Aerobits

Subsystems for the
UAS intergration into
the airspace

OEM TT-SF2

Data sheet & User manual

Introduction

TT-SF1 is a high quality and low price OEM **ADS-B/GNSS** receiver/**FLARM** transceiver series operating at 1090MHz and region dependent FLARM frequency.

It is based on the proven **FPGA-In-The-Loop™** technology, which is a unique combination of a single-core processor and FPGA. The patented solution allows high-speed RF data processing with significantly reduced number of electronic components. Simultaneous miniaturization of the module and its OEM nature open a wide range of possible applications.

The basic version of module offers the possibility of receiving and decoding **ADS-B** and **Mode-A/C/S** in different modes. The analysis of the power/quality of the RF signal and the use of time stamps facilitates the implementation of multilaterations, 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. There are several communication interfaces, protocols and special functionalities available on request.

NOTE: The device to operate on FLARM frequency requires FLARM UAS license. The license can be obtained with the device from Aerobits upon purchase. FLARM library expire after year and must be updated with latest firmware.

Applications

- SAA / DAA (Sense and Avoid / Detect and Avoid)
- UAS ground stations and high-density traffic surveillance
- UTM / U-Space construction (traffic surveillance network)
- Traffic-flow analysis and statistics
- Monitoring of 1090MHz band (signal integrity check)
- ADS-B/In/Out devices that meet the NextGen/SESAR philosophy

Main features

- Fastest ADS-B implementation on a surface of <4cm2
- Receiving of ADS-B, Mode-A/C/S with RF signal strength/quality analysis
- Time stamp (raw data only) for multilateration
- Multiple supported protocols, i.a. RAW HEX, CSV, AERO, MAVLink, ASTERIX, GDL90
- Integrated high quality GNSS position source
- Licensed FLARM transceiver
- High-resolution ADC with real-time signal processing; best-in-class aircraft tracking
- High sensitive front-end, jamming and ESD protection (only version b) with ranges over 150 km (open space, 1dBi antenna)
- Simple module integration via UART interface and AT commands
- Scalable OEM solution with enormous customization potential (additional functions or interfaces on request)
- Firmware update capability (uC and FPGA)

- Designed to meet MOPS defined in TSO-C199

For more information please contact: support@aerobits.pl.

Contents

1 Technical parameters

1.1 Basic technical information

Parameter	Description	Min.	Typ.	Max.	Unit
Carrier frequency ADS-B		-	1090	-	MHz
RX sensitivity GNSS		-	approx. -167	-	dBm
RX sensitivity FLARM		-	approx. -104	-	dBm
TX power FLARM		-	-	14	dBm
RX sensitivity ADS-B		-	approx. -84	-	dBm
UART (baud)	AT commands	115200	921600	3000000	bps
MSL	Moisture Sensitivity Level	-	4	-	-

Table 1: General technical parameters.

1.2 Electrical specification

1.2.1 Absolute maximum ratings

Parameter	Min.	Max.	Unit
Storage temperature	-5	+40	°C
Supply voltage (VCC)	3.0	3.6	DCV
Other pin voltage	-0.3	VCC + 0.3	DCV
RF input ADS-B	-	+10	dBm

Table 2: Absolute maximum ratings.

1.2.2 Recommended operation conditions

Parameter	Min.	Max.	Unit
Operation temperature	-30	+85	°C
Supply voltage (VCC)	2.7	3.6	DCV

Table 3: Recommended operation conditions.

1.2.3 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	VCC - 0.7	-	VCC + 0.3	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

Table 4: General electrical parameters.

1.2.4 Pin definition

Pin arrangement of OEM TT-SF2 is shown on the figure below (1).

Pin number	Pin Name	Pin Type	Description
1	3V3	IN	Power supply input
2	TX1	OUT	Aux UART TX
3	RX1	IN	Aux UART RX
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	NC	N/A	No commercial use (keep floating)
9	NC	N/A	No commercial use (keep floating)
10	NC	N/A	No commercial use (keep floating)
11	10M	OUT	10Mhz pulse output
12	CS	OUT	SPI chip select
13	MISO	IN	SPI Master Input Slave Output
14	MOSI	OUT	SPI Master Output Slave Input
15	SCK	OUT	SPI Clock
16	GND	PWR	Ground
17	AGND	PWR	Analog ground
	AGND	PWR	Analog ground
20	AGND	PWR	Analog ground
21	GNSS	IN	GNSS RF Input
22	AGND	PWR	Analog ground
23	AGND	PWR	Analog ground
24	ADSB	IN	ADSB RF Input
25	AGND	PWR	Analog ground
26	FIO0	N/A	No commercial use (keep floating)
27	FIO1	N/A	No commercial use (keep floating)
	LED	OUT	Info LED ADS-B/Configuration mode
30	PPS	OUT	Pulse per second output from GNSS
31	GND	PWR	Ground
32	B/C	IN	Bootloader/Configuration mode trigger
33	SDA	IN/OUT	I2C data
34	SCL	OUT	I2C clock
35	C2TX	OUT	CAN logic interface TX
36	C2RX	IN	CAN logic interface RX
37	VBUS	IN	USB connection info (3.3V)
38	UDP	IN/OUT	USB Data+
39	UDM	IN/OUT	USB Data-
40	RST	IN	Module reset
41	NC	N/A	No commercial use (keep floating)

Table 5: Pin definitions of OEM TT-SF2.

