RX window 1.*/
# AT+RX1DT<CR>
+RX1DT=1000    /* The module returns the delay time of RX windows 1 */
OK<CR>              /* module returns the command error code */
#                    /* ready for next command */

/* Example2: Set delay time */
# AT+RX1DT=2000<CR>   /*set the delay time is 2000 millisecond */
OK<CR>                      /* module returns the command error code */
#                           /* ready for next command */
/* Note: The setting value is at least smaller 1000 ms than the delay time of RX windows 2. */
```

Examples with brief response:

```
/* Example1: Inquiring the delay time for RX window 1.*/
# AT+RX1DT<CR>
1000
OK<CR>
#
```

3.4.2.17 Get/Set RX2 Delay Time (AT+ RX2DT)

Description	The command is for set the delay time of RX window 2 for LoRaWAN.
Syntax	AT+RX2DT[=<time>]<CR>
Arguments	<time> It is time value in millisecond, the default is 2000 ms.
Response	[+RX2DT=< time ><CR>] (while verbose = 1) [<time><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring the delay time for RX window 2.*/
# AT+RX2DT<CR>
+RX2DT=2000    /* The module returns the delay time of RX windows 1 */
OK<CR>              /* module returns the command error code */
#                    /* ready for next command */

/* Example2: Set delay time */
# AT+RX2DT=3000<CR>   /*set the delay time is 3000 millisecond */
OK<CR>                      /* module returns the command error code */
#                           /* ready for next command */
/* Note: The setting value is at least larger 1000 ms than the delay time of RX windows 1*/
```

Examples with brief response:

```
/* Example1: Inquiring the delay time for RX window 2.*/
# AT+RX2DT<CR>
2000
OK<CR>
```

3.4.2.18 Get/Set Join Accept RX1 Delay Time (AT+ JRX1DT)

Description	The command is for set the join accept delay time of RX window 1 for LoRaWAN.
Syntax	AT+JRX1DT[=<time>]<CR>
Arguments	<time> It is time value in millisecond, the default is 5000 ms.
Response	[+JRX1DT=<time><CR>] (while verbose = 1) [<time><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring the delay time for RX window 1.*/
# AT+JRX1DT<CR>
+JRX1DT=5000  /* The module returns the join accept delay time of RX windows 1 */
OK<CR>                    /* module returns the command error code */
#                         /* ready for next command */

/* Example2: Set delay time */
# AT+JRX1DT=6000<CR>   /*set the delay time is 6000 millisecond */
OK<CR>                    /* module returns the command error code */
#                         /* ready for next command */
/* Note: The setting value is at least smaller 1000 ms than the delay time of RX windows 2. */
```

Examples with brief response:

```
/* Example1: Inquiring the delay time for RX window 1.*/
# AT+JRX1DT<CR>
5000
 OK<CR>
#
```

3.4.2.19 Get/Set Join Accept RX2 Delay Time (AT+ JRX2DT)

Description	The command is for set the join accept delay time of RX window 2 for LoRaWAN.
Syntax	AT+JRX2DT[=<time>]<CR>
Arguments	<time> It is time value in millisecond, the default is 6000 ms.
Response	[+JRX2DT=<time><CR>] (while verbose = 1) [<time><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring the delay time for RX window 2.*/
# AT+JRX2DT<CR>
+JRX2DT=6000  /* The module returns the join accept delay time of RX windows 2 */
OK<CR>                    /* module returns the command error code */
#                         /* ready for next command */

/* Example2: Set delay time */
# AT+JRX2DT=7000<CR>   /*set the delay time is 7000 millisecond */
OK<CR>                    /* module returns the command error code */
#                         /* ready for next command */
/* Note: The setting value is at least smaller 1000 ms than the delay time of RX windows 1. */
```

Examples with brief response:

```
/* Example1: Inquiring the delay time for RX window 2.*/
# AT+JRX2DT<CR>
6000
OK<CR>
```

3.4.2.20 Get/Set RX2 Data Rate (AT+ RX2DR)

Description	The command is for set the date rate of RX window 2 for LoRaWAN.
Syntax	AT+RX2DR[=<date_rate>]<CR>
Arguments	<data_rate> The data rate setting, the default is 0 (DR_0) for EU, 8 (DR_8) for US. 0 = DR_0 /* EU868: SF12-BW125, US915: SF10-BW125 */ 1 = DR_1 /* EU868: SF11-BW125, US915: SF9-BW125 */ 2 = DR_2 /* EU868: SF10-BW125, US915: SF8-BW125 */

	3 = DR_3 /* EU868: SF9-BW125, US915: SF7-BW125 */ 4 = DR_4 /* EU868: SF8-BW125, US915: SF8-BW500 */ 5 = DR_5 /* EU868: SF7-BW125, US915: RFU */ 6 = DR_6 /* EU868: SF7-BW250, US915: RFU */ 7 = DR_7 /* EU868: FSK, US915: RFU */ 8 = DR_8 /* US915:SF12-BW500 */ 9 = DR_9 /* US915:SF11-BW500 */ 10 = DR_10 /* US915:SF10-BW500 */ 11 = DR_11 /* US915:SF9-BW500 */ 12 = DR_12 /* US915:SF8-BW500 */ 13 = DR_13 /* US915:SF7-BW500 */ 14 = DR_14 /* US915:RFU */ 15 = DR_15 /* US915:RFU */ Note: EU (Band 868) : RX data rate is DR_0~DR_7 US (Band 915) : RX data rate is DR_8~DR_13
Response	[+RX2DR=<date_rate><CR>] (while verbose = 1) [<date_rate><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring the date rate for RX window 2.*/
# AT+RX2DR<dr>
+RX2DR=3  /* The module returns the data rate of RX windows 2 */
OK<CR>          /* module returns the command error code */
#               /* ready for next command */

/* Example2: Set date rate for RX window 2 */
```

```
# AT+RX2DR=0<CR>       /*set the data rate of RX windows 2 is DR_0 */
OK<CR>                 /* module returns the command error code */
#                      /* ready for next command */
```

Examples with brief response:

```
/* Example1: Inquiring the date rate for RX window 2.*/
# AT+RX2DR<CR>
3
OK<CR>
#
```

3.4.2.21 Get Joined Status (AT+JSTA)

Description	The command is for get the joined status of a network for LoRaWAN.
Syntax	AT+JSTA<CR>
Arguments	None
Response	[+JSTA=<joined_status><CR>] (while verbose = 1) [<joined_status><CR>] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: read current join network status. */
# AT+JOIN=0<CR>
OK<CR>              /* module returns the command error code */
# AT+JSTA<CR>       /* ready for next command */
+JSTA=1             /* joined a network */
OK<CR>              /* module returns the command error code */
#                   /* ready for next command */
```

