FAQ Software

MOTIX™ Embedded Power ICs based on Arm® Cortex®-M

About this document

Scope and purpose

This document is intended to answer frequently asked questions regarding software topics in the context of programming MOTIX™ Embedded Power devices and related tools.

Intended audience

Software engineers, embedded power designers, application engineers

Table of contents

1 Keil uVision Topics

1.1 Code completion

By default, Keil uVision offers an auto-completion feature. Sometimes the feature is not visible, e.g. when the cursor is changed to another position and then put back. It appears again by placing the cursor to the position to auto-complete and then by clicking **Ctrl + Space**. Then an alphabetical list of suggestions is shown.

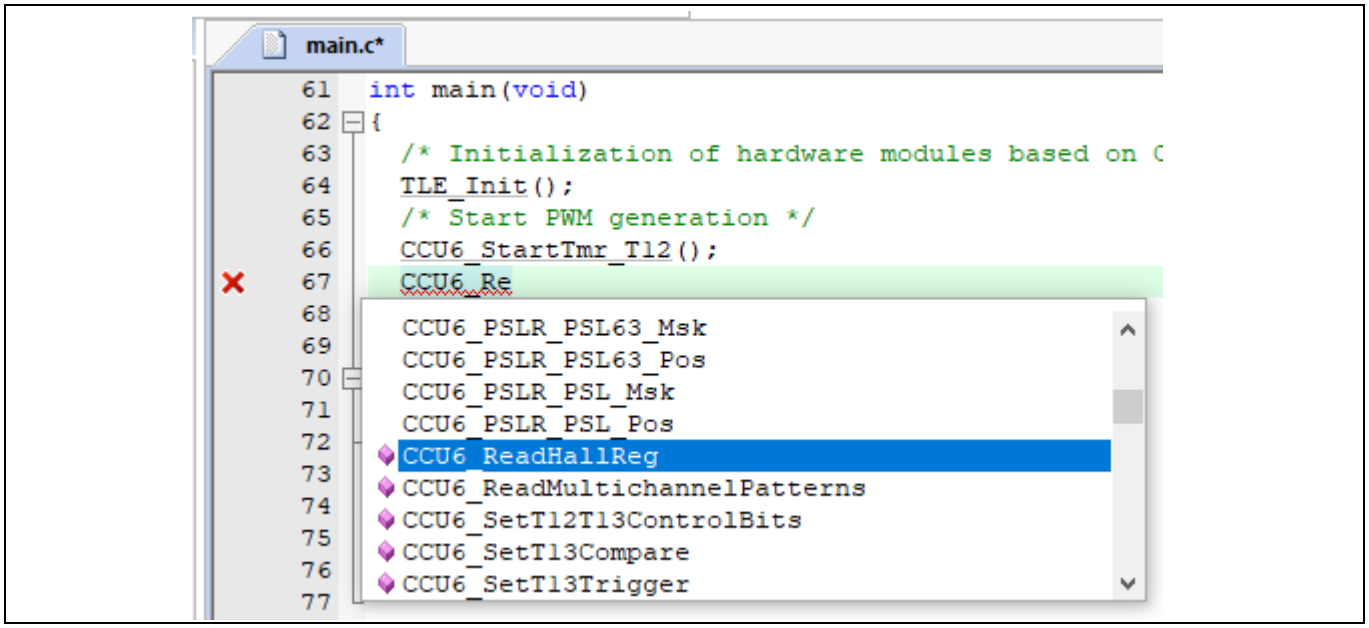

Figure 1 Auto-completion in Keil uVision

When typing, the suggestions change based on the current text. They can be refined by adding more letters to specify the function or variable as well as extended by removing text.

1.2 Help with F1

We offer a help for the functions and types used in the SDK. This help can be accesses by placing the cursor into the name or marking it and then clicking F1.

```
/* Start PWM generation */
CCU6_StartTmr_T12();
```

Figure 2 Placing the cursor in a function name

After clicking F1, the web browser opens and displays the help for the selected function/type.
For functions, the following items are described:

- Short description
- Parameters
- Return value
- Optional: Short example where the function is used

For types or macros, the following items are described:

- Description
- Further details if applicable

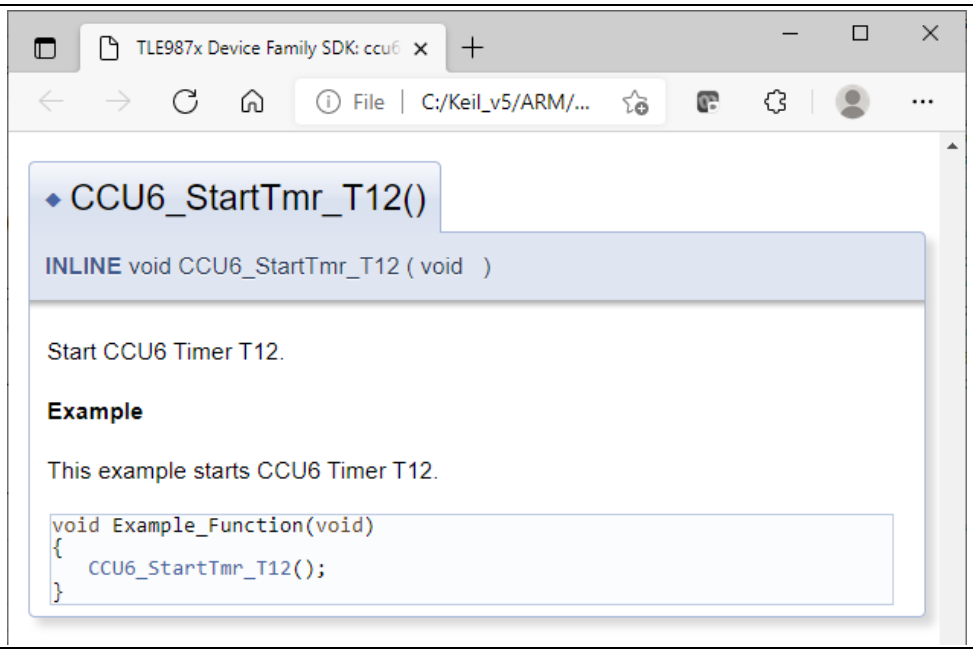

Figure 3 F1 Help for the function CCU6_StartTmr_T12()

Within the help, it is also possible to browse and access the help for other functions.

1.3 SDK Help and general documentation

You can access the SDK help and the general documentation in the section ***Books*** of your project.

The ***SDK Help*** is intended to give you an overview of the release notes (from the current and older pack versions), as well as some tips regarding the creation of a project from scratch or the change of the target device for example.

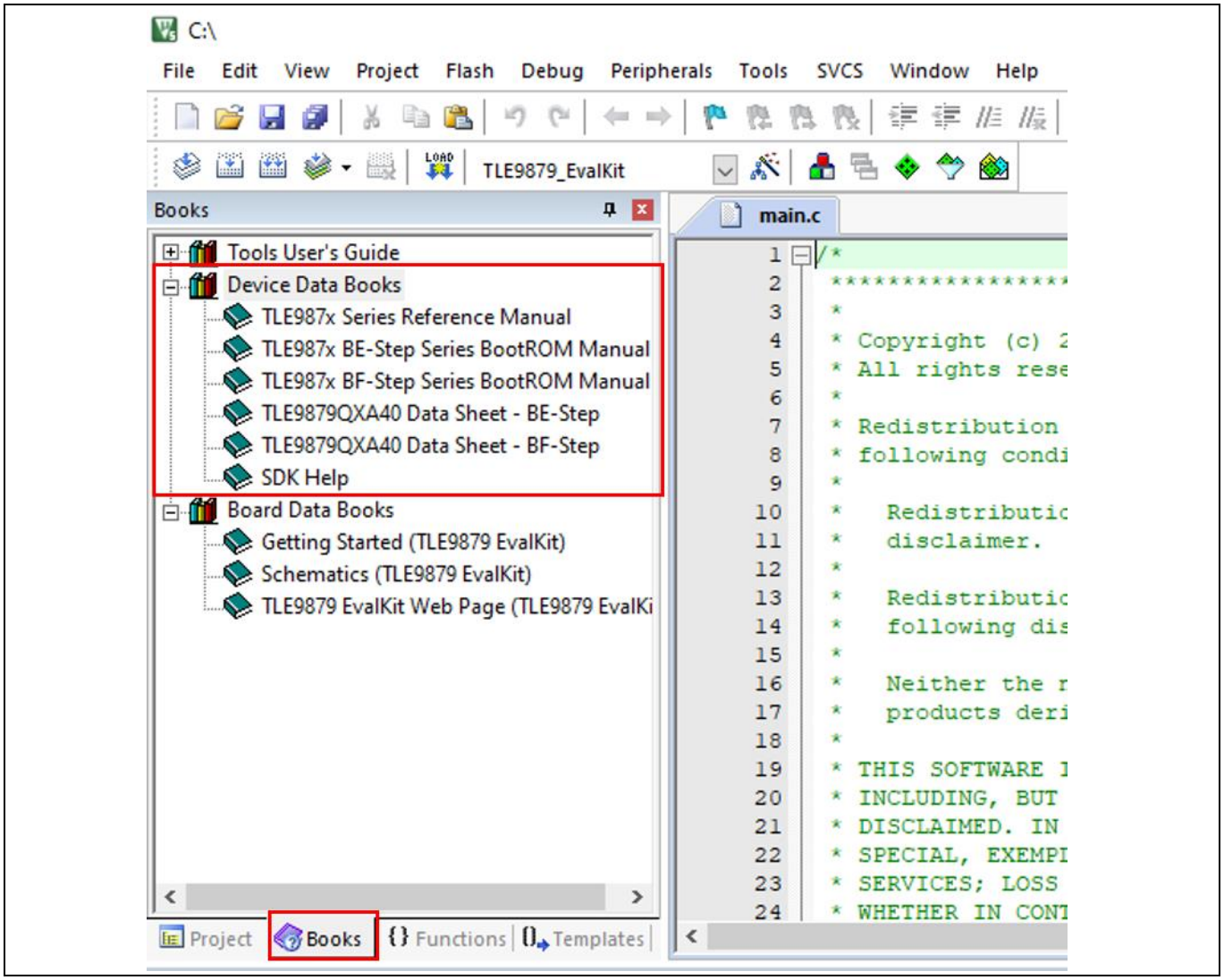

Figure 4 General documentation in the tab Books in Keil uVision

1.4 How to use a fixed pack version

Per default, a project runs with the latest installed pack of the used chip. You still have the possibility to use an older fixed pack.

To do so, click on the icon , which opens the ***Manage Run-Time Environment*** window. To change the pack version, select ***Select Packs*** at the bottom.

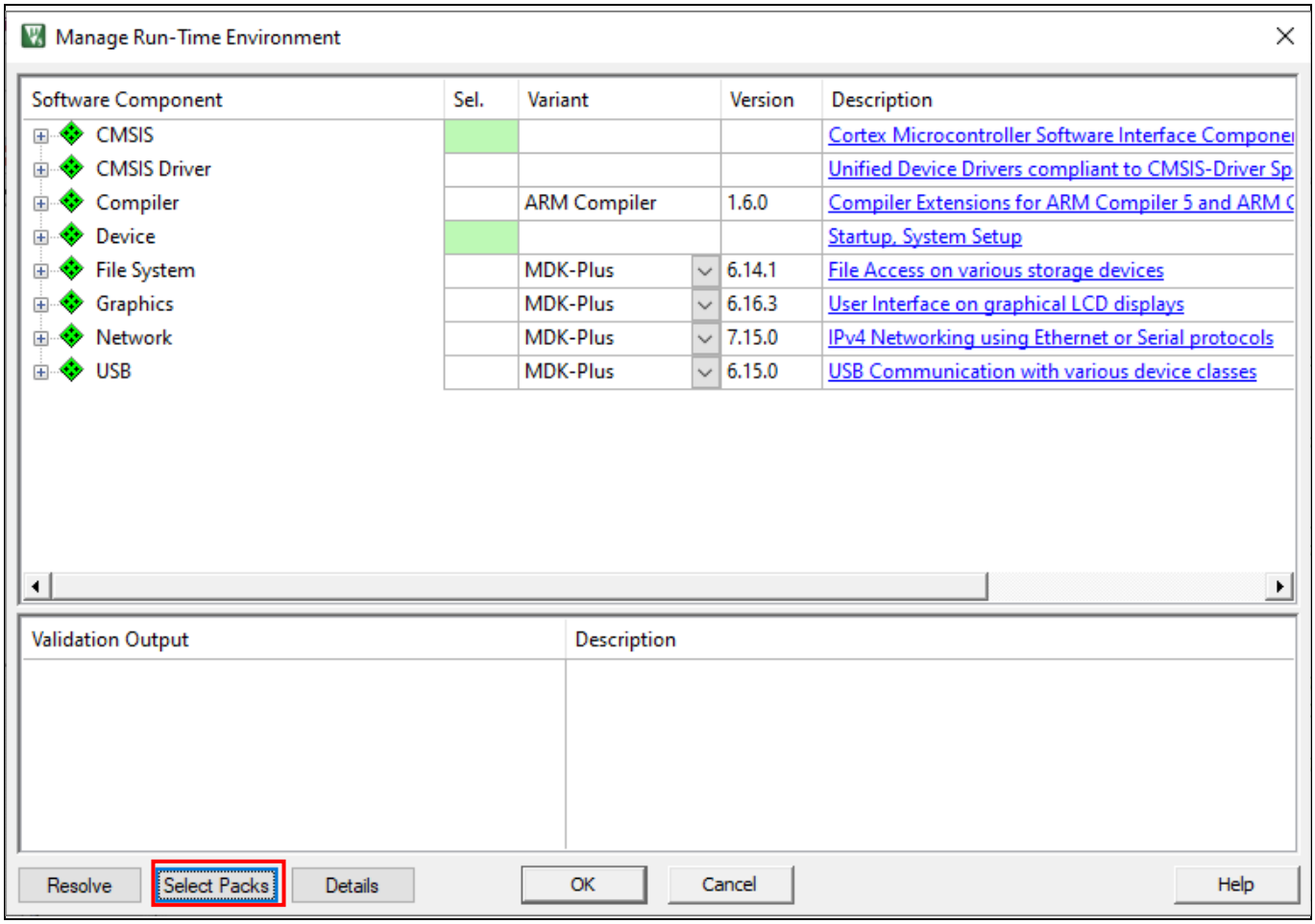

Figure 5 Window Manage Run-Time Environment in Keil uVision

The window ***Select Software Packs for Target*** opens. As mentioned above, the checkbox to use the latest version of the installed pack is checked per default and the name and version of the used pack are greyed out.

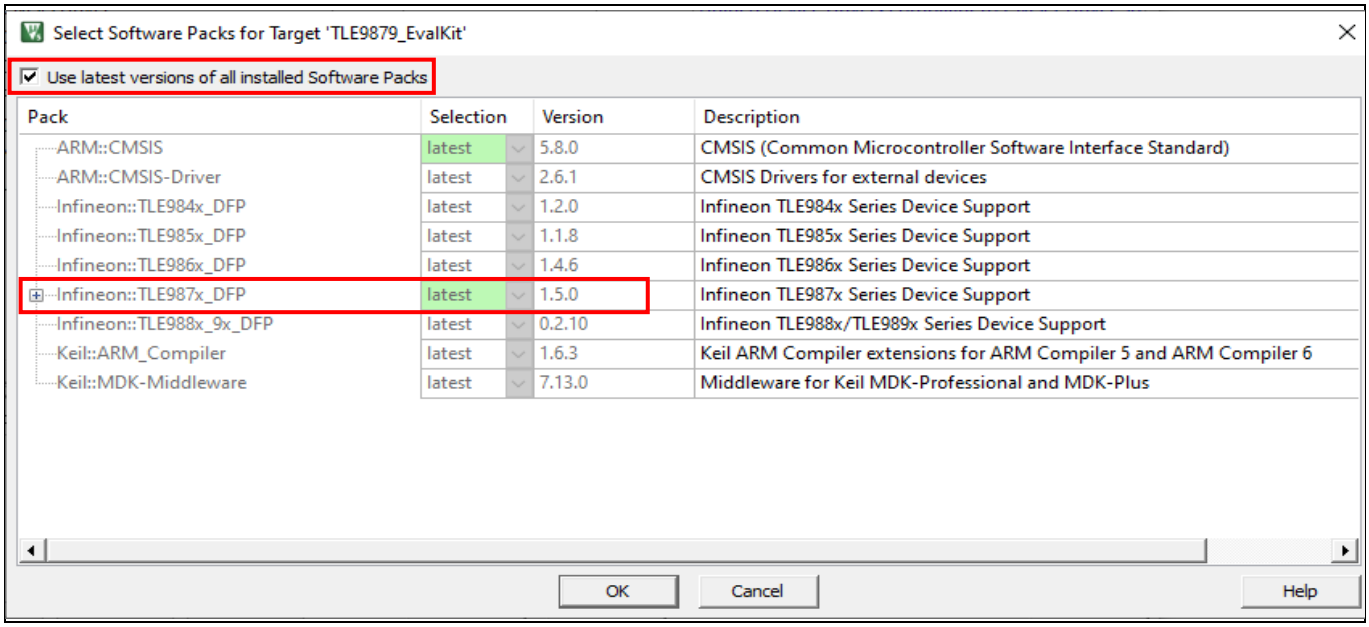

Figure 6 Window Select Software Packs for Target in Keil uVision

To use a different pack, uncheck the checkbox on the top and select the pack version that you want to work with. Click ***OK*** to exit.

