IAR Embedded Workbench for ModusToolbox™ user guide

ModusToolbox™ tools package version 3.1.0

About this document

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.

Document conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, menus and sub-menus.
Italics	Denotes file names and paths.
`Courier New`	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™ tools package installation guide –Provides information and instructions about installing the tools package 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

1 Download/install software

1.1 ModusToolbox™ tools package

Refer to the instructions in the ModusToolbox™ tools package installation guide for how to download and install the ModusToolbox™ tools package.

1.2 IAR Embedded Workbench (Windows only)

IAR Embedded Workbench version 8.42.2 or later. Recommended version 9.32.1.

1.3 Python

Python 3.8 is installed in the tools_3.x directory, and the make build system has been configured to use it. You don't need to do anything if you use the modus-shell/Cygwin.bat file to run command line tools.

However, if you plan to use your own version of Cygwin or some other type of bash, you will need to ensure your system is configured correctly to use Python 3.8. Use the `CY_PYTHON_PATH` as appropriate.

1.4 J-Link

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

https://www.segger.com/downloads/J-Link/J-Link_Windows.exe

2 Getting started

This section covers the ways to get started using IAR Embedded Workbench with ModusToolbox™ software.

- Create new application
- Exporting existing application
- Open application in IAR Embedded Workbench
- Build the application

2.1 Create new application

Creating an application includes several steps, as follows:

2.1.1 Step 1: Open Project Creator tool

The ModusToolbox™ Project Creator tool is used to create applications based on code examples and template applications. By default, the tool is installed in the following directory:

<user_home>/ModusToolbox/tools_<version>/project-creator

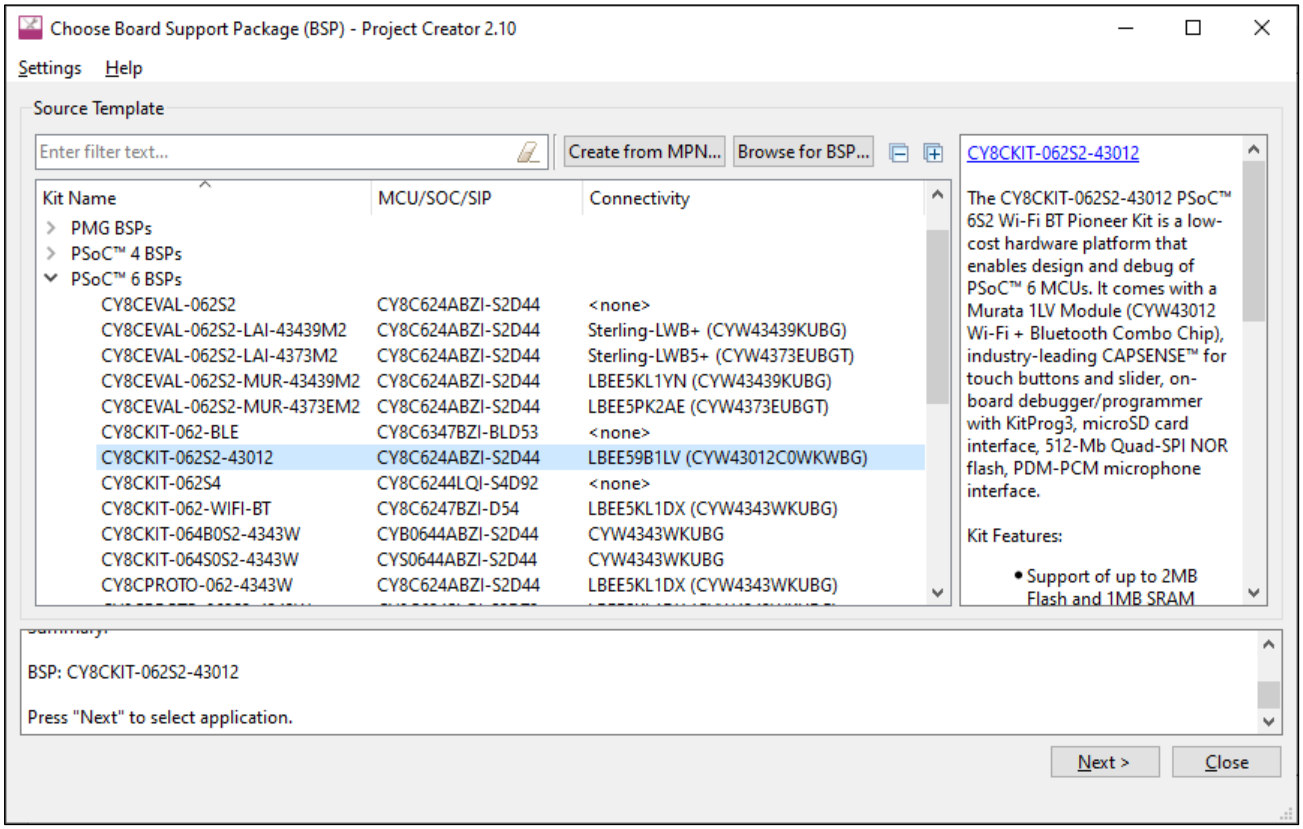
The tool is provided in GUI form and as a command line interface. For more details, refer to the Project Creator user guide.

You can open the tool from the Windows **Start** menu, or by launching the executable in the installation directory.

Note: *You can also launch the Project Creator tool from the ModusToolbox™ Dashboard. Refer to the Dashboard user guide for more details.*

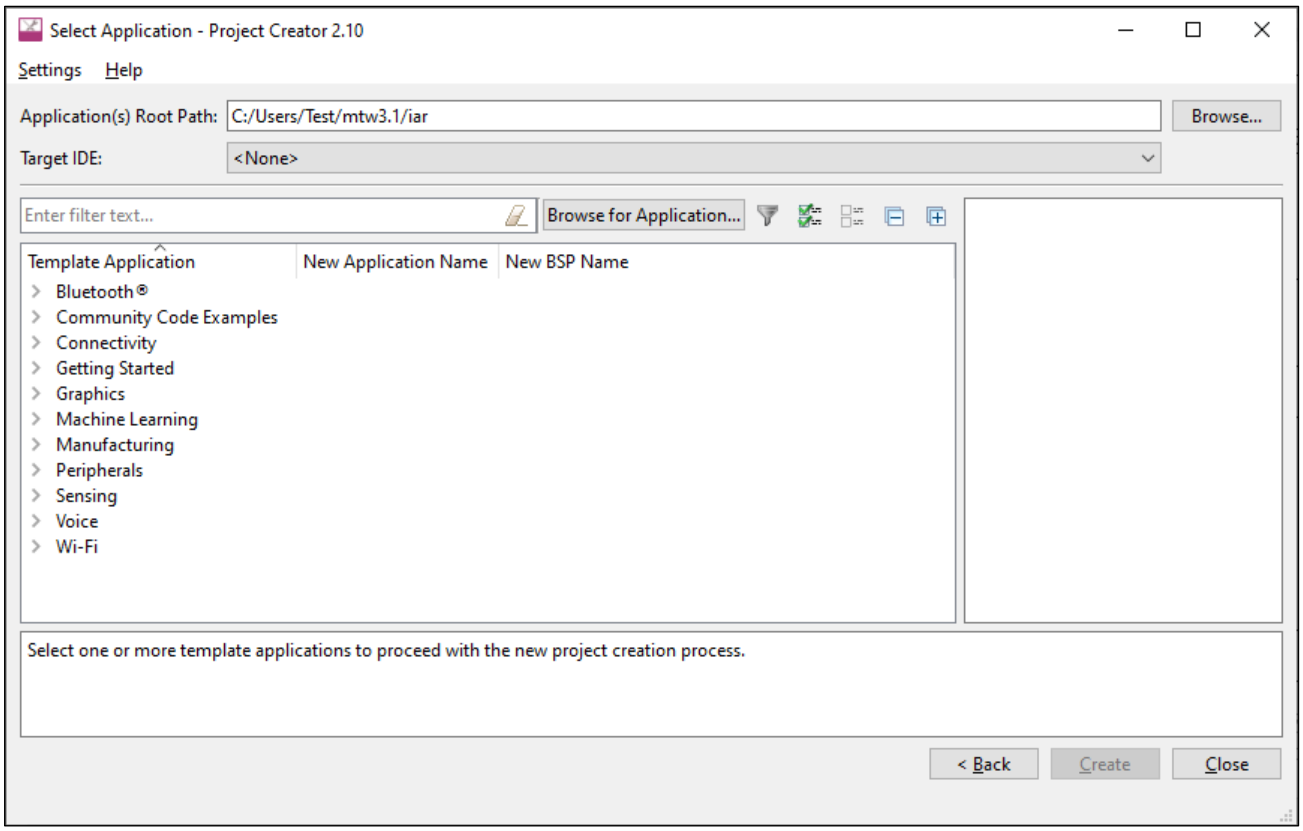
2.1.2 Step 2: Choose Board Support Package (BSP)

