IAR Embedded Workbench for ModusToolbox user guide

About this document

A newer version of this document may be available on the web here.

Scope and purpose

ModusToolbox™ software is a set of tools and libraries that support device configuration and application development. These tools enable you to integrate our devices into your existing development methodology. This document provides information and instructions for using IAR Embedded Workbench with ModusToolbox™ software.

The general flow for working with an Infineon device in IAR Embedded Workbench includes:

- Download/install software
- Create ModusToolbox™ application using Project Creator
- Create project(s) in IAR Embedded Workbench
- Configure and Build projects
- Program the device
- Debug the application

Document conventions

- **Bold** - Emphasizes heading levels, column headings, menus and sub-menus.
- *Italics* - Denotes file names and paths.
- `Monospace` - Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets.
- **File > New** - Indicates that a cascading sub-menu opens when you select a menu item.

Reference documents

Refer to the following documents for more information as needed:

- ModusToolbox™ software installation guide – Provides information and instructions about installing the software on Windows, Linux, and macOS.
- ModusToolbox™ tools package user guide –Provides information about all the tools included with ModusToolbox™ tools package.
- Project Creator user guide – Provides specific information about the Project Creator tool.

Table of contents

About this document . . . 1

Table of contents . . . 2

1 Download/install software . . . 4

1.1 ModusToolbox™ software . . . 4

1.2 IAR Embedded Workbench (Windows only) . . . 4

1.3 J-Link . . . 4

2 Create/export application for IAR Embedded Workbench . . . 5

2.1 Create/export ModusToolbox™ application . . . 5

2.2 Create IAR workspace and project(s) . . . 7

3 Miscellaneous notes . . . 9

3.1 Supported debugger probes . . . 9

3.2 Configure applications with C++ files . . . 12

3.3 Build options, and prebuild/postbuild settings . . . 13

3.4 Enable softfp VFP mode . . . 14

3.5 RTOS settings . . . 14

3.6 Patched flashloaders . . . 15

3.7 Perform ETM/ITM trace . . . 15

3.8 Multi-core toolbar and CTI usage (I-jet and CMSIS-DAP only) . . . 15

4 PSOC™ Control application . . . 16

4.1 Configure and build projects . . . 16

4.2 Device with default policy . . . 17

4.3 Provisioned device . . . 17

4.4 Program and debug . . . 17

5 PSOC™ Edge E84 multi-core application . . . 20

5.1 Build configuration . . . 20

5.2 Debug configuration . . . 22

5.3 Target Programming . . . 28

5.4 Launch multi-core debugging with CMSIS-DAP/I-jet . . . 30

5.5 Launch multi-core debugging with J-Link . . . 31

6 PSOC™ 4 and PSOC™ 6 single-core application . . . 32

6.1 Configure and build . . . 32

6.2 Program/Debug with KitProg3/MiniProg4 (CMSIS-DAP) . . . 32

6.3 Program/Debug with J-Link . . . 34

6.4 Program external memory . . . 35

6.5 Erase PSOC™ 6 MCU with external memory enabled . . . 36

7 PSOC™ 6 and XMC7000/TRAVEO™ II multi-core applications . . . 38

7.1 Configure CM4/CM7 (slave) core(s) . . . 38

Table of contents

7.2 Configure and build CM0+ (master) core . . . 41
7.3 Configure CMSIS-DAP/I-jet debug options . . . 44
7.4 Launch multi-core debug session with CMSIS-DAP/I-jet . . . 45
7.5 Launch multi-core debug session with J-Link . . . 46

8 PSOC™ 64 secure single-core application . . . 47
8.1 Configure and build . . . 47
8.2 Program and debug . . . 47

9 AIROC™ CYW20829 application . . . 49

10 Make application changes and re-export . . . 50
10.1 Configure hardware . . . 50
10.2 Add/update libraries . . . 50
10.3 Opening ModusToolbox™ tools and configurators . . . 50
10.4 Re-export to update application . . . 52

Revision history . . . 53

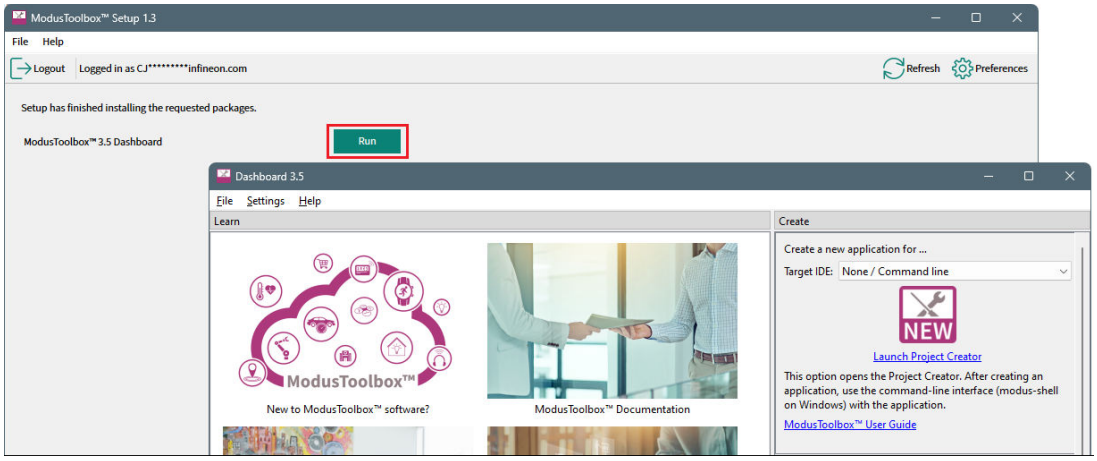
Disclaimer . . . 54

1 Download/install software

1.1 ModusToolbox™ software

Download the ModusToolbox™ Setup program from https://softwaretools.infineon.com/tools/com.ifx.tb.tool.modustoolboxsetup. Refer to the instructions in the ModusToolbox™ software installation guide for how to install the necessary ModusToolbox™ tools and packages.

Once installation of all the tools is complete, click **Run** to launch the Dashboard.



1.2 IAR Embedded Workbench (Windows only)

We recommend and have tested IAR Embedded Workbench version 9.60.3 or later for PSOC™ Control C3 devices. This version can be used with all ModusToolbox™ supported devices.

The default installation location for the IAR compiler is *C:\iar\[version]\arm*. Set the `CY_COMPILER_IAR_DIR` environment variable to the correct installation path using forward slashes. Alternatively, you can set this variable in the *Makefile* for each application. Refer to the ModusToolbox™ tools package user guide for more details about build system variables.

1.3 J-Link

For J-Link debugging, download and install J-Link software:

https://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack

2 Create/export application for IAR Embedded Workbench

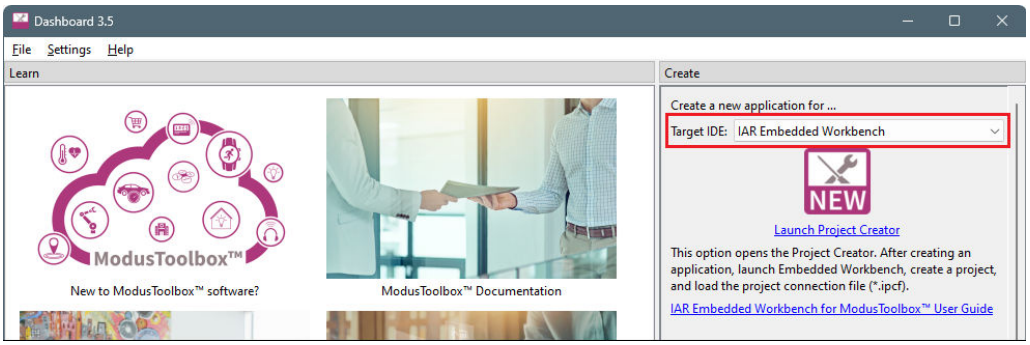
This section covers how to create or export a ModusToolbox™ application for use in the IAR Embedded Workbench IDE.

- Create/export ModusToolbox™ application
- Create IAR workspace and project(s)

2.1 Create/export ModusToolbox™ application

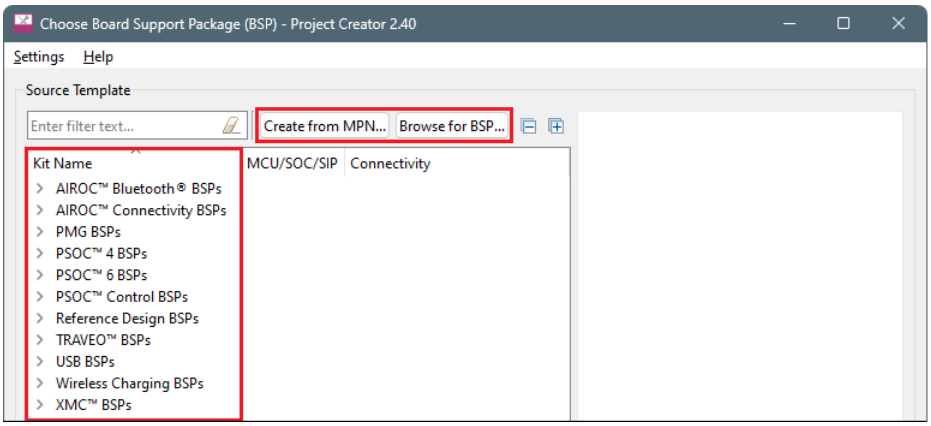
Create new application

1. Use the Dashboard to open the Project Creator tool and create a ModusToolbox™ application for IAR Embedded Workbench.



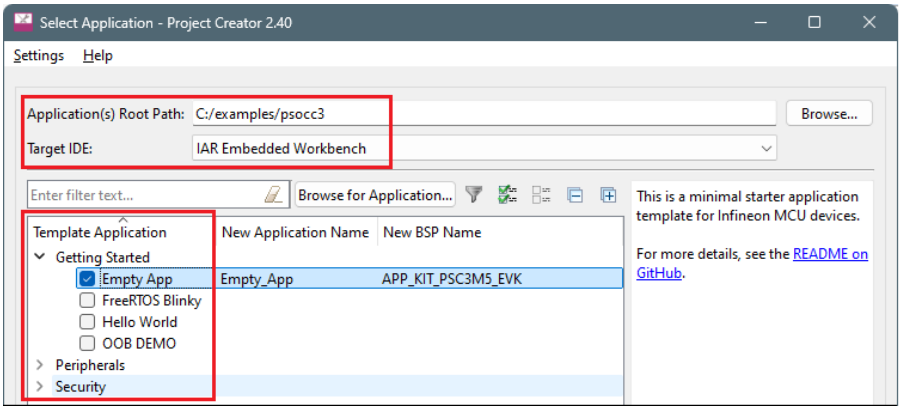
Refer to the Project Creator user guide for more details.

2. Select the BSP from the list or use one of the buttons to create a BSP from an MPN or select a BSP on disk.

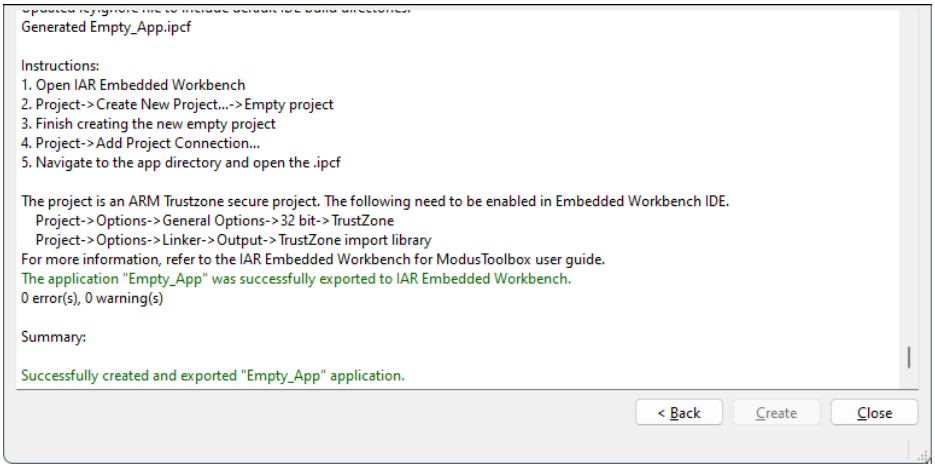


3. On the next page, select the location and the application to create. Notice Target IDE is already selected from the Dashboard.

2 Create/export application for IAR Embedded Workbench



4. Click **Create**. When the process completes see the messages in the console.



5. Click **Close**.

Export existing application

Instead of creating a new application, if you have a ModusToolbox™ application that was created for another IDE or for the command line, you can export that application to be used in IAR. Open a terminal window (modus-shell in Windows) and type the following:

```
make ewarm CY_IDE_PRJNAME=[project-name] TOOLCHAIN=IAR
```

Where `[project-name]` is the name of the root application/project folder.

***Note**: For applications that were created using core-make-3.0 or older, you must use the `make ewarm8` command instead.*

This sets the `TOOLCHAIN` to IAR in the Embedded Workbench configuration files but **not** in the ModusToolbox™ application's *Makefile*. Therefore, builds inside IAR Embedded Workbench will use the IAR toolchain, while builds in the ModusToolbox™ environment will continue to use the toolchain that was previously specified in the *Makefile*. You can edit the `TOOLCHAIN` variable if you also want ModusToolbox™ builds to use the IAR toolchain.

2.2 Create IAR workspace and project(s)

After creating or exporting an application, an IAR project connection file (.ipcf) appears in the ModusToolbox™ application/project directory. This an XML file that contains the hierarchy of all the files and directories from the original ModusToolbox™ application. For example: *Hello_World.ipcf*. To use the ModusToolbox™ application with IAR, do the following:

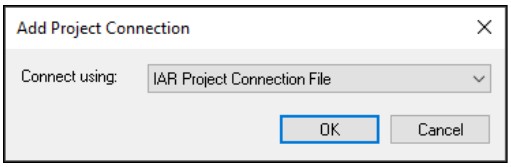
1. Launch IAR Embedded Workbench.

2. Select **File > New Workspace**, select **File > Save Workspace**, and enter a desired workspace name (*.eww) in the directory containing the ModusToolbox™ workspace.

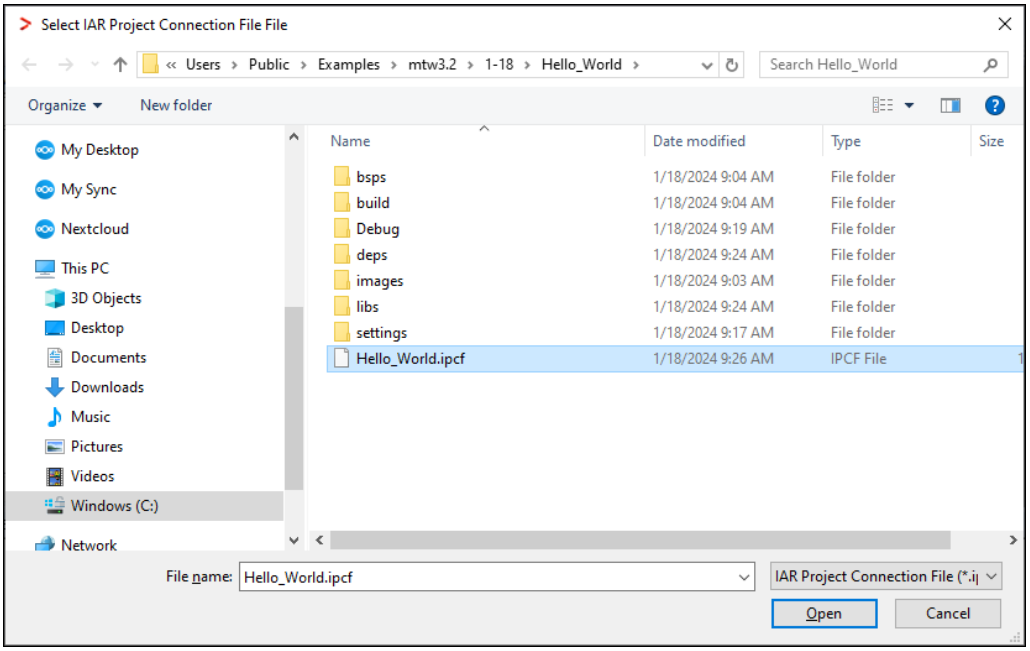
3. Select **Project > Create New Project > Empty project** and click **OK**.

4. Browse to the ModusToolbox™ project directory, enter a desired project name (*.ewp), and click **Save**.

5. Select **Project > Add Project Connection** and on the dialog ensure that "IAR Project Connection File" is selected; click **OK**.

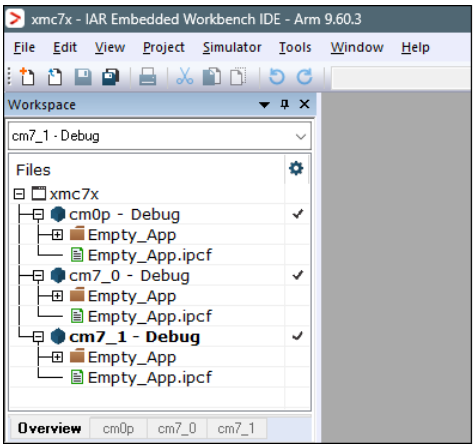


6. On the Select IAR Project Connection File dialog, select the *.ipcf* file and click **Open**:



7. If your application has multiple cores/projects, repeat the process to create an IAR project for each core and add the *.ipcf* file for each as well. For example:

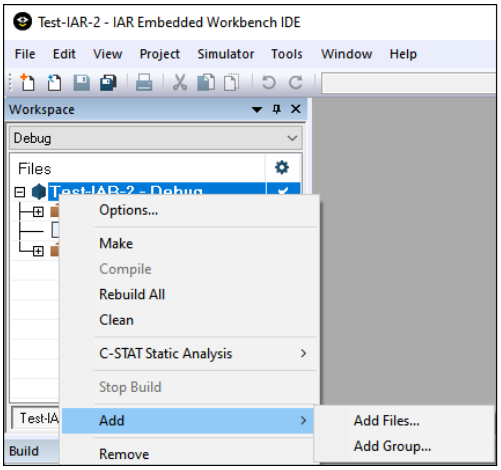
2 Create/export application for IAR Embedded Workbench



8. Once all projects have been created and connected, make sure to save the workspace and projects.

Note: *If you subsequently add more files and libraries in the ModusToolbox™ environment, you need to run "make ewarm" again to update the .ipcf file in the IAR Embedded Workbench project. For example:* `make ewarm TOOLCHAIN=IAR CY_IDE_PRJNAME=Hello_World`

- *If you don't use the* `CY_IDE_PRJNAME` *option, the generated ipcf file may revert to the original code example name, and it will not update your application.*
- *If you don't care about staying connected to the ModusToolbox™ tools that generate the project files, you can delete the .ipcf file from the workspace and restart IAR Embedded Workbench.*
- *If you want to make changes in IAR Embedded Workbench, you need to do that at the workspace level using the* ***Add*** *option. These additions will not be included in the .ipcf file.*



Next steps

In the simplest cases, the .ipcf file contains all required build settings, so configuring the build settings is generally not required. However, there are several cases and different devices for which configuring build settings is required, such as using TrustZone mode, post-build image processing like combining several additional images, or signing images or remaps to another memory region. Refer to the applicable device section for steps to configure build settings, build the application, then program the device if it is attached, and debug the application.

3 Miscellaneous notes

This section covers general instructions that may or may not pertain to your device and application.