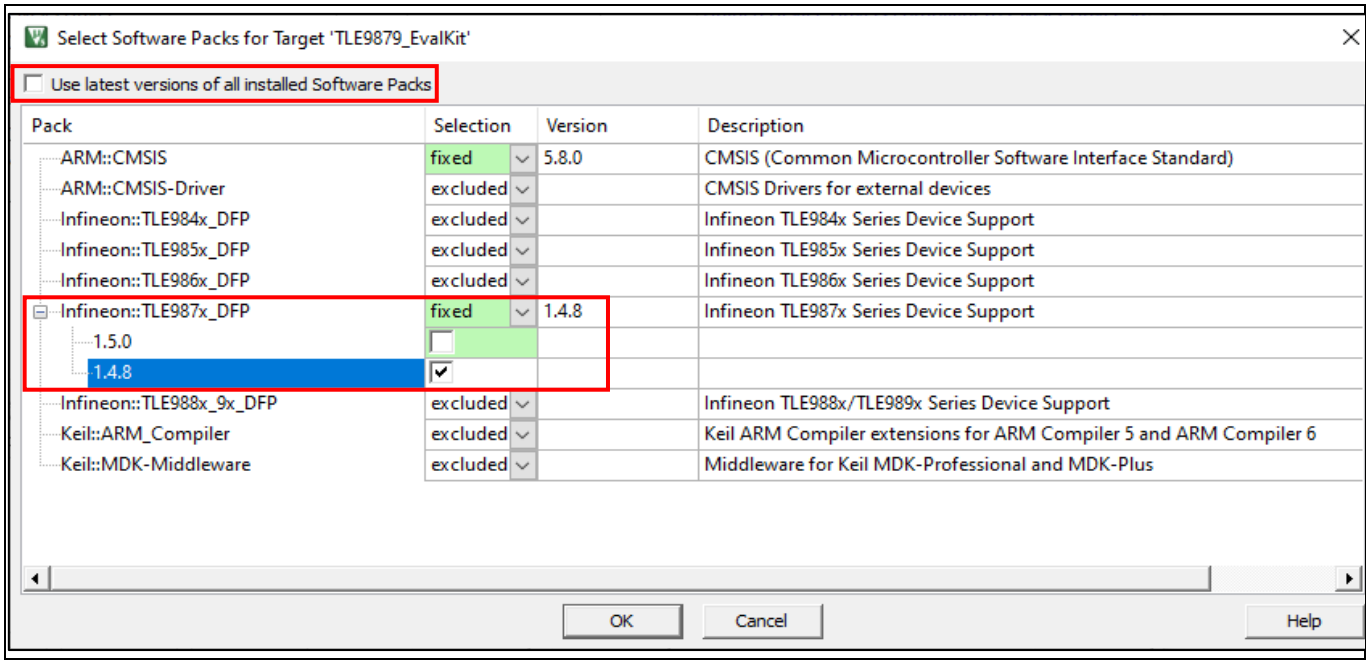

Figure 7 Use of an older pack in Keil uVision

Check that there is no conflict in the ***Manage Run-Time Environment*** window, otherwise solve them, before clicking ***OK***.

1.5 How to update a project with a new pack version

When you open a project created with an older pack, there may be files which have been updated in the newer pack. Keil informs you about these files with a small icon next to their names.

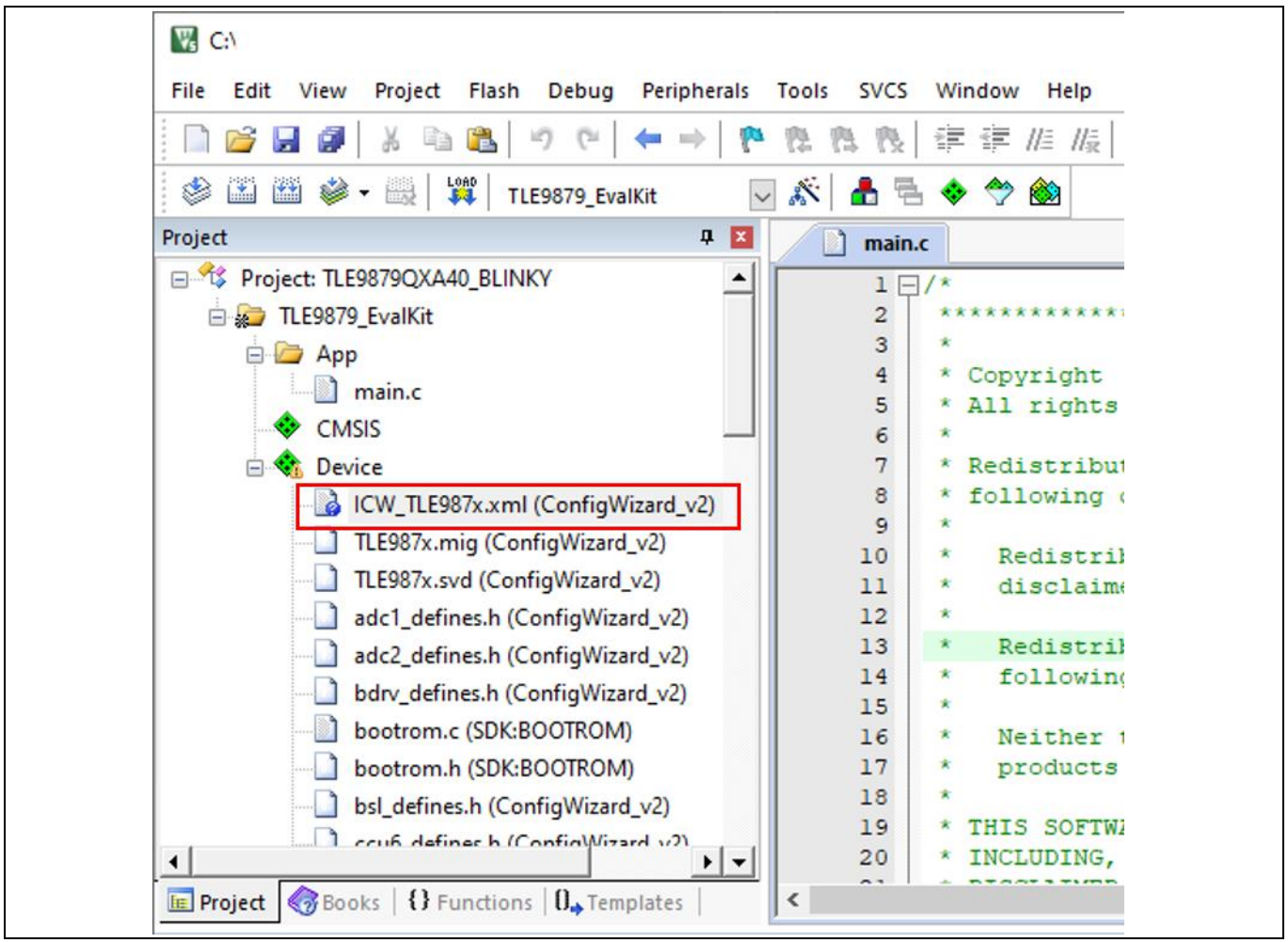

Figure 8 Example of file to update in Keil uVision

You can encounter the following icons:

-  for a subminor version update, e.g. v1.2.3 to v1.2.4
-  for a minor version update, e.g. v1.2.3 to v1.3.0
-  for a major version update, e.g. v1.2.3 to v2.0.0

You may want to update the concerned files so that your project is running with the newer pack. To do, right-click on the files to display the options and select ***Update Config File…***.

Unfortunately, there is no way to update all modified files at once.

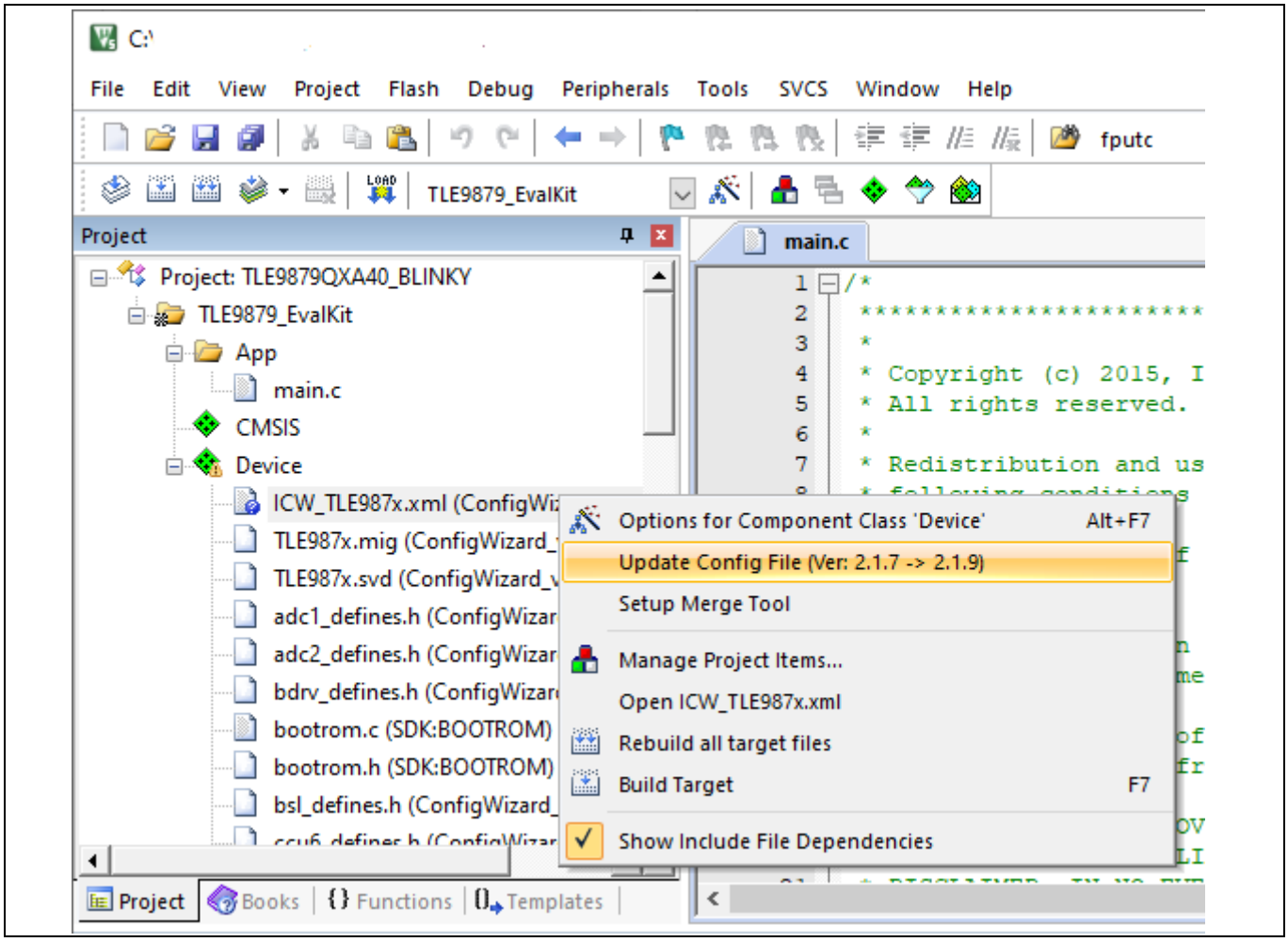

Figure 9 Update Config File in Keil uVision

1.6 How to show variables/registers/memory in the debugger

Sometimes a variable is not shown in the debugger. This is because **local** variables don't have a fix address in RAM, they are placed in the stack or in registers. On the contrary, **static** or **global** variables have a reserved space in RAM and can be shown in the debugger.

```
uint8 u8_globalVar;

void function(void)
{
  static uint8 u8_staticVar = 0;
  uint8 u8_localVar;

  /* code... */
}
```

Figure 10 Function with different variables

In the example code, the two variables `u8_globalVar` and `u8_staticVar` can be seen in the debugger, while `u8_localVar` can only be watched when the function is being executed. By adding `static` to the local variable, it can be made static and also be seen in the debugger.

You can see the variable then by right-clicking it and selecting **Add *variable_name* to... → Watch 1** or **Watch 2**. The Watch windows can also be opened from the menu **View**, by selecting **Watch Windows** and then **Watch 1** or **Watch 2**.

In Keil uVision, the **System Viewer Windows** offers access to the registers of the device, and also to single fields of the registers.

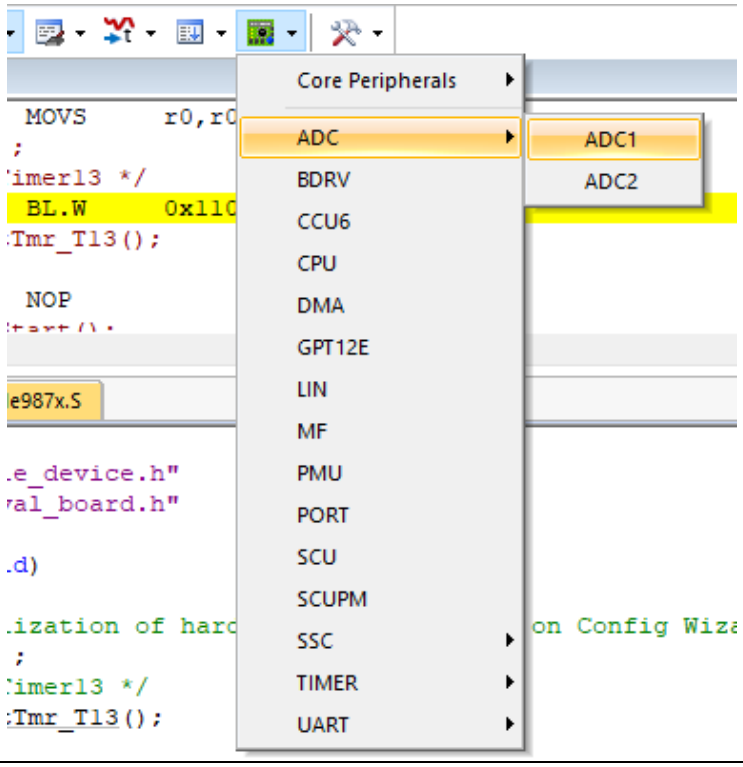

Figure 11 Register view in Keil uVision

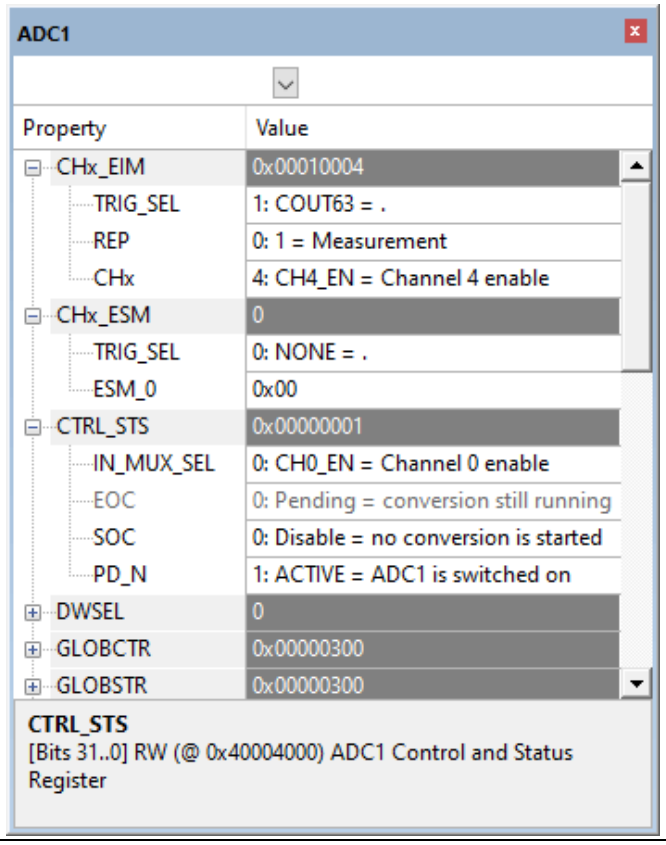

Figure 12 Registers for ADC1 module in Keil uVision

The values in the Watch and Register windows can be updated periodically by selecting the option **Periodic Window Update** in the menu **View** (when checked, the values are updated automatically).