When the Project Creator tool opens, expand one of the BSP categories under **Kit Name** and select an appropriate kit; see the description for it on the right. For this example, select the **CY8CKIT-062S2-43012** kit. The following image is an example; the precise list of boards available in this version will reflect the platforms available for development.



2.1.3 Step 3: Select application

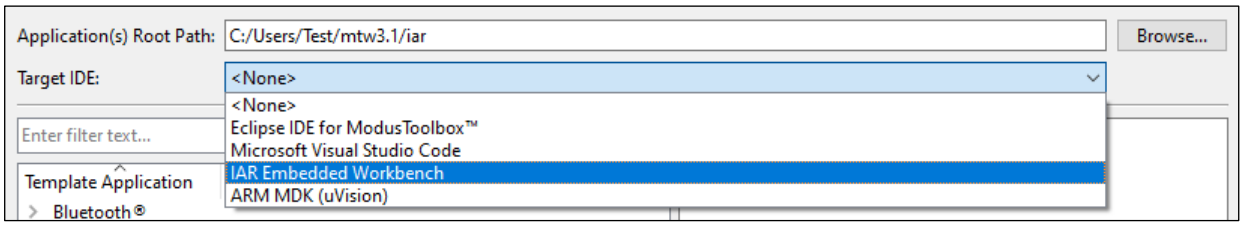
Click **Next >** to open the Select Application page.



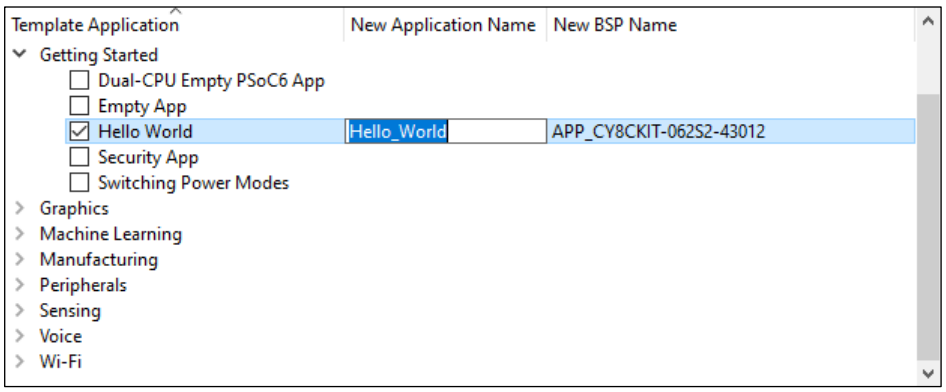
This page displays example applications, which demonstrate different features available on the selected BSP. In this case, the CY8CKIT-062S2-43012 provides the PSoC™ 62 MCU and the AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip. You can create examples for PSoC™ 6 MCU resources such as CAPSENSE™ and QSPI, as well as numerous examples for other capabilities.

Click **Browse…** next to **Application(s) Root Path** to create or specify a folder where the application will be created.

Pull down the **Target IDE** menu, and select **IAR Embedded Workbench**.



Under the **Template Application** column, expand **Getting Started** and select **Hello World** from the list. This example exercise uses the PSoC™ 6 MCU to blink an LED.



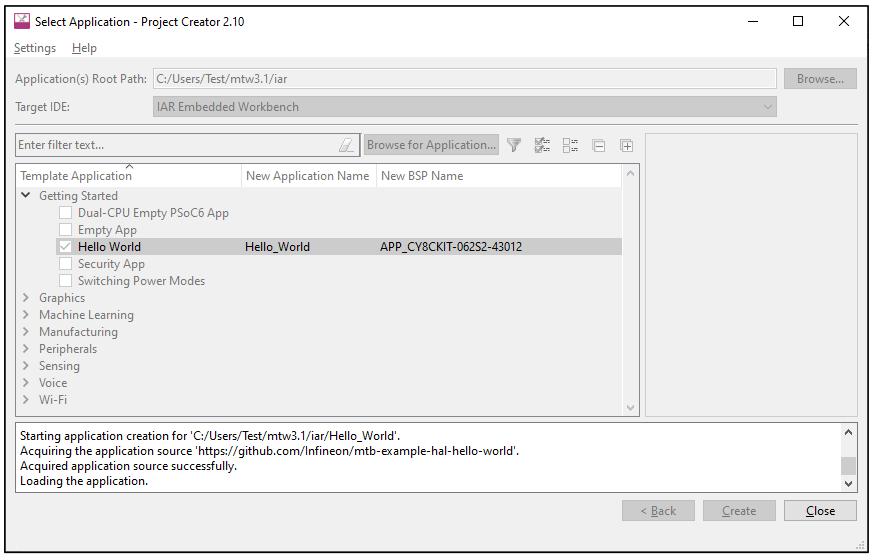
Note: The actual application names available might vary.

Type a name for your application and/or BSP, or leave the default names. Do not use spaces. Also, do not use common illegal characters, such as:

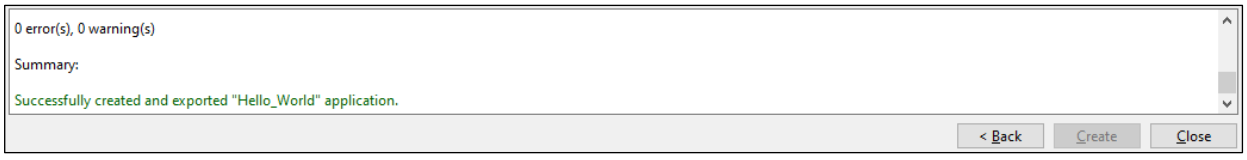
```
* . " ` / \ [ ] : ; | = ,
```

2.1.4 Step 4: Create application

Click **Create** to start creating the application. The tool displays various messages.



When the process completes, a message states that the application was created. Click **Close** to exit the Project Creator tool.



2.2 Export existing 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 in the application directory, and run the command `make ewarm TOOLCHAIN=IAR`.

