RN2903 SA (AU915) LoRa® Technology Module Command Reference User's Guide

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

ISBN: 978-1-5224-5043-6

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Table of Contents

Preface

NOTICE TO CUSTOMERS

All documentation becomes dated, and this manual is no exception. Microchip tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our web site (www.microchip.com) to obtain the latest documentation available.

Documents are identified with a "DS" number. This number is located on the bottom of each page, in front of the page number. The numbering convention for the DS number is "DSXXXXXXXXA", where "XXXXXXXX" is the document number and "A" is the revision level of the document.

For the most up-to-date information on development tools, see the MPLAB IDE online help. Select the Help menu, and then Topics to open a list of available online help files.

INTRODUCTION

This chapter contains general information that will be useful to know before using the RN2903 SA module. Topics discussed in this chapter include:

- Document Layout
- Conventions Used in this Guide
- Recommended Reading
- Revision History

DOCUMENT LAYOUT

This command reference user's guide provides information for configuring the RN2903 SA low-power long-range LoRa technology transceiver module, including a description of communication and command references. The document is organized as follows:

- **Chapter 1. "Introduction"** – Introduces the RN2903 SA module and provides a brief overview of its features.
- **Chapter 2. "Command Reference"** – Provides information on the commands used to configure the RN2903 SA module with examples.
- **Chapter 3. "Bootloader Usage"** – Provides the information on the bootloader usage and protocol commands.
- **Appendix A. "Current Firmware Features and Fixes"** – Provides information on the release notes for each revision of the firmware.
- **"The Microchip WebSite"** - Website information, Customer Change Notifications and Customer Support
- **"Worldwide Sales and Service"** - Worldwide Sales and Service Locations and Contact Information

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic characters	Referenced books	*MPLAB® IDE User's Guide*
	Emphasized text	...is the *only* compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic text with right angle bracket	A menu path	*File>Save*
Bold characters	A dialog button	Click **OK**
	A tab	Click the **Power** tab
N'Rnnnn	A number in verilog format, where N is the total number of digits, R is the radix and n is a digit.	4'b0010, 2'hF1
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier New font:		
Plain Courier New	Sample source code	`#define START`
	Filenames	`autoexec.bat`
	File paths	`c:\mcc18\h`
	Keywords	`_asm, _endasm, static`
	Command-line options	`-Opa+, -Opa-`
	Bit values	`0, 1`
	Constants	`0xFF, 'A'`
Italic Courier New	A variable argument	*`file`*`.o`, where *`file`* can be any valid filename
Square brackets [ ]	Optional arguments	`mcc18 [options]` *`file`* `[options]`
Curly brackets and pipe character: { | }	Choice of mutually exclusive arguments; an OR selection	`errorlevel {0|1}`
Ellipses...	Replaces repeated text	`var_name [, var_name...]`
	Represents code supplied by user	`void main (void) { ... }`

RECOMMENDED READING

This command reference user's guide describes how to configure the RN2903 SA module. The module-specific data sheet contains current information on the module specifications. Other useful documents are listed below. The following documents are available and recommended as supplemental reference resources:

RN2903 Low-Power Long-Range LoRa® Technology Transceiver Module Data Sheet (DS50002390)

This data sheet provides detailed specifications for the RN2903 SA module.

LoRa® Alliance: LoRaWAN® Specification V1.0.1

This document describes the LoRaWAN® protocol, which is optimized for battery-powered end devices. This specification is available from the LoRa Alliance at www.lora-alliance.org.

To obtain any of Microchip's documents, visit the Microchip web site at www.microchip.com.

REVISION HISTORY

Revision A (September 2019)

Initial document release.

Chapter 1. Introduction

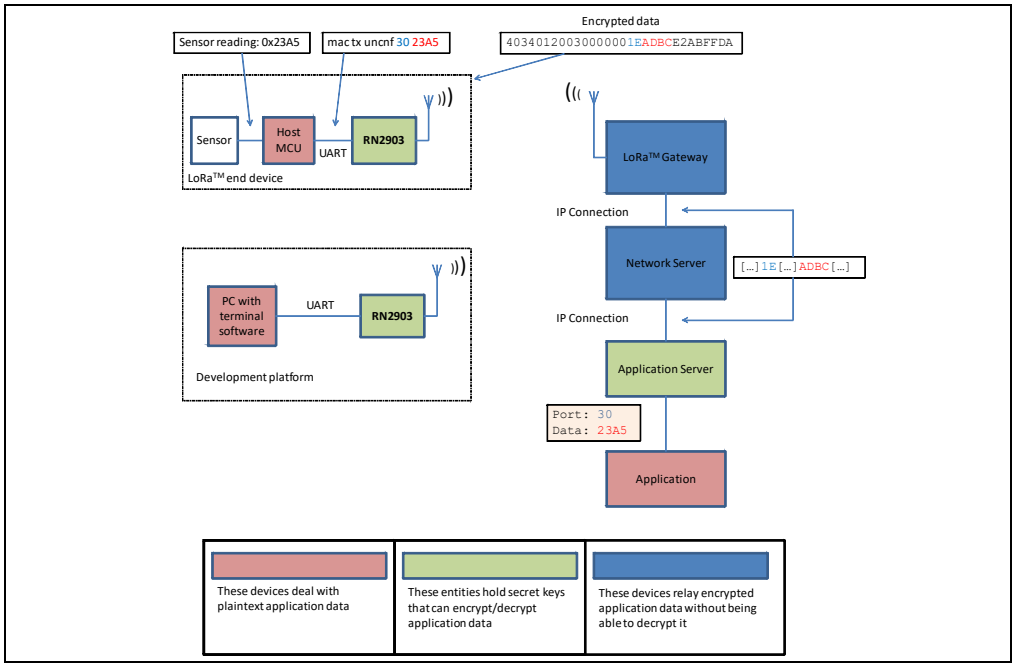
1.1 OVERVIEW

The Microchip RN2903 SA (Ordering Part number: RN2903-I/RMSAxxx) module provides LoRaWAN® protocol connectivity using a simple UART interface. This module handles the LoRaWAN Class A protocol and provides an optimized text command/response interface to the host system. This document is intended to describe an implementation of the LoRaWAN protocols. LoRaWAN Class A protocol terms are described in more detail in the *LoRaWAN® Specification V1.0.1* available from the LoRa Alliance (www.lora-alliance.org). Thus, it is recommended to review the *LoRaWAN® Specification V1.0.1* before using the RN2903 SA module.

The required configuration for accessing a LoRa technology network is minimal and can be stored in the module's EEPROM, allowing for factory configuration of these parameters, lowering the requirements for the host system while also increasing system security. The module also features GPIO pins that can be configured through the UART interface.

A simple use case is described in Figure 1-1 where an end device, containing a host MCU which reads a sensor, commands the RN2903 SA to transmit the sensor reading over the LoRa network. Data are encrypted by the RN2903 SA and the radio packet is received by one or multiple gateways which forward it to the network server. The network server sends the data to the application server which has the key to decrypt the application data. Similarly, a development platform may consist of an RN2903 SA directly connected over UART to a PC, which becomes the host system in this case. Users can then type commands into the module using a terminal program.

FIGURE 1-1: SIMPLE LORAWAN® NETWORK DIAGRAM



The flow of data can be followed as it gets generated by an end device and transported on the network.

1.2 FEATURES

- LoRaWAN Class A protocol compliance
- Integrated FSK, GFSK and LoRa technology transceiver allowing the user to transmit custom packets using these protocols
- Globally unique 64-bit identifier (EUI-64™)
- Configurable GPIOs
- Intelligent Low-Power mode with programmable/on-demand wake-up
- Bootloader for firmware upgrade
- All configuration and control done over UART using simple ASCII commands

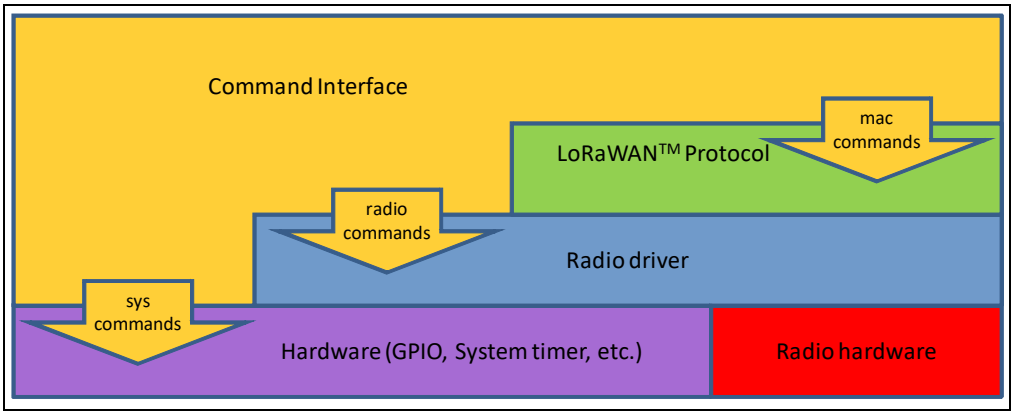
Refer to the *RN2903 Low-Power Long-Range LoRa™ Technology Transceiver Module Data Sheet* (DS50002390) for details on the hardware specifications of the module.

1.3 CONFIGURATION

The RN2903 SA module's architecture is described in Figure 1-2 from the command interface point of view. There are three types of commands that can be used, and each allows access to different module functions:

- LoRaWAN Class A configuration and control, using the `mac` group of commands
- Low level radio configuration and control, using the `radio` group of commands
- Other module functions, using the `sys` group of commands

FIGURE 1-2: RN2903 SA COMMAND INTERFACE (YELLOW) AND ITS RELATIONSHIP TO THE MODULE'S INTERNAL COMPONENTS



The available commands can be used to configure and control the LoRaWAN protocol layer, the radio driver and some system peripherals.

In order to communicate with a LoRa network, a specific number of parameters need to be configured. Since two distinctive methods are offered for a device to become part of the network, each of these requires different parameters:

- Over-the-Air Activation (OTAA), where a device negotiates network encryption keys at the time it joins the network. For this, the device EUI, application EUI and application key need to be configured and then the OTAA procedure can start.
- Activation by Personalization (ABP) where the device already contains the network keys and can directly start communication with the network. Configuring the device address, network session key and application session key is sufficient for this type of initialization.

For increased security, these parameters can be configured and stored in the module's EEPROM during manufacturing of devices requiring LoRaWAN connectivity. Thus, the keys do not need to be sent over the UART interface by the host system every time the device powers up.

1.4 UART INTERFACE

All of the RN2903 SA module's settings and commands are transmitted over UART using the ASCII interface.

All commands need to be terminated with <CR><LF> and any replies they generate will also be terminated by the same sequence.