In Keil uVision, the memory viewer can be opened by selecting **View**, then **Memory Windows** and **Memory 1/2/3/4**. In the Memory window, the memory can be watched, specified by an address (entered as hexadecimal, with the prefix 0x).

1.7 How to check for a stack over-/underflow

In case a global variable is changed without apparent reason, the root cause might be a stack overflow. This happens when the stack size is too low. In our Keil uVision examples, the stack size can be defined in the startup file.

startup_tle987x.S

```
30
31  ; <h> Stack Configuration
32  ; </h>
33
34  Stack_Size      EQU     0x200
35
36                  AREA    STACK, NOINIT, READWRITE, ALIGN=3
37  Stack_Mem       SPACE   Stack_Size
38  __initial_sp
```

Figure 13 Stack size definition in the startup file

The map file indicates where the stack begins (lowest address of the stack), as highlighted in the following figure. It can be opened after compilation by double-clicking on the **Target** in the Project explorer, or alternatively by navigating to it in the File explorer. It is located in the **Listings** folder of the project.

ADC1_POTI_EIM.map

```
    i.main                          0x11000f4c   Section        0  main.o(i.main)
    .constdata                      0x11000f78   Section       92  bootrom.o(.constdata)
    .ARM.__at_0x1101EFFC            0x1101effc   Section        4  system_tle987x.o(.ARM.__at_0x1101EFFC)
    __tagsym$$used                  0x1101effc   Number         0  system_tle987x.o(.ARM.__at_0x1101EFFC)
    .data                           0x18000018   Section        5  wdt1.o(.data)
    bSOWactive                      0x1800001c   Data           1  wdt1.o(.data)
    STACK                           0x18000020   Section      512  startup_tle987x.o(STACK)
```

Figure 14 Stack begin indicated in the map file

In this case, the stack is from 0x18000020 to 0x1800021F. If the stack pointer is not in this range, a stack under-or overflow has occurred.

In case there is a stack overflow, the stack pointer doesn't point to an address within the stack range.

2 IAR Embedded Workbench Topics

2.1 SDK help and general documentation

You can access the SDK help and the general documentation in the section ***Device data books*** of the tab ***Device*** in the rteconfig file of your project.

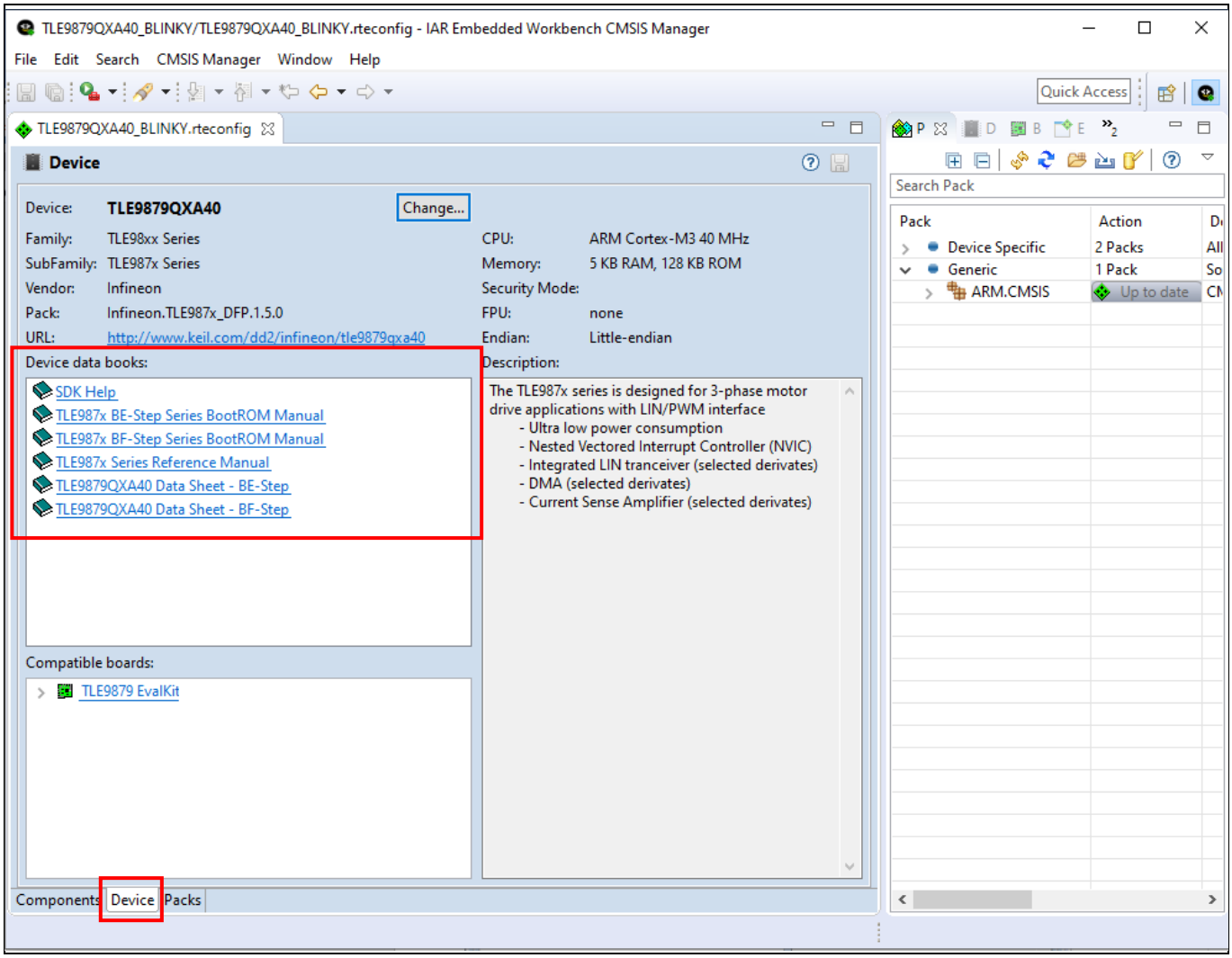

Figure 15 General documentation in the tab Device in IAR Embedded Workbench

2.2 How to use a fixed pack version

Per default, a project runs with the latest installed pack of the used chip. You still have the possibility to use an older fixed pack.

In the CMSIS Manager, open the ***.rteconfig*** file of your project and select the tab ***Packs***. In the window, you can see the packs that are installed on your computer for the device that you are using.

In the figure below, there are two packs installed for the device TLE9879QXA40: v1.5.1 and v1.5.0.

- The checkbox ***Use all latest Packs*** is checked by default, so that the pack v1.5.1 is used for your current project.
- The icon in front of the used pack version is yellow
- The label right to the pack name in the Selection column is "*latest*"
- The cell right to the used pack version in the Selection column is green and checked.

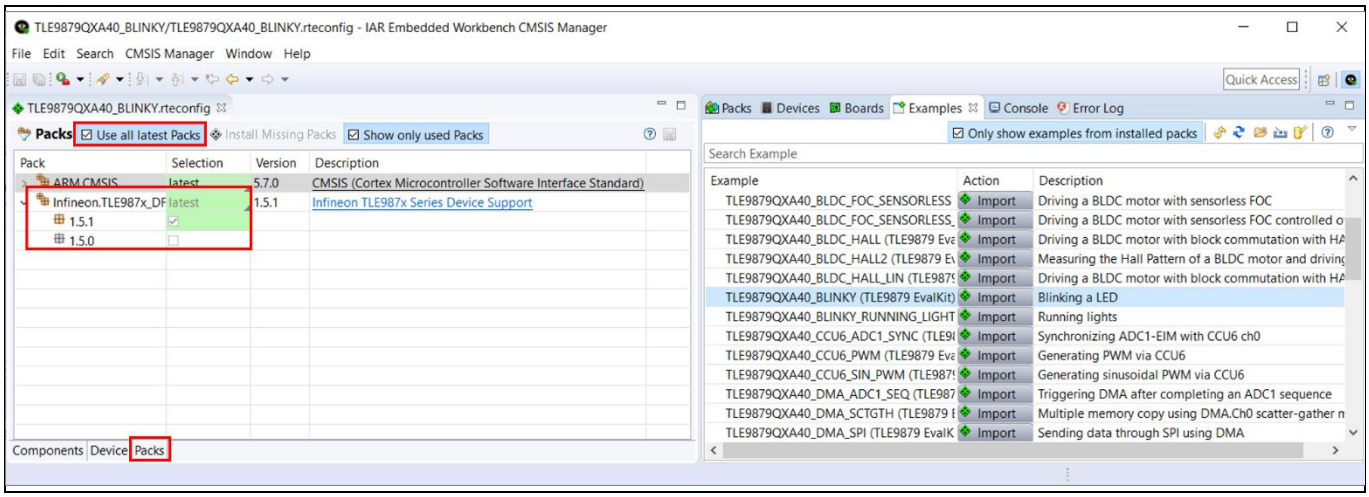

Figure 16 Tab Packs in the .rteconfig file in the CMSIS Manager in IAR Embedded Workbench

To use the previous version v1.5.0, uncheck the checkbox ***Use all latest Packs*** and select the version 1.5.0.

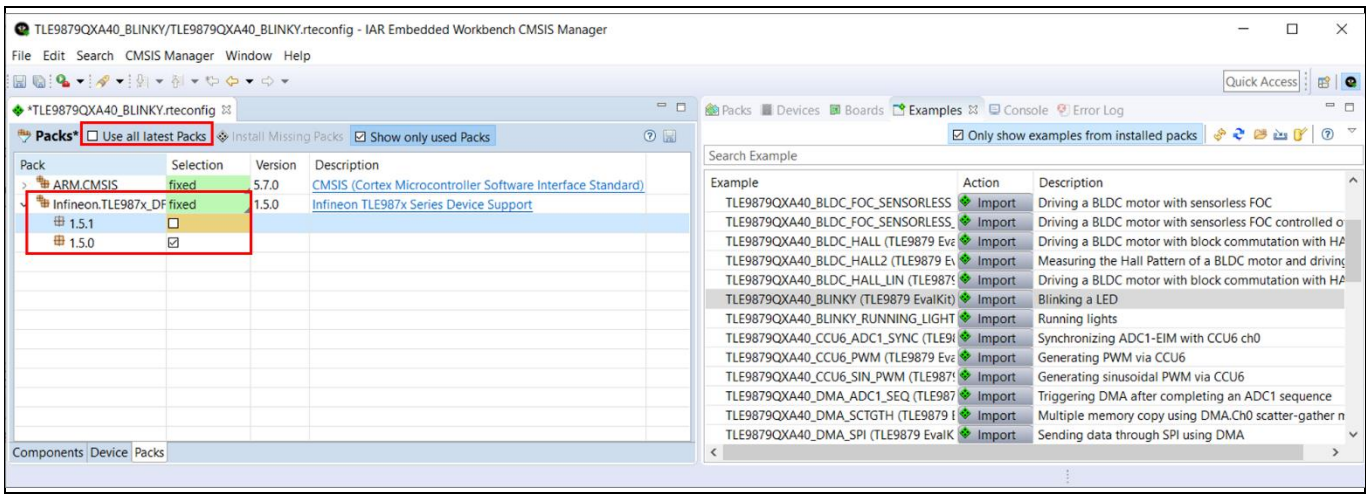

Figure 17 Selection of a fixed pack version in IAR Embedded Workbench

When selecting this fixed version:

- The icon in front of v1.5.0 is now yellow – it was grey before
- The label right to the pack name in the Selection column is "*fixed*"
- The cell right to the used pack version (1.5.0) in the Selection column is white and checked.
- The cell right to the previously used pack version (1.5.1) is yellow

When saving your configuration, the cell right to the version 1.5.0 becomes green. In the ***Components*** tab, you can see that the fixed pack version 1.5.0 has been considered by hoovering onto the different components of the project.

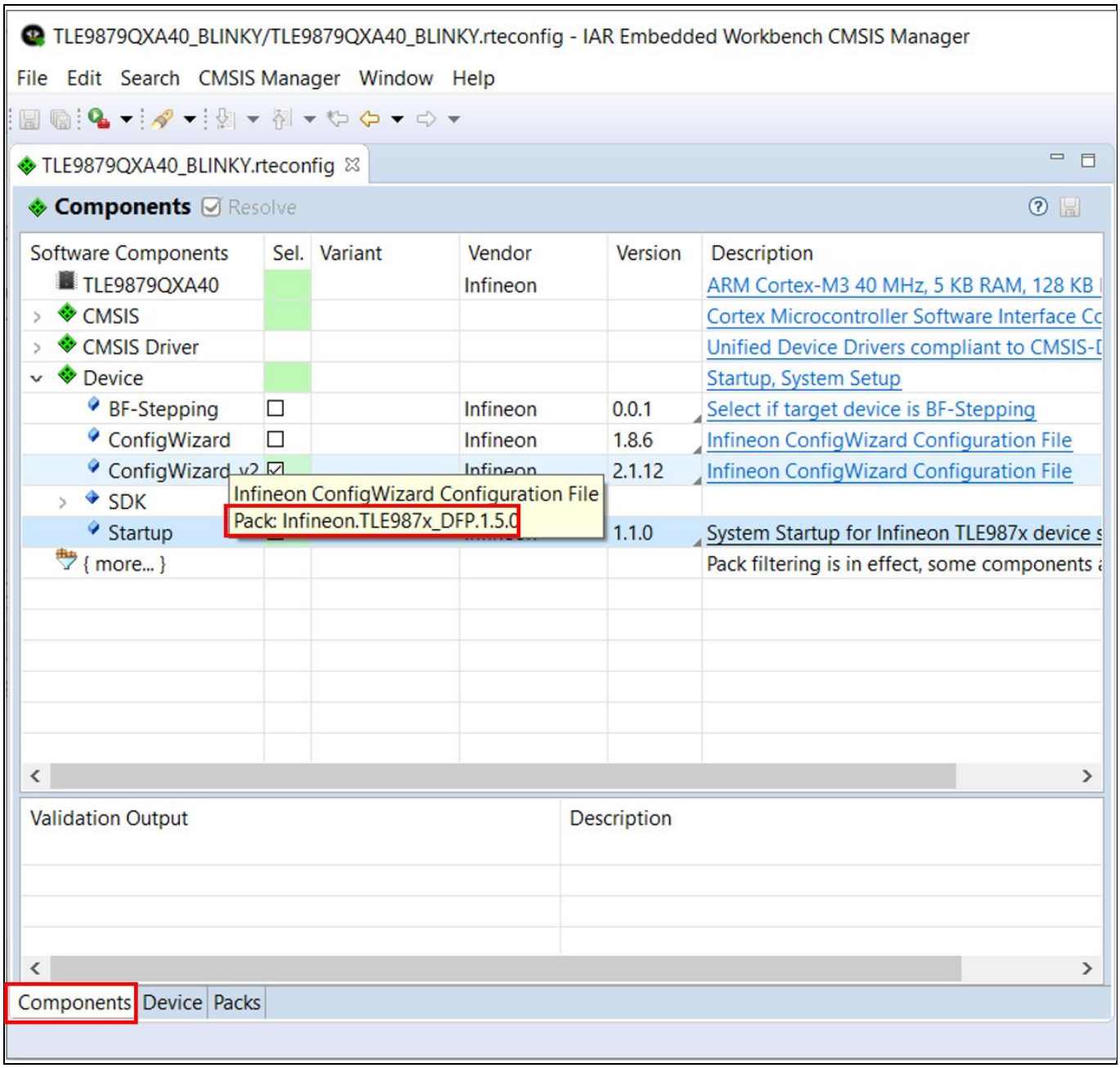

Figure 18 Use of a fixed pack version in the tab Components in IAR Embedded Workbench

2.3 How to show variables/registers/memory in the debugger

In IAR Embedded Workbench, you can access the registers of the devices in the Debug mode via the menu **View** and by selecting **Registers**.

