UDOKA

Acam 1.7p6
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 Canapé, 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.

- Modifier – The swiss army knife. A collection of actions which doesn't fit naturally in the other tools.

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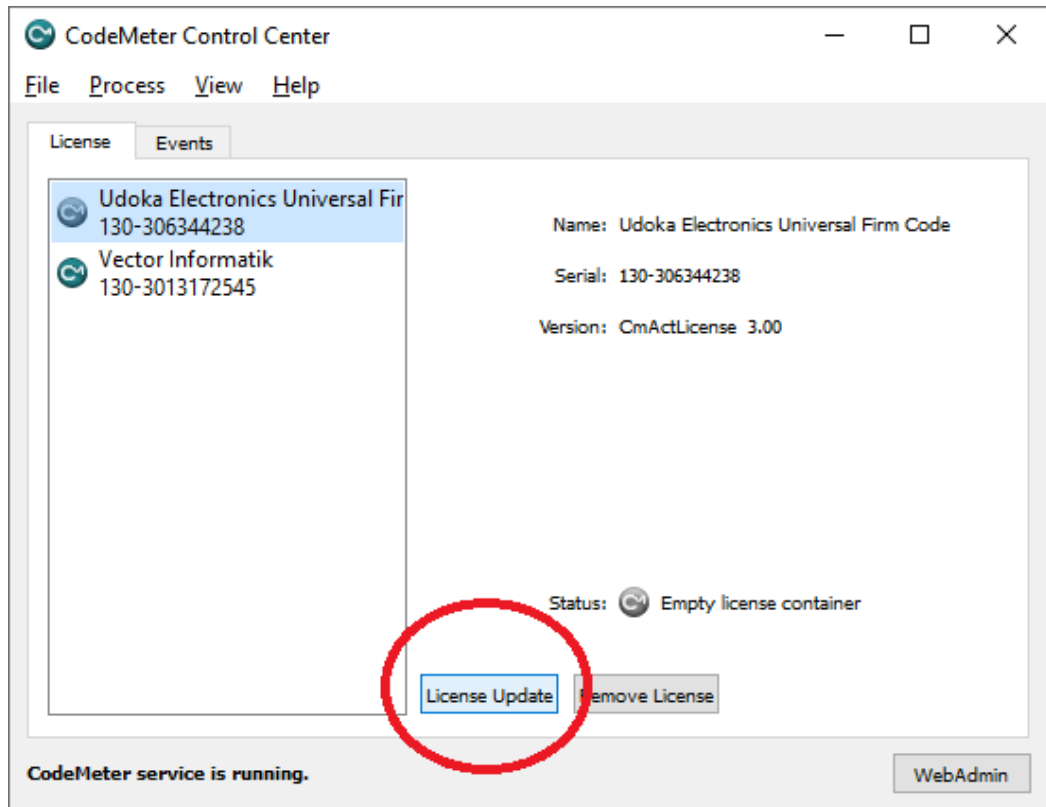
4. A wizard will be started. Use the wizard to create a license request. Send the exported license request file to sales@udokaelectronics.com.

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.

Exit status

Acam tools have exit status zero on success and non-zero on failure. Generally an issue which is classified as an error results in immediate abort and exit with non-zero error status. When an issue classified as a warning is encountered processing will continue and if there are no errors exit status will be zero. For some issues the classification (warning or error) can be controlled by command-line options. See the documentation for each individual tool for details.

Creator

Overview

Creator generates A2L files for both 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 applications instead of writing and maintaining A2L information. The result is fast development and efficient software development.

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

Capabilities and syntax

- 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 the only tag which is mandatory is the "a2l on".

- Both // and /**/ is supported.

- The first line with an alphanumeric character in the comment block will be interpreted as the description of the variable/type. It will be written as the description in the generated A2L file.

- A2l info can be given per instance and per type. Information in tags for an instance takes precedence over information in tags for the type.

- A2l information specified for base classes are inherited by derived classes.

- C++ templates can be used. The A2L information will be the same for every instantiation of the template.

- Individual A2L data for different elements in an array is not support. Each element in an array will get the same properties.

- For compound types (struct, class and union) and instances thereof it is possible to specify a2l properties for class members and struct fields by writing the member name after the a2l tag. For example to turn off a2l generation for struct field x write "a2l x off" Class members and struct fields inherits some A2L properties from the parent object.

- A2L properties on/off, type (measurement or characteristic), max_refresh, read_only and read_write will be used for all members of compound types if they are set for set for the instance or type.

- Note that if memory area boundaries are provided on the command line then Acam Creator can determine whether an item is a measurement or characteristic. This can be overridden by tags in the comments.

