

Quick Start Guide

AmbiqSuite Visual Studio Code Project Example

QS-AVSPE1-1p0



Legal Information and Disclaimers

AMBIQ MICRO INTENDS FOR THE CONTENT CONTAINED IN THE DOCUMENT TO BE ACCURATE AND RELIABLE. THIS CONTENT MAY, HOWEVER, CONTAIN TECHNICAL INACCURACIES, TYPOGRAPHICAL ERRORS OR OTHER MISTAKES. AMBIQ MICRO MAY MAKE CORRECTIONS OR OTHER CHANGES TO THIS CONTENT AT ANY TIME. AMBIQ MICRO AND ITS SUPPLIERS RESERVE THE RIGHT TO MAKE CORRECTIONS, MODIFICATIONS, ENHANCEMENTS, IMPROVEMENTS AND OTHER CHANGES TO ITS PRODUCTS, PROGRAMS AND SERVICES AT ANY TIME OR TO DISCONTINUE ANY PRODUCTS, PROGRAMS, OR SERVICES WITHOUT NOTICE.

THE CONTENT IN THIS DOCUMENT IS PROVIDED "AS IS". AMBIQ MICRO AND ITS RESPECTIVE SUPPLIERS MAKE NO REPRESENTATIONS ABOUT THE SUITABILITY OF THIS CONTENT FOR ANY PURPOSE AND DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS CONTENT, INCLUDING BUT NOT LIMITED TO, ALL IMPLIED WARRANTIES AND CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON- INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHT.

AMBIQ MICRO DOES NOT WARRANT OR REPRESENT THAT ANY LICENSE, EITHER EXPRESS OR IMPLIED, IS GRANTED UNDER ANY PATENT RIGHT, COPYRIGHT, MASK WORK RIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT OF AMBIQ MICRO COVERING OR RELATING TO THIS CONTENT OR ANY COMBINATION, MACHINE, OR PROCESS TO WHICH THIS CONTENT RELATE OR WITH WHICH THIS CONTENT MAY BE USED.

USE OF THE INFORMATION IN THIS DOCUMENT MAY REQUIRE A LICENSE FROM A THIRD PARTY UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF THAT THIRD PARTY, OR A LICENSE FROM AMBIQ MICRO UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF AMBIQ MICRO.

INFORMATION IN THIS DOCUMENT IS PROVIDED SOLELY TO ENABLE SYSTEM AND SOFTWARE IMPLEMENTERS TO USE AMBIQ MICRO PRODUCTS. THERE ARE NO EXPRESS OR IMPLIED COPYRIGHT LICENSES GRANTED HEREUNDER TO DESIGN OR FABRICATE ANY INTEGRATED CIRCUITS OR INTEGRATED CIRCUITS BASED ON THE INFORMATION IN THIS DOCUMENT. AMBIQ MICRO RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN. AMBIQ MICRO MAKES NO WARRANTY, REPRESENTATION OR GUARANTEE REGARDING THE SUITABILITY OF ITS PRODUCTS FOR ANY PARTICULAR PURPOSE, NOR DOES AMBIQ MICRO ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT, AND SPECIFICALLY DISCLAIMS ANY AND ALL LIABILITY, INCLUDING WITHOUT LIMITATION CONSEQUENTIAL OR INCIDENTAL DAMAGES. "TYPICAL" PARAMETERS WHICH MAY BE PROVIDED IN AMBIQ MICRO DATA SHEETS AND/OR SPECIFICATIONS CAN AND DO VARY IN DIFFERENT APPLICATIONS AND ACTUAL PERFORMANCE MAY VARY OVER TIME. ALL OPERATING PARAMETERS, INCLUDING "TYPICALS" MUST BE VALIDATED FOR EACH CUSTOMER APPLICATION BY CUSTOMER'S TECHNICAL EXPERTS. AMBIQ MICRO DOES NOT CONVEY ANY LICENSE UNDER NEITHER ITS PATENT RIGHTS NOR THE RIGHTS OF OTHERS. AMBIQ MICRO PRODUCTS ARE NOT DESIGNED, INTENDED, OR AUTHORIZED FOR USE AS COMPONENTS IN SYSTEMS INTENDED FOR SURGICAL IMPLANT INTO THE BODY, OR OTHER APPLICATIONS INTENDED TO SUPPORT OR SUSTAIN LIFE, OR FOR ANY OTHER APPLICATION IN WHICH THE FAILURE OF THE AMBIQ MICRO PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR. SHOULD BUYER PURCHASE OR USE AMBIQ MICRO PRODUCTS FOR ANY SUCH UNINTENDED OR UNAUTHORIZED APPLICATION, BUYER SHALL INDEMNIFY AND HOLD AMBIQ MICRO AND ITS OFFICERS, EMPLOYEES, SUBSIDIARIES, AFFILIATES, AND DISTRIBUTORS HARMLESS AGAINST ALL CLAIMS, COSTS, DAMAGES, AND EXPENSES, AND REASONABLE ATTORNEY FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PERSONAL INJURY OR DEATH ASSOCIATED WITH SUCH UNINTENDED OR UNAUTHORIZED USE, EVEN IF SUCH CLAIM ALLEGES THAT AMBIQ MICRO WAS NEGLIGENT REGARDING THE DESIGN OR MANUFACTURE OF THE PART.