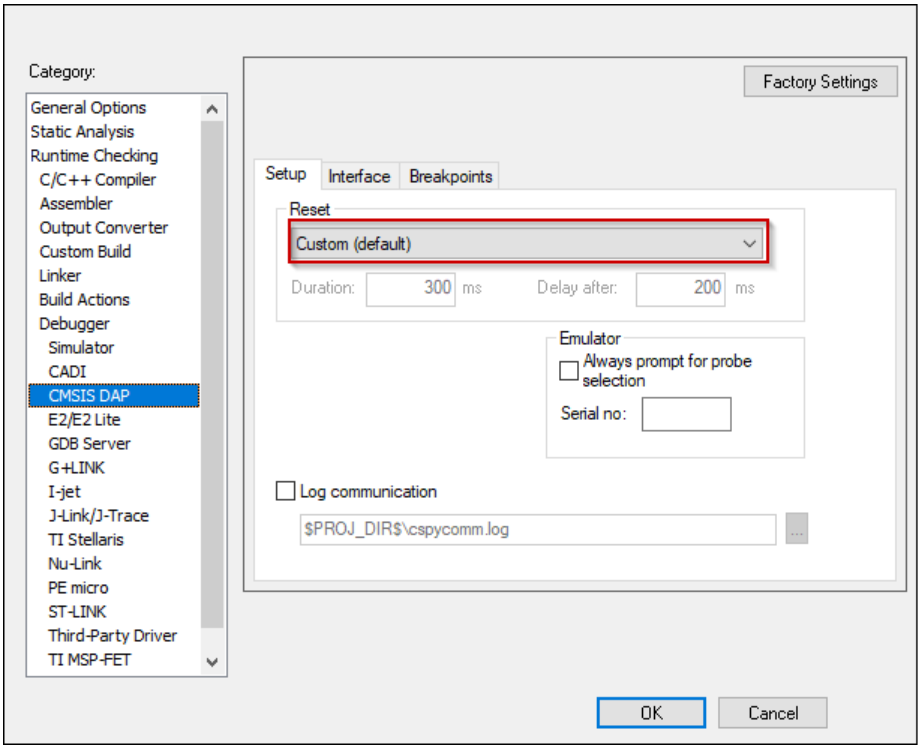
- Supported debugger probes
- Configure applications with C++ files
- Build options, and prebuild/postbuild settings
- RTOS settings
- Patched flashloaders
- Perform ETM/ITM trace
- Multi-core toolbar and CTI usage (I-jet and CMSIS-DAP only)

3.1 Supported debugger probes

- KitProg3 onboard programmer / MiniProg4 (CMSIS-DAP)
- JTAG (PSOC™ Control and PSOC™ Edge devices currently do not support the JTAG interface with CMSIS-DAP debugger.)
- IAR I-jet
- J-Link

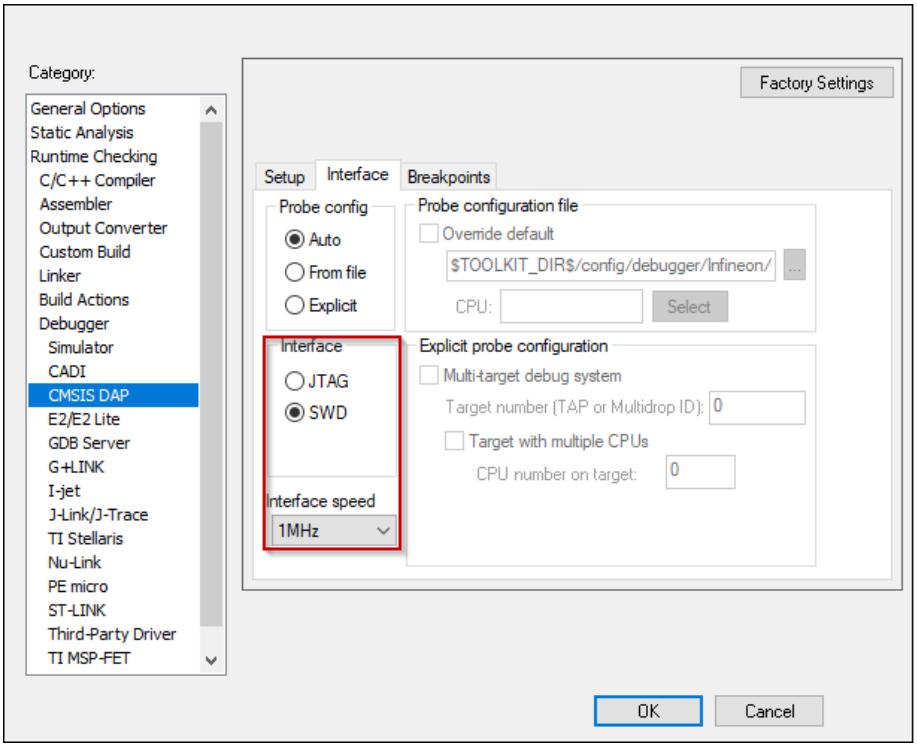
CMSIS-DAP

1. Switch to the **Setup** tab and select "Custom" from the **Reset** pull-down menu.



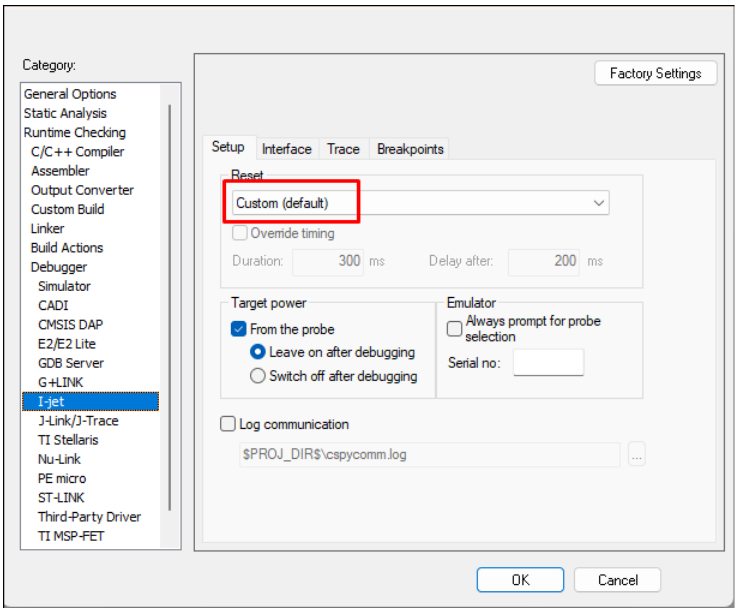
2. Switch to the **Interface** tab, and select **Interface** SWD, and set the **Interface speed** to 1MHZ.

3 Miscellaneous notes



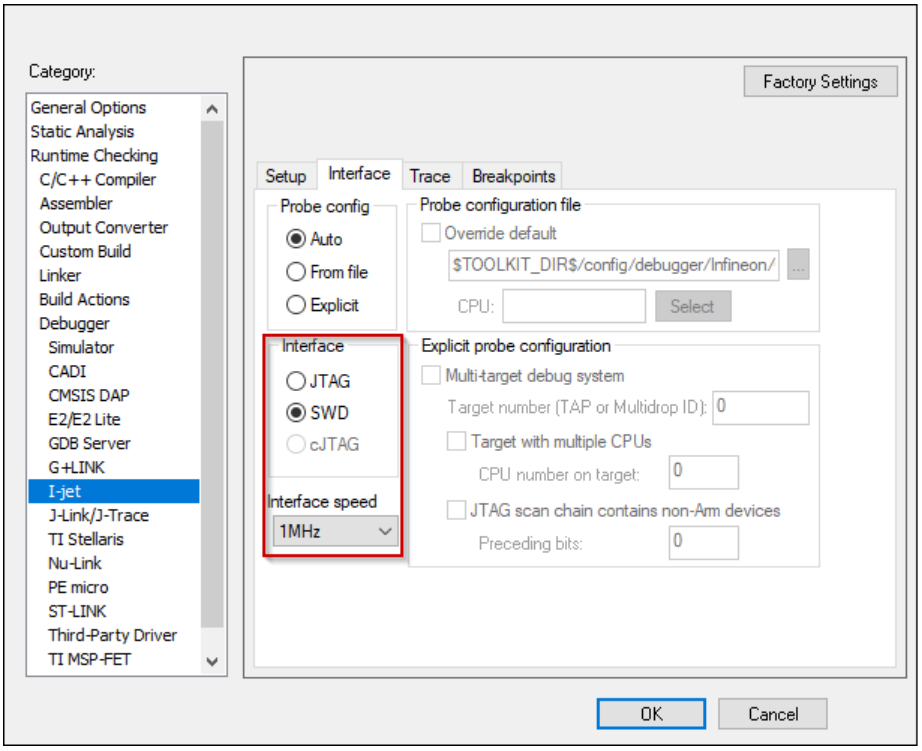
I-jet

1. Switch to the **Setup** tab and select "Custom" from the **Reset** pull-down menu.



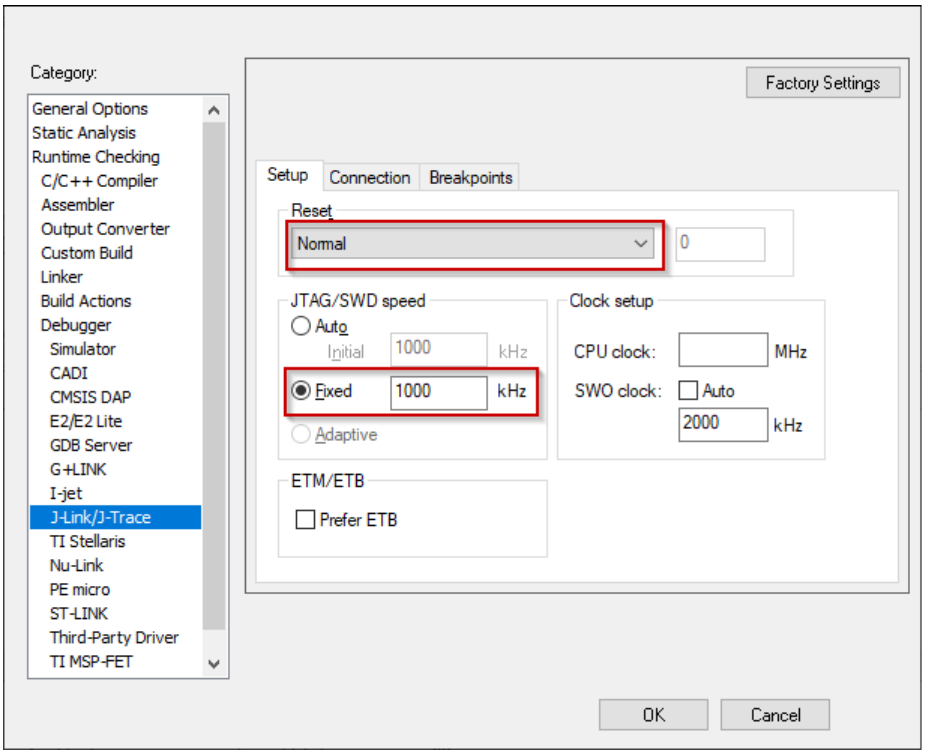
2. Switch to the **Interface** tab, and select **Interface** JTAG or SWD, and set the **Interface speed** to 1MHZ.

3 Miscellaneous notes

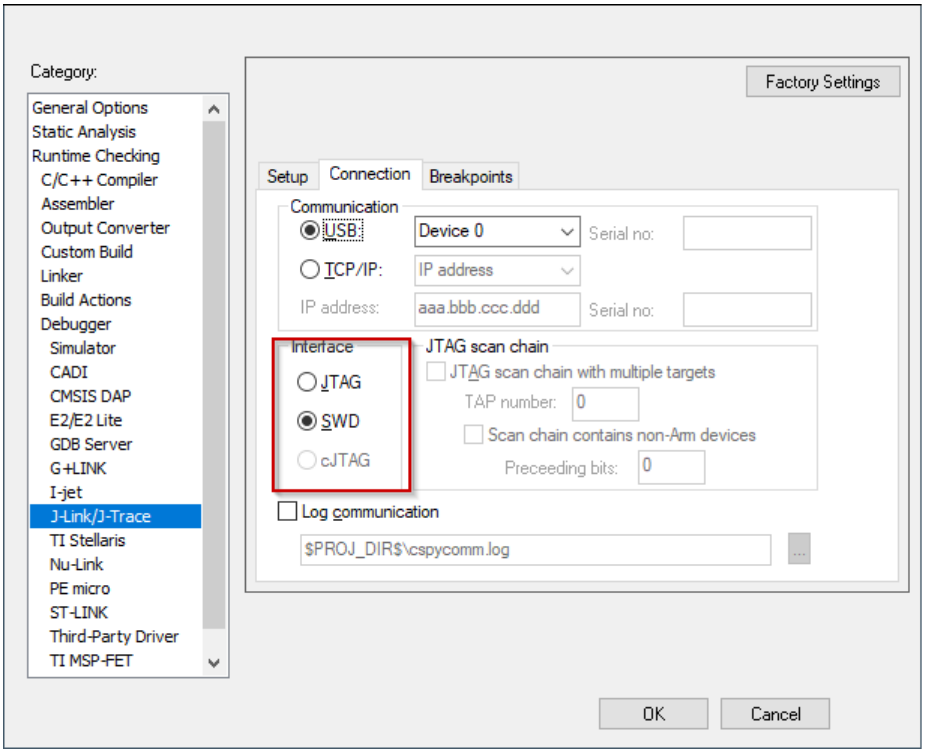


J-Link

1. Switch to the **Setup** tab and select "Normal" from the **Reset** pull-down menu and set the **JTAG/SWD** speed to **Fixed** 1000 KHz.

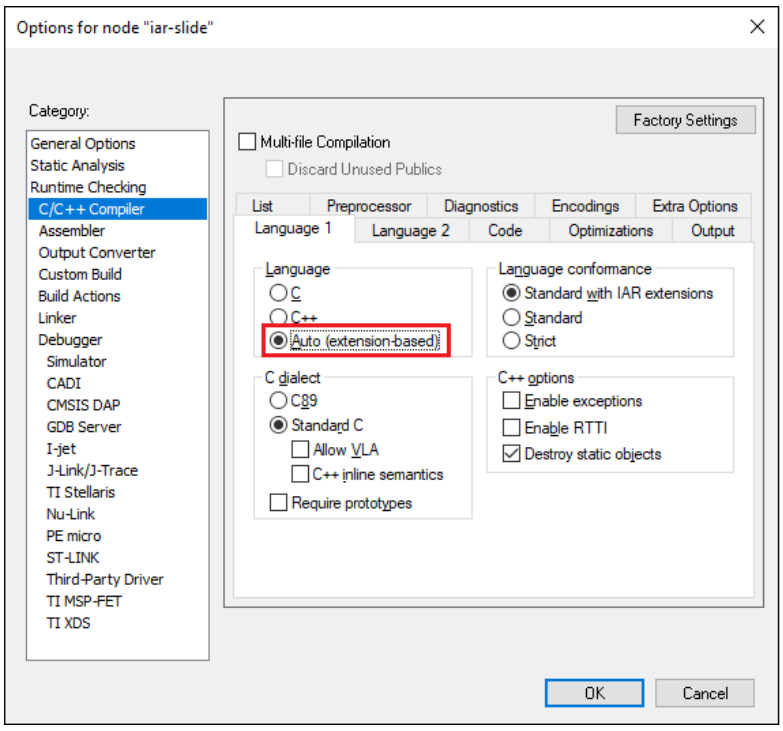


2. Switch to the **Connection** tab, and select **Interface** JTAG or SWD.



3.2 Configure applications with C++ files

In cases where your application includes C++ files, you need to configure the C/C++ Compiler option. Right-click on the project and select **Options > C/C++ Compiler > Language 1**, select **Auto (extension-based)**, and click **OK**.



3.3 Build options, and prebuild/postbuild settings

To set various build options, use the sections on the IAR Embedded Workbench Project Options dialog where you can enter command-line options:

- **C / C++ Compiler > Extra Options**
- **Assembler > Extra Options**
- **Linker > Extra Options**

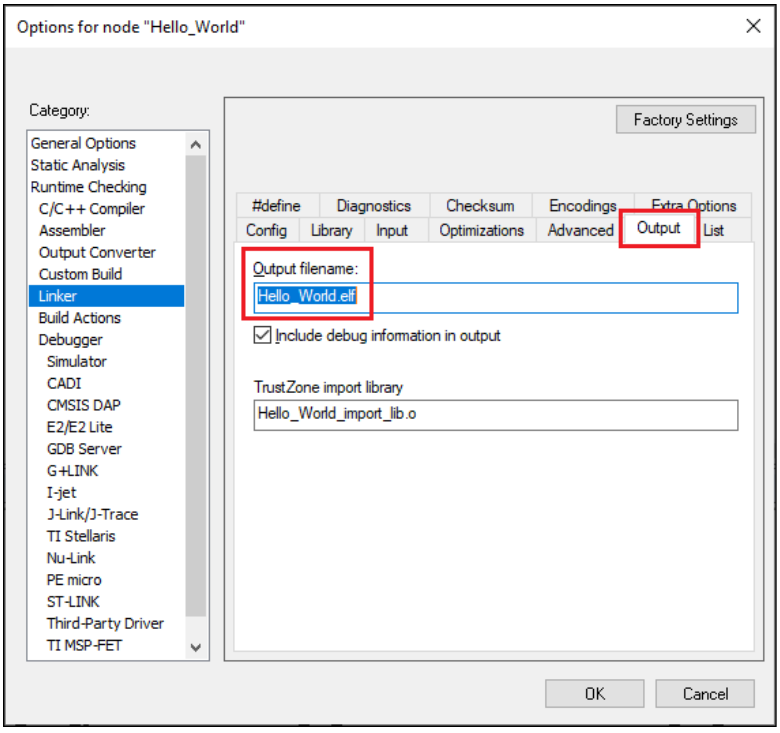
Prebuild and postbuild steps from a ModusToolbox application are not included in the export process by default for most devices at this time. If your original application has these steps, you must include them in the IAR Embedded Workbench manually as follows:

1. In IAR Embedded Workbench, open the Options dialog, go to the **Build Actions** category.

2. Click the **New** button to open the New Build Action dialog.

3. On the New Build Action dialog, enter a command-line argument, output and/or input files, the working directory, and the build order.

4. If using a postbuild script that works with .elf files, switch to the **Linker > Output** tab and change the file extension from ".out" to ".elf" in the **Output filename** field:



5. Click **OK** to close the Options dialog.
   Refer to the IAR Embedded Workbench Help for more details.

6. On the IAR main menu, select **Project > Make** to build the application. You should see the following:

```
Hello_World.elf

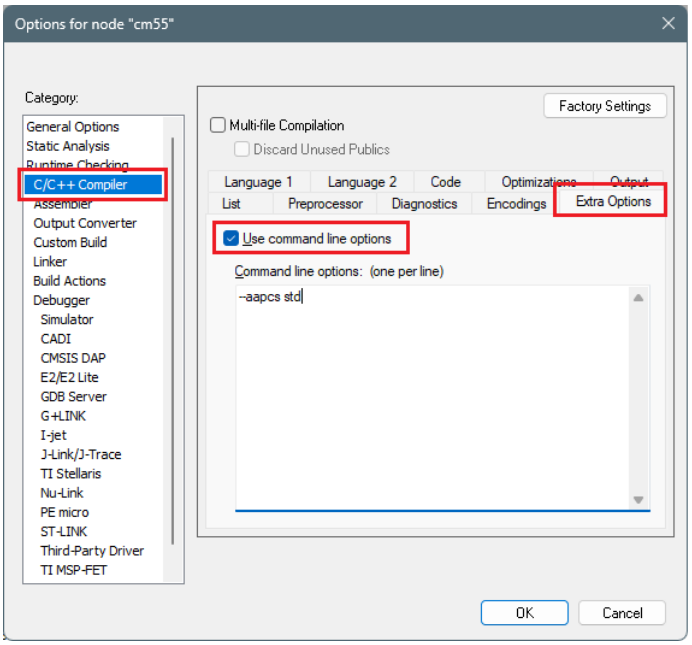
Total number of errors: 0
Total number of warnings: 0
Resolving dependencies...
Build succeeded
```

3.4 Enable softfp VFP mode

If your application has been set to use softfp, then some libraries may export softfp variant of libraries that are not compatible with the default IAR Embedded Workbench hardfp mode. You must enable softfp VFP mode as follows:

1. Open the Options dialog for you project/application.
2. Go to **C/C++ Compiler > Extra Options**, and enable **Use command line options**.
3. Enter the following in the entry box, and click **OK**:

```
--aapcs std
```



For more information, refer to https://wwwfiles.iar.com/arm/webic/doc/EWARM_DevelopmentGuide.ENU.pdf.

3.5 RTOS settings

If your application includes an RTOS, you will need to update compiler and linker settings in IAR Embedded Workbench. This is needed to make the C/C++ library implementation thread safe. This process is described in the IAR C/C++ Development Guide

1. Open **Project > Options > General Options > Library Configuration**.

2. In the **Library** menu, select "Full".

3. Select the **Enable thread support in library** check box.

4. Click **OK**.

Once enabled, this will require implementations of functions that perform the locking that are provided by the clib-support library that we provide. https://github.com/Infineon/clib-support/blob/master/README.md

***Note**: This is not specific to Infineon devices but more about how the IAR compiler/linker and C library implementation work. All compilers have similar enabling needed*

3.6 Patched flashloaders

To enable support for different QSPI settings, the ModusToolbox™ QSPI Configurator patches flashloader files and stores them in the application directory. When exporting such applications to IAR Embedded Workbench, these patched flashloader files must be copied into the appropriate directory.

Copy the **.out* file located in the *[app-dir]\bsps\[Kit-Name]\config\GeneratedSource* directory.

Paste the flashloader file to the *[install-path]\arm\config\flashloader\Infineon\[device]* directory.

3.7 Perform ETM/ITM trace

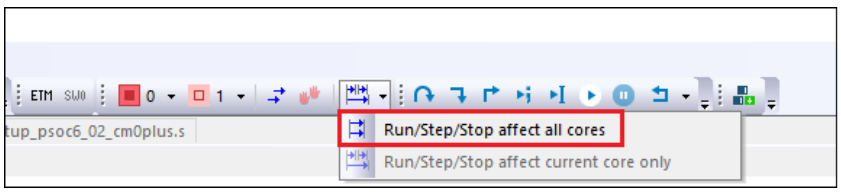
For PSOC™ Control and PSOC™ Edge devices, refer to application note AN242338 - Performing ETM and ITM trace on PSOC™ Control C3 and PSOC™ Edge E8 MCUs.