The table below shows the supported tags.

Tag	Description
a2l	on or off. If it's not "on" the then no A2L entry will be created
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.
a2l-characteristic-type	Applicable to CHARACTERISTIC only. Use it to override type, i.e. to force ASCII instead of VAL_BLK.
a2l-description	Description to write in the A2L file.
a2l-min	Min value to write in the A2l file. This overrides the min value derived from datatype.
a2l-max	Max value to write in the A2l file. This overrides the min value derived from datatype.
a2l-linearCoeffs	The created COMPU_METHOD will be of type LINEAR (default is IDENTICAL) and have COEFFS according to tag value.
a2l-ratFuncCoeffs	The created COMPU_METHOD will be of type RAT_FUNC (default is IDENTICAL) and have COEFFS according to tag value.
a2l-displayIdentifier	The optional A2L parameter DISPLAY_IDENTIFIER will be added to the output A2L with the identifier being the argument to a2l-displayIdentifier.
a2l-format	Format string to write in the generated A2L file.
a2l-max_refresh	The created A2L object will have MAX_REFRESH with the parameters passed to @a2l-max_refresh.
a2l-read_only	Marks the created CHARACTERISTIC or AXIS_PTS as READ_ONLY.
a2l-read_write	Marks the created MEASUREMENT as READ_WRITE.
a2l-unit	Unit to write in the generated A2L file.

Table 1. Creator comment tags

Examples

Scalar instances

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


//////////////////////////////////////////////////////
// This description of foobar will be in the generated A2L
//
// @a2l on
//////////////////////////////////////////////////////
float32 foobar;

// A description of bar which shall be or type measurement.
// @a2l on
// @a2l-type measurement
```

```
uint8 bar;
```

Struct and class instances

```
// A description of this entire struct which is ignored by
// Acam Creator 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;
```

Scalar types

```
// Description of this type
// @a2l on
// @a2l-type measurement
// @a2l-min 0
typedef double myPositiveType;
```

Every instance of myPositiveType will be an A2L measurement with min value 0. This applies also of myPositiveType is used in structs and classes.

Enums

```
// @a2l on
// @a2l-type measurement
enum Ape {measGorilla, measChimpanzee, measOrangutan};

// @a2l on
// @a2l-type measurement
enum class Ape {measGorilla, measChimpanzee, measOrangutan};

// @a2l on
// @a2l-type measurement
typedef enum {measGorilla, measChimpanzee, measOrangutan} Ape;
```

Structs and classes

A2L properties applying for all members can be provided in a comment block before the class/struct declaration. One can can also provide information for members here as is done with x. Tags for members can also be written right before they are declared as is the case with y.

```
// @a2l on
// @a2l-type measurement
// @a2l-description x x is a nice variable
// @a2l-min x 0
class AClass
```

```
{
 public:
 int x;
 // @a2l-min 3
 // @a2l-description Hello from y
 int y;
};
```

Typedef struct is also supported, see example below.

```
// @a2l on
// @a2l-type measurement
// @a2l-description x x is a nice variable
// @a2l-min x 0
typedef struct
{
 int x;
 // @a2l-min 3
 // @a2l-description Hello from y
 int y;
} AStruct;
```

A2L information for types will be applied both when the type is used stand-alone and when it is used in a class or struct. In the following example there will be a MEASUREMENT generated for aClassInstance.p.x but a CHARACTERISTIC for aClassInstance.p.y since the default from Point is overridden at the instance declaration.

```
// @a2l on
// @a2l-type measurement
struct
{
 int x;
 int y;
} Point;


class AClass
{
 public:
   Point p;
};

// @a2l-type p.y characteristic
AClass aClassInstance;
```

Limitations

The following scenarios are not supported:

• Static class members (no A2L generated)

• Pointers (no A2L generated)

• Virtual inheritance (aka "The diamond problem")

- A class inheriting a member with the same name as a member declared in the class itself.

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 extensions .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, name 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 Canapé, 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 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 can be given multiple times to merge several files. Note that this argument isn't compulsory - one might want to do only preprocessing for /include.

-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.

-remove-block-comments-in-included-files-start

Block comments in included files can be removed. Comments to remove are defined by a start and a stop Perl regular expression. These expressions are matched against one line at at time. For example, to remove the block comment

