ASAM MCD-2MC
Version 1.6

Measurement and Calibration Data Specification

Released

Association for Standardisation of
Automation and Measuring Systems

Dated:01.02.2009

Status of Document

Date:	01.02.2009	
Author:	ASAM MCD-2MC Workgroup:	
	Continental Automotive GmbH	Kunz, Hans-Georg
	dSPACE GmbH	Amsbeck, Hendirk
	ETAS GmbH	Wenzel, Thilo
	M&K GmbH	Wenzel, Bernd
	Robert Bosch GmbH	Bauer, Hartmut
	Vector Informatik GmbH	Lampert, Peter
	Vector Informatik GmbH	Schnorr, Elke
Version:	Version 1.6	
Doc-ID:		
Status:	Released	
Type	Specification	

Revision History

See What's New slides of Version 1.6 [What's New]

Table of contents

1 INTRODUCTION

1.1 ASAM MCD-2MC OVERVIEW

First versions of the standard were developed already before foundation of ASAM e.V. in 1998. These versions were named ASAP-2. They have been part of a 3-layer base architecture, which is described in the next chapter.
ASAP-2 was renamed to ASAM MCD-2MC. MCD stands for Measurement, Calibration and Diagnostics.
Calibration means the adaption of characteristics (scalars, curves and maps) within the functional code of ECUs to achieve and optimize an appropriate system behavior. These calibration operations are either performed manually by a calibration engineer or are executed by external client applications, such as optimization programs or test bed automation systems.
This adaption requires a WRITE-access to the ECU to set a new value or a READ-access to retrieve the current one.
Whether this calibration already produces the intended effect or not will normally be checked by inspection of other ECU variables by MEASUREMENT access. Therefore the host tool configures so-called measurement tasks, transmits them to the ECU and henceforth takes the values which are automatically sent by the ECU.
For all of these variables elementary information like addresses, data types, dimensions, identifiers or other more descriptive data are formulated in ASAM MCD-2MC.
The ECU normally stores the measurement and calibration quantities internally in an implementation optimized format. This format is very often a fixed-point format. Outside the ECU physical models are used. The ASAM MCD-2MC standard describes by so-called record layouts how data are stored inside the ECU and which computation methods are needed to transform the ECU internal data representation into the physical one and vice versa.
The ASAM MCD-2MC standard also allows to describe and configure the ECU interfaces or vendor specific extensions by a meta description language (AML). For ASAM standardized ECU interfaces, such as CCP and XCP the content of these AML parts are also standardized. But there are also a lot of vendor specific instantiation in the market which use the same mechanism.
Measurement and calibration tools are normally only used during development phase of ECUs. They allow a direct, address-oriented write- and read-access but also a synchronous, continuous measurement access to ECU internal variables.
MC tools also offer features for flashing of new software versions comprising of new code and/or new parameter sets using the relevant ASAM MCD-1 interfaces. For calibration and flashing purposes the ASAM MCD-2MC standards describes the memory segment configuration to rebuild code and data externally.
Address-oriented information of ASAM MCD-2MC files may become obsolete with a new ECU software version running a new compiler/linker run because variables may be relocated in the memory. Therefore code generator tools but also other utilities very often generate or update ASAM MCD-2MC files.
The ASAM MCD-2MC format is widely spread in the community of ECU development worldwide.
It is used by many code generators, calibration and diagnostics tools, rapid control prototyping tools, data loggers, measurement systems, automation systems, etc.

The ASAM MCD-2MC reuses the same notation format of the former ASAP2 standards to ensure a downward compatibility. This is necessary as a wide set of existing ASAP2 tools is in the market and a switch to a different language format (e.g. XML) leads to high efforts to modify these tools.

1.2 ASAM MCD-2MC WITHIN THE ASAM AE OVERALL CONTEXT

The MCD standards of ASAM Automotive Electronics are structured in a 3 layer system as follows:

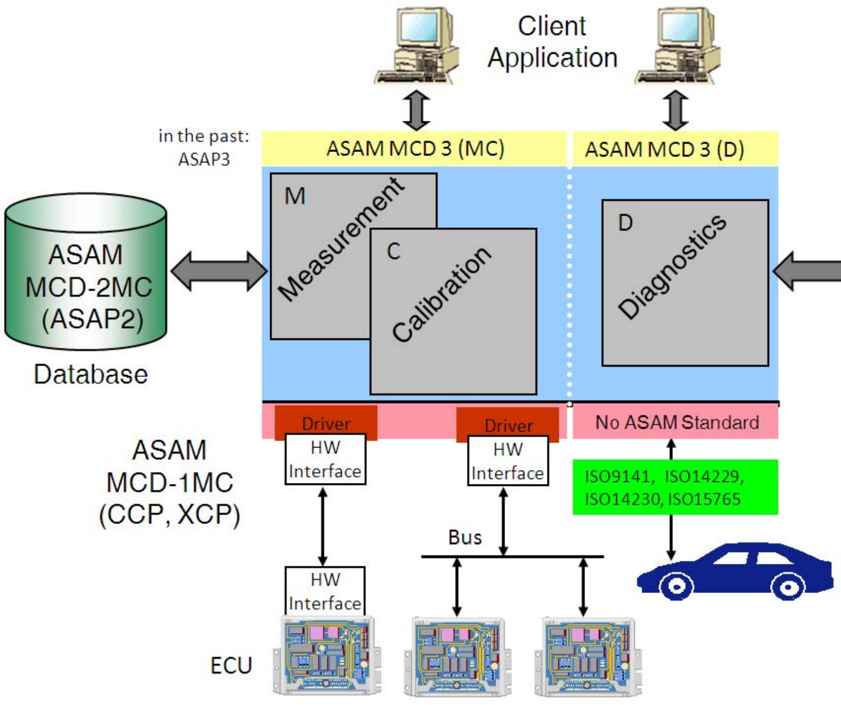


Figure 1: Structure of the MCD standards of ASAM AE

1.2.1 ASAM MCD-1

This summarizing term today denotes a set of standards to define the data interface primarily of ECUs. As shown in figure 1 the ASAM defined interfaces are used only for measurement and calibration tasks.
Diagnostic protocols and interfaces are not standardized by ASAM but by ISO (ISO 9141 [ISO 9141], 14229 [ISO 14229], 14230 [ISO 14230], 15765 [ISO 15765] or the PDU-API in ISO/DIS 22900-2 [ISO 22900-2]).
The most import standard in this sector, implemented in many ECUs worldwide, is the so-called CAN Calibration Protototcol. All market relevant calibration tools support this protocol as a base feature. But also specific measurement equipment, such as data loggers are able to connect to ECUs directly via the CAN bus.
Based on several years of experience with this CAN bus limited protocol ASAM developed the XCP standard, which in contrast to CCP is defined independently from the specific transport layer. Today several layers are available, such as CAN, USB, Ethernet, Flexray and SxI.

Very important for the success of XCP was the functional extension. As CCP was restricted to measurement and calibration, XCP also added a standardized option to write values back to the ECU in a synchronous manner (stimulation).
CCP as well as XCP allow to flash ECUs during development phase. This flashing access will normally be eliminated when the ECU is given to production and therefore is not available during service phase. Here the corresponding diagnostics services are applied.

1.2.2 ASAM MCD-2

The different ASAM MCD-2 formats build the data basis of an MCD system. ASAM MCD-2MC describes the necessary data for an MC oriented access, ASAM MCD-2D ODX [ASAM MCD-2D ODX] for a service-oriented diagnostics access and ASAM MCD-2FIBEX [ASAM MCD-2FIBEX] for an access via the different bus systems.

ASAM MCD-2D ODX in its versions since 2.0 is a unique, open XML exchange format for diagnostics data. The seamless data exchange between different partners along the process chain (suppliers, OEMs or service partners) is a very important process improvement.
Diagnostic tools like service testers or more development oriented tools can be parameterized via this format. The ODX standard defines an object-oriented data model, which is described in UML (Unified Modeling language). Inheritance and associations help to avoid data redundancies.
In contrast to the ASAM MCD-2MC standard, ODX data describe the parameters and access information for a diagnostic service oriented ECU access.
The ODX standard is also available as ISO 22901-1 [ISO 22901-1].

ASAM MCD-2FIBEX is also an XML format. It is capable to describe the entire vehicle communication network. All relevant bus technologies, such as CAN, MOST, LIN or Flexray are supported. Information exchange between tools of different suppliers is a standard FIBEX use case. Partial information can be completed step by step along the development progress.

1.2.3 ASAM MCD-3

ASAM MCD-3 specifies an object-oriented programming resp. remote-control interface to an MCD-server system.
This standard comprises of a base standard which is coded as a technology independent UML model with corresponding interface implementations for (D)COM, Java and C++.
The functional columns M (measurement), C (calibration) and D(diagnostics) can be applied independently of each other, but also in combined manner as M, MD, MC or entire MCD systems. Common parts, such as project or hardware setup, are defined as single source.
There is a clear market trend that more and more combined and integrated MCD systems will become available.
The diagnostics part of ASAM MCD-3 [ASAM MCD-3] is available as 22900-3 [ISO 22900-3].

It is important to mention the well-established remote-control interface ASAP-3MC [ASAP-3MC]. This standard was developed in the early nineties for RS232 and TCP/IP based communication between test bed automation systems and calibration tools. It is still in usage in many applications until today.

1.2.4 More ASAM AE Standards with an ASAM MCD-2MC relation

Beside the MCD standards additional standards either have been developed from scratch or have been modified and adapted based on pre-developments on MSR side.
The ASAM AE CDF [ASAM AE CDF] standard specifies an XML format to store calibration data, their level of maturity and other development process related data.
The ASAM AE MDF [ASAM AE MDF] standard, which is still under development, will allow to store measurement data in a very efficient binary format.

1.3 Abbreviations and terms

1.3.1 Abbreviations

The following abbreviations are used within the document:

A2L	ASAM MCD-2MC language
AE	Automotive Electronics
AML	ASAM MCD-2MC metalanguage
ASCII	American Standard for Character Information Interchange
CAN	Control Area Network
CCP	CAN Calibration Protocol
ECU	Electronic Control Unit
HW interface	Hardware interface, interface converter
IF	Interface
MCD	Measuring, Calibration and Diagnostics
ODX	Open Diagnostic data eXchange
ROM	Read-Only Memory
XCP	Universal Measurement and Calibration Protocol

1.3.2 Terms

1.3.2.1
ASAM MCD-2MC
Standardized interface for the description data.

1.3.2.2
ASAM MCD-2MC metalanguage
Formal description language for the description of non-standardized, interface-specific ASAM MCD-2MC description data.

1.3.2.3
Characteristic block
List of characteristics of the same data type (equal conversion method), which are stored sequentially in the data area of the control unit program (array) and which are considered as representing an adjustable object.

1.3.2.4
Deposit of axis points
This concept describes how the axis point values of a characteristic curve or characteristic map are deposited in memory:

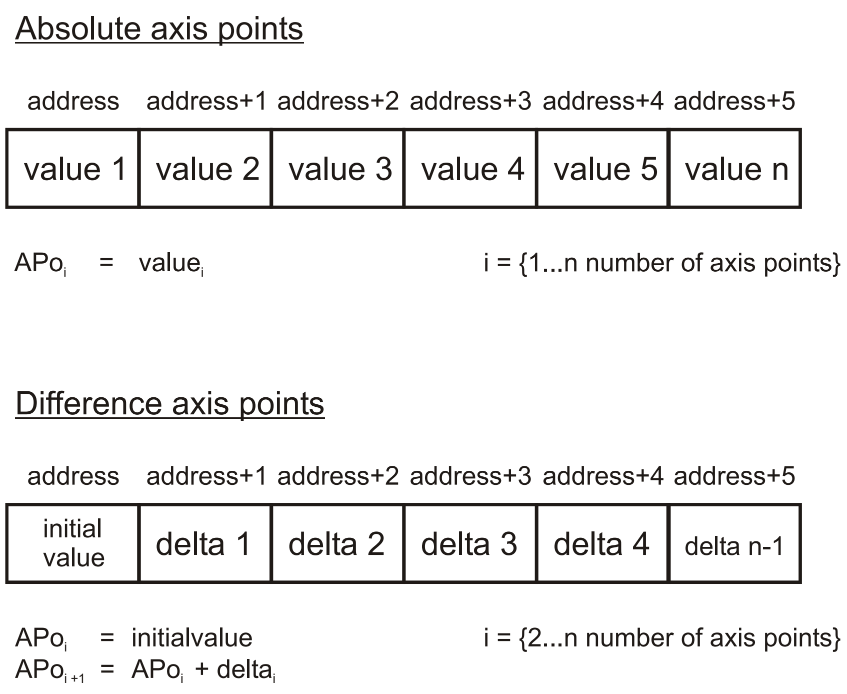


Figure 1 data deposition

1.3.2.5
Description data
For the calibration of a control unit program it must be possible to display and edit adjustable objects. In addition, it must be possible to display, collect and store measurements. This requires a description of the control unit program, which must contain all information needed to read and write adjustable objects in the emulation memory and to collect measurements. Moreover, information is needed which describes the display format of the adjustable and measurement objects.

1.3.2.6
Display table
Method for the output of control unit internal measurements:
1) The measurement and calibration system manipulates an address table in the data area of the control unit program.
2) The control unit program reads these tables in a predefined time pattern and outputs the corresponding data on defined addresses in the dual-ported RAM.

1.3.2.7
EPROM identifier
String in the data area of the control unit program for the description of the control unit program.

1.3.2.8
Fixed characteristic curve, fixed characteristic map

Characteristic curve or characteristic map in which the axis point values are contained as absolute or difference values in the data record but are calculated as follows (equidistant axis points):

$$Apo_i = offset + (i - 1)*2^{shift} \qquad i = \{ 1...numberofaxispoints\}$$

Both parameters <offset> and <shift> are contained either in the description file or in the data record of the control unit program.

1.3.2.9
Function orientation

For the structuring of projects involving a very large number of adjustable objects and measurement objects, functions can be defined in ASAM MCD-2MC. These functions shall be used in the measurement and calibration system to allow the selection lists for the selection of the adjustable objects and measuring channels to be represented in a structured manner on the basis of functional viewpoints.

1.3.2.10
Group characteristic curve, group characteristic map

In a number of control unit programs, "group characteristic curve" or "group characteristic map" denotes those characteristic curves or characteristic maps that have axis point distributions in common with other characteristic curves or characteristic map. Such an axis point distribution is allocated not to a single characteristic curve or characteristic map but to several characteristic curves and characteristic maps. If such an axis point distribution is changed, the behavior of all allocated characteristic curve or characteristic map changes accordingly.

1.3.2.11
Verbal conversion table

Conversion table for the visualization of bit patterns. This conversion method is used for special measurements. As a rule, parts of the measurements are masked out via bit masks. Each bit sample of the quantity thus obtained is allocated a string in the verbal conversion table, which describes the state of this quantity.

1.4 Compatibility

The current version ASAM MCD-2MC V1.6 defined in this document is in general downward compatible to the former version ASAM MCD-2MC V1.51. This means that the keywords and keyword combinations used to describe ECU software are in the same way supported as in the former version. The version 1.6 only adds new keywords to allow to describe more ECU software constructions than before.
Because of the maintenance some details are no longer downward compatible. This chapter lists the critical ones.

1.4.1 Incompatibility of Keyword FORMULA

The former definition of the keyword was not compatible to ANSI-C notation. The ANSI-C compatibility is important to convert formula descriptions easily between different systems engineering system (compiler) and software description systems.
In detail these are the logical operators:
Some operators have in ASAM MCD-2MC V1.51 a different meaning than in ASAM MCD-2MC V1.6 (ANSI-C).

Table 1 formula operator compatibility

Operator	ASAM MCD-2MC V1.51	ASAM MCD-2MC V1.6 (ANSI-C)
&	logical AND	bitwise AND
|	logical OR	bitwise OR
XOR	exclusive OR	not supported
~	logical NOT	bitwise NOT
^	power	bitwise exclusive OR
&&	not supported	logical AND
||	not supported	logical OR
!	not supported	logical NOT
ln(x)	supported, but not specified	not supported
log(x)	supported, but not specified	natural logarithm
log10(x)	not supported	decimal logarithm

Some operators have in ASAM MCD-2MC V1.51 a different notation than in ASAM MCD-2MC V1.6 (ANSI-C).

Table 2 formula operator notiation

ASAM MCD-2MC V1.51	ASAM MCD-2MC V1.6 (ANSI-C)
arcsin(x)	asin(x)
arcos(x)	acos(x)
arctan(x)	atan(x)

Note: If the keyword ASAP2_VERSION is missing (former ASAM MCD-2MC version) or states a version smaller than V1.6 tools shall use the formula interpretation of ASAM MCD-2MC V1.51.

1.4.2 Restriction for Brackets

ASAM MCD-2MC V1.6 requests always brackets of the form '/begin' '/end'. Curly brackets '{' '}' are no longer supported.

1.4.3 Win32 APIs for Seed&Key and Checksum Calculation

The definition of Win32 APIs for Seed&Key and checksum calculation which was formerly part of the ASAM MCD-2MC V1.51 specification is no longer part of this specifiation. These definitions are now found at [ASAM AE COMMON SEED&KEY].

1.5 Encoding of the A2L file

ASAM MCD-2MC files are used in different language areas. To support the different character sets used in the different language areas it is necessary to add to the ASAM MCD-2MC file the information about the used character set.
The ASCII and ISO-8859-x character sets, defined for several language areas are not sufficient. The ASAM MCD-2MC files needs to be exchangeable world wide.

1.5.1 Unicode Transformation Format

World wide exchange is supported by "Unicode Transformation Format" (UTF). Currently there are 3 relevant versions of UTF available: UTF -8, UTF-16, UTF-32.
UTF-8 is compact and supports nearly every possible character world wide. Therefore UTF-8 is the preferred encoding for ASAM MCD-2MC files.

Note: Tools shall support at least UTF-8.

1.5.2 Byte-Order Mark

The encoding that is used for the ASAM MCD-2MC file is defined in a Byte-Order Mark (BOM). The BOM is a byte sequence at the beginning of the ASAM MCD-2MC file.

Currently defined BOM sequences:

Table 3 byte-order mark coding

Bytes	Encoding Form
00 00 FE FF	UTF-32, big-endian
FF FE 00 00	UTF-32, little-endian
FE FF	UTF-16, big-endian
FF FE	UTF-16, little-endian
EF BB BF	UTF-8

If no encoding can be detected, ISO-8859-1 (Latin-1) encoding is used.

Note: For data type "ident" the restrictions listed in chapter 3.2 "Predefined data types" are valid.
Note: For user defined tags and enum values in AML the restrictions defined in chapter 5.2 "Format of the ASAM MCD-2MC Meta Language" are valid.

2 DIVISION OF THE DESCRIPTION DATA

The definition of the ASAM MCD-2MC interface and hence the specification of the ASAM MCD-2MC data base is aimed at defining a database independently of a computer or an operating system in such a way that a transparent and manufacturer-independent standard is established. As exchange format for such ECU descriptions *.a2l files are used.
From the calibration point of view the database in accordance with the ASAM MCD-2MC interface contains the complete description of all control unit relevant data in a project. A project consists of project specific header, which is typically created by the project manager, data and one or more control unit specific descriptions. These control unit descriptions (= description of an ECU) include all conversion formulas and explanations about the applicable (adjustable) and measurable (non-adjustable) quantities and present a format description of the interface specific parameters. The measurement and calibration system needs only to evaluate the quantities (and their conversion etc.), but not the interface specific parameters. The latter are only passed on to the structures of the driver. To make sure that these structures are correctly filled the MCD must know the parameter type. The type is communicated with the ASAM MCD-2MC metalanguage. As exchange format for such ECU IF Data descriptions *.aml files are used.
A project may include the control unit descriptions of various control units from different suppliers. The descriptions differ in terms of content, but use a common information storage methodology to allow for a global management of the project components. An INCLUDE mechanism allows to summarize the various control unit descriptions of various projects (Single-Source-Concept).
The ASAM MCD-2MC database thus consists of a number of different subcomponents structured in accordance with the following diagram. The MODULE keyword denotes an independent ECU or device.

```
PROJECT
{
  HEADER{...}             /* Project description */

  MODULE Device
  {
    MOD_PAR{...}          /* Control unit management data */
    MOD_COMMON {...}      /* Module-wide (ECU specific) definitions */

    CHARACTERISTIC{...} /* Adjustable objects */
    CHARACTERISTIC{...}
    ...
    AXIS_PTS{...}         /* Axis points objects */
    AXIS_PTS{...}
    ...
    MEASUREMENT{...}      /* Measurement objects */
    MEASUREMENT{...}
    ...
    COMPU_METHOD{...}     /* Conversion method */
    COMPU_METHOD{...}
    ...
    COMPU_TAB{...}        /* Conversion tables */
    COMPU_TAB{...}
```

```
            FUNCTION{...}       /* Function allocations */
            FUNCTION{...}
            ...
            GROUP{...}          /* Groups */
            GROUP{...}
            ...
            RECORD_LAYOUT{...}  /* Record layouts of adjustable objects */
            RECORD_LAYOUT{...}
        }
    }                           /* END OF PROJECT */
```

The following rules apply for a valid a2l file:
The file must contain exactly one PROJECT. The PROJECT must contain at least one MODULE.

The keywords defined in the ASAM MCD-2MC database are described in the following chapter.

3 FORMAT OF THE DESCRIPTION FILE

3.1 HIERARCHIC DIVISION OF THE KEYWORDS

Table 4 Hierarchic division of the keywords

Keyword	Multiple	Meaning
ASAP2_VERSION		ASAM MCD-2MC version identification
A2ML_VERSION		Version number of ASAM MCD-2MC Meta Language
PROJECT		Project description
HEADER		Project header description
PROJECT_NO		Project number
VERSION		Project version number
MODULE	X	Description of the ECU
A2ML	X	ASAM MCD-2MC Meta-Language (interface-specific description data)
AXIS_PTS	X	Axis points distribution
ANNOTATION	X	Set of notes
ANNOTATION_LABEL		Title of annotation
ANNOTATION_ORIGIN		Creator of annotation
ANNOTATION_TEXT		Text of annotation
BYTE_ORDER		Byte order of axis points
CALIBRATION_ACCESS		Access for calibration
DEPOSIT		Absolute or difference axis points
DISPLAY_IDENTIFIER		Optional display name
ECU_ADDRESS_EXTENSION		Address extension of the ECU address
EXTENDED_LIMITS		extended range of values
FORMAT		Display format of axis points
FUNCTION_LIST		Function orientation
GUARD_RAILS		Indicates the use of guardrails
IF_DATA	X	Interface-specific description data
MONOTONY		Monotony with respect to this axis
PHYS_UNIT		Physical unit of the axis points
READ_ONLY		'Read Only' attribute
REF_MEMORY_SEGMENT		reference to memory segment
STEP_SIZE		delta value
SYMBOL_LINK		reference to symbol of linker map file
CHARACTERISTIC	X	Adjustable objects
ANNOTATION	X	Description

Keyword	Multiple	Meaning
ANNOTATION_LABEL		Title of annotation
ANNOTATION_ORIGIN		Creator of annotation
ANNOTATION_TEXT		Text of annotation
AXIS_DESCR	X	Axis description
ANNOTATION	X	Set of notes
ANNOTATION_LABEL		Title of annotation
ANNOTATION_ORIGIN		Creator of annotation
ANNOTATION_TEXT		Text of annotation
AXIS_PTS_REF		Reference to axis point distribution
BYTE_ORDER		Byte order of axis points
CURVE_AXIS_REF		Used to normalize or scale an axis
DEPOSIT		Absolute or difference axis points
EXTENDED_LIMITS		Extended limits, e.g. hard limits
FIX_AXIS_PAR		Fixed axis parameters
FIX_AXIS_PAR_DIST		Fixed axis parameters (variant)
FIX_AXIS_PAR_LIST		Fixed axis values
FORMAT		Display format of axis points
MAX_GRAD		Maximum gradient with respect to this axis
MONOTONY		Monotony with respect to this axis
PHYS_UNIT		Physical unit of the axis points
READ_ONLY		'Read Only' attribute
STEP_SIZE		delta value
BIT_MASK		Bit mask
BYTE_ORDER		Byte order
CALIBRATION_ACCESS		Access for calibration
COMPARISON_QUANTITY		Comparison quantity
DEPENDENT_CHARACTERISTIC		References to characteristics
DISCRETE		Attribute for discrete object values
DISPLAY_IDENTIFIER		Optional display name
ECU_ADDRESS_EXTENSION		Address extension of the ECU address
EXTENDED_LIMITS		Extended limits, e.g. hard limits
FORMAT		Display format of values
FUNCTION_LIST		Function orientation
GUARD_RAILS		Indicates the use of guardrails
IF_DATA	X	Interface-specific description data
MAP_LIST		For cuboids: comprising maps
MATRIX_DIM		Dimensions of multidimensional arrays
MAX_REFRESH		Maximum refresh rate
NUMBER		Number of ASCII characters or fixed values

Keyword	Multiple	Meaning
PHYS_UNIT		Physical unit of the characteristic values
READ_ONLY		'Read Only' attribute
REF_MEMORY_SEGMENT		Reference to memory segment
STEP_SIZE		Delta value
SYMBOL_LINK		reference to symbol of linker map file
VIRTUAL_CHARACTERISTIC		Mark for being virtual
COMPU_METHOD	X	Conversion method
COEFFS		Coefficients for fractional rational function
COEFFS_LINEAR		Coefficients for linear function
COMPU_TAB_REF		Reference to conversion table
FORMULA		Conversion formula
FORMULA_INV		Inverse conversion formula
REF_UNIT		Reference to a measurement unit
STATUS_STRING_REF		Reference to an additional conversion table with status strings
COMPU_TAB	X	Conversion table
DEFAULT_VALUE		default output string
DEFAULT_VALUE_NUMERIC		default value
COMPU_VTAB	X	Verbal conversion table
DEFAULT_VALUE		Default output string
COMPU_VTAB_RANGE	X	Description of range based verbal conversion tables
DEFAULT_VALUE		Default output string
FRAME		Frame
FRAME_MEASUREMENT		Frame measurement objects
IF_DATA	X	Interface-specific description data
FUNCTION	X	Function description
ANNOTATION	X	Set of notes
ANNOTATION_LABEL		Title of annotation
ANNOTATION_ORIGIN		Creator of annotation
ANNOTATION_TEXT		Text of annotation
DEF_CHARACTERISTIC		Defined adjustable objects
FUNCTION_VERSION		Version of the function
IF_DATA	X	Interface-specific description data
IN_MEASUREMENT		Input quantity
LOC_MEASUREMENT		Local quantity
OUT_MEASUREMENT		Output quantity
REF_CHARACTERISTIC		Referenced adjustable objects
SUB_FUNCTION		Sub function of respective function
GROUP	X	Declaration of groups
ANNOTATION	X	Set of notes

Keyword	Multiple	Meaning
ANNOTATION_LABEL		Title of annotation
ANNOTATION_ORIGIN		Creator of annotation
ANNOTATION_TEXT		Text of annotation
FUNCTION_LIST		Function list
IF_DATA	X	Interface-specific description data
REF_CHARACTERISTIC		Reference to characteristic objects
REF_MEASUREMENT		Reference to measurement objects
ROOT		Flag for root node
SUB_GROUP		Sub group
IF_DATA	X	Interface-specific description data
MEASUREMENT	X	Measurement object
ANNOTATION	X	Set of notes
ANNOTATION_LABEL		Title of annotation
ANNOTATION_ORIGIN		Creator of annotation
ANNOTATION_TEXT		Text of annotation
ARRAY_SIZE		Array size of measurement objects
BIT_MASK		Bit mask to decode single-bit values
BIT_OPERATION		Bit operation
LEFT_SHIFT		Number of bit positions to shift left
RIGHT_SHIFT		Number of bit positions to shift right
SIGN_EXTEND		sign extension for measurement data
BYTE_ORDER		Byte order of measurement object
DISCRETE		Attribute for discrete object values
DISPLAY_IDENTIFIER		Optional display name
ECU_ADDRESS		Address
ECU_ADDRESS_EXTENSION		Address extension of the ECU address
ERROR_MASK		Mask error bits
FORMAT		Display format of measurement object
FUNCTION_LIST		Function orientation
IF_DATA	X	Interface-specific description data
LAYOUT		Layout of multidimensional arrays
MATRIX_DIM		Dimensions of multidimensional arrays
MAX_REFRESH		Refresh rate in the control unit
PHYS_UNIT		Physical unit of the measurement values
READ_WRITE		'Writeable'
REF_MEMORY_SEGMENT		reference to memory segment
SYMBOL_LINK		reference to symbol of linker map file
VIRTUAL		Virtual measurement
MOD_COMMON		Module-wide (ECU specific) valid definitions
ALIGNMENT_BYTE		Alignment border for byte values

Keyword	Multiple	Meaning
ALIGNMENT_FLOAT32_IEEE		Alignment border for float32 values
ALIGNMENT_FLOAT64_IEEE		Alignment border for float64 values
ALIGNMENT_INT64		Alignment border for int64 values
ALIGNMENT_LONG		Alignment border for long values
ALIGNMENT_WORD		Alignment border for word values
BYTE_ORDER		Byte order
DATA_SIZE		Data size in bits
DEPOSIT		Standard deposit mode for axis
S_REC_LAYOUT		Reference to the standard record layout
MOD_PAR		Control unit management data
ADDR_EPK	X	Address of EPROM identifier
CALIBRATION_METHOD	X	Access method
CALIBRATION_HANDLE	X	Handle for calibration method
CALIBRATION_HANDLE_TEXT		Additional Text for caibration method
CPU_TYPE		CPU
CUSTOMER		Firm or customer
CUSTOMER_NO		Customer number
ECU		Control unit
ECU_CALIBRATION_OFFSET		Address offset
EPK		EPROM identifier
MEMORY_LAYOUT	X	Memory layout
IF_DATA	X	Interface-specific description data
MEMORY_SEGMENT	X	Memory segment
IF_DATA	X	Interface-specific description data
NO_OF_INTERFACES		Number of interfaces
PHONE_NO		Phone number of calibration engineer responsible
SUPPLIER		Manufacturer or supplier
SYSTEM_CONSTANT	X	System-defined constants
USER		User
VERSION		Module-specific version identifier
RECORD_LAYOUT	X	Description of the record layout
ALIGNMENT_BYTE		Alignment border for byte values
ALIGNMENT_FLOAT32_IEEE		Alignment border for float32 values
ALIGNMENT_FLOAT64_IEEE		Alignment border for float64 values
ALIGNMENT_INT64		Alignment border for int64 values
ALIGNMENT_LONG		Alignment border for long values
ALIGNMENT_WORD		Alignment border for word values
AXIS_PTS_X / _Y / _Z / _4 / _5		Axis points
AXIS_RESCALE_X / _Y / _Z / _4 / _5		Rescaling axis points

Keyword	Multiple	Meaning
DIST_OP_X / _Y / _Z / _4 / _5		Parameter 'distance' for fixed characteristics
FIX_NO_AXIS_PTS_X / _Y / _Z / _4 / _5		Fixed number of axis points
FNC_VALUES		Table values
IDENTIFICATION		Identification
NO_AXIS_PTS_X / _Y / _Z / _4 / _5		Number of X axis points
STATIC_RECORD_LAYOUT		Flag for non-compact data
NO_RESCALE_X / _Y / _Z / _4 / _5		Number of rescale pairs for axis
OFFSET_X / _Y / _Z / _4 / _5		Parameter 'offset' for fixed characteristics
RESERVED	X	Parameter is skipped (not interpreted)
RIP_ADDR_W		Table value: Address 'result of interpolation'
RIP_ADDR_X / _Y / _Z / _4 / _5		Address 'result of interpolation'
SHIFT_OP_X / _Y / _Z / _4 / _5		Parameter 'shift' for fixed characteristics
SRC_ADDR_X / _Y / _Z / _4 / _5		Address of input quantity
UNIT	X	Measurement unit
REF_UNIT		Reference to another unit
SI_EXPONENTS		Exponential of base dimensions
UNIT_CONVERSION		Specifies relationship between two units
USER_RIGHTS	X	Groups with constitute access rights
READ_ONLY		Read only
REF_GROUP	X	List of referenced groups
VARIANT_CODING		Variant coding
VAR_CHARACTERISTIC	X	Definition of variant coded adjustable objects
VAR_ADDRESS		Adjustable objects address list (start address of variants)
VAR_CRITERION	X	Definition of variant criterion
VAR_MEASUREMENT		Measurement object which indicates criterion value
VAR_SELECTION_ CHARACTERISTIC		Characteristic object which modifies criterion value
VAR_FORBIDDEN_COMB	X	Forbidden combinations of different variants
VAR_NAMING		Naming of variant coded adjustable objects
VAR_SEPERATOR		Separator of adjustable objects names

3.2 PREDEFINED DATA TYPES

Table 5 Predefined data types

Pre-defined data type		**description**
ident	typedef char [MAX_IDENT + 1] ident	String with *MAX_IDENT (at present = 1024)* alphanumerical characters including points and brackets, interpreted as hierarchical concatenation of partial strings separated by points. Every partial string may not exceed *MAX_PARTIAL_IDENT (at present = 128)* characters, including the length of an optional array index (numeric or as a symbolic string) in brackets at the end of the partial string. One string without a point in between is also possible, in this case MAX_IDENT = MAX_PARTIAL_IDENT. The number of partial strings within ident is not limited. The character chain must correspond with the identifier laws defined in programming language C. Identifiers can represent instances of array elements or instances of elements of complex C types or nested combinations of these. An instance of the element of a struct type would be represented by the concatenation of the instance name, a point and the element name. An instance of an array element would be represented by an instance name followed by a pair of brackets which contain either a numeric value or a symbolic string which is defined as an enumerator of an ENUM definition of the C program. Identifiers are random names which may contain characters A through Z, a through z, underscore (_), numerals 0 through 9, points ('.') and brackets ('[','] ') . However, the following limitations apply: the first character must be a letter or an underscore, brackets must occur in pairs at the end of a partial string and must contain a number or an alphabetic string (description of the index of an array element). Note: Identifiers consisting of partial identifiers separated by points (concatenation of instance name and element name) may be presented by the MCD system in a hierarchical manner (show instance name first, then allow access to an element of the instance). This allows existing MCD systems to restrict the display length of the identifier to MAX_PARTIAL_IDENT. Note: Identifiers generally must not match to the following defined ASAM MCD-2MC keywords and enum values. All keyword and enum values are listed in the Index of Keywords and Enum Values. Note: A lower case "x" and a upper case "X" can be used as unique identifiers. In other words, variables are case sensitive so that x and X are different identifiers.