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

Note: *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 from the ModusToolbox™ environment will continue to use the toolchain that was previously specified in the Makefile. You can edit the Makefile's TOOLCHAIN variable if you also want ModusToolbox™ builds to use the IAR toolchain.*

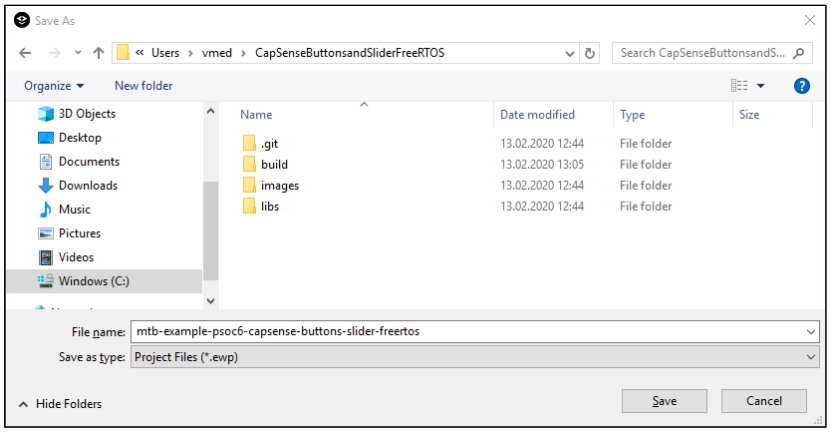
Note: *Check the output log for instructions and information about various flags.*

2.3 Open application in IAR Embedded Workbench

An IAR connection file appears in the application directory. For example:

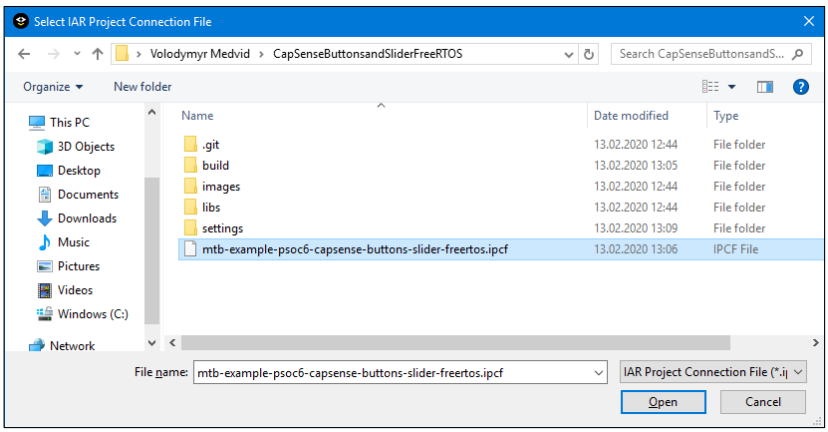
mtb-example-psoc6-capsense-buttons-slider-freertos.ipcf

1. Start IAR Embedded Workbench.
2. On the main menu, select **Project > Create New Project > Empty project** and click **OK**.
3. Browse to the ModusToolbox™ application directory, enter a desired application name, and click **Save**.



4. After the application is created, select **File > Save Workspace**. Then, enter a desired workspace name.
5. Select **Project > Add Project Connection** and click **OK**.

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



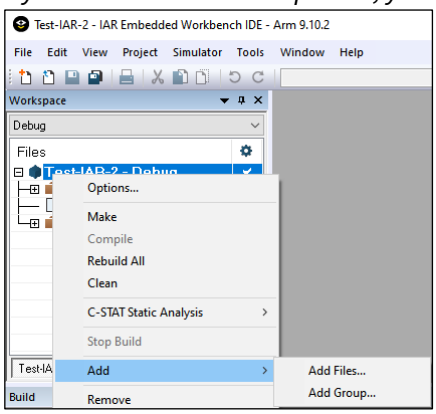
2.4 Build the application

For applications using the PSoC™ 64 secure MCU, skip this step. Instead, perform the steps outlined in the following section.

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

Note: *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. The official IAR site discusses this option: https://github.com/IARSystems/project-migration-tools*

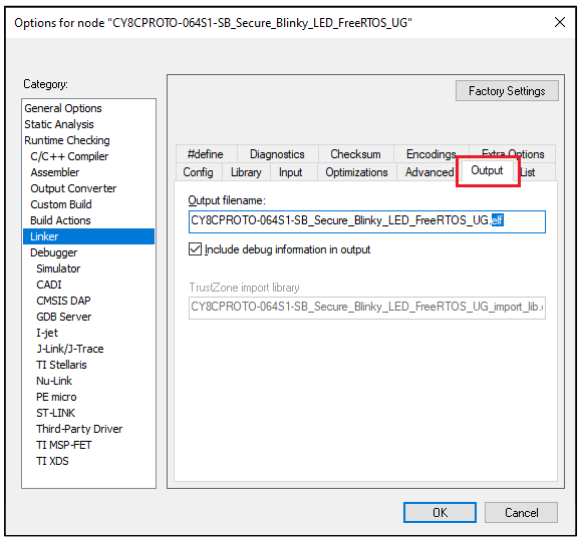
If you don't remove the .ipcf file, you need to make all file/group additions at the workspace level.



2.5 PSoC™ 64 application configuration

Before building an application for a PSoC™ 64 secure MCU in IAR, you must perform the following configuration steps.

1. Select **Project > Options > Linker > Output** and change file extension from ".out" to ".elf" in **Output filename** field:



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

3. Build the application using the ModusToolbox™ `make build` command.

   You can do this by using a Terminal or by exporting the application to Eclipse or VS Code.

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

```
C:/Infineon/Tools/ModusToolbox/tools_3.1/python/python.exe C:/UG/CY8CPROTO-064S1-SB_Secure_Blinky_LED_FreeRTOS_UG/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:/Infineon/Tools/ModusToolbox/tools_3.1/gcc --toolchain GCC_ARM --build-dir C:/UG/CY8CPROTO-064S1-SB_Secure_Blinky_LED_FreeRTOS_UG/build/APP_CY8CPROTO-064S1-SB/Debug --app-name mtb-example-psoc6-secure-blinkyled-freertos --cm0-app-path ../mtb_shared/cat1cm0p/release-v1.0.0/COMPONENT_CAT1A/COMPONENT_CM0P_SECURE --cm0-app-name psoc6_01_cm0p_secure
```

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

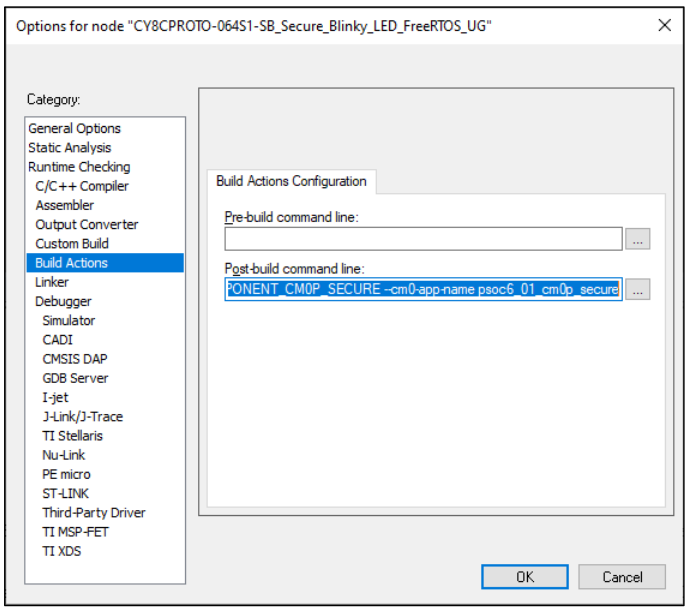
   - Add path to the policy file e.g.: `--policy-path ../policy`
   - Change `--build-dir` parameter to `Exe`
   - Change `--app-name` to you project in IAR name

   Example of command after edit:

```
C:/Infineon/Tools/ModusToolbox/tools_3.1/python/python.exe C:/UG/CY8CPROTO-064S1-SB_Secure_Blinky_LED_FreeRTOS_UG/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:/Infineon/Tools/ModusToolbox/tools_3.1/gcc --toolchain GCC_ARM --build-dir Exe --app-name CY8CPROTO-064S1-SB_Secure_Blinky_LED_FreeRTOS_UG --cm0-app-path ../mtb_shared/cat1cm0p/release-v1.0.0/COMPONENT_CAT1A/COMPONENT_CM0P_SECURE --cm0-app-name psoc6_01_cm0p_secure
```

