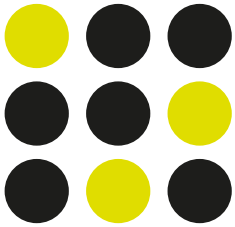


UDOKA

Acam 1.5p7
User Guideline

Acam User Guideline

Table of Contents

List of Tables

Introduction

Acam is a suite of four independent tools:

- Creator - Creator generates A2L files for hand-written C-code with a minimum of developer input. As much information as possible is extracted from debug information. This approach minimizes development and maintenance effort hence enabling rapid development.
- Merge - Merge merges several A2L files into a single one. It’s a must when your software is composed of several components. These components might be for example Simulink models, TargetLink models, Ascet or hand-written C code. The A2L files for these components can be generated by the tool which generates the C-code, hand-written or generated by UeAl2Creator. Regardless of how they’re created the A2L files describing the components must be merged into a single A2L file before they can be used with a measurement and calibration tool such as ATI Vision, Vector CANape, ETAS INCA etc. Merge accomplishes this task smoothly.
- Updater - Every measurement and calibration tool must know the memory address of the signals and calibration parameters. These addresses frequently change when the software is rebuilt. Hence the A2L file which describes the software must be updated with the memory address of each signal and calibration parameter whenever the software is rebuilt. UeAl2Updater performs this task.
- Filter - To protect intellectual property or preventing accidental calibration changes its often a good idea to remove some measurements, characteristics, groups and functions before sending the software and A2L to customers or partners. Filter accomplishes this task by filtering the A2L file. It offers a rich set of expressions for selecting which items shall be kept or removed.

Installation

Core

There is no setup or installation program. Just unzip the delivered archive and execute the tool.

License management

Acam uses CodeMeter technology for license management. A CodeMeter runtime is required to use Acam. It can be downloaded for free from www.udokaelectronics.com or codemeter.com. Install CodeMeter Runtime on the computer on which Acam will be executed and then proceed according to the instructions below according to the chosen license model.

Evaluation license

This license model ties a license to a single computer. No additional hardware is required. Follow the steps below to activate your license.

