# SD Card Data Format - Animal Tracking Project

## SD Card Organization

SD cards are organized in blocks each of which is 512 bytes. The tracking data will always be written in complete blocks starting from the first one and continuing sequentially. No block will normally be written after an unwritten block. (This latter situation may occur if a write fails leaving a, possibly unusable, block in an unwritten state.)

Each block starts with a 4 byte sequence number between $00000001 and $FFFFFFFE. The bit patterns $00000000 and $FFFFFFFF are not used as either can be used to fill unwritten blocks when the device is erased. Thus a valid sequence number indicates that the block has been written.

The data written to these blocks is organized in segments. Segments do not cross block boundaries. Each segment starts and ends in a single block. At the end of each segment is the segment trailer, a two byte data element which identifies the type of segment that precedes it. Segment trailers always contain the segment type and the length of the segment (excluding the length of the segment trailer itself). Putting them at the end of the segments allows segments to be written as their data accumulates rather than requiring the data to be buffered so that the length can be determined and written to a header before the data is. The end of each block (bytes 510 and 511) will always be a segment trailer. If no valid trailer is found here it indicates that the block is invalid and must be ignored.


## Segment Types

There are currently a number of segment types. The most basic is the Padding Segment (segment type $01). It is used to mark the bytes within it as unused. Normally it is found at the end of a block to fill it out to the end. It is placed here when the next segment will not fit in the remains of the block or when the block needs to be immediately written out, even though not yet full.

The Status Segment (segment type $02) contains the status of the system at the time it is written to the block. The SD data structure will be associated with a project compile time. Parameters of the system such as IMU and audio sample rates will also be tied to a project compile time. Version 1 segments have the following format:

- 1 Bytes Version Number = 0x01
- 4 Bytes Time of Project Compile (Can be inserted with TCL scripting)
- 4 Bytes Time of Project Commit

- 9 Bytes FPGA Time. The FPGA time at which the status segment was assembled.
- 9 Bytes Last Accelerometer Sample FPGA Time
- 9 Bytes Last Magnetometer Sample FPGA Time
- 9 Bytes Last Gyroscope Sample FPGA Time
- 9 Bytes Last Temperature Sample FPGA Time
- 9 Bytes Last Audio Sample FPGA Time
- 1 Byte Number of microphones active

The GPS Time Mark Segment (segment type $03) contains time stamp data provided by the UBX TIM_TM2 packet. It contains the following information:

- 9 Bytes GPS Time Stamp. This GPS Time stamp comes from a TIM_TM2 packet where time is separated into week number, milliseconds into week and nanoseconds of millisecond.
- 9 Bytes FPGA Time that GPS Time Stamp is requested.

The GPS Position Segment (segment type $04) contains the position of the tracker at the given time as given by the UBX NAV_SOL packet:

- 1 Byte GPSFix Type
- 12 Bytes Signed XYZ ECEF Coordinates
- 4 Bytes 3D Position Accuracy Estimate
- 2 Bytes Position DOP (Dilution of Precision)
- 9 Bytes FPGA Time Associated with this coordinate

The following IMU segments do not contain FPGA time inside the segments. To associate a time with a sample segment, one should look for the next status segment. In the next status segment will be the FPGA time associated with the all the immediately previous IMU samples written before forming that status segment. Knowing the sample rates of the IMU sensors allows one to work backwards in the stream, associating samples with FPGA times.

The IMU Gyroscope Segment (segment type $05) contains the IMU Gyroscope details of the tracker at the given time:

- 6 Bytes XYZ Gyroscope. (2 bytes per axis)

The IMU Accelerometer Segment (segment type $06) contains the IMU Accelerometer details of the tracker at the given time:

- 6 Bytes XYZ Accelerometer.

The IMU Magnetometer Segment (segment type $07) contains the IMU Magnetometer details of the tracker at the given time:

- 6 Bytes XYZ Magnetometer.

The IMU Temperature Segment (segment type $0A) contains the IMU Temperature details of the tracker at the given time:

- 2 Bytes Temperature.

