OnFlight Hub Binary Data Log Description

Firmware v1.0

Document Revision 1.0

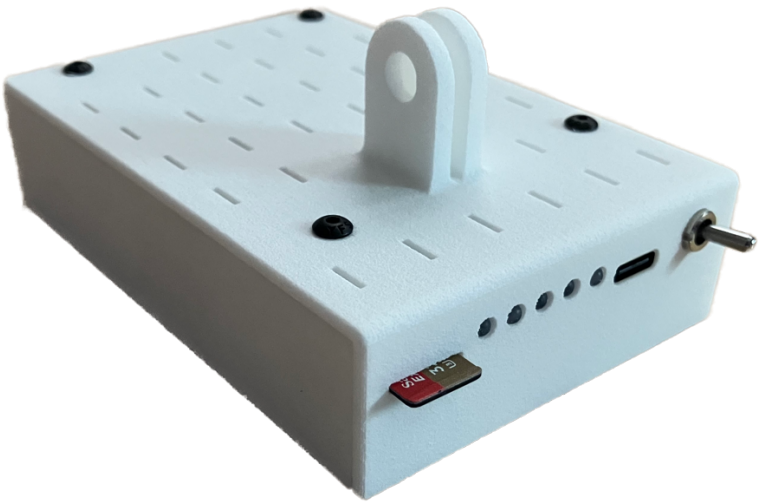

Table of Contents

1 Technical Documentation 3
2 Support 3
3 Introduction 3
4 Data Format 3
5 Status Bit Field 6
6 Fletcher 16 Checksum 7

1 Technical Documentation

The following documentation and support software are included with OnFlight and available from our website:

- **User Manual:** describes the OnFlight Hub, specifications, and operations.
- **CSV Data Log Description:** describes the fields available in the CSV formatted data logs.
- **Binary Data Log Description (this document):** describes the binary data log format that OnFlight Hub uses to write data. This is useful for application developers who would like to natively read and use these data logs.
- **UDP Broadcast Description:** describes the real-time UDP broadcast packet format that is sent by OnFlight Hub.
- **Data Converter:** application for Windows or MacOS, which converts the data from OnFlight to CSV format.

2 Support

If you have technical problems or cannot find the information you need in the provided documents, please contact our technical support team by email at: support@bolderflight.com. Our team is committed to providing the support necessary to ensure that you are successful using our products.

3 Introduction

OnFlight Hub writes data logs in a binary format, which can then be converted to CSV and other data log formats using a Data Converter application for Windows or MacOS. This document describes the binary format to enable application developers to natively read and use our binary data logs.

OnFlight Hub logs data to a removable micro-SD card. Data log files are named *dataX.onflight* where *X* increments from 0 to avoid overwriting files present on the card.

OnFlight Hub runs at a hard real-time rate of 50 Hz, which is driven by the Inertial Measurement Unit (IMU) data-ready interrupt. The binary data for each frame is buffered and then written to the micro-SD card in a lower-priority task. Each data frame starts with a two-byte header ('B', 'F'), followed by the data payload, and a two-byte Fletcher 16 checksum. The payload length remains constant. Power is applied to OnFlight Hub via an electro-mechanical switch and no effort is made to flush the data to the micro-SD card on power-off. This means that the last few frames of data may be lost and the last data frame on the card is likely a partial frame of data and should be discarded.

Data is written in little-endian format.

4 Data Format

The data format is described below with the status bit fields described in Section 5 and the checksum described in Section 6.

Byte Offset	Type	Name	Scale	Unit	Description
0	U1[2]	header	-	-	Header {'B', 'F'}.
2	U1	version	-	-	Version number, currently 0.
3	U1[5]	status	-	-	Status, see Section 5 for bit field description.
8	U4	sys_time_ms	-	ms	Time since boot.
12	U1	input_volt	1 / 25	V	Input voltage.
13	U1	filt_input_volt	1 / 25	V	Filtered input voltage.
14	I1	cpu_die_temp_c	1	C	CPU die temperature
15	I1	imu_die_temp_c	1	C	IMU die temperature.
16	I2	imu_accel_x_g	1 / 1000	G	X-axis accelerometer.

