Aerobits

Subsystems for the
UAS integration into
the airspace

TT-RR Remote ID OEM transceiver

Data sheet - User manual

Contents

1 Introduction

The **OEM TT-RR** and **OEM TT-RRn** transceivers are designed to meet requirements of remote drone identification and localization in **ASTM/ASD-STAN** standard. Using the **BLE** broadcast and **WiFi Nan, Beacon** technology the device provides surveillance and drone operator identification capability based on any modern mobile devices such as smartphone or tablet. It is equipped with a high quality **multi-GNSS** receiver and a barometric altitude sensor.

The analysis of the power/quality of the RF signal, and the fast UART interface and easy configuration with AT-commands allows for the simple integration of the module with the user's system. In addition, extra interfaces open the way to customize the firmware and extend the module with non-standard functions.

As Remote Identification becomes a standard practice in airspace management, Aerobits solutions are at the forefront of this technological advancement, promoting safer and more accountable drone operations.

Important:

Each firmware version is documented separately. This document is relevant for firmware version v1.29.3. If your firmware version is different, please find relevant documentation on our website aerobits.pl.

1.1 Available variants

VARIANT	BLE	Wi-Fi	GNSS	Pressure sensor
TT-RRn	•	•		
TT-RR	•	•	•	•

Warning:

OEM TT-RRn without GNSS requires an external position source to send Remote ID frames.

1.2 Features

- **Ability to receive RemoteID frames using BLE and Wi-Fi standards**
- **Capability to work with MAVLINK devices**
- **Broadcast WiFi Nan and Beacon frames**
- **BLE broadcast technology compliant with ASTM and ASD-STAN**
- **Interfaces: UART, USB**
- **Supports Bluetooth 4.0 and 5.2**
- **Free Android application available on Google Play** OpenDroneID OSM
- **Integrated GNSS source and pressure sensor**

For more information please contact support@aerobits.pl.

2 Technical parameters

2.1 Basic technical information

Table 1: General technical parameters

Parameter	Description	Typ.	Unit
First Band	BLE	2400	MHz
Second Band	Wi-Fi	2400	MHz
Third Band	GNSS	1575	MHz
Max. output (BLE)	Maximum output power	+18	dBm
Max. output (Wi-Fi)	Maximum output power	+20	dBm
Sensitivity (GNSS)		-167	dBm
UART	AT commands	921600	bps
USB	AT commands		
MSL	Moisture Sensitivity Level	4	

2.2 Electrical specification

2.2.1 Absolute maximum ratings

Table 2: Absolute maximum ratings.

Parameter	Min	Typical	Max	Unit
Storage temperature	-5	–	+40	°C
Supply voltage (VCC)	3.0	3.3	3.6	DCV
Other pin voltage	-0.3	–	VCC + 0.3	DCV
RF input BLE	–	–	+5	dBm
RF input Wi-Fi	–	–	+5	dBm
RF input GNSS	–	–	0	dBm

2.2.2 Recommended operation conditions

Table 3: Recommended operation conditions.

Parameter	Min	Typ	Max	Unit
Operation temperature	-30	–	+85	°C
Supply voltage (VCC)	3.0	3.3	3.6	DCV

2.2.3 General electrical parameters

Table 4: General electrical parameters.

Parameter	Description	Min	Typ	Max	Unit
Current consumption		–	70	–	mA
Input Low Voltage	RESET, UARTs, CAN, USB, SPI, I2C	-0.3	–	0.8	DCV
Input High Voltage	RESET, UARTs, CAN, USB, SPI, I2C, GPIO	-0.3	–	0.8	DCV
Output Low Voltage	UARTs, CAN, USB, I2C, SPI, GPIO	–	–	0.4	DCV
Output High Voltage	UARTs, CAN, USB, I2C, SPI, GPIO	VCC - 0.4	–	–	DCV

2.2.4 Pin definition

Pin arrangement of OEM TT-RR is shown on the figure below:

