Reference Guide

AT3 application

1.14

Revisions

Version	Primary Author	Description of Version	Date Completed
1.0	Patrick Beatini	Initial draft	April 5th 2024
1.1	Patrick Beatini	Actual queue used for core events Add BLE_ASSERT reset cause Belong to candidate 1.0-193	April 30th 2024
1.2	Patrick Beatini	Correct various error Configuration parameter have changed. Payloads have changed. Parameter name changed	May 28h 2024
1.3	Patrick Beatini	Add the second BLE scan. This impacts: • Configuration (renumbering) • Uplink position payloads Add the tamper support (new system notif). Release 1.0-195	June 17h 2024
1.4	Patrick Beatini	Correct error in cellular group Rework the almanac upload via the network Update almanac management Add system parameter for almanac management	August 13h 2024
1.5	Patrick Beatini	Add the UI (User Interface) management (chapter 18.) Modify the system configuration Enhance the button configuration	
1.6	Hamza Saadi	Add section 9.1.11 Update active state in section 5.2 Update section 15.4.10 Update section 4.1.3 (classes Button, geoloc) Update section 5 (Remove on-hold state) Correct section 9.1.6 (max offset) Update section 16.3.5	October 4th 2024
-	Stéphane Boudaud	Update Section 6.1 with additional hardware parameters	October 4th 2024
-	Hamza Saadi	Correct altbeacon data size in section 9.1.11	October 7th 2024
-	Patrick Beatini	Clean up	October 8th 2024
-	Hamza Saadi	Update section 4.1.3, Add class 11 Update section 16.2, add section 16.2.3	October 10th 2024
-	Patrick Beatini	Add motion config parameters for GNSS Add cell_rai_timeout config parameter for cell. Add cell_seach_bands parameter for cell+ note	October 14th 2024
-	Patrick Beatini	Add confirmed message configuration in the LORAWAN group. Replace the parameters core_notif_enable1 and core_notif_enable2 [integer] by a single one [byte-array]	October 15th 2024

Version	Primary Author	Description of Version	Date Completed
-	Hamza Saadi	Update section 6.1	October 22nd 2024
1.7	Patrick Beatini	Release the doc to follow 1.0-197	October 23nd 2024
	Hamza Saadi	Update section 9.1.11 (ble_cnx_init_config -> ble_cnx_behavior)	October 31st 2024
	Patrick Beatini	Add the motion counter in the UL position header (section 15.4.1)	November 6th 2024
	Hamza Saadi	Section 9.1.2: Update parameters id, description and notes, update 8th event type (Start connectivity) section 18.1.2: updates related to core_buttonX_map and core_buttons_timing parameters.	November 13th 2024
1.8	Patrick Beatini	Release the doc to follow 1.0-197	November 13th 2024
1.9	Olivier Hersent	Merge with previous edits	November 17th 2024
1.10	Patrick Beatini	Add the low battery management Correct formatting Add low battery management Add the type column in configuration tables.	December 2nd 2024
-	Patrick Beatini	Add the config CRC in the status uplink page 0 Add the CRC field in the Response to a parameter class configuration set request. Add the config CRC request and response. Add the **Annex A** for configuration CRC calculation	December 3rd 2024
-	Patrick Beatini	Add the new BLE scan parameter ble_scan_min_beacons+ re-indexing WIFI configuration: • Rename parameter lr_wifi_max_bssid to lr_wifi_report_nb_bssid. • Add lr_wifi_min_nb_bssid parameter • Add lr_wifi_min_rssi parameter	December 5th 2024
-	Patrick Beatini	Provisioning. Buzzer volume = 0 => Buzzer disabled	December 9th 2024
-	Patrick Beatini	Button press/long press duration configuration (system profile). Long press is a delay from the button press. Duration and delay expressed in 0.5 second unit.	December 10th 2024
-	Patrick Beatini	Correct typo enumeration of the cell parameter	December 10th 2024
-	Patrick Beatini	Add new query and its response: echo-request & echo-reply. Add new cell parameters + renumbering. Update the cell network management section	December 11th 2024
-	Hamza Saadi	Update section 9.1.2: Add new parameter cli password	January 8th 2025

Version	Primary Author	Description of Version	Date Completed
-	Ali Chouchene	Add a new LR11XX parameter for BSSID administration type reporting. Update the default value of cellular eDRX parameters and add the related bit mapping.	January 10th 2025
-	Hamza Saadii	Section 9.1.7: Update motion sensitivity range, add note.	January 13th 2025
-	Patrick Beatini	Update the CRC calculation (Annex A) Add the CRC in the answer of a global configuration set (ulpink).	January 14th 2025
-	Ali Chouchene	System time update procedure and clock drift correction.	January 17th 2025
-	Patrick Beatini	Add comments on parameter group internal. Add a DL command to reset motion percentage Add a DL request to read sensor value and the associated UL response	January 17th 2025
-	Patrick Beatini	Update default parameters	January 27th 2025
1.11	Patrick Beatini	Candidate 1.0.200	January 27th 2025
-	Patrick Beatini	Reduce the almanac entry in the response and the command sent by the network. Add explanations about the almanac update process	January 29th 2025
-	Patrick Beatini	Put the CRC on 4 bytes in the answer of a generic set request	February 5th 2025
-	Saadi Hamza	- In section 15.3.1 add Heartbeat type in system class. - In section 9.1.2 correct class range [1 to 4] become [0 to 5]. - In section 4.1.3 Update class 9, add events types vbus_on vbus_off and dl_trigger. - In section 4.1.3 Update class 0/1: Correct numbering and add missing event. - In section 4.1.3 Update class 3: Add missing event clear motion. - In section 4.1.3 Update class 6: Add missing event geo complete and correct numbering. - In section 4.1.3 Update class 8: Add missing event cellular_lpm_changed. - In section 4.1.3 Update class 10: correct numbering.	February 26th 2025
-	Saadi Hamza	- Update section 11.3.1: mode "Moving and used" become "Techno used and moving", update values description and add note 2. - Update section 9.1.3: update geoloc_gnss_hold_on_mode range and description (remove values description and reference the section 11.3.1 for the values) - Update section 9.1.10: Update min value for	February 27th 2025

Version	Primary Author	Description of Version	Date Completed
		cell_cnx_timeout_motion and ell_cnx_timeout_static	
-	Ali Chouchene	Add the joining mode related bit in the provisioning parameters. Update the main FSM section to include the joining mode.	March 5th 2025
-	Ali Chouchene	Update default parameters	March 13th 2025
1.12	Patrick Beatini	New version start	March 20th 2025
-	Saadi Hamza	Rename Parameter 0x0A05	March 24th 2025
1.13	Patrick Beatini	New version start	April 1st 2025
-	Saadi Hamza	Section 4.1.3: Update class 9 and class 10 Section 9.1.11: Set min value for ble beacon fast_adv_interval and slow_adv_interval to 40ms	April 2nd 2025
–	Patrick Beatini	Add section 16.2.1 (reset to factory default DL command)	April 23th 2025
1.14	Patrick Beatini	New version start	May 6, 2025

Table of Contents

Introduction

AT3 is version 3 of the Abeeway Asset-tracker multi-technology geolocation application, the successor of AT2.

1.1 Document Purpose

The document covers all technical design aspects to the firmware. It particularly addresses:

1. The design of the application
2. The dynamic behavior of the tracker
3. The configuration and the user interface
4. The LoRaWAN application payloads.

1.2 Intended Audience and Document Overview

The intended audience for this document includes the product, support and engineering teams.

1.3 Definitions, Acronyms and Abbreviations

The terms *Software* and *Firmware* are used interchangeably in this document.

Acronym	Description
AT3	Assert-tracker 3 application firmware
BLE	Bluetooth Low Energy
LoRaWAN	Low Range Radio protocol standardized by the LoRa Alliance
MTOS	Micro-tracker Operating system (EFM32)
AOS-SDK	Abeeway Operating system, Software Development Kit.
CLI	Command Line Interface
UI	User interface (LED, Buzzer, Buttons).
LID	16-bit parameter Local Identifier
FID	32-bit parameter Full Identifier
FQDN	Fully Qualified Doman Neme
RFU	Reserved for Future Use

2 AT3 versus AT3 feature set

2.1 Legacy AT2

The legacy AT2 firmware supports:

- **Basic features**
  - Single button management (clicks, presses and sequences, configurable actions)
  - Two LEDs management (patterns)
  - Buzzer management (several melodies are supported)
  - CLI interface (Support multiple commands, two access levels)
  - Log facility (Per module log)
  - Accelerometer management (Motion and shock detection)
  - Temperature management (Permanent storage of the min/max values, actions taken when min/max thresholds reached)
  - Battery management (Primary vs rechargeable batteries, remaining capacity measurement, estimated consumption.
  - Configuration parameters management (Permanent storage, Configuration files, Parameter name parser)
  - Startup modes (manufacturing, shipping, LoRa joining)
  - Firmware update over USB.
- **LoRa features**
  - Class A only
  - LoRa regions support (EU, AS, IN, AU, US)
  - Network access: ABP vs OTAA.
  - Network join management
  - Network monitoring (Link check, Reset after no answer)
  - Multiple transmissions strategies (configurable ADR, Dual transmissions, ...)
  - Uplink queue (Up to 8 messages)
  - Downlink messages processing
  - Configurable heartbeat (LoRa live)
  - Configurable confirmed uplinks.
- **Geolocation/collections**
  - Multiple geolocation schemes (gps-only, agps-only, wifi-only, ble-only, wgps, wagps, bgps, bagps).
  - Multiple collection schemes following the geolocation (ble or wifi)
  - BLE position filtering (beacon type, filters)
  - BLE collection filtering (beacon type, filters)
  - Configurable timing for geolocation (timeout, LoRa reporting)
- **Operational modes**
  - Multiple behavioral modes (off, standby, tracking on motion, tracking on start/stop, permanent tracking, activity tracking)
  - Configurable timings
  - Position on demand (using another geolocation scheme than the mode's one)
  - Extra periodic positions (in addition to the mode's one using another geolocation scheme)
- **BLE geozoning**
  - Multiple area detection (entry, exit, safe, hazardous)
  - LoRa reporting (zone type events, beacon collection reporting)

    - Configurable filters
- **SOS support**
    - Switch to Fast tracking mode
    - Specific UI
    - SOS tagged LoRa uplinks.
- **BLE connectivity**
    - Single bonding support
    - Passkey authentication support
    - Export standard characteristics (temperature, battery, …)
    - Export vendor specific characteristics (parameters, mode change, …)
    - Notification support (send notifications to mobile phone)
    - BLE firmware update over the air
    - MCU firmware update over the air
    - CLI over BLE
    - Find me support
    - Tracker stops geolocation/collection when securely connected.
    - Behavioral profile support
- **BLE beaconing support**
    - Multiple beaconing type support(Eddystone, Ibeacon, Qquppa, alt-beacons)
    - Advertisement data configurable (up to five 32bits word available)
    - TX power configurable.
- **Angle detection**
    - Multiple modes (critical-angle, shock, angle-deviation)
    - Multiple reference vector acquisition modes (manual, configured, automatic, assisted)

2.2 AT3 feature set

Most of the legacy AT2 features will be supported by the AT3 firmware. While some features will be supported as-is (without any behavioral/configuration changes), some of them will be removed (as they are never used) or enhanced to provide more flexibility.

Unsupported features:

- LoRaWAN ABP activation (no plan to support ABP in AT3, ABP is considered obsolete)
- BLE geozoning (will be supported in a future release)
- Angle detection (will be supported in a future release)

New features:

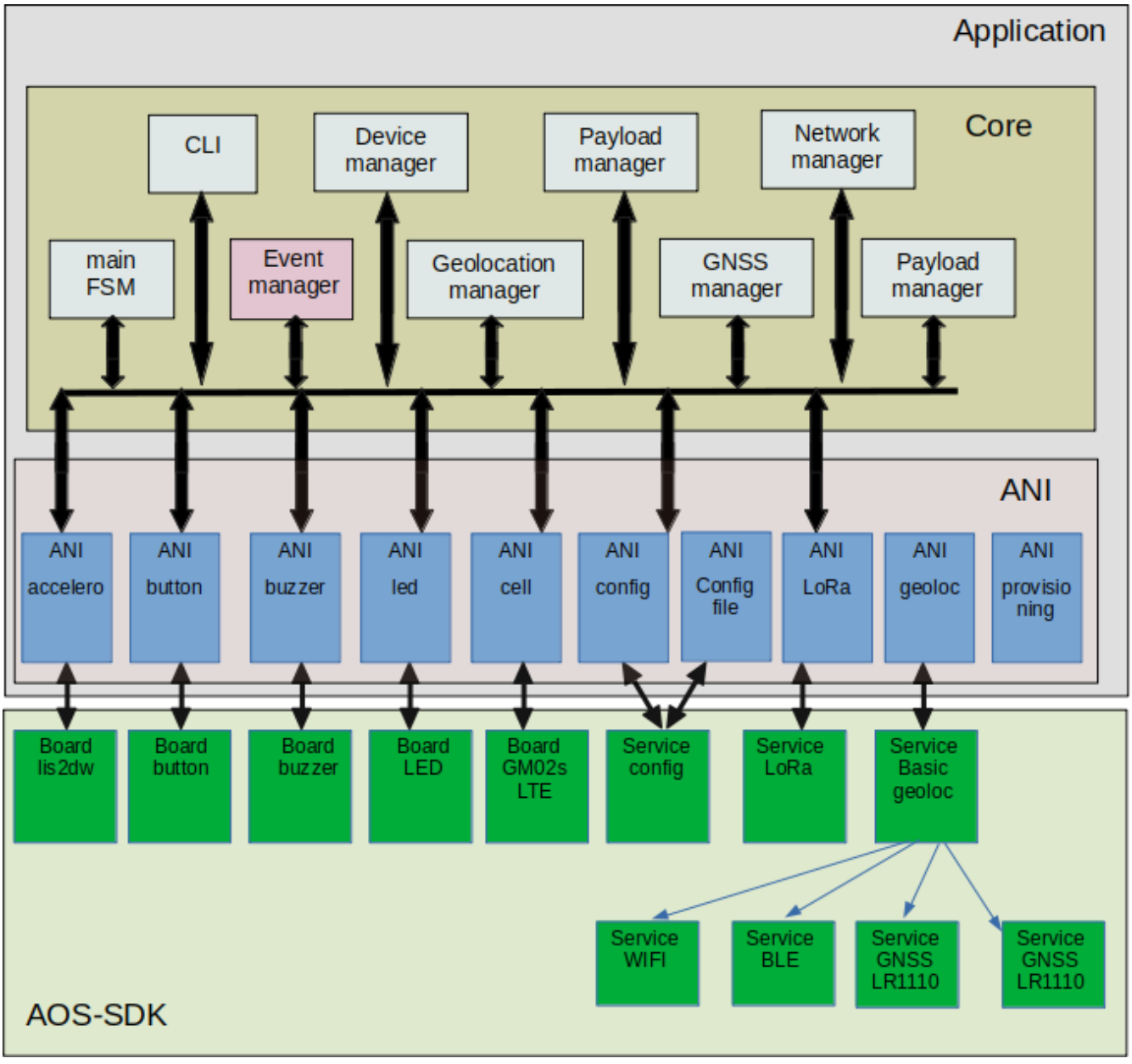
- LoRaWAN class B

TO BE COMPLETED

3 Application design

3.1 Overview

The application design can be pictured as follow.



The application is structured in two layers:

- ANI (Application Node Interface). It performs the link between the application and the underlying service. It usually controls a single service. It registers against the associated service callback and transforms the service events to application events. Note that an ANI can also register against the event manager to catch specific events and perform dedicated action regarding the service it manages.
- Core. This is the main part of the application. It includes:
  - A main FSM (Finite State Machine), which manages the system state (off, running hold, and so on).
  - An event manager, which distributes the application events to its registered clients.
  - A geolocation manager, which schedules the position acquisitions.
  - A GNSS manager, which manages the almanacs and aiding positions.
  - A network manager, which handles the networking part using LoRa and LTE.
  - A payload manager, which creates unsolicited uplink messages (position, notifications).
  - A device manager, which monitors the heath of the system (temperature, battery level, …).
  - A CLI allowing the configuration and the monitoring of the system.

The application runs in a single thread (a Free RTOS task). Application events are handled within this thread. The actions taken by the application are always executed under the application thread. This mechanism enforces the synchronization required for a stable application in a multi-threaded system.

4 Event manager

The event manager handles the application events. Any part of the application can register with the event manager to receive these events.

Each application event is composed of a class and a type. A class represents a set of events related to a specific functionality, while a type refers to a specific event within the class.

The event manager maintains an event queue (a FreeRTOS queue), that can hold up to 40 events.

The events are distributed one by one to the software components registered with the event manager.

Note that the registration can be done for one or multiple event classes.

4.1.1 Event classes

A software component registers with the event manager for one or several classes of events.

4.1.2 Event types

The event types belong to a class. Each class has its own event types.

4.1.3 Current event classes and types

Currently, the AT3 application supports the following classes and types

Class 0 & 1. **Button_1** (0) & **Button_2** (1)		
Event type	ID	**Comments**
type_btn_press	0	Button press
type_btn_long_press	1	Button long press
type_btn_1_click	2	Single click
type_btn_2_clicks	3	Double clicks
type_btn_3_clicks	4	Triple clicks
type_btn_simple_seq	5	Button simple sequence has been entered.
type_btn_seq_reset	6	Button reset sequence has been entered.
type_btn_seq_bond_del	7	Button bond delete sequence has been entered.
type_btn_seq_ble_boot	8	Button BLE bootloader sequence has been entered.
type_btn_ble_connectivity	9	Button event to start connectivity.

Note that the very long press event (14s+) and the detection of special command sequences is implemented by the button driver and not available for customization by the application, hence these special command actions are not reported as events.

Class 2. **Buzzer**		
Event type	**ID**	**Comments**
type_buzzer_on	0	The buzzer is playing
type_buzzer_off	1	The buzzer is no more playing

Class 3. **Accelerometer**		
Event type	**ID**	**Comments**
type_acc_motion_start	0	Motion start detected.
type_acc_motion_end	1	Motion end detected (after motion duration timeout).
type acc_shock_detected	2	A shock has been detected.
type_acc_clear_motion	3	A clear motion event received.

Class 4. **Power**		
Event type	**ID**	**Comments**
type_power_battery_low	0	Critically low battery level detected
type_power_charging	1	Battery is charging
type_power_not_charging	2	Battery is no more charging

Class 5. **Temperature**		
Event type	**ID**	**Comments**
type_temp_normal	0	Temperature back to normal (within minimum and maximum thresholds)
type_temp_low	1	Temperature fell below the minimum threshold
type_temp_high	2	Temperature increased above the maximum threshold
type_temp_update	3	Temperature updated

Class 6. **Geolocation**		
Event type	**ID**	**Comments**
type_geoloc_start	0	Geolocation has started

type_geoloc_done	1	Geolocation is done
type_geoloc_complete	2	Geolocation engine completed an acquisition.
type_geoloc_abort	3	Geolocation has been aborted
type_geoloc_timeout_acq	4	Geolocation timeout occurred.
type_geoloc_timeout_hold	5	GNSS hold mode timeout occurred.
type_geoloc_geozoning_in	6	Geozoning indoor event has been raised.
type_geoloc_geozoning_out	7	Geozoning outdoor event has been raised.

Class 7. **Configuration**	
Event type	**Comments**
Bitmap of config group	A configuration change has occurred. Impacted groups are in the bitmap (encoded as 1<<groupID). This event is managed internally to AT3.

Class 8. **Network**		
Event type	**ID**	**Comments**
type_network_main_up	0	Main network up.
type_network_backup_up	1	Backup network up.
type_network_down	2	Network is down. No connectivity at all
type_network_hold	3	Network is temporarily not available
type_network_pipe1_up	4	Data pipe 1 up. Traffic accepted on this pipe.
type_network_pipe1_down	5	Data pipe 1 down. Traffic refused on this pipe.
type_network_pipe1_rx_data	6	Data pipe 1 has available RX data
type_network_cellular_lpm_changed	7	`Cellular network lpm changed`

Class 9. **Core**		
Event type	**ID**	**Comments**
type_core_app_init_done	0	Application has ended initialization
type_core_off	1	Core application moved to the off state
type_core_skip_off	2	Core was expected to move in off mode but the configuration prevents it
type_core_running	3	The core application is running

type_core_hold	4	The core application moved to the hold state. Reserved for future use.
type_core_time_update	5	The system time has been updated
type_core_device_start	6	A start action for the core has been issued
type_core_device_stop	7	A stop action for the core has been issued
type_core_sos_start	8	A SOS start action has been issued
type_core_sos_stop	9	A SOS stop action has been issued
type_core_status_ul	10	The sending of a status uplink has been requested
type_core_monitoring	11	The timeout of the monitoring timer has elapsed
type_core_pod	12	A position on demand has been requested
type_core_refresh_alm	13	Almanac refresh
type_core_tamper_on	14	Tacker casing is open
type_core_tamper_off	15	Tacker casing is closed
type_core_gnss_timeout	16	GNSS core timeout
type_core_battery_level	17	Show battery level
type_core_vbus_on	18	VBUS connected
type_core_vbus_off	19	VBUS disconnected
type_core_dl_trigger	20	Send HeartBeat message to trigger downlink
type_core_join_to_activate	21	This event is sent when the tracker must join LoRa before being activated (meaning fully running).

Class 10. **BLE connectivity/beaconing**		
Event type	**ID**	**Comments**
type_ble_conn_start	0	Start advertisement for connectivity
type_ble_conn_stop	1	Stop connectivity
type_ble_conn_idle	2	No connection nor advertisement
type_ble_conn_fast_adv	3	Fast advertisement for connectivity
type_ble_conn_slow_adv	4	slow advertisement for connectivity
type_ble_connected	5	BLE is in the connection state
type_ble_conn_link_lost	6	Connection lost.
type_ble_bonded	7	BLE is securely connected (bonded).

type_ble_beacon_start	8	BLE beacon started
type_ble_beacon_stop	9	BLE beacon stopped
type_ble_reduce_cnx_speed_rqst	10	Request the BLE to slow down connection parameters to let the app to save config
type_ble_reduce_cnx_speed_done	11	BLE connection parameters slowed down
type_ble_restore_cnx_speed_rqst	12	Request the BLE to restore connection parameters after the configuration saved

Class 11. **User**		
Event type	**ID**	**Comments**
core_event_type_user_1	0	User event 1
core_event_type_user_2	1	User event 2
core_event_type_user_3	2	User event 3
core_event_type_user_4	3	User event 4
core_event_type_user_5	4	User event 5

5 Main application FSM

5.1 Overview

The main FSM controls the application. It manages the:

- Manufacturing tests if any (according to provisioned flags, see section Error: Reference source not found). Manufacturing tests are performed only once during the product lifetime.
- Shipping mode: The tracker is waiting for a button long press to leave this mode. It is entered only once during the lifetime of the product.
- Joining mode: The device is sending join requests to join the network, and the BLE is activated with its corresponding configuration. It is entered only once during the product lifetime.
- Off mode: The device is waiting for a button action to start the device. Unlike Shipping mode, the device can enter OFF mode repeatedly.
- Hold mode: All activities except the BLE connectivity are suspended. This mode is not used in the application in the current version, it is expected to be used with the companion mobile app for power saving when the tracker is BLE connected and the position is reported from the mobile app.

5.2 Functional State Machine

The main FSM manages the different states of the tracker, which are:

- **init**: The system is initialized and waits for a start event. This start is automatic if the manufacturing and the shipping sequences have been already performed (as recorded in the flash memory flags). Otherwise a button long press is required to trigger the start. A start will move the FSM in the state **startup**.
- **Startup (**substates**: *manufacturing_test, shipping*)**. In the Startup state the manufacturing and the shipping stages are executed if they are configured in the device factory provisioning configuration. A flag in flash memory is allocated to each of these sub-states and cleared once the corresponding actions have been completed, so they will run only once. Once the startup process is complete, the FSM moves to the **off** state if allowed by the configuration. Otherwise, it moves to the **active** state.
- Join-then-activate: This state is entered only when the joining mode is enabled by setting the related bit in the provisioning parameters. Once joined, the FSM moves to the **active** state if there is no device start event mapped to the buttons; otherwise, it moves to the **off** state and waits for a button trigger.
- **off**. The tracker is in deep sleep mode. No communication is allowed (LoRaWAN, LTE, BLE). The periodic system monitoring is also suspended. All peripherals (accelerometer, barometer, …)

are stopped. To move to the **active** state, a button action is required.

- **active**. The network (LoRaWAN or LTE) is started. The periodic system monitoring is active and the peripherals are running. BLE connections are also possible.

- **on-hold.** (not used for now) This state is designed to minimize power consumption when the device is paired with a phone and the mobile application takes over geolocation and communication with the back-end location engine. In this state, the periodic system monitoring as well as the peripherals are active. However the position acquisitions and the LoRaWAN/LTE network communications are suspended to preserve the battery. This state is not entered automatically, the application logic is responsible for entering on-hold state and leaving on-hold state back to active state.

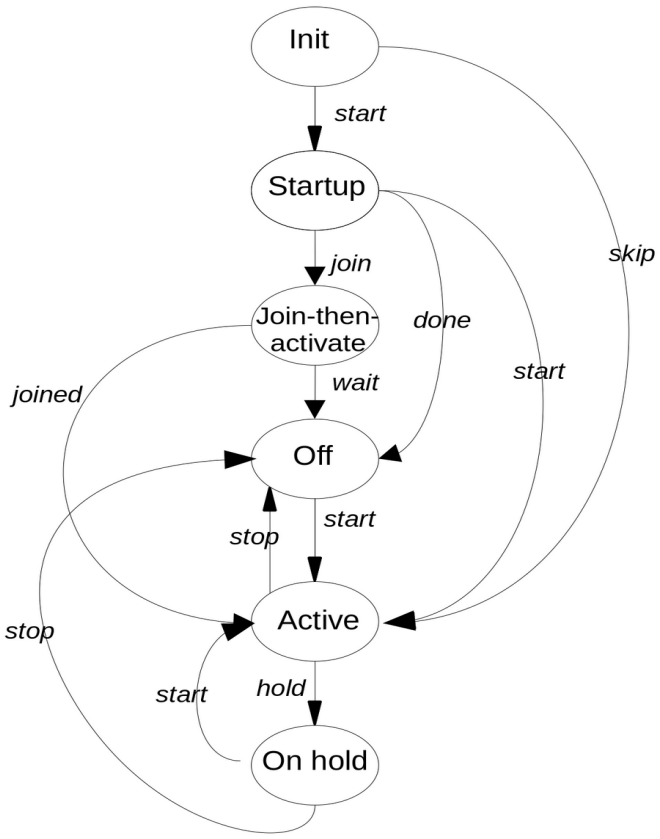
The processed events are:

- **start**: Sent to trigger the transition to the next state as per FSM transition diagram below.
- **skip**: Sent to move directly to the active state (usually sent when the device restarts due to an error).
- **stop**: Stop the FSM. Usually used to move to the **off** state,
- **done.** Sent when the startup is complete.
- **Join**. Sent once startup is complete and the joining mode in enabled.
- **Joined**. Sent once the device has joined the network and there is no device start event mapped to the buttons.
- **Wait**. Sent once the device has joined the network and the device start event is mapped to the buttons.
- **hold**: (not used for now) Sent to enter the on-hold state (under application control, e.g. when the tracker is securely connected via BLE and a mobile application takes over geolocation).

Note

- The Off state is entered after the startup completion only if one of the two button maps contains the **device_start** event.
- Once active, the Off state can be entered only if one of the two button maps contains the **device_stop** event.

The following picture outlines the FSM state transition. Only the relevant events are represented.



Note that the main FSM registers to the system power events. The behavior is the following:

On reception of a ***type_power_low_battery*** system power event (generated by the power manager), the AT3 state machine moves to the off state and sends a core_off system event.

- In the case of a tracker equipped with primary batteries, the main FSM starts a timer with a timeout of 6 hours (hard-coded). Once the timer elapses, a core_start event is sent, which will moves the FSM to the running state. This means that for a very low battery system, the system will restarts each 6 hours.
- In the case of a tracker equipped with a rechargeable battery, the main FSM remains in the off state until a ***type_power_charging*** event (USB cable connected) or a button press (if the start device action is mapped on the button). If the system moved to Off state due to the low battery, then the main FSM generates a FSM skip event, which will move it back to the ACTIVE state. Note that power manager generates the ***type_power_charging*** system event only for devices with a rechargeable battery.

The network, geolocation, and BLE managers are registered to the **core_running** and **core_off** system events, enabling them to start or stop their operations accordingly. The accelerometer is managed

directly by the main FSM (Finite State Machine) and aligns with its states as follows:

- **core_running**: Accelerometer is activated.
- **core_off**: Accelerometer is deactivated.

When the core transitions to the **off** state, the following features and devices are disabled:

- BLE beaconing
- BLE connectivity
- Accelerometer
- Geolocation (ongoing acquisitions are canceled)
- Network (LoRaWAN or cellular connections are shut down)

6 Provisioning and manufacturing

6.1 Provisioning

The provisioning is performed during manufacturing. It setups LoRaWAN keys as well as hardware specific parameters that are not expected to change during normal operation of the device.

The LoRaWAN keys are stored in the write-only secure memory of the LR1110 transceiver cryptography engine The LoRaWAN device identifier is stored in the dedicated register of the LR1110.

The other hardware parameters are stored in the LR1110 user parameter flash memory, which can contain up to 16 integer parameters (32 bits).

Currently the AOS SDK uses parameter 15 to store the LoRaWAN nonce value.

AT3 uses the parameters 0 and 1 to store the hardware specific parameters. The parameters are coded as follow:

Parameter 0		
Bits	**Identifier**	**Description**
0	DCR	Enable/disable the LoRa duty cycle control. Set to 1 to disable.
1	DWELL	Enable/disable the LoRa dwell time control. Set to 1 to disable. AT3 does not support this flag.
5 - 2	TX_POW	TX power compensation. Signed value ranging from -7 to +7 dB (2's complement encoding)
6	SHIPPING	When set to 1, the shipping mode is enabled.
7	MANUF	When set to 1, the manufacturing mode is enabled.
8	JOINING	When set to 1, the joining mode is enabled.
9-10	ATEX zone	00: non ATEX; 01: ATEX 1; 10:ATEX 2; 11:ATEX 0
11– 31	RFU	Reserved for future use

Parameter 1		
Bits	**Identifier**	**Description**
15 - 0	Battery capacity	Battery capacity in mAh.
22 - 16	Buzzer volume	Limit for the buzzer volume. Range 0 – 127. Values • 0: Disabled • 1 – 100: 1% to 100% • 101 – 127: 100%
27 - 23	Resistor value	Resistor value on the power supply. Step: 0.1 Ohm. Used to

		compensate battery voltage estimation for ATEx devices.
30 - 28	LDO regulated voltage	=0 VDD=3.0V =1 VDD=3.1V =2 VDD=3.3V =[7:3] RFU
31	Battery type	Battery type. Values • 0: Rechargeable battery • 1: Primary battery

Parameter 2. Components feature set			
Bit ID	**Description (=1 when device is mounted)**	**Bit ID**	**Description (=1 when device is mounted)**
0	Geoloc Module type (Note 1) 01=1WL-633 with gnss chipset; 00= 1WL-857 without gnss chipset	16	=1 CLI available =0 CLI disabled
1		17	= 1 JTAG interface enable = 0 JTAG interfcace disabled
2	External memory (1MB) (SPI interface)	18	VBUS detection (note 2) =0 Unavailable =1 Available
3	Accelerometer (I2C)	19	RFU
4	Barometer (I2C)	20	RFU
5	RFU	21	RFU
6	RFU	22	RFU
7	RFU	23	RFU
8	Button 1 or Hall effect 1, =0 not mounted =1 mounted	24	LED 1 =1 if mounted
9	Button 1 / Hall effect 1 polarity =0 : active low (GPIO=0V when button pressed) =1 : active high (GPIO=VDD when button pressed)	25	LED 2 =1 if mounted
10	Button 2 or Hall effect 2, =0 not mounted =1 mounted	26	BUZZER / piezzo (assume single ended) (PWM output) =1 if mounted
11	Button 2 / Hall effect 2 polarity =0 : active low (GPIO=0V when button pressed) =1 : active high (GPIO=VDD when button pressed)	27	ADC input =1: ADC input enabled
12	Anti-tampering, =0 not mounted =1 mounted	28	RFU
13	RFU	29	RFU

14	RFU	30	RFU
15	RFU	31	RFU

Parameter 3. BOM identifier		
Bits	**Identifier**	**Description**
15 - 0	Batch ID	Hardware batch identifier
31 – 16	BOM version	Hardware BOM version composed in 31:27: product type 26:24: Major PCB 23:21: Minor PCB 20-16: B0M version

Parameter 4. Power consumption		
Bits	**Identifier**	**Description**
6 - 0	LED1 current	Current Consumption of LED1 in 2mA step
7	LED1 polarity	=0 (active low, LED ON w/ out=GND) =1 (active high)
14-8	LED2 current	Current Consumption of LED2 in 2mA step
15	LED2 polarity	=0 (active low, LED ON w/ out=GND) =1 (active high)
22-16	Buzzer current	Average current Consumption of the buzzer in 2mA step
23	RFU	
31-24	RFU	

Notes:

1- There are 2 versions of the Geoloc Modules. The version without the GNSS chipset (1WL-857) can enable the USART and 2 extras GPIO

2- All product with a USB interface includes a detection of VBUS to enable the USB (if bit 16=1). The anti-tampering switch is combined and VBUS detection is using the same GPIO. Presence of VBUS or release of anti-tampering enable USB interface.

