

ASAP2 Tool-Set

User Manual

Version 16.0
English

Imprint

Vector Informatik GmbH
Ingersheimer Straße 24
D-70499 Stuttgart

Vector reserves the right to modify any information and/or data in this user documentation without notice. This documentation nor any of its parts may be reproduced in any form or by any means without the prior written consent of Vector. To the maximum extent permitted under law, all technical data, texts, graphics, images and their design are protected by copyright law, various international treaties and other applicable law. Any unauthorized use may violate copyright and other applicable laws or regulations.

© Copyright 2021, Vector Informatik GmbH. Printed in Germany.
All rights reserved.

Contents

1	Introduction	4
1.1	Overview	4
1.2	About This User Manual	6
1.2.1	Warranty	7
1.2.2	Support	7
1.2.3	Trademarks	7
2	Configuration of the Command Line Tools	8
2.1	Initialization File	8
2.2	Cross-Tool General Options	8
2.3	Cross-Tool Tolerance Settings	10
3	ASAP2 Creator	13
3.1	Functionality	14
3.2	Command Line Parameters	14
3.3	Exit Code	14
3.4	Initialization File	14
3.4.1	[FILES]	15
3.4.2	[OPTIONS]	15
3.4.3	[SYNTAX_TOLERANCE]	17
3.5	Syntax for Comments in C Code	18
3.5.1	Object Name and Data Type	18
3.5.2	Conversion Rules	21
3.5.3	Comment	25
3.5.4	Display Name	25
3.5.5	Display Color	26
3.5.6	Linker Map Reference	27
3.5.7	Group Assignment	29
3.5.8	Dimensions	32
3.5.9	Record Layout	36
3.5.10	Curves and Maps	37
3.5.11	Hard-coded Addresses and Address Extensions	41
3.5.12	Default Events	42
3.5.13	Structures	45
3.5.14	Overwrite Element Properties	53
3.5.15	References to Structure Components	55
3.5.16	Byte Order	57
3.6	Generating Prefixes for Object Names	57
3.7	Usage of Macros	59
3.8	Variant Coding	60
3.9	Format Description in Backus-Naur Form	61
4	ASAP2 Updater	69
4.1	Functionality	69
4.2	Command Line Parameters	70
4.3	Exit Code	70

4.4	Initialization File	71
4.4.1	[OPTIONS]	71
4.4.2	[ADDRESS_RANGE_x]	77
4.4.3	[CREATION_RANGE_x]	77
4.4.4	[EPK]	79
4.4.5	[COFF]	80
4.4.6	[ELF]	80
4.4.7	[IEEE]	87
4.4.8	[GREENHILL]	88
4.4.9	[PDB]	88
4.4.10	[DATATYPES]	89
4.4.11	[CAL_PARAM_GROUPS]	89
4.4.12	[SYNTAX_TOLERANCE]	90
4.5	Warning Levels	91
4.6	Interface to MAP Readers	91
4.7	Examples	92
5	ASAP2 Merger	94
5.1	Functionality	94
5.2	Command Line Parameters	94
5.3	Exit Code	95
5.4	Initialization File	95
5.4.1	[OPTIONS]	96
5.4.2	[SLAVE_X]	99
5.4.3	[TYPE_SPECIFIC_CONFLICT_SOLUTION]	100
5.4.4	[SYNTAX_TOLERANCE]	101
5.5	Examples	102
6	ASAP2 Comparer	104
6.1	Functionality	104
6.2	Command Line Parameters	104
6.3	Thesaurus Files	105
6.4	Exit Code	106
6.5	Initialization File	106
6.5.1	[OPTIONS]	106
6.5.2	[FILTER]	110
6.5.3	[SYNTAX_TOLERANCE]	112
7	ASAP2 Checker	114
7.1	Functionality	114
7.2	Command Line Parameters	114
7.3	Exit Code	115
7.4	Initialization File	115
7.4.1	[OPTIONS]	115
7.4.2	[SYNTAX_TOLERANCE]	116
7.4.3	[EXTENDED_CHECK]	117
7.4.4	[AUTO_CORRECTION]	121
7.5	Error Numbers	122
8	ASAP2 Modifier	126
8.1	Functionality	126

8.2	Command Line Parameters	126
8.3	Exit Code	127
8.4	Initialization File	127
8.4.1	[OPTIONS]	127
8.4.2	[MODIFICATIONS]	129
8.4.3	[SYNTAX_TOLERANCE]	134
8.4.4	[FILTER]	134
8.4.5	[OPTIMIZATION]	140
8.4.6	[EXCEL_UPDATE]	142
8.4.7	[CREATE_MEASURE_ARRAYS]	148
8.4.8	[CREATE_STRUCTURES]	150
8.4.9	[CREATE_XCP_DAQ_EVENTS]	152
8.5	Warning Level	152
8.6	CANape DBU Format	153
8.6.1	Format description in BNF	153
8.7	JSON File for XCP Event Description	154
9	ASAP2 Studio	159
9.1	Functionality	159
9.2	Structure of the User Interface	160
10	Addresses	162

1 Introduction

In this chapter, you will find the following information:

1.1	Overview	page 4
1.2	About This User Manual Warranty Support Trademarks	page 6

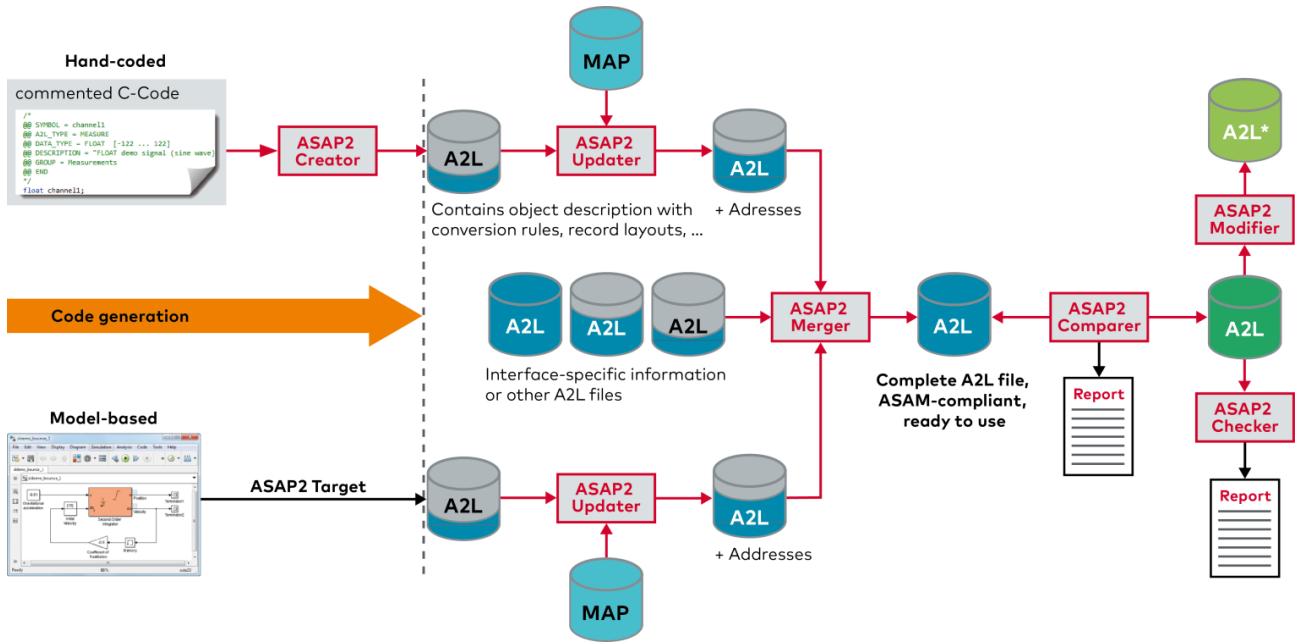
1.1 Overview

Components

The **ASAP2 Tool-Set** is used to create and modify ECU description files in ASAP2 format (A2L) which are a prerequisite for using CCP and XCP as a measurement and calibration protocol. It consists of the **ASAP2 Studio** and the six command line programs **ASAP2 Creator**, **ASAP2 Updater**, **ASAP2 Merger**, **ASAP2 Comparer**, **ASAP2 Checker** and **ASAP2 Modifier**.

Achievements of the ASAP2 Tool-Set

- > **ASAP2 Creator**: Automatically create A2L files based on special comments in the C code
- > **ASAP2 Updater**: Update address and structure information in an A2L file based on the contents of Linker MAP and debug files
- > **ASAP2 Merger**: Merge multiple A2L fragments into a common A2L file
- > **ASAP2 Comparer**: Comparison of two A2L files and export of the comparison result to different report formats
- > **ASAP2 Checker**: Check A2L files for syntactic and semantic errors
- > **ASAP2 Modifier**: Targeted filtering, modifying, supplementing and optimizing of A2L files
- > **ASAP2 Studio**: Creation, visualization and editing of A2L files, as well as import and export to foreign formats



What's new in the ASAP2 Tool-Set

The **ASAP2 Tool-Set** has been expanded in version 16.0 by the following functionalities:

- > Improved update of structures allows quick and easy adaptation of the A2L file when changes are made to the ECU code.
- > Enhanced configuration options for the **ASAP2 Merging** allow more flexible merging of A2L fragments.
- > Additional columns during Excel import facilitate data transfer from third-party formats.
- > The newly integrated Comparer in **ASAP2 Studio** allows efficient comparison of A2L files and visualization and filtering of differences.
- > **ASAP2 Studio** now supports import from DBU format for improved compatibility with older **CANape** versions.
- > For documentation purposes, comparison results can be exported to various file formats.
- > With the new detail window for MAP symbols, the content of a MAP file can be visualized even better.

Functionality

All tools run under **Windows 7 (SP1)** and **Windows 8.1/10**. They 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 automated build 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 A2L files of all versions 1.x released by ASAM, including the current version 1.71. The generated A2L files have format ASAP2 1.71 by default.

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

VECTOR.INI

The file `VECTOR.INI` includes the setting for the language of error messages and warnings.

This applies across all tools from the **ASAP2 Tool-Set**.

If this file is missing, the appropriate default settings are used. For the language, this is English.

VCUSTOM.INI

The file `VCUSTOM.INI` includes path information corresponding to the installation.

Essentially, the path information for the `VECTOR.INI` is found here.

Since this must be editable by the user to change the language, it must not be in the subdirectory `Bin` because there is (depending on the selected installation directory) possibly no write permission.

If the file `VCUSTOM.INI` is missing, the file `VECTOR.INI` is searched from the tools by default in the subdirectory `Bin`.

1.2 About This User Manual

To Find information quickly






This user manual provides you with the following access help:

- > At the beginning of each chapter you will find a summary of the contents
- > The header shows in which chapter of the manual you are
- > The footer shows the version of the manual

Conventions

In the two tables below, you will find the notation and icon conventions used throughout the manual.

Style	Utilization
bold	Fields/blocks, user/surface interface elements, window- and dialog names of the software, special emphasis of terms. [OK] Push buttons in square brackets File Save Notation for menus and menu entries
MICROSAR	Legally protected proper names and marginal notes.
Source Code	File and directory names, source code, class and object names, object attributes and values
Hyperlink	Hyperlinks and references.
<Ctrl>+<S>	Notation for shortcuts.

Symbol	Utilization
	This icon indicates notes and tips that facilitate your work.
	This icon warns of dangers that could lead to damage.
	This icon indicates examples.
	This icon indicates step-by-step instructions.
	This icon indicates that something has changed.

1.2.1 Warranty

Restriction of warranty

We reserve the right to modify the contents of the documentation or the software without notice. Vector disclaims all liabilities for the completeness or correctness of the contents and for damages which may result from the use of this documentation.

1.2.2 Support

Need support?

You can get through to our hotline by calling
+49 (0)711 80670-200
or you can send a problem report to support@vector.com.

1.2.3 Trademarks

Protected trademarks

All brand names in this documentation are either registered or non-registered trademarks of their respective owners.

> **Windows, Windows 7 and Windows 8/8.1/10** are trademarks of the Microsoft Corporation.

2 Configuration of the Command Line Tools

In this chapter, you will find the following information:

2.1	Initialization File	page 8
2.2	Cross-Tool General Options	page 8
2.3	Cross-Tool Tolerance Settings	page 10

2.1 Initialization File

- Behavior** With the initialization file (INI file) the behavior of each tool can be controlled.
- ANSI encoding** The INI file must use ANSI encoding.
Detecting the initialization file uses UTF8 encoding with a Byte Order Mark (BOM), will abort the program, because in this case the file content could be interpreted incorrectly.
- Parameter** The following lists for the possible INI options contain all parameters, their possible values and their default value. The names inside brackets are the sections for the parameters. The sequence of the parameters inside the sections and the order of the sections in the INI file are irrelevant. Furthermore, the names of the parameters and sections are not case sensitive.

2.2 Cross-Tool General Options

- General Options** This section describes those options that can be used for all command-line tools in the same way. They are listed again under [OPTIONS] of those tools to which they apply but are not described in detail again.
- Navigation help** The different descriptions of the options are linked from the tools to this page. To navigate back again, press **<Alt>+<Left Arrow>**.

Parameter	Default Value	Description
ACCEPT_LONG_STRINGS	0	If this option is set, strings that are too long are kept. Otherwise, they are truncated to 1024 characters. A warning is issued in both cases.
ASAP2_VERSION	171	Optionally, an output file of older ASAP2 format 1.70, 1.60, 1.50 or 1.40 can be created.
CREATE_INVALID_CALIBRATION_HANDLES	0	If this option is set, found identifier values for CALIBRATION_HANDLE (which are normally not allowed in ASAP2 syntax!) will be written to output file.

Parameter	Default Value	Description
DEFAULT_ENCODING_FOR_READ	""	Specifies the default encoding to be used when reading A2L files and C source code if they do not have a BOM ¹ . For files with BOM only the encoding according to the BOM is used. Valid values for the option are e.g.: SHIFT_JIS, UTF-16, UTF-32, US-ASCII, ISO-8859-1, UTF-8
HIDE_PROGRESS_VIEW	0	If this option is set, the progress display is deactivated.
MAX_INI_STRING_LENGTH	2048	Specifies the maximum length of strings that can be read from the INI file.
MINIMIZE_RESULT_FILE	0	If this option is activated, any superfluous information is omitted when creating the output file in order to keep the file as small as possible. Keywords that are omitted with this option (even if they were available in the input file): <ul style="list-style-type: none"> > ECU_ADDRESS_EXTENSION, if the value is 0 > Local indication of BYTE_ORDER, if the value does not differ from the global BYTE_ORDER > BIT_MASK, if the bit mask value corresponds with the default bit mask of the data type > Local indication of DEPOSIT, if the value corresponds to the global DEPOSIT value > FORMAT at measurement and calibration objects, if the format does not differ from the format of the referenced conversion rule > PHYS_UNIT at measurement and calibration objects, if the unit does not differ from the unit of the referenced conversion rule > DISPLAY_IDENTIFIER, if the name does not differ from the object name > ALIGNMENT at record layout, if the value does not differ from the global ALIGNMENT value > EXTENDED_LIMITS at calibration objects if they are consistent with the normal limits
PRESERVE_GLOBAL_COMMENTS	1	If this option is set, comments preceding the first keyword in the A2L input file are transferred to the result file.
REMOVE_DUPLICATE_IF_DATA_CANAPE_EXT	0	If this option is set, multiple occurrences of IF_DATA CANAPE_EXT information at measurement and calibration objects are removed. The first syntactically correct IF_DATA CANAPE_EXT (if any) is kept.
REMOVE_ROOT	0	If this option is set, no ROOT flag for groups is

¹ The BOM (**Byte Order Mark**) is the Unicode character U+FEFF referred at the start of a text stream, where it is used as an identifier for defining the byte order and encoding form in UCS/Unicode strings, especially text files.

Parameter	Default Value	Description
		written to the created A2L file.
REMOVE_TRAILING_DIMENSION_VALUES_1	0	If the A2L file is read using a version < 1.70, all the ones at the end of MATRIX_DIM are removed with the option set. MATRIX_DIM is never completely removed. A dimension value is kept in any case. Example: If you read a file with version 1.60 and save it with the current version 1.71, it will be converted as follows: > MATRIX_DIM 20 1 1 in MATRIX_DIM 20 > MATRIX_DIM 1 1 1 in MATRIX_DIM 1
SIGNIFICANT_DIGITS_FOR_DOUBLE	12	Specifies the number of significant digits used when writing Double values in the A2L file. > permissible values = 6 .. 17
SUPPRESS_VERSION_WARNINGS	0	Suppresses warnings about keywords that cannot be saved due to the set ASAP2 version. > 0 = All warnings are issued. > 1 = Warnings about individual object attributes that cannot be saved due to the ASAP2 version, are suppressed. Warnings about completely missing objects are however issued. > 2 = Also warnings about completely missing objects are suppressed.
WARNING_FOR_MISSING_ADDRESS_INFORMATION	0	If this option is set, messages reporting missing address information in measurement values are categorized as warnings instead of errors.
WRITE_UTF8	1	If this option is set, the created A2L file is saved in UTF8 format. Otherwise ASCII encoding is used.

2.3 Cross-Tool Tolerance Settings

Tolerance settings This section describes the tolerance settings that can be used for all command line tools in the same way. They are listed again under [SYNTAX_TOLERANCE] of those tools to which they apply but are not described in detail again.

Navigation help The different descriptions of the options are linked from the tools to this page. To navigate back again, press <Alt>+<Left Arrow>.

Parameter	Default Value	Description
DIVERSE_BLOCK_BRACKETS	0	When you activate this option, you will not get error messages about wrong block brackets in IF_DATA sections, if they do not match the AML definition.
IDENTS_START_WITH_DIGIT	0	Pursuant to ASAP2 standard, identifiers must start with a letter or with "_". Digits are not allowed for the first character of an identifier. If you activate this option, the tool will accept identifiers

Parameter	Default Value	Description
		starting with a digit as well. Attention: Activating this option can possibly lead to consecutive faults: Special words of the A2L file are no longer identifiable unambiguously as number or identifier – e.g. 1e7.
IGNORE_AML	0	By activating this option, you force the tool to completely ignore all AML and IF_DATA parts of the A2L file without any syntax check.
IGNORE_UNKNOWN_IF_DATA	0	If this option is set, the tool ignores all IF_DATA blocks where no AML description is available for.
KEYWORD_AS_SYMBOL	0	Reserved ASAP2 keywords must not be used for identifiers. If this option is set, the usage of keywords will be tolerated wherever it is possible without conflicts in ASAP2 grammar.
KEYWORDS_ALL_VERSIONS	0	Per default, all keywords are checked against the ASAP2 version. For keywords valid only in a higher ASAP2 version than set in current input file, a warning is created. This option deactivates this check: If set, all keywords are accepted which are valid with any ASAP2 version.
NESTED_COMMENTS	0	In A2L files, multi-line comments can be put into /* ... */. According to ASAP2 standard, these comments must not be nested. However, by setting this option, you enable nested comments for the tool .
PROJECT_NO_WITH_VALUE	0	In header of A2L file, after keyword PROJECT_NO, an identifier is expected. When activating this option, a number is accepted after PROJECT_NO as well.
PURE_OBJECT_LIST	0	If this option is set, the tool will also accept a file which contains only the content of a MODULE block – i.e. a pure list of measurement and calibration objects and conversion methods without the ASAP2 frame PROJECT, HEADER etc.
RESERVED_WITH_DATATYPE	0	By activating this option, you allow the usage of keyword RESERVED together with a data type (e.g. UBYTE, SBYTE or UWORD) in record layouts. Without this setting, according to ASAP2 standard only BYTE, WORD and LONG may be used with RESERVED.
SPECIAL_FORMAT_SYNTAX	0	According to ASAP2 standard, the string describing the significant digits resp. after dot digits must be specified in "". FORMAT "%x.y" When activating this option in INI file, a format setting without "" is accepted as well: FORMAT %x.y
SPECIAL_IDENT_INNER	""	Identifier in A2L files may contain only letters, digits and the special characters "_", ".", "[", and "]". With this option in INI file you can specify some more characters which shall be accepted by the tool inside an identifier (not as first character of an identifier). Sample: If you set SPECIAL_IDENT_INNER=-?+* then the characters "-", "?", "+", and "*" will be accepted as valid identifier characters.

Parameter	Default Value	Description
		<p>Attention: The usage of this option can possibly lead to consecutive faults! Especially space characters must not be marked for valid identifier characters here.</p>
SPECIAL_IDENT_START	""	<p>According to ASAP2 standard, identifiers may start only with a letter or with "_".</p> <p>With this option you can specify some more characters which shall be accepted by the tool as first character of an identifier.</p> <p>Sample: If you set SPECIAL_IDENT_START=#% then the characters "#" and "%" will be accepted as first character of an identifier.</p> <p>Attention: The usage of this option can possibly lead to consecutive faults!</p>
SYMBOL_AS_STRING	0	<p>If you activate this option, the tool will accept a string (any text inside "...") at each position of the A2L file where an identifier is expected.</p>

3 ASAP2 Creator

In this chapter, you will find the following information:

3.1	Functionality	page 14
3.2	Command Line Parameters	page 14
3.3	Exit Code	page 14
3.4	Initialization File	page 14
	[FILES]	
	[OPTIONS]	
	[SYNTAX_TOLERANCE]	
3.5	Syntax for Comments in C Code	page 18
	Object Name and Data Type	
	Conversion Rules	
	Comment	
	Display Name	
	Display Color	
	Linker Map Reference	
	Group Assignment	
	Dimensions	
	Record Layout	
	Curves and Maps	
	Hard-coded Addresses and Address Extensions	
	Default Events	
	Structures	
	Overwrite Element Properties	
	References to Structure Components	
	Byte Order	
3.6	Generating Prefixes for Object Names	page 57
3.7	Usage of Macros	page 59
3.8	Variant Coding	page 60
3.9	Format Description in Backus-Naur Form	page 61

3.1 Functionality

- Fragment** 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 A2L file, but only an "A2L fragment" file. It cannot be used independently, but must be added to the A2L master file via include statement inside the MODULE block.
- A2L master file** The A2L 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.
- Merge** If the A2L fragment to be created already exists and is already included in the Master A2L 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.

3.2 Command Line Parameters

- Program call** The following command line parameters can be entered when calling the program:

-L <name>	Name and path of the log file for warnings and error messages.
-I <name>	Name for the initializing file, if this file is not located in working directory or if its name is not <code>ASAP2Creator.INI</code> .

3.3 Exit Code

- Determining success** Via the exit code can be determined whether errors occurred during the **ASAP2 Creator** run:

0	No errors, no warnings
1	No errors, but warnings in log file
2	Error messages in log file
3	No license available

3.4 Initialization File

- Directory** The initialization file `ASAP2Creator.INI` normally is expected in current directory. Alternatively, a different name and directory can be set in the command line.

General For general information on the structure and usage of the initialization file, see section [Configuration of the Command Line Tools](#) on page 8.

3.4.1 [FILES]

Settings This section contains settings for the used files:

Parameter	Default Value	Description
ASAP2_MASTER	""	Name of the A2L master file.
MACRO_DEFINITION	""	Name of an initialization file containing macro definitions.
OBJECT_PREFIX_<n>	""	Prefix which is added to the object names from the <n> th source file. The numbering starts with 1 and corresponds to the numbering at SOURCE_FILE_<n>.
OUTPUT	""	Name of the A2L file to be created.
SEARCH_RECURSIVE	0	When searching for source files, sub directories are searched, too.
SOURCE_FILE_<n>	""	Name of C code file number <n>, the numbering starts with 1. The name may contain path information and wildcards. You can also use wildcards in the path specification itself. This allows you, for example, to include multiple subdirectories with the same naming pattern (e.g., "subDir_1", "subDir_2", "subDir_3", etc.) all from one line in the INI file. Notice: If wildcards are used for paths specifications, SOURCE_PATH and SOURCE_FILE_<n> must not contain relative directory specifications (like ".\Source" or "..\Source").
SOURCE_FILES	0	Number of C code files to be parsed.
SOURCE_PATH	""	Specifies a base directory for the source files to be read. All further path specifications under SOURCE_FILE_<n> are interpreted relatively to this base directory if they are not absolute path specifications. If no base directory is specified, the current working directory is used as base directory. An environment variable can be referenced. The usual Windows description is expected, e.g. \$(MyBasePathForASAP2CreatorSourceFiles)
USE_WILDCARDS	1	Set this parameter to 0 in order to directly use the specified source file names without interpreting them as wildcard expressions and searching for existing files matching. This option is particularly recommended if the ASAP2 Creator is to be used on a computer with ClearCase, since under these circumstances the API methods for finding files do not work reliably.

3.4.2 [OPTIONS]

Controlling behavior This section contains settings to control the behavior of the **ASAP2 Creator**.

Parameter	Default Value	Description
ACCEPT_LONG_STRINGS	0	See section Cross-Tool General Options.
ASAP2_VERSION	171	See section Cross-Tool General Options.
CREATE_FUNCTIONS	0	To be set to 1 to create FUNCTION objects instead of GROUP objects in A2L file.
CREATE_INVALID_CALIBRATION_HANDLES	0	See section Cross-Tool General Options.
CREATE_ROOT_GROUP	1	With setting 1 for each run of ASAP2 Creator one root group is generated in target file. All other user-defined groups are child groups of this root group. With setting 0 this implicit root group is not created, so that more than one user root groups are possible.
CREATE_STRUCTURES	0	If this option is set, the structure and instance definitions from the source files are not resolved into split elements. The corresponding type definitions are generated in the A2L file instead. This option can only be used if the target format specified for the A2L file is at least 170 (see <code>ASAP2_VERSION</code>). Otherwise the option will be ignored. SPLIT specification with set option: <ul style="list-style-type: none"> > is completely ignored on structure elements > is no longer mandatory with multidimensional instances, but can be used optionally to resolve instance arrays into split instances.
CREATE_WHOLE_FILE	0	If this option is set, a whole A2L file is created instead of a fragment. This whole A2L file contains both the newly created measurement and calibration objects and the objects and global settings of the master file.
DEFAULT_ENCODING_FOR_READ	""	See section Cross-Tool General Options.
DISABLE_REFERENCE_CHECK	0	If this option is set, the ASAP2 Creator does not check the referenced master objects for existence. Such referenced objects are conversion rules, record layouts, input quantities and common axes. Caution: The usage of this option may lead to an inconsistent A2L file.
FORCE_STANDARD_ARRAY_INDICES_FOR_MAP_SYMBOL	0	If this option is set, the index string for the MAP symbol name is always generated in the form [index] – no matter how the index string is defined for the ASAP2 object name. In this case, the index strings for object and symbol name may differ. Notice: This option is only effective if the option <code>USE_SPLIT_POSTFIX_FOR_MAP_SYMBOL</code> is also set.
HIDE_PROGRESS_VIEW	0	See section Cross-Tool General Options.

Parameter	Default Value	Description
MAP_FORMAT_SUPPORTS_STRUCTURES	1	To be set to 0, if the used MAP reader does not support structures. In that case, for creating Linker MAP references for structure components the base address of the structure must be used with an address offset.
MAP_SYMBOL_PREFIX	""	This prefix is added to the symbol names when creating the Linker MAP references for a measurement or calibration object.
MAX_INI_STRING_LENGTH	2048	See section Cross-Tool General Options.
MINIMIZE_RESULT_FILE	0	See section Cross-Tool General Options.
PRESERVE_GLOBAL_COMMENTS	1	See section Cross-Tool General Options.
REMOVE_DUPLICATE_IF_DATA_CANAPE_EXT	0	See section Cross-Tool General Options.
REMOVE_ROOT	0	See section Cross-Tool General Options.
REMOVE_TRAILING_DIMENSION_VALUES_1	0	See section Cross-Tool General Options.
SIGNIFICANT_DIGITS_FOR_DOUBLE	12	See section Cross-Tool General Options.
START_SYMBOL	"@@"	Start symbol which indicates the beginning of an ASAP2 Creator declaration inside a comment line in C code. This symbol must consist of 2 characters and must neither contain "/" nor "*".
SUPPRESS_VERSION_WARNINGS	0	See section Cross-Tool General Options.
USE_SPLIT_POSTFIX_FOR_MAP_SYMBOL	0	If the option is set to 1 and arrays are depacketized with the keyword SPLIT into individual objects, the MAP symbol name also is extended by the specified index string. For this, the appropriate offset calculation is not required. Notice: If the option FORCE_STANDARD_ARRAY_INDICES_FOR_MAP_SYMBOL is not set, with this option the same index string is generated for the object and symbol name.
WARNING_FOR_MISSING_ADDRESS_INFORMATION	0	See section Cross-Tool General Options.
WRITE_UTF8	1	See section Cross-Tool General Options.