6. Copy the edited command.

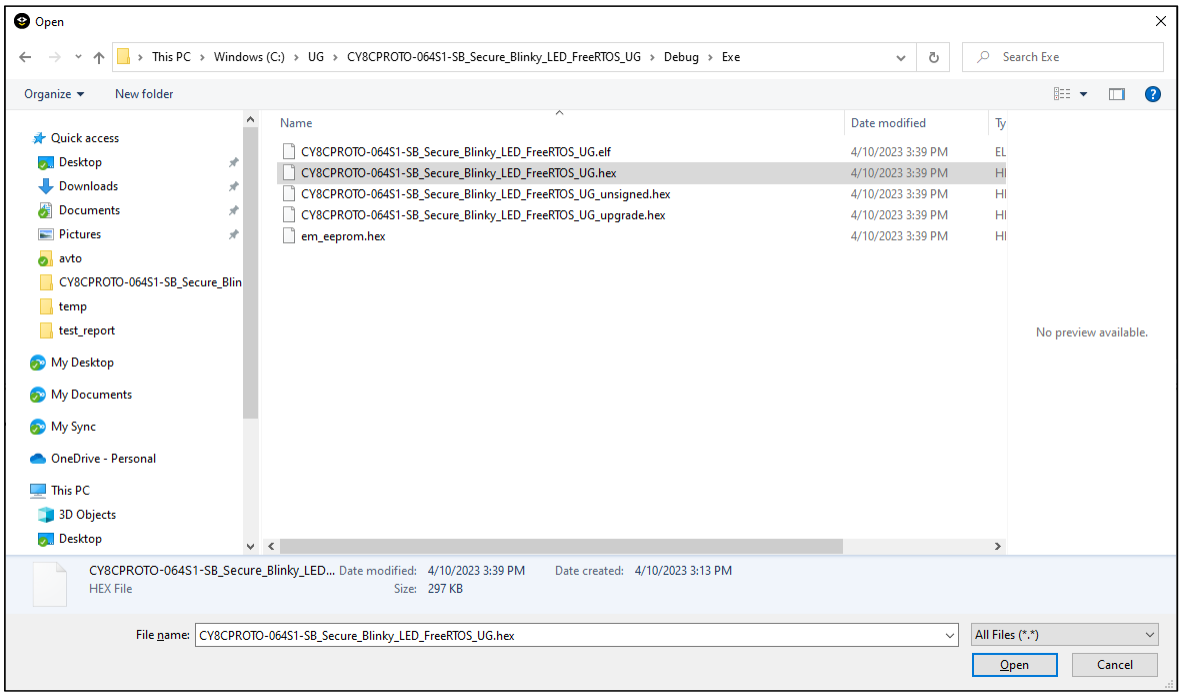
7. In IAR, select **Project > Options > Build Actions** and paste the edited command in the **Post-build command line** field:



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

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

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



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

3 Programming/Debugging

Connect the development kit to the host PC.

3.1 XMC7000 and TRAVEO™ II specific steps

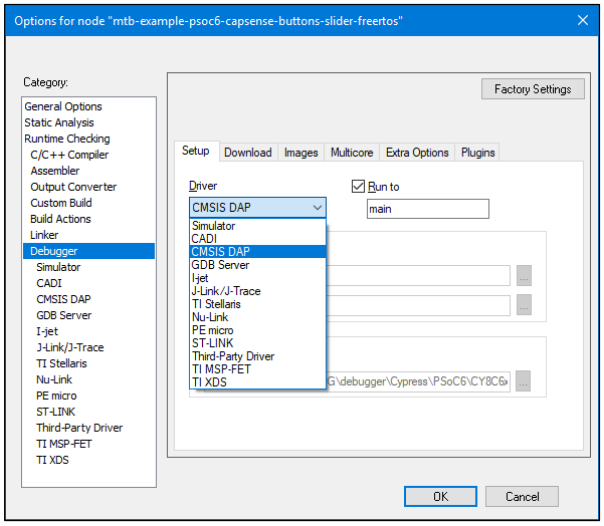
Because the XMC7000 and TRAVEO™ II devices have multiple cores – even if they are not used in a single-core application – you must perform special steps to modify the linker script for the IAR project. See XMC7000/TRAVEO™ II specific steps for more details.

3.2 To use KitProg3/MiniProg4

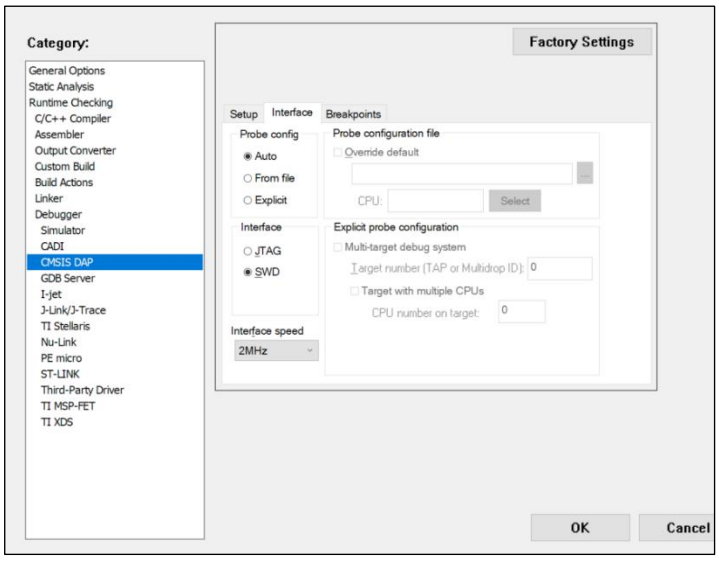
1. As needed, run the fw-loader tool to make sure the board firmware is upgraded to KitProg3. See the KitProg3 User Guide for details. The tool is in the following directory by default:

