ASAM

ASAM MDF

History

Origins

MDF was originally developed as a proprietary file format in the 90's by Vector Informatik GmbH and Robert Bosch GmbH. It was designed for use in the automotive industry, primarily for the areas of ECU development, calibration and testing. Since then, the format has evolved into a de-facto industry standard and is supported by many tools on the market, particularly by all leading tools in the measurement & calibration area.

One reason for the format's success may have been its stability and compatibility: The first public version was MDF 2.0 in 1991; MDF 3.0 was released in 2002. Over the years only gradual extensions were added. The current non-ASAM version MDF 3.3 is still fully backward compatible to all MDF 2.x and 3.x versions.

However, the file size of the proprietary MDF 3.x format was limited to 4 GB due to the usage of a 32-bit data type for file internal links. The rapid growth of measured data and thus file sizes was not foreseeable in 1991: After the turn of the century, this caused more and more requests for an update of MDF to support larger file sizes and to meet nowadays requirements. In addition, major OEMs expressed their desire for transferring MDF to an official industry standard.

ASAM Standard

As a result, a newly founded ASAM working group started in 2008 with the revision and standardization of MDF. This led to the release of ASAM MDF 4.0 in 2009.

ASAM MDF 4.0 overcomes the size restrictions of the previous MDF 3.x version and offers a range of new features like flexible extensibility via XML, custom signal grouping, events or attachments. However, due to fundamental changes like a 64-bit data type for links, MDF 4.x is not compatible to MDF 3.x any longer. Nevertheless, MDF 4.0 was adopted rather quickly by the industry and is supported by an increasing number of tools.

The latest release of ASAM MDF 4.1 in 2012 introduced:

- compression of measurement data
- memory-efficient storage for channels with constant value or variable data length
- storage of bus traffic for common bus systems
- storage of classification results
- storage of additional information about the measurement environment

Motivation

The measurement of signal values is an essential task for many applications. Most use cases not only require to process and display the acquired signals, but also to store them for documentation and/or post-processing. Typical signals for an automotive application are sensor data, ECU-internal variables/states, bus traffic in a vehicle network, or internally calculated values.

Many software applications still use proprietary file formats to store acquired or calculated measurement data. As a consequence, an exchange of data between different tools usually requires time-consuming data conversions that involve potential loss or alteration of information. The development of such converters is expensive and error-prone; hence, a commonly accepted standard format greatly improves the seamless exchange of data between tools.

Such a format needs to be well suited not only to store the measurement data itself, but also additional descriptive information about the measurement environment. A standard should also take into account special requirements for measuring bus traffic and ECU variables like Boolean data or endianness (Motorola or Intel convention). Due to the huge data rates already acquired today and expected to further increase in the future, a text-based format is not acceptable. Apart from file size, also the performance both for writing during measurement and for offline reading is an important aspect.

ASAM MDF was designed to meet the mentioned requirements and to team up with other ASAM standards like ASAM MCD-2 MC (ASAP2). Based on the widely accepted MDF 3.x format, ASAM MDF shows great promise to continue the success of its predecessor as a useful and stable standard format that is fit for today's and future needs.

Application Area

ASAM MDF focuses on the persistent storage of recorded or calculated measurement data for post-measurement processing, off-line evaluation and long-term storage. As standardized format it may also be used for data exchange between different tools.

A primary application area in the automotive industry is the recording of signal data during measurement, calibration and testing of ECU systems. Typically the signal values are acquired from sensors or transferred over a vehicle bus system; this includes recording of ECU internal variables (e.g. transferred via ASAM XCP) or of the complete bus traffic itself in form of bus events. Since the emergence of advanced driver assistance systems (ADAS), the recording of synchronized video data and radar information turned up as a new use case which is also supported by ASAM MDF.

Many recording tools not only store the measurement data and the corresponding signal description; they also need to document additional information about the measurement environment (like driving conditions, test equipment …) or even want to attach other files to the MDF file that are relevant for offline evaluation or reproduction of the measurement.

After the measurement, MDF files can be used to display the recorded signals in graphical or tabular representations. Offline evaluation tools offer manual or automated processing, classification and evaluation of the measurement data. Results of such an analysis can be stored as new MDF file or appended/attached to the input MDF file.

ASAM MDF files can also be used for a replay of measured signals or bus events, e.g. for stimulation in a software- or hardware-in-the-loop system (see ASAM XIL).

If a data logger application stores its measurements still in a proprietary format, it may provide a conversion to ASAM MDF to exchange the measurement data with other tools or to store it in a test data management system like ASAM ODS.

Technical Content

MDF Blocks

An ASAM MDF file (file extension: *.mf4) is organized in binary blocks where each block consists of a number of contiguous bytes that can be seen as a record or structure of data fields.

There are different types of blocks, and the block type defines the block's purpose and content. Each block type is abbreviated with two uppercase characters which are also used for the block identification ("block ID"). An overview of the block types in ASAM MDF is listed in the following table.

