UDOKA ELECTRONICS

UeA2lTools 1.3.p11
User Guideline

UeA2lTools User Guideline

Table of Contents

List of Tables

Introduction

UeA2lTools is a suite of four independent tools:

- UeA2lCreator - UeA2lCreator 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.

- UeA2lMerge - UeA2lMerge 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. UeA2lMerge accomplishes this task smoothly.

- UeA2lAddressUpdate - 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. UeAl2AddressUpdate performs this task.

- UeA2lFilter - 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. UeA2lFilter 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 to run. The delivered zip archive contains one 32-bit Windows executable per tool. It also contains a DLL, antlr4-runtime.dll. This DLL must be in the same folder as the UeA2lTools executables or in a folder in the path. We recommend just unpacking the delivery zip and keep the content as it is since that will keep the executables, this manual and the Software License Agreement etc. together.

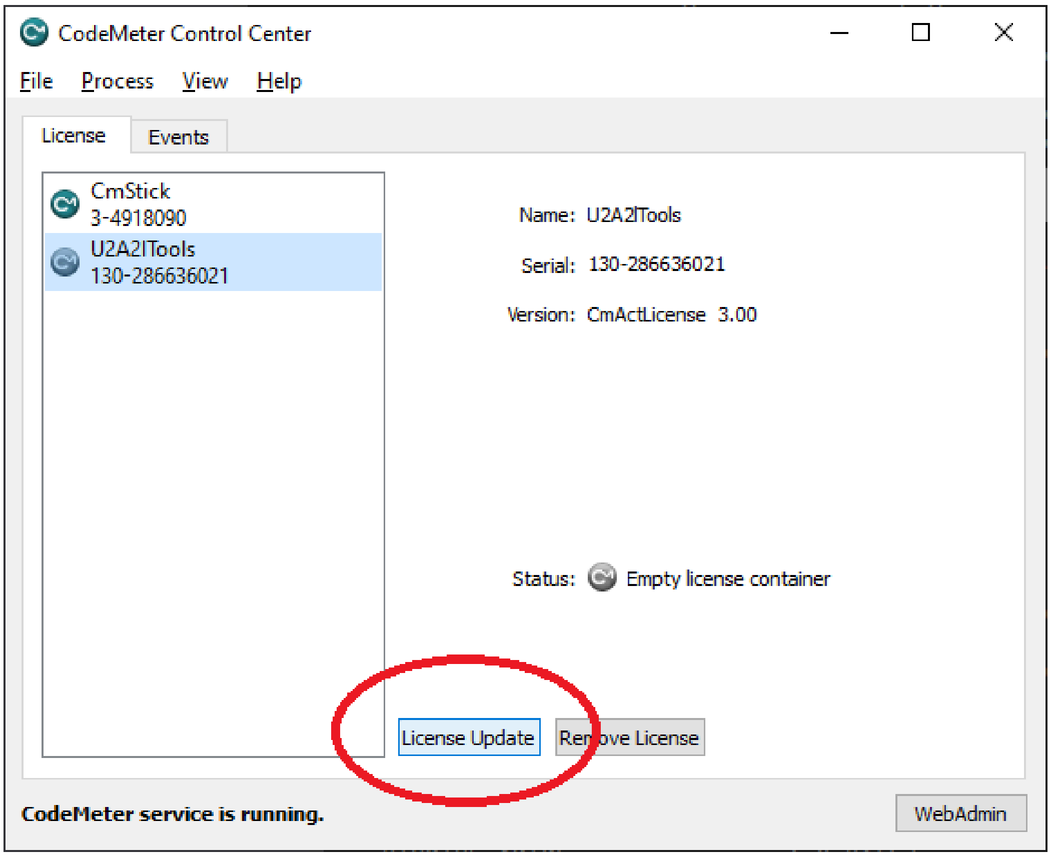
License management

UeA2lTools uses CodeMeter technology for license management. A CodeMeter runtime is required to use UeA2lTools. It can be downloaded for free from www.udokaelectronics.com or codemeter.com. Install CodeMeter Runtime on the computer on which UeA2lTools will be executed and then proceed according to the instructions below according to the chosen license model.