Pre-defined data type		description
string	typedef char [MAX_STRING + 1] string	ANSI C compliant 'C type' string with maximum MAX_STRING (at present = 255) characters**.** Begin and end of the string are indicated by a double inverted comma. The following escape sequences are allowed: \´ inverted comma \" quotation mark \\ backslash \n new line \r carriage return \t horizontal tab Additionally, for compatibility with ASAP2 V1.2 and prior, the following is allowed: "" quotation mark Examples: "hello \"world\" how are you ?" "hello ""world"" how are you ?" MCD systems may ignore the carriage return sequence and/or apply wrapping or scrolling of strings when displayed.
float	8-byte floating point number (IEEE format)	The character for a decimal point is fixed as a dot ".". A comma "," is not allowed for use as a decimal point.
int	2-byte signed integer	The notation of hexadecimal values is fixed, e.g. 0xE0, 0xFF, etc.
uint	2-byte unsigned integer	
long	4-byte signed long	
ulong	4-byte unsigned integer	
datatype	typedef enum datatype { UBYTE, SBYTE, UWORD, SWORD, ULONG, SLONG, A_UINT64, A_INT64, FLOAT32_IEEE, FLOAT64_IEEE }	Enumeration for description of the basic data types in the ECU program (format of FLOAT32/64_IEEE: see Appendix C). Note: If ECU values of type integer 64 are converted in physical values depending on the computation formula a higher precision in the physical area is necessary. The currently used float 64 format supports less precision than int 64. Therefore the precision of the physical representation is reduced to the precision of float 64. This is relevant for ASAM MCD-3 standards [ASAM MCD-3] where data transfer is defined as physical. This is additionally relevant for all tools working on PCs / Operating systems that do not support higher precision than float 64. Here the representation is rounded for physical and maybe also for internal representation.

Pre-defined data type		**description**
datasize	typedef enum datasize { BYTE, WORD, LONG}	Enumeration for description of the word lengths in the ECU program
addrtype	typedef enum addrtype {PBYTE PWORD PLONG DIRECT }	Enumeration for description of the addressing of table values or axis point values: The relevant memory location has a 1 byte pointer to this table value or axis point value. The relevant memory location has a 2 byte pointer to this table value or axis point value. The relevant memory location has a 4 byte pointer to this table value or axis point value. The relevant memory location has the first table value or axis point value, all others follow with incrementing address.
byteorder	typedef enum byteorder { LITTLE_ENDIAN, BIG_ENDIAN, MSB_LAST, MSB_FIRST }	Enumeration for description of the byte order in the control unit program. Note: Use of LITTLE_ENDIAN and BIG_ENDIAN defined with keyword BYTE_ORDER leads to mistakes because it is in contradiction to general use of terms „little endian“ and „big endian“. The keywords LITTLE_ENDIAN and BIG_ENDIAN should no longer be used, they should be replaced by MSB_LAST and MSB_FIRST which are equivalent (definition of MSB_LAST and MSB_FIRST: see keyword BYTE_ORDER).
indexorder	typedef enum indexorder	Enumeration for description of the axis point sequence in the memory.
	{INDEX_INCR INDEX_DECR}	Increasing index with increasing address decreasing index with increasing address

3.3 MAPPING OF PREDEFINED DATA TYPES TO ASAM DATA TYPES

The following table shows, how the ASAM data types are mapped to predefined data types.

Table 6 Map ASAM data types to predefined data types

ASAM data type	predefined data type
A_INT16	int
A_UINT16	uint
A_INT32	long
A_UINT32	ulong
A_FLOAT64	float

3.4 COMMENTS

Single line and multi line comments may be added everywhere in an aml and a2l file.

Single line comments start with the character string "//" and end at the end of the same line.
Multi line comments start with the character string "/*" and end with the character string "*/". Nested multi line comments are not allowed.

Example for a single line comment:

```
// This is a single line comment
```

Example for a multi line comment

```
/*
This is a
multi line comment
*/
```

3.5 ALPHABETICAL LIST OF KEYWORDS

3.5.1 GENERAL

Some individual elements of the database are delimited by '/begin' and '/end' keywords. The delimiters are applied to those elements that contain an optional part, to prevent ambiguous expressions. The delimiters following defined with the ASAM MCD-2MC keywords are mandatory, i.e. the delimiters have to be used if defined and mustn't be used if not defined.

Since ASAM MCD-2MC version 1.6 the use of short delimiters '{' and '}' is not supported any longer.

3.5.2 A2ML

Prototype:

```
/begin A2ML                FormatSpecification
/end A2ML
```

Parameters:

FormatSpecification	AML code for description of interface specific description data.

Description:

This keyword identifies the format description of the interface specific description data.

Example:

See B.1 SUPP1_IF.AML

3.5.3 A2ML_VERSION

Prototype:

```
A2ML_VERSION                uint      VersionNo
                            uint      UpgradeNo
```

Parameters:

uint	VersionNo	Version number of AML part
uint	UpgradeNo	Upgrade number of AML part

Description:

The reason for this keyword is, to declare what kind of BLOBs should be generated from the AML parts. Since ASAP2 version 1.31 a specification for the storage layout of the BLOBs exist. The keyword is optional. When the keyword is omitted, or the version number is below 1.31 then the old BLOB format is used. When the A2ML version number is 1.31, then the new format must be generated.

The A2ML version can be expressed by two numerals:

- VersionNo
- UpgradeNo

where ‘VersionNo’ represents the main version number and ‘UpgradeNo’ the upgrade number (fractional part of version number). The upgrade number will be incremented if additional functionality is added to ASAM MCD-2MC Meta Language standard which has no effect on existing applications (compatible modifications). The version number will be incremented in case if incompatible modifications.

Example:

```
A2ML_VERSION                1
                            31     /* Version 1.31 */
```

3.5.4 ADDR_EPK

Prototype:

```
ADDR_EPK                    ulong     Address
```

Parameters:

ulong	Address	Address of the EPROM identifier

Description:

Address of the EPROM identifier

Example:

```
ADDR_EPK                    0x145678
```

3.5.5 ALIGNMENT_BYTE

Prototype:

```
ALIGNMENT_BYTE              uint      AlignmentBorder
```

Parameters:

uint	AlignmentBorder	describes the border at which the value is aligned to, i.e. its memory address must be dividable by the value AlignmentBorder.

Description:

In complex objects (maps and axis) the alignment of a value may not coincide with the bitwidth of a value. This keyword is used to define the alignment in the case of bytes.

Example:

```
 ALIGNMENT_BYTE            4      /* bytes have a 4-byte alignment */
```

3.5.6 ALIGNMENT_FLOAT32_IEEE

Prototype:

```
ALIGNMENT_FLOAT32_IEEE    uint     AlignmentBorder
```

Parameters:

uint	AlignmentBorder	describes the border at which the value is aligned to, i.e. its memory address must be dividable by the value AlignmentBorder.

Description:

In complex objects (maps and axis) the alignment of a value may not coincide with the bitwidth of a value. This keyword is used to define the alignment in the case of 32bit floats.

Example:

```
ALIGNMENT_FLOAT32_IEEE   4       /* 32bit floats have a 4-byte alignment */
```

3.5.7 ALIGNMENT_FLOAT64_IEEE

Prototype:

```
ALIGNMENT_FLOAT64_IEEE    uint     AlignmentBorder
```

Parameters:

uint	AlignmentBorder	describes the border at which the value is aligned to, i.e. its memory address must be dividable by the value AlignmentBorder.

Description:

In complex objects (maps and axis) the alignment of a value may not coincide with the bitwidth of a value. This keyword is used to define the alignment in the case of 64bit floats.

Example:

```
  ALIGNMENT_ FLOAT64_IEEE  4      /* 64bit floats have a 4-byte alignment */
```

3.5.8 ALIGNMENT_INT64

Prototype:

```
ALIGNMENT_INT64             uint      AlignmentBorder
```

Parameters:

uint	AlignmentBorder	describes the border at which the value is aligned to, i.e. its memory address must be dividable by the value AlignmentBorder.

Description:

In complex objects (maps and axis) the alignment of a value may not coincide with the bitwidth of a value. This keyword is used to define the alignment in the case of int64.

Example:

```
ALIGNMENT_INT64          4       /* int64 have a 4-byte alignment */
```

3.5.9 ALIGNMENT_LONG

Prototype:

```
ALIGNMENT_LONG            uint      AlignmentBorder
```

Parameters:

uint	AlignmentBorder	describes the border at which the value is aligned to, i.e. its memory address must be dividable by the value AlignmentBorder.

Description:

In complex objects (maps and axis) the alignment of a value may not coincide with the bitwidth of a value. This keyword is used to define the alignment in the case of longs.

Example:

```
ALIGNMENT_LONG            8      /* longs have a 8-byte alignment */
```

3.5.10 ALIGNMENT_WORD

Prototype:

```
ALIGNMENT_WORD             uint      AlignmentBorder
```

Parameters:

uint	AlignmentBorder	describes the border at which the value is aligned to, i.e. its memory address must be dividable by the value AlignmentBorder.

Description:

In complex objects (maps and axis) the alignment of a value may not coincide with the bitwidth of a value. This keyword is used to define the alignment in the case of words. The alignment is 2 if the parameter is missing.

Example:

```
ALIGNMENT_WORD          4       /* words have a 4-byte alignment */
```

3.5.11 ANNOTATION

Prototype:

```
/begin ANNOTATION
                         [-> ANNOTATION_LABEL]
                         [-> ANNOTATION_ORIGIN]
                         [-> ANNOTATION_TEXT]
/end ANNOTATION
```

Parameters:

none

Optional Parameters:

-> ANNOTATION_LABEL	label or title of the annotation
-> ANNOTATION_ORIGIN	creator or creating system of the annotation
-> ANNOTATION_TEXT	text of the annotation, voluminous description text

Description:

One ANNOTATION may represent a voluminous description. Its purpose is to be e.g. an application note which explains the function of an identifier for the calibration engineer.

Example:

```
/begin CHARACTERISTIC  annotation.example1
....
  /begin ANNOTATION
    ANNOTATION_LABEL     "Luftsprungabhängigkeit"
    ANNOTATION_ORIGIN    "Graf Zeppelin"
    /begin ANNOTATION_TEXT
      "Die luftklasseabhängigen Zeitkonstanten t_hinz\r\n"
      "& t_kunz können mit Hilfe von Luftsprüngen ermittelt werden.\r\n"
      "Die Taupunktendezeiten in großen Flughöhen sind stark "
      "schwankend."
    /end ANNOTATION_TEXT
  /end ANNOTATION

  /begin ANNOTATION
    ANNOTATION_LABEL     "Taupunktendezeiten"
    /begin ANNOTATION_TEXT
      "Flughöhe          Taupunktendezeit\r\n"
      " 13000ft          20 sec\r\n"
      " 25000ft          40 sec\r\n"
      " 35000ft          12 sec"
    /end ANNOTATION_TEXT
  /end ANNOTATION
....
/end CHARACTERISTIC
```

3.5.12 ANNOTATION_LABEL

Prototype:

```
ANNOTATION_LABEL          string    label
```

Parameters:

string	label	label or title of the annotation

Description:

Assign a title to an annotation. Useful as a definition can contain more than one annotation.
Recommendation : The ANNOTATION_LABEL shall describe the use-case of the ANNOTATION, e.g. „Calibration Note“.

Example:

```
 ANNOTATION_LABEL         "Calibration Note‟
```

3.5.13 ANNOTATION_ORIGIN

Prototype:

```
ANNOTATION_ORIGIN          string   origin
```

Parameters:

string origin	creator or creating system of the annotation

Description:

To identify who or which system has created an annotation.

Example:

```
ANNOTATION_ORIGIN        "from the calibration planning department“
```

3.5.14 ANNOTATION_TEXT

Prototype:

```
/begin ANNOTATION_TEXT
                              {string   annotation_text}*
/end ANNOTATION_TEXT
```

Parameters:

string	annotation_text

Description:

One ANNOTATION_TEXT may represent a multi-line ASCII description text (voluminous description). Its purpose is to be an application note which explains the function of an identifier for the calibration engineer.

Example:

```
/begin CHARACTERISTIC  annotation.example2 ...
....
  /begin ANNOTATION
    ANNOTATION_LABEL  "Calibration Note"
    /begin ANNOTATION_TEXT
      "The very nice ASAM MCD-2MC Specification."
      "Text.\r\n"
      "In case of a quotation mark "
      "use \" or "" to mark it."
    /end ANNOTATION_TEXT
  /end ANNOTATION
....
/end CHARACTERISTIC
```

3.5.15 ARRAY_SIZE

Prototype:

```
ARRAY_SIZE                 uint      Number
```

Parameters:

uint	Number	Number of measurement values included in respective measurement object (maximum value of ‘Number’: 32767).

Description:

This keyword marks a measurement object as an array of <Number> measurement values.

Note: The use of this keyword should be replaced by MATRIX_DIM.

Example:

```
/begin MEASUREMENT  N                  /* name */
                    "Engine speed"     /* long identifier */
                    UWORD              /* datatype */
                    R_SPEED_3          /* conversion */
                    2                  /* resolution */
                    2.5                /* accuracy */
                    120.0              /* lower limit */
                    8400.0             /* upper limit */
  ARRAY_SIZE        8                  /* array of 8 values */
  BIT_MASK          0x0FFF
  BYTE_ORDER        MSB_FIRST
  /begin FUNCTION_LIST
                    ID_ADJUSTM
                    FL_ADJUSTM
  /end FUNCTION_LIST
  /begin IF_DATA
                    ISO
                    SND
                    0x10
                    0x00
                    0x05
                    0x08
                    RCV
                    4
                    long
  /end IF_DATA
/end MEASUREMENT
```

3.5.16 ASAP2_VERSION

Prototype:

```
ASAP2_VERSION                uint      VersionNo
                             uint      UpgradeNo
```

Parameters:

uint	VersionNo	Version number of ASAM MCD-2MC standard
uint	UpgradeNo	Upgrade number of ASAM MCD-2MC standard

Description:

The ASAM MCD-2MC version can be expressed by two numerals:

- VersionNo
- UpgradeNo

where ‘VersionNo’ represents the main version number and ‘UpgradeNo’ the upgrade number (fractional part of version number). The upgrade number will be incremented if additional functionality is implemented to ASAM MCD-2MC standard which has no effect on existing applications (compatible modifications). The version number will be incremented in case if incompatible modifications.

The ASAP2_VERSION keyword is mandatory.

Example:

```
ASAP2_VERSION             1
                          60    /* Version 1.60 */
```

3.5.17 AXIS_DESCR

Prototype:

```
/begin AXIS_DESCR           enum     Attribute
                            ident    InputQuantity
                            ident    Conversion
                            uint     MaxAxisPoints
                            float    LowerLimit
                            float    UpperLimit
                            {-> ANNOTATION}*
                            [-> AXIS_PTS_REF]
                            [-> BYTE_ORDER]
                            [-> CURVE_AXIS_REF]
                            [-> DEPOSIT]
                            [-> EXTENDED_LIMITS]
                            [-> FIX_AXIS_PAR]
                            [-> FIX_AXIS_PAR_DIST]
                            [-> FIX_AXIS_PAR_LIST]
                            [-> FORMAT]
                            [-> MAX_GRAD]
                            [-> MONOTONY]
                            [-> PHYS_UNIT]
                            [-> READ_ONLY]
                            [-> STEP_SIZE]
/end AXIS_DESCR
```

Parameters:

enum	Attribute	Description of the axis points:	
		CURVE_AXIS	Curve axis. This axis type uses a separate CURVE CHARACTERISTIC to rescale the axis. The referenced CURVE is used to lookup an axis index, and the index value is used by the controller to determine the operating point in the CURVE or MAP. See Appendix D for more details.
		COM_AXIS	Group axis points or description of the axis points for deposit. For this variant of the axis points the axis point values are separated from the table values of the curve or map in the emulation memory and must be described by a special AXIS_PTS data record. The reference to this record occurs with the keyword 'AXIS_PTS_REF'.
		FIX_AXIS	This is a curve or a map with virtual axis points that are not deposited at EPROM. The axis points can be calculated from parameters defined with keywords FIX_AXIS_PAR, FIX_AXIS_PAR_DIST and FIX_AXIS_PAR_LIST. The axis points can‘t be modified.
		RES_AXIS	Rescale axis. For this variant of the axis points the axis point values are separated from the table values of the

		curve or map in the emulation memory and must be described by a special AXIS_PTS data record. The reference to this record occurs with the keyword 'AXIS_PTS_REF'.
		STD_AXIS Standard axis
ident	InputQuantity	Reference to the data record for description of the input quantity (see MEASUREMENT). If there is no input quantity assigned, parameter 'InputQuantity' should be set to "NO_INPUT_QUANTITY" (measurement and calibration systems must be capable to treat this case).
ident	Conversion	Reference to the relevant record of the description of the conversion method (see COMPU_METHOD). If there is no conversion method, as in the case of CURVE_AXIS, the parameter 'Conversion' should be set to "NO_COMPU_METHOD" (measurement and calibration systems must be able to handle this case).
uint	MaxAxisPoints	Maximum number of axis points Note: The measurement and calibration system can change the dimensions of a characteristic (increase or decrease the number of axis points). The number of axis points may not be increased at random as the address range reserved for each characteristic in the ECU program by the measurement and calibration system cannot be changed.
float	LowerLimit	Plausible range of axis point values, lower limit
float	UpperLimit	Plausible range of axis point values, upper limit Note: Depending on the type of conversion, the limit values are interpreted as physical or internal values. For conversions of type COMPU_VTAB and COMPU_VTAB_RANGE, the limit values are interpreted as internal values. For all other conversion types, the limit values are interpreted as physical values.

Optional parameters:

-> ANNOTATION	Set of notes (represented as multi-line ASCII description texts) which are related. Can serve e.g. as application note. When a COM_AXIS is referenced it is sufficient to place the ANNOTATION with its AXIS_PTS in order to avoid redundant information.
-> AXIS_PTS_REF	Reference to the AXIS_PTS record for description of the axis points distribution.
-> BYTE_ORDER	Where the standard value does not apply this parameter can be used to specify the byte order (Intel format, Motorola format) of the axis point value.
-> CURVE_AXIS_REF	When the axis type is CURVE_AXIS, this keyword must be used to specify the CURVE CHARACTERISTIC that is used to normalize or scale this axis.

-> DEPOSIT	The axis points of a characteristic can be deposited in two different ways: a) The individual axis point values are deposited as absolute values. b) The individual axis point are stored as differences. Each axis point value is determined from the adjacent axis point (predecessor). Where the standard value does not apply this parameter can be used to specify the axis point deposit.
-> EXTENDED_LIMITS	This keyword can be used to specify an extended range of values. In the measurement and calibration system, for example, when leaving the standard range of values (lower limit...upper limit) a warning could be generated (extended limits enabled only for "power user").
-> FIX_AXIS_PAR	For curves or maps, the axis points distribution is not stored in memory but it is computed on the basis of the offset (initial value) and a difference. For the record layouts used today, these parameters must be included in the description file. The specification occurs with keyword 'FIX_AXIS_PAR'.
-> FIX_AXIS_PAR_DIST	Similar to FIX_AXIS_PAR but with a different computing method
-> FIX_AXIS_PAR_LIST	The original values of the axis are directly contained in the file. The assigned COMPU_METHOD is applied to achieve the actual display values from the values with this keyword
-> FORMAT	With deviation from the display format specified with keyword COMPU_METHOD referenced by parameter <Conversion> a special display format can be specified to be used to display the axis points.
-> MAX_GRAD	This keyword can be used to specify a maximum permissible gradient for the adjustable object with respect to this axis (MaxGrad= max (abs(($W_{i,k}$-$W_{i-1,k}$)/(X_i-X_{i-1})))).
-> MONOTONY	This keyword can be used to specify a monotonous behavior for the adjustable object with respect to this axis.
-> PHYS_UNIT	With this keyword a physical unit can be specified for the axis points if no conversion rule is referenced (NO_COMPU_METHOD). Note: If a conversion rule is referenced the additional usage of PHYS_UNIT overrules the unit specified at the referenced conversion rule.
-> READ_ONLY	This keyword can be used to indicate that the axis points of adjustable object cannot be changed (but can be read only). Note: This optional keyword used at CHARACTERISTIC record indicates the adjustable object to be read only at all (table values and axis points).

-> STEP_SIZE	This keyword can be used to define a delta value which is added to or subtracted from the current value when using up/down keys while calibrating.

Description:

Axis description within an adjustable object

Note: With the 'input quantity' parameter a reference is made to a measurement object (MEASUREMENT). The MEASUREMENT keyword also specifies the 'conversion', 'lower limit' and 'upper limit' parameters.
It is expected that both conversions are equivalent, i.e. they must lead to the same result. The 'upper limit' and 'lower limit' parameters may be different.

Note: The keywords FIX_AXIS_PAR, FIX_AXIS_PAR_DIST, DEPOSIT and FIX_AXIS_PAR_LIST are mutually exclusive, i.e. at most one of these keywords is allowed to be used at the same AXIS_DESCR record.

Note: For the axis types COM_AXIS, RES_AXIS and CURVE_AXIS some attributes are defined twice: both at the AXIS_DESCR record and at the referenced AXIS_PTS resp. CHARACTERISTIC record. These redundant attributes are *InputQuantity*, *Conversion*, *MaxAxisPoints*, *LowerLimit*, *UpperLimit* and some optional parameters (e.g.: PHYS_UNIT). To support existing use cases where one common axis is used with different input quantities (e.g. multiple cylinders) it is recommended to ignore the redundant attributes defined at AXIS_PTS and use the values of the AXIS_DESCR record instead. Exeptions are *MaxAxisPoints* and MONOTONY which are used from AXIS_PTS.

Example:

```
/begin  AXIS_DESCR  STD_AXIS     /* Standard axis points */
                    N            /* Reference to input quantity */
                    CONV_N       /* Conversion */
                    14           /* Max.number of axis points*/
                    0.0          /* Lower limit */
                    5800.0       /* Upper limit*/
        MAX_GRAD    20.0         /* Axis: maximum gradient*/
/end AXIS_DESCR
```

3.5.18 AXIS_PTS

Prototype:

```
/begin AXIS_PTS           ident     Name
                          string    LongIdentifier
                          ulong     Address
                          ident     InputQuantity
                          ident     Deposit
                          float     MaxDiff
                          ident     Conversion
                          uint      MaxAxisPoints
                          float     LowerLimit
                          float     UpperLimit
                          {-> ANNOTATION}*
                          [-> BYTE_ORDER]
                          [-> CALIBRATION_ACCESS]
                          [-> DEPOSIT]
                          [-> DISPLAY_IDENTIFIER]
                          [-> ECU_ADDRESS_EXTENSION]
                          [-> EXTENDED_LIMITS]
                          [-> FORMAT]
                          [-> FUNCTION_LIST]
                          [-> GUARD_RAILS]
                          {-> IF_DATA}*
                          [-> MONOTONY]
                          [-> PHYS_UNIT]
                          [-> READ_ONLY]
                          [-> REF_MEMORY_SEGMENT]
                          [-> STEP_SIZE]
                          [-> SYMBOL_LINK]
/end AXIS_PTS
```

Parameters:

ident	Name	unique identifier in the ECU program Note: The name of the axis points object has to be unique within all measurement objects and adjustable objects of the ASAM MCD-2MC MODULE, i.e. there must not be another AXIS_PTS, CHARACTERISTIC or MEASUREMENT object with the same identifier in the MODULE.
string	LongIdentifier	comment, description
ulong	Address	address of the adjustable object in the emulation memory
ident	InputQuantity	reference to the data record for description of the input quantity (see MEASUREMENT). If there is no input quantity assigned, parameter 'InputQuantity' should be set to "NO_INPUT_QUANTITY" (measurement and calibration systems must be capable to treat this case).
ident	Deposit	reference to the relevant data record for description of the record layout (see RECORD_LAYOUT)
float	MaxDiff	maximum float with respect to the adjustment of a table value
ident	Conversion	Reference to the relevant record of the description of the conversion method (see COMPU_METHOD). If there is no conversion method, as in the case of CURVE_AXIS, the parameter ‘Conversion’ should be

		set to "NO_COMPU_METHOD" (measurement and calibration systems must be able to handle this case).
uint	MaxAxisPoints	maximum number of axis points
float	LowerLimit	plausible range of axis point values, lower limit
float	UpperLimit	plausible range of axis point values, upper limit

Note: Depending on the type of conversion, the limit values are interpreted as physical or internal values.
For conversions of type COMPU_VTAB and COMPU_VTAB_RANGE, the limit values are interpreted as internal values. For all other conversion types, the limit values are interpreted as physical values.

Note: The parameters of AXIS_PTS are dominate. The values defined at AXIS_DESCR have to be ignored in case of COM_AXIS.

Optional parameters:

-> ANNOTATION	Set of notes (represented as multi-line ASCII description texts) which are related. Can serve e.g. as application note.
-> BYTE_ORDER	Where the standard value does not apply, this parameter can be used to specify the byte order (Intel format, Motorola format) of the axis points.
-> CALIBRATION_ACCESS	This keyword specifies the access of the axis points for calibration. It replaces the READ_ONLY attribute.
-> DEPOSIT	The axis points of a characteristic can be deposited in one of the following two modes: a) the individual axis points are deposited as absolute values; b) the individual axis points are deposited as differences. Each axis point is determined from the adjacent point (predecessor). Where the standard value does not apply, this parameter can be used to specify the deposit of axis points.
-> DISPLAY_IDENTIFIER	Can be used as a display name (alternative to the 'name' attribute).
-> ECU_ADDRESS_EXTENSION	This keyword is an additional address information. For instance it can be used, to distinguish different address spaces of an ECU (multi-micro controller devices).
-> EXTENDED_LIMITS	This keyword can be used to specify an extended range of values. In the measurement and calibration system, for example, when leaving the standard range of values (lower limit...upper limit) a warning could be generated (extended limits enabled only for "power user").
-> FORMAT	With deviation from the display format specified with keyword COMPU_TAB referenced by parameter <Conversion> a special display format can be specified to be used to display the axis points.

-> FUNCTION_LIST	This keyword can be used to specify a list of 'functions' to which the axis points distribution is allocated (function orientation).
-> GUARD_RAILS	This keyword is used to indicate that an AXIS_PTS uses guard rails. The Measurement and Calibration System does not allow the user to edit the outermost axis breakpoints (see GUARD_RAILS).
-> IF_DATA	Data record to describe interface specific data of the axis points. The parameters associated with this keyword have to be described in the ASAM MCD-2MC metalanguage.
-> MONOTONY	This keyword can be used to specify a monotonous behavior for the adjustable object with respect to this axis.
-> PHYS_UNIT	With this keyword a physical unit can be specified for the axis points if no conversion rule is referenced (NO_COMPU_METHOD). Note: If a conversion rule is referenced the additional usage of PHYS_UNIT overrules the unit specified at the referenced conversion rule.
-> READ_ONLY	This keyword can be used to indicate that the axis points of axis points distribution cannot be changed (but can be read only). Note: This optional keyword used at CHARACTERISTIC record indicates the adjustable object to be read only at all (table values and axis pts).
-> REF_MEMORY_SEGMENT	Reference to the memory segment which is needed if the address is not unique (this occurs in the case of lapping address ranges (overlapping memory segments).
-> STEP_SIZE	This keyword can be used to define a delta value which is added to or subtracted from the current value when using up/down keys while calibrating.
-> SYMBOL_LINK	Reference to symbol name within a linker map file.

Description:

Specification of parameters for the handling of an axis points distribution.

Example:

```
/begin AXIS_PTS        STV_N       /* name */
                       "axis points distribution speed"
                                   /* long identifier */
                       0x9876      /* address */
                       N           /* input quantity */
                       DAMOS_SST   /* deposit */
                       100.0       /* maxdiff */
                       R_SPEED     /* conversion */
                       21        /* maximum number of axis points */
                       0.0         /* lower limit */
                       5800.0      /* upper limit */
                       GUARD_RAILS /* uses guard rails*/
  REF_MEMORY_SEGMENT   Data3
  /begin FUNCTION      LIST
```

```
                               ID_ADJUSTM
                               FL_ADJUSTM
                               SPEED_LIM
      /end FUNCTION_           LIST
      /begin IF_DATA           DIM
                               EXTERNAL
                               DIRECT
      /end IF_DATA
      CALIBRATION_ACCESS       CALIBRATION
   /end AXIS_PTS
```

3.5.19 AXIS_PTS_REF

Prototype:

```
AXIS_PTS_REF                  ident     AxisPoints
```

Parameters:

ident AxisPoints	Name of the AXIS_PTS data record which describes the axis points distribution (group axis points and record layout: see AXIS_PTS).

Description:

If the addresses of the axis point values are separated from the table values in the emulation memory and must be described by a special AXIS_PTS data record, the data record is referenced by means of the keyword AXIS_PTS_REF.

Example:

```
/* Group characteristic curve with reference to axis points distribution GRP_N */
/begin CHARACTERISTIC    TORQUE                    /* name */
                         "Torque limitation"       /* long identifier */
                         CURVE                     /* type*/
                         0x1432                    /* address */
                         DAMOS_GKL                 /* deposit */
                         0.2                       /* maxdiff */
                         R_TORQUE                  /* conversion */
                         0.0                       /* lower limit */
                         43.0                      /* upper limit */
   /begin IF_DATA        DIM
                         EXTERNAL
                         INDIRECT
   /end IF_DATA
   /begin AXIS_DESCR     /* description of X-axis points */
                         COM_AXIS                  /* common axis points */
                         N                         /* input quantity */
                         CONV_N                    /* conversion */
                         14                        /* max. no. of axis p.*/
                         0.0                       /* lower limit */
                         5800.0                    /* upper limit */
      AXIS_PTS_REF       GRP_N
   /end AXIS_DESCR
/end CHARACTERISTIC

/* Axis points distribution data record */
/begin AXIS_PTS          GRP_N                     /* name */
                         "Group axis points speed" /* long identifier */
                         0x1032                    /* address */
                         N                         /* input quantity */
                         DAMOS_GST                 /* deposit */
                         50.0                      /* maxdiff */
                         CONV_N                    /* conversion */
                         11                        /* max. no. of axis points */
                         0.0                       /* lower limit */
                         5800.0                    /* upper limit */
```

```
        /begin IF_DATA       DIM
                             EXTERNAL
                             INDIRECT
        /end IF_DATA
     /end AXIS_PTS
```

3.5.20 AXIS_PTS_X / _Y / _Z / _4 / _5

Prototype:

```
AXIS_PTS_X / _Y / _Z / _4 / _5
                              uint        Position
                              datatype    Datatype
                              indexorder  IndexIncr
                              addrtype    Addressing
```

Parameters:

uint	Position	Position of the axis point values in the deposit structure (description of sequence of elements in the data record).
datatype	Datatype	Data type of the axis point values
indexorder	IndexIncr	Decreasing or increasing index with increasing addresses
addrtype	Addressing	Addressing of the table values (see enum addrtype).

Description:

Description of the X, Y, Z, Z4 or Z5 axis points in the memory (see keyword RECORD_LAYOUT).

Note: If the Alternate option is used with FNC_VALUES, the position parameter determines the order of values and axis points.

Example:

```
AXIS_PTS_X              3
                        ULONG
                        INDEX_INCR
                        DIRECT
```

3.5.21 AXIS_RESCALE_X / _Y / _Z / _4 / _5

Prototype:

```
AXIS_RESCALE_X /_Y /_Z / _4 / _5
                               uint          Position
                               datatype      Datatype
                               uint          MaxNumberOfRescalePairs
                               indexorder    IndexIncr
                               addrtype      Adressing
```

Parameters:

uint	Position	position of the rescale axis point value pairs in the deposit structure (description of sequence of elements in the data record).
datatype	Datatype	Data type of the rescale axis point values
uint	MaxNumberOfRescalePairs	maximum number of rescaling axis point pairs (see NO_RESCALE_PTS_X / _Y / _Z / _4 / _5)
indexorder	IndexIncr	Decreasing or increasing index with increasing addresses
addrtype	Adressing	Addressing of the table values (see enum addrtype).

Description:

Description of rescaling the axis values of an adjustable object. A rescale axis consists mainly of a number of rescaling axis points pairs ($axis_i$, $virtual_i$) which describe a rescale mapping between the axis points and a virtual axis that is used for the access of the table function values deposited in the control unit. Between two pairs the mapping is linear. Both, the axis points and the virtual axis points must be in ascending order. Consider, for example, the three rescale pairs (0x00, 0x00), (0x64, 0xC0) and (0xD8, 0xFF). Then all axis points between 0x00 and 0x64 are mapped linear to the virtual axis [0x00, 0xC0], and all axis points between 0x64 and 0xD8 are mapped linear to the virtual axis [0xC0, 0xFF]:

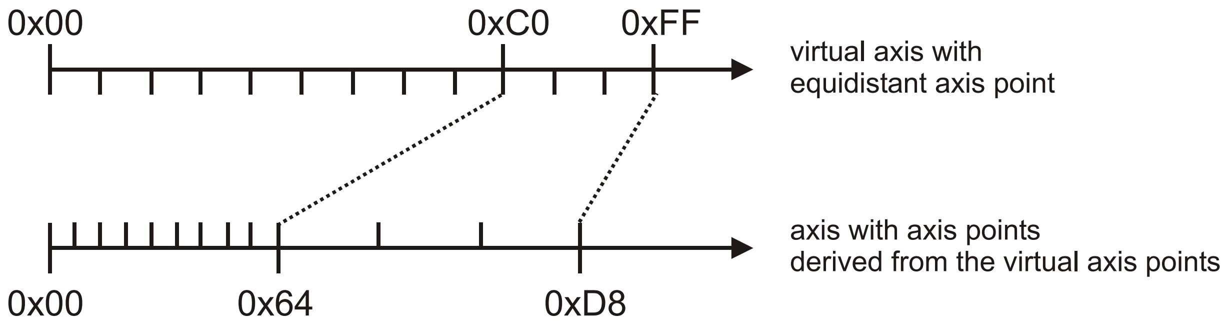


Figure 2 AXIS_RESCALE_X / _Y / _Z / _4 / _5

Accordingly, to each axis point there is a virtual axis point. The virtual axis points are distributed equidistantly on the virtual axis including the axis limits, e.g. the virtual axis points can be derived from the size of the virtual axis and the number of axis points. According to the rescale mapping the axis point can be computed from the virtual axis points. The following algorithm can be applied, where D is the length of the (equidistant) intervals on virtual axis:

```
$D = \frac{\text{last virtual axis point - first virtual axis point} + 1}{no\_axis\_pts - 1}$        $k = 1$

FOR i = 1 TO $(no\_rescale\_x - 1)$

    FOR $k * d + virtual_1 < virtual_{i+1}$
    /* repeat for the number of points in the interval on the virtual axis */

    $k = k + 1$

    $X_k = axis_i + ((k-1)D - virtual_i)\frac{axis_{i+1} - axis_i}{virtual_{i+1} - virtual_i}$

$X_1 = axis_1$
$X_{no\_axis\_pts} = axis_{no\_rescale\_x}$
```

It is recommended that D is a power of 2, i.e. if the size of the virtual axis is 256, the number of axis points should be $no\_axis\_pts = 2^n + 1$ = {3, 5, 9, 17, 33}.

The following example makes clear how the evaluation of the formula can be used to derive the actual axis points. We have no_of_rescale_pairs = 3 and virtual$_1$ = 0x00 = 0, virtual$_2$ = 0xC0 = 192, virtual$_3$ = 0xFF = 255, axis$_1$ = 0x00 = 0, axis$_2$= 0x64 = 100, axis$_3$ = 0xD8 = 216. Assume no_axis_pts = 9, and therefore D = 32. The first of the two executions of the inner loop (j-loop) is on virtual$_2$ – virtual$_1$/ D = 192/32 = 6 iterations. For each iteration (axis$_2$ – axis$_1$)/(virtual$_2$ – virtual$_1$) = 100/192, and therefore
X$_2$ = 0 + 32 * 100/192 = 16,666,
X$_3$ = 0 + 64 * 100/192 = 33,333,
X$_4$ = 0 + 96 * 100/192 = 50,
X$_5$ = 0 + 128 * 100/192 =66,666,
X$_6$ = 0 + 160 * 100/192 = 83,333.
For the second execution there are virtual$_3$ – virtual$_2$ / D = 2 iterations with (axis$_3$ – axis$_2$)/(virtual$_3$ – virtual$_2$) = 116/64. Consequently
X$_7$ = 100 + (192 – 192) * 116/64 = 100 and
X$_8$ = 100 + (224 – 192) * 116/64 = 158.
Also X$_1$ = axis$_1$ = 0 and X$_9$ = axis$_3$ = 216.