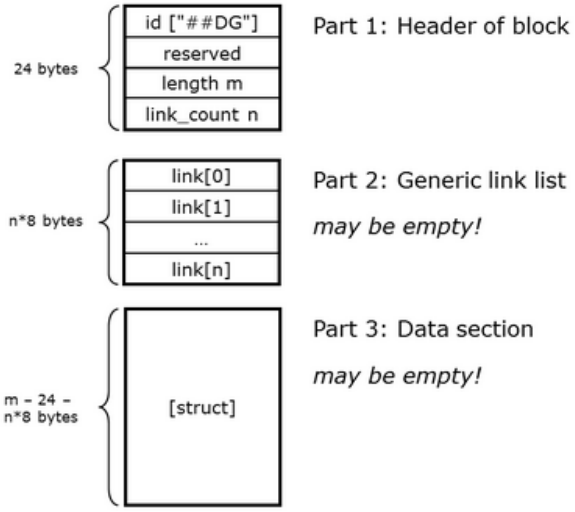
Abbreviation	Name	Decription
ID	Identification block	Identification of the file as MDF file
HD	Header block	General description of the measurement file
MD	Metadata block	Container for an XML string of variable length
TX	Text block	Container for a plain string of variable length
FH	File history block	Change history of the MDF file
CH	Channel hierarchy block	Definition of a logical structure/hierarchy for channels
AT	Attachment block	Container for binary data or a reference to an external file
EV	Event block	Description of an event (trigger, marker, …)
DG	Data group block	Description of data block that may refer to one or more channel groups
CG	Channel group block	Description of a channel group, i.e. channels which are always measured jointly
SI	Source information block	Specifies source information for a channel or for the acquisition of a channel group, for instance name and type of the sensor.
CN	Channel block	Description of a channel, i.e. information about the measured signal and how the signal values are stored.
CC	Channel conversion block	Description of a conversion formula for a channel
CA	Channel array block	Description of an array dependency (N-dimensional matrix of channels with equal data type) including axes.
DT	Data block	Container for data records with signal values
SR	Sample reduction block	Description of reduced/condensed signal values which can be used for preview
RD	Reduction data block	Container for data record triples with result of a sample reduction
SD	Signal data block	Container for signal values of variable length
DL	Data list block	Ordered list of links to (signal/reduction) data blocks if the data is spread over multiple blocks
DZ	Data zipped block	Container for zipped (compressed) data section of a DT/SD/RD block as replacement of such a block.
HL	Header of list block	Header information for linked list of data blocks

Each MDF block (with one exception) is divided into three sections: header, links and corresponding data section.

- The general header section contains the block identification (block ID), the block length and the number of links in the subsequent link section.
- The link section contains pointers to other blocks given as absolute byte positions within the file, starting at the beginning of the file. Thus a normally tree-like hierarchy of blocks is formed. The link section may be empty for some block types.
- The data section finally contains other properties of the MDF block depending on the block type.

The parts of an MDF block can be seen in the next figure.

The introduction of a general link section in MDF 4.0 has the advantage that a re-organization of the MDF blocks is possible even if there are unknown MDF blocks, i.e. MDF blocks introduced in a future MDF version: links to other blocks whose file position has changed can be adapted even if the content of a block type is unknown.



Each MDF block (except of ID block) consists of three successive parts

Block Hierarchy

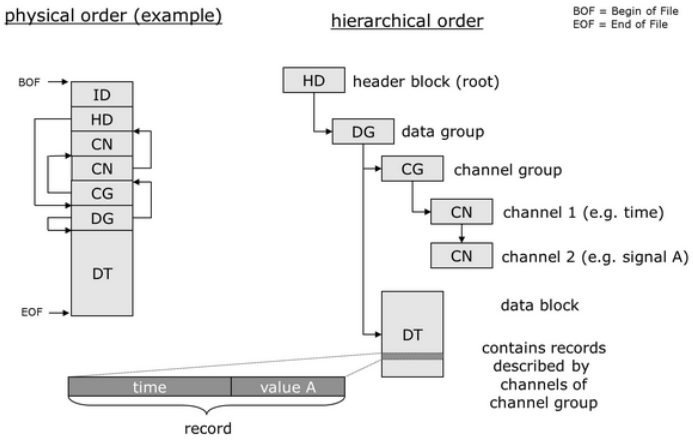
An MDF file starts with a “file identification” (ID) block which is exactly 64 bytes long. This is the only block that has no general block header and which is compatible to the previous MDF 2.x and 3.x format. Its main purpose is to identify the file as MDF file and to specify the MDF version.

Directly after the ID block, there is a “header” (HD) block which contains a general description of the MDF file, for instance the measurement start time or a link to the global file comment. The HD block can be seen as “root” of the MDF structure because it points to the start of several lists, most importantly to the linked list of “data group” (DG) blocks.

ID and HD block are the only block types which occur only once and which have a fixed position in the MDF file. All other block types have an arbitrary file position and may have several instances.

Independent of the physical ordering of the MDF blocks in the file, the links between the blocks form a usually tree-like hierarchy with the HD block at its top. As mentioned before, the most important “branch” in this tree is the list of data groups (DG blocks).

Each DG block refers to plain measurement data which is stored in so-called “records”. The records can either be contained in one single “data” (DT) block or distributed over several DT blocks using a “data list” (DL) block. Starting with MDF 4.1, the measurement data even can be compressed (using a DZ block). Since all the distributed and/or compressed measurement data can be consolidated to a single DT block, we will just consider one single “data block” for simplicity.



Example of a simple block structure of MDF files

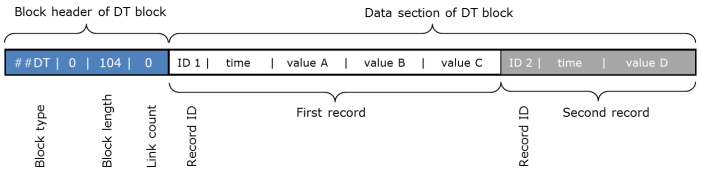
Each DG block not only points to a data block but also gathers all information that is necessary to understand and decode the data. One important task is to describe the layout of the records stored in the data block. For this, a DG block opens its own sub-tree with channel group (CG) blocks and channel (CN) blocks as next levels. The next section shows how the layout of a record is described by the channels of its associated channel group.