1. Send a mail to sales@udokaelectronics.com requesting an evaluation license.

2. Start CodeMeter Control Center.

3. Choose *File->Import License...* and select the .WibuCmRau file you received in reply from sales@udokaelectronics.com

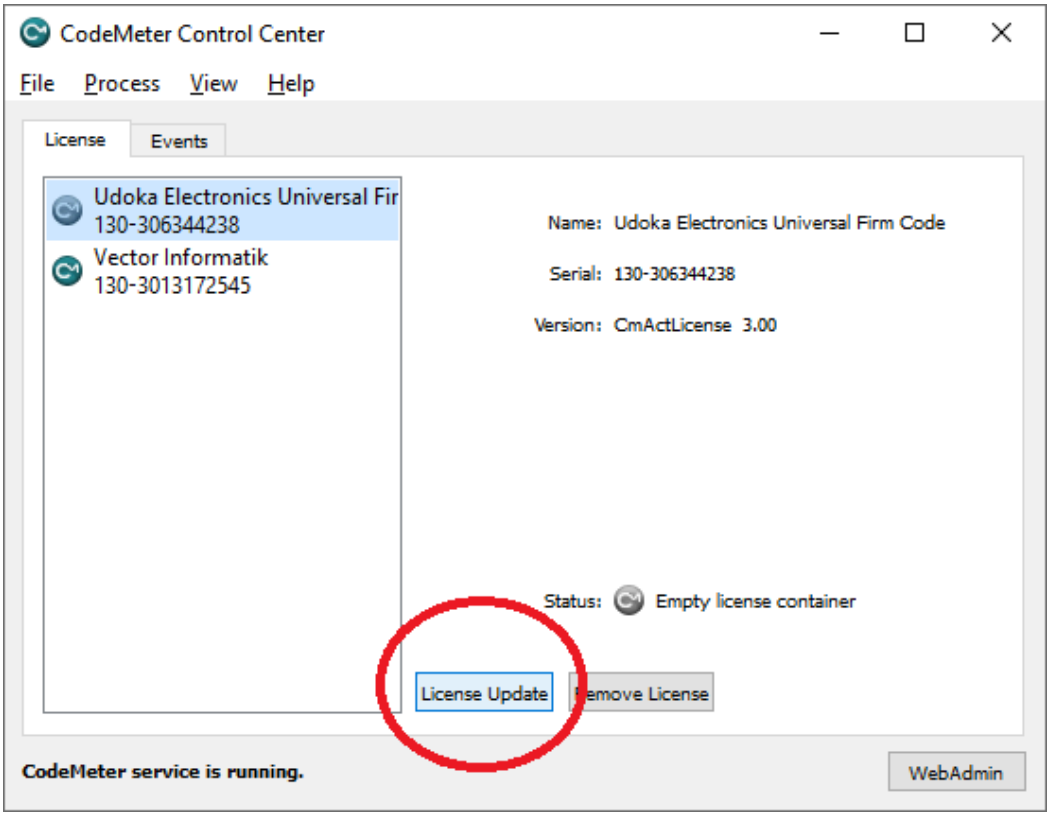
Node-locked license

This license model ties a license to a single computer. No additional hardware is required. Follow the steps below to activate your license.

1. Start CodeMeter Control Center.

2. Choose *File->Import License...* and select UdokaElectronis-Perpetual.WibuCmLIF (included in Acam delivery package).

3. Select the CmActLicense imported in previous step and click *License Update*. See screenshot below for example.



4. A wizard will be started. Use the wizard to create a license request. Send the exported license request file to sales@udokaelectronics.com.

5. In return from sales@udokaelectronics.com you will get a license update file. Use the same wizard for importing this license update files as you used to create the license request.

USB Dongle

In this case UdokaElectronics will send you the dongle. Once you're received it secure that the dongle is connected to the computer.

Floating license

A floating license can be tied to a computer or a USB dongle. Follow the instructions for Node-locked or USB dongle.

Invocation

All tools have a command line interface. They can be used directly from a command line or integrated in an automated build environment. There is an extensive help including example invocations built-in in each tool. Just pass the command line option -h to display it, i.e. **Merge.exe -h.**

A2L file encoding

Both ANSI and UTF-8 is supported for input A2L files. The encoding will be detected by Acam, i.e. there is no command line option for encoding. Output A2L files will be encoded in UTF-8 (default) or UTF-8 BOM.

Creator

Overview

Creator generates A2L files for hand-written and generated C/C++-code with a minimum of developer input. The core is a unique algorithm which analyzes the build products and derives the A2L information from this analysis. This approach, to automate the A2L creation to as far as possible, lets developers to focus on developing their applications instead of writing an maintaining A2L information. The result is fast development and an A2L file without bugs.

Not all information can be derived from build products. The developer enters this information in comments in the source code.

Adding A2L information in source code comments

Description

The following bullets must be adhered in order for Creator to be able to associate the information in the source code comments with the variable to be published in the A2L file.

- Both // and /**/ is supported.
- The variable (measurement or description) must be immediately adjacent to the comment block. There must not be any whitespace between the comment block and the variable declaration.
- Variable declarations must not span multiple lines.
- The first line in the comment block with an alphanumeric character will be interpreted as a one-line description for scalars and arrays of scalars. It wil be written as the description in the generated A2L file.
- A2l info is given in tags similar to javadoc. See the examples in the end of this section for an illustration of how to write the tags. Note that no tag is mandatory. U2A2lCreator will try to derive all information. Warnings will be issued if Creator fails to derive a necessary piece of information.
- When the variable is a scalar or an array of scalars, i.e. no struct involved, then naturally the tagged A2L info applies to the entire object. Limits, description etc. are the same for each member of an array.
- For structs some tagged A2L info must be provided on a by field basis. For example not the same min value applies to all struct fields. Some info applies to the entire struct, for example type. See the examples section for how to enter A2L info which applies to a certain struct field only.
- The table below shows the supported tags and whether they applies to the entire object or a struct/class/union member in case the variable is a compund type. For non-compound types it always applies to the entire object.