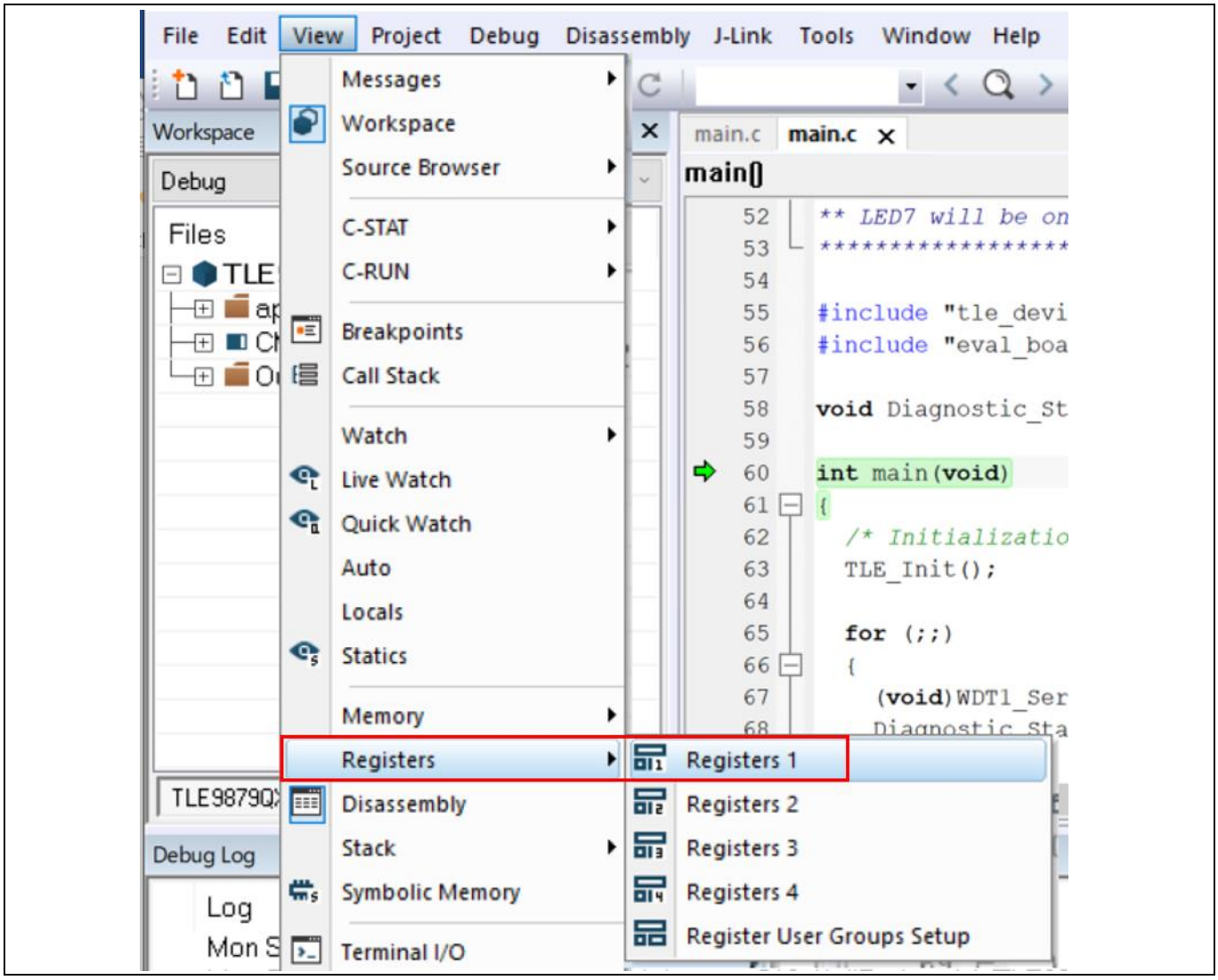

Figure 19 Register view in IAR Embedded Workbench

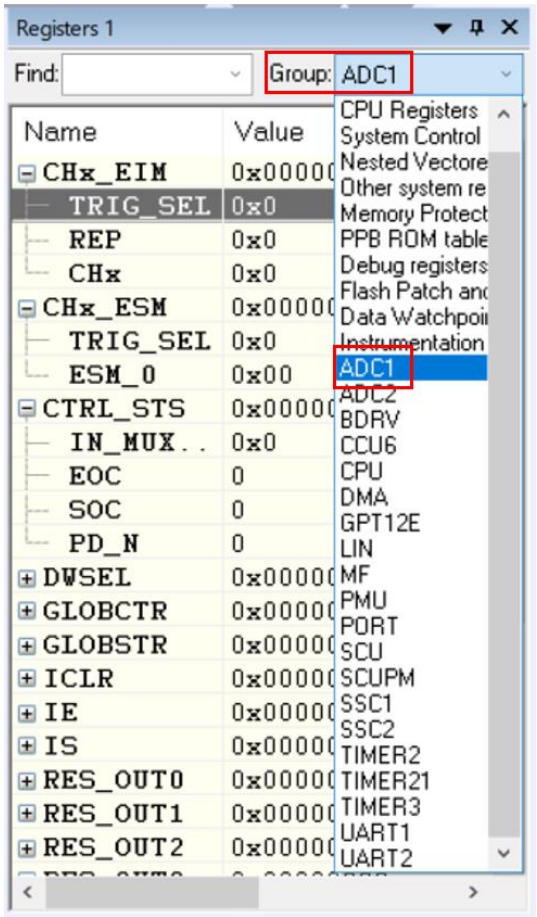

Figure 20 Registers for ADC1 module in IAR Embedded Workbench

In IAR Embedded Workbench, the memory viewer can be opened by selecting **View**, then **Memory Windows** and **Memory 1/2/3/4**. In the Memory window, the memory can be watched, specified by an address (entered as hexadecimal, with the prefix 0x).

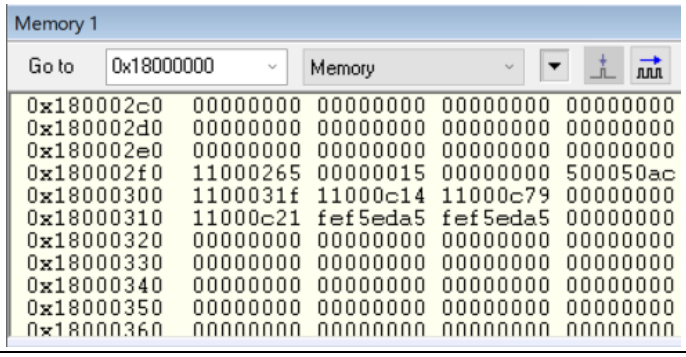

Figure 21 Memory view in IAR Embedded Workbench

3 Config Wizard for MOTIX™ MCU Topics

3.1 Why to use Config Wizard for MOTIX™ MCU

Config Wizard for MOTIX™ MCU (Embedded Power ICs) is a tool which allows an easy configuration of Automotive Embedded Power IC products.

The GUI is designed to be intuitive. It is divided into several tabs, each tab focusing on the configuration of one chip module. The settings are done via comboboxes, checkboxes, edits, radiobuttons, spinboxes, …. In some cases, a graphic overview is provided to summarize the configuration and enhance the understanding of the user.

It is also designed to facilitate the configuration, e.g. by locking elements which are influenced by other elements (see Help within Config Wizard for MOTIX™ MCU) or by greying out elements that must not be configured in certain cases.

It is available via the Infineon Toolbox.

3.2 Principle of Config Wizard for MOTIX™ MCU

When saving a configuration in Config Wizard, several files are created/updated:

- **config.icwp**

This file saves the settings selected in the GUI. By opening the project the next time in the Config Wizard, the previously configured settings are loaded from the config.icwp file.

- ***_defines.h header files**

Config Wizard exports several *_defines.h header files, one per module. These header files contain of macro defines only and are further processes by the SDK files.

ccu6_defines.h

```
28  ********************************************
29  #ifndef CCU6_DEFINES_H
30  #define CCU6_DEFINES_H
31
32  /* XML Version 2.2.3 */
33  #define CCU6_XML_VERSION (20203)
34
35  #define CCU6_CC60SR (0xFA) /*decimal 250*/
36
37  #define CCU6_CC61SR (0x1F4) /*decimal 500*/
38
39  #define CCU6_CC62SR (0x2EE) /*decimal 750*/
40
41  #define CCU6_CC63SR (0x0) /*decimal 0*/
```

Figure 22 Snippet of ccu6_defines.h

The macro defines are then further used by the according initialization routines, written to the according registers directly to apply the settings from Config Wizard.

ccu6.c

```
37  /** \brief Initializes the CCU6 module.
38   *
39   * \param None
40   * \return None
41   *
42   * \ingroup drv_api
43   */
44  void CCU6_Init(void)
45  {
46    CCU6->TCTR0.reg = (uint16) CCU6_TCTR0;
47    CCU6->T12PR.reg = (uint16) CCU6_T12PR;
48    CCU6->T13PR.reg = (uint16) CCU6_T13PR;
49    CCU6->T12DTC.reg = (uint16) CCU6_T12DTC;
50    CCU6->CC60SR.reg = (uint16) CCU6_CC60SR;
51    CCU6->CC61SR.reg = (uint16) CCU6_CC61SR;
52    CCU6->CC62SR.reg = (uint16) CCU6_CC62SR;
53    CCU6->CC63SR.reg = (uint16) CCU6_CC63SR;
54    CCU6->TCTR2.reg = (uint16) CCU6_TCTR2;
55    CCU6->TRPCTR.reg = (uint16) CCU6_TRPCTR;
56    CCU6->MODCTR.reg = (uint16) CCU6_MODCTR;
57    CCU6->MCMCTR.reg = (uint16) CCU6_MCMCTR;
58    CCU6->T12MSEL.reg = (uint16) CCU6_T12MSEL;
59    CCU6->PSLR.reg = (uint16) CCU6_PSLR;
60    CCU6->INP.reg = (uint16) CCU6_INP;
61    CCU6->IEN.reg = (uint16) CCU6_IEN;
62    CCU6->CMPSTAT.reg = (uint16) CCU6_CMPSTAT;
63    CCU6->PISEL0.reg = (uint16) CCU6_PISEL0;
64    CCU6->PISEL2.reg = (uint16) CCU6_PISEL2;
65    CCU6_T12_Str_En();
66    CCU6_T13_Str_En();
67  }
68
```

Figure 23 Initialization of the CCU6 modules with the defines from Config Wizard

After this initialization routine, the registers of the device contain the values as configured in Config Wizard.

3.3 Help within Config Wizard for MOTIX™ MCU

Locked Element

As mentioned above, the Config Wizard is intended to facilitate the configuration of your device. For this reason, some elements are locking other UI elements to a certain value, meaning that this UI element is disabled for user input and set to the stated value.

For example, selecting the pin P1.2 as input for the Auxiliary Timer 2 of the module GPT12E will automatically lock the direction of the port P1.2 to Input. The yellow tooltip enables you to see the name of the related element.

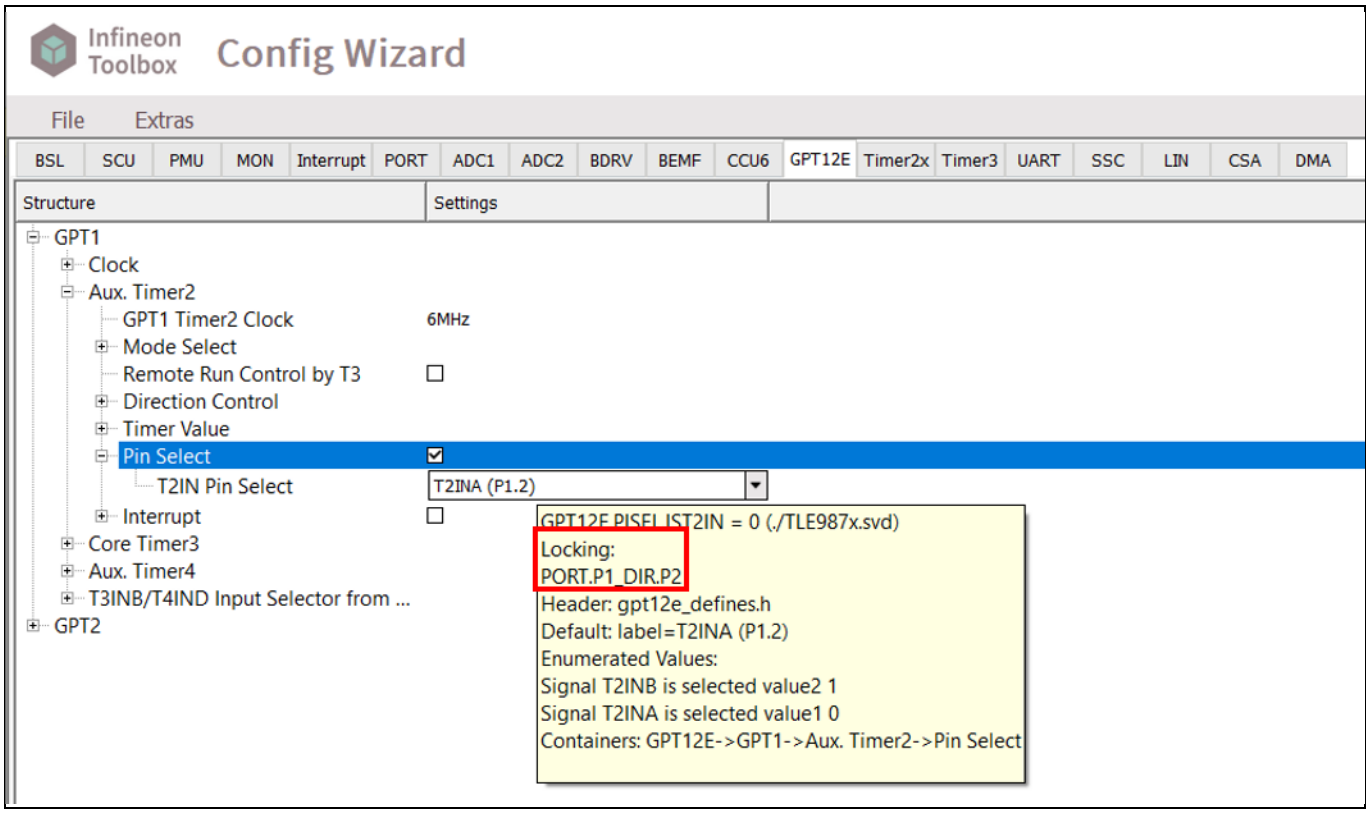

Figure 24 Locking element in the Config Wizard

In the Port tab, you can see that the locked element is now greyed out, which prevents you from changing its value. By hoovering onto the element, the yellow tooltip shows which element is locking the port direction, namely the pin selection in the Auxiliary Timer 2 of the GPT12E module.

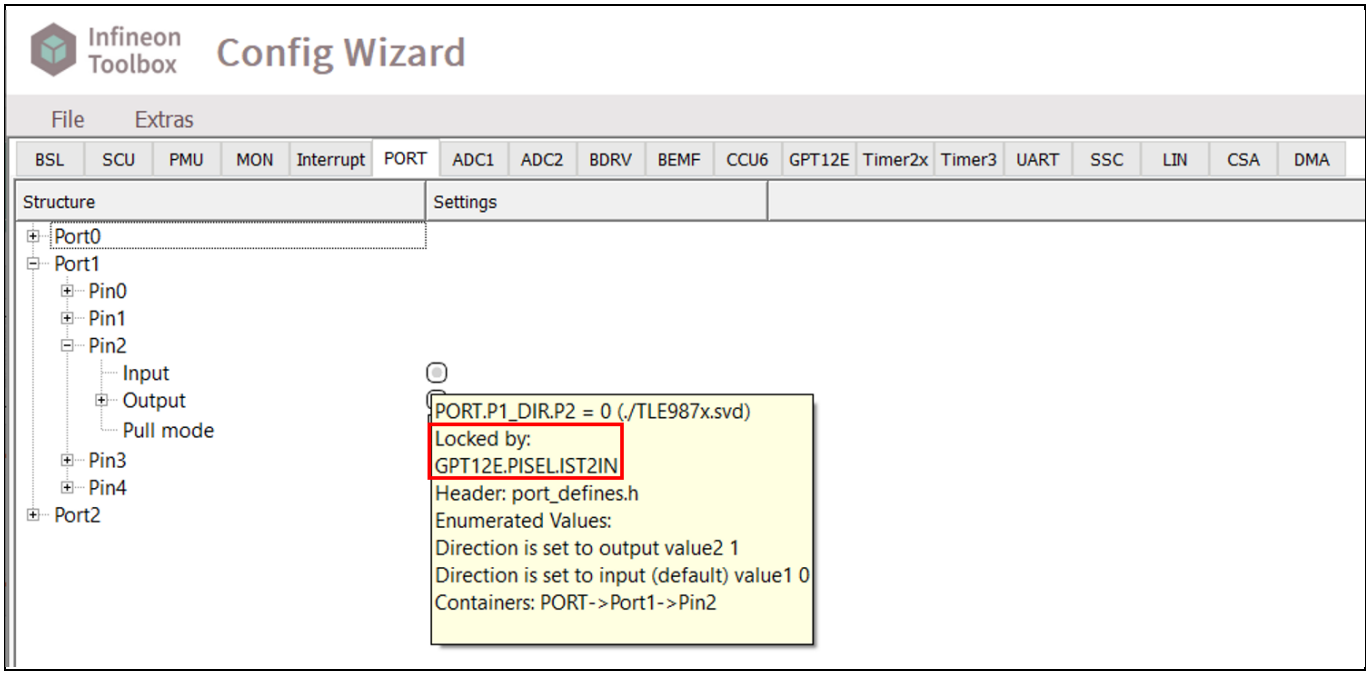

Figure 25 Locked element in the Config Wizard

Conflict due to Locked Elements

It may happen that you select elements which are locking the same UI element but with two different values. In that case, there is a so-called conflict which is notified in the *Log* window in the Config Wizard.

