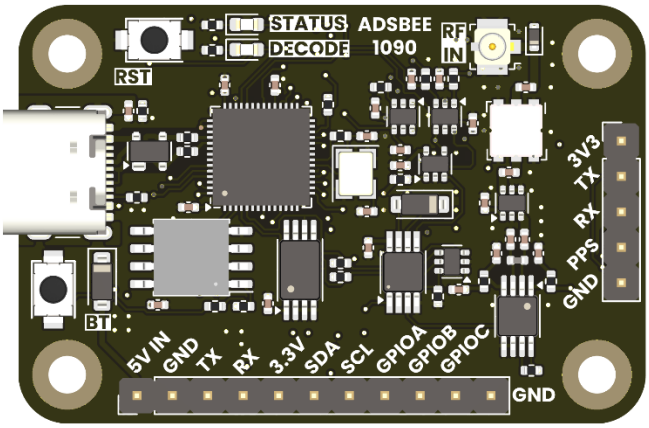
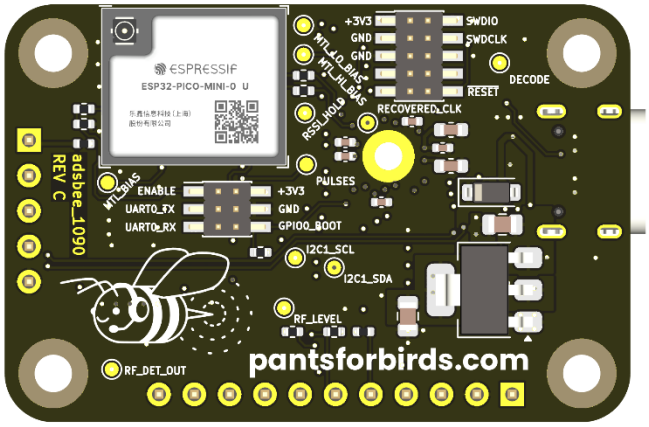
ADSBee 1090

Open Source Embedded ADS-B Receiver





Features

- 1090MHz Mode S and ADS-B packet decoding.
- Adjustable receive gain and trigger levels for customized tuning in diverse RF environments.
- Multiple output formats over UART or USB:
  - ADSBee CSV
  - MAVLink
  - GDL90 (not yet implemented)
  - More to come!
- Built-in EEPROM for storing configuration parameters in non-volatile memory.
- GNSS module input (UART + PPS) for MLAT or Remote ID applications.
- 2.4GHz 802.11 module for connecting directly to ADS-B databases via WiFi or broadcasting Remote ID beacon frames in UAS applications (not yet implemented).
- Integrated M2.5 mounting holes.
- Firmware updates over USB.

Applications

- Standalone feeder device for online ADS-B databases. No external compute required, just add power and WiFi!
- Aircraft detection for robotics and embedded projects.

Quick Specs

Supply Voltage	5V (Via USB or 5V pin)
Supply Current	75mA (WiFi disabled) ~400mA (WiFi enabled)
Minimum RF Input Power Level	-70dBm (not yet tested)
Simultaneous Aircraft Tracks Supported	≤100
Connectors	1090MHz RF In: U.FL / MHF1 802.11 RF Out: W.FL / MHF3 Power / Data: USB C GPIO / UART: 0.1” Pin Headers

Open Source Hardware + Software

Github Repository: https://github.com/coolnamesalltaken/ads-bee

All hardware design files and source code files required to build ADSBee 1090 are available under a GNU GPL v3 license. This means that they can be freely incorporated into other open-source projects that utilize a compatible license. The hope is that by opening the design to contributions and feedback from a community of users, the functionality of the ADSBee 1090 will be enhanced over time.

Note that the GPL v3 license applies to design and source code files, and not devices. ADSBee 1090 units purchased from Pants for Birds may be used in commercial applications without any licensing restrictions.

For commercial licensing requests of ADSBee hardware or software design files, please contact john@pantsforbirds.com.

