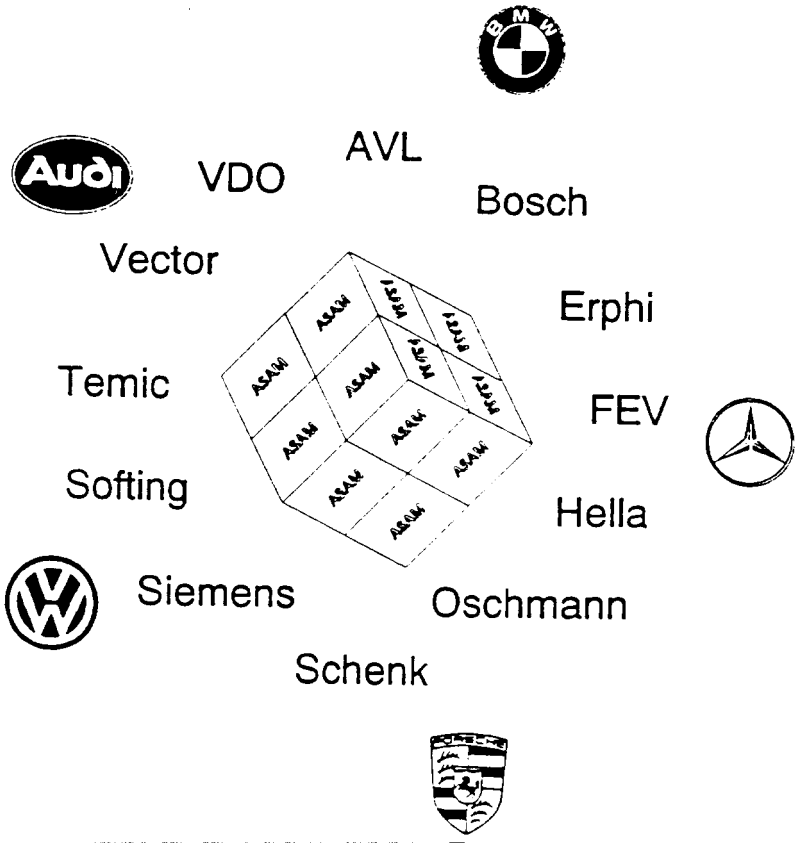


ASAP

Applications Systems Standardization Working Group

Interface Specifications

Interface 2

Version 1.0 of 31 March 1994

ASAP2: STANDARDIZED

DESCRIPTION DATA

Authors: Dr. H. Schelling — Fa. Vector Informatik
Dipl.-Ing. P.Lampert — Fa. Vector Informatik
Dr. C. Dallmayr — Fa. Softing
Dipl.-Inf. G.Luderböck — Fa. Softing

Version: 1.0

Date: 31 March 1994

Contents

1 ASAP: Goals, Method, Interfaces

The working group for the standardization of application systems (ASAP) was created in the autumn of 1991 at the initiative of the development boards of the German automobile manufacturers.The motivation is cost reduction based on co-operation in non-product relevant fields. The initiative is supported by the automobile manufacturers Audi, BMW, Mercedes-Benz, Porsche and VW together with the contractors active in this segment Bosch, Hella, Siemens, Temic, VDO and free suppliers and automation suppliers such as AVL, Erphi, FEV, Schenk, Softing and Vector.

The working group for the standardization of application systems (ASAP) aims at making the tools and methods generated during the development phase of vehicle electronics compatible with each other and hence interchangeable.
To this end the system parts required for the application as well as for verification and testing are adapted to the current, technical requirements in parallel with the design and development phase of the actual control units and thus brought to a high maturity level. In practice these efforts make it possible to incorporate individual components of the overall system into the process chain and to integrate them with other application environments.

To reach these goals, the ASAP group has agreed to subdivide the overall system into sub-components (Figure 1) using commonly defined interfaces that are compatible and interchangeable.

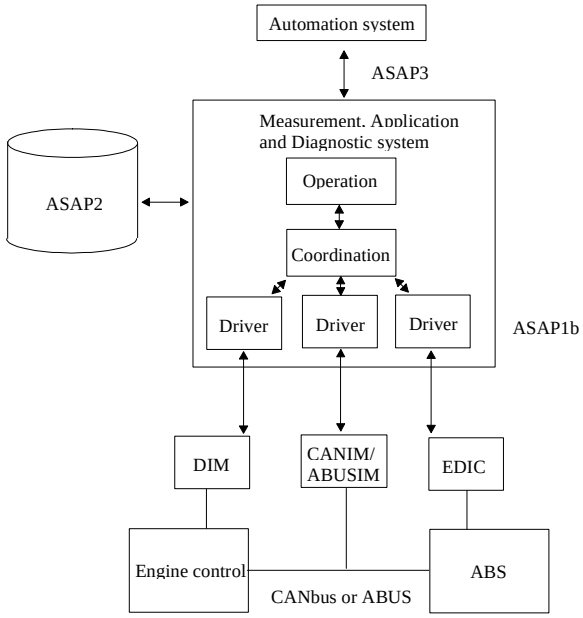


Figure 1 Overall system and interfaces

The individual application systems (AS) of the measurement, application and diagnostic system (MAD) are linked to the automation via interface ASAP3, they obtain information about the control unit's internal elements, its interfaces and communication methods from the ASAP2 description file, and are in turn linked to the control units (ECU) and the control unit dependent measurement technology (ADC) via the ASAP1 interface via ROM emulators, CAN, ABUS or the diagnostic bus (D bus). This structure allows current monolithic applications to be divided into compatible subsystems.

Via the ASAP1b interface the standard connection of the control units and of the control unit dependent measurement technology is realised independently of the chosen communication path or the relevant supplier of the control unit. To obtain this functionality, the control unit or the measurement system is linked to the measurement, application and diagnostic system via the transport path, the interface hardware and a driver in accordance with the ASAP specifications. This subsystem below interface 1b is identified by the ASAP device (Figure 2):

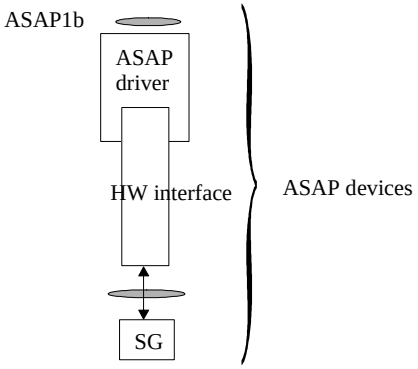


Figure 2 ASAP device (principle structure)

An ASAP device thus defined may be composed of different elements depending on the selected connection :

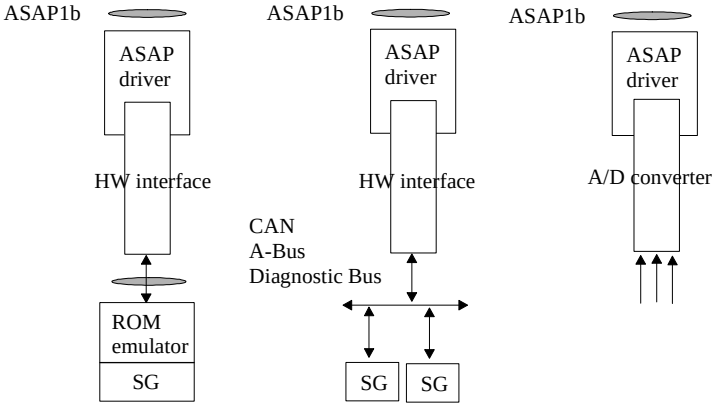


Figure 3 various versions of the ASAP device

The ASAP 1b interface is a functional interface which was initially defined independently of the MAD operating system. It offers a range of services controlling the exchange of parameters and data via the ASAP 1b. These ASAP services are:

Service	Name	Function description
1	INIT_READ	Initialisation of the measurement system
2	INIT_WRITE	Initialisation of the adjustment system
3	SYNC	Synchronisation of the individual subsystems
4	READ	Data transfer upstream of ASAP 1b
5	WRITE	Data transfer downstream of ASAP 1b
6	STOP	End of measurement data collection for intelligent module type 2..4
7	FREE_HANDLES	Enable references - reject subsystem measurement data
8	GIVE_STATUS	Explicit status interrogation in the event of an error

Table 1 ASAP Services Interface 1b

This allows similar subsystems such as ROM emulators of the firms AB and XY to be accessed in the same way. This 'similarity' is related to the use of the above-mentioned ASAP services which imply above all commonly agreed communication procedures. Special features and different communication methods within the ASAP device are embedded in this device, the setting of parameters occurs by allocating equally special binary objects and parameters from the ECU description file. Interpretation of the objects is only possible by the ASAP device. Furthermore, uniform communication with the control unit is possible independently of the selected connection. Different features of the various control unit connections can be balanced within the ASAP device, while the functional communication, e.g. the setting of parameters, is also standardised.

The ASAP description file (ASAP2) is used to describe the ECU internal data. This ASAP subsystem can be created from a number of different subelements:

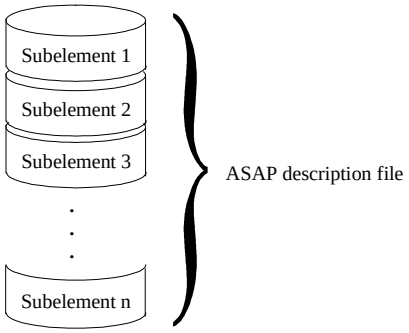


Figure 4 ASAP description file

The subelements of the ASAP description file are :

- project relevant information,
- data structure in the control unit,
- external interfaces,
- communication methods as ASAP device and
- the conversion procedures for representation in physical units.

An ASAP description file constitutes the reference for an individual control unit and its link to the ASAP interface 1b. It contains the following information :

Subelement	Contents	Examples
1	project-related data	Name of the relevant user of the subcontractor, data reference number
2	ECU-internal structures	Status and structure of the warm-up characteristic map, content of the user information field
3	Conversion rules	Scaling values for conversion from hexadecimal to physical for subelements 2, 4 and 7
4	Measurement channels	Addresses, resolution and update rates of the measurable RAM cells
5	Methods	Parameters of ASAP device for communication setup with the ECU, e.g. hexcode for emulator
6	HW layout	Segmentation of the related memory modules in the ECU e.g. data block size
7	SW interfaces	Description of the communication contents on the CANbus or ABUS, e.g. identifier and data content of the messages

Table 2 Subelements of ASAP interface 2

The individual subelements differ in terms of content, supplier and person functionally responsible as well as in terms of customer and his application system. However, the same information storage method is used throughout to allow for the global management of the individual components.

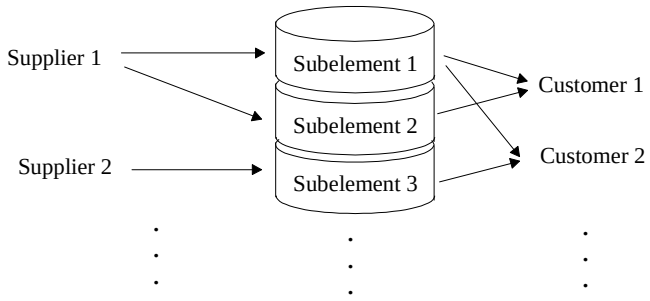


Figure 5 Customers and suppliers of the ASAP 2 subelements

This makes it possible to replace independent subelements e.g. in the case of a functional extension of the CAN interface: subelement 7, SW interfaces, section CAN, as well as subelement 1, project-related data, and thus to generate different description files corresponding to the current state of the control unit, which may then serve as input for other application systems.

Interface ASAP3 links up the measurement, application and diagnostic system (MAD) to an automation system (AuSy). From the standpoint of the AuSy it can therefore be considered as an intelligent device as e.g. an indexing device or a fuel scale. It incorporates both the various methods for access to control units as well as the individual structures in the control units and offers the following higher-level functions:

- starting an application on the MAD
- executing the diagnostic function
- executing the application function
- executing the measurement system function

These functions are offered by applications with an ASAP3 interface (e.g. give current engine speed, give content of warm-up characteristic map, give error memory) without the AuSy having to know the control unit specific details. The MAD offers its services to the AuSy at a quasi higher level. These services are based on the information on addresses, scalings, methods etc. in control units, interfaces and ASAP devices, which is obtained from the ASAP2 description file:

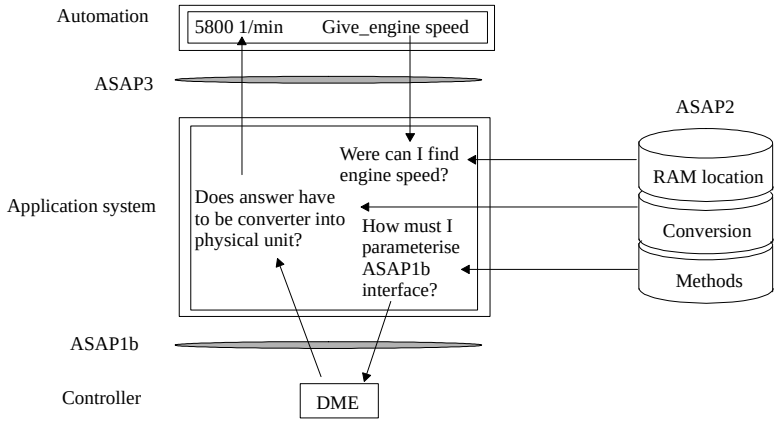


Figure 6 Interworking of Automation system, Application system, DME with ASAP interfaces

The AuSy does not require any special knowledge about the methodology and the parameter setting of the interfaces to the control units for its automation sequences. This service is provided by the MAD. The connection is established via the ASAP 3 interface.

PART A: DIVISION OF THE DESCRIPTION DATA

2 Division

The definition of the ASAP 2 interface and hence the specification of the ASAP2 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.

From the application point of view the database in accordance with the ASAP 2 interface contains the complete description of all control unit relevant data in a project. A project consists of project specific header data and one or more control unit specific descriptions. These control unit descriptions (= description of an ASAP device) 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 (for ASAP interface 1b). The measurement, application and diagnostic system need only evaluate the quantities (and their conversion etc.), but not the interface specific parameters. The latter are only passed on to the structures of the ASAP 1b driver. To make sure that these structures are correctly filled the MAD must know the parameter type. The type is communicated with the ASAP2 metalanguage (see part C).