Tag	Description	Applicability
a2l	on or off. If it's not "on" the then no A2L entry will be created	Entire object
a2l-type	Measurement or characteristic. Use it to override type derived from memory address or when measurements and characteristics are not placed in fixed memory regions.	Entire object
a2l-description	Description to write in the A2L file.	member
a2l-min	Min value to write in the A2l file. This overrides the min value derived from datatype.	member
a2l-max	Max value to write in the A2l file. This overrides the min value derived from datatype.	member
a2l-coeffs	The created COMPU_METHOD will be of type RAT_FUNC (default is IDENTICAL) and have COEFFS according to tag value.	member
a2l-displayIdentifier	The optional A2L parameter DISPLAY_IDENTIFIER will be added to the output A2L with the identifier being the argument to a2l-displayIdentifier.	member
a2l-format	Format string to write in the generated A2L file.	member
a2l-mute	Do not create A2L objects for the class/struct/union members mentioned in a2l-mute. Tag value is a space-separated list of member identifiers, for example @a2l-mute bUnion.a x. will suppress A2L creation for members bUnion.a and x.	member
a2l-unit	Unit to write in the generated A2L file.	member

Table 1. Creator comment tags

Examples

```
/**
* This first line will be the description in the A2L file.
*
* @a2l on
* @a2l-min 0
* @a2l-max 20
*/
float32 vehicle_length;


// A description of bar.
// @a2l on
// @a2l-type measurement
uint8 bar;


////////////////////////////////////////////////////////////
// This description of foobar will be in the generated A2L
//
// @a2l on
////////////////////////////////////////////////////////////
```

```
float32 foobar;

// A description of this entire struct which is ignored since
// description must be given for struct fields.
//
// @a2l on
// @a2l-type measurement
// @a2l-description fieldX A description of struct field x
// @a2l-max fieldY 123.5
struct AStructType anInstanceName;
```

Limitations

The following scenarios are not supported:

- Static class members (no A2L generated)
- Pointers (no A2L generated)
- Virtual inheritance
- A class inheriting a member with the same name as a member declard in the class iteself.

Integration in automated builds

Integration in an automated build system is straightforward thanks to the command line interface and exit code behavior. Creator will set its exit code to 0 in case of success, 0xFF for license issues and 1 in case of other failures.

Command-line options

-source-path

File or directory to scan for measurements and characteristics. This option can be given multiple times. If the argument is a directory the file exensions .c, .cc, .cpp, .h and .hpp will be scanned. The search for files to scan is done recursively, i.e. also subdirectories will be scanned.

-e

ELF file to use.

-o

Output a2l, i.e. where to write the created a2l.

-measurement-memory-block

Memory block for measurements. Can be given multiple times in case the ECU have measurements in several memory blocks. The argument is a start and a stop address, i.e. 0x080000-0x0F0000. Both hexadecimal and decimal notation is supported.

-characteristic-memory-block

Memory block for characteristics. Can be given multiple times in case the ECU have characteristics in several memory blocks. The argument is a start and a stop address, i.e. 0x080000-0x0F0000. Both hexadecimal and decimal notation is supported.

-mute

Mute output for measurements/characteristics matching one of several wildcard patterns. * matches any character any number of times and ? matches any single character.

-uncondensed-output

The standard output format is to keep /begin, namd and description on the same line. Also the first properties of measurements, characteristics and axispts are kept on the same line. However some broken parsers requires name, description and properties to be on individual lines. With uncondensed output selected

```
/begin CHARACTERISTIC dummy_array "Dummy unsigned array description."
   VAL_BLK 0x0 CREATOR_UWORD 0 COMPU_METHOD_dummy_array 0 100
   MATRIX_DIM 4 1 1
/end CHARACTERISTIC
```

becomes

```
/begin CHARACTERISTIC
   dummy_array
   "Dummy unsigned array description."
   VAL_BLK
   0x0
   CREATOR_UWORD
   0
   COMPU_METHOD_dummy_array
   0
   100
   MATRIX_DIM 4 1 1
/end CHARACTERISTIC
```

-explode-matrices

Create one measurement/characteristic per matrix/array index instead of creating one measurement/characteristic describing the entire matrix/array. This is useful for example when measurement and calibration tools doesn't support matrices.

