LoRaWAN Modem Module Command Reference Manual

Semtech

April 2020

Contents

1. Introduction

The LoRaWAN® Modem Module (also referred to as Modem) is a software set running on a LoRa®-based module to provide an API for end-node operation. It includes the LoRaWAN MAC layer functions, as well as application layer functions which can be consumed via the Device and Application Services in the Semtech LoRa Cloud Service (https://www.loracloud.com/portal/).

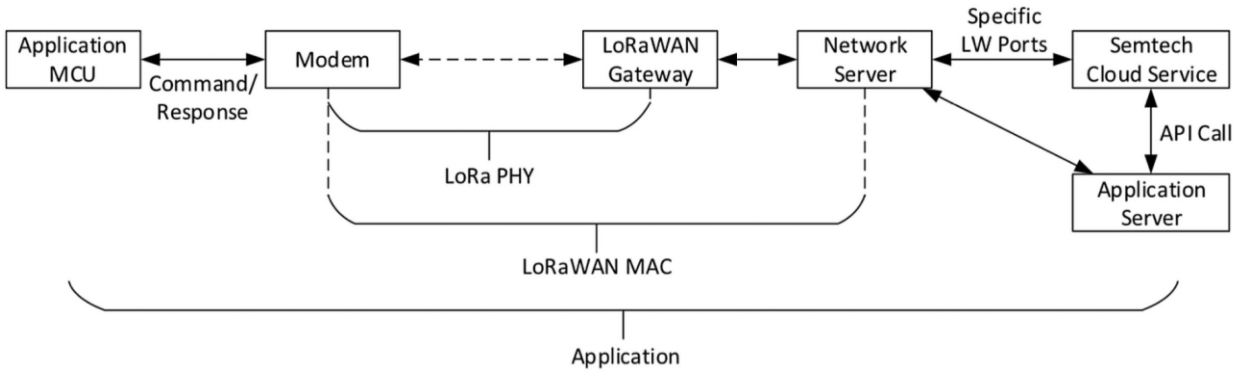

Figure 1: Scope of the Modem and modem service

2. Murata LoRa Module Pins

Table 1: Murata LoRa Module Pin Descriptions

Pin No.	Terminal Name	Type	Description
1	PA12/USB_DP	O	TCXO power supply output. Must be externally connected to pin 48.
2	PA11/USB_DM	I/O	Do Not Connect. Internal Use
3	GND	GND	Ground.
4	VDD_USB	PWR	Power supply for TCXO (pin 1).
5	VDD_MCU	PWR	Power supply from MCU.
6	VDD_RF	PWR	Power supply for RF section.
7	GND	GND	Ground.
8	DBG_SX1276_DIO2	—	Do Not Connect. Internal Use
9	DBG_SX1276_DIO3	—	Do Not Connect. Internal Use
10	DBG_SX1276_DIO4	—	Do Not Connect. Internal Use
11	DBG_SX1276_DIO5	—	Do Not Connect. Internal Use
12	DBG_SX1276_DIO1	—	Do Not Connect. Internal Use
13	DBG_SX1276_DIO0	—	Do Not Connect. Internal Use
14	PB15/SPI2_MOSI	I/O	Do Not Connect.
15	PB14/SPI2_MISO	I/O	Do Not Connect.
16	PB13/SPI2_SCK	I/O	Do Not Connect.
17	PB12/SPI2_NSS	I/O	Do Not Connect.
18	PA10/USART1_RX	I	UART RX (host-to-modem)
19	PA9/USART1_TX	O	UART TX (modem-to-host)
20	PA8/MCO	O	BUSY signaling line.
21	PA5/ADC5/DAC2	O	Diagnostics LED (optional). Active high, connect to LED with appropriate in-series resistor
22	PA4/ADC4/DAC1	I/O	Do Not Connect.

Pin No.	Terminal Name	Type	Description
23	PA3/ADC3	I	Internal Use. Debug UART_RX
24	PA2/ADC2	O	Internal Use. Debug UART_TX
25	GND	GND	Ground.
26	ANT	I/O	Transmit/Receive antenna port
27	GND	GND	Ground.
28	DBG_CRF1	—	Do Not Connect. Internal Use
29	DBG_CRF3	—	Do Not Connect. Internal Use
30	DBG_CRF2	—	Do Not Connect. Internal Use
31	STSAFE_nRST	I	Do Not Connect. Internal Use
32	VREF+	PWR	Reference voltage for the MCU ADC section. Must be externally connected to VDD
33	PA0/WKUP1	I/O	Do Not Connect.
34	MCU_nRST	I	MCU reset pin. Has internal pull-up and may be left floating.
35	PB8/I2C1_SCL	I	COMMAND signaling line.
36	PB9/I2C1_SDA	O	EVENT signaling line.
37	PB2/LPTIM1_OUT	O	Host MCU reset line. Active low, Z (high impedance) when inactive.
38	PB7/LPTIM1_IN2	I/O	Do Not Connect.
39	PB6/LPTIM1_ETR	I/O	Do Not Connect.
40	PB5/LPTIM1_IN1	I/O	Do Not Connect.
41	PA13/SWDIO	I/O	Internal Use. Programming SWDIO
42	PA14/SWCLK	I/O	Internal Use. Programming SWCLK
43	BOOT0	I	Do Not Connect. Internal Use
44	GND	GND	Ground
45	PH1-OSC_OUT	I/O	Do Not Connect.
46	PH0-OSC_IN	I/O	Do Not Connect.

Pin No.	Terminal Name	Type	Description
47	TCXO_OUT	O	TCXO output.
48	VDD _TCXO	PWR	TCXO power supply input. Externally connected to pin 1.
49-57	GND	GND	Ground

Note: GPIO pins are to be left OPEN if not used.

3. Modem Interface

Communication between the host MCU and the modem is performed using a USART serial port and four additional signalling lines.

3.1 Serial Port

USART1 of the modem is used to exchange commands and responses.

- Pin 18: receive (host-to-modem)
- Pin 19: transmit (modem-to-host)

The following serial port parameters are used:

Table 2: Serial Port Parameters

Parameter	Value
Baud Rate	115,200 bps
Data Bits	8
Stop Bits	1
Parity	None

3.2 GPIO Signaling Lines

3.2.1 COMMAND Line

Active-high, GPIO output line

This line is used to signal to the Modem that the Application MCU is about to transmit a command. After driving the line low, the host has to wait until the BUSY line goes low (or at least 10 ms) before sending the first character. This allows the Modem to wake up and start reception. After the command is sent, the line must be driven high, at which point the Modem will process the command. If a valid command was received, the Modem will transmit its response within 200ms.

3.2.2 BUSY Line

Active-high, GPIO output line

This line signals whether the Modem is busy or ready to receive commands. It is high while the Modem is busy and will go low as soon as the Modem is ready to receive a command. The Modem should be ready to receive after a maximum of 10 ms after the COMMAND line has been asserted. The BUSY line will go high again after the command has been received and the COMMAND line has been released.

3.2.3 EVENT Line

Active-high, GPIO output line

This line signals to the Application MCU that the Modem has asynchronous event data pending. The Application MCU can use the *GetEvent* command to retrieve such data.

3.2.4 MCURESET line

Active-low, GPIO output and floating line

This line signals a request to reset the Application MCU and is pulled low by the modem for 10ms.

4. Message Flow

Messages are exchanged via USART1 using the COMMAND, BUSY and EVENT GPIO lines for synchronization. This section describes the timing for message flow between Application MCU and the Modem.

The COMMAND line is set by the Application MCU; the BUSY and EVENT lines are set by the Modem.

All serial communication consists strictly of Application MCU-initiated command-response sequences

4.1 Message Flow for Command and Response

Flow for sending commands and reading synchronous responses:

a. Sending commands is initiated via the COMMAND line: the Application MCU pulls the COMMAND line low.

b. The Modem drives the BUSY line low.

c. The Application MCU sends the command using the USART1 RX line.

d. The Application MCU must drive the COMMAND line high after the last character of the command has been transmitted. The Modem will only start interpreting the command after the COMMAND line is high.

e. The Modem drives the BUSY line high.

f. If the checksum has been validated the Modem will process the command and will send a matching response within 200 ms, otherwise no response will be sent. The spacing between characters of the response will not exceed 5ms.

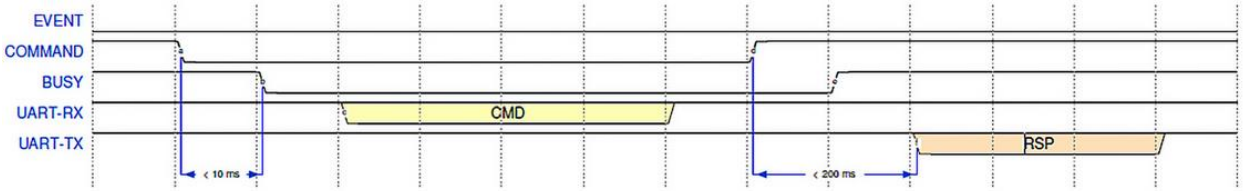

Figure 2: Timing for synchronous command/response pair

Notes:

- There is one exception to when the response might be returned later than 200 ms. When the last part of an update is processed by the *FirmwareUpdate* command, a synchronous long-running digital signature verification is triggered and the response is returned within one second.
- The Application MCU must not send a new command before the response has been fully received.
- The Modem will not send any unsolicited response to the serial port. If event data becomes available asynchronously, the Modem will signal this using the EVENT line.

