IAR Embedded Workbench for ModusToolbox™ user guide

ModusToolbox™ tools package version 3.5.0

About this document

Scope and purpose

A newer version of this document may be available on the web here.

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

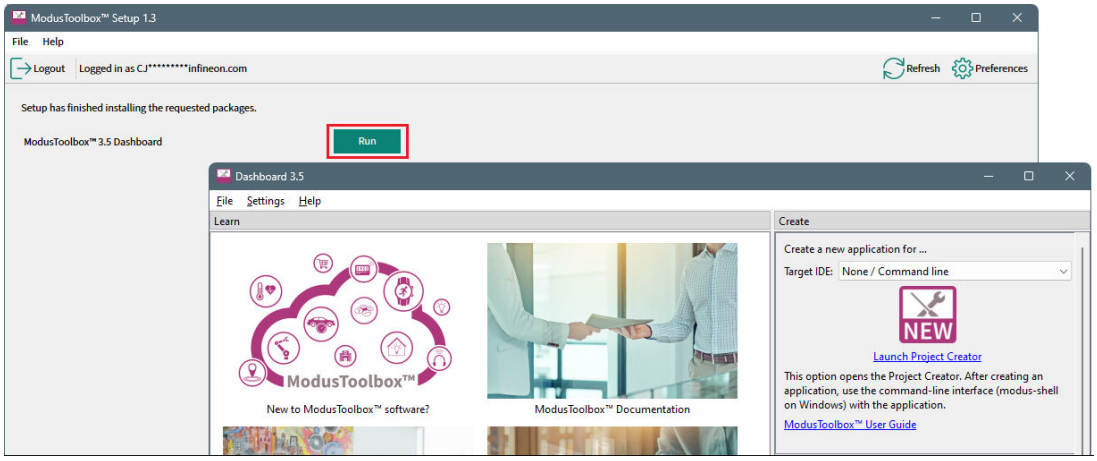
Table of contents

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_ARM_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

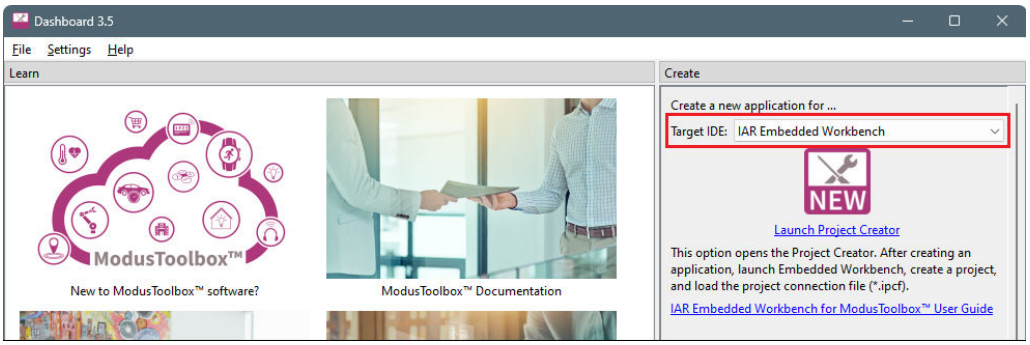
This section covers how to create or export a ModusToolbox™ application for use in the IAR Embedded Workbench IDE.

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

2.1 Create/export ModusToolbox™ application

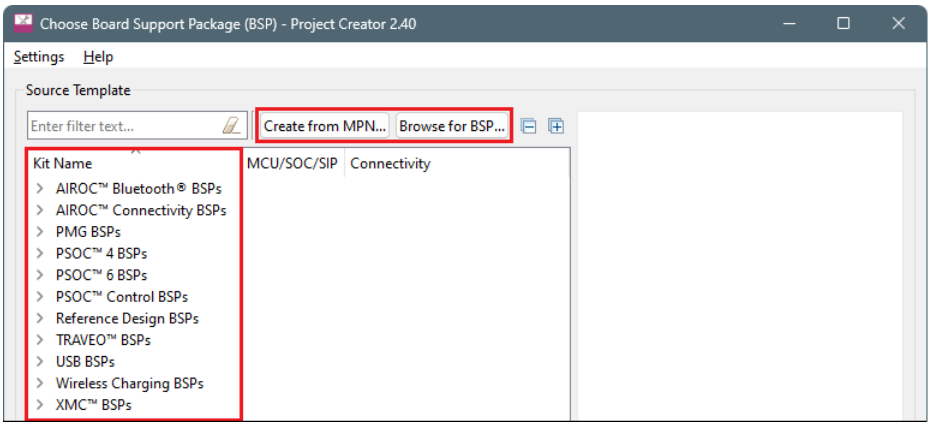
Create new application

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



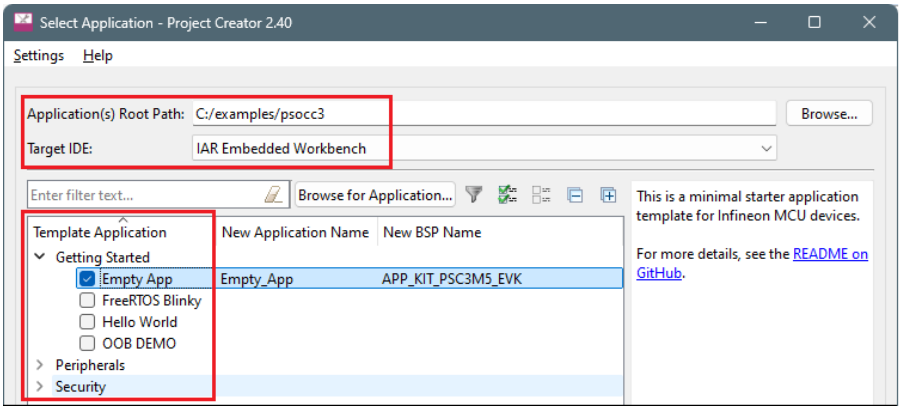
Refer to the Project Creator user guide for more details.

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.

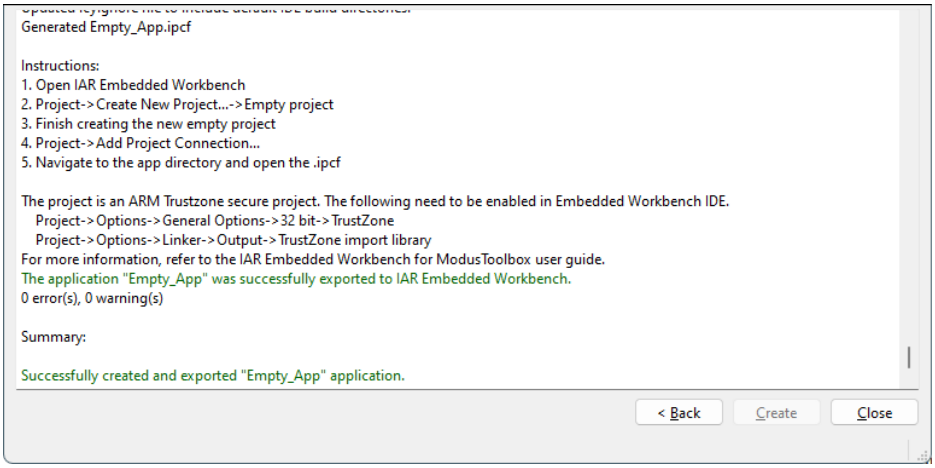


3. On the next page, select the location and the application to create. Notice Target IDE is already selected from the Dashboard.

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:

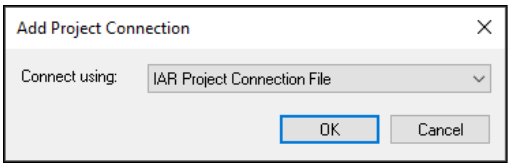
1. Launch IAR Embedded Workbench.

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

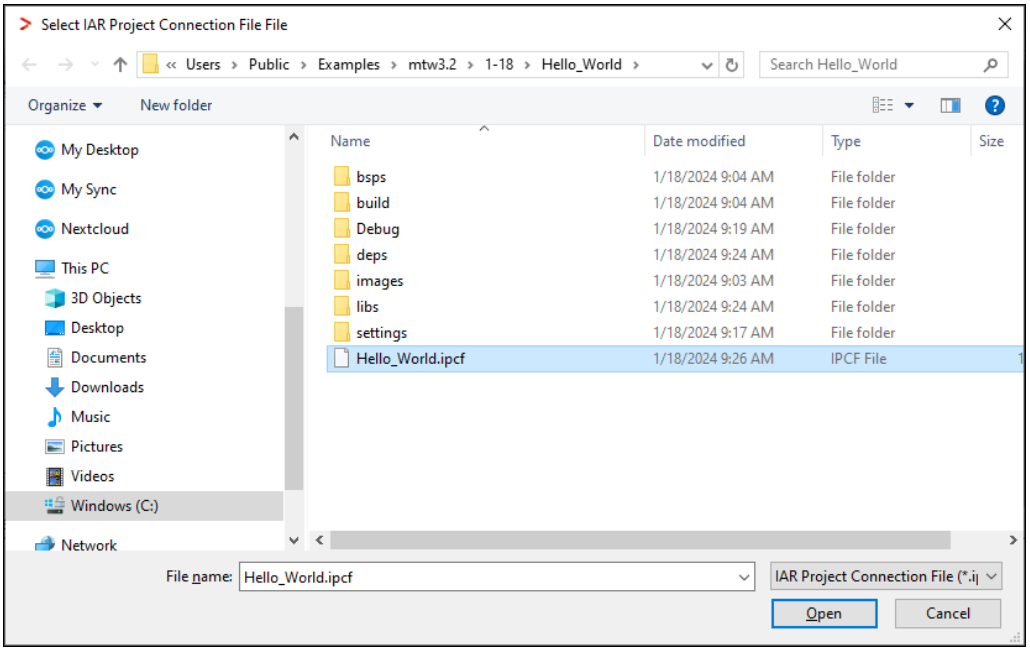
3. Select **Project > Create New Project > Empty project** and click **OK**.

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**.

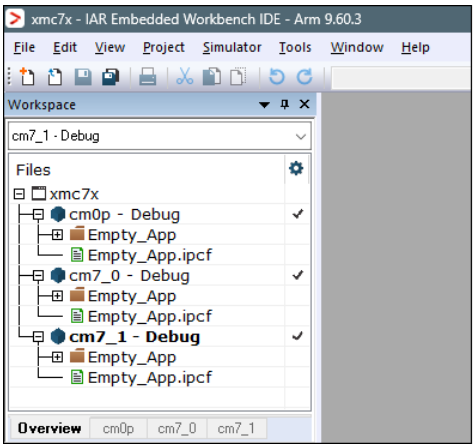


6. On the Select IAR Project Connection File dialog, select the *.ipcf* file and click **Open**:



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:

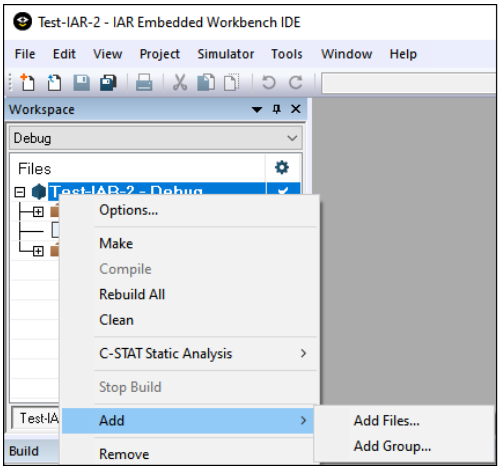
2 Create/export application for IAR Embedded Workbench



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

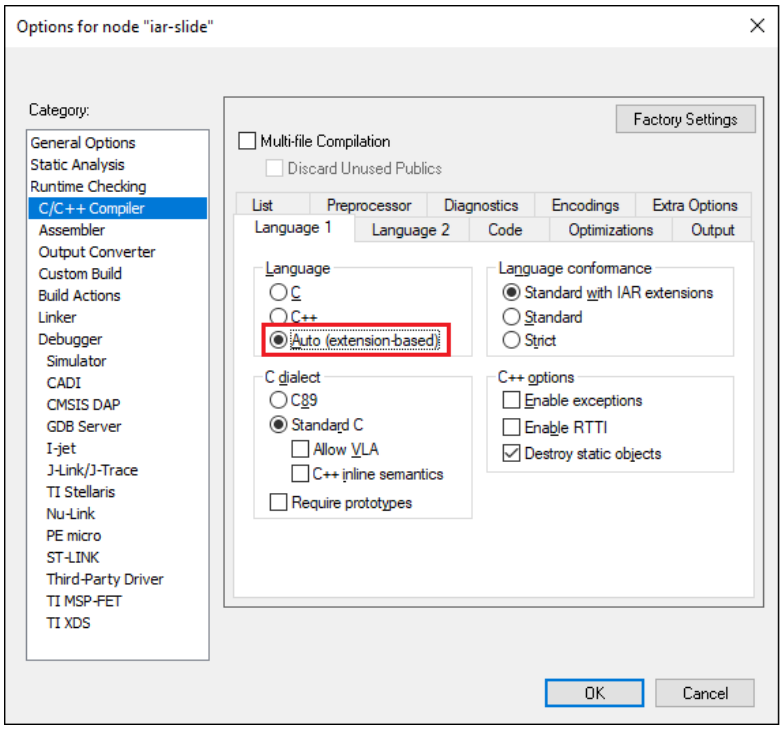
This section covers general instructions that may or may not pertain to your device and application.