Revision History

Revision	Date	Description
1.0	June 2024	Initial release

Table of Contents

1	Introduction	5
2	Software Setup	6
3	Installation	8
	Example Quick-Start.....	8
	Hello World Walkthrough.....	8

Table of Figures

Figure 1 - Hello_World Explorer View.....	8
Figure 2 - launch.json Configuration	9
Figure 3 - Hello_World Breakpoint.....	10
Figure 4 - Start Debugging	10
Figure 5 - Hello_World Running	11
Figure 6 - First Breakpoint.....	12
Figure 7 - Second Breakpoint SWO Output.....	13
Figure 8 - Stop Button	13
Figure 9 - Adding Powershell to the Terminal View.....	13
Figure 10 - Make Command.....	14
Figure 11 - XRTOS View.....	14
Figure 12 - Adding a New View Address	15
Figure 13 - Memory View	15

1

Introduction

This document provides a list of the necessary software to get started with using Visual Studio Code with AmbiqSuite. After installing the necessary packages, a brief walkthrough is provided using the Hello_World example project.

2

Software Setup

- Code Editor
 - VSCode <https://code.visualstudio.com/download>
 - Plugins
 - Cortex-Debug (by marus25)
<https://marketplace.visualstudio.com/items?itemName=marus25.cortex-debug>
 - Also requires debug-tracker-vscode
<https://marketplace.visualstudio.com/items?itemName=mcu-debug.debug-tracker-vscode>
 - MemoryView
<https://marketplace.visualstudio.com/items?itemName=mcu-debug.memory-view>
 - RTOS Views
<https://marketplace.visualstudio.com/items?itemName=mcu-debug.rtos-views>
 - MCU Peripheral Viewer
<https://marketplace.visualstudio.com/items?itemName=mcu-debug.peripheral-viewer>
 - Arm Assembly (by dan-c-underwood)
<https://marketplace.visualstudio.com/items?itemName=dan-c-underwood.arm>
 - RedHat YAML
<https://marketplace.visualstudio.com/items?itemName=redhat.vscode-yaml>
 - Microsoft Makefile Tools
<https://marketplace.visualstudio.com/items?itemName=ms-vscode.makefile-tools>
 - Microsoft C/C++ Extension Pack
<https://marketplace.visualstudio.com/items?itemName=ms-vscode.cpptools-extension-pack> (Includes C/C++ IntelliSense & debugging extension)
 - Microsoft WSL (Windows Subsystem for Linux)
<https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-wsl>

- Optional - Python by Microsoft
<https://marketplace.visualstudio.com/items?itemName=ms-python.python>
- Optional - GitLens by GitKraken
<https://marketplace.visualstudio.com/items?itemName=eamodio.gitlens>
- Compiler
 - GCC 13.2r1 <https://developer.arm.com/downloads/-/arm-gnu-toolchain-downloads> look for the Windows bare-metal target (arm-none-eabi):
<https://developer.arm.com/-/media/Files/downloads/gnu/13.2.rel1/binrel/arm-gnu-toolchain-13.2.rel1-mingw-w64-i686-arm-none-eabi.exe?rev=07af46c1f7574a77969b0f764a1255f0&hash=E5598DC9AB1C892D26C25B6158FFA65C> Note- this actually comes with a GDB server if you prefer it over the Segger GDB server.
- Debugger
 - SeggerGDB Command Line (comes bundled with Segger J-Link tools)
<https://www.segger.com/downloads/jlink/>
- Tools - Required for AmbiqSuite build scripts
 - Python 3.8
 - PyYaml

3

Installation

Install all software packages and ensure the relevant executable directories are added to system PATH variables.