Example:

```
    AXIS_RESCALE_X            3
                              UBYTE
                              5
                              INDEX_INCR
                              DIRECT
```

3.5.22 BIT_MASK

Prototype:

```
BIT_MASK                    ulong    Mask
```

Parameters:

ulong	Mask	mask to mask out single bits

Description:

The BIT_MASK keyword can be used to mask out single bits of the value to be processed. The least significant bit in BIT_MASK determines how far the masked value is shifted to the right.

Example:

BIT_MASK 0x00000FFF

Value to be masked	BIT_MASK	Result
10110110	0x1 = 1 (bin)	0 (bin)
10110110	0x2 = 10 (bin)	1 (bin)
10110110	0x6 = 110 (bin)	11 (bin)
10110110	0xC = 1100 (bin)	01 (bin)
10111010	0xC = 1100 (bin)	10 (bin)
10111110	0xC = 1100 (bin)	11 (bin)
10111110	0xA = 1010 (bin)	101 (bin)

Note: The newly added comments about the least significant bit and the inserted samples are valid only while no keyword BIT_OPERATION is used. If the keyword BIT_OPERATION is used then its defined parameters dominate those parameters of the BIT_MASK keyword.

If it is required to use BIT_MASK without a shift operation, then use BIT_OPERATION with a right or left shift of zero, as shown in the following example.

Example:

```
BIT_MASK 0x40
/begin BIT_OPERATION
            LEFT_SHIFT    0
/end BIT_OPERATION
```

3.5.23 BIT_OPERATION

Prototype:

```
/begin BIT_OPERATION
                          [-> LEFT_SHIFT]
                          [-> RIGHT_SHIFT]
                          [-> SIGN_EXTEND]
/end BIT_OPERATION
```

Parameters:

none

Optional parameters

-> LEFT_SHIFT	Number of positions to left shift data, zeros will be shifted in from the right.
-> RIGHT_SHIFT	Number of positions to right shift data, zeros will be shifted in from the left.
-> SIGN_EXTEND	Gives a sign extension of sign bit for measurement data.

Description:

The BIT_OPERATION keyword can be used to perform operation on the masked out value.
First BIT_MASK will be applied on measurement data, then LEFT_SHIFT / RIGHT_SHIFT is performed and last the SIGN_EXTEND is carried out.

SIGN_EXTEND means that the sign bit (masked data's leftmost bit) will be copied to all bit positions to the left of the sign bit. This results in a new datatype with the same signed value as the masked data.

Example:

```
/begin BIT_OPERATION
            RIGHT_SHIFT   4                 /*4 positions*/
            SIGN_EXTEND
/end BIT_OPERATION
```

Table 7 BIT_OPERATION

Explanation	Data	Comment
Data after mask operation	0000000000100000	
Data after shift operation	0000000000000010	shifted right 4 positions
Data after sign extend	1111111111111110	

3.5.24 BYTE_ORDER

Prototype:

```
BYTE_ORDER                  byteorder ByteOrder
```

Parameters:

byteorder	ByteOrder	Byte order of the relevant quantity in the ECU program Note: Use of LITTLE_ENDIAN and BIG_ENDIAN defined with keyword BYTE_ORDER in version 1.0 leads to mistakes because it is in contradiction to general use of terms „little endian“ and „big endian“. Since version 1.2 the keywords LITTLE_ENDIAN and BIG_ENDIAN are permissible but should not longer be used. They should be replaced by MSB_LAST and MSB_FIRST which are equivalent: MSB_LAST corresponds to the Intel format (equivalent former keyword is BIG_ENDIAN). MSB_FIRST corresponds to the Motorola format (equivalent former keyword is LITTLE_ENDIAN).

Description:

Where the standard value does not apply this parameter can be used to specify the byte order (Intel format, Motorola format).

Example:

```
BYTE_ORDER                  MSB_LAST
```

Table 8 Byte order - memory data deposition

Byte Order	Keyword	Former Keyword	Increasing address -->				
			n	n+1	....	n + (N-1)	n + N
Motorola Format	MSB_FIRST	LITTLE_ENDIAN	$Byte_N$ (Most Significant Byte)	$Byte_{N-1}$	....	$Byte_1$	$Byte_0$ (Least Significant Byte)
Intel Format	MSB_LAST	BIG_ENDIAN	$Byte_0$ (Least Significant Byte)	$Byte_1$	....	$Byte_{N-1}$	$Byte_N$ (Most Significant Byte)

3.5.25 CALIBRATION_ACCESS

Prototype

```
CALIBRATION_ACCESS          enum      Type
```

Parameters

enum Type	Possible Types: CALIBRATION Characteristic or axis points with calibration allowed. NO_CALIBRATION This keyword can be used to indicate that the axis points cannot be changed (but can be read only). Note: This optional keyword used at CHARACTERISTIC record indicates the adjustable object to be read only at all (table values and axis pts). NOT_IN_MCD_SYSTEM Internal characteristic or axis points which are not readable or writeable by the MCD-System. If there are references between AXIS_PTS and CHARACTERISTICS with one of them gets this value of CALIBRATION_ACCESS, the other one must get this value too. OFFLINE_CALIBRATION Variables which can be flashed but not emulated ore calibrated, e.g. values representing safety relevant property while driving.

Description

This keyword specifies the access of a CHARACTERISTIC or AXIS_PTS for calibration. It substitutes the READ_ONLY attribute since ASAM-MCD-2MC V1.4

Example

```
    CALIBRATION_ACCESS       CALIBRATION
```

3.5.26 CALIBRATION_HANDLE

Prototype:

```
/begin CALIBRATION_HANDLE
                          (long    Handle)*
                          [-> CALIBRATION_HANDLE_TEXT]
/end CALIBRATION_HANDLE
```

Parameters:

long	Handle	Handle for the calibration method

Optional parameters

-> CALIBRATION_HANDLE_TEXT	Additional text for a calibration handle.

Description:

Definition of the calibration method specific. The interpretation of this data depends on the calibration method used.

Example:

```
/begin CALIBRATION_HANDLE
                       0x10000 /* start address of pointer table */
                       0x200   /* length of pointer table */
                       0x4     /* size of one pointer table entry */
                       0x30000 /* start address of flash section */
                       0x20000 /* length of flash section */
                       CALIBRATION_HANDLE_TEXT "Nmot"
/end CALIBRATION_HANDLE
```

3.5.27 CALIBRATION_HANDLE_TEXT

Prototype:

```
CALIBRATION_HANDLE_TEXT   string   text
```

Parameters:

string	text	text string

Description:

Additional text for a calibration handle.

Example:

```
 CALIBRATION_HANDLE_TEXT   "Torque"
```

3.5.28 CALIBRATION_METHOD

Prototype:

```
/begin CALIBRATION_METHOD string   Method
                          ulong    Version
                          {-> CALIBRATION_HANDLE}
/end CALIBRATION_METHOD
```

Parameters:

string Method	the string identifies the calibration method to be used. A convention regarding the meaning of the calibration methods. The following strings are already in use: 'InCircuit', 'SERAM', 'DSERAP', 'BSERAP'
ulong Version	Version number of the method used

Optional Parameters:

-> CALIBRATION_HANDLE	Contains the (method specific) arguments for the calibration method. The arguments themselves and their meaning are dependent of the calibration method.

Description:

This keyword is used to indicate the different methods of access that are implemented in the ECU and that can be used regardless of the actual interface of the ECU.

Example:

```
/begin CALIBRATION_METHOD
    „InCircuit"
    2
    /begin CALIBRATION_HANDLE
         0x10000         /* start address of pointer table */
         0x200           /* length of pointer table */
         0x4             /* size of one pointer table entry */
         0x10000         /* start address of flash section */
         0x10000         /* length of flash section */
    /end CALIBRATION_HANDLE
/end CALIBRATION_METHOD
```

3.5.29 CHARACTERISTIC

Prototype:

```
/begin CHARACTERISTIC      ident    Name
                           string   LongIdentifier
                           enum     Type
                           ulong    Address
                           ident    Deposit
                           float    MaxDiff
                           ident    Conversion
                           float    LowerLimit
                           float    UpperLimit
                           {-> ANNOTATION}*
                           {-> AXIS_DESCR}*
                           [-> BIT_MASK]
                           [-> BYTE_ORDER]
                           [-> CALIBRATION_ACCESS]
                           [-> COMPARISON_QUANTITY]
                           [-> DEPENDENT_CHARACTERISTIC]
                           [-> DISCRETE]
                           [-> DISPLAY_IDENTIFIER]
                           [-> ECU_ADDRESS_EXTENSION]
                           [-> EXTENDED_LIMITS]
                           [-> FORMAT]
                           [-> FUNCTION_LIST]
                           [-> GUARD_RAILS]
                           {-> IF_DATA}*
                           [-> MAP_LIST]
                           [-> MATRIX_DIM]
                           [-> MAX_REFRESH]
                           [-> NUMBER]
                           [-> PHYS_UNIT]
                           [-> READ_ONLY]
                           [-> REF_MEMORY_SEGMENT]
                           [-> STEP_SIZE]
                           [-> SYMBOL_LINK]
                           [-> VIRTUAL_CHARACTERISTIC]
/end CHARACTERISTIC
```

Parameters:

ident	Name	unique identifier in the ECU program Note: The name of the adjustable object has to be unique within all measurement objects and adjustable objects of the ASAM MCD-2MC MODULE, i.e. there must not be another AXIS_PTS, CHARACTERISTIC or MEASUREMENT object with the same identifier in the MODULE.
string	LongIdentifier	comment, description
enum	Type	possible Types:

ASCII	(string)
CURVE	(1-dimensional array with axes)
MAP	(2-dimensional array with axes)
CUBOID	(3-dimensional array with axes)
CUBE_4	(4-dimensional array with axes)
CUBE_5	(5-dimensional array with axes)
VAL_BLK	(array without axes)
VALUE	(scalar)

ulong	Address	address of the adjustable object in the emulation memory
ident	Deposit	reference to the corresponding data record for description of the record layout (see RECORD_LAYOUT)
float	Maxdiff	maximum float with respect to an adjustment of a table value Note: This value is interpreted as an absolute value, not as a percentage.
ident	Conversion	Reference to the relevant record of the description of the conversion method (see COMPU_METHOD). If there is no conversion method, as in the case of CURVE_AXIS, the parameter 'Conversion' should be set to "NO_COMPU_METHOD (measurement and calibration systems must be able to handle this case).
float	LowerLimit	plausible range of table values, lower limit
float	UpperLimit	plausible range of table values, upper limit Note: Depending on the type of conversion, the limit values are interpreted as physical or internal values. For conversions of type COMPU_VTAB and COMPU_VTAB_RANGE, the limit values are interpreted as internal values. For all other conversion types, the limit values are interpreted as physical values.

Optional parameters

-> ANNOTATION	Set of notes (represented as multi-line ASCII description texts) which are related. Can serve e.g. as application note.
-> AXIS_DESCR	This keyword is used to specify the parameters for the axis description (with characteristic curves and maps). The first parameter block describes the X-axis, the second parameter block the Y-axis, the third parameter block the Z-axis (CUBOID), the fourth parameter block the Z4-axis (CUBE_4), the fifth parameter block the Z5-axis (CUBE_5). Exception: For MAP_LIST only one AXIS_DESCR is accepted that describes the Z-axis. (The X- and Y-axes are described at the MAPs referenced by MAP_LIST)
-> BIT_MASK	This parameter can be used to specify a bit mask for the handling of single bits.
-> BYTE_ORDER	Where the standard value does not apply this parameter can be used to specify the byte order (Intel format, Motorola format) if the standard value is not to be used.
-> CALIBRATION_ACCESS	This keyword specifies the access of the characteristic for calibration. Use it instead of the READ_ONLY Attribute.
-> COMPARISON_QUANTITY	This keyword references a valid MEASUREMENT in the ASAM MCD-2MC file.

<u>Semantic Interpretation</u> (for a CURVE, a CHARACTERISTIC with only one AXIS_DESC) : The conventional workpoint for a -CURVE has only one input quantity (assigned to AXIS_DESCR) and moves on the CURVE. The 'free-moving' workpoint in an xy diagram of a CURVE is described by two quantities (the conventional input quantity with the AXIS_DESC, the x-axis, and an additional comparison quantity described as an optional attribute directly with the CURVE, the y-axis).The 'free-moving' workpoint does not move on the CURVE, but on the xy-diagram in which the CURVE is located. The crossing of the free-moving workpoint and the CURVE would describe an EVENT. Such display is required by calibration engineers of automatic transmission control (EVENT=gear shift). When this keyword with a CURVE is present, the workpoint display of the MCD system shall apply the INPUT_QUANTITY and the COMPARISON_QUANTITY in the xy-diagram.

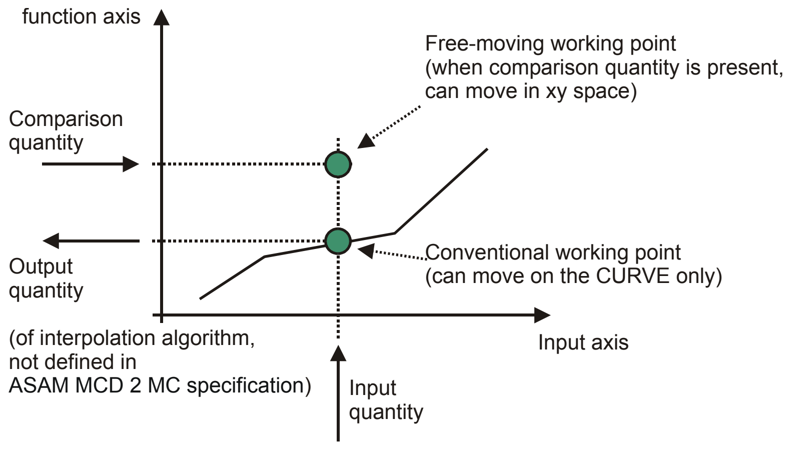

function axis
Free-moving working point
(when comparison quantity is present,
can move in xy space)
Comparison
quantity
Output
quantity
Conventional working point
(can move on the CURVE only)
(of interpolation algorithm,
not defined in
ASAM MCD 2 MC specification)
Input axis
Input
quantity

Figure 3 Comparison quantity

-> DEPENDENT_CHARACTERISTIC Describes the formula and references to characteristics, upon which this characteristic depends on.

<u>Note:</u> The dependence graph described by the dependence relation must be acyclic. This must be ensured by the producer of the ASAM MCD-2MC file. This keyword is only valid for characteristics of type VALUE

-> DISCRETE This keyword indicates that the characteristic values are discrete values which should not be interpolated – e.g. in graphic display windows or further calculations. This flag can be used e.g. for integer objects describing states. If the keyword is not specified the values are interpreted as continuous values which can be interpolated.

-> DISPLAY_IDENTIFIER Can be used as a display name (alternative to the ‘name’ attribute).

-> ECU_ADDRESS_EXTENSION	This keyword is an additional address information. For instance it can be used, to distinguish different address spaces of an ECU (multi-microcontroller devices).
-> EXTENDED_LIMITS	This keyword can be used to specify an extended range of values. In the measurement and calibration system, for example, when leaving the standard range of values (lower limit...upper limit) a warning could be generated (extended limits enabled only for "power user").
-> FORMAT	With deviation from the display format specified with keyword COMPU_TAB referenced by parameter <Conversion> a special display format can be specified to be used to display the table values.
-> FUNCTION_LIST	This keyword can be used to specify a list of 'functions' to which the relevant adjustable object is allocated (function orientation).
-> GUARD_RAILS	This keyword is used to indicate that an adjustable CURVE or MAP uses guard rails. The Measurement and Calibration System does not allow the user to edit the outermost values of the adjustable object (see GUARD_RAILS).
-> IF_DATA	Data record to describe interface specific data of the characteristic. The parameters associated with this keyword have to be described in the ASAM MCD-2MC metalanguage. -> MAP_LIST For the adjustable object type CUBOID which are `sliced', this keyword specifies the MAPs which comprise the cuboids. Note: The MAPs referenced by MAP_LIST may have different number of axis points. Note: MAP_LIST is supported only for CUBOID, not for CUBE_4 or CUBE_5.
-> MATRIX_DIM	Shows the size and dimension of a multidimensional characteristic (e.g. VAL_BLK). If the MATRIX_DIM keyword is used, then the option NUMBER is not needed. However, if the keywords NUMBER and MATRIX_DIM are both used, the resulting value in NUMBER must be the same as xDim * yDim * zDim for MATRIX_DIM. If the keyword is missing the array has only one dimension with the size given at NUMBER.
-> MAX_REFRESH	Maximum refresh rate of this (adaptive) characteristic in the control unit. The existence of the keyword implies that the value of the characteristic is changed by the control unit (adaptive characteristics).
-> NUMBER	For the adjustable object types 'fixed value block' (VAL_BLK) and 'string' (ASCII), this keyword specifies the number of fixed values and characters respectively.
-> PHYS_UNIT	With this keyword a physical unit can be specified for the characteristic object if no conversion rule is referenced (NO_COMPU_METHOD). Note: If a conversion rule is referenced the additional usage of PHYS_UNIT overrules the

	unit specified at the referenced conversion rule.
-> READ_ONLY	This keyword can be used to indicate that the adjustable object cannot be changed (but can be read only). This keyword indicates the adjustable object to be read only at all (table values and axis points). The optional keyword used at AXIS_DESCR record indicates the related axis points to be read only
-> REF_MEMORY_SEGMENT	Reference to the memory segment which is needed if the address is not unique (this occurs in the case of lapping address ranges (overlapping memory segments).
-> STEP_SIZE	This keyword can be used to define a delta value which is added to or subtracted from the current value when using up/down keys while calibrating.
-> SYMBOL_LINK	Reference to symbol name within a linker map file.
-> VIRTUAL_CHARACTERISTIC	Marks a characteristic as being virtual, i.e. not existing in the memory of the control unit. The address can therefore be ignored for virtual characteristic. Initial value of the virtual characteristic depends on the values of other characteristic.
	Note: The corresponding graph (in analogy to the dependence graph) must also be acyclic and each sink of the graph must be a non virtual characteristic. This must be ensured by the producer of the ASAM MCD-2MC file. This keyword is only valid for characteristics of type VALUE.

Description:

Specification of the parameters for the processing of an adjustable object.

Example:

```
/begin CHARACTERISTIC    PUMKF     /* name */
                         "Pump characteristic map"
                                   /* long identifier */
                         MAP       /* type */
                         0x7140    /* address */
                         DAMOS_KF  /* deposit */
                         100.0     /* maxdiff */
                         R_VOLTAGE /* conversion */
                         0.0       /* lower limit */
                         5000.0    /* upper limit */
          MAX_REFRESH    3 15      /* 15 msec */
      /begin DEPENDENT_CHARACTERISTIC
                         "sin(X1)"
                         ALPHA
      /end DEPENDENT_CHARACTERISTIC
      /begin VIRTUAL_CHARACTERISTIC
                         „sqrt(X1)"
                         B_AREA
      /end VIRTUAL_CHARACTERISTIC
      REF_MEMORY_SEGMENT Data1
      /begin FUNCTION_LIST
                         NL_ADJUSTMENT
```

```
                            FL_ADJUSTMENT
                            SPEED_LIM
          /end FUNCTION_LIST
          /begin IF_DATA
                            DIM
                            EXTERNAL
                            INDIRECT
          /end IF_DATA
          /begin AXIS_DESCR /* description of X-axis points */
                            STD_AXIS /* standard axis points */
                            N        /* reference to input quantity */
                            CON_N    /* conversion */
                            13       /* maximum number of axis points*/
                            0.0      /* lower limit */
                            5800.0   /* upper limit */
               MAX_GRAD     20.0     /* X-axis: maximum gradient */
          /end AXIS_DESCR
          /begin AXIS_DESCR /* description of Y-axis points */
                            STD_AXIS /* standard axis points */
                            AMOUNT   /* reference to input quantity */
                            CON_ME   /* conversion */
                            17       /* maximum number of axis points*/
                            0.0      /* lower limit */
                            43.0     /* upper limit */
          /end AXIS_DESCR
    /end CHARACTERISTIC
```

3.5.30 COEFFS

Prototype:

`COEFFS`	`float a b c d e f`

Parameters:

float a, b, c, d, e, f:	coefficients for the specified formula: f(x) = (axx + bx + c) / (dxx + ex + f)

Description:

Specification of coefficients for the formula f(x) = (axx + bx + c) / (dxx + ex + f). This term describes the conversion from physical values to control unit internal values:

INT = f(PHYS);

Note: For these coefficients restrictions have to be defined because this general equation cannot always be inverted.

Example:

```
COEFFS   0 4 8 0 0 5
/* Control unit internal values of revolutions (INT) is calculated  from */
/* physical values (PHYS: unit of PHYS is [rpm]) as follows: */
/* INT = (4/5) * PHYS/[rpm] + (8/5)                    */
/* inverted:    PHYS/[rpm] = 1.25 * INT - 2.0    */
```

3.5.31 COEFFS_LINEAR

Prototype:

```
COEFFS_LINEAR                float     a   b
```

Parameters:

float a, b:	coefficients for the specified formula: f(x) = ax + b

Description:

Specification of coefficients for the formula f(x) = ax + b. This term describes the conversion from control unit internal values to physical values:

PHYS = f(INT);

Example:

```
COEFFS_LINEAR   1.25  -2.0
/* The physical value (PHYS) with unit is calculated from the      */
/* control unit’s internal value of revolutions (INT) as follows:  */
/*       PHYS = 1.25 * INT - 2.0          */
```

3.5.32 COMPARISON_QUANTITY

Prototype:

```
COMPARISON_QUANTITY        ident     Name
```

Parameters:

ident Name	Unique identifier in the program (Reference to a valid MEASUREMENT)

Description:

This keyword references a valid MEASUREMENT in the ASAM MCD-2MC file.

Semantic Interpretation (for a CURVE, a CHARACTERISTIC with only one AXIS_DESC): The conventional work point for a -CURVE has only one input quantity (assigned to AXIS_DESCR) and moves on the CURVE. The 'free-moving' work point in an xy diagram of a CURVE is described by two quantities (the conventional input quantity with the AXIS_DESC, the x-axis, and an additional comparison quantity described as an optional attribute directly with the CURVE, the y-axis).The 'free-moving' work point does not move on the CURVE, but on the xy-diagram in which the CURVE is located. The crossing of the free-moving work point and the CURVE would describe an EVENT. Such display is required by calibration engineers of automatic transmission control (EVENT=gear shift). When this keyword with a CURVE is present, the work point display of the MCD system shall apply the INPUT_QUANTITY and the COMPARISON_QUANTITY in the xy-diagram.

3.5.33 COMPU_METHOD

Prototype:

```
/begin COMPU_METHOD         ident    Name
                            string   LongIdentifier
                            enum     ConversionType
                            string   Format
                            string   Unit
                            [-> COEFFS]
                            [-> COEFFS_LINEAR]
                            [-> COMPU_TAB_REF]
                            [-> FORMULA]
                            [-> REF_UNIT]
                            [-> STATUS_STRING_REF]
/end COMPU_METHOD
```

Parameters:

ident Name

unique identifier in the program for the conversion method

Note: The name of the conversion method has to be unique within all conversion methods of the ASAM MCD-2MC MODULE, i.e. there must not be another COMPU_METHOD object with the same identifier in the MODULE.

string LongIdentifier

comment, description

enum ConversionType

possible Types:

IDENTICAL: no conversion of the internal source value
The following equation is applied:
PHYS = INT

FORM: conversion based on the formula specified by the FORMULA keyword.

LINEAR: linear function of the following type:
f(x)=ax + b
for which:
PHYS=f(INT)
The coefficients a and b have to be specified by the COEFFS_LINEAR keyword.

RAT_FUNC: fractional rational function of the following type
f(x)=(axx + bx + c)/(dxx + ex + f)
for which:
INT = f(PHYS)
Coefficients a, b, c, d, e, f have to be specified by the COEFFS keyword.
Note: For linear functions, use the ConversionType LINEAR, for ident functions the ConversionType IDENT. For non linear functions it must be possible to invert the formula within the limits of the

		AXIS_PTS, CHARACTERISTIC or MEASUREMENT where it is used. Otherwise use the ConversionType FORM.
		TAB_INTP table with interpolation
		TAB_NOINTP table without interpolation
		TAB_VERB verbal conversion table
string	Format	display format in %[length].[layout]; length indicates the overall length; layout indicates the decimal places. The format string should never be empty as "".
string	Unit	physical unit

Optional parameters:

-> COEFFS	This keyword is used to specify coefficients a, b, c, d, e, f for the fractional rational function of the following type: f(x)=(axx + bx + c) / (dxx + ex + f)
-> COEFFS_LINEAR	This keyword is used to specify the coefficients a and b of the linear function of the following type: f(x)=ax + b
-> COMPU_TAB_REF	This keyword is used to specify a conversion table (reference to COMPU_TAB data record).
-> FORMULA	Formula to be used for the conversion
-> REF_UNIT	This keyword is used to reference a measurement unit (e.g. an object of type UNIT). The string parameter Unit is a redundant information because the record referenced by REF_UNIT contain it too. Just for the purpose of compatibility with previous versions of ASAM MCD-2MC the parameter REF_UNIT is optional. -> STATUS_STRING_REF This keyword is used to split up the value range of ECU internal values into a numerical and a verbal part. The verbal part can be used to visualize status information (e.g. “Sensor not connected”).

Description:

Specification of a conversion method

Examples:

```
/begin COMPU_METHOD       TMPCON1       /* name */
                          "conversion method for engine temperature"
                          TAB_NOINTP    /* convers_type */
                          "%4.2"        /* display format */
                          "°C"          /* physical unit */
    COMPU_TAB_REF         MOTEMP1
/end COMPU_METHOD

/begin COMPU_METHOD       CM_IDENTITY   /* name */
                          "conversion method identity (no formula)"
                          IDENTICAL/* convers_type */
                          "%4.0"    /* display format */
                          ""        /* physical unit */
/end COMPU_METHOD
```

```
/begin COMPU_METHOD      CM_LINFUNC    /* name */
                         "conversion method for linear function"
                         LINEAR    /* convers_type */
                         "%4.0"    /* display format */
                         "rpm"     /* physical unit */
                         COEFFS_LINEAR  2.0 5.0
/end COMPU_METHOD

/begin COMPU_METHOD      TMPCON2      /* name */
                         "conversion method for air temperature"
                         FORM         /* convers_type */
                         "%4.2"       /* display format */
                         "°C"         /* physical unit */
  /begin FORMULA         "3*X1/100 + 22.7"
  /end FORMULA
/end COMPU_METHOD

/begin COMPU_METHOD      CM_DiagStatus /* name */
                         ""            /*convers_type */
                         TAB_VERB      /*convers_type */
                         "%0.0"        /* display format */
                         ""            /* physical unit */
  COMPU_TAB_REF          CT_DiagStatus
/end COMPU_METHOD

/begin COMPU_METHOD      CM_RPM        /* name */
                         "conversion method for engine rpm"
                         TAB_INTP      /*convers_type */
                         "%7.1"        /* display format */
                         "rpm "        /* physical unit */
  COMPU_TAB_REF          CT_RPM
/end COMPU_METHOD

/begin COMPU_METHOD      CM_NM         /* name */
                         " conversion method for air temperature "
                         TAB_INTP      /* convers_type */
                         "%7.1"        /* display format */
                         "nm "         /* physical unit */
  COMPU_TAB_REF          CT_NM
/end COMPU_METHOD

/begin COMPU_METHOD      FIXED_UW_03
                         "Conversion method for FIXED_UW_03"
                         RAT_FUNC      /* convers_type */
                         "%8.3"        /* display format */
                         "NO_PHYSICAL_QTY"
                         COEFFS 0 8 0 0 0 1
/end COMPU_METHOD

/begin COMPU_METHOD      BYTE
                         "Conversion method for BYTE"
                         RAT_FUNC      /* convers_type */
                         "%3.0"        /* display format */
                         "NO_PHYSICAL_QTY"
                         COEFFS 0 1 0 0 0 1
/end COMPU_METHOD
```

```
/begin COMPU_METHOD     SHORTINT
                        “Conversion method for SHORTINT”
                        RAT_FUNC
                        “%4.0”
                        “NO_PHYSICAL_QTY”
                        COEFFS 0 1 0 0 0 1
/end COMPU_METHOD

/begin COMPU_METHOD     WORD
                        “Conversion method for WORD”
                        RAT_FUNC
                        “%5.0”
                        “NO_PHYSICAL_QTY”
                        COEFFS 0 1 0 0 0 1
/end COMPU_METHOD

/begin COMPU_METHOD     INTEGER
                        “Conversion method for INTEGER”
                        RAT_FUNC
                        “%6.0”
                        “NO_PHYSICAL_QTY”
                        COEFFS 0 1 0 0 0 1
/end COMPU_METHOD

/begin COMPU_METHOD     LONGWORD
                        “Conversion method for LONGWORD”
                        RAT_FUNC
                        “%10.0”
                        “NO_PHYSICAL_QTY”
                        COEFFS 0 1 0 0 0 1
/end COMPU_METHOD

/begin COMPU_METHOD     LONGINT
                        “Conversion method for LONGINT”
                        RAT_FUNC
                        “%11.0”
                        “NO_PHYSICAL_QTY”
                        COEFFS 0 1 0 0 0 1
/end COMPU_METHOD
```

3.5.34 COMPU_TAB

Prototype:

```
/begin COMPU_TAB          ident    Name
                          string   LongIdentifier
                          enum     ConversionType
                          uint     NumberValuePairs
                          { float  InVal
                            float  OutVal }*
                          [-> DEFAULT_VALUE]
                          [-> DEFAULT_VALUE_NUMERIC]
/end COMPU_TAB
```

Parameters:

ident	Name	unique identifier in the program for the conversion table Note: The name of the conversion table has to be unique within all conversion tables of the ASAM MCD-2MC MODULE, i.e. there must not be another COMPU_TAB, COMPU_VTAB or COMPU_VTAB_RANGE object with the the same identifier in the MODULE.
string	LongIdentifier	comment, description
enum	ConversionType	possible Types: TAB_INTP table with interpolation TAB_NOINTP table without interpolation Note: This parameter is redundant information because the record defined with COMPU_METHOD also contains it.
uint	NumberValuePairs	number of successive value pairs for this conversion table
float	InVal	axis point
float	OutVal	axis value

Optional parameters

-> DEFAULT_VALUE	string used as OutVal for display when the ECU value is out of any declared range. This string is not selectable for calibration (when writing to the ECU). This parameter cannot be used in combination with DEFAULT_VALUE_NUMERIC. For COMPU_TAB it is recommended to use DEFAULT_VALUE_NUMERIC rather than DEFAULT_VALUE.
-> DEFAULT_VALUE_NUMERIC	Float value used as OutVal for display when the ECU value is out of any declared range. This value is not selectable for calibration (when writing to the ECU). The value is handled like a physical value (transferred via ASAM MCD-3 [ASAM MCD-3]). This parameter must not be used in combination with DEFAULT_VALUE.

Description:

Conversion table for conversions that cannot be represented as a function.

Example:

```
/begin COMPU_TAB        TT                  /* name */
                        "conversion table for oil temperatures"
                        TAB_NOINTP          /* convers_type */
                        7                   /* number_value_pairs */
                        1  4.3  2  4.7  3  5.8  4  14.2  5  16.8
                        6  17.2  7  19.4    /* value pairs */
                        DEFAULT_VALUE_NUMERIC 99.0
/end COMPU_TAB
```

3.5.35 COMPU_TAB_REF

Prototype:

```
COMPU_TAB_REF              ident    ConversionTable
```

Parameters:

ident	ConversionTable	reference to the data record which contains the conversion table (see COMPU_TAB).

Description:

Reference to the data record which contains the conversion table (see keyword COMPU_TAB).

Note: COMPU_TAB_REF may only refer to objects of type COMPU_TAB, COMPU_VTAB or COMPU_VTAB_RANGE.

Example:

```
  COMPU_TAB_REF              TEMP_TAB    /*TEMP_TAB: conversion table*/
```

3.5.36 COMPU_VTAB

Prototype:

```
/begin COMPU_VTAB          ident    Name
                           string   LongIdentifier
                           enum     ConversionType
                           uint     NumberValuePairs
                           { float  InVal
                             string OutVal }*
                           [-> DEFAULT_VALUE]
/end COMPU_VTAB
```

Parameters:

ident	Name	unique identifier in the program for the verbal conversion table Note: The name of the conversion table has to be unique within all conversion tables of the ASAM MCD-2MC MODULE, i.e. there must not be another COMPU_TAB, COMPU_VTAB or COMPU_VTAB_RANGE object with the same identifier in the MODULE.
string	LongIdentifier	comment, description
enum	ConversionType	at present only the following types are possible: TAB_VERB verbal conversion table Note: This parameter is a redundant information because the record defined with COMPU_METHOD also contains it.
uint	NumberValuePairs	number of successive value pairs for this conversion table
float	InVal	internal value Note: Datatype “float” is used for the input value of COMPU_TAB and COMPU_VTAB. Since the accepted use case of COMPU_VTAB input values are integers, all float values are rounded to the nearest integer by following the formula (0.5 <= x_raw < 1.5 lead to x = 1). Only float input values used with COMPU_VTAB_RANGE remain float values without truncation If you want to use float as internal values (without truncation), you should use COMPU_VTAB_RANGE.
string	OutVal	description (meaning) of the corresponding byte value

Optional parameters

-> DEFAULT_VALUE:	string used as OutVal for display when the ECU value is out of any declared range. This string shall not be selectable for calibration (when writing to the ECU).

Description:

Conversion table for the visualization of bit patterns

Example:

```
/begin COMPU_VTAB        TT                /* name */
                         "engine status conversion"
                         TAB_VERB          /* convers_type */
                         4                 /* number_value_pairs */
                         0 "engine off"    /* value pairs */
                         1 "idling"
                         2 "partial load"
                         3 "full load"
/end COMPU_VTAB
/begin COMPU_VTAB        CT_DiagStatus
                         ""
                         TAB_VERB          /* convers_type */
                         3                 /* number_value_pairs */
                         0 "C_Fail"
                         1 "C_Pass"
                         2 "C_Indeterminate"
/end COMPU_VTAB
```

3.5.37 COMPU_VTAB_RANGE

Prototype:

```
/begin COMPU_VTAB_RANGE   ident     Name
                          string    LongIdentifier
                          uint      NumberValueTriples
                          { float   InValMin
                            float   InValMax
                            string  OutVal }*
                          [-> DEFAULT_VALUE]
/end COMPU_VTAB_RANGE
```

Parameters:

ident	Name	unique identifier in the program for the verbal range based conversion table Note: The name of the conversion table has to be unique within all conversion tables of the ASAM MCD-2MC MODULE, i.e. there must not be another COMPU_TAB, COMPU_VTAB or COMPU_VTAB_RANGE object with the same identifier in the MODULE.
string	LongIdentifier	comment, description
uint	NumberValueTriples	number of successive value triples for this verbal range based conversion table
float	InValMin	lower limit as float value, needs to be integer ECU value when assigned to “non-float” definitions.
float	InValMax	upper limit as float value, needs to be integer ECU value when assigned to “non-float” definitions.
string	OutVal	display string for the value range

Optional parameters

-> DEFAULT_VALUE	string used as OutVal for display when the ECU value is out of any declared range. This string shall not be selectable for calibration (when writing to the ECU).