Background

ADSBee 1090 was created as an attempt to build a low-cost ADS-B receiver without the use of an FPGA (found in most embedded ADS-B receivers on the market) and without the need for external compute (a requirement of most SDR-based ADS-B receivers). ADSBee 1090 accomplishes this through the use of a low-cost dual core microcontroller (RP2040) with flexible IO peripherals (PIO) that run a set of custom-written programs for preamble detection and packet decoding. By dedicating the RP2040's PIO peripherals to the tasks required to find and decode ADS-B packets, ADSBee 1090 frees up its remaining cores to perform the logical functions required to validate checksums on decoded packets and decipher aircraft information (position, altitude, callsign, etc).

Communication Interfaces

CONSOLE Interface

The CONSOLE interface (USB-C connector) on the ADSBee 1090 can be used to supply the device with power, configure the device's internal parameters via AT commands, and receive data from the device.

No baud rate configuration is necessary for the CONSOLE interface. Note that AT commands must be suffixed with CR+LF ("\r\n") in order to be processed by the AT command parser.

COMMS_UART Interface

The COMMS_UART interface is used for data output, and can support a number of different protocols. Data can be streamed out of the CONSOLE and COMMS_UART interfaces simultaneously. The baud rate of the COMMS_UART interface can be adjusted via AT commands.

COMMS_UART Parameters

Parameter	Value
Baud Rate	115200 baud (default)
Data Bits	8
Stop Bits	0 (N)
Parity Bits	1
Logic Level	3.3V

GNSS_UART Interface

The ADSBee 1090's GNSS module connector includes a UART interface which can be used to receive NMEA sentences from a GNSS module. The baud rate of the GNSS_UART interface can be adjusted via AT commands.

GNSS_UART Parameters

Parameter	Value
Baud Rate	9600 baud (default)
Data Bits	8
Stop Bits	0 (N)
Parity Bits	1
Logic Level	3.3V

AT Commands

AT Commands are used to configure the ADSBee 1090 receiver's internal parameters via the CONSOLE interface.

All AT command arguments are optional. Arguments will be ignored if left as blank or whitespace. For instance, to set the second parameter of AT+TL_SET to 3000 without changing the value of the first parameter, the command "AT+TL_SET=,3000" can be sent. Likewise, to change the first parameter to 2000 without changing the value of the second, the command "AT+TL_SET=2000," can be sent.

Command	Parameters
AT+LOG_LEVEL	**Log Level Command**
AT+TL_SET *Write with echo of values that were set.* AT+MTLSET=<tl_lo_mv:uint16_t>,<tl_hi_mv:uint16_t> +MTLSET=<tl_lo_mv:uint16_t>,<tl_hi_mv:uint16_t> *Read present set value (stored setpoint, not read by ADC).* AT+MTLSET? +MTLSET=<tl_lo_mv:uint16_t>,<tl_hi_mv:uint16_t>	**RF Comparator Trigger Level (TL) Setpoint Command** NOTE: tl_lo_mv should be set to a value lower than tl_hi_mv. Reducing the level of tl_lo_mv will make the receiver more sensitive to weak RF signals, but will also increase the noise that it receives. Increasing the difference between tl_lo_mv and tl_hi_mv will filter out signals with smaller dynamic range (difference in power level between max amplitude and min amplitude), thereby requiring a higher Signal to Noise ratio for a transponder signal to be decoded. This may reduce the likelihood that the ADSBee tries to decode a transponder signal with invalid bits that will trigger a checksum error. tl_lo_mv: TL Low Threshold [milliVolts] • 0-3300 = Low-side trigger threshold of the comparator circuit on the output of the RF detector. Refer to the AD8313 datasheet and adjustable gain stuff for a conversion from mV (RF detector output signal amplitude) to dBm (RF signal power level in). tl_hi_mv: TL High Threshold [milliVolts] • 0-3300 = High-side trigger threshold of the comparator circuit.
AT+TL_READ *Read with echo of values that were read.* AT+MTLREAD +MTLREAD=<tl_lo_mv>,<tl_hi_mv>	**RF Comparator Trigger Level (TL) Read Command** Used an ADC to read the value of tl_lo_mv and tl_hi_mv. Should be roughly in line with the values of tl_lo_mv and tl_hi_mv set in the AT+TL_SET section.
AT+HELP AT+HELP <command>:<command help string>	**Help Command** Prints out a list of available commands and their associated help strings.

