Keil µVision 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A newer version of this document may be available on the web here.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 Keil µVision with ModusToolbox™ software.

The general flow for working with an Infineon device in Keil µVision Workbench includes:

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

Document conventions

Convention	Explanation
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 tools package on Windows, Linux, and macOS.
- ModusToolbox™ tools package user guide – Provides information about all the tools included with ModusToolbox™ tools package.
- Dashboard user guide – Provides specific information about the Dashboard tool.
- Project Creator user guide – Provides specific information about the Project Creator tool.
- Device Configurator guide – Provides specific information about the Device Configurator.

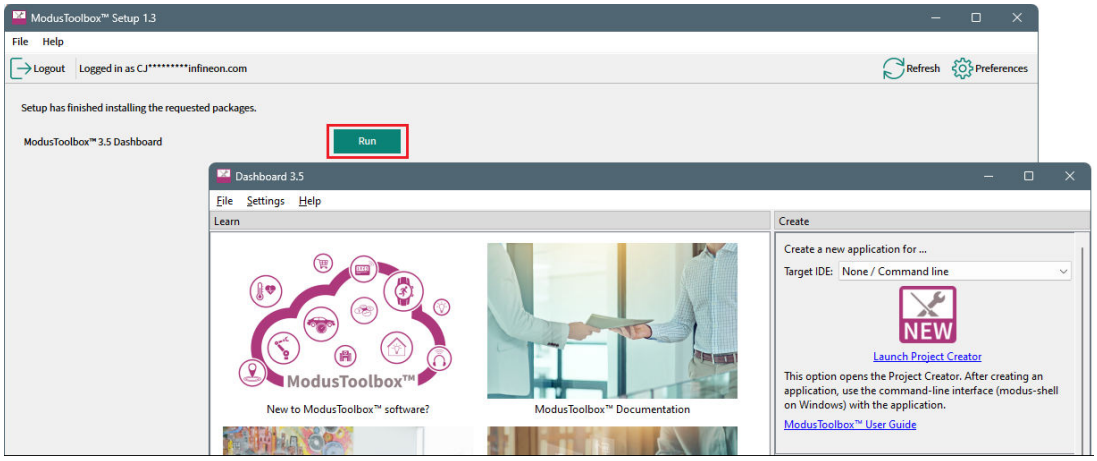
Table of contents

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 Keil µVision (Windows only)

We recommend and have tested Keil µVision version 5.39 for PSOC™ Control C3 devices. This version can be used with all ModusToolbox™ supported devices.

The default installation location for the ARM compiler is *C:\Keil_v5\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/J-Link/J-Link_Windows.exe

2 Create/export application for Keil µVision

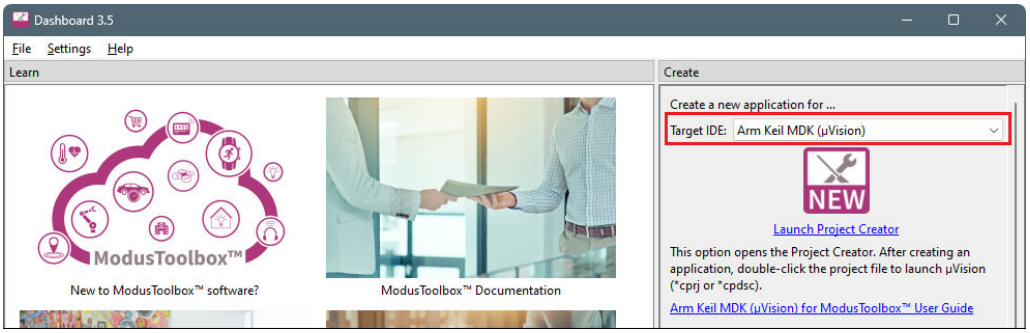
This section covers the ways to get started using Keil µVision with ModusToolbox™ software.

- Create/export ModusToolbox™ application
- Create Keil µVision project(s)

2.1 Create/export ModusToolbox™ application

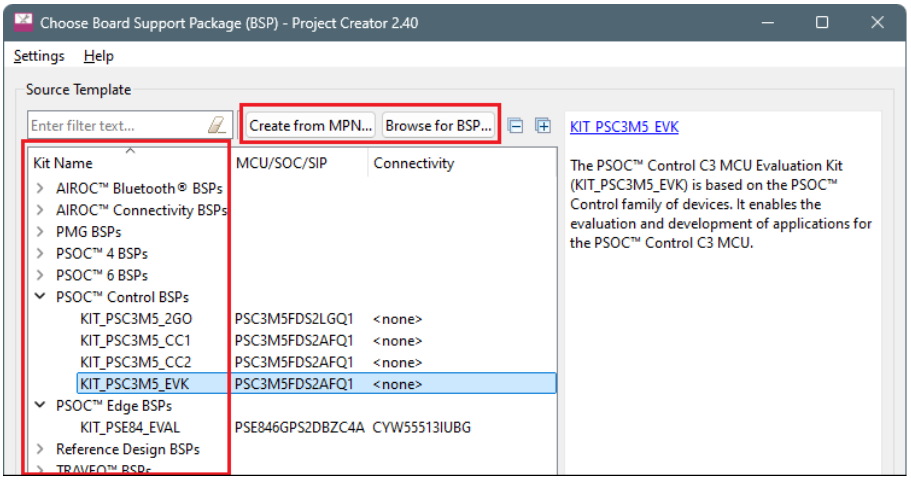
Create new application

1. Use the Dashboard to open the Project Creator tool and create a ModusToolbox™ application for Keil µVision.

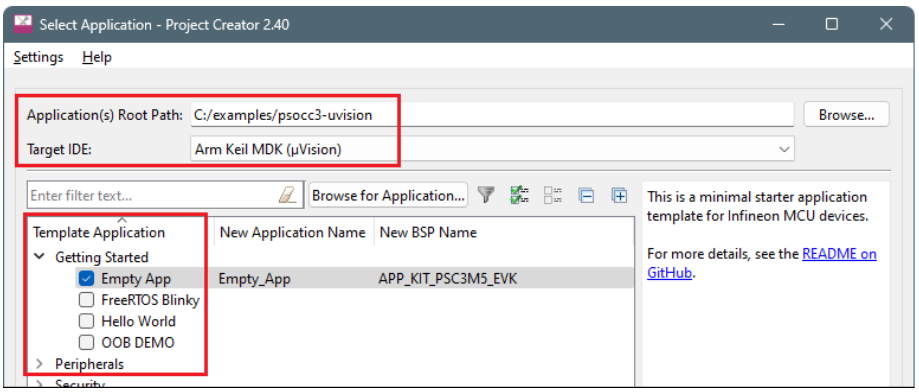


Refer to the Project Creator user guide for more details.

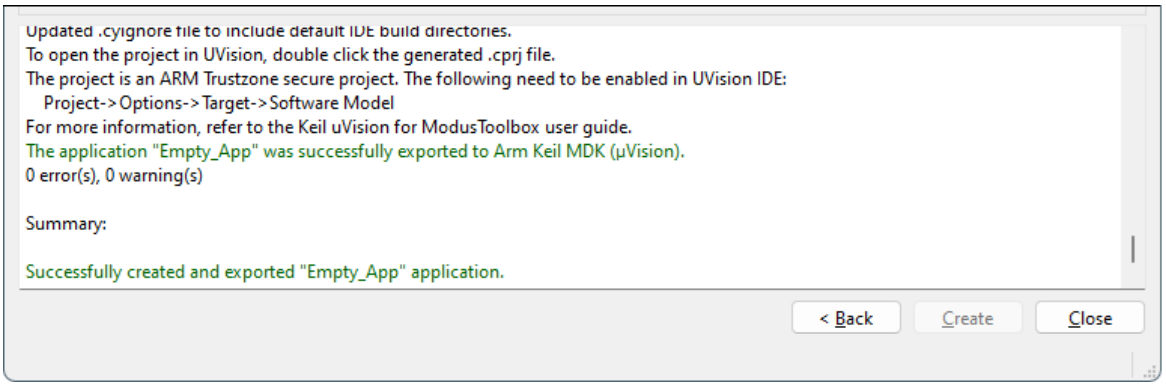
2. Select the BSP from the list or use a buttons to create one or select one on disk.



3. On the next page, select the location, target IDE, and the application to create.



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 Keil µVision. Open a terminal window (modus-shell in Windows) and type the following:

```
make uvision CY_IDE_PRJNAME=[project-name] TOOLCHAIN=ARM
```

***Note**: For applications that were created using core-make-3.0 or older, you must use the `make uvision5` command instead.*

This sets the `TOOLCHAIN` to ARM in the Keil µVision configuration files but **not** in the ModusToolbox™ application's Makefile. Therefore, builds inside Keil µVision will use the ARM toolchain, while builds in 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 ARM toolchain.

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

2.2 Create Keil µVision project(s)

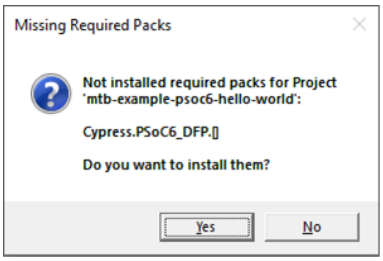
Creating or exporting the application generates the following file in the application/project directory:

**.cprj*

The *cprj* file extension should have the association enabled to open it in Keil µVision.

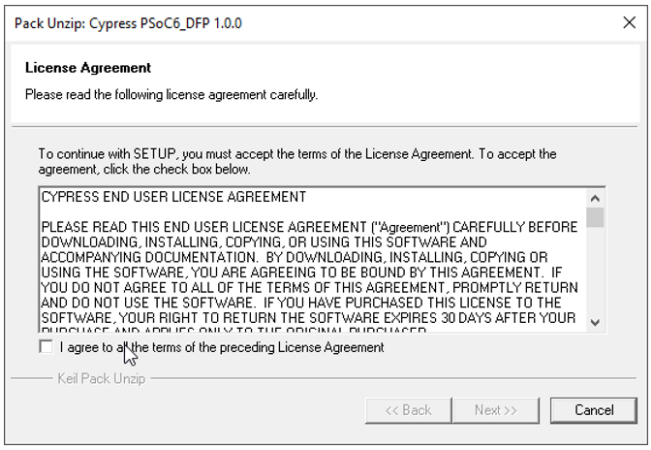
1. Double-click the **.cprj* file. This launches the Keil µVision IDE.

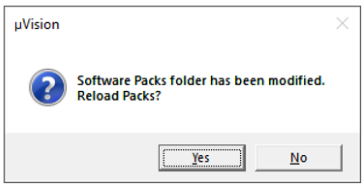
   The first time you do this, a dialog similar to the following displays:



2. Click **Yes** to install the device pack. You only need to do this once per device type.

3. Follow the steps in the Pack Installer to properly install the device pack.





***Note**: In some cases, you may see the following error message:*
SSL caching disabled in Windows Internet settings. Switched to offline mode.
See this link for how to solve this problem: https://developer.arm.com/documentation/ka002253/latest

4. When complete, close the Pack Installer and close the Keil µVision IDE.

5. Then double-click the *.cprj* file again and the application will be created for you in the IDE.

6. If you're working with a multi-core or multi-project application, a cprj file is created in each sub-project directory. Double-click the *.cprj file in each folder (for example, cm0p, cm7_0, and cm7_1).

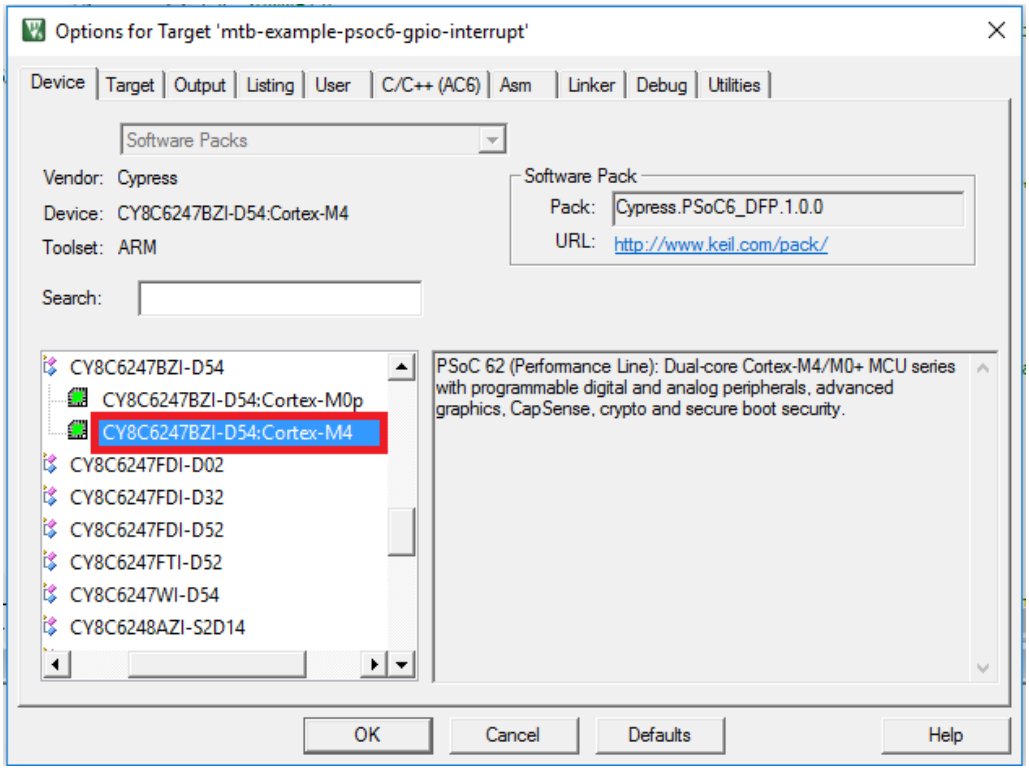
3 Miscellaneous notes

3.1 Supported debugger probes

- KitProg3 on-board programmer
- MiniProg4
- ULINK2 in CMSIS-DAP mode
- J-Link

3.2 To use KitProg3/MiniProg4, CMSIS-DAP, and ULink2 debuggers

1. Select the **Device** tab in the Options for Target dialog and check that M4 core is selected:

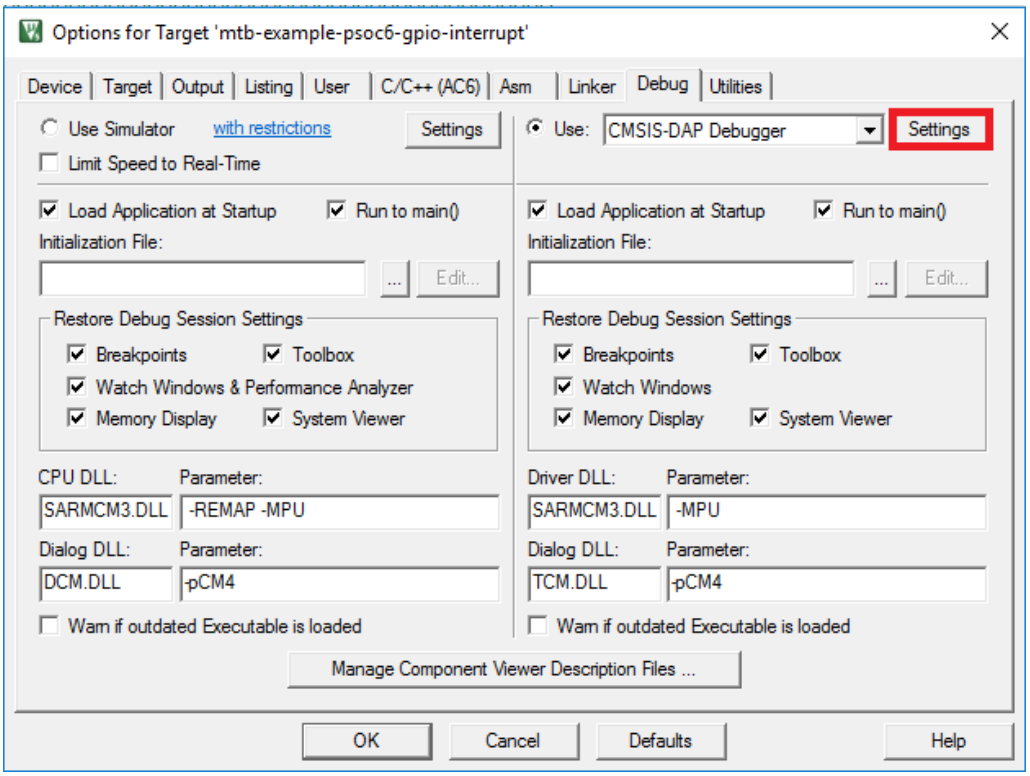


2. Select the **Debug** tab.

***Note**: To use the ULink2 probe for multi-core debugging, select the CMSIS-DAP Debugger instead of ULink for each core of the project (CM4 and CM0P).*

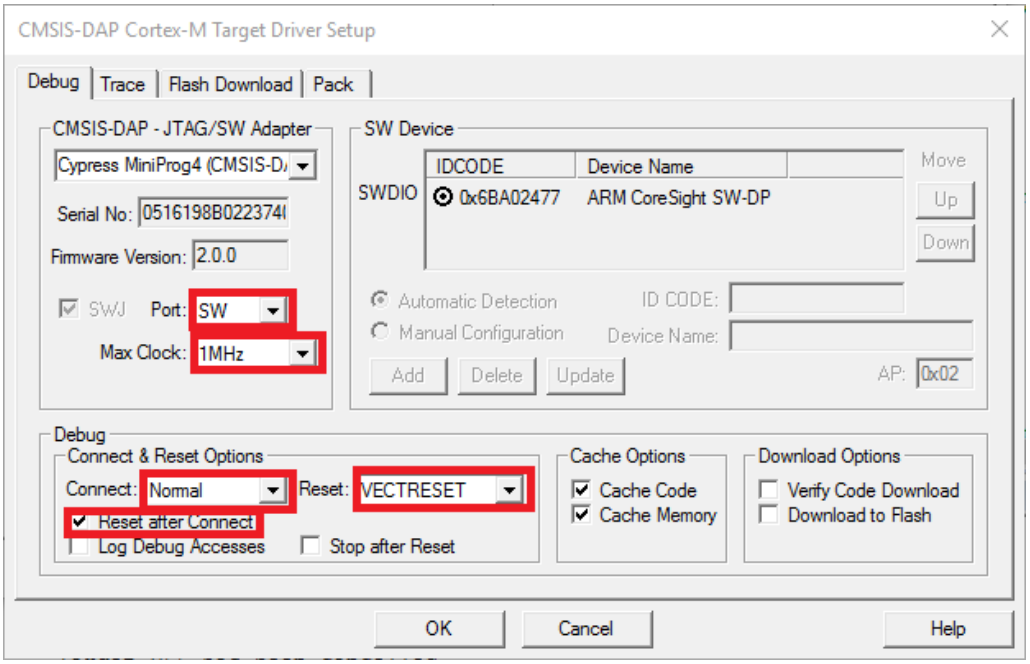
3. Click the **Settings** button.