For PSOC™ 6 devices, refer to application note AN235279 - Performing ETM and ITM trace on PSOC™ 6 MCU.

3.8 Multi-core toolbar and CTI usage (I-jet and CMSIS-DAP only)

When multi-core debugging is established through I-jet or CMSIS-DAP drivers, a multi-core toolbar becomes available. It allows you to halt and resume all/single core(s) from within a single IDE instance.

Also, there is a feature called cross trigger interface (CTI). This allows you to immediately halt/resume one core when another core is halted/resumed. For example, this might be useful if you need to check what code is executing one of your cores when another hits a breakpoint. To use CTI, select the **Run/Step/Stop affect all cores** option available for multi-core applications:



4 PSOC™ Control application

Follow steps in Create/export application for IAR Embedded Workbench.

***Note**: Depending on your device, your application may be structured as follows:*

- ***Single-project application** – A stand-alone project targeting the secure software model. Follow the proj_cm33_s project configuration in this case.*
- ***Dual-project application** – A multi-core application consisting of two inner projects:*
  - *proj_cm33_s – TrustZone Secure project*
  - *proj_cm33_ns – TrustZone Non-Secure project*

For more details about TrustZone technology, refer to the Arm® website: https://www.arm.com/technologies/trustzone-for-cortex-m.

In addition to the TrustZone technology from Arm, PSOC™ Control C3 devices have various security life cycle stages (LCS). For more details about security, refer to Application Note AN240106 - Getting started with PSOC™ Control C3 security.

The following sections provide details about working with a device with the default out of the box policy versus a device that has been provisioned.

***Note**: By default, applications created from code examples default to TrustZone "Secure" mode. For details about PSOC™ Control C3 security, refer to Application Note AN240106 - Getting started with PSOC™ Control C3 security*

4.1 Configure and build projects

Single-core application

When you open the application, you need to check the TrustZone setting. Open the Options dialog, go to **General Options > 32-bit**, and Select:

- **FPU**: VFPv5 single precision
- **DSP Extension**: selected
- **Mode**: Secure

Close the options dialog, and select **Project > Make** to build the application.

Multi-core application

After the application has been created, configure and build the secure CM33 project first and then the non-secure CM33 project. The CM33 project may contain a non-secure-callable file, and it must be built before the non-secure CM33 project.

1. Select the secure CM33 project, open the Options dialog, and go to **General Options > 32-bit**.
   Select:
   - **FPU**: VFPv5 single precision
   - **DSP Extension**: selected
   - **Mode**: Secure
2. Switch to **Linker > Output** tab, and verify that the TrustZone import library is set.
3. Click **OK** to close the Options dialog.
4. Build using **Project > Make** on the main menu, and this generates the library file and hex file for secure cm33 project.
5. Select the non-secure CM33 project, open the Options dialog, and go to **General Options > 32-bit**.
   Select:
   - **FPU**: VFPv5 single precision

- **DSP Extension**: selected
- **Mode**: Non-secure

6. Switch to **Linker > Library** tab. Under **Additional libraries**, click the ellipsis button [**. . .**] and navigate to the TrustZone import library file in the main_cm33_s/Debug/Exe folder.
7. Click **OK** to close the Options dialog.
8. Build using **Project > Make** on the main menu.

4.2 Device with default policy

Devices are shipped with a default policy, so you can develop and debug your application repeatedly without any knowledge about security or code signing. There is nothing to configure before programming and debugging in this state. Go the Program and debug section.

4.3 Provisioned device

If you have provisioned the device, the hex file must be signed with the same key used during provisioning using the ModusToolbox™ Edge Protect Security Suite. Use the Basic Secure Application as your reference; it automatically configures signing steps in the application.

On the IAR main menu, select **Project > Make** to build the application.

Go the Program and debug section.

4.4 Program and debug

Select debugger options

Before programming and debugging, select the default debugger options for your desired probe. See Supported debugger probes.

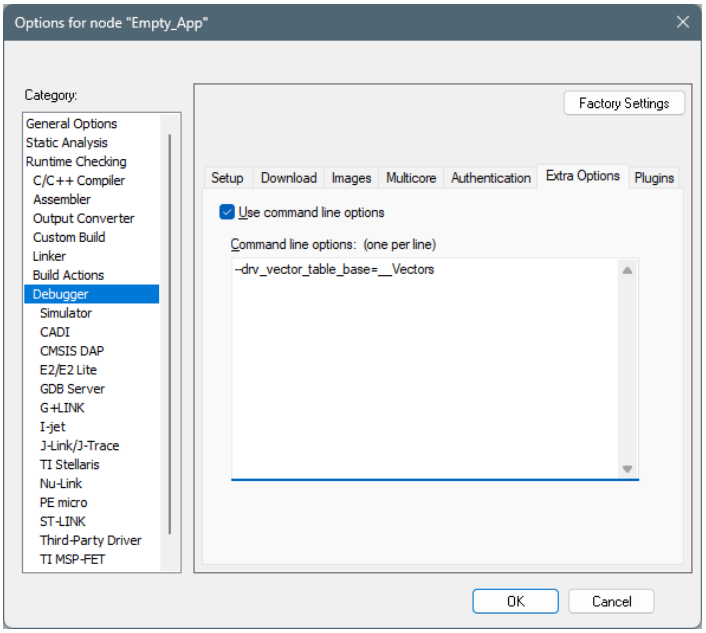
Fix vector table

Secure CM33 project

There is a known issue with an invalid for IAR name for the vector table in the C start-up code. To resolve it:

1. Select the secure CM3 project, open the Options dialog, and select the **Debugger** item under **Category**.
2. Select the **Extra Options** tab, select the **Use command line options** check box, and paste the following command-line option:

```
--drv_vector_table_base=__Vectors
```



3. Click **OK** to close the Options dialog.

Non-secure SM33 project

1. Open the .map file of the secure CM33 application (usually located in the main_cm33_s\Debug\List folder) and find the address in it that corresponds to the symbol `__Vectors`.

   ***Note**: The current step is a workaround for the following problem. IAR starts debugging based on the vector table (object named '__vector_table'). For a non-secure application, the object named '__vector_table' must point to the secure application's vector table. Otherwise, execution of the secure application will simply be skipped during debugging.*

2. Select the non-secure CM3 project, open the Options dialog, and select the **Debugger** item under **Category**.
3. Switch to the **Extra Options** tab, select the **Use command line options** check box, and paste the following command-line parameter, specifying the actual address found for `__Vectors`. symbol in the .map file.

   ```
   --drv_vector_table_base=0x12000000
   ```

   ***Note**: For MP4/I-jet you could paste the following line instead.*

   ```
   --drv_vector_table_base=__Vectors
   ```

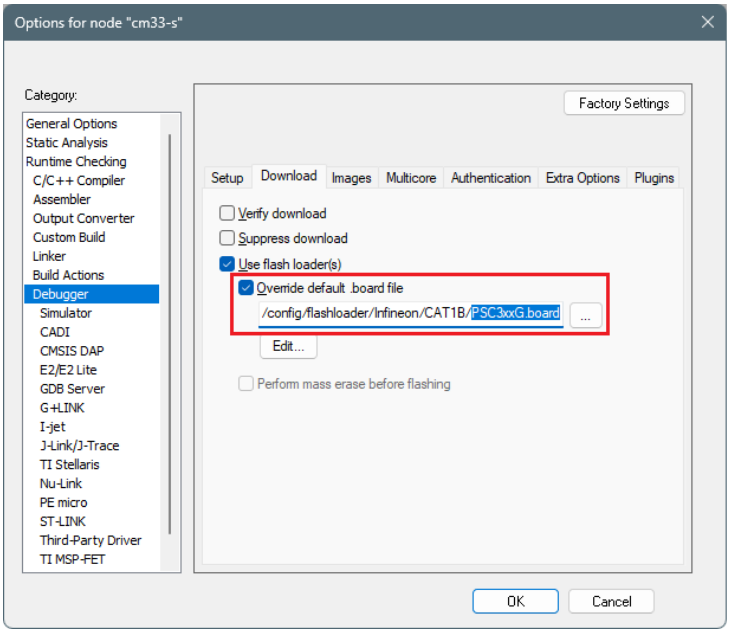
4. Click **OK** to close the Options dialog.

Use dual-bank mode

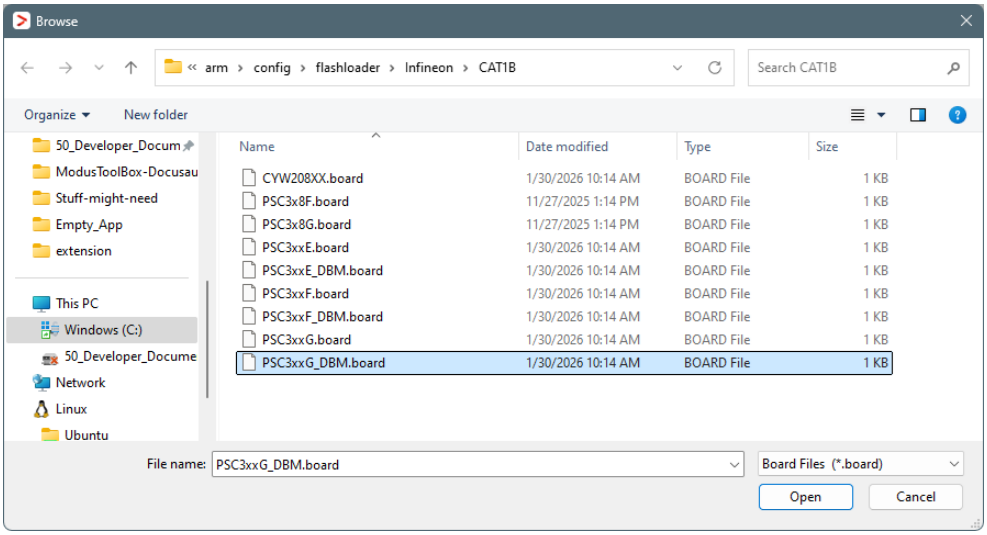
In order to use dual-bank mode, you must enable the non-default *.board file manually for a device. To do that:

1. Open the Options dialog and select the **Debugger** item under **Category**.
2. Select the **Download** tab and select the **Override default .board file** check box.

4 PSOC™ Control application



3. Identify the default .board file currently used for this project (for example, PSC3xxG.board).
   - Click the **Browse [...]** button, then navigate to and select the same .board file that also includes "DBM".



   - Click **Open**.
4. Click **OK** to close the Options dialog.

Program and debug

1. Select **Project > Build Target** to build the application.

2. Select **Project > Download > Download file...** and select the *[project_name.hex* file in *[application_folder]/build/last_config* (you might have to switch to **All Files (*.*)**).

   ***Note**: This programming method should only be used when the application has a signer combiner. In other cases, the **Download current application** button should be enough.*

3. Select **Project >Debug without Downloading**.

5 PSOC™ Edge E84 multi-core application

Follow steps in Create/export application for IAR Embedded Workbench.

Then, use the instructions in this section to configure, build, program, and debug a multi-core application for a PSOC™ Edge 84 device in IAR Embedded Workbench.

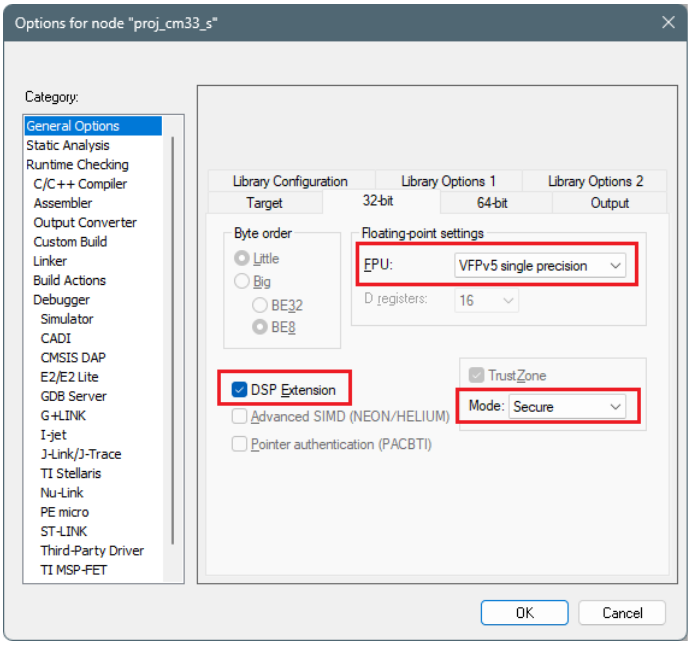
5.1 Build configuration

After all the core projects have been created, configure and build the secure CM33 project first and then the non-secure CM33 project. The CM33 project contains a non-secure-callable file, and it must be built before the non-secure CM33 project. There is nothing to configure for the CM55 core.

The project creation process creates and adds postbuild commands automatically. For the secure CM33 project, the postbuild command will sign and shift the hex file. For the non-secure CM33 project, the postbuild command will shift the hex file. For each project, the postbuild command will also generate a combined file if all projects have been built at least once.

5.1.1 Configure secure CM33 project

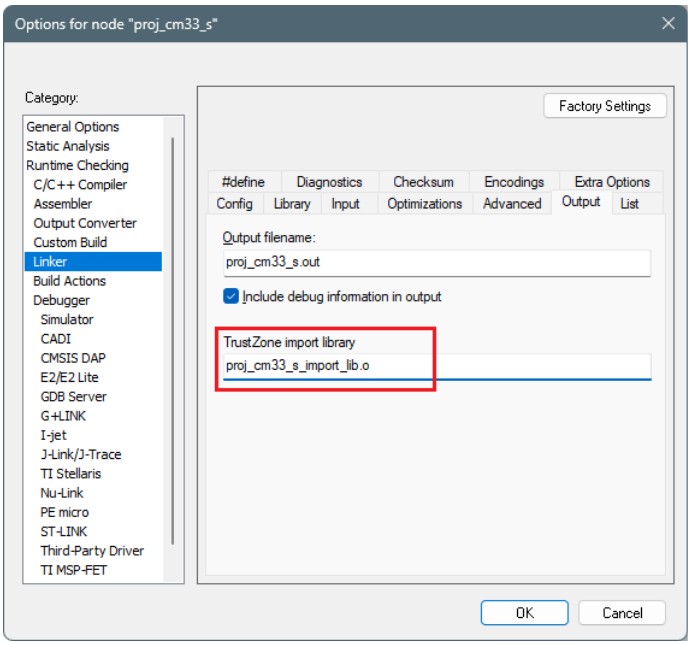
1. Select the non-secure CM33 project, open the Options dialog, and go to **General Options > 32-bit**.



   Select:

   - **FPU**: VFPv5 single precision
   - **DSP Extension**: selected
   - **Mode**: Secure

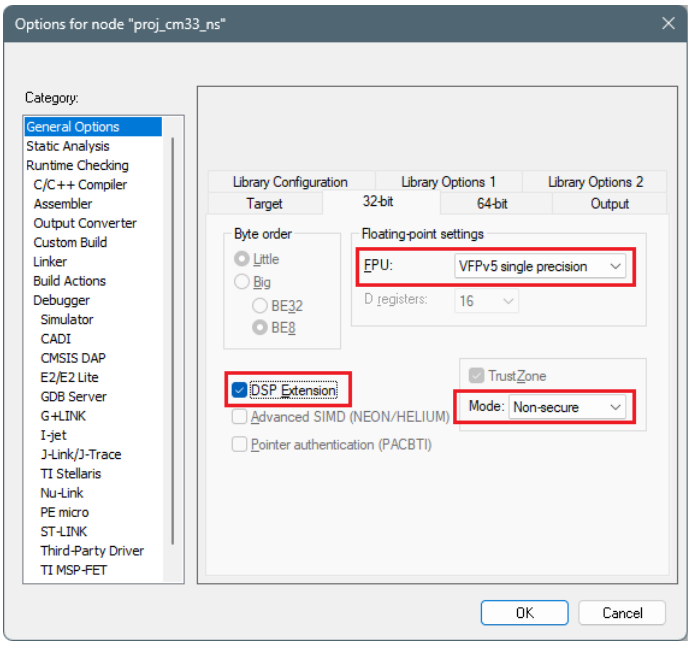
2. Switch to **Linker > Output** tab, and verify that the **TrustZone import library** is set.



3. Click **OK** to close the Options dialog.
4. Save the project and build it using **Project > Make** on the main menu, and this generates the library file.

5.1.2 Configure non-secure CM33 project

1. Select the non-secure CM33 project, open the Options dialog, and go to **General Options > 32-bit**.

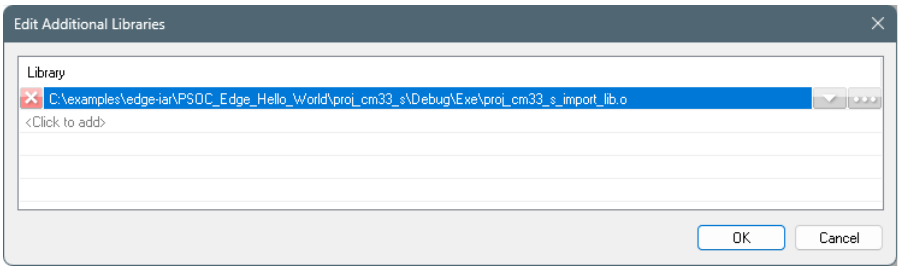


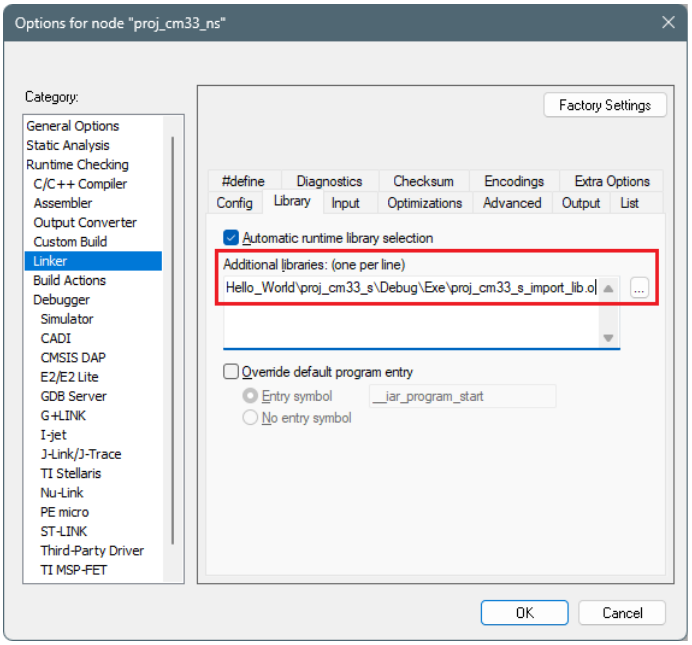
Select:

- **FPU**: VFPv5 single precision
- **DSP Extension**: selected
- **Mode**: Non-secure

2. Switch to **Linker > Library** tab. Under **Additional libraries**, click the ellipsis button **[. . .]** and navigate to the *lib.o* file in the secure cm33 project. For example (update the path and file name as needed for your system):

C:\examples\edge-iar\PSOC_Edge_Hello_World\proj_cm33_s\Debug\Exe\proj_cm33_s_import_lib.o





3. Click **OK** to close the Options dialog.

4. Save the project and build it using **Project > Make** on the main menu to include the *lib.o* file.

***Note**:* *You will likely see an error in the postbuild similar to "cm55.hex not found". This is expected because that project has not been built yet.*

5.1.3 Build CM55 project

There is nothing to configure for the CM55 project. Just build it using **Project > Make** on the main menu after you have already build the secure CM33 and non-secure CM33 projects, and this will generate a single combined image at *..\build\app_combined.hex*.

After all three projects have been built at least once, any time you build any of the projects, the combined hex file will be regenerated.

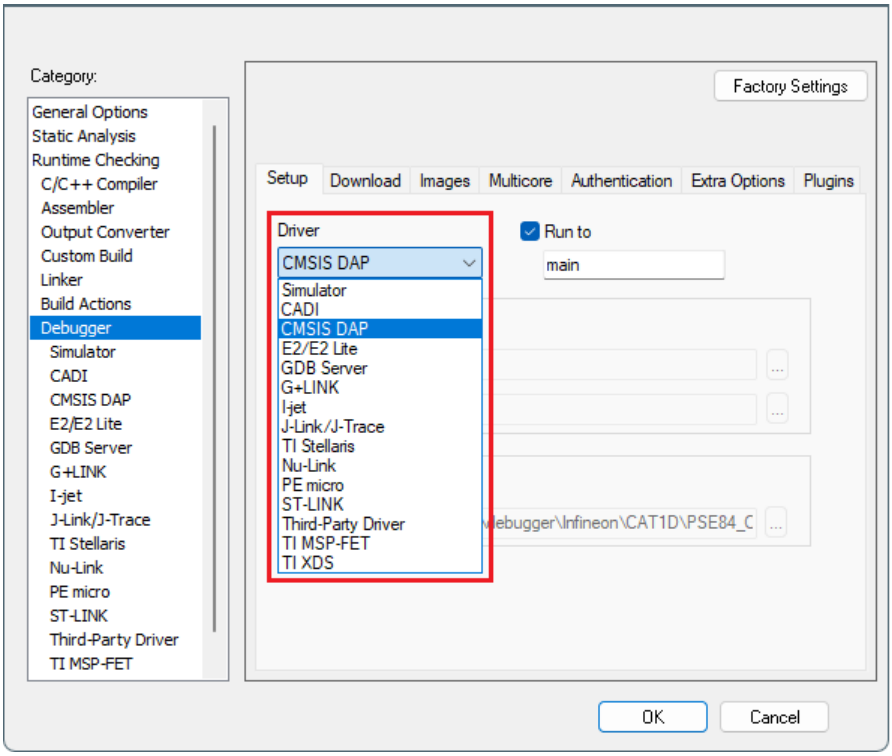
Refer to Debug configuration multi-core programming and debugging instructions.