Node-locked and 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. Start CodeMeter Control Center.
2. Choose *File->Import License...* and select UdokaElectronics-Evaluation.WibuCmLIF or UdokaElectronis-Perpetual.WibuCmLIF (both are included in UeA2lTools delivery package).
3. Select the CmActLicense imported in previous step and click *License Update*. See screenshot below for example.



Name: U2A2lTools
Serial: 130-286636021
Version: CmActLicense 3.00
Status: Empty license container
CodeMeter service is running.

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. **UeA2lMerge.exe -h.**

A2L file encoding

Both ANSI and UTF-8 is supported for input A2L files. The encoding will be detected by UeA2lTools, i.e. there is no command line option for encoding. Output A2L files will be encoded in UTF-8 (default) or UTF-8 BOM.

UeA2lCreator

Overview

UeA2lCreator 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 UeA2lCreator 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 UeA2lCreator 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 field in case the variable is a struct.

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.	Struct field
a2l-min	Min value to write in the A2l file. This overrides the min value derived from datatype.	Struct field
a2l-max	Max value to write in the A2l file. This overrides the min value derived from datatype.	Struct field
a2l-unit	Unit to write in the generated A2L file.	Struct field
a2l-format	Format string to write in the generated A2L file.	Struct field
a2l-coeffs	The created COMPU_METHOD will be of type RAT_FUNC (default is IDENTICAL) and have COEFFS according to tag value.	Struct field

Table 1. UeA2lCreator 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;
```

Integration in automated builds

Integration in an automated build system is straightforward thanks to the command line interface and exit code behavior. UeA2lCreator will set its exit code to 0 in case of success and 1 in case of failure. Ideal for integrating in a build driven by make.

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.

-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 UEA2LCREATOR_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
   UEA2LCREATOR_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 UeA2lTools.

-h

Print extensive help including invocation examples and exit.

-v

Print version information and exit.

Command line examples

UeA2lCreator -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

UeA2lMerge

Overview

UeA2lMerge 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.

-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.

-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 UEA2LCREATOR_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
   UEA2LCREATOR_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 UeA2lTools.

-h

Print extensive help including invocation examples and exit.

-v

Print version information and exit.

Command line examples

UeA2lMerge -m master.a2l -i a2lfile1.a2l -i 2lfile2.a2l -o out.a2l

UeA2lMerge -m master.a2l -i a2lfile1.a2l -i 2lfile2.a2l -o out.a2l -merge-functions

UeA2lMerge -m master.a2l -i a2lfile1.a2l -i 2lfile2.a2l -o out.a2l -merge-groups

```
UeA2lMerge -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. UeA2lMerge will set its exit code to 0 in case of success and 1 in case of failure. Ideal for integrating in a build driven by make.

UeA2lAddressUpdate

Overview

UeA2lAddressUpdate 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.

-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 UEA2LCREATOR_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
   UEA2LCREATOR_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 UeA2lTools.

-h

Print extensive help including invocation examples and exit.

-v

Print version information and exit.

Command line examples

UeA2lAddressUpdate -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. UeA2lAddressUpdate will set its exit code to 0 in case of success and 1 in case of failure. Ideal for integrating in a build driven by make.

UeA2lFilter

Overview

UeA2lFilter 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 deleted or kept, i.e. it is possible to remove a group but keep measurements and characteristics in the group. It is also possible to delete measurements but keep characteristics and the other way around. A group rule is always applied to subgroups as well. Useful to preserve software structure confidentiality.
- Functions can be filtered by name. Wildcards in function name is supported. Choose whether the function members shall be deleted or kept, i.e. it is possible to remove a function but keep measurements and characteristics in the group. It is also possible to delete measurements but keep characteristics and the other way around. A function rule is always applied to subfunctions as well. Useful to preserve software structure confidentiality.

Filter definition

The filter definition is a text file with one rule per line. It starts with default actions (taken when no rule matches a measurement/characteristic/group/function). Then follows the rules with one rule per line. These rules are processed one by one from top to bottom. The first rule which applies to a measurement/characteristic/group/function have precedence, i.e. if the first rule states that measurement foo shall be deleted but the second states that it shall be kept then it will be deleted.

