Distr. Systems Development

ECU Calibration

ECU Test

ECU Software

Diagnostics

Process Management

Documentation

ASAP2 Tool-Set

vector

ASAP2 Tool-Set

User Manual

Version 6.0

Vector Informatik GmbH, Ingersheimer Straße 24, D-70499 Stuttgart
Phone: +49.711.80670.0, Fax: +49.711.80670.111, Email info@de.vector.com
Internet http://www.vector.de

80388

Certified Quality/Process Management

The Quality/Process Management of the whole Vector Group is certified according to DIN EN ISO 9001:2000-12.

Worldwide Subsidiaries:

Germany and all countries not named below **Vector Informatik GmbH** Ingersheimer Straße 24 70499 Stuttgart Germany Phone: +49.711.80670.0 Fax: +49.711.80670.111 info@de.vector.com http://www.vector.de	**USA, Canada, Mexico** **Vector CANtech, Inc.** Suite 550 39500 Orchard Hill Place Novi, Mi 48375 USA Phone: +1.248.449.9290 Fax: +1.248.449.9704 info@us.vector.com http://www.vector-cantech.com
France, Belgium, Luxemburg **Vector France SAS** 168, Boulevard Camélinat 92240 Malakoff France Phone: +33.1.4231.4000 Fax: +33.1.4231.4009 info@fr.vector.com http://www.vector-france.com	**Japan** **Vector Japan Co., Ltd.** Seafort Square Center Bld. 18F 2-3-12, Higashi-shinagawa, Shinagawa-ku Tokyo 140-0002 Japan Phone: +81.3.5769.7800 Fax: +81.3.5769.6975 info@jp.vector.com http://www.vector-japan.co.jp
United Kingdom & Ireland **Vector GB Limited** Rhodium, Central Boulevard Blythe Valley Park Solihull, Birmingham West Midlands, B90 8AS United Kingdom Phone: +44.121.50681.50 Fax: +44.121.50681.66 info@uk.vector.com http://www.vector-gb.co.uk	**Korea** **Vector Korea IT Inc.** Daeryung Post Tower III, 508 182-4 Guro-dong, Guro-gu Seoul, 152-790 Republic of Korea Phone: +82.2.2028.0600 Fax: +82.2.2028.0604 info@vector-korea.com http://www.vector-korea.com
Sweden, Denmark, Norway, Finland, Iceland **VecScan AB** Theres Svenssons Gata 9 417 55 Göteborg Sweden Phone: +46.31.764.7600 Fax: +46.31.764.7619 info@se.vector.com http://www.vecscan.com	**India** **Vector Informatik India Private Limited** Regus Connaught Place Level 2, Connaught Place Bund Garden Road Pune 411 001 India Phone: +91.20.4014.7673 Fax: +91.20.4014.7576 info@in.vector.com http://www.vector-france.com

Typographic Conventions

Attention
This symbol identifies warnings.

Note
This symbol identifies important notes

Example
This symbol identifies examples

Table of Contents

1 Introduction

ASAP2 Tool Set consists of the tools ASAP2 Creator, ASAP2 Updater, ASAP2 Merger and ASAP2 Comparer. ASAP2 Creator creates an ASAP2 description file from special comments in C code files. ASAP2 Updater updates the address and data type information in an ASAP2 file based on the entries of a linker map file. ASAP2 Merger merges multiple ASAP2 files into a common ASAP2 file. With ASAP2 Comparer you can compare the contents of two ASAP2 files.

Functionality

All tools run under Windows 7/XP/Vista and are configured with the command line resp. with an initialization file. No interactive user input is required, so that the tools can also be integrated into make processes. Warnings and error messages can be saved in a log file. The results can be determined via the exit code.

All tools can read ASAP2 files of all versions 1.x released by ASAM, including the current version 1.60. The generated ASAP2 files have format ASAP2 1.60.

The functioning and cooperation of the tools is demonstrated in the workflow sample, which is located in the demo directory of your installation.

2 ASAP2 Creator

The ASAP2 Creator reads in a set of source code files (e.g. all files belonging to a sub system of the ECU program) and scans their comments. From a special syntax it creates ASAP2 description code for the measurement and calibration objects, groups, conversion rules and if necessary record layouts. The created file is no complete ASAP2 file, but only an “ASAP2 fragment” file. It cannot be used independently, but must be added to the ASAP2 master file via include statement inside the MODULE block.

The ASAP2 master file is also read by the ASAP2 Creator. It can contain global conversion rules and record layouts and communication settings. Furthermore, it can include other created fragments which are checked by the ASAP2 Creator for naming conflicts.

Because the ASAP2 Creator creates CANape linker map references for all objects, the ASAP2 master file must contain the AML definition for IF_DATA CANAPE_EXT.

If the ASAP2 fragment to be created already exists and is already included in the Master ASAP2 file, the description of the objects to be created are merged: attributes which are not described in C code are kept from the existing objects (e.g. extended annotations, changed range). Attributes which are described in C code overwrite the settings of the existing object: e.g. data type and object type are always defined in C code and therefore always overwrite the settings of an existing object.

With this merge feature the objects created by ASAP2 Creator can be target-oriented reworked and the changes are kept if the fragment is recreated later.

Objects from the existing fragment, which no description in C code are not saved to the fragment file when recreating.

2.1 Command Line Parameters

The following command line parameters can be entered when calling the program:

-L <name> Name of the log file for warnings and error messages. If no log file is specified, the messages are output to the screen.

-I <name> Name for the initializing file, if this file is not located in working directory or if its name isn’t UPDATER.INI

2.2 Initialization File

The initialization file ASAP2Creator.INI normally is expected in current directory. Alternatively, a different name and directory can be set in the command line. In the initialization file the behaviour of the Creator can be controlled.

The following list contains all parameters of the INI file, their possible values and their default value. The names inside brackets are the sections for the parameters. The order of the parameters inside the sections and the order of the sections are no odds. Furthermore the names of the parameters and sections are not case sensitive.

[FILES]

This section contains settings for the used files.

Parameter name	Default value	Description
SOURCE_FILES	0	Number of C code files to be parsed.
SOURCE_FILE_<n>	“”	Name of C code file number <n>, numbers start with 1. The name may contain path information and wildcards.
SEARCH_RECURSIVE	0	When searching for source files, sub directories are searched, too.
ASAP2_MASTER	“”	Name of the ASAP2 master file
MACRO_DEFINITION	“”	Name of an initialization file containing macro definitions
OUTPUT	“”	Name of the ASAP2 fragment file to be created

[OPTIONS]

This section contains settings to control the behaviour of the Creator.

Parameter name	Default value	Description
START_SYMBOL	“@@“	Start symbol which indicates the beginning of a Creator declaration inside a comment line in C code. This symbol must consist of 2 characters and must neither contain ‚/’ nor ‚*’.
MAP_SYMBOL_PREFIX	“”	This prefix is added to the symbol names when creating the linker map references for a measure or calibration object.
MAP_FORMAT_SUPPORTS_STRUCTURES	1	To be set to 0, if the used Map format doesn’t support structures. In that case, for creating linker map references for structure components the base address of the structure has to be used with an address offset.
DISABLE_ REFERENCE_CHECK	0	If this option is set, the ASAP2 Creator doesn’t check the referenced master objects for existance. Such referenced objects are conversion rules, record layouts, input qiantities and common axes. Note: The usage of this option may lead to an inconsistant ASAP2 file.

Parameter name	Default value	Description
CREATE_WHOLE_FILE	0	If this option is set, a whole ASAP2 file is created instead of a fragment. This whole ASAP2 fie contains both the newly created measure and calibration objects and the objects and global settings of the master file.
CREATE_ROOT_GROUP	1	With setting 1 (default) for each run of Creator one root group is generated in destination file. All other user defined groups are child groups of this one. With setting 0 this implicit root group is not created, so that more than one user root groups are possible.
CREATE_FUNCTIONS	0	To be set to 1 to create FUNCTION objects instead of GROUP objects in ASAP2 file
ASAP2_VERSION	160	Optionally, an output file of older ASAP2 format 1.50 or 1.40 can be created

2.3 Exit-Codes

How successful the Creator process was can be determined by the exit code of the program.

0 No errors, no warnings

1 No errors, but warnings in log file

2 Error messages in log file

2.4 Syntax for comments in C code

Each line to be parsed by the ASAP2 Creator must be located inside a C comment (enclosed in /* */ or starting after //) and must start with a special string indicating a definition is following. This string must contain 2 characters (not allowed is '/' and '*') and is configurable in the initialization file. Default value for the string is „@@"

2.4.1 Object name and Data type

Each object definition must start with the symbol name, the object type and its data type and must be completed with the keyword END. This information is mandatory. All other information for an object is optional - if necessary, default values are used to create the ASAP2 code.

Valid object types are measure, parameter, string, curve and map which can optionally be given write access attributes.

The symbol name per default is used as linker map reference – optionally with a configurable prefix. If the A2L object name shall be different from the map symbol name, this name can be optionally added with the object type.

For the data type the ASAP2 data types can be used. For integer types an additional bitmask can be specified, which must not be zero. Furthermore, together with the data type an optional range can be specified (values in physical format). If the range is missing, it is calculated from the data type range.

Format in Backus-Naur Form: see chapter 2.6 on page 35.

Sample: Minimal configuration for creation of a measure object

From a C code comment with minimal settings the following ASAP2 code will be created:

```
/*
@@ SYMBOL = sample1
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UBYTE
@@ END
*/
```

➔

```
/begin MEASUREMENT sample1 ""
  UBYTE NO_COMPU_METHOD 0 0 0 255
  ECU_ADDRESS 0
   /begin IF_DATA CANAPE_EXT
     100
     LINK_MAP "sample1" 0 0 0 0 0 0 0
   /end IF_DATA
/end MEASUREMENT
```

Note
All addresses will be created to 0 and can be updated using the ASAP2 Updater and the Linker map file.

Sample: Object name differs from linker map name

```
/*
@@ SYMBOL = sample2
@@ A2L_TYPE = PARAMETER DifferentName
@@ DATA_TYPE = SWORD
@@ END
*/
```

➔

```
/begin CHARACTERISTIC DifferentName ""
  VALUE 0 __SWORD_Z 0 NO_COMPU_METHOD -32768 32767
  BIT_MASK 0x18
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "sample2" 0 0 0 0 0 0 0
  /end IF_DATA
```

```
/end CHARACTERISTIC
```

Sample: Parameter without write access

```
/*
@@ SYMBOL = sample3
@@ A2L_TYPE = PARAMETER READ_ONLY
@@ DATA_TYPE = UBYTE
@@ END
*/
```

➔

```
/begin CHARACTERISTIC sample3 ""
  VALUE 0 __UBYTE_Z 0 NO_COMPU_METHOD 0 3
  READ_ONLY
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "sample3" 0 0 0 0 0 0 0
  /end IF_DATA
/end CHARACTERISTIC
```

Sample: Creation of a string object with string length 10

```
/*
@@ SYMBOL = myString
@@ A2L_TYPE = STRING 10
@@ END
*/
```

➔

```
/begin CHARACTERISTIC myString ""
  ASCII 0 __UBYTE_Z 0 NO_COMPU_METHOD 0 255
  NUMBER 10
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "myString" 0 0 0 0 0 0 0
  /end IF_DATA
/end CHARACTERISTIC
```

Sample: Usage of a bit mask

If a bit mask is specified, the calculated range is adopted automatically:

```
/*
@@ SYMBOL = sample4
@@ A2L_TYPE = PARAMETER
@@ DATA_TYPE = UBYTE 0x18
@@ END
*/
```

➔

```
/begin CHARACTERISTIC sample4 ""
  VALUE 0 __UBYTE_Z 0 NO_COMPU_METHOD 0 3
```

```
  BIT_MASK 0x18
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "sample4" 0 0 0 0 0 0 0
  /end IF_DATA
/end CHARACTERISTIC
```

Sample: Parameter object with given range

For measure values only one range can be specified, for parameters an additional „extended“ range is possible:

```
/*
@@ SYMBOL = sample5
@@ A2L_TYPE = PARAMETER
@@ DATA_TYPE = UBYTE [20 ... 60] [20 ... 100]
@@ END
*/
```

➔

```
/begin CHARACTERISTIC sample5 ""
  VALUE 0 __UBYTE_Z 0 NO_COMPU_METHOD 20 60
  EXTENDED_LIMITS 20 100
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "sample5" 0 0 0 0 0 0 0
  /end IF_DATA
/end CHARACTERISTIC
```

2.4.2 Conversion Rules

To specify the conversion rule of a measure or calibration object, it is either possible to refer to an existing conversion rule (defined in ASAP2 master file) or to define a “local” conversion rule together with the object.

For string objects no conversion rule is allowed.

If an existing conversion rule is used, only its name is necessary, Optional the total length and the number of digits can be specified, if the default precision of the conversion rule shall not be used. The unit is defined together with the referred conversion rule and cannot be changed for the object.

If the conversion rule is locally defined together with the object, a linear conversion with factor and offset, an algebraic conversion with formula text or a conversion table can be specified. Furthermore, the unit and optional the number of total digits and the precision can be specified. If no number of digits is specified, the default value 3 is used.

If no conversion rule is to be used but a physical unit shall be specified for the object the keyword UNIT can be used.

With version 6.0 of ASAP2 Tool-Set it is also possible to define global conversion rules in C code and refer to them from a measure or calibration object.

Format in Backus-Naur Form: see chapter 2.6 on page 35.