-skip-address-update

Will not update addresses for newly created A2L objects. This is useful to optimize very large builds which uses Updater.

-utf-8-bom

According to the Unicode standard, the BOM for files encoded in UTF-8 is not recommended. However some programs, for example Vector CANApe, requires it in order to correctly interpret

non-english characters such as Swedish or Chinese. Try adding this option if non-english characters doesn't work properly in a tool which reads a file written by Acam.

-h

Print extensive help including invocation examples and exit.

-v

Print version information and exit.

Command line examples

Creator -source-path C:\Development\MyProject -source-path C:\Development\MyGenericLib\foo.c -e elffile.elf -o out.a2l -measurement-memory-block 0xEF0000-0xFFF000 -measurement-memory-block 10000-12000 -characteristic-memory-block 0x2F0000-0x4FF000 -characteristic-memory-block 0xCFF000-0xEFFE00

Merge

Overview

Merge merges several a2l files into one. One of the files is the master one. The interface definitions, byte ordering etc. from this master a2l will be used in the output file. Interface information, byte ordering etc. in remaining a2l files to merge is ignored.

Redefinitions of items, i.e. the same item is defined in several input files.

If a measurement, characteristic, or axis is defined in several input a2ls then the merge is aborted with an error message. The merge is considered failed.

The action when encountering two groups or functions with identical name depends on the command line options **-merge-functions** and **-merge-groups** respectively. Default is to rename one of the functions/groups during merge to avoid name clash. A warning message is issued in this case. The merge is considered successful. However if the command line option **-merge-functions** is given the the function content will be merged. Pass **-merge-groups** to merge group content.

If a compu method, record layout etc. is defined in several input a2ls then one of them will be renamed to avoid name clashes. A warning message is issued in this case. The merge is considered successful.

If a function is defined in several input a2ls the default action is to rename one of them to avoid name clashes. A warning message is issued in this case. The merge is considered successful. However if the command line option **-merge-functions** was given then function contents will be merged. This merge is considered successful.

Command-line options

-m

Master a2l. Interface information, a2ml etc. will be taken from this a2l. This argument is compulsory.

-i

Additional (except master) a2l files to merge. This argument is compulsory and can be given multiple times to merge several files.

-o

Output a2l, i.e. where to write the merged a2l.

-process-includes

process "/include" statements in input files. Default is to not process includes.

-merge-groups

Merge group contents if a function is defined more than once in the a2ls to merge.

-merge-function

Merge function contents if a function is defined more than once in the a2ls to merge.

-Wno-rename

Do not issue warning when renaming items to avoid name clash.

-Wno-defined-twice

Default is to issue a warning if a MEASUREMENT/CHARACTERISTIC/AXIS_PTS is defined twice with identical definition. This option silences this warning.

-Wwarning=variable-redefined

Default is to abort with error if a MEASUREMENT/CHARACTERISTIC/AXIS_PTS is redefined, i.e. two definitions with different properties such as for example datatype is found in the input. This option turns the error into a warning. Note that the redefined variable will not be in the output A2L because different definitions is an indication of an error which might have serious consequences. For example a too large datatype of a characteristic could cause memory overwrite.

-uncondensed-output

The standard output format is to keep /begin, nam and description on the same line. Also the first properties of measurements, characteristics and axispts are kept on the same line. However some broken parsers requires name, description and properties to be on individual lines. With uncondensed output selected

```
/begin CHARACTERISTIC dummy_array "Dummy unsigned array description."
   VAL_BLK 0x0 CREATOR_UWORD 0 COMPU_METHOD_dummy_array 0 100
   MATRIX_DIM 4 1 1
/end CHARACTERISTIC
```

becomes

```
/begin CHARACTERISTIC
   dummy_array
   "Dummy unsigned array description."
   VAL_BLK
   0x0
   CREATOR_UWORD
   0
   COMPU_METHOD_dummy_array
   0
   100
   MATRIX_DIM 4 1 1
/end CHARACTERISTIC
```

-utf-8-bom

According to the Unicode standard, the BOM for files encoded in UTF-8 is not recommended. However some programs, for example Vector CANApe, requires it in order to correctly interpret

non-english characters such as Swedish or Chinese. Try adding this option if non-english characters doesn't work properly in a tool which reads a file written by Acam.

