Aerobits

Subsystems for the
UAS integration into
the airspace

trkME

Data sheet - User manual

Contents

---

1 Introduction

The **trkME** tracker is a versatile, multi-system tracking device designed to enhance the safety and operational efficiency of UAS. It features a comprehensive range of RF systems, including ADS-B receiver. This ensures robust communication and positioning capabilities, essential for navigating congested airspaces. It is available in three different functional variants: **trkME**, **trkME+** and **trkME-PRO**

Table 1: Product Variants

Feature	trkME	trkME+	trkME PRO
			
Role	Remote identification	Remote identification Anti-collision	Remote identification Anti-collision - surrounding awareness Cooperation with bigger aviation
RF type	transmitter	transmitter receiver	transmitter receiver
RemoteID – LTE	✓ (Cat. 1)	✓ (Cat. 1)	✓ (Cat. 1)
RemoteID OUT [Wi-Fi, BT]	✓	✓	✓
RemoteID IN [Wi-Fi, BT]	×	✓ (Up to 1km)	✓
FLARM IN/OUT	×	✓	✓
ADS-B IN	×	✓ (up to 15km)	✓
ADS-B OUT	×	×	✓
Built-in GNSS	✓	✓	✓
Built-in pressure sensor	✓	✓	✓
Transmitter power [dBm]	+18 (BT) +20 (Wi-Fi)	+18 (BT) +20 (Wi-Fi) +14 (FLARM)	+35 (ADS-B) +18 (BT) +20 (Wi-Fi) +14 (FLARM)
Receiver sensitivity [dBm]	-167 (GNSS)	-94 (ADS-B) -109 (FLARM) -167 (GNSS) -103 (BT) -103 (Wi-Fi)	-94 (ADS-B) -109 (FLARM) -167 (GNSS) -103 (BT) -103 (Wi-Fi)
UART interface [baud]	(115k / 921k / 3M)	(115k / 921k / 3M)	(115k / 921k / 3M)
USB interface	(2.0)	(2.0)	(2.0)
Power supply [V]	5	5	5
Current consumption [mA]	320	400	450
Weight [g]	50	50	50
Dimensions [mm]	65x46x12	65x46x12	65x46x12

Note:

× - not included in this variant, ✓ - included

Note:

Currently available is the **trkME** product variant. Products **trkME+** and **trkME PRO** will be available soon. Check availability with our sales department at: **sales@aerobits.pl**

trkME has FLARM IN/OUT transceiver, BLE/Wi-Fi RemoteID transmitter and also has multi-constellation GNSS sensor on board to provide best accuracy. The LTE connectivity allows usage in all LTE/4G rich environments without the need for any additional cabling to send data.

trkME opens the way to the safe integration of UAS into non-segregated airspace, implementation of the **Detect and Avoid** algorithms and reduce separation between airspace users.

Aerobits **trkME** is designed to meet requirements of direct remote drone identification and localization in **ASTM/ASD-STAN** standard. Using the BLE broadcast and WiFi Nan, Beacon frames technology the device provides surveillance and drone operator identification capability based on any modern mobile devices such as smartphone or tablet. Our device fits in with the concept of Network Remote Identification and allow broadcast remote drone identification via LTE.

It is a perfect solution for conducting VLOS/BVLOS operation where safety is critical.

Note:

The device to operate on FLARM frequency requires FLARM UAS license. The license must be obtained with the device from Aerobits upon purchase.

Important:

Each firmware version becomes its own documentation. This document is relevant for firmware version v2.78.0. If your firmware version is different please find relevant documentation on our website aerobits.pl.

1.1 Features

- **Real-time aircraft tracking on 1090 MHz and 868 MHz**
- **Nationwide traffic management systems (manned and unmanned)**
- **Integrated GNSS source and pressure sensor**
- **Licensed FLARM transceiver (0.025 Watt output power)**
- **Remote drone identification BLE and Wi-Fi standards**
- **Network based Remote Identification (central monitoring)**
- **Programming via AT commands**
- **LTE modem to track aircraft via LTE 4G Cat1**

For more information please contact support@aerobits.pl.

2 Technical parameters

2.1 Basic technical information

Table 2: General technical parameters

Parameter	Description	Typ.	Unit
First Band	ADS-B	1090	MHz
Second Band	FLARM	868 or 915	MHz
Third Band	RID BLE	2400	MHz
Fourth Band	RID Wi-Fi	2400	MHz
Fifth Band	GNSS	1575	MHz
Sensitivity (ADS-B)		-94	dBm
Sensitivity (FLARM)		-109	dBm
Sensitivity (BLE)		-103	dBm
Sensitivity (Wi-Fi)		-103	dBm
Sensitivity (GNSS)		-167	dBm
RF Output power (FLARM)		+14	dBm
RF Output power (BLE)		+18	dBm
RF Output power (Wi-Fi)		+20	dBm
LTE Cat. 1	Data transport layer (global bands)		

2.2 Electrical specification

2.2.1 Basic electrical parameters

Table 3: General electrical parameters

Parameter	Value
Power connector	USB-C and JST GH 6 pin
Power supply	5 V
Power consumption	< 600 mA

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

2.2.2 JST PIN definition

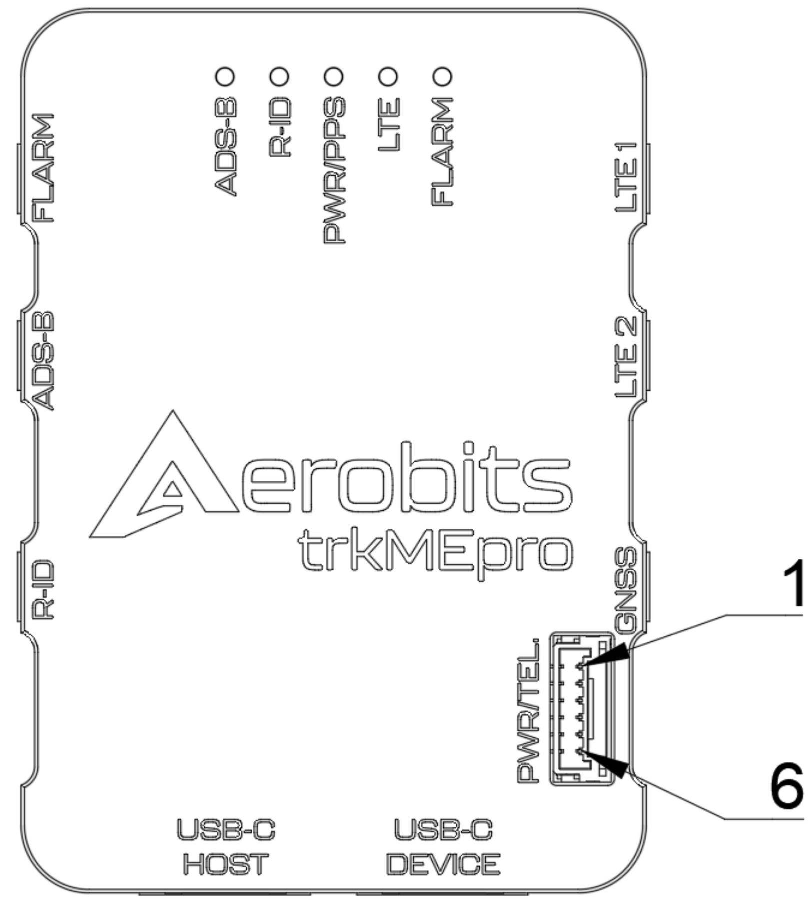


Table 4: Descriptions of JST connector pins.

PIN	Name	Function
1	+5 V	Power supply
2	RX	MAVLink, AERO RXD
3	TX	MAVLink, AERO TXD
4	NC	Not connected
5	NC	Not connected
6	GND	Ground

2.2.3 LED indicators

Table 5: Descriptions of LEDs.

LED	Color	Function
ADS-B	Green	Flashing – reception of 1090 MHz avionics frame (ADS-B)
ADS-B	Red	Flashing – send of 1090 MHz avionics frame (ADS-B)
FLARM	Green	Flashing – reception of FLARM frame
FLARM	Red	Flashing – send of FLARM frame
RID	Green	Flashing – reception of RID frame
RID	Red	Flashing – send of RID frame

continues on next page

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

Table 5 – continued from previous page

LED	Color	Function
PWR/PPS	Green	Constant light - Power supply presence Off – No power, connect or recharge power source
PWR/PPS	Red	Flashing – reception of PPS signal
LTE	Red	Flashing – LTE communication initialized Constant light – LTE communication in progress Off – No mobil network, wait or change position for better network coverage

