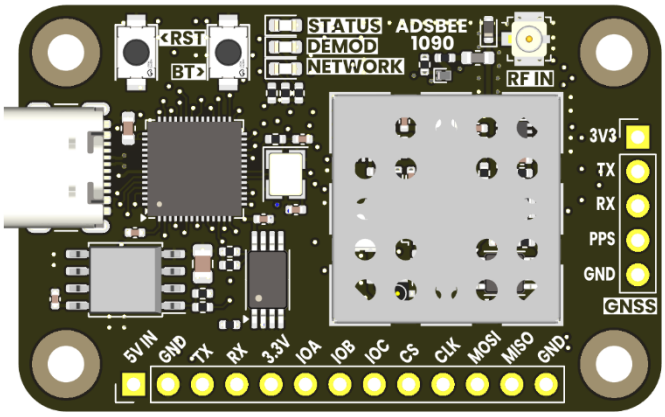
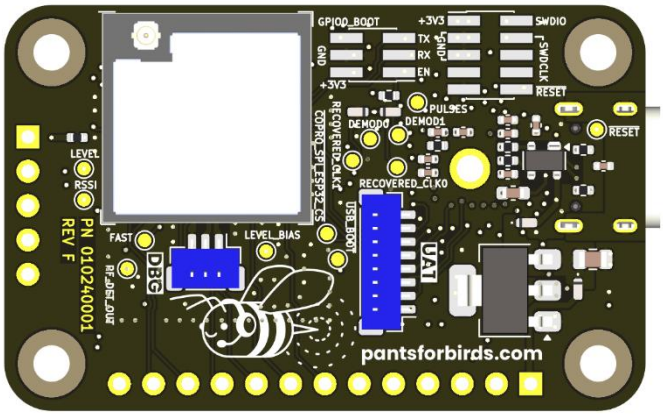
ADSBee 1090

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:
  - Raw Packets
  - ADSBee CSV
  - MAVLink1 / MAVLink2
  - GDL90
  - Mode S Beast
- 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	120mA (WiFi disabled) ~400mA (WiFi enabled)
Minimum RF Input Power Level	-70dBm
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 schematics 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.

Contents

1 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).

The ADSBee 1090 reports decoded aircraft information over UART and USB interfaces in a variety of protocols, and can utilize its attached ESP32 S3 module to host WiFi networks for data streaming to other devices, or connect to existing WiFi networks in order to upload data to the internet or other devices on the network.

Provisions are included for connecting an external GNSS module and UAT radio receiver to the ADSBee 1090. These features will enter development in the near future.

2 Communication Interfaces

2.1 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.

2.2 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

2.3 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

3 AT Commands

AT Commands are used to configure the ADSBee 1090 receiver's internal parameters via the CONSOLE interface. AT commands can be used to set values (with the '=' operator, e.g. "AT+BAUD_RATE=COMMS_UART,115200") or query values (with the '?' operator, e.g. "AT+BAUD_RATE?").

AT commands and parameters must be all-caps. Whitespace is not ignored and should only be used intentionally (e.g. as part of a WiFi network SSID which contains spaces). Arguments are separated by a single comma and no spaces.

All AT command arguments are optional. Arguments will be ignored if left as blank.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.

Commands from the user to the ADSBee 1090 are prefixed with "AT+" (e.g. "AT+BAUD_RATE…"). Replies may be prefixed with the relevant command (e.g. "+BAUD_RATE=…") or nothing (e.g. "OK"). Replies to AT command queries may include text enclosed by parentheses (e.g. "AT+BAUD_RATE=115200(COMMS),9600(GNSS). These text blobs are annotations to improve human readability of the reply.

IMPORTANT NOTE

No settings will be persisted after shutdown unless `AT+SETTINGS=SAVE` is executed in order to write new settings values to non-volatile storage.

3.1 AT+BAUD_RATE

3.1.1 Set Baud Rate

```
AT+BAUD_RATE=<iface>,<baud_rate>
```

Set the baud rate for a particular serial interface. Note that the USB C port (CONSOLE interface) is a virtual COM port and does not have a baud rate that can be changed.

Parameter	Description	Values
iface	Serial interface to set the baud rate for.	COMMS_UART, GNSS
baud_rate	Baud rate, in characters per second.	9600-115200

3.1.2 Query Baud Rate

```
AT+BAUD_RATE?
+BAUD_RATE=<comms_uart_baud_rate>(COMMS_UART),<gnss_uart_baud_rate>(GNSS_UART)
```

Queries the baud rate for the COMMS_UART and GNSS_UART interfaces. Note that the baud rates returned by this query are the actual system baud rates, which should be close to the baud rates that were requested with AT+BAUDRATE but may be slightly different due to the particulars of how baud rates are implemented inside the RP2040.

Parameter	Description	Values

comms_uart_baud_rate	Baud rate on the COMMS_UART interface.	9600-115200
gnss_uart_baud_rate	Baud rate on the GNSS_UART interface.	9600-115200

3.2 AT+BIAS_TEE_ENABLE

3.2.1 Turn the Bias Tee On or Off

```
AT+BIAS_TEE_ENABLE=<enabled>
```

Enable or disable the bias tee to enable use of an external LNA.