3.4.3 [SYNTAX_TOLERANCE]

Tolerance settings

This section contains tolerance settings for the ASAP2 reader. Without additional configuration, the A2L file will be parsed strictly according to the ASAP2 standard. However, it is possible to tolerate special errors if configured so in this section.

Parameter	Default Value	Description
DIVERSE_BLOCK_BRACKETS	0	See section Cross-Tool Tolerance Settings.
IDENTS_START_WITH_DIGIT	0	See section Cross-Tool Tolerance Settings.
IGNORE_AML	0	See section Cross-Tool Tolerance Settings.
IGNORE_UNKNOWN_IF_DATA	0	See section Cross-Tool Tolerance Settings.
KEYWORD_AS_SYMBOL	0	See section Cross-Tool Tolerance Settings.

Parameter	Default Value	Description
KEYWORDS_ALL_VERSIONS	0	See section Cross-Tool Tolerance Settings .
NESTED_COMMENTS	0	See section Cross-Tool Tolerance Settings .
PROJECT_NO_WITH_VALUE	0	See section Cross-Tool Tolerance Settings .
PURE_OBJECT_LIST	0	See section Cross-Tool Tolerance Settings .
RESERVED_WITH_DATATYPE	0	See section Cross-Tool Tolerance Settings .
SPECIAL_FORMAT_SYNTAX	0	See section Cross-Tool Tolerance Settings .
SPECIAL_IDENT_INNER	""	See section Cross-Tool Tolerance Settings .
SPECIAL_IDENT_START	""	See section Cross-Tool Tolerance Settings .
SYMBOL_AS_STRING	0	See section Cross-Tool Tolerance Settings .

3.5 Syntax for Comments in C Code



Introduction: 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 `"@@"`.

3.5.1 Object Name and Data Type

- Object definition** 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.
- Object types** Valid object types are measurement, parameter, string, curve and map which can optionally be given write access attributes.
- Symbol name** 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.
- Data type** For the data type the ASAP2 terms 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** Format in Backus-Naur Form (see section [Format Description in Backus-Naur Form](#) on page 61).



Sample: Minimal configuration for creation of a measurement 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: Write-protected Parameter

```

/*
@@ 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
  /end IF_DATA
/end CHARACTERISTIC

```

```
LINK_MAP "sample4" 0 0 0 0 0 0 0
/end IF_DATA
/end CHARACTERISTIC
```



Sample: Parameter object with given range

For measurement values only one range can be specified, for parameters an additional extended range is optionally 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
```

3.5.2 Conversion Rules

Specifying conversion rule

To specify the conversion rule of a measurement or calibration object, it is either possible to refer to a global conversion rule or to define a "local" conversion rule together with the object.

For string objects, no conversion rule is allowed.

Global conversion rule

If a global 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.

Global conversion rules must either exist in the A2L master file or can be globally defined in the C code.

Locally defined conversion rule

If the conversion rule is defined locally together with the object or globally at a central position in the C code, a linear conversion with factor and offset, an algebraic conversion with formula text or a symbolic conversion table can be specified. Furthermore, the unit and optional the number of total digits and the precision can be specified.

For symbolic conversion tables, all value pairs must be listed. A value pair consists of either an input value and a string or an input range (min/max value) and a string.

UNIT

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

Format

Format in Backus-Naur Form (see section [Format Description in Backus-Naur Form](#) on page 61).



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

```

3.5.3 Comment

DESCRIPTION

To specify a comment for a measurement or calibration object the keyword **DESCRIPTION** can be used.

Format

Format in Backus-Naur Form (see section [Format Description in Backus-Naur Form on page 61](#)).



Sample:

```

/*
@@ SYMBOL = sample9
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = SLONG
@@ DESCRIPTION = "Das ist ein Kommentar"
@@ END
*/

→

/begin MEASUREMENT sample9 "Das ist ein Kommentar"
  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

```

3.5.4 Display Name

ALIAS

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

Format

Format in Backus-Naur Form (see section [Format Description in Backus-Naur Form on page 61](#)).

**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

```

3.5.5 Display Color

COLOR

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

Format

Format in Backus-Naur Form (see section **Format Description in Backus-Naur Form** on page 61).

**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

```

3.5.6 Linker Map Reference

BASE_OFFSET

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

Format in Backus-Naur Form (see section [Format Description in Backus-Naur Form](#) on page 61).



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

```



Sample: If the symbol names available in Linker MAP file are slightly different to the symbols names in C file declaration (e.g. leading underlines) 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.

```

[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

```

3.5.7 Group Assignment

Main group

For each set of parsed C code files one ASAP2 main group is created with the name of the file and stored into the A2L 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

Format in Backus-Naur Form (see section [Format Description in Backus-Naur Form](#) on page 61).



Sample:

```

/*
@@ MAIN_GROUP = SubSystemA
@@ DESCRIPTION = "Objekte des Subsystems A"
@@ END
*/

```

Changing group

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 A2L 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.

The group assignment can also be specified for a complete instance.

If a group is defined for the instance, all local group definitions (including the overwrites) for substructures and structure elements are ignored (see section [Structures](#) on page 45).

CREATE_ FUNCTIONS

With option `CREATE_FUNCTIONS` in initialization file, you can define whether ASAP2 objects of type `GROUP` or `FUNCTION` are to be created from the groups defined here.

For functions, there is a further possibility to assign measurement objects as input or output signals. In case of characteristic objects, it can be optionally specified if they are defined in the corresponding function.

Format

Format in Backus-Naur Form (see section [Format Description in Backus-Naur Form](#) on page 61).



Sample:

```

/*
@@ SYMBOL = channel
@@ A2L_TYPE = MEASURE

```

```

@@ DATA_TYPE = DOUBLE [ -100 ... 100 ]
@@ GROUP = SubGroup1 | Inputs
@@ END
*/

→

/begin GROUP SubSystemA "Objekte des Subsystems A"
  ROOT
  /begin SUB_GROUP
    SubGroup1
  /end SUB_GROUP
/end GROUP

/begin GROUP SubGroup1 ""
  /begin SUB_GROUP
    Inputs
  /end SUB_GROUP
/end GROUP

/begin GROUP Inputs ""
  /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 `GROUP` several times. The object is assigned to the last group in each group path.

Commenting a sub group

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 subgroup, the subgroup must be defined explicitly.

Format

Format in Backus-Naur Form (see section [Format Description in Backus-Naur Form](#) on page 61).



Sample:

```
/*
```



```

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

```



Sample: Creating functions

```

[OPTIONS]
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
*/

/*
@@ SYMBOL = signal4
@@ A2L_TYPE = PARAMETER
@@ DATA_TYPE = DOUBLE [ -100 ... 100 ]
@@ GROUP DEF = function2
@@ END
*/

/*
@@ SYMBOL = signal5
@@ A2L_TYPE = PARAMETER
@@ DATA_TYPE = DOUBLE [ -100 ... 100 ]
@@ GROUP = function2

```

```

@@ END
*/

→

/begin FUNCTION SubSystemA "All Objects of Subsystems A"
  /begin SUB_FUNCTION
    function1 function2
  /end SUB_FUNCTION
/end FUNCTION

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

/begin FUNCTION function2 ""
  /begin REF_CHARACTERISTIC
    signal5
  /end REF_CHARACTERISTIC
  /begin DEF_CHARACTERISTIC
    signal4
  /end DEF_CHARACTERISTIC
/end FUNCTION

```

3.5.8 Dimensions

Handling arrays

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 of the array (consisting of up to 5 individual dimensions) must be specified.

Several ASAP2 objects with SPLIT

To split an 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 for each dimension 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)
- > "%x" (numeric values are created from the index values in hexadecimal format using lower case letters 0, 1, 2,..., 9, a, b, c, d, e, f, 10,...)

- > "%X" (numeric values are created from the index values in hexadecimal format using upper case letters 0, 1, 2,..., 9, A, B, C, D, E, F, 10,...)

If neither an index template nor an index list is specified, the index suffixes are created as [index] by default, e.g., for a three-dimensional object as name[indexX][indexY][indexZ].

If the array is split to several ASAP2 objects, for all created ASAP2 objects the same basic map name is used by default with an individual offset calculated from data type and index. Alternatively, the index string by which the object is extended at each splitting, can also be used as an extension for the MAP symbol name (see option `USE_SPLIT_POSTFIX_FOR_MAP_SYMBOL`). In this case, no additional address offsets are generated.

When defining an array of strings, the keyword `SPLIT` must be used, i.e. the array must be split into single string objects, because the ASAP2 format does not support string arrays.

Format

Format in Backus-Naur Form (see section `Format Description in Backus-Naur Form` on page 61).



Sample: Create a two-dimensional 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 10 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

```

3.5.9 Record Layout

Optional keyword LAYOUT

To describe the record layout of multidimensional value blocks, the optional keyword **LAYOUT** can be used.

Record layout of parameters

According to the ASAP2 syntax the name of a record layout must be specified with parameters. This record layout must already exist in the A2L master file and must have a data type matching the parameter definition. If the keyword **LAYOUT** is not specified with a parameter, the **ASAP2 Creator** creates a new row-major ordered record layout corresponding to the data type of the parameter.

Record layout of measurement values

As measurement values in the A2L file are not assigned to a record layout, only the memory layout must be specified for them on the keyword **LAYOUT**:

- > COLUMN_DIR for column-major order
- > ROW_DIR for row-major order

If the optional keyword **LAYOUT** is missing for a measurement value, the row-major order applies by default.

Mandatory for maps/curves

For maps and curves specifying the keyword **LAYOUT** is mandatory (see section **Curves and Maps** on page 37).



Examples:

```

/*
@@ SYMBOL = ValueBlock1
@@ A2L_TYPE = PARAMETER
@@ DATA_TYPE = UBYTE
@@ DIMENSION = 2 3
@@ LAYOUT = RL_UBYTE_COLUMN_DIR
@@ END
*/

/*
@@ SYMBOL = ValueBlock2
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UBYTE
@@ DIMENSION = 2 3
@@ LAYOUT = COLUMN_DIR
@@ END
*/

```

```

→
/begin CHARACTERISTIC ValueBlock1 ""
  VAL_BLK 0x0 RL_UBYTE_COLUMN_DIR 0 NO_COMPU_METHOD 0 255
  MATRIX_DIM 2 3 1
  SYMBOL_LINK "ValueBlock1" 0
/end CHARACTERISTIC

/begin MEASUREMENT ValueBlock2 ""
  UBYTE NO_COMPU_METHOD 0 0 0 255
  ECU_ADDRESS 0x0
  LAYOUT COLUMN_DIR
  MATRIX_DIM 2 3 1
  SYMBOL_LINK "ValueBlock3" 0
/end MEASUREMENT

```

3.5.10 Curves and Maps

Record layout

To create curves and maps, you have to define record layouts in the A2L 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.

Supported axis types

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 A2L file. For the axis description, these values must be specified in raw format (ECU internal values). This can be done either in the form of a value list or for equidistant axis point values by specifying the range and the distance between the individual axis points. 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

Format in Backus-Naur Form (see section [Format Description in Backus-Naur Form](#) on page 61).


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

```

Record layout

The record layout `RL_Curve1` must exist in the A2L master file and must contain the description of the axis point values and curve values, 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 ... 14 ],2
@@ 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
    FIX_AXIS_PAR_DIST 2 2 7
  /end AXIS_DESCR
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "myMap1" 0 0 0 0 0 0 0
  /end IF_DATA
/end CHARACTERISTIC

```



Note: If axis points are equidistant, the specification of the distance is optional. If no distance is specified, a default value of 1 is assumed.



Note: Equidistant axis point values can only be displayed as such in ASAP2 format if all parameters are integer values. Otherwise, the list form will be chosen for the display in ASAP2 format.

Example: If `X_AXIS = FIX [0 ... 10], 2.5` is specified, the list of axis points `0, 2.5, 5, 7.5, 10` would be saved in ASAP2 format.

Record layout

The record layout `RL_UBYTE` must exist in A2L master file and must contain the description of the map values, 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

```

Record layout

The record layouts `RL_AXIS_FLOAT` and `RL_UWORD` must exist in A2L master file and must contain the description of the axis values and curve values, 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)
```

3.5.11 Hard-coded Addresses and Address Extensions

ADDRESS

If a variable is stored at a fixed address which is not available in Linker MAP file and which can/should not to be updated with the **ASAP2 Updater**, 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.

ADDRESS_EXTENSION

For variables, it is also possible to specify the address extension in the source code.

Format

Format in Backus-Naur Form (see section [Format Description in Backus-Naur Form](#) on page 61).



Sample 1: Hard-coded address

```
/*
@@ 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: Fixed address for split arrays

```
/*
@@ 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

```



Example 3: Address extension

```

/*
@@ SYMBOL = sample
@@ A2L_TYPE = MEASURE
@@ ADDRESS_EXTENSION = 3
@@ END
*/

```

3.5.12 Default Events

EVENT for CCP or XCP protocol

For measurement 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 A2L master file should contain the corresponding AML description.

EVENTS for the XCP protocol

For the XCP protocol, either a fixed event can be specified (event type `FIXED`) or a list of all events supported for this measurement values (event type `VARIABLE`). When using the event type `VARIABLE` the first event of the list is set as a default event in the A2L file.

Alternatively, only the default event for XCP with the event type `DEFAULT` can be specified.

EVENT for the CCP protocol

For CCP events there is no distinction of the event type.

Format

Format in Backus-Naur Form (see section [Format Description in Backus-Naur Form](#) on page 61).

**Sample: Event 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: Fixed event 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

```

```

/end DAQ_EVENT
/end IF_DATA
/end MEASUREMENT

```



Sample: Event list for XCP

```

/*
@@ SYMBOL = EventTestXCP2
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UBYTE
@@ EVENT XCP = VARIABLE 2 3 4
@@ END
*/

→

/begin MEASUREMENT EventTestXCP2 ""
  UBYTE NO_COMPU_METHOD 0 0 0 255
  ECU_ADDRESS 0x0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "EventTestXCP2" 0 0 0 0 0 0 0
  /end IF_DATA
  SYMBOL_LINK "variable" 0
  /begin IF_DATA XCP
    /begin DAQ_EVENT VARIABLE
      /begin AVAILABLE_EVENT_LIST
        EVENT 2
        EVENT 3
        EVENT 4
      /end AVAILABLE_EVENT_LIST
      /begin DEFAULT_EVENT_LIST
        EVENT 2
      /end DEFAULT_EVENT_LIST
    /end DAQ_EVENT
  /end IF_DATA
/end MEASUREMENT

```



Sample: Default event for XCP

```

/*
@@ SYMBOL = EventTestXCP3
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UBYTE
@@ EVENT XCP = DEFAULT 3
@@ END

```

```

*/
→

/begin MEASUREMENT EventTestXCP3 ""
  UBYTE NO_COMPU_METHOD 0 0 0 255
  ECU_ADDRESS 0x0
  /begin IF_DATA CANAPE_EXT
    100
    LINK_MAP "EventTestXCP3" 0 0 0 0 0 0
  /end IF_DATA
  SYMBOL_LINK "variable" 0
  /begin IF_DATA XCP
    /begin DAQ_EVENT VARIABLE
      /begin DEFAULT_EVENT_LIST
        EVENT 3
      /end DEFAULT_EVENT_LIST
    /end DAQ_EVENT
  /end IF_DATA
/end MEASUREMENT

```

3.5.13 Structures

- Structure definition** 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 this structure definition and then to 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.
- Instance definition** For each instance, the structure used and previously defined must be specified. The object names for the instance are then created by adding the leaf name to the given variable name – separated by a dot.
- Structure before instance** The structure must be defined before it is specified in the instance definition. In particular, if the structure is defined in a different source file than the instance, it must be ensured through the sequence of the source files that the **ASAP2 Creator** reads in the structure definition before the instance definition.
- Substructures** Substructures which are arrays are always split, because the ASAP2 language does not support arrays of structured objects, i.e. the `SPLIT` keyword must always be used here. The dimension and syntax for split array indices can be specified like described above.
- For the description of substructures which are already defined outside the current structure a simple reference can be used to minimize the description code.
- For substructures which are no arrays, no description is necessary.
- Format** Format in Backus-Naur Form (see section [Format Description in Backus-Naur Form](#) on page 61).

**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 substructure 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 reader does not supply detailed structure information and contains only the base addresses of structures, for each element the offset to the base address must 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 substructures the offsets are specified relative to the current substructure. Therefore, it is necessary to specify additionally the offset of substructures to the parent structure – also considering the alignment settings.

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

Format

Format in Backus-Naur Form (see section [Format Description in Backus-Naur Form](#) on page 61).



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 innerhalb der Struktur xyz

  @@ SIZE = 6
  Total size of the substructure abc including the dummy
  component no ASAP2 object is created for.

  @@ DIMENSION = 2

```

```

@@ END
*/
struct {

    /* create no ASAP2 object for this component */
    short Dummy;

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

```

3.5.14 Overwrite Element Properties

Properties of a structure instance

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

Format

Format in Backus-Naur Form (see section [Format Description in Backus-Naur Form](#) on page 61).



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
    /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
```

3.5.15 References to Structure Components

INSTANCE_NAME Inside a structure definition, to describe the reference to a component of same structure (e.g. for input quantity or common axis), the keyword `INSTANCE_NAME` can be used. When creating the instance variables, the creator then prefixes the instance name to the name of the referenced object automatically – separated by a dot.

Format Format in Backus-Naur Form (see section [Format Description in Backus-Naur Form](#) on page 61).



Sample:

```
struct ReferenceTest {

    /*
    @@ ELEMENT = Source
    @@ STRUCTURE = ReferenceTest
    @@ A2L_TYPE = MEASURE
    @@ DATA_TYPE = UWORD
    @@ END
    */
    unsigned short Source;

    /*
    @@ ELEMENT = Axis
    @@ STRUCTURE = ReferenceTest
    @@ A2L_TYPE = AXIS
    @@ DATA_TYPE = UWORD
    @@ LAYOUT = RL_AXIS_UWORD
    @@ DIMENSION = 3
    @@ INPUT = INSTANCE_NAME Source
    @@ END
    */
    unsigned short Axis[3];

    /*
    @@ ELEMENT = Curve
    @@ STRUCTURE = ReferenceTest
    @@ A2L_TYPE = CURVE
    @@ DATA_TYPE = UWORD
    @@ LAYOUT = RL_CURVE_UWORD
    @@ X_AXIS = COMMON INSTANCE_NAME Axis
    @@ END
    */
}
```

```

    unsigned short Curve[3];

};

/*
@@ INSTANCE = x
@@ STRUCTURE = ReferenceTest
@@ DIMENSION = 1 SPLIT
@@ END
*/
struct ReferenceTest x;

→

/begin MEASUREMENT x[0].Source ""
    UWORD NO_COMPU_METHOD 0 0 0 65535
    ECU_ADDRESS 0x0
    SYMBOL_LINK "x[0].Source" 0
/end MEASUREMENT

/begin AXIS_PTS x[0].Axis ""
    0x0 x[0].Source RL_AXIS_UWORD 0 NO_COMPU_METHOD 3 0 65535
    SYMBOL_LINK "x[0].Axis" 0
/end AXIS_PTS

/begin CHARACTERISTIC x[0].Curve ""
    CURVE 0x0 RL_CURVE_UWORD 0 NO_COMPU_METHOD 0 65535
    /begin AXIS_DESCR
        COM_AXIS x[0].Source NO_COMPU_METHOD 3 0 65535
        AXIS_PTS_REF x[0].Axis
    /end AXIS_DESCR
    SYMBOL_LINK "x[0].Curve" 0
/end CHARACTERISTIC

/begin MEASUREMENT x[1].Source ""
    UWORD NO_COMPU_METHOD 0 0 0 65535
    ECU_ADDRESS 0x0
    SYMBOL_LINK "x[1].Source" 0
/end MEASUREMENT

/begin AXIS_PTS x[1].Axis ""
    0x0 x[1].Source RL_AXIS_UWORD 0 NO_COMPU_METHOD 3 0 65535
    SYMBOL_LINK "x[1].Axis" 0
/end AXIS_PTS

/begin CHARACTERISTIC x[1].Curve ""
    CURVE 0x0 RL_CURVE_UWORD 0 NO_COMPU_METHOD 0 65535

```

```

/begin AXIS_DESCR
  COM_AXIS x[1].Source NO_COMPU_METHOD 3 0 65535
  AXIS_PTS_REF x[1].Axis
/end AXIS_DESCR
SYMBOL_LINK "x[1].Curve" 0
/end CHARACTERISTIC

```

3.5.16 Byte Order

- BYTE_ORDER** If the byte order of individual objects differs from the default byte order specified in the A2L master file, this can be locally overwritten using the keyword `BYTE_ORDER`.
- Local specification** The specification of a local byte order is possible for individual objects as well as for structure components.
- Format** Format in Backus-Naur Form (see section [Format Description in Backus-Naur Form](#) on page 61).



Example:

```

/*
@@ SYMBOL = MeasureIntel
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UBYTE
@@ BYTE_ORDER = INTEL
@@ END
*/

→

/begin MEASUREMENT MeasureIntel ""
  UBYTE NO_COMPU_METHOD 0 0 0 255
  BYTE_ORDER MSB_LAST
  ECU_ADDRESS 0x0
  SYMBOL_LINK "MeasureIntel" 0
/end MEASUREMENT

```

3.6 Generating Prefixes for Object Names

- Uniqueness of object names** When the object name in the created A2L file is not unique (e.g. if a parameter with the same name is defined in multiple source files), it is possible to specify a prefix per source file. This prefix is prepended to all created object names from this file.
- Validation** This applies only to the ASAP2 names, not to the symbol names of the MAP file.
- Location** The prefixes are specified in the INI file.



Sample 1:

If the prefix `ABC.` is specified for the file, a measurement value `ABC.Sample1` is

created from the following code. The measurement value references a MAP symbol with the name `Sample1`.

```
/*
@@ SYMBOL = Sample1
@@ A2L_TYPE = MEASURE
@@ DATA_TYPE = UBYTE
@@ END
*/
```



Sample 2:

From the following code, a measurement value `ABC.NewName` is created that references a MAP symbol named `Sample2`.

```
/*
@@ SYMBOL = Sample2
@@ A2L_TYPE = MEASURE NewName
@@ DATA_TYPE = UBYTE
@@ END
*/
```



Caution: Using the ASAP2 Creator, it is possible to specify references to other ASAP2 objects.

This comprises on one hand references to shared axes and on the other hand references to input values of axes (see section *Curves and Maps* on page 37).

The referenced objects can be

- > available in the master file and therefore receive no prefix
- > created from the same C file with the **ASAP2 Creator**, and thus receive the same prefix as the referencing object
- > created from a different C file with the **ASAP2 Creator** (in the same Creator-run) and may thus receive a different prefix.

Hence, the prefix cannot be added automatically. The name specified in the reference must therefore already be unique. If necessary, the prefix must be specified here directly.



Sample:

```
/*
@@ SYMBOL = Curve1
@@ A2L_TYPE = CURVE
@@ DATA_TYPE = SWORD
@@ LAYOUT = RL_SWORD
@@ X_AXIS = FIX 10 20 30 40 50
@@ INPUT = ABC.Sample1
@@ END
*/
```

3.7 Usage of Macros

Macro definitions in additional initialization file

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: Initialization file

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



Sample: 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
  /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".

3.8 Variant Coding

Generate information The **ASAP2 Creator** can also be used to generate information on variant coding.

Limitation Because the variant coding must be written as a whole in an A2L file, the following limitations apply:

- > Either the **ASAP2 Creator** must be configured in such a way that it generates a complete file (`CREATE_WHOLE_FILE` option). In this case, the variant coding already available in the master file is merged with the variant information newly created in the Creator comments.
- > Or, the master file must not contain any variant information at all.

Conditions for master file with variant coding If the master file already contains variant coding, this can only be supplemented by the **ASAP2 Creator** if it is consistent and the following minimum conditions are met:

- > Each variant criterion must have a selector. The associated selector variable must exist in the master file and have a verbal conversion rule that writes the assignment of the variants to the selector values.
- > Each variant-coded calibration value references exactly one variant criterion.
- > The variant offsets can be calculated explicitly for each variant criterion. In particular, this requires that each variant criterion be referenced by at least one variant-coded calibration value.
- > Variant criteria defined in the Creator comments must not yet exist in the master.

Definition The new variant criteria are defined in the **ASAP2 Creator** by specifying the name, an optional comment, and a selector variable (this can be a measurement or calibration value) and listing the variants - each with associated selector value and address offset. The selector variable must either already be available in the master file or be defined in the Creator comments.



Attention: The address offset of the variants can only be saved in the A2L file if the variant criterion is referenced by at least one variant-coded calibration value.



Example: Creation of a variant criterion and a conversion rule

```

/*
@@ VAR_CRITERION = Color
@@ DESCRIPTION = "... "
@@ SELECTOR = MEASURE ColorSelector
@@ VARIANT = Blue 1 0
@@ VARIANT = Red 2 0x1000
@@ VARIANT = Yellow 3 0x2000
@@ END
*/

```

Result	From this definition, the variant criterion <code>Color</code> is created with the variants <code>Blue</code> , <code>Red</code> , and <code>Yellow</code> . In addition, a conversion rule <code>Color.Selector.Conversion</code> is created, which must be assigned to the <code>ColorSelector</code> selector variable.
Automatic assignment of the conversion rule	This assignment can only be made automatically by the ASAP2 Creator if the selector variable is contained in the result file. This is the case when <ul style="list-style-type: none"> > the <code>CREATE_WHOLE_FILE</code> option for creation of a complete A2L file is active or > the selector variable was generated by the ASAP2 Creator and is contained in the generated fragment.
Name of the variant criterion	For variant-coded calibration values, the name of the variant criterion on which the calibration value depends can be specified. The corresponding variant criterion must either already be available in the master file or be created through corresponding definitions in the ASAP2 Creator (see above).
Updating of variant addresses	According to the number of variants of the assigned variant criterion and the variant offsets, variant addresses are generated for the variant-coded calibration value. Of these, one address is typically 0 in the case of newly generated objects. The addresses can be updated with the ASAP2 Updater . The <code>UPDATE_VARIANT_ADDRESSES</code> option must be activated for updating the variant addresses.
Interpretation by CANape	So that CANape can interpret the variant coding, it must also be ensured that all variant-coded calibration values that are dependent on the same variant criterion are assigned to a common subgroup.

**Example:**

```

/*
@@ SYMBOL = VariantCodedParameter
@@ A2L_TYPE = PARAMETER
@@ DATA_TYPE = UBYTE
@@ VAR_CRITERION = Color
@@ GROUP = ColorGroup
@@ END
*/

```

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

<write_access> ::= WRITEABLE

```



```

| READ_ONLY

<length> ::= <value>

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

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

<bitmask> ::= <value>

<attribute> ::= <conversion >
| <description>
| <alias>
| <base_offset>
| <group_assignment>
| <dimension> [ <split> ]
| <address>
| <address_extension>
| <event>
| <color>
| VAR_CRITERION = <name>
| LAYOUT = <name>
| <byte_order>

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

<conversion> ::= CONVERSION = <conversion_name>
| [ [ <length> ] <digits> ]
| CONVERSION = LINEAR <factor> <offset> <unit>
| [ [ <length> ] <digits> ]

```

```

| CONVERSION = FORMULA <string> [ INVERSE <string> ]
| <unit> [[ <length> ] <digits> ]
| CONVERSION = TABLE ( <value> [ <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 | DEF ] =
<group_name> ( | <group_name> ) *

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

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

<address> ::= ADDRESS = <value>

<address_extension> ::= ADDRESS_EXTENSION = <value>

<color> ::= COLOR = <value>

<event> ::= EVENT CCP = <value>
| EVENT XCP = FIXED <value>
| EVENT XCP = VARIABLE ( <value> )+
| EVENT XCP = DEFAULT <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>
    [ DATA_TYPE = STRUCTURE <structure_name> ]
    ( <structure_attribute> )*
    END

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

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

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

<instance_definition> ::= INSTANCE = <symbol_name> [ <a2l_name> ]
    STRUCTURE = <structure_name>
    [ <address> ]
    [ <dimension> [ <split> ] ]
    [ SIZE = <value> ]
    [ <group_assignment> ]
    ( <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 [ <write_access> ]
[ <a2l_name> ]
<datatype>
LAYOUT = <layout_name>
DIMENSION = <dimension>
[ <input_signal> ]
( <map_attribute> )*
END

<map_attribute> ::= <conversion>
| <description>
| <alias>
| <base_offset>
| <group_assignment>
| <address>
| <address_extension>
| VAR_CRITERION = <name>
| <byte_order>

<byte_order> ::= BYTE_ORDER = INTEL
| BYTE_ORDER = MOTOROLA

<axis> ::= STANDARD <datatype>
DIMENSION = <dimension>
[ <input_signal> ]
[ <conversion> ]
| FIX <list_of_axis_points>
[ <input_signal> ]
[ <conversion> ]
| FIX [ <min_value> ... <max_value> ] [, <distance> ]
[ <input_signal> ]
[ <conversion> ]
| COMMON [ INSTANCE_NAME ] <common_axis_name>

<input_signal> ::= INPUT = [ INSTANCE_NAME ] <input_signal_name>