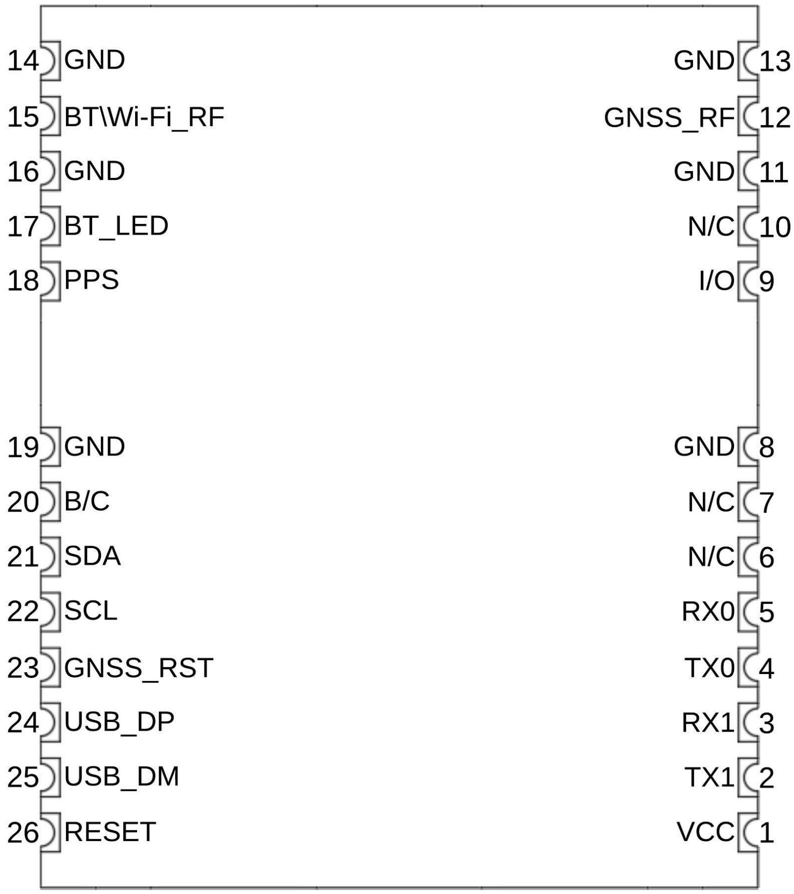


Fig. 1: Pin arrangement of OEM TT-RR

Table 5: Pin definitions of OEM TT-RR

Pin number	Pin Name	Pin Type	Description
1	VCC	PWR	Power supply input
2	TX1	OUT	UART1 GNSS module NMEA data output
3	RX1	IN	UART1 GNSS module data input - not recommend sending anything due to possible malfunctioning main module
4	TX0	OUT	Main UART TX
5	RX0	IN	Main UART RX
6	NC	N/A	No commercial use (keep floating)
7	NC	N/A	No commercial use (keep floating)
8	GND	PWR	Common ground
9	USER_PIN_1	OUT	USER_PIN_1 logical HIGH/LOW set by AT-command
10	NC	N/A	No commercial use (keep floating)
11	GND	PWR	Common ground
12	GNSS_RF	IN	GNSS RF Input (antenna)
13	GND	PWR	Common ground
14	GND	PWR	Common ground

continues on next page

Table 5 – continued from previous page

Pin number	Pin Name	Pin Type	Description
15	BT_WIFI_RF	IN	RF Output (antenna)
16	GND	GND	Common ground
17	BT_LED	OUT	LED Output (digital)
18	PPS	CMOS Input	PPS GNSS signal
19	GND	PWR	Common ground
20	B/C	IN	Bootloader / Configuration mode
21	SDA	IN/OUT	I2C Data line
22	SCL	IN/OUT	I2C Clock line
23	RST_GNSS	IN	Reset input / active low
24	USB_DP	OUT	USB+
25	USB_DM	IN	USB-
26	RESET	IN	Reset input / active low