Parameter	Description	Acceptable Values
enabled	Whether the bias tee is enabled (3.3VDC supplied to the RF IN connector) or disabled (no DC voltage supplied to the RF IN connector).	0 – Bias tee is disabled. 1 – Bias tee is enabled.

3.2.2 Query the Status of the Bias Tee

```
AT+BIAS_TEE_ENABLE?
+BIAS_TEE_ENABLE=<enabled>
```

Queries the status of the bias tee. See table in 3.1.1 for values and their meanings.

3.3 AT+DEVICE_INFO

Queries the device information of the ADSBee. An example query and reply is shown below.

```
> AT+DEVICE_INFO?
Part Code: 010240002F-20240907-000001
RP2040 Firmware Version: 0.6.2
OTA Key 0:
cf737c4fce892fab44c6ec0d5d0f66dccf06c5630ac5b531f4067599df014d5ef213bb73205a9ce37
8318f259e3d01e3b99c089a75ac0735b204256b9371fbce
OTA Key 1:
1701a736dc40c837f318c9e7ae82557a39fe07588d3745602bb9269b13843b87ca7962daecc9897c6
dfb0ac4ec77997e9cc34cd356e415fc46eb4ae3dd42b717
ESP32 Firmware Version: 0.6.2
ESP32 Base MAC Address: 64:E8:33:5D:0B:C4
ESP32 WiFi Station MAC Address: 64:E8:33:5D:0B:C4
ESP32 WiFi AP MAC Address: 64:E8:33:5D:0B:C5
ESP32 Bluetooth MAC Address: 64:E8:33:5D:0B:C6
ESP32 Ethernet MAC Address: 64:E8:33:5D:0B:C7
```

3.4 AT+ESP32_ENABLE

3.4.1 Turn the ESP32 On or Off

```
AT+ESP32_ENABLE=<enabled>
```

Turns the ESP32 on or off using its enable line. When the ESP32 is turned off, the ADSBee 1090 will draw less power, but the WiFi and network capabilities will be unavailable. This operational mode is useful for conserving energy in applications where the ADSBee is only being communicated with via its serial interfaces.

Parameter	Description	Acceptable Values
enabled	Whether the ESP32 is turned on.	0 – ESP32 is turned on. 1 – ESP32 is turned off.

3.4.2 Query the Status of the ESP32

```
AT+ESP32_ENABLE?
+ESP32_ENABLE=<enabled>
```

Queries whether the ESP32 is turned on or off. See table in 3.4.1 for the values and their meanings.

3.5 AT+FEED

3.5.1 Configure a Feed

```
AT+FEED=<index>,<uri>,<port>,<active>,<protocol>
```

Example command for configuring feed 0 to send data to airplanes.live, which accepts Mode S Beast protocol on port 30004 at the static IP 78.46.234.18:

```
AT+FEED=0,78.46.234.18,30004,1,BEAST
```

Parameter	Description	Acceptable Values
index	Which feed slot to configure.	0-5
uri	Static IP address or hostname of the feed destination. If the field is given as a hostname, a DNS lookup will be performed in order to convert the hostname to an IP address.	Public IP address, e.g. 78.46.234.18 Hostname, e.g. feed.whereplane.xyz Maximum length 64 characters.
port	Port number to feed to.	0-65535
active	Whether the feed should be activated. This can be used to temporarily disable a feed while still keeping its configuration information intact.	0 – Feed is active. 1 – Feed is disabled.
protocol	Which protocol to use when feeding.	BEAST – Send validated packets in Mode S Beast format. BEAST_RAW – Send both validated and invalid (bad checksum) packets in Mode S Beast format.

1.1.1 Query Configuration of All Feeds

```
AT+FEED?
+FEED=0(INDEX),192.168.1.103(URI),30004(PORT),1(ACTIVE),BEAST(PROTOCOL),bee00038172d18c(RECEIVER_ID)
+FEED=1(INDEX),(URI),0(PORT),0(ACTIVE),NONE(PROTOCOL),bee00038172d18c(RECEIVER_ID)
+FEED=2(INDEX),(URI),0(PORT),0(ACTIVE),NONE(PROTOCOL),bee00038172d18c(RECEIVER_ID)
+FEED=3(INDEX),(URI),0(PORT),0(ACTIVE),NONE(PROTOCOL),bee00038172d18c(RECEIVER_ID)
+FEED=4(INDEX),78.46.234.18(URI),30004(PORT),1(ACTIVE),BEAST(PROTOCOL),bee00038172d18c(RECEIVER_ID)
+FEED=5(INDEX),feed.whereplane.xyz(URI),30004(PORT),0(ACTIVE),BEAST(PROTOCOL),bee00038172d18c(RECEIVER_ID)
```

1.1.2 Query Configuration of a Single Feed

```
AT+FEED?<index>
```

Example query for configuration of feed 0 only:

```
AT+FEED?0
+FEED=0(INDEX),192.168.1.103(URI),30004(PORT),1(ACTIVE),BEAST(PROTOCOL),bee00038172d18c(RECEIVER_ID)
```

1.2 AT+FLASH_ESP32

```
AT+FLASH_ESP32
```

Flashes the ESP32 with the firmware image stored on the RP2040. This should never be needed, since the RP2040 does a firmware version check of the ESP32 on startup and automatically reflashes he firmware if it is out of date. The AT+FLASH_ESP32 command is only used in weird debugging scenarios where the version number of the firmware on the ESP32 matches the version of the firmware stored on the RP2040, but the firmware stored on the RP2040 is different and needs to be flashed onto the ESP32.