Examples with brief response:

```
/* Example1: read current join network status */
# AT+JOIN=0<CR>
OK<CR>
# AT+JSTA<CR>
1
OK<CR>
#
```

3.4.3 Radio Test Commands

3.4.3.1 Transmit Text Packet (AT+TXT)

Description	The command is for transmit modulation packets with specified payload to the air. And can also be used for sensitivity testing.
Syntax	AT+TXT=<count>,<payload><CR>
Arguments	<payload> the payload is a string in hexadecimal format, the maximum text length is 64 bytes. the default content of payload is 'TEST_PACKET'
Related Event	+TX: Done<CR> (Host will receive this message while all packets transmitted.)
Result Code	OK<CR>

Examples:

```
/* Example1: transmit 100 packet with current radio configuration */

# AT+TXT=100<CR>  /* transmit 100 packets */
OK<CR>            /* module returns the command error code */
# +TX: Done<CR>   /*  '#'  is for prompt that module is ready for next command */
```

```
                    /*  '+TX: Done'  is for notice all packets is transmitted */

/* Example2: transmit 10 packet with payload content  'HELLO'  */

# AT+TXT=10,48454C4C4F<CR>  /* transmit 10 packets with payload content  'HELLO'  */
OK<CR>                      /* module returns the command error code */
# +TX: Done<CR>             /*  '#'  is for prompt that module is ready for next command */
                            /*  '+TX: Done'  is for notice all packets is transmitted */
```

3.4.3.2 Get Packet Statistics (AT+STAT)

Description	The command is for read the current status of TX/RX counters, RSSI and NSR.
Syntax	AT+STAT[=<type>]<CR>
Arguments	<type> 0 to reset counters value to 0
Response	(while verbose = 0) <tx_good_cnt>,<tx_err_cnt>,<rx_good_cnt>,<rx_miss_cnt>,<rx_err_cnt>,<rssi>,<snr>,<downlink_ack_cnt>,<uplink_cnt><CR> (while verbose = 1) +STAT=<tx_good_cnt>,<tx_err_cnt>,<rx_good_cnt>,<rx_miss_cnt>,<rx_err_cnt>, <rssi>,<snr>,<downlink_ack_cnt>,<uplink_cnt><CR>
Result Code	OK<CR>
Remarks	The meaning of the variable in responses: <tx_good_cnt> the counter of transmitted packets <tx_err_cnt> the counter of packets failed to transmit <rx_good_cnt> the counter of received packets

	<rx_miss_cnt> the counter of received packets <rx_err_cnt> the counter of packets failed to receive <rssi> the RSSI value to the last received packet <snr> the SNR value to the last received packet <downlink_ack_cnt> the number of received ACK packets from gateway <uplink_cnt> the number of transmitted packets to gateway

Examples with verbose response:

```
/* Example1: read TX/RX statistics after transmitted 10 packet */

# AT+TXT=10<CR>    /* transmit 10 packets */
OK<CR>                  /* module returns the command error code */
# +TX: Done<CR>     /*  '#'  is for prompt that module is ready for next command */
                          /*  '+TX: Done'  is for notice all packets is transmitted */
# AT+STAT<CR>                    /* read TX/RX counter status */
+STAT=10,0,0,0,0,0,0,0,0,0<CR>     /* 10 packets was transmitted successfully*/
OK<CR>               /* module returns the command error code */
#                         /* ready for next command */
```

Examples with brief response:

```
/* Example1: read TX/RX statistics after transmitted 10 packet */

# AT+TXT=10<CR>   /* transmit 10 packets */
OK<CR>                  /* module returns the command error code */
# +TX: Done<CR>     /*  '#'  is for prompt that module is ready for next command */
                        /*  '+TX: Done'  is for notice all packets is transmitted */
```

```
# AT+STAT<CR>    /* read TX/RX counter status */
10,0,0,0,0,0,0,0,0    /* 10 packets was transmitted successfully */
OK<CR>              /* module returns the command result code */
#                   /* ready for next command */
```

3.4.3.3 Set Radio Settings (AT+RF)

Description	The command usually be used with command AT+MODE for RF quality testing, use this command to change the radio-related settings, like frequency, spread factor , band width and coding rate, etc. before switching operation mode to continue-tx or continue-rx configuration.
Syntax	AT+RF=[<pwr>][,<freq>][,<sf>][,<bw>][,<cr>][,<crcOn>][,<preamble>][,<iq_inverted>][,<fix_len][,<hopOn>][,<hopPeriod>]<CR>
Arguments	<pwr> the target output power, the default is 20dbm. the configurable range is 5dbm ~ 20dbm <freq> the frequency in Hz, the default is 868000000. The configurable range: 860000000 ~ 1020000000 <sf> the spreading factor setting, the default is 7 (SF7). 6 = SF6 (64 symbol/chip rate) 7 = SF7 (128 symbol/chip rate) 8 = SF8 (256 symbol/chip rate) 9 = SF9 (512 symbol/chip rate) 10 = SF10 (1024 symbol/chip rate) 11 = SF11 (2048 symbol/chip rate) 12 = SF12 (4096 symbol/chip rate) <bw> the bandwidth value, the default is 0 (125KHz). 0 = 125KHz

	1 = 250KHz 2 = 500KHz <cr> the cyclic coding rate setting, the default is 1 (4/5). 1 = 4/5 2 = 4/6 3 = 4/7 4 = 4/8 <crcOn> 1 = Enable CRC generation and check on payload. 0 = Disable CRC generation and check on payload <preamble> the preamble length (symbol length), the default is 5, the configurable range: 5 ~ 65535 <iq_inverted> 1 = that indicates the inverted IQ signal is enabled 0 = that indicates the inverted IQ signal is disabled <fix_len> 1 = fixed payload length 0 = variable payload length <hopOn> 0 = Frf is validated when FSTx or FSRx is requested 1 = Frf is validated triggered when RegFrfLsb is written <hopPeriod> Symbol periods between frequency hops.
Response	None
Result Code	OK<CR>