The above figure shows the block structure of a very simple MDF file, on the left side as physical order of blocks (one of many possible permutations), and on the right side as block hierarchy.

Record Layout

Each record contains signal values that have been acquired or “sampled” simultaneously, i.e. all signal values have the same time stamp. The layout of the record is defined by the channels of the record’s channel group. In case the data block contains records with different layouts, i.e. from different channel groups, each record must start with a “record ID”: it associates the record with one of the channel groups in the current data group.

As an example, the next figure shows a simple DT block with two records which have different layouts. Note that a DT block does not have a link section (link count is zero) and that the records in the data section must be stored without any “gaps”. The record layout for ID 1 will be described by one channel group, the record layout for ID 2 by another channel group. Both channel groups must be children of the DT block’s parent data group.



Example for a DT block containing two records with different layout

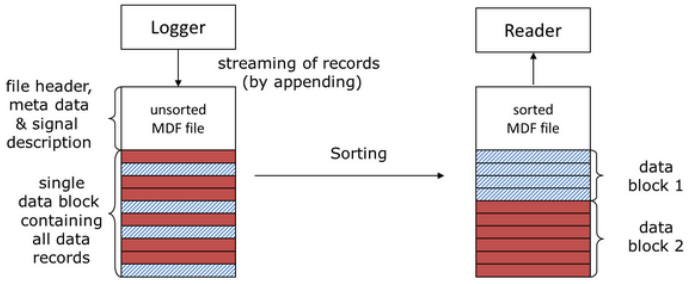
Sorted and Unsorted MDF

If a data group has only one single child channel group, its data block can only contain records with exactly the same layout. Such a data group is denoted as “sorted” in contrast to an “unsorted” data group with multiple channel groups. For a sorted data group, the record ID can be omitted. If an MDF file only contains sorted data groups, the complete file is denoted as “sorted”.

For instance, the example MDF file illustrated above is sorted because its only data group has only one channel group. Hence, the record does not require a record ID.

The record layout described by a channel group always has a fixed length. For a sorted data group it thus is very easy to calculate the start position of a record within its data block because all records have exactly the same length. This permits a fast index-based access to the records.

Another requirement for MDF demands that the time stamps (or generally, the master channel values, see below) must be strictly monotonic increasing within the series of records of equal type (record ID) in a data block. Together with the index-based access in sorted data groups, this allows to use a binary search for quickly finding the record for a certain time stamp (master channel value), i.e. to quickly jump to a certain position on the time axis. This is an important benefit for many MDF reading tools.

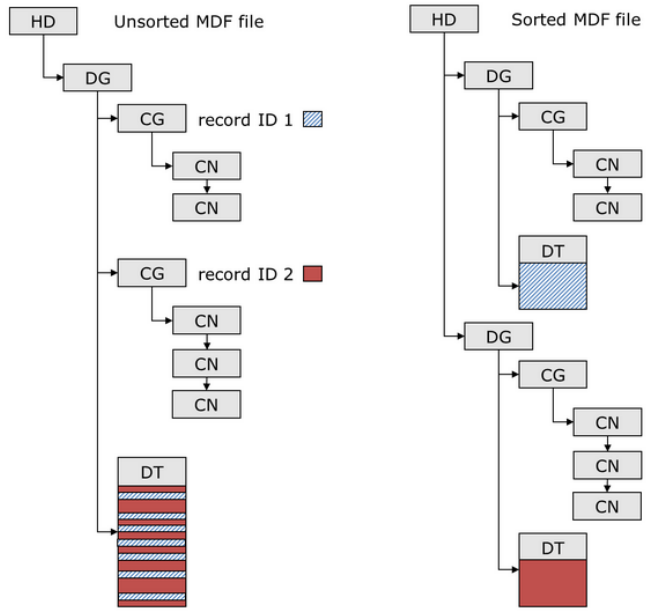


Data block structure within an MDF file before and after sorting

On the other hand, writing an MDF file with one single unsorted data group has the advantage that the records simply can be appended to the end of the file (see figure above). This “streaming” of records is very easy to implement because it avoids the necessity to buffer the records during recording. This may be important for data logger tools with restricted memory or which may be shut down by “power off” so that there is no time left for saving the buffered data.

The data block of an unsorted data group contains records with different layouts (record IDs) and possibly different lengths; as a result, reading and especially seeking a certain record within it is rather inconvenient because all records must be read from the start of data block until the required record is found.

Therefore most applications prefer to “sort” an unsorted MDF file before opening it for reading (see right side of the above figure) The sorting of MDF is a loss-free transformation where the records of unsorted data groups are separated and stored in an own data block for each record ID. Typically the



Block hierarchy within an MDF file before and after sorting

record IDs are omitted to reduce the file size. The channel groups of an unsorted data group therefore also must be separated so that each gets an individual parent data group which is a copy of the original DG block. The above figure shows an example MDF block hierarchy before and after sorting.

Sorted Writing using Distributed Data Blocks

Unsorted writing as explained before has the disadvantage that usually a time-consuming sorting step is required for efficient navigation within the MDF records. An alternative is to immediately write sorted MDF files. “Sorted writing” requires a bit more implementation effort and the ability of data buffering during measurement.