Description:

Conversion table for the assignment of display strings to a value range. In particular this is useful for ASAM MCD-2MC definitions with the data type ‘floating point’ (referred as FLOAT definitions).
For FLOAT definitions, the declared string is displayed for InValMin <= ECU value < InValMax, with InValMin, InValMax as floating point values.
For non-FLOAT definions, the declared string is displayed for InValMin <= ECU value <= InValMax, with InValMin, InVal as integer values.

Note: InValMin and InValMax can have the same value to express an assignment of one ECU value to a string (as in COMPU_VTAB); this is not realistic for floating point (and therefore not supported).

Note: Overlapping ranges may not be declared. The ASAM MCD-2MC file is invalid in case of overlapping ranges within COMPU_VTAB_RANGE. But still, the upper limit of one range may be the same FLOAT value than the lower limit of the following range in case of a FLOAT definition (see display rules).

Note: When a COMPU_METHOD with COMPU_VTAB_RANGE is used for calibration (writing of values to ECU), the InValMin is used when the assigned STRING (OutVal) is selected in the user interface.

Note: If the optional DEFAULT_VALUE is declared, this string is displayed when the ECU value is out of any declared range. This string shall not be selectable for calibration.

Example:

```
/begin COMPU_VTAB_RANGE TT        /* name */
                        "engine status conversion"
                        5
                        0 0    "ONE"
                        1 2    "first_section"
                        3 3    "THIRD"
                        4 5    "second_section"
                        6 500  "usual_case"
                        DEFAULT_VALUE "Value_out_of_Range"
/end COMPU_VTAB_RANGE
```

3.5.38 CPU_TYPE

Prototype:

```
CPU_TYPE                     string    CPU
```

Parameters:

string	CPU	CPU identifier

Description:

CPU identification

Example:

```
 CPU_TYPE                    "INTEL 4711"
```

3.5.39 CURVE_AXIS_REF

Prototype:

```
CURVE_AXIS_REF              ident      CurveAxis
```

Parameters:

ident CurveAxis	Name of the CURVE CHARACTERISTIC that is used to normalize or scale the axis that references the curve.

Description:

This keyword is used in conjunction with AXIS_DESCR definitions that use the CURVE_AXIS attribute. It is used to specify the adjustable CURVE CHARACTERISTIC that is used to normalize or scale the axis. See Appendix D for more details.
Note: The same parameters for MaxAxisPoints apply as those for AXIS_DESCR.

Example:

```
/begin CHARACTERISTIC     FUEL_ADJ            /* name */
                          "Air fuel table"    /* long identifier */
                          MAP                 /* type */
                          0x7140              /* address */
                          DEP_12E             /* deposit */
                          1.0                 /* maxdiff */
                          R_MULT              /* conversion */
                          0.0                 /* lower limit */
                          2.0                 /* upper limit */
  /begin AXIS_DESCR       /* description of X-axis points */
                          CURVE_AXIS  /* curve axis points */
                          SPEED       /* reference to input quantity*/
                          NO_COMPU_METHOD     /* conversion */
                          13         /*maximum number of axis points*/
                          0          /*lower limit */
                          12         /*upper limit */
     CURVE_AXIS_REF       SPD_NORM
  /end AXIS_DESCR
  /begin AXIS_DESCR       /* description of Y-axis points */
                          CURVE_AXIS  /* curve axis points */
                          LOAD        /* reference to input quantity*/
                          NO_COMPU_METHOD     /* conversion */
                          17         /*maximum number of axis points*/
                          0          /*lower limit */
                          16         /*upper limit */
     CURVE_AXIS_REF       MAF_NORM
  /end AXIS_DESCR
/end CHARACTERISTIC

/begin RECORD_LAYOUT      DEP_12E
       FNC_VALUES         1 FLOAT32_IEEE ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin CHARACTERISTIC     SPD_NORM /* name */
                          "Speed normalizing function"
```

```
                                        /* long identifier */
                              CURVE     /* type */
                              0x8210    /* address */
                              SPD_DEP   /* deposit */
                              100       /* maxdif */
                              R_NORM    /* conversion */
                              0 6       /* lower limit, upper limit */
     /begin AXIS_DESCR        /* description of X-axis points */
                              STD_AXIS  /* standard axis */
                              SPEED     /* reference to input quantity */
                              R_SPEED   /* conversion */
                              7         /* maximum number of axis points*/
                              0         /* lower limit */
                              10000     /* upper limit */
     /end AXIS_DESCR
   /end CHARACTERISTIC

   /begin RECORD_LAYOUT       SPD_DEP
          AXIS_PTS_X          1 FLOAT32_IEEE INDEX_INCR DIRECT
          FNC_VALUES          2 FLOAT32_IEEE ALTERNATE_WITH_X DIRECT
   /end RECORD_LAYOUT

   /begin CHARACTERISTIC      MAF_NORM  /* name */
                              "Load normalizing function"
                                        /* long identifier */
                              CURVE     /* type */
                              0x8428    /* address */
                              LOAD_DEP  /* deposit */
                              100       /* maxdif */
                              R_NORM    /* conversion */
                              0 16      /* lower limit, upper limit */
     /begin AXIS_DESCR        /* description of X-axis points */
                              STD_AXIS  /* standard axis */
                              LOAD      /* reference to input quantity */
                              R_LOAD    /* conversion */
                              17        /* maximum number of axis points*/
                              0.0       /* lower limit */
                              100.0     /* upper limit */
     /end AXIS_DESCR
   /end CHARACTERISTIC
```

3.5.40 CUSTOMER

Prototype:

```
CUSTOMER                    string   Customer
```

Parameters:

string	Customer	customer name

Description:

This keyword allows a customer name to be specified.

Example:

```
 CUSTOMER                   "LANZ - Landmaschinen"
```

3.5.41 CUSTOMER_NO

Prototype:

```
CUSTOMER_NO                    string    Number
```

Parameters:

string Number	customer number

Description:

Customer number as string.

Example:

```
 CUSTOMER_NO                   "191188"
```

3.5.42 DATA_SIZE

Prototype:

```
DATA_SIZE                     uint      Size
```

Parameters:

uint	Size	data size in bits

Description:

Data size in bits

Example:

```
  DATA_SIZE                    16
```

3.5.43 DEF_CHARACTERISTIC

Prototype:

```
/begin DEF_CHARACTERISTIC { ident  Identifier }
/end DEF_CHARACTERISTIC
```

Parameters:

ident	Identifier	Identifier of those adjustable objects that are defined in respective function.

Description:

This keyword can be used to declare some adjustable objects to be defined in respective function (function orientation).

Note: DEF_CHARACTERISTIC may only refer to objects of type AXIS_PTS or CHARACTERISTIC.

Example:

```
  /begin DEF_CHARACTERISTIC    INJECTION_CURVE
                               DELAY_FACTOR
  /end DEF_CHARACTERISTIC
```

3.5.44 DEFAULT_VALUE

Prototype:

```
DEFAULT_VALUE                 string    display_string
```

Parameters:

string	display_string

Description:

Optional String which can be applied with COMPU_TAB, COMPU_VTAB and COMPU_VTAB_RANGE, used as OutVal for display when the ECU value is out of any declared range. This string shall not be selectable for calibration (when writing to the ECU).
The use of this keyword excludes the use of the keyword DEFAULT_VALUE_NUMERIC.

Example:

```
  DEFAULT_VALUE             "overflow_state"
```

3.5.45 DEFAULT_VALUE_NUMERIC

Prototype:

```
DEFAULT_VALUE_NUMERIC     float     display_value
```

Parameters:

float	display_value

Description:

Optional value which can be applied with COMPU_TAB, used as OutVal for display when the ECU value is out of any declared range. This value is not selectable for calibration (when writing to the ECU). The DEFAULT_VALUE_NUMERIC is handled like a physical value (transferred via ASAM MCD-3 [ASAM MCD-3]). The use of this keyword excludes the use of the keyword DEFAULT_VALUE.

Example:

```
DEFAULT_VALUE_NUMERIC    999.0
```

3.5.46 DEPENDENT_CHARACTERISTIC

Prototype:

```
/begin DEPENDENT_CHARACTERISTIC    string   Formula
                                   (ident   Characteristic)*
/end DEPENDENT_CHARACTERISTIC
```

Parameters:

string	Formula	Formula to be used for the calculation of the physical value of the characteristic from the physical value of other characteristics.
ident	Characteristic	Identifier of those adjustable objects that are used for the calculation of this characteristic.

Description:

This keyword allows dependent characteristics to be specified. For this, other characteristics can be combined into one characteristic whose consistent value is automatically derived by the measurement and calibration system. Upon adjusting one of the characteristics, this characteristic is then also automatically adjusted according to the chosen formula (see also VIRTUAL_CHARACTERISTIC). Consider for example a rectangular triangle with a hypotenuse of length 1,

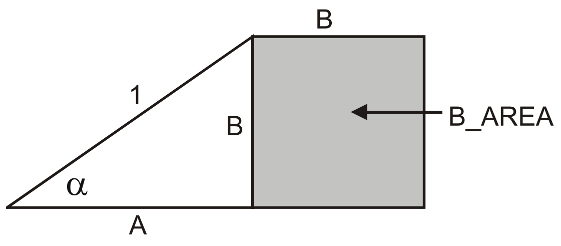


Figure 4 DEPENDENT_CHARACTERISTIC

where the length of the other sides are the characteristics A and B. When adjusting A the characteristic B has to be adjusted accordingly to B = sqrt (1- A*A). The relation between the involved characteristics is described on the physical level. Also other characteristic might depend on B, e.g. B_AREA = B * B. A dependent characteristic should not be adjustable by itself, but only through the adjustment of a characteristic it depends on.
The following example makes clear how the calibration process takes place. Assume for each of the characteristics A, B, and B_AREA a conversion formula of *internal = f(phys) = 100 * phys* and assume that the value A_{int} is 60 (decimal). Then $A_{phys} = A_{int} / 100 = 0.6$. According to the formula B = sqrt (1- A*A), $B_{phys} = 0.8$ and $B_{int} = B_{phys} * 100 = 80$ (decimal). According to B_AREA = B*B, we have $B\_AREA_{phys} = 0.64$ and therefore $B\_AREA_{int} = 64$ (decimal).

The references used in the dependency formula are named X1, X2, X3, … . The reference X1 references the first parameter of the attached parameter list, X2 the second, X3 the third, ….
If there is only one reference used it is allowed to use X instead of X1.

Example:

```
/begin DEPENDENT_CHARACTERISTIC
              „sqrt(1-X1*X1)‟
              A
/end DEPENDENT_CHARACTERISTIC

/* Example for ParamB - ParamA */
/begin DEPENDENT_CHARACTERISTIC
              "X2-X1"
              ParamA    /* is referenced by X1 */
              ParamB    /* is referenced by X2 */
/end DEPENDENT_CHARACTERISTIC
```

3.5.47 DEPOSIT

Prototype:

```
DEPOSIT                        enum      Mode
```

Parameters:

enum Mode	Deposit of the axis points of a characteristic curve or map: ABSOLUTE absolute axis points DIFFERENCE difference axis points

Description:

The axis points of a characteristic can be deposited in two different ways in the memory:
a) The individual axis point values are deposited as absolute values.
b) The individual axis points are deposited as differences. Each axis point value is determined on the basis of the adjacent axis point (predecessor) and the corresponding difference. As reference point for the first axis point <maxvalue> is used:

1-byte-size: <maxvalue> = 2^8 (256)
2-byte-size: <maxvalue> = 2^{16} (65536)
4-byte-size: <maxvalue> = 2^{32}

Example:

```
DEPOSIT                     DIFFERENCE
```

3.5.48 DISCRETE

Prototype:

```
DISCRETE
```

Description:

This keyword indicates that a measure or calibration object has discrete values which should not be interpolated – e.g. in graphic display windows or further calculations. This flag can be used e.g. for integer objects describing states. If the keyword is not specified the values are interpreted as continuous values which can be interpolated. The keyword can be used at MEASUREMENT and CHARACTERISTIC.

Example:

```
/begin MEASUREMENT
   counter
   "…"
   UBYTE
   NO_COMPU_METHOD
   2
   1
   0
   255
   DISCRETE
/end MEASUREMENT
```

3.5.49 DISPLAY_IDENTIFIER

Prototype:

```
DISPLAY_IDENTIFIER         ident     display_name
```

Parameters:

ident	display_name

Description:

This identifier can be used as a alternative name in the Measurement and Calibration System. DISPLAY_IDENTIFIERs can constitute an alternative set of names.

Note: The display_name does not have to be unique and is not referenced elsewhere. But is recommended that the display identifier shall be unique in order to avoid confusion in the user interface of the MCD system.

Example:

```
DISPLAY_IDENTIFIER       load_engine
```

3.5.50 DIST_OP_X / _Y / _Z / _4 / _5

Prototype:

```
DIST_OP_X /_Y /_Z / _4 / _5
                          uint      Position
                          datatype  Datatype
```

Parameters:

uint	Position	Position of the distance operand in the deposit structure.
datatype	Datatype	Data type of the distance operand.

Description:

Description of the distance operand in the deposit structure to compute the axis points for fixed characteristic curves and fixed characteristic maps (see also keyword FIX_AXIS_PAR_DIST). The axis points distribution for fixed characteristic curves or fixed characteristic maps is derived from the two 'offset' and 'distance' parameters as follows:

X_i = Offset + (i - 1)*Distance   i = { 1...numberofaxispts }

or

Y_k = Offset + (k - 1)* Distance   k = { 1...numberofaxispts }

or

Z_m = Offset + (m - 1)* Distance   m = { 1...numberofaxispts }

or

$Z4_n$ = Offset + (n - 1)* Distance   n = { 1...numberofaxispts }

or

$Z5_o$ = Offset + (o - 1)* Distance   o = { 1...numberofaxispts }

Example:

```
DIST_OP_X                  21
                           UWORD
```

3.5.51 ECU

Prototype:

```
ECU                         string    ControlUnit
```

Parameters:

string	ControlUnit	control unit identifier

Description:

String for identification of the control unit.

Example:

```
ECU                         "Steering control"
```

3.5.52 ECU_ADDRESS

Prototype:

```
ECU_ADDRESS                 ulong    Address
```

Parameters:

ulong Address	Address of the measurement in the memory of the control unit.

Description:

ECU_ADDRESS is used to describe the address of a measurement. It should replace the specific IF_DATA. It can be used in MEASUREMENT only.

Example:

```
ECU_ADDRESS                 0x12FE
```

3.5.53 ECU_ADDRESS_EXTENSION

Prototype:

```
ECU_ADDRESS_EXTENSION     int       Extension
```

Parameters:

int	Extension	Address extension of the ECU address

Description:

This keyword is used to specify additional address information. For instance it can be used, to distinguish different address spaces of an ECU (multi-micro controller devices). ECU_ADDRESS_EXTENSION is an optional keyword of MEASUREMENT, AXIS_PTS and CHARACTERISTIC.

Note:

Some calibration interfaces, such as CCP and XCP need an address extension to access ECU data. To avoid the need for additional IF_DATA section at calibration and measurement objects, the keyword ECU_ADDRESS_EXTENSION has been introduced.

Example:

```
/begin MEASUREMENT       N                /* name */
                         "Engine speed"   /* long identifier */
                         UWORD            /* datatype */
                         R_SPEED_3        /* conversion */
                         2                /* resolution */
                         2.5              /* accuracy */
                         120.0            /* lower limit */
                         8400.0           /* upper limit */
  ECU_ADDRESS            0x12345
  ECU_ADDRESS_EXTENSION 1
/end MEASUREMENT

/begin CHARACTERISTIC    MAX_N            /* name */
                         "max speed"      /* long identifier */
                         VALUE            /* type */
                         0x7140           /* address */
                         DAMOS_Word       /* deposit */
                         100.0            /* maxdiff */
                         R_SPEED          /* conversion */
                         0.0              /* lower limit */
                         5000.0           /* upper limit */
  ECU_ADDRESS_EXTENSION 1
/end CHARACTERISTIC
```

3.5.54 ECU_CALIBRATION_OFFSET

Prototype:

```
ECU_CALIBRATION_OFFSET    long     Offset
```

Parameters:

long	Offset	Offset that has to be added to each address of a characteristic

Description:

ECU_CALIBRATION_OFFSET is used to describe a fixed address offset when accessing characteristics in the control unit due to

- near pointers in calibration objects. Some record layouts include near pointers inside a calibration objects from which the calibration system has to compute the absolute values by adding the ECU_CALIBRATION_OFFSET (CDAMOS)
- variant coding. Some ECU projects include multiple data sets for different engine or vehicle projects served by one common ECU. By using the ECU_CALIBRATION_OFFSET, a selection for project base address can be made

Example:

```
ECU_CALIBRATION_OFFSET          0x1000
```

3.5.55 EPK

Prototype:

```
EPK                             string    Identifier
```

Parameters:

string Identifier	EPROM identifier

Description:

EPROM identifier string.

Example:

```
  EPK                           "EPROM identifier test"
```

3.5.56 ERROR_MASK

Prototype:

```
ERROR_MASK                        ulong     Mask
```

Parameters:

ulong	Mask	mask to mask out selected bits

Description:

The ERROR_MASK keyword can be used to mask bits of a MEASUREMENT which indicate that the value is in error. The Measurement and Calibration System may apply this mask to display the error status of a measurement value. The error mask is usually a single bit; separate measurements should be defined in situations where each bit indicates a different type of error.

Example:

```
ERRROR_MASK                   0x00000001
```

3.5.57 EXTENDED_LIMITS

Prototype:

```
EXTENDED_LIMITS           float    LowerLimit
                          float    UpperLimit
```

Parameters:

float	LowerLimit	extended range of table values, lower limit
float	UpperLimit	extended range of table values, upper limit Note: Depending on the type of conversion, the limit values are interpreted as physical or internal values. For conversions of type COMPU_VTAB and COMPU_VTAB_RANGE, the limit values are interpreted as internal values. For all other conversion types, the limit values are interpreted as physical values.

Description:

This keyword can be used to specify an extended range of values. In the measurement and calibration system, for example, when leaving the standard range of values (mandatory parameters 'lower limit' and 'upper limit' in the CHARACTERISTIC data record) a warning could be generated (extended limits enabled only for "power user")

Example:

```
  EXTENDED_LIMITS         0
                          6000.0
```

3.5.58 FIX_AXIS_PAR

Prototype:

```
FIX_AXIS_PAR                    int       Offset
                                int       Shift
                                uint      Numberapo
```

Parameters:

int	Offset	'offset' parameter to calculate the axis points of fixed characteristic curves or maps (see description).
int	Shift	'shift' parameter to calculate the axis points of fixed characteristic curves or maps (see description).
uint	Numberapo	number of axis points

Description:

Typical of fixed characteristic curves and fixed characteristic maps is that, in contrast with standard and group characteristics, the axis points are not deposited individually in the program data of the ECU program but are derived from the two parameters 'offset' and 'shift'. In the current deposit methods both parameters are contained in the description file. In future deposit methods both methods could well be part of the deposit structure of the adjustable objects.
The axis points of fixed characteristic curves or maps are calculated as follows:

X_i = Offset + (i - 1)*2^{Shift}i = { 1...numberapo }

or

Y_k = Offset + (k - 1)*2^{Shift} k = { 1...numberapo }
or

Z_m = Offset + (m - 1)*2^{Shift} m = { 1...numberapo }

Note: This keyword is equivalent to FIX_AXIS_PAR_DIST but differs in parameter ‘Shift’ (see FIX_AXIS_PAR_DIST).

Example:

```
  /* Define axis points 0, 16, 32, 48, 64, 80 */
  FIX_AXIS_PAR               0
                             4
                             6
```

3.5.59 FIX_AXIS_PAR_DIST

Prototype:

```
FIX_AXIS_PAR_DIST           int      Offset
                            int      Distance
                            uint     Numberapo
```

Parameters:

int	Offset	'offset' parameter to calculate the axis points of fixed characteristic curves or maps (see description).
int	Distance	'distance' parameter to calculate the axis points of fixed characteristic curves or maps (see description).
uint	Numberapo	number of axis points

Description:

Typical of fixed characteristic curves and fixed characteristic maps is that, in contrast with standard and group characteristics, the axis points are not deposited individually in the program data of the ECU program but are derived from the two parameters 'offset' and 'distance'. In the current deposit methods both parameters are contained in the description file. In future deposit methods both methods could well be part of the deposit structure of the adjustable objects.
The axis points of fixed characteristic curves or maps are calculated as follows:

X_i = Offset + (i - 1)*Distance   i = { 1...numberapo }

or

Y_k = Offset + (k - 1)*Distance   k = { 1...numberapo }
or

Z_m = Offset + (m - 1)*Distance   m = { 1...numberapo }

Note: This keyword is equivalent to FIX_AXIS_PAR but differs in parameter ‘Distance’.

Example:

```
FIX_AXIS_PAR_DIST        0
                         100
                         8
```

3.5.60 FIX_AXIS_PAR_LIST

Prototype:

```
/begin FIX_AXIS_PAR_LIST
                            { float   AxisPts_Value }*
/end FIX_AXIS_PAR_LIST
```

Parameters:

float	AxisPts_Value	List of "ECU-Original" Values as implied by the ECU algorithm. The number of values must match with the MaxAxisPoints attribute of the AXIS_DESCR referencing FIX_AXIS_PAR_LIST. The COMPU_METHOD assigned to the AXIS_DESCR shall be applied to achieve the actual display values. Note: The data type shall be integer in case of an assignment to a non-float definition).

Description:

Allows the description of any value combination of a virtual axis (FIX_AXIS, axis points not in the ECU memory). Other methods (FIX_AXIS_PAR, FIX_AXIS_PAR_DIST) implicitly assume an interpolation algorithm in the ECU. But axis descriptions are also used e.g. to span status tables.
The values are the input for the COMPU_METHOD assigned to the axis. Even a verbal table could be applied as COMPU_METHOD (i.e. for the axis description of status tables on which no interpolation is applied).

Example:

```
/begin FIX_AXIS_PAR_LIST        2 5 9
/end FIX_AXIS_PAR_LIST
```

3.5.61 FIX_NO_AXIS_PTS_X / _Y / _Z / _4 / _5

Prototype:

```
FIX_NO_AXIS_PTS_X /_Y /_Z / _4 / _5
                                    uint      NumberOfAxisPoints
```

Parameters:

uint	NumberOfAxisPoints	Dimensioning of characteristic curves or characteristic maps with a fixed number of axis points

Description:

This keyword indicates that all characteristics of type CURVE, MAP, CUBOID, CUBE_4 or CUBE_5 allocate a fixed number of axis points. In a RECORD_LAYOUT data record, this keyword cannot be used simultaneously with the keyword NO_AXIS_PTS_X / _Y / _Z / _4 / _5

Example:

```
FIX_NO_AXIS_PTS_X        17
```

3.5.62 FNC_VALUES

Prototype:

```
FNC_VALUES                    uint      Position
                              datatype  Datatype
                              enum      IndexMode
                              addrtype  Addresstype
```

Parameters:

uint	Position	position of table values (function values) in the deposit structure (description of sequence of elements in the data record).
datatype	DataType:	data type of the table values
enum	IndexMode:	for characteristic maps, this attribute is used to describe how the 2-dimensional table values are mapped onto the 1-dimensional address space:

ALTERNATE_CURVES

curves:	curves which share a common axis are deposited in columns; each row of memory contains values for all the shared axis curves at a given axis breakpoint. Required in order to represent characteristics which correspond to arrays of structures in ECU program code. In the example code below, DT10, DT20, etc are treated as separate curves which may have different conversions or limits:-
maps:	alternate curves not supported
cuboid:	alternate curves not supported
cube_4:	alternate curves not supported
cube_5:	alternate curves not supported

```
typedef struct    {
        int DT10;
        int DT20;
        int DT30;
        int DT40;
    } VXP_TYPE;

const VXP_TYPE VX_PLUS_DELAY_TIMES[5] = {
        { 10, 3, 4, 8 },
        { 12, 2, 4, 6 },
        { 17, 9, 5, 8 },
        { 10, 1, 4, 8 },
        { 18, 3, 8, 8 },
    };
```

ALTERNATE_WITH_X

maps:	deposited in columns, the columns of table values alternate with the respective X-coordinates.

			curves: table values and X-coordinate values are deposited alternating.
			cuboid: alternate with X not supported
			cube_4: alternate with X not supported
			cube_5: alternate with X not supported
		ALTERNATE_WITH_Y	
			maps: deposited in rows, the rows of table values alternate with the respective Y-coordinates (maps only).
			cuboid: alternate with Y not supported
			cube_4: alternate with Y not supported
			cube_5: alternate with Y not supported
		COLUMN_DIR	deposited in columns
		ROW_DIR	deposited in rows
addrtype	Addresstype	addressing of the table values (see enum addrtype).	

Description:

Description of the table values (function values) of an adjustable object. If the ALTERNATE option is used, the position parameter of values and axis-points indicates their order. The concepts 'columns' and 'rows' relate to the XY coordinate system (see also Appendix C).
For characteristic cuboids each XY plane is mapped as above. The cuboid is stored as an array of maps with incremented or decremented Z coordinates. The CUBE_4 is stored as an array of CUBOID with incremented or decremented Z2 coordinates. The CUBE_5 is stored as an array of CUBE_4 with incremented or decremented Z3 coordinates. (Alternate not supported)

Example for ROW_DIR:

A 2 x 3 matrix M = $\begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \end{bmatrix}$
would be stored as follows: a_{11}, a_{12}, a_{13}, a_{21},a_{22}, a_{23}

More generally, a matrix a $i\,x\,j\,x\,k$
would be listed as a_{111}, … a_{11k}, a_{121}, … a_{1jk}, a_{211}, … a_{ijk}

Example for COLUMN_DIR:

A 2 x 3 matrix M = $\begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \end{bmatrix}$
would be stored as follows: a_{11}, a_{21}, a_{12}, a_{22}, a_{13}, a_{23}

More generally, a matrix a $i\,x\,j\,x\,k$
would be listed as a_{111}, … a_{i11}, a_{121}, … a_{ij1}, a_{112}, … a_{ijk}

Example:

```
FNC_VALUES                  7
                            SWORD
                            COLUMN_DIR
                            DIRECT
```

3.5.63 FORMAT

Prototype:

```
FORMAT                      string   FormatString
```

Parameters:

string FormatString	display format in %[length].[layout]; length indicates the overall length; layout indicates the decimal places

Description:

This keyword allows a special display format to be specified for some MEASUREMENT, CHARACTERISTIC or AXIS_PTS object. If exists this display format is used instead of display format specified in respective COMPU_METHOD data record. The format string should never be empty as "".

Example:

```
  FORMAT                      "%4.2"
```

3.5.64 FORMULA

Prototype:

```
/begin FORMULA              string    f(x)
                            [-> FORMULA_INV]
/end FORMULA
```

Parameters:

string f(x)

function to calculate the physical value from the control unit internal value. The interpretation proceeds from left to right. Operator preferences, such as power before product/quotient before sum/difference, are taken into account. Brackets are allowed.

System constants which are defined in SYSTEM_CONSTANT can be used here, if their names comply to the restrictions of data type ident. For that the name of the system constant has to be put in "sysc()".The value of a system constant used in FORMULA must contain either a numerical value or a string that contains a further FORMULA part (recursive replacement of the text in the formula). Endless loop for the system constant usage is not allowed.

The notation of operators and function names is conform to the ANSI C notation.
The following operation symbols can be used:

Basic operations:

+	for sums
-	for differences
*	for products
/	for quotients

Binary operators: interpretation from left to right

&	bitwise AND
|	bitwise OR
>>	bitwise shift right
<<	bitwise shift left

^ bitwise exclusive OR
~bitwise NOT

Logical operators: interpretation from left to right

&&	logical AND
||	logical OR
!	logical NOT

Trigonometric functions:
sin(x), cos(x), tan(x)
asin(x), acos (x), atan (x)
sinh(x), cosh(x), tanh(x)

Exponential function:
exp(x) for base e

Logarithmic functions:
log(x) for base e
log10(x) for base 10

Square root, absolute amount and power:
sqrt(x)
abs(x)
pow(x1, x2)

Optional parameters:

-> FORMULA_INV	function to calculate the control unit internal value from the physical value. This parameter is mandatory in formulas used for the conversion of adjustable objects. It is optional only for measurement objects. Note: Certain functions in the measurement and calibration system can only be used for those measurement objects for which this parameter is specified (e.g. scalable DAC output, triggering).

Description:

This keyword allows any kind of formula to be specified for the conversion of measurement values, axis points or table values of an adjustable object from their ECU internal format into the physical format. The interpretation of the formula must be supported by a formula interpreter in the operating system.

Note: The references used in the formula are named X1, X2, X3, … . The reference X1 references the first input, X2 the second, X3 the third, ….
If there is only one reference used it is allowed to use X instead of X1.

Example 1:

```
/begin FORMULA          "sqrt( 3 - 4*sin(X1) )"
/end FORMULA
```

Example 2:

```
/* Example to explain reference to SYSTEM_CONSTANT */
SYSTEM_CONSTANT "PI" "3.1415"
SYSTEM_CONSTANT "PI_half" "$(PI) / 2"
 […]
/begin FORMULA      "$(PI_half) * X"
/end FORMULA
```

Note: Do not use FORMULA to describe identical, linear, or rational functions.

Note: Some of the FORMULA operators are no longer downward compatible to earlier ASAM MCD-2MC versions! For details see chapter 1.4.1.

3.5.65 FORMULA_INV

Prototype:

```
FORMULA_INV                     string   g(x)
```

Parameters:

string g(x)	function for calculation of the control unit internal value from the physical value. The interpretation proceeds from left to right. Operator preferences, such as power before product/quotient before sum/difference, are taken into account. Brackets are allowed. Permissible operation symbols: see keyword FORMULA. System constants which are defined in SYSTEM_CONSTANT can be used here, if their names comply to the restrictions of data type ident. The value of a system constant used in FORMULA must contain either a numerical value or a string that contains a further FORMULA part (recursive replacement of the text in the formula). Endless loop for the system constant usage is not allowed.

Description:

This keyword allows any kind of formula to be specified for the conversion of measurement values, axis points or table values of an adjustable object from their physical format into the ECU internal format. The interpretation of the formula must be supported by a formula interpreter in the operating system.

Note: FORMULA_INV is necessary if used for CHARACTERISTIC objects. Only MEASUREMENT objects do not need an inverse formula.

Example:

Inversion function e.g. for keyword FORMULA

```
 FORMULA_INV              "asin( sqrt( (3 - X1)/4 ) )"
```

3.5.66 FRAME

Prototype:

```
/begin FRAME                ident     Name
                            string    LongIdentifier
                            uint      ScalingUnit
                            ulong     Rate
                            [-> FRAME_MEASUREMENT]
                            {-> IF_DATA}*

/end FRAME
```

Parameters:

ident	Name	unique identifier in the program, referencing is based on this 'name' Note: The name of the frame has to be unique within all frames of the ASAM MCD-2MC MODULE, i.e. there must not be another FRAME object with the same identifier in the MODULE.
string	LongIdentifier	comment, description
uint	ScalingUnit	This parameter defines the basic scaling unit. The following parameter 'Rate' relates on this scaling unit. The value of ScalingUnit is coded as shown in Table 10 Codes for scaling units (CSE)
ulong	Rate	The maximum refresh rate of the concerning measurement source in the control unit. The unit is defined with parameter 'ScalingUnit'.

Optional parameters:

-> FRAME_MEASUREMENT	Use this keyword to define the frames measurement objects.
-> IF_DATA	Data record to describe interface specific data of the frame. The parameters associated with this keyword have to be described in the ASAM MCD-2MC metalanguage.

Description:

For the structuring of a car network involving a very large number of measuring channels, function frames can be defined.
These function frames shall be used in the measurement and calibration system to allow the selection lists for the selection of measuring channels to be represented in a structured manner on the basis of functional viewpoints (function orientation).
This will also be used to describe the packaging of measurement data into sources for CAN frames in a network environment.

Example:

```
/begin FRAME    ABS_ADJUSTM
                          "function group ABS adjustment"
                          3
                          2       /* 2 msec. */
   FRAME_MEASUREMENT      LOOP_COUNTER TEMPORARY_1
/end FRAME
```

3.5.67 FRAME_MEASUREMENT

Prototype:

```
FRAME_MEASUREMENT          { ident   Identifier }*
```

Parameters:

ident	Identifier	Identifier of quantity of respective FRAME (reference to measurement object).

Description:

This keyword can be used to define quantities of respective FRAME.

Example:

```
FRAME_MEASUREMENT          WHEEL_REVOLUTIONS
                           ENGINE_SPEED
```

3.5.68 FUNCTION

Prototype:

```
/begin FUNCTION          ident    Name
                         string   LongIdentifier
                         {-> ANNOTATION}*
                         [-> DEF_CHARACTERISTIC]
                         [-> FUNCTION_VERSION]
                         {-> IF_DATA}*
                         [-> IN_MEASUREMENT]
                         [-> LOC_MEASUREMENT]
                         [-> OUT_MEASUREMENT]
                         [-> REF_CHARACTERISTIC]
                         [-> SUB_FUNCTION]
/end FUNCTION
```

Parameters:

ident Name	unique Identifier in the program, referencing is based on this 'name' Note: The name of the function has to be unique within all functions of the ASAM MCD-2MC MODULE, i.e. there must not be another FUNCTION object with the same identifier in the MODULE.
string LongIdentifier	comment, description

Optional parameters:

-> ANNOTATION	Set of notes (represented as multi-line ASCII description texts) which are related. Can serve e.g. as application note.
-> DEF_CHARACTERISTIC	This keyword can be used to define those adjustable objects which are defined in respective function.
-> FUNCTION_VERSION	String to define the version of the function. An measurement and calibration tool should be able to handle function oriented characteristic data.
-> IF_DATA	Data record to describe interface specific data of the function. The parameters associated with this keyword have to be described in the ASAM MCD-2MC
-> IN_MEASUREMENT	Use this keyword to define the input measurement objects of respective function (input variables).
-> LOC_MEASUREMENT	Use this keyword to define the local measurement objects of respective function (local variables: scope is limited to this function).
-> OUT_MEASUREMENT	Use this keyword to define the output measurement objects of respective function (output variables).
-> REF_CHARACTERISTIC	If the function contains references to some adjustable objects, this keyword can be used to describe this references.
-> SUB_FUNCTION	This keyword can be used to describe the function hierarchy. If the respective function is subdivided into subfunctions, use this keyword to define the subfunctions.

Description:

For the structuring of projects involving a very large number of adjustable objects and measuring channels, functions can be defined. These functions shall be used in the measurement and calibration system to allow the selection lists for the selection of adjustable objects and measuring channels to be represented in a structured, hierarchical manner following the order of input on the basis of functional viewpoints (function orientation).

Note: Since ASAP2 version 1.20 the references between functions and measurement objects resp. adjustable objects can be described either with keyword CHARACTERISTIC, AXIS_PTS and MEASUREMENT (see FUNCTION_LIST) or with keyword FUNCTION.