5.2 Debug configuration

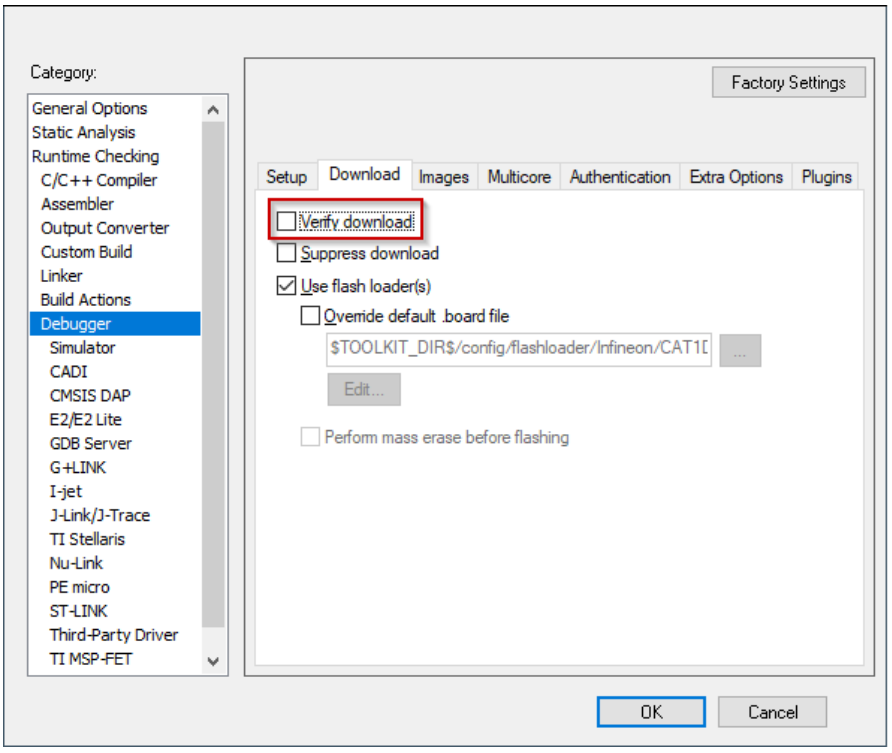
To launch a multi-core debug session, all projects within the workspace must be properly configured. In IAR there is a concept of 'master' and 'slave' projects. Configure the secure CM33 or non-secure CM33 project as the master project, and configure the CM55 as the slave project.

5.2.1 Secure CM33 project

1. Select the secure CM33 project, open the Options dialog, and go to the **Debugger** category. Select the **Setup** tab, and then select the applicable **Driver** (I-jet, CMSIS-DAP, J-Link):



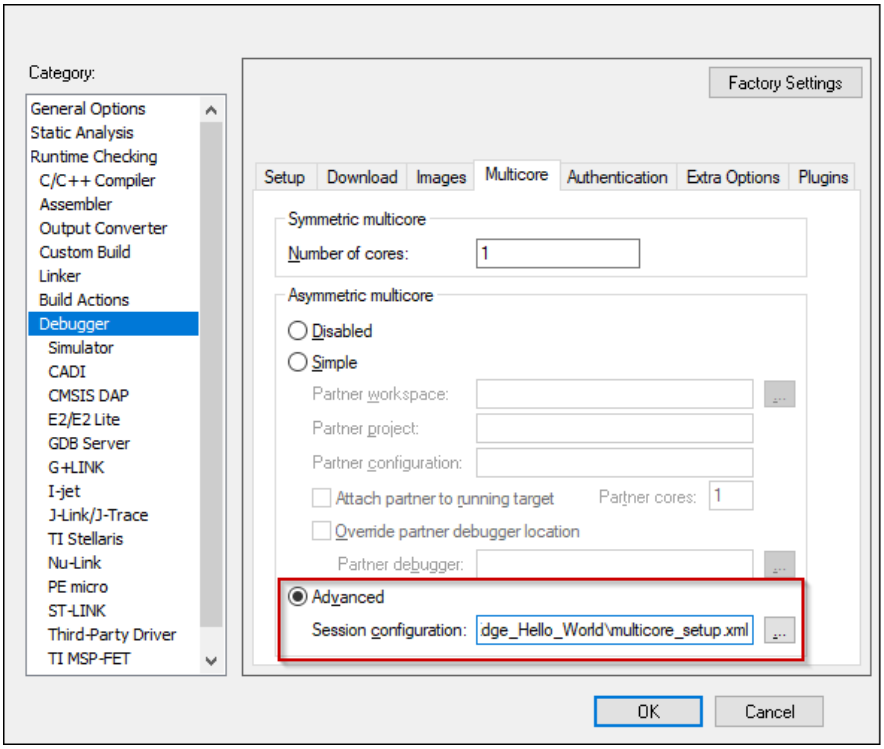
2. If any application (secure CM33, non-secure CM33, or CM55) needs to be programmed into external memory, switch to the **Download** tab and ensure the **Verify** check box is not selected.



3. If using CMSIS-DAP or I-jet, create a session configuration file. This is an xml file containing a projects list that will be launched in a multi-core debug session. The following shows an example for multi-core debug secure CM33 or non-secure CM33 and CM55 applications.

```
<?xml version="1.0" encoding="utf-8"?>
<sessionSetup>
    <partner>
        <name>Partner0</name>
        <workspace>$WS_PATH$</workspace>
        <project>$PROJ_PATH$</project>
        <config>Debug</config>
        <numberOfCores>1</numberOfCores>
    </partner>
    <partner>
        <name>Partner1</name>
        <workspace>$WS_PATH$</workspace>
        <project>cm55</project>
        <config>Debug</config>
        <numberOfCores>1</numberOfCores>
        <attachToRunningTarget>true</attachToRunningTarget>
    </partner>
</sessionSetup>
```

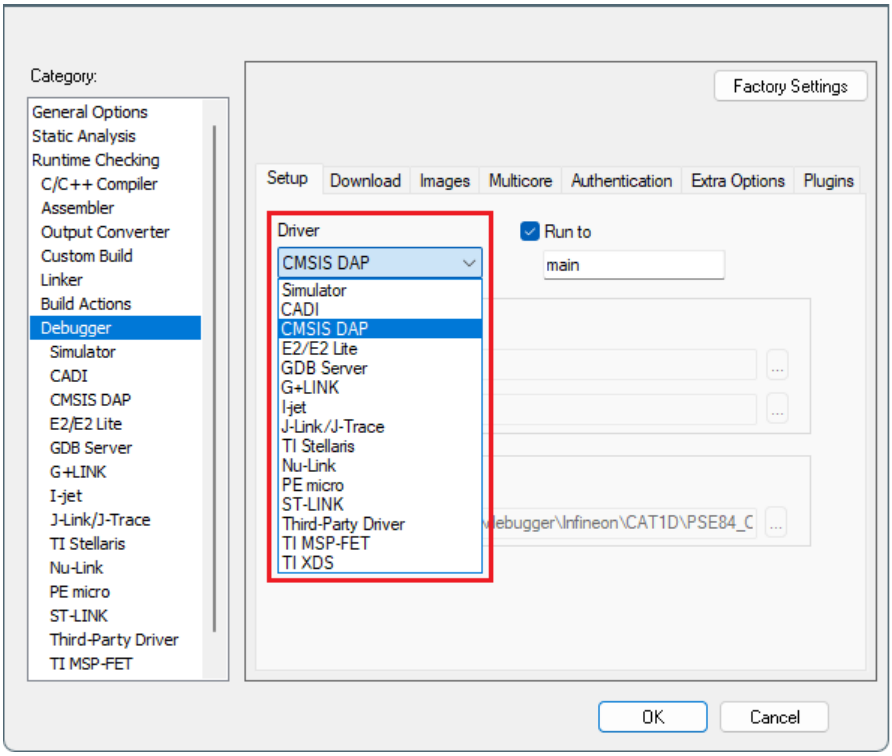
4. Switch to the **Multicore** tab, select the **Advanced** radio button, and specify a path to the session configuration file in the **Session configuration** field.



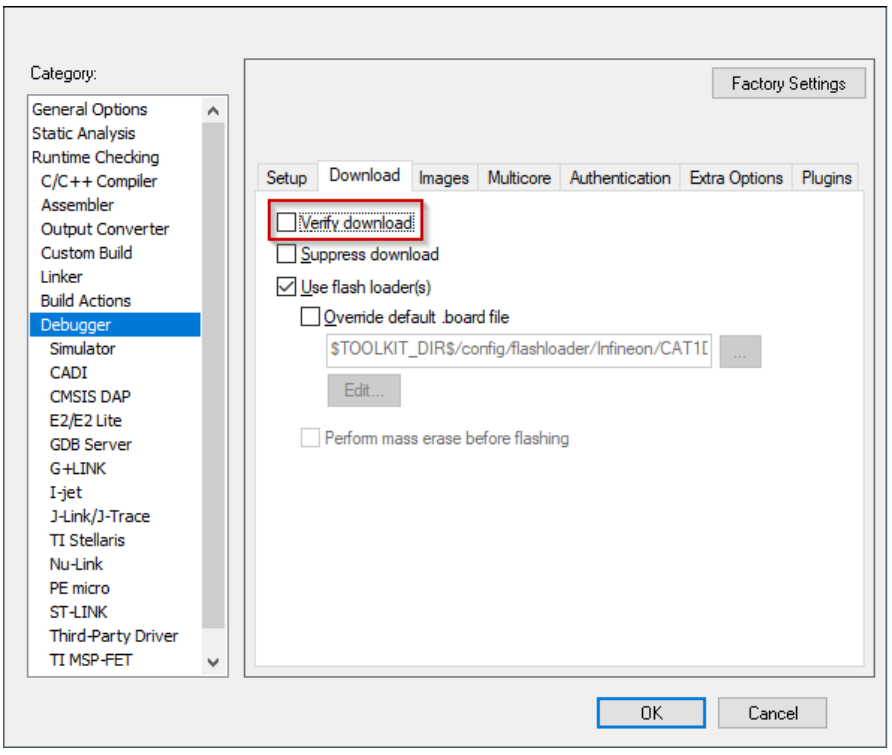
5. Select the driver for the **Debugger** category that you selected. See Supported debugger probes.
6. Click **OK** to close the Options dialog.

5.2.2 Non-secure CM33 project

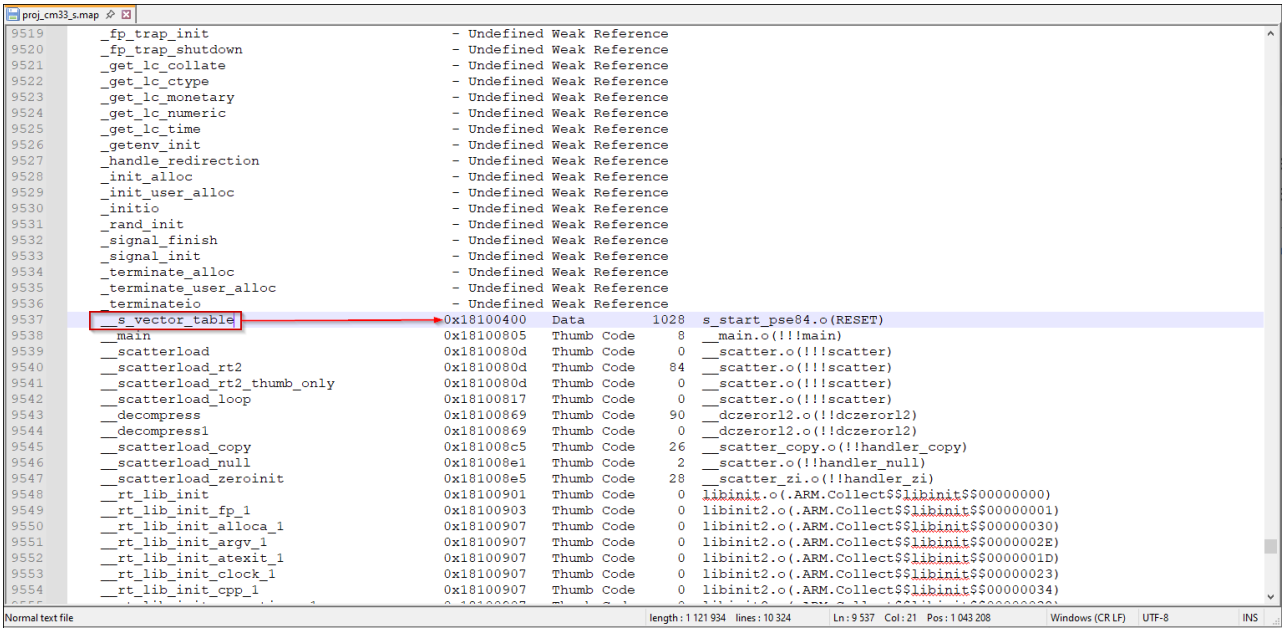
1. Select the non-secure CM33 project, open the Options dialog, and go to the **Debugger** category. Select the **Setup** tab, and then select the applicable **Driver** (I-jet, CMSIS-DAP, J-Link):



2. If using CMSIS-DAP or I-jet, switch to the **Multicore** tab and select the **Advanced** radio button. Then, specify the path to the session configuration file (same file created for Secure CM33 project) in the **Session configuration** field.

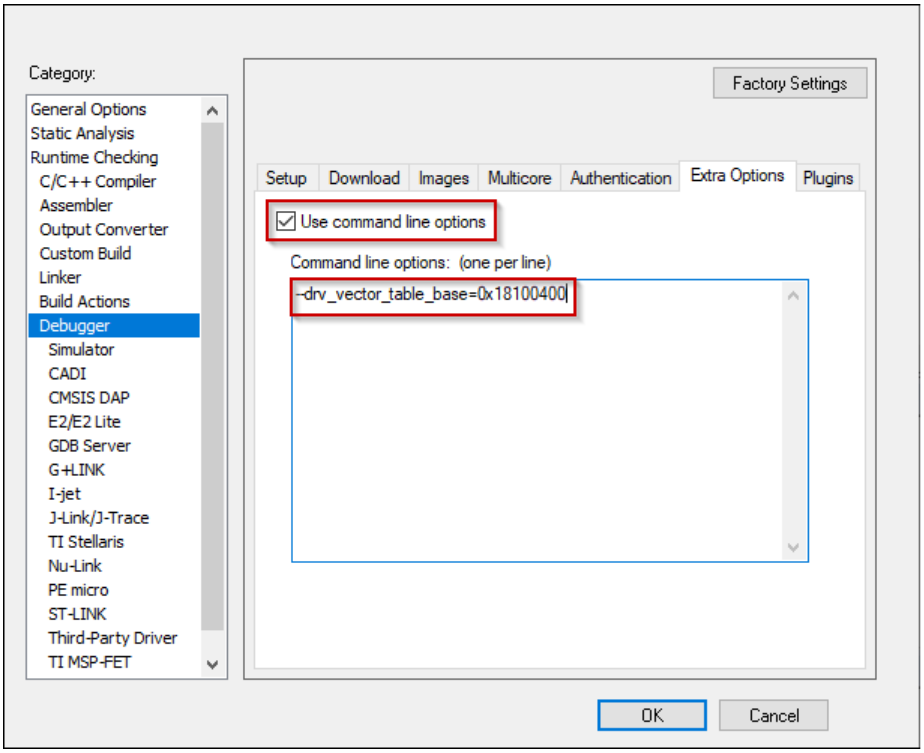


3. Open the .map file of the secure CM33 application (usually located in the *\proj_cm33_s\Debug\List* folder) and find the address in it that corresponds to the symbol `__vector_table`.



4. Switch to the **Extra Options** tab, select the **Use command line options** check box, and paste the following command-line parameter, specifying the actual address found for `__vector_table`. symbol in the .map file.

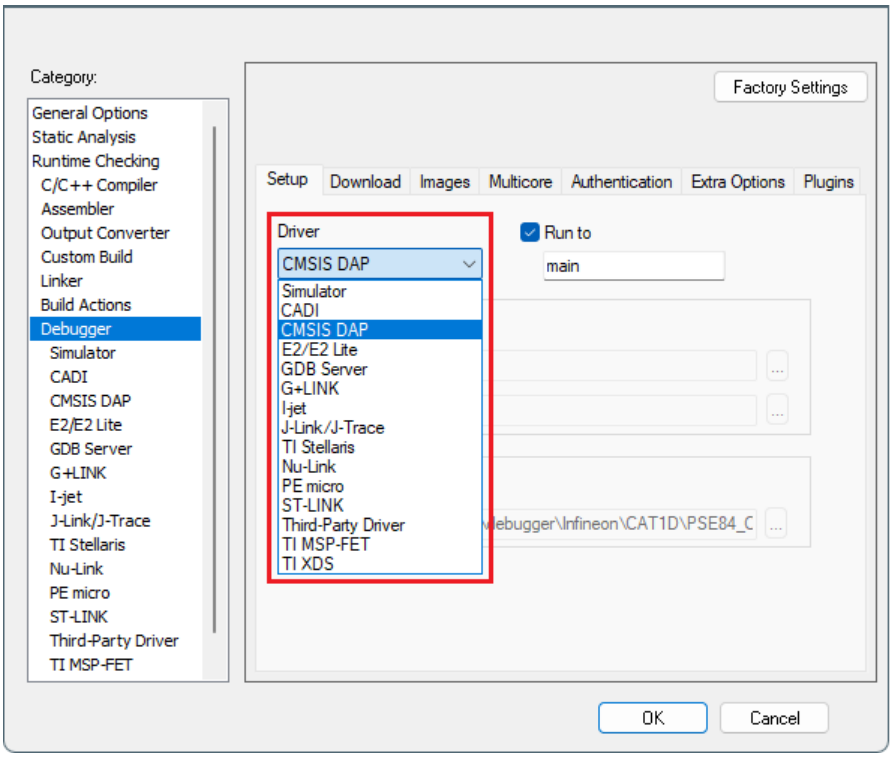
```
--drv_vector_table_base=0x18100400
```



5. Select the driver for the **Debugger** category that you selected. See Supported debugger probes.
6. Click **OK** to close the Options dialog.

5.2.3 CM55 project

1. Select the CM55 project, open the Options dialog, and got to the **Debugger** category. Select the **Setup** tab, and then select the applicable **Driver** (I-jet, CMSIS-DAP, J-Link):



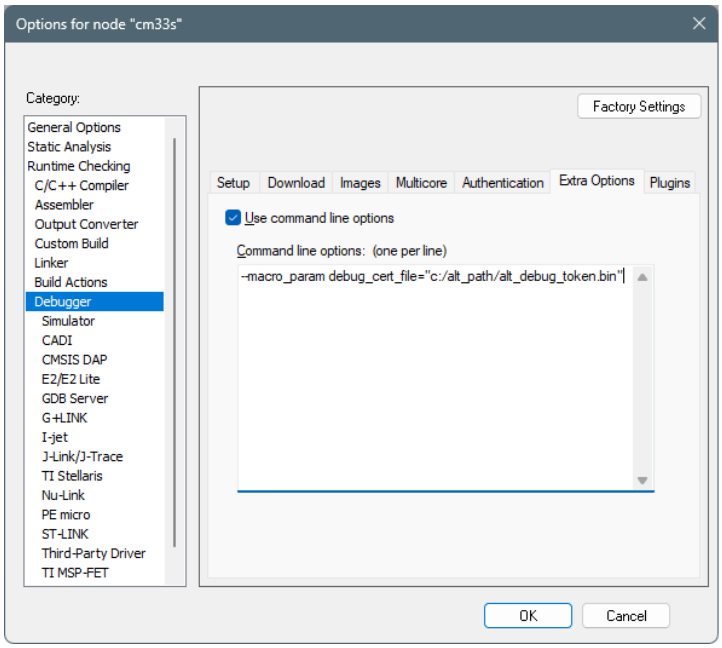
2. Select the driver for the **Debugger** category that you selected. See Supported debugger probes.
3. Click **OK** to close the Options dialog.

