Binary MLG Logging (MLVLG) file format specification

No index entries found.

Document purpose:

This document specifies the MLVLG file format.

Format purpose:

This MLVLG file format is to serve as a standardized format for capturing controller data in binary format for viewing software to open directly with no external dependencies.

Authors:

Philip Tobin

Revision history:

Rev. 0.9 – August 28, 2017- Phil Tobin – draft spec.
Rev. 0.91 – August 31, 2017- Phil Tobin – support for bit fields added, incorporated edits per review..
Rev. 0.92 – Sept 8, 2017- Phil Tobin – updated error in block ID's..
Rev. 0.93 – Sept 12, 2017- Phil Tobin – Added simple crc to record data..

Definitions:

- Datalog: A set of runtime data captured to a file
- Datalog session: time interval between datalog-on (begin datalog) and datalog-off (stop datalog) user actions
- UNIX Standard 32bit timestamp: number of seconds elapsed since midnight Coordinated Universal Time (UTC) of January 1, 1970, not counting leap seconds

Format structure:

One MLVLG file associates with (and therefore contains data of) one datalog session contained within 1 binary file generally of the extension *.MLG. The *.MLG file extension will be associated to the appropriate EFI Analytics Applications for viewing.

All data is expected in Big Endianess

A MLVLG file is made up of two sections:
1. Header
2. Data Block

Header	Data Block	Data Block	Data Block	….

The header will always be at the start of the file, followed by n Data Block records.

Header

Purpose

Header is a data structure to define the data captured in the remainder of the file.

1.2 Header format fields

- File format (6 bytes): unique identifier for MLVLG file format (same for any *. MLG file)
- Format version (2 bytes): unique identifier of the particular version of the MLVLG file format
- Date/Time (4 bytes): date/time encoded to Unix 32bit standard (1 second resolution) – *This is an Optional field, fill with 0's if correct time stamp unavailable.*
- Data Block start offset (4 bytes): offset in bytes of where Data Block starts in the file, relative to the beginning of the file
- Record Length (2 bytes): length of output in bytes

Field Name	Offset	Length (bytes)	Required	Value
File Format	0	6	Yes	MLVLG padded by 0 or in hex x4D x4C x56 x4C x47 x00
Format Version	6	2	Yes	Currently x00 x01, increase values may be used in the future as the format is updated or enhanced.
Time Stamp	8	4	No	Unix 32 bit timestamp. Time in seconds since the epoch to the log start. If TimeStamp is not available set to all 0's - x00 x00 x00 x00
Info Data Start	12	2	No	This start offset must be after the LoggerField[] definitions and before Data Begin Index Set to 0 if no Log info Data.
Data Begin Index	14	4	Yes	The address of the 1st byte containing Type-Data pairs
Record Length	18	2	Yes	The length of a single data record for Logger Field data not including Type-Pair overhead
Num Logger Fields	20	2	Yes	Number of expected Logger Fields. A Logger Field Definition will be expected for each
Logger Field []	22	55 * n	Yes	This will be an array of (Num Logger Fields * 55) in length
Bit Field Names	22+n*55	varies	No	Only available if Logger Fields contains Bit
Info Data	Info Data Start	varies	No	Optional null terminated String. Unstructured informational data that may include the firmware version, date captured or any other informational text.

Logger Field – scalar

Field Name	Offset	Length (bytes)	Value
Type	0	1	0=U08, 1=S08, 2=U16, 3=S16, 4=U32, 5=S32, 6=S64, 7=F32
Name	1	34	ASCII Character Representation, null terminated
Units	35	10	ASCII Character Representation, null terminated
Display Style	45	1	0=Float, 1=Hex, 2=bits, 3=Date, 4=On/Off, 5=Yes/No, 6=High/Low, 7=Active/Inactive
scale	46	4	A IEEE 754 float representing the scale applied to (raw+transform)
transform	50	4	A IEEE 754 float representing any shift of raw value before scaling
digits	54	1	S08 representing the number of decimal places to display to the right

DisplayValue = (rawValue + transform) * scale
and

rawValue = DisplayValue / scale – transform

Logger Field - Bit

Field Name	Offset	Length (bytes)	Value
Type	0	1	10 = U08_BITFIELD, 11 = U16_BITFIELD, 12 = U32_BITFIELD
Name	1	34	ASCII Character Representation, null terminated. If empty string, this field will be suppressed and only the bit fields will be displayed
Units	35	10	ASCII Character Representation, null terminated
Display Style	45	1	0=Float, 1=Hex, 2=bits, 3=Date, 4=On/Off, 5=Yes/No, 6=High/Low, 7=Active/Inactive – Display for this field
Bit Field Style	46	1	0=Float, 1=Hex, 2=bits, 4=On/Off, 5=Yes/No, 6=High/Low, 7=Active/Inactive – Display for BitFields based on this field.
Bit Field Names Index	47	4	Index in file of Bit Field Names start. Array of null terminated strings to be used as bit field name. Array is defined by the bits field of this structure. To suppress display of a BitField, use “INVALID”
bits	51	1	Number of valid bits, (right justified) in this bitfield
UNUSED	52	3	Filler to maintain consistent Logger Field size.

Info Data:

Core Header	Bit Field Names	Info Data	Type-Data pairs.

Info Data is optional ASCII text that would normally be found in the header of a delimited log. It is unstructured informational data that may include the firmware version, date captured or any other informational text.

2.2 Data Block structure

Consists of "back-to-back" sequence of type-data pairs. Type-Data pairs allow for multiple data types to be interleaved.

2.2.1 Type-Data pair format

- Type (1 byte): unique identifier describing the structure, contents and length of the following data
- Data (length is determined by the type): single data item characterized by the preceding type identifier

Field Name	Offset	Length (bytes)	Required	Value
Block Type	0	1	Yes	Identifier for the type of block, a standard Data Block is type defined below
Counter	1	1	Yes	A rolling 1 byte block counter. Modulus of the record number.
Data block	2	Variable	Yes	Contains different data depending on type. See valid types below.

2.2.1.1 Type: Logger Field Data

Block Type: 0
Length: 4 + record length determined by adding the size of all Logger Fields defined in the Header

Field Name	Offset	Length (bytes)	Required	Value
Block Type	0	1	Yes	Identifier for the type of block, a standard Data Block is type 0 or x00
Counter	1	1	Yes	A rolling 1 byte block counter. Modulus of the record number. Counter % 255
Timestamp	2	2	Yes	A 2-byte Timestamp 10 us/bit.
Logger Field Record	4	Record Length	Yes	RAW data for the Logger Fields in the same order as defined in the header with no spacing.
crc	Record Length + 4	1	Yes	1 byte overflow from adding all values in the Logger Field Record, does not include header.

2.2.1.2 Type: Marker

Markers are positions with the data log that indicate a graphical mark generally displayed as a vertical red line in the log viewer to more easily locate specific events.

Block Type: 1
Data length: 54 - [2 byte time] [50 byte null terminated comment.] + 2-bytes for record data.
data description/format: Date/Time timestamp encoded in standard Unix 32bit format – *This is an Optional field, fill with 0's if correct time stamp unavailable.*

Field Name	Offset	Length (bytes)	Required	Value
Block Type	0	1	Yes	Identifier for the type of block, a standard Data Block is type 1 or x01
Counter	1	1	Yes	A rolling 1 byte block counter. Modulus of the record number.
Timestamp	2	2	Yes	If available, otherwise fill with 0's.
Message String	4	50	Yes	Null Terminated freeform string that will be displayed in the data log as the reason for the MARK