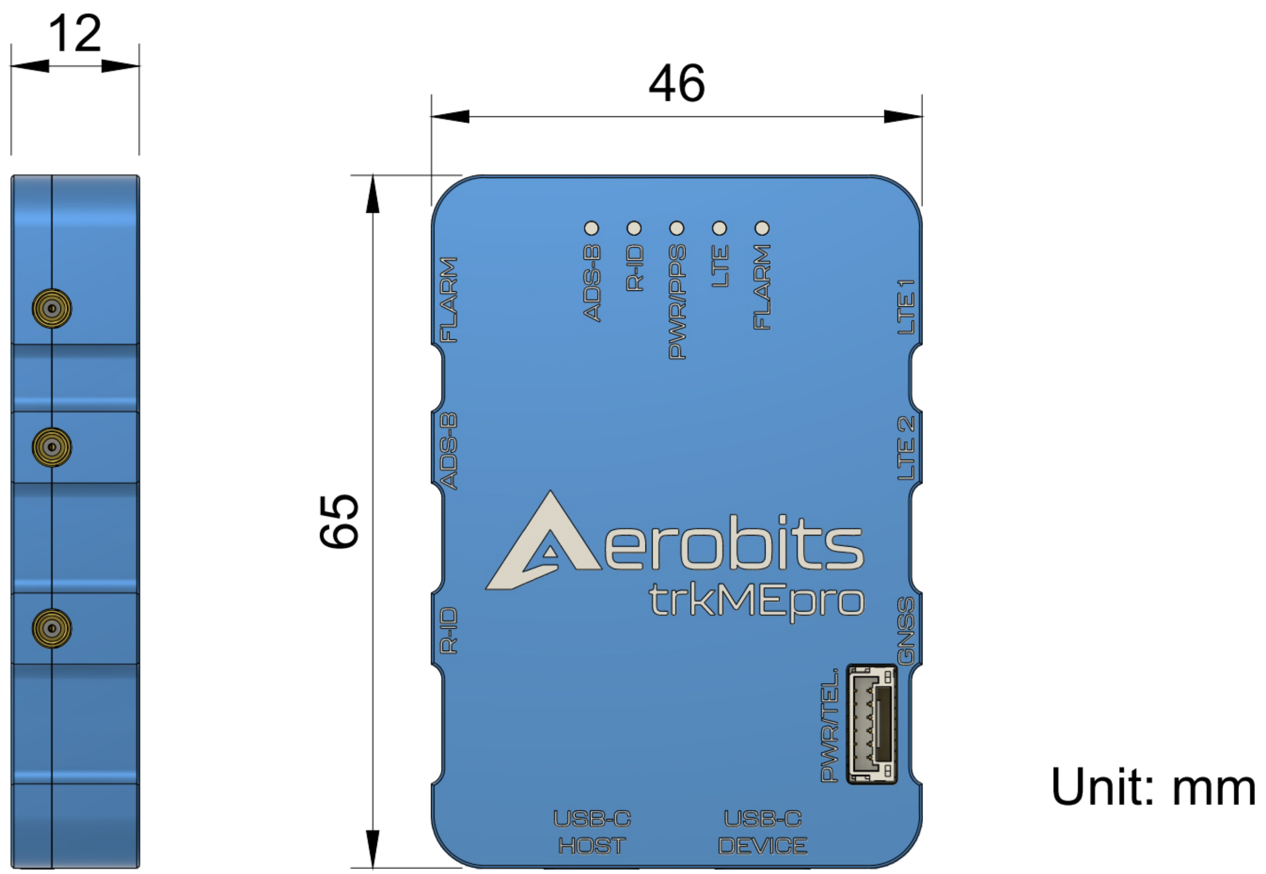
2.3 Mechanical specification

2.3.1 Mechanical parameters

Table 6: Mechanical parameters of the trkME

Parameter	Value
Dimensions	65 mm x 46 mm x 12 mm
Weight	60 g

2.3.2 Dimensions



This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

2.3.3 Connectors

Table 7: Descriptions of used connectors.

Description	Type	Function	Mating connector
USB-C Device	USB4110-GF-A	Power and Data	USB-C
USB-C Host	USB4110-GF-A	Data	USB-C
PWR/TEL.	BM06B-GHS-TBT	Power and Data	JST GH 6 pin
ADS-B FLARM RID GNSS LTE1 LTE2	MMCX-4058JEGR	Antenna	ASMK025X174S11

3 UART configuration

Communication between module and host device is done using UART interface.

In CONFIGURATION and BOOTLOADER state transmission baud is fixed at 115200bps.

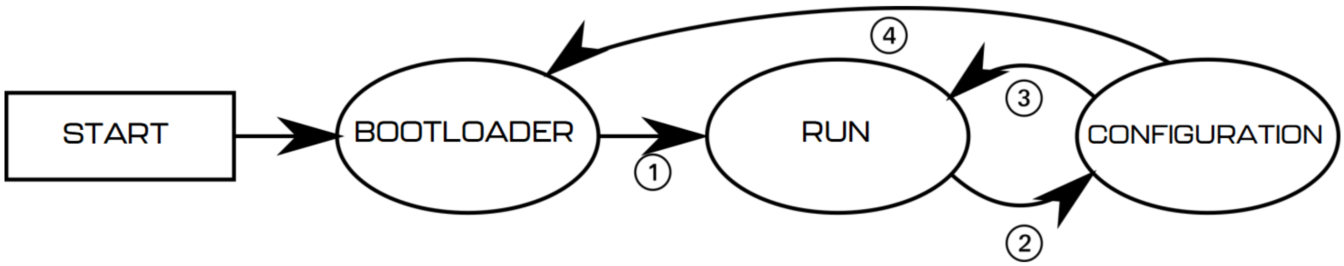
The UART interface uses settings as described in table below:

Table 8: Descriptions of UART settings.

Parameter	Min.	Typ.	Max	Unit
Baud	57600	921600	3000000	bps
Stop Bits Number	–	1	–	–
Flow Control	–	None	–	–
Parity Bit	–	None	–	–

4 Principle of operation

During work module goes through multiple states. In each state operation of the module is different. Each state and each transition is described in paragraphs below.



4.1 States of operation

4.1.1 BOOTLOADER state

This is an initial state of after restart. Firmware update is possible here. Typically module transits automatically to RUN state. It is possible to lock module in this state (prevent transition to RUN state) using one of BOOTLOADER triggers. UART baud is constant and is set to 115200bps. After powering up module, it stays in this state for up to 3 seconds. If no BOOTLOADER trigger is present, module will transit to RUN state. Firmware upgrade is possible using Micro ADS-B App software. For automated firmware upgrading scenarios, aerobits_updater software is available. To acquire this program please contact: support@aerobits.pl.

4.1.2 RUN state

In this state module is broadcasting drone identification data. In this state module is working and receiving the data from aircrafts. It uses selected protocol to transmit received and decoded data to the host system. In this state of operation module settings are loaded from non-volatile internal memory, including main UART interface's baud.

4.1.3 CONFIGURATION state

In this mode change of stored settings is possible. Operation of the module is stopped and baud is set to fixed 115200bps. Change of settings is done by using AT-commands. Changes to settings are stored in non-volatile memory on exiting this state. Additional set of commands is also available in this state, allowing to e.g. reboot module into BOOTLOADER state, check serial number and firmware version. It is possible to lock module in this state (similarly to BOOTLOADER) using suitable command.

4.2 Transitions between states

For each of state transitions, different conditions must be met, which are described below. Generally, the only stable state is RUN. Module always tends to transit into this state. Moving to other states requires host to take some action.

4.2.1 BOOTLOADER to RUN transition

BOOTLOADER state is semi-stable: the module requires additional action to stay in BOOTLOADER state. The transition to RUN state will occur automatically after short period of time if no action will be taken. To prevent transition from BOOTLOADER state, one of following actions must be processed:

- Send *AT+LOCK=1* command while device is in BOOTLOADER state (always after power on for up to 3s)
- Send `AT+REBOOT_BOOTLOADER` command in CONFIGURATION state. This will move to BOOTLOADER state and will lock module in this state.

If none of above conditions are met, the module will try to transit into RUN state. Firstly it will check firmware integrity. When firmware integrity is confirmed, module will transit into RUN state, if not, it will stay in BOOTLOADER state.

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

To transit into RUN state:

- If module is locked, send `AT+LOCK=0` command

When module enters RUN mode it will send `AT+RUN_START` command.

4.2.2 RUN to CONFIGURATION transition

To transit from RUN into CONFIGURATION state, host should do one of the following:

- Send `AT+CONFIG=1` (using current baud).

When module leaves RUN state it sends `AT+RUN_END` message, then `AT+CONFIG_START` message on entering CONFIGURATION state. The former is sent using baud from settings, the latter always uses 115200bps baud.

4.2.3 CONFIGURATION to RUN transition

To transit from CONFIGURATION into RUN state, host should do one of the following:

- Send `AT+CONFIG=0` command.

When module leaves CONFIGURATION state it sends `AT+CONFIG_END` message, then `AT+RUN_START` message on entering RUN state. The former is always sent using 115200bps baud, the latter uses baud from settings.

4.2.4 CONFIGURATION to BOOTLOADER transition

To transit from CONFIGURATION into BOOTLOADER state, host should do one of the following:

- Send `AT+REBOOT_BOOTLOADER` command.
- Send `AT+REBOOT` and when module enters BOOTLOADER state, prevent transition to RUN state.

When entering the bootloader state, the module sends `AT+BOOTLOADER_START` .

5 System configuration

In RUN state, operation of the module is determined based on stored settings. Settings can be changed in CONFIGURATION state using AT-commands. Settings can be written and read.

Note:

New values of settings are saved in non-volatile memory when transitioning from CONFIGURATION to RUN state.

Settings are restored from non-volatile memory during transition from BOOT do RUN state. If settings become corrupted due to memory fault, power loss during save, or any other kind of failure, the settings restoration will fail, loading default values and displaying the `AT+ERROR (Settings missing, loaded default)` message as a result. This behavior will occur for each device boot until new settings are written by the user.

5.1 System settings

5.1.1 Write settings

After writing a new valid value to a setting, an `AT+OK` response is always sent.

`AT+SETTING=VALUE`

For example `AT+SYSTEM_STATISTICS=1`

Response: `AT+OK`

5.1.2 Read settings

`AT+SETTING?`

For example: `AT+SYSTEM_STATISTICS?`

Response: `AT+SYSTEM_STATISTICS=1`

5.1.3 Settings description

`AT+SETTING=?`

For example: `AT+SYSTEM_STATISTICS=?`

Response:

```
Setting: SYSTEM_STATISTICS
Description: System statistics protocol(0:none, 1:CSV, 2:JSON)
Access: Read Write
Type: Integer decimal
Range (min.): 0
Range (max.): 2
Preserved: 1
Requires restart: 0
```

5.1.4 Errors

Errors are reported using following structure:

`AT+ERROR (DESCRIPTION)`

DESCRIPTION is optional and contains information about error.

5.1.5 Command endings

Every command must be ended with one of the following character sequences: `"\n"`, `"\r"` or `"\r\n"`. Commands without suitable ending will be ignored.

5.1.6 Uppercase and lowercase

All characters (except preceding `AT+`) used in command can be both uppercase and lowercase, so following commands are equal:

`AT+SYSTEM_STATISTICS?`

`AT+sYSTEM_staTISTICS?`

Note:

This statement is true in configuration state, not in bootloader state. in bootloader state all letters must be uppercase.

5.1.7 Settings

Table 9: Descriptions of system settings.

Setting	Min	Max	Def	Comment
BAUDRATE	0	3	0	Baudrate in RUN state 0 - 115200bps 1 - 921600bps 2 – 3000000bps 3 – 57600bps
SYSTEM_LOG	0	1	0	System logs 0 - disable 1 - enable
SYSTEM_STATISTICS	–	–	None	System statistics protocol: None CSV

5.1.8 Example

As an example, to switch The Aerobits device to CSV protocol, one should send following commands. "<<" indicates command sent to module, ">>" is a response.

```
<< AT+CONFIG=1\r\n
>> AT+OK\r\n
<< AT+ADSB_RX_PROTOCOL_DECODED=1\r\n
>> AT+OK\r\n
<< AT+CONFIG=0\r\n
>> AT+OK\r\n
```

5.2 Commands

Apart from settings, module supports set of additional commands. Format of this commands are similar to those used for settings, but they do not affect operation of module in RUN state.