-h

Print extensive help including invocation examples and exit.

-v

Print version information and exit.

Command line examples

Merge -m master.a2l -i a2lfile1.a2l -i 2lfile2.a2l -o out.a2l

Merge -m master.a2l -i a2lfile1.a2l -i 2lfile2.a2l -o out.a2l -merge-functions

Merge -m master.a2l -i a2lfile1.a2l -i 2lfile2.a2l -o out.a2l -merge-groups

Merge -m master.a2l -i a2lfile1.a2l -i 2lfile2.a2l -o out.a2l -merge-functions -merge-groups

Integration in automated builds

Integration in an automated build system is straightforward thanks to the command line interface and exit code behavior. Merge will set its exit code to 0 in case of success, 0xFF for license issues and 1 in case of other failures.

Updater

Overview

Updater updates the addresses of measurements and characteristics from addresses found in the symbol table of and ELF file and DWARF debugging information from the same ELF file. The debug information is needed to calculate array index offsets and the address of struct fields. The following expressions are supported:

- foo - Plain variable or constant
- foo[x] - An item in an array
- foo.bar - Struct field
- foo[x].bar[y].foo - Any combination of array indexes and struct fields

Command-line options

-i

A2l which addresses shall be updated.

-o

Output a2l, i.e. where to write the merged a2l.

-e

ELF file to take address information from.

-set-readonly-memory-block

Updater can optionally set all characteristics and axispts in a memory are to readonly. This feature is useful for example when safety-related characteristics shall be protected.

-uncondensed-output

The standard output format is to keep /begin, namd and description on the same line. Also the first properties of measurements, characteristics and axispts are kept on the same line. However some broken parsers requires name, description and properties to be on individual lines. With uncondensed output selected

```
/begin CHARACTERISTIC dummy_array "Dummy unsigned array description."
   VAL_BLK 0x0 CREATOR_UWORD 0 COMPU_METHOD_dummy_array 0 100
   MATRIX_DIM 4 1 1
/end CHARACTERISTIC
```

becomes

```
/begin CHARACTERISTIC
```

```
    dummy_array
    "Dummy unsigned array description."
    VAL_BLK
    0x0
    CREATOR_UWORD
    0
    COMPU_METHOD_dummy_array
    0
    100
    MATRIX_DIM 4 1 1
/end CHARACTERISTIC
```

-utf-8-bom

According to the Unicode standard, the BOM for files encoded in UTF-8 is not recommended. However some programs, for example Vector CANApe, requires it in order to correctly interpret non-english characters such as Swedish or Chinese. Try adding this option if non-english characters doesn't work properly in a tool which reads a file written by Acam.

-h

Print extensive help including invocation examples and exit.

-v

Print version information and exit.

Command line examples

Updater -i a2lfile.a2l -e elffile.elf -o out.a2l

Integration in automated builds

Integration in an automated build system is straightforward thanks to the command line interface and exit code behavior. Updater will set its exit code to 0 in case of success, 0xFF for license issues and 1 in case of other failures.

Filter

Overview

Filter filters an a2l file, keeping or deleting measurements, characteristics, groups and functions according to a filter definition. The supported actions are:

- Filter measurements and characteristics by name. Wildards (* and ?) are supported.
- Filter measurements and characteristics by group and function membership. Wildards (* and ?) in group and function name are supported.
- Filter by labfile contents. Very powerful for if the measurements and calibrations to keep or delete a kept in tools such as ETAS INCA or Vector CANape. Enable reuse of filter definition across several projects which improves consistency and maintainability.
- Groups can be filtered by name. Wildcards in group name is supported. Choose whether the group members shall be selected.
- Functions can be filtered by name. Wildcards in function name is supported. Choose whether the function members shall selected or not.

