

# **Applications Systems Standardization Working Group**

## **Interface Specification**

### **Interface 2**

**(ASAM MCD 2MC / ASAP2)**

Version 1.51

Release

2003-03-11

---

# ASAP2: STANDARDIZED

## DESCRIPTION DATA

### Status of Document

Date:	<b>2003-03-11</b>
Author:	Brian Pavlik, Ford William Debs, GM Tom Zagotta, Siena Systems Don Nutter, ETAS Joe Pallatin, SII Dr. H. Schelling, Vector Informatik Dipl.-Ing. P.Lampert, Vector Informatik Dr. C. Dallmayr, Softing Dipl.-Inf. G.Luderböck, Softing Dipl.-Ing. M.Eisele, Vector Informatik Dipl.-Math. E.Schnorr, Vector Informatik
Version:	Version 1.51
Doc-ID:	
Status:	Release
Type	Specification

**Contents**

<b>1</b>	<b>Introduction .....</b>	<b>7</b>
<b>2</b>	<b>ASAP: Goals, Method, Interfaces .....</b>	<b>7</b>
<b>3</b>	<b>Division .....</b>	<b>13</b>
<b>4</b>	<b>System Description (SG Verbund) .....</b>	<b>14</b>
<b>5</b>	<b>Description Application Devices .....</b>	<b>15</b>
5.1	Control unit management data .....	15
5.2	General description data (control unit internal structures).....	16
5.3	Interface Parameters (general parameters).....	16
5.3.1	Interface module (memory emulator) .....	16
5.3.2	CAN bus .....	17
5.3.3	ABUS .....	17
5.3.4	Bus parameters for serial protocols (ISO) .....	17
5.3.5	Analog interface .....	18
5.4	Adjustment objects.....	18
5.4.1	Deposit structure [CHARACTERISTIC] .....	18
5.4.2	Bit pattern conversion (axis points and function values) .....	19
5.4.3	Function orientation (Reference) .....	19
5.5	Measurement channel .....	19
5.5.1	Deposit structure (measurement channel) .....	20
5.5.2	Bit pattern conversion.....	21
5.5.3	Function orientation (reference) .....	21
5.6	Conversion method.....	21
5.7	Conversion tables.....	21
5.8	Function description.....	22
5.9	Record layout .....	22
<b>6</b>	<b>Format of the description file.....</b>	<b>23</b>
6.1	Hierarchic division of the keywords .....	23
6.2	Predefined data types.....	28
6.3	Alphabetical list of keywords .....	30
6.3.1	A2ML .....	30
6.3.2	A2ML_VERSION .....	31
6.3.3	ADDR_EPK .....	32
6.3.4	ADDRESS_MAPPING .....	33
6.3.5	ALIGNMENT_BYTE.....	34
6.3.6	ALIGNMENT_FLOAT32_IEEE .....	35
6.3.7	ALIGNMENT_FLOAT64_IEEE .....	36
6.3.8	ALIGNMENT_LONG .....	37
6.3.9	ALIGNMENT_WORD .....	38
6.3.10	ANNOTATION.....	39
6.3.11	ANNOTATION_LABEL .....	41
6.3.12	ANNOTATION_ORIGIN .....	42
6.3.13	ANNOTATION_TEXT .....	43
6.3.14	ARRAY_SIZE.....	44
6.3.15	ASAP2_VERSION.....	45
6.3.16	AXIS_DESCR .....	46
6.3.17	AXIS_PTS .....	50

6.3.18	AXIS_PTS_REF .....	53
6.3.19	AXIS_PTS_X/_Y/_Z .....	55
6.3.20	AXIS_RESCALE_X/_Y/_Z .....	56
6.3.21	BIT_MASK .....	58
6.3.22	BIT_OPERATION .....	59
6.3.23	BYTE_ORDER .....	60
6.3.24	CALIBRATION_ACCESS .....	61
6.3.25	CALIBRATION_HANDLE .....	62
6.3.26	CALIBRATION_METHOD .....	63
6.3.27	CHARACTERISTIC .....	64
6.3.28	CHECKSUM .....	69
6.3.29	COEFFS .....	70
6.3.30	COMPARISON_QUANTITY .....	71
6.3.31	COMPU_METHOD .....	72
6.3.32	COMPU_TAB .....	76
6.3.33	COMPU_TAB_REF .....	77
6.3.34	COMPU_VTAB .....	78
6.3.35	COMPU_VTAB_RANGE .....	80
6.3.36	CPU_TYPE .....	82
6.3.37	CURVE_AXIS_REF .....	83
6.3.38	CUSTOMER .....	85
6.3.39	CUSTOMER_NO .....	86
6.3.40	DATA_SIZE .....	87
6.3.41	DEF_CHARACTERISTIC .....	88
6.3.42	DEFAULT_VALUE .....	89
6.3.43	DEPENDENT_CHARACTERISTIC .....	90
6.3.44	DEPOSIT .....	92
6.3.45	DISPLAY_IDENTIFIER .....	93
6.3.46	DIST_OP_X/_Y/_Z .....	94
6.3.47	ECU .....	95
6.3.48	ECU_ADDRESS .....	96
6.3.49	ECU_ADDRESS_EXTENSION .....	97
6.3.50	ECU_CALIBRATION_OFFSET .....	98
6.3.51	EPK .....	99
6.3.52	ERROR_MASK .....	100
6.3.53	EVENT_GROUP .....	101
6.3.54	EXTENDED_LIMITS .....	102
6.3.55	FIX_AXIS_PAR .....	103
6.3.56	FIX_AXIS_PAR_DIST .....	104
6.3.57	FIX_AXIS_PAR_LIST .....	105
6.3.58	FIX_NO_AXIS_PTS_X/_Y/_Z .....	106
6.3.59	FNC_VALUES .....	107
6.3.60	FORMAT .....	109
6.3.61	FORMULA .....	110
6.3.62	FORMULA_INV .....	112
6.3.63	FRAME .....	113
6.3.64	FRAME_MEASUREMENT .....	115
6.3.65	FUNCTION .....	116
6.3.66	FUNCTION_LIST .....	118
6.3.67	FUNCTION_VERSION .....	119
6.3.68	GROUP .....	120
6.3.69	GUARD_RAILS .....	125
6.3.70	HEADER .....	126
6.3.71	IDENTIFICATION .....	127
6.3.72	IF_DATA (AXIS_PTS, CHARACTERISTIC, MEMORY_LAYOUT) .....	128
6.3.73	IF_DATA (MEASUREMENT) .....	129
6.3.74	IF_DATA ASAP1B_ADDRESS (MEASUREMENT) .....	131
6.3.75	IF_DATA ASAP1B_CAN (MEASUREMENT) .....	132
6.3.76	IF_DATA (FRAME) .....	133

6.3.77	IF_DATA (MEMORY_SEGMENT) .....	134
6.3.78	IF_DATA (MODULE).....	135
6.3.79	IN_MEASUREMENT.....	137
6.3.80	LEFT_SHIFT .....	138
6.3.81	LOC_MEASUREMENT .....	139
6.3.82	MAP_LIST.....	140
6.3.83	MATRIX_DIM.....	141
6.3.84	MAX_GRAD.....	142
6.3.85	MAX_REFRESH .....	143
6.3.86	MEASUREMENT.....	145
6.3.87	MEMORY_LAYOUT.....	149
6.3.88	MEMORY_SEGMENT .....	151
6.3.89	MODULE.....	154
6.3.90	MOD_COMMON .....	156
6.3.91	MOD_PAR.....	158
6.3.92	MONOTONY .....	160
6.3.93	MULTIPLEX .....	161
6.3.94	NO_AXIS_PTS_X/_Y .....	162
6.3.95	NO_OF_INTERFACES .....	163
6.3.96	NO_RESCALE_X/_Y/_Z .....	164
6.3.97	NUMBER.....	165
6.3.98	OFFSET_X/_Y/_Z.....	166
6.3.99	OUT_MEASUREMENT.....	167
6.3.100	PHONE_NO .....	168
6.3.101	PROJECT .....	169
6.3.102	PROJECT_NO .....	170
6.3.103	RASTER.....	171
6.3.104	READ_ONLY .....	172
6.3.105	READ_WRITE.....	173
6.3.106	RECORD_LAYOUT.....	174
6.3.107	REF_CHARACTERISTIC.....	181
6.3.108	REF_GROUP .....	182
6.3.109	REF_MEASUREMENT.....	183
6.3.110	REF_MEMORY_SEGMENT .....	184
6.3.111	REF_UNIT .....	185
6.3.112	RESERVED .....	186
6.3.113	RIGHT_SHIFT .....	187
6.3.114	RIP_ADDR_X/_Y/_Z/_W .....	188
6.3.115	ROOT .....	191
6.3.116	SEED_KEY.....	192
6.3.117	SIGN_EXTEND.....	193
6.3.118	SHIFT_OP_X/_Y/_Z .....	194
6.3.119	SI_EXPONENTS .....	195
6.3.120	SOURCE .....	196
6.3.121	SRC_ADDR_X/_Y/_Z.....	197
6.3.122	SUB_FUNCTION.....	198
6.3.123	SUB_GROUP .....	199
6.3.124	SUPPLIER.....	200
6.3.125	SYSTEM_CONSTANT .....	201
6.3.126	S_REC_LAYOUT.....	202
6.3.127	UNIT .....	203
6.3.128	UNIT_CONVERSION .....	205
6.3.129	USER.....	206
6.3.130	USER_RIGHTS.....	207
6.3.131	VARIANT_CODING.....	209
6.3.132	VAR_ADDRESS.....	212
6.3.133	VAR_CHARACTERISTIC.....	213
6.3.134	VAR_CRITERION .....	214
6.3.135	VAR_FORBIDDEN_COMB .....	215

6.3.136	VAR_SELECTION_CHARACTERISTIC.....	216
6.3.137	VAR_MEASUREMENT.....	217
6.3.138	VAR_NAMING.....	218
6.3.139	VAR_SEPARATOR.....	219
6.3.140	VERSION.....	220
6.3.141	VIRTUAL.....	221
6.3.142	VIRTUAL_CHARACTERISTIC.....	222
<b>7</b>	<b><i>Include mechanism.....</i></b>	<b>224</b>
7.1	Description of complex projects.....	224
7.2	Description of interface-specific parameters.....	224
<b>8</b>	<b><i>Interface-specific description data.....</i></b>	<b>225</b>
8.1	Format of the ASAP2 metalanguage.....	226
8.2	Designing A2ML-file.....	228
8.3	Structure of BLOBs.....	232
8.4	Example of ASAP2 metalanguage.....	232
8.5	Example of description file.....	237
8.6	AML example for ASAP1a-CCP.....	245
8.7	Win32 API for the ASAP1a CCP Seed&Key Algorithm DLL.....	248
8.8	Win32 API for the ASAP1a Checksum Algorithm DLL.....	250
<b>9</b>	<b><i>Appendix A: Record layouts.....</i></b>	<b>253</b>
<b>10</b>	<b><i>Appendix B: IEEE-Floating-Point-Format.....</i></b>	<b>256</b>
10.1	32-Bit Format.....	256
10.2	64-Bit Format.....	256
<b>11</b>	<b><i>Appendix C: Using Reference Curves as Normalization Axes for Maps.....</i></b>	<b>257</b>
11.1	Background.....	257
11.2	Example.....	258
<b>12</b>	<b><i>Appendix D: modifications from version 1.0 to version 1.21.....</i></b>	<b>260</b>
<b>13</b>	<b><i>Appendix E: modifications from version 1.21 to version 1.31.....</i></b>	<b>262</b>
<b>14</b>	<b><i>Appendix F: modifications from version 1.31 to version 1.4.....</i></b>	<b>265</b>
<b>15</b>	<b><i>Appendix G: modifications from version 1.4 to version 1.5.....</i></b>	<b>266</b>
<b>16</b>	<b><i>Appendix H: Glossary.....</i></b>	<b>268</b>
<b>17</b>	<b><i>Appendix I: ASAP2 keywords.....</i></b>	<b>273</b>

## **1 Introduction**

This document is based on specification "Application Systems Standardization Working Group: Interface Specifications - Interface 2, Version 1.0 of 31 March 1994". The new requirements result from experience applying this standard. The modifications are decided at the meeting of ASAP2 Working Group at 24 January 1996. Appendix D contains a list of modifications relating to version 1.0. At the meeting of ASAP-plenum on 11 March 1997 the acceptance of the present first draft of this specification had been decided.

At ASAP2-Plenum on 24 June 1997 a final revision was decided to correct some details (version 1.21: no functional extensions).

The version 1.3 contains the decisions made during the ASAP2 workgroup meeting at 8<sup>th</sup> and 9<sup>th</sup> September 1998 at Vector Informatik. In the current version 1.31 the results of the inconsistency meeting at Vector Informatik GmbH on 02/24/1999 have been added. All modifications between version V1.31 and V1.21 are listed in Appendix E.

The version 1.4 contains the decisions made during the ASAM-MCD-MC2 meeting at Robert Bosch GMBH K5 on 18<sup>th</sup> January 2000. All modifications between version V1.31 and V1.4 are listed in Appendix F. **Note: They are no longer marked on the outside border.**

Version 1.5 contains decisions made during the ASAM-MCD-MC2 P05-01 workgroup in session over the course of the year 2001 and finalized in November, 2002. All modifications are listed in Appendix G and are marked on the outside border.

## **2 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.

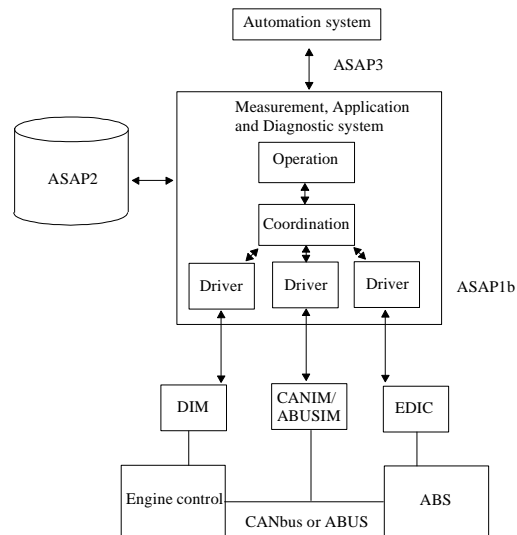


Figure 1: Overall system and interfaces

The individual application systems (AS) of the measurement, application and diagnostic system (MCD) 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 or 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):

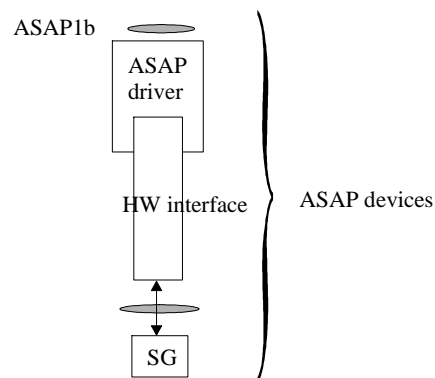


Figure 2: ASAP device (principle structure)



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

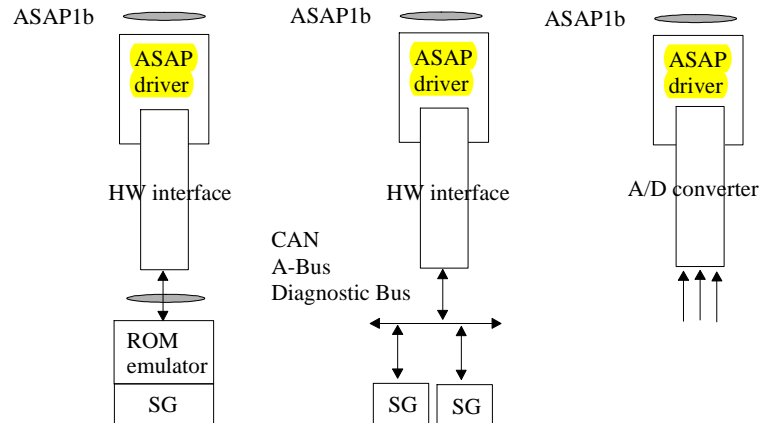


Figure 3: Various versions of the ASAP device

The ASAP 1b interface is a functional interface which was initially defined independently of the MCD 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_ACCESS	Initialisation of the adjustment system
3	SYNC	Synchronisation of the individual subsystems
4	READ	Data transfer upstream of ASAP 1b
5	ACCESS	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
9	COMMAND	Handling of manufacturer defined commands

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 connec-

tions 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:

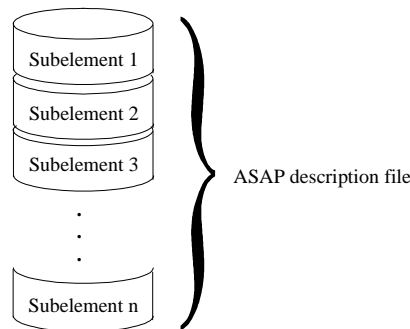


Figure 4: ASAP description file

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 CAN bus 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.

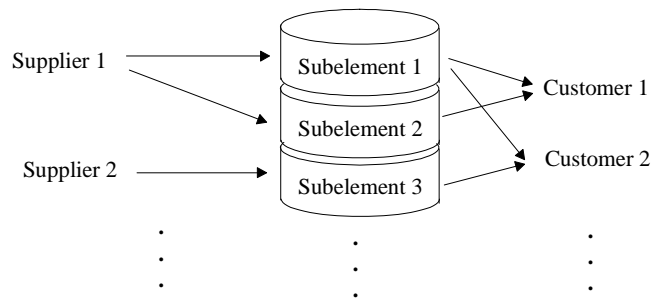


Figure 5: Customers and suppliers of the ASAP 2 subelements

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 (MCD) 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 MCD
- ? 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 MCD 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:

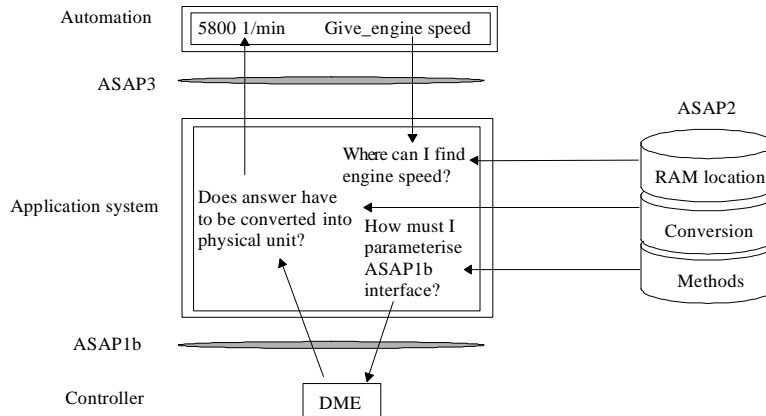


Figure 6: Interworking of Automation system, Application system, DME with ASAP interfaces

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 MCD. The connection is established via the ASAP 3 interface.

## PART A: DIVISION OF THE DESCRIPTION DATA

### 3 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 MCD 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.

## 4 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

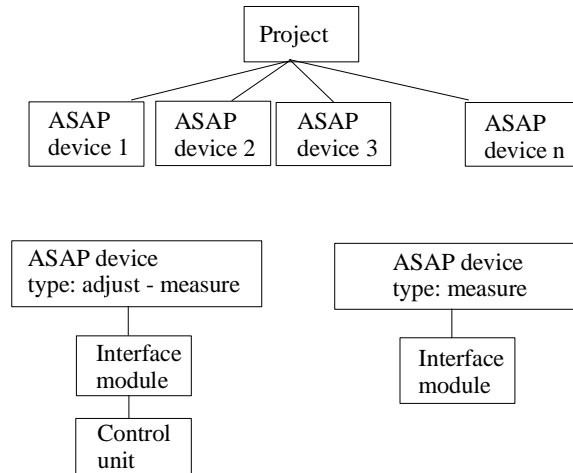


Figure 7: Model application system

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 or CAN (bus monitor).

The project header (keyword HEADER, page 119) is typically created by the project manager and includes the following entries:

- ? version of the parameter file format [keyword VERSION, page 220]
- ? project
- ? project identifier
- ? project number [keyword PROJECT\_NO, page 170]
- ? remarks, comments

## 5 Description Application Devices

For each ASAP device a complete description (keyword MODULE, page 151) 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 158]
- ? control unit internal structures (e.g. standard record layout) [MOD\_COMMON, page 156]
- ? communication interfaces [IF\_DATA, see Part C]
- ? adjustable and measurement objects [CHARACTERISTIC, page 61 and MEASUREMENT, page 145]
- ? conversion rules [COMPU\_METHOD, page 71]

### 5.1 Control unit management data

The control unit management data are created with the MOD\_PAR (page 158) keyword. Using optional parameters the following can be specified:

? data status	[VERSION, page 220]
? comments, remarks	
? EPROM identifier address	[ADDR_EPK, page 31]
? EPROM identifier	[EPK, page 96]
? Manufacturer or supplier	[SUPPLIER, page 199]
? Firm or customer	[CUSTOMER, page 83]
? Customer number	[CUSTOMER_NO, page 86]
? User	[USER, page 203]
? Telephone numbers (applications engineer responsible)	[PHONE_NO, page 168]
? Control unit	[ECU, page 95]
? Processor	[CPU_TYPE, page 80]
? Number of interfaces	[NO_OF_INTERFACES, page 163]
? Memory layout	[MEMORY_LAYOUT, page 149]
? System constants	[SYSTEM_CONSTANT, page 201]
? Project-basis-address (see MEMORY_LAYOUT: memory layout)	

## 5.2 General description data (control unit internal structures)

These description data (keyword MOD\_COMMON, page 156) makes 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 <u>one</u> standard layout?)	[S_REC_LAYOUT, page 202]
? Standard deposit mode of the axis points (difference, absolute values)	[DEPOSIT, page 89]
? Byte order	[BYTE_ORDER, page 59]
? Data size in bits	[DATA_SIZE, page 87]

## 5.3 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.

### 5.3.1 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



### 5.3.2 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 parameters 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 for 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

### 5.3.3 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

### 5.3.4 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

### 5.3.5 Analog interface

Does not require any interface parameters!

## 5.4 Adjustment objects

Each adjustment object must have its own description to be created with the CHARACTERISTIC keyword (see page 61). Between the /begin CHARACTERISTIC and /end CHARACTERISTIC brackets further keywords can be nested (as may also be the case for other keywords).