3- Bit[12], Bit[18]: When set to 1, the GPIO is configured as an GPIO input.
Application:
a) The GPIO detects when the casing is opened using a security (anti-tampering) switch. A low signal (0V) indicates the casing is open, allowing the application to enable the USB interface if the CLI is available.
b) In some product, a low signal (0V) indicates the USB cable is connected (presence of 5V supply), allowing the application to enable the USB interface if CLI available

Bit[12]	Bit[18]	Description
0	0	No USB cable detection – No security switch (not advised)
1	0	Security switch ; Application: if GPIO=0 : USB ON and notification casing open
0	1	USB detection : Application : If GPIO = 0 :USB ON

Note that same GPIO can be used for those 2 applications.

6.2 Manufacturing

The manufacturing process should provision:

- The LoRaWAN keys as well as the LoRaWAN device unique identifier (DevEUI & JoinEUI)..
- LoRaWAN Regional parameters
- The STM32 user byte must be configured as follows:
  - **IWGDSTDBY** flag: Must be unchecked.
  - **IWDGSTOP**: Must be unchecked
  - **IWDGSW**: Must be unchecked (hardware) for customer platforms and checked (software) if you want to use the debugger. Refer to the next section for more details.

7 Bootloader, watchdog and debugger

7.1 Watchdog

7.1.1 Overview

The SDK uses the STM32 Independent watchdog (IWDG). It can run in either hardware or software mode. This selection is done via the option register in the STM32 flash memory. This register is usually written via the MFG and its value is preserved across resets.

The difference between hardware/software resides in the watchdog behavior:

- Hardware mode: The watchdog starts automatically when the MCU is powered on.
- Software mode: The watchdog starts only when the software enables it,

When the watchdog is used in hardware mode, the refresh period is initially set to 0.5 second. Then the bootloader v2 as well as AT3 modify the watchdog period to 30 seconds (max value).

The hardware watchdog is used in the LoRaWAN/LTE Combo Compact trackers to overcome a potential startup issue due to slow ramp-up of VCC upon battery insertion when super capacitors are empty (in recent designs, an additional circuitry has been added to avid slow VCC ramp-up for the cellular module in such case, but it can still happen to the STM 32 MCU)..

Due to the small refresh period supported by the watchdog, during the long device sleep periods, we would need to schedule wake-ups only to refresh the watchdog, wasting energy: to avoid this, the watchdog is frozen when sleeping (STOP and STANDBY modes).

7.1.2 Configuration

The option register of the Flash register must be initialized to the correct value. This is accomplished using the STM32 programmer, via the OB tab. The user configuration should be opened and the following flags should be modified:

- IWGDSTDBY flag: Must be unchecked.
- IWDGSTOP: Must be unchecked
- IWDGSW: Must be unchecked (hardware mode) for customer platforms and checked (software mode) if you want to use the debugger.

7.2 Bootloader

The bootloader is a small piece of code residing at the beginning of the user flash space. It is connected

to the usual ARM boot vector (vector 0: Stack address, vector 1: Reset address).
The bootloader checks whether it should wait for an application binary download or it should jump to the application. The decision to immediately start the application depends on the following criteria:

- The boot vector of the application should be correct (Stack address must be in the RAM and reset address must be in the flash).
- A special RTC backup register should not contain a value that forces the bootloader mode.

If one of these conditions is not met, the bootloader will wait for a new binary.

Note that the special RTC backup register is written with a non null value when the CLI command ***system boot*** is entered.
Once under the bootloader prompt, you have 1 minute to enter a command otherwise the system resets and will attempt to start the application (if we have a valid application pointer). Two main commands are available:

- ABWu: download a new application binary using Xmodem.
- ABWe: Erase the configuration in flash (recommended after an upgrade unless you must preserve the previous parameters and the configuration format is compatible with the new firmware).

There are two bootloader versions:

- version 1 does not manage the independent watchdog.
- version 2 starts the independent watchdog (regardless of the watchdog mode, software or hardware).

Care should be taken while changing to a different version of bootloader due to the watchdog. The following rules apply:

- If the watchdog is in software mode, you can use indifferently the two bootloader versions for AT3.
- If the watchdog is in software mode and you want to use the MFG firmware (the current version does not support the watchdog), you must use the bootloader v1.
- If the watchdog is in hardware mode, you must use the bootloader version 2 and AT3.

7.3 Debugger

To be able to use the debugger while the watchdog is active, the option register of the Flash register should have the flag IWDGSW set. This will avoid continuous reset under the debugger.

Also note that the debug configurations include a specific setting in the startup tab of Cube IDE (run commands) to freeze the watchdog while the debugger is active.

8 Device and power monitoring

This manager monitors the system. It generates a core system event when the monitoring timer elapses. The timer period is defined by the ***core_monitoring_period*** parameter. This manager is paused in OFF state.

8.1 Device manager

The device manager monitors the temperature.

The temperature monitoring process consists of the following steps:

- Measure the current temperature
- Compare it to the minimum and maximum values stored in the internal configuration parameter group .
- Update the associated internal parameter if a new minimum or a new maximum is detected.
- Compare the current temperature against the configuration parameters ***core_temp_high_threshold*** and ***core_temp_low_threshold*** and deduce whether we are in critical high, critical low or normal temperature state.

8.2 Power manager

8.2.1 Overview

The power manager is also scheduled on the monitoring system event. Once this even is received, it collects the consumption of all consumers and calculates the total consumption.

Based on the battery type a specific process takes place:

- Primary battery: Each time the total consumption exceeds by 2% the previous one, the manager saves the consumption in the associated internal configuration parameter (in flash).
- Rechargeable battery: The battery voltage is measured and converted into a remaining charge percentage. The manager does not store permanently the consumption in the internal configuration parameter. The manager also detects the presence of the USB cable connection. If connected, the battery is charging and the charge percentage is set to the special 0 value (reported via uplinks). Once the cable is disconnected, the manager resets the consumption of all consumers, so the power consumption reported is counted from last battery charge. Please note that the battery level reported may show less than 100% as it is estimated based on the voltage for rechargeable batteries. The charger operates with a voltage hysteresis to prevent continuous charging of a full battery: it allows the battery to discharge to a threshold before recharging begins again. If the cable is disconnected at the lowest voltage point, the battery may be reported as less than 100%.

When the battery monitoring is called, for devices with a rechargeable battery, the USB cable state is checked:

- If connected the system event ***type_power_charging*** is sent .
- If disconnected the system event ***type_power_not_charging*** is sent .

At the end of the processing, the power manager ensures that there is enough energy in the battery. If it is not the case it sends a ***type_power_low_battery*** to the system to signal a critically low level..

The condition to trigger the event is:

- Primary battery: If the last reset cause was a brown-out the event is sent and battery level is critically low. This was designed to avoid Join loops on low battery condition. This will put the system to OFF state, preventing a potential condition of cyclic brownout resets followed by network activity (e.g. LoRaWAN startup message).
- Rechargeable battery: If the remaining battery charge is lower or equal to 5% the event is sent.

8.2.2 Low battery management

Rechargeable battery

Once the main FSM receives a low battery system event (detected by the power manager), the following actions are done:

- Send a battery low power notification via the network
- Stop the geolocation
- Stop the accelerometer
- Stop the BLE activites (connectivity, beaconing)
- Stop the network
- The main FSM moves to the off state and sends a core_off system event.
  To restart the system, one of the following action should be done:
  - A button press if the start device action is mapped on the button.
  - A USB cable is plugged to the tracker (recharging battery).

Primary battery

The only action taken when the main FSM receives a low battery system event is the battery low notification sent toward the network.

The system remains in the running state until a brownout reset occurs. Once restarted (with the brownout reset and a battery low), the tracker checks the presence and the state of the tamper:

- Tamper absent
  - The main FSM stays in the off state and starts a 6 hours timer.
  - Once the timer expires, the system moves to the running state and attempt to connect to the network, restart the accelerometer, restart the geolocation and the BLE activities.
- Tamper present
  - If rebooting with the tamper off (casing closed), the 6 hours timer is started.
  - If rebooting with the tamper on (casing open), the 6 hours timer is not started and the system restarts immediately.
  - Note that in the case where the tamper was off and the 6 hours timer started, opening the casing (tamper on) will force a system restart. This mechanism has been implemented for the following reason:
    If the batteries are changed while in off state due to low battery, It is possible that the tracker does not reset due to the super cap discharge (takes long time since it drains a very low .current).

9 Configuration

This section describes the current AT3 configuration parameters. It relies on the configuration service, which supports different types of parameters (integer, float, ASCII string and byte array).

Note

The configuration service does not support parameter names (parameters only have a numeric identifier) or value ranges (parameters only have a type). In AT3 it is the application which provides parameter names as well as acceptable ranges for each parameter identifier. Unlike AT2, an out of bonds value is never written (AT2, in this case, uses the minimum acceptable value if we are below and the maximum value if we are above).

9.1 Parameter identifiers and groups

The configuration parameters are grouped per functionality domain. This eases management, particularly when a configuration change affects a single group: Registered software components against the configuration change can decide if the modified parameter group affects them or not.

The parameter identifiers are unique across the entire configuration. These unique identifiers called full-identifiers, are built from the group identifier and the local identifier. The local identifier is unique only inside a group. The full-identifier is a 16 bits value built with the most significant byte being the group identifier and the local-identifier as the least significant byte.

Full identifier	
Bits[15..8]	Bits[7..0]
Group identifier	Local identifier

Note that in the rest of the document the acronym LID refers to the Local Identifier within a group and FID refers to the Full identifier.

There are 4 types of parameters, which are indicated in the T column of the following tables and take the following value:

- **I**: 32-bit signed integer
- **F**: 32-bit floating-point number
- **S**: String, Null-terminated, ASCII encoded, Max 32 chars including Null
- **B**: Byte array, Max 32 Bytes

Currently, the following groups and parameters are defined. Note that the parameter identifiers and names can be subject to change before the first release.

9.1.1 Internal group

This group is specific because it contains the internal system variables that need to be preserved across power down and reset. The system reads from and writes to these variables.

Group 0. I**nternal**								
FID	**LID**	**T**	**Name**	**Min**	**Max**	**Dflt**	**U**	**Comments**
0x0000	0	I	sys_highest_temperature	-100	100	-100	^{0}C	Highest temperature reached
0x0001	1	I	sys_lowest_temperature	-100	100	100	^{0}C	Lowest temperature reached
0x0002	2	I	sys_power_consumption	0	-	0	mAH	Total power consumed

The CLI has an erase command, which can take the optional keyword **all**. When this keyword is omitted, the internal group (which stores extreme temperatures, battery levels, etc.) is backup'ed before erasing the flash, then restored. When the keyword **all** is provided, the internal group is not backup'ed and restarts with the factory default values.

To reset the values of the ***sys_highest_temperature*** or ***sys_lowest_temperature***, just write their default values.
The internal parameter ***sys_power_consumption*** is used only for trackers embedding primary batteries. A write to this parameter in presence of rechargeable battery must be avoided.

9.1.2 System core group

The core group contains the configuration parameters of the system.

Group 1. **System core**								
FID	**LID**	**T**	**Name**	**Min**	**Max**	**Dflt**	**U**	**Comments**
0x0100	0	I	core_monitoring_period	15	-	300	s	Device monitoring period.
0x0101	1	I	core_status_period	0	-	3600	s	Status reporting period.
0x0102	2	B	core_notif_enable	{0...}	{ff...}	See note	-	Notification enable bit map class 0 - 5
0x0103	3	I	core_temp_high_threshold	-100	100	60	^{0}C	Highest temperature detection threshold
0x0104	4	I	core_temp_low_threshold	-100	100	-0	^{0}C	Lowest temperature detection threshold
0x0105	5	I	core_temp_hysteresis	-100	100	5	^{0}C	Temperature hysteresis.
0x0106	6	I	core_button1_map	0	max	0	-	Button 1 mapping
0x0107	7	I	core_button2_map	0	max	0	-	Button 2 mapping

0x0108	8	I	core_buttons_timing	See note	max			Button timers (press, long press, debounce)
0x0109	9	B	core_led0_map	-	-	{0}	-	LED 0 mapping
0x010A	10	B	core_led1_map	-	-	{0}	-	LED 1 mapping
0x010B	11	B	core_buzzer_map	-	-	See note	-	Buzzer mapping
0x010C	12	I	core_almanac_validity	7	365	120	days	Number of days for which the GNSS almanac is considered as valid.
0x010D	13	I	core_almanac_outdated_ratio	0	100	100	%	Percentage of outdated GNSS almanac entries which will trigger network update requests. A value of 100% disables the network requests. Applicable for both LR11xx and MT33xx GNSS devices
0x010E	14	I	core_cli_password	0	Max	123	-	User cli password

core_monitoring_period: Period at which the device manager measures and manages the system variables such as the temperature, the battery level and so on. Refer to section Error: Reference source not found.

core_status_period: Period at which the status notification is sent. This notification plays the role of the LoRa live uplink in AT2. See also section Notificationson notification uplinks.

core_notif_enable: This byte-array configures the uplink notification classes and types to be sent via the network. The index of the array represents the class while the value is a bitmap indicating the notification types inside the class to be sent. The value is so coded as the sum of 2^x, where x are the types. Example, Notifications to be sent:

- tamper: Class 0, type 3. Value: $2^3 = 8$.
- SOS: class 1, type 0 and 1 (SOS on, SOS off). Value: $2^0 + 2^1 = 3$
- Temperature high: class 2, type 0. Value: $2^0 = 1$

The byte-array will be set to

{08,03,02,00,00,00}

Default/ Enable Status, Heartbeat, Low battery and tamper, temperature (all), motion start/end:

{1B,00,07,03,00,00}

Refer to the section Error: Reference source not found for classes and types.

core_button1_map and ***core_button2_map***: Define the events generated by the button action.
Each event type is coded on 4 bits.
Map:

- Bit 0-3: Event generated on a button press.
- Bit 4-7: Event generated on a button long press.
- Bit 8-11: Event generated on a button single click.
- Bit 12-15: Event generated on a button double click.
- Bit 16-19: Event generated on a button triple click or above.
- Bit 20-23: Event generated on a button simple sequence
- Bit 24-31: RFU

Type of events:

- 0 – No action
- 1 – Display battery level on the LED
- 2 – Start/Stop the SOS
- 3 – Request a position on demand (POD)
- 4 – Force an uplink system status notification transmission
- 5 – Start device
- 6 – Stop device
- 7 – Start only SOS
- 8 – Start BLE advertising for connectivity
- 9..15 - RFU

Button 1 default value: 0x00000380 (click: POD, long press: start BLE advertisement)

core_buttons_timing: Define the buttons timing parameters.
Each part of the bitmap is coded as below.
Map:

- Bit 0-3: Duration of the button press in 0.5 second.
- Bit 4-7: Delay in 0.5 second to be added to the button press to detect a long press.
- Bit 8-15: Debounce duration on button 1 in milliseconds.
- Bit 16-23: Debounce duration on button 2 in milliseconds.
- Bit 24-31: RFU

Notes

- The button press and long press timer values are the same for the two buttons.
- If the event *start device* is not mapped on either ***core_button1_map*** or ***core_button2_map***, the device automatically starts after the startup procedure (skip the off mode)
- If the event *stop device* is not mapped on either ***core_button1_map*** or ***core_button2_map***, the device cannot move to the off state.
- ***core_buttons_timing :*** the min value for these parameters is set to **`0x000A0A11`** and the default value is **`0x003232A2`**:
  - Button press: min duration is 0.5 seconds, default duration is 1 seconds
  - Button long press: Default delay is 5s, leading to a total duration of 6s (press delay + press duration).
  - Button debounce on button 1 and 2: min duration is 10ms, default duration is 50ms
- The duration of the button press cannot be equal or greater than long press.
- The button simple sequence is a hard coded sequence that could be used without starting the special button sequence (with a 14 second press), the simple sequence is as below:
  - Button press for button press duration (by default 3s)
  - Button release for button press duration (by default 3s)
  - Button press for button press duration (by default 3s)

core_led0_map and ***core_led1_map***: Define the LED patterns to be displayed upon given events.
The byte array is split in 10 slices of 3 bytes each. Each slice configures a pattern for a system event.
The parameter is defined as:

Slice 1	Slice 2	...	Slice 10

Byte 0	Byte 1	Byte 2	Byte 0	Byte 1	Byte 2		Byte 0	Byte 1	Byte 2
ext/cls	type	pattern	ext/cls	type	pattern		ext/cls	type	pattern

Where

- **ext/cls:** Pattern extension and system event class.
  - bit 0..4: System event class. Refer to the Event manager section.
  - bit 5..6: Pattern loop extension (Most significant bits).
  - bit 7: Pattern inversion:
    - 0 – The pattern is played as defined.
    - 1 – The pattern is inverted.
- **type**: System event type. Refer to the Event manager section.
- **pattern**: Pattern configuration:
  - bit 0..3. Pattern identifier:
    - 0 – None. Not configured
    - 1 – LED off. Pattern duration: Infinite
    - 2 – LED on. Pattern duration: 1s.
    - 3 – Fade in. Pattern duration: 2.5s. Usually used for device power on.
    - 4 – Fade out. Pattern duration: 2.5s. Usually used for device power off.
    - 5 – Blink slow: On: 1000ms, Off: 1000ms. Pattern duration: 2s.
    - 6 – Blink medium: On: 500ms, Off: 500ms. Pattern duration: 1s.
    - 7 – Blink fast: On: 250 ms. Off: 250 ms. Pattern duration: 0.5s
    - 8 – Flash slow: On: 100ms, Off: 2s. Pattern duration: 2.1s.
    - 9 – Flash fast: On: 100ms, Off: 1s. Pattern duration: 1.1s. Usually used for SOS
    - 10 – Heart: On: 100ms, Off: 250s, On: 100ms, Off: 1s. Pattern duration: 1.35s.
  - bit 4..7. Pattern loop (Least significant bits): Number of times the pattern is displayed. Used in combination with the **Pattern loop extension**. A combined value of 0 means infinite.

For more details, refer to the section Error: Reference source not found.

core_buzzer_map: Define the buzzer melodies to be played upon given events.
The byte array is split in 10 slices of 3 bytes each. Each slice configures a melody for a system event.
The parameter is defined as:

Slice 1			Slice 2			...	Slice 10		
Byte 0	Byte 1	Byte 2	Byte 0	Byte 1	Byte 2		Byte 0	Byte 1	Byte 2
ext/cls	type	melody	ext/cls	type	melody		ext/cls	type	melody

Where

- **ext/cls:** Melody extension and system event class.
  - bit 0..4: System event class. Refer to the Event manager section.
  - bit 5..7: Melody count extension (Most significant bits).
- **type**: System event type. Refer to the Event manager section.
- **melody**: Melody configuration:
  - bit 0..4. Melody identifier:

- 0 – None. No configured
- 1 – Melody 1: Off. Melody duration: Infinite
- 2 – Melody 2.
- 3 – Melody 3
- 4 – Melody 4
- ...
- 21 – Melody 21

◦ Bit 5..7. (Least significant bits): Number of times the pattern is displayed. Used in combination with the **Pattern count extension**. A combined value of 0 means infinite.

Default value:

{09,03,24,09,01,25,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00}

Buzzer plays melody 4 when tracker is running an plays the melody 5 when moving to OFF mode.

For more details, refer to the section Error: Reference source not found.

9.1.3 Geolocation engine group

This group configures the geolocation engine local processing.

Group 2. **GEOLOC**								
FID	**LID**	**T**	**Name**	**Min**	**Max**	**Dflt**	**U**	**Comments**
0x0200	0	I	geoloc_motion_period	10	86400	300	s	Position acquisition period while in motion
0x0201	1	I	geoloc_static_period	10	86400	3600	s	Position acquisition period while static
0x0202	2	I	geoloc_sos_period	10	86400	60	s	Position acquisition period while in sos
0x0203	3	I	geoloc_motion_nb_start	0	10	1	-	Number of acquisitions on motion start event
0x0204	4	I	geoloc_motion_nb_stop	0	10	1	-	Number of acquisitions on motion stop event
0x0205	5	I	geoloc_start_stop_period	10	86400	120	s	Interval between position acquisitions while acquiring consecutive positions on motion start or stop
0x0206	6	I	geoloc_gnss_hold_on_mode	0	5	0	-	Select the GNSS hold on mode. Value: Check section 11.3.1 0
0x0207	7	I	geoloc_gnss_hold_on_timeout	0	86400	0	s	GNSS hold on mode timeout, applicable to all hold-on modes except Disabled. 0 disables the Hold-on mode.
0x0208	8	I	geoloc_profile0_triggers	0	0xFFFFFFFF	0x3D	-	Geolocation event triggers 0
0x0209	9	I	geoloc_profile1_triggers	0	0xFFFFFFFF	1	-	Geolocation event triggers 1
0x020A	10	I	geoloc_profile2_triggers	0	0xFFFFFFFF	1	-	Geolocation event triggers 2
0x020B	11	B	geoloc_gbe_profile0_tec	-	-		-	Technologies to schedule using the

			hno					basic engine for events in triggers 0
0x020C	12	B	geoloc_gbe_profile1_techno	-	-	-	-	Technologies to schedule using the basic engine for events in triggers 1
0x020D	13	B	geoloc_gbe_profile2_techno	-	-	-	-	Technologies to schedule using the basic engine for events in triggers 2

- **geoloc_profileX_triggers** : the set bits select the events that will be handled according to profile0 (i.e. the sequence of technologies defined in geoloc_gbe_profileX_techno):
  - bit 0: geo_trigger_pod: Geoloc triggered on Position-on-demand via downlink or via button.
  - bit 1: geo_trigger_sos: SOS started
  - bit 2: geo_trigger_motion_start: Geoloc triggered on motion start event
    - Require the configuration of **geoloc_motion_nb_start** and **geo_start_stop_period**
  - bit 3: geo_trigger_motion_stop: Geoloc triggered on motion stop event
    - Require the configuration of **geoloc_motion_nb_stop** and **geo_start_stop_period**
  - bit 4: geo_trigger_in_motion: Periodic geoloc while the tracker is in motion
    - Require the configuration of geo_motion_period.
  - bit 5: geo_trigger_in_static: Periodic geoloc running while the tracker is static
    - Require the configuration of geo_static_period.
  - bit 6: geo_trigger_shock: Geoloc triggered on shock action
    - Require the shock detection configured (accelerometer)
  - bit 7: geo_trigger_temp_high_threshold: Geoloc triggered on temperature high.
  - bit 8: geo_trigger_temp_low_threshold: Geoloc triggered on temperature low.
  - bit 9: geo_trigger_geozoning: Geoloc stopped while in monitored area. Enabled when leaving the monitored area.

gbe_profileX_techno

Each byte represents a technology to schedule, coded as follow:

- Bit 7: Action Identifier
- Bits [6..0]: Technology Identifier, encoded as a 6-bit unsigned integer.

Available actions are

Identifier	Action
0	Skip on success
1	always_done

- **Always_Done** : the technology must be always scheduled regardless of the success or failure of previous technologies.
- **Skip on success**: The technology should be scheduled only if all previous technologies failed to acquire a position, i.e. it will be skipped if any of them succeeded.
- Note that technology “None” is always skipped silently (no notification), so the next technology if any will be triggered.

Available technologies are:

Identifier	Technology
0	None
1	LR11xx_A_GNSS
2	WIFI
3	BLE scan 1
4	BLE scan 2
5	aided_GNSS
6	GNSS

See also section Geolocation manager for details on the geolocation manager and configuration examples.

Default value for profile 0 (GNSS only): {06, 00, 00, 00, 00,00}

9.1.4 GNSS group

This group configures the MT3333 GNSS.

Group 3. **gnss**								
FID	**LID**	**T**	**Name**	**Min**	**Max**	**Dflt**	**U**	**Comments**
0x0300	0	I	gnss_constellation	0	6	2	-	GNSS constellations to be used. Supported values: • 0. GPS only • 1. GLONASS only • 2 . GPS and GLONASS • 3. GPS and GALILEO • 4. GPS, GLONASS and GALILEO • 5. BEIDOU only • 6. BEIDOU and GPS,
0x0301	1	I	gnss_max_time	30	300	300	s	GNSS max acquisition time.
0x0302	2	I	gnss_t0_timeout_static	0	300	30	s	Max time to acquire at least one satellite when the tracker is static
0x0303	3	I	gnss_ehpe_static	0	100	20	m	Expected Estimated horizontal position error. used when the tracker is static.
0x0304	4	I	gnss_convergence_static	0	300	20	s	Extra-time after a first fix to refine the fix. Used when the tracker is static
0x0305	5	I	gnss_t0_timeout_motion	0	300	30	s	Max time to acquire at least one satellite when the tracker is in motion.

0x0306	6	I	gnss_ehpe_motion	0	100	30	m	Expected Estimated horizontal position error. used when the tracker is in motion.
0x0307	7	I	gnss_convergence_motion	0	300	20	s	Extra-time after a first fix to refine the fix. Used when the tracker is in motion.
0x0308	8	I	gnss_standby	0	-	604800	s	Max time to let the device in standby mode.
0x0309	9	I	gnss_agnss_max_time	15	240	45	s	Aided GNSS max acquisition time.
0x030A	10	I	gnss_t1_timeout	0	300	0	s	Extra time let in Aided GNSS mode to try doing a fix.

9.1.5 LR11xx group

This group configures the LR1110 GNSS.

Group 4. **LR1110**								
FID	**LID**	**T**	**Name**	**Min**	**Max**	**Dflt**	**U**	**Comments**
0x0400	0	I	lr_constellation	1	6	6	-	GNSS constellations to be used.
0x0401	1	I	lr_scan_mode	1	2	1	-	LR1100 GNSS mode (NAV1 / NAV2)
0x0402	2	I	lr_nb_scans	1	4	2	-	Number of scans for one position acquisition
0x0403	3	I	lr_inter_scan_time	0	15	5	s	Time to wait between the scans for a position
0x0404	4	I	lr_wifi_report_nb_bssid	1	32	4	-	Max number of WIFI BSSID per scan
0x0405	5	I	lr_wifi_min_nb_bssid	1	10	3	-	Minimum number of BSSID to consider the scan as success (solvable). Below this value the result will be not-solvable. **Must be less or equal to lr_wifi_report_nb_bssid.**
0x0406	6	I	lr_wifi_min_rssi	-100	0	3	-	Minimum RSSI to consider the BSSID. A null value disable the filter.
0x0407	7	I	lr_wifi_bssid_mac_type	0	2	1	-	MAC administration type of the BSSID to report.

lr_constellation values

- 0. GPS only
- 5. BEIDOU only
- 6. BEIDOU and GPS,

lr_scan_mode values

The LR11xx GNSS firmware implements several variants called “NAV<X>”, not all maybe available depending on the loaded firmware:

- 1. NAV1 scan
- 2. NAV2 scan

lr_wifi_bssid_mac_type

- 0. All BSSID administration types (universally and locally administered).
- 1. Universally administered BSSID only.
- 2. Locally administered BSSID only.

9.1.6 BLE scan group

There are 2 groups to configures two different BLE scans. Each scan has its own parameter set. Filters are available which are defined as masks references to the beginning of the Advertising payload (ADV). Refer to sectionBLE scan collection for details, including offset zero reference for each type of beacon.

Group 5. **BLE_SCAN1** Group 6**. BLE SCAN2**								
FID	**LID**	**T**	**Name**	**Min**	**Max**	**Dflt**	**U**	**Comments**
0x0500 0x0600	0	I	ble_scan_duration	50	61440	3000	ms	Total time for a BLE scan
0x0501 0x0601	1	I	ble_scan_window	3	10240	120	ms	Scan window
0x0502 0x0602	2	I	ble_scan_interval	1	10240	130	ms	Scan interval
0x0503 0x0603	3	I	ble_scan_type	0	7	0	-	Type of beacons to scan
0x0504 0x0604	4	I	ble_scan_min_rssi	-120	0	80	dB	Min RSSI to consider the beacon
0x0505 0x0605	5	I	ble_scan_min_nb_beacons	1	20	1	-	Min number of beacons to consider the scan as success (solvable). Below this value the result will be not-solvable. **Must be less or equal to ble_scan_nb_beacons.**
0x0506 0x0606	6	B	ble_scan_filter1_mask	-	-	{0}	-	Mask (10 bytes) to be applied to the ADV frame.
0x0507 0x0607	7	B	ble_scan_filter1_value	-	-	{0}	-	Comparison value (10 bytes) belonging to filter1
0x0508 0x0608	8	I	ble_scan_filter1_offset	0	21	0	-	Offset in the ADV from which we apply the filter1
0x0509 0x0609	9	B	ble_scan_filter2_mask	-	-	{0}	-	Mask (10 bytes) to be applied to the ADV frame.
0x050A 0x060A	10	B	ble_scan_filter2_value	-	-	{0}]	-	Comparison value (10 bytes) belonging to filter2
0x050B 0x060B	11	I	ble_scan_filter2_offset	0	21	0	-	Offset in the ADV from which we apply the filter2
0x050C 0x060C	12	I	ble_scan_nb_beacons	1	20	4	-	Number of beacons to report
0x050D 0x060D	13	I	ble_scan_report_type	0	2	0	-	Scan report type
0x050E 0x060E	14	I	ble_scan_report_id_ofs	0	25	4	-	Offset in ADV to extract the beacon identifier

Note

ADV means BLE advertisement frame

ble_scan_type values

- 0. All beacons.
- 1. Eddystone UUID beacons only.
- 2. Eddystone URL beacons only.
- 3. All Eddystone beacons.
- 4. iBeacon beacons only
- 5. AltBeacon beacons only
- 6. Custom (only based on filters)
- 7. Exposure advertisement

ble_scan_report_type values

- 0. MAC address
- 1. Beacon identifier in short format (2 bytes)
- 2. Beacon identifier in long format (16 bytes).

Refer to the Geolocation section for the configuration

9.1.7 Accelerometer group

This group configures the accelerometer.