1.3 AT+HELP

```
AT+HELP
AT Command Help Menu:
+BAUDRATE:
        AT+BAUDRATE=<iface>,<baudrate>
        Set the baud rate of a serial interface.
        AT_BAUDRATE?
        Query the baud rate of all serial interfaces.
+BIAS_TEE_ENABLE:
        AT+BIAS_TEE_ENABLE=<enabled>
        Enable or disable the bias tee.
        BIAS_TEE_ENABLE?
        Query the status of the bias tee.
...
```

Prints the available AT commands as well as how to use them. The list of commands in the preview above is abridged.

1.4 AT+HOSTNAME

1.4.1 Set the Device Hostname

```
AT+HOSTNAME=<hostname>
```

Sets the device hostname, which can make things easier when configuring a local network. Hostname is also used for mDNS, allowing you to visit the device's webpage at <hostname>.local.

1.4.2 Query the Device Hostname

```
AT+HOSTNAME?
+HOSTNAME=ADSBee1090-0240907000001
```

1.5 AT+LOG_LEVEL

```
AT+LOG_LEVEL=<log_level>
```

Parameter	Description	Acceptable Values
log_level	Log verbosity level for the console. Anything being printed with a lower verbosity than the specified level will be suppressed. A verbosity level of AT+LOG_LEVEL=WARNINGS is recommended. Terminal prints are color coded on the Serial CLI (but not on the Web CLI). Errors are printed in red, warnings are printed in yellow, and other prints have no color coding.	INFO – Informational logs. Details every packet received by the ADSBee and whether it passed checksum. Also prints all warnings and errors. WARNINGS – Warning logs. Prints out every time something unexpected happens. Warnings include recoverable errors and unexpected values encountered within ADS-B packets being decoded. Also prints all errors. ERRORS – Error logs. Prints every time there is a severe error, such as packets being lost in communication between the RP2040 and ESP32, a peripheral failing to communicate properly, or fatal errors in code. SILENT – No logs. Only AT command replies and console protocols will be printed. Use this setting to avoid having log messages injected into the console text stream if it is being used to report a protocol to another device.

1.6 AT+PROTOCOL

1.6.1 Set the Reporting Protocol for a Serial Interface

```
AT+PROTOCOL=<iface>,<protocol>
```

Parameter	Description	Values
iface	Serial interface to set the baudrate for.	CONSOLE, COMMS_UART
protocol	Reporting protocol for the specified serial interface	NONE – No data reported. RAW – Raw plain text reporting protocol. BEAST – Mode S Beast binary reporting protocol. CSBEE – CSBEE plain text reporting protocol. MAVLINK1 – MAVLINK 1 binary reporting protocol. MAVLINK2 – MAVLINK 2 binary reporting protocol. GDL90 – Garmin GDL90 binary reporting protocol.

1.6.2 Query the Reporting Protocol for All Serial Interfaces

```
AT+PROTOCOL?
+PROTOCOL=CONSOLE,<console_protocol>
+PROTOCOL=COMMS_UART,<comms_uart_protocol>
```

Queries the reporting protocol set for each of the serial interfaces. See the table in 1.6.1 for protocol definitions.

1.7 AT+REBOOT

```
AT+REBOOT
```

Reboots the ADSBee 1090 immediately.

1.8 AT+RX_ENABLE

1.8.1 Enable or Disable the Receiver

```
AT+RX_ENABLE=<enabled>
```

Parameter	Description	Values
enabled	Flag indicating whether the receiver should be enabled or disabled. NOTE: Disabling the receiver does not conserve any power, as the RF frontend is still active. This receiver disable flag simply disables an interrupt within the RP2040 that is used to check for incoming transponder messages.	1 – Receiver is enabled. 0 – Receiver is disabled.

1.8.2 Check Whether the Receiver is Currently Enabled

```
AT+RX_ENABLE?
+RX_ENABLE=1
```

1.9 AT+SETTINGS

1.9.1 Query Current Settings

```
AT+SETTINGS?
Settings Struct
        Receiver: ENABLED
        Trigger Level: 1300 milliVolts (-62 dBm)
        Bias Tee: DISABLED
        Watchdog Timeout: 10 seconds
        Log Level: WARNINGS
        Reporting Protocols:
                CONSOLE: NONE
                COMMS_UART: CSBEE
        Comms UART Baud Rate: 115207 baud
        GNSS UART Baud Rate: 9600 baud
        ESP32: ENABLED
        WiFi AP: ENABLED
                Channel: 3
                SSID: ADSBee1090-0240907000001
                Password: yummyflowers
        WiFi STA: ENABLED
                SSID: Smart Bidet Wifi
                Password: ********
        Feed URIs:
                0 URI:192.168.1.103 Port:30004 ACTIVE Protocol:BEAST
ID:0xbee00038172d18c1
                1 URI: Port:0 INACTIVE Protocol:NONE ID:0xbee00038172d18c1
                2 URI: Port:0 INACTIVE Protocol:NONE ID:0xbee00038172d18c1
                3 URI: Port:0 INACTIVE Protocol:NONE ID:0xbee00038172d18c1
                4 URI:78.46.234.18 Port:30004 ACTIVE Protocol:BEAST ID:0xbee00038172d18c1
                5 URI:feed.whereplane.xyz Port:30004 INACTIVE Protocol:BEAST
ID:0xbee00038172d18c1
OK
```