   <user_home>/ModusToolbox/tools_3.1/fw-loader/bin/

2. Select **Project > Options > Debugger** and select **CMSIS-DAP** in the Driver list:



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

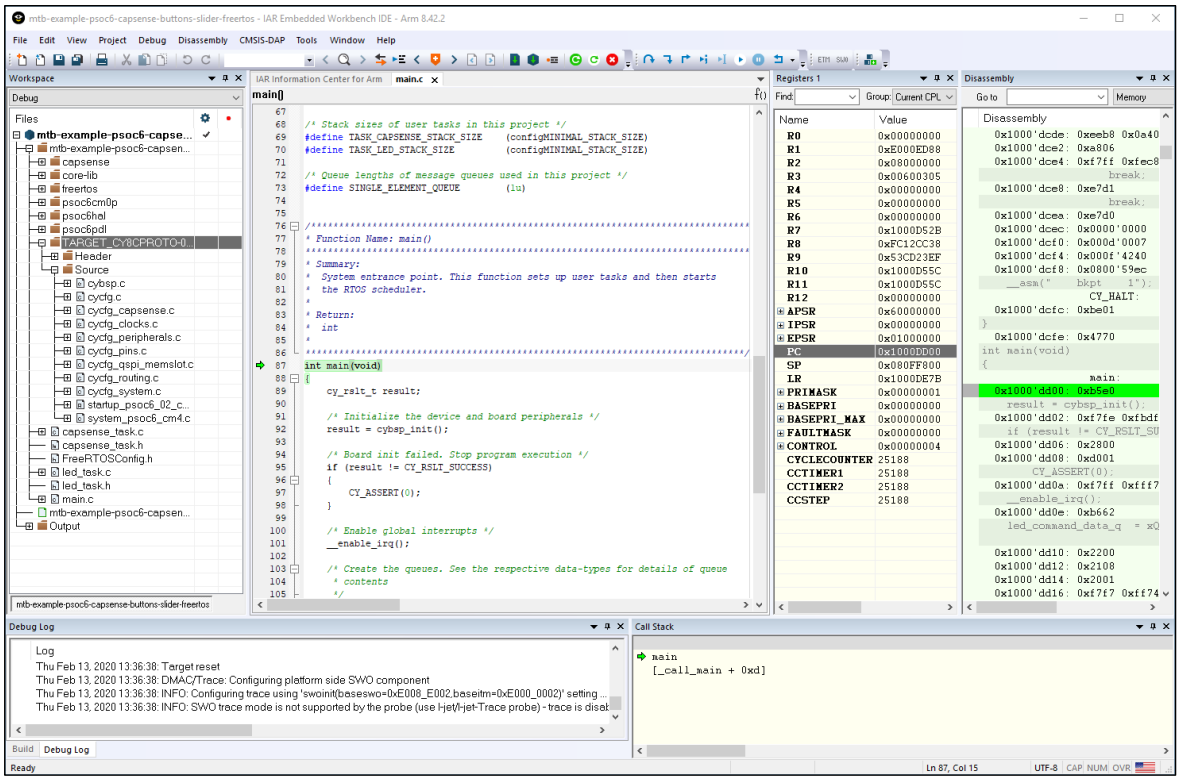


4. Click **OK**.

Programming/Debugging

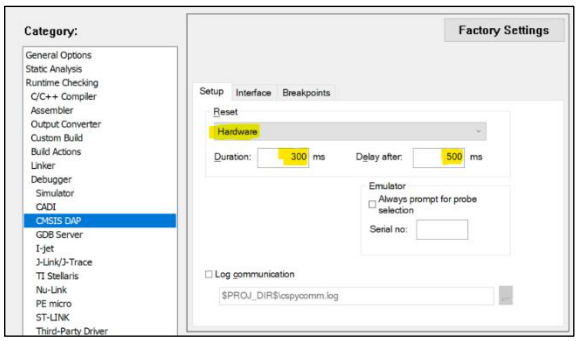
5. Select **Project > Download and Debug**.

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



3.3 To use MiniProg4 with PSoC™ 6 single core and PSoC™ 6 256K

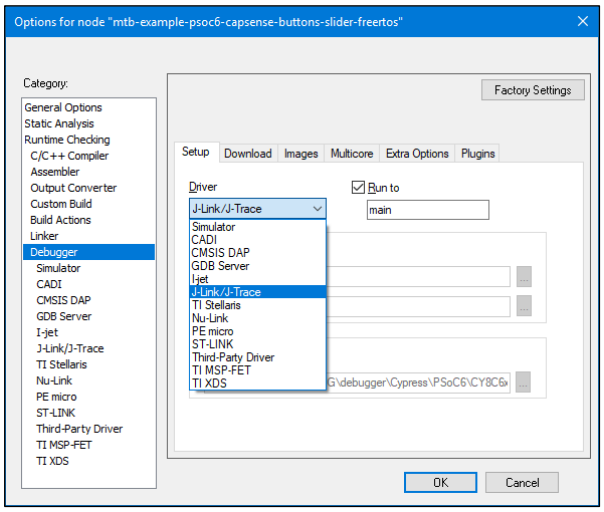
For a single-core PSoC™ 6 MCU, you must specify a special type of reset, as follows:



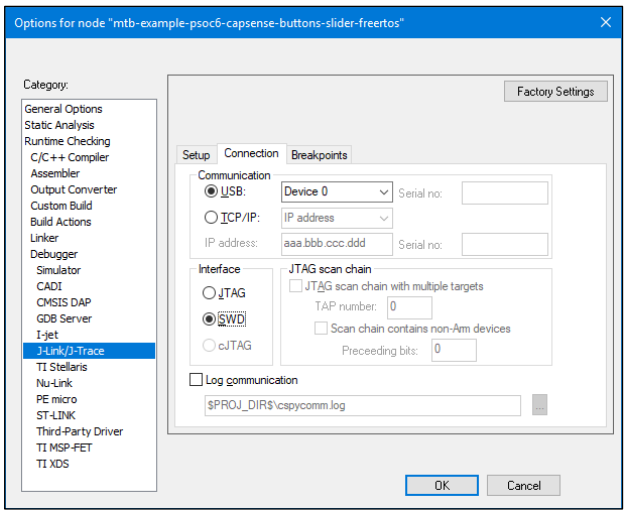
3.4 To use J-Link

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

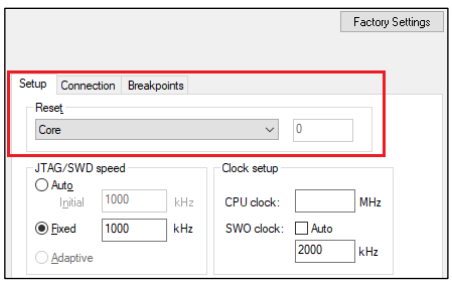
1. Open the Options dialog and select the **Debugger** item under **Category**.
2. Then select **J-Link/J-Trace** as the active driver:



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



Note: For PSoC™ 64 "Secure Boot" MCU, you must specify a special type of reset, as follows:

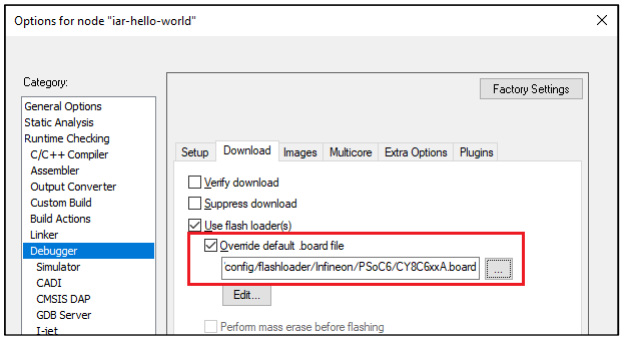


4. Connect a J-Link debug probe to the 10-pin adapter (this needs to be soldered on the prototyping kits), and start debugging.

3.5 Program external memory

IAR EWARM has disabled external memory programming by default. The SMIF region in the *.board file must be enabled manually for PSoC™ 6, AIROC™, and XMC7000 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.
3. Identify the default .board file currently used for this project.



4. Copy the default .board file from the IAR Installation directory and paste it next to the IAR project file.
5. Use a text editor to remove comment tags for the SMIF region around the <pass> element, and then comment out the ignore element for the XIP region in the recently copied file.

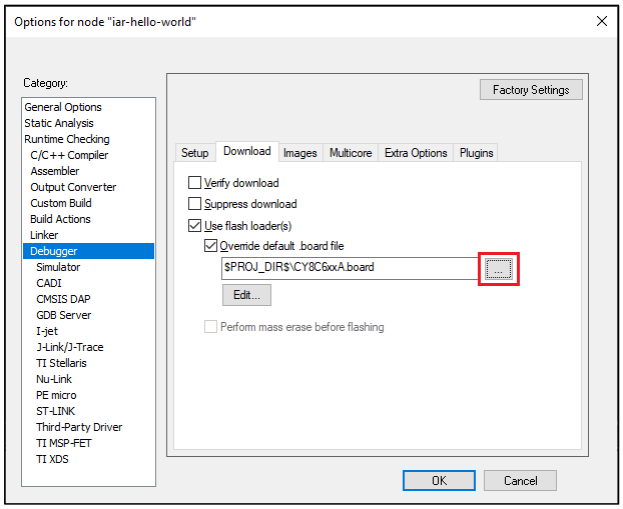
   Original file:

```
  <pass> <!-- SFLASH: Public TOC2 -->
    <loader>$TOOLKIT_DIR$/config/flashloader/Infineon/PSoC6/CY8C6xxA_SFLASH.flash</loader>
    <range>CODE 0x16007C00 0x16007FFF</range>
  </pass>
  <!-- <pass> -->
  <!-- SMIF (XIP Region) -->
    <!-- <loader>$TOOLKIT_DIR$/config/flashloader/Infineon/PSoC6/CY8C6xxA_SMIF.flash</loader> -->
    <!-- <range>CODE 0x18000000 0x1FFFFFFF</range> -->
  <!-- </pass> -->
  <!-- Exclude regions -->
  <ignore>CODE 0x08000000 0x080FFFFF</ignore> <!-- Exclude SRAM Region -->
  <ignore>CODE 0x16000000 0x160007FF</ignore> <!-- Exclude SFLASH [SFLASH Start - User Data Start] -->
  <ignore>CODE 0x16001000 0x160019FF</ignore> <!-- Exclude SFLASH [User Data End - NAR Start] -->
  <ignore>CODE 0x16001C00 0x160059FF</ignore> <!-- Exclude SFLASH [NAR End - Public Key Start] -->
  <ignore>CODE 0x16006600 0x16007BFF</ignore> <!-- Exclude SFLASH [Public Key End - TOC2 Start] -->
  <ignore>CODE 0x90300000 0x903FFFFF</ignore> <!-- Exclude Cy Checksum Region -->
  <ignore>CODE 0x90500000 0x905FFFFF</ignore> <!-- Exclude Cy Metadata Region -->
  <ignore>CODE 0x90700000 0x907FFFFF</ignore> <!-- Exclude eFuse Region -->
  <ignore>CODE 0x18000000 0x1FFFFFFF</ignore> <!-- Exclude XIP Region -->
</flash_board>
```