Sample: Use an existing conversion rule with total length 8 and precision 2

```
/*
@@ SYMBOL = sample6
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UBYTE
@@ CONVERSION = existingConversionName 8 2
@@ END
*/
```

➔

```
/begin MEASUREMENT sample6 ""
  UBYTE existingConversionName 0 0 0 255
  ECU_ADDRESS 0
  FORMAT "%8.2"
   /begin IF_DATA CANAPE_EXT
     100
     LINK_MAP "sample6" 0 0 0 0 0 0 0
   /end IF_DATA
/end MEASUREMENT
```

Sample: Local linear conversion rule with factor 2 and offset 0

```
/*
@@ SYMBOL = sample7
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UBYTE
@@ CONVERSION = LINEAR 2 0 "Volt"
@@ END
*/
```

➔

```
/begin MEASUREMENT sample7 ""
  UBYTE sample7.Conversion 0 0 0 255
  ECU_ADDRESS 0
   /begin IF_DATA CANAPE_EXT
     100
     LINK_MAP "sample7" 0 0 0 0 0 0 0
   /end IF_DATA
/end MEASUREMENT

/begin COMPU_METHOD sample7.Conversion ""
  LINEAR "%.3" "Volt"
  COEFFS_LINEAR 2 0
/end COMPU_METHOD
```

Sample: Local conversion rule wit formula text and precision 2

```
/*
@@ SYMBOL = sample8
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UBYTE
@@ CONVERSION = FORMULA "X*X*5" "bar" 2
@@ END
*/
```

➔

```
/begin MEASUREMENT sample8 ""
  UBYTE sample8.Conversion 0 0 0 255
  ECU_ADDRESS 0
   /begin IF_DATA CANAPE_EXT
     100
     LINK_MAP "sample8" 0 0 0 0 0 0 0
   /end IF_DATA
/end MEASUREMENT

/begin COMPU_METHOD sample8.Conversion ""
  FORM "%.2" "bar"
  /begin FORMULA
    "X*X*5"
  /end FORMULA
/end COMPU_METHOD
```

Sample: Local conversion table

```
/*
@@ SYMBOL = sampleX
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UBYTE
@@ CONVERSION = TABLE 0 "Off" 1 "On" DEFAULT_VALUE "Unknown"
@@ END
*/
```

➔

```
/begin MEASUREMENT sampleX ""
  UBYTE sampleX.Conversion 0 0 0 255
  ECU_ADDRESS 0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "sampleX" 0 0 0 0 0 0 0
  /end IF_DATA
/end MEASUREMENT

/begin COMPU_METHOD sampleX.Conversion ""
  TAB_VERB "%.0" ""
  COMPU_TAB_REF sampleX.Conversion
/end COMPU_METHOD

/begin COMPU_VTAB sampleX.Conversion ""
  TAB_VERB 2 0 "Off" 1 "On"
  DEFAULT_VALUE "Unknown"
/end COMPU_VTAB
```

Sample: Physical unit without conversion rule

```
/*
@@ SYMBOL = sampleZ
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UBYTE
@@ UNIT = "metres"
```

```
@@ END
*/
```

➔

```
/begin MEASUREMENT sampleZ ""
  UBYTE sampleX.Conversion 0 0 0 255
  ECU_ADDRESS 0
  PHYS_UNIT "metres"
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "sampleZ" 0 0 0 0 0 0 0
  /end IF_DATA
/end MEASUREMENT
```

Sample: Global linear conversion rule

```
/*
@@ CONVERSION = sampleConversion
@@ A2L_TYPE = LINEAR 2 0
@@ UNIT = "Volt" 6 3
@@ DESCRIPTION = "description text"
@@ END
*/
```

➔

```
/begin COMPU_METHOD sampleConversion "description text"
  LINEAR "%6.3" "Volt"
  COEFFS_LINEAR 2 0
/end COMPU_METHOD
```

2.4.3 Comment

To specify a comment for a measure or calibration object the keyword DESCRIPTION can be used:

Format in Backus-Naur Form: see chapter 2.6 on page 35.

Sample:

```
/*
@@ SYMBOL = sample9
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = SLONG
@@ DESCRIPTION = "This is a comment"
@@ END
*/
```

➔

```
/begin MEASUREMENT sample9 "This is a comment "
  SLONG NO_COMPU_METHOD 0 0 -2147483648 2147483647
  ECU_ADDRESS 0
   /begin IF_DATA CANAPE_EXT
```

```
      100
      LINK_MAP "sample9" 0 0 0 0 0 0 0
    /end IF_DATA
/end MEASUREMENT
```

2.4.4 Display Name

To specify a short name resp. display name for a measure or calibration object the keyword ALIAS can be used:

Format in Backus-Naur Form: see chapter 2.6 on page 35.

Sample:

```
/*
@@ SYMBOL = sample10
@@ A2L_TYPE = PARAMETER
@@ DATA_TYPE = UBYTE
@@ ALIAS = ShortName
@@ END
*/
```

➔

```
/begin CHARACTERISTIC sample10 ""
  VALUE 0 __UBYTE_Z 0 NO_COMPU_METHOD 0 255
  DISPLAY_IDENTIFIER ShortName
   /begin IF_DATA CANAPE_EXT
     100
     LINK_MAP "sample10" 0 0 0 0 0 0 0
   /end IF_DATA
/end CHARACTERISTIC
```

2.4.5 Display Color

To specify the display color for a measure or calibration object in CANape the keyword COLOR can be used together with the RGB value of the desired color.

Format in Backus-Naur Form: see chapter 2.6 on page 35.

Sample:

```
/*
@@ SYMBOL = sample
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UBYTE
@@ COLOR = $BLUE$
@@ END
*/

[MACRO_DEFINITIONS]
BLUE = 0xFF0000
RED = 0x0000FF
GREEN = 0x00FF00
```

➔

```
/begin CHARACTERISTIC sample ""
  VALUE 0 __UBYTE_Z 0 NO_COMPU_METHOD 0 255
  /begin IF_DATA CANAPE_EXT
     100
     LINK_MAP "sample10" 0 0 0 0 0 0 0
     DISPLAY 0xFF0000 0 255
   /end IF_DATA
/end CHARACTERISTIC
```

2.4.6 Linker Map Reference

The ASAP2 Creator creates CANape linker map references for all objects from the symbol name. By default, it uses the same symbol name as object name for the created ASAP2 object. If the name of the ASAP2 object shall be different from linker map symbol, the ASAP2 object name can be specified optionally together with the ASAP2 object type and an optional address offset to the map symbol can be specified with keyword BASE_OFFSET.

This is useful e.g. for arrays, if the linker map file provides only the base address of the array, but you want to create one ASAP2 object for each single array element.

Format in Backus-Naur Form: see chapter 2.6 on page 35.

Sample:

```
/*
@@ Symbol = counters
@@ A2L_TYPE = MEASURE counter1
@@ DATA_TYPE = UWORD
@@ END
*/
/*
@@ Symbol = counters
@@ A2L_TYPE = MEASURE counter2
@@ DATA_TYPE = UWORD
@@ BASE_OFFSET = 2
@@ END
*/
/*
@@ Symbol = counters
@@ A2L_TYPE = MEASURE counter3
@@ DATA_TYPE = UWORD
@@ BASE_OFFSET = 4
@@ END
*/
unsigned short counters[3];
```

➔

```
 /begin MEASUREMENT counter1 ""
   UWORD NO_COMPU_METHOD 0 0 0 65535
   ECU_ADDRESS 0
   /begin IF_DATA CANAPE_EXT
     100
     LINK_MAP "counters" 0 0 0 0 0 0 0
   /end IF_DATA
 /end MEASUREMENT
```

```
/begin MEASUREMENT counter2 ""
  UWORD NO_COMPU_METHOD 0 0 0 65535
  ECU_ADDRESS 0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "counters" 0 0 0 2 0 0 0
  /end IF_DATA
/end MEASUREMENT

/begin MEASUREMENT counter3 ""
  UWORD NO_COMPU_METHOD 0 0 0 65535
  ECU_ADDRESS 0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "counters" 0 0 0 4 0 0 0
  /end IF_DATA
/end MEASUREMENT
```

If the symbol names available in Linker Map file are slightly different to the symbols names in C file declaration (e.g. leading underlines etc.) it is possible to define a global prefix in initialization file which is added to each symbol name when creating the CANape linker map reference.

Sample:

```
[OPTIONS]
MAP_SYMBOL_PREFIX = "__"

/*
@@ Symbol = alpha
@@ A2L_TYPE = MEASURE Alpha
@@ DATA_TYPE = UWORD
@@ END
*/
```

➔

```
/begin MEASUREMENT Alpha ""
  UWORD NO_COMPU_METHOD 0 0 0 65535
  ECU_ADDRESS 0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "__alpha" 0 0 0 0 0 0 0
  /end IF_DATA
/end MEASUREMENT
```

2.4.7 Group assignment

For each set of parsed C code files one ASAP2 main group is created with the name of the file and stored into the ASAP2 fragment file as a root group. The name of this main group can be modified and optionally completed by a comment with a specification independent of the object definitions:

Format in Backus-Naur Form: see chapter 2.6 on page 35.

Sample:

```
/*
@@ MAIN_GROUP = SubSystemA
@@ DESCRIPTION = "All Objects of Subsystems A"
@@ END
*/
```

Without any further configuration all newly created objects are assigned to this main group. It's possible to assign an object to another group by specifying the group path at the object definition. The group path is always relative to the created main group. The sub groups are automatically created if necessary and stored into the ASAP2 fragment file, too.

It is not possible to assign an object to a group already existing in ASAP2 main file because the assignment would have to be saved at the group definition and the ASAP2 master file shall not be modified.

With option CREATE_FUNCTIONS in initialization file it is possible to create ASAP2 function objects instead of group objects. For functions there is a further possibility to assign measure object as input or output signals.

Format in Backus-Naur Form: see chapter 2.6 on page 35.

Sample:

```
/*
@@ SYMBOL = channel
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = DOUBLE [ -100 ... 100 ]
@@ GROUP = SubGroup1 | Inputs
@@ END
*/
```

➔

```
/begin GROUP SubSystemA "All Objects of Subsystems A"
  ROOT
  /begin SUB_GROUP
    SubGroup1
  /end SUB_GROUP
  /begin REF_CHARACTERISTIC
  /end REF_CHARACTERISTIC
  /begin REF_MEASUREMENT
  /end REF_MEASUREMENT
/end GROUP

/begin GROUP SubGroup1 ""
  /begin SUB_GROUP
    Inputs
  /end SUB_GROUP
  /begin REF_CHARACTERISTIC
  /end REF_CHARACTERISTIC
  /begin REF_MEASUREMENT
  /end REF_MEASUREMENT
/end GROUP
```

```
/begin GROUP Inputs ""
  /begin REF_CHARACTERISTIC
  /end REF_CHARACTERISTIC
  /begin REF_MEASUREMENT
    channel1
  /end REF_MEASUREMENT
/end GROUP

/begin MEASUREMENT channel1 ""
  FLOAT64_IEEE NO_COMPU_METHOD 0 0 -100 100
  ECU_ADDRESS 0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "channel1" 0 0 0 0 0 0 0
  /end IF_DATA
/end MEASUREMENT
```

Note
It is possible to assign an object to more than one group by using the keyword GOUP several times. The object is assigned to the last group in each group path.

Sub groups which are implicitly defined using group paths at an object definition, have no comment. If you want to specify a description for a sub group, the sub group has to be defined explicitly.

Format in Backus-Naur Form: see chapter 2.6 on page 35.

Sample:

```
/*
@@ SUB_GROUP = Inputs
@@ DESCRIPTION = "Eingangssignale für SubGroup1"
@@ END
*/
```

Sample for creating functions:

```
[MACRO_DEFINITIONS]
CREATE_FUNCTIONS = 1

/*
@@ SYMBOL = inputSignal
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = DOUBLE [ -100 ... 100 ]
@@ GROUP IN = function1
@@ END
*/

/*
@@ SYMBOL = outputSignal
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = DOUBLE [ -100 ... 100 ]
@@ GROUP OUT = function1
```

```
@@ END
*/

/*
@@ SYMBOL = signal3
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = DOUBLE [ -100 ... 100 ]
@@ GROUP = function1
@@ END
*/
```

➔

```
 /begin FUNCTION SubSystemA "All Objects of Subsystems A"
   /begin SUB_FUNCTION
     function1
   /end SUB_FUNCTION
 /end FUNCTION

 /begin FUNCTION function1 ""
   /begin REF_CHARACTERISTIC
   /end REF_CHARACTERISTIC
   /begin IN_MEASUREMENT
     inputSignal
   /end IN_MEASUREMENT
   /begin OUT_MEASUREMENT
     outputSignal
   /end OUT_MEASUREMENT
   /begin LOC_MEASUREMENT
     Signal3
   /end LOC_MEASUREMENT
 /end FUNCTION
```

2.4.8 Dimensions

If a source code file contains an array declaration it is either possible to create one ASAP2 object for each array element or to create an ASAP2 block object for the whole array. In both cases the dimension(s) must be specified.

To split the array to several ASAP2 objects the keyword SPLIT must be used - optional together with an index template or a list of index strings to be appended to the original object name. The index template must contain a format specifier to describe how to resolve the index values. Possible format specifiers are

- “%d” (creates 0, 1, 2, … from the index values)
- “%c” (creates a, b, c,… from the index values)
- “%C” (creates A, B, C,… from the index values)

If no index form is specified and no index list is given, the object names are created by default as: *name[index]*.

If the array is splitted to several ASAP2 objects, for all created ASAP2 objects the same basic map name is used with an individual offset calculated from data type and index.

When defining an array of strings, the keyword SPLIT must be used, i.e. the array must be splitted into single strings, because the ASAP2 format doesn't support string arrays.

Format in Backus-Naur Form: see chapter 2.6 on page 35.