The introduction of distributed data blocks in ASAM MDF 4.0 was essential to support a practicable implementation of sorted writing: Instead of writing one huge DT block, an application now may decide to write smaller “chunks” of measurement data in separate DT blocks. A data list (DL) block keeps the single DT blocks together.

Writing measurement data in distributed data blocks frees the writer application from buffering the complete measurement. Instead only smaller memory buffers are required to collect the measurement data in form of records. For sorted writing, each channel group (i.e. each raster) gets its own buffer. If one of the buffers is full, its content is written to the MDF file as DT block. Then the buffer is cleared and filled again. In the end, or already during the measurement, the writer application needs to write the DL block - or a linked list of DL blocks - for keeping the single DT blocks together. So an extra buffer is necessary to collect the DL block information like the start addresses of the DT blocks.

Note that there are special cases where sorted writing can be achieved without any buffering. One case is when the writer application already knows the amount of measurement data for each raster beforehand, for instance when carrying out offline conversion from some other file format. Here the start addresses and sizes of the DT blocks in the sorted MDF file and the position of each record can be calculated beforehand.

Compressed Data Blocks

ASAM MDF 4.1 introduced the possibility to optionally compress data blocks using a new block type “DZ” (for “data zipped”). It uses the commonly known “Deflate” algorithm for lossless compression of the records. A very popular implementation of the algorithm is the free software library “zlib” (see external link section) which can be used for both compression and decompression.

The DZ block also defines an optional transposition of the measurement data before compression which leads to a usually much better compression rate. Depending on the measurement data, the resulting file size can be significantly smaller than compared to a none-transposed, zipped MDF file.

In order to compress and decompress measurement data “on the fly”, usually the complete content of the DZ block must be loaded into memory. To avoid extreme memory requirements, the maximum size of the uncompressed data was restricted to 4 Mbyte. If this size is exceeded, the data must be distributed to several DZ blocks.

Note that the newly introduced block type in MDF 4.1 means that tools supporting only MDF 4.0 will be able to read the MDF block structure for the descriptive data but will not be able to decode the compressed measurement data. In general, this statement applies to all new features introduced in future MDF versions.

Channels

The possibly most important elements in an MDF file, apart from the data blocks, are the channels (CN blocks). Each channel describes a recorded signal and how its values are encoded in the records for the channel’s parent CG block.

Each channel has a name, typically the name of the recorded signal. It is also possible to specify “alternative” names, for instance a shorter “display name”.

Note that an MDF file may contain signals from different sources (sensors/devices/ECUs), and the same signal name may be used for different sources; as a consequence, the channel name alone is not sufficient for a unique identification of a channel within an MDF file. Instead, ASAM MDF uses for this the channel name in combination with other fields that contain (optional) information about the channel's "source" and "acquisition" (information about the acquisition is used because MDF even allows storing the same signal for different acquisition modes - i.e. different sampling rates - within the same file). The associated standard "ASAM MDF Naming of Channels and Channel Groups" shows typical use cases and gives recommendations where to store which kind of information.

Whereas fields and flags in the data section of the CN block describe the characteristics of the recorded signal and its value encoding in the record, text and more complex information – like the channel name or source –are stored in additional MDF blocks referenced from the link section of the CN block.

Text and Meta Data Information

Strings for descriptive information in MDF are encoded in UTF-8, either in a "text" (TX) block or, when using XML fragments, in a "meta data" (MD) block. UTF-8 encoding allows storing characters of any language like Japanese or Chinese, but minimizes the required memory for characters typically used for Western languages: For instance ASCII characters still only require 1 byte.

Important information - like the channel name - only can be stored in a TX block for fast reading without parsing of XML. Less important or generic information like the alternative channel names is stored in MD blocks, for instance the one used for the channel comment.

The MDF block types HD, DG, CG, CN and many others offer to store commentary information as XML fragments (MD blocks). This allows choosing the hierarchy level with the best matching scope in order to avoid unnecessary duplication.

XML Fragments and Generic XML Tags

Each XML fragment stored in a MD block must follow the rules of a specific XML schema (XSD) file depending on the origin of the MD block link. For example, if an XML fragment is used as comment of a channel, it must contain a <CNcomment> tag as main tag.

Within the main tag, there must be always a <TX> tag with the main text, i.e. the plain comment text. If no other tag is used except of the <TX> tag, a TX block may be used instead of the MD block. After the <TX> tag, there can be a number of schema specific tags, like for storing the alternative names of a channel. In most cases, these additional tags are optional.

The XML schema for comments always contains two final sections: <common_properties> and <extensions>. Both are optional. Whereas <extensions> may be used to store tool-specific XML which follows a tool-specified schema, the <common_properties> can be used to store information in generic XML tags.

The most important generic XML tag is the <e> tag. It simply defines a key-value pair, where the key is given as attribute of the tag, and the value as its content. Other optional attributes can be used to specify further properties like the data type, unit, language, description or read-only state of the content. Further generic tags may be used to define a tree or list structure for the name-value pairs.

Generic XML tags serve to store information that should be tool-independent and that is not covered by schema specific tags or binary attributes of the MD block's parent block. Any tool should be able to display the information in the generic tags without understanding the semantics; it even may allow its users editing the values.

The two associated standards "ASAM MDF Measurement Environment" and "ASAM MDF Classification Results" define application models based on generic XML tags for storing additional descriptive information about the respective topic.

Conversion Rules

The signal value encoded in the record is denoted as "raw" value. It can be converted to a "physical", human-readable value using a conversion rule assigned to the channel that describes the signal.