5.2.1 Commands in BOOTLOADER and CONFIGURATION state

AT+LOCK

`AT+LOCK=1` - Set lock to enforce staying in BOOTLOADER or CONFIGURATION state `AT+LOCK=0` - Remove lock `AT+LOCK?` - Check if lock is set

AT+BOOT

`AT+BOOT?` - Check if module is in BOOTLOADER state

Response:

`AT+BOOT=0` - module in CONFIGURATION state `AT+BOOT=1` - module in BOOTLOADER state

5.2.2 Commands in CONFIGURATION state

AT+CONFIG

`AT+CONFIG=0` - Transition to RUN state. `AT+CONFIG?` - Check if module is in CONFIGURATION state.

Response:

`AT+CONFIG=0` - module in RUN state `AT+CONFIG=1` - module in CONFIGURATION state (baudrate 115200) `AT+CONFIG=2` - module in CONFIGURATION state (baudrate as set)

AT+SETTINGS?

`AT+SETTINGS?` - List all settings. Example output:

```
AT+BAUDRATE=0
AT+BOOT=0
AT+CONFIG=1
AT+DEVICE=TR-1F
AT+FIRMWARE_VERSION=2.72.1.0 (Jun 17 2024)
AT+LOCK=0
AT+SERIAL_NUMBER=22-0000309
AT+SYSTEM_LOG=0
AT+SYSTEM_STATISTICS=0
AT+ADSB_RX_PROTOCOL_DECODED=1
AT+ADSB_RX_PROTOCOL_INC=0
AT+ADSB_RX_PROTOCOL_RAW=0
AT+ADSB_STATISTICS=1
AT+ADSB_TX_EMITTER_CAT=0
AT+ADSB_TX_ENABLED=1
AT+ADSB_TX_ICAO=000000
AT+ADSB_TX_IDENT=
AT+ADSB_TX_ON_BOOT=1
```

(continues on next page)

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

(continued from previous page)

```
AT+ADSB_TX_PWR=2
AT+ADSB_TX_SQUAWK=0000
AT+ADSB_TX_SURFACE=0
AT+ADSB_TX_TRANSPONDER_PRESENT=0
AT+FLARM_INFO=LIBFLARM-2.03, expires: 2025-03-01, status: OK
AT+FLARM_RX_PROTOCOL_DECODED=1
AT+FLARM_STATISTICS=0
AT+FLARM_TX=1
AT+FLARM_TX_AIRCRAFT_TYPE=13
AT+GNSS_RX_PROTOCOL_RAW=0
AT+SENSOR_PROTOCOL_DECODED=0
AT+ASTERIX_SAC=1
AT+ASTERIX_SIC=129
```

AT+HELP

AT+HELP - Show all settings and commands with descriptions. Example output:

```
SETTINGS:
SYSTEM:
    AT+BAUDRATE=0 [Baudrate of serial interface (0:115200, 1:921600, 2:3000000,
↪3:57600)]
    AT+BOOT=0 [Is firmware in bootloader mode]
    AT+CONFIG=1 [CONFIG mode (0:disable, 1:baudrate 115200, 2:baudrate as set)]
    AT+DEVICE=IDME-PRO [Device type's name]
    AT+LOCK=0 [Device in CONFIG mode (0:no lock, 1:lock)]
    AT+SERIAL_NUMBER=18099300000323 [Device's serial number]
    AT+SYSTEM_LOG=0 [System logs (0:disable, 1:enable)]
    AT+SYSTEM_STATISTICS=0 [System statistics protocol(0:none, 1:CSV, 2:JSON)]
    AT+FIRMWARE_VERSION=1.22.5.0 (Aug 7 2024) [Device's firmware version]
GNSS:
    AT+GNSS_RX_PROTOCOL_RAW=0 [GNSS_RX RAW protocol (0:none, 5:NMEA)]
SENSORS:
    AT+SENSORS_PROTOCOL_DECODED=0 [SENSORS decoded protocol (0:none, 1:CSV,3:JSON)]
COMMANDS:
    AT+3RD_PARTY_LICENSES [Displays licenses of third party software]
    AT+BLUETOOTH_MAC [Bluetooth device mac address]
    AT+DRONE_ID_OPERATOR_ID [Operator message payload]
    AT+HELP [Show this help]
    AT+INFO [Display device information]
    AT+REBOOT [Reboot system]
    AT+REBOOT_BOOTLOADER [Reboot to bootloader]
    AT+SETTINGS_DEFAULT [Loads default settings]
    AT+TEST [Responds "AT+OK"]
    AT+WIFI_MAC [WiFI device mac address]
```


This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

AT+SETTINGS_DEFAULT

`AT+SETTINGS_DEFAULT` - Set all settings to their default value.

AT+SERIAL_NUMBER

`AT+SERIAL_NUMBER?` - Read serial number of module.

Response:

`AT+SERIAL_NUMBER=07-0001337`

AT+FIRMWARE_VERSION

`AT+FIRMWARE_VERSION?` - Read firmware version of module.

Response:

`AT+FIRMWARE_VERSION=2.73.1.0 (Jun 27 2024)`

AT+REBOOT

`AT+REBOOT` - Restart module.

AT+REBOOT_BOOTLOADER

`AT+REBOOT_BOOTLOADER` - Restart module to BOOTLOADER state.

> **Note:**
>
> NOTE: This command also sets lock.

5.2.3 Commands in RUN state

`AT+CONFIG=1` - transition to CONFIGURATION state (baudrate 115200). `AT+CONFIG=2` - transition to CONFIGURATION state (baudrate as set).

> **Note:**
>
> NOTE: This command also sets lock.

6 Protocols

Each system has protocols unique to it, but protocols common to all systems such as the CSV protocol are also used. All the protocols used in our products will be presented below.

6.1 Decoded protocols

- CSV - comma separated values as plain text
- Mavlink - binary protocol used by Pixhawk and other flights controllers
- JSON - text based format represents data as structured text
- GDL90 - binary protocol for ingestion into `Electronic Flight Bag` applications
- ASTERIX - binary protocol used for exchanging surveillance-related information in air traffic management

6.2 RAW protocols

- HEX - hexadecimal protocol is unprocessed data sended by aircraft
- BEAST - binary protocol used by program like dump1090
- JSON - it is JSON standard format with raw HEX frames inside structures
- HEXd - it is HEX protocol without extra fields, special prepared for dump1090

6.3 Statistics protocol

- CSV - comma separated values as plain text

6.4 CSV protocol (AERO)

CSV protocol is simple text protocol, that allows fast integration and analysis of tracked aircrafts. CSV messages start with '`#`' character and ends with "`\r\n`" characters. There are following types of messages:

1. ADS-B Aircraft message,
2. FLARM Aircraft message,
3. UAT Aircraft message,
4. RID Aircraft message,
5. Systems statistics messages,
6. Sensors messages.

> **Note:**
>
> In future versions, additional comma-separated fields may be introduced to any CSV protocol message, just before CRC field, which is guaranteed to be at the end of message. All prior fields are guaranteed to remain in same order.

6.4.1 CRC

Each CSV message includes CRC value for consistency check. CRC value is calculated using standard CRC16 algorithm and its value is based on every character in frame starting from '#' to last comma ',' (excluding last comma). After calculation, value is appended to frame using hexadecimal coding. Example function for calculating CRC is shown below.

```
uint16_t crc16(const uint8_t* data_p, uint32_t length){
    uint8_t x;
    uint16_t crc = 0xFFFF;
    while (length--){
        x = crc>>8 ^ *data_p++;
        x ^= x>>4;
        crc = (crc<<8) ^ ((uint16_t)(x<<12)) ^ ((uint16_t)(x<<5)) ^ ((uint16_t)x);
    }
    return swap16(crc);
}
```

6.5 MAVLink protocol

MAVLink (Micro Air Vehicle Link) is a lightweight, efficient communication protocol designed primarily for unmanned aerial vehicles (UAVs), but it is also used in other robotic systems, including ground and marine vehicles. MAVLink facilitates communication between a ground control station (GCS) and an onboard autopilot, as well as between onboard components such as sensors, cameras, and controllers.(here).

6.5.1 Common Use Cases

- Flight Control: Communicating flight commands and receiving telemetry from UAVs.
- Sensor Integration: Transmitting data from onboard sensors to the ground station or other components.
- Mission Planning: Sending waypoints and mission plans to the UAV from the ground station.
- Remote Monitoring: Monitoring the health and status of the UAV during flight.

Overall, MAVLink is a versatile and robust protocol that has become the standard for UAV communication, particularly in the open-source community.

6.6 JSON protocols

JSON (JavaScript Object Notation) is a lightweight, text-based data interchange format that is easy for humans to read and write and easy for machines to parse and generate. JSON is widely used for transmitting data between a server and a web application, as well as for configuration files, data storage, and APIs.

Each message is encoded as separate JSON object, without any excess whitespace, consisting of fields described in table below:

```
{
    "src": "ID-0000001",
    "ts": 69061337,
    "ver": 1,
    "gnss": {
    }
}
```

Table 10: Description of main JSON fields.

JSON Field	Unit	Example	Description
src	–	ID-0000001	OEM TT serial number.
ts	milliseconds	69061337	Timestamp in milliseconds, relative to last UTC midnight. Value 69061337 encodes 19:11:01.337. Omitted if unknown.
ver	–	1	JSON protocol version. See details below.
gnss	–	{...}	One or more of the data fields, described in subchapters below.

Note:

The order of JSON object fields in any part of message may vary between firmware revisions and messages.

Some JSON objects have fields, of which values may sometimes be unknown. In this case, they are skipped in JSON output. In following chapters, each of those fields are explicitly marked as omittable.

Note:

In case of JSON objects consisting of only omittable fields, if none of them are set, the whole object may be omitted.

The *ver* field indicates JSON protocol version. Future ICD versions may introduce additional fields without chang- ing the version number. If a breaking change occurs in Ground Station with Linux JSON specification, the version number is guaranteed to be incremented.

Note:

The version number of JSON protocol described in this document is 1.

6.6.1 Status section

The “status” section contains status information related to OEM TT-Multi-RF itself. The example JSON message with this section fields described:

```
{
    "src": "ID-0000001",
    "ts": 69061337,
    "ver": 1,
    "status": {
        "fw": "30903679(Jan 15 2021)",
    }
}
```

Table 11: Description of status JSON fields.

JSON Field	Unit	Example	Description
src	–	ID-0000001	See table *Description of main JSON fields.* (page 20).
ts	milliseconds	69061337	See table *Description of main JSON fields.* (page 20).
ver	–	1	See table *Description of main JSON fields.* (page 20).
status	–	type of message	
fw	–	30903679(Jan 15 2021)	Firmware version, with same syntax as AT+FIRMWARE_VERSION command. Value 30903679 is version 3.9.3.679.