For example, let's select the pin P1.2 for the Auxiliary Timer 2 in the GPT12E module. As seen previously, it will lock the port P1.2 as Input.

In the SSC2 module configured as Slave, let's select P1.2 as Transmitter Output. This is not forbidden by the tool but a warning appears at the bottom.

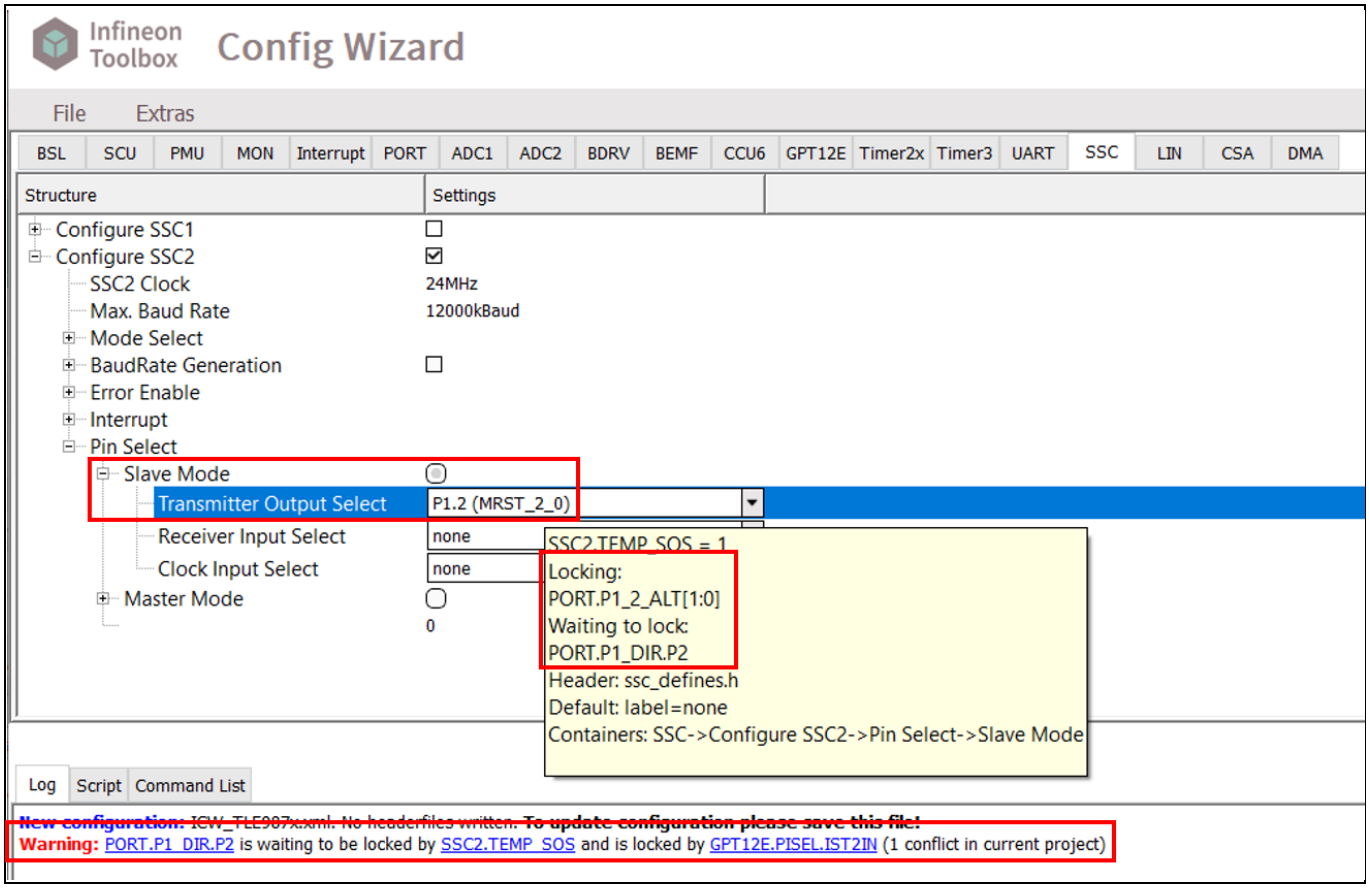

Figure 26 Warning due a conflict in locking in Config Wizard

The conflict comes from the fact that the port P1.2 is locked on two contradictory ways: as input for GPT12E and as output for SSC2.

As long as the conflict is not solved, it is not possible to save the whole configuration of the Config Wizard.

3.4 How to integrate the Config Wizard for MOTIX™ MCU into Keil uVision

If the Config Wizard notices on startup that a Keil µVision 5 is installed on the system, it asks to integrate itself into the Keil develop environment. A dialog comes up where already installed Config Wizards are shown and where you can decide if you want to integrate the newly installed Config Wizard into Keil. This would mean that it can be started from Keil as an external tool.

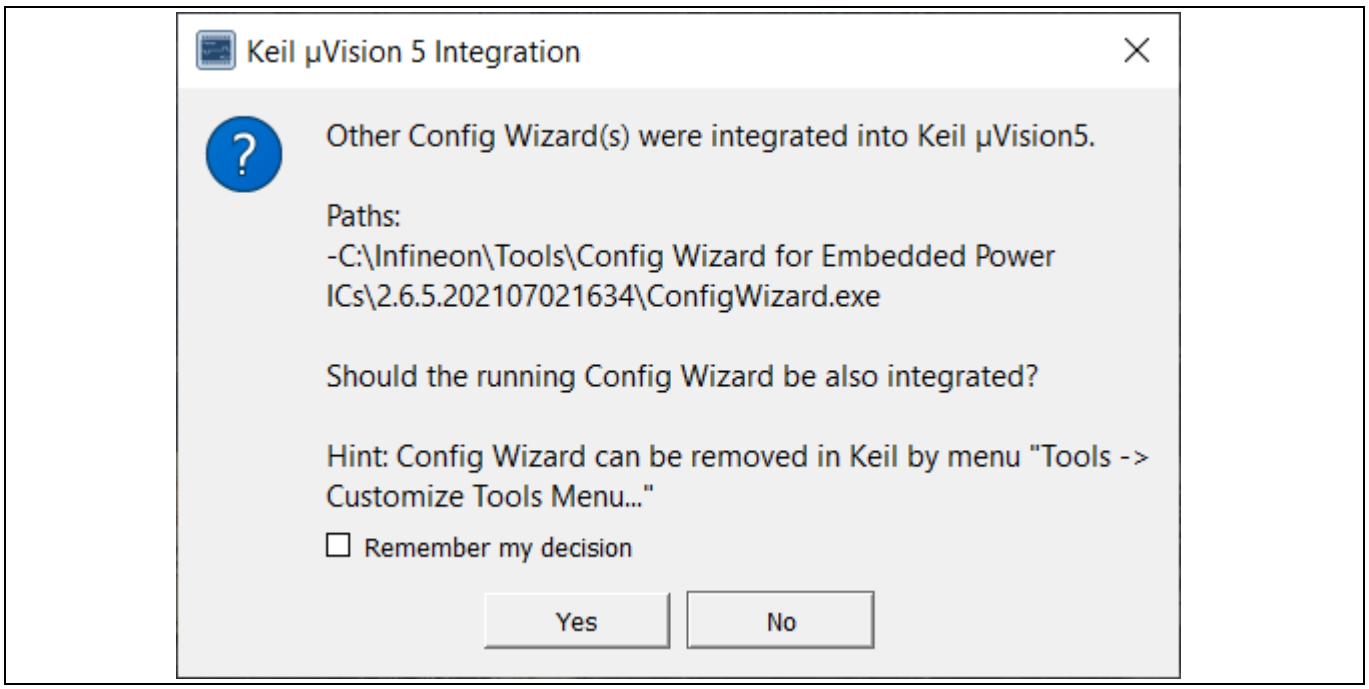

Figure 27 Automatic integration of Config Wizard into Keil uVision

You can also manually add the Config Wizard in the Tools menu of Keil uVision.

Open Keil uVision and select ***Tools > Customize Tools Menu***…

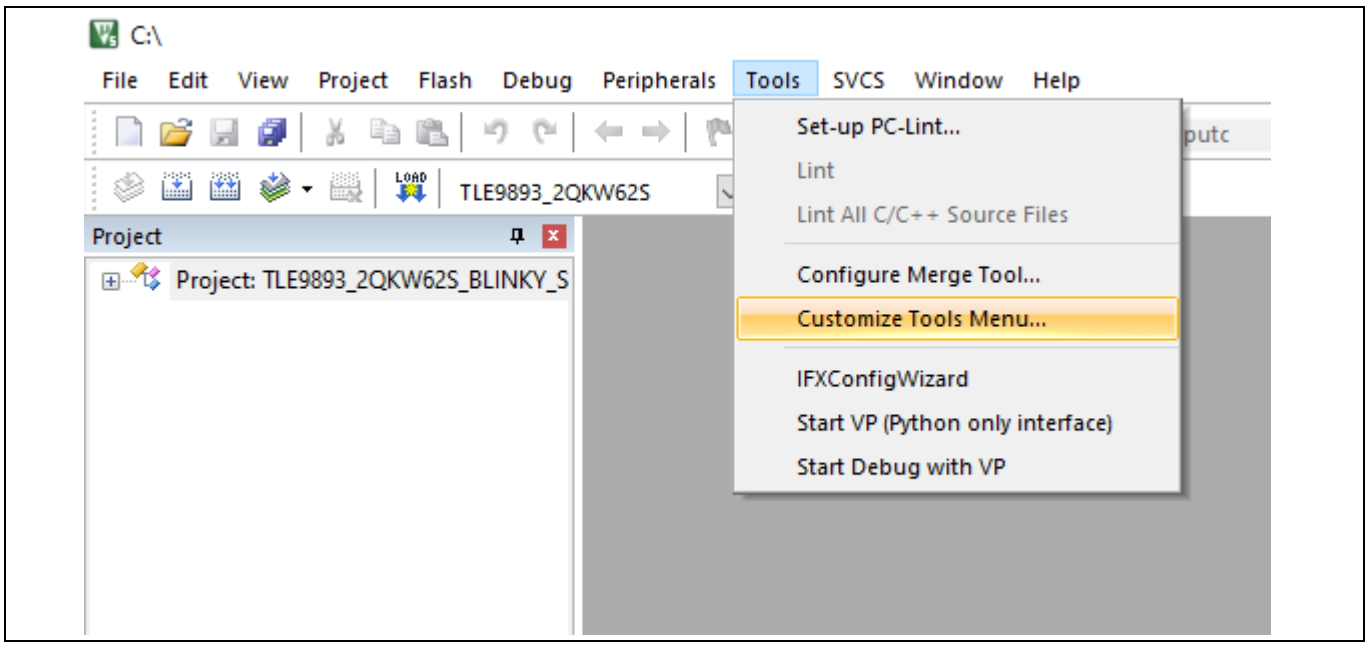

Figure 28 Menu Tools > Customize Tools Menu… in Keil uVision

The figure below shows the window in which you can integrate the Config Wizard.

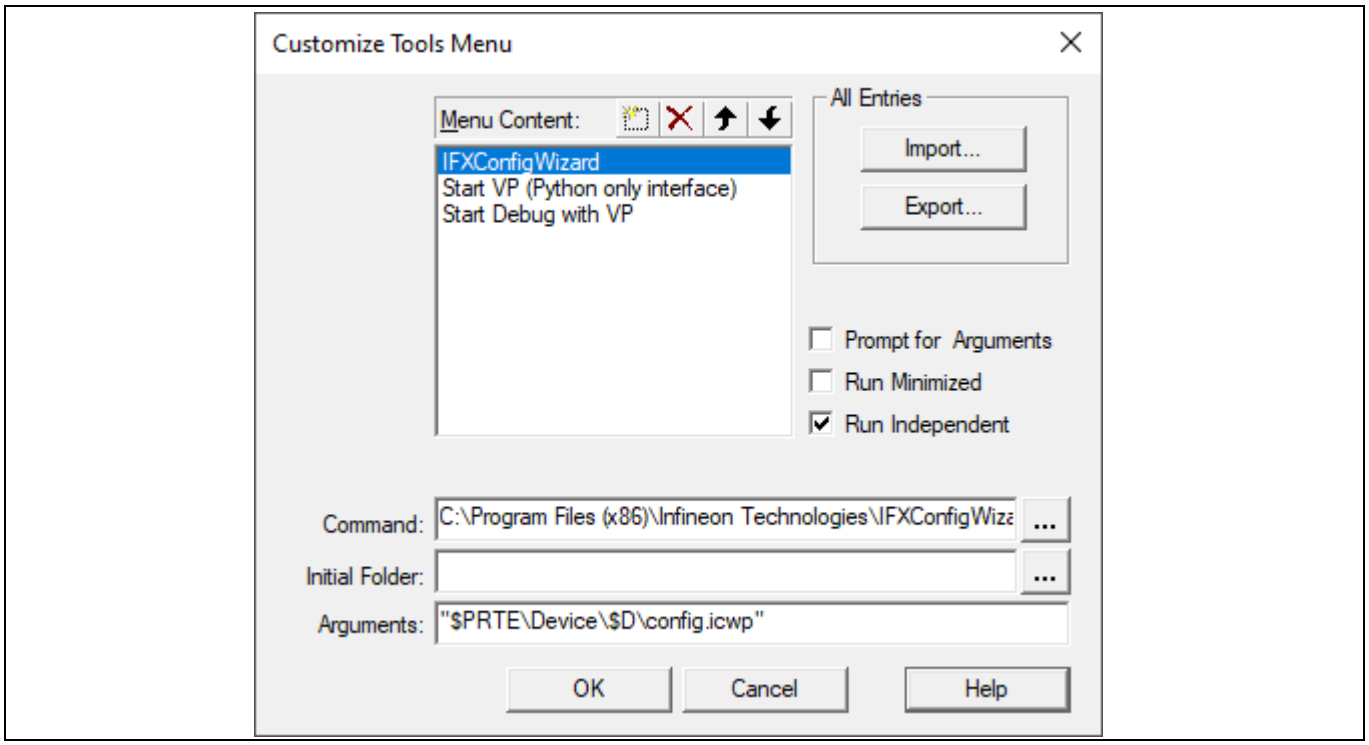

Figure 29 Window Customize Tools Menu

Click on the Add button  . A new empty line appears in the tools list, where you can add a meaningful name for the Config Wizard.
The edits at the bottom must be configured as follows:

- ***Command***: The path where your Config Wizard application is stored
- ***Initial Folder***: Not necessary, unless you always want to start the Config Wizard from the same project folder
- ***Arguments***:
  - Older than version 2.6.x: *"$PRTE\Device\$D\config.icwp"*
  - From version 2.6.x: *"$PRTE\Device\$D\config.icwp" -ddevice=$D*

Select ***Run Independent*** to be able to use the Config Wizard independently from Keil uVision.

Click ***OK*** to save your new tool. It appears now at the bottom of the Tools menu.