The default settings for the UART interface are 57600 bps, 8 bits, no parity, 1 Stop bit, no flow control. The baud rate can be changed by triggering the auto-baud detection sequence of the module. To do this, the host system needs to transmit to the module a Break condition followed by a 0x55 character at the new baud rate. The auto-baud detection mechanism can also be triggered during sleep to wake up the module before the predetermined time has expired.

Note: A Break condition is signaled to the module by keeping the UART_RX pin low for longer than the time to transmit a complete character. For example, at the default baud rate of 57600 bps, keeping the UART_RX pin low for 226 μs is a valid 13-bit break condition, whereas at 9600 bps, this would be interpreted as a 0x00 character. The length of the Break condition should be long enough to be identified as a break condition for the UART's lowest configured baud rate.

Break condition calculation is as follows:

$T_b = N_b/bps$

Where:

*T_b = time required for Break condition

*N_b = number of bits required for Break condition

*bps = current bit rate

$T_b = 13/57600$

$T_b = 226\ \mu s$

Chapter 2. Command Reference

The RN2903 SA LoRa technology module supports a variety of commands for configuration. This section describes these commands in detail and provides examples.

2.1 COMMAND SYNTAX

To issue commands to the RN2903 SA module, the user sends keywords followed by optional parameters. Commands (keywords) are case sensitive, and spaces must not be used in parameters. Hex input data can be uppercase or lowercase. String text data, such as `OTAA` used for the join procedure, can be uppercase or lowercase.

The use of shorthand for parameters is *NOT* supported.

Depending on the command, the parameter may expect values in either decimal or hexadecimal form; refer to the command description for the expected form. For example, when configuring the frequency, the command expects a decimal value in Hertz such as `923300000` (923.3 MHz). Alternatively, when configuring the LoRaWAN device address, the hex value is entered into the parameter as `aabbccdd`. To enter a number in hex form, use the value directly. For example, the hex value 0xFF would be entered as FF. The hex values 'a' through 'f' can be uppercase or lowercase.

2.2 COMMAND ORGANIZATION

There are three general command categories, as shown in Table 2-1.

TABLE 2-1: COMMAND TYPES

Command Type	Keyword	Description
System	`<sys>`	Issues system level behavior actions, gathers status information on the firmware and hardware version, or accesses the module user EEPROM memory.
LoRaWAN Class A Protocol	`<mac>`	Issues LoRaWAN Class A protocol network communication behaviors, actions and configurations commands.
Transceiver commands	`<radio>`	Issues radio specific configurations, directly accessing and updating the transceiver setup.

After configuring the LoRaWAN protocol settings, the user must save them to EEPROM with the `<mac save>` command. Once the settings have been saved, they will be retained after a reboot or Reset.

Note: Upon successful reception of commands, the module will respond with one of the following:

- `ok`
- `invalid_param`
- Requested Information
- Descriptive Error Message

Note: To facilitate the sharing of the radio between user custom applications and the LoRaWAN MAC, refer to the `mac pause` and `mac resume` commands. Since no sharing exists between `sys` and other types of commands, there is no need for additional `pause` commands.

2.3 SYSTEM COMMANDS

System commands begin with the system keyword `<sys>` and include the categories shown in Table 2-2, Table 2-3 and Table 2-4.

TABLE 2-2: SYSTEM COMMANDS

Parameter	Description
`sleep`	Puts the system in sleep for a finite number of milliseconds.
`reset`	Resets and restarts the RN2903 SA module.
`eraseFW`	Deletes the current RN2903 SA module application firmware and prepares it for firmware upgrade. The RN2903 SA module bootloader is ready to receive new firmware.
`factoryRESET`	Resets the RN2903 SA module's configuration data and user EEPROM to factory default values and restarts the RN2903 SA module.
`set`(1)	Sets specified system parameter values.
`get`(1)	Gets specified system parameter values.

Note 1: Refer to Table 2-3 for system `<set>` and Table 2-4 for system `<get>` command summaries.

2.3.1 sys sleep <length>

<length>: decimal number representing the number of milliseconds the system is put to sleep, from 100 to 4294967295.

Response: ok after the system gets back from Sleep mode

invalid_param if the length is not valid

This command puts the system to sleep for the specified number of milliseconds. The module can be forced to exit from sleep by sending the UART a Break condition followed by a 0x55 character. Forcing the module from sleep in the manner also triggers the UART auto baud detection. The module will adjust the UART baud rate to match the baud rate at which the 0x55 character is sent. Refer to the Note box in 1.4 "UART Interface".

Example: **sys sleep 120** // Puts the system to sleep for 120 ms.

2.3.2 sys reset

Response: RN2903 SAX.Y.Z MMM DD YYYY HH:MM:SS, where X.Y.Z is the firmware version, MMM is month, DD is day, HH:MM:SS is hour, minutes, seconds (format: [HW] [FW] [Date] [Time]). [Date] and [Time] refer to the release of the firmware.

This command resets and restarts the RN2903 SA module; stored LoRaWAN protocol settings will be loaded automatically upon reboot.

Example: **sys reset** // Resets and restarts the RN2903 SA module.

2.3.3 sys eraseFW

Response: no response

This command deletes the current RN2903 SA module application firmware and prepares it for firmware upgrade. The RN2903 SA module bootloader is ready to receive new firmware.

Example: **sys eraseFW** // Deletes the current RN2903 SA module application firmware.

2.3.4 sys factoryRESET

Response: RN2903 SAX.Y.Z MMM DD YYYY HH:MM:SS, where X.Y.Z is the firmware version, MMM is month, DD is day, HH:MM:SS is hour, minutes, seconds (format: [HW] [FW] [Date] [Time]). [Date] and [Time] refer to the release of the firmware.

This command resets the module's configuration data and user EEPROM to factory default values and restarts the module. After factoryRESET, the RN2903 SA module will automatically reset and all configuration parameters are restored to factory default values. All LoRaWAN protocol settings set by the user will be lost.

Example: **sys factoryRESET** // Restores factory default values.

2.3.5 System Set Commands

TABLE 2-3: SYSTEM SET COMMANDS

Parameter	Description
nvm	Stores <data> to a location <address> of user EEPROM.
pindig	Allows the user to set and clear available digital pins.
pinmode	Allows the user to set the state of the pins as digital output, digital input or analog.

2.3.5.1 sys set nvm <address> <data>

<address>: hexadecimal number representing user EEPROM address, from 300 to 3FF

<data>: hexadecimal number representing data, from 00 to FF

Response: ok if the parameters (address and data) are valid

invalid_param if the parameters (address and data) are not valid

This command allows the user to modify the user EEPROM at <address> with the value supplied by <data>. Both <address> and <data> must be entered as hex values. The user EEPROM memory is located inside the MCU on the module.

Example: **sys set nvm 300 A5** // Stores the value 0xA5 at user EEPROM address 0x300.

2.3.5.2 sys set pindig <pinname> <pinstate>

<pinname>: string representing the pin. Parameter values can be: GPIO0-GPIO13, UART_CTS, UART_RTS, TEST0, TEST1

<pinstate>: decimal number representing the state. Parameter values can be 0 or 1.

Response: ok if the parameters (<pinname>, <pinstate>) are valid

invalid_param if the parameters (<pinname>, <pinstate>) are not valid

This command allows the user to modify the unused pins available for use by the module. The selected <pinname> is driven high or low depending on the desired <pinstate>.

Default: GPIO0-GPIO13, UART_CTS, UART_RTS, TEST0 and TEST1 are driven low (value 0).

Example: **sys set pindig GPIO5 1** // Drives GPIO5 high 1, VDD.

2.3.5.3 sys set pinmode <pinname> <pinmode>

<pinname>: string representing the pin. Parameter can be: GPIO0-GPIO13, UART_CTS, UART_RTS, TEST0, TEST1

<pinmode>: string representing the functional mode of the pin. Parameters can be: digout, digin, ana

Response: ok if all the parameters are valid

invalid_param if any of the parameters are not valid.

This command allows the user to configure the functional mode of a pin. A pin can be configured as:

- A digital output by using the <digout> parameter
- A digital input by using the <digin> parameter
- An analog input by using the <ana> parameter

Default: GPIO0-GPIO14, UART_CTS, UART_RTS, TEST0 and TEST1 are output pins, driven low (value 0).

Note: Only the GPIO0-3, GPIO5-GPIO13 pins can be configured as analog pins.

Example: **sys set pinmode GPIO5 ana** //Sets pin GPIO5 as analog pin

Note: This command must be called prior to reading or setting the value of a pin to have the correct behavior.

2.3.6 System Get Commands

TABLE 2-4: SYSTEM GET COMMANDS

Parameter	Description
ver	Returns the information on hardware platform, firmware version, release date.
nvm	Returns data from the requested user EEPROM <address>.
vdd	Returns measured voltage in mV.
hweui	Returns the preprogrammed EUI node address.
pindig	Returns the state of the pin, either low ('0') or high ('1').
pinana	Returns the state of an analog input.

2.3.6.1 sys get ver

Response: RN2903 SAX.Y.Z MMM DD YYYY HH:MM:SS, where X.Y.Z is the firmware version, MMM is month, DD is day, YYYY is year, HH:MM:SS is hour, minutes, seconds (format: [HW] [FW] [Date] [Time]). [Date] and [Time] refer to the release of the firmware.

This command returns the information related to the hardware platform, firmware version, release date and time-stamp on firmware creation.

Example: **sys get ver** // Returns version-related information.

2.3.6.2 sys get nvm <address>

<address>: hexadecimal number representing user EEPROM address, from 300 to 3FF

Response: 00–FF (hexadecimal value from 00 to FF) if the address is valid

invalid_param if the address is not valid

This command returns the data stored in the user EEPROM of the RN2903 SA module at the requested <address> location.

Example: **sys get nvm 300** // Returns the 8-bit hex value stored at 300.

2.3.6.3 sys get vdd

Response: 0–3600 (decimal value from 0 to 3600)

This command requires the RN2903 SA module to do an ADC conversion on the VDD. The measurement is converted and returned as a voltage (mV).

Example: **sys get vdd** // Returns mV measured on the VDD module.

> **Note:** The upper limit is given for consideration only, considering the module's maximum supply voltage. Should the module's maximum supply voltage be exceeded, the response to this command will reflect the true supply voltage (i.e., will be higher than 3600).

2.3.6.4 sys get hweui

Response: hexadecimal number representing the preprogrammed EUI node address

This command reads the preprogrammed EUI node address from the RN2903 SA module. The value returned by this command is a globally unique number provided by Microchip.

Example: **sys get hweui** // Reads the preprogrammed EUI node address.

> **Note:** The preprogrammed EUI node address is a read-only value and cannot be changed or erased. This value can be used to configure the device EUI using the `mac set deveui` command (see Section 2.4.8.9).

