SEMTECH

LoRa Basics™ Modem

WIRELESS & SENSING PRODUCTS

Command Reference Manual

Table of Contents

List Of Figures

List Of Tables

1. Introduction

The LoRa Basics™ Modem (also referred to as modem) software provides an API for end-node operations.

It includes LoRaWAN MAC layer functions and some application layer functions used by the Device and Application Services in the Semtech LoRa® Cloud Service (https://www.loracloud.com/portal/).

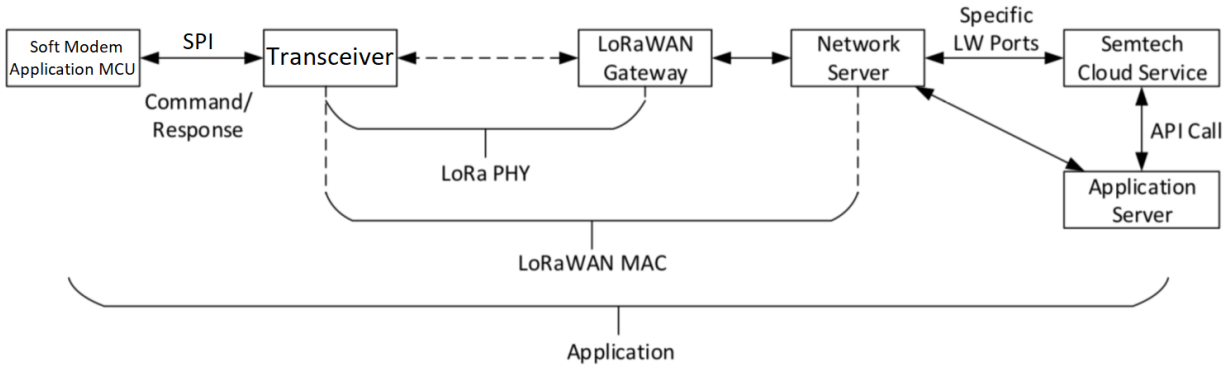

Figure 1: Scope of LoRa Basics™ Modem Services

2. LoRa Basics™ Modem Interface

Communication between the host MCU running the LoRa Basics™ Modem and the transceiver is performed using an SPI bus and four additional signalling lines.

2.1 Serial Port

The SPI of the modem sends commands to the transceiver.

- SCK: (MCU to Transceiver)
- MOSI: Transmit (MCU to Transceiver)
- MISO: Receive (Transceiver to MCU)
- NSS: Chip select (MCU to Transceiver)

The following SPI parameters are used:

Table 1: SPI Parameters

Frequency	Up to 18MHz
Data Size (Bits)	8
Clock Polarity	0
Clock Phase	0

2.2 GPIO Signaling Lines

2.2.1 BUSY Line

Active-high, GPIO output line
This line signals whether the transceiver is busy or ready to receive commands from the modem. It is high while the transceiver is busy and goes low when the transceiver is ready to receive commands.

2.2.2 DIOx Line

Active-high, GPIO output line
This line signals to the modem that the transceiver has asynchronous event data pending. The Application MCU can use the *GetEvent* command to retrieve such data.

2.2.3 Transceiver RESET line

Active-low, GPIO output and floating line
This line signals a request to reset the transceiver and is pulled LOW by the modem for 5 ms.

3. Message Flow

Messages are exchanged via the SPI using BUSY and DIOx GPIO lines for synchronization. This section describes the timing for message flow between the modem and the transceiver.

The NSS line is set by the modem, the BUSY and DIOx lines are set by the transceiver.

All communication consists strictly of modem-initiated command-response sequences.

3.1 SPI transaction

Flow for sending commands to the transceiver:

a. Sending commands are initiated by the NSS line: the modem pulls the NSS line low.

b. The modem drives the BUSY line high.

c. The modem sends the command using the SPI MOSI line.

d. The modem must drive the NSS line high after the last byte of the command has been transmitted. The transceiver only starts interpreting the command when the NSS line is high.

e. The modem drives the BUSY line low.