```
/*****
* header line 1
* header line 2
**/
```

the start expression is /** and stop expression is **/.

-remove-block-comments-in-included-files-stop

See -remove-block-comments-in-included-files-start above

-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, name 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 Canapé, 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

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 modified a2l.

-e

ELF file to take address information from.

-process-includes

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

-uncondensed-output

The standard output format is to keep /begin, name 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 Canapé, 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

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. Wildcards (* and ?) are supported.

- Filter measurements and characteristics by group and function membership. Wildcards (* 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 Canapé. 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

// Select measurements and characteristics in the lab file foo.lab.
SELECT ITEMS IN LABFILE "src\com\udokaelectronics\a2l\test\foo.lab"
// Select 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
// Select all measurements in group Group1.
// Do not select measurements in subgroups.
SELECT MEASUREMENT IN GROUP NAME MATCHES Group1 EXCLUDE SUBGROUPS
// Select all measurements in group Group2.
// Do select 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
            // Select group Motor_Limit_Check and its subgroups but doesn't
            // select measurements and characteristics in group.
            SELECT GROUP WHICH NAME MATCHES Li_Check INCLUDE SUBGROUPS INCLUDE MEASUREMENTS
            // Select groups which name starts with Generated but keep
            // measurements and characteristics in groups.
            SELECT GROUP WHICH NAME MATCHES Generated* INCLUDE SUBGROUPS EXCLUDE MEASUREMENT
            // Select all measurements in function fooFunction but don't touch
            // measurements in subfunctions.
            SELECT MEASUREMENT IN FUNCTION NAME MATCHES fooFunction EXCLUDE SUBFUNCTIONS
            // Select all characteristics in function fooFunction but don't touch
            // measurements in subfunctions.
            SELECT CHARACTERISTIC IN FUNCTION NAME MATCHES fooFunction EXCLUDE SUBFUNCTIONS
            // Select all measurements in function matching foobar?unction and include
            // measurements in subfunctions in selection.
            SELECT MEASUREMENT IN FUNCTION NAME MATCHES foobar?unction INCLUDE SUBFUNCTIONS
            // Select function aFunction and its subgroups, selects
            // measurements and doesn't select characteristics in function.
            SELECT FUNCTION WHICH NAME MATCHES aFunction INCLUDE MEASUREMENTS EXCLUDE CHARA
            // Select function functions matching m* and its subgroups,
            // doesn't select measurements and selects characteristics in function.
            SELECT FUNCTION WHICH NAME MATCHES m* EXCLUDE MEASUREMENTS INCLUDE CHARACTERIST
```

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, name 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 Canapé, 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

::: UDOKA

Modifier

Overview

Modifier performs various actions on an A2L file. It's a collection of actions which doesn't fit naturally in the other tools.

Actions definition

The syntax of the actions definition is described in the exampe action definition which follows.

```
// A single line comment
/* A comment spanning
   multiple lines */
/* The actions are processed one by one.*/
SET READONLY WHERE ADDRESS IN 0x80004000-0x800043ff // SET READONLY supports bo

REMOVE SYMBOL_LINK WHERE NAME MATCHES ParameterManager.* // REMOVE SYMBOL_LINK

RENAME Chimp TO AnonymousMonkey

OFFSET ADDRESS BY 1 WHERE NAME MATCHES *rtDW_CtApPressurePreCondTLPb* // OFFSET_
OFFSET ADDRESS BY -1 WHERE NAME MATCHES *x*

REMOVE EMPTY GROUPS
REMOVE EMPTY FUNCTIONS

REMOVE COMPU_VTAB WHERE NAME MATCHES *Regen*
REMOVE COMPU_VTAB_RANGE WHERE NAME MATCHES *StateCoord*

RENAME COMPU_METHOD CompuMethod1 TO CompuMethod2
```

The following commands supports both address range and name matching object selection: SET READONLY, REMOVE SYMBOL_LINK, OFFSET ADDRESS.

Command-line options

-i

A2l which shall be modifired.

-o

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

-a

Actions definition file.

-uncondensed-output

The standard output format is to keep /begin, name 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 Canapé, 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

Modifier -i a2lfile.a2l -a actions.txt -o out.a2l

Appendix A. Supported ASAM elements

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

AXIS_DESCR

Supported axis points types: COM_AXIS, STD_AXIS,FIX_AXIS

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

AXIS_PTS

Supported optional parameters: ANNOTATION, BYTE_ORDER, DISPLAY_IDENTIFIER, 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: ANNOTATION, 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: ANNOTATION, FUNCTION_VERSION, DEF_CHARACTERISTIC, IN_MEASUREMENT, LOC_MEASUREMENT, OUT_MEASUREMENT, REF_CHARACTERISTIC, SUB_FUNCTION

GROUP

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

MEASUREMENT

Supported types: CURVE, MAP, VAL_BLK, VALUE

Supported optional parameters: ANNOTATION, 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