Example Quick-Start

AmbiqSuite R4.5.0 provides Visual Studio Code configuration files for two examples:

- /examples/peripherals/hello_world
- /examples/usb/tinyusb_cdc_msc_freertos

This section provides a brief overview of how to get started running and debugging examples using the prepackaged hello_world Visual Studio Code template.

Hello World Walkthrough

1. Open Visual Studio Code, navigate to File -> Open Folder, and open the AmbiqSuite_R4.5.0/boards/<<EVB>>/examples/peripherals/hello_world folder. The explorer view will look like Figure 1.

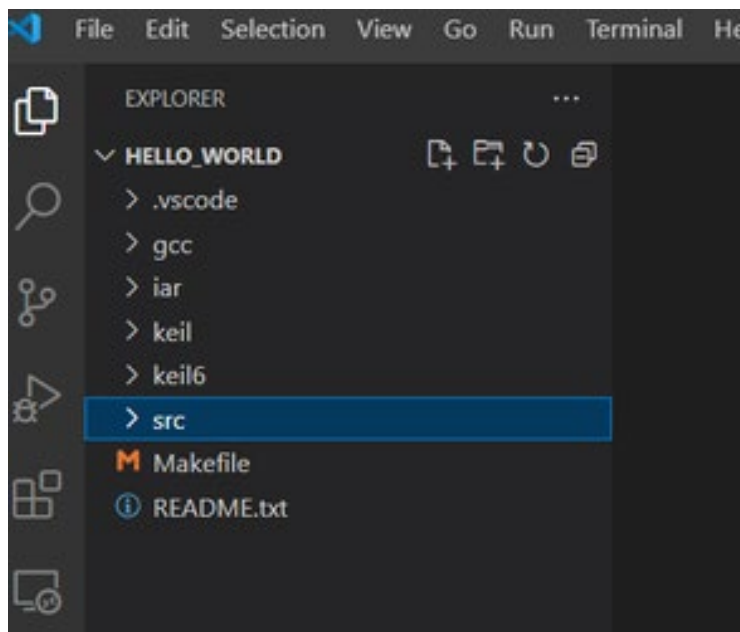


Figure 1- Hello_World Explorer View

2. Navigate to the launch.json file under .vscode and ensure the configurations match your environment.
 1. Ensure "serverpath" points to your J-Link GDB Server executable.
 2. Ensure the "armToolchainPath" path is correct.
 3. Select the appropriate Apollo4 "device" for your selected evaluation board.
 4. If using a debugger other than thew onboard J-Link debugger on the EVB, add the serial number or nick name for "serialNumber". Otherwise, this can be left blank.
 5. Select the "svdFile" path appropriate for your evaluation board.
 6. Ensure "swoFrequency" is set to "1000000" or 1Mhz.

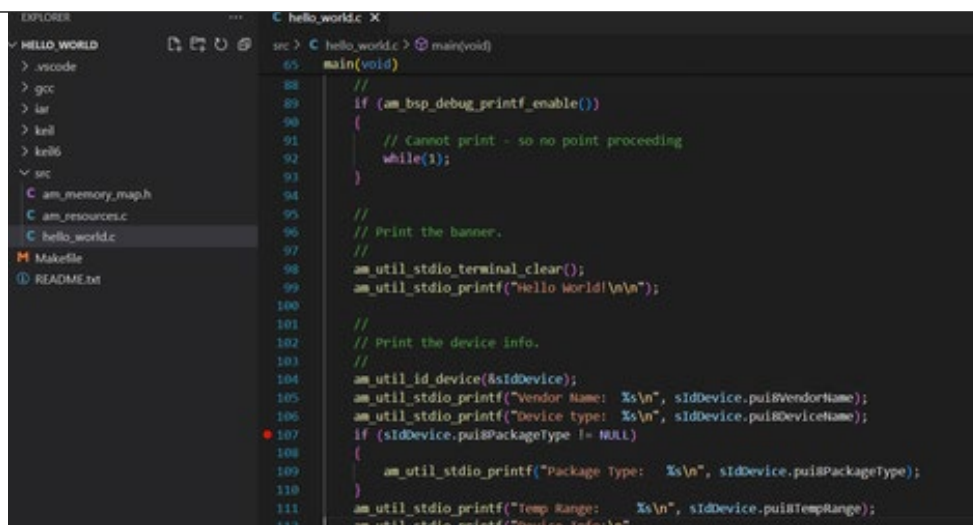
```

1 {
2   "version": "0.2.0",
3   "configurations": [
4     {
5       "cwd": "${fileDirname}",
6       "executable": "../gcc/bin/${fileBasenameWithoutExtension}.axf",
7       "name": "Debug Microcontroller",
8       "request": "launch",
9       "type": "cortex-debug",
10      "servertype": "jlink",
11      "serverpath": "C:/Program Files/SIEMENS/Jlink/JlinkGDBServerCL.exe",
12      "gdbpath": "C:/Program Files (x86)/Arm GNU Toolchain arm-none-eabi/13.2 Rel1/bin/arm-none-eabi-gdb.exe",
13      "armtoolchainPath": "C:/Program Files (x86)/Arm GNU Toolchain arm-none-eabi/13.2 Rel1/bin",
14      "device": "ARM420P-KM8",
15      "interface": "swd",
16      "serialNumber": "",
17      // "runtoMain": true,
18      "runtimeEntryPoint": "main",
19      "svdFile": "${workspaceRoot}/../pack/000/apollo4.svd",
20      "showDeviceOutput": "both",
21      "swoConfig": {
22        "enabled": true,
23        "decoders": [
24          { "type": "console", "label": "JTH", "port": 0, "encoding": "ascii" }
25        ],
26        // "cpuFrequency": 95105000, // This is usually measured, not specified
27        "swoFrequency": 1000000,
28        "source": "probe"
29      }
30    ]
31  }