3 Miscellaneous notes



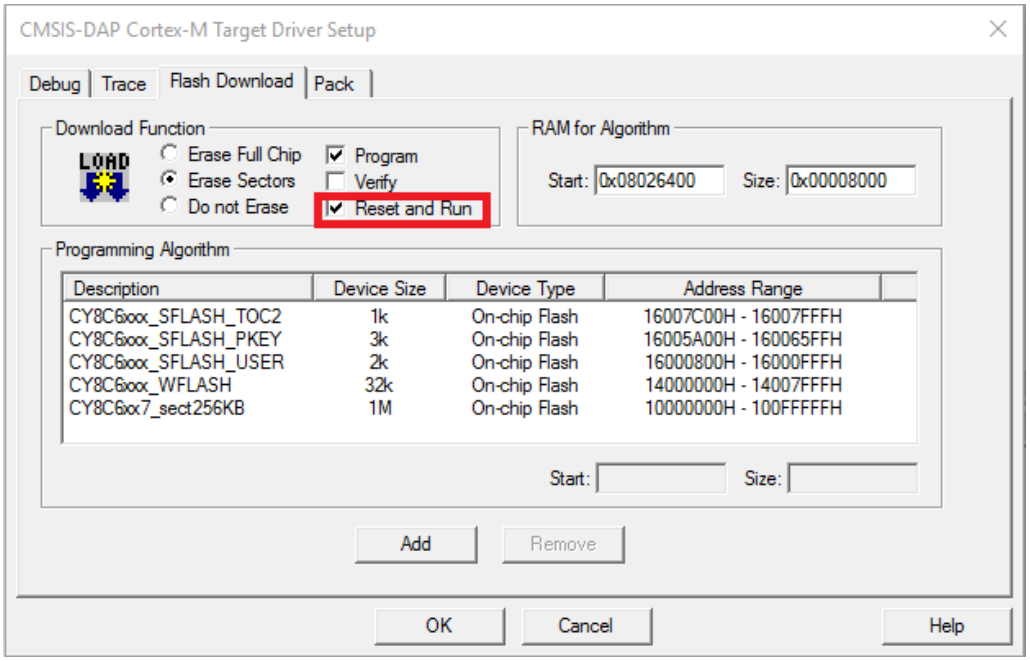
4. On the Target Driver Setup dialog on the **Debug** tab, select the following:

- set **Port** to "SW"
- set **Max Clock** to "1 MHz"
- set **Connect** to "Normal"
- set **Reset**:
  - For PSOC™ 6, to "VECTRESET"
  - For PSOC™ 4, PMG1, and AIROC™ CYW208xx, to "SYSRESETREQ"
- enable **Reset after Connect** option

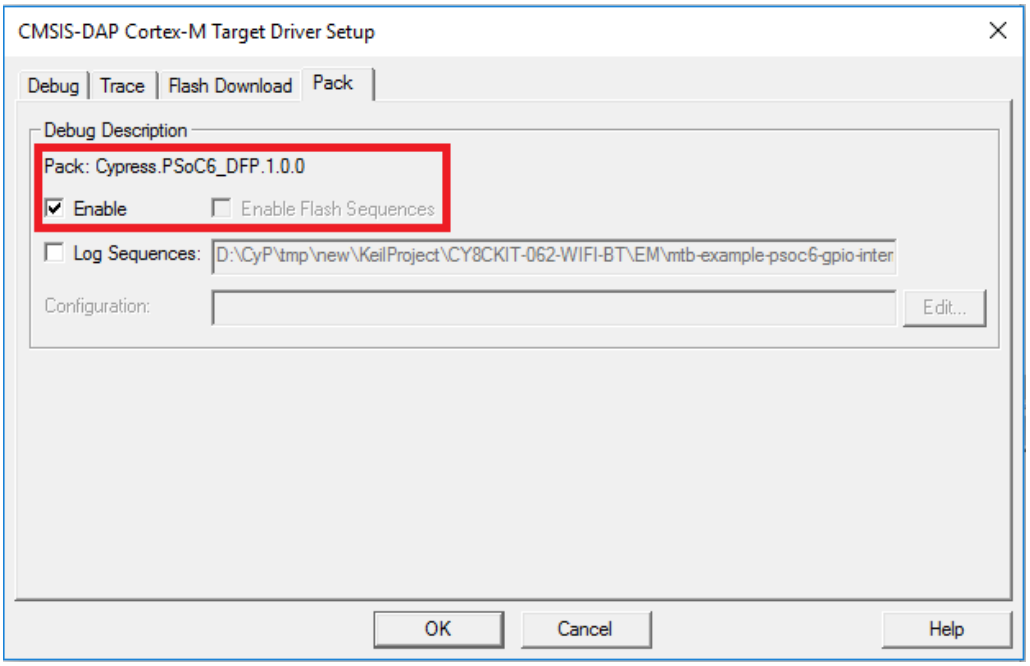


5. For PSOC™ 4, PMG1, and AIROC™ CYW208xx, to "SYSRESETREQ".

6. Select the **Flash Download** tab and select "Reset and Run" option after download, if needed:



7. Select the **Pack** tab and check if "Cypress.PSoC6_DFP" is enabled:



3.3 Patched flashloaders

Step 1: Identify the CMSIS Pack and FLM file

- Determine the name and version of the CMSIS Pack containing the FLM file you want to overwrite (for example, *AIROC_DFP.1.2.0*).
- Identify the name of the patched flashloader (the FLM file) you want to overwrite (for example, *CYW208xx_SMIF.FLM*). This file is located in the *<app-dir>\bsps\<Kit-Name>\config\GeneratedSource* directory.

Step 2: Locate the destination FLM file

- Open File Explorer and navigate to the directory: *%LocalAppData%\Arm\Packs\<Pack_Name>\<Version>\Flash*
- Replace *<Pack_Name>* with the name of the CMSIS Pack (for example, *Infineon\AIROC_DFP*) and *<Version>* with the version number of the CMSIS Pack (for example, *1.2.0*).
- In this directory, find the file named *<FLM_File_Name>.FLM* (for example, *CYW208xx_SMIF.FLM*).

Step 3: Overwrite the FLM file

- Ensure that Keil µVision and any other applications using the FLM file are closed.
- Replace the destination *<FLM_File_Name>.FLM* file with the patched one you want to use.
- Make sure to keep the same file name and extension (.FLM) to avoid any issues.

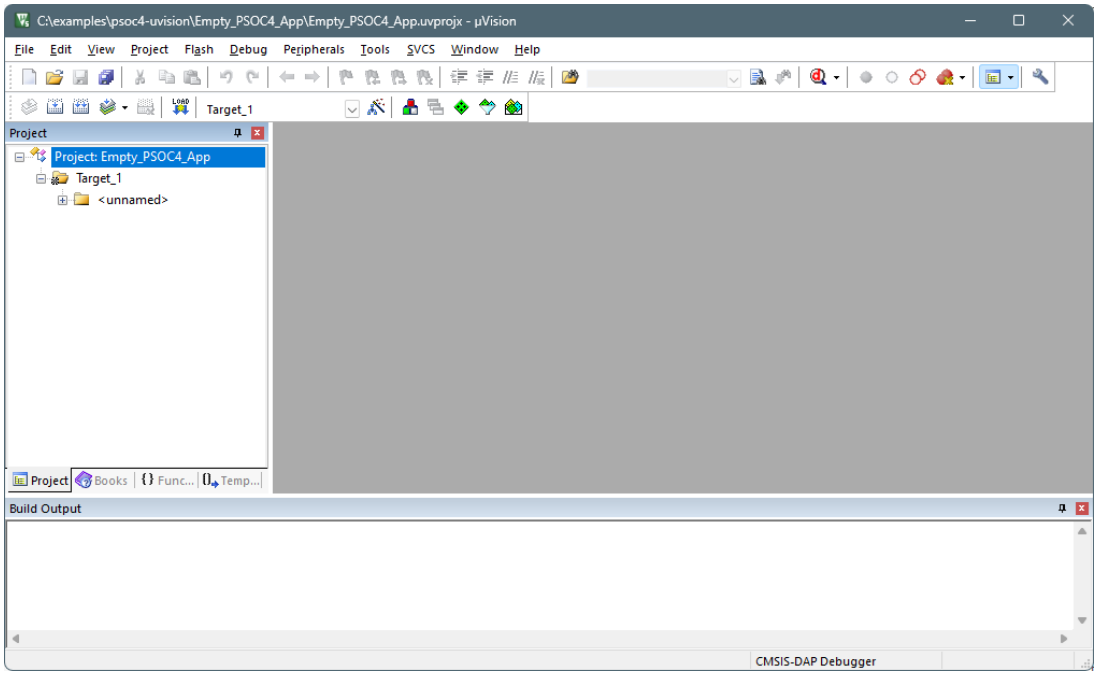
***Note**: When using J-Link with µVision, the flashloaders will be taken from the CMSIS DFP. So, there is no need to replace flash loaders in SEGGER J-Link software.*

3.4 Perform ETM/ITM trace

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

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

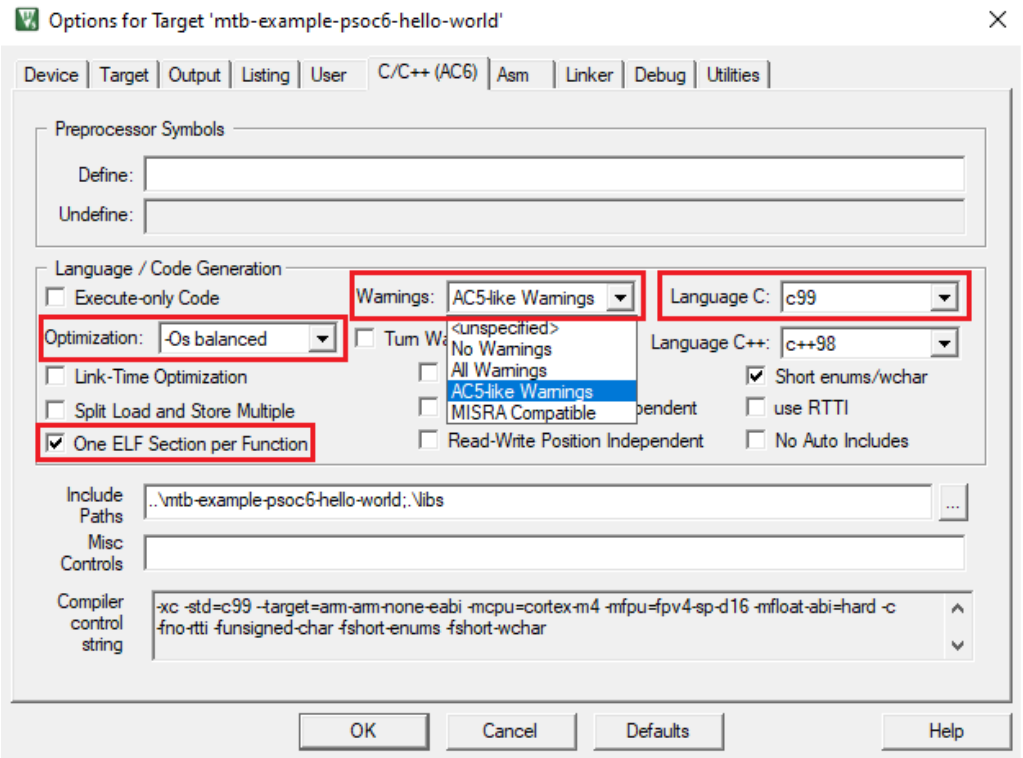
Follow instructions in Create/export application for Keil µVision. When complete, your project should look similar to this:



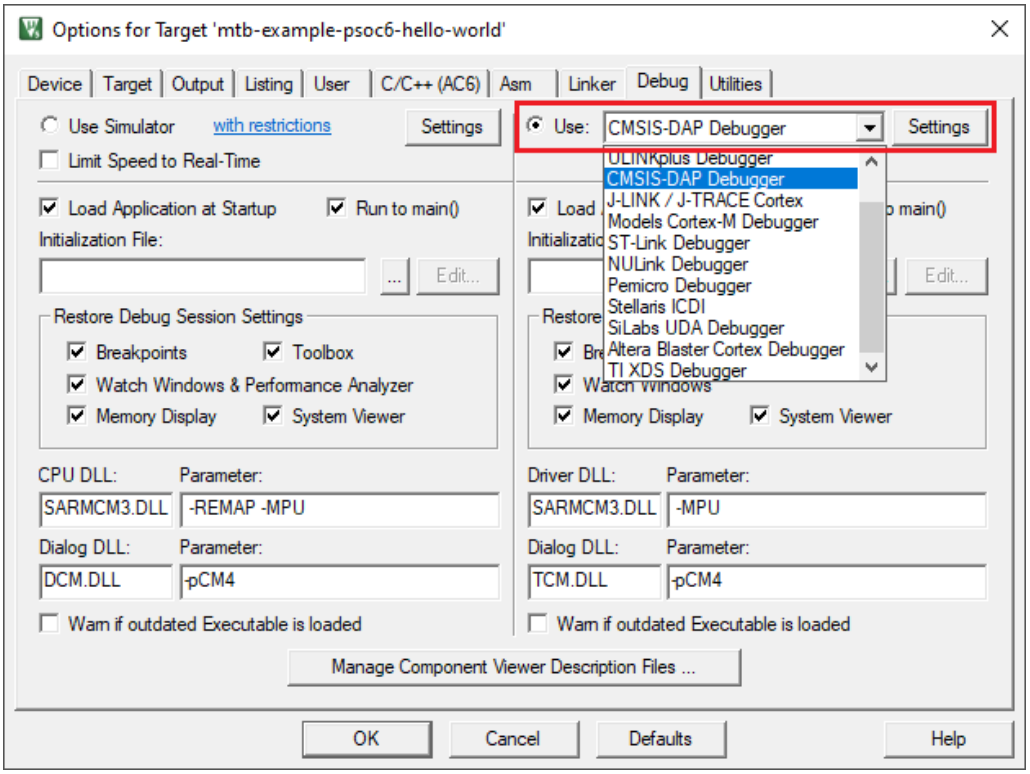
Then, use these instructions to configure, build, and debug your application.

1. Open the Options dialog and select the **C/C++ (AC6)** tab.

- Check that the Language C version was automatically set to c99.
- Select "AC5-like warnings" in the Warnings drop-down list.
- Select "-Os balanced" in the Optimization drop-down list.
- To reduce memory usage, select the One ELF Section per Function check box.

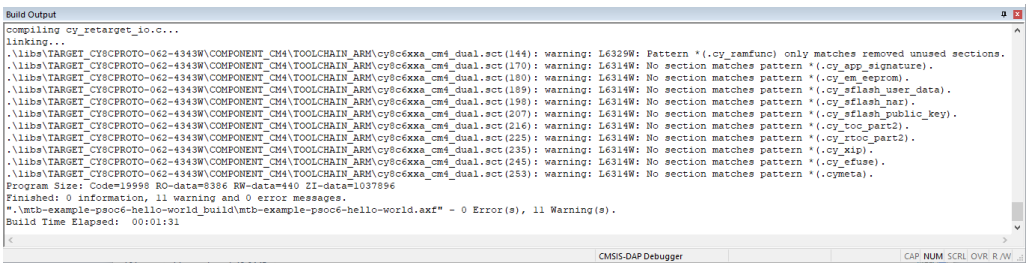


2. Switch to the **Debug** tab, and select KitProg3 CMSIS-DAP as an active debug adapter:



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

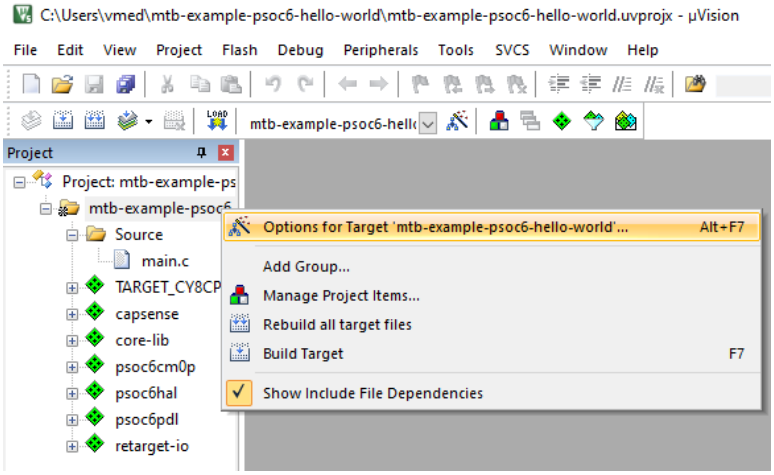
4. Select **Project > Build target**.



5. To suppress the linker warnings about unused sections defined in the linker scripts, add "6314,6329" to the **Disable Warnings** setting in the Project Linker Options.

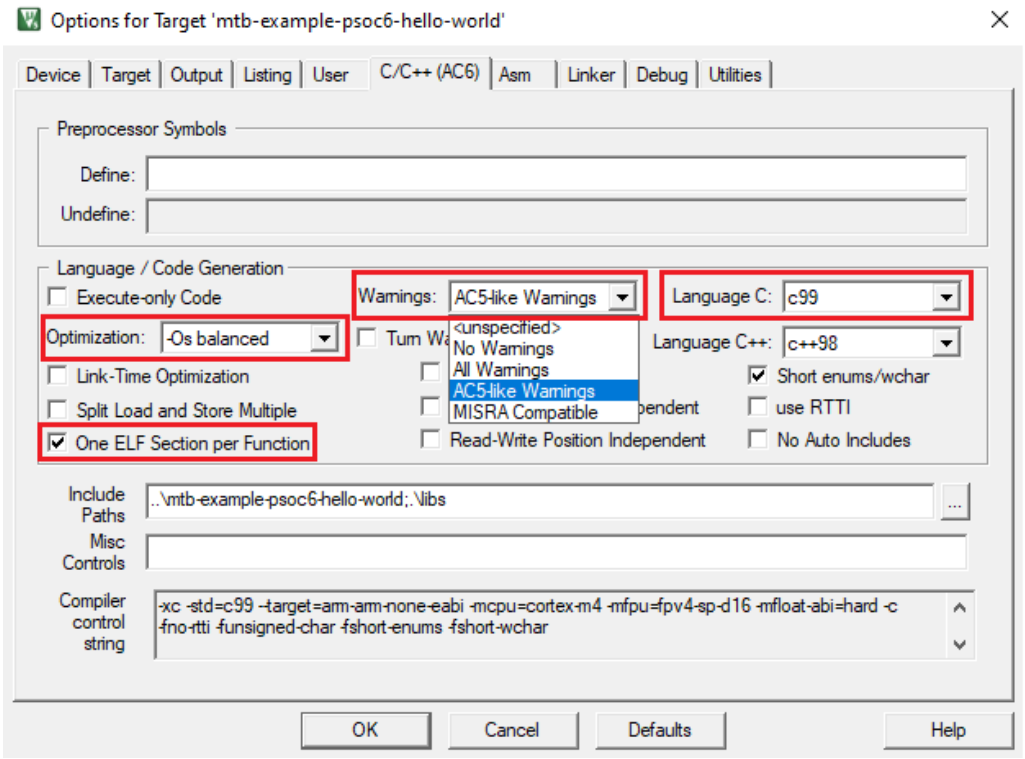
4.1 Configure and build the application

1. Right-click on the *mtb-example-psoc6-hello-world* directory in the µVision Project view, and select **Options for Target '<application-name>' ...**

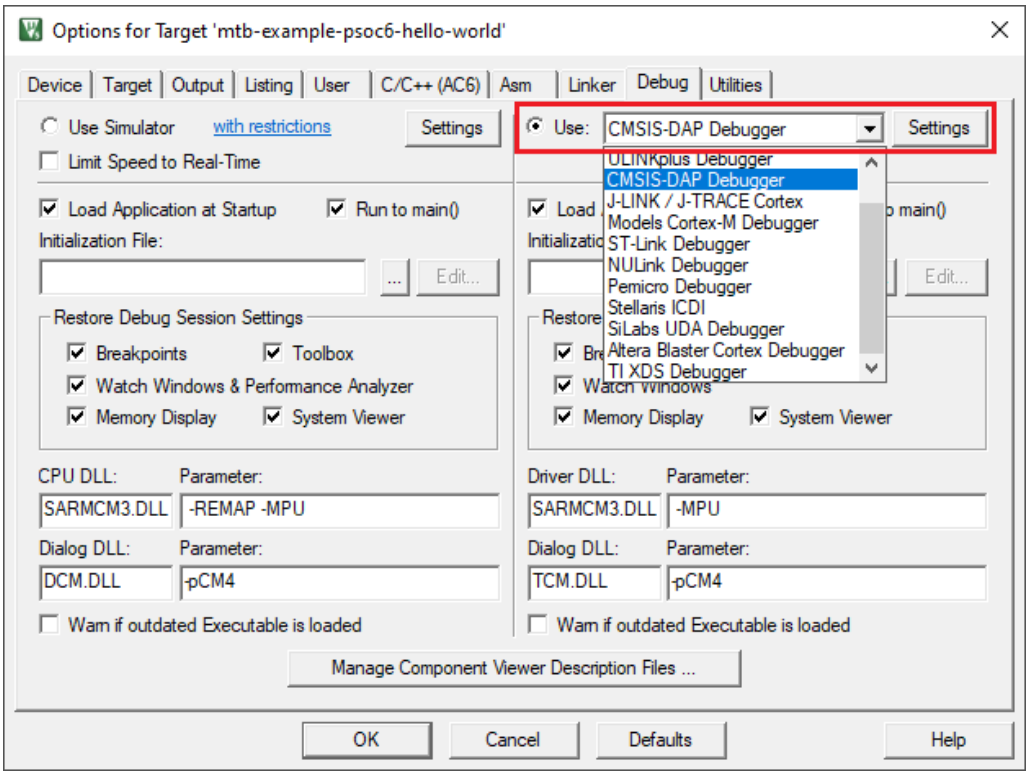


2. On the dialog, select the **C/C++ (AC6)** tab.

- Check that the Language C version was automatically set to c99.
- Select "AC5-like warnings" in the Warnings drop-down list.
- Select "-Os balanced" in the Optimization drop-down list.
- To reduce memory usage, select the One ELF Section per Function check box.