Example:

```
/begin FUNCTION             ID_ADJUSTM          /* name */
                            "function group idling adjustment"
  /begin DEF_CHARACTERISTIC INJECTION_CURVE
  /end DEF_CHARACTERISTIC
  /begin REF_CHARACTERISTIC FACTOR_1
  /end REF_CHARACTERISTIC
  /begin IN_MEASUREMENT     WHEEL_REVOLUTIONS  ENGINE_SPEED
  /end IN_MEASUREMENT
  /begin OUT_MEASUREMENT    OK_FLAG  SENSOR_FLAG
  /end OUT_MEASUREMENT
  /begin LOC_MEASUREMENT    LOOP_COUNTER  TEMPORARY_1
  /end LOC_MEASUREMENT
  /begin SUB_FUNCTION       ID_ADJUSTM_SUB
  /end SUB_FUNCTION
/end FUNCTION
```

3.5.69 FUNCTION_LIST

Prototype:

```
/begin FUNCTION_LIST       ( ident   Name)
/end FUNCTION_LIST
```

Parameters:

ident	Name	list of references to higher-order functions (see FUNCTION)

Description:

This keyword can be used to specify a list of 'functions' to which the relevant adjustable object has been allocated (function orientation).

Note: Since ASAP2 version 1.20 the keyword FUNCTION comprises some additional features to describe functional structure and dependencies.

Example:

```
  /begin FUNCTION_LIST    ID_ADJUSTM
                          FL_ADJUSTM
                          SPEED_LIM
  /end FUNCTION_LIST
```

3.5.70 FUNCTION_VERSION

Prototype:

```
FUNCTION_VERSION          string    VersionIdentifier
```

Parameters:

string	VersionIdentifier	short identifier for the version

Description:

String for identification of the version of a function with maximum MAX_STRING characters.

Example:

```
  FUNCTION_VERSION         "BG5.0815"
```

3.5.71 GROUP

Prototype:

```
/begin GROUP              ident    GroupName
                          string   GroupLongIdentifier
                          {-> ANNOTATION}*
                          [-> FUNCTION_LIST]
                          {-> IF_DATA}*
                          [-> REF_CHARACTERISTIC]
                          [-> REF_MEASUREMENT]
                          [-> ROOT]
                          [-> SUB_GROUP]
/end GROUP
```

Parameters:

ident	GroupName	unique identifier of the group Note: The name of the group has to be unique within all groups of the ASAM MCD-2MC MODULE, i.e. there must not be another GROUP object with the same identifier in the MODULE.
string	GroupLongIdentifier	Comment, description of the group within a grouping mechanism.

Optional parameters:

-> ANNOTATION	Set of notes (represented as multi-line ASCII description texts) which are related. Can serve e.g. as application note.
-> FUNCTION_LIST	This keyword can be used to specify a list of references to functions.
-> IF_DATA	Data record to describe interface specific data of the group. The parameters associated with this keyword have to be described in the ASAM MCD-2MC
-> REF_CHARACTERISTIC	If the group contains references to some adjustable objects, this keyword can be used to describe these references.
-> REF_MEASUREMENT	If the group contains references to some measurement objects, this keyword can be used to describe these references.
-> ROOT	This keyword indicates that the group constitutes an independent grouping mechanism (root level) which the MCD system may use as a root point for the hierarchical presentation of groups. All groups referenced via SUB_GROUP (including nested references) constitute a set of groups belonging to the grouping mechanism. Examples for such grouping mechanisms : Group Name = {Software_Components, Calibration_Components, Editor_Selection_Lists}
-> SUB_GROUP	This keyword can be used to describe the group hierarchy. If the respective group is subdivided into sub-groups, use this keyword to define the subgroups. In particular, SUB_GROUP references the groups

belonging to a grouping mechanism indicated with the optional keyword ROOT (see above).

Description:

These GROUPs shall be used in the measurement and calibration system to provide selection lists (groups) of adjustable objects and measuring channels.
For the structuring of projects involving a very large number of adjustable objects and measuring channels, an unlimited number of grouping mechanisms, each constituted from a root group containing subgroups (including nested references)**,** can be defined. Such root groups are used in the MCD system for initial display of the available groups, as the root of a tree containing the referenced subgroups. Use cases are e.g. software components which define the C file assignment, calibration components which describe the calibration engineer's viewpoint, editor selection lists which can define the presentation of calibration objects and their related measurement quantities.

Example:

```
/begin GROUP            SOFTWARE_COMPONENTS
                        "assignment of the definitions to C files"
                        ROOT
  /begin SUB_GROUP      INJE
                        C6TD
  /end SUB_GROUP
/end GROUP

/begin GROUP            INJE
                        “Subsystem Injection”
  /begin SUB_GROUP      injec1
                        injec2
  /end SUB_GROUP
/end GROUP

/begin GROUP            Injec1
                        “Module filename Injec1”
  /begin REF_CHARACTERISTIC
                        INJECTION_CURVE
  /end REF_CHARACTERISTIC
  /begin REF_MEASUREMENT
                        LOOP_COUNTER
                        TEMPORARY_1
  /end REF_MEASUREMENT
/end GROUP

/begin GROUP            Injec2
                        “Module filename Injec2”
  /begin REF_CHARACTERISTIC
                        INJECTION_ADJUST
  /end REF_CHARACTERISTIC
  /begin REF_MEASUREMENT
                        GAS_INPUT
                        WHEEL_SPEED
  /end REF_MEASUREMENT
/end GROUP


/begin GROUP            C6TD
```

```
                              "Shift Point Control"
   /begin SUB_GROUP           c6tdvder
                              c6tdertf
   /end  SUB_GROUP
/end GROUP

/begin GROUP                  c6tdvder
                              "Module filename c6tdvder"
   /begin  REF_CHARACTERISTIC
                              SHIFT23_CURVE
   /end  REF_CHARACTERISTIC
   /begin REF_MEASUREMENT
                              LOOP_COUN2
                              NO_GEAR
   /end REF_MEASUREMENT
/end GROUP

/begin GROUP                  c6tderft
                              "Module filename c6tderft"
   /begin REF_CHARACTERISTIC
                              LUP23_CURVE
   /end  REF_CHARACTERISTIC
   /begin REF_MEASUREMENT
                              TRANSMISSION_SP
                              ENGINE_SPEED
   /end REF_MEASUREMENT
/end GROUP


/begin GROUP                  CALIBRATION_COMPONENTS
                              "assignment of the definitions to
                              calibration components"
                              ROOT
   /begin SUB_GROUP
                              Winter_Test
                              Summer_Test
   /end SUB_GROUP
/end GROUP

/begin GROUP                  CALIBRATION_COMPONENTS_L4
                              "L4-PCM 2002 cals"
                              ROOT
   /begin SUB_GROUP           LUFT
                              CLOSED_LOOP
   /end SUB_GROUP
/end GROUP

/begin GROUP                  LUFT
                              "Cals in LUFT Subsystem"
   /begin REF_CHARACTERISTIC
                              KfLUFT_n_EngSpdThrsh
                              KtLUFT_ScaledVE
                              KaLUFT_AirPerCylCoeff
   /end  REF_CHARACTERISTIC
/end GROUP

/begin GROUP                  CLOSED_LOOP
```

```
                          "Cals in FCLS, FCLP & FCLL Subsystem"
   /begin REF_CHARACTERISTIC
                          KaFCLP_U_O2LeanThrsh
                          KfFCLP_t_O2AgainstMax
   /end REF_CHARACTERISTIC
 /end GROUP

 /begin GROUP             Winter_Test
                          "Flash this in winter time"
   /begin REF_CHARACTERISTIC
                          GASOLINE_CURVE
   /end  REF_CHARACTERISTIC
 /end GROUP

 /begin GROUP             Summer_Test
                          "Flash that in summer time"
   /begin REF_CHARACTERISTIC
                          SUPER_CURVE
   /end  REF_CHARACTERISTIC
 /end GROUP

 /begin GROUP             SOFTWARE_COMPONENTS
                          " L4-PCM 2002 C modules"
                          ROOT
   /begin SUB_GROUP
                          luftkmgr.c
                          fclpkout.c
                          viosmeng.c
   /end SUB_GROUP
 /end GROUP

 /begin GROUP             luftkmgr.c
                          "Objects in luftkmgr.c"
   /begin REF_CHARACTERISTIC
                          KtLUFT_ScaledVE
   /end REF_CHARACTERISTIC
 /end GROUP

 /begin GROUP             fclpkout.c
                          "Objects in fclpkout.c"
   /begin REF_CHARACTERISTIC
     KaFCLP_U_O2LeanThrsh
                            KfFCLP_t_O2AgainstMax
   /end REF_CHARACTERISTIC
 /end GROUP

 /begin GROUP             viosmeng.c
                          "Objects in viosmeng.c"
   /begin REF_CHARACTERISTIC
                          VfVIOS_n_EngSpdLORES
                          VfVIOS_p_AmbientAirPres
   /end REF_CHARACTERISTIC
 /end GROUP
```

3.5.72 GUARD_RAILS

Prototype:

```
GUARD_RAILS
```

Description:

This keyword is used to indicate that an adjustable CURVE, MAP or AXIS_PTS uses guard rails. The Measurement and Calibration System does not allow the user to edit the outermost values or axis points of the adjustable object, but calculates them as follows:

Table 9 GUARD_RAILS

AXIS_PTS	CURVE	MAP
(X_0) = AXIS_PTS.LowerLimit	$(X_0) = (X_1)$	$(X_i, Y_0) = (X_j, Y_1)$
(X_m) = AXIS_PTS.UpperLimit	$(X_m) = (X_{m-1})$	$(X_i, Y_n) = (X_j, Y_{n-1})$
		$(X_0, Y_j) = (X_1, Y_j)$
		$(X_m, Y_j) = (X_{m-1}, Y_j)$

$0 < i < m$, m = Number of X-axis points
$0 < j < n$, n = Number of Y-axis points

Example:

```
/begin CHARACTERISTIC   F_INJ_CORR  /* name */
                        "Injector correction factor''
                                    /* long identifier */
                        CURVE       /* type */
                        0x7140      /* address */
                        REC12       /* deposit */
                        10.0        /* maxdiff */
                        C_INJF      /* conversion */
                        0.0         /* lower limit */
                        199.0       /* upper limit */
                        GUARD_RAILS /* uses guard rails */
  /begin AXIS_DESCR     /* description of X-axis points */
                        STD_AXIS    /* standard axis points */
                        N           /* reference to input quantity*/
                        C_TEMP      /* conversion */
                        10     /* maximum number of axis points*/
                        -40.0       /* lower limit */
                        150.0       /* upper limit */
  /end AXIS_DESCR
/end CHARACTERISTIC
```

3.5.73 HEADER

Prototype:

```
/begin HEADER                string    Comment
                             [-> PROJECT_NO]
                             [-> VERSION]
/end HEADER
```

Parameters:

string Comment:	comment, description

Optional parameters:

-> VERSION	version number
-> PROJECT_NO	project number

Description:

Header information on a project. A project can comprise several ECU's or devices.

Example:

```
  /begin HEADER            "see also specification XYZ of 01.02.1994"
        VERSION            "BG5.0815"
        PROJECT_NO         M4711Z1
  /end HEADER
```

3.5.74 IDENTIFICATION

Prototype:

```
IDENTIFICATION              uint      Position
                            datatype  Datatype
```

Parameters:

uint	Position	position of the 'identifier' in the deposit structure.
datatype	Datatype	word length of the 'identifier"

Description:

Description of an 'identifier' in an adjustable object.

Example:

```
 IDENTIFICATION             1
                            UWORD
```

3.5.75 IF_DATA (EXAMPLE)

Prototype:

```
/begin IF_DATA            ident        Name
                          [-> ...]
/end IF_DATA
```

Parameters:

ident Name	identifier of Interface, The prefix "ASAP1B_" is reserved for ASAM and can be not used for proprietary Interfaces.

Optional parameters:

-> ...	Data record to describe interface specific data. The parameters associated with this keyword have to be described in the ASAM MCD-2MC metalanguage. These parameters describe e.g. the access methods to the measurement data collection, serial communication and so on.

Description:

Definition of interface-specific description data.

Example:

```
/begin IF_DATA
   ASAP1B_EXAMPLE   /* Name of device */
             /* interface-specific parameters described in A2ML */
   /begin DP_BLOB
             0x12129977
             0xFF
   /end DP_BLOB
             /* interface-specific parameters described in A2ML */
   /begin PA_BLOB
             "Pumpenkennfeld"
             1
             2
             17
   /end PA_BLOB
/end IF_DATA
```

3.5.76 IN_MEASUREMENT

Prototype:

```
/begin IN_MEASUREMENT     { ident   Identifier }*
/end IN_MEASUREMENT
```

Parameters:

ident	Identifier	Identifier of input quantity of respective function (reference to measurement object).

Description:

This keyword can be used to define input quantities of respective function.
Note: IN_MEASUREMENT may only refer to objects of type MEASUREMENT.

Example:

```
/begin IN_MEASUREMENT    WHEEL_REVOLUTIONS
                         ENGINE_SPEED
/end IN_MEASUREMENT
```

3.5.77 LAYOUT

Prototype:

```
LAYOUT                         enum      IndexMode
```

Parameters:

enum IndexMode:	For multi-dimensional measurement arrays, this attribute is used to describe how the array values are mapped onto the one-dimensional address space: ROW_DIR: The array is deposited in rows. COLUMN_DIR: The array is deposited in columns. For an example, see FNC_VALUES**.**

Description:

This keyword describes the layout of a multi-dimensional measurement array. It can be used at MEASUREMENT.

3.5.78 LEFT_SHIFT

Prototype:

```
LEFT_SHIFT                    ulong     Bitcount
```

Parameters:

ulong	Bitcount	Shift ‚Bitcount' bits to the left

Description:

The LEFT_SHIFT keyword is only used within the BIT_OPERATION keyword. See description of BIT_OPERATION.

3.5.79 LOC_MEASUREMENT

Prototype:

```
/begin LOC_MEASUREMENT    { ident   Identifier }*
/end LOC_MEASUREMENT
```

Parameters:

ident	Identifier	Identifier of local quantity of respective function (reference to measurement object).

Description:

This keyword can be used to define local quantities of respective function.
Note: LOC_MEASUREMENT may only refer to objects of type MEASUREMENT.

Example:

```
  /begin LOC_MEASUREMENT   LOOP_COUNTER
                           TEMPORARY_1
  /end LOC_MEASUREMENT
```

3.5.80 MAP_LIST

Prototype:

```
/begin MAP_LIST             { ident   Name }*
/end MAP_LIST
```

Parameters:

ident Name	identifier of a MAP (see CHARACTERISTIC)

Description:

This keyword can be used to specify the list of MAPs which comprise a CUBOID. This keyword is required because CUBOID data will not be at contiguous memory locations if a CUBOID is composed of several MAPs. If MAP_LIST is used at CHARACTERISTIC, only one AXIS_DESCR is accepted. This AXIS_DESCR describes the Z-axis. (The X- and Y-axes are described at the MAPs referenced by MAP_LIST.

3.5.81 MATRIX_DIM

Prototype:

```
MATRIX_DIM                    uint      xDim
                              uint      yDim
                              uint      zDim
```

Parameters:

uint	xDim	number of values in dimension of x
uint	yDim	number of values in dimension of y
uint	zDim	number of values in dimension of z

Description:

This keyword is used to describe the dimensions of a multidimensional array of values (MEASUREMENT or CHARACTERISTIC).
xDim * yDim *zDim = number of values.
If NUMBER or ARRAY_SIZE are used in the CHARACTERISTIC or MEASUREMENT record the result must be the same as the value given at this option.

Example:

```
 MATRIX_DIM                  2
                             4
                             3
```

3.5.82 MAX_GRAD

Prototype:

```
MAX_GRAD                    float     MaxGradient
```

Parameters:

float	MaxGradient	maximum permissible gradient

Description:

This keyword is used to specify a maximum permissible gradient for an adjustable object in relation to an axis:

MaxGrad_x = maximum(absolut($(W_{i,k} - W_{i-1,k})/(X_i - X_{i-1})$))

MaxGrad_y = maximum(absolut($(W_{i,k} - W_{i,k-1})/(Y_i - Y_{k-1})$))

Example:

```
  MAX_GRAD                    200.0
```

3.5.83 MAX_REFRESH

Prototype:

```
MAX_REFRESH                  uint      ScalingUnit
                             ulong     Rate
```

Parameters:

uint	ScalingUnit	this parameter defines the basic scaling unit. The following parameter 'Rate' relates on this scaling unit. The value of ScalingUnit is coded as shown below in 'Table Codes for scaling (CSE)'.
ulong	Rate	the maximum refresh rate of the concerning measurement object in the control unit. The unit is defined with parameter 'ScalingUnit'

Description:

This optional keyword can be used to specify the maximum refresh rate in the control unit.

Example:

```
MAX_REFRESH 3
            15  /* ScalingUnit = 1 msec --> refresh rate = 15 msec */
MAX_REFRESH 998
            2   /* ScalingUnit = 998 --> Every second frame */
```

Table 10 Codes for scaling units (CSE)

Code	Unit	Referred to	Comment
0	1 µsec	Time	
1	10 µsec	Time	
2	100 µsec	Time	
3	1 msec	Time	
4	10 msec	Time	
5	100 msec	Time	
6	1 sec	Time	
7	10 sec	Time	
8	1 min	Time	
9	1 hour	Time	
10	1 day	Time	
100	Angular degrees	Angle	
101	Revolutions 360 degrees	Angle	
102	Cycle 720 degrees	Angle	e.g. in case of IC engines
103	Cylinder segment	Combustion	e.g. in case of IC engines
998	When frame available	Event	Source defined in keyword Frame

Code	Unit	Referred to	Comment
999	Always if there is new value		Calculation of a new upper range limit after receiving a new partial value, e.g. when calculating a complex trigger condition
1000	Non deterministic		Without fixed scaling

3.5.84 MEASUREMENT

Prototype:

```
/begin MEASUREMENT         ident    Name
                           string   LongIdentifier
                           datatype Datatype
                           ident    Conversion
                           uint     Resolution
                           float    Accuracy
                           float    LowerLimit
                           float    UpperLimit
                           {-> ANNOTATION}*
                           [-> ARRAY_SIZE]
                           [-> BIT_MASK]
                           [-> BIT_OPERATION]
                           [-> BYTE_ORDER]
                           [-> DISCRETE]
                           [-> DISPLAY_IDENTIFIER]
                           [-> ECU_ADDRESS]
                           [-> ECU_ADDRESS_EXTENSION]
                           [-> ERROR_MASK]
                           [-> FORMAT]
                           [-> FUNCTION_LIST]
                           {-> IF_DATA}*
                           [-> LAYOUT]
                           [-> MATRIX_DIM]
                           [-> MAX_REFRESH]
                           [-> PHYS_UNIT]
                           [-> READ_WRITE]
                           [-> REF_MEMORY_SEGMENT]
                           [-> SYMBOL_LINK]
                           [-> VIRTUAL]
/end MEASUREMENT
```

Parameters:

ident	Name	unique identifier in the ECU program Note: The name of the measurement object has to be unique within all measurement objects and adjustable objects of the ASAM MCD-2MC MODULE, i.e. there must not be another AXIS_PTS, CHARACTERISTIC or MEASUREMENT object with the same identifier in the MODULE.
string	LongIdentifier	comment, description
datatype	Datatype	data type of the measurement
ident	Conversion	Reference to the relevant record of the description of the conversion method (see COMPU_METHOD). If there is no conversion method, as in the case of CURVE_AXIS, the parameter ‘Conversion’ should be set to “NO_COMPU_METHOD" (measurement and calibration systems must be able to handle this case).
uint	Resolution	smallest possible change in bits
float	Accuracy	possible variation from exact value in %
float	LowerLimit	plausible range of table values, lower limit
float	UpperLimit	plausible range of table values, upper limit

Note: Depending on the type of conversion, the limit values are interpreted as physical or internal values.
For conversions of type COMPU_VTAB and COMPU_VTAB_RANGE, the limit values are interpreted as internal values. For all other conversion types, the limit values are interpreted as physical values.

Optional parameters:

-> ANNOTATION	Set of notes (represented as multi-line ASCII description texts) which are related. Can serve e.g. as application note.
-> ARRAY_SIZE	This keyword marks a measurement object as an array of measurement values.
-> BIT_MASK	With deviation from the standard value 0xFFFFFFFF this parameter can be used to mask out bits.
-> BIT_OPERATION	The BIT_OPERATION keyword can be used to perform operation on the masked out value.
-> BYTE_ORDER	With deviation from the standard value this parameter can be used to specify the byte order (Intel format, Motorola format)
-> DISCRETE	This keyword indicates that the measurement values are discrete values which should not be interpolated – e.g. in graphic display windows or further calculations. This flag can be used e.g. for integer objects describing states. If the keyword is not specified the values are interpreted as continuous values which can be interpolated.
-> DISPLAY_IDENTIFIER	Can be used as a display name (alternative to the 'name' attribute).
-> ECU_ADDRESS	Address of the measurement in the memory of the control unit.
-> ECU_ADDRESS_EXTENSION	This keyword is an additional address information. For instance it can be used, to distinguish different address spaces of an ECU (multi-micro controller devices).
-> ERROR_MASK	With deviation from the standard value 0x00000000 this parameter can be used to mask bits of a MEASUREMENT which indicate that the value is in error.
-> FORMAT	With deviation from the display format specified with keyword COMPU_TAB referenced by parameter <Conversion> a special display format can be specified to be used to display the measurement values.
-> FUNCTION_LIST	This keyword can be used to specify a list of 'functions' to which this measurement object has been allocated.

Note: Since ASAP2 version 1.20 the keyword FUNCTION comprises some additional features to describe functional structure and dependencies. -> IF_DATA Data

	record to describe interface specific data of the measurement. The parameters associated with this keyword have to be described in the ASAM MCD-2MC metalanguage.
-> LAYOUT	For multi-dimensional measurement arrays this keyword can be used to specify the layout of the array. If the keyword is missing, multi-dimensional measurement arrays are interpreted row by row (ROW_DIR).
-> MATRIX_DIM	Shows the size and dimension of a multidimensional measurement. If the MATRIX_DIM keyword is used, then the option ARRAY_SIZE is not needed. However, if the keywords ARRAY_SIZE and MATRIX_DIM are both used, the resulting value in ARRAY_SIZE must be the same as xDim * yDim *zDim for MATRIX_DIM. If the keyword is missing the array has only one dimension with the size given at ARRAY_SIZE.
-> MAX_REFRESH	Maximum refresh rate of this measurement in the control unit
-> PHYS_UNIT	With this keyword a physical unit can be specified for the measurement object if no conversion rule is referenced (NO_COMPU_METHOD). Note: If a conversion rule is referenced the additional usage of PHYS_UNIT overrules the unit specified at the referenced conversion rule
-> READ_WRITE	Keyword to mark this measurement object as 'writeable'
-> REF_MEMORY_SEGMENT	Reference to the memory segment which is needed if the address is not unique (this occurs in the case of lapping address ranges (overlapping memory segments).
-> SYMBOL_LINK	Reference to symbol name within a linker map file.
-> VIRTUAL	For description of a virtual measurement (see VIRTUAL)

Description:

The MEASUREMENT keyword is used to describe the parameters for the processing of a measurement object.

Example:

```
/begin MEASUREMENT      N                /* name */
                        "Engine speed"   /* long identifier */
                        UWORD            /* datatype */
                        R_SPEED_3        /* conversion */
                        2                /* resolution */
                        2.5              /* accuracy */
                        120.0            /* lower limit */
                        8400.0           /* upper limit */
  PHYS_UNIT             "mph"
  BIT_MASK              0x0FFF
  /begin BIT_OPERATION
      RIGHT_SHIFT       4                /*4 positions*/
```

```
                                 SIGN_EXTEND
     /end BIT_OPERATION
        BYTE_ORDER               MSB_FIRST
     REF_MEMORY_SEGMENT          Data2
     /begin FUNCTION_LIST        ID_ADJUSTM
                                 FL_ADJUSTM
     /end FUNCTION_LIST
     /begin IF_DATA ISO          SND
                                 0x10
                                 0x00
                                 0x05
                                 0x08
                                 RCV
                                 4
                                 long
     /end IF_DATA
   /end MEASUREMENT

   /begin MEASUREMENT            VdiagStatus         /* name */
                                 "VdiagStatus"       /* long identifier */
                                 SWORD               /* datatype */
                                 CM_DiagSTatus       /* conversion */
                                 16                  /* resolution */
                                 1                   /* accuracy */
                                 -32768              /* lower limit */
                                 32767               /* upper limit */
        ECU_ADDRESS              0x003FDFE0
   /end MEASUREMENT

   /begin MEASUREMENT            VfSpinLoss          /* name */
                                 "VfSpinLoss"        /* long identifier */
                                 UWORD               /* datatype */
                                 CM_RPM              /* conversion */
                                 16                  /* resolution */
                                 1                   /* accuracy */
                                 -4096               /* lower limit */
                                 4095.875            /* upper limit */
        ECU_ADDRESS              0x003FE380
   /end MEASUREMENT
```

3.5.85 MEMORY_LAYOUT

Prototype:

```
/begin MEMORY_LAYOUT       enum      PrgType
                           ulong     Address
                           ulong     Size
                           long[5]   Offset
                           {-> IF_DATA}*
/end MEMORY_LAYOUT
```

Parameters:

enum	PrgType	Description of the program segments divided into:
		PRG_CODE = program code
		PRG_DATA = program data
		PRG_RESERVED = other

ulong	Address	Initial address of the program segment to be described.
ulong	Size	Length of the program segment to be described.
long [5]	Offset	In special ECU programs, so-called 'mirrored segments' may occur (see Figure 5 Memory layout (mirrored segments)). A mirrored segment is a copy of another program segment. During adjustment the data changes are introduced in the relevant memory segment as well as in all mirrored segments.

Optional parameters

-> IF_DATA	Data record to describe interface specific data of the memory layout. The parameters associated with this keyword have to be described in the ASAM MCD-2MC metalanguage.

Description:

This data record is used to describe an ECU program. The description indicates how the emulation memory is divided into the individual segments.

Example:

```
/begin MEMORY_LAYOUT     PRG_RESERVED
                         0x0000
                         0x0400
                         -1 -1 -1 -1 -1
/end MEMORY_LAYOUT
/begin MEMORY_LAYOUT     PRG_CODE
                         0x0400
                         0x3C00
                         -1 -1 -1 -1 -1
/end MEMORY_LAYOUT
/begin MEMORY_LAYOUT     PRG_DATA
                         0x4000
                         0x0200
                         0x10000
                         0x20000
                         -1 -1 -1
/end MEMORY_LAYOUT
```

```
/begin MEMORY_LAYOUT     PRG_DATA
                         0x4200
                         0x0E00
                         -1 -1 -1 -1 -1
/end MEMORY_LAYOUT
/begin MEMORY_LAYOUT     PRG_DATA
                         0x14200
                         0x0E00
                         -1 -1 -1 -1 -1
/end MEMORY_LAYOUT
/begin MEMORY_LAYOUT     PRG_DATA
                         0x24200
                         0x0E00
                         -1 -1 -1 -1 -1
/end MEMORY_LAYOUT
```

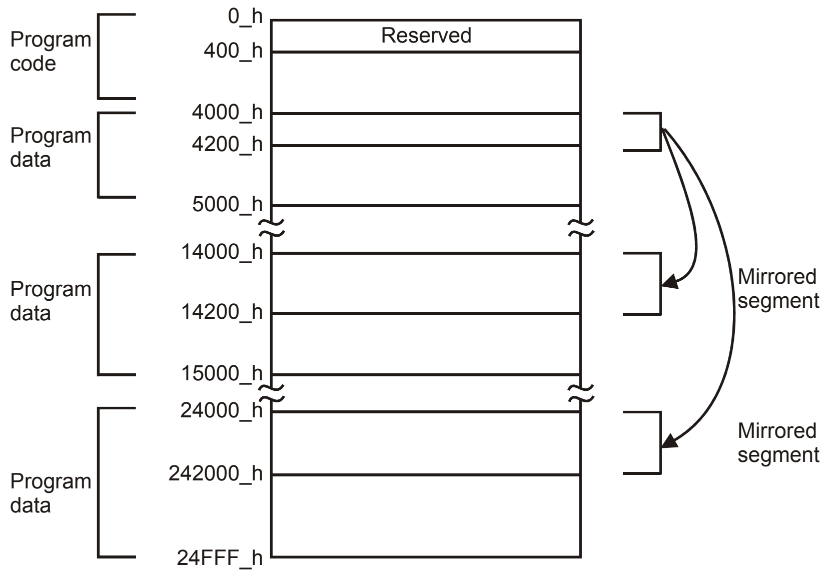


Figure 5 **Memory layout (mirrored segments)**

3.5.86 MEMORY_SEGMENT

Prototype:

```
/begin  MEMORY_SEGMENT    ident     Name
                          string    LongIdentifier
                          enum      PrgType
                          enum      MemoryType
                          enum      Attribute
                          ulong     Address
                          ulong     Size
                          long[5]   Offset
                          {-> IF_DATA}*
/end  MEMORY_SEGMENT
```

Parameters:

ident	Name	identifier, reference to IF_DATA Blob is based on this ´name´
string	LongIdentifier	comment, description
enum	PrgType	PrgTypes:
		CALIBRATION_VARIABLES = Values which are available in the ECU but do not exist in the Hex-file. There is no upload required to get access to the ECU data. The ECU will never be touched by the instrumentation tool except by upload.
		CODE = program code
		DATA = program data allowed for online calibration
		EXCLUDE_FROM_FLASH = values existing in the ECU but not dropped down in the binary file. There should no upload be needed to get access to the ECU data. The ECU will never be touched by the instrumentation tool except by upload.
		OFFLINE_DATA = program data allowed only for offline calibration
		RESERVED = reserved segments
		SERAM = program data for serial emulation
		VARIABLES = program variables
enum	MemoryType	Description of the type of memory used
		EEPROM = segment of EEPROM
		EPROM = segment of EPROM
		FLASH = segment of FLASH
		RAM = segment of RAM
		ROM = segment of ROM
		REGISTER = segment of CPU registers
enum	Attribute	attributes
		INTERN = internal segment
		EXTERN = external segment
ulong	Address	Initial address
ulong	Size	Length of the segment
long[5]	Offset	Offset address of mirrored segments

Optional Parameters:

-> IF_DATA	Data record to describe interface specific data of the memory segment. The parameters associated with this keyword have to be described in the ASAM MCD-2MC metalanguage.

Description:

The new keyword MEMORY_SEGMENT is used to replace the existing keyword MEMORY_LAYOUT. The advantages of MEMORY_SEGMENT are that they are given a name which can be used for references from IF_DATA Blobs and the more accurate description of the memory by memory types and attributes (INTERN and EXTERN).
The keywords MEMORY_SEGMENT and MEMORY_LAYOUT can be used in parallel.
The parameter Offset is to be used (as within the former MEMORY_LAYOUT) to describe several mirrored segments.
MEMORY_SEGMENTS with the same MemoryType and the same Attribute may not overlap. Also all MEMORY_SEGMENTS with the PrgType CODE, DATA, OFFLINE_DATA, RESERVED may not overlap mutually to get a linear address space for access on calibration data. All other MEMORY_SEGMENTS with different MemoryType or different Attribute may however overlap, e.g. internal and external memory segments.

The following table gives a description for some useful combinations of PrgType and MemoryType and their meanings:

Table 11 MEMORY_SEGMENT

Combination	Meaning
CODE / FLASH	Executable code, has to be preserved for download and HEX-file generation
DATA / FLASH or DATA / EEPROM	Calibration data, can be modified by the user via calibration systems.
RESERVED / FLASH	ECU specific code or data, has to be preserved for HEX-file generation but not for download.
DATA / RAM	Calibration data, will be modified by ECU and calibration systems.
OFFLINE_DATA / FLASH	Calibration data, will be modified only without ECU access, online calibration is not allowed.
VARIABLES / RAM	RAM of the ECU for variables (measurement values and others).
REGISTER / RAM	RAM of the ECU for special purpose values.
SERAM / RAM	ECU-RAM section available for serial calibration. For usage see also: CALIBRATION_METHOD.

Note: The MemoryType FLASH has been used as synonym for EPROM and ROM

Example:

```
/begin MEMORY_SEGMENT    Data1
                         "Data internal Flash"
                         DATA
                         FLASH
                         INTERN
                         0x4000
                         0x0200
                         0x10000
                         -1 -1 -1 -1 -1
/end MEMORY_SEGMENT
/begin MEMORY_SEGMENT    Data2
                         "Data external Flash"
                         DATA
                         FLASH
                         EXTERN
                         0x7000
                         0x2000
                         -1 -1 -1 -1 -1
/end MEMORY_SEGMENT
/begin MEMORY_SEGMENT    Code1
                         "Code external Flash"
                         CODE
                         FLASH
                         EXTERN
                         0x9000
                         0x3000
                         -1 -1 -1 -1 -1
/end MEMORY_SEGMENT
/begin MEMORY_SEGMENT    ext_Ram
                         "external RAM"
                         DATA
                         RAM
                         EXTERN
                         0x30000
                         0x1000
                         -1 -1 -1 -1 -1
/end MEMORY_SEGMENT
/begin MEMORY_SEGMENT    int_Ram
                         "internal RAM"
                         DATA
                         RAM
                         INTERN
                         0x0000
                         0x0200
                         -1 -1 -1 -1 -1
/end MEMORY_SEGMENT
/begin MEMORY_SEGMENT    Seram1
                         "emulation RAM 1"
                         SERAM
                         RAM
                         EXTERN
                         0x7000
                         0x1000
                         -1 -1 -1 -1 -1
/end MEMORY_SEGMENT
```

```
/begin MEMORY_SEGMENT   Seram2
                        "emulation RAM 2"
                        SERAM
                        RAM
                        INTERN
                        0x8000
                        0x1000
                        -1 -1 -1 -1 -1
/end MEMORY_SEGMENT
```

3.5.87 MOD_COMMON

Prototype:

```
/begin MOD_COMMON          string   Comment
                           [-> ALIGNMENT_BYTE]
                           [-> ALIGNMENT_FLOAT32_IEEE]
                           [-> ALIGNMENT_FLOAT64_IEEE]
                           [-> ALIGNMENT_INT64]
                           [-> ALIGNMENT_LONG]
                           [-> ALIGNMENT_WORD]
                           [-> BYTE_ORDER]
                           [-> DATA_SIZE]
                           [-> DEPOSIT]
                           [-> S_REC_LAYOUT]
/end MOD_COMMON
```

Parameters:

string Comment	comment, description

Optional parameters:

-> ALIGNMENT_BYTE	Declares the alignment of bytes in the complete module. The alignment is 1 if parameter is missing.
-> ALIGNMENT_LONG	Declares the alignment of longs in the complete module. The alignment is 4 if parameter is missing.
-> ALIGNMENT_FLOAT32_IEEE	Declares the alignment of 32 bit floats in the complete module. The alignment is 4 if parameter is missing.
-> ALIGNMENT_FLOAT64_IEEE	Declares the alignment of 64 bit floats in the complete module. The alignment is 4 if parameter is missing.
-> ALIGNMENT_INT64	Declares the alignment of int64 in the complete module. The alignment is 8 if parameter is missing.
-> ALIGNMENT_WORD	Declares the alignment of words in the complete module. The alignment is 2 if parameter is missing.
-> BYTE_ORDER	Byte order for the whole device. If this optional parameter is not declared, MSB_LAST (Intel format) is used as a default.
-> DATA_SIZE	Data size in bits
-> DEPOSIT	Standard deposit mode for axis points: ASBOLUTE or DIFFERENCE
-> S_REC_LAYOUT	Reference to the standard record layout

Description:

This keyword is used to specify general description data for the module, which are then used as standard in this module. Should other methods be used for an object (e.g. adjustable object or measurement object) of this module, this must then be indicated in the description of the relevant object.