3.1 Supported debugger probes

- KitProg3 onboard programmer
- MiniProg4
- IAR I-jet
- J-Link

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**

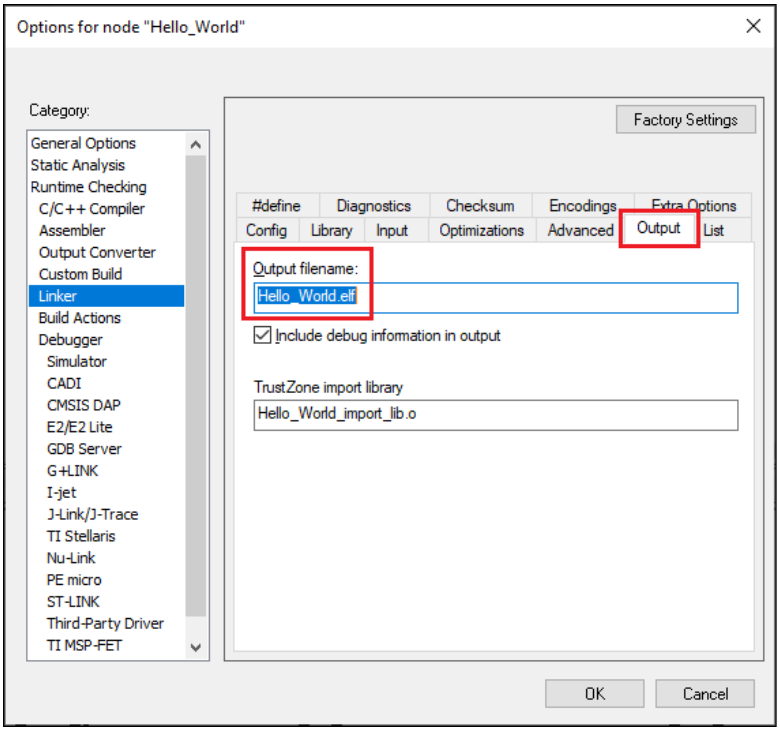
Prebuild and postbuild steps from a ModusToolbox application are not included in the export process by default. If your original application has these steps, you must include them in the IAR Embedded Workbench manually as follows:

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 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*

3.5 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.

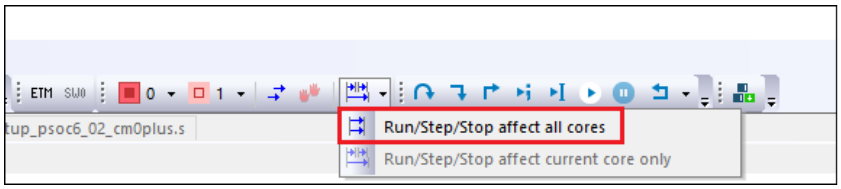
3.6 Perform ETM/ITM trace

Refer application note AN235279 - Performing ETM and ITM trace on PSOC™ 6 MCU for details.

3.7 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™ 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.

4.1 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:

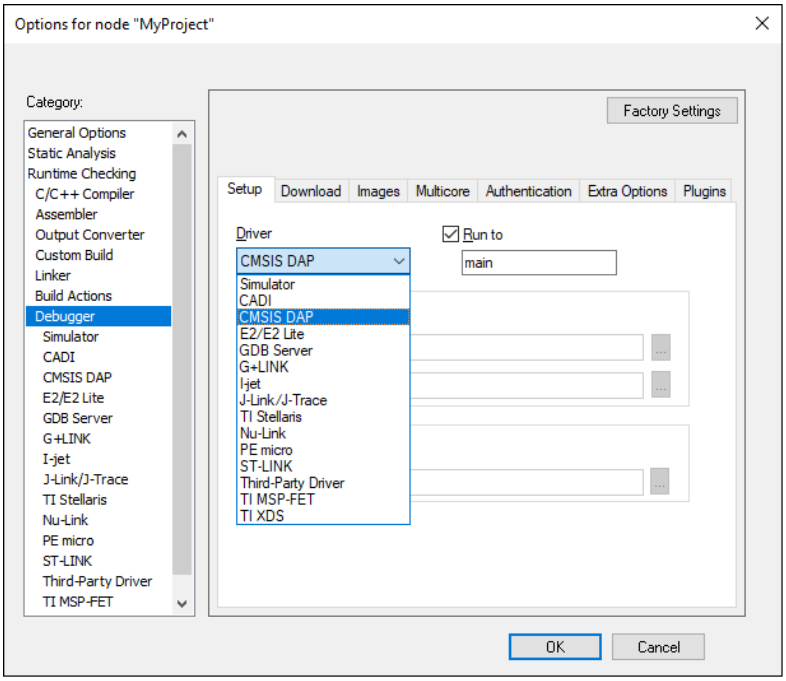
```
Hello_World.out

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

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

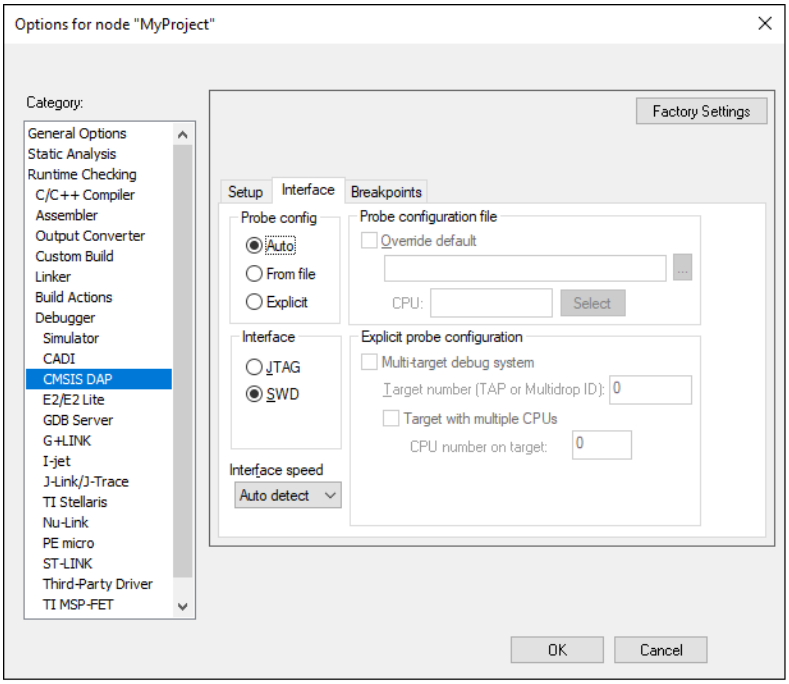
As needed, run the fw-loader tool to make sure the board firmware is upgraded to KitProg3. See the KitProg3 User Guide for details. The tool is in the following directory by default:

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

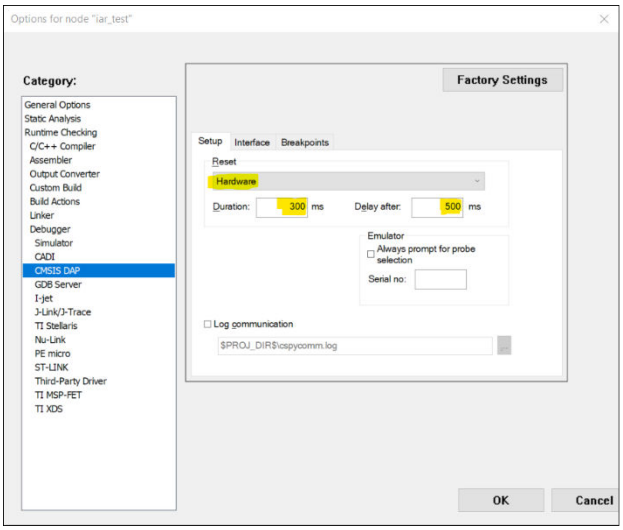
1. In IAR Embedded Workbench, open the Options dialog, go to **Debugger**, and select **CMSIS-DAP** in the driver list:



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



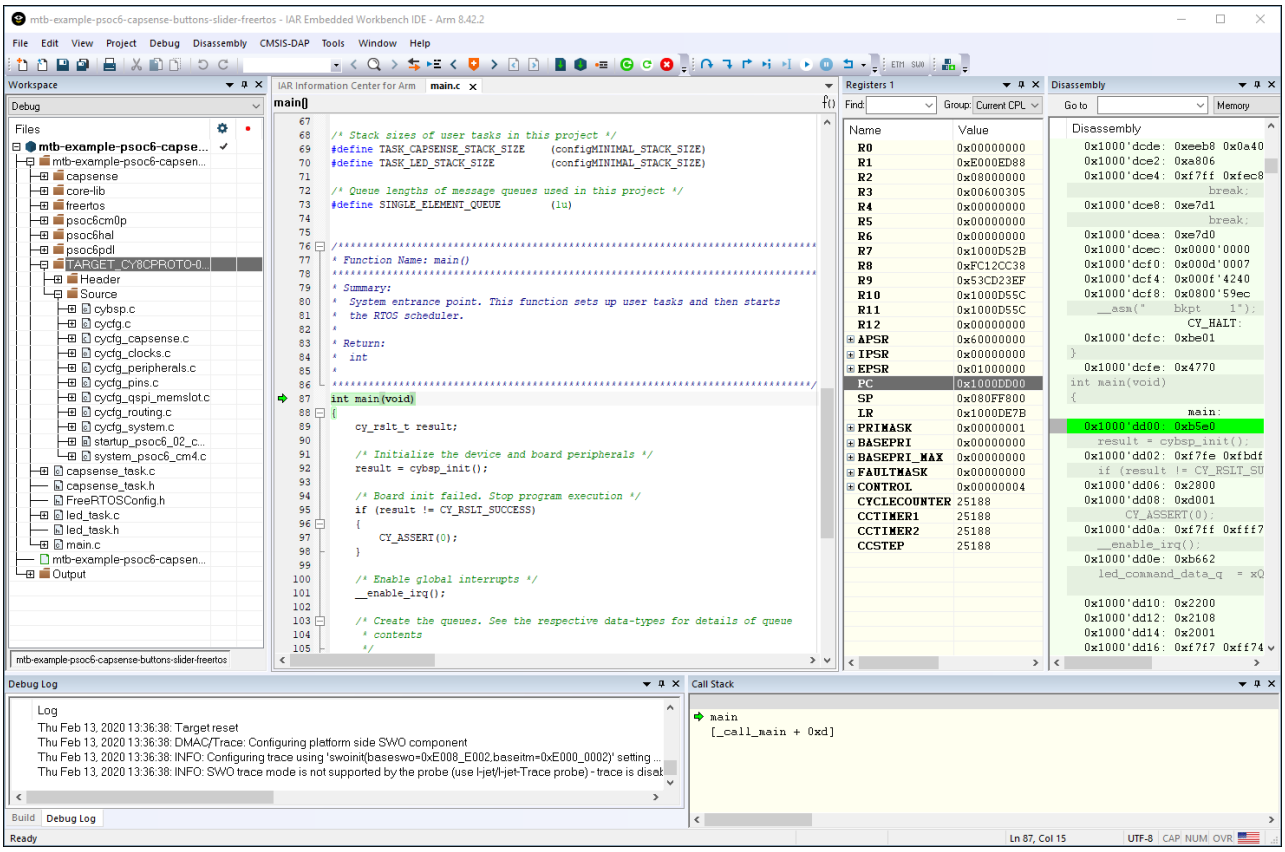
***Note**: When using MiniProg4 with a single-core PSOC™ 6 MCU, you must specify a special type of **Reset** under the **Setup** tab, as follows:*



3. Click **OK**.

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

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

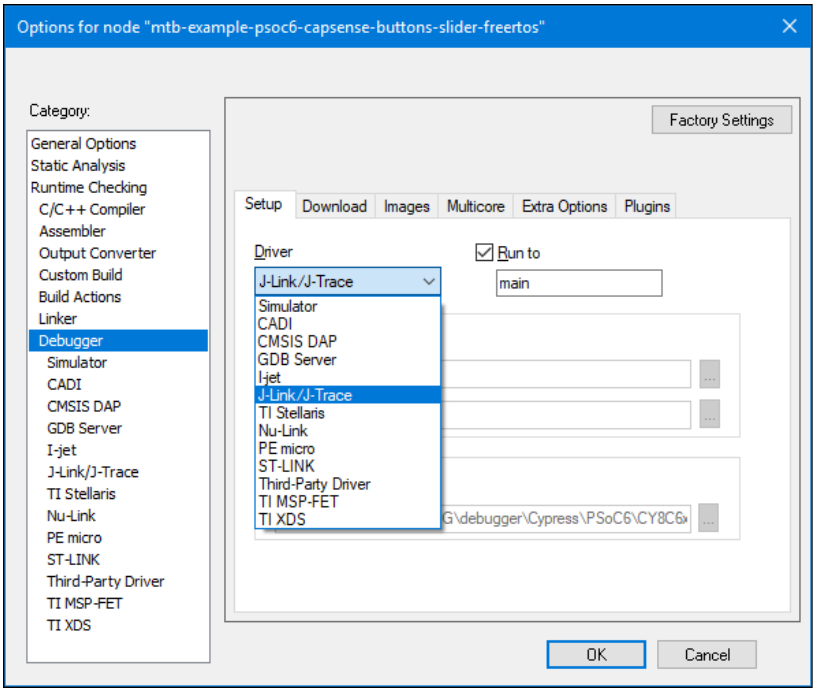


4.3 Program/Debug with J-Link

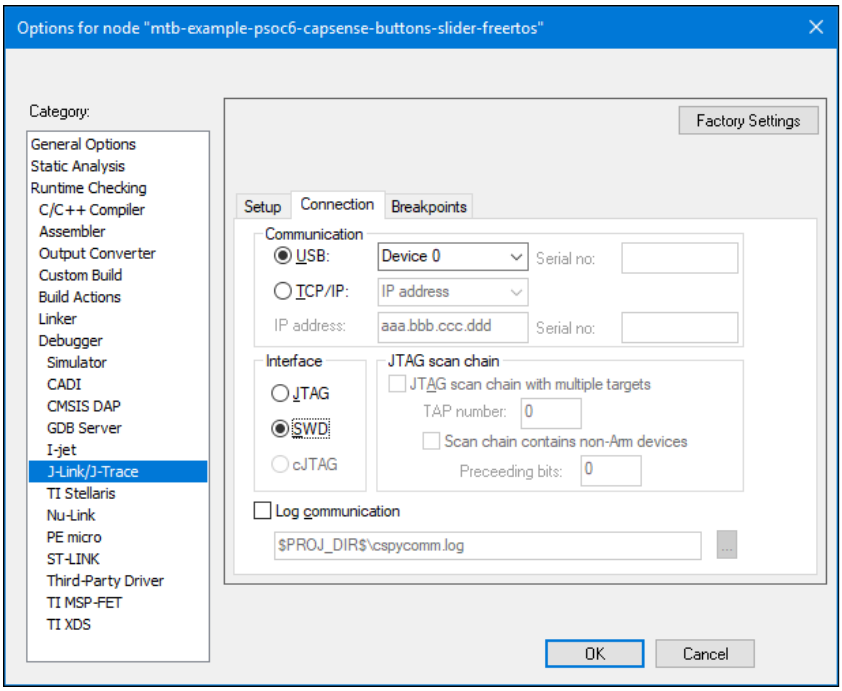
You can use a J-Link debugger probe to debug the application.

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).

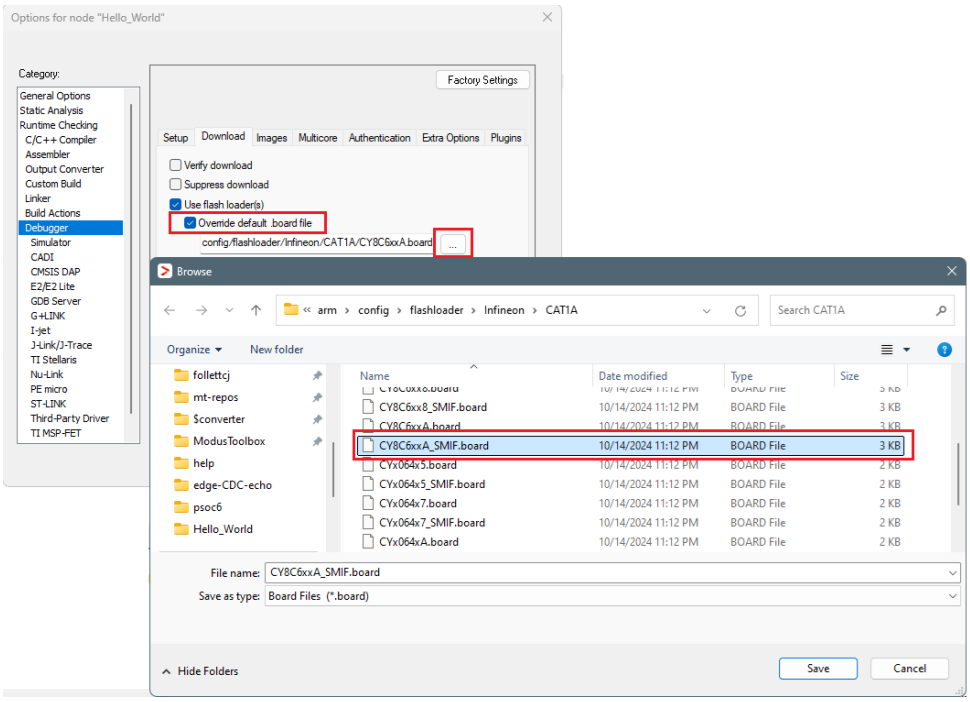
5. Select **Project > Download and Debug** to launch the debugger.

4.4 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.



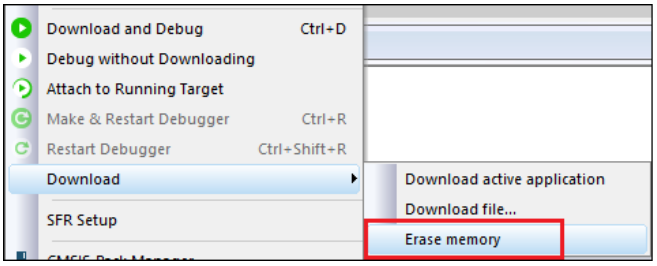
- Identify the default *.board* file currently used for this project.
- Click the **Browse [...]** button, then navigate to and select the same *.board* file that also includes "SMIF".
- Click **Save**.

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

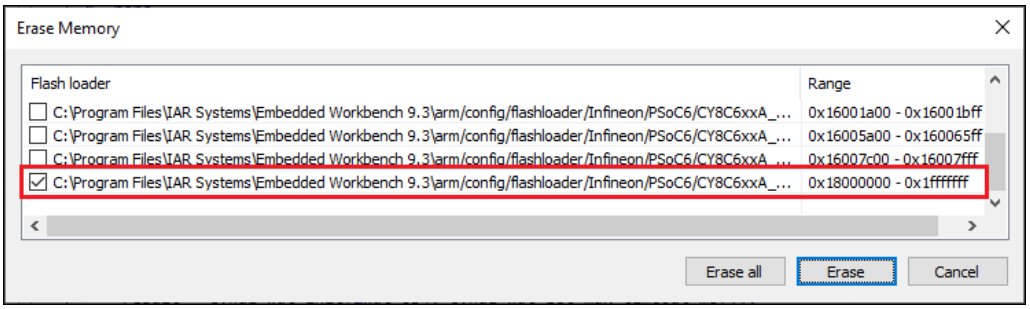
4.5 Erase PSOC™ 6 MCU with external memory enabled

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**.



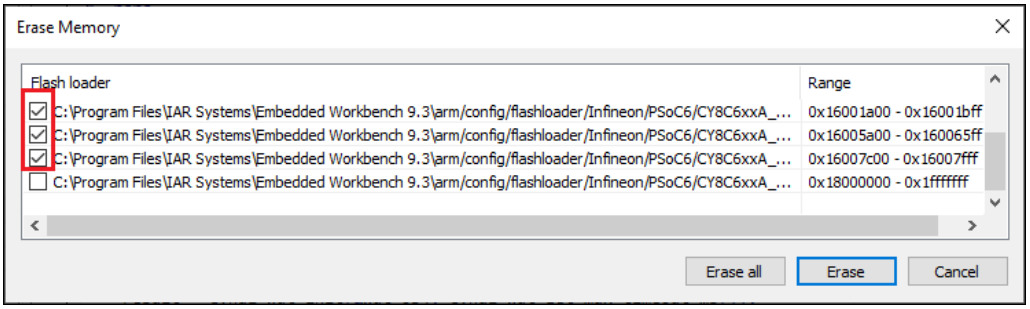
2. Deselect the check boxes for all regions, except for 0x18000000-0x1fffffff.



3. Click **Erase**.

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

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



6. Click **Erase**.

5 PSOC™ 64 secure single-core application

Follow steps in Create/export application for IAR Embedded Workbench. For a PSOC™ 64 secure MCU, you must also follow the instructions in the Secure Boot SDK user guide to provision the device and generate keys.

***Note**: PSOC™ 64 applications require python and cysecuretools to be installed.*

Then, use the instructions in this section to configure, build, program, and debug the application for a PSOC™ 64 secure MCU in IAR Embedded Workbench.

5.1 Configure and build

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

1. Edit the *Makefile* located in the project folder as follows:

```
APPNAME=Hello_World
# Use the same App_Name used to create the project

TOOLCHAIN=IAR
```

2. Open a modus-shell terminal from the ModusToolbox™ install directory, navigate to the IAR project directory, and run this command:

```
make build -j8
```

3. Copy the post-build command from the log. For example:

```
../mtb_shared/core-make/release-v3.6.1/make/scripts/python.bash C:/examples/psoc64-iar/
Hello_World/bsps/TARGET_APP_CY8CPROTO-064S1-SB/psoc64_postbuild.py --core CM4 --secure-
boot-stage single --policy policy_single_CM0_CM4 --target cyb06xx7 --toolchain-
path C:/Users/follettcj/Infineon/Tools/mtb-gcc-arm-eabi/11.3.1/gcc --toolchain IAR
--build-dir C:/examples/psoc64-iar/Hello_World/build/APP_CY8CPROTO-064S1-SB/Debug --app-
name Hello_World --cm0-app-path ../mtb_shared/cat1cm0p/release-v1.8.0/COMPONENT_CAT1A/
COMPONENT_CM0P_SECURE --cm0-app-name psoc6_01_cm0p_secure
```

4. Paste the command into an appropriate editor, and make the following edits:

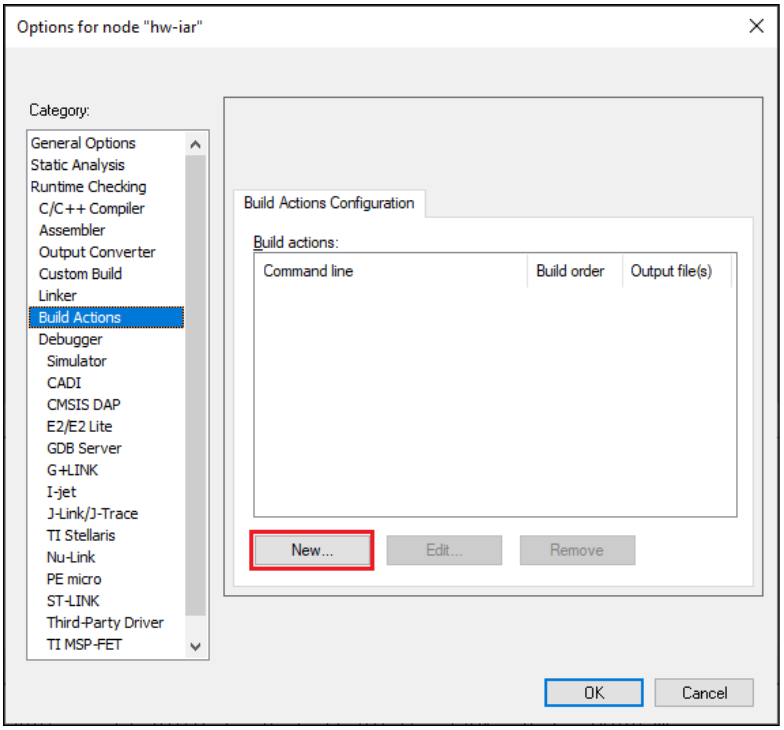
- Update the path to python
- Add path to the policy file, for example: `--policy-path ./policy`
- Change `--build-dir` to `Debug/Exe`
- Change `--app-name` to your project in IAR name, if needed

Example of command after edit:

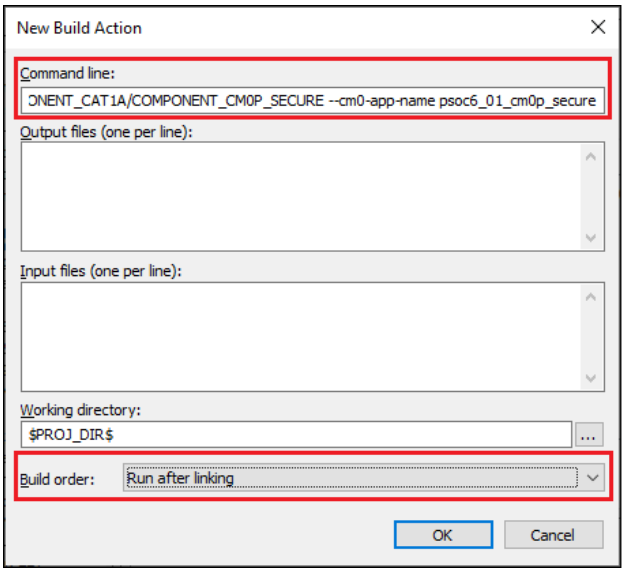
```
[path to python.exe] C:/examples/psoc64-iar/Hello_World/bsps/TARGET_APP_CY8CPROTO-064S1-SB/psoc64_postbuild.py --core CM4 --secure-boot-stage single --policy-path ./policy --policy policy_single_CM0_CM4 --target cyb06xx7 --toolchain-path C:/Users/follettcj/Infineon/Tools/mtb-gcc-arm-eabi/11.3.1/gcc --toolchain IAR --build-dir Debug/Exe --app-name Hello_World --cm0-app-path ../mtb_shared/cat1cm0p/release-v1.8.0/COMPONENT_CAT1A/COMPONENT_CM0P_SECURE --cm0-app-name psoc6_01_cm0p_secure
```

5. Copy the edited command.

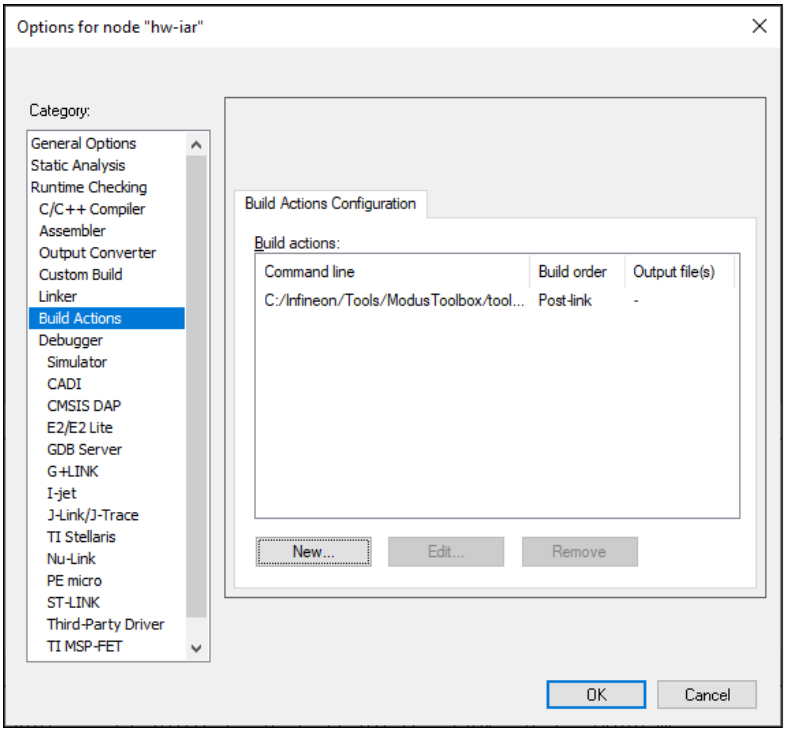
6. In IAR embedded Workbench, open the Options dialog, go to **Build Actions**, and click **New**:



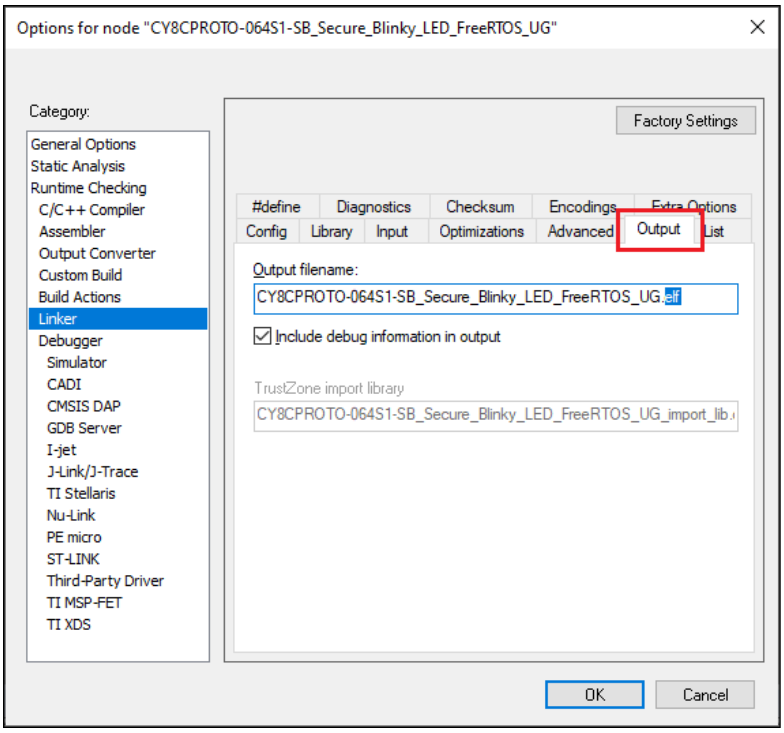
7. On the New Build Action dialog, paste the edited command in the **Command line** field, select "Run after linking" in the **Build order** field, and click **OK**:



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



9. Switch to **Linker > Output** and change the file extension from ".out" to ".elf" in **Output filename** field:



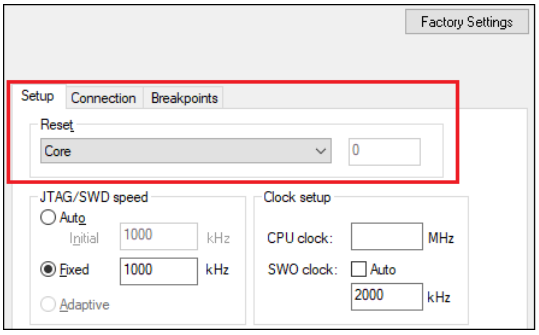
10. Click **OK** to close the dialog.

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

5.2 Program and debug

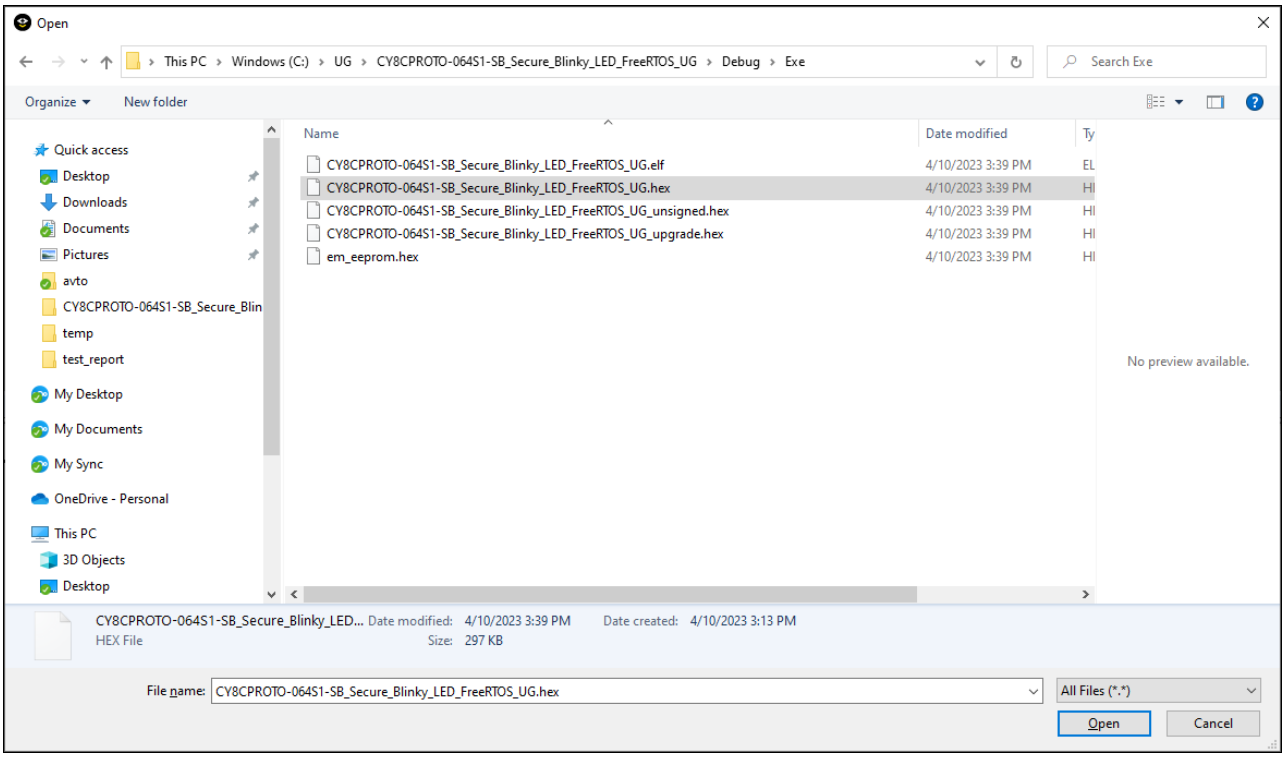
1. You can set the default debugger options for CMSIS-DAP or J-Link. See Program/Debug with KitProg3/MiniProg4 (CMSIS-DAP) or Program/Debug with J-Link.

***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*.

5 PSOC™ 64 secure single-core application



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

6 AIROC™ CYW20829 application

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

Then, use the instructions in this section to configure, build, program, and debug the application for an AIROC™ CYW20829 device in IAR Embedded Workbench.

6.1 Configure and build

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

1. Edit the *Makefile* located in the project folder as follows:

```
APPNAME=Hello_World
# Use the same App_Name used to create the project

TOOLCHAIN=IAR
```

2. Using your favorite code editor, create a *postbuild.bat* file in the IAR project directory.

3. Add `SET PATH=%PATH%;"/usr/bin"` at the start of file.

4. Open a modus-shell terminal from the ModusToolbox™ install directory, navigate to the IAR project directory, and run this command:

```
make build -j8
```

5. Copy the IAR-related post-build output from the modus-shell terminal. For example:

```
C:/iar/ewarm-9.60.3/arm/bin/ielftool C:/examples/20829/Hello_World/build/
APP_CYW920829M2EVK-02/Debug/Hello_World.elf --bin-multi C:/examples/20829/Hello_World/
build/APP_CYW920829M2EVK-02/Debug/Hello_World.bin; C:/iar/ewarm-9.60.3/arm/bin/
ielfdumparm  -a C:/examples/20829/Hello_World/build/APP_CYW920829M2EVK-02/Debug/
Hello_World.elf > C:/examples/20829/Hello_World/build/APP_CYW920829M2EVK-02/Debug/
Hello_World.dis; "[install-path]/ModusToolbox/tools_[version]/modus-shell/bin/bash.exe"
--norc --noprofile ../mtb_shared/recipe-make-cat1b/release-v2.8.0/make/scripts/20829/
flash_postbuild.sh "IAR" "C:/examples/20829/Hello_World/build/APP_CYW920829M2EVK-02/
Debug" "Hello_World" "[install-path]/Infineon/Tools/mtb-gcc-arm-eabi/11.3.1/gcc/bin" "C:/
Users/follettcj/ModusToolbox/tools_3.5/srecord/bin/srec_cat.exe" "0x00003A00"; "[install-
path]/ModusToolbox/tools_[version]/modus-shell/bin/bash.exe" --norc --noprofile ../
mtb_shared/recipe-make-cat1b/release-v2.8.0/make/scripts/20829/run_toc2_generator.sh
"NON_SECURE" "C:/examples/20829/Hello_World/build/APP_CYW920829M2EVK-02/Debug"
"Hello_World" "flash" "bsps/TARGET_APP_CYW920829M2EVK-02" "NONE" "[install-path]/Infineon/
Tools/mtb-gcc-arm-eabi/11.3.1/gcc" "" 0x20000 0 "" "0x00003A00" "256"; [install-path]/
ModusToolbox/tools_[version]/srecord/bin/srec_cat.exe C:/examples/20829/Hello_World/build/
APP_CYW920829M2EVK-02/Debug/Hello_World.final.bin -Binary -offset 0x60000000 -o C:/
examples/20829/Hello_World/build/APP_CYW920829M2EVK-02/Debug/Hello_World.final.hex -Intel
-Output_Block_Size=16;
```

6. Paste the post-build output into the *postbuild.bat* file, make the following edits, and save the file:

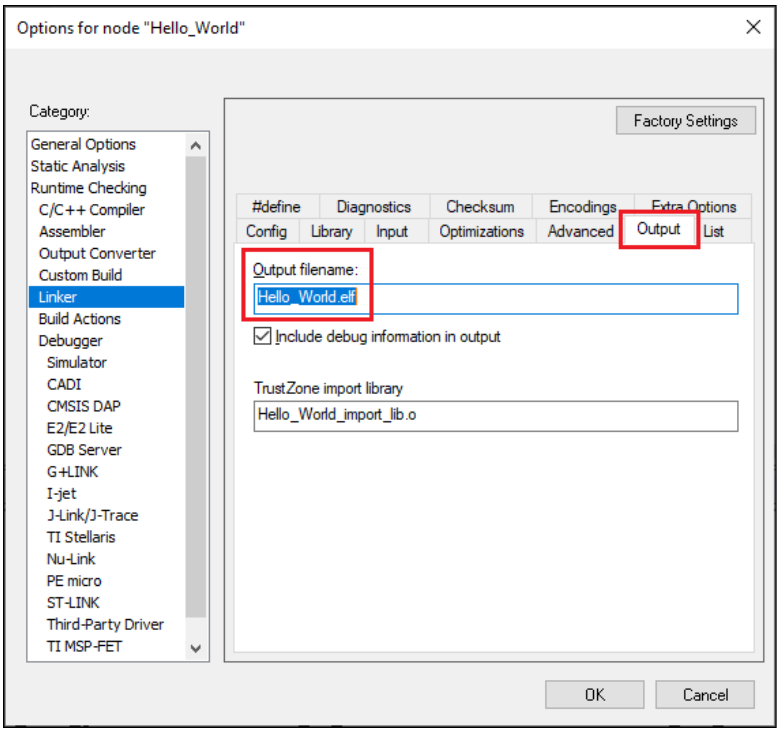
- Change the paths from "build/APP_CYW920829M2EVK-02/Debug/..." to "Debug/Exe/…".
- Remove all the semi-colons. We recommend making a line break for each semi-colon.
- Delete un-needed commands (e.g., ielfdumparm, rm -rf, and cp).

Example edited file:

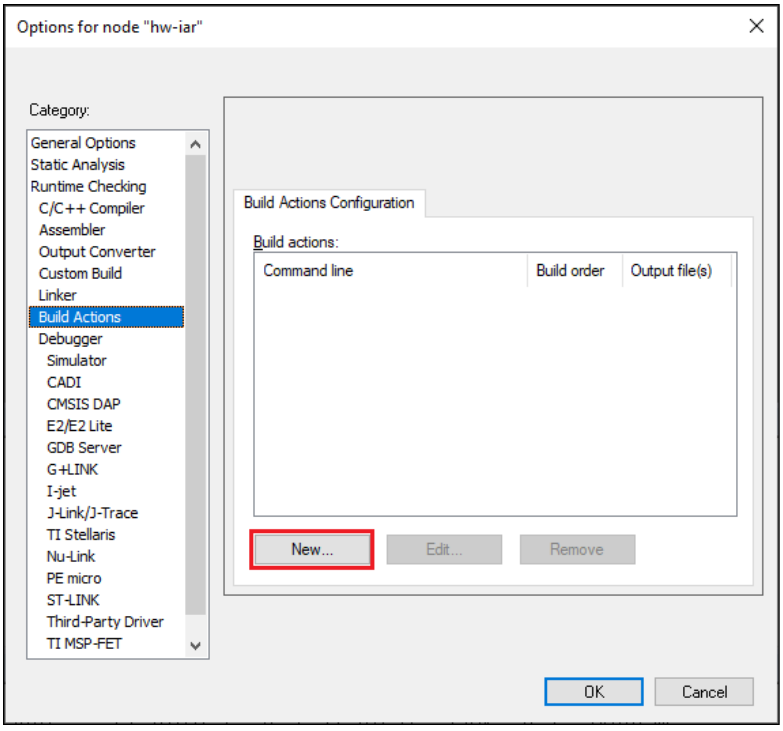
```
PATH=%PATH%;"/usr/bin"

C:/iar/ewarm-9.60.3/arm/bin/ielftool C:/examples/20829/Hello_World/Debug/Exe/
Hello_World.elf --bin-multi C:/examples/20829/Hello_World/Debug/Exe/Hello_World.bin

"[install-path]/ModusToolbox/tools_[version]/modus-shell/bin/bash.exe" --
norc --noprofile ../mtb_shared/recipe-make-cat1b/release-v2.8.0/make/scripts/20829/
flash_postbuild.sh "IAR" "C:/examples/20829/Hello_World/Debug/Exe"
"Hello_World" "[install-path]/Infineon/Tools/mtb-gcc-arm-eabi/11.3.1/gcc/bin" "[install-
path]/ModusToolbox/tools_[version]/srecord/bin/srec_cat.exe" "0x00003A00"

"[install-path]/ModusToolbox/tools_[version]/modus-shell/bin/bash.exe" --
norc --noprofile ../mtb_shared/recipe-make-cat1b/release-v2.8.0/make/scripts/20829/
run_toc2_generator.sh "NON_SECURE" "C:/examples/20829/Hello_World/Debug/Exe"
"Hello_World" "flash" "bsps/TARGET_APP_CYW920829M2EVK-02" "NONE" "[install-path]/Infineon/
Tools/mtb-gcc-arm-eabi/11.3.1/gcc" "" 0x20000 0 "" "0x00003A00" "256"