Example:

```

/begin CHARACTERISTIC PUMPKF      /* Description of a characteristic map */
...
...
  /begin AXIS_DESCR                /* axis description for 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 adjustment 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.

### 5.4.1 Deposit structure [CHARACTERISTIC]

#### 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 46]
  - ? 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.

### **5.4.2 Bit pattern conversion (axis points and function values)**

- ? Reference to the list of conversion methods
- ? Byte order [BYTE\_ORDER, page 59], signed-unsigned information
- ? Plausibilities (limit values)
- ? Physical representation: see Conversion method

### **5.4.3 Function orientation (Reference)**

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

Remark:

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

## **5.5 Measurement channel**

For each measurement (e.g. AD value, CAN signal, source data, RAM cell) a description is created with the keyword MEASUREMENT (page 145). 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

### **5.5.1 Deposit structure (measurement channel)**

#### **5.5.1.1 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)

#### **5.5.1.2 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

#### **5.5.1.3 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

#### **5.5.1.4 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)

#### **5.5.1.5 Analog interface deposit structure**

Here the following can be described (see Part C):

- ? Analog input (No. 1-8)
- ? Resolution in bits
- ? Gain, prescaler

### 5.5.2 Bit pattern conversion

? Reference to conversion method list

? byte order [BYTE\_ORDER, page 59], signed-unsigned information

*Physical representation*: see conversion methods

### 5.5.3 Function orientation (reference)

Under the keyword FUNCTION\_LIST (page 118) a list of the functions allocated to this object is given. Referencing occurs via names (see comment in chapter 6.3.66 page 118).

## 5.6 Conversion method

The keyword COMPU\_METHOD (repeatedly possible, see page 71) 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\_TAB\_REF, page 77]

? Physical representation (significant positions, decimal places, physical unit)

? Coefficients for fractional rational function [COEFFS, page 69]

#### Remark:

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).

## 5.7 Conversion tables

Under the keyword COMPU\_TAB (page 76) 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 78) can be used (allocation table: bit pattern <--> string)).

## 5.8 Function description

All functions referenced under CHARACTERISTIC and MEASUREMENT can be described as follows:

- ? Name of the function
- ? Comment, description of the *function*

## 5.9 Record layout

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

## PART B: FORMAT OF THE DESCRIPTION FILE

### 6 Format of the description file

#### 6.1 Hierarchic division of the keywords

Keyword	(*)=multiple	Meaning
ASAP2_VERSION		ASAP2 version identification
A2ML_VERSION		Version number of ASAP2 Meta Language
PROJECT		Project description
HEADER		Project header description
PROJECT_NO		Project number
VERSION		Project version number
MODULE	(*)	Description of the ASAP devices
A2ML	(*)	ASAP2-Meta-Language (interface-specific description data)
AXIS_PTS	(*)	Axis points distribution
ANNOTATION	(*)	Set of notes
ANNOTATION_LABEL		Title of annotation
ANNOTATION_ORIGIN		Creator of annotation
ANNOTATION_TEXT		Text of annotation
BYTE_ORDER		Byte order of axis points
CALIBRATION_ACCESS		Access for calibration
DEPOSIT		Absolute or difference axis points
ABSOLUTE		
DIFFERENCE		
DISPLAY_IDENTIFIER		Optional display name
EXTENDED_LIMITS		extended range of values
ECU_ADDRESS_EXTENSION		Address extension of the ECU address
FORMAT		Display format of axis points
FUNCTION_LIST		Function orientation
GUARD_RAILS		Indicates the use of guardrails
IF_DATA	(*)	Interface-specific description data
READ_ONLY		'Read Only' attribut
REF_MEMORY_SEGMENT		reference to memory segment
CHARACTERISTIC	(*)	Adjustable objects
VALUE		
CURVE		
MAP		
CUBOID		
VAL_BLK		
ASCII		
ANNOTATION	(*)	Description
ANNOTATION_LABEL		Title of annotation
ANNOTATION_ORIGIN		Creator of annotation
ANNOTATION_TEXT		Text of annotation
AXIS_DESCR	(*)	Axis description
STD_AXIS		
FIX_AXIS		
COM_AXIS		
RES_AXIS		
CURVE_AXIS		
ANNOTATION	(*)	Set of notes
ANNOTATION_LABEL		Title of annotation
ANNOTATION_ORIGIN		Creator of annotation
ANNOTATION_TEXT		Text of annotation
AXIS_PTS_REF		Reference to axis point distribution
CURVE_AXIS_REF		Used to normalize or scale an axis
BIT_MASK		Bit mask to decode single-bit values
BYTE_ORDER		Byte order of axis points
DEPOSIT		Absolute or difference axis points
FIX_AXIS_PAR		Fixed axis parameters
FIX_AXIS_PAR_DIST		Fixed axis parameters (variant)

## ASAM MCD 2MC / ASAP2 Interface Specification

Keyword	(*)=multiple	Meaning
FIX_AXIS_PAR_LIST		Fixed axis values
FORMAT		Display format of axis points
MAX_GRAD		Maximum gradient with respect to this axis
MONOTONY		Monotony with respect to this axis
MON_INCREASE		
MON_DECREASE		
STRICT_INCREASE		
STRICT_DECREASE		
READ_ONLY		'Read Only' attribut
BYTE_ORDER		Byte order
BIT_MASK		Bit mask
CALIBRATION_ACCESS		Access for calibration
COMPARISON_QUANTITY		Comparison quantity
DEPENDENT_CHARACTERISTIC		References to characteristics
DISPLAY_IDENTIFIER		Optional display name
ECU_ADDRESS_EXTENSION		Address extension of the ECU address
EXTENDED_LIMITS		Extended limits, e.g. hard limits
FORMAT		Display format of values
FUNCTION_LIST		Function orientation
GUARD_RAILS		Indicates the use of guardrails
IF_DATA	(*)	Interface-specific description data
MAP_LIST		For cuboids: compricing maps
MATRIX_DIM		Dimensions of multidimensioned arrays
MAX_REFRESH		Maximum refresh rate
NUMBER		Number of ASCII characters or fixed values
READ_ONLY		'Read Only' attribut
REF_MEMORY_SEGMENT		reference to memory segment
VIRTUAL_CHARACTERISTIC		mark for being virtual
COMPU_METHOD	(*)	Conversion method
TAB_INTP		
TAB_NOINTP		
TAB_VERB		
RAT_FUNC		
FORM		
COEFS		Coefficients for fractional rational function
COMPU_TAB_REF		Reference to conversion table
FORMULA		Conversion formula
FORMULA_INV		Invers conversion formula
REF_UNIT		Reference to a measurement unit
COMPU_TAB	(*)	Conversion table
TAB_INTP		
TAB_NOINTP		
DEFAULT_VALUE		default output string
COMPU_VTAB	(*)	Verbal conversion table
TAB_VERB		
DEFAULT_VALUE		Default output string
COMPU_VTAB_RANGE	(*)	Description of range based verbal conversion tables
DEFAULT_VALUE		Default output string
FRAME		Frame
FRAME_MEASUREMENT		Frame measurement objects
IF_DATA	(*)	Interface-specific description data
FUNCTION	(*)	Function description
ANNOTATION	(*)	Set of notes
ANNOTATION_LABEL		Title of annotation
ANNOTATION_ORIGIN		Creator of annotation
ANNOTATION_TEXT		Text of annotation
DEF_CHARACTERISTIC		Defined adjustable objects



Keyword	(*)=multiple	Meaning
FUNCTION_VERSION		Version of the function
IN_MEASUREMENT		Input quantity
LOC_MEASUREMENT		Local quantity
OUT_MEASUREMENT		Output quantity
REF_CHARACTERISTIC		Referenced adjustable objects
SUB_FUNCTION		Subfunction of respective function
GROUP	(*)	Declaration of groups
ANNOTATION	(*)	Set of notes
ANNOTATION_LABEL		Title of annotation
ANNOTATION_ORIGIN		Creator of annotation
ANNOTATION_TEXT		Text of annotation
ROOT		Flag for root node
REF_CHARACTERISTIC		Reference to characteristic objects
REF_MEASUREMENT		Reference to measurement objects
FUNCTION_LIST		Function list
SUB_GROUP		Sub group
IF_DATA	(*)	Interface-specific description data
MEASUREMENT	(*)	Measurement object
ANNOTATION	(*)	Set of notes
ANNOTATION_LABEL		Title of annotation
ANNOTATION_ORIGIN		Creator of annotation
ANNOTATION_TEXT		Text of annotation
ARRAY_SIZE		Array size of measurement objects
BIT_MASK		Bit mask to decode single-bit values
BIT_OPERATION		Bit operation
LEFT_SHIFT		Number of bit positions to shift left
RIGHT_SHIFT		Number of bit positions to shift right
SIGN_EXTEND		sign extension for measurement data
BYTE_ORDER		Byte order of measurement object
DISPLAY_IDENTIFIER		Optional display name
ECU_ADDRESS		Address
ECU_ADDRESS_EXTENSION		Address extension of the ECU address
ERROR_MASK		Mask error bits
FORMAT		Display format of measurement object
FUNCTION_LIST		Function orientation
IF_DATA	(*)	Interface-specific description data
MATRIX_DIM		Dimensions of multidimensioned arrays
MAX_REFRESH		Refresh rate in the control unit
READ_WRITE		'Writeable'
REF_MEMORY_SEGMENT		reference to memory segment
VIRTUAL		Virtual measurement
MOD_COMMON		Module-wide (ECU specific) valid definitions
ALIGNMENT_BYTE		Alignment border for byte values
ALIGNMENT_FLOAT32_IEEE		Alignment border for float32 values
ALIGNMENT_FLOAT64_IEEE		Alignment border for float64 values
ALIGNMENT_LONG		Alignment border for long values
ALIGNMENT_WORD		Alignment border for word values
BYTE_ORDER		Byte order
DATA_SIZE		Data size in bits
DEPOSIT		Standard deposit mode for axis
S_REC_LAYOUT		Reference to the standard record layout
MOD_PAR		Control unit management data
ADDR_EPK		Address of EPROM identifier
CPU_TYPE		CPU
CUSTOMER		Firm or customer
CUSTOMER_NO		Customer number
CALIBRATION_METHOD	(*)	Access method
CALIBRATION_HANDLE		Handle for calibration method
ECU		Control unit
ECU_CALIBRATION_OFFSET		Address offset

## ASAM MCD 2MC / ASAP2 Interface Specification

Keyword	(*)=multiple	Meaning
EPK		EPROM identifier
MEMORY_LAYOUT	(*)	Memory layout
PRG_CODE		
PRG_DATA		
PRG_RESERVED		
IF_DATA	(*)	Interface-specific description data
MEMORY_SEGMENT	(*)	Memory segment
CODE		
DATA		
OFFLINE_DATA		
VARIABLES		
SERAM		
RESERVED		
CALIBRATION_VARIABLES		
EXCLUDE_FROM_FLASH		
RAM		
EEPROM		
EPROM		
ROM		
REGISTER		
FLASH		
INTERN		
EXTERN		
IF_DATA	(*)	Interface-specific description data
NO_OF_INTERFACES		Number of interfaces
PHONE_NO		Phone number of application engineer responsible
SUPPLIER		Manufacturer or supplier
SYSTEM_CONSTANT	(*)	System-defined constants
USER		User
VERSION		Module-specific version identifier
RECORD_LAYOUT	(*)	Description of the record layout
ALIGNMENT_BYTE		Alignment border for byte values
ALIGNMENT_FLOAT32_IEEE		Alignment border for float32 values
ALIGNMENT_FLOAT64_IEEE		Alignment border for float64 values
ALIGNMENT_LONG		Alignment border for long values
ALIGNMENT_WORD		Alignment border for word values
AXIS_PTS_X		X axis points
AXIS_PTS_Y		Y axis points
AXIS_PTS_Z		Z axis points
AXIS_RESCALE_X		rescaling X axis points
AXIS_RESCALE_Y		rescaling Y axis points
AXIS_RESCALE_Z		rescaling Z axis points
DIST_OP_X		X axis: parameter 'distance' for fixed characteristics
DIST_OP_Y		Y axis: parameter 'distance' for fixed characteristics
DIST_OP_Z		Z axis: parameter 'distance' for fixed characteristics
FIX_NO_AXIS_PTS_X		Fixed number of X axis points
FIX_NO_AXIS_PTS_Y		Fixed number of Y axis points
FIX_NO_AXIS_PTS_Z		Fixed number of Z axis points
FNC_VALUES		Table values
IDENTIFICATION		Identification
NO_RESCALE_X		number of rescale pairs for X axis
NO_RESCALE_Y		number of rescale pairs for Y axis
NO_RESCALE_Z		number of rescale pairs for Z axis
NO_AXIS_PTS_X		Number of X axis points
NO_AXIS_PTS_Y		Number of Y axis points
NO_AXIS_PTS_Z		Number of Z axis points
OFFSET_X		X axis: parameter 'offset' for fixed characteristics
OFFSET_Y		Y axis: parameter 'offset' for fixed characteristics
OFFSET_Z		Z axis: parameter 'offset' for fixed characteristics

Keyword	(*)=multiple	Meaning
RESERVED	(*)	Parameter is skipped (not interpreted)
RIP_ADDR_W		Table value: Address 'result of interpolation'
RIP_ADDR_X		X axis: Address 'result of interpolation'
RIP_ADDR_Y		Y axis: Address 'result of interpolation'
RIP_ADDR_Z		Z axis: Address 'result of interpolation'
SHIFT_OP_X		X axis: parameter 'shift' for fixed characteristics
SHIFT_OP_Y		Y axis: parameter 'shift' for fixed characteristics
SHIFT_OP_Z		Z axis: parameter 'shift' for fixed characteristics
SRC_ADDR_X		X axis: Address of input quantity
SRC_ADDR_Y		Y axis: Address of input quantity
SRC_ADDR_Z		Z axis: Address of input quantity
UNIT	(*)	Measurement unit
EXTENDED_SI		
DERIVED		
REF_UNIT		Reference to another unit
SI_EXPONENTS		Exponents of base dimensions
UNIT_CONVERSION		Specifies relationship between two units
USER_RIGHTS	(*)	Groups with constitute access rights
REF_GROUP	(*)	List of referenced groups
READ_ONLY		Read only
VARIANT_CODING		Variant coding
VAR_CHARACTERISTIC	(*)	Definition of variant coded adjustable objects
VAR_ADDRESS		Adjustable objects address list (start address of variants)
VAR_CRITERION	(*)	Definition of variant criterion
VAR_MEASUREMENT		Measurement object which indicates criterion value
VAR_SELECTION_CHARACTERISTIC		Characteristic object which modifies criterion value
VAR_FORBIDDEN_COMB		Forbidden combinations of different variants
VAR_NAMING		Naming of variant coded adjustable objects
NUMERIC		
VAR_SEPARATOR		Separator of ajustable objects names

## 6.2 Predefined data types

**ident**      `typedef char [MAX_IDENT + 1] ident;`  
String with *MAX\_IDENT* (*at present* = 255) alphanumeric characters including points and brackets, interpreted as hierarchical concatenation of partial strings separated by points. Every partial string may not exceed *MAX\_PARTIAL\_IDENT* (*at present* = 32) characters, including the length of an optional array index (numeric or as a symbolic string) in brackets at the end of the partial string. One string without a point in between is also possible, in this case *MAX\_IDENT* = *MAX\_PARTIAL\_IDENT*. The number of partial strings within *ident* is not limited. The character chain must correspond with the identifier laws defined in programming language C. Identifiers can represent instances of array elements or instances of elements of complex C types or nested combinations of these. An instance of the element of a struct type would be represented by the concatenation of the instance name, a point and the element name. An instance of an array element would be represented by an instance name followed by a pair of brackets which contain either a numeric value or a symbolic string which is defined as an enumerator of an ENUM definition of the C program. Identifiers are random names which may contain characters A through Z, a through z, underscore (`_`), numerals 0 through 9, points (`'.'`) and brackets (`'['`, `']'`). However, the following limitations apply: the first character must be a letter or an underscore, brackets must occur in pairs at the end of a partial string and must contain a number or an alphabetic string (description of the index of an array element).

**Note:**

Identifiers consisting of partial identifiers separated by points (concatenation of instance name and element name) may be presented by the MCD system in a hierarchical manner (show instance name first, then allow access to an element of the instance). This allows existing MCD systems to restrict the display length of the identifier to *MAX\_PARTIAL\_IDENT*.

**Note:**

Identifiers generally must not match to the following defined ASAP2 keywords. The ASAP2 keywords are completely listed in Appendix G.

**Note:**

A lower case "x" and an upper case "X" can be used as unique identifiers. In other words, variables are case sensitive so that x and X are different identifiers.

**string**      `typedef char [MAX_STRING + 1] string;`  
ANSI C compliant 'C type' string with maximum *MAX\_STRING* (*at present* = 255) characters. Begin and end of the string are indicated by a double inverted comma. Only the following sequence of escape characters is allowed : `"\r\n"`, which describes a carriage return. MCD systems may ignore the carriage return sequence and/or apply wrapping or scrolling of strings when displayed. If the string contains one or more double inverted commas, a backslash (escape char. indicator) concatenated with a double inverted comma must be inserted ( example: `"hello \"world\" how are you ?"` ). Alternatively, two double inverted commas can be inserted in this case ( example: `"hello ""world"" how are you ?"` ) for compatibility with ASAP2 V1.2 and prior.

float	8-byte floating point number (IEEE format)
	<p><b><u>Note:</u></b> The character for a decimal point is fixed as a dot “.”. A comma “,” is not allowed for use as a decimal point.</p>
int	2-byte signed integer
	<p><b><u>Note:</u></b> The notation of hexadecimal values is fixed, e.g. 0xE0, 0xFF, etc.</p>
long	4-byte signed long
datatype	<pre>typedef enum datatype    {  UBYTE,   SBYTE,   UWORD,   SWORD,                            ULONG,      SLONG,      FLOAT32_IEEE,                            FLOAT64_IEEE                         }</pre> <p>Enumeration for description of the basic data types in the ECU program (format of FLOAT32_IEEE: see Appendix B: IEEE-Floating-Point-Format).</p>
datasize	<pre>typedef enum datasize {BYTE, WORD, LONG}</pre> <p>Enumeration for description of the word lengths in the ECU program</p>
addrtype	<p>Enumeration for description of the addressing of table values or axis point values:</p> <p>PBYTE:      The relevant memory location has a 1 byte pointer to this table value or axis point value.</p> <p>PWORD:      The relevant memory location has a 2 byte pointer to this table value or axis point value.</p> <p>PLONG:      The relevant memory location has a 4 byte pointer to this table value or axis point value.</p> <p>DIRECT:     The relevant memory location has the first table value or axis point value, all others follow with incrementing address.</p>
byteorder	<pre>typedef enum byteorder    {  LITTLE_ENDIAN, BIG_ENDIAN,                            MSB_LAST, MSB_FIRST}</pre> <p>Enumeration for description of the byte order in the control unit program.</p> <p><b><u>Note:</u></b> Use of LITTLE_ENDIAN and BIG_ENDIAN defined with keyword BYTE_ORDER leads to mistakes because it is in contradiction to general use of terms „little endian“ and „big endian“. The keywords LITTLE_ENDIAN and BIG_ENDIAN should no longer be used, they should be replaced by MSB_LAST and MSB_FIRST which are equivalent (definition of MSB_LAST and MSB_FIRST: see keyword BYTE_ORDER).</p>
indexorder	<p>Enumeration for description of the axis point sequence in the memory.</p> <p>INDEX_INCR: Increasing index with increasing address</p> <p>INDEX_DECR: decreasing index with increasing address</p>

## 6.3 Alphabetical list of keywords

Some individual elements of the database are delimited by ‘/begin’ and ‘/end’ keywords, similarly to the opening '{' and closing '}' brackets in 'C'. The delimiters are applied to those elements that contain an optional part, to prevent ambiguous expressions. The delimiters following defined with the ASAP2 keywords are mandatory, i.e. the delimiters have to be used if defined and mustn't be used if not defined.

To reduce the size of description files the short delimiters '{' and '}' can be used:

```
<keyword> { <description_body> }
```

instead of:

```
/begin <keyword> <description_body> /end <keyword>
```

The short delimiters should preferably be used. The delimiters should not be mixed, i.e. don't use short delimiters and '/begin' or '/end' respectively in the same ASAP2 description file.

**Note: Since version 1.31 the usage of `{` and `}` is no more recommended.** None of the currently existing systems uses brackets, so the useage of brackets will probably cause problems.

### 6.3.1 A2ML

#### Prototype:

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

#### Parameters:

FormatSpecification	A2ML code for description of interface specific description data.
---------------------	---

#### Description:

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

#### Example:

See page 237: file SUPP1\_IF.AML and file SUPP2\_IF.AML

### 6.3.2 A2ML\_VERSION

#### Prototype:

A2ML\_VERSION                      Version Upgrade

#### Parameters:

int VersionNo:	Version number of AML part
int UpgradNo:	Upgrade number of AML part

#### Description:

The keyword A2ML\_VERSION is new in ASAP2 version 1.31. The reason for this keyword is, to declare what kind of BLOBs should be generated from the A2ML parts. Since ASAP2 version 1.31 we have a specification for the storage layout of the BLOBs. The keyword is optional. When the keyword is omitted, or the version number is below 1.31 then the ASAP2 parser may produce the old BLOB format. When the A2ML version number is 1.31, then the new format must be generated.

The A2ML version can be expressed by two numerals:

- VersionNo
- UpgradNo

where 'VersionNo' represents the main version number and 'UpgradNo' the upgrade number (fractional part of version number). The upgrade number will be incremented if additional functionality is added to ASAP2 Meta Language standard which has no effect on existing applications (compatible modifications). The version number will be incremented in case if incompatible modifications.

#### Example:

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

### 6.3.3 ADDR\_EPK

**Prototype:**

ADDR\_EPK                      Address

**Parameters:**

    long Address:              Address of the EPROM identifier

**Description:**

    Address of the EPROM identifier

**Example:**

ADDR\_EPK                      0x145678



### 6.3.4 ADDRESS\_MAPPING

#### Prototype:

ADDRESS\_MAPPING                      Orig\_Address Mapping\_Address Length

#### Parameters:

long Orig\_Address:              Base address of the memory segment to be mapped.  
long Mapping\_Address: Address to which the base address is to be mapped  
long Length                      Length of the segment to be mapped

#### Description:

Remapping of the address space of the ECU to an access address. This is needed for special application methods (for example, special address calculation is needed in KWP2000, where only 24 bit addresses are available.) The address mapping may only be used inside an IF\_DATA section of a MEMORY\_SEGMENT.

#### Example:

```
/begin IF_DATA ASAP1B_ETK
    /* ADDRESS_MAPPING   orig_addr   mapping_addr   length */
    ADDRESS_MAPPING   0x4000   0x8000   0x0200
/end IF_DATA

/begin IF_DATA ASAP1B_KWP2000
    /* ADDRESS_MAPPING   orig_addr   mapping_addr   length */
    ADDRESS_MAPPING   0x4000   0x6000   0x0200
/end IF_DATA
```

### 6.3.5 ALIGNMENT\_BYTE

#### Prototype:

ALIGNMENT\_BYTE            AlignmentBorder

#### Parameters:

int AlignmentBorder:    describes the border at which the value is aligned to, i.e. its memory address must be dividable by the value Alignment-Border.

#### Description:

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

#### Example:

ALIGNMENT\_BYTE            4 /\* bytes have a 4-byte alignment \*/

### 6.3.6 ALIGNMENT\_FLOAT32\_IEEE

#### Prototype:

ALIGNMENT\_FLOAT32\_IEEE AlignmentBorder

#### Parameters:

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

#### Description:

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

Used in MOD\_COMMON and RECORD\_LAYOUT.

#### Example:

ALIGNMENT\_FLOAT32\_IEEE 4 /\* 32bit floats have a 4-byte alignment \*/

### 6.3.7 ALIGNMENT\_FLOAT64\_IEEE

#### Prototype:

ALIGNMENT\_FLOAT64\_IEEE AlignmentBorder

#### Parameters:

int AlignmentBorder: describes the border at which the value is aligned to, i.e. its memory address must be dividable by the value Alignment-Border.

#### Description:

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

Used in MOD\_COMMON and RECORD\_LAYOUT.

#### Example:

ALIGNMENT\_FLOAT64\_IEEE 4 /\* 64bit floats have a 4-byte alignment \*/

### 6.3.8 ALIGNMENT\_LONG

**Prototype:**

ALIGNMENT\_LONG            AlignmentBorder

**Parameters:**

int AlignmentBorder:    describes the border at which the value is aligned to, i.e. its memory address must be dividable by the value Alignment-Border.

**Description:**

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

Used in MOD\_COMMON and RECORD\_LAYOUT.

**Example:**

ALIGNMENT\_LONG    8 /\* longs have a 8-byte alignment \*/

### 6.3.9 ALIGNMENT\_WORD

**Prototype:**

ALIGNMENT\_WORD            AlignmentBorder

**Parameters:**

int AlignmentBorder:    describes the border at which the value is aligned to, i.e. its memory address must be dividable by the value Alignment-Border.

**Description:**

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

Used in MOD\_COMMON and RECORD\_LAYOUT.

**Example:**

ALIGNMENT\_WORD            4 /\* words have a 4-byte alignment \*/

### 6.3.10 ANNOTATION

**Prototype:**

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

**Parameters:**

none

**Optional Parameters:**

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

**Description:**

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

**Note:** An ANNOTATION may occur several times within a definition (due to compatibility with MSR/MEDOC SW-DTD, the future ASAP2 V2.0).

**Example:**

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

/begin ANNOTATION
  ANNOTATION_LABEL "Taupunktendezeiten"
  /begin ANNOTATION_TEXT
    "Flughöhe  Taupunktendezeit\r\n"
```

" 13000ft 20 sec\r\n"  
" 25000ft 40 sec\r\n"  
" 35000ft 12 sec"

/end ANNOTATION\_TEXT

/end ANNOTATION

....

/end CHARACTERISTIC



### 6.3.11 ANNOTATION\_LABEL

#### Prototype:

ANNOTATION\_LABEL label

#### Parameters:

string	label	label or title of the annotation
--------	-------	----------------------------------

#### Description:

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

#### Example:

ANNOTATION\_LABEL "Calibration Note"

### 6.3.12 ANNOTATION\_ORIGIN

**Prototype:**

ANNOTATION\_ORIGIN origin

**Parameters:**

string origin                      creator or creating system of the annotation

**Description:**

To identify who or which system has created an annotation.

**Example:**

ANNOTATION\_ORIGIN "from the calibration planning department"

### 6.3.13 ANNOTATION\_TEXT

**Prototype:**

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

**Parameters:**

string	annotation_text
--------	-----------------

**Description:**

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

**Example:**

```
/begin CHARACTERISTIC annotation.example2 ...  
....  
  /begin ANNOTATION  
    ANNOTATION_LABEL "Calibration Note"  
    /begin ANNOTATION_TEXT  
      "Blariblara plumpaquatsch. Oder sonst ein beliebiger "  
      "Text.\r\n"  
      "Lediglich, wenn ein Doppelhochkomma "  
      "auftritt, muß man das als \" oder \"\" markieren."  
    /end ANNOTATION_TEXT  
  /end ANNOTATION  
....  
/end CHARACTERISTIC
```

### 6.3.14 ARRAY\_SIZE

#### Prototype:

ARRAY\_SIZE                      Number

#### Parameters:

int Number:                      Number of measurement values included in respective measurement object (maximum value of 'Number': 32767).

#### Description:

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

**Note:** The use of this keyword should be replaced by MATRIX\_DIM.

#### Example:

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

### 6.3.15 ASAP2\_VERSION

#### Prototype:

ASAP2\_VERSION                      Version Upgrad

#### Parameters:

int VersionNo:	Version number of ASAP2 standard
int UpgradNo:	Upgrade number of ASAP2 standard

#### Description:

The ASAP2 version can be expressed by two numerals:

- VersionNo
- UpgradNo

where 'VersionNo' represents the main version number and 'UpgradNo' the upgrade number (fractional part of version number). The upgrade number will be incremented if additional functionality is implemented to ASAP2 standard which has no effect on existing applications (compatible modifications). The version number will be incremented in case if incompatible modifications.

#### Example:

```
ASAP2_VERSION      1 31                      /* Version 1.31 */
```

### 6.3.16 AXIS\_DESCR

#### Prototype:

```
/begin AXIS_DESCR      Attribute InputQuantity Conversion MaxAxisPoints
                        LowerLimit UpperLimit
                        [-> READ_ONLY]
                        [-> FORMAT]
                        {-> ANNOTATION} *
                        [-> AXIS_PTS_REF]
                        [-> MAX_GRAD]
                        [-> MONOTONY]
                        [-> BYTE_ORDER]
                        [-> EXTENDED_LIMITS]
                        [-> FIX_AXIS_PAR]
                        [-> FIX_AXIS_PAR_DIST]
                        [-> FIX_AXIS_PAR_LIST]
                        [-> DEPOSIT]
                        [-> CURVE_AXIS_REF]
|
/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 virtual axis points that are not deposited at EPROM. The axis points can be calculated from parameters defined with keywords FIX_AXIS_PAR, FIX_AXIS_PAR_DIST and FIX_AXIS_PAR_LIST. The axis points can't be modified.
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'.
RES_AXIS:	Rescale axis. For this variant of the axis points the axis point values are separated from the table values of the curve or map in the emulation memory and must be described by a special AXIS_PTS data record. The reference to this record occurs with the keyword 'AXIS_PTS_REF'.
CURVE_AXIS:	Curve axis. This axis type uses a separate CURVE CHARACTERISTIC to rescale the

axis. The referenced CURVE is used to lookup an axis index, and the index value is used by the controller to determine the operating point in the CURVE or MAP. See Appendix C for more details.

ident InputQuantity: Reference to the data record for description of the input quantity (see MEASUREMENT). If there is no input quantity assigned, parameter 'InputQuantity' should be set to "NO\_INPUT\_QUANTITY" (application systems must be capable to treat this case).

ident Conversion: Reference to the relevant record of the description of the conversion method (see COMPU\_METHOD). If there is no conversion method, as in the case of CURVE\_AXIS, the parameter 'Conversion' should be set to "NO\_COMPU\_METHOD" (application systems must be able to handle this case).

int MaxAxisPoints : 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 LowerLimit: Plausible range of axis point values, lower limit

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

**Note:** The limit values are to be interpreted as physical values and not internal values.

### Optional parameters:

-> READ\_ONLY: This keyword can be used to indicate that the axis points of adjustable object cannot be changed (but can be read only).

**Note:** This optional keyword used at CHARACTERISTIC record indicates the adjustable object to be read only at all (table values and axis points).

-> FORMAT: With deviation from the display format specified with keyword COMPU\_TAB referenced by parameter <Conversion> a special display format can be specified to be used to display the axis points.

-> 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 ( $\text{MaxGrad} = \max ( \text{abs}((W_{i,k} - W_{i-1,k}) / (X_i - X_{i-1})) )$ ).

- > 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'.
- > FIX\_AXIS\_PAR\_DIST: Similar to FIX\_AXIS\_PAR but with a different computing method.
- > FIX\_AXIS\_PAR\_LIST: The original values of the axis are directly contained in the file. The assigned COMPU\_METHOD is applied to achieve the actual display values from the values with this keyword.
- > DEPOSIT: The axis points of a characteristic can be deposited in two different ways:  
a) The individual axis point values are deposited as absolute values.  
b) The individual axis point are stored as differences. Each axis point value is determined from the adjacent axis point (predecessor).  
Where the standard value does not apply this parameter can be used to specify the axis point deposit.
- > ANNOTATION: Set of notes (represented as multi-line ASCII description texts) which are related. Can serve e.g. as application note. When a COM\_AXIS is referenced it is sufficient to place the ANNOTATION with its AXIS\_PTS in order to avoid redundant information.
- > EXTENDED\_LIMITS: 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 (lower limit...upper limit) a warning could be generated (extended limits enabled only for "power user").
- > CURVE\_AXIS\_REF: When the axis type is CURVE\_AXIS, this keyword must be used to specify the CURVE CHARACTERISTIC that is used to normalize or scale this axis.

### **Description:**

Axis description within an adjustable object

#### **Note:**

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 process-



ing 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 145). The MEASUREMENT keyword also specifies the 'conversion', 'lower limit' and 'upper limit' parameters.

These parameters are not 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.

**Note:** The keywords FIX\_AXIS\_PAR, FIX\_AXIS\_PAR\_DIST, DEPOSIT and FIX\_AXIS\_PAR\_LIST are mutually exclusive, i.e. please use at most one of these keywords at the same AXIS\_DESCR record.

### Example:

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

### 6.3.17 AXIS\_PTS

#### Prototype:

```

/begin AXIS_PTS      Name LongIdentifier Address InputQuantity Deposit Max-
                      Diff Conversion MaxAxisPoints LowerLimit UpperLimit
                      [-> DISPLAY_IDENTIFIER]
                      [-> READ_ONLY]
                      [-> FORMAT]
                      [-> DEPOSIT]
                      [-> BYTE_ORDER]
                      [-> FUNCTION_LIST]
                      [-> REF_MEMORY_SEGMENT]
                      [-> GUARD_RAILS]
                      [-> EXTENDED_LIMITS]
                      {-> ANNOTATION} *
                      {-> IF_DATA } *
                      [-> CALIBRATION_ACCESS]
                      [-> ECU_ADDRESS_EXTENSION]

/end AXIS_PTS

```

#### Parameters:

identName: unique identifier in the ECU program

#### Note:

The name of the axis points object has to be unique within all measurement objects and adjustable objects of the ASAP2 MODULE, i.e. there must not be another AXIS\_PTS, CHARACTERISTIC or MEASUREMENT object with the the same identifier in the MODULE.

string LongIdentifier:	comment, description
long Address:	address of the adjustable object in the emulation memory
ident InputQuantity:	reference to the data record for description of the input quantity (see MEASUREMENT). If there is no input quantity assigned, parameter 'InputQuantity' should be set to "NO_INPUT_QUANTITY" (application systems must be capable to treat this case).
ident Deposit:	reference to the relevant data record for description of the record layout (see RECORD_LAYOUT)
float MaxDiff:	maximum float with respect to the adjustment of a table value
ident Conversion:	Reference to the relevant record of the description of the conversion method (see COMPU_METHOD). If there is no conversion method, as in the case of CURVE_AXIS, the parameter 'Conversion' should be set to "NO_COMPU_METHOD" (application systems must be able to handle this case).

int MaxAxisPoints:	maximum number of axis points
float LowerLimit:	plausible range of axis point values, lower limit
float UpperLimit:	plausible range of axis point values, upper limit

**Note:** The limit values are to be interpreted as physical values and not internal values.

**Note:** The parameters of AXIS\_PTS are dominate. The values defined at AXIS\_DESCR have to be ignored in case of COM\_AXIS.

### Optional parameters:

- > DISPLAY\_IDENTIFIER: Can be used as a display name (alternative to the 'name' attribute).
- > READ\_ONLY: This keyword can be used to indicate that the axis points of axis points distribution cannot be changed (but can be read only).  
**Note:** This optional keyword used at CHARACTERISTIC record indicates the adjustable object to be read only at all (table values and axis pts).
- > FORMAT: With deviation from the display format specified with keyword COMPU\_TAB referenced by parameter <Conversion> a special display format can be specified to be used to display the axis points.
- > 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).  
**Note:** Since ASAP2 version 1.20 the keyword FUNCTION comprises some additional features to describe functional structure and dependencies. The keyword FUNCTION\_LIST is going to be canceled at ASAP2 version 2.00.
- > REF\_MEMORY\_SEGMENT: Reference to the memory segment which is needed if the address is not unique (this occurs in the case of lapping address ranges (overlapping memory segments)).

- > **GUARD\_RAILS:** This keyword is used to indicate that an AXIS\_PTS uses guard rails. The Measurement and Calibration System does not allow the user to edit the outermost axis breakpoints (see GUARD\_RAILS).
- > **IF\_DATA:** Data record for description of the interface specific description data (BLOB: binary large object). The parameters associated with this keyword are described in the ASAP2 meta-language (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.
- > **ANNOTATION:** Set of notes (represented as multi-line ASCII description texts) which are related. Can serve e.g. as application note.
- > **EXTENDED\_LIMITS:** 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 (lower limit...upper limit) a warning could be generated (extended limits enabled only for "power user").
- > **CALIBRATION\_ACCESS:** This keyword specifies the access of the axis points for calibration. It replaces the READ\_ONLY attribute.
- > **ECU\_ADDRESS\_EXTENSION:** This keyword is an additional address information. For instance it can be used, to distinguish different address spaces of an ECU (multi-microcontroller devices).

### **Description:**

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

### **Example:**

```

/begin AXIS_PTS
    STV_N          /* name */
    "axis points distribution speed" /* long identifier */
    /*
    0x9876          /* address */
    N               /* input quantity */
    DAMOS_SST      /* deposit */
    100.0           /* maxdiff */
    R_SPEED        /* conversion */
    21              /* maximum number of axis points */
    0.0 5800.0     /* lower and upper limit */
    GUARD_RAILS    /* uses guard rails*/
    REF_MEMORY_SEGMENT Data3
    /begin FUNCTION_LIST ID_ADJUSTM FL_ADJUSTM SPEED_LIM
    /end FUNCTION_LIST
    /begin IF_DATA      DIM EXTERNAL DIRECT
    /end IF_DATA
    CALIBRATION_ACCESS CALIBRATION
/end AXIS_PTS

```

**6.3.18 AXIS\_PTS\_REF****Prototype:**

AXIS\_PTS\_REF                      AxisPoints

**Parameters:**

ident AxisPoints:                      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 key-word 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 */
/begin IF_DATA            DIM EXTERNAL INDIRECT
/end IF_DATA
/begin AXIS_DESCR        /* description of X-axis points */
                          COM_AXIS    N     /* common axis points, input quantity
                                         */
                          CONV_N 14       /* conversion, max. no. of axis p.*/
                          0.0 5800.0      /* lower limit, upper limit */
                          AXIS_PTS_REF    GRP_N
/end AXIS_DESCR
/end CHARACTERISTIC
```

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

### 6.3.19 AXIS\_PTS\_X/\_Y/\_Z

**Prototype:**

AXIS\_PTS\_X/\_Y/\_Z                      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 Datatype:	Data type of the axis point values
indexorder IndexIncr:	Decreasing or increasing index with increasing addresses
addrtype Addressing:	Addressing of the table values (see enum addrtype).

**Description:**

Description of the X axis points or Y axis points in the memory (see keyword RECORD\_LAYOUT).

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

**Example:**

AXIS\_PTS\_X                              3 ULONG INDEX\_INCR DIRECT

### 6.3.20 AXIS\_RESCALE\_X/\_Y/\_Z

#### Prototype:

AXIS\_RESCALE\_X/\_Y/\_Z    Position   Datatype   MaxNumberOfRescalePairs   IndexIncr  
Addressing

#### Parameters:

int Position:                      position of the rescale axis point value pairs in the deposit structure (description of sequence of elements in the data record).

datatype DataType:              Data type of the rescale axis point values

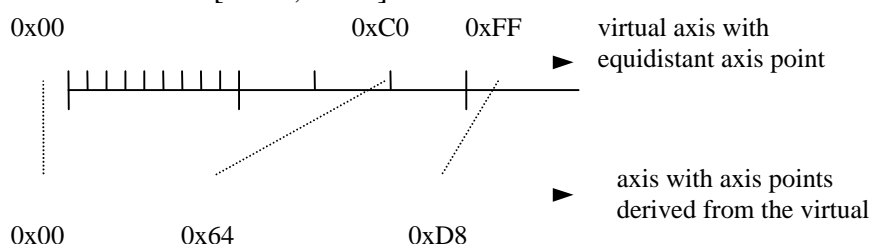
int MaxNumberOfRescalePairs:    maximum number of rescaling axis point pairs  
(see NO\_RESCALE\_PTS\_X/\_Y/\_Z)

indexorder IndexIncr:          Decreasing or increasing index with increasing addresses

addrtype Addressing:            Addressing of the table values (see enum addrtype).