2.3.6.5 sys get pindig <pinname>

<pinname>: string representing the pin. Parameter values can be: GPIO0-GPIO13, UART_CTS, UART_RTS, TEST0, TEST1

Response: a bit representing the state of the pin, either '0' (low) or '1' (high), if <pinname> is valid
invalid_param if <pinname> is not valid

This command returns the state of the queried pin, either '0' (low) or '1' (high).

Example: **sys get pindig GPIO5** // Returns the state of GPIO5.

2.3.6.6 sys get pinana <pinname>

<pinname>: string representing the pin. Parameter values can be: GPIO0-GPIO3, GPIO5-GPIO13

Response: decimal number representing the 10-bit analog value, from 0 to 1023, if <pinname> is valid, and invalid_param if <pinname> is not valid

This command returns a 10-bit analog value for the queried pin, where 0 represents 0V and 1023 represents VDD. An ADC conversion on the VDD pin can be performed by using the command `sys get vdd`.

Example: **sys get pinana GPIO0** // Returns the state of GPIO0.

> **Note:** The `sys set <pinname> ana` command must be called to configure the functional mode of the pin prior to reading its analog input value.

2.4 MAC COMMANDS

LoRaWAN protocol commands begin with the system keyword `mac` and include the categories shown in Table 2-5.

TABLE 2-5: MAC COMMANDS

Parameter	Description
`reset`	Resets the RN2903 SA module and sets default values for most of the LoRaWAN parameters.
`tx`	Sends the data string on a specified port number.
`join`	Informs the RN2903 SA module to join the configured network.
`save`	Saves LoRaWAN configuration parameters to the user EEPROM.
`forceENABLE`	Enables the RN2903 SA module after the LoRaWAN network server commanded the end device to become silent immediately.
`pause`	Pauses LoRaWAN stack functionality to allow transceiver (radio) configuration.
`resume`	Restores the LoRaWAN stack functionality.
`set`	Accesses and modifies specific MAC-related parameters.
`get`	Reads back current MAC-related parameters from the module.

2.4.1 `mac reset`

Response: `ok`

This command will automatically reset the software LoRaWAN stack and initialize it with the default parameters.

Example: **`mac reset`**

> **Note:** This command will set default values for most of the LoRaWAN parameters. Everything set prior to this command will lose its set value, being reinitialized to the default value, including setting the cryptographic keys to 0.

2.4.2 `mac tx <type> <portno> <data>`

`<type>`: string representing the uplink payload type, either `cnf` or `uncnf` (`cnf` – confirmed, `uncnf` – unconfirmed)

`<portno>`: decimal number representing the port number, from 1 to 223

`<data>`: hexadecimal value. The length of `<data>` bytes capable of being transmitted are dependent upon the set data rate (for further details, refer to the *LoRaWAN® Specification V1.0.1*).

Response: this command may reply with two responses. The first response will be received immediately after entering the command. In case the command is valid (`ok` reply received), a second response will be received after the end of the data transfer. For further details, refer to the *LoRaWAN® Specification V1.0.1*.

First response after entering the command:

- `ok` – if parameters and configurations are valid and the packet was forwarded to the radio transceiver for transmission
- `invalid_param` – if parameters (`<type> <portno> <data>`) are not valid
- `not_joined` – if the network is not joined
- `no_free_ch` – no channels are available
- `silent` – if the module is in a Silent Immediately state

- `fram_counter_err_rejoin_needed` – if the frame counter rolled over
- `busy` – if MAC state is not in an Idle state
- `mac_paused` – if MAC was paused and not resumed back
- `invalid_data_len` if application payload length is greater than the maximum application payload length corresponding to the current data rate

Second response after the first uplink transmission attempt:

- `mac_tx_ok` if uplink transmission was successful and no downlink data was received back from the server;
- `mac_rx <portno> <data>` if transmission was successful, `<portno>`: port number, from 1 to 223; `<data>`: hexadecimal value that was received from the server;
- `mac_err` if transmission was unsuccessful, ACK not received back from the server
- `invalid_data_len` if application payload length is greater than the maximum application payload length corresponding to the current data rate. This can occur after an earlier uplink attempt if retransmission back-off has reduced the data rate.

A confirmed message will expect an acknowledgment from the server; otherwise, the message will be retransmitted by the number indicated by the command `mac set retx <value>`, whereas an unconfirmed message will not expect any acknowledgment back from the server. Please refer to the *LoRaWAN® Specification V1.0.1* for further details.

The port number allows multiplexing multiple data streams on the same link. For example, the end device may send measurements on one port number and configuration data on another. The server application can then distinguish the two types of data based on the port number.

Example: **`mac tx cnf 4 5A5B5B`** // Sends a confirmed frame on port 4 with application payload 5A5B5B.

If the automatic reply feature is enabled and the server sets the Frame Pending bit or initiates downlink confirmed transmissions, multiple responses will be displayed after each downlink packet is received by the module. A typical scenario for this case would be (prerequisites: free LoRaWAN channels available and automatic reply enabled):

- The module sends a packet on port 4 with application payload 0xAB
- Radio transmission is successful and the module will display the first response: `ok`
- The server needs to send two separate downlink confirmed packets back on port 1 with the following data: 0xAC, then 0xAF. First it will transmit the first one (0xAC) and will set the Frame Pending bit. The module will display the second response `mac_rx 1 AC`
- The module will initiate an automatic uplink unconfirmed transmission with no application payload because the Frame Pending bit was set in the downlink transmission
- The server will send back the second confirmed packet (0xAF). The module will display a third response `mac_rx 1 AF`
- The module will initiate an automatic unconfirmed transmission with no application payload because the last downlink transmission was confirmed, so the server needs an ACK
- If no reply is received back from the server, the module will display the fourth response after the end of the second Receive window: `mac_tx_ok`
- After this scenario, the user is allowed to send packets when at least one enabled channel is free

Based on this scenario, the following responses will be displayed by the module after running the `mac tx cnf 4 AB` command:

- `ok`
- `mac_rx 1 AC`
- `mac_rx 1 AF`
- `mac_tx_ok`

2.4.3 mac join <mode>

`<mode>`: string representing the join procedure type (case-insensitive), either `otaa` or `abp` (`otaa` – over-the-air activation, `abp` – activation by personalization).

Response: this command may reply with two responses. The first response will be received immediately after entering the command. In case the command is valid (`ok` reply received) a second response will be received after the end of the join procedure. For further details, refer to the *LoRaWAN® Specification V1.0.1*.

First response after entering the command:

- `ok` – if parameters and configurations are valid and the join request packet was forwarded to the radio transceiver for transmission
- `invalid_param` – if `<mode>` is not valid
- `keys_not_init` – if the keys corresponding to the Join mode (`otaa` or `abp`) were not configured
- `no_free_ch` – no channels are available
- `silent` – if the device is in a Silent Immediately state
- `busy` – if MAC state is not in an Idle state
- `mac_paused` – if MAC was paused and not resumed back

Second response after the join procedure:

- `denied` if the join procedure was unsuccessful (the module attempted to join the network, but was rejected);
- `accepted` if the join procedure was successful;

This command informs the RN2903 SA module it should attempt to join the configured network. Module activation type is selected with `<mode>`. Parameter values can be `otaa` (over-the-air activation) or `abp` (activation by personalization). The `<mode>` parameter is not case sensitive. Before joining the network, the specific parameters for each activation type should be configured (for over the air activation: device EUI, application EUI, application key; for activation by personalization: device address, network session key, application session key).

Example: **`mac join otaa`** // Attempts to join the network using over-the-air activation.

2.4.4 mac save

Response: ok

The mac save command must be issued after configuration parameters have been appropriately entered from the mac set <cmd> commands. This command will save LoRaWAN protocol configuration parameters to EEPROM. Upon the next system reset the LoRaWAN protocol configuration will be initialized with the last saved parameters. The system may reset by power cycling or a pulse on the MCLR pin as well as by using the sys reset command.

The LoRaWAN protocol configuration savable parameters are:

- adr: ADR state
- appeui: Application Identifier
- appkey: Application Key
- appskey: Application Session key
- ch: All Channel Parameters
  - drrange: Data Rate Range
  - status: Status
- devaddr: End-Device Address
- deveui: End-Device Identifier
- dnctr: Downlink Frame Counter
- dr:Data Rate
- join flags: Join parameter flags
  - appeui: Application Identifier set
  - appmultiskey: Application Multicast Session Key set
  - appskey: Application Key set
  - devaddr: End-Device Address set
  - deveui: End-Device Identifer set
  - devmultiskey: End-Device Multicast Session Key set
  - nwkmultiskey: Network Multicast Session Key set
  - nwkskey: Network Session Key set
- nwkskey: Network Session Key
- rx2 parameters: RX Window 2 parameters
  - freq: Frequency
  - dr: Data Rate
- upctr: Uplink Frame Counter

Example: **mac save**	// Saves the LoRaWAN protocol configuration parameters to the user EEPROM.

2.4.5 mac forceENABLE

Response: ok

The network can issue a certain command (Duty Cycle Request frame with parameter 255) that would require the RN2903 SA module to go silent immediately. This mechanism disables any further communication of the module, effectively isolating it from the network. Using mac forceENABLE, after this network command has been received, restores the module's connectivity by allowing it to send data.

Example: **mac forceENABLE** // Disables the Silent Immediately state.

Note: The silent immediately status bit of the MAC status register indicates the device has been silenced by the network. Refer to **Figure 2-1: "MAC Status bit-Mapped Register (1)"**

2.4.6 mac pause

Response: 0 – 4294967245 (decimal number representing the number of milliseconds the mac can be paused)

This command pauses the LoRaWAN stack functionality to allow transceiver (radio) configuration. Through the use of mac pause, radio commands can be generated between a LoRaWAN Class A protocol uplink application (mac tx command), and the LoRaWAN Class A protocol Receive windows (second response for the mac tx command). This command will reply with the time interval in milliseconds that the transceiver can be used without affecting the LoRaWAN functionality. The maximum value (4294967245) is returned whenever the LoRaWAN stack functionality is in Idle state and the transceiver can be used without restrictions. '0' is returned when the LoRaWAN stack functionality cannot be paused.

After the radio configuration is complete, the mac resume command must be used to return to LoRaWAN Class A protocol commands.

Example: **mac pause** // Pauses the LoRaWAN stack functionality if the response is different from 0.

Note: If already joined to a network, this command *MUST* be called *BEFORE* configuring the radio parameters, initiating radio reception, or transmission.

2.4.7 mac resume

Response: ok

This command resumes LoRaWAN stack functionality, in order to continue normal functionality after being paused.

Example: **mac resume** // Resumes the LoRaWAN stack functionality.

Note: This command *MUST* be called *AFTER* all radio commands have been issued and all the corresponding asynchronous messages have been replied.