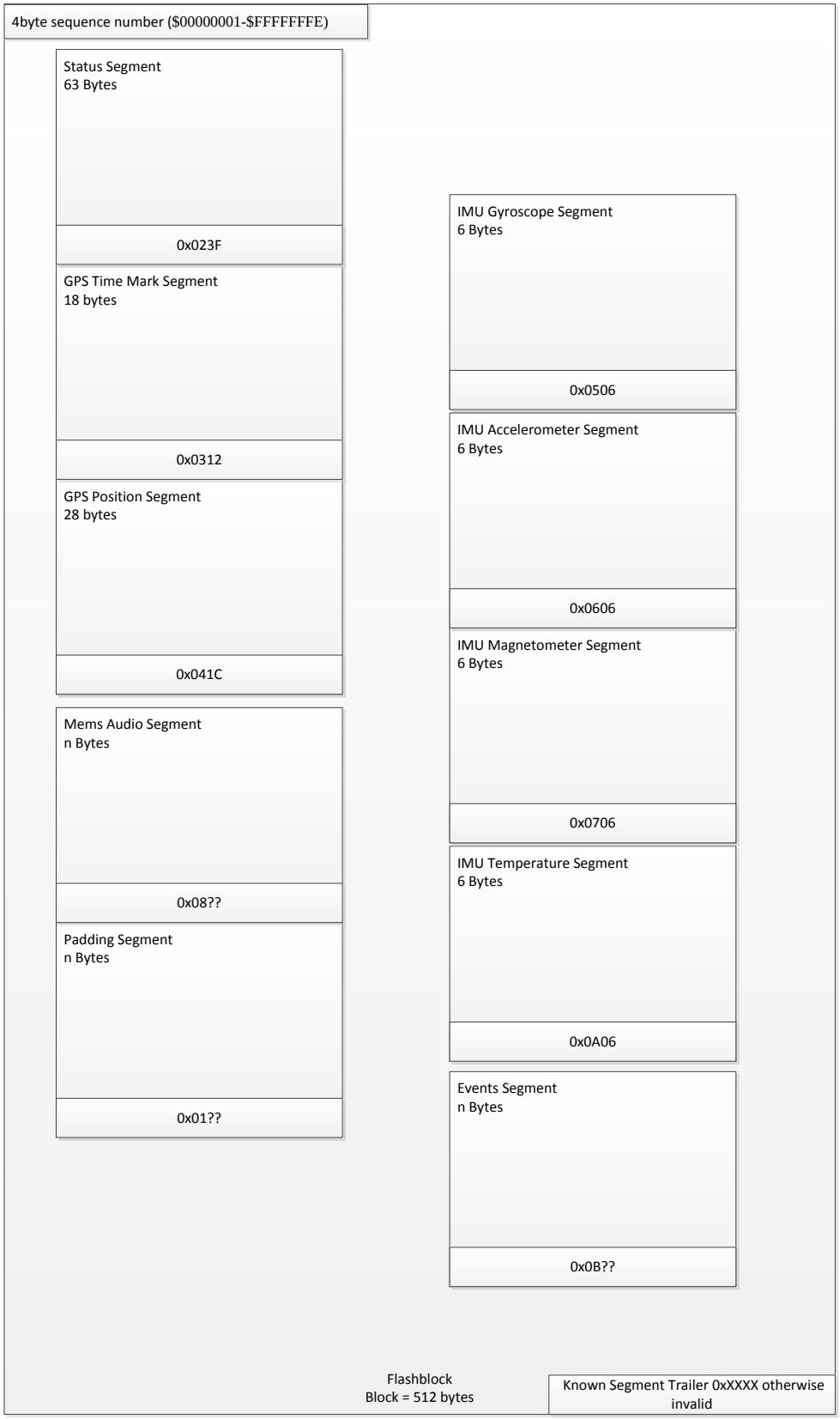
Like the IMU sample segments, the audio segments do not contain a time. To associate audio data with time, one should look for the next status segment written. In it will be the FPGA time associated with the last audio sample taken and written before forming that status segment. We know the audio sample rate and can thus work backwards in the stream.

The Audio Segment (segment types $08 - $0F) contains audio sample data acquired continuously. The format is as follows:

- n Bytes Audio Sample Bytes. Collected audio samples are stored here. They may not be multiples of 8 bits each. Consequently, the number of samples in a segment may not use all the bits in its last byte. The lower three bits of the Audio Segment Type indicate how many bits of the last byte are unused. The unused bits will be the top bits of the next byte. Version number 0x01 will use 16bit audio words and thus segment type will be 0x08.

The Event Segment (segment types $0B) contains event types and associated counts since last event segment write. Event segments are written whenever a status segment write is requested. The format is as follows:

- n Bytes Event Count Bytes. This segment contains the counts of events since the last event count segment was formed. Writing an event count segment will clear the event counts. Event count of 1 byte is followed by the event type which is 1 byte. A VHDL file will enumerate the various event numbers and what they represent as the system grows.


4byte sequence number ($00000001-$FFFFFFFE)
Status Segment
63 Bytes
0x023F
GPS Time Mark Segment
18 bytes
0x0312
GPS Position Segment
28 bytes
0x041C
Mems Audio Segment
n Bytes
0x08??
Padding Segment
n Bytes
0x01??
IMU Gyroscope Segment
6 Bytes
0x0506
IMU Accelerometer Segment
6 Bytes
0x0606
IMU Magnetometer Segment
6 Bytes
0x0706
IMU Temperature Segment
6 Bytes
0x0A06
Events Segment
n Bytes
0x0B??
Flashblock
Block = 512 bytes
Known Segment Trailer 0xXXXX otherwise invalid

Changes:

Sector terminology changed to blocks.