Example:

```
/begin MOD_COMMON        "Characteristic maps always deposited in
                         same mode"
      S_REC_LAYOUT       S_ABL
      DEPOSIT            ABSOLUTE
      BYTE_ORDER         MSB_LAST
      DATA_SIZE          16
      ALIGNMENT_BYTE     2
/end MOD_COMMON
```

3.5.88 MOD_PAR

Prototype:

```
/begin MOD_PAR              string    Comment
                            {-> ADDR_EPK}*
                            {-> CALIBRATION_METHOD}*
                            [-> CPU_TYPE]
                            [-> CUSTOMER]
                            [-> CUSTOMER_NO]
                            [-> ECU]
                            [-> ECU_CALIBRATION_OFFSET]
                            [-> EPK]
                            {-> MEMORY_LAYOUT}*
                            {-> MEMORY_SEGMENT}*
                            [-> NO_OF_INTERFACES]
                            [-> PHONE_NO]
                            [-> SUPPLIER]
                            {-> SYSTEM_CONSTANT}*
                            [-> USER]
                            [-> VERSION]
/end MOD_PAR
```

Parameters:

string Comment	comment, description relating to the ECU-specific management data

Optional parameters:

-> ADDR_EPK	Address of EPROM identifier
-> CALIBRATION_METHOD	Declares the implemented calibration methods in the control unit.
-> CPU_TYPE	CPU
-> CUSTOMER	Firm or customer
-> CUSTOMER_NO	Customer number
-> ECU	Control unit
-> ECU_CALIBRATION_OFFSET	Offset that has to be added to each address of a characteristic.
-> EPK	EPROM identifier
-> MEMORY_LAYOUT	Memory layout
-> MEMORY_SEGMENT	Declares the available memory segments.
-> NO_OF_INTERFACES	Number of interfaces
-> PHONE_NO	Phone number of the calibration engineer responsible
-> SUPPLIER	Manufacturer or supplier
-> SYSTEM_CONSTANT	System-defined constants
-> USER	User
-> VERSION	Version identifier

Description:

The MOD_PAR keyword describes the management data to be specified for an device. Except for the comment all parameters are optional.

Example:

```
    /begin MOD_PAR           "Note: Provisional release for test purposes
                             only!"
          VERSION            "Test version of 01.02.1994"
          ADDR_EPK           0x45678
          EPK                EPROM identifier test
          SUPPLIER           "M&K GmbH Chemnitz"
          CUSTOMER           "LANZ-Landmaschinen"
          CUSTOMER_NO        "0123456789"
          USER               "A.N.Wender"
          PHONE_NO           "09951 56456"
          ECU                "Engine control"
          CPU_TYPE           "Motorola 0815"
          NO_OF_INTERFACES   2
      /begin MEMORY_SEGMENT  ext_Ram
                             "external RAM"
                             DATA
                             RAM
                             EXTERN
                             0x30000
                             0x1000
                             -1 -1 -1 -1 -1
      /begin IF_DATA ASAP1B_KWP2000
      /* ADDRESS_MAPPING     orig_addr    mapping_addr   length */
          ADDRESS_MAPPING    0x4000       0x6000         0x0200
      /end IF_DATA
      /end MEMORY_SEGMENT
      /begin MEMORY_LAYOUT   PRG_RESERVED
                             0x0000
                             0x0400
                             -1 -1 -1 -1 -1
      /end MEMORY_LAYOUT
      /begin MEMORY_LAYOUT   PRG_CODE
                             0x0400
                             0x3C00
                             -1 -1 -1 -1 -1
      /end MEMORY_LAYOUT
      /begin MEMORY_LAYOUT   PRG_DATA
                             0x4000
                             0x5800
                             -1 -1 -1 -1 -1
      /end MEMORY_LAYOUT
      SYSTEM_CONSTANT        "CONTROLLERx constant1" "0.33"
      SYSTEM_CONSTANT        "CONTROLLERx constant2" "2.79"
    /end MOD_PAR
```

3.5.89 MODULE

Prototype:

```
/begin MODULE             ident    Name
                          string   LongIdentifier
                          [-> A2ML]
                          {-> AXIS_PTS}*
                          {-> CHARACTERISTIC}*
                          {-> COMPU_METHOD}*
                          {-> COMPU_TAB}*
                          {-> COMPU_VTAB}*
                          {-> COMPU_VTAB_RANGE}*
                          [-> FRAME]
                          {-> FUNCTION}*
                          {-> GROUP}*
                          {-> IF_DATA}*
                          {-> MEASUREMENT}*
                          [-> MOD_COMMON]
                          [-> MOD_PAR]
                          {-> RECORD_LAYOUT}*
                          {-> UNIT}*
                          {-> USER_RIGHTS}*
                          [-> VARIANT_CODING]
/end MODULE
```

Parameters:

ident	Name	device identifier
string	LongIdentifier	comment, description

Optional parameters:

-> A2ML	Format description of the interface-specific parameters. Note: The interface-specific parameters must be specified directly after the last mandatory parameter 'long identifier'.
-> AXIS_PTS	Keyword for the description of the axis points
-> CHARACTERISTIC	Keyword for the description of the adjustable objects
-> COMPU_METHOD	Keyword for the description of the conversion method
-> COMPU_TAB	Keyword for the description of the conversion tables
-> COMPU_VTAB	Keyword for the description of the verbal conversion tables
-> COMPU_VTAB_RANGE	Keyword for the description of range-based verbal conversion tables
-> FUNCTION	Keyword for the description of the functions
-> FRAME	Keyword for the declaration of frames
-> GROUP	Keyword for the declaration of groups
-> IF_DATA	Data record to describe interface specific data of the device. The parameters associated with this keyword have to be described in the ASAM MCD-2MC metalanguage. -> MEASUREMENT Keyword for the description of the measurement objects
-> MOD_COMMON	Module-wide description data
-> MOD_PAR	Keyword for the description of module-specific (device-specific) management data.
-> RECORD_LAYOUT	Keyword for the description of the record layouts

-> UNIT	Keyword for the description of the measurement units
-> USER_RIGHTS	Keyword to reference the groups which constitute access rights.
-> VARIANT_CODING	Keyword to describe the variant coding of adjustable objects.

Description:

The MODULE keyword describes a complete ECU or device with all adjustable and measurement objects, conversion methods and functions. To this, the format description of the interface-specific parameters by the ECU supplier must be added.

Note: It is possible to have a measurement object and a computation method with equal names within the same MODULE. It is NOT possible to have a measurement object and a calibration object with equal names within the same MODULE.

Example:

see B.3 ENGINE_ECU.A2L

3.5.90 MONOTONY

Prototype:

```
MONOTONY                      enum      Monotony
```

Parameters:

enum	Monotony	Description of the monotony:	
		MON_DECREASE	monotonously decreasing
		MON_INCREASE	monotonously increasing
		STRICT_DECREASE	strict monotonously decreasing
		STRICT_INCREASE	strict monotonously increasing
		MONOTONOUS	monotonously in- or decreasing
		STRICT_MON	strict monotonously in- or decreasing
		NOT_MON	no monotony required.

Description:

This keyword can be used to specify the monotony of an adjustment object. The monotony is always related to an axis (see keyword "AXIS_DESCR"). With each adjustment operation the measurement and calibration system (user interface) verifies whether the monotony is guaranteed. Changes that do not correspond to the monotony are not allowed.

Note: NOT_MON requires additionally an inverse computation method. If the inverse computation method is missing it is not possible to calculate both directions. A tool may offer then only restricted calibration access.

Note: Monotony is used in reference to internal values, not physical values.

Note: If the keyword is missing the monotony check is tool dependent.

Example:

```
  MONOTONY                    MON_INCREASE
```

3.5.91 NO_AXIS_PTS_X / _Y / _Z / _4 / _5

Prototype:

```
NO_AXIS_PTS_X /_Y /_Z / _4 / _5
                          uint      Position
                          datatype  Datatype
```

Parameters:

uint	Position	Position of the number of axis points in the deposit structure
datatype	Datatype	Data type of the number of axis points

Description:

Description of the number of axis points in an adjustable object

Example:

```
  NO_AXIS_PTS_X             2
                            UWORD
```

3.5.92 NO_OF_INTERFACES

Prototype:

```
NO_OF_INTERFACES          uint      Num
```

Parameters:

uint	Num	Number of interfaces

Description:

Keyword for the number of interfaces

Example:

```
  NO_OF_INTERFACES        2
```

3.5.93 NO_RESCALE_X / _Y / _Z / _4 / _5

Prototype:

```
NO_RESCALE_X /_Y /_Z / _4 / _5
                            uint      Position
                            datatype  Datatype
```

Parameters:

uint	Position	position of the actual number of rescale axis point value pairs in the deposit structure (description of sequence of elements in the data record).
datatype	DataType	Data type of the number of rescale axis point value pairs

Description:

Actual number of rescaling axis point value pairs.

Example:

```
 NO_RESCALE_X             1
                          UBYTE
```

3.5.94 NUMBER

Prototype:

```
NUMBER                        uint      Number
```

Parameters:

uint Number	Number of values (array of values) or characters (string).

Description:

In the CHARACTERISTIC data record, this keyword can be used to specify the number of values and characters for the adjustable object types 'array of values' (VAL_BLK) and 'string' (ASCII) respectively.
Note: The use of this keyword should be replaced by MATRIX_DIM.

Example:

```
NUMBER                        7
```

3.5.95 OFFSET_X / _Y / _Z / _4 / _5

Prototype:

```
OFFSET_X /_Y /_Z / _4 / _5
                                    uint      Position
                                    datatype  Datatype
```

Parameters:

uint	Position	Position of the 'offset' parameter in the deposit structure to compute the axis points for fixed characteristic curves and fixed characteristic maps.
datatype	Datatype	Data type of the 'offset' parameter.

Description:

Description of the 'offset' parameter in the deposit structure to compute the axis points for fixed characteristic curves and fixed characteristic maps (see also keyword FIX_AXIS_PAR). The axis points for fixed characteristic curves or fixed characteristic maps are derived from the two 'offset' and 'shift' parameters as follows:

$$X_i = \text{Offset} + (i - 1)*2^{\text{Shift}} \quad i = \{ 1...\text{numberofaxispts} \}$$

Example:

```
 OFFSET_X                    16
                             UWORD
```

3.5.96 OUT_MEASUREMENT

Prototype:

```
/begin OUT_MEASUREMENT    { ident   Identifier } *
/end OUT_MEASUREMENT
```

Parameters:

ident	Identifier	Identifier of output quantity of respective function (reference to measurement object).

Description:

This keyword can be used to define output quantities of respective function.
Note: OUT_MEASUREMENT may only refer to objects of type MEASUREMENT.

Example:

```
/begin OUT_MEASUREMENT   OK_FLAG
                         SENSOR_FLAG
/end OUT_MEASUREMENT
```

3.5.97 PHONE_NO

Prototype:

```
PHONE_NO                        string    Telnum
```

Parameters:

string Telnum	phone number

Description:

This keyword is used to specify a phone number, e.g. of the calibration engineer responsible.

Example:

```
  PHONE_NO                        "09498 594562"
```

3.5.98 PHYS_UNIT

Prototype:

```
PHYS_UNIT                    string Unit
```

Parameters:

string	Unit	Physical unit.

Description:

With this keyword a physical unit can be specified for a measure or calibration object if no conversion rule is used (NO_COMPU_METHOD). If a conversion rule is used, the additional usage of PHYS_UNIT overrules the default unit specified at the referenced conversion rule. The keyword can be used at AXIS_PTS, AXIS_DESCR, CHARACTERISTIC and MEASUREMENT.

Example:

```
PHYS_UNIT            "°C"
```

3.5.99 PROJECT

Prototype:

```
/begin PROJECT              ident     Name
                            string    LongIdentifier
                            [-> HEADER]
                            {-> MODULE}*
/end PROJECT
```

Parameters:

ident	Name	Project identifier in the program
string	LongIdentifier	Comment, description

Optional parameters:

-> HEADER	Project header
-> MODULE	This keyword is used to describe the devices belonging to the project.

Description:

Project description with project header and all devices belonging to the project. The PROJECT keyword covers the description of several control units, and possibly also of several suppliers.

Example:

```
/begin PROJECT           RAPE-SEED ENGINE
                         "Engine tuning for operation with rape oil"
  /begin HEADER          "see also specification XYZ of 01.02.1994"
     VERSION             "BG5.0815"
     PROJECT_NO          M4711Z1
  /end HEADER

  /include ENGINE_ECU.A2L    /* Include for engine control module */
  /include ABS_ECU.A2L       /* Include for ABS module */
/end PROJECT
```

3.5.100 PROJECT_NO

Prototype:

```
PROJECT_NO                  ident     ProjectNumber
```

Parameters:

ident	ProjectNumber	Short identifier of the project number

Description:

String used to identify the project number with maximum MAX_IDENT characters.

Example:

```
  PROJECT_NO                M4711Z1
```

3.5.101 READ_ONLY

Prototype:

```
READ_ONLY
```

Description:

This keyword is used to indicate that an adjustable object cannot be changed (but can only be read).

Example:

```
/begin CHARACTERISTIC   KI "I-share for speed limitation"
                        VALUE         /* type: fixed value */
                        0x408F        /* address */
                        DAMOS_FW      /* deposit */
                        0.0           /* max_diff */
                        FACTOR01      /* conversion */
                        0.0           /* lower limit */
                        255.0         /* upper limit */

  /* interface-specific parameters: address location, addressing */
  /begin IF_DATA "DIM"  EXTERNAL
                        DIRECT
  /end IF_DATA
  /begin FUNCTION_LIST  V_LIM         /* Reference to functions */
  /end FUNCTION_LIST
  READ_ONLY
/end CHARACTERISTIC
```

3.5.102 READ_WRITE

Prototype:

```
READ_WRITE
```

Description:

This keyword is used to mark a measurement object to be writeable.

Example:

```
/begin MEASUREMENT     N               /* name */
                       "Engine speed"  /* long identifier */
                       UWORD           /* datatype */
                       R_SPEED_3       /* conversion */
                       2               /* resolution */
                       2.5             /* accuracy */
                       120.0           /* lower limit */
                       8400.0          /* upper limit */
  READ_WRITE
  /begin IF_DATA ISO   SND
                       0x10
                       0x00
                       0x05
                       0x08
                       RCV
                       4
                       long
  /end IF_DATA
/end MEASUREMENT
```

3.5.103 RECORD_LAYOUT

Prototype:

```
   /begin RECORD_LAYOUT        ident    Name
                               [-> ALIGNMENT_BYTE]
                               [-> ALIGNMENT_FLOAT32_IEEE]
                               [-> ALIGNMENT_FLOAT64_IEEE]
                               [-> ALIGNMENT_INT64]
                               [-> ALIGNMENT_LONG]
                               [-> ALIGNMENT_WORD]
                               [-> AXIS_PTS_X/_Y/_Z/_4/_5]
                               [-> AXIS_RESCALE_X/_Y/_Z/_4/_5]
                               [-> DIST_OP_X/_Y/_Z/_4/_5]
                               [-> FIX_NO_AXIS_PTS_X/_Y/_Z/_4/_5]
                               [-> FNC_VALUES]
                               [-> IDENTIFICATION]
                               [-> NO_AXIS_PTS_X/_Y/_Z/_4/_5]
                               [-> NO_RESCALE_X/_Y/_Z/_4/_5]
                               [-> OFFSET_X/_Y/_Z/_4/_5]
                               {-> RESERVED}*
                               [-> RIP_ADDR_W/_X/_Y/_Z/_4/_5]
                               [-> SRC_ADDR_X/_Y/_Z/_4/_5]
                               [-> SHIFT_OP_X/_Y/_Z/_4/_5]
                               [-> STATIC_RECORD_LAYOUT]
   /end RECORD_LAYOUT
```

Parameters:

ident Name	Identification of the record layout, which is referenced via this 'name'. Note: The name of the record layout has to be unique within all record layouts of the ASAM MCD-2MC MODULE, i.e. there must not be another RECORD_LAYOUT object with the same identifier in the MODULE.

Optional parameters:

-> ALIGNMENT_BYTE	Declares the alignment of bytes for all characteristics which use this record layout. If the keyword is missing, the alignment defined at MOD_COMMON is used.
-> ALIGNMENT_LONG	Declares the alignment of longs for all characteristics which use this record layout. If the keyword is missing, the alignment defined at MOD_COMMON is used. -> ALIGNMENT_FLOAT32_IEEE Declares the alignment of 32 bit floats for all characteristics which use this record layout. If the keyword is missing, the alignment defined at MOD_COMMON is used. -> ALIGNMENT_FLOAT64_IEEE Declares the alignment of 64 bit floats for all characteristics which use this record layout. If the keyword is missing, the alignment defined at MOD_COMMON is used. -> ALIGNMENT_INT64 Declares the alignment of int64 for all characteristics which use this record

	layout. If the keyword is missing, the alignment defined at MOD_COMMON is used.
-> ALIGNMENT_WORD	Declares the alignment of words for all characteristics which use this record layout. . If the keyword is missing, the alignment defined at MOD_COMMON is used.-> AXIS_PTS_X / _Y / _Z / _4 / _5 Describes where the X, Y, Z, Z4 or Z5 axis points are deposited in memory.
-> AXIS_RESCALE_X / _Y / _Z / _4 / _5	Describes where the rescale mapping for the X, Y, Z, Z4 or Z5 axis is deposited in memory.
-> DIST_OP_X / _Y / _Z / _4 / _5	'Distance' parameter to compute the axis points of fixed Characteristics.
-> FIX_NO_AXIS_PTS_X / _ Y / _Z / _4 / _5	Indicates that a fixed number of axis points is allocated. Note: In a RECORD_LAYOUT data record, this keyword may not be used simultaneously with the keyword 'NO_AXIS_PTS_X / _Y / _Z / _4 / _5.
-> FNC_VALUES	This keyword describes how the table values (function values) of the adjustable object are deposited in memory.
-> IDENTIFICATION	This keyword is used to describe that an 'identifier' is deposited in a specific position in the adjustable object.
-> NO_AXIS_PTS_X / _Y / _Z / _4 / _5	Describes in which position the parameter 'number of axis points' is deposited in memory.
-> NO_RESCALE_X / _Y / _Z / _4 / _5	Describes at which position the parameter 'current number of rescale pairs' for the axis is deposited (see AXIS_RESCALE_X / _Y / _Z / _4 / _5).
-> OFFSET_X / _Y / _Z / _4 / _5	Offset to compute the axis points of fixed Characteristics.
-> RESERVED	This keyword can be used to skip specific elements in the adjustable object whose meaning must not be interpreted by the measurement and calibration system (e.g. for extensions: new parameters in the adjustable objects).
-> RIP_ADDR_W	Final result (table value) of the ECU-internal inter-polation. (output value)
-> RIP_ADDR_X / _Y / _Z / _4 / _5	Is used to describe at which position the address of this RIP_X, RIP_Y, RIP_Z, RIP_Z4 or RIP_Z5 quantity is deposited, which contains the current value of the ECU-internal interpolation. (input values)
-> SHIFT_OP_X / _Y / _Z / _4 / _5	Shift operand to compute the axis points of fixed Characteristics.
-> SRC_ADDR_X / _Y / _Z / _4 / _5	Describes at which position the address of the input quantity of the axis is deposited in memory.

-> STATIC_RECORD_LAYOUT	For calibration objects with dynamic number of axis points this keyword indicates that the calibration object **does not** compact or expand data when removing resp. inserting axis points. All record layout elements are stored at the same address as for the max. number of axis points specified at the calibration object - independent of the actual number of axis points. If the parameter STATIC_RECORD_LAYOUT is missing, the calibration objects referencing this record layout **do** compact / extend data when removing resp. inserting axis points and the addresses of the record layout elements depend on the actual number of axis points. This parameter may be used only together with NO_AXIS_PTS_X/_Y/_Z/_4/_5 .

Description:

The 'RECORD_LAYOUT' keyword is used to specify the various data structures of the adjustable objects in the memory. The structural buildup of the various adjustable object types must be described in such a way that a standard measurement and calibration system will be able to process all adjustable object types (reading, writing, operating point display etc.).
In particular, if the ALTERNATE option is used with FNC_VALUES, the position parameter determines the order of values and axis points.

Note: To describe the record layouts, use is made of a predefined list of parameters which may be part of an adjustable object (characteristic) in the emulation memory. This list represents the current status of the record layouts. With each change or extension of the record layouts contained in this predefined list of parameters the ASAM MCD-2MC description file format must be modified accordingly.

Note: The keywords describing axis parameters for CUBE_4 and CUBE_5 are extended by _X, _Y, _Z, _4, _5. This allows an easier understanding which dimension the axis description belongs to. In the textual description the axes for CUBE_4 and CUBE_5 are named X, Y, Z, Z4, Z5. This allows textual description without a reference to a keyword (Z4-axis describes the axis better as 4-axis)

Note: For CUBOID, CUBE_4 and CUBE_5 the RECORD_LAYOUT supports only one dedicated way to sort the data in the memory. These objects are always stored as array of MAP with incremented or decremented axes. The exchange of dimensions (e.g. X, Z, Y, Z5, Z4) in the memory is not supported.

Example:

```
/begin RECORD_LAYOUT     DAMOS_KF
       FNC_VALUES        7 SWORD COLUMN_DIR DIRECT
       AXIS_PTS_X        3 SWORD INDEX_INCR DIRECT
       AXIS_PTS_Y        6 UBYTE INDEX_INCR DIRECT
       NO_AXIS_PTS_X     2 UBYTE
       NO_AXIS_PTS_Y     5 UBYTE
       SRC_ADDR_X        1
       SRC_ADDR_Y        4
```

```
      ALIGNMENT_BYTE      2
/end RECORD_LAYOUT

/begin RECORD_LAYOUT      RESCALE_SST
      NO_RESCALE_X        1 UBYTE
      RESERVED            2 BYTE
      AXIS_RESCALE_X      3 UBYTE 5 INDEX_INCR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT      SHORTINT
      FNC_VALUES          1 SBYTE ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT      BYTE
      FNC_VALUES          1 UBYTE ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT      INTEGER
      FNC_VALUES          1 SWORD ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT      WORD
      FNC_VALUES          1 UWORD ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT      LONGINT
      FNC_VALUES          1 SLONG ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT      LONGWORD
      FNC_VALUES          1 ULONG ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT      2D_structure_table_int
      NO_AXIS_PTS_X       1 UWORD
      FNC_VALUES          2 SWORD ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT      2D_structure_table_word
      NO_AXIS_PTS_X       1 UWORD
      FNC_VALUES          2 UWORD ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT      2D_structure_table_byte
      NO_AXIS_PTS_X       1 UBYTE
      RESERVED            2 BYTE
      FNC_VALUES          3 UBYTE ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT      2D_structure_table_shortint
      NO_AXIS_PTS_X       1 UBYTE
      RESERVED            2 BYTE
      FNC_VALUES          3 SBYTE ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT      3D_structure_table_int
      NO_AXIS_PTS_X       1 UWORD
```

```
        NO_AXIS_PTS_Y       2 UWORD
        FNC_VALUES          3 SWORD ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT        3D_structure_table_word
        NO_AXIS_PTS_X       1 UWORD
        NO_AXIS_PTS_Y       2 UWORD
        FNC_VALUES          3 UWORD ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT        3D_structure_table_byte
        NO_AXIS_PTS_X       1 UBYTE
        NO_AXIS_PTS_Y       2 UBYTE
        RESERVED            3 BYTE
        FNC_VALUES          4 UBYTE ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT        3D_structure_table_shortint
        NO_AXIS_PTS_X       1 UBYTE
        NO_AXIS_PTS_X       2 UBYTE
        RESERVED            3 BYTE
        FNC_VALUES          4 SBYTE ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT        2D_array_table_int
        FNC_VALUES          1 SWORD ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT        2D_array_table_word
        FNC_VALUES          1 UWORD ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT        2D_array_table_byte
        FNC_VALUES          1 UBYTE ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT        2D_array_table_shortint
        FNC_VALUES          1 SBYTE ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT        3D_array_table_int
        FNC_VALUES          1 SWORD ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT        3D_array_table_word
        FNC_VALUES          1 UWORD ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT        3D_array_table_byte
        FNC_VALUES          1 UBYTE ROW_DIR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT        3D_array_table_shortint
        FNC_VALUES          1 SBYTE ROW_DIR DIRECT
/end RECORD_LAYOUT
```

3.5.104 REF_CHARACTERISTIC

Prototype:

```
/begin REF_CHARACTERISTIC { ident   Identifier }*
/end REF_CHARACTERISTIC
```

Parameters:

ident	Identifier	Identifier of those adjustable objects that are referred to respective function or group.

Description:

This keyword can be used to define some adjustable objects that are referenced in respective function or group.
Note: REF_CHARACTERISTIC may only refer to objects of type CHARACTERISTIC or AXIS_PTS.

Example:

```
/begin REF_CHARACTERISTIC    ENG_SPEED_CORR_CURVE
/end REF_CHARACTERISTIC
```

3.5.105 REF_GROUP

Prototype:

```
/begin REF_GROUP            { ident   Identifier }*
/end REF_GROUP
```

Parameters:

ident	Identifier	Identifier of those groups which are referred in USER_RIGHTS

Description:

This keyword can be used to refer groups which control the access rights of users logging into an MCD system.

Example:

```
  /begin REF_GROUP          GROUP_1
                            GROUP_2
  /end REF_GROUP
```

3.5.106 REF_MEASUREMENT

Prototype:

```
/begin REF_MEASUREMENT    { ident   Identifier }*
/end REF_MEASUREMENT
```

Parameters:

ident	Identifier	Identifier of those measurement quantities which are referred to the group.

Description:

This keyword can be used to define measurement quantities which are member of the respective function.

Example:

```
/begin REF_MEASUREMENT   LOOP_COUNTER
                         TEMPORARY_1
/end REF_MEASUREMENT
```

3.5.107 REF_MEMORY_SEGMENT

Prototype:

```
REF_MEMORY_SEGMENT          ident     Name
```

Parameters:

ident	Name	Name of memory segments

Description:

The reference to a memory segment is needed in characteristics and measurements. The memory segment, the characteristic belongs to can not be detected by the address itself in the case of overlapping memory segments.

Example:

```
REF_MEMORY_SEGMENT        Data1
```

3.5.108 REF_UNIT

Prototype:

```
REF_UNIT                    ident     Unit
```

Parameters:

ident Unit	reference to the data record which describes a measurement unit

Description:

This keyword can be used to reference to the data record which describes a measurement unit.
REF_UNIT may only refer to objects of type UNIT.
Note: The string parameter Unit of a COMPU_METHOD is a redundant information because the record referenced by REF_UNIT contain it too. Just for the purpose of compatibility with previous versions of ASAM MCD-2MC the parameter REF_UNIT is optional.

Example:

```
/begin COMPU_METHOD   Velocity
                      "conversion method for velocity"
                      RAT_FUNC
                      "%6.2"
                      "[km/h]"
               COEFFS 0 100 0 0 0 1
             REF_UNIT kms_per_hour    /* new (optional) parameter */
/end COMPU_METHOD
```

3.5.109 RESERVED

Prototype:

```
RESERVED                     uint      Position
                             datasize  DataSize
```

Parameters:

uint	Position	Position of the reserved parameter in the deposit structure
datasize	DataSize	Word length of the reserved parameter.

Description:

This keyword can be used to skip specific elements in an adjustable object whose meaning must not be interpreted by the measurement and calibration system (e.g. for extensions: new parameters in the adjustable objects).

Example:

```
 RESERVED                    7
                             LONG
```

Note: Only BYTE, WORD and LONG are valid datasize values. A datatype (UBYTE, SBYTE, UWORD, etc.) cannot be used in place of the datasize.

3.5.110 RIGHT_SHIFT

Prototype:

```
RIGHT_SHIFT                 ulong     Bitcount
```

Parameters:

ulong	Bitcount	Shift ‚Bitcount‘ bits to the right

Description:

The RIGHT_SHIFT keyword is only used within the BIT_OPERATION keyword. See description of BIT_OPERATION.

3.5.111 RIP_ADDR_W / _X / _Y / _Z / _4 / _5

Prototype:

```
RIP_ADDR_W / _X / _Y / _Z / _4 / _5
                                 uint      Position
                                 datatype  Datatype
```

Parameters:

uint	Position	Position of the address to the result of the ECU-internal interpolation (see below) in the deposit structure.
datatype	Datatype	Data type of the address

Description:

When the ECU program accesses a characteristic curve it determines an output value based on an input quantity. First it searches the adjacent axis points of the current value of the input quantities (Xi, Xi+1 or Yi, Yi+1 or Zi, Zi+1 or Z4i, Z4i+1 or Z5i, Z5i+1). The output value is derived from these axis points and the allocated table values by means of interpolation. This produces an 'intermediate result' known as the RIP_X / _Y / _Z / _4 / _5 quantity (Result of Interpolation), which describes the relative distance between the current value and the adjacent axis points (see Figure 6 Linear interpolation for a characteristic curve). The output value is derived from these axis points and the two allocated table values by means of interpolation. This produces as intermediate results the quantities RIP_X and RIP_Y, which describe the distance between the current value and the adjacent axis points:

$$RIP\_X = (X_{current} - X_i)/(X_{i+1} - X_i)$$

For a characteristic map the ECU program determines this intermediate result both in the X-direction and in the Y-direction. For a characteristic cuboid the result in the direction of all three axes are calculated.

$$RIP\_Y = (Y_{current} - Y_k)/(Y_{k+1} - Y_k)$$

$$RIP\_Z = (Z_{current} - Z_m)/(Z_{m+1} - Z_m)$$

For a characteristic curve the result of the interpolation is calculated as follows:

$$RIP\_W = W_i + (RIP\_X * (W_{i+1} - W_i)$$

for a characteristic map as follows:

$$RIP\_W = (W_{i,k} * (1 - RIP\_X) + W_{i+1,k} * RIP\_X)) * (1 - RIP\_Y) +$$
$$(W_{i,k+1} * (1 - RIP\_X) + W_{i+1,k+1} * RIP\_X)) * RIP\_Y$$

and for a characteristic cuboid as follows:

Interpolation for the map Z = m

$$RIP\_W_m = (W_{i,k,m} * (1 - RIP\_X) + W_{i+1,k,m} * RIP\_X)) * (1 - RIP\_Y) +$$
$$(W_{i,k+1,m} * (1 - RIP\_X) + W_{i+1,k+1,m} * RIP\_X)) * RIP\_Y$$

Interpolation for the map Z = m+1

$RIP\_W_{m+1} = (W_{i,k,m+1} * (1 - RIP\_X) + W_{i+1,k,m+1} * RIP\_X)) * (1 - RIP\_Y) +$

$(W_{i,k+1,m+1} * (1 - RIP\_X) + W_{i+1,k+1,m+1} * RIP\_X)) * RIP\_Y$

Interpolation in Z direction between the two points RIP_W_m and RIP_W_{m+1}.

$RIP\_W = RIP\_W_m +(RIP\_Z*(RIP\_W_{m+1} - RIP\_W_m)$

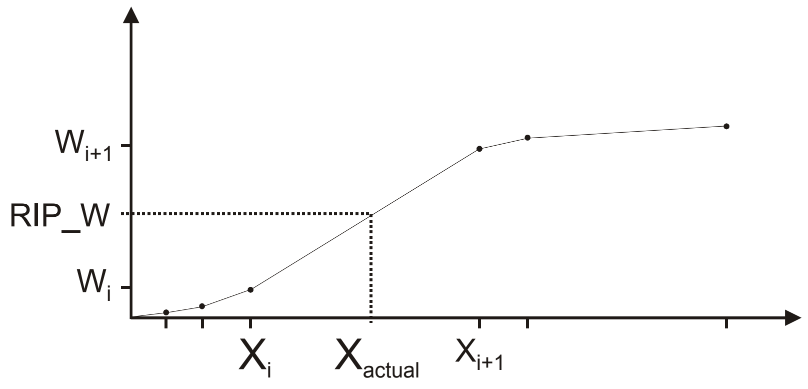


Figure 6 Linear interpolation for a characteristic curve

Example:

```
RIP_ADDR_X              19
                        UWORD
```

3.5.112 ROOT

Prototype:

```
ROOT
```

Parameters:

none

Description:

This keyword ROOT indicates that the related group is presented as a root of a navigation tree in the group selection mechanism of the MCD system. The keyword ROOT can indicate that groups referred to this root group constitute a grouping mechanism.

Example:

```
/begin GROUP            SOFTWARE_COMPONENTS
                        "assignment of the definitions to C files"
  ROOT
  /begin SUB_GROUP      INJE
                        C6TD
  /end SUB_GROUP
/end GROUP
```

3.5.113 SHIFT_OP_X / _Y / _Z / _4 / _5

Prototype:

```
SHIFT_OP_X / _Y / _Z / _4 / _5
                                 uint      Position
                                 datatype  Datatype
```

Parameters:

uint	Position	Position of the shift operand in the deposit structure.
datatype	Datatype	Data type of the shift operand.

Description:

Description of the shift operand in the deposit structure to compute the axis points for fixed characteristic curves and fixed characteristic maps (see also keyword FIX_AXIS_PAR). The axis points distribution for fixed characteristic curves or fixed characteristic maps is derived from the two 'offset' and 'shift' parameters as follows:

$$X_i = \text{Offset} + (i - 1)*2^{\text{Shift}} \quad i = \{ 1...\text{numberofaxispts} \}$$

Example:

```
 SHIFT_OP_X                  21
                             UWORD
```

3.5.114 SIGN_EXTEND

Prototype:

```
SIGN_EXTEND
```

Parameters:

none

Description:

The SIGN_EXTEND keyword is only used within the BIT_OPERATION keyword. See description of BIT_OPERATION.

3.5.115 SI_EXPONENTS

Prototype:

```
SI_EXPONENTS                 int       Length
                             int       Mass
                             int       Time
                             int       ElectricCurrent
                             int       Temperature
                             int       AmountOfSubstance
                             INT       LuminousIntensity
```

Parameters:

int	Length	exponent of the base dimension length with unit metre
int	Mass	exponent of the base dimension mass with unit kilogram
int	Time	exponent of the base dimension time with unit second
int	ElectricCurrent	exponent of the base dimension electric current with unit ampere
int	Temperature	exponent of the base dimension thermodynamic temperature with unit kelvin
int	AmountOfSubstance	exponent of the base dimension amount of substance with unit mole
int	LuminousIntensity	exponent of the base dimension luminous intensity with unit candela

Description:

Specification of the seven base dimensions required to define an extended SI unit.

Example:

```
/begin UNIT
                        newton
                        "extended SI unit for force"
                        "[N]"
                        EXTENDED_SI
        SI_EXPONENTS    1 1 -2 0 0 0 0     /*[N] = [m]*[kg]*[s]-2 */
/end UNIT
```

3.5.116 SRC_ADDR_X / _Y / _Z / _4 / _5

Prototype:

```
SRC_ADDR_X / _Y / _Z / _4 / _5
                                   uint      Position
                                   datatype  Datatype
```

Parameters:

uint	Position	Position of the address of the input quantity in the deposit structure.
datatype	Datatype	Data type of the address.