2.4.8 MAC Set Commands

TABLE 2-6: MAC SET COMMANDS

Parameter	Description
adr	Sets the adaptive data rate to enabled or disabled.
appeui	Sets the application identifier for the RN2903 SA module.
appkey	Sets the application key for the RN2903 SA module.
appskey	Sets the application session key for the RN2903 SA module.
ar	Sets the state of the automatic reply.
bat	Sets the battery level needed for Device Status Answer frame command response.
ch	Allows modification of channel-related parameters.
devaddr	Sets the unique network device address for the RN2903 SA module.
deveui	Sets the globally unique identifier for the RN2903 SA module.
dnctr	Sets the value of the downlink frame counter that will be used for the next downlink reception.
dr	Sets the data rate to be used for the next transmissions.
linkchk	Sets the time interval for the link check process to be triggered.
nwkskey	Sets the network session key for the RN2903 SA module.
pwridx	Sets the output power to be used on the next transmissions.
retx	Sets the number of retransmissions to be used for an uplink confirmed packet.
rx2	Sets the data rate and frequency used for the second Receive window.
rxdelay1	Sets the value used for the first Receive window delay.
sync	Sets the synchronization word for the LoRaWAN communication.
upctr	Sets the value of the uplink frame counter that will be used for the next uplink transmission.

2.4.8.1 mac set adr <state>

<state>: string value representing the state, either on or off.

Response: ok if state is valid

invalid_param if state is not valid

This command enables/disables the adaptive data rate (ADR). The server is informed about the status of the module's ADR in every uplink frame it receives from the ADR field in uplink data packet. If ADR is enabled, the server optimizes the data rate and the transmission power of the module based on the information collected from the network.

Example: **mac set adr on** //This will enable the ADR mechanism.

2.4.8.2 mac set appeui <appEUI>

<appEUI>: 8-byte hexadecimal number representing the application EUI.

Response: ok if address is valid

invalid_param if address is not valid

This command sets the application identifier for the module. The application identifier should be used to identify device types (sensor device, lighting device, and so on) within the network.

Example: **mac set appEUI FEDCBA9876543210**

Note: If this parameter was previously saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.3 `mac set appkey <appKey>`

`<appKey>`: 16-byte hexadecimal number representing the application key

Response: `ok` if key is valid

`invalid_param` if key is not valid

This command sets the application key for the module. The application key is used to derive the security credentials for communication during over-the-air activation.

Example: **`mac set appkey 00112233445566778899AABBCCDDEEFF`**

Note: If this parameter was previously saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

`2.4.8.4 mac set appskey <appSesskey>`

`<appSessKey>`: 16-byte hexadecimal number representing the application session key

Response: `ok` if key is valid

`invalid_param` if key is not valid

This command sets the application session key for the module. This key provides security for communication between module and application server.

Example: **`mac set appskey AFBECD56473829100192837465FAEBDC`**

Note: If this parameter was previously saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.5 `mac set ar <state>`

`<state>`: string value representing the state, either `on` or `off`.

Response: `ok` if state is valid

`invalid_param` if state is not valid

This command sets the state of the automatic reply. By enabling the automatic reply, the module will transmit a packet without a payload immediately after a confirmed downlink is received, or when the Frame Pending bit has been set by the server. If set to OFF, no automatic reply will be transmitted.

Example: **`mac set ar on`** // Enables the automatic reply process inside the module.

Note: The RN2903 SA module implementation will initiate automatic transmissions with no application payload if the automatic reply feature is enabled and the server sets the Frame Pending bit or initiates a confirmed downlink transmission. The user will not be able to initiate uplink transmissions until the automatic transmissions are done.

2.4.8.6 mac set bat <level>

<level>: decimal number representing the level of the battery, from 0 to 255. '0' means external power, '1' means low level, 254 means high level, 255 means the end device was not able to measure the battery level.

Response: ok if the battery level is valid

invalid_param if the battery level is not valid

This command sets the battery level required for Device Status Answer frame in use with the LoRaWAN Class A protocol.

Example: **mac set bat 127** // Battery is set to ~50%.

2.4.8.7 MAC SET CHANNEL COMMANDS

TABLE 2-7: MAC SET CHANNEL COMMANDS

Parameter	Description
drrange	Sets the module allowed data rate range (min.- max.) allowed on a given channel ID.
status	Sets the use of the specified channel ID.

2.4.8.7.1 mac set ch drrange <channelID> <minRange> <maxRange>

<channelID>: decimal number representing the channel number, from 0 to 71

<minRange>: decimal number representing the minimum data rate, from 0 to 3

<maxRange>: decimal number representing the maximum data rate, from 0 to 3

Response: ok if parameters are valid

invalid_param if parameters are not valid

This command sets the operating data rate range, min. to max., for the given <channelID>. By doing this the module can vary data rates between the <minRange> and <maxRange> on the specified <channelID>. For the actual values of the data rates and the corresponding spreading factors (SF), refer to the *LoRaWAN® Specification V1.0.1*.

Example: **mac set ch drrange 13 0 2** // Using AU 915-928 band: on channel 13 the data rate can range from 0 (SF10/125 kHz) to 2 (SF8/125 kHz) as required.

Note: In the AU 915-928 band, channels 64 to 71 are restricted to the DR4 data rate. This command is valid for those channels, however, the only allowed <minRange> and <maxRange> values are 4 4.

Note: If this parameter was previously saved to user EEPROM by issuing the mac save command, after modifying its value, the mac save command should be called again.

2.4.8.7.2 mac set ch status <channelID> <status>

<channelID>: decimal number representing the channel number, from 0 to 71.

<status>: string value representing the state, either on or off.

Response: ok if parameters are valid

invalid_param if parameters are not valid

This command sets the operation of the given <channelID>.

Example: **mac set ch status 4 off** // Channel ID 4 is disabled from use.

> **Note:** If this parameter was previously saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.8 `mac set devaddr <address>`

`<address>`: 4-byte hexadecimal number representing the device address, from 00000000 – FFFFFFFF

Response: `ok` if address is valid

`invalid_param` if address is not valid

This command configures the module with a 4-byte unique network device address `<address>`. The `<address>` *MUST* be *UNIQUE* to the current network. This must be directly set solely for activation by personalization devices. This parameter must not be set before attempting to join using over-the-air activation because it will be overwritten once the join process is over.

Example: **mac set devaddr ABCDEF01**

> **Note:** If this parameter had previously been saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.9 `mac set deveui <devEUI>`

`<devEUI>`: 8-byte hexadecimal number representing the device EUI

Response: `ok` if address is valid

`invalid_param` if address is not valid

This command sets the globally unique device identifier for the module. The identifier must be set by the host MCU. The module contains a pre-programmed unique EUI and can be retrieved using the `sys get hweui` command (see Section 2.3.6.4) or user provided EUI can be configured using the `mac set deveui` command.

Example: **mac set deveui 0004A30B001A55ED**

> **Note:** If this parameter was previously saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.10 `mac set dnctr <fCntDown>`

`<fCntDown>`: decimal number representing the value of the downlink frame counter that will be used for the next downlink reception, from 0 to 4294967295.

Response: `ok` if parameter is valid

`invalid_param` if parameter is not valid

This command sets the value of the downlink frame counter that will be used for the next downlink reception.

Example: **mac set dnctr 30**

> **Note:** If this parameter had previously been saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.11 `mac set dr <dataRate>`

`<dataRate>`: decimal number representing the data rate, from 0 and 4, but within the limits of the data rate range for the defined channels.

Response: `ok` if data rate is valid

`invalid_param` if data rate is not valid

This command sets the data rate to be used for the next transmission. For the description of data rates and the corresponding spreading factors, refer to the *LoRaWAN® Specification V1.0.1*.

Example: **`mac set dr 4`** // On AU 915-928; SF8/500 kHz.

> **Note:** If this parameter had previously been saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.12 `mac set linkchk <linkCheck>`

`<linkCheck>`: decimal number that sets the time interval in seconds for the link check process, from 0 to 65535

Response: `ok` if the time interval is valid

`invalid_param` if the time interval is not valid

This command sets the time interval for the link check process to be triggered periodically. A `<value>` of '0' will disable the link check process. When the time interval expires, the next application packet that will be sent to the server will include also a link check MAC command. For more information on the Link Check MAC command, refer to the *LoRaWAN® Specification V1.0.1*.

Example: **`mac set linkchk 600`** // The module will attempt a link check process at 600-second intervals.

> **Note:** If the command `mac reset` is issued, the link check process will be set as disabled.

2.4.8.13 `mac set nwkskey <nwkSessKey>`

`<nwkSessKey>`: 16-byte hexadecimal number representing the network session key

Response: `ok` if key is valid

`invalid_param` if key is not valid

This command sets the network session key for the module. This key is 16 bytes in length, and provides security for communication between the module and network server.

Example: **`mac set nwkskey 1029384756AFBECD5647382910DACFEB`**

> **Note:** If this parameter was previously saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.14 `mac set pwridx <pwrIndex>`

`<pwrIndex>`: decimal number representing the index value for the output power. The values 5, and 7 to 10 can be set.

Response: `ok` if power index is valid

`invalid_param` if power index is not valid

This command sets the output power to be used on the next transmissions. Refer to the *LoRaWAN® Specification V1.0.1* for the output power corresponding to the <pwrIndex> and also to the *RN2903 Low-Power Long-Range LoRa® Technology Transceiver Module Data Sheet* (DS50002390) for the actual radio power capabilities.

Example: **mac set pwridx 5** // Sets the TX output power to 18 dBm on the next transmission for a 915 MHz SA module.

2.4.8.15 `mac set retx <reTxNb>`

<reTxNb>: decimal number representing the number of retransmissions for an uplink confirmed packet, from 0 to 255.

Response: `ok` if <retx> is valid

`invalid_param` if <retx> is not valid

This command sets the number of retransmissions to be used for an uplink confirmed packet, if no downlink acknowledgment is received from the server.

Example: **mac set retx 5** // The number of retransmissions made for an uplink confirmed packet is set to 5.

2.4.8.16 `mac set rx2 <dataRate> <frequency>`

<dataRate>: decimal number representing the data rate, from 8 to 13.

<frequency>: decimal number representing the frequency, from 923300000 to 927500000 Hz in 600kHz steps.

Response: `ok` if parameters are valid

`invalid_param` if parameters are not valid

This command sets the data rate and frequency used for the second Receive window. The configuration of the Receive window parameters should be in accordance with the server configuration.

Example: **mac set rx2 8 927500000** // Receive window 2 is configured with SF12/500 kHz data rate with a center frequency of 927.5 MHz.