#### Description:

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



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

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

FOR i = 1 TO no\_rescale\_x - 1



```

FOR k = 1 TO virtual1 - virtuali-1
/* repeat for the number of points in the interval on the virtual axis */

k = k + 1

Xk = axisi + ((k - 1) * D * virtuali - virtuali-1) * (axisi-1 - axisi) /
(virtuali-1 - virtuali)

X1 = axis1
Xno_axis_pts = axisno_rescale_x

```

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

The following example makes clear how the evaluation of the formula can be used to derive the actual axis points. We have `no_of_rescale_pairs = 3` and `virtual1 = 0x00 = 0`, `virtual2 = 0xC0 = 192`, `virtual3 = 0xFF = 255`, `axis1 = 0x00 = 0`, `axis2 = 0x64 = 100`, `axis3 = 0xD8 = 216`. Assume `no_axis_pts = 9`, and therefore `D = 32`. The first of the two executions of the inner loop (j-loop) is on `virtual2 - virtual1 / D = 192/32 = 6` iterations. For each iteration  $(axis_2 - axis_1) / (virtual_2 - virtual_1) = 100/192$ , and therefore

$$\begin{aligned}
 X_2 &= 0 + 32 * 100/192 = 16,666, \\
 X_3 &= 0 + 64 * 100/192 = 33,333, \\
 X_4 &= 0 + 96 * 100/192 = 50, \\
 X_5 &= 0 + 128 * 100/192 = 66,666, \\
 X_6 &= 0 + 160 * 100/192 = 83,333.
 \end{aligned}$$

For the second execution there are `virtual3 - virtual2 / D = 2` iterations with  $(axis_3 - axis_2) / (virtual_3 - virtual_2) = 116/64$ . Consequently

$$\begin{aligned}
 X_7 &= 100 + (192 - 192) * 116/64 = 100 \text{ and} \\
 X_8 &= 100 + (224 - 192) * 116/64 = 158.
 \end{aligned}$$

Also `X1 = axis1 = 0` and `X9 = axis3 = 216`.

Used in `RECORD_LAYOUT`.

### Example:

```

AXIS_RESCALE_X          3 UBYTE 5 INDEX_INCR DIRECT

```

### 6.3.21 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. The least significant bit in BIT\_MASK determines how far the masked value is shifted to the right.

**Example:**

BIT\_MASK                      0x00000FFF

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

**Note:**

The newly added comments about the least significant bit and the inserted samples are valid only while no keyword BIT\_OPERATION is used. If the keyword BIT\_OPERATION is used then its defined parameters dominate those parameters of the BIT\_MASK keyword.

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

**Example:**

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

### 6.3.22 BIT\_OPERATION

#### Prototype:

```

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

```

#### Parameters:

#### Optional parameters

- > LEFT\_SHIFT: Number of positions to left shift data, zeros will be shifted in from the right.
- > RIGHT\_SHIFT: Number of positions to right shift data, zeros will be shifted in from the left.
- > SIGN\_EXTEND: Gives a sign extension of sign bit for measurement data.

#### Description:

The BIT\_OPERATION keyword can be used to perform operation on the masked out value.

First BIT\_MASK will be applied on measurment data, then LEFT\_SHIFT/(RIGHT\_SHIFT) is performed and last the SIGN\_EXTEND is carried out.

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

#### Example:

```

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

```

Explanation	Data	Comment
Data after mask operation	00000000000100000	
Data after shift operation	00000000000000010	shifted right 4 positions
Data after sign extend	11111111111111110	

### 6.3.23 BYTE\_ORDER

#### Prototype:

BYTE\_ORDER                      ByteOrder

#### Parameters:

byteorder ByteOrder:      Byte order of the relevant quantity in the ECU program  
**Note:** Use of LITTLE\_ENDIAN and BIG\_ENDIAN defined with keyword BYTE\_ORDER in version 1.0 leads to mistakes because it is in contradiction to general use of terms „little endian“ and „big endian“. Since version 1.2 the keywords LITTLE\_ENDIAN and BIG\_ENDIAN are permissible but should not longer be used. They should be replaced by MSB\_LAST and MSB\_FIRST which are equivalent: MSB\_LAST corresponds to the Intel format (equivalent former keyword is BIG\_ENDIAN). MSB\_FIRST corresponds to the Motorola format (equivalent former keyword is LITTLE\_ENDIAN).

#### Description:

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

#### Example:

BYTE\_ORDER                      MSB\_LAST

Byte Order	Keyword	Former Keyword	Increasing address -->				
			n	n+1	....	n + (N-1)	n + N
Motorola Format	MSB_FIRST	LITTLE_ENDIAN	Byte <sub>N</sub> (Most Significant Byte)	Byte <sub>N-1</sub>	....	Byte <sub>1</sub>	Byte <sub>0</sub> (Least Significant Byte)
Intel Format	MSB_LAST	BIG_ENDIAN	Byte <sub>0</sub> (Least Significant Byte)	Byte <sub>1</sub>	....	Byte <sub>N-1</sub>	Byte <sub>N</sub> (Most Significant Byte)

Table 3: Byte order - memory data deposition

### 6.3.24 CALIBRATION\_ACCESS

#### Prototype

CALIBRATION\_ACCESS    Type

#### Parameters

enum Type:

Possible Types:

**CALIBRATION:** Characteristic or axis points with calibration allowed.

**NO\_CALIBRATION:** This keyword can be used to indicate that the axis points cannot be changed (but can be read only).

**Note:** This optional keyword used at CHARACTERISTIC record indicates the adjustable object to be read only at all (table values and axis pts).

**NOT\_IN\_MCD\_SYSTEM:** Internal characteristic or axis points which are not readable or writable by the MCD-System. If there are references between AXIS\_PTS and CHARACTERISTICS with one of them gets this value of CALIBRATION\_ACCESS, the other one must get this value too.

**OFFLINE\_CALIBRATION**

Variables which can be flashed but not emulated ore calibrated, e.g. values representing savety relevant property while driving.

#### Optional Parameters

#### Description

This keyword specifies the access of a CHARACTERISTIC or AXIS\_PTS for calibration. It substitutes the READ\_ONLY attribute since ASAM-MCD-2MC V1.4

#### Example

CALIBRATION\_ACCESS

CALIBRATION

### 6.3.25 CALIBRATION\_HANDLE

#### Prototype:

```
/begin CALIBRATION_HANDLE          (Handle)*  
/end CALIBRATION_HANDLE
```

#### Parameters:

long Handle	Handle for the calibration method
-------------	-----------------------------------

#### Description:

Definition of the calibration method specific. The interpretation of this data depends on the calibration method used. Used in CALIBRATION\_METHOD

#### Example:

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

### 6.3.26 CALIBRATION\_METHOD

#### Prototype:

```

/ begin CALIBRATION_METHOD                Method Version
                                           {-> CALIBRATION_HANDLE}*
/ end CALIBRATION_METHOD

```

#### Parameters:

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

#### Optional Parameters:

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

#### Description:

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

#### Example:

```

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

```

### 6.3.27 CHARACTERISTIC

#### Prototype:

```

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

```

#### Parameters:

ident Name: unique identifier in the ECU program

#### **Note:**

The name of the adjustable object has to be unique within all measurement objects and adjustable objects of the ASAP2 MODULE, i.e. there must not be another AXIS\_PTS, CHARACTERISTIC or MEASUREMENT object with the the same identifier in the MODULE.

string LongIdentifier: 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



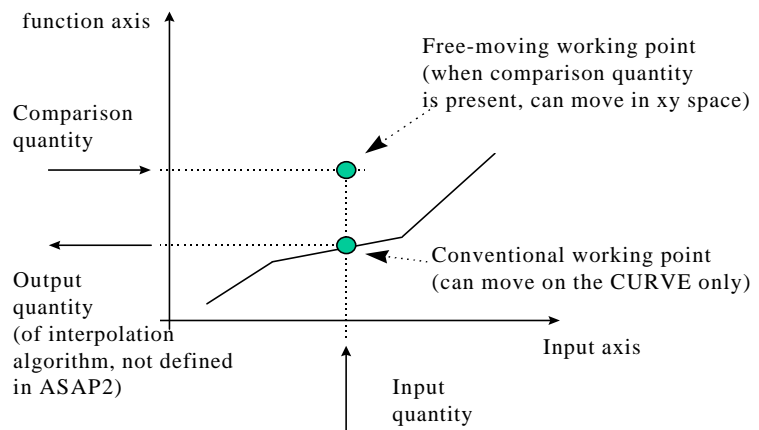
ident Deposit:	reference to the corresponding data record for description of the record layout (see RECORD_LAYOUT)
float Maxdiff:	maximum float with respect to an adjustment of a table value <b>Note:</b> This value is interpreted as an absolute value, not as a percentage.
ident Conversion:	Reference to the relevant record of the description of the conversion method (see COMPU_METHOD). If there is no conversion method, as in the case of CURVE_AXIS, the parameter 'Conversion' should be set to "NO_COMPU_METHOD" (application systems must be able to handle this case).
float LowerLimit:	plausible range of table values, lower limit
float UpperLimit:	plausible range of table values, upper limit
<b>Note:</b> The limit values are to be interpreted as physical values and not internal values.	

### Optional parameters

- > DISPLAY\_IDENTIFIER: Can be used as a display name (alternative to the 'name' attribute).
- > FORMAT: With deviation from the display format specified with keyword COMPU\_TAB referenced by parameter <Conversion> a special display format can be specified to be used to display the table values.
- > 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).  
**Note:** Since ASAP2 version 1.20 the keyword FUNCTION comprises some additional features to describe functional structure and dependencies. The keyword FUNCTION\_LIST is going to be canceled at ASAP2 version 2.00.
- > 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. In 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). This keyword indicates the adjustable object to be read only at all (table values and axis points). The optional keyword used at **AXIS\_DESCR** record indicates the related axis points to be read only.
- > **GUARD\_RAILS:** This keyword is used to indicate that an adjustable **CURVE** or **MAP** uses guard rails. The Measurement and Calibration System does not allow the user to edit the outermost values of the adjustable object (see **GUARD\_RAILS**).
- > **MAP\_LIST:** For the adjustable object type **CUBOID** which are 'sliced', this keyword specifies the **MAPs** which comprise the cuboid.
- > **MAX\_REFRESH:** Maximum refresh rate of this (adaptive) characteristic in the control unit. The existence of the keyword implies that the value of the characteristic is changed by the control unit (adaptive characteristics).
- > **DEPENDENT\_CHARACTERISTIC:** Describes the formula and references to characteristics, upon which this characteristic depends on.  
**Note:** The dependence graph described by the dependence relation must be acyclic. This must be ensured by the producer of the ASAP2 file. This keyword is only valid for characteristics of type **VALUE**
- > **VIRTUAL\_CHARACTERISTIC:** Marks a characteristic as being virtual, i.e. not existing in the memory of the control unit. The address can therefore be ignored for virtual characteristic. Initial value of the virtual characteristic depends on the values of other characteristic.  
**Note:** The corresponding graph (in analogy to the dependence graph) must also be acyclic and each sink of the graph must be a non virtual characteristic. This must be ensured by the producer of the ASAP2 file. This keyword is only valid for characteristics of type **VALUE**.
- > **REF\_MEMORY\_SEGMENT:** Reference to the memory segment which is needed if the address is not unique (this occurs in the case of lapping address ranges (overlapping memory segments)).
- > **IF\_DATA:** Data 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.
- > **ANNOTATION:** Set of notes (represented as multi-line ASCII description texts) which are related. Can serve e.g. as application note.
- > **COMPARISON\_QUANTITY:** This keyword references a valid **MEASUREMENT** in the ASAP2 file. Semantic Interpreta-

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



- > CALIBRATION\_ACCESS: This keyword specifies the access of the characteristic for calibration. Use it instead of the READ\_ONLY Attribute.
- > MATRIX\_DIM Shows the size and dimension of a multidimensional characteristic (e.g. VAL\_BLK). If the MATRIX\_DIM keyword is used, then the option NUMBER is not needed. However, if the keywords NUMBER and MATRIX\_DIM are both used, the resulting value in NUMBER must be the same as  $xDim * yDim * zDim$  for MATRIX\_DIM. If the keyword is missing the array has only one dimension with the size given at NUMBER.
- > ECU\_ADDRESS\_EXTENSION: This keyword is an additional address information. For instance it can be used, to distinguish different address spaces of an ECU (multi-microcontroller devices).

## 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 5000.0   /* lower limit, upper limit */
MAX_REFRESH                3 15 /* 15 msec */
/begin DEPENDENT_CHARACTERISTIC "sin(X1)" ALPHA
/end DEPENDENT_CHARACTERISTIC
/begin VIRTUAL_CHARACTERISTIC „sqrt(X1)" B_AREA
/end VIRTUAL_CHARACTERISTIC
REF_MEMORY_SEGMENT Data1
/begin FUNCTION_LIST      NL_ADJUSTMENT FL_ADJUSTMENT SPEED_LIM
/end FUNCTION_LIST
/begin IF_DATA            DIM EXTERNAL INDIRECT /end IF_DATA
/begin AXIS_DESCR        /* description of X-axis points */
                           STD_AXIS     /* standard axis points */
                           N             /* reference to input quantity */
                           CON_N         /* conversion */
                           13            /* maximum number of axis points */
                           0.0 5800.0    /* lower limit, upper limit */
                           MAX_GRAD     /* 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 43.0      /* lower limit, upper limit */
/end AXIS_DESCR
/end CHARACTERISTIC

```

### 6.3.28 CHECKSUM

**Prototype:**

```
/begin CHECKSUM          ChecksumDll  
/end CHECKSUM
```

**Parameters:**

string ChecksumDll: Reference to DLL file name containing the ECU checksum algorithm.

**Description:**

Description of the checksum algorithm implemented in the ECU. The ECU supplier provides a DLL.

**Note:** This keyword is only used in the context of the IF\_DATA(MODULE) applied for the ASAP1a-CCP.

**Note:** The usage of the DLL is based on a standard API definition published in the Annex.

**Example:**

```
/begin CHECKSUM          "crc16.dll"  
/end CHECKSUM
```

### 6.3.29 COEFS

#### Prototype:

COEFS                      a b c d e f

#### Parameters:

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

#### Description:

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

$$INT = f(PHYS);$$

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

#### Example:

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

### 6.3.30 COMPARISON\_QUANTITY

**Prototype:**

COMPARISON\_QUANTITY    Name

**Parameters:**

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

**Description:**

This keyword references a valid MEASUREMENT in the ASAP2 file. Semantic Interpretation (for a CURVE, a CHARACTERISTIC with only one AXIS\_DESC) : The conventional workpoint for a -CURVE has only one input quantity (assigned to AXIS\_DESCR) and moves on the CURVE. The 'free-moving' workpoint in an xy diagram of a CURVE is described by two quantities (the conventional input quantity with the AXIS\_DESC, the x-axis, and an additional comparison quantity described as an optional attribute directly with the CURVE, the y-axis). The 'free-moving' workpoint does not move on the CURVE, but on the xy-diagram in which the CURVE is located. The crossing of the free-moving workpoint and the CURVE would describe an EVENT. Such display is required by calibration engineers of automatic transmission control (EVENT=gear shift). When this keyword with a CURVE is present, the workpoint display of the MCD system shall apply the INPUT\_QUANTITY and the COMPARISON\_QUANTITY in the xy-diagram.

### 6.3.31 COMPU\_METHOD

#### Prototype:

```

/begin COMPU_METHOD      Name LongIdentifier ConversionType Format Unit
                          [-> FORMULA]
                          [-> COEFS]
                          [-> COMPU_TAB_REF]
                          [-> REF_UNIT]

/end COMPU_METHOD

```

#### Parameters:

ident Name: unique identifier in the program for the conversion method

#### Note:

The name of the conversion method has to be unique within all conversion methods of the ASAP2 MODULE, i.e. there must not be another COMPU\_METHOD object with the the same identifier in the MODULE.

string LongIdentifier: comment, description

enum ConversionType: possible types:

TAB\_INTP: table with interpolation  
TAB\_NOINTP: table without interpolation  
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 COEFS 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. The format string should never be empty as "".

string Unit: physical unit

#### Optional parameters:

-> FORMULA: Formula to be used for the conversion

-> COEFS: 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).

-> REF\_UNIT: This keyword is used to reference a measurement unit (e.g. an object of type UNIT). The string parameter Unit is a redundant information because the record referenced by REF\_UNIT contain it too. Just for the purpose of compatibility with previous versions of ASAP2 the parameter REF\_UNIT is optional. The string parameter Unit is going to be canceled at ASAP2 version 2.00.

### 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 */
                          /begin FORMULA    "3*X1/100 + 22.7" /end FORMULA
/end COMPU_METHOD

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

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

```

```

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

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

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

/ begin COMPU_METHOD      SHORTINT
                           "Conversion method for SHORTINT"
                           RAT_FUNC
                           "%4.0"
                           "NO_PHYSICAL_QTY"
                           COEFFS 0 1 0 0 0 1
/ end COMPU_METHOD

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

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

```

```

/begin COMPU_METHOD    LONGWORD
                        "Conversion method for LONGWORD"
                        RAT_FUNC
                        "%10.0"
                        "NO_PHYSICAL_QTY"
                        COEFFS 0 1 0 0 1

/end COMPU_METHOD

```

```

/begin COMPU_METHOD    LONGINT
                        "Conversion method for LONGINT"
                        RAT_FUNC
                        "%11.0"
                        "NO_PHYSICAL_QTY"
                        COEFFS 0 1 0 0 1

/end COMPU_METHOD

```

**Note:** For the next example, X is overall length of the display format which is sum of number of whole number digits, value one for radix point and number of fractional digits. Y is number of decimal places for radix point.  
<Physical\_Units> can be "NO\_PHYSICAL\_QTY"

Z is the coefficient required for the conversion.  
 $Z = 2^{\text{<radix>}}$ .

```

/begin COMPU_METHOD    FIXED_<sign><length>_<radix>
                        "Conversion method for FIXED_<sign><length>_<radix>"
                        RAT_FUNC
                        "%X.Y"
                        "<Physical_Units>"
                        COEFFS 0 Z 0 0 1

/end COMPU_METHOD

```

### 6.3.32 COMPU\_TAB

#### Prototype:

```

/begin COMPU_TAB      Name LongIdentifier ConversionType NumberValuePairs
                        { InVal OutVal }*
                        [->DEFAULT_VALUE]

/end COMPU_TAB

```

#### Parameters:

ident Name:	unique identifier in the program for the conversion table <b><u>Note:</u></b> The name of the conversion table has to be unique within all conversion tables of the ASAP2 MODULE, i.e. there must not be another COMPU_TAB, COMPU_VTAB or COMPU_VTAB_RANGE object with the the same identifier in the MODULE.
string LongIdentifier:	comment, description
enum ConversionType:	following types are possible: TAB_INTP:           table with interpolation TAB_NOINTP:        table without interpolation <b><u>Note:</u></b> This parameter is redundant information because the record defined with COMPU_METHOD also contains it.
int NumberValuePairs:	number of successive value pairs for this conversion table
float InVal:	axis point
float OutVal:	axis value

#### Optional parameters

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

#### 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

```

### 6.3.33 COMPU\_TAB\_REF

**Prototype:**

COMPU\_TAB\_REF                      ConversionTable

**Parameters:**

ident ConversionTable: reference to the data record which contains the conversion table (see COMPU\_TAB).

**Description:**

Reference to the data record which contains the conversion table (see keyword COMPU\_TAB).

**Note:** COMPU\_TAB\_REF may only refer to objects of type COMPU\_TAB, COMPU\_VTAB or COMPU\_VTAB\_RANGE.

**Example:**

COMPU\_TAB\_REF                      TEMP\_TAB                      /\* TEMP\_TAB: conversion table \*/

### 6.3.34 COMPU\_VTAB

#### Prototype:

```

/begin COMPU_VTAB      Name LongIdentifier ConversionType NumberValuePairs
                        { InVal OutVal }*
                        [->DEFAULT_VALUE]

/end COMPU_VTAB

```

#### Parameters:

ident Name: unique identifier in the program for the verbal conversion table

##### **Note:**

The name of the conversion table has to be unique within all conversion tables of the ASAP2 MODULE, i.e. there must not be another COMPU\_TAB, COMPU\_VTAB or COMPU\_VTAB\_RANGE object with the the same identifier in the MODULE.

string LongIdentifier: comment, description

enum ConversionType: at present only the following types are possible:

TAB\_VERB: verbal conversion table

**Note:** This parameter is a redundant information because the record defined with COMPU\_METHOD also contains it. Therefore this parameter is going to be canceled at ASAP2 version 2.00.

int NumberValuePairs: number of successive value pairs for this conversion table

float InVal: internal value

**Note:** Datatype “float” is used for the input value of COMPU\_TAB and COMPU\_VTAB. Since the accepted use case of COMPU\_VTAB input values are intergers, all float values are rounded to the nearest integer by following the formula  $(0.5 \leq x_{\text{raw}} < 1.5 \text{ lead to } x = 1)$ . Only float input values used with COMPU\_VTAB\_RANGE remain float values without truncation. If you want to use float as internal values (without truncation), you should use COMPU\_VTAB\_RANGE.

string OutVal: description (meaning) of the corresponding byte value

#### Optional parameters

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

#### Description:

Conversion table for the 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
/begin COMPU_VTAB      CT_DiagStatus
                        ""
                        TAB_VERB              /* convers_type */
                        3                      /* number_value_pairs */
                        0 "C_Fail"
                        1 "C_Pass"
                        2 "C_Indeterminate"

/end COMPU_VTAB
```

---

### 6.3.35 COMPU\_VTAB\_RANGE

#### Prototype:

```

/ begin COMPU_VTAB_RANGE
    Name LongIdentifier NumberValueTriples
    { InValMin InValMax OutVal } *
    [->DEFAULT_VALUE]
/ end COMPU_VTAB_RANGE

```

#### Parameters:

ident Name: unique identifier in the program for the verbal range based conversion table

#### Note:

The name of the conversion table has to be unique within all conversion tables of the ASAP2 MODULE, i.e. there must not be another COMPU\_TAB, COMPU\_VTAB or COMPU\_VTAB\_RANGE object with the the same identifier in the MODULE.

string LongIdentifier: comment, description

int NumberValueTriples: number of successive value triples for this verbal range based conversion table

float InValMin: lower limit as float value, needs to be integer ECU value when assigned to “non-float” definitions.

float InValMax: upper limit as float value, needs to be integer ECU value when assigned to “non-float” definitions.

string OutVal: display string for the value range

#### Optional parameters

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

#### Description:

Conversion table for the assignment of display strings to a value range. In particular this is useful for ASAP2 definitions with the data type ‘floating point’ (referred as FLOAT definitions).

For FLOAT definitions, the declared string is displayed for

InValMin <= ECU value < InValMax, with InValMin, InValMax as floating point values.

For non-FLOAT definitions, the declared string is displayed for

InValMin <= ECU value <= InValMax, with InValMin, InVal as integer values.



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

**Note:** Overlapping ranges may not be declared. The ASAP2 file is invalid in case of overlapping ranges within COMPU\_VTAB\_RANGE. But still, the upper limit of one range may be the same FLOAT value than the lower limit of the following range in case of a FLOAT definition (see display rules).

**Note:** When a COMPU\_METHOD with COMPU\_VTAB\_RANGE is used for calibration (writing of values to ECU), the InValMin is used when the assigned STRING(OutVal) is selected in the user interface.

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

**Example:**

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

### 6.3.36 CPU\_TYPE

#### Prototype:

CPU\_TYPE                      CPU

#### Parameters:

    string CPU:              CPU identifier

#### Description:

    CPU identification

#### Example:

CPU\_TYPE                      "INTEL 4711"

### 6.3.37 CURVE\_AXIS\_REF

#### Prototype:

CURVE\_AXIS\_REF            CurveAxis

#### Parameters:

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

#### Description:

This keyword is used in conjunction with AXIS\_DESCR definitions that use the CURVE\_AXIS attribute. It is used to specify the adjustable CURVE CHARACTERISTIC that is used to normalize or scale the axis. See Appendix C for more details.

**Note:** The same parameters for MaxAxisPoints apply as those for AXIS\_DESCR.

#### Example:

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

```

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

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

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

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

```

### 6.3.38 CUSTOMER

**Prototype:**

CUSTOMER                      Customer

**Parameters:**

        string Customer:          customer name

**Description:**

        This keyword allows a customer name to be specified.

**Example:**

CUSTOMER                      "LANZ - Landmaschinen"

### 6.3.39 CUSTOMER\_NO

**Prototype:**

CUSTOMER\_NO                      Number

**Parameters:**

        string Number:              customer number

**Description:**

        Customer number as string.

**Example:**

CUSTOMER\_NO                      "191188"

#### 6.3.40 DATA\_SIZE

##### Prototype:

DATA\_SIZE                      Size

##### Parameters:

int Size:                      data size in bits

##### Description:

Data size in bits

##### Example:

DATA\_SIZE                      16

### 6.3.41 DEF\_CHARACTERISTIC

#### Prototype:

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

#### Parameters:

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

#### Description:

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

**Note:** DEF\_CHARACTERISTIC may only refer to objects of type AXIS\_PTS or CHARACTERISTIC.

#### Example:

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



### 6.3.42 DEFAULT\_VALUE

**Prototype:**

DEFAULT\_VALUE    display\_string

**Parameters:**

string display\_string

**Description:**

Optional String which can be applied with COMPU\_TAB, COMPU\_VTAB and COMPU\_VTAB\_RANGE, used as OutVal for display when the ECU value is out of any declared range. This string shall not be selectable for calibration (when writing to the ECU).

**Example:**

DEFAULT\_VALUE “overflow\_state“

### 6.3.43 DEPENDENT\_CHARACTERISTIC

#### Prototype:

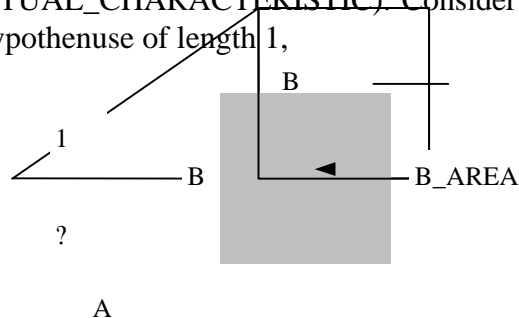
```
/begin DEPENDENT_CHARACTERISTIC      Formula (Characteristic)*
/end DEPENDENT_CHARACTERISTIC
```

#### Parameters:

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

#### Description:

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



where the length of the other sides are the characteristics A and B. When adjusting A the characteristic B has to be adjusted accordingly to  $B = \sqrt{1 - A^2}$ . The relation between the involved characteristics is described on the physical level. Also other characteristic might depend on B, e.g.  $B\_AREA = B * B$ . A dependent characteristic should not be adjustable by itself, but only through the adjustment of a characteristic it depends on.

The following example makes clear how the calibration process takes place. Assume for each of the characteristics A, B, and B\_AREA a conversion formula of  $hex = f(phys) = 100 * phys$  and assume that the value  $A_{hex}$  is 60 (decimal). Then  $A_{phys} = A_{hex} / 100 = 0.6$ . According to the formula  $B = \sqrt{1 - A^2}$ ,  $B_{phys} = 0.8$  and  $B_{hex} = B_{phys} * 100 = 80$  (decimal). According to  $B\_AREA = B * B$ , we have  $B\_AREA_{phys} = 0.64$  and therefore  $B\_AREA_{hex} = 64$  (decimal).

Used in CHARACTERISTIC.

#### Example:

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



### 6.3.45 DISPLAY\_IDENTIFIER

**Prototype:**

DISPLAY\_IDENTIFIER      display\_name

**Parameters:**

ident display\_name

**Description:**

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

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

**Example:**

DISPLAY\_IDENTIFIER      load\_engine

### 6.3.46 DIST\_OP\_X/\_Y/\_Z

#### Prototype:

DIST\_OP\_X/\_Y/\_Z                      Position Datatype

#### Parameters:

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

#### Description:

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

$$X_i = \text{Offset} + (i - 1) * \text{Distance} \quad i = \{ 1 \dots \text{numberofaxispts} \}$$

or

$$Y_k = \text{Offset} + (k - 1) * \text{Distance} \quad k = \{ 1 \dots \text{numberofaxispts} \}$$

or

$$Z_m = \text{Offset} + (m - 1) * \text{Distance} \quad m = \{ 1 \dots \text{numberofaxispts} \}$$

#### Example:

DIST\_OP\_X                      21 UWORD



### 6.3.48 ECU\_ADDRESS

#### Prototype:

ECU\_ADDRESS                      Address

#### Parameters:

long Address	Address of the measurement in the memory of the control unit.
--------------	---

#### Description:

ECU\_ADDRESS is used to describe the address of a measurement. It should replace the specific IF\_DATA (IF\_DATA ASAP1B\_ADDRESS). It can be used in MEASUREMENT only.

#### Example:

ECU\_ADDRESS                      0x12FE



### 6.3.49 ECU\_ADDRESS\_EXTENSION

#### Prototype:

ECU\_ADDRESS\_EXTENSION                      Extension

#### Parameters:

int Extension:                      Address extension of the ECU address

#### Description:

This keyword is used to specify additional address information. For instance it can be used, to distinguish different address spaces of an ECU (multi-microcontroller devices). ECU\_ADDRESS\_EXTENSION is an optional keyword of MEASUREMENT, AXIS\_PTS and CHARACTERISTIC.

#### Example:

```

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

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

```

### 6.3.50 ECU\_CALIBRATION\_OFFSET

#### Prototype:

ECU\_CALIBRATION\_OFFSET                      Offset

#### Parameters:

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

#### Description:

ECU\_CALIBRATION\_OFFSET is used to describe a fixed address offset when accessing characteristics in the control unit due to

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

#### Example:

ECU\_CALIBRATION\_OFFSET                      0x1000



### 6.3.52 ERROR\_MASK

**Prototype:**

ERROR\_MASK Mask

**Parameters:**

long Mask:                      mask to mask out selected bits

**Description:**

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

**Example:**

ERROR\_MASK 0x00000001

### 6.3.53 EVENT\_GROUP

**Prototype:**

```
/begin EVENT_GROUP      RasterGrpName ShortName  
                        {RasterID}*  
/end EVENT_GROUP
```

**Parameters:**

string RasterGrpName:	Name of the Group of Event Channels (name for one sample rate of an ECU supported data acquisition mechanism, i.e. the ASAP1a-CCP)
string ShortName:	Short Display Name of the Event Channel Name <u>Recommendation:</u> The string length shall not exceed 9 characters
int RasterID:	Event Channel No., references to the data acquisition event channels of the ECU which are a member of this group.

**Description:**

Each available „service“ can be described by one ECU\_DAQ\_EVENT within IF\_DATA(Module) and then can be referenced within the ASAP1b-QP\_BLOB statement of the SOURCE keyword.