```

Figure 2 - launch.json Configuration

3. Open hello_world.c in the Visual Studio Code explorer, navigate to line 107, and add a breakpoint by clicking to the left of the line number. Do the same for line 166.

*Figure 3 - Hello_World Breakpoint*

4. Proceed to flash the Apollo4 device by navigating to Run -> Start Debugging. The GDB server will take a few moments to start up, and then the debugger will pause at the first line of main().

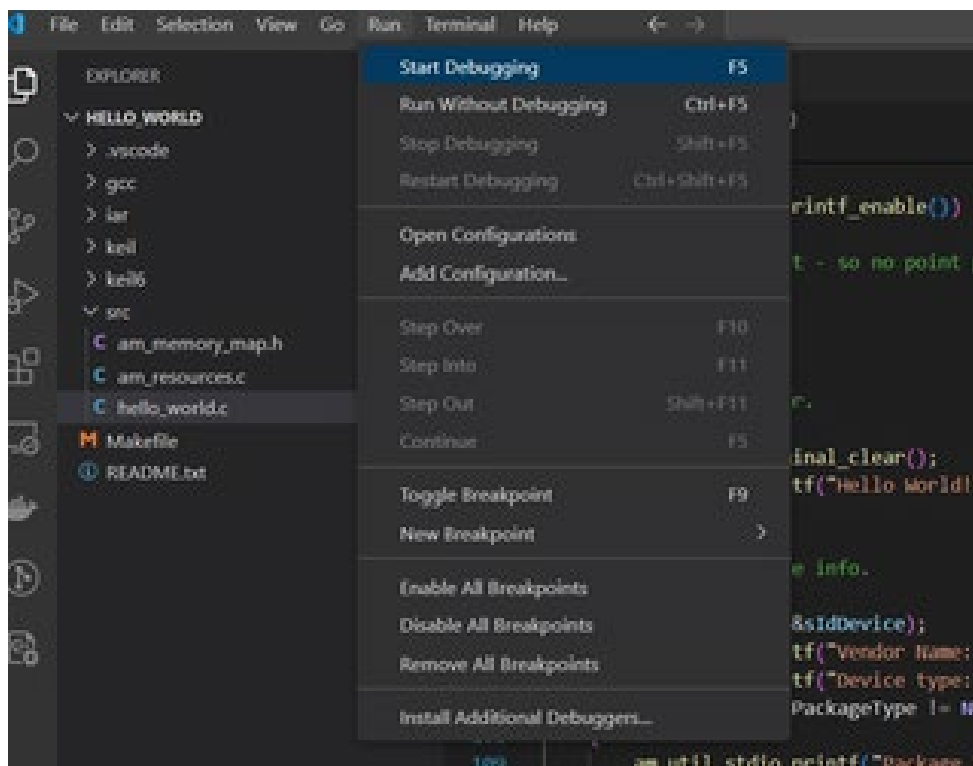
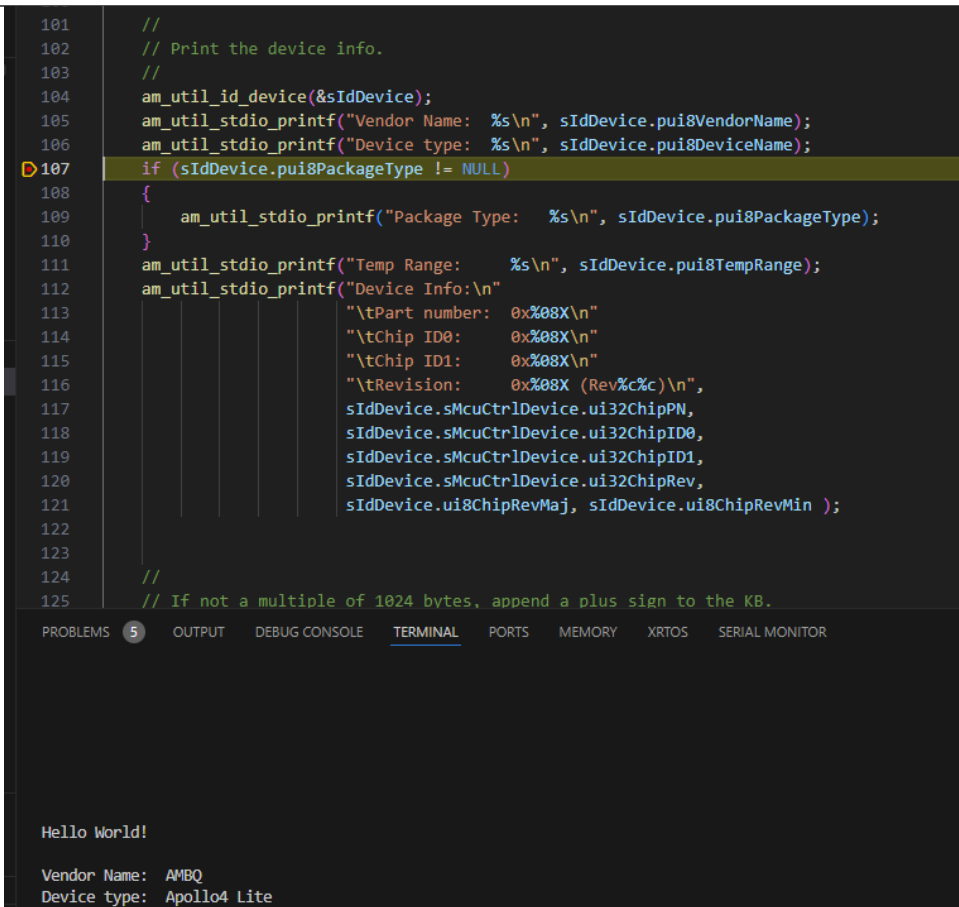
*Figure 4 - Start Debugging*