Examples:

```
/* Example1: transmit packet at 869MHz with SF7, 125K BW, CR1 modulation and 15dbm output power */

# ATZ<CR>    /* Reset module */
# AT+DEFMODE=0<CR>    /* Set Module in Idle Mode (Exit LoraWAN Mode) */
OK<CR>       /* module returns the command error code */
```

```
# AT+RF=15,869000000,7,0,1<CR>     /* set transceiver at 869MHz with  SF7/125K BW/CR1 modulation and
                                       15dbm output power */
OK<CR>                              /* module returns the command error code */
# AT+TXT=1,48454C4C4F<CR>    /* transmit a packet with payload content   'HELLO'   */
OK<CR>                              /* module returns the command error code */
#                                   /* ready for next command */

/* On RX endpoint */
# +RCV=-1, 5,48454C4C4F<CR>    /* module reports a packet received, the payload is encoded in
                                    hexadecimal format string  */
```

3.4.4 Peripheral Commands

3.4.4.1 Get/Set GPIO Level (AT+GPIO)

Description	The command is for set or read the voltage level on a specified GPIO.
Syntax	AT+GPIO=<gpio>[,<level>]<CR>
Arguments	<gpio> an available name of GPIO for wm-sg-sm-42, see Appendix 1. Pin List <level> 1 is for set the specified GPIO at high level. 0 is for set the specified GPIO at low level.
Response	[+GPIO=<gpio>,<level><CR>] (while verbose = 1) [<level><CR>] (while verbose = 0)
Result Code	OK<CR>
Remark	This command will be GET command and just report the voltage level to the specified GPIO if <level> be omitted.

Examples with brief response:

```
/* Example1:  Inquiring the current voltage level on pin PA4 */

# AT+GPIO=PA4<CR>   /* Query the voltage level of PA4 */
1                   /* the module returns the voltage level on PA4 */
OK<CR>              /* module returns the command error code */
#                   /* ready for next command */

/* Example2: Set pin PA4 at high level */

# AT+GPIO=PA4,1<CR>    /* set PA4 at high level */
OK<CR>                 /* module returns the command error code */
#                      /* ready for next command */
```

Examples with verbose response:

```
/* Example1:  Inquiring the current voltage level on pin PA4 */

# AT+GPIO=PA4<CR>   /* Query the voltage level of PA4 */
+GPIO=PA4,1<CR>     /* the module returns the high level on PA4 */
OK<CR>              /* module returns the command error code */
#                   /* ready for next command */
```

3.4.4.2 Read SPI Register (AT+RREG)

Description	The command is for read SPI register value from sx127x chip.
Syntax	AT+RREG=<reg_addr>[,<len>]<CR>
Arguments	<reg_addr> the address of register in sx127x chip, see sx127x datasheet <len> the length of register to be read. the default is read 1 byte if <len> be omitted.

	The maximum readable length is < 64.
Response	+Reg=<value>[,<value>…] (while verbose = 1) <value>[,<value>…] (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: Inquiring the value of RegOpMode in sx127x chip */
# AT+RREG=0x01<CR>     /* Query the value of RegOpMode in sx127x */
+Reg=0x85<CR>          /* the module returns the current RegOpMode is 0x85 */
OK<CR>                 /* module returns the command error code */
#                      /* ready for next command */

/* Example2: read register from address 0x01 to 0x05 */
# AT+RREG=0x01,5<CR>   /* read register data from address 0x01 and read length is 5 */
+Reg=0x85,0x1A,0x0B,0x00,0x52<CR>
OK<CR>                 /* module returns the command error code */
#                      /* ready for next command */
```

Examples with brief response:

```
/* Example1: Inquiring the value of RegOpMode in sx127x chip */
# AT+RREG=0x01<CR>     /* Query the value of RegOpMode in sx127x */
0x85<CR>               /* the module returns the current RegOpMode is 0x85 */
OK<CR>                 /* module returns the command error code */
#                      /* ready for next command */

/* Example2: read register from address 0x01 to 0x05 */
# AT+RREG=0x01,5<CR>   /* read register data from address 0x01 and read length is 5 */
0x85,0x1A,0x0B,0x00,0x52<CR>     /* the value in register 01~05 */
```

```
OK<CR>                    /* module returns the command error code */
#                         /* ready for next command */
```

3.4.4.3 Write SPI Register (AT+WREG)

Description	The command is for write SPI register value to sx127x chip.
Syntax	AT+WREG=<reg_addr>[,<value>]<CR>
Arguments	<reg_addr> the address of register in sx127x chip, see sx127x datasheet <value> the vale to be written, please refer to sx127x datasheet for suitable value.
Response	None
Result Code	OK<CR>

Examples:

```
/* Example1: write 0x01 to the RegOpMode register of sx127x chip */

# AT+WREG=0x01,x01<CR>    /* Write 0x01 to RegOpMode register */
OK<CR>                    /* module returns the command error code */
#                         /* ready for next command */
```

3.4.4.4 Set UART Interface (AT+UART)

Description	The command is for set the configuration of the UART. But the change must be written in to the DCT and would be effective after system reboot.
Syntax	AT+UART=<baud>[,<parity>][,<data_bits>][,<stop_bits>][,<flow_ctrl>]<CR>
Arguments	<baud> the default baud rate is 115200, the acceptable range is 2400 ~ 115200 <parity> 0 is NO PARITY, the default is the no parity 1 is ODD PARITY

	2 EVEN PARITY <data_bits> the default data width is 8 bits, the acceptable range is 5 ~ 9 bits <stop_bits> 0 is 1 stop-bits, the default is the 1 stop-bits. 1 is 2 stop-bits <flow_ctrl> not supported.
Response	None
Result Code	OK<CR>

Examples:

```
/* Example1: change baud rate of UART to 9600 */

# AT+UART=9600<CR>    /* set baud rate of UART to 9600 */
OK<CR>                /* module returns the command error code */
# AT+WDCT<CR>         /* update current setting to DCT */
OK<CR>                /* module returns the command error code */
# ATZ<CR>             /* reset module */
```

3.4.4.5 Set I2C Interface (AT+SIIC)

Description	The command is for initiate the I2C bus.
Syntax	AT+SIIC=[<speed>][,<mode>][,<addr_size>]<CR>
Arguments	<speed> 100000 or 400000, the default clock speed is 400000 <mode> 0 is master mode, the default is the master mode 1 is slave mode, now is not supported <addr_size> 0 is 7-bit address mode, the default is the 7-bit address mode. 1 is 10-bit address mode, now is unsupported

Response	None
Result Code	OK<CR>

Examples:

```
/* Example1: initial I2C bus with default setting: 400K, master mode and 7-bit address mode */