**Note:** This keyword is only used in the context of the IF\_DATA(MODULE) applied for the ASAP1a-CCP.

**Example:**

```
/begin EVENT_GROUP      "Group of Events" "G1" 2 5 8 9  
/end EVENT_GROUP
```

### 6.3.54 EXTENDED\_LIMITS

#### Prototype:

EXTENDED\_LIMITS            LowerLimit UpperLimit

#### Parameters:

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

**Note:** The limit values are to be interpreted as physical values and not internal values.

#### 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

**6.3.55 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:

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

or

$$Y_k = \text{Offset} + (k - 1) * 2^{\text{Shift}} \quad k = \{ 1 \dots \text{numberapo} \}$$

or

$$Z_m = \text{Offset} + (m - 1) * 2^{\text{Shift}} \quad m = \{ 1 \dots \text{numberapo} \}$$

**Remark:**

This keyword is equivalent to FIX\_AXIS\_PAR\_DIST but differs in parameter 'Shift' (see FIX\_AXIS\_PAR\_DIST).

**Example:**

FIX\_AXIS\_PAR                      0 32 8

### 6.3.56 FIX\_AXIS\_PAR\_DIST

#### Prototype:

FIX\_AXIS\_PAR\_DIST            Offset Distance Numberapo

#### Parameters:

int Offset:	'offset' parameter to calculate the axis points of fixed characteristic curves or maps (see description).
int Distance:	'distance' parameter to calculate the axis points of fixed characteristic curves or maps (see description).
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 'distance'. In the current deposit methods both parameters are contained in the description file. In future deposit methods both methods could well be part of the deposit structure of the adjustable objects.

The axis points of fixed characteristic curves or maps are calculated as follows:

$$X_i = \text{Offset} + (i - 1) * \text{Distance} \quad i = \{ 1 \dots \text{numberapo} \}$$

or

$$Y_k = \text{Offset} + (k - 1) * \text{Distance} \quad k = \{ 1 \dots \text{numberapo} \}$$

or

$$Z_m = \text{Offset} + (m - 1) * \text{Distance} \quad m = \{ 1 \dots \text{numberapo} \}$$

#### Note:

This keyword is equivalent to FIX\_AXIS\_PAR but differs in parameter 'Distance' (see FIX\_AXIS\_PAR).

#### Example:

FIX\_AXIS\_PAR\_DIST            0 100 8



### 6.3.57 FIX\_AXIS\_PAR\_LIST

#### Prototype:

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

#### Parameters:

float AxisPts_Value	List of "ECU-Original" Values as implied by the ECU algorithm. The number of values must match with the MaxAxisPoints attribute of the AXIS_DESCR referencing FIX_AXIS_PAR_LIST. The COMPU_METHOD assigned to the AXIS_DESCR shall be applied to achieve the actual display values.
---------------------	---

**Note:** The data type shall be integer in case of an assignment to a non-float definition).

#### Description:

Allows the description of any value combination of a virtual axis (FIX\_AXIS, axis points not in the ECU memory). Other methods (FIX\_AXIS\_PAR, FIX\_AXIS\_PAR\_DIST) implicitly assume an interpolation algorithm in the ECU. But axis descriptions are also used e.g. to span status tables.

The values are the input for the COMPU\_METHOD assigned to the axis. Even a verbal table could be applied as COMPU\_METHOD (i.e for the axis description of status tables on which no interpolation is applied).

#### Example:

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

### 6.3.58 FIX\_NO\_AXIS\_PTS\_X/\_Y/\_Z

#### Prototype:

FIX\_NO\_AXIS\_PTS\_X/\_Y/\_Z NumberOfAxisPoints

#### Parameters:

int NumberOfAxisPoints: 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

### 6.3.59 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 DataType:              data type of the table values

enum IndexMode:                  for characteristic maps, this attribute is used to describe how the 2-dimensional table values are mapped onto the 1-dimensional address space:

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).

For characteristic cuboids each XY plane is mapped as above. The cuboid is stored as an array of maps with incremental Z coordinates.

#### Additional indexMode (since version 1.2):

##### ALTERNATE\_WITH\_X

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

curves: table values and X-coordinate values are deposited alternating.

##### ALTERNATE\_WITH\_Y

maps:    deposited in rows, the rows of table values alternate with the respective Y-coordinates (maps only).

## ALTERNATE\_CURVES

curves: curves which share a common axis are deposited in columns; each row of memory contains values for all the shared axis curves at a given axis breakpoint. Required in order to represent characteristics which correspond to arrays of structures in ECU program code. In the example code below, DT10, DT20, etc are treated as separate curves which may have different conversions or limits:-

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

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

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

### Description:

Description of the table values (function values) of an adjustable object. If the ALTERNATE option is used, the position parameter of values and axis-points indicates their order.

### Example:

FNC\_VALUES                      7 SWORD COLUMN\_DIR DIRECT

### 6.3.60 FORMAT

**Prototype:**

FORMAT                                      FormatString

**Parameters:**

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

**Description:**

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

**Example:**

FORMAT                                      "%4.2"

### 6.3.61 FORMULA

#### Prototype:

```

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

```

#### 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
	logical OR
>>	shift right
<<	shift left
XOR	exclusive OR
~	logical NOT

##### Trigonometric functions:

sin(x), cos(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)

**Note:** Parameter names should be used as follows:  
X or X1 if only one input quantity is possible,  
X1, X2, ... in case of several input quantities (applies only to VIRTUAL\_MEASUREMENT), it is suggested to not use X, Y, Z ... in this case.

### 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.

**Note:** When converting from internal values to physical values, the use of “X” is customary. For reasons of backward compatibility, if only one input value is necessary, use either X or X1.

### Example:

```
/begin FORMULA          "sqrt( 3 - 4*(sin(X1))^2 )"
/end FORMULA
```

**Note:** Do not use FORMULA to describe rational fractions if the conversion can be defined using the COEFFS keyword.

### 6.3.62 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 110.
--------------	---

#### 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.

**Note:** FORMULA\_INV is necessary if used for CHARACTERISTIC objects. Only MEASUREMENT objects do not need an inverse formula.

#### Example:

Inversion function e.g. for keyword FORMULA (see page 110)

FORMULA\_INV                      "arcsin( sqrt( (3 - (X1)^2)/4 ) )"



### 6.3.63 FRAME

#### Prototype:

```

/begin FRAME
    Name
    LongIdentifier
    ScalingUnit
    Rate
    [-> FRAME_MEASUREMENT]
    {-> IF_DATA} *

/end FRAME

```

#### Parameters:

ident Name:	<p><b>unique</b> identifier in the program, referencing is based on this 'name'</p> <p><b><u>Note:</u></b> The name of the frame has to be unique within all frames of the ASAP2 MODULE, i.e. there must not be another FRAME object with the the same identifier in the MODULE.</p>
string LongIdentifier:	comment, description
int ScalingUnit:	This parameter defines the basic scaling unit. The following parameter 'Rate' relates on this scaling unit. The value of ScalingUnit is coded as shown in 'Table 4: Codes for scaling units (CSE)' (page144).
long Rate:	The maximum refresh rate of the concerning measurement source in the control unit. The unit is defined with parameter 'ScalingUnit'.

#### Optional parameters:

-> FRAME_MEASUREMENT:	Use this keyword to define the frames measurementobjects.
-> IF_DATA:	Interface-specific description data (BLOB: binary large object) used at ASAP1b device at call of the command InitRead(). The parameters associated with this keyword are described in the ASAP2 metalanguage (in short A2ML) by the ECU supplier or the interface module supplier.

#### Description:

For the structuring of a car network involving a very large number of measuring channels, function frames can be defined.

These function frames shall be used in the application system to allow the selection lists for the selection of measuring channels to be represented in a structured manner on the basis of functional viewpoints (function orientation).

This will also be used to describe the packaging of measurement data into sources for CAN frames in a network environment.

**Example:**

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

### 6.3.64 FRAME\_MEASUREMENT

**Prototype:**

FRAME\_MEASUREMENT { Identifier } \*

**Parameters:**

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

**Description:**

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

**Example:**

FRAME\_MEASUREMENT      WHEEL\_REVOLUTIONS ENGINE\_SPEED

### 6.3.65 FUNCTION

#### Prototype:

```

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

/end FUNCTION

```

#### Parameters:

ident Name:	unique Identifier in the program, referencing is based on this 'name'
-------------	---

#### Note:

The name of the function has to be unique within all functions of the ASAP2 MODULE, i.e. there must not be another FUNCTION object with the the same identifier in the MODULE.

string LongIdentifier:	comment, description
------------------------	----------------------

#### Optional parameters:

-> ANNOTATION:	Set of notes (represented as multi-line ASCII description texts) which are related. Can serve e.g. as application note.
-> DEF_CHARACTERISTIC:	This keyword can be used to define those adjustable objects which are <b>defined</b> in respective function.
-> REF_CHARACTERISTIC:	If the function contains <b>references</b> to some adjustable objects, this keyword can be used to describe this references.
-> IN_MEASUREMENT:	Use this keyword to define the input measurement objects of respective function (input variables).
-> OUT_MEASUREMENT:	Use this keyword to define the output measurement objects of respective function (output variables).
-> LOC_MEASUREMENT:	Use this keyword to define the local measurement objects of respective function (local variables: scope is limited to this function).
-> SUB_FUNCTION:	This keyword can be used to describe the function hierarchy. If the respective function is subdivided into subfunctions, use this keyword to define the subfunctions.

-> FUNCTION\_VERSION: String to define the version of the function. An application tool should be able to handle function oriented characteristic data.

### 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, hierachial manner following the order of input on the basis of functional viewpoints (function orientation).

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

### Example:

```

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

```

### 6.3.66 FUNCTION\_LIST

#### Prototype:

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

#### Parameters:

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

#### Description:

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

**Note:** Since ASAP2 version 1.20 the keyword FUNCTION comprises some additional features to describe functional structure and dependencies. The keyword FUNCTION\_LIST is going to be canceled at ASAP2 version 2.00.

#### Example:

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

### 6.3.67 FUNCTION\_VERSION

**Prototype:**

FUNCTION\_VERSION      VersionIdentifier

**Parameters:**

string VersionIdentifier: short identifier for the version

**Description:**

String for identification of the version of a function with maximum MAX\_STRING (at present 255) characters.

**Example:**

FUNCTION\_VERSION      "BG5.0815"

### 6.3.68 GROUP

#### Prototype:

```

/begin GROUP      GroupName GroupLongIdentifier
                  {-> ANNOTATION} *
                  [-> ROOT]
                  [-> REF_CHARACTERISTIC]
                  [-> REF_MEASUREMENT]
                  [-> FUNCTION_LIST]
                  [-> SUB_GROUP]

/end GROUP

```

#### Parameters:

ident GroupName: unique identifier of the group

##### Note:

The name of the group has to be unique within all groups of the ASAP2 MODULE, i.e. there must not be another GROUP object with the the same identifier in the MODULE.

string GroupLongIdentifier: Comment, description of the group within a grouping mechanism.

#### Optional parameters:

- > ANNOTATION: Set of notes (represented as multi-line ASCII description texts) which are related. Can serve e.g. as application note.
- > ROOT: This keyword indicates that the group constitutes an independent grouping mechanism (root level) which the MCD system may use as a root point for the hierarchical presentation of groups. All groups referenced via SUB\_GROUP (including nested references) constitute a set of groups belonging to the grouping mechanism.  
Examples for such grouping mechanisms :  
Group Name = {Software\_Components, Calibration\_Components, Editor\_Selection\_Lists}
- > REF\_CHARACTERISTIC: If the group contains **references** to some adjustable objects, this keyword can be used to describe these references.
- > REF\_MEASUREMENT: If the group contains **references** to some measurement objects, this keyword can be used to describe these references.
- > FUNCTION\_LIST: This keyword can be used to specify a list of references to functions.
- > SUB\_GROUP: This keyword can be used to describe the group hierarchy. If the respective group is subdivided into



sub-groups, use this keyword to define the sub-groups. In particular, SUB\_GROUP references the groups belonging to a grouping mechanism indicated with the optional keyword ROOT (see above).

**Description:**

These GROUPs shall be used in the application system to provide selection lists (groups) of adjustable objects and measuring channels.

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

**Example:**

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

/begin GROUP INJE "Subsystem Injection"
    /begin SUB_GROUP                injec1 injec2
    /end SUB_GROUP
/end GROUP

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

/begin GROUP Injec2 "Module filename Injec2"
    /begin REF_CHARACTERISTIC        INJECTION_ADJUST
    /end REF_CHARACTERISTIC
    /begin REF_MEASUREMENT GAS_INPUT WHEEL_SPEED
    /end REF_MEASUREMENT
/end GROUP
```

```
/begin GROUP C6TD "Shift Point Control"
  /begin SUB_GROUP          c6tdvder c6tdertf
  /end SUB_GROUP
/end GROUP

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

/begin GROUP c6tdertf "Module filename c6tdertf"
  /begin REF_CHARACTERISTIC          LUP23_CURVE
  /end REF_CHARACTERISTIC
  /begin REF_MEASUREMENT  TRANSMISSION_SP ENGINE_SPEED
  /end REF_MEASUREMENT
/end GROUP

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

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

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

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

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

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

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

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

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

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

### 6.3.69 GUARD\_RAILS

#### Prototype:

GUARD\_RAILS

#### Description:

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

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

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

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

#### Example:

```

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

```

### 6.3.70 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

```

### 6.3.71 IDENTIFICATION

#### Prototype:

IDENTIFICATION                      Position Datatype

#### Parameters:

int Position:	position of the 'identifier' in the deposit structure.
datatype Datatype:	word length of the 'identifier'

#### Description:

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

#### Example:

IDENTIFICATION                      1 UWORD

### 6.3.72 IF\_DATA (AXIS\_PTS, CHARACTERISTIC, MEMORY\_LAYOUT)

#### Prototype:

```

/begin IF_DATA          Name
                        [-> DP_BLOB Dp_Data]
                        [-> PA_BLOB Pa_Data]
/end IF_DATA

```

#### Parameters:

ident Name:	identifier of ASAP device. This string must be defined in the A2ML-file and should match to ASAP1b. See chapter 8.2 (Designing A2ML-file) for details.
-------------	--

#### Optional parameters:

-> DP_BLOB:	Interface-specific description data (BLOB: binary large object) used at ASAP1b device at call of the command InitAccess(). The parameters associated with this keyword are described in the ASAP2 metalanguage (in short A2ML) by the ECU supplier or the interface module supplier.
-> PA_BLOB:	Interface-specific description data (BLOB: binary large object) used at ASAP1b device at call of the command Access(). The parameters associated with this keyword are described in the ASAP2 metalanguage (in short A2ML) by the ECU supplier or the interface module supplier.

#### Description:

Definition of interface-specific description data (topic keyword AXIS\_PTS, CHARACTERISTIC and MEMORY\_LAYOUT).

#### Example:

```

/begin IF_DATA
  ASAP1B_CCP          /* Name of ASAP device */
  DP_BLOB             /* interface-specific parameters described in A2ML */
    0x12129977 0xFF
  PA_BLOB             /* interface-specific parameters described in A2ML */
    "Pumpenkennfeld" 1 2 17
/end IF_DATA

```



### 6.3.73 IF\_DATA (MEASUREMENT)

#### Prototype:

```

/begin IF_DATA      Name
                    [-> KP_BLOB KP_Data]
                    [-> DP_BLOB Dp_Data]
                    [-> PA_BLOB Pa_Data]
/end IF_DATA

```

#### Parameters:

ident Name: identifier of ASAP device. This string must be defined in the A2ML-file. See chapter 8.2 (Designing A2ML-file) for details.

#### Optional parameters:

-> KP\_BLOB: Interface-specific description data (BLOB: binary large object) used at ASAP1b device at call of the command InitRead(). The parameters associated with this keyword are described in the ASAP2 metalanguage (in short A2ML) by the ECU supplier or the interface module supplier.

-> DP\_BLOB: Interface-specific description data (BLOB: binary large object) used at ASAP1b device at call of the command InitAccess(). The parameters associated with this keyword are described in the ASAP2 metalanguage (in short A2ML) by the ECU supplier or the interface module supplier.

**Note:** This keyword is required only if MCD wants to 'write' to the corresponding measurement object (see keyword READ\_WRITE of topic MEASUREMENT).

-> PA\_BLOB: Interface-specific description data (BLOB: binary large object) used at ASAP1b device at call of the command Access(). The parameters associated with this keyword are described in the ASAP2 metalanguage (in short A2ML) by the ECU supplier or the interface module supplier.

**Note:** This keyword is required only if MCD wants to 'write' to the corresponding measurement object (see keyword READ\_WRITE of topic MEASUREMENT).

**Description:**

Definition of interface-specific description data (topic keyword MEASUREMENT).

**Example:**

```
/begin IF_DATA
  ASAP1B_CCP          /* Name of ASAP device */
  KP_BLOB 0x12FE 17 3  /* interface-specific parameters described in A2ML */
  DP_BLOB 0x121277 0xFF /* interface-specific parameters described in A2ML */
  PA_BLOB             /* interface-specific parameters described in A2ML */
    "Pumpenkennfeld" 1 2 17
/end IF_DATA
```

**Minimal Requirements for Data Acquisition:**

The interface-specific parameters of measurement objects can be described using keyword IF\_DATA.

On the one hand this keyword is optional, i.e. the description file is ASAP2-conform even if this keyword is missing. On the other hand in real application access to any measurement object is only possible if a suitable IF\_DATA record is available.

Recommendation: If the description file doesn't include any interface-specific parameters corresponding to real application system, description file should either contain a record to describe an emulator address (IF\_DATA ASAP1B\_ADDRESS) or description parameters to acquire the measurement data from CAN-bus (IF\_DATA ASAP1B\_CAN). Usage of one or both of this IF\_DATA-records depends on real case.

#### 6.3.74 IF\_DATA ASAP1B\_ADDRESS (MEASUREMENT)

##### Prototype:

```
/begin IF_DATA ASAP1B_ADDRESS KP_BLOB Address  
/end IF_DATA
```

##### Parameters:

ASAP1B_ADDRESS:	Identifier of ASAP device.
KP_BLOB:	This keyword marks the interface-specific parameters used at ASAP1b device at call of the command InitRead().
long Address:	Address of measurement object (e.g. emulator RAM address)

##### Description:

Definition of interface-specific description data (topic keyword MEASUREMENT). This record is recommended if there isn't any other record available, corresponding to a real application system.

##### Example:

```
/begin IF_DATA      ASAP1B_ADDRESS      KP_BLOB      0x12FE  
/end IF_DATA
```

### 6.3.75 IF\_DATA ASAP1B\_CAN (MEASUREMENT)

#### Prototype:

```

/begin IF_DATA ASAP1B_CAN
    /begin KP_BLOB      MessageName Identifier MessageSize
                        MessageSource Startbit DataSize
                        [-> MULTIPLEX]

    /end KP_BLOB
/end IF_DATA

```

#### Parameters:

ASAP1B_CAN:	Identifier of ASAP device.
KP_BLOB:	This keyword marks the interface-specific parameters used at ASAP1b device at call of the command InitRead().
ident MessageName:	Description of CAN-message.
long Identifier:	CAN-Identifier of that message, the expected measurement object is included in.
int MessageSize:	Size of CAN-message (message data only).
string MessageSource:	Description of message sender.
int Startbit:	The expected measurement data are included in message data at offset 'Startbit' (in bits, index of first bit of message data is 0).
int DataSize:	Size of measurement data included in message data.

#### Optional parameters:

-> MULTIPLEX:	In case of multiplexed message data this keyword can be used to define the 'mode-signal' (see MULTIPLEX).
---------------	---

#### Description:

Definition of interface-specific description data (topic keyword MEASUREMENT). This record is recommended if there isn't any other record available corresponding to a real application system.

#### Example:

```

/begin IF_DATA ASAP1B_CAN
    /begin KP_BLOB      „MOTOINFO“ 0x123 8 „MOTOR-SG“ 5 8
    /end KP_BLOB
/end IF_DATA

```

### 6.3.76 IF\_DATA (FRAME)

#### Prototype:

```
/begin IF_DATA      Name
                    [-> QP_BLOB QP_Data]
/end IF_DATA
```

#### Parameters:

ident Name:                      identifier of ASAP device. This string must be defined in the A2ML-file. See chapter 8.2 (Designing A2ML-file) for details.

#### Optional parameters:

-> QP\_BLOB:                      Interface-specific description data (BLOB: binary large object) used at ASAP1b device at call of the command InitRead(). The parameters associated with this keyword are described in the ASAP2 metalanguage (in short A2ML) by the ECU supplier or the interface module supplier.

#### Description:

Definition of interface-specific description data (topic keyword FRAME).

### 6.3.77 IF\_DATA (MEMORY\_SEGMENT)

**Prototype:**

```
/begin IF_DATA          Name
                        {-> ADDRESS_MAPPING } *
/end IF_DATA
```

**Parameters:**

ident Name: identifier of ASAP device. This string must be defined in the A2ML-file. See chapter 8.2 (Designing A2ML-file) for details.

**Optional parameters:**

-> ADDRESS\_MAPPING: This keyword can be used to remap the address space of the ECU to an access address.

**Description:**

Remapping of the address space.

**Example:**

```
/begin MEMORY_SEGMENT ext_Ram "external RAM"
  DATA RAM EXTERN 0x30000 0x1000 -1 -1 -1 -1 -1
/end MEMORY_SEGMENT
/begin IF_DATA ASAP1B_KWP2000
  /* ADDRESS_MAPPING orig_addr      mapping_addr      length */
  ADDRESS_MAPPING    0x4000         0x6000            0x0200
/end IF_DATA
```

### 6.3.78 IF\_DATA (MODULE)

#### Prototype:

```

/begin IF_DATA
    Name
    {-> SOURCE } *
    {-> RASTER } *
    {-> EVENT_GROUP}*
    [-> SEED_KEY]
    [-> CHECKSUM]
    [-> TP_BLOB TP_Data]
/end IF_DATA

```

#### Parameters:

ident Name: Identifier of ASAP device. This string must be defined in the A2ML-file. See chapter 8.2 (Designing A2ML-file) for details.

#### Optional parameters:

-> SOURCE: This keyword can be used to describe different 'sources' of measurement, e.g. different acquisition modes (e.g. time synchronous 10 msec, time synchronous 50 msec etc.)

-> RASTER: This keyword can be used to declare an event channel which is part of the SOURCE of data acquisition.  
Only used in context of the ASAP 1a-CCP.

-> EVENT\_GROUP: This keyword can be used to declare a group of event channels (group of ECU\_DAQ\_EVENT)  
Only used in context of the ASAP 1a-CCP.

-> SEED\_KEY: Description of the authentication process. References to dynamic link libraries with a standard API, which contain the authentication algorithms.

-> CHECKSUM: Description of the ECU checksum algorithm. References to dynamic link libraries with a standard API, which contain the checksum algorithm.

-> TP\_BLOB: Interface-specific description data (BLOB: binary large object) used at ASAP1b device at call of the command InitRead(), InitAccess() and Command(). The parameters associated with this keyword are described in the ASAP2 meta-language (in short A2ML) by the ECU supplier or the interface module supplier.

#### Description:

Definition of interface-specific description data (topic keyword MODULE).

#### Example:

```

/begin IF_DATA

```

```
ASAP1B_CCP      /* Name of ASAP device */
  /begin SOURCE  Time10ms  4  1  /* Source parameter */
  /end SOURCE
  /begin SOURCE  Time50ms  4  5  /* Source parameter */
  /end SOURCE
  TP_BLOB        0xFF 0xFA 0x0A /* interface-specific parameters */
/end IF_DATA
```



### 6.3.79 IN\_MEASUREMENT

#### Prototype:

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

#### Parameters:

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

#### Description:

This keyword can be used to define input quantities of respective function.

**Note:** IN\_MEASUREMENT may only refer to objects of type MEASUREMENT.

#### Example:

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

### 6.3.80 LEFT\_SHIFT

#### Prototype:

LEFT\_SHIFT                      Bitcount

#### Parameters:

long bitcount:                  Shift ,bitcount‘ bits to the left

#### Description:

The LEFT\_SHIFT keyword is only used within the BIT\_OPERATION keyword. See description of BIT\_OPERATION.

### 6.3.81 LOC\_MEASUREMENT

#### Prototype:

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

#### Parameters:

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

#### Description:

This keyword can be used to define local quantities of respective function.

**Note:** LOC\_MEASUREMENT may only refer to objects of type MEASUREMENT.

#### Example:

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

### 6.3.82 MAP\_LIST

#### Prototype:

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

#### Parameters:

ident Name:                    identifier of a MAP (see CHARACTERISTIC)

#### Description:

This keyword can be used to specify the list of MAPs which comprise a CUBOID. This keyword is required because CUBOID data will not be at contiguous memory locations if a CUBOID is composed of several MAPs.

**6.3.83 MATRIX\_DIM****Prototype:**

MATRIX\_DIM                      xDim yDim zDim

**Parameters:**

int xDim:	number of values in dimension of x
int yDim:	number of values in dimension of y
int zDim:	number of values in dimension of z

**Description:**

This keyword is used to describe the dimensions of a multidimensioned array of values (MEASUREMENT or CHARACTERISTIC).

$xDim * yDim * zDim = \text{number of values.}$

If NUMBER or ARRAY\_SIZE are used in the CHARACTERISTIC or MEASUREMENT record the result must be the same as the value given at this option.

**Example:**

MATRIX\_DIM                      2 4 3

### 6.3.84 MAX\_GRAD

#### Prototype:

MAX\_GRAD                      MaxGradient

#### Parameters:

float MaxGradient:        maximum permissible gradient

#### Description:

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

$$\text{MaxGrad}_x = \text{maximum}(\text{absolut}((W_{i,k} - W_{i-1,k})/(X_i - X_{i-1})))$$

$$\text{MaxGrad}_y = \text{maximum}(\text{absolut}((W_{i,k} - W_{i,k-1})/(Y_i - Y_{k-1})))$$

#### Example:

MAX\_GRAD                      200.0

### 6.3.85 MAX\_REFRESH

#### Prototype:

MAX\_REFRESH                      ScalingUnit Rate

#### Parameters:

int ScalingUnit:	this parameter defines the basic scaling unit. The following parameter 'Rate' relates on this scaling unit. The value of ScalingUnit is coded as shown below in 'Table 4: Codes for scaling units (CSE)'.
long Rate:	the maximum refresh rate of the concerning measurement object in the control unit. The unit is defined with parameter 'ScalingUnit'.

#### Description:

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

#### Example:

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

Code	Unit	Referred to	Comment
0	1 ?sec	Time	
1	10 ?sec	Time	
2	100 ?sec	Time	
3	1 msec	Time	
4	10 msec	Time	
5	100 msec	Time	
6	1 sec	Time	
7	10 sec	Time	
8	1 min	Time	
9	1 hour	Time	
10	1 day	Time	
100	Angular degrees	Angle	
101	Revolutions 360 degrees	Angle	
102	Cycle 720 degrees	Angle	e.g. in case of IC engines
103	Cylinder segment	Combustion	e.g. in case of IC engines
998	When frame available	Event	Source defined in keyword Frame
999	Always if there is new value		Calculation of a new upper range limit after receiving a new partial value, e.g. when calculating a complex trigger condition
1000	Non deterministic		Without fixed scaling

Table 4: Codes for scaling units (CSE)



### 6.3.86 MEASUREMENT

#### Prototype:

```

/begin MEASUREMENT      Name LongIdentifier Datatype Conversion Resolution
                        Accuracy LowerLimit UpperLimit
                        [-> DISPLAY_IDENTIFIER]
                        [-> READ_WRITE]
                        [-> FORMAT]
                        [-> ARRAY_SIZE]
                        [-> BIT_MASK]
                        [-> BIT_OPERATION]
                        [-> BYTE_ORDER]
                        [-> MAX_REFRESH]
                        [-> VIRTUAL]
                        [->FUNCTION_LIST]
                        [-> ECU_ADDRESS]
                        [-> ERROR_MASK]
                        [-> REF_MEMORY_SEGMENT]
                        {-> ANNOTATION} *
                        {-> IF_DATA } *
                        [-> MATRIX_DIM]
                        [-> ECU_ADDRESS_EXTENSION]

/end MEASUREMENT

```

#### Parameters:

ident Name:	unique identifier in the ECU program <b>Note:</b> The name of the measuement object has to be unique within all measurement objects and adjustable objects of the ASAP2 MODULE, i.e. there must not be another AXIS_PTS, CHARACTERISTIC or MEASUREMENT object with the the same identifier in the MODULE.
string LongIdentifier:	comment, description
datatype Datatype:	data type of the measurement
ident Conversion:	Reference to the relevant record of the description of the conversion method (see COMPU_METHOD). If there is no conversion method, as in the case of CURVE_AXIS, the parameter ‘Conversion’ should be set to “NO_COMPU_METHOD” (application systems must be able to handle this case).
int Resolution:	smallest possible change in bits
float Accuracy:	possible variation from exact value in %
float LowerLimit:	plausible range of table values, lower limit
float UpperLimit:	plausible range of table values, upper limit

**Note:** The limit values are to be interpreted as physical values and not internal values.

## Optional parameters

- > DISPLAY\_IDENTIFIER: Can be used as a display name (alternative to the 'name' attribute).
- > READ\_WRITE: Keyword to mark this measurement object as 'writeable'.
- > FORMAT: With deviation from the display format specified with keyword COMPU\_TAB referenced by parameter <Conversion> a special display format can be specified to be used to display the measurement values.
- > ARRAY\_SIZE: This keyword marks a measurement object as an array of measurement values.
- > BIT\_MASK: With deviation from the standard value 0xFFFFFFFF this parameter can be used to mask out bits.
- > BIT\_OPERATION: The BIT\_OPERATION keyword can be used to perform operation on the masked out value.
- > BYTE\_ORDER: With deviation from the standard value this parameter can be used to specify the byte order (Intel format, Motorola format)
- > MAX\_REFRESH: Maximum refresh rate of this measurement in the control unit
- > VIRTUAL: For description of a virtual measurement (see VIRTUAL)
- > ERROR\_MASK: With deviation from the standard value 0x00000000 this parameter can be used to mask bits of a MEASUREMENT which indicate that the value is in error.
- > FUNCTION\_LIST: This keyword can be used to specify a list of 'functions' to which this measurement object has been allocated.  
Remark: Since ASAP2 version 1.20 the keyword FUNCTION comprises some additional features to describe functional structure and dependencies. The keyword FUNCTION\_LIST is going to be canceled at ASAP2 version 2.00.
- > 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.
- > ECU\_ADDRESS: Address of the measurement in the memory of the control unit.
- > REF\_MEMORY\_SEGMENT: Reference to the memory segment which is needed if the address is not unique (this occurs in the case of lapping address ranges (overlapping memory segments)).
- > ANNOTATION: Set of notes (represented as multi-line ASCII description texts) which are related. Can serve e.g. as application note.
- > MATRIX\_DIM Shows the size and dimension of a multidimensional measurement. If the MATRIX\_DIM keyword is used, then the option ARRAY\_SIZE is not needed. However, if the keywords ARRAY\_SIZE and MATRIX\_DIM are both used, the resulting value in ARRAY\_SIZE must be the same as  $xDim * yDim * zDim$  for MATRIX\_DIM. If the keyword is

missing the array has only one dimension with the size given at ARRAY\_SIZE.

-> ECU\_ADDRESS\_EXTENSION: This keyword is an additional address information. For instance it can be used, to distinguish different address spaces of an ECU (multi-microcontroller devices).

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

/begin MEASUREMENT      VdiagStatus /* name */
                        "VdiagStatus" /* long identifier */
                        SWORD        /* datatype */
                        CM_DiagStatus /* conversion */
                        16           /* resolution */
                        1            /* accuracy */
                        -32768        /* lower limit */
                        32767         /* upper limit */
                        ECU_ADDRESS   0x003FDfE0
/end MEASUREMENT

```

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

### 6.3.87 MEMORY\_LAYOUT

**Prototype:**

```
/begin MEMORY_LAYOUT  PrgType Address Size Offset
                      { -> IF_DATA}*
/end MEMORY_LAYOUT
```

**Parameters:**

enum PrgType:	Description of the program segments divided into:
	PRG_CODE = program code
	PRG_DATA = program data
	PRG_RESERVED = other
long Address:	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 (see Figure 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.