4.2 Message Flow for Asynchronous Events

When the Modem has asynchronous data available it drives the EVENT Line. The event data can be retrieved via the *GetEvent* command using the message flow described above. The EVENT line is cleared when all pending events have been retrieved (see Events).

5. Message Format

This section describes the packet format running on the USART1 interface, for both command packets and response packets. The maximum packet size is 258 bytes.

5.1 Command Packets

All command messages consist of a command code that defines the operation to be performed, a payload length and optional payload, and a trailing checksum character which is the XOR of all previous bytes.

Table 3: Command message structure

Field	Size (bytes)	Description
cmd	1	Command code (see Table 6)
len	1	Length of the payload field
payload	0-255	Parameters for the command as binary data
chk	1	Packet checksum (xor over cmd, len, payload)

The Modem begins analyzing the command message after the COMMAND line is released. After the packet structure is successfully verified (correct number of bytes and correct check character), the Modem will interpret the command and, will send a response. If the packet structure is invalid, the Modem ignores the packet and will send a response with the return code *FrameError.*

5.2 Response Packets

The response format is similar to the command format; instead of the command code, it holds a return code that indicates the successful completion of the command or an error condition.

Table 4: Response message structure

Field	Size (bytes)	Description
rc	1	Return code of the command
len	1	Length of the payload field
payload	0-255	Response data of the command as binary data
chk	1	Packet checksum (xor over rc, len, payload)

The return code bytes are described in Table 5 below.

Table 5: List of Return Codes

Code	Name	Description
0x00	*Ok*	Command executed without errors
0x01	*Unknown*	Command code unknown
0x02	*NotImpl*	Command not yet implemented
0x03	*NotInit*	Command not initialized
0x04	*Invalid*	Command parameters invalid
0x05	*Busy*	Command cannot be executed now
0x06	*Fail*	Command execution failed
0x07	*BadFmt*	Format check failed
0x08	*BadCrc*	CRC check failed
0x09	*BadSig*	Signature verification failed
0x0A	*BadSize*	Size check failed
0x0B	*NoSession*	Command requires session
0x0F	*FrameError*	Serial port framing error

6. Commands and Responses

6.1 Overview of Commands and Responses

The serial communication with the Modem is performed using strictly alternating command and response messages. Response messages convey a return code and the result of the operation triggered by the preceding command message.

Table 6: List of Commands and Responses

Name	Code	Description	Input	Output
GetEvent	0x00	Retrieve pending events		type[1], count[1], eventdata[n]
GetVersion	0x01	Get version		bootversion[4], firmware version[4], lorawan[2]
Reset	0x02	Reset Modem		
FactoryReset	0x03	Perform Modem factory reset		
ResetCharge	0x04	Reset charge counter		
GetCharge	0x05	Get accumulated charge offset		charge[4]
GetTxPowerOffset	0x06	Get power correction offset		offset[1]
SetTxPowerOffset	0x07	Set TX power correction offset	offset[1]	
Test	0x08	Radio Test function	(see description)	(see description)
FirmwareUpdate	0x09	Write firmware update	part[4+128]	
GetTime	0x0A	Get GPS wall time		timestamp[4]
GetStatus	0x0B	Get Modem status		status[1]

Name	Code	Description	Input	Output
SetAlarmTimer	0x0C	Set alarm / wakeup timer	seconds[4]	
GetTrace	0x0D	Get diagnostic crash log		backtrace log[n]
GetPin	0x0E	Get device registration PIN		PIN[4]
GetChipEui	0x0F	Get device chip EUI		ChipEUI[8]
GetJoinEui	0x10	Get join EUI		JoinEUI[8]
SetJoinEui	0x11	Set join EUI and derive keys	JoinEUI[8]	
GetDevEui	0x12	Get device EUI		DeviceEUI[8]
SetDevEui	0x13	Set device EUI and derive keys	DeviceEUI[8]	
SetNwkKey	0x14	Set network key	NwkKey[16]	
GetClass	0x15	Get the LoRaWAN device Class		class[1]
SetClass	0x16	Set LoRaWan class A/C	class[1]	
SetMulticast	0x17	Set multicast session parameters	param[40]	
GetRegion	0x18	Get regulatory region		region[1]
SetRegion	0x19	Set regulatory region	region[1]	
ListRegions	0x1A	List the supported regulatory regions		region[1-5]
GetAdrProfile	0x1B	Get ADR profile		type[1]
SetAdrProfile	0x1C	Set ADR profile and optional parameters	type[1]+list[16]	
GetDmPort	0x1D	Get DM port		port[1]
SetDmPort	0x1E	Set DM port	port[1]	
GetDmInfoInterval	0x1F	Get DM reporting interval		interval[1]

Name	Code	Description	Input	Output
SetDmInfoInterval	0x20	Set DM reporting interval	interval[1]	
GetDmInfoFields	0x21	Get default info fields for DM status		inflist[n]
SetDmInfoFields	0x22	Set default info fields for DM status	inflist[n]	
SendDmStatus	0x23	Send DM status now	inflist[n]	
SetAppStatus	0x24	Set application-specific status for DM	appstatus[8]	
Join	0x25	Start joining the network		
LeaveNetwork	0x26	Leave the network		
SuspendModemComm	0x27	Suspend/resume radio operations	suspend[1]	
GetNextTxMaxPayload	0x28	Get max payload size for next TX		size[1]
RequestTx	0x29	Transmit frame unconfirmed or confirmed	port[1], conf[1], data[n]	
EmergencyTx	0x2A	Transmit frame immediately (smoke alarm)	port[1], conf[1], data[n]	
UploadInit	0x2B	Set file upload port, encryption mode, size, delay	port[1], enc[1], sz[2], delay[2]	
UploadData	0x2C	Write data for file upload transmission	data[n]	
UploadStart	0x2D	Verify data and start file upload	crc[4]	
StreamInit	0x2E	Set data stream parameters	port[1], mode[1]	
SendStreamData	0x2F	Send data stream record	port[1] + record[n]	

Name	Code	Description	Input	Output
StreamStatus	0x30	Retrieve stream status	port[1]	pending[2] + free[2]

6.2 Command Details

All commands have a command code and, optionally, can have parameters to control the requested operation. All responses have a return code indicating success or failure and, optionally, data provided by the command.

Detailed parameters, return codes, and return data are described for every command in the following subsections. If not mentioned otherwise, the commands do not have parameters or return data.

Note: All multi-byte fields representing integers contained in command or response payloads are encoded in big endian byte order (MSBF, most-significant-byte-first).

The return codes listed below are common to all commands and are not explicitly explained in the per-command description:

Table 7: Common Return Codes

Name	Description
Ok	Successful operation
Invalid	Bad parameter values
Busy	Radio is required but is muted/suspended, or parameters of active session cannot be changed
NoSession	Session is required but doesn't exist
FrameError	Serial port framing error

6.2.1 EmergencyTx

This command sends the given data on the specified port as an unconfirmed (0x00) or confirmed (0x01) frame immediately. It has a higher priority than all other services and does not take duty cycle or payload size restrictions into account. It can be used to signal an alarm condition (like smoke alarm) in real-time, but it should be used with caution. A *TxDone* event is generated when the frame has been sent. The parameter of the *TxDone* event indicates whether the frame was sent and acknowledged (0x02), or sent but not acknowledged (0x01).

Note: The application shall not use port 0 or the LoRaWAN test port 224 as well as the ports from 225 to 255 since they are reserved for future standardized application extensions.

Command Payload Format

Field	Size (bytes)	Description
port	1	Uplink port number
conf	1	`0x00`=unconfirmed, `0x01`=confirmed
data	1-242	Data

Return codes: *OK, Invalid, Busy, NoSession*

See also: RequestTx, Join, GetNextTxMaxPayload

6.2.2 FactoryReset

This command performs a factory reset. In addition to the MCU reset all persistent settings are reset back to their factory state. Table 8 provides the default values.

Table 8: Default Configuration after Factory Reset

Config Setting	Default Value
DM port	200
Reporting interval	1h
Periodic *dminfo* fields	status, charge, temp, signal, uptime, rxtime
ADR profile	0 (network-controlled)
Regulatory region	`0x01` (EU868)
Board Tx power offset	0
Data streaming port	disabled

Return codes: *OK, Invalid*

See also: Reset

6.2.3 FirmwareUpdate

This command is used to repeatedly store parts of a firmware update to be installed. The firmware data must be split into blocks of 128 bytes each and must be provided in ascending order.

Command Payload Format

Field	Size (bytes)	Description
blkno	2	Current block number (0 to blkcnt-1)
blkcnt	2	Total number of blocks
blkdata	128	Firmware section starting at byte blkno*128. If the firmware size is not a multiple of 128, the last block (blkno=blkcnt-1) may contain less than 128 bytes.