5.2.4 Secure debug configuration (optional)

The BootROM on the PSOC™ Edge can temporarily disable access to the core's AP. This behavior is determined by the security policy during device provisioning. However, access can be re-enabled using a debug certificate. If a debugger identifies that the AP is disabled, it will upload and verify the certificate found in the "packets/debug_token.bin" file within the current project folder. You can provide a non-default path to the certificate in the debug macro by using the `--macro_param` option.

1. Select the secure CM33 project, open Options dialog, and go to the **Debugger** category.
2. Select the **Extra Options** tab and the **Use command line options** check box.
3. Paste the following command-line parameter, specifying the actual path to the certificate.

```
--macro_param debug_cert_file="c:/alt_path/alt_debug_token.bin"
```

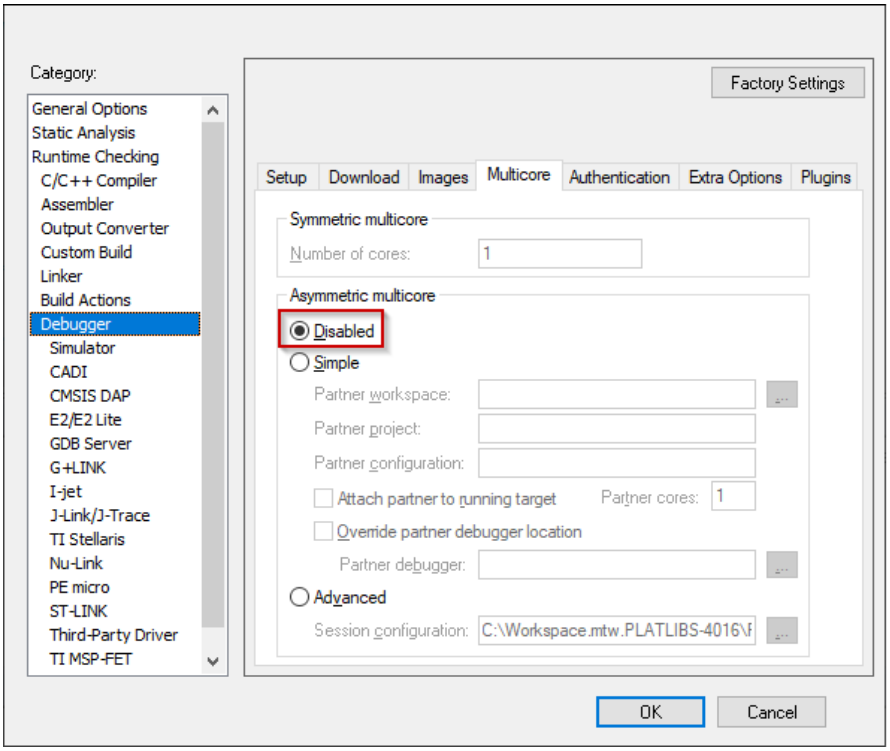


4. Click **OK** to close the Options dialog.
5. Perform the same modifications for non-secure CM33 and CM55 projects.

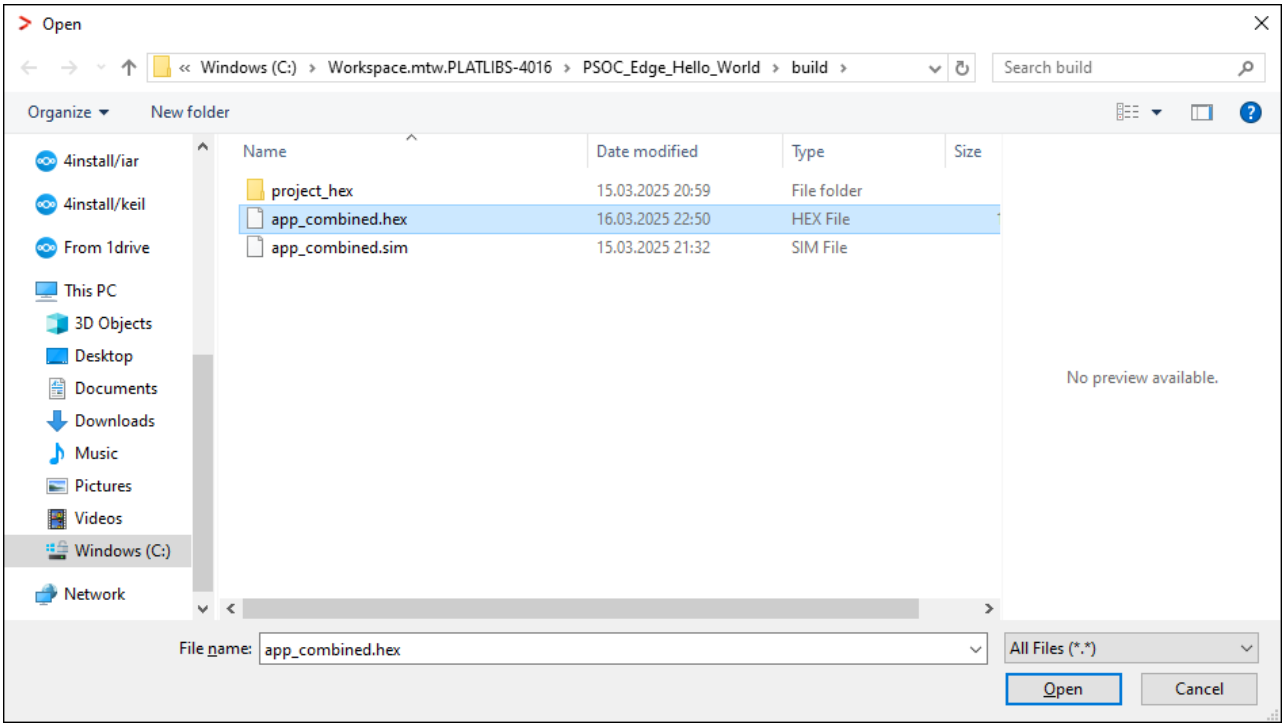
5.3 Target Programming

After preparing the combined image and setting up the debugger connection, the next step is to program the target. To accomplish this, follow these steps:

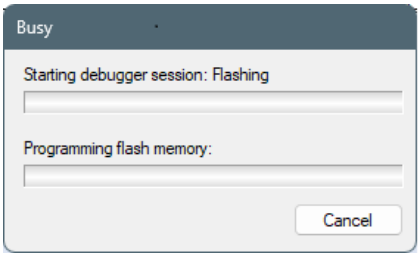
1. Select the secure CM33 project, open the Options dialog, and select the **Debugger** category.
2. On the **Multicore** tab, and temporarily select the **Disabled** radio-button to allow erasing and downloading.



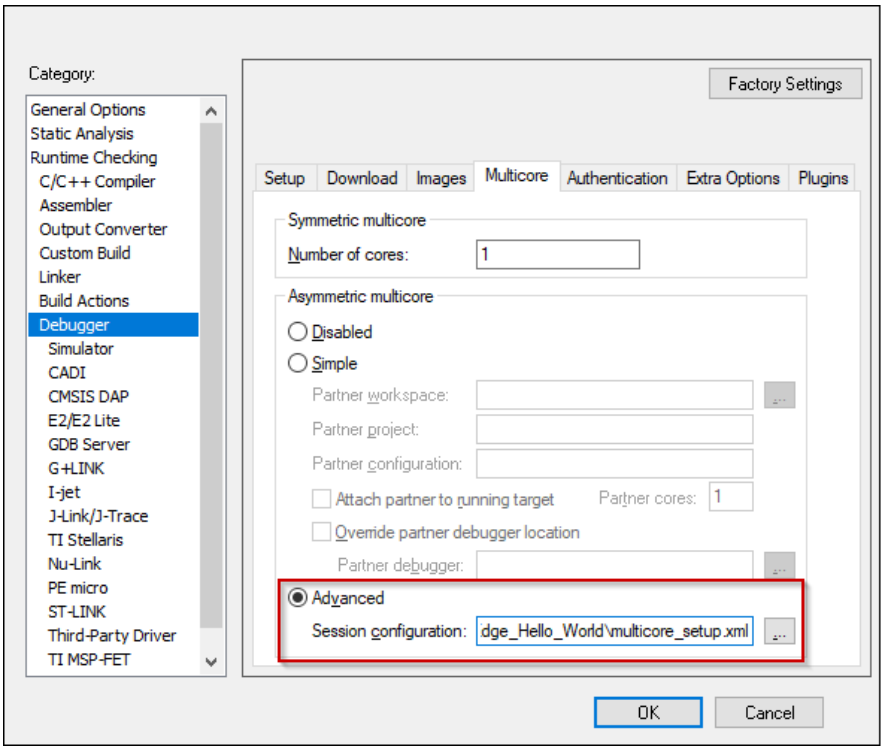
3. Click **OK** to close the Options dialog.
4. Select **Project > Download > Download file…** and select the *build\app_combined.hex* file (you may have to select all files).



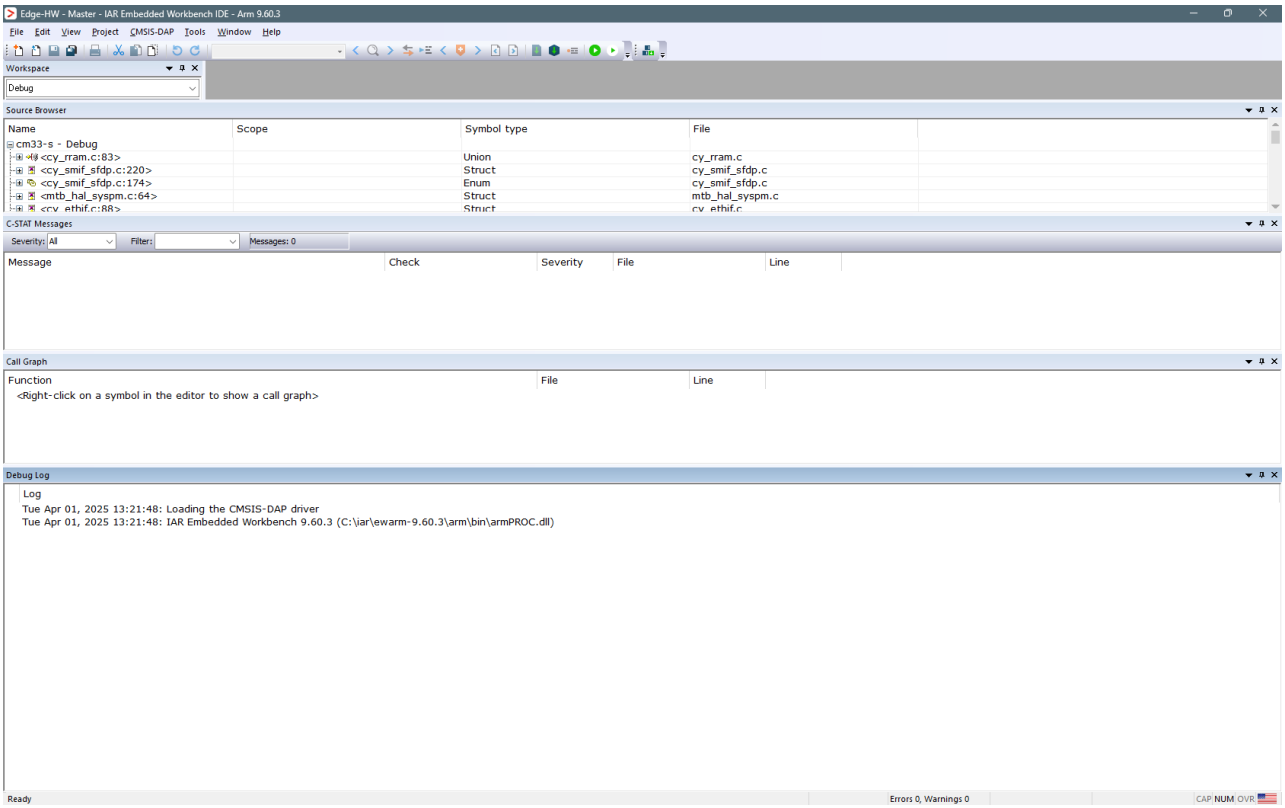
5. Click **Open**. The following dialog displays while programming. This may take a few minutes.



6. When complete, open the Options dialog to the **Debugger** category, select the **Multicore** tab, and re-enable the **Advanced** radio-button.



7. Click **OK** and the GUI changes appearance.



5.4 Launch multi-core debugging with CMSIS-DAP/I-jet

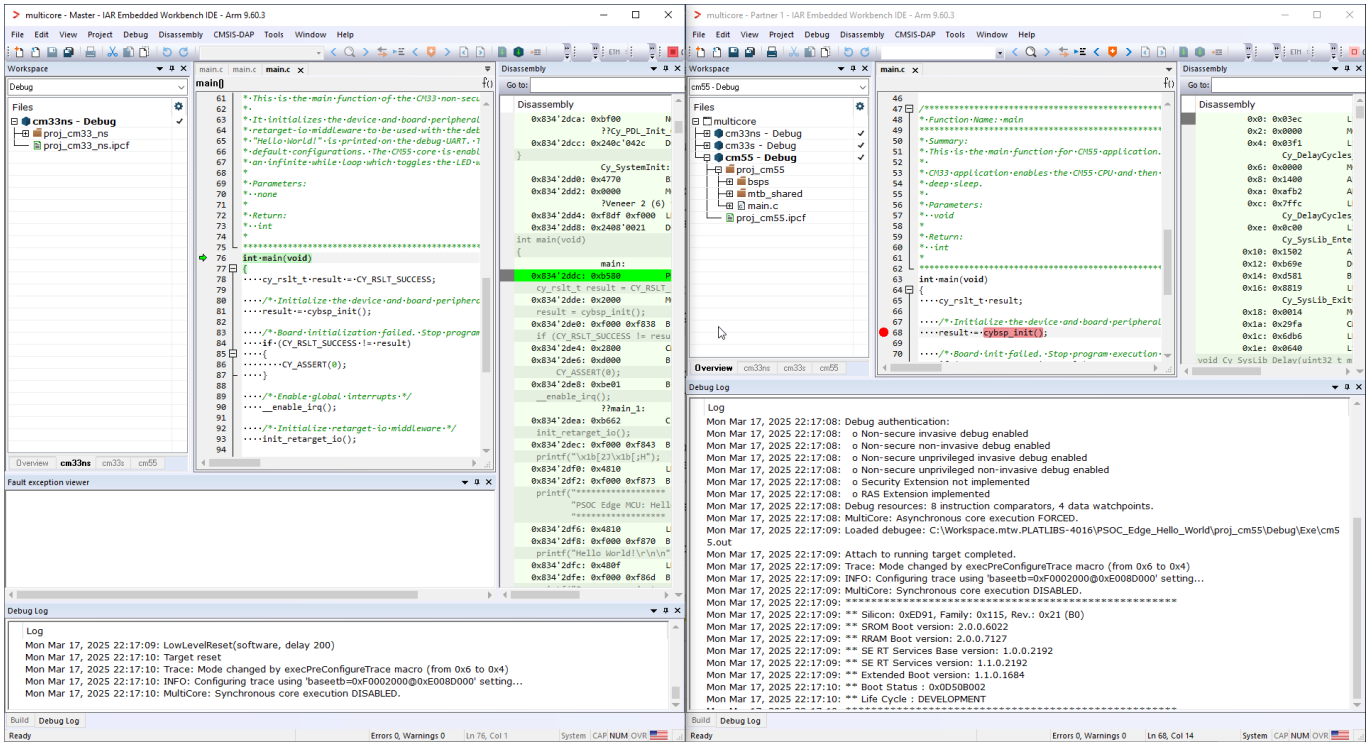
Since two instances of IAR cannot debug the same core simultaneously, it is not possible to debug a secure CM33 application and a non-secure CM33 application at the same time. The following multi-core debug combinations are possible:

- Debugging a secure CM33 application alongside a CM55 application.
- Debugging a non-secure CM33 application alongside a CM55 application.

To initiate a multi-core debug session, you must open one of the CM33 projects and **Select Project > Debug without Downloading**.

***Note**: Ensure all projects are built and programmed before launching a debug session.*

IAR opens a separate IDE instance for each project specified in the session file. It should look like to this. One is Master (cm33) and the other is Partner 1 (cm55):



The left side of the screen shows the IAR IDE instance attached to the CM33 core. The right side shows the CM55 core not started yet. Once the Cy_SysEnableCM55() function is executed on the CM33 core, the CM55 will start executing its application. You can step through the code by switching back and forth between the two IAR IDE instances.

5.5 Launch multi-core debugging with J-Link

To launch multi-core debugging with J-Link:

1. Open two instances of IAR Embedded Workbench.
2. Select the CM33 project in the first IDE instance and click Project > Debug without Downloading. The debugger will reset the target, and halt at the beginning of the CM33 project's main().
3. Select the the CM55 project in the second instance and select: Project > Attach to Running Target.
4. After performing the attach, run the CM55 core before continuing with the debugging of the CM33 project.

6 PSOC™ 4 and PSOC™ 6 single-core application

Follow steps in Create/export application for IAR Embedded Workbench.

Then, use the instructions in this section to build, program, and debug the application for a single-core PSOC™ 4 and PSOC™ 6 device in IAR Embedded Workbench.

6.1 Configure and build

In most cases, there is very little required to configure and build a simple single-core application. For some cases where configuration may be required, see the following as applicable:

- Configure applications with C++ files
- Build options, and prebuild/postbuild settings
- RTOS settings

On the IAR main menu, select **Project > Make** to build the application. You should see output like this:

```
Hello_World.out

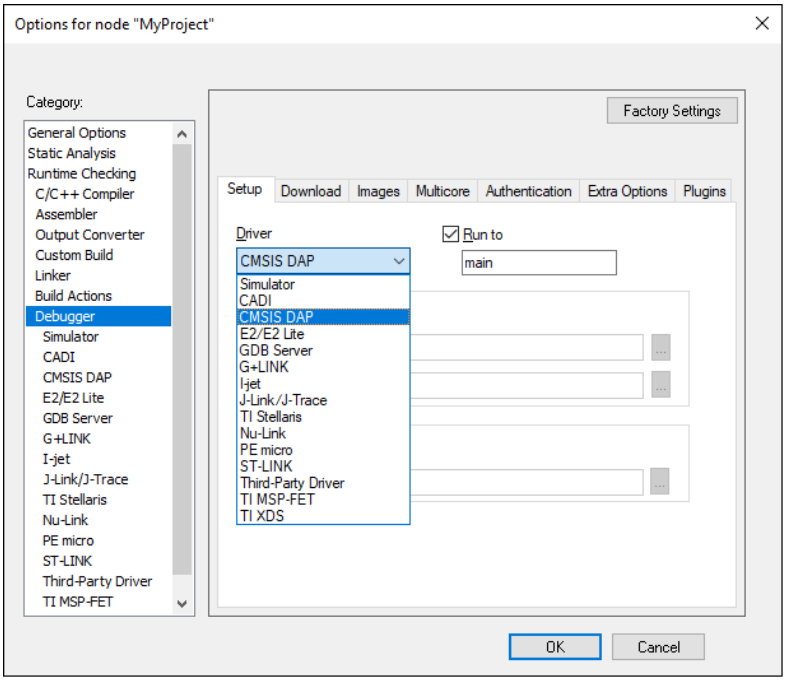
Total number of errors: 0
Total number of warnings: 0
Resolving dependencies...
Build succeeded
```

6.2 Program/Debug with KitProg3/MiniProg4 (CMSIS-DAP)

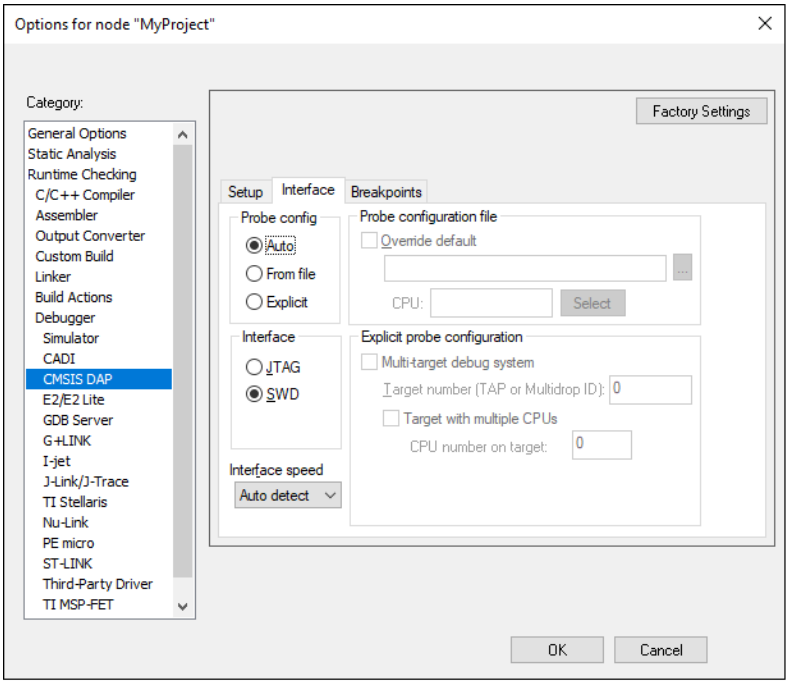
As needed, run the fw-loader tool to make sure the board firmware is upgraded to KitProg3. See the KitProg3 User Guide for details. The tool is in the following directory by default:

C:\Infineon\Tools\ModusToolboxProgtools-[version]\fw-loader\bin

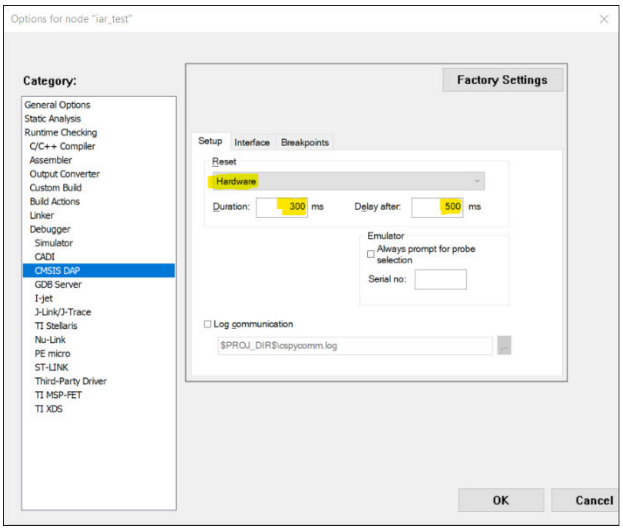
1. In IAR Embedded Workbench, open the Options dialog, go to **Debugger**, and select **CMSIS-DAP** in the driver list:



2. Select the **CMSIS-DAP** node, switch the interface from **JTAG** to **SWD**, and set the Interface speed to **2MHZ**.