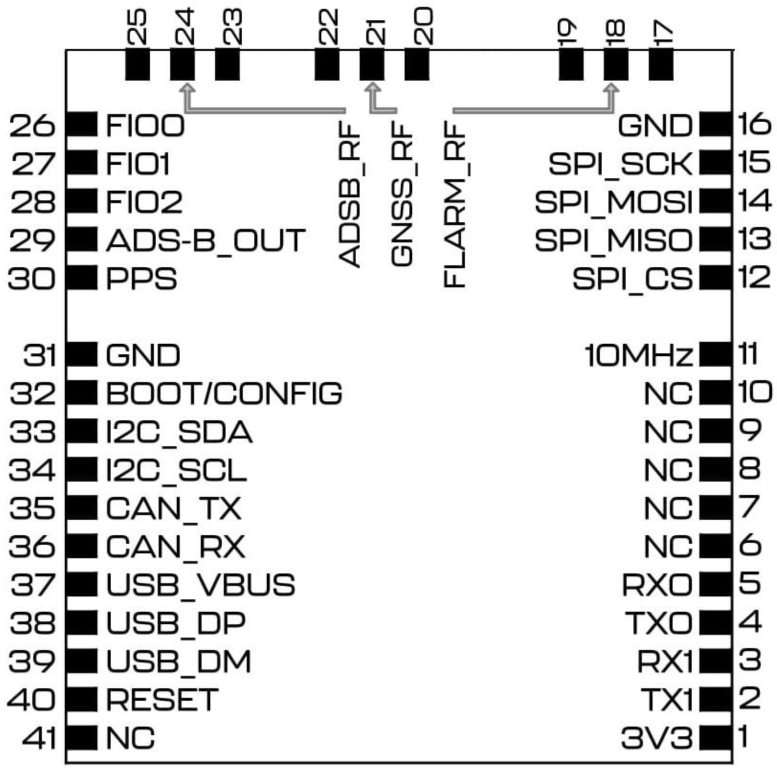

Figure 1: Pin arrangement of OEM TT-SF2.