When the update has been fully stored (last part, blkno = blkcnt-1), the code signature and integrity will be validated and the update will be registered for installation. Since the signature verification is computationally intensive, the processing of the last part will take longer than other commands and the response will be returned within one second (see section 4.1 Message Flow for Command and Response). The update will be installed at the next reset of the modem. When processing the last part, the following special return codes can be returned:

Return Codes when Processing Signature Verification

RC	Description
BadSize	Bad size field in firmware update header
BadCrc	Firmware update CRC checksum error
BadSig	Invalid or non-trusted code signature
Fail	Firmware update rejected by bootloader

Return codes: *OK, Invalid, BadSize, BadCrc, BadSig, Fail*

See also: Reset

6.2.4 GetAdrProfile

This command returns the ADR profile type.

Response Payload Format

Field	Size (bytes)	Description
type	1	Current ADR profile type

Return codes: *OK, Invalid*

See also: SetAdrProfile

6.2.5 GetCharge

This command returns the total charge counter of the Modem in mAh. This value includes the accumulated charge since the production of the Modem, or since the last invocation of the *ResetCharge* command.

Response Payload Format

Field	Size (bytes)	Description
charge	4	Accumulated charge counter in mAh

Return codes: *OK, Invalid*

6.2.6 GetChipEui

This command returns the ChipEUI. The ChipEUI is also the default DeviceEUI. It is programmed during manufacturing and is immutable.

Response Payload Format

Field	Size (bytes)	Description
chipeui	8	Chip EUI

Return codes: *OK, Invalid*

See also: GetDevEui, GetJoinEui

6.2.7 GetClass

This command returns the LoRaWAN device class.

Response Payload Format

Field	Size (bytes)	Description
class	1	LoRaWAN device class

Return codes: *OK, Invalid*

See also: SetClass

6.2.8 GetDevEui

This command returns the Device EUI.

Response Payload Format

Field	Size (bytes)	Description
deveui	8	Device EUI

Return codes: *OK, Invalid*

See also: GetDevEui, GetChipEui, GetJoinEui, SetJoinEui

6.2.9 GetDmInfoFields

This command lists the info fields to be included in the periodic DM status messages.

Response Payload Format

Field	Size (bytes)	Description
inflist	n	List of tag bytes

Return codes: *OK, Invalid*

See also: SetDmInfoFields, Uplink Message Format

6.2.10 GetDmInfoInterval

This command returns the device management status reporting interval. The interval is specified in seconds, minutes, hours or days.

Response Payload Format

Field	Size (bytes)	Description
interval	1	Status reporting interval

Return codes: *OK, Invalid*

See also: SetDmInfoInterval, Format of Reporting Interval

6.2.11 GetDmPort

This command returns the device management port.

Response Payload Format

Field	Size (bytes)	Description
port	1	Device management port

Return codes: *OK, Invalid*

See also: SetDmPort

6.2.12 GetEvent

This command can be used to retrieve pending events from the Modem. Pending events are indicated by the EVENT line. The EVENT line will be de-asserted after all events have been retrieved and no further events are available. When no event is available, this command returns with empty response payload.

Response Payload Format

Field	Size (bytes)	Description
type	1	Event type, see Table 14
count	1	Number of missed events of this type in case of overrun
data	0-253	Event-specific data

Events that are not retrieved by the application might be overwritten by new event data of the same type. In this case, only the latest event data will be returned. The count field indicates how many events of this type have been missed.

Return codes: *OK, Invalid*

See also: Events

6.2.13 GetJoinEui

This command returns the Join EUI.

Response Payload Format

Field	Size (bytes)	Description
joineui	8	Join EUI

Return codes: *OK, Invalid*

See also: SetJoinEui, GetDevEui, SetDevEui, GetChipEui

6.2.14 GetNextTxMaxPayload

This command returns the maximum application payload size possible (according to the LoRaWAN regional parameters) for the next transmission, using the current data rate while assuming no FOpts are present and that a device is not behind a repeater.

Response Payload Format

Field	Size (bytes)	Description
maxpayload	1	Max app payload size for next TX

Return codes: *OK, Invalid, NoSession*

See also: RequestTx

6.2.15 GetPin

This command returns the device registration PIN.

Response Payload Format

Field	Size (bytes)	Description
pin	4	Device registration PIN

Return codes: *OK, Invalid*

6.2.16 GetRegion

This command returns the regulatory region.

Response Payload Format

Field	Size (bytes)	Description
region	1	Region code

Table 9: Region Codes

Region	Code
EU868	`0x01`
US915	`0x03`

Return codes: *OK, Invalid*

See also: SetRegion, ListRegions

6.2.17 GetStatus

This command returns the Modem status, which may indicate one or more notification conditions.

Response Payload Format

Field	Size (bytes)	Description
status	1	Modem status

Table 10: Status Bits

Bit	Value	Name	Description
0	0x01	*Brownout*	Reset after brownout
1	0x02	*Crash*	Reset after panic
2	0x04	*Mute*	Device is muted
3	0x08	*Joined*	Device has joined the network
4	0x10	*Suspend*	Radio operations suspended (low power)
5	0x20	*Upload*	File upload in progress
6	0x40	*Joining*	Device is trying to join the network
7	0x80	*Stream*	Data stream in progress

Return codes: *OK, Invalid*

6.2.18 GetTime

Queries the current GPS time. The application-layer clock synchronization protocol is used to link the device clock to GPS time. The returned time specifies the seconds since the GPS epoch (00:00:00, Sunday 6th of January 1980). If the device is not yet synchronized to GPS time then the returned value is zero. This may happen if the server has not yet answered time sync requests. The accuracy of the synchronization is in the range of seconds and depends on latencies in the network infrastructure.

Response Payload Format

Field	Size (bytes)	Description
seconds	4	Seconds since GPS epoch

Return codes: *OK, Invalid*

6.2.19 GetTrace

This command returns the latest crash log saved by the current firmware. The format of the crash log data is proprietary and will not be disclosed.

Response Payload Format

Field	Size (bytes)	Description
trace	n	Proprietary crash log data

Return codes: *OK, Invalid*

6.2.20 GetTxPowerOffset

This command returns the board-specific correction offset for transmission power to be used (signed integer in dB).

Response Payload Format

Field	Size (bytes)	Description
offset	1	Tx power correction offset

Return codes: *OK, Invalid, NoSession*

See also: SetTxPowerOffset

6.2.21 GetVersion

This command returns the version of the bootloader, the version of the installed firmware, and the version of the implemented LoRaWAN standard (in BCD, e.g. 0x0103 for LoRaWAN 1.0.3).

Response Payload Format

Field	Size (bytes)	Description
bootversion	4	Boot loader version
fwversion	4	Modem firmware version
lwversion	2	LoRaWAN version (BCD)

Return codes: *OK, Invalid*

6.2.22 Join

This command starts joining or re-joining a device to the network. During the join procedure, no further transmissions can occur. When the network has been successfully joined, a *Joined* event is generated. If the device is already joined to a network, or is in the process of joining, this command has no effect.

Once this command has been issued, the Modem will attempt to join the network indefinitely while adhering to the join duty cycle mandated by the LoRaWAN specification. To stop an ongoing join attempt, use the *LeaveNetwork* command. Whenever a cycle of trying all data rates has failed, a *JoinFail* event is generated. After a short back-off time, the join procedure continues. Since the join procedure uses an internal strategy of trying different data rates, it does not adhere to the ADR profile configured by the *SetAdrProfile* command. The ADR profile only applies to uplinks after the device has joined the network and the session is already established.

Return codes: *OK, Invalid, Busy*

See also: LeaveNetwork, RequestTx

6.2.23 LeaveNetwork

This command causes the device to leave the network if already joined, or cancels an ongoing join process. After leaving the network, no further transmissions can occur.

Return codes: *OK, Invalid*

See also: Join

6.2.24 ListRegions

This command returns the regulatory regions supported by the Modem (EU868 and US915).

Response Payload Format

Field	Size (bytes)	Description
regionlist	n	List of supported region codes

Return codes: *OK, Invalid*

See also: GetRegion, SetRegion

6.2.25 RequestTx

This command requests sending the given data on the specified port as an unconfirmed or confirmed frame.

Command Payload Format

Field	Size (bytes)	Description
port	1	Uplink port number
conf	1	`0x00`=unconfirmed, `0x01`=confirmed
data	n	Data

Note: The application shall not use port 0 or the LoRaWAN test port 224 nor the ports from 225 to 255 since they are reserved for future standardized application extensions.

The request will be queued and the frame will be sent as soon as the current bandwidth usage of the regulatory region permits. A *TxDone* event is generated when the frame either has been sent, or couldn't be sent because the specified data exceeded the maximum possible payload size. The parameter of the *TxDone* event indicates whether the frame was sent and acknowledged (`0x02`), sent but not acknowledged (`0x01`) or not sent (`0x00`). When application downlink data has been received in RX1 or RX2 a *DownData* event will be generated containing the port and data received. If a *RequestTx* command is issued before the *TxDone* event of a previous transmission request is generated the command will fail

with *Busy* return code. If the command is issued before the network has been joined it will fail with *NotInit* return code.

6.2.26 Reset

This command performs a reset of the Modem MCU. All transient state information (including session data) will be lost and the Modem will need to join the network again.

Return codes: *OK, Invalid*

See also: FactoryReset

6.2.27 ResetCharge

This command resets the accumulated charge counter to zero.

Return codes: *OK, Invalid*

See also: GetCharge

6.2.28 SendDmStatus

This command sends the specified set of information fields in one or more DM status messages immediately. The set is specified as a list of field codes as defined in Uplink Message Format. Duplicate and invalid fields will be rejected (see note in Periodic Status Reporting).