```

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

<variant_criterion> ::= VAR_CRITERION = <name>
                       [ <description> ]
                       SELECTOR = ( PARAMETER | MEASURE ) <name>
                       ( <variant> )*
                       END

<variant > ::= VARIANT = <name> <selector_value> <offset>
```

4 ASAP2 Updater

In this chapter, you will find the following information:

4.1	Functionality	page 69
4.2	Command Line Parameters	page 70
4.3	Exit Code	page 70
4.4	Initialization File	page 71
	[OPTIONS]	
	[ADDRESS_RANGE_x]	
	[CREATION_RANGE_x]	
	[EPK]	
	[COFF]	
	[ELF]	
	[IEEE]	
	[GREENHILL]	
	[PDB]	
	[DATATYPES]	
	[CAL_PARAM_GROUPS]	
	[SYNTAX_TOLERANCE]	
4.5	Warning Levels	page 91
4.6	Interface to MAP Readers	page 91
4.7	Examples	page 92

4.1 Functionality

Updating ASAP2 source files

The **ASAP2 Updater** reads an A2L source file and updates the address and structure information of measurement and calibration objects based on the content in the Linker MAP file. The most prevalent MAP and debug formats are supported, such as ELF, PDB and COFF. The desired MAP reader is configured in an initialization file UPDATER.INI.

The addresses of memory segments, EPK and calibration methods can also be updated if the IF_DATA CANAPE_ADDRESS_UPDATE exists in ASAP source file.

Updating data types

The update of data type information and dimensions as well as the update of structure components 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 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.

Filter function


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 MAP 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.

4.2 Command Line Parameters

Program call

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

-I <name>	Name of the A2L input file.
-O <name>	Name of the A2L output file to be generated.
-A <name>	Name of the MAP file/address file.
-L <name>	Name and path of the log file for warnings and error messages.
-G <name>	<p>Group filter mode: Name of a group file with the names of groups and functions, which should enter the result file. The file must contain the name of a group/function for each line. Neither wildcards nor regular expressions are allowed.</p> <div style="border: 1px solid #ccc; padding: 5px; background-color: #f0f0f0;"> <p> Note: When using additional filter options, we recommend 2 separate Updater runs to avoid mutual influencing of the filters. For example, modes 3, 4 and 5 of the <code>FILTER_MODE</code> option cannot be combined with the group filter.</p> </div> <p>The group filter can be set recursively. This is done using the setting <code>GROUP_FILTER_RECURSIVE</code> the INI file.</p>
-T <name>	Name for the initializing file, if this file is not located in working directory or if its name is not <code>UPDATER.INI</code> .
-H <name>	<p>Name of the HEX file from which the EPK identifier shall be read out. This option is only relevant if the switch <code>UPDATE_EPK</code> is set in the INI file. This parameter cannot be used together with the option <code>-E</code>.</p>
-E <name>	<p>EPK identifier, which shall be used for updating. This option is only relevant if the switch <code>UPDATE_EPK</code> is set in the INI file. This parameter cannot be used together with the option <code>-H</code>.</p>
-@<name>	<p>Name of a text file with all command line parameters. The preceding hyphen is optional.</p>



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



Sample: Call

```
ASAP2Updater @CMDLINE.TXT
```

4.3 Exit Code

Determining success The success of the update process 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
3	No license available

4.4 Initialization File

Directory The initialization file `UPDATER.INI` normally is expected in current directory. Alternatively, a different directory can be set in the command line.

General For general information on the structure and usage of the initialization file, see section [Configuration of the Command Line Tools](#) on page 8.

4.4.1 [OPTIONS]

Common settings The following section contains common settings.



Note:

If you are using a MAP format that is not supported and cannot switch to one of the more commonly supported MAP formats, please contact Vector (see section [Support](#) on page 7).

Parameter	Default Value	Description
<code>ACCEPT_LONG_STRINGS</code>	0	See section Cross-Tool General Options .
<code>ADDRESS_RANGES</code>	0	Number of address ranges, for which an offset is defined (see section [ADDRESS_RANGE_x]).
<code>ASAP2_VERSION</code>	171	See section Cross-Tool General Options .
<code>CREATE_ARRAYS</code>	1	Instead of individual ASAP2 objects per array element, only one single object is generated for the whole array in the A2L file (using <code>MATRIX_DIM</code>). Arrays can be generated for the following MAP file formats: <code>COFF</code> , <code>ELF</code> and <code>PDB</code> . Notice: This option is automatically set when <code>CREATE_STRUCTURES</code> is activated. Caution: Starting with version 14.0 the default value is 1!
<code>CREATE_INVALID_CALIBRATION_HANDLES</code>	0	See section Cross-Tool General Options .
<code>CREATE_LINKER_MAP_REFERENCES</code>	0	Creates the missing CANape Linker MAP references (<code>IF_DATA CANAPE_EXT</code>) for updated objects. If the result file has the ASAP2 format ≥ 1.60 , also <code>SYMBOL_LINK</code> keywords are created if necessary.
<code>CREATE_MAPCONV_TMP_FILE</code>	0	Forces the creation of the temporary file

Parameter	Default Value	Description
		MAPCONV.TMP which is the ASCII representation of the Map file.
CREATE_MISSING_STRUCTURE_COMPONENTS	0	Creates missing components and the related typedefs according to the assigned MAP file structure. The type of the newly created components (Measurement/Calibration) follows the previous type of structure in the A2L file. Notice: This option is relevant only if the option UPDATE_STRUCTURES is activated.
CREATE_STRUCTURES	0	If the option CREATION_RANGES is activated, this option generates typedefs and structures instead of flattened individual elements. Notice: This option includes the CREATE_ARRAYS option, which is activated automatically.
CREATION_RANGES	0	Number of address ranges, for which objects shall be created.
DEFAULT_ADDRESS_EXTENSION	0	Specifies the default value for the address extension used for updating the variables when the current MAP reader does not support any address extensions. Notice: This option is relevant only if the option UPDATE_ADDRESS_EXTENSIONS is activated.
DEFAULT_ENCODING_FOR_READ	""	See section Cross-Tool General Options.
DELETE_EMPTY_GROUPS	0	Empty groups/functions are removed in result file. Notice: This option does not work together with FILTER_IGNORE_REFERENCES.
DELETE_MISSING_STRUCTURE_COMPONENTS	0	If a structure component with the given symbol name is not found for a structure in the MAP file, this structure component is removed from the structure in the A2L file with this option activated. Notice: This option is relevant only if the option UPDATE_STRUCTURES is activated.
DELETE_WRONG_REFERENCES	0	If this option is activated, references that are no longer valid are ignored when writing groups and functions. Empty groups are removed if the option DELETE_EMPTY_GROUPS is activated additionally.
EXPORT_ONLY_REFERENCED	0	If this option is activated, only the storage schemes and conversion rules and tables actually referenced within the A2L file are transferred to the result file. > 0 = Not active. All storage schemes and conversion rules and tables are transferred to the result file. > 1 = Option active

Parameter	Default Value	Description
<code>FILTER_IGNORE_REFERENCES</code>	0	<p>If this option is set to 1, objects referencing removed variables are not checked and adjusted, while filtering.</p> <p>This results in a significant performance improvement when having many removed objects.</p> <p>Notice: This option does not work together with <code>DELETE_EMPTY_GROUPS</code>.</p>
<code>FILTER_MODE</code>	0	<p>This flag is used to specify which measurement and calibration objects of source file are saved in result file:</p> <ul style="list-style-type: none"> > 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 measurement objects enter the result file. > 4 = Only calibration objects enter the result file. > 5 = Only virtual measurement and calibration objects enter the result file. > 6 = Only the variables found in MAP file as well as virtual measurement and calibration objects enter the result file (equals a combination of filter mode 1 and 5). <p>Attention: Modes 3, 4 and 5 cannot be used together with group filters.</p>
<code>GROUP_FILTER_RECURSIVE</code>	0	<p>This option is only relevant if a group filter is configured with the command line option <code>-G</code>. So when this option is activated, the group filter acts recursively, also for all subgroups/subfunctions.</p>
<code>HIDE_PROGRESS_VIEW</code>	0	<p>See section Cross-Tool General Options.</p>
<code>IGNORE_MAP_REFERENCES</code>	0	<p>With this option activated, the names of the Linker MAP references in <code>IF_DATA</code>, <code>CANAPE_EXT</code> resp. <code>SYMBOL_LINK</code> or <code>SYMBOL_TYPE_LINK</code> for structures are ignored. Instead, the object names are used to search for the corresponding symbol in the MAP file.</p>
<code>IGNORE_OBJECTS_WITHOUT_SYMBOL_LINK</code>	0	<p>With this option activated, all objects that have no explicit reference to a symbol in the MAP file are completely ignored when updating. I.e. they are not searched alternatively by their object names in the MAP file.</p> <p>This applies both to variables and to structures. Ignored variables are not removed on <code>FILTER_MODE 1</code>.</p>
<code>INCLUDE_SAVE_MODE</code>	0	<p>If this option is activated, the assignment of objects to included files is retained. After updating, the objects are saved to their original</p>

Parameter	Default Value	Description
		<p>file again, and corresponding include instructions are generated in the result file of the ASAP2 Updater.</p> <p>Attention: All original include files are overwritten with this option!</p> <p>The assignment to an include file is retained for the following objects: CHARACTERISTIC, MEASUREMENT, AXIS_PTS, FUNCTION, GROUP, COMPU_METHOD, COMPU_TAB, COMPU_VTAB, COMPU_VTAB_RANGE, RECORD_LAYOUT, UNIT, FRAME, USER_RIGHTS, TYPEDEF_xx, TRANSFORMER, BLOB</p> <p>AML definitions and IF_DATA blocks are always transferred to the main file.</p> <ul style="list-style-type: none"> > 0 = Assignment to include files is NOT retained in the result file. > 1 = Assignment to include files is retained in the result file.
MAP_FILE_NAMES_SUFFIX	""	The given suffix is added to the names created from Linker MAP file.
MAP_FORMAT	0	<p>Specifies the format of the MAP file:</p> <ul style="list-style-type: none"> > 0 = Standard > 6 = IEEE (Cosmic, Tasking, etc.) > 17 = Texas Instruments TMS470 > 19 = COFF > 25 = DiabData > 30 = ELF/DWARF 16 bit > 31 = ELF/DWARF 32/64 bit > 37 = Greenhill Multi 2000 > 39 = COFF settings auto detected > 40 = NEC CC78K/0 v35 > 41 = Microsoft extended > 44 = COFF/DWARF > 51 = Microsoft standard VC8 > 52 = Microsoft VC8 (MATLAB DLL) > 53 = Microsoft VC8 C++ (MATLAB DLL) > 54 = Microsoft VC8 Debug File (pdb) <p>Notice: We recommend using 154, as 54 will not be further developed.</p> <ul style="list-style-type: none"> > 55 = Microsoft VC++ (MATLAB DLL) > 56 = EXE/DWARF > 131 = ELF 32/64 Bit extended > 144 = COFF/DWARF extended > 154 = Microsoft VC8 Debug File PDB extended

Parameter	Default Value	Description
		<p>> 156 = EXE/DWARF extended</p> <p>Notice: The new MAP readers (>100) each read the same formats as their predecessors (corresponding number minus 100), but provide extended information about classes and typedefs.</p> <p>The MAP readers 131, 144 and 156 are currently only supported starting with DWARF2.</p>
MAX_INI_STRING_LENGTH	2048	See section Cross-Tool General Options.
MINIMIZE_RESULT_FILE	0	See section Cross-Tool General Options.
PRESERVE_GLOBAL_COMMENTS	1	See section Cross-Tool General Options.
REMOVE_DUPLICATE_IF_DATA_CANAPE_EXT	0	See section Cross-Tool General Options.
REMOVE_ROOT	0	See section Cross-Tool General Options.
REMOVE_TRAILING_DIMENSION_VALUES_1	0	See section Cross-Tool General Options.
SIGNIFICANT_DIGITS_FOR_DOUBLE	12	See section Cross-Tool General Options.
SUPPRESS_UPDATE_OF_VIRTUAL	0	Suppress the address and data type update of virtual objects.
SUPPRESS_VERSION_WARNINGS	0	See section Cross-Tool General Options.
UPDATE_ADDRESS_EXTENSIONS	0	<p>If this option is activated, address extensions of the variables are updated.</p> <p>The address extensions are determined from the MAP file by the Linker MAP reader and have priority.</p> <p>If the current MAP reader does not support address extensions, you can specify a default address extension using the option <code>DEFAULT_ADDRESS_EXTENSION</code>. This is then set for all variables.</p>
UPDATE_ADDRESSES_TO_ZERO	0	The addresses of objects, which could not be updated, are set to zero.
UPDATE_CALIBRATION_METHODS	0	<p>If this option is activated, address and size of pointer tables for pointer-based calibration methods are updated.</p> <p>The updating is based on the <code>IF_DATA CANAPE_ADDRESS_UPDATE</code>.</p> <p>In addition, if the option is activated, the addresses in <code>IF_DATA CANAPE_CAL_METHOD</code> are updated.</p>
UPDATE_CALIBRATION_OFFSET	0	If this option is activated, the value for <code>ECU_CALIBRATION_OFFSET</code> is updated based on the <code>IF_DATA CANAPE_ADDRESS_UPDATE</code> .
UPDATE_DIMENSIONS	0	If this option is activated, the dimension of value blocks, measurement arrays and instance arrays is updated with the data from the MAP file.
UPDATE_ENUM_TABLES	0	If this option is activated, for objects whose MAP symbol represents an ENUM variable, verbal conversion tables are created.

Parameter	Default Value	Description
		This feature is not supported by all MAP readers. The following formats support this option: ELF (supported starting with DWARF2), PDB
UPDATE_EPK	0	<ul style="list-style-type: none"> > 0 = The EPK addresses and the EPK identifier are not updated. > 1 = The addresses for the EPROM identifier are updated, based on the IF_DATA CANAPE_ADDRESS_UPDATE. > 2 = For the EPK addresses MAP symbols can be specified in the INI file which shall be used for updating. <p>For updating the EPK identifier either an Intel HEX file can be specified from which the identifier is read out of the (if necessary updated) EPK addresses. Alternatively, the EPK identifier can be specified directly on the command line.</p> <p>When the EPK identifier is read out from an Intel HEX file, 256 characters are read starting from the (updated, if necessary) address (maximum length of a string in ASAP2 format). If the Intel HEX file does not contain that much data, a message is output, and the available characters are used anyway. The length of the EPK is detected based on the printable characters starting from the defined address.</p>
UPDATE_MEMORY_SEGMENTS	1	<p>If this option is activated, the addresses for the memory segments are updated. The updating is based on the IF_DATA CANAPE_ADDRESS_UPDATE.</p> <p>Simultaneously, the address mapping for XCP is updated when the corresponding symbol information exist.</p>
UPDATE_STRUCTURES	0	<p>If this option is activated, the following is updated with the data from the MAP file:</p> <ul style="list-style-type: none"> > the size for structures > the address offset and the dimension for all structure components <p>Notice: This option only works for those MAP readers that provide extended typedef information (MAP reader >100 and ELF reader with activated option ELF_USE_CPP_EXTENSION_DWARF2).</p>
UPDATE_VARIANT_ADDRESSES	0	<p>For variant-coded parameters, the variant addresses are also updated. In so doing, an offset is added to all variant addresses of a parameter. This offset results from the difference between the new object address (after updating with the MAP file) and the old object address (before updating with the MAP file).</p>

Parameter	Default Value	Description
		<ul style="list-style-type: none"> > 0 = Not active > 1 = Active
WARNING_FOR_MISSING_ADDRESS_INFORMATION	0	See section Cross-Tool General Options .
WARNING_FOR_MISSING_REFERENCES	0	Categorizes the messages about missing references in the input files as follows: <ul style="list-style-type: none"> > 0 = Error messages > 1 = Warnings > 2 = Information
WARNING_LEVEL	2	Warning level (see section Warning Levels on page 91). To the log file only warnings are issued, whose level is less or equal to the value set here.
WARNING_UNKNOWN_OBJECT	0	In filter mode: display warnings for all names of filter file which are not found in A2L file.
WRITE_UTF8	1	See section Cross-Tool General Options .

4.4.2 [ADDRESS_RANGE_x]

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

For all measurement and calibration objects inside one address range the configured offset is subtracted from the address available in Linker MAP file.

Parameter	Default Value	Description
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.
START	0	Start address of range in Linker MAP file.

4.4.3 [CREATION_RANGE_x]

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

The **ASAP2 Updater** creates additional parameter and measurement objects for all MAP file symbols that

- > are inside the configured address range,
- > match the specified name pattern and
- > for which no parameter and measurement object exists yet.

For defining the data type of parameter objects the **CANape** default record layouts are created if not available yet.

If the used MAP reader does not support data types the objects are created with data type `UBYTE`.



Note on Structures: Only objects are created that are within the specified address range. If you want to create child elements of structures, the entire structure must be in the specified address range. The structure with all data members is completely ignored if it is only partially in the address range.

To restrict the generation to individual data members, you can additionally use the name filter.

If structure generation is not activated (see `CREATE_STRUCTURES`), objects are only created for those MAP symbols that have no child elements.



Note: Starting with version 16.0, the option `IGNORE_VOID_POINTERS` has been eliminated and is always set to 1.

Parameter	Default Value	Description
<code>BYTE_ORDER</code>	""	Byte order for the objects to be created. Possible values: <code>INTEL</code> and <code>MOTOROLA</code> . If no byte order is specified, the global byte order of the input file is used.
<code>CREATE_INSTANCES_FOR_BASE_TYPE_ARRAYS</code>	0	When creating measurement arrays and value blocks, typedefs and instances are generated instead of the conventional objects <code>MEASUREMENT</code> and <code>CHARACTERISTIC</code> . <ul style="list-style-type: none"> > 0 = The conventional objects are created for all arrays. > 1 = Instances are created for all arrays regardless of the count of dimensions. > 2 = Instances are only created for arrays with more than 2 dimensions. For all other arrays the conventional objects are generated. Notice: <code>MEASUREMENT</code> and <code>CHARACTERISTIC</code> with more than 2 dimensions are only supported by CANape in form of instances.
<code>CREATE_STRUCTURES_FOR_CHILDREN</code>	0	If this option is active, the global settings of <code>CREATE_STRUCTURES</code> can locally be overridden within a <code>CREATION_RANGE</code> section for creating subelements. <p>Example:</p> <p>Goal: Only a special single structure element <code>A.B[1]</code> shall be generated, but the elements below <code>B</code> shall not be flat, but shall be created as substructure.</p> <p>Solution: Set this option to 1 and <code>CREATE_STRUCTURES</code> to 0 and specify name filter.</p>
<code>IGNORE_POINTERS_FOR_STRUCTURE_COMPONENTS</code>	0	If this option is active, all structure components with indirect addressing are ignored for creating structures. <p>Notice: All options defining restrictions regarding the structure components to be created, can be combined with each other.</p>

Parameter	Default Value	Description
IGNORE_STRUCTURE_COMPONENTS_PER_NAME	""	Structure components matching the specified name pattern (wildcard and regular expressions are allowed) are ignored when structures are created. Whether wildcards or regular expressions are used, is controlled by the existing option <code>USE_REGULAR_EXPRESSION</code> . Prerequisite: This option is only relevant with the <code>CREATE_STRUCTURES</code> option activated. Notice: All options defining restrictions regarding the structure components to be created, can be combined with each other.
IGNORE_VOID_POINTERS	1	Structure components and instances representing a pointer object of type <code>VOID</code> are ignored. Notice: Starting with version 16.0, this option has been eliminated and is always set to 1.
LENGTH	0	Length of the range
NAME	""	Name pattern for the objects to be created. Objects are only created for those MAP symbols whose names match the name pattern. If no name pattern is specified, only the given address range is evaluated. Notice: The filter is case-sensitive.
START	0	Start address of range in Linker MAP file
TYPE	""	Object type to be created. Possible values are: > <code>MEASURE</code> > <code>PARAMETER</code> > <code>MEASURE_WRITABLE</code> (measurement object with set <code>READ_WRITE</code> flag)
USE_REGULAR_EXPRESSION	0	> 0 = Name pattern is interpreted as wildcard expression. > 1 = Name pattern is interpreted as regular expression.

4.4.4 [EPK]

EPK addresses

This section contains settings for the updating of EPK addresses:

Parameter	Default Value	Description
ADDRESS_COUNT	0	Number of symbols to be used for updating the EPK addresses. This setting is only effective when the option <code>UPDATE_EPK</code> is set.
SYMBOL_x	""	x = 1..n Symbol names that are to be used for updating the EPK addresses. These settings are only effective if the option <code>UPDATE_EPK</code> is activated.

4.4.5 [COFF]

COFF reader

This section contains settings for the COFF reader. They are relevant only if you set COFF (19) or COFF settings auto detected (39) for the MAP reader:

Parameter	Default Value	Description
COFF_ADDRESS_MODE	8	Address mode: > 8 = Byte addressing > 16 = Word addressing Notice: Only relevant for COFF, not for COFF settings auto detected
COFF_FORMAT_MODE	0	Byte order: > 0 = Intel format > 1 = Motorola format Notice: Only relevant for COFF, not for COFF settings auto detected
COMPATIBLE_MODE	0	Array representation compatible to CANape 3.5 .
CONVERT_DOUBLETOFLOAT	0	Converts DOUBLE objects to FLOAT.
ENABLE_ENUM	1	Enables enumerations.
ENABLE_MULTIDIMARRAY	1	Multidimensional arrays are supported: > 0 = One-dimensional representation > 1 = Multidimensional representation
ENABLE_POINTER	1	Enables pointers.
MAP_MAX_ARRAY	16	Maximum count of expanded array elements
OLD_EXPAND_SYNTAX	0	Expand syntax compatible to CANape 2.0 .
REMOVE_UNDERLINE_PREFIX	0	Removes leading underscores.
SIMPLE_ARRAY_VIEW	0	Array representation without leading zeros

4.4.6 [ELF]

ELF reader

This section contains settings for the ELF/DWARF reader.

They are relevant only if you set ELF (30 or 31), COFF/DWARF (44) or EXE/DWARF (56) for the MAP reader.

However, the following options do not apply to the new MAP reader ELF with the number 131!



Note: If there is not any note about DWARF support in the option's description, the option is supported by all DWARF versions.

Parameter	Default Value	Description
ADD_ARRAY_BASE_ADDRESS_FOR_EACH_DIMENSION	0	For multidimensional arrays, the base addresses of each dimension are output. Notice: Use this option to solve problems with backward compatibility, e.g. if the compiler has changed the interpretation of

Parameter	Default Value	Description
		the multidimensional arrays - from multiple nested one-dimensional arrays to single array with multiple dimensions. This option is supported starting with DWARF2 and must be used together with ELF_USE_CPP_EXTENSION_DWARF2.
CALC_ARRAY_ITEM_SIZE	1	Calculation of the array item size. Notice: Supported starting with DWARF2
COFF_DWARF_ADDRESS_MODE	16	Sets address mode for COFF/DWARF: > 0 = Auto detect > 8 = Byte addressing > 16 = Word addressing
COMPUNIT_NAME_AS_PREFIX	0	Creates compile unit name (source file) prefix for STATIC variables, not for GLOBAL.
COMPUNIT_NAME_AS_PREFIX_ALL	0	Creates compile unit name (source file) prefix for STATIC and GLOBAL variables.
ELF_ADD_BASE_CLASSNAME_PREFIX	1	Adds the name of the base class in front of derived members. Notice: Supported only for DWARF1
ELF_ADD_INHERITED_CLASSNAME_PREFIX	1	Inserts the inherited class name in front of all inherited data members. The member name and inherited class name will be separated with scope separator (see ELF_SCOPE_SEPARATOR). > 0 = Feature is deactivated. > 1 = Insert only the name of the base class in which the data member is declared. > 2 = If the variable is inherited from multiple (sequence of) base classes, inserts 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) Notice: This option is only supported starting with DWARF2. For DWARF1 the value is always set to 1.
ELF_ARRAY_INDEX_FORM	0	Specifies the form of variable indexes expanded from array: > 0 = array._1_.22_.333_

Parameter	Default Value	Description
		<ul style="list-style-type: none"> > 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_BITFIELD_BYTESIZE_FROM_BASETYPE	0	<p>If the option is active, the reader uses the byte size of the bit fields from the referenced section <code>DW_TAG_base_type</code> (<code>DW_TAG_member</code>).</p> <p>If the option is not active, the byte size information is taken from bit fields itself.</p>
ELF_BOOL_AS_UNSIGNED	0	<p>The variables with data type <code>bool</code> will be generated as unsigned instead of signed.</p> <p>Notice: Supported starting with DWARF2</p>
ELF_CHECK_TYPE_ENDLESS_RECURSION	0	<p>If this option is active, reader will inspect type reference for endless recursion before processing it. If a recursion is detected, type reference will be ignored.</p> <p>Notice: Supported starting with DWARF2.</p> <p>Use this option in situation when reader stops processing ELF file, reporting Stack Overflow error. Reading of ELF file may slow down significantly when this option is active.</p>
ELF_DEMANGLE_SYMTAB_NAMES	1	<p>Use this option to configure the demangling of mangled symbol names.</p> <ul style="list-style-type: none"> > 0 = Do not demangle symbol names. > 1 = Simple demangling- Demangle only symbol names from debug section Variable names from symbol table are not demangled. > 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. Not supported for: DWARF1 > 3 = Tasking 3.2 Compiler Extension - Demangle symbols mangled by Tasking 3.2 Compiler for TriCore. <p>Notice: Starting with DWARF2 simple and extended demangling must be used together with option <code>ELF_USE_CPP_EXTENSION_DWARF2</code>.</p>
ELF_DOUBLE_AS_FLOAT	0	Interpretation of the double values as float

Parameter	Default Value	Description
ELF_ENABLE_ARRAY_DOT_DOT	0	<ul style="list-style-type: none"> > 0 = Removes two dots from array indexes created by some compilers. > 1 = Turns off automatic removing. Notice: Supported starting with DWARF2
ELF_ENUM_AS_INT	1	Interpretation of ENUM variables as INT
ELF_ERROR_ON_AMBIGUOUS	0	Reports errors when duplicated variables are found. Notice: Supported for DWARF1. Starting with DWARF2, this option is only supported together with ELF_USE_CPP_EXTENSION_DWARF2.
ELF_FILE_READ_BUFFER_SIZE	0	Size of the read buffer for ELF file to optimize speed.
ELF_FOLLOW_REDIRECTION_REFERENCES	0	Depending on the compiler used, for static variables declared in a function, the function name prefixes are missing. This can lead to name conflicts and thus to a very slow processing time. If this option is activated, additional special DWARF attributes are evaluated to avoid conflicts. Notice: It is recommended to activate this option. Supported starting with DWARF3.
ELF_FORCE_DEBUG_SECTION_VERSION	0	Forces reader to read symbols from specific debug sections. Possible values are: <ul style="list-style-type: none"> > -1 = Read symbols from all available debug sections. > 0 = Read symbols from single debug section. (Notice: If DWARF2 debug section is presented, DWARF1 debug section is ignored.) > 1 = Read symbols from DWARF1 (.debug) debug section. > 2 = Read symbols from DWARF2/3 (debug_info) debug section. > 3 = Read symbols from NEC (.vdebug) debug section.
ELF_FORCE_SYMBOL_TABLE	0	Reads the information from symbol table: <ul style="list-style-type: none"> > 0 = Reads symbol table only when the DWARF section is not available. > 1 = Ignores DWARF section and read the symbol table. > 2 = Reads both the symbol table and DWARF section.
ELF_IGNORE_B_MEMBERS_DWARF2	0	Activate this option to ignore data members of classes, unions or structures containing the "__b_" prefix in its name.

Parameter	Default Value	Description
		<p>Possible values are:</p> <ul style="list-style-type: none"> > 0 = The option is inactive > 1 = Ignores data members beginning with the "<code>__b_</code>" prefix. This option was customized for Diab Data 5.5.1.0 compiler. > 2 = Ignores data members beginning with the "<code>__b_</code>" prefix. This option was customized for Tricore VX-ToolSet compiler. <p>Notice: This option is supported starting with DWARF2 and only works together with <code>ELF_USE_CPP_EXTENSION_DWARF2</code>.</p>
<code>ELF_IGNORE_GLOBAL_PADDING</code>	0	<p>Special option for Metrowerks compiler: Compile unit tag is directly followed by padding tag, so the rest of the variables in compile unit are not processed. With this option the padding tags which are direct children of the compile units are ignored.</p> <p>Notice: Supported only for DWARF1</p>
<code>ELF_IGNORE_LEADING_UNDERSCORE</code>	0	Removes the leading underscore from all variable names.
<code>ELF_LOG_FILE</code>	0	<p>Creates a log file (dump) from the processed ELF file. This option is available only for <code>.vdebug</code> section (NEC compiler):</p> <ul style="list-style-type: none"> > 0 = Off > 1 = Log > 3 = Dump
<code>ELF_MAP_VERSION_FLAG</code>	0	<p>Uses special version of ELF reader compatible with older versions.</p> <ul style="list-style-type: none"> > 0 = Current CANape version > 1 = CANape version 4.0.20 or earlier
<code>ELF_NO_BASE_ADDRESS</code>	0	Use this option to disable writing addresses of the classes, structures, unions and arrays to the output.
<code>ELF_NO_BASE_ADDRESS_DATATYPE</code>	0	Creates base address of structures, classes, unions and array elements without data type information.
<code>ELF_NO_DOT_AFTER_ARRAYITEM</code>	0	Removes "." placed behind the array item index as separator between index and next name parts.
<code>ELF_REPLACE_ANGLEBRACKETS</code>	1	Replacing of angle brackets (" <code><</code> " und " <code>></code> ") occurring within symbol names by " <code>_</code> " character.
<code>ELF_REPLACE_CLASSNAME_SCOPESEPARATOR</code>	0	<p>Replaces occurrence of the scope separator <code>::</code> also in the name of the class/union/structure type with scope separator defined by option <code>ELF_SCOPE_SEPARATOR</code>.</p> <ul style="list-style-type: none"> > 0 = Off

Parameter	Default Value	Description
		<ul style="list-style-type: none"> > 1 = Replaces only symbols found in debug section. > 2 = Replace symbols found in debug section and symbol table. <p>Notice: This option does not have effect when reading symbols from DWARF1 debug section.</p> <p>Option value 2 fixes also following issues: Scope separator is sometimes not correctly replaced if it occurs in name of static class members. Some characters which do not comply with ASAP2 standard are sometimes not correctly replaced with "_". Not supported for: DWARF1</p>
ELF_REPLACE_DOT_CHARACTER	0	Use this option to turn on/off replacing of dot "." occurring within symbol names by "_" character.
ELF_SCOPE_PREFIX_FOR_STATIC_VARIABLES	0	Use this option to always insert compile-unit name and function name as prefix of static variables. Notice: The option is supported starting with DWARF2 and replaces <code>COMPUNIT_NAME_AS_PREFIX</code> (which does not work for static variables flagged as global in the symbol table).
ELF_SCOPE_SEPARATOR	"_."	String which is used in names of inherited data members as separator between name of base class and base class member.
ELF_SKIP_NONVAR_TYPEDEF_DWARF2	1	Skips processing of the type definition when the reader is currently not reading any variable. Notice: Supported starting with DWARF2
ELF_SKIP_NULL_RANGE_ARRAYS	0	For ELF files with multiple definitions of arrays: Ignore definitions with size 0. Notice: Supported starting with DWARF2
ELF_SKIP_UNEXPECTED_TAGS	1	Ignoring all unknown tags. Notice: Supported starting with DWARF2
ELF_SYMTAB_SEARCH_MODE	0	<ul style="list-style-type: none"> > 1 = The ELF reader takes the first symbol having the same address as the processed variable. When this value is set, the reader can potentially pick a wrong symbol if multiple variables share the same address. > 2 = The ELF reader uses address and name of the variable from debug section to identify the appropriate symbol from symbol table. <p>Notice: This option only works together with <code>ELF_DEMANGLE_SYMTAB_NAMES</code>.</p>

Parameter	Default Value	Description
ELF_TRY_READ_CORRUPTED_COMPILE_UNITS	0	If the ELF file contains compile units with corrupted length information, try to read the file without reporting and error.
ELF_TRY_RESOLVE_FORWARD_DECLARATIONS	0	If this option is active, the ELF reader searches for full specifications of forward declared data types. Notice: Use this option when type definition of found variables does not match the type definition defined in source code, even though your binary file contains debug information. This option may slow down processing of the input file. This option is supported starting with DWARF2.
ELF_TRY_RESOLVE_MEMADDR_FROM_SYMBOLTABLE	0	For static data members without valid address information in debug section the ELF reader tries to find the symbol with an extended symbol name in symbol table. Notice: This option is supported starting with DWARF2 and must be used together with ELF_USE_CPP_EXTENSION_DWARF2.
ELF_TRY_RESOLVE_VARADDR_FROM_SYMBOLTABLE	0	Searches in the symbol table missing addresses of variables declared as External. Notice: This option is supported starting with DWARF2 and must be used together with ELF_USE_CPP_EXTENSION_DWARF2.
ELF_UNDERLINE_PREFIX	0	Creates "_" prefix for all global variables names.
ELF_UNDERLINE_PREFIX_PREFIX	1	Creates "_" prefix for compile unit prefix and subfunction prefix, too.
ELF_USE_CPP_EXTENSION_DWARF2	0	Extended handling of inherited classes, duplicated variables, demangling of symbol names.
ELF_USE_OLD_VERSION	0	Activates old ELF reader (compatibility switch). Notice: Supported only for DWARF1
ELF_WRITE_PTR_AS_INT	0	If this option is active, pointer objects are not being ignored but interpreted as INTs. Notice: This option is necessary if tables shall be read-in by pointer-based calibration methods.
ELF_WRITE_VOID_PTR_AS_INT	0	If this option is active, pointer objects of type VOID are not being ignored but interpreted as integer.
FUNCTION_NAME_AS_PREFIX	0	Creates function name prefix for STATIC variables defined in function scope.
INCREMENT_BIT_FIELDS_ADDRESS_DISABLED	0	For bit field variables with bit offset greater than 7, the bit offset is divided by 8 and the byte address is incremented. This option is

Parameter	Default Value	Description
		active only when used with any bit inversion flags. Notice: This option replaces the deprecated option <code>INCREMENT_BIT_FIELDS_ADDRESS</code> .
<code>INVERTED_BIT_FIELDS</code>	0	The bit offset of variables inside bit field structure is inverted: <ul style="list-style-type: none"> > 0 = No inversion > 1 = The bit offset is inverted. The flag is VALID ONLY IN MOTOROLA format. > 4 = Inverted bit fields (Tasking Tricore Vx) > 5 = Inverted Bit Fields v5
<code>MAP_MAX_ARRAY</code>	16	Maximum count of expanded array elements

4.4.7 [IEEE]

IEEE reader

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

Parameter	Default Value	Description
<code>IGNORE_COMPILER_GLOBAL_VARIABLES</code>	0	Ignores variables which have the attribute <code>compiler global variable</code> .
<code>MAP_ARRAY_LEADING_ZERO</code>	1	Generates array element names with leading zeros.
<code>MAP_BIT_SIZE_OF_INTEGER</code>	0	Bit size of integer: <ul style="list-style-type: none"> > 0 = MAUs² per address * bit per MAU² > 8 = 8 Bit > 16 = 16 Bit > 32 = 32 Bit
<code>MAP_BYTE_SIZE_OF_ENUMERATION</code>	-1	Byte size of enumerations: <ul style="list-style-type: none"> > -1 = MAUs² per address > 0 = C Standard (smallest integer type to save all values) > 1 = 1 Byte > 2 = 2 Bytes > 4 = 4 Bytes
<code>MAP_CASE_SENSITIVE_LINKER</code>	1	Case sensitive linker
<code>MAP_DIRECT_BIT_BASE_ADDR</code>	0xFD00	Base address for direct addressable bits
<code>MAP_DIRECT_BIT_BIT_OFFSET_MASK</code>	0x0007	Bit mask for direct addressable bits
<code>MAP_DIRECT_BIT_BYTE_OFFSET_MASK</code>	0xFFFF8	Byte mask for direct addressable bits
<code>MAP_FIX_TASKING_TYPE_ERROR</code>	1	Fixes tasking IEEE output type error

² MAU is the abbreviation for **Minimum Addressable Unit**.

Parameter	Default Value	Description
MAP_IGNORE_INSTRUCTION_ADDRESS	1	Ignores exported symbols with type index 15 (instruction address). Example: The Mitsubishi linker uses the type index 15 for all exported symbols.
MAP_IGNORE_LEADING_UNDERSCORE	1	Ignores leading underscores of exported symbol names.
MAP_MAX_ARRAY	16	Maximum count of expanded array elements

4.4.8 [GREENHILL]

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

Parameter	Default Value	Description
GHS_INCLUDE_RODATA	0	If this option is set, the section <code>.rodata</code> of the MAP file is read, otherwise it is ignored.
GHS_REMOVE_UNDERLINES	0	Removes leading "_" in symbol names.

4.4.9 [PDB]

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

However, the following options do not apply to the new MAP reader `PDB` with the number 154!

Parameter	Default Value	Description
MAP_MAX_ARRAY	16	Maximum count of expanded array elements
PDB_ARRAY_INDEX_FORM	-1	Specifies form of variable indexes expanded from array: <ul style="list-style-type: none"> > -1 = array[001][022][333] > 0 = array_1_22_333_ > 1 = array[1][22][333] > 2 = array_1_22_333_ > 3 = array_01_22_333_ > 4 = array_01_22_333_
PDB_MAX_TYPE_IN_DETAIL	10	Uses this option to get a deeper resolution of substructures.
PDB_REMOVE_LEADING_UNDERSCORE	0	Removes the leading underscore from all variable names.
PDB_RESOLVE_BITFIELD_ADDRESS	0	If this option is set, bit offsets which are greater than 7, are adjusted automatically and allocated with the starting address. Sample: Start address 0x1000 and bit offset 17 are converted to start address 0x1002

Parameter	Default Value	Description
		and bit offset 1 when using this option.
PDB_UNDERLINE_PREFIX	0	Creates "_" prefix for global variables names and data member names.
PDB_WRITE_PTR_AS_INT	0	If this option is activated, pointer objects are not ignored, but interpreted as INTs. Notice: This option is required when tables are read from pointer-based calibration methods.

4.4.10 [DATATYPES]

Updating data types This section contains settings for the update of data types:



Note: A prerequisite for all options in this section is the global switch `ENABLE_UPDATE`. If this option is set, data type updating is permitted in principle.

Parameter	Default Value	Description
DELETE_ADDRESS_OFFSETS	0	If this option is set, the address offsets in <code>IF_DATA CANAPE_EXT</code> will be deleted for those objects whose data type has changed.
DELETE_ALL_ADDRESS_OFFSETS	0	This could be used if objects with a big bit offset (larger than 7) are described in the source A2L file by a combination of an address offset and a bit offset ≤ 7 .
ENABLE_UPDATE	0	Enables the updating of data types for measurement objects and scalar calibration objects (VALUES). Notice: As a global switch, this option is a prerequisite for all other options in this section.
UPDATE_BASE_TYPES	0	Updates the base data types.
UPDATE_BIT_OFFSETS	0	Updates also the bit mask.
UPDATE_CURVES_AND_MAPS	0	If this option is set, the update of data types also for maps, curves and shared axes is enabled additionally to the scalar objects.

4.4.11 [CAL_PARAM_GROUPS]

Parameter group Parameter groups contain a group of characteristic objects indicated by a single pointer in the pointer table.

Creating parameter groups In this section you can configure how to create from HEX and MAP file the parameter groups for the calibration method AUTOSAR Single Pointered.

In the A2L file, `IF_DATA` information of type `CANAPE_CAL_METHOD` and `CANAPE_ADDRESS_UPDATE` is generated from this. The required AML is added if necessary, unless it is not in the A2L file already.



Note: Prerequisite for creating parameter groups is always that the indicated symbol must be found in the MAP file. If not, no parameter groups are created.

Parameter	Default Value	Description
ADDRESS_FORMAT	INTEL	Specifies the address format in the HEX file. Possible values are: > INTEL > MOTOROLA > S12X_COSMIC
CREATE	0	If this option is active, parameter groups are created from the MAP file. Precondition: This option only works if a HEX file is specified in the command line.
ORIGINAL_POINTER_TABLE_START_ADDRESS	""	Specifies the name of the MAP symbol for the original base pointer table.
POINTER_SIZE	4	Specifies the pointer size in bytes. The count of pointers in the pointer table is determined from the size of the pointer table and the pointer size. If the value is 0, the actual size is determined respectively from the MAP file.
POINTER_TABLE_SIZE	0	Specifies the size of the pointer table in bytes. If the value is 0, the actual size is determined respectively from the MAP file.
POINTER_TABLE_START_ADDRESS	""	Specifies the name of the MAP symbol for the pointer table in RAM memory.

4.4.12 [SYNTAX_TOLERANCE]

Tolerance settings

This section contains tolerance settings for the ASAP2 reader. Without additional configuration, the A2L file will be parsed strictly according to the ASAP2 standard. However, it is possible to tolerate special errors if configured so in this section.

Parameter	Default Value	Description
DIVERSE_BLOCK_BRACKETS	0	See section Cross-Tool Tolerance Settings.
IDENTS_START_WITH_DIGIT	0	See section Cross-Tool Tolerance Settings.
IGNORE_AML	0	See section Cross-Tool Tolerance Settings.
IGNORE_UNKNOWN_IF_DATA	0	See section Cross-Tool Tolerance Settings.
KEYWORD_AS_SYMBOL	0	See section Cross-Tool Tolerance Settings.
KEYWORDS_ALL_VERSIONS	0	See section Cross-Tool Tolerance Settings.
NESTED_COMMENTS	0	See section Cross-Tool Tolerance Settings.
PROJECT_NO_WITH_VALUE	0	See section Cross-Tool Tolerance Settings.
PURE_OBJECT_LIST	0	See section Cross-Tool Tolerance Settings.
RESERVED_WITH_DATATYPE	0	See section Cross-Tool Tolerance Settings.

Parameter	Default Value	Description
<code>SPECIAL_FORMAT_SYNTAX</code>	0	See section Cross-Tool Tolerance Settings .
<code>SPECIAL_IDENT_INNER</code>	""	See section Cross-Tool Tolerance Settings .
<code>SPECIAL_IDENT_START</code>	""	See section Cross-Tool Tolerance Settings .
<code>SYMBOL_AS_STRING</code>	0	See section Cross-Tool Tolerance Settings .

4.5 Warning Levels

Different warning levels

The warnings in the log file are grouped into different warning levels:

Level	Messages/Meaning
0	<ul style="list-style-type: none"> > Error messages/warnings related to syntax errors in the A2L file, but are tolerated by the ASAP2 Updater and do not lead to abortion > Messages about ASAP2 keywords that were available in the input file, but due to the set ASAP2 version cannot be saved in the result file > Messages about objects that are completely missing in the result file, because they cannot be saved due to the set ASAP2 version (e.g. data type <code>INT64</code>) > Messages about obsolete command line parameters that are no longer supported
1	<ul style="list-style-type: none"> > Messages of objects that are specified in the filter file, but not found in the A2L file
2	<ul style="list-style-type: none"> > Messages about MAP symbols that are specified in the A2L file, but not found in the MAP file > Messages about objects whose addresses could not be updated > Messages about failure to address update due to incorrect AML description
3	<ul style="list-style-type: none"> > All error messages/warnings that are created from a MAP reader

4.6 Interface to MAP Readers

MAPCONV.TMP

The former interface between the MAP readers and the **ASAP2 Updater** is the file `MAPCONV.TMP`.

It can be created as needed by the option `CREATE_MAPCONV_TMP_FILE` in the current working directory. This file 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 types.

Syntax

Each line of that file has the following syntax:

```
<name> <address> <data type> <byte size>
[Bitfield: <bit offset>:<bit length>]
```

With:

<code><name></code>	Symbol name from MAP file
<code><address></code>	Address string in hexadecimal form, optionally together with the address extension in decimal form (e.g. 3:0F815)

<data type>	<p>Data type, can be one of: Array, Class, Union, FunctionType, Ptr to <data type>, UnknownType, EnumType:<EnumName>, TypedefType:<TypeDefName>, Function, NoType, void, Char, UChar, WChar, Int32, UInt32, Float16, Float32, Bool, Long, ULONG, Double, Byte, UByte, Short, UShort, Int64, UInt64</p> <p>With: <EnumName> name of the enumeration type <TypeDefName> name of the typedef</p>
<byte size>	Size in bytes
<bit offset>	Bit offset of the bitfield
<bit length>	Bit length of the bitfield

4.7 Examples

Demo for usage of the ASAP2 Updater

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 A2L input file `Demo.a2l` and the MAP file `demo.map`.

Input file `Demo.a2l`

`Demo.a2l` contains the measurement 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`.

Map file `Demo.map`

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



Sample 1:

Call the **ASAP2 Updater** via the command line:

```
ASAP2Updater -I demo.a2l -A Demo.map - O Result1.a2l -L Log.txt
-T Updater1.INI
```

The used INI file has the settings:

```
[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.a21` 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 `__amp1` (which has priority against the object name) is not found in MAP file.



Sample 2:

Call the **ASAP2 Updater** via the command line:

```
ASAP2Updater -I Demo.a21 -A Demo.map -O Result2.a21 -L Log.txt
-T Updater2.INI
```

The used INI file has the settings:

```
[OPTIONS]
MAP_FORMAT = 0
CREATION_RANGES = 0
FILTER_MODE = 4
IGNORE_MAP_REFERENCES = 1
```

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

Because of the option `FILTER_MODE=4` all measurement objects are now removed and the result file `Result2.a21` contains only the parameter `amplitude` which has now an updated address because of the option `IGNORE_MAP_REFERENCES=1`.



Sample 3:

Call the **ASAP2 Updater** via the command line:

```
ASAP2Updater -I Demo.a21 -A Demo.map -O Result3.a21 -L Log.txt
-T Updater3.INI
```

The used INI file has the settings:

```
[OPTIONS]
MAP_FORMAT = 0
CREATION_RANGES = 1
FILTER_MODE = 0
IGNORE_MAP_REFERENCES = 1
```

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

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

Because of the defined creation range the result file `Result3.a21` now contains two additional measurement objects `channel3` and `channel4`.

5 ASAP2 Merger

In this chapter, you will find the following information:

5.1	Functionality	page 94
5.2	Command Line Parameters	page 94
5.3	Exit Code	page 95
5.4	Initialization File [OPTIONS] [SLAVE_X] [TYPE_SPECIFIC_CONFLICT_SOLUTION] [SYNTAX_TOLERANCE]	page 95
5.5	Examples	page 102

5.1 Functionality

Merging of multiple A2L files

ASAP2 Merger merges multiple A2L files or A2L fragments into a common A2L file. One of the source files must be specified as the Master; the other source files are Slaves.

Master and Slave


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.

The objects in the result file can optionally receive a prefix and/or a suffix, which indicates from which source file these objects originate.

5.2 Command Line Parameters

Program call

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

-M <name>	Name of the A2L master file.
-L <name>	Name and path of the log file for warnings and error messages.
-S <name>	Name of an A2L slave file. The first module of the slave file will be merged into the first module of the master file by default. The -s parameter can be used multiple in command line to merge more than two files at once.
	 Note: The parameter -s is optional. The slaves can be configured completely in the INI file. This is recommended in particular with many slaves, since it is clearer. Both configuration variations cannot be used in combination!
-O <name>	Name of the A2L file to be generated.

-P <name>	Name for the initializing file, if this file is not located in working directory or if its name is not <code>MERGER.INI</code> .
-@<name>	Name of a text file with all command line parameters. The preceding hyphen is optional.



Attention: There must be no space characters between @ and the file name. A call is e.g. `ASAP2Merger @CMDLINE.TXT`.



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



Sample: Calling

```
ASAP2Merger -M Master.a21 -S slave1.a21 -S slave2.a21 -O
Result.a21
```

Or:

```
ASAP2Merger @CMDLINE.TXT
```

With `Cmdline.txt` file:

```
-M Master.a21
-S slave1.a21
-S slave2.a21
-O Result.a21
-P Merger.INI
-L Log.txt
```

5.3 Exit Code

Determining success The success of the merge process 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
3	No license available

5.4 Initialization File

Directory

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.

Behavior

In the initialization file the behavior of the merger can be controlled in addition to the command line parameters.

However, the configuration of slaves in the INI file is only applied if no `-s` parameter

is used in the command line.

General

For general information on the structure and usage of the initialization file, see section [Configuration of the Command Line Tools](#) on page 8.

5.4.1 [OPTIONS]

Common settings This section contains common settings:

Parameter	Default Value	Description
ACCEPT_LONG_STRINGS	0	See section Cross-Tool General Options .
ASAP2_VERSION	171	See section Cross-Tool General Options .
AVOID_MULTIPLE_OBJECTS	0	<ul style="list-style-type: none"> > 0 = In case of name conflicts, both objects are accepted for the result file. An appropriate warning is output. > 1 = In case of name conflicts, the first object is accepted for the result file; the second one is ignored. > 2 = In case of name conflicts, the last object is accepted for the result file; in doing so an already existing one is overwritten. > 3 = In case of name conflicts, unique names are created. The second name is extended by <code>_CopyX</code> with a serial no. X. > 4 = In case of name conflicts, unique names are created for all objects involved. The name is expanded with <code>_M</code> (for master file) and <code>_SX</code> with consecutive no. X (for slave files). If this name also already exists, it is expanded with <code>_CopyX</code> with a consecutive no. X. <p>Notice: On checking, the namespaces defined according to ASAP2 standard are used. If this option is set, the option <code>CHECK_FOR_DUPLICATE_NAMES</code> is mandatory and is activated automatically by the ASAP2 Merger if necessary. You can define object-specific deviations of this option (see section [TYPE_SPECIFIC_CONFLICT_SOLUTION] on page 100).</p>
CHECK_FOR_DUPLICATE_NAMES	1	<ul style="list-style-type: none"> > 0 = No check for duplicate names (better performance).
CREATE_IF_DATA_CANAPE	0	Creates the AML for <code>CANAPE_EXT</code> in output file if not exists in master file.
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	See section Cross-Tool General Options .

Parameter	Default Value	Description
CREATE_LOCAL_ALIGNMENT	1	> 0 = Local alignment information are not created.
CREATE_LOCAL_BYTE_ORDER	1	> 0 = Local byte order information are not created.
CREATE_PARENT_GROUP_AS_FUNCTION	0	If this option is set, functions are created instead of groups in the result file with option <code>PARENT_FOR_SLAVE_x</code> .
DEFAULT_ENCODING_FOR_READ	""	See section Cross-Tool General Options.
DISABLE_SUFFIXES	0	This option is no longer supported.
GROUP_FOR_SLAVE_x	""	This option is no longer supported. Use <code>GROUP_FOR_SLAVE</code> in the section of the respective slave configuration [SLAVE_x] instead.
HIDE_PROGRESS_VIEW	0	See section Cross-Tool General Options.
IGNORE_GROUP_COMMENTS	0	If group contents are merged (see option <code>MERGE_GROUP_CONTENTS</code>) 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.
INCLUDE_SAVE_MODE	0	<p>If this option is activated, the assignment of objects to included files is retained. After merging, the objects are saved back to their original file, and corresponding include instructions are generated in the result file of the merger.</p> <p>Attention: All original include files are overwritten with this option! This is imperative for some applications of the ASAP2 Merger (e.g., suffix generation, merging group contents, resolving name conflicts).</p> <p>The assignment to an include file is retained for the following objects: CHARACTERISTIC, MEASUREMENT, AXIS_PTS, FUNCTION, GROUP, COMPU_METHOD, COMPU_TAB, COMPU_VTAB, COMPU_VTAB_RANGE, INSTANCE, RECORD_LAYOUT, UNIT, FRAME, USER_RIGHTS, TYPEDEF_xx, TRANSFORMER, TYPEDEF_xx, TRANSFORMER, BLOB</p> <p>AML definitions, module-global data and global IF_DATA blocks are always transferred to the main file.</p> <p>> 0 = Assignment to include files is NOT retained in the result file.</p> <p>> 1 = Assignment to include files is retained in the result file.</p>
MAX_INI_STRING_LENGTH	2048	See section Cross-Tool General Options.
MERGE_ALL_TO_FIRST_MASTER_MODULE	0	If this option is set, all modules of all slave files are merged into the first module of the master

Parameter	Default Value	Description
		file. Otherwise, only the first module of each slave is merged into the first master module.
MERGE_GROUP_CONTENTS	0	<ul style="list-style-type: none"> > 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. <p>Merge of contents is only possible with disabled suffix generation (see <code>DISABLE_SUFFIXES</code>).</p>
MINIMIZE_RESULT_FILE	0	See section Cross-Tool General Options.
NUMBER_OF_SLAVES	0	<p>Specifies the number of slave files.</p> <p>Each slave file has its own section in the INI file: <code>SLAVE_x</code> with x = continuous number, counting starts with 1.</p> <p>For these additional slave files, you can define separate settings each that are described in section <code>[SLAVE_X]</code>.</p> <p>Notice: This option is only evaluated if no slave file is specified in the command line. If slave files have been configured already via <code>-S</code> in the command line, this option is completely ignored.</p>
PARENT_FOR_SLAVE_x	""	This option is no longer supported. Use <code>PARENT_FOR_SLAVE</code> in the section of the respective slave configuration <code>[SLAVE_x]</code> instead.
PREFIX_FOR_MASTER	""	<p>Specifies the prefix that is to prepend to the object names for the objects from the master file.</p> <p>The prefix must follow the ASAP2 standard (only letters, digits, dot, underscore and not starting with a digit or dot).</p> <p>It is placed directly prefixing the object name without any additional separator.</p>
PREFIX_SUFFIX_USAGE	3	<p>Defines how to apply the configured suffixes and prefixes for the master and slave files.</p> <p>The name extension is done before the actual merge. If name conflicts occur, the configured conflict resolution is applied (see <code>AVOID_MULTIPLE_OBJECTS</code> or section <code>[TYPE_SPECIFIC_CONFLICT_SOLUTION]</code>).</p> <p>The name extension is applied:</p> <ul style="list-style-type: none"> > 0 = to all objects. With this option, no name conflicts should occur if a unique prefix is specified for all files involved. > 1 = to variables, functions and groups only. For all other objects, the original names are kept.

Parameter	Default Value	Description
		<ul style="list-style-type: none"> > 2 = to all objects, but not variables and functions. For variables and functions, the original names are kept. This option value equals the obsolete configuration with <code>DISABLE_SUFFIXES = 0</code>. > 3 = only if name conflicts occur and the conflict resolution <code>AVOID_MULTIPLE_OBJECTS</code> is set to 4.
<code>PRESERVE_GLOBAL_COMMENTS</code>	1	See section Cross-Tool General Options.
<code>REMOVE_DUPLICATE_IF_DATA_CANAPE_EXT</code>	0	See section Cross-Tool General Options.
<code>REMOVE_ROOT</code>	0	See section Cross-Tool General Options.
<code>REMOVE_TRAILING_DIMENSION_VALUES_1</code>	0	See section Cross-Tool General Options.
<code>SIGNIFICANT_DIGITS_FOR_DOUBLE</code>	12	See section Cross-Tool General Options.
<code>SUFFIX_FOR_MASTER</code>	"_M"	Specifies the suffix that is to be added to the object names for the objects from the master file. The suffix must follow the ASAP2 standard (only letters, digits, dot, underscore and not starting with a digit). It is added directly to the end of the object name without any additional separator.
<code>SUPPRESS_VERSION_WARNINGS</code>	0	See section Cross-Tool General Options.
<code>WARNING_FOR_MISSING_ADDRESS_INFORMATION</code>	0	See section Cross-Tool General Options.
<code>WARNING_FOR_MISSING_REFERENCES</code>	0	Categorizes the messages about missing references in the input files as follows: <ul style="list-style-type: none"> > 0 = Error messages > 1 = Warnings > 2 = Information
<code>WRITE_UTF8</code>	1	See section Cross-Tool General Options.

5.4.2 [SLAVE_X]

Organization with many slaves

The following table contains all settings for A2L slave files. Only those sections are evaluated whose names match the set option `NUMBER_OF_SLAVES`.

In addition, no slave files must be configured at the same time using `-s` in the command line.

Parameter Name	Default Value	Description
<code>FILENAME</code>	""	File name and/or file path of the A2L slave file.
<code>GROUP_FOR_SLAVE</code>	""	Adds all objects of the slave file to the master group with given name. The master group is created automatically if it does not exist.
<code>PARENT_FOR_SLAVE</code>	""	The complete object hierarchy of the slave file is inserted into the master group with given name.