2.3 Mechanical specification

2.3.1 Dimensions

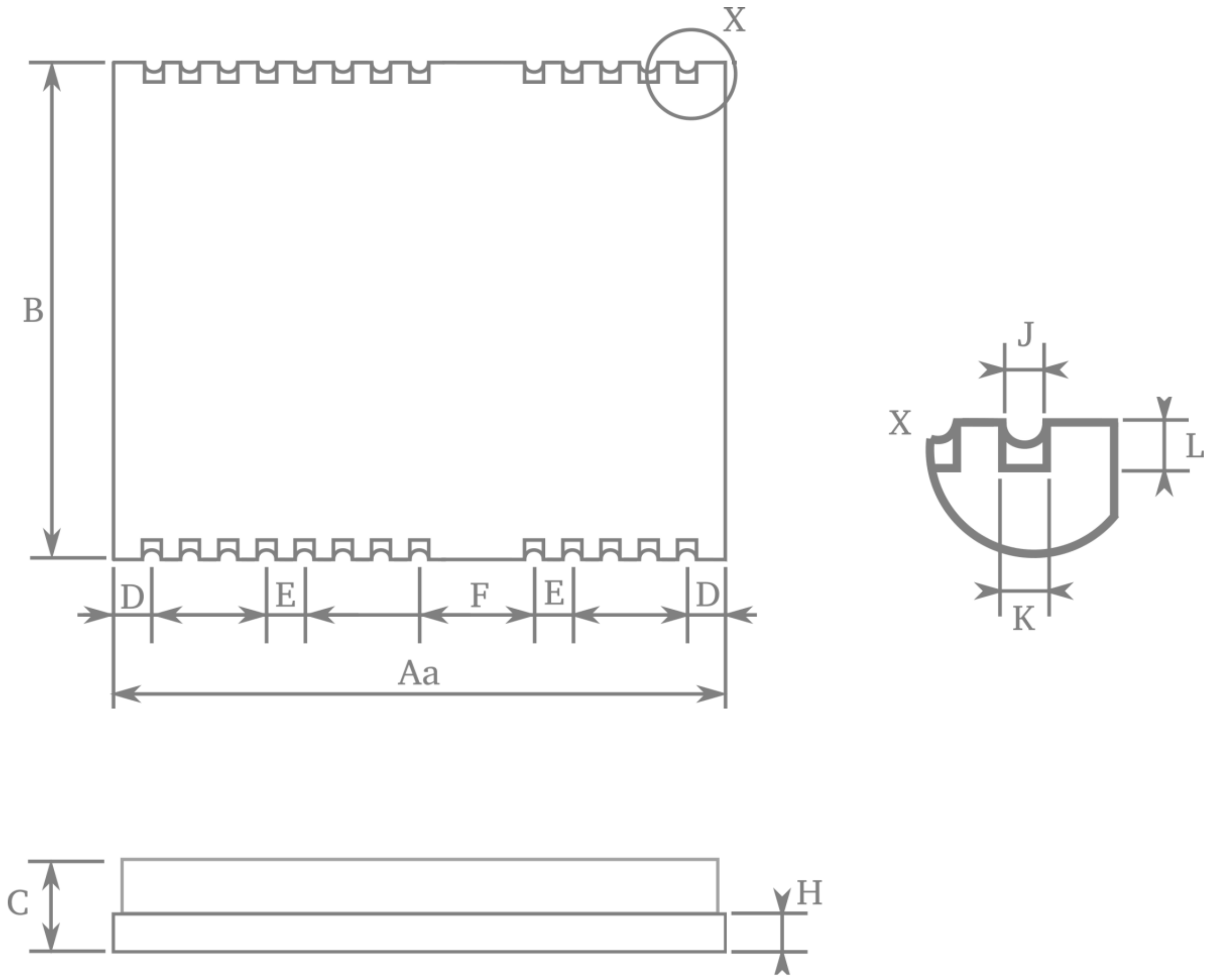


Fig. 2: Mechanical drawing of OEM TT-RR

Table 6: Dimensions and tolerances.