Groups and functions which becomes empty after processing of all rules are deleted.

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

```
// A single line comment
/* A comment spanning
multiple lines */

// Default action for measurements and characteristics which no
// rule applies to.
DEFAULT KEEP MEASUREMENT
DEFAULT KEEP CHARACTERISTIC
DEFAULT KEEP GROUP KEEP MEASUREMENTS KEEP CHARACTERISTICS
// Delete measurements and characteristics in the lab file foo.lab.
DELETE ITEMS IN LABFILE "src\com\udokaelectronics\a2l\test\foo.lab"
// Delete measurement named exactly abc
DELETE MEASUREMENT WHICH NAME MATCHES abc
// * is a wildcard which matches any string including an empty one.
DELETE MEASUREMENT WHICH NAME MATCHES Hello*
```

```
// ? is a wildcard which matched any single character
DELETE MEASUREMENT WHICH NAME MATCHES xyz?.dev
// Delete all measurements in group Group1.
// Do not delete measurements in subgroups.
DELETE MEASUREMENT IN GROUP NAME MATCHES Group1 EXCLUDE SUBGROUPS
// Delete all measurements in group Group2.
// Do delete measurements in subgroups.
DELETE MEASUREMENT IN GROUP NAME MATCHES Group2 INCLUDE SUBGROUPS
// Wildcards can be used in groups too.
DELETE MEASUREMENT IN GROUP NAME MATCHES Def*ection INCLUDE SUBGROUPS
// Handling of characteristics is identical to measurements
// except for CHARACTERISTIC in the rule.
DELETE CHARACTERISTIC WHICH NAME MATCHES x.Increment
DELETE CHARACTERISTIC WHICH NAME MATCHES y.*tCount
DELETE CHARACTERISTIC WHICH NAME MATCHES z.?ctive
DELETE CHARACTERISTIC IN GROUP NAME MATCHES GroupX INCLUDE SUBGROUPS
DELETE 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.
DELETE GROUP WHICH NAME MATCHES Li_Check DELETE MEASUREMENTS DELETE CHARACTERIS
// Delete groups which name starts with Generated but keep
// measurements and characteristics in groups.
DELETE GROUP WHICH NAME MATCHES Generated* KEEP MEASUREMENTS KEEP CHARACTERISTI
// Delete all measurements in function fooFunction but don't touch
// measurements in subfunctions.
DELETE MEASUREMENT IN FUNCTION NAME MATCHES fooFunction EXCLUDE SUBFUNCTIONS
// Delete all characteristics in function fooFunction but don't touch
// measurements in subfunctions.
DELETE CHARACTERISTIC IN FUNCTION NAME MATCHES fooFunction EXCLUDE SUBFUNCTIONS
// Delete all measurements in function matching foobar?unction and remove
// measurements in subfunctions.
DELETE MEASUREMENT IN FUNCTION NAME MATCHES foobar?unction INCLUDE SUBFUNCTIONS
// Delete function aFunction and its subgroups, doesn't delete
// measurements and keeps characteristics in function.
DELETE FUNCTION WHICH NAME MATCHES aFunction DELETE MEASUREMENTS KEEP CHARACTER
// Delete function functions matching m* and its subgroups,
// keeps measurements and deletes characteristics in function.
DELETE FUNCTION WHICH NAME MATCHES m* KEEP MEASUREMENTS DELETE CHARACTERISTICS
```

Integration in automated builds

Integration in an automated build system is straightforward thanks to the command line interface and exit code behavior. UeA2lFilter will set its exit code to 0 in case of success and 1 in case of failure. Ideal for integrating in a build driven by make.

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 UEA2LCREATOR_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
   UEA2LCREATOR_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 UeA2lTools.

-h

Print extensive help including invocation examples and exit.

-v

Print version information and exit.

Command line examples

UeA2lFilter -i a2lfile.a2l -f filter.txt -o out.a2l