Parameter Name	Default Value	Description
		The master group is created automatically if it does not exist. All measurement 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.
PREFIX	""	Specifies the prefix that is to prepend to the object names for the objects from the slave file. The prefix must follow the ASAP2 standard (only letters, digits, dot, underscore and not starting with a digit or dot). It is placed directly prefixing the object name without any additional separator.
SUFFIX	"_Sx"	Specifies the suffix that is to be added to the object names for the objects from the slave file. The suffix must follow the ASAP2 standard (only letters, digits, dot, underscore and not starting with a digit). It is added directly to the end of the object name without any additional separator. The "x" corresponds to the number x in the respective section.

5.4.3 [TYPE_SPECIFIC_CONFLICT_SOLUTION]

General information

This section contains settings for object-specific deviations from the option `AVOID_MULTIPLE_OBJECTS`.

As default applies the value set there.

Options for object-specific overwriting

With this option, the handling of name conflicts specified with `AVOID_MULTIPLE_OBJECTS` can be overwritten differently for each object type.

Advantage

Name conflicts for variables, for example, can be handled differently than name conflicts for conversion rules.

Allowed values and description

Possible values and description of the values correspond those of `AVOID_MULTIPLE_OBJECTS` and are the following:

- > 0 = In case of name conflicts, both objects are accepted for the result file. An appropriate warning is output.
- > 1 = In case of name conflicts, the first object is accepted for the result file; the second one is ignored.
- > 2 = In case of name conflicts, the last object is accepted for the result file; in doing so an already existing one is overwritten.
- > 3 = In case of name conflicts, unique names are created. The second name is extended by `_CopyX` with a serial no. X.
- > 4 = In case of name conflicts, unique names are created for all objects involved. The name is expanded with `_M` (for master file) and `_SX` with consecutive no. X (for slave files). If this name also already exists, it is expanded with `_CopyX` with a consecutive no. X.

Notice: On checking, the namespaces defined according to ASAP2 standard are used.

If this option is set, the option `CHECK_FOR_DUPLICATE_NAMES` is mandatory and is

activated automatically by the **ASAP2 Merger** if necessary.

Value = 0

If no value is specified for an object type, the value of `AVOID_MULTIPLE_OBJECTS` applies to this object type.

Parameter	Objekt Type Description
CONVERSION_METHODS	This option applies to conversion methods.
CONVERSION_TABLES	This option applies to conversion tables.
FRAMES	This option applies to frames.
FUNCTIONS	This option applies to functions.
GROUPS	This option applies to groups.
MEMORY_SEGMENTS	This option applies to memory segments.
RECORD_LAYOUTS	This option applies to record layouts.
TRANSFORMERS	This option applies to transformers.
TYPEDEFS	This option applies to type definitions.
UNITS	This option applies to units.
VARIABLES	This option applies to variables.
VARIANT_CRITERIA	This option applies to variant criteria.

5.4.4 [SYNTAX_TOLERANCE]

Tolerance settings

This section contains tolerance settings for the ASAP2 reader. Without additional configuration, the A2L file will be parsed strictly according to the ASAP2 standard. However, it is possible to tolerate special errors if configured so in this section.

Parameter	Default Value	Description
DIVERSE_BLOCK_BRACKETS	0	See section <code>Cross-Tool Tolerance Settings</code> .
IDENTS_START_WITH_DIGIT	0	See section <code>Cross-Tool Tolerance Settings</code> .
IGNORE_AML	0	See section <code>Cross-Tool Tolerance Settings</code> .
IGNORE_UNKNOWN_IF_DATA	0	See section <code>Cross-Tool Tolerance Settings</code> .
KEYWORD_AS_SYMBOL	0	See section <code>Cross-Tool Tolerance Settings</code> .
KEYWORDS_ALL_VERSIONS	0	See section <code>Cross-Tool Tolerance Settings</code> .
NESTED_COMMENTS	0	See section <code>Cross-Tool Tolerance Settings</code> .
PROJECT_NO_WITH_VALUE	0	See section <code>Cross-Tool Tolerance Settings</code> .
PURE_OBJECT_LIST	0	See section <code>Cross-Tool Tolerance Settings</code> .
RESERVED_WITH_DATATYPE	0	See section <code>Cross-Tool Tolerance Settings</code> .
SPECIAL_FORMAT_SYNTAX	0	See section <code>Cross-Tool Tolerance Settings</code> .
SPECIAL_IDENT_INNER	""	See section <code>Cross-Tool Tolerance Settings</code> .
SPECIAL_IDENT_START	""	See section <code>Cross-Tool Tolerance Settings</code> .
SYMBOL_AS_STRING	0	See section <code>Cross-Tool Tolerance Settings</code> .

5.5 Examples

- Usage of the ASAP2 Mergers** 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.a21`, `Slave1.a21` and `Slave2.a21`.
- Slave1.a21** `Slave1.a21` contains two measurement signals `channel11` and `channel12` which are assigned to the group `SlaveGroup`.
- Slave2.a21** `Slave2.a21` contains two measurement signals `channel13` and `channel14` which are assigned to a group with the same name `SlaveGroup`.
- Master.a21** `Master.a21` contains the measurement signal `channel15` which is assigned to the group `MasterGroup`. Furthermore, it contains an empty group `SlaveGroup` and also a measurement signal `channel11` with modified comment.



Sample 1:

Call the **ASAP2 Merger** via the command line:

```
ASAP2Merger -M Master.a21 -O Result1.a21 -P Merger1.INI -L Log.txt
```

The used INI file has the settings:

```
[OPTIONS]
AVOID_MULTIPLE_OBJECTS=1
MERGE_GROUP_CONTENTS = 1
NUMBER_OF_SLAVES = 2
PREFIX_FOR_MASTER =
SUFFIX_FOR_MASTER =
```

```
[SLAVE_1]
FILENAME=Slave1.a21
PREFIX=
SUFFIX=
```

```
[SLAVE_2]
FILENAME=Slave2.a21
PREFIX=
SUFFIX=
```

The result file `Result1.a21` then contains 5 signals: `channel11`, `channel12`, `channel13`, `channel14` and `channel15` and the two groups `SlaveGroup` and `MasterGroup`.

Because of the suffix and prefix option `DISABLE_SUFFIXES = 1` the original group names are not modified and their contents can be merged: `SlaveGroup` contains all signals `channel11` to `channel14`. `MasterGroup` contains only `channel15`.

Because of the option `AVOID_MULTIPLE_OBJECTS = 1` the result file contains only one `channel11` signal: The one from master file with modified comment is used because the master file was the first file named in command line.

**Sample 2:**

Now use the second INI file and call the **ASAP2 Merger** via the command line:

```
ASAP2Merger -M Master.a21 -o Result2.a21 -L Log.txt -P  
Merger2.INI
```

The used INI file has the settings:

```
[OPTIONS]  
AVOID_MULTIPLE_OBJECTS = 0  
MERGE_GROUP_CONTENTS = 0  
NUMBER_OF_SLAVES = 2  
PREFIX_FOR_MASTER =  
SUFFIX_FOR_MASTER = _M  
PREFIX_SUFFIX_USAGE = 2
```

```
[SLAVE_1]  
FILENAME=Slave1.a21  
PREFIX=  
SUFFIX=_S1
```

```
[SLAVE_2]  
FILENAME=Slave2.a21  
PREFIX=  
SUFFIX=_S2
```

Now, because of the suffix and prefix option `DISABLE_SUFFIXES = 0`, for all objects except measurement 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 log file.

6 ASAP2 Comparer

In this chapter, you will find the following information:

6.1	Functionality	page 104
6.2	Command Line Parameters	page 104
6.3	Thesaurus Files	page 105
6.4	Exit Code	page 106
6.5	Initialization File	page 106
	[OPTIONS]	
	[FILTER]	
	[SYNTAX_TOLERANCE]	

6.1 Functionality

Comparison of two A2L files

The **ASAP2 Comparer** reads in two A2L files, compares their contents and creates a report in a configurable result file. The formatting of the A2L 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`.

Frequently used abbreviations

When comparing objects, in the result file frequently used abbreviations are LHS and RHS:

- > LHS is **left-hand sided** and refers to the file specified in the command line as `-A`.
- > RHS is **right-hand sided** and refers to the file specified in the command line as `-B`.

6.2 Command Line Parameters

Program call

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

<code>-A <name></code>	Name and path of the first A2L input file.
<code>-B <name></code>	Name and path of the second A2L input file.
<code>-D <name></code>	Name and path for the result file which will contain all compared attributes. If this parameter is not specified, the console will be used.
<code>-I <name></code>	Name and path for the initialization file if this file is not located in working directory or if its name is not <code>COMPARER.INI</code> .
<code>-L <name></code>	Name and path of the log file for warnings and error messages.
<code>-T <name></code>	Name and path of the thesaurus file. You can specify more than one thesaurus file.

6.3 Thesaurus Files

Functionality A thesaurus file contains a list of synonyms for object names. If while comparing a measurement or characteristic object is not found in the comparison file, it is checked by the thesaurus, whether there is a name replacement. If applicable, the comparison is performed with the renamed object.

Syntax Thesaurus files are ASCII files with the default file extensions THS or TXT. Other extensions are also accepted.

Evaluating the entries is case-sensitive.

Comment lines start with #. They are not evaluated.

Line types There are two types of lines:

- > Alias lists
- > Wildcard patterns

Both can occur together in one thesaurus file.

Alias lists All lines describing alias lists, begin with <LIST;>. They contain a list of variable names that are each enclosed within quotation marks and separated by a comma, semicolon, blank space, or tab.



Example: Alias List

```
LIST; "name1"; "name2"; "name3"
```



Note: One name must not occur in multiple lines - the second name is ignored with an error message, if applicable.

The names within a line are all synonyms for one another and must not contain any wildcards.

If a variable name is contained in this type of line, the thesaurus delivers all other names of this line as possible alternative names - irrespective of the position in the line where the searched for name is located. If there are multiple valid variable names, the first valid name in the line is used.

Wildcard patterns All lines describing wildcard patterns, begin with <PATTERN;> followed by a source pattern and a target pattern - each enclosed within quotation marks and separated by a comma, semicolon, blank space, or tab.

The patterns may contain one * or one ? each.



Example: Wildcard Pattern

```
PATTERN; "ABC*"; "PWM*
```

In contrast to the alias lists, the names are replaced here in one direction only. In the example, a replacement for the variable name PWM_Sample would not be found. However, for ABC_Sample the thesaurus would return PWM_Sample as a possible alternative name.



Note: Due to the thesaurus definition, the wildcard pattern described above will search e.g., for variable `ABC_Test` the corresponding variable `PWM_Test` in the other A2L. It is important that the search string is identical to the string of the original variable (in both cases `_test`). Thus, for example, for the variable `ABC_Test` no connection is found to a variable `PWM_Testing` (`_test` does not match `_TESTING`).

6.4 Exit Code

Determining success The success of the compare process 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
3	No license available

6.5 Initialization File

Directory 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.

General For general information on the structure and usage of the initialization file, see section [Configuration of the Command Line Tools](#) on page 8.

6.5.1 [OPTIONS]

Common settings This section contains common settings for the comparison:

Parameter	Default Value	Description
<code>ACCEPT_LONG_STRINGS</code>	0	See section Cross-Tool General Options .
<code>BITMASK_VS_DATATYPE</code>	1	This option configures the comparison of bitmasks for measurement and calibration objects. If set, missing bitmasks, empty bitmasks (0) and full bitmasks (e.g. 0xFF for data type <code>UBYTE</code>) are considered as equal. For example, no difference is reported with this option if a measurement object of data type <code>SWORD</code> has no bitmask in the first A2L file and bitmask 0xFFFF in the second A2L file.
<code>CREATE_THESAURUS_FOR_EQUAL_DISPLAY_NAMES</code>	""	Name of a thesaurus file to be created. If a file name is specified in this parameter, a new thesaurus is created by comparing the 2 A2L files and is saved in this file. For objects that have the same display identifier, a thesaurus entry is created at the same time. The generated thesaurus is then used directly

Parameter	Default Value	Description
		in the actual file comparison.
CSV_SEPARATOR	","	Only relevant for the output format CSV: Separator to be used to create result files in the CSV format.
DEEP_COMPARE_FOR_REFERENCES	0	With this option the comparison status of referenced objects is considered. Sample: A measurement object references a conversion rule. If the name of the conversion rule in both files is the same, but the conversion rule e.g. in one file having a different unit, the measurement object is considered as equal without the option. If the option is activated, the measurement value is considered as unequal.
DEFAULT_ENCODING_FOR_READ	""	See section Cross-Tool General Options .
DIFF_FILE_FORMAT	1	Format of the result file which contains all found differences. This option is operative only if a distinct result file has been specified. Possible values are: <ul style="list-style-type: none"> > 1 – Log file: One line is written to the result file for each found difference. > 2 - CSV: The differences are written as a semicolon-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. > 4 – XLSX/XLS The differences are written in the XLSX/XLS format. The result file can be opened with a spreadsheet application. Notice: For reasons of performance an XLSX file is to be preferred in any case, because it leads to a significant better run-time (shorter conversion time).
DIFF_FILE_GROUPING	1	This option specifies the grouping of entries in the result file. Possible values are: <ul style="list-style-type: none"> > 0 = no grouping > 1 = group by object type of the differences
DIFF_FILE_XML_STYLESHEET	""	Only relevant for the output format XML: In case of XML format of the result file, this parameter can be used to specify a path to an XSL style sheet. This is a textual option. Notice: The ASAP2 Comparer demo contains the style sheet <code>COMPARISON.XSL</code> which can be used to format the XML file as HTML page. Just open the generated XML file with a web browser.

Parameter	Default Value	Description
EXCEL_AUTOFIT	1	Only relevant for the output format XLSX/XLS: If this option is activated, the column widths are adjusted automatically. I.e. all columns are displayed as wide as the longest cell value is still completely visible.
EXCEL_AVOID_EMPTY_SHEETS	0	Only relevant for the output format XLSX/XLS: For each of the different ASAP2 comparison types (e.g., MODULE, CHARACTERISTIC, MEASUREMENT) 3 Excel pages are created with the following names: <ul style="list-style-type: none"> > <ASAP2 comparison type>-missing LHS = Object is missing in the first A2L file, e.g., MODULE-missing LHS > <ASAP2 comparison type>-missing RHS = Object is missing in the second A2L file > <ASAP2 comparison type>-compared = Comparison of the objects as object exists in both A2L files If this option is activated, an Excel page is only created when there are also objects of this ASAP2 comparison type.
EXCEL_BACKGROUND_COLOR_SEPARATOR	0xC0C0C0 (grey)	Only relevant for the output format XLSX/XLS: This option specifies the background color (as an RGB value) of the separating row that is displayed after the comparison of LHS and RHS values between the individual object comparisons.
EXCEL_BACKGROUND_COLOR_UNEQUAL	0x80FFFF (yellow)	Only relevant for the output format XLSX/XLS: This option specifies the background color (as an RGB value) that indicates the differing comparison value (on the RHS side) compared to the value on the LHS side.
EXCEL_MAX_COLUMN_WIDTH	80	Only relevant for the output format XLSX/XLS: This option specifies the maximum displayed width (in characters) of an Excel column. If an Excel column must be reduced due to this option, the word wrap is set automatically.
HIDE_PROGRESS_VIEW	0	See section Cross-Tool General Options .
IGNORE_ADDRESS_CHANGE	0	If this option is activated, measurement and calibration objects are considered as equal, if only the address has changed.
IGNORE_ADDRESS_FOR_VIRTUAL	0	If this option is activated, the optionally available addresses of virtual measurement objects are ignored.
IGNORE_DIMENSION_VALUE_1	1	If this option is active, dimension values 1 are ignored at the end of the dimension list for the keyword <code>MATRIX_DIM</code> when comparing. Example: Equal are: <code>MATRIX_DIM 7 1 1</code> ,

Parameter	Default Value	Description
		ARRAY_SIZE 7 and MATRIX_DIM 7 Not equal are in contrast: MATRIX_DIM 1 1 7 1 and ARRAY_SIZE 7
IGNORE_MODULE_NAME	0	If this option is activated, the first MODULE from the LHS database is compared with the first MODULE of the RHS database – regardless of whether both having the same name.
LIST_EQUAL_ATTRIBUTES	0	Possible values are: <ul style="list-style-type: none"> > 0 = In the result file, only the differences will be output – i.e. from objects that are not equal, only the unequal attributes. > 1 = In the result file all different objects are output but here with all its attributes - including the attributes that are equal. The comparison status is output with each attribute. > 2 = All objects are listed in the result file - with all its attributes. In this case also objects that are completely equal are output as well. The comparison status is output with each attribute. This option is suitable for creating a complete report.
MAX_INI_STRING_LENGTH	2048	See section Cross-Tool General Options.
REMOVE_DUPLICATE_IF_DATA_CANAPE_EXT	0	See section Cross-Tool General Options.
REMOVE_TRAILING_DIMENSION_VALUES_1	0	See section Cross-Tool General Options.
SIGNIFICANT_DIGITS_FOR_DOUBLE	12	See section Cross-Tool General Options.
SPLIT_CONVERSION_COEFFICIENTS	0	<ul style="list-style-type: none"> > 0 = Only one common difference value is created for the coefficients of linear and rational conversion rules. > 1 = For the coefficients of linear and rational conversion rules, the difference values are created for each coefficient.
USE_ADDRESS_EXTENSION_DEFAULT	0	If this option is activated, the default value 0 is used for comparison when no address extension is specified.
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 measurement 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 activated, the ASAP2 Comparer uses the default limits of a calibration object for comparison when EXTENDED_LIMITS is missing.

Parameter	Default Value	Description
USE_FORMAT_DEFAULT	1	With this option the format settings of the corresponding conversion method are used for comparison if no <code>FORMAT</code> is specified for a measurement or calibration object.
USE_UNIT_DEFAULT	1	If no <code>PHYS_UNIT</code> is specified for a measurement or calibration object, the unit of the referenced conversion method is used for comparison.
WARNING_FOR_MISSING_ADDRESS_INFORMATION	0	See section Cross-Tool General Options .

6.5.2 [FILTER]

Reduction

This section contains settings to reduce the data to compare:

Parameter	Default Value	Description
COMPARE_GLOBAL_DATA	1	If the value is 0, the header information and module-global data from <code>MOD_PAR</code> and <code>MOD_COMMON</code> are not compared.
COMPARE_IF_DATA	1	With this option active the <code>IF_DATA</code> sections will be compared, too. Otherwise they are ignored for comparison.
COMPARE_IF_DATA_CANAPE_EXT	0	If this option is set, the additional information from <code>IF_DATA CANAPE_EXT</code> is compared for measurement and calibration objects. Specifically, this is the standard value range for the display scaling and the display color.
COMPARE_IF_DATA_XCP	0	If this option is set, the event information from <code>IF_DATA XCP</code> is compared in detail for measurement objects.
USE_REGULAR_EXPRESSIONS	1	In the <code>FILTER</code> section, you can define <code>Name filter</code> 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 <code>USE_REGULAR_EXPRESSIONS</code> is active, the name filters are interpreted as regular expressions – otherwise as wildcard expressions.

Name filter

The default value of all the following name filters depends on the setting `USE_REGULAR_EXPRESSIONS`:

- > If the option `USE_REGULAR_EXPRESSIONS` is activated (= 1), the name filters are interpreted as regular expressions. The default value of the name filter options will then be `".*"`.
- > If the option `USE_REGULAR_EXPRESSIONS` is not activated (= 0), the name filters are interpreted as wildcard expressions. The default value of the name filter options will then be `"*"`.

Parameter	Default Value	Description
AXIS_PTS_TO_COMPARE		Here you can define a name filter for axis objects. Only axis objects whose name matches this filter

Parameter	Default Value	Description
		are compared. If this parameter is missing in the INI file, all axes are compared. If you want to ignore all axis objects for comparison, this filter has to be set empty.
BLOB_TO_COMPARE		Here you can define a name filter for BLOBs. Only BLOBs whose name matches this filter are compared. If this parameter is missing in the INI file, all BLOBs are compared. If you want to ignore all BLOBs for comparison, this filter has to be set empty.
CHARACTERISTIC_TO_COMPARE		Here you can define a name filter for characteristic objects. Only characteristic objects whose name matches this filter are compared. If this parameter is missing in the INI file, all characteristic objects are compared. If you want to ignore all characteristic objects for comparison, this filter has to be set empty.
CONVERSION_METHOD_TO_COMPARE		Here you can define a name filter for conversion methods. Only conversions whose name matches this filter are compared. If this parameter is missing in the INI file, all conversion methods are compared. If you want to ignore all conversion methods for comparison, this filter has to be set empty.
CONVERSION_TABLE_TO_COMPARE		Here you can define a name filter for conversion tables. Only tables whose name matches this filter are compared. If this parameter is missing in the INI file, all conversion tables are compared. If you want to ignore all conversion tables for comparison, this filter has to be set empty.
CRITERION_TO_COMPARE		Here you can define a name filter for variant criteria. Only variants whose name matches this filter are compared. If this parameter is missing in the INI file, all variant criteria are compared. If you want to ignore all variant criteria for comparison, this filter has to be set empty.
FRAME_TO_COMPARE		Here you can define a name filter for frames. Only frames whose name matches this filter are compared. If this parameter is missing in the INI file, all frames are compared. If you want to ignore all frames for comparison, this filter has to be set empty.
FUNCTION_TO_COMPARE		Here you can define a name filter for functions. Only functions whose name matches this filter are compared. If this parameter is missing in the INI file, all

Parameter	Default Value	Description
		functions are compared. If you want to ignore all functions for comparison, this filter has to be set empty.
GROUP_TO_COMPARE		Here you can define a name filter for groups. Only groups whose name matches this filter are compared. If this parameter is missing in the INI file, all groups are compared. If you want to ignore all groups for comparison, this filter has to be set empty.
MEASUREMENT_TO_COMPARE		Here you can define a name filter for measurement objects. Only measurement objects whose name matches this filter are compared. If this parameter is missing in the INI file, all measurement objects are compared. If you want to ignore all measurement objects for comparison, this filter has to be set empty.
RECORD_LAYOUT_TO_COMPARE		Here you can define a name filter for record layouts. Only layouts whose name matches this filter are compared. If this parameter is missing in the INI file, all record layouts are compared. If you want to ignore all record layouts for comparison, this filter has to be set empty.
TRANSFORMER_TO_COMPARE		Here you can define a name filter for transformers. Only transformers whose name matches this filter are compared. If this parameter is missing in the INI file, all transformers are compared. If you want to ignore all transformers for comparison, this filter has to be set empty.
TYPEDEF_TO_COMPARE		Here you can define a name filter for type definitions. Only type definitions whose name matches this filter are compared. If this parameter is missing in the INI file, all type definitions are compared. If you want to ignore all type definitions for comparison, this filter has to be set empty.
UNIT_TO_COMPARE		Here you can define a name filter for units. Only units whose name matches this filter are compared. If this parameter is missing in the INI file, all units are compared. If you want to ignore all units for comparison, this filter has to be set empty.

6.5.3 [SYNTAX_TOLERANCE]

Tolerance settings

This section contains tolerance settings for the ASAP2 reader. Without additional configuration, the A2L file will be parsed strictly according to the ASAP2 standard. However, it is possible to tolerate special errors if configured

so in this section.

Parameter	Default Value	Description
DIVERSE_BLOCK_BRACKETS	0	See section Cross-Tool Tolerance Settings .
IDENTS_START_WITH_DIGIT	0	See section Cross-Tool Tolerance Settings .
IGNORE_AML	0	See section Cross-Tool Tolerance Settings .
IGNORE_UNKNOWN_IF_DATA	0	See section Cross-Tool Tolerance Settings .
KEYWORD_AS_SYMBOL	0	See section Cross-Tool Tolerance Settings .
KEYWORDS_ALL_VERSIONS	0	See section Cross-Tool Tolerance Settings .
NESTED_COMMENTS	0	See section Cross-Tool Tolerance Settings .
PROJECT_NO_WITH_VALUE	0	See section Cross-Tool Tolerance Settings .
PURE_OBJECT_LIST	0	See section Cross-Tool Tolerance Settings .
RESERVED_WITH_DATATYPE	0	See section Cross-Tool Tolerance Settings .
SPECIAL_FORMAT_SYNTAX	0	See section Cross-Tool Tolerance Settings .
SPECIAL_IDENT_INNER	""	See section Cross-Tool Tolerance Settings .
SPECIAL_IDENT_START	""	See section Cross-Tool Tolerance Settings .
SYMBOL_AS_STRING	0	See section Cross-Tool Tolerance Settings .

7 ASAP2 Checker

In this chapter, you will find the following information:

7.1	Functionality	page 114
7.2	Command Line Parameters	page 114
7.3	Exit Code	page 115
7.4	Initialization File [OPTIONS] [SYNTAX_TOLERANCE] [EXTENDED_CHECK] [AUTO_CORRECTION]	page 115
7.5	Error Numbers	page 122

7.1 Functionality

Verifying of an A2L file	The ASAP2 Checker reads in an A2L file, performs a syntax check and optionally a configurable set of extended checks for consistency, semantic and plausibility of the A2L file. For each error or inconsistency found, an error message resp. warning message is created in a log file.
Automatic correction possibility	Optionally, some of the detected errors can be automatically corrected and a correspondingly corrected A2L file can be created.
Syntax check	The syntax check is done for the ASAP2 version specified in the A2L input file. If the A2L file does not contain the optional version info, the ASAP2 version to be checked can be specified in INI file, too.
Settings	Per default, all possible checks are done strictly. However, the semantic checks can be reduced and special error messages can be suppressed by settings in initialization file. Furthermore, a set of tolerance settings for syntax check is available in INI file.

7.2 Command Line Parameters

Program call When calling the program, the following command line parameters can be used:

-A <name>	Name and path of A2L file to be checked.
-L <name>	Name and path of the log file for error and warning messages.
-I <name>	Name and path of the initialization file if this file is not located in working directory or if its name is not <code>CHECKER.INI</code> .
-O <file name>	Name and path of the automatically corrected file after error detection.

-R <file name>	Name and path of the additionally generated HTML report file for error messages and check overview.
-S <file name>	Name and path of the stylesheet file for formatting the HTML report file. If no stylesheet file is specified, an embedded standard formatting is used. An example stylesheet file can be found in the Demos.

7.3 Exit Code

Determining success The success of the syntax and semantic check 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
3	No license available

7.4 Initialization File

Directory The initialization file `CHECKER.INI` is expected in current working directory. Alternatively, both a different file name and a different directory can be set in command line.

General For general information on the structure and usage of the initialization file, see section [Configuration of the Command Line Tools](#) on page 8.

7.4.1 [OPTIONS]

Common settings This section contains common settings for the checker:

Parameter	Default Value	Description
<code>ACCEPT_LONG_STRINGS</code>	0	See section Cross-Tool General Options .
<code>ADDITIONAL_SUCCESS_MESSAGE</code>	0	If this option is activated, an additional success message is output in the log file after all configured checks have been successfully executed.
<code>ASAP2_VERSION</code>	171	See section Cross-Tool General Options .
<code>CREATE_INVALID_CALIBRATION_HANDLES</code>	0	See section Cross-Tool General Options .
<code>DEFAULT_ENCODING_FOR_READ</code>	""	See section Cross-Tool General Options .
<code>HIDE_ERROR_MESSAGES</code>	0	Here you can specify a mask to suppress special error and warning messages. Each potential error message is identified by a single bit (see section Error Numbers on page 122). If you do not want to see a special message in your log file, you have to set the

Parameter	Default Value	Description
		<p>corresponding bit in this mask.</p> <p>Sample: If you set the value 0x19 for this mask, the error messages with numbers 0x10, 0x80 and 0x01 will not be written to log file. Nevertheless, the checks leading to these error messages will be executed, because they can possibly create other error messages. To completely deactivate a special check, you can use the options from section [EXTENDED_CHECK].</p> <p>The mask must be specified in hexadecimal format using the 0x... notation.</p>
HIDE_PROGRESS_VIEW	0	See section Cross-Tool General Options.
MESSAGE_SORTING	0	<ul style="list-style-type: none"> > 0 = alphabetical sorting > 1 = Messages in the log file are sorted by error numbers.
MAX_INI_STRING_LENGTH	2048	See section Cross-Tool General Options.
MINIMIZE_RESULT_FILE	0	See section Cross-Tool General Options.
PRESERVE_GLOBAL_COMMENTS	1	See section Cross-Tool General Options.
REMOVE_DUPLICATE_IF_DATA_CANAPE_EXT	0	See section Cross-Tool General Options.
REMOVE_ROOT	0	See section Cross-Tool General Options.
REMOVE_TRAILING_DIMENSION_VALUES_1	0	See section Cross-Tool General Options.
SIGNIFICANT_DIGITS_FOR_DOUBLE	12	See section Cross-Tool General Options.
SUPPRESS_VERSION_WARNINGS	0	See section Cross-Tool General Options.
WARNING_FOR_MISSING_ADDRESS_INFORMATION	0	See section Cross-Tool General Options.
WRITE_UTF8	1	See section Cross-Tool General Options.

