

Binary File Transmission

Product: P920380 - MIC6

Revision: 40579

Date: 2019-07-13 16:34:13

Distribution and release list (alphabetical)

Name	Company and Department	Rel.	Signature

History of changes

Revision	Date	Changes	Author	Status
40579	2019-07-13			
0.1	2013-02-25	Initial revision	Axel Ludszuweit	Draft
0.2	2013-03-11	Add description of data coding	Axel Ludszuweit	Draft
0.3	2013-03-13	Fix description of header in case of fixed data length	Axel Ludszuweit	Draft
0.4	2013-03-18	Fix type definitions for variable length	Axel Ludszuweit	Draft
0.5	2016-06-15	Adaptation for MIC6 Minor corrections (e.g. typos, white space)	Judita Kruse	Draft

Copyright © 2006 MOTORTECH GmbH		For internal use only	
Author	Axel Ludszuweit	Inspector	
Department	R & D	Department	
Signature		Signature	
URL (Revision)	svn://motdev01.motortech.local/development/projects/98.007.0244/trunk/900-Software/910-Requirements/J1939/J1939_BinaryFileTransmission.odt (40579)		
Creation Date	2013-02-25	Status	IUS (Inspected Updated Stored)

Document Management

Persons authorized to make changes

Name	Company	Department
Axel Ludszuweit	MOTORTECH GmbH	R&D Software Development
Judita Kruse	MOTORTECH GmbH	R&D Software Development

Tools used for the creation of this document

Tool	Description	Version
LibreOffice	Text processing tool	3.6.4.3
LibreOffice	Text processing tool	4.4.2.2

Content

1 INTRODUCTION.....	4
1.1 PURPOSE OF THIS DOCUMENT.....	4
1.2 FURTHER APPLICABLE DOCUMENTS.....	4
1.3 GLOSSARY.....	4
1.4 NOTATION.....	4
1.4.1 Byte Position of Binary File.....	4
1.4.2 Byte and Bit Positions of PGNs.....	4
2 BINARY FILE TRANSFER.....	4
2.1 BINARY FILE.....	5
2.1.1 Layout of File Header (Optional).....	5
2.1.2 CRC32 checksum (Optional).....	6
2.1.3 ID-Value List (Fixed Data Length).....	6
2.1.3.1 ID-Value List Item.....	6
2.1.3.2 Strings.....	6
2.1.3.3 Example.....	6
2.1.4 ID-Value List (Variable Data Length).....	7
2.1.4.1 Varints.....	7
2.1.4.2 Strings.....	7
2.1.4.3 Id-Value Items.....	8
2.1.4.4 Id-Value Lists.....	8
2.2 TRANSMISSION OF BINARY FILE FROM TOOL TO DEVICE.....	9
2.2.1 Request Memory Write Access via DM14 / PGN0xD9xy (xy=Sourceaddress).....	9
2.2.2 Sequence diagram.....	9
2.2.2.1 Transmission via one Memory Access (length <= 1784 Bytes).....	10
2.2.2.2 Transmission via more than one Memory Access (length > 1784 Bytes).....	11
3 COPYRIGHT.....	12

1 Introduction

1.1 Purpose of this Document

This document describes the layout and data coding of binary files transmitted via J1939 Memory Access (DM14, DM15 and DM16, SAE J1939-73) in conjunction with the J1939 transport protocol (SAE J1939-21).

1.2 Further applicable documents

SAE J1939-21
SAE J1939-73
Google Protocol Buffers(varints and zigzag encoding) <https://developers.google.com/protocol-buffers/docs/encoding?hl=de-DE>

The above mentioned MOTORTECH document is stored under:
`svn://motdev01.motortech.local/development/projects/98.007.0086/trunk/600-Miscellaneous/ld_value_list.odt`.

1.3 Glossary

Device	Ignition controller MIC6, which memory should be accessed
lsb	least significant byte/bit
msb	most significant byte/bit
TFloat32	IEEE 754 single precision floating point number (32 bits)
TFloat64	IEEE 754 double precision floating point number (64 bits)
TIntN	signed N-bit integer
TUIntN	unsigned N-Bit integer
Tool	Initiator of Memory Write Access
big-endian	values are stored with the msb first
little-endian	value are stored with the lsb first

1.4 Notation

1.4.1 Byte Position of Binary File

Bytes in binary files are numbered from 0 to (filesize – 1), bits are numbered from 7 (MSB) to 0 (LSB).

