

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:

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



Table of contents

Table of contents

Abou	It this document	, 1
Tabl	e of contents	. 2
1	Download/install software	, 3
1.1	ModusToolbox™ tools package	3
1.2	IAR Embedded Workbench (Windows only)	3
1.3	Python	3
1.4	J-Link	3
2	Getting started	. 4
2.1	Create new application	4
2.2	Export existing application	8
2.3	Open application in IAR Embedded Workbench	8
2.4	Build the application	.9
2.5	PSoC [™] 64 application configuration	10
3	Programming/Debugging	12
3.1	XMC7000 and TRAVEO™ II specific steps	
3.2	To use KitProg3/MiniProg4	
3.3	To use MiniProg4 with PSoC™ 6 single core and PSoC™ 6 256K	13
3.4	To use J-Link	
3.5	Program external memory	
3.6	Erase PSoC™ 6 MCU with external memory enabled	16
4	Multi-core debugging	18
4.1	Supported debugger probes	18
4.2	Create IAR workspace and projects	18
4.3	Configuring IAR projects	18
4.4	Launch multi-core debug session with CMSIS-DAP/I-Jet	24
4.5	Launch multi-core debug session with J-Link	
4.6	Multi-core toolbar and CTI usage (I-Jet and CMSIS-DAP only)	25
5	Patched flashloaders for AIROC™ CYW208xx devices	26
Revi	sion history	27



Download/install software

1 Download/install software

1.1 ModusToolbox[™] tools package

Refer to the instructions in the <u>ModusToolbox[™] tools package installation guide</u> 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 <u>Project Creator</u> <u>user guide</u>.

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 <u>Dashboard user guide</u> 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.

Enter filter text	<u>a</u>	Create from MPN Browse for BSP	Ē₽	CY8CKIT-062S2-43012
Kit Name > PMG BSPs > PSoC™ 4 BSPs > PSoC™ 6 BSPs CY8CEVAL-062S2 CY8CEVAL-062S2-LAI-43439M2 CY8CEVAL-062S2-LAI-4373M2 CY8CEVAL-062S2-LAI-4373M2 CY8CEVAL-062S2-MUR-4373M2 CY8CEVAL-062S2-MUR-4373M2 CY8CEVAL-062S2-MUR-4373EM2 CY8CEVAL-062S2-MUR-4373EM2 CY8CKIT-062S2-HUR-4373EM2 CY8CKIT-062S4 CY8CKIT-062S4 CY8CKIT-064B0S2-4343W CY8CKIT-064S0S2-4343W CY8CKIT-064S0S2-4343W		<none> Sterling-LWB+ (CYW43439KUBG) Sterling-LWB5+ (CYW4373EUBGT) LBEE5KL1YN (CYW43439KUBG) LBEE5PK2AE (CYW4373EUBGT) <none> LBEE59B1LV (CYW43012C0WKWBG) <none> LBEE5KL1DX (CYW4343WKUBG) CYW4343WKUBG LBEE5KL1DX (CYW4343WKUBG)</none></none></none>	~	The CY8CKIT-062S2-43012 PSoC [™] 6S2 Wi-Fi BT Pioneer Kit is a low- cost hardware platform that enables design and debug of PSoC [™] 6 MCUs. It comes with a Murata 1LV Module (CYW43012 Wi-Fi + Bluetooth Combo Chip), industry-leading CAPSENSE [™] for touch buttons and slider, on- board debugger/programmer with KitProg3, microSD card interface, 512-Mb Quad-SPI NOR flash, PDM-PCM microphone interface. Kit Features: • Support of up to 2MB Flash and 1MB SRAM
SP: CV8CKIT-062S2-43012				



2.1.3 Step 3: Select application

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

Select Application - Pr	oject Creator 2.10 -	-		
<u>S</u> ettings <u>H</u> elp				
Application(s) Root Path:	C:/Users/Test/mtw3.1/iar		Browse	
Target IDE:	<none></none>	\sim		
Enter filter text	🖉 Browse for Application 💎 🐲 🕒 🗉 🗈			٦
Template Application > Bluetooth > Community Code Ex. > Connectivity > Getting Started > Graphics > Machine Learning > Manufacturing > Peripherals > Sensing > Voice > Wi-Fi				
Select one or more temp	ate applications to proceed with the new project creation process.			
	< <u>B</u> ack <u>C</u> reate	e	<u>C</u> lose	

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.