Symbol	Min. (mm)	Typ. (mm)	Max. (mm)
A	15.9	16	16.1
B	12.9	13	13.1
C	2.6	2.7	2.8
D	0.9	1	1.1
E	0.9	1	1.1
F	2.9	3	3.1
G	0.6	0.7	0.8
H	0.4	0.5	0.6
I	0.6	0.7	0.8
J	0.7	0.8	0.9
K	5.4	5.5	5.6
L	0.25	0.35	0.45

2.3.2 Recommended layout

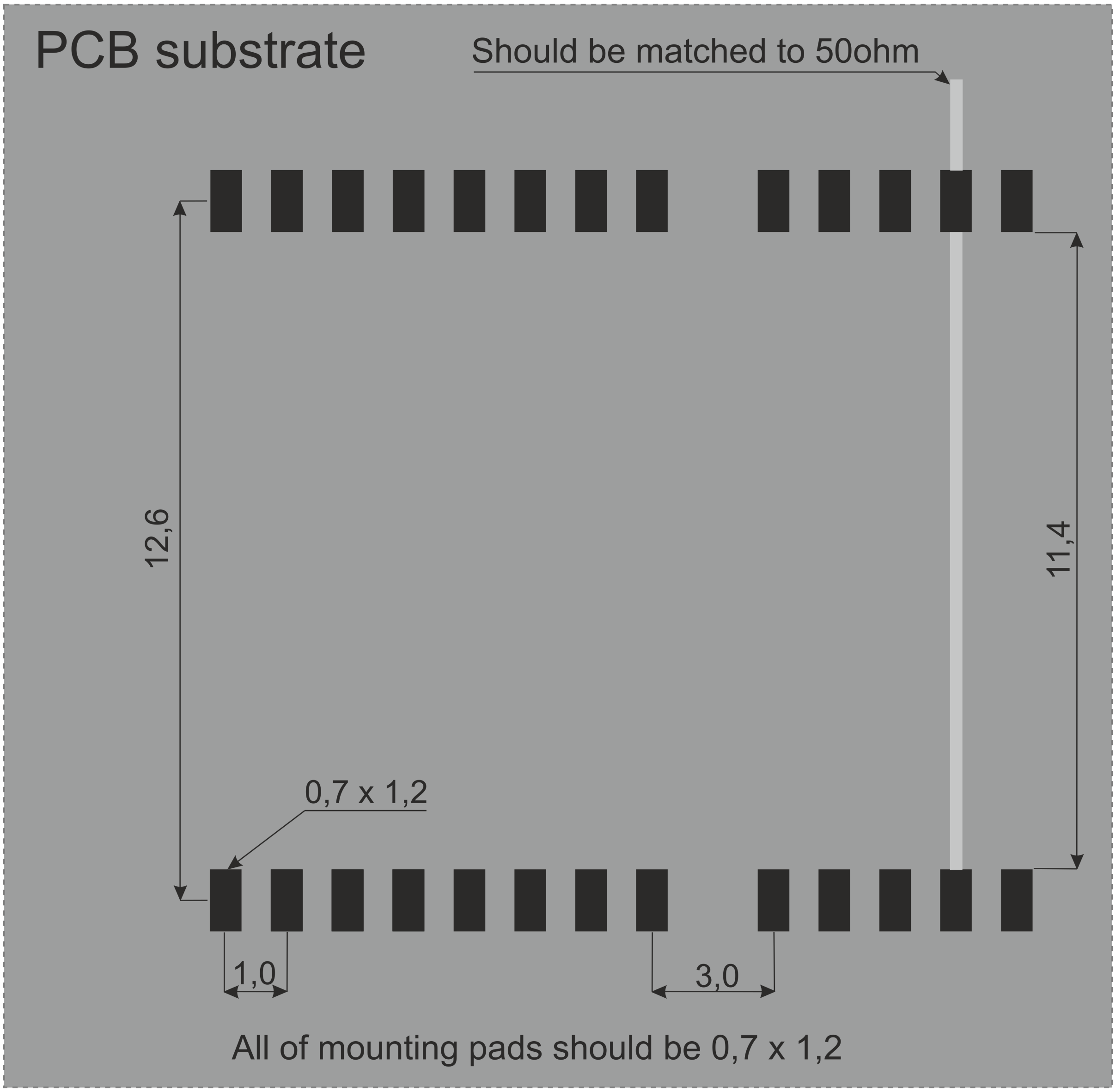


Fig. 3: Footprint of OEM TT-RR

Important:

In case of OEM the RF pads indicated in the *footprint* (page 9) should be matched to 50ohm.