**Optional parameters**

-> IF_DATA:	Date record to describe the interface specific description data used at ASAP1b device. 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.
-------------	---

**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.

```
/begin MEMORY_LAYOUT  PRG_RESERVED
                      0x0000  0x0400  -1      -1      -1      -1      -1
/end MEMORY_LAYOUT
/begin MEMORY_LAYOUT  PRG_CODE
                      0x0400  0x3C00  -1      -1      -1      -1      -1
/end MEMORY_LAYOUT
/begin MEMORY_LAYOUT  PRG_DATA
                      0x4000  0x0200  0x10000  0x20000  -1      -1      -1
/end MEMORY_LAYOUT
/begin MEMORY_LAYOUT  PRG_DATA
                      0x4200  0x0E00  -1      -1      -1      -1      -1
```

```

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

```

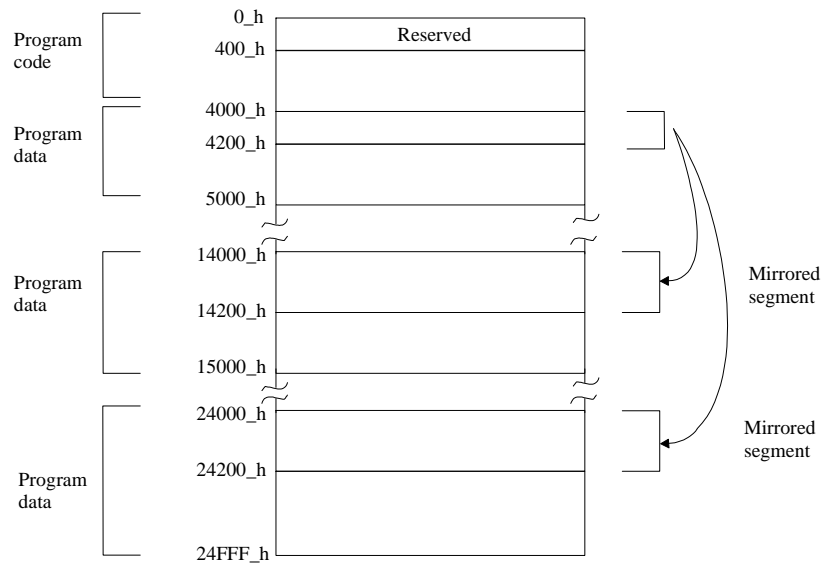


Figure 8: Memory layout (mirrored segments)

### 6.3.88 MEMORY\_SEGMENT

#### Prototype:

```

/begin MEMORY_SEGMENT Name LongIdentifier PrgType MemoryType
                        Attribute Address Size Offset
                        { -> IF_DATA}*
/end MEMORY_SEGMENT

```

#### Parameters:

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

	INTERN	= internal segment
	EXTERN	= external segment
long Address:	Initial address	
long Size:	Length of the segment	
long[5] Offset:	Offset address of mirrored segments	

### Optional Parameters:

-> IF_DATA:	Date record to describe the interface specific description data used at ASAP1b device. 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
-------------	--

### Description:

The new keyword MEMORY\_SEGMENT is used to replace the existing keyword MEMORY\_LAYOUT. The advantages of MEMORY\_SEGMENT are that they are given a name which can be used for references from IF\_DATA Blobs and the more accurate description of the memory by memory types and attributes (INTERN and EXTERN).

Used in MOD\_PAR. The keywords MEMORY\_SEGMENT and MEMORY\_LAYOUT can be used in parallel. The parameter Offset is to be used (as within the former MEMORY\_LAYOUT) to describe several mirrored segments.

MEMORY\_SEGMENTS with the same MemoryType and the same Attribute may not overlap. Also all MEMORY\_SEGMENTS with the PrgType CODE, DATA, OFFLINE\_DATA, RESERVED may not overlap mutually to get a linear address space for access on calibration data. All other MEMORY\_SEGMENTS with different MemoryType or different Attribute may however overlap, e.g. internal and external memory segments.

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

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



**SERAM / RAM**

ECU-RAM section available for serial application. For usage see also: CALIBRATION\_METHOD.

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

**Example:**

```

/begin MEMORY_SEGMENT Data1 "Data internal Flash"
    DATA FLASH INTERN 0x4000 0x0200 0x10000 -1 -1 -1 -1 -1
/end MEMORY_SEGMENT

/begin MEMORY_SEGMENT Data2 "Data external Flash"
    DATA FLASH EXTERN 0x7000 0x2000 -1 -1 -1 -1 -1
/end MEMORY_SEGMENT

/begin MEMORY_SEGMENT Code1 "Code external Flash"
    CODE FLASH EXTERN 0x9000 0x3000 -1 -1 -1 -1 -1
/end MEMORY_SEGMENT

/begin MEMORY_SEGMENT ext_Ram "external RAM"
    DATA RAM EXTERN 0x30000 0x1000 -1 -1 -1 -1 -1
/end MEMORY_SEGMENT

/begin MEMORY_SEGMENT int_Ram "internal RAM"
    DATA RAM INTERN 0x0000 0x0200 -1 -1 -1 -1 -1
/end MEMORY_SEGMENT

/begin MEMORY_SEGMENT Seram1 "emulation RAM 1"
    SERAM RAM EXTERN 0x7000 0x1000 -1 -1 -1 -1 -1
/end MEMORY_SEGMENT

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

```

### 6.3.89 MODULE

#### Prototype:

```

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

/end MODULE

```

#### Parameters:

ident Name:	ASAP device identifier
string LongIdentifier:	comment, description

#### Optional parameters:

-> A2ML:	<p>Format description of the interface-specific parameters.  <u>Attention:</u> The interface-specific parameters must be specified directly after the last mandatory parameter 'long identifier'.</p> <pre> /begin MODULE ENGINE_ECU "Comment"   /begin A2ML   /end A2ML   : /end MODULE </pre>
-> 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 meta-language (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
- > COMPU\_VTAB\_RANGE: Keyword for the description of range-based verbal conversion tables
- > FUNCTION: Keyword for the description of the functions
- > GROUP: Keyword for the declaration of groups
- > FRAME: Keyword for the declaration of frames
- > RECORD\_LAYOUT: Keyword for the description of the record layouts
- > VARIANT\_CODING: Keyword to describe the variant coding of adjustable objects.
- > USER\_RIGHTS : Keyword to reference the groups which constitute access rights.
- > UNIT: Keyword for the description of the measurement units

**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.

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

**Example:**

see part C: file engine\_ecu.a2l, abs\_ecu.a2l

### 6.3.90 MOD\_COMMON

#### Prototype:

```

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

/end MOD_COMMON

```

#### Parameters:

string Comment:        comment, description

#### 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 for the whole ASAP device. If this optional parameter is not declared, MSB\_LAST (Intel format) is used as a default.
- > DATA\_SIZE:     Data size in bits
- > ALIGNMENT\_BYTE: Declares the alignment of bytes in the complete module. The alignment is 1 if parameter is missing.
- > ALIGNMENT\_WORD: Declares the alignment of words in the complete module. The alignment is 2 if parameter is missing.
- > ALIGNMENT\_LONG: Declares the alignment of longs in the complete module. The alignment is 4 if parameter is missing.
- > ALIGNMENT\_FLOAT32\_IEEE: Declares the alignment of 32 bit floats in the complete module. The alignment is 4 if parameter is missing.
- > ALIGNMENT\_FLOAT64\_IEEE: Declares the alignment of 64 bit floats in the complete module. The alignment is 4 if parameter is missing.

#### 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         MSB_LAST
    DATA_SIZE         16
    ALIGNMENT_BYTE     2
/end MOD_COMMON
```

### 6.3.91 MOD\_PAR

#### Prototype:

```

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

/end MOD_PAR

```

#### Parameters:

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

#### Optional parameters:

- > CALIBRATION\_METHOD: Declares the implemented calibration methods in the control unit.
- > MEMORY\_SEGMENT: Declares the available memory segments.
- > 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
- > ECU\_CALIBRATION\_OFFSET: Offset that has to be added to each address of a characteristic.
- > 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                                "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
/begin MEMORY_SEGMENT ext_Ram "external RAM"
  DATA RAM EXTERN 0x30000 0x1000 -1 -1 -1 -1 -1
  /begin IF_DATA ASAP1B_KWP2000
    /* ADDRESS_MAPPING    orig_addr    mapping_addr    length */
    ADDRESS_MAPPING      0x4000      0x6000      0x0200
  /end IF_DATA
/end MEMORY_SEGMENT
/begin MEMORY_LAYOUT PRG_RESERVED
  0x0000 0x0400 -1 -1 -1 -1 -1
/end MEMORY_LAYOUT
/begin MEMORY_LAYOUT PRG_CODE
  0x0400 0x3C00 -1 -1 -1 -1 -1
/end MEMORY_LAYOUT
/begin MEMORY_LAYOUT PRG_DATA
  0x4000 0x5800 -1 -1 -1 -1 -1
/end MEMORY_LAYOUT
SYSTEM_CONSTANT "CONTROLLERx constant1" "0.33"
SYSTEM_CONSTANT "CONTROLLERx constant2" "2.79"
/end MOD_PAR

```

### 6.3.92 MONOTONY

**Prototype:**

MONOTONY                      Monotony

**Parameters:**

enum Monotony:	Description of the monotony:
MON_INCREASE:	monotonously increasing
MON_DECREASE:	monotonously decreasing
STRICT_INCREASE:	strict monotonously increasing
STRICT_DECREASE:	strict 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.

**Note:**

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

**Example:**

MONOTONY                      MON\_INCREASE



### 6.3.93 MULTIPLEX

**Prototype:**

MULTIPLEX                      Startbit DataSize Tag

**Parameters:**

int Startbit:	The 'mode-signal' is included in CAN-message at offset 'Startbit' (in bits, index of first bit of message data is 0).
int DataSize:	Size of 'mode-signal'.
long Tag:	The corresponding measurement data are valid only if the value of 'mode-signal' included in CAN-message agrees with the value specified with 'Tag'.

**Description:**

In 'Standard-Mode' a fixed data segment is assigned to each data object of CAN-message. 'Multiplexing' is a special method to transmit different data objects using the same CAN-message data segment. For that purpose an additional 'mode-signal' is used indicating the data object contained in multiplexed data segment.

**Example:**

```
/begin IF_DATA ASAP1B_CAN
  /begin KP_BLOB      „MOTOINFO“ 0x123 8 „MOTOR-SG“ 5 8
                      MULTIPLEX 48 16 0x4321
  /end KP_BLOB
/end IF_DATA
```

This example describes the measurement object „MOTO-INFO“ contained in data segment bit5...bit12 of CAN-message transmitted from „MOTOR-SG“. This data segment is a multiplexed segment. That means, that the measurement object „MOTO-INFO“ is contained in data segment bit5...bit12 only if the 'mode-signal' (data segment bit48...bit63) agrees to the value 0x4321.

### 6.3.94 NO\_AXIS\_PTS\_X/\_Y

#### Prototype:

NO\_AXIS\_PTS\_X/\_Y/\_Z      Position Datatype

#### Parameters:

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

#### Description:

Description of the number of axis points in an adjustable object

#### Example:

NO\_AXIS\_PTS\_X      2 UWORD

### 6.3.95 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

### 6.3.96 NO\_RESCALE\_X/\_Y/\_Z

#### Prototype:

NO\_RESCALE\_X/\_Y/\_Z      Position Datatype

#### Parameters:

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

#### Description:

Actual number of rescaling axis point value pairs. Used in RECORD\_LAYOUT.

#### Example:

NO\_RESCALE\_X      1 UBYTE

**6.3.97 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.

**Note:** The use of this keyword should be replaced by MATRIX\_DIM.

**Example:**

NUMBER                      7

### 6.3.98 OFFSET\_X/\_Y/\_Z

#### Prototype:

OFFSET\_X/\_Y/\_Z                      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 Datatype:	Data type of the 'offset' parameter.

#### Description:

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

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

or

$$Y_k = \text{Offset} + (k - 1) * 2^{\text{Shift}} \quad k = \{ 1 \dots \text{numberofaxispts} \}$$

or

$$Z_m = \text{Offset} + (m - 1) * 2^{\text{Shift}} \quad m = \{ 1 \dots \text{numberofaxispts} \}$$

#### Example:

OFFSET\_X                              16 UWORD

### 6.3.99 OUT\_MEASUREMENT

#### Prototype:

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

#### Parameters:

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

#### Description:

This keyword can be used to define output quantities of respective function.

**Note:** OUT\_MEASUREMENT may only refer to objects of type MEASUREMENT.

#### Example:

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

### 6.3.100 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"



### 6.3.101 PROJECT

**Prototype:**

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

/end PROJECT
```

**Parameters:**

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

**Optional parameters:**

-> HEADER:	Project header
-> MODULE:	This keyword is used to describe the 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"
                    "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
```

### 6.3.102 PROJECT\_NO

#### Prototype:

PROJECT\_NO                      ProjectNumber

#### 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

### 6.3.103 RASTER

#### Prototype:

```
/begin RASTER          RasterName ShortName RasterID ScalingUnit Rate
/end RASTER
```

#### Parameters:

string RasterName:	Event channel name (name for one sample rate of an ECU supported data acquisition mechanism, i.e. the ASAP1a-CCP)
string ShortName:	Short display name of the event channel name. <u>Recommendation</u> : The string length shall not exceed 9 characters.
int RasterID:	Event channel No., e.g. for ASAP1a-CCP START_STOP. This parameter can be used as a reference, therefore the number must be unique among all declared RASTER.
int ScalingUnit:	Period definition : basic scaling unit (see Table 4 : Codes for scaling unit)
long Rate:	ECU sample rate of the event channel, period definition based on the scaling unit

#### Description:

Applied for ECU supported data acquisition mechanisms, i.e. the DAQ lists of the ASAP1a-CCP (CAN Calibration Protocol) or the monitoring copy routines for the microcontroller internal RAM memory when using a memory probe. Such mechanisms typically provide „services“ based on a sample rate (periodic event) or a cyclic event.

Each available „service“ can be described by one RASTER (data acquisition event supported by the ECU) within IF\_DATA(Module) and then can be referenced within the ASAP1b-QP\_BLOB statement of the SOURCE keyword.

**Note:** This keyword is only used in the context of the IF\_DATA(MODULE) applied for the ASAP1a-CCP.

#### Example:

```
/begin RASTER          "Segment synchronous cylinder" "CYL1" 1 103 1
/end RASTER
```

### 6.3.104 READ\_ONLY

**Prototype:**

READ\_ONLY

**Description:**

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

**Example:**

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

                           /* interface-specific parameters: address location, addressing */
                           /begin IF_DATA "DIM"  EXTERNAL DIRECT /end IF_DATA

                           /begin FUNCTION_LIST  V_LIM          /* Reference to functions */
                           /end FUNCTION_LIST
                           READ_ONLY
/end CHARACTERISTIC
```

### 6.3.105 READ\_WRITE

**Prototype:**

READ\_WRITE

**Description:**

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

**Example:**

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

### 6.3.106 RECORD\_LAYOUT

#### Prototype:

```

/begin RECORD_LAYOUT      Name
                           [-> FNC_VALUES]
                           [-> IDENTIFICATION]
                           [-> AXIS_PTS_X]
                           [-> AXIS_PTS_Y]
                           [-> AXIS_PTS_Z]
                           [-> AXIS_RESCALE_X]
                           [-> AXIS_RESCALE_Y]
                           [-> AXIS_RESCALE_Z]
                           [-> NO_AXIS_PTS_X]
                           [-> NO_AXIS_PTS_Y]
                           [-> NO_AXIS_PTS_Z]
                           [-> NO_RESCALE_X]
                           [-> NO_RESCALE_Y]
                           [-> NO_RESCALE_Z]
                           [-> FIX_NO_AXIS_PTS_X]
                           [-> FIX_NO_AXIS_PTS_Y]
                           [-> FIX_NO_AXIS_PTS_Y]
                           [-> SRC_ADDR_X]
                           [-> SRC_ADDR_Y]
                           [-> SRC_ADDR_Z]
                           [-> RIP_ADDR_X]
                           [-> RIP_ADDR_Y]
                           [-> RIP_ADDR_Z]
                           [-> RIP_ADDR_W]
                           [-> SHIFT_OP_X]
                           [-> SHIFT_OP_Y]
                           [-> SHIFT_OP_Z]
                           [-> OFFSET_X]
                           [-> OFFSET_Y]
                           [-> OFFSET_Z]
                           [-> DIST_OP_X]
                           [-> DIST_OP_Y]
                           [-> DIST_OP_Z]
                           [->ALIGNMENT_BYTE]
                           [->ALIGNMENT_WORD]
                           [->ALIGNMENT_LONG]
                           [->ALIGNMENT_FLOAT32_IEEE]
                           [->ALIGNMENT_FLOAT64_IEEE]
                           {-> RESERVED}*

/end RECORD_LAYOUT

```

#### Parameters:

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

**Note:**

The name of the record layout has to be unique within all record layouts of the ASAP2 MODULE, i.e. there must not be another RECORD\_LAYOUT object with the the same identifier in the MODULE.

**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.

-> AXIS\_PTS\_Z: This keyword describes where the Z-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.

-> NO\_AXIS\_PTS\_Z: This keyword describes in which position the parameter 'number of Z-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 (!)'.  
 -> FIX\_NO\_AXIS\_PTS\_Z: This keyword indicates that all characteristic curves or characteristic maps relating to the Z-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\_Z (!)'.  
 -> 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.

- > SRC\_ADDR\_Z: This keyword describes in which position the address of the input quantity of the Z-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 Interpolation), which describes the relative distance between the current value and the adjacent axis points:  

$$\text{RIP\_X} = (X(t) - X_i) / (X_{i+1} - X_i).$$
- > RIP\_ADDR\_Y/\_Z: 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. See RIP\_ADDR\_Y, but for Y/Z-axis points.
- > RIP\_ADDR\_W: Final result (table value) of the ECU-internal interpolation.
- > 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).
- > AXIS\_RESCALE\_X: This keyword describes where the rescale mapping for the x-axis is deposited in memory.
- > AXIS\_RESCALE\_Y: This keyword describes where the rescale mapping for the y-axis is deposited in memory.
- > AXIS\_RESCALE\_Z: This keyword describes where the rescale mapping for the z-axis is deposited in memory.
- > NO\_RESCALE\_X: This keyword describes at which position the parameter 'actual number of rescale pairs' for the x-axis is deposited (see AXIS\_RESCALE).
- > NO\_RESCALE\_Y: This keyword describes at which position the parameter 'actual number of rescale pairs' for the y-axis is deposited (see AXIS\_RESCALE).
- > NO\_RESCALE\_Z: This keyword describes at which position the parameter 'actual number of rescale pairs' for the z-axis is deposited (see AXIS\_RESCALE).
- > ALIGNMENT\_BYTE: Declares the alignment of bytes for all characteristics who use this record layout.. The alignment is 1 if parameter is missing.
- > ALIGNMENT\_WORD: Declares the alignment of words for all characteristics who use this record layout. The alignment is 2 if parameter is missing.
- > ALIGNMENT\_LONG: Declares the alignment of longs for all characteristics who use this record layout. The alignment is 4 if parameter is missing.
- > ALIGNMENT\_FLOAT32\_IEEE: Declares the alignment of 32 bit floats for all characteristics who use this record layout. The alignment is 4 if parameter is missing.



-> ALIGNMENT\_FLOAT64\_IEEE: Declares the alignment of 64 bit floats for all characteristics who use this record layout. The alignment is 4 if parameter is missing.

Only for fixed curves or fixed maps:

-> SHIFT\_OP\_X: Shift operand to compute the X-axis points  
-> SHIFT\_OP\_Y: Shift operand to compute the Y-axis points  
-> SHIFT\_OP\_Z: Shift operand to compute the Z-axis points  
-> OFFSET\_X: Offset to compute the X-axis points  
-> OFFSET\_Y: Offset to compute the Y-axis points  
-> OFFSET\_Z: Offset to compute the Z-axis points  
-> DIST\_OP\_X: 'Distance' parameter to compute the X-axis points  
-> DIST\_OP\_Y: 'Distance' parameter to compute the Y-axis points  
-> DIST\_OP\_Z: 'Distance' parameter to compute the Z-axis points

### 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.).

In particular, if the ALTERNATE option is used with FNC\_VALUES, the position parameter determines the order of values and axis points.

### Important:

To describe the record layouts, use is made of a predefined list of parameters which may be part of an adjustable object (characteristic) in the emulation memory. This list represents the current status of the record layouts. With each change or extension of the record layouts contained in this predefined list of parameters the 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 DIRECT
  AXIS_PTS_Y          6 UBYTE INDEX_INCR DIRECT
  NO_AXIS_X           2 UBYTE
  NO_AXIS_Y           5 UBYTE
  SRC_ADDR_X           1
  SRC_ADDR_Y           4
  ALIGNMENT_BYTE      2
/end RECORD_LAYOUT

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

```

/end RECORD_LAYOUT

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

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

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

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

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

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

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

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

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

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

```

/begin RECORD_LAYOUT	3D_structure_table_int
NO_AXIS_PTS_X	1 UWORD
NO_AXIS_PTS_Y	2 UWORD
FNC_VALUES	3 SWORD ROW_DIR DIRECT
/end RECORD_LAYOUT	
/begin RECORD_LAYOUT	3D_structure_table_word
NO_AXIS_PTS_X	1 UWORD
NO_AXIS_PTS_Y	2 UWORD
FNC_VALUES	3 UWORD ROW_DIR DIRECT
/end RECORD_LAYOUT	
/begin RECORD_LAYOUT	3D_structure_table_byte
NO_AXIS_PTS_X	1 UBYTE
NO_AXIS_PTS_Y	2 UBYTE
RESERVED	3 BYTE
FNC_VALUES	4 UBYTE ROW_DIR DIRECT
/end RECORD_LAYOUT	
/begin RECORD_LAYOUT	3D_structure_table_shortint
NO_AXIS_PTS_X	1 UBYTE
NO_AXIS_PTS_X	2 UBYTE
RESERVED	3 BYTE
FNC_VALUES	4 SBYTE ROW_DIR DIRECT
/end RECORD_LAYOUT	
/begin RECORD_LAYOUT	2D_array_table_int
FNC_VALUES	1 SWORD ROW_DIR DIRECT
/end RECORD_LAYOUT	
/begin RECORD_LAYOUT	2D_array_table_word
FNC_VALUES	1 UWORD ROW_DIR DIRECT
/end RECORD_LAYOUT	
/begin RECORD_LAYOUT	2D_array_table_byte
FNC_VALUES	1 UBYTE ROW_DIR DIRECT
/end RECORD_LAYOUT	
/begin RECORD_LAYOUT	2D_array_table_shortint
FNC_VALUES	1 SBYTE ROW_DIR DIRECT
/end RECORD_LAYOUT	
/begin RECORD_LAYOUT	3D_array_table_int
FNC_VALUES	1 SWORD ROW_DIR DIRECT
/end RECORD_LAYOUT	
/begin RECORD_LAYOUT	3D_array_table_word
FNC_VALUES	1 UWORD ROW_DIR DIRECT
/end RECORD_LAYOUT	

---

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

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

### 6.3.107 REF\_CHARACTERISTIC

**Prototype:**

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

**Parameters:**

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

**Description:**

This keyword can be used to define some adjustable objects that are referenced in respective function or group.

**Note:** REF\_CHARACTERISTIC may only refer to objects of type CHARACTERISTIC or AXIS\_PTS.

**Example:**

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

### 6.3.108 REF\_GROUP

#### Prototype:

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

#### Parameters:

ident Identifier:            Identifier of those groups which are referred in  
USER\_RIGHTS

#### Description:

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

#### Example:

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

### 6.3.109 REF\_MEASUREMENT

#### Prototype:

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

#### Parameters:

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

#### Description:

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

#### Example:

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

### 6.3.110 REF\_MEMORY\_SEGMENT

#### Prototype:

REF\_MEMORY\_SEGMENT    Name

#### Parameters:

ident Name:                      Name of memory segments

#### Description:

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

#### Example:

REF\_MEMORY\_SEGMENT Data1



### 6.3.111 REF\_UNIT

#### Prototype:

REF\_UNIT                      Unit

#### Parameters:

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

#### Description:

This keyword can be used to reference to the data record which describes a measurement unit. It can be used in COMPU\_METHOD and UNIT.

REF\_UNIT may only refer to objects of type UNIT.

**Note:** The string parameter Unit of a COMPU\_METHOD is a redundant information because the record referenced by REF\_UNIT contain it too. Just for the purpose of compatibility with previous versions of ASAP2 the parameter REF\_UNIT is optional. Therefore the string parameter Unit is going to be canceled at ASAP2 version 2.00.

#### Example:

```

/begin COMPU_METHOD
  Velocity
  "conversion method for velocity"
  RAT_FUNC
  "%6.2"
  "[km/h]"          /* to be canceled at ASAP2 version 2.00 */
  COEFFS 0 100 0 0 0 1
  REF_UNIT kms_per_hour /* new (optional) parameter */
/end COMPU_METHOD

```

### 6.3.112 RESERVED

**Prototype:**

RESERVED                      Position DataSize

**Parameters:**

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

**Description:**

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

**Example:**

RESERVED                      7 LONG

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

### 6.3.113 RIGHT\_SHIFT

#### Prototype:

RIGHT\_SHIFT                      Bitcount

#### Parameters:

long bitcount:                  Shift ,bitcount‘ bits to the right

#### Description:

The RIGHT\_SHIFT keyword is only used within the BIT\_OPERATION keyword.  
See description of BIT\_OPERATION.

### 6.3.114 RIP\_ADDR\_X/\_Y/\_Z/\_W

#### Prototype:

RIP\_ADDR\_X/\_Y/\_Z/\_W      Position Datatype

#### Parameters:

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

Remark: Relating to version 1.0 of ASAP2-Specification this is an additional parameter.

#### 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:

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

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

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

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

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

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

for a characteristic map as follows:

$$\begin{aligned} \text{RIP\_W} = & (W_{i,k} * (1 - \text{RIP\_X}) + W_{i+1,k} * \text{RIP\_X}) * (1 - \text{RIP\_Y}) + \\ & (W_{i,k+1} * (1 - \text{RIP\_X}) + W_{i+1,k+1} * \text{RIP\_X}) * \text{RIP\_Y} \end{aligned}$$

and for a characteristic cuboid as follows:

Interpolation for the map  $Z = m$

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

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

Interpolation for the map  $Z = m+1$

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

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

Interpolation in Z direction between the two points  $\text{RIP\_W}_m$  and  $\text{RIP\_W}_{m+1}$ .

$$\text{RIP\_W} = \text{RIP\_W}_m + (\text{RIP\_Z} * (\text{RIP\_W}_{m+1} - \text{RIP\_W}_m))$$

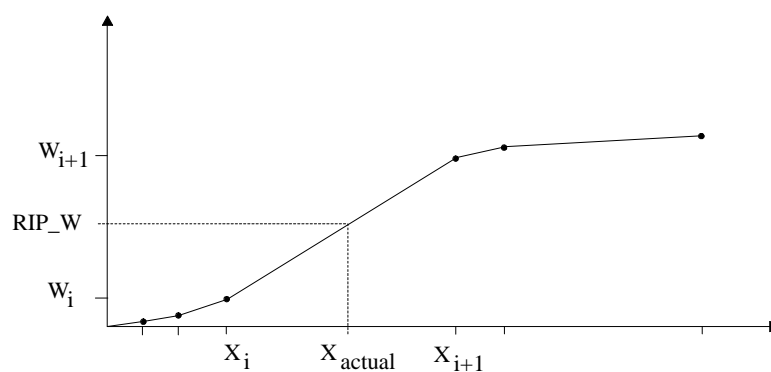


Figure 9: Linear interpolation for a characteristic curve

**Example:**

RIP\_ADDR\_X

19 UWORD

### 6.3.115 ROOT

**Prototype:**

ROOT

**Parameters:**

none

**Description:**

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

**Example:**

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

### 6.3.116 SEED\_KEY

**Prototype:**

```
/begin SEED_KEY          CalDll DaqDll PgmDll
/end SEED_KEY
```

**Parameters:**

string CalDll:	Reference to DLL file name for CAL (ECU Calibration) access privilege.
string DaqDll:	Reference to DLL file name for DAQ (ECU Data Acquisition) access privilege.
string PgmDll:	Reference to DLL file name for PGM (ECU Flash Programming) access privilege.

**Description:**

Description of the authentication process (standardized interface to the confidential seed-key algorithms for serial ECU access). References to dynamic link libraries (DLL) with a standard application interface (API), contain the authentication algorithms. The ECU supplier provides the DLLs, the MCD systems reads them to enable calibration, data acquisition or reprogramming access to the ECU.

**Note:** This keyword is only used in the context of the IF\_DATA(MODULE) applied for the ASAP1a-CCP.

**Note:** The usage of the DLL is based on a standard API definition published in chapter 8.7.

**Example:**

```
/begin SEED_KEY          "access.dll" "access.dll" "access.dll"
/end SEED_KEY
```



### 6.3.117 SIGN\_EXTEND

**Prototype:**

SIGN\_EXTEND

**Parameters:**

**Description:**

The SIGN\_EXTEND keyword is only used within the BIT\_OPERATION keyword.  
See description of BIT\_OPERATION.

### 6.3.118 SHIFT\_OP\_X/\_Y/\_Z

#### Prototype:

SHIFT\_OP\_X/\_Y/\_Z                      Position Datatype

#### Parameters:

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

#### Description:

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

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

or

$$Y_k = \text{Offset} + (k - 1) * 2^{\text{Shift}} \quad k = \{ 1 \dots \text{numberofaxispts} \}$$

or

$$Z_m = \text{Offset} + (m - 1) * 2^{\text{Shift}} \quad m = \{ 1 \dots \text{numberofaxispts} \}$$

#### Example:

SHIFT\_OP\_X                      21 UWORD

### 6.3.119 SI\_EXPONENTS

#### Prototype:

SI\_EXPONENTS                      Length Mass Time ElectricCurrent Temperature AmountOfSubstance LuminousIntensity

#### Parameters:

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

#### Description:

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

#### Example:

```

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

```

### 6.3.120 SOURCE

#### Prototype:

```

/begin SOURCE          Name ScalingUnit Rate
                        [-> DISPLAY_IDENTIFIER]
                        [-> QP_BLOB QP_Data]