# AT+SIIC<CR>    /* initial I2C bus with default setting */
OK<CR>           /* module returns the command error code */
#                /* ready for next command */
```

3.4.4.6 Read I2C (AT+RIIC)

Description	The command is for read data from the I2C bus.
Syntax	AT+RIIC=<dev_addr>,<data_addr>[,<len>]<CR>
Arguments	<dev_addr> the I2C address for the device, depended on the I2C device <data_addr> the register address for read, depended on the I2C device <len> the data length to be ready
Response	+RIIC=<value> (while verbose = 1) <value[,value][,value][...]> (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: read 2 bytes data from the register address 0, device address 0x92 */

# AT+SIIC<CR>  /* initialize the I2C bus */
```

```
OK<CR>            /* initial done */
# AT+RIIC=0x92,0,2<CR>    /* read  2 bytes data from the I2C bus */
+RIIC= 0x12,0x34<CR>          /* get 0x1234 from the I2C bus */
OK<CR>                       /* read done */
#                           /* ready for next command */
```

Examples with brief response:

```
/* Example1: read 2 bytes data from the register address 0, device address 0x92 */

# AT+SIIC<CR>   /* initialize the I2C bus */
OK<CR>            /* initial done */
# AT+RIIC=0x92,0,2<CR>    /* read  2 bytes data from the I2C bus */
0x12,0x34<CR>                 /* get 0x12, 0x34 from the I2C bus */
OK<CR>                       /* read done */
#                           /* ready for next command */
```

3.4.4.7 Write I2C (AT+WIIC)

Description	The command is for write data to the I2C bus.
Syntax	AT+WIIC=<dev_addr>,<data_addr>,<data><CR>
Arguments	<dev_addr> the I2C address for the device, depended on the I2C device <data_addr> the register address for read, depended on the I2C device <data> the data in hexadecimal string format
Response	None
Result Code	OK<CR>

Examples:

```
/* Example1: write 0x74 to register address 0, device address 0x70 */

# AT+WIIC<CR>   /* initialize the I2C bus */
OK<CR>          /* initial done */
# AT+WIIC=0x70,0,0x74<CR>   /* write  0x74 to the I2C device address 0x70, data address 0 */
OK<CR>                      /* write done */
#                           /* ready for next command */
```

3.4.4.8 Read ADC (AT+RADC)

Description	The command is for read analog value from specified GPIO pin. The resolution of analog converter is 12-bits, so the possible range of analog value is 0 ~ 4096.
Syntax	AT+RADC=<channel><CR>
Arguments	<channel> the available channel number: 7 => pin ADC_IN7/PA7 8 => pin ADC_IN8/PB0 9 => pin ADC_IN9/PB1
Response	+ADC=<channel>,<value><CR> (while verbose = 1) <value><CR> (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: floating the pin ADC_IN7/PA7, then read back the analog value on the PA7*/

# AT+RADC=7<CR>   /* read analog value on PA7 */
+ADC=7,127        /* the module returns the analog value  127 */
```

```
OK<CR>                  /* read done */
#                       /* ready for next command */
```

Examples with brief response:

```
/* Example1: floating the pin ADC_IN7/PA7, then read back the analog value on the PA7*/

# AT+RADC=7<CR>   /* read analog value on PA7 */
127               /* the module returns the analog value  127 */
OK<CR>            /* read done */
#                 /* ready for next command */
```

3.4.4.9 Read Battery Value (AT+BAT)

Description	The command is for read battery level. The battery level is converted from the voltage level on the VREF+, and the battery level is a percentage value to the analog value 1800 ~ 3000, the possible converted battery range is 0 ~ 254.
Syntax	AT+BAT<CR>
Arguments	None
Response	+BAT=<bat_level><CR> (while verbose = 1) <bat_level><CR> (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: read the battery level */

# AT+BAT<CR>   /* read analog value on PA7 */
```

```
+BAT=254   /* the module returns the battery level 254 (=100%, because the range is 0 ~ 254) */
OK<CR>     /* read done */
#          /* ready for next command */
```

Examples with brief response:

```
/* Example1: read the battery level */

# AT+BAT<CR>   /* read analog value on PA7 */
254            /* the module returns the battery level 254 (=100%, because the range is 0 ~ 254) */
OK<CR>     /* read done */
#          /* ready for next command */
```

3.4.4.10 Set Default Channel Mask (AT+CHMSK)

Description	Since this Lora module supported the full channel number of US915, but the number of channel is more than the number of channel that most gateway supported, in case gateway losses data from the unsupported channel, please disable these channel that the gateway unsupported using the command.
Syntax	AT+CHMSK=<channel_mask><CR>
Arguments	<channel_mask> a hexadecimal format string with little endian encoding, the maximum string length is 32 bytes depends on region spec, echo byte indicates an 4 bits-mask. This chMask is corresponding to the chMask mapping that defined in the table of ChMaskCntl in each region spec. [byte1] for ch8..ch5 mask, [byte2] for ch4..ch1 mask [byte3] for ch16..ch13 mask [byte4] for ch12..ch9 mask : :

	[byte29] for ch120..ch117 mask [byte30] for ch116..ch113 mask [byte31] for ch128..ch125 mask [byte32] for ch124..ch121 mask
Response	1. no data response if channel_mask does not specified. 2. if channel_mask specified. +BAT=<channel_mask><CR> (while verbose = 1) <channel_mask><CR> (while verbose = 0)
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: disable ch1, ch5, ch12, ch16 */
# AT+CHMSK=ee77ffffffffffff<CR>  /* disable ch1, ch5, ch12, ch16 */
OK<CR>    /* command done */
# AT+WDCT<CR>            /* update change to eeprom */
OK<CR>      /* command done */
# ATZ<CR>            /* reboot with new setting */
#          /* reboot done */

/* Example2: read current channel mask */
# AT+CHMSK<CR>  /* read out the channel mask */
+CHMSK=ee77ffffffffffff  /* the module returns that ch1, ch5,ch12,ch16 was disabled */
OK<CR>    /* read done */
#            /* ready for next command */
```

Examples with brief response:

```
/* Example1: read current channel mask */
# AT+CHMSK<CR>  /* read current the channel mask setting */
ee77ffffffffffff        /* the module returns that ch1, ch5,ch12,ch16 was disabled */
```