2.3.3 Soldering

Reflow Soldering

It’s advisable to opt for a convection-based soldering oven instead of an infrared radiation one. Convection ovens offer precise temperature control, ensuring uniform heating of all components regardless of their properties, thick- ness, or surface color. For more details, you can refer to the ”IPC-7530 Guidelines for temperature profiling for mass soldering (reflow and wave) processes,” issued in 2001.

We highly suggest take look at our soldering *profile* (page 10) and try to adapt it to your process. The soldering process strictly depends on the thickness of the PCB, solder paste and soldering profile, for this reason we recommend to choose the soldering method directly to the specific project.

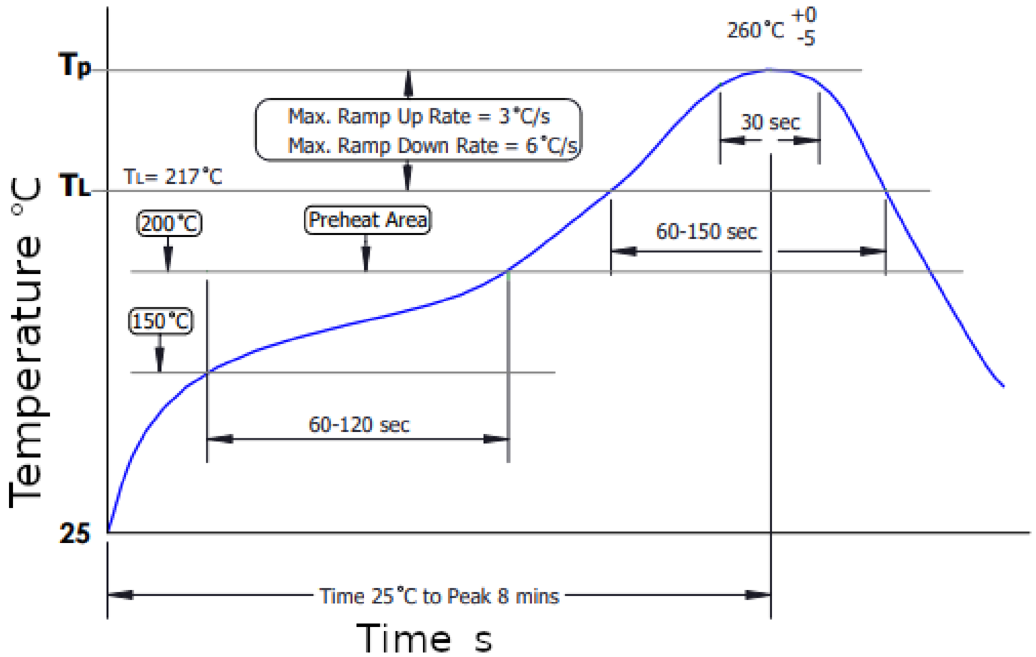


Fig. 4: Recommended soldering profile for OEM TT-RR

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.

