

Visual Studio Code for ModusToolbox[™] user guide

ModusToolbox tools package version 3.4.0

About this document

Scope and purpose

This document provides information and instructions for using Visual Studio Code (VS Code) with ModusToolbox™ software.

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 A newer version of this document may be available on the web here.

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 > NewIndicates that a cascading sub-menu opens when you select a menu item.				

Reference documents

Refer to the following documents for more information as needed:

- tools package installation guide Provides information and instructions about installing the tools package on Windows, Linux, and macOS.
- tools package user guide Provides information about all the tools included with ModusToolbox[™] tools package.
- Debugging in Visual Studio Code
- GitHub Marus /cortex-debug: Visual Studio Code extension for enhancing debug capabilities for Cortex-M Microcontrollers



Table of contents

Table of contents

	About this document
	Table of contents
1	Download/install software
1.1	ModusToolbox™ tools package3
1.2	VS Code
1.3	J-Link
2	Getting Started
2.1	Create new application
2.2	Export existing application
2.3	Open workspace in VS Code7
3	Add/modify application code 10
4	Using ModusToolbox $^{ imes}$ tools $\dots \dots \dots$
4.1	ModusToolbox™ Assistant extension11
4.2	Command line
5	Build the Application
6	Program/debug common
6.1	Program
6.2	Debug
6.3	Changing programming interface SWD/JTAG16
6.4	Update debugger serial number
6.5	Add Live Watch
7	Program/debug using KitProg3/MiniProg4 19
7.1	Connect the Kit
7.2	KitProg Firmware Loader
8	Program/debug using J-Link
8.1	Configure J-Link programmer/debugger settings
8.2	Connect the Kit
9	Multi-core debugging
9.1	Configurations
9.2	Launch the configuration
	Revision history
	Disclaimer



1 Download/install software

1 Download/install software

1.1 ModusToolbox[™] tools package

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

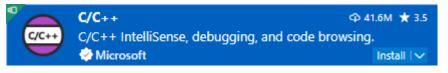
1.2 VS Code

The ModusToolbox[™] tools package includes various tools to create and manage applications, but it does not include VS Code. If you do not already have VS Code installed on your computer, you can download it from the website:

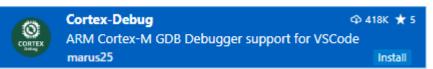
https://code.visualstudio.com/

After opening an application in VS Code, it will recommend several extensions. The C/C++ tools and Cortex-Debug extensions are required for build and debug. Other extensions such as Arm[®] Assembly, ModusToolbox[™] Assistant, and clangd improve the development and debug experience.

• C/C++ tools



Cortex-Debug



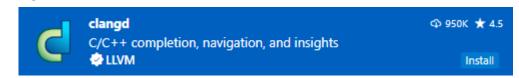
• Arm[®] Assembly

Arm Assembly $33K \star 5$ Arm assembly syntax support for Visual Studio Code dan-c-underwood Install

• ModusToolbox[™] Assistant

<u>Ŷ</u>	ModusToolbox Assistant	Ф 141
ౖం	An Assistant for ModusToolbox	
\sim	C & T Software	Install

clangd



1.3 J-Link

For J-Link debugging, download and install J-Link software: https://www.segger.com/downloads/J-Link



2 Getting Started

This section covers the ways to get started using VS Code with ModusToolbox[™] software

- Create new application
- Export existing application
- Open workspace in VS Code

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

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

Open the Project Creator tool as applicable for your operating system. You can launch it from the ModusToolbox™ Dashboard or the VS Code ModusToolbox™ Assistant extension.

Settings Help Source Template	
Enter filter text Browse for BSP 📄 🖪	
AIROC™ Bluetooth® BSPs Connectivity > AIROC™ Connectivity BSPs PMG BSPs > PSoC™ 4 BSPs PSoC™ 4 BSPs > PSoC™ 4 BSPs PSoC™ 4 BSPs > TRAVEO™ BSPs Virteless Charging BSPs > XMC™ BSPs VMC™ BSPs	
Finished download of file 'https://gitlab.intra.infineon.com/repo-staging/mtb-mw-manifest/raw/v2.X/mtb-mw-manifest.fv2.X/ml' Finished download of file 'https://gitlab.intra.infineon.com/repo-staging/mtb-mw-manifest/raw/v2.X/mtb-mw-dependencies-manifest.xml' Finished download of file 'https://gitlab.intra.infineon.com/repo-staging/mtb-bt-mw-manifest/raw/v2.X/mtb-tmw-manifest.rawl' Finished download of file 'https://gitlab.intra.infineon.com/repo-staging/mtb-bt-mw-manifest/raw/v2.X/mtb-th-mw-manifest.rawl' Finished download of file 'https://gitlab.intra.infineon.com/repo-staging/mtb-bt-mw-manifest/raw/v2.X/mtb-bt-mw-manifest.rawl' Finished download of file 'https://gitlab.intra.infineon.com/repo-staging/mtb-bt-mw-manifest/raw/v2.X/mtb-bt-mw-manifest.rawl' Finished download of file 'https://gitlab.intra.infineon.com/repo-staging/mtb-wifi-mw-manifest/raw/v2.X/mtb-bt-mw-manifest.rawl' Finished download of file 'https://gitlab.intra.infineon.com/repo-staging/mtb-wifi-mw-manifest/raw/v2.X/mtb-wifi-mw-manifest.rawl' Finished download of file 'https://gitlab.intra.infineon.com/repo-staging/mtb-wifi-mw-manifest/raw/v2.X/mtb-wifi-mw-manifest.rawl' Finished download of file 'https://gitlab.intra.infineon.com/repo-staging/mtb-wifi-mw-manifest/raw/v2.X/mtb-wifi-mw-manifest.rd2.xml' Finished download of file 'https://gitlab.intra.infineon.com/repo-staging/mtb-wifi-mw-manifest/raw/v2.X/mtb-wifi-mw-manifest.rd2.xml' Finished download of file 'https://gitlab.intra.infineon.com/repo-staging/mtb-wifi-mw-manifest/raw/v2.X/mtb-wifi-mw-manifest.rd2.xml' Finished loading the manifest data (3164 ms) Loading the device db	~
<u>N</u> ext >	<u>C</u> lose

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 list of boards available in this version will reflect the platforms available for development.

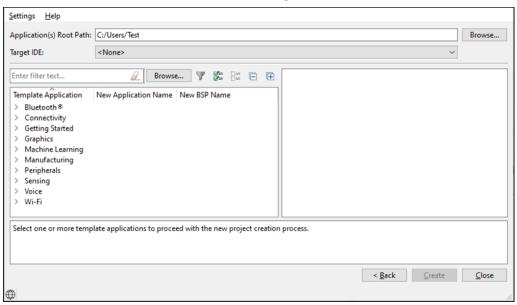


nter filter text	Browse	. ∈ ⊕	CY8CKIT-06252-43012
Git Name AIRCC ⁺⁺ Genetic by BSPs > AIRCC ⁺⁺⁺ connectivity BSPs > PMG BSPs > PSoC ⁺⁺⁺ 6 BSPs > PSoC ⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺⁺	MCU/SOC/SIP CY8C624A8ZI-S2D44 CY8C624A8ZI-S2D44 CY8C624A8ZI-S2D44 CY8C624A8ZI-S2D44 CY8C624A8ZI-S2D44 CY8C624A8ZI-S2D44 CY8C624A1C0-S4092 CY8C6247BZI-D54 CY8C624A1C0-S4092 CY8C624A1C0-S4092 CY8C624A1C0-S4092 CY8C624A1C0-S4092 CY8C624A1C0-S4092 CY8C624A1C0-S4092 CY8C624A1C0-S4092 CY8C624A1C0-S4092 CY8C624A1C0-S4092 CY8C624A1N-S2D43 CY8C624AFN-S2D43 CY8C624AFN-S2D43 CY8C624AFN-S2D43 CY8C624AFN-S2D43 CY8C624AFN-S2D43 CY8C624AFN-S2D43 CY8C624AFN-S2D43 CY8C624AFN-S2D43 CY8C624AFN-S2D43 CY8C624AFN-S2D43 CY8C624AFD0-D52 CY8C624FD0-D02 CY8C624FD0-D02 CY8C624FD0-D02 CY8C634FB2-BLD53	<none> <none> <none></none></none></none>	 The CY8CKIT-06252-43012 PSoC[™] 652 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 11V Module (CYW43012 Wi-Fi + Biuetoth Combo Chip), industry-leading CAPSENE[™] for touch buttons and slider, on-board debugger/programmer with KIPPog3, microSD card interface, 512-Mb Quad-SPI NOR flash, PDM-PCM microphone interface. Kit Features: Support of up to 2MB Flash and 1MB SRAM Dedicated SDHC to interface with WICED wireless devices. Delivers dual-cores, with a 150-MHz Arm Cortex-M4 as the primary application processor and a 100-MHz Arm Cortex-W4 as the primary application processor and a 100-MHz Arm Cortex-W4 as the primary application processor and a 100-MHz Arm Cortex-104 as the scondary processor for low-power operations. Supports Full-Speed USB, capacitive-sensing with CAPSENSE, a PDM-PCM digital microphone interface, a Quad-SPI interface, 13 serial communication blocks, 7 programmable analog blocks, and 56 programmable digital blocks. Kit Contents: Soport Fill Spee. At Micro-B cable Quick Stant Guide Four jumper wires (4 inches each) Two jumper wires (5 inches each)
mmary: P: CY8CKIT-06252-43012			
ess "Next" to select application.			