1.3 Mechanical specification

1.3.1 Dimensions

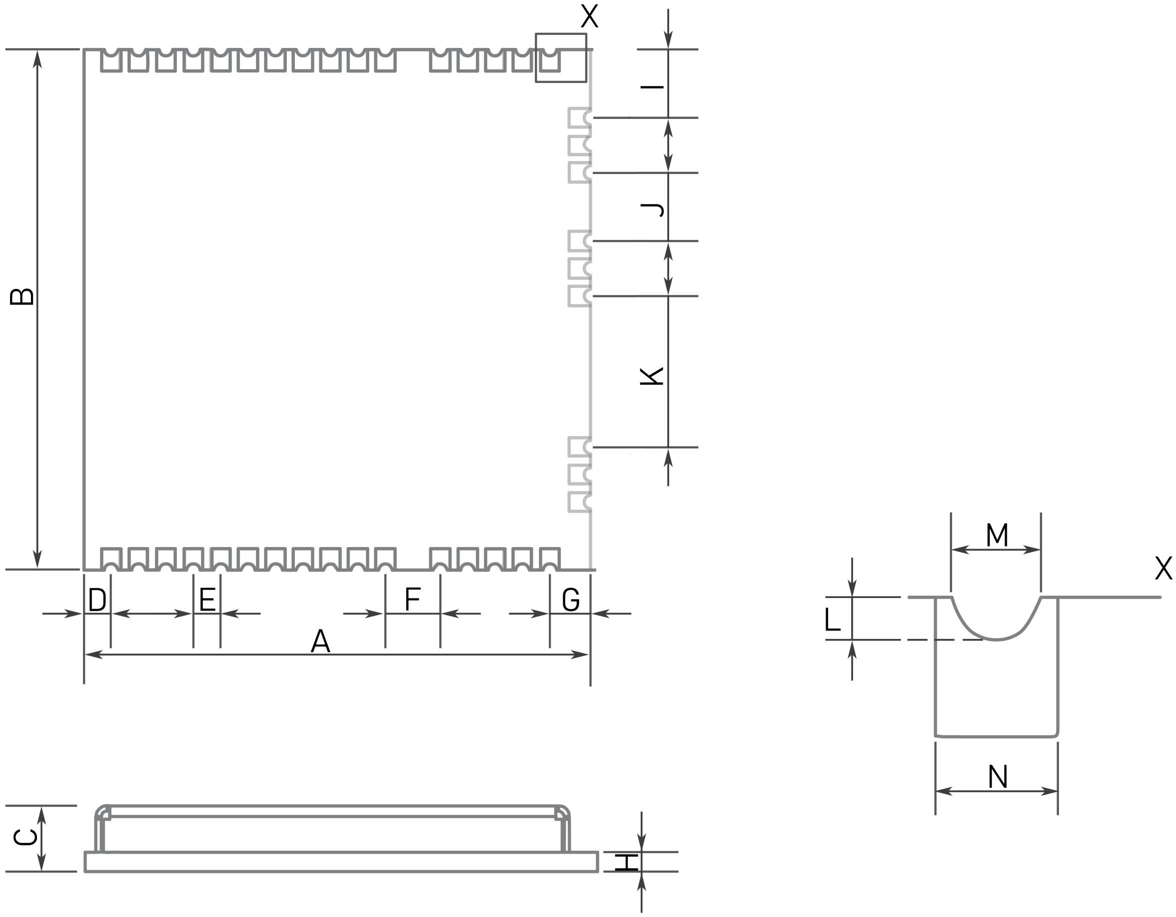

Figure 2: Mechanical drawing of OEM TT-SF2

Symbol	Min. (mm)	Typ. (mm)	Max. (mm)
A	18.4	18.5	18.6
B	18.9	19.0	19.1
C	2.6	2.7	2.8
D	0.9	1.0	1.1
E	0.9	1.0	1.1
F	1.9	2.0	2.1
G	1.4	1.5	1.6
H	0.6	0.7	0.8
I	2.45	2.55	2.65
J	2.3	2.4	2.5
K	5.4	5.5	5.6
L	0.25	0.35	0.45
M	0.4	0.5	0.6
N	0.6	0.7	0.8

Table 6: Dimensions and tolerances.

1.3.2 Recommended layout

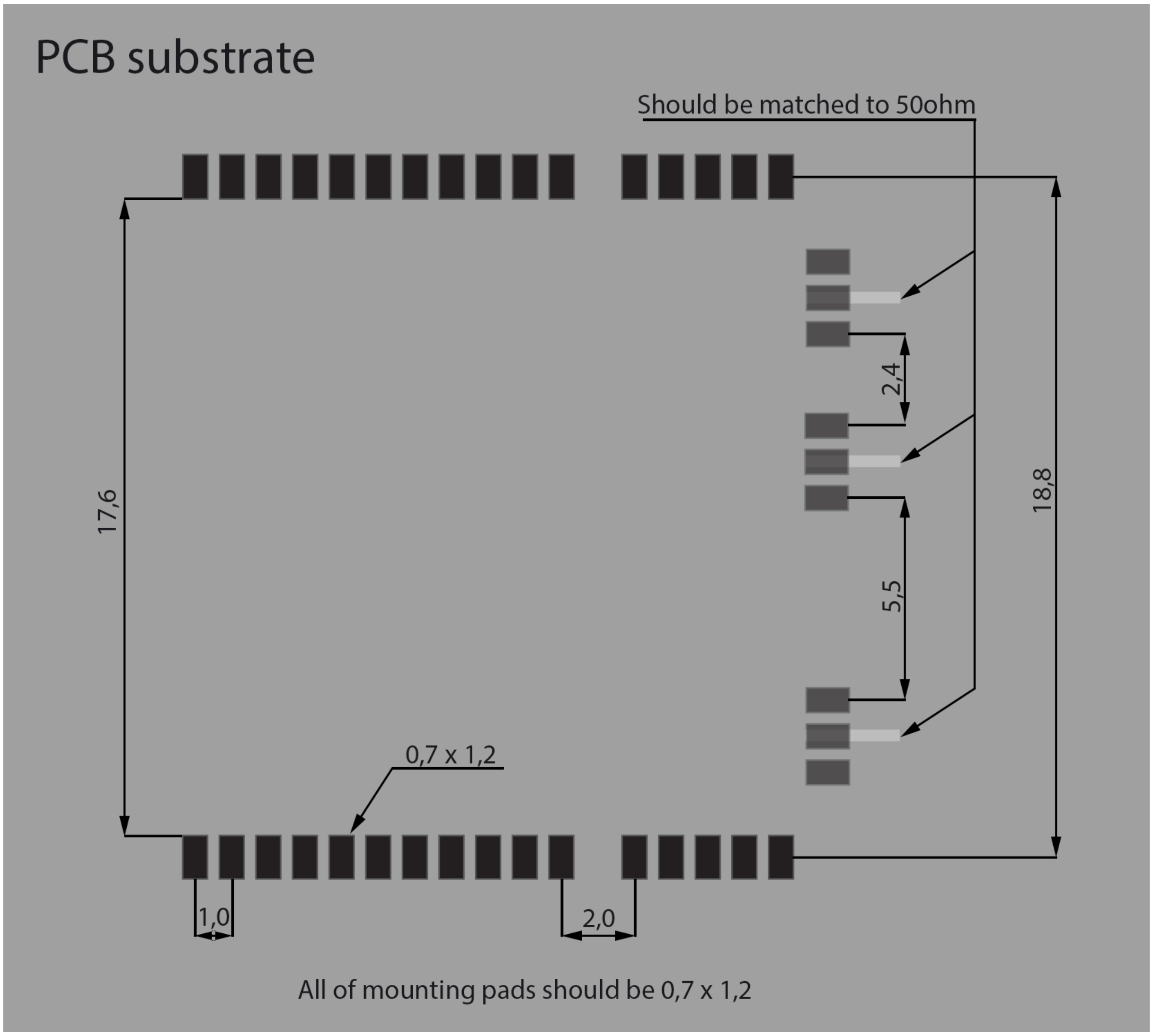

Figure 3: Footprint of OEM TT-SF2

NOTE: In case of OEM the RF inputs indicated in the footprint(3) should be matched to 50ohm.