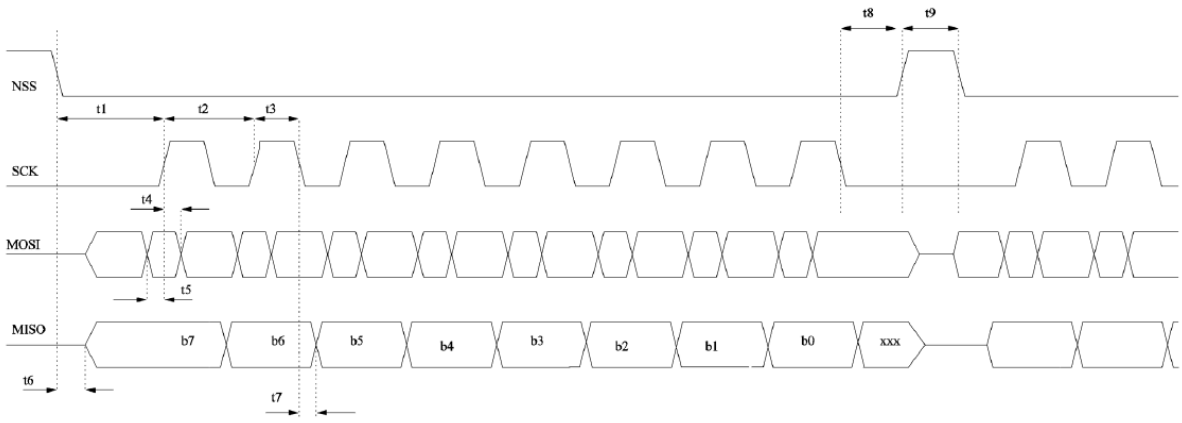

Figure 2: SPI Transaction

Notes:

- The Application MCU must not send a new command before a response has been fully received.
- For timing definitions, refer to the transceiver datasheet.

3.2 Message Flow for Asynchronous Events

When the modem has an asynchronous event pending, it calls a user callback (defined at Init). In this callback it is recommended to check the event status using the *getevent* feature.

4. Return codes

The modem's return code byte is defined as follows.

Table 2: 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
0x0F	FrameError	Serial port framing error

5. Commands and Responses

5.1 Overview of Commands and Responses

Serial communication with the modem uses 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 3: List of Commands and Responses

Name	Description	Input	Output
Init	Initialize the modem	Event callback	
GetEvent	Retrieve pending events		type[1], count[1], eventdata[n] eventdatalen[1] asynchronous_msg number[1]
GetVersion	Get version		bootversion[4], firmware version[4], lorawan[2]
Reset	Reset modem		
FactoryReset	Perform modem factory reset		
ResetCharge	Reset charge counter		
GetCharge	Get accumulated charge		charge[4]
GetTxPowerOffset	Get power correction offset		offset[1]
SetTxPowerOffset	Set power correction offset	offset[1]	
Test	Radio Test function	(see description)	(see description)
GetStatus	Get modem status		status[1]
SetAlarmTimer	Set alarm / wakeup timer	seconds[4]	
GetJoinEui	Get join EUI		JoinEUI[8]
SetJoinEui	Set join EUI, derive keys	JoinEUI[8]	
GetDevEui	Get device EUI		DeviceEUI[8]
SetDevEui	Set device EUI, derive keys	DeviceEUI[8]	
SetNwkKey	Set network key	NwkKey[16]	
GetClass	Get the LoRaWAN device Class		class[1]
SetClass	Set LoRaWan class A/C	class[1]	

Name	Description	Input	Output
GetRegion	Get regulatory region		region[1]
SetRegion	Set regulatory region	region[1]	
ListRegions	List supported regulatory regions		region[1-5]
GetAdrProfile	Get ADR profile		type[1]
SetAdrProfile	Set ADR profile and optional parameters	type[1]+list[16]	
GetDmPort	Get device management (DM) port		port[1]
SetDmPort	Set DM port	port[1]	
GetDmInfoInterval	Get DM reporting interval		interval[1]
SetDmInfoInterval	Set DM reporting interval	interval[1]	
GetDmInfoFields	Get default info fields for DM status		inflist[n]
SetDmInfoFields	Set default info fields for DM status	inflist[n]	
SendDmStatus	Send DM status now	inflist[n]	
SetAppStatus	Set application-specific status for DM	appstatus[8]	
Join	Start joining the network		
LeaveNetwork	Leave the network		
SuspendModemComm	Suspend/resume radio operations	suspend[1]	
GetNextTxMaxPayload	Get max payload size for next Tx		size[1]
RequestTx	Transmit frame unconfirmed or confirmed	port[1], conf[1], data[n], len[1]	
EmergencyTx	Transmit frame immediately (smoke alarm)	port[1], conf[1], data[n], len[1]	
UploadInit	Set file upload port, encryption mode, size, delay	port[1], enc[1], sz[2], delay[2]	
UploadStart	Write data for file upload and start file upload	data[n], len[2]	