Application(s) Root Path:	C:/Users/Test/mtw3.1/iar	Browse
Target IDE:	<none> ~</none>	
	<none></none>	
Enter filter text	Eclipse IDE for ModusToolbox™ Microsoft Visual Studio Code	
Template Application	IAR Embedded Workbench ARM MDK (uVision)	



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.

Template Application	New Application Name	New BSP Name	^
✓ Getting Started			
Dual-CPU Empty PSoC6 App			
Empty App			
Hello World	Hello_World	APP_CY8CKIT-062S2-43012	
Security App			
Switching Power Modes			
> Graphics			
> Machine Learning			
> Manufacturing			
> Peripherals			
> Sensing			
> Voice			
> Wi-Fi			
> Wi-Fi			~

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.

Select Application - Pr	roject Creator 2.1	0						-			×
<u>S</u> ettings <u>H</u> elp											
Application(s) Root Path:	C:/Users/Test/m	ntw3.1/iar							Br	owse.	
Target IDE:	IAR Embedded	Workbench							~		
Enter filter text		ß	Browse for Application 🖓	gin gin	 E	Ŧ					
Template Application		New Application Na	me New BSP Name			^					
 Getting Started Dual-CPU Em Empty App 	pty PSoC6 App										
Hello World Security App Switching Pov Graphics	ver Modes	Hello_World	APP_CY8CKIT-062S2-43012								
 Machine Learning Manufacturing Peripherals 											
> Sensing > Voice											
> Wi-Fi						\mathbf{v}					
Starting application creat Acquiring the application Acquired application sou Loading the application.	n source 'https://	github.com/Infineon/	o_World'. 'mtb-example-hal-hello-world'.								^
						<	<u>B</u> ack	<u>C</u> reate	4	lose	

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

0 error(s), 0 warning(s)	^
Summary:	
Successfully created and exported "Hello_World" application.	¥
	< <u>B</u> ack <u>Create</u> <u>Close</u>



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.

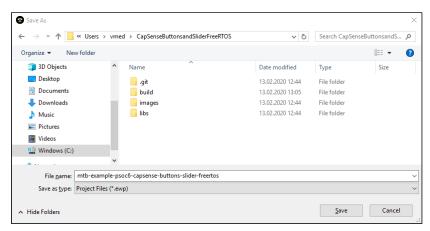
- *Note:* For applications that were created using core-make-3.0 or older, you must use the make ewarm8 command instead.
- Note: This sets the TOOLCHAIN to IAR in the Embedded Workbench configuration files but **not** in the ModusToolbox[™] application's Makefile. Therefore, builds inside IAR Embedded 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**:

Select IAR Project	Connect	tion File		
- > • ↑ 📘	> Volo	odymyr Medvid > CapSenseButtonsandSliderFreeRTOS	✓ ひ Search CapSer	nseButtonsandS 🔎
Organize 🔻 🛛 Ne	ew folder			🖽 🕶 🔳 🔮
💻 This PC	^	Name	Date modified	Туре
3D Objects		git	13.02.2020 12:44	File folder
Desktop		📙 build	13.02.2020 13:05	File folder
Documents		images	13.02.2020 12:44	File folder
Downloads		libs	13.02.2020 12:44	File folder
		settings	13.02.2020 13:09	File folder
Music		mtb-example-psoc6-capsense-buttons-slider-freertos.ipcf	13.02.2020 13:06	IPCF File
Pictures				
Videos				
🎬 Windows (C:)			
🔿 Network	× ·	<		
	File <u>n</u> ar	me: mtb-example-psoc6-capsense-buttons-slider-freertos.ipcf	✓ IAR Project Complexity of the second s	onnection File (*.ij 🗸
			Open	Cancel