```
OK<CR>     /* read done */
#          /* ready for next command */
```

3.4.5 DCT Commands

3.4.5.1 Update DCT Setting (AT+WDCT)

Description	The command is for update last RADIO/MAC/Watchdog/PowerControl settings into DCT.
Syntax	AT+WDCT[=<type>]<CR>
Arguments	<type> 0 to restore DCT content with default value
Response	None
Result Code	OK<CR>

Examples:

```
/* Example1:  update DCT with current settings */

#AT+FRE=915000000<CR>   /* set radio frequency at 915MHz */
OK<CR>                  /* command done */
# AT+WDCT<CR>    /* update DCT with current settings (including the frequency setting above*/
OK<CR>                /* command done */
#                     /* ready for next command */

/* Example2:  restore DCT with default value */

#AT+WDCT=0<CR>   /* restore DCT with default settings */
OK<CR>              /* command done */
```

```
#                    /* ready for next command */
```

3.4.5.2 Get/Set Default Operation Mode (AT+DEFMODE)

Description	The command is for set the operation mode.
Syntax	AT+DEFMODE[=<op_mode>]<CR>
Arguments	<op_mode> the possible operation mode. 0 = IDLE 1 = continue single-tone test (for frequency error test) 2 = continue tx test (for tx quality test) 3 = continue rx test (for sensitivity test) 4 = text packet test (for sensitivity test) 5 = listen mode (for simple p2p communication) 6 = Lora WAN mode
Response	[+MODE=<op_mode><CR>] (while verbose = 1) [<op_mode><CR>] (while verbose = 0)
Result Code	OK<CR>
Remark	This command will be a GET command and just report current operation mode if <op_mode> be omitted.

Examples with verbose response:

```
/* Example1: set default operation mode to LoraWAN mode.*/

# AT+DEFMODE=6<CR>    /* set default operation mode to LoraWAN mode */
OK<CR>                /* command done */
# AT+WDCT<CR>         /* update to DCT*/
```

```
OK<CR>                          /* command done */
# ATZ<CR>                 /* system reboot */
#                              /* now is in LoraWAN mode */

/* Example2: inquiry current operation mode. */
# AT+DEFMODE<CR>          /* inquiry current operation mode */
+MODE=6                          /* the module returns now is working at Lora WAN mode
OK<CR>                          /* command done */
```

Examples with brief response:

```
/* Example1: inquiry current operation mode. */
# AT+DEFMODE<CR>          /* inquiry current operation mode */
6                                  /* the module returns now is working at Lora WAN mode
OK<CR>                          /* command done */
```

3.4.6 Power Control Commands

3.4.6.1 Get/Set Power Control Settings (AT+PS)

Description	The command is for read or set the sleep mode.
Syntax	AT+PS[=0,<sleep_mode>] <CR>
Arguments	<sleep_mode> 0: No power save, 1: using MCU Stop mode as system sleep mode. 2: select MCU Standby mode as system sleep mode. (this is the default setting)
Response	(while verbose = 1)

	[+PS=<sleep_mode>,1,0,0<CR>] (while verbose = 0) [<sleep_mode>,1,0,0<CR>]
Result Code	OK<CR>

Examples with verbose response:

```
/* Example1: set standby mode as sleep mode */

# AT+PS=0,2<CR>   /* set standby mode as sleep mode */
OK<CR>                /* set done */
#                      /* ready for next command */

/* Example2: inquiry current sleep mode setting. */
# AT+PS<CR>   /* query current sleep mode setting */
+PS=2,1,0,0     /* the module returns current sleep mode setting */
                    /* sleep mode = 2   (STANDBY MODE) */
                    /* system clock type = 1   (MSI) */
                   /* auto sleep = 0   (unused) */
                   /* auto sleep time = 0  (unused)*/
OK<CR>            /* read done */
#                    /* ready for next command */
```

Examples with brief response:

```
/* Example1: inquiry current power control setting. */
# AT+PS<CR>     /* query current power control setting */
2,1,0,0             /* the module returns current sleep mode setting */
```

```
OK<CR>          /* read done */
#               /* ready for next command */
```

3.4.6.2 Enter Sleep Mode (AT+SLEEP)

Description	The command is for enter sleep mode immediately based on power control setting. Host can control the sleep mode by using this command immediately.
Syntax	AT+SLEEP<CR>
Arguments	None
Event	+PS=<sleep_mode>,<clock_type><CR>
Result Code	OK<CR>

Examples:

```
/* Example1: enter sleep mode based on current power control settings */

# AT+SLEEP<CR>  /* enabling auto sleep while system idle over 5 seconds */
+PS=1,0<CR>     /* notice host that MCU is going to sleep (enter STOP mode) */
(Module is in sleep until watchdog counter to be 0, or UART received input, or interrupt on wakeup pin )
+PS=0,0<CR>     /* notice host that MCU is waked up from sleep mode and work with HSI clock source */
OK<CR>          /* command done */
#               /* ready for next command */
```

3.4.7 Watchdog Commands

3.4.7.1 Enable/Disable Watchdog (AT+WDG)

removed from firmware version 4.0.

3.4.8 Update Firmware Commands

3.4.8.1 Enable firmware update (AT+DFU)

Description	From version 4.0, SM42 supports firmware update, a firmware binary can be transmitted and programmed via UART with XMODEOM/CRC protocol. This command is used to set device in the firmware update mode.
Syntax	AT+DFU<CR>
Arguments	None
Event	+DFU=0<CR>
Result Code	OK<CR>
Remark	Once the firmware update mode was activated, SM42 will response a '>' to indicating now is in firmware update mode, then will request host to start the transmission by sending 'C' every 4 seconds until received the first packet from host or a timeout about 100 seconds around, a 'EOT' can used to stop the request and exit the firmware update mode. The following are the detailed about the XMODEM/CRC: **One packet length:** 133 bytes (including header, data and checksum as below) <SOH>,<block#>,<~block#>,<data byte0>,.....,<data byte127>,<checksum> **Data size:** 128 bytes **Control Characters:** SOH = 0x01 EOT = 0x04 ACK = 0x06 NAK = 0x15 REQ = 0x43 **Checksum:** CRC-16 with polynomial 0x1021

```
/* Example1:  update firmware through the UART  */

# AT+DFU<CR>   /* set SM42 in firmware update mode  */
OK<CR>          /* command done */
>C              /* a prompt indicates now is in firmware update mode,  */
/* transmit firmware binary with XMODEM protocol by the UART */
+DFU=0<CR>   /* SM42 response +DFU=0 when the transmission and programming completed */
# ATZ           /* reboot device with the new firmware */
# AT+VER        /* check if the version the what you transmitted  */
```