Note: If this parameter had previously been saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.17 `mac set rxdelay1 <rxDelay>`

<rxDelay>: decimal number representing the delay between the transmission and the first Reception window in milliseconds, from 0 to 65535.

Response: `ok` if <rxDelay> is valid

`invalid_param` if <rxDelay> is not valid

This command will set the delay between the transmission and the first Reception window to the <rxDelay> in milliseconds. The delay between the transmission and the second Reception window is calculated in software as the delay between the transmission and the first Reception window + 1000 (ms).

Example: **mac set rxdelay1 1000** // Set the delay between the transmission and the first Receive window to 1000 ms.

2.4.8.18 `mac set sync <synchWord>`

<synchWord>: one byte long hexadecimal number representing the synchronization word for the LoRaWAN communication

Response: `ok` if parameters are valid

`invalid_param` if parameter is not valid

This command sets the synchronization word for the LoRaWAN communication. The configuration of the synchronization word should be in accordance with the Gateway configuration.

Example: **`mac set sync 34`** //Synchronization word is configured to use the 0x34 value

2.4.8.19 `mac set upctr <fCntUp>`

`<fCntUp>`: decimal number representing the value of the uplink frame counter that will be used for the next uplink transmission, from 0 to 4294967295.

Response: `ok` if parameter is valid

`invalid_param` if parameter is not valid

This command sets the value of the uplink frame counter that will be used for the next uplink transmission.

Example: **`mac set upctr 10`**

Note: If this parameter had previously been saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.9 MAC Get Commands

TABLE 2-8: MAC GET COMMANDS

Parameter	Description
adr	Gets the state of adaptive data rate for the device.
appeui	Gets the application identifier for the end device.
ar	Gets the state of the automatic reply.
ch	Gets parameters related information which pertains to channel operation and behaviors.
dcycleps	Gets the duty cycle prescaler which can only be configured by the server.
devaddr	Gets the current stored unique network device address for that specific end device.
deveui	Gets the current stored globally unique identifier for that specific end device.
dnctr	Gets the value of the downlink frame counter that will be used for the next downlink reception.
dr	Gets the data rate to be used for the next transmission.
gwnb	Gets the number of gateways that successfully received the last Link Check Request frame.
mrgn	Gets the demodulation margin as received in the last Link Check Answer frame.
pwridx	Gets the output power index value.
retx	Gets the number of retransmissions to be used for an uplink confirmed packet.
rx2	Gets the data rate and frequency used for the second Receive window.
rxdelay1	Gets the interval value stored for rxdelay1.
rxdelay2	Gets the interval value stored for rxdelay2.
status	Gets the current status of the RN2903 SA module.
sync	Gets the synchronization word for the LoRaWAN communication.
upctr	Gets the value of the uplink frame counter that will be used for the next uplink transmission.

2.4.9.1 mac get adr

Response: string representing the state of the adaptive data rate mechanism, either on or off.

This command will return the state of the adaptive data rate mechanism. It will reflect if the ADR is on or off on the requested device.

Default: off

Example: **mac get adr**

2.4.9.2 mac get appeui

Response: 8-byte hexadecimal number representing the application EUI.

This command will return the application identifier for the module. The application identifier is a value given to the device by the network.

Default: 0000000000000000

Example: **mac get appeui**

2.4.9.3 mac get ar

Response: string representing the state of the automatic reply, either on or off.

This command will return the current state for the automatic reply (AR) parameter. The response will indicate if the AR is on or off.

Default: off

Example: **mac get ar**

2.4.9.4 MAC GET CHANNEL COMMANDS

TABLE 2-9: MAC GET CHANNEL COMMANDS

Parameter	Description
freq	Gets the module operation frequency for the specified channel ID.
drrange	Gets the valid data rate range (min. to max.) allowed for the module on the specified channel ID
status	Gets the status for the specified channel ID to indicate if it is enabled for use.

TABLE 2-10: DEFAULT PARAMETERS FOR CHANNELS

Channel Number	Parameters	AU 915-928 MHz Band
Channels 0-63	Frequency (Hz)	915200000 + (20000 * channel number)
	Data rate range	0-3
	Status	On
Channels 64-71	Frequency (Hz)	915900000 + (1600000 * (channel number - 65))
	Data rate range	4-4
	Status	On

2.4.9.4.1 mac get ch freq <channelID>

<channelID>: decimal number representing the channel number, from 0 to 71.

Response: decimal number representing the frequency of the channel, from 915200000 to 927800000 in Hz.

This command returns the frequency on the requested <channelID>, entered in decimal form.

Default: see Table 2-10

Example: **mac get ch freq 0**

2.4.9.4.2 mac get ch drrange <channelID>

<channelID>: decimal number representing the channel number, from 0 to 71.

Response: decimal number representing the minimum data rate of the channel, from 0 to 4 and a decimal number representing the maximum data rate of the channel, from 0 to 4.

This command returns the allowed data rate index range on the requested <channelID>, entered in decimal form. The <minRate> and <maxRate> index values are returned in decimal form and reflect index values. For the description of data rates and the corresponding spreading factors, refer to the *LoRaWAN® Specification V1.0.1*.

Default: see Table 2-10

Example: **mac get ch drrange 0**

2.4.9.4.3 mac get ch status <channelID>

<channelID>: decimal number representing the channel number, from 0 to 71.

Response: string representing the state of the channel, either on or off.

This command returns if <channelID> is currently enabled for use. <channelID> is entered in decimal form and the response will be on or off reflecting the channel is enabled or disabled appropriately.

Default: see Table 2-10

Example: **mac get ch status 2**

Note: <channelID> parameters must be issued prior to enabling the status of that channel. If a channel is disabled through the <status>, all channel parameters must be reconfigured prior to enabling.

2.4.9.5 mac get dcycleps

Response: decimal number representing the prescaler value, from 0 to 65535.

This command returns the duty cycle prescaler. The value of the prescaler can be configured *ONLY* by the *SERVER* through use of the Duty Cycle Request frame. Upon reception of this command from the server, the duty cycle prescaler is changed for all enabled channels.

Default: 1

Example: **mac get dcycleps**

2.4.9.6 mac get devaddr

Response: 4-byte hexadecimal number representing the device address, from 00000000 to FFFFFFFF.

This command will return the current end-device address of the module.

Default: 00000000

Example: **mac get devaddr**

2.4.9.7 mac get deveui

Response: 8-byte hexadecimal number representing the device EUI.

This command returns the globally unique end-device identifier, as set in the module.

Default: pre-programmed EUI node address

Example: **mac get deveui**

Note: After the mac reset command is explicitly called, the device EUI value will be set to all zeros. Make certain that a valid value is given to the device EUI.

2.4.9.8 mac get dnctr

Response: decimal number representing the value of the downlink frame counter that will be used for the next downlink reception, from 0 to 4294967295.

This command will return the value of the downlink frame counter that will be used for the next downlink reception.

Default: 0

Example: **mac get dnctr**

2.4.9.9 mac get dr

Response: decimal number representing the current data rate.

This command will return the current data rate.

Default: 0

Example: **mac get dr**

2.4.9.10 `mac get gwnb`

Response: decimal number representing the number of gateways, from 0 to 255.

This command will return the number of gateways that successfully received the last Link Check Request frame command, as received in the last Link Check Answer.

Default: 0

Example: **`mac get gwnb`**

2.4.9.11 `mac get mrgn`

Response: decimal number representing the demodulation margin, from 0 to 255.

This command will return the demodulation margin as received in the last Link Check Answer frame. For the description of the values, refer to the *LoRaWAN® Specification V1.0.1*.

Default: 255

Example: **`mac get mrgn`**

2.4.9.12 `mac get pwridx`

Response: decimal number representing the current output power index value, from 5 to 10. This command returns the current output power index value.

Default: 6

Example: **`mac get pwridx`**

2.4.9.13 `mac get retx`

Response: decimal number representing the number of retransmissions, from 0 to 255.

This command will return the currently configured number of retransmissions which are attempted for a confirmed uplink communication when no downlink response has been received.

Default: 7

Example: **`mac get retx`**

2.4.9.14 `mac get rx2`

Response: decimal number representing the data rate configured for the second Receive window, from 8 to13 and a decimal number for the frequency configured for the second Receive window, from 923300000 to 927500000 in Hz.

This command will return the current data rate and frequency configured to be used during the second Receive window.

Default: 8 923300000

Example: **`mac get rx2`**

2.4.9.15 `mac get rxdelay1`

Response: decimal number representing the interval, in milliseconds, for `rxdelay1`, from 0 to 65535.

This command will return the interval, in milliseconds, for `rxdelay1`.

Default: 1000

Example: **`mac get rxdelay1`**

2.4.9.16 mac get rxdelay2

Response: decimal number representing the interval, in milliseconds, for `rxdelay2`, from 0 to 65535.

This command will return the interval, in milliseconds, for `rxdelay2`.

Default: 2000

Example: **mac get rxdelay2**

2.4.9.17 mac get status

Response: 4-byte hexadecimal number representing the current status of the module.

This command will return the current status of the module. The value returned is a bit mask represented in hexadecimal form. For the significance of the bit mask, refer to Figure 2-1.

Default: 00000000

Example: **mac get status**

2.4.9.18 mac get sync

Response: one byte long hexadecimal number representing the synchronization word for the LoRaWAN communication.

This command will return the synchronization word for the LoRaWAN communication.

Default: 34

Example: **mac get sync**

2.4.9.19 mac get upctr

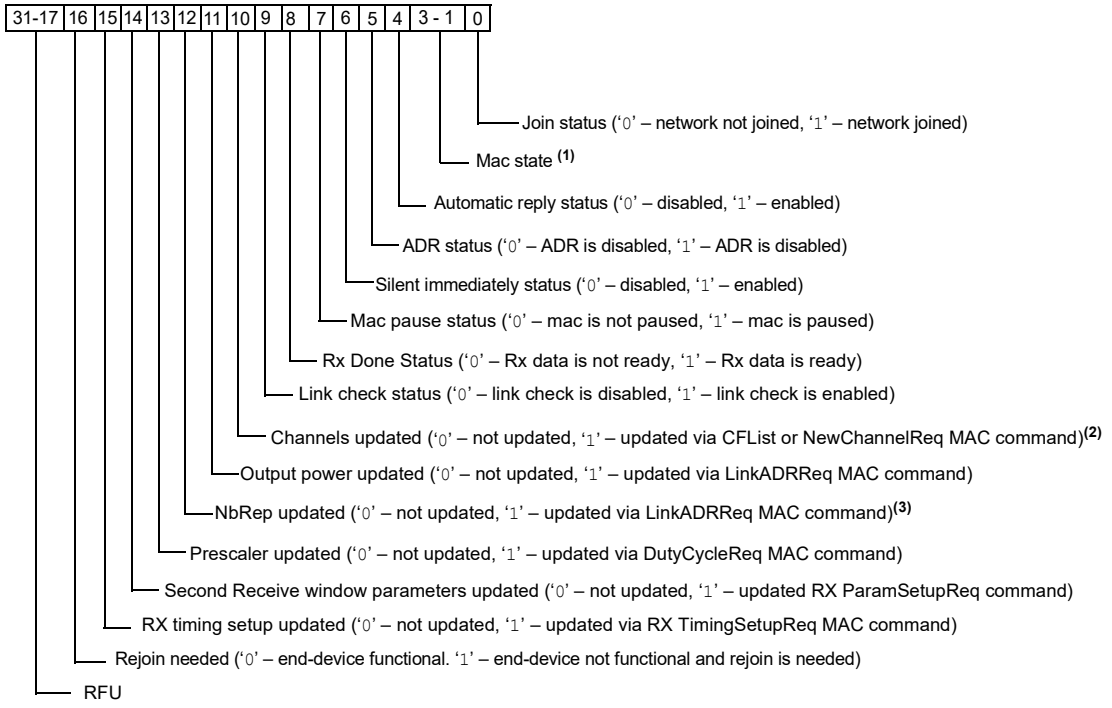
Response: decimal number representing the value of the uplink frame counter that will be used for the next uplink transmission, from 0 to 4294967295.