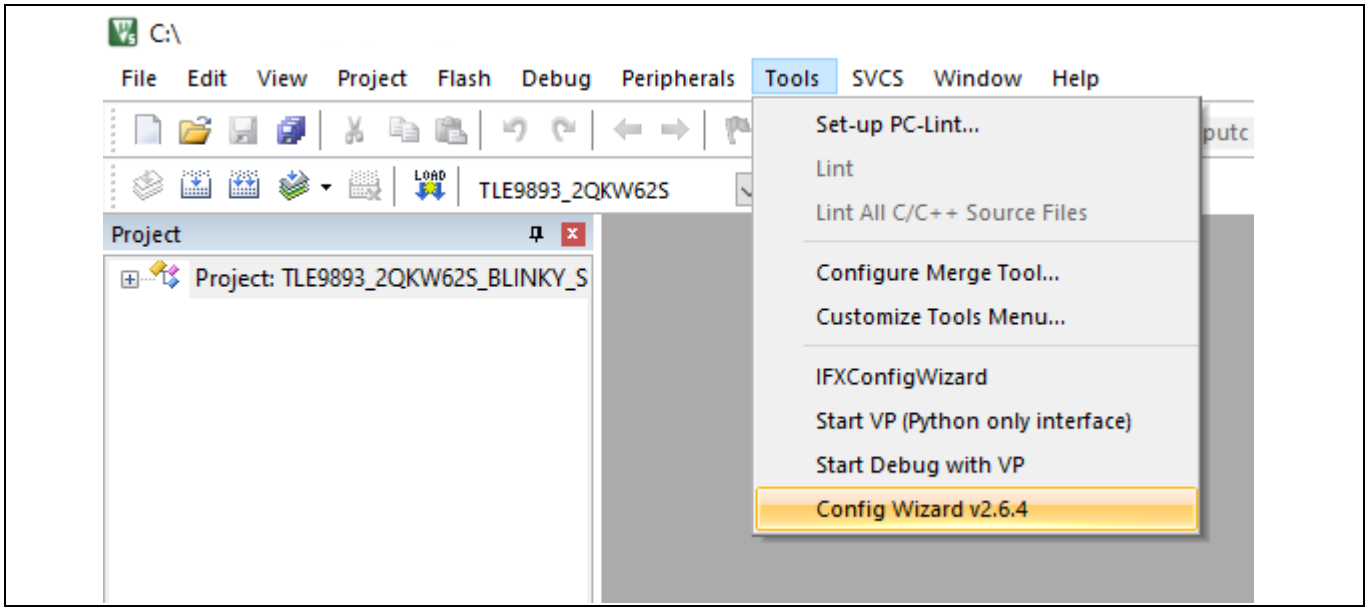

Figure 30 Config Wizard as new tool in Keil uVision

Note: For Config Wizard versions older than 2.6.x, the argument phrase is incorrect. Please correct it to ***"$PRTE\Device\$D\config.icwp" -ddevice=$D***

3.5 How to integrate the Config Wizard for MOTIX™ MCU into IAR Embedded Workbench

To integrate the Config Wizard into IAR Embedded Workbench, open the IDE and select ***Tools > Configure Tools…***

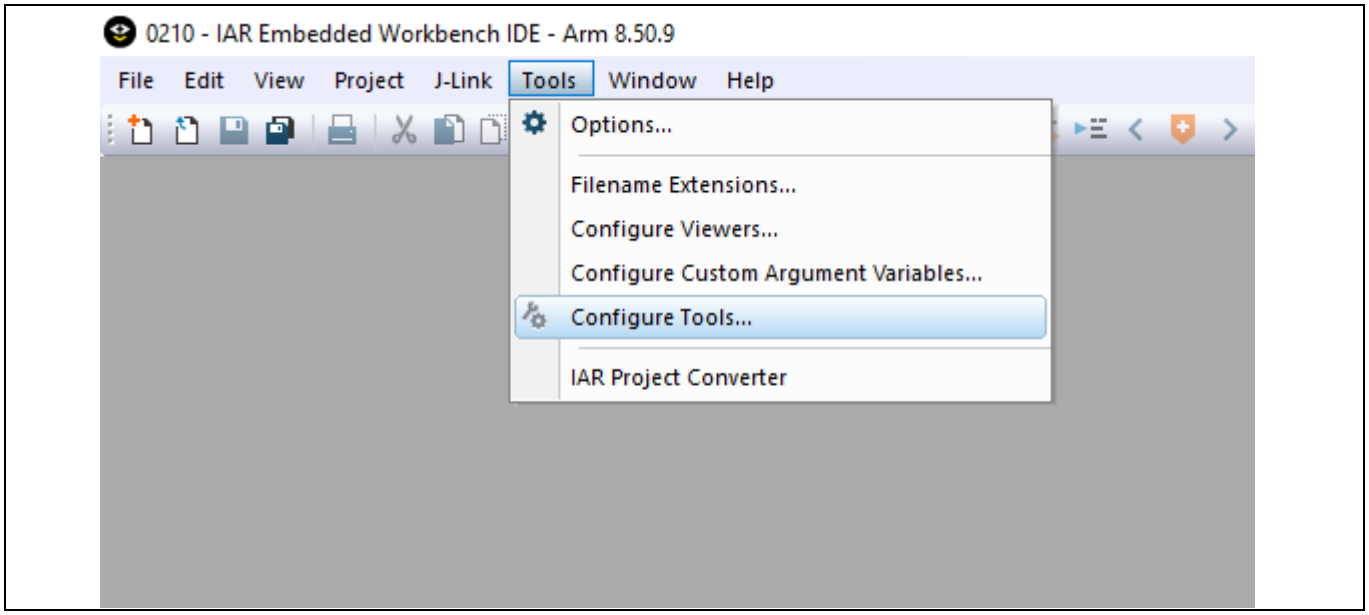

Figure 31 Menu Tools > Configure Tools… in IAR Embedded Workbench

The figure below shows the window in which you can integrate the Config Wizard.

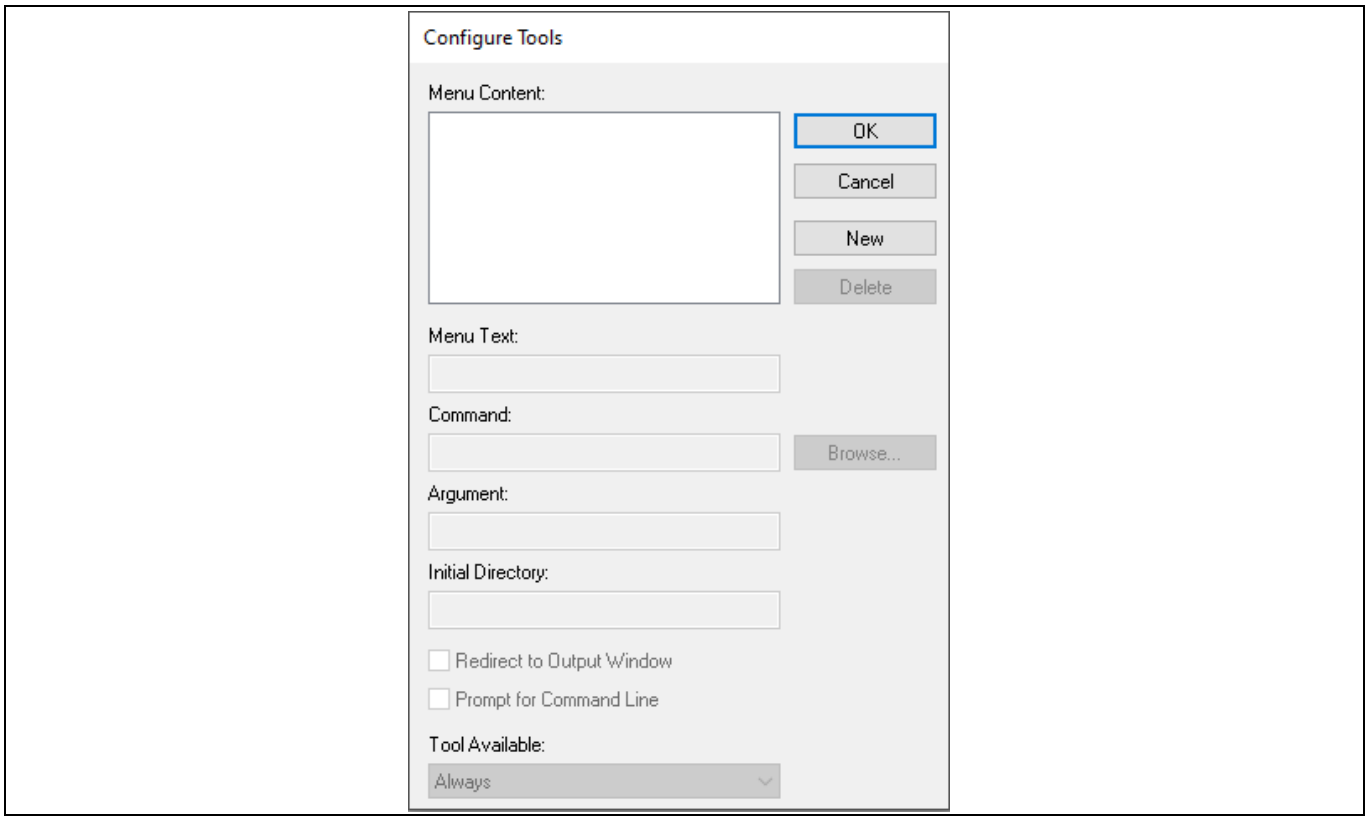

Figure 32 Window Configure Tools

Click on the ***New*** button. A new empty line appears in the Menu Content. The edits at the bottom must be configured as follows:

- ***Menu Text***: A meaningful name for the Config Wizard to add
- ***Command***: The path where your Config Wizard application is stored
- ***Argument***:
  - Older than version 2.6.x: "$PROJ_DIR$\RTE\Device\D\config.icwp"
  - From version 2.6.x: "$PROJ_DIR$\RTE\Device\D\config.icwp" -ddevice=D

  Where $PROJ_DIR$ refers to your project directory and D to the device name

Select Always for ***Tool Available*** to be able to use the Config Wizard independently from the IAR IDE.

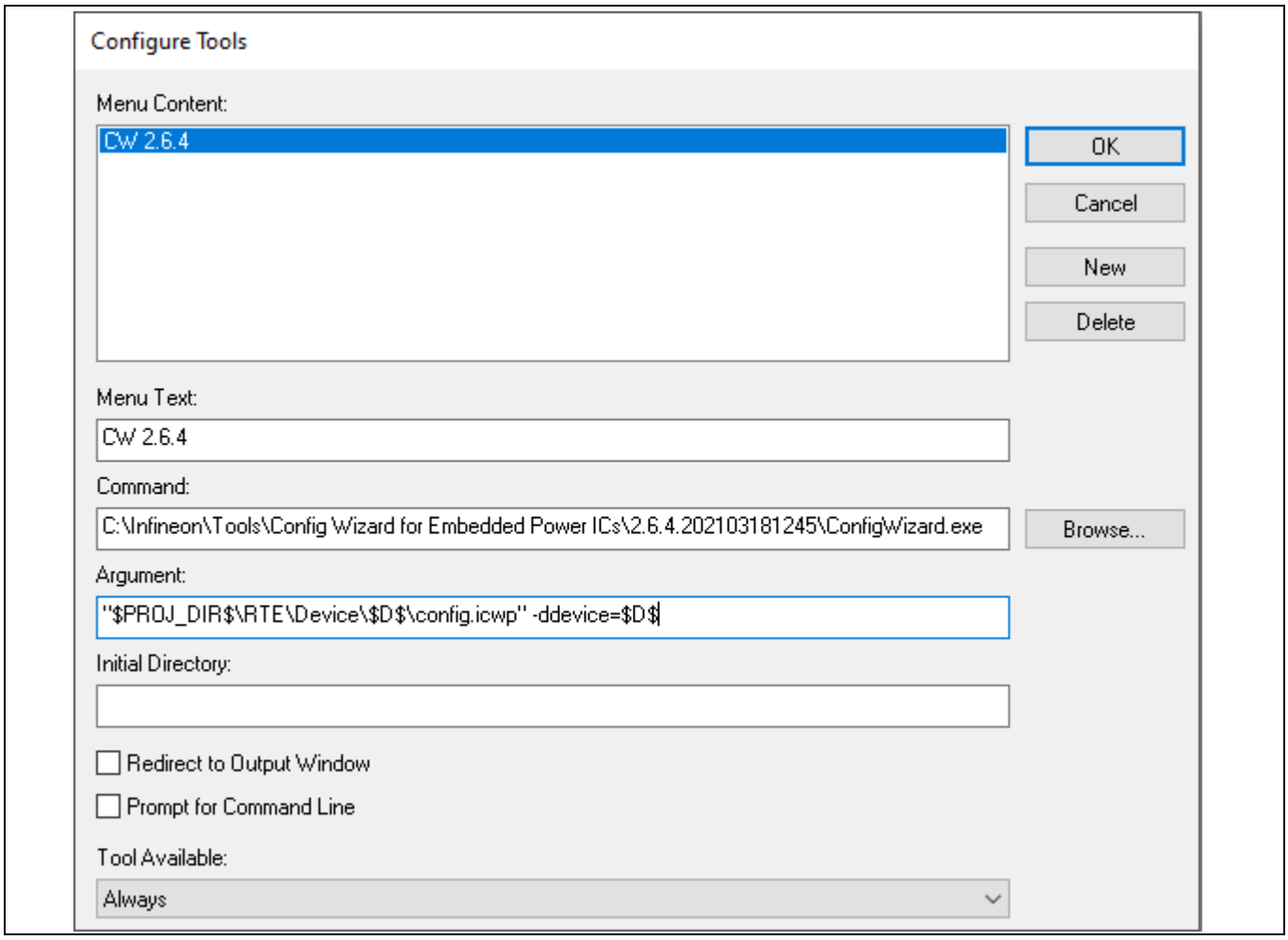

Figure 33 Configuration of Config Wizard as new tool

Click ***OK*** to save your new tool. It appears now at the bottom of the ***Tools*** menu.

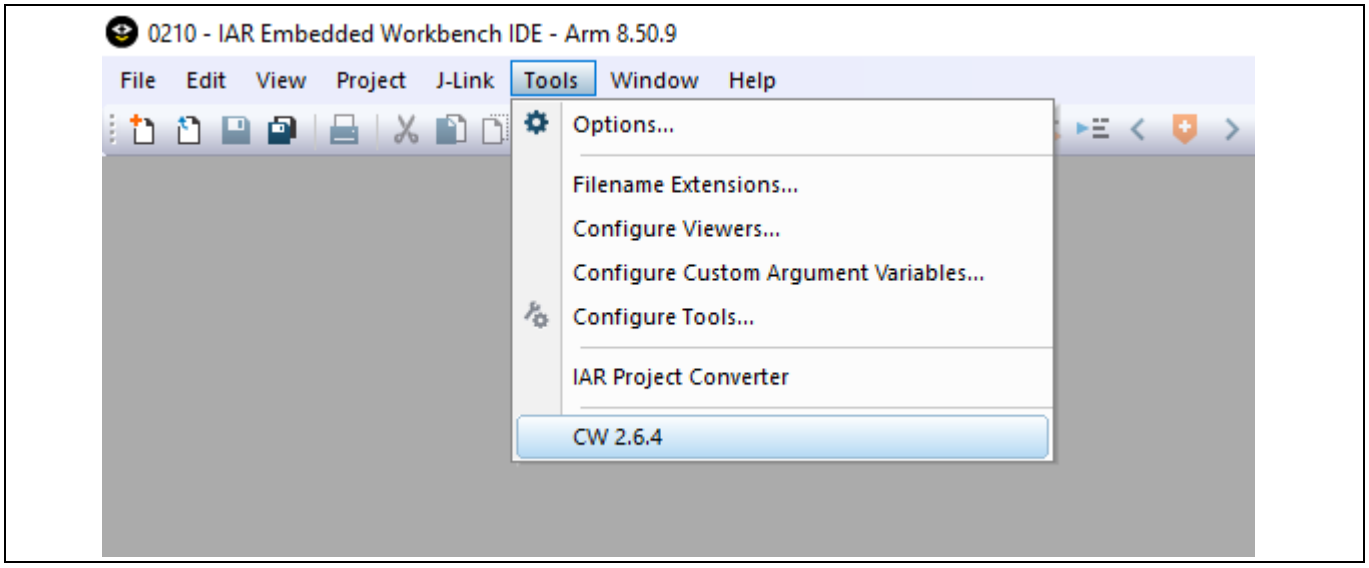

Figure 34 Config Wizard as new tool in IAR Embedded Workbench

As there is no argument variable defined in the IAR Workbench, you have to define one manually. To do so, select ***Tools >Configure Custom Argument Variables…***