5.2 Command Details

All commands have parameters to control the requested operation. All responses have a return code indicating success or failure and may also have data provided by the command. Detailed parameters, return codes and return data are described for every command in the following subsections. Please note that the return codes listed in Table 4 are common to all commands and are not re-explained. If not mentioned otherwise, the commands do not have parameters or return data.

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

Table 4: 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
FrameError	Serial port framing error

5.2.1 EmergencyTx

This command has a higher priority than all other services. It immediately sends the given data on the specified port. It does not take duty cycle or payload size restrictions into account and can be used to signal an alarm condition (like smoke alarm) in real-time, but should be used with caution! A *TxDone* event is generated when the frame has been sent to indicate whether transmission was successful or not.

Note: The application must not use ports 0, 224 (LoRaWAN test port) and 225 to 255 (reserved).

Table 5: Command Payload Format

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

Return codes: *OK, Invalid, Busy*

See also: RequestTx, Join, GetNextTxMaxPayload

5.2.2 FactoryReset

This command performs a factory reset. In addition to the MCU reset, all persistent settings are reset back to their factory state.

Table 6: Factory Reset values

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	0x06 (WW2G4)
Board Tx power offset	0

Return codes: *OK, Invalid*

See also: Reset

5.2.3 GetAdrProfile

This command returns the adaptive data rate (ADR) profile type.

Response Payload Format

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

Return codes: *OK, Invalid*

See also: SetAdrProfile

5.2.4 GetCharge

This command returns the total charge counter of the modem in mAh. This value is 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

5.2.5 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

5.2.6 GetDevEui

This command returns the Device EUI.

Response Payload Format

Field	Size (bytes)	Description
deveui	8	Device EUI

Return codes: *OK, Invalid*

See also: SetDevEui, GetChipEui, GetJoinEui, SetJoinEui

5.2.7 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

5.2.8 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

5.2.9 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

5.2.10 GetEvent

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

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 is returned, and the count field indicates how many events of this type have been missed.

Response Payload Format

Field	Size (bytes)	Description
type	1	Event type, (see Table 12)
count	1	Number of missed events of this type in case of overrun
data	0-253	Event-specific data

Return codes: *OK, Invalid*

See also: Events

5.2.11 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

5.2.12 GetNextTxMaxPayload

This command returns the maximum application payload size, 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*

See also: RequestTx

5.2.13 GetRegion

This command returns the regulatory region code.

Response Payload Format

Field	Size (bytes)	Description
region	1	Region code

Table 7: Region Codes

Region	Code
WW2G4	0x06

Return codes: *OK, Invalid*

See also: SetRegion, ListRegions

5.2.14 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 8: Status Bits

Bit	Value	Name	Description
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

Return codes: *OK, Invalid*

5.2.15 GetTxPowerOffset

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

Response Payload Format

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

Return codes: *OK, Invalid*

See also: SetTxPowerOffset

5.2.16 GetVersion

This command returns the versions of the bootloader, the installed firmware, and 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 (always 0xFFFFFFFF as it is not supported by this modem)
fwversion	4	Modem firmware version
lwversion	2	LoRaWAN version (BCD)

Return codes: *OK, Invalid*

5.2.17 Join

This command joins or re-joins the network. During this 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 attempts to join the network indefinitely while adhering to the join duty cycle mandated by the LoRaWAN specification. To abort an ongoing join attempt, use the *LeaveNetwork* command. Whenever a cycle of trying all data rates has failed a *JoinFail* event is generated, but 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 network is joined and the session is already established.

Return codes: *OK, Invalid, Busy*
See also: LeaveNetwork, RequestTx

5.2.18 LeaveNetwork

This command leaves 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

5.2.19 ListRegions

This command returns the regulatory region codes supported by the modem (WW2G4).