7.4.2 [SYNTAX_TOLERANCE]

Tolerance settings

This section contains tolerance settings for the syntax check. Without additional configuration, the A2L file will be checked strictly according to the ASAP2 standard. However, it is possible to tolerate special errors if configured so in this section.

Parameter	Default Value	Description
DIVERSE_BLOCK_BRACKETS	0	See section Cross-Tool Tolerance Settings.
IDENTS_START_WITH_DIGIT	0	See section Cross-Tool Tolerance Settings.
IGNORE_AML	0	See section Cross-Tool Tolerance Settings.
IGNORE_UNKNOWN_IF_DATA	0	See section Cross-Tool Tolerance Settings.
KEYWORD_AS_SYMBOL	0	See section Cross-Tool Tolerance Settings.
KEYWORDS_ALL_VERSIONS	0	See section Cross-Tool Tolerance Settings.
NESTED_COMMENTS	0	See section Cross-Tool Tolerance Settings.
PROJECT_NO_WITH_VALUE	0	See section Cross-Tool Tolerance Settings.
PURE_OBJECT_LIST	0	See section Cross-Tool Tolerance Settings.

Parameter	Default Value	Description
RESERVED_WITH_DATATYPE	0	See section Cross-Tool Tolerance Settings .
SPECIAL_FORMAT_SYNTAX	0	See section Cross-Tool Tolerance Settings .
SPECIAL_IDENT_INNER	""	See section Cross-Tool Tolerance Settings .
SPECIAL_IDENT_START	""	See section Cross-Tool Tolerance Settings .
SYMBOL_AS_STRING	0	See section Cross-Tool Tolerance Settings .

7.4.3 [EXTENDED_CHECK]

Semantic and plausibility checks

This section contains the configuration of semantic and plausibility checks to be performed additionally to the syntax check:

Parameter	Default Value	Description
CANAPE_COMPATIBILITY	1	Some syntactically and semantically correct ASAP2 constructs are not yet supported by CANape in full functional range. You can check the A2L file for usage in CANape by activating this option. If set, the following additional tests will be performed: <ul style="list-style-type: none"> > Virtual measurement objects are supported only if they are scalar values – i.e. if they have no <code>MATRIX_DIM</code> resp. <code>ARRAY_SIZE</code> information. > Conversion tables must have at least one value pair.
CHECK_ALIGNMENT	1	The start address of all measurement and calibration objects must go with the alignment specified in A2L file.
CHECK_ASCII_STRING_LENGTH	0	If this option is active, it is checked whether ASCII strings have a length greater than 0.
CHECK_BITMASKS	1	This option activates the bit mask check for all measurement and calibration objects: <ul style="list-style-type: none"> > The bit mask must not be larger than predetermined by the data type of the object. Sample: For data type <code>UBYTE</code> the maximum allowed bit mask is <code>0xFF</code>. > The bit mask must be continuous – it must not have gaps (<code>0x9</code> is not allowed, for example).
CHECK_DUPLICATE_DISPLAY_IDENTIFIERS	0	If this option is active, all variables are checked for unique display names.
CHECK_FORMAT_STRINGS	1	This option activates the syntactical check of all format strings for both measurement and calibration objects and for conversion methods.
CHECK_IDENTIFIERS_STRICT	1	You can activate an extended syntax check of all identifiers in A2L file using this option. The basic syntax test only checks for valid characters: Identifiers are allowed to contain only letters, digits and the special characters <code>"_"</code> , <code>"."</code> ,

Parameter	Default Value	Description
		<p>"[" and "]" and they must start with a letter or with "_".</p> <p>The extended syntax check also verifies the structure of the name:</p> <ul style="list-style-type: none"> > A dot separates an identifier into partial identifiers. > The length of a partial identifier is limited to 128 characters. Especially, the total length of an identifier containing no dot at all is also limited to 128. > Each partial identifier must be a valid identifier itself – i.e. it must start with a letter or with "_". > Brackets ("[" and "]") are only allowed in pairs at the end of a partial identifier and they must contain either a number or another valid identifier.
CHECK_IF_DATA	1	This option activates the syntactical check of all IF_DATA blocks in A2L file.
CHECK_LIMITS	1	<p>Checks the value ranges of measurement and calibration objects. All referenced textual conversion rules (type FORM) are checked for correct syntax.</p> <ul style="list-style-type: none"> > The minimum value of a measurement or calibration object must not be larger than its maximum value. > The standard range of a calibration object must be inside its extended range (if specified with EXTENDED_LIMITS). > The physical range must be inside the converted data type range (using the specified conversion method). <p>The objects for which the check is performed can be set using the value of the option as follows:</p> <ul style="list-style-type: none"> > 0 = No check > 1 = For all variables > 2 = Only for calibration objects (CHARACTERISTIC, AXIS_PTS) > 3 = Only for variables with write access > 4 = for all variables except those that have a nonlinear conversion <p>Notice: As nonlinear conversions are considered:</p> <ul style="list-style-type: none"> > algebraic conversions > numeric conversions > rational conversion rules with square share
CHECK_OVERLAPPING	1	<p>If this parameter has a value other than 0, all non-virtual measurement and calibration objects are checked for address overlapping.</p> <p>If the value of this parameter is 2, objects without</p>

Parameter	Default Value	Description
		write access are ignored during the check (calibration objects with READ_ONLY resp. measurement objects without READ_WRITE).
CHECK_OVERLAPPING_ONLY_VALID_ADDRESSES	0	<p>If this option is active, only measurement and calibration objects are checked for address overlapping whose address does not equal 0.</p> <p>Precondition: The option <code>CHECK_OVERLAPPING</code> is activated.</p>
CHECK_RECORD_LAYOUTS	1	<p>This option can be used to activate the check of record layouts.</p> <p>First, a record layout has to be consistent for itself:</p> <ul style="list-style-type: none"> > The position numbers of its components must be unique and consecutive. > Some components of a record layout are mutually exclusive and must not be used together in same layout (e.g. <code>NO_AXIS_PTS_X</code>, <code>NO_RESCALE_X</code> und <code>FIX_NO_AXIS_PTS_X</code>). > For non-static layouts with dynamic number of axis points, the number of axis points must be before the axis points and table values. <p>Second, the usage of a record layout must be compatible with the object type of the referencing calibration object. ASAP2 Checker verifies e.g. the following conditions:</p> <ul style="list-style-type: none"> > The record layout for a scalar parameter must not contain any components for axis points. > The record layout for a common axis must not contain table values. > The record layout for an ASCII string must provide the data type <code>UBYTE</code> or <code>SBYTE</code>. > The record layout for a curve with standard axis must not contain fix axis components such as <code>OFFSET_X</code>, <code>SHIFT_OP_X</code> or <code>DIST_OP_X</code>.
CHECK_STATUS_TABLES	1	<p>Activate this option to enable the check of status text tables:</p> <ul style="list-style-type: none"> > Verbal conversion tables which are referenced as status text table by any conversion method must not have a default value. > The input range of a status text table must not overlap the raw value range of any measurement or calibration object using this status text table. <p>The objects for which the check is performed can be set using the value of the option as follows:</p> <ul style="list-style-type: none"> > 0 = No check

Parameter	Default Value	Description
		<ul style="list-style-type: none"> > 1 = For all variables > 2 = Only for calibration objects (CHARACTERISTIC, AXIS_PTS) > 3 = Only for variables with write access
CHECK_UNIQUE_ADDRESS_INFORMATION	0	<p>If set, this option checks whether conflicting address information are specified with a measurement or characteristic object (e.g., over several IF_DATA blocks).</p> <p>If this option is activated, the <code>CHECK_IF_DATA</code> option is automatically set.</p>
CHECK_UNIQUE_NAMES	1	<p>If set, this option activates the check for unique identifiers in all name spaces of the A2L file.</p>
CHECK_VARIANT_CODING	1	<p>If this option is active, the variant coding will be checked for consistency:</p> <ul style="list-style-type: none"> > Each measurement or calibration object defined as selector of a variant criterion, must have a verbal conversion table assigning the variant names to the selector values. > The usage of <code>VAR_FORBIDDEN_COMB</code> in variant coding is not supported. > Variant coded calibration objects must dependent on exactly one variant criterion. > Each variant coded calibration object must be assigned to a group which contains only objects dependent on the same variant criterion. > All variant coded calibration objects which depend on the same variant criterion must have equal address offsets for their variants.
CIRCULAR_DEPENDENCIES	0	<p>If this option is active, the following cyclic dependencies are checked:</p> <ul style="list-style-type: none"> > Input values of virtual measurement > Input values of virtual/dependent parameter > Subgroups/subfunctions > Derived units > TRANSFORMER dependencies > Structure component
ONLY_SYNTAX_CHECK	0	<p>If you activate this option, ASAP2 Checker terminates the investigation of A2L file immediately after the syntax check. All other options of this section will be ignored.</p> <p>In particular, those checks which cannot be disabled explicitly will not be performed if this option is set. (e.g. check for resolvable references and evaluation of data types for calibration objects)</p>

7.4.4 [AUTO_CORRECTION]

Correction settings This section contains settings for automatic correction of errors.



Note: In order for the automatic correction to be performed, the associated check must by itself also be activated.

For example, the option `LIMITS` of the auto correction is ineffective if `CHECK_LIMITS` in section `[EXTENDED_CHECK]` is not activated.

Parameter	Default Value	Description
<code>BITMASKS</code>	0	<p>If this option is set, errors associated with bit masks are automatically corrected:</p> <ul style="list-style-type: none"> > Bit masks that are too large for the data type are correspondingly shortened. Example: A bit mask <code>0xFFFF</code> of data type <code>Byte</code> is shortened to <code>0xFF</code>. > Bits masks of ASCII strings that do not equal <code>0xFF</code> are removed. > Gaps in bit masks are filled. Example: A bit mask <code>0x14</code> is converted to <code>0x1C</code>.
<code>DUPLICATE_DISPLAY_IDENTIFIERS</code>	0	<p>If this option is set, duplicate display names are corrected by creating unique names through appending <code>_Copyx</code>.</p>
<code>FORMAT_STRINGS</code>	0	<p>If this option is set, the following errors in the format strings are automatically corrected:</p> <ul style="list-style-type: none"> > Empty format strings for conversion rules are replaced by <code>%.9</code>. > Additional characters are removed or the percent sign is added if needed. <p>Examples: <code>%.2f</code> → <code>%.2</code> <code>6.2</code> → <code>%.2</code> <code>_____.2</code> → <code>%.2</code></p>
<code>LIMITS</code>	0	<p>If this option is set, errors associated with physical limits are automatically corrected:</p> <ul style="list-style-type: none"> > Interchanged minimum-maximum values: If the minimum value is greater than the maximum value, both are interchanged. > If the expanded limits of a calibration value are not within the (physically converted) data type limits, they are limited correspondingly. > If the standard limits of a calibration value are not within the expanded limits, they are correspondingly limited.
<code>OVERLAPPING</code>	0	<p>If this option is set, variables that cause address overlapping are removed from the database. Depending on the value of option <code>CHECK_OVERLAPPING</code> in section <code>[EXTENDED_CHECK]</code>, only variables with write access are considered here.</p> <p>If variables are removed due to address overlapping, the variable that is first in the alphabetical order is always kept.</p>
<code>UNIQUE_NAMES</code>	0	<ul style="list-style-type: none"> > 1 = For duplicate identifiers in the same namespace, unique names are generated. > 2 = Objects with duplicate identifiers in the same namespace are removed from the database.

7.5 Error Numbers

Suppressing warning and error messages You can suppress some individual warning and error messages of **ASAP2 Checker** by specifying an error mask in INI file: You have to set the bit numbers of the messages you want to suppress in the `HIDE_ERROR_MESSAGES` option (see section [OPTIONS] on page 115).

The following error messages can be disabled that way:

Error Number	Message/Meaning
0x00000001	Syntax error in a textual conversion method
0x00000002	Invalid value for an ASAP2 parameter: <ul style="list-style-type: none"> > The (maximum) number of axis points/rescale pairs must be ≥ 1. > The factor of a linear conversion method must not be 0. > The distance value in <code>FIX_AXIS_PAR_DIST</code> must not be 0. > If the dimension of an ASCII string is specified via <code>MATRIX_DIM</code>, only the X dimension may have a value other than 1.
0x00000004	The usage of an ASAP2 keyword is not allowed in context: <ul style="list-style-type: none"> > <code>AXIS_PTS_REF</code> is allowed only for axes of type <code>COM_AXIS</code> or <code>RES_AXIS</code>. > <code>CURVE_AXIS_REF</code> is allowed only for axes of type <code>CURVE_AXIS</code>. > <code>DEPOSIT</code> is not allowed for axes of type <code>FIX_AXIS</code>. > <code>FIX_AXIS_PAR</code>, <code>FIX_AXIS_PAR_DIST</code> und <code>FIX_AXIS_PAR_LIST</code> are allowed only for axes of type <code>FIX_AXIS</code>. Only one of these keywords may be used for one axis. > <code>MAP_LIST</code> is allowed only for characteristic objects of type <code>CUBOID</code>. > <code>COMPARISON_QUANTITY_LIST</code> is allowed only for characteristic objects of type <code>CURVE</code>. > <code>NUMBER</code> and <code>MATRIX_DIM</code> are allowed only for characteristic objects of type <code>VAL_BLK</code> or <code>ASCII</code>. > <code>COEFFS</code> is allowed only for conversion methods of type <code>RAT_FUNC</code>. > <code>COEFFS_LINEAR</code> is allowed only for conversion methods of type <code>LINEAR</code>. > <code>COMPU_TAB_REF</code> is allowed only for tabular conversion methods. > <code>FORMULA</code> is allowed only for textual conversion methods (type <code>FORM</code>). > <code>REF_UNIT</code> and <code>UNIT_CONVERSION</code> are allowed only for units of type <code>DERIVED</code>. > <code>SI_EXPONENTS</code> is allowed only for units of type <code>EXTENDED_SI</code>.
0x00000008	Invalid number of axes for a calibration object. The number of axes is specified corresponding to the object type as follows: <ul style="list-style-type: none"> > <code>VALUE</code>, <code>VAL_BLK</code>, <code>ASCII</code>: 0 > <code>CURVE</code>: 1 > <code>MAP</code>: 2 > <code>CUBOID</code>: 3 (if <code>MAP_LIST</code> is used: 1) > <code>CUBE_4</code>: 4 > <code>CUBE_5</code>: 5
0x00000010	A referenced object with the corresponding name does not exist in the A2L file. The following references by name are checked: <ul style="list-style-type: none"> > Input quantities of axes > Common axes and rescale axes > Curve axes > Record layouts of calibration objects

Error Number	Message/Meaning
	<ul style="list-style-type: none"> > Input parameters of virtual calculation methods > Measure objects in <code>COMPARISON_QUANTITY</code> > Characteristic maps in <code>MAP_LIST</code> > Units in <code>REF_UNIT</code> > Conversion tables in <code>COMPU_TAB_REF</code> > Measure and calibration objects in group and function assignment > Sub groups and sub functions in group and function assignment > Measure objects in frames > Groups in user rights setting > Memory segments in <code>REF_MEMORY_SEGMENT</code> > Conversion methods for measurement and calibration objects > Selector object for variant criterions
0x00000020	The record layout of a calibration object does not allow determining the data type due to missing information.
0x00000040	<p>An optional ASAP2 keyword is missing in context. The following optional keywords are checked:</p> <ul style="list-style-type: none"> > Fix axes must have one of the keywords <code>FIX_AXIS_PAR</code>, <code>FIX_AXIS_PAR_DIST</code> or <code>FIX_AXIS_PAR_LIST</code> if the corresponding parameters are not specified in assigned record layout. > Axes of type <code>COM_AXIS</code> or <code>RES_AXIS</code> must refer to the axis object via <code>AXIS_PTS_REF</code>. > Axes of type <code>CURVE_AXIS</code> must refer to the normalization curve via <code>CURVE_AXIS_REF</code>. > Characteristic objects of type <code>VA_BLK</code> (value blocks) and ASCII strings must have a dimension setting via <code>NUMBER</code> or <code>MATRIX_DIM</code>. > For linear conversion methods, the conversion parameters must be specified using keyword <code>COEFFS_LINEAR</code>. > For rational conversion methods (type <code>RAT_FUNC</code>) the coefficients must be set in keyword <code>COEFFS</code>. > Textual conversion methods (type <code>FORM</code>) must have keyword <code>FORMULA</code> to specify the formula text. > Tabular conversion methods must refer to the conversion table via keyword <code>COMPU_TAB_REF</code>. > Derived units must refer to a base unit using both the keywords <code>REF_UNIT</code> and <code>UNIT_CONVERSION</code>. > SI units must specify their basic exponents in keyword <code>SI_EXPONENTS</code>. > For non-virtual measurement objects the address must be specified in keyword <code>ECU_ADDRESS</code> or in one of the standardized <code>IF_DATA</code> blocks (e.g. <code>ASAP1B_ADDRESS</code> or <code>ASAP1B_CCP</code>). > Each variant criterion must specify its selector object using either the keyword <code>VAR_SELECTION_CHARACTERISTIC</code> or <code>VAR_MEASUREMENT</code>. > For each variant coded calibration object, the variant criterion must be specified.

Error Number	Message/Meaning
0x00000080	<p>A referenced object has the wrong object type:</p> <ul style="list-style-type: none"> > The characteristic object referenced via <code>CURVE_AXIS_REF</code> must have object type <code>CURVE</code>. > The input parameters for the calculation of virtual/dependent characteristic objects must have object type <code>VALUE</code>. > The characteristic objects referenced via <code>MAP_LIST</code> must have object type <code>MAP</code>. > The conversion tables referenced via <code>STATUS_STRING_REF</code> must be verbal tables. > The type of a conversion table referenced by <code>COMPU_TAB_REF</code> must match the type of the conversion method (numeric/verbal table).
0x00000100	The start address of a measurement or calibration object does not go with the specified alignment.
0x00000200	An identifier does not fulfill the strict ASAP2 syntax rules.
0x00000400	The format string of a conversion method or of a measurement or calibration object is syntactically wrong.
0x00000800	<p>Conflicting or ambiguous components in a record layout:</p> <ul style="list-style-type: none"> > The position numbers of the components are non-consecutive: They have gaps or they are not unique. > The number of axis points is both specified as fixed and as dynamic. > For non-static record layouts with dynamic number of axis points the number of axis points must be before axis points and table values. > For record layouts with alternating deposit of table values, the axis points and table values must have consecutive position numbers. > To describe fix axis parameters, the components <code>SHIFT_OP</code> and <code>DIST_OP</code> must not be used together.
0x00001000	<p>The usage of a record layout is incompatible with the object type of the referencing calibration object – the record layout does not go with the object. Sample:</p> <ul style="list-style-type: none"> > The record layout of a scalar parameter contains axis points. > The number of axis points fixed in the record layout of a curve is larger than the maximum number of axis points specified in the axis description of the curve.
0x00002000	<p>An identifier is not unique in its namespace. The following name spaces are checked:</p> <ul style="list-style-type: none"> > <code>MEASUREMENT</code>, <code>CHARACTERISTIC</code> and <code>AXIS_PTS</code> > <code>COMPU_METHOD</code> > <code>COMPU_TAB</code>, <code>COMPU_VTAB</code> and <code>COMPU_VTAB_RANGE</code> > <code>GROUP</code> > <code>FUNCTION</code> > <code>RECORD_LAYOUT</code> > <code>FRAME</code> > <code>UNIT</code> > <code>USER_RIGHTS</code> > <code>SYSTEM_CONSTANT</code> > <code>MEMORY_SEGMENT</code> > <code>VARIANT_CRITERION</code> > Variants of a variant criterion
0x00004000	An axis object must not be referenced both as common axis and as rescale axis.

Error Number	Message/Meaning
0x00008000	<p>Conflicting information for parameters which are describable in different ways:</p> <ul style="list-style-type: none"> > Dimension for value blocks and ASCII strings (via <code>MATRIX_DIM</code> or <code>NUMBER</code>) > Dimension for measurement arrays (via <code>MATRIX_DIM</code> or <code>ARRAY_SIZE</code>) > Address for measurement and calibration objects in different <code>IF_DATA</code> blocks > Number of axis points for fix axes > Selector object for a variant criterion
0x00010000	<p>Inconsistent conversion table, e.g.:</p> <ul style="list-style-type: none"> > An input value is multiple defined. > The number of value pairs does not match. > The conversion table has no value pairs at all. > Overlapping input ranges.
0x00020000	<p>Implausible value range for a measurement or calibration object, e.g.:</p> <ul style="list-style-type: none"> > Minimum value is larger than maximum value. > The standard range of a calibration object is (partly) outside the extended range (<code>EXTENDED_LIMITS</code>). > The value range is (partly) outside the converted data type range.
0x00040000	<p>The axis points of a fix axis are outside its specified physical range.</p>
0x00080000	<p>The usage of a verbal conversion table as status text table is not valid because:</p> <ul style="list-style-type: none"> > It has a default value. > There are overlapping value ranges with a referencing object.
0x00100000	<p>Measure and calibration objects with more than two dimensions (X and Y) are not supported yet by CANape.</p>
0x00200000	<p>The bit mask of a measurement or calibration object is not valid because:</p> <ul style="list-style-type: none"> > It is not continuous. > It is too big for the data type or not valid for the object type.
0x00400000	<p>The <code>IF_DATA</code> block of a measurement or calibration object cannot be interpreted to get the address information due to an unknown AML description.</p>
0x00800000	<p>Virtual or dependent objects are supported by CANape only if they are scalar values.</p>
0x01000000	<p>The selector of a variant criterion has an invalid conversion method (the conversion must be a verbal table assigning the variant names to the selector values).</p>
0x02000000	<p>The variant coding of the A2L file is not supported by CANape because some additional restrictions are not fulfilled. For example:</p> <ul style="list-style-type: none"> > Variant coded calibration objects must depend on exactly one variant criterion. > Variant coded calibration objects which depend on the same variant criterion must be assigned to the same group. > Variant coded calibration objects which depend on the same variant criterion must have equal offsets for their variants.

8 ASAP2 Modifier

In this chapter, you will find the following information:

8.1	Functionality	page 126
8.2	Command Line Parameters	page 126
8.3	Exit Code	page 127
8.4	Initialization File	page 127
	[OPTIONS]	
	[MODIFICATIONS]	
	[SYNTAX_TOLERANCE]	
	[FILTER]	
	[OPTIMIZATION]	
	[EXCEL_UPDATE]	
	[CREATE_MEASURE_ARRAYS]	
	[CREATE_STRUCTURES]	
	[CREATE_XCP_DAQ_EVENTS]	
8.5	Warning Level	page 152
8.6	CANape DBU Format	page 153
	Format description in BNF	
8.7	JSON File for XCP Event Description	page 154

8.1 Functionality

Modification of data The **ASAP2 Modifier** reads in an A2L file, performs modifications to it that are configured in the INI file, and stores the modified data in a new A2L file. Optionally, a label file can be created that contains the name of all measurement and calibration values of the result file.

8.2 Command Line Parameters

Program call The following command line parameters can be specified when calling the program.

-A <name>	Name and path of the A2L input file.
-B <name>	Name and path of the label file to be created.
-E <name>	Name and path of the event description file to be read. A file in JSON format is expected, which complies with the format description in section JSON File for XCP Event Description on page 154. In section [CREATE_XCP_DAQ_EVENTS] on page 152 the settings of the INI file are described.
-I <name>	Name and path of the initialization file if this file is not located in the working directory or if its name is not MODIFIER.INI .

-I <name>	Name and path of the log file for warnings and error messages.
-O <name>	Name and path of the A2L result file to be created.
-U <name>	Name and path of the CANape DBU Format file to be imported.
-X <name>	Name and path of the Excel file (as alternative CSV file) to be read. See section [EXCEL_UPDATE] on page 142 for settings of the INI file. The options are used to set which columns are to be imported for the update. Notice: The import is only possible if the Excel file to be imported is not opened in any other way, e.g. in Excel.

8.3 Exit Code

Determining success The success of the modification process 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
3	No license available

8.4 Initialization File

Directory The initialization file `MODIFIER.INI` is expected in the current directory by default. Alternatively, a different directory can also be specified via the command line.

General For general information on the structure and usage of the initialization file, see section `Configuration of the Command Line Tools` on page 8.

8.4.1 [OPTIONS]

General This section contains general settings:

Parameter	Default Value	Description
<code>ACCEPT_LONG_STRINGS</code>	0	See section <code>Cross-Tool General Options</code> .
<code>ASAP2_VERSION</code>	171	See section <code>Cross-Tool General Options</code> .
<code>CREATE_INVALID_CALIBRATION_HANDLES</code>	0	See section <code>Cross-Tool General Options</code> .
<code>DEFAULT_ENCODING_FOR_READ</code>	""	See section <code>Cross-Tool General Options</code> .
<code>DELETE_EMPTY_GROUPS</code>	0	Empty groups, functions, and frames (without references to measurement and calibration values and without non-empty subgroups) are removed from the result file. Groups that were not yet empty in the input file and became empty groups only as a result of active filtering are also removed in the process. Notice: This option is only useful,

Parameter	Default Value	Description
		if the <code>REMOVE_INVALID_REFERENCES</code> option is set simultaneously. For this reason, <code>REMOVE_INVALID_REFERENCES</code> is set automatically when this option is activated.
<code>GROUP_FILTER_RECURSIVE</code>	0	This option is only relevant if a group filter has been configured with the command line <code>-G</code> . If this option is then active, the group filter works recursively, hence also for all subgroups or subfunctions.
<code>HIDE_PROGRESS_VIEW</code>	0	See section <code>Cross-Tool General Options</code> .
<code>INCLUDE_SAVE_MODE</code>	0	<p>If this option is activated, the assignment of objects to included files is retained. The objects from include files are saved back to their original file, and corresponding include instructions are generated in the result file of the ASAP2 Modifier.</p> <p>Attention: All original include files are overwritten with this option!</p> <p>Notice: This option is can be combined with the filter mode <code>SPLIT</code> (<code>DIRECTION = 2</code>). However, the include file (<code>INCLUDE_NAME</code>) specified for this purpose must not yet belong to the original file.</p> <p>The assignment to an include file is retained for the following objects: <code>CHARACTERISTIC, MEASUREMENT, AXIS_PTS, FUNCTION, GROUP, COMPU_METHOD, COMPU_TAB, COMPU_VTAB, COMPU_VTAB_RANGE, INSTANCE, RECORD_LAYOUT, UNIT, FRAME, USER_RIGHTS, TYPEDEF_xx, TRANSFORMER, TYPEDEF_xx, TRANSFORMER, BLOB</code></p> <p>AML definitions, module-global data and global <code>IF_DATA</code> blocks are always transferred to the main file.</p> <ul style="list-style-type: none"> > 0 = Assignment to include files is NOT retained in the result file. > 1 = Assignment to include files is retained in the result file.
<code>MAX_INI_STRING_LENGTH</code>	2048	See section <code>Cross-Tool General Options</code> . Notice: The value set here can be relevant for other options, such as <code>SEARCH_NAME</code> .
<code>MINIMIZE_RESULT_FILE</code>	0	See section <code>Cross-Tool General Options</code> .
<code>PRESERVE_GLOBAL_COMMENTS</code>	1	See section <code>Cross-Tool General Options</code> .
<code>REMOVE_DUPLICATE_IF_DATA_CANAPE_EXT</code>	0	See section <code>Cross-Tool General Options</code> .
<code>REMOVE_INVALID_REFERENCES</code>	1	If the input file contains references to non-existing measurement and calibration values and to non-existing subgroups and sub-functions, these references are removed from groups, functions, and frames.

Parameter	Default Value	Description
		This option is independent of whether any filtering is active. > 0 = Invalid references are retained > 1 = Invalid references are removed
REMOVE_ROOT	0	See section Cross-Tool General Options.
REMOVE_TRAILING_DIMENSION_VALUES_1	0	See section Cross-Tool General Options.
SIGNIFICANT_DIGITS_FOR_DOUBLE	12	See section Cross-Tool General Options.
SUPPRESS_VERSION_WARNINGS	0	See section Cross-Tool General Options.
WARNING_FOR_MISSING_ADDRESS_INFORMATION	0	See section Cross-Tool General Options.
WARNING_LEVEL	2	Warning level (see section Warning Level on page 152). To the log file only warnings are issued, whose level is less or equal to the value set here.
WRITE_FRAGMENT	0	If this option is activated, only an A2L fragment is saved to the result file. This fragment can be added to other A2L files via an include. In particular, the PROJECT and MODULE keywords are not written to the result file with this option. > 0 = Not active > 1 = Active
WRITE_UTF8	1	See section Cross-Tool General Options.