Conversion rules are typically used to optimize memory consumption for values of ECU internal variables (see ASAM MCD-2-MC) and for signal values encoded in bus messages, e.g. in CAN messages. For instance, a single unsigned byte value could be translated to a physical range of -5 Volts to +5 Volts. Another use case is to assign human-readable strings to certain signal values, often denoted as "status texts", e.g. to describe specific states or error codes.

The input of a conversion rule either is a numeric value or a string (see Channel Data Types below). Its result again either is a numeric value (by definition a 64 bit floating point value) or a string (by design always UTF-8 encoded due to the usage of TX blocks in tabular look-up conversions).

Of course, physical values can be directly stored in the record. For this either an identity conversion can be used, or the conversion rule simply can be omitted. In both cases, the raw value can be used immediately as physical value, without any data type or string conversion.

ASAM MDF currently supports the following conversion rules to convert the raw signal value to a physical value:

- Identity (“1:1”) conversion (also assumed if no conversion rule is specified)
- Linear conversion (2 parameters: factor * X + offset)
- Rational conversion formula (6 parameters)
- Algebraic conversion (like ASAM MCD-2 MC text formula, e.g. “PI*(X*X)/ 2”)
- Value to value tabular look-up with interpolation
- Value to value tabular look-up without interpolation
- Value range to value tabular look-up
- Value to text/scale conversion tabular look-up (often used for status texts)
- Value range to text/scale conversion tabular look-up (often used for status texts)
- Text to value tabular look-up
- Text to text tabular look-up (translation)

The capability of MDF to store raw signal values and conversion rules has several advantages:

- the conversion to a physical value can be omitted during online measurements (performance optimization)
- the record size and thus the file size may become significantly smaller (memory optimization)
- receiving a complete data chunk containing several signals for instance from an ECU or via a bus message can be stored “en-bloc” in the record without any signal extraction (performance optimization)
- during offline analysis both raw values and converted physical values can be analyzed

Channel Data Types

The encoding of the (raw) value of a channel in a record is defined by the position (byte and bit offset), the number of used bits and the data type of the value. ASAM MDF currently supports the following channel data types (LE = Little Endian/Intel convention, BE = Big Endian/Motorola convention):

Integer data types (1-64 bit):

- unsigned integer (LE byte order)
- unsigned integer (BE byte order)
- signed integer (two’s complement) (LE byte order)
- signed integer (two’s complement) (BE byte order)

Floating-point data types (32 or 64 bit):

- IEEE 754 floating-point format (LE byte order)
- IEEE 754 floating-point format (BE byte order)

String data types (any number of complete bytes):

- String (Single Byte Character ISO-8859-1 Latin encoded, NULL terminated)
- String (UTF-8 encoded, NULL terminated)
- String (UTF-16 encoded LE Byte order, NULL terminated)
- String (UTF-16 encoded BE Byte order, NULL terminated)

Complex data types (any number of complete bytes, except CANopen types):

- Byte Array with unknown content (e.g. for a structure)
- MIME sample (sample is Byte Array with MIME content-type specified for the CN block)

- MIME stream (all samples of channel represent a stream with MIME content-type specified for the CN block)
- CANopen date (Based on 7 Byte “CANopen Date” data structure)
- CANopen time (Based on 6 Byte “CANopen Time” data structure)

Note that a bit offset is only allowed for Integer data types. This quite often is used for “bit fields” as in the C/C++ programming language, again to save memory for variables or signals in bus messages.

For complex data types currently there are no conversion rules defined. “Byte Array” type is the most general complex data type because it allows storing any kind of data “blob” as signal value. It is also used for “structure” channels (see section about Structure Channels). With types MIME sample and MIME stream, it is possible to specify the content of such a blob. This may be used, for example, to store image data as signal values where the image format can be documented with the respective MIME type (for example “image/jpeg”). Another way to save synchronized video and audio data uses the special “synchronization” channel type which will be explained below.

Channel Types

ASAM MDF knows different types of channels. Each channel type adds some meaning to the signal values of the channel. Currently the following channels types are supported:

- Fixed length data channel
- Virtual data channel
- Master channel
- Virtual master channel
- Variable length signal data channel (VLSD)
- Maximum length signal data channel (MLSD)
- Synchronization channel

Value Channels with Fixed Length

The **fixed length data channel** is the most commonly used channel type. It simply denotes that this channel describes an ordinary signal whose raw values are stored in the record with a fixed length, i.e. in a fixed number of bytes.

The **virtual data channel** - introduced in ASAM MDF 4.1 - indicates that its signal value is not stored in the record. Instead the record index is used as raw value; it can be converted by the channel’s conversion rule to a physical value. The virtual data channel can be used to define a signal whose values can be calculated from the record index, or which only has a single constant value (implemented by a linear conversion rule with factor equal to zero and offset equal to the constant value).

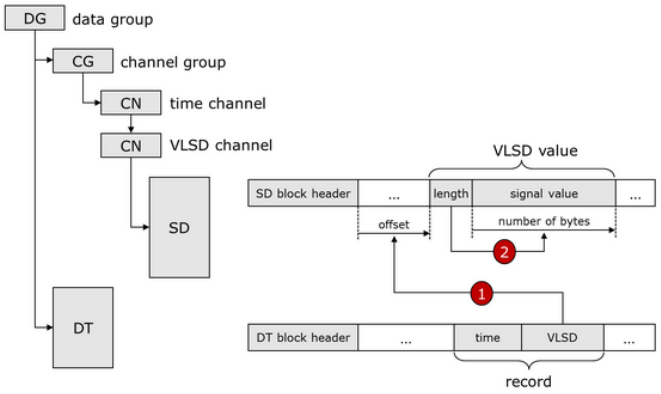
Value Channels with Variable Length

As mentioned earlier, the MDF record for a specific channel group must have a fixed length. However, for strings or complex data types, the size of the value may vary, i.e. string values may have different lengths. Reserving a maximum length for the signal value in the record may only be a solution if such an upper limit can be determined at start of the measurement. But even if this is possible, it may mean unnecessary wasting a lot of space in the record and thus blowing up the file size.

