Aerobits

Subsystems for the
UAS intergration into
the airspace

Transceiver TR-1A

Data sheet & User manual

Introduction

TR-1A belongs to second generation of the smallest ADS-B transceivers on market and has been developed for civil and commercial Unmanned Aircraft Systems. The device operates on 1090 MHz and allows to receive and transmit ADS-B data with defined **0.25, 0.5 or 1 Watt output power for ADS-B**. The transceiver does not require external devices to operate. It is equipped with a high quality **multi-GNSS** receiver and a **pressure sensor**. The aluminum housing and ESD protection guarantee high resistance of the device to work in difficult conditions.

TR-1A opens the way to the implementation of the **Detect and Avoid** algorithms, supporting the integration of UAS into the airspace.

NOTE: ICAO addresses are used to provide a unique identity normally allocated to an individual aircraft or registration.

Please do not use random ICAO!

Address becomes a part of the aircraft's Certificate of Registration and MUST be given by Civil Aviation Authority and registered in aircraft database.

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

Main features

- Real-time aircraft tracking on 1090 MHz and 868 MHz
- Patented FPGA-In-The-Loop^{TM} technology with the capability of receiving thousands of frames per second
- Integrated GNSS source and pressure sensor
- Configurable 0.25, 0.5 or 1 Watt RF output power for ADS-B
- Implemented MAVLink and AERO^{TM} protocol
- Programming via AT commands
- Simple plug&play integration

Contents

1 Technical parameters

1.1 Basic technical information

Parameter	Description	Typ.	Unit
Frequency	ADS-B	1090	MHz
Sensitivity	ADS-B	-87	dBm
RF Output power	configurable	+30/+27/+24	dBm
ESD protection	All lines		-
MAVLink (baud)		115200	bps
AERO (baud)	AT commands	115200	bps
Main connector	PXMBNI05RPM04APC		-
Antenna connector	2 x MMCX		-
Temperature range	Operating temperature	-30 to +85	°C
Storage temperature	Optimal storage temperature	-5 to +40	°C
Dimension		35.0 x 25.0 x 8.5	mm
Weight	without cables and antennas	14	grams

Table 1: General technical parameters.

1.2 Electrical specification

1.2.1 Basic electrical parameters

Parameter	Value
Input voltage	5 V
Current consumption	130mA

Table 2: General electrical parameters.

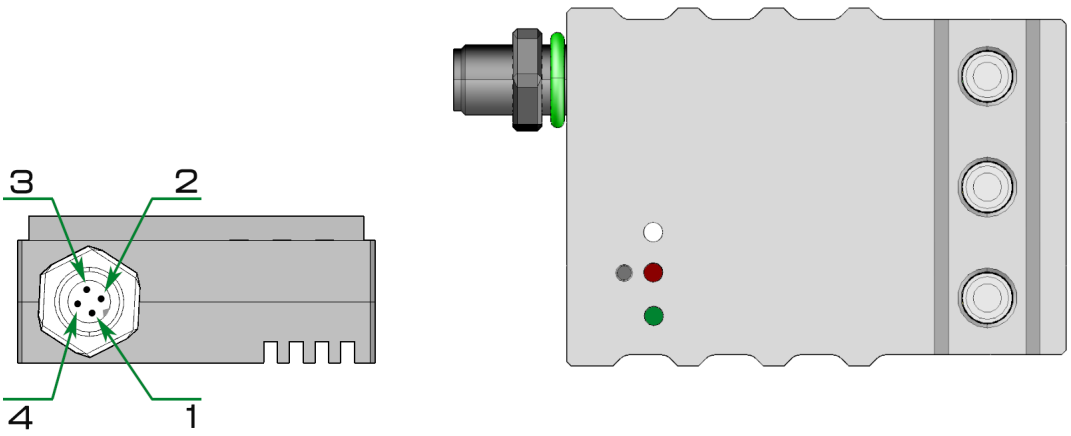


Figure 1: Appendant drawing of Transceiver TR-1A .

1.2.2 PIN definition

PIN	Color	Name	Function
1	Red	+5 V	Power supply
2	Green	TX	Data from device to host
3	White	RX	Data from host to device
4	Black	GND	Ground

Table 3: Pin definition.

1.2.3 LED indicators

LED	Color	Function
POWER	Green	Power supply indicator
GNSS	White	Frame detection / receive indicator
ADS-B	Red	ADS-B OUT indicator 1. OFF – Disabled 2. Blink – Wait for FIX 3. ON – Active

Table 4: LED indicators.