1.9.2 Dump Settings in AT Format

Sending AT+SETTINGS?DUMP outputs a dump of all nonvolatile settings in AT command format, allowing them to be easily re-entered when restoring the ADSBee settings after a firmware upgrade, or for moving setting between devices.

```
AT+SETTINGS?DUMP
AT+SETTINGS=RESET
AT+BAUD_RATE=CONSOLE,115200
AT+BAUD_RATE=COMMS_UART,115200
AT+BAUD_RATE=GNSS_UART,115200
AT+BIAS_TEE_ENABLE=0
AT+ESP32_ENABLE=1
AT+FEED=0,,0,0,NONE
AT+FEED=1,,0,0,NONE
AT+FEED=2,,0,0,NONE
AT+FEED=3,,0,0,NONE
AT+FEED=4,feed.airplanes.live,30004,1,BEAST
AT+FEED=5,feed.adsb.fi,30004,1,BEAST
AT+LOG_LEVEL=WARNINGS
AT+PROTOCOL=CONSOLE,NONE
AT+PROTOCOL=COMMS_UART,MAVLINK1
AT+RX_ENABLE=1
AT+TL_SET=1300
AT+WATCHDOG=10
AT+WIFI_AP=1,ADSBee1090-0240907000001,yummyflowers,11
AT+WIFI_STA=0,,
OK
```

1.9.3 Load Settings from EEPROM

```
AT+SETTINGS=LOAD
```

1.9.4 Save Settings to EEPROM

```
AT+SETTINGS=SAVE
```

Note that this command is also used to synchronize settings between the RP2040 and ESP32. When changing parameters in the Serial CLI or Web CLI, run `AT+SETTINGS=SAVE` or reboot the device to synchronize new settings (e.g. new feed parameters or network settings).

1.9.5 Reset Settings to Factory Default

```
AT+SETTINGS=RESET
```

Note that this command resets settings to factory defaults, but will not persist them to the next power on unless `AT+SETTINGS=SAVE` is run subsequently.

1.10 AT+TEST

Used for running internal self-tests. Can only be run once per boot cycle.

```
AT+TEST
[==========] Running 4 test cases.
[ RUN      ] SpiCoprocessor.WriteReadScratchNoAck
[       OK ] SpiCoprocessor.WriteReadScratchNoAck (687000ns)
[ RUN      ] SpiCoprocessor.WriteReadScratchWithAck
[       OK ] SpiCoprocessor.WriteReadScratchWithAck (751000ns)
[ RUN      ] SPICoprocessor.ReadWriteReadRewriteRereadBigNoAck
[       OK ] SPICoprocessor.ReadWriteReadRewriteRereadBigNoAck (25336000ns)
[ RUN      ] SPICoprocessor.ReadWriteReadRewriteRereadBigWithAck
[       OK ] SPICoprocessor.ReadWriteReadRewriteRereadBigWithAck (29287000ns)
[==========] 4 test cases ran.
[  PASSED  ] 4 tests.
```

1.11 AT+TL_READ

```
AT+TL_READ?
+TL_READ=1258mV (-64 dBm)
```

Queries the current value of the internal trigger level of the RF frontend's data slicer circuit. Signals peaking above this level will trigger a demodulation. This value should be set substantially above the noise floor in order to avoid false triggers.

Note that the value returned from TL_READ is different from querying TL_SET, since the value returned by TL_READ is read from the RP2040's ADC, while the value returned from querying TL_SET is the setpoint for the PWM output that generates the trigger level signal. They should be roughly similar, but slightly offset.

1.12 AT+TL_SET

1.12.1 Set Trigger Level

```
AT+TL_SET=<command>
```

Parameter	Description	Values
command	Trigger level, in milliVolts, or the activation word to begin a self-learning process for the trigger level using simulated annealing.	0-3300 LEARN

The `AT+TL_SET` command sets the value of the internal trigger level of the RF frontend's data slicer circuit. Signals peaking above this level will trigger a demodulation. This value should be set substantially above the noise floor in order to avoid false triggers.

A higher trigger level makes the receiver less sensitive, while a lower trigger level makes the receiver more sensitive (and thus more likely to be triggered by unwanted noise).

A self-learning process can be activated by sending `AT+TL_SET=LEARN` instead of providing a trigger value in milliVolts. The self-learning process uses simulated annealing, where a random trigger value is chosen, and then random neighbors are traversed to if their trigger levels seem to provide an improved number of valid packets within the learning window. The allowable traversal distance is gradually decreased over time until a final value is settled upon. The self-learning process takes approximately 20 minutes. Note that as of the current firmware revision, hand-tuning the trigger value (or using the provided default value) may outperform the self-learning routine.

