SEMTECH

LoRa Basics Modem-E

Reference Manual

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Scope

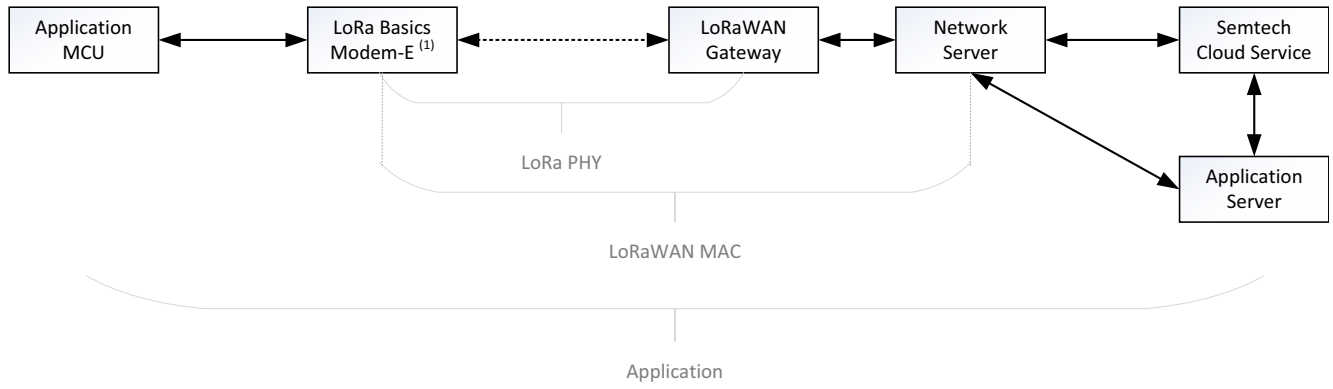
This document provides information on how to use the LoRa Basics Modem-E device in an application. For circuit specifications and details on the LR1110 functions used by LoRa Basics Modem-E please refer to the LR1110 Datasheets.

This document only details the specificities of LoRa Basics Modem-E, which are unavailable to the LR1110 used as an RF transceiver. Common functionalities of the LoRa Basics Modem-E and the LR1110 used as an RF transceiver are described in the LR1110 Transceiver User Manual.

1.2 Overview

LoRa Basics Modem-E is the embedded software programmed inside the LR1110 that supports the LoRaWAN protocol modem stack along with application layer functions and API's to interface with cloud based geolocation and device and application services. It uses the Semtech LoRa Cloud™ Device & Application Services. The LoRa Basics Modem-E and LoRaWAN® complete system is depicted in Figure 1-1: LoRa Basics Modem-E System Overview here below.

Figure 1-1: LoRa Basics Modem-E System Overview



1.) LoRa Basics Modem-E Firmware running on the LR1110 device

1.3 Implemented Features

The following features are embedded in the LoRa Basics Modem-E:

- LoRaWAN® class A protocol version 1.0.3 [1]
- EU868 & US915 Regional parameters
- Device Management Periodic Status Messages
- Remote reset & re-keying
- Temperature monitoring

1. **certified against LoRaWAN® specification v1.0.2**

- Embedded certification mode
- LoRa Cloud™ Large File Upload Service
- LoRa Cloud™ Application-Layer Clock Synchronization (ALCSYNC)
- LoRa Cloud™ Reliable Octet Stream Encoding (ROSE) Streaming service
- Wi-Fi passive scanning
- GNSS passive scanning
- Over The Air (OTA) GNSS Almanac update service

2. Host-Controller Interface

The LoRa Basics Modem-E communicates with the host MCU using a set of API commands and responses implemented over SPI. It uses two additional signals to communicate status with the host: BUSY and EVENT.

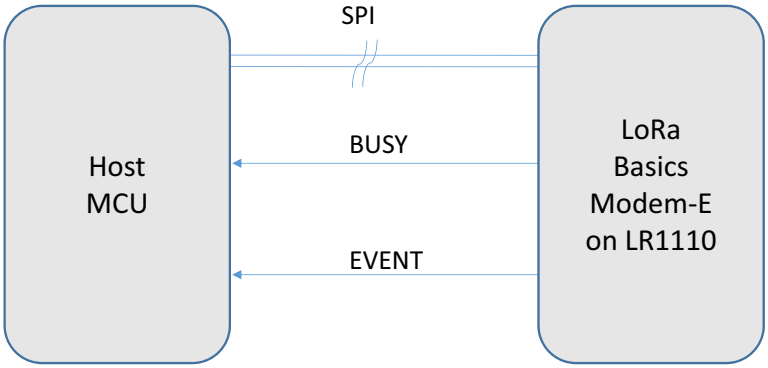


Figure 2-1: LoRa Basics Modem-E Interface

All serial communications consist strictly of host-initiated commands, followed by LoRa Basics Modem-E responses. The device does not send any unsolicited response on the SPI port. If some event data becomes available asynchronously, the device signals this using the EVENT line.

The LoRa Basics Modem-E SPI protocol allows the device to return to sleep mode automatically for lowest power consumption. It also features an automatic CRC protection and parameters verification in the SPI commands, so that the host MCU can read a feedback (a.k.a return code) immediately after the command is sent.

Moreover, the SPI interface is protected against de-synchronization: for example, if incomplete commands are sent by the Host MCU (e.g. missing bytes), the watchdog timer will trigger a device reset after 120 seconds, preventing the device from waiting for command bytes that never arrive.

2.1 SPI Commands

An SPI transaction is initiated by a falling edge of the NSS signal.

Upon generation of an SPI command, the LoRa Basics Modem-E analyses the command message (correct number of bytes and correct CRC), interprets the command, and finally sends a response. If the packet structure is invalid or the CRC wrong, the modem ignores the packet and sends a FrameError response. A response is always sent to any command.

In all SPI transactions, multiple octet fields use the big endian format.

2.1.1 Command Format

The SPI command consists of a 2 Byte Opcode, followed by a Payload of variable length, and then a CRC, as shown in Table 2-1: SPI Command Structure here below.

Table 2-1: SPI Command Structure

Byte Length	1	1	N	1
Field	Id	Cmd	Payload	CRC

- The field Id is defined as indicated in Table 2-2: Command Group Id Definition. The fields ID and Cmd are referred to as the command Opcode in tables Table 10-1 to Table 10-5.

Table 2-2: Command Group Id Definition

Group	Id
System	0x01
Wi-Fi	0x03
GNSS	0x04
Modem	0x06

- The CRC is a crc-8 computed over the whole SPI command packet, using a polynomial generator 0x65 (reversed reciprocal), and a CRC initial value or 0xFF.

2.1.2 Response Format

The SPI response consists of a 1 Byte Return Code, followed by an optional 1 Byte Payload, and then a CRC, as shown in Table 2-3: SPI Response Structure here below:

Table 2-3: SPI Response Structure

Byte Length	1	N	1
Field	Return Code	Payload (Optional)	CRC

The Return Codes (also referred as RC) are defined in Table 2-4: Return Code Definitions.

Table 2-4: Return Code Definitions

Value	Return Code (RC)	Description	Commands
0x00	OK	Command executed without error	System/Wi-Fi/GNSS/Modem
0x01	Unknown	Command code unknown	System/Wi-Fi/GNSS/Modem
0x02	NotImpl	Command not yet implemented	Modem

Table 2-4: Return Code Definitions

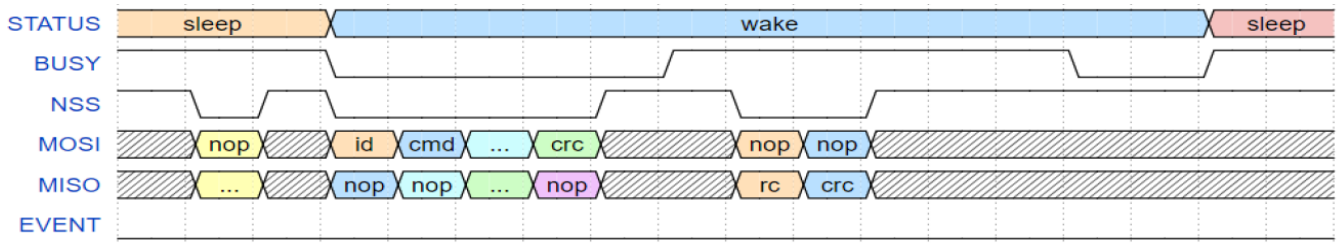
Value	Return Code (RC)	Description	Commands
0x03	NotInit	Command not initialized	Modem
0x04	Invalid	Command parameters invalid	System/Wi-Fi/GNSS/Modem
0x05	Busy	Command cannot be executed now	Modem
0x06	Fail	Command execution failed	System/Wi-Fi/GNSS/Modem
0x07	BadFmt	Format check failed	Modem
0x08	BadFileUploadCrc	File Upload CRC Check failed	Modem
0x09	BadSig	Signature verification failed	Modem
0x0A	BadSize	Size check failed	Modem
0x0F	FrameError	SPI command checksum or CRC error	System/Wi-Fi/GNSS/Modem
0x10	NoTime	GNSS Time sync lost	GNSS

2.1.3 SPI Write Diagram

An SPI Write consists of three SPI transactions. The host MCU must ensure that the BUSY line is high before initiating the SPI transactions. The SPI write transaction is depicted in Figure 2-2: SPI Write Transaction hereafter.

1. Before the host issues the SPI command, the LoRa Basics Modem-E is in sleep mode, with the BUSY signal set to high.
2. The host MCU toggles the NSS line to wake up the LoRa Basics Modem-E .
3. Once BUSY line is low, the host MCU drives the NSS line low and sends the bytes stream.
4. After the bytes stream is sent, the host MCU drives the NSS line high, and waits for the BUSY line to go high.
5. Once the BUSY line is high, the host MCU drives the NSS line to low again and sends two NOPs (0x00 Bytes).
6. After sending the 2 NOPs (0x00 Bytes), the host MCU drives the NSS line high, and waits for the BUSY line to go low.
7. The LoRa Basics Modem-E drives the BUSY line low when it is processing the command.
8. If no more processing is needed, the LoRa Basics Modem-E goes to sleep mode and drives the BUSY line high

Figure 2-2: SPI Write Transaction



NOTE: No Status bits nor IRQ Status bits are returned on the MISO line during an SPI Write transaction.

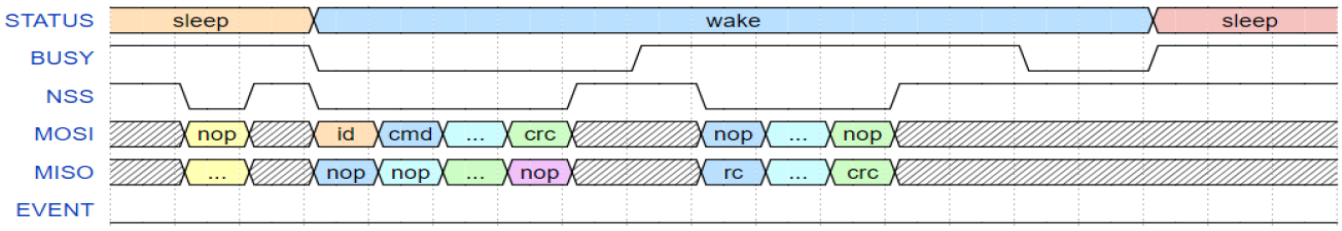
NOTE: the *Reset*(...) command does not result in a second phase of BUSY signal to low (step 7.) in the above description).

2.1.4 SPI Read Diagram

An SPI read consists of three SPI transactions. The host MCU must ensure that the BUSY line is high before initiating the SPI transactions. The SPI read transaction is depicted in Figure 2-3: SPI Read Transaction hereafter.

1. Before the host issues the SPI command, the LoRa Basics Modem-E is in sleep mode, with the BUSY signal set to high.
2. The host MCU toggles the NSS line to wake up the LoRa Basics Modem-E.
3. Once BUSY line is low, the host MCU drives the NSS line low and sends the bytes stream.
4. After the bytes stream is sent, the host MCU drives the NSS line high, and waits for the BUSY line to go high.
5. Once the BUSY line is high, the host MCU drives the NSS line to low again and sends the expected number of NOPs (0x00 Bytes).
6. After sending the expected number of NOPs (0x00 Bytes), the host MCU drives the NSS line high, and waits that the BUSY line goes low.
7. The LoRa Basics Modem-E drives the BUSY line low when it is processing.
8. If no more process is needed, the LoRa Basics Modem-E goes to sleep mode and drives the BUSY line high.

Figure 2-3: SPI Read Transaction



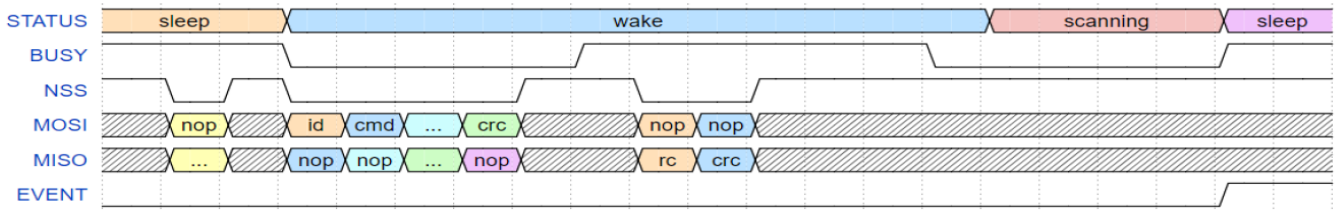
NOTE: No status bits or IRQ status bits are returned on the MISO line during an SPI Read transaction.