1.3 Mechanical specification

1.3.1 Mechanical parameters

Parameter	Value
Dimensions	35.0 x 25.0 x 8.5 mm
Weight	14 g

Table 5: Mechanical parameters of Transceiver TR-1A

1.3.2 Dimensions

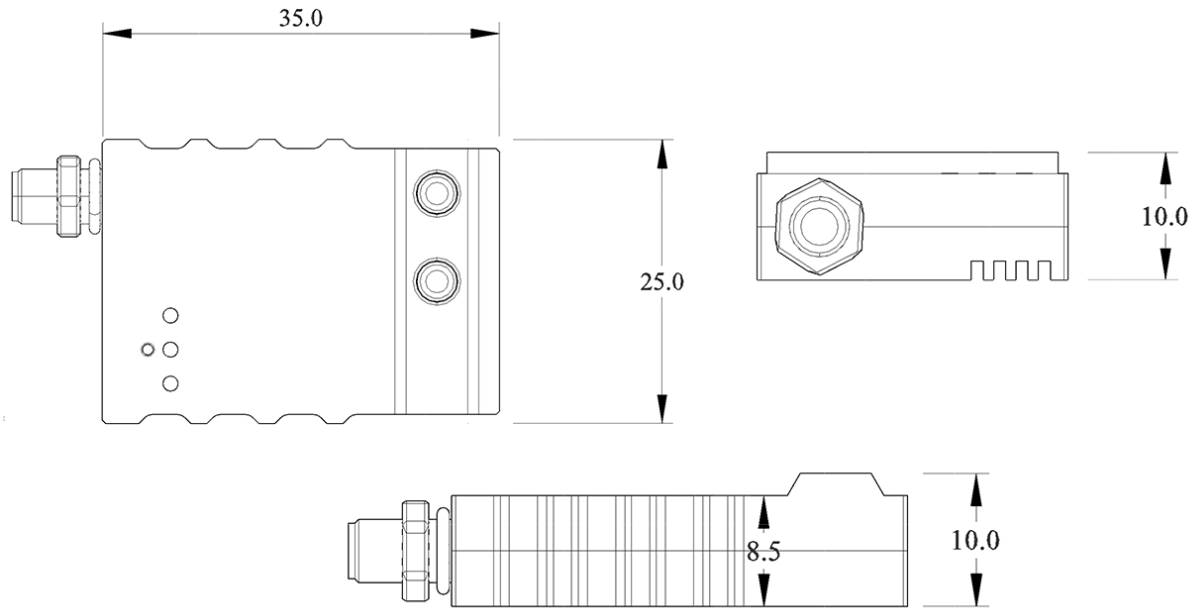


Figure 2: Dimensions of Transceiver TR-1A

1.3.3 Connectors

Connector	Type	Example
Main	Installed on board	BULGIN, PXMBNI05RPM04APC
	Mating connector	BULGIN, PXPPVC05FBF04ACL010PVC
Antenna	Installed on board	MMCX, 73415-0961
	Mating connector	ASMK025X174S11

Table 6: Connectors

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.

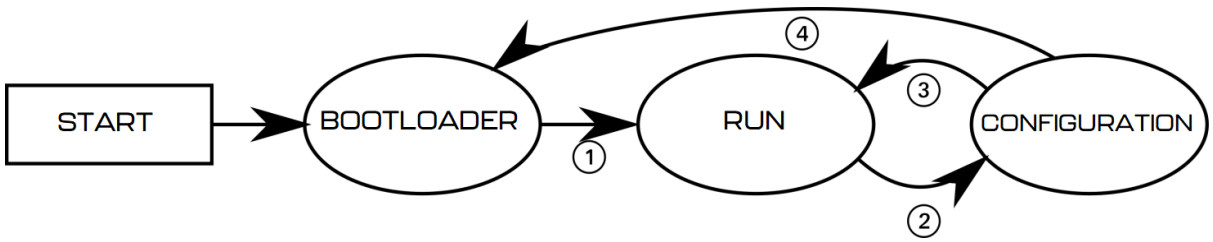


Figure 3: State machine of Transceiver TR-1A

2.1 States of operation

2.1.1 BOOTLOADER state

This is an initial state of Transceiver TR-1A 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:

- 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:

- 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:

- 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.

2.2.3 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.

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.

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+PROTOCOL=1`
Response: `AT+OK`

4.2 Read settings

`AT+SETTING?`
For example: `AT+PROTOCOL?`
Response: `AT+PROTOCOL=1`

4.3 Settings description

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

```
Setting: PROTOCOL
       Description: Selected protocol (0: NONE, 2: CSV, 3: MAVLINK)
       Type: Integer decimal
       Range (min.): 0
       Range (max.): 5
       Is preserved: 1
       Is restart needed: 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+PROTOCOL?
AT+pRoToCoL?
```

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
ICAO	-	-	0	ICAO number broadcasted by this device
SQUAWK	-	-	0	SQUAWK broadcasted by this device
EMITTER_CAT	-	-	0	Emitter category broadcasted by this device
ADSB_TX_ENABLED	0	1	1	ADSB broadcasting enabled
ADSB_TX_ON_BOOT	0	1	1	Device broadcasting ADSB on boot enabled
ADSB_TX_SURFACE	0	1	0	Device broadcasting ADSB when on surface enabled
ADSB_PWR	0	2	2	Trasmiting power of ADS-B (0 - 0.25W, 1 - 0.5W, 2 - 1W)
PROTOCOL	0	8	2	Selected protocol. Not all values are valid for all devices. 0 - None 1 - RAW HEX 2 - CSV (AERO) 3 - MAVLink 4 - ASTERIX 5 - GDL90 7 - BEAST 8 - JSON
SUBPROTOCOL	0	0	0	Reserved for future use
AERO_JSON_BITMASK	0	3F	3B	Determine, what data will be sent over Json protocol. Value is the sum in hexadecimal of required data according to value below. 0x01 - ADSB 0x02 - FLARM 0x04 - RAW 0x08 - STATUS 0x10 - GNSS 0x20 - SENSOR e.g ADSB, FLARM, STATUS, GNSS, SENSOR 0x01 + 0x02 + 0x08 + 0x10 + 0x20 = 0x3B e.g ADSB, FLARM 0x01 + 0x02 = 0x03
PRESSURE_LOG	0	1	0	Show barometer log

Table 8: Settings

4.8 Example

As an example, to switch Transceiver TR-1A 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+PROTOCOL=2\r\n
>> AT+OK\r\n
>> AT+OK\r\n
<< AT+CONFIG=0\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+PROTOCOL=2
AT+SUBPROTOCOL=0
AT+BAUDRATE=0
```

5.2.3 AT+HELP

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

```
SETTINGS:
AT+PROTOCOL=2         [Selected protocol (0: NONE, 2: CSV, 3: MAVLINK)]
AT+SUBPROTOCOL=0      [Subprotocol of selected protocol]
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. 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.