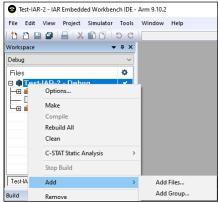
2.4 Build the application

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: <u>https://github.com/IARSystems/project-migration-tools</u>

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:

ategory:					Factory 9	ettings
eneral Options tatic Analysis						
untime Checking						
C/C++ Compiler	#define	Diagnostics	Checksum	Encodings		options
Assembler	Config I	Library Input	Optimizations	Advanced	Output	List
Output Converter Custom Build	Output file	ename:				
Build Actions	CY8CPR	OTO-064S1-SB	Secure Blinky L	ED FreeRTO	S UG.eff	
Linker		-		-		
Debugger	✓ Includ	e debug informat	on in output			
Simulator						
CADI	TrustZon	e import library				
CMSIS DAP	CY8CPR	OTO-064S1-SB_	Secure_Blinky_L	ED_FreeRTO	S_UG_imp	ort_lib.
GDB Server						
I-jet J-Link/J-Trace						
TI Stellaris						
Nu-Link						
PE micro						
ST-LINK						
Third-Party Driver						
TI MSP-FET						
TI XDS						

- 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:

Options for node "CY8CPR	OTO-064S1-SB_Secure_Blinky_LED_FreeRTOS_UG"	×
Calegory: General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Custom Build Actions Linker Debugger Simulator CADI CMSIS DAP GDB Server I-jet J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET	Build Actions Configuration Pre-build command line: Post-build command line: PONENT_CMUP_SECURE -cm0-app-name psoc6_01_cm0p_secure]	
TI XDS	OK Cancel	

- 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.

🕑 Open			×
\leftarrow \rightarrow \checkmark \uparrow \square \Rightarrow This	C → Windows (C:) → UG → CY8CPROTO-064S1-SB_Secure_Blinky_LED_FreeR	TOS_UG → Debug → Exe v Ō	
Organize 🔻 New folder			III 🕶 🔲 😮
 Quick access Desktop Downloads Documents Pictures avto CY8CPROTO-064S1-SI temp test_report My Desktop My Documents My Sync OneDrive - Personal This PC 3D Objects Desktop 	Name CY8CPROTO-06451-SB_Secure_Blinky_LED_FreeRTOS_UG.i CY8CPROTO-06451-SB_Secure_Blinky_LED_FreeRTOS_UG_i CY8CPROTO-06451-SB_Secure_Blinky_LED_FreeRTOS_UG_i em_eeprom.hex	hex 4/10/2023 3:39 PM unsigned.hex 4/10/2023 3:39 PM	1 HI 1 HI 1 HI
CY8CPROTO-00 HEX File	4S1-SB_Secure_Blinky_LED Date modified: 4/10/2023 3:39 PM Date crea Size: 297 KB	ated: 4/10/2023 3:13 PM	
File <u>n</u> ar	e: CY8CPROTO-064S1-SB_Secure_Blinky_LED_FreeRTOS_UG.hex		All Files (*.*)
			<u>O</u> pen Cancel

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 <u>XMC7000/TRAVEO[™] II specific steps</u> 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 <u>KitProg3 User Guide</u> 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:

Options for node "mtb-exa	mple-psoc6-capsense-buttons-slider-freertos"
Category:	Factory Settings
General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter	Setup Download Images Multicore Extra Options Plugins
Custom Build Build Actions Linker	CMSIS DAP v main Simulator CADI
Debugger Simulator CADI CMSIS DAP GDB Server I-jet J-Link/J-Trace TI Stellaris Nu-Link PE micro	CMSIS DAP GDB Server Het J-Link/J-Trace TI Stellaris Nu-Link PE micro ST.LINK Third-Party Driver TI MSP-FET TI XDS Stebugger/Cypress/PSoC6/CY8C6
ST-LINK Third-Party Driver TI MSP-FET TI XDS	OK Cancel