Response Payload Format

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

Return codes: *OK, Invalid*

See also: GetRegion, SetRegion

5.2.20 RequestTx

This command requests to send the given data on the specified port as an unconfirmed or confirmed frame. The request is queued and the frame is sent as soon as the current bandwidth usage of the regulatory region permits. A *TxDone* event is generated when the frame has been sent, and its parameter indicates whether it was sent successfuly or not.

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 a previous transmission's *TxDone* has been 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.

Command Payload Format

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

Note: The application must not use ports 0, 224 (LoRaWAN test port) and 225 to 255 (reserved).

5.2.21 Reset

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

Return codes: *OK, Invalid*
See also: FactoryReset

5.2.22 ResetCharge

This command resets the accumulated charge counter to zero.

Return codes: *OK, Invalid*

See also: GetCharge

5.2.23 SendDmStatus

This command immediately sends the specified set of information fields in one or more DM status messages.

The set is specified as a list of field codes as defined in Uplink Message Format. Duplicate and invalid fields are 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*

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

5.2.24 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 9: ADR Profile Types and Parameters

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 are predefined and do not need parameters. They 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

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.

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

5.2.25 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 started again with the new period. A value of 0 cancels any pending alarm timer.

Command Payload Format

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

Return codes: *OK, Invalid*

5.2.26 SetAppStatus

This command sets application-specific status information that is sent as part of the periodic status reports to the DM service. It is an application-defined, arbitrary 8-byte data blob.

Note that this command does not trigger an immediate status report. If the application desires 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 is reported.

Command Payload Format

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

Return codes: *OK, Invalid*

See also: SendDmStatus, SetDmInfoInterval, Uplink Message Format

5.2.27 SetClass

This command sets the LoRaWAN device class.

Command Payload Format

File	Size (bytes)	Description
class	1	Class value

Table 10: Supported LoRaWAN Classes

LoRaWAN Class	Value	Description
A	0x00	Open short RX windows after TX

Return codes: *OK, Invalid*

See also: SetClass

5.2.28 SetDevEui

This command sets the Device EUI. The command can only be issued if the modem has not yet joined or is still trying to join the network, otherwise the command will fail with *Busy* return code.

Command Payload Format

Field	Size (bytes)	Description
deveui	8	Device EUI

Return codes: *OK, Invalid, Busy*

See also: GetDevEui, GetJoinEui, SetJoinEui

5.2.29 SetDmInfoFields

This command specifies the set of info 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. 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

5.2.30 SetDmInfoInterval

This command sets the device management reporting interval (in seconds, minutes, hours or days). The value zero (seconds, minutes, hours or days) disables status reporting.

Command Payload Format

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

Return codes: *OK, Invalid*

See also: GetDmInfoInterval, Format of Reporting Interval

5.2.31 SetDmPort

This command sets the device management port. Port 0, 224 and 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

5.2.32 SetJoinEui

This command sets the join EUI. It can only be issued if the modem has not yet joined or is still trying to join the network, otherwise the command will fail with *Busy* return code.

Command Payload Format

Field	Size (bytes)	Description
joineui	8	Join EUI

Return codes: *OK, Invalid, Busy*

See also: GetJoinEui, GetDevEui, SetDevEui

5.2.33 SetNwkKey

This command sets the LoRaWAN 1.0.3 Network key. It can only be issued if the modem has not yet joined or is still trying to join the network, otherwise the command will fail with *Busy* return code.

Command Payload Format

Field	Size (bytes)	Description
NwkKey	16	Network key

Return codes: *OK, Invalid, Busy*

See also: SetDevEui, SetJoinEui

5.2.34 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

5.2.35 SetTxPowerOffset

This command sets the board-specific offset correction for transmission power to be used. The offset depends on board design and antenna matching and is expressed in dB (signed integer). The offset value is subtracted from the programmed output power allowing direct programming of the transmit power in dBm by the modem user.

Command Payload Format

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

Return codes: *OK, Invalid*

See also: GetTxPowerOffset

5.2.36 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 doesn't want to exceed limits.

Command Payload Format

Field	Size (bytes)	Description
suspend	1	Suspend=0x01, resume=0x00

Return codes: *OK, Invalid*

5.2.37 Test

This command implements 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 11: Encoding for SF, BW, and CR

Value	0	1	2	3	4	5	6	7	8
SF	FSK	SF7	SF8	SF9	SF10	SF11	SF12	SF5	SF6
BW	125kHz	250kHz	500kHz	203.125kHz	406.25kHz	812.5kHz	1.625MHz		
CR	4/5	4/6	4/7	4/8	4/5 LI	4/6 LI	4/8 LI		

Return codes: *OK, Invalid, Busy*

5.2.37.1 TST_START

Start test mode. Enables all other test functions.