18	I2	imu_accel_y_g	1 / 1000	G	Y-axis accelerometer.
20	I2	imu_accel_z_g	1 / 1000	G	Z-axis accelerometer.
22	I2	imu_gyro_x_dps	1 / 10	deg/s	X-axis gyro.
24	I2	imu_gyro_y_dps	1 / 10	deg/s	Y-axis gyro.
26	I2	imu_gyro_z_dps	1 / 10	deg/s	Z-axis gyro.
27	I1	mag_die_temp_c	1	C	Magnetometer die temperature.
28	I2	mag_x_ut	1 / 80	uT	X-axis magnetometer.
30	I2	mag_y_ut	1 / 80	uT	Y-axis magnetometer.
32	I2	mag_z_ut	1 / 80	uT	Z-axis magnetometer.
34	I1	pres_die_temp_c	1	C	Static pressure die temperature
35	U2	pres_pa	2	Pa	Static pressure.
37	U1	gnss_fix_num_sv	-	-	Lower 3 bits are GNSS fix (0 = no fix, 2 = 2D fix, 3 = 3D fix, 4 = differential fix). Upper 5 bits are the number of satellite vehicles used in the solution.
38	U1	gnss_utc_year	-	-	UTC year from 1970 from GNSS receiver.
39	U1	gnss_utc_month	-	-	UTC month from GNSS receiver.
40	U1	gnss_utc_day	-	-	UTC day from GNSS receiver.
41	U1	gnss_utc_hour	-	-	UTC hour from GNSS receiver.
42	U1	gnss_utc_min	-	-	UTC minute from GNSS receiver.
43	U1	gnss_utc_sec	-	-	UTC second from GNSS receiver.
44	U1	gnss_horz_pos_acc_ft	1 / 10	ft	Estimated GNSS horizontal position accuracy.
45	U1	gnss_vert_pos_acc_ft	1 / 10	ft	Estimated GNSS vertical position accuracy.
46	U1	gnss_vel_acc_kts	1 / 10	kts	Estimated GNSS velocity accuracy.
47	I2	gnss_ned_vel_x_kts	1 / 10	kts	GNSS North velocity.
49	I2	gnss_ned_vel_y_kts	1 / 10	kts	GNSS East velocity.
51	I2	gnss_ned_vel_z_kts	1 / 100	kts	GNSS Down velocity.
53	U2	gnss_alt_wgs84_ft	-	ft	GNSS altitude above the WGS84 ellipsoid biased by +10,000 ft (i.e. alt = gnss_alt_wgs84_ft - 10000).
55	I2	gnss_geoid_height_ft	1 / 10	ft	GNSS geoid height, (MSL = WGS84 – geoid).
57	I4	gnss_lat_deg	1e-7	deg	GNSS latitude.
61	I4	gnss_lon_deg	1e-7	deg	GNSS longitude.
65	I2	ins_pitch_deg	1 / 100	deg	Pitch attitude.
67	I2	ins_roll_deg	1 / 100	deg	Roll attitude.
69	I2	ins_mag_var_deg	1 / 100	deg	Magnetic variation, + east.
71	U2	ins_heading_true_deg	1 / 100	deg	Heading angle, true, 0 – 360.
73	U2	ins_heading_mag_deg	1 / 100	deg	Heading angle, magnetic, 0 – 360.
75	I2	ins_climb_rate_ftpm	1	ft/min	Climb rate.
77	I2	ins_load_factor	1 / 1000	G	Load factor.
79	I2	ins_accel_x_g	1 / 1000	G	X-axis accelerometer, bias estimated and removed, low pass filtered.
81	I2	ins_accel_y_g	1 / 1000	G	Y-axis accelerometer, bias estimated and removed, low pass filtered.
83	I2	ins_accel_z_g	1 / 1000	G	Z-axis accelerometer, bias estimated and removed, low pass filtered.