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

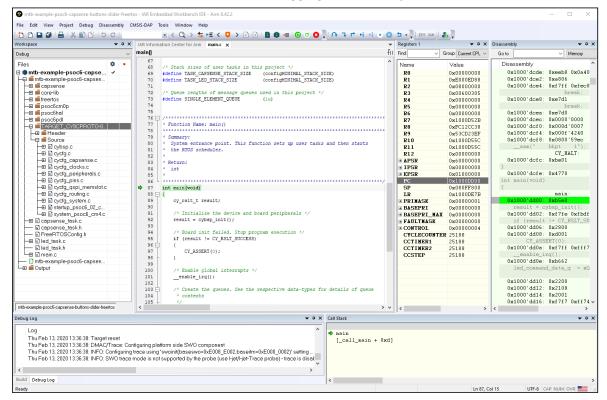
Category:			Factory Settings
General Options Static Analysis Runtime Checking C/C++ Compiler	Setup Interface	Breakpoints	
Assembler Output Converter Custom Build Build Actions	Probe config	Probe configuration file	
Build Actions Linker Debugger	 Explicit 		Select
Simulator CADI CMSIS DAP	Interface O JTAG	Explicit probe configuration Multi-target debug system	
GDB Server I-jet J-Link/J-Trace	⊛ <u>S</u> WD	Iarget number (TAP or Multidrop Target with multiple CPUs CPU number on target	
TI Stellaris Nu-Link	Interface speed	CPU number on target.	
PE micro ST-LINK Third-Party Driver TI MSP-FET	2MHz ~		
TI XDS			
11.05			
			OK

4. Click **OK**.



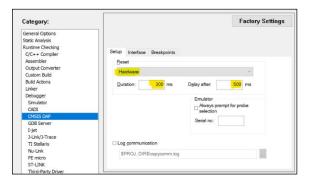
5. Select Project > Download and Debug.

The IAR Embedded 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:

Options for node "mtb-example	e-psoc6-capsense-buttons-slider-freertos" X
Category: General Options Static Analysis Runtime Checking	Factory Settings Setup Download Images Multicore Extra Options Plugins Driver
Build Actions Linker Debugger Simulator CADI CMSIS DAP GOB Server I-jet J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK	J-Link/ofTrace / main Simulator CADI GDB Server Het Het Het Het FindParty Driver T1 Stellans Nu-Link PE micro ST-LINK Tind-Party Driver T1 MSP-FET T1 XDS G\debugger\Cypres\PSoC6\CY8C6
Third-Party Driver TI MSP-FET TI XDS	OK Cancel