5.2.37.2 TST_NOP

No operation. Terminates an ongoing continuous TX operation.

5.2.37.3 TST_MODE_TX

Transmits a single packet.

Command Payload Format for TST_TX_SINGLE subcommand

Field	Size (bytes)	Description
freq	4	Frequency (Hz)
pow	1	Transmission power (dBm)
sf	1	Spreading factor (see Table 11)
bw	1	Bandwidth (see Table 11)
cr	1	Coding rate (see Table 11)
len	1	Payload length
tx_mode	1	0x00=single, 0x01=continue

5.2.37.4 TST_TX_CW

Transmits a continuous wave.

Command Payload Format for TST_TX_CW subcommand

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

5.2.37.5 TST_RX_CONT

Continuously receives packets.

Command Payload Format for TST_RX_CONT subcommand

Field	Size (bytes)	Description
subcmd	1	Value `07`
freq	4	Frequency (Hz)
sf	1	Spreading factor (see Table 11)
bw	1	Bandwidth (see Table 11)
cr	1	Coding rate (see Table 11)

5.2.37.6 TST_RADIO_RST

Resets the radio.

5.2.37.7 TST_RADIO_WRITE

Sends/receives SPI commands directly to/from the radio.

Command Payload Format for TST_RADIO_WRITE subcommand

Field	Size (bytes)	Description
radio_cmd	1	Radio command
cmd_len	2	Radio command length
data	Variable	Related data to the command
data_len	2	Data length

5.2.37.8 TST_RADIO_READ

Send and receive SPI commands directly to the radio.

Command Payload Format for TST_RADIO_READ subcommand

Field	Size (bytes)	Description
radio_cmd	1	Radio command
cmd_len	2	Radio command length
data	Variable	Output of the requested command
data_len	2	Return data length must be set by the user

5.2.37.9 TST_EXIT

Exits test mode and resets modem.

Command Payload Format for TST_EXIT subcommand

Field	Size (bytes)	Description
subcmd	1	Value 0B

5.2.38 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 which 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 being processed by the upload service. For the derivation of the decryption key see 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 8 Kbytes)
interval	2	Average frame transmission interval in seconds

Return codes: *OK, Invalid, Fail*

See also: UploadData, UploadStart

5.2.39 UploadStart

The file data needs to be split into parts of maximum 255 bytes each and the submitted parts are appended to an internal buffer. The total number of bytes must match the *UploadInit* command's *size* value. The buffer allocated for file uploads is 8K bytes.

After all data bytes indicated by the *UploadInit* command have been provided, this command can be issued to start the transmission stream.
If the total amount of data does not match the size specified by *UploadInit* command, the command is rejected with *BadSize* return code.
If encryption was requested with the *UploadInit* command, the data is encrypted before the stream is started (see Format of Defragmented Upload Application File Data).
When the upload stream starts, redundant fragments are 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 stops, an *UploadDone* event is generated which carries a status byte that 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
data	Variable	Data to upload
size	2	Total size of file data to be uploaded (up to 8 Kbytes)

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

See also: UploadInit, UploadDone

6. 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 might 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 *GetEvent* command response.

6.1 Event Overview

Table 12: 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	File upload 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]`
LinkStatus	0x09	Network connectivity status changed	`status[1]`
JoinFail	0x0A	Attempt to join network failed	

6.2 Event Details

6.2.1 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

6.2.2 Alarm

This event indicates that the programmed alarm timer has expired.

6.2.3 Joined

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

6.2.4 TxDone

This event indicates that the previously initiated *RequestTx* operation has completed.

It returns a status byte which indicates whether the frame was sent or not.

Event Payload Format

Field	Size (bytes)	Description
status	1	Tx status: 0x02 = frame sent and acknowledged 0x01 = sent but not acknowledged 0x00 = not sent, length exceeded maximum payload size for current data rate.