2.1.5 Wi-Fi Passive Scan / GNSS Scan SPI Command Diagram

A Wi-Fi Passive Scan or GNSS Scan SPI command consists of three SPI transactions; more specifically, the GNSS/Wi-Fi scan is triggered by the 3rd transaction after two NOPs. If the 3rd transaction does not occur, the scan will not be launched. Before the GNSS/WIFI scan is launched, the LoRa Basics Modem-E masks those IRQs (like NSS) which might impact the radio capture. After the GNSS/WIFI scan is finished, the EVENT line is driven high to signal to the host MCU that scan results are available.

After the Wi-Fi Passive Scan or GNSS Scan, the LoRa Basics Modem-E returns to sleep mode and drives the BUSY line to high.

Figure 2-4: Wi-Fi Passive Scan / GNSS Scan SPI Transaction



NOTE: No status bits or IRQstatus bits are returned on the MISO line during a Wi-Fi Passive Scan / GNSS Scan SPI transaction.

2.2 BUSY Signalling

The BUSY line is an output signal of the LoRa Basics Modem-E. It is mapped on the IO0 pin of the LoRa Basics Modem-E.

The BUSY line is high when the device is in sleep mode, and ready to accept a command. Therefore, it is necessary to check the status of BUSY prior to sending a command.

Note: The BUSY signal has inverted polarity in the Transceiver and Modem implementations.

2.3 EVENT Signalling

2.3.1 Description

The EVENT signal is an output signal of the LoRa Basics Modem-E. It is mapped on the DIO9 pin of the LoRa Basics Modem-E.

This line signals to the host MCU that the device has an asynchronous event data pending. The event can be LoRaWAN® stack, GNSS, or Wi-Fi events. The host MCU must use the *GetEventSize (...)* command to determine the event size and *GetEvent (...)* command to retrieve such data.

The event line stays high until all events are cleared. The host MCU must clear all the events before returning to sleep mode. This prevents the host MCU from missing the rising edge of the EVENT line when there is a new event. Secondly, this saves several tens of uA of current consumption when the EVENT line is kept high by the LoRa Basics Modem-E.

2.3.2 Event Handling Examples

Some SPI commands generate an immediate response by the LoRa Basics Modem-E, and therefore do not generate any event, such as the command *GnssReadAssistancePosition(...)*, as shown in Figure 2-5: GnssAlmanacRead Transaction.

Figure 2-5: GnssAlmanacRead Transaction

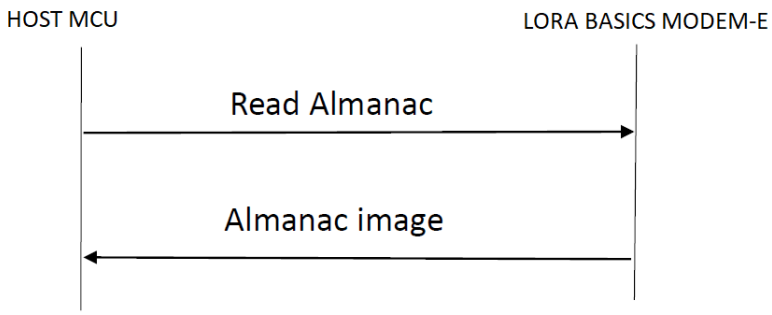


Figure 2-6: Wi-Fi Passive Scan/ GNSS Scan Transaction shows the event sequence in case of a Wi-Fi Passive Scan or GNSS Scan command: the EVENT signal indicates that some asynchronous data is available. The *GetEventSize(...)* command allows to determine the number of Bytes of the event stored in the LoRa Basics Modem-E, and *GetEvent(...)* allows to read back this event.

Figure 2-6: Wi-Fi Passive Scan/ GNSS Scan Transaction

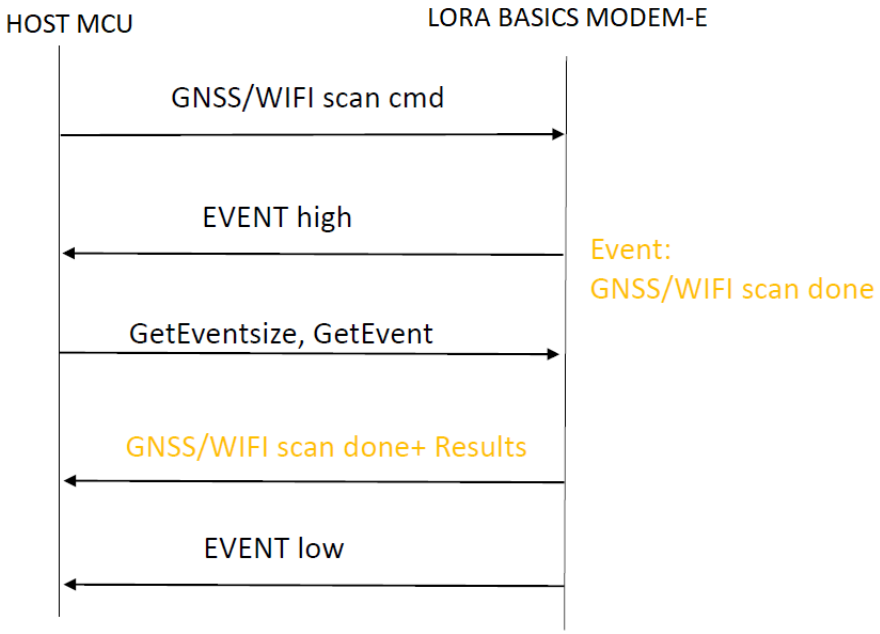
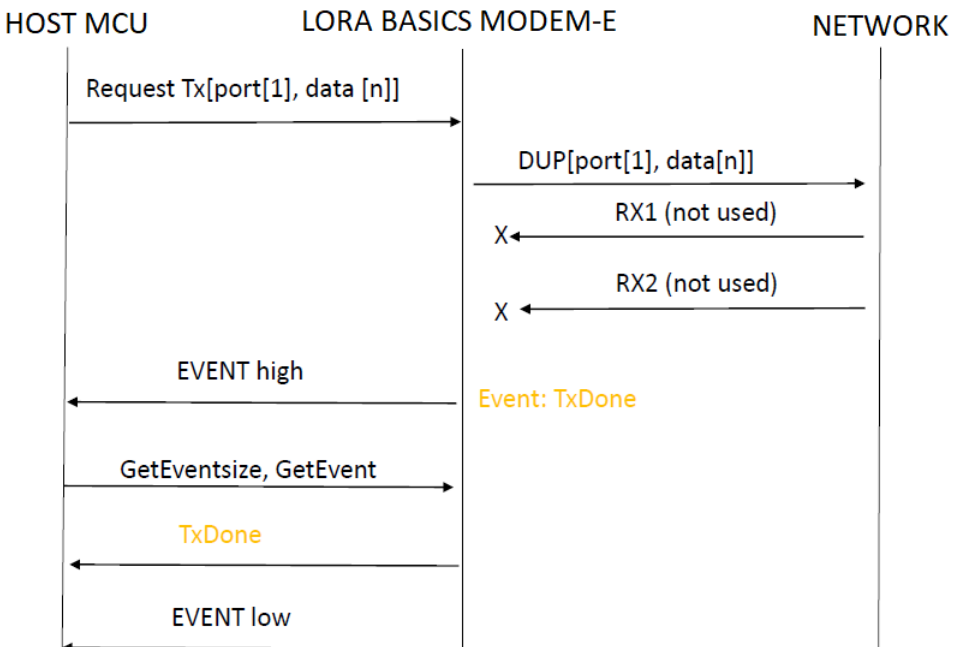


Figure 2-7: Class A TX Transaction shows the event sequence of a LoRaWAN® class A packet transmission.

Figure 2-7: Class A TX Transaction



2.3.3 GetEventSize Command Description

The command *GetEventSize(...)* reads back the Length in Bytes of the event stored in the LoRa Basics Modem-E.

Table 2-5: GetEventSize Command

Byte	0	1	2
Data from Host	0x06	0x33	CRC

Table 2-6: GetEventSize Response

Byte	0	(1:2)	3
Data to Host	ReturnCode	Length	CRC

- The *ReturnCode* (RC) is defined in Table 2-4: Return Code Definitions.

2.3.4 GetEvent Command Description

The command *GetEvent(...)* retrieves 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 will be returned and the count field indicates how many events of this type have been missed.

Table 2-7: GetEvent Command

Byte	0	1	2
Data from Host	0x06	0x00	CRC

Table 2-8: GetEvent Response

Byte	0	1	2	N	N+1
Data to Host	ReturnCode	EventType	Count	Data	CRC

- The *ReturnCode* (RC) is defined in Table 2-4: Return Code Definitions.
- The *EventType* is defined in Table 2-9: EventType Definition.
- *Count* indicates the number of events of this type missed.

Table 2-9: EventType Definition

EventType	Code	Description	Data
Reset	0x00	Modem has been reset	ResetCount(0:1)
Alarm	0x01	Alarm timer expired	--
Joined	0x02	Network successfully joined	--
TxDone	0x03	Frame transmitted	Status(1)
DownData	0x04	Downlink data received	Rssi(1), Snr(1), Flags(1), Port(1), Downdata(n)
UploadDone	0x05	File upload completed	Status(1)
SetConf	0x06	Config has been changed by DM	InfoTag(1)
Mute	0x07	Modem has been muted or unmuted by DM	Mute(1)
StreamDone	0x08	Last data stream fragment sent	--
JoinFail	0x0A	Attempt to join network failed	--
Wi-Fi	0x0B	Wi-Fi Passive scan done	*Variable*
GNSS	0x0C	GNSS scan done, SPI Almanac update done	*Variable*
TimeUpdatedAlcsync	0x0D	Time Updated by Alcsync (status=1: synchronization with the network, status = 0: synchronization lost after 6 days without answer)	Status(1)

Table 2-9: EventType Definition

EventType	Code	Description	Data
SwitchAdrMobiletoStatic	0x0E	Indicates the host MCU when the modem automatically switches from mobile to static when connection time-out occurs.	--
NewLinkAdr	0x0F	Modem receives an Link ADR request, the host MCU might check if need to change the ADR profile from static to mobile.	--
NoEvent	0xFF	No event exists. Notifies the user that no data must be read.	

2.3.5 GNSS Event Format

The GNSS Scan Done Event structure is shown in Table 2-10: GNSS Event Format.

Table 2-10: GNSS Event Format

Destination	Content
Destination	*Content*

- *Destination*=0x00 indicates that the event is intended for the Host MCU.

The *Content* field can be one of the following values, as shown in Table 2-11: GNSS Events for the Host MCU.

Table 2-11: GNSS Events for the Host MCU

Content	Description
0	Process OK
5	Capture failed
6	No time
7	No satellite detected
8	Almanac too old
9	Almanac update failed (CRC errors)
10	Almanac update failed (Flash integrity error)
13	Global Almanac CRC error at boot
14	Almanac version not supported

- *Dest*=0x01 indicates that the event is intended for the GNSS Solver:

Table 2-12: GNSS Events for the GNSS Solver

Content		Description
0x00	Payload=0x00	Request aiding position
0x01	Variable	NAV message

3. System, Register, Memory and Radio

Only the differences with the LR1110 Transceiver usage are described in this section. Refer to the LR1110 Transceiver User Manual for additional information.

3.1 System Modes

The LoRa Basics Modem-E uses the same system modes as the LR1110, but ensures automatic transitions between the different modes and sleep mode.

By default, the LoRa Basics Modem-E is in sleep mode for lowest power consumption. Toggling the NSS pin wakes-up the LoRa Basics Modem-E, so that it is ready to accept any command, or send a response byte stream (refer to Section 2.1.3 "SPI Write Diagram" on page 13, and Section 2.1.4 "SPI Read Diagram" on page 14). At the end of the command processing, the LoRa Basics Modem-E automatically returns to sleep mode.

The watchdog is not stopped during the LoRa Basics Modem-E sleep mode.

3.2 System Commands

The following commands, supported on the LR1110 Transceiver use case, are not supported on the LoRa Basics Modem-E:

- *GetStatus(...)*
- *GetVersion(...)*
- *GetErrors(...)*
- *ClearErrors*
- *Calibrate(...)*
- *CalibImage(...)*
- *SetDioIrqParams(...)*
- *ClearIrq(...)*
- *GetVbat(...)*
- *GetTemp(...)*
- *SetSleep(...)*
- *SetStandby(...)*
- *SetFs(...)*
- *GetSemtechJoinEui(...)*
- *DeriveRootKeysAndGetPin(...)*

The other commands are identical (same opcode, same arguments, same functionality).

Note 1: the DCDC to LDO selection is done manually by the user, and must be done depending on the application hardware configuration (electrical schematic). The DCDC configuration is set by default.

Note 2: The pin DIO10 can be configured as both input for XTAL 32kHz (command *ConfigLfClk(...)*) and output (command *SetDioAsRfSwitch(...)*). However these two usages are incompatible, therefore care must be taken to not configure the LF clock with Crystal Oscillator and enabling the DIO10 pin as RF Switch at the same time.

3.3 Reset Sources

In the LoRa Basics Modem-E, there are four ways to trigger a software reset:

- Using the SPI API modem command *Reset*(...), described in Section 6.4 "Reset" on page 29 .
- Upon a Network downlink reset request
- Through a watchdog time-out
- If the modem does not receive any downlink for a certain number of uplinks, configured using the command *SetConnectionTimeout(...)*

The command *Reboot(...)* has also the effect of triggering a reset, but this function is meant for firmware updates. For a description of this command, refer to the LR1110 Transceiver User Manual.