2.1.3 Step 3: Select application

To select an application:

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



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

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



3. Pull down the **Target IDE** menu and select Microsoft Visual Studio Code.

C:/Users/Test	Browse
<none> ~</none>	
<none></none>	
ARM MDK (uVision)	
	<none> <none> Eclipse IDE for ModusToolbox™ Microsoft Visual Studio Code IAR Embedded Workbench</none></none>

4. Under the **Template Application** column, expand **Getting Started** and select **Hello World** from the list. This example exercises the PSOC[™] 6 MCU to blink an LED.

Ter	nplate Application	New Application Name
>	Bluetooth ®	
¥	Getting Started	
	Dual-CPU Empty PSoC6 App	
	Empty App	
	Hello World	Hello_World
	Switching Power Modes	
>	Graphics	
>	Machine Learning	
>	Manufacturing	
>	Peripherals	
>	Sensing	
>	Wi-Fi	

Note: The actual application names available might vary.

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

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

2.1.4 Step 4: Create application

To create the application:

1. Click Create.

The tool displays various messages.



									_	~
Select Application - P	roject Creator							-		×
<u>S</u> ettings <u>H</u> elp										
Application(s) Root Path:	C:/Users/Test,								Brows	se
Target IDE:	Microsoft Visua	al Studio Code						\sim		
Search			Browse	7 8:		This code example demonstrates simple UA			"Hello	
Template Application		New Application	Name			World" message on a terminal and blinks a	n LED using a Timer reso	urce.		
> Bluetooth®		New Application	INDITIC			For more details, see the README on GitHu	<u>b</u> .			
✓ Getting Started										
Dual-CPU Em	pty PSoC6 App									
Empty App										
Switching Por	wer Modes	Hello_World								
> Graphics										
> Machine Learning										
> Manufacturing										
> Peripherals										
> Sensing > Wi-Fi										
2 WITT										
nttpsgtnub.com_cyp	TRANSFERTING FUTURE									
					cypressno-c	eckout C:/Users/follettcj/.modustoolbox/cac	he/git/			^
httpsgithub.com_cyp						-				
Starting: git -C C:/Users/ httpsgithub.com_cyp						neckout C:/Users/follettcj/.modustoolbox/cac	he/git/			
Cloning into 'TARGET_C'			And par-carrie							~
							< Back Cre	eate	Clos	se
\oplus										

When the process completes, a message states that the application was created.

0 error(s), 0 warning(s)		^
Summary:		
Successfully created and exported "Hello_World" application.		~
	< Back Create	Close

- 2. Click **Close** to exit the Project Creator tool.
 - **Note**: If you opened the Project Creator tool using the VS Code ModusToolbox Assistant extension, the tool will close automatically upon successful completion.

2.2 Export existing application

If you have a ModusToolbox[™] application that was created for another IDE or the command line, you can export that application to be used in VS Code. Open a terminal window in the application directory and run the command make vscode.

2.3 Open workspace in VS Code

In VS Code, select **File > Open Workspace from File**, navigate to the location of the application that was just created, select the workspace file, and click **Open**.



→ ✓ ↑	> Test > mtw3.1 > vscode > Hello_World >	• 5	Search Hello_World
rganize 🔻 New folder			III 🔻 🔟 🤇
💿 My Desktop	↑ Name	Date modified	Type Siz
	.vscode	1/23/2023 12:50 PM	File folder
My Documents	bsps	1/23/2023 12:49 PM	File folder
💿 My Sync	build	1/23/2023 12:50 PM	File folder
This PC	deps	1/23/2023 12:49 PM	File folder
3D Objects	images	1/23/2023 12:49 PM	File folder
-	libs	1/23/2023 12:49 PM	File folder
Desktop	E Hello_World.code-workspace	1/23/2023 12:50 PM	Code Workspace
Documents			
Downloads			
👌 Music			
Pictures			
📑 Videos			
🚔 Windows (C:)			
A	v <		
	Hello World.code-workspace	∽ Code	e Workspace (*.code-works 🗸

Depending on your settings in VS Code, you may see a message about trusting the authors. If so, click **Yes, I** trust the authors.

•	Do you trust the authors of the files in this workspace?						
	Code provides features that may automatically execute files in this workspace.						
	If you don't trust the authors of these files, we recommend to continue in restricted mode as the files may be malicious. See our docs to learn more.						
	C:\Users\Test\mtw3.1\vscode\Hello_World\Hello_World (Workspace)						
	Yes, I trust the authorsNo, I don't trust the authorsTrust workspace and enable all featuresBrowse workspace in restricted mode						

VS Code opens with the Hello_World workspace in the EXPLORER view.



<u>Eile Edit Selection View Go R</u> un <u>T</u> erminal <u>H</u> elp	\leftarrow \rightarrow \square Hello_World (Worksp	ace)
EXPLORER ····		
✓ HELLO_WORLD (WORKSPACE)		
V Hello_World > .vscode		
> bsps > build		
> deps > images		
) libs		
ry inus ≣ .cyignore		
 ♦ .gitignore 		
Hello_World.code-workspace		
<pre>% LICENSE</pre>		
C main.c		
Makefile		
≣ openocd.tcl		
README.md		
\sim mtb_shared		
> cat1cm0p	Show All Commands	Ctrl + Shift + P
> cmsis		
> core-lib	Go to File	Ctrl + P
> core-make	Find in Files	Ctri + Shift + F
> mtb-hal-cat1		
> mtb-pdl-cat1	Start Debugging	F5
> recipe-make-cat1a	Toggle Terminal	Chil 1
> retarget-io	loggie lerminal	cui +
> OUTLINE > TIMELINE		
0 A 0		8



3 Add/modify application code

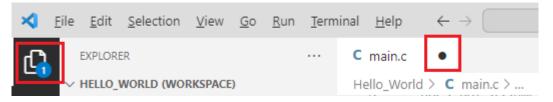
3 Add/modify application code

Code example applications work as they are, and there is no need to add or modify code in order to build or program them. However, if you want to update and change the application to do something else, open the appropriate file in the code editor.

Double-click the *main.c* file to open it.

× .	<u>Eile E</u> dit <u>S</u> election <u>V</u> iew <u>G</u> o <u>R</u> un <u>T</u> erm	$\texttt{Help} \leftarrow \rightarrow \qquad \qquad$	\Box \times
Ð	EXPLORER ····	C main.c ×	∕ ⊕ ⊞ …
	〜 HELLO_WORLD (WORKSPACE) 📭 巳言 ひ 🗗	Hello_World > C main.c >	
り で い の の の の の の の の の の の の の の の の の の	 > Hello_World > vscode > bsps > build > deps > images > libs ≅ .cyignore 	 Software or any product or circuit described in the Software. Cypress does * not authorize its products for use in any products where a malfunction or * failure of the Cypress product may reasonably be expected to result in * significant property damage, injury or death ("High Risk Product"). By * including Cypress's product in a High Risk Product, the manufacturer * of such system or application assumes all risk of such use and in doing * so agrees to indemnify Cypress against all liability. ************************************	
	{} Hello_World.code-workspace	43 #include "cybsp.h"	HALL BOARD AND AND AND AND AND AND AND AND AND AN
	🕺 LICENSE	44 #include "cy_retarget_io.h" 45	A NUMBER OF CONTRACTOR OF CONTRACTON OF CONTRACTOR OF CONTRA
8	C mainc M Makefile E openocd.tdl ③ README.mdl ✓ mtb_shared > cat1cmOp > cmsis > core-lib > core-make > mtb-hal-cat1 > mtb-pdl-cat1a > retarget-io	<pre>46 47 48 4 Macros 48 4 Macros 49 45 50 50 51 52 53 54 55 55 55 55 55 55 55 55 55 55 55 55</pre>	
5	> OUTLINE	62 63	
	> TIMELINE	64 /************************************	
⊗ 0 .	▲ 0	Ln 1, Col 1 Spaces: 4 UTF-8 LF C I	мтв 🔊 🕻

As you type into the file, a dot will appear in the file's tab to indicate changes were made. The file icon will also indicate that there are unsaved changes.