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

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 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 13: 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 14: 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 15: 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

Transceiver TR-1A can be switched to use MAVLink protocol. This can be achieved by altering 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 16: MAVLink ADSB_VEHICLE message description

The ADS-B vehicle may transmit barometric, as well as geometric altitude. The SUBPROTOCOL setting allows for toggling 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.

6.4 ASTERIX protocol

Transceiver TR-1A can be switched to use ASTERIX binary protocol. This can be achieved by altering 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.

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

6.5 GDL90 protocol

Transceiver TR-1A can be configured to use GDL90 binary protocol. This can be achieved by altering 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.

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 17.

Name	Description	Value type
`{`		
`"src": "23-0000001",`	HOD's 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 17: 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 HOD 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 HOD itself. The example JSON message with this section fields described, is shown in table 18.

JSON field	Description	Value type
{		
"src": "23-0000001",	see table 17.	
"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 18: 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 19.

JSON field	Description	Value type
{		
"src": "23-0000001",	see table 17.	
"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 19: Descriptions of JSON GNSS section fields.

NOTE: The nacp, nacv and nic values are not available in regular HOD hardware.

6.7.3 Raw ADS-B section

The `"raw"` section contains raw, unprocessed and unfiltered ADS-B frames gathered by HOD, 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 20

JSON field	Description	Value type
`{`		
`"src": "23-0000001",`	see table 17.	
`"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
`696,`	Signal strength, in mV.	Unsigned integer
`68,`	Signal quality, in mV.	Unsigned integer
`"295CAB573A77"`	UTC-calibrated time of reception, formatted as uppercase hexadecimal, in nanoseconds. Example translates to 12:37:57.988350583	String
`]`		
`]`		
`}`		

Table 20: Descriptions of JSON raw ADS-B section fields.

NOTE: Due to constrained throughput of HOD 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 HOD's 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 21. Reports for each ADS-B aircraft are updated once every second.

JSON field	Description	Value type
`{`		
`"src": "23-0000001",`	see table 17.	
`"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, -180°..180°. 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 22. 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 21: Descriptions of JSON ADS-B section fields.

The emitter category values returned in `"ecat"` field is shown in table 22.

"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 22: ADS-B emitter category values in JSON protocol.

7 Quick start

Transceiver TR-1A is a stand-alone device and in the simplest case of its operation requires only a power supply. However during the first start-up, you must configure the device. That can be performed in the few steps described below. First install the antennas using the MMCXto SMA adapters included in the kit. Also connect the configuration cable that will help you set the device parameters. The following figure shows the installation method.

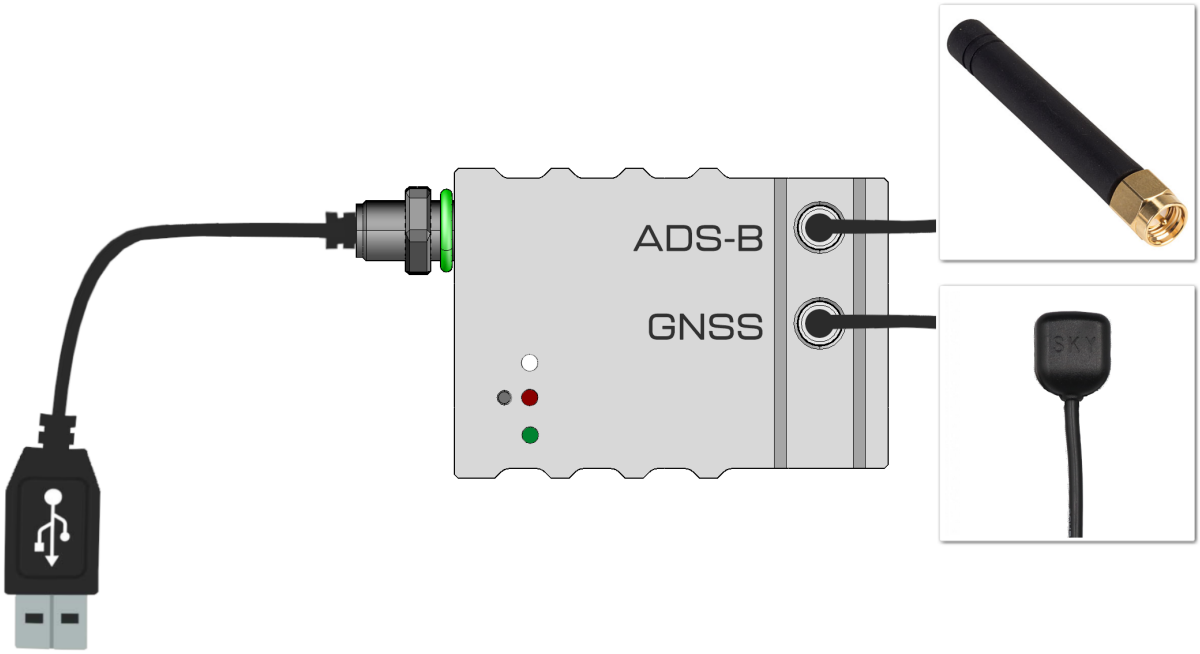


Figure 4: Combination overview

7.1 Specification of used anntena

No.	Part number	Connector type
1 - ADS-B	DELTA1A/X/SMAM/S/S/11	GSM/GPRS, 3G and Ism Stubby Antenna SMA Male
2 - GNSS	AMC-ANTGPSSM-A2M	GPS Active antenna MMCX Male

Table 23: Description of commonly used anntena

7.2 Alternative anntenas

You can also use **effective alternative** to commonly used anntenas. The following one is proper option to increase performence of your device.