1.3.3 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, thickness, 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 4 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.

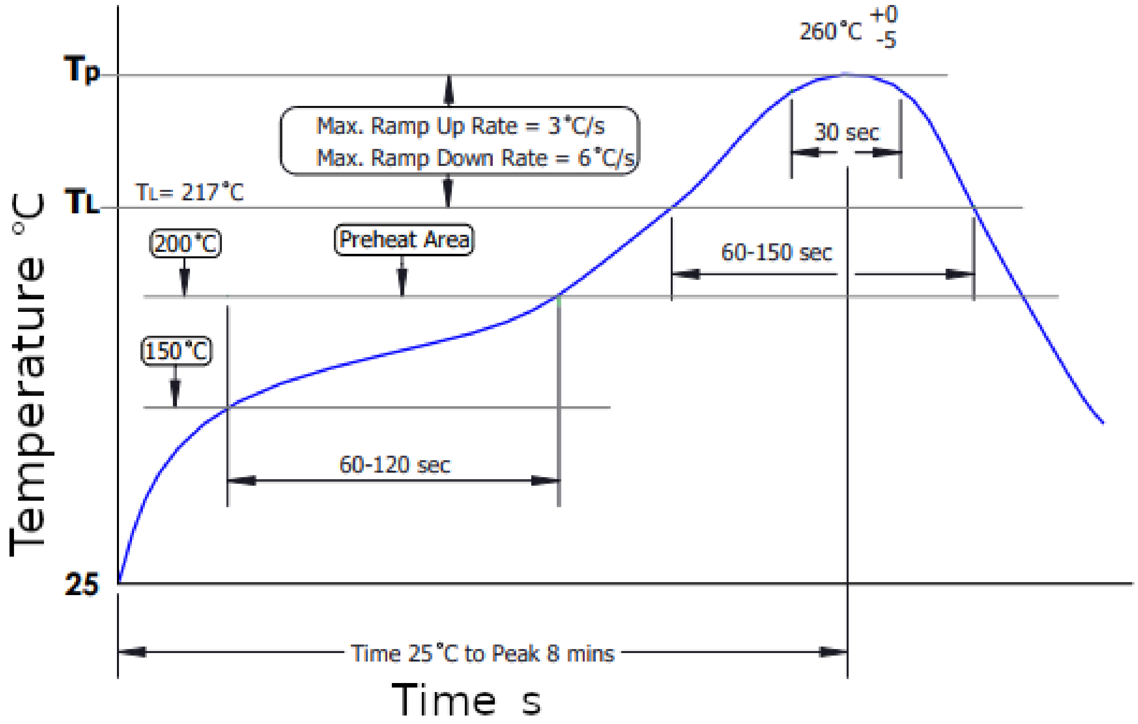

Figure 4: Recommended soldering profile for OEM TT-SF2

2 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.

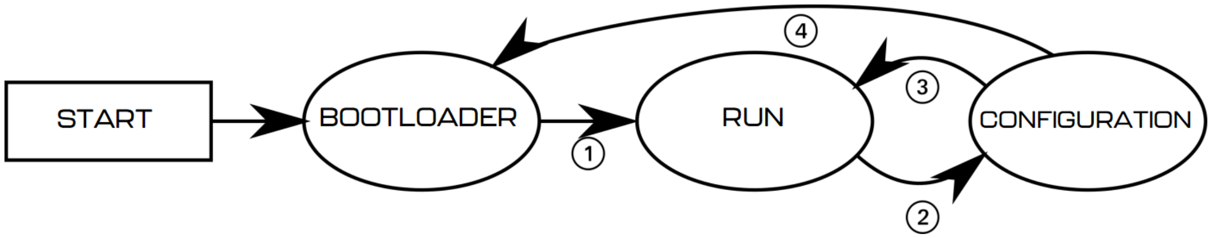

Figure 5: State machine of OEM TT-SF2

2.1 States of operation

2.1.1 BOOTLOADER state

This is an initial state of OEM TT-SF2 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.

2.1.2 RUN state

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.

2.1.3 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.

2.2 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.

2.2.1 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:

- Pull BOOT/CONFIG pin low during start of module
- 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.

To transit into RUN state:

- Release or pull high BOOT/CONFIG pin
- If module is locked, send `AT+LOCK=0` command

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

2.2.2 RUN to CONFIGURATION transition

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

- Pull BOOT/CONFIG pin low
- Send `AT+CONFIG=1` (using current baud). This method is not recommended, because module will support multiple protocols in future and Aerobits Sp. z o.o. cannot ensure that this command will be present in all protocols.

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.

2.2.3 CONFIGURATION to RUN transition

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

- Release or pull high BOOT/CONFIG pin
- 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.

2.2.4 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` .

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 7.