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

iereral Options tatic Analysis unitme Checking C/C++ Compiler Assembler Output Converter Output Conver Output Co	ategory:	_	Factory Settings
untime Criseding C/C++ Compiler Sasembler Output Converter Output Converter Out			
C/C+- Compler Assembler Output Converter Cutom bluid Build Actions Linker Debugger Simulator CADI CALI			
Assembler Dutput Converter Dutput Converter Suida Actons Jinker Paddress Smulator CADI OLSB OSB Server I-jet TLStelaris Nu-Link PE micro ST-LINK Strukt Driver		Setup Connection Breakpoints	
Curitom Build O ICP/IP: P address Similator IP address IP address Debugger IP address JTAG scan chain Similator JTAG scan chain CADI O JTAG GBS Server JTAG scan chain with multiple targets Iget IF address Similator JTAG scan chain CADI O JTAG Similator IF address GBS Server IF address Iget If an number: If address If a scan chain with multiple targets If address If a scan chain on Am devices Pinte/Diritor If a scan chain contains non-Am devices Pinte/Diritor If a scan chain with multiple targets Structure If a scan chain with multiple targets If a scan chain with multiple targets If a scan chain with multiple targets Structure If a scan chain with multiple targets If a scan chain with multiple targets		Communication	
Juid Actions IP address Jinker IP address Sebugger IP address Simulator JTAG cocc ddd CADI JTAG JTAG cocc ddd OLTAG JTAG Cocc ddd I'refrace JTAG Cocc ddd JtAG TAP number: JtAG Cocc ddd Server Server I'st Call OLTAG Preceeding bit: OLTAG Preceeding bit: Stallin/J-Trace Log gommunication PE miroo SPROJ_DIR\$'cspycomm.log		● USB: Device 0	
Alla Actors IP address: aaa.bbb.cc.ddd Serial no: Debugger IP address: aaa.bbb.cc.ddd Serial no: Simulator JTAG scan chain CADI JTAG scan chain with multiple targets CADI JTAG scan chain with multiple targets CADI JTAG GDB Server JTAG Jelrikt/JTrace CJTAG Pinikt/JTrace CJTAG Dirikt/JTrace Log communication PE micro SPROJ_DIRS'cspycomm.log		O TCP/IP IP address	
Debugger II P address Deaa doo doc add Senal no: Simulator JTAG scan chain If address CADI JTAG JTAG scan chain CMSIS DAP JTAG TAF number: GDB Server If address Scan chain contains non-Am devices Jtrik/Dirace CITAG Preceeding bits: Nul-Ink Log communication PE micro ST-LINK		<u> </u>	
CADI CASI CMSIS DAP GDB Server Ijst Jurk/J.Yrace Jurk/J.Yrace O_ITAG IJTAG Contain with multiple targets TAP number: 0 Scan chain with multiple targets TAP number: 0 Scan chain contains non-Arm devices Preceeding bits: 0 Log gommunication PF micro ST-LINK Third Party Driver		IP address: aaa.bbb.ccc.ddd Serial no:	
CMSIS DAP GDB Server Tjet IAP rumber: Image: Comparison of the form of	Simulator		
CMSIS DAP TAP number: 0 GSB Server SSWD Scan chain contains non-Arm devices I-jet Status Preceeding bits: 0 Jint/J/Strace Log gommunication Preceeding bits: 0 VLink SPR0J_DIRS\cspycomm.log		O JTAG JTAG □ JTAG scan chain with multiple targets	
L-jet		TAP number: 0	
Jurk/J-Trace cJTAG Preceeding bits: 0 TI Stellaris Display the second sec		Scan chain contains non-Am devic	es
TI Stellaris Nu-Link PE micro SPROJ_DIR\$'cspycomm.log Third-Party Driver		CJTAG Preceeding bits: 0	
PE micro \$PROJ_DIR\$\cspycomm.log \$T+LINK Third Party Driver	TI Stellaris		
ST-LINK SPRUJ_DIRS\cspyconm.log Third-Party Driver		L Log communication	
Third-Party Driver		\$PROJ_DIR\$\cspycomm.log	
	TI MSP-FET		

Note:

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

	Factory Settings
Setup Connection Breakpoints	
Core V	
JTAG/SWD speed Clock setup	
O Auto Initial 1000 kHz CPU clock:	MHz
Eixed 1000 kHz SWO clock: Auto	_
Adaptive 2000	kHz

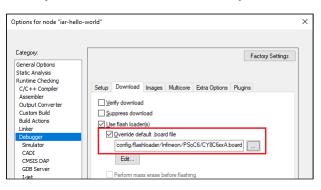
4. Connect a J-Link debug probe to the 10-pin adapter (this needs to be soldered on the prototyping kits), and start debugging.