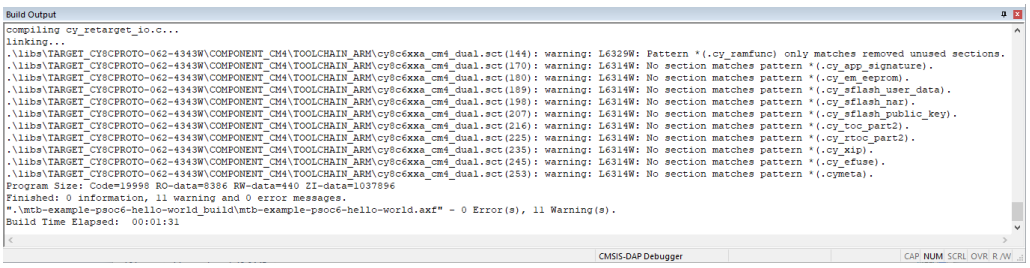
3. Select the **Debug** tab, and select KitProg3 CMSIS-DAP as an active debug adapter:



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

***Note**:* *For applications using the PSOC™ 64 secure single-core application, AIROC™ CYW20829 single-core application, or PSOC™ Control C3 application skip the next step. Instead, perform the steps outlined in the applicable section.*

5. Select **Project > Build target**.

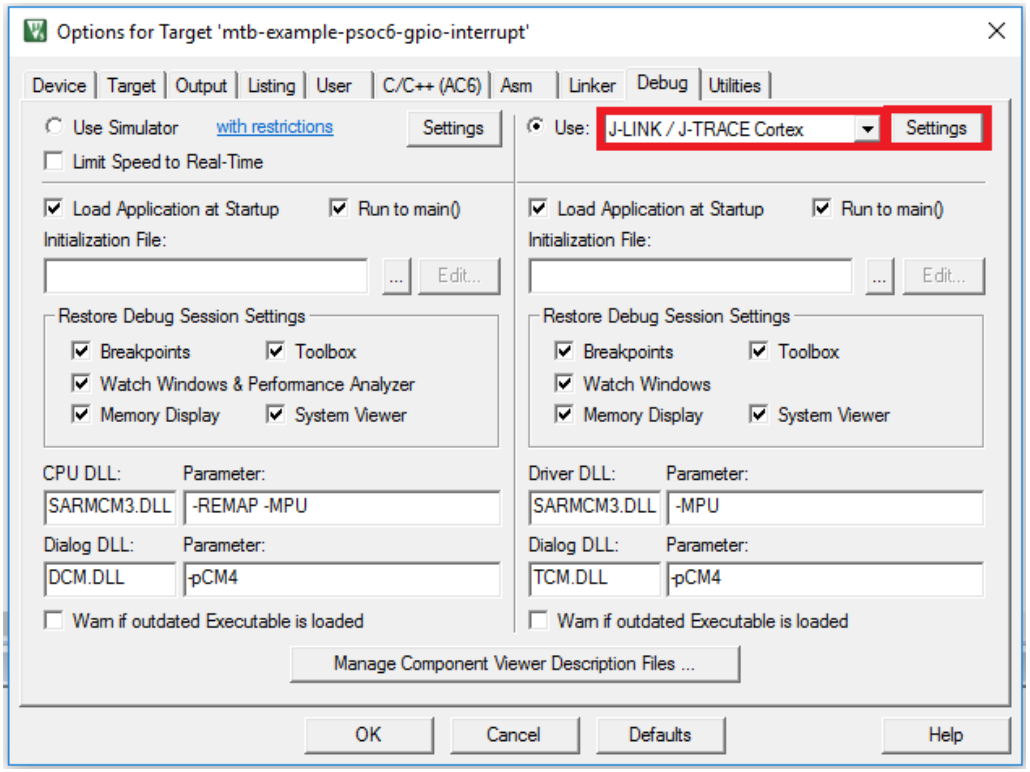


6. To suppress the linker warnings about unused sections defined in the linker scripts, add "6314,6329" to the **Disable Warnings** setting in the Project Linker Options.

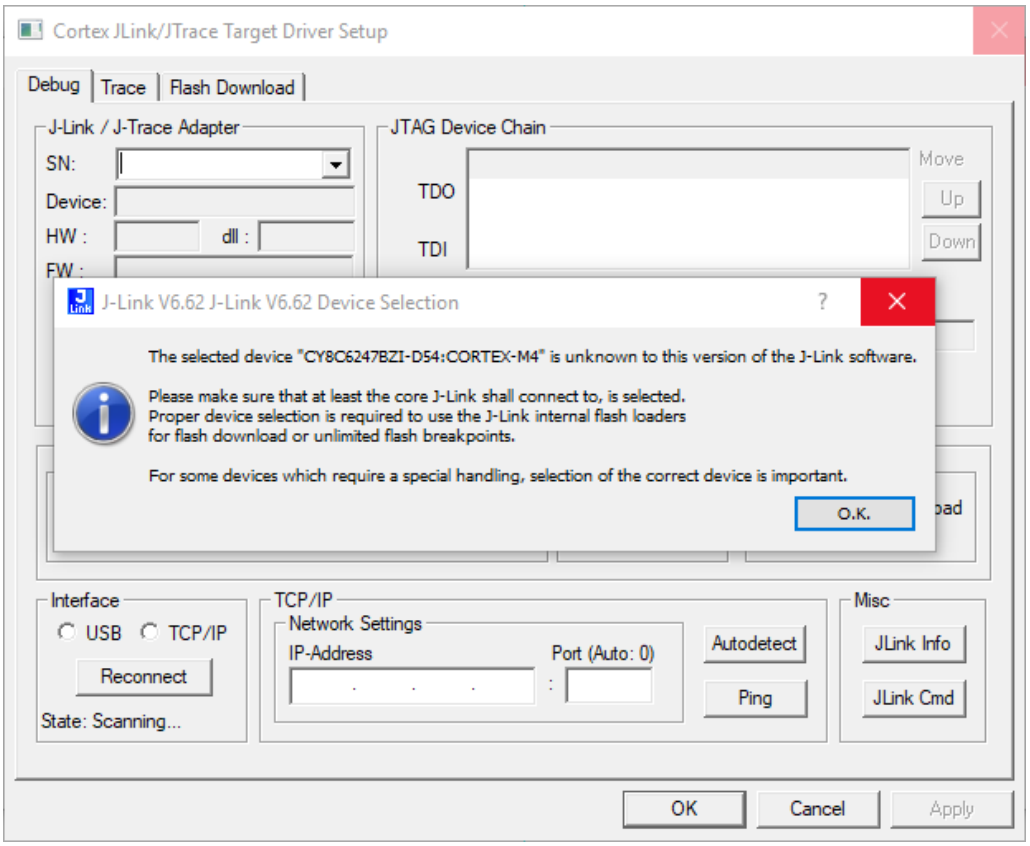
4.2 To use J-Link debugger with PSOC™ MCUs

1. Make sure you have J-Link software version 6.62 or newer.

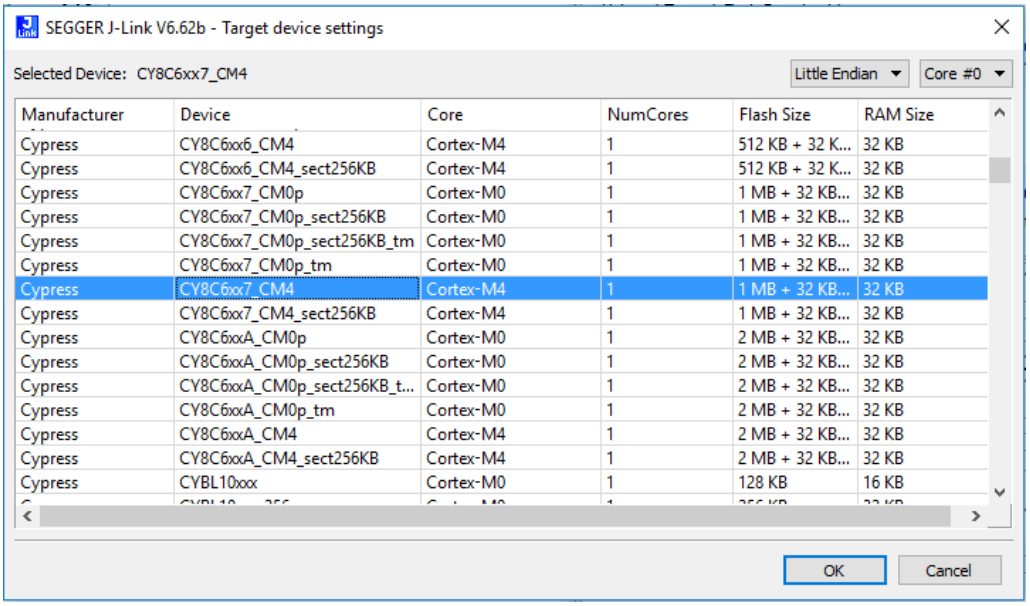
2. Select the **Debug** tab in the Options for Target dialog, select J-LINK / J-TRACE Cortex as debug adapter, and click "Settings":



3. If you see the following message, click **OK** in the Device selection message box:

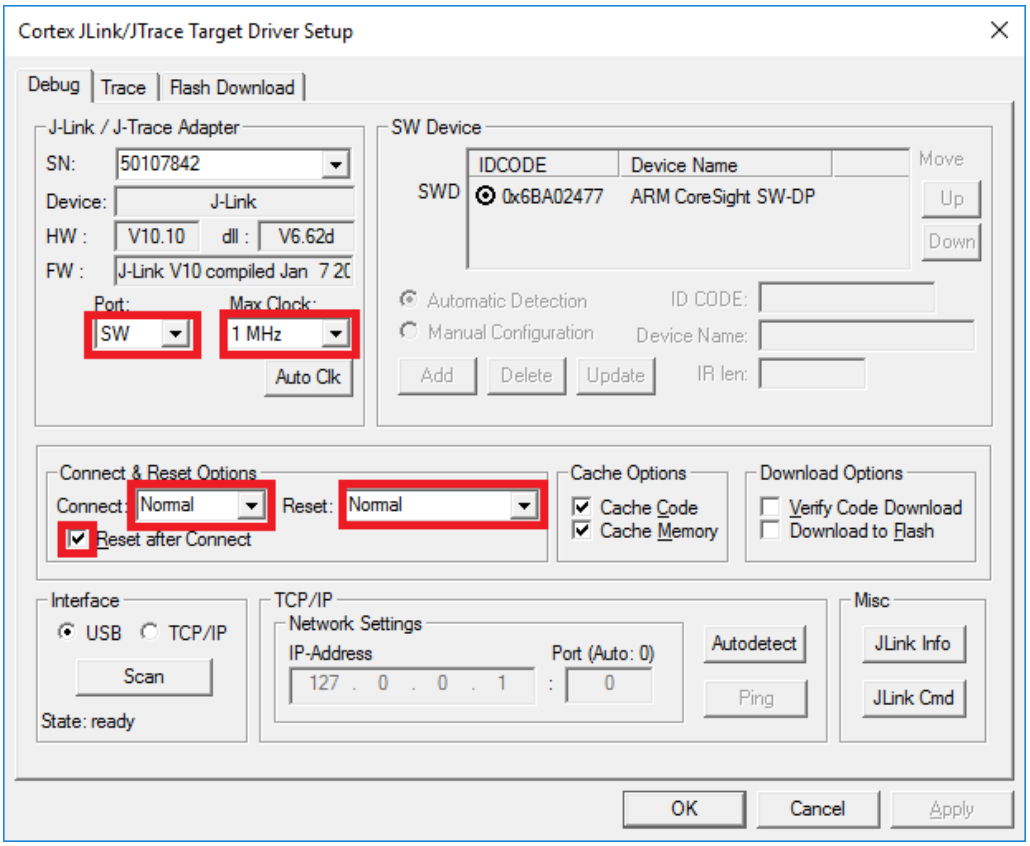


4. Select the appropriate target in Wizard:

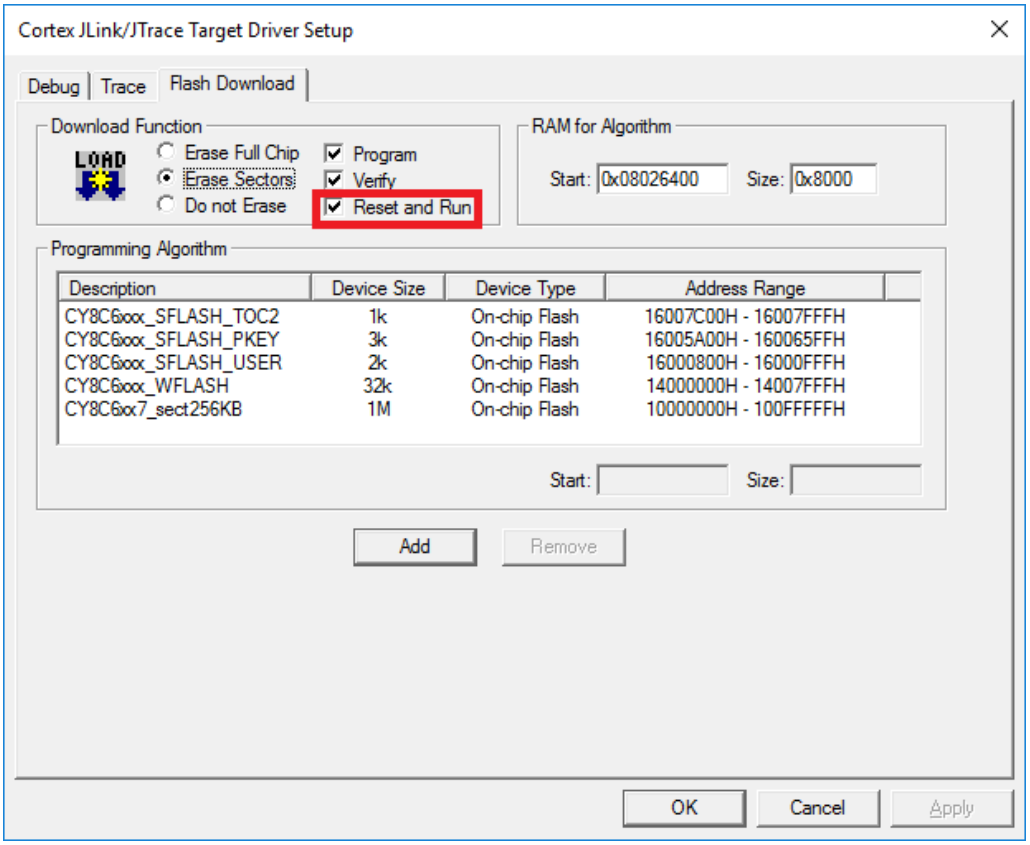


5. Go to **Debug** tab in Target Driver Setup dialog and select:

- set Port to "SW"
- set Max Clock to "1 MHz"
- set Connect to "Normal"
- set Reset to "Normal"
- enable Reset after Connect option



6. Select the **Flash Download** tab in Target Driver Setup dialog and select "Reset and Run" option after download if needed:

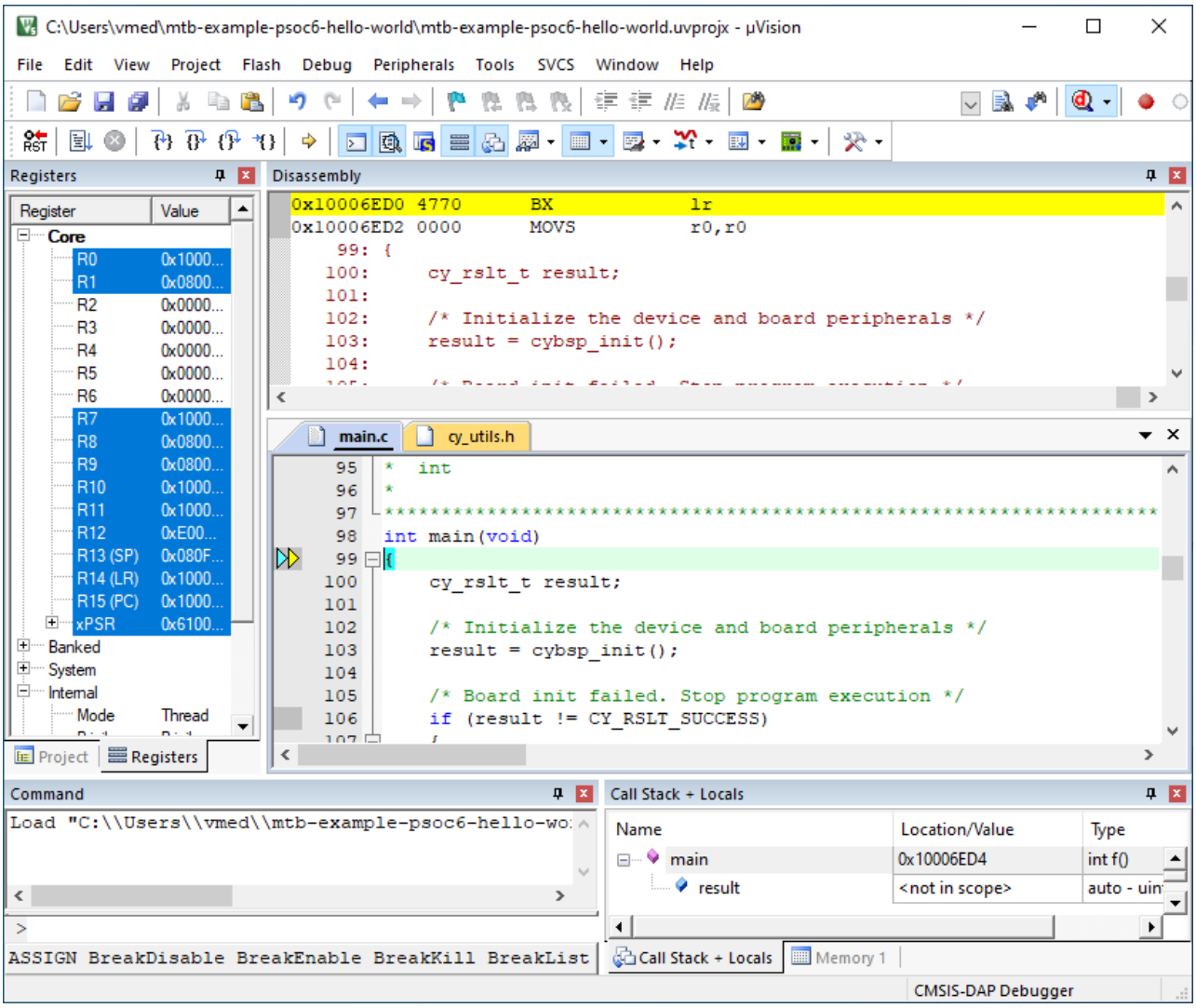


4.3 Programming/Debugging

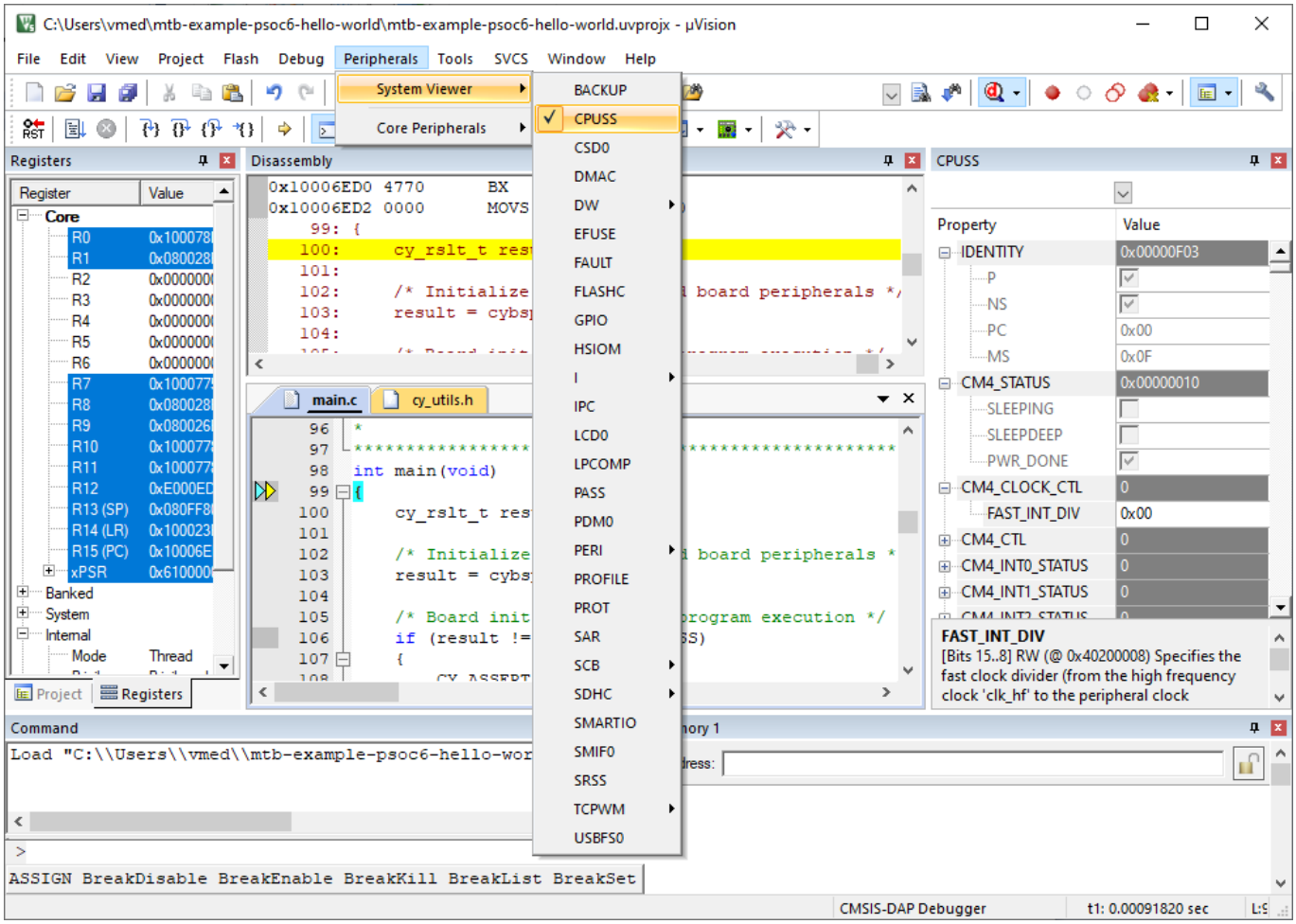
1. Connect the PSOC™ 6 kit to the host PC.