This command will return the value of the uplink frame counter that will be used for the next uplink transmission.

Default: 0

Example: **mac get upctr**

FIGURE 2-1: MAC STATUS BIT-MAPPED REGISTER (1)



Note 1: Mac state:

0 – Idle (transmissions are possible)

1 – Transmission occurring

2 – Before the opening of Receive window 1

3 – Receive window 1 is open

4 – Between Receive window 1 and Receive window 2

5 – Receive window 2 is open

6 – Retransmission delay - used for ADR_ACK delay, FSK can occur

7 – APB_delay

2: Bits 10 (Channels updated), 11 (Output power updated), 12 (NbRep updated), 13 (Prescaler updated), 14 (Second Receive window parameters updated) and 15 (RX timing setup updated) are cleared after issuing a "mac get status" command.

3: NbRep is the number of repetitions for unconfirmed packets (for more details, refer to the *LoRaWAN® Specification V1.0.1*).

2.5 RADIO COMMANDS

TABLE 2-11: RADIO COMMANDS(1)

Parameter	Description
rx	This command configures the radio to receive simple radio packets according to prior configuration settings.
tx	This command configures a simple radio packet transmission according to prior configuration settings.
cw	This command will put the module into a Continuous Wave (cw) Transmission for system tuning or certification use.
set	This command allows modification to the radio setting directly. This command allows for the user to change the method of radio operation within module type band limits.
get	This command grants the ability to read out radio settings as they are currently configured.

Note 1: The `mac pause` command must be called before any radio transmission or reception, even if no MAC operations have been initiated before.

TABLE 2-12: RADIO PARAMETER AVAILABILITY FOR DIFFERENT OPERATIONS

Command	radio get	radio set	Availability for LoRa Modulation	Availability for FSK Modulation
bt	√	√	—	√
mod	√	√	√	√
freq	√	√	√	√
pwr	√	√	√	√
sf	√	√	√	—
afcbw	√	√	—	√
rxbw	√	√	—	√
bitrate	√	√	—	√
fdev	√	√	—	√
prlen	√	√	—	√
crc	√	√	√	√
iqi	√	√	√	—
cr	√	√	√	—
wdt	√	√	√	√
sync	√	√	√	√
bw	√	√	√	—
snr	√	—	√	—

2.5.1 radio rx <rxWindowSize>

<rxWindowSize>: decimal number representing the number of symbols (for LoRa modulation) or time out in milliseconds (for FSK modulation) that the receiver will be opened, from 0 to 65535. Set <rxWindowSize> to '0' in order to enable the Continuous Reception mode. Continuous Reception mode will be exited once a valid packet is received.

Response: this command may reply with two responses. The first response will be received immediately after entering the command. If the command is valid (ok reply is received), a second response will be received after the reception of a packet or after the time out occurred.

First response after entering the command:

- ok – if parameter is valid and the transceiver is configured in Receive mode
- invalid_param – if parameter is not valid
- busy – if the transceiver is currently busy

Second response after the receive process:

- radio_rx <data> – if reception was successful, <data>: hexadecimal value that was received;
- radio_err – if reception was not successful, reception time-out occurred

Example: **radio rx 0** // Puts the radio into Continuous Receive mode.

Note:	Ensure the radio Watchdog Timer time-out is higher than the Receive window size.

Note:	The mac pause command must be called before any radio transmission or reception, even if no MAC operations have been initiated before.

2.5.2 radio tx <data>

<data>: hexadecimal value representing the data to be transmitted, from 0 to 255 bytes for LoRa modulation and from 0 to 64 bytes for FSK modulation.

Response: this command may reply with two responses. The first response will be received immediately after entering the command. If the command is valid (ok reply received), a second response will be received after the effective transmission.

First response after entering the command:

- ok – if parameter is valid and the transceiver is configured in Transmit mode
- invalid_param – if parameter is not valid
- busy – if the transceiver is currently busy

Second response after the effective transmission:

- radio_tx_ok – if transmission was successful
- radio_err – if transmission was unsuccessful (interrupted by radio Watchdog Timer time-out)

This command transmits the <data> passed.

Example: **radio tx 48656c6C6F** // Transmits a packet of [0x48][0x65][0x6c][0x6C][0x6F]; Hello.

Note: The mac pause command must be called before any radio transmission or reception, even if no MAC operations have been initiated before.

2.5.3 radio cw <state>

<state>: string representing the state of the Continuous Wave (CW) mode, either on or off.

Response: ok if state is on

RN2903 SAX.Y.Z MMM DD YYYY HH:MM:SS, where X.Y.Z is the firmware version, MMM is month, DD is day, YYYY is year, HH:MM:SS is hour, minutes, seconds (format: [HW] [FW] [Date] [Time]). [Date] and [Time] refer to the firmware release; if state is off.

invalid_param if state is not valid

This command will enable or disable the CW mode on the module. CW mode allows the user to put the transceiver into Transmission mode to observe the generated signal. By altering the radio settings the user can observe the changes in transmissions levels.

Example: **radio cw on**

Note: Using radio cw off resets the module, this command being semantically identical to **2.3.2 "sys reset"**.

2.5.4 Radio Set Commands

TABLE 2-13: RADIO SET COMMANDS

Parameter	Description
afcbw	Set the value used by the automatic frequency correction bandwidth.
bitrate	Set the frequency shift keying (FSK) bit rate.
bt	Set the data shaping for frequency shift keying (FSK) modulation type.
bw	Set the value used for the radio bandwidth.
cr	Set the coding rate used by the radio.
crc	Set if a CRC header is to be used.
fdev	Set the frequency deviation allowed by the end device.
freq	Set the current operation frequency for the radio.
iqi	Set if IQ inversion is used.
mod	Set the module Modulation mode.
prlen	Set the preamble length used during transmissions.
pwr	Set the output power level used by the radio during transmission.
rxbw	Set the operational receive bandwidth.
sf	Set the requested spreading factor (SF) to be used during transmission.
sync	Set the sync word used.
wdt	Set the time-out limit for the radio Watchdog Timer.

2.5.4.1 radio set afcbw <autoFreqBand>

<autoFreqBand>: float representing the automatic frequency correction, in kHz. Parameter values can be: 250, 125, 62.5, 31.3, 15.6, 7.8, 3.9, 200, 100, 50, 25, 12.5, 6.3, 3.1, 166.7, 83.3, 41.7, 20.8, 10.4, 5.2, 2.6.

Response: ok if the automatic frequency correction is valid

invalid_param if the automatic frequency correction is not valid

This command modifies the automatic frequency correction bandwidth for receiving/transmitting.

Example: **radio set afcbw 125**

2.5.4.2 radio set bitrate <fskBitRate>

<fskBitRate>: decimal number representing the FSK bit rate value, from 1 to 300000.

Response: ok if the bit rate value is valid

invalid_param if the bit rate value is not valid

This command sets the FSK bit rate value.

Example: **radio set bitrate 5000** // FSK bit rate is set to 5 kb/s.

2.5.4.3 radio set bt <gfBT>

<gfBT>: string representing the Gaussian baseband data shaping, enabling GFSK modulation. Parameter values can be: none, 1.0, 0.5, 0.3.

Response: ok if the data shaping is valid

invalid_param if the data shaping is not valid

This command modifies the data shaping applied to FSK transmissions. Entering any <gfBT> other than none will result in a Gaussian Filter BT being applied to transmissions in FSK mode.

Example: **radio set bt none** // Data shaping in FSK mode is disabled or null.

2.5.4.4 radio set bw <bandWidth>

<bandWidth>: decimal representing the operating radio bandwidth, in kHz. Parameter values can be: 125, 250, 500.

Response: ok if the bandwidth is valid

invalid_param if the bandwidth is not valid

This command sets the operating radio bandwidth for LoRa operation.

Example: **radio set bw 250** // The operating bandwidth is 250 kHz.

2.5.4.5 radio set cr <codingRate>

<codingRate>: string representing the coding rate. Parameter values can be: 4/5, 4/6, 4/7, 4/8.

Response: ok if the coding rate is valid

invalid_param if the coding rate is not valid

This command modifies the coding rate currently being used by the radio.

Example: **radio set cr 4/7** // The coding rate is set to 4/7.

2.5.4.6 radio set crc < crcHeader >

<crcHeader>: string representing the state of the CRC header, either on or off.

Response: ok if the state is valid

invalid_param if the state is not valid

This command enables or disables the CRC header for communications.

Example: **radio set crc on** // Enables the CRC header.

2.5.4.7 radio set fdev <freqDev>

<freqDev>: decimal number representing the frequency deviation, from 0 to 200000.

Response: ok if the frequency deviation is valid

invalid_param if frequency deviation is not valid

This command sets the frequency deviation during operation.

Example: **radio set fdev 5000** // Frequency deviation is 5 kHz.

2.5.4.8 radio set freq <frequency>

<frequency>: decimal representing the frequency, from 902000000 to 928000000 in Hz.

Response: ok if the frequency is valid

invalid_param if the frequency is not valid

This command changes the communication frequency of the radio transceiver.

Example: **radio set freq 923300000**

2.5.4.9 radio set iqi <iqInvert>

<iqInvert>: string representing the state of the invert IQ, either on or off.

Response: ok if the state is valid

invalid_param if the state is not valid

This command enables or disables the Invert IQ for communications.

Example: **radio set iqi on** // Invert IQ is enabled.

2.5.4.10 radio set mod <mode>

<mode>: string representing the modulation method, either lora or fsk.