1.12.2 Query Trigger Level

```
AT+TL_SET?
+TL_SET=1300mV (-62 dBm)
```

Queries the trigger level that is set internally. Note that this may differ from the actual value that is read using an ADC, as provided by `AT+TL_READ`.

1.13 AT+UPTIME

Queries the uptime of the ADSBee, in seconds.

```
AT+UPTIME?
+UPTIME=71
```

1.14 AT+WATCHDOG

The ADSBee features a watchdog timer on the RP2040, which will automatically reset all microcontrollers on the board if code executing on the RP2040 freezes up, or if SPI communication with the ESP32 breaks down. This is intended to provide a means of automatic recovery in case of a hard fault caused by firmware bugs (likely), power glitches (maybe), or cosmic rays (probably not). As the watchdog may not be wanted in some cases, like debugging hardfaults, or during device programming / beakpoint debugging, it can be disabled. Additionally, means are provided for self-testing the watchdog via AT command in order to verify that it is still active.

1.14.1 Configure the Watchdog Timer

```
AT+WATCHDOG=<timeout_sec>
```

Parameter	Description	Values
command	Trigger level, in milliVolts, or the activation word to begin a self-learning process for the trigger level using simulated annealing.	0-3300 LEARN

1.14.2 Query the Watchdog Timer

```
AT+WATCHDOG?
+WATCHDOG=10
```

Prints the value of the watchdog timer in seconds, or 0 if the watchdog timer is disabled.

1.14.3 Test the Watchdog Timer

```
AT+WATCHDOG=TEST
Blocking for 11 seconds.
```

(ADSBee Reboots here)

The AT+WATCHDOG=TEST command blocks the processor for the currently configured watchdog timeout plus one second. This should induce a device reboot, otherwise an error message will be displayed indicating that the watchdog failed to activate.

1.15 AT+WIFI_AP

The ADSBee creates its own WiFi access point that allows other devices to join its network in order to access the Web CLI and receive aircraft data (e.g. GDL90 traffic info over UDP port 4000). The WiFi AP can be enabled/disabled and can have its SSID, password, and WiFi channel customized. Note that WiFi settings don't have any affect unless the ESP32 is enabled, and are not propagated to the ESP32 unless settings are saved with `AT+SETTINGS=SAVE`.

1.15.1 Configure the WiFi Access Point

```
AT+WIFI_AP=<enabled>,<ap_ssid>,<ap_pwd>,<ap_channel>
```

Parameter	Description	Values
enabled	Toggle whether the AP is enabled or disabled.	0 – Disable WiFi access point. 1 – Enable WiFi access point.
ap_ssid	SSID for the access point. Defaults to a unique SSID with the format "ADSBee1090-0240907000001". Can be set to a value up to 32 characters long.	ASCII string up to 32 characters in length.
ap_pwd	Password for the access point. Defaults to "yummyflowers". Can be set to a value up to 64 characters long.	ASCII string up to 64 characters in length.
ap_channel	WiFi channel number for the access point. Can be set to a value from 1-11. A random channel value is chosen by default during initial programming of the device. It is recommended to change the channel value to a frequency with low utilization in the installation location.	1-11

1.15.2 Query the WiFi Access Point Configuration

```
AT+WIFI_AP?
+WIFI_AP=1,ADSBee1090-0240907000001,yummyflowers,3
```

Reply format matches argument values and ordering from the configuration command.

1.16 AT+WIFI_STA

The ADSBee can join external WiFi networks as a WiFi station in order to allow connections from other devices on the existing network, or to connect to the internet in order to feed remote endpoints with air traffic data.

1.16.1 Configure the WiFi Station

```
AT+WIFI_STA=<enabled>,<sta_ssid>,<sta_pwd>
```

Parameter	Description	Values
enabled	Toggle whether the AP is enabled or disabled.	0 – Disable WiFi station. 1 – Enable WiFi station.
sta_ssid	SSID for the access point to connect to. Can be set to a value up to 32 characters long.	ASCII string up to 32 characters in length.
sta_pwd	Password for the access point to connect to. Can be set to a value up to 64 characters long.	ASCII string up to 64 characters in length.

1.16.2 Query the WiFi Station Configuration

```
+WIFI_STA=1,My Supercool WiFi Network,**********
```

Queries the configuration used by the ADSBee to join an external WiFi network. Note that the value and ordering of the arguments is identical to the configuration command, except the password is censored.

2 Reporting Protocols

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

- CSBee
- GDL90
- MAVLINK 1
- MAVLINK 2
- Mode S Beast
- Raw Packets

The CONSOLE interface reports debug messages and AT command responses in addition to the selected reporting protocol. If the CONSOLE interface is being used as a reporting interface, it is recommended to send AT+LOG_LEVEL=SILENT to silence any debug logs that might corrupt the reported data, and to avoid sending additional AT commands while reading reported data in order to avoid the reported data being interspersed with OK and other AT command responses from the ADSBee 1090.

2.1 CSBee

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

The CSBee protocol is heavily inspired by the Aerobits Aero CSV protocol.