Sample: Create a block object

```
/*
@@ SYMBOL = switches
@@ A2L_TYPE = PARAMETER
@@ DATA_TYPE = UBYTE
@@ DIMENSION = 2 3
@@ END
*/
signed char switches[2][3];
```

➔

```
 /begin CHARACTERISTIC switches ""
   VAL_BLK 0 __UBYTE_Z 0 NO_COMPU_METHOD -128 127
   MATRIX_DIM 2 3 1
   /begin IF_DATA CANAPE_EXT
     100
     LINK_MAP "switches" 0 0 0 0 0 0 0
   /end IF_DATA
 /end CHARACTERISTIC
```

Sample: Split using standard indices

```
/*
@@ SYMBOL = stringArray
@@ A2L_TYPE = STRING 25
@@ DIMENSION = 3 SPLIT
@@ END
*/
unsigned char stringArray[25][3];
```

➔

```
 /begin CHARACTERISTIC stringArray[0] ""
   ASCII 0 __UBYTE_Z 0 NO_COMPU_METHOD 0 255
   NUMBER 25
   /begin IF_DATA CANAPE_EXT
     100
     LINK_MAP "stringArray" 0 0 0 0 0 0 0
   /end IF_DATA
 /end CHARACTERISTIC

 /begin CHARACTERISTIC stringArray[1] ""
   ASCII 0 __UBYTE_Z 0 NO_COMPU_METHOD 0 255
   NUMBER 25
   /begin IF_DATA CANAPE_EXT
```

```
    100
    LINK_MAP "stringArray" 0 0 0 1 0 0 0
  /end IF_DATA
 /end CHARACTERISTIC

 /begin CHARACTERISTIC stringArray[2] ""
  ASCII 0 __UBYTE_Z 0 NO_COMPU_METHOD 0 255
  NUMBER 25
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "stringArray" 0 0 0 2 0 0 0
  /end IF_DATA
 /end CHARACTERISTIC
```

Sample: Split using index template

```
/*
@@ SYMBOL = array2
@@ A2L_TYPE = MEASURE counter
@@ DATA_TYPE = UWORD
@@ DIMENSION = 2
@@ SPLIT USE_TEMPLATE ".%C__"
@@ END
*/
unsigned int array2[2];
```

➔

```
 /begin MEASUREMENT counter.A__ ""
  UWORD NO_COMPU_METHOD 0 0 0 65535
  ECU_ADDRESS 0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "array2" 0 0 0 0 0 0 0
  /end IF_DATA
 /end MEASUREMENT

 /begin MEASUREMENT counter.B__ ""
  UWORD NO_COMPU_METHOD 0 0 0 65535
  ECU_ADDRESS 0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "array2" 0 0 0 2 0 0 0
  /end IF_DATA
 /end MEASUREMENT
```

Sample: Split using index list

```
/*
@@ SYMBOL = array3
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UBYTE
@@ DIMENSION = 3
@@ SPLIT USE "_Eins " "_Zwei" "_Drei"
@@ END
*/
```

➔

```
/begin MEASUREMENT array3_Eins  ""
  UWORD NO_COMPU_METHOD 0 0 0 65535
  ECU_ADDRESS 0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "array3" 0 0 0 0 0 0 0
  /end IF_DATA
/end MEASUREMENT

/begin MEASUREMENT array3_Zwei ""
  UWORD NO_COMPU_METHOD 0 0 0 65535
  ECU_ADDRESS 0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "array3" 0 0 0 1 0 0 0
  /end IF_DATA
/end MEASUREMENT

/begin MEASUREMENT array3_Drei ""
  UWORD NO_COMPU_METHOD 0 0 0 65535
  ECU_ADDRESS 0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "array3" 0 0 0 2 0 0 0
  /end IF_DATA
/end MEASUREMENT
```

2.4.9 Curves and Maps

To create curves and maps, you have to define record layouts in the ASAP2 master file which describe the mapping onto the linear address range in memory. A record layout describes both the order of the components (e.g. axis points and curve values) and the data type of the components.

When creating curves and maps with the ASAP2 Creator the following axis types are supported:

- ***Standard axes:***

  The axis point values are stored together with the curve values. For the description the data type of the axis points and the maximum dimension is necessary. Optional a conversion rule and an input quantity for working point display can be configured.

- ***Fix axes:***

  The axis point values are not stored in memory, but are defined in ASAP2 file. For the axis description these values have to be specified in raw format (ECU internal values). Furthermore, a conversion rule and an input quantity for working point display can be configured.

- ***Common axes:***

  The axis point values are separated from the curve values in memory and must be described in a special axis object. They can be referenced by more

than one curve or map. For the axis description at the curve / map only the name of this special axis object is necessary.

When describing the special axis object, the data type of the axis point values and the maximum dimension must be specified. Optional a conversion rule and an input quantity for working point display can be configured.

Format in Backus-Naur Form: see chapter 2.6 on page 35.

Sample: Description of a write protected curve with standard axis

```
/*
@@ SYMBOL = myCurve1
@@ A2L_TYPE = CURVE READ_ONLY
@@ DATA_TYPE = DOUBLE [50 ... 55]
@@ LAYOUT = RL_Curve1
@@ X_AXIS = STANDARD
@@   DATA_TYPE = UWORD [2000 ... 8000]
@@   DIMENSION = 5
@@   INPUT = Drehzahl
@@ END
*/
struct {
  unsigned short input[5];
  double output[5];
} myCurve1;
```

➔

```
 /begin CHARACTERISTIC myCurve1 ""
   CURVE 0 RL_Curve1 0 NO_COMPU_METHOD 50 55
   /begin AXIS_DESCR
      STD_AXIS Drehzahl NO_COMPU_METHOD 5 2000 8000
   /end AXIS_DESCR
   /begin IF_DATA CANAPE_EXT
     100
     LINK_MAP "myCurve1" 0 0 0 0 0 0 0
   /end IF_DATA
 /end CHARACTERISTIC
```

The record layout RL_Curve1 must exist in ASAP2 master file, e.g.:

```
/begin RECORD_LAYOUT RL_Curve1 ""
  AXIS_PTS_X 1 UWORD INDEX_INCR DIRECT
  FNC_VALUES 2 FLOAT64_IEEE COL_DIR DIRECT
/end RECORD_LAYOUT
```

Sample: Description of a map with 2 fix axes without input quantities

```
/*
@@ SYMBOL = myMap1
@@ A2L_TYPE = MAP
@@ DATA_TYPE = UBYTE
@@ LAYOUT = RL_UBYTE
```

```
@@ X_AXIS = FIX 10 20 30 40 50
@@ Y_AXIS = FIX 2 4 6 8 10 12 14
@@ END
*/
unsigned char myMap1[5][7];
```

➔

```
 /begin CHARACTERISTIC myMap1 ""
   MAP 0 RL_UBYTE 0 NO_COMPU_METHOD 50 55
   /begin AXIS_DESCR
     FIX_AXIS NO_INPUT_QUANTITY NO_COMPU_METHOD 5 10 50
     /begin FIX_AXIS_PAR_LIST
       10 20 30 40 50
     /end FIX_AXIS_PAR_LIST
   /end AXIS_DESCR
   /begin AXIS_DESCR
      FIX_AXIS NO_INPUT_QUANTITY NO_COMPU_METHOD 7 2 14
     /begin FIX_AXIS_PAR_LIST
       2 4 6 8 10 12 14
     /end FIX_AXIS_PAR_LIST
   /end AXIS_DESCR
   /begin IF_DATA CANAPE_EXT
     100
     LINK_MAP "myMap1" 0 0 0 0 0 0 0
   /end IF_DATA
 /end CHARACTERISTIC
```

The record layout RL_UBYTE must exist in ASAP2 master file, e.g.:

```
 /begin RECORD_LAYOUT RL_UBYTE ""
   FNC_VALUES 1 UBYTE COL_DIR DIRECT
 /end RECORD_LAYOUT
```

Sample: Description of a curve with common axis

```
/*
@@ SYMBOL = sampleAxis
@@ A2L_TYPE = AXIS
@@ DATA_TYPE = FLOAT [ 1.0 ... 30.8]
@@ LAYOUT = RL_AXIS_FLOAT
@@ DIMENSION = 12
@@ END
*/
/*
@@ SYMBOL = sampleCurve
@@ A2L_TYPE = CURVE
@@ DATA_TYPE = UWORD [ -200 ... 400]
@@ LAYOUT = RL_UWORD
@@ X_AXIS = COMMON sampleAxis
@@ END
*/
float sampleAxis[12];
unsigned short sampleCurve[12];
```

➔

```
/begin AXIS_PTS sampleAxis ""
  0 NO_INPUT_QUANTITY RL_AXIS_FLOAT 0 NO_COMPU_METHOD 12 1.0 30.8
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "sampleAxis" 0 0 0 0 0 0 0
  /end IF_DATA
/end AXIS_PTS

/begin CHARACTERISTIC sampleCurve ""
  CURVE 0 RL_UWORD 0 NO_COMPU_METHOD -200 400
  /begin AXIS_DESCR
    COM_AXIS NO_INPUT_QUANTITY NO_COMPU_METHOD 12 1.0 30.8
    AXIS_PTS_REF sampleAxis
  /end AXIS_DESCR
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "sampleCurve" 0 0 0 0 0 0 0
  /end IF_DATA
/end CHARACTERISTIC
```

The record layouts RL_AXIS_FLOAT and RL_UWORD must exist in ASAP2 master file, e.g.:

```
/begin RECORD_LAYOUT RL_AXIS_FLOAT ""
  AXIS_PTS_X 1 FLOAT32_IEEE INDEX_INCR DIRECT
/end RECORD_LAYOUT

/begin RECORD_LAYOUT RL_UWORD ""
  FNC_VALUES 1 UWORD COL_DIR DIRECT
/end RECORD_LAYOUT
```

2.4.10 Hard coded addresses

If a variable is stored at a fixed address which is not available in Linker map file and which is not to be updated, this address can be specified with the optional keyword ADDRESS. In that case, no CANape linker map reference for that object is created. If no hard coded address is specified, the address is always created to 0.

Format in Backus-Naur Form: see chapter 2.6 on page 35.

Sample 1:

```
/*
@@ SYMBOL = sample11
@@ A2L_TYPE = STRING 128
@@ ADDRESS = 0x1200
@@ END
*/
```

➔

```
/begin CHARACTERISTIC sample11 ""
  ASCII 0x1200 __UBYTE_Z 0 NO_COMPU_METHOD 0 255
  NUMBER 128
  /begin IF_DATA CANAPE_EXT
    100
```

```
  /end IF_DATA
/end CHARACTERISTIC
```

Sample 2:

```
/*
@@ SYMBOL = sample12
@@ A2L_TYPE = PARAMETER
@@ DATA_TYPE = ULONG
@@ ADDRESS = 0x1200
@@ DIMENSION = 3 SPLIT
@@ END
*/
```

➔

```
/begin CHARACTERISTIC sample12[0] ""
  VALUE 0x1200 __ULONG_Z 0 NO_COMPU_METHOD 0 1
  /begin IF_DATA CANAPE_EXT
    100
  /end IF_DATA
/end CHARACTERISTIC

/begin CHARACTERISTIC sample12[1] ""
  VALUE 0x1204 __ULONG_Z 0 NO_COMPU_METHOD 0 1
  /begin IF_DATA CANAPE_EXT
    100
  /end IF_DATA
/end CHARACTERISTIC

/begin CHARACTERISTIC sample12[2] ""
  VALUE 0x1208 __ULONG_Z 0 NO_COMPU_METHOD 0 1
  /begin IF_DATA CANAPE_EXT
    100
  /end IF_DATA
/end CHARACTERISTIC
```

2.4.11 Default Events

For measure objects it is possible to specify the default event for measurement by using the optional keyword EVENT. The default event can be specified for CCP and XCP protocol. The ASAP2 Creator builds an IF_DATA block for ASAP1B_CCP resp. XCP from this. Therefore - when using default events - the ASAP2 master file should contain the corresponding AML description.

Format in Backus-Naur Form: see chapter 2.6 on page 35.

Sample for CCP:

```
/*
@@ SYMBOL = EventTestCCP
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UWORD
@@ EVENT CCP = 10
@@ END
*/
```

➔

```
/begin MEASUREMENT EventTestCCP ""
  UWORD NO_COMPU_METHOD 0 0 0 65535
  ECU_ADDRESS 0x0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "EventTestCCP" 0 0 0 0 0 0 0
  /end IF_DATA
  /begin IF_DATA ASAP1B_CCP
    KP_BLOB 0 0 0 RASTER 10
  /end IF_DATA
/end MEASUREMENT
```

Sample for XCP:

```
/*
@@ SYMBOL = EventTestXCP1
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UWORD
@@ EVENT XCP = FIXED 10
@@ END
*/
```

➔

```
/begin MEASUREMENT EventTestXCP1 ""
  UWORD NO_COMPU_METHOD 0 0 0 65535
  ECU_ADDRESS 0x0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "EventTestXCP1" 0 0 0 0 0 0 0
  /end IF_DATA
  /begin IF_DATA XCP
    /begin DAQ_EVENT FIXED_EVENT_LIST
      EVENT 10
    /end DAQ_EVENT
  /end IF_DATA
/end MEASUREMENT

/*
@@ SYMBOL = EventTestXCP2
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UWORD
@@ EVENT XCP = VARIABLE 10
@@ END
*/
```

➔

```
/begin MEASUREMENT EventTestXCP2 ""
  UWORD NO_COMPU_METHOD 0 0 0 65535
  ECU_ADDRESS 0x0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "EventTestXCP2" 0 0 0 0 0 0 0
  /end IF_DATA
  /begin IF_DATA XCP
    /begin DAQ_EVENT VARIABLE
```

```
      /begin AVAILABLE_EVENT_LIST
        EVENT 10
      /end AVAILABLE_EVENT_LIST
      /begin DEFAULT_EVENT_LIST
        EVENT 10
      /end DEFAULT_EVENT_LIST
    /end DAQ_EVENT
  /end IF_DATA
/end MEASUREMENT
```

2.4.12 Structures

If the C Code contains a structure definition and a few instances of this structure, it is possible to define the element properties of the structure components only once at the structure definition and re-use them for each instance.

For each leaf of the structure tree which shall result in an ASAP2 object, one description block is necessary.

For each structure instance a reference description to the used structure tree is necessary. The object names for the instance are then created by adding the leaf name to the given variable name – separated by a dot.