```
/* Example2: Cancel the firmware update */

# AT+DFU<CR>   /* set SM42 in firmware update mode  */
OK<CR>          /* command done */
>               /* a prompt indicates now is in firmware update mode */
/* send EOT on the UART to stop the firmware update mode */
+DFU=0<CR>   /* SM42 response +DFU=0 when the transmission was cancelled */
# ATZ           /* reboot device  */
```

1. Command Examples

1.1 Start Continue-TX for RF tests

The following command sequence is for generating a continuous waveform with on frequency 868.0MHz:

```
# ATZ                    /* SYSTEM RESET */
# AT+DEFMODE=0    /* Set Module in Idle Mode (Exit LoraWAN Mode) */
/* SET RF = 868MHz SF7/125KHz BW/CR1/CRC On/Preamble Length 8/No-IQ-Inverted/20dbm output power */
# AT+RF=20,868000000,7,0,1,1,8,0
# AT+DEFMODE=2    /* START CONTINUE-TX */
```

1.2 Stop Continue-TX for RF tests

Using reset command to stop transmitter/receiver.

```
# ATZ        /* SYSTEM RESET */
```

1.3 Start Single Tone for RF tests

The following command sequence is for generating a continuous waveform without modulation on frequency 868.0MHz:

```
# ATZ                          /* SYSTEM RESET */
# AT+DEFMODE=0    /* Set Module in Idle Mode (Exit LoraWAN Mode) */
# AT+RF=20,868000000    /* SET RF @ 868MHz */
# AT+DEFMODE=1          /* START SINGLE TONE */
```

1.4 Stop Single Tone

Using reset command to stop transmitter/receiver.

```
# ATZ            /* SYSTEM RESET */
```

1.5 Start Continue-RX for RF tests

The following command sequence is for set device in a silent receiving mode on frequency 868.0MHz:

```
# ATZ                       /* SYSTEM RESET */
# AT+DEFMODE=0    /* Set Module in Idle Mode (Exit LoraWAN Mode) */
/* SET RF = 868MHz SF7/125KHz BW/CR1/CRC On/Preamble Length 8/No-IQ-Inverted/20dbm output power */
# AT+RF=20,868000000,7,0,1,1,8,0
# AT+DEFMODE=3        /* START CONTINUE-RX MODE */
```

1.6 Generate TX Packet for RF tests

The following command sequence is for generating waveform with specified payload on frequency 868.0MHz:

```
# ATZ                              /* SYSTEM RESET */
# AT+DEFMODE=0    /* Set Module in Idle Mode (Exit LoraWAN Mode) */
/* SET RF = 868MHz SF7/125KHz BW/CR1/CRC On/Preamble Length 8/No-IQ-Inverted/20dbm output power */
# AT+RF=20,868000000,7,0,1,1,8,0
# AT+DEFMODE=4                /* START PACKET TRANSMISSION MODE */
# AT+TXT=100,48454C4C4F      /* Generate 100 TX PACKET CONTENT ‘HELLO’ */
```

1.7 Check TX/RX Statistics

Command AT+STAT can be used for checking the TX/RX statistics for RF tests.

```
# AT+STAT                       /* INQUIRY THE STATISTIC OF TX/RX COUNTERS */
+STAT=0,0,99,0,1,0,0,0,0     /* PACKETS NUMBER BE TRANSMITED TO CONCERNTRACTOR*/
```

1.8 Stop Continue-RX

Using reset command to stop transmitter/receiver.

```
# ATZ                    /* SYSTEM RESET */
```

1.9 Set GPIO Output Level

```
# AT+GPIO=PA7,1    /* SET GPIO-PA7 AT HIGH LEVEL, 0: LOW LEVEL, 1: HIGH LEVEL */
```

1.10 Read GPIO Input Level

```
# AT+GPIO=PA7      /* READ GPIO-PA7 INPUT LEVEL */
+GPIO=PA7,1        /* REPORT GPIO PA7 IS AT HIGH LEVEL, 0: LOW LEVEL, 1: HIGH LEVEL */
```

1.11 Communicate with Gateway

1.11.1 ABP

```
# ATZ                                                  /* reset module */
# AT+DC=0                                              /* (optional) disable duty cycle for test */
# AT+DR=3                                              /* set DR3 for TX window */
# AT+RX2DR=3                                           /* set DR3 for RX2 window */
# AT+ADDR=11223344                                     /* set device address */
# AT+NSK=112233445566778899aabbccddeeff00              /* set network session key */
# AT+ASK=112233445566778899aabbccddeeff00              /* set application session key */
# AT+WDCT                                              /* (optional) save changes to eeprom */
# AT+JOIN=0                                            /* join gateway with ABP protocol */
# AT+SEND=2,000000000000007F0000000000000000,0         /* APP port :2, battery level 50%, unconfirmed
                                                       message*/
```

1.11.2 OTAA

```
# ATZ                                                  /* reset module */
# AT+DC=0                                              /* (optional) disable duty cycle for test */
# AT+DR=3                                              /* set DR3 for TX window */
# AT+RX2DR=3                                           /* set DR3 for RX2 window */
# AT+APPEUI=1122334455667788                           /* set application EUI */
# AT+AK=112233445566778899aabbccddeeff00               /* set application key */
# AT+JOIN=1                                            /* join gateway with OTAA protocol */
+JoinAccepted                                          /* Event : OTAA join successful event */
# AT+SEND=2,000000000000007F0000000000000000,1         /* APP port :2, battery level 50%, confirmed
                                                       message*/
```

1.11.3 Send data by OTAA and receive data from gateway

```
# AT+JOIN=1
```

```
# AT+SEND=2,000000000000007F0000000000000000,1  /*APP port :2, battery level 50%, confirmed
                                                  message*/
+RCV=2,1,1          /* Gateway send packet to module for APP port 2 */
                    /* The packet's payload size is 1 */
                    /* The payload data in hexadecimal format strings */
```

1.11.4 Switch to US915 example

```
# ATZ               /* reset module */
# AT+BAND=1         /* set us915 band  */
# AT+WDCT           /* save status to eeprom */
# ATZ               /* reset module for APB or OTAA operations  */
```

1.11.5 Switch to EU868 example