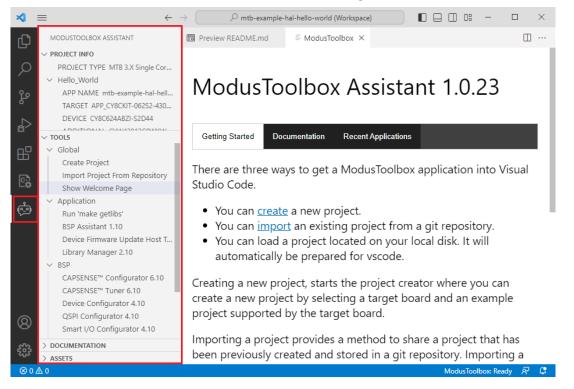


4 Using ModusToolbox[™] tools

4 Using ModusToolbox[™] tools

4.1 ModusToolbox[™] Assistant extension

The easiest way to open various ModusToolbox[™] tools with VS Code is by installing the ModusToolbox[™] Assistant extension, which provides access to tools, configurators, and documentation.



4.2 Command line

Alternatively, you can open various ModusToolbox[™] tools using make commands in the terminal. Select **Terminal > New Terminal**, then select the main project folder for your application (in this case, Hello_World):

View	Select current working directory for new terminal]]
	Hello_World C:\Users\Test\mtw3.1\vscode	
KSPAC	mtb_shared C:\Users\Test\mtw3.1\vscode	

Note: On Windows, use the modus-shell (Cygwin) terminal.

This section covers a few of the tools you might open more frequently. For a complete list of the tools available, refer to the tools package user guide.

4.2.1 Library Manager

To add, remove, or modify libraries, open the Library Manager using the following command:

make library-manager

Refer to the Library Manager user guide for details about that tool.



4 Using ModusToolbox[™] tools

4.2.2 BSP Assistant

To create or modify a BSP, open the BSP Assistant using the following command:

make bsp-assistant

Refer to the BSP Assistant user guide for details about that tool.

4.2.3 Device Configrator

To view peripherals, pins, clocks, etc., open the Device Configurator using the following command:

make device-configurator

The Device Configurator provides access to the BSP resources and settings. Each enabled resource contains one or more links to the related API documentation. There are also buttons to open other configurators for CAPSENSE™, QSPI, Smart I/O, etc. For more information, refer to the Device Configurator user guide, which is also available by selecting **View Help** from the tool's **Help** menu.

C:/Users/Test/mt	tw3.1/vscode/Hello_World/bsps/T#	ARGET_APP_CY8CKIT-06	2S2-43012/config/design.m	nodus* - Device Configurator 4.10	- 🗆 ×
CY8C624ABZI-S2D4		WBG		Real Time Clock (RTC) - Parameters	ē ×
Peripherals Pins	Analog-Routing System	Peripheral-Clocks	DMA	Enter filter text	/ U 🖻 🕀
Enter filter text			/ 🖂 🖶 🖌 🗎 🛍	Name	Value ^
Resource		Name(s)	Personality ^	✓ Overview	
TCPW	/M[0] 32-bit Counter 1	tcpwm_0_cnt_1		⑦ Configuration Help	Open RTC Documentation
TCPW	/M[0] 32-bit Counter 2	tcpwm 0 cnt 2		✓ General	
_	/MI01 32-bit Counter 3	tcpwm 0 cnt 3		⑦ Date Format	MM/DD/YYYY ~
	/M[0] 32-bit Counter 4	tcpwm_0_cnt_4		② Enable Daylight Savings (D	IST)
	/M[0] 32-bit Counter 5	tcpwm 0 cnt 5	-	✓ Time and Date	
	/M[0] 32-bit Counter 6	tcpwm 0 cnt 6	-	? Seconds	0
			-	(?) Minutes	0
	/M[0] 32-bit Counter 7	tcpwm_0_cnt_7		⑦ Hours Format	24H ~
 System 	er, and PWM (TCPWM) 1			? Hour	12
	Sense, etc.) 0	CYBSP CSD	CSD-3.0	⑦ Day of the Month	1
		Icd 0 drive 0		? Month	January 🗸
	unter Watchdog Timer (MCWDT) 0			? Year	0
	unter Watchdog Timer (MCWDT) 0 unter Watchdog Timer (MCWDT) 1		_	⑦ Day of the Week	SATURDAY
Real Time			Real Time Clock-3.0 🗸	✓ Advanced	
	Clock (RIC)	srss_0_rtc_0	Keal Time Clock-3.0 V	Store Config in Flash	×
<			,	Real Time Clock (RTC) - Parameters	Code Preview
Notice List					8 ×
🙁 0 Errors 👍 0) Warnings 📋 0 Tasks 🚺 4	Infos			
Fix Description				Location	^
WCO is ready	enabled. Chip startup will be slowe y. See the device datasheet for WC ting it in main() for faster chip star	O startup timing. If WCO			~
Ready					

Note: The Device Configurator cannot be used to open Library Configurators, such as Bluetooth[®].



5 Build the Application

5 Build the Application

Building the application is not specifically required, because building will be performed as part of the programming and debugging process. However, if you are running VS Code without any hardware attached you may wish to build your application to ensure all the code is correct.

Select Terminal > Run Task. Then select Build Hello_World.

EXPLORER	New Terminal	Ctrl+Shift+` Ctrl+Shift+5	Build Hello_World	recently used -🛱 😫 🗡
✓ HELLO_WORLD (WORKSPACE)		Ctri+Shirt+5	Clean Hello_World	
> Hello_World	Run Task		Rebuild Hello_World	configure
> mtb_shared	Run Build Task	Ctrl+Shift+B	Tool: Library Manager Hello_World	
	Run Active File		En grunt	contribute
			El gulp	
	Run Selected Text		🖻 jake	
			En npm	
			E typescript	
			E cppbuild	
			Show All Tasks	
	Configure Tasks			
	Configure Default Bu	ild Task		

Build information will display in the Terminal.

<pre>tfile cy_ctdac.c tfile cy_dma.c tfile cy_dmac.c tfile cy_efuse.c tfile cy_efuse_v3. tfile cy_ephy.c tfile cy_ethif.c tfile cy_evtgen.c tfile cy_flash.c</pre>	c									
<pre>t file cy_dmac.c t file cy_efuse.c t file cy_efuse_v3. t file cy_ephy.c t file cy_ethif.c t file cy_evtgen.c</pre>	.c									
t file cy_efuse.c t file cy_efuse_v3. t file cy_ephy.c t file cy_ethif.c t file cy_evtgen.c	с									
t file cy_efuse_v3. t file cy_ephy.c t file cy_ethif.c t file cy_evtgen.c	c									
t file cy_ephy.c t file cy_ethif.c t file cy_evtgen.c	c									
t file cy_ethif.c t file cy_evtgen.c										
t file cy_evtgen.c										
t file cy_flash.c										
t file cy_flash_sro	om.c									
t file cy_flash_v2.	c									
t file cy_gpio.c										
t file cy_i2s.c										
t file cy_ipc_bt.c										
t file cy_ipc_drv.c	:									
t file cy_ipc_pipe.	c									
t file cy_ipc_sema.	c									
t file cy_keyscan.c	:									
t	file cy_ipc_sema.	file cy_ipc_sema.c file cy_keyscan.c	: file cy_ipc_sema.c : file cy_keyscan.c							

The build should complete successfully with messages similar to the following:



5 Build the Application

= Build complete =

Calculating memory consumption: CY8C624ABZI-S2D44 GCC_ARM

Section Name	Address	Size
<pre>.cy_m0p_image .text .ARM.exidx .copy.table .zero.table .data .cy_sharedmem .noinit .bss .heap</pre>	0x10000000 0x1000ca90 0x1000ca90 0x1000ca98 0x1000cab0 0x080022e0 0x08002918 0x08002918 0x08002920 0x08002a00 0x08003000	6224 43664 8 24 8 1592 8 224 1532 1030144
Total Internal Flash (Av Total Internal Flash (Ut	· · · · · · · · · · · · · · · · · · ·	2097152 53504

* Terminal will be reused by tasks, press any key to close it.



6 Program/debug common

The VS Code GUI shows these launch configurations by default:

- **Launch**: This builds the associated project, programs project-specific output file, and then starts a Cortex-M4 debugging session.
- Attach: This starts a Cortex-M4 debugging session attaching to a running PSOC[™] 6 target without programming or reset.
- Erase Device: This erases all internal memories.
- Erase All: If present, erases all internal and external memories.
- **Program**: This builds the associated project, programs project-specific output file, and then runs the program.

6.1 Program

Open the main menu, select **Terminal > Run Task**. On the selection menu, select the "program" task.