2.1.1 Aircraft Message

This message contains information about an aircraft being tracked via ADS-B (1090MHz). Aircraft reports are provided once per second, per aircraft, until contact with the aircraft has been lost for 60 seconds.

```
#A:ICAO,FLAGS,CALL,SQUAWK,ECAT,LAT,LON,ALT_BARO,ALT_GEO,TRACK,VELH,VELV,SIGS,
SIGQ,ACFPS,SFPS,SYSINFO,CRC\r\n
```

#A	Aircraft message start indicator	Format	Example value
ICAO	ICAO number of aircraft (3 bytes).	Hex Integer	3C65AC
FLAGS	Flags bitfield, see table 2.1.1.1.	Hex Integer	12F356A8
CALL	Callsign of aircraft.	String	N61ZP
SQUAWK	SQUAWK of aircraft.	Octal Integer	7232
ECAT	Emitter category, see table 2.1.1.2.	Integer	14
LAT	Latitude, in degrees.	Float	57.57634
LON	Longitude, in degrees.	Float	17.59554
ALT_BARO	Barometric altitude, in feet.	Integer	5000
ALT_GEO	Geometric altitude, in feet.	Integer	5000
DIR	Direction of aircraft, in degrees [0,360). Consult bit flags to determine whether this field is true heading, magnetic heading, or track.	Integer	35
VELH	Horizontal velocity of aircraft, in knots.	Integer	464
VELV	Vertical velocity of aircraft, in ft/min.	Integer	-1344
SIGS	Signal strength, in dBm.	Integer	-92
SIGQ	Signal quality, in dB.	Integer	2
SFPS	Number of valid 56-bit Squitter Mode S frames received from the aircraft during the last second.	Integer	2
ESFPS	Number of valid 112-bit Extended Squitter Mode S frames received from the aircraft during the last second.	Integer	5
SYSINFO	Aircraft data integrity and physical dimensions, see table 2.1.1.3.	Hex Integer	31BE89F2
CRC	CRC16 (described in 2.1.1.4).	Hex Integer	2D3E

2.1.1.1 FLAGS Bitfield

Note: All bits 17-32 are momentary (cleared and updated every reporting interval).

Bit	Bit Name	Meaning if the bit is set (1)
0	IS_AIRBORNE	Emitter is airborne.
1	BARO_ALTITUDE_VALID	Emitter has provided a barometer altitude.
2	GNSS_ALTITUDE_VALID	Emitter has provided a GNSS altitude.
3	POSITION_VALID	Emitter has provided a pair of even and odd Compact Position Reporting packets that were decoded to a location.
4	DIRECTION_VALID	Emitter has provided a direction (may be ground track, magnetic compass heading, or true compass heading).
5	HORIZONTAL_VELOCITY_VALID	Emitter has provided a horizontal velocity.
6	VERTICAL_VELOCITY_VALID	Emitter has provided a vertical velocity.
7	IS_MILITARY	Emitter has transmitted at least one packet using a military format, such as Military Extended Squitter (DF=19).
8	IS_CLASS_B2_GROUND_VEHICLE	Emitter is actually a ground vehicle using a Class B2 transponder with a transmission power < 70W.
9	HAS_1090_ES_IN	Emitter has receive capability for 1090MHz Extended Squitter transmissions.
10	HAS_UAT_IN	Emitter has receive capability for UAT (978MHz Universal Access Transceiver) transmissions.
11	TCAS_OPERATIONAL	Emitter has a functional TCAS (Traffic Collision Avoidance System) onboard.
12	SINGLE_ANTENNA	Emitter is using a single antenna, instead antennas above and below the fuselage. Transmissions may be weak or irregular during maneuvering.
17	DIRECTION_IS_HEADING	Surface position messages provided by the aircraft indicate a heading and not a track angle.
18	HEADING_USES_MAGNETIC_NORTH	Heading reported by the aircraft while on the surface uses magnetic north instead of true north.
19	IDENT	The aircraft has its SPI (Special Position Identification) bits set in Mode A/C or Mode S messages. This indicates that the pilot has depressed the momentary IDENT switch on their transponder, most likely at the request of air traffic control.
20	ALERT	The aircraft is issuing either a permanent or momentary alert. This could correspond to an operational mode change or something else.
21	TCAS_RA	The aircraft has an active TCAS resolution advisory (i.e. the aircraft is warning the pilot to take action in order to avoid colliding with another aircraft).
22	RESERVED_0	
23	RESERVED_1	
24	RESERVED_2	
25	RESERVED_3	
26	UPDATED_BARO_ALTITUDE	Barometric altitude has been updated within the last reporting interval.
27	UPDATED_GNSS_ALTITUDE	GNSS altitude has been updated within the last reporting interval.
28	UPDATED_POSITION	Position (latitude / longitude) has been updated within the last reporting interval.
29	UPDATED_DIRECTION	Direction has been updated within the last reporting interval.
30	UPDATED_HORIZONTAL_VELOCITY	Horizontal velocity has been updated within the last reporting interval.
31	UPDATED_VERTICAL_VELOCITY	Vertical velocity has been updated within the last reporting interval.

2.1.1.2 ECAT Field

The ECAT field indicates the Emitter Category (i.e. airframe type) for each ADSB emitter that is being tracked. This field contains information about what kind of aircraft, ground vehicle, obstacle, or other airspace user is emitting ADS-B packets, and can be used to understand the emitter's maneuvering capability and potential for wake vortex impact.