[install-path]/ModusToolbox/tools_[version]/srecord/bin/srec_cat.exe "Debug/Exe/
Hello_World.final.bin" -Binary -offset 0x60000000 -o "Debug/Exe/Hello_World.final.hex"
-Intel -Output_Block_Size=16
```

***Note**: Make sure to update the paths as needed for your system.*

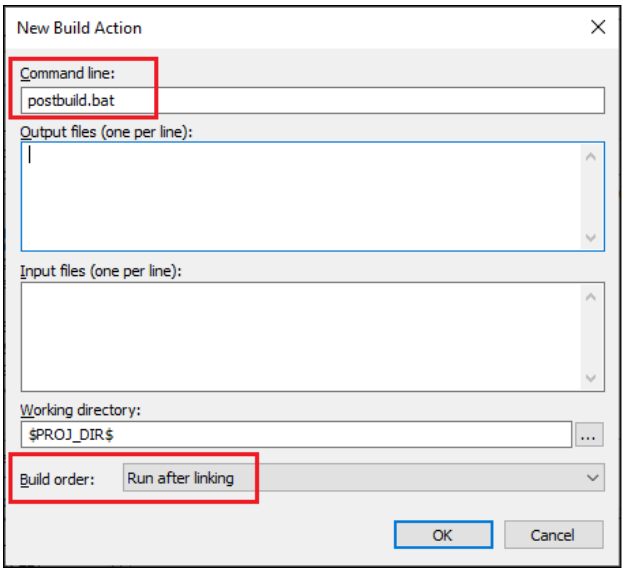
7. In IAR Embedded Workbench, open the Options dialog, go to **Linker > Output** and change the file extension from ".out" to ".elf" in the **Output filename** field:

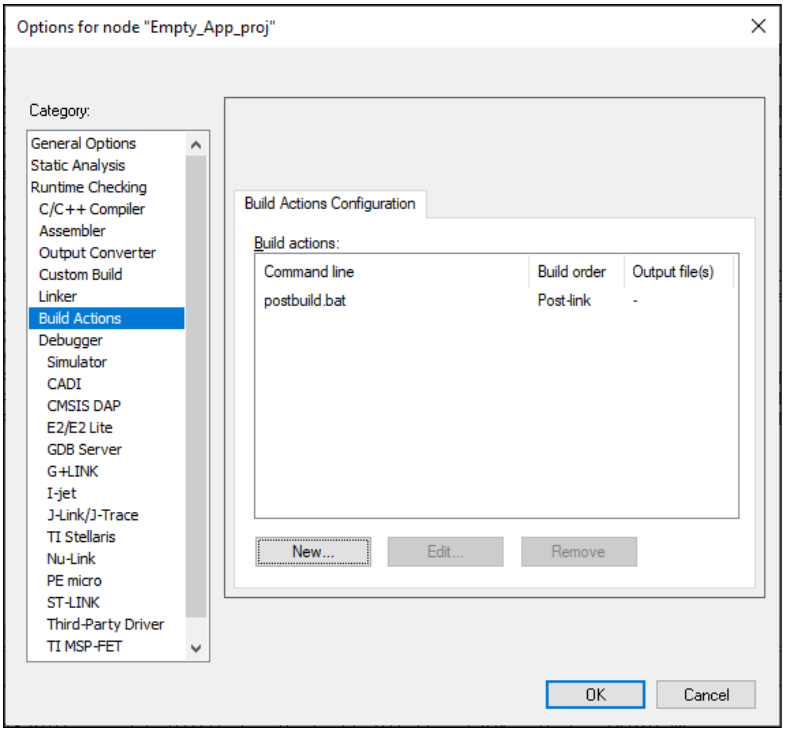


8. Switch to **Build Actions** and click **New**:



9. On the New Build Action dialog, type the *postbuild.bat* file in the **Command line** field, select "Run after linking" in the **Build order** field, and click **OK**:





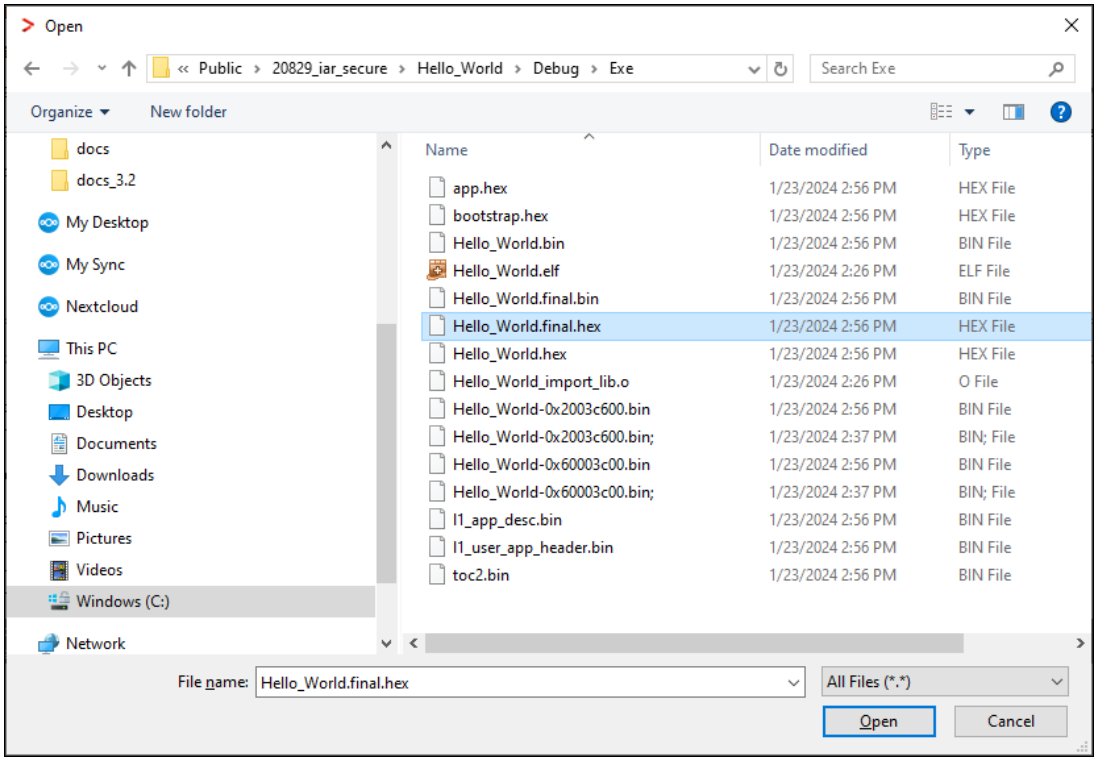
10. Click **OK** to close the Options dialog.

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

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

6.2 Program and debug

Select **Project > Download > Download file…** and select the *<project_name>.final.hex* file in *<project_root>\Debug\Exe* [you might have to switch to All Files (*.*)].



Select **Project >Debug without Downloading**.

7 PSOC™ Control C3 secure application

Follow steps in Create/export application for IAR Embedded Workbench. At the end of the process, you should see messages in the console. For a PSOC™ Control C3 device, one of the messages should read as follows:

```
The project is an ARM Trustzone secure project. The following need to be enabled in Embedded
Workbench IDE.
    Project->Options->General Options->32 bit->TrustZone
```

In most cases, PSOC™ Control C3 applications are set to Trust Zone Secure by default. For more details about TrustZone technology, refer to the Arm® website: https://www.arm.com/technologies/trustzone-for-cortex-m.

When you open the application in IAR Embedded Workbench, you need to check the TrustZone setting. Open the Options dialog, go to **General Options > 32-bit** and verify that the TrustZone **Mode** is set to "Secure."

Save the application and select **Project > Make** to build it. 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
```

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*

7.1 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. This is also referred to as Development LCS. There is nothing to configure before programming and debugging in this state. Go the Program and debug section.

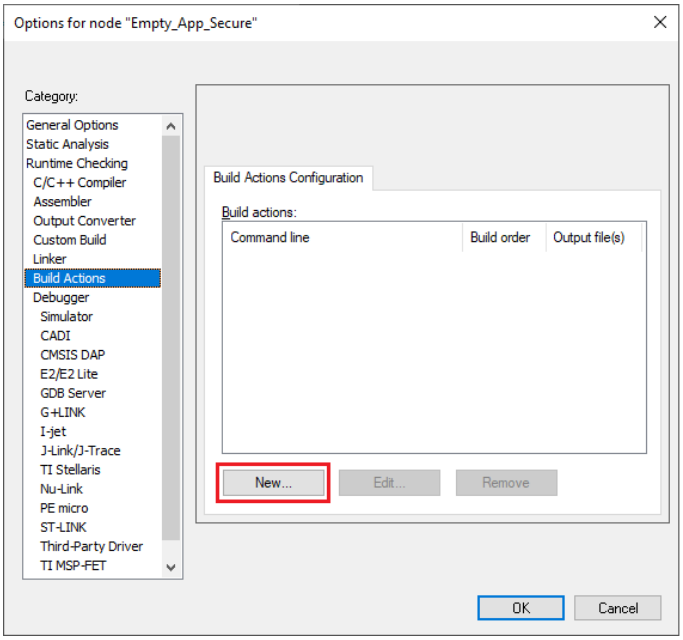
7.2 Provisioned device

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. To do this, add a *postbuild.bat* file with a command similar to the following, all on one line. Update the paths, file names, and key to match your configuration.

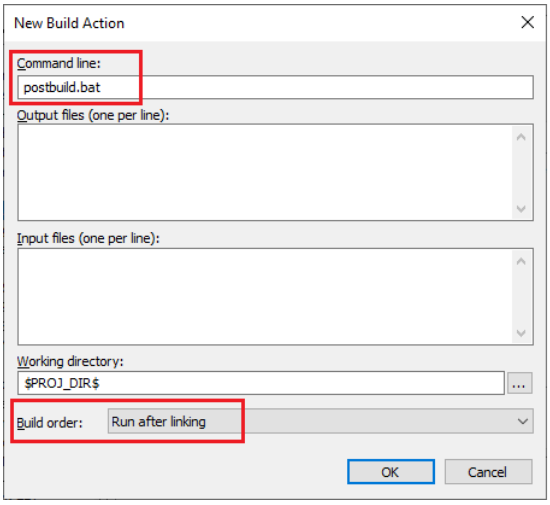
```
[path-to-edgeprotecttools]\bin\edgeprotecttools sign-image --image "Debug/Exe/Empty_App.hex"
--output "Debug/Exe/Empty_App.hex" --header-size 0x400 --slot-size 0x20000 --key keys/
oem_rot_priv_key_0.pem --hex-addr 0x32000000 --align 1 --pad --security-counter 0 --erased-val
0 --public-key-format full --pubkey-encoding raw --signature-encoding raw --min-erase-size
0x200 --overwrite-only
```

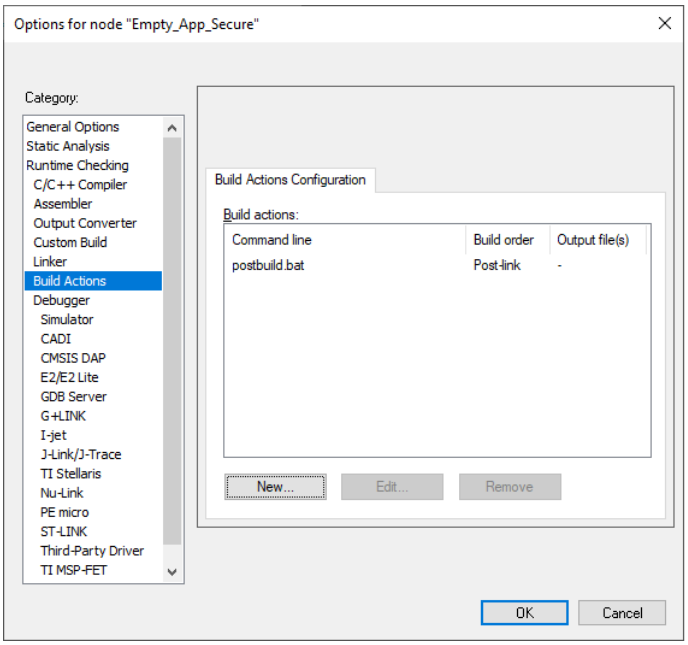
Then, there are a few settings to configure the device before programming and debugging.

1. Open the options dialog to **Build Actions** and click **New**:

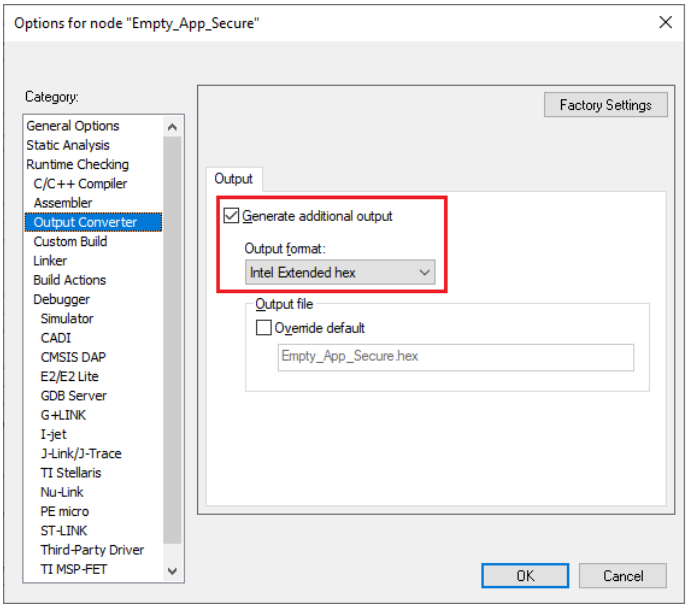


2. On the New Build Action dialog, type the *postbuild.bat* file in the **Command line** field, select "Run after linking" in the **Build order** field, and click **OK**:





3. Switch to **Output Converter**, enable **Generate additional output**, and select "Intel Hex file" in **Output format**:



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

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

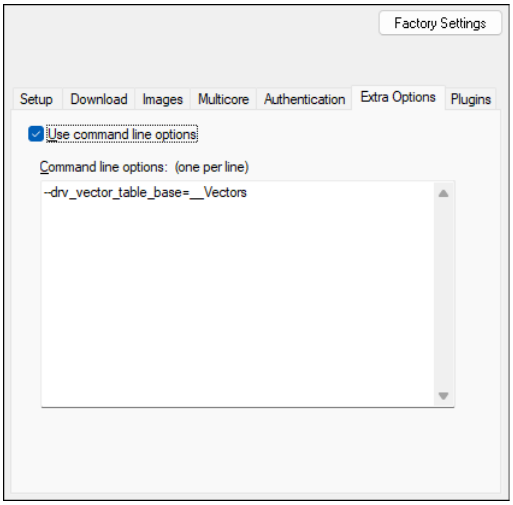
Go the Program and debug section.

7.3 Program and debug

Before programming and debugging, the default debugger options for your desired probe. See Program/Debug with KitProg3/MiniProg4 (CMSIS-DAP) or Program/Debug with J-Link.