<command>:<command help string> <…>	
AT+RX_GAIN *Set gain to 100x with echo of gain value that was set.* AT+RX_GAIN=100 +RX_GAIN=100 *Read gain value.* AT+RX_GAIN? +RX_GAIN=100 *Test +RX_GAIN command.* +RX_GAIN=<gain:uint16_t>	**Receiver Gain Command** Adjust the gain of the operational amplifier located after the AD8313 in the receive signal chain. Gain is set as a positive integer value between 1-101. gain: Receiver Gain [ratio] • 1-101 = Gain value of operational amplifier operating on AD8313 output.

Reporting Protocols

ADSBee 1090 supports the following reporting protocols on CONSOLE and COMMS_UART.

- CSBEE_PACKETS
- CSBEE_AIRCRAFT
- MAVLINK
- GDL90 (not yet implemented)

CSBEE_PACKETS

Comma Separated Bee protocol containing raw packet information as plain text. Data is sent as packets are received.

CSBEE_AIRCRAFT

Comma Separated Bee protocol containing information about tracked aircraft as plain text.

MAVLINK

Tracked aircraft information is sent in MAVLINK ADSB_VEHICLE messages, in a data burst once per second. The data burst consists of Nx ADSB_VEHICLE messages, where N is the number of tracked aircraft, and 1x MESSAGE_INTERVAL message as a delimiter which indicates the end of the list of tracked aircraft. Note that this is a binary protocol which is not human readable.

WARNING: Windows has a bug, which causes some machines to recognize a serial port reporting MAVLINK packets as a mouse, which can result in phantom mouse movements and clicks (ask me how I know). Please use caution while running MAVLINK on a serial port while the computer is unattended.

MAVLINK ADSB_VEHICLE (Message ID 246) Packet Definition

From: https://mavlink.io/en/messages/common.html#ADSB_VEHICLE

Field Name	Type	Units	Values	Description
ICAO_address	`uint32_t`			ICAO address
lat	`int32_t`	degE7		Latitude
lon	`int32_t`	degE7		Longitude
altitude_type	`uint8_t`		ADSB_ALTITUDE_TYPE	ADSB altitude type.
altitude	`int32_t`	mm		Altitude(ASL)
heading	`uint16_t`	cdeg		Course over ground
hor_velocity	`uint16_t`	cm/s		The horizontal velocity
ver_velocity	`int16_t`	cm/s		The vertical velocity. Positive is up
callsign	`char[9]`			The callsign, 8+null
emitter_type	`uint8_t`		ADSB_EMITTER_TYPE	ADSB emitter type.
tslc	`uint8_t`	s		Time since last communication in seconds
flags	`uint16_t`		ADSB_FLAGS	Bitmap to indicate various statuses including valid data fields
squawk	`uint16_t`			Squawk code

MAVLINK MESSAGE_INTERVAL (Message ID 244) Packet Definition

From: https://mavlink.io/en/messages/common.html#MESSAGE_INTERVAL

Field Name	Type	Units	Description
message_id	`uint16_t`		The ID of the requested MAVLink message. v1.0 is limited to 254 messages. NOTE: For ADSBee 1090, message_id is always 246, corresponding to the ADSB_VEHICLE message.
interval_us	`int32_t`	us	The interval between two messages. A value of -1 indicates this stream is disabled, 0 indicates it is not available, > 0 indicates the interval at which it is sent. NOTE: For ADSBee 1090, message_id is always 1000 us.