85	I2	ins_gyro_x_dps	1 / 10	deg	X-axis gyro, bias estimated and removed, low pass filtered.
87	I2	ins_gyro_y_dps	1 / 10	deg	Y-axis gyro, bias estimated and removed, low pass filtered.
89	I2	ins_gyro_z_dps	1 / 10	deg	Z-axis gyro, bias estimated and removed, low pass filtered.
91	I2	ins_mag_x_ut	1 / 80	uT	X-axis magnetometer, low pass filtered.
93	I2	ins_mag_y_ut	1 / 80	uT	Y-axis magnetometer, low pass filtered.
95	I2	ins_mag_z_ut	1 / 80	uT	Z-axis magnetometer, low pass filtered.
97	I2	ins_ned_vel_x_kts	1 / 10	kts	INS North velocity.
99	I2	ins_ned_vel_y_kts	1 / 10	kts	INS East velocity.
101	I2	ins_ned_vel_z_kts	1 / 100	kts	INS Down velocity.
103	U2	ins_gnd_spd_kts	1 / 100	kts	INS ground speed.
105	U2	ins_gnd_track_true_deg	1 / 100	deg	INS ground track, true, 0 – 360.
107	U2	ins_gnd_track_mag_deg	1 / 100	deg	INS ground track, mag, 0 – 360.
109	I2	ins_flt_path_deg	1 / 100	deg	INS flight path angle, +/- 90
111	U2	ins_alt_wgs84_ft	-	ft	INS height above the WGS84 ellipsoid, biased by +10,000 ft (i.e. alt = ins_alt_wgs84_ft - 10000).
113	I4	ins_lat_deg	1e-7	deg	INS latitude.
117	I4	ins_lon_deg	1e-7	deg	INS longitude.
121	U2	adc_pres_pa	2	Pa	Air data computer static pressure.
123	U2	adc_pres_alt_ft	-	ft	Air data computer pressure altitude, biased by +10,000 ft (i.e. alt = adc_pres_alt_ft - 10000).
125	I1	airdata_die_temp_c	1	C	External airdata board die temperature.
126	U2	airdata_static_pres_pa	2	Pa	External airdata static pressure.
128	U2	airdata_diff_pres_pa	1	Pa	External airdata differential pressure.
130	I2	airdata_oat_c	1 / 100	C	External airdata outside air temperature.
132	U2	airdata_ias_kts	1 / 100	kts	External airdata Indicated Air Speed (IAS).
134	U2	airdata_tas_kts	1 / 100	kts	External airdata True Air Speed (TAS).
136	U2	airdata_pres_alt_ft	-	ft	External airdata pressure altitude biased by +10,000 ft (i.e. alt = airdata_pres_alt_ft - 10000).
138	U2	airdata_density_alt_ft	-	ft	External airdata density altitude biased by +10,000 ft (i.e. alt = airdata_density_alt_ft - 10000).
140	I2	airdata_aoa	1 / 100	-	External airdata angle of attack, this is either an angle or a pressure ratio as defined by the status bitfield.
142	I1	agl_alt_die_temp_c	1	C	External AGL altimeter board die temperature.
143	I2	agl_alt_in	1	in	External AGL altimeter altitude
145	U2	checksum	-	-	Fletcher 16 checksum, as described in Section 6.

5 Status Bit Field

Status bytes are used to efficiently encode data, below is the description and bit masking of these bytes. The description describes the case if a bit occupies that position.

Byte	Mask	Description
0	0x01	Filtered input voltage between 3.4V and 3.6V.
0	0x02	Filtered input voltage below 3.4V.
0	0x04	CPU die temperature between -30C and +70C.
0	0x08	New IMU data received.
0	0x10	IMU healthy.
0	0x20	IMU die temperature between -30C and +70C.
0	0x40	New magnetometer data received.
0	0x80	Magnetometer healthy.
1	0x01	Magnetometer die temperature between -30C and +70C.
1	0x02	New static pressure data received.
1	0x04	Static pressure healthy.
1	0x08	Static pressure die temperature between -30C and +70C.
1	0x10	New GNSS data received.
1	0x20	GNSS healthy.
1	0x40	INS initialized.
1	0x80	INS healthy.
2	0x01	New external airdata message received.
2	0x02	External airdata module connected.
2	0x04	External airdata battery status warning.
2	0x08	External airdata battery status critically low.
2	0x10	External airdata board die temperature ok.
2	0x20	External airdata OAT measurement available.
2	0x40	External airdata AOA measurement available.
2	0x80	External airdata new static pressure data received.
3	0x01	External airdata static pressure healthy.
3	0x02	External airdata new differential pressure data received.
3	0x04	External airdata differential pressure healthy.
3	0x08	External airdata new OAT data received.
3	0x10	External airdata OAT healthy.
3	0x20	External airdata new AOA data received.
3	0x40	External airdata AOA healthy.
3	0x80	External airdata AOA data type is angle in degrees, otherwise data type is pressure ratio.
4	0x01	New external AGL altimeter message received.
4	0x02	External AGL altimeter module connected.
4	0x04	External AGL altimeter battery status warning.
4	0x08	External AGL altimeter battery status critically low.
4	0x10	External AGL altimeter board die temperature ok.
4	0x20	External AGL altimeter new sensor data received.
4	0x40	External AGL altimeter sensor healthy.
4	0x80	External AGL altimeter sensor in range.

6 Fletcher 16 Checksum

The Fletcher 16 checksum is computed over the entire packet up to the checksum bytes, including the header. The checksum is computed as:

```
uint16_t Checksum(uint8_t * data, size_t len) {
  uint16_t sum0 = 0;
  uint16_t sum1 = 0;
  for (size_t i = 0; i < len; i++) {
    sum0 = (sum0 + data[i]) % 255;
    sum1 = (sum1 + sum0) % 255;
  }
  return sum1 << 8 | sum0;
}
```