```
# ATZ               /* reset module */
# AT+BAND=0         /* set eu868 band  */
# AT+WDCT           /* save status to eeprom */
# ATZ               /* reset module for APB or OTAA operations  */
```

Appendix 1. Pin List

PIN	PIN NAME	GPIO_NAME	DEFAULT FUNCTION
1	GND		
2	RCC_MCO	PA8	
3	M_LPUART_TX	PB10	CONSOLE-TX
4	LPTIM1_OUT	PB2	
5	ADC_IN7	PA7	EVB_LED
6	COMP1_INP	PA1	EVB_LED
7	VREF+		
8	GND		
9	VDDA		
10	LPTIM1_IN1	PB5	
11	SYS_WKUP1	PA0	
12-19	GND		
20	MICRO_RST_N		
21	M_USART1_TX/RST	PA9	RF_RESET
22	M_USART2_TX/DIO0	PA2	RF_DIO0
23	M_USART2_RX/DIO1	PA3	RF_DIO1
24	GPIO_2	PA5	RF_DIO2
25	GPIO_3	PA6	RF_DIO3
26	M_USART1_RX/DIO4	PA10	RF_DIO4
27	DIO5		RF_DIO5
28	BOOT0		
29	GND		
30-31	VDD_RFS		
32	GND		
33-34	VDD_3V3		
35	GND		
36	SPI1_SCK	PB3	RF_SPI

37	SPI1_MISO	PB4	RF_SPI
38	SPI1_MOSI	PA12	RF_SPI
39	SPI1_NSS	PA15	RF_SPI
40-42	GND		
43	RF_SW_CTRL2	PB8	RF_ANT
44	RF_SW_CTRL1	PA4	RF_ANT
45-46	GND		
47	RF_OUT		
48	GND		
49	I2C1_SDA	PB7	
50	I2C1_SCL	PB6	
51	SYS_SWDIO	PA13	PROGRAMMING
52	SYS_SWCLK	PA14	PROGRAMMING
53	ADC_IN9	PB1	EVB_LED
54	COMP1_OUT	PA11	EVB_LED
55	M_LPUART_RX	PB11	CONSOLE-RX
56	ADC_IN8	PB0	EVB_LED
57-81	GND		

Appendix 2. Error Code List

ERROR CODE	ASCII STRING	MEANING
0	OK	no error occurs.
-1	ERROR	unknown error
-2	ERROR_UNKNOW_COMMAND	unsupported command
-3	ERROR_LESS_ARGUMENTS	the number of argument is not enough.
-4	ERROR_MORE_ARGUMENETS	the number of argument is too much.
-5	ERROR_INVALID_ARGUMENTS	invalid content of argument
-6	ERROR_NOT_SUPPORTED	function or argument not supported
-7	ERROR_OUT_OF_RANGE	argument out of range
-8	ERROR_RX_TIMEOUT	receive timeout
-9	ERROR_RX_ERROR	receive error
-10	ERROR_TX_TIMEOUT	transmit timeout
-11	ERROR_TX_ERROR	transmit error
-12	ERROR_RF_BUSY	radio is busy for another transmit or receive
-13	ERROR_TIMEOUT	a timeout occurs in a command process
-14	ERROR_NO_ARGUMENETS_NEEDED	command does not need any argument
-15	ERROR_HAL_ERROR	HAL occurs error.
-16	ERROR_INVALID_HEX_FORMAT	specified hexadecimal string is invalid
-17	ERROR_OUT_OF_ADDRESS	address/id/number out of range
-100	ERROR_WAN_SEND	transmit uplink data fail (busy or duty cycle)
-101	ERROR_WAN_GETPARAM	get the parameter of LoRaWAN fail
-102	ERROR_WAN_SETPARAM	set the parameter of LoRaWAN fail
-103	ERROR_WAN_NON_JOINED	didn’t join to gateway

Appendix 3. Data Rate Configurations in Device

EU868 Data Rate Table

Data Rate	Configuration	Bit Rate	ID (for AT command)
DR0	LoRa: SF12 / 125KHz	250	0
DR1	LoRa: SF11 / 125KHz	440	1
DR2	LoRa: SF10 / 125KHz	980	2
DR3	LoRa: SF9 / 125KHz	1760	3
DR4	LoRa: SF8 / 125KHz	3125	4
DR5	LoRa: SF7 / 125KHz	5470	5
DR6	LoRa: SF7 / 250KHz	11000	6
DR7	FSK: 50Kbps	50000	7
DR8 ~ DR15	RFU		8..15

US915 Data Rate Table

Data Rate	Configuration	Bit Rate	ID (for AT command)