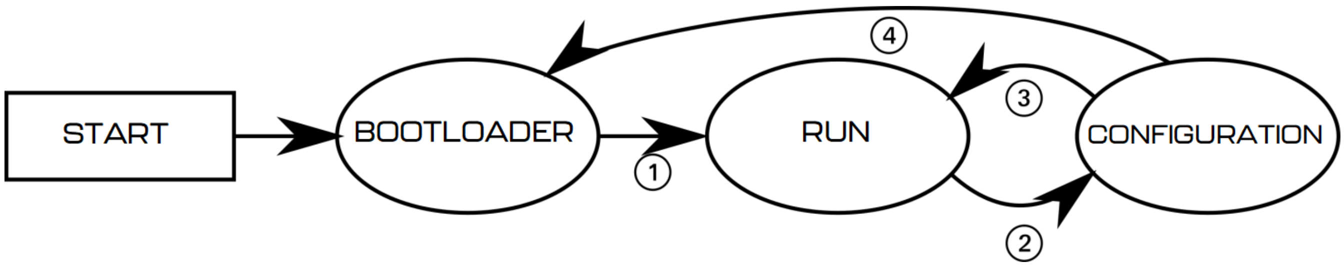
The UART interface uses settings as described in table below:

Table 7: 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 transitions 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 3 seconds. If no BOOTLOADER trigger is present, module will transition 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 state transition, different conditions must be met, which are described below. Generally, the only stable state is RUN. Module always tends to transition 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 a short period of time if no action is taken. To prevent transition from BOOTLOADER state, one of following actions must be taken:

- 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 transition into RUN state. Firstly it will check firmware integrity. When firmware integrity is confirmed, module will transition into RUN state, if not, it will stay in BOOTLOADER state.

To transition 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 transition from RUN into CONFIGURATION state:

- 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 transition from CONFIGURATION into RUN state:

- 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 to 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 8: 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
USER_PIN_1	0	1	0	Set the logical level high 0 - disable 1 - enable

Note:

USER_PIN_1 in the MP1 product series turns on power to the active GNSS antenna.

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 a set of additional commands. Format of these commands is 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
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)]
```

(continues on next page)

(continued from previous page)

```
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]
```

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 9: 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 10: Description of status JSON fields.

JSON Field	Unit	Example	Description
src	–	ID-0000001	See table *Description of main JSON fields.* (page 22).
ts	milliseconds	69061337	See table *Description of main JSON fields.* (page 22).
ver	–	1	See table *Description of main JSON fields.* (page 22).
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

7 RemoteID transceiver subsystem

7.1 Settings

Table 11: 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
DRONE_ID_OPERATION_CATEGORY	0	3	0	Operation category: 0 – None 1 – Open 2 – Specific 3 - Certified
DRONE_ID_OPERATOR_ID	–	–	–	Operator message payload

continues on next page

Table 11 – continued from previous page

Setting	Min	Max	Def	Comment
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_SCAN_ENABLE_BT	0	1	0	Scan enable Bluetooth
DRONE_ID_SCAN_ENABLE_WIFI	0	1	0	Scan enable Wi-Fi
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

7.2 Protocols

7.2.1 RemoteID CSV protocol

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