UART Settings				
Parameter	Min.	Typ.	Max	Unit
Baud	115200	921600	3000000	bps
Stop Bits Number	-	1	-	-
Flow Control	-	None	-	-
Parity Bit	-	None	-	-

Table 7: UART settings.

4 Settings

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.

4.1 Write settings

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

`AT+SETTING=VALUE`
For example `AT+ADSB_MODE=3`
Response: `AT+OK`

4.2 Read settings

`AT+SETTING?`
For example: `AT+ADSB_MODE?`
Response: `AT+ADSB_MODE=3`

4.3 Settings description

`AT+SETTING=?`
For example: `AT+ADSB_MODE=?`
Response:

```
Setting: ADSB_MODE
Description: ADS-B receiver mode: 0 - Disabled, 1 - Raw only, 2 - Decoded only,
 3 - Raw and Decoded
Access: Read Write
Type: Integer decimal
Range (min.): 0
Range (max.): 3
Preserved: 1
Requires restart: 0
```

4.4 Errors

Errors are reported using following structure:
`AT+ERROR (DESCRIPTION)`
DESCRIPTION is optional and contains information about error.

4.5 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.

4.6 Uppercase and lowercase

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

```
AT+ADSB_MODE?
AT+aDsB_mOdE?
```

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

4.7 Available settings

Setting	Min	Max	Def	Comment
BAUDRATE	0	2	0	Baudrate in RUN state 0 - 115200bps 1 - 921600bps 2 - 3000000bps
GNSS_LOG	0	2	0	GNSS NMEA forwarding 0 - No forwarding 1 - RMC Messages only 2 - All
FLARM_TX	0	1	1	Flarm TX enabled
FLARM_LOG	0	2	0	Flarm debugging 1 - basic, 2 - detailed
FLARM_AIRCRAFT_TYPE	0	15	0	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 - UFO, 11 - BALLOON, 12 - AIRSHIP, 13 - UAV, 15 - STATIC)
FLARM_MODE	0	1	1	Select mode of FLARM receiver 0 - None 1 - Decoded only
FLARM_DECODED_PROTOCOL	1	5	1	Selected decoded protocol. 1 - CSV (AERO) 2 - JSON 3 - MAVLink 5 - ASTERIX
ASTERIX_SIC	0	255	129	Setting SIC for ASTERIX protocol (Visible when ADSB_DECODED_PROTOCOL=5)
ASTERIX_SAC	0	255	1	Setting SAC for ASTERIX protocol (Visible when ADSB_DECODED_PROTOCOL=5)

ADSB_MODE	0	3	2	Select mode of ADSB receiver 0 - None 1 - RAW only, 2 - Decoded only, 3 - Both RAW and Decoded
ADSB_DECODED_PROTOCOL	1	5	1	Selected decoded protocol. Not all values are valid for all devices. 1 - CSV (AERO) 2 - MAVLink 3 - JSON 4 - GDL90 5 - ASTERIX
ADSB_RAW_PROTOCOL	1	2	1	Selected RAW protocol. Not all values are valid for all devices. 1 - RAW HEX 2 - BEAST
SUBPROTOCOL	0	0	0	Reserved for future use

Table 8: Settings

4.8 Example

As an example, to switch OEM TT-SF2 module 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_MODE=2\r\n
>> AT+OK\r\n
<< AT+ADSB_DECODED_PROTOCOL=1\r\n
>> AT+OK\r\n
<< AT+CONFIG=0\r\n
>> AT+OK\r\n
```

5 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.

5.1 Commands in BOOTLOADER and CONFIGURATION state

5.1.1 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

5.1.2 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 Commands in CONFIGURATION state

5.2.1 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)

5.2.2 AT+SETTINGS?

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

```
AT+ADSB_MODE=2
AT+ADSB_DECODED_PROTOCOL=1
AT+SUBPROTOCOL=0
AT+BAUDRATE=0
```

5.2.3 AT+HELP

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

```
SETTINGS:
AT+ADSB_MODE=2              [0 - Disabled,1 - Raw, 2 - Decoded, 3 - Raw and Decoded]
AT+ADSB_DECODED_PROTOCOL=1  [0 - None, 1 - CSV, 2 - Mavlink, 3 - JSON,
4 - GDL90, 5 - ASTERIX]
COMMANDS:
AT+HELP                [Show this help]
AT+TEST                [Responds "AT+OK"]
AT+SETTINGS_DEFAULT    [Load default settings]
AT+REBOOT              [Reboot system]
```

5.2.4 AT+SETTINGS_DEFAULT

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

5.2.5 AT+SERIAL_NUMBER

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

Response:

`AT+SERIAL_NUMBER=07-0001337`

5.2.6 AT+FIRMWARE_VERSION

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

Response:

`AT+FIRMWARE_VERSION=10101017(May 11 2018)`

5.2.7 AT+REBOOT

`AT+REBOOT` - Restart module.

5.2.8 AT+REBOOT_BOOTLOADER

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

NOTE: This command also sets lock.

5.3 Commands in RUN state

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

NOTE: This command also sets lock.

6 Protocols

6.1 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. Statistics message.

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.

6.1.1 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);
}
```