6.7 Statistics protocol

Statistic protocols contains system information. These information can be used to diagnose system health.

6.7.1 CSV statistic protocol

Format of that frame is shown below:

`#S:CPL,UPT,CRC\r\n`

CPL - CPU load in ‰

UPT - Time since statistic was enabled

CRC - Value is calculated using standard CRC16 algorithm

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

7 ADS-B receiver subsystem

Important:

This part of documentation is relevant only for devices which have **ADS-B IN** functionality

7.1 Settings

Table 12: Descriptions of ADS-B settings.

Setting	Min	Max	Def	Comment
ADSB_RX_LNA_MODE	0	1	1	ADS-B LNA mode: 1 - High gain 0 - Bypass
ADSB_RX_PROTOCOL_DECODED	–	–	CSV	ADS-B decoded protocol: None CSV Mavlink JSON GDL90 ASTERIX
ADSB_RX_PROTOCOL_INC	0	2	0	Reporting mode of decoded ADS-B targets: 0 - once per second, always 1 – once per second, if data updated 2 – immediately, only after position update
ADSB_RX_PROTOCOL_RAW	–	–	None	ADS-B raw protocol: None HEX BEAST JSON HEXd – dump1090
ADSB_STATISTICS	–	–	CSV	ADS-B statistics protocol: None CSV JSON

Note:

To reduce sensitivity, set the LNA in bypass mode.

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

7.1.1 ADS-B reports

ADS-B reports update received data per aircraft, not per all received airplanes.

For example:

If we have *ADSB_RX_PROTOCOL_INC = 1*, then all received ADS-B airplanes will be updated once per second, one by one, rather than all at the same time.

Another example:

If we have *ADSB_RX_PROTOCOL_INC = 2*, then all received ADS-B airplanes will be updated ASAP, but only if the position data has been changed.

7.1.2 ASTERIX settings

Note:

Works only if `ADSB_RX_PROTOCOL_DECODED=ASTERIX` is selected

Table 13: Descriptions of Asterix settings.

Setting	Min	Max	Def	Comment
ASTERIX_SAC	0	255	1	Setting SAC for ASTERIX protocol (Visible when ADSB_DECODED_PROTOCOL=5)
ASTERIX_SIC	0	255	129	Setting SIC for ASTERIX protocol (Visible when ADSB_DECODED_PROTOCOL=5)

7.2 Protocols

7.2.1 ADS-B decoded protocols

ADS-B CSV protocol

This message describes state vector of aircraft determined from ADS-B messages and is sent once per second. The message format is as follows:

```
#A:ICAO,FLAGS,CALL,SQ,LAT,LON,ALT_BARO,TRACK,VELH,VELV,SIGS,SIGQ,FPS,NICNAC,ALT_GEO,
ECAT,CRC\r\n
```

Table 14: Descriptions of ADS-B fields.

#A	Aircraft message start indicator	Example value
ICAO	ICAO number of aircraft (3 bytes)	3C65AC
FLAGS	Flags bitfield, see table below *Descriptions of ADS-B FLAGS field.* (page 24)	1
CALL	Callsign of aircraft	N61ZP
SQ	SQUAWK of aircraft	7232
LAT	Latitude, in degrees	57.57634
LON	Longitude, in degrees	17.59554

continues on next page

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

Table 14 – continued from previous page

#A	Aircraft message start indicator	Example value
ALT_BARO	Barometric altitude, in feet	5000
TRACK	Track of aircraft, in degrees [0,360)	35
VELH	Horizontal velocity of aircraft, in knots	464
VELV	Vertical velocity of aircraft, in ft/min	-1344
SIGS	Signal strength, in dBm	-92
SIGQ	Signal quality, in dB	2
FPS	Number of raw MODE-S frames received from aircraft during last second	5
NICNAC	NIC/NAC bitfield, see table 11 (v2.6.0+)	31B
ALT_GEO	Geometric altitude, in feet (v2.6.0+)	5000
ECAT	Emitter category, *ADS-B emitter category values in CSV protocol.* (page 24) (v2.7.0+)	14
CRC	CRC16 (described in CRC section)	2D3E

Table 15: Descriptions of ADS-B FLAGS field.

Value	Flag name	Description
0x0001	PLANE_ON_THE_GROUND	The aircraft is on the ground
0x0002	PLANE_IS_MILITARY	The aircraft is military object
0x0100	PLANE_UPDATE_ALTITUDE_BARO	During last second, barometric altitude of this aircraft was updated
0x0200	PLANE_UPDATE_POSITION	During last second, position (LAT & LON) of this aircraft was updated
0x0400	PLANE_UPDATE_TRACK	During last second, track of this aircraft was updated
0x0800	PLANE_UPDATE_VELO_H	During last second, horizontal velocity of this aircraft was updated
0x1000	PLANE_UPDATE_VELO_V	During last second, vertical velocity of this aircraft was updated
0x2000	PLANE_UPDATE_ALTITUDE_GEO	During last second, geometric altitude of this aircraft was updated

The NIC/NAC bitfield is transmitted in big endian hexadecimal format without leading zeros. Table 11 describes its bitfield layout. The meaning of NIC/NAC indicators is exactly the same as described in ED-102A.

Table 16: Structure of NIC/NAC bitfield in CSV protocol.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$Reserved$				NAC_p				NAC_v			NIC_{baro}	NIC			

The emitter category values returned in ecat field is shown in table below:

Table 17: ADS-B emitter category values in CSV protocol.

"ecat" value	Description
0	Unknown.
1	Light (below 15500 lbs.).
2	Small (15500 - 75000 lbs.).
3	Large (75000 - 300000 lbs.).
4	High-Vortex Large (aircraft such as B-757).
5	Heavy (above 300000 lbs.).
6	High performance (above 5g acceleration and above 400 knots).
7	Rotorcraft.
8	Reserved.
9	Glider, Sailplane.

continues on next page

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

Table 17 – continued from previous page

"ecat" value	Description
10	Lighter-Than-Air.
11	Parachutist, Skydiver.
12	Ultralight, hang-glider, paraglider.
13	Reserved.
14	Unmanned Aerial Vehicle.
15	Space, Trans-atmospheric Vehicle.
16	Reserved.
17	Surface Vehicle - Emergency Vehicle.
18	Surface Vehicle - Service Vehicle.
19	Point Obstacle (includes Tethered Balloons).
20	Cluster obstacle.
21	Line obstacle.

If data of any field of frame is not available, then it is transmitted as empty. For example:

```
#A:4D240E,3F00,,7273,53.47939,14.55892,28550,23,510,1408,-71,5,9,938,28850,,A9FE\r\n

#A:4D240E,3F00,,7273,53.52026,14.58906,29075,23,506,1600,,,,,,,C1EC\r\n
```

Note:

SIGS and **SIGQ** fields are updated based on raw MODE-S frames. They are calculated from frames received in last second. If there were no receiver frames (FPS=0), those fields will not be updated.

Note:

LAT and **LON** are transmitted differently for aircraft on the surface and in airborne. ADSB messages send from airborne aircrafts are unambiguous. Surface messages needs reference position which is used to determine final position of the aircraft. Aerobits devices if it is possible use their own position as reference. For devices without GNSS functionality reference position is set using last received airborne aircraft.

ADS-B MAVLink protocol

The device can be switched to use MAVLink protocol. This can be achieved by altering ADSB_RX_PROTO- COL_DECODED setting. When MAVLink protocol is used, module is sending list of aircraft's every second. MAVLink messages have standardized format, which is well described on official protocol webpage (here).

ADS-B Aircraft message

Aircrafts are encoded using ADSB_VEHICLE message (ADSB_VEHICLE). MAVLink message contains several data fields which are described below.

Table 18: MAVLink ADSB_VEHICLE message description.

Field Name	Type	Description
ICAO_address	uint32_t	ICAO address
lat	int32_t	Latitude, expressed as degrees * 1E7
lon	int32_t	Longitude, expressed as degrees * 1E7
altitude	int32_t	Barometric/Geometric Altitude (ASL), in millimeters
heading	uint16_t	Course over ground in centidegrees
hor_velocity	uint16_t	The horizontal velocity in centimeters/second
ver_velocity	uint16_t	The vertical velocity in centimeters/second, positive is up
flags	uint16_t	Flags to indicate various statuses including valid data fields
squawk	uint16_t	Squawk code
altitude_type	uint8_t	Type from ADSB_ALTITUDE_TYPE enum
callsign	char[9]	The callsign, 8 chars + NULL
emitter_type	uint8_t	Type from ADSB_EMITTER_TYPE enum
tslc	uint8_t	Time since last communication in seconds

ADS-B ASTERIX protocol

The device can be switched to use ASTERIX binary protocol. This can be achieved by altering ADSB_RX_PROTOCOL_DECODED setting. When ASTERIX protocol is used, module is sending list of aircrafts every second. Aircrafts are encoded using I021 ver. 2.1 message. Also, once per second the device sends a heartbeat message using I023 ver. 1.2 format in Ground Station Status variant. When running Transceiver TR-1F with ASTERIX, ASTERIX_SIC and ASTERIX_SAC settings are available.

For further reference of parsing ASTERIX frames, please see relevant official documentation:

- I021 messages: CAT021 - EUROCONTROL Specification for Surveillance Data Exchange Part 12: Category 21
- I023 messages: CAT023 - EUROCONTROL Specification for Surveillance Data Exchange Part 16: Category 23

ADS-B GDL90 protocol

The device can be configured to use GDL90 binary protocol. This can be achieved by altering ADSB_RX_PROTOCOL_DECODED setting. When GDL90 protocol is used, module is sending list of aircrafts every second. Aircrafts are encoded using Traffic Report (#20) message. Also, once per second device sends Heartbeat (#0), Ownship Report (#10) and Ownship Geometric Altitude (#11) messages.

For further reference of parsing GDL90 frames see relevant documentation: GDL90 Data Interface Specification

The ADS-B vehicle may transmit barometric, as well as geometric altitude.The ADSB_RX_PROTOCOL setting allows for toggling Traffic Report altitude transmit priority:

- When set to 0, altitude field will be filled with geometric altitude first. If not available, barometric altitude will be used.
- When set to 1, barometric altitude wil be preferred.

Note:

Currently, only ADS-B aircrafts are reported via this protocol. To obtain information about aircrafts reported from FLARM hardware, please use any other supported protocol.