Group 7. **Accelerometer**								
FID	**LID**	**T**	**Name**	**Min**	**Max**	**Dflt**	**U**	**Comments**
0x0700	0	I	accelero_motion_sensi	1	96	1	mg	Motion sensitivity. Step 31 mg.
0x0701	1	I	accelero_motion_duration	10	3600	120	s	Motion duration
0x0702	2	I	accelero_full_scale	0	3	3	-	Scale use (2,4,8,16 g).
0x0703	3	I	accelero_output_data_rate	0	4	0	-	Output data rate (12.5, 25, 50, 100, 200 Hz)
0x0704	4	I	accelero_shock_threshold	0	128	0	mg	Shock threshold. Increments of 63 mg

accelero_full_scale

Acceptable values:

- 0. Scale 2g
- 1. Scale 4g
- 2. Scale 8g
- 3. Scale 16g

Note

The low pass filter is set to ODR/20 for 2G full scale and to ODR/2 for the other scales. The sensitivity is kapped to 63 for 2G full scale.

accelero_output_data_rate
Acceptable values:

- 0. 12.5 Hz
- 1. 25 Hz
- 2. 50 Hz
- 3. 100 Hz
- 4. 200 Hz

There is currently no filtering on GADD index level, but only on shock level. The back-end application needs to filter based on GADD index value.

9.1.8 Network group

This group configures the general networking.

Group 8. **Network**								
FID	**LID**	T	**Name**	**Min**	**Max**	**Dflt**	**U**	**Comments**
0x0800	0	I	net_selection	0	4	0	-	Define the networking type: • 0: LoRaWAN only • 1: Cellular only • 2: LoRaWAN fallback cellular • 3: Cellular fallback LoRaWAN
0x0801	1	I	net_reconnection_spacing	0	Max int	600	s	Interval between connection retries on the primary network, when operating on the fallback network.
0x0802	2	I	net_main_probe_timeout	120	Max int	600	s	Maximum duration of an attempt to reconnect to the main network . Parameter available only for combo compact tracker..

Refer to section Networking for more details on the AT3 network manager.

9.1.9 LoRaWAN group

This group configures the LoRaWAN networking.

Only LoRaWAN OTA mode is supported in AT3. A join is triggered upon initial boot if LoRaWAN is the primary network, or every time LoRaWAN is activated as secondary network. It is also triggered after detection of network loss after failure of lorawan_probe_max_attempts number of link-checks, which are sent only if no downlink message has been received for lorawan_probe_period. All Joins are randomized (the randomization is managed by the LBM stack).

Group 9. **LoRaWAN**								
FID	**LID**	**T**	**Name**	**Min**	**Max**	**Dflt**	**U**	**Comments**
0x0900	0	I	lorawan_cnx_timeout	0	Max int	0	-	Use only on dual network configurations. Max time to wait for

								joining the network including all retries (timeout triggers a switch to the back-up network). Value 0 disables the timer (use this value for LoRaWAN only configurations)..
0x0901	1	I	lorawan_dl_trigger_period	0	Max int	600	s	Period at which an empty uplink is sent to trigger a Rx window for downlinks (if no uplink has been sent within this period). 0 disables the function.
0x0902	2	I	lorawan_probe_max_attempts	0	10	4	-	Number of link-check sent before declaring the network as lost
0x0903	3	I	lorawan_probe_period	120	Max int	43200	s	Time between link-check requests, or since last downlink activity.
0x0904	4	B	lorawan_confirm_notif_map	{0}	{FF}	{0}	-	Map enabling the LoRaWAN confirmed message for notifications.
0x0905	5	I	lorawan_confirm_notif_retry	0	15	0	-	Number of retries for confirmed messages. Value 0: the number of TX follow ADR.
0x0906	6	I	lorawan_s1_tx_strategy	1	-	0x7380E	-	Socket 1. Transmission strategy
0x0907	7	I	lorawan_s1_ul_port	1	252	19	-	Socket 1. Uplink port
0x0908	8	I	lorawan_s1_dl_port	1	252	3	-	Socket 1. Downlink port

TX strategy

The parameter *lorawan_s1_tx_strategy* is a bit-field on 3 bytes providing the LoRa transmission strategy. The bitfield is defined as follow:

Byte #2 Datarates for the second TX	Byte #1 Datarates for the first TX	Byte #0 Control
x x x x x x x x	x x x x x x x x	0 0 0 0 D M S A

Byte #0 – Control

- Bit #0 (A). Set to enable LoRaWAN network ADR control when the tracker is static. Reset to disable the network ADR control regardless the motion state of the tracker.
- Bit #1 (S). Control the dual transmission when the tracker is static. Set to enable the dual transmission in static state. Reset to disable it.
- Bit #2 (M). Control the dual transmission when the tracker is in motion. Set to enable the dual transmission in motion state. Reset to disable it in motion state.
- Bit #3 (D). Control the DR (Datarate) modification for over-sized messages. Set to allow AOS to increase the DR for messages not fitting the allowed maximum LoRaWAN payload size for a given datarate. Reset this flag to prevent sending of over-sized messages.
- Bit #4-7: Unused.

Byte #1 – Datarates enabled for the first transmission.

- Bit #0. Enable datarate DR0. When set, the datarate is enabled.
- Bit #1. Enable datarate DR1. When set, the datarate is enabled.
- ...
- Bit #7. Enable datarate DR7. When set, the datarate is enabled.

Byte #2 – Datarates enabled for the second transmission.

- Bit #0. Enable datarate DR0. When set, the datarate is enabled.
- Bit #1. Enable datarate DR1. When set, the datarate is enabled.
- ...
- Bit #7. Enable datarate DR7. When set, the datarate is enabled.

Default: Double transmit for motion and static. TX1: DR3,4,5, TX2: DR0, DR1, DR2.

Confirm_notif_map.

This byte-array configures the notification classes and uplink types to be sent as LoRaWAN confirmed uplinks. The index of the array represents the class while the value is a bitmap indicating the notification types inside the class to be acknowledged. The value is so coded as the sum of 2^x, where x are the types. Example, Notifications to be acknowledged:

- tamper: Class 0, type 3. Value: $2^3 = 8$.
- SOS:class 1, type 0 and 1 (SOS on, SOS off). Value: $2^0 + 2^1 = 3$
- Temperature high: class 2, type 0. Value: $2^0 = 1$

The byte-array will be set to

{08,03,02,00,00,00}

Default value

{00,00,00,00,00,00}

9.1.10 Cellular group

This group configures the cellular network. Available only for combo compact trackers.

Group 10. **Cellular**								
FID	**LID**	**T**	**Name**	**Min**	**Max**	**Dflt**	**U**	**Comments**
0x0A00	0	I	cell_sim_interface	0	3	0	-	SIM interface 0: SIM0 (0), 1: e-SIM, value 2 and 3 reserved for future use.
0x0A01	1	I	cell_network_type	1	2	1	-	Network type. • 0: Cellular not used • 1: LTE-M • 2: NB-IOT
0x0A02	2	I	cell_search_bands	{0}	See note	See note		Radio frequency bands scanned to search a cell. See note.
0x0A03	3	I	cell_cnx_timeout_static	180	900	180	s	Maximum search duration for a cellular network. Applicable when the tracker is static.
0x0A04	4	I	cell_cnx_timeout_motion	180	900	300	s	Maximum search duration for a cellular network. Applicable when the tracker is in

								motion.
0x0A05	5	I	cell_cnx_nw_reconnect_timeout	0	900	60	s	When the modem loses the network connection, an auto-reconnect timeout is triggered to allow the modem to recover the network connection on its own... There are no retries for an administrative disconnect reason, otherwise there are up to cell_cnx_max_attempts retres After this the network manager will attempt to use the fallback technology if any, and try to reconnect to the primary technology with interval net_reconnection_spacing, which is expected to be larger than . .
0x0A06	6	I	cell_cnx_max_attempts	1	10	3	-	Maximum number of times the network search may be repeated before shutting down the modem and passing control to the network manager.
0x0A07	7	S	cell_access_point_name	-	-	“”	-	String (max 32 bytes) providing the service access point name (APN).. If not provided, this information is retrieved from the SIM.
0x0A08	8	S	cell_operator_sim_slot_0	-	-	“”	-	Cellular operator name when using SIM0
0x0A09	9	S	cell_operator_sim_slot_1	-	-	“”	-	Cellular operator name when using SIM1 (E.SIM).
0x0A0A	10	I	cell_low_power_mode	0	3	0	-	Low power mode: • 0: disabled. • 1: PSM • 2: eDRX • 3. both PSM and eDRX.
0x0A0B	11	I	cell_psm_tau_period	0	255	24	-	Bit-field giving the requested TAU period.
0x0A0C	12	I	cell_psm_active_time	0	255	2	-	Bit-field giving the requested active time.
0x0A0D	13	I	cell_edrx_pcl	0	15	15	-	Requested paging cycle length
0x0A0E	14	I	cell_edrx_ptw	0	15	3	-	Requested paging time window
0x0A0F	15	I	cell_rai_timeout	0	10000	500	ms	RAI (Release Assistance Indication) timeout . A null value disables the feature. **Use only with UDP protocol**.
0x0A10	16	I	cell_probe_max_attempts	0	10	0	-	Number of echo-request sent before declaring the network as lost. Set 0 to disable the feature.
0x0A11	17	I	cell_probe_period	120	Max int	600	s	Time between echo-request, or since the last downlink activity. It is recommended to have this value greater than the aggregation timer.
0x0A12	18	I	cell_s1_transport_proto	0	1	1	-	Socket 1 transport protocol : • 0: TCP • 1: UDP
0x0A13	19	S	cell_s1_ip_url_addr	-	-	“”	-	Socket 1 remote IP address or FQDN in string format (max 32 bytes)

0x0A14	20	I	cell_s1_dst_ip_port	0	65535	0	-	Socket 1 destination UDP/TCP port number.
0x0A15	21	I	cell_s1_src_ip_port	0	65535	0	-	Socket 1 local UDP/TCP port number. Value 0 means that the modem will choose one.
0x0A16	22	I	cell_s1_tx_aggr_time	0	3600	120	s	Duration in second for which the messages are buffered in the socket 1 transmit queue before being transmitted
0x0A17	23	S	cell_apn_user_id	-	-	“”	-	String specifying the user identifier for private APN.
0x0A18	24	S	cell_apn_user_pwd	-	-	“”	-	String specifying the user password for private APN.
0x0A19	25	I	cell_apn_auth_protocol	1	2	1	-	Authentication protocol used for private APN connection. • 1: PAP • 2: CHAP

Search bands

This byte array parameter contains the list of cellular bands (**encoded in hexadecimal form, e.g. band 28 is coded as 0x18**) to scan to find an appropriate cell.

Notes

- Filling this array to 0 instructs the modem to scan all possible bands.
- The array must end by a null value
- Default value:
  {01,03,08,13,14,1C,00,00,00,00,00,00,00,00,00,00,00,00,00,00}
- Supported bands
  {01,02,03,04,05,08,0C,0D,0E,11,12,13,15,19,1A,1C,42,47,55}

Active time bit-field

Bits 4 to 0 represent the binary coded timer value and bits 5 to 7 the timer value unit .

Bits			Timer value unit
B7	B6	B5	
0	0	0	Value is incremented in multiples of 2 seconds
0	0	1	Value is incremented in multiples of 1 minute
0	1	0	Value is incremented in multiples of deci-hours
1	1	1	Value indicates that the timer is deactivated.

TAU Time bit-field

Bits 4 to 0 represent the binary coded timer value and bits 5 to 7 the timer value unit.

Bits			Timer value unit
B7	B6	B5	
0	0	0	Value is incremented in multiples of 10 minutes
0	0	1	Value is incremented in multiples of 1 hour
0	1	0	Value is incremented in multiples of 10 hours
0	1	1	Value is incremented in multiples of 2 seconds
1	0	0	Value is incremented in multiples of 30 seconds
1	0	1	Value is incremented in multiples of 1 minutes
1	1	0	Value is incremented in multiples of 320 hours
1	1	1	Value indicates that the timer is deactivated

eDRX PCL and PTW bit-field

The eDRX parameters PCL (Paging Cycle Length) and PTW (Paging Transmission Window) are each encoded on 4 bits.

Bits				PCL duration in seconds	WB-S1 (LTE-M) PTW value in seconds	NB-S1 (Nb-IoT) PTW value in seconds
B3	B2	B1	B0			
0	0	0	0	5.12 (see note 1)	1.28	2.56
0	0	0	1	10.24 (see note 1)	2.56	5.12
0	0	1	0	20.48	3.84	7.68
0	0	1	1	40.96	5.12	10.24
0	1	0	0	61.44 (see note 2)	6.40	12.80
0	1	0	1	81.92	7.68	15.36
0	1	1	0	102.40 (see note 2)	8.96	17.92
0	1	1	1	122.88 (see note 2)	10.24	20.48
1	0	0	0	143.36 (see note 2)	11.52	23.04
1	0	0	1	163.84	12.80	25.60
1	0	1	0	327.68	14.08	28.16
1	0	1	1	655.36	15.36	30.72
1	1	0	0	1310.72	16.64	33.28
1	1	0	1	2621.44	17.92	35.84
1	1	1	0	5242.88 (see note 3)	19.20	38.40
1	1	1	1	10485.76 (see note 3)	20.48	40.96

Notes:

- 1 : The value is applicable only in WB-S1 (LTE-M) mode. I**f used in NB-S1 (NB-IoT) mode, it will cause the modem to fail.**

- 2 : The value is applicable only in WB-S1 (LTE-M) mode. If used in NB-S1 (NB-IoT) mode, it is interpreted as **"0010"**, equivalent to **20.48 seconds**.
- 3 : The value is applicable only in NB-S1 (NB-IoT) mode. If used in WB-S1 (LTE-M) mode, it is interpreted as **"1101"**, equivalent to **2621.44 seconds**.

9.1.11 BLE group

This group configures the BLE parameters.
section 9.1.11 (ble_cnx_init_config -> ble_cnx_behavior)

Group 11. **BLE**								
FID	**LID**	**T**	**Name**	**Min**	**Max**	**Dflt**	**U**	**Comments**
0x0B00	0	I	ble_cnx_tx_power	0	31	19	-	BLE Tx power level, see table below.
0x0B01	1	I	ble_cnx_adv_duration	30	Max int	60	s	Time to wait before stopping advertising or switching to slow advertising.
0x0B02	2	I	ble_cnx_behavior	0	4	1	-	The connectivity configuration.
0x0B03	3	I	ble_beacon_tx_power	0	31	19	dBm	BLE Tx power level for beaconing.
0x0B04	4	I	ble_beacon_type	0	5	0	-	Beacon Type to start.
0x0B05	5	B	ble_beacon_identifier	-	-	{}	-	BLE beaconing ID parameter, array of 16 to 24 bytes depending on beacon type.
0x0B06	6	I	ble_beacon_fast_adv_interval	40	10240	333	ms	BLE beacon fast advertising interval.
0x0B07	7	I	ble_beacon_slow_adv_interval	40	10240	1000	ms	BLE beacon slow advertising interval.

cnx tx power level

The user can specify the desired TX power via the ble_cnx_tx_power parameter, representing the TX level as defined by ST microelectronics for the STM32WB55, the correspondence between TX power level and TX power dBm is as follows:

Level	dBm	Level	dBm	Level	dBm	Level	dBm	Level	dBm	Level	dBm
0x00	-40	0x06	-15.25	0x0B	-9.9	0x10	-4.95	0x15	-1.3	0x1A	+1
0x01	-20.85	0x07	-14.1	0x0C	-8.85	0x11	-4	0x16	-0.85	0x1B	+2
0x02	-19.75	0x08	-13.15	0x0D	-7.8	0x12	-3.15	0x17	-0.5	0x1C	+3
0x03	-18.85	0x09	-12.05	0x0E	-6.9	0x13	-2.45	0x18	-0.15	0x1D	+4
0x04	-17.6	0x0A	-10.9	0x0F	-5.9	0x14	-1.8	0x19	0	0x1E	+5
0x05	-16.5									0X1F	+6

cnx adv duration

This parameter defines the duration of the fast advertising timer. When the device initiates connectivity, it starts with fast advertising parameters, with the fast advertising interval set to 500ms.

When the device is connected to a mobile app, the fast advertising timer is stopped, and a new timer (ble_timer) is started for 60 seconds to allow the mobile app to proceed with the bonding procedure. If the ble_timer expires, the device will disconnect and restart fast advertising.

When the fast advertising timer is triggered, depending on the configuration and BLE bond status, there could be two actions:

<table>
<tr><th>ble_cnx_init_config</th><th>Bond exist</th><th>No bond</th></tr>
<tr><td>enable_no_passkey</td><td rowspan="2">Switch to slow advertising</td><td rowspan="2">Connectivity stopped</td></tr>
<tr><td>enable_passkey</td></tr>
<tr><td>enable_no_passkey_no_slow_adv</td><td colspan="2" rowspan="2">Connectivity stopped</td></tr>
<tr><td>enable_passkey_no_slow_adv</td></tr>
</table>

Slow advertising interval is set to 2 seconds.

cnx behavior
This parameter is used to define the behavior of the connectivity in terms of passkey usage and the actions to take after the fast advertising timer is triggered.

ble_cnx_init_config	Value	Comment
Disable	0	- BLE connectivity not started when the device is turned ON
enable_no_passkey	1	- BLE connectivity started when the device is turned ON - No passkey needed to establish a secure connection
enable_passkey	2	- BLE connectivity started when the device is turned ON - Passkey needed to establish a secure connection
enable_no_passkey_no_slow_adv	3	- BLE connectivity started when the device is turned ON - No passkey needed to establish a secure connection - Switch to BLE Idle state when the fast advertising timer is triggered (Slow advertising disabled)
enable_passkey_no_slow_adv	4	- BLE connectivity started when the device is turned ON - Passkey needed to establish a secure connection - Switch to BLE Idle state when the fast advertising timer is triggered (Slow advertising disabled)

In case of passkey usage, the pin code will be set to the Unique device number value modulo 1000000, this will gives a pin code value between 000000 and 999999.

Beacon tx power level
Beaconing Tx power level, same as **cnx tx power level**.

Beacon type

Beacon type	Value	Comment
Disabled	0	Beaconing disabled
Eddystone UID	1	Eddystone UUID beacon emulation
I-Beacon	2	I-Beacon beacon emulation
AltBeacon	3	AltBeacon beacon emulation
QUUPPA	4	QUUPPA beacon emulation
Exposure	5	Exposure beacon emulation

Beacon identifier
This parameter will host the data to advertise in beaconing mode, the max size of this parameter is 24 bytes and minimum size is 16 bytes.

The data to advertise and the data size depends on the beacon type:

- Eddystone UID: 10 bytes of namespace followed by 6 bytes of instance.
- Ibeacon: 16 bytes for company UUID followed by 2 bytes for major number and 2 bytes for minor number.
- Altbeacon: 4 bytes for manufacturer ID followed by 20 bytes for beacon ID.
- QUUPPA: 1 byte for compensated Tx power followed by 6 bytes for identifier.
- Exposure: 16 bytes for random public identifier followed by 4 bytes for meta data.

By default the data is set to 0.

Beacon fast advertising interval
The fast advertising interval is used when the device is in motion state.

Beacon slow advertising interval
The slow advertising interval is used when the device is in static state.

9.2 Flash and RAM cache

The parameters are managed by the service called srv_config_param. It uses a fixed flash segment (0x080B7000) to store the parameters. In addition, the service maintains a cache in RAM.

Note that AT3 implements a versioning based compatibility control for the configuration. This information is used by the app_config ANI to decide whether the current version in the flash is acceptable for a given AT3 version. If the version differ, the ANI overwrites the parameter flash segment withs the factory default.

The flash and RAM cache coherency is managed by the ANI..

For changes configured through the CLI an explicit “conf save’ command synchronizes RAM cached values to Flash, on other application controlled parameters changes, flash flush may be delayed by up to a few seconds..

9.3 Configuration file

9.3.1 Overview

AT3 supports a configuration file appended to the firmware binary. The configuration file contains:

- A header including the keyword CONFIGURATION followed by the VERSION keyword and its version number.
- The list of the parameters that differs from the hard-coded ones.
- A trailer indicating the end of the configuration

Each parameter is addressed by its full identifier coded in hexadecimal and prefixed with the letter **P**. The full identifier must contains all 4 digits.

Examples

- **P0102** is a valid identifier (parameter group 1, local identifier 2), while **P102**, **P12** are not a valid. .
- **P0000** is a valid identifier (parameter group 0, local identifier 0), while any other writing for this parameter identifier is invalid.

Important notes

- All keywords as well as the parameter identifier prefix must be in upper case.
- There can be only one keyword per line.
- There can be only one parameter per line
- Each line is ended by a CR or LF or both.

9.3.2 Version format

The configuration version starts on a single line starting with the keyword VERSION followed by a space and the version number.

The number contains 4 fields separated by a dot.The general format is Major.minor.iteration.user.

Example
VERSION 1.0.0.1

The meaning and the management of the version is described hereafter.

9.3.3 Parameter format

There is a single parameter setting per line. The line is composed by:

- The parameter full identifier prefixed by the letter P. The identifier is expressed in hexadecimal and must contain the 4 digits.
- The equal sign, separating the identifier from its value
- The desired value of the parameter

Note that spaces may be inserted before and after the equal sign.

AT3 uses several parameter types. They are automatically discovered based on the value syntax:

- An integer value type is detected if it contains only numerical digits [0..9].
  Examples: 123, 0141, 1234 are valid integer numbers.
- An hexadecimal value is detected if the value starts with 0x and numerical digits [0..9] or letters [a..f] or [A..F] only follows the 0x prefix.
  Examples: 0x12aa, 0x45678, 0xFF123 are valid.
- A decimal number must contain a dot. Ie value 1 must be configured as “1.0.”
  Examples: 1.23, 11234.0, 12345 are valid.
- An ASCII string must be enclosed by the double quote (“). A string cannot include a double quote and cannot exceed 31 characters. Empty strings are supported.
  Examples: “abeeway”, “Hello word”, “” are valid.
- A byte array must start with the opening bracket ({) and end with the closing bracket (}). Each value is expressed in hexadecimal on 2 digits and separated by a comma (,). No space are allowed between the 2 brackets. The hexadecimal value is not prefixed with 0x.
  Examples: {00, 01,aa,ff,BB,1C} is valid.

9.3.4 Configuration file example

In this example, we modify the parameters core_status_period (0x0102), geoloc_motion_period (0x0200), geoloc_static_period (0x0201), geoloc_start_stop_period (0x0205) and geoloc_gbe_profile0_techno (0x020b).

```
CONFIGURATION
VERSION 1.0.0.1
P0102=900
P0200= 800
P0201 =300
P0205 = 0x125
P020b = {02,85,00,00,00}
END
```

9.4 Configuration management

9.4.1 Overview

The versioning control field of a configuration allows to check its compatibility with the firmware, and also provides indications for the intended use.

The versioning control field is coded as follows:

<Major>.<Minor>.<Iteration>.<User>

<Major>.<Minor> reflect the first (oldest) firmware version compatible with this configuration. Firmware with the same major version and same or higher minor versions will also be compatible with this configuration as Abeeway ensures upward compatibility within a given major version. A change of major version may break such compatibility (for example it may change the type of a parameter).

<Iteration> is intended to capture the serial number of configurations, and it is expected that it will be incremented at each change.

<User> is a free field, but the intended use is to identify a specific use case or context, and allow the people in charge of device management to rapidly check whether a configuration was designed for the specific use case. Abeeway will use <User>=0 for its generic product configurations out of factory.

Firmware upgrade behavior:

- FW1/ If the new firmware major version is the same as the current configuration stored in flash memory, and the minor version of the firmware is the same or higher, the flash configuration is valid and will be used. Configuration defined values will replace the default values of the firmware.
- The flash configuration <Minor> will be updated to match the one of the new firmware (as the new firmware may update a parameter, we can no longer guarantee the compatibility of the configuration with the previous firmware).

- FW2/ In all other cases, the current flash configuration is erased, and the firmware will use the default new firmware configuration. Implicitly the versioning control field of this default configuration is <FW_Major>.<MW.Minor>.0.0

Configuration update behavior

- The following must be both true:
  - The new configuration <Major> must be identical to the flash configuration >Major> (which itself always matches the running firmware Major due to rule FW2).
  - The new configuration <Minor> must be lower or equal to the flash configuration <Minor>, which itself always matches the running firmware Minor due to rule FW1).

  If it is not the case, the configuration is rejected, as a firmware upgrade is required first to ensure compatibility with the configuration.

  If both checks pass the configuration is accepted.
- If the configuration command flag is “replace” then the flash configuration is first erased and replaced with the new parameters. Parameters not specified in the new configuration return to their firmware-defined default values.
- If the configuration command flag is “patch” and either <Iteration> or <User> differ from the current configuration, then the configuration is updated with the new parameters,
- The flash <Iteration> and <User> elements are updated to match the new configuration.

Note that in the case of FUOTA FW upgrade which may also include a configuration, the order of execution is the following:

1/ Firmware upgrade and merge of current config with rules FW1 and FW2

2/ Configuration upgrade (and therefore at the time, the running firmware version is the new FUOTA upgraded firmware).

9.4.2 Monitoring commands and actions

The configuration in flash can be reset to the default values at any time. This action can be done either via a CLI command, a network command or the bootloader:

- The networking interface integrates a command, which clears and resets the tracker (refer to section Commands).
- The Bootloader includes the command **ABWe**, which erases the flash segment containing the configuration. Once reset, this flash section will be recreated by the application.
- The CLI commands **config erase** and **system reset**, respectively erase and reformat the flash and reset the device.

10 GNSS manager

10.1 Overview

The AT3 application contains a software subsystem called gnss-mgr.

This manager is responsible of:

- Managing the almanac
  - Monitoring the validity of the almanacs.
  - Requesting an update through the network when the almanacs are out of date
  - Updating the LR1110 and MT3333 almanacs
- Managing the aiding position
  - Monitor the aiding position (if moving during long time, then the aiding position may not be valid anymore)
  - Request the aiding position via the network
  - Update the aiding position

Satellite identifiers

The satellite identifiers start at 1 regardless the device type (MT3333 or LR1110) and the constellation. The manager adapts it regarding the device it addresses (For the LR, the first GPS satellite is 0, for the MT it is 1).

GNSS chip used

The manager checks whether a given GNSS chip is used (LR1110 and/or MT3333). To do this, it checks the technologies defined in the GBE profiles. All GBE profiles are parsed regardless if the geolocation actually uses them.
An unused device is not managed (neither almanac monitoring nor updating).

Note regarding the MT3333

It has been observed that a partial almanac update removes the existing entries.
For this reason, the manager maintains a local image of the GPS almanac using the LR format.
This image will used to perform a full update of the MT almanac.

Terminology used

- Refresh: Process which reads the almanac from the devices and update the validity.
- Request: Process which sends almanac requests toward the network.
- Update: process which updates the device almanac

10.2 Almanac management

The manager maintains the validity of the almanacs (GPS and BEIDOU) for the devices actually used (LR11xx and GNSS).

10.2.1 Refresh process

The almanacs are read at the start time and once a day. When the refresh operation is performed, the outdated (last refresh date < now() – 60 days) entries count per GNSS chip and per constellation (called validity) is updated.
The local almanac image is updated with the entries read from the GNSS chips.

When the two GNSS chips are used, the process always starts by the LR1110 and continues with the MT3333.

At the end of the refresh operation, the number of outdated entries of the almanacs are compared against the acceptable configured ratio. If this number exceeds the ratio, the almanac update process starts.

Notes

- The current firmware supports only the GPS almanac update (BEIDOU will be done later).
- If a change in the use of a device occurs via the configuration, the manager starts the almanac refresh process.
- Use of the 2 devices: since the LR11xx is not able to update its own almanacs, the MT GNSS is for now the only source of local almanac updates.
- The almanac monitoring period is fixed to 1 day and is based on the system device monitoring period.

10.2.2 Request and update process

Update process initiated by the device.

Once the almanac has been read and updated locally from the GNSS chip (if this chip is enabled in any profile), the process checks the outdated almanac entries per device and per constellation.
Note that the current firmware supports only the GPS almanac update.
If the number of outdated entries is greater than the configured ratio, the update process starts.
The first action is to determine which entries needs to be updated. The process maintains the variable gps_needed_entry for this purpose, which is built before each network request. This variable is simply a copy of the gps_outdated_entry of the used device. If the two devices are used, the gps_needed_entry will be the copy of the MT device since this one can automatically update its almanac via the satellites and so entries will be fresher than the LR ones.

The request is sent towards the network and a timer is started for 120 seconds. If the timer elapses, the process is aborted and state update is entered If at least one almanac entry has been received,

The almanac answer (or a command) can be received asynchronously at any time. In this case the local cache is updated and an another request is done if there are still missing entries to reach the target validity ratio.

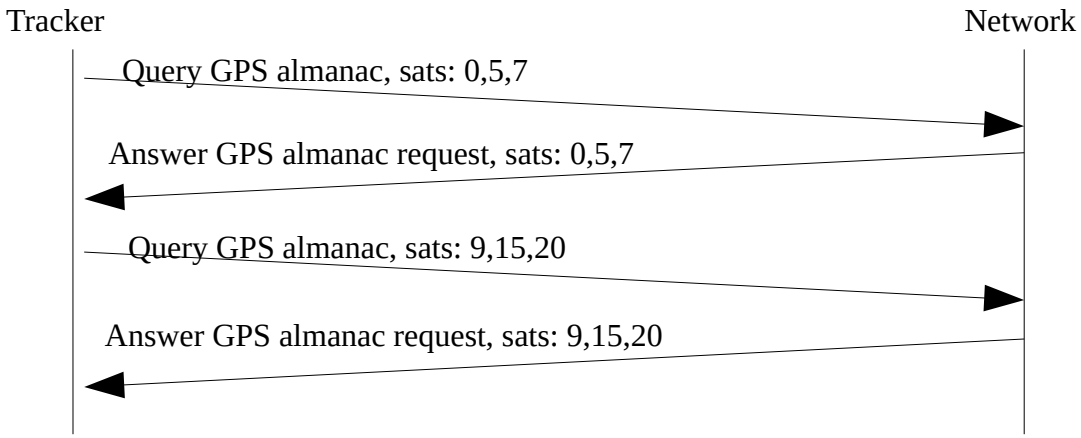
Note that the request is done for a maximum of 3 satellites.
Once all requests have been received, the process enters the state updating. The device almanac is actually updated from the local almanac cache.
If both devices are used, the update starts with the LR device and ends with the GNSS device.

Once the entry has been successfully written to a device, the manager updates the outdated bitmap

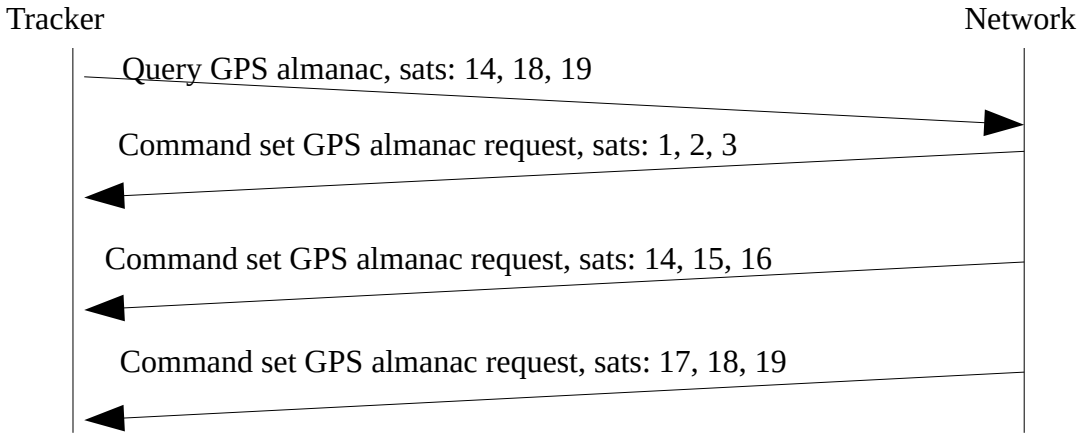
related to the device and the constellation.
At the end of the update process, the manager set the time for the next refresh (24 h).