Sub structures which are arrays are always splitted, because the ASAP2 language doesn't support arrays of structured objects, i.e. the SPLIT keyword must always be used here! The dimension and syntax for splitted array indices can be specified like described above.

For the description of sub structures which are already defined outside the current structure a simple reference can be used to minimize the description code.

For sub structures which are no arrays, no description is necessary.

Format in Backus-Naur Form: see chapter 2.6 on page 35.

Sample 1:

```
struct xyz {

  /*
  @@ ELEMENT = sMember
  @@ STRUCTURE = xyz
  @@ A2L_TYPE = MEASURE DiffName
  @@ DATA_TYPE = UWORD [ -2000 ... 5000 ]
  @@ END
  */
  unsigned short sMember;

  /*
  @@ SUB_STRUCTURE = abc
  @@ STRUCTURE = xyz
  @@ DIMENSION = 2 SPLIT
  @@ END
  */
  struct {
```

```
    /*
    @@ ELEMENT = ArrayInStructure
    @@ STRUCTURE = xyz | abc
    @@ A2L_TYPE = PARAMETER READ_ONLY
    @@ DATA_TYPE = SBYTE [-100 ... 100]
    @@ DIMENSION = 2
    @@ END
    */
    int ArrayInStructure[2];

  } abc[2];
};

/*
@@ INSTANCE = instance1
@@ STRUCTURE = xyz
@@ END
*/
struct xyz instance1;

/*
@@ INSTANCE = instance2
@@ STRUCTURE = xyz
@@ END
*/
struct xyz instance2;
```

➔

```
/begin MEASUREMENT instance1.DiffName ""
  UWORD NO_COMPU_METHOD 0 0 0 65535
  ECU_ADDRESS 0x0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "instance1.sMember" 0 0 0 0 0 0 0
  /end IF_DATA
/end MEASUREMENT

/begin CHARACTERISTIC instance1.abc[0].ArrayInStructure ""
  VAL_BLK 0x0 __SBYTE_Z 0 NO_COMPU_METHOD -100 100
  MATRIX_DIM 2 1 1
  READ_ONLY
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "instance1.abc[0].ArrayInStructure" 0 0 0 2 0 0 0
  /end IF_DATA
/end CHARACTERISTIC

/begin CHARACTERISTIC instance1.abc[1].ArrayInStructure ""
  VAL_BLK 0x0 __SBYTE_Z 0 NO_COMPU_METHOD -100 100
  MATRIX_DIM 2 1 1
  READ_ONLY
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "instance1.abc[1].ArrayInStructure" 0 0 0 2 0 0 0
  /end IF_DATA
/end CHARACTERISTIC

/begin MEASUREMENT instance2.DiffName ""
  UWORD NO_COMPU_METHOD 0 0 0 65535
```

```
    ECU_ADDRESS 0x0
    /begin IF_DATA CANAPE_EXT
      100
      LINK_MAP "instance1.sMember" 0 0 0 0 0 0 0
    /end IF_DATA
  /end MEASUREMENT

  /begin CHARACTERISTIC instance2.abc[0].ArrayInStructure ""
    VAL_BLK 0x0 __SBYTE_Z 0 NO_COMPU_METHOD -100 100
    MATRIX_DIM 2 1 1
    READ_ONLY


   /begin IF_DATA CANAPE_EXT
      100
      LINK_MAP "instance1.abc[0].ArrayInStructure" 0 0 0 2 0 0 0
    /end IF_DATA
  /end CHARACTERISTIC

  /begin CHARACTERISTIC instance2.abc[1].ArrayInStructure ""
    VAL_BLK 0x0 __SBYTE_Z 0 NO_COMPU_METHOD -100 100
    MATRIX_DIM 2 1 1
    READ_ONLY
    /begin IF_DATA CANAPE_EXT
      100
      LINK_MAP "instance1.abc[1].ArrayInStructure" 0 0 0 2 0 0 0
    /end IF_DATA
  /end CHARACTERISTIC
```

Sample 2 – Use sub structure defined outside:

```
typedef struct {

  /*
  @@ ELEMENT = x
  @@ STRUCTURE = ValPair
  @@ A2L_TYPE = MEASURE
  @@ DATA_TYPE = UBYTE
  @@ END
  */
  unsigned char x;

  /*
  @@ ELEMENT = y
  @@ STRUCTURE = ValPair
  @@ A2L_TYPE = MEASURE
  @@ DATA_TYPE = UBYTE
  @@ END
  */
  unsigned char y;
} ValPair;

/*
@@ INSTANCE = MyValPair
@@ STRUCTURE = ValPair
@@ END
*/
ValPair MyValPair;
```

```
struct {

  /*
  @@ ELEMENT = name
  @@ STRUCTURE = ConversionTable
  @@ A2L_TYPE = STRING 32
  @@ END
  */
  const char name[32];

  /*
  @@ SUB_STRUCTURE = valPairList
  @@ STRUCTURE = ConversionTable
  @@ DATA_TYPE = STRUCTURE ValPair
  @@ DIMENSION = 2 SPLIT
  @@ END
  */
  ValPair valPairList[2];

} ConversionTable;

/*
@@ INSTANCE = MyConversionTable
@@ STRUCTURE = ConversionTable
@@ END
*/
ConversionTable MyConversionTable;
```

➔

```
/begin MEASUREMENT MyValPair.x ""
  UBYTE NO_COMPU_METHOD 0 0 0 255
  ECU_ADDRESS 0x0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "MyValPair.x" 0 0 0 0 0 0 0
  /end IF_DATA
  SYMBOL_LINK "MyValPair.x" 0
/end MEASUREMENT

/begin MEASUREMENT MyValPair.y ""
  UBYTE NO_COMPU_METHOD 0 0 0 255
  ECU_ADDRESS 0x0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "MyValPair.y" 0 0 0 0 0 0 0
  /end IF_DATA
  SYMBOL_LINK "MyValPair.y" 0
/end MEASUREMENT

/begin CHARACTERISTIC MyConversionTable.name ""
  ASCII 0x0 __UBYTE_Z 0 NO_COMPU_METHOD 0 255
  NUMBER 32
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "MyConversionTable.name" 0 0 0 0 0 0 0
```

```
  /end IF_DATA
  SYMBOL_LINK "MyConversionTable.name" 0
/end CHARACTERISTIC

/begin MEASUREMENT MyConversionTable.valPairList[0].x ""
  UBYTE NO_COMPU_METHOD 0 0 0 255
  ECU_ADDRESS 0x0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "MyConversionTable.valPairList[0].x" 0 0 0 0 0 0 0
  /end IF_DATA
  SYMBOL_LINK "MyConversionTable.valPairList[0].x" 0
/end MEASUREMENT

/begin MEASUREMENT MyConversionTable.valPairList[0].y ""
  UBYTE NO_COMPU_METHOD 0 0 0 255
  ECU_ADDRESS 0x0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "MyConversionTable.valPairList[0].y" 0 0 0 0 0 0 0
  /end IF_DATA
  SYMBOL_LINK "MyConversionTable.valPairList[0].y" 0
/end MEASUREMENT

/begin MEASUREMENT MyConversionTable.valPairList[1].x ""
  UBYTE NO_COMPU_METHOD 0 0 0 255
  ECU_ADDRESS 0x0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "MyConversionTable.valPairList[1].x" 0 0 0 0 0 0 0
  /end IF_DATA
  SYMBOL_LINK "MyConversionTable.valPairList[1].x" 0
/end MEASUREMENT

/begin MEASUREMENT MyConversionTable.valPairList[1].y ""
  UBYTE NO_COMPU_METHOD 0 0 0 255
  ECU_ADDRESS 0x0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "MyConversionTable.valPairList[1].y" 0 0 0 0 0 0 0
  /end IF_DATA
  SYMBOL_LINK "MyConversionTable.valPairList[1].y" 0
/end MEASUREMENT
```

Map files without structure information

If the used map format doesn't supply detailed structure information and contains only the base addresses of structures, for each element the offset to the base address has to be specified with keyword BASE_OFFSET. This is also necessary if you want to assign a fixed address to an instance of this structure instead of using the linker map references. Use the INI File option MAP_FORMAT_SUPPORTS_STRUCTURES = 0.

When specifying the offsets the alignment settings have to be kept in mind.

To spare the user from calculating absolute address offsets inside nested sub structures the offsets are specified relative to the current sub structure. Therefore it is

necessary to specify additionally the offset of sub structures to the parent structure – also considering the alignment settings.

Furthermore, for arrays of sub structures and arrays of structure instances the total size of the sub structure must be specified with the keyword SIZE, so that the Creator can calculate the offsets for each splitted array element. This is necessary because the sub structure can contain alignment bytes and components no ASAP2 object shall be created for.

Format in Backus-Naur Form: see chapter 2.6 on page 35.

Sample:

```
struct xyz {

  /*
  @@ ELEMENT = member1
  @@ STRUCTURE = xyz
  @@ A2L_TYPE = MEASURE
  @@ DATA_TYPE = UWORD
  @@ BASE_OFFSET = 0
  @@ END
  */
  unsigned short member1;

  /*
  @@ SUB_STRUCTURE = abc
  @@ STRUCTURE = xyz

  @@ BASE_OFFSET = 2
  Offset inside structure xyz

  @@ SIZE = 6
  Size of sub structure abc including Dummy component
  For which no ASAP2 Object is created

  @@ DIMENSION = 2 SPLIT
  @@ END
  */
  struct {

    /* Ceate no ASAP2 Object */
    short Dummy;

    /*
    @@ ELEMENT = ArrayInStructure
    @@ STRUCTURE = xyz | abc
    @@ A2L_TYPE = MEASURE
    @@ DATA_TYPE = SBYTE [-100 ... 100]
    @@ DIMENSION = 2
    @@ BASE_OFFSET = 2
    @@ END
    */
    short ArrayInStructure[2];

  } abc[2];

  /*
```

```
  @@ ELEMENT = member2
  @@ STRUCTURE = xyz
  @@ A2L_TYPE = MEASURE
  @@ DATA_TYPE = UWORD
  @@ BASE_OFFSET = 14
  @@ END
  */
  unsigned short member2;

};

/*
@@ INSTANCE = structInstance
@@ STRUCTURE = xyz
@@ END
*/
struct xyz structInstance;
```

➔

```
/begin MEASUREMENT structInstance.member1 ""
  UWORD NO_COMPU_METHOD 0 0 0 65767
  ECU_ADDRESS 0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "structInstance" 0 0 0 0 0 0 0
  /end IF_DATA
/end MEASUREMENT

/begin MEASUREMENT structInstance.abc[0].ArrayInSubStructure ""
  SBYTE NO_COMPU_METHOD 0 0 -100 100
  ECU_ADDRESS 0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "structInstance" 0 0 0 4 0 0 0
  /end IF_DATA
  MATRIX_DIM 2 1 1
/end MEASUREMENT

/begin MEASUREMENT structInstance.abc[1].ArrayInSubStructure ""
  SBYTE NO_COMPU_METHOD 0 0 -100 100
  ECU_ADDRESS 0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "structInstance" 0 0 0 10 0 0 0
  /end IF_DATA
  MATRIX_DIM 2 1 1
/end MEASUREMENT

/begin MEASUREMENT structInstance.member2 ""
  UWORD NO_COMPU_METHOD 0 0 0 65767
  ECU_ADDRESS 0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "structInstance" 0 0 0 14 0 0 0
  /end IF_DATA
/end MEASUREMENT
```

2.4.13 Overwrite Element Properties

The definition of a structure instance can overwrite special properties (e.g. range, …) of the default objects available in the corresponding structure tree.

Format in Backus-Naur Form: see chapter 2.6 on page 35.

Sample:

```
struct overwriteTest {

  struct {
    /*
    @@ ELEMENT = x
    @@ STRUCTURE = overwriteTest | A
    @@ A2L_TYPE = MEASURE
    @@ DATA_TYPE = UWORD
    @@ END
    */
    unsigned short x;
    ...
  } A;

  struct {
    /*

    @@ ELEMENT = x
    @@ STRUCTURE = overwriteTest | B
    @@ A2L_TYPE = MEASURE
    @@ DATA_TYPE = UWORD
    @@ END
    */
    unsigned short x;
    ...
  } B;
};

/*
@@ INSTANCE = ov
@@ STRUCTURE = overwriteTest
@@ OVERWRITE A | x  RANGE = [ -50 ... 50 ]
@@ OVERWRITE B | x  DESCRIPTION = "ein anderer Kommentar"
@@ END
*/
struct overwriteTest ov;
```

➔

```
/begin MEASUREMENT ov.A.x ""
  UWORD NO_COMPU_METHOD 0 0 -50 50
  ECU_ADDRESS 0x0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "ov.A.x" 0 0 0 0 0 0 0
  /end IF_DATA
/end MEASUREMENT

/begin MEASUREMENT ov.B.x "ein anderer Kommentar"
  UWORD NO_COMPU_METHOD 0 0 0 65535
```

```
  ECU_ADDRESS 0x0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "ov.B.x" 0 0 0 0 0 0 0
  /end IF_DATA
/end MEASUREMENT
```

2.5 Usage of Macros

It is possible to provide a list of macro definitions in an additional initialization file to be read by the Creator. Macros can be defined recursively and can be used at each position in a Creator declaration – also inside strings. They can also be concatenated (e.g. to define new symbol names).

To use a macro the macro name must be written in $...$.