Each reset source generates a *Reset* event. The host MCU waits for this *Reset* event before sending any command.

The following parameters are stored in non-volatile memory, and therefore are kept after a software reset:

- ADR Custom Profile.
- DM Port.
- DevEUI. Default value: as per device provisioning in Semtech production flow
- JoinEUI. Default value: as per device provisioning in Semtech production flow
- Class. Default value= Class A
- Region. Default value= EU868

3.4 Register/Memory Commands

The following LR1110 Transceiver commands are not supported on the LoRa Basics Modem-E:

- *WriteBuffer8(...)*
- *ReadBuffer8(...)*
- *ClearRxBuffer(...)*
- *WriteRegMemMask32(...)*

The other commands are identical (same opcode, same arguments, same functionality).

3.5 Radio Commands

The radio blocks of the LoRa Basics Modem-E are directly controlled by the modem. Therefore, no LR1110 radio command is accepted. However, some test commands allow an evaluation of the LoRa Basics Modem-E radio parameters. Please refer Section 7. "Test Commands" on page 57 for the description of the radio test commands.

4. Wi-Fi Passive Scanning

The LoRa Basics Modem-E supports the same Wi-Fi passive scanning feature as for the LoRa Basics Modem-E Transceiver use case. Only the differences with the LoRa Basics Modem-E Transceiver use case are described in this section. Refer to the LoRa Basics Modem-E Transceiver User Manual for additional information.

4.1 Differences on LoRa Basics Modem-E Implementation

The differences between the LoRa Basics Modem-E and the Transceiver implementations of the Wi-Fi Passive Scanning in the LR1110 are as follows:

- The LoRa Basics Modem-E Wi-Fi Passive Scanning is initiated using three SPI transactions (refer to Figure 2-4).
- Events are used to indicate that the Wi-Fi Passive Scanning is done, and that results are available (refer to Section 2.3).
- The *GetEvent(...)* and *GetEventSize(...)* commands retrieve the Wi-Fi results.
- The command *WiFiPassiveScanMD(...)* is specific to the LoRa Basics Modem-E, and replaces the command *WifiScan(...)*.

As for the LR1110 Transceiver implementation, the Wi-Fi Passive Scanning results are sent to LoRa Cloud™ for the geolocation.

4.2 WiFiPassiveScanMD

The command *WiFiPassiveScanMD(...)* captures the Wi-Fi packets on the RFIO_HF pin. It is the LoRa Basics Modem-E implementation of the *WiFiPassiveScan(...)* command.

Table 4-1: WiFiPassiveScanMD Command

Byte	0	1	2	3	4	5	6	7	8	9	10	11
Data from Host	0x03	0x30	Wi-Fi Type	Chan Mask (15:8)	Chan Mask (7:0)	Acq Mode	Nb Max Res	Nb Scan Per Chan	Time out (15:8)	Time out (7:0)	Abort On Time out	Result Format

- *Wi-Fi Type* defines the type of the 801.11 signal to be scanned:
  - 0x01: Wi-Fi 802.11b type
  - 0x02: Wi-Fi 802.11g type
  - 0x03: Wi-Fi 802.11n type
  - 0x04: All signals: Wi-Fi b, then Wi-Fi g/n on the same channel
- *ChanMask* defines which Wi-Fi channels to be scanned, as per Table 4-2: Wi-Fi Channels for Wi-Fi Passive Scanning.]

Table 4-2: Wi-Fi Channels for Wi-Fi Passive Scanning

bit	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Channel	0	0	Ch 14	Ch 13	Ch 12	Ch 11	Ch 10	Ch 9	Ch 8	Ch 7	Ch 6	Ch 5	Ch 4	Ch 3	Ch 2	Ch 1

  - channel bit at 1 indicates that this channel must be scanned
- *AcqMode* indicates the *WifiScan* acquisition mode:
  - 0x01: Beacon search mode. Use only Wi-Fi beacons to extract MAC addresses.
  - 0x02: Beacon and Packet search mode. Use both Wi-Fi beacons and WI-Fi data packets to extract MAC addresses.
  - Other values are RFU.
- *NbMaxRes:* maximum total number of different MAC addresses wanted as a result for all scans on the various channels and Wi-Fi types (must be inferior or equal to 32). If this number is reached, passive scanning is stopped. If a MAC address already present in the result structure is detected a second time with a different RSSI value, then the new result is ignored.
- *NbScanPerChan:* number of Wi-Fi passive scans to be executed per channel (range: 1 to 255).
- *Timeout:* 16-bit time-out of the Preamble Search mode. Unit of *Timeout* is ms. For example, for a beacon period of 102.4 ms, a 105 ms time-out value can be set to ensure the *WifiScan* covers the whole beacon period.
- *AbortOnTimeout:* if set to 1, when a time-out preamble detect occurs, the passive scanning on this channel is aborted and the device jumps to the other channel to scan.
- ResultFormat: defines the format of the Wi-Fi Passive Scanning.
  - 1: Full results, as described in the LR1110 Transceiver User Manual.
  - 4: Basic results, as described in the LR1110 Transceiver User Manual.
  - Other values are RFU.

For example: the configuration

- Scan Wi-Fi b
- Channels 1, 6, 11
- Beacon and Packet search mode
- 10 Maximum Results
- 6 scans per channel
- 70 ms time-out for Preamble Search mode
- No abort on time-out
- Basic results format

Will be coded as:

Opcode	Wi-Fi Type	ChanMask	AcqMode	NbMaxRes	NbScanPer Chan	Timeout	AbortOn Timeout	Result Format
0x0300	0x01	0x0421	0x02	0x0A	0x06	0x0046	0x00	0x04

5. GNSS Scanning

The LoRa Basics Modem-E supports the same GNSS scanning feature as for the LR1110 Transceiver use case. Only the differences with the LR1110 Transceiver use case will be described in this section. Please refer to the LR1110 Transceiver User Manual for additional information.

5.1 Differences in LoRa Basics Modem-E Implementation

The differences between the LoRa Basics Modem-E and the LR1110 Transceiver implementations of the GNSS Scanning are as follows:

• The LoRa Basics Modem-E GNSS Scanning is initiated using three SPI transactions (refer to Figure 2-4).

• Events are used to indicate that the GNSS Scanning is done, and that results are available (refer to Section 2.3).

• The *GetEvent(...)* and *GetEventSize(...)* commands are used to retrieve the GNSS Scanning results.

• The command *GnssAutonomousMD(...)* is specific to the LoRa Basics Modem-E, and replaces the command *GnssAutonomous(...)*.

The command *GnssAssistedMD(...)* is specific to the LoRa Basics Modem-E, and replaces the command *GnssAssisted(...)*.

The SPI transactions and the event handling sequence are described in Figure 2-4 and Section 2.3.

As for the LR1110 Transceiver implementation, the GNSS Scanning results are sent to LoRa Cloud™ for the geolocation.

5.1.1 GnssAutonomousMD

The command *GnssAutonomousMD(...)* captures the GNSS signals in autonomous mode, for example if no assistance information is available, or for fast indoor/outdoor detection. It is the LoRa Basics Modem-E implementation of the *GnssAutonomous(...)* command.

Autonomous mode can be used if the assisted position is not available. In this case the solver can provide a coarse calculation of the position and send it back to the LoRa Basics Modem-E.

Table 5-1: GnssAutonomous Command

Byte	0	1	2	3	4
Data from Host	0x04	0x30	EffortMode	ResultMask	NbSvMax

- *EffortMode* =0x00. Other values are RFU.
- *ResultMask:* bit mask indicating which information is added in the NAV message.
  - Bit 0: If set, includes pseudo-ranges in the NAV message: 19 bits per satellite. Warning: if this bit is not set, then the solver cannot compute the location.
  - Bit 1: If set, include the Doppler information in the NAV message: 15 bits per satellites. Note however that at least 5 satellites will have their Doppler reported in the NAV message independently of this configuration. The Doppler information is used by the solver to compute a coarse location of the device. It is advised to set this bit.

- Bit 2: If set, includes bit changes in the NAV message: 1 byte per satellite. This bit is currently not used by the solver so setting it does not bring any improvement.

- *NbSvMax* defines the maximum number of satellites wanted as a result of the *GnssAutonomousMD(...)*. If more satellites are detected during the scanning than *NbSvMax,* then the satellites with the highest C/N0 are returned. If *NbSvMax*=0, then all the detected satellites are returned.

Please note that calling this command resets the previous GNSS results, if any.

5.1.2 GnssAssistedMD

The command *GnssAssistedMD(...)* captures the GNSS signals using assistance data (approximate position, and Almanac information). It is the LoRa Basics Modem-E implementation of the *GnssAssisted(...)* command.

Table 5-2: GnssAssistedMD Command

Byte	0	1	2	3	4
Data from Host	0x04	0x31	Effort Mode	ResultMask	NbSvMax

- *EffortMode*
  - 0x00: Low Power mode. The GNSS scanning stops the detection if no strong satellite is detected.
  - 0x01: Best Effort mode. The GNSS scanning continues the detection even if no strong satellite is detected.
- *ResultMask :* bit mask indicating which information is added in the NAV message.
  - Bit 0: If set, includes pseudo-ranges in the NAV message: 19 bits per satellite. Warning: if this bit is not set, then the solver cannot compute the location.
  - Bit 1: If set, include the Doppler information in the NAV message: 15 bits per satellites. Note however that at least 5 satellites will have their Doppler reported in the NAV message independently of this configuration. The Doppler information is used by the solver to compute a coarse location of the device. It is advised to set this bit.
  - Bit 2: If set, includes bit changes in the NAV message: 1 byte per satellite. This bit is currently not used by the solver so setting it does not bring any improvement.
- *NbSvMax* defines the maximum number of satellites to detect.

  If *NbSvMax*=0, all the detected satellites will be returned. Otherwise, only the *NbSvMax* satellites with higher C/N will be returned.

Please note that calling this command resets the previous GNSS results, if any.

5.1.3 GnssPushSolverMessage

The command *GnssPushSolverMessage(...)* is to be called to propagate a message from the LoRa Cloud™ DAS to the LoRa Basics Modem-E device. Typical usage is when a DAS downlink is received by the Host MCU through applicative downlink, and the Host transfers it to the LoRa Basics Modem-E device. .

Table 5-3: GnssPushSolverMessage Command

Byte	0	1	2	...	N+2
Data from Host	0x04	0x14	Payload (0)	...	Payload (N)

Payload: LoRa Cloud™ DAS payload to be transferred to the LoRa Basics Modem-E deviceas it is received, without alteration by the Host MCU.

6. LoRaWAN® Modem Commands

The LoRaWAN® Modem commands of the LoRa Basics Modem-E are listed here in increasing opcode order. Those commands are specific to the LoRa Basics Modem-E, and therefore not implemented in the LR1110 Transceiver use case.

Refer to Section 10. for the exhaustive list of commands.

6.1 LoRaWAN® Port

A few commands use the LoRaWAN® *Port* field as an argument. However, the following ports are used by Semtech service or reserved by LoRaWAN® standard. It is not recommended to change these ports, or use them for other purposes.

- port 0 (reserved for LoRaWAN MAC commands)
- port 199 (default Semtech device management port)
- port 200 (LoRaWAN® multi-cast)
- port 201 (LoRaWAN® fragmentation service)
- port 202 (LoRaWAN® Clock-sync service)
- port 224 (LoRaWAN® test port)
- port 225 - 255: Reserved by LoRaWAN® for future standardized application extensions

Note: If the user decides to change the ports of service, it is up to the user to ensure that the changed ports are not used by other purposes or services.

6.2 GetEvent

Refer to Section 2.3.4 "GetEvent Command Description" on page 17 for the *GetEvent*(...) command description.

6.3 GetVersionMD

The command *GetVersionMD (...)* returns the version of the bootloader, the functionality, the version of the installed firmware, and the version of the implemented LoRaWAN® standard of the LoRa Basics Modem-E.

Table 6-1: GetVersionMD Command

Byte	0	1
Data from Host	0x06	0x01

Table 6-2: GetVersionMD Response

Byte	(0:3)	4	(5:7)	(8:9)
Data to Host	Bootloader	Functionality	Firmware	Lorawan

- Bootloader: LoRa Basics Modem-E bootloader version, as per Table 6-3.

Table 6-3: GetVersionMD Bootloader Field

Byte	0	1	2	3
Meaning	HW Version	Use Case	Major Version	Minor Version

- Functionality: device supported functionalities
  - 0x01 = Transceiver + Wi-Fi /GPS scanner
  - 0x02= Modem + Wi-Fi scanner
  - 0x03= Modem + Wi-Fi/GPS scanner
  - 0x04= Modem + Wi-Fi/GPS/BeiDou scanner
  - 0x05= Modem only
  - Other fields are RFU
- Firmware: coded as per Table 6-4

Table 6-4: GetVersionMD Firmware Field

Byte	0	1	2
Meaning	FW Major	FW Minor	Patch

For example 0x000104 for FW Major=0x00, FW Minor=0x04, Patch=0x00

- Lorawan: version of the LoRaWAN® standard in BCD. For example, 0x0103 for LoRaWAN® 1.0.3.

The command *GetVersionMD (...)* does not generate any event.

6.4 Reset

The command *Reset (...)* performs a reset of the LoRa Basics Modem-E. All transient state (including session information) is lost and the LoRa Basics Modem-E must rejoin the LoRaWAN® network.