Figure 5 - Hello_World Running

5. On the left side of the Visual Studio Code interface, you can see the registers listed under XPERIPHERALS. Please see Figure 5. Note that if the registers are not populated, your SVD path in launch.json may be incorrect.
6. At the bottom of the viewer, if you switch the console over to "TERMINAL", you can choose between gdb-server and the SWO interface. Switch the terminal view to "SWO:ITM[port:0]" so that you can view the program output, and click on the continue button to get to the first breakpoint.



```

101 //
102 // Print the device info.
103 //
104 am_util_id_device(&sIdDevice);
105 am_util_stdio_printf("Vendor Name: %s\n", sIdDevice.pui8VendorName);
106 am_util_stdio_printf("Device type: %s\n", sIdDevice.pui8DeviceName);
107 if (sIdDevice.pui8PackageType != NULL)
108 {
109     am_util_stdio_printf("Package Type: %s\n", sIdDevice.pui8PackageType);
110 }
111 am_util_stdio_printf("Temp Range: %s\n", sIdDevice.pui8TempRange);
112 am_util_stdio_printf("Device Info:\n"
113                     "\tPart number: 0x%08X\n"
114                     "\tChip ID0: 0x%08X\n"
115                     "\tChip ID1: 0x%08X\n"
116                     "\tRevision: 0x%08X (Rev%c%c)\n",
117                     sIdDevice.sMcuCtrlDevice.ui32ChipPN,
118                     sIdDevice.sMcuCtrlDevice.ui32ChipID0,
119                     sIdDevice.sMcuCtrlDevice.ui32ChipID1,
120                     sIdDevice.sMcuCtrlDevice.ui32ChipRev,
121                     sIdDevice.ui8ChipRevMaj, sIdDevice.ui8ChipRevMin );
122
123 //
124 // If not a multiple of 1024 bytes, append a plus sign to the KB.
125