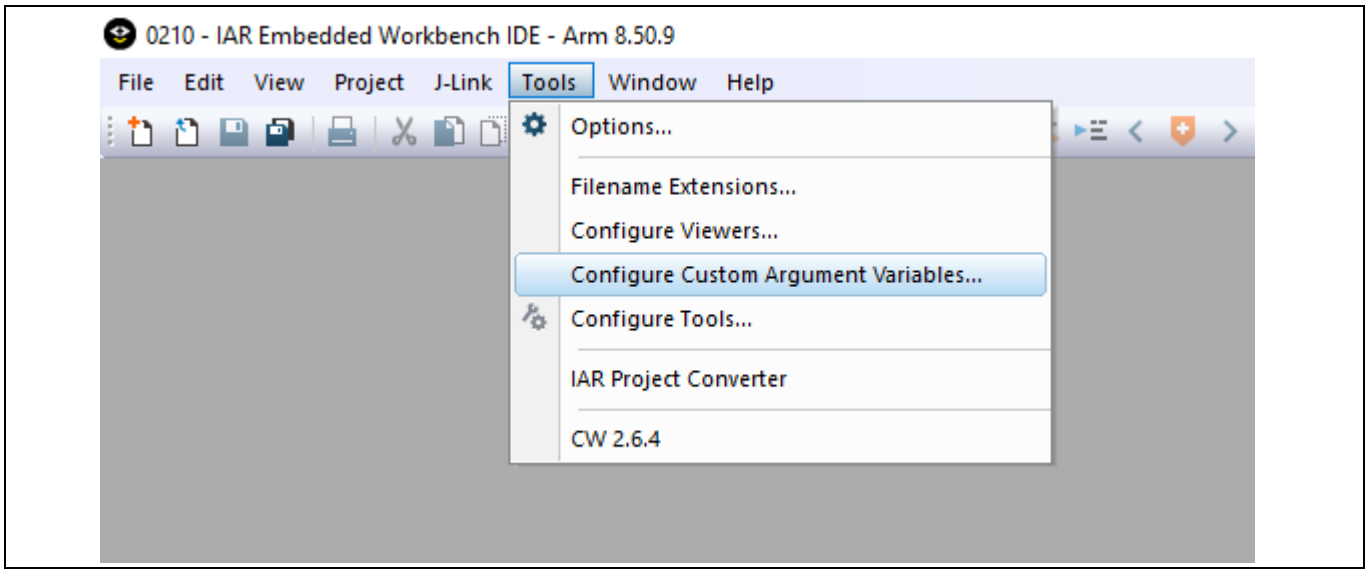

Figure 35 Menu Tools > Configure Custom Argument Variables... in IAR Embedded Workbench

The figure below shows the window in which you can set up the device variable D. Select the tab ***Global*** then ***New Group…***

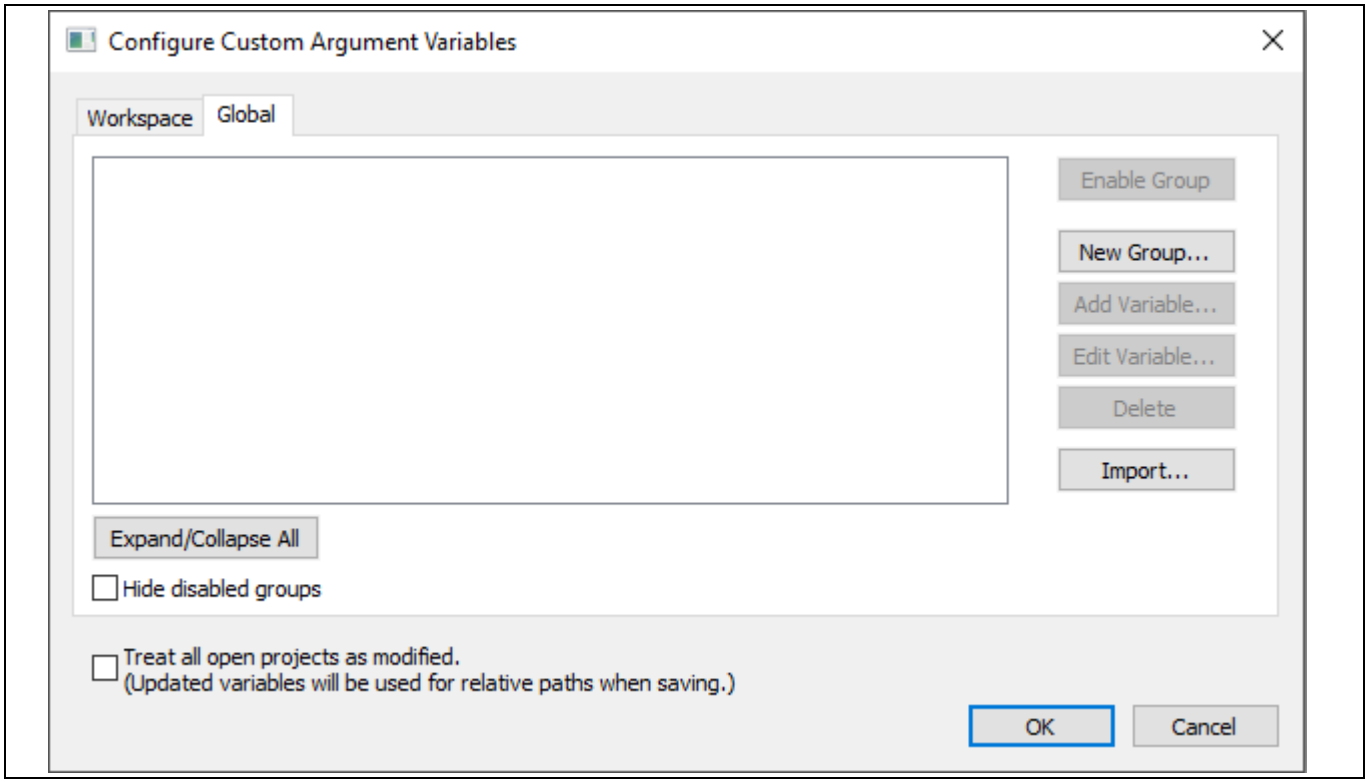

Figure 36 Window Configure Custom Argument Variables, tab Global

In the popup window, give a name to your group, for example Device. Click ***OK*** to close the window.

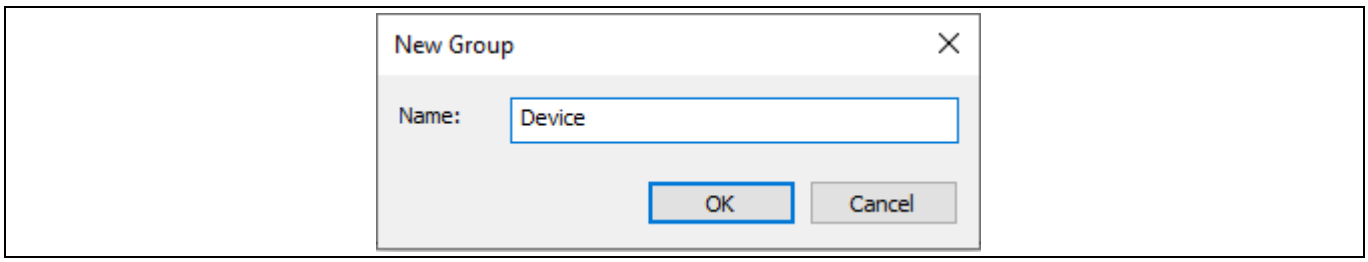

Figure 37 Window New Group

The new group appears then in the window Configure Custom Argument Variables, as shown in the next figure.

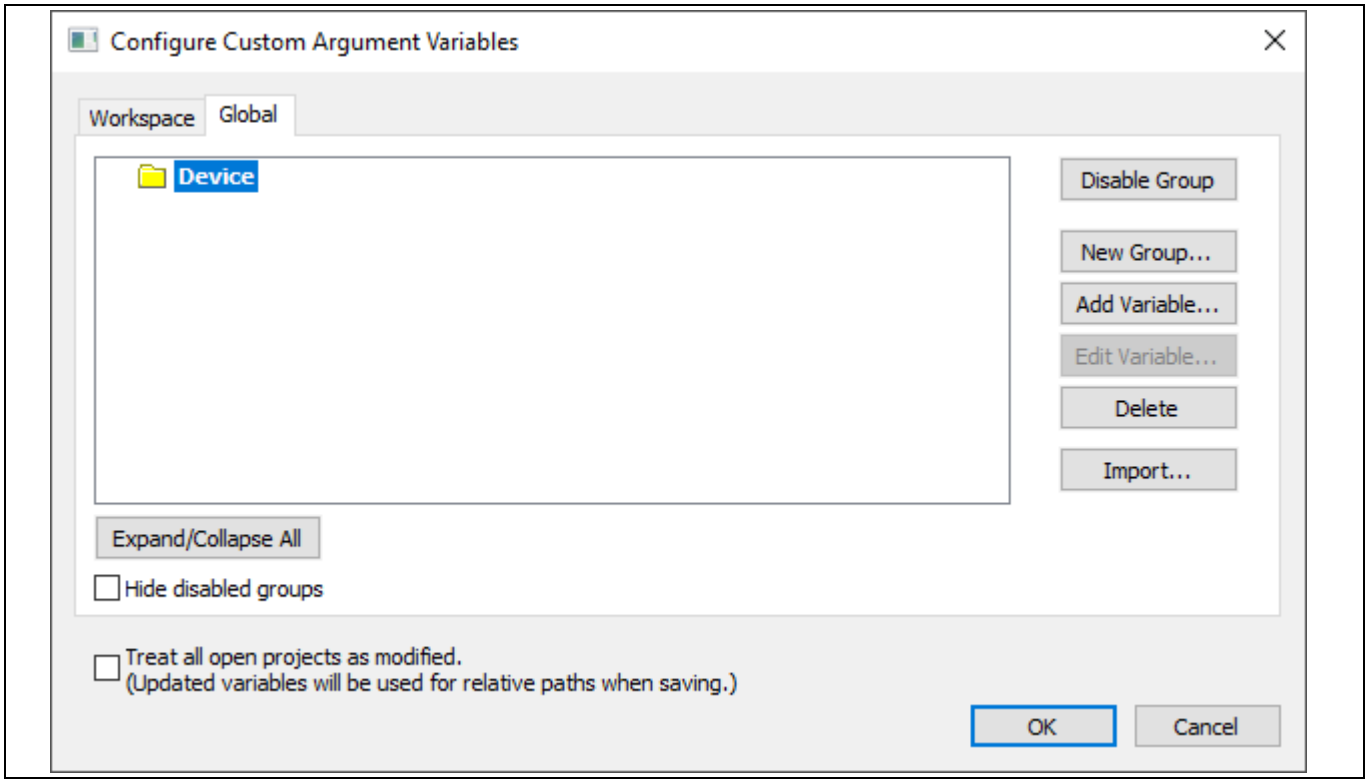

Figure 38 New group added in the Configure Custom Argument Variables window

Select ***Add Variable...*** to add a variable and define its value. Here the name is D as defined in the arguments when adding the Config Wizard as new tool. The value is the device name that you are currently using.

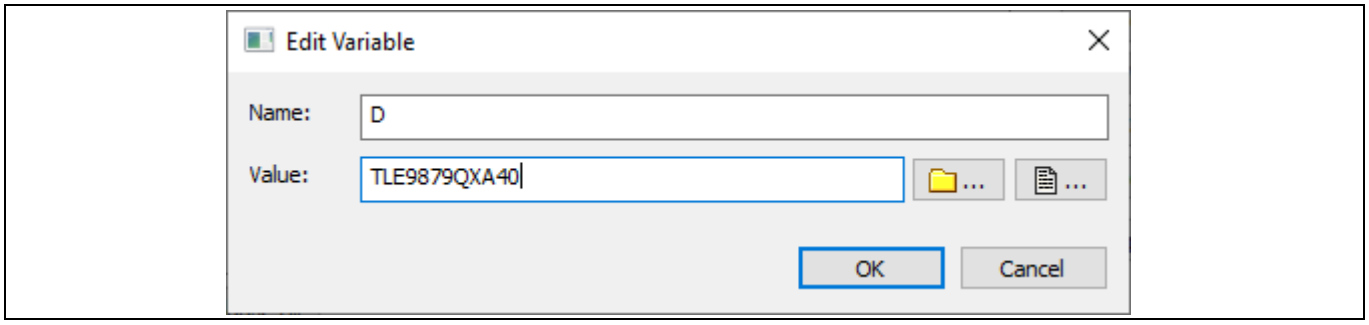

Figure 39 Window Add Variable

Click ***OK*** to save this variable, which then appears in the window Configure Custom Argument Variables. Click ***OK*** to save and close this window.

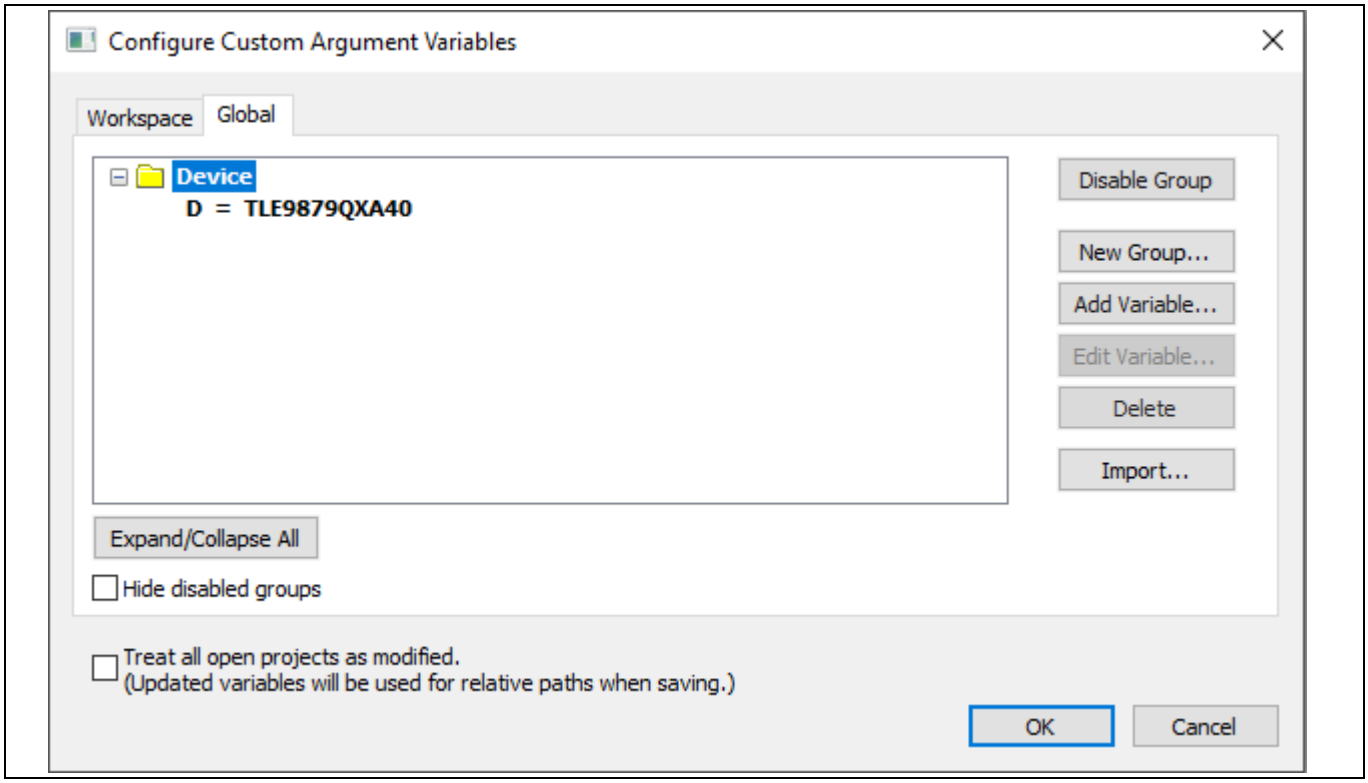

Figure 40 New argument variable in the window Configure Custom Argument Variables

3.6 How to use the Config Wizard for MOTIX™ MCU from the command line

The Config Wizard can be started with several command line options that are enumerated in the following:

ConfigWizard [<Filename.xml|icwp>] [-g] [-q] [-s] [-b<Batchfile>] [-c<Outfile>] [-o<Outfile>] [-?]

Where:

- -g: Generate header files
- -q: Quiet mode (no popup window on errors)
- -s: Save project file, with correct hash, and generate header files
- -x: Terminate program after Keil integration
- -k: Enable Keil uVision integration
- -b<Batchfile>: Run the batchfile whose path/name is specified in <Bathfile>
- -c<Outfile>: Set console logging file (for output of console and, if in quiet mode, for message box output)
- -o<Outfile>: Set output logging file (for output of batchfile/scriptiong)
- -?: Help

4 Evalkits and Evalboards Topics

4.1 How to access documentation from Keil uVision

After creating a new project, opening an existing one or importing an example from the Pack Installer, select the tab ***Books***.