ASAM MDF 4.0 therefore introduced the **variable length signal data (VLSD) channel**type which allows the efficient storage of signal values with variable length. Here the channel value in the fixed-length record is only an Integer offset, independent of the data type of the channel. The actual signal value is stored as so-called “VLSD value” at the given offset in a separate “signal data” (SD) block which must be referenced by the VLSD channel. Like DT blocks, SD blocks can be distributed and/or compressed. Each VLSD value in the SD block starts with a 4-byte Integer length value that specifies the number of bytes used for the actual signal value, for instance the text of a string value. The next figure illustrates the steps to read a VLSD value.

The indirection for storing VLSD values keeps the length of the main record fixed at the cost of having to read another file location in order to read the signal value itself.

Note that ASAM MDF also supports the “streaming” use case for VLSD values, i.e. to write an unsorted data block with VLSD values. For this, a special VLSD record is written to the data block which must start with an own record ID for the VLSD channel, followed by the VLSD value (4-byte length + respective number of bytes for the signal value). VLSD records are only allowed for unsorted data blocks and must be transferred into an SD block during sorting.



Block hierarchy and record layout for variable length signal data

ASAM MDF 4.1 also introduced a second channel type with variable but **maximum length signal data (MLSD),** which can be applied in case the upper limit of the VLSD value is known. It reserves the respective number of bytes in the record and defines a second signal (in the same record) for the actual size of the channel value. A use case for the MLSD channel type is to store the payload of CAN messages without any unnecessary overhead where the variable length payload typically is limited to a maximum of 8 bytes (see associated standard ASAM MDF “Bus Logging”).

Master Channels for Synchronization

A **master channel** is stored in the record just like a fixed length data channel whereas a **virtual master channel** uses the record index as raw value in the same way as a virtual data channel. What makes both types particular is that their signal values can be used to synchronize different channel groups or events in one of the “synchronization domains” of ASAM MDF.

A synchronization domain allows the ordering of recorded samples or events with respect to their occurrence. Typically, time is used as synchronization domain, i.e. each sample and each event in the MDF file stores a time stamp based on the same clock. For special use cases, ASAM MDF also defines distance, angle or (only implicitly) index as additional synchronization domains.

A master channel for the synchronization domain “time” – or short: “time channel” – thus simply defines how the time stamps are stored in the record. Its physical values must be in seconds; however internally (as raw value) very often a higher resolution is used, for example nanoseconds.

Values of a (virtual) master channel listed over the record index must be strictly monotonic increasing. They are always relative to the respective start value in the HD block, i.e. the absolute start time/date of the measurement for the time channel.

Synchronization of Streams (Video/Audio)

Another channel type, called **synchronization channel**, allows synchronizing the MDF record with another stream, typically a video or audio stream. Here the data of the stream is not stored in the record as MIME type value; instead it is stored in a separate file format, for instance as AVI file. This file then is attached to the MDF file, either as external link or as embedded data (see attachments below), and the attachment is referenced by the synchronization channel. The value of the synchronization channel is either an index or a value from one of the synchronization domains in ASAM MDF and must be used to synchronize with the stream, assuming that the stream uses the same synchronization domain or indexed frames.

Composition of Channels

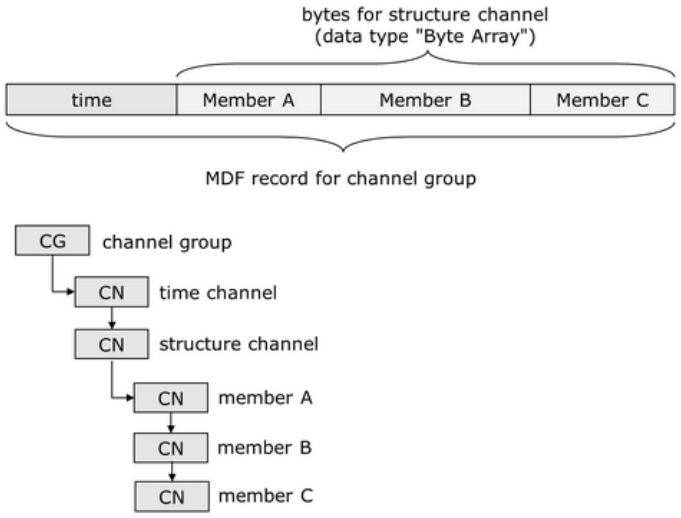
MDF is able to describe channels that are composed of other channels. Both composition types natively supported in ASAM MDF are known from most programming languages: structures and arrays.

A composed channel is said to have component channels - or simply “components” - which are seen as children of the composed channel. The components of an array are its array elements, for a structure these are its members. All array elements must have exactly the same size and data type whereas this does not apply for structure members. Note that a component channel again may be a composition, i.e. the member of a structure or the element of an array again can be a structure or an array itself.

Typically all components of the composed channel are acquired simultaneously, i.e. they are all stored in one channel group and all bytes of the structure or array are stored in the same record. This could be denoted as “compact” storage in contrast to a “fragmented” storage where the components are acquired independently. ASAM MDF supports both use cases, but since the fragmented storage is rather exotic, we will only explain the compact storage here.

