Quick Start Guide

AmbiqSuite Visual Studio Code Project Example for the Apollo3 Family

QS-AVSPEA3-1p0

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	September 2024	Initial release

Table of Contents

Table of Figures

1 Introduction

This document provides a list of the necessary software and the installation procedure to get started using Visual Studio Code with AmbiqSuite for the Apollo3 family. 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. Note that this document assumes you have your GCC build environment already configured.

Example Quick-Start

AmbiqSuite R3.2.0 provides Visual Studio Code configuration files for two examples:

- /examples/hello_world
- /examples/ freertos_mspi_display

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_R3.2.0/boards/<<EVB>>/examples/hello_world folder. The explorer view will look like Figure 1.

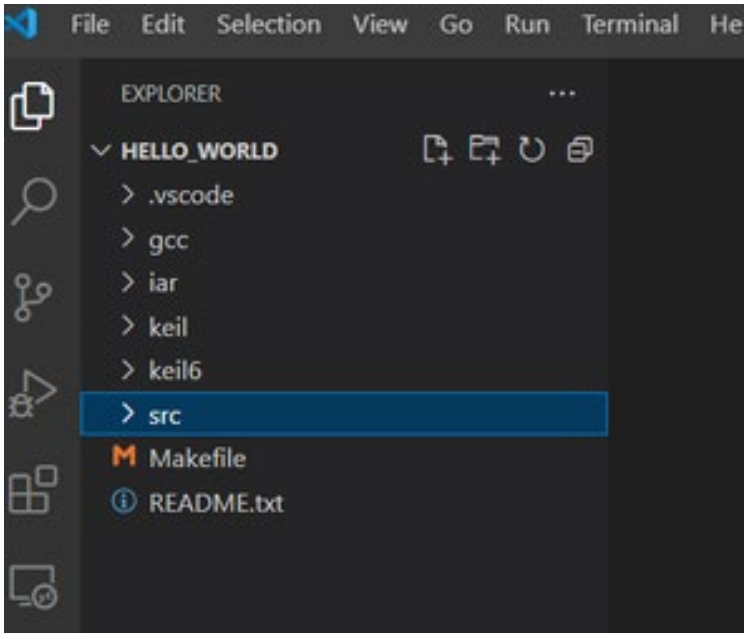

Figure 1- Hello_World Explorer View

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 Apollo3 “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.

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "cwd": "${fileDirname}",
            "executable": "../gcc/bin/${fileBasenameNoExtension}.axf",
            "name": "Debug Microcontroller",
            "request": "launch",
            "type": "cortex-debug",
            "servertype": "jlink",
            "serverpath": "C:/Program Files/SEGGER/JLink/JLinkGDBServerCL.exe", // Your installation path may vary
            "armToolchainPath": "C:/Program Files (x86)/Arm GNU Toolchain arm-none-eabi/13.2 Rel1/bin", // Your installation path may vary
            "device": "AMA3B2KK-KBR",  // For Apollo3p device selection
            //"device": "AMA3B1KK-KBR", // For Apollo3 device selection
            "interface": "swd",
            "serialNumber": "", // If using external JLink, enter the serial number or nickname here
            "runToEntryPoint": "main",
            "svdFile": "${workspaceRoot}/../../../../pack/SVD/apollo3p.svd", // For Apollo3p device selection
            //"svdFile": "${workspaceRoot}/../../../../pack/SVD/apollo3.svd", // For Apollo3 device selection
            "showDevDebugOutput": "both",
            "swoConfig": {
                "enabled": true,
                "decoders": [
                    { "type": "console", "label": "ITM", "port": 0, "encoding": "ascii" }
                ],
                // "cpuFrequency": 48000000, // This is usually measured, not specified
                "swoFrequency": 1000000,
                "source": "probe"
            },        }
    ]
}
```

Figure 2 - launch.json Configuration

3. Open hello_world.c in the Visual Studio Code explorer, navigate to line 99, and add a breakpoint by clicking to the left of the line number. Do the same for line 125.