   Edited file:

```
  <pass> <!-- SFLASH: Public TOC2 -->
    <loader>$TOOLKIT_DIR$/config/flashloader/Infineon/PSoC6/CY8C6xxA_SFLASH.flash</loader>
    <range>CODE 0x16007C00 0x16007FFF</range>
  </pass>
  <pass><!-- SMIF (XIP Region) -->
    <loader>$TOOLKIT_DIR$/config/flashloader/Infineon/PSoC6/CY8C6xxA_SMIF.flash</loader>
    <range>CODE 0x18000000 0x1FFFFFFF</range>
  </pass>
  <!-- Exclude regions -->
  <ignore>CODE 0x08000000 0x080FFFFF</ignore> <!-- Exclude SRAM Region -->
  <ignore>CODE 0x16000000 0x160007FF</ignore> <!-- Exclude SFLASH [SFLASH Start - User Data Start] -->
  <ignore>CODE 0x16001000 0x160019FF</ignore> <!-- Exclude SFLASH [User Data End - NAR Start] -->
  <ignore>CODE 0x16001C00 0x160059FF</ignore> <!-- Exclude SFLASH [NAR End - Public Key Start] -->
  <ignore>CODE 0x16006600 0x16007BFF</ignore> <!-- Exclude SFLASH [Public Key End - TOC2 Start] -->
  <ignore>CODE 0x90300000 0x903FFFFF</ignore> <!-- Exclude Cy Checksum Region -->
  <ignore>CODE 0x90500000 0x905FFFFF</ignore> <!-- Exclude Cy Metadata Region -->
  <ignore>CODE 0x90700000 0x907FFFFF</ignore> <!-- Exclude eFuse Region -->
  <!-- <ignore>CODE 0x18000000 0x1FFFFFFF</ignore> --> <!-- Exclude XIP Region -->
</flash_board>
```

6. Save the file.
7. In IAR, click the **Browse [...]** button, then navigate to and select the edited .board file.

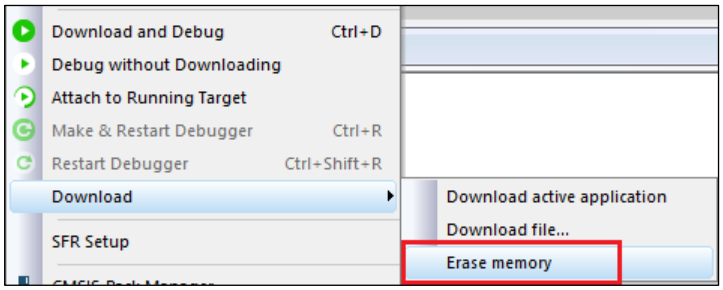


8. Click **OK** when you are finished.

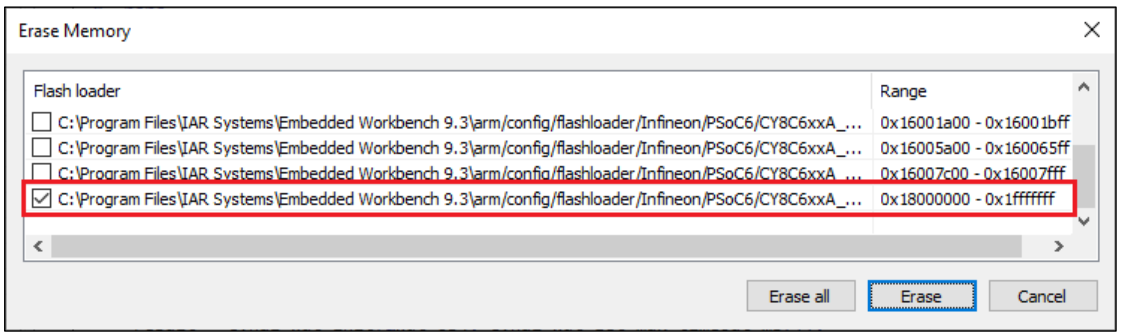
3.6 Erase PSoC™ 6 MCU with external memory enabled

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**.



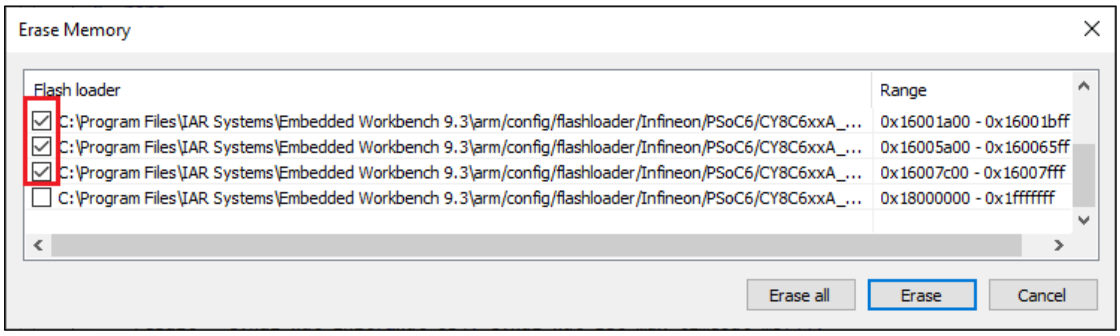
2. Deselect the check boxes for all regions, **except** 0x18000000-0x1fffffff.



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

Programming/Debugging

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



6. Click **Erase**.

4 Multi-core debugging

This section describes how to set up multi-core debugging in IAR Embedded Workbench for Arm IDE (IAR). For this purpose, we need to create an IAR workspace containing a few projects (one project per MCU core).

4.1 Supported debugger probes

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

4.2 Create IAR workspace and projects

After creating a ModusToolbox™ multi-core application for use with IAR, do the following:

1. Launch IAR.
2. On the main menu, select **Project** > **Create New Project** > **Empty project** and click **OK**.
3. Browse to the ModusToolbox™ project directory for one of the cores, enter a desired project name, and click **Save**.
4. After the project is created, select **File** > **Save Workspace**. Then, enter a desired workspace name.
5. Select **Project** > **Add Project Connection** and click **OK**.
6. On the **Select IAR Project Connection File** dialog, select the .ipcf file located in the project directory for CM0+ core and click **Open**.
7. Repeat steps 2-3 and 5-6 for all other core projects in the application.

Once you have a working workspace, you need to properly configure IAR projects in order to be able to establish a multi-core debug session. Also, for some MCUs you must edit linker scripts in order to organize flash allocation properly.

4.3 Customizing linker scripts

There are separate linker scripts for each core. Linker scripts generally require updating to allocate a proper amount of flash for each core. The following process uses a PSoC™ 6 MCU as an example.

4.3.1 PSoC™6 MCU example

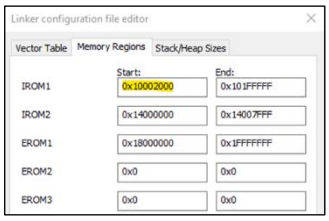
By default, the memory layout for a PSoC™ 6 MCU is distributed as follows:

- 2 KB is allocated for CM0+ core.
- The whole flash is allocated for the CM4 core (1 MB or 2 MB depending on the target MCU).