ECAT Value	Emitter Category
0	Invalid
1	Reserved
2	No Category Information
3	Surface Emergency Vehicle
4	Surface Service Vehicle
5	Ground Obstruction
6	Glider / Sailplane
7	Parachutist / Skydiver
8	Ultralight / Hang Glider / Paraglider
9	Unmanned Aerial Vehicle
10	Space / Transatmospheric Vehicle
11	Light Aircraft (< 7,000kg)
12	Medium 1 (7,000kg – 34,000kg)
13	Medium 2 (34,000kg – 136,000kg)
14	High Vortex Aircraft
15	Heavy (> 136,000kg)
16	High Performance (> 5 G acceleration and > 400 kts speed)
17	Rotorcraft

2.1.1.3 SYSINFO Bitfield

SYSINFO Bitfield															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			Aircraft Maximum Dimension (MDIM)							GNSS Antenna Offset Direction (GAOR)	GNSS Antenna Offset Distance (GAOD)		GNSS Antenna Offset Known (GAOK)	System Design Assurance (SDA)	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Source Integrity Level (SIL)		Geometric Vertical Accuracy (GVA)		Navigation Accuracy Category: Position (NAC_p)				Navigation Accuracy Category: Velocity (NAC_v)			Navigation Integrity Category: Barometer (NIC_{baro})	Navigation Integrity Category (NIC)			

SYSINFO[0-3]: Navigation Integrity Category (NIC)

The radius of containment (NIC) indicates how much trust should be placed in an aircraft's reported location in the horizonal plane. The NIC reports a radius of containment specified by the avionics system of the emitter. The probability that the aircraft is outside of this radius of containment due to its avionics system receiving a faulty signal from one of its inputs (without displaying an error) is provided by another bitfield called the Source Integrity Level (SIL). Combined, the NIC and SIL indicate how likely it is that an aircraft is not actually contained by a bubble of a specified size, centered at the aircraft's reported location, assuming that the avionics onboard the aircraft are functioning correctly but may be given faulty inputs.

A higher NIC value indicates more trust in an aircraft's reported latitude / longitude position.

NIC Value	0	1	2	3	4	5	6	7	8	9	10	11
Radius of Containment	Unknown	< 20 NM	< 8 NM	< 4 NM	< 2 NM	< 1 NM	< 0.6 NM	< 0.2 NM	< 0.1 NM	< 75 m	< 25 m	< 7.5 m

SYSINFO[4]: Navigation Integrity Category: Barometer (NIC_{baro})

The barometric altitude integrity (NIC_{baro}) indicates how much trust should be placed in an aircraft's reported altitude. The field is a single bit that indicates whether the aircraft uses an altimeter that has been cross-checked against other sources. Old school encoding altimeters have many parallel wires and output an altitude in a format called a Gillham Code, and have no built-in method of error checking. A single faulty wire can result in erroneous readings, so this bit lets air traffic control know whether to take altitude readings from the aircraft with a grain of salt. A 0 indicates that the transponder is outputting altitude from a Gillham coded source (with no way to cross check the value), while a 1 indicates that the transponder is outputting altitude from a Gillham coded source while using another sensor to cross-check it, or is using a more modern barometer that supports a protocol with built-in error checking.

A higher NIC_{baro} value (i.e. 1 instead of 0) indicates more trust in the aircraft's reported altitude.

NIC_{baro} Value	0	1
Barometric Altitude Integrity	Altitude is from a Gillham-coded input, not cross checked.	Altitude is from a Gillham-coded input that is being cross-checked with another source, or from a non Gillham-coded input with built-in error checking features.

SYSINFO[5-7]: Navigation Accuracy Category: Velocity (NAC_v)

The horizontal velocity error (NAC_v) indicates the expected accuracy of the reported velocity of the aircraft when systems are operating nominally. This varies depending on the accuracy capabilities of the measurement equipment onboard the aircraft, and not how often we expect said equipment to fail.

A higher NAC_v indicates a more accurate velocity measurement system.

NAC_v Value	**Horizontal Velocity Error**
0b000	Unknown or ≥ 10 m/s
0b110	< 10 m/s
0b010	< 3 m/s
0b011	< 1 m/s
0b100	< 0.3 m/s

SYSINFO[8-11]: Navigation Accuracy Category: Position (NAC_p)

The estimated position uncertainty (NAC_p) indicates the expected accuracy of the aircraft's reported location when systems are operating nominally. This varies depending on the capabilities of the aircraft's positioning system and not on how often we expect said equipment to fail.

A higher NAC_p indicates a more accurate positioning system.

NAC_p Value	0	1	2	3	4	5	6	7	8	9	10	11
Estimated Position Uncertainty	Unknown or ≥ 10NM	< 10 NM	< 4 NM	< 2 NM	< 1 NM	< 0.5 NM	< 0.3 NM	< 0.1 NM	< 0.5 NM	< 30 m	< 10 m	< 3 m

SYSINFO[12-13] Geometric Vertical Accuracy (GVA)

The geometric vertical accuracy indicates the 95% confidence interval (vertical figure of merit) provided by the aircraft's onboard GNSS system (i.e. assuming some distribution of altitudes, the aircraft's GNSS system is confident that 95 out of 100 times, the aircraft falls within some height of the reported geometric altitude.