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 ASAP2 database thus consists of a number of different subcomponents structured in accordance with the following diagram. The MODULE keyword denotes an independent ASAP device

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

        MODULE ASAP-DEVICE1
        {
        MOD_PAR{...}                            /*Control unit management data*/
        MOD_COMMON {...}                        /*Module-wide (ECU specific) definitions*/

        CHARACTERISTIC{...}                     /*Adjustable objects*/
        CHARACTERISTIC{...}
        ...
        MEASUREMENT{...}                        /*Measurement objects*/
        MEASUREMENT{...}
        ...
        COMPU_METHOD{...}                       /*Conversion method*/
        COMPU_METHOD{...}
        ...
        COMPU_TAB{...}                  /*Conversion tables*/
        COMPU_TAB{...}

        FUNCTION{...}                           /*Function allocations*/
        FUNCTION{...}
        ...
        RECORD_LAYOUT{...}                      /*Record layouts of adjustable objects*/
        RECORD_LAYOUT{...}
        }
        MODULE ASAP-DEVICE2
        {
        MOD_PAR{...}                            /*Control unit management data*/
        MOD_COMMON {...}                        /*Module-wide (ECU specific) definitions*/

        CHARACTERISTIC{...}                     /*Adjustable objects*/
        CHARACTERISTIC{...}
        ...
        MEASUREMENT{...}                        /*Measurement objects*/
        MEASUREMENT{...}
        ...
        COMPU_METHOD{...}                       /*Conversion method*/
        COMPU_METHOD{...}
        ...
        COMPU_TAB{...}                  /*Conversion tables*/
        COMPU_TAB{...}
        ...
        FUNCTION{...}                           /*Function allocations*/
```

```
FUNCTION{...}
...
RECORD_LAYOUT{...}                          /*Record layouts of adjustable objects*/
RECORD_LAYOUT{...}
}
MODULE ASAP DEVICE 3
{
...
}
}                                           /* END OF PROJECT*/
```

The keywords defined in the ASAP2 database are described in Part B.

3 System Description (SG Verbund)

Within the scope of the standardization of application systems, it is intended to simultaneously apply control units of various manufacturers (e.g. engine control/ transmission control/anti-slip control). The ASAP2 database supports this: under one project header, which describes the overall system, the control unit descriptions (ASAP devices) of a number of manufacturers can be included, e.g. by 'INCLUDE'. Thus, an efficient overall standardization can be implemented with a common application system.

Figure 7 gives a schematic representation of the ASAP devices to be commonly applied, summarized under one project

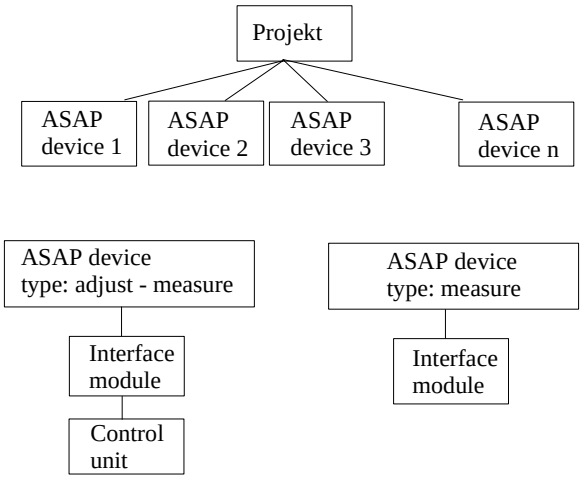


Figure 7 Model application system

An ASAP device can thus be a control unit with related interface and ASAP 1b driver or only an interface with driver. The second case applies if e.g. quantities, exchanged via the bus as message, are monitored via ABUS CAN (bus monitor).

The project header (keyword HEADER, page 56) is typically created by the project manager and includes the following entries:

- version of the parameter file format [keyword VERSION, page 90]
- project
- project identifier
- project number [keyword PROJECT_NO, page 76]
- remarks, comments

4 Description Application Devices

For each ASAP devicea complete description (keyword MODULE, page 64) is created. The description is supplied by the relevant supplier of the control unit. The components of this description are:

- control unit management data (e.g. users responsible,...)[MOD_PAR, page 67]
- control unit internal structures (e.g. standard record layout) [MOD_COMMON, page 66]
- communication interfaces [IF_DATA, see Part C]
- adjustable and measurement objects [CHARACTERISTIC, page 32 and MEASUREMENT, page 60]
- conversion rules [COMPU_METHOD, page 35]

4.1 Control unit management data

The control unit management data are created with the MOD_PAR (page 67) keyword.
Using optional parameters the following can be specified:

- data status [VERSION, page 90]
- comments, remarks

- EPROM identifier address [ADDR_EPK, page 23]
- EPROM identifier [EPK, page 46]

- Manufacturer or supplier [SUPPLIER, page 86]
- Firm or customer [CUSTOMER, page 41]
- Customer number [CUSTOMER_NO, page 42]

- User [USER, page 89]
- Telephone numbers (applications engineer responsible) [PHONE_NO, page 74]
- Control unit [ECU, page 45]
- Processor [CPU_TYPE, page 40]
- Number of interfaces [NO_OF_INTERFACES, page 71]
- Memory layout [MEMORY_LAYOUT, page 62]
- System constants [SYSTEM_CONSTANT, page 87]

- Project-basis-address (see MEMORY_LAYOUT: memory layout)

4.2 General description data (control unit internal structures)

These description data (keyword MOD_COMMON, page 66) make it possible to specify a number of parameters for the module as a whole (i.e. for this ASAP device). The following optional parameters can be specified:

- Standard record layout (is there one standard layout?) [S_REC_LAYOUT, page 88]
- Standard deposit mode of the axis points (difference, absolute values) [DEPOSIT, page 44]
- Byte order [BYTE_ORDER, page 31]
- Data size in bits [DATA_SIZE, page 43]

4.3 Interface Parameters (general parameters)

A format description of the interface specific parameters must be made available by the supplier in ASAP2 metalanguage (A2ML: see Part C). The measurement, application, diagnostic system need not know the driver parameter settings: to read the interface specific parameters only a description of the data types is required.

4.3.1 Interface module (memory emulator)

These parameters describe the access methods to the measurement data collection:

- Display table type (code patch in the ECU for measurement data output), address display table(s), maximum length of the display table(s)
- Output: which memory address
- Triggering: trigger segment address

4.3.2 CAN bus

If in a specific project a number of application devices have access to the CAN bus, a consistency check must be carried out for these paramteres in the conversion program (physically only one CAN bus, all corresponding data records of the 'interface parameter' type must match).

- Bus timing
- Configuration
- Bus parameters
- Access methods fo measurement data collection:
    * Collection through passive hearing or remote frames
    * Note: Here there is no description of the CAN identifier
- possible description of a protocol for data adjustment

4.3.3 ABUS

These parameters describe the communication via the ABUS (VW):

- Bus operating mode (reference, difference)
- Bus parameters (number of scannings, error figures etc.)
- ADEX functions realised in the drivers

4.3.4 Bus parameters for serial protocols (ISO)

Here parameters describing serial communication are stored:

- Protocol identification (which serial protocol)
- Protocol parameters (timing), protocol specific
    - byte interval of tester
    - byte time-out of control unit
    - lock sequence time of tester

- block time-out of control unit
- timeout during communication setup
- entry time during free-running

- ECU address
- Access methods
  - telegram answer
  - telegram-answer-answer ... (free-running)
  - display table (free-running)
    - length and structure of the display table
- Communication setup process (if necessary according to the protocol)
  - type of trigger
    - none
    - 5 Baud
    - low level
    - fast trigger (10.4 kBaud)
  - send telegram for initialisation (dummy)
  - source type
    - time synchronous
    - crankshaft synchronous
    - event synchronous
    - asynchronous

4.3.5 Analog interface

Does not require any interface parameters!

4.4 Adjustment objects (one description per adjustment object)

Each adjustment object must have its own description to be created with the CHARACTERISTIC keyword (see page 32). Between the /begin CHARACTERISTIC and /end CHARACTERSITIC brackets further keywords can be nested (as may also be the case for other keywords).

Example:

```
/begin CHARACETRISTIC PUMPKF            /*Description of a characteristic map*/
    ...
    ...
    /begin AXIS_DESCR                       /*axis descriptionfor x axis*/
    ...                                     /*...nesting depth 2*/
    ...
    MAX_GRAD      7.0                       /*gradient limited, nesting depth 3*/
    /end AXIS_DESCR
    /begin AXIS_DESCR                       /*axis description for Y axis*/
    ...
    ...
    /end AXIS_DESCR

    /end CHARACTERISTIC
```

The description of the adjusment objects contains references to (possibly) common conversion methods for a number of adjustment objects, record layouts or functions. The referenced objects are described only once under their own keyword.

4.4.1 Deposit structure [CHARACTERISTIC, page 32]

Mandatory parameters
- Name of the adjustable object
- Comment, function description
- Type of adjustable object
  * value, value block, curve, map, ASCII, pointer
  * fixed curve, fixed map
  * group curve, group map, axis point distribution,
  * vector curve, vector map
- address, address location, addressing type
- record layout (enumeration or reference)
- maximum increment for incrementing or decrementing a function value

Optional parameters
- Axis description [AXIS_DESCR, page 24]
  * reference to input quantities
  * fixed curve or field: parameters for calculation of the axis point values
- Maximum gradient of the adjustable object between two neighbouring axis points (Delta_W/Delta_St)
- Monotony (with reference to an axis)

Remark:
The orientation (deposit: in rows or columns) as well as the word length of the axis points and function values (8 bit or 16 bit) is given in the layout.

4.4.2 Bit pattern conversion (axis points and function values)

- Reference to the list of conversion methods
- Byte order [BYTE_ORDER, page 31], signed-unsigned information
- Plausibilities (limit values)
- Physical representation: see Conversion method

4.4.3 Function orientation (Reference)[FUNCTION_LIST, page 55]

- List of those 'functions' that are allocated to this object. The referencing occurs by names.

Note:
This solution does not correspond with the current method realised in DAMOS (reference occurs inversely with an allocation list in the 'Function Record'.

4.5 Measurement channel (one description per measurement; e.g. AD value, CAN signal, source data, RAM cell)

For each measurement a description is created with the keyword MEASUREMENT (page 60). The description of the interface-specific data must be supplied by the supplier in A2ML.
As for the adjustable objects the description of the measurement object contains a reference to common conversion methods and function descriptions that can be referenced from various measurement objects.

Mandatory parameters
- name of the measurement channel
- comments, function description
- word length, bit mask for individual bit sizes

4.5.1 Deposit structure (measurement channel)

4.5.1.1 Description of the interface module deposit structure

Here the following can be described (see Part C):
- addresses, address location, address length (e.g. for entry in the display table)

4.5.1.2 Description of the CAN deposit structure

Here the following can be described (see Part C):
- Name of the CAN message
- CAN identifier
- Message length
- Sender
- Signal type (mode signal, mode dependent signal, standard signal)
- Reference to a mode signal (optional: if signal type = = mode dependent signal)
- start bit, signal length

4.5.1.3 Description of the ABUS deposit structure

Here the following can be described (see Part C):
- Name of the ABUS telegram
- ABUS identifier
- Repetition rate of the sender
- Faulty reactions and receiver timeouts

4.5.1.4 Description of the series protocol deposit structure (ISO)

Here the following can be described (see Part C):
- Send telegram (containing full telegram with dummy for address/length)
    - Command byte
    - Command data (detailing the components of 2.3.4)
        - -address
        - -number of bytes
    - Result data
        - -position in answer telegram
        - -deposit type (byte order, signed-unsigned information)
    - Display tables
        - -address, address location, address length
        - -deposit type (byte order, signed-unsigned information)

4.5.1.5 Analog interface deposit structure

Here the following an be described (see Part C):
- Analog input (No. 1-8)
- Resolution in bits
- Gain, prescaler

4.5.2 Bit pattern conversion

- Reference to conversion method list
- byte order [BYTE_ORDER, page 31], signed-unsigned information

Physical representation: see conversion methods

4.5.3 Function orientation (reference)

Under the keyword FUNCTION_LIST (page 55) a list of the functions allocated to this object is given. Referencing occurs via names (see comment in chapter 5.3.29 page 55).

4.6 Conversion method

The keyword COMPU_METHOD (repeatedly possible, see page 35) creates a list of the conversion methods used during the data adjustment and the collection of measurement data (conversion from the internal format in the emulation memory to the 'physical' representation of a quantity).

Parameters for the conversion method:
- Name of the conversion method
- Comment, function description
- Conversion method type (table with/without interpolation, polynome, verbal conversion table)
- Reference to conversion table [COMPU_TABLE_REF, page 38]
- Physical representation (significant positions, decimal places, physical unit)
- Coefficients for fractional rational function [COEFFS, page 34]

Note:
The 'physical representation' and 'plausibilities' parameters are allocated to the conversion method, as this significantly benefits the size of the description file (empirically there are fewer conversion methods than adjustable objects or measurement objects).

4.7 Conversion tables

Under the keyword COMPU_TAB (page 37) a list is given of the conversion tables used during the data adjustment and the measurement data collection (physical conversion and verbal conversion). It contains the mandatory parameters :

- Name of the conversion table
- Conversion table type (table without/with linear interpolation)
- number of value pairs
- value pairs

For the visualisation of the bit patterns verbal conversion tables (COMPU_VTAB, page 39) can be used (allocation table: bit pattern <--> string).

4.8 Function description [FUNCTION, page 54]

All functions referenced under CHARACTERISTIC and MEASUREMENT can be described as follows:
- Name of the function
- Comment, description of the function

4.9 Record layout [RECORD_LAYOUT, page 78]

Description of the various record layouts of the adjustable objects (see also Appendix A).

PART B: FORMAT OF THE DESCRIPTION FILE

5 FORMAT OF THE DESCRIPTION FILE

5.1 Hierarchic division of the keywords

Keyword	(*) = multiple	Meaning
PROJECT		
HEADER		Project description
VERSION		Version number
PROJECT_NO		Project number
MODULE	(*)	Description of the ASAP devices
MOD_PAR...		Control unit management data
MOD_COMMON		Module-wide (ECU specific) valid definitions
S_REC_LAYOUT		Reference to the standard layout
DEPOSIT		Standard d

5.2 Predefined data types

ident
typedef char [MAX_IDENT + 1] ident;
String with MAX_IDENT (at present = 32) characters. The character chain must correspond with the identifier laws defined in programming language C.
Identifiers are random names of maximum 32 characters long which may contain characters A through Z, a through z, underscore (_) and numerals 0 through 9. However, the following limitation applies: the first character must be a letter or an underscore.
Important:
In the available application systems only 10 characters can be displayed, i.e. longer identifiers are represented in abbreviated form. It is therefore recommended to choose the identifiers in such a way that the individual identifiers still differ when reduced to 10 characters and hence do not generate any ambiguities.

string
typedef char [MAX_STRING + 1] string:
String with maximum MAX_STRING (at present = 100 characters). Begin and end of the string are indicated by a double inverted comma. If the string contains one or more double inverted commas, two consecutive double inverted commas must be inserted.

float
8-byte floating point number (IEEE format)

int
2-byte signed integer

long
4-byte signed long

datatyp
typedef enum datatyp {UBYTE, SBYTE, UWORD, SWORD, ULONG, SLONG}
Enumeration for description of the basic data types in the ECU program

datasize typedef enum datasize {BYTE, WORD, LONG}
Enumeration for description of the word lengths in the ECU program

addrtyp
Enumeration for description of the addressing of table values or axis point values:

PBYTE :	The relevant memory location has a 1 byte pointer to this table value or axis point value.
PWORD:	The relevant memory location has a 2 byte pointer to this table value or axis point value.
PLONG :	The relevant memory location has a 4 byte pointer to this table value or axis point value.
DIRECT :	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}
Enumeration for description of the byte order in the control unit program.

indexorder
Enumeration for description of the axis point sequence in the memory.
INDEX_INCR: Increasing index with increasing address
INDEX_DECR: decreasing index with increasing address

5.3 Alphabetical list of keywords

The individual elements of the database are delimited by /begin and /end keywords, similarly to the opening '{' and closing '}' brackets in 'C'. Keywords whose validity range only extends over one line have no begin and end identifier.

To allow for a flexible description of the sub-components, optional keywords may be added to elements inserted within the begin and end keywords; i.e. nested keywords can be inserted.

5.3.1 A2ML

Prototype:

/start A2ML Format specification
/end A2ML

Parameters:

Format specification A2ML code for description of interface specific description data

Description:

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

Example:

See page 101: file ZULF1_IF.AML and file ZULF2_IF.AML

5.3.2 ADDR_EPK

Prototype:

ADDR_EPK Address

Parameters:

long address: Address of the EPROM identifier

Description:

Address of the EPROM identifier

Example:

ADDR_EPK Ox145678

5.3.3 AXIS_DESCR

Prototyp:

```
/begin AXIS_DESCR        Attribute InputQuantity Conversion MaxAxisPoints
                         LowerLimit UpperLimit
                         [-> AXIS_PTS_REF]
                         [-> MAX_GRAD]
                         [-> MONOTONY]
                         [-> BYTE_ORDER]
                         [-> FIX_AXIS_PAR]
                         [-> DEPOSIT]

/end AXIS_DESCR
```

Parameters:

enum attribute: Description of the axis points:
- STD_AXIS: Standard axis
- FIX_AXIS: This is a curve or a map with equidistant axis points
- COM_AXIS: Group axis points or description of the axis points for SIEMENS 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'.

ident input quantity: Reference to the data record for description of the input quantity (see MEASUREMENT).

ident conversion: Reference to the relevant record of the description of the conversion method (see COMPU_METHOD).

int Max axis points : Maximum number of axis points
Note: The application 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 application system cannot be changed.

float bottom limit: Plausible range of axis point values, lower limit

float upper limit: Plausible range of axis point values, upper limit

Optional parameters:

->AXIS_PTS_REF: Reference to the AXIS_PTS record for description of the axis points distribution.

->MAX_GRAD: This keyword can be used to specify a maximum permissible gradient for the adjustable object with respect to this axis (MaxGrad= max $((W_{i,k}-W_{i-1,k})/(X_i-X_{i-1}))$).

-> MONOTONY: This keyword can be used to specify a monotonous behaviour for the adjustable object with respect to this axis.

-> 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.

-> 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'.

-> 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 pointare stored as differences. Each axis point value is determined from the adjacent axis point (predecessor).
Where the standard value does not appply this parameter can be used to specify the axis point deposit.

Description:

Axis description within an adjustable object

Notes:
For the BOSCH group characteristics and for the SIEMENS standard curves or maps, the mandatory parameters 'input quantity', 'conversion','max axis points', 'lower limit', and 'upper limit' can be specified with the keyword AXIS_DESCR as well as with the keyword AXIS_PTS. This redundancy has been introduced to simplify the processing program. For these redundant parameters the processing programs should include a consistency check.
With the 'input quantity' parameter reference is made to a measurement object (MEASUREMENT, Page 60). The MEASUREMENT keyword also specifies the 'conversion', 'lower limit' and 'upper limit' parameters. These parameters arenot always identical to the parameters specified under AXIS_DESCR, as for the relevant measurement object either another conversion method (e.g. other solution) or other plausibility limits can apply than for the axis points of the relevant characteristic.

Example:

```
/begin AXIS_DESCR     ST_AXIS          /*Standard axis points */
                      N                /*Reference to input quantity */
                      CONV_N14         /*Conversion, max.number of axis points*/
                      0.0 5800.0       /*Upper limit, lower limit*/
        MAX_GRAD      20.0             /*Axis: maximum gradient*/
/end AXIS_DESCR
```

5.3.4 AXIS_PTS

Prototype:

```
/begin AXIS_PTS                 Name Long-Identifier Address Input quantity Deposit MaxDiff
                                Conversion  MaxAxisPoints  LowerLimit  UpperLimit
                                [-> DEPOSIT]
                                [-> BYTE_ORDER]
                                [-> FUNCTION_LIST]
                                [-> IF_DATA]
/end AXIS_TS
```

Parameters:

ident name:	identifier in the program
string long identifier:	comment, description
long address:	address of the adjustable object in the emulation memory
ident input quantity:	reference to the data record for description of the input quantity (see MEASUREMENT)
ident deposit:	reference to the relevant data record for description of the record layout (see RECORD_LAYOUT)
floax maxdiff:	maximum floatwith respect to the adjustment of a table value
ident conversion:	reference to the relevant data record for description of the conversion method (see COMPU_METHOD)
int maxaxis points:	maximum number of axis points
float lower limit:	plausible range of axis point values, lower limit
float upper limit:	plausible range of axis point values, upper limit

Optional parameters:

-> 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.
-> 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.
-> FUNCTION_LIST:	This keyword can be used to specify a list of 'functions' to which the axis points distribution is allocated (function orientation).
-> IF_DATA:	Data record for description of the interface specific description data (BLOB: binary large object). The parameters associated with this keyword are described in the ASAP2 metalanguage (in short A2ML) by the control unit supplier or interface module supplier. The structure of this data record corresponds with the structure of the interface-specific description data of the CHARACTERISTIC data record.

Description:

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

Example:

```
/begin AXIS_PTS                 STV_N                           /* name */
                                "axis points disrtribution 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 */
        FUNCTION_LIST       ID_ADJUSTM  FL_ADJUSTM SPEED_LIM
         IF_DATA                DIM EXTERNAL DIRECT
/end AXIS_PTS
```

5.3.5 AXIS_PTS_REF

Prototype:

AXIS_PTS_REF AxisPoints

Parameters:

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

Description:

In the BOSCH adjustable types 'group characteristic curve' and 'group characteristic map' and in the SIEMENS record layout, 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. This 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 0x1432            /* type, address */
                                   DAMOS_GKL 0.2 /* deposit, maxdiff */
                                   R_TORQUE                /* conversion */
                                   0.0  43.0               /* lower limit, upper limit */
    IF_DATA                        DIM EXTERNAL INDIRECT
    /begin AXIS_DESCR                                      /* description of X-axis points */
                                   COM_AXIS                /* standard axis points */
                                   N                       /* reference to input quantity */
                                   CONV_N 14               /* conversion, max. no. of axis p.*/
                                   0.0  5800.0             /* lower limit, upper limit */
      MAX_GRAD                     20.0                    /* X axis: maximum gradient */
       AXIS_PTS_REF          GRP_N
/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 */
    IF_DATA                        DIM EXTERNAL INDIRECT
/end AXIS_PTS
```

5.3.6 AXIS_PTS_X, AXIS_PTS_Y

Prototype:

AXIS_PTS_X	Position Datatype IndexIncr Addressing
or	
AXIS_PTS_Y	Position Datatype IndexIncr Addressing

Parameters:

int position:	Position of the axis point values in the deposit structure (description of sequence of elements in the data record).
datatype data type:	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 axis points or Y axis points in the memory (see keyword RECORD_LAYOUT)

Example:

AXIS_PTS_X	3 ULONG INDEX_INCR DIRECT

5.3.7 BIT_MASK

Prototype:

BIT_MASK Mask

Parameters:

long 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.

Example:

BIT_MASK 0x00000FFF

5.3.8 BYTE_ORDER

Prototype:

BYTE_ORDER Byte order

Parameters:

byteorder byte order: byte order of the relevant quantity in the ECU program (BIG_ENIAN corresponds to the Intel format; LITTLE_ENDIAN corresponds to the Motorola format)

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 BIG_ENDIAN

5.3.9 CHARACTERISTIC

Prototype:

```
/begin CHARACTERISTIC      Name Long-Identifier Type Address Deposit MaxDiff
                                Conversion Lower-Limit Upper-Limit
                                [-> BYTE_ORDER]
                                [-> BIT_MASK]
                                [-> FUNCTION_LIST]
                                [-> NUMBER]
                                [-> EXTENDED_LIMITS]
                                [-> READ_ONLY]
                                {-> IF_DATA } *
                                {-> AXIS_DESCR } *
/end CHARACTERISTIC
```

Parameters:

ident name:	identifier in the program
string long identifier:	comment, description
enum type:	possible types: VALUE CURVE MAP CUBOID VAL_BLK (array of values) ASCII (string)
long address:	address of the adjustable object in the emulation memory
ident deposit:	reference to the corresponding data record for description of therecord laout (see RECORD_LAYOUT)
float maxdiff:	maximum float with respect to an adjustment of a table value
ident conversion:	reference to the relevant data record for description of the conversion method (see COMPU_METHOD).
float lower limit:	plausible range of table values, lower limit
float upper limit:	plausible range of table values, upper limit

Optional parameters

-> 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.
-> BIT_MASK:	This parameter can be used to specify a bit mask for the handling of single bits.
-> FUNCTION_LIST:	This keyword can be used to specify a list of 'functions' to which the relevant adjustable object is allocated (function orientation).
-> 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.
-> EXTENDED_LIMITS:	This keyword can be used to specify an extended range of values. the application 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").

-> READ_ONLY:	This keyword can be used to indicate that the adjustable object cannot be changed (but can be read only).
-> IF_DATA:	Date record to describe the interface specific description data (BLOB: binary large object). The parameters associated with this keyword are described in the ASAP2 metalanguage (in short A2ML) by the control unit supplier or the interface module supplier.
-> 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.

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 */
    FUNCTION_LIST               NL_ADJUSTMENTFL_ADJUSTMENT SPEED_LIM
    IF_DATA                     DIM EXTERNAL INDIRECT
    /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
```

5.3.10 COEFFS

Prototype:

COEFFS a b c d e f

Parameters:

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

Description:

Specification of coefficients for the formula (axx + bx + c) / (dxx + ex + f).

Example:

COEFFS 2 17 3 0 0 4

5.3.11 COMPU_METHOD

Prototype:

```
/begin COMPU_METHOD     Name Long-Identifier Convers_type Format Unit
                            [-> FORMULA]
                            [-> COEFFS]
                            [-> COMPU_TAB_REF]
/end COMPU_METHOD
```

Parameters:

ident name:	identifier in the program for the conversion method	
string long-identifier:	comment, description	
enum convers_type:	possible types:	
	TAB_INTP:	table with interpolation
	TAB_NOINTP:	table without interpolation
	TAB_VERB:	verbal conversion table
	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 are specified by the optional COEFFS keyword. Important: For these coefficients restrictions have to be defined because this general equation cannot always be inverted.
	FORM:	conversion based on the formula specified by the optional FORMULA keyword.
string format:	display format in %[length] [layout]; length indicates the overall length; layout indicates the decimal places	
string unit:	physical unit	

Optional parameters:

-> FORMULA:	Formula to be used for the conversion
-> COEFFS:	This keyword is used to specify coefficients a, b, c, d, e, f for the fractional rational function of the following type (axx + bx + c) / (dxx + ex + f)
-> COMPU_TAB_REF:	This keyword is used to specify a conversion table (reference to COMPU_TAB data record).

Description:

Specification of a conversion method

Example:

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

5.3.12 COMPU_TAB

Prototype:

```
/begin COMPU_TAB                Name Long-Identifier Convers_type Number_Value_Pairs
                                { In_Val Out_Val }*
/end COMPU_TAB
```

Parameters:

ident name:	identifier in the program for the conversion table
string long-identifier:	comment, description
enum convers_type:	following types are possible:
	TAB_INTP: table with interpolation
	TAB_NOINTP: table without interpolation
int number_value_pairs:	number of successive value pairs for this conversion table
long in_val:	axis point
float out_val:	axis 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 */
/end COMPU_TAB
```

5.3.13 COMPU_TAB_REF

Prototype:

COMPU_TAB_REF	Conversion table

Parameters:

ident conversion table:	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).