To change memory allocation for the CM4 core, perform the following:

1. Select the CM4 core project and go to **Project** > **Options** > **Linker** > **Edit**.

2. Select the **Memory Regions** tab and change **IROM1 Start:** from 0x10000000 to 0x10002000 as shown:



3. Click **Save**, and then **OK** to save the project.

4. Adjust memory allocation (flash or ram) according to your needs.

Note: *Be aware that when updating the CM4 start address, linker script modification is not enough. You must update the CM0+ project, since the CM4 core is started by the CM0+ core. Update the* `CY_CORTEX_M4_APPL_ADDR` *variable defined in the system_psoc6.h file, or directly substitute the proper value in the main.c file of CM0+ application:*

```
/* Enable CM4. CY_CORTEX_M4_APPL_ADDR must be updated if CM4 memory layout is changed. */
Cy_SysEnableCM4(CY_CORTEX_M4_APPL_ADDR);
```

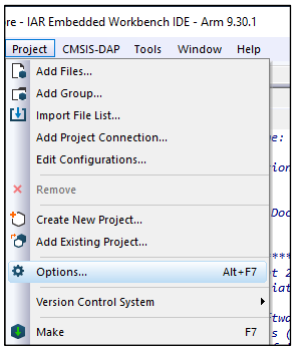
In the same manner you can edit linker scripts for XMC7000 if needed. Just make sure memory layouts of all cores are aligned with each other. Then, build all projects before moving forward.

4.4 Configuring IAR projects

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 CM0+ core project as the master project, and configure the other cores (CM4 for PSoC™ 6 and CM7 for XMC7000) as slave projects.

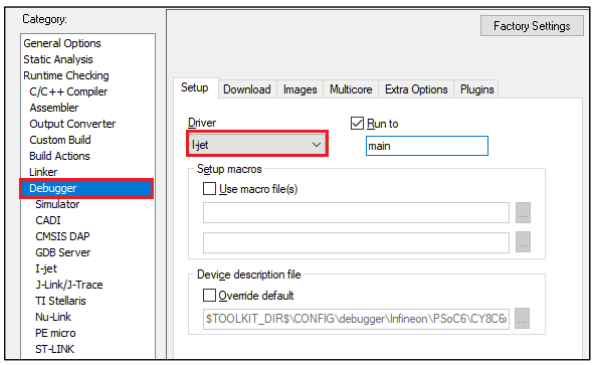
4.4.1 Project configuration for CM4/CM7 (slave) core(s)

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



Multi-core debugging

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.

   a. In the **Runtime Checking > Output Converter** category, select the **Generate additional output** check box.

   b. Ensure **Output format** is set to **Intel Extended hex**.

   c. Click **OK**.

4. Repeat these steps for your all projects for CM4/CM7 (for triple-core MCUs)

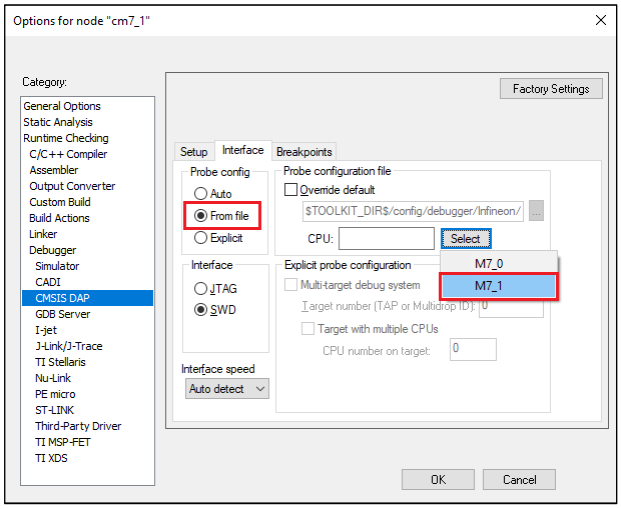
4.4.2 XMC7000/TRAVEO™ II specific steps

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**.

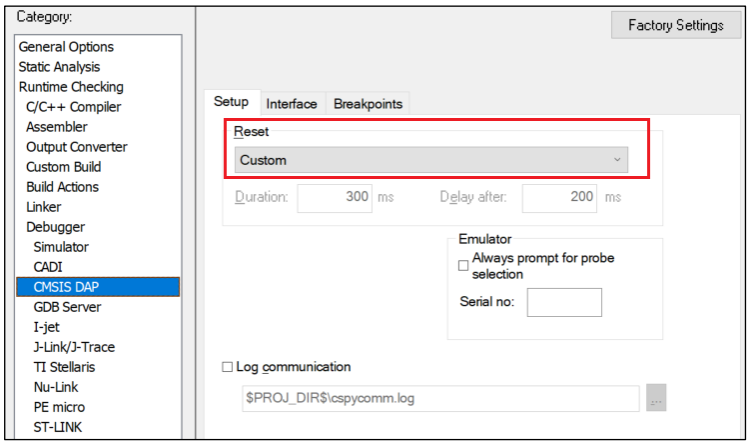
2. Select the probe in the **Debugger** category, ans switch to the **Interface** tab.

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



Multi-core debugging

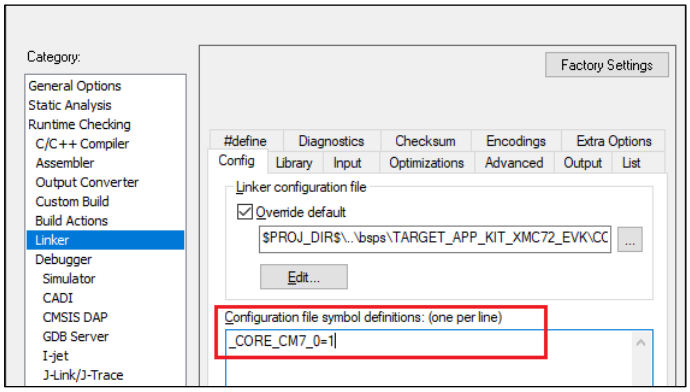
4. Switch to the **Setup** tab, and select "Custom" from the **Reset** pull-down menu.



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:

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

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



Do the same for the second CM7 core:

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

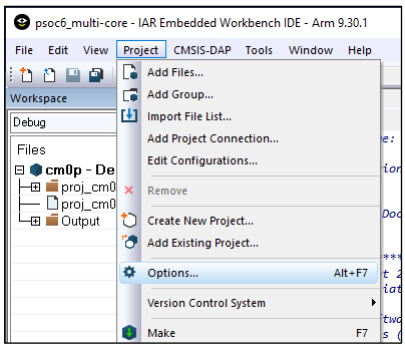
8. Add `_CORE_CM7_1_=1` in the **Configuration file symbol definitions** field, and click **OK**.

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.*

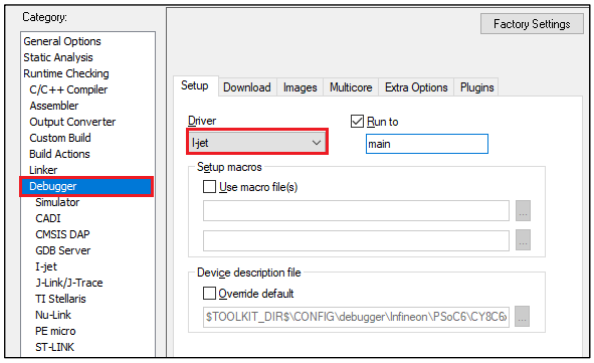
Build your CM4/CM7 project(s) before moving forward.