The diagrams below show the two type of messages

Using the answer downlink



Using the command downlink

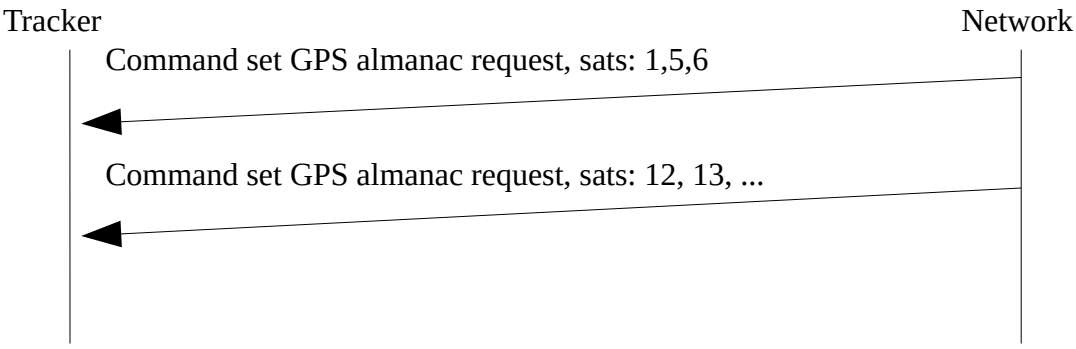


Update process initiated by the network

At any time the network may decide to update the almanacs using a command. For example, this decision can be taken on the reception of the status message or when receiving an almanac update request.

The process is identical than the one initiated by the device, except that the command may contain up to 15 almanac entries (tuned to fit the network capability. Once the set GPS command is processed, the device may requests missing entries.

The message diagram is simply:



Special case for LoRa

LoRa requires an uplink to trigger a downlink. It is very unusual that the network cloud answers a tracker request within the two RX windows following the request. So, another uplink is required to trigger the downlink transmission.

Since the GNSS manager has no idea when such an uplink will be sent, a smart process has been implemented to cover this case. Once the uplink request is sent, the system changes temporarily the dl trigger period (which usually sends periodically an empty downlink to trigger the reception). The timer is set to 20 seconds (hardcoded value). This means that while waiting for the answer (during 120s), at least one uplink will be sent. Once the empty downlink has been sent, the DL trigger period comes back to the configured value.

10.3 Aiding-position update

LR1110: Based on the Doppler error of entries in each scan, the tracker may request an aiding-position request.

MT3333: No aiding position support?

Network: If the tracker is static and several positions fails the position resolution, the network could send an aiding-position update.

Questions

If the tracker has few satellites in view, what we should do:

1. Request an almanac update?
2. Request an aiding position?
3. Do nothing?

11 Geolocation manager

AT2 defines an operational mode used to select the geolocation mode (permanent, start/stop, motion, and so on).
AT3 does not use such a concept. Instead, it is event based: application events trigger the geolocation activities.

The geolocation manager responsibilities are:

- Handling the applications events defined as geolocation triggers and start accordingly the geolocation process.
- Supporting the periodic position acquisitions including the SOS mode.
- Performing consecutive position acquisitions if needed (for motion start or stop).
- Managing the priorities between the requests due to the geolocation triggers.

11.1 Configuration and behavior

This section is dedicated to the customers and the support. It highlights the geolocation configuration of the manager and associated behavior

11.1.1 Configuration

The configuration is done using the configuration class 1 (geolocation). It defines various periods for periodic (or semi-periodic) acquisitions, counters for motion-start and motion-stop events, event triggers profiles and GBE (Geolocation Basic Engine) profiles.
This paragraph focuses on the triggers and GBE profiles.

The event trigger parameters 0, 1 and 2, associated to the corresponding geolocation profiles 0, 1 and 2 define the list of events for which a geolocation activity should be performed. The geolocation activity will be profiled by the matching profile which define the technologies to use and their sequencing.
An event trigger is a bit-field containing the list of events that should use the same profile. The bit definitions are:

- bit 0: geo_trigger_pod: Geoloc triggered on Position-on-demand via downlink or via button.
- bit 1: geo_trigger_sos: SOS started
- bit 2: geo_trigger_motion_start: Geoloc triggered on motion start event
  - Require the configuration of geoloc_motion_nb_start and geoloc_start_stop_period
- bit 3: geo_trigger_motion_stop: Geoloc triggered on motion stop event
  - Require the configuration of geoloc_motion_nb_stop and geoloc_start_stop_period
- bit 4: geo_trigger_in_motion: Periodic geoloc while the tracker is in motion
  - Require the configuration of geoloc_motion_period.
- bit 5: geo_trigger_in_static: Periodic geoloc running while the tracker is static

    - Require the configuration of geoloc_static_period.
- bit 6: geo_trigger_shock: Geoloc triggered on shock action
    - Require the shock detection configured (accelerometer)
- bit 7: geo_trigger_temp_high_threshold: Geoloc triggered on temperature high.
- bit 8: geo_trigger_temp_low_threshold: Geoloc triggered on temperature low.
- bit 9: geo_trigger_geozoning: Geoloc stopped while in monitored area. Enabled when leaving the monitored area.

11.1.2 Event priority groups

Each event belongs to a priority group. This priority is used by the manager to properly schedule (or abort and reschedule) the geolocation. The priority is based on the nature of the event and not on the profile it is defined.

The priority groups are hard-coded and defined as follow:

Priority	Events
0 (lowest)	Periodic events group (moving, static)
1	Motion events group (motion-start and motion-end)
2	User event group (POD, shock detection). Non critical events
3	System event group (Temperature low/high)
4	SOS group
5	Geozoning.

Notes

- If a given event is defined in two profiles, the geolocation profile used for this event will be the one of the profile having the lowest identifier. This means that a given event can have only one geolocation profile.

  Example: Both geoloc_profile0_triggers and geoloc_profile1_triggers have geo_trigger_pod bit set (bit 0), the geolocation profile used for a POD will be geoloc_gbe_profile0_techno.
- In the rest of the document, we use the term:
    - **periodic events** for *in_motion*, *in_static* events and *sos* events.
    - **semi-periodic events** for *motion-start motion*-stop.

11.1.3 Behavior for cases involving interactions between events

The main objectives of the geolocation manager are:

- To respect the event priority. An event with a higher priority takes precedence regarding the geolocation over a lowest priority.
- Avoid having more than necessary geolocation acquisitions: If an event B occurs while an acquisition is in progress for an event A and both have the same profile, don't abort the current

acquisition (event if event B has higher priority). Instead, indicate that the geolocation result relates to both triggers.

- To manage properly extra-positions for motion-start or motion-end events: When the multiple acquisitions are in progress for a motion event (start or end) and the opposite event occurs the multiple acquisition stops for the previous event (and may restart for the opposite event if configured for).
- To avoid having no geolocation when periodic triggers are in use (this case may occur when the tracker continuously moves and stops within the configured periods).
- Since events can have different geolocation profiles, the trigger events are not lost (unless an opposite event occur).
- When SOS is active for the geolocation point of view, all other triggers except the geozoning do not apply
- When the geozoning is active (NOT YET SUPPORTED – TO UPDATE) :
  - Inside the monitored area, the geolocation is disabled.
  - Outside the monitored area, the geolocation is enabled.

11.1.4 SOS management

The SOS geolocation is not active if the geozoning (NOT YET SUPPORTED – TO UPDATE) indicates that we are in a monitored area. In this case, the SOS is managed directly by the geozoning.

When we are not in a geozoning monitored area and the SOS start event is received, all triggers are reset except the SOS. This means that the current triggers are removed and subsequent events (except geozoning) are discarded.

Once leaving the SOS, only the periodic triggers are reactivated.

11.1.5 Periodic and semi-periodic triggers management

The periodic and semi-periodic events generate acquisitions spaced over time. This section discusses about the timing management for such triggers except the SOS. We remind that the SOS cannot be interrupted since it has the highest priority.

To properly space over time the geolocation activities, the geolocation manager uses a timer for which the timeout can be dynamically calculated on certain circumstances.

The periodic and semi-periodic triggers are generated from the accelerometer events :

- A motion-start event from the accelerometer will setup the *motion_start* and the *in_motion* triggers if they are configured. It also clears the *in_static* triggers if it was set. If the *motion-stop* trigger was previously set and an acquisition is in progress, the *motion-stop* trigger is not cleared unless a different profile belongs to each trigger. Once the geolocation completes, the motion-stop trigger is cleared and the event is reported with both flags stop and start. If the start trigger has a different profile, the ongoing acquisition aborts and is restarted with the new profile.
- A motion-stop event from the accelerometer will setup the *motion_stop* and the *in_static* triggers if they are configured. It also clears the *in_motion* triggers if it was set. If the *motion-start* trigger was

previously set and an acquisition is in progress, the *motion-start* trigger is not cleared unless a different profile belongs to each trigger. Once the geolocation completes, the *motion-start* trigger is removed.

- In SOS mode, from SOS start and SOS stop events

Timing adjustments

A timing adjustment may be needed when the geolocation manager is processing a given periodic/semi-periodic trigger and another periodic/semi-periodic trigger is being activated.

The manager uses the event priority to know whether the timing should be adapted. The processing rules are:

- If the new trigger has a higher priority than the current one, the timing is not adapted and the geolocation starts immediately for the new event.
- If the new trigger has a lower or equal priority than the current one, the timing is adapted as if the previous acquisition was done for the new event, i.e. as soon as the acquisition completes it will be used for the new trigger as well..

Example 1

The tracker is configured for *motion_start* and *motion_stop* triggers. While waiting for the next acquisition related to a *motion_start* trigger (assuming we have multiple acquisitions on start event),, we get a motion-end event. The timer is recalculated to coincide with the next motion start acquisition:

timeout = geoloc_start_stop_period – elapsed_timer_time.

This means that the total time elapsed between the two last geolocation acquisitions remains geoloc_start_stop_period.

Example 2:

The tracker is configured for *in_static* and *motion_start* triggers. The tracker is static, and the periodic acquisition is in progress. While waiting for the next geolocation acquisition, a motion-start event is received. The geolocation acquisition for the motion-start event starts immediately.

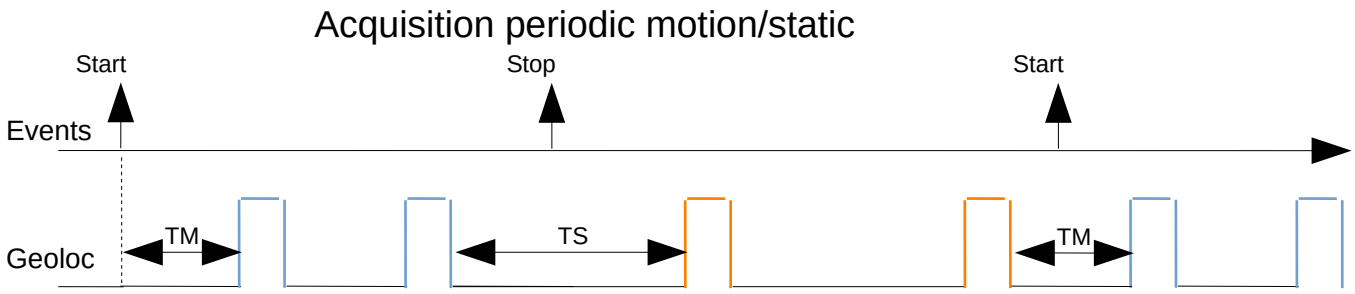
While waiting for the next acquisition due to the motion-start event a motion-end system event (not configured as trigger) occurs. The next periodic static geolocation acquisition will be adjusted to start from end of previous motion start acquisition:

timeout = geoloc_static_period – elapsed_timer_time.

This means that the total time elapsed between the two last geolocations acquisitions is geoloc_static_period .

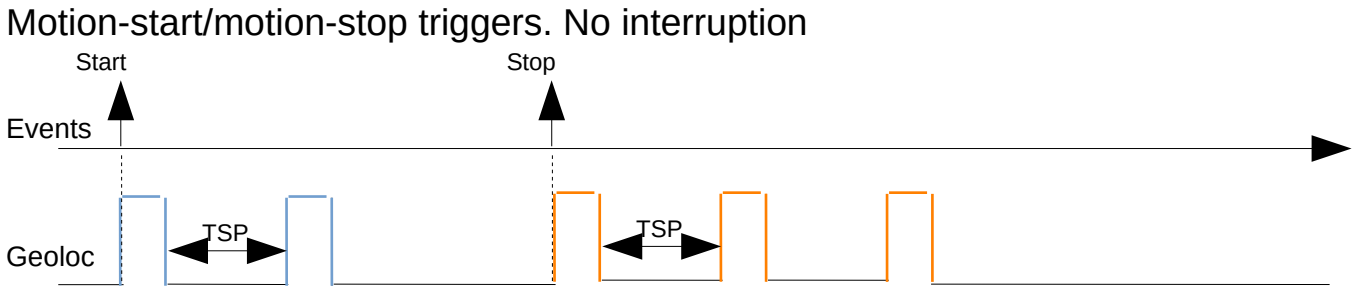
The following chronograms highlights the time spacing between the geolocation acquisitions for periodic or semi-periodic triggers

11.1.6 Geolocation chronograms



TM: motion period
TS: static period

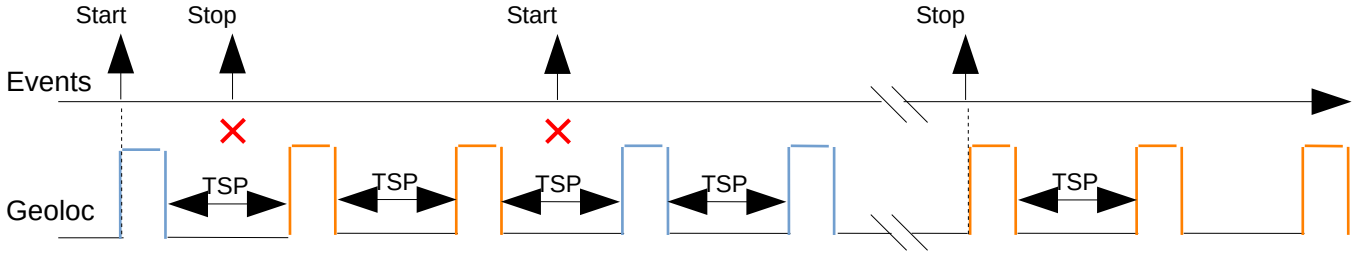
Blue: Acquisition for periodic motion
Orange: Acquisition for periodic static



TSP: Start/stop period
Nb-start: 2
Nb-stop: 3

Blue: Acquisition for motion-start
Orange: Acquisition for motion-stop

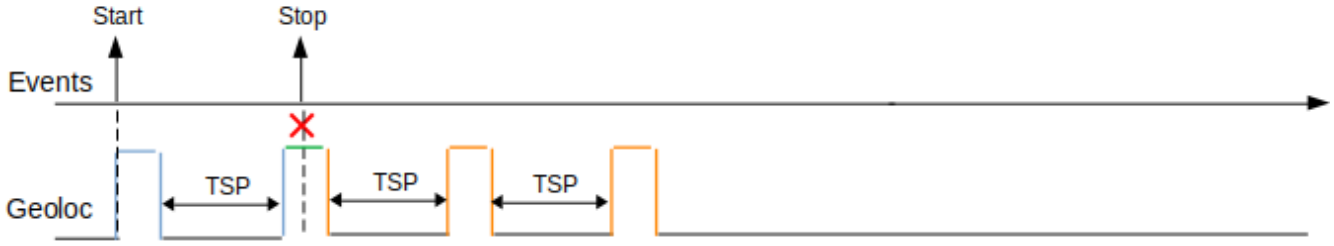
Motion-start/motion-stop triggers with cycle interruption



TSP: Start/stop period
Nb-start: 2
Nb-stop: 3

Blue: Acquisition for motion-start
Orange: Acquisition for motion-stop
Red: Interruption

Motion-start/motion-stop triggers with acquisition interruption. Same profile

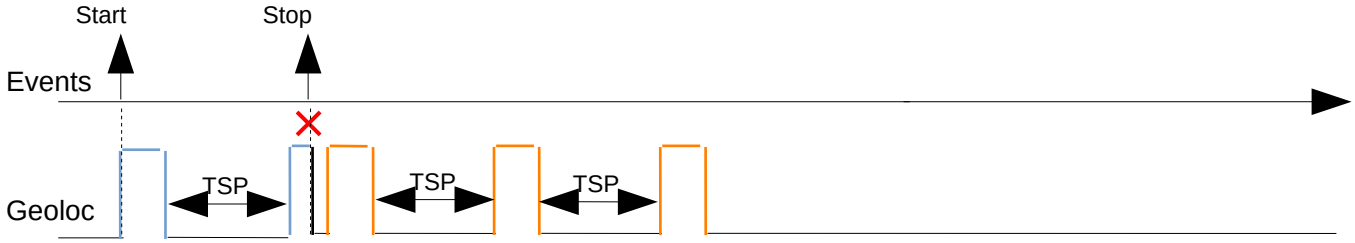


TSP: Start/stop period
Nb-start: 2
Nb-stop: 3

Blue: Acquisition for motion-start
Orange: Acquisition for motion-stop
Red: Interruption

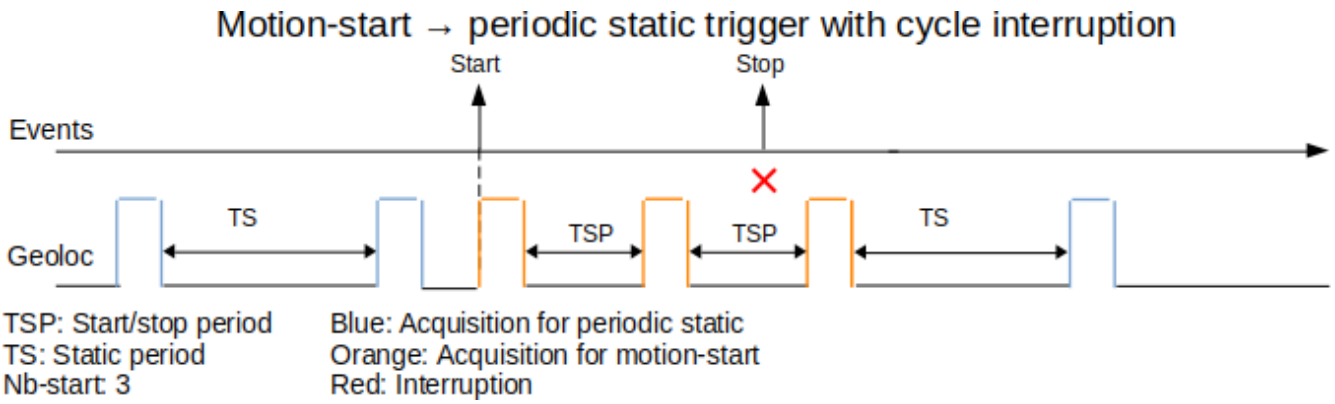
Green: Acquisition for motion-start and motion-stop

Motion-start/motion-stop triggers with acquisition interruption. Different profiles



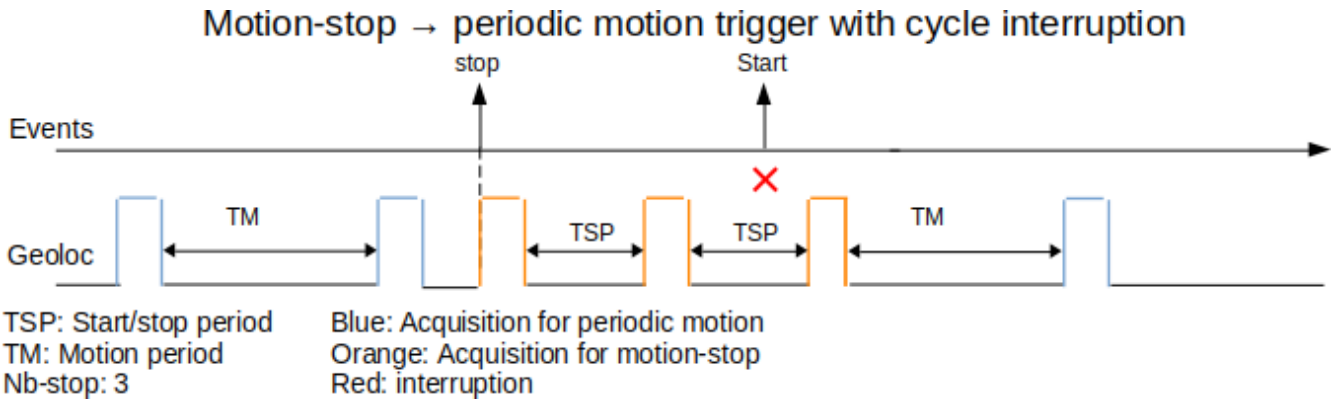
TSP: Start/stop period
Nb-start: 2
Nb-stop: 3

Blue: Acquisition for motion-start
Orange: Acquisition for motion-stop
Red: Abort



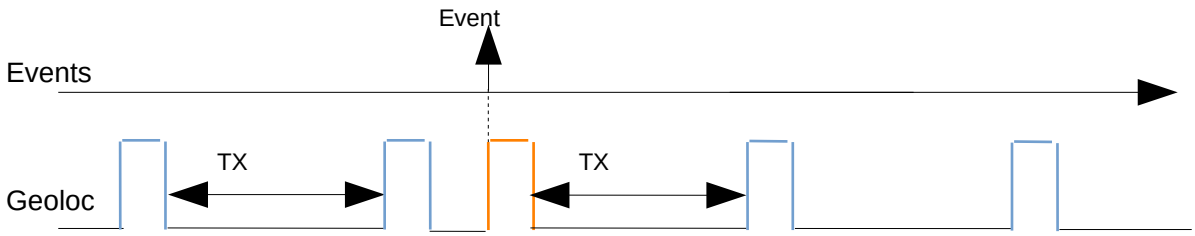
TSP: Start/stop period
TS: Static period
Nb-start: 3

Blue: Acquisition for periodic static
Orange: Acquisition for motion-start
Red: Interruption



TSP: Start/stop period
TM: Motion period
Nb-stop: 3

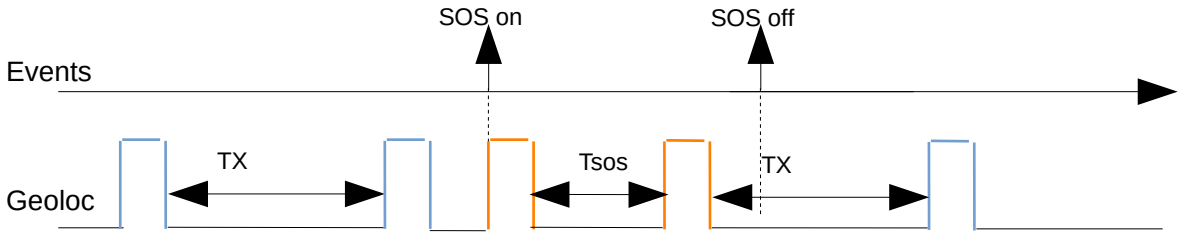
Blue: Acquisition for periodic motion
Orange: Acquisition for motion-stop
Red: interruption



TX: Motion/static period

Blue: Acquisition for periodic motion/static
Orange: Acquisition for event

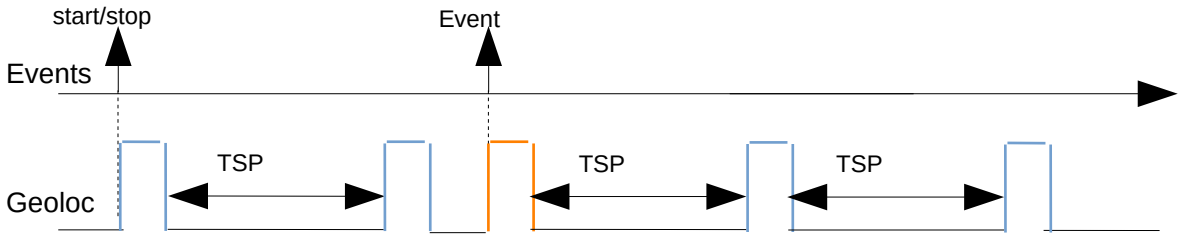
Periodic trigger interrupted by a SOS event



TX: Motion/static period
Tsos: SOS period

Blue: Acquisition for periodic motion/static
Orange: Acquisition for SOS

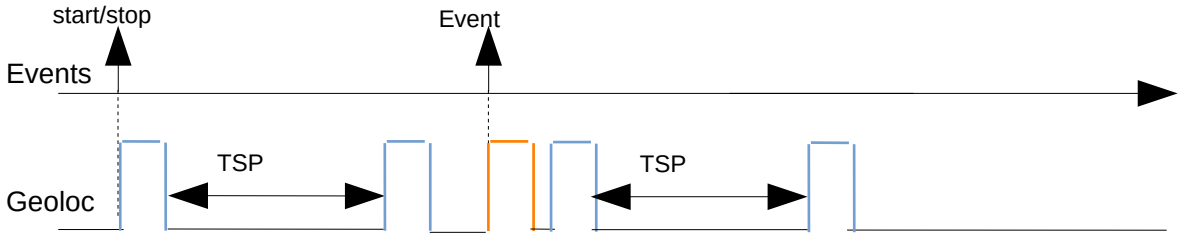
Motion start or stop cycle interrupted by a system/user event. Same profile



TSP: start/stop period
Nb start/stop: 4

Blue: Acquisition for start/stop
Orange: Acquisition for event only

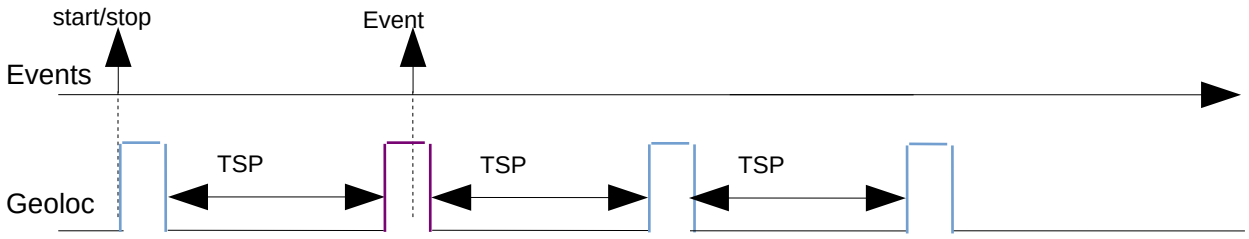
Motion start or stop cycle interrupted by a system/user event. Different profiles



TSP: start/stop period
Nb start/stop: 4

Blue: Acquisition for start/stop
Orange: Acquisition for event only

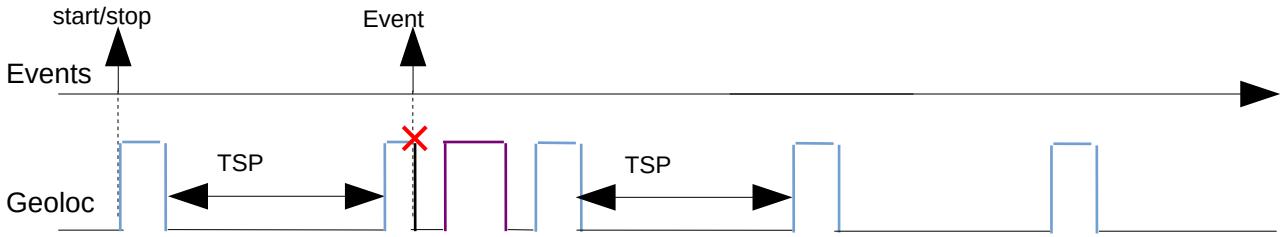
Motion start or stop acquisition interrupted by a system/user event. Same profiles



TSP: start/stop period
Nb start/stop: 4

Blue: Acquisition for start/stop
Purple: Acquisition for both triggers

Motion start or stop acquisition interrupted by a system/user event. Different profiles



TSP: start/stop period
Nb start/stop: 4

Blue: Acquisition for start/stop
Purple: Acquisition for event
Red: Aborted acquisition

11.2 Geolocation Basic Engine (GBE)

The GBE is a basic geolocation machine, working in fallback / forced mode. It replicates and extends the geolocation modes and the BLE scan collections available in AT2.

11.2.1 Technology scheduling

Overview

The GBE supports several geolocation profiles. Each profile is defined by a a mask and an array of bytes:

- The mask is a bit field indicating the application events belonging to this profile.
- The array of byte provides the list of the technologies.

Each byte in the array defines a geolocation technology. The GBE schedules the technologies in the order of the byte array. This means that the technology defined at the first array index is always done and is scheduled first. Then the engine schedules the second technology (if needed) defined by the next index in the array. The process continues until the last technology is done.

Note that the values in the array define the technology to use as well an action indicating that the technology must be either always scheduled or scheduled only if the previous technologies failed to acquire a position .

In order to schedule a BLE scan collection, this technology must be configured with “always done” flag.

Also note that it is strongly recommended to have the “always done” technologies configured as the last technologies.

The GBE supports up to 3 profiles. These profiles are prioritized based on their identifier. The profile 1 has top priority, the second has medium priority and the last has the lowest priority. These priorities are used to know whether a running geolocation process should be interrupted.

For example, the button 1 press is associated to geolocation profile 1. The accelerometer events are associated to the profile 2. Suppose that the geolocation process is running due to an accelerometer event (priority 2) and the user press the button 1. At this stage, the geolocation process is canceled and restarted with profile 1. Suppose now that the geolocation process is running due to a button 1 press and an accelerometer event is received. The geolocation process won’t be stopped and the geolocation request due to the accelerometer event will be discarded.

Configuration

A geolocation profile is defined by two parameters:

- gbe_profileX_triggers(X=1,2,3): 32 bits integer defining the bitmap list of the triggers.
- gbe_profileX_techno (X=1,2,3): Byte array containing the list of technology to schedule.

The bytes of the gbe_profileX_techno parameters are coded as follow:

- Bit 7: Action
- Bits [6..0]: Technology

Available technologies are:

Identifier	Technology
0	None
1	LR11xx_A_GNSS
2	WIFI
3	BLE scan 1
4	BLE scan 2
5	aided_GNSS
6	GNSS

Available actions are

Identifier	Action
0	Skip on success
1	always_done

It is clear that the first technology is always performed. So, it does not require the **always_done** action.

Scheduling rules

The following scheduling rules apply (refer to the chronograms of section Configuration and behavior for details and examples):

1. The geolocation triggers inside a profile are non-exclusive. When the geolocation is already running due to a given trigger inside a profile and another trigger in the same profile is solicited, the geolocation is not restarted. Instead, the geoloc stores the two triggers. These triggers will be reported (as a bitmap) in the application position uplink.
2. If the geoloc timer is running and a trigger is solicited, the timer is stopped and the geoloc is started. Once done, the geoloc will check if the timer need to be restarted:
    - If we were in position acquisition due to nb_start or nb_stop, the timer is restarted with geoloc_start_stop_period if the number of expected positions is not reached, otherwise moves to step #2.
    - Tracker is in motion and geo_trigger_in_motion bit set, the timer is restarted with geoloc_motion_period.

- Tracker is static and geo_trigger_in_static bit set, the timer is restarted with geoloc_static_period.
- Tracker is in the geozoning monitored area and geo_trigger_geozoning_monitor bit set, the timer is restarted with geoloc_static_period If static or geoloc_motion_period if moving.

11.2.2 Technologies scheduling examples

The examples in this section use a single profile.