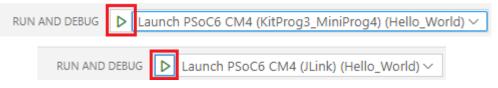
Select the task to run	
Build Hello_World	configured
Clean Hello_World	
Erase Device Hello_World	
Program Hello_World	⊕ ⇔
Rebuild Hello_World	
Tool: Library Manager Hello_World	

If needed, VS Code builds the application and messages display in the Terminal. If the build is successful, device programming starts immediately. If there are build errors, then error messages will indicate as such. When programming completes successfully, the LED will start blinking.



6.2 Debug

Select the **Run and Debug** icon in the VS Code Activity Bar, select the **Launch PSOC6 CM4** (KitProg3_Miniprog4) or Launch PSOC6 CM4 (JLink) Launch Configuration, and click Start Debugging icon or press F5.



If needed, VS Code builds the application and messages display in the Console. If the build is successful, VS Code switches to debug mode automatically. If there are build errors, then error messages will indicate as such.



🗙 Ei	ile <u>E</u> dit <u>S</u> election <u>V</u> iew <u>G</u> o <u>R</u> un … \leftarrow \rightarrow	Q	-example-hal-hello-world (Workspace)			- 🗆 ×
Ch	RUN AND DEBUG D Launch PSoC6 CM4 (KitProg3_MiniProg4) (Hello_World	i)∨ ⊜ ≋ d			é	⊳∽ @ □ …
	✓ VARIABLES		Hello_World > bsps > TARGET_APP_CY8CKIT-		Cybsp_init(void)	
	<pre> Local result: <optimized out=""> </optimized></pre>		<pre>90 cy_rslt_t cybsp_init(void)</pre>			
مع	> Global		91 { 92 // Setup hardware manage	ger to track resource u	sage then initialize all	ROBARD/TRAINING
8	> Static		93 // configuration		isage chen inicialize all	Minner-Hanner-
	> Registers		94 #if defined(CY_USING_HA 95 cy_rslt_t result = cyha	AL)		
_			96			ENGINE AND
₿			97 if (CY_RSLT_SUCCESS == 98 {	result)		NAMES AND A DESCRIPTION OF A DESCRIPTION
E.	> WATCH		99 result = cyhal_sysp	om_init();		A LEWIS CONTRACT
C)	V CALL STACK	Paused on step	100 }			Real Property and
	cybsp_init@0x100028d6 bsps/TARGET_APP_CY8CKIT-062S2-4301		PROBLEMS OUTPUT DEBUG CONSOLE	TERMINAL MEMORY XRTO	s C/C++ Configurat ∨ ≣	A m ^ ×
	main@0x10002984	main.c (110)	[5/8/2023, 2:49:33 PM] " <u>c:\Users\</u> compile_commands.json" could not folder 'HELLO_WORLD' will be used	follettcj\mtw3.1\5-8-v be parsed. 'includePat	scode\Hello_World/build/	
	✓ BREAKPOINTS					
	CORTEX PERIPHERALS BACKUP @ 0x40270000					
	> BACKUP @ 0x40270000 > CPUSS @ 0x40200000					
	> CSD0 @ 0x40360000					
	> DMAC @ 0x402a0000					
	> DW0 @ 0x40280000 > DW1 @ 0x40290000					
	> EFUSE @ 0x402c0000					
	FAULT @ 0x40210000 CORTEX REGISTERS					
	r0 134228280					
	r1 0					
_	r2 134217728					
8						
8	r3 0					
8 8	r3 0 r4 268488968			In 95 Col 1	1 Snares: 4 ITTE-8 IF () (мтв ₽ ∩
8	r3 0 r4 268488968			Ln 95, Col	1 Spaces: 4 UTF-8 LF () C	: мтв 🔊 🗘
⊗ ∰ ⊗ • ▲	r3 0 r4 266489668 c54489668 c54489666 CM4 (KtiProg3_MiniProg4) (Hello_World)	₽ m	example-hal-hello-world (Workspace)	Ln 95, Col		: мтв <i>Я</i> Д
⊗ \$ \$ \$ \$	r3 0 r4 266488968 20 ∲ Launch PSoC6 CM4 (KitProg3_MiniProg4) (Hello_Workd) Eile Edit Selection View ⊆o ···· ← →		example-hal-hello-world (Workspace)	Ln 95, Col `	∎ ■ □ 0% -	
⊗ ∰ ⊗ • ▲	r3 0 r4 266489968 20 $$ Launch PSoC6 CM4 (KitProg3_MiniProg4) (Hello_World) Eile Edit Selection View $@o \cdots & \leftarrow \rightarrow$ RUN AND DEBUG \triangleright Launch PSoC6 CM4 (L'Link) (He \checkmark $$ $$ \cdots	c 🗄 ტ	~□ C 1 ¥ ♡ <		- :::	
⊗ \$ \$ \$ \$ \$	r3 0 r4 266488968 20 $$ Launch PSoC6 CM4 (KiProg3_MiniProg0 (Hello_Work)) Eile Edit Selection Yiew Go ···· ← → RUN AND DEBUG ▷ Launch PSoC6 CM4 (Llink) (He \checkmark ③ ··· \checkmark VARIABLES	c ແປ	> ? * 1 D ~ > bsps > TARGET_APP_CY8CKIT-062S2-	-43012 > C cybsp.c > 🏵	Cybsp_init(void) □ □ □ □ □	
⊗ ∰ ⊗•▲	r3 0 r4 268483968 2.0 $\xrightarrow{1}$ Launch PSoC6 CM4 (KtProg3_MiniProg4) (Helio_World) Elle Edit Selection View Go ··· ← → RUN AND DEBUG ▷ Launch PSoC6 CM4 (Link) (He \checkmark ③ ··· \checkmark VARIABLES \checkmark Local	C ස ල Hello_Wor 87 /	> ? * * D . ~		Cybsp_init(void) □ □ □ □ □	· · · · · · · · · · · · · · · · · · ·
⊗ \$ • • • • • •	r3 0 r4 266488968 20 $$ Launch PSoC6 CM4 (KiProg3_MiniProg0 (Hello_Work)) Eile Edit Selection Yiew Go ···· ← → RUN AND DEBUG ▷ Launch PSoC6 CM4 (Llink) (He \checkmark ③ ··· \checkmark VARIABLES	С <u><u></u> <u></u> <u></u></u>	> C + C D - > bsps > TARGET_APP_CY8CKIT-06252- cybsp_init	-43012 > C cybsp.c > 🏵	Cybsp_init(void) □ □ □ □ □	· · · · · · · · · · · · · · · · · · ·
⊗ ∰ ⊗ •	r3 0 r4 266488968 20 ∲ Launch PSoc6 CM4 (KtProg3_MiniProg4) (Helo_Workd) Elle Edit Selection Yiew Go ··· ← → RUN AND DEBUG ▷ Launch PSoC6 CM4 (Link) (He ✓ ④ ··· ∨ VARIABLES ∨ Local result: <optimized out=""></optimized>	C :: U Hello_Wor 87 / 88 / 89 / 90 c	> C + C D - > bsps > TARGET_APP_CY8CKIT-06252- cybsp_init	-43012 〉 C cybsp.c 〉 �	Cybsp_init(void) □ □ □ □ □	
8 \$ € € \$ \$ \$ \$	r3 0 r4 266489968 20	C :: U Hello_Wor 87 / 88 / 89 / 90 c 91 (<pre>> ? * ? D _ ~ > bsps > TARGET_APP_CY8CKIT-06252- cybsp_init rslt_t cybsp_init(void)</pre>	43012 > C cybsp.c > Q	DB − s> cybsp_init(void)	
⊗ \$ • • • • • •	r3 0 r4 266489968 20	C :: U Hello_Wor 87 / 88 / 89 / 90 c	> C + C D	43012 > C cybsp.c > Q	DB − s> cybsp_init(void)	
	r3 0 r4 268489968 20	C :: () Hello_Wor 87 / 88 / 90 c 91 7 92 93 94	<pre>> ? * ? D > bsps > TARGET_APP_CY8CKIT-06252- cybsp_init rslt_t cybsp_init(void) // Setup hardware manager to // configuration #if defined(CY_USING_HAL)</pre>	43012 > C cybspc > ↔ track resource usag	DB − s> cybsp_init(void)	
8 \$ € C C C C S S S S S S S S S S S S S S S	r3 0 r4 268489968 2.0 ∂ Launch PSoC6 CM4 (KtProg3_MiniProg4) (Helio_World) Elle Edit Selection View Go ··· ← → RUN AND DEBUG ▷ Launch PSoC6 CM4 (Link) (He ✓ ③ ··· ✓ VARIABLES ✓ Local result: <optimized out=""> > Global > Static > Registers > WATCH</optimized>	C :: () Hello_Wor 87 / 88 / 90 c 91 (92 93 94 D 95	<pre>> ? * ? D > bsps > TARGET_APP_CY8CKIT-06252- cybsp_init rslt_t cybsp_init(void) // Setup hardware manager to // configuration</pre>	43012 > C cybspc > ↔ track resource usag	DB − s> cybsp_init(void)	
	r3 0 r4 268489968 20 ∲ Launch PSoC6 CM4 (KtProg3_MinProg4) (Helio_Workd) Elle Edit Selection View Go ··· ← → RUN AND DEBUG ▷ Launch PSoC6 CM4 (Link) (He ✓ ④ ··· ∨ VARIABLES ∨ Local result: <optimized out=""> > Global > Static > Result: <optimized out=""> > Global > Static > WATCH ∨ CALL STACK Paused on step</optimized></optimized>	C :: () Hello_Wor 87 / 88 / 90 c 91 7 92 93 94	<pre>> ? * ? D U ~ > bsps > TARGET_APP_CV8CKIT-06252- cybsp_init rslt_t cybsp_init(void) // Setup hardware manager to // configuration #if defined(CV_USING_HAL) cy_rslt_t result = cyhal_hwm</pre>	43012 > C cybsp.c > ↔ track resource usage gr_init();	DB − s> cybsp_init(void)	
8 € € € € €	r3 0 r4 266489968 20	C :: U Hello_Wor 87 / 88 / 90 c 91 (92 93 94 D 95 96	<pre>> ? * ? D > bsps > TARGET_APP_CY8CKIT-06252- cybsp_init rslt_t cybsp_init(void) // Setup hardware manager to // configuration #if defined(CY_USING_HAL)</pre>	43012 > C cybsp.c > ↔ track resource usage gr_init();	DB − s> cybsp_init(void)	
	r3 0 r4 266489968 20	C :: U Hello_Wor 87 / 88 / 89 / 90 c 91 92 93 94 0 95 94 D 95 96 97 98 99	<pre>> ? * ? D > bsps > TARGET_APP_CY8CKIT-06252- cybsp_init rslt_t cybsp_init(void) // Setup hardware manager to // configuration #if defined(CY_USING_HAL) cy_rslt_t result = cyhal_hwm if (CY_RSLT_SUCCESS == resul</pre>	<pre>43012 > C cybsp.c > ③ track resource usager_init(); t)</pre>	DB − s> cybsp_init(void)	
	r3 0 r4 266489968 20	C :: U Hello.Wor 87 / 89 / 90 c 91 0 93 94 D 95 96 97 95 96 97 98 99 180	<pre>> ? \$? D _ ~ > bsps > TARGET_APP_CV8CKIT-06252- cybsp_init rslt_t cybsp_init(void) // Setup hardware manager to // configuration #if defined(CY_USING_HAL) cy_rslt_t result = cyhal_hum if (CY_RSLT_SUCCESS == resul {</pre>	<pre>43012 > C cybsp.c > ③ track resource usager_init(); t)</pre>	DB − s> cybsp_init(void)	
	r3 0 r4 266489968 20	C :: U Hello_Wor 87 / 88 / 89 / 90 c 91 92 93 94 0 95 94 D 95 96 97 98 99	<pre>> ? \$? D _ ~ > bsps > TARGET_APP_CV8CKIT-06252- cybsp_init rslt_t cybsp_init(void) // Setup hardware manager to // configuration #if defined(CY_USING_HAL) cy_rslt_t result = cyhal_hum if (CY_RSLT_SUCCESS == resul {</pre>	<pre>43012 > C cybsp.c > ③ track resource usager_init(); t)</pre>	DB − s> cybsp_init(void)	