Example:

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

5.3.14 COMPU_VTAB

Prototype:

```
/begin COMPU_VTAB          Name Long-Identifier Convers_type Number_Value_Pairs
                           { In_Val Out_Val }*
/end COMPU_VTAB
```

Parameters:

ident name:	identifier in the program for the verbal conversion table
string long-identifier:	comment, description
enum convers_type:	at present only the following types are possible: TAB_VERB: verbal conversion table
int number_value_pairs:	number of successive value pairs for this conversion table
long In_Val:	byte value
string Out_Val:	description (meaning) of the corresponding byte value

Description:

Conversion table for the visualisation 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
```

5.3.15 CPU_TYPE

Prototype:

CPU_TYPE	CPU

Parameters:

string CPU:	CPU identifier

Description:

CPU identification

Example:

CPU_TYPE	"INTEL 4711"

5.3.16 CUSTOMER

Prototype:

CUSTOMER	Customer

Parameters:

string customer:	customer name

Description:

This keyword allows a customer name to be specified.

Example:

CUSTOMER	"LANZ - Landmaschinen"

5.3.17 CUSTOMER_NO

Prototype:

CUSTOMER_NO	Number

Parameters:

string number:	customer number

Description:

Customer number as string

Example:

CUSTOMER_NO	"191188"

5.3.18 DATA_SIZE

Prototype:

DATA_SIZE	Size

Parameters:

int size:	data size in bits

Description:

Data size in bits

Example:

DATA_SIZE	16

5.3.19 DEPOSIT

Prototype:

DEPOSIT 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)
3-byte-size: <maxvalue> = 2^{32}

Example:

DEPOSIT DIFFERENCE

5.3.20 ECU

Prototype:

ECU Control unit

Parameters:

string control_unit: control unit identifier

Description:

String for identification of the control unit.

Example:

```
ECU  "Steering control"
```

5.3.21 EPK

Prototype:

EPK Identifier

Parameters:

string identifier: EPROM identifier

Description:

EPROM identifier string

Example:

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

5.3.22 EXTENDED LIMITS

Prototype:

EXTENDED_LIMITS	Lower limit upper limit

Parameters:

float lower limit:	extended range of table values, lower limit
float upper limit:	extended range of table values, upper limit

Description:

This keyword can be used to specify an extended range of values. In the application 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

5.3.23 FIX_AXIS_PAR

Prototype:

FIX_AXIS_PAR Offset Shift 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).
int 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

$Y_k = offset + (k - 1)*2^{shift}$ $k = \{ 1...numberapo \}$

Example:

FIX_AXIS_PAR 0 32 8

5.3.24 FIX_NO_AXIS_PTS_X, FIX_NO_AXIS_PTS_Y

Prototype:

FIX_NO_AXIS_PTS_X	Number of axis points
or	
FIX_NO_AXIS_PTS_Y	Number of axis points

Parameters:

int number_of_axis_points:	Dimensioning of characteristic curves or characteristic maps with a fixed number of axis points	

Description:

This keyword indicates that all characteristic curves or characteristic maps are allocated a fixed number of X-axis and Y-axis points. In a RECORD_LAYOUT data record, this keyword cannot be used simultaneously with the keyword NO_AXIS_PTS_X (for FIX_NO_AXIS_PTS_X) or NO_AXIS_PTS_Y (for FIX_NO_AXIS_PTS_Y)

Example:

FIX_NO_AXIS_PTS_X	17

5.3.25 FNC_VALUES

Prototype:

FNC_VALUES Position Datatype IndexMode Addresstype

Parameters:

int position: position of table values (function values) in the deposit structure (description of sequence of elements in the data record).

datatype data type: 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:

COLUMN_DIR:	deposited in columns
ROW_DIR:	deposited in rows

Both concepts 'columns'and 'rows' relate to the XY coordinate system (see also Appendix B: Record layouts).

addrtype address type: addressing of the table values (see enum addrtype).

Description:

Description of the table values (function values) of an adjustable object

Example:

FNC_VALUES 7 SWORD COLUMN_DIR DIRECT

5.3.26 FORMULA

Prototype:

FORMULA f(x)
[-> FORMULA_INV]

Parameters:

string f(x): function to calculate the physical value from the hexadecimal, 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. The following operation symbols can be used:

Basic operations:
+ for sums
- for differences
* for products
/ for quotients
^ for powers

Logical operators: interpretation from left to right
& logical AND
½ logical OR
>> shift right
<< shift left
XOR exclusive OR
~ logical NOT

Trigonometric functions:
sin(x), con(x), tan(x)
arcsin(x), arccos (x), arctan (x)
sinh(x), cosh(x), tanh(x)

Exponential function:
exp(x)

Logarithmic functions:
ln(x)
log(x)

Square root, absolute amount:
sqrt(x)
abs(x)

Optional parameters:

-> FORMULA_INV:	function to calculate the hexadecimal, 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 application 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 hexadecimal (ECU internal) format into the physical format. The interpretation of the formula must be supported by a formula interpreter in the operating system.

Example:

FORMULA	"sqrt(3 - 4*(sin(X1))^2)"

5.3.27 FORMULA_INV

Prototype:

FORMULA_INV	g(x)

Parameters:

string g(x):	function for calculation of the hexadecimal, 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/differenc, are taken into account. Brackets are allowed. Permissible operation symbols: see keyword FORMULA, page 51.

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 hexadecimal (ECU internal) format. The interpretation of the formula must be supported by a formula interpreter in the operating system.

Example:

Inversion function e.g. for keyword FORMULA (see page 51)

FORMULA_INV	"arcsin(sqrt((3 - (X1)^2)/4))"

5.3.28 FUNCTION

Prototype:

/begin FUNCTION Name Long-Identifier
/end FUNCTION

Parameters:

ident name:	Identifier in the program, referencing is based on this 'name'
string long-identifier:	comment, description

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 application system to allow the selection lists for the selection of adjustable objects and measuring channels to be represented in a structured manner on the basis of functional viewpoints (function orientation).

Example:

```
/begin FUNCTION
                    ID_ADJUSTM                          /* name */
                    "function group idling adjustment"  /* long identifier */
/end FUNCTION
```

5.3.29 FUNCTION_LIST

Prototype:

FUNCTION_LIST (Name) *

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).

Example:

```
FUNCTION_LIST ID_ADJUSTM FL_ADJUSTM SPEED_LIM
```

5.3.30 HEADER

Prototype:

```
/begin HEADER                   Comment
                                        [-> VERSION]
                                        [-> PROJECT_NO]
/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 ASAP devices.

Example:

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

5.3.31 IDENTIFICATION

Prototype:

IDENTIFICATION	Position Datatype

Parameters:

int position:	position of the 'identifier' in the deposit structure.
datatype data type:	word length of the 'identifier''

Description:

Description of an 'identifier' in an adjustable object (see BOSCH: C-DAMOS deposit).

Example:

IDENTIFICATION	1 UWORD

5.3.32 MAX_GRAD

Prototype:

MAX_GRAD max_gradient

Parameters:

float max_gradient: 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($(W_{ik} - W_{i-1,k})/(X_i - X_{i-1})$)

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

Example:

MAX_GRAD 200.0

5.3.33 MAX_REFRESH

Prototype:

MAX_REFRESH	Value Addition

Parameters:

float value:	value in 'miliseconds' or value in 'grad crankshaft'
enum addition:	this parameter is used to define the 'value' parameter. Possible values are: MSEC = value in 'milliseconds' GRAD_CS = value in 'grad crankshaft'

Description:

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

Example:

MAX_REFRESH	10.0 MSEC

5.3.34 MEASUREMENT

Prototype:

```
/begin MEASUREMENT      Name Long-Identifier Datatype Conversion Resolution
                             Accuracy Lower-Limit Upper-Limit
                             [-> BIT_MASK]
                             [-> BYTE_ORDER]
                             [-> MAX_REFRESH]
                             [-> VIRTUAL]
                             [-> FUNCTION_LIST]
                             {-> IF_DATA } *
/end MEASUREMENT
```

Parameters:

ident name:	identifier in the program
string long identifier:	comment, description
datatype data type:	data type of the measurement
ident conversion:	reference to the relevant data record for description of the conversion method (see COMPU_METHOD)
int resolution:	smalles possible change float in bits
float accuracy:	possible variation from exact value in %
float lower limit:	plausible range of table values, lower limit
float upper limit:	plausible range of table values, upper limit

Optional parameters

-> BIT_MASK:	With deviation from the standard value 0xFFFFFFFF this parameter can be used to mask out bits.
-> BYTE_ORDER:	With deviation from the standard value this parameter can be used to specify the byte order (Intel format, Motorola format)
-> MAX_REFRESH:	Maximum refresh rate of this measurement in the control unit
-> VIRTUAL:	For description of a virtual measurement (see VIRTUAL)
-> FUNCTION_LIST:	This keyword can be used to specify a list of 'functions' to which this measurement object has been allocated.
-> IF_DATA:	Date record to describe the interface specific description data. The parameters associated with this keyword are described in A2ML by the control unit supplier or the interface module supplier.

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 */
                      BIT_MASK                          0x0FFF
                      BYTE_ORDER                        LITTLE_ENDIAN
                      FUNCTION_LIST               ID_ADJUSTM  FL_ADJUSTM
                      IF_DATA ISO                       SND 0x10 0x00 0x05 0x08 RCV 4 long
/end MEASUREMENT
```

5.3.35 MEMORY_LAYOUT

Prototype:

MEMORY_LAYOUT	prg_typ addr size offset

Parameters:

enum prg_typ:	Description ofthe program segments divided into: PRG_CODE = program code PRG_DATA = program data PRG_RESERVED = other
long addr:	Initial address of the program segment to be described.
long size:	Length of the program segment to be described.
long[5] offset:	BOSCH feature: In special ECU programs, so-called 'mirrored segments' may occur (seeFigure 8). 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.

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:

See also Figure 8.

```
MEMORY_LAYOUT PRG_RESERVED    0x0000   0x0400   -1       -1       -1   -1   -1
MEMORY_LAYOUT PRG_CODE        0x0400   0x3C00   -1       -1       -1   -1   -1
MEMORY_LAYOUT PRG_DATA        0x4000   0x0200   0x10000  0x20000  -1   -1   -1
MEMORY_LAYOUT PRG_DATA        0x4200   0x0E00   -1       -1       -1   -1   -1
MEMORY_LAYOUT PRG_DATA       0x14200   0x0E00   -1       -1       -1   -1   -1
MEMORY_LAYOUT PRG_DATA       0x24200   0x0E00   -1       -1       -1   -1   -1
```

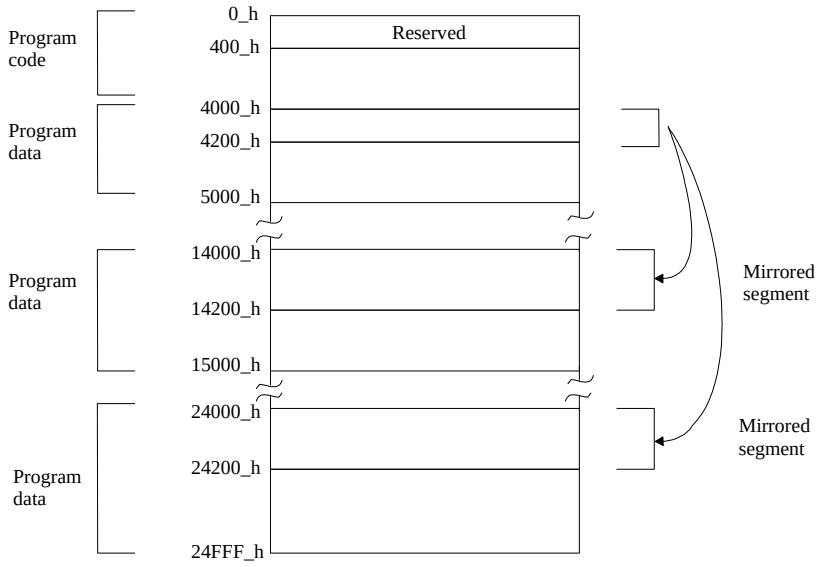


Figure 8 Memory layout (mirrored segments)

5.3.36 MODULE

Prototype:

```
/begin MODULE                   Name Long-Identifier
                                [-> A2ML]
                                [-> MOD_PAR]
                                [-> MOD_COMMON]
                                {-> IF_DATA}*
                                {-> CHARACTERISTIC}*
                                {-> AXIS_PTS}*
                                {-> MEASUREMENT}*
                                {-> COMPU_METHOD}*
                                {-> COMPU_TAB}*
                                {-> COMPU_VTAB}*
                                {-> FUNCTION}*
                                {-> RECORD_LAYOUT}*
/end MODULE
```

Parameters:

ident name:	ASAP device identifier
string long-identifier:	comment, description

Optional parameters:

-> A2ML:	Format description of the interface-specific parameters. Attention: The interface-specific parameters must be specified directly after the last mandatory parameter 'long-identifier'. /begin MODULE ENGINE_ECU "Comment" /begin A2ML /end A2ML : /end MODULE
-> MOD_PAR:	Keyword for the description of module-specific (ASAP device-specific) management data.
-> MOD_COMMON:	Module-wide description data
-> IF_DATA:	Data record for the description of interface-specific description data (BLOB: binary large object). The parameters associated with this keyword are described in the ASAP2 metalanguage (in short A2ML) by the ECU supplier or the interface module supplier.
-> CHARACTERISTIC:	Keyword for the description of the adjustable objects
-> AXIS_PTS:	Keyword for the description of the axis points
-> MEASUREMENT:	Keyword for the description of the measurement 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
-> FUNCTION:	Keyword for the description of the functions
-> RECORD_LAYOUT:	Keyowrd for the description of the record layouts

Description:

The MODULE keyword describes a complete ASAP 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.

Example:

see part C: file engine_ecu.a2l, abs_ecu.a2l

5.3.37 MOD_COMMON

Prototype:

```
/begin MOD_COMMON                Comment
                                 [-> S_REC_LAYOUT]
                                 [-> DEPOSIT]
                                 [-> BYTE_ORDER]
                                 [-> DATA_SIZE]
/end MOD_COMMON
```

Parameters:

string comment:	comment

Optional parameters:

-> S_REC_LAYOUT:	Reference to the standard record layout
-> DEPOSIT:	Standard deposit mode for axis points: ASBOLUTE or DIFFERENCE
-> BYTE_ORDER:	Byte order
-> DATA_SIZE:	Data size in bits

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         SIEMENS_ABL
      DEPOSIT                    ABSOLUTE
      BYTE_ORDER                 BIG_ENDIAN
      DATA_SIZE                  16
/end MOD_COMMON
```