6.1.2 ADS-B Aircraft message

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
```

#A	Aircraft message start indicator	Example value
ICAO	ICAO number of aircraft (3 bytes)	`3C65AC`
FLAGS	Flags bitfield, see table 10	`1`
CALL	Callsign of aircraft	`N61ZP`
SQ	SQUAWK of aircraft	`7232`
LAT	Latitude, in degrees	`57.57634`
LON	Longitude, in degrees	`17.59554`
ALT_BARO	Barometric altitude, in feets	`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 feets (v2.6.0+)	`5000`
ECAT	Emitter category, see table 12 (v2.7.0+)	`14`
CRC	CRC16 (described in CRC section)	`2D3E`

Table 9: Descriptions of ADS-B message fields.

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

Table 10: ADS-B message Flags description.

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>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">Reserved</td><td colspan="4">NAC$_p$</td><td colspan="3">NAC$_v$</td><td>NIC$_{baro}$</td><td colspan="4">NIC</td></tr>
</table>

Table 11: Structure of NIC/NAC bitfield in CSV protocol.

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

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 Ballons).
20	Cluster obstacle.
21	Line obstacle.

Table 12: ADS-B emitter category values in CSV protocol.

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: SIGS is measured based on analog RF signal. This signal has DC offset of about 700mV.

6.1.3 FLARM Aircraft message

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
```

#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. See table 14.	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
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 13: FLARM Aircraft message fields.

Aircraft type index	Description
0	Unknown.
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.

Table 14: FLARM aircraft type values.

6.1.4 FLARM Info message

This message returns information about FLARM module and is sent every 20 seconds.

```
#INFO:ID,ID_TYPE,DEV_TYPE,REGION,SERIAL_ID,SERIAL_ID_TYPE,SW_VER,HW_VER,
EXT_TYPE,EXT_DATA\r\n
```

#INFO	FLARM Info message start indicator	Example value
ID	Own id value, in hexadecimal format.	1600BF
ID_TYPE	Own id type. 0: randomized id value, 1: ICAO, 2: FLARM id	2
DEV_TYPE	Type of FLARM transmitter. Always 66.	66
REGION	FLARM Region, if applicable. 0: no region, 69: Europe, 65: America, 87: Worldwide.	0
SERIAL_ID	ID value based on serial number, for OEM licenses only.	1600BF
SERIAL_ID_TYPE	ID_TYPE value based on serial number, for OEM licenses only.	2
SW_VER	Software version number, in hexadecimal.	000804
HW_VER	Hardware version number, in hexadecimal.	4E41
EXT_TYPE	Reserved for future use, decimal format.	0
EXT_DATA	Reserved for future use, 10 hexadecimal bytes.	00000000000000000000

Table 15: FLARM Info message fields.

6.1.5 Statistics message

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

```
#S:CPU,RES,RES,FPSS,RES,RES,CRC
```

#S	Statistics message start indicator	Example
CPU	CPU load in %	12.1
RES	Reserved for future use	-
RES	Reserved for future use	-
FPSS	Number of MODE-S frames received in last second	3
RES	Reserved for future use	-
RES	Reserved for future use	-
CRC	CRC16 (described in CRC section)	2D3E

Table 16: Statistics message fields.