Description:

Description of the address of the input quantity in an adjustable object

Example:

```
  SRC_ADDR_X                1
                            UWORD
```

3.5.117 STATIC_RECORD_LAYOUT

Prototype:

```
STATIC_RECORD_LAYOUT
```

Description:

This keyword is used to indicate that an adjustable object with dynamic number of axis points **does not** compact or expand data when removing resp. inserting axis points. All record layout elements are stored at the same address as for the max. number of axis points specified at the calibration object - independent of the current number of axis points
The FNC_VALUES are handled as static as well. I.e., the addresses of the single data cells do not change if the dimension of the map changes.

Example 1 (requires STATIC_RECORD_LAYOUT):

```
struct {
   unsigned short noX;      /* number of x axis coordinates */
   unsigned short noY;      /* number of y axis coordinates */
   unsigned char xAxis[10]; /* MaxAxisPoints = 10 */
   unsigned char yAxis[5];  /* MaxAxisPoints = 5 */
   unsigned char values[10][5];
} mapBeingNotCompact;

#define GetXCoordinate(map, index) (map.xAxis[index])
#define GetYCoordinate(map, index) (map.yAxis[index])
#define GetMapValue(map, xIndex, yIndex) (map.value[xIndex][yIndex])
```

This example leads to the following description:

```
/begin CHARACTERISTIC
   mapBeingNotCompact "This curve compacts data"
   MAP
   0x0815
   mapLayoutNotCompact
   0.0
   NO_COMPU_METHOD
   0.0
   255.0
   /begin AXIS_DESCR
      STD_AXIS
      NO_INPUT_QUANTITY
      NO_COMPU_METHOD
      10
      0 200
   /end AXIS_DESCR
   /begin AXIS_DESCR
      STD_AXIS
      NO_INPUT_QUANTITY
      NO_COMPU_METHOD
      5
      0 200
   /end AXIS_DESCR
/end CHARACTERISTIC
```

```
/begin RECORD_LAYOUT
   mapLayoutNotCompact
   NO_AXIS_PTS_X 1 UWORD
   NO_AXIS_PTS_Y 2 UWORD
   AXIS_PTS_X 3 UBYTE INDEX_INR DIRECT
   AXIS_PTS_Y 4 UBYTE INDEX_INR DIRECT
   FNC_VALUES 5 UBYTE ROW_DIR DIRECT
   STATIC_RECORD_LAYOUT
/end RECORD_LAYOUT
```

Example 2 (does NOT require STATIC_RECORD_LAYOUT):

```
stuct mapBeingCompact_t {
   unsigned short noX;          /* number of x axis coordinates */
   unsigned short noY;          /* number of y axis coordinates */
   unsigned char values[65];    /* 10 x points, 5 y points  */
                                /* 10x5 map values */
} mapBeingCompact;

#define GetXCoordinate(map, index) (map.values[index])
#define GetYCoordinate(map, index) (map.values[map.noX + index])
#define GetMapValue(map, xIndex, yIndex) \
    (map.values[map.noX + map.noY + xIndex + (yIndex * map.noX)])
```

This example leads to the following description:

```
/begin CHARACTERISTIC
   mapBeingCompact "This curve doesn't compact data"
   MAP
   0x0815
   mapLayoutCompact
   0.0
   NO_COMPU_METHOD
   0.0
   255.0
   /begin AXIS_DESCR
      STD_AXIS
      NO_INPUT_QUANTITY
      NO_COMPU_METHOD
      10
      0 200
   /end AXIS_DESCR
   /begin AXIS_DESCR
      STD_AXIS
      NO_INPUT_QUANTITY
      NO_COMPU_METHOD
      5
      0 200
   /end AXIS_DESCR
/end CHARACTERISTIC

/begin RECORD_LAYOUT
   mapLayoutCompact
   NO_AXIS_PTS_X 1 UWORD
   NO_AXIS_PTS_Y 2 UWORD
```

```
      AXIS_PTS_X 3 UBYTE INDEX_INR DIRECT
      AXIS_PTS_Y 4 UBYTE INDEX_INR DIRECT
      FNC_VALUES 5 UBYTE ROW_DIR DIRECT
   /end RECORD_LAYOUT
```

3.5.118 STATUS_STRING_REF

Prototype:

```
STATUS_STRING_REF          ident     ConversionTable
```

Parameters:

ident ConversionTable	Reference to a verbal conversion table (COMPU_VTAB or COMPU_VTAB_RANGE)

Description:

This keyword is used to split up the value range of ECU internal values into a numerical and a verbal part. The verbal part can be used to visualize status information (e.g. “Sensor not connected”). It is used at COMPU_METHOD.

The conversion table referenced by the keyword STATUS_STRING_REF must not define a default value.

Note: The MC tool at first checks whether the internal value is in the range defined at the STATUS_STRING_REF conversion table. In this case, the tool displays the corresponding text. Otherwise, the MC tool uses the regular computation method.
The MC tool must not respect the limits when evaluating the STATUS_STRING_REF.

Note: PHYS Values defined by STATUS_STRING_REF are not selectable for calibration. To ensure this, the values referenced by STATUS_STRING_REF must be outside the ECU internal limits of all calibration objects (CHARACTERISTIC, AXIS_PTS) using the corresponding conversion method.

Example:

```
/begin COMPU_METHOD      CM_LINFUNC_SENSOR_A      /* name */
                         "conversion method for Sensor A"
                         LINEAR           /* convers_type */
                         "%4.0"           /* display format */
                         "rpm"            /* physical unit */
                         COEFFS_LINEAR  2.0 5.0
                         STATUS_STRING_REF CT_SensorStatus
/end COMPU_METHOD

/begin COMPU_VTAB        CT_SensorStatus
                         ""
                         TAB_VERB         /* convers_type */
                         2                /* number_value_pairs */
                         0x00 "Sensor not Connected"
                         0xFF "Sensor defect"
/end COMPU_VTAB
```

3.5.119 STEP_SIZE

Prototype:

```
STEP_SIZE                    float     StepSize
```

Parameters:

float	StepSize	delta

Note: The values are to be interpreted as physical values.

Description:

This keyword can be used to define a value which is added to or subtracted from a CHARACTERISTIC, AXIS_PTS or AXIS_DESCR data value when using up/down keys while calibrating.

Example:

```
 STEP_SIZE                   0.025
```

3.5.120 SUB_FUNCTION

Prototype:

```
/begin SUB_FUNCTION        { ident   Identifier } *
/end SUB_FUNCTION
```

Parameters:

ident	Identifier	Reference to function record. This function record is declared as subfunction of respective function.

Description:

This keyword can be used to define the hierarchical structure of functions.
Note: SUB_FUNCTION may only refer to objects of type FUNCTION.

Example:

```
  /begin SUB_FUNCTION      ID_ADJUSTM_SUB
  /end SUB_FUNCTION
```

3.5.121 SUB_GROUP

Prototype:

```
/begin SUB_GROUP          { ident   Identifier } *
/end SUB_GROUP
```

Parameters:

ident	Identifier	Reference to a group record. This group record is declared as sub-group of the respective GROUP.

Description:

This keyword can be used to define the hierarchical structure of groups. In particular, a set of groups referenced from a root group (with optional keyword ROOT) constitute a grouping mechanism.

Example:

```
 /begin  SUB_GROUP        ID_ADJUSTM_SUB
 /end SUB_GROUP
```

3.5.122 SUPPLIER

Prototype:

```
SUPPLIER                    string   Manufacturer
```

Parameters:

string	Manufacturer	Name of the ECU manufacturer

Description:

String used to identify the manufacturer or supplier.

Example:

```
 SUPPLIER                   "Smooth and Easy"
```

3.5.123 SYMBOL_LINK

Prototype:

```
SYMBOL_LINK                string   SymbolName
                           long     Offset
```

Parameters:

string	SymbolName	Name of the symbol within the corresponding linker map file
long	Offset	Offset of the Symbol relative to the symbol's address in the linker map file

Description:

This keyword can be used to specify the name of a symbol within a linker map file that corresponds to the respective CHARACTERISTIC, MEASUREMENT, or AXIS_PTS of the A2L file. The offset parameter defines the offset of the element defined in the A2L file relative to the address of the corresponding symbol of the linker map file (e.g. for arrays).
Using this information, an automatic address update can be performed according to a provided linker map file.

Example:

```
 SYMBOL_LINK                "_VehicleSpeed"    /* Symbol name */
                            0                  /* Offset */
```

3.5.124 SYSTEM_CONSTANT

Prototype:

```
SYSTEM_CONSTANT               string    Name
                              string    Value
```

Parameters:

string	Name	system constant identifier
string	Value	value of the system constant as a string

Description:

System-defined constant.

Note: If the system constant shall be used in conversion formulas the name of the system constant has to be unique within all system constants of the MODULE, i.e. there must not be another system constant with the same name in the MODULE. Furthermore, the name must comply to the restrictions of data type ident and the value of the system constant must contain either a numerical value or a string that contains a further formula part.

Example:

```
  SYSTEM_CONSTANT             "CONTROLLER_CONSTANT12"
                              "2.7134"
```

3.5.125 S_REC_LAYOUT

Prototype:

```
S_REC_LAYOUT                ident     Name
```

Parameters:

ident Name	Name of the standard record layout (see RECORD_LAYOUT)

Description:

This keyword can be used to specify the name of a standard record layout which will then apply to all characteristics in the entire module. Exceptions can be specified for the relevant characteristics.

Note: S_REC_LAYOUT may only refer to objects of type RECORD_LAYOUT.

Example:

```
  S_REC_LAYOUT            S_ABL         /* record layout */
```

3.5.126 UNIT

Prototype:

```
/begin UNIT                 ident    Name
                            string   LongIdentifier
                            string   Display
                            enum     Type
                            [-> REF_UNIT]
                            [-> SI_EXPONENTS]
                            [-> UNIT_CONVERSION]
/end UNIT
```

Parameters:

ident	Name	identifier in the program, referencing is based on this name Note: The name of the unit has to be unique within all units of the ASAM MCD-2MC MODULE, i.e. there must not be another UNIT object with the same identifier in the MODULE.
string	LongIdentifier	comment, description
string	Display	string to be used to display the measurement unit of a physical value
enum	Type	Type of the Unit: DERIVED: measurement unit derived from another one referenced by the optional parameter REF_UNIT. EXTENDED_SI: extended SI unit. Recommendation: The principle of describing "real" measurement units is to refer to SI units. Therefore, this relationship should be given by using the optional parameter SI_EXPONENTS.

Optional Parameters:

-> SI_EXPONENTS	This keyword is used to specify the exponents of the seven base dimensions required to define an extended SI unit.
-> REF_UNIT	This keyword is used to reference to another measurement unit from which the one using REF_UNIT is derived.
-> UNIT_CONVERSION	This keyword is used to specify the linear relationship between two measurement units.

Description:

Specification of a measurement unit.

Example:

```
/begin UNIT
                        metres_per_second
                        "extended SI unit for velocity"
                        "[m/s]"
                        EXTENDED_SI
        SI_EXPONENTS    1 0 -1 0 0 0 0     /* [m] * [s]^-1 */
/end UNIT

/begin UNIT
                        kms_per_hour
                        "derived unit for velocity: kilometres per
                        hour"
                        "[km/h]"
                        DERIVED
            REF_UNIT    metres_per_second
     UNIT_CONVERSION    3.6  0.0     /* y [km/h] = (60*60/1000) * x
                                     [m/s] + 0.0 */
/end UNIT

/begin UNIT
                        miles_per_hour
                        "derived unit for velocity: miles per hour"
                        "[mph]"
                        DERIVED
            REF_UNIT    metres_per_second
     UNIT_CONVERSION    2.237  0.0   /* y [mph] = (60*60/1609) * x
                                     [m/s] + 0.0 */
/end UNIT
```

3.5.127 UNIT_CONVERSION

Prototype:

```
UNIT_CONVERSION                float     Gradient
                               float     Offset
```

Parameters:

float	Gradient, Offset	gradient and offset of the linear relationship between two measurement units: f(x) = gradient * x + offset

Description:

Specification of the linear relationship between two measurement units given by describing the conversion from the referenced unit to the derived unit:

derived_unit = f(referenced_unit)

The referenced measurement unit had to be specified with parameter REF_UNIT.

Example:

```
/begin UNIT
                         kelvin
                         "base unit for temperature: Kelvin"
                         "[K]"
                         EXTENDED_SI
        SI_EXPONENTS     0 0 0 0 1 0 0
/end UNIT

/begin UNIT
                         degC
                         "unit for temperature: degree Celsius"
                         "[°C]"
                         DERIVED
            REF_UNIT     kelvin
     UNIT_CONVERSION     1.0 -273.15  /* y [°C] = 1.0 * x [K] + (-
                                      273.15) */
/end UNIT
```

3.5.128 USER

Prototype:

```
USER                      string    UserName
```

Parameters:

string	UserName	Name of the user

Description:

Specification of the user name.

Example:

```
 USER                     "Nigel Hurst"
```

3.5.129 USER_RIGHTS

Prototype:

```
/begin USER_RIGHTS          ident    UserLevelId
                            [-> READ_ONLY]
                            {-> REF_GROUP}*
/end USER_RIGHTS
```

Parameters:

ident UserLevelId	When a user logs into the MCD system, a UserLevelId is assigned.

Optional parameters:

-> REF_GROUP:	Reference to groups. Only the CHARACTERSITIC and MEASUREMENT members of the referenced groups including the members of nested subgroups (and functions nested in such groups) are visible to the user of the MCD system. If the READ_ONLY attribute is set, the CHARACTERISTICs are visible but not available for calibration (not tunable). Note: The restrictions are applied by the MCD system as a global filter in the user interface (active for all manual selection or calibration operations). When navigating by GROUPs, only the GROUPs declared in USER_RIGHTS need to be provided in the selection list.
-> READ_ONLY:	This keyword can be used to define all characteristics of the groups referenced by this USER_RIGHT statement as READ_ONLY (not tunable). Use Case : A group can be defined to specify a set of characteristics to be not tunable for a group of users (control of access rights). In order to achieve this, the group is referenced in a USER_RIGHT statement with the READ_ONLY attribute, related to the user group. When a login to the MCD system identifies the user as member of a group for which the USER_RIGHT statement contains the READ_ONLY attribute, all CHARACTERISTICs of this group shall be treated as if the READ_ONLY attribute was directly related to the CHARACTERISTICs.

Description:

This keyword can be used to define groups accessible for certain users. All USER-RIGHTS groups are listed to the user who can select one of these groups. All measurements and characteristics belonging to that group and its subgroups (and sub subgroups and so on) are accessible (i.e. visible) to the user. The keyword READ_ONLY is used to define the referred group(s) as containing characteristics that are only readable but not writeable (i.e. they can not be adjusted). This property is also inherited by subgroups, i.e. if a group is marked as READ_ONLY all its subgroups (with respect to that USER RIGHT) are also only READ_ONLY.

Example:

```
/begin  USER_RIGHTS     calibration_engineers
  /begin REF_GROUP      group_1
  /end REF_GROUP
/end  USER_RIGHTS

/begin  USER_RIGHTS     measurement_engineers
  /begin REF_GROUP      group_1
  /end REF_GROUP
  READ_ONLY
/end USER_RIGHTS

/begin  GROUP           group_1
  /begin REF_CHARACTERISTIC
                        KF1
                        KF2
  /end REF_CHARACTERISTIC
  /begin REF_MEASUREMENT
                        NMOT
                        TMOT
  /end REF_MEASUREMENT
/end  GROUP
```

3.5.130 VAR_ADDRESS

Prototype:

```
/begin VAR_ADDRESS          { ulong   Address}*
/end VAR_ADDRESS
```

Parameters:

ulong Address	Start address of one variant of variant coded adjustable object.

Description:

This keyword can be used to define a list of start addresses of variant coded adjustable objects (see keyword VAR_CHARACTERISTIC). The number of addresses agrees with number of valid combinations of adjustable objects variant criteria (forbidden combinations excluded). The order of addresses corresponds to the order of variant criteria defined with parameter 'CriterionName' at keyword VAR_CHARACTERISTIC. The priority of index increment is according to the following rules:

- the priority of index increment is inverse to the order of variant criteria definition at keyword VAR_CHARACTERISTIC, e.g.:
- the first variant criterion has the lowest priority
- the last variant criterion has the highest priority

The following example describes the order of addresses of an adjustable object depending on three variant criterions with 'L', 'N', and 'M' criterion values:

Example:

Crit1 = { $Val_{1,1}$, $Val_{1,2}$, ...$Val_{1,L}$)
Crit2 = { $Val_{2,1}$, $Val_{2,2}$, ...$Val_{2,M}$)
Crit3 = { $Val_{3,1}$, $Val_{3,2}$, ...$Val_{3,N}$)

Corresponding address list:
Address[0] = Address ($Val_{1,1}$, $Val_{2,1}$, $Val_{3,1}$)
Address[1] = Address ($Val_{1,1}$, $Val_{2,1}$, $Val_{3,2}$)
:

Address[N - 1]	= Address ($Val_{1,1}$, $Val_{2,1}$, $Val_{3,N}$)
Address[N]	= Address ($Val_{1,1}$, $Val_{2,2}$, $Val_{3,1}$)
Address[N + 1]	= Address ($Val_{1,1}$, $Val_{2,2}$, $Val_{3,2}$)
	:
Address[N + N - 1]	= Address ($Val_{1,1}$, $Val_{2,2}$, $Val_{3,N}$)
	:

Example:

```
/begin VAR_ADDRESS
                         0x8840
                         0x8858
                         0x8870
                         0x8888
/end VAR_ADDRESS
```

3.5.131 VAR_CHARACTERISTIC

Prototype:

```
/begin VAR_CHARACTERISTIC    ident  Name
                             { identCriterionName }*
                             [-> VAR_ADDRESS]
/end VAR_CHARACTERISTIC
```

Parameters:

ident Name	Identifier of variant coded adjustable object (refers to CHARACTERISTIC or AXIS_PTS record).
ident CriterionName	Corresponding to each combination of variant criteria defined with this parameter the control unit software contains variants of concerning adjustable object.

Optional Parameters:

-> VAR_ADDRESS	Definition of start address of adjustable objects variants.

Description:

This keyword defines one adjustable object to be variant coded, i.e. this adjustable objects is multiple deposited in control unit software corresponding to the assigned variant criteria. The number of variants results on valid combinations (forbidden combinations excluded) of variant criteria.

Note: If an AXIS_PTS object is variant coded, the curves and maps using this common axis must be variant coded in the same way, i.e. they have to refer the same variant criterions in the same order.

Example:

```
/begin VAR_CHARACTERISTIC      /* define NLLM as variant coded */
                        NLLM
                        Gear  Car
  /* gear box including the 2 variants "Manual" and "Automatic" */
  /* car body including the 3 variants "Limousine",  "Kombi" and
  "Cabrio" */

  /* four addresses corresponding to the four valid combinations */
  /* of criterion 'Gear' and 'Car' (see example for VAR_CRITERION)*/
  /begin VAR_ADDRESS
                        0x8840
                        0x8858
                        0x8870
                        0x8888
  /end VAR_ADDRESS
/end VAR_CHARACTERISTIC
```

3.5.132 VAR_CRITERION

Prototype:

```
/begin VAR_CRITERION       ident    Name
                           string   LongIdentifier
                           {ident   Value }*
                           [-> VAR_MEASUREMENT]
                           [-> VAR_SELECTION_CHARACTERISTIC]
/end VAR_CRITERION
```

Parameters:

ident	Name	Identifier of variant criterion.
string	LongIdentifier	Comment to describe the variant criterion.
ident	Value	Enumeration of criterion values.

Optional Parameters:

-> VAR_MEASUREMENT	This keyword can be used to specify a special measurement object. This measurement object indicates with its current value the variant which has effect on running control unit software.
-> VAR_SELECTION_CHARACTERISTIC	This keyword is used to specify a special characteristic to change the variant of software which is running on control unit.

Description:

This keyword describes a variant criterion, i.e. some adjustable objects are multiple deposited in control unit software corresponding to the enumeration of variant criterion values.

Example:

```
/* variant criterion "Car body" with three variants */
/begin VAR_CRITERION                Car
                                    "Car body"
                                    /*Enumeration of criterion values*/
                                    Limousine  Kombi  Cabrio
              VAR_MEASUREMENT       S_CAR
 VAR_SELECTION_CHARACTERISTIC       V_CAR
/end VAR_CRITERION
```

3.5.133 VAR_FORBIDDEN_COMB

Prototype:

```
/begin VAR_FORBIDDEN_COMB {ident    CriterionName
                           ident    CriterionValue }*
/end VAR_FORBIDDEN_COMB
```

Parameters:

ident	CriterionName	Identifier of variant criterion.
ident	CriterionValue	Value of variant criterion ' CriterionName '.

Description:

This keyword describes a forbidden combination of values of different variant criteria.

Example:

```
/* forbidden variant combination (doesn't exist in control unit software): */
/begin VAR_FORBIDDEN_COMB
   Car Limousine  /* variant value 'Limousine' of criterion 'Car'  */
   Gear Manual    /* variant value 'Manual' of criterion 'Gear'  */
/end VAR_FORBIDDEN_COMB
```

3.5.134 VAR_MEASUREMENT

Prototype:

```
VAR_MEASUREMENT              ident     Name
```

Parameters:

ident Name	Identifier of measurement object which indicates the current criterion value. This parameter refers to a MEASUREMENT record of description file.

Description:

This keyword can be used to specify a special measurement object. This measurement object indicates with its current value the variant which has effect on running control unit software. To be able to map the current object value to the variant the referenced measurement object must have a conversion method of type COMPU_VTAB and the strings defined at the conversion table must correspond to the criterion values at the VAR_CRITERION record.

Note: VAR_MEASUREMENT may only refer to objects of type MEASUREMENT.

Example:

```
/begin COMPU_VTAB       V_GEAR_BOX
                        "variants of criterion ""Type of Gear Box"""
                        3
                        17 "Limousine"
                        39 "Kombi"
                        41 "Cabrio"
                        DEFAULT_VALUE "unknown"
/end COMPU_VTAB


/begin VAR_CRITERION    Car
                        "Car body"
                        Limousine  Kombi  Cabrio
      VAR_MEASUREMENT   S_GEAR_BOX
/end VAR_CRITERION
```

3.5.135 VAR_NAMING

Prototype:

```
VAR_NAMING                    enum      Tag
```

Parameters:

enum Tag	Format of variant extension (index). Possible values: NUMERIC variant extension is a number (integer: 0,1,2,3...). This parameter is reserved for future extension (e.g. ALPHA = { A, B, C, D....}).

Description:

This keyword defines the format of variant extension (index) of adjustable objects name. The extension is used at MCD to distinguish the different variants of adjustable objects.

Example:

```
/* variant extension: see example VAR_CRITERION*/
VAR_NAMING                 NUMERIC
```

3.5.136 VAR_SELECTION_CHARACTERISTIC

Prototype:

```
VAR_SELECTION_CHARACTERISTIC        ident     Name
```

Parameters:

ident Name	Identifier of characteristic object which indicates the current criterion value. This parameter refers to a CHARACTERISTIC record of description file.

Description:

This keyword can be used to specify a special characteristic object. This characteristic object changes with its current value the variant which has effect on running control unit software.
To be able to map the current object value to the variant the referenced characteristic object must have a conversion method of type COMPU_VTAB and the strings defined at the conversion table must correspond to the criterion values at the VAR_CRITERION record.

Note: VAR_SELECTION_CHARACTERISTIC may only refer to objects of type CHARACTERISTIC.

Example:

```
/begin COMPU_VTAB        V_GEAR_BOX
                         "variants of criterion ""Type of Gear Box"""
                         3
                         17 "Limousine"
                         39 "Kombi"
                         41 "Cabrio"
                         DEFAULT_VALUE "unknown"
/end COMPU_VTAB

/begin VAR_CRITERION            Car
                                "Car body"
                                Limousine  Kombi  Cabrio
VAR_SELECTION_CHARACTERISTIC    S_GEAR_BOX
/end VAR_CRITERION
```

3.5.137 VAR_SEPARATOR

Prototype:

```
VAR_SEPARATOR                string    Separator
```

Parameters:

string	Separator	This parameter defines the separating symbol of variant extension.

Description:

This keyword can be used to define the separating symbol between the two parts of adjustable objects name: 1.) identifier 2.) variant extension.

Note: The identifier of description record of variant coded adjustable objects contains no variant extension. The extension is needed to distinguish the variants at MCD.

Example:

```
VAR_SEPARATOR                 "."     /* example: "PUMKF.1"  */
/* three parts of variant coded adjustable objects name:     */
/* 1.) Identifier of adjustable object:    "PUMKF"     */
/* 2.) Separator:                "." (decimal point)      */
/* 3.) Variants extension:    "1"     */
```

3.5.138 VARIANT_CODING

Prototype:

```
/begin VARIANT_CODING
                         {-> VAR_CHARACTERISTIC}*
                         {-> VAR_CRITERION}*
                         {-> VAR_FORBIDDEN_COMB}*
                         [-> VAR_NAMING]
                         [-> VAR_SEPARATOR]
/end VARIANT_CODING
```

Optional Parameters:

-> VAR_CHARACTERISTIC	This keyword defines one adjustable object to be variant coded, i.e. this adjustable objects is multiple deposited in control unit software corresponding to the assigned variant criteria.
-> VAR_CRITERION	This keyword describes a variant criterion, i.e. some adjustable objects are multiple deposited in control unit software corresponding to the enumeration of variant criterion values
-> VAR_FORBIDDEN_COMB	This keyword describes a forbidden combination of different variant criteria.
-> VAR_NAMING	This keyword defines the format of variant extension (index) of adjustable objects name (index is used at MCD to distinguish the variants
-> VAR_SEPARATOR	This keyword can be used to define the separating symbol between the two parts of adjustable objects name: 1.) identifier 2.) variant extension. Note: The identifier of description record of variant coded adjustable objects contains no variant extension. This extension is needed to distinguish the variants at MCD

Description:

The information of variant coding is grouped to this keyword. Variant coding means, that control unit software contains several variants (copies) of some adjustable objects, whereas description file contains only one record to describe. In a real ECU only one variant is in use, depending on car-specific parameters.

Example:

```
 /begin VARIANT_CODING
         VAR_SEPARATOR    "."         /* PUMKF.1 */
         VAR_NAMING       NUMERIC

   /* variant criterion  "Car body" with three variants */
   /begin VAR_CRITERION  Car
                         "Car body"
         Limousine Kombi Cabrio
   /end VAR_CRITERION
   /* variant criterion "Type of gear box" with two variants */
   /begin VAR_CRITERION  Gear
         "Type of gear box"
```

```
        Manual  Automatic
  /end VAR_CRITERION

  /begin VAR_FORBIDDEN_COMB         /* forbidden: Limousine-Manual*/
                      Car Limousine
                      Gear Manual
  /end VAR_FORBIDDEN_COMB
  /begin VAR_FORBIDDEN_COMB         /* forbidden: Cabrio-Automatic*/
                      Car Cabrio
                      Gear Automatic
  /end VAR_FORBIDDEN_COMB
  /begin VAR_CHARACTERISTIC
                      PUMKF      /*define PUMKF as variant coded*/
                      Gear       /* Gear box variants */
    /begin VAR_ADDRESS
                      0x7140
                      0x7168
  /end VAR_ADDRESS
  /end VAR_CHARACTERISTIC
  /begin VAR_CHARACTERISTIC
                      NLLM       /*define NLLM as variant coded */
                      Gear  Car  /*car body and gear box
                                  variants*/
    /begin VAR_ADDRESS
                      0x8840
                      0x8858
                      0x8870
                      0x8888
      /end VAR_ADDRESS
  /end VAR_CHARACTERISTIC
/end VARIANT_CODING
```

Table 12 Example of NLLM - variants coding

Type of gear box / Car body	MANUAL	AUTOMATIC
Limousine	doesn't exist	NLLM.3 (address = 0x8870)
Kombi	NLLM.1 (address = 0x8840)	NLLM.4 (address = 0x8888)
Cabrio	NLLM.2 (address = 0x8858)	doesn't exist

Table 13 Example of PUMKF - variants coding

Type of gear box	
MANUAL	**AUTOMATIC**
PUMKF.0 (address = 0x7140)	PUMKF.1 (address = 0x7168)

3.5.139 VERSION

Prototype:

```
VERSION                    string   VersionIdentifier
```

Parameters:

string	VersionIdentifier	short identifier for the version

Description:

String for identification of the version with maximum MAX_STRING characters.

Example:

```
 VERSION                    "BG5.0815"
```

3.5.140 VIRTUAL

Prototype:

```
/begin VIRTUAL              (ident    MeasuringChannel)*
/end VIRTUAL
```

Parameters:

ident	MeasuringChannel	Reference to a measurement (MEASUREMENT) or a virtual measurement (MEASUREMENT, VIRTUAL)

Description:

This keyword allows virtual measurements to be specified. For this, constants, measurements and virtual measurements can be combined into one quantity. The list specified with the VIRTUAL keyword indicates the quantities to be linked (reference). These quantities are combined into one measurement by means of a single conversion formula. The conversion formula must be capable of processing several input quantities.

The references used in the computation formula are named X1, X2, X3, … . The reference X1 references the first parameter of the attached parameter list, X2 the second, X3 the third, ….
If there is only one reference used it is allowed to use X instead of X1.

Example:

```
/begin MEASUREMENT       PHI_FIRING          /* Name */
                         "Firing angle"      /* Long identifier */
                         UWORD               /* Data type */
                         R_PHI_FIRING        /* Conversion */
                         1                   /* Resolution */
                         0.01                /* Accuracy */
                         120.0               /* Lower limit */
                         8400.0              /* Upper limit */
  /*Quantities to be linked: 2 measurements */
  /begin VIRTUAL         PHI_BASIS
                         PHI_CORR

  /end VIRTUAL
/end MEASUREMENT

/begin COMPU_METHOD      R_PHI_FIRING        /* Name */
                         "Addition of two measurements"
                         FORM                /* Convers_type */
                         "%4.2"              /* Display format */
                         "GRAD_CS"           /* physical unit */
  /begin FORMULA         "X1 + X2"           /* X1 -> PHI_BASIS */
                                             /* X2 -> PHI_CORR */

  /end FORMULA
/end COMPU_METHOD
```

3.5.141 VIRTUAL_CHARACTERISTIC

Prototype:

```
/begin VIRTUAL_CHARACTERISTIC       string   Formula
                                    (ident   Characteristic)*
/end VIRTUAL_CHARACTERISTIC
```

Parameters:

string	Formula	Formula to be used for the calculation of the initial physical value of the characteristic from the physical value of other characteristics.
ident	Characteristic	Identifier of those adjustable objects that are used for the calculation of this characteristic.

Description:

This keyword allows to define characteristics that are not deposited in the memory of the control unit, but can be used to indirectly calibrate other characteristic values in the control unit, if these are declared to be dependent on this characteristic. The introduction of virtual characteristic is therefore useful for saving memory in the case the calibration with dependent characteristics is used.

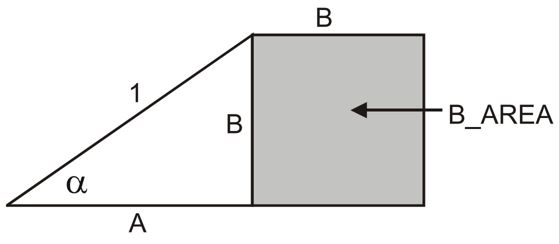


Figure 7 VIRTUAL_CHARACTERISTIC

For the initial value of the virtual characteristic must be derived from the values of other characteristics. The mechanism to implement this is the same as for dependent characteristics by a list of characteristics and a formula, e.g. α = asin(B). Also B might be virtual, i.e. its value has to be derived from B_AREA.
The following example makes clear how the calibration process takes place. When the virtual characteristic α is initialized, the value of α is calculated from the value of B. Therefore B_{Int} is read from the ECU and $B_{phys} = B_{Int}/100$ is computed. Assuming the value $B_{Int} = 80$, $B_{phys} = 0.8$ and $\alpha_{phys} = \text{asin}(B_{phys}) = 53.13$. Since virtual characteristics are not in the memory of an ECU, α_{Int} and α_{phys} may coincide if the datatype for α_{Int} is chosen an float datatype and the conversion formula is the identity (one to one formula).

The references used in the virtual dependency formula are named X1, X2, X3, … . The reference X1 references the first parameter of the attached parameter list, X2 the second, X3 the third, ….
If there is only one reference used it is allowed to use X instead of X1.

Example:

```
/begin VIRTUAL_CHARACTERISTIC
                        „sin(X1)“
                        B
/end VIRTUAL_CHARACTERISTIC

/* Example for ParamB - ParamA */
/begin DEPENDENT_CHARACTERISTIC
              "X2-X1"
              ParamA    /* is referenced by X1 */
              ParamB    /* is referenced by X2 */
/end DEPENDENT_CHARACTERISTIC
```

4 INCLUDE MECHANISM

For the description of projects involving several control units or measurement and calibration devices of various manufacturers the Include statement can be used.

/include <filename>

The parameter <filename> may include a relative or absolute path or an UNC path. Relative paths are relative to the location of the ASAM AE MC-2MC file containing the /include statement. The relative path uses backslashes without escape sequences.

This statement allows several description files to be integrated into one project description. The filename may be put between quotation marks. If the filename contains spaces or path information the quotation marks are required.

Example:

```
/******************File PROJECT1.A2L*******************************/
/begin PROJECT          RAPE-SEED ENGINE "Engine tuning for
                        operation with rape oil"
     /begin HEADER      "General project description"
        VERSION         "0815"
        PROJECT_NO      1188
     /end

     /include "C:\ENG_ECU.A2L"
     /include "..\includes\ABS_ECU.A2L"
     /include "SPEC_ECU.A2L"
     /include "\\MyServer\VariableDescriptions\ESP_ECU.A2L"
/end
/*****************End of file PROJECT1.A2L************************/
```

5 ASAM MCD-2MC METALANGUAGE

5.1 GENERAL

Between MCD System and the devices or ECU's an interface is used. The description data in the measurement and calibration system are divided into two categories:

1. Parameters that are used by the control interface.
2. Parameters that are only analyzed by the driver and whose meaning is hidden to the control interface (interface-specific parameters). They are transferred to the driver as a binary block.

These two measures should make it possible that new interface module types can be handled without having to introduce any changes in the control part of the measurement and calibration system but simply by incorporating a new driver.
For the description of the interface-specific parameters a description language (ASAM MCD-2MC metalanguage, in short AML) will be defined on the following pages. Each manufacturer can specify a special set of parameters for their own interface module types (format description). Using this format description (in AML) the standardized measurement and calibration system must be capable of reading in the *interface-specific parameters* of the description file and transferring them to the drivers.