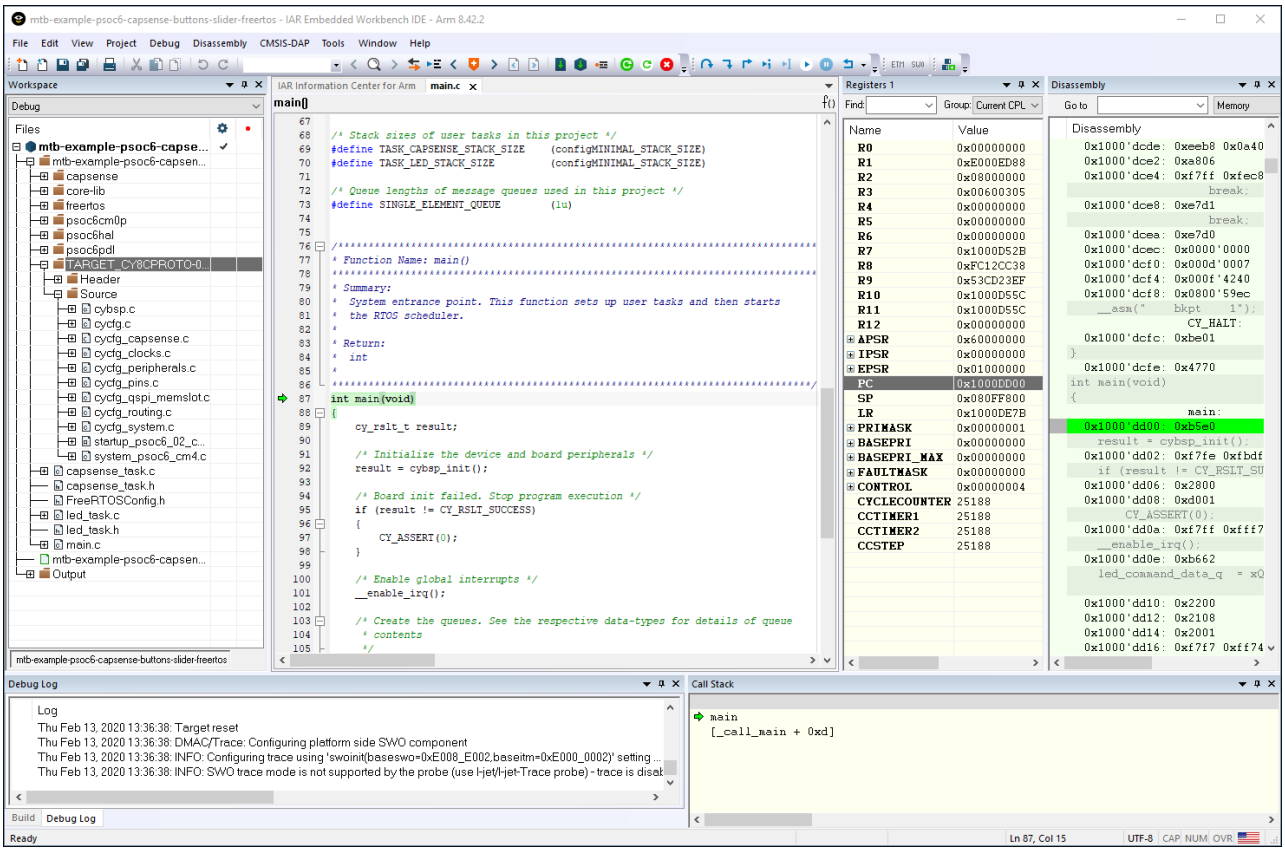
***Note**: When using MiniProg4 with a single-core PSOC™ 6 MCU, you must specify a special type of **Reset** under the **Setup** tab, as follows:*



3. Click **OK**.

4. Select **Project > Download and Debug**.

The IAR Embedded Workbench starts a debugging session and jumps to the main function.

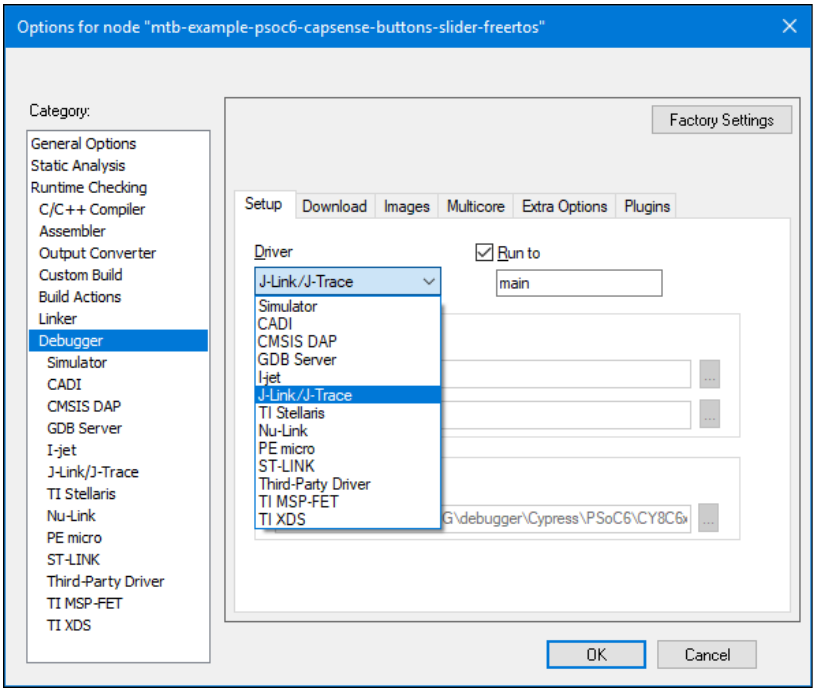


6.3 Program/Debug with J-Link

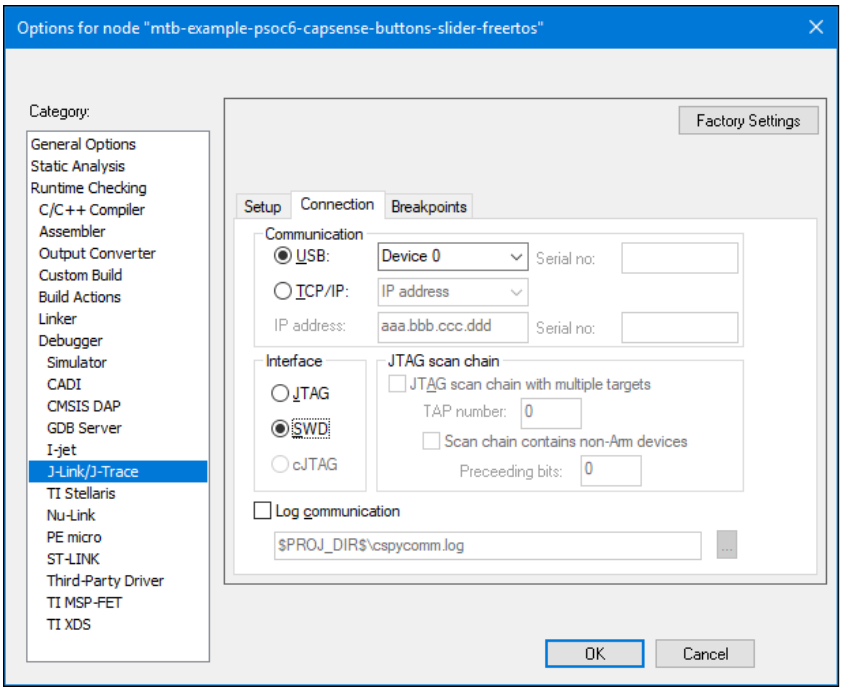
You can use a J-Link debugger probe to debug the application.

1. Open the Options dialog and select the **Debugger** item under **Category**.

2. Then select **J-Link/J-Trace** as the active driver:



3. Select the **J-Link/J-Trace** item under **Category**, and under the **Connection** tab, switch the interface to **SWD**:



4. Connect a J-Link debug probe to the 10-pin adapter (this needs to be soldered on the prototyping kits).

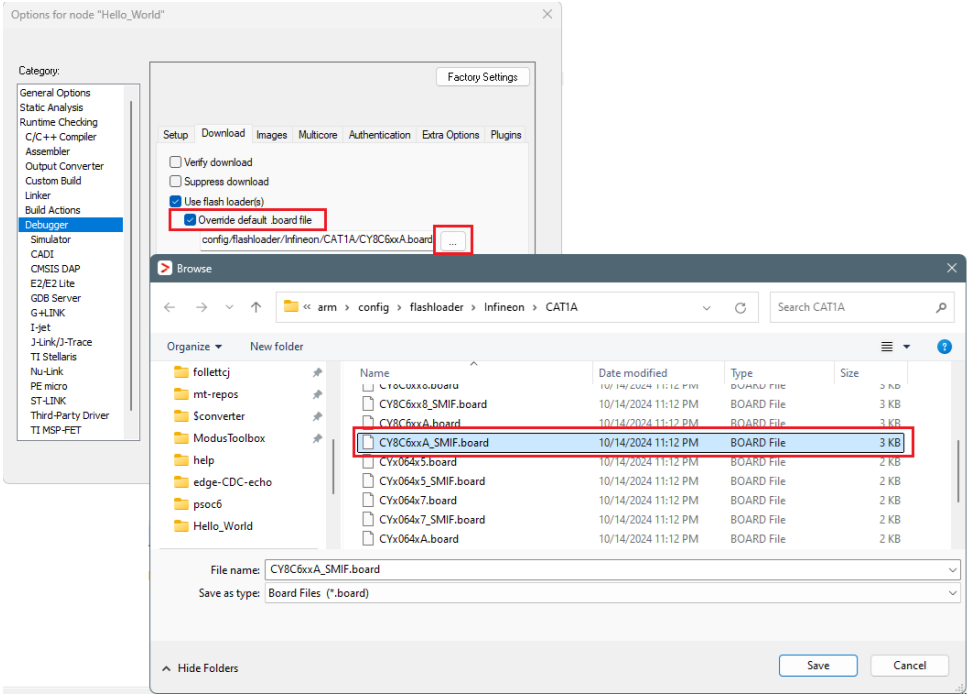
5. Select **Project > Download and Debug** to launch the debugger.

6.4 Program external memory

IAR Embedded Workbench has disabled external memory programming by default. The SMIF region in the *.board file must be enabled manually for PSOC™ 6 devices. To do that:

1. Open the Options dialog and select the **Debugger** item under **Category**.

2. Click the **Download** tab and select the **Override default .board file** check box.



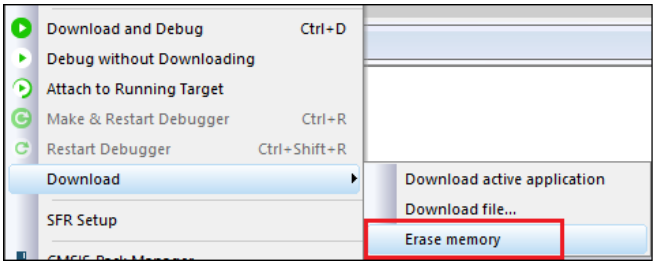
- Identify the default *.board* file currently used for this project.
- Click the **Browse [...]** button, then navigate to and select the same *.board* file that also includes "SMIF".
- Click **Save**.

3. Click **OK** to close the Options dialog.

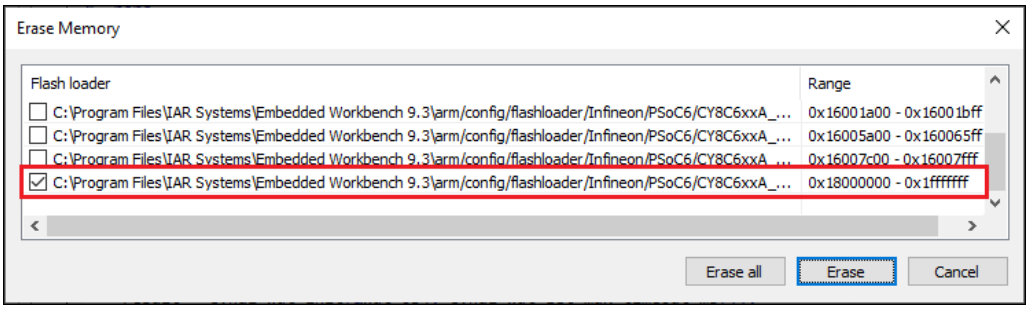
6.5 Erase PSOC™ 6 MCU with external memory enabled

To successfully erase external memory using flashloaders on PSOC™ 6 MCUs, the device's internal flash must contain valid QSPI configuration data. It may be part of a previously programmed application, such as the QSPI_XIP example. For more details, review section 7 of application note AN228740.

1. Select **Project > Download > Erase memory**.



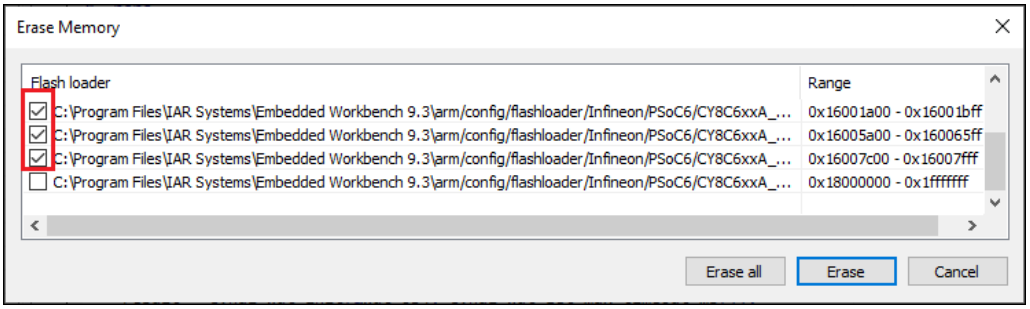
2. Deselect the check boxes for all regions, except for 0x18000000-0x1fffffff.



3. Click **Erase**.

4. Select **Project > Download > Erase memory** again.

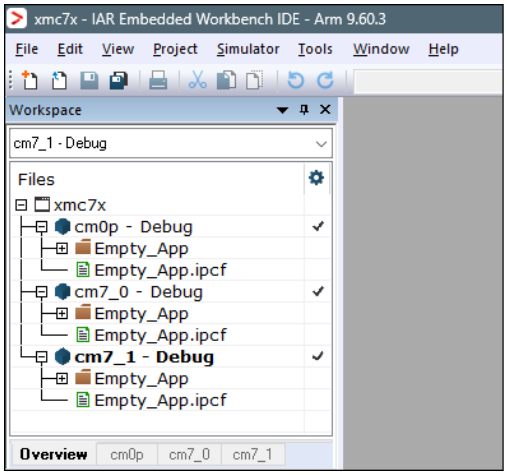
5. Select all other regions and deselect 0x18000000-0x1fffffff.



6. Click **Erase**.

7 PSOC™ 6 and XMC7000/TRAVEO™ II multi-core applications

Follow steps in Create/export application for IAR Embedded Workbench. When complete, you should have a multi-core workspace similar to the following:

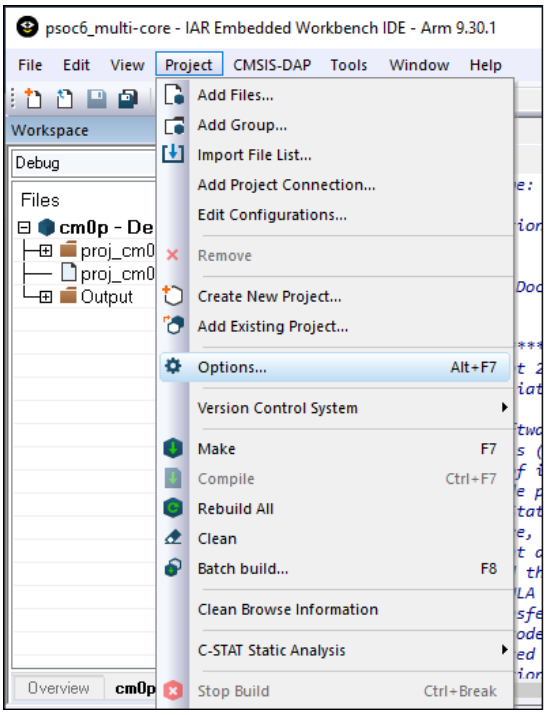


Then, use the instructions in this section to configure, build, program, and debug a multi-core application for PSOC™ 6 and XMC7000 multi-core applications in IAR Embedded Workbench.

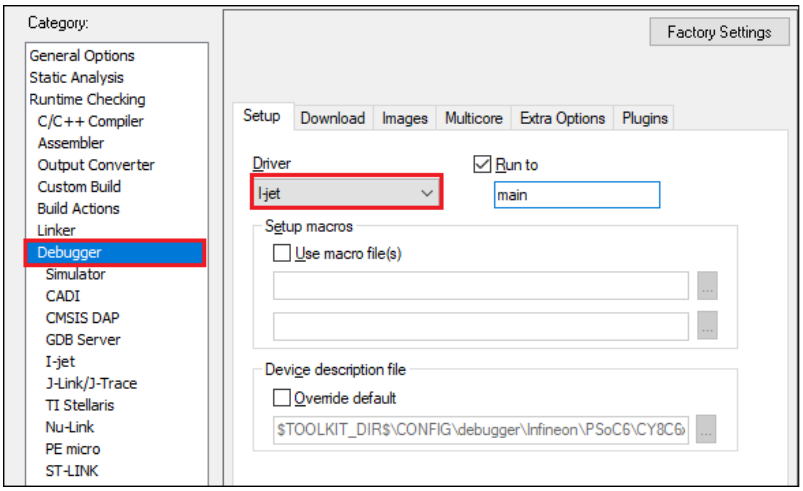
Before you can launch a multi-core debug session, all projects within the workspace must be properly configured. In IAR there is a concept of 'master' and 'slave' projects. Configure the CM0+ core project as the master project, and configure the other cores (CM4 for PSOC™ 6 and CM7 for XMC7000) as slave projects.

7.1 Configure CM4/CM7 (slave) core(s)

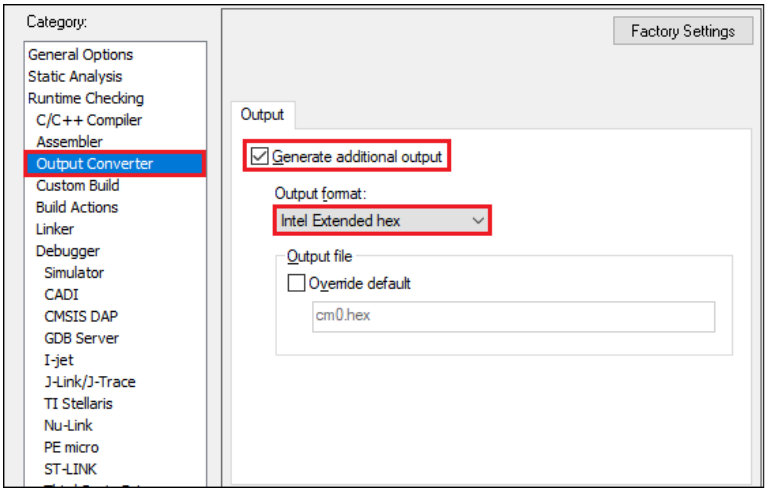
1. Select the CM4/CM7 core project and go to **Project > Options**:



2. On the dialog, select the **Debugger** category in the **Setup** tab, and then select the appropriate Driver (I-jet, CMSIS-DAP, J-Link):



3. Enable hex file generation.



- In the **Runtime Checking > Output Converter** category, select the **Generate additional output** check box.
- Ensure **Output format** is set to **Intel Extended hex**.
- Click **OK**.