Structure Channels

A structure with compact storage is represented in ASAM MDF by a parent CN block for the complete structure which points to a linked list of child CN blocks for the members of the structure. The structure parent channel always has the data type “Byte Array” and describes a range of contiguous bytes in the record that must contain the bytes for all member channels. The right figure illustrates the block hierarchy and record layout for a structure channel with three member channels. Each member channel can have a different size and data type.



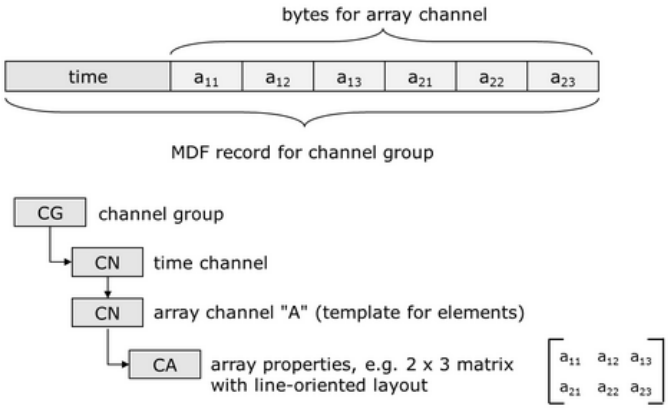
Block hierarchy and record layout for a structure channel

Array Channels

An array with compact storage is represented in ASAM MDF by a "template" CN block for the array elements which has a child "channel array" (CA) block that defines the array layout and other array properties like its axes. The next figure shows an array channel with 2 x 3 array elements where each element channel is derived from the template CN block by changing the byte offset.

An array in ASAM MDF can be a one-dimensional vector or a multi-dimensional matrix, and it can be used to model "maps" and "curves" as known from ASAM MCD-2 MC including properties like fixed or variable axes and input quantity ("working point") signals for each dimension.

Typically the size of an array – i.e. its number of elements per dimension – is fixed. However, ASAM MDF also supports arrays with variable size: Similar to the MLSD channel type, a size channel can be used to indicate the current size for each dimension while its maximum size is limited, i.e. each record needs to reserve the maximum possible number of bytes for the array.



Block hierarchy and template for an array channel

Special Features

Classification Results

Array channels in ASAM MDF can also be used to store "classification results" like a frequency distribution or a histogram. Instead of axis "points", a special axis type with value ranges can be used for this. Classification parameters and other properties can be stored in the comment of the template CN block using generic XML tags as explained in the associated standard "ASAM MDF Classification Results".

Bus Logging

The associated standard “ASAM MDF Bus Logging” describes how to store the bus traffic for commonly used bus systems like CAN, LIN, FlexRay, MOST and Ethernet in the form of bus events in an MDF file. The application model for the bus events is based on structure channels (see above) and defines mainly the channel names and the expected content of the bus event structures and its members. For instance, when storing a CAN message the structure must be called

“CAN_DataFrame”, and it must contain a few mandatory member channels like “ID”, “DLC” and “DataBytes”. The standard also defines optional members (e.g. “CRC”) and even allows custom members as long as their channel name does not conflict with the standardized members.

Attachments

Attachments can be used to refer to an external file or to embed it into the MDF file. For external references, the integrity may be secured by calculating the MD5 check sum and storing it in the “attachment” (AT) block. Embedded attachments optionally can be compressed, again using the deflate algorithm as for compressed data blocks.

Events

Events in ASAM MDF can be used to store

- events that influenced the recording like begin, end or an interruption
- conditional events, i.e. triggers
- markers, i.e. comments related to a point in time or a time range

Each “event” (EV) block contains a synchronization value to specify when the corresponding event occurred. Typically this is a time stamp, but the event can also be synchronized with some other master channel type or the record index of a channel group. Generally an event defines a "point" in time or some other synchronization domain. For some event types, two "points" can be used to define a "range".

The “scope” of an event defines which channels it applies to. In addition, each event optionally can have a name, a comment and references to attachments.

Sample Reduction

The “sample reduction” feature of ASAM MDF allows high-speed drawing of signals that have millions of samples in their original data set. For this, a “reduced” set of signal data with a lower resolution (less sampling steps) can be stored as alternative to the original data. Each sample of the reduced data contains the minimum, maximum and a (pseudo) average value that occurred for the original signal within the “reduction interval”. Typically several sample reduction resolutions are stored in the MDF file so that an application can choose one that is matching best for the current display resolution. The high-speed access advantage of the “sample reduction” feature comes at the cost of an increased file size.

Invalidation Bits

“Invalidation bits” serve to mark signal values in a record as “invalid” (sometimes also denoted as “no value”). Invalidation bits are an optional feature and require at least one additional byte in the record.

Channel Hierarchy

The channel hierarchy can be used to build an own logical structuring of channels totally independent of channel groups or structure and array channels. Each “channel hierarchy” (CH) block represents a node in a nested tree where each node may contain multiple leafs with references to channels. A channel even may be referenced more than one time or not at all.

Unfinalized MDF

In case of "power off" or a system crash, the writer application may not be able to correctly finalize the MDF file. For instance, the cycle counters stored in each channel group may not be up to date. In order to detect such a problem, the MDF file can be labelled as “unfinalized MDF” including information about necessary steps to correctly finalize the MDF file later. The "unfinalized" label in the ID block should be replaced after correct finalization, either at the end of the measurement or after offline finalization. This feature may be used for any MDF format version, even for pre-ASAM MDF.