Table 6-5: Reset Command

Byte	0	1
Data from Host	0x06	0x02

The command *Reset (...)* generates a *Reset* event (refer to Table 2-9).

6.5 GetTxPowerOffset

The command *GetTxPowerOffset(...)* returns the board-specific correction offset of the transmission power to be used.

Table 6-6: GetTxPowerOffset Command

Byte	0	1
Data from Host	0x06	0x06

Table 6-7: GetTxPowerOffset Response

Byte	0
Data to Host	TXOffset

- *TXOffset*: Tx power offset signed integer in dB.

The command *GetTxPowerOffset(...)* does not generate any event.

6.6 SetTxPowerOffset

The command *SetTxPowerOffset(...) s*ets the board-specific correction offset of the transmission power to be used.

Table 6-8: SetTxPowerOffset Command

Byte	0	1	2
Data from Host	0x06	0x07	TXOffset

- *TXOffset*: Tx power offset in dB, expressed as signed int8. *TXOffset* depends on the board and antenna design.

The command *SetTxPowerOffset*(...) does not generate any event.

6.7 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.

For a complete description of the command *Test(...)*, refer to Section 7.

6.8 GetTime

The command *GetTime(...)* returns the GPS wall time (number of seconds since 00:00:00 January 6th 1980).

Table 6-9: GetTime Command

Byte	0	1
Data from Host	0x06	0x0A

Table 6-10: GetTime Response

Byte	(0:3)
Data to Host	Timestamp

- *Timestamp*: Current GPS time in seconds since GPS epoch (00:00:00, January, 6 1980).

The application layer clock synchronization (ALC Sync) protocol links the device clock to GPS time. The host MCU can enable the clock sync service by the command *SetAlcSyncMode(...)*. The LoRa Basics Modem-E will then manage the periodic uplink to achieve clock sync with the network.

The ALC Sync will add some offset to the device's time if LoRa Cloud™ notices an offset between device's GPS time and the network's GPS time. 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.

The command *GetTime*(...) does not generate any event.

6.9 GetStatus

The command *GetStatus(...)* returns the LoRaWAN® modem status which may indicate one or more notification conditions.

Table 6-11: GetStatus Command

Byte	0	1
Data from Host	0x06	0x0B

Table 6-12: GetStatus Response

Byte	0
Data to Host	Status

- *Status*: LoRaWAN® modem status, according to the table Section Table 6-13: hereafter:

Table 6-13: LoRaWAN® modem status

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

The command *GetStatus(...)* does not generate any event.

6.10 SetAlarmTimer

The command *SetAlarmTimer(...)* sets an application alarm or wakeup timer.

Table 6-14: SetAlarmTimer Command

Byte	0	1	(2:5)
Data from Host	0x06	0x0C	Seconds

- *Seconds*: Alarm or wakeup timer in seconds.

When the timer has expired, an Alarm event is generated. If this command is applied again before the timer has expired, the timer restarts with the new period. A value of 0 cancels an eventually pending previous alarm timer.

The command *SetAlarmTimer(...)* generates an alarm event (refer to Table 2-9).

6.11 GetPin

The command *GetPin(...)* returns the device registration PIN.

Table 6-15: GetPin Command

Byte	0	1
Data from Host	0x06	0x0E

Table 6-16: GetPin Response

Byte	(0:3)
Data to Host	PIN

- *PIN*: Device registration PIN.

The command *GetPin(...)* does not generate any event.

6.12 GetChipEui

The command *GetChipEUI(...)* returns the device ChipEUI

Table 6-17: GetChipEui Command

Byte	0	1
Data from Host	0x06	0x0F

Table 6-18: GetChipEui Response

Byte	(0:7)
Data to Host	ChipEUI

- *ChipEUI*: ChipEUI

The byte array order example is as follows:

- *ChipEUI* field read by the host MCU: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08
- LoRa Cloud™ uplink of Chip EUI: 0x08 0x07 0x06 0x05 0x04 0x03 0x02 0x01
- LoRa Cloud™ shows Chip EUI: 0x01 0x02 0x03 0x04 0x5 0x06 0x07 0x08

ChipEUI is also the default Device EUI. It is programmed during manufacturing and is immutable.

The command *GetChipEui(...)* does not generate any event.

6.13 GetJoinEui

The command *GetJoinEui(...)* returns the join EUI.

Table 6-19: GetJoinEui Command

Byte	0	1
Data from Host	0x06	0x10

Table 6-20: GetJoinEui Response

Byte	(0:7)
Data to Host	JoinEUI

- *JoinEUI*: join EUI

The byte array order example is as follows:

- *JoinEUI* field read by the host MCU: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08
- LoRa Cloud™ uplink of Join EUI: 0x08 0x07 0x06 0x05 0x04 0x03 0x02 0x01
- LoRa Cloud™ shows Join EUI: 0x01 0x02 0x03 0x04 0x5 0x06 0x07 0x08

The command *GetJoinEui(...)* does not generate any event.

6.14 SetJoinEui

The command *SetJoinEui(...)* sets the join EUI. It does not derive the keys.

Table 6-21: SetJoinEui Command

Byte	0	1	(2:9)
Data from Host	0x06	0x11	JoinEUI

- *JoinEUI*: join EUI.

The command *SetJoinEui(...)* does not generate any event.

6.15 GetDevEui

The command *GetDevEui(...)* returns the device EUI.

Table 6-22: GetDevEui Command

Byte	0	1
Data from Host	0x06	0x12

Table 6-23: GetDevEui Response

Byte	(0:7)
Data to Host	DevEUI

- *DevEUI*: device EUI.

The byte array order example is as follows:

- *DevEUI* field read by the host MCU: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08

- LoRa Cloud™ uplink of Dev EUI: 0x08 0x07 0x06 0x05 0x04 0x03 0x02 0x01
- LoRa Cloud™ shows Dev EUI: 0x01 0x02 0x03 0x04 0x5 0x06 0x07 0x08

The command *GetDevEui(...)* does not generate any event.

6.16 SetDevEui

The command *SetDevEui(...)* configures the device EUI.It does not derive the keys.

Table 6-24: SetDevEui Command

Byte	0	1	(2:9)
Data from Host	0x06	0x13	DevEUI

- *DevEUI*: device EUI.

The command *SetDevEui(...)* does not generate any event.

6.17 SetAppKey

The command *SetAppKey(...)* configures the LoRaWAN® application key.

Table 6-25: SetAppKey Command

Byte	0	1	(2:17)
Data from Host	0x06	0x14	AppKey

- *AppKey*: LoRaWAN® Application key.

A factory reset erases *AppKey*. The device is then required to rejoin the network.

The command *SetDevEui(...)* does not generate any event.

6.18 GetClass

The command *GetClass(...)* returns the LoRaWAN® device class.

Table 6-26: GetClass Command

Byte	0	1
Data from Host	0x06	0x15

Table 6-27: GetClass Response

Byte	0
Data to Host	Class

- *Class*: LoRaWAN® device class

The command *GetClass(...)* does not generate any event.

6.19 SetClass

The command *SetClass(...)* configures the LoRaWAN® device class.

Table 6-28: SetClass Command

Byte	0	1	2
Data from Host	0x06	0x16	Class

- *Class*: LoRaWAN® device class:
  - 0x00: Class A
  - Other values are RFU.

A factory reset sets *Class* to its default value (Class A).

The command *SetClass(...)* does not generate any event.

6.20 GetRegion

The command *GetRegion(...)* returns the regulatory region.

Table 6-29: GetRegion Command

Byte	0	1
Data from Host	0x06	0x18

Table 6-30: GetRegion Response

Byte	0
Data to Host	Region

- *Region*: regulatory region, according to Table 6-31.

Table 6-31: Regulatory region

Region	Code
EU868	0x01
US915	0x03

The command *GetRegion(...)* does not generate any event.

6.21 SetRegion

The command *SetRegion(...)* configures the regulatory region.

Table 6-32: SetRegion Command

Byte	0	1	2
Data from Host	0x06	0x19	Region

- *Region*: regulatory region, according to Table 6-31.

Not all regions may be supported. Use the command *ListRegions(...)* to enumerate the available regions. This command resets the ADR profile to Network Server Controlled. If a different ADR profile is desired, the profile needs to be set again.

The command *SetRegion(...)* does not generate any event.

6.22 ListRegions

The command *ListRegions(...)* returns the list of supported regulatory regions.

Table 6-33: ListRegions Command

Byte	0	1
Data from Host	0x06	0x1A

Table 6-34: ListRegions Response

Byte	N
Data to Host	Regions

- *Regions*: list of supported regulatory regions, according to Table 6-31.

It is necessary to send the command *GetCmdRspSize(...)* before sending this command, since the number of Bytes to read in the response must be known before issuing the command.

The command *ListRegions(...)* does not generate any event.

6.23 GetAdrProfile

The command *GetAdrProfile(...)* returns the ADR profile type.

Table 6-35: GetAdrProfile Command

Byte	0	1
Data from Host	0x06	0x1B

Table 6-36: GetAdrProfile Response

Byte	0
Data to Host	Type

- *Type*: ADR profile, according to Table 6-38.

The command *GetAdrProfile(...)* does not generate any event.

6.24 SetAdrProfile

The command *SetAdrProfile(...)* configures the regulatory region.

Table 6-37: SetAdrProfile Command

Byte	0	1	2	(3:18) (optional)
Data from Host	0x06	0x1C	Type	List (optional)

- *Type*: ADR profile, according to Table 6-38.

Table 6-38: ADR profile types

Code	Name	Parameters
0x00	Network Server Controlled	-
0x01	Mobile Long Range	-
0x02	Mobile Low Power	-
0x03	Custom	list of preferred data rates [16]

- Network Server Controlled: the network server controls the ADR, and the device is static.
- Mobile Long Range: used when the device is mobile, and intends for long range to reach distant LoRaWAN® gateways.

- Mobile Low Power: used when the device is mobile, and intends for low power.
- Custom profile: the host MCU defines the ADR profile in 16 bytes, if the host MCU knows its network configuration.

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 types 0-2 use the following distributions:

- Network Server Controlled: dynamic (for stationary devices).
- Mobile Long Range and Mobile Low Power: as indicated in Table 6-40

Table 6-39: Mobile Long Range and Mobile Low Power ADR Profile DR Usage

Region	Mode	DR0	DR1	DR2	DR3	DR4	DR5
EU868	Long Range	20%	20%	30%	30%	-	-
	Low Power	-	-	10%	30%	30%	30%
US915	Long Range	20%	20%	30%	30%	-	-
	Low Power	-	10%	40%	50%	-	-

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.

The command *SetAdrProfile(...)* does not generate any event.

6.25 GetDmPort

The command *GetDmPort(...)* returns the device management port.

Table 6-40: GetDmPort Command

Byte	0	1
Data from Host	0x06	0x1D

Table 6-41: GetDmPort Response

Byte	0
Data to Host	Port

- *Port*: device management port.

The command *GetDmPort(...)* does not generate any event.

6.26 SetDmPort

The command *SetDmPort(...)* configures the device management port.

Table 6-42: SetDmPort Command

Byte	0	1	2
Data from Host	0x06	0x1E	Port

- *Port*: device management port. Refer to Section 6.1 for additional information.

The command *SetDmPort(...)* does not generate any event.

6.27 GetDmInfoInterval

The command *GetDmInfoInterval(...)* returns the device management reporting interval.

Table 6-43: GetDmInfoInterval Command

Byte	0	1
Data from Host	0x06	0x1F

Table 6-44: GetDmInfoInterval Response

Byte	0
Data to Host	Interval

- *Interval*: device management reporting interval, as described in Table 6-45. The interval is specified in seconds, minutes, hours or days. The two top-most bits specify the unit (sec/min/hour/day), and the lower six bits the value 0-63.

Table 6-45: Reporting Interval Format

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

The command *GetDmInfoInterval(...)* does not generate any event.

6.28 SetDmInfoInterval

The command *SetDmInfoInterval(...)* configures the device management reporting interval.

Table 6-46: SetDmInfoInterval Command

Byte	0	1	2
Data from Host	0x06	0x20	Interval

- *Interval*: device management reporting interval. The interval is specified in seconds, minutes, hours or days, as described in Table 6-45.

Any value of 0 (seconds, minutes, hours, or days) disables device management reporting.

The command *SetDmInfoInterval(...)* does not generate any event.

6.29 GetDmInfoFields

The command *GetDmInfoFields(...)* lists the info fields to be included in the periodic device management status messages.

Table 6-47: GetDmInfoFields Command

Byte	0	1
Data from Host	0x06	0x21

Table 6-48: GetDmInfoFields Response

Byte	(0:N-1)
Data to Host	InfoList

- *InfoList*: list of tag bytes, status information codes, as shown in Table 6-49 here below.

Table 6-49: Device Management InfoList

Value	Meaning
0x00	DM_INFO_TYPE_STATUS
0x03	DM_INFO_TYPE_TEMPERATURE
0x04	DM_INFO_TYPE_SIGNAL
0x05	DM_INFO_TYPE_UPTIME
0x06	DM_INFO_TYPE_RXTIME
0x07	DM_INFO_TYPE_FIRMWARE
0x08	DM_INFO_TYPE_ADR_MODE

Table 6-49: Device Management InfoList

Value	Meaning
0x09	DM_INFO_TYPE_JOIN_EUI
0x0A	DM_INFO_TYPE_INTERVAL
0x0B	DM_INFO_TYPE_REGION
0x0F	DM_INFO_TYPE_RESET_COUNT
0x10	DM_INFO_TYPE_DEV_EUI
0x11	DM_INFO_TYPE_OWNER_COUNTER
0x12	DM_INFO_TYPE_SESSION_ID
0x13	DM_INFO_TYPE_CHIP_EUI
0x15	DM_INFO_TYPE_STREAM_PARAMETERS
0x16	DM_INFO_TYPE_APPLICATION_SPECIFIC_STATUS
0x18	DM_INFO_TYPE_GNSS_ALMANAC_STATUS

It is necessary to send the command *GetCmdRspSize(...)* before sending this command, since the number of Bytes to read in the response must be known before issuing the command.