5.3.38 MOD_PAR

Prototype:

```
/begin MOD_PAR          Comment
                            [-> VERSION]
                            [-> ADDR_EPK]
                            [-> EPK]
                            [-> SUPPLIER]
                            [-> CUSTOMER]
                            [-> CUSTOMER_NO]
                            [-> USER]
                            [-> PHONE_NO]
                            [-> ECU]
                            [-> CPU_TYPE]
                            [-> NO_OF_INTERFACES]

                            {-> MEMORY_LAYOUT}*
                            {-> SYSTEM_CONSTANT}*
/end MOD_PAR
```

Parameters:

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

Optional parameters:

-> VERSION:	Version identifier
-> ADDR_EPK:	Address of EPROM identifier
-> EPK:	EPROM identifier
-> SUPPLIER:	Manufacturer or supplier
-> CUSTOMER:	Firm or customer
-> CUSTOMER_NO:	Customer number
-> USER:	User
-> PHONE_NO:	Phone number of the applications engineer responsible
-> ECU:	Control unit
-> CPU_TYPE:	CPU
-> NO_OF_INTERFACES:	Number of interfaces
-> MEMORY_LAYOUT:	Memory layout
-> SYSTEM_CONSTANT:	System-defined constants

Description:

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

Example:

```
/begin MOD_PAR              "Please note: provisional release for test purposes only!"
        VERSION                     "Test version of 01.02.1994"
        ADDR_EPK                    0x45678
        EPK                         EPROM identifier test
        SUPPLIER                    "Mustermann"
        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
        MEMORY_LAYOUT               PRG_RESERVED 0x0000 0x0400 -1 -1 -1 -1 -1
        MEMORY_LAYOUT               PRG_CODE 0x0400 0x3C00 -1 -1 -1 -1 -1
        MEMORY_LAYOUT               PRG_DATA 0x4000 0x5800 -1 -1 -1 -1 -1
        SYSTEM_CONSTANT             "CONTROLLERx constant1" "0.33"
        SYSTEM_CONSTANT             "CONTROLLERx constant2" "2.79"
/end MOD_PAR
```

5.3.39 MONOTONY

Prototype:

MONOTONY	Monotony

Parameters:

enum monotony:	Description of the monotony:	
	MON_INCREASE:	monotonously increasing
	MON_DECREASE:	monotonously decreasing

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 application system (user interface) verifies whether the monotony is guaranteed. Changes that do not correspond to the monotony are not allowed.

Example:

MONOTONY	MON_INCREASE

5.3.40 NO_AXIS_PTS_X, NO_AXIS_PTS_Y

Prototype:

NO_AXIS_PTS_X	Position Datatype
or	
NO_AXIS_PTS_Y	Position Datatype

Parameters:

int position:	Position of the number of axis points in the deposit structure
datatype data type:	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

5.3.41 NO_OF_INTERFACES

Prototype:

NO_OF_INTERFACES	Num

Parameters:

int num:	Number of interfaces

Description:

Keyword for the number of interfaces

Example:

NO_OF_INTERFACES	2

5.3.42 NUMBER

Prototype:

NUMBER	Number

Parameters:

int 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.

Example:

NUMBER	7

5.3.43 OFFSET_X, OFFSET_Y

Prototype:

OFFSET_X	Position Datatype
or	
OFFSET_Y	Position Datatype

Parameters:

int position:	Position of the 'offset' parameter in the deposit structure to compute the X-axis points for fixed characteristic curves and fixed characteristic maps.
datatype data type:	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:

$X_i = offset + (i - 1)*2^{shift}$ i = { 1...numberofaxispts }

or

$Y_k = offset + (k - 1)*2^{shift}$ k = { 1...numberofaxispts }

Example:

OFFSET_X	16 UWORD

5.3.44 PHONE_NO

Prototype:

PHONE_NO	Telnum

Parameters:

string telnum:	phone number

Description:

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

Example:

PHONE_NO	"09498 594562"

5.3.45 PROJECT

Prototype:

```
/begin PROJECT          Name Long-Identifier
                            [-> HEADER]
                            {-> MODULE}*
```

Parameters:

ident name:	Project identifier in the program
string long-identifier:	Comment, description

Optional parameters:

-> HEADER:	Project header
-> MODULE:	This keyword is used to describe the module (ASAP device) belonging to the project.

Description:

Project description with project header and all ASAP 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
```

5.3.46 PROJECT_NO

Prototype:

PROJECT_NO	Project number

Parameters:

ident projectnumber:	Short identifier of the project number

Description:

String used to identify the project number with maximum MAX_IDENT (at present MAX_IDENT = 10) characters.

Example:

PROJECT_NO	M4711Z1

5.3.47 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 */
        IF_DATA "DIM"               EXTERNAL DIRECT
        FUNCTION_LIST       V_LIM                   /*Reference to functions */
        READ_ONLY
/end CHARACTERISTIC
```

5.3.48 RECORD_LAYOUT

Prototype:

```
/begin RECORD_LAYOUT        Name
                                [-> FNC VALUES]
                                [-> IDENTIFICATION]
                                [-> AXIS_PTS_X]
                                [-> AXIS_PTS_Y]
                                [-> NO_AXIS_PTS_X]
                                [-> NO_AXIS_PTS_Y]
                                [-> FIX_NO_AXIS_PTS_X]
                                [-> FIX_NO_AXIS_PTS_Y]
                                [-> SRC_ADDR_X]
                                [-> SRC_ADDR_Y]
                                [-> RIP_ADDR_X]
                                [-> RIP_ADDR_Y]
                                [-> SHIFT_OP_X]
                                [-> SHIFT_OP_Y]
                                [-> OFFSET_X]
                                [-> OFFSET_Y]
                                {-> RESERVED}*
/end RECORD_LAYOUT
```

Parameters:

ident name: Identification of the record layout, which is referenced via this 'name'.

Optional parameters:

-> 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' (see BOSCH: C-DAMOS) is deposited in a specific position in the adjustable object.

only characteristic curves or characteristic maps:

-> AXIS_PTS_X: This keyword describes where the X-points are deposited in memory.

-> AXIS_PTS_Y: This keyword describes where the Y-points are deposited in memory.

-> NO_AXIS_PTS_X: This keyword describes in which position the parameter 'number of X-axis points' is deposited in memory.

-> NO_AXIS_PTS_Y: This keyword describes in which position the parameter 'number of Y-axis points' is deposited in memory.

-> FIX_NO_AXIS_PTS_X: This keyword indicates that all characteristic curves or characteristic maps relating to the X-axis points are allocated a fixed number of axis points. In a RECORD_LAYOUT data record, this keyword may not be used simultaneously with the keyword 'NO_AXIS_PTS_X(!!).

-> FIX_NO_AXIS_PTS_Y: This keyword indicates that all characteristic curves or characteristic maps relating to the Y-axis points are allocated a fixed number of axis points. In a RECORD_LAYOUT data record, this keyword may not be used simultaneously with the keyword 'NO_AXIS_PTS_Y (!!).

-> SRC_ADDR_X: This keyword describes in which position the address of the input quantity of the X-axis points is deposited in memory.

-> SRC_ADDR_Y: This keyword describes in which position the address of the input quantity of the Y-axis points is deposited in memory.

-> RIP_ADDR_X: Future record layouts: 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 (X_i, X_{i+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 quantity (Result of Inter- polation), which describes the relative distance between the current value and the adjacent axis points:
$RIP\_X = (X(t) - X_i)/(X_{i+1} - X_i)$.

This keyword is used to describe in which position the address of this RIP_X quantity is deposited, which contains the current value of the ECU-internal interpolation.

-> RIP_ADDR_Y: See RIP_ADDR_Y, but for Y-axis points.

-> RIP_ADDR_W: Final result (table value) of the ECU-internal inter- polation.

only for fixed characteristic curves or fixed characteristic maps (at the request of Mr Hünerfeld):

-> SHIFT_OP_X: Shift operand to compute the X-axis points

-> SHIFT_OP_Y: Shift operand to compute the Y-axis points

-> OFFSET_X: Offset to compute the X-axis points

-> OFFSET_Y: Offset to compute the Y-axis points

-> RESERVED: This keyword can be used to skip specific elements in the adjustable object whose meaning must not be interpreted by the application system (e.g. for extensions: new parameters in the adjustable objects).

Description:

The 'RECORD_LAYOUT' keyword is used to specify the various *record layouts* 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 application system will be able to process all adjustable object types (reading, writing, operating point display etc.).

Important:

To describe the recordlayouts, 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 ASAP2 description file format must be modified accordingly.

Example:

```
/begin RECORD_LAYOUT        DAMOS_KF
        FNC_VALUES                  7 SWORD COLUMN_DIR DIRECT
        AXIS_PTS_X                  3 SWORD INDEX_INCR ABSOLUT DIRECT
        AXIS_PTS_Y                  6 UBYTE INDEX_INCR ABSOLUT DIRECT
        NO_AXIS_X                   2 UBYTE
        NO_AXIS_Y                   5 UBYTE
        SRC_ADDR_X                  1
        SRC_ADDR_Y                  4
/end RECORD_LAYOUT
```

5.3.49 RESERVED

Prototype:

RESERVED	Position Datatype

Parameters:

int position:	Position of the reserved parameter in the deposit structure
datasize data type:	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 application system (e.g. for extensions: new parameters in the adjustable objects).

Example:

RESERVED	7 LONG

5.3.50 RIP_ADDR_X, RIP_ADDR_Y, RIP_ADDR_W

Prototype:

RIP_ADDR_X	Position
or	
RIP_ADDR_Y	Position
or	
RIP_ADDR_W	Position

Parameters:

int position:	Position of the address to the result of the ECU-internal interpolation (see below) in the deposit structure.

Description:

The description of this parameter should be based on the example of a characteristic curve (RIP: Result of Interpolation).
When the ECU program accesses the characteristic curve it first determines the adjacent axis points of the current value of the input quantity (see Figure 9: X_i, X_{i+1}). 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.

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

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

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

and for a characteristic map as follows:

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

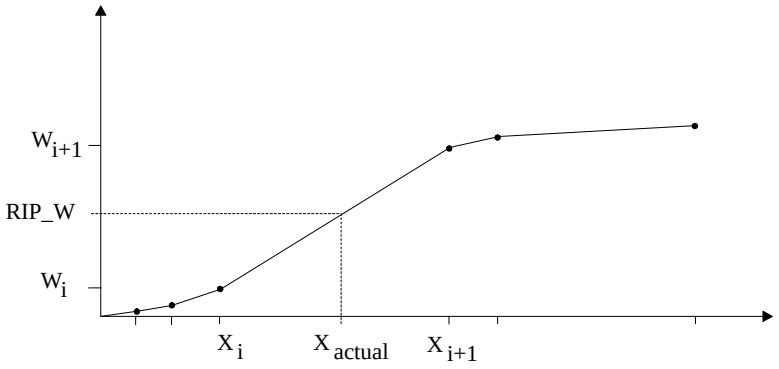


Figure 9 Linear interpolation for a characteristic curve

Example:

RIP_ADDR_X 19

5.3.51 SHIFT_OP_X, SHIFT_OP_Y

Prototype:

SHIFT_OP_X	Position Datatype
or	
SHIFT_OP_Y	Position Datatype

Parameters:

int position:	Position of the shift operand in the deposit structure.
data type data type:	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:

$X_i = offset + (i - 1)*2^{shift}$ i = { 1...numberofaxispts }

or

$Y_k = offset + (k - 1)*2^{shift}$ k = { 1...numberofaxispts }

Example:

SHIFT_OP_X	21 UWORD

5.3.52 SRC_ADDR_X, SRC_ADDR_Y

Prototype:

SRC_ADDR_X	Position
or	
SRC_ADDR_Y	Position

Parameters:

int position:	Position of the address of the input quantity in the deposit structure. The data type of this address is specific to the adjustable object and contained in the corresponding measuring channel data record of the description file.

Description:

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

Example:

SRC_ADDR_X	1

5.3.53 SUPPLIER

Prototype:

SUPPLIER	Manufacturer

Parameters:

string supplier:	Name of the ECU manufacturer

Description:

String used to identify the manufacturer or supplier.

Example:

SUPPLIER	"Smooth and Easy"

5.3.54 SYSTEM_CONSTANT

Prototype:

SYSTEM_CONSTANT Name Value

Parameters:

string name: system constant identifier
string value: value of the system constant as a string

Description:

System-defined constant.

Example:

SYSTEM_CONSTANT "CONTROLLER_CONSTANT12" "2.7134"

5.3.55 S_REC_LAYOUT

Prototype:

S_REC_LAYOUT Name

Parameters:

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

Description:

This keyword can be usedto 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.

Example:

```
S_REC_LAYOUT SIEMENS_ABL /* Siemens record layout */
```

5.3.56 USER

Prototype:

USER	User name

Parameters:

user name string	Name of the user

Description:

Specification of the user name.

Example:

USER	"Nigel Hurst"

5.3.57 VERSION

Prototype:

VERSION Version identifier

Parameters:

string version identifier: short identifier for the version

Description:

String for identification of the version with maximum MAX_LEN (at present MAX_LEN = 100) characters.

Example:

VERSION "BG5.0815"

5.3.58 VIRTUAL

Prototype:

VIRTUAL (Measuring channel) *

Parameters:

measuring channel ident	Reference to a fixed value (CHARACTERISTIC) or 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.

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 */
        VIRTUAL                 PHI_BASIS  PHI_CORR
/end MEASUREMENT

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

6 Include mechanism

6.1.1 Description of complex projects

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

/include <filename>

This statement allows several description files to be integrated into one project description.