1.4.2 Byte and Bit Positions of PGNs

The bit / byte numbering is described in SAE J1938-71.

Byte 1								Byte 2								Byte 3								Byte 4							
8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1

Byte 5								Byte 6								Byte 7								Byte 8							
8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1

Bits are numbered from 8 (MSB) to 1 (LSB).

2 Binary File Transfer

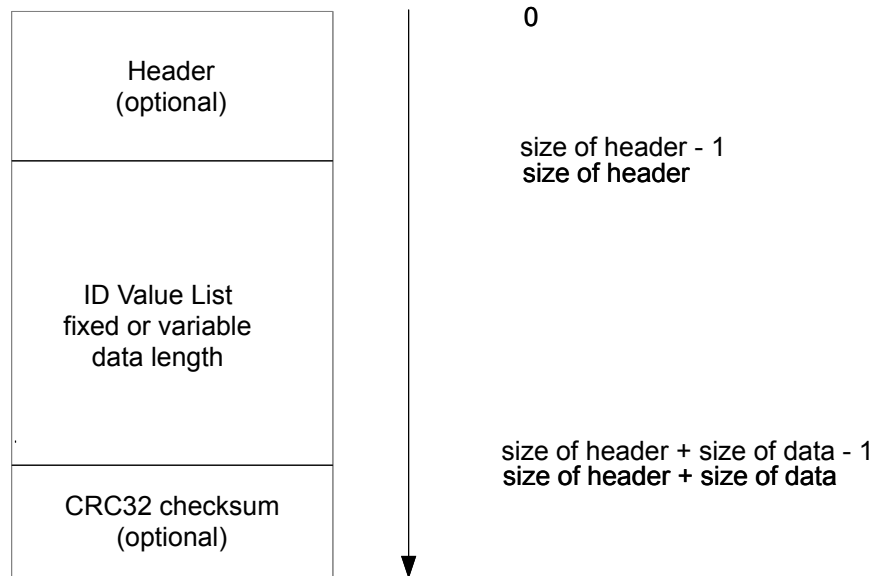
Until now, only Memory Write Accesses for configuration purposes are supported.

2.1 Binary File

Binary files can be transmitted with or without a file header and data can be coded via several methods. In case of data header the file contains also a CRC32 checksum in the last 4 bytes.

The parts of binary files are:

- Header (optional)
- Data (id-value list, with fixed or variable data length)
- CRC32 checksum (only if header is used)



2.1.1 Layout of File Header (Optional)

The header contains information about:

- Size of header (16 bit)
- Size of data (32 bit)
- Data type (16 bit)
- Version (16 bit)

Header data are big-endian coded.

Byte	Meaning	Remark
0	Size of header (most significant byte)	size of header in bytes
1	Size of header (least significant byte)	
2	Size of data (most significant byte)	size of data
3	Size of data	
4	Size of data	
5	Size of data (least significant byte)	
6	Data type (most significant byte)	enumeration of data type 0 – configuration data
7	Data type (least significant byte)	
8	Version (most significant byte)	version
9	Version (least significant byte)	

The *data type* describes the purpose of transmitted data, until now, only configuration data is foreseen. The header and data size should be part of the header. The size of header value points to the begin of data. To be future proof a version number and header size is part of the header.

2.1.2 CRC32 checksum (Optional)

The CRC32 checksum should be calculated from byte position 0 to sizeof(Header) + sizeof(data) – 1 with polynomial

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1,$$

initial value

0x04C11DB7

and stored in the last four bytes of binary file.

Byte	Meaning	Remark
size of header + size of data	most significant byte	CRC32 checksum about header and data
size of header + size of data + 1		
size of header + size of data + 2		
size of header + size of data + 3	least significant byte	

2.1.3 ID-Value List (Fixed Data Length)

All data are encoded in big-endian. A ID value list contains one or more ID value list item without padding or spare bytes between them.

2.1.3.1 ID-Value List Item

A ID-value list item is 3 byte long, the first three bytes contain a tag, to identify the meaning of the data, the last bytes describes the data format of the following data.