The command *GetDmInfoFields(...)* does not generate any event.

6.30 SetDmInfoFields

The command *SetDmInfoFields(...)* configures the default info fields to be included in the periodic device management status messages.

Table 6-50: SetDmInfoFields Command

Byte	0	1	(2:N+1)
Data from Host	0x06	0x22	InfoList(N)

- *InfoList*: list of tag bytes.

The set is specified as list of field codes as defined in Uplink Message Format. Duplicate and invalid fields are rejected. An empty set is valid and effectively disables the device management status message.

The command *SetDmInfoFields(...)* does not generate any event.

6.31 SendDmStatus

The command *SendDmStatus(...)* sends the specified set of information fields in one or more device management status messages immediately.

Table 6-51: SendDmStatus Command

Byte	0	1	(2:N+1)
Data from Host	0x06	0x23	InfoList(N)

- *InfoList*: list of tag bytes.

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

The command *SendDmStatus(...)* does not generate any event.

6.32 SetAppStatus

The command *SetAppStatus(...)* configures application-specific status information to be reported to the device management service.

Table 6-52: SetAppStatus Command

Byte	0	1	(2:9)
Data from Host	0x06	0x24	AppStatus

- *AppStatus*: application status information.

This information is an application-defined, arbitrary 8-byte data stream. Once set, it is included in the *AppStatus* info field sent as part of the periodic status reports to the DM service. On the LoRa Cloud™ side, this information can then be retrieved from the service.

> **Note: this command does not trigger an immediate status report (configured via the command *SetDmInfoInterval(...)*. If the application desires to send the status immediately, it can issue the command *SendDmStatus(...)* with the *AppStatus* tag.**

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

The command *SetAppStatus(...)* does not generate any event.

6.33 Join

The command *Join* starts joining the network.

Table 6-53: Join Command

Byte	0	1
Data from Host	0x06	0x25

During the join procedure no further transmissions can occur. 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. The join process stops if the modem receives an ACK or if the host MCU aborts the ongoing join attempt using the command *LeaveNetwork(...)*.

A *JoinFail* event is generated for every join without valid ACK, but the join process continues.

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 command *SetAdrProfile(...)*. The ADR profile only applies to uplinks after the network is joined and the session is already established. The data rates used during the join procedure are detailed in Table 6-54.

Table 6-54: LoRa Basics Modem-E Data Rate Usage for the Join Procedure

EU868	US915
DR0: 5% (SF12, 125 kHz BW)	DR0: 50% (SF10, 125 kHz BW)
DR1: 10% (SF11, 125 kHz BW)	DR1: 0% (SF9, 125 kHz BW)
DR2: 15% (SF10, 125 kHz BW)	DR2: 0% (SF8, 125 kHz BW)
DR3: 20% (SF9, 125 kHz BW)	DR3: 0% (SF7, 125 kHz BW)
DR4: 20% (SF8, 125 kHz BW)	DR4: 50% (SF8, 500kHz BW)
DR5: 30% (SF7, 125 kHz BW)	-

When the network has been successfully joined, a *Joined* event is generated

6.34 LeaveNetwork

The command *LeaveNetwork* leaves the network if already joined, or cancels an ongoing join process.

Table 6-55: LeaveNetwork Command

Byte	0	1
Data from Host	0x06	0x26

After leaving the network, no further transmissions can occur.

The command *LeaveNetwork* does not generate any event.

6.35 SuspendModemComm

The command *SuspendModemComm(...)* temporarily suspends or resumes the modem's radio operations.

Table 6-56: SuspendModemComm Command

Byte	0	1	2
Data from Host	0x06	0x27	Suspend

- *Suspend*:
  - 0x00: resume radio operation
  - 0x01: suspend radio operation.

The command *SuspendModemComm(...)* does not generate any event.

6.36 GetNextTxMaxPayload

The command *GetNextTxMaxPayload(...)* 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 Frame Options (FOpts) are present and that a device is not behind a repeater.

Table 6-57: GetNextTxMaxPayload Command

Byte	0	1
Data from Host	0x06	0x28

Table 6-58: GetNextTxMaxPayload Response

Byte	0
Data from Host	Size

- *Size*: maximum payload size in bytes.

The command *GetNextTxMaxPayload(...)* does not generate any event.

6.37 RequestTx

The command *RequestTx(...)* requests to send the given data on the specified port as an unconfirmed or confirmed frame.

Table 6-59: RequestTx Command

Byte	0	1	2	3	(4:245)
Data from Host	0x06	0x29	Port	Conf	Data

- *Port*: port. Refer to Section 6.1 for additional information.
- Conf: frame confirmation
  - 0x00: unconfirmed frame
  - 0x01: confirmed frame

  Note: use confirmed frames with caution in order to avoid channel congestion.
- Data: data to be transmitted, in Bytes (1-242).

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 either been sent, or if it couldn't be sent because the specified data exceeded the maximum possible payload size.

The parameter of the *TxDone* event indicates if 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 windows a *DownData* event is generated containing the port and data received.

For confirmed frames, the LoRa Basics Modem-E sends 4 frames (3 repetitions) if no ACK is received.

If a command *RequestTX(...)* is issued before the *TxDone* event of a previous transmission request is generated, the command fails with Busy return code. If the command is issued before the network has been joined, it fails with NotInit return code.

6.38 EmergencyTx

The command *EmergencyTx(...)* requests to send the given data on the specified port as an unconfirmed or confirmed frame immediately.

Table 6-60: EmergencyTx Command

Byte	0	1	2	3	(4:245)
Data from Host	0x06	0x2A	Port	Conf	Data

- *Port*: port. Refer to Section 6.1 for additional information.
- Conf: frame confirmation
  - 0x00: unconfirmed frame
  - 0x01: confirmed frame

  Note: use confirmed frames with caution in order to avoid channel congestion.
- *Data*: data to be transmitted, in Bytes (1-242).

EmergencyTx(...) has 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!

For confirmed frames, the LoRa Basics Modem-E sends 4 frames (3 repetitions) if no ACK is received.

A *TxDone* event is generated when the frame has been sent. The parameter of the *TxDone* event indicates if the frame was sent and acknowledged (0x02), or sent but not acknowledged (0x01).

6.39 UploadInit

The command *UploadInit(...)* prepares a fragmented file upload.

Table 6-61: UploadInit Command

Byte	0	1	2	3	(4:5)	6
Data from Host	0x06	0x2B	Port	EncryptMode	Size	Delay

- *Port*: port. Refer to Section 6.1 for additional information.
- EncryptMode: encryption mode
  - 0x00: unencrypted frame
  - 0x01: encrypted frame
- *Size*: file size.
- *Delay*: average frame transmission interval (in seconds).

The port is included as meta data in the file stream and the stream is sent on the device management port so it can be processed by the Semtech LoRa Cloud™ Service.

When encryption is used (mode value 0x01), the file data is encrypted using a 128-bit AES key derived from the AppSKey before it is further processed by the upload service. This command should also be used to cancel an ongoing file upload, or in case of file upload error (ex: badCrc), by specifying a file size of zero for the port in use.

Note: 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!

The command *UploadInit(...)* does not generate any event.

6.40 UploadData

The command *UploadData(...)* can be used to repeatedly set file data to be uploaded.

Table 6-62: UploadData Command

Byte	0	1	N+1
Data from Host	0x06	0x2C	Data

- *Data*: file data to be uploaded (N Bytes).

The file data needs to be split into parts of maximum 255 Bytes each, and the submitted parts will be appended to an internal buffer. In total, exactly as many Bytes as specified by the *UploadInit(...)* command have to be provided. The maximum size of file data is 2KBytes.

The command *UploadData(...)* does not generate any event.

6.41 UploadStart

After all data bytes indicated to *UploadInit*(...) have been provided using *UploadData*(...), *UploadStart*(...) can be issued to actually start the transmission stream.

Table 6-63: UploadStart Command

Byte	0	1	(2:5)
Data from Host	0x06	0x2D	FileCrc

- *Data*: file data to be uploaded (N Bytes).

The amount of data provided by the commands UploadData must match the size specified by *UploadInit(...)* command, otherwise the command is rejected with *BadSize* return code. The FileCrc parameter must match the CRC of the supplied data, otherwise the command is rejected with *BadCrc* return code.

If encryption was requested with the command *UploadInit(...),* the data is encrypted before the stream is started.

When the stream is started, 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.

A running file upload is indicated by the Upload flag in the modem status.

When the upload stream stops, a *UploadDone* event is generated, which carries a status byte *Success* or *Timeout*.

6.42 StreamInit

The command *StreamInit(...)* initializes redundant data streaming on a specific port.

Table 6-64: StreamInit Command

Byte	0	1	2	3
Data from Host	0x06	0x2E	Port	Mode

- *Port*: port on which the streaming protocol operates. 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. Refer to Section 6.1 for additional information.
- *Mode*: encryption mode of the data stream.
  - 0x00: unencrypted stream
  - 0x01: encrypted stream

If the data stream is encrypted, the decoding engine can reassemble the fragments, but is not able to see the data.

The *StreamInit(...)* command can only be issued before the stream has been started using the command *SendStreamData(...)*. Currently the modem supports only one data stream. After the stream has been initialized, data records can be submitted using the command *SendStreamData(...)* 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.

The command *StreamInit(...)* does not generate any event.

6.43 SendStreamData

The command *SendStreamData(...)* adds a new data record to the buffer of the data streaming encoder for the given port.

Table 6-65: SendStreamData Command

Byte	0	1	2	2+N
Data from Host	0x06	0x2F	Port	Record

- *Port*: port on which the streaming protocol operates. Refer to Section 6.1 for additional information.
- *Record*: data record (N Bytes).

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 (total buffer size is 512 bytes).

Encryption of the data stream is done according to the mode parameter of the previously issued command *StreamInit(...)* for the port. If the command *StreamInit(...)* was never issued, the data stream is sent unencrypted to the given port.

When all data of the buffer has been sent, a *StreamDone* event is generated.

6.44 StreamStatus

The command *StreamStatus(...)* queries the status of the data streaming buffer on the specified port.

Table 6-66: StreamStatus Command

Byte	0	1	2
Data from Host	0x06	0x30	Port

Table 6-67: StreamStatus Response

Byte	(0:1)	(2:3)
Data to Host	Pending	Free

- *Port*: port on which the streaming protocol operates. Refer to Section 6.1 for additional information.
- *Pending*: number of Bytes pending for transmission
- *Free*: number of Bytes still free in the buffer.

The command *StreamStatus(...)* does not generate any event.

6.45 GetCmdRspSize

The command *GetCmdRspSize(...)* returns the size of response to a read command.

Table 6-68: GetCmdRspSize Command

Byte	0	1	2
Data from Host	0x06	0x31	Cmd

Table 6-69: GetCmdRspSize Response

Byte	0
Data to Host	ByteLength

- *Cmd*: read command
  - 0x08: Test
  - 0x1A: List Regions
  - 0x21: GetDmInfoFields

The return code will be invalid if other commands are used as input.

- *ByteLength*: number of Bytes of the application payload response.

The command *GetCmdRspSize(*...) does not generate any event.

6.46 SetTime

The command *SetTime(...)* sets the GPS time. It is used by the application to set the modem clock in case the network clock synchronization is not used.

Table 6-70: SetTime Command

Byte	0	1	(2:5)
Data from Host	0x06	0x32	Time

- *Time*: GPS time, as number of seconds since January 6, 1980 00:00:00 in big endian format.

The command *SetTime(*...) does not generate any event.

6.47 GetEventSize

The command *GetEventSize(...)* is described in Section 2.3.3 "GetEventSize Command Description" on page 17 .

6.48 DeriveKeys

The command *DeriveKeys(...)* uses the previously set JoinEUI/DevEUI to derive the AppKey.

Table 6-71: DeriveKeys Command

Byte	0	1
Data from Host	0x06	0x34

The AppKey derivation algorithm used (smtc1) is described in Semtech LoRa Cloud™ at the following location: https://www.loracloud.com/documentation/join_service?url=bkey_schemes.html

The command *DeriveKeys(...)* does not generate any Event.

6.49 ManageRFOutput

The command *ManageRFOutput(...)* defines the RF output configuration.

Table 6-72: ManageRFOutput Command

Byte	0	1	2
Data from Host	0x06	0x36	Config

- Config: sub-GHz RF output configuration.
  - 0x00: RFO_LP_LF path, up to +15dBm (default configuration)
  - 0x01: RFO_HP_LF path, up to +22dBm
  - 0x02: Both RFO_LP_LF and RFO_HP_LF paths, automatically controlled by the LoRa Basics Modem-E. The application has to ensure that the RF path is connected to the antenna using the command *SetDioAsRfSwitch(...)*
  - 0x03 to 0xFF: RFU

The command *ManageRFOutput(...)* does not generate any Event.

6.50 SetAlcSyncPort

The command *SetAlcSyncPort(...)* defines the port for the application layer clock synchronization service.

Table 6-73: SetAlcSyncPort Command

Byte	0	1	2
Data from Host	0x06	0x37	Port

- *Port*: ALC Sync port. Refer to Section 6.1 for additional information.