Part number	Connector type
GSM-ANT822	GSM Antenna SMA Male

Table 24: Description of alternative anntena

7.3 Antennas mount

For better performance antenna must be mounted on a ground-plane (carbon plate, PCB plate etc.) to radiate efficiently. The antenna should be mounted at the edge of the ground-plane of the mainboard of the device. Also no metal should be used near the antenna, with at least 20mm of clearance required, the more clearance the better. The best way to properly mount antennas is read manufacturer documentation. Example of proper mounting vertical antenna **DELTA1A/X/SMAM/S/S/11** below:

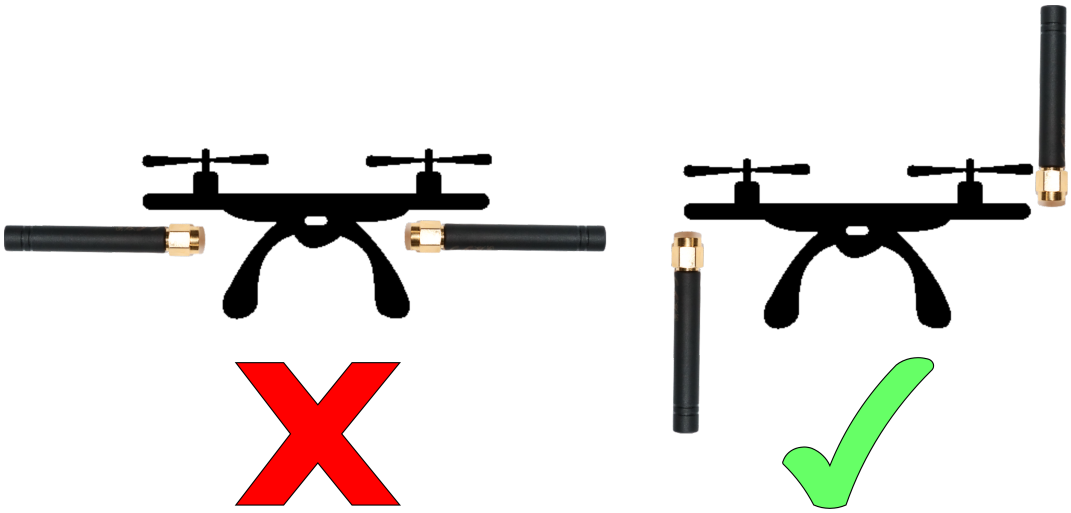
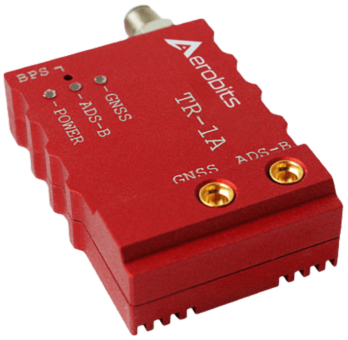


Figure 5: Proper antennas mount

7.4 Scope of delivery

(a) Transceiver TR-1A
(b) ADS-B antenna
(c) MMCX -> SMA cables (250mm)
(d) GNSS antenna
(e) Additional cable bulgin (PXPPVC05FBF04ACL010PVC)
(f) UART -> bulgin converter (500mm)



(a) Device TR1A

(b) ADS-B Antenna

(c) MMCX -> SMA Cable

(d) GNSS Antenna

(e) Bulgin Cable

(f) Converter USB-UART

7.5 Configuration using Micro ADS-B software

1. Connect the device to the PC. The converter is supplied with the FTDI chip. In this case, the installation of the controller takes place automatically.

2. Download the latest Micro ADS-B software from www.aerobits.pl. Install Micro ADS-B on your Windows computer. If the device is connected to a PC, it should be found automatically after clicking the ”Connect” button. The connection window should look similar to the one in the picture. Select the device found and press ”OK”.

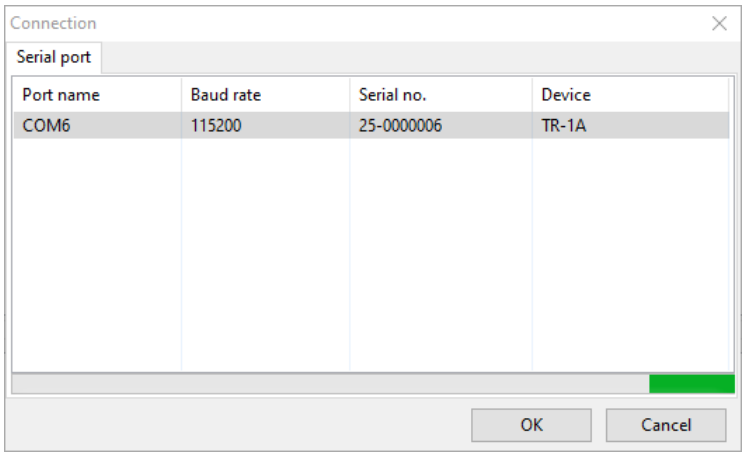


Figure 7: Port select window

3. Press **Settings** to enter the parameterization mode of the module. After setting the parameters, press the ”Ok” button to save the settings. Device is ready to work.