Single technology

In this example we use only the MT3333 GNSS technology.

gbe_profile1_techno[0] = 6	MT3333 GNSS
gbe_profile1_techno[1] = 0	None
gbe_profile1_techno[2] = 0	None
gbe_profile1_techno[3] = 0	None
gbe_profile1_techno[4] = 0	None

Dual technologies in fallback mode

In this example we do WIFI fallback MT3333 aided GNSS: In case of WIFI success, the GNSS is not scheduled. In case of WIFI failure, the GNSS is scheduled.

gbe_profile1_techno[0] = 2	WIFI
gbe_profile1_techno[1] = 5	MT3333 A-GNSS
gbe_profile1_techno[2] = 0	None
gbe_profile1_techno[3] = 0	None
gbe_profile1_techno[4] = 0	None

Dual technologies in fallback and single collection

In this example we do LR1100 aided GNSS fallback MT3333 aided GNSS: In case of LR1110 success, the MT3333 A-GNSS is not scheduled, otherwise it is. The BLE collection is always scheduled.

gbe_profile1_techno[0] = 1	LR1110 A-GNSS
gbe_profile1_techno[1] = 5	MT3333 A-GNSS
gbe_profile1_techno[2] = 0x83	BLE scan1 + Always_done action
gbe_profile1_techno[3] = 0	None
gbe_profile1_techno[4] = 0	None

Dual technologies in fallback and dual collections

In this example we do LR1100 aided GNSS fallback MT3333 aided GNSS: In case of LR1110 success,

the MT3333 A-GNSS is not schedule, otherwise it is. The BLE and WIFI collections are always done.

gbe_profile1_techno[0] = 1	LR1110 A-GNSS
gbe_profile1_techno[1] = 5	MT3333 A-GNSS
gbe_profile1_techno[2] = 0x83	BLE scan 1+ always_done action
gbe_profile1_techno[3] = 0x82	WIFI + always_done action
gbe_profile1_techno[4] = 0	None

Single technology and three collections

In this example we do MT3333 GNSS. Then we do collections for MT3333 aided GNSS, WIFI and BLE.

gbe_profile1_techno[0] = 6	MT3333 GNSS
gbe_profile1_techno[1] = 0x81	LR1110 A-GNSS
gbe_profile1_techno[2] = 0x83	BLE scan1 + always_done action
gbe_profile1_techno[3] = 0x82	WIFI + always_done action
gbe_profile1_techno[4] = 0	None

Three technologies in fallback single collection

In this example we do LR11110 A-GNSS fallback MT3333 A-GNSS fallback WIFI. Then we do a BLE collection.

gbe_profile1_techno[0] = 1	LR1110 A-GNSS
gbe_profile1_techno[1] = 5	MT3333 A-GNSS
gbe_profile1_techno[2] = 2	WIFI
gbe_profile1_techno[3] = 0x83	BLE scan1 + always_done action
gbe_profile1_techno[4] = 0	None

Four technologies in fallback no collection

In this example we do BLE fallback WIFI fallback LR11110 A-GNSS fallback MT3333 A-GNSS.

gbe_profile1_techno[0 = 3	BLE scan1
gbe_profile1_techno[1] = 2	WIFI
gbe_profile1_techno[2] = 1	LR1110 A-GNSS
gbe_profile1_techno[3] = 6	MT3333 A-GNSS
gbe_profile1_techno[4] = 0	None

11.2.3 Geolocation reporting

The reporting of the geolocation depends on what has been configured. The sending rules are the following:

- If only one technology is present in the list, the reporting is always done regardless the status (success/failure).
- A skipped or aborted geolocation technology is never reported.

- The result of an **always done** technology is always reported even if the technology fails.
- The first scheduled technology of the list is always performed even if the **always done** flag is not set. However the reporting of this technology is done as follow:
  - If the **always done** flag is reset and we are in multi-technology, the reporting is not done.
  - f the **always done** flag is set and we are in multi-technology, the reporting is not done.
- Some cases in multi-technology mode where mixed state of the **always done** flag are used, the failed technologies in fallback mode will not be reported.

Example 1

Technologies used: BLE fallback WIFI fallback GNSS

gbe_profile1_techno = {03,02,06,00,00,00}

On BLE success: only BLE is reported

On BLE failure and WIFI success: only WIFI is reported

On BLE and WIFI failure: GNSS (success or failure) only is reported

Example 2

Technologies used: BLE fallback WIFI. GNSS always-done

gbe_profile1_techno = {03, 02, 86,00,00,00}

On BLE success: BLE and GNSS (success or failure) are reported

On BLE failure and WIFI success: WIFI and GNSS (success or failure) are reported

On BLE and WIFI failure: Only GNSS (success or failure) is reported

Example 3

Technologies used: BLE fallback WIFI. WIFI and GNSS always-done

gbe_profile1_techno = {03,82,86,00,00,00}

On BLE success: BLE, WIFI (success or failure) and GNSS (success or failure) are reported

On BLE failure: WIFI (success or failure) and GNSS (success or failure) are reported

Example 4

Technologies used: BLE always done. WIFI fallback GNSS.

gbe_profile1_techno = {83,02,06,00,00,00}

On BLE success: GNSS only is reported.

On BLE failure and WIFI success, BLE and WIFI are reported.

On BLE and WIFI failure, GNSS success: BLE and GNSS are reported.

On BLE and WIFI and GNSS failure:BLE only is reported.

Example 5

Technologies used: BLE always done. WIFI fallback GNSS. GNSS always done

gbe_profile1_techno = {83,02,86,00,00,00}

On BLE success, BLE and GNSS are reported

On BLE failure and WIFI success: BLE, WIFI and GNSS are reported

On BLE and WIFI failure: BLE and GNSS reported

On BLE and WIFI and GNSS failure: BLE and GNSS are reported.

Example 6

Technologies used: BLE always done. WIFI always done. GNSS fallback

gbe_profile1_techno = {83,82,06,00,00,00}

On GNSS success, WIFI success or failure, GNSS and WIFI are reported

On GNSS failure, WIFI failure, BLE success. GNSS,WIFI and BLE are reported

On GNSS failure, WIFI failure, BLE failure. GNSS and WIFI are reported

Example 7

Technologies used: BLE always done. WIFI always done. GNSS always done

gbe_profile1_techno = {83,82,86,00,00,00}

Regardless the success/failure status of each techno, GNSS, WIFI and BLE are reported.

11.3 GNSS hold-on mode

The GNSS hold-on mode is a special mode applicable only on the MT3333 device. It allows to maintain the main power of the device while it is not actually used for an ongoing location resolution. The motivation is to prepare for a future location resolution event in order to maximize the accuracy of this upcoming resolution, as a GNSS chip position accuracy typically improves over time after the chip is initially powered.

Maintaining the power on of the component, makes future GNSS fixes very quickly and allows a very good position accuracy.

Notes

- The GNSS Hold-on mode is power consuming. It is advised to use only if an accuracy of 5m or better is required, or if the period of the position acquisitions is less than about 1mn. An acquisition after chip restart will typically require 20 to 30s, versus only 1s in hold-on mode, and also the tracking mode of the GNSS typically is more power efficient than the initial acquisition, therefore the overlal power budget of the hold-on mode is comparable or favorable for acquisition periods below 1mn.
- In general we advise using this mode only for rechargeable devices. However there are exceptions, for example to track vehicles on a factory or new vehicle storage parking (very infrequent movements, which last only a few minutes), it will be very beneficial to use the hold-on mode during motion as the stop position will be very accurate and the additional energy cost is

negligible.

- This mode cannot be used in conjunction to the GNSS part of the LR1110. This means that any trigger pointing to a profile having the lr1110_agnss configured will disable automatically this mode. This restriction comes from the fact that the two GNSS components share the same antenna.

11.3.1 Configuration

This mode is configured via the parameters:

- ***geoloc_gnss_hold_on_mode***: Configure the behavior as well the triggers starting this feature. Refer hereafter for the acceptable values.
- ***geoloc_gnss_hold_on_timeout:*** Provide the maximum duration for which the hold-on mode is kept. Once the period elapsed and in absence of events matching the feature triggers, the hold-on mode is stopped and the normal process (standby then power off) resumes. Hold-on mode is deactivated if this parameter is set to 0.

The values that the GNSS hold-on mode can take are provided by the following table.

Value	Mode	Comment
0	None	Hold-on mode deactivated.
1	Always	Hold-on mode always activated. Only controlled by the timer.
2	Techno used	Hold-on mode activated only when the GNSS technology is actually used (meaning GNSS techno not skipped).
3	Moving	Hold-on mode activated only if the tracker is in motion.
4	Static	Hold-on mode activated only if the tracker is static.
5	Techno used and Moving	Hold-on mode activated only when the GNSS technology is actually used (meaning GNSS techno not skipped) and the tracker is in motion.

Notes

1. The ***geoloc_gnss_hold_on_timeout*** is applicable of all modes except **None** (value 0).
2. In modes 2 and 5, the GNSS is immediately stopped if another location technology (e.g. BLE) has been successful and the GNSS technology is configured to be skipped.

11.3.2 GNSS hold-on feature behavior

When hold-on mode is activated, the geolocation manager instructs the GNSS to remain powered, even though it does not need an immediate fix. This lets the GNSS component to continue to refine its fix. When the geolocation manager needs a new GNSS fix (via the GBE), the fix is immediately ready and

usually has a very good accuracy.

It is important to understand that when a hold-on feature trigger occurs the geolocation manager will indicate to the GNSS component that it should keep its power on once it finishes its acquisition. So the first acquired position may be not as accurate as expected since prior the acquisition the component was either sleeping or off.

There are two different triggers related to this feature:

- Motion start/stop: Events indicating whether the tracker is moving or static (used by the mode 3, 4 and 5).
- Technology used. If the GNSS is actually used (meaning GNSS technology not skipped) the hold-mode is activated. This means that if the GNSS is used in fallback mode and another technology is successful, then the hold mode is left (mode 4 and 5).

Note that when the hold-mode is configured to **Always**, it is advised to also configure ***geoloc_gnss_hold_on_timeout*** to limit the power consumption.

Let's take an example to clarify the feature behavior in techno-used mode. Suppose that the GBE has a single profile, which configures the first technology being BLE and the second as GNSS in fallback mode.

In the case where the BLE technology successes, the GNSS hold-on mode is deactivated. It remains inactive until the BLE technology fails. At this stage, the GNSS technology will be actually used and the GNSS hold-mode will be activated. It will remain activated until the BLE technology fails to acquire a position. On BLE scan success, the GNSS hold mode is left.

11.4 Geolocation vs geozoning management

The geozoning feature defines a notion of monitored area, which is defined by BLE beacons matching certain filters. Entering the monitored area means that any type of BLE beacons except **exit** has been detected. Leaving the monitored area means that either a **exit** beacon has been detected or a a configured period of time without visibility of any BLE beacon has been reached.

While inside the monitored area, the geolocation manager ignores all triggers other than the periodic triggers and the SOS (if they have been activated).

Once leaving the monitored area, all triggers are re-enabled.

It also means that the SOS processing is preserved regardless the type of area (monitored or not).

11.5 General design

This section focuses on the design and the implementation. It should not be provided to customers **or**

external partners.
The geolocation manager is built around a finite state machine, which reacts on the applications core events. The configuration provides the applications events that should be taken into account. These events are referred to geolocation triggers events, or simply triggers, in the rest of the chapter.

To reach the above objectives, the manager uses context variables and decision rules. The relevant context variables are:

- **active_triggers**: Bit-map containing the event list for which a geolocation is in progress or waits for the timer expiry. Note that the timer is started for periodic or semi-periodic events.
- **pending_triggers**: Bit-map containing the event list that has been triggered but the geolocation is not yet available (already running for other triggers).
- **current_priority**: Highest priority of the list of events in the active triggers.
- **current_profile**: Current GBE profile identifier used by the geolocation (matches the profile identifier used by all events in the active_triggers).

The rules dictating the behavior of the geolocation manager are:

On core event received:

1. If an event is received while the manager is idle, the active trigger is set with the new event and the geolocation starts ASAP or may be delayed in case of active periodic or semi-periodic event triggers (except SOS).
2. If an event is received while a geolocation is in progress or waits for the timer expiry and has a higher priority than the ones in the active_triggers then:
   a) If the new event geolocation profile is the same than the one being scheduled then we merge the new triggers with the active_triggers ones and start ASAP the geolocation if it is not already in progress. In such a case, the geolocation result will contain all triggers.
   b) If the new event profile is not the same as the one already scheduled, the current geolocation process is aborted, the active_triggers are moved in the pending_triggers (to avoid losing them) as well as the new event. Pending triggers will be processed at step “On geolocation abort done”.
3. If an event is received while a geolocation is in progress or waits for the timer expiry and has a same priority than the ones in the active_triggers then:
   a) If the new event geolocation profile is the same than the one being scheduled then we merge the new triggers with the active_triggers ones and start ASAP the geolocation if it is not already in progress. In such a case, the geolocation result will contain all triggers.
   b) If the new event profile is not the same than the one being scheduled, the event is set in the pending_triggers list.
4. If an event is received while a geolocation is in progress or waits for the timer expiry and has a lower priority than the ones in the active_triggers then we put it in the pending_trigger list. Note that in this case we don’t merge the event with the active_trigger list even if the geolocation profile

is the same. The reason is to avoid of having both motion (start or stop) triggers and periodic triggers.

On geolocation acquisition done

1. Clear all triggers in active_trigger except periodic and semi-periodic (including SOS)
2. If SOS still active, restart timer with SOS timeout, otherwise continue to the next rule.
3. Process remaining motion-start or motion-end geolocation acquisitions if any, otherwise continue to the next rule.
4. Process periodic events in active_trigger if any, otherwise continue to the next rule.
5. Check for new periodic events if any, otherwise continue to the next rule.
6. Process as if core events were received using the pending_triggers,

On geolocation abort done

1. Process as if core events were received using the pending_triggers.

12 BLE scan collection

This section focuses on the BLE scan geolocation technology.

12.1 Configuration

12.1.1 Standard scan parameters and filtering

The BLE scan configuration is done via the configuration group 4.

The ***ble_scan_duration***, ***ble_scan_window*** and ***ble_scan_interval*** define the configuration of the scan.

The parameter ***ble_scan_type*** configures the type of beacon we expect.

The ***ble_scan_min_rssi*** configures the minimum received power level of a beacon for processing. Beacons received below this threshold will be ignored.

The ***ble_scan_min_nb_beacons*** configures the minimum number of beacons in the scan to consider that the result is a success (position is solvable). A number of beacons below this threshold is considered as not solvable. This setting is important regarding the geolocation basic engine since it may impact the decision to skip or continue to the next geolocation.

This service offers an enhanced filtering scheme based on two filters. Each filter contains an offset (***ble_scan_filterX_offset)*** locating the first byte of the advertisement data part. The beginning of the advertisement part is related to the type of expected beacons:

- Eddystone UUID beacons: the offset 0 locates the first byte of the name-space.
- Eddystone URL beacons: The offset 0 locates the URL schema prefix.
- Eddystone TLM beacons: The offset 0 locates the first byte of the battery voltage.
- iBeacons: The offset 0 locates the first byte of the company UUID.
- altBeacons: The offset 0 locates the first byte of the beacon D.
- custom: The offset 0 locates the beginning of the advertisement frame.

The filter mask (***ble_scan_filterX_mask***) with its offset (***ble_scan_filterX_offset*****)** and the filter value (***ble_scan_filterX_value***) define the application of the filter: For each byte of the advertisement data, the process performs a logical AND between the data and the mask from the offset until the actual length of the mask and the result is finally compared to the filter value. If there is no match, then the advertisement is ignored.

Note that the length of the filter is given by the size of the filter (10) minus the number of null bytes starting from the end. Examples:

- A filter mask equals to {FF,FF,FF,FF,00,00,00,00,00,00} has an actual size of 4.
- A filter mask equals to {FF,FF,FF,FF,00,FF,00,00,00,00} has an actual size of 6.
- A filter mask equals to {00,00,00,00,00,FF,00,00,00,00} has an actual size of 6.

The filter match is done on the actual size of the filter. So, If the data part length is less than the actual filter size, the process filters out the beacon.

When configuring the two filters, the beacon will be valid only if the two filters match.

Notes

- The filtering is not applicable if the parameter ***ble_scan_type*** is all-beacon or exposure.
- If the ***ble_scan_type*** is set to custom, it is advised to setup filters unless you expect to receive all type of beacons.

12.1.2 Reporting

The BLE scan reporting is the result of the scan operation. The configuration defines what should be reported via the Network (LoRaWAN or LTE).

The following parameters configure the reporting:

- ***ble_scan_nb_beacons***: Configure the number of beacons that should be reported.
- ***ble_scan_report_id_ofs***: Configure the type of information the report should contain:
  - MAC address: Beacon MAC address.
  - Short beacon ID: Identifier on 2 bytes extracted from the advertisement data.
  - Long beacon ID: Identifier on 16 bytes extracted from the advertisement data.
- ***ble_scan_report_id_ofs***: Locate the first byte that should be extracted from the advertisement data to form the short or long beacon ID.

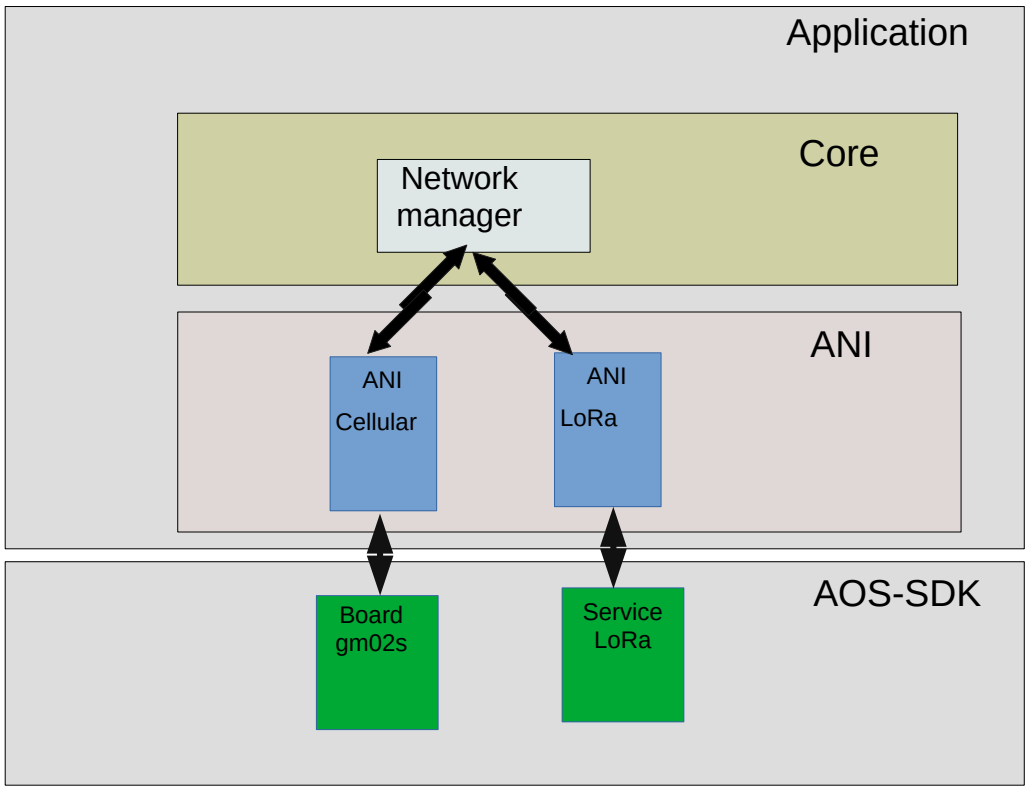
Notes

- The ***ble_scan_report_id_ofs*** parameter is applicable only if the ***ble_scan_report_id_ofs*** is either short or long beacon ID.
- As for the filtering, the beginning of the advertisement part is related to the type of expected beacons. Refer to the filtering section to locate properly the first byte to form the beacon ID.
- The short beacon ID cannot be used is the ***ble_scan_type*** is all-beacon or exposure.

13 Networking

13.1 Overview

The networking subsystem is able to handle both LoRaWAN and cellular networks. It is designed as follows:



13.2 Network manager

The network manager is responsible of:

- Managing the network connectivity state
- Switching between the LoRaWAN or the cellular networks based on availability of the primary network.
- Routing the application messages over a cellular or LoRaWAN generalized socket.

13.2.1 Network connectivity management

The network manager drives the LoRaWAN or the cellular ANI in term of connectivity. The cellular and LoRaWAN networks cannot be active at the same time due to antenna sharing.

The network connectivity management is built around a FSM.

Operations

The network connectivity is managed differently based on the *net_type_selection* parameter:

- lorawan_only: The network manager uses only LoRaWAN. It never switches to the cellular network
- cell_only: The network manager uses only the cellular network. It never switches to LoRaWAN
- lorawan_fallback_cell: The network manager uses LoRaWAN by default. In case of LoRaWAN network loss, it switches to the cellular network. LoRaWAN is considered as the main network, while the cellular is considered as the backup network.
- cell_fallback_lora: The network manager uses the cellular network by default. In case of network loss, it switches to the LoRaWAN network. Cellular is considered as the main network, while LoRaWAN is considered as the backup network.

When the manager is in fallback mode (backup network active), it temporarily suspends the network operations and attempts to reconnect the main network.

FSM states

- **Off**: Network manager is down.
- **Main-connecting**: The connection to the main network is in progress
- **Main-connected**: The main network is connected
- **Backup-connecting**: The connection to the backup network is in progress
- **Backup-connected:** The backup network is connected
- **Main-checking**: The backup network is suspended. The network connection manager tries to reconnect the main network.
- **Wait**: Wait for the end of the inter-connection duration

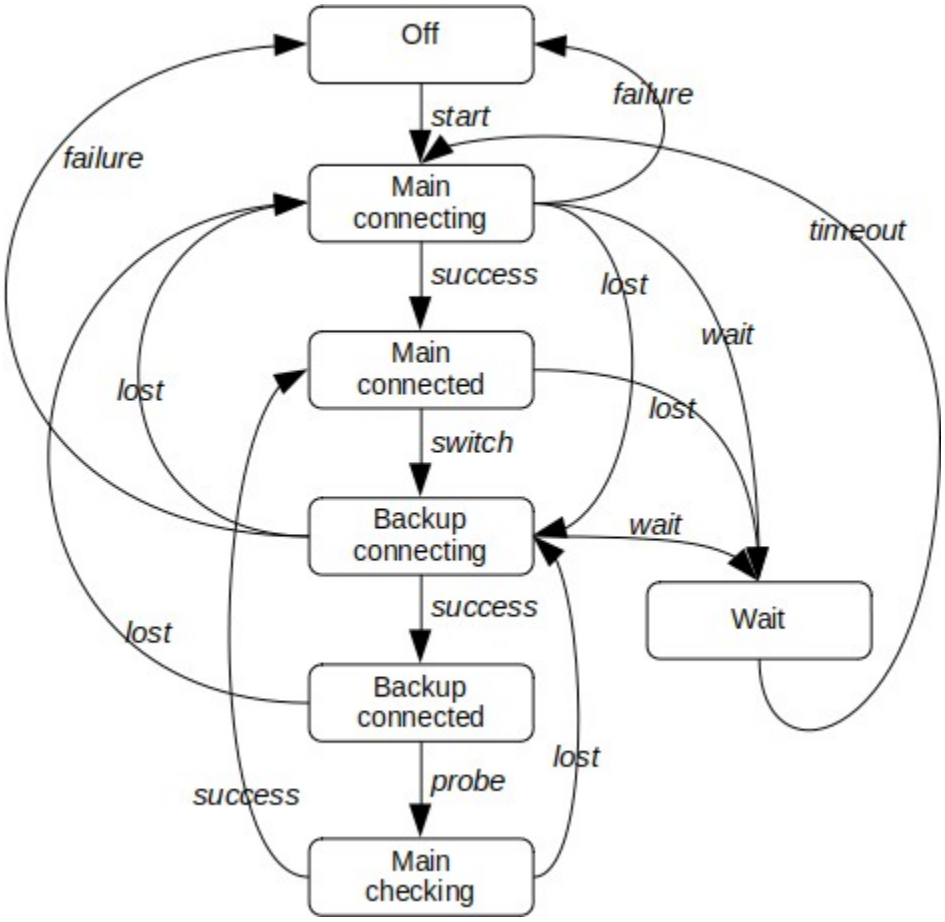
FSM events

- *start*: Start the networking.
- *failure*: Unrecoverable failure (SIM error for example in case of cell-only). In case of both networks used failure is generated only if both networks have unrecoverable failure.
- *lost*: Network lost or fails to connect (recoverable failure)
- *switch*: Switch between the main and backup network. This event is generated when the main network is lost and the network switching is allowed by configuration. If not allowed to switch, this

event is not generated and a lost event is sent to the FSM.

- *probe*: Start the main re-connection.
- *wait*: Inter-connection spacing requested. This event is sent only if the connection spacing timeout is configured and when:
  - There is no backup network defined or in permanent error.
  - We are in **backup-connecting** state (meaning that the main network failed), and the backup network also fails.
- *timeout*: connection spacing timeout

FSM diagram



13.2.2 Application message routing

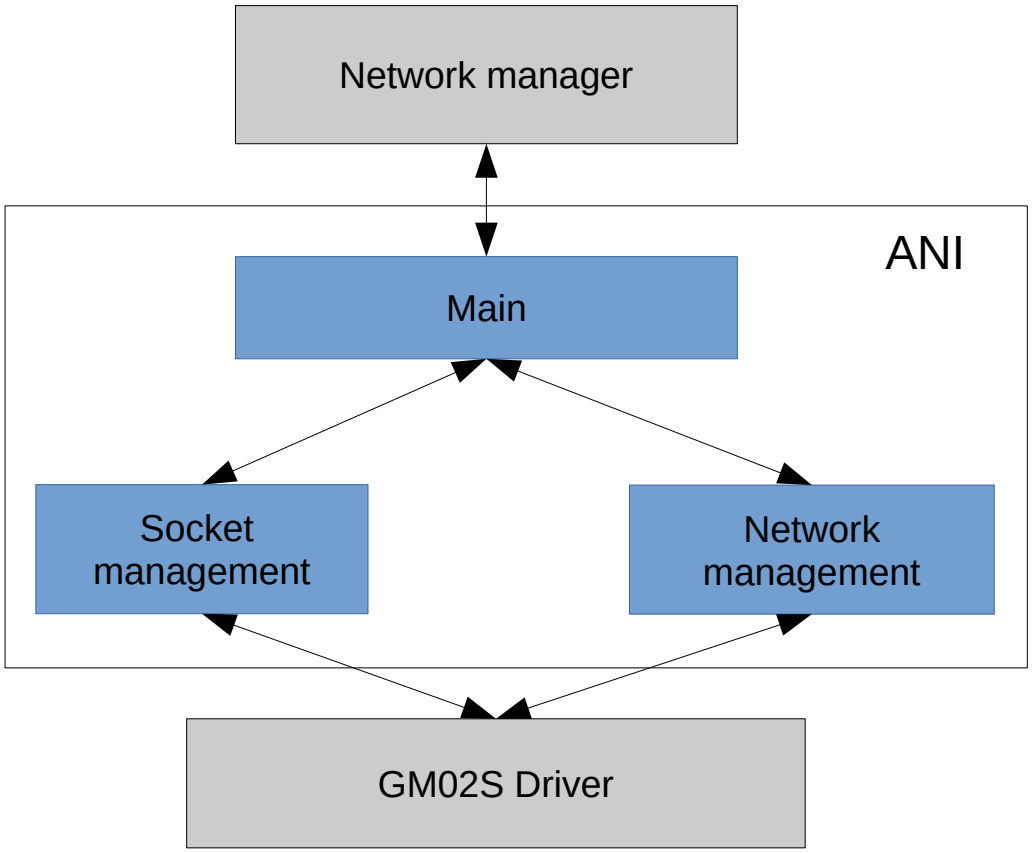
The network manager offers data several pipes to the application. It maps a given data pipe to a given cellular UDP/TCP socket or a LoRaWAN “socket”.

13.3 Cellular ANI

The cellular ANI performs the following tasks:

- Drives the cellular driver (Sequans GM02s module supporting both LTE-M and NB-IOT).
- Manages the cellular connectivity (configuration, network attachment/detachment,).
- Manages the UDP/TCP sockets
- Manages the messages received from the LTE network and forwards them to the network manager.
- Manages the messages sent by application (via the network manager) and forwards them to the network.

ANI design



The interface with the Network manager is based on direct call from the Network manager to the ANI and

notification from the ANI to the Network manager is done via a callback.

ANI operation

- The Network manager opens the ANI for LTE connectivity
- The request is forwarded to the socket management
- The socket manager opens the cellular network management
- The connection management forwards the connectivity result to the socket management
- On connection success, the socket management opens its socket(s) and informs the Network manager of the success.
- On connection failure, the socket management informs the Network manager of the failure.
- Once connection is established and a network failure is reported by the connection management, the socket management closes its socket(s) and informs the Network manager;.

13.3.1 Main part

The main part of the ANI drives the network management and the socket management.

13.3.2 Network management

The cellular network requires a SIM card containing information about the LTE operator, credential and services. Depending on the Abeeway hardware platform, up to two SIMs are available:

- SIM0 connected to the GM02S on the SIM slot 0
- E-SIM connected to the GM02S on the SIM slot 1

The SIM0 is the usual SIM card provided by the LTE operator that we plug in our mobile phones. A dedicated hardware socket is available for this purpose.

The E-SIM is an embedded SIM welded on the hardware board (if mounted). This SIM has a unique identifier called ICCID (Integrated Circuit Card Identifier), which should be registered against the E-SIM provider. The provider will grant the different operators and services that the system can access.

13.3.2.1 Cellular connection management

Overview

The main ANI part controls the cellular network (open and close).

The modem notifies (unsolicited notification) the status of the connection as well as its establishment. This notification is the CEREG URC, which contains the following status:

- 0: No search
- 1: Registered against the home network
- 2: Searching
- 3: Registration denied

- 4: Out of network coverage
- 5: Connected via roaming
- 80: Network temporarily lost.

When using a eSIM, It has been observed that there are multiple CEREG notifications, indicating probably the internal state of the modem particularly when it switches from different operators and bands. In this case CEREG status 0, 2, 3 and 4 can occur during the network search. Due to this, the cellular connection management cannot directly rely on all CEREG notification.

The cellular connection management enables the SIM status unsolicited notification, which provides the following states:

- 0: No SIM card detected.
- 1: SIM under initialization
- 2: SIM locked (PIN or PUK required)
- 3: SIM invalid
- 4: SIM card failure
- 5: SIM card ready
- 6: PH-NET pin required
- 7: Phone-to-SIM password required
- 8: Invalid SIM card in PS (Packet Switched) domain
- 9: Invalid SIM card in PS (Packet Switched) and CS (Carrier Switched) domain
- 10: Invalid SIM card in CS (Carrier Switched) domain

The cellular connection management is based on a finite state machine (FSM) described below. It is driven by the main part of ANI.

Network probing

In order to prevent the cellular connection being blocked, it is possible to enable a network probing mechanism. It is configured via the configuration parameter *cell_probe_max_attempts* and *cell_probe_period*. Setting *cell_probe_max_attempts* to a non null value enables the mechanism. At each period, a query *echo-request* is sent toward the network and expects an reply. After the *cell_probe_max_attempts* queries without answer, the network is declared as lost. Note that the probe period is restarted each time a valid downlink is received.

Contextual variables

- state: Current state of the FSM
- timer: Timer used by the FSM
- reason: Disconnection reason.
- connected_once: Boolean indicating whether cellular network has been connected at least once.