2. As needed, run the fw-loader tool to make sure the board firmware is upgraded to KitProg3. See KitProg3 User Guide for details. The tool is located in this directory by default:
   [install-path]/ModusToolboxProgtools-[version]fw-loader/bin/

3. Select **Debug > Start/Stop Debug Session**.



You can view the system and peripheral registers in the SVD view.

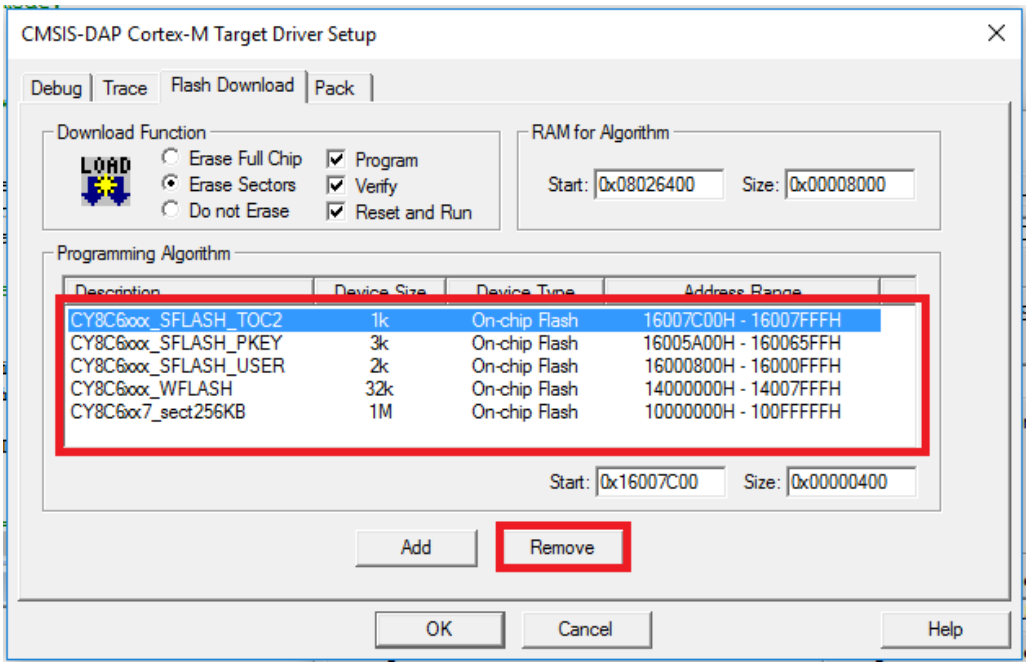


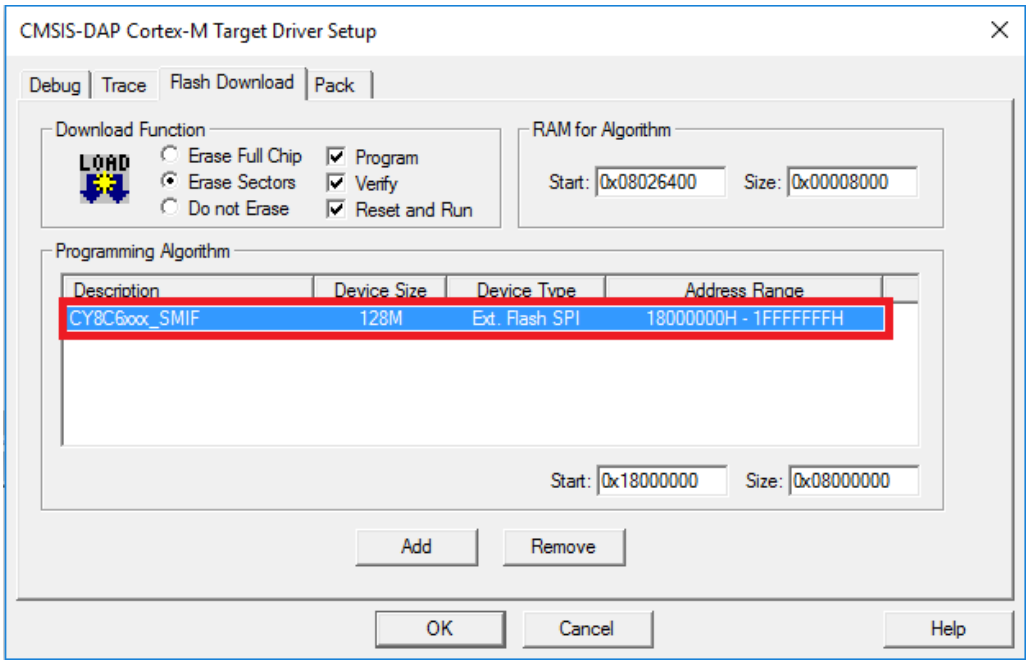
4.3.1 Program external memory

1. Download internal flash as described above.

Notice "No Algorithm found for: 18000000H - 1800FFFFH" warning.

2. Select the Flash Download tab in Target Driver Setup dialog and remove all programming algorithms for On-chip Flash and add programming algorithm for External Flash SPI:





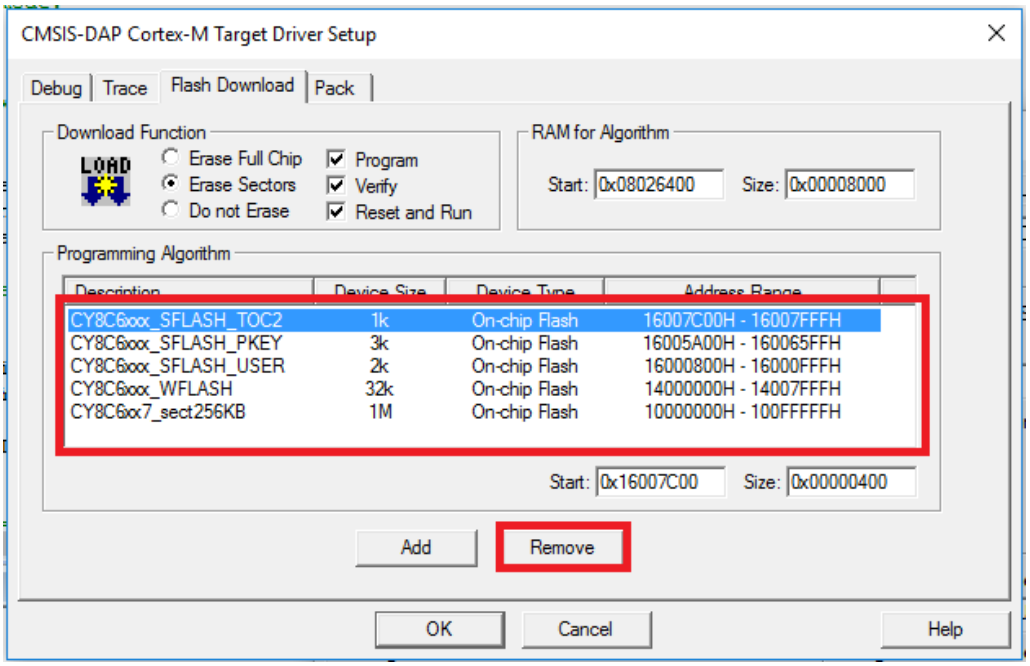
3. Download flash.

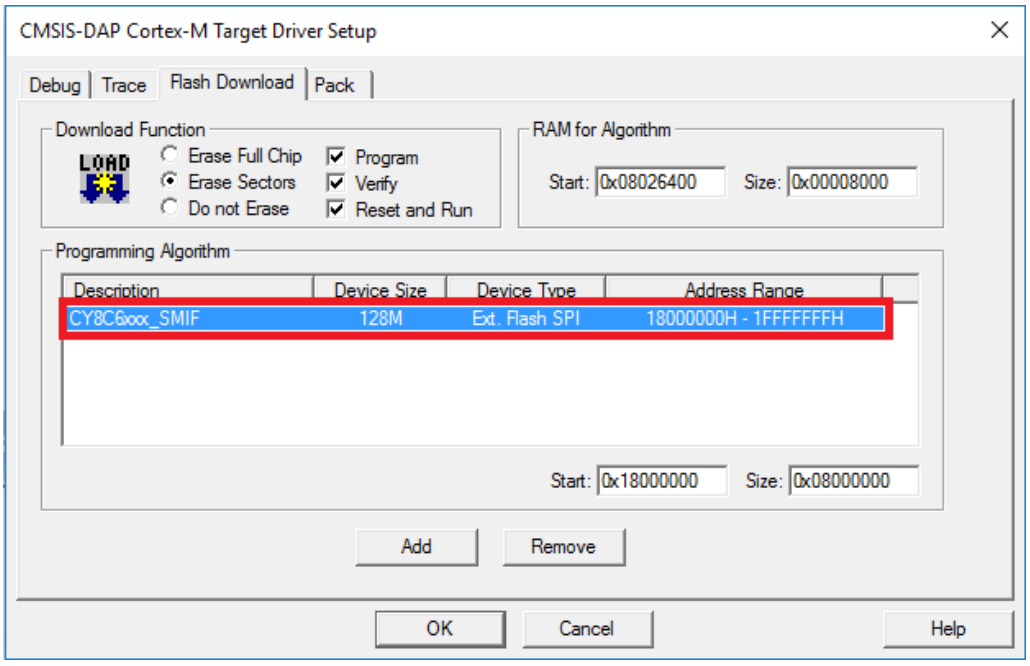
Notice warnings:

- No Algorithm found for: 10000000H - 1000182FH
- No Algorithm found for: 10002000H - 10007E5BH
- No Algorithm found for: 16007C00H - 16007DFFH

4.3.2 Erase external memory

1. Select the **Flash Download** tab in Target Driver Setup dialog and remove all programming algorithms for On-chip Flash and add programming algorithm for External Flash SPI:





2. Click **Flash > Erase** in menu bar.

5 PSOC™ 64 secure single-core application

Follow instructions in Create/export application for Keil µVision. 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 these instructions to configure, build, and debug your application.

1. Build the application using the ModusToolbox™ `make build` command. You can do this by using a stand-alone terminal (modus-shell for Windows), or by exporting the application to Eclipse or VS Code and using the terminal in the IDE.

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

```
.../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_3.3/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
```

   ***Note**: Python was removed from the tools package in version 3.2 and later. The path here is a suggestion. Refer to this KBA for more details:* https://community.infineon.com/t5/Knowledge-Base-Articles/Using-Python-with-a-ModusToolbox-application/ta-p/709737

3. Paste the command into an appropriate editor, and make the following edits:
   - Change `--toolchain-path` to the ARM toolchain; for example, *C:/Keil_v5/ARM/ARMCLANG*
   - Change `--toolchain` to ARM
   - Change `--build-dir` to Keil µVision build directory

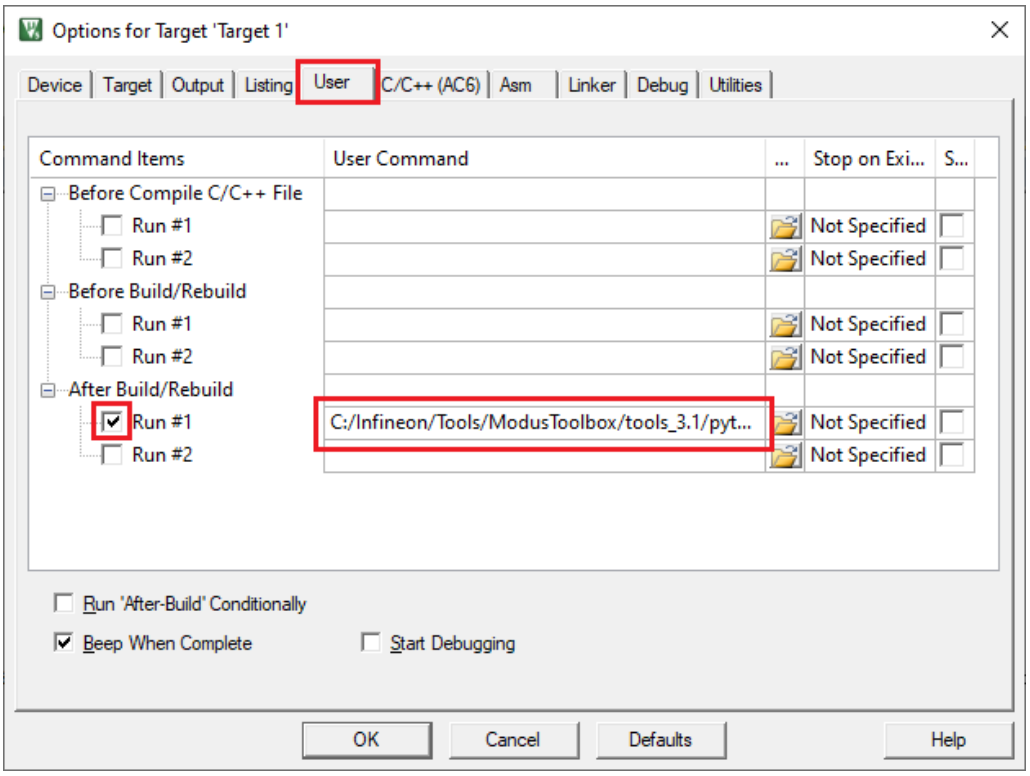
   Example of command after edit:

```
[path-topython]/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:/Keil_v5/ARM/ARMCLANG
--toolchain ARM --build-dir C:/UG/CY8CPROTO-064S1-SB_Secure_Blinky_LED_FreeRTOS_UG/
mtb-example-psoc6-secure-blinkyled-freertos_Objects --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
```

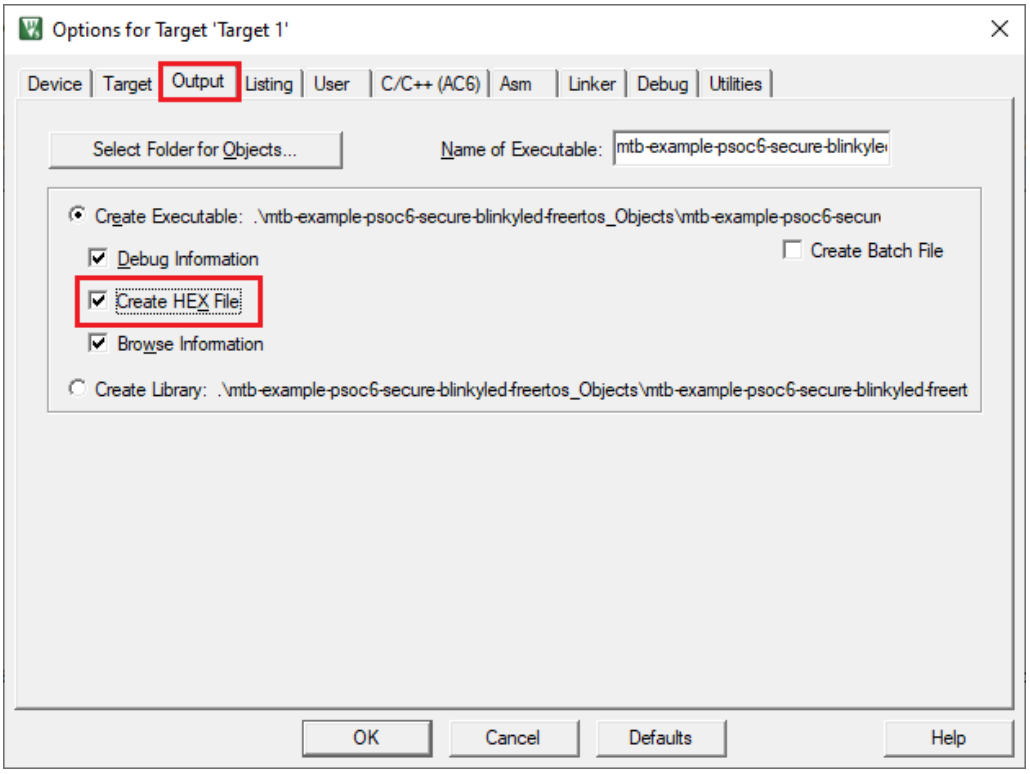
4. Copy the edited command.

5. In µVision, select **Project > Options for Target 'Target 1'...** to open the Options dialog.

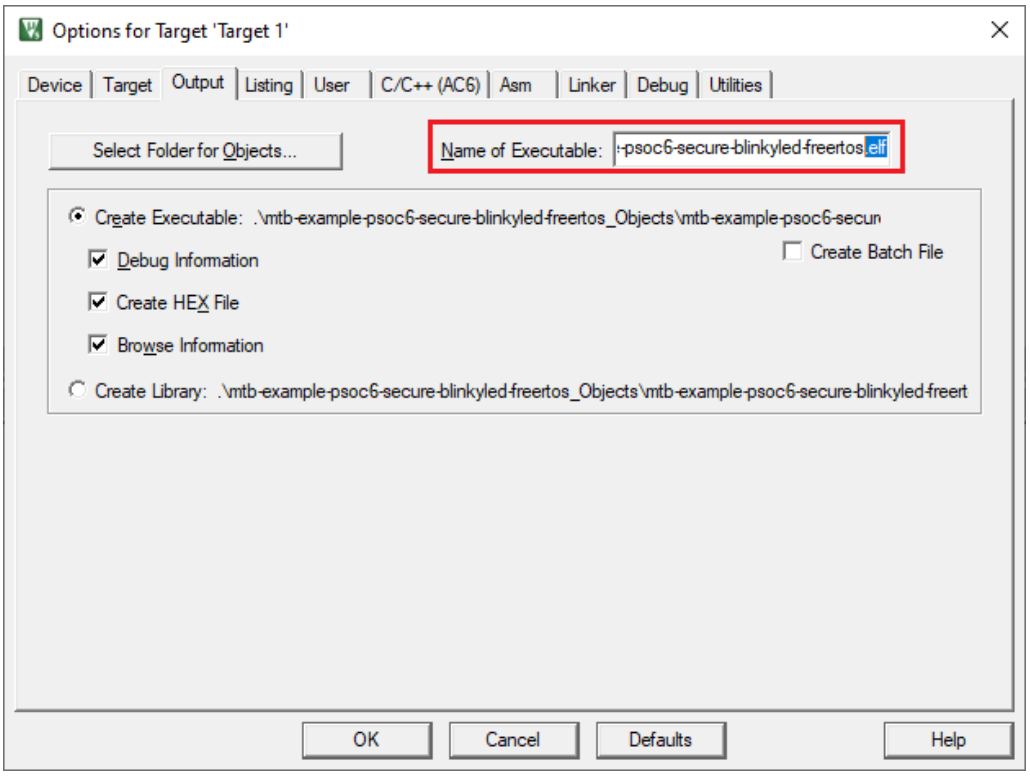
6. Select the **User** tab and enable the **Run #1** check box under **After Build/Rebuild**. Then, paste the edited command.