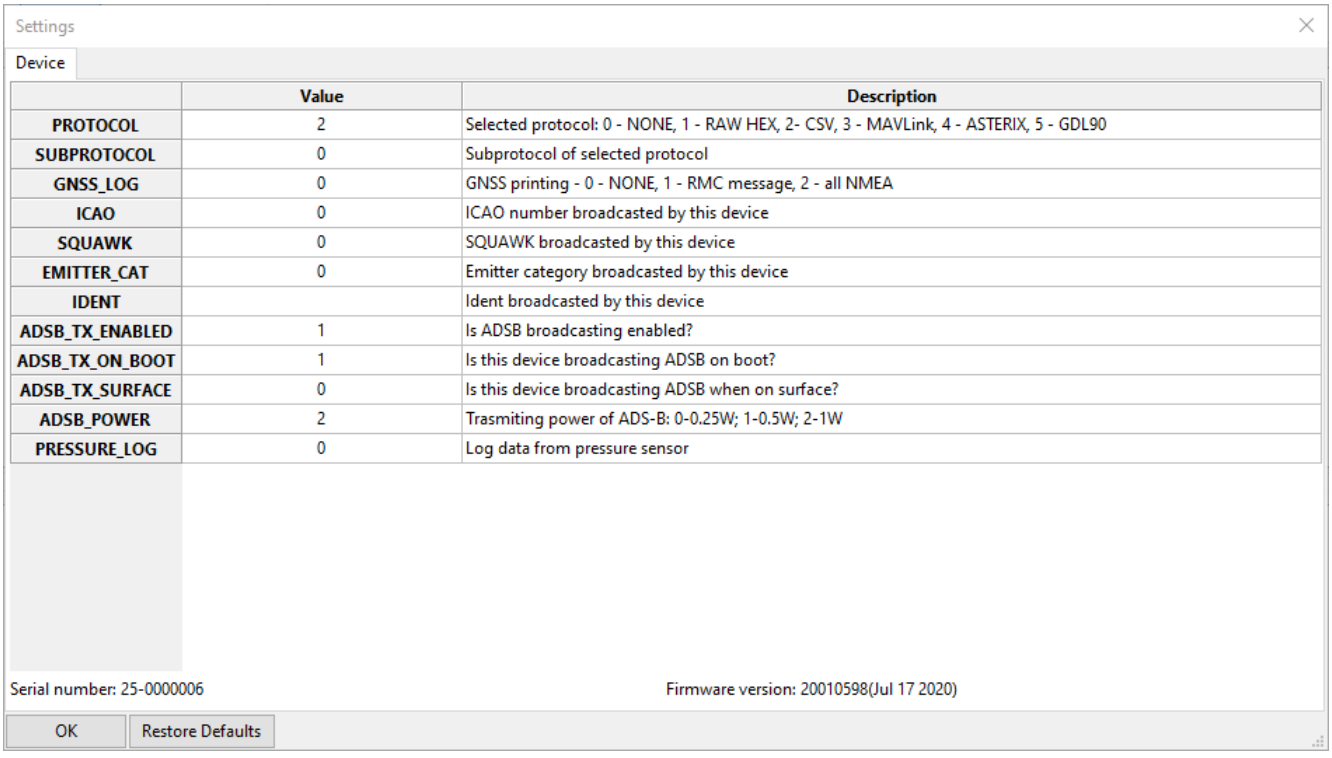


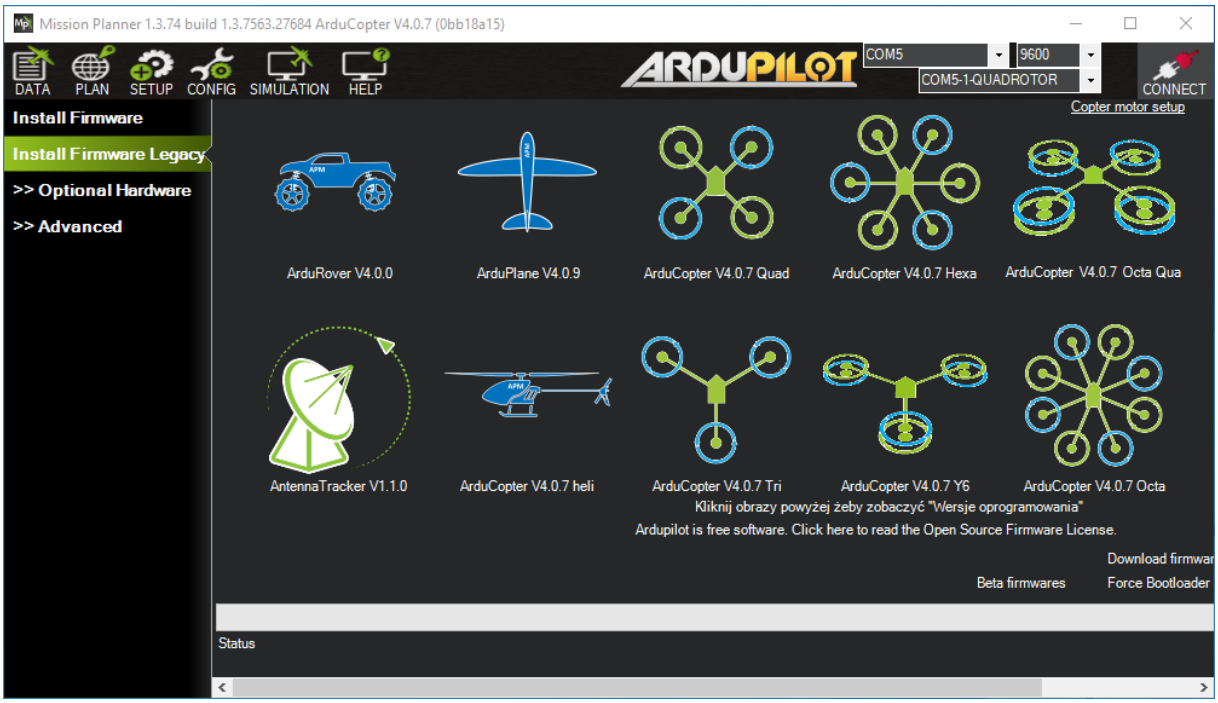
Figure 8: Settings window

7.6 Configuration with Pixhawk

NOTE: MAKE SURE YOU ARE USING MAVLINK PROTOCOL !!! Not all Mission Planner versions display ADS-B signals correctly. Make sure that your version of Mission Planer and Pixhawk is up to date.