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 ENG_ECU.A2L
   /include ABS_ECU.A2L
   /include SPEC_ECU.A2L
/end
_____________________________________________ End of file PROJECT1.A2L _____________________________
```

6.1.2 Description of interface-specific parameters

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 ASAP2 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
```

PART C: ASAP2 METALANGUAGE

7 Interface-specific description data

Between the control part of the standardised application system and the program parts for access to the application interface an interface (ASAP 1b) has been defined. The program parts for access to the application interface shall in future be realised as linkable drivers.
Furthermore, the description data in the application system are divided into two categories:

1) Parameters that are used by the control interface.
2) Parameters that are only analysed 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 application system but simply by incoporating a new driver.

For the description of the *interface-specific parameters* a description language (ASAP2 metalanguage, in short **A2ML**) 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 A2ML) the standardised application system must be capable of reading in the *interface-specific parameters* of the description file and transferring them to the drivers (see Figure 10).

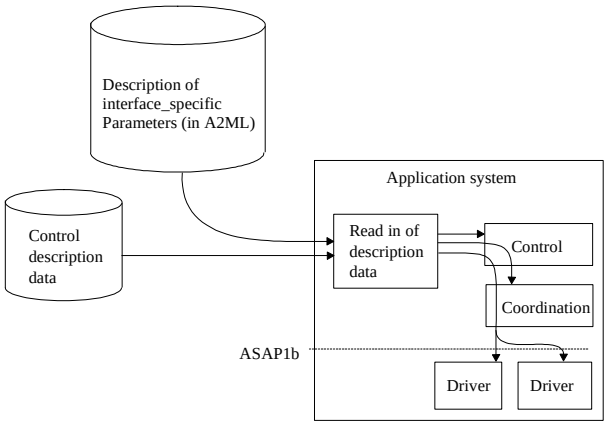


Figure 10 Schematic data flow of description data

7.1 Format of the ASAP2 metalanguage

To describe the grammar of the *ASAP2 metalanguage*, an extended Backus-Naur format is used:

- A non-terminal is represented as a simple identifier:

  block_definition

- The symbol used as inference symbol in the production rules is '::='

  type_definition ::= type_name

- A terminal (keyword) is enclosed within quotes:

  "struct"

- Non-formally defined parts are enclosed within angle brackets. In the description given hereafter the following two identifiers are used in particular:

  <keyword>: <keyword> is used to define the keywords of the ASAP2 description file by means of a character sequence enclosed within double inverted commas.
  <identifier>: identifier for the definition of data structures.

- An optional part is enclosed within square brackets:

  [<identifier>]

- Alternative parts are separated by '|' :

  "char" | "int" | "long"

- Explanations are enclosed within comment symbols:

  /* comment */

7.1.1 Grammar in the extended Backus-Naur format

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_name ::= predefined_type_name | struct_type_name | taggedstruct_type_name | taggedunion_type_name | enum_type_name

predefined_type_name ::= "char" | "int" | "long" | "uchar" | "uint" | "ulong" | "double" | "float"

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.

block_definition ::= "block" <keyword> type_name

Definition of an enumeration:

enum_type_name ::= "enum" [<identifier>] "{" enumerator_list "}" | "enum" <identifier>
enumerator_list ::= enumerator | enumerator enumerator_list
enumerator ::= <keyword> ["=" <constant>]

Definition of data records of the ASAP2 description file with fixed sequence of the data record elements.

struct_type_name ::= "struct" [<identifier>] "{struct_member_list '}' | "struct' <identifier>

struct_member_list ::= struct_member | struct_member struct_member_list

struct_member ::= member ";"

member ::= type_name [array_specifier]

array_specifier ::= "[" <constant> "]" | "[" <constant> "]" array_specifier

Definition of data records of the ASAP2 description file whose elements can specified in a random sequence. All elements are optional and each element is identified by its own keyord (see <keyword>).
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_type_name ::= "taggedstruct" [<identifier>] "{" taggedstruct_member_list "}" |
"taggedstruct" <identifier>

taggedstruct_member_list ::= taggedstruct_member |
taggedstruct_member taggedstruct_member_list

taggedstruct_member ::= taggedstruct_definition ";" | "(" taggedstruct_definition ")* ;" |
block_definition ";" | "(" block_definition ")* ;"

taggedstruct_definition ::= <keyword> member | <keyword> "(" member ")*"

Definition of variants in data records of the ASAP2 description file. Similar to the 'union' data type used in programming language C, the ASAP2 description file allows only one variant to be specified at a time in a 'taggedunion'. Each variant is assigned a keyword for identification purposes (see <keyword>).

```
taggedunion_type_name   ::=     "taggedunion" [ <identifier> ] "{" taggedunion_member_list "}" |
                                "taggedunion" <identifier>

taggedunion_member_list ::= tagged_union_member |
                                    tagged_union_member taggedunion_member_list

taggedunion_member      ::= <keyword> member ";" | block_definition ";"
```

7.2 Example of ASAP2 metalanguage

The following example illustrates the ASAP2 metalanguage (A2ML) on the basis of the format definition.

Remark:
The ASAP2 metalanguage shall be used to specify the format of the interface-specific parameters. The following format specification for the complete ASAP2 description file is only intended as an **example** to illustrate the ASAP2 metalanguage.

File EXAMPLE.AML

```
block  "PROJECT"  struct project;

struct project (
  struct (                                          /* mandatory part */
     char[20];                                              /* name */
     char[100];                                     /* long identifier */
  }
  taggedstruct {                                            /* optional part */
     block "HEADER" struct projectheader;
     ( block "MODULE" struct modul )*;              /* block "MODULE" can occur several times */
  }
};

struct modul {
   struct                                           /* mandatory part */
     char[20];                                              /* name */
     char[100];                                     /* description */
  }
  taggedstruct {                                            /* optional part */
     block  "MOD_PAR"           struct mod_par;
     block  "MOD_COMMON"                struct mod_common;
                                                    /* taggedunion interface_param: to be specified by manufacturer */
     ( block  "IF DATA"                 taggedunion interface_param )*;
     ( block  "CHARACTERISTIC"          struct characteristic )*;   /* (0...n) times  */
     ( block  "AXIS_PTS"                struct axis_pts )*;
     ( block  "MEASUREMENT"             struct measurement )*;
     ( block  "COMPU_METHOD"            struct conversion )*;
     ( block  "COMPU_TAB"       struct conv_tab )*;
     ( block  "COMPU_VTAB"      struct conv_tab_verb )*;
     ( block  "FUNCTION"        struct function )*;
     ( block  "RECORD_LAYOUT"           taggedstruct record_layout )*;
  };
};

/* ******************************** project header ************************** */
struct projectheader {
   char[100];                                        /* comment */
   taggedstruct {
      "VERSION"         char[100];
      "PROJECT_NO"      char[100];
   };
};

/* ***************************** module description ************************** */
struct memory_loc {
   enum prg_type { "PRG_CODE" = 0, "PRG_DATA" = 1, "PRG_RESERVED = 2 };     /* prg_type */
   long;                                                    /* addr */
   long;                                                    /* size */
   long[5]                                                  /* offset */
};

struct sdk {
   char[20]                                                 /* label_name */
   char[20]                                                 /* value */
};
```

```
struct mod_par {
   char[100];                                                    /* comment */
   taggedstruct {                                          /* optional part */
      "VERSION"            char[100];                      /* data status */
      "ADD_EPK"            long;
      "EPK"                char[100];
      "SUPPLIER"           char[100];
      "CUSOTMER"                 char[100];
      "CUSTOMER_NO"        char[40];
      "USER"                     char[40];                       /* applications engineer */
      "PHONE_NO"                 char[40];
      "ECU"                char[40];
      "CPU_TYPE"           char[40];
      "NO_OF_INTERFACES"  uint;
      ( "MEMORY_LAYOUT"    struct memory_loc )*;
      ( "SYSTEM_CONSTANT" struct sdk )*;
   };
};

enum endian { "BIG_ENDIAN" = 2, "LITTLE_ENDIAN" = 1 };

enum datatype { "UBYTE" = 0, "SBYTE" = 1, "UWORD" = 2, "SWORD" = 3, "ULONG" = 4, "SLONG" = 5 };

/* ******************** control unit description ******************** */
struct mod_common {
   char[100];                                                    /* comment */
   taggedstruct {
      "S_REC_LAYOUT"             char[20];
      "DEPOSIT"            enum {  "DIFFERENCE" = 0,  "ABSOLUTE" = 1 };
      "BYTE_ORDER"               enum endian;
      "DATA_SIZE"          int;                            /* 8/16 bit */
   };
};

/*×********************* description of characteristics ********************* */
struct axis_descr {
   struct {                                                      /* mandatory part */
      enum { "STD_AXIS" = 0, "COM_AXIS" = 1, "FIX_AXIS" = 2 };
      char[20];                                                  /* input quantity */
      char[20];                                                  /* conversion */
      int;                                                /* maximum number of axis points */
      double;                                                    /* lower limit */
      double;                                                    /* upper limit */
   };
   taggedstruct {                                          /* optional part */
      "MAX_GRAD" double;                                         /* maximum gradient */
      "MONOTONY" enum { "MON_INCREASE" = 0, "MON_DECREASE" = 1 }; /* monotony */
      "BYTE_ORDER" enum endian;                                  /* byte order */
      "FIX_AXIS_PAR" struct {                                    /* parameters of fixed characteristic curve/map */
         int;                                                    /* offset */
         int;                                                    /* shift */
         int;                                                    /* number of axis points */
      };
      "DEPOSIT" enum { "DIFFERENCE" = 0,  "ABSOLUTE" = 1 };  /* with deviation from standard */
      "AXIS_PTS_REF" char[20];                                   /* reference to axis points distribution */
   };
};
```

```
struct characteristic {
   struct {                                                    /* mandatory part */
      char[20];                                                /* name      */
      char[120];                                               /* descr     */
      enum { "VALUE"=0, "CURVE"=1, "MAP"=2, "CUBOID"=3, "VAL_BLK"=4, "ASCII"=5};    /* type */
      long;                                                    /* address   */
      char[20];                                                /* deposit */
      double;                                                  /* maxdiff */
      char[20];                                                /* conversion */
      double;                                                  /* lower limit */
      double;                                                  /* upper limit */
   };
   taggedstruct {                                              /* optional part */
      "BYTE_ORDER" enum endian;                                /* byte order */
      "BIT_MASK"  long;                                        /* bit mask */
      "FUNCTION_LIST" ( char[20] )*;                           /* functions */
      "NUMBER" int;                                            /* number of constants or characters */
      "EXTENDED_LIMITS" struct {
         double;                                               /* extended lower limit */
         double;                                               /* extended upper limit */
       };
      ( block "AXIS_DESCR" struct axis_descr )*;               /* axis description */
      ( "IF_DATA" taggedunion ifp_characteristic )*;           /* interface-specific part */
   };                                                          /* taggedunion ifp_characteristic: to be specified by manufacturer */
};

/* ********** Description of axis point distributions ********** */
struct axis_pts {
   struct {                                                    /* mandatory part */
      char[20];                                                /* name      */
      char[120];                                               /* descr     */
      long;                                                    /* address   */
      char[20];                                                /* input quantity */
      char[20];                                                /* deposit */
      double;                                                  /* maxdiff */
      char[20];                                                /* conversion */
      int;                                                     /* maximum number of axis points */
      double;                                                  /* lower limit */
      double;                                                  /* upper limit */
   };
   taggedstruct {                                              /* optional part */
      "DEPOSIT" enum {  "DIFFERENCE" = 0,  "ABSOLUTE" = 1 };
      "BYTE_ORDER" enum endian;                                /* byte order */
      "FUNCTION_LIST" ( char[20] )*;                           /* functions */
      ( "IF_DATA" taggedunion ifp_characteristic )*;           /* interface-specific part */
   };                                                          /* taggedunion ifp_characteristic: to be specified by manufacturer */
};

/* ********** Description of measurements ********** */
struct measurement {
   struct {                                                    /* mandatory part */
      char[20];                                                /* name      */
      char[120];                                               /* descr     */
      enum datatype;                                           /* data type */
      char[20];                                                /* conversion method */
      int;                                                     /* resolution in bits */
      double;                                                  /* accuracy */
      double;                                                  /* lower limit */
      double;                                                  /* upper limit */
   };
   taggedstruct {                                              /* optional part */
      "BIT_MASK"        long;                                  /* bit mask */
      "BYTE_ORDER"      enum endian;                           /* byte order */
      "FUNCTION_LIST" ( char[20] )*;                           /* functions: keyword only once,
                                                                  followed by list of function names */
      "MAX_REFRESH    struct {
         double;                                               /* value */
         enum { "MSEC" = 0, "GRAD_CS" = 1 };                   /* value in 'msec' or in 'crankshaft grad' */
      };
      "VIRTUAL"         ( char[20] )*;                         /* list of measurements to be linked */
                                                               /* taggedunion ifp_measurement: to be specified by manufacturer */
      ( "IF_DATA" taggedunion ifp_measurement )*;              /* interface-specific part */
   };
};
```

```
/* ********************* Description of conversion method ******************* */
struct conversion {
   struct {                                          /* mandatory part */
      char[20];                                      /* name      */
      char[120];                                     /* descr     */
      enum { "TAB_INTP" = 0, "TAB_NOINTP" = 1, "RAT_FUNC" = 2, "FORM" = 3};     /* type */
      char[30];                                      /* format (number of digits or fractional digits) */
      char[30];                                      /* unit        */
   }
   taggedstruct {
      "COEFFS" double[6];                            /* coefficients  */
      "COMPU_TAB" char [20]                          /* reference to table */
      "FORMULA" char [100]                           /* formula */
   }
};

/* ******************* Description of conversion tables ********************** */
struct conv_tab {
   char[20];                                         /* name          */
   char[120];                                        /* descr    */
   enum { "TAB_INTP" = 0, "TAB_NOINTP" = 1};         /* type      */
   int;                                              /* value_pairs_no */
   struct tab {
      long;                                          /* int_val       */
      double;                                        /* phys_val      */
   } [256];                                          /* tab           */
};

/* ************** Description of verbal conversion tables ************* */
struct conv_tab_verb {
   char[20];                                         /* name          */
   char[120];                                        /* descr    */
   int;                                              /* type          */
   int;                                              /* value_pairs_no */
   struct tab {
      long;                                          /* int_val       */
      char[40];                                      /* text          */
   } [20];
};

/* ************* Description of functions ************** */
struct function {
   char[20];                                         /* name      */
   char[120];                                        /* descr     */
};


/* ********************** Description of record layouts ****************** */
enum addrtyp { "PBYTE" = 1, "PWORD" = 2, "PLONG" = 3, "DIRECT" = 4 };

struct fnc_values {
   int no;
   enum datatyp;                                     /* data type of table values */
   enum { "COLUMN_DIR" = 1, "ROW_DIR" = 2 };         /* row or column oriented */
   enum addrtyp;                                     /* addressing */
};

struct axis_pts {
   int no;
   enum datatyp;                                     /* data type of axis points */
   enum { "INDEX_INCR" = 1, "INDEX_DECR" = 2 };      /* increasing, decreasing index */
   enum addrtyp;                                     /* addressing */
};

struct abl_addr {
   int no;
};

struct abl_datatyp {
   int no;
   enum datatype;                                    /* data type */
};
```

```
struct record_layout {
   struct {                                                    /* optional part */
      char[20];                                          /* name */
   };
   taggedstruct {                                              /* mandatory part */
      "FNC_VALUES"              struct fnc_values;
      "IDENTIFICATION"          struct abl_datatyp;
      ("RESERVED"          struct abl_datatyp) *;
      "AXIS_PTS_X"         struct axis_pts;
      "AXIS_PTS_Y"         struct axis_pts;
      "NO_AXIS_PTS_X"           struct abl_datatyp;
      "NO_AXIS_PTS_Y"           struct abl_datatyp;
      "FIX_NO_AXIS_PTS_X"       int;                          /* number of axis points */
      "FIX_NO_AXIS_PTS_Y"       int;                          /* number of axis points */
      "SCR_ADDR_X"              struct abl_addr;
      "SCR_ADDR_Y"              struct abl_addr;
      "RIP_ADDR_X"              struct abl_addr;
      "RIP_ADDR_Y"              struct abl_addr;
      "SHIFT_OP_X"         struct abl_datatyp;
      "SHIFT_     OP_Y"         struct abl_datatyp;
      "OFFSET_X"           struct abl_datatyp;
      "OFFSET_Y"           struct abl_datatyp;
   };
};
```