	r3 0 r4 268489968 20 \$2 Launch PSoC6 CM4 (KtProg3_MiniProg4) (Helio_Workd) Elle Edit Selection View Go ··· ← → RUN AND DEBUG ▷ Launch PSoC6 CM4 (Link) (He ✓ ③ ··· ∨ VARIABLES ∨ Local result: <optimized out=""> > Global > Static > WatCH ∨ CALL STACK Passed on step cybsp_init@0x10002384 main.c 110</optimized>	C :: U Hello_Wor 87 / 88 / 90 / 91 92 93 94 0 95 96 97 98 99 98 99 9100 101	<pre>> ? \$? D U ~ > bsps > TARGET_APP_CV8CKIT-06252- cybsp_init rslt_t cybsp_init(void) // Setup hardware manager to // configuration #if defined(CV_USING_HAL) cy_rslt_t result = cyhal_hum if (CY_RSLT_SUCCESS == resul { result = cyhal_syspm_ini }</pre>	<pre>43012 > C cybsp.c > ♀ track resource usag gr_init(); t) t();</pre>	DB − s> cybsp_init(void)	
	r3 0 r4 268489968 20	C # U Hello.Wor 87 / 88 / 90 c 91 92 93 94 94 95 96 97 98 99 100 101 102 103	<pre>> ? * ? D > bsps > TARGET_APP_CV8CKIT-06252- cybsp_init rslt_t cybsp_init(void) // Setup hardware manager to // configuration #if defined(CV_USING_HAL) cy_rslt_t result = cyhal_hwm if (CY_RSLT_SUCCESS == resul { result = cyhal_syspm_ini } #ifdef CY_CFG_PWR_VDDA_MV if (CY_RSLT_SUCCESS == resul</pre>	<pre>43012 > C cybsp.c > ③ track resource usag gr_init(); t) t(); t)</pre>	Cybsp_init(void)	
	r3 0 r4 268489968 20 \$2 Launch PSoC6 CM4 (KtProg3_LMinProg4) (Helio_Workd) Elle Edit Selection View Go ··· ← → RUN AND DEBUG ▷ Launch PSoC6 CM4 (Link) (He ✓ ③ ··· ∨ VARIABLES ∨ Local result: <optimized out=""> > Global > Static > Registers > WATCH < CALL STACK Paused on step cybsp_init@0x10002846 bsps/TARGET_APP_CV8CKT-062 main@0x10002984 main.c 110 > BREAKPOINTS < CORTEX PERIPHERALS > BACKUP @ 0x40270000 > CPUSS @ 0x40200000</optimized>	C # U Hello.Wor 87 // 88 // 90 cc 91 7 92 93 94 94 95 96 97 98 99 99 100 101 102 103 PROBLEMS	<pre>> ? * ? D > bsps > TARGET_APP_CV8CKIT-06252- cybsp_init rslt_t cybsp_init(void) // Setup hardware manager to // configuration #if defined(CV_USING_HAL) cy_rslt_t result = cyhal_hwm if (CV_RSLT_SUCCESS == resul { result = cyhal_syspm_ini } #ifdef CY_CFG_PWR_VDDA_MV if (CY_RSLT_SUCCESS == resul OUTPUT DEBUG CONSOLE TERMINU</pre>	<pre>43012 > C cybsp.c > ③ track resource usag gr_init(); t) t(); t)</pre>	DB − s> cybsp_init(void)	
	r3 0 r4 268489968 20	C # 0 Hello, Work 87 / 88 / 99 c 91 99 c 92 93 92 93 92 95 96 97 98 99 97 98 99 90 100 101 102 103 100 100 100 100 100 100 100 100 100	<pre>> ? * ? D > bsps > TARGET_APP_CV8CKIT-06252- cybsp_init rslt_t cybsp_init(void) // Setup hardware manager to // configuration #if defined(CV_USING_HAL) cy_rslt_t result = cyhal_hwm if (CY_RSLT_SUCCESS == resul { result = cyhal_syspm_ini } #ifdef CY_CFG_PWR_VDDA_MV if (CY_RSLT_SUCCESS == resul</pre>	<pre>43012 > C cybspc > ◊ track resource usag gr_init(); t) t(); t) al MEMORY XRTOS</pre>	Cybsp_init(void)	
	r3 0 r4 268489968 20	C # U Hello, Wor 87 // 88 // 99 0 91 92 93 94 94 95 96 97 98 99 99 100 101 102 103 PROBLEMS A prog Ar you	<pre>> ? * ? D > bsps > TARGET_APP_CV8CKIT-06252- cybsp_init rslt_t cybsp_init(void) // Setup hardware manager to // configuration #if defined(CV_USING_HAL) cy_rslt_t result = cyhal_hwm if (CV_RSLT_SUCCESS == resul { result = cyhal_syspm_ini } #ifdef CY_CFG_PWR_VDDA_MV if (CY_RSLT_SUCCESS == resul OUTPUT DEBUG CONSOLE TERMINU m is being debugged already. sure you want to change the f [answered Y; input not from</pre>	<pre>43012 > C cybsp.c > ③ track resource usag gr_init(); t) t(); t) aL MEMORY XRIOS iile?</pre>	Cybsp_init(void)	
	r3 0 r4 268489968 20	C # U Hello, Wor 87 // 88 // 99 0 91 92 93 94 94 95 96 97 98 99 99 100 101 102 103 PROBLEMS A prog Ar you	<pre>> ? \$? D U ~ > bsps > TARGET_APP_CVBCKIT-06252- cybsp_init rslt_t cybsp_init(void) // Setup hardware manager to // configuration #if defined(CV_USING_HAL) cy_rslt_t result = cyhal_hwm if (CY_RSLT_SUCCESS == resul { result = cyhal_syspm_ini } #ifdef CY_CFG_PMR_VDDA_NV if (CY_RSLT_SUCCESS == resul if (CY_RSLT_SUCCESS == resul cyrsult = cyhal_syspm_ini } UUTPUT DEBUG CONSOLE TERMINU m is being debugged already. sure you want to change the f</pre>	<pre>43012 > C cybsp.c > ③ track resource usag gr_init(); t) t(); t) aL MEMORY XRIOS iile?</pre>	Cybsp_init(void)	
	r3 0 r4 268489968 20 \$2 Launch PSoC6 CM4 (KtProg3_MiniProg4) (Helio_Workd) Elle Edit Selection Yiew Go ··· ← → RUN AND DEBUG ▷ Launch PSoC6 CM4 (Link) (He ♥ ③ ··· ∨ VARIABLS ∨ Local result; <optimized out=""> > Global > Static > Registers > WATCH ∨ CALL STACK Paused on step cybsp_init@0x10002846 bsps/TARGET_APP_CY8CKIT-062 main@0x10002784 main.c [1]0 > BREAMPOINTS × CORTEX PERIPHERALS > BACKUP @ 0x40220000 > CDUSS @ 0x40220000 > CDWC @ 0x40220000 > DMAC @ 0x40220000 > DMAC @ 0x40220000 > DWAC @ 0x4022000 > DWAC @ 0x4022000 > DWAC @ 0x4022000 > DWAC @ 0x402200 > DWAC @ 0x4020 > DWAC @ 0x40020 > D</optimized>	C # 0 Hello.Wor 87 / 88 / 99 0 91 93 94 94 95 96 97 98 99 99 180 181 182 183 PROBLEMS A prog (y or (2 Res	<pre>> ? \$? D U ~ > bsps > TARGET_APP_CV8CKIT-06252- cybsp_init rslt_t cybsp_init(void) // configuration #if defined(CV_USING_HAL) cy_rslt_t result = cyhal_hum if (CV_RSLT_SUCCESS == resul { result = cyhal_syspm_ini } #ifdef CY_CFG_PMR_VDDA_NV if (CV_RSLT_SUCCESS == resul if (CV_RSLT_SUCCESS == resul cymput = cyhal_syspm_ini } UUTPUT DEBUG CONSOLE TERMINV m is being debugged already. sure you want to change the ff [answered Y; input not from ting target</pre>	<pre>43012 > C cybsp.c > ③ track resource usage gr_init(); t) t(); t) AL MEMORY XRTOS ile? terminal]</pre>	Cybsp_init(void)	
	r3 0 r4 268489968 20 \$2 Launch PSoC6 CM4 (KtProg3_LMinProg4) (Helio_Workd) Elle Edit Selection View Go ··· ← → RUN AND DEBUG > Launch PSoC6 CM4 (Link) (He < ③ ··· < VARIABLES < Local result: <optimized out=""> > Global > static > Registers > WATCH < CALL STACK Peased on step cybsp_init@0x10002846 bsps/TARGET_APP_CV8CKT-062 main@0x10002984 main.c (110) > BREAKPOINTS < CORTEX PERIPHERALS > BACKUP @ 0x40280000 > CPUSS @ 0x40280000 > CONTEX REGISTERS r0 134228280</optimized>	C # 0 Hello.Wor 87 / 88 / 99 0 91 93 94 94 95 96 97 98 99 99 180 181 182 183 PROBLEMS A prog (y or (2 Res	<pre>> ? * ? D > bsps > TARGET_APP_CV8CKIT-06252- cybsp_init rslt_t cybsp_init(void) // Setup hardware manager to // configuration #if defined(CV_USING_HAL) cy_rslt_t result = cyhal_hwm if (CV_RSLT_SUCCESS == resul { result = cyhal_syspm_ini } #ifdef CY_CFG_PWR_VDDA_MV if (CY_RSLT_SUCCESS == resul OUTPUT DEBUG CONSOLE TERMINU m is being debugged already. sure you want to change the f [answered Y; input not from</pre>	<pre>43012 > C cybsp.c > ③ track resource usage gr_init(); t) t(); t) AL MEMORY XRTOS ile? terminal]</pre>	Cybsp_init(void)	
	r3 0 r4 268489968 20 \$2 Launch PSoC6 CM4 (KtProg3_MiniProg4) (Helio_Workd) Elle Edit Selection Yiew Go ··· ← → RUN AND DEBUG ▷ Launch PSoC6 CM4 (Link) (He ♥ ③ ··· ∨ VARIABLS ∨ Local result; <optimized out=""> > Global > Static > Registers > WATCH ∨ CALL STACK Paused on step cybsp_init@0x10002846 bsps/TARGET_APP_CY8CKIT-062 main@0x10002784 main.c [1]0 > BREAMPOINTS × CORTEX PERIPHERALS > BACKUP @ 0x40220000 > CDUSS @ 0x40220000 > CDWC @ 0x40220000 > DMAC @ 0x40220000 > DMAC @ 0x40220000 > DWAC @ 0x4022000 > DWAC @ 0x4022000 > DWAC @ 0x4022000 > DWAC @ 0x402200 > DWAC @ 0x4020 > DWAC @ 0x40020 > D</optimized>	C # U Hello.Wor 87 / 88 / 99 0 91 9 92 9 94 94 94 95 96 97 98 99 98 99 180 181 182 183 PROBLEMS A prog Are you (y or (2 Res Tempor	<pre>> ? \$? D U ~ > bsps > TARGET_APP_CV8CKIT-06252- cybsp_init rslt_t cybsp_init(void) // configuration #if defined(CV_USING_HAL) cy_rslt_t result = cyhal_hum if (CV_RSLT_SUCCESS == resul { result = cyhal_syspm_ini } #ifdef CY_CFG_PMR_VDDA_NV if (CV_RSLT_SUCCESS == resul if (CV_RSLT_SUCCESS == resul cymput = cyhal_syspm_ini } UUTPUT DEBUG CONSOLE TERMINV m is being debugged already. sure you want to change the ff [answered Y; input not from ting target</pre>	<pre>43012 > C cybsp.c > ③ track resource usage gr_init(); t) t(); t) AL MEMORY XRTOS ile? terminal]</pre>	Cybsp_init(void)	