```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY XRTOS SERIAL MONITOR

```

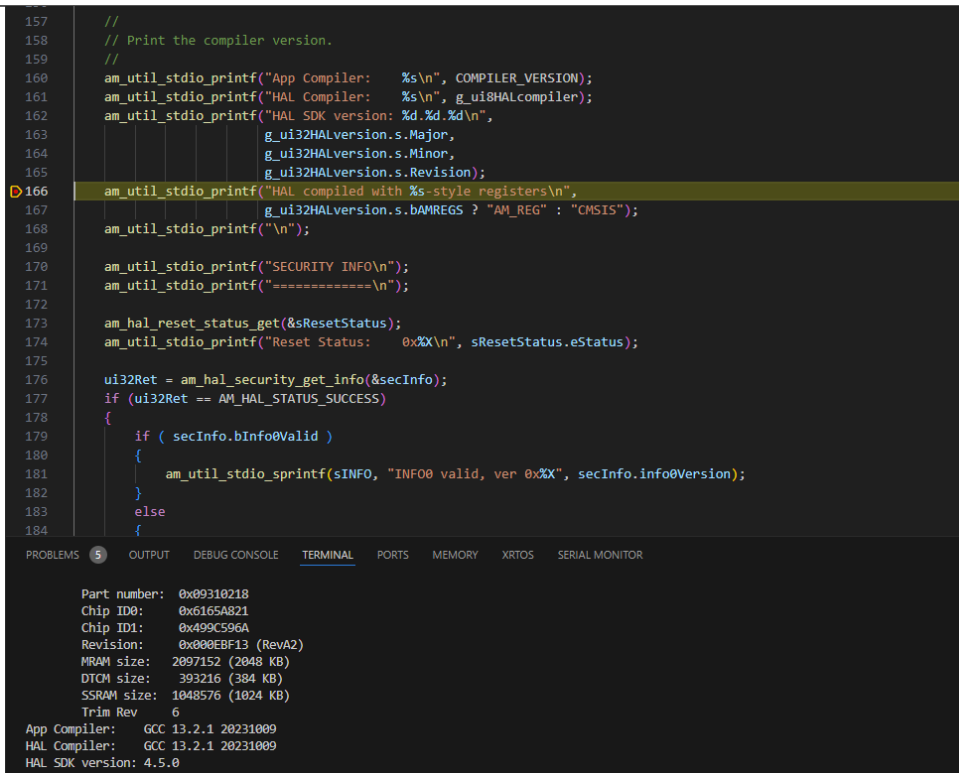
Hello World!

Vendor Name: AMBQ
Device type: Apollo4 Lite

```

Figure 6 - First Breakpoint

- As shown in Figure 6, you will see some of the print statements have already executed and printed to the SWO view in the terminal. Proceeding to the second breakpoint will print more to the terminal. As shown in Figure 7.



```

157 //
158 // Print the compiler version.
159 //
160 am_util_stdio_printf("App Compiler:  %s\n", COMPILER_VERSION);
161 am_util_stdio_printf("HAL Compiler:  %s\n", g_ui8HALcompiler);
162 am_util_stdio_printf("HAL SDK version: %d.%d.%d\n",
163                     g_ui32HALversion.s.Major,
164                     g_ui32HALversion.s.Minor,
165                     g_ui32HALversion.s.Revision);
166 am_util_stdio_printf("HAL compiled with %s-style registers\n",
167                     g_ui32HALversion.s.bAMREGS ? "AM_REG" : "CMSIS");
168 am_util_stdio_printf("\n");
169
170 am_util_stdio_printf("SECURITY INFO\n");
171 am_util_stdio_printf("=====\n");
172
173 am_hal_reset_status_get(&sResetStatus);
174 am_util_stdio_printf("Reset Status:  0x%08X\n", sResetStatus.eStatus);
175
176 ui32Ret = am_hal_security_get_info(&secInfo);
177 if (ui32Ret == AM_HAL_STATUS_SUCCESS)
178 {
179     if ( secInfo.bInfoValid )
180     {
181         am_util_stdio_sprintf(sINFO, "INFO0 valid, ver 0x%08X", secInfo.info0Version);
182     }
183     else
184     {

```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY XRTOS SERIAL MONITOR

```

Part number: 0x09310218
Chip ID0:    0x6165A821
Chip ID1:    0x499C596A
Revision:    0x000EBF13 (RevA2)
MRAM size:   2097152 (2048 KB)
DTCM size:   393216 (384 KB)
SSRAM size:  1048576 (1024 KB)
Trim Rev     6
App Compiler: GCC 13.2.1 20231009
HAL Compiler:  GCC 13.2.1 20231009
HAL SDK version: 4.5.0

