CU InSpace

Data Logging Format

2021-09-12

Samuel Dewan

Contents

1 Introduction

To do.

2 SD Card Data Layout

2.1 Master Boot Record

The CU InSpace Avionics system requires SD cards that have been formated with a Master Boot Record (MBR) in block 0.

2.2 Data Partitions

CU InSpace data is logged into partitions with partion type **0x89**. A data logging system will always log into the first valid partition on the disk. If the first partition becomes full it may continue logging into later partitions.

CU InSpace logging data partitions are layed out with a single super block followed by one or more SD Card blocks containing "blocks" of application data. The application data "blocks" each start with a header which indicates the type of data in the block and the length of the block. Application data blocks are stored contiguously and may be stored accross SD Card block boundries.

2.2.1 Super Block

Every CU InSpace data partition must start with a Super Block which follows the format shown in figure 2.1.

2.3 Data Blocks

Every data block starts with a four byte block header. This header indicates the type of information contained in the block. The block header format is described in figure 2.2.

The *Class* and *Type* fields are used to specify how the data block should be interpreted. The *Length* field specifies the length in bytes of the data block including the block header. This length field can be used to find the start of the next data block when parsing and can also be used in parsing data blocks for which the format does not specify a fixed length. The *Length* value must always be a multiple of four bytes. A length of zero is invalid.

Table 2.1 shows the possible values for the *Class* field in the data block header. Section 3 describes the data formats for blocks of each of the classes.

Data Block Class	Description
0x0	Logging Metadata (see section 3.1)
0x1	Telemetry Data (see section 3.2)
0x2	Diagnostic Data (see section 3.3)
0x3 through 0x3F	Reserved for future use