The command *SetAlcSyncPort(...)* does not generate any Event.

6.51 GetAlcSyncPort

The command *GetAlcSyncPort(...)* returns the port for application layer clock synchronization service.

Table 6-74: GetAlcSyncPort Command

Byte	0	1
Data from Host	0x06	0x38

Table 6-75: GetAlcSyncPort Response

Byte	0
Data from Host	Port

- *Port*: ALC Sync port. Refer to Section 6.1 for additional information.

The command *GetAlcSyncPort(...)* does not generate any Event.

6.52 SetAlcSyncMode

The command *SetAlcSyncMode(...)* enables or disables the application layer clock synchronization (ALC) service.

Table 6-76: SetAlcSyncPort Command

Byte	0	1	2
Data from Host	0x06	0x39	Mode

- *Mode*:
  - 0x00 (default): disables the ALC Sync service
  - 0x01: enables the ALC Sync service

The ALC Sync service can also be enabled when the modem has not joined. Once joined, the modem automatically sends the ALC Sync uplink to get the time from the network. The ALC sync service is managed by the modem, and the ALC Sync works in three phases: sync-recent, sync-longtime and not-sync:

- Sync-recent: If the modem receives an ALC Sync downlink, an Event time sync is generated to inform the host MCU. During the following 3 days, the ALC Sync sends an uplink every 36 hours with the bit *AnsRequired* =0.
- Sync-longtime: 3 days after the modem receives the ALC Sync downlink, the modem continues to send ALC Sync uplinks every 36 hours, but with the bit *AnsRequired* =1.
- Not-sync: the modem does not have time, or the time is not updated by the host MCU or by the ALC Sync service during 6 days. In that case, time synchronisation is lost and an Event time desync is generated to inform the host MCU. Therefore, the modem does not provide time, and the command *GetTime(...)* returns 0. The modem accelerates the time re-sync in the following ways: 3 ALC Sync uplinks at 128s intervals, then 1 uplink every 4 hours for the first 24 hours, and then 1 uplink every 36 hours. All these ALC Sync uplinks have the bit *AnsRequired* =1.

The command *SetAlcSyncMode(...)* does not generate any Event.

6.53 GetAlcSyncMode

The command *GetAlcSyncMode(...)* returns the application layer clock synchronization service mode (enable/disable).

Table 6-77: GetAlcSyncMode Command

Byte	0	1
Data from Host	0x06	0x3A

Table 6-78: GetAlcSyncMode Response

Byte	0
Data to Host	Mode

- *Mode*: ALC Sync mode.

The command *GetAlcSyncMode(*...) does not generate any Event, but an event is generated when the date is updated.

6.54 SetConnectionTimeout

The command *SetConnectionTimeout(...)* configures the number of uplinks without downlinks from network before the LoRaWAN® Modem resets and change the ADR profile from mobile (Mobile Long Range and Mobile Low Power) to static (Network Server Controlled). This means that receiving downlink packets will allow the LoRa Basics Modem-E to ensure that it is connected.

Table 6-79: SetConnectionTimeout Command

Byte	0	1	(2:5)
Data from Host	0x06	0x3C	NbUplink

- *NbUplink*: (4 Bytes) number of uplinks without a downlink before disconnection.
  - The first 2 Bytes define the number of uplinks without a downlink for the modem ADR profile to switch from mobile to static. Default value= 255.
  - The second 2 Bytes define the number of uplinks without a downlink for the modem to automatically reset. Default value= 2400.

It is recommended to have the first counter smaller than the second one.

The value 0 deactivates the recovery function. When the Modem falls back from ADR mobile to static mode, the modem triggers the event *SwitchAdrMobiletoStatic* (0x0E).

After reset, these default values are lost, and the host MCU should re-configure them.

> **Note: the LoRaWAN® bit ADRAckReq can check the connection status. Please refer to the LoRaWAN® specification for more details.**

The command *SetConnectionTimeout(*...) does not generate any Event.

6.55 GetConnectionTimeout

The command *GetConnectionTimeout(...)* returns the configured number of uplinks without a downlink from the network before the LoRaWAN® Modem resets and changes the ADR profile from mobile to static.

Table 6-80: GetConnectionTimeout Command

Byte	0	1
Data from Host	0x06	0x3D

Table 6-81: GetConnectionTimeout Response.

Byte	(0:3)
Data to Host	NbUplink

- *NbUplink*: number of uplinks before disconnection.

The command *GetConnectionTimeout(*...) does not generate any Event.

6.56 SetCertificationMode

The command *SetCertificationMode(...)* enables/disables the LoRaWAN® certification mode.

Table 6-82: SetCertificationMode Command

Byte	0	1	2
Data from Host	0x06	0x3E	Mode

- *Mode*: LoRaWAN® certification mode
  - 0x00 (default): disable
  - 0x01: enable

After the certification mode is enabled, all the tests run automatically when the host MCU launches a join. After the modem is joined, the certification mode deactivates the duty cycle, catches and analyses all the events (the host MCU could see the event pin is driven high, but no event data is available).

When all the tests are over, the host MCU should deactivate certification mode. Before leaving certification mode, the duty cycle will be re-enabled by the modem. The host MCU is recommended to do a soft reset or leave the network and then re-join the network.

The command *SetCertificationMode(*...) does not generate any Event.

6.57 GetCertificationMode

The command *GetCertificationMode(...)* returns the LoRaWAN® certification mode.

Table 6-83: GetCertificationMode Command

Byte	0	1
Data from Host	0x06	0x3F

Table 6-84: GetCertificationMode Response.

Byte	0
Data to Host	Mode

- *Mode*: LoRaWAN® certification mode

The command *GetCertificationMode(...)* does not generate any Event.

6.58 GetLRWANState

The command *GetLRWANState(...)* returns the state of the LoRaWAN® stack.

Table 6-85: GetLRWANState Command

Byte	0	1
Data from Host	0x06	0x40

Table 6-86: GetLRWANState Response.

Byte	0
Data to Host	LoRaWANState

- *LoRaWANState*: LoRaWAN® state.
  - 0x00: idle, no LoRaWAN® operation on-going. The host MCU can freely send Wi-FI or GNSS commands.
  - 0x01: not idle, LoRaWAN® operation on-going. It is not recommended that the host MCU sends Wi-FI or GNSS commands, as this will interrupt the LoRaWAN® stack.

The command *GetLRWANState(...)* does not generate any Event.

6.59 GetStatusDutyCycle

The command *GetStatusDutyCycle(...)* returns the LoRaWAN® stack radio transmit duty cycle.

Table 6-87: GetStatusDutyCycle Command

Byte	0	1
Data from Host	0x06	0x44

Table 6-88: GetStatusDutyCycle Response.

Byte	(0:3)
Data to Host	DutyCycleStatus

- *DutyCycleStatus*: Duty Cycle Status

When *DutyCycleStatus* >0, *DutyCycleStatus* indicates the time in ms that host MCU should wait before transmitting a frame. Therefore, *DutycCycleStatus*=0 indicates that the LoRa Basics Modem-E is allowed to transmit a frame.

When the Duty cycle is deactivated, the returned value is always 0.

The command *GetStatusDutyCycle(...)* does not generate any Event.

7. Test Commands

TEST commands are used to implement test functionality for regulatory conformance, certification, and functional testing. With the exception of the *TST_START(...)* command, test commands are only available if test mode is active. Test mode can only be activated if the device has not yet received a command that results in a radio operation. Once test mode is active, all other modem commands are disabled. If the device is already in test mode with a test running, the user must quit the test mode with *TST_EXIT(...)* command before sending a new test mode command.

7.1 Test Commands Summary

The list of test commands is summarized in Table 7-1 here below. The commands are described in details in the following sections..

Table 7-1: List of Test Commands

Command	Opcode	Description
TST_START	0x00	Starts test mode.
TST_NOP	0x01	No operation.
TST_TX_SINGLE	0x02	Transmits a single packet.
TST_TX_CONT	0x03	Continuously transmits packets
TST_TX_CW	0x06	Transmits a Continuous Wave signal.
TST_RX_CONT	0x07	Continuously receiver mode
TST_RSSI	0x08	Measure RSSI on the sub-GHz path
TST_RADIO_RST	0x09	Resets the LoRa Basics Modem-E device
TST_EXIT	0x0B	Exits test mode and resets the device
TST_BUSYLOOP	0x0C	Puts the device in an endless loop with interrupts enabled.
TST_PANIC	0x0D	Causes an unrecoverable fault condition (panic).
TST_WATCHDOG	0x0E	Causes the device to enter an endless loop with interrupts disabled.
TST_TX_SINGLE_PREAM	0x14	Tx single, operation, with configurable number of preamble Bytes.
TST_READ_RSSI	0x15	Read RSSI
TST_RSSI_2G4	0x16	Measure RSSI on the Wi-Fi path
TST_RSSI_GNSS	0x17	Measure RSSI on the GNSS path
TST_READ_NB_PKTS_RX_CONT	0x18	Read the number of packets received in Rx Continuous test.

7.2 Test Commands Details

7.2.1 TST_START

The command *TST_START(...)* starts the test mode. This command enables all other test functions.

Table 7-2: TST_START Command Payload Format

Field	0x00	"TESTTEST"
Length (Bytes)	1	8

7.2.2 TST_NOP

No operation. The command *TST_NOP(...)* can terminate an ongoing continuous TX operation.

Table 7-3: TST_NOP Command Payload Format

Field	0x01
Length (Bytes)	1

7.2.3 TST_TX_SINGLE

The command *TST_TX_SINGLE(...)* transmits a single RF packet on the sub-GHz RF path. The payload are randomly generated.

Table 7-4: TST_TX_SINGLE Command Payload Format

Field	0x02	Frequency	TxPower	SF	BW	CR	Payload Length
Length (Bytes)	1	4	1	1	1	1	1

With SF, BW and CR defined in Table 7-5

Table 7-5: Test Commands Encoding of SF, BW and CR

Value	SF	BW	CR
0	FSK	125 kHz	4/5
1	SF7	250 kHz	4/6
2	SF8	500 kHz	4/7

7.2.4 TST_TX_CONT

The command *TST_TX_CONT(...)* continuously transmits RF packets on the sub-GHz path. The payload are randomly generated.

Table 7-6: TST_TX_CONT Command Payload Format

Field	0x03	Frequency	TxPower	SF	BW	CR	Payload Length
Length (Bytes)	1	4	1	1	1	1	1

With SF, BW and CR defined in Table 7-5

7.2.5 TST_TX_CW

The command *TST_TX_CW(...)* transmits a Continuous Wave signal on the sub-GHz path.

Table 7-7: TST_TX_CW Command Payload Format

Field	0x06	Frequency	TxPower
Length (Bytes)	1	4	1

7.2.6 TST_RX_CONT

The command *TST_RX_CONT(...)* sets the device in continuous Rx mode, for packet reception on the sub-GHz path.

Table 7-8: TST_RX_CONT Command Payload Format

Field	0x07	Frequency	SF	BW	CR
Length (Bytes)	1	4	1	1	1

Only BW=0, BW=1 and BW=2 are allowed.

7.2.7 TST_RSSI

The command *TST_RSSI(...)* measures the RSSI of the received signal on the sub-GHz path.

Table 7-9: TST_RSSI Command Payload Format

Field	0x08	Frequency	Time_ms	BW
Length (Bytes)	1	4	1	1

Only BW=0, BW=1 and BW=2 are allowed.

Table 7-10: TST_RSSI Response

Field	RSSI+64
Length (Bytes)	1

7.2.8 TST_RADIO_RST

The command *TST_RADIO_RST(...)* resets the LoRa Basics Modem-E device.

Table 7-11: TST_RADIO_RST Command Payload Format

Field	0x09
Length (Bytes)	1

7.2.9 TST_EXIT

The command *TST_EXIT(...)* exits test mode and resets the device.

Table 7-12: TST_EXIT Command Payload Format

Field	0x0B
Length (Bytes)	1

7.2.10 TST_BUSYLOOP

The command *TST_BUSYLOOP(...)* puts the device in an endless loop with interrupts enabled. Eventually, the software watchdog will flash the diagnostics LED (if available) and reset the device. Note that this command will render the device completely unresponsive until it is reset..

Table 7-13: TST_BUSYLOOP Command Payload Format

Field	0x0C
Length (Bytes)	1

7.2.11 TST_PANIC

The command *TST_PANIC(...)* causes an unrecoverable fault condition (panic). The device immediately saves a crash-log and halts, the diagnostics LED (if available) flashes for a period, and then the modem resets.

Table 7-14: TST_PANIC Command Payload Format

Field	0x0D
Length (Bytes)	1

NOTE: this command will render the modem completely unresponsive until it is reset.

7.2.12 TST_WATCHDOG

The command *TST_WATCHDOG(...)* causes the device to enter an endless loop with interrupts disabled. Eventually, the hardware watchdog will reset the device. No crash-log is written, diagnostic LED does not flash.

Table 7-15: TST_WATCHDOG Command Payload Format

Field	0x0E
Length (Bytes)	1

NOTE: this command will render the modem completely unresponsive until it is reset.

7.2.13 TST_SINGLE_PREAM

The command *TST_SINGLE_PREAM(...)* performs a TX operation of a single RF packet on the sub-GHz path, with a configurable number of preamble Bytes. The payload are randomly generated.