Byte	Meaning	
0	MSB	Tag
1	LSB	
2		data type

The data types are encoded as follows:

Value	Data Type
0	unsigned integer, 8 bit,
1	unsigned integer, 16 bit
2	unsigned integer, 32 bit
3	unsigned integer, 64 bit
4	signed integer, 8 bit
5	signed integer, 16 bit
6	signed integer, 32 bit
7	signed integer, 64 bit
8	IEEE 754 single precision floating point number (32 bits)
9	IEEE 754 double precision floating point number (64 bits)
10	string with max 256 byte length, ISO 8859-1 (Latin-1)
11 - 255	reserved

2.1.3.2 Strings

The first byte of a string must contain the length of the string. If the string is zero terminated the length including termination must be set.

2.1.3.3 Example

Transmission of a the ID-value items:

- Tag 1, uint8, 54
- Tag 2, int16, -4

- Tag 3, string "Hello"
- would create following binary data:
- 0x00, 0x01, 0x00, 0x36
 - 0x00, 0x02, 0x05, 0xFF, 0xFC
 - 0x00, 0x03, 0x0A, 0x05, 'H', 'e', 'l', 'l', 'o'

2.1.4 ID-Value List (Variable Data Length)

The original description of these data coding is stored under
svn://motdev01.motortech.local/development/projects/98.007.0086/trunk/600-Miscellaneous/Id_value_list.odt.

2.1.4.1 Varints

2.1.4.1.1 Unsigned Varints

Varints are used to encode integers using one or more bytes depending on the actual value. Each byte in a varint, except for the last one, has the msb set. The set msb indicates that the varint is continued in the next byte. The remaining bits of each byte are used to store up to 7 bits of the value. The encoding begins with the lower 7 bits of the value in the first byte of the resulting varint.

Examples:

The value 1 = 0000 0001 encoded as varint 0000 0001.

The value 400 = 0000 0001 1001 0000 encoded as varint 1001 0000 0000 0011.

Leading 0 bits are not stored because they are not needed to determine the value.

Although theoretically unlimited, varints are allowed for encoding source values of up to 64 bits.

2.1.4.1.2 Signed Varints

The two's complement representation of a negative number always has the msb set. If such a value is encoded as a varint as described in the previous section it would always occupy the maximum number of bytes allowed. This would be 10 bytes for 64 bit values.

Because of this a representation for signed integers where the number of used bits corresponds to the magnitude of the number is desirable.

This can be achieved by the so-called zigzag encoding.

Signed value	Encoded value
0	0
-1	1
1	2
-2	3
...	...

A signed number n stored in a N -bit variable can be encoded to an unsigned value x using

$$x = (n \ll 1) \oplus (n \gg (N - 1))$$

Note that arithmetic shifts must be used.

The previously encoded value x can be decoded using

$$n = (x \gg 1) \oplus \text{-static_cast< T>}(x \& 1)$$

2.1.4.2 Strings

Strings begin with length and encoding information stored in a single varint (LengthAndEncoding) followed by the string data bytes.

The lower two bits of LengthAndEncoding indicate the used encoding and the remaining bits indicate the string length in bytes. The length in bytes may be different from the number of characters in the string. Strings are not terminated by a trailing 0 or any other special character.

$\text{LengthAndEncoding} = (\text{LengthInBytes} \ll 2) \mid \text{Encoding}$

Encoding	Description
0	ISO 8859-1 (Latin-1)
1	UTF-8
2	reserved
3	reserved (encoding stored in first byte of string data, length includes this byte)

Example:

Encoding = Latin-1 (0)

LengthInBytes = 3

String = "ABC"

0000 1100 0100 0001 0100 0010 0100 0011

2.1.4.3 Id-Value Items

An id-value item is used to store a single value including an identifier and information on its data type. The first part (IdAndType) of an id-value item contains the id and type information and is stored as a single unsigned varint.

The lower three bits of IdAndType contain the data type and the next bits contain up to 61 bits for the id.

$\text{IdAndType} = (\text{Id} \ll 3) \mid \text{Type}$

Type	Description
0	Unsigned varint
1	Signed varint
2	reserved
3	TFloat32 (big-endian)
4	TFloat64 (big-endian)
5	String
6	reserved
7	reserved

After the first part follows the value encoded depending on the specified data type as the second part.

Example:

Id = 2

Type = unsigned varint (0)

Value = 400

0001 0000 1001 0000 0000 0011

2.1.4.4 Id-Value Lists

Id-value lists consist of concatenated id-value items. Information on the size of the id-value list is not part of the list itself and must be known by other means for working with an id-value list.