7.6.1 Pixhawk update

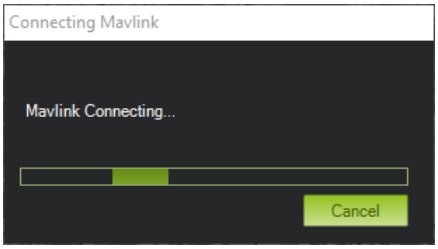
When installing via Mission Planner, disconnect by clicking the button, then select the appropriate firmware for your device in Install Firmware Legacy tab and follow the instructions. During installation by another method - compatible release is the newest version of **ArduPilot Flight Stack**.



7.6.2 Mission Planner

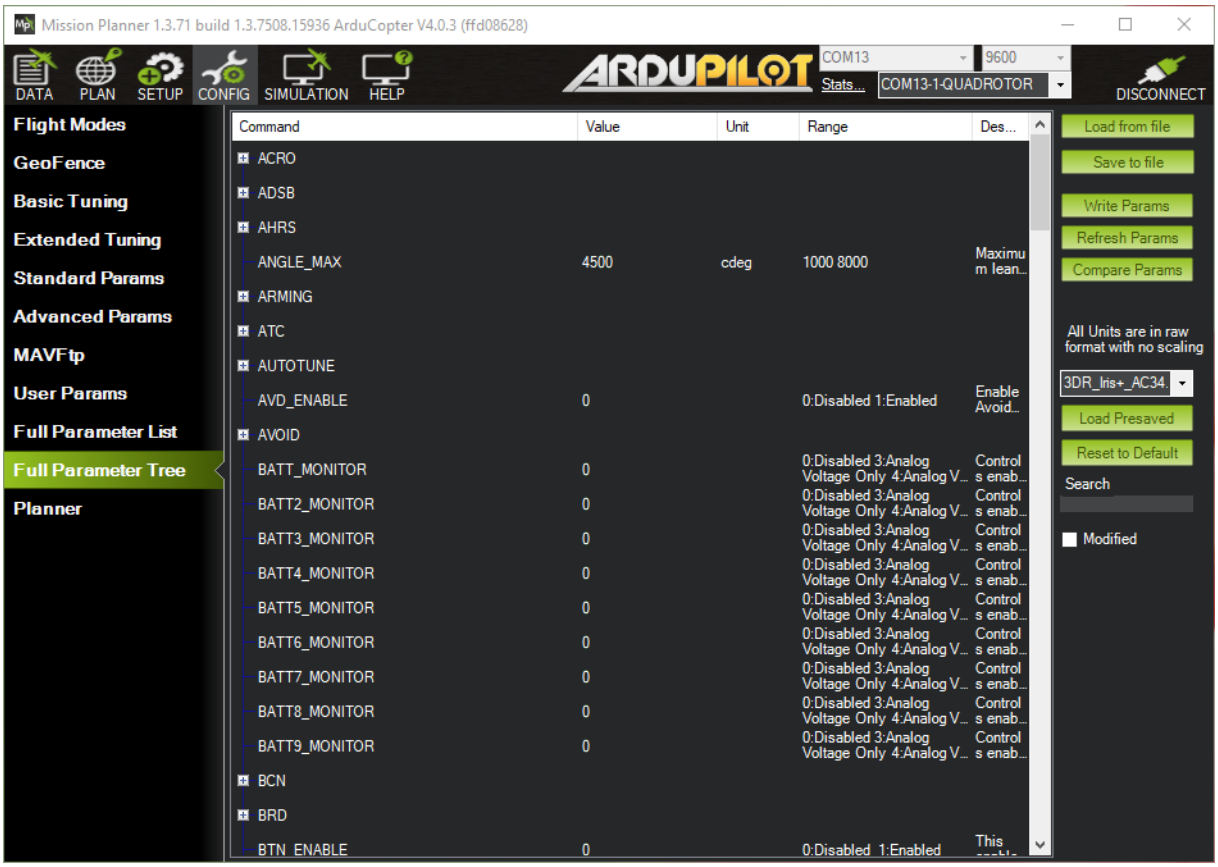
Five steps to integrate **TR1A** with Pixhawk4:

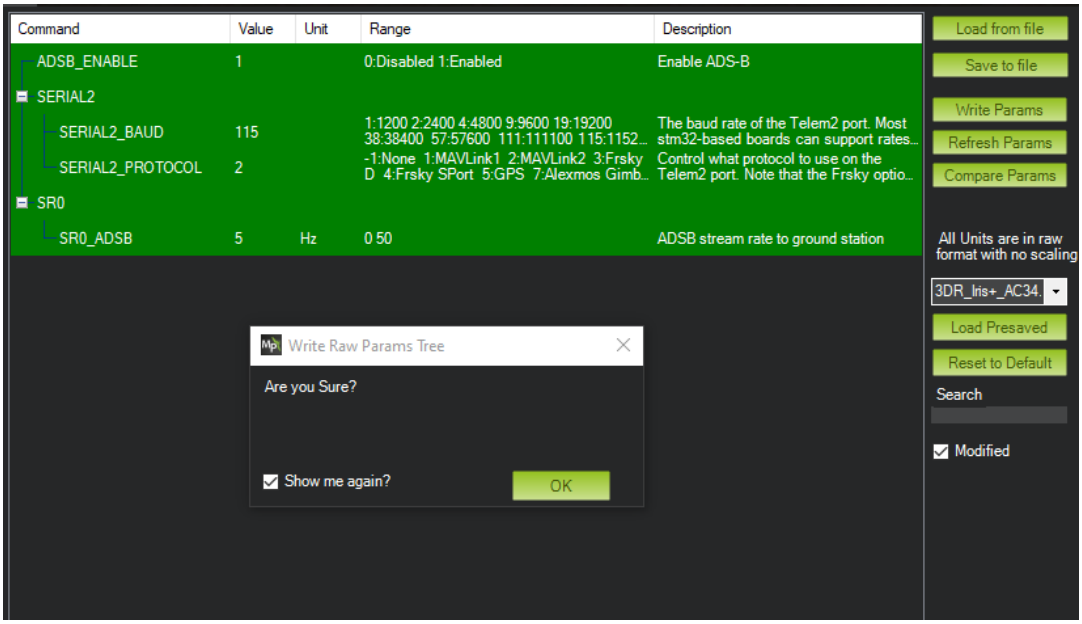
1. With the power turned off, connect TR1A to Pixhawk4 using a standard telemetry cable. The following settings apply to the installation on the TELEM2 port.
2. Connect the USB cable between Pixhawk4 and your PC and run Mission Planner.
3. Connect to Pixhawk4 by clicking ”Connect”, then go to the ”CONFIG” tab.



4. In the menu, go to ”Full Parameter Tree” and set the following parameters:

a.	ADSB	->	ADSB_ENABLE	1
b.	SERIAL2	->	SERIAL2_BAUD	115
		->	SERIAL2_PROTOCOL	2
c.	SR0	->	SR0_ADSB	range 1 to 50 Hz





NOTE: Remember to send the changed settings to the controller by clicking ”Write params”.

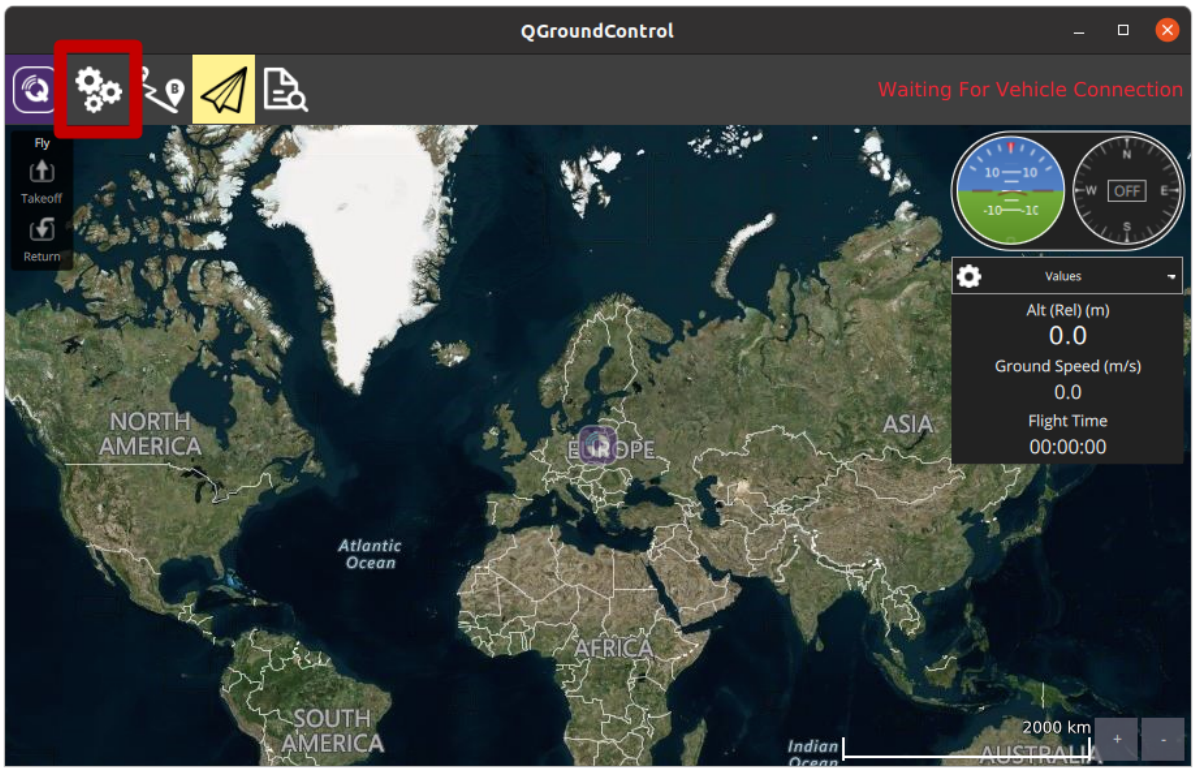
5. **Restart your Pixhawk** and go to the main view. If there is air traffic in your area, you should see it on the map.

For more information visit: ardupilot ADS-B documentation.