ADS-B Decoded JSON protocol

The “adsb” section contains aircraft information determined by OEM TT-Multi-RF internal ADS-B processing engine. The messages are encoded as JSON array with at least one entry. Each entry is an object consisting of fields denoted in table *adsb* (page 28).. Reports for each ADS-B aircraft are updated once every second.


```json
{
    "src": "33-0000683",
    "ts": 69061337,
    "ver": 1,
    "adsb": [
        {
            "icao": "780A3F",
            "flags": {
                "groundState": false,
                "updAltBaro": true,
                "updAltGeo": true,
                "updPosition": true,
                "updTrack": true,
                "updVeloH": true,
                "updVeloV": true
            },
            "sigStr": -67,
            "sigQ": 9,
            "lat": 34.39696,
            "lon": -85.1055,
            "altBaro": 35000,
            "geoAlt": 36975,
            "track": 143.78,
            "velH": 528,
            "velV": 0,
            "mag_heading": 123.1,
            "true_heading": 125.5,
            "ias": 100,
            "tas": 100,
            "roll": 2.1,
            "nav_qnh": 1013.59,
            "nav_altitude_mcp": 35008,
            "nav_altitude_fms": 35008,
            "nav_modes": {
                "althold": false,
                "approach": false,
                "autopilot": false,
                "lnav": false,
                "tcas": true,
                "vnav": false
            },
            "nav_heading": 151.17,
            "call": "CPA3174",
            "ecat": 5,
            "squawk": "5730",
            "nacp": 9,
            "nacv": 1,
            "nicBaro": 1,
            "nic": 8
        }
```


(continues on next page)

(continued from previous page)

```
    ]
}
```

Table 19: Descriptions of JSON ADS-B section fields.

JSON Field	Unit	Example	Description
src	–	ID-0000001	See table *Description of main JSON fields.* (page 20).
ts	milliseconds	69061337	See table *Description of main JSON fields.* (page 20).
ver	–	1	See table *Description of main JSON fields.* (page 20).
adsb	–	type of message	
icao	–	DABABE	ICAO address, 24-bit value encoded in uppercase hexadecimal, with leading zeros.
flags	–	type of message	
groundState	bool	True	
updPosition	bool	True	
updTrack	bool	True	
updVeloH	bool	True	
updVeloV	bool	True	
updAltGeo	bool	True	
sigStr	dBm	-95	Signal strength, in dBm.
sigQ	dB	2	Signal quality, in dB.
lat	–	53.42854	Latitude. Omitted if position is unknown.
lon	–	14.55281	Longitude. Omitted if position is unknown.
altBaro	ft	1725	Barometric altitude, in feet. Omitted if unknown.
geoAlt	ft	1712	Geometric altitude, in feet. Omitted if unknown.
track	degree °	72.18	Track angle, 0°..360°. Omitted if unknown.
velH	knots	10.5	Horizontal velocity, in knots. Omitted if unknown.
velV	ft/min	50	Vertical velocity, in ft/min, positive value is upwards. Omitted if unknown.
mag_heading	degree °	123.1	Magnetic Heading.
true_heading	degree °	125.5	True Heading.
ias	knots	100	Indicated airspeed.
tas	knots	100	True airspeed.
roll	degree °	2.1	Aircraft roll angle.
nav_qnh	hPa	1013.59	Aviation “Q” Code for “Nautical Height”
nav_altitude_mcp	ft	35008	Refence altitude manually entered into the MCP/FCU
nav_altitude_fms	ft	35008	Altitude selected by the Flight Management System
nav_modes	–	type of message	
althold	bool	False	
approach	bool	False	
autopilot	bool	False	
lnav	bool	False	
tcas	bool	False	
vnav	bool	False	

continues on next page

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

Table 19 – continued from previous page

JSON Field	Unit	Example	Description
nav_heading	degree °	43.1	Heading selected by the Flight Management System
call	–	TEST8	Callsign, up to 8 chars. Omitted if unknown.
ecat	–	13	Emitter category code, see table *ecat* (page 29).. Omitted if unknown.
squawk	–	7232	Squawk, 8 octal digits. Omitted if unknown.
nacp	–	3	NAC_p value, as described in ED-102A. Omitted if value is 0 (unknown).
nacv	–	1	NAC_v value, as described in ED-102A. Omitted if value is 0 (unknown).
nicBaro	–	1	NIC_BARO value, as described in ED-102A. Omitted if value is 0 (unknown).
nic	–	2	NIC value, as described in ED-102A. Omitted if value is 0 (unknown).

The emitter category values returned in *ecat* field is shown in table below:

Table 20: ADS-B emitter category values in JSON protocol.

"ecat" value	Description
0	Unknown.
1	Light (below 15500 lbs.).
2	Small (15500 - 75000 lbs.).
3	Large (75000 - 300000 lbs.).
4	High-Vortex Large (aircraft such as B-757).
5	Heavy (above 300000 lbs.).
6	High performance (above 5g acceleration and above 400 knots).
7	Rotorcraft.
8	Reserved.
9	Glider, Sailplane.
10	Lighter-Than-Air.
11	Parachutist, Skydiver.
12	Ultralight, hang-glider, paraglider.
13	Reserved.
14	Unmanned Aerial Vehicle.
15	Space, Trans-atmospheric Vehicle.
16	Reserved.
17	Surface Vehicle - Emergency Vehicle.
18	Surface Vehicle - Service Vehicle.
19	Point Obstacle (includes Tethered Balloons).
20	Cluster obstacle.
21	Line obstacle.

7.2.2 ADS-B raw protocols

ADS-B HEX protocol

This protocol is dedicated for raw Mode-A/C/S frames acquisition. In this special mode of operation, output frames are not processed, nor validated in any way. All processing, checksum validation, etc. must be done on user's side. All raw frames, regardless of type, start with '*' and end with ';' ASCII characters, whereas their content is encoded in hexadecimal format, MSB first. At the end, extended fields are appended to frame.

```
*RAW_FRAME;(SIGS,SIGQ,TS1s,TS24h)\r\n
```

Table 21: Descriptions of RAW extended messages.

Var.	Description	Example
SIGS	Signal strength in dBm	-95
SIGQ	Signal quality in dB	2
TS1s	Timestamp for multilateration. Time from last PPS pulse, hex format, in nanoseconds.	75BCD15 (0.123456789s)
TS24h	Timestamp for multilateration. Time from midnight, hex format, in nanoseconds.	2B5792B49315 (47655.123456789s = 13:14:15.123456789)

Note:

To use multilateration, TS value must be calibrated using calibration value from statistics message.

Note:

TS field is available when precise PPS signal from GNSS source is applied to module to 1PPS pin.

Mode-S raw frames

Short and long frames consist accordingly of 7 or 14 data bytes. Examples of raw MODE-S frames:

- Short frame: `*5D4B18FFFC710B;(-70,3,75BCD15, 2B5792B49315)\r\n`
- Long frame: `*8D4CA7E858B9838206BA422BBD7B;(-71,4,75BCD15, 2B5792B49315)\r\n`

Mode-AC raw frames

Note:

It is impossible to reliably distinguish between MODE-A and MODE-C frames based only on received signal on 1090MHz.

Starting with firmware 2.7.0, each frame is interpreted as squawk and formatted as 4 octal digits. They can also be read as binary frame with 4 hexadecimal digits, with bits being set as shown in table below.

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

Table 22: Description of bits in raw Mode-A/C frames in new protocol version.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	A4	A2	A1		B4	B2	B1		C4	C2	C1		D4	D2	D1

Examples of raw MODE-A/C frames using this format are as follows:

- `*0363;(979,151,75BCD15, 2B5792B49315)\r\n`
- `*7700;(995,167,75BCD15, 2B5792B49315)\r\n`

ADS-B HEXd protocol

Important:

This is RAW HEX protocol standardized for dump1090, without additional fields after `;`.

ADS-B Beast protocol

Original specification: documentation

Format

All data is escaped: 0x1a -> 0x1a 0x1a. Note that synchronization is still complex, since 0x1a 0x31 may be the start of a frame or mid-data, depending on what preceded it. To synchronize, you must see, in order:

- != 0x1a
- 0x1a
- 0x31, 0x32, 0x33

Escaping makes frame length for a given type variable, up to 2 + (2 * data_length_sum)

Frame structure

- 0x1a
- 1 byte frame type (see types below)
- 6 byte MLAT timestamp (see below)

Frame types

- 0x31: Mode-AC frame
  - 1 byte RSSI
  - 2 byte Mode-AC data
- 0x32: Mode-S short frame
  - 1 byte RSSI
  - 7 byte Mode-S short data
- 0x33: Mode-S long frame
  - 1 byte RSSI
  - 14 byte Mode-S long data

MLAT timestamp

The MLAT timestamp included in each frame is the big-endian value of a 12 MHz counter at the time of packet reception. This counter isn't calibrated to external time, but receiving software can calculate its offset from other receiving stations across multiple packets, and then use the differences between station receive timing to calculate signal source position.

FlightAware's dump1090 fork sends 0x00 0x00 0x00 0x00 0x00 0x00 when it has no MLAT data.

RSSI

FlightAware's dump1090 fork sends 0xff when it has no RSSI data.

Examples

- 0x1a 0x32 0x08 0x3e 0x27 0xb6 0xcb 0x6a 0x1a 0x1a 0x00 0xa1 0x84 0x1a 0x1a 0xc3 0xb3 0x1d
  - 0x1a: Frame start
  - 0x32: Mode-S short frame
  - 0x08 0x3e 0x27 0xb6 0xcb 0x6a: MLAT counter value
    - Decimal: 9063047285610
  - 0x1a 0x1a: Signal level
    - Unescaped: 0x1a
    - Decimal: 26
    - 26 / 255 * 100
  - 0x00 0xa1 0x84 0x1a 0x1a 0xc3 0xb3 0x1d: Mode-S short data
    - Unescaped: 0x00 0xa1 0x84 0x1a 0xc3 0xb3 0x1d

ADS-B raw JSON protocol

The "raw" section contains raw, unprocessed and unfiltered ADS-B frames gathered by OEM TT-Multi-RF , which can be used e.g. for multilateration and other low-level analysis. Raw messages are encoded as JSON array with at least one entry. Each array entry is a separate array containing values as described below

```
{
    "src": "ID-0000001",
    "ts": 69061337,
    "ver": 1,
    "raw": [
        [
            "18A9725A4C842D",
            -78,
            2,
            "295CAB573A77"
        ]
    ]
}
```

Table 23: Descriptions of JSON ADS-B Raw section fields.