1. Open the options dialog to the **Extra Options** tab, select the **Use command line options** check box, and paste the following command-line option:

```
--drv_vector_table_base=__Vectors
```

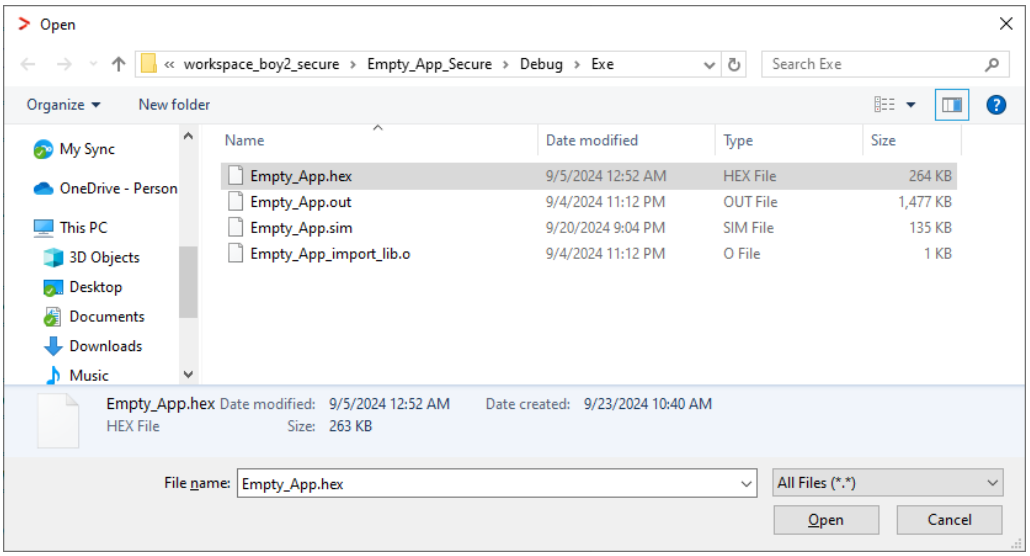


This is needed to resolve an issue with an invalid for IAR name for the vector table in the C start-up code.

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

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

4. Select **Project > Download > Download file...** and select the *<project_name>.hex* file in *<project_root>\Debug\Exe* [you might have to switch to **All Files (*.*)**].



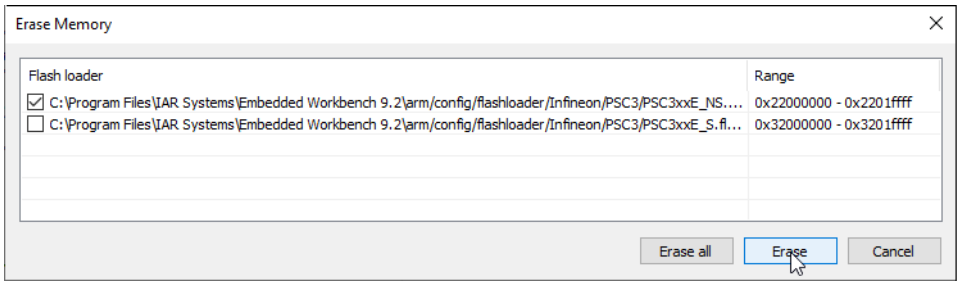
5. Select **Project >Debug without Downloading**.

7 PSOC™ Control C3 secure application

You may encounter flash access restrictions based on the policy that was used to provision the target. The same physical flash memory can be erased/programmed over S-BUS either over Non-Secure addresses (0x22000000-0x2203FFFF) or over Secure addresses (0x32000000-0x3203FFFF).

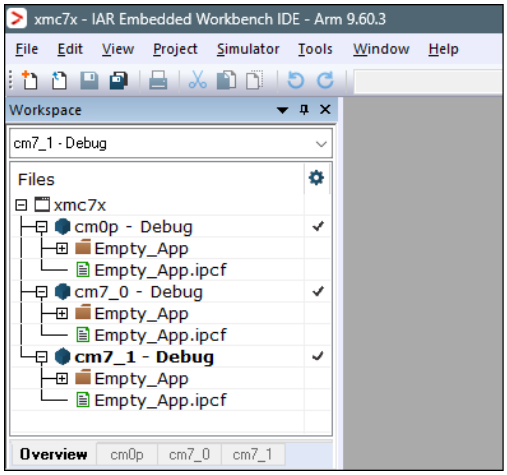
During flash programming operations, the CPU core operates in the Secure world. In this state, the Non-secure addresses are always accessible, but Secure addresses may be unavailable if certain regions are configured for Non-secure access (for PSOC™ Control C3 it means Non-secure access only).

To erase the entire flash memory, it's important to only erase the non-secure range of addresses. Just deselect the secure address range:



8 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:

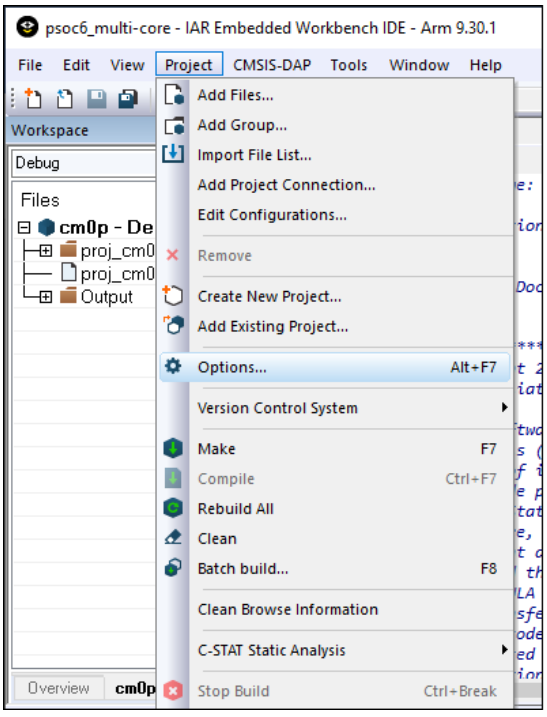


Then, use the instructions in this section to configure, build, program, and debug a multi-core application for PSOC™ 6 and XMC7000 multi-core applications in IAR Embedded Workbench.

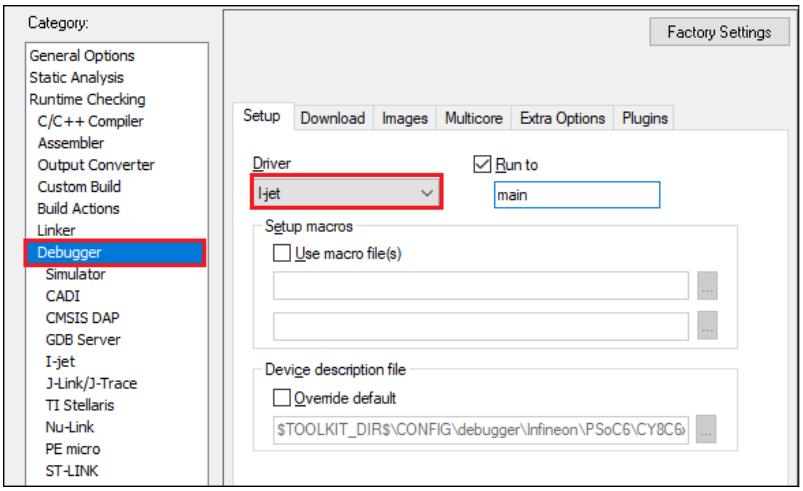
Before you can launch a multi-core debug session, all projects within the workspace must be properly configured. In IAR there is a concept of 'master' and 'slave' projects. Configure the CM0+ core project as the master project, and configure the other cores (CM4 for PSOC™ 6 and CM7 for XMC7000) as slave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

8.1 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 your all projects for CM4/CM7 (for triple-core MCUs),

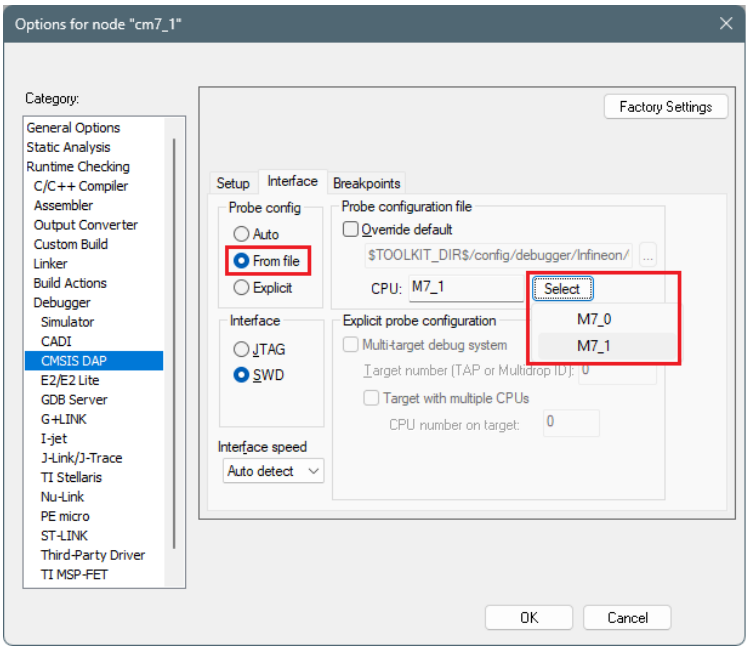
8.1.1 XMC7000/TRAVEO™ II specific steps

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.

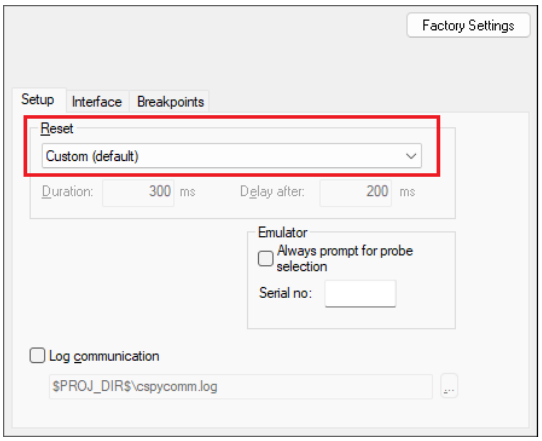
1. Select the project for the second CM7 core and go to **Project > Options**.

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**:



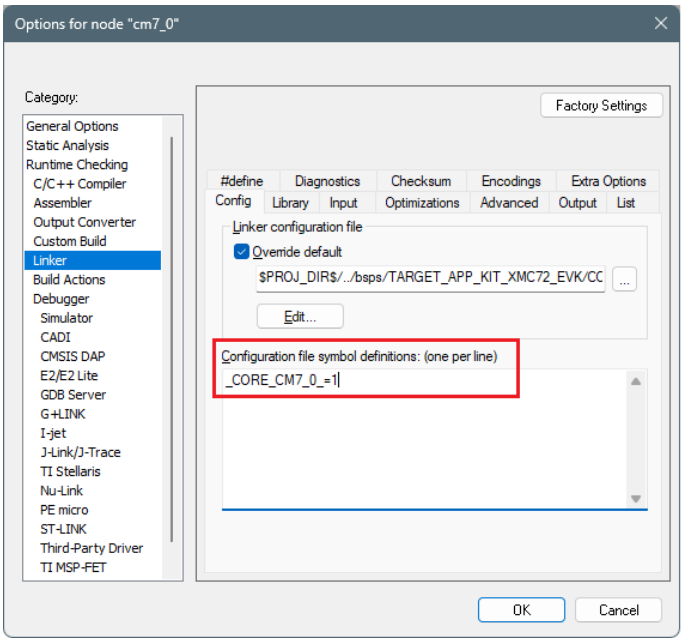
4. Switch to the **Setup** tab, and select "Custom" from the **Reset** pull-down menu.



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**.