7. Select the **Output** tab and enable the **Create HEX File** check box.

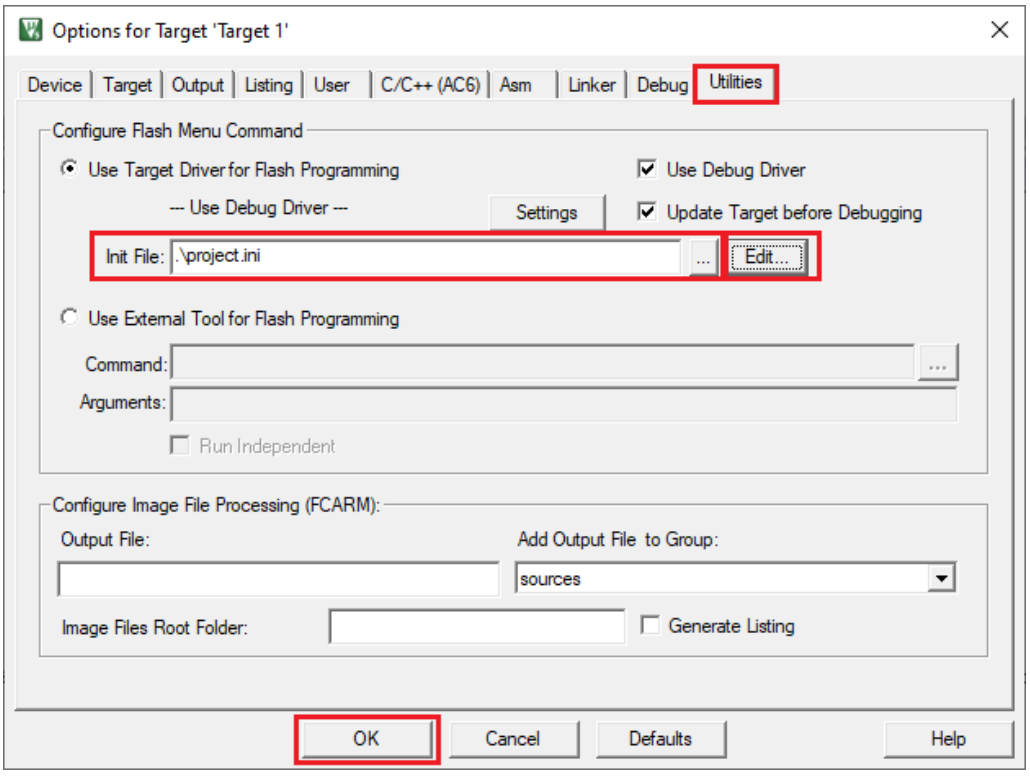


8. On the same **Output** tab, add an ".elf" extension in the **Name of Executable** field.

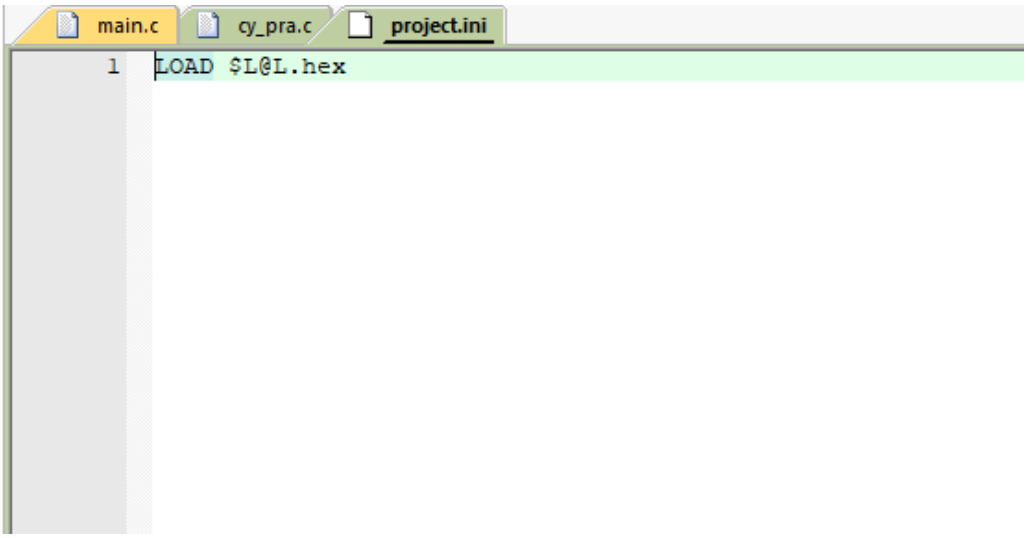


9. Create an empty *project.ini* file in the project root folder.

10. On the Options dialog, select the **Utilities** tab and select the *project.ini* file using the [**...**] button. Then, click **Edit** to open the file and click **OK** to close the options dialog.



11. Type `LOAD $L@L.hex` in the *project.ini* file and save the file.



12. Select **Project > Build Target** to build the application and execute post-build commands.

After performing these steps, you should be able to run debug, erase, and program for PSOC™ 64 secure MCUs.

6 AIROC™ CYW20829 single-core application

Follow instructions in Create/export application for Keil µVision. Then, use these instructions to configure, build, and debug your application.

1. Create a *postbuild.bat* file in the root directory of the project.

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

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

You can do this by using a stand-alone terminal (modus-shell for Windows), or by exporting the application to Eclipse or VS Code and using the terminal in the IDE.

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

```
[path-to-GCC]/bin/arm-none-eabi-objcopy -O ihex C:/20829_keil/Hello_World/build/
APP_CYW920829M2EVK-02/Debug/mtb-example-hal-hello-world.elf C:/20829_keil/Hello_World/
build/APP_CYW920829M2EVK-02/Debug/mtb-example-hal-hello-world.hex

[path-to-GCC]/bin/arm-none-eabi-objcopy C:/20829_keil/Hello_World/build/
APP_CYW920829M2EVK-02/Debug/mtb-example-hal-hello-world.elf -S -O binary C:/20829_keil/
Hello_World/build/APP_CYW920829M2EVK-02/Debug/mtb-example-hal-hello-world.bin;

"[path-to-modustoolbox-install/tools_[version]/modus-shell/bin/bash.exe" --
norc --noprofile ../mtb_shared/recipe-make-cat1b/release-v2.2.1/make/scripts/20829/
flash_postbuild.sh "GCC_ARM" "C:/20829_keil/Hello_World/build/APP_CYW920829M2EVK-02/
Debug" "mtb-example-hal-hello-world" "[path-to-GCC]/bin" "[path-to-modustoolbox-install/
tools_[version]/srecord/bin/srec_cat.exe" "0x00002400"

"[path-to-modustoolbox-install/tools_[version]/modus-shell/bin/bash.exe" --
norc --noprofile ../mtb_shared/recipe-make-cat1b/release-v2.2.1/make/scripts/20829/
run_toc2_generator.sh "NORMAL_NO_SECURE" "C:/20829_keil/Hello_World/build/
APP_CYW920829M2EVK-02/Debug" "mtb-example-hal-hello-world" "flash" "bsps/
TARGET_APP_CYW920829M2EVK-02" "NONE" "[path-to-GCC]" "" 0x20000 0 "" "0x00002400" "256"

[path-to-modustoolbox-install/tools_[version]/srecord/bin/srec_cat.exe C:/20829_keil/
Hello_World/build/APP_CYW920829M2EVK-02/Debug/mtb-example-hal-hello-world.final.bin
-Binary -offset 0x60000000 -o C:/20829_keil/Hello_World/build/APP_CYW920829M2EVK-02/Debug/
mtb-example-hal-hello-world.final.hex -Intel -Output_Block_Size=16

rm -rf C:/20829_keil/Hello_World/build/APP_CYW920829M2EVK-02/Debug/mtb-example-hal-hello-
world.bin;
```

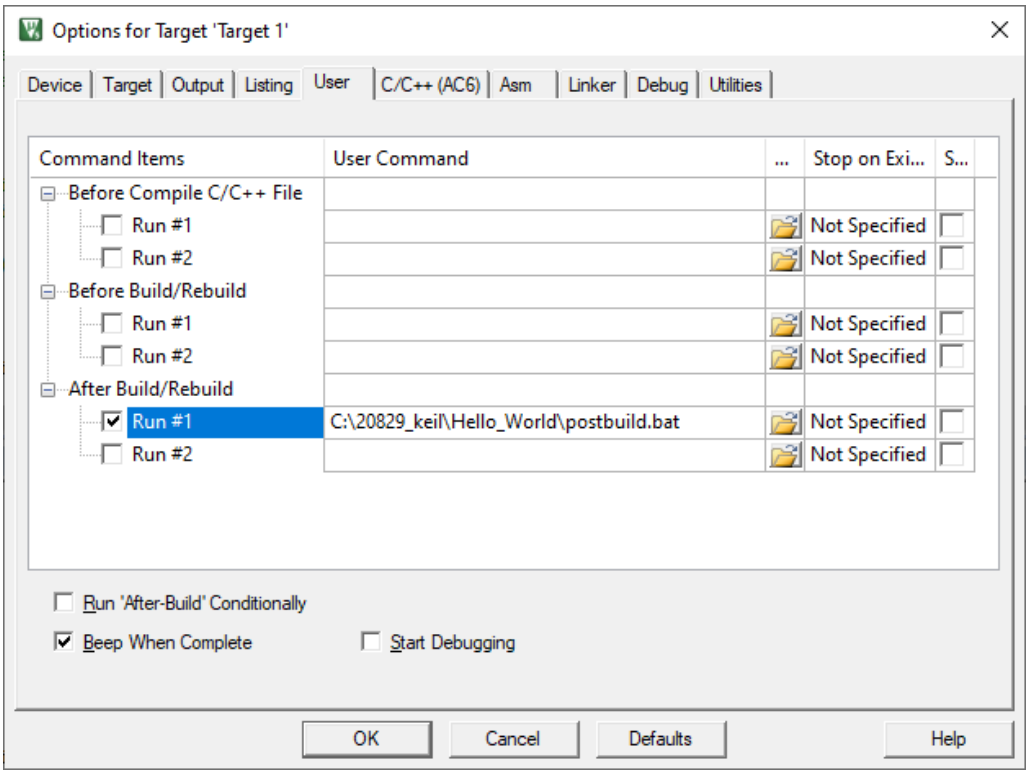
5. Paste the command into the *postbuild.bat* file and make changes to the file to match the Keil µVision structure (changing paths to .elf and .hex files, etc.):

Example of file after edit:

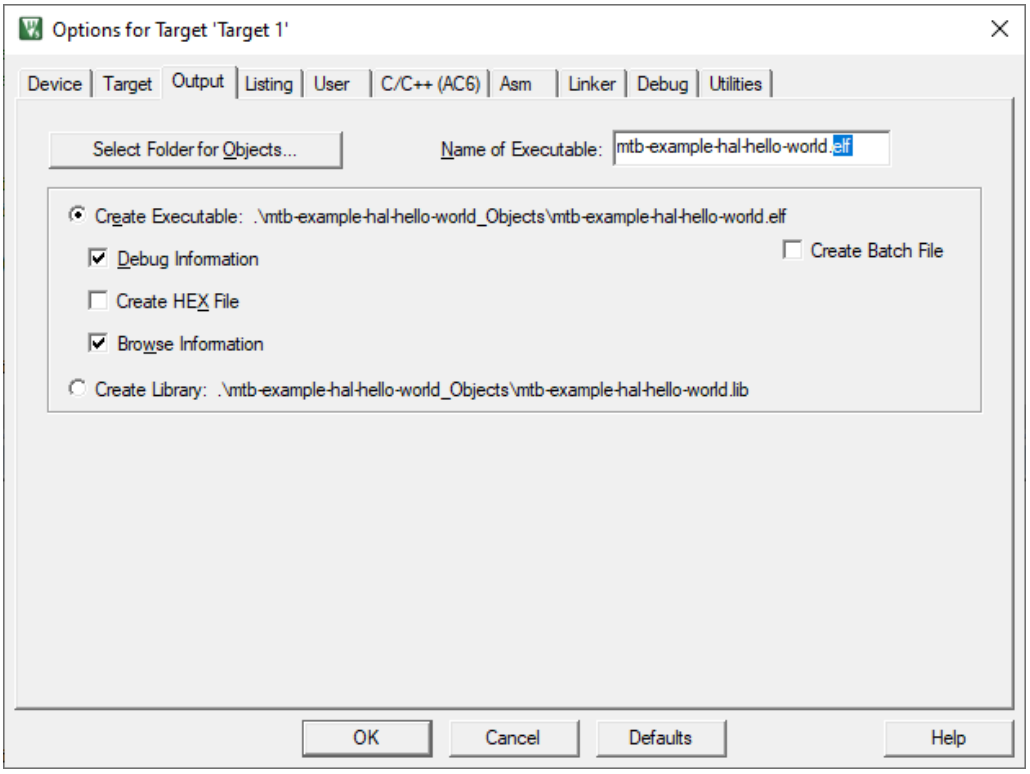
```
SET PATH=%PATH%;"/usr/bin"

"C:/Keil_v5/ARM/ARMCLANG/bin/fromelf" C:/20829_keil/Hello_World/mtb-example-hal-hello-
world_Objects/mtb-example-hal-hello-world.elf --bin --output=C:/20829_keil/Hello_World/
mtb-example-hal-hello-world_Objects/mtb-example-hal-hello-world.bin

"[path-to-modustoolbox-install/tools_[version]/modus-shell/bin/bash.exe" --
norc --noprofile ../mtb_shared/recipe-make-cat1b/release-v2.2.1/make/scripts/20829/
flash_postbuild.sh "ARM" "C:/20829_keil/Hello_World/mtb-example-hal-hello-world_Objects"
"mtb-example-hal-hello-world" "[path-to-GCC]/bin" "[path-to-modustoolbox-install/
tools_[version]/srecord/bin/srec_cat.exe" "0x00002400"

"[path-to-modustoolbox-install/tools_[version]/modus-shell/bin/bash.exe" --
norc --noprofile ../mtb_shared/recipe-make-cat1b/release-v2.2.1/make/scripts/20829/
run_toc2_generator.sh "NORMAL_NO_SECURE" "C:/20829_keil/Hello_World/mtb-example-hal-hello-
world_Objects" "mtb-example-hal-hello-world" "flash" "bsps/TARGET_APP_CYW920829M2EVK-02"
"NONE" "[path-to-GCC]" "" 0x20000 0 "" "0x00002400" "256"

[path-to-modustoolbox-install/tools_[version]/srecord/bin/srec_cat.exe C:/20829_keil/
Hello_World/mtb-example-hal-hello-world_Objects/mtb-example-hal-hello-world.final.bin
-Binary -offset 0x60000000 -o C:/20829_keil/Hello_World/mtb-example-hal-hello-
world_Objects/mtb-example-hal-hello-world.final.hex -Intel -Output_Block_Size=16

del "C:\20829_keil\Hello_World\mtb-example-hal-hello-world_Objects\mtb-example-hal-hello-
world.bin"
```

6. Save the *postbuild.bat* file.

7. Open the **Project > Options for Target 'Target 1'** dialog, select the **User** tab, select the **After build/ Rebuild run #1** check box, and select the *postbuild.bat* file:

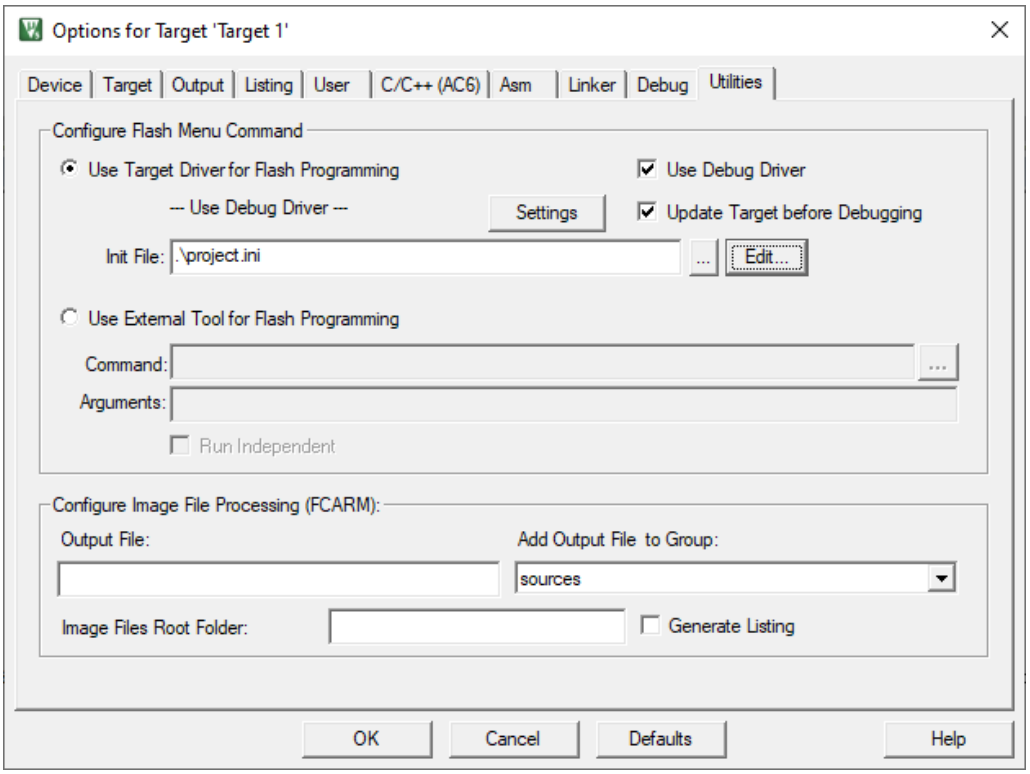


8. Select the **Output** tab and add a ".elf" extension to the file in the **Name of Executable** field:



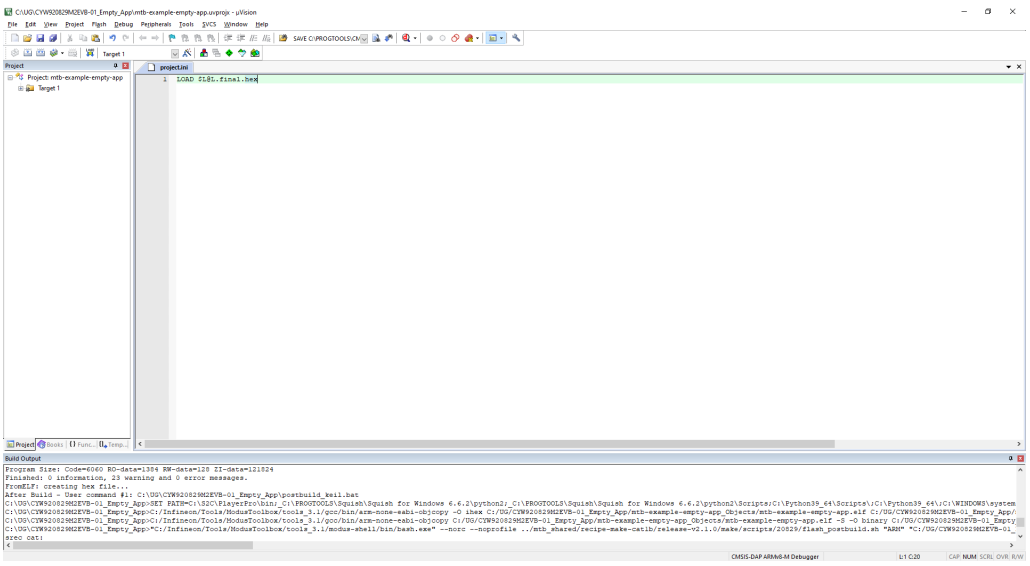
9. Create empty *project.ini* and *project_2.ini* files in the project root folder.