6.2 RAW 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
```

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 nano secounds.	`75BCD15` `(0.123456789s)`
TS24h	Timestamp for multilateration. Time from midnight hex format hex format in nano secounds.	`2B5792B49315` `(47655.123456789s = 13:14:15.123456789)`

Table 17: Extended messages description.

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.

6.2.1 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`

6.2.2 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.

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

Table 18: Description of bits in raw Mode-A/C frames in new protocol version.

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`

6.3 MAVLink protocol

OEM TT-SF2 can be switched to use MAVLink protocol. This can be achieved by altering ADSB_DECODED_PROTOCOL setting. When MAVLink protocol is used, module is sending list of aircrafts every second. MAVLink messages have standarized format, which is well described on official protocol webpage (mavlink.io/en/messages).

6.3.1 ADS-B Aircraft message

Aircrafts are encoded using ADSB_VEHICLE message (mavlink.io/en/messages/common.html#ADSB_VEHICLE). MAVLink message contains several data fields which are described below.

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

Table 19: MAVLink ADSB_VEHICLE message description

6.3.2 FLARM Aircraft message

Aircrafts reported by FLARM use ADSB_VEHICLE message in same format as described in ADS-B Aircraft message section, with following restrictions:

- The FLARM “Aircraft Type” field is translated to MAVLink “Emitter Category” field as shown in table 20.
- ICAO field contains FLARM id value.

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
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

Table 20: FLARM Aircraft Type to Emitter Category translation table

6.3.3 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.

6.4 ASTERIX protocol

OEM TT-SF2 can be switched to use ASTERIX binary protocol. This can be achieved by altering ADSB_DECODED_PROTOCOL 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 OEM TT-SF2 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

6.4.1 FLARM aircraft messages

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 in table 21.

Aircraft Type index	Aircraft Type description	Emitter Category index	Emitter Category description
0	Reserved	0	No information
1	Glider, Motor glider	11	Glider, sailplane
2	Tow plane, tug plane	1	Light aircraft
3	Helicopter, gyrocopter, rotocraft	10	Rotocraft
4	Skydiver, parachute	16	Parachutist, skydiver
5	Drop plane for skydivers	2	Small aircraft
6	Hang glider (hard)	15	Ultralight, handglider, paraglider
7	Hang glider (soft)	15	Ultralight, handglider, paraglider
8	Aircraft with reciprocating engine	1	Light aircraft
9	Aircraft with jet / turboprop engine	2	Small aircraft
10	Reserved	0	No information
11	Balloon (hot, gas, weather, static)	12	Lighter-than-air
12	Airship, blimp, zeppelin.	12	Lighter-than-air
13	Unmanned Aerial Vehicle (UAV)	13	UAV
14	Reserved	0	No information
15	Static obstacle	23	Cluster obstacle

Table 21: FLARM Aircraft Type to ASTERIX Emitter Category translation table

6.5 GDL90 protocol

OEM TT-SF2 can be configured to use GDL90 binary protocol. This can be achieved by altering ADSB_DECODED_PROTOCOL 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 SUBPROTOCOL 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.

6.6 Beast protocol

Original specification: https://github.com/firestuff/adsb-tools/blob/master/protocols/beast.md.

6.6.1 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)

6.6.2 Frame structure

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

6.6.3 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

6.6.4 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.

6.6.5 RSSI

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

6.6.6 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

6.7 JSON Protocol

Each message is encoded as separate JSON object, without any excess whitespace, consisting of fields described in table 22.

Name	Description	Value type
{		
"src": "ID-0000001",	OEM TT-SF2 serial number.	String
"ts": 69061337,	Timestamp in milliseconds, relative to last UTC midnight. Value 69061337 encodes 19:11:01.337. Omitted if unknown.	Unsigned integer
"ver": 1,	JSON protocol version. See details below.	Unsigned integer
"gnss": {...}	One or more of the data fields, described in subchapters below.	Object or array
}		

Table 22: Description of main JSON fields.

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 ommitable.

NOTE: In case of JSON objects consisting of only ommitable 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 changing the version number. If a breaking change occurs in OEM TT-SF2 JSON specification, the version number is guaranteed to be incremented.

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

6.7.1 Status section

The `"status"` section contains status information related to OEM TT-SF2 itself. The example JSON message with this section fields described, is shown in table 23.

JSON field	Description	Value type
`{`		
`"src": "ID-0000001",`	see table 22.	
`"ts": 69061337,`		
`"ver": 1,`		
`"status": {`		
`"fw": "30903679(Jan 15 2021)",`	Firmware version, with same syntax as `AT+FIRMWARE_VERSION` command. Value `30903679` is version 3.9.3.679.	String
`}` `}`		

Table 23: Descriptions of JSON sensor section fields.

6.7.2 GNSS section

The `"gnss"` section contains basic GNSS information. This message is sent once per second. The example JSON message with `"gnss"` section fields described, is shown in table 24.

JSON field	Description	Value type
`{`		
`"src": "ID-0000001",`	see table 22.	
`"ts": 69061337,`		
`"ver": 1,`		
`"gnss": {`		
`"fix": 1,`	Set to 1 if onboard GNSS currently has fix, otherwise 0.	Unsigned integer
`"lat": 53.42854,`	Last known latitude. Omitted if there was no GNSS fix since device boot.	Floating point
`"lon": 14.55281,`	Last known longitude. Omitted if there was no GNSS fix since device boot.	Floating point