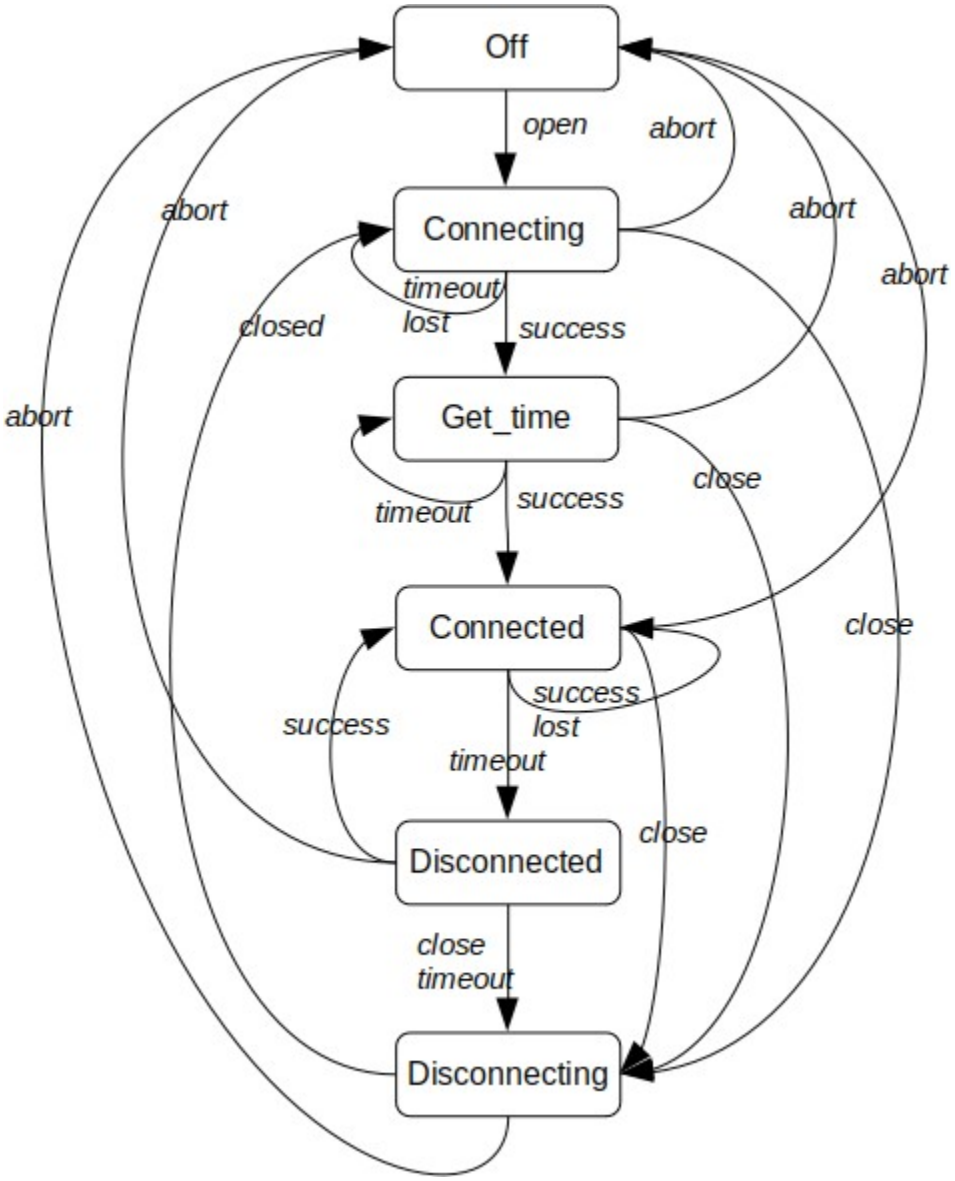
FSM states

- Off: The modem is shutdown
- Connecting: The modem is attaching to the network.
- Get_time: Request the UTC time to the network.
- Connected: The modem is attached to the network.
- Disconnected: The network has been lost.
- Disconnecting: The modem is being disconnected.
- Suspending: The modem is being suspended.
- Suspended: The mode is suspended.

FSM events

- *open*: Administrative event sent by the network manager to open the cellular connection or when the suspended state should be left.
- close: Event sent by the network manager or by the ANI itself if there are no more sockets open.
- *timeout*: Timer elapsed.
- *success*: Successfully connected and attached to the network (Got CEREG 1 or 5).
- *get_time*: Get time request successfully answered.
- *lost*: Network temporary lost (CEREG=80).
- *closed*: Sent to the FSM once the disconnection is complete
- *abort*: Abort requested by the manager to shutdown the network (non graceful close).

State diagram



Notes

- In the state Connecting: Each *timeout* event reset and restart the GM02S and increases the attempts counter. After n attempts, the *abort* event is generated.
- In the state Get_time, if the network has been connected at least once, the state is skipped.
- In the state Get_time, the request can be sent up to 3 times. The ***timeout*** event for this state is generated each 15 s. If the request definitely fails after 3 attempts, the FSM moves in the connected state (since there can be other sources to get the UTC time, e.g LoRa or GPS).
- In the state Connected, if a lost event occurs, the timer is started letting time for the modem to recover the network. When it elapses, the event timeout is generated.

13.3.2.2 Cellular socket management

The socket management is built around a finite state machine (FSM) described below. The FSM is replicated for each configured sockets.

We remind that the GM02s driver is able to process one socket at a time. So the cellular socket management should serializes the opening/closing socket requests.

The cellular network management sends the *sock_open* to the socket FSM whenever the cellular network manager enters the Connected state and sends the *net_down* event whenever the network manager reaches the Off state.

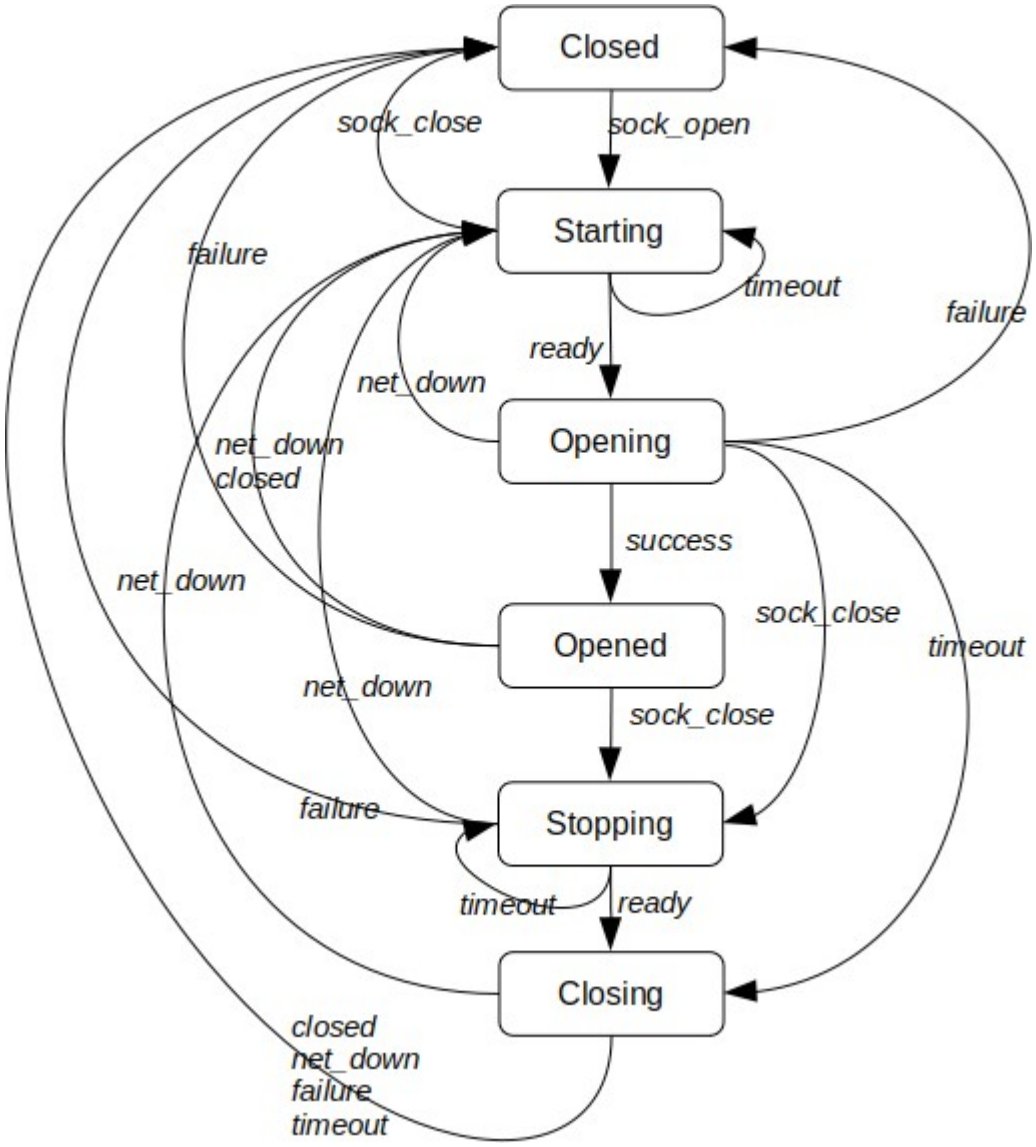
FSM states

- Closed: The socket is closed.
- Starting: The socket should be open and is waiting for either the network up or the GM02S ready to send socket requests.
- Opening: The socket is being opened.
- Opened: The socket is open. Data traffic can flow
- Stopping: The socket should be close and is waiting for the GM02S ready to send socket requests.
- Closing: The socket is being closed.

FSM events

- *sock_open*: Event issued to open the socket
- *sock_close*: Either an administrative close request has been issued either by the above-layer/configuration or due to a failure.
- *ready:* Event issued when the network is up or when the GM02S is ready to accept requests.
- *net_down*: Event issued by GM02S driver indicating that the network is down.
- *success*: Event triggered by the GM02S driver indicating that the socket opening is successful.
- *failure*: Event triggered by the GM02S driver indicating that the socket opening failed or if the socket has been remotely closed or if a driver socket related error occurs.
- *closed*: Event triggered by the GM02S driver indicating that the socket is closed.

FSM diagram



Notes

- The *ready* event is sent by the Network FSM when it reaches the connected state.
- The *net_down* event is generated by the Network FSM when the state Off is entered.
- In state Opened, the data traffic can flow if and only if the network FSM is in state Connected.
- Only one socket gets the *ready* event at a time.
- The *net_down* event is sent to all sockets.

Dynamic configuration

To be properly configured the socket should have at least the destination IP/URL address and the destination port.

When configured and the network opened but not connected, the socket goes in state starting and remains in this state until the network is reachable.

When the network is connected and the socket is unconfigured (port = 0 or IP address empty), the socket moves to the close state. In addition if there is no more sockets opened, the network is shutdown.

Transmission over a socket

Each socket has its own transmission queue handling up to 5 messages.

In order to optimize the power consumption, the actual transmission follows the rules:

- Send ASAP if the network is connected and the modem is active (not in deep sleep).
- Postponed by the TX aggregation timer if the network is connected and the modem is sleeping and the transmission queue is not almost full (less than 4 buffers)
- Send ASAP if the network is connected and the queue is almost full (greater than or equal to 4 buffers including the new one)
- Send ASAP when the aggregation timer elapsed and the network is connected.,
- Postponed until recovery if the network is disconnected.

Note

- When the opportunity of transmission occurs, all buffers are sent.
- When the transmission queue becomes full (can arise if network disconnected), the oldest message is removed from the queue.
- Each socket has its own TX aggregation timer (and configuration)

13.4 LoRa ANI

The LoRaWAN ANI controls the LoRaWAN part of the LR1110. It sits on the top of the LoRaWAN service. As of today, only the LoRaWAN class A is supported. Next releases will address also class B.

ANI operation

- Attach to the LoRaWAN network (Join sequences).
- Leave the network (useful when LTE is also present on the board). Note that both LTE and LoRa cannot run at the same time (antenna sharing).
- Support multiple lora-sockets. Each sockets are characterized by a specific UL port and a specific DL port. The term lora-socket (which is not an usual UDP/TCP socket) is generalized by analogy to TCP/IP over LTE.
- As LTE, each lora-socket has it own queue to send traffic (uplinks).

- Support multiple type of transmission strategies (ADR, random, custom, and so on)
- Support single or dual transmissions.
- Probing the network once joined
- Send periodic empty uplinks to trigger downlinks.

Network probing

A non-null value of parameter lorawan_probe_max_attempts starts the LoRaWAN network probing. One the duration lorawan_probe_period is reached, a link-check request is sent. Once the number of max attempt is reached, the network is considered as lost.
If network probing fails, the ANI attempts to rejoin.

Join process

AT3 uses the LBM randomization and datarate distribution (which consists of an array of 16 entries). Each entry is a datarate. The LBM selects randomly one entry at a time. Once used, the entry is tagged to avoid reusing it. AT3 fills the array as follow:

- Non US915 regions: The sequence {DR0, DR1, DR2} is repeated 5 times. The last entry is set to DR0.
- US915: The sequence {DR0, DR0, DR0, DR0, DR0, DR0, DR4} is repeated twice. The last entry is set to DR0.

During this process, a connection timer may be started (parameter lorawan_cnx_timeout not null). Once the timer elapses, a timeout event is sent to the FSM and a LoRa leave is done.

The ANI is built around the following FSM.

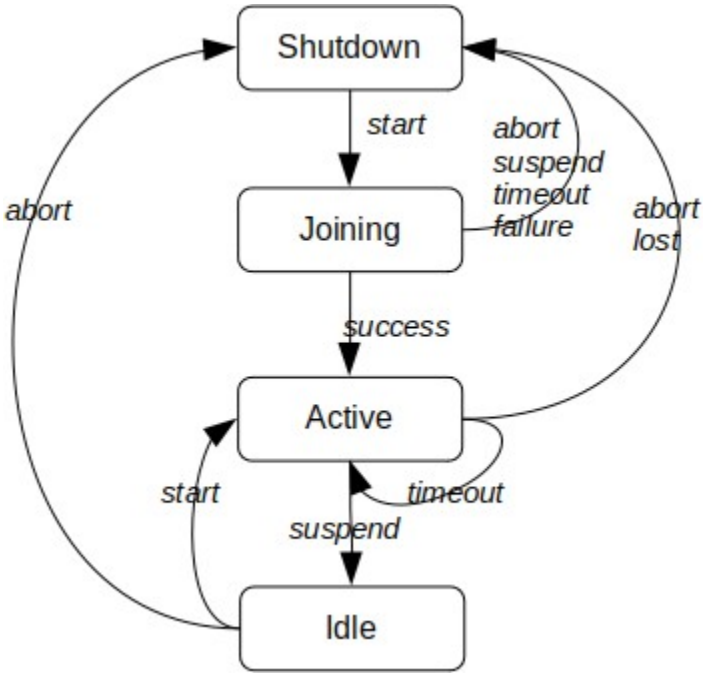
FSM states

- Shutdown: Network not joined.
- Joining: The join process is in progress.
- Idle: Network joined but not accessible (e.g. LTE is active).
- Active: Network accessible (joined and no LTE).

FSM events

- *start*: Start the join process or resume the LoRa activity
- *suspend*: Suspend LoRa
- *timeout*: Timer expiry
- *failure*: Unrecoverable failure (no LoRa socket)
- *success*: Join success
- *lost*: Network probing failure.
- *abort*: Leave and stop the network.

FSM diagram



Note

The *timeout* event in the active state is triggered by the network probing timer, which triggers a link-check request.

13.5 Operational modes and configuration

This section covers the networking configuration as well as the operational modes. Relevant configuration is handled by the dedicated configuration groups. Note that in case of non LTE capable trackers, the cellular configuration group is not accessible and the network group has a reduced parameter set.

13.5.1 Configuring LoRaWAN

LoRaWAN is configured via the LoRa parameter group. The relevant parameters are:

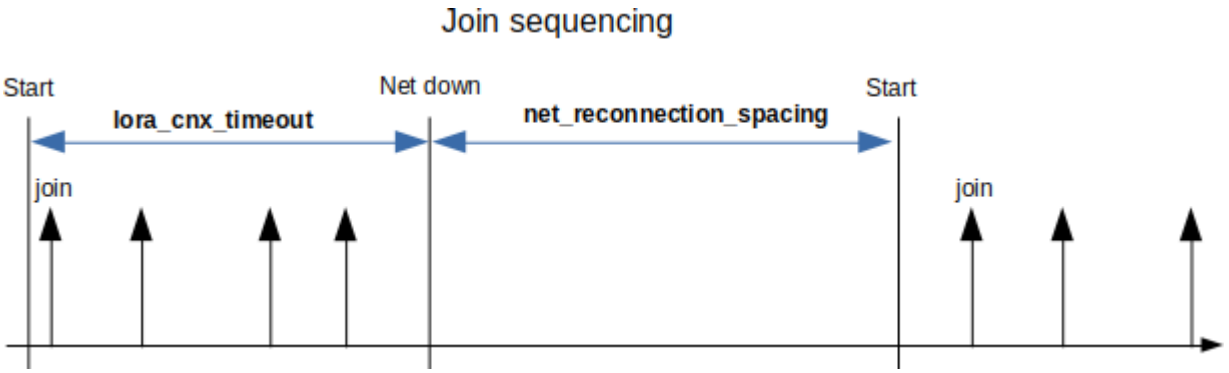
- **net_selection**: Network selection has the value 0 (LoRa only).
- **net_reconnection_spacing**: Duration in seconds between the network down detection and the reconnection with a new join attempt.
- **lorawan_cnx_timeout**: Maximum duration for successfully joining a LoRaWAN network. Once this delay expires, the LoRaWAN network is considered as down. A null value will let the join mechanism running indefinitely. To emulate an Asset-tracker II behavior, this parameter should

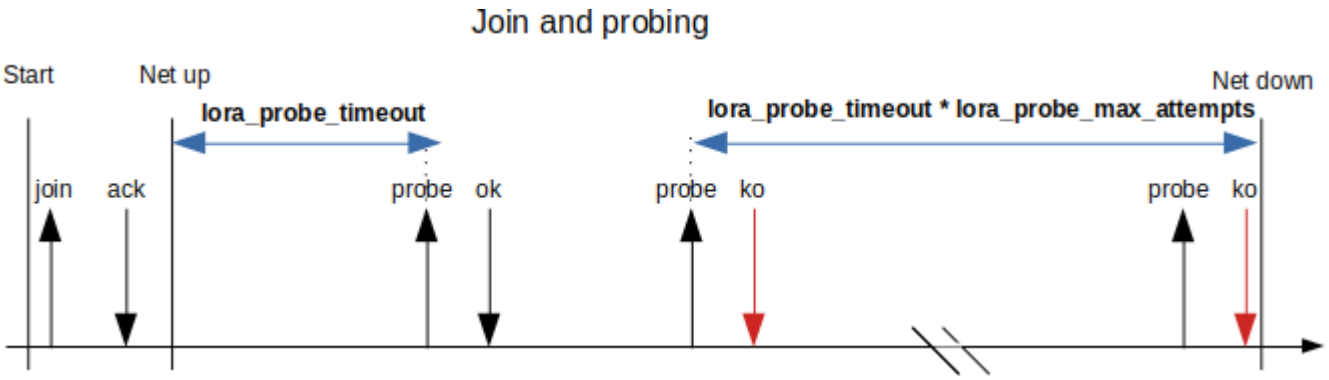
be set to 0. In that case, only the LoRa join duty-cycle will space the different join attempts.

- **lorawan_dl_trigger_period:** Value in second used by a dedicated timer. Once the timer elapses, an empty uplink is sent to trigger an eventual downlink. A value of 0 disables this timer. The timer is restarted each time any uplink is sent.
- **lorawan_probe_max_attempts**: Number of link-check requests sent before declaring the LoRa network down. Note that, the link-check mechanism is started only if the network has been joined.
- **lorawan_probe_period**: Duration between 2 link-check requests.

Note that the l*orawan_probe_max_attempt*s * *lorawan_probe_period* represents the maximum time during which the LoRa network will be unavailable. Once this duration elapses, the network is left and considered as down. The network manager will reconnect the LoRaWAN network if needed. The reconnection will restart with a join sequence.

The following diagrams provide the different timings.



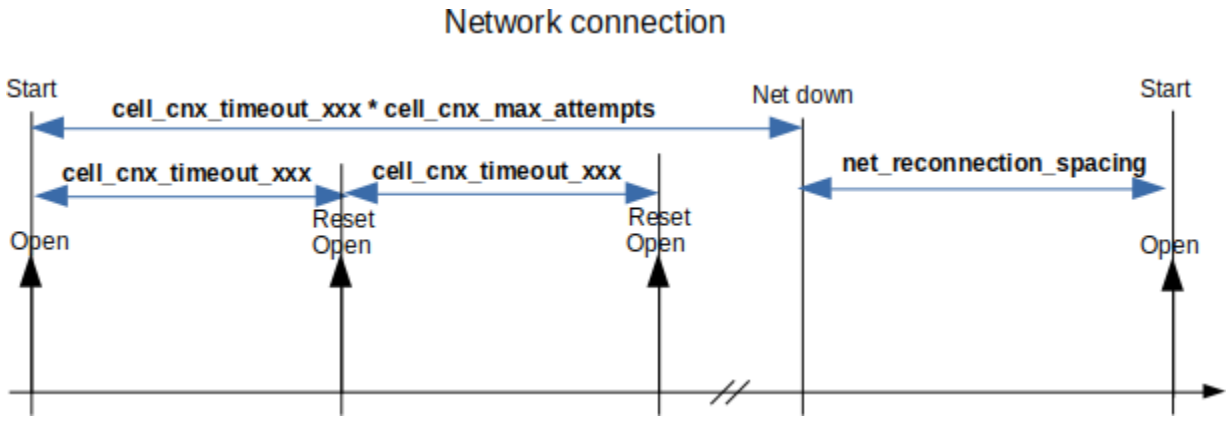


13.5.2 Configuring the cellular network

The cellular network is configured via the cellular parameter group. The relevant networking parameters are:

- **net_selection**: Network selection has the value 1 (cellular only).
- **net_reconnection_spacing**: Duration in seconds between the network down detection and the reconnection.
- **cell_cnx_timeout_static**: Duration in second to connect to the cellular network when the tracker is static.
- **cell_cnx_timeout_motion**: Duration in second to connect to the cellular network when the tracker is in motion.
- **cell_cnx_max_attempts**: Maximum number of attempts to connect to the network before indicating a network lost to the network manager.

The following timing diagram shows the connection processing.



13.5.3 Configuring main and backup networks

In this configuration, the two networks are configured. The setting of the **net_selection** parameter indicates which network play the roles of the main and backup network.The generic behavior is the following:

- The connection manager tries to connect to the main network.
- In case of the main network failure, the connection manager attempts to connect to the backup network.
- If both network fails to connect, the network manager waits for the **net_reconnection_spacing** duration and retries to connect to the main network.
- At the time the connection manager switched to the backup network, the main network probing

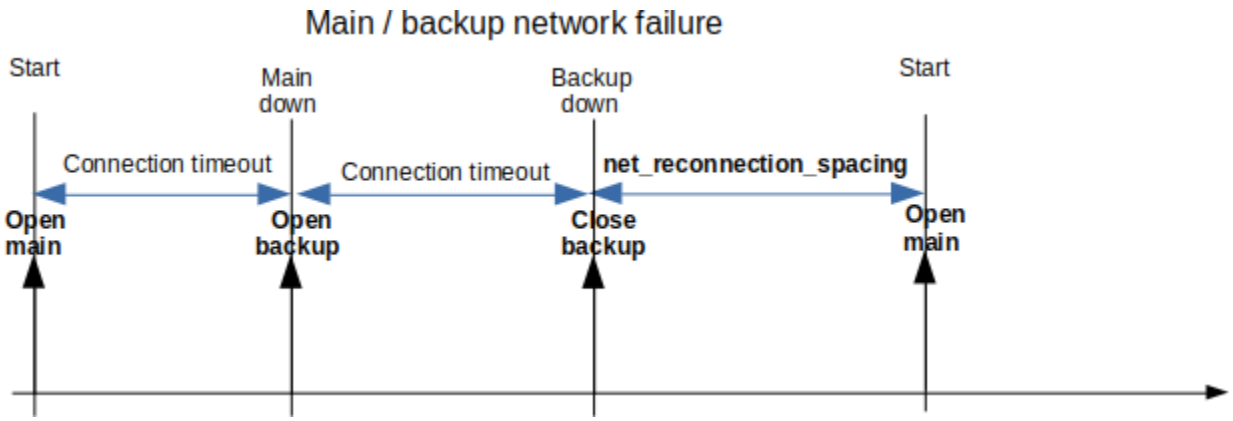
will take place. After each **net_main_probe_timeout,** the network manager suspends the backup network, and attempts to reconnect to the main network.

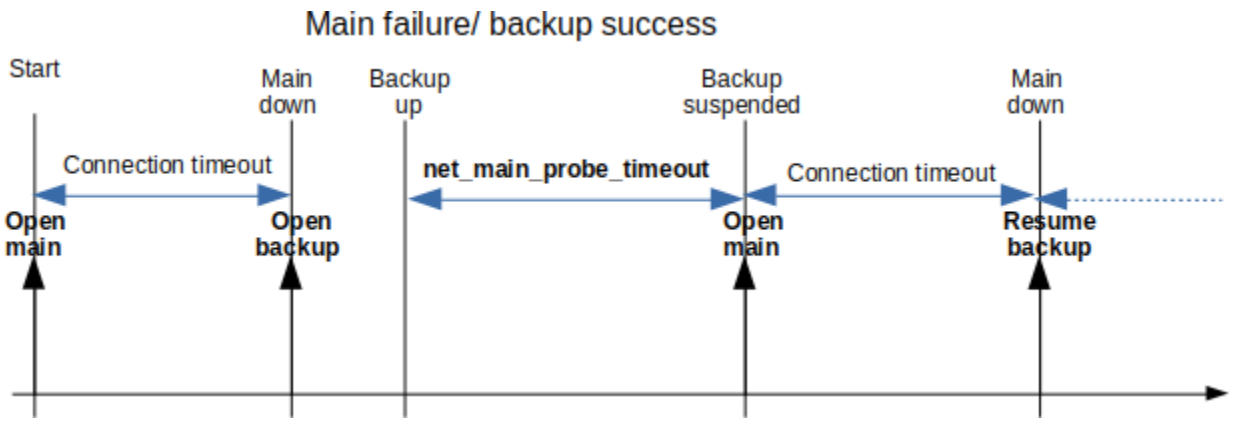
- If the main network connection successes, the main network is kept as the active network.
- Otherwise, the network manager resumes the backup network and restart the probing timer.

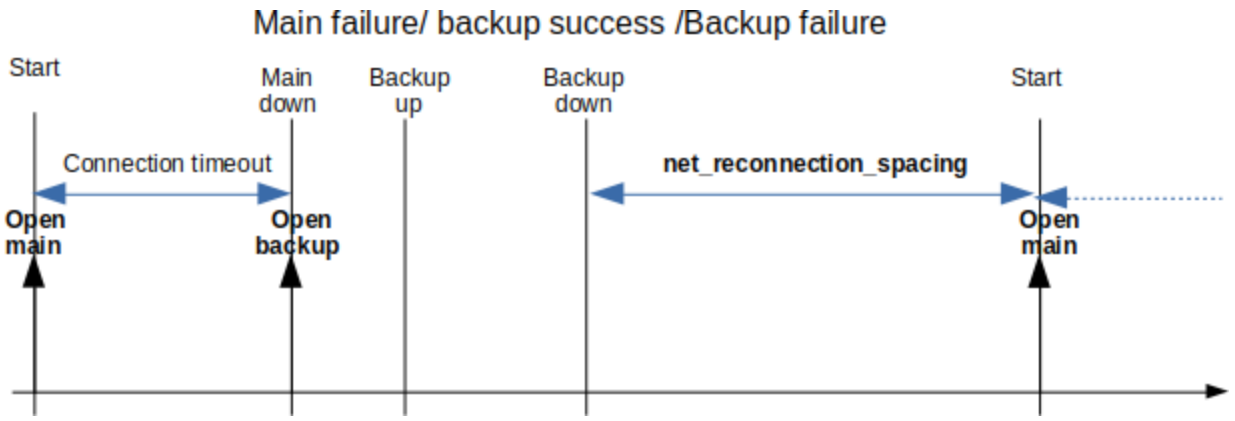
Note that during the main network probing, the connectivity is suspended (The application cannot send payloads).

LoRaWAN and the cellular network should be configured as described in the previous sections. Additional parameters are:

- **net_selection**: The network selection has the value:
  - 2: LoRa is the main network and cellular is the backup one.
  - 3: Cellular is the main network and LoRa is the backup one.
- **net_reconnection_spacing**: Duration in seconds between the network down detection and the reconnection. Used only in the case where the two networks fail to connect.
- **net_main_probe_timeout**: Duration in seconds between each main network probe attempts. This duration is used when the active network is the backup one.






Main failure/ backup success /Backup failure
Start
Main down
Backup up
Backup down
Start
Connection timeout
net_reconnection_spacing
Open main
Open backup
Open main

14 Payload manager

This manager is responsible of creating the uplink messages and distributing the downlinks.

The payload manager works closely with the network manager.

14.1 Uplink processing

The manager is registered against system events that requires a message transmission. It creates the payloads by selecting the appropriate header (single versus multi-frame) and build the data part.

Multi-frame processing

In the case of position messages, the manager may use the multi-frame format. In this case, an extended header is added to the basic header. The extended header contains a group identifier, the last frame bit and the frame identifier.

The group identifier helps the application server to ensure that all frames received belong to the same group. The group identifier is incremented for each new multi-frame payloads. It should be used by the application server to identify which frames belong to a given group. This is particularly useful in case of message lost or unordered reception.

The last frame bit in the extended header indicates whether the group is ended. However, note that in a particular case, this bit may not be set. This case arises only if there are more than the maximum frame in a group and the payload manager stopped the multi-frame processing. In such a case, the application server will rely on the frame identifier to know that the group is ended: the identifier will be the maximum value for the frame identifier.

Note that it is possible that the extended header is used with a single frame. This case may arise for configuration replies, since the data part of the parameters is variable.

To conclude, the application server should process as follow:

- The last bit is set, frame ID less than the max: End of group detected. Group complete.
- The last bit is set, frame ID equals to the max: End of group detected. Group complete.
- The last bit is not set, frame ID equals to the max: End of group detected. Group truncated by the payload manager.
- The last bit is set, frame ID = 0: Single payload sent using the multi-frame format.

Refer to the Application uplink paragraph, Extended header section for the maximum number of groups and frame identifiers.

Network down

When the network is not available (pipe down), the payload manager stores the notifications and the last position message. Only one notification type per class is kept. These notifications are the ones sent via the network and not the system event (refer to the the uplink message section). The notification storage is done via an array of notification classes, for which, each entry consists of the notification type and the date at which it occurred.

Once the network is recovered, the payload manager will create all notifications and the last position message (if any) and send it to the network.

Network hold-on

While connected to the backup network, the network manager probes regularly the main network. In this case the application cannot transmit (since we don't know the actual network to use) and the data pipe is down.

14.2 Position reporting

14.2.1 Overview

The payload manager is responsible of reporting the positions. The position can be either in single frame or multi-frames depending on the geolocation results.

When multiple technology must be reported, each single frame contains only one technology type.

If a scan (BLE, WIFI or LR-AGPS) requires more than one frame to be sent, they are also sent in multi-frame mode. All frames related to a given technology are sent contiguously and they are ordered. In such a case, each frames contains the technology type it refers to.

This means that a complete position report in multi-frames mode can contain multiple technology types and multiple frames related to a single type.

14.2.2 Geolocation mode Always vs fallback

A technology having the always-done mode is always reported even if it fails or it is not solvable.

A technology having the **fallback** mode is reported only if is successful or if there is no other technologies

Note that a position report is always sent. For example if a geolocation consists of a single technology and it fails, the report is sent regardless the mode (Always-done or fallback).