End File EXAMPLE.AML

7.3 Example of description file

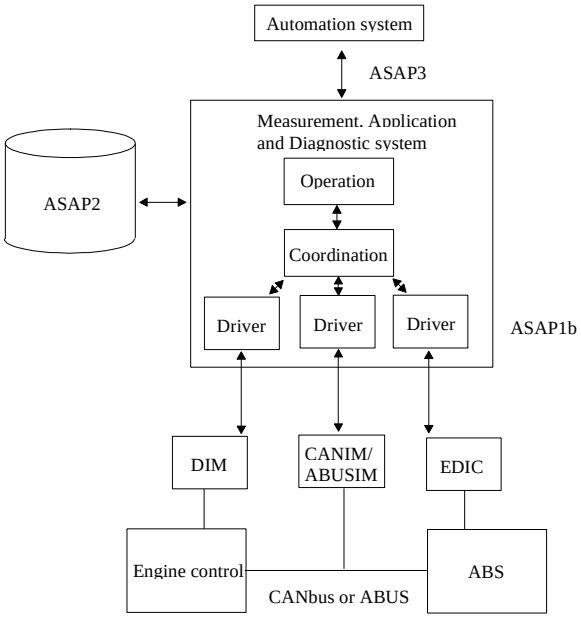


Figure 11 Project example

On the basis of the example shown in Figure 11 it is assumed that the drivers for DIM and CANIM are supplied by supplier 1 and the driver for EDIC by supplier 2. Both suppliers must then specify the formats for the interface-specific parameters. This is done below in the two files SUPP1_IF.AML and SUPP2.IF.AML.

File SUPP1_IF.AML

```
/begin A2ML
   /* ********************* General interface-specific parameters ****************** */
   struct dtab_par {
      long;                                                    /* adr_distab */
      int;                                                     /* len_distab */
      long;                                                    /* addr_outp  */
      long;                                                    /* trgid      */
      char[40];                                                /* comment    */
   }

   taggedunion interface_param {
      "DIM"  taggedstruct  {
               "DIM_DISPLAY_TAB"  int;
               ( "DIM_DISPLAY_PAR"  struct dtab_par )*;        /* (0...n)times */
            };
      "CAN"  taggedstruct  {
               "CAN_BT"  int;
            };
   };
```

```
   /* ***************** interface-specific parameters for measurements ************* */
   enum mem_typ {  "INTERN" = 0,  "EXTERN" = 1 };
   enum addr_typ {  "BYTE" = 1,  "WORD" = 2,  "LONG" = 4 };

   taggedunion ifp_measurement {
      "DIM"  struct {
               long;                                          /* address   */
               enum mem_typ;                          /* mem_typ   */
               enum addr_typ;                         /* addr_size */
            };
      "CAN"  struct {
               char[20];                                      /* message_name */
               enum { "STD" = 1,  "MODE" = 2,  "MODE_DEP" = 3}; /* signal_typ */
               char[20];                                      /* mode_signal */
               long;                                          /* identifier */
               int;                                           /* mes_size */
               char[20];                                      /* sender */
               int;                                           /* start_bit */
               int;                                           /* bit_size */
            };
   };

   /* ***************** interface-specific parameters for characteristics *************** */
   enum addr_mode { "DIRECT" = 0, "INDIRECT" = 1 };

   taggedunion ifp_characteristic {
      "DIM" struct {
               enum mem_typ;                                  /* address location */
               enum addr_mode;                                /* addressing mode */
            };
      "CAN" struct {
            };
   };
/end A2ML
```

_____ End of file SUPP1_IF.AML _____

_____ File SUPP2_IF.AML _____

```
/begin A2ML
   /* ***************** General interface-specific parameters ***************** */
   taggedunion interface_param {
      "EDIC" taggedstruct {
               :
               :
            };
   };

   /* ***************** interface-specific parameters for measurements ***************** */
   enum datasize {  "BYTE", "WORD", "LONG" };

   taggedunion ifp_measurement {
      "EDIC" taggedstruct {
               "SND" ( byte )*;
               "RCV" struct {
                        int;                                  /* start byte of answer */
                        enum datasize;                                /* data size to be analysed */
                     };
            };
   };

   /* x*************** interface-specific parameters for characteristics ****************** */
   taggedunion ifp_characteristic {
      "EDIC" struct {
               :
               :
            };
   };
/end A2ML
```

_____ end of file SUPP2_IF.AML _____

The ASAP2 description files of both suppliers could look as follows:

```
_____________________________________ File MST_ABS.A2L _____________________________________
/begin PROJECT          MST_ABS  "Project example see Figure 11"
   /begin HEADER        "General project description"
      VERSION           "0815"
      PROJECT_NO        1188
   /end HEADER

   /include engine_ecu.a2l
   /include abs_ecu.a2l
/end PROJECT
_____________________________________ end of file MST_ABS.A2L _____________________________

_____________________________________ File ENGINE_ECU.A2L _____________________________________
/begin MODULE  DIM  "Comment on module"          /* Detailed description of an application 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"          */ Applications engineer */
      PHONE_NO                      "(01111) 22222"
      ECU                           "Engine control"
      CPU_TYPE                  "Intel 0815"
      NO_OF_INTERFACES              2
      MEMORY_LAYOUT             PRG_RESERVED   0x0000   0x0400   -1   -1   -1   -1   -1
      MEMORY_LAYOUT                 PRG_CODE            0x0400   0x3000   -1   -1   -1   -1   -1
      MEMORY_LAYOUT             PRG_DATA            0x4000   0x1800   -1   -1   -1   -1   -1
      MEMORY_LAYOUT                 PRG_RESERVED   0x5800   0x0800   -1   -1   -1   -1   -1
      MEMORY_LAYOUT                 PRG_DATA            0x6000   0x2000   -1   -1   -1   -1   -1
      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                    BIG_ENDIAN
      DATA_SIZE                     16                      /* bit */
   /end MOD_COMMON

   /begin IF_DATA   DIM
      DIM_DISPLAY_TAB           14
      DIM_DISPLAY_PAR           0x5661  20   0xE001    2    "angular synchonous"
      DIM_DISPLAY_PAR           0x3441  20   0xE041    3    "time synchronous, rate 20ms"
   /end IF_DATA

   /begin IF_DATA   CAN
      CAN_BT                        0xFE
   /end IF_DATA

   /begin CHARACTERISTIC                KI "I share for speed limitation"
                                        VALUE                   /* type: constant */
                                        0/408F                  /* address */
                                        DAMOS_FW                /* deposit */
                                        5.0             /* max_diff */
                                        FACTOR01                /* conversion */
                                        0.0             /* lower limit */
                                        255.0                   /* upper limit */

                                                                /* interface-spec. parameters: address location, addressing */
            IF_DATA  "DIM"      EXTERN  DIRECT
            FUNCTION_LIST       V_LIM                           /* reference to functions */
   /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 */
                                                    /* interface-spec. parameters: address location, addressing */
  IF_DATA  "DIM"        EXTERNAL  INDIRECT
  /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
  FUNCTION_LIST             CLDSTRT  FLLD           /* reference to functions */
/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              0xOFF                       /* bit mask */
  IF_DATA  "DIM"                                    /* interface-specific part */
                            0x8038                  /* address */
                            EXTERNAL                /* memory type */
                            WORD                    /* address length */
  IF_DATA  "CAN"                                    /* interface-specific part */
                            "message-x"             /* message name */
                            STD                     /* signal type */
                            ""                      /* mode signal: here "don't care" */
                            0x0123                  /* identifier */
                            8                       /* message length */
                            "sender-Y"          /* sender */
                            5                       /* start bit */
                            16                      /* bit length */
  FUNCTION_LIST             CLDSTRT  FLLD           /* reference to functins */
/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 */
  IF_DATA  "DIM"                                        /* interface-specific part */
                            0x8020                      /* address */
                            EXTERNAL                    /* memory type */
                            WORD                        /* address length */
  IF_DATA  "CAN"                                        /* interface-specific part */
                            "message-X"                 /* message name */
                            STD                         /* signal type */
                            ""                          /* mode signal: here "don't care" */
                            0x0123                      /* identifier */
                            8                           /* message length */
                            "sender-Y"          /* sender */
                            21                          /* start bit */
                            16                          /* bit length */
  FUNCTION_LIST             V_LIM  CLDSTRT  FLLD        /* reference to functions */
/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           "AMOUNT"                  /* reference to table */
/end COMPU_METHOD

/begin COMPU_METHOD         N_RULE                      /* 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
/begin FUNCTION             CLDSTRT "cold start"        /end
/begin FUNCTION             FLLD  "full load"     /end
```

```
                                                    /* 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 */
    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 */
    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 */
                                                /* 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
/end MODULE
_____ end of file ENGINE_ECU.A2L _____
```

```
_____________________________________________ file ABS.ECU.A2L _____________________________________
/begin MODULE  EDIC  "Comment on module"                        /* detailed description of an application device  */


   /include "supp2_if_aml"                                      /* specification of the interface-specific parts */

   /begin MOD_PAR          "comments"
      VERSION              "test version 09.11.93"
      ECU                        "ABS control"
   /end MOD_PAR

   /begin MOD_COMMON             ”comments on these parameters"
      DEPOSIT                    ABSOLUTE
      BYTE_ORDER                 BIG_ENDIAN
      DATA_SIZE            16 /* bit */
   /end MOD_COMMON

   /begin IF_DATA  EDIC
      :
      :
   /end IF_DATA

   /begin CHARACTERISTIC
      :
   /end CHARACTERISTIC

   /begin MEASUREMENT           N
                           "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                 LITTLE_ENDIAN
      IF_DATA  "EDIC"                                    /* interface-specific part */
                           "SND"  0x10  0x00  0x05  0x08
                           "RCV"  4  UWORD
      FUNCTION_LIST              ID_ADJUSTM  FL_ADJUSTM  /* reference to functions */
   /end MEASUREMENT
   :
   :
   :
/end MODULE
_____________________________________ end of file ABS_ECU.A2L _____________________________________-
```