10. Select the **Utilities** tab, select the *project.ini* file in the **Init File** field, click the **Edit** button, and click **OK**:



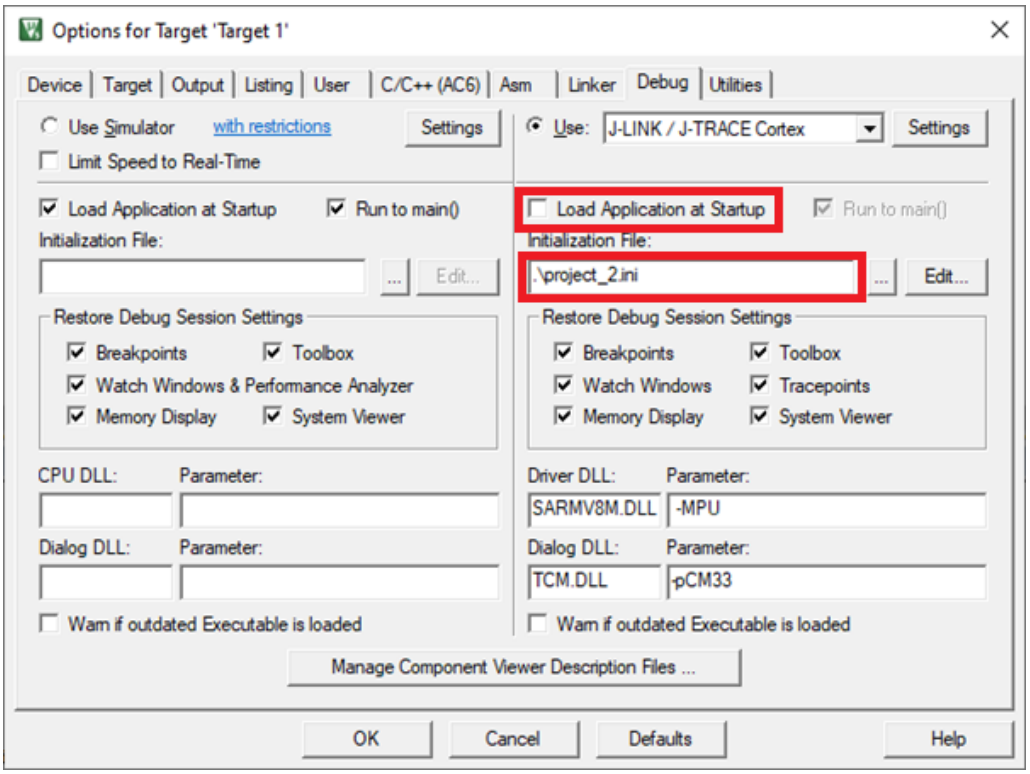
11. Add the following text to the *project.ini file* and click **Save**:

```
LOAD $L@L.final.hex
```



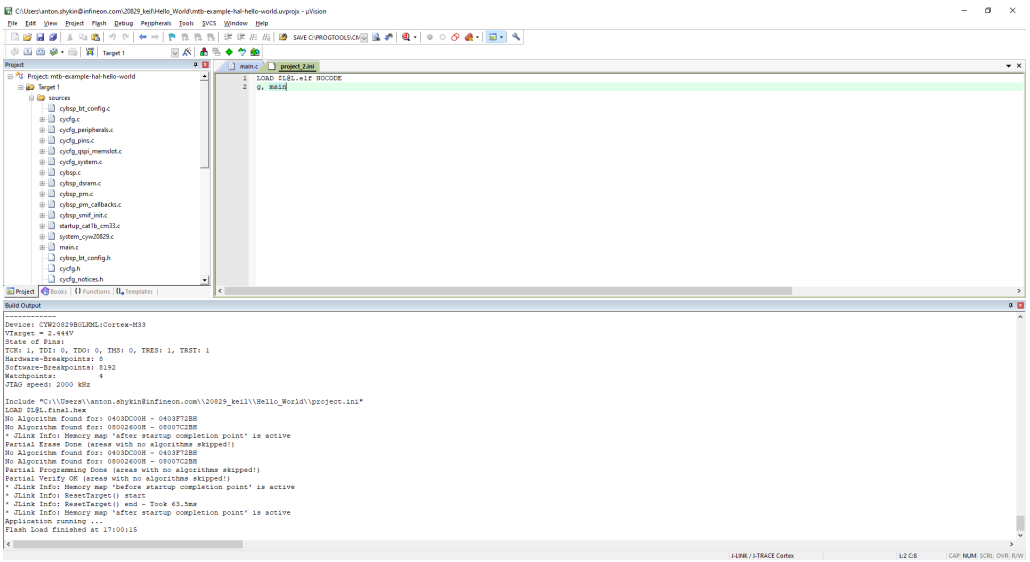
12. Reopen the **Project > Options for Target 'Target 1'** dialog and select the **Debug** tab. Make sure that the **Load Application at Startup** check box is not checked.

13. Select the *project_2.ini* file in the **Initialization File** field, click the **Edit** button, and click **OK**:



14. Add the following text to the *project_2.ini* file and click **Save**.

```
LOAD $L@L.elf NOCODE
                g, main
```



15. Select **Project > Build Target** to build the application and execute post-build commands.

After following these steps, you should be able to run Debug, Erase, and Program for the AIROC™ CYW20829 device.

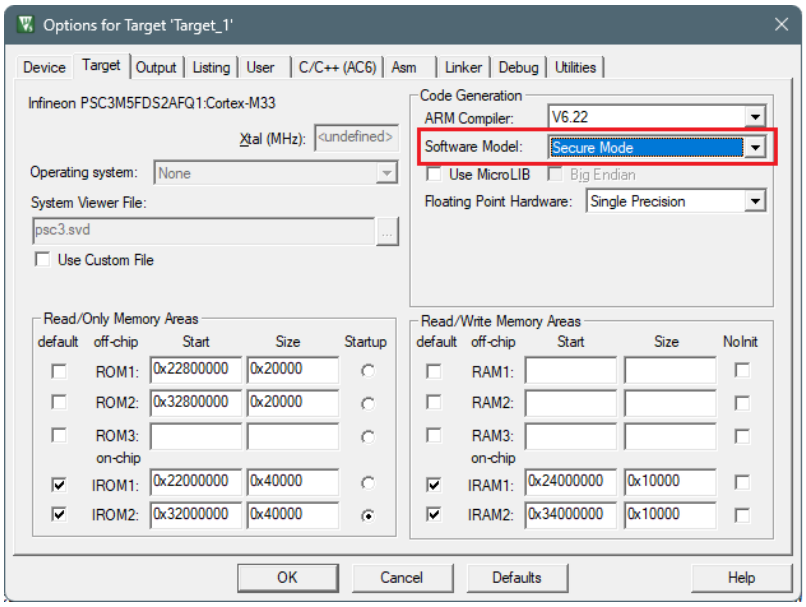
7 PSOC™ Control C3 application

Follow instructions in Create/export application for Keil µVision. 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 UVision IDE:
    Project->Options->Target->Software Model
```

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 Keil µVision, you need to check the TrustZone setting. Open the Options for Target dialog, select the **Target** tab, and ensure the **Software Model** option is set to "Secure Mode".



Save the application and select **Project > Build Target** to build it. The Output should display the progress, ending with text similar to this:

```
linking...
bsps/TARGET_APP_KIT_PSC3M5_EVK/TOOLCHAIN_ARM/linker_s_flash.sct(107): warning: L6329W: Pattern
*(.cy_sharedmem) only matches removed unused sections.
bsps/TARGET_APP_KIT_PSC3M5_EVK/TOOLCHAIN_ARM/linker_s_flash.sct(113): warning: L6314W: No
section matches pattern *(Veneer$$CMSE).
Program Size: Code=16902 RO-data=1438 RW-data=312 ZI-data=64844
Finished: 0 information, 2 warning and 0 error messages.
".\Hello_World_Objects\Hello_World.axf" - 0 Error(s), 2 Warning(s).
Build Time Elapsed:  00:00:22
```

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. The process for each is similar, but there are important differences.

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. In order to create a valid hex file for programming and debugging in this state, we must configure the application

1. Create a *postbuild.bat* file in the root directory of the project. The primary purpose of this is to shift the application image from the C-Bus region to the S-Bus region. This is done because the ARM compiler attempts to copy the Load Region to the Execution Region, which is read-only. The bat file contains three separate lines, for example:

   ```
   C:\Keil_v5\ARM\ARMCLANG\bin\fromelf --output "Hello_World_Objects/tmp.hex" -- i32combined
   "Hello_World_Objects/Hello_world.elf"

   [path-to-gcc]\bin\arm-none-eabi-objcopy -O ihex "Hello_world_Objects/tmp.hex"
   "Hello_world_Objects/Hello_world.hex" --change-addresses 0x20000000

   del "Hello_world_Objects\tmp.hex"
   ```

   Update the paths and file names to match your configuration.

2. Create *project.ini* and *project2.ini* files in the project root folder. These initialization files provide `LOAD` commands that are needed for the application.

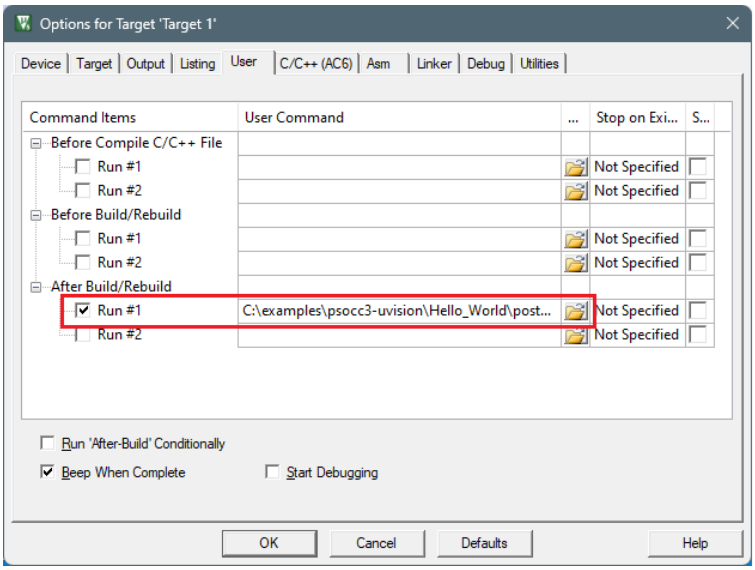
3. Add the following text to the *project.ini* file and click **Save**.

   ```
   LOAD $L@L.hex
   ```

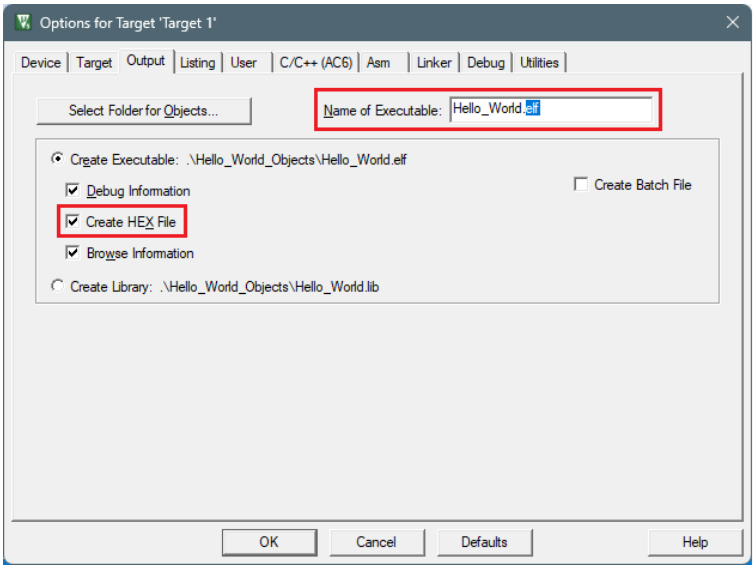
4. Add the following text to the *project2.ini* file and click **Save**.

   ```
   LOAD $L@L.elf NOCODE
   g, main
   ```

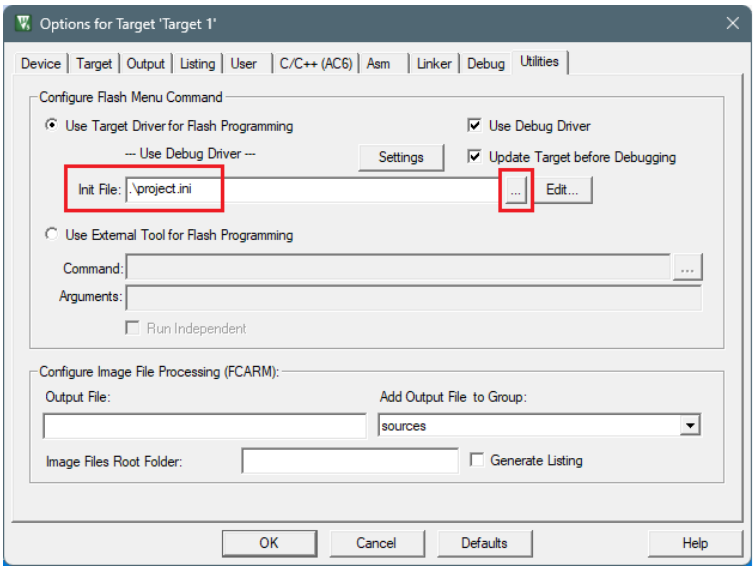
5. Open the Options for Target dialog, select the **User** tab, select the **After Build/Rebuild Run #1** check box, and select the *postbuild.bat* file:



6. Switch to the **Output** tab and add a ".elf" extension to the file in the **Name of Executable** field, and select the **Create HEX File** check box:

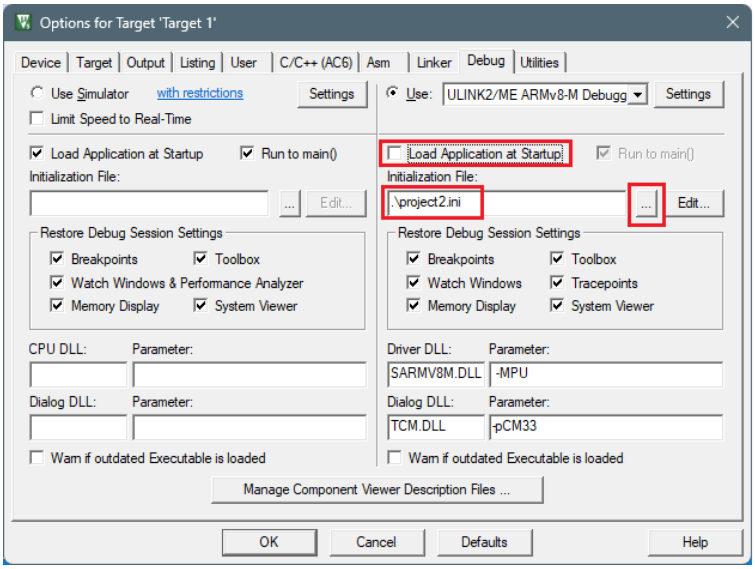


7. Switch to the **Utilities** tab, select the *project.ini* file in the **Init File** field:



8. Switch to the **Debug** tab. Make sure that the **Load Application at Startup** check box is not checked.

9. Select the *project2.ini* file in the **Initialization File** field:



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

11. Select **Project > Build Target** to build the application and execute post-build commands.

12. Configure the applicable debugger settings.

13. To program the device, select **Flash > Download**.

14. To start the debugger, select **Debug > Start/Stop Debug Session**.

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, modify or replace the *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 "Hello_World_Objects/
Hello_World.hex" --output "Hello_World_Objects/Hello_World.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
```

All other configuration steps are the same as those in Device with default policy.

Then, build the application.

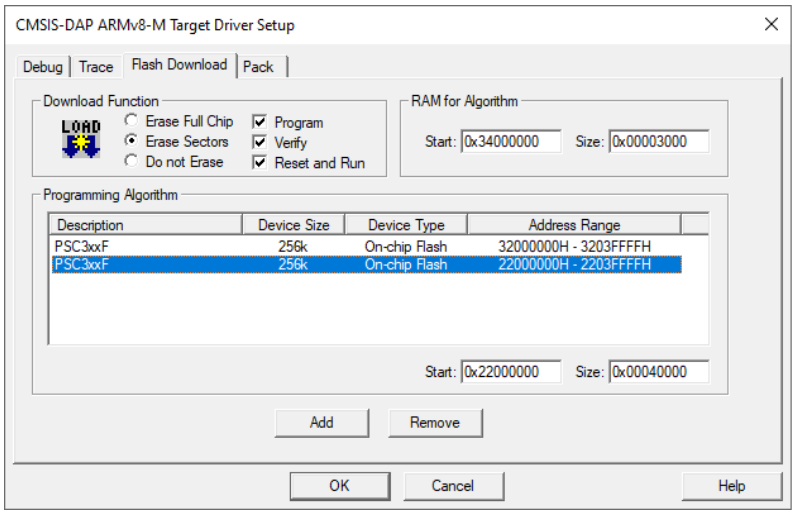
After that, you can program the device and run the debugger.

7.3 Erase external memory

For Infineon PSOC™ Control C3 devices, the same physical flash memory can be erased/programmed over SBUS 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, this means non-secure access only).

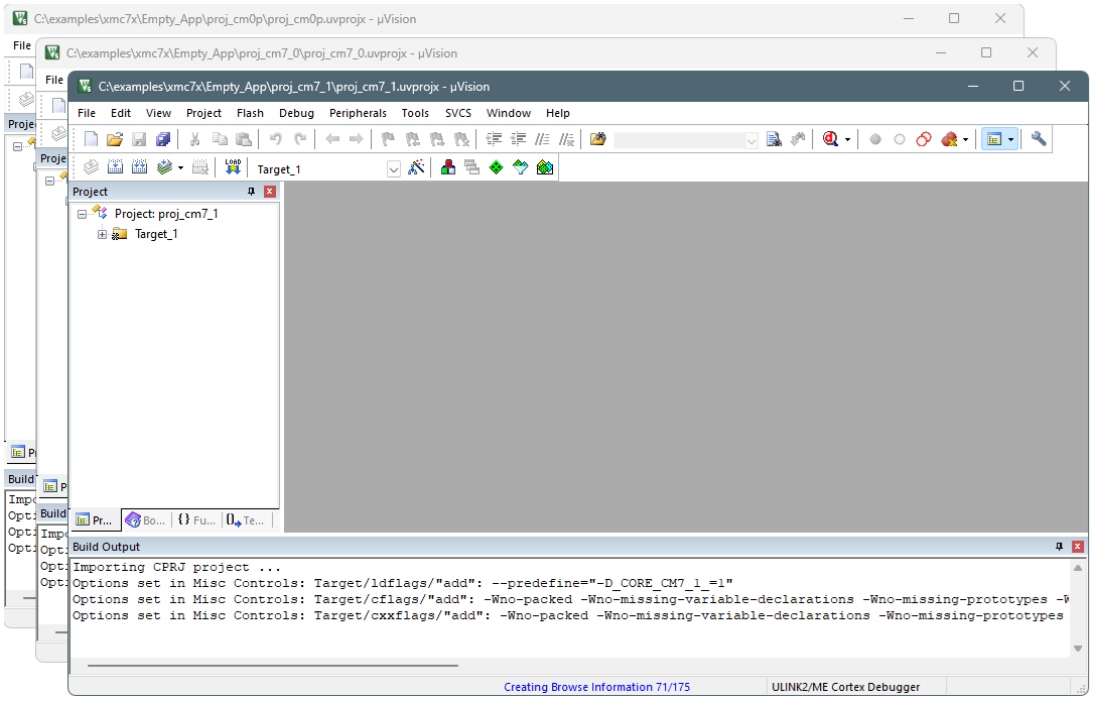
To erase the entire flash memory, it's important to only erase the non-secure range of addresses. Remove the flashloader with the secure address range from the list prior to performing the erase.



However, if the application has data that must be programmed through a secure address range, the previously deleted flashloader needs to be re-added by using the **Add** button and specifying the secure address range.

8 PSOC™ 6 and XMC7xxx multi-core application

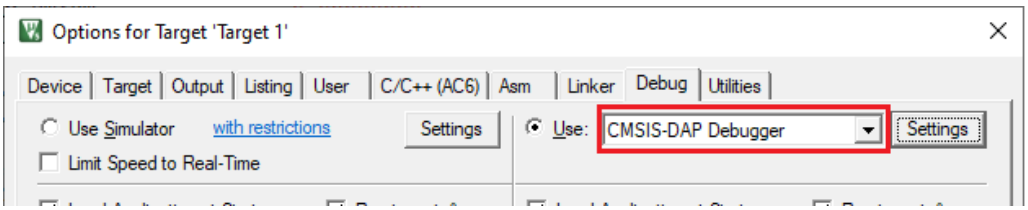
Follow instructions in Create/export application for Keil µVision. After creating all the projects in separate instances, your application should look like this:



Then, use the instructions in this section to configure, build, and debug your application.