JSON Field	Unit	Example	Description
src	–	ID-0000001	See table *Description of main JSON fields.* (page 20).
ts	milliseconds	69061337	See table *Description of main JSON fields.* (page 20).
ver	–	1	See table *Description of main JSON fields.* (page 20).
raw	–	type of message	
	hexadecimal	18A9725A4C842D	Raw frame bytes, formatted as uppercase hexadecimal. Short Mode-S frames encode 7 bytes, long frames contain 14 bytes.
	dBm	-78	Signal strength, in dBm.
	dB	2	Signal quality, in dB.
	nanoseconds	295CAB573A77	UTC-calibrated time of reception, formatted as uppercase hexadecimal, in nanoseconds. Example translates to 12:37:57.988350583

Warning:

Due to constrained throughput of device communication, transmission of some raw frames may be skipped in heavy aircraft traffic situations.

7.2.3 ADS-B statstics protocols

ADS-B CSV statistic protocol

This message contains some useful statistics about operation of module. Format of that frame is shown below:

`#AS:FPSS,FPSAC,CALIB,CRC\r\n`

FPSS - All received mode S frames per second

FPSAC - All received mode A/C frames per second

CALIB - Real uC frequency based on GNSS module (PPS)

CRC - Value is calculated using standard CRC16 algorithm

8 FLARM receiver or transceiver subsystem

Important:

This part of documentation is relevant only for devices which have **FLARM IN/OUT** functionality

Attention:

The DRS-1 or MP1 devices are receivers only. Despite possible availability of FLARM out settings, some products such as the MP1, GS2L and DRS-1 do not support FLARM out. Any settings will not affect the transmit system, it is recommended to set all transmit settings to 0.

8.1 FLARM ID calculation

Aerobits device equipped with FLARM out has a unique FLARM ID and can be calculated using the formula:

$$FLARM\ ID = Device_{SN} + Offset$$

where the offsets for Aerobits devices are as follows:

Device	Number
TT-SF1	1024
TT-SF2	2048
TR-1F	4046
DRS-1F	5096
TR-2F	6096
trkME	8096
TT-SF2n	10096
trkME PRO	12096

Note:

Offset displayed by device is in hexadecimal.

8.2 Settings

Table 24: Descriptions of FLARM settings.

Setting	Min	Max	Def	Comment
FLARM_RX_PROTOCOL_DECODED	–	–	CSV	FLARM decoded protocol: None CSV Mavlink JSON ASTERIX

continues on next page

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

Table 24 – continued from previous page

Setting	Min	Max	Def	Comment
FLARM_STATISTICS	–	–	CSV	FLARM statistics protocol: None CSV JSON
FLARM_TX	0	1	1	Enable FLARM out: 0 - disable 1 – enable
FLARM_TX_AIRCRAFT_TYPE	0	15	13	Flarm aircraft type: 0 – UNKNOWN 1 – GLIDER 2 – TOWPLANE 3 – HELICOPTER 4 – PARACHUTE 5 – DROPPLANE 6 – FIXED_HG 7 – SOFT_HG 8 – ENGINE 9 – JET 10 – RESERVED 11 – BALLOON 12 – AIRSHIP 13 – UAV 15 - STATIC

8.3 Protocols

8.3.1 FLARM decoded protocols

FLARM CSV protocol

This message describes state vector of aircraft received through FLARM radio and is sent once per second.

```
#ALRM:TYPE,ID,ID_TYPE,AIRCRAFT_TYPE,ALARM_LVL,LAT,LON,ALT,TRACK,VELH,VELV,MOVE_MODE,
REL_N,REL_E,R_DIST_H,REL_DIST_V,NEAR_DIST,DIR,STEALTH,NOTRACK\r\n
```

Table 25: Descriptions of FLARM fields.

#ALRM	FLARM Aircraft message start indicator	Example value
TYPE	Target type. 0: stationary, 2: regular aircraft.	0
ID	Id value, in hexadecimal format.	1600BF
ID_TYPE	Id type: 0: randomized id value, 1: ICAO, 2: FLARM id	2
AIRCRAFT_TYPE	Target type *Descriptions of FLARM aircraft types field.* (page 37)	13
ALARM_LVL	Alarm threat level (0-3). 0: no danger, 3: high danger.	0
LAT	Latitude, in 1−7 degrees.	535668736
LON	Longitude, in 1−7 degrees.	163101952
ALT	Altitude, in meters.	61
TRACK	Track angle, in degrees.	90
VELH	Ground speed, in m/s.	0
VELV	Climbing rate, in m/s.	20

continues on next page

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

Table 25 – continued from previous page

#ALRM	FLARM Aircraft message start indicator	Example value
MOVE_MODE	Movement mode. 1: Stationary (not flying), 4: circling right, 5: cruising, 7: circling left.	1
REL_N	Distance to target on South-North axis, in meters.	2
REL_E	Distance to target on West-East axis, in meters.	-3
REL_DIST_H	Relative horizontal distance, in meters.	3
REL_DIST_V	Relative vertical separation, in meters. Value is positive if target is on higher altitude.	8
NEAR_DIST	Target proxy distance, for priority sorting in NEAREST mode.	9
DIR	Relative bearing in degrees from the own position and true ground track to the target's position. Value ranges from -180 to 180, positive values are clockwise.	-56
STEALTH	Set to 1 if target has stealth (privacy) flag set, otherwise 0.	0
NOTRACK	Set to 1 if target has notrack flag set, otherwise 0.	0

Table 26: Descriptions of FLARM aircraft types field.

Aircraft type index	Description
0	Reserved.
1	Glider, Motor glider.
2	Tow plane, tug plane.
3	Helicopter, gyrocopter, rotocraft.
4	Skydiver, parachute.
5	Drop plane for skydivers.
6	Hang glider (hard).
7	Hang glider (soft).
8	Aircraft with reciprocating engine.
9	Aircraft with jet / turboprop engine.
10	Reserved.
11	Balloon (hot, gas, weather, static).
12	Airship, blimp, zeppelin.
13	Unmanned Aerial Vehicle (UAV).
14	Reserved.
15	Static obstacle.

FLARM MAVLink protocol

Aircrafts reported by FLARM use ADSB_VEHICLE message in same format as described in *MAVLink ADSB_VEHICLE message description.* (page 26) section, with following restrictions:

- The FLARM "Aircraft Type" field is translated to MAVLink "Emitter Category" field as shown in table below.
- ICAO field contains FLARM id value.

Table 27: FLARM Aircraft Type to Emitter Category translation.

Aircraft Type Index	Aircraft Type description	Emitter Category Index	Emitter Category description
0	Reserved	0	No information
1	Glider, Motor glider	9	Glider

continues on next page

Table 27 – continued from previous page

Aircraft Type Index	Aircraft Type description	Emitter Category Index	Emitter Category description
2	Tow plane, tug plane	1	Light
3	Helicopter, gyrocopter, rotocraft	7	Rotorcraft
4	Skydiver, parachute	11	Parachute
5	Drop plane for skydivers	1	Light
6	Hang glider (hard)	12	Ultra light
7	Hang glider (soft)	12	Ultra light
8	Aircraft with reciprocating engine	1	Light
9	Aircraft with jet / turboprop engine	3	Large
10	Reserved	0	No information
11	Balloon (hot, gas, weather, static)	10	Lighter than air
12	Airship, blimp, zeppelin.	10	Lighter than air
13	Unmanned Aerial Vehicle (UAV)	14	UAV
14	Reserved	0	No information
15	Static obstacle	0	No information

FLARM Collision message

Apart from ADS-B messages, FLARM subsystem emits COLLISION messages (Mavlink documentation). Detailed information about given aircraft can be obtained from ADSB_VEHICLE message directly preceding given COLLISION message.

FLARM ASTERIX protocol

All aircrafts detected by FLARM hardware are reported in same way as ADS-B vehicles, with following restrictions:

- FLARM messages are using SIC = 161, SAC = 0 values. This is the preferred way to distinguish FLARM messages from ADS-B.
- The I021/040 (Target Report Descriptor) field has ATP subfield set to 3 if aircraft id is not ICAO-based (e.g. FLARM id, random id).
- The I021/210 (MOPS Version) field has VNS subfield set to 1.
- The I021/170 (Target Identification) is filled with `STEALTH` value if FLARM “stealth” flag is set, or `NOTRACK` value if “notrack” flag is set.
- The I021/020 Emitter Category value is determined from FLARM “Aircraft Type” field as shown below.

Table 28: FLARM Aircraft Type to ASTERIX Emitter Category translation.

Aircraft Type Index	Aircraft Type description	Emitter Category Index	Emitter Category description
0	Reserved	0	No information
1	Glider, Motor glider	9	Glider
2	Tow plane, tug plane	1	Light
3	Helicopter, gyrocopter, rotocraft	7	Rotorcraft
4	Skydiver, parachute	11	Parachute
5	Drop plane for skydivers	1	Light

continues on next page

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

Table 28 – continued from previous page

Aircraft Type Index	Aircraft Type description	Emitter Category Index	Emitter Category description
6	Hang glider (hard)	12	Ultra light
7	Hang glider (soft)	12	Ultra light
8	Aircraft with reciprocating engine	1	Light
9	Aircraft with jet / turboprop engine	3	Large
10	Reserved	0	No information
11	Balloon (hot, gas, weather, static)	10	Lighter than air
12	Airship, blimp, zeppelin.	10	Lighter than air
13	Unmanned Aerial Vehicle (UAV)	14	UAV
14	Reserved	0	No information
15	Static obstacle	0	No information

FLARM JSON protocol

The "flarm" section contains aircraft information determined by OEM TT-Multi-RF internal FLARM processing engine. The messages are encoded as JSON array with at least one entry. Each entry is an object consisting of fields denoted in table *FLARM* (page 39).. Reports for each FLARM aircraft are updated once every second.


```
{
    "src": "ID-0000001",
    "ts": 69061337,
    "ver": 1,
    "flarm": [
        {
            "idType": 1,
            "id": "DABABE",
            "type": 13,
            "danger": 1,
            "lat": 53.42854,
            "lon": 14.55281,
            "alt": 1725,
            "track": 72.18,
            "hVelo": 10.5,
            "vVelo": 50,
            "movMode": 5,
            "stealth": 1,
            "notrack": 1

        }
    ]
}
```


Table 29: Descriptions of JSON FLARM section fields.