GVA Value	0	1	2	3
95% Vertical Figure of Merit (VFOM)	Unknown or ≥ 150 m	< 150 m	≤ 45 m	< 45 m (Was previously "reserved", the actual value of this field may change but is guaranteed to be < 45 m).

SYSINFO[14-15] Source Integrity Level (SIL)

The source integrity level indicates the probability that the aircraft exceeds the bounds of its horizontal radius of containment (NIC) due to a silent fault in signals received by the aircraft (no avionics failure).

SIL Value	**Probability of Exceeding NIC Radius of Containment Due to Silent Fault**
0	Unknown or > $1x10^{-3}$ per flight hour.
1	≤ $1x10^{-3}$ per flight hour.
2	≤ $1x10^{-5}$ per flight hour.
3	≤ $1x10^{-7}$ per flight hour.
4	Unknown or > $1x10^{-3}$ per sample.
5	≤ $1x10^{-3}$ per sample.
6	≤ $1x10^{-5}$ per sample.
7	≤ $1x10^{-7}$ per sample.

SYSINFO[16-17] System Design Assurance (SDA)

The system design assurance indicates how robust the aircraft's position reporting systems are to failures of various severities. For instance, SDA = 1, a low SDA value, corresponds to Software and Hardware Design Assurance Level D, which states that a minor failure could cause the aircraft to transmit misleading position information with a probability of $\leq 1x10^{-3}$ per flight hour. A more robust system with SDA = 3, corresponding to Software and Hardware Design Assurance Level B, is expected to transmit misleading position information with a probability of $1x10^{-7}$ per flight hour even under a Hazardous failure condition.

Software Design Assurance categories used in this field are classified under RTCA DO-178B, Airborne Electronic Hardware Design Assurance are classified under RTCA DO-254, and failure classification levels are defined in FAA Advisory Circular AC-23.1309-1E.

SDA Value	Supported Failure Condition	Probability of Undetected Fault causing transmission of False or Misleading Information	Software and Hardware Design Assurance Level
0	Unknown / No Safety Effect	$> 1x10^{-3}$ per flight hour or unknown.	N/A
1	Minor	$\leq 1x10^{-3}$ per flight hour.	D
2	Major	$\leq 1x10^{-5}$ per flight hour.	C
3	Hazardous	$\leq 1x10^{-7}$ per flight hour.	B

SYSINFO[18] GNSS Antenna Offset Known (GAOK)

This field indicates whether the aircraft has reported the installation location of its GNSS antenna relative to its centerline (roll axis). A field for reporting this value is only available in ADS-B messages emitted by aircraft on the ground, and even then, the aircraft may not report a value in this field.

GANTO Value	Aircraft reported location of its GNSS antenna relative to roll axis?
0	No
1	Yes

SYSINFO[19-20] GNSS Antenna Offset Distance (GAOD)

This field indicates the distance that the GNSS antenna is offset from the centerline (roll axis) of the aircraft. Aircraft only report even values for their GNSS antenna offset distance, between 2-6 meters, so the reported offset distance can be calculated using the equation below.

GNSS antenna offset distance = GAOD << 1

Note that this value is only reported by some aircraft while operating on the ground. Aircraft operating in the air do not report this value. Always check the value of the GAOK bit to see if the value of GAOD is worth paying attention to.

SYSINFO[21] GNSS Antenna Offset Direction (GAOR)

GAOR Value	GNSS Antenna Offset Direction
0	GNSS antenna is offset to the left of centerline (roll axis).
1	GNSS antenna is offset to the right of centerline (roll axis).

This field indicates the value of the maximum dimension (length or width) of an aircraft, and is only reported by aircraft while on the ground. This field has no special coding, and can be interpreted directly as a binary unsigned integer value.

2.1.1.4 CRC Field

CSBee messages use a 16-bit Cyclical Redundancy Checksum (CRC-16), which can be calculated using the algorithm in the C++ code snippet below. Note the “swap16” helper function which also needs to be included.

```
uint16_t swap16(uint16_t value) { return (value << 8) | (value >> 8); }

uint16_t CalculateCRC16(const uint8_t *data_p, int32_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);
}
```

2.1.2 Statistics Message

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

```
#S:DPS,ACFPS,SFPS,TSCAL,UPTIME,CRC\r\n
```

#S	Statistics message start indicator	Example
DPS	Number of attempted demodulations in the last second.	106
RAW_SFPS	Number of squitter (56-bit Mode S) frames received in the last second.	
SFPS	Number of valid squitter (56-bit Mode S) frames received in the last second.	20
RAW_ESFPS	Number of extended squitter (112-bit Mode S) frames received in the last second.	
ESFPS	Number of valid Mode S frames received in the last second.	3
NUM_AIRCRAFT	Number of aircraft tracked in the last second.	
TSCAL	Calibration value for TS field in raw frames	13999415
UPTIME	Time from last enter to RUN mode, in seconds.	134
CRC	CRC16 (described in 2.1.2.1).	2D3E

2.1.2.1 CRC Field

See 2.1.1.4.

2.3 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.

2.3.1 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

2.3.2 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.