14.2.3 Geolocation not-solvable/failure status

The geolocation may report a status not-solvable for a given technology. This section covers this specific case and indicates the way the position is reported in the case where an acquisition is done with multiple technologies.

The rules are:

- If there is a single technology type, the position is reported with the not-solvable/failure status.
- If there are a multiple technology types, a **fallback** technology with the not-solvable/failure status is reported only if:
  - There is no other technologies to schedule in the geolocation profile.
  - All other technologies are also in fallback mode and they fail or are not solvable. In this case, only the last technology is sent.

Examples:

BLE fallback only. Scan not-solvable .Geolocation profile: {03,00,00,00,00,00}.
The position (single frame) should be BLE with status not-solvable

BLE fallback not-solvable, WIFI fallback success. Geolocation profile: {03,02,00,00,00,00}
The position (single frame) should be WIFI with status success

BLE fallback not-solvable, wifi fallback no-solvable. Geolocation profile: {03,02,00,00,00,00}
The position (single frame) should be WIFI with status not-solvable

BLE Always not-solvable, wifi fallback no-solvable .Geolocation profile: 83,02,00,00,00,00}
The position (single frame) should be BLE with status not-solvable

BLE Always not-solvable, wifi always no-solvable. Geolocation profile: {83,82,00,00,00,00}
The position (multi frames frame) should be BLE with status not-solvable

BLE Always not-solvable, wifi always no-solvable. Geolocation profile: 83,82,00,00,00,00}
The position (multi frames frame) should be BLE with status not-solvable followed by a WIFI position with status not solvable.

BLE fallback not-solvable, wifi always no-solvable. Geolocation profile: {03,82,00,00,00,00}
The position (single frame) should be WIFI with status not-solvable

15 Application uplinks

This section defines the frame format of uplinks sent to the network. The network can be either LoRaWAN or LTE.

Most of the messages are sized to fit the maximum LoRa payload size in the worst case (DR0), which is 51 bytes for EU-868 (repeater compatible). Note that only responses to a query can exceed this limit and restrict the MTU to 63, which may trigger a datarate change if configured in tx_start paramters.

15.1 Cellular network header

On cellular network only, the following message header is added:

Bytes [0- 7]	Bytes [8-9]	
-	b15-0	b7-0
DevEUI	MSB Frame up counter	LSB Frame up counter

Byte 0-1 description

- **DevEUI.** Device unique identifier. Used on cellular networks to discriminate the device which sent the uplink. For convenience, it is identical to the LoRaWAN DevEUI.
- **Frame up counter**. Uplink frame counter. Incremented for each uplink. It wraps to 0 after reaching 65535. It is an emulation of the LoRaWAN uplink frame counter and serves the same purpose: to be able to detect and quantify uplink packet loss when using UDP transport.

15.2 Message header

The uplinks are structured with a common header followed by a data section. The data section varies depending on the type of uplink.

15.2.1 Basic header

The basic header has 4 bytes and is present in both single-frame and multi-frame modes.

Byte 0				Byte 1		Bytes [2-3]	
b7	b6	b5-3	b2-0	b7	b6-0	b15-0	b7-0
M	S	Type	ACK-TK	F	Battery %	MSB timestamp	LSB timestamp

Byte 0 description

- **M**. Multi-frame (0: single frame; 1: multi-frame).
  - 0 - Single frame.
  - 1 – Multi-frame
- **S**. SOS.
  - 0 –SOS mode not active
  - 1 – SOS mode active
- **Type**. Frame type.:
  - 1 – **Notification**. Unsolicited message (System start, Critical/normal temperature, motion start/stop, SOS start/stop, geozoning, battery alert, etc.).
  - 2 – **Position**. Unsolicited message. Geolocation positions.
  - 3 – **Query**. Expects a response from the network (aiding position request, almanac request, etc.).
  - 4 – **Response**: Solicited message. Response to network query (config get/set, POD, Almanac get/set, etc.).
  - Other values: Reserved for future use.
- **ACK-TK**: Ack-token. Value (in [0..7]) extracted from the last downlink received. It is used to acknowledge the corresponding downlink.

Byte 1 description

- **F**. Free for use
- **Battery**: Remaining battery capacity expressed as percentage of full capacity. Special values:
  - 0: **in charge**
  - 127: **unknown**.

Byte 2 – 3 contains the timestamp indicating when the payload was generated. The value represents the number of seconds elapsed since the start of the most recent half-day period (a value of 0 corresponds to either noon or midnight).

15.2.2 Extended header

This header is used when an application message needs to be fragmented to multiple uplinks ("multi-frame" case).

It consists of a basic header (with M = 1) extended with an extra byte called m-frame.

Bytes [0-3]	Byte 4		
Basic header with M=1	Group	m-frame	
	b7-5	b4	b3-0
	Group ID	Last	Fragment number

Byte 4 description

- **Group ID**. Group identifier [0..7]. It is incremented each time a new group of uplinks are sent. It is used to uniquely identify a group. It should be used to avoid mixing uplinks belonging to two different groups.
- **Last**. Set to 1 to indicate the last uplink in the group.
- **Fragment number**. Fragment identifier inside a group [0..15]. It starts at 0 (first uplink) and is incremented for each subsequent uplinks belonging to the same group.

15.3 Notifications

This section describes the notification uplinks.

15.3.1 Overview

The notifications are categorized into classes, with each class containing different types. The class of notification allows a simple mechanism (and configuration) for requesting network acknowledgments: A user may require some classes of notifications to be acknowledged by the network.

The format of a notification is the following

Bytes [0-3]	Byte 4		Bytes [5-variable]
Basic header (**Type =1**)	Notification header		Data
	b7-4	b3-0	
	Class	Type	

Notification header

- Class: Class of notification
- Type: Type of notification

The classes and types are presented below

Class 0. **System**			
Type	**Name**	**Purpose**	**Data**
0	Status	System status	Versions, temperature, reset cause (in case of crash)
1	Low-battery	Low battery detected	Consumption + battery voltage
2	BLE	BLE securely connected	Securely connected/disconnected
3	Tamper	Tamper detection	Casing open/closed

4	Heartbeat	Heart beat message	Temperature, reset cause and CRC

Class 1. **SOS**			
Type	**Name**	**Purpose**	**Data**
0	SOS-on	SOS activated	None
1	SOS-off	SOS deactivated	None

Class 2. **Temperature**			
Type	**Name**	**Purpose**	**Data**
0	Temp_high	Critically high temperature reached	Temperature
1	Temp_low	Critically low temperature reached	Temperature
2	Temp_normal	Temperature back to normal	Temperature

Class 3. **Accelerometer**			
Type	**Name**	**Purpose**	**Data**
0	Motion_start	Motion start detected	None
1	Motion_end	Motion end detected	Acceleration vector + motion percent
2	Shock	Shock detected	Acceleration vector + GADD index + nb shock

Class 4. **Network**			
Type	**Name**	**Purpose**	**Data**
0	Main_up	Main network is up.	Network data
1	Backup_up	Main network down. Backup is up.	Network data

Class 5. **Geozoning**			
Type	**Name**	**Purpose**	**Data**
0	Entry	Entry beacon detected	Beacon ID or MAC address
1	Exit	Exit beacon detected	Beacon ID or MAC address
2	In_hazard	Entry in a hazardous area detected	Beacon ID or MAC address
3	Out_hazard	Exit from a hazardous area detected	Beacon ID or MAC address
4	Meeting_point	Meeting point detected	Beacon ID or MAC address

15.3.2 Class system

Status notification

The status is transmitted with a common section and a page section. Pages are used to publish information that does not require frequent updates, using a round robin method. Each time a status notification is sent, the page number increments until the last page is reached. Once the last page is sent, the page identifier resets to 0.

Common part

1 byte	1 byte	
Current temperature	Reset cause and page ID	
b7- 0	b7-3	b2-0
[-126 .. +127]	Reset cause	Page ID

Note

- **Current temperature**. Current temperature. Signed integer (2's complement) expressed in degree Celsius.
- **Reset cause**:
  - AOS_ERROR_NONE (0) : No error.
  - AOS_ERROR_HW_NMI (1): Hardware non-maskable interrupt fault.
  - AOS_ERROR_HW_FAULT (2): Hardware fault.
  - AOS_ERROR_HW_MPU (3): Hardware MPU fault.
  - AOS_ERROR_HW_BUS (4): Hardware bus fault.
  - AOS_ERROR_HW_USAGE (5): Hardware usage fault.
  - AOS_ERROR_HW_IRQ (6): Unexpected interruption.
  - AOS_ERROR_HW_WDOG (7): Watchdog
  - AOS_ERROR_HW_BOR(8): Brown out reset
  - AOS_ERROR_SW_ST_HAL_ERROR(9): ST HAL error.
  - AOS_ERROR_SW_FREERTOS_ASSERT (10): FreeRTOS assertion.
  - AOS_ERROR_SW_FREERTOS_TASK_OVF (11):FreeRTOS Task stack overflow.
  - AOS_ERROR_SW_BLE_ASSERT(12). BLE core assertion
  - AOS_ERROR_SW_RTC_FAIL (13): Real Time Clock peripheral fails to start.
  - AOS_ERROR_SW_LORA_FAIL (14): LoRa unrecoverable failure
  - AOS_ERROR_SW_DEBUG (15): Used to debug
  - AOS_ERROR_SW_APP_START (16): Application assertions.
- Page Id
  - Max page ID = 1 for LoRaWAN only trackers (2 pages).
  - Max page ID= 3 for Cellular Combo tracker (4 pages).

Page 0. General status

3 bytes			4 bytes				3 bytes			2 bytes	
AT3 version			Configuration version				LR-HW version			HW batch ID	
b23-16	b15-8	b7-0	b31-24	b23-16	b15-8	b7-0	b23-16	b15-8	b7-0	b15-8	b7-0
M	m	i	M	m	i	u	HW	type	FW	MSB	LSB

2 bytes		1 byte	1 byte	1 byte	2 bytes		2 bytes	
HW BOM ID		Max temperature	Min temperature	Motion %	Battery voltage (mV)		Total consumption (mAh)	
b15-8	b7-0	b7-0	b7-0	b7-0	b15-8	b7-0	b15-8	b7-0
MSB	LSB	[-126 .. +127]	[-126 .. +127]	[0..100]	MSB	LSB	MSB	LSB

2 bytes		2 bytes		2 bytes		2 bytes		2 bytes		2 bytes	
Cellular consumption (mAh)		GNSS consumption (mAh)		WIFI consumption (mAh)		LR GNSS consumption (mAh)		BLE consumption (mAh)		MCU consumption (mAh)	
b15-8	b7-0	b15-8	b7-0	b15-8	b7-0	b15-8	b7-0	b15-8	b7-0	b15-8	b7-0
MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB

4 bytes			
Config CRC			
b31-24	b23-16	b15-b8	b7-0
MSB			LSB

Notes

- M: AT3 major version (for firmware version and configuration version)
- **m**: AT3 minor version (for firmware version and configuration version)
- **i** : AT3 iteration (for firmware version and configuration version)
- **u**: AT3 configuration user byte
- **HW:** Hardware version
- **Type:** Type of hardware
- **FW:** Firmware version
- **Max temperature**. Max temperature since the beginning of the product life. Signed integer (2's complement) expressed in degree Celsius.

- **Min temperature.** Min temperature since the beginning of the product life. Signed integer (2's complement) expressed in degree Celsius.
- **Battery voltage:** Current battery voltage **(**mV)
- **Consumption fields**. All consumption fields are in mAh
- **Config CRC**. CRC of the configuration (all groups excepts internal).

Page 1. Almanac status

2 bytes		4 bytes	1 byte	2 bytes		5 byte	1 byte
LR1110 GPS date		LR1110 GPS outdated	LR1110 GPS good	LR1110 BEIDOU date		LR1110 BEIDOU outdated	LR1110 BEIDOU good
b15-8	b7-0	b31-0	b7-0	b15-8	b7-0	b39-0	b7-0
MSB	LSB	Bit field	[0..31]	MSB	LSB	Bit field	[0-37]

2 bytes		4 bytes	1 byte	2 bytes		5 byte	1 byte
GNSS GPS date		GNSS GPS outdated	GNSS GPS good	GNSS BEIDOU date		GNSS BEIDOU outdated	GNSS BEIDOU good
b15-8	b7-0	b31-0	b7-0	b15-8	b7-0	b39-0	b7-0
MSB	LSB	Bit field	[0..31]	MSB	LSB	Bit field	[0-37]

Notes

- **LR1110/GNSS GPS almanac date**. Highest date (GPS week number) over all GPS almanac entries.
- **LR1110/GNSS GPS outdated**. Bitmap of outdated entries in the GPS almanac.
- **LR1110/GNSS GPS good**. Number of entries matching the highest date.
- **LR1110/GNSS BEIDOU almanac date**. Highest date (GPS week number) over all BEIDOU almanac entries.
- **LR1110/GNSS BEIDOU outdated**. Bitmap of outdated entries in the BEIDOU almanac.
- **LR1110/GNSS BEIDOU good**. Number of entries having the highest date.

Each bit i of the bitmap represents satellite i (1 << i).

Page 2. Cellular part I

6 bytes					21 bytes	16 bytes
FW version					ICCID	IMSI
b47-40	b39-32	b31-24	b23-16	b15-0	Ascii	Ascii
branch	mode	image	delivery	release		

Note that the ICCID and IMSI are available only if a cellular connection has been established. Otherwise, these values are set to 0.

Page 2. Cellular part II

33 bytes	16 bytes
EUICCID	IMEISV
Ascii	Ascii

Notes

- The EUICCID is available only if a cellular connection using the eSIM has been established. Otherwise, this value is set to 0 (33 null bytes).
- The IMEISV is not yet supported. It is set to a null value (16 NULL bytes).

Low battery notification

2 bytes		2 bytes	
Consumption		Battery voltage	
b15-8	b7-0	b15-8	b7-0
MSB	LSB	MSB	LSB

Notes

- Consumption is in mAh
- Battery voltage is in mV

BLE notification

1 bytes	
State	
b7-1	b0

RFU	state

Notes
- State: 0: Disconnected, 1: Connected
- RFU: Reserved for future use

Tamper notification

1 bytes	
State	
b7-1	b0
RFU	state

Notes
- State: 0: Casing closed, 1: Casing open
- RFU: Reserved for future use

15.3.3 Class SOS
No data. The SOS status is in the basic header.

15.3.4 Class temperature
The 3 types of notification are : Critically high temperature / Critically low temperature / Return to normal temperature range
Each of these notifications includes the temperature which triggered the notification.

1 byte
Temperature
b7-0
[-126 .. +127]

The temperature is signed (2's complement coding) and is in degrees Celsius.

15.3.5 Class accelerometer

The data part of this notification class depends on the type.

Motion start

No data

Motion end

2 bytes		2 bytes		2 bytes		1 byte
Acceleration X		Acceleration Y		Acceleration Z		Motion %
b15-8	b7-0	b15-8	b7-0	b15-8	b7-0	b7-0
MSB	LSB	MSB	LSB	MSB	LSB	[0..100]

The 3 axis measurement of gravity after the device has stopped allows to infer the angular position of the device. They are expressed as mg signed values (2's complement coding)

Shock

2 bytes		2 bytes		2 bytes		1 byte	1 byte
Acceleration X		Acceleration Y		Acceleration Z		GADD index	Nb shocks
b15-8	b7-0	b15-8	b7-0	b15-8	b7-0	b7-0	b7-0
MSB	LSB	MSB	LSB	MSB	LSB	[0..100]	[1-255]

The 3 axis measurement of maximum shock accelerations are expressed in mg signed values (2's complement coding), allowing to infer the direction of the shock.

Body damage is more accurately represented by the Gadd index, which integrates acceleration intensity over total shock duration.

15.3.6 Class network

The data part of this class of notifications does not depend on the type and is the following

1 byte	1 byte	1 byte
Active network	Main network	backup network

b7-0	b7-0	b7-0
[0..2]	[0..2]	[0..2]

Each byte encodes a network technology:

- 0: No network
- 1: LoRaWAN network
- 2: Cellular network in low power mode
- 3: Cellular network in high power mode

15.3.7 Class geozoning

TO BE DONE

15.4 Positions

The position messages can be sent in either single-frame mode or multi-frames mode.

15.4.1 Position header

The basic position header is defined as follows:

Single frame mode

Bytes [0-3]	Byte 4	Bytes [7-variable]
Basic header (**Type = 2**)	Position header	Data

Multi-frame mode

Bytes [0-4]	Byte 5	Bytes [8-variable]
Extended header (**Type = 2**)	Position header	Data

The position header is defined as follows:

Position header 4 bytes					
Information			Extended info		Triggers (2 bytes)
b7	b6-5	b4-0	b7-4	b3-0	b15-b0
M	Status	Position type	RFU	M-cnt	Trigger bit-map

Position header

Information

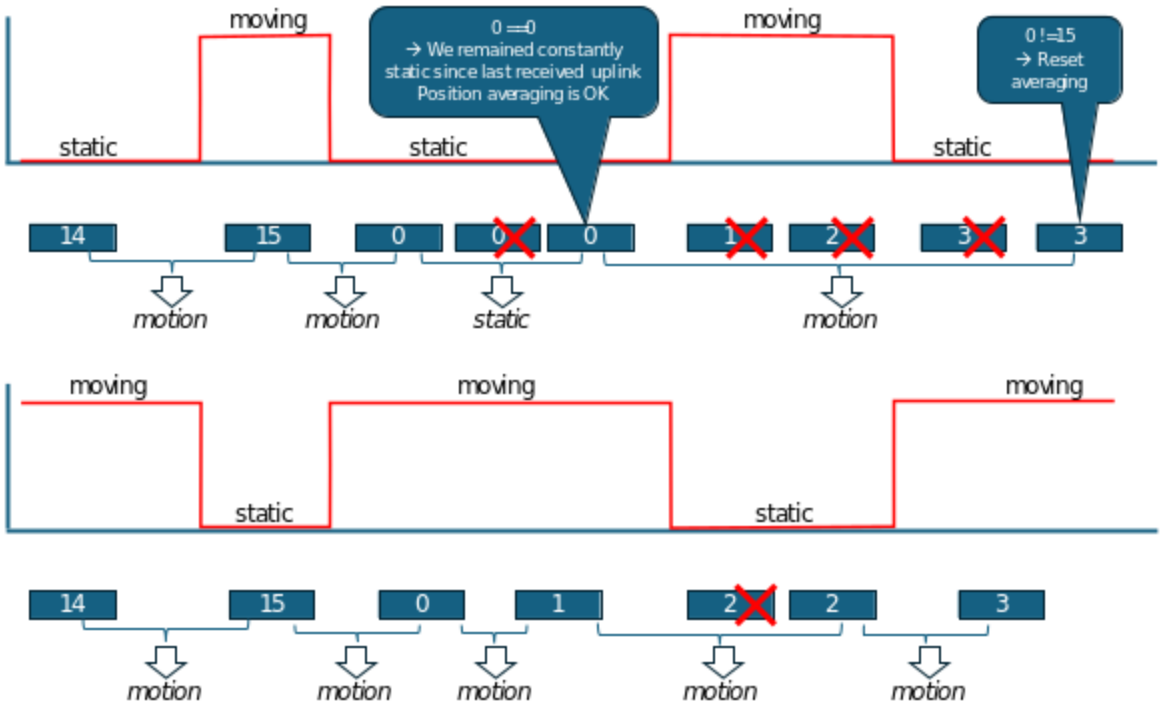
- b7: **M**. Motion bit. Set if a motion event has been detected since the transmission of the previous position payload.
- b6-5: **Status**. Status of the position
  - 0 – Success. Data contains the position.
  - 1 – Timeout. Max time allowed to acquire the position. Optional data.
  - 2 – Failure. Position acquisition failed for other reasons. Data contains failure reason or no data.
  - 3 – Not-solvable: The acquisition has been done but is not solvable (WIFI or BLE). Optional data.
- b4-0: **Position type**
  - 0 – LR1110 GNSS formatted Nav1
  - 1 – LR1110 GNSS Semtech Nav1
  - 2 – LR1110 GNSS Semtech Nav 2

- 3 – WIFI
- 4 - BLE scan1 (report mode: MAC-address)
- 5 – BLE scan1 (report mode: Short IDs)
- 6 – BLE scan1 (report mode Long IDs)
- 7 - BLE scan2 (report mode: MAC-address)
- 8 – BLE scan2 (report mode: Short IDs)
- 9 – BLE scan2 (report mode Long IDs)
- 10 - MT3333 GNSS fix
- 11 – MT3333 LP-GNSS

Extended info

- b7-4: RFU (Reserved For Future Use)
- b3-0: M-cnt. 4-bit motion counter. Increments if a motion event has been detected since the transmission of the previous position payload. The counter rolls over once the max value is reached.

This 4-bit motion counter allows the geolocation back-end application to detect whether there has been motion between 2 successive geolocation uplinks by comparing the value of the motion counter with its previously received value. This works even if we have lost up to 14 consecutive in-motion uplinks, and any number of static uplinks. This enables the static position averaging filters to reset properly if the location of the tracker has changed.



In order to work properly, a reliable deduplication algorithm should be implemented in order to avoid comparing the counter value with the same value from a packet transmitted multiple times (custom retransmit policy). In AT3, this can be based on the value of the basic header 2-byte timestamp field.

Trigger

Bit-field providing the geolocation triggers that were active for this position. Refer to the geolocation manager section for the bit meaning.

15.4.2 LR1110 GNSS Formatted Nav1

This position message can carry up to 10 satellite information. The multi-frame format can be used. In this case each frame carries a single scan.

2 bytes	4 bytes					4 bytes
Time	Sat 1 info					Sat 2 info
-	b31-30	b29-24	b23-22	b21-19	b18-0	Same as sat 1
-	C	ID	CN	000	PRN	Same

Time : LR1110 SW Time in step of 16 seconds.

Sat x info

- b31-30: **C**. Constellation:
  - 0 – GPS
  - 1 – BEIDOU
  - 2, 3 - Spare
- b29-24: **ID**. Satellite identifier starting at 0 regardless the constellation type.
- b23-22: **CN**. Carrier to noise information
  - 0 – C/N greater than 45 dB
  - 1 – C/N in range [41.. 45] dB
  - 2 - C/N in range [37..41] dB
  - 3 - C/N less than 37 dB
- b21-19: Unused. Set to 0
- b18-0: **PRN**. Pseudo range value

To fit the LoRaWAN maximum payload size in the worst condition, the number of satellites info should be limited to 10.

15.4.3 LR1110 GNSS Semtech Nav1

The data part is the actual data payload received from the LR1110. It is encoded and only Semtech (or licensed users like Abeeway) can decode it. It should be forwarded as-is to the Semtech cloud.

Note that the number of sniffed satellites should be set appropriately to allow LoRaWAN transmission with low data rate (DR0 – DR2 for Europe).

The multi-frame format can be used. In this case each frame carries a single scan.

15.4.4 LR1110 GNSS Semtech Nav2

The data part is the actual data payload received from the LR1110. It is encoded and only Semtech (or licensed users like Abeeway) can decode it. It should be forwarded as-is to the Semtech cloud.

The multi-frame format can be used. In this case each frame carries a single scan.

15.4.5 WIFI

This uplink message can carry up to 6 BSSIDs per frame. To carry more BSSIDs, the multi-frame will be used if the payload size requires it.

Payload format

7 bytes							7 bytes
BSSID 1 info							BSSID 2 info
MAC address (6 bytes)						1 byte	Same as BSSID 1
MSB					LSB	RSSI	Same

Note

The RSSI is a negative value (2's complement) carrying the Received Signal Strength Information (in dB).

To fit the LoRa maximum payload size in the worst condition, the number of BSSI per frame is limited to 6. Above this value, the multi-frame is used.

15.4.6 BLE scan. MAC address

This uplink message can carry up to 6 beacon information per frame. To carry more beacons, the multi-frame can be used.

Payload format

7 bytes							7 bytes
Beacon 1 info							Beacon 2 info
Identifier (6 bytes)						1 byte	Same as Beacon 1
MSB					LSB	RSSI	Same

Notes

- The identifier is the MAC address if the report type is MAC address, otherwise it is 6 bytes extracted as the beacon identifier.
- The RSSI is a negative value (2's complement) carrying the Received Signal Strength Information (in dB).
- To fit the LoRaWAN maximum payload size in the worst condition, the number of Beacon info per frame is limited to 6. Above this value, the multi-frame is used.

15.4.7 BLE scan. Short ID

This uplink message can carry up to 14 beacon information fields per frame. To carry more beacons, the multi-frame can be used.

Payload format

3 bytes			3 bytes	3 bytes
Beacon 1 info			Beacon 2 info	Beacon 3 info
Beacon ID (2 bytes)		1 byte	Same as beacon 1	Same as beacon 1
MSB	LSB	RSSI	Same as beacon 1	Same as beacon 1

Note

The RSSI is a negative value (2's complement) carrying the Received Signal Strength Information (in dB).

To fit the LoRaWAN maximum payload size in the worst conditions, the number of Beacon info per frame is limited to 14. Above this value, the multi-frame is used.

15.4.8 BLE scan. Long ID

This uplink message can carry up to 2 beacon information fields per frame. To carry more beacons, the multi-frame format can be used.

Payload format

17 bytes							17 bytes
Beacon 1 info							Beacon 2 info
Beacon ID (16 bytes)						1 byte	Same as Beacon 1
MSB					LSB	RSSI	Same

Note

The RSSI is a negative value (2's complement) carrying the Received Signal Strength Information (in dB).

To fit the LoRaWAN maximum payload size in the worst condition, the number of Beacon info per frame is limited to 2. Above this value, the multi-frame is used.

15.4.9 MT3333 GNSS fix

This uplink message carries a GNSS fix or the GNSS satellites tracking if the fix failed.

Payload format in case of success

4 bytes				4 bytes				2 bytes		2 bytes		2 bytes		1 byte	1 byte
Latitude				Longitude				Altitude		COG		SOG		EHPE	Quality
MSB			LSB	MSB			LSB	MSB	LSB	MSB	LSB	MSB	LSB	-	-

Notes

- The latitude the longitude are signed value (2's complement) and expressed with a unit of 10^{-7} degree.
- The altitude is signed and expressed in meters.
- COG: Course Over Ground expressed in 1/100 of degrees
- SOG: Speed Over Ground expressed cm/second
- EHPE is encoded as follow:

Coded value	EHPE in meters
0 – 250	0 – 250 meters
251	250 < EHPE < 500

252	500 < EHPE < 1000
253	1000 < EHPE < 2000
254	2000 < EHPE < 4000
255	EHPE > 4000

- Quality: Fix quality coded as follow:
  - Bits b7-5: Quality:
    - 0 – invalid: The GNSS gave a fix but consider it as invalid
    - 1 – valid: The GNSS gave a fix but has no idea if it is a 2D or 3D
    - 2 – fix 2D: Fix valid for 2 dimension (latitude and longitude valid).
    - 3 – fix 3D Fix valid for 3 dimension (latitude, longitude and altitude valid).
  - Bits b4-0: Number of satellites used for the fix. Max 12

Payload format in case of failure or timeout

1 byte	2 bytes		2 bytes		...
Status	Sat 0		Sat 1		...
-	svId	Info	svId	Info	...

Notes

- The Status field is coded as follow:
  - b7 – b5: Cause
    - 0: T0 timeout
    - 1: T1 timeout
    - 2: acquisition timeout
  - b4 – b0: Number of satellites seen.
- The info field is coded on 1 byte and is formatted as follow:
  - b7-b6: Constellation
    - 0: GPS
    - 1: GLONASS
    - 2: BEIDOU
    - 3: GALILEO
  - b5-0: C/N0 in dBm

15.4.10 MT3333 LP-GNSS

This position message can carry up to 9 satellites information fields. The multi-frame format can be used. In this case each frame carries a single scan.

4 bytes	4 bytes				4 bytes
Time	Sat 1 info				Sat 2 info
-	b31-30	b29-24	b23-22	b21-0	Same as sat 1
-	C	ID	CN	PRN	Same

Time : MT333 SW Time (TOW) in microseconds. It is sent MSB first (big endian).

Bits	31 - 20	19 - 0
Description	Nb seconds in the hour	Nb microseconds

Sat x info. Coded the same way than LR1110 NAV1 (with bit 21-19 used).

- b31-30: **C**. Constellation:
  - 0 – GPS
  - 1 – BEIDOU
  - 2, 3 - Spare
- b29-24: **ID**. Satellite identifier starting at 0 regardless the constellation type.
- b23-22: **CN**. Carrier to noise information
  - 3 – C/N greater than 34 dB
  - 2 – C/N in range [28.. 33] dB
  - 1 - C/N in range [22..27] dB
  - 0 - C/N less than 21 dB
- b21-0: PRN. Pseudo range value. Coded for a full scale of 1ms shifted by 2 to the right.

15.5 Query

These messages expect a response. They are always sent in single-frame mode.

The queries are formatted as follow.

Bytes [0-3]	Byte 4	Bytes [5-variable]
Basic header (**Type = 3**)	Query header	Data

Query header

- b7-b5: Spare bits
- b4-0: **Query type**:
  - 0 – Aiding-position. The aiding position is needed by the tracker. **No data**.
  - 1 – Echo-request. The tracker is probing the network (cellular only).
  - 2 – Update GPS almanac. GPS almanac entries need to be refreshed. Data contains the list of satellites for which the update is needed.
  - 3 – Update BEIDOU almanac. BEIDOU almanac entries need to be refreshed. Data contains the list of satellites for which the update is needed.

Note

The tracker system time update could be done using the appropriate LoRa MAC request. However, remind that the application messages should not be based on the underlying transport protocol.

15.5.1 Echo-request

This query is used to probe the cellular network. AT3 does not send it for LoRa (since it uses the LoRa link-check request for this purpose).

Data part
1 byte
Sequence number

Note

- The tracker increases the sequence number for each transmission. The answer should contain the same value than the transmitted one.

15.5.2 Update GPS almanac

Several satellites may be requested at a time, The data part is a list of satellite identifiers defined as follow:

1 Byte	1 Byte	...
SV ID1 1 [0..31]	SV ID 2 [0..31]	...

Notes

- **SV ID** means Space Vehicle identifier. First satellite identifier is 0.
- Due to the limited payload size on the LoRa network (EU868, DR0-2, Max: 59 bytes), only 3 entries can be updated in a downlink. So the number of almanac entries requested in a single request is limited to 3.

15.5.3 Update BEIDOU almanac

Several satellites may be requested at a time, The data part is a list of satellite identifiers defined as follow:

1 Byte	1 Byte	...
SV ID1 [0..34]	SV ID 2 [0..34]	...

Note

- **SV ID** means Space Vehicle identifier. First satellite identifier is 0.
- Due to the limited payload size on the LoRa network (EU868, DR0-2, Max: 59 bytes), only 3 entries can be updated in a downlink. So the number of almanac entries requested in a single request is limited to 3.

15.6 Response

These messages carry the responses to network requests. Multi-frames format can be used.

Single frame mode

Bytes [0-3]	Byte 4	Bytes [5-variable]
Basic header (**Type = 4**)	Response header	Data

Multi-frame mode

Bytes [0-4]	Byte 5	Bytes [6-variable]
Extended header (**Type = 4**)	Response header	Data

Response header

- b7-b5: Spare bits
- b4-0: **Response type**
  - 0 – Answer of a generic configuration set request.
  - 1 – Answer of a parameter class configuration set request.
  - 2 – Answer of a generic configuration get request.
  - 3 – Answer of a parameter class configuration get request.
  - 4 – Answer of a BLE connectivity status request.
  - 5 – Answer of a configuration CRC request
  - 6 – Answer of a sensor get request

15.6.1 Response of a generic configuration set request.

The data part consists of up to 16 parameters can be acknowledged at a time.