```
#B4\B5\WN\WB :UAS_ID,ID_TYPE,UAS_TYPE,LAT,LON,HEIGHT,ALT_BARO,ALT_GEO,TRACK,
VELH,VELV,STATUS_FLAG,OPERATOR_ID,OPERATOR_ID_TYPE,OPERATOR_LAT,OPERATOR_LON,
OPERATOR_LOC_TYPE,TIMES,RSSI,CRC\r\n
```

Table 12: Descriptions of RemoteID fields.

#B4-B5-WN-WB	Aircraft message start indicator	Example value
UAS_ID	aircraft ID	18099300000132
ID_TYPE	Flags bitfield *Descriptions of RemoteID ID Type field.* (page 26)	1
UAS_TYPE	Callsign of aircraft *Descriptions of RemoteID UAS_TYPE field.* (page 27)	2
LAT	Latitude, in degrees, accuracy 0.6 degree	57.57634
LON	Longitude, in degrees, accuracy 0.6 degree	17.59554
HEIGHT	Height based on start up altitude, in meters	0.5
ALT_BARO	Barometric altitude, in meters	50
ALT_GEO	Geometric altitude, in meters	50
TRACK	Track of aircraft, in degrees [0,360)	35
VELH	Horizontal velocity of aircraft, in m/s, accuracy 0.1 m/s	464
VELV	Vertical velocity of aircraft, in m/s, accuracy 0.1 m/s	-1344
STATUS_FLAG	Operation status	0
OPERATOR_ID	The operator number from local FAA department	AAABBBBBBBBBBBBC-DDD
OPERATOR_ID_TYPE	Specific type of Operator ID	5
OPERATOR_LAT	The operator latitude in degrees, accuracy 0.6 degree	57.52614
OPERATOR_LON	The operator longitude in degrees, accuracy 0.6 degree	17.60154
OPERATOR_LOC_TYPE	The operator location type	0
TIMES	Timestamp of the sent frame expressed in seconds since current hour, accuracy 0.1 s-1.5 s	408.5
RSSI	Signal strength, in dBm	0
SELF_ID_TYPE	Self id type *Descriptions of RemoteID SELF_ID_TYPE field.* (page 27)	0
SELF_ID	Self id	
FTYPE_TYPE	Frame type *Descriptions of RemoteID FTYPE_TYPE field.* (page 27)	15
MAC	MAC address	df:a5:c3:84:78:66
CRC	CRC16 (described in CRC section)	2D3E

Whereby the following prefixes mean:

- #B4 - Bluetooth 4.0(Legacy) frame
- #B5 - Bluetooth 5.0 frame
- #WN - Wi-Fi NaN frame
- #WB - Wi-Fi becon frame

Table 13: Descriptions of RemoteID ID Type field.

ID Type value	Description
0	None.
1	Serial Number.
2	CAA Assigned Registration ID.
3	UTM Assigned UUID.

Below is a list of emitter category values returned in UAS_TYPE value field.

Table 14: Descriptions of RemoteID UAS_TYPE field.

UAS_TYPE value	Description
0	None.
1	Aeroplane.
2	Helicopter or Multirotor.
3	Gyroplane.
4	Hybrid Lift.
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.

Table 15: Descriptions of RemoteID SELF_ID_TYPE field.

Self Id Type value	Description
0	Text Description.
1	Emergency Description.
2	Extended Status Description.
3–200	Reserved.
201–255	Available for private use.

Table 16: Descriptions of RemoteID FTYPE_TYPE field.

Frame Type value	Description
0	Basic ID.
1	Location.
3	Self ID.
4	System.
5	Operator ID.
15	Packed all in one.

Note:

Referring to the ASD-STAN prEN 4709-002 standard, our product displays all the required information (ASD-STAN prEN 4709-002 Table 1 - Data Dictionary), optional data is only available upon special request.

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

```
#B5:18099300000170,1,0,53.3960175,14.6283543,-0.5,58.0,86.5,0,0.0,0.0,0,,0,0.
0000000,0.0000000,0,103.7,-46,0,,15,84:f7:03:28:e3:1a,420C\r\n
```


```
#B5:18099300000170,1,0,53.3960175,14.6283543,-0.5,58.0,86.5,0,0.0,0.0,0,,0,0.
0000000,0.0000000,0,103.7,-46,0,,15,84:f7:03:28:e3:1a,420C\r\n
```

Note:

RSSI is measured based on analog RF signal.

Statistics message

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

```
#SR:FPS,FPB4_0S,FPB4_1S,FPB4_3S,FPB4_4S,FPB4_5S,FPB5S,FPNS,FPBS,CRC\r\n
```

Table 17: Descriptions of RemoteID Statistic frame.

#SR	Statistics message start indicator	Example
FPS	Number of frame received in last second %	1
FPB4_0S	Number of frame received in last second %	1
FPB4_1S	Number of legacy basic ID Bluetooth 4.0 frame send in last second	1
FPB4_3S	Number of legacy location Bluetooth 4.0 frame send in last second	1
FPB4_4S	Number of legacy self ID Bluetooth 4.0 frame send in last second	1
FPB4_5S	Number of legacy system Bluetooth 4.0 frame send in last second	1
FPB5S	Number of packed all in one Bluetooth 5.0 frame send in last second	1
FPNS	Number of packed all in one Wi-Fi NaN frame send in last second	1
FPBS	Number of packed all in one Wi-Fi beacon frame send in last second	1
CRC	CRC16 (described in CRC section)	2D3E

7.2.2 RemoteID MAVLink protocol

RemoteID has MAVLink protocol autodetect mode, if input data will be in MAVLink mode then device automatically switch to this protocol. This option will be available only when standalone mode is not enabled. For fully detailed information about MAVLink protocol take a look here

8 GNSS receiver subsystem

8.1 Settings

Table 18: Descriptions of GNSS settings

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

8.2 Protocols

8.2.1 GNSS NMEA RAW protocol

Note:

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

8.2.2 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,
        "nic": 12
    }
}
```