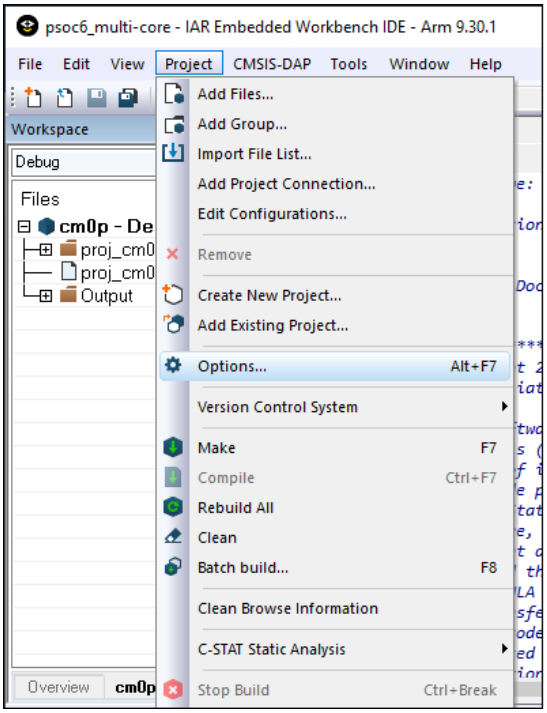
   b. Add `_CORE_CM7_1_=1` in the **Configuration file symbol definitions field**, and click **OK**.

   ***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.*

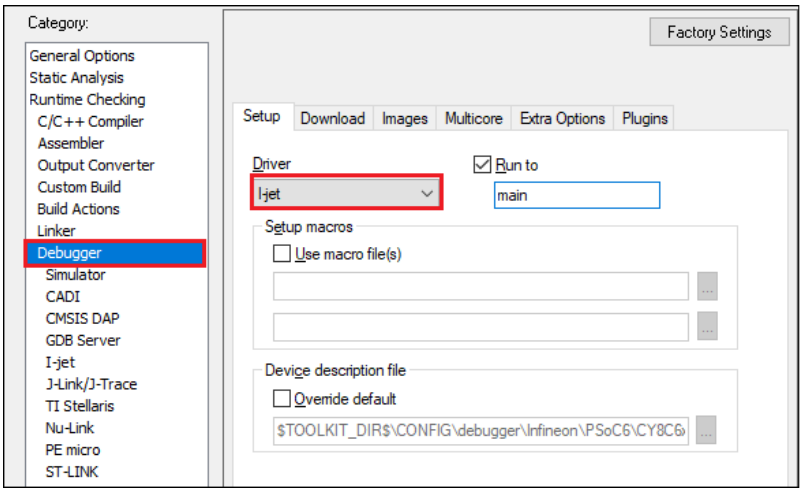
7. Build your CM7 project(s) before moving forward.

8.2 Configure and build CM0+ (master) core

1. Select the CM0+ project and go to **Project > Options**:

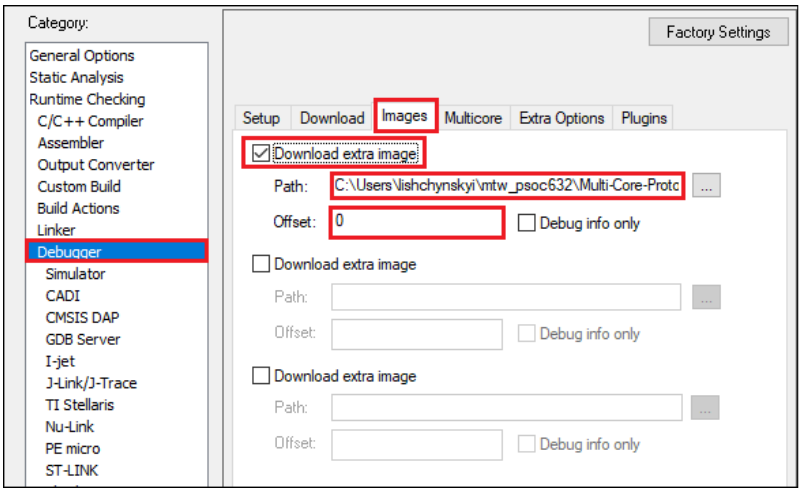


2. On the dialog, select the **Debugger** category in the **Setup** tab, and then select the applicable **Driver** (I-jet, CMSIS-DAP, J-Link):



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**.

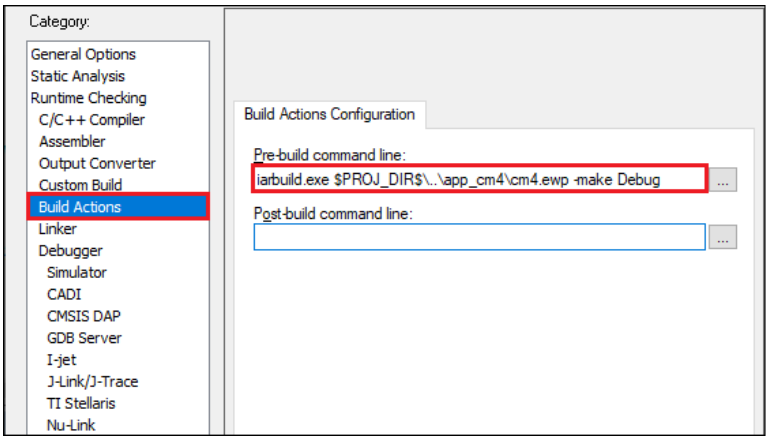


If you provide an OUT file instead of a HEX file, the IAR IDE will fail to halt at the beginning of `main()` due to the main function present in both the CM0+ and CM4/CM7 OUT files.

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

4. Add a prebuild command to build all projects prior to programming/debugging. In the **Build Actions** category set **Pre-build command line** to:

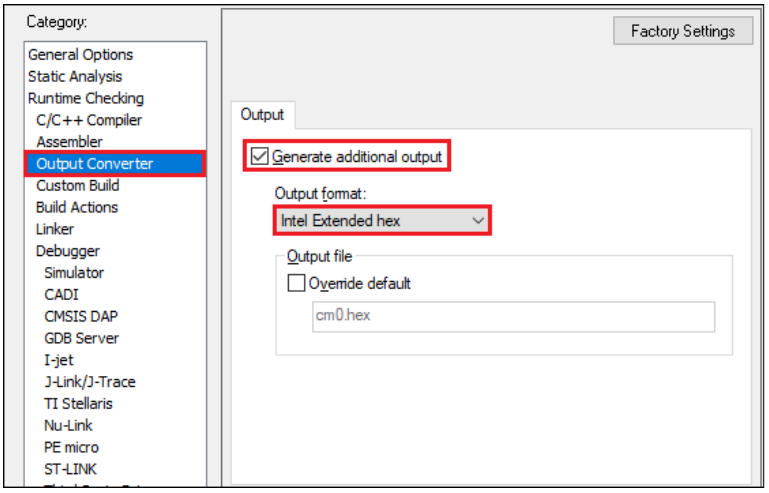
```
iarbuild.exe "<cm4/cm7_proj_loc>.ewp" -make Debug
```



5. If your MCU has three cores, you might want to also specify a post-build action to build the project for the third core in the same manner.

6. 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**.



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

8. Build the project.

8.3 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>
```

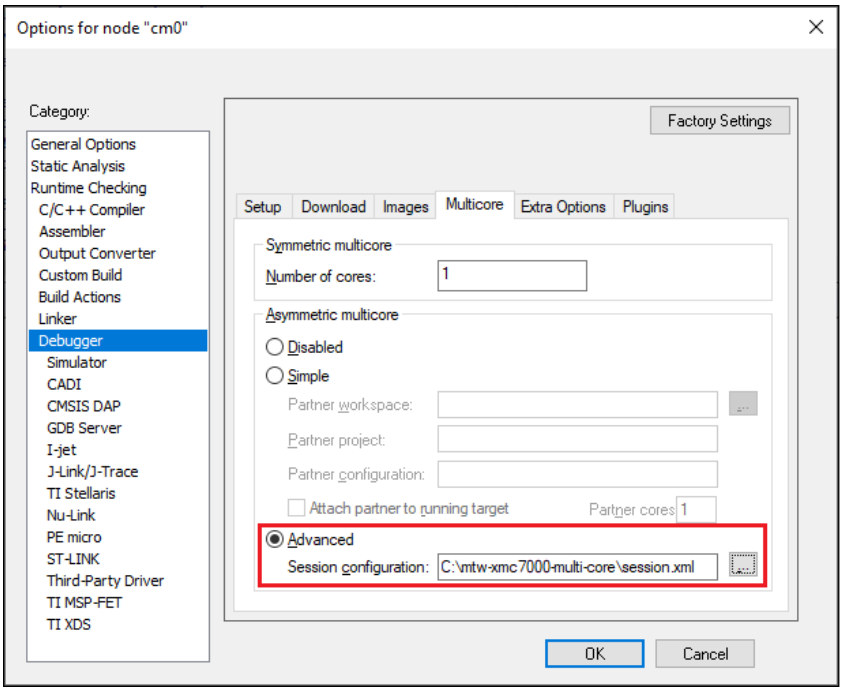
2. Configure multi-core debugging for the CM0+ project.

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**.

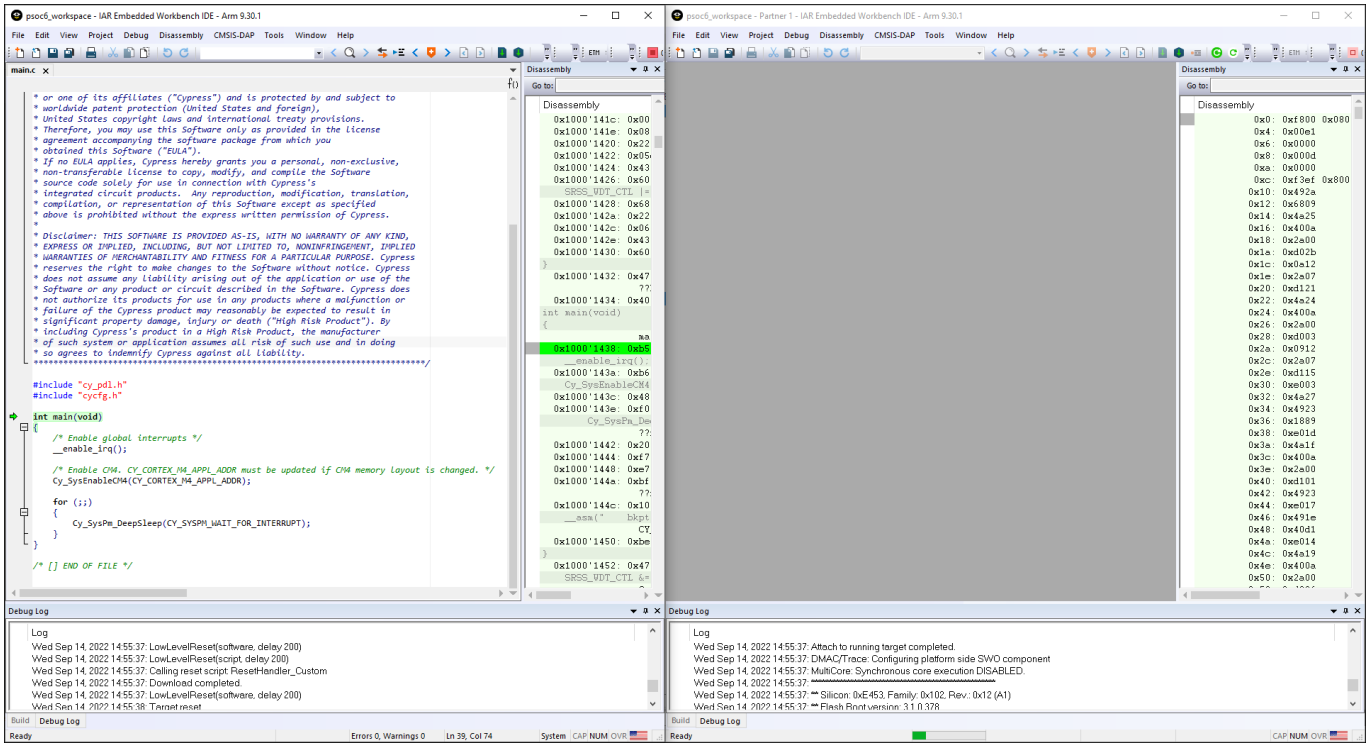


3. Save the workspace.

8.4 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.

8.5 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, thus 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**.

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.

Trademarks

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

Edition 2025-04-28

Published by

Infineon Technologies AG

81726 Munich, Germany

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

Email: erratum@infineon.com

Document reference
IFX-efq1743008065249

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.

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.