4.4.3 Project configuration for 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):

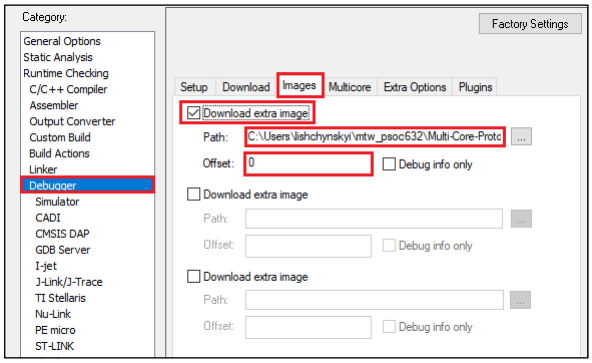


3. Switch to the **Images** tab to specify the extra image to be downloaded prior to debugging in order to download images of all projects in one process.

   a. Select the **Download extra image** check box.

   b. Provide a **Path** to the CM4/CM7's **HEX** image.

   c. Enter 0 for **Offset**.



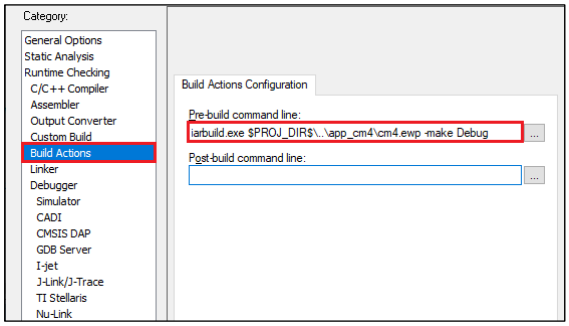
   If you provide an **OUT** file instead of a **HEX** file, the IAR IDE will fail to halt at the beginning of `main()` due to the main function present in both the CM0+ and CM4/CM7 **OUT** files.

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

Multi-core debugging

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
   ```

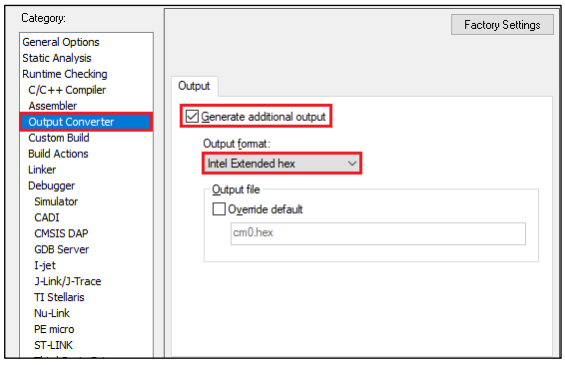
   

   If your MCU has three cores, you might want to also specify a post-build action to build project for the third core in the same manner.

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

   a. Select the **Generate additional output** check box.

   b. 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.

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.

4.4.4 CMSIS-DAP/I-Jet-specific configuration

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>
   ```

Multi-core debugging

```
            <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>
```

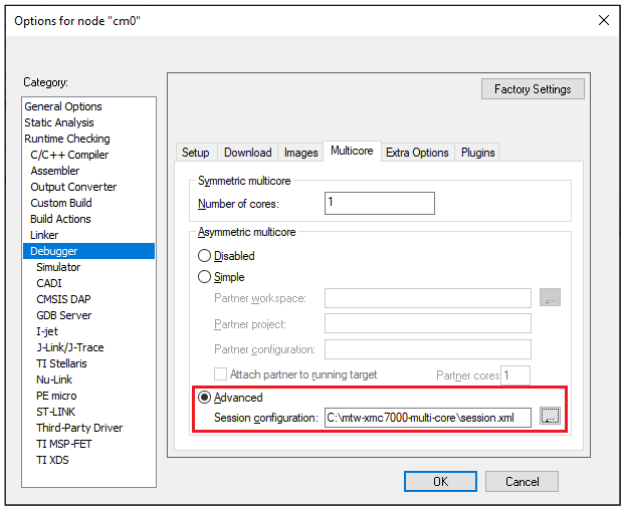
2. Configure multi-core debugging for the CM0+ project.

   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**.

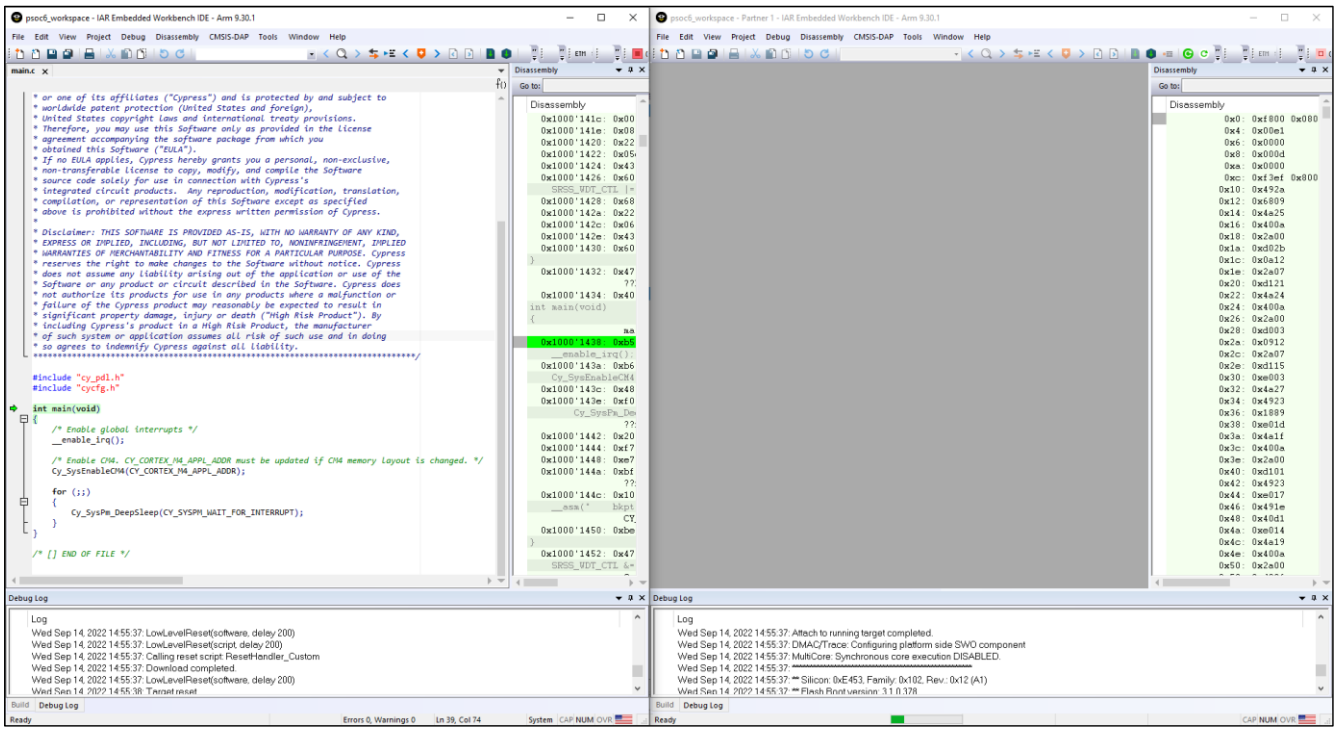


3. Save the workspace.

4.5 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 similar to this:



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

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

4.6 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 will not automatically open separate IDE instances for each core, thus you need to do it manually.
- Some enhanced features like are not available; see Multi-core toolbar and CTI usage.

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**.

4.7 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:

5 Patched flashloaders for AIROC™ CYW208xx devices

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

1. Copy the *CYW208xx_SMIF.out* file located in the *<app-dir>\libs\<Kit-Name>\COMPONENT_BSP_DESIGN_MODUS\GeneratedSource* directory.
2. Paste the flashloader file to the *C:\Program Files\IAR Systems\Embedded Workbench 9.0\arm\config\flashloader\Infineon\CYW208XX* directory.
3. Also, to use the SEGGER J-Link debugger, paste the *CYW208xx_SMIF.out* file to the *C:\Program Files\SEGGER\J-Link\Devices\Cypress\cat1b* directory.

Revision history

Revision	Date	Description
**	2023-05-15	New document.

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

Edition 2023-05-15

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

002-37598 Rev. **

Important notice

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie")

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, 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.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

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.