8.1 Configure CM0+ project

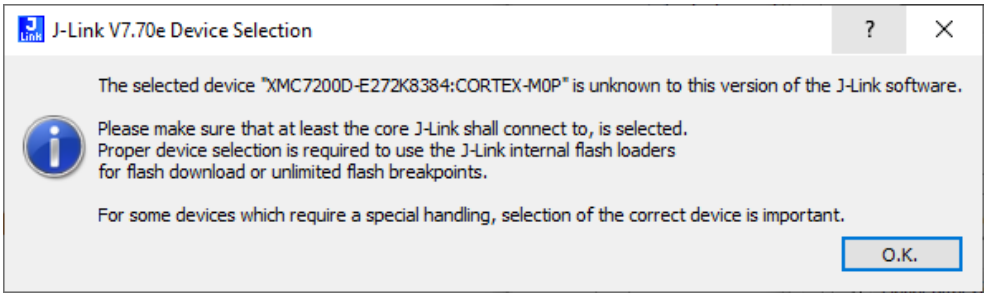
1. Go to **Project > Options for Target <target_name>**, switch to the **Debug** tab, select the applicable debug probe (CMSIS-DAP or J-Link) as shown:



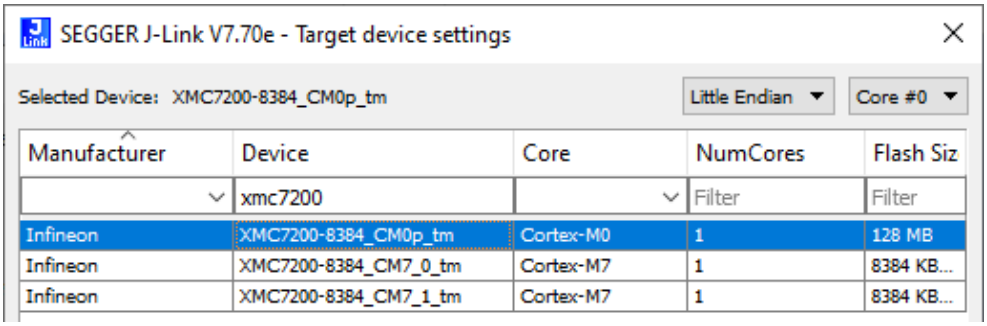
- If using ULINK2, select the **CMSIS-DAP Debugger** option as the debug probe, because the ULINK2 driver does not support multi-core debugging

2. Click the **Settings** button to configure the target driver.

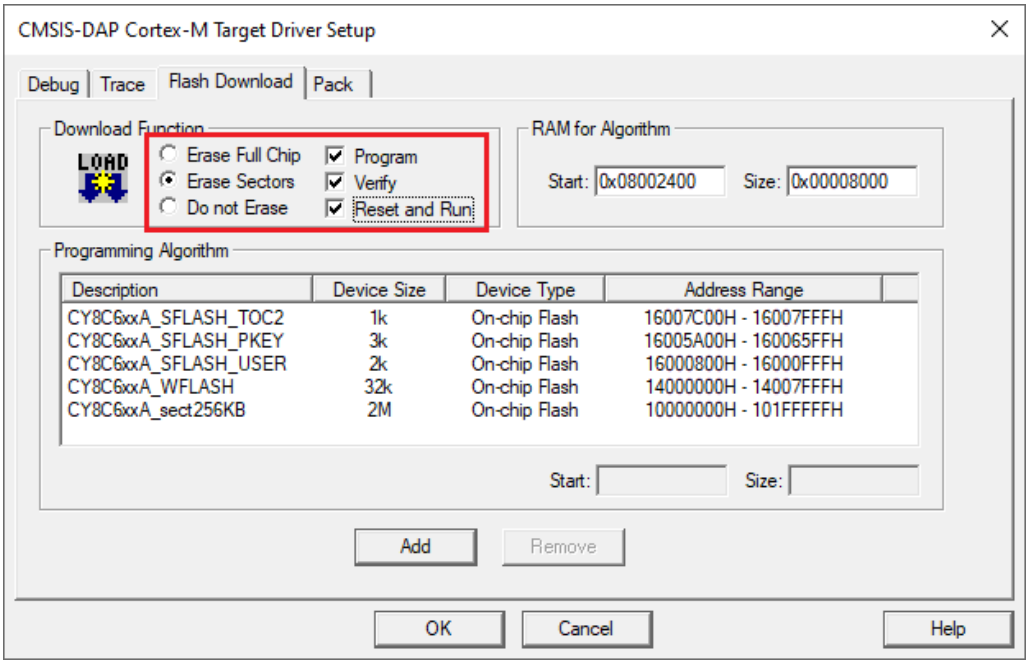
- If you select the J-Link probe, a pop-up window might display reporting that the device is unknown to J-Link software.



- If so, click **OK** and select the device manually in the opened Target device settings dialog. For XMC7200 devices, there will be three aliases, each dedicated to a separate core.



3. Switch to the **Flash Download** tab, select the **Erase Sectors** radio button, and select the **Program**, **Verify**, and **Reset and Run** check boxes.



4. Click **OK** to close the Target Driver Setup dialog.

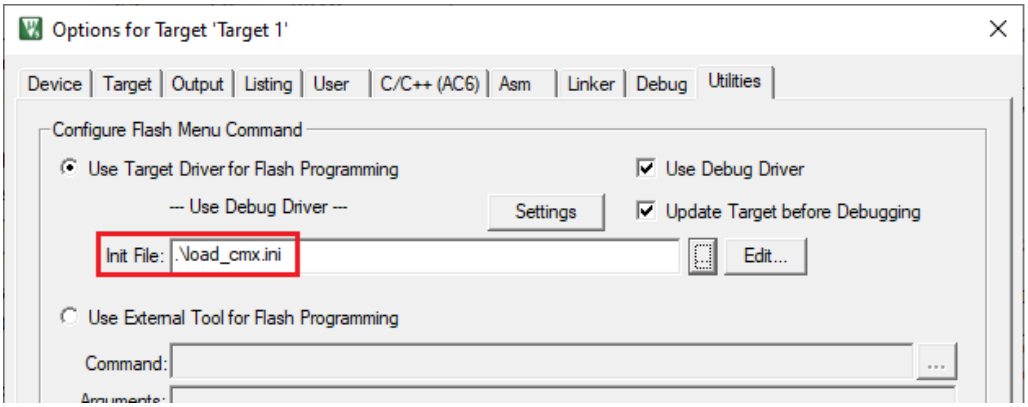
5. Next, configure the project so that it also programs other image(s) from the CM4/CM7 project(s). Do this using the **.ini* file.

- Create a new empty file named load_cmx.ini and save it inside the CM0+ project directory.
- Add a LOAD command with a path to the CM4/CM7 images. For example:

```
LOAD "..\\proj_cm4\\proj_cm4_Objects\\proj_cm4.axf"
```

- Add as many LOAD commands for all the CM4/CM7 projects as you have.

6. Go to **Project > Options for Target <target_name>**, select the **Utilities** tab, and specify the created *load_cmx.ini* file in the *Init File* edit field.



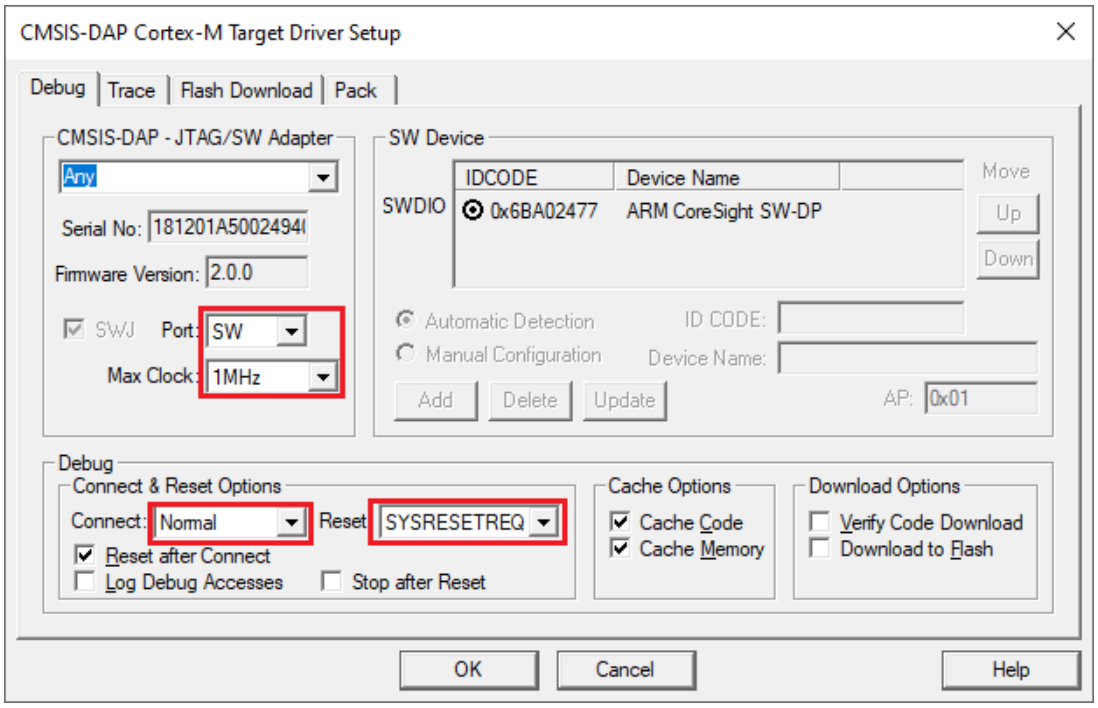
7. Switch to the **Debug** tab, and click the **Settings** button.

The configuration settings are different for CMSIS-DAP/ULINK2 and J-Link. Refer to the following sections for the applicable options:

CMSIS-DAP/ULINK2 Target Driver Setup

Use the following options:

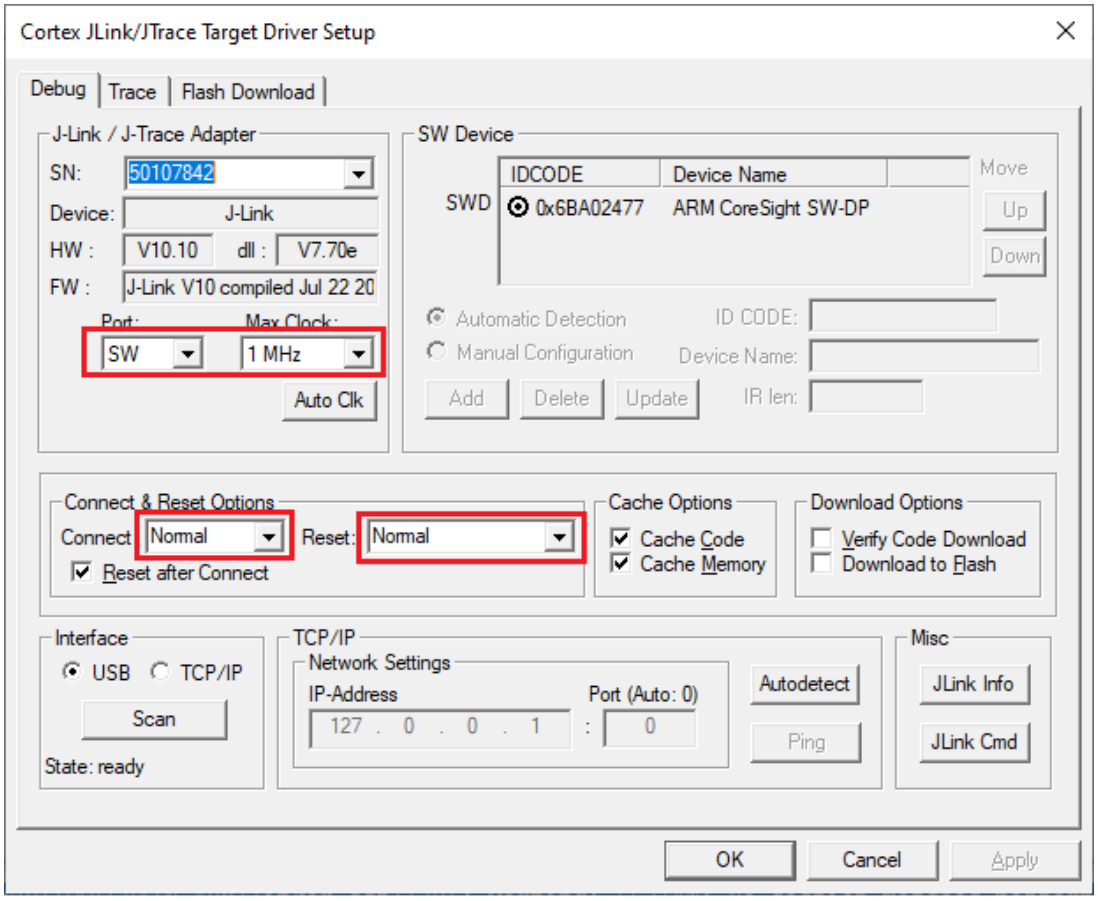
- **Port**: SW
- **Max Clock**: 1 MHz
- **Connect**: Normal
- **Reset**: SYSRESETREQ



J-Link Target Driver Setup

Use the following options:

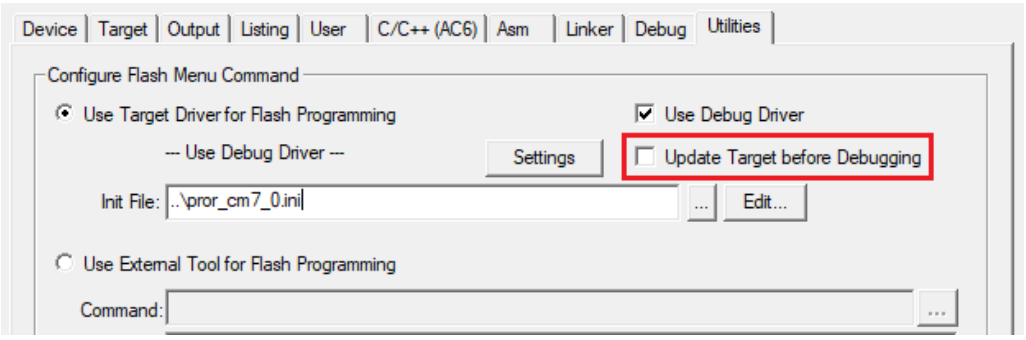
- **Port**: SW
- **Max clock**: 1 MHz
- **Connect**: Normal
- **Reset**: Normal



That completes configuring the CM0+ project. The next step is to configure CM4/CM7 project(s).

8.2 Configure CM4/CM7 project

1. Open the Options dialog to the **Utilities** tab, and deselect the **Update Target before the Debugging** check box.

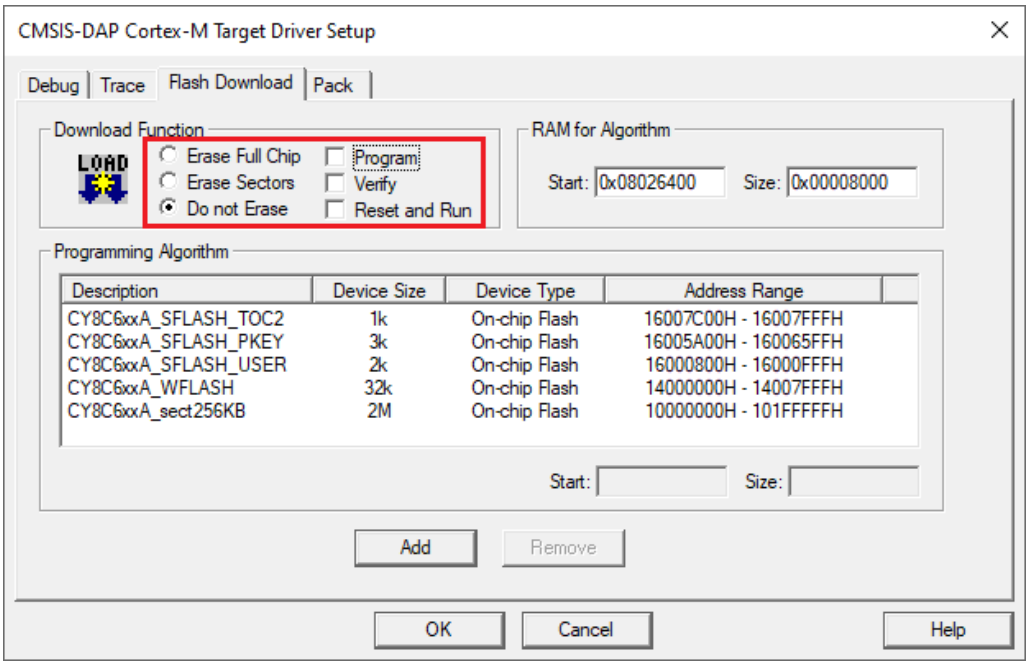


2. Switch to the **Debug** tab, select the applicable debug probe (CMSIS-DAP or J-Link).

- If using ULINK2, select the **CMSIS-DAP Debugger** option as the debug probe, the ULINK2 driver does not support multi-core debugging.

3. Click the **Settings** button to configure the target driver.

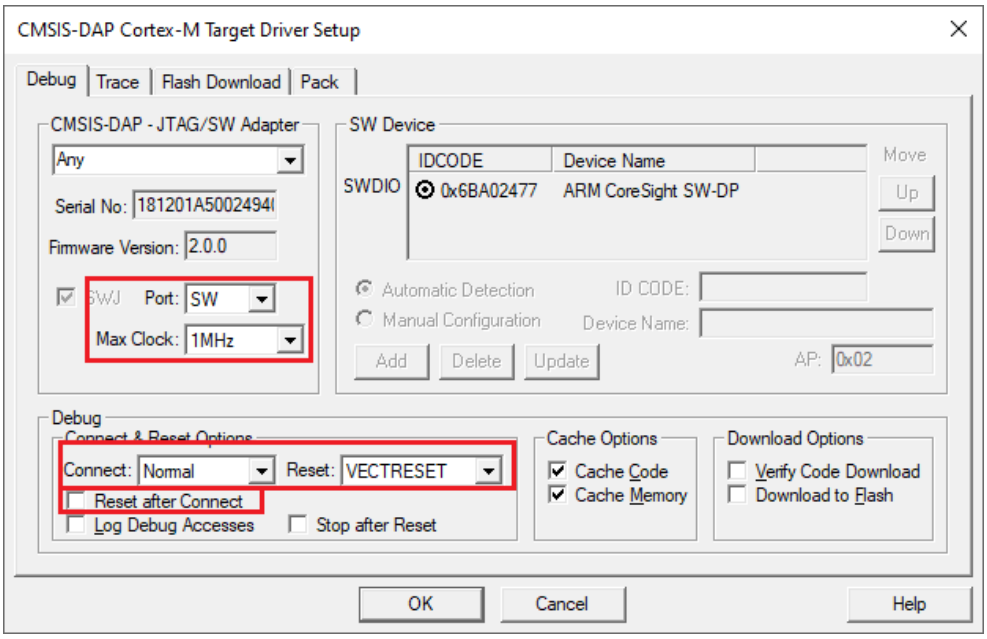
4. On the Target Driver Setup dialog, switch to the **Flash Download** tab, select the **Do not Erase** radio button, and deselect the **Program**, **Verify**, and **Reset and Run** check boxes.



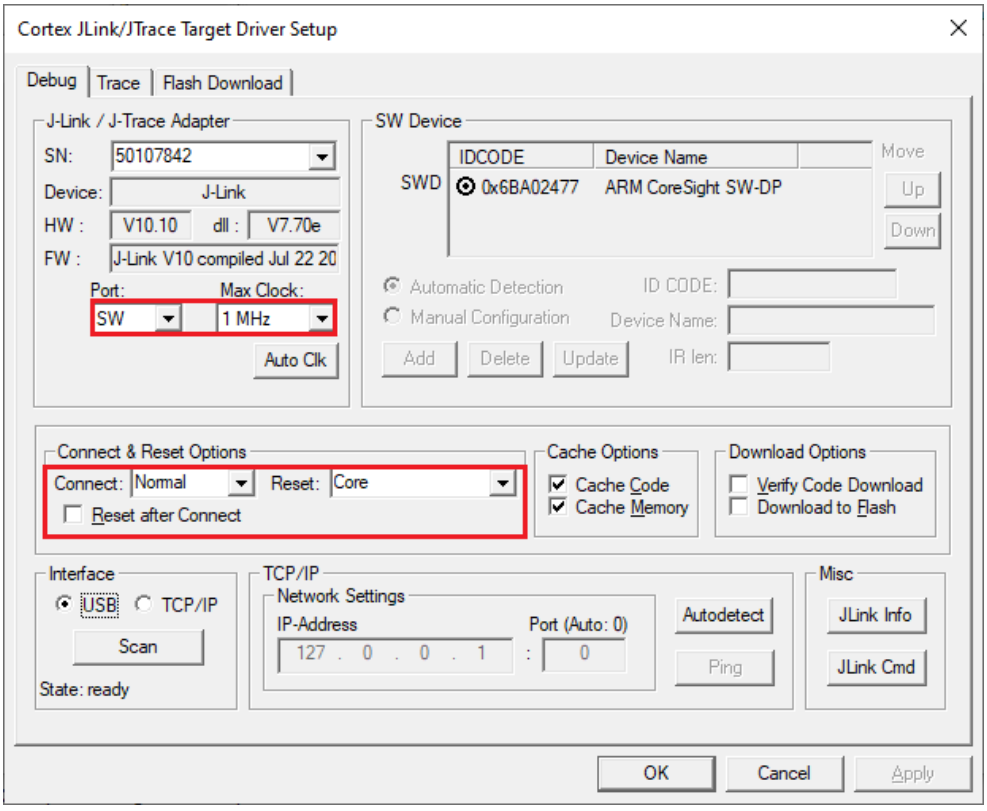
5. Switch to the **Debug** tab.