Command payload format

Field	Size (bytes)	Description
taglist	1	List of information field tags

Return codes: *OK, Invalid, NoSession*

See also: SetDmInfoFields, Uplink Message Format, Periodic Status Reporting

6.2.29 SendStreamData

This command adds a new data record to the buffer of the data streaming encoder for the given port. Whenever the buffer contains data records, the Modem autonomously retrieves data from the buffer, optionally encrypts it, adds redundancy, and sends uplinks containing the redundant stream. New data records can be added by the application at any time, provided there is enough space in the buffer. (The total buffer size is 512 bytes). When all data in the buffer has been sent, a *StreamDone* event is generated. Encryption of the data stream is done according to the mode parameter of the previously-issued *StreamInit* command for the port. If the *StreamInit* command was never issued, the data stream is sent unencrypted to the given port.

Command Payload Format

Field	Size (bytes)	Description
port	1	Port
record	1-254	Record

Return codes: *OK, Invalid, Busy, NoSession, NotInit, BadSize, Fail*

See also: StreamInit, StreamStatus, GetDmPort

6.2.30 SetAdrProfile

This command sets the ADR profile and parameters.

Command Payload Format

Field	Size (bytes)	Description
type	1	ADR profile type 0-3
param	0 or 16	Optional profile parameters

The following ADR profiles types are supported:

Table 11: ADR Profile Types

Name	Code	Parameters
Network Server Controlled	`0x00`	none
Mobile Long Range	`0x01`	none
Mobile Low Power	`0x02`	none
Custom	`0x03`	list of preferred data rates [16]

ADR profile types 0-2 have predefined settings and do not need parameters. The custom ADR profile (type 3) takes a list of 16 preferred data rates as parameter. For every transmission, a random entry in that list is selected. This makes it possible to create distributions of preferred data rates. For example, the list `00 00 00 00 00 00 00 00 01 01 01 01 02 02 03 03` results in a distribution of 50% DR0, 25% DR1, 12.5% DR2, and 12.5% DR 3. If the selected data rate is unavailable at the time of transmission, the closest available data rate is used instead.

The predefined profiles use the following distributions:

- Network Server Controlled: dynamic (for stationary devices)
- Mobile Long Range: 50% MinDr, 25% MinDr+1, 25% MinDr+2
- Mobile Low Power: 25% MaxDr, 25% MaxDr-1, 25% MaxDr-2, 25% MaxDr-3

For the US915 region, MaxDr=4, and MinDr=0; for the EU868 region, MaxDr=7, and MinDr=0. As with custom profiles, if the selected data rate is unavailable at the time of transmission, the closest available data rate is used instead.

Note: The configured ADR profile applies to all uplinks generated by the modem once it has established a session. However it does not apply to the join procedure which has its own data rate strategy.

Return codes: *OK, Invalid*

See also: GetAdrProfile

6.2.31 SetAlarmTimer

This command sets an application alarm timer (in seconds). When the timer expires an *Alarm* event is generated. If this command is issued again before the timer has expired, the timer will be restarted with the new period. A value of 0 will cancel a previously-set pending alarm timer.

Command Payload Format

Field	Size (bytes)	Description
period	4	Alarm time in seconds

Return codes: *OK, Invalid*

6.2.32 SetAppStatus

This commands sets application-specific status information to be reported to the DM service. This information is an application-defined and arbitrary 8-byte data blob. Once set, it is included in the *appstatus* information field that is sent as part of the periodic status reports to the DM service.

Note: This command does not trigger an immediate status report. If the application needs to send the status immediately, it can issue the *SendDmStatus* command with the *appstatus* tag.

The application status is not stored persistently, i.e., after reset, no application status will be reported.

Command Payload Format

Field	Size (bytes)	Description
appstatus	8	Application-specific data

Return codes: *OK, Invalid*

See also: SendDmStatus, SetDmInfoInterval, Uplink Message Format

6.2.33 SetClass

This command sets the LoRaWAN device class. If the command is successful, a change from Class A to Class C is effective after a completed TX transaction. The network server should also be informed about the class change, typically on a separate channel (for LoRaWAN 1.0.3). For a change from Class C to Class A, the RX remains enabled until the next TX transaction. Note that the class settings are not persistent.

Command Payload Format

File	Size (bytes)	Description
class	1	Class value

Table 12: Supported LoRaWAN Classes

LoRaWAN Class	Value	Description
A	`0x00`	Open short RX windows after TX
C	`0x01`	RX remains on

Return codes: *OK, Invalid*

See also: SetClass

6.2.34 SetDevEui

This command sets the Device EUI.

Command Payload Format

Field	Size (bytes)	Description
deveui	8	Device EUI

Note: This command triggers the derivation of a new DevKey, unless a custom key was set using the *SetNwkKey* command.

The command can only be issued as long as the Modem has not yet joined, or is trying to join, the network. Otherwise, the command will fail with the *Busy* return code.

Return codes: *OK, Invalid, Busy*

See also: GetDevEui, GetJoinEui, SetJoinEui

6.2.35 SetDmInfoFields

This command is used to specify the set of information fields to be reported in the periodic DM status messages. The set is specified as a list of field codes as defined in section 9.1 Uplink Message Format.

Duplicate and invalid fields will be rejected (see note in section 9.1.1). An empty set is valid and will effectively disable the DM status message.

Command Payload Format

Field	Size (bytes)	Description
taglist	n	List of status information field tags

Return codes: *OK, Invalid*

See also: SetDmPort, Uplink Message Format, Periodic Status Reporting

6.2.36 SetDmInfoInterval

This command sets the device management reporting interval. The interval is specified in seconds, minutes, hours or days.

Command Payload Format

Field	Size (bytes)	Description
interval	1	Status reporting interval (see section 9.1.2)

The value zero (seconds, minutes, hours or days) disables the status reporting.

Return codes: *OK, Invalid*

See also: GetDmInfoInterval, Format of Reporting Interval

6.2.37 SetDmPort

This command sets the device management port. Port 0 and the ports from 224 to 255 must not be used since they are reserved for future standardized application extensions.

Command Payload Format

Field	Size (bytes)	Description
port	1	Device management port

Return codes: *OK, Invalid, Busy*

See also: GetDmPort

6.2.38 SetJoinEui

This command sets the JoinEUI.

Command Payload Format

Field	Size (bytes)	Description
joineui	8	Join EUI

Note: This command triggers the derivation of a new DevKey, unless a custom key was set using the SetNwkKey command.

The command can only be issued as long as the Modem has not yet joined, or is trying to join, the network. Otherwise, the command will fail with the *Busy* return code.

Return codes: *OK, Invalid, Busy*

See also: GetJoinEui, GetDevEui, SetDevEui

6.2.39 SetMulticast

This command configures a LoRaWAN downlink multicast session. For details, refer to the *LoRaWAN Specification* v 1.0.3 and the *LoRaWAN Remote Multicast Setup Specification*. Multicast groups are not persistent and are always kept until the next reset. A maximum of two multicast groups is allowed. If the group address is already configured it will be kept and the session parameters will be overwritten.

Command Payload Format

Field	Size (bytes)	Description
grpaddr	4	Multicast group address
nwkkeydn	16	Multicast group network key to verify the MIC of downlink traffic
appkey	16	Multicast group application key to decrypt the application payload
seqnoadn	4	The initial sequence counter for the application downlinks

Return codes: *OK, Invalid*

See also: SetClass

6.2.40 SetNwkKey

This command sets the LoRaWAN 1.0.3 device key.

Command Payload Format

Field	Size (bytes)	Description
devkey	16	Device key

The command can only be issued as long as the Modem has not yet joined, or is trying to join, the network. Otherwise, the command will fail with the *Busy* return code.

Return codes: *OK, Invalid, Busy*

See also: SetDevEui, SetJoinEui

6.2.41 SetRegion

This command sets the regulatory region. Only region codes listed in Table 9 may be used. Additionally, this command resets the ADR profile to *Network Server Controlled*. If a different ADR profile is desired, the profile needs to be set again.

Command Payload Format

Field	Size (bytes)	Description
region	1	Region code

Return codes: *OK, Invalid, Busy*

See also: GetRegion, ListRegions, SetAdrProfile, Region Codes

6.2.42 SetTxPowerOffset

This command sets the board-specific correction offset for the transmission power to be used. The offset depends on the board design and antenna matching, and is expressed in dB (signed integer).

Command Payload Format

Field	Size (bytes)	Description
offset	1	TX power correction offset

Return codes: *OK, Invalid*

See also: GetTxPowerOffset

6.2.43 StreamInit

This command initializes redundant data streaming on a specific port. The *StreamInit* command can only be issued before the stream has been started using the *SendStreamData* command. Currently the Modem supports only one data stream. After the stream has been initialized data records can be submitted using the *SendStreamData* command to send them as optionally-encrypted redundant fragments to the specified port. The port and mode parameters are stored persistently and are still valid after a device reset.

Command Payload Format

Field	Size (bytes)	Description
port	1	Data streaming port
mode	1	Encryption mode 0x00 or 0x01

The *port* parameter specifies the port the streaming protocol operates on. If this value is zero then the streaming protocol is multiplexed on the DM port. No extra port needs to be allocated in this case. Ports from 224 to 255 must not be used. They are reserved for future standardized application extensions.