Changing programming interface SWD/JTAG

To change the target interface, update the application's *bsp.mk* file by adding a make variable as shown (possible values are 'swd' and 'jtag').

MTB_PROBE_INTERFACE=swd

Then, regenerate launch configurations:

6.4 Update debugger serial number

If there are two or more debugger probes connected to your computer, the first detected probe will be used by default. There should not be more than one probe with the same serial number. Use this method if you want to use only one specific device. Use OS-specific tools to determine the serial number of connected USB devices.

6.3



Update application's *bsp.mk* file by adding variable below with the serial number specified, and regenerate launch configurations:

MTB_PROBE_SERIAL=0B0B0F9701047400

6.5 Add Live Watch

While debugging an application in VS Code, it is possible to add a Live Watch variable. This topic provides an example using the Hello World application.

- 1. Open the application's *launch.json* file, and locate the launch configuration. In this case "Launch PSOC6 CM4 (KitProg3_MiniProg4)".
- **2.** Scroll to the end of the configuration and add the following:

```
"liveWatch": {
    "enabled": true,
    }
```

3. Open the *main.c* file and a declare global variable:

```
static uint8_t count = 0;
```

4. Add the code to increment the variable every time the LED is blinking:

```
count=count+1;
```

```
/* Check if timer elapsed (interrupt fired) and toggle the LED */
if (timer_interrupt_flag)
/* Clear the flag */
timer_interrupt_flag = false;
/* Invert the USER LED state */
cyhal_gpio_toggle(pin: CYBSP_USER_LED);
count=count+1;
}
```

5. Launch a serial terminal such as PuTTY to monitor the output.

Note: This step may only be required for applications that have output, such as Hello World.

6. Start the debugger. When it stops at main(), add the count variable to the **CORTEX LIVE WATCH** section and press the [**Enter**] key.



File Edit Selection View count		
RUN AND DEBUG D Laur Enter Live Watch Expression (Press	'Enter' to confirm or 'Escape' to cancel)	□ …
> VARIABLES	Hello_World > C main.c > 🏵 main	
> WATCH > CALL STACK > BREAKPOINTS	 87 * 'Enter' key was pressed and stops/restarts LED blinking. 88 * 89 * Parameters: 	
CORTEX LIVE WATCH	90 * none 91 * 92 * Return:	Economistra T
Hint: Use & Enable "livewatch" in your launch,json to e	93 * int 94 *	 Anno Maria

- 7. Check that the variable was added to the list and click on **Run/Continue** to proceed with debugging.
- 8. Observe that the variable's value increments as the code is running.