Response: ok if the modulation is valid

invalid_param if the modulation is not valid

This command changes the modulation method being used by the module. Altering the mode of operation does not affect previously set parameters, variables or registers. FSK mode also allows GFSK transmissions when data shaping is enabled.

Example: **radio set mod lora**

2.5.4.11 radio set prlen <preamble>

<preamble>: decimal number representing the preamble length, from 0 to 65535.

Response: ok if the preamble length is valid

invalid_param if the preamble length is not valid

This command sets the preamble length for transmit/receive.

Example: **radio set prlen 8** // Preamble length is 8.

2.5.4.12 radio set pwr <pwrOut>

<pwrOut>: signed decimal number representing the transceiver output power, from 2 to 20.

Response: ok if the output power is valid

invalid_param if the output power is not valid

This command changes the transceiver output power. It is possible to set the output power above the regulatory limits. This power setting allows some compensation on the cable or transmission line loss. For additional details on output power, please see the *RN2903 Low-Power Long-Range LoRa® Technology Transceiver Module Data Sheet* (DS50002390).

The actual radio power capabilities are from 2 to 17 dBm or 20 dBm.

Example: **radio set pwr 14**

2.5.4.13 radio set rxbw <rxBandwidth>

<rxBandwidth>: float representing the signal bandwidth, in kHz. Parameter values can be: 250, 125, 62.5, 31.3, 15.6, 7.8, 3.9, 200, 100, 50, 25, 12.5, 6.3, 3.1, 166.7, 83.3, 41.7, 20.8, 10.4, 5.2, 2.6.

Response: ok if the signal bandwidth is valid

invalid_param if signal bandwidth is not valid

This command sets the signal bandwidth when receiving.

Example: **radio set rxbw 250** // Signal bandwidth for receiving is 250 kHz.

2.5.4.14 radio set sf <spreadingFactor>

<spreadingFactor>: string representing the spreading factor. Parameter values can be: sf7, sf8, sf9, sf10, sf11 or sf12.

Response: ok if the spreading factor is valid

invalid_param if the spreading factor is not valid

This command sets the spreading factor used during transmission.

Example: **radio set sf sf7**

2.5.4.15 radio set sync <syncWord>

<syncWord>: hexadecimal value representing the Sync word used during communication. For LoRa modulation one byte is used, for FSK up to eight bytes can be entered.

Response: ok if the sync word is valid

invalid_param if the sync word is not valid

This command configures the sync word used during communication.

Example: **radio set sync 12** // Set the sync word to a single byte with the value 0x12.

2.5.4.16 `radio set wdt <watchDog>`

`<watchDog>`: decimal number representing the time-out length for the Watchdog Timer, from 0 to 4294967295. Set to '`0`' to disable this functionality.

Response: `ok` if the watchdog time-out is valid

`invalid_param` if the watchdog time-out is not valid

This command updates the time-out length, in milliseconds, applied to the radio Watchdog Timer. If this functionality is enabled, then the Watchdog Timer is started for every transceiver reception or transmission. The Watchdog Timer is stopped when the operation in progress in finished.

Example: **`radio set wdt 2000`** // The Watchdog Timer is configured for 2000 ms.

Note: Ensure the value configured for the Watchdog Timer matches the radio configurations. For example, set the `<watchDog>` value to '`0`' in order to disable this functionality during the radio continuous reception.

2.5.5 Radio Get Commands

TABLE 2-14: RADIO GET COMMANDS

Parameter	Description
`afcbw`	Get the value used by the automatic frequency correction bandwidth.
`bitrate`	Get the frequency shift keying (FSK) bit rate.
`bt`	Get the data shaping for frequency shift keying (FSK) modulation type.
`bw`	Get the value used for the radio bandwidth.
`cr`	Get the coding rate used by the radio.
`crc`	Get if a CRC header is to be used.
`fdev`	Get the frequency deviation allowed by the end device.
`freq`	Get the current operation frequency for the radio.
`iqi`	Get if IQ inversion is used.
`mod`	Get the module Modulation mode.
`prlen`	Get the preamble length used during transmissions.
`pwr`	Get the output power level used by the radio during transmission.
`rxbw`	Get the operational receive bandwidth.
`sf`	Get the requested spreading factor (SF) to be used during transmission.
`snr`	Get the signal noise ratio (SNR) of the last received packet.
`sync`	Get the synchronization word used for communication.
`wdt`	Get the time-out limit for the Watchdog Timer.

2.5.5.1 `radio get afcbw`

Response: float representing the automatic frequency correction band, in kHz. Parameter values can be: 250, 125, 62.5, 31.3, 15.6, 7.8, 3.9, 200, 100, 50, 25, 12.5, 6.3, 3.1, 166.7, 83.3, 41.7, 20.8, 10.4, 5.2, 2.6.

This command reads back the status of the Automatic Frequency Correction Bandwidth.

Default: 41.7

Example: **`radio get afcbw`** // Reads back the current automatic frequency correction bandwidth.

2.5.5.2 radio get bitrate

Response: signed decimal representing the configured bit rate, from 1 to 300000.

This command reads back the configured bit rate for FSK communications.

Default: 50000

Example: **radio get bitrate** // Reads back the current FSK bit rate setting.

2.5.5.3 radio get bt

Response: string representing the configuration for data shaping. Parameter values can be: none, 1.0, 0.5, 0.3.

This command reads back the current configuration for data shaping applied to FSK transmissions.

Default: 0.5

Example: **radio get bt** // Reads the current data shaping FSK configuration.

2.5.5.4 radio get bw

Response: decimal representing the current operating radio bandwidth, in kHz. Parameter values can be: 125, 250 or 500.

This command reads back the current operating radio bandwidth used by the transceiver.

Default: 125

Example: **radio get bw** // Reads back the current operational bandwidth applied to transmissions.

2.5.5.5 radio get cr

Response: string representing the current value settings used for the coding rate. Parameter values can be: 4/5, 4/6, 4/7, 4/8.

This command reads back the current value settings used for the coding rate during communication.

Default: 4/5

Example: **radio get cr** // Reads back the current coding rate transceiver settings.

2.5.5.6 radio get crc

Response: string representing the status of the CRC header, either on or off

This command reads back the status of the CRC header, to determine if it is to be included during operation.

Default: on

Example: **radio get crc** // Reads back if the CRC header is enabled for use.

2.5.5.7 `radio get fdev`

Response: signed decimal representing the frequency deviation setting, from 0 to 200000.

This command reads frequency deviation setting on the transceiver.

Default: 25000

Example: **`radio get fdev`** // Reads back current configured frequency deviation setting.

2.5.5.8 `radio get freq`

Response: decimal number representing the frequency, from 902000000 to 928000000 in Hz.

This command reads back the current operation frequency of the module.

Default: 923300000

Example: **`radio get freq`** // Reads back the current frequency the transceiver communicates on.

2.5.5.9 `radio get iqi`

Response: string representing the status of the Invert IQ functionality, either `on` or `off`.

This command reads back the status of the Invert IQ functionality.

Default: `off`

Example: **`radio get iqi`** // Reads back the status of the Invert IQ functionality.

2.5.5.10 `radio get mod`

Response: string representing the current mode of operation of the module, either `lora` or `fsk`.

This command reads back the current mode of operation of the module.

Default: `lora`

Example: **`radio get mod`** // Reads if module is modulating in LoRa or FSK.

2.5.5.11 `radio get prlen`

Response: signed decimal representing the preamble length, from 0 to 65535.

This command reads the current preamble length used for communication.

Default: 8

Example: **`radio get prlen`** // Reads back the preamble length used by the transceiver.

2.5.5.12 `radio get pwr`

Response: signed decimal representing the current power level, from 2 to 20.

This command reads back the current power level settings used in operation.

Default: 2

Example: **`radio get pwr`** // Reads back the current transmit output power.

2.5.5.13 `radio get rxbw`

Response: float representing the signal bandwidth, in kHz. Parameter values can be: 250, 125, 62.5, 31.3, 15.6, 7.8, 3.9, 200, 100, 50, 25, 12.5, 6.3, 3.1, 166.7, 83.3, 41.7, 20.8, 10.4, 5.2, 2.6.

This command reads back the signal bandwidth used for receiving.

Default: 25

Example: **`radio get rxbw`** // Reads back the receive signal bandwidth.

2.5.5.14 `radio get sf`

Response: string representing the current spreading factor.

This command reads back the current spreading factor being used by the transceiver. Parameter values can be: `sf7`, `sf8`, `sf9`, `sf10`, `sf11`, `sf12`"

Default: `sf12`

Example: **`radio get sf`** // Reads back the current spreading factor settings.

2.5.5.15 `radio get snr`

Response: signed decimal number representing the signal to noise ratio (SNR), from -128 to 127.

This command reads back the Signal Noise Ratio (SNR) for the last received packet.

Default: -128

Example: **`radio get snr`** // Reads back the measured SNR for the previously packet reception.

2.5.5.16 `radio get sync`

Response: hexadecimal number representing the synchronization word used for radio communication.

This command reads back the configured synchronization word used for radio communication. One byte long synchronization word is used for the LoRa modulation while up to eight bytes can be entered for FSK.

Default: 34

Example: `radio get sync`

2.5.5.17 `radio get wdt`

Response: decimal number representing the length used for the watchdog time-out, from 0 to 4294967295.

This command reads back, in milliseconds, the length used for the watchdog time-out.

Default: 15000

Example: **`radio get wdt`** // Reads back the current time-out value applied to the Watchdog Timer

Chapter 3. Bootloader Usage

This chapter describes the operation of the bootloader on the RN2903 SA LoRa modules. The bootloader can be used to upgrade the firmware in the field without requiring the use of a hardware programming tool.

3.1 BOOTLOADER HOSTS

The bootloader requires a host computer to control the bootloader operations and provide the firmware file to be programmed into the RN2903 SA module. The host computer can either be the embedded host microcontroller in the end product, or a PC can be used. If the product's embedded microcontroller will be used with the bootloader, the user must implement the bootloader protocol described later in this section. If a PC is being used as a host for the RN2903 SA module, for example during development, Microchip has a bootloader host application that can be used.

The bootloader on the RN2903 SA module is based on the Microchip's standard 8-bit UART bootloader. This document describes the differences between the standard bootloader and the RN2903 SA bootloader implementation.

The RN2903 SA module bootloader is invoked automatically when the firmware becomes corrupted, or if the firmware is erased.

To invoke the RN2903 SA bootloader to update the firmware, the RN2903 SA system command to erase firmware needs to be executed. For more information, refer to **2.3.3 "sys eraseFW"**.

> **Note:** Once the `sys eraseFW` command is executed, the RN2903 SA module becomes unresponsive to all Microchip LoRa Stack commands. The RN2903 SA module will be in Bootloader mode, and will remain in Bootloader mode until new firmware is successfully installed. The RN2903 SA firmware must be programmed into the module using the bootloader or a hardware programming tool.