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</pre>
  <loader>$TOOLKIT DIR$/config/flashloader/Infineon/PSoC6/CY8C6xxA SFLASH.flash</loader>
  <range>CODE 0x16007C00 0x16007FFF</range>
</pass>
              -><!-- SMIF (XIP Region)
   - <pass>
  <!-- <loader>$TOOLKIT_DIR$/config/flashloader/Infineon/PSoC6/CY8C6xxA_SMIF.flash</loader>
  <!-- <range>CODE 0x18000000 0x1FFFFFFF</range> -
   <mark>- </pass></mark>
<!-- Exclude regions
<ignore>CODE 0x08000000 0x080FFFFF</ignore> <!-- Exclude SRAM Region -->
<ignore>CODE 0x16000000 0x160007FF</ignore> <!-- Exclude SFLASH [SFLASH Start - User Data Start]</pre>
<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 0x18000000 0x1FFFFFF</ignore> <!-- Exclude EFuse Region
(flash_board>
 <ignore>CODE 0x90700000 0x907FFFFF</ignore> <!-- Exclude eFuse Region</pre>
```

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>
     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</pre>
<ignore>CODE 0x90500000 0x905FFFFF</ignore> <!-- Exclude Cy Metadata Region</pre>
<ignore>CODE 0x90700000 0x907FFFFF</ignore> <!-- Exclude eFuse Region -->
<!-- <ignore>CODE 0x18000000 0x1FFFFFFF</ignore> --> <!-- Exclude XIP Region -->
 lash board>
```



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

Options for node "iar-hello-v	world"	×
Options for node "iar-hello-v Calegory. Static Analysis Runtime Checking C/C++ Compiler Assembier Output Converter Custon Build Build Actions Linker Deburger Simulator CADI OMSIS DAP GDB Server I jet J-Link()-Trace TI Stellaris Nu-Link PE micro ST-LINK Thrid-Party Driver	Factory Settings Setup Download Yenfy download Suppress download Yue flash loader(s) Yue flash loader(s) Setup Edit Perform mass erase before flashing	×
TI MSP-FET TI XDS	OK Cancel	

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 <u>AN228740</u>.

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

0	Download and Debug	Ctrl+D				
	Debug without Downloadin	g		_	_	
۲	Attach to Running Target					
C	Make & Restart Debugger	Ctrl+R				
C	Restart Debugger	Ctrl+Shift+R		_		
	Download	•		l	Do	ownload active appli
	SFR Setup				Do	ownload file
					Er	ase memory

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

Erase Memory		×
Flash loader	Range	^
C:\Program Files\IAR Systems\Embedded Workbench 9.3\arm/config/flashloader/Infineon/PSoC6/CY8C6xxA	0x16001a00 - 0x16001bff	:
C:\Program Files\IAR Systems\Embedded Workbench 9.3\arm/config/flashloader/Infineon/PSoC6/CY8C6xxA	0x16005a00 - 0x160065ff	-
C: \Program Files\IAR Systems\Embedded Workbench 9.3\arm/config/flashloader/Infineon/PSoC6/CY8C6xxA	0x16007c00 - 0x16007fff	
C:\Program Files\IAR Systems\Embedded Workbench 9.3\arm/config/flashloader/Infineon/PSoC6/CY8C6xxA	0x18000000 - 0x1fffffff	
		*
<	>	
Erase all	Erase Cancel	

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



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

Erase Memory		Х
Elash loader	Range	^
C:\Program Files\IAR Systems\Embedded Workbench 9.3\arm/config/flashloader/Infineon/PSoC6/CY8C6xxA	0x16001a00 - 0x16001bff	F
C:\Program Files\IAR Systems\Embedded Workbench 9.3\arm/config/flashloader/Infineon/PSoC6/CY8C6xxA	0x16005a00 - 0x160065ff	F
C:\Program Files\IAR Systems\Embedded Workbench 9.3\arm/config/flashloader/Infineon/PSoC6/CY8C6xxA	0x16007c00 - 0x16007fff	
C:\Program Files\IAR Systems\Embedded Workbench 9.3\arm/config/flashloader/Infineon/PSoC6/CY8C6xxA	0x18000000 - 0x1fffffff	
		×
<	>	
Erase all	Erase Cancel	

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.

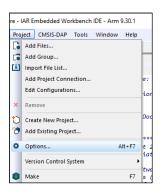
4.3 Configuring IAR projects

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.