Items can be stored in any order but should not be reordered during transfer because the order might be relevant for some uses.

Lists or arrays of values can be stored using multiple items with the same id. In case there are multiple items with the same id the order of these items must not be changed because they are stored in the same order as the values in the source list or array.

2.2 Transmission of Binary File from Tool to Device

The binary file should be transmitted via memory access described in J1939-73. The authentication should be done by short seed/key exchange.

2.2.1 Request Memory Write Access via DM14 / PGN0xD9xy (xy=Sourceaddress)

In case of transmission of more than 1784 bytes the binary file must be divided into appropriate parts and transmitted via several *Memory Write Access Requests*.

Therefore the *Memory Write Access Request* must contain an offset and length value.

For example:

Size	Position	Name	Value / Meaning
11 bit	1.8 – 1.1; to 2.8 – 2.6	Length/Number Requested	Number of bytes transmitted (9 – 1784)
1 bit	2.5	Pointer Type	1
3 byte	3.8 – 5.1	Pointer	Offset in bytes
1 byte	6.8 – 6.1	Pointer Extension	see following table
2 bytes	7.8 – 8.1	Key	

<i>Pointer Extension</i>	<i>Meaning</i>
0x80	Configuration data without header, fixed data length
0x90	Configuration data with header, variable data length

In case of using configuration data without header, transmission of with more then 1784 bytes is not supported. The whole configuration must be transmitted via one transport protocol session.

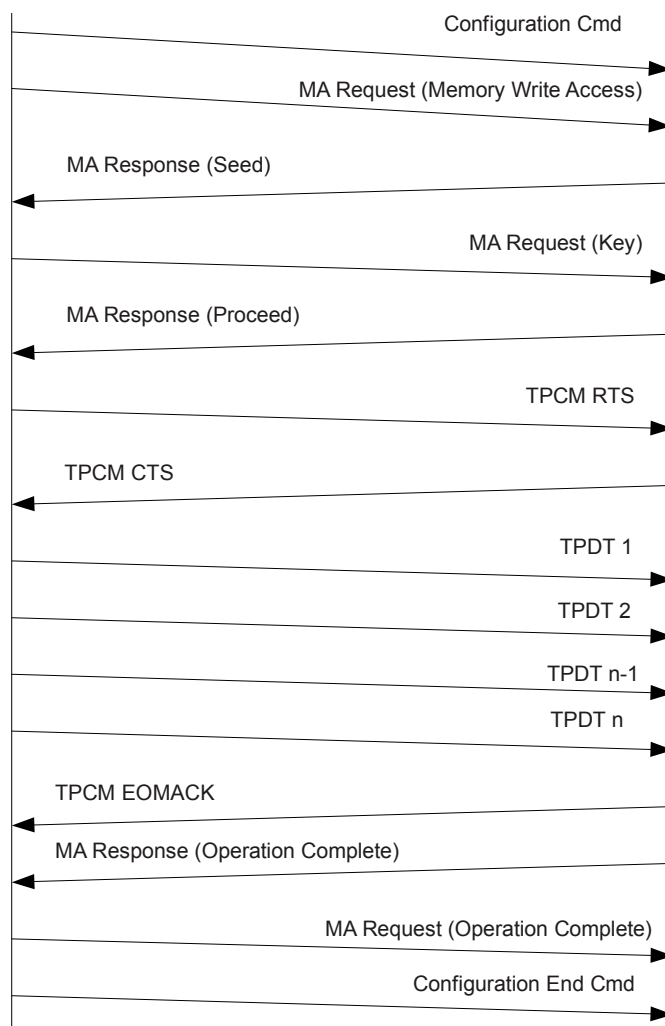
2.2.2 Sequence diagram

The following sequence diagrams show the transmission in case of one or multiple memory write accesses. The tool must switch the device into configuration state before transmission and leave configuration state after memory write accesses.