Table 7-16: TST_SINGLE_PREAM Command Payload Format

Field	0x14	Freq	TxPower	SF	BW	CR	Payload Length	Nbr Of Preamble
Length (Bytes)	1	4	1	1	1	1	1	2

7.2.14 TST_READ_RSSI

The command *TST_READ_RSSI(...)* measures the RSSI of the received signal on the sub-GHz path.

Table 7-17: TST_READ_RSSI Command Payload Format

Field	0x15
Length (Bytes)	1

Table 7-18: TST_READ_RSSI Response

Field	RSSI+64
Length (Bytes)	1

7.2.15 TST_RSSI_2G4

The command *TST_RSSI_2G4(...)* measures the RSSI of the received signal on the Wi-Fi path.

Table 7-19: TST_RSSI_2G4 Command Payload Format

Field	0x16	Channel	Time_ms	BW
Length (Bytes)	1	1	2	1

- Channel: Wi-Fi channel, 1 to 14
- Time_ms
- BW: signal bandwidth:
  - 15: 12MHz
  - 17: 24MHz
  - Other values are RFU

7.2.16 TST_RSSI_GNSS

The command *TST_RSSI_GNSS(...)* measures the RSSI of the received signal on the GNSS path.

Table 7-20: TST_RSSI_GNSS Command Payload Format

Field	0x17	Constellation	Time_ms	BW
Length (Bytes)	1	1	2	1

- Constellation:
  - 0: GPS
  - 1: BeiDou
- Time_ms
- BW: signal bandwidth:
  - 15: 12MHz

Other values are RFU

7.2.17 TST_READ_NB_PKTS_RX_CONT

The command *TST_READ_NB_PKTS_RX_CONT(...)* reads the number of packets received in Rx Continuous test on the sub-GHz path.

Table 7-21: TST_READ_NB_PKTS_RX_CONT Command Payload Format

Field	0x18
Length (Bytes)	1

Table 7-22: TST_READ_NB_PKTS_RX_CONT Command

Field	Nbr of packets
Length (Bytes)	4

8. Interface Modem - Device and Application Services

The Semtech LoRa Cloud™ Device and Application 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 LoRa Cloud™ service.

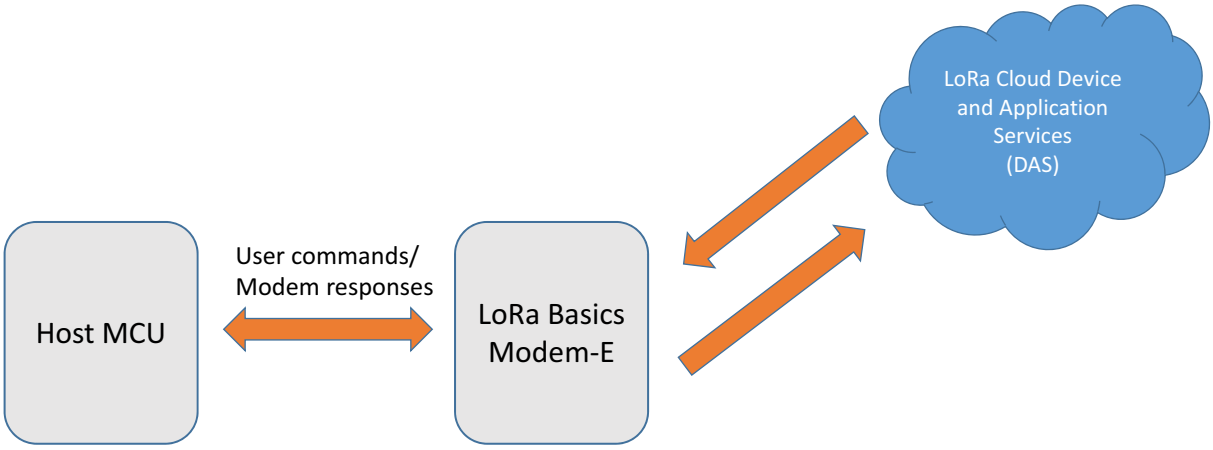


Figure 8-1: Modem Interfaces Overview

8.1 Uplink Messages

8.1.1 Uplink Message Format

All the fields begin with a one byte field tag code and are followed by the field information. The length of most fields is implicit and is defined in the table below. A variable length field can only be the last one (for example the GNSS NAV message, Wi-Fi passive scanning uplinks).

Table 8-1: Modem to DAS Uplink Field Descriptions summarizes the list of uplink command code space and the field information following each code.

Table 8-1: Modem to DAS Uplink Field Descriptions

Name	Code	Description	Payload Size (in Bytes)
Status	0x00	Modem Status	1
Temp	0x03	LR1110 device Junction Temperature (°C)	1
Signal	0x04	Signal strength of the last downlink (RSSI [in dBm]-64, SNR[0.25dB steps])	2
Uptime	0x05	Duration since last reset (h)	2
Rxtime	0x06	Duration since last downlink (h)	2

Table 8-1: Modem to DAS Uplink Field Descriptions

Name	Code	Description	Payload Size (in Bytes)
Firmware	0x07	Firmware CRC and FUOTA progress (completed/total chunks)	4+4
ADRmode	0x08	ADR profile (0-3)	1
JoinEUI	0x09	Join EUI	8
Interval	0x0A	Reporting Interval (Values 0-63, units s/m/h/d)	1
Region	0x0B	Regulatory region	1
Crashlog	0x0D	Crash log data	Variable
Upload	0x0E	Application file fragments	Variable
RstCount	0x0F	Modem reset count	2
DevEUI	0x10	Dev EUI	8
Session	0x12	Session id/join nonce	2
ChipEUI	0x13	Chip EUI	8
Stream	0x14	Data stream fragments	Variable
StreamPar	0x15	Data stream parameters	2
AppStatus	0x16	Application specific status	8
AlcSync	0x17	Application layer clock sync data	Variable
GnssAlmStatus	0x18	Almanac status	7
GnssDbgRsp	0x19	Almanac debug response	Variable
GnssLoc	0x1A	GNSS Scan NAV messages	Variable
WiliLoc	0x1B	Wi-Fi Passive Scanning results	Variable

8.1.2 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 message. The reporting interval can be queried and set using the *GetDmInfoInterval(...)* and *SetDMInfoInterval(...)* modem commands. Additionally it can be set by the DM via the *SetConf* downlink message, 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(...)* and SetDmInfoFields(...) commands, nor by the *GetInfo* downlink request.

8.2 Downlink Messages

8.2.1 Downlinks Format

The format of the DAS downlinks is depicted in Table 8-2.

Table 8-2: DAS Downlinks Format (codes 0x00 to 0x09)

Field	Upcount	Updelay	Request Code	Payload
Length	1 Byte	1 Byte	1 Byte	Variable

Three additional commands have a dedicated format: AlmanacUpdate (0x0A), AlmanacDebug(0x0B) and SolverUpdate(0x0C), as described in Table 8-3, Table 8-5 and Table 8-7.

Table 8-3: DAS Downlink Format for AlmanacUpdate (Code 0x0A)

Field	Upcount	Updelay	Request Code	CRC	Almanac Date	Almanac Update Block 1	Almanac Update Block 2	...
Length	1 Byte	1 Byte	1 Byte (0x0A)	2 Bytes	2 Bytes	Variable	Variable	...

Each of the AlmanacUpdate blocks being defined as in Table 8-4:

Table 8-4: AlmanacUpdate Block Structure

Field	Update Sub Code	Update Payload
Length	1 Byte	Variable

Table 8-5: DAS Downlink Format for AlmanacDebug (Code 0x0B)

Field	Upcount	Updelay	Request Code	Token	Almanac Debug Request Block1	Almanac Debug Request Block2	...
Length	1 Byte	1 Byte	1Byte (0x0B)	1 Byte	Variable	Variable	...

Each of the AlmanacDebugRequest blocks being defined as in Table 8-6:

Table 8-6: AlmanacDebugRequest Block Structure

Field	AlmanacDebug Sub Code	AlmanacDebug Payload
Length	1 Byte	Variable

Each of the SolverUpdate blocks being defined as in Table 8-8:

Table 8-7: DAS Downlink Format for GnssSolverUpdate (Code 0x0C)

Field	Upcount	Updelay	Request Code	SolverUpdate Block1	SolverUpdate Block2
Length	1 Byte	1 Byte	1 Byte (0x0C)	Variable	Variable

Table 8-8: SolverUpdate Block Structure

Field	SolverUpdate Sub Code	SolverUpdate Payload
Length	1 Byte	Variable

8.2.2 Downlink Requests

The table Table 8-9: DAS Downlinks Field Descriptions summarizes the supported downlink requests from DAS to LoRa Basics Modem-E. These downlink commands from DAS are processed directly inside the Modem. No host MCU actions are needed.

Table 8-9: DAS Downlinks Field Descriptions

Request	Code	Description	Payload
Reset	0x00	Reset modem of application MCU	Mode(1) +RstCnt(2)
FileDone	0x02	File upload complete	SessionIdCtr(...)
GetInfo	0x03	Get info fields	Taglist(n) add GNSS almanac status inside
SetConf	0x04	Set Configuration fields	Tag(1) Value(N)
Rejoin	0x05	Rejoin Network	SessCnt(2)
Mute	0x06	Permanently disable/enable Modem	Mute(2)
SetDmInfo	0x07	Set list of default info fields	Taglist(n) add GNSS almanac status inside
Stream	0x08	Set data stream parameter	Param(N)
AlcSync	0x09	Application layer clock sync	Variable
AlmanacUpdate	0x0A	Almanac update, short, long, full update	Variable
AlmanacDbg	0x0B	Almanac Debug	Variable
SolverUpdate	0x0C	Assistance position, Xtal precision	Variable
AlmanacForceUpdate	0x0D	Force partial almanac update	Variable

9. Command Sequences

This section provides guidelines on how to configure the LoRa Basics Modem-E in order to fulfil some specific functions.

9.1 Device Configuration After Reset

After a reset, the LoRa Basics Modem-E device requires the *Join(...)* command to rejoin the network.

Before sending the *Join(...)* command, the host MCU must check and reconfigure the following fields:

- Configure the application hardware:
  - Regulator mode (DC-DC or LDO), using the command *SetRegMode(...)*
  - TCXO, with the command *SetTCXOMode(...)*
  - PA configuration, through the command *ManageRFOutput(...)*
  - RF switch control, using the command *SetDioAsRfSwitch(...)*
- Configure the Tx power offset: *SetTxPowerOffset(...)*
- Configure the Alarm timer if necessary: *SetAlarmTimer(...)*
- Configure the LoRaWAN® Class: *SetClass(...)*
- Check the DM port, and set if needed: *GetDmPort(...)* and *SetDmPort(...)*
- Configure the DM reportig interval: *SetDminfoInterval(...)*
- Configure the DM info fields: *SetDminfoFields(...)*
- Configure the LoRaWAN® Region: *SetRegion(...)*
- Configure the ALC Sync: *SetAlcSyncPort(...)* and *SetAlcSyncMode(...)*
- Configure the connection time-out: *SetConnectionTimeout (...)*

After a reset, time is lost. Therefore after joining, the host MCU should provide the time to the LoRa Basics Modem-E device.

The Host MCU can either use the command *SetTime(...)* if it has knowledge of the time, or it can enable the ALC Sync Service.

9.2 LoRaWAN® Join and Packets Transmission

- Without using Semtech Join Server:
  - Get Device EUI using *GetDevUI(...)* command
  - Register Device EUI on the Network Server
  - Get Join EUI and Application Key from the Network server
  - Set the Join EUI using *SetJoinEUI(...)* command
  - Set the Application Key using *SetAppKey(...)* command

- Using the Semtech Join Server:
  - Get Device EUI using *GetDevUI(...)* command and PIN with *GetPin(...)*
  - Claim the device on Semtech Join Server

    - Make the Modem reset its Application key using *DeriveKeys(...)* command.

- After configuring the keys:
    - Configure the LoRaWAN® parameters:
        - Set the class with the *SetClass(...)* command
        - Set the region with the *SetRegion(...)* command
        - Set the ADR profile using the *SetAdrProfile(...)* command
        - Set the DM info field with the *SetDmInfoFields(...)* command
        - Set the DM info interval with the *SetDmInfoInterval(...)* command
    - Start joining with the *Join(...)* command
    - At this point the Host MCU waits for the event JOIN_ACCEPT
    - Send packets using the *RequestTx(...)* command
    - Waits for the event *TxDone*

9.3 Streaming

- Configure the stream port and encoding with the command *StreamInit(...)*
- Join a network
- Before setting a payload to stream:
    - Check the remaining space in stream buffer with the command *StreamStatus(...)*, the field *Free* must high enough to contain the payload to stream
- Append the buffer to stream with the command *SendStreamData(...)*

The LoRa Basics Modem-E encodes the payload by adding redundant data to ensure robustness against uplink packet loss. It also manages the frame transmissions with respect to the duty cycle and maximum payload size at each transmission. The modem maintains the ROSE protocol with Semtech LoRa Cloud™. The host MCU only needs to send the stream, all the above complexities are managed by the Modem.