4 Bytes	3 Bytes			3 Bytes			...
CRC	C-ID	L-ID	Status	C-ID	L-ID	Status	...

Notes

- **CRC.** Global configuration CRC (big endian)
- **C-ID**. Parameter class identifier
- **L-ID:** Local parameter identifier
- **Status**. Local status for the parameter.

- 0 – Success
- 1 – Not found
- 2 – Below lower bound. For string and byte-array, length below the minimum
- 3 – Above higher bound. For string and byte-array, length above the maximum
- 4 – Bad value. Value in acceptable range but not is not supported.
- 5 - Type mismatch
- 6 – Operation error (operation failure)
- 7 – Read-only variable.

15.6.2 Response to a parameter class configuration set request

The data part consists of up to 24 parameter status fields.

1 byte	3 bytes	2 Bytes		2 Bytes		...
C-ID	CRC	L-ID	Status	L-ID	Status	...

Notes

- **C-ID**. Parameter class identifier
- **CRC**. Three lowest bytes of the configuration group CRC
- **L-ID**: Local parameter identifier
- **Status**. Local status for the parameter.
  - 0 – Success
  - 1 – Not found
  - 2 – Below lower bound. For string and byte-array, length below the minimum
  - 3 – Above higher bound. For string and byte-array, length above the maximum
  - 4 – Bad value. Value in acceptable range but not is not supported.
  - 5 - Type mismatch
  - 6 – Operation error (operation failure)
  - 7 – Read-only variable.

15.6.3 Response to a generic configuration get request

The data part consists of a list of parameters belonging to any classes. The multi-frame format can be used. Note that for some parameters, the size is variable. So it is advised to avoid requesting too many parameters at a time.

Parameter entry description

Parameter entry				...
3 Bytes			Variable	...
C-ID	L-ID	S/T	Data	...

Notes

- **C-ID**. Parameter class identifier
- **L-ID:** Local parameter identifier
- **S/T:** Parameter size and type
  - Bit 7-3: Variable size
  - Bit 2-0: type
    - 0 – Deprecated.
    - 1 – Integer 32 bits
    - 2 – Floating point (4 bytes)
    - 3 – ASCII string
    - 4 – Byte array
    - 5 - Error
- **Data**: Variable data part
  - Deprecated. No data.
  - Integer 32 bits. 4 bytes in big endian (MSB first)
  - Floating point. Single precision, 4 bytes n big endian (MSB first)
  - ASCII string. Max 31 characters. It does not include the NULL char.
  - Byte array. Max 32 bytes. For 32 bytes, the parameter size will be 0.

15.6.4 Response to a parameter class configuration get request

The data part consists of a list of parameters belonging to any classes. The multi-frame format can be used. Note that for some parameters, the size is variable. So it is advised to avoid requesting too many parameters at a time.

The data part starts with the parameter class identifier and is followed by a list of parameter entries.

Common	Parameter entry 1			Parameter entry 2			...
1 Byte	2 Bytes		Variable	2 Bytes		Variable	...
C-ID	L-ID	S/T	Data	L-ID	S/T	Data	...

Notes

- **C-ID**. Parameter class identifier
- **L-ID:** Local parameter identifier
- **S/T** Parameter size and type.

- Bit 7-3: Parameter size
- Bit 2-0: type
  - 0 – Deprecated.
  - 1 – Integer 32 bits
  - 2 – Floating point (4 bytes)
  - 3 – ASCII string
  - 4 – Byte array
  - 5 - Error

➢ **Data**: Variable data part
- Deprecated. No data.
- Integer 32 bits. 4 bytes in big endian (MSB first)
- Floating point. Single precision, 4 bytes n big endian (MSB first)
- ASCII string. Max 31 characters. It does not include the NULL chan.
- Byte array. Max 32 bytes. For 32 bytes, the parameter size will be 0.

15.6.5 Response of a BLE connectivity status request

The data part consists of the BLE status

Status
1 Byte

The **status** can take one of the following value:
- 0 – Idle: No connectivity activity.
- 1 – Advertising: The device is advertising for BLE connectivity
- 2 – Connected: The device is connected but not bonded
- 3 – Bonded: The device is connected and bonded.

15.6.6 Response of a configuration CRC request

The data part consists of a list of configuration CRC groups matching the request.

No groups defined in the request (bitmap null)

The data part consists of the requested bitmap and global configuration CRC (all groups except the internal)

2 bytes	4 Bytes

Bitmap	CRC

The **CRC** has the following form

CRC			
4 Bytes			
b31-24	b23-16	b15-8	b7-0
MSB			LSB

The **bitmap** has the same format than the request. It is formed as follow: Sum of $2^{group\_identifier}$.
Example: The groups 1, 3, 7 are requested. The bitmap will be $2^1 + 2^3 + 2^7 = 138 = 0x8A$

Groups defined in the request

The data part consists of the requested bitmap, group configuration CRC on 3 bytes. The CRC calculation is the same than the global one except that only the 3 lowest bytes are sent.

2 bytes	3 Bytes	3 Bytes	...
Bitmap	CRC1	CRC2	...

The **CRC** has the following form

CRC		
3 Bytes		
b23-16	b15-8	b7-0
MSB		LSB

The **bitmap** has the same format than the request. It is formed as follow: Sum of $2^{group\_identifier}$.
Example: The groups 1, 3, 7 are requested. The bitmap will be $2^1 + 2^3 + 2^7 = 138 = 0x8A$

15.6.7 Response of a sensor get request

The data part is the list of the sensor values requested.

Sensor ID 1	Value for ID 1	Sensor ID 2	Value for ID 1	...
1 byte	variable	1 byte	variable	...

Where

- Sensor ID x: Sensor identifier:
  - 0: Do not use
  - 1: Accelerometer
- Value for ID x: Output value of the sensor. The size, format and the unit depends on the sensor type.

Accelerometer sensor

Acceleration vector		
X (mg)	Y (mg)	Z (mg)
2 bytes	2 bytes	2 bytes

Where

- X: Signed integer value (in mg) of the acceleration belonging to the X axis
- Y: Signed integer value (in mg) of the acceleration belonging to the Y axis
- Z: Signed integer value (in mg) of the acceleration belonging to the Z axis

Note

Values are in big endian.

16 Application downlinks

This section defines the format of downlinks sent by the network. The network can be either LoRaWAN or LTE.

16.1 Message header

The downlinks are composed by a common header followed by a data part. The data part content depends on the downlink type.

16.1.1 Basic header

The basic header has a single byte. The multi-frame mode is not supported for downlink messages (since an uplink is needed to trigger the downlink message in LoRaWAN class A).

Basic header

Byte 0		
b7-6	b5-3	b2-0
RFU	Type	ACK-TK

Byte 0 description

- **RFU**. Reserved for Future Use.
- **Type**. Frame type:
  - 1 – **Command**. Commands are neither acknowledged nor answered (Reset, BLE bootloader, SOS enter/leave, and so on).
  - 2 – **Request**. Requests expect a response uplink.
  - 3 – **Answer**. Answer a query from the tracker.
- **ACK-TK**: Ack-token. Value (in [0..7]) extracted from the last downlink received. It is used to acknowledge the downlinks.

16.2 Commands

The commands do not expect either responses or statuses from the tracker. The proper receipt and execution of a command can be inferred by analyzing the ACK-TK field of uplinks received after sending the command.

The format of a command is the following

Byte 0	Byte 1	Variable
Basic header (**Type =1**)	Command	Data

Note

- Byte 0. Basic header with type = 1.
- Byte 1. Command:
  - 0 – Clear the config in flash and reset the tracker
  - 1 – Reset the tracker (no data)
  - 2 – Start SOS (no data).
  - 3 – Stop SOS (no data).
  - 4 – System status request (no data).
  - 5 – Position-On-Demand (POD) (no data)
  - 6 – Set GPS almanac
  - 7 – Set BEIDOU almanac
  - 8 – Start BLE connectivity (no data)
  - 9 – Stop BLE connectivity (no data)
  - 10 – System event
  - 11 – Clear motion percentage (no data)

16.2.1 Clear config in flash

This command may contain additional data.

If no data are provided, the configuration will be set to the default factory values (except the internal group).

If data are provided, it represents the list of parameter full identifiers that we want to keep their current values (and so, not reset to the factory default). Up to 10 parameters can be preserved.

2 bytes		2 bytes		...	2 bytes	
Full ID 1		Full ID 2		...	Full ID 10	
1 byte	1 byte	1 byte	1 byte	...	1 byte	1 byte
Class ID	Local ID	Class ID	Local ID	...	Class ID	Local ID

16.2.2 Set GPS almanac

This command is used to setup GPS almanac entries. The format of the data part is the following

Common	Almanac entry 1	Almanac entry 2	...
2 bytes	15 bytes	15 bytes	...
Date	Data-entry 1	Data-entry 2	...

Notes

- **Date**: Number of days since April 7th 2019. Big endian.
- **Data-entry** is formatted as follow:

Almanac Entry	
1 byte	14 bytes
SV ID	Almanac data

- **SV-ID**: Satellite identifier in the constellation. Starts at index 0.
- **Almanac data**: Normally the almanac entry contains 15 bytes. It has been observed that the last byte of the entry is always 0xFF (MSB of Omega dot). So sending 14 bytes for an entry is enough and AT3 will add the missing byte upon reception.
- It's up to the server to adapt the number of entries to send based on the network type (LoRaWAN or Cellular) and the data-rate used for the downlink if LoRaWAN is used. **However, the maximum size of 255 (full downlink size including network encapsulation) must be respected.**

16.2.3 Set BEIDOU almanac entry request

TO BE DONE.

16.2.4 System event command

This command offer the possibility to trigger events, the possible events are related to user class and core class.

The format of the data part is the following:

Class id	Event type
1 byte	1 byte

16.3 Requests

Downlink requests expect a response from the tracker.
The format of a request is the following

Byte 0	Byte 1	Bytes [2-variable]
Basic header (**Type =2**)	Request	Data

Note

- **Byte 0.** Basic header with type = 2.
- **Byte 1**. Request type:
  - 0 – Generic configuration set request.
  - 1 – Parameter class configuration set request.
  - 2 – Generic configuration get request.
  - 3 – Parameter class configuration get request.
  - 4 – BLE connectivity status request.
  - 5 – CRC configuration request
  - 6 – Get sensor values

16.3.1 Generic configuration set request

This request is used to configure any parameter in the tracker. This downlink should be used when multiple classes of parameters are affected by the downlink.

The data part is a list of parameter entries as defined below. Note that the format is the same than the **response of generic configuration get request** from the tracker.

Parameter entry 1				...
3 Bytes			Variable	...
C-ID	L-ID	S/T	Data	...

Notes

- **C-ID**. Parameter class identifier
- **L-ID:** Local parameter identifier
- **S/T:** Parameter size and type
  - Bit 7-3: Variable size
  - Bit 2-0: type
    - 0 – Deprecated.
    - 1 – Integer 32 bits
    - 2 – Floating point (4 bytes)

- 3 – ASCII string
- 4 – Byte array

- **Data**: Variable data part
  - Deprecated. No data.
  - Integer 32 bits. 4 bytes in big endian (MSB first)
  - Floating point. Single precision, 4 bytes n big endian (MSB first)
  - ASCII string. Max 31 characters. It should exclude the NULL char.
  - Byte array. Max 32 bytes. For 32 bytes, the parameter size will be 0.

This message is answered by the tracker with the **response of generic configuration set request.**

16.3.2 Parameter class configuration set request

This request is used to configure a single parameter class in the tracker. Multiple parameters belonging to this class can be modified.

The data part is a list of parameter entries as defined below. Note that the format is the same than the **answer of a parameter class get request** from the tracker.

Common	Parameter entry 1			Parameter entry 2			…
1 Byte	2 Bytes		Variable	2 Bytes		Variable	...
C-ID	L-ID	S/T	Data	L-ID	S/T	Data	...

Notes

- **C-ID**. Parameter class identifier. This field appears only once per message.
- **L-ID:** Local parameter identifier
- **S/T** Parameter size and type.
  - Bit 7-3: Parameter size
  - Bit 2-0: type
    - 0 – Deprecated.
    - 1 – Integer 32 bits
    - 2 – Floating point (4 bytes)
    - 3 – ASCII string
    - 4 – Byte array
- **Data**: Variable data part
  - Deprecated. No data.
  - Integer 32 bits. 4 bytes in big endian (MSB first)
  - Floating point. Single precision, 4 bytes n big endian (MSB first)ASCII string. Max 31 characters. It should exclude the NULL char.
  - Byte array. Max 32 bytes. For 32 bytes, the parameter size will be 0.

This message is answered by the tracker with the **response of a parameter class set request.**

16.3.3 Generic configuration get request

This request is used to query some configuration parameters from the tracker. This downlink should be used when multiple classes of parameters are affected by the downlink.
The data part is a list of parameter entries as defined below.

2 Bytes		2 Bytes		...
C-ID	L-ID	C-ID	L-ID	...

Notes

- C-ID. Parameter class identifier
- L-ID: Local parameter identifier

This message is answered by the tracker with the **response of generic configuration get request.**

16.3.4 Parameter class configuration get request

This request is used to query configuration parameters belonging to a same parameter
The data part is a list of parameter entries as defined below.

1 Byte	1 Byte	1 Byte	...
C-ID	L-ID	L-ID	...

Notes

- C-ID. Parameter class identifier. This field appears only once per message.
- L-ID: Local parameter identifier

This message is answered by the tracker with the **resp[onse of a parameter class get request.**

16.3.5 BLE connectivity status request

This request is used to query the BLE connectivity status, There is no data part for this request.

16.3.6 Configuration CRC request

This request is used to query the CRC of the configuration.

The data part is a bitmap of parameter groups for which we expect the CRC. If the value 0 is provided, then the global configuration CRC is returned.

2 Bytes
Group bitmap

Notes

The bitmap is formed as follow: Sum of $2^{group\_identifier}$.

Example: The groups 1, 3, 7 are requested. The bitmap will be $2^1 + 2^3 + 2^7 = 138 = 0x8A$

This message is answered by the tracker with the **Response of a CRC configuration request.**

16.3.7 Get sensor value

This generic request is used to read the sensor values.

The data part is the list of sensor identifiers to be read.

Sensor ID 1	---	Sensor ID n
1 byte		1 byte

Notes

- This message is answered by the tracker with the **Response of a sensor get.**
- Sensor ID:
  - 0: Do not use.
  - 1: Accelerometer

16.4 Answers

The answers are responses to the tracker queries.

The format of a request is the following

Byte 0	Byte 1	Bytes [2-variable]
Basic header (**Type =3**)	Answer type	Data

Note

- **Byte 0.** Basic header with type = 3.
- **Byte 1**. Answer types matches the query type:
  - 0 – Aiding-position. The aiding position is needed by the tracker.
  - 1 – Echo-reply: Used only for cellular (LTE network probing).
  - 2 – Update GPS almanac. GPS almanac entries need to be refreshed. Data contains the list of satellites for which the update is needed.
  - 3 – Update BEIDOU almanac. BEIDOU almanac entries need to be refreshed. Data contains the list of satellites for which the update is needed.

16.4.1 Aiding position

TO BE DEFINED

16.4.2 Echo-reply

This answer is the reply of an ***echo-request*** query from the tracker. Remind that the echo request/reply is only used for cellular network. The data part (sequence number) should be copied from the ***echo-request*** query.

Data part
1 byte
Sequence number

16.4.3 Update GPS almanac

This answer is used to setup GPS almanac entries. The format of the data part is the following

Common	Almanac entry 1	Almanac entry 2	...
2 bytes	15 bytes	15 bytes	...
Date	Data-entry 1	Data-entry 2	...

Notes

- **Date**: Number of days since April 7th 2019 in big endian format.
- **Data-entry** is formatted as follow:

Almanac Entry	
1 byte	14 bytes
SV ID	Almanac data

- **SV-ID**: Satellite identifier in the constellation. Start at 0.
- **Almanac data**: Normally the almanac entry contains 15 bytes. It has been observed that the last byte of the entry is always 0xFF (MSB of Omega dot). So sending 14 bytes for an entry is enough and AT3 will add the missing byte upon reception.
- Due to the limited payload size on the LoRa network (EU868, DR0-2, Max: 51 bytes), only 3 entries can be updated in a downlink (2 + 2 + 3*15 = .49).

17 System time update

Currently, the tracker requests the current date/time via the LoRaWAN or the LTE network at the startup time (once successfully joined).
Note that the system clock will drift over time (due to the accuracy of the 32 kHz clock, about 10 ppm) and therefore the system time should be updated from time to time.

The system time is also actualized each time a successful GPS fix is done.
This technique works correctly only if the MT3333 is present in the module and the GPS fix technology is selected as **always-done**.

17.1.1 Systime update using the active network

The system time update is done when connecting to the main or backup network, joining a LoRa network, or registering with an operator on an LTE network. However, the system time update via the network may fail.

To address such a failure, the system automatically request the UTC time once per hour.

Note that this one hour frequency is a multiple of the monitoring period. Example for a monitoring period of 23 minutes this will lead to a retry frequency of 60/23 = 2.6 rounded to 3. So the actual retry period would be 3*23 = 69 seconds.

The netwok manager will redirect the system time update request to the active network (LoRa or LTE).

17.1.2 Clock drift correction

Once the UTC time is acquired, the system time is updated **once every week** when the monitoring period elapses.

18 User Interface

This chapter explains how to configure the User Interface (UI), which includes the following components:

- LEDs
- Buzzer
- Buttons

Please note that certain boards may lack one or more of these UI components.

18.1 User button

18.1.1 Overview

AT3 supports up to 2 buttons, referred to as respectively button1 and button2.

The value of timers used to detect button press and long press actions are configured in the configuration parameter **core_buttons_timing**.

Each button has a specific configuration parameter (***core_button1_map*** / ***core_button2_map***) which is defined by a 32-bit integer bitmap. This parameter establishes a link between a button action and a system event to be triggered.

When an action is performed on a button (such as press, long press, etc), the following process occurs:

1. The system event associated with the button action is triggered (class **button_x**, event **type_btn_y**).
2. If the action is configured to trigger an extra system event via the button's configuration, a second system event is sent.

For example, if button1's configuration links the Position On Demand event to a button press, each press on button1 will generate the following two system events:

- Event 1: class: **button_1**, type: **type_btn_press**
- Event 2: class: **core**, type: **type_core_pod**

18.1.2 Configuration

In the ***core_button1_map*** and ***core_button2_map*** parameters, the link defining the system event linked to each button action is coded on 4 bits:

- Bit 0-3: Event generated on a button press.
- Bit 4-7: Event generated on a button long press.
- Bit 8-11: Event generated on a button single click.
- Bit 12-15: Event generated on a button double clicks.
- Bit 16-19: Event generated on a button triple clicks or above.
- Bit 20-23: Event generated on a button simple sequence (see below)
- Bit 24-31: RFU

The button simple sequence is a hard coded sequence designed for magnet feature activation through Reed switch of Hall effect sensors (it will not activate in case of continuous magnet presence):

1. Switch closed (equivalent to button press) for “button press” duration (by default 3s)
2. Switch release for “button press” action duration (by default 3s)
3. Switch closed for “button press” duration (by default 3s)

System event codes:

- 0 – No action
- 1 – Send the battery level display system event
- 2 – Start and Stop the SOS
- 3 – Request a position on demand (POD)
- 4 – Force an uplink system status notification transmission
- 5 – Start device
- 6 – Stop device
- 7 – Start only SOS
- 8 – Start BLE advertisement for connectivity
- 9..15 - RFU

For example, to trigger the “Battery level display” system event on the single click action, the 4-bit code is 0b0001 (0x1), and of no other action is linked, the ***core_buttonX_map*** parameter would be: 0x00000100.

Each part of the bitmap of ***core_buttons_timing*** *is coded as below*:
Map:

- Bit 0-3: Duration of the button press in seconds.
- Bit 4-7: Duration of the button long press in seconds.
- Bit 8-15: Debounce duration on button 1 in milliseconds.
- Bit 16-23: Debounce duration on button 2 in milliseconds.
- Bit 24-31: RFU

Notes

- The button press and long press duration are the same for all buttons, the duration of the button press cannot be equal or greater than long press, the minimum value for a button press duration is 1s and for long press duration 2s.
- **The duration of the button press and long press cannot be null.**
- If the event *start device* is not mapped on either ***core_button1_map*** or ***core_button2_map***, the device automatically starts after the startup procedure (skip the off mode)
- If the event *stop device* is not mapped on either ***core_button1_map*** or ***core_button2_map***, the device cannot move to the off state.

18.1.3 Special sequences

AT3 supports complex button sequences to trigger system configuration commands.

When a button is held for more than 14 seconds, the AT3 firmware switches into a special configuration command mode, indicated by a unique melody. On devices equipped with a buzzer, the system will emit a beep every second to help gauge the duration.

In the special configuration command mode, AT3 will be expecting one of the following button actions:

Special Sequence	Action	User Interface behavior
1 click, Triple-click or more, 1 Press	Enter the BLE bootloader	Tracker will play 3 beeps
1 click, Double-click, 1 Press	Enter the MCU bootloader	Tracker will play 2 beeps followed by MCU bootloader reset melody
1 click, 1 click, 1 Press	Remove the BLE bond information	Tracker will play 1 beep followed by Bluetooth bond removal melody

Notes

- On success, the tracker plays a success melody followed by the number of beeps defined in the table above.
- On failure, the tracker plays an error melody and exits the special-sequence mode. Failures are detected upon:
  - No button action is performed within 10 seconds
  - An unknown sequence pattern has been detected

18.2 LEDs

18.2.1 Overview

AT3 supports two LEDs. For each of them, AT3 provides a flexible way to configure a specific pattern according to a given system event (refer to Error: Reference source not found). In other word, a given system event can play a given pattern.

Note that in order to choose your favorite pattern, a system CLI command has been implemented to play them.

18.2.2 Configuration

The configuration consists of a pattern and a sequence. The sequence is the number of time the pattern will be played.

For each LED, there is a configuration parameter typed as byte array. The byte array is split in 10 slices of 3 bytes each. Each slice configures a pattern for a system event.
The parameter is defined as:

Slice 1			Slice 2			...	Slice 10		
Byte 0	Byte 1	Byte 2	Byte 0	Byte 1	Byte 2		Byte 0	Byte 1	Byte 2
ext/cls	type	pattern	ext/cls	type	pattern		ext/cls	type	pattern

Where

- **ext/cls:** Pattern extension and system event class.
  - bit 0..4: System event class. Refer to the Event manager section.

    - bit 5..6: Pattern count extension (Most significant bits).
    - bit 7: Pattern inversion:
        - 0 – The pattern is played as defined.
        - 1 – The pattern is inverted.
- **type**: System event type. Refer to the Event manager section.
- **pattern**: Pattern configuration:
    - bit 0..3. Pattern identifier:
        - 0 – None. Not configured
        - 1 – LED off. Pattern duration: Infinite
        - 2 – LED on. Pattern duration: 1s.
        - 3 – Fade in. Pattern duration: 2.5s. Usually used for device power on.
        - 4 – Fade out. Pattern duration: 2.5s. Usually used for device power off.
        - 5 – Blink slow: On: 1000ms, Off: 1000ms. Pattern duration: 2s.
        - 6 – Blink medium: On: 500ms, Off: 500ms. Pattern duration: 1s.
        - 7 – Blink fast: On: 250 ms. Off: 250 ms. Pattern duration: 0.5s
        - 8 – Flash slow: On: 100ms, Off: 2s. Pattern duration: 2.1s.
        - 9 – Flash fast: On: 100ms, Off: 1s. Pattern duration: 1.1s. Usually used for SOS
        - 10 – Heart: On: 100ms, Off: 250s, On: 100ms, Off: 1s. Pattern duration: 1.35s.
    - bit 4..7. Pattern count (Least significant bits): Number of times the pattern is displayed. Used in combination with the **Pattern count extension**. A combined value of 0 means infinite.

Notes

- The pattern inversion is used to alternate a pattern on two LEDs.
- To avoid confusion, while a pattern is played it cannot be interrupted by another pattern except the SOS or pattern ***LED off.***
- If a slice maps the *SOS start* event, the associated pattern will not be interrupted until the sequence duration is complete or if the *SOS stop* event is received. So there is no need to map the SOS stop event. Note that the pattern ***LED off*** won't stop the SOS.
- The battery remaining percentage display has a particular handling. Refer to the dedicated paragraph. **The LED should not be configured for this** (only the button needs to be configured).

18.2.3 Sequence duration calculation

The number of pattern loops is built using the bits 5 and 6 of the **ext/cls** field and the bits 4..7 of the **pattern** field.

The formula to use:

n = **Pattern count** + ((**ext/cls** & 0x60) >> 1)

Where

- n means the number of times the pattern will be played
- The symbol **>>** means right shift
- The symbol **&** means logical AND.

This gives a maximum of 63 number of times the pattern will be played.

Another formula would be:

n = **Pattern count** + (**ext**/**cls**[bit 5] * 16) + (**ext**/**cls**[bit 6] * 32)

The sequence duration becomes
duration = n * pattern duration

18.2.4 Examples

Example 1

The LED 0 is configured with to slowly blink 6 time upon the geolocation complete event.

{06, 02,65,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00}

Only the slice 1 is used and configured with:

- **ext**/**cls** = 0x06.
  - System event class: 6 (Class geolocation)
  - Pattern count extension: 0
  - Inversion = 0
- **type** = 0x02. System event type: geolocation complete
- **Pattern** = 0x65.
  - Pattern identifier: 5 (blink slow)
  - Pattern count = 6.

Number of times the pattern is played (taking into account the extension): 6
Sequence duration = 6 * 2s = 12 seconds.

Example 2

The LED 0 and LED 1 are configured to blink 18 times alternatively upon the button 2 press.

LED 0

{21,00,26,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00}

Only the slice 1 is used and configured with:

- **ext**/**cls** = 0x21.
  - System event class: 1 (Class button 1)
  - Pattern count extension: 1 (bit 5 set)
  - Inversion = 0
- **type** = 0x00. System event type: button press
- **Pattern** = 0x26.
  - Pattern identifier: 6 (blink medium)
  - Pattern count = 16*1 + 2. (16*<Pattern count MSB> + <Pattern Count LSB>)

LED 1

{A1,00,25,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00}

Only the slice 1 is used and configured with:

- **ext**/**cls** = 0xA1.
  - System event class: 1 (Class button 1)
  - Pattern count extension: 1 (bit 5 set)
  - Inversion = 1
- **type** = 0x00. System event type: button press

- **Pattern** = 0x25.
  - Pattern identifier: 5 (blink medium)
  - Pattern count = 2.

Number of times the pattern is played (taking into account the extension): 2 + 1*16 + 0*32 = 18
Sequence duration = 18 * 1s = 18 seconds.

18.2.5 Battery percentage display

On reception of the system event ***type_core_battery_level***, the power manager calls a specific function of the LED manager, which will display the remaining battery percentage on LED 0 (hard-coded):

- 4 blinks: Battery level above 75 % or battery in charge.
- 3 blinks: Battery level between 50% and 75%.
- 2 blinks: Battery level between 25% and 50%.
- 1 blink: Battery level below 25 %.

So there is no need to map the system event ***type_core_battery_level*** to the LED.

18.3 Buzzer

18.3.1 Overview

AT3 manages the buzzer as a melody player. Each melody has an identifier, which can be associated to a given system event.

Note that in order to choose your favorite melody, a system CLI command has been implemented to play them.

18.3.2 Configuration

The configuration consists of a melody and a sequence. The sequence is the number of time the melody will be played.

There is a single configuration parameter typed as byte array. The byte array is split in 10 slices of 3 bytes each. Each slice configures a melody for a system event.

The parameter is defined as:

Slice 1			Slice 2			...	Slice 10		
Byte 0	Byte 1	Byte 2	Byte 0	Byte 1	Byte 2		Byte 0	Byte 1	Byte 2
ext/cls	type	melody	ext/cls	type	melody		ext/cls	type	melody

Where

- **ext/cls:** Melody extension and system event class.
  - bit 0..4: System event class. Refer to the Event manager section.
  - bit 5..7: Melody count extension (Most significant bits).
- **type**: System event type. Refer to the Event manager section.

- **melody**: Melody configuration:
  - bit 0..4. Melody identifier:
    - 0 – None. No configured
    - 1 – Melody 1: Off. Melody duration: Infinite
    - 2 – Melody 2.
    - 3 – Melody 3
    - 4 – Melody 4
    -
  - Bit 5..7. (Least significant bits): Number of times the pattern is displayed. Used in combination with the **Pattern count extension**. A combined value of 0 means infinite.

Notes

- To avoid confusion, while a melody is played it cannot be interrupted by another melody except the SOS or the melody OFF (***melody 1***).
- If a slice maps the SOS start event, the associated melody will not be interrupted until the duration is complete or if the SOS stop event is received.
- Note that the pattern melody Off (**melody 1**) won't stop the SOS.

Examples

Configure the buzzer to play melody 3 infinitely under the button 1 click event and stop on button 1 press:

{00,02,03,00,00,01,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00}

Configure the buzzer to play:

- Infinitely the melody 7 on SOS (class 9, type 8)
- The melody 6 played 10 times on button 2 click (class 1, type 2).
- The melody 8 played 3 times on button 1 click (class 0. type 2)

{09,08,07,21,02,46,00,02,68,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00}

19 Annex A. Configuration CRC calculation

This annex describes the way to calculate the CRC of the configuration.

19.1 CRC algorithm

The CRC algorithm is provided in C code.

```
#define CFG_INIT_CRC 0xFFFFFFFA
static uint32_t _cfg_do_crc(uint32_t crc,  uint8_t* buf, int len )
{
    while( len-- > 0 ) {
        crc = crc ^ *buf++;
        for( int i = 0; i < 8; i++ ) {
            uint32_t mask = -( crc & 1 );
            crc           = ( crc >> 1 ) ^ ( 0xEDB88320 & mask );
        }
    }
    return ~crc + 3;
}
```

Note that before calling the function, the CRC parameter should be initialized to the value `CFG_INIT_CRC`.

19.2 Calculating the CRC for a parameter

A configuration parameter contains a descriptor and the value defined as follow:

```
/*!
 * \brief Value of a parameter
 */
typedef union {
    int32_t integer;                //!< Signed integer
    float decimal;                  //!< Floating point
    char* ascii;                    //!< Pointer to an ASII char (NULL terminated)
    uint8_t* barray;                //!< Pointer to an array of bytes
} srv_config_param_value_t;

/*!
 * \brief Parameter descriptor
 *
 * \warning Do not change the order
 */
typedef struct {
      struct {
            uint16_t identifier;    //!< Unique parameter identifier
```

```
            uint8_t type;          //!< Value type refer to srv_config_param_type_t
            uint8_t length;        //!< string length of the barray value
        } descriptor;              //!< Descriptor associated to the parameter
    srv_config_param_value_t value;  //!< Parameter value
} __attribute__((__packed__)) srv_config_param_descriptor_t;
```

The CRC of a parameter is calculated over:

- The full descriptor (`srv_config_param_descriptor_t`) for integer and floating point types.
- The descriptor length is set to 0 for all parameter type except for byte array.
- The descriptor only (`descriptor`) followed by the value pointed by the value field for barray and string types. The byte array size is given by the `length` field of the descriptor.
- The CRC for a string includes the null char (C string termination). The length should be determined (strlen() + 1) and not exctracted from the descriptor.

19.3 Calculating the CRC for a single group

A parameter group CRC is calculated over all parameters belonging to the group. Note that the CRC must be initialized only once before starting the CRC calculation of all parameters.

Notes

- The parameter ordering must be respected.
- A missing parameter must be skipped for the CRC calculation.

19.4 Calculating the global CRC

The global CRC is calculated over all parameter groups except the ***internal*** one. Note that the CRC must be initialized only once before starting the CRC calculation of all groups.

Notes

- The parameter group ordering must be respected.
- A non-existent group must be skipped for the calculation (e.g. For non combo-compact tracker, the ***cellular*** group does not exist).