8 Appendix A: A2ML Grammar

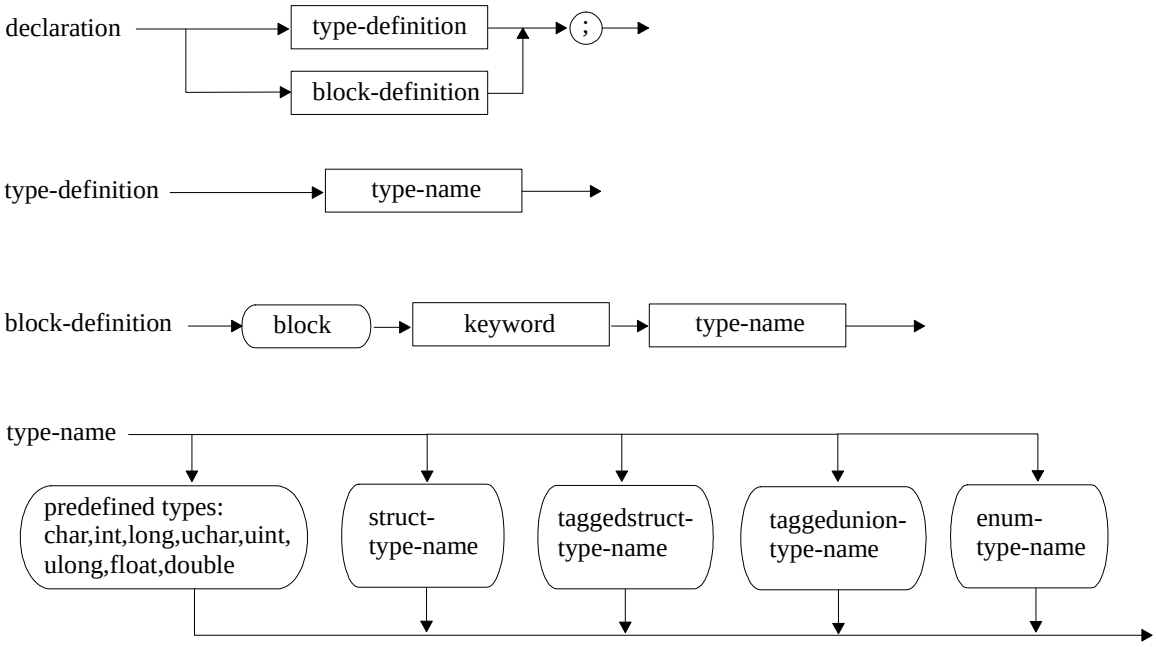


Figure 12 Diagram for description of the A2ML grammar

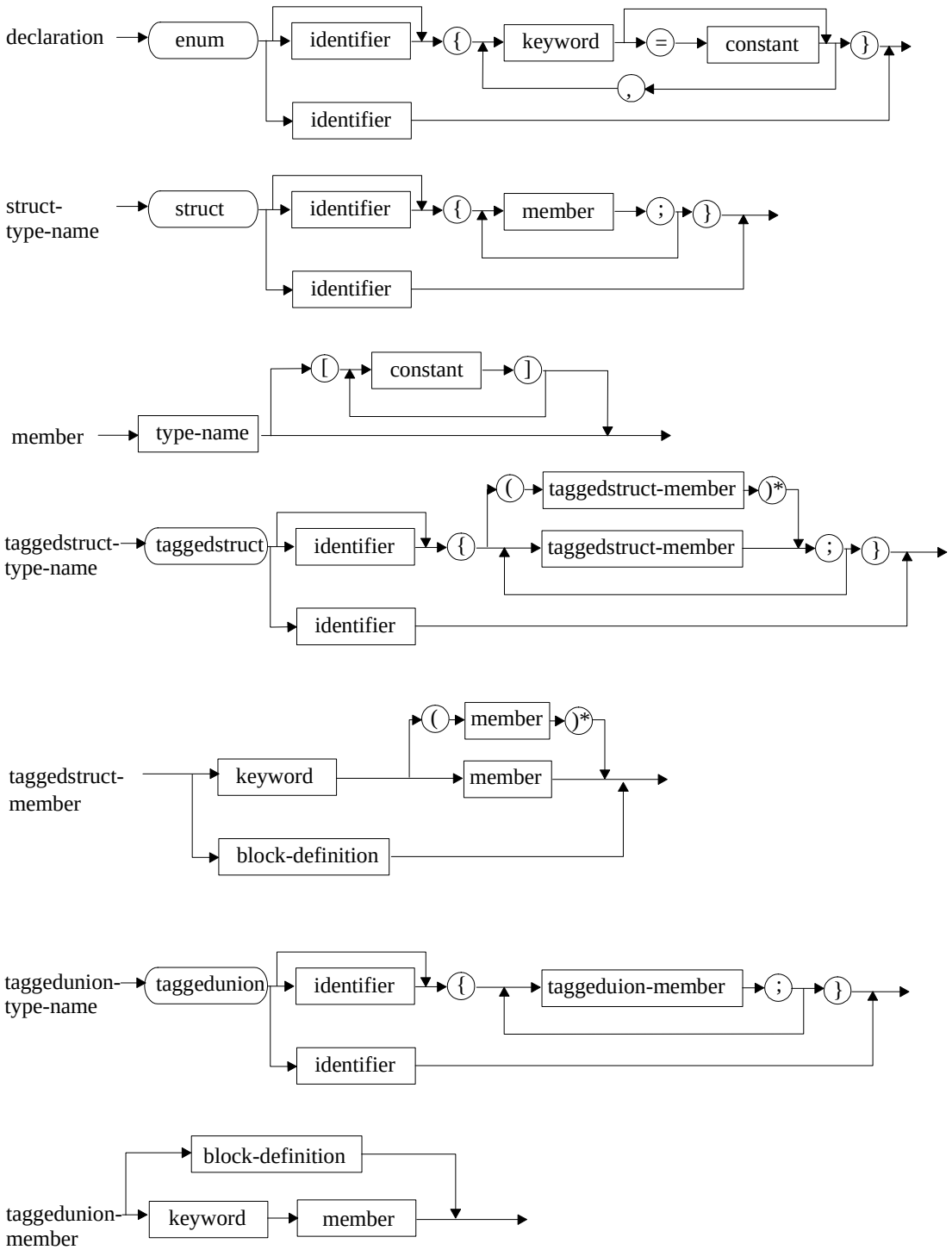
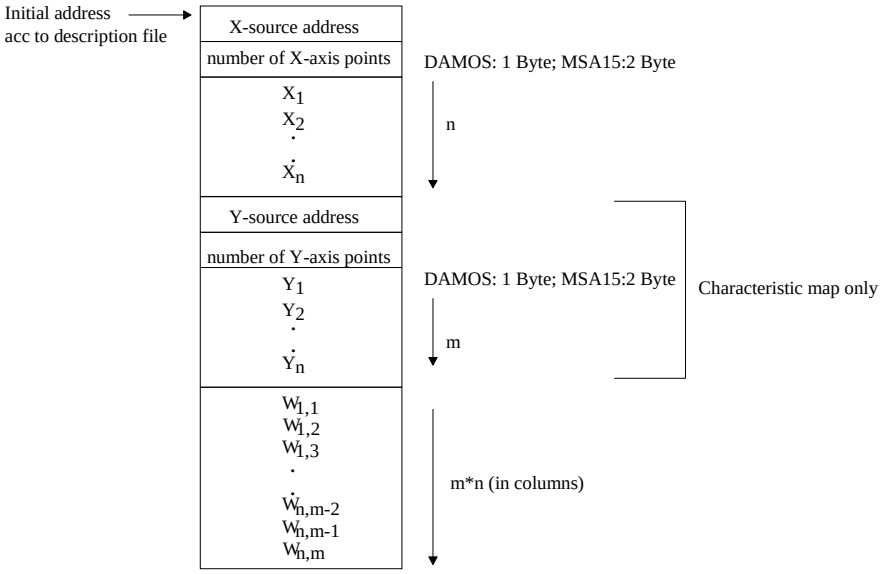


Figure 13 Diagram for description of the A2ML grammar (data types)

9 Appendix B: Record layouts

BOSCH: DAMOS, MSA15

CC, CD, AP distribution

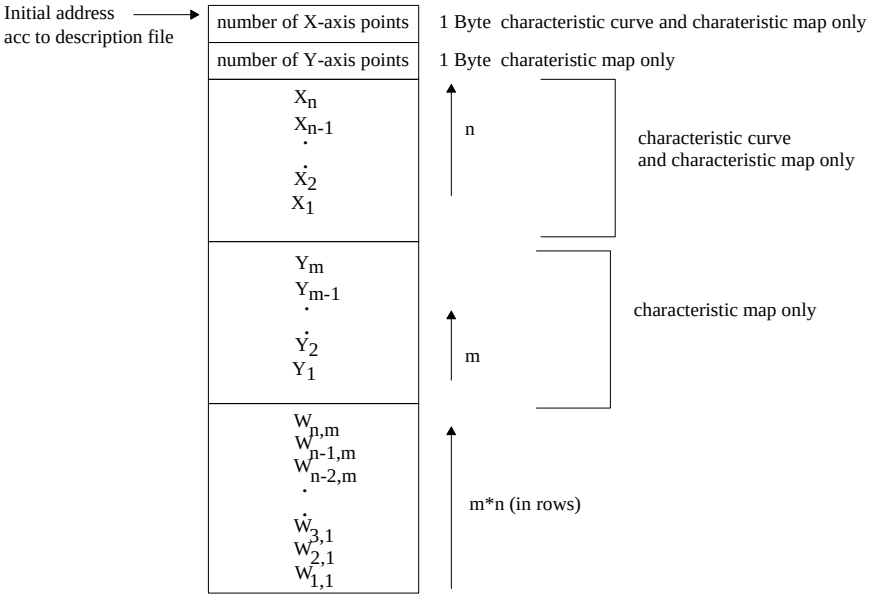


FV, FVB, FCC, FCD, GCC, GCD, ASCII:
Address points directly to the function value

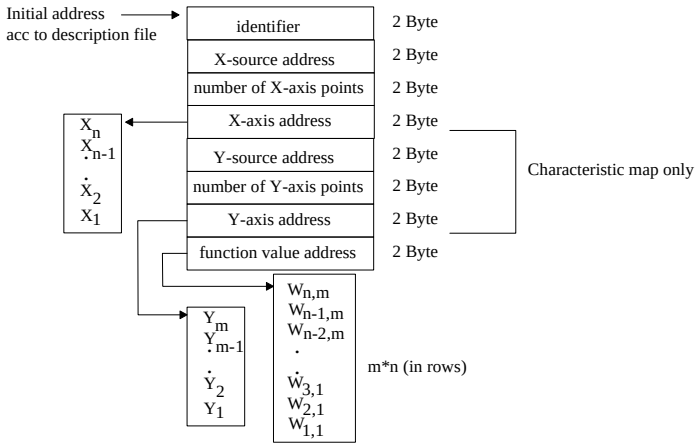
VFV, VFCC, VFCD:
Function values are addressed indirectly (via vector table)

BOSCH: KEBUSS

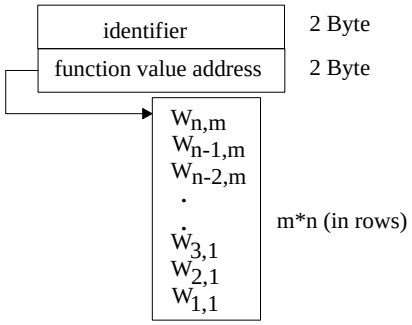
FV, CC, CD



CC, CD, AP distribution



FCC, FCD, GCC, GCD, ASCII



FV, FVB:

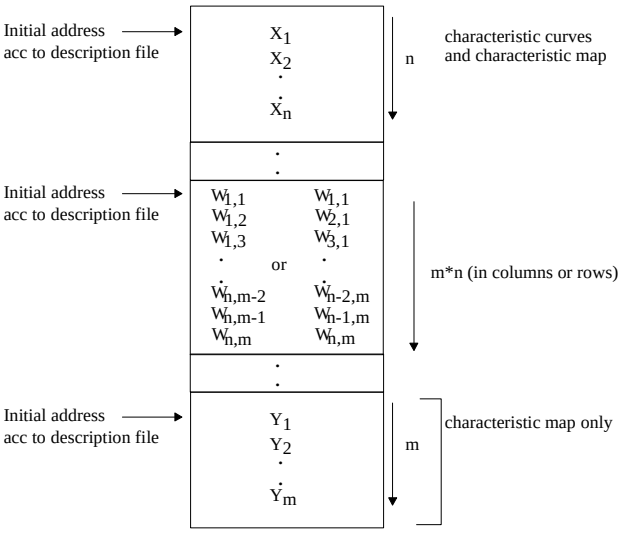
Address points directly to the value(s)

VFCC, VFCD:

Indirect addressing via vector table (in all other respects identical to FCC, FCD)

Siemens: Record layout

FV, CC, CD (other types???)



10 Glossary

ABUS
Automobile Bit-serial Universal Interface

Record layout
Description of the data structure with which an **adjustable object** of the control unit program is stored in memory.

ADC
Analog-Digital Converter

ADEX
Application Data Exchange System: Communication between a higher-order application system and the control units of a vehicle for the exchange of data.

AS
Application System

ASAP
Arbeitskreis zur Standardisierung von Applikationshilfsmitteln
(Working Group on the Standardisation of Application Tools)

ASAP 1b
Programmed standardised interface in the application system for access to the ASAP devices (see Figure 10).

ASAP 2
Standardised interface for the description data (description of control unit program: see Figure 10 and Figure 11).

ASAP device
The concept 'ASAP device' denotes a driver in the application system and the corresponding application device and control unit (if a control can be assigned).

ASAP2 metalanguage
Formal description language for the description of non-standardised, interface-specific ASAP2 description data.

ASCII
Adjustable object of the string type.

AuSy
Automation System

Description data
For the application 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.

Byte order
This concept denotes how the inidvidual bytes of a multibyte of the control unit program are to be interpreted (Intel format or Motorola format).

CAN
Control Area Network

C-DAMOS deposit
This concept denotes a specific data structure with which the adjustable objects (characteristics) are deposited in memory (BOSCH control units).

D-bus
Diagnostics bus

DAMOS deposit
This concept denotes a specific data structure with which the adjustable objects (characteristics) are deposited in memory (BOSCH control units).

Display table
Method for the output of control unit internal measurements (BOSCH):
1) The application 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.

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

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.

Function orientation
For the structuring of projects involving a very large number of adjustable objects and measurement objects, functions can be defined in ASAP2. These functions shall be used in the application 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.

Group characteristic curve, group characteristic map
In a number of BOSCH 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 behaviour of all allocated characteristic curve or characteristic map changes accordingly.

HW interface
Hardware interface, interface converter

KEBUSS deposit
This concept denotes a specific data structure with which the adjustable objects (characteristics) are deposited in memory (BOSCH control units).

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.

MAD
Measuring, Application and Diagnostics system

MSA15 deposit
This concept denotes a specific data structure with which the adjustable objects (characteristics) are deposited in memory (BOSCH control units).

MT
Module Type

ROM
Read-Only Memory

SG
Steuergerät (control unit)

SIEMENS deposit
This concept denotes a specific data structure with which the adjustable objects (characteristics) are deposited in memory (SIEMENS control units).

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

Abslolute axis points

address	address+1	address+2	address+3	address+4	address+5
value 1	value 2	value 3	value 4	value 5	value n

$APo_i = value_i$ i= {1...number of axispoints}

Difference axis points

address	address+1	address+2	address+3	address+4	address+5
initial value	delta 1	delta 2	delta 3	delta 4	delta n-1

APo_1 = initialvalue
$APo_{i+1} = APo_i + delta_i$ i= {2...number of axispoints}

Verbal conversion table
Conversion table for the visualisation 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 stringin the verbal conversion table, which describes the state of this quantity.

11 Keyword index