Sample for Initialization file:

```
[MACRO_DEFINITIONS]
FACTOR_25 = 25
PI = "3.14"
PI_HALBE = "$PI$/2"
MAX_AMPL = 7
MainName = Short
SubName = Name
```

Sample for Usage of macros in C code:

```
/*
@@ SYMBOL = amplitude
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UBYTE [ 0 ... $MAX_AMPL$ ]
@@ CONVERSION = FORMULA "$PI_HALBE$*X" "bar" 2
@@ ALIAS = $MainName$$SubName$
@@ END
*/
unsigned char amplitude;
```

➔

```
 /begin MEASUREMENT amplitude ""
   UBYTE macroTest.Conversion 0 0 0 7
   ECU_ADDRESS 0
   DISPLAY_IDENTIFIER ShortName
   /begin IF_DATA CANAPE_EXT
     100
     LINK_MAP "amplitude" 0 0 0 0 0 0 0
   /end IF_DATA
 /end MEASUREMENT

 /begin COMPU_METHOD amplitude.Conversion ""
   FORM "%.2" "bar"
   /begin FORMULA
     "3.14/2*X"
   /end FORMULA
 /end COMPU_METHOD
```

Attention
To use the single character '$' inside a string (e.g. in a comment) it has to be masked out using '\', e.g. "This is a \$ inside a comment"

2.6 Format description in Backus Naur Form

```
<object_definition>      ::= <measure_definition>
                           | <parameter_definition>
                           | <string_definition>
                           | <structure_definition>
                           | <instance_definition>
                           | <maingroup_definition>
                           | <subgroup_definition>
                           | <conversion_definition>

<maingroup_definition>   ::= MAIN_GROUP = <group_name>
                             [ <description> ]
                             END

<subgroup_definition>    ::= SUB_GROUP = <group_name>
                             [ <description> ]
                             END

<measure_definition>     ::= SYMBOL = <symbol_name>
                             A2L_TYPE = MEASURE
                             [ <write_access> ]
                             [ <a2l_name> ]
                             <datatype>
                             ( <attribute> )*
                             END

<parameter_definition>   ::= SYMBOL = <symbol_name>
                             A2L_TYPE = PARAMETER
                             [ <write_access> ]
                             [ <a2l_name> ]
                             <datatype>
                             ( <attribute> )*
                             END

<string_definition>      ::= SYMBOL = <symbol_name>
                             A2L_TYPE = STRING <length>
                             [ <write_access> ]
                             [ <a2l_name>  ]
                             ( <string_attribute> )*
                             END

<conversion_definition> ::= CONVERSION = <conversion_name>
                             A2L_TYPE = LINEAR <factor> <offset>
                             UNIT = <unit> <length> <digits>
                             [ <description> ]
                             END
                           | CONVERSION = <conversion_name>
                             A2L_TYPE = FORMULA <string>
                             UNIT = <unit> <length> <digits>
                             [ <description> ]
```

```
                          END
                        | CONVERSION = <conversion_name>
                          A2L_TYPE = TABLE ( <value> <string> )+
                          [ DEFAULT_VALUE <string> ]
                          UNIT <unit> <length> <digits>
                          [ <description> ]
                          END

<write_access>          ::= WRITEABLE
                          | READ_ONLY

<length>                ::= <value>

<datatype>              ::= DATATYPE = UBYTE  [ <bitmask> ] [ <range> ]
                          | DATATYPE = SBYTE  [ <bitmask> ] [ <range> ]
                          | DATATYPE = UWORD  [ <bitmask> ] [ <range> ]
                          | DATATYPE = SWORD  [ <bitmask> ] [ <range> ]
                          | DATATYPE = ULONG  [ <bitmask> ] [ <range> ]
                          | DATATYPE = SLONG  [ <bitmask> ] [ <range> ]
                          | DATATYPE = UINT64 [ <bitmask> ] [ <range> ]
                          | DATATYPE = INT64  [ <bitmask> ] [ <range> ]
                          | DATATYPE = FLOAT                [ <range> ]
                          | DATATYPE = DOUBLE               [ <range> ]

<range>                 ::= [ <value> ... <value> ]
                            [ [ <value> ... <value> ] ]

<bitmask>               ::= <value>

<attribute>             ::= <conversion >
                          | <description>
                          | <alias>
                          | <base_offset>
                          | <group_assignment>
                          | <dimension> [ <split> ]
                          | <address>
                          | <event>
                          | <color>

<string_attribute>      ::= <description>
                          | <alias>
                          | <base_offset>
                          | <group_assignment>
                          | <address>
                          | <dimension> <split>

<conversion>            ::= CONVERSION = <conversion_name>
                            [[ <length> ] <digits> ]
                          | CONVERSION = LINEAR <factor> <offset> <unit>
                            [[ <length> ] <digits> ]
                          | CONVERSION = FORMULA <string> <unit>
                            [[ <length> ] <digits> ]
                          | CONVERSION = TABLE ( <value> <string> )+
                            [ DEFAULT_VALUE <string> ]
                            [ FORMAT <length> <digits> ]
                          | UNIT = <unit> [[ <length> ] <digits> ]

<factor>                ::= <value>

<offset>                ::= <value>
```

```
<digits>                ::= <value>

<unit>                  ::= <string>

<description>           ::= DESCRIPTION = <string>

<alias>                 ::= ALIAS = <alias_name>

<base_offset>           ::= BASE_OFFSET = <value>

<group_assignment>      ::= GROUP [ IN | OUT ] =
                            <group_name> ( | <group_name> )*

<dimension>             ::= DIMENSION = <value> [ <value> ]

<split>                 ::= SPLIT
                          | SPLIT USE ( <string> )+
                          | SPLIT USE_TEMPLATE <string>

<address>               ::= ADDRESS = <value>

<color>                 ::= COLOR = <value>

<event>                 ::= EVENT CCP = <value>
                          | EVENT XCP = FIXED <value>
                          | EVENT XCP = VARIABLE <value>

<structure_definition>  ::= ELEMENT = <element_name>
                            STRUCTURE = <structure_path>
                            A2L_TYPE = MEASURE
                            [ <write_access> ]
                            [ <a2l_name> ]
                            <datatype>
                            ( <attribute> )*
                            END
                          | ELEMENT = <element_name>
                            STRUCTURE = <structure_path>
                            A2L_TYPE = PARAMETER
                            [ <write_access> ]
                            [ <a2l_name> ]
                            <datatype>
                            ( <attribute> )*
                            END
                          | ELEMENT = <element_name>
                            STRUCTURE = <structure_path>
                            A2L_TYPE = STRING <length>
                            [ <write_access> ]
                            [ <a2l_name> ]
                            ( <string_attribute> )*
                            END
                          | ELEMENT = <element_name>
                            STRUCTURE = <structure_path>
                            A2L_TYPE = CURVE
                            [ <write_access> ]
                            [ <a2l_name> ]
                            <datatype>
                            LAYOUT = <layout_name>
                            ( <map_attribute> )*
                            X_AXIS = <axis>
```

```
                          END
                        | ELEMENT = <element_name>
                          STRUCTURE = <structure_path>
                          A2L_TYPE = MAP
                          [ <write_access> ]
                          [ <a2l_name> ]
                          <datatype>
                          LAYOUT = <layout_name>
                          ( <map_attribute> )*
                          X_AXIS = <axis>
                          Y_AXIS = <axis>
                          END
                        | ELEMENT = <element_name>
                          STRUCTURE = <structure_path>
                          A2L_TYPE = AXIS
                          [ <write_access> ]
                          [ <a2l_name> ]
                          <datatype>
                          LAYOUT = <layout_name>
                          DIMENSION = <dimension>
                          [ <input_signal> ]
                          ( <map_attribute> )*
                          END
                        | SUB_STRUCTURE = <structure_name> [ <a2l_name> ]
                          STRUCTURE = <structure_path>
                          [ DATATYPE = STRUCTURE <structure_name> ]
                          ( <structure_attribute> )*
                          END

<structure_path>        ::= <structure_name> ( | <structure_name> )*

<element_path>          ::= [ <structure_path> | ] <element_name>

<structure_attribute>   ::= DIMENSION = <value> [ <value> ] <split>
                          | BASE_OFFSET = <value>
                          | SIZE = <value>

<instance_definition>   ::= INSTANCE = <symbol_name> [ <a2l_name> ]
                            STRUCTURE = <structure_name>
                            [ <address> ]
                            [ <dimension> <split> ]
                            [ SIZE = <value> ]
                            ( <overwrite_definition> )*
                            END

<overwrite_definition>  ::= OVERWRITE <element_path> <overwrite>

<overwrite>             ::= <conversion>
                          | <description>
                          | <alias>
                          | <color>
                          | <group_assignment>
                          | RANGE = [ <value> ... <value> ]

<curve_definition>      ::= SYMBOL = <symbol_name>
                            A2L_TYPE = CURVE
                            [ <write_access> ]
                            [ <a2l_name> ]
                            <datatype>
                            LAYOUT = <layout_name>
```

```
                              ( <map_attribute> )*
                              X_AXIS = <axis>
                              END

<map_definition>          ::= SYMBOL = <symbol_name>
                              A2L_TYPE = MAP
                              [ <write_access> ]
                              [ <a2l_name> ]
                              <datatype>
                              LAYOUT = <layout_name>
                              ( <map_attribute> *)
                              X_AXIS = <axis>
                              Y_AXIS = <axis>
                              END

<axis_definition>         ::= SYMBOL = <symbol_name>
                              A2L_TYPE = AXIS [ <a2l_name> ]
                              [ <write_access> ]
                              <datatype>
                              LAYOUT = <layout_name>
                              DIMENSION = <dimension>
                              [ <input_signal> ]
                              ( <map_attribute> )*
                              END

<map_attribute>           ::= <conversion >
                            | <description>
                            | <alias>
                            | <base_offset>
                            | <group_assignment>
                            | <address>

<axis>                    ::= STANDARD <datatype>
                              DIMENSION = <dimension>
                              [ <input_signal_name> ]
                              [ <conversion> ]
                            |  FIX <list_of_axis_points>
                              [ <input_signal_name> ]
                              [ <conversion> ]
                            |  COMMON <common_axis_name>

<input_signal>            ::=  INPUT = <input_signal_name>

<list_of_axis_points>     ::=  ( <value> )+
```

3 ASAP2 Updater

The ASAP2 Updater reads an ASAP2 source file and updates the address and data type information on the basis of the entries in a linker map file. The most prevalent linker MAP formats are supported, such as IEEE, COFF, ELF and the ASCII map formats of many compilers. The desired MAP format is entered in an initialization file UPDATER.INI.

The addresses of measurement and calibration objects are updated in the common ASAP2 part and in the known interface specific parts ASAP1B_CCP, ASAP1B_MCMESS, ETK, ASAP1B_ETK, ASAP1B_KWP2000, DIM, DIM_X, CANAPE_EXT, CNP_KWPONCAN and ASAP1B_DIAGNOSTIC_SERVICES.

The other interface-specific parts of the ASAP2 file are interpreted and rewritten to the output file based on the given meta-language. All IF_DATA types are supported.

The update of the data type information is optional and needs to be activated in the initialization file. Furthermore, the data type information (basic data type and where necessary bit offset) is updated only for measurement objects and scalar calibration objects (CHARACTERISTIC objects of type VALUE), but not for more complex objects like curves or maps, because this is not generally possible with the information from the map file.
The data type of a measurement object is modified directly at the object itself. To modify the base data type of a calibration object a new record layout with the name of the calibration object is created and assigned to the calibration object.

The ASAP2 Updater also provides a filter function for masking certain objects and object groups: A special filter mode uses the variable names from the address file as a filter. User choice determines whether only those variables in the address file or only those variables NOT in the address file will be in the result file.

3.1 Command Line Parameters

When the program is called the following command line parameters can be entered:

-I <name> Name of the ASAP2 Input file

-O <name> Name of the ASAP2 Output file to be generated

-A <name> Name of the MAP file / address file

-L <name> Name of the log file for warnings and error messages. If no name is entered for this parameter, the messages are output to the screen.

-G[R] <name> Group filter mode, R: recursive

Name of a group file with the names of groups and functions, which should enter the result file.

This option can be used together with the “normal” filter mode. In this case all objects which are in one of the specified groups or functions (or in a subgroup or sub function when using the recursive option) AND all objects of the filter mode option enter the result file.

-T <name> Name for the initializing file, if this file is not located in working directory or if its name isn't UPDATER.INI

-@<name> Name of a text file with all command line parameters.

Attention
There must be NO space characters between @ and the file name.

Example
Call e.g.:
```
ASAP2Updater -@CMDLINE.TXT
```

3.2 Initialization File

The initialization file UPDATER.INI normally is expected in current directory. Alternatively, a different directory can be set in the command line. In the initialization file the behaviour of the updater can be controlled.

The following list contains all parameters of the INI file, their possible values and their default value. The names inside brackets are the sections for the parameters. The order of the parameters inside the sections and the order of the sections are no odds. Furthermore the names of the parameters and sections are not case sensitive.

[OPTIONS]

This section contains common settings.

Parameter name	Default value	Description
MAP_FORMAT	0	Specifies the format of the MAP file: 0 = Standard 1 = Borland C 2 = BSO TaskingC 166 3 = Watcom 4 = HiTech 68HC05 6 = IEEE (Cosmic, Tasking,..) 7 = Cosmic 8 = SDS 9 = Fujitsu 10 = GNU 11 = Keil 16x 12 = Borland C 32 Bit 13 = Keil 16x (including static symbols) 14 = Keil 8051 15 = ISI MatrixX 16 = Hiware HC12 17 = Texas Instruments TMS470 18 = Archimedes HC11 19 = COFF 20 = IAR 21 = VisualDSP 22 = Gnu16x

Parameter name	Default value	Description
		23 = GnuVxWorks 24 = Gnu68K 25 = DiabData 26 = VisualDSPDos 27 = SH7055 HEW 28 = Metrowerks 29 = Microsoft 30 = ELF 16 Bit 31 = ELF 32 Bit 32 = Fujitsu Softune 3..8 33 = Microware Hawk 34 = TI C6711 35 = Hitachi H8S 36 = IAR HC12 37 = Greenhill Multi 2000 38 = LN308(MITSUBISHI) für M16C/80 39 = COFF settings auto detected 40 = NEC CC78K/0 v35 41 = Microsoft extended 42 = ICCAVR 43 = Omf96 (.m96) 44 = COFF/DWARF 45 = OMF96 Binary (Tasking C196) 46 = OMF166 Binary (Keil C166) 47 = Microware Hawk Plug&Play 48 = UBROF Binary (IAR) 49 = Renesas M32R/M32192 ASCII 50 = OMF251 Binary (Keil C251) 51 = Microsoft standard VC8 52 = Microsoft VC8 (MATLAB DLL) 53 = Microsoft VC8 C++ (MATLAB DLL) 54 = Microsoft VC8 Debug File (pdb) 55 = Microsoft VC++ (MATLAB DLL)
UPDATE_VARIANT_ADDRESSES	0	For variant coded calibration objects also the addresses of the variants are updated. Attention! The updater expects a fix address offset inside the variants of one calibration object. If this is not fulfilled you should not use this setting!
DELETE_WRONG_REFERENCES	0	If this parameter is set, the group and function references to objects removed by filter are removed in destination file.
CREATE_INVALID_CALIBRATION_HANDLES	0	If this parameter is set, found identifier values for CALIBRATION_HANDLE (which are normally not allowed in ASAP2 syntax) will be written to output file
MAP_FILE_NAMES_SUFFIX	“”	The given suffix is added to the names created from linker MAP file
WARNING_UNKNOWN_OBJECT	0	In filter mode: display warnings for all names of filter file which are not found in ASAP2 file.