6.2.5 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
RFU	1	
port	1	Downlink port
data	1-242	Downlink payload
timestamp	4	timestamp of the received message

6.2.6 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 if the upload stream has been aborted 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)

6.2.7 SetConf

This event indicates that a configuration setting has been changed by the DM service. Tag codes are described in Section 7.1 Uplink Message Format.

Event Payload Format

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

6.2.8 Mute

This event indicates that the modem has been muted or unmuted by the DM service.

Event Payload Format

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

6.2.9 LinkStatus

This event indicates a change in network connectivity. This is a purely informational event; the device will continue to function normally, as offline conditions may be transient, and the device could move back into coverage.

Event Payload Format

Field	Size (bytes)	Description
status	1	Link status (see Table 13)

Table 13: 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.

6.2.10 JoinFail

This event indicates that the attempt to join the network did not succeed yet. The join process is not aborted and after a back-off time the modem repeats the process, keeping the status *Joining,* trying all data rates on all channels, generating *JoinFail* events, until the network is joined or it is aborted by the *LeaveNetwork* command.

7. 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.

7.1 Uplink Message Format

Uplink messages contain one or more concatenated device information fields. All fields begin with a one byte code, followed by the device information whose length is defined in the table below. Variable length fields are usually larger and are sent as the last field of the frame. The content length of these fields is the remaining payload length. If duty cycle limitations mean that 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 14: Device Information Fields

Name	Code	Size (bytes)	Description
status	0x00	1	Modem status (see Table 13)
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]
adrmode	0x08	1	ADR profile type (see Table 9)
joineui	0x09	8	JoinEUI
interval	0x0A	1	Reporting interval (see section 9.1.2)
region	0x0B	1	Regulatory region (see Table 7)

Name	Code	Size (bytes)	Description
upload	0x0E	Variable	Application file stream fragments
rstcount	0x0F	2	Modem reset count
deveui	0x10	8	DevEUI
session	0x12	2	Session id / join nonce
appstatus	0x16	8	Application-specific status

7.1.1 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. The first status message after joining additionally contains the *rstcount*, *session* and *firmware* fields. Other fields might be explicitly requested using the *GetInfo* downlink request. The reporting interval can be queried and set using the *GetDmInfoInterval* and *SetDMInfoInterval* modem commands. Additionally, it can 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*, *SetDmInfoFields* commands or by the *GetInfo* downlink request.

7.1.2 Reporting Interval Format

The periodic status reporting *interval* field is encoded in one byte. A value of zero disables the periodic status reporting.

Table 15: Encoding of the Reporting Interval

Bits	Variable	Description
7-6	Unit	Sec=00, day=01, hour=10, min=11
5-0	Value	0-63

7.1.3 Upload Application File Data Fragments Format

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:
  - 2-bit session id (always `00`)
  - 4-bit session counter (incremental)
  - 10-bit file block count (number of 8-byte blocks minus one)
- followed by one or more encoded 8-byte fragments of spread file data.

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

7.1.4 Defragmented Upload Application File Data Format

When the cloud service has received enough fragments to reconstruct the complete file, it verifies the integrity and returns a downlink request to stop streaming fragments. This downlink message needs to 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 16: 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	Variable	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:

Table 17: 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.

7.2 Downlink Message Format

The cloud service might return request messages which must be delivered to the modem's device management port via the network. All downlink messages have the following format:

Table 18: Downlink Message Format

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

As well as 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.

7.2.1 Downlink Requests Summary

The following downlink requests are defined:

Table 19: Downlink Requests

Request	Code	Parameters	Description
Reset	0x00	mode[1]+rstcnt[2]	Reset modem or Application MCU
UploadDone	0x02	upload[1]	Signal file upload complete
GetInfo	0x03	inflist[n]	Report specified info fields
SetConf	0x04	tag[1]+value[n]	Set value of specified field
Rejoin	0x05	sesscnt[2]	Re-join network
Mute	0x06	mute[1]	Permanently disable/enable modem
SetDmInfo	0x07	taglist[n]	Set list of default info fields

7.2.2 Downlink Request Details

7.2.2.1 Reset

This request performs a reset of the modem and/or the Application MCU. The modem performs the reset request only if its reset counter matches the expected value. At start-up after a reset the modem generates a *Reset* event.

Table 20: Reset Request Parameters

Field	Size (bytes)	Description
mode	1	The unit to be reset: Modem=01, AppMCU=02, both=03
rstcnt	2	Modem's expected reset counter

7.2.2.2 UploadDone

This request signals the successful reception of a file upload. Its parameters identify the completed upload session. On reception of this request the modem stops streaming file fragments and generates an *UploadDone* event.