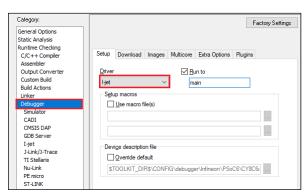
4.3.1 Project configuration for CM4/CM7 (slave) core(s)

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





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



- 3. Enable hex file generation.
 - 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)

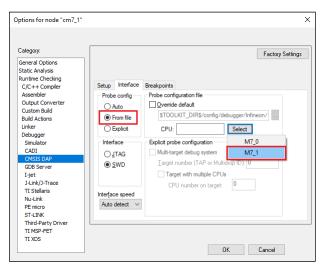
4.3.2 XMC7000/TRAVEO[™] II specific steps

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:



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

Category:	Factory Settings
General Options Static Analysis Runtime Checking C/C++ Compiler	Setup Interface Breakpoints
Assembler	Reset
Output Converter Custom Build	Custom
Build Actions Linker	Duration: 300 ms Delay after: 200 ms
Debugger Simulator CADI	Emulator Always prompt for probe selection
CMSIS DAP GDB Server	Serial no:
I-jet J-Link/J-Trace	
TI Stellaris Nu-Link PE micro	Log communication \$PROJ_DIR\$\cspycomm.log
ST-LINK	

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.

ategory:						Factory 9	Settings
eneral Options tatic Analysis untime Checking	#define	Dia	anostics	Checksum	Encodinas	Eutro (Options
C/C++ Compiler					_		
Assembler	Config	Library	Input	Optimizations	Advanced	Output	List
Output Converter	Linke	Linker configuration file					
Custom Build		verride de	f au dt				
Build Actions							
Linker	5	<pre>\$PROJ_DIR\$\\bsps\TARGET_APP_KIT_XMC72_EVK\CC</pre>					
Debugger							
Simulator		<u>E</u> dit					
CADI							
CMSIS DAP	<u>C</u> onfigu	ration file	symbol de	finitions: (one pe	r line)		
GDB Server	COB	_CORE_CM7_0=1					
I-iet	_0011						

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.



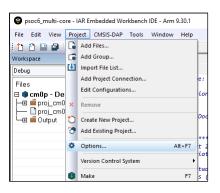
Note:

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.

4.3.3 Project configuration for CM0+ (master) core

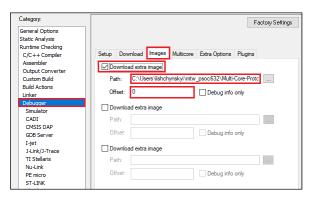
1. Select the CM0+ project and go to **Project > Options**:



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

Category:	Factory Setting
Seneral Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Output Converter Custom Build Build Actions	Setup Download Images Multicore Extra Options Plugins Driver Jet Setup macros
Linker Debugger Simulator CADI CMSIS DAP GDB Server	Use marco file(s)
I-jet J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK	Devige description file Qvenide default STOOLKIT_DIRS\CONFIG\debugger\Infineon\PSoC6\CY8C6)

- 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.

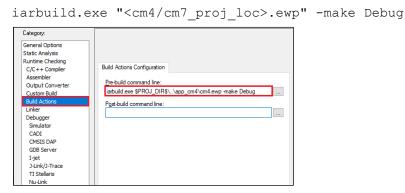




If you provide an **OUT** file instead of a **HEX** file, the IAR IDE will fail to halt at the beginning of main () due to the main function present in both the CM0+ and CM4/CM7 **OUT** files.

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

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



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.

Category:	Factory Settings
General Options Static Analysis Runtime Checking C/C++ Compiler Assembler	Output
Output Converter Custom Build Build Actions Linker Debugger Simulator CADI CMSIS DAP GDB Server I-jet J-tink/J-Trace TI Stelleris Nu-Link PE micro ST-LINK	<u>Generate additional output</u> Output format: Intel Extended hex Qutput file Qutput file Oygenide default (m0.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.