× -	File Edit Selection View \cdots \leftarrow \rightarrow	: 🔊 II 🤃 🐇 ↑ Ϧ 🗖 ∽ rkspac
Ð	RUN AND DEBUG 🕨 Launch PSoC6 CM4 (KitPr 🗸 🕲 …	C main.c 1 × 🗊 Preview READ
	> VARIABLES	Hello_World > C main.c > 😚 main
Q	> WATCH	87 * 'Enter' key was pre:
	> CALL STACK	88 *
00	> BREAKPOINTS	89 * Parameters:
sو م		90 * none
		91 *
N	count: <mark>6 '\006'</mark>	92 * Return:
a		93 * int
		Q/ *



7 Program/debug using KitProg3/MiniProg4

Most PSOC[™]-based kits use KitProg3/MiniProg4 as the default programmer/debugger, so there is nothing to configure for them.

7.1 Connect the Kit

Follow the instructions provided with the kit to connect it to the computer with the USB cable.

7.2 KitProg Firmware Loader

The PSOC[™] MCU kits include on-board programmer/debug firmware, called KitProg. The CY8CPROTO-062-4343W kit has KitProg3 by default. However, some older kits come with KitProg2 firmware installed, which does not work with the ModusToolbox[™] software and you must update them to KitProg3. KitProg3 provides the CMSIS-DAP (Bulk) protocol by default, which is up to ~2.5 times faster than the CMSIS-DAP (HID) protocol. Both modes can be used via OpenOCD.

ModusToolbox[™] software includes a command-line tool "fw-loader" to update kits and switch the KitProg firmware from KitProg2 to KitProg3, and back. The following is the default installation directory of the tool:

<install_path>\ModusToolbox\tools_<version>\fw-loader\bin\

Use the fw-loader tool to update the KitProg firmware as needed. KitProg2 does not work with the ModusToolbox[™] software. Likewise, if you update to KitProg3, PSOC[™] Creator won't work with the kits until you restore KitProg2.

Note: On a Linux machine, you must run the udev_rules\install_rules.sh script before the first run of the fw-loader.

For more details, refer to the KitProg3 user guide. The fw-loader tool also provides a readme text file in the fw-loader installation directory.

7.2.1 Supplying power with KitProg3_MiniProg4

If using the KitProg3 connector on a kit, power is generally supplied by the host PC. When using a MiniProg4, power is not supplied via the MiniProg4 by default. It is expected that the target MCU will be powered externally. However, the MiniProg4 does provide the ability to supply power to the target MCU.

Note: Verify the voltage range supported by the target MCU, since it can be damaged by supplying unsupported voltage. Make sure that your MCU is not powered externally before supplying power via the KitProg3_MiniProg4 launch configuration. This supply is limited to approximately 200 mA and is protected against excess current draw. You can select 1.8 V, 2.5 V, 3.3 V, or 5 V.

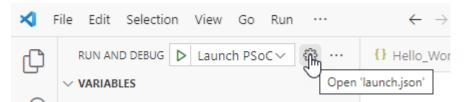
7.2.1.1 Turning power supply on

Debug session

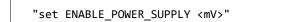
To turn power supply on during a debug session, edit the Launch configurations:

1. Open the **Run and Debug** view, select the launch configuration to be modified, and click the gear icon that opens the selected launch configurations in *launch.json* file.





2. Look for the openOCDPreConfigLaunchCommands property. If it is not present, add it. Update the property to include the following value:



Where <mv> defines target voltage in millivolts. For example:

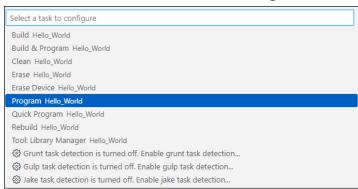
{} Hello_World.code-wo	space () launch.json •
Hello_World > .vscode >	() launch.json > [] configurations > { } 1 > [] openOCDPreConfigLaunchCommands
99	"\${workspaceFolder}",
100	"\${config:modustoolbox.toolsPath}/openocd/scripts/"
101	,
102	openOCDPreConfigLaunchCommands": [
103	"set ENABLE_POWER_SUPPLY 3300",
104	"set ENABLE_ACQUIRE 1"
105	,
106	configFiles": [
107	"openocd.tcl"
108	,

3. Save the changes to the file.

Programing or erasing

To turn power supply on when programing or erasing, edit the Task configurations:

1. On the main menu, select **Terminal > Configure Task...**.



- 2. On the dialog, select the task to be modified, which opens the *tasks.json* file to that task.
- **3.** Update the task to add the following value to the args property:

_MTB_RECIPE__OPENOCD_INTERFACE=\"source [find interface/kitprog3.cfg]; set ENABLE_POWER_SUPPLY <mV>";\



Where <mv> defines target voltage in millivolts. For example:

```
{
    "label": "Program",
            "type": "process",
    "command": "bash",
    "args": [
        "--norc",
        "-c",
        "make -j8 program _MTB_RECIPE_OPENOCD_INTERFACE=\"source [find interface/
kitprog3.cfg]; set ENABLE_POWER_SUPPLY 3300";\ --output-sync"
    ],
    "windows": {
        "command": "${config:modustoolbox.toolsPath}/modus-shell/bin/bash.exe",
        "args": [
            "--norc",
            "-c",
            "export PATH=/bin:/usr/bin:$PATH ; ${config:modustoolbox.toolsPath}/modus-
shell/bin/make.exe -j8 program _MTB_RECIPE_OPENOCD_INTERFACE=\"source [find interface/
kitprog3.cfg]; set ENABLE POWER SUPPLY 3300";\ --output-sync"
        1
    },
    "problemMatcher": "$gcc",
    "group": {
        "kind": "build"
    }
},
```

4. Save the changes to the file.

7.2.2 Power cycle programming mode with KitProg3_MiniProg4

By default, Launch Configurations use Reset mode to program the device. However, Reset mode is not available in all situations (for example, if the XRES pin is not available on the part's package). In these cases, Launch Configurations use an alternative reset with software. However, using the software reset type is not sufficient in cases in which access to the device's DAP is restricted (such as when set by security settings).

If there is no XRES pin available and DAP access is restricted, the only way to reset a part is to use Power Cycle mode. Follow these instructions to add commands to the launch configuration and switch to Power Cycle mode.

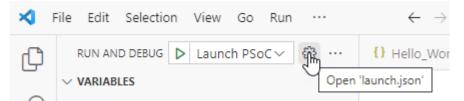
Note: Verify the voltage range supported by the target MCU, since it can be damaged by supplying unsupported voltage. Make sure that your MCU is not powered externally before supplying power via the KitProg3_MiniProg4.

Debug session

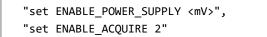
To enable power cycle during a debug session, edit the Launch configurations:

1. Open the **Run and Debug** view, select the launch configuration to be modified, and click the gear icon that opens the selected launch configuration in the *launch.json* file.





- 2. Look for the openOCDPreConfigLaunchCommands property. If it is not present, add it. Update the property to include the following value:
 - For PSOC[™] 6:



• For PSOC[™] 4:

```
"set ENABLE_POWER_SUPPLY <mV>",
"set PSOC4_USE_ACQUIRE 2"
```

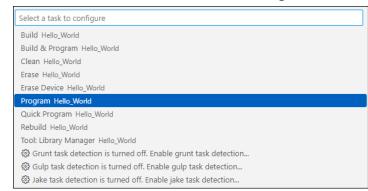
Where <mV> defines target voltage in millivolts. For example:

3. Save the changes to the file.

Programing or erasing

To enable power cycle when programing or erasing, edit the Task configurations:

1. On the main menu, select **Terminal > Configure Task...**.



2. On the dialog, select the task to be modified, which opens the *tasks.json* file to that task.



- **3.** Update the task to add the following value to the args property:
 - For PSOC[™] 6:

_MTB_RECIPE__OPENOCD_INTERFACE=\"source [find interface/kitprog3.cfg]; set ENABLE_POWER_SUPPLY <mV>; set ENABLE_ACQUIRE 2;\"

• For PSOC[™] 4:

```
_MTB_RECIPE__OPENOCD_INTERFACE=\"source [find interface/kitprog3.cfg]; set
ENABLE_POWER_SUPPLY <mV>; set PSOC4_USE_ACQUIRE 2;\"
```

Where <mV> defines target voltage in millivolts. For example:

4. Save the changes to the file.



8 Program/debug using J-Link

8 Program/debug using J-Link

Most PSOC[™]-based BSPs default to using the KitProg3/MiniProg4 programmer/debugger launch configurations. This section covers how to use J-Link.