Parameter name	Default value	Description
CREATE_LINKER_MAP_REFERENCES	0	Create the missing CANape linker map references (IF_DATA CANAPE_EXT) for updated objects
RETURN_CODES_SIMPLE	0	Create only rudimental return codes: 0 = everything OK 1 = no errors, but warnings 2 = errors The concrete error / warning messages can be found in log file.
CHECK_LOCAL_FUNCTION_LISTS	1	Prevent checking local FUNCTION_LIST definitions for removing empty groups and functions
ADDRESS_RANGES	0	Number of address ranges, for which an offset is defined
FILTER_MODE	0	This flag is used to specify which measure and calibration objects of source file are saved in result file. 0: All variables enter the result file 1: Only the variables found in map file enter the result file. 2: Only the variables NOT found in map file enter the result file. 3: Only measure objects enter the result file 4: Only calibration objects enter the result file 5: Only virtual measure and calibration objects enter the result file **Attention!** Modes 3, 4 and 5 can not be used together with group filters
CREATE_ADDRESSES	0	Missing addresses for measurement objects are created using the keyword ECU_ADDRESS.
DELETE_EMPTY_GROUPS	0	Empty groups / functions are removed in result file
CALCULATE_BASE_ADDRESS_FOR_MEASURE	0	Activates calculation of base addresses and offsets by object names of syntax "name[index]"

Parameter name	Default value	Description
WARNING_LEVEL	2	Warning level. If a lower value is specified some warning messages are suppressed. If a higher value is specified some additional messages are given.
IGNORE_MAP_REFERENCES	0	1: The name of the linker map references in IF_DATA CANAPE_EXT is ignored, instead only the object name is used
EXPORT_ONLY_REFERENCED	0	Exports only referenced conversion rules and record layouts into result file
CREATION_RANGES	0	Number of address ranges , for which objects shall be created
CREATE_ARRAYS	0	During generation of objects, arrays in the ASAP2-file are generated for arrays of base-types in the MAP-file. Instead of one object for each entry in an array only one object for the whole array is generated (using MATRIX_DIM). Arrays can be generated for the following MAP-file formats: IEEE, COFF, ELF, UBROF, OMF, PDB.
UPDATE_ADRESSES_TO_ZERO	0	The addresses of objects, which could not be updated are set to zero.
INCLUDE_SAVE_MODE	0	Mode, in which the mapping of ASAP2 objects to included files (via include-statement) is preserved. The objects are stored in their original file after the update. This applies for the following types of objects: GROUP, FUNCTION, CHARACTERISTIC, MEASUREMENT, CONVERSION_METHOD, AXIS_POINTS, CONVERSION_TAB, CONVERSION_VTAB, CONVERSION_VTAB_RANGE, RECORD_LAYOUT
CREATE_MAPCONV_TMP_FILE	0	Forces the creation of the temporary file MAPCONV.TMP which is the ASCII representation of the Map file.
ASAP2_VERSION	160	Optionally, an output file of older ASAP2 format 1.50 or 1.40 can be created
SUPPRESS_UPDATE_OF_VIRTUAL	0	Suppress the address and data type update of virtual objects

[ADDRESS_RANGE_x]

This section contains settings for the xth address range for offset definitions (numbers start with 1).

For all measure and calibration objects inside one address range the configured offset is subtracted from the address available in linker map file.

Parameter name	Default value	Description
START	0	start address of range in linker map file
LENGTH	0	length of the range
OFFSET	0	Offset for all addresses from the linker map file which are inside the defined range. The offset is subtracted. It can be negative, too.

[CREATION_RANGE_x]

This section contains settings for the xth address range for creating new objects (numbers start with 1).

The ASAP2 Updater creates additional parameter and measure objects for all map file symbols inside the configured address range. Symbols which are already used by existing objects, are ignored. For defining the data type of parameter objects the CANape default record layouts are created if not available yet.

If the used map format doesn’t support data types the objects are created with data type UBYTE.

Parameter name	Default value	Description
START	0	start address of range in linker map file
LENGTH	0	length of the range
BYTE_ORDER	“”	Byte order for the objects to be created. Possible values: INTEL and MOTOROLA. If no byte order is specified, the global byte order of the input file is used.
TYPE	“”	Object type to be created. Possible values: MEASURE and PARAMETER.

[COFF]

This section contains settings for the COFF reader. They are relevant only if you set COFF for the MAP format.

Parameter name	Default value	Description
MAP_MAX_ARRAY	16	Maximum array element count
OLD_EXPAND_SYNTAX	0	Expand syntax compatible to CANape 2.0
ENABLE_POINTER	1	Enable pointers
ENABLE_ENUM	1	Enable enumerations
ENABLE_MULTIDIMARRAY	1	Enable multidimensional arrays
CONVERT_DOUBLETOFLOAT	0	Convert DOUBLE objects to float
SIMPLE_ARRAY_VIEW	0	Array representation without leading zeros
COMPATIBLE_MODE	0	Array representation compatible to CANape 3.5
COFF_ADDRESS_MODE	8	Address Mode: 8 = Byte addressing 16 = Word addressing
COFF_FORMAT_MODE	0	Byte order: 0 = Intel 1 = Motorola

[ELF]

This section contains settings for the ELF reader. They are relevant only if you set ELF for the MAP format.

Parameter name	Default value	Description
ELF_STDMAP_RAM_STORAGE	0	Performance optimization (the whole ELF file is read to memory before processing)

Parameter name	Default value	Description
ELF_DEMANGLE_SYMTAB_NAMES	1	Use this option to configure the demangling of mangled variable names Notice: Extended demangling is supported only by DWARF2. It must be used together with option USE_CPP_EXTENSION_DWARF2. 0: Do not demangle symbol names 1: Simple demangling- Demangle only symbol names from debug section Variable names from symbol table are not demangled. Simple demangling is supported by DWARF1 and DWARF2/3. 2: Extended demangling - Demangle symbol names from both debug section and symbol table. The demangling algorithm is optimized to reduce output changes when porting from one compiler to another. 3: Tasking 3.2 Compiler Extension - Demangle symbols mangled by Tasking 3.2 Compiler for TriCore.
ELF_ERROR_ON_AMBIGUOUS	0	Report errors when duplicated variables are found
MAP_MAX_ARRAY	16	Maximum number of variables to be expanded from arrays
ELF_USE_OLD_VERSION	0	Activate old behaviour: Read variables from DWARF and write them down immediately
ELF_USE_CPP_EXTENSION_DWARF2	0	Extended handling of inherited classes, duplicated variables, demangling of symbol names

Parameter name	Default value	Description
ELF_IGNORE_B_MEMBERS_DWARF2	0	Set this option to ignore class, union or struct member variables containing the "__b_" prefix in its name. Possible values are: 0: the option is inactive 1: Ignore member variables beginning with the "__b_" prefix. This option was customized for Diab Data 5.5.1.0 compiler 2: Ignore member variables beginning with the "__b_" prefix. This option was customized for Tricore VX-ToolSet compiler Notice: This option is active only together with option ELF_USE_CPP_EXTENSION_DWARF2=1
ELF_ARRAY_INDEX_FORM	0	Specify form of variable indexes expanded from array 0: array._1_._22_._333_ 1: array[1] [22] [333] 2: array_1_22_333_ 3: array_001_022_333_ 4: array._001_._022_._333_ 5: array._01_._22_._333_ (COFF compatible) 6: array_1_22_333_ 7: array_1__22__333_ 8: array.1.22.333
ELF_DOUBLE_AS_FLOAT	0	Interpretation of the double values as float
ELF_ENUM_AS_INT	1	Interpretation of ENUM variables as INT
ELF_UNDERLINE_PREFIX	0	Create '_' prefix for global variables names and data member names
ELF_UNDERLINE_PREFIX_PREFIX	0	Create '_' prefix for compile unit prefix and sub function prefix, too
ELF_NO_DOT_AFTER_ARRAYITEM	0	Remove '.' placed behind the array item index as separator between index and next name parts
WRITE_PTR_AS_INT or ELF_WRITE_PTR_AS_INT	0	Interpretation of pointer variables as integer
COMPUNIT_NAME_AS_PREFIX	0	Create compile unit name (source file) prefix for STATIC variables, NOT GLOBAL)
COMPUNIT_NAME_AS_PREFIX_ALL	0	Create compile unit name (source file) prefix for STATIC and GLOBAL variables)

Parameter name	Default value	Description
FUNCTION_NAME_AS_PREFIX	0	Create function name prefix for STATIC variables defined in function scope
INVERTED_BIT_FIELDS	0	The bit offset of variables inside bit field structure is inverted 0: No inversion 1: The bit offset is inverted. The flag is VALID ONLY IN MOTOROLA format. 2: **(OBSOLETE)** The bit offset is inverted in both MOTOROLA and INTEL formats. Address is incremented if bit field starts on offset greater than 8'th bit (INCREMENT_BIT_FIELDS_ADDRESS is ignored for this case) 3: **(OBSOLETE)** The bit fields offset is inverted in both MOTOROLA and INTEL formats. Inversion size: Bit offset is inverted by byte size of complete structure. It this information is not available in the elf file the flag is ignored. 4: Inverted bit fields.
INVERTED_BIT_FIELDS_INTEL	0	Invert the bit offset of variables inside bit field structures. This option is valid just for ELF files in INTEL format. If the elf file is in Motorola format the option is ignored
INVERTED_BIT_FIELDS_MIRROR	0	Invert the bit offsets by the size of complete structure class or union
INCREMENT_BIT_FIELDS_ADDRESS	1	For bit-field variables with bit offset greater than 7, the bit offset is devided by 8 and the byte address is incremented. This option is active only when used with any bit inversion flags
ELF_FILE_READ_BUFFER_SIZE	0	Size of the read buffer for elf file to optimize speed
ELF_FORCE_SYMBOL_TABLE	0	Read the information from symbol table 0: Read symbol table only when the DWARF section is not available 1: Ignore DWARF section and read the symbol table 2: Read both the symbol table and DWARF section

Parameter name	Default value	Description
ELF_MAP_VERSION_FLAG	0	Use special version of elf reader compatible with older versions
ELF_SCOPE_SEPARATOR	"._."	string which is used in names of inherited member variables as separator between name of base class and base class member
ELF_ADD_INHERITED_CLASSNAME_PREFIX	1	Insert the inherited class name in front of all inherited data members. The member name and inherited class name will be separated with scope separator 0: Feature is deactivated 1: Insert only the name of the base class in which the member variable is declared 2: If the variable is inherited from multiple (sequence of) base classes, insert the names of all of them in front of variable name 3: same as 1, but should be used for elf file missing DW_Tag_inheritance, where derived class are defined as "__b_" members of base class. (Tasking 3.2 compiler for TriCore) 4: same as 2, but should be used for elf file missing DW_Tag_inheritance, where derived class are defined as "__b_" members of base class. (Tasking 3.2 compiler for TriCore)
ELF_SKIP_UNEXPECTED_TAGS	1	Ignore the tags, which are not expected in context of the previous parent tags
ELF_ENABLE_ARRAY_DOT_DOT	0	0: remove two dots from array indexes created by some compilers 1: turns off automatic removing
ELF_IGNORE_GLOBAL_PADDING	0	Special option for Metrowerks compiler: compile unit tag is directly followed by padding tag, so the rest of the variables in compile unit is not processed. With this option the padding tags which are direct children of the compile units are ignored
ELF_NO_ARRAY_BASEADDR_DATATYPE	0	Create base address of array elements without data type
ELF_ADD_BASE_CLASSNAME_PREFIX	1	Add the name of the base class in front of derived members
ELF_IGNORE_LEADING_UNDERSCORE	0	Remove the leading underscore from all variable names

Parameter name	Default value	Description
ELF_LOG_FILE	0	Create a log file (dump) from the processed elf file. This option is available only for .vdebug section (NEC compiler)
ELF_SKIP_NULL_RANGE_ARRAYS	0	For ELF files with multiple definitions of arrays: ignore definitions with size 0
ELF_NO_BASE_ADDRESS	0	Use this option to disable writing addresses of the classes, structures, unions and arrays to the output.
ELF_REPLACE_CLASSNAME_SCOPESEPARATOR	0	Replace occurence of the scope separator '::' also in the name of the class/union/structure type with scope separator defined by option ELF_SCOPE_SEPARATOR Notice: This option is relevant only for DWARF2 reader.
ELF_TRY_RESOLVE_MEMADDR_FROM_SYMBOLTABLE	0	For static member variables without valid address information in debug section the ELF reader tries to find the symbol with an extended symbol name in symbol table. This option is supported only by DWARF2/DWARF3 and must be used together with ELF_USE_CPP_EXTENSION_DWARF2

[UBROF]

This section contains settings for the UBROF reader. They are relevant only if you set UBROF for the MAP format.