4.3.4 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"?>

IAR Embedded Workbench for ModusToolbox™ user guide



Multi-core debugging

```
<sessionSetup>
    <partner>
        <name>cm0</name>
        <workspace>C:\Users\mtw-multi-core\Multicore App\multi-
core workspace.eww</workspace>
        <project>cm0</project>
        <config>Debug</config>
        <numberOfCores>1</numberOfCores>
        <attachToRunningTarget>false</attachToRunningTarget>
    </partner>
    <partner>
        <name>cm7_0</name>
        <workspace>C:\Users\mtw-multi-core\Multicore_App\multi-
core_workspace.eww</workspace>
        <project>cm7 0</project>
        <config>Debug</config>
        <numberOfCores>1</numberOfCores>
        <attachToRunningTarget>true</attachToRunningTarget>
    </partner>
    <partner>
        <name>cm7 1</name>
        <workspace>C:\Users\mtw-multi-core\Multicore_App\multi-
core workspace.eww</workspace>
        <project>cm7 1</project>
        <config>Debug</config>
        <numberOfCores>1</numberOfCores>
        <attachToRunningTarget>true</attachToRunningTarget>
    </partner>
</sessionSetup>
```

- 2. Configure multi-core debugging for the CM0+ project.
 - a. Go to **Project > Options -> Debugger**.
 - b. Switch to the **Multicore** tab.
 - c. Select the **Advanced** radio button and specify a path to the session configuration file in the **Session configuration** field.
 - d. Click OK.

ptions for node "cm0"		
Category:	Factory Settings	
General Options Static Analysis Runtime Checking		
C/C++ Compiler Assembler	Setup Download Images Multicore Extra Options Plugins	
Assembler Output Converter	Symmetric multicore	
Custom Build	Number of cores: 1	
Build Actions		
Linker	Asymmetric multicore	
Debugger	O <u>D</u> isabled	
Simulator	○ Simple	
CADI CMSIS DAP		
GDB Server	Partner workspace:	
I-jet	Partner project:	
J-Link/J-Trace	Partner configuration:	
TI Stellaris		
Nu-Link	Attach partner to running target Partner cores 1	
PE micro	Advanced	
ST-LINK	Session configuration: C:\mtw-xmc7000-multi-core\session xml	
Third-Party Driver		
TI MSP-FET		
TI XDS		

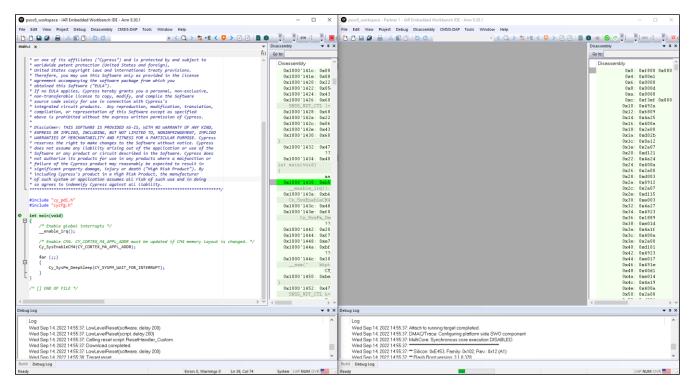


3. Save the workspace.

4.4 Launch multi-core debug session with CMSIS-DAP/I-Jet

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

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

4.5 Launch multi-core debug session with J-Link

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

- IAR 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 <u>Multi-core toolbar and CTI usage</u>.

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 CMO+ 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 CMO+ project's main().
- 3. Switch to the other IDE instances and select: **Project > Attach to Running Target**.



4.6 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:

🕴 ETM SWO 🕴 🔳 O 👻 🔲 1 👻 🚅	[백국] (6 국 (* 여 제 🕑 😐 - 📜 畾 📜
up_psoc6_02_cm0plus.s	Run/Step/Stop affect all cores
	Run/Step/Stop affect current core only



Patched flashloaders for AIROC[™] CYW208xx devices

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 history

Revision	Date	Description
**	2023-05-15	New document.
*A	2023-06-02	Removed obsolete instructions for customizing linker scripts.

Trademarks

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

Edition 2023-06-02 Published by

Infineon Technologies AG 81726 Munich, Germany

© 2023 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document? Email: erratum@infineon.com

Document reference 002-37598 Rev. *A

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.