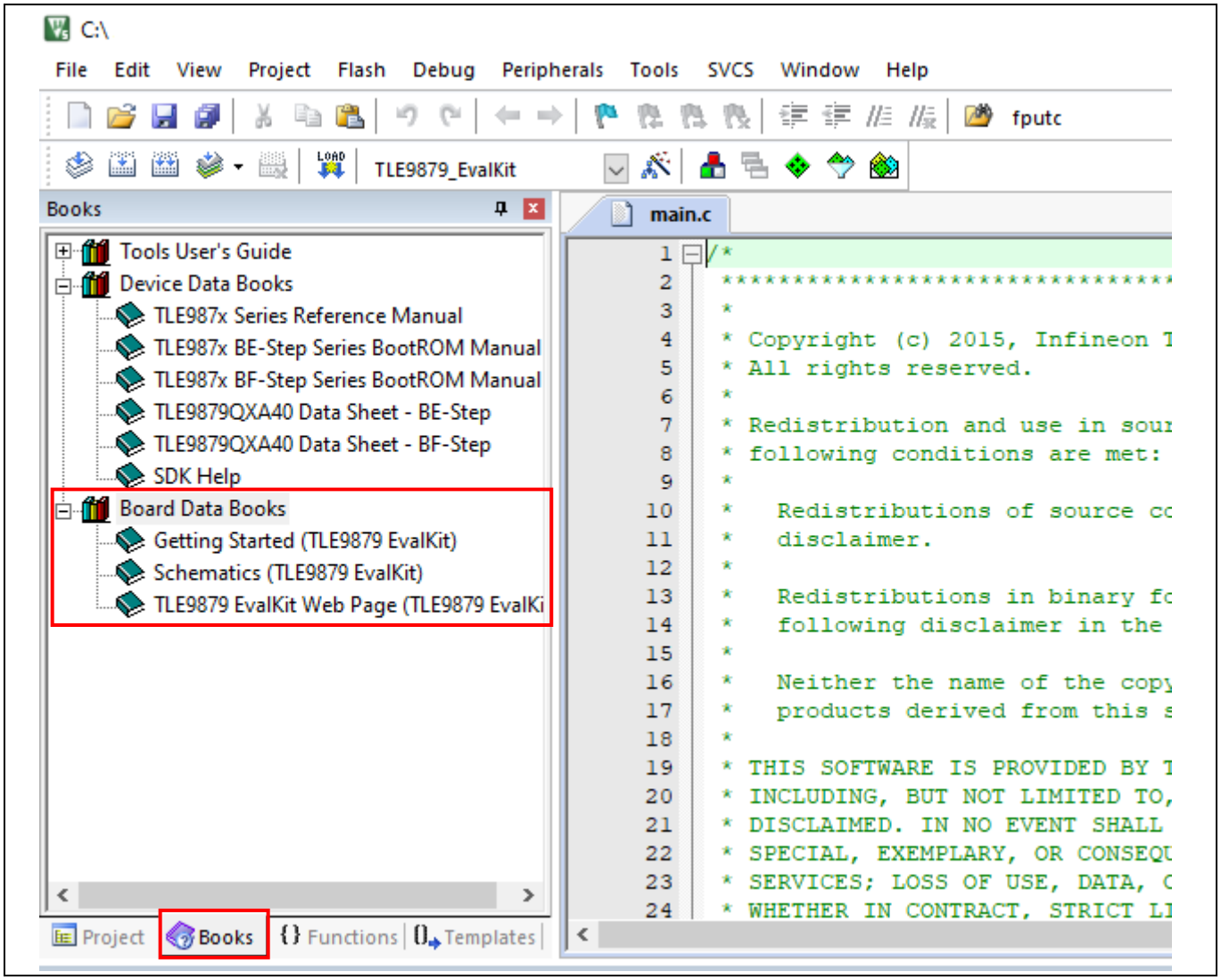

Figure 41 Main window in Keil uVision, tab Books

In the section ***Board Data Books***, you have an overview of the documentation available for the evalboards or evalkits that you can use with your project. The type of board refered to by the documentation is indicated in parenthesis.

4.2 How to access documentation from IAR Embedded Workbench

After creating a new project, opening an existing one or importing an example from the CMSIS Manager, open the rteconfig file of your project and select the tab ***Device***.

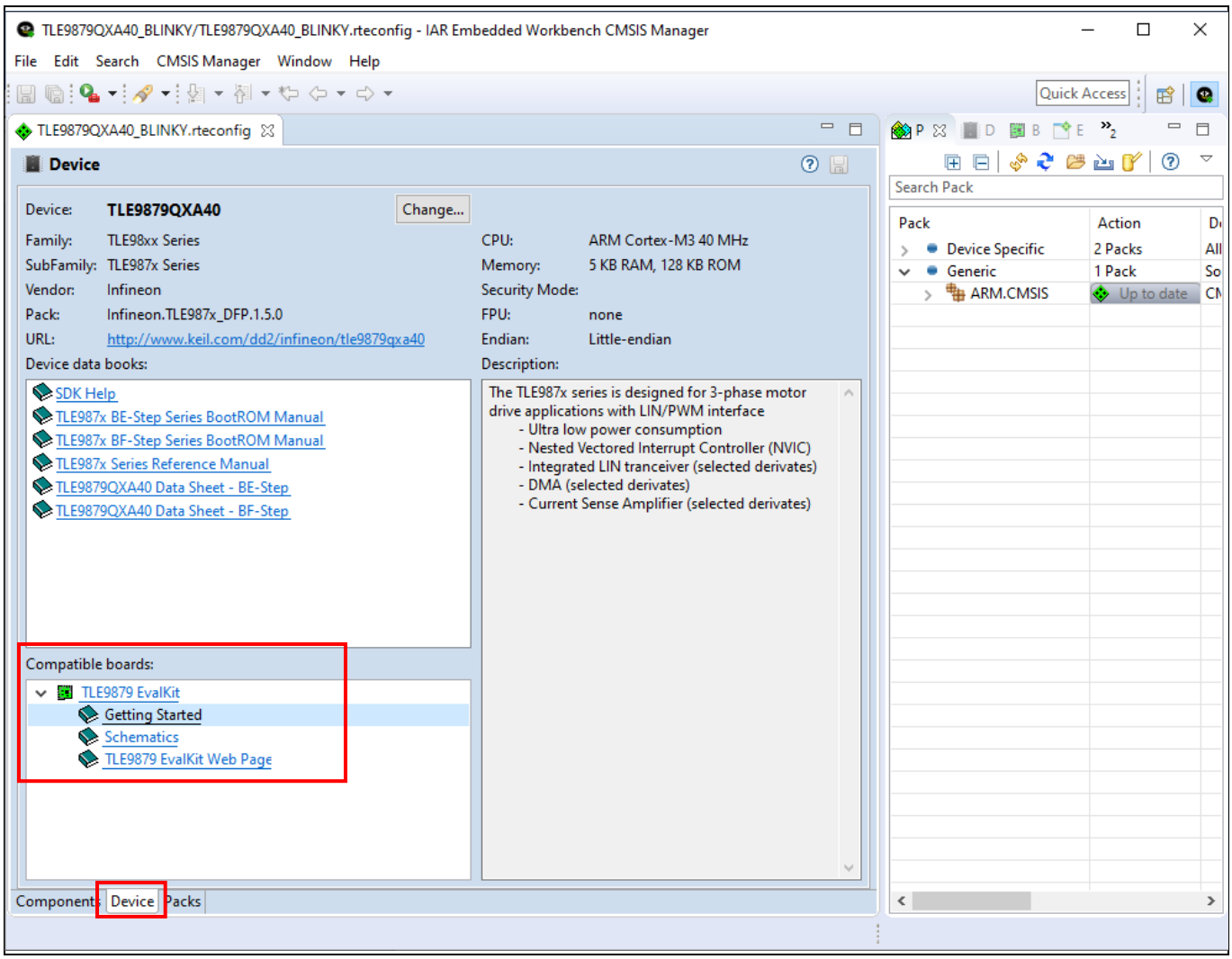

Figure 42 CMSIS Manager for IAR Embedded Workbench, tab Device

In the section ***Compatible boards***, you have an overview of the evalboards or evalkits that you can use with your project, as well as documentation for these boards: Getting started, Schematics, …

5 Software Topics

5.1 How to get the address of the instruction that triggered a HardFault

When there is a hardfault, it it often important to know which instruction caused the fault. This can be done as follows:

1. Modify the **HardFault_Handler** in the startup file as follows:

```
124  HardFault_Handler\
125                  PROC
126                  ;EXPORT  HardFault_Handler          [WEAK]
127                  ;B       .
128                  BX LR
129                  ENDP
```

Figure 43 Modified HardFault_Hander

The `BX LR` instruction jumps back to the instruction that caused the HardFault.

2. Set a breakpoint at the BX LR instruction and execute the code

When the HardFault Hander is executed, the **Stack Pointer** (green marked) shows the location of the stack frame (marked red).

In the ‘Memory 1’ window in Keil uVision the preferred setting here is ‘Unsigned > Long’ (appears by right-clicking in the Memory window).

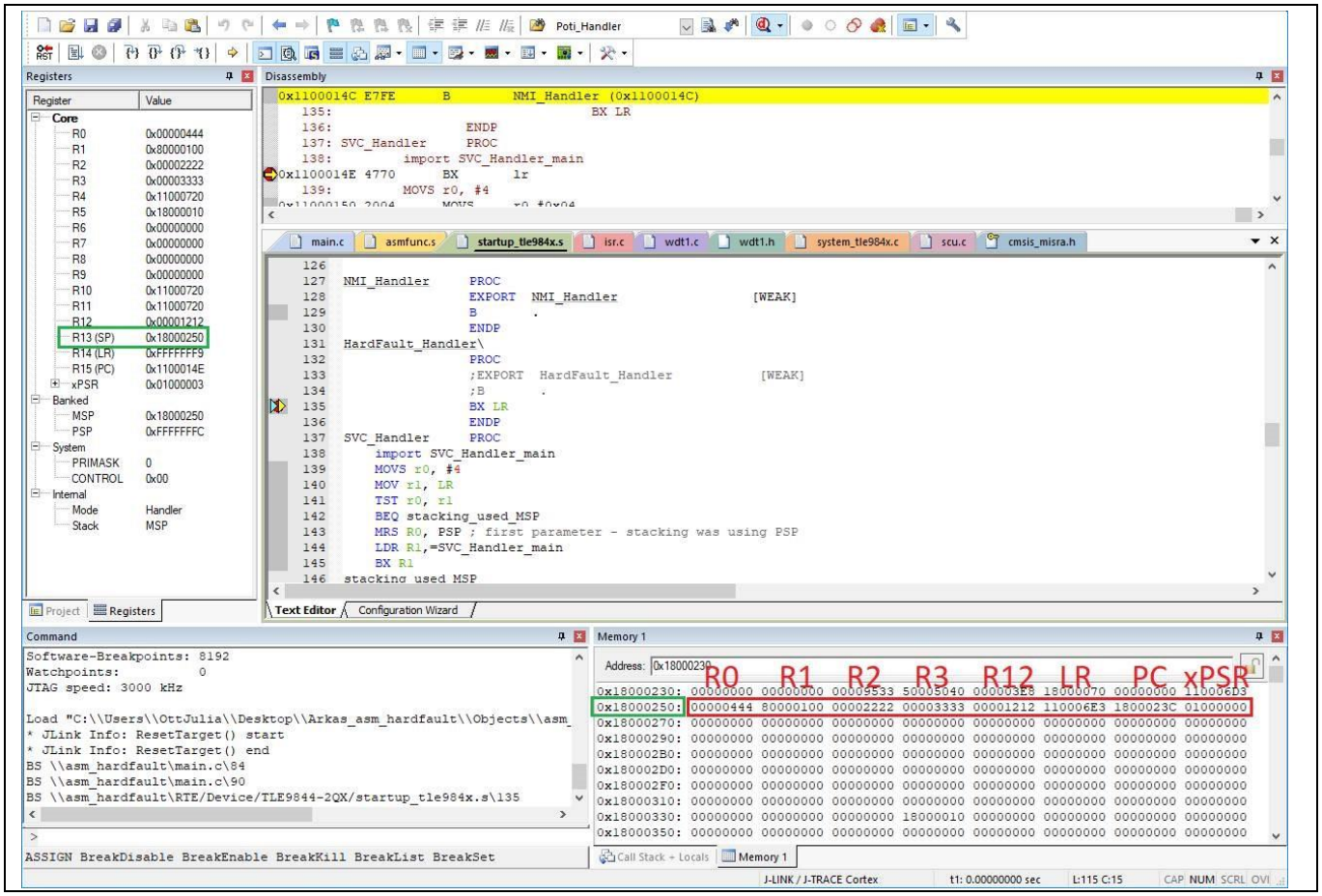

Figure 44 Stack pointer in register view and stack frame in memory view

3. Click into the **Disassembly** window and perform a single step. This leads to the instruction that caused the HardFault (red marked). It is the same as in the Stack frame above (PC).

Without clicking into the Disassembly window (step in C code), this instruction might not be shown.

In this test, the instruction is in RAM at address 0x1800023C.

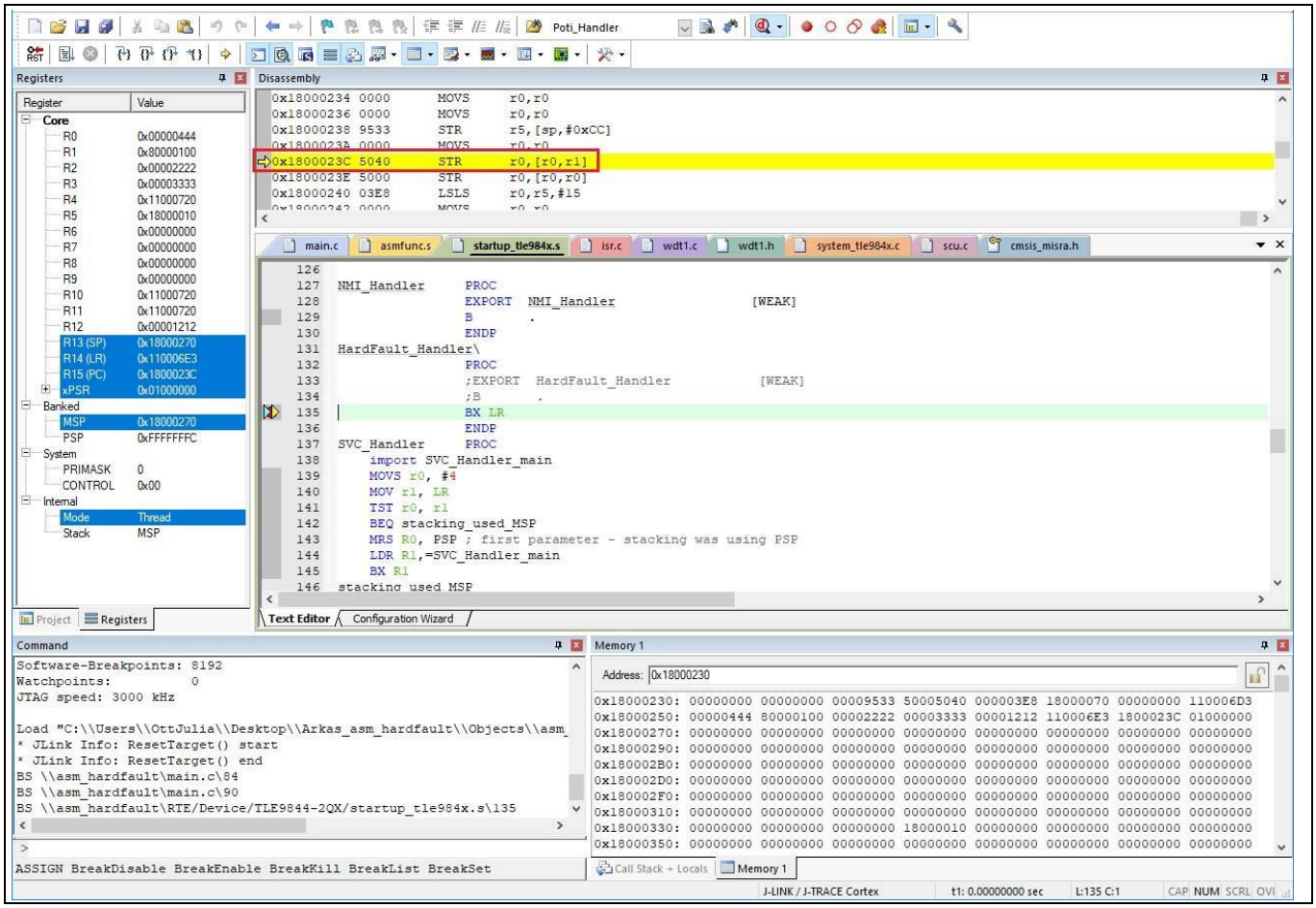

Figure 45 Instruction that caused the HardFault

Another step in the Disassembly leads back to the HardFault Hander, because the faulty instruction has been executed again.

Revision history

Document version	Date of release	Description of changes
V 1.0	2021-10-01	Initial version

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2021-10-01

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
Z8F80196569

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.