Parameter name	Default value	Description
UBROF_POINTER_AS_VARIABLE	1	Use pointers and references as variables (ADDRESS type), otherwise pointers and references will be ignored
MAP_MAX_ARRAY	16	Maximum number of variables to be expanded from arrays
UBROF_WRITE_BUFFER_SIZE	32767	Buffer size for reader output (Minimal 2048)
UBROF_SCOPE_SEPARATOR	“._.”	string which is used in names of inherited member variables as separator between name of base class and base class member

Parameter name	Default value	Description
UBROF_ARRAY_INDEX_FORM	0	Specify form of variable indexes expanded from array 0: array._1_._22_._333_ 1: array[1] [22] [333] 2: array_1_22_333_ 3: array_001_022_333_ 4: array._001_._022_._333_ 5: array._01_._22_._333_ (COFF compatible)
UBROF_NO_DOT_AFTER_ARRAYITEM	0	Remove '.' placed behind the array item index as separator between index and next name parts
UBROF_SKIP_DUPLICATED_ADDRESSES	1	If found more than one variable with same address in the UBROF file, only the first is used. The others are ignored.
UBROF_MAKE_LOGFILE	0	Generate Log file for with UBROF information
UBROF_INVERTED_BIT_FIELDS	0	Invert bit fields positions inside defining structure
UBROF_INCREMENT_BIT_FIELDS_ADDRESS	1	For bit-field variables with bit offset greater than 7, the bit offset is devided by 8 and the byte address is incremented. This option is active only when used with any bit inversion flags

[IEEE]

This section contains settings for the IEEE reader. They are relevant only if you set IEEE for the MAP format.

Parameter name	Default value	Description
MAP_MAX_ARRAY	16	Maximum array element count
MAP_FIX_TASKING_TYPE_ERROR	1	Fix Tasking IEEE output type error
MAP_ARRAY_LEADING_ZERO	1	Generate array element names with leading zeros
MAP_IGNORE_LEADING_UNDERSCORE	1	Ignore leading underscores of exported symbol names
MAP_IGNORE_INSTRUCTION_ADDRESS	1	Ignore exported symbols with type index 15 (instruction address)
MAP_CASE_SENSITIVE_LINKER	1	Case sensitive linker

Parameter name	Default value	Description
MAP_BIT_SIZE_OF_INTEGER	0	Bit size of integer (0, 8, 16, 32)
MAP_BYTE_SIZE_OF_ENUMERATION	-1	Byte size of enumeration
MAP_DIRECT_BIT_BASE_ADDR	0xFD00	Base address for direct addressable bits
MAP_DIRECT_BIT_BYTE_OFFSET_MASK	0xFFF8	Byte mask for direct addressable bits
MAP_DIRECT_BIT_BIT_OFFSET_MASK	0x0007	Bit mask for direct addressable bits

[OMF]

This section contains settings for the OMF reader. They are relevant only if you set OMF for the MAP format.

Parameter name	Default value	Description
MAP_MAX_ARRAY	16	Maximum array element count
OMF_ARRAY_INDEX_FORM	0	Specify array index form 0: array._1_._22_._333_ 1: array[1][22][333] 2: array_1_22_333_ 3: array_001_022_333 4: array._001_._022_._333_ 5: array._01_._22_._333_ (COFF compatible)
OMF_LOG_FILE	0	Create log file
OMF_MCI_IN_ADDR	1	Encode the memory class index in MSB of the variable address (only for ECU with 24bit addressing)

[GREENHILL]

This section contains settings for the GREENHILL reader. They are relevant only if you set GREENHILL for the MAP format.

Parameter name	Default value	Description
GHS_REMOVE_UNDERLINES	0	Remove leading '_' in symbols names
GHS_INCLUDE_RODATA	0	If this parameter is set, the section „.rodata“ of MAP file is read, otherwise it is ignored

[PDB]

This section contains settings for the PDB reader. They are relevant only if you set PDB for the MAP format.

Parameter name	Default value	Description
MAP_MAX_ARRAY	16	Maximum number of variables to be expanded from arrays
PDB_ARRAY_INDEX_FORM	0	Specify form of variable indexes expanded from array 0: array._1_._22_._333_ 1: array[1] [22] [333] 2: array_1_22_333_ 3: array_001_022_333_ 4: array._001_._022_._333_
PDB_UNDERLINE_PREFIX	0	Create '_' prefix for global variables names and data member names
PDB_REMOVE_LEADING_UNDERSCORE	0	Remove the leading underscore from all variable names
PDB_MAX_TYPE_IN_DETAIL	10	Use this option to get a deeper resolution of sub structures

[DATATYPES]

This section contains settings for the update of data types.

Parameter name	Default value	Description
ENABLE_UPDATE	0	Enables the update of data types for measurement objects and scalar calibration objects (VALUEs)
UPDATE_BASE_TYPES	1	Updates the base data types if the data type update is enabled.
UPDATE_BIT_OFFSETS	0	Updates the bit masks if the data type update is enabled.
DELETE_ADDRESS_OFFSETS	0	If this parameter is set, the address offsets in IF_DATA CANAPE_EXT will be deleted for those objects whose data type has changed. This could be used if objects with a big bit offset (larger than 7) are described in the source ASAP2 file by a combination of an address offset and a bit offset ≤ 7.

Parameter name	Default value	Description
DELETE_ALL_ADDRESS_OFFSETS	0	If this parameter is set, the address offsets in IF_DATA CANAPE_EXT will be deleted for **ALL** objects, independent of changed data type or address.
UPDATE_CURVES_AND_MAPS	0	To be set to 1 to enable the data type update also for curves, maps and common axes

3.3 Exit Codes

The success of the update process can be determined by the exit code of the program.

0 no errors, no warnings

100 no errors, but warnings in the log file

the possible warnings:

- Syntax error in an A2L file (later leads to program termination, but the file is still parsed at the end)
- Hybrid of interface components from both ASAP2 V1.0 and ASAP2 V >= 1.20
- Syntax error in the address file (incorrect line is ignored)
- Label from the address file not found

501 Incorrect command line.

502 Log file cannot be opened.

503 More than one source file was entered.

504 No source file was entered.

506 Syntax error in the ASAP2 source file.

507 Output file cannot be written.

508 More than one Output file was entered.

509 No Output file was entered.

510 File with command line parameters cannot be read

511 No address file was entered.

514 Parameters -X and -F cannot be combined.

515 More than one MAP file given

516 Path for INI file not unique

600 Not enough memory space.

3.4 Interface to MAP Readers

The interface between the MAP readers and the updater is the file “MAPCONV.TMP”. This file is written in current working directory by the selected MAP reader and read by the updater. It is an ASCII file with MAP format “Standard” and contains a list of all variables of the MAP file together with their address and (if available) data type. Each line of that file has the following syntax:

<name> <address> [<data type> [<bit offset>]]

with

<name>: symbol name from MAP file

<address>: address string in hexadecimal form, optionally together with address extension in decimal form (e.g. 3:0F815)

<data type>: can be one of BYTE, UBYTE, WORD, UWORD, DWORD, UDWORD, FLOAT, DOUBLE, BITx, UBITx where x is the bit number

<bit offset>: optional bit offset for single bit values or bit slices

3.5 Examples

The following examples show the usage of ASAP2 Updater. You find all necessary files for the examples in your ASAP2 Tool Set delivery.

The result file is generated from the ASAP2 input file *Demo.a2l* and the map file *demo.map*.

Demo.a2l contains the measure signals ***channel1*** and ***channel2*** – each with address 0, data type UBYTE and no reference to a linker map symbol. Furthermore, it contains the parameter ***amplitude*** with address 0, data type double and reference to a map symbol ***__ampl***.

The map file *Demo.map* contains 5 entries ***channel1*** – ***channel4*** and ***amplitude*** – each with data type double.

3.5.1 Sample 1

Call ASAP2 Updater via the command line:

```
ASAP2Updater -i demo.a2l -a demo.map -o result1.a2l -l log.txt
```

And use the following settings in initialization file

```
[OPTIONS]
MAP_FORMAT = 0
FILTER_MODE = 0
IGNORE_MAP_REFERENCES = 0
CREATION_RANGES = 0

[DATATYPES]
ENABLE_UPDATE = 1
UPDATE_BASE_TYPES = 1
```

The result file *result1.a2l* then contains all original objects ***channel1***, ***channel2*** and ***amplitude***. The address and data type of ***channel1*** and ***channel2*** are updated whereas ***amplitude*** keeps unchanged because the linked map symbol ***__ampl*** (which has priority against the object name) is not found in map file.

3.5.2 Sample 2

Change the settings in inititialization file to

```
FILTER_MODE = 4
IGNORE_MAP_REFERENCES = 1
```

and call the ASAP2 Updater via the same command line:

```
ASAP2Updater -i demo.a2l -a demo.map -o result2.a2l -l log.txt
```

Because of the option `FILTER_MODE=4` all measure objects are now removed and the result file *result2.a2l* contains only the parameter ***amplitude*** which has now an updated address because of the option `IGNORE_MAP_REFERENCES=1`

3.5.3 Sample 3

Change the settings in inititialization file to

```
CREATION_RANGES = 1
```

add the section

```
[CREATION_RANGE_1]
START  = 0x1000
LENGTH = 0x100
TYPE   = MEASURE
```

and call the ASAP2 Updater via the same command line:

```
ASAP2Updater -i demo.a2l -a demo.map -o result3.a2l -l log.txt
```

Because of the defined creation range the result file *result3.a2l now* contains two additional measure objects ***channel3*** and ***channel4***.

4 ASAP2 Merger

ASAP2 Merger merges multiple ASAP2 files into a common ASAP2 file. One of the source files must be specified as the Master; the other source files are Slaves. All information is taken from the Master into the resulting output file. The only information taken from the slaves are the variables and adjustable values, conversion formulas, functions, groups and record layouts. For each slave you have to configure which slave modules are to be appended to which master module. The objects in the result file can optionally receive a suffix, which indicates from which source file these objects originate.
The interface-specific parts of the ASAP2 file are interpreted and rewritten to the output file based on the given meta-language. All IF_DATA types are supported.

4.1 Command Line Parameters

The following command line parameters can be entered when calling the program:

-M <name>	Name of the ASAP2 master file
-L <name>	Name of the log file for warnings and error messages. If no name is entered for this parameter, the messages are output to the screen.
-S <name>	Name of an ASAP2 slave file and optional:
[<number>	• Number of modules for this file which are to be merged into the master file
(<name1> <name2>)*]	• Module assignment to the master file (name1 = MODULE name in the slave file, name2 = MODULE name in the master file)
	If no number and assignment is specified the first module of the slave file will be merged into the first module of the master file. So, if both the master file and the slave file contain only one MODULE, the number and assignment do not need to be specified. The –S parameter can be used multiple in command line to merge more than two files at once.
-O <name>	Name of the ASAP2 file to be generated
-P <name>	Name for the initializing file, if this file is not located in working directory or if its name isn't MERGER.INI
-@<name>	Name of a text file with all command line parameters. .

Attention
There must be NO space characters between @ and the file name.

Example
Call e.g.:
`ASAP2Merger -@CMDLINE.TXT`

Note
The text file may contain comment lines: lines starting with a ';' are ignored.

Calling example:

```
ASAP2Merger -M Master.a2l -S slave1.a2l -S slave2.a2l 2 DIM1 DIM
DIM2 DIM -O Result.a2l
```

or:

```
ASAP2Merger -@CMDLINE.TXT
```

with Cmdline.txt file:

```
-M Master.a2l
-S slave1.a2l
-S slave2.a2l 2 DIM1 DIM DIM2 DIM
-O Result.a2l
```

When this call is made the Slave1 MODULE DIM and the modules DIM1 and DIM2 of Slave2 are appended to the Master's MODULE DIM (see Fig. 1).

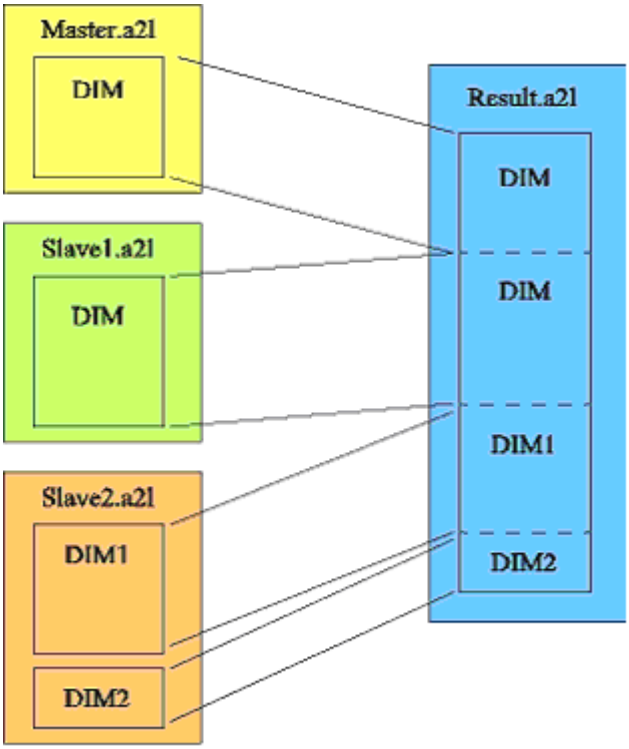

Fig. 1: Merging A Master File With Two Slave Files

4.2 Initialization File

The initialization file MERGER.INI usually is expected in current directory. Alternatively, a different file name – also in different directory can be set in the command line. In the initialization file the behaviour of the merger can be controlled.

The following list contains all parameters of the INI file, their possible values and their default value. The names inside brackets are the sections for the parameters. The order of the parameters inside the sections and the order of the sections are no odds. Furthermore the names of the parameters and sections are not case sensitive.

[OPTIONS]

This section contains common settings.