```

Figure 7 - Second Breakpoint SWO Output

8. To exit the debugger mode, click on the stop button, as shown in Figure 8.

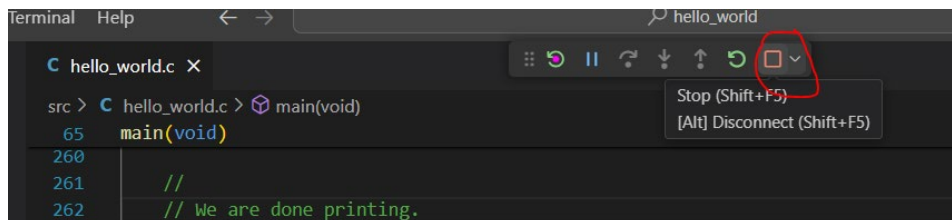


Figure 8 - Stop Button

9. If you make edits to your source file, you'll want to recompile before you start the debugger session again. One way to do this is to open the powershell under TERMINAL (if its not visible, click on the "+" on the right). Please see Figure 9 below.

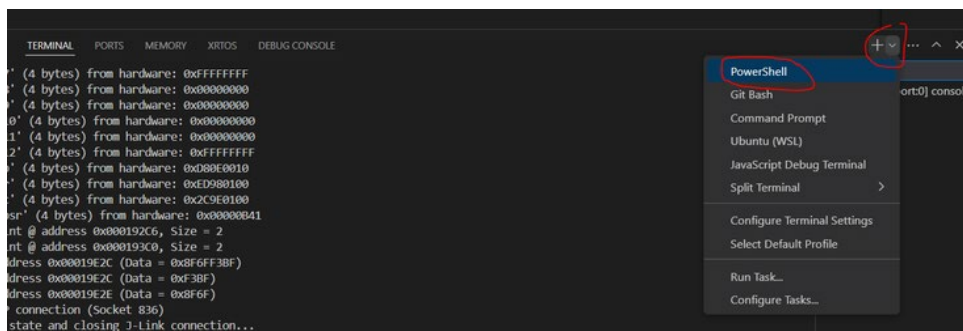


Figure 9 - Adding Powershell to the Terminal View

10. Within the powershell terminal, if you've already made your edits and are ready to rebuild, type in "make" and hit "return". This will rebuild the project. You can then proceed from step 4 and you should see the effect of the changes made to the project.

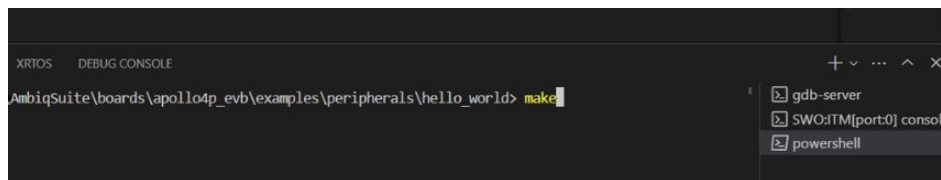


Figure 10 - Make Command

11. If you are working with an example/project that uses an RTOS, you can get more information from the XRTOS view. Below is from tinyusb_cdc_msc_freertos example:

 A screenshot of the XRTOS View window. It displays a table of threads with columns for ID, Address, Task Name, Status, Priority, Stack Start, Stack Top, Stack End, Stack Size, Stack Used, Stack Free, Stack Peak, and Runtime. The table shows three threads: 'usbcd' (SUSPENDED), 'cdc' (RUNNING), and 'IDLE' (READY). The 'cdc' thread is highlighted in yellow. Below the table, it says 'Data collected at 2024-05-15T21:12:44.474Z in 563 ms'.

Thread ID	Address	Task Name	Status	Priority	Stack Start	Stack Top	Stack End	Stack Size	Stack Used	Stack Free	Stack Peak	Runtime
?? 0x100056a8		usbcd	SUSPENDED	0x3,0x3	0x10005098	0x100055bc	0x????????	???	???	1316	???	??,??%
?? 0x10005f18		cdc	RUNNING	0x2,0x2	0x10005708	0x10005e2c	0x????????	???	???	1828	???	??,??%
?? 0x10006388		IDLE	READY	0x0,0x0	0x10005f78	0x1000632c	0x????????	???	???	948	???	??,??%
?? 0x100067f8		Tmr Svc	BLOCKED	0x3,0x3	0x100063e8	0x1000675c	0x????????	???	???	884	???	??,??%

Figure 11 - XRTOS View

12. To get a memory view, you'll go to the MEMORY tab, and then click on the "+" icon to add a new view address. In this case we've entered "0x18000" which is just the starting address of the application. You can create multiple address views.