Table 21: UploadDone Request Parameters

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

7.2.2.3 GetInfo

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

Table 22: GetInfo Request Parameters

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

7.2.2.4 SetConf

This request sets the *adrmode*, *joineui* or *interval* configuration fields. The length of the *value* must match the required field size (see Table 23). When the configuration has been updated a *SetConf* event is generated. When the *joineui* field is updated the modem automatically re-joins the network.

Table 23: SetConf Request Parameters

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

7.2.2.5 Rejoin

This request instructs the modem to re-join the network. The modem tries to re-join only if its session id matches the expected session ID, otherwise it reports its current *session* field.

Table 24: Rejoin Request Parameters

Field	Size (bytes)	Description
session	2	Expected session ID

7.2.2.6 Mute

This request mutes or unmutes the modem. When muted the modem cannot transmit any application messages, so modem commands which would trigger transmissions will fail. As a backup mechanism the modem could be allowed to send status messages infrequently when muted, so the modem could be unmuted remotely.

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

Table 25: Mute Request Parameters

Field	Size (bytes)	Description
mute	1	00 = Unmute modem so it can send regular application and status messages 01-FE = low-frequency interval of mute status, in days (1-254) FF = kill/mute modem so it never sends messages (255)

7.2.2.7 SetDmInfo

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

Table 26: SetDmInfo Request Parameters

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

8. Mbed Shield

For development purposes, the transceiver is available on the SEMTECH Mbed Shield.

This section refers to Revision 2A of the SX1280RF1ZHP shield (PCB_E394V02A).

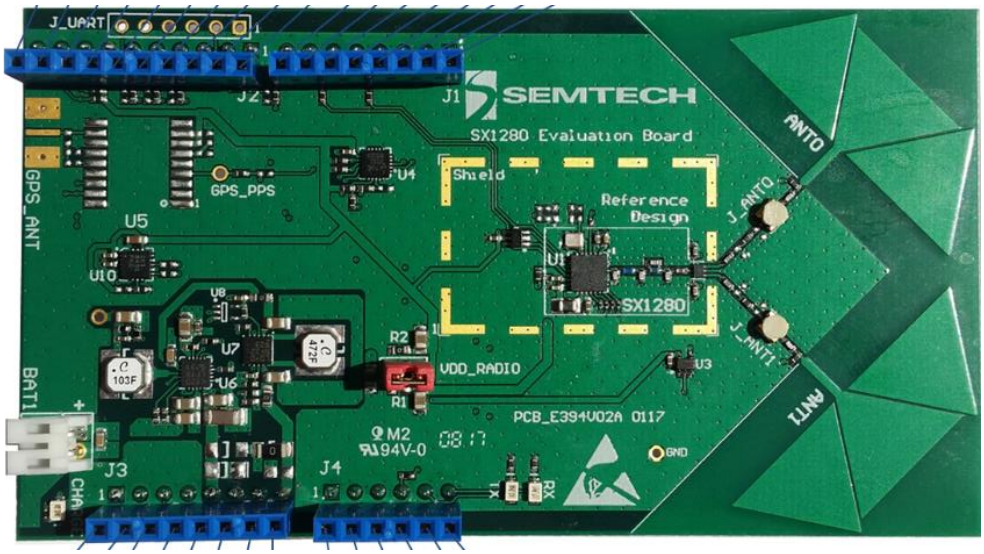

Figure 3: SX1280RF1ZHP Mbed shield

8.1 Power Supply

The SX1280RF1ZHP Mbed shield needs to be powered with 3.3V via the following pins:

Table 27: Power Supply

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

8.2 Serial Interface

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

Table 28: Serial Interface

Function	STM32 GPIO	Shield Header/Pin
SPI MOSI	PA7	J2/4
SPI MISO	PA6	J2/5

Function	STM32 GPIO	Shield Header/Pin
SPI SCLK	PA5	J2/6
NSS	PA8	J1/8
BUSY	PB3	J1/4
DIOx	PB4	J1/6
RESET	PA0	J3/1

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.

Contact Information

Semtech Corporation

200 Flynn Road, Camarillo, CA 93012

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

www.semtech.com