Parameter name	Default value	Description
DISABLE_SUFFIXES	0	Disables suffix generation
AVOID_MULTIPLE_OBJECTS	0	If there are objects with equal names, only the corresponding object of the first source file in command line is accepted for the result file.

Parameter name	Default value	Description
CREATE_IF_DATA_FOR_SLAVES	0	IF_DATA module information of ALL input files is used for the result file. Without this option only the IF_DATA module information of the master file is used. Precondition: the corresponding AML definition must be available in master file.
CREATE_INVALID_CALIBRATION_HANDLES	0	If this parameter is set, found identifier values for CALIBRATION_HANDLE (which are normally not allowed in ASAP2 syntax) will be written to output file
MERGE_GROUP_CONTENTS	0	0: Group resp. function contents are not merged 1: For groups resp. functions with equal names and comments the contents (sub groups, sub functions and referenced objects) are merged. 2: Additionally the annotations of the slave group/function are appended to the master group/function. Merge of contents is only possible with disabled suffix generation
CHECK_FOR_DUPLICATE_NAMES	1	0: No check for duplicate names (better performance)
CREATE_IF_DATA_CANAPE	0	Create the AML for CANAPE_EXT in output file if not exists in master file
RETURN_CODES_SIMPLE	0	Create only rudimental return codes: 0 = everything OK 1 = no errors, but warnings 2 = errors The concrete error / warning messages can be found in log file.
GROUP_FOR_SLAVE_x	“”	Add all objects of slave x to master group with given name. Master group is created if it doesn't exist. Numbering of slaves start with 1
ASAP2_VERSION	160	Optionally, an output file of older ASAP2 format 1.50 or 1.40 can be created
IGNORE_GROUP_COMMENTS	0	If group contents are merged (see option MERGE_GROUP_CONTENTS) the group comments are ignored - i.e. groups/functions with equal names can be merged although their comments are different. The merged group then gets the comment corresponding to the first source file in command line.

Parameter name	Default value	Description
PARENT_FOR_SLAVE_x	""	The complete object hierarchy of slave file number x is inserted into the master group with given name. All measure and calibration objects of the slave without group assignment and all slave groups resp. functions without parent group in the slave file are assigned to the master parent group. The Master group is created if it doesn't exist. Numbering of slaves start with 1

4.3 Exit Codes

The success of the merge process can be determined by the exit code of the program.

0 no errors, no warnings

100 no errors, but warnings in the log file

the possible warnings:

- Syntax error in an A2L file (later leads to program termination, but the file is, if possible, still parsed at the end)
- MODULE with the entered name not found
- One MODULE from a slave file is not assigned to a Master MODULE (not present in the result file)
- Hybrid of interface components from both ASAP2 V1.0 and ASAP2 V >= 1.20

501 Corrupt command line data.

502 The log file cannot be opened.

503 More than one master file was entered.

504 No master file was entered.

505 No slave file was entered.

506 Syntax error in one of the ASAP2 source files.

507 Result file cannot be written.

508 More than one result file was entered.

509 No result file was entered.

510 File with command line parameters cannot be read.

600 Not enough memory locations

4.4 Examples

The following example shows the usage of ASAP2 Merger. You find all necessary files for the examples in the demo directory of your ASAP2 Tool Set installation.

The result file is generated by merging the files *Master.a2l, Slave1.a2l* and *Slave2.a2l*.

Slave1.a2l contains two measure signals ***channel1*** and ***channel2*** which are assigned to the group ***SlaveGroup***.

Slave2.a2l contains two measure signals ***channel3*** and ***channel4*** which are assigned to a group with the same name ***SlaveGroup***.

Master.a2l contains the measure signal ***channel5*** which is assigned to the group ***MasterGroup***. Furthermore it contains an empty group ***SlaveGroup*** and also a measure signal ***channel1*** with modified comment.

4.4.1 Sample 1

Call ASAP2 Merger via the command line:

```
ASAP2Merger -m master.a2l -s slave1.a2l -s slave2.a2l -o result1.a2l -l log.txt
```

And use the following settings in initialization file

```
[OPTIONS]
DISABLE_SUFFIXES = 1
AVOID_MULTIPLE_OBJECTS = 1
MERGE_GROUP_CONTENTS = 1
```

The result file *result1.a2l* then contains 5 signals: ***channel1***, ***channel2***, ***channel3***, ***channel4*** and ***channel5*** and the two groups ***SlaveGroup*** and **MasterGroup**.

Because of the option `DISABLE_SUFFIXES=1` the original group names are not modified and their contents can be merged: ***SlaveGroup*** contains all signals ***channel1*** – ***channel4***. ***MasterGroup*** contains only ***channel5***.

Because of the option `AVOID_MULTIPLE_OBJECTS=1` the result file contains only one ***channel1*** signal: the one from master file with modified comment is used because master.a2l was the first file named in command line.

4.4.2 Sample 2

Now change the settings in initialization file

```
DISABLE_SUFFIXES = 0
AVOID_MULTIPLE_OBJECTS = 0
MERGE_GROUP_CONTENTS = 0
```

And call the ASAP2 Merger via the same command line:

```
ASAP2Merger -m master.a2l -s slave1.a2l -s slave2.a2l -o result2.a2l -l log.txt
```

Now, because of the option `DISABLE_SUFFIXES=0`, for all objects except measure and calibration objects suffixes are created and the result file contains the groups ***MasterGroup_M***, ***SlaveGroup_S1***, ***SlaveGroup_S2*** and ***SlaveGroup_M*** which contain only their originally assigned objects.

Furthermore, because of the option `AVOID_MULTIPLE_OBJECTS=0` the result file contains two ***channel1*** signals which is written as a warning to the logfile.

5 ASAP2 Comparer

The ASAP2 Comparer reads in two ASAP2 files, compares their contents and creates messages for the differences found between the two files in a log file or a distinct result file. The formatting of the ASAP2 files, any comments and the order of the objects are irrelevant for the comparison.

Per default the whole files are compared, but it is possible to reduce the data to be compared by special settings in the initialization file COMPARER.INI.

5.1 Command Line Parameters

When the program is called the following command line parameters can be entered:

-A <name> Name and path of the first ASAP2 input file

-B <name> Name and path of the second ASAP2 input file

-L <name> Name and path of the log file for warnings and error messages

-I <name> Name and path for the initialization file if this file is not located in working directory or if its name is not COMPARER.INI

-D <name> Name and path for the result file which will contain all found differences. If this parameter is omitted, the console will be used.

5.2 Initialization file

The initialization file COMPARER.INI usually is expected in current working directory. Alternatively, a different file name – also in different directory can be set in the command line. In the initialization file the behaviour of the compare tool can be controlled.

The following list contains all parameters of the INI file, their possible values and their default value. The names inside brackets are the sections for the parameters. The order of the parameters inside the sections and the order of the sections are no odds. Furthermore the names of the parameters and sections are not case sensitive.

[OPTIONS]

This section contains common settings for the comparison.

Parameter name	Default value	Description
BITMASK_VS_DATATYPE	1	This option configures the comparison of bitmasks for measure and calibration objects. If set, missing bitmasks, empty bitmasks (0) and full bitmasks (e.g. 0xFF for data type UBYTE) are considered as equal.

Parameter name	Default value	Description
		For example, no difference is reported with this option if a measure object of data type SWORD has no bitmask in the first ASAP2 file and bitmask 0xFFFF in the second ASAP2 file.
IGNORE_ADDRESS_FOR_VIRTUAL	0	If this option is set, the optionally available addresses of virtual measure objects are ignored.
USE_ALIGNMENT_DEFAULT	1	With this option, the default alignment according to ASAP2 specification is used for the comparison if no alignment is specified at a record layout and / or in the device settings. For example, no difference is reported with this option if one ASAP2 file has specified Float-Alignment as 4 and the other ASAP2 file has not specified Float Alignment at all.
USE_ARRAY_SIZE_DEFAULT	1	If this option is set, a missing ARRAY_SIZE flag for a measure object is interpreted as ARRAY_SIZE 1
USE_BYTE_ORDER_DEFAULT	1	With this option, the default byte order according to ASAP2 specification is used for comparison if no byte order is specified for a measure or calibration object resp. in the device settings.
USE_DEPOSIT_DEFAULT	1	With this option, the default deposit of the device is used for comparison if no DEPOSIT is specified for an axis. If no deposit is specified for the device, the default value ABSOLUTE is used for comparison.
USE_EXTENDED_LIMITS_DEFAULT	1	If this option is set, the Comparer uses the default limits of an calibration object for comparison when EXTENDED_LIMITS is missing.
USE_MATRIX_DIM_DEFAULT	1	With this option, a missing MATRIX_DIM keyword for a measure or calibration object is interpreted as MATRIX_DIM 1 1 1
MATRIX_DIM_VS_ARRAY_SIZE	0	With this option the obsolete keyword ARRAY_SIZE is converted into MATRIX_DIM and used for comparison. For example, no difference is reported with this option, if a measure object has ARRAY_SIZE 7 in one ASP2 file and MATRIX_DIM 7 1 1 in the other one.

Parameter name	Default value	Description
IGNORE_ADDRESS_CHANGE	0	If this option is set, measure and calibration objects are considered as equal, if only the address has changed.
USE_FORMAT_DEFAULT	1	With this option the format settings of the corresponding conversion method are used for comparison if no FORMAT is specified for a measure or calibration object.
USE_UNIT_DEFAULT	1	If no PHYS_UNIT is specified for a measure or calibration object, the unit of the referenced conversion method is used for comparison.
IGNORE_READONLY_FOR_FIX_AXIS	1	With this option, READ_ONLY settings for fix axes are ignored for comparision.
DIFF_FILE_FORMAT	1	Format of the result file which contains all found differences. This parameter is operative only if a distinct result file has been specified. Possible values are: 1 – Logfile: One line is written to the result file for each found difference. 2 – CSV: The differences are written as a ';'-separated table which can be opened with a spreadsheet application. 3 – XML: The differences are written using a simple XML-Syntax which is convenient for further processing.
DIFF_FILE_XML_STYLESHEET		In case of XML format of the result file, this parameter can be used to specify a path to a xsl-stylesheet. This is a textual option. **Note:** the Comparer Demo contains the stylesheet COMPARISON.XSL which can be used to format the XML file as HTML page. Just open the generated XML file with a web browser.
DIFF_FILE_GROUPING	1	This option specifies the grouping of entries in the result file. Possible values are: 0 – no grouping 1 – group by object type of the differences

[FILTER]

This section contains settings to reduce the data to compare.

Parameter name	Default value	Description
COMPARE_IF_DATA	1	With this option active the IF_DATA sections will be compared, too. Otherwise they are ignored for comparison.
USE_REGULAR_EXPRESSIONS	1	In the FILTER section you can define filters for the several object types to reduce the compare data. For this name filters you can use either regular expressions or wildcards. If the option USE_REGULAR_EXPRESSIONS is active, the name filters are interpreted as regular expressions – otherwise as wildcard expressions.
MEASUREMENT_TO_COMPARE		Here you can define a filter for measure objects. Only measure objects whose name matches this filter are compared. Default value is ***** if you configured wildcard expressions and. **.*** if you configured regular expressions. If you want to ignore all measure objects for comparision, this filter has to be set empty.
CHARACTERISTIC_TO_COMPARE		Here you can define a filter for characteristic objects. Only characteristic objects whose name matches this filter are compared. Default value is ***** if you configured wildcard expressions and. **.*** if you configured regular expressions. If you want to ignore all characteristic objects for comparision, this filter has to be set empty.
AXIS_PTS_TO_COMPARE		Here you can define a filter for axis objects. Only axis objects whose name matches this filter are compared. Default value is ***** if you configured wildcard expressions and. **.*** if you configured regular expressions. If you want to ignore all axis objects for comparision, this filter has to be set empty.

Parameter name	Default value	Description
CONVERSION_METHOD_TO_COMPARE		Here you can define a filter for conversion methods. Only conversions whose name matches this filter are compared. Default value is ***** if you configured wildcard expressions and. **.*** if you configured regular expressions. If you want to ignore all conversion methods for comparision, this filter has to be set empty.
CONVERSION_TABLE_TO_COMPARE		Here you can define a filter for conversion tables. Only tables whose name matches this filter are compared. Default value is ***** if you configured wildcard expressions and. **.*** if you configured regular expressions. If you want to ignore all conversion tables for comparision, this filter has to be set empty.
GROUP_TABLE_TO_COMPARE		Here you can define a filter for groups. Only groups whose name matches this filter are compared. Default value is ***** if you configured wildcard expressions and. **.*** if you configured regular expressions. If you want to ignore all groups for comparision, this filter has to be set empty.
FUNCTION_TO_COMPARE		Here you can define a filter for functions. Only functions whose name matches this filter are compared. Default value is ***** if you configured wildcard expressions and. **.*** if you configured regular expressions. If you want to ignore all functions for comparision, this filter has to be set empty.
RECORD_LAYOUT_TO_COMPARE		Here you can define a filter for record layouts. Only layouts whose name matches this filter are compared. Default value is ***** if you configured wildcard expressions and. **.*** if you configured regular expressions. If you want to ignore all record layouts for comparision, this filter has to be set empty.
FRAME_TO_COMPARE		Here you can define a filter for frames. Only frames whose name matches this filter are compared. Default value is ***** if you configured wildcard expressions and. **.*** if you configured regular expressions. If you want to ignore all frames for comparision, this filter has to be set empty.

Parameter name	Default value	Description
UNIT_TO_COMPARE		Here you can define a filter for units. Only units whose name matches this filter are compared. Default value is ***** if you configured wildcard expressions and. **.*** if you configured regular expressions. If you want to ignore all units for comparision, this filter has to be set empty.

5.3 Exit-Codes

How successful the compare process was can be determined by the exit code of the program.

0 No errors, no warnings

1 No errors, but warnings in log file

2 Error messages in log file

Get more Information!

Visit our Website for:

> News

> Products

> Demo Software

> Support

> Training Classes

> Addresses

www.vector-worldwide.com