JSON Field	Unit	Example	Description
src	–	ID-0000001	See table *Description of main JSON fields.* (page 20).
ts	milliseconds	69061337	See table *Description of main JSON fields.* (page 20).
ver	–	1	See table *Description of main JSON fields.* (page 20).
flarm	–	type of message	

continues on next page

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

Table 29 – continued from previous page

JSON Field	Unit	Example	Description
idType	–	1	Aircraft id type. 0: randomized, 1: ICAO, 2: FLARM.
id	–	DABABE	Aircraft id, 32-bit value encoded in uppercase hexadecimal, with leading zeros.
type	–	13	Aircraft type, see table *FLARM aircraft type category values in JSON protocol.* (page 40).
fps	fps	2	Number of raw Mode-S frames received from aircraft during last second.
lat	–	53.42854	Latitude. Omitted if position is unknown.
lon	–	14.55281	Longitude. Omitted if position is unknown.
alt	m	1725	Barometric altitude, in meters.
track	degree °	72.18	Track angle, 0°..360°. Omitted if unknown.
hVelo	m/s	10.5	Horizontal velocity, in m/s. Omitted if unknown.
vVelo	m/s	50	Vertical velocity, in m/s., positive value is upwards. Omitted if unknown.
movode	–	5	Movement mode.1: stationary, 4: circling right, 5: flying,7: circling left.
stealth	–	1	Set to 1 if target has Stealth flag set, otherwise omitted.
notrack	–	1	Set to 1 if target has Notrack flag set, otherwise omitted.

The list of possible FLARM "Aircraft type" values returned in *type* field is shown in table *ECAT-FLARM* (page 40).

Table 30: FLARM aircraft type category values in JSON protocol.

"ecat" value	Description
0	Reserved.
1	Glider, Motor glider.
2	Tow plane, tug plane.
3	Helicopter, gyrocopter, rotocraft.
4	Skydiver, parachute.
5	Drop plane for skydivers.
6	Hang glider (hard).
7	Hang glider (soft).
8	Aircraft with reciprocating engine.
9	Aircraft with jet / turboprop engine.
10	Reserved.
11	Balloon (hot, gas, weather, static).
12	Airship, blimp, zeppelin.
13	Unmanned Aerial Vehicle (UAV).
14	Reserved.
15	Static obstacle.

8.3.2 FLARM statstics protocols

FLARM CSV statistic protocol

This message contains some useful statistics about operation of module. Format of that frame is shown below:

```
#FS:FPS,VFR,ERD,ERI,ERW,ERR,FTX\r\n
```

FPS - All received frames per second

VFR - All valid received frames per second

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

ERD - For developer purpose only

ERI - For developer purpose only

ERW - For developer purpose only

ERR - For developer purpose only

FTX - All sent frames per second

FLARM JSON statistic protocol

Format of that frame is shown below:

```
{"ver":1,"src":"32-0000009","flarm_statistics":[{"errorDebug":ERD,"errorInfo":ERI,
"errorWarning":ERW,"errorReal":ERR,"frameReceived":VFR,"frameReceivedAll":FPS,
"frameSent":FTX}]}\r\n
```

FPS - All received frames per second

VFR - All valid received frames per second

ERD - For developer purpose only

ERI - For developer purpose only

ERW - For developer purpose only

ERR - For developer purpose only

FTX - All sent frames per second

9 GNSS receiver subsystem

9.1 Settings

Table 31: Descriptions of GNSS settings

Setting	Min	Max	Def	Comment
GNSS_RX_PROTOCOL_RAW	NONE	NMEA	NMEA	GNSS_RX RAW protocol select NONE NMEA
GNSS_RX_PROTOCOL_DECODED	NONE	JSON	NONE	GNSS_RX Decoded protocol select NONE JSON

9.2 Protocols

9.2.1 GNSS NMEA RAW protocol

Note:

For more information about all NMEA GNSS fields go to docs.

9.2.2 GNSS JSON protocol

The *gnss* section contains basic GNSS information. This message is sent once per second. The example JSON message with "gnss" section fields described:


```
{
    "src": "ID-0000001",
    "ts": 69061337,
    "ver": 1,
    "gnss": {
        "fix": 1,
        "lat": 53.42854,
        "lon": 14.55281,
        "altWgs84": 499.6,
        "altMsl": 508.6,
        "track": 127.3,
        "hVelo": 10.5,
        "vVelo": 25,
        "gndSpeed": [
            5.2,
            2.1
        ],
        "acc": {
            "lat": 5.2,
            "lon": 2.1,
            "alt": 3.6
        },
        "nacp": 12,
        "nacv": 2,
```


(continues on next page)

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

(continued from previous page)

```
        "nic": 12
    }
}
```

Table 32: Descriptions of JSON GNSS section fields.

JSON Field	Unit	Example	Description
gnss			Type of message
fix	–	1	Set to 1 if onboard GNSS currently has fix, otherwise 0.
lat	–	53.42854	Last known latitude. Omitted if there was no GNSS fix since device boot.
lon	–	14.55281	Last known longitude. Omitted if there was no GNSS fix since device boot.
altWgs84	–	499.6	Last known WGS-84 Altitude, in meters. Omitted if there was no GNSS fix since device boot.
altMsl	–	508.6	Last known MSL Altitude, in meters. Omitted if there was no GNSS fix since device boot.
track	–	127.3	Track angle, 0°..360°, relative to true north. Omitted if unknown.
hVelo	–	10.5	Horizontal velocity, in knots. Omitted if unknown.
vVelo	–	25	Vertical velocity, in m/s. Positive value is upwards. Omitted if unknown.
gndSpeed	knots	[5.2,2.1]	Ground speed in east-west and north-south axes respectively, in knots. Positive value is East and North. Derived from track / hVelo values. Omitted if unknown.
acc	m/s2	struct	Acceleration in all 3 dimensions
lat	–	5.2	Accuracy of latitude, in meters. Omitted if unknown.
lon	–	2.1	Accuracy of longitude, in meters. Omitted if unknown.
alt	–	3.6	Accuracy of altitude, in meters. Omitted if unknown.
nacp	–	12	Navigational Accuracy Category for Position value, as defined in ED-282. Omitted if unknown.
nacv	–	2	Navigational Accuracy Category for Velocity value, as defined in ED-282. Omitted if unknown.
nic	–	12	Navigation Integrity Category as defined in ED-282. Omitted if unknown.

10 RemoteID transmitter subsystem

Aerobits devices with **RemoteID OUT** functionality broadcast UAV data using Bluetooth and/or WiFi interface. These data can be captured by any device with RemoteID IN functionality which supports ASTM/ASD-STAN standard including modern smartphones with proper mobile applications.

10.1 Settings

The following settings allow users to control the broadcasting feature and configure specific Remote ID data.

Table 33: Descriptions of RemoteID settings.

Setting	Min	Max	Def	Comment
DRONE_ID_ADVERTISING_ENABLE	0	1	1	Advertising enable
DRONE_ID_BASIC_BROADCAST_PERIOD	200	3000	1500	Basic frame broadcast period in [ms]
DRONE_ID_BROADCAST_BLUETOOTH_4	0	1	1	Enable Bluetooth 4.0 broadcast
DRONE_ID_BROADCAST_BLUETOOTH_5	0	1	1	Enable Bluetooth 5.0 broadcast
DRONE_ID_BROADCAST_WIFI_BEACON	0	1	1	Enable Wifi Standard Beacon broadcast
DRONE_ID_BROADCAST_WIFI_NAN_BEACON	0	1	1	Enable WiFi NaN Beacon broadcast
DRONE_ID_DRONE_CATEGORY_CLASS	0	7	0	Drone category class: 0 – None 1 – C0 2 – C1 3 – C2 4 – C3 5 – C4 6 – C5 7 – C6
DRONE_ID_HEIGHT_TYPE	0	1	0	Height type: 0 – Relative to take-off location 1 – Relative to ground
DRONE_ID_LOCALIZATION_BROADCAST_PERIOD	100	1000	500	Localization frame broadcast period in [ms]
DRONE_ID_MAVLINK_CONNECTION_TIMEOUT	2	30	5	Mavlink timeout in [s]
DRONE_ID_MODE	0	1	1	Determines Mavlink reception: 0 - Full mavlink support 1 - Ignore all mavlink messages 2 - Ignore only location messages
DRONE_ID_OPERATIONAL_STATUS	0	2	0	Operational status: 0 – Undeclared 1 – Ground 2 – Airborne

continues on next page

Table 33 – continued from previous page

Setting	Min	Max	Def	Comment
DRONE_ID_OPERATION_CATEGORY	0	3	0	Operation category: 0 – None 1 – Open 2 – Specific 3 - Certified
DRONE_ID_OPERATOR_ID	–	–	–	Operator message payload
DRONE_ID_OPERATOR_ID_TYPE	0	255	0	Operator ID type: 0 - Operator ID 1 – 200 Reserved 201 – 255 Available for private use
DRONE_ID_SELF_ID	–	–	–	Self message payload
DRONE_ID_SELF_ID_TYPE	0	255	0	Self ID type: 0 - Text description 1 – 200 Reserved 201 – 255 Available for private use
DRONE_ID_TYPE	0	3	0	UAS ID type: 0 – None 1 - Serial Number 2 - CAA Assigned Registration ID 3 - UTM Assigned UUID
DRONE_ID_UAS_TYPE	0	15	0	Specification of the type of UAS: 0 – None 1 – Aeroplane 2 - Helicopter or Multirotor 3 – Gyroplane 4 - Hybrid Lif 5 - Ornithopter 6 – Glider 7 – Kite 8 - Free Balloon 9 - Captive Balloon 10 – Airship 11 - Free Fall 12 – Rocket 13 - Tethered Powered Aircraft 14 - Ground Obstacle 15 – Other

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

11 RemoteID receiver subsystem

Aerobits devices with **RemoteID IN** functionality capture Remote ID using Bluetooth and WiFi interfaces, following ASTM/ASD-STAN standard.

Important:

This part of documentation is relevant only for devices which have **RemoteID IN** functionality

Note:

Devices trkME+ and trkME PRO are not released yet, documentation will be updated

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

12 Network communication system

12.1 Network communication modes MQTT

The trkME communicates through the **LTE** network using **MQTT** 3.1 protocol. Connection can be configured to use username and password authentication, as well as **TLS** encryption. All data are transmitted into specific **MQTT** topic.

12.2 Settings

12.2.1 LTE

Table 34: Descriptions of LTE settings.

Setting	Min	Max	Def	Comment
APN_NAME	–	–	–	LTE APN name
APN_PASSWORD	–	–	–	LTE APN user password
APN_TYPE	0	5	0	Type of APN: 0 – auto 1 – GSM 2 – CDMA 3 – WCDMA 4 – EVDO 5 – LTE
APN_USER	–	–	–	LTE APN user name
LTE_LOG	0	1	0	Show LTE log: 0 – disable 1 – enable

12.2.2 MQTT

Table 35: Descriptions of MQTT settings.