8.4.2 [MODIFICATIONS]

Modification

In this section, you can configure the specific changes you want to make to objects of the A2L file.

Parameter	Default Value	Description
ADDRESS_OFFSET	0	Adds to the addresses of all measurement and calibration values that meet the filter conditions, the fixed offset stated here. The offset is also added to the variant addresses if needed. If no filter is defined or active, the modifications are applied to all variables.
CONVERT_GROUP_TO_FUNCTION	0	If this option is activated, all groups are converted to functions for the result file. In so doing, the referenced measurement values are displayed as local measurement values of the respective function. > 0 = No conversion > 1 = Conversion to measurement values
CONVERT_PARAMETER_TO_MEASURE	0	If this option is activated, all calibration objects of type VALUE or VAL_BLK (scalar values or value blocks) that meet the filter conditions, are converted to measurement values for the result file. > 0 = No conversion

Parameter	Default Value	Description
		<ul style="list-style-type: none"> > 1 = Conversion to measurement values <p>If no filter is defined or active, the modifications are applied to all variables.</p>
CREATE_DISPLAY_IDENTIFIERS	0	<p>Creates display names for measurement and calibration objects which meet the filter conditions:</p> <ul style="list-style-type: none"> > 0 = No display names are created. > 1 = Only display names for those measurement and calibration objects are created, which have no display name yet. Existing display names are not modified. <p>The display name is set to the object name. If no filter is defined or active, the modifications are applied to all variables.</p>
CREATE_SYMBOL_LINKS	0	<p>Creates symbol links for measurement and calibration objects which meet the filter conditions:</p> <ul style="list-style-type: none"> > 0 = No symbol links are created. > 1 = Only the symbol links for those measurement and calibration objects are created, which have not symbol links yet. Existing symbol links are not modified. > 2 = Symbol links are created for all measurement and calibration objects. Existing symbol links will be overwritten. For all symbol links, the offset 0 is created. > 3 = Symbol links are created for all measurement and calibration objects. Existing symbol links are kept, but prefix and postfix are added to them if they are specified. Existing offsets are kept. <p>The symbol name of the symbol link is formed from the object name of the source file and an optional configurable prefix or suffix (see <code>SYMBOL_LINK_POSTFIX</code> and <code>SYMBOL_LINK_PREFIX</code>).</p> <p>If no filter is defined or active, the modifications are applied to all variables.</p>
DEFAULT_BYTE_ORDER	""	<p>Specifies the global default byte order that is to be set in all modules of the A2L file.</p> <p>The global byte order applies to all measurement and calibration values that have no local byte order specified.</p> <p>Local specifications of the byte order for measurement and calibration values remain unchanged.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> > INTEL > MOTOROLA
MAKE_MEASUREMENT_WRITABLE	0	<p>Adds in the result file a READ_WRITE flag to all measurement values which meet the filter conditions.</p>

Parameter	Default Value	Description
		<ul style="list-style-type: none"> > 0 = READ_WRITE flag is not added. > 1 = READ_WRITE flag is added. <p>If no filter is defined or active, the modifications are applied to all variables.</p>
NORMALIZE_ARRAY_INDEX_FORM	0	<p>Normalizes array indexes in symbol names. Symbol names are created in the normalized form <code>array[0][1][2]</code>. This option specifies which original form of array indexes shall be converted. Different values can be combined with each other via OR operation.</p> <ul style="list-style-type: none"> > 0x001 = array._0_.1_.2_ (e.g. <code>ELF_ARRAY_INDEX_FORM = 0</code> in Updater) > 0x002 = array._0_1_2_ (e.g. <code>ELF_ARRAY_INDEX_FORM = 2</code> in Updater) > 0x004 = array_0_1_2_ (e.g. <code>ELF_ARRAY_INDEX_FORM = 3</code> in Updater) > 0x008 = array_0_1_2 (e.g. <code>ELF_ARRAY_INDEX_FORM = 6</code> in Updater) > 0x010 = array.0.1.2 (e.g. <code>ELF_ARRAY_INDEX_FORM = 8</code> in Updater) <p>Attention: If the A2L file contains array indexes of both form 0x004 and 0x008, a unique detection and conversion is not possible. In this case, using the option is not recommended as it may lead to incorrect conversions.</p>
REMOVE_AML	0	<p>Removes the A2ML block from the result file.</p> <ul style="list-style-type: none"> > 0 = A2ML block is retained. > 1 = A2ML block is removed.
REMOVE_COMMENTS	0	<p>If this option is activated, comments are removed from the result file.</p> <ul style="list-style-type: none"> > 0 = All comments are retained. > 1 = Only the comments and annotations of the measurement and calibration values are removed. > 2 = All comments are removed from the A2L file.
REMOVE_FROM_GROUPS	""	<p>Specifies a list of groups/functions and frames from which measurement and calibration values of the result file are removed.</p> <p>Only the assignment is removed here in each case. The measurement and calibration values themselves are retained in the database.</p> <p>Wildcards can be used in the group name. Multiple group names can be specified separated by a semicolon.</p> <p>If a filter is active, only the variables of the filter result are removed from the groups involved. If no filter is defined or active, all objects from the specified list of groups/functions and frames are removed.</p>

Parameter	Default Value	Description
REMOVE_IF_DATA	0	If this switch is activated, the IF_DATA blocks are removed from the result file. <ul style="list-style-type: none"> > 0 = All IF_DATA blocks are retained. > 1 = IF_DATA blocks are removed from the header only. > 2 = All IF_DATA blocks are removed – thus, for example, also local information for measurement and calibration values.
REMOVE_IF_DATA_CANAPE_EXT	0	If this option is activated, the IF_DATA blocks CANAPE_EXT and CANAPE are removed from the A2L file for all variables. In addition, the corresponding AML is removed from the file.
REMOVE_LOCAL_ALIGNMENT_SETTINGS	0	If this option is activated, the local alignment settings for all record layouts are removed. Subsequent the alignment settings of the module globally apply.
REMOVE_MOD_COMMON	0	Removes the MOD_COMMON block from the result file. <ul style="list-style-type: none"> > 0 = MOD_COMMON block is retained. > 1 = MOD_COMMON block is removed.
REMOVE_MOD_PAR	0	Removes the MOD_PAR block from the result file. <ul style="list-style-type: none"> > 0 = MOD_PAR block is retained. > 1 = MOD_PAR block is removed.
REMOVE_SYMBOL_LINK	0	If this option is activated, all occurrences of the keyword SYMBOL_LINK are removed from the result file.
RENAME_VARIABLE_SOURCE RENAME_VARIABLE_TARGET	"" ""	If one pattern each is entered for both options, all measurement and calibration values whose name corresponds to the source pattern are renamed according to the target pattern. <p>Example: Source-Pattern ABC*, Target-Pattern XYZ*</p> <ul style="list-style-type: none"> > ABCtest is renamed to XYZtest > xABCtest is not renamed <p>The patterns may contain a * or a ? in each case. The Wildcard patterns from source pattern and target pattern must correspond to one another. If the new variable name already exists in the database, a unique name is generated by adding a suffix. For each renamed value, information is output to the log file.</p> <p>If a filter is active simultaneously, renaming occurs only for measurement and calibration values that meet the filter conditions.</p> <p>If no filter is defined or active, the renaming occurs for all objects that correspond to the source pattern.</p>
RENAME_VARIABLES_TO_SYMBOL_LINK	0	If this option is set, all measurement and calibration values that meet the filter conditions and that have a MAP symbol name are renamed. New object name is the MAP symbol name.

Parameter	Default Value	Description
		<p>If this already exists in the corresponding namespace, a unique name is generated by adding a suffix. For each unnamed value, information is written to the log file</p> <p>If no filter is defined or active, the renaming occurs for all objects with MAP symbol name, otherwise only for the variables of the filter result.</p>
REPLACE_INVALID_IDENT_CHARACTERS	0	<p>If this option is set, all identifiers in the A2L file are renamed such that invalid characters are replaced by an underscore.</p> <p>According to ASAP2 standard, only letters, digits, dots, square brackets, and underscores are permitted in identifiers.</p> <p>In order for invalid characters in identifiers to be accepted at all when reading the A2L file, the options <code>SPECIAL_IDENT_INNER</code> and <code>SPECIAL_IDENT_START</code> must be set correspondingly.</p> <p>A correction in the sense of the expanded, strict identifier check (<code>CHECK_IDENTIFIERS_STRICT</code>) does not occur with this option. Invalid characters are replaced with ' _ ' only.</p>
SET_ADDRESS_EXTENSION	""	<p>Sets the corresponding value as address extension for all measurement and calibration objects, that meet the filter conditions:</p> <ul style="list-style-type: none"> > Possible values = 0 – 65535 > Default value = Empty string (thus no address extension) <p>If no filter is defined or active, the modifications are applied to all variables.</p>
SET_CONVERSION_METHOD	""	<p>Sets the name specified here (if not empty) as the name of the conversion rule for all measurement and calibration objects, that meet the filter conditions.</p> <p>It is not checked whether a conversion rule with this name actually exists in the database.</p>
SET_FORMAT	""	<p>Sets the format string specified here (as long as it is not empty) for all measurement and characteristic objects that meet the filter conditions.</p> <p>The format string should have the following form, whereby the number of significant digits is optional: "%<number of significant digits>.<number of decimal places>"</p> <p>Examples: "%12.0" or "%.6"</p>
SET_PROJECT_VERSION	""	<p>Sets the string specified here (as long as it is not empty) as project version in the A2L file.</p> <ul style="list-style-type: none"> > Possible values = valid strings according to ASAP2 standard
SPLIT_MEASUREMENT_ARRAYS	0	<p>If this option is set, single elements are created from all measurement arrays that meet the filter conditions.</p>

Parameter	Default Value	Description
		This is needed, for example, if the individual elements of an array are to be measured with different measurement events.
SYMBOL_LINK_POSTFIX	""	This optional postfix is succeeded to the object name when creating symbol names of symbol links for a measurement and calibration object.
SYMBOL_LINK_PREFIX	""	This optional prefix is prepended to the object name when creating symbol names of symbol links for a measurement and calibration object.

8.4.3 [SYNTAX_TOLERANCE]

Tolerance settings

This section contains tolerance settings for the syntax check. Without additional configuration, the A2L file will be parsed strictly according to the ASAP2 standard. However, it is possible to tolerate special errors if configured so in this section.

Parameter	Default Value	Description
DIVERSE_BLOCK_BRACKETS	0	See section Cross-Tool Tolerance Settings .
IDENTS_START_WITH_DIGIT	0	See section Cross-Tool Tolerance Settings .
IGNORE_AML	0	See section Cross-Tool Tolerance Settings .
IGNORE_UNKNOWN_IF_DATA	0	See section Cross-Tool Tolerance Settings .
KEYWORD_AS_SYMBOL	0	See section Cross-Tool Tolerance Settings .
KEYWORDS_ALL_VERSIONS	0	See section Cross-Tool Tolerance Settings .
NESTED_COMMENTS	0	See section Cross-Tool Tolerance Settings .
PROJECT_NO_WITH_VALUE	0	See section Cross-Tool Tolerance Settings .
PURE_OBJECT_LIST	0	See section Cross-Tool Tolerance Settings .
RESERVED_WITH_DATATYPE	0	See section Cross-Tool Tolerance Settings .
SPECIAL_FORMAT_SYNTAX	0	See section Cross-Tool Tolerance Settings .
SPECIAL_IDENT_INNER	""	See section Cross-Tool Tolerance Settings .
SPECIAL_IDENT_START	""	See section Cross-Tool Tolerance Settings .
SYMBOL_AS_STRING	0	See section Cross-Tool Tolerance Settings .

8.4.4 [FILTER]

Filtering of A2L file

This section contains settings for filtering the A2L file. The filter settings are used to remove individual measurement and calibration values according to various search criteria.

Multiple filters can be linked each other.

Filter result

The filter result is relevant for the application of most modifications, to generate the label file and, if necessary for the definition of the objects that are applied to the target file.

If no filter is active, the quantity of all measurement and calibration values is used as filter result.

- Filter direction** When filtering is active, either all measurement and calibration values that satisfy the search criteria are ignored when saving the result file, or only these objects are transferred to the result file. This is specified via the filter direction.

- Search criteria of individual filters** Within a filter, various search criteria (e.g., name, address, and type) can be combined (AND operation of search criteria). I.e. in the filter result of a single, only the measurement and calibration values are included which meet all search criteria.

- Search criteria of linked filters** In order that a measurement and calibration value is part of the overall filter result of multiple linked filters, the value has to belong to the filter result of at least one filter (OR operation of individual filters).

- Handling of references** References to measurement and calibration values removed by the filter are automatically removed from groups and functions – regardless of whether the REMOVE_INVALID_REFERENCES option is set.

- Definition in the INI file** The first filter is defined in the main section [FILTER].
 Shall be defined multiple filters, the appropriate sections must be called [Filter_1], [Filter_2], etc. The settings within these sections correspond essentially to those from the section [FILTER].

- Global settings** The following settings are global and are valid for all filters, so these only exist in the main section [FILTER]:
 - > CONVERSION_METHODS
 - > DIRECTION
 - > INCLUDE_NAME
 - > PROVIDE_DETAILED_SEARCH_RESULT
 - > RECORD_LAYOUTS
 - > RESPECT_REFERENCES

Parameter	Default Value	Description
ACTIVE	0	Global switch for activating the filtering: > 0 = Not active > 1 = Active Notice: All other options in the [FILTER] section are ignored when the ACTIVE option is deactivated.
COMPARE_ADDRESS	0	Specifies whether the measurement and calibration values are to be filtered by address. In this case, of the non-virtual objects only those objects that are completely within the address range defined by START_ADDRESS and END_ADDRESS are transferred to the filter result. > 0 = No filtering by address > 1 = Filtering by address is active
CONVERSION_METHODS	2	Specifies the conversion rules and conversion tables that are to be transferred to the filter result when filtering is active. > 0 = No conversion rules are transferred to the filter result > 1 = Only referenced conversion rules are transferred to the filter result.

Parameter	Default Value	Description
		<p>> 2 = All conversion rules are transferred to the filter result.</p> <p>Notice: This option applies globally for all the filters and is always defined in the main section [FILTER].</p>
DATATYPE_MASK	0xFFFF	<p>Specifies which data type the measurement and calibration values that are transferred to the filter result must have.</p> <p>Various values can be combined in the mask (OR operation).</p> <ul style="list-style-type: none"> > 0x001 = Unsigned Byte > 0x002 = Signed Byte > 0x004 = Unsigned Word > 0x008 = Signed Word > 0x010 = Unsigned Long > 0x020 = Signed Long > 0x040 = Unsigned Int64 > 0x080 = Signed Int64 > 0x100 = Float > 0x200 = Double > 0xFFFF = No data type restriction
DEPENDENT_OBJECTS	0	<p>Specifies whether dependent measurement and calibration values are considered during filtering:</p> <ul style="list-style-type: none"> > 0 = All values are considered. > 1 = Only dependent values are included in the filter result. > 2 = Dependent values are not included in the filter result.
DIRECTION	0	<p>Specifies the filter direction:</p> <ul style="list-style-type: none"> > 0 = Only the filter result is transferred to the result file – thus, all measurement and calibration values that satisfy the search criteria. > 1 = The filter result is removed from the result file. Only the measurement and calibration values that do NOT satisfy the search criteria are transferred to the result file. > 2 = The filter result is removed from the result file and saved instead in a separate file that is added to the result file via an include. The name of the include file must be specified in the INCLUDE_NAME option. > 3 = All variables are transferred to the result file. The filter result in this case is only to select the variables that the modifications specified in [MODIFICATIONS] are to be applied to. <p>Notice: This option applies globally for all the filters and is always defined in the main section</p>

Parameter	Default Value	Description
		[FILTER].
END_ADDRESS	0xFFFFFFFF	Specifies the end address of the address range to be filtered by. In doing so, at END_ADDRESS the first address must be specified which no longer belongs to the address range. See also COMPARE_ADDRESS.
FILTER_NAME	""	With this option, an individual name can be assigned to each partial filter Messages about measurement and calibration values that satisfy the filter criteria can thus be interpreted correctly in the log file, for example.
INCLUDE_NAME	""	Specifies the name of include file to be created for the filter mode to SPLIT. The file name must be specified without path and file extension. The file extension *.A2L will be added automatically and the file is saved in the same directory as the result file. The created include file is an A2L fragment and not a complete A2L file. However, it can also be used by the other tools of the ASAP2 Tool-Set and by CANape as a stand-alone file. Notice: This option applies globally for all the filters and is always defined in the main section [FILTER].
MISSING_REFERENCES	0	Specifies whether measurement and calibration objects with missing references are considered during filtering: <ul style="list-style-type: none"> > 0 = All objects are considered. > 1 = Only objects with missing references are included in the filter result. > 2 = Objects with missing references are not included in the filter result.
OBJECT_TYPE_MASK	0xFFF	Specifies which object types are transferred to the filter result. Various values can be combined in the mask (OR operation). <ul style="list-style-type: none"> > 0x001 = Measurement values > 0x002 = Parameters > 0x004 = ASCII strings > 0x008 = Shared axes > 0x010 = Characteristic curves > 0x020 = Maps > 0x040 = Cuboids (3-, 4-, 5-dimensional) > 0x080 = Value blocks > 0x100 = BLOBs > 0x200 = Structures > 0xFFF = No object type restriction

Parameter	Default Value	Description
PROVIDE_DETAILED_SEARCH_RESULT	0	<p>If this option is set, information about the measurement and calibration values that satisfy the filter criteria is written in the log file for all partial filters.</p> <p>If no measurement or calibration values satisfy the filter criteria, a corresponding message is written to the log file.</p>
REAL_ENABLED	1	<p>Specifies whether real measurement and calibration values are transferred to the filter result.</p> <ul style="list-style-type: none"> > 0 = Real objects not included in the filter result > 1 = Real objects may be included in the filter result.
RECORD_LAYOUTS	2	<p>Specifies the storage schemes that are transferred to the filter result when filtering is active.</p> <ul style="list-style-type: none"> > 0 = No storage schemes are transferred to the filter result > 1 = Only referenced storage schemes are transferred to the filter result. > 2 = All storage schemes are transferred to the filter result. <p>Notice: This option applies globally for all the filters and is always defined in the main section [FILTER].</p>
RESPECT_REFERENCES	0	<p>If this option is activated, it is ensured that an invalid result file does not result from removal of measurement and calibration values.</p> <p>With filter direction <code>In</code>, all measurement and calibration values that are referenced by at least one object in the result file are also transferred to the result file regardless of whether they satisfy the filter criteria.</p> <p>With filter direction <code>Out</code>, all measurement and calibration values that reference a removed object are also removed from the result file regardless of whether they satisfy the filter criteria.</p> <ul style="list-style-type: none"> > 0 = Object references are not considered. > 1 = Object references are considered. <p>Notice: This option applies globally for all the filters and is always defined in the main section [FILTER].</p>
SEARCH_CASE_SENSITIVE	0	<p>Specifies whether text comparisons are to be case-sensitive.</p> <ul style="list-style-type: none"> > 0 = Not case-sensitive. > 1 = Case-sensitive. <p>Notice: If the option <code>USE_FAST_FILTERS</code> is set, this option is ineffective.</p>
SEARCH_COMMENT	""	<p>Search string for the object comment.</p> <p>If filtering by object comment is not to be performed, an empty string must be specified here.</p>

Parameter	Default Value	Description
		Several search strings can be specified, separated by semicolon.
SEARCH_CONVERSION	""	Search string for the name of the conversion rule. Only measurement and calibration values whose conversion rule corresponds to the search string are transferred to the filter result. For maps, the axes are also taken into consideration here. If filtering by conversion rule is not to be performed, an empty string must be specified here. If filtering by measurement and calibration values that use NO conversion rule is to be performed, the name NO_COMPU_METHOD must be specified. Several search strings can be specified, separated by semicolon.
SEARCH_FULL_TEXT	0	Specifies for text comparisons whether the search string is to be contained in the object name or is to match the object name completely. > 0 = Object name must match the search string exactly. > 1 = Object name must contain the search string. Notice: This option cannot be combined with the <Regular expressions> mode. Therefore, the option is ignored when <Regular expressions> is set. If the option USE_FAST_FILTERS is set, this option is ineffective.
SEARCH_GROUP	""	Search string for the group name. Only measurement and calibration values that belong to at least one group whose name corresponds to the search string are transferred to the filter result. If filtering by group name is not to be performed, an empty string must be specified here. Several search strings can be specified, separated by semicolon.
SEARCH_NAME	""	Search string for the object name. If filtering by object name is not to be performed, an empty string must be specified here. Several search strings can be specified, separated by semicolon. Example: SEARCH_NAME=A*;B* (in combination with search mode <Wildcards>): only the measurement and calibration values whose name begins with an A or B are transferred to the filter result.
SEARCH_SUB_GROUPS	0	Specifies whether subgroups are to be taken into consideration when filtering by group membership: > 0 = Subgroups are not considered. > 1 = Subgroups are considered.
START_ADDRESS	0	Specifies the start address of the address range to

Parameter	Default Value	Description
		be filtered by; see also <code>COMPARE_ADDRESS</code> .
<code>TEXT_SEARCH_MODE</code>	1	Specifies the mode for text comparisons for the search criteria: <ul style="list-style-type: none"> > 0 = Exact comparison > 1 = Use of wildcards > 2 = Use of regular expressions Notice: If the option <code>USE_FAST_FILTERS</code> is set, this option is ineffective.
<code>USE_DISPLAY_NAMES</code>	0	Specifies for filtering of object names whether the display name, and not the (standard) object name, is to be compared with the search string. <ul style="list-style-type: none"> > 0 = The object name is compared. > 1 = The display name is compared. Notice: If the option <code>USE_FAST_FILTERS</code> is set, this option is ineffective.
<code>USE_FAST_FILTERS</code>	0	If this option is set, only the variable name is used as filter criterion. All other search criteria are ignored. If the fast filter is activated, the following options are ineffective: <ul style="list-style-type: none"> > <code>SEARCH_CASE_SENSITIVE</code> > <code>SEARCH_FULL_TEXT</code> > <code>TEXT_SEARCH_MODE</code> > <code>USE_DISPLAY_NAMES</code> When searching, please note also the following: <ul style="list-style-type: none"> > Match cases. > No display names are used. > Wildcards and semicolons are not allowed.
<code>VIRTUAL_ENABLED</code>	1	Specifies whether virtual measurement and calibration values are transferred to the filter result: <ul style="list-style-type: none"> > 0 = Virtual objects are not included in the filter result > 1 = Virtual objects may be included in the filter result.
<code>WRITE_ACCESS</code>	0	Specifies which access rights the objects that are transferred to the filter result must have. <ul style="list-style-type: none"> > 0 = All objects are transferred. > 1 = Only read-only objects are transferred. > 2 = Only writable objects are transferred.

8.4.5 [OPTIMIZATION]

Optimization

This section contains settings for optimization (minimization) of the result file. Superfluous information or keywords can be removed from the result file even if these were available in the input file.

Parameter	Default Value	Description
MERGE_EQUAL_CONVERSION_METHODS	0	Minimizes the A2L file. If this option is set, identical conversion rules (equal except for the name) are merged to a single conversion rule. Redundant conversion rules are removed. The references at variables and typedefs are adapted accordingly.
MERGE_EQUAL_RECORD_LAYOUTS	0	Minimizes the A2L file. If this option is set, identical record layouts (equal except for the name) are merged to a single record layout. Redundant record layouts are removed. The references at variables and typedefs are adapted accordingly.
MERGE_EQUAL_TYPEDEFES	0	If this option is set, identical typedefs (equal except for the name) are merged to a single typedef. Redundant typedefs are removed. The references at instances and structures are adapted accordingly.
REMOVE_DISPLAY_IDENTIFIER	0	Removes the local <code>DISPLAY_IDENTIFIER</code> information for measurement and calibration values if the display name matches the object name. <ul style="list-style-type: none"> > 0 = <code>DISPLAY_IDENTIFIER</code> is not removed. > 1 = <code>DISPLAY_IDENTIFIER</code> is removed.
REMOVE_ECU_ADDRESS_EXTENSION	0	Removes the <code>ECU_ADDRESS_EXTENSION</code> information for measurement values. <ul style="list-style-type: none"> > 0 = <code>ECU_ADDRESS_EXTENSION</code> is not removed. > 1 = <code>ECU_ADDRESS_EXTENSION</code> is removed.
REMOVE_EMPTY_BIT_MASK	0	Removes the <code>BIT_MASK</code> information for measurement and calibration values if the bit mask is 0 or corresponds to the default bit mask of the data type (e.g., bit mask 0xFFFF for data type <code>SWORD</code>). <ul style="list-style-type: none"> > 0 = <code>BIT_MASK</code> is not removed. > 1 = <code>BIT_MASK</code> is removed.
REMOVE_EXTENDED_LIMITS	0	Removes the <code>EXTENDED_LIMITS</code> for calibration values if the extended limits match the "normal" limits. <ul style="list-style-type: none"> > 0 = <code>EXTENDED_LIMITS</code> is not removed. > 1 = <code>EXTENDED_LIMITS</code> is removed.
REMOVE_LOCAL_ALIGNMENT	0	Removes the local <code>ALIGNMENT</code> information for storage schemes if the value matches the corresponding global default value of the module. <ul style="list-style-type: none"> > 0 = <code>ALIGNMENT</code> is not removed. > 1 = <code>ALIGNMENT</code> is removed.

Parameter	Default Value	Description
REMOVE_LOCAL_BYTE_ORDER	0	Removes the local <code>BYTE_ORDER</code> information for measurement and calibration values if the local value matches the global default value of the module. > 0 = <code>BYTE_ORDER</code> is not removed. > 1 = <code>BYTE_ORDER</code> is removed.
REMOVE_LOCAL_DEPOSIT	0	Removes the local <code>DEPOSIT</code> information for axis descriptions if the local value matches the global default value of the module. > 0 = <code>DEPOSIT</code> is not removed. > 1 = <code>DEPOSIT</code> is removed.
REMOVE_LOCAL_FORMAT	0	Removes the local <code>FORMAT</code> information for measurement and calibration values if the local value matches the default value of the referenced conversion rule. > 0 = <code>FORMAT</code> is not removed. > 1 = <code>FORMAT</code> is removed.
REMOVE_LOCAL_UNIT	0	Removes the local <code>PHYS_UNIT</code> information for measurement and calibration values if the local value matches the default value of the referenced conversion rule. > 0 = <code>PHYS_UNIT</code> is not removed. > 1 = <code>PHYS_UNIT</code> is removed.

8.4.6 [EXCEL_UPDATE]

Updating with Excel tables

This section contains settings for updating existing variables and creating new variables based on Excel tables or CSV files.

Prerequisite

All variables, whose name is in the name column (see option `COLUMN_NAME`), can be updated according to the content of the remaining columns.

Optionally, new variables can be created for names that are not yet available in the database.

Identification

One row per variable is expected in the Excel file by default.

To identify the variables, the variable name in the name column is used.

An exception are objects that have axes (curves, maps and cuboids). For these objects, one row per axis is expected directly below the main row for the variable. The name field at the axes must be empty.

If objects with axes are created during import, the axes are always generated of type standard axis. Fixed axes and references to common axes are currently not yet supported.

	A	B	C	D	E	F	G	H
1	Name	Type	Dimension	Min	Max	Data Type	Factor	Offset
2	Measure1	Measure		-3	7	UWORD	2	1
3	MyParameter	Parameter		0	255	UBYTE	3	2
4	MyCurve	Curve		0	255	UBYTE	4	3
5			16	0	255	UBYTE		
6	MyMap	Map2D		0	255	UWORD	5	4
7			8	10	12	SBYTE		
8			7	11	13	DOUBLE		
9	ASCII	ASCII		0	255	UBYTE		
10	MyAxis	Axis	22	0	255	UBYTE	3	2
11	Measure1	Measure		-3	7	UWORD	2	1

Example: Excel table contains objects with axes (variables *MyCurve* and *MyMap*)

COLUMN_XX

The parameter names listed in the following table each describe a column header in the Excel file.

If the Excel table contains a column with the name **XX** specified in the option, the information of all variables is modified according to the content of this column.

Empty cells

Empty cells in the columns are ignored unless there is a different behavior listed in the respective option description.