/end SOURCE

```

#### Parameters:

ident Name:	Identifier of measurement source (data acquisition mode).
int ScalingUnit:	This parameter defines the basic scaling unit. The following parameter 'Rate' relates on this scaling unit. The value of ScalingUnit is coded as shown in 'Table 4: Codes for scaling units (CSE)' (page 144).
long Rate:	The maximum refresh rate of the concerning measurement source in the control unit. The unit is defined with parameter 'ScalingUnit'.

#### Optional parameters:

-> DISPLAY_IDENTIFIER:	Display name for the data acquisition channel of the SOURCE.
-> QP_BLOB:	Interface-specific description data (BLOB: binary large object) used at ASAP1b device at call of the command InitRead(). The parameters associated with this keyword are described in the ASAP2 metalanguage (in short A2ML) by the ECU supplier or the interface module supplier.

#### Description:

This keyword can be used to describe different 'sources' of measurement, e.g. different data acquisition modes (e.g. time synchronous 10 msec, time synchronous 50 msec etc.).

#### Example:

```

/* refresh rate of acquisition mode 'Time35' is 35 milli-seconds, */
/* (value of ScalingUnit = 3 corresponds to 1 msec) */
/begin SOURCE          Time35  3  35
/end SOURCE

```

**6.3.121 SRC\_ADDR\_X/\_Y/\_Z****Prototype:**

SRC\_ADDR\_X/\_Y/\_Z            Position Datatype

**Parameters:**

int Position:	Position of the address of the input quantity in the deposit structure.
datatype Datatype:	Data type of the address. <u>Remark:</u> Relating to version 1.0 of ASAP2-Specification this is an additional parameter. The appropriate parameter in the corresponding measuring channel data record is not any longer used.

**Description:**

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

**Example:**

SRC\_ADDR\_X            1 UWORD

### 6.3.122 SUB\_FUNCTION

#### Prototype:

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

#### Parameters:

ident Identifier:	Reference to function record. This function record is declared as subfunction of respectiv function.
-------------------	--

#### Description:

This keyword can be used to define the hierarchical structur of functions.

**Note:** SUB\_FUNCTION may only refer to objects of type FUNCTION.

#### Example:

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

### 6.3.123 SUB\_GROUP

**Prototype:**

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

**Parameters:**

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

**Description:**

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

**Example:**

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

### 6.3.124 SUPPLIER

**Prototype:**

SUPPLIER                      Manufacturer

**Parameters:**

    string Manufacturer:      Name of the ECU manufacturer

**Description:**

    String used to identify the manufacturer or supplier.

**Example:**

SUPPLIER                      "Smooth and Easy"



**6.3.125 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"

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

**Note:** S\_REC\_LAYOUT may only refer to objects of type RECORD\_LAYOUT.

**Example:**

S\_REC\_LAYOUT                      SIEMENS\_ABL                      /\* Siemens record layout \*/

### 6.3.127 UNIT

#### Prototype:

```

/begin UNIT                                     Name LongIdentifier Display Type
                                                [-> SI_EXPONENTS]
                                                [-> REF_UNIT]
                                                [-> UNIT_CONVERSION]
/end UNIT

```

#### Parameters:

ident Name: identifier in the program, referencing is based on this name  
**Note:**  
The name of the unit has to be unique within all units of the ASAP2 MODULE, i.e. there must not be another UNIT object with the the same identifier in the MODULE.

string LongIdentifier: comment, description  
string Display: string to be used to display the measurement unit of a physical value

enum Type: Type of the Unit:  
EXTENDED\_SI: extended SI unit.  
Recommendation: The principle of describing "real" measurement units is to refer to SI units. Therefore, this relationship should be given by using the optional parameter SI\_EXPONENTS.  
DERIVED: measurement unit derived from another one referenced by the optional parameter REF\_UNIT.

#### Optional Parameters:

-> SI\_EXPONENTS: This keyword is used to specify the exponents of the seven base dimensions required to define an extended SI unit.  
-> REF\_UNIT: This keyword is used to reference to another measurement unit from which the one using REF\_UNIT is derived.  
-> UNIT\_CONVERSION: This keyword is used to specify the linear relationship between two measurement units.

#### Description:

Specification of a measurement unit.

#### Example:

```

/begin UNIT
metres_per_second
"extended SI unit for velocity"

```

```

"[m/s]"
EXTENDED_SI
SI_EXPONENTS 1 0 -1 0 0 0 0          /* [m] * [s]-1 */
/end UNIT

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

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

```

### 6.3.128 UNIT\_CONVERSION

**Prototype:**

UNIT\_CONVERSION            Gradient Offset

**Parameters:**

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

**Description:**

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

$$\text{derived\_unit} = f(\text{referenced\_unit})$$

The referenced measurement unit had to be specified with parameter REF\_UNIT.

**Example:**

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

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

### 6.3.129 USER

#### Prototype:

USER                                  UserName

#### Parameters:

        string UserName:          Name of the user

#### Description:

        Specification of the user name.

#### Example:

USER                                  "Nigel Hurst"

### 6.3.130 USER\_RIGHTS

#### Prototype:

```

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

/end USER_RIGHTS

```

#### Parameters:

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

#### Optional parameters:

-> REF\_GROUP: Reference to groups.  
Only the CHARACTERISTIC and MEASUREMENT members of the referenced groups including the members of nested subgroups (and functions nested in such groups) are visible to the user of the MCD system. If the READ\_ONLY attribute is set, the CHARACTERISTICS are visible but not available for calibration (not tuneable).

*The restrictions are applied by the MCD system as a global filter in the user interface (active for all manual selection or calibration operations). When navigating by GROUPs, only the GROUPs declared in USER\_RIGHTS need to be provided in the selection list.*

-> READ\_ONLY: This keyword can be used to define all characteristics of the groups referenced by this USER\_RIGHT statement as READ\_ONLY (not tunable).

Use Case : A group can be defined to specify a set of characteristics to be not tunable for a group of users (control of access rights). In order to achieve this, the group is referenced in a USER\_RIGHT statement with the READ\_ONLY attribute, related to the user group.

When a login to the MCD system identifies the user as member of a group for which the USER\_RIGHT statement contains the READ\_ONLY attribute, all CHARACTERISTICS of this group shall be treated as if the READ\_ONLY attribute was directly related to the CHARACTERISTICS.

#### Description:

This keyword can be used to define groups accessible for certain users. All USER\_RIGHTS groups are listed to the user who can select one of these groups. All measurements and characteristics belonging to that group and its subgroups (and subsubgroups and so on) are accessible (i.e. visible) to the user. The keyword

READ\_ONLY is used to define the referred group(s) as containing characteristics that are only readable but not writable (i.e. they can not be adjusted). This property is also inherited by subgroups, i.e. if a group is marked as READ\_ONLY all its subgroups (with respect to that USER RIGHT) are also only READ\_ONLY.

**Example:**

```
/begin USER_RIGHTS      application_engineers
  /begin REF_GROUP group_1 /end REF_GROUP
/end USER_RIGHTS

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

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



### 6.3.131 VARIANT\_CODING

#### Prototype:

```

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

```

#### Optional Parameters:

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

#### Description:

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

**Example:**

```

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

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

  /begin VAR_FORBIDDEN_COMB      /* forbidden: Limousine - Manual */
                                Car Limousine
                                Gear Manual
  /end VAR_FORBIDDEN_COMB
  /begin VAR_FORBIDDEN_COMB      /* forbidden: Cabrio - Automatic */
                                Car Cabrio
                                Gear Automatic
  /end VAR_FORBIDDEN_COMB
  /begin VAR_CHARACTERISTIC PUMKF /* define PUMKF as variant coded */
                                Gear      /* Gear box variants */
                                /begin VAR_ADDRESS
                                    0x7140 0x7168
                                /end VAR_ADDRESS
  /end VAR_CHARACTERISTIC
  /begin VAR_CHARACTERISTIC NLLM /* define NLLM as variant coded */
                                Gear Car /* car body and gear box variants */
                                /begin VAR_ADDRESS
                                    0x8840 0x8858 0x8870 0x 8888
                                /end VAR_ADDRESS
  /end VAR_CHARACTERISTIC
/end VARIANT_CODING

```

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

<b>Type of gear box</b>	
<b>MANUAL</b>	<b>AUTOMATIC</b>
PUMKF.0 (address = 0x7140)	PUMKF.1 (address = 0x7168)

Table 5: Example of NLLM and PUMKF - variants coding

### 6.3.132 VAR\_ADDRESS

**Prototype:**

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

**Parameters:**

long Address:            Start address of one variant of variant coded adjustable object.

**Description:**

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

- ? the priority of index increment is invers to the order of variant criteria definition at keyword VAR\_CHARACTERISTIC, e.g.:
  - ? the first variant criterion has the lowest priority
  - ? the last variant criterion has the highest priority

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

Example:

Crit1 = { Val<sub>1,1</sub>, Val<sub>1,2</sub>, ... Val<sub>1,L</sub> }

Crit2 = { Val<sub>2,1</sub>, Val<sub>2,2</sub>, ... Val<sub>2,M</sub> }

Crit3 = { Val<sub>3,1</sub>, Val<sub>3,2</sub>, ... Val<sub>3,N</sub> }

Corresponding address list:

Address[0]            = Address (Val<sub>1,1</sub>, Val<sub>2,1</sub>, Val<sub>3,1</sub>)

Address[1]            = Address (Val<sub>1,1</sub>, Val<sub>2,1</sub>, Val<sub>3,2</sub>)

:

Address[N - 1]        = Address (Val<sub>1,1</sub>, Val<sub>2,1</sub>, Val<sub>3,N</sub>)

Address[N]            = Address (Val<sub>1,1</sub>, Val<sub>2,2</sub>, Val<sub>3,1</sub>)

Address[N + 1]        = Address (Val<sub>1,1</sub>, Val<sub>2,2</sub>, Val<sub>3,2</sub>)

:

Address[N + N - 1]    = Address (Val<sub>1,1</sub>, Val<sub>2,2</sub>, Val<sub>3,N</sub>)

:

**Example:**

```
/begin VAR_ADDRESS  
0x8840 0x8858 0x8870 0x 8888            /* see example, page 211 */  
/end VAR_ADDRESS
```

### 6.3.133 VAR\_CHARACTERISTIC

**Prototype:**

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

**Parameters:**

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

**Optional Parameters:**

->VAR\_ADDRESS: Definition of start address of adjustable objects variants.

**Description:**

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

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

**Example:**

```
/begin VAR_CHARACTERISTIC      /* define NLLM as variant coded */  
  NLLM  
  Gear Car  
  /* gear box including the 2 variants "Manual" and "Automatic" */  
  /* car body including the 3 variants "Limousine", "Kombi" and "Cabrio" */  
  
  /* four addresses corresponding to the four valid combinations */  
  /* of criterion 'Gear' and 'Car' (see example, page 211) */  
  /begin VAR_ADDRESS  0x8840 0x8858 0x8870 0x 8888  
  /end VAR_ADDRESS  
/end VAR_CHARACTERISTIC
```

### 6.3.134 VAR\_CRITERION

#### Prototype:

```

/begin VAR_CRITERION      Name LongIdentifier { Value }*
                           [-> VAR_MEASUREMENT]
                           [-> VAR_SELECTION_CHARACTERISTIC]

/end VAR_CRITERION

```

#### Parameters:

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

#### Optional Parameters:

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

#### Description:

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

#### Example:

```

/* variant criterion "Car body" with three variants */
/begin VAR_CRITERION      Car "Car body"

                           /* Enumeration of criterion values */
                           Limousine Kombi Cabrio

      VAR_MEASUREMENT      S_CAR
      VAR_SELECTION_CHARACTERISTIC  V_CAR
/end VAR_CRITERION

```

### 6.3.135 VAR\_FORBIDDEN\_COMB

**Prototype:**

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

**Parameters:**

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

**Description:**

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

**Example:**

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

### 6.3.136 VAR\_SELECTION\_CHARACTERISTIC

**Prototype:**

VAR\_SELECTION\_CHARACTERISTIC                      Name

**Parameters:**

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

**Description:**

This keyword can be used to specify a special characteristic object. This characteristic object changes with its current value the variant which has effect on running control unit software. Referenced VAR\_SELECTION\_CHARACTERISTIC could have a 'verbal conversion table' and the strings defined at COMPU\_VTAB could correspond to criterion values at VAR\_CRITERION record.

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

**Note:** VAR\_SELECTION\_CHARACTERISTIC may only refer to objects of type CHARACTERISTIC.

**Example:**

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



### 6.3.137 VAR\_MEASUREMENT

**Prototype:**

VAR\_MEASUREMENT      Name

**Parameters:**

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

**Description:**

This keyword can be used to specify a special measurement object. This measurement object indicates with its current value the variant which has effect on running control unit software. The value 0 (zero) of measurement object corresponds to the first variant value defined at relative VAR\_CRITERION record (see parameter 'Value' at keyword VAR\_CRITERION), the value 1 to the second and so on.

Question:

Which value of measurement object should be used to get the effective variant?

- a) internal value acquired from control unit software or?
- b) physical value computed from internal value using COMPU\_METHOD record?
- c) Referenced COMPU\_METHOD record could be a 'verbal conversion table' and the strings defined at COMPU\_VTAB could correspond to criterion values at VAR\_CRITERION record?

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

**Note:** VAR\_MEASUREMENT may only refer to objects of type MEASUREMENT.

**Example:**

```
/begin VAR_CRITERION                      Car "Car body" Limousine Kombi Cabrio
  VAR_MEASUREMENT                      S_GEAR_BOX
/end VAR_CRITERION
/* S_GEAR_BOX = 0:                      indicates variant "Limousine" to be effectiv */
/* S_GEAR_BOX = 1:                      indicates variant "Kombi" to be effectiv */
/* S_GEAR_BOX = 2:                      indicates variant "Cabrio" to be effectiv */
```

### 6.3.138 VAR\_NAMING

**Prototype:**

VAR\_NAMING                      Tag

**Parameters:**

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

**Description:**

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

**Example:**

```
/* variant extension: see example page 211*/  
VAR_NAMING                      NUMERIC
```

### 6.3.139 VAR\_SEPARATOR

**Prototype:**

VAR\_SEPARATOR                      Separator

**Parameters:**

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

**Description:**

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

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

**Example:**

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

### 6.3.140 VERSION

#### Prototype:

VERSION                      VersionIdentifier

#### Parameters:

    string VersionIdentifier: short identifier for the version

#### Description:

    String for identification of the version with maximum MAX\_STRING (at present 255) characters.

#### Example:

VERSION                      "BG5.0815"

### 6.3.141 VIRTUAL

**Prototype:**

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

**Parameters:**

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

**Description:**

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

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

### 6.3.142 VIRTUAL\_CHARACTERISTIC

#### Prototype:

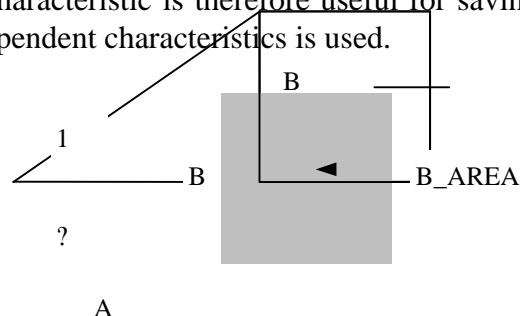
```
/begin VIRTUAL_CHARACTERISTIC          Formula (Characteristic)*
/end VIRTUAL_CHARACTERISTIC
```

#### Parameters:

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

#### Description:

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



For the initial value of the virtual characteristic must be derived from the values of other characteristics. The mechanism to implement this is the same as for dependent characteristics by a list of characteristics and a formula, e.g.  $? = \arcsin(B)$ . Also B might be virtual, i.e. its value has to be derived from B\_AREA.

The following example makes clear how the calibration process takes place. When the virtual characteristic ? is initialized, the value of ? is calculated from the value of B. Therefore  $B_{\text{hex}}$  is read from the ECU and  $B_{\text{phys}} = B_{\text{hex}}/100$  is computed. Assuming the value  $B_{\text{hex}} = 80$ ,  $B_{\text{phys}} = 0.8$  and  $?_{\text{phys}} = \arcsin(B_{\text{phys}}) = 53.13$ . Since virtual characteristics are not in the memory of an ECU,  $?_{\text{hex}}$  and  $?_{\text{phys}}$  may coincide if the datatype for  $?_{\text{hex}}$  is chosen an float datatype and the conversion formula is the identity (one to one formula).

Used in CHARACTERISTIC.

**Example:**

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

## 7 Include mechanism

### 7.1 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. The filename may be put between quotation marks. If the filename contains spaces the quotation marks are required.

#### Example:

```

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

  /include ENG_ECU.A2L
  /include ABS_ECU.A2L
  /include ?SPEC_ECU.A2L"
/end
_____ End of file PROJECT1.A2L _____

```

### 7.2 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

### 8 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 incorporating 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).

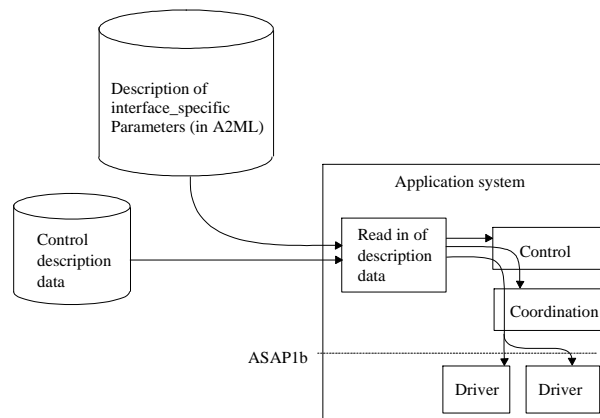


Figure 10: Schematic data flow of description data

## 8.1 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:

<tag>:                    <tag> 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 \*/

## 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" <tag> 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 tag.

For the description of lists with a variable number of elements, the symbols "(" and ")"\* are used. The sequences identified by these symbols can be repeated any number of times.

```

taggedstruct_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  ::= <tag> member |
                             <tag> "(" 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 tag for identification purposes (see <tag>).

```

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

taggedunion_member_list ::= tagged_union_member |
                             tagged_union_member taggedunion_member_list

taggedunion_member      ::= <tag> [ member ] ";" |
                             block_definition ";"

```

## 8.2 Designing A2ML-file

This chapter is describing how to design an A2ML-file for interface-specific data used together with a ASAP1b compatible driver.

To be compatible with ASAP2 V1.2 a tag "IF\_DATA" should be defined in the A2ML-file. This tag is then used by the MCD to interpret the data that is written in the various IF\_DATA-fields in the ASAP2-file. Figure 12 shows a template that preferably should be used to design a A2ML-file for interface-specific data.

The string "ASAP1B\_\*" in the beginning of Figure 12 is a tag which is used to separate different interfaces. This makes it possible to define different types of data for different interfaces. The \*-character in the template could be substituted with a string that describes the interface (e.g. ASAP1B\_RS232 for a serial interface).

The tag SOURCE is described by the ASAP2 specification but because it is located in the IF\_DATA it should be defined in the A2ML-file.

**Note:** The syntax of user-defined tags in AML must follow the rule of an 'ident'.

Example: "IF\_DATA 4EXAMPLE...." is not valid.

The different BLOB tags are used to define interface dependent data that should be put into BLOB-objects and sent to the ASAP1b-driver when different services are called. Figure 11

shows which BLOB-objects that is used by the various ASAP1b-services. The ASAP1b-services missing in the figure does not use any BLOB-objects.

<b>ASAP1b service</b>	<b>BLOB-objects used</b>
INIT_READ	TP_BLOB, QP_BLOB and KP_BLOB
INIT_ACCESS	TP_BLOB and DP_BLOB
ACCESS	PA_BLOB
COMMAND	TP_BLOB

Figure 11: Description of BLOB-objects used in ASAP1b-services

There are two interfaces already defined in the ASAP2 documentation, ASAP1B\_CAN and ASAP1B\_ADRESS, these interfaces define a minimum of information that could be used to reference measurement parameters inside an ECU respectively on a CAN-network. The A2ML-file for these interfaces is displayed in Figure 13.

```

/begin A2ML
/* template.aml *****/
/* By: Volvo Car Corporation, 961011 */
/* */
/* Template for designing IF_DATA fields for ASAP2 files and BLOB's */
/* for ASAP1b interface. */
/* *****/
block "IF_DATA" taggedunion if_data
{
    "ASAP1B_" /* The tag of ASAP1b specific information should */
              /* start with ASAP1B_ then the * character can be */
              /* substituted with a name of manufacturer's choice. */

    taggedstruct /* optional parameters */

        (block "SOURCE" struct
        {
            struct /* indispensable */
            {
                char [101]; /* source name (string)*/
                int; /* min period ( conforming together with min factor */
                    /* the fastest samplingrate available ). */
                long; /* min factor */
            };
            taggedstruct /* optional parameters */
            {
                block "QP_BLOB" struct /* QP_BLOB for ASAP1b */
                {
                    /* QP_BLOB specification */
                };
            };
        });
    ); /* multiple SOURCE may exist */

    block "TP_BLOB" struct /* TP_BLOB for ASAP1b */
    {
        /* TP_BLOB specification */
    };

    block "DP_BLOB" struct /* DP_BLOB for ASAP1b */
    {
        /* DP_BLOB specification */
    };

    block "PA_BLOB" struct /* PA_BLOB for ASAP1b */
    {
        /* PA_BLOB specification */
    };

    block "KP_BLOB" struct /* KP_BLOB for ASAP1b */
    {
        /* KP_BLOB specification */
    };

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

};

/* Extra tags can be defined here */

};

/*****/
/end A2ML

```

Figure 12: Template for A2ML-file

```

/begin A2ML
/* asap2std.aml ***** */
/* By: Volvo Car Corporation, 961217 */
/* */
/* A2ML-file defining the interfaces ASAP1B_ADDRESS and ASAP1B_CAN. */
/* ***** */
block "IF_DATA" taggedunion if_data
{
    "ASAP1B_ADDRESS" taggedstruct /* optional parameters */
    {
        (block "SOURCE" struct
        {
            struct /* indispensable */
            {
                char [101]; /* source name (string)*/
                int; /* min period ( conforming together with min factor the fastest */
                /* ...samplingrate available ). */
                long; /* min factor */
            };
        }
        ); /* multiple SOURCE may exist */

        block "KP_BLOB" struct /* KP_BLOB specification for ASAP1b */
        {
            long; /* address of measurement object (e.g. emulator RAM addressing) */
        };
    };

    "ASAP1B_CAN" taggedstruct /* optional parameters */
    {
        (block "SOURCE" struct
        {
            struct /* indispensable */
            {
                char [101]; /* source name (string)*/
                int; /* min period ( conforming together with min factor */
                /* the fastest samplingrate available ). */
                long; /* min factor */
            };
        }
        ); /* multiple SOURCE may exist */

        block "KP_BLOB" struct /* KP_BLOB specification for ASAP1b */
        {
            char[33]; /* messagename */
            long; /* identifier */
            int; /* messagesize */
            char[101]; /* messagesource */
            int; /* startbit */
            int; /* datasize */
            taggedstruct
            {
                "MULTIPLEX" struct
                {
                    int; /* startbit */
                    int; /* datasize */
                    long; /* tag */
                };
            };
        }; /* end of: block "KP_BLOB" */
    }; /* end of: "ASAP1B_CAN" taggedstruct */
}; /* end of: taggedunion if_data */

/***** */
/end A2ML

```

Figure 13: A2ML-file for ASAP2 standard interfaces

### 8.3 Structure of BLOBs

As described at ASAP1b specification the ASAP1b device needs some 'binary large objects' (BLOBs) containing interface depending parameters. As long as the structure of BLOB's doesn't contain any optional parts or any parameter arrays with variable length, the structure of BLOBs at ASAP1b device need no further explanation:

? The structure of BLOBs is equivalent to the structure described at A2ML-file (also see "6.2 Predefined data types").

Since ASAP2 Version 1.31 there is a specification for the storage layout of the BLOBs which also describes BLOBs with optional parts.

### 8.4 Example of ASAP2 metalanguage

The following example illustrates the ASAP2 metalanguage (A2ML) on the basis of the format definition of ASAP2 version 1.0. It is not adjusted to the modifications of version 1.20 and version 1.21.

#### Note:

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 {
        enum { "STD_AXIS" = 0, "COM_AXIS" = 1, "FIX_AXIS" = 2 };
        char[20];
        char[20];
        int;
        double;
        double;
    };
    taggedstruct {
        "MAX_GRAD" double;
        "MONOTONY" enum { "MON_INCREASE" = 0, "MON_DECREASE" = 1 }; /* monotony */
        "BYTE_ORDER" enum endian;
        "FIX_AXIS_PAR" struct {
            int;
            int;
            int;
        };
        "DEPOSIT" enum { "DIFFERENCE" = 0, "ABSOLUTE" = 1 }; /* with deviation from standard */
        "AXIS_PTS_REF" char[20];
    };
};

struct characteristic {
    struct {
        char[20];
        char[120];
        enum { "VALUE"=0, "CURVE"=1, "MAP"=2, "CUBOID"=3, "VAL_BLK"=4, "ASCII"=5}; /* type */
        long;
        char[20];
        double;
        char[20];
        double;
        double;
    };
    taggedstruct {
        "BYTE_ORDER" enum endian;
        "BIT_MASK" long;
        "FUNCTION_LIST" ( char[20] )*;
        "NUMBER" int;
        "EXTENDED_LIMITS" struct {
            double;
            double;
        };
        ( block "AXIS_DESCR" struct axis_descr );
        ( "IF_DATA" taggedunion ifp_characteristic );
    };
};

/* ***** Description of axis point distributions ***** */
struct axis_pts {
    struct {
        char[20];
        char[120];
        long;
        char[20];
        char[20];
        double;
        char[20];
        int;
        double;
        double;
    };
    taggedstruct {
        "DEPOSIT" enum { "DIFFERENCE" = 0, "ABSOLUTE" = 1 };
        "BYTE_ORDER" enum endian;
        "FUNCTION_LIST" ( char[20] )*;
        ( "IF_DATA" taggedunion ifp_characteristic );
    };
};

```

```

/* ***** Description of measurements ***** */
struct measurement {
    struct {
        char[20];           /* mandatory part */
        char[120];          /* name */
        enum datatype;      /* descr */
        char[20];           /* data type */
        int;                /* conversion method */
        double;             /* resolution in bits */
        double;             /* accuracy */
        double;             /* lower limit */
        double;             /* upper limit */
    };
    taggedstruct {
        "BIT_MASK"          long;           /* optional part */
        "BYTE_ORDER"        enum endian;    /* bit mask */
        "FUNCTION_LIST"      ( char[20] )*;  /* byte order */
        "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 {
        char[20];           /* mandatory part */
        char[120];          /* name */
        enum { "TAB_INTP" = 0, "TAB_NOINTP" = 1, "RAT_FUNC" = 2, "FORM" = 3 }; /* descr */
        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 addrtype { "PBYTE" = 1, "PWORD" = 2, "PLONG" = 3, "DIRECT" = 4 };

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

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

struct abl_addr {
    int no;
};

struct abl_datatype {
    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_datatype;
        ("RESERVED"         struct abl_datatype) *;
        "AXIS_PTS_X"        struct axis_pts;
        "AXIS_PTS_Y"        struct axis_pts;
        "NO_AXIS_PTS_X"     struct abl_datatype;
        "NO_AXIS_PTS_Y"     struct abl_datatype;
        "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_datatype;
        "SHIFT_ OP_Y"       struct abl_datatype;
        "OFFSET_X"          struct abl_datatype;
        "OFFSET_Y"          struct abl_datatype;
    };
};
```

---

End File EXAMPLE.AML

---

## 8.5 Example of description file

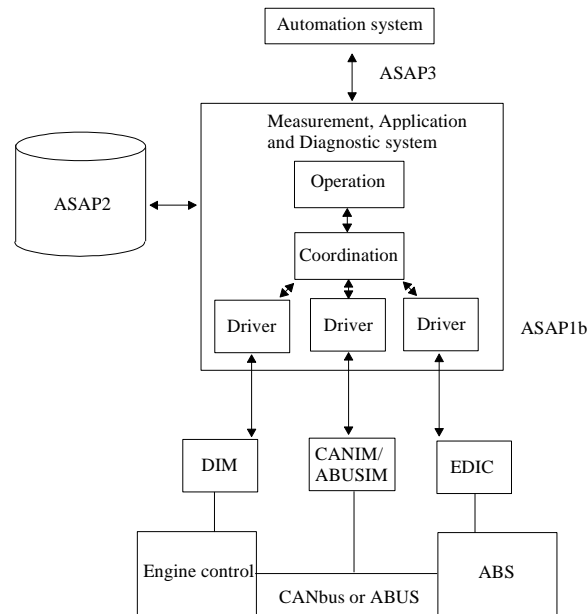


Figure 14: Project example

On the basis of the example shown in Figure 14 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
/* A2ML-file defining the interfaces ASAP1B_DIM and ASAP1B_CAN. */
/* ***** */

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

taggedunion if_data {
    "ASAP1B_DIM" taggedstruct { /* optional parameters */
        (block "SOURCE" struct {
            struct { /* indispensable */
                char [101]; /* source name (string) */
                int; /* min period (conforming together with min factor the fastest) */
                /* ...samplingrate available */
                long; /* min factor */
            };
        }
        taggedstruct {
            block "QP_BLOB" struct {
                long; /* adr_distab */
                int; /* len_distab */
                long; /* addr_outp */
                long; /* trgid */
            };
        }
    };
}
/* multiple SOURCE */

```

```

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

block "KP_BLOB" struct {               /* KP_BLOB specification for ASAP1b */
    long;                               /* address */
    enum addr_typ;                     /* addr_size */
};

block "DP_BLOB" struct {               /* DP_BLOB specification for ASAP1b */
    enum mem_typ;                      /* mem_typ */
};
block "PA_BLOB" struct {               /* PA_BLOB specification for ASAP1b */
    enum addr_mode;                   /* addressing mode */
};
};                                     /* end of: "ASAP1B_DIM" taggedstruct */

"ASAP1B_CAN" taggedstruct {           /* optional parameters */
    (block "SOURCE" struct {
        struct {                       /* indispensable */
            char [101];                /* source name (string)*/
            int;                       /* min period ( conforming together with min factor */
                                           /* the fastest samplingrate available ). */
            long;                      /* min factor */
        };
    });
};                                     /* multiple SOURCE may exist */

block "TP_BLOB" struct {
    int;                                /* bus timing */
};

block "KP_BLOB" struct {               /* KP_BLOB specification for ASAP1b */
    char[33];                          /* messagename */
    long;                              /* identifier */
    int;                              /* messagesize */
    char[101];                         /* messagesource */
    int;                              /* startbit */
    int;                              /* datasize */
    taggedstruct {
        "MULTIPLEX" struct {
            int;                       /* startbit */
            int;                       /* datasize */
            long;                      /* tag */
        };
    };
};                                     /* end of: block "KP_BLOB" */
};                                     /* end of: "ASAP1B_CAN" taggedstruct */
};                                     /* end of: taggedunion if_data */
/end A2ML

```

---

End of file SUPP1\_IF.AML

---



---

File SUPP2\_IF.AML

---