Filter definition

The filter definition is a text file with one rule per line. It starts with filter type (keep or delete selected items). Then follows the rules with one rule per line. These rules are processed one by one from top to bottom.

Groups and functions which becomes empty after processing of all rules are deleted, unused compu methods etc. are removed.

The syntax of the filter definition is described in the filter definition which follows.

```
// A single line comment
/* A comment spanning
multiple lines */
// Selected items shall be deleted (other option is KEEP).
FILTER_TYPE DELETE

// Delete measurements and characteristics in the lab file foo.lab.
SELECT ITEMS IN LABFILE "src\com\udokaelectronics\a2l\test\foo.lab"
// Delete measurement named exactly abc
SELECT MEASUREMENT WHICH NAME MATCHES abc
// * is a wildcard which matches any string including an empty one.
SELECT MEASUREMENT WHICH NAME MATCHES Hello*
// ? is a wildcard which matched any single character
SELECT MEASUREMENT WHICH NAME MATCHES xyz?.dev
// Delete all measurements in group Group1.
// Do not delete measurements in subgroups.
SELECT MEASUREMENT IN GROUP NAME MATCHES Group1 EXCLUDE SUBGROUPS
// Delete all measurements in group Group2.
// Do delete measurements in subgroups.
SELECT MEASUREMENT IN GROUP NAME MATCHES Group2 INCLUDE SUBGROUPS
// Wildcards can be used in groups too.
SELECT MEASUREMENT IN GROUP NAME MATCHES Def*ection INCLUDE SUBGROUPS
// Handling of characteristics is identical to measurements
```

```
// except for CHARACTERISTIC in the rule.
SELECT CHARACTERISTIC WHICH NAME MATCHES x.Increment
SELECT CHARACTERISTIC WHICH NAME MATCHES y.*tCount
SELECT CHARACTERISTIC WHICH NAME MATCHES z.?ctive
SELECT CHARACTERISTIC IN GROUP NAME MATCHES GroupX INCLUDE SUBGROUPS
SELECT CHARACTERISTIC IN GROUP NAME MATCHES foo*ection INCLUDE SUBGROUPS
// Delete group Motor_Limit_Check and its subgroups but doesn't
// delete measurements and characteristics in group.
SELECT GROUP WHICH NAME MATCHES Li_Check INCLUDE MEASUREMENTS INCLUDE CHARACTER
// Delete groups which name starts with Generated but keep
// measurements and characteristics in groups.
SELECT GROUP WHICH NAME MATCHES Generated* EXCLUDE MEASUREMENTS EXCLUDE CHARACT
// Delete all measurements in function fooFunction but don't touch
// measurements in subfunctions.
SELECT MEASUREMENT IN FUNCTION NAME MATCHES fooFunction EXCLUDE SUBFUNCTIONS
// Delete all characteristics in function fooFunction but don't touch
// measurements in subfunctions.
SELECT CHARACTERISTIC IN FUNCTION NAME MATCHES fooFunction EXCLUDE SUBFUNCTIONS
// Delete all measurements in function matching foobar?unction and remove
// measurements in subfunctions.
SELECT MEASUREMENT IN FUNCTION NAME MATCHES foobar?unction INCLUDE SUBFUNCTIONS
// Delete function aFunction and its subgroups, doesn't delete
// measurements and keeps characteristics in function.
SELECT FUNCTION WHICH NAME MATCHES aFunction INCLUDE MEASUREMENTS EXCLUDE CHARA
// Delete function functions matching m* and its subgroups,
// keeps measurements and deletes characteristics in function.
SELECT FUNCTION WHICH NAME MATCHES m* EXCLUDE MEASUREMENTS INCLUDE CHARACTERIST
```

Integration in automated builds

Integration in an automated build system is straightforward thanks to the command line interface and exit code behavior. Filter will set its exit code to 0 in case of success, 0xFF for license issues and 1 in case of other failures.

Command-line options

-i

A2l which shall be filtered.

-o

Output a2l, i.e. where to write the merged a2l.

-f

Filter definition file.