8.1 Configure J-Link programmer/debugger settings

1. Open your ModusToolbox[™] application's *bsp.mk* file and enter the following variable:

BSP_PROGRAM_INTERFACE=JLink

2. Also enter the following variable to specify the path to the J-Link install directory:

MTB_JLINK_DIR=<path to J-Link>

- **3.** Save the *bsp.mk* file.
- **4.** In a bash Terminal run:

make vscode

When the command completes, J-Link configurations will be shown. These are the same configurations described in Program/debug common, but applicable to J-Link.

ſIJ	RUN AND DEBUG	Launch PSoC6 CM4 (JLink) (Hello_World) >
5	> VARIABLES	Launch PSoC6 CM4 (JLink) (Hello_World)
Q	> WATCH	Attach PSoC6 CM4 (JLink) (Hello_World)
	> CALL STACK	
የያ	∨ BREAKPOINTS	Node.js
8		Python Debugger
		C++ (Windows)
±>		C++ (GDB/LLDB)
		Add Config (Hello_World)
80		Add Config (mtb_shared)
		Add Config (workspace)

5. Open the *settings.json* file and *<app>.code-workpace* file to verify the path to the J-Link GDB server. For example, the default on Windows is:

"cortex-debug.JLinkGDBServerPath": "C:/Program Files/SEGGER/JLink/JLinkGDBServerCL.exe"



8 Program/debug using J-Link

-lello_Wor	d > .vscode > {} settings.json >						
12	//mtb//						
13	<pre>"modustoolbox.toolsPath": "C:/Users/follettcj/ModusToolbox/tools_3.1",</pre>						
14	<pre>"cortex-debug.armToolchainPath": "\${config:modustoolbox.toolsPath}/gcc/bin",</pre>						
15	<pre>"cortex-debug.openocdPath": "\${config:modustoolbox.toolsPath}/openocd/bin/openocd.exe",</pre>						
16	"cortex-debug.JLinkGDBServerPath.windows": "C:/Program Files/SEGGER/JLink/JLinkGDBServerCL.exe",						
17	<pre>"cortex-debug.JLinkGDBServerPath.osx": "/Applications/SEGGER/JLink/JLinkGDBServerCLExe",</pre>						
18	"cortex-debug.JLinkGDBServerPath.linux": "JLinkGDBServerCLExe"						
19 } mtb-ex	ample-hal-hello-world.code-workspace ×						
mtb-ex	ample-hal-hello-world.code-workspace × d > {} mb-example-hal-hello-world.code-workspace >						
mtb-ex							
) mtb-ex	d > {} mtb-example-hal-hello-world.code-workspace >						
mtb-exite	<pre>id > {} mtb-example-hal-hello-world.code-workspace > "modustoolbox.toolsPath": "C:/Users/follettcj/ModusToolbox/tools_3.1",</pre>						
mtb-exa lello_Wor 21 22	<pre>id > {} mtb-example-hal-hello-world.code-workspace > "modustoolbox.toolsPath": "C:/Users/follettcj/ModusToolbox/tools_3.1", "cortex-debug.armToolchainPath": "\${config:modustoolbox.toolsPath}/gcc/bin", "cortex-debug.openocdPath": "\${config:modustoolbox.toolsPath}/openocd/bin/openocd.exe",</pre>						
ello_Wor 21 22 23	<pre>id > {} mtb-example-hal-hello-world.code-workspace > "modustoolbox.toolsPath": "C:/Users/follettcj/ModusToolbox/tools_3.1", "cortex-debug.armToolchainPath": "\${config:modustoolbox.toolsPath}/gcc/bin",</pre>						

8.2 Connect the Kit

Follow the instructions provided with the kit and from SEGGER to connect it to the computer with the J-Link probe.



9 Multi-core debugging

9 Multi-core debugging

Projects created for VS Code also provide debug configurations for multi-core applications. They support these probes:

- KitProg3 onboard programmer
- MiniProg4
- J-Link (See Configure J-Link programmer/debugger settings)

9.1 Configurations

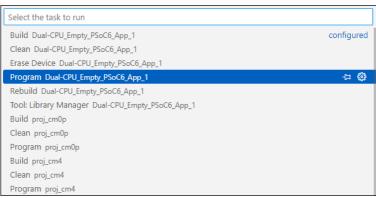
The configurations support debugging one core at a time and multiple cores as well. After the application has opened, there will be several configurations available for use in the **Run and Debug** tab of Activity Bar as shown.

ß	RUN AND DEBUG	Multi-Core Debug (KitProg3_MiniProg4) (Dual-CPU_Empty_PSoC6_App_1) >					
	> VARIABLES	Multi-Core Debug (KitProg3_MiniProg4) (Dual-CPU_Empty_PSoC6_App_1)					
Q	> WATCH	Launch PSoC6 CM0+ (KitProg3_MiniProg4) (proj_cm0p)					
1	> CALL STACK						
с С	> BREAKPOINTS	Attach PSoC6 CM0+ (KitProg3_MiniProg4) (proj_cm0p)					
~		Launch PSoC6 CM4 (KitProg3_MiniProg4) (proj_cm4)					
\$ ²		Attach PSoC6 CM4 (KitProg3_MiniProg4) (proj_cm4)					

These include:

- **Multi-Core Debug**: programs multiple hex files, launches OpenOCD/J-Link GDB Server and starts multicore debug session
- Launch <device>: launches debug session on the chosen core
- Attach <device>: attaches to the running core

In addition to these configurations, there is associated VSCode tasks available through the main menu **Terminal > Run Task**:



- Erase Device: erases all internal memory banks.
- **Program <application_name>**: downloads combined hex file into the flash.
- **Program <project_name>**: downloads project-specific hex file into the flash.
- Erase All: If present, erases all internal and external memories.

9.2 Launch the configuration

To launch multi-core debugging, run the **Multi-Core Debug** configuration. You will end up with a debug session containing two debug processes in CALL STACK view.



9 Multi-core debugging

<u>File Edit Selection View Go</u> ··· $\leftrightarrow \rightarrow$ $\land \land \land$	oual-CPU_E	mpty_PSoC	6_App (Workspace)		08 —	
RUN AND DEBUG 🕨 Multi-Core Debug (JLink) (Dual-CPU_Empty_PSoC6 🗄 🖞	I⊳ ?	* 1 *	O □ ~ ty_PSoC6_App.code-workspace	C main.c ×	~<1	© ⊡
<pre>v VARIABLES v Local result: <optimized out=""> > Global</optimized></pre>	proj_cmOp > C mainc > ③ main(void) 38 * including Cypress's product in a High Risk Product, the main 39 * of such system or application assumes all risk of such us 40 * so agrees to indemnify Cypress against all liability.					
> Static > Registers	41 42 43 #include "cy_pdl.h" 44 #include "cycfg.h" 45 #include "cybsp.h" 46				8 3	
WATCH ア CALL STACK ア ☆ ① Multi-Core Debug (JLink) (Dual-CPU_Empty_PSoC6_App) PAUSED ON BREAKPOINT main@0x10000118 main.c ② Attach PSoC6 CM4 (JLink) (proj_cm4) RUNNING			<pre>int main(void) { /* Enable global interrupts */ enable_irq(); cy rslt t result;</pre>			
✓ BREARPOINTS OCOTEX PERIPHERALS OCOTEX REGISTERS		Receiv Haltir Tar Read 2 Settir THUME	MS OUTPUT DEBUGICONSOLE TERMINAL 2 bytes @ address &x00000012A (Data = 0x yed monitor command: halt 1g target CPU rget halted (PC = 0x0000012A) 1: bytes @ address 0x10000476C (Data = 0x 1g breakpoint @ address 0x1000476C, Kin), BPHandle = 0x0001 ing target CPU	(8508)	>S + ∨ ∑ bash Ti ∑ gdb-se ∑ gdb-se	rver

Once a session has started, the CM0+ core is halted at the beginning of main(), while the CM4 core is spinning in an endless loop in boot code, waiting for start. It will start and halt at main() as soon as the application running on the CM0+ executes the Cy_SysEnableCM4() function.

In the CALL STACK view you can observe two debug processes, each of them associated with a specific core. You can switch between the cores by selecting the appropriate process.

Note: There is one limitation for XMC7000 MCUs. Before launching a multi-core debug session, you must program the MCU by launching the **Program Application** configuration.



Revision history

Revision history

Revision	Date	Description
**	2023-05-16	New document.
*A	2023-07-18	Added instructions for using MiniProg4 and powering the MCU.
*В	2024-01-25	Updates for version 3.2 tools package.
*C	2024-10-02	Updates for version 3.3 tools package.
*D	2024-10-11	Updated information about tasks and power control.
*E	2024-12-06	Updates for version 3.4 tools package.

Trademarks

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

Edition 2024-12-06 Published by Infineon Technologies AG 81726 Munich, Germany

© 2024 Infineon Technologies AG All Rights Reserved.

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

Document reference IFX-kpt1712774185041

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.