```

/begin A2ML
taggedunion if_data {
    "ASAP1B_EDIC" taggedstruct {       /* optional parameters */
        (block "SOURCE" struct {
            struct {                   /* indispensable */
                char [101];            /* source name (string)*/
                int;                   /* min period ( conforming together with min factor the fastest */
                                           /* ...samplingrate available ). */
                long;                  /* min factor */
            };
            taggedstruct {
                block "QP_BLOB" struct {
                    :
                };
            };
        }
    );
};                                     /* multiple SOURCE */

block "TP_BLOB" struct {

```

```

        :
    };

    block "KP_BLOB" struct {          /* KP_BLOB specification for ASAP1b */
        :
    };

    block "DP_BLOB" struct {          /* DP_BLOB specification for ASAP1b */
        :
    };

    block "PA_BLOB" struct {          /* PA_BLOB specification for ASAP1b */
        :
    };

    };                                /* end of: "ASAP1B_EDIC" taggedstruct */
};                                    /* end of: taggedunion if_data */
/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 14"
/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        "(0111) 22222"
    ECU              "Engine control"
    CPU_TYPE         "Intel 0815"
    NO_OF_INTERFACES 2
    /begin MEMORY_LAYOUT PRG_DATA      0x0000 0x8000 -1 -1 -1 -1 -1
      /begin IF_DATA ASAP1B_DIM
        /begin DP_BLOB EXTERN /end DP_BLOB      /* memory type */
        /begin PA_BLOB DIRECT /end PA_BLOB      /* addressing mode */
      /end IF_DATA
    /end MEMORY_LAYOUT
    SYSTEM_CONSTANT  "CONTROLLERx CONSTANT1" "0.99"
    SYSTEM_CONSTANT  "CONTROLLERx CONSTANT2" "2.88"
    SYSTEM_CONSTANT  "CONTROLLERx CONSTANT3" "-7"
    SYSTEM_CONSTANT  "ANY-PARAMETER"         "3.14159"
  /end MOD_PAR

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

```

```

/begin IF_DATA ASAP1B_DIM
  /begin SOURCE "angular synchronous" 101 1
    /begin QP_BLOB 0x5661 20 0xE001 2
    /end QP_BLOB
  /end SOURCE
  /begin SOURCE "time synchronous, rate 20ms" 4 2
    /begin QP_BLOB 0x3441 20 0xE041 3
    /end QP_BLOB
  /end SOURCE
  /begin TP_BLOB 14 /end TP_BLOB
/end IF_DATA

/begin IF_DATA ASAP1B_CAN
  /begin SOURCE "observing CAN-objects" 1000 1 /end SOURCE
  /begin TP_BLOB $FA /end TP_BLOB
/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 */
/begin IF_DATA ASAP1B_DIM
  /begin DP_BLOB EXTERN /end DP_BLOB /* memory type */
  /begin PA_BLOB DIRECT /end PA_BLOB /* addressing mode */
/end IF_DATA
/begin FUNCTION_LIST V_LIM /* reference to functions */
/end FUNCTION_LIST
/end CHARACTERISTIC

/begin CHARACTERISTIC PUMCD "Pump characteristic map"
  MAP /* type: characteristic map
  0x7140 /* address */
  DAMOS_KF /* deposit */
  100.0 /* max_diff */
  VOLTAGE /* conversion */
  0.0 /* lower limit */
  5000.0 /* upper limit */
  /* interface-spec. parameters: address location, addressing */
/begin IF_DATA ASAP1B_DIM
  /begin DP_BLOB EXTERN /end DP_BLOB /* memory type */
  /begin PA_BLOB INDIRECT /end PA_BLOB /* addressing mode */
/end IF_DATA
/begin AXIS_DESCR /* X-axis: */
  STD_AXIS /* standard axis (no group or fixed characteristic map) */
  N /* input quantity */
  N_RULE /* conversion */
  16 /* maximum number of axis points */
  0.0 /* lower limit */
  5800.0 /* upper limit */
  MAX_GRAD 20.0 /* max_grad */
/end AXIS_DESCR
/begin FUNCTION_LIST CLDSTRT FLLD /* reference to functions */
/end FUNCTION_LIST
/end CHARACTERISTIC

```



```

/begin MEASUREMENT    M_ECORR
                        "corrected fuel mass"
                        UWORD                               /* data type */
                        M_E                                  /* reference to conversion method */
                        1                                    /* resolution in bits */
                        0.001                               /* accuracy in '%' */
                        0.0                                 /* lower limit */
                        43.0                                /* upper limit */
                        BIT_MASK 0xFF                      /* bit mask */
/begin IF_DATA  ASAP1B_DIM
    /begin DP_BLOB EXTERN /end DP_BLOB                    /* memory type */
    /begin PA_BLOB DIRECT /end PA_BLOB                    /* addressing mode */
    /begin KP_BLOB 0x8038 WORD /end KP_BLOB                /* address, address length */
/end IF_DATA
/begin IF_DATA  ASAP1B_CAN
    /begin KP_BLOB
        "message-x"                                       /* interface-specific part */
        0x0123                                           /* message name */
        8                                                 /* identifier */
        8                                                 /* message length */
        "sender-Y"                                       /* sender */
        5                                                 /* start bit */
        16                                                /* bit length */
    /end KP_BLOB
    /end IF_DATA
    /begin FUNCTION_LIST  CLDSTRT FLLD                    /* reference to functins */
    /end FUNCTION_LIST
/end MEASUREMENT

/begin MEASUREMENT    N
                        "current speed"
                        UWORD                               /* data type */
                        N_RULE                             /* reference to conversion method */
                        4                                    /* resolution in bits */
                        0.006                               /* accuracy in '%' */
                        0.0                                 /* lower limit */
                        5800.0                              /* upper limit */
                        BIT_MASK 0xFFFF                    /* bit mask */
/begin IF_DATA  ASAP1B_DIM
    /begin DP_BLOB EXTERN /end DP_BLOB                    /* memory type */
    /begin PA_BLOB DIRECT /end PA_BLOB                    /* addressing mode */
    /begin KP_BLOB 0x8020 WORD /end KP_BLOB                /* address, address length */
/end IF_DATA
/begin IF_DATA  ASAP1B_CAN
    /begin KP_BLOB
        "message-x"                                       /* interface-specific part */
        0x0123                                           /* message name */
        8                                                 /* identifier */
        8                                                 /* message length */
        "sender-Y"                                       /* sender */
        21                                                /* start bit */
        16                                                /* bit length */
    /end KP_BLOB
    /end IF_DATA
    /begin FUNCTION_LIST  V_LIM CLDSTRT FLLD              /* reference to functions */
    /end FUNCTION_LIST
/end MEASUREMENT

/begin COMPU_METHOD    FACTOR01                             /* name */
                        "factor 1"                         /* long identifier */
                        RAT_FUNC                           /* fractional rational function */
                        "%4.0"                             /* format string */
                        ""                                  /* unit */
                        /* coefficients for polynome conversion */
                        COEFS 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_INT                             /* conversion table with interpolation */
                        "%4.0"                             /* format string */
                        "mg/H"                             /* unit */
                        COMPU_TAB  "AMOUNT"                /* reference to table */
/end COMPU_METHOD

```

## ASAM MCD 2MC / ASAP2 Interface Specification

```
/begin COMPU_METHOD  N_RULE                                /* name */
                                                                /* long identifier */
                                                                /* fractional rational function */
                                                                /* format string */
                                                                /* 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 */
                                                                /* long identifier */
                                                                /* fractional rational function */
                                                                /* format string */
                                                                /* 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 */
                                                                /* table with interpolation */
                                                                /* number of value pairs */
                                                                /* value pairs */
        TAB_INTP              4
        0 0.0 100 10.0 156 30.0 255 43.0
/end COMPU_TAB

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

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

/begin RECORD_LAYOUT  DAMOS_KF                                /* DAMOS characteristic diagram */
        SRC_ADDR_X          1                                /* description of the addresses of the X-input quantities */
                                                                /* position in memory */
                                                                /* datatype */
        UWORD
        NO_AXIS_PTS_X        2                                /* description of the number of X-axis points */
                                                                /* position in memory */
                                                                /* word length */
        UBYTE
        AXIS_PTS_X          3                                /* description of the X-axis point values */
                                                                /* position in memory */
                                                                /* data type of the axis point values */
                                                                /* increasing index with increasing addresses */
                                                                /* direct addressing */
        INDEX_INCR
        DIRECT
        SRC_ADDR_Y          4                                /* description of the addresses of the Y-input quantities */
                                                                /* position in memory */
                                                                /* datatype */
        UWORD
        NO_AXIS_PTS_Y        5                                /* description of the number of Y-axis points */
                                                                /* position in memory */
                                                                /* word length */
        UBYTE
        AXIS_PTS_Y          6                                /* description of the Y-axis point values */
                                                                /* position in memory */
                                                                /* data type of the axis point values */
                                                                /* increasing index with increasing addresses */
                                                                /* direct addressing */
        INDEX_INCR
        DIRECT
        FNC_VALUES          7                                /* description of the function values */
                                                                /* position in memory */
                                                                /* data type of the table values */
                                                                /* deposited in columns */
                                                                /* direct addressing */
        UBYTE
        COLUMN_DIR
        DIRECT
/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 ASAP1B_EDIC
  /begin SOURCE
    :
    /begin QP_BLOB
      :
      /end QP_BLOB
    /end SOURCE
    /begin TP_BLOB
      :
      /end TP_BLOB
  /begin 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
/begin IF_DATA ASAP1B_EDIC
    /begin KP_BLOB
        "SND" 0x10 0x00 0x05 0x08
        "RCV" 4 UWORD
    /end KP_BLOB
    /begin PA_BLOB
        :
    /end PA_BLOB
    /begin DP_BLOB
        :
    /end DP_BLOB
/begin FUNCTION_LIST    ID_ADJUSTM FL_ADJUSTM /* reference to functions */
/begin FUNCTION_LIST
/end MEASUREMENT
:
:
:
/end MODULE
_____ end of file ABS_ECU.A2L _____
```

## 8.6 AML example for ASAP1a-CCP

### Note:

1. Some elements in this example are not compatible with ASAP2 V1.2. Therefore extensions are necessary which lead to new keywords in ASAP2 V1.3
2. The example is only valid with ASAP2 V1.3 and higher (new keywords)

```

/*****
/*
/*  ASAP2 Meta Language for CCP CAN Calibration Protocol V2.1
/*  Assumes ASAP2 V1.2 or later
/*
/*  AML Version V2.6, 18.10.2002
/*
/*  Vector Informatik, Zaiser
/*  Hewlett Packard, Krueger
/*  ETAS, Maier
/*  Vector Informatik, Schuermans
/*
/*  Datatypes:
/*
/*  A2ML      ASAP2      Windows  Erlaeuterung
/*  -----
/*  uchar     UBYTE      BYTE     unsigned 8 Bit
/*  char      SBYTE      char     signed 8 Bit
/*  uint      UWORD      WORD     unsigned integer 16 Bit
/*  int       SWORD      int      signed integer 16 Bit
/*  ulong     ULONG      DWORD    unsigned integer 32 Bit
/*  long      SLONG      LONG     signed integer 32 Bit
/*  float     FLOAT32_IEEE float 32 Bit
/*
*****/
"ASAP1B_CCP" taggedstruct {
  /* Beschreibung der DAQ-Listen */
  (block "SOURCE" struct {
    struct {
      char [101]; /* Name of the DAQ-List (data acquisition list),
                  measurement source . */
      /* If the DAQ-Liste only supports one fixed ECU
      sampling rate, it can be declared below
      to achieve compatibility with the ASAP2 standard.
      Otherwise description of the possible ECU
      sampling rates in QP_BLOB */
      int; /* Period definition : Basic scaling unit in
            CSE defined in ASAP1b (CSE=Code for Scaling Unit) */
      long; /* Period definition : Rate in Scaling Units */
    };
    taggedstruct {
      "DISPLAY_IDENTIFIER" char[32];
      block "QP_BLOB" struct {
        uint; /* Number of the DAQ-List 0..n */
        taggedstruct {
          "LENGTH" uint; /* Length of the DAQ-Liste, maximum number of
                          the useable ODTs */
          "CAN_ID_VARIABLE"; /* CAN-Message-ID is variable */
          "CAN_ID_FIXED" ulong; /* CAN-Message-ID of the DTOs is fixed,
                                Default DTO
                                Bit31 = 1: extended Identifier
                                Bit31 = 0: standard Identifier
                                */
          /* Not applied if the ECU uses the DTM-Id */
          ("RASTER" uchar)*; /* Supported CCP Event Channel Names
                              of this DAQ List */
          ("EXCLUSIVE" int)*; /* Exclusion of other DAQ-Lists */
          "REDUCTION_ALLOWED"; /* Data reduction possible */
          "FIRST_PID" uchar; /* First Packet ID (PID) of the DAQ List */
        };
      };
    };
  })*; /* end of SOURCE */
  /* Description of the available ECU Sampling Rates (Event Channels) */
  (block "RASTER" struct {
    char [101]; /* CCP Event Channel Name

```

```

char [9];      /* Short Display Name of the Event Channel Name */
uchar;        /* Event Channel No., used for CCP START_STOP) */
int;          /* Period definition : basic scaling unit in CSE */
              /* as defined in ASAP1b */
long;         /* ECU sample rate of the event channel, */
              /* period definition based on the basic scaling unit */
taggedstruct {
    ("EXCLUSIVE" uchar)*;
    /* Exclusion of other CCP Event Channels */
};
}); /* end of RASTER */
/* Group several event channels to form one combined event */
/* e.g. group all cylinder synchronous events to one combined element */
(block "EVENT_GROUP" struct {
    char [101]; /* Event group name */
    char [9];   /* Short name for the event group */
    taggedstruct {
        ("RASTER" uchar)*;
    };
    /* all event channels belonging to group
    (CCP Event Channel Numbers for START_STOP) */
}); /* end of EVENT_GROUP */
/* Description of the authentication process */
block "SEED_KEY" struct {
    char[256]; /* Name of the Seed&Key DLL for CAL Priviledge,
    including file-Extension without path */
    char[256]; /* Name of the Seed&Key DLL for DAQ Priviledge,
    including file-Extension without path */
    char[256]; /* Name of the Seed&Key DLL for PGM Priviledge,
    including file-Extension without path */
}; /* end of SEED_KEY */
/* Description of the checksum calculation process */
block "CHECKSUM" struct {
    char[256]; /* Name of the Checksum DLL representing the ECU Algorithm,
    including file-Extension without path */
}; /* end of CHECKSUM */
block "TP_BLOB" struct {
    uint; /* CCP Version, High Byte: Version
    Low Byte : subversion (dec.) */
    uint; /* Blob-Version, High Byte: Version
    Low Byte : subversion (dec.) */
    ulong; /* CAN-Message ID for 'Transmitting to ECU (CRM)'
    Bit31 = 1: extended Identifier
    Bit31 = 0: standard Identifier */
    ulong; /* CAN-Message ID for 'Receiving from ECU (DTM)'
    Bit31 = 1: extended Identifier
    Bit31 = 0: standard Identifier */
    uint; /* Logical CCP-Address of the (station address) */
    uint; /* Byte order of Multiple-byte-items
    1 = high Byte first, 2 = low byte first */
    taggedstruct {
        block "CAN_PARAM" struct {
            uint; /* Quartz freq. of the elec. control unit */
            uchar; /* BTR0 */
            uchar; /* BTR1 */
        };
        "BAUDRATE" ulong; /* Baud rate in Hz. */
        "SAMPLE_POINT" uchar; /* sampling point of time in percent */
        "SAMPLE_RATE" uchar; /* number of samples per Bit (1 oder 3) */
        "BTL_CYCLES" uchar; /* number of BTL-cycles */
        "SJW" uchar; /* SJW-parameter in BTL-cycles */
        "SYNC_EDGE" enum {
            "SINGLE" = 0, /* Synchronisation only on fallende edge */
            "DUAL" = 1 /* Synchr. on falling and rising edge */
        };
        "DAQ_MODE" enum {
            "ALTERNATING" = 0, /* mode of cylcic data acquisition */
            "BURST" = 1 /* ECU is sending one ODT per cycle */
        };
        "BYTES_ONLY"; /* ECU supports max. elements of one Byte size */
        /* otherwise ECU supports different dataTypes */
        "RESUME_SUPPORTED"; /* ECU supports the Resume function */
        "STORE_SUPPORTED"; /* ECU supports the Store function */
        "CONSISTENCY" enum {
            "DAQ" = 0, /* consistency of a complete DAQ ist guaranteed */
            "ODT" = 1 /* consistency of a complete ODT ist guaranteed */
        };
        "ADDRESS_EXTENSION" enum { /* address extension */
            "DAQ" = 0, /* ECU supports only one Address extension
            within an DAQ */
            "ODT" = 1 /* ECU supports only one Address extension
            within an ODT */
        };
    };
};

```

```

};
block "CHECKSUM_PARAM" struct {
    uint; /* checksum calculation procedure
           standard types not yet defined,
           if greater of equal 1000 : manufacturer specific */
    ulong; /* Maximum block length used by an ASAP1a-CCP
           command, for checksum calculation procedure */
    taggedstruct {
        "CHECKSUM_CALCULATION" enum {
            "ACTIVE_PAGE" = 0,
            "BIT_OR_WITH_OPT_PAGE" = 1
        };
    };
}; /* end of CHECKSUM_PARAM */
(block "DEFINED_PAGES" struct {
    struct {
        uint; /* Logical No. of the memory page (1,2,..) */
        char[101]; /* Name of the memory page */
        uint; /* Address-Extension of the memory page (only
              Low Byte significant) */
        ulong; /* Base address of the memory page */
        ulong; /* Length of the memory page in Bytes */
    };
    taggedstruct {
        "RAM"; /* memory page in RAM */
        "ROM"; /* memory page in ROM */
        "FLASH"; /* memory page in FLASH */
        "EEPROM"; /* memory page in EEPROM */
        "RAM_INIT_BY_ECU"; /* memory page is initialised by ECU start-up */
        "RAM_INIT_BY_TOOL"; /* RAM- memory page is initialised by the MCS */
        "AUTO_FLASH_BACK"; /* RAM memory page is automatically flashed back */
        "FLASH_BACK"; /* feature available to flash back the RAM memory page */
        "DEFAULT"; /* memory page is standard (fallback mode) */
    };
} ) *; /* end of DEFINED_PAGES */
( "OPTIONAL_CMD" uint ) *; /* CCP-Code of the optional command available
                           in the ECU. It is recommended to declare all
                           non-standard ECU commands here */
};
}; /* end of TP_BLOB */

/* for MEMORY_SEGMENT */
("ADDR_MAPPING" struct {
    ulong; /* from */
    ulong; /* to */
    ulong; /* length */
})*; /* end of ADDR_MAPPING */

/* for CHARACTERISTIC and AXIS_PTS and MEMORY_LAYOUT */
"DP_BLOB" struct {
    uint; /* Address extension of the calibration data
           (only Low Byte significant) */
    ulong; /* Base address of the calibration data */
    ulong; /* Number of Bytes belonging to the calibration data */
}; /* end of DP_BLOB */
/* for MEASUREMENT */
block "KP_BLOB" struct {
    uint; /* Address extension of the online data
           (only Low Byte significant) */
    ulong; /* Base address of the online data */
    ulong; /* Number of Bytes belonging to the online data (1,2 or 4) */
    taggedstruct {
        ("RASTER" uchar)*;
        /* Array of event channel initialization values */
    };
}; /* end of KP_BLOB */
}; /* end of ASAP1B_CCP */

```

## 8.7 Win32 API for the ASAP1a CCP Seed&Key Algorithm DLL

Author : Michael Rossmann, SIEMENS

In order to have a common implementation of the Seed&Key algorithms used for getting access to a locked, CCP accessed ECU, the following API is proposed:

Function name:	ASAP1A_CCP_ComputeKeyFromSeed
Parameter 1:	Pointer to the Seed data, retrieved from ECU's GET_SEED cmd, 4 BYTES of data.
Parameter 2:	Pointer to 6 BYTES of data, returning the calculated Key.

An example of an implementation of such an API written in C++ is like follows:

```

/*
// Header file for ASAP1a CCP V2.1 Seed&Key Algorithm
*/

#ifndef _SEEDKEY_H_
#define _SEEDKEY_H_

#ifndef DllImport
#define DllImport    __declspec( dllimport )
#endif
#ifndef DllExport
#define DllExport    __declspec( dllexport )
#endif
#ifdef SEEDKEYAPI_IMPL    // only defined by implementor of SeedKeyApi
#define SEEDKEYAPI DllExport __cdecl
#else
#define SEEDKEYAPI DllImport __cdecl
#endif

#ifdef __cplusplus
extern "C" {
#endif

BOOL SEEDKEYAPI ASAP1A_CCP_ComputeKeyFromSeed (BYTE *Seed,
                                                unsigned short SizeSeed,
                                                BYTE *Key,
                                                unsigned short MaxSizeKey,
                                                unsigned short *SizeKey);

// Seed:      Pointer to seed data
// SizeSeed:  Size of seed data (length of ,Seed')
// Key:       Pointer, where DLL should insert the calculated key data.
// MaxSizeKey: Maximum size of ,Key'.
// SizeKey:   Should be set from DLL corresponding to the number of data
//            inserted to ,Key' (at most ,MaxSizeKey')
// Result:    The value FALSE (= 0) indicates that the key could not be
//            calculated from seed data (e.g. ,MaxSizeKey' is too small).
//            TRUE (!= 0) indicates success of key calculation.

#ifdef __cplusplus
}
#endif
#endif // _SEEDKEY_H_

```



```
//  
// Implementation of Seed&Key DLL for ASAP1a CCP V2.1  
//  
  
#include <windows.h>  
#include <memory.h>  
#define SEEDKEYAPI_IMPL  
#include "..\seedkey.h"  
  
extern "C" {  
  
BOOL SEEDKEYAPI ASAP1A_CCP_ComputeKeyFromSeed (BYTE *Seed,  
                                                unsigned short SizeSeed,  
                                                BYTE *Key,  
                                                unsigned short MaxSizeKey,  
                                                unsigned short *SizeKey)  
{  
    /* ... implementation of seed&key algorithm.  
       The result should be written in Key.  
       the result of the CCP 2.1 cmd GET_SEED.  
    */  
    MessageBox(NULL, "Call to ASAP1A_CCP_ComputeKeyFromSeed", "Seed&Key",  
MB_OK);  
}  
}
```

## 8.8 Win32 API for the ASAP1a Checksum Algorithm DLL

Author : Michael Rossmann, SIEMENS

In order to have a common interface to the implementation of the checksum algorithms used for verifying ECU calibration and program data, the following API is proposed:

Function name:	BOOL CalcChecksum(struct TRange *ptr, int nRanges, BYTE *pnChecksum, int *pnSignificant, WORD nFlags)
Parameter 1:	Pointer to an array of ranges, stored in structures of type TRange.
Parameter 2:	Number of ranges stored in the array which parameter 1 points to.
Parameter 3:	Pointer to a byte array where the checksum has to be stored. A maximum of 8 bytes will be written by the DLL, so the caller should reserve space for 8 bytes of data.
Parameter 4:	Length of actually calculated checksum (1 ... 8)
Parameter 5:	Flag field for commanding the way how the algorithm should work. Currently, only bit 0 is defined: Bit 0 = 0 means: pnChecksum shall receive the result of the checksum calculation of the algorithm. Bit 0 = 1 means: pnChecksum points to a checksum which shall be compared within the DLL with the checksum calculated by the algorithm. Return TRUE if checksums are identical, FALSE otherwise. All other bits are reserved and should be set to zero.

A TRange is defined as follows:

```
struct TRange
{
    char                *pMem;
    unsigned long       lLen;
}
```

The Calling convention is like defined in „WIN32 API specification for ASAP1b“, chapter 2.4.

An example of an implementation of such an API written in C++ is like follows:

```

/*
// checksum.h
// Header file for ASAP1a CCP V2.1 Checksum Algorithm
*/
#ifndef _CHECKSUM_H
#define _CHECKSUM_H

#ifdef __cplusplus
extern "C" {
#endif

#ifndef DllImport
#define DllImport      __declspec( dllimport )
#endif
#ifndef DllExport
#define DllExport      __declspec( dllexport )
#endif

#ifdef CHECKSUMAPI_IMPL // only defined by implementor of ChecksumApi
#define CHECKSUMAPI DllExport __cdecl
#else
#define CHECKSUMAPI DllImport __cdecl
#endif

struct TRange
{
    char          *pMem;
    unsigned long lLen;
};

#ifdef __cplusplus
extern "C" {
#endif

BOOL CHECKSUMAPI CalcChecksum(struct TRange *ptr, int nRanges, BYTE
*pnChecksum, int *pnSignificant, WORD nFlags);

#ifdef __cplusplus
}
#endif

#endif // _CHECKSUM_H

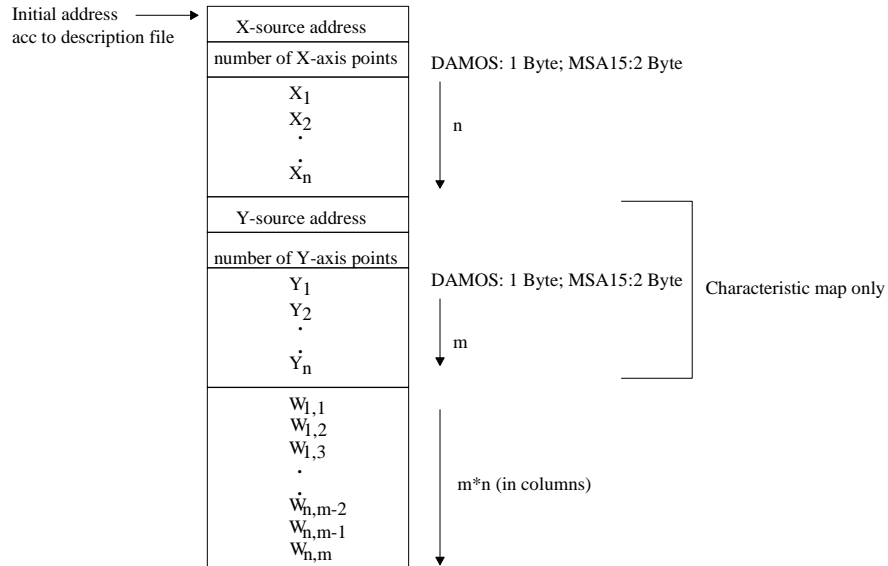
```

```
//  
// Implementation of checksum DLL for ASAP1a CCP V2.1  
//  
#include <windows.h>  
#include <memory.h>  
#define CHECKSUMAPI_IMPL  
#include "checksum.h"  
  