Table 2.1: Data Block Classes

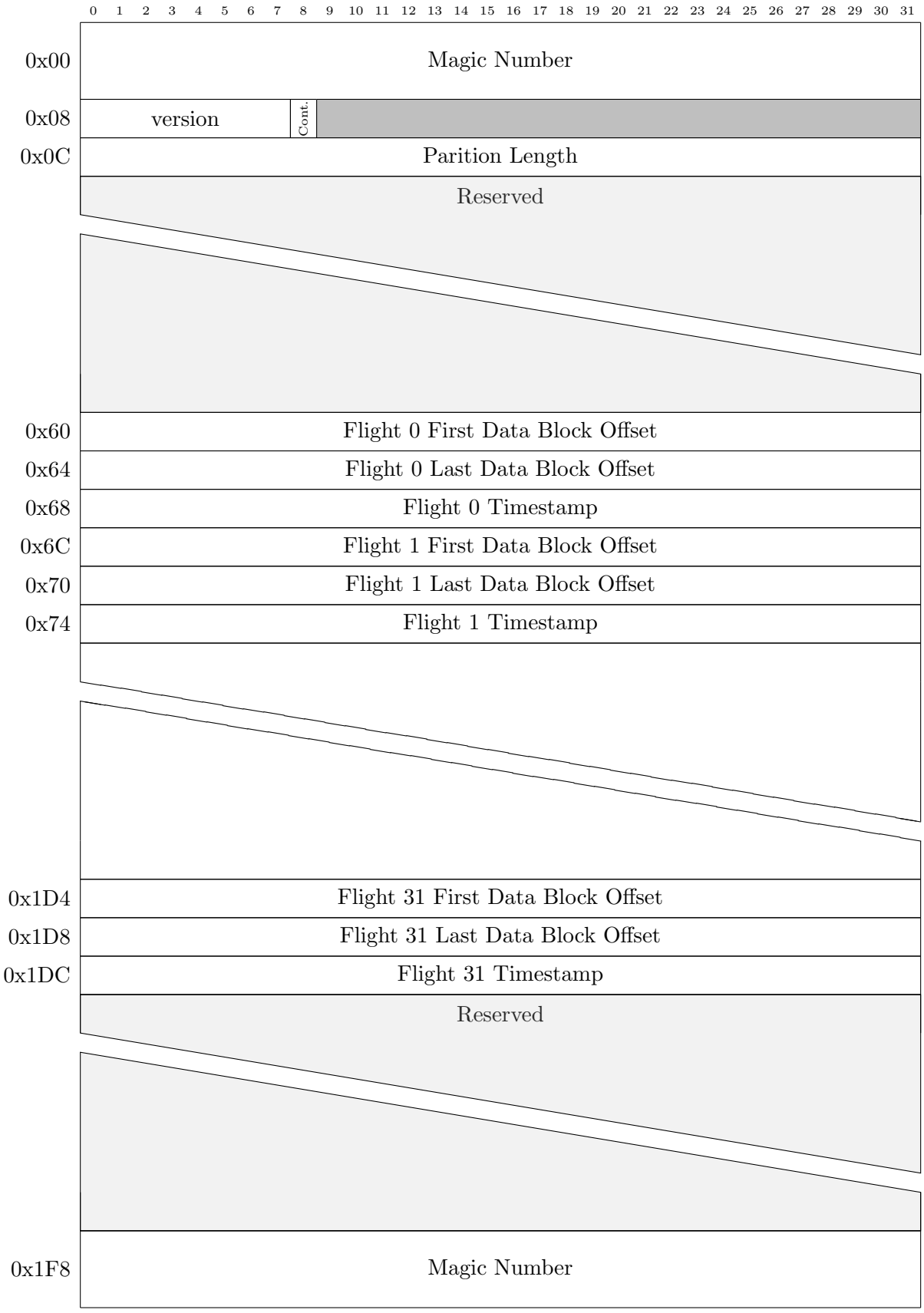

Figure 2.1: Super block format

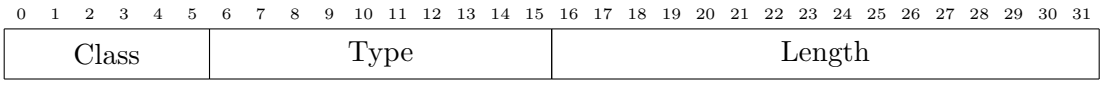

Figure 2.2: Block header format

3 Block Formats

3.1 Logging Metadata Blocks

Logging metadata blocks are used to store metadata generated by the logging system and information required in order to correctly parse the logged data.

The block types within the Logging Metadata class are described in table 3.1.

Data Type	Description
0x0	Spacer block
0x1 through 0x3FF	Reserved for future use

Table 3.1: Logging metadata block types

3.1.1 Spacer Block

Spacer blocks are used when writing telemetry data to indicate a region that does not contain any valid data. When telemetry is being written one SD card block at a time without the ability to have telemetry data overflow from one block into another, a spacer block can be used at the end of the SD card block to take up any excess bytes.

When parsing, spacer blocks should always be skiped using the length field in the block header and no attempt should be made to parse a spacer block's payload.

3.2 Telemetry Data Blocks

Data blocks of the Telemetry class follow the formats specified in the CU InSpace Radio Packet Format document.

3.3 Diagnostic Data Blocks

Data blocks with the Diagnostic Data class are used to store information which is intended to be used for debugging.

The block types within the Diagnostic Data class are described in table 3.2.

Data Type	Description
0x0	Log message
0x1	Outgoing radio packet
0x2	Incoming radio packet
0x3 through 0x3FF	Reserved for future use

Table 3.2: Diagnostic data block types

3.3.1 Log Message

Log messages are string messages intended to provide human readable debugging output. They are encoded as UTF-8 strings and do not require a terminating NUL character because the string length can be extrapolated from the block length.

Note that like all other data blocks log messages must be a multiple of four bytes long. If the string to be logged is not a multiple of four bytes it must be padded with NUL characters at the end to make it fit the required alignment. The length recorded in the block header must be a multiple of four, and therefore must include any NUL padding bytes at the end of the string.

3.3.2 Outgoing Radio Packet

3.3.3 Incoming Radio Packet