Setting	Min	Max	Def	Comment
MQTT_BROKER_ADDRESS	–	–	–	MQTT broker address (IP, URL)
MQTT_BROKER_PORT	–	–	–	MQTT broker port (0-65535)
MQTT_CLIENT_ID	–	–	–	MQTT client ID, leave blank to use '{user}_{imsi}_{rand}'
MQTT_KEEPALIVE	0	3600	0	MQTT keepalive interval in seconds
MQTT_LOG	0	1	0	Debug log of MQTT module: 0 – disable 1 – enable
MQTT_PASS	–	–	–	MQTT broker password
MQTT_TLS	0	1	0	Enables TLS in MQTT: 0 – disable 1 – enable
MQTT_USER	–	–	–	MQTT broker user name

continues on next page

Table 35 – continued from previous page

Setting	Min	Max	Def	Comment
MQTT_WEBSOCKET	0	1	0	Use websocket when connecting to broker: 0 – disable 1 – enable
REMOTE_ID_TOPIC	–	–	–	OpenDroneID MQTT topic
RID_RX_PROTOCOL_DECODED	–	–	CSV	Remote ID decoded protocol: None CSV JSON

For devices **TrkMe+ and TrkMe PRO** available are additional settings:

Table 36: Descriptions of MQTT settings.

Setting	Min	Max	Def	Comment
ADSB_PROTOCOL_MQTT	–	–	JSON	Format of mqtt messages: JSON ASTERIX
ADSB_TOPIC	–	–	–	ADS-B plot MQTT topic
AUX_DATA_TOPIC	–	–	–	Topic for transmitting data received by aux port
CSV_TOPIC	–	–	–	CSV plot MQTT topic
FLARM_TOPIC	–	–	–	FLARM plot MQTT topic
MLAT_TOPIC	–	–	–	MLAT plot MQTT topic
TELEMETRY_TOPIC	–	–	–	JSON Telemetry plot MQTT topic

12.3 Protocols

Remote ID data transmitted to REMOTE_ID_TOPIC are in JSON format described below. Example of JSON data:

```
{
    "remoteid": {
        "uasid": "18099440000001",
        "id_type": 1,
        "uas_type": 0,
        "uas_latitude": 53.39689,
        "uas_longitude": 14.628,
        "height": 0.06,
        "altitudeBaro": -15.04,
        "altitudeGeo": 1775.59,
        "direction": 0,
```

(continues on next page)

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

(continued from previous page)


```
        "speedH": 0,
        "speedV": 0.2,
        "status": 0,
        "operator_id": "POL7f4b6c87bbd5w",
        "operator_id_type": 0,
        "operator_latitude": 53.38649,
        "operator_longitude": 14.63821,
        "operator_location_type": 0,
        "timestamp": 83.3,
        "self_id_type": 0,
        "self_id": "TEST1",
        "pressure": 1015.06,
        "temperature": 47.88
    }
}
```


Table 37: Description of JSON fields.

JSON Field	Unit	Example	Description
uasid	–	18099440000001	string value representing UAV ID
id_type	–	1	Type of ID
uas_type	–	0	Type of UAV
uas_latitude	degrees	53.39689	Latitude of UAV position in degrees
uas_longitude	degrees	14.62835	Longitude of UAV position in degrees
height	m	0.06	Height based on start up altitude, in meters
altitudeBaro	m	200	Altitude based on Pressure sensor
altitudeGeo	m	1775.59	Altitude based on GNSS
direction	degrees	120	Direction angle
speedH	m/s	14	Horizontal speed
speedV	m/s	2	Vertical speed
status	–	0	Status of UAV
operator_id	–	POL7f4b6c87bbd5w	The operator number from local FAA department
operator_id_type	–	0	Type of operator id
operator_latitude	degrees	53.39689	The operator latitude in degrees

continues on next page

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

Table 37 – continued from previous page

JSON Field	Unit	Example	Description
operator_longitude	degrees	14.62843	The operator longitude in degrees
operator_location_type	–	0	Type od operator location
timestamp	milliseconds	69061337	Timestamp in milliseconds, relative to last UTC midnight
self_id_type	–	0	Type of Self ID string
self_id	–	TEST1	Self ID string
pressure	hPa	1015.06	Pressure measured by onboard sensor
temperature	Celsius	32.21	Temperature measured by onboard sensor

Note:

The order of JSON object fields in any part of message may vary between firmware revisions and messages.

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

13 Sensors receiver subsystem

13.1 Settings

Table 38: Descriptions of Sensors settings.

Setting	Min	Max	Def	Comment
SENSORS_RX_PROTOCOL_RAW	–	–	None	Sensors decoded protocol: None CSV JSON

13.2 Protocols

13.2.1 Pressure CSV protocol

This message describes state vector of sensor determined from SENSORS messages and is sent once per second. The message format is as follows:

```
#SP:CALIB,PRESS,TEMP,CRC
```

Table 39: Descriptions of SENSORS fields.

#SP	Sensors message start indicator	Example value
CALIB	Pressure sensor calibration value	1
PRESS	Current pressure value	1002.213742
TEMP	Current temperature value	56.420123
CRC	CRC16 (described in CRC section)	2D3E

13.2.2 Sensor JSON protocol

The *sensor* section contains values acquired from miscellaneous sensors present in Aerobits device hardware and consists of fields shown below. This message is sent once per second. All fields are optional - they are sent only if appropriate sensor is enabled.

```
{
    "ver": 1,
    "sensor": {
        "pressure": 1006.87,
        "temp": 39.8
        },
    "HumiditySensor": {
        "Temperature": 36.9,
        "Humidity": 19,
        }
    }
}
```

Table 40: Descriptions of JSON Sensor section fields.

JSON Field	Unit	Example	Description
ver	–	1	See table *Description of main JSON fields.* (page 20).
sensor	–	type of sensor	

continues on next page

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

Table 40 – continued from previous page

JSON Field	Unit	Example	Description
pressure	hPa	1006.87	Current pressure sensor value in hPa.
temp	°C	39.8	Current temprerature sensor value in °C.
HumiditySensor	–	type of sensor	
Temperature	°C	36.9	Current temperature sensor value in °C.
Humidity	%	19	Current humidity sensor value in %.

14 Quick start

This section will describe how to start up with **trkME** devices.

Note:

Details could vary between **trkME**, **trkME+** and **trkME-PRO** devices.

14.1 Dismantling aluminium housing

This step is neccessary for inserting your SIM card inside trkMe device.

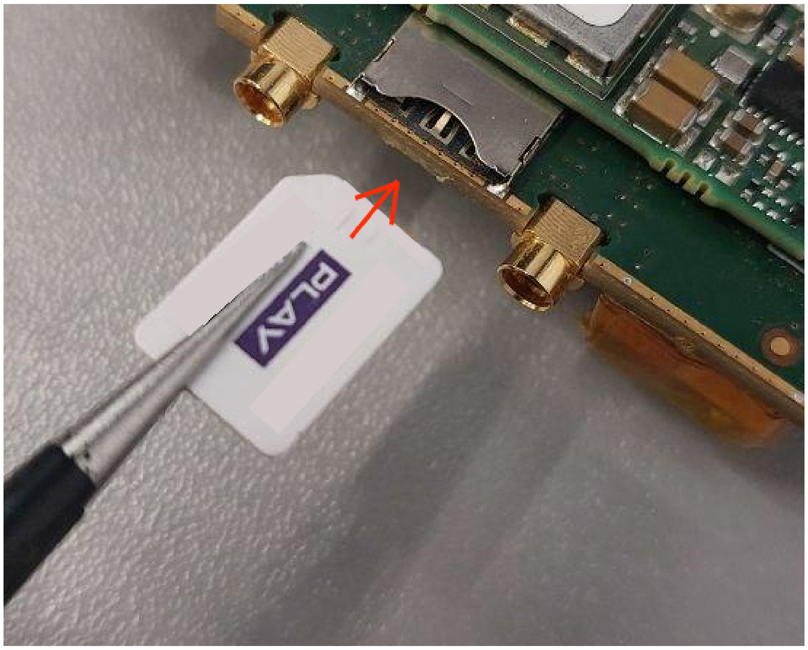
Unscrew four screws and open the aluminium lid.

Gently take out PCB from the housing.

14.2 Inserting SIM card

Insert valid nano SIM card from your local operator.

In case of connectivity issues please check if SIM card is working. You can do that with other LTE modem, for example mobile phone.



14.3 Powering up

Connect USB C cable to **trkMe** device. Use **USB-C Device** connector, like on below picture.

14.4 Setting the correct APN name in device settings

Connect trkME to your PC. Open serial connection and write commands:

```
AT+CONFIG=1
AT+APN_NAME=<apn_name_from_your_lte_provider>
AT+CONFIG=0
```

14.5 Setting mqtt parameters

For collecting data from trkME to your mqtt broker you should use commands:

```
AT+CONFIG=1
AT+REMOTE_ID_TOPIC=<topic_where_remote_data_will_be_Send>
AT+MQTT_BROKER_PORT=1883
AT+MQTT_BROKER_ADDRESS=<address_of_your_broker>
AT+CONFIG=0
```

14.6 Example minimal setup

```
AT+CONFIG=1
AT+APN_NAME=internet
AT+REMOTE_ID_TOPIC=TEST_TOPIC_REMOTE
AT+MQTT_BROKER_PORT=1883
AT+MQTT_BROKER_ADDRESS=test.mosquitto.org
AT+CONFIG=0
```

15 Disclaimer

Information contained in this document is provided solely in connection with Aerobits products. Aerobits reserves the right to make changes, corrections, modifications or improvements to this document, and to products and services described herein at any time, without notice. All Aerobits products are sold pursuant to our own terms and conditions of sale. Buyers are solely responsible for the choice, selection and use of the Aerobits products and services described herein, and Aerobits assumes no liability whatsoever, related to the choice, selection or use of Aerobits products and services described herein. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services, it shall not be deemed a license granted by Aerobits for use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering use, in any manner whatsoever, of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN AEROBITS TERMS AND CONDITIONS OF SALE, AEROBITS DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO USE AND/OR SALE OF AEROBITS PRODUCTS INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OF MERCHAN-TABILITY, FITNESS FOR A PAR- TICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGE- MENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED AEROBITS REPRESENTATIVE, AEROBITS PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE.

Information in this document supersedes and replaces all previously supplied information.

© 2024 Aerobits - All rights reserved