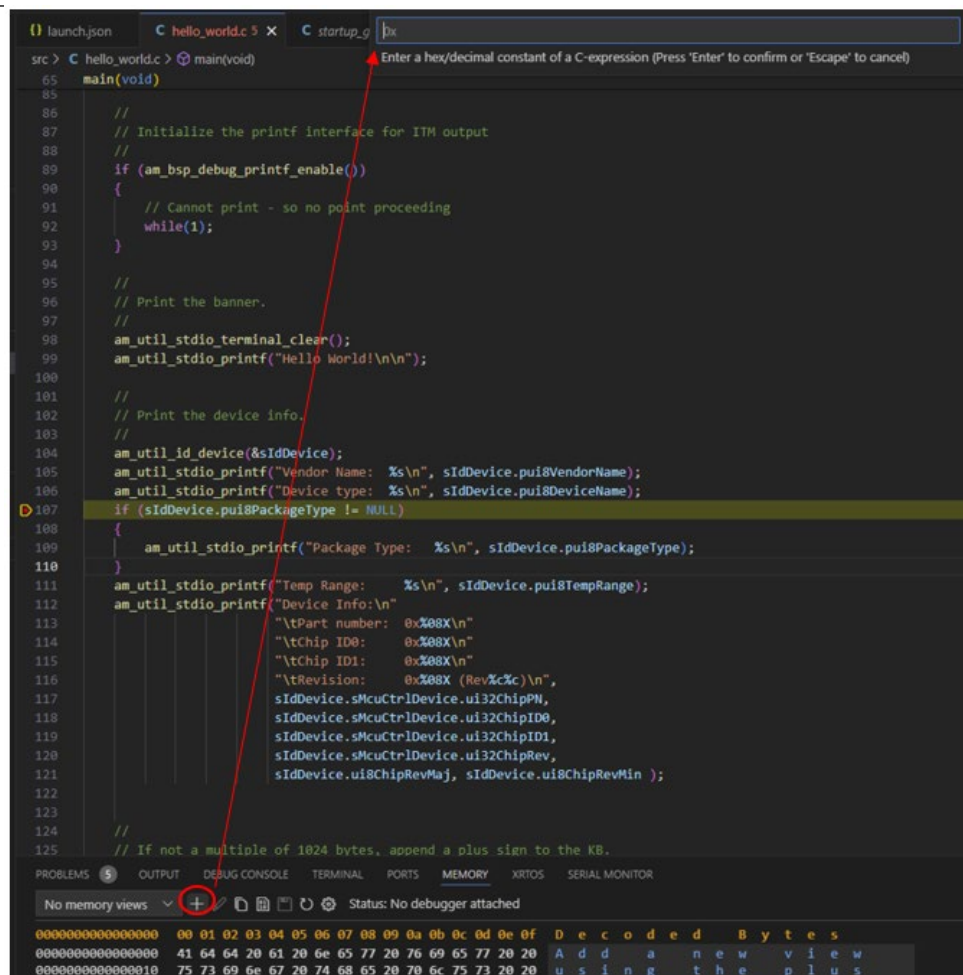


Figure 12 - Adding a New View Address

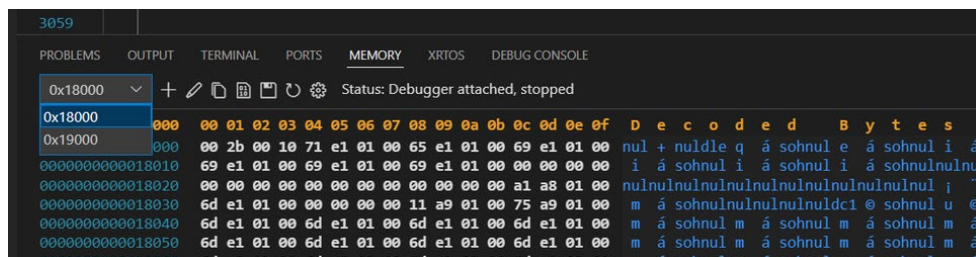


Figure 13 - Memory View



© 2024 Ambiq Micro, Inc. All rights reserved.

6500 River Place Boulevard, Building 7, Suite 200, Austin, TX 78730

www.ambiq.com

sales@ambiq.com

+1 (512) 879-2850

QS-AVSPE1-1p0

June 2024