```
81
82      //
83      // Initialize the printf interface for ITM output
84      //
85      am_bsp_itm_printf_enable();
86
87      //
88      // Print the banner.
89      //
90      am_util_stdio_terminal_clear();
91      am_util_stdio_printf("Hello World!\n\n");
92
93      //
94      // Print the device info.
95      //
96      am_util_id_device(&sIdDevice);
97      am_util_stdio_printf("Vendor Name: %s\n", sIdDevice.pui8VendorName);
98      am_util_stdio_printf("Device type: %s\n", sIdDevice.pui8DeviceName);
99      if (*sIdDevice.pui8PackageType != NULL)
100     {
101         am_util_stdio_printf("Package Type:   %s\n", sIdDevice.pui8PackageType);
102     }
103     am_util_stdio_printf("Temp Range:     %s\n", sIdDevice.pui8TempRange);
104     am_util_stdio_printf("Device Info:\n"
105                          "\tPart number: 0x%08X\n"
106                          "\tChip ID0:    0x%08X\n"
107                          "\tChip ID1:    0x%08X\n"
108                          "\tRevision:    0x%08X (Rev%c%c)\n",
109                          sIdDevice.sMcuCtrlDevice.ui32ChipPN,
```

Figure 3 - Hello_World Breakpoint

4. Proceed to flash the Apollo3 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().

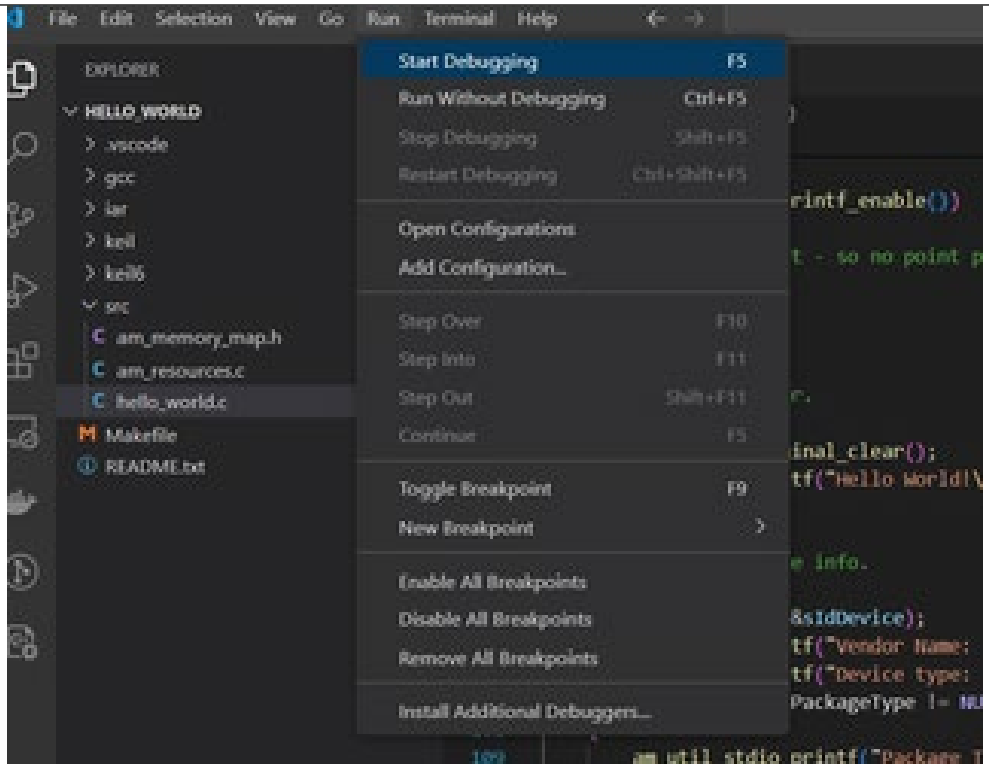

Figure 4 - Start Debugging

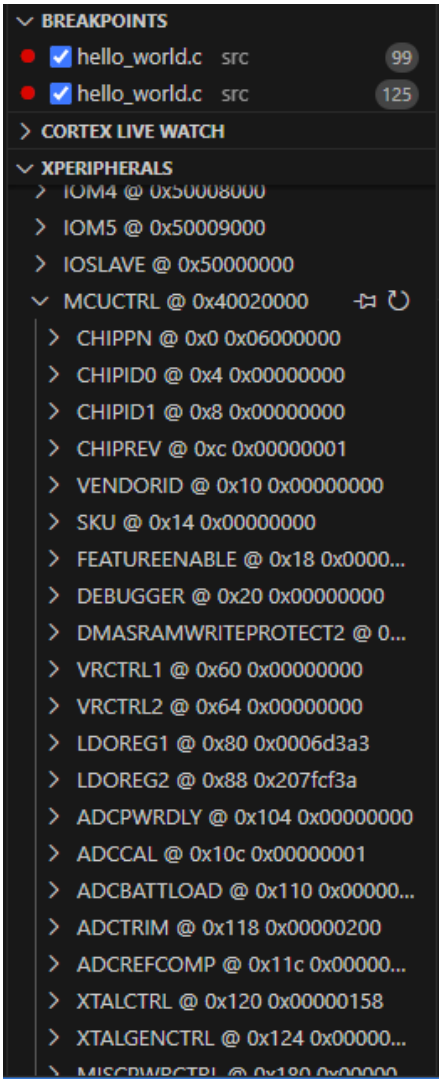

Figure 5 - Hello_World Running

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.