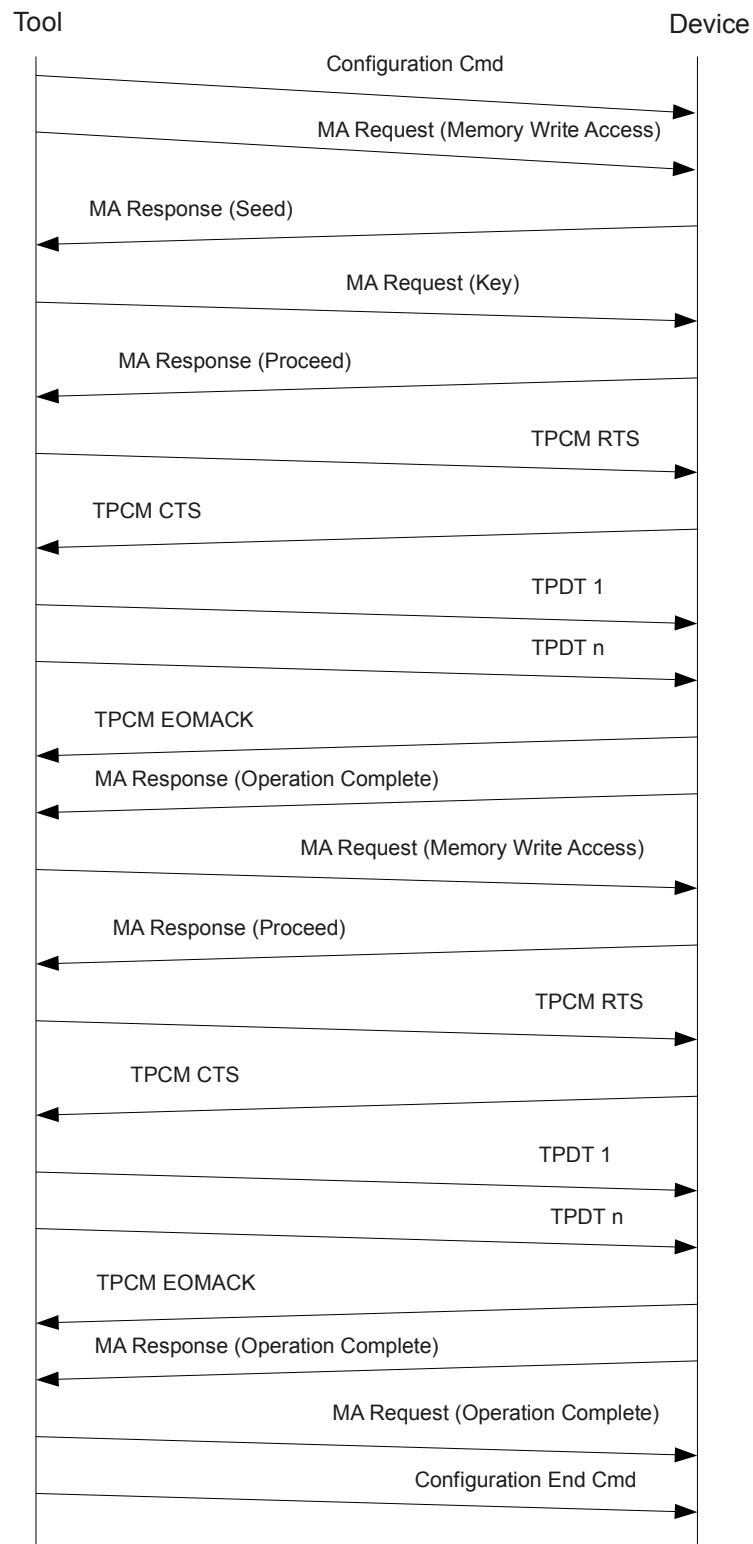
The ignition controller can be switched to/from configuration mode via a proprietary PGN.

2.2.2.1 *Transmission via one Memory Access (length <= 1784 Bytes)*

Tool



2.2.2.2 *Transmission via more than one Memory Access (length > 1784 Bytes)*



3 Copyright

Weitergabe sowie Vervielfältigung dieser Unterlage, ihre Verwertung und Mitteilung ihres Inhalts ist nicht gestattet, außer es wird ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder Gebrauchsmustereintragung vorbehalten.

Dieses Dokument liegt in seiner aktuellen Version bei der Projektleitung und kann von dort angefordert werden. Alle Änderungen werden an oben genannte Unterzeichner weitergeleitet.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without explicit authorization is prohibited. Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent or utility model.

The project management owns a current version of this document where it can be requested. All changes will be forwarded to the undersigned mentioned above.