4. Repeat these steps for all CM4/CM7 projects.

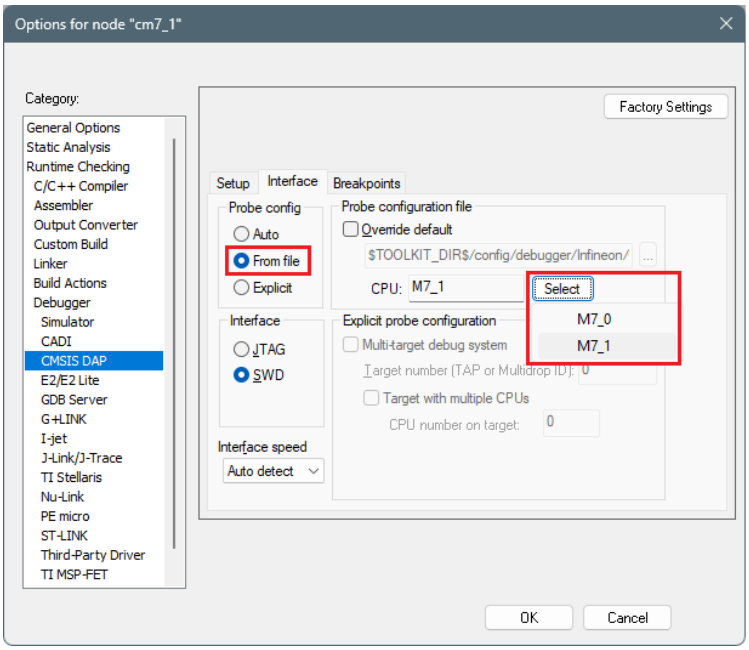
7.1.1 XMC7000/TRAVEO™ II specific steps

Some XMC7000 MCUs are triple-core devices. If you are going to use a second CM7 core in your IAR workspace, you need to implicitly set the target core in project settings so that IAR understands this project is targeting a second CM7 core. By default, IAR connects to the first CM7 core, so specifying the target core for it can be skipped.

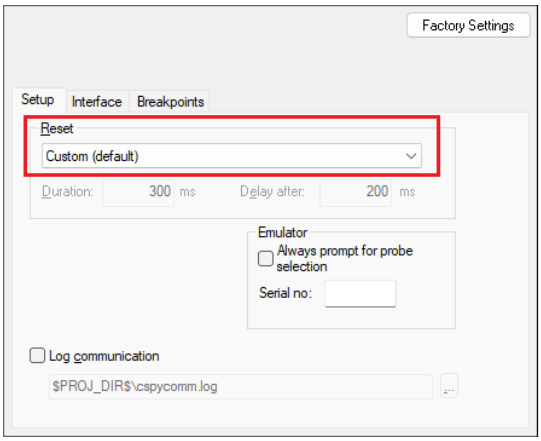
1. Select the project for the second CM7 core and go to **Project > Options**.

2. Select the probe in the **Debugger** category, and switch to the **Interface** tab.

3. Select the **From file** radio button, click **Select** next to the **CPU** label, and choose **M7_1**:



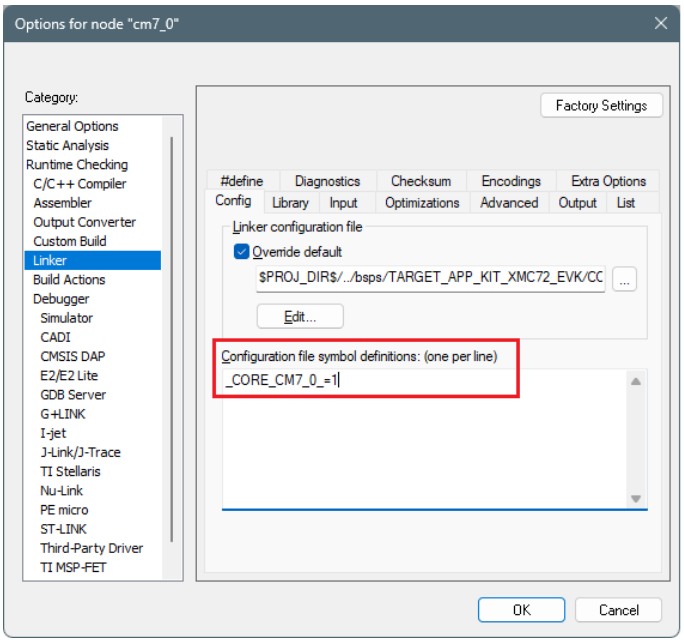
4. Switch to the **Setup** tab, and select "Custom" from the **Reset** pull-down menu.



5. In addition, specify a special linker script symbol in the project settings to distinguish CM7_0 from CM7_1, since there is a single linker script for the two CM7 cores:

a. Select the project for the first CM7 core and go to **Project > Options > Linker**.

b. Add `_CORE_CM7_0_=1` in the **Configuration file symbol definitions** field, and click **OK**.



6. Do the same for the second CM7 core:

   a. Select the project for the second CM7 core and go to **Project > Options > Linker**.

   b. Add `_CORE_CM7_1_=1` in the **Configuration file symbol definitions field**, and click **OK**.

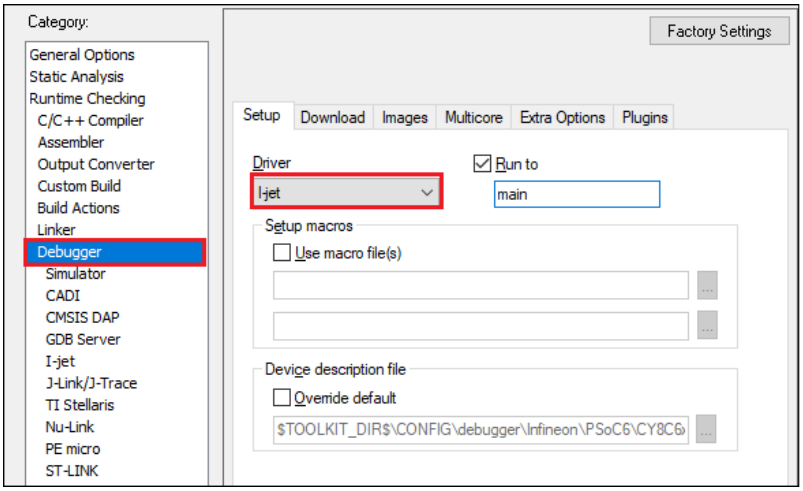
   ***Note**: When debugging CM4/CM7 core stand-alone, make sure to rebuild the CM0+ project in case any changes were made, since launching a debug session only loads the CM0+ image, but does not build that CM0+ project.*

7. Build your CM7 project(s) before moving forward.

7.2 Configure and build CM0+ (master) core

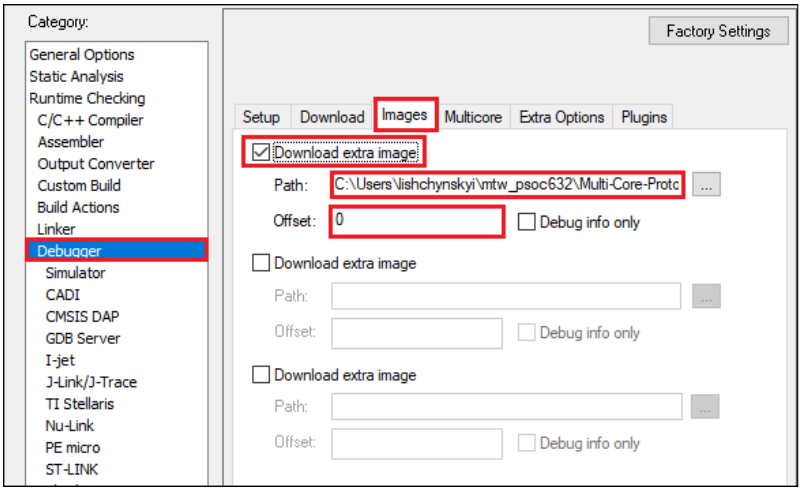
1. Select the CM0+ project and open the Options dialog.

2. On the dialog, select the **Debugger** category in the **Setup** tab, and then select the applicable **Driver** (I-jet, CMSIS-DAP, J-Link):



3. Switch to the **Images** tab to specify the extra image to be downloaded prior to debugging in order to download images of all projects in one process.

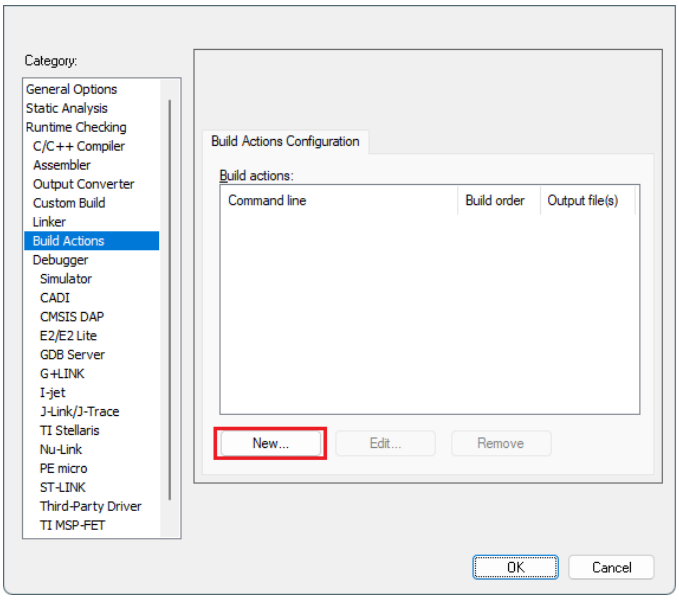
- Select the **Download extra image** check box.
- Provide a **Path** to the CM4/CM7's **HEX** image.
- Enter 0 for **Offset**.



If you provide an OUT file instead of a HEX file, the IAR IDE will fail to halt at the beginning of `main()` due to the main function present in both the CM0+ and CM4/CM7 OUT files.

***Note**: For triple-core MCUs you should download two extra images.*

4. Switch to the **Build Actions** category.



- Click the **New** button to open the Build Action dialog.
- Paste the following command to build the CM7_0 (or CM4) project in the **Command line** field as single line (edit the file name for your project)

```
iarbuild.exe "..\proj_cm7_0\[project-name].ewp" -make Debug
```

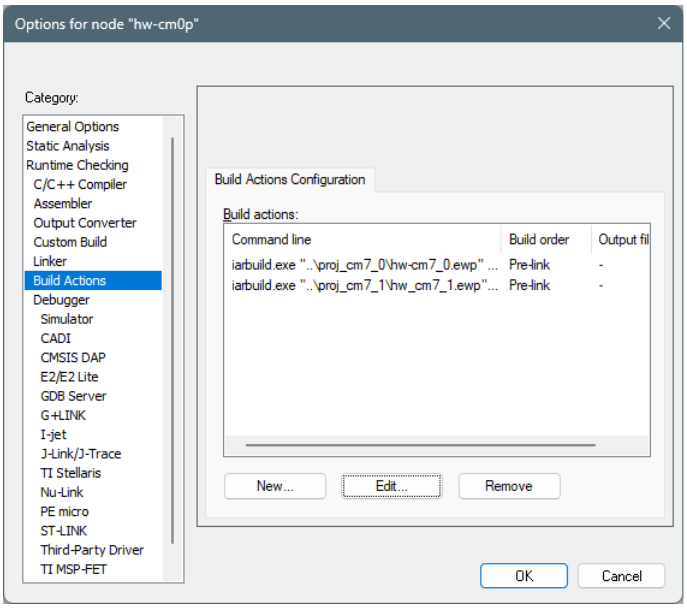
- Select **Build Order** “Run before linking”.
- Click **OK** to close the Build Action dialog.

- Click the **New** button again.
- Paste the following command to build the CM7_1 project (if applicable) in the **Command line** field as single line (edit the file name for your project)

```
iarbuild.exe "..\proj_cm7_1\[project-name].ewp" -make Debug
```

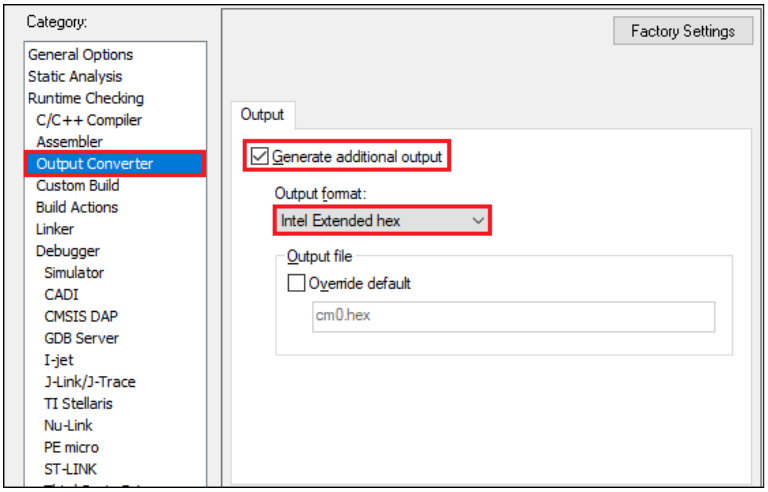
- Select **Build Order** “Run before linking”.
- Click **OK** to close the Build Action dialog.

The final list should look similar to this:



5. Enable hex file generation. In the **Runtime Checking > Output Converter** category:

- Select the **Generate additional output** check box.
- Ensure **Output format** is set to **Intel Extended hex**.



6. Click **OK**, and then select **File > Save All** to save all the changes.

7. Build the project.

7.3 Configure CMSIS-DAP/I-jet debug options

1. Create a session configuration file.

This is an xml file containing a projects list that should be launched in a multi-core debug session. The following shows an example for a triple-core device. For a dual-core device, remove the third partner node.

```
<?xml version="1.0" encoding="utf-8"?>

<sessionSetup>
    <partner>
        <name>cm0</name>
        <workspace>C:\Users\mtw-multi-core\Multicore_App\multi-core_workspace.eww</
workspace>
        <project>cm0</project>
        <config>Debug</config>
        <numberOfCores>1</numberOfCores>
        <attachToRunningTarget>false</attachToRunningTarget>
    </partner>
    <partner>
        <name>cm7_0</name>
        <workspace>C:\Users\mtw-multi-core\Multicore_App\multi-core_workspace.eww</
workspace>
        <project>cm7_0</project>
        <config>Debug</config>
        <numberOfCores>1</numberOfCores>
        <attachToRunningTarget>true</attachToRunningTarget>
    </partner>
    <partner>
        <name>cm7_1</name>
        <workspace>C:\Users\mtw-multi-core\Multicore_App\multi-core_workspace.eww</
workspace>
        <project>cm7_1</project>
        <config>Debug</config>
        <numberOfCores>1</numberOfCores>
        <attachToRunningTarget>true</attachToRunningTarget>
    </partner>
</sessionSetup>
```

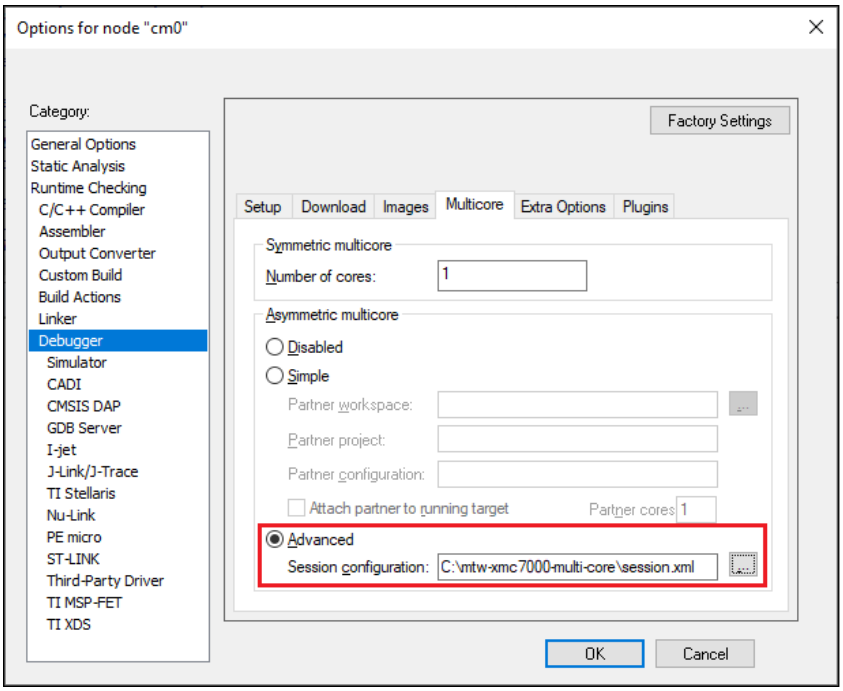
2. Configure multi-core debugging for the CM0+ project.

a. Go to **Project > Options -> Debugger**.

b. Switch to the **Multicore** tab.

c. Select the **Advanced** radio button and specify a path to the session configuration file in the **Session configuration** field.

d. Click **OK**.

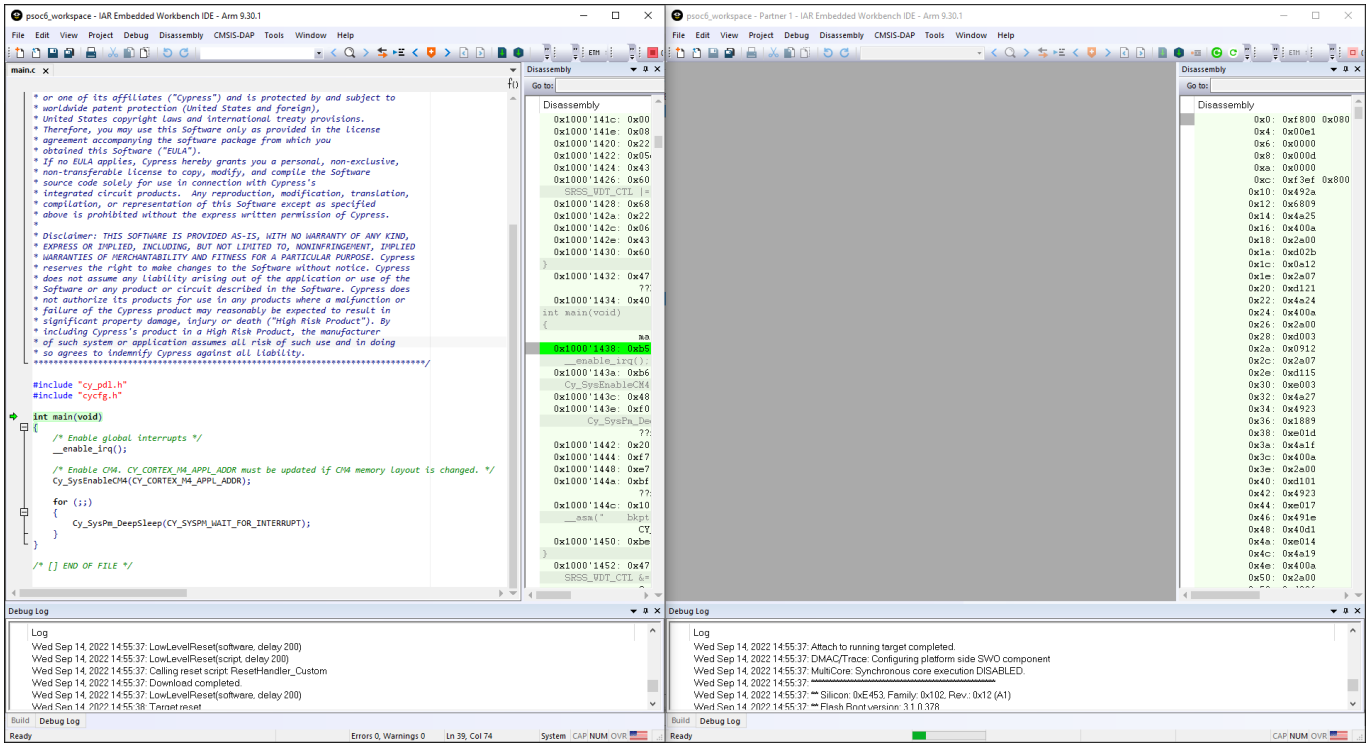


3. Save the workspace.

7.4 Launch multi-core debug session with CMSIS-DAP/I-jet

Select the CM0 project and click the **Download and debug** button.

IAR builds all projects, programs all the separate images, and launches a multi-core debug session. IAR opens a separate IDE instance for each project specified in the session file. For dual-core MCUs, it should look like to this:



The left side of the screen shows the IAR IDE instance attached to the CM0+ core. The right side shows the CM4 (or CM7) core not started yet. Once the `Cy_SysEnableCM4()` function is executed on the CM0+ core, the CM4/CM7 will start executing its application.