DR0	LoRa: SF10 / 125KHz	980	0
DR1	LoRa: SF9 / 125KHz	1760	1
DR2	LoRa: SF8 / 125KHz	3125	2
DR3	LoRa: SF7 / 125KHz	5470	3
DR4	LoRa: SF8 / 500KHz	12500	4
DR5..DR7	RFU	RFU	5..7
DR8	LoRa: SF12 / 500KHz	980	8
DR9	LoRa: SF11 / 500KHz	1760	9
DR10	LoRa: SF10 / 500KHz	3900	10
DR11	LoRa: SF9 / 500KHz	7000	11

DR12	LoRa: SF8 / 500KHz	12500	12
DR13	LoRa: SF7 / 500KHz	12900	13
DR14..DR15	RFU	RFU	14..15

IN865 Data Rate Table

Data Rate	Configuration	Bit Rate	ID (for AT command)
DR0	LoRa: SF12 / 125KHz	250	0
DR1	LoRa: SF11 / 125KHz	440	1
DR2	LoRa: SF10 / 125KHz	980	2
DR3	LoRa: SF9 / 125KHz	1760	3
DR4	LoRa: SF8 / 125KHz	3125	4
DR5	LoRa: SF7 / 125KHz	5470	5
DR6	RFU	RFU	RFU
DR7	FSK: 50Kbps	50000	7
DR8 ~ DR15	RFU		8..15

AS923 Data Rate Table

Data Rate	Configuration	Bit Rate	ID (for AT command)
DR0	LoRa: SF12 / 125KHz	250	0
DR1	LoRa: SF11 / 125KHz	440	1
DR2	LoRa: SF10 / 125KHz	980	2
DR3	LoRa: SF9 / 125KHz	1760	3
DR4	LoRa: SF8 / 125KHz	3125	4
DR5	LoRa: SF7 / 125KHz	5470	5
DR6	LoRa: SF7 / 250KHz	11000	6
DR7	FSK: 50Kbps	50000	7

DR8 ~ DR15	RFU		8..15

TH923 Data Rate Table (Thailand 923..925)

Data Rate	Configuration	Bit Rate	ID (for AT command)
DR0	LoRa: SF12 / 125KHz	250	0
DR1	LoRa: SF11 / 125KHz	440	1
DR2	LoRa: SF10 / 125KHz	980	2
DR3	LoRa: SF9 / 125KHz	1760	3
DR4	LoRa: SF8 / 125KHz	3125	4
DR5	LoRa: SF7 / 125KHz	5470	5
DR6	LoRa: SF7 / 250KHz	11000	6
DR7	FSK: 50Kbps	50000	7
DR8 ~ DR15	RFU		8..15

JP923 Data Rate Table (Japan 920..923)

Data Rate	Configuration	Bit Rate	ID (for AT command)
DR0	LoRa: SF12 / 125KHz	250	0
DR1	LoRa: SF11 / 125KHz	440	1
DR2	LoRa: SF10 / 125KHz	980	2
DR3	LoRa: SF9 / 125KHz	1760	3
DR4	LoRa: SF8 / 125KHz	3125	4
DR5	LoRa: SF7 / 125KHz	5470	5
DR6	LoRa: SF7 / 250KHz	11000	6
DR7	FSK: 50Kbps	50000	7
DR8 ~ DR15	RFU		8..15

KR920 Data Rate Table

Data Rate	Configuration	Bit Rate	ID (for AT command)
DR0	LoRa: SF12 / 125KHz	250	0
DR1	LoRa: SF11 / 125KHz	440	1
DR2	LoRa: SF10 / 125KHz	980	2
DR3	LoRa: SF9 / 125KHz	1760	3
DR4	LoRa: SF8 / 125KHz	3125	4
DR5	LoRa: SF7 / 125KHz	5470	5
DR6 ~ DR15	RFU		6..15

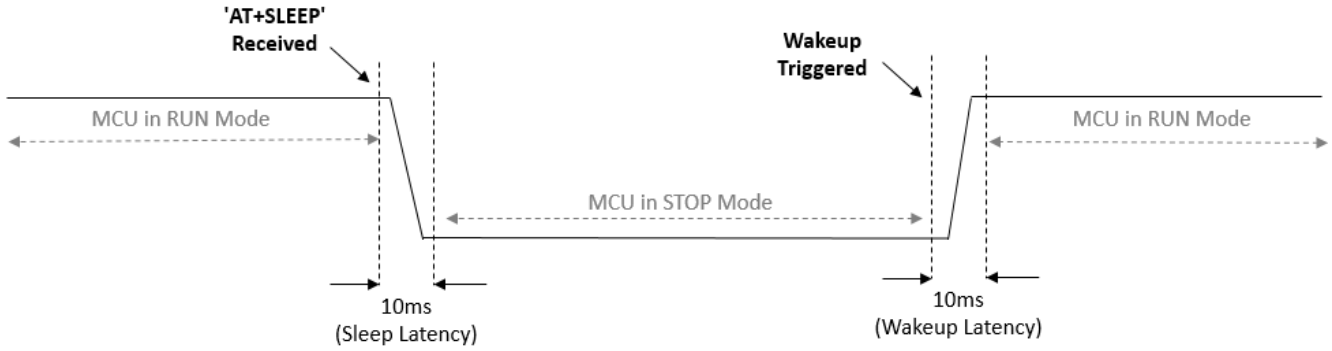
AU915 Data Rate Table

Data Rate	Configuration	Bit Rate	ID (for AT command)
DR0	LoRa: SF12 / 125KHz	250	0
DR1	LoRa: SF11 / 125KHz	440	1
DR2	LoRa: SF10 / 125KHz	980	2
DR3	LoRa: SF9 / 125KHz	1760	3
DR4	LoRa: SF8 / 125KHz	3125	4
DR5	LoRa: SF7 / 125KHz	5470	5
DR6	LoRa: SF8 / 500KHz	12500	6
DR7	RFU	RFU	7
DR8	LoRa: SF12 / 500KHz	980	8
DR9	LoRa: SF11 / 500KHz	1760	9
DR10	LoRa: SF10 / 500KHz	3900	10
DR11	LoRa: SF9 / 500KHz	7000	11
DR12	LoRa: SF8 / 500KHz	12500	12
DR13	LoRa: SF7 / 500KHz	12900	13

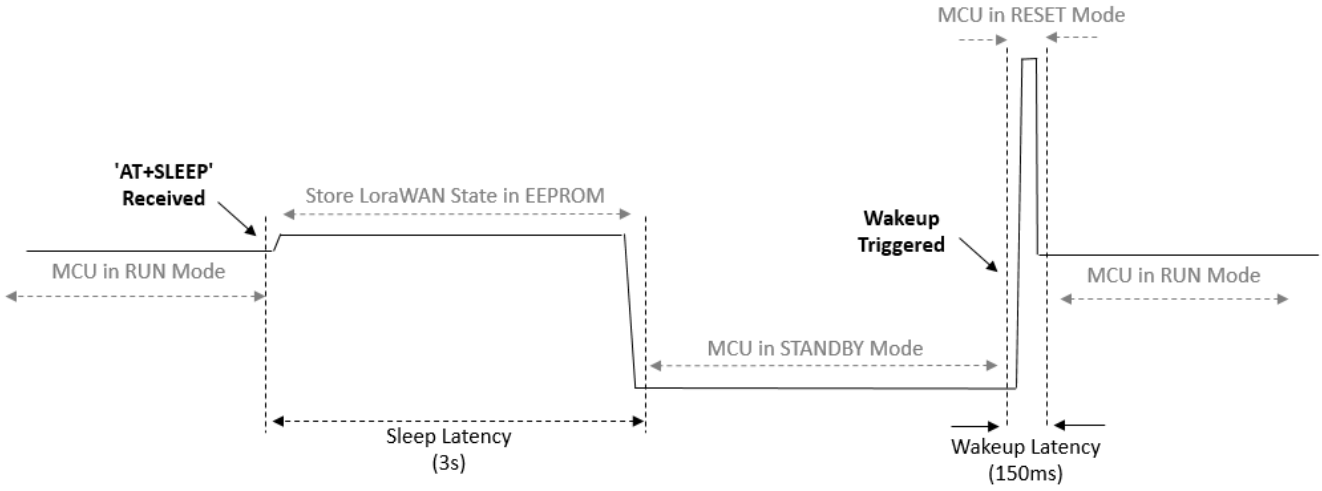
DR14..DR15	RFU	RFU	14..15

Appendix 4. Suspend & Resume Latency Timing

STOP Mode



STANDBY Mode



===== END OF DOCUMENT =====