For additional information on bootloader usage, refer to **3.5 "Bootloader Usage Examples"**.

3.2 PROTOCOL

The RN2903 SA bootloader is similar to the other Microchip bootloaders. This section describes the specifics of RN2903 SA bootloader protocol. Table 3-1 provides the list of commands required to update the RN2903 SA module firmware.

TABLE 3-1: SUPPORTED COMMANDS

Command	Description
0x00	Get Version and other info
0x01	Reserved
0x02	Write Flash
0x03	Erase Flash
0x04	Reserved

TABLE 3-1: SUPPORTED COMMANDS

Command	Description
0x05	Reserved
0x06	Reserved
0x07	Reserved
0x08	Calculate and return Flash checksum
0x09	Reset Device

The Microchip standard 8-bit UART bootloader has a common command protocol for all commands.

TABLE 3-2: COMMAND PROTOCOL

Byte:	0	1	2	3	4	5	6	7	8
Fields:	CMD	Length(LSB··MSB)		Key 1	Key 2	Address (LSB··MSB)			

<Command><LenLSB><LenMSB><Key1><Key2><address(4 bytes)LSB··MSB>

Byte Order

There are two multi-byte fields common to all commands. These are the Length field, and the Address field. These values are sent in little-endian format. This means that the low-order byte (Least Significant Byte) is sent first, and the high-order byte (Most Significant Byte) is sent last.

Write Operations

When an Erase or Write command is issued, the two key fields must be supplied with correct values. For read operations, they key fields are not used. The values of the keys are always:

- Key1 = 0x55
- Key2 = 0xaa

General Differences from 8-Bit Bootloader

- The module bootloader only uses the first-length byte and ignores the second-length byte in all commands. The protocol still requires that the second length-byte be sent, but the bootloader ignores this value.
- Each command is preceded by 0x55 (ASCII 'U'); this is used for auto-baud detection.

3.3 RN MODULE BOOTLOADER COMMANDS

The command format has an initial byte of 0x55, and the second byte of the length field (MSB) is always 0x00.

The typical bootloader response to commands starts with repeating back the command and follows with a command specific response. Exceptions to this format are described in the commands in which they occur.

TABLE 3-3: RN MODULE BOOTLOADER COMMANDS

Byte:	0	1	2	3	4	5	6	7	8	9
Fields:	0x55	CMD	Len	0x00	Key1	Key2	Address (LSB··MSB)			

<0x55><Command><Len><0x00><Key1><Key2><address(4 bytes)LSB··MSB>

3.4 COMMAND DETAILS

TABLE 3-4: GET VERSION INFO

Byte:	0	1	2	3	4	5	6	7	8	9
Values:	0x55	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

This command returns bootloader version and memory information.

Response:

<get version info command><bytes described in Table 3-5>

TABLE 3-5: GET VERSION INFO COMMAND (BYTES)

Byte	Value
0	Bootloader version – low byte
1	Bootloader version – high byte
2	Max. packet size – low byte (not used)
3	Max. packet size – high byte (not used)
4	ACK packet size – low byte (not used)
5	ACK packet size – high byte (not used)
6	Device ID – low byte
7	Device ID – high byte
8	(not used)
9	(not used)
10	Erase Row size
11	Write Latch size
12	User ID 1
13	User ID 2
14	User ID 3
15	User ID 4

Table 3-5 describes the bytes of information that follow the `get version info` command..

TABLE 3-6: WRITE FLASH

Byte:	0	1	2	3	4	5	6	7	8	9
Values:	0x55	0x02	Len	0x00	Key1	Key2	Address			

Byte:	Len Number of bytes (9 through 8+Len)
Values:	Data[0]-Data[Len-1]

This command writes data into the Flash memory. The length field can range from 0 to 255. This determines the number of bytes written to Flash. The Write Flash command does not erase any Flash memory before writing data; therefore, this command should be preceded by an Erase Flash command for proper operation.

Response:

<write flash command><Status>

`Status` is either a '0' indicating that the command failed, or a '1' indicating the command was successful.

> **Note:** The "Values" data is not returned in the command response. Only the `write flash` command followed by the `Status` is returned.

TABLE 3-7: ERASE FLASH

Byte:	0	1	2	3	4	5	6	7	8	9
Values:	0x55	0x03	Blocks	0x00	Key1	Key2	Address			

This command erases one or more blocks of Flash memory, starting at address *Address*. The Blocks field can range from 0 to 255. The 1-255 value of the Blocks field represents the number of blocks to erase. If the Blocks field is '0', the bootloader will erase 256 blocks.

Response:

<erase flash command><Status>

Status is either a '0' indicating that the command failed, or a '1' indicating the command was successful.

TABLE 3-8: CALCULATE AND RETURN CHECKSUM

Byte:	0	1	2	3	4	5	6	7	8	9
Values:	0x55	0x08	Len	0x00	0x00	0x00	Address			

This command returns the checksum calculated over the Flash memory range beginning at Address for a length of Len bytes. If Len is odd, 1 is added to make Len even.

The checksum algorithm treats the Flash memory as an array of 16-bit values during the calculation, which is not performed byte by byte. In MPLAB® X, this is known as Checksum Algorithm 2.

Response:

<calculate and return checksum command (modified)><CSumLSB><CSumMSB>

Note: The Len field in the returned calculate and return checksum command is set to 0x00 in the reply.

Status is either a '0' indicating that the command failed, or a '1' indicating the command was successful.

TABLE 3-9: RESET DEVICE

Byte:	0	1	2	3	4	5	6	7	8	9
Values:	0x55	0x09	0x00	0x00	0x00	0x00	0x00000000			

This command does not generate a response and immediately performs a software reset of the module.If the firmware update was successful, the RN2903 SA firmware will execute after the reset. If the firmware was not updated, or the update failed, the bootloader will execute after the reset.

3.5 BOOTLOADER USAGE EXAMPLES

3.5.1 Using Bootloader with an Embedded Host

This section provides an overview of the requirements to use an embedded host microcontroller to control the bootloader.

The end user is responsible for implementing application software for an embedded bootloader host. The bootloader communication protocol is described in **3.2 "Protocol"**.

The following is a list of the suggested steps that the bootloader host application should perform:

Erasing the RN2903 SA firmware:

- Check the existing firmware version
  - sys get ver (2.3.6.1)
- Erase the existing firmware
  - sys eraseFW (2.3.3)

The RN2903 SA module will automatically reset, and execute the bootloader.

Installing a new firmware using the bootloader:

- Check the bootloader firmware version
  - Get Version Info (Table 3-4)
- Erase all of the Flash from address `0x300` to `0xFFFF`
  - Erase Flash (Table 3-7) – The erase command must be issued (4) times to erase all the blocks of flash program memory
- Write the new blocks of firmware. Repeat until all blocks have been written
  - Write Flash (Table 3-6)
- Request the firmware checksum from the bootloader. Compare to the value calculated by the host application
  - Calculate and return Flash checksum (Table 3-8)
- Reset the device (Table 3-9)

3.5.2 Using Bootloader with a PC Host

This section describes using the Microchip LoRa Development Utility software application on a PC to update the firmware of the RN2903 SA module.

As of this writing, the current version of the Microchip LoRa Development Utility application for the PC is "`LoRa Development Utility.jar`" version 1.0.1.

Connect the UART on the RN2903 SA to a COM port on the PC. For example, the Microchip MCP2221 USB to Serial Breakout Module (ADM00559) can be used. For information on connecting the UART on the RN2903 SA module, refer to the *RN2903, Low-Power Long-Range LoRa® Technology Transceiver Module Data Sheet* (DS50002390).

Note:	The LoRaDevUtility supports other Microchip products and was intended for other purposes beyond the bootloader functionality. This document only describes the bootloader for Device Firmware Update (DFU) function as used by the RN2903 SA modules.

3.5.2.1 UPDATE RN2903 SA MODULE FIRMWARE USING LORADEVUTILITY

1. Launch the `LoRaDevUtility.jar` application.
2. The LoraDevUtility probes all the available serial ports on the PC. The left win-

dow pane lists all or the Microchip LoRaWAN devices found.

3. The RN2903 SA module is displayed as `RN Module n`. Where `n` is a unique number assigned by the LoRaDevUtility. Select "RN Module" in the left pane.
4. The center and right panes are displayed. In the center pane, click the `DFU` (Device Firmware Update) tab.
5. In the DFU pane click the button next to the "Select File" field. Navigate to the RN2903 SA LoRaWAN `hex` file containing the firmware update.
6. Click "Update Firmware" button. Notice the RN Module Console pane on the right. A message `sys eraseFW` appears at the bottom of the list. After a few seconds, the bootloader starts and the message `Device: COMn Bootloading Started` is displayed. When the bootloading operation has completed, the message `Device: COMn Bootloading Successful` appears. The bootloading operation may take 20 seconds or more depending on the speed of the connection.

Note: Sometimes the communication link may be disconnected after the `sys eraseFW` command, and before the bootloading operation completes. If the messages `Device: COMn Bootloading started`, or `Device: COMn Bootloading Successful` do not appear, additional steps must be performed. The RN2903 SA module is in Bootloader mode and will remain in Bootloader mode until new firmware is successfully installed. If bootloading process does not complete, follow the steps in **3.5.2.2 "LoRaDevUtility - Boot Load Recover"** to complete the bootloading operation.

3.5.2.2 LORADEVUTILITY - BOOT LOAD RECOVER

Skip this section if the `Device: COMn Bootloading Successful` message is displayed in the right pane.

If the communication link is disconnected, the message `RN Module n disconnected. Communications Link Failure` appears in the bottom pane of the LoRaDevUtility screen. Perform the following steps to use the Boot Load Recover function:

1. Remove any `RN Module n` devices listed in the left pane by clicking on the `X` next to the name.
2. From the LoRaDevUtility Module menu, select "Boot Load Recover".
3. A list of COM ports appear in the left pane. Select the COM port connected to the RN2903 SA device.
4. The `DFU` tab appears in the center pane. To complete the bootloading operation follow steps 5 and 6 from **3.5.2.1 "Update RN2903 SA Module Firmware Using LoRaDevUtility"**.

3.5.2.3 BOOTLOADING OPERATION COMPLETE

After successfully updating the RN2903 SA module firmware using the bootloader, close the LoRaDevUtility application. The RN2903 SA module is now ready for use.

Appendix A. Current Firmware Features and Fixes

Please check the product web page for the current RN2903 SA firmware version at www.microchip.com/lora.

A.1. Version 1.0.3

Initial release of the firmware.

THE MICROCHIP WEBSITE

Microchip provides online support via our WWW site at www.microchip.com. This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip website at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the website at: http://microchip.com/support

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820