Parameter Name	Default Value	Description
COLUMN_ADDRESS	""	Column for the address information. The address must be specified in hexadecimal form.
COLUMN_ADDRESS_EXTENSION	""	Column for the address extension. The address extension must be specified in decimal form.
COLUMN_CALIBRATION_ACCESS	""	Column for the calibration access. Only special predefined values are permitted for the values in the calibration-access column: <ul style="list-style-type: none"> > ReadOnly > Full > None > NotInMcd > Offline
COLUMN_COMMENT	""	Column for the comment. Empty cells in this column are treated according to the <code>IGNORE_EMPTY_CELLS_FOR_STRING_VALUES</code> option.
COLUMN_DAQ_EVENT_CCP	""	Column for the CCP default raster of the measurement. The raster must be specified in decimal form. The AML for the <code>IF_DATA ASAP1B_CCP</code> must be already available in the A2L file.
COLUMN_DAQ_EVENT_XCP	""	Column for the XCP default event of the measurement. The event must be specified in decimal form. Multiple events can be specified separated by a comma or space character. The AML for the <code>IF_DATA XCP</code> must be already available in the A2L file.
COLUMN_DAQ_EVENT_XCP_PLUS	""	Column for the XCPPlus default event of the measurement. The event must be specified in decimal form. Multiple events

Parameter Name	Default Value	Description
		can be specified separated by a comma or space character. The AML for the IF_DATA XCPplus must be already available in the A2L file.
COLUMN_DATATYPE	""	<p>Column with the data type of the variable.</p> <p>The data type is mandatory when creating new variables. If the Excel file does not contain a column for the data type, the value specified for DEFAULT_DATATYPE is used.</p> <p>Only special predefined values are permitted for the values in the datatype column:</p> <ul style="list-style-type: none"> > DOUBLE > FLOAT > SBYTE > SINT64 > SLONG > SWORD > UBYTE > UINT64 > ULONG > UWORD <p>You can also define your own additional datatype values that are accepted in the Excel file, e.g. using DATATYPE_UBYTE.</p>
COLUMN_DIMENSION	""	<p>Column for the dimension of common axes and axes of curves/maps.</p> <p>In addition, the dimension for variables of the variable type ASCII, BLOB, VALUEBLOCK and MEASURE can be set.</p> <p>For multidimensional objects, the individual dimension values must be separated by space characters.</p> <p>Example: To specify a two-dimensional value block, the dimension column contains a string like "100 20".</p>
COLUMN_DISPLAY_IDENTIFIER	""	Column for the display identifier.
COLUMN_EXTENDED_MAX	""	Column for the maximum value of the extended limit.
COLUMN_EXTENDED_MIN	""	Column for the minimum value of the extended limit.
COLUMN_FACTOR	""	<p>Column for the factor of the linear conversion method.</p> <p>If the variable does not have a linear conversion method, a new one is generated if necessary.</p>
COLUMN_INPUT_QUANTITY	""	<p>Column for the input value of common axes and axes of curves/maps.</p> <p>The names listed in the Excel file in this column must be valid ASAP2 identifiers. If necessary, they are automatically corrected by the tool. Additionally, a measurement object with the referenced name should exist in the A2L file, otherwise an inconsistency will occur.</p>
COLUMN_LOWER_LIMIT	""	Column for the minimum value of the default limits.
COLUMN_MODELLINK	""	Column for the model name.
COLUMN_NAME	"Name"	<p>Column with the variable name.</p> <p>This column must be available in the Excel file, since the</p>

Parameter Name	Default Value	Description
		variables are identified by the names in this column. Notice: If the option <code>CREATE_NEW_VARIABLES</code> is active, new variables are created for all names in the name column for which no corresponding variable is found in the database.
<code>COLUMN_OFFSET</code>	""	Column for the offset of the linear conversion method. If the variable does not have a linear conversion method, a new one is generated if necessary.
<code>COLUMN_PHYS_UNIT</code>	""	Column for the physical unit. Empty cells in this column are treated according to the <code>IGNORE_EMPTY_CELLS_FOR_STRING_VALUES</code> option.
<code>COLUMN_READ_WRITE</code>	""	Column for the <code>READ_WRITE</code> flag: <ul style="list-style-type: none"> > Value unequal 0 = <code>READ_WRITE</code> flag is added > Value equals 0 = <code>READ_WRITE</code> flag is removed if applicable
<code>COLUMN_SYMBOL_NAME</code>	""	Column for the symbol name (name in the MAP file).
<code>COLUMN_SYMBOL_OFFSET</code>	""	Column for the offset to the symbol name. The offset must be specified in decimal form.
<code>COLUMN_UPPER_LIMIT</code>	""	Column for the maximum value of the default limits.
<code>COLUMN_VARIABLETYPE</code>	""	Column with the variable type. This column is only needed to create new variables. However, you cannot change the variable type of already existing variable. Only special predefined values are permitted for the values in this column: <ul style="list-style-type: none"> > ASCII > AXIS > BLOB > CUBE4 > CUBE5 > CUBOID > CURVE > MAP > MEASURE > PARAMETER > VALUEBLOCK You can also define your own additional values that are accepted in the Excel file, e.g. using <code>VARIABLETYPE_MAP</code> . Notice: If the Excel file does not contain a column for the variable type, the type specified for <code>DEFAULT_VARIABLETYPE</code> is used to create new variables.

Further options

The next table contains all further settings for updating with Excel tables:

Parameter Name	Default Value	Description
CREATE_GROUPS	1	<p>This option specifies whether a group per worksheet should be generated in the database when creating variables (see <code>CREATE_NEW_VARIABLES</code>).</p> <p>The names of the worksheets are used as group names and adapted if required, since the group names in the A2L file must be valid identifiers. If necessary, the generated variables are assigned automatically to the newly created group.</p>
CREATE_NEW_VARIABLES	0	<p>If this option is active, new variables are created for all names in the name column for which no corresponding variable is found in the database. The variable type is defined either by the content of the column <code>COLUMN_VARIABLETYPE</code> or (if this column does not exist) by <code>DEFAULT_VARIABLETYPE</code>.</p> <p>The data type of the new variable is defined either by the content of the column <code>COLUMN_DATATYPE</code> or (if this column does not exist) by <code>DEFAULT_DATATYPE</code>.</p> <p>All other variable properties are taken from the optionally configured columns.</p>
DATATYPE_<datatype>	""	<p>These options always relate to the data type that is defined for the placeholder <datatype>. It specifies an alternative text for the defined data type that is to be accepted as such in the Excel file.</p> <p>The option is available for the following data types and the parameters are named accordingly as follows:</p> <ul style="list-style-type: none"> > DATATYPE_DOUBLE > DATATYPE_FLOAT16 > DATATYPE_FLOAT32 > DATATYPE_SBYTE > DATATYPE_SINT64 > DATATYPE_SLONG > DATATYPE_SWORD > DATATYPE_UBYTE > DATATYPE_UINT64 > DATATYPE_ULONG > DATATYPE_UWORD
DEFAULT_DATATYPE	"UBYTE"	<p>This option specifies the data type that is to be used to create new variables if no separate column for the data type is available in the Excel file.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> > "DOUBLE" > "FLOAT16" > "FLOAT32"

Parameter Name	Default Value	Description
		<ul style="list-style-type: none"> > "SBYTE" > "SINT64" > "SLONG" > "SWORD" > "UBYTE" > "UINT64" > "ULONG" > "UWORD"
DEFAULT_VARIABLETYPE	"MEASUREMENT"	<p>This option specifies the variable type that is to be used to create new variables if no separate column for the variable type is available in the Excel file.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> > "ASCII" > "AXIS" > "BLOB" > "CUBE3" > "CUBE4" > "CUBE5" > "CURVE" > "MAP" > "MEASUREMENT" > "PARAMETER" > "VALUEBLOCK"
HEADER_ROW	0	This option specifies the row number where the column headers are in the imported worksheet.
IGNORE_EMPTY_CELLS_FOR_STRING_VALUES	0	<p>This option specifies whether empty cells in the Excel table shall be ignored or empty strings (e.g., for the physical unit or the comment) shall be taken.</p> <p>Notice: Whether this option is used, depends on the column and is explained under the respective column-related option.</p>
IMPORT_ALL_SHEETS	0	If this option is active, the name specified with option SHEET_NAME is ignored and all visible worksheets are imported.
SHEET_NAME	""	<p>This option specifies the name of the worksheet that is to be imported. If no name is specified, the first worksheet is imported.</p> <p>Notice: This option is only relevant when importing Excel files, not CSV files.</p>
VARIABLETYPE_<variabletype>	""	These options always relate to the variable type that is defined for the placeholder <variabletype>. It specifies an alternative text for the defined variable type that is to be accepted as such in the Excel file.

Parameter Name	Default Value	Description
		<p>The option is available for the following variable types and the parameters are named accordingly as follows:</p> <ul style="list-style-type: none"> > VARIABLETYPE_ASCII > VARIABLETYPE_AXIS > VARIABLETYPE_BLOB > VARIABLETYPE_CUBE3 > VARIABLETYPE_CUBE4 > VARIABLETYPE_CUBE5 > VARIABLETYPE_CURVE > VARIABLETYPE_MAP > VARIABLETYPE_MEASUREMENT > VARIABLETYPE_PARAMETER > VARIABLETYPE_VALUEBLOCK

8.4.7 [CREATE_MEASURE_ARRAYS]

Merging measurement arrays

This section contains settings for merging individual measurements into measurement arrays in an A2L file.

Method

To create measurement arrays, the variable names of all scalar measurements are parsed to identify array indices.

All individual measurements except the base variables (those with index 0) are removed from the A2L. The measurement with the index 0 is converted into a measurement array with the determined dimension.



Note:

The following conditions must be kept:

- > All individual measurements that are to be merged to an array, basically must have the same definition. That is, they must match in all essential properties except the address and the name (e.g. the same conversion rule, the same limits, etc.).
- > The addresses and offsets must match the datatype size.

Parameter	Default Value	Description
ARRAY_INDEX_FORM	0	<p>Specifies the index form for array indices:</p> <ul style="list-style-type: none"> > 0 = No measurement arrays are created. > 1 = Array indices are expected in the form array._0_.1_.2_. > 2 = Array indices are expected in the form array[0][1][2].
REMOVE_ELEMENTS_WITH_WRONG_ADDRESS	0	<p>If this option is activated, unnecessary base variables are removed from the database while generating the array. As base variables, those variables with</p>

Parameter	Default Value	Description
		<p>implausible addresses are considered.</p> <p>Example: An A2L file has the following measurements of data type UBYTE:</p> <ul style="list-style-type: none"> > Address 0x1000: A[0] > Address 0x1000: A[0].X > Address 0x1001: A[0].Y > Address 0x1002: A[1] > Address 0x1002: A[1].X > Address 0x1003: A[1].Y <p>When creating the array, A[0] and A[1] are first detected as potential elements of an array A. While checking data type and address, however, it is determined that A cannot really be an array of scalar values of type UBYTE. Thus, no measurement array A is created.</p> <p>If this option is activated, additionally the implausible variables A[0] and A[1] are removed from the database.</p>
REMOVE_UNNEEDED_BASE_VARIABLES	0	<p>If this option is activated, unnecessary base variables are removed from the database while generating the array.</p> <p>Variables with less than the maximum number of dimensions or with child elements are considered as base variables.</p> <p>Example: An A2L file has the following measurements: A, A[0], A[1], A[0][0], A[0][1], A[1][0], A[1][1]</p> <p>For the variable A[0][0], the array generation sets the dimension 2x2 and removes the variables A[0][1], A[1][0], A[1][1].</p> <p>If this option is</p> <ul style="list-style-type: none"> > activated, additionally the variables A, A[0] and A[1] are removed and the variable A[0][0] is then renamed into A. > not activated, the variables A, A[0] and A[1] are kept and the variable A[0][0] is renamed to A_Copy1.
RESTRICT_TO_2_DIMENSIONS	0	<p>If this option is activated, only those measurement arrays are created that have a maximum of 2 dimensions.</p> <p>Individual measurements belonging to a 3-dimensional or multidimensional array remain unchanged.</p> <p>Notice: CANape does not support measurement arrays with more than 2 dimensions yet.</p>
SKIP_COMPARE_TYPEDEF	0	<p>This option deactivates the check whether the definitions of variables, that should be merged</p>

Parameter	Default Value	Description
		into a measurement array, have the same content. This significantly shortens the runtime. Notice: This option should be activated only if it is ensured that the relevant individual measurements are really identical in content and belong to an array, e.g. by automated A2L generation.

8.4.8 [CREATE_STRUCTURES]

Creating structures and typedefs This section contains settings for creating structures and typedefs in the A2L file.

Method For creating structures, the variable names of all configured objects are parsed to identify structure components and array indices and afterwards generating typedefs and instances.

Interpretation Thereby a dot "." is interpreted as a delimiter between structure name and component name, and array indices are only supported in the form "[x]". If necessary, the array indices must first be normalized in a separate Modifier run.



Note:

The following conditions must be kept:

- > All objects that are to be described by a common typedef, basically must have the same definition. That is, they must match in all essential properties except the address and the name (e.g. the same conversion rule, the same limits, etc.).
- > The addresses and offsets must match the datatype size.

Parameter	Default Value	Description
ADDITIONAL_SCOPE_SEPARATOR_X	""	Specifies a string that is interpreted as scope separator when parsing the object names. Before actually parsing the names, the scope separators are replaced by a dot ".". X may have continuous numbers starting with 1. All options are read until one option returns the value "". Example: An A2L file has an object name X._.Y. Without additional scope separator a structure X would be created, which has a substructure with the name "_" and this, in turn, a data member named Y. However, if "." is configured as additional scope separator, the object name is renamed from X._.Y to X.Y before parsing; thereafter a structure X with the data member Y is created.
CREATE	0	Global switch for activating structure creation: > 0 = Not active

Parameter	Default Value	Description
		> 1 = Active
CREATE_SYMBOL_LINKS	0	<p>If this option is activated, also symbol links are created for the generated structures and typedefs in the A2L.</p> <p>Precondition: This option should only be activated if all variables belonging to a structure are included in the A2L file and are considered during structure generation. Particularly ensure that the first element of a structure used for generation (the element with the smallest address) is actually at the start address of the entire structure.</p> <p>Notice: This option should not be activated if the structure contains both measurement and calibration objects. In this case, not all structure elements can be used when generating.</p>
LOG_DETAILED	0	If this option is activated, detailed messages are written to the log file during generation.
OBJECT_TYPE	MEASURE	<p>This option specifies which objects you want to include in the structure creation.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> > MEASURE (exclusive measurement objects) > PARAMETER (exclusive characteristic objects) > ALL (There is no restriction for the created structures. However, it is ensured that the generated structures contain either only measurement objects or only characteristic objects. If necessary, the structures are kept appropriately small.)
REMOVE_UNNEEDED_BASE_VARIABLES	0	<p>If this option is activated, unnecessary base variables are removed from the database while generating the structure.</p> <p>As base variables, those variables with less than the maximum number of dimensions are considered.</p> <p>Example: An A2L file has the following objects: A, A.B, A.C, A.B.X, A.B.Y, A.C.Z</p> <p>The structure generation removes the variables A.B.X, A.B.Y and A.C.Z and inserts a structure type and an instance instead.</p> <p>If this option is</p> <ul style="list-style-type: none"> > activated, additionally the variables A, A.B and A.C are removed, and the new instance gets the name A. > not activated, the variables A, A.B and A.C are kept, and the newly created instance gets the name A_Copy1.
SKIP_COMPARE_TYPEDEF	0	This option deactivates the check whether the definitions of variables, that should be merged to structures and arrays, have the same content.

Parameter	Default Value	Description
		This significantly shortens the runtime. Notice: This option should be activated only if it is ensured that the relevant variables really belong to a structure, e.g. by automated A2L generation.

8.4.9 [CREATE_XCP_DAQ_EVENTS]

Event assignment in A2L file

In this section, you can configure how to generate default XCP events for measurement objects in the A2L file.

JSON file

Therefor the DAQ events and their assignment to measurement signals are described in a separate JSON file. Its structure is shown in section **JSON File for XCP Event Description** on page 154.

Call

The JSON file is specified using the **Command Line Parameters** `-E <name>`.



Note: A prerequisite for all options in this section is the global switch `CREATE`. If this option is set, the event assignment read from the JSON file is imported into the A2L file in principle.

Parameter	Default Value	Description
CREATE	0	If this option is active, the event assignment read from the JSON file is imported into the A2L file. Notice: As a global switch, this option is a prerequisite for all other options in this section.
REMOVE_EXISTING_ASSIGNMENTS	0	If this option is active, the existing event assignments are removed from the A2L file before the import of the event assignments.
REMOVE_EXISTING_EVENTS	0	If this option is active, the existing event definitions are removed from the A2L file before the import of the events.
TAG	XCP	Specifies the AML type for which the event assignment is generated. Possible values are XCP and XCPplus.

8.5 Warning Level

Various warning levels

The warnings in the log file are subdivided into various warning levels. These also reference messages of type `Information`:

Level	Message/Meaning
0	<ul style="list-style-type: none"> > Error messages/warnings concerning syntax errors in the A2L file that are nevertheless tolerated by the ASAP2 Modifier and do not cause a program abort > Messages about ASAP2 keywords that were available in the input file but are not saved in the result file due to the ASAP2 version setting. > Messages about objects that are missing in the result file because they cannot be saved due to the ASAP2 version setting (e.g., data type <code>INT64</code>)

1	<ul style="list-style-type: none"> > Messages about objects that do not satisfy the filter conditions but are nevertheless also taken into consideration because of references. > Messages about program settings that are set implicitly.
2	<ul style="list-style-type: none"> > Messages about objects that satisfy the filter criteria.

8.6 CANape DBU Format

DBU

The **database-update** format is a simple ASCII file format to update variable properties in the ECU's database.

If a name does not exist in the ECU's database while importing, a new measurement object is created for it.

Information supplied

- > Addresses
- > Data types
- > Conversion rules with units and positions after decimal point
- > Physical limiting values
- > Comments
- > New names
- > Annotations

Deleting variables

Variables can be renamed or deleted. Groups can be deleted (optionally including their content).

8.6.1 Format description in BNF

```

<update_definition> ::= UPDATE <name>[ <address> ][ <data_type> ]
                        [ '<' <factor> <offset> '>' [ <precision> ] <unit> ]
                        [ '(' <min> <max> ')' ]
                        [ <comment> ]
                        [ RENAME <new_name> ]
                        [ GROUP <group_name> ]
                        [ ADD_ANNOTATION <label> <origin> <text> ]
                        [ SET_ANNOTATION <label> <origin> <text> ]
                        | UPDATE <name> DELETE
                        | UPDATE DELETE_GROUP <name>
                        [ DELETE_SUB_GROUPS ] [ DELETE_VARIABLES ]

```

<name>	Name of the object in single quotation marks
<address>	Hexadecimal address of the variable
<factor> <offset>	Factor and offset of the linear conversion rule
<min>, <max>	Range limits for the variable
<unit>	Physical unit, string in double quotation marks
<comment>	Comment, string in double quotation marks
<precision>	Number of places after the decimal point as an integer
<new_name>	New variable name when renaming in single quotation marks

<group_name>	Name of the group to which the variable is to be assigned in single quotation marks
<label> <origin> <text>	Description of the annotation, string in double quotation marks



Note: Parameters in squared brackets are optional.



Example:

```
UPDATE 'RPM' 0x1AB1 UINT(16)
UPDATE 'RPM' UINT(16) "Comment text"
UPDATE 'RPM' UINT(16) < 2 5 > "Unit" "Comment text"
UPDATE 'RPM' < 2 5 > "Unit" ( 0 5 )
UPDATE 'RPM' < 2 5 > 1 "Unit" ( 0 5 ) RENAME 'N'
UPDATE 'RPM' GROUP 'Test'
UPDATE 'RPM' DELETE
```

8.7 JSON File for XCP Event Description

Purpose The JSON file describes XCP events and their assignment to measurement signals.

Formal description The formal description of the expected JSON scheme expected is as follows:

```
{
  "$schema": "http://json-schema.org/draft-07/schema",
  "type": "object",
  "required": [
    "XcpDaqEventDefinitions",
    "XcpDaqEventAssignments"
  ],
  "properties": {
    "XcpDaqEventDefinitions": {
      "type": "array",
      "default": [],
      "items": {
        "type": "object",
        "description": "one event definition corresponding to XCP AML",
        "default": {},
        "required": [
          "Name",
          "ShortName",
          "ChannelNumber",
          "DaqListType",
          "MaxDaqList",
```

```

    "TimeCycle",
    "TimeUnit",
    "Priority"
  ],
  "properties": {
    "Name": {
      "type": "string",
      "default": "",
    },
    "ShortName": {
      "type": "string",
      "default": "",
    },
    "ChannelNumber": {
      "type": "integer",
      "default": 0,
    },
    "DaqListType": {
      "type": "string",
      "description": "type of DAQ list",
      "default": "DAQ",
      "enum": ["DAQ", "STIM", "DAQ_STIM"]
    },
    "MaxDaqList": {
      "type": "integer",
      "default": 0,
    },
    "TimeCycle": {
      "type": "integer",
      "default": 0,
    },
    "TimeUnit": {
      "type": "integer",
      "default": 0,
    },
    "Priority": {
      "type": "integer",
      "default": 0,
    }
  }
}
},
"XcpDaqEventAssignments": {
  "type": "array",
  "default": [],
  "items": {
    "type": "object",

```

```
"description": "Assignment of DAQ events to measurement signals",
"default": {},
"required": [
  "SignalName",
],
"properties": {
  "SignalName": {
    "type": "string",
    "default": "",
  },
  "FixedEvents": {
    "type": "array",
    "default": [],
    "items": {
      "type": "object",
      "default": {},
      "required": [
        "ChannelNumber"
      ],
      "properties": {
        "ChannelNumber": {
          "type": "integer",
          "default": 0,
        }
      }
    }
  },
  "VariableEvents": {
    "type": "array",
    "default": [],
    "items": {
      "type": "object",
      "default": {},
      "required": [
        "ChannelNumber",
      ],
      "properties": {
        "ChannelNumber": {
          "type": "integer",
          "default": 0,
        },
        "IsDefault": {
          "type": "boolean",
          "default": false,
        }
      }
    }
  }
}
```



```
]
},
{
  "SignalName": "MyStructure.A2.D",
  "FixedEvents": [
    { "ChannelNumber" : 68 }
  ]
},
],
}
```


9 ASAP2 Studio

In this chapter, you will find the following information:

9.1	Functionality	page 159
9.2	Structure of the User Interface	page 160

9.1 Functionality

Create, visualize, manage

The **ASAP2 Studio** is the right solution for creating and visualizing standardized ECU description files. The **ASAP2 Studio** is used to manage ECU databases in A2L format. With its help you can view, enter and change all the information in a dialog-box-driven manner.

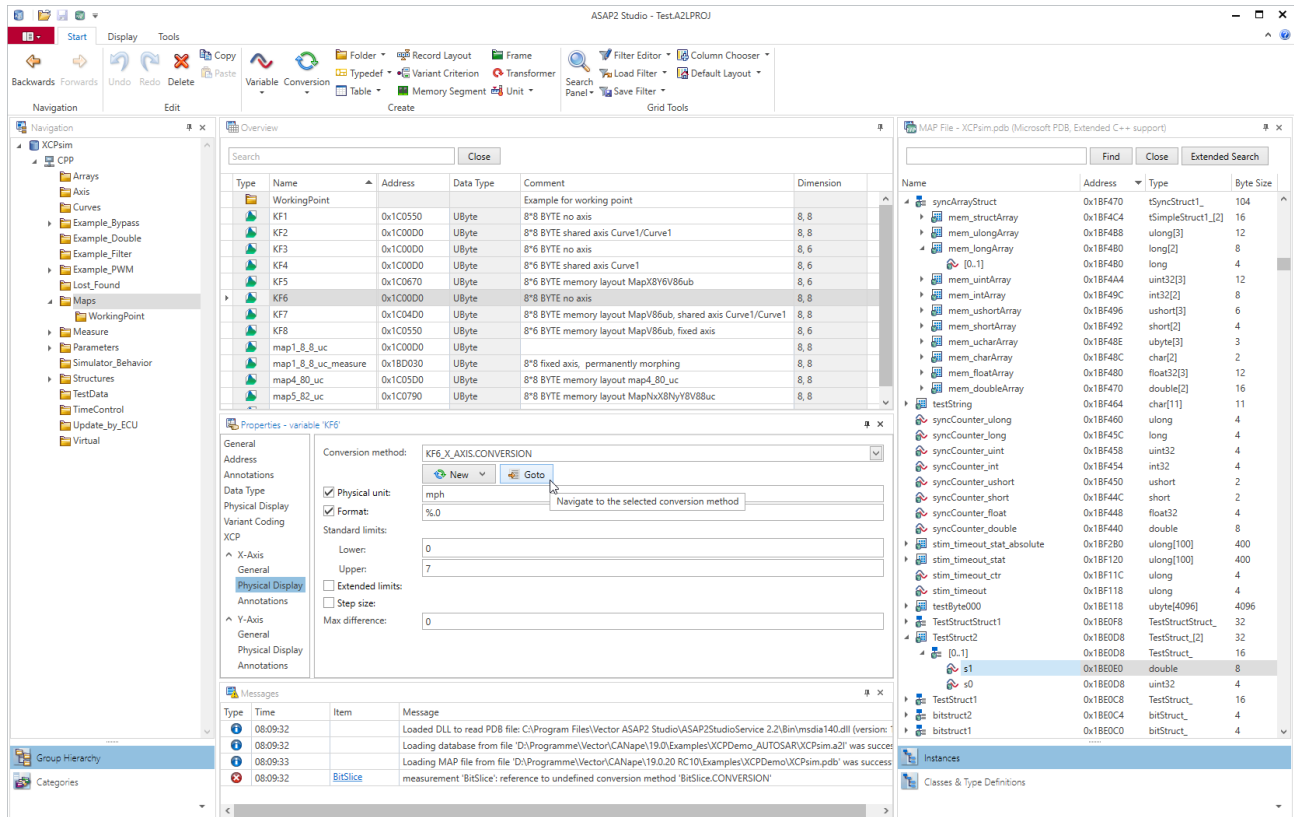
With the **ASAP2 Studio**, you can visualize A2L files and manually create and edit them. All information can be entered and changed via dialogs. The entire command line tools of the Tool-Set are integrated and can be configured and controlled dialog-guided, too. In addition, the contents of MAP and debug files can be visualized.

Functional range

The **ASAP2 Studio** offers the following functions:

- > Individual database objects can be visualized and edited in specific partial windows.
- > Changes can be undone and restored step by step.
- > Bulk operations facilitate simultaneously modifying multiple objects.
- > User-friendly navigation allows you to navigate between objects.
- > The configuration of A2L and associated MAP files can be managed in A2L project files.
- > Flexibly dockable partial windows allow an individual layout adapted to the respective requirements.
- > A2L files can be opened from Windows Explorer using drag-and-drop.
- > MAP and debug files can be visualized. They can also be used to generate new measurement and calibration objects.
- > Address and structure information can be updated via the integrated **ASAP2 Updater**.
- > Database objects can be imported from other A2L files or from third-party formats such as Excel or DBU.
- > Subsets of the database can be exported to other A2L files or to third-party formats such as Excel.
- > The integrated **ASAP2 Comparer** compares A2L files. The comparison result can be visualized and exported for documentation reasons.
- > Advanced filter criteria allow a fast search - both within the database and within the MAP file.
- > The integrated **ASAP2 Checker** can perform various plausibility checks.
- >

9.2 Structure of the User Interface



Main window

By default, the **ASAP2 Studio** is divided into five subwindows. The **Overview** window is always visible; it cannot be hidden.

The **Overview window** displays various properties of the elements selected in the **Navigation window** in a table view. The display varies here depending on the type of element that is selected. At the same time, the individual properties of the selected element are displayed in the **Properties window**.

Navigation

The left subwindow **Navigation** shows a tree view of all groups of the database objects and conversion rules, storage schemes, version criteria and system constants defined in the database description file.

Message window

A list of status and any error messages appears at the bottom in the **Message window**.

MAP File window

The MAP file belonging to the database is displayed in the **MAP File window**. With the tabs at the bottom you can be switch between the display of type definitions and instances.

Symbols

Groups, measurement signals, parameters, curves, maps, ASCII strings and virtual signals are differentiated from one another using different symbols.

Compare Result

The **Compare Result window** shows detailed differences when comparing two A2L files.

MAP Symbol - Properties

The detailed window for **MAP symbols** provides a better visualization of complex elements in the MAP file.



Note: In the help of the **ASAP2 Studio** all user interface elements and settings of the dialogs are described in detail. Open the help via **|Help|Contents**, or use the

 icon on the top right-hand side or **<F1>** which opens the help for the currently active tab.

10 Addresses

Addresses on Vector
homepage

Please find the contacts of Vector Informatik GmbH and all subsidiaries worldwide
via:

<https://www.vector.com/int/en/company/contacts/>



More Information

- > News
- > Products
- > Demo Software
- > Support
- > Training Classes
- > Addresses

www.vector.com