-uncondensed-output

The standard output format is to keep /begin, namd and description on the same line. Also the first properties of measurements, characteristics and axispts are kept on the same line. However some broken parsers requires name, description and properties to be on individual lines. With uncondensed output selected

```
/begin CHARACTERISTIC dummy_array "Dummy unsigned array description."
   VAL_BLK 0x0 CREATOR_UWORD 0 COMPU_METHOD_dummy_array 0 100
   MATRIX_DIM 4 1 1
/end CHARACTERISTIC
```

becomes

```
/begin CHARACTERISTIC
   dummy_array
   "Dummy unsigned array description."
   VAL_BLK
   0x0
   CREATOR_UWORD
   0
   COMPU_METHOD_dummy_array
   0
   100
   MATRIX_DIM 4 1 1
/end CHARACTERISTIC
```

-utf-8-bom

According to the Unicode standard, the BOM for files encoded in UTF-8 is not recommended. However some programs, for example Vector CANApe, requires it in order to correctly interpret non-english characters such as Swedish or Chinese. Try adding this option if non-english characters doesn't work properly in a tool which reads a file written by Acam.

-h

Print extensive help including invocation examples and exit.

-v

Print version information and exit.

Command line examples

Filter -i a2lfile.a2l -f filter.txt -o out.a2l

Appendix A. Suppported ASAM elements

Note that this appendix doesn't cover the header with interface data, memory segments etc. The header will not be modifed by Acam.

AXIS_DESCR

Supported axis points types: COM_AXIS, STD_AXIS,FIX_AXIS

Supported optional parameters: AXIS_PTS_REF, BYTE_ORDER, DEPOSIT, EXTENDED_LIMITS, FORMAT, READ_ONLY, SYMBOL_LINK

AXIS_PTS

Supported optional parameters: BYTE_ORDER, DEPOSIT, ECU_ADDRESS_EXTENSION, EXTENDED_LIMITS, FORMAT, IF_DATA, READ_ONLY, SYMBOL_LINK

CHARACTERISTIC

Supported types: CURVE, MAP, VAL_BLK, VALUE, ASCII

Supported optional parameters: AXIS_DESCR, BIT_MASK, BYTE_ORDER, DISPLAY_IDENTIFIER, ECU_ADDRESS_EXTENSION, EXTENDED_LIMITS, FORMAT, IF_DATA, MATRIX_DIM, MAX_REFRESH, NUMBER, READ_ONLY, SYMBOL_LINK

COMPU_METHOD

Supported types: RAT_FUNC, TAB_VERB, IDENTICAL, LINEAR

Supported optional parameters: COEFFS, COEFFS_LINEAR, COMPU_TAB_REF

COMPU_VTAB

Supported optional parameters: DEFAULT_VALUE

COMPU_VTAB_RANGE

Supported optional parameters: DEFAULT_VALUE

FUNCTION

Supported optional parameters: DEF_CHARACTERISTIC, IN_MEASUREMENT, LOC_MEASUREMENT, OUT_MEASUREMENT, REF_CHARACTERISTIC, SUB_FUNCTION

GROUP

Supported optional parameters: FUNCTION_LIST, REF_CHARACTERISTIC, REF_MEASUREMENT, ROOT, SUB_GROUP

MEASUREMENT

Supported types: CURVE, MAP, VAL_BLK, VALUE

Supported optional parameters: ARRAY_SIZE, BIT_MASK, BYTE_ORDER, DISPLAY_IDENTIFIER, ECU_ADDRESS_EXTENSION, EXTENDED_LIMITS, FIX_AXIS_PAR_DIST, FORMAT, IF_DATA, LAYOUT, MATRIX_DIM, MAX_REFRESH, READ_WRITE, SYMBOL_LINK

RECORD_LAYOUT

Supported optional parameters: ALIGNMENT_BYTE, ALIGNMENT_FLOAT32_IEEE, ALIGNMENT_FLOAT64_IEEE, ALIGNMENT_INT64, ALIGNMENT_LONG, ALIGNMENT_WORD, AXIS_PTS_X/_Y, FNC_VALUES, NO_AXIS_PTS_X/_Y