7.6.3 QGroundControl

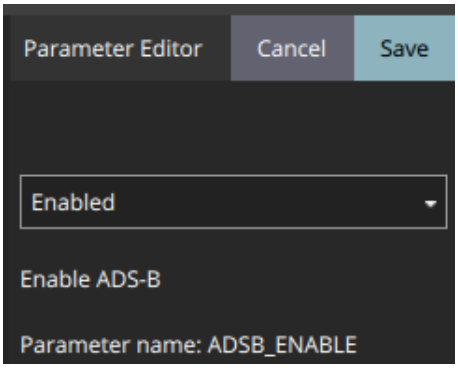
Mission Planer is a program designed for the Windows platform. QGroundControl is an alternative to Mission Planer with similar functionality. First steps are the same for both environments.

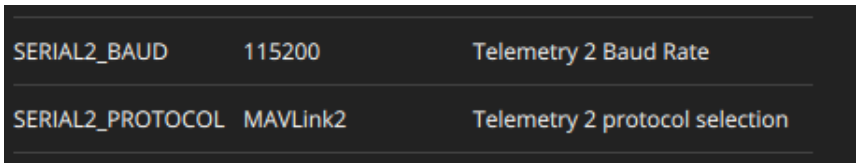
After connection device to Pixhawk4, the program should detect it.



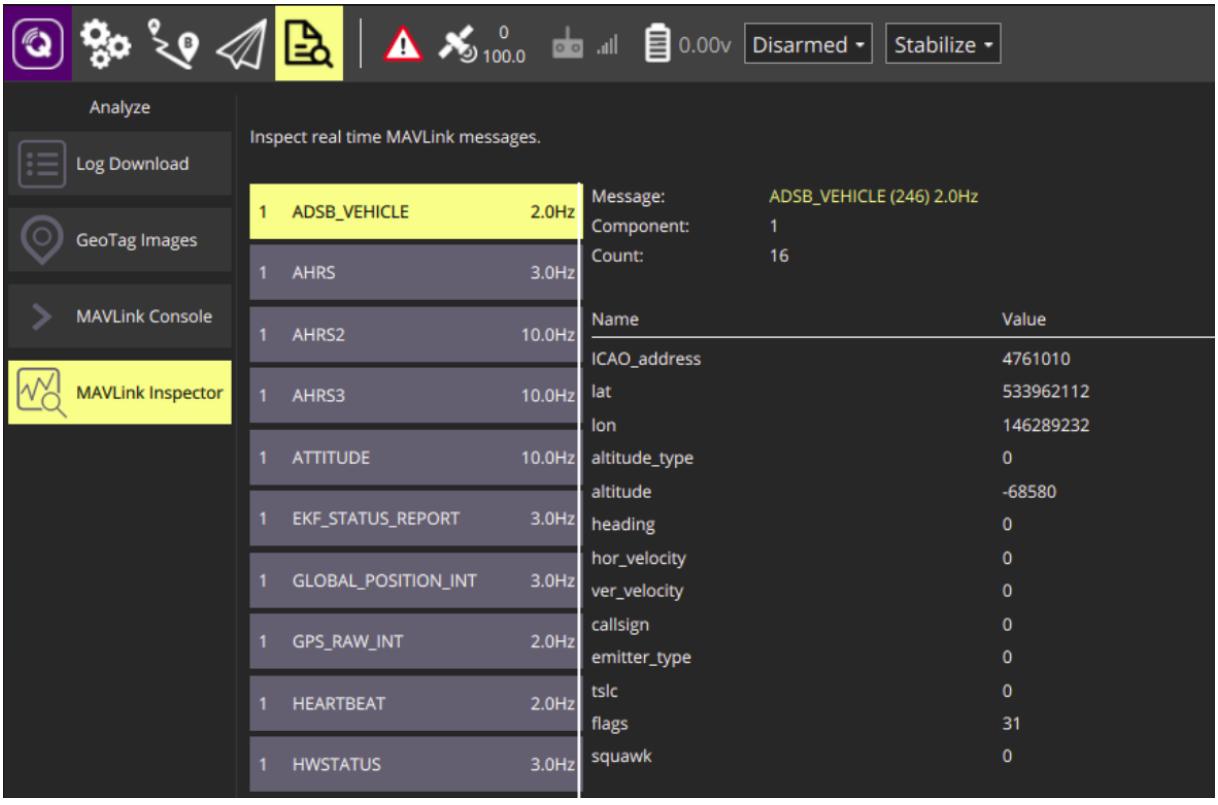
Set the parameters the same as in Mission Planer:

a.	ADSB	->	ADSB_ENABLED	
b.	SERIAL2	->	SERIAL2_BAUD	115200
		->	SERIAL2_PROTOCOL	MAVLink2
c.	SR0	->	SR0_ADSB	range 1 to 50 Hz





To make sure that the device receives the ADS-B signal correctly, you can check the MAVLink Inspector tab. Parameters like ADSB_VEHICLE and HEARTBEAT should be greater than 0 and count of received frames shoud have increasing tendency.



For more information visit: QGroundControl documentation.

8 General information

8.1 Module installation

There is a high concentration of various electronic systems on a small area at UAS. Try to keep as much separation between **TR1A** and other devices, especially radio ones. Despite the high robustness of **TR1A** to jamming, try to install the antenna away from other on-board systems.

8.2 AERO vs. MAVLink protocol

TR-1A is based on OEM TT-SG1a module. The default is in AERO protocol mode, which is an ASCII protocol. If you want to use the module to work with MAVLink system, it is possible to switch the protocol to MAVLink, which has the binary representation. Details of the module programming can be found on the website.

Please read carefully

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 PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT 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.