`"altWgs84": 499.6,`	Last known WGS-84 Altitude, in meters. Omitted if there was no GNSS fix since device boot.	Floating point
`"altMsl": 508.6,`	Last known MSL Altitude, in meters. Omitted if there was no GNSS fix since device boot.	Floating point
`"track": 127.3,`	Track angle, 0°..360°, relative to true north. Omitted if unknown.	Floating point
`"hVelo": 10.5,`	Horizontal velocity, in knots. Omitted if unknown.	Floating point
`"vVelo": 25.00,`	Vertical velocity, in m/s. Positive value is upwards. Omitted if unknown.	Floating point
`"gndSpeed": [`		
`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.	Floating point
`],`		
`"acc": {`		
`"lat": 5.2,`	Accuracy of latitude, in meters. Omitted if unknown.	Floating point
`"lon": 2.1,`	Accuracy of longitude, in meters. Omitted if unknown.	Floating point
`"alt": 3.6`	Accuracy of altitude, in meters. Omitted if unknown.	Floating point
`},`		
`"nacp": 12`	Navigational Accuracy Category for Position value, as defined in ED-282. Omitted if unknown.	Unsigned integer
`"nacv": 2`	Navigational Accuracy Category for Velocity value, as defined in ED-282. Omitted if unknown.	Unsigned integer
`"nic": 12`	Navigation Integrity Category as defined in ED-282. Omitted if unknown.	Unsigned integer
`}` `}`		

Table 24: Descriptions of JSON GNSS section fields.

6.7.3 Raw ADS-B section

The `"raw"` section contains raw, unprocessed and unfiltered ADS-B frames gathered by OEM TT-SF2 , 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 in table 25

JSON field	Description	Value type
`{`		
`"src": "ID-0000001",`	see table 22.	
`"ts": 69061337,`		
`"ver": 1,`		
`"raw": [`		
`[`		
`"18A9725A4C842D",`	Raw frame bytes, formatted as uppercase hexadecimal. Short Mode-S frames encode 7 bytes, long frames contain 14 bytes.	String
`-78,`	Signal strength, in dBm.	Signed integer
`2,`	Signal quality, in dB.	Unsigned integer
`"295CAB573A77"`	UTC-calibrated time of reception, formatted as uppercase hexadecimal, in nanoseconds. Example translates to 12:37:57.988350583	String
`]`		
`]` `}`		

Table 25: Descriptions of JSON raw ADS-B section fields.

NOTE: Due to constrained throughput of OEM TT-SF2 communication, transmission of some raw frames may be skipped in heavy aircraft traffic situations.

6.7.4 Processed ADS-B reports

The "adsb" section contains aircraft information determined by OEM TT-SF2 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 26. Reports for each ADS-B aircraft are updated once every second.

JSON field	Description	Value type
{		
"src": "ID-0000001",	see table 22.	
"ts": 69061337,		
"ver": 1,		
"adsb": [		
{		
"icao": "DABABE",	ICAO address, 24-bit value encoded in uppercase hexadecimal, with leading zeros.	String
"sigStr": -95,	Signal strength.	Signed integer
"sigQ": 2,	Signal quality.	Unsigned integer
"fps": 5,	Number of raw Mode-S frames received from aircraft during last second.	Unsigned integer
"lat": 53.42854,	Latitude. Omitted if position is unknown.	Floating point
"lon": 14.55281,	Longitude. Omitted if position is unknown.	Floating point
"baroAlt": 1725,	Barometric altitude, in feet. Omitted if unknown.	Signed integer
"geoAlt": 1712,	Geometric altitude, in feet. Omitted if unknown.	Signed integer
"track": 72.18,	Track angle, 0°..360°. Omitted if unknown.	Floating point
"hVelo": 10.5,	Horizontal velocity, in knots. Omitted if unknown.	Floating point
"vVelo": 50,	Vertical velocity, in ft/min, positive value is upwards. Omitted if unknown.	Signed integer
"ident": "TEST8",	Callsign, up to 8 chars. Omitted if unknown.	String
"squawk": "7232",	Squawk, 8 octal digits. Omitted if unknown.	String
"ecat": 13,	Emitter category code, see table 27. Omitted if unknown.	Unsigned integer
"nacp": 3,	NAC_P value, as described in ED-102A. Omitted if value is 0 (unknown).	Unsigned integer
"nacv": 1,	NAC_V value, as described in ED-102A. Omitted if value is 0 (unknown).	Unsigned integer
"nicBaro": 1,	NIC_{BARO} value, as described in ED-102A. Omitted if value is 0.	Unsigned integer
"nic": 2,	NIC value, as described in ED-102A. Omitted if value is 0 (unknown).	Unsigned integer
"surf": 1	Set to 1 if plane is on earth surface. Omitted if plane is in air or unknown.	Unsigned integer
}		
] }		

Table 26: Descriptions of JSON ADS-B section fields.

The emitter category values returned in "ecat" field is shown in table 27.

"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.

Table 27: ADS-B emitter category values in JSON protocol.

6.7.5 Processed FLARM reports

The "flarm" section contains aircraft information determined by OEM TT-SF2 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 28. Reports for each FLARM aircraft are updated once every second.

JSON field	Description	Value type
`{`		
`"src": "ID-0000001",`	see table 22.	
`"ts": 69061337,`		
`"ver": 1,`		
`"flarm": [`		
`{`		
`"idType": 1,`	Aircraft id type. 0: randomized, 1: ICAO, 2: FLARM.	Unsigned integer
`"id": "DABABE",`	Aircraft id, 32-bit value encoded in uppercase hexadecimal, with leading zeros.	string
`"type": 13,`	Aircraft type, see table 29.	Unsigned integer
`"danger": 1,`	Danger level (1-3). Omitted if 0 (no danger)	Unsigned integer
`"lat": 53.42854,`	Latitude.	Floating point
`"lon": 14.55281,`	Longitude.	Floating point
`"alt": 525,`	Barometric altitude, in meters.	Signed integer
`"track": 72`	Track angle, in degrees.	Unsigned integer
`"hVelo": 50,`	Horizontal velocity, in m/s.	Unsigned integer
`"vVelo": 200,`	Vertical velocity, in m/s.	Unsigned integer
`"movMode": 5,`	Movement mode. 1: stationary, 4: circling right, 5: flying, 7: circling left.	Unsigned integer
`"stealth": 1,`	Set to 1 if target has Stealth flag set, otherwise omitted.	Unsigned integer
`"notrack": 1`	Set to 1 if target has Notrack flag set, otherwise omitted.	Unsigned integer
`}`		
`]` `}`		

Table 28: Descriptions of JSON FLARM section fields.

The list of possible FLARM “Aircraft type” values returned in "type" field is shown in table 29.

"type" 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.

Table 29: FLARM Aircraft type values in JSON protocol.