extern "C" {  
  
BOOL CHECKSUMAPI CalcChecksum(struct TRange *ptr, int nRanges,  
                             BYTE *pnChecksum, int *pnSignificant,  
                             WORD nFlags)  
{  
    int i;  
    unsigned long crc32_value = 0xffffffff; /* pre-condition - bits all  
                                           ones */  
    BOOL bRet = TRUE;  
    *pnSignificant = 4;  
    for(i = 0; i < nRanges; i++)  
    {  
        crc32_value = calculate_crc32(ptr[i].pMem, ptr[i].lLen,  
                                     crc32_value);  
    }  
    crc32_value = crc32_value ^ 0xffffffff; /* post-condition - one's  
                                           complement */  
    if(nFlags & 0x0001)  
    {  
        if(memcmp(&crc32_value, pnChecksum, *pnSignificant) == 0)  
            bRet = TRUE;  
        else  
            bRet = FALSE;  
    }  
    memcpy(pnChecksum, &crc32_value, sizeof(unsigned long));  
    return bRet;  
}  
} // extern "C"
```

## 9 Appendix A: Record layouts

### BOSCH: DAMOS, MSA15

#### CC, CD, AP distribution



#### FV, FVB, FCC, FCD, GCC, GCD, ASCII:

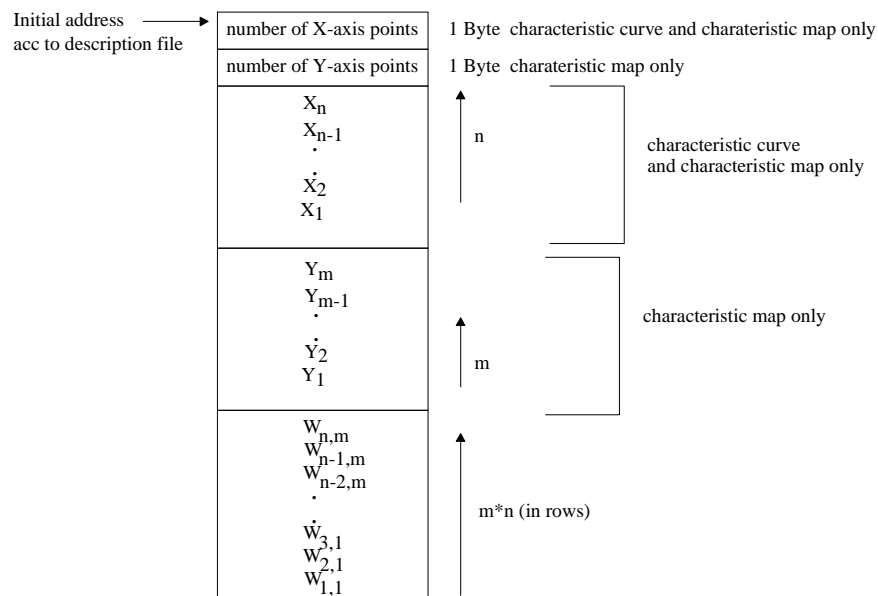
Address points directly to the function value

#### VFV, VFCC, VFCD:

Function values are addressed indirectly (via vector table)

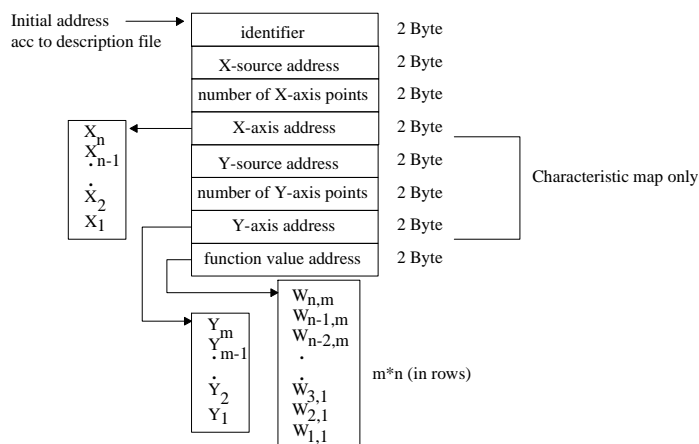
### BOSCH: KEBUSS

#### FV, CC, CD

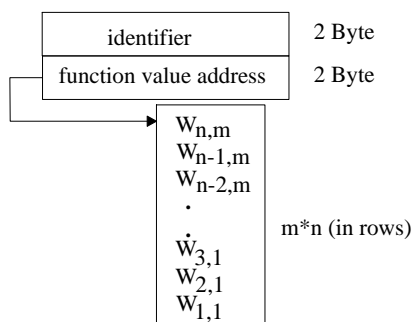


## BOSCH: C-DAMOS

### CC, CD, AP distribution



### FCC, FCD, GCC, GCD, ASCII



### FV, FVB:

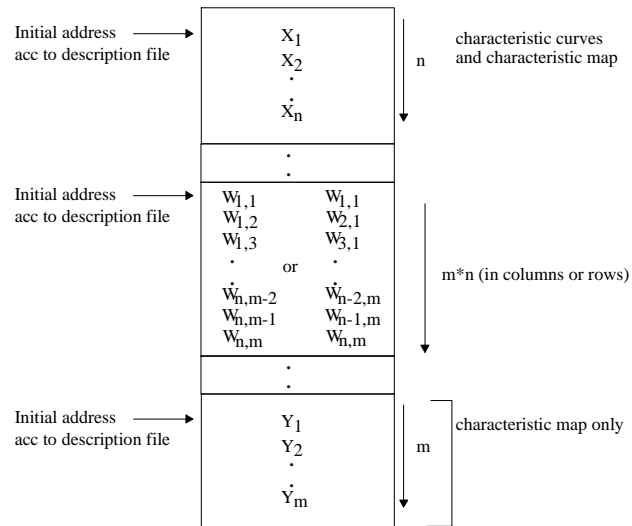
Address points directly to the value(s)

### VFCC, VFCD:

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

# Siemens: Record layout

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



## 10 Appendix B: IEEE-Floating-Point-Format

### 10.1 32-Bit Format

Sign	Biases Exponent					Significant						
S	e7	e6	...	e1	e0	B1	b2	b3	...	b21	b22	b23
31	23					0						

Table 6: IEEE-Floating-Point-Format (32-Bit)

Representation of real numbers:  $1^s 2^E b_0 b_1 b_2 b_3 \dots b_{23}$

s: 0 or 1

E: any integer between -126 and +127 ( $E = e - 127$ )

$b_i$ : 0 or 1 (where  $b_0 = 1$ )

$$\text{RealNumber} = 1^s 2^{(127) + e_i 2^i} \sum_{i=0}^{23} \frac{b_i}{2^i} \quad \text{where } b_0 = 1$$

### 10.2 64-Bit Format

Sign	Biases Exponent					Significant						
S	e10	e9	...	e1	e0	b1	b2	b3	...	b50	b51	b52
63	52					0						

Table 7: IEEE-Floating-Point-Format (64-Bit)

Representation of real numbers:  $1^s 2^E b_0 b_1 b_2 b_3 \dots b_{52}$

s: 0 or 1

E: any integer between -1022 and +1023 ( $E = e - 1023$ )

$b_i$ : 0 or 1 (where  $b_0 = 1$ )

$$\text{RealNumber} = 1^s 2^{(1023) + e_i 2^i} \sum_{i=0}^{52} \frac{b_i}{2^i} \quad \text{where } b_0 = 1$$



## 11 Appendix C: Using Reference Curves as Normalization Axes for Maps

### 11.1 Background

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

#### Overall Technique

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

#### Determining the Map Indices

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

#### Determining the Map Normalized Value

The map cells are artificially numbered with integers beginning with 0 in both the X and Y direction. Two floating-point values are obtained from two curves, one for the X direction and one for the Y direction.

If the X direction floating-point value is less than 0, then the corresponding two column indices are set to 0. Otherwise the first column index is set to the whole number portion of the value and the second column index is set to one larger. Now, if the second column index is greater than or equal to the number of columns in the map, the first and second column indices are set to the number of columns minus 1.

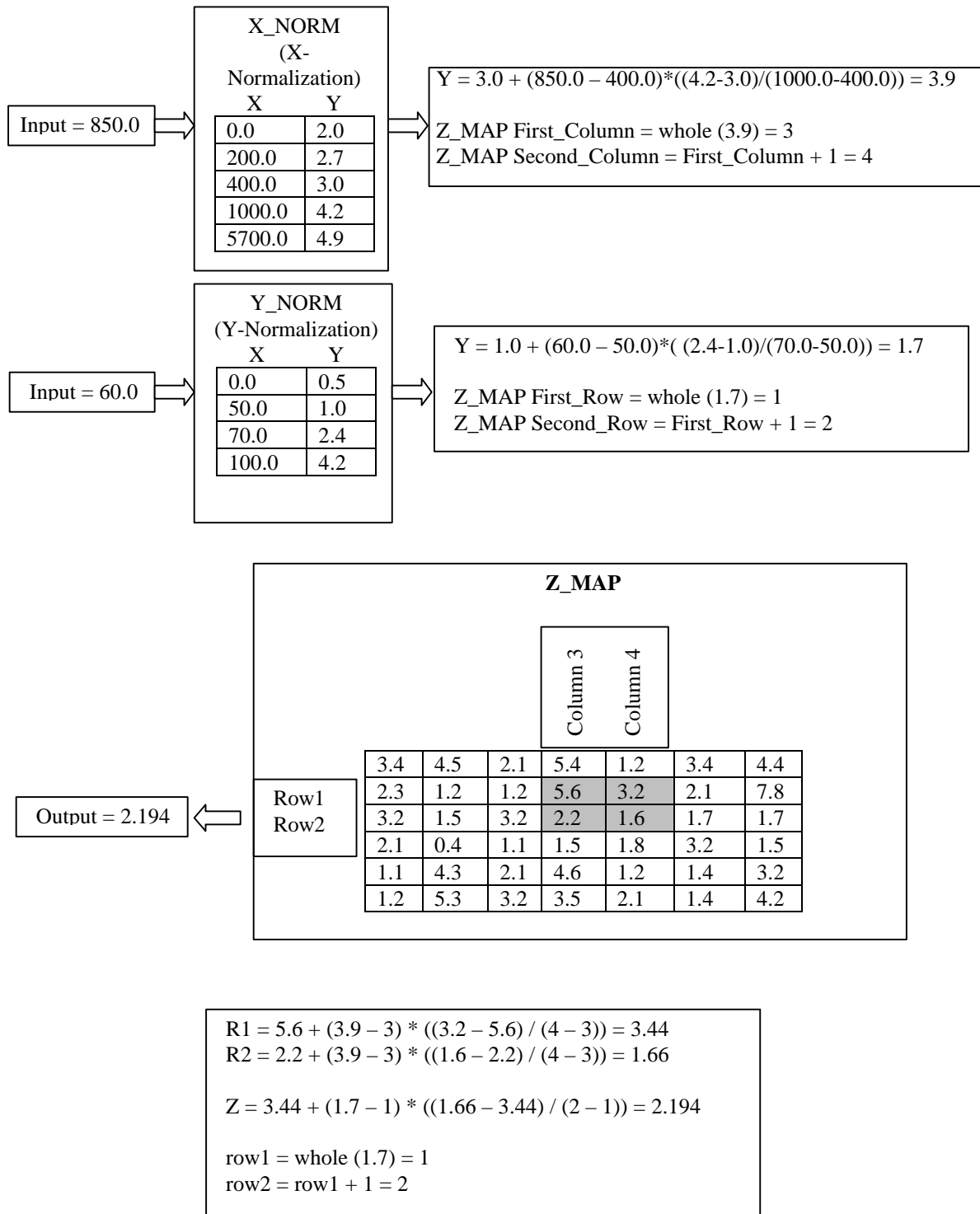
If the Y direction floating-point value is less than 0, then the corresponding two row indices are set to 0. Otherwise the first row index is set to the whole number portion of the value and the second row index is set to one larger. Now, if the second row index is greater than or

equal to the number of rows in the map, the first and second row indices are set to the number of rows minus 1.

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

## **11.2 Example**

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



## 12 Appendix D: modifications from version 1.0 to version 1.21

ARRAY_SIZE - additional keyword: array of quantities (MEASUREMENT record) .....	44, 147
ASAP2 keywords: complete list .....	274
ASAP2 keywords: <i>prohibited as ident</i> .....	28
ASAP2_VERSION: additional keyword .....	31, 45
AXIS_DESCR: No input quantity assigned.....	47
AXIS_PTS: No input quantity assigned.....	50
AXIS_PTS_X, -Y	
faulty example corrected with version 1.21.....	178
BYTE_ORDER: contradiction to general use .....	29, 61
COEFFS: Datatype of coefficients expanded to float .....	71
COMPU_TAB: redundant information (note) .....	77
COMPU_VTAB: redundant information (note) .....	79
DEF_CHARACTERISTIC: additional keyword (function orientation) .....	89, 117
delimiters added with version 1.21.....	89
Delimiters (use of /begin, /end), final definition with version 1.21.....	30
Designing A2ML-file.....	229
DIST_OP_X: additional keyword.....	95, 178
DIST_OP_Y: additional keyword.....	178
FIX_AXIS_PAR_DIST: additional keyword (see AXIS_DESCR).....	48, 105
FIX_AXIS_PAR_LIST: additional keyword (see AXIS_DESCR) .....	48, 106
FLOAT32_IEEE: additional datatype .....	29, 257
FNC_VALUES: additional enumerations of 'IndexMode' .....	108
FORMAT - additional keyword	
display format of adjustable object .....	66, 110
display format of axis points .....	47, 51, 110
display format of measurement .....	110, 147
FORMULA: delimiters added with version 1.21 .....	111
FUNCTION_LIST: additional features .....	51, 66, 118, 119, 147
FUNCTION_LIST: delimiters added with Version 1.21 .....	119
IF_DATA	
ASAP1B_ADDRESS (minimal requirements for data acquisition).....	131, 132
ASAP1B_CAN (minimal requirements for data acquisition).....	131, 133
AXIS_PTS, CHARACTERISTIC, MEMORY_LAYOUT: revised.....	129
MEASUREMENT: revised.....	130
MEMORY_SEGMENT: revised .....	135
MODULE: revised.....	136
Supplemented to MEMORY_LAYOUT.....	150
IN_MEASUREMENT: additional keyword (function orientation) .....	117, 138
IN_MEASUREMENT: delimiters added with version 1.21 .....	138
LOC_MEASUREMENT: additional keyword (function orientation) .....	117, 140
LOC_MEASUREMENT: delimiters added with version 1.21 .....	140
MAX_GRAD: absolut value of maximum gradient.....	47, 143
MAX_REFRESH - revised.....	144
MEMORY_LAYOUT: delimiters added with version 1.21 .....	150
MONOTONY	
strict monotonous behavior .....	161
MULTIPLEX: see IF_DATA ASAP1B_CAN.....	162
OUT_MEASUREMENT: additional keyword (function orientation) .....	117, 168
OUT_MEASUREMENT: delimiters added with version 1.21 .....	168
READ_ONLY	
supplemented to AXIS_DESCR .....	47, 67
supplemented to AXIS_PTS .....	51, 67
READ_WRITE - additional keyword: measurement object write access permissible.....	147, 174
REF_CHARACTERISTIC: additional keyword (function orientation) .....	117, 121, 182
REF_CHARACTERISTIC: delimiters added with version 1.21 .....	182

REF_MEASUREMENT: additional keyword (function orientation).....	121
RIP_ADDR_X, _Y, _W: additional parameter 'Datatype'.....	189
SOURCE: additional keyword, see keyword IF_DATA(MODULE).....	197
SOURCE: delimiters added with version 1.21 .....	197
SRC_ADDR_X, _Y: additional parameter 'Datatype' .....	198
Structure of BLOBs.....	233
SUB_FUNCTION: additional keyword (function orientation) .....	117, 199
SUB_FUNCTION: delimiters added with version 1.21 .....	199
SUB_GROUP: additional keyword (grouping).....	122, 200
VAR_ADDRESS: additional keyword (variant coding) .....	213, 214
VAR_ADDRESS: delimiters added with version 1.21.....	213
VAR_CHARACTERISTIC: additional keyword (variant coding).....	210, 214
VAR_CRITERION: additional keyword (variant coding) .....	210, 215
VAR_FORBIDDEN_COMB: additional keyword (variant coding) .....	210, 216
VAR_MEASUREMENT: additional keyword (variant coding) .....	215, 218
VAR_NAMING: additional keyword (variant coding).....	210, 219
VAR_SEPARATOR: additional keyword (variant coding) .....	210, 220
VARIANT_CODING: additional keyword (variant coding) .....	156, 210
VIRTUAL: delimiters added with version 1.21 .....	222

## 13 Appendix E: modifications from version 1.21 to version 1.31

### A

ADDRESS_MAPPING: description of new keyword .....	33
ALIGNMENT_FLOAT32_IEEE	
decription of new keyword.....	35
use in MOD_COMMON.....	157
use in RECORD_LAYOUT .....	175
ALIGNMENT_FLOAT64_IEEE	
decription of new keyword.....	36
use in MOD_COMMON.....	157
use in RECORD_LAYOUT .....	175
ALIGNMENT_BYTE	
decription of new keyword.....	34
use in MOD_COMMON.....	157
use in RECORD_LAYOUT .....	175
ALIGNMENT_LONG	
decription of new keyword.....	37
use in MOD_COMMON.....	157
use in RECORD_LAYOUT .....	175
ALIGNMENT_WORD	
decription of new keyword.....	38
use in MOD_COMMON.....	157
use in RECORD_LAYOUT .....	175
ALTERNATE_CURVES: new keyword in enum IndexMode .....	109
ANNOTATION	
decription of new keyword.....	39
use in AXIS_DESCR .....	46
use in AXIS_PTS .....	50
use in CHARACTERISTIC .....	65
use in FUNTION.....	117
use in MEASUREMENT .....	146
ANNOTATION_LABEL: decription of new keyword.....	41
ANNOTATION_ORIGIN: decription of new keyword .....	42
ANNOTATION_TEXT: decription of new keyword .....	43
AXIS_RESCALE_X	
description of new keywords.....	57
use in RECORD_LAYOUT .....	175
AXIS_RESCALE_Y	
description of new keywords.....	57
use in RECORD_LAYOUT .....	175
AXIS_RESCALE_Z	
description of new keywords.....	57
use in RECORD_LAYOUT .....	175

### B

BIT_MASK: use in MEASUREMENT .....	146
BIT_OPERATION	
description of new keyword .....	60
use in MEASUREMENT .....	146

### C

CALIBRATION_HANDLE: description of new keyword .....	63
CALIBRATION_METHOD	
description of new keyword .....	64
use in MOD_PAR .....	159
CHECKSUM	
description of new keyword .....	70

use in IF_DATA .....	136
COMPARISON_QUANTITY	
description of new keyword .....	72
use in CHARACTERISTIC .....	65
COMPU_VTAB_RANGE	
description of new keyword .....	81
use in MODULE .....	155

## D

DEFAULT_VALUE	
description of new keyword .....	90
use in COMPU_TAB .....	77
use in COMPU_VTAB .....	79
DEPENDENT_CHARACTERISTIC	
description of new keyword .....	91
use in CHARACTERISTIC .....	65
DISPLAY_IDENTIFIER	
description of new keyword .....	94
use in AXIS_PTS .....	50
use in CHARACTERISTIC .....	65
use in MEASUREMENT .....	146
use in SOURCE .....	197

## E

ECU_ADDRESS	
description of new keyword .....	97
use in MEASUREMENT .....	146
ECU_CALIBRATION_OFFSET	
description of new keyword .....	99
use in MOD_PAR .....	159
ERROR_MASK	
description of new keyword .....	101
use in MEASUREMENT .....	146
EVENT_GROUP	
description of new keyword .....	102
use in IF_DATA .....	136

## F

FIX_AXIS_PAR_LIST	
description of new keyword .....	106
use in AXIS_DESCR .....	46
FLOAT64_IEEE: new keyword in enum datatype .....	29
FRAME	
description of new keyword .....	114
use in MODULE .....	155
FRAME_MEASUREMENT: description of new keyword .....	116
FUNCTION_LIST: use in MEASUREMENT .....	146

## G

GROUP	
description of new keyword .....	121
use in MODULE .....	155
GUARD_RAILS	
description of new keyword .....	126
use in AXIS_PTS .....	50
use in CHARACTERISTIC .....	65

## I

ident

additional characters allowed .....	28
maximal length changed to 255.....	28

**M**

MAP_LIST	
description of new keyword .....	141
use in CHARACTERISTIC .....	65
MAX_REFRESH	
description of new keyword .....	144
use in CHARACTERISTIC .....	65
MAX_REFRESH: new scaling unit.....	144
MEMORY_SEGMENT	
new keyword .....	152
use in MOD_PAR .....	159

**N**

NO_RESCALE_X	
description of new keywords.....	165
use in RECORD_LAYOUT .....	175
NO_RESCALE_Y	
description of new keywords.....	165
use in RECORD_LAYOUT .....	175
NO_RESCALE_Z	
description of new keywords.....	165
use in RECORD_LAYOUT .....	175

**R**

RASTER	
description of new keyword .....	172
use in IF_DATA.....	136
REF_GROUP: description of new keyword .....	183
REF_MEASUREMENT: description of new keyword.....	184
REF_MEMORY_SEGMENT	
description of new keyword .....	185
use in AXIS_PTS .....	50
use in CHARACTERISTIC .....	65
use in MEASUREMENT .....	146
RES_AXIS: new keyword in enum Attribute .....	46
ROOT: description of new keyword .....	192

**S**

SEED_KEY	
description of new keyword .....	193
use in IF_DATA.....	136
string: maximal length changed to 255 .....	28
SUB_GROUP: description new keyword .....	200

**U**

USER_RIGHTS	
new keyword .....	208
use in MODULE .....	155

**V**

VIRTUAL_CHARACTERISTIC	
description of new keyword .....	223
use in CHARACTERISTIC .....	65



## 14 Appendix F: modifications from version 1.31 to version 1.40

### **B**

BYTE\_ORDER: default value MSB\_LAST ..... 157

### **C**

#### **CALIBRATION\_ACCESS**

description of new keyword ..... 62

use in AXIS\_PTS ..... 50

use in CHARACTERISTIC ..... 65

CALIBRATION\_HANDLE: multiple use in CALIBRATION\_METHOD possible ..... 64

### **F**

FLOAT64\_IEEE: faulty description corrected ..... 257

FNC\_VALUES: use of the ALTERNATE option ..... 56, 109

#### **FUNCTION\_VERSION**

description of new keyword ..... 120

use in FUNCTION ..... 117

### **M**

#### **MATRIX\_DIM**

description of new keyword ..... 142

MEMORY\_SEGMENT: new keywords in enum PrgType ..... 152

### **R**

#### **REF\_UNIT**

description of new keyword ..... 186

use in COMPU\_METHOD ..... 73

### **S**

SI\_EXPONENTS: description of new keyword ..... 196

### **U**

#### **UNIT**

description of new keyword ..... 204

use in MODULE ..... 155

UNIT\_CONVERSION: description of new keyword ..... 206

### **V**

VAR\_CHARACTERISTIC: restriction for axis points ..... 214

#### **VAR\_SELECTION\_CHARACTERISTIC**

description of new keyword ..... 217

use in VAR\_CRITERION ..... 215

VERSION: maximal length corrected ..... 221

## 15 Appendix G: modifications from version 1.40 to version 1.50

### A

#### Appendix C

Using Reference Curves as Normalization Axes for Maps ..... 258

#### Appendix G

modifications from version 1.4 to version 1.5 ..... 267

#### AXIS\_DESCR

as used in CURVE\_AXIS\_REF with CHARACTERISTIC ..... 84

for use with multiple inputs ..... 47

### B

#### BIT\_MASK

least significant bit determines shifting ..... 59

Value to be masked ..... 59

### C

#### CHARACTERISTIC

as used in CURVE\_AXIS\_REF for MAF\_NORM ..... 85

as used in CURVE\_AXIS\_REF with FUEL\_ADJ ..... 84

use with CURVE\_AXIS\_REF for SPD\_NORM ..... 85

#### COMPU\_METHOD

conversion method for air temperature ..... 75

conversion method for BYTE ..... 75

conversion method for FIXED\_<sign><length><radix> ..... 76

conversion method for FIXED\_UW\_03 ..... 75

conversion method for INTEGER ..... 75

conversion method for LONGINT ..... 76

conversion method for LONGWORD ..... 76

conversion method for SHORTINT ..... 75

conversion method for WORD ..... 75

for DiagStatus ..... 74

for RPM ..... 74

where to use NO\_PHYSICAL\_QUANTITY ..... 76

COMPU\_TAB:new datatype for input values ..... 77

#### COMPU\_VTAB

for DiagStatus ..... 80

COMPU\_VTAB:new datatype for input values ..... 79

#### CURVE\_AXIS

used to rescale axis ..... 46

#### CURVE\_AXIS\_REF

use in AXIS\_DESCR ..... 46

CURVE\_AXIS\_REF ..... 84

### E

#### ECU\_ADDRESS\_EXTENSION

description of new keyword ..... 98

use at AXIS\_PTS ..... 50

use in CHARACTERISTIC ..... 65

use in MEASUREMENT ..... 146

### F

float:fixing of the decimal point ..... 29

#### FUNCTION

added "hierarchial manner" to description ..... 118

### G

#### GROUP

added example for LUFT.....	123
as used fir objects in "luftmgr.c".....	124
as used for CLOSED_LOOP .....	124
as used for objects in "fclpkout.c" .....	124
as used for pbjects in "viosmeng.c" .....	125
as used with CALIBRATION_COMPONENTS for "L4-PCM 2002 cals".....	123
as used with SOFTWARE_COMPONENTS for "L4_PCM 2002 C modules".....	124
 <b>H</b>	
hexadecimal values:fixing of the notation .....	29
 <b>I</b>	
identifier	
case sensitivity .....	28
name spaces .....	50, 65, 73, 77, 79, 81, 114, 117, 121, 146, 156, 176, 204
IF_DATA	
clarification of parameter ident Name string	
must match ASAP1b.....	129
 <b>M</b>	
MEASUREMENT	
example as used for VdiagStatus .....	148
example as used for VfSpinLoss.....	149
MinMax values:interpretation .....	47, 51, 66, 103, 146
MODULE	
clarifiacion of description for use .....	156
MONOTONY:interpretation .....	161
 <b>N</b>	
NO_COMPU_METHOD	
for Conversion setting.....	47, 51, 66, 146
 <b>R</b>	
RECORD_LAYOUT	
use with 2D_array_table_byte .....	180
use with 2D_array_table_int.....	180
use with 2D_array_table_shortint .....	180
use with 2D_array_table_word .....	180
use with 2D_structure_table_byte.....	179
use with 2D_structure_table_int .....	179
use with 2D_structure_table_shortint .....	179
use with 2D_structure_table_word .....	179
use with 3D_array_table_byte .....	181
use with 3D_array_table_int.....	180
use with 3D_array_table_shortint .....	181
use with 3D_array_table_word.....	180
use with 3D_structure_table_byte.....	180
use with 3D_structure_table_int .....	180
use with 3D_structure_table_shortint .....	180
use with 3D_structure_table_word .....	180
use with BYTE .....	179
use with CURVE_AXIS_REF for SPD_DEP.....	85
use with INTEGER.....	179
use with LONGINT .....	179
use with LONGWORD.....	179
use with SHORTINT .....	179
use with WORD.....	179

## 16 Appendix H: 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 device (see Figure 10).

### **ASAP 2**

Standardised interface for the description data (description of control unit program: see Figure 10 and Figure 14).

### **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 individual 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):

$$Apo_i = \text{offset} + (i - 1) * 2^{\text{shift}} \quad i = \{ 1 \dots \text{number of axis points} \}$$

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.

**MCD**

Measuring, Calibration 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:

### Absolute 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

$$APo_i = \text{value}_i \quad i = \{1 \dots \text{number of axispoints}\}$$

### 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

$$APo_1 = \text{initialvalue} \quad i = \{2 \dots \text{number of axispoints}\}$$

$$APo_{i+1} = APo_i + \text{delta}_i$$

Figure 15: data deposition

## 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 string in the verbal conversion table, which describes the state of this quantity.



## 17 Appendix I: ASAP2 keywords

/begin.....	30
/end.....	30
A2ML.....	30
ABSOLUTE.....	93
ADDR_EPK.....	32
ADDRESS_MAPPING.....	33
ALIGNMENT_FLOAT32_IEEE.....	35
ALIGNMENT_FLOAT64_IEEE.....	36
ALIGNMENT_BYTE.....	34
ALIGNMENT_LONG.....	37
ALIGNMENT_WORD.....	38
ALTERNATE_CURVES.....	109
ALTERNATE_WITH_X.....	108
ALTERNATE_WITH_Y.....	108
ANNOTATION.....	39
ANNOTATION_LABEL.....	41
ANNOTATION_ORIGIN.....	42
ANNOTATION_TEXT.....	43
ARRAY_SIZE.....	44
ASAP2_VERSION.....	31, 45
ASCII.....	65
AXIS_DESCR.....	46
AXIS_PTS.....	50
AXIS_PTS_Z.....	56
AXIS_PTS_REF.....	54
AXIS_PTS_X.....	56
AXIS_PTS_Y.....	56
AXIS_RESCALE_X.....	57
AXIS_RESCALE_Y.....	57
AXIS_RESCALE_Z.....	57
BIG_ENDIAN.....	29
BIT_MASK.....	59, 139
BIT_OPERATION.....	60
BYTE.....	29
BYTE_ORDER.....	61
CALIBRATION.....	62
CALIBRATION_ACCESS.....	62
CALIBRATION_HANDLE.....	63
CALIBRATION_METHOD.....	64
CALIBRATION_VARIABLES.....	152
CHARACTERISTIC.....	65
CHECKSUM.....	70
CODE.....	152
COEFFS.....	71
COLUMN_DIR.....	108
COM_AXIS.....	46
COMPARISON_QUANTITY.....	72
COMPU_METHOD.....	73
COMPU_TAB.....	77

COMPU_TAB_REF.....	78
COMPU_VTAB .....	79
COMPU_VTAB_RANGE.....	81
CPU_TYPE .....	83
CUBOID .....	65
CURVE.....	65
CURVE_AXIS.....	46
CURVE_AXIS_REF .....	84
CUSTOMER.....	86
CUSTOMER_NO.....	87
DATA .....	152
DATA_SIZE.....	88
DEF_CHARACTERISTIC.....	89
DEFAULT_VALUE.....	90
DEPENDENT_CHARACTERISTIC.....	91
DEPOSIT.....	93
DERIVED.....	204
DIFFERENCE .....	93
DIRECT.....	29
DISPLAY_IDENTIFIER.....	94
DIST_OP_X .....	95
DIST_OP_Y .....	95
DIST_OP_Z.....	95
DP_BLOB .....	129, 130
ECU .....	96
ECU_ADDRESS .....	97
ECU_ADDRESS_EXTENSION.....	98
ECU_CALIBRATION_OFFSET .....	99
EEPROM.....	152
EPK.....	100
EPROM .....	152
ERROR_MASK.....	101
EVENT_GROUP.....	102
EXCLUDE_FROM_FLASH .....	152
EXTENDED_LIMITS.....	103
EXTENDED_SI .....	204
EXTERN .....	153
FIX_AXIS.....	46
FIX_AXIS_PAR.....	104
FIX_AXIS_PAR_DIST .....	105
FIX_AXIS_PAR_LIST.....	106
FIX_NO_AXIS_PTS_X .....	107
FIX_NO_AXIS_PTS_Y .....	107
FIX_NO_AXIS_PTS_Z.....	107
FLASH.....	152
FLOAT32_IEEE.....	29
FLOAT64_IEEE.....	29
FNC_VALUES .....	108
FORM.....	73
FORMAT.....	110
FORMULA.....	111
FORMULA_INV.....	113
FRAME.....	114

FRAME_MEASUREMENT .....	116
FUNCTION.....	117
FUNCTION_LIST .....	119
FUNCTION_VERSION .....	120
GROUP .....	121
GUARD_RAILS .....	126
HEADER.....	127
IDENTIFICATION .....	128
IF_DATA .....	129, 130, 132, 133, 134, 135, 136
INDEX_DECR.....	29
INDEX_INCR.....	29
INTERN .....	153
KP_BLOB .....	130
LITTLE_ENDIAN .....	29
LONG.....	29
MAP .....	65
MAP_LIST.....	141
MATRIX_DIM .....	142
MAX_GRAD .....	143
MAX_REFRESH .....	144
MEASUREMENT.....	146
MEMORY_LAYOUT.....	150
MEMORY_SEGMENT .....	152
MOD_COMMON .....	157
MOD_PAR.....	159
MODULE.....	155
MON_DECREASE .....	161
MON_INCREASE .....	161
MONOTONY.....	161
MSB_FIRST .....	29
MSB_LAST .....	29
MULTIPLEX .....	162
NO_AXIS_PTS_X.....	163
NO_AXIS_PTS_Y .....	163
NO_AXIS_PTS_Z .....	163
NO_CALIBRATION .....	62
NO_INPUT_QUANTITY.....	47, 50
NO_OF_INTERFACES .....	164
NO_RESCALE_X.....	165
NO_RESCALE_Y.....	165
NO_RESCALE_Z.....	165
NOT_IN_MCD_SYSTEM.....	62
NUMBER.....	166
NUMERIC .....	219

OFFLINE_CALIBRATION .....	62
OFFLINE_DATA .....	152
OFFSET_X .....	167
OFFSET_Y .....	167
OFFSET_Z .....	167
PA_BLOB .....	129, 130
PBYTE .....	29
PHONE_NO .....	169
PLONG .....	29
PRG_CODE .....	150
PRG_DATA .....	150
PRG_RESERVED .....	150
PROJECT .....	170
PROJECT_NO .....	171
PWORD .....	29
QP_BLOB .....	134, 197
RAM .....	152
RASTER .....	172
RAT_FUNC .....	73
READ_ONLY .....	173
READ_WRITE .....	174
RECORD_LAYOUT .....	175
REF_CHARACTERISTIC .....	182
REF_MEASUREMENT .....	184
REF_MEMORY_SEGMENT .....	185
REF_UNIT .....	186
REGISTER .....	152
RESERVED .....	152, 187
RIGHT_SHIFT .....	188
RIP_ADDR_W .....	189
RIP_ADDR_X .....	189
RIP_ADDR_Y .....	189
RIP_ADDR_Z .....	189
ROM .....	152
ROOT .....	192
ROW_DIR .....	108
S_REC_LAYOUT .....	203
SBYTE .....	29
SEED_KEY .....	193
SERAM .....	152
SHIFT_OP_X .....	195
SHIFT_OP_Y .....	195
SHIFT_OP_Z .....	195
SI_EXPONENTS .....	196
SIGN_EXTEND .....	194
SLONG .....	29
SOURCE .....	197
SRC_ADDR_X .....	198
SRC_ADDR_Y .....	198

SRC_ADDR_Z.....	198
STD_AXIS.....	46
STRICT_DECREASE.....	161
STRICT_INCREASE.....	161
SUPPLIER.....	201
SWORD.....	29
SYSTEM_CONSTANT.....	202
TAB_INTP.....	73, 77
TAB_NOINTP.....	73, 77
TAB_VERB.....	73, 79
TP_BLOB.....	136
UBYTE.....	29
ULONG.....	29
UNIT.....	204
UNIT_CONVERSION.....	206
USER.....	207
USER_RIGHTS.....	208
UWORD.....	29
VAL_BLK.....	65
VALUE.....	65
VAR_ADDRESS.....	213
VAR_CHARACTERISTIC.....	214
VAR_CRITERION.....	215
VAR_FORBIDDEN_COMB.....	216
VAR_MEASUREMENT.....	218
VAR_NAMING.....	219
VAR_SELECTION_CHARACTERISTIC.....	217
VAR_SEPARATOR.....	220
VARIABLES.....	152
VARIANT_CODING.....	210
VERSION.....	221
VIRTUAL.....	222
VIRTUAL_CHARACTERISTIC.....	223
WORD.....	29