The configuration settings are different for CMSIS-DAP/ULINK2 and J-Link. Refer to the following for the appropriate options:

- **CMSIS-DAP/ULINK2 Target Driver Setup** – Use the following options:
  - **Port**: SW
  - **Max Clock**: 1 MHz
  - **Connect**: Normal
  - **Reset**: VECTRESET
  - **Reset after Connect** check box: deselected



- **J-Link Target Driver Setup** – Use the following options:
  - **Port**: SW
  - **Max Clock**: 1 MHz
  - **Connect**: Normal
  - **Reset**: Core
  - **Reset after Connect** check box: deselected



6. Click **OK** to close the Target Driver Setup dialog.

7. Save the project(s).

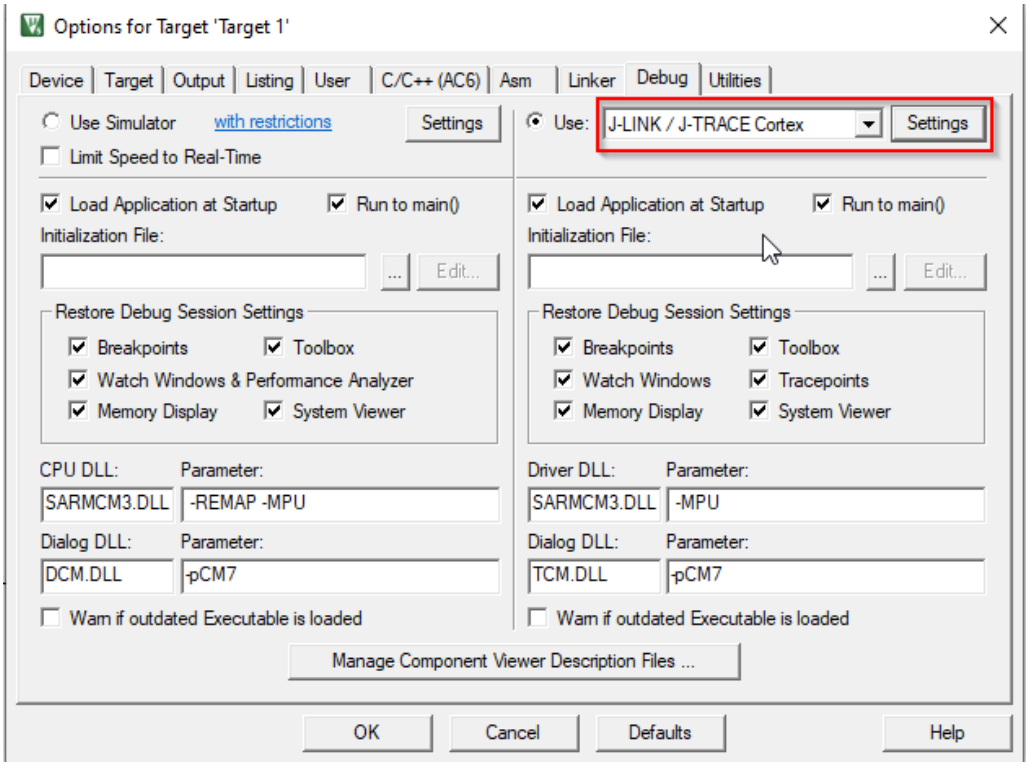
8.3 Building µVision multi-core projects

Once all projects are open, go ahead and build each one using **Project > Build target**.

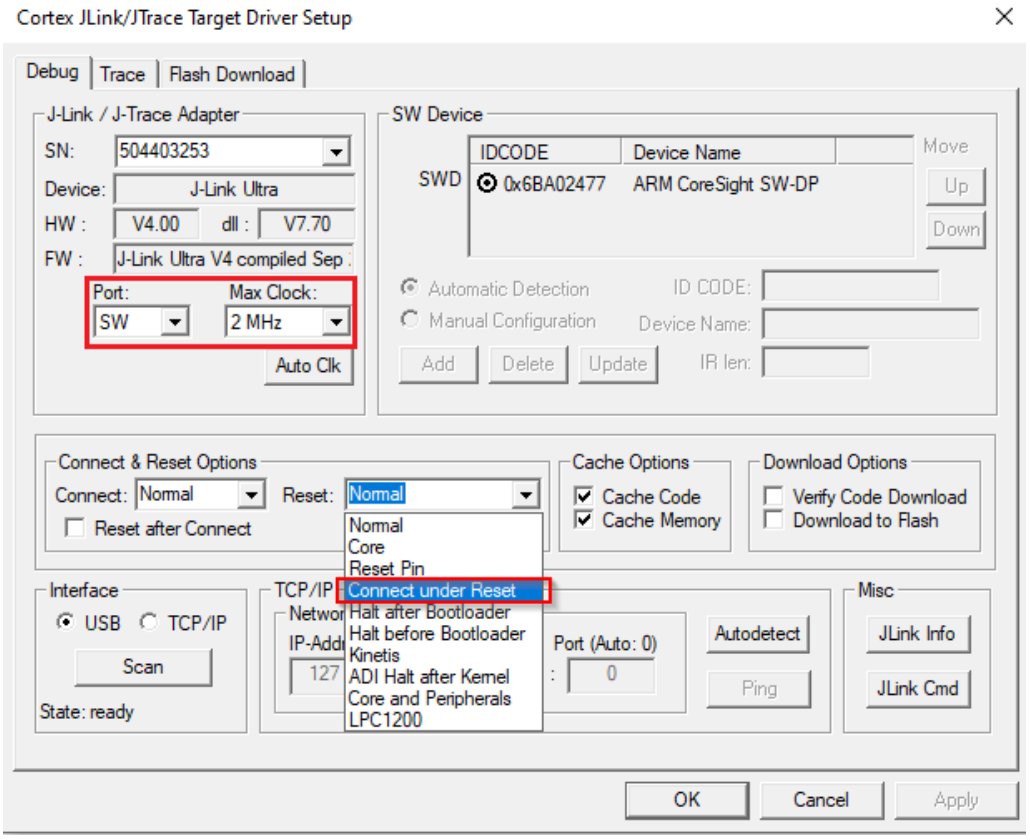
8.4 To use J-Link debugger with XMC7000 devices

8.4.1 Program set-up instructions

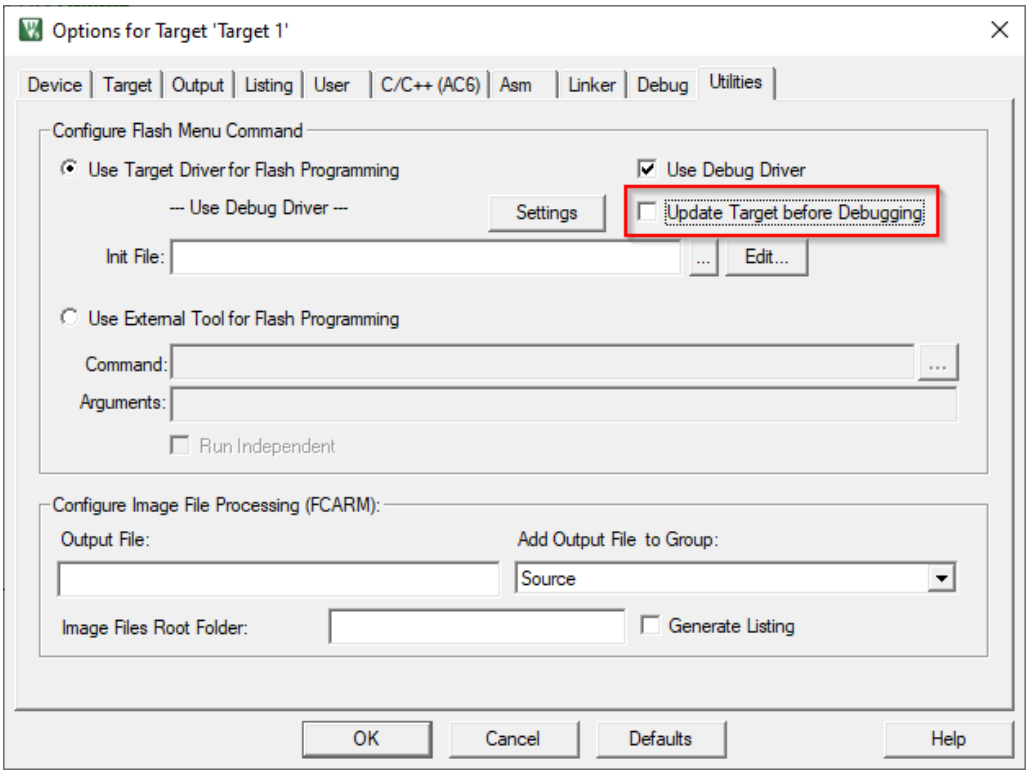
1. Select the **Debug** tab in the Options for Target dialog, select "J-LINK / J-TRACE Cortex" as the debug adapter, and click **Settings**.



2. On the **Debug** tab in the Cortex J-Link/JTrace Target Driver Setup dialog, select "SW" for **Port**, 2 MHz for **Max Clock**, and "Connect under Reset" on the **Reset** pull-down menu, and then click **OK**.

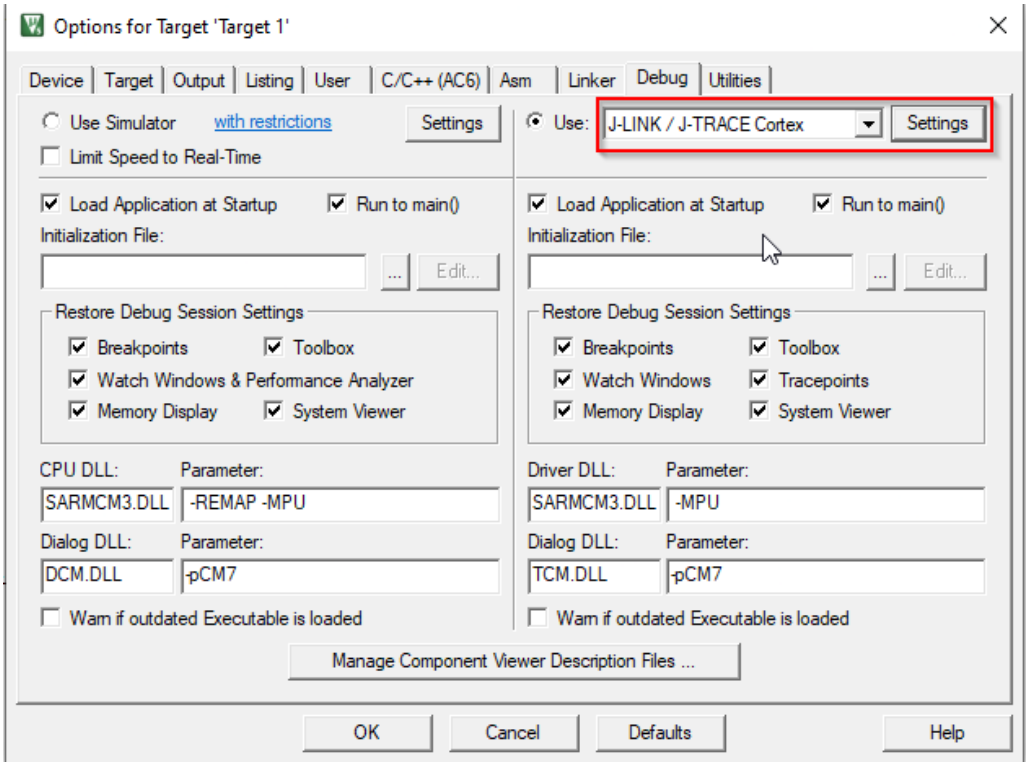


3. Select the **Utilities** tab in the Options for Target dialog and deselect the **Update target before Debugging** check box.



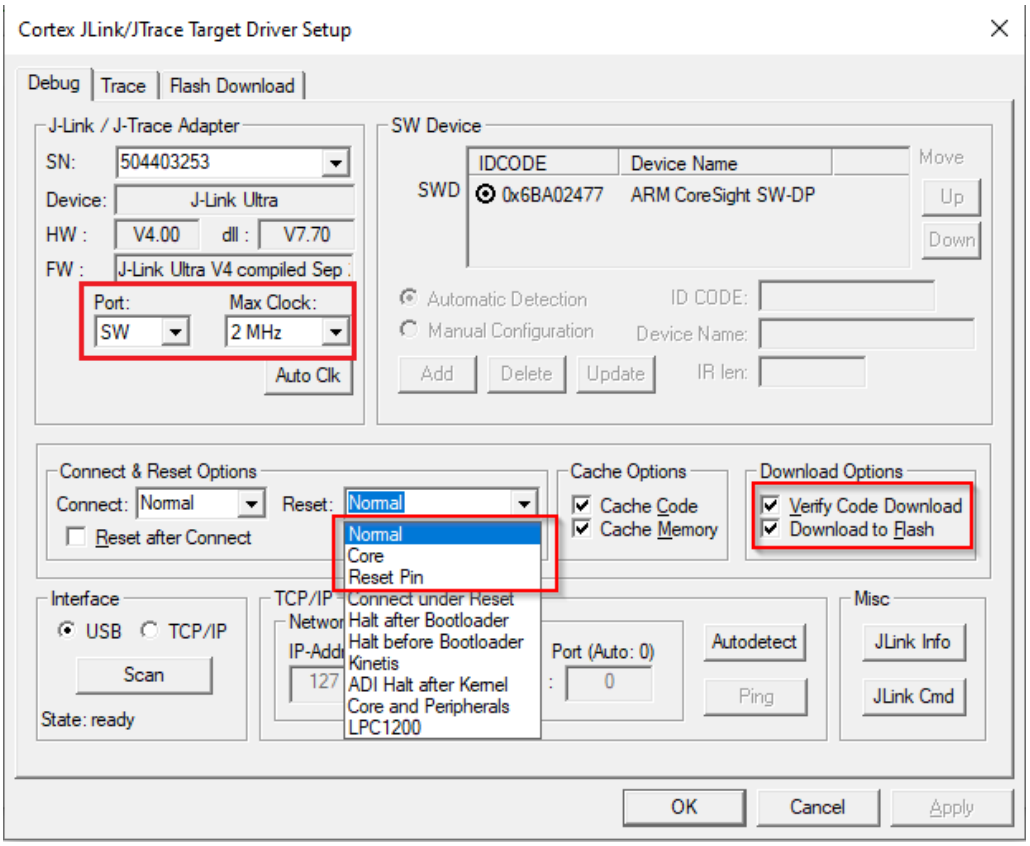
8.4.2 Debug set-up instructions

1. Select the **Debug** tab in the Options for Target dialog, select "J-LINK / J-TRACE Cortex" as the debug adapter, and click **Settings**.



2. On the **Debug** tab in the Target Driver Setup dialog:

- **Port**: select "SW"
- **Max Clock**: select 2 MHz
- **Reset** pull-down: select "Normal," "Core" or "Reset Pin"
- **Download Options**: enable "Verify Code Download" and "Download to Flash"



8.5 Launch multi-core debug session

To launch a multi-core debug session, all your µVision projects must be opened in separate IDE instances.

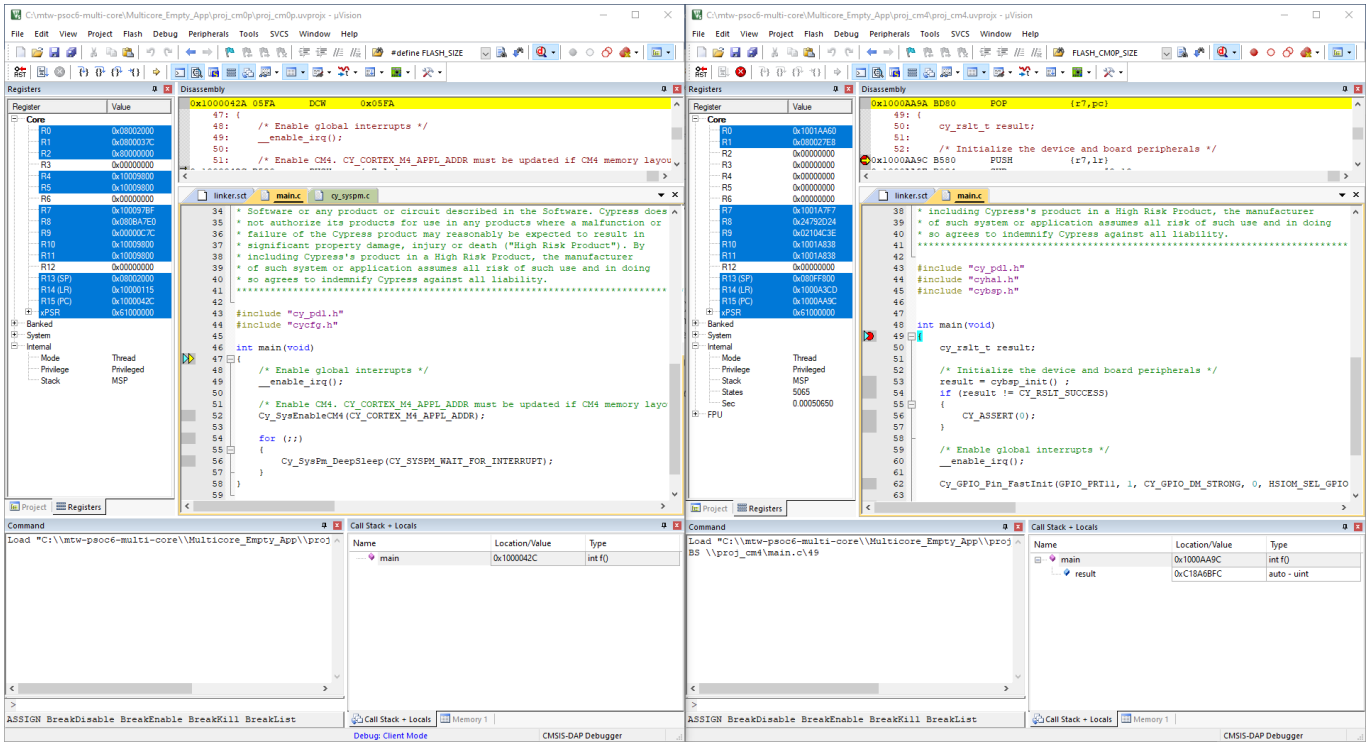
1. Open a µVision IDE session with the project for the CM0+ core and start debugging by pressing **Debug > Start/Stop Debug Session**. This will program all images, reset the target, and halt at the beginning of the CM0+ project `main()`.

2. Repeat the same process for the CM4/CM7 core(s). This will attach the running CM4/CM7 core that will be spinning in the boot code until the CM0+ project starts it.

***Note**: Ensure both projects are built before launching a debug session.*

For dual-core MCUs, the projects will be similar to these images:

8 PSOC™ 6 and XMC7xxx multi-core application



The left side of the screen shows a µVision IDE instance attached to the CM0+ core. The right side shows the CM4 core has not started yet. Once the `Cy_SysEnableCM4()` function on the CM0+ core has been executed, the CM4 will start executing its application. You can step through the code by switching back and forth between the two µVision IDE instances.

Revision history

Revision	Date	Description
**	2023-05-15	New document.
*A	2023-06-02	Removed obsolete instructions for customizing linker scripts.
*B	2024-01-25	Updates for version 3.2.0. Removed Python. Add instructions for CYW20829 devices. Added link for ETM/ITM trace instructions.
*C	2024-09-27	Updates for version 3.3.0.
*D	2024-12-02	Added instructions to configure a PSOC™ Control C3 device.
*E	2024-12-06	Updates for version 3.4.0.
*F	2025-03-25	Updates for version 3.5.0.
*G	2025-04-28	Separated 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-ggj1743010908890

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.