File History

The file history – a linked list of (FH) blocks - documents the creator of the MDF file (tool name, version number, etc.) and should also list all tools that have altered the MDF file.

Relation to Other Standards

ASAM MDF is closely related to ASAM MCD-2-MC (aka ASAP2) which is a description format for internal ECU variables. This means that most concepts, data types, conversion rules and description fields from ASAM MCD-2-MC can easily be stored in ASAM MDF, which allows a seamless transfer of ECU signal descriptions.

ASAM XIL uses ASAM MDF for streaming of measurement data.

ASAM ODS can use ASAM MDF files if the files are in compliance to some basic rules listed in the ASAM ODS specification. This allows the mixed-mode operation of an ODS server, where data is either directly stored inside a relational database or externally stored on a file server and just referenced via meta data in the database. The latter is usually done for large amount of data, which cannot be efficiently stored inside a relational database.

Furthermore, ASAM ODS re-uses the ASAM MDF application model for bus logging. The description of classification results in ASAM MDF refers to elements and definitions from ASAM ODS.

Benefits & Advantages

ASAM MDF as a binary format allows a compact storage of huge amounts of measurement data, especially with the optional compression introduced since ASAM MDF 4.1.0.

The use of a 64 bit data type for links and other properties helps to overcome the limitations of the predecessor format (MDF 3.x). This allows, at least from the current point of view, a practically unlimited file and measurement data size and promises that the format will fulfill size requirements for the next decades.

ASAM MDF allows high data rates for both reading and writing, which makes the format capable for many real-time applications. For instance, it is possible to directly write the raw values received from an ECU. The respective conversion rule is saved in the channel description and can be applied during offline reading. Fixed-formatted data chunks received from an ECU even may be stored “en-bloc” which helps to further decrease CPU load during a measurement.

Simple logger applications may implement “unsorted” MDF writing by simply appending the records to the end of the file. For reading, a loss-free re-organization of the file ("sorting") allows the usually desired fast index-based access to the samples, e.g. for fast binary search in the time axis.

More sophisticated tools may implement writing directly “sorted” MDF files by using the distributed data blocks in MDF 4.0.0. As a result, the possibly time consuming “sorting” step before opening a file for reading can be avoided.

Due to its organization in loosely coupled blocks, it is possible to read an ASAM MDF file only partially and to access only the descriptive information without loading any signal data. With sorted MDF files, tools can read only the currently required part of the signal data which may be the only practical solution in case the amount of signal data exceeds the available memory.

ASAM MDF supports multiple and non-periodic sampling rates. Synchronization is achieved via a master channel concept. Signals are typically recorded based upon time, but it is also possible to use angle- or distance-based recording. Being able to combine signal values always recorded simultaneously into one record has the advantage that only one master (i.e. time) value must be stored.

ASAM MDF stores the description of measurement data within the same file as the data itself. In addition, ASAM MDF can refer to other files or even embed those using so-called attachments. This can be used for documentation or for combining important information required for later offline analysis or reproduction of a measurement.

Important and commonly used descriptive information is stored in compact binary blocks. Optional and customized information can be stored in XML fragments on different hierarchy levels (e.g. file, channel group or channel level). Generic XML tags allow easy storage and display of your custom information even by other tools.

Although ASAM MDF is a general purpose file format for storing signal data and descriptions, it is primarily designed for use in the automotive area. The standard supports special data types and information particularly required in the automotive area, for example structures and arrays (curves/maps), bus events and synchronized video data.

Last not least, ASAM MDF has widely been adopted by the industry and is supported by many leading tools.

Industry Adoptions

The primary application area of ASAM MDF is the measurement of ECU software and sensor values. All major ECU measurement and calibration tools either use ASAM MDF as their native recording format or provide importers and exporters for the format.

A wide range of offline evaluation tools—esp. those used in the automotive area—provide import and sometimes also export of ASAM MDF.

Data logger tools that are able to record bus traffic on CAN, LIN, FlexRay or Ethernet started to use the bus logging capability of ASAM MDF by directly writing MDF files with bus events or providing a converter from their own proprietary format.

Tools supporting the ASAM XIL API (see ASAM XIL) also support ASAM MDF.

As a relatively new application area, ASAM ODS servers (see ASAM ODS) support ASAM MDF files by using them as means to externally store data.

Open-source and commercial library solutions for reading and writing ASAM MDF have become available on the market, which help to speed up implementation. Those libraries and the plan of major OEMs to establish ASAM MDF as company standard will entail a growing tool support.

List of Deliverables

The standard includes the following deliverables:

- Base Standard
  - Standard document for base standard
  - XSD schema files for XML fragments
  - MF4 example files for different MF4 features
- Standard documents for associated standards
  - Bus Logging
  - Measurement Environment
  - Naming of Channels and Channel Groups
  - Classification Results

Downloads

The downloadable archive contains an mf4-file and the corresponding a2l-file for demonstration purposes. The archive also includes further files that are typically used in an ECU measurement and calibration project.

ECU_Description.zip	ZIP-archive including files for the standard formats a2l, cdfx and mf4.

External Links

- Vector MDF Validator for standard versions 2.x, 3.x and 4.x, free download.

- Vector MDF 3.3, predecessor of ASAM MDF 4.x, included in the download of the Vector MDF Validator.
- Turbolab MDF4 Open-Source Library, for reading, writing and viewing of MDF4 files
- Zlib for compression and decompression of measurement data and attachments