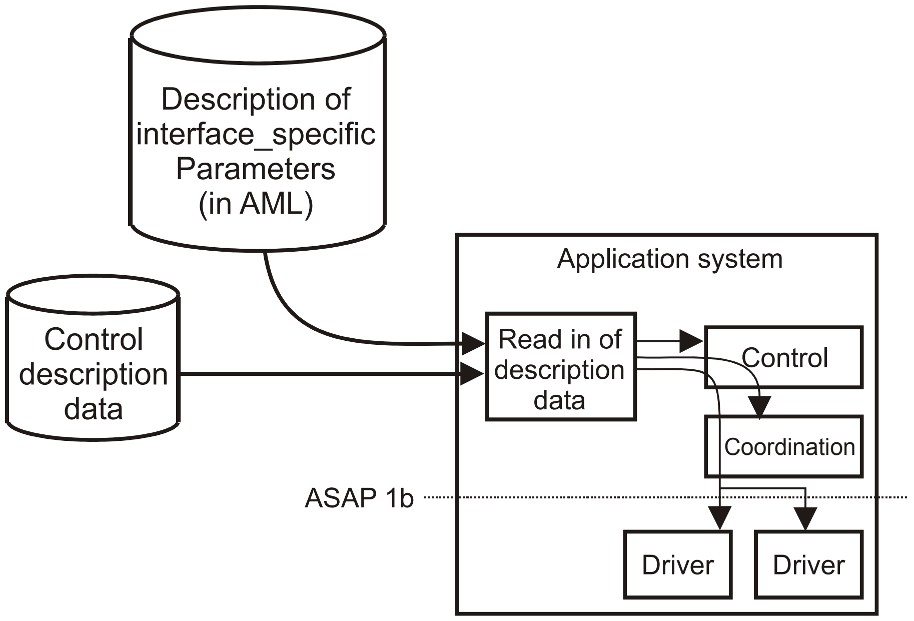

Description of interface_specific Parameters (in AML)
Control description data
Application system
Read in of description data
Control
Coordination
ASAP 1b
Driver
Driver

Figure 8 Schematic data flow of description data

For parameterization of the drivers and for access to the adjustable and measurement objects different parameters have to be used within the various drivers. The user interface, which does not need to know these parameters, in fact only requires a description of the data types to be able to read in these interface-specific parameters. This description, based on the ASAM MCD-2MC metalanguage described hereafter, occurs either INLINE within the description file, or in separate files. In the second case the Include statement can be used to integrate the description of interface- specific parameters:

```
    /begin MODULE ...
            /include <filename>
      ....
    /end MODULE
```

5.2 Format of the ASAM MCD-2MC Metalanguage

Table 14 BNF Terminology

Symbol	Meaning
<...>	Place marker
[...]	optional part
|	or (in the exclusive sense)

To describe the grammar of the ASAM MCD-2MC metalanguage, an extended Backus-Naur format is used:

The non-terminals are listed in the first column of Table 15 Grammar in the extended Backus-Naur format, the production rules are given in the second column.

A terminal (keyword) is written bold and is enclosed within quotes "**struct**".

An optional part is enclosed within square brackets: **[** identifier **].**

Alternative parts are separated by a pipe symbol: "char" **|** "int" **|** "long".

Non-formally defined parts are in italics. These are in particular:

-	*tag:*	is used to define a keyword of the ASAM MCD-2MC metalanguage by means of a character sequence enclosed within double inverted commas.
-	*identifier:*	identifier for the definition of data structures.
-	*keyword:*	an identifier for an enumeration
-	*constant:*	a numerical constant

Note: User defined *tags* and *keywords* must meet the rules for an ident (see 3.2: Predefined data types). Within the AML own name spaces are used. In this case it is allowed to reuse ASAM MCD-2MC keyword names.

Table 15 Grammar in the extended Backus-Naur format

Non-terminals	Production rule	Explanation
Declaration	type_definition ";" | block_definition ";"	definition of a data structure to be used for defining a data record of the description file.
type_definition	type_name	type definition
type_name	predefined_type_name |	type name

Non-terminals	Production rule	Explanation
	struct_type_name | taggedstruct_type_name | taggedunion_type_name | enum_type_name	
predefined_type_name	**"char"** | **"int"** | **"long"** | **"uchar"** | **"uint"** | **"ulong"** | **"double"** | **"float"**	predefined type name
block_definition	**"block"** *tag* type_name	definition of a block. A block consists of a special begin keyword ("/begin"), a keyword identifying the record type (e.g. "FUNCTION_LIST") , the relevant data record and an end keyword ("/end"). Nested blocks are also possible.
enum_type_name	**"enum"** [*identifier*] **"{"** enumerator_list **"}"** | **"enum"** *identifier*	definition of an enumeration
enumerator_list	Enumerator | Enumerator **","** enumerator_list	list of enumeration values
Enumerator	*keyword* [**"="** *constant*]	an enumeration
struct_type_name	**"struct"** [*identifier*] **"{"** [struct_member_list] **"}"** | **"struct"** *identifier*	definition of data records of the ASAM MCD-2MC description file with fixed sequence of the data record elements.
struct_member_list	struct_member | struct_member struct_member_list	list of structure members
struct_member	member **";"**	member of the structure
member	type_name [array_specifier]	a member of a data type

Non-terminals	Production rule	Explanation
array_specifier	**"["** *constant* **"]"** | **"["** *constant* **"]"** array_specifier	the length of an array
taggedstruct_type_name	"**taggedstruct**" [*identifier*] "**{**" [taggedstruct_member_list] "**}**" | "**taggedstruct**" *identifier*	definition of data records of the ASAM MCD-2MC description file whose elements can specified in a random sequence. All elements are optional and each element is identified by its tag. For the description of lists with a variable number of elements, the symbols "(" and ")*" are used. The sequences identified by these symbols can be repeated any number of times.
taggedstruct_member_list	taggedstruct_member | taggedstruct_member taggedstruct_member_list	list of members of a taggedstruct
taggedstruct_member	taggedstruct_definition ";" | "**(**" taggedstruct_definition "**)***;" | block_definition ";" "**(**" block_definition "**)***;"	a member of a taggedstruct
taggedstruct_definition	*tag* [member] | *tag* "**(**" member "**)***;"	an entry of a taggedstruct
taggedunion_type_name	"**taggedunion**" [*identifier*] "**{**" [taggedunion_member_list] "**}**" | "**taggedunion**" *identifier*	definition of variants in data records of the ASAM MCD-2MC description file. Similar to the 'union' data type used in programming language C, the ASAM MCD-2MC description file allows only one variant to be specified at a time in a 'taggedunion'. Each variant is assigned a tag for identification purposes (see *tag*).

Non-terminals	Production rule	Explanation
taggedunion_member_list	tagged_union_member | tagged_union_member taggedunion_member_list	list of members of a taggedunion
tagged_union_member	*tag* [member] ";" | block_definition ";"	a member of a taggedunion

Hint: "**(**" <Content> "**)*;**" describes a sequence with base type <Content>. The "**(**" and "**)*;**" are not symbols of the BNF but terminals.

5.3 DESIGNING AML-FILE

This chapter describes how to design an AML-file for interface-specific data.
To be compatible with ASAM MCD-2MC a tag ”IF_DATA” must be defined in the AML-file. This tag is then used by the MCD tool to interpret the data that is written in the various IF_DATA-fields in the ASAM MCD-2MC file. Template for AML-file shows a example that preferably should be used to design a AML-file for interface-specific data.

6 APPENDIXES

A TEMPLATE FOR AML-FILE

```
/begin A2ML
/* template.aml ************************************************************/
/*                                                *//*                                                                       */
/* Template for designing IF_DATA fields for ASAM MCD-2MC files and BLOB's      */
/* for driver interface.                                                        */
/* *************************************************************************/
block "IF_DATA" taggedunion if_data
{
   "ASAP1B_EXAMPLE"      /* The tag of ASAP1B is reserved for ASAM Interfaces      */
                         /* EXAMPLE shall be substituted with a name of            */
                         /* manufacturer's choice.                                 */

   taggedstruct          /* optional parameters  */

      (block "SOURCE" struct
      {
         struct          /*  indispensable */
         {
            char [101];  /* source name (string)*/
            int;         /* min period ( conforming together with min factor */
                         /* the fastest samplingrate available ).           */
            long;        /* min factor */
         };
         taggedstruct    /* optional parameters  */
         {
            block "QP_BLOB" struct      /* QP_BLOB for driver */
            {
               /* QP_BLOB specification  */
            };
         };
      }
      )*;                /*  multiple SOURCE may exist */

      block "TP_BLOB" struct            /* TP_BLOB for driver */
      {
         /* TP_BLOB specification  */
      };

      block "DP_BLOB" struct            /* DP_BLOB for driver */
      {
                         /* DP_BLOB specification  */
      };

      block "PA_BLOB" struct            /* PA_BLOB for driver */
      {
         /* PA_BLOB specification  */
      };
```

```
        block "KP_BLOB" struct            /* KP_BLOB for driver */
        {
           /* KP_BLOB specification  */
        };

        /*  for MODULE          may only TP_BLOB and SOURCE be specified            */
        /*  for CHARACTERISTIC  may only DP_BLOB and PA_BLOB be specified           */
        /*  for AXIS_PTS        may only DP_BLOB and PA_BLOB be specified           */
        /*  for MEMORY_LAYOUT   may only DP_BLOB and PA_BLOB be specified           */
        /*  for MEASUREMENT     may only KP_BLOB, DP_BLOB and PA_BLOB be specified  */
     };

     /* Extra tags can be defined here */

  };

  /************************************************************************/
  /end A2ML
```

B EXAMPLE OF DESCRIPTION FILE

B.1 SUPP1_IF.AML

```
/begin A2ML
/* A2ML-file defining the interface DIM.       */
/* ******************************************** */

enum mem_typ   { "INTERN" = 0, "EXTERN" = 1 };
enum addr_typ  { "BYTE" = 1, "WORD" = 2, "LONG" = 4 };
enum addr_mode { "DIRECT" = 0, "INDIRECT" = 1 };

taggedunion IF_DATA {
  "DIM" taggedstruct {      /* optional parameters  */
  (block "SOURCE" struct  {
                    struct {          /*  indispensable */
                          char [101]; /* source name (string)*/
                          int;        /* min period ( conforming
                                      together with min factor the
                                      fastest */
                                      /* ...samplingrate available ). */
                          long;       /* min factor */
                          };
                    taggedstruct {
                          block "QP_BLOB" struct  {
                                ulong;/* adr_distab */
                                int;  /* len_distab */
                                ulong;/* addr_outp  */
                                long; /* trgid      */
                                };
                          };
                          }
                    )*;               /* multiple SOURCE */

  block "TP_BLOB" struct    {
                          int;        /* display table type */
                          };

        block "KP_BLOB" struct {  /* KP_BLOB specification for Driver */
                          ulong;            /* address   */
                          enum addr_typ;    /* addr_size */
                          };

        block "DP_BLOB" struct {  /* DP_BLOB specification for Driver */
                          enum mem_typ;     /* mem_typ   */
                          };
        block "PA_BLOB" struct {  /* PA_BLOB specification for Driver */
                          enum addr_mode;   /* addressing mode */
                          };
  /* additional data types could be defined here*/
                    };
}; /* end of:  taggedunion if_data */
/end A2ML
```

B.2 MST_ABS.A2L

```
/begin PROJECT              MST_ABS  "Project example"
    /begin HEADER           "General project description"
        VERSION             "0815"
        PROJECT_NO         _1188
    /end HEADER

    /include engine_ecu.a2l
/end PROJECT
```

B.3 ENGINE_ECU.A2L

```
/begin MODULE  DIM  "Comment on module"
                                /* Detailed description of a device */

    /include "supp1_if.aml"     /* Specification of the interface-specific parts */

    /begin MOD_PAR              "Comment"
        VERSION                 "Test version 09.11.93"
        ADDR_EPK                0x12345
        EPK                     "EPROM identifier test"
        SUPPLIER                "Mustermann"
        CUSTOMER                "LANZ-Landmaschinen"
        CUSTOMER_NO             "0987654321"
        USER                    "Ignaz Lanz"           /* calibration engineer */
        PHONE_NO                "(01111) 22222"
        ECU                     "Engine control"
        CPU_TYPE                "Intel 0815"
        NO_OF_INTERFACES        2
        /begin MEMORY_LAYOUT    PRG_DATA  0x0000  0x8000   -1   -1   -1   -1   -1
   /begin IF_DATA DIM
          /begin DP_BLOB EXTERN /end DP_BLOB         /* memory type */
          /begin PA_BLOB DIRECT /end PA_BLOB         /* addressing mode */
   /end IF_DATA
        /end MEMORY_LAYOUT
        SYSTEM_CONSTANT         "CONTROLLERx CONSTANT1"   "0.99"
        SYSTEM_CONSTANT         "CONTROLLERx CONSTANT2"   "2.88"
        SYSTEM_CONSTANT         "CONTROLLERx CONSTANT3"   "-7"
        SYSTEM_CONSTANT         "ANY-PARAMETER"           "3.14159"
    /end MOD_PAR


     /begin MOD_COMMON          "Characteristic maps always deposited in same mode"
        DEPOSIT                 ABSOLUTE
        BYTE_ORDER              MSB_LAST
        DATA_SIZE               16                     /* bit */
    /end MOD_COMMON

    /begin IF_DATA DIM
   /begin SOURCE "angular synchonous" 101 1
          /begin QP_BLOB        0x5661  20   0xE001    2
          /end QP_BLOB
   /end SOURCE
   /begin SOURCE "time synchronous, rate 20ms" 4 2
          /begin QP_BLOB        0x3441  20   0xE041    3
          /end QP_BLOB
   /end SOURCE
   /begin TP_BLOB 14     /end TP_BLOB
    /end IF_DATA

    /begin CHARACTERISTIC       KI "I share for speed limitation"
                                VALUE                  /* type: constant */
                                0x0408                 /* address */
                                DAMOS_FW               /* deposit */
                                5.0                    /* max_diff */
                                FACTOR01               /* conversion */
```

```
                      0.0                  /* lower limit */
                      255.0                /* upper limit */

/* interface-spec. parameters: address location, addressing */
/begin IF_DATA  DIM
       /begin DP_BLOB EXTERN /end DP_BLOB  /* memory type */
       /begin PA_BLOB DIRECT /end PA_BLOB  /* addressing mode */
/end IF_DATA
/begin FUNCTION_LIST  V_LIM                /* reference to functions */
/end FUNCTION_LIST
 /end CHARACTERISTIC


  /begin CHARACTERISTIC      PUMCD   "Pump characteristic map"
                             MAP           /* type: characteristic map */
                             0x7140        /* address */
                             DAMOS_KF      /* deposit */
                             100.0         /* max_diff */
                             VOLTAGE       /* conversion */
                             0.0           /* lower limit */
                             5000.0        /* upper limit */


/begin IF_DATA  DIM
       /begin DP_BLOB EXTERN /end DP_BLOB  /* memory type */
       /begin PA_BLOB INDIRECT /end PA_BLOB       /* addressing mode */
/end IF_DATA
/begin AXIS_DESCR                                 /* X-axis: */
                             STD_AXIS             /* standard axis (no group or
                                                  fixed characteristic map) */
                             N                    /* input quantity */
                             N_RULE               /* conversion */
                             16                   /* maximum number of axis
                                                  points */
                             0.0                  /* lower limit */
                             5800.0               /* upper limit */
                             MAX_GRAD      20.0   /* max_grad */
/end AXIS_DESCR
/begin FUNCTION_LIST  CLDSTRT  FLLD        /* reference to functions */
/end FUNCTION_LIST
 /end CHARACTERISTIC

 /begin MEASUREMENT          M_ECORR
                             "corrected fuel mass"
                             UWORD                /* data type */
                             M_E                  /* reference to conversion
                                                  method */
                             1                    /* resolution in bits */
                             0.001                /* accuracy in '%' */
                             0.0                  /* lower limit */
                             43.0                 /* upper limit */
                             BIT_MASK  0x0ff      /* bit mask */

/begin IF_DATA   DIM
       /begin DP_BLOB EXTERN /end DP_BLOB         /* memory type */
       /begin PA_BLOB DIRECT /end PA_BLOB         /* addressing mode */
       /begin KP_BLOB 0x8038 WORD /end KP_BLOB    /* address, address length */
```

```
/end IF_DATA

/begin FUNCTION_LIST  CLDSTRT  FLLD                 /* reference to functins */
/end FUNCTION_LIST

 /end MEASUREMENT


 /begin MEASUREMENT           N
                             "current speed"
                             UWORD                 /* data type */
                             N_RULE                /* reference to conversion
                                                   method */
                             4                     /* resolution in bits */
                             0.006                 /* accuracy in '%' */
                             0.0                   /* lower limit */
                             5800.0                /* upper limit */
                             BIT_MASK  0xFFFF      /* bit mask */
/begin IF_DATA  DIM
       /begin DP_BLOB EXTERN /end DP_BLOB          /* memory type */
       /begin PA_BLOB DIRECT /end PA_BLOB          /* addressing mode */
       /begin KP_BLOB 0x8020 WORD /end KP_BLOB     /* address, address length */
/end IF_DATA
/begin FUNCTION_LIST         V_LIM  CLDSTRT  FLLD  /* reference to functions */
/end FUNCTION_LIST
 /end MEASUREMENT


/begin COMPU_METHOD          FACTOR01      /* name */
                             "factor 1"    /* long identifier */
                             RAT_FUNC      / *fractional rational function */
                             "%4.0"        /* format string */
                             ""            /* unit */
                             /* coefficients for polynome conversion */
                             COEFFS   0.0  1.0  0.0  0.0  1.0  0.0
 /end COMPU_METHOD

 /begin COMPU_METHOD  M_E                  /* name */
                             "amount"      /* long identifier */
                             TAB_INTP      /* conversion table with interpolation*/
                             "%4.0"        /* format string */
                             "mg/H"        /* unit */
                             COMPU_TAB_REF AMOUNT          /* reference to table */
 /end COMPU_METHOD

 /begin COMPU_METHOD          N_RUL                 /* name */
                             "speed"               /* long identifier */
                             RAT_FUNC              /* fractional rational function */
                             "%4.0"                /* format string */
                             "1/min"               /* unit */
                             /* coefficients for polynome conversion: "don't care" */
                             COEFFS   0.0 255.0  0.0  0.0  5800.0  0.0
 /end COMPU_METHOD


 /begin COMPU_METHOD          VOLTAGE               /* name */
                             "voltage"             /* long identifier */
```

```
                              RAT_FUNC              /* fractional rational function */
                              "%4.0"                /* format string */
                              "mV"                  /* unit */
                              /* coefficients for polynome conversion: "don't care" */
                              COEFFS   0.0 255.0  0.0  0.0  5000.0  0.0
  /end COMPU_METHOD

  /begin COMPU_TAB            AMOUNT                /* name */
                              "conversion table for AMOUNT"
                              TAB_INTP              /* table with interpolation */
                              4                     /* number of value pairs   */
                              0  0.0  100  10.0  156  30.0  255  43.0 /* value pairs*/
  /end COMPU_TAB

  /begin FUNCTION             V_LIM  "speed limitation"    /end FUNCTION
  /begin FUNCTION             CLDSTRT "cold start"         /end FUNCTION
  /begin FUNCTION             FLLD  "full load"            /end FUNCTION

  /* BOSCH record layout */
  /begin RECORD_LAYOUT DAMOS_FW             /* DAMOS constant */
                              FNC_VALUES    /* description of function value: */
                                 1          /* position in memory */
                                 UBYTE      /* data type of the constant */
                                 COLUMN_DIR  /* deposited in columns (don't care) */
                              DIRECT        /* direct addressing */
  /end RECORD_LAYOUT


/begin RECORD_LAYOUT   DAMOS_KF             /* DAMOS characteristic diagram*/
                              SRC_ADDR_X    /* description of the addresses of the X-
                                            input quantities */
                                  1         /* position in memory */
                                  UWORD     /* datatype */
                              NO_AXIS_PTS_X /* description of the number of X-axis
                                            points */
                                  2         /* position in memory */
                                  UBYTE     /* word length */
                              AXIS_PTS_X    /*description of the X-axis point values*/
                                  3         /* position in memory */
                                  UBYTE     /* data type of the axis point values */
                              INDEX_INCR    /* increasing index with increasing
                                            addresses */
                              DIRECT        /* direct addressing */
                              SRC_ADDR_Y    /* description of the addresses of the Y-
                                            input quantities */
                                  4         /* position in memory */
                                  UWORD     /* datatype */
                              NO_AXIS_PTS_Y /* description of the number of Y-axis
                                            points */
                                  5         /* position in memory */
                                  UBYTE     /* word length */
                              AXIS_PTS_Y    /*description of the Y-axis point values*/
                                  6         /* position in memory */
                                  UBYTE     /* data type of the axis point values */
                              INDEX_INCR    /* increasing index with increasing
                                            addresses */
                              DIRECT        /* direct addressing */
                              FNC_VALUES     /* description of the function values */
                                 7           /* position in memory */
                                 UBYTE       /* data type of the table values */
                                 COLUMN_DIR  /* deposited in columns */
                              DIRECT         /* direct addressing */
     /end RECORD_LAYOUT
     /* SIEMENS record layout */
     /begin RECORD_LAYOUT     SIEMENS_KF     /* SIEMENS characteristic map */
                              AXIS_PTS_X     /* description of the function values:
                                             axis points are described in an
                                             additional specification */
                                  1          /* position in memory */
                                  UWORD      /* data type of the table values */
                              COLUMN_DIR     /* deposited in columns */
                              DIRECT         /* direct addressing */
     /end RECORD_LAYOUT

     /begin RECORD_LAYOUT     SIEMENS_SST    /* SIEMENS axis points distribution */
                              AXIS_PTS_X     /* description of the axis point values*/
                                  1          /* position in memory */
                                  UWORD      /* data type of the axis point values */
                              INDEX_INCR     /* increasing index with increasing
                                             addresses */
                              DIRECT         /* direct addressing */
     /end RECORD_LAYOUT

     /begin MEASUREMENT       N1
                              "engine speed"
                              UWORD          /* data type */
                              R_SPEED_3      /* reference to conversion method */
                              2              /* resolution in bits */
                              2.5            /* accuracy in '%' */
                              120.0          /* lower limit */
                              8400.0         /* upper limit */
    BIT_MASK             0x0FFF              /* bit mask */
    BYTE_ORDER           MSB_FIRST
    /begin FUNCTION_LIST ID_ADJUSTM  FL_ADJUSTM      /* reference to functions */
    /end FUNCTION_LIST
     /end MEASUREMENT


    /begin COMPU_METHOD       R_SPEED_3              /* name */
                              "processing the speed"       /* long identifier */
                              RAT_FUNC               /* fractional rational function */
                              "%4.0"                 /* format string */
                              "kmh"                  /* unit */
                              /* coefficients for polynome conversion */
                              COEFFS   0.0  0.0  0.0  0.0  1.0  1.0
     /end COMPU_METHOD

     /begin FUNCTION          ID_ADJUSTM  "ID adjustment"        /end FUNCTION
     /begin FUNCTION          FL_ADJUSTM "FL adjustment"         /end FUNCTION
 /end MODULE
```

C IEEE-Floating-Point-Format

C.1 32-Bit Format

Table 16 IEEE-Floating-Point-Format (32-Bit)

Sign	Biases Exponent					Significant						
S	e7	e6	...	e1	e0	B1	b2	b3	...	b21	b22	b23

31 23 0

Representation of real numbers: $(-1)^s * 2^E * b_{0\Delta} b_1 b_2 b_3 \dots b_{23}$

s: 0 or 1
E: any integer between -126 and +127 (E = e - 127)
b_i: 0 or 1 (where b_0 = 1)

RealNumber = $(-1)^s * 2^{(-127)+\sum_{i=0}^{7}\left(e_i * 2^i\right)} * \sum_{i=0}^{23}\left(\frac{b_i}{2^i}\right)$ where b_0 = 1

C.2 64-Bit Format

Table 17 IEEE-Floating-Point-Format (64-Bit)

Sign	Biases Exponent					Significant						
S	e10	e9	...	e1	e0	b1	b2	b3	...	b50	b51	b52

63 52 0

Representation of real numbers: $(-1)^s * 2^E * b_{0\Delta} b_1 b_2 b_3 \dots b_{52}$

s: 0 or 1
E: any integer between -1022 and +1023 (E = e - 1023)
b_i: 0 or 1 (where b_0 = 1)

RealNumber = $(-1)^s * 2^{(-1023)+\sum_{i=0}^{10}\left(e_i * 2^i\right)} * \sum_{i=0}^{52}\left(\frac{b_i}{2^i}\right)$ where b_0 = 1

D USING REFERENCE CURVES AS NORMALIZATION AXES FOR MAPS

D.1 BACKGROUND

D.1.1 GENERAL

In certain calibration environments, the use of the same curves for normalization of two or more maps can save time during calibration and ROM storage while providing calibration control of resolution. A map specifies an axis as a CURVE_AXIS (See keywords AXIS_DESCR and CURVE_AXIS_REF) with a reference to the name of an existing curve defined elsewhere in the symbolic information. Typically, two such curves would be specified, one to normalize the map in the X direction, and one to normalize the map in the Y direction. Because of the interpolation within the curve and then in the map, the number of pairs in the curve does not have to match the number of points along the corresponding map axis.

D.1.2 OVERALL TECHNIQUE

Two curves are needed; one for the map X direction and one for the map Y direction. The X values for the curves must be ordered such that their values are always staying the same or increasing. The output of each curve are indices used to determine four cells within a map, whose values are interpolated to result in a single Z value. Since the output of a curve is a floating-point value, the value is truncated to determine one cell in the map, with the adjacent cell one larger. The interpolation is done twice in the X direction (once for each row) resulting in two Z values. These two Z values are then interpolated to result in a final Z value. The interpolation uses the identical-slope method, where for two existing known points, (X1,Y1) and (X2,Y2), the corresponding Y value for a given X value that lies on the line between the two points is given by: $Y = Y1 + (X-X1)((Y2-Y1)/(X2-X1))$

D.1.3 DETERMINING THE MAP INDICES

When the input value to a reference curve is less than or equal to the lowest X value, the result is set to the Y value corresponding to the lowest X value. When the input value to a curve is greater than or equal to the highest X value, the result is set to the Y value corresponding to the highest X value. When the input value matches an X value, the result is set to the corresponding Y value. Otherwise, the result is determined by interpolation between the two adjacent pairs whose X values form boundaries around the input value.

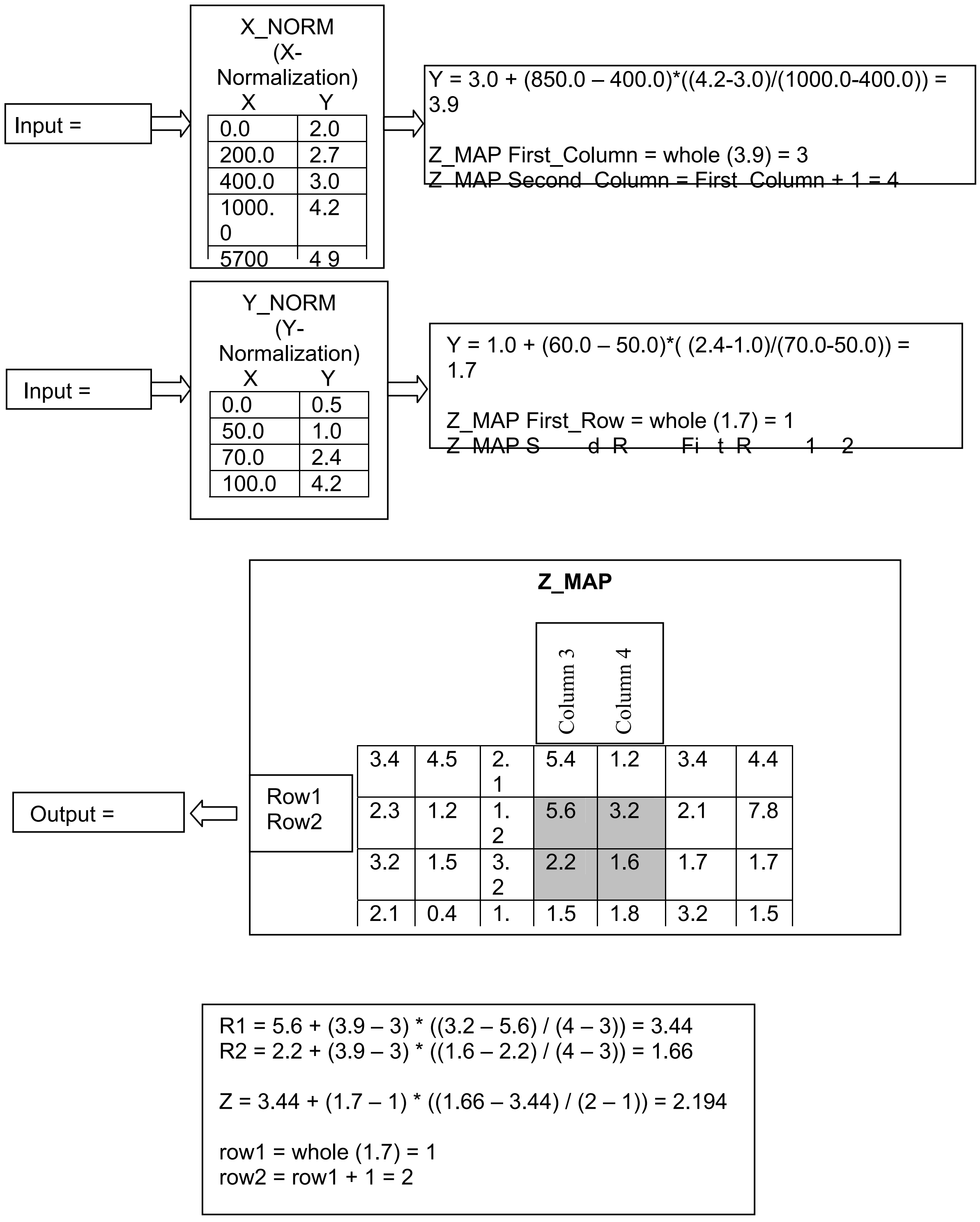
D.1.4 DETERMINING THE MAP NORMALIZED VALUE

The map cells are artificially numbered with integers beginning with 0 in both the X and Y direction. Two floating-point values are obtained from two curves, one for the X direction and one for the Y direction.
If the X direction floating-point value is less than 0, then the corresponding two column indices are set to 0. Otherwise the first column index is set to the whole number portion of the value and the second column index is set to one larger. Now, if the second column index is greater than or equal to the number of columns in the map, the first and second column indices are set to the number of columns minus 1.
If the Y direction floating-point value is less than 0, then the corresponding two row indices are set to 0. Otherwise the first row index is set to the whole number portion of the value and the second row index is set to one larger. Now, if the second row index is greater than

or equal to the number of rows in the map, the first and second row indices are set to the number of rows minus 1.
The intersection of the rows and columns determine four cells whose values are interpolated to result in a single Z value. The two values at the intersecting columns of one row are interpolated to obtain an intermediate result, and likewise for the other row. The two intermediate results are then interpolated in the Y direction to come up with the final Z result.

D.2 EXAMPLE

The following example shows a map (Z_MAP) with 7 columns and 6 rows. The normalization curve for the column is a referenced curve (X_NORM) that contains 5 pairs. The normalization curve for the row is a referenced curve (Y_NORM) that contains 4 pairs. The number of pairs for each normalizing curve does not match the number of row and columns in the map. An input value of 850.0 into X_NORM produces a Y output of 3.9, and an input value of 60.0 for Y_NORM produces a Y output of 1.7. The four points in the table to interpolate would be the intersection of columns 3 and 4, and rows 1 and 2. First the two points in row 1, columns 3 and 4 (5.6, 3.2) are interpolated to get a value of 3.44 . Then the two points in row 2, column 3 and 4 (2.2, 1.6) are interpolated to get a value of 1.66. These two resulting values are then interpolated to produce a final Z value of 2.194.



Input =

X_NORM (X-Normalization)

X	Y
0.0	2.0
200.0	2.7
400.0	3.0
1000.0	4.2
5700	4 9

Y = 3.0 + (850.0 – 400.0)*((4.2-3.0)/(1000.0-400.0)) = 3.9

Z_MAP First_Column = whole (3.9) = 3
Z_MAP Second_Column = First_Column + 1 = 4

Input =

Y_NORM (Y-Normalization)

X	Y
0.0	0.5
50.0	1.0
70.0	2.4
100.0	4.2

Y = 1.0 + (60.0 – 50.0)*((2.4-1.0)/(70.0-50.0)) = 1.7

Z_MAP First_Row = whole (1.7) = 1
Z MAP S d R Fi t R 1 2

Z_MAP

				Column 3	Column 4		
	3.4	4.5	2.1	5.4	1.2	3.4	4.4
Row1	2.3	1.2	1.2	▨ 5.6	▨ 3.2	2.1	7.8
Row2	3.2	1.5	3.2	▨ 2.2	▨ 1.6	1.7	1.7
	2.1	0.4	1.	1.5	1.8	3.2	1.5

Output =

R1 = 5.6 + (3.9 – 3) * ((3.2 – 5.6) / (4 – 3)) = 3.44
R2 = 2.2 + (3.9 – 3) * ((1.6 – 2.2) / (4 – 3)) = 1.66

Z = 3.44 + (1.7 – 1) * ((1.66 – 3.44) / (2 – 1)) = 2.194

row1 = whole (1.7) = 1
row2 = row1 + 1 = 2

Index of Keywords and Enum Values

Figuredirectory

Tabledirectory

Books

[ASAM AE CDF] ASAM: Calibration Data Format V2.0.0, 2006

[ASAM AE MDF] ASAM: Format Specification MDF Format V3.2, 2008

[ASAM MCD-2FIBEX] ASAM: FIBEX - Field Bus Exchange Format V3.0, 2008

[ASAM MCD-3] ASAM: Application Programming Interface Specification V2.2, 2008

[ASAP-3MC] ASAM: Application Systems Interface Specification ASAP3-MC, 1999

[ASAM MCD-2D ODX] ASAM: Diagnostic Data Model Specification V2.2, 2008

[ASAM AE COMMON SEED&KEY] ASAM: Seed&Key and Checksum Calculation API V1.0

[ISO 9141] ISO/DIS 9141
Road vehicles -- Diagnostic systems –
Part 1: Requirements for interchange of digital information, 1989
Part 2: CARB requirements for interchange of digital information, 1994
Part 3: Verification of the communication between vehicle and OBD II scan tool, 1998

[ISO 14229] ISO/DIS 14229
Road vehicles -- Unified diagnostic services (UDS) –
Part 1: Specification and requirements, 2006

[ISO 14230] ISO/DIS 14230
Road vehicles -- Diagnostic systems -- Keyword Protocol 2000 --
Part 1: Physical layer, 1999
Part 2: Data link layer, 1999
Part 3: Application layer, 1999
Part 4: Requirements for emission-related systems, 2000

[ISO 15765] ISO/DIS 15765
Road vehicles -- Diagnostics on Controller Area Networks (CAN) –
Part 1: General information, 2004
Part 2: Network layer services, 2004
Part 3: Implementation of unified diagnostic services (UDS on CAN), 2004
Part 4: Requirements for emissions-related systems, 2005

[ISO 22901-1] ISO: ISO/DIS 22901-1
Road vehicles — Open diagnostic data exchange — Part 1: Data model specification, 2008

[ISO 22900-2] ISO: ISO/DIS 22900-2
Road vehicles — Modular vehicle communication interface (MVCI) — Part 2: Diagnostic protocol data unit application programmer interface (D-PDU API), 2006

[ISO 22900-3] ISO: ISO/DIS 22900-3
Road vehicles - Modular vehicle communication interface (MVCI) - Part 3: Diagnostic server application programming interface (D-Server API), 2007

[What's New] ASAM: What's new in ASAM MCD-2MC V1.6

ASAM e.V.

Arnikastrasse 2

D-85635 Höhenkirchen

Germany

Tel.:	(+49) 08102 / 8953 17
Fax.:	(+49) 08102 / 8953 10
E-mail:	info@asam.net
Internet:	www.asam.net