9.4 Clock Synchronization

- Join a network
- The Host MCU waits for the *Joined* event
- On the *Joined* event: call the *SetAlcSyncMode(...)* command
- Call the command *GetTime(...)* until it returns something different from 0
- When *GetTime(...)* returns something different from 0, it means the time has been synced and GNSS scanning can be used

9.5 Wi-Fi Passive Scanning

The steps to perform a Wi-Fi passive scanning are the following:

- Reset the cumulative Wi-Fi timing with the command *WifiResetCumulTimingPhase(...)*

-> This step is not mandatory, it allows later on to obtain the time spent in different states of the radio

- Start the Scanning procedure using the command *WiFiPassiveScanMD(...)*
  - Arguments are:
    - Wi-Fi signal type to scan
    - Channel mask to scan
    - Wi-Fi scan mode: beacon or beacon&packets
    - Max results
    - Number of retrials per channel
    - Timeout in ms
    - Flag to indicate if the scanning should jump to next channel when a time-out occurs or not
    - Result format expected

->At this point the Host can wait for the event *Wi-Fi*

- When the event *Wi-Fi* occurs, the Wi-Fi results are contained in the buffer of the event
  - The data format of the Wi-Fi results in the event buffer depends on the selection at step 2.
- Read cumulative timings, with the command *WifiReadCumulTimingPhase(...)*

-> this step is not mandatory: it returns the time spent in different states of the radio during the last scanning procedure

9.6 GNSS Scanning

When performing a GNSS Assisted Scanning, the LoRa Basics Modem-E must already have the GPS time, either by having joined the network with ALC sync feature enabled, or by using the command *GetTime(...)*.

Additionally, the assistance position must have been set issuing the command *GnssSetAssistancePosition(...)*.

- Set the constellation to use, using the command *GnssSetConstellationToUse(...)*
- Set the Assistance position, using the command *GnssSetAssistancePosition(...)*. This is mandatory for an assisted scanning.
- Start the scanning procedure
  - For an assisted scanning: *GnssAssistedMD(...)*
    - Search mode (default or best effort)
    - Bit mask of parameters to report in NAV message:
    - 0x01: Pseudo range
    - 0x02: Doppler
    - 0x04: GNSS bit change
    - Number of satellites
  - For Autonomous scan: *GnssAssistedMD(...)*
    - Search mode (can only be default)
    - Bit mask of parameters to report in NAV message:
    - 0x01: Pseudo range
    - 0x02: Doppler
    - 0x04: GNSS bit change
    - Number of satellites

- At this point the Host can waits or the event *GNSS*
- When the event *GNSS* occurs, the NAV message is in the event buffer
- Details about scanned GNSS satellites can be optionally obtained by issuing commands *GnssGetNbSvDetected(...)* and *GnssGetSvDetected(...)*

9.7 Almanac Update

The Over The Air (OTA) GNSS Almanac update service allows LoRa Basics Modem-E to automatically perform partial almanac updates over the LoRaWAN® network. This ensures that the Almanac data stored in the LoRa Basics Modem-E non-volatile memory performs a GNSS scan with the minimum power consumption without any action on the host MCU side.

The almanac status info field (DM_INFO_TYPE_GNSS_ALMANAC_STATUS) has to be configured in the device management uplinks (using the command *SetDmInfoField(...)*) in order to communicate the almanac status to Semtech LoRa Cloud™ Device, and therefore allow the OTA GNSS Almanac update.

The latest almanac data is stored in the device memory during Semtech production flow. In the case of an important delay (typically months) between the production flow and the first connection to the LoRaWAN® network, it is likely that the almanac will be outdated. It is then advised to perform a full almanac update to avoid numerous partial updates.

10. List Of Commands

10.1 System/ Register / Memory Operations

Table 10-1: Register / Memory Access Operations

Command	Opcode	Input	Output	Description	Event
WriteRegMem32	0x0105	Addr (31:0) Data (1..256)	--	Writes data at given register/memory address. Address must be 32 bit aligned and data length must be a multiple of 4.	No
ReadRegMem32	0x0106	Addr (31:0) Len (7:0)	--	Reads data at given register/memory address. Address must be 32-bit aligned and data length in words (32-bit) < 65	No

10.2 System Configuration / Status Operations

Table 10-2: System Configuration / Status Operations

Command	Opcode	Input	Output	Description	Event
SetRegMode	0x0110	RegMode (0 = LDO, 1 = DC-DC)	--	Sets if DC-DC may be enabled for XOSC, FS, RX or TX mode	No
SetDioAsRfSwitch	0x0112	RfswEnable (7:0), RfswStbyCfg (7:0), RfswRxCfg (7:0), RfswTxCfg (7:0), RfswTxHPCfg (7:0), RfswTxHFCfg (7:0), RfswGnssCfg (7:0), RfswWifiCfg (7:0)	--	Setup the RFSWx outputs configurations for each radio mode	No
ConfigLFclock	0x0116	LfClockSetup (7:0)	--	Configures the used LF clock	No
SetTcxoMode	0x0117	Tune (7:0) Delay (24:0)	--	Configure the device for a connected TCXO	No
Reboot	0x0118	StayInBootloader	--	Allows FW update and reboots the device (SW reset). 0=Stay in bootloader for firmware update	No

10.3 Wi-Fi Configuration / Status Operations

Table 10-3: Wi-Fi Scanning Configuration / Status Operations

Command	Opcode	Input	Output	Description	Event
WifiResetCumul TimingPhase	0x0307	---	--	Initializes cumulative times per phases for power consumption measurements	No
WifiReadCumul TimingPhase	0x0308	---	--	Returns cumulative time per phase for power consumption measurements	No
WiFiPassiveScanMD	0x0330	---	--	Launches a Wi-Fi passive scanning	Yes

10.4 GNSS Configuration / Status Operations

Table 10-4: GNSS Scanning Configuration / Status Operations

Command	Opcode	Input	Output	Description	Event
GnssSet ConstellationToUse	0x0400	ConstellationBitMask	---	Sets the GNSS constellation to use for the GNSS Scanning	No
GnssAlmanac FullUpdate	0x040E	AlmanacFullUpdate Payload	---	Updates all the Almanac Data	No
GnssSet Assistance Position	0x0410	Latitude Longitude	---	Configures the approximate position for the GNSS Assisted mode	No
GnssGetNbSv Detected	0x0417	---	---	Returns the number of SV detected during the last GNSS Scanning	No
GnssGetSv Detected	0x0418	---	---	Returns the list of SV detected during the last GNSS Scanning, with their C/N0	No
GnssGet Consumption	0x0419	---	---	Returns the radio capture and CPU processing duration of the last GNSS Scanning	No
GnssAutonomousMD	0x0430	EffortMode ResultMask NbSvMax	---	Triggers the GNSS Autonomous Scanning	Yes
GnssAssistedMD	0x0431	EffortMode ResultMask NbSvMax	---	Triggers the GNSS Assisted Scanning	Yes

10.5 Modem Configuration / Status Operations

Table 10-5: Modem Configuration / Status Operations

Command	Opcode	Input	Output	Description	Event
GetEvent	0x0600	---	--	Retrieves pending events	No
GetVersionMD	0x0601	---	--	Returns Firmware version	No
Reset	0x0602	---	--	Resets the Modem	Yes
GetTxPowerOffset	0x0606	--	TXOffset	Returns the Tx power correction offset	No
SetTxPowerOffset	0x0607	TXOffset	--	Sets the Tx power correction offset	No
Test	0x0608	*Test Function specific*	*Test Function specific*	Radio Test Function	No
GetTime	0x060A	--	Timestamp (3:0)	Get current modem time, synchronized on network's GPS time	No
GetStatus	0x060B	--	Status	Returns device status	No
SetAlarmTimer	0x060C	Seconds (3:0)	--	Sets alarm/wakeup timer	Yes
GetPin	0x060E	--	PIN (3:0)	Returns device registration PIN	No
GetChipEui	0x060F	--	ChipEUI (7:0)	Returns device chip EUI	No
GetJoinEui	0x0610	--	JoinEUI (7:0)	Returns Join EUI	No
SetJoinEui	0x0611	JoinEUI (7:0)	--	Sets Join EUI without deriving keys	No
GetDevEui	0x0612	--	DevEUI (7:0)	Returns Device EUI	No
SetDevEui	0x0613	DevEUI (7:0)	--	Sets Device EUI without deriving keys	No
SetAppKey	0x0614	AppKey (15:0)	--	Sets the application key	No
GetClass	0x0615	--	Class	Returns the LoRaWAN® device class	No
SetClass	0x0616	Class	--	Sets the LoRaWAN® device class to A or C	No
GetRegion	0x0618	--	Region	Returns the regulatory region	No
SetRegion	0x0619	Region	--	Sets the regulatory region	No
ListRegions	0x061A	--	Regions (n-1:0)	Lists the supported regulatory regions	No
GetAdrProfile	0x061B	--	Type	Returns the ADR Profile	No
SetAdrProfile	0x061C	Type, List (15:0)	--	Sets the ADR profile and the optional parameters	No

Table 10-5: Modem Configuration / Status Operations

Command	Opcode	Input	Output	Description	Event
GetDmPort	0x061D	--	Port	Returns the DM port	No
SetDmPort	0x061E	Port	--	Sets the DM port	No
GetDmInfoInterval	0x061F	--	Interval	Returns the DM reporting interval	No
SetDmInfoInterval	0x0620	Interval	--	Sets the DM reporting interval	No
GetDmInfoFields	0x0621	--	Inflict (n-1:0)	Returns the default info fields for the DM status	No
SetDmInfoFields	0x0622	Inflict (n-1:0)	--	Sets the default info fields for the DM status	No
SendDmStatus	0x0623	Inflict (n-1:0)	--	Sends the DM status now	No
SetAppStatus	0x0624	AppStatus (7:0)	--	Sets the application-specific status for the DM	No
Join	0x0625	--	--	Starts joining the network	Yes
LeaveNetwork	0x0626	--	--	Leaves the network	No
SuspendModem Comm	0x0627	Suspend	--	Suspends/Resumes radio operations	No
GetNextTxMax Payload	0x0628	--	Size	Returns the maximum payload size for the next Tx	No
RequestTx	0x0629	Port, Conf, Data (n-1:0)	--	Transmits the frame unconfirmed or confirmed	Yes
EmergencyTx	0x062A	Port, Conf, Data (n-1:0)	--	Transmits the frame immediately (e.g. smoke alarm), unconfirmed or confirmed	Yes
UploadInit	0x062B	Port, EncryptMode, Size (1:0), Delay (1:0)	--	Sets the file upload port the encryption mode, the payload size	No
UploadData	0x062C	Data (n-1:0)	--	Writes data for file upload transmission	No
UploadStart	0x062D	FileCrc (3:0)	--	Verifies data and starts the file upload	Yes
StreamInit	0x062E	Port, Mode	--	Sets the data stream parameters	No
SendStreamData	0x062F	Port, Record (n-1:0)	--	Sends the data stream record	Yes

Table 10-5: Modem Configuration / Status Operations

Command	Opcode	Input	Output	Description	Event
StreamStatus	0x0630	Port	Pending (1:0), Free (1:0)	Retrieves the stream status	No
GetCmdRspSize	0x0631	Cmd	ByteLength	Reads the application payload size of response of a read command	No
SetTime	0x0632	Time (3:0)	--	Sets the GPS Time	No
GetEventSize	0x0633	--	ByteLength (1:0)	Returns the event size	No
DeriveKeys	0x0634	--	--	Uses the previously set of JoinEUI/DevEUI to derive the LoRaWAN® keys	No
ManageRFOutput	0x0636	Config	--	Configures the RF output configuration	No
SetAlcSyncPort	0x0637	Port	--	Sets the port for Application Layer Clock Synchronization	No
GetAlcSyncPort	0x0638	--	Port	Returns the port for Application Layer Clock Synchronization	No
SetAlcSyncMode	0x0639	Mode	--	Enables/disables ALC Sync service	No
GetAlcSyncMode	0x063A	--	Mode	Gets Alcsync service mode	No
SetConnection Timeout	0x063C	NbUplink(3:0)	--	Sets the number of uplink without downlink from network before Modem resets and change ADR profile	No
GetConnection Timeout	0x063D	--	NbUplink(1:0)	Gets the number of uplink without downlink from network before Modem resets and change ADR profile	No
SetCertificationMode	0x063E	Mode	--	Enables/disables Set LoRaWAN certification mode	No
SetCertificationMode	0x063F	--	Mode	Gets LoRaWAN certification mode	No
GetLRWANState	0x0640	--	LoRaWANState	Returns the state of the LoRaWAN® stack	No
GetStatusDutyCycle	0x0644	--	DutyCycleStatus (3:0)	Gets the LoRaWAN® stack radio transmit duty cycle	No

11. Revision History

The following table details the versions of the Reference Manual document issued, and the corresponding LoRa Basics Modem-E versions supported (Use Case and FW Major.FW Minor), as returned by the command *GetVersionMD(...)*.

Table 11-1: Revision History

Ref. Manual Version	ECO	Date	Applicable to	Changes
1.0	053305	Sep 2020	Use Case: 02 FW Version: 1.0.5 or later	Initial Release

Glossary

List of Acronyms and their Meaning

Acronym	Meaning
ADR	LoRaWAN® Adaptive Data Rate
ALC	Application Layer Clock Synchronization (ALC Sync)
BW	LoRa® modulation Bandwidth
CR	LoRa® modulation Coding Rate
DM	Device Manager
DR	Datarate
FOpt	LoRaWAN® Frame Options
FUOTA	Firmware Update Over The Air
LoRa®	Long Range Communication the LoRa® Mark is a registered trademark of the Semtech Corporation
RSSI	Received Signal Strength Indicator
RFU	Reserved for Future Use
LoRa Cloud	Semtech LoRa Cloud™ available on www.loracoud.com
SF	LoRa® modulation Spreading Factor

Contact Information

Semtech Corporation
Wireless, Sensing & Timing Products Division
200 Flynn Road, Camarillo, CA 93012
Phone: (805) 498-2111, Fax: (805) 498-3804
www.semtech.com