The *mode* parameter specifies the encryption mode of the data stream. The stream will be encrypted if *mode* is `0x01`, or plain if *mode* is `0x00`. If the data stream is encrypted the decoding engine can only reassemble the fragments. It is not able to see the data of the records. In this case, the reassembled records need to be decrypted as described in Format of Stream Application Data Record Fragments.

Return codes: *OK, Invalid, Busy*

See also: SendStreamData, StreamStatus, GetDmPort

6.2.44 StreamStatus

This command queries the status of the data-streaming buffer on the specified port. It returns two unsigned 16-bit integer values indicating the number of bytes pending for transmission and the number of bytes still free in the buffer.

Command Payload Format

Field	Size (bytes)	Description
port	1	Data streaming port

Response Payload Format

Field	Size (bytes)	Description
pending	2	Number of bytes pending for transmission
free	2	Number of bytes free in buffer

6.2.45 SuspendModemComm

This command temporarily suspends or resumes the Modem's radio operations. It can be used to prevent extra power consumption by the Modem in case the Application MCU temporarily needs more power itself and to prevent the Modem from exceeding power consumption limits. Operations are suspended with parameter value `0x01` and resumed with parameter value `0x00`.

Command Payload Format

Field	Size (bytes)	Description
suspend	1	Suspend=0x01; Resume=0x00

Return codes: *OK, Invalid*

6.2.46 Test

This command is used to implement test functionality for regulatory conformance, certification, and functional testing. All available tests are implemented as subcommands to the *Test* command and are identified by the *subcmd* parameter.

With the exception of the *Test(TST_START)* subcommand, test subcommands are only available if *test* mode is active. *Test* mode can only be activated if the Modem has not yet received a command that results in radio operation. Once *test* mode is active, all other Modem commands are disabled.

The subcommands with parameters are described in the following subsections. The spreading factor, bandwidth, and coding-rate parameters for the subcommands are encoded as follows.

Table 13: Encoding for SF, BW, and CR

Value	0	1	2	3	4	5	6
Spreading Factor	FSK	SF7	SF8	SF9	SF10	SF11	SF12
Bandwidth	125kHz	250kHz	500kHz	-	-	-	-
Coding Rate	4/5	4/6	4/7	4/8	-	-	-

Return codes: *OK, Invalid, Busy*

6.2.46.1 TST_START

Start *test* mode. This command enables all other test functions.

Command Payload Format for TST_START subcommand

Field	Size (bytes)	Description
subcmd	1	Value: 00
magic	8	Value: 54 45 53 54 54 45 53 54 (“TESTTEST”)

6.2.46.2 TST_NOP

No operation. This command may be used to terminate an ongoing continuous TX operation.

Command Payload Format for TST_NOP subcommand

Field	Size (bytes)	Description
subcmd	1	Value: 01

6.2.46.3 TST_TX_SINGLE

Transmit a single packet.

Command Payload Format for TST_TX_SINGLE subcommand

Field	Size (bytes)	Description
subcmd	1	Value: 02
freq	4	Frequency in Hz
pow	1	Transmission power in dBm
sf	1	Spreading Factor (see Table 13)
bw	1	Bandwidth (see factor (see Table 13)
cr	1	Coding Rate (see factor (see Table 13)
len	1	Payload length

6.2.46.4 TST_TX_CONT

Continuously transmit packets as fast as possible.

Command Payload Format for TST_TX_CONT subcommand

Field	Size (bytes)	Description
subcmd	1	Value: 03
freq	4	Frequency in Hz
pow	1	Transmission power in dBm
sf	1	Spreading Factor (see Table 13)
bw	1	Bandwidth (see Table 13)
cr	1	Coding Rate (see Table 13)
len	1	Payload length

6.2.46.5 TST_TX_HOP

Continuously transmit packets as fast as possible, while respecting regional regulatory constraints (channel hopping, output power, dwell time, duty cycle).

Command Payload Format for TST_TX_HOP subcommand

Field	Size (bytes)	Description
subcmd	1	Value: 04
region	1	Region code (see Table 9)
dr	1	Data Rate
pow	1	Transmission power in dBm
len	1	Payload length

6.2.46.6 TST_TX_CW

Transmit a continuous wave.

Command Payload Format for TST_TX_CW subcommand

Field	Size (bytes)	Description
subcmd	1	Value: 06
freq	4	Frequency in Hz
pow	1	Transmission power in dBm

6.2.46.7 TST_RX_CONT

Continuously receive packets.

Command Payload Format for TST_RX_CONT subcommand

Field	Size (bytes)	Description
subcmd	1	Value: 07
freq	4	Frequency in Hz
sf	1	Spreading Factor (see Table 13)
bw	1	Bandwidth (see Table 13)
cr	1	Coding Rate (see Table 13)

6.2.46.8 TST_RSSI

Measure RSSI.

Command Payload Format for TST_RSSI subcommand

Field	Size (bytes)	Description
subcmd	1	Value: 08
freq	4	Frequency in Hz
time	2	Duration in ms
dr	1	Data Rate

The supplied data rate determines the bandwidth.

Response Payload Format for TST_RSSI subcommand

Field	Size (bytes)	Description
rssi	1	RSSI + 64

6.2.46.9 TST_RADIO_RST

Reset the SX1276 radio.

Command Payload Format for TST_RADIO_RST subcommand

Field	Size (bytes)	Description
subcmd	1	Value: 09

6.2.46.10 TST_SPI

Send and receive SPI commands directly to the SX1276 radio.

Command Payload Format for TST_SPI subcommand

Field	Size (bytes)	Description
subcmd	1	Value: 0A
spicmd	variable	SPI command to send
reslen	1	Number of response bytes to fetch

Response Payload Format for TST_SPI subcommand

Field	Size (bytes)	Description
spirsp	*reslen*	SPI response

6.2.46.11 TST_EXIT

Exit *test* mode and reset Modem.

Command Payload Format for TST_EXIT subcommand

Field	Size (bytes)	Description
subcmd	1	Value: 0B

6.2.46.12 TST_BUSY_LOOP

This subcommand causes the Modem to enter an endless loop with interrupts enabled. Eventually, the software watchdog will save a crash log, flash the diagnostics LED (if available) and reset the device.

Note: This command will render the Modem completely unresponsive until it is reset.

Command Payload Format for TST_BUSY_LOOP subcommand

Field	Size (bytes)	Description
subcmd	1	Value 0C

6.2.46.13 TST_PANIC

This subcommand causes an unrecoverable fault condition (panic). The Modem will immediately save a crash log and halt, the diagnostics LEDs (if available) will flash for some time, and then the Modem will reset.

Note: This command will render the Modem completely unresponsive until it is reset.

Command Payload Format for TST_PANIC subcommand

Field	Size (bytes)	Description
subcmd	1	Value 0D

6.2.46.14 TST_WATCHDOG

This subcommand causes the Modem to enter an endless loop with interrupts disabled. Eventually, the hardware watchdog will reset the device. No crash log will be written, diagnostic LED will not be flashing.

Note: This command will render the Modem completely unresponsive until it is reset.

Command Payload Format for TST_WATCHDOG subcommand

Field	Size (bytes)	Description
subcmd	1	Value 0E

6.2.47 UploadData

This command can be used to repeatedly set file data to be uploaded. The file data must be split into parts of no more than 255 bytes each. The submitted parts will be appended to an internal buffer. In total, exactly as many bytes as specified by the *UploadInit* command must be provided. The buffer allocated for file uploads is 8K bytes, but considering the file header the maximum size of file data is 8180 bytes.

Command Payload Format

Field	Size (bytes)	Description
filedata	1-255	File data part

Return codes: *OK, Invalid, NotInit*

See also: UploadInit, UploadStart

6.2.48 UploadInit

This command prepares a fragmented file upload. It specifies the port for the subsequent upload, optional encryption mode, file size, and average frame transmission interval. The port is included as metadata in the file stream and the stream is sent on the device management port so it can be processed by the *Semtech Cloud Service*. When encryption is used, the file data is encrypted using a 128-bit AES key derived from the AppSKey before it is further processed by the upload service. For the derivation of the decryption key see Format of Defragmented Upload Application File Data (Section 9.1.4). Using encryption, full privacy can be ensured even if the file data is reassembled by the Semtech DM service. In this case Semtech has no knowledge of the contents of the file data being uploaded.

This command can also be used to cancel an ongoing file upload by specifying a file size of zero for the port in use.

Command Payload Format

Field	Size (bytes)	Description
port	1	Port
mode	1	Encryption mode (`01`=encrypted, `00`=plain)
size	2	Total size of file data to be uploaded (up to 8180 bytes)
interval	2	Average frame transmission interval in seconds

Return codes: *OK, Invalid, Fail*

See also: UploadData, UploadStart

6.2.49 UploadStart

After all data bytes indicated by the *UploadInit* command have been provided using the *UploadData* command, this command can be issued to actually start the transmission stream. The amount of data provided by the *UploadData* commands must match the size specified by the *UploadInit* command; otherwise the command is rejected with the *BadSize* return code. The *crc* parameter must match the CRC of the supplied data; otherwise the command is rejected with the *BadCrc* return code. If encryption was requested with the *UploadInit* command, the data will be encrypted before the stream is started (see Format of Defragmented Upload Application File Data (Section 9.1.4). When the stream is started, redundant fragments will be continuously sent in the background until a confirmation is received from the server that it has fully decoded the file, or until twice the required number of chunks have been sent and no confirmation is received (see Format of Upload Application File Data Fragments). When the upload stream is stopped, an *UploadDone* event is generated. This event carries a status byte which indicates success or timeout. A running file upload is indicated by the *Upload* flag in the Modem status.