Table 19: 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.

9 Sensors receiver subsystem

9.1 Settings

Table 20: Descriptions of Sensors settings.

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

9.2 Protocols

9.2.1 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 21: 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

9.2.2 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 22: Descriptions of JSON Sensor section fields.

JSON Field	Unit	Example	Description
ver	–	1	See table *Description of main JSON fields.* (page 22).
sensor	–	type of sensor	

continues on next page

Table 22 – 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 %.

10 General information

TT-RR is an OEM solution. This means that the device is designed for drones and UTM development, it is equipped with a high-quality multi-GNSS receiver and a barometric altitude sensor. Using BLE and WiFi transmission technology, the device provides surveillance and the ability to identify the drone operator based on any modern mobile device such as a smartphone or tablet.

The device automatically detects the drone's start and immediately starts transmitting a broadcast until the drone is turned off.

Its small size and low power consumption allow it to be used in ultralight drones. AT commands provide the ability to configure the messages to be transmitted, such as the drone's identification number, aircraft type, etc. Additional authentication mechanisms are also available.

10.1 Operator number

The operator number can be obtained from the country's state registration system. This number must be entered into the device using the AT command or the Micro ADS-B software. The writing process requires 3 additional extra digits used to check integrity or provide temperament.

Table 23: Operator number

OPERATOR NUMBER	SEPARATOR	SECURE CHARACTERS
AAABBBBBBBBBBBBC	–	DDD

Example command: *AT+DRONE_ID_OPERATOR_ID=AAABBBBBBBBBBBBC-DDD*

If the Operator ID contains any error, the message '"Operator ID not correct!"' will appear.

By default, the device broadcasts the serial number assigned to the device in the manufacturing process, this number cannot be overwritten.

10.2 Status Led

When the device is in boolader or configuration mode the led diode lights up continuously. In boot mode or when the device has an error the led blinks very fast. If the device is ready to fly the led blinks slowly, once per second.

Table 24: Status LED

DEVICE STATUS	STATUS LED	POWER LED
boot	light	light
configuration	light	light
error or calibration	blink fast	light
ready to fly, airborne	blink slowly	light

10.3 Device status indicator

If an error occurs, it can be easily detected by observing the STATUS LED. The device automatically changes its Remote ID status to emergency. Additional information is described in the SelfID message, which can be easily identified by other airspace users.

10.4 Troubleshooting

10.4.1 Range problem

Most coverage problems occur when the device is mounted in the wrong place or with the wrong orientation.

- check the antenna and ufl connector
- change the position and orientation of the device

10.4.2 Low Frame rate

In long range or radio interference environments, some frames may be missing.

- decrease period between broadcasts using DRONE_ID_BASIC_BROADCAST_PERIOD and DRONE_ID_LOCALIZATION_BROADCAST_PERIOD parameters in configuration mode

10.4.3 TT-RR still blinking fast

In most cases, the problem is the lack of a GNSS fix.

- Check GNSS antenna connection
- Device is placed in wrong orientation
- Device is placed in heavy noisy environment.

10.4.4 After start no message are sending

The device does not receive GNSS corrections before launch. It is very important to wait until the device is ready for launch.

- Reset the module and wait for the GNSS position to be fixed, then start again.

11 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.