You can step through the code by switching back and forth between the two IAR IDE instances.

7.5 Launch multi-core debug session with J-Link

The IAR IDE does not have native support for the J-Link driver, which imposes some limitations:

IAR does not provide native multi-core debugging support when using a J-Link probe. This means that in order to launch multi-core debugging, you must open a few IAR IDE instances manually (one instance per core). Also, multi-core debugging with a J-Link probe lacks some features available with CMSIS-DAP and I-jet probes. Therefore, depending on the target probe, you need to configure projects slightly differently.

- IAR will not automatically open separate IDE instances for each core, so you need to do it manually.
- Some enhanced features are not available; see the Multi-core toolbar and CTI usage section for more details.

To launch multi-core debugging with J-Link:

1. Open your multi-core IAR workspace in separate IDE instances (the number of IDE instances should be equal to the number of cores on your MCU).

2. Select the CM0+ project in the first IDE instance and click **Download and Debug**. The debugger will download all images, reset the target, and halt at the beginning of the CM0+ project's `main()`.

3. Switch to the other IDE instances and select: **Project > Attach to Running Target**.

8 PSOC™ 64 secure single-core application

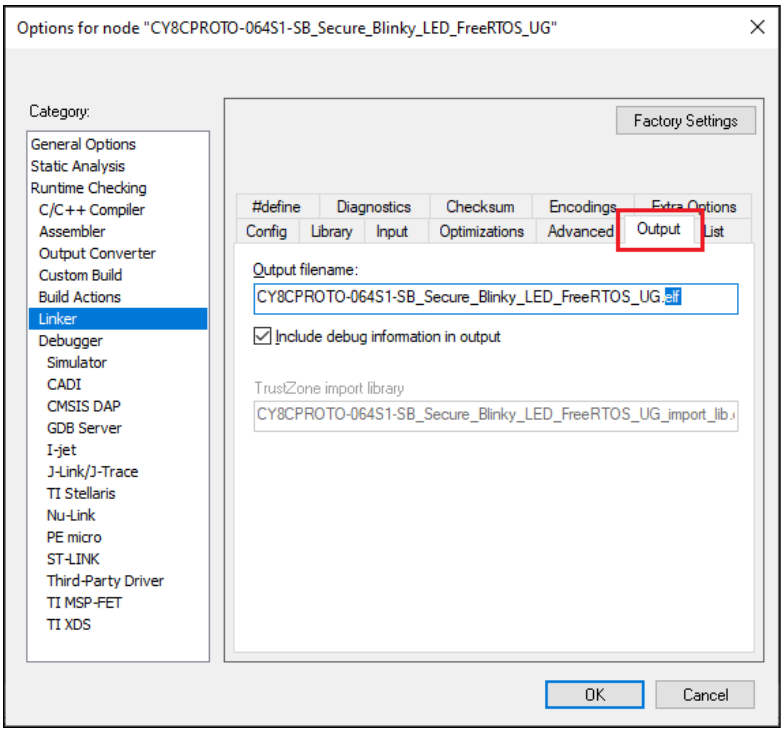
Follow steps in Create/export application for IAR Embedded Workbench. The project creation process also generates an *ide_postbuild.bat* file used to execute postbuild commands for the application. When complete, use the instructions in this section to configure, build, program, and debug the application for a PSOC™ 64 secure MCU in IAR Embedded Workbench.

***Note**: For a PSOC™ 64 secure MCU, you must also follow the instructions in the Secure Boot SDK user guide to generate keys and provision the device.*

8.1 Configure and build

After creating an application, make a few changes before attempting to build it:

1. In IAR embedded Workbench, open the Options dialog, go to **Linker > Output**, and change the file extension from ".out" to ".elf" in **Output filename** field:



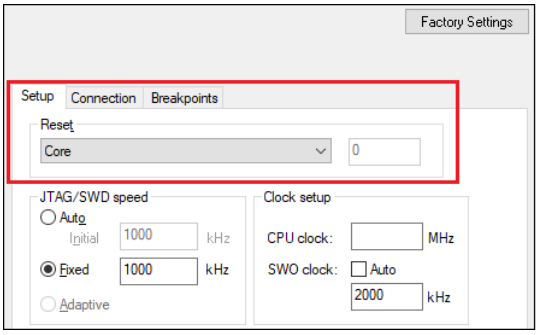
2. Click **OK** to close the dialog.

3. On the IAR main menu, select **Project > Make** to build the application.

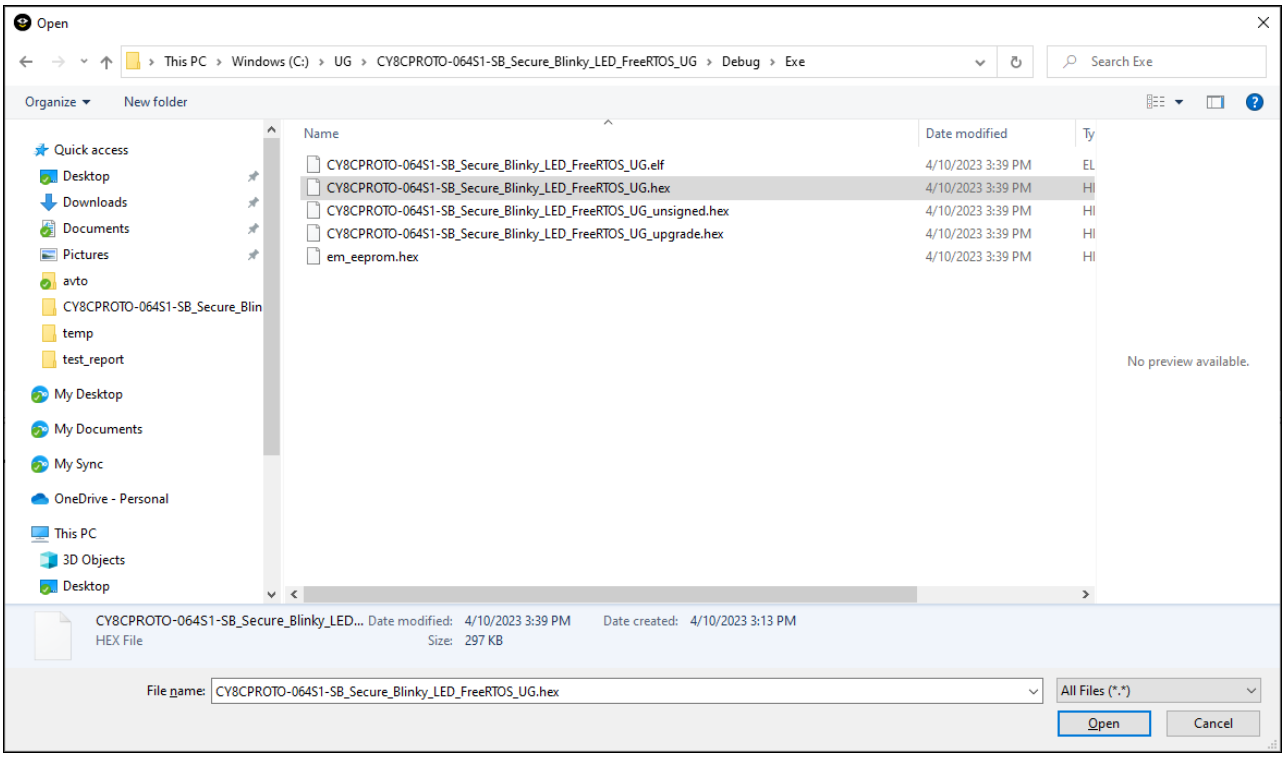
8.2 Program and debug

1. You can set the default debugger options for CMSIS-DAP or J-Link. See Program/Debug with KitProg3/MiniProg4 (CMSIS-DAP) or Program/Debug with J-Link.

***Note**: If using J-Link with a PSOC™ 64 "Secure Boot" MCU, you must specify a special type of **Reset** "Core" under the **J-Link > Setup** tab, as shown:*



2. Select **Project > Download > Download file…** and select the *<project_name>.hex* file in *<project_root >\Debug\Exe*.



3. Select **Project > Debug without Downloading**.

9 AIROC™ CYW20829 application

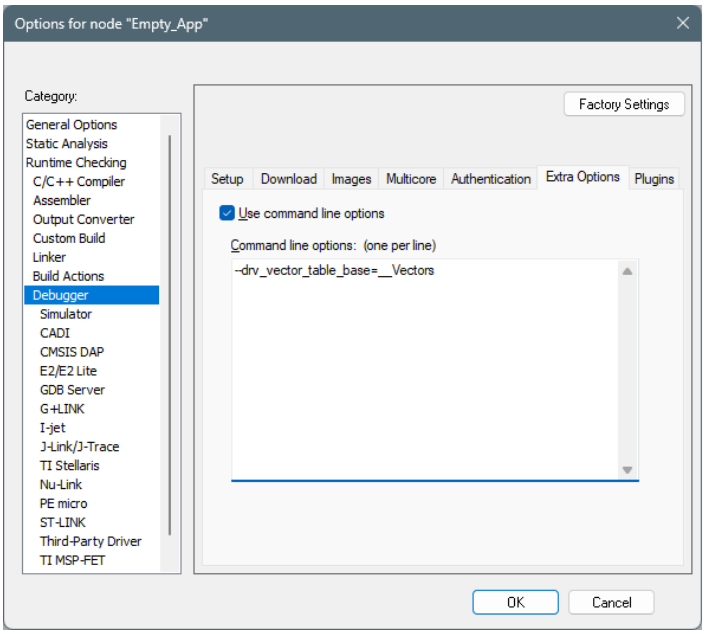
Follow steps in Create/export application for IAR Embedded Workbench.

Fix vector table

There is a known issue with an invalid for IAR name for the vector table in the C start-up code. To resolve it:

1. Open the Options dialog and select the **Debugger** item under **Category**.
2. Select the **Extra Options** tab, select the **Use command line options** check box, and paste the following command-line option:

```
--drv_vector_table_base=__Vectors
```



3. Click **OK** to close the Options dialog.

Build, program, debug

To build, save the application and select **Project > Make**. The Output should display the progress, ending with text similar to this:

```
Total number of errors: 0
Total number of warnings: 0
Resolving dependencies...
Build succeeded
```

To program, select **Project > Download > Download file…** and select the *[project_name].final.hex* file in *[project_root]\build\last_config* [you might have to switch to All Files (*.*)].

To debug, select **Project > Debug without Downloading**.

10 Make application changes and re-export

After you have successfully built the application and programmed the device, you may wish to make changes, such as configuring hardware settings for the BSP and the device, as well as adding and updating libraries. To do that, you will need to open a specific ModusToolbox™ tool or configurator, make appropriate hardware or library changes, and then re-export to update the application.

When working with a ModusToolbox™ application in IAR Embedded Workbench, it is not as convenient to open various tools and configurators as it is using Eclipse or Visual Studio Code. Those IDEs have customized plug-ins or extensions that provide a more integrated experience. For IAR Embedded Workbench, the best way to open tools and configurators and update the application is to use the modus-shell terminal.

You can also run any tool or configurator from the installation directory, under *tools_[version]*. After the tool opens, you will have to find the configuration file or project directory from the tool.

Every tool and configurator includes a separate guide for how to use it.

10.1 Configure hardware

Configuring hardware involves using the BSP Assistant tool to change the device and/or configurations. We recommend that you back up your project files first. Then, open the Library Manager using one of the methods described in Opening ModusToolbox™ tools and configurators. Refer to the BSP Assistant user guide for all the details about how to use this tool.

If you just want to modify parts of the existing device such as peripherals, ports/pins, clocks, etc., you can open the Device Configurator from the BSP Assistant, or as described in Opening ModusToolbox™ tools and configurators. For more details about that tool, refer to the Device Configurator user guide.

10.2 Add/update libraries

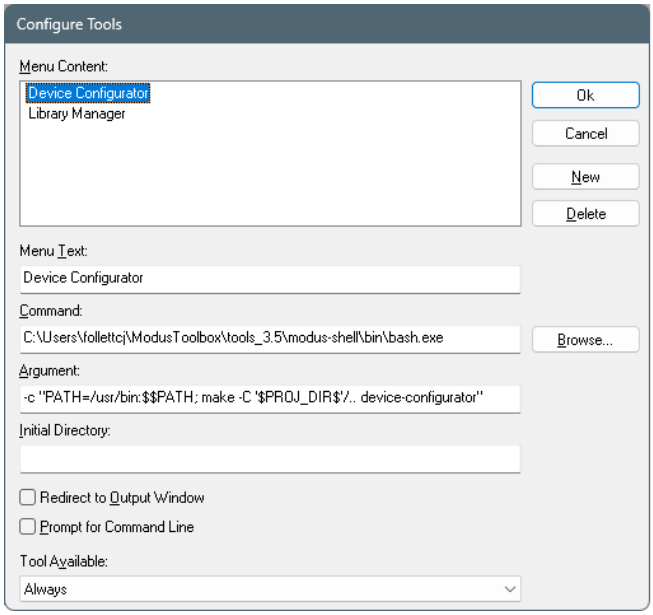
Another way to make changes to your application is by adding and updating various libraries. You do this with the Library Manager. We recommend that you back up your project files first. Then, open the Library Manager using one of the methods described in Opening ModusToolbox™ tools and configurators. For more details about the Library Manager, refer to the user guide.

10.3 Opening ModusToolbox™ tools and configurators

When working with a ModusToolbox™ application in IAR Embedded Workbench, it is not as convenient to open various tools and configurators as it is using Eclipse or Visual Studio Code. Those IDEs have customized plug-ins or extensions that provide a more integrated experience. Still, there are some ways to open various ModusToolbox™ tools for use with IAR Embedded Workbench. Here are a few of them:

10.3.1 Configuring IAR Embedded Workbench

One option to open ModusToolbox™ tools is to configure the IAR Embedded Workbench **Tools** menu using the **Configure Tools** dialog to add shortcuts to the various tools. This is a one-time configuration for each tool you wish to add. Then, the tools will be available for all your ModusToolbox™ applications running in IAR Embedded Workbench.



On this dialog, enter the following:

- **Menu Text**: Name of the tool.
- **Command**: `[path-to-modustoolbox-tools]\modus-shell\bin\bash.exe`

  This opens the bash shell from the tools package installation location. You can navigate to the executable using the [**Browse...**] button.
- **Argument**: `-c "PATH=/usr/bin:$$PATH; make -C '$PROJ_DIR$' [tool-name]"`

  This passes a command to the bash shell to open the specified `[tool-name]` (for example, device-configurator, bsp-assistant, library-manager, etc.).

  The variable `'$PROJ_DIR$'` is the IAR Embedded Workbench project directory, and it is quoted because it returns a Windows-style path, which does not work without the quotes in bash. If the application is multi-core, add `/..` to navigate to one directory higher where the *Makefile* is located.
- **Redirect to Output Window**: This check box is optional. If selected, the bash shell commands run in the output window instead of a separate terminal.

For more details about this dialog, refer to the IAR Embedded Workbench user guide/help.

10.3.2 Using modus-shell

Another way to open tools and configurators and update the application is to use the modus-shell bash terminal. Open this tool from the installation directory or by typing "modus-shell" in the Windows **Search** box.

As described in the ModusToolbox™ tools package user guide "ModusToolbox™ build system'' chapter, you can run numerous make commands in the application directory. These include launching tools and configurators. In the modus-shell terminal, navigate to the application or project directory containing the *Makefile*, and run a `make [tool-name]` command. For example:

```
cd mtw/example/Hello_World

make library-manager
```

For consistency, the `[tool-name]` is always the name of the tool file name on disk, such as library-manager, bsp-assistant, device-configurator, etc. You can view tool/configurator names by using the `make help` command.

When you open tools and configurators this way, they are opened in context with your application.

10.3.3 Opening directly

Another way to open any ModusToolbox™ tool or configurator is directly by launching it from the installation directory, under *tools_[version]*. On Windows, you can also use the **Search** box and type the name of the tool. After the tool opens, you will have to find the configuration file or project directory from the tool to use it with your application. Each tool and configurator includes a user guide for how to do this.

10.4 Re-export to update application

After running a tool or configurator to change some part of your application, you will need to update the application using the `make ewarm` command with the modus-shell terminal.

Attention**: **It's very important to pay attention to the name of your ipcf file(s). The name is always the one you specified when creating the project. However, if you exported from another IDE without using the `CY_IDE_PRJNAME` variable, the name will come from the project Makefile, which might be different than the ipcf file name..

The basic command is as follows:

```
make ewarm CY_IDE_PRJNAME=[existing-cprj-name] TOOLCHAIN=IAR
```

After running the command, verify the *ipcf* file(s) is(are) updated. When you view the project in IAR Embedded Workbench, you will see the updated file structure.

Revision history

Revision	Date	Description
**	2023-05-15	New document.
*A	2023-06-02	Removed obsolete instructions for customizing linker scripts.
*B	2024-01-24	Updated for ModusToolbox™ version 3.2. Updated recommended version. Removed Python. Added instructions for projects with C++ files. Added instructions for AIROC™ CYW20829 devices.
*C	2024-09-27	Updated for ModusToolbox™ version 3.3.
*D	2024-12-02	Included instructions for configuring a PSOC™ Control C3 device.
*E	2024-12-06	Updated for ModusToolbox™ version 3.4.
*F	2025-03-25	Updated for ModusToolbox™ version 3.5.
*G	2025-04-28	Updated document to separate the instructions per each device type.
*H	2025-09-03	Updated for ModusToolbox™ version 3.6 and upcoming production version of PSOC™ Edge device.
*I	2025-12-12	Updated for ModusToolbox™ version 3.7.
*J	2025-12-18	Removed section from PSOC™ Control Program and Debug instructions that is no longer needed.
*K	2026-03-21	Updated for ModusToolbox™ version 3.8; simplified instructions for various devices; added link to ETM/ITM Tracing application note for PSOC™ Edge and PSOC™ Control devices.
*L	2026-06-26	Updates to PSOC™ Control and AIROC™ CYW20829 instructions.
*M	2027-07-24	Updated PSOC™ Control single-core configuration instructions.

Trademarks

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

Edition 2026-07-24

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2026 Infineon Technologies AG

All Rights Reserved.

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-efq1743008065249

Important notice

Products which may also include samples and may be comprised of hardware or software or both ("Product(s)") are sold or provided and delivered by Infineon Technologies AG and its affiliates ("Infineon") subject to the terms and conditions of the frame supply contract or other written agreement(s) executed by a customer and Infineon or, in the absence of the foregoing, the applicable Sales Conditions of Infineon. General terms and conditions of a customer or deviations from applicable Sales Conditions of Infineon shall only be binding for Infineon if and to the extent Infineon has given its express written consent.

For the avoidance of doubt, Infineon disclaims all warranties of non-infringement of third-party rights and implied warranties such as warranties of fitness for a specific use/purpose or merchantability. Infineon shall not be responsible for any information with respect to samples, the application or customer's specific use of any Product or for any examples or typical values given in this document.

If this document is marked as 'Preliminary', 'Target', 'Draft' or 'Proposal', Infineon reserves the right to change all information given in this document at any time without notice. Subject to the development and release of a Product for series supply by Infineon, the technical specifications of the Product are set forth in the relevant datasheet provided by Infineon without such marking.

The data contained in this document is exclusively intended for technically qualified and skilled customer representatives. It is the responsibility of the customer to evaluate the suitability of the Product for the intended application and the customer's specific use and to verify all relevant technical data contained in this document in the intended application and the customer's specific use. The customer is responsible for properly designing, programming, and testing the functionality and safety of the intended application, as well as complying with any legal requirements related to its use.

Unless otherwise explicitly approved by Infineon, Products may not be used in any application where a failure of the Products or any consequences of the use thereof can reasonably be expected to result in personal injury.

However, the foregoing shall not prevent the customer from using any Product in such fields of use that Infineon has explicitly designed and sold it for, provided that the overall responsibility for the application lies with the customer. Infineon expressly reserves the right to use its content for commercial text and data mining (TDM) according to applicable laws, e.g. Section 44b of the German Copyright Act (UrhG). If the Product includes security features:

Because no computing device can be absolutely secure, and despite security measures implemented in the Product, Infineon does not guarantee that the Product will be free from intrusion, data theft or loss, or other breaches ("Security Breaches"), and Infineon shall have no liability arising out of any Security Breaches.

If this document includes or references software:

The software is owned by Infineon under the intellectual property laws and treaties of the United States, Germany, and other countries worldwide. All rights reserved. Therefore, you may use the software only as provided in the software license agreement accompanying the software.

If no software license agreement applies, Infineon hereby grants you a personal, non-exclusive, non-transferable license (without the right to sublicense) under its intellectual property rights in the software (a) for software provided in source code form, to modify and reproduce the software solely for use with Infineon hardware products, only internally within your organization, and (b) to distribute the software in binary code form externally to end users, solely for use on Infineon hardware products. Any other use, reproduction, modification, translation, or compilation of the software is prohibited.

For further information on the Product, technology, delivery terms and conditions, and prices, please contact your nearest Infineon office or visit www.infineon.com.