Command Payload Format

Field	Size (bytes)	Description
crc	4	CRC32 over file data

Return codes: *OK, Invalid, Busy, NoSession, NotInit, BadSize, BadCrc*

See also: UploadInit, UploadData, UploadDone

Note: All CRCs used by the Modem are computed as CRC-32 using the polynomial 0x04C11DB7 on the start value 0xFFFFFFFF with reversed bit ordering (LSBF) and inverted output. The results are compliant with ISO 3309, ITU-T V.42, Gzip and PNG. Test vector: CRC32("123456789") = 0xCBF43926.

7. Events

Events are notifications that occur asynchronously to the command-response flow. Pending events are indicated via the EVENT line and can be retrieved via the *GetEvent* command. Multiple events for different event types may be queued and can be retrieved subsequently. However, multiple events for the same event type will overwrite the previous event of that type. For example, if multiple downlinks are received, and a new downlink arrives before the *DownData* event of the previous one has been retrieved, only the new *DownData* event can be retrieved. Therefore, it is good practice to retrieve pending events as soon as the EVENT line is set. Eventually, missed events are indicated in the *count* field of the response of the *GetEvent* command.

7.1 Event Overview

Table 14: Event Codes

Name	Code	Description	Data
Reset	0x00	Modem has been reset	rstcnt[2]
Alarm	0x01	Alarm timer expired	
Joined	0x02	Network successfully joined	
TxDone	0x03	Frame transmitted	status[1]
DownData	0x04	Downlink data received	port[1], data[n]
UploadDone	0x05	Fileupload completed	status[1]
SetConf	0x06	Config has been changed by DM	inftag[1]
Mute	0x07	Modem has been muted or unmuted by DM	mute[1]
StreamDone	0x08	Data stream fragments sent	
LinkStatus	0x09	Network connectivity status changed	status[1]
JoinFail	0x0A	Attempt to join network failed	

7.2 Event Details

7.2.1 Reset

This event indicates that the Modem has been reset. It returns the current reset counter which is incremented at every reset. A reset can be caused by the UART command, DM request, power failure, or an internal error condition. The Application MCU might use this indication to eventually restore some state of the Modem (e.g. re-join).

Event Payload Format

Field	Size (bytes)	Description
rstcnt	2	Modem reset counter

7.2.2 Alarm

This event indicates that the programmed alarm timer has expired.

7.2.3 Joined

This event indicates that the Modem has successfully joined the network.

7.2.4 TxDone

This event indicates that the previously-initiated *RequestTx* operation has completed. It returns a status byte which indicates whether the frame was sent and acknowledged, the frame was sent, or the frame could not be sent because its length exceeded the maximum payload size for the current data rate.

Event Payload Format

Field	Size (bytes)	Description
status	1	Tx status (frame sent and acknowledged=`02`, frame sent=`01`, frame not sent=`00`)

7.2.5 DownData

This event indicates that a downlink with application data has been received. It returns the RSSI, SNR and RX flags of that downlink, followed by the port and the payload of that frame.

Event Payload Format

Field	Size (bytes)	Description
rssi	1	Signal strength RSSI + 64 in dBm
snr	1	Signal quality SNR in 0.25 dB
flags	1	Reception flags (see Table 15)
port	1	Downlink port
data	1-242	Downlink payload

Table 15: Encoding of RX flags

Flag	Value	Description
ACK	0x80	Confirmed UP frame was acknowledged
NAK	0x40	Confirmed UP frame was not acknowledged
DNW1	0x01	Received in first DN slot
DNW2	0x02	Received in second DN slot
PING	0x04	Received in scheduled RX slot

7.2.6 UploadDone

This event indicates the end of a previously-initiated file upload. It returns a status byte which indicates whether the file has been successfully received by the server or the upload stream has been terminated because no confirmation has been received from the server.

Event Payload Format

Field	Size (bytes)	Description
status	1	File upload status (confirmed=01, timeout=00)

7.2.7 SetConf

This event indicates that a configuration setting has been changed by the DM service. It returns the tag code of the information field changed. Tag codes are described in section 9.1, Uplink Message Format.

Event Payload Format

Field	Size (bytes)	Description
inftag	1	Tag of updated information field

7.2.8 Mute

This event indicates that the Modem has been muted or unmuted by the DM service. It returns a one-byte mute status.

Event Payload Format

Field	Size (bytes)	Description
status	1	Mute status (muted=01, unmuted=00)

7.2.9 StreamDone

This event indicates that the Modem has sent all fragments of the previously-submitted data record and is now ready to accept new data records.

7.2.10 LinkStatus

This event indicates a change in network connectivity.

Event Payload Format

Field	Size (bytes)	Description
status	1	Link status (see Table 16)

Table 16 Link Status Types

Status	Value	Description
Connection lost	`00`	Periodic connectivity checking is implemented using the LoRaWAN *ADRAckReq* bit. If network-managed ADR is active, the event is triggered after reaching the end of the ADR back-off sequence (i.e. the lowest data rate has been selected and all default channels are enabled). If a custom ADR profile is active, the event is triggered as soon as the standard back-off algorithm would have caused a change in data rate.
Connection restored	`01`	Once connection has been lost, this event is triggered if a downlink is received in a class A receive window.

These are purely informational events; the device will continue to function normally, as offline conditions may be transient and the device could move back into coverage.

7.2.11 JoinFail

This event indicates that the attempt to join the network has not yet succeeded. It is generated after trying all data rates on all channels without success. However, the join process is not terminated. After a back-off time, the modem will continue trying, keeping the status *Joining*. The *JoinFail* event can be generated repeatedly and the join process continues until the network is joined or the join process is terminated by the *LeaveNetwork* command.

8. Command Sequence Examples

The following subsections illustrate the command flow and message exchange between the Application MCU, the Modem, and the Network in different scenarios.

8.1 Class A Uplink

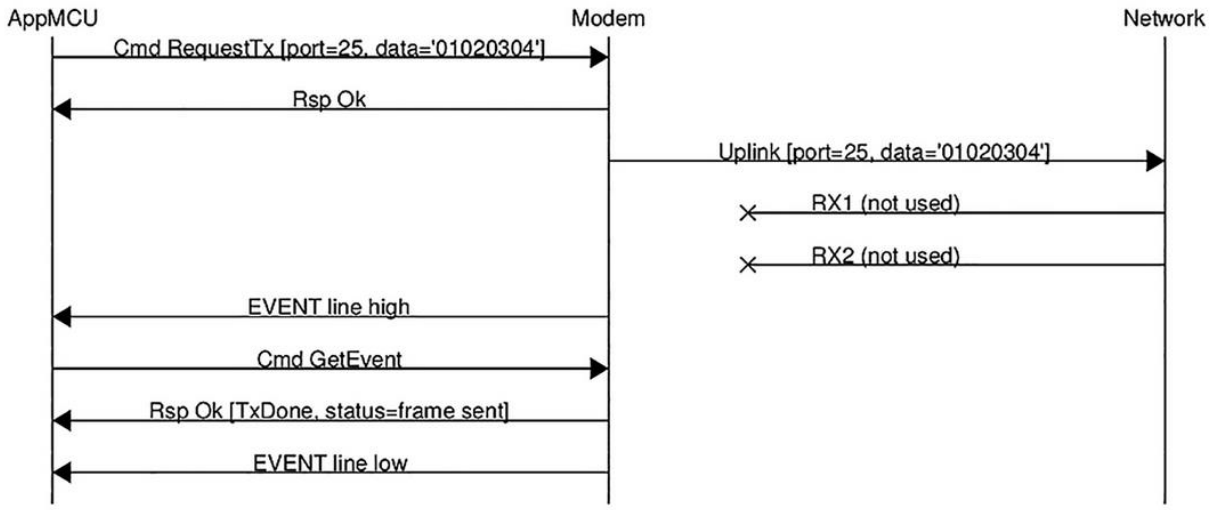

Figure 3: Message Sequence for Uplink Without Downlink

8.2 Class A Uplink with Downlink

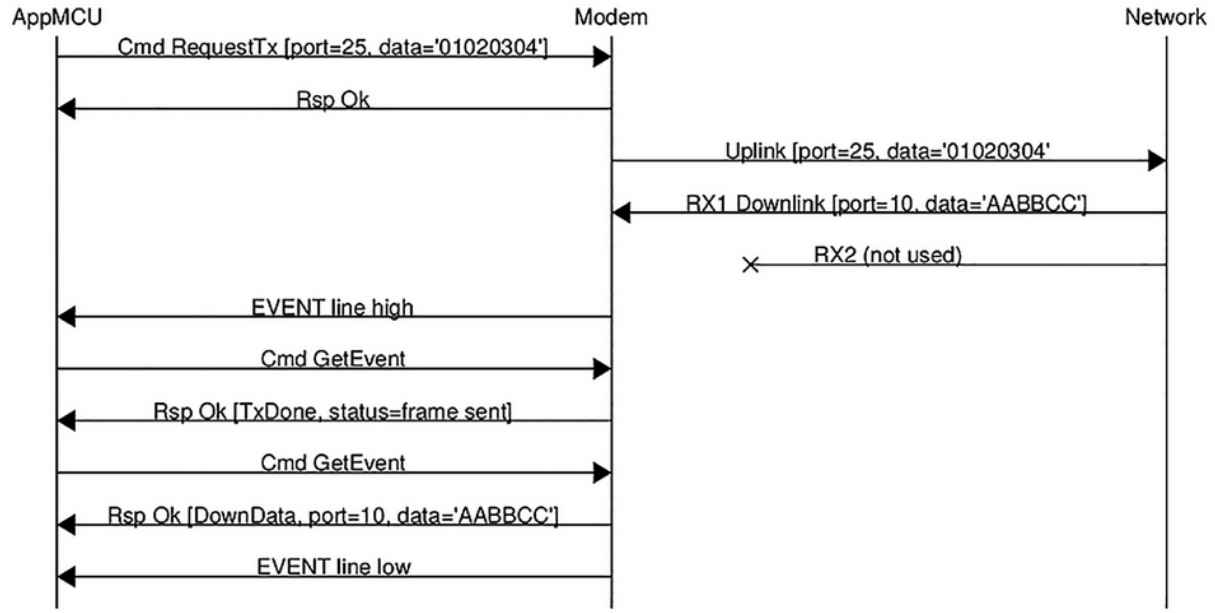

Figure 4: Message Sequence for Uplink With Downlink

8.3 Join

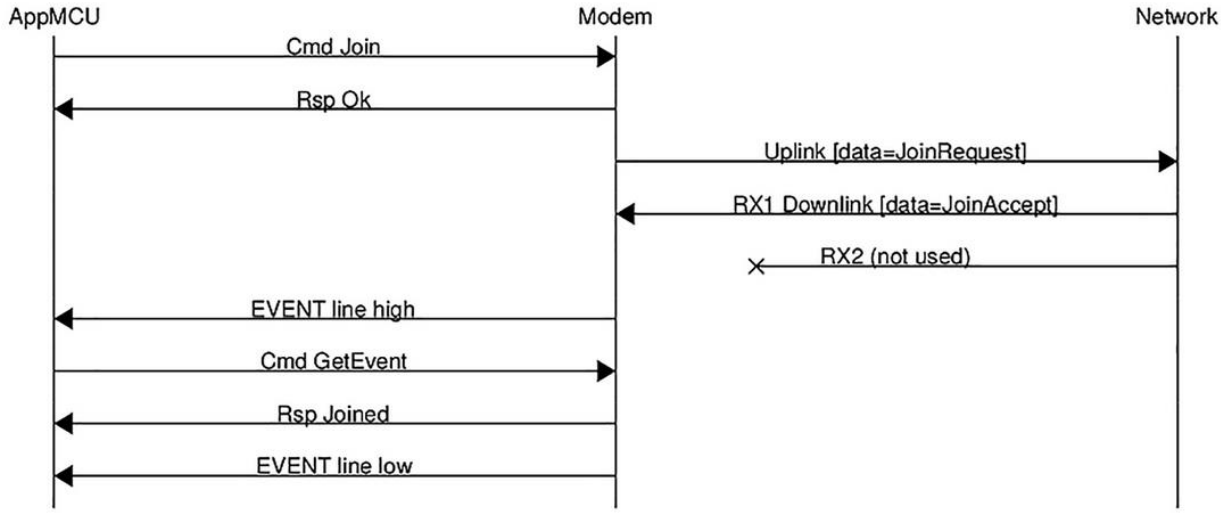

Figure 5: Message Sequence for Join Request And Response

9. Modem Service

The *Semtech Device Management Cloud Service* has an HTTP-based API and is fed with the uplink service messages sent by the Modem. In return to each call it can provide downlink messages to be delivered back to the Modem. It also can return defragmented application data records and defragmented application file data. All uplink messages on the device management port should be forwarded to the cloud service.

9.1 Uplink Message Format

Uplink messages contain one or more concatenated device information fields. All fields begin with a one-byte field tag code and are followed by the field information. The content length of most fields is implicit and is defined in the Table 17 below. Fields which are specified as having a variable length usually have larger content and are sent as the last field of the frame. The content length of these fields is the remaining payload length. When, due to duty cycle limitations, not all requested fields fit into one message, the fields are split over multiple messages.

All multi-byte integers contained in message payloads are transmitted least-significant-byte-first (LSBF).

Table 17: Status Information Fields

Name	Code	Size (bytes)	Description
status	0x00	1	Modem status (see Table 10)
charge	0x01	2	Charge counter [mAh]
voltage	0x02	1	Supply voltage [1/50 V]
temp	0x03	1	Junction temperature [deg Celsius]
signal	0x04	1+1	Signal strength of last downlink (RSSI [dBm]+64, SNR [0.25 dB])
uptime	0x05	2	Duration since last reset [h]
rxtime	0x06	2	Duration since last downlink [h]
firmware	0x07	4+2+2	Firmware CRC and FUOTA progress (completed/total)
adrmode	0x08	1	ADR profile type (see Table 11)
joineui	0x09	8	JoinEUI
interval	0x0A	1	Reporting interval (see Format of Reporting Interval, section 9.1.2)
region	0x0B	1	Regulatory region (see Table 9)
crash log	0x0D	variable	Crash log data
upload	0x0E	variable	Application file stream fragments
rstcount	0x0F	2	Modem reset count

Name	Code	Size (bytes)	Description
deveui	0x10	8	DevEUI
session	0x12	2	Session id / join nonce
chipeui	0x13	8	ChipEUI
stream	0x14	variable	Data stream fragments
streampar	0x15	2	Data stream parameters
appstatus	0x16	8	Application-specific status
alcsync	0x17	variable	Application layer clock sync data

9.1.1 Periodic Status Reporting

The Modem periodically sends a status message which contains some of the above fields. The set of fields included can be changed by the *SetDmInfoFields* Modem command or by the *SetDmInfo* DM request. Additionally, the first status message after joining contains the *rstcount*, *session* and *firmware* fields. Other fields might be explicitly requested using the *GetInfo* downlink message. The reporting interval can be queried and set using the *GetDmInfoInterval* and *SetDMInfoInterval* Modem commands. It can also be set via the *SetConf* DM request specifying a value for the *interval* field.

Note: The *upload*, *stream* and *alcsync* fields are not part of periodic status messages and are sent as separate messages by the respective protocols. Therefore these fields cannot be requested by the *SendDmStatus* or *SetDmInfoFields* commands, nor by the *GetInfo* downlink request.

9.1.2 Format of Reporting Interval

The periodic status reporting *interval* field is encoded in one byte, where the two top-most bits specify the unit (seconds/minutes/hours/days), and the lower six bits the value 0-63. A value of zero disables the periodic status reporting.

Table 18: Encoding of the Reporting Interval

Bits	Variable	Description
7-6	*unit*	sec=00, day=01, hour=10, min=11
5-0	*value*	0-63

9.1.3 Format of Upload Application File Data Fragments

Upload fields are not part of the periodic status reporting but are sent independently by the file upload protocol. The upload fields are of variable length and are sent in a separate message containing only this field. Each upload field begins with a 16-bit header (LSBF) consisting of a 2-bit session id (always 00), a 4-bit session counter (incremental), and a 10-bit file block count (number of 8-byte blocks minus one). This header is followed by one or more encoded 8-byte fragments of spread file data. The fragmentation

algorithm is based on a pseudo-hash function seeded by the LoRaWAN frame counter of the frame transporting the upload field. The exact encoding of the fragments is outside the scope of this document.

9.1.4 Format of Defragmented Upload Application File Data

When the cloud service has received enough fragments to reconstruct the complete file, it will verify the integrity and will return a downlink request to stop the streaming of fragments. This downlink message must be delivered back to the device. The defragmented and reconstructed file consists of a 12-byte header followed by the file data, which is optionally encrypted.

Table 19: Format of Defragmented upload Application File Data

Field	Size (bytes)	Description
port	1	Application port
flags	1	File encryption flags (encrypted=`01`, plain=`00`)
size	2	Size of file data
hash1	4	Truncated SHA256 hash over received file data
hash2	4	Outer or inner hash over received file data (see note)
data	size	File data

Note: If the file data is not encrypted, *hash2* is the next 32-bits of the SHA256 hash over the received file data. When the file data is encrypted, *hash2* is the truncated SHA256 hash of the plain file data. Decryption of the file data can be performed using the AppSKey for AES in *counter* mode with the following 14-byte nonce and initial counter 0001:

AES-CTR nonce

Byte Length	1	4	1	4	4
Field:	`0x49`	`0x00`	`0x40`	file size (LSBF)	hash2

After decryption, *hash2* can be used to verify the integrity of the plain file data.

9.1.5 Format of Stream Application Data Record Fragments

Stream fields are not part of the periodic status reporting but are sent independently by the data streaming protocol. Stream fields are of variable length and are sent in a separate message containing only this field. The stream data consists of one or more fragmented data records. The exact encoding of the fragments is outside the scope of this document

9.1.6 Format of Defragmented Stream Application Data Records

When the cloud service has received enough frames to reconstruct one or more application data records it will return these records retaining the order of transmission. That is, when a missing record is reconstructed from redundancy information, the original order of the returned records is retained. Since

the returned data records may be encrypted, the application might need to decrypt them. Decryption of the record data can be performed using the AppSKey for AES in *counter* mode with the following 14-byte nonce and initial counter 0001:

AES-CTR nonce

Byte Length	1	4	1	4	4
Field:	0x01	0x00	0x42	0x00	stream offset (LSBF)

9.2 Downlink Message Format

The cloud service might return request messages which have to be delivered over the network back to the Modem on the Device Management port. All downlink messages have the following format:

Table 20 Downlink Message Format

Field	Size (bytes)	Description
upcount	1	Uplink count
updelay	1	Uplink delay[s]
reqcode	1	Request code
reqpar	variable	Request parameters

Next to the request code and parameters, each message contains an *upcount* field which indicates the number of uplinks to generate. These uplinks can be used to create additional downlink opportunities and should be generated at the rate specified by the *updelay* field. The reception of a new request message resets the uplink generation.

9.2.1 Overview of Downlink Requests

The following downlink requests are defined:

Table 21: Downlink Requests

Request	Code	Parameters	Description
Reset	0x00	mode[1]+rstcnt[2]	Reset Modem or Application MCU
Fuota	0x01	n	Firmware update fragments (Firmware Updates Over-the-Air)
FileDone	0x02	upload[1]	Signal file upload complete
GetInfo	0x03	inflist[n]	Report specified info fields
SetConf	0x04	tag[1]+value[n]	Set value of a specified field
Rejoin	0x05	sesscnt[2]	Re-join network

Request	Code	Parameters	Description
Mute	0x06	mute[1]	Permanently disable/enable Modem
SetDmInfo	0x07	taglist[n]	Set list of default information fields
Stream	0x08	param[n]	Set data stream parameters
ALCSync	0x09	n	Application layer clock sync data

9.2.2 Downlink Request Details

9.2.2.1 Reset

This request performs a reset of the Modem or Application MCU. The first parameter indicates which units need to be reset (Modem=01, AppMCU=02, both=03). The second parameter is the expected reset count of the Modem. The modem will perform the reset request only if its reset counter matches the expected value. At start-up after reset the modem will generate a *Reset* event.

Request Parameters

Field	Size (bytes)	Description
mode	1	Modem=01, AppMCU=02, both=03
rstcnt	2	Expected reset counter

9.2.2.2 FUOTA

This request delivers a Firmware Update Over-the-Air (FUOTA) firmware fragment to the Modem. In return, the Modem will send an uplink with the *firmware* status field indicating the progress of the update. When the Modem has received enough fragments to reconstruct the full update, it will automatically reset and install the update. The detailed encoding of the FUOTA fragments are outside the scope of this document.

9.2.2.3 UploadDone

This request signals the successful reception of a file upload in progress. It has the session ID and session counter of the completed file upload as a parameter it can use to identify the completed upload session. On reception of this request, the Modem will stop streaming file fragments and will generate an *UploadDone* event.

Request Parameters

Field	Size (bytes)	Description
upload	1	Upload session ID (bits 4-5), Session counter (bits 0-3)

9.2.2.4 GetInfo

This request specifies a list of information tags to be reported by the Modem as soon as possible.

Request Parameters

Field	Size (bytes)	Description
inflist	n	List of status information field tags

9.2.2.5 SetConf

This request allows setting a specific configuration field. Currently the *adrmode*, *joineui* and *interval* fields can be set. The request parameters begin with the tag of the field to be changed, followed by the new value for the field. The length of the value must match the required field size (see Table 17). When the configuration has been updated, a *SetConf* event will be generated.

Request Parameters

Field	Size (bytes)	Description
tag	1	Tag of field to be changed
value	n	New value of field

Note: When the *joineui* field is updated the modem will automatically re-join the network.

9.2.2.6 Rejoin

This request instructs the Modem to re-join the network. The Modem will perform the re-join procedure only if its session ID matches the expected session parameter. Otherwise, it will report its current *session* field.

Request Parameters

Field	Size (bytes)	Description
session	2	Expected session id

9.2.2.7 Mute

This request will mute or unmute the Modem. When muted the Modem is prevented from transmitting any application messages. Therefore, Modem commands which would trigger transmissions will fail.

As a backup mechanism, the Modem is allowed to send status messages very infrequently when muted. This might be used to remotely reconfigure and unmute the Modem. The parameter of the *mute* request specifies this low-frequency mute status interval in days (1-254). The special value 255 will fully mute the Modem and it will never send messages. The special value zero will unmute the Modem and it will then send application and status messages regularly.

A *Mute* event will be generated for the Application MCU indicating the current mute state in the status byte.

Request Parameters

Field	Size (bytes)	Description
mute	1	unmute=00, mute=01-FE [days], kill=FF

9.2.2.8 SetDmInfo

This request specifies the set of information fields to be reported in the periodic status messages. The parameter is a list of information field tags. By default, the *status*, *charge*, *voltage*, *temp*, *signal*, *uptime* and *rxtime* fields are reported.

Request Parameters

Field	Size (bytes)	Description
inflist	n	List of status information fields

9.2.2.9 Stream

This request provides feedback for the data streaming protocol. The format of the request parameter is outside the scope of this document.

9.2.2.10 ALCSync

This request provides time correction feedback for the application-layer clock sync protocol. The format of the request parameter is outside the scope of this document.

10. Mbed Shield

For development purposes, the Modem is available on the SEMTECH Mbed Shield. This section refers to Revision 3 of the CMWX1ZZABZ-104 shield (PCB_E545V03A).

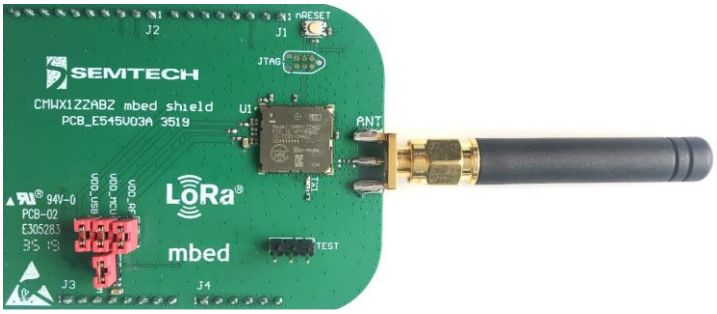

Figure 6: CMWX1ZZABZ-104 mbed shield

10.1 Power Supply

The CMWX1ZZABZ-104 Mbed shield must be powered with 3.3V via the following pins:

Table 22 Power Supply

Function	Shield Header/Pin
VDD	J3/4
GND	J3/6 or J2/7

10.2 Serial interface

The serial interface of the Modem shield is accessible through the following pins.

Table 23 Serial Interface

Modem Function	STM32 GPIO	Shield Header/Pin
UART RX	PA10	J2/1
UART TX	PA9	J1/3
COMMAND	PB8	J2/4
BUSY	PA8	J2/5
EVENT	PB9	J2/3
MCURESET	PB2	J2/2

10.3 Debug Interface

The Modem provides diagnostic output on pin 1 of the TEST header with 2000000bps / 8N1 parameters. To view the output, this pin must be connected to the RX line of a serial port or of a serial-to-USB converter, and displayed with a terminal application.

10.4 JTAG/SWD Programming

The firmware of the Modem shield can be programmed via the 6-pin JTAG header using a JTAG debugger/programmer and a Tag-Connect™ adapter cable (TC2030-MCP-NL)[1].

Table 24 JTAG/SWD Programming

Function	JTAG Pin
VCC	1
SWDIO	2
nRESET	3
SWCLK	4
GND	5

[1] Tag-Connect is the registered trademark of Tag-Connect. For more information visit www.tag-connect.com

Important Notice

Information relating to this product and the application or design described herein is believed to be reliable, however such information is provided as a guide only and Semtech assumes no liability for any errors in this document, or for the application or design described herein. Semtech reserves the right to make changes to the product or this document at any time without notice. Buyers should obtain the latest relevant information before placing order and should verify that such information is current and complete. Semtech warrants performance of its products to the specifications applicable at the time of sale, and all sales are made in accordance with Semtech's standard terms and conditions of sale.

SEMTECH PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS, OR IN NUCLEAR APPLICATIONS IN WHICH THE FAILURE COULD BE REASONABLY EXPECTED TO RESULT IN PERSONAL INJURY, LOSS OF LIFE OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. INCLUSION OF SEMTECH PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE UNDERTAKEN SOLELY AT THE CUSTOMER'S OWN RISK. Should a customer purchase or use Semtech products for any such unauthorized application, the consumer shall indemnify and hold Semtech and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages and attorney fees which could arise.

The Semtech name and logo are registered trademarks of the Semtech Corporation. All other trademarks and trade names mentioned may be marks and names of Semtech or their respective companies. Semtech reserves the right to make changes to, or discontinue any products described in this document without further notice. Semtech makes no warranty, representation guarantee, express or implied, regarding the suitability of its products for any particular purpose. All rights reserved.

©Semtech 2020

Contact Information

Semtech Corporation

200 Flynn Road, Camarillo, CA 93012

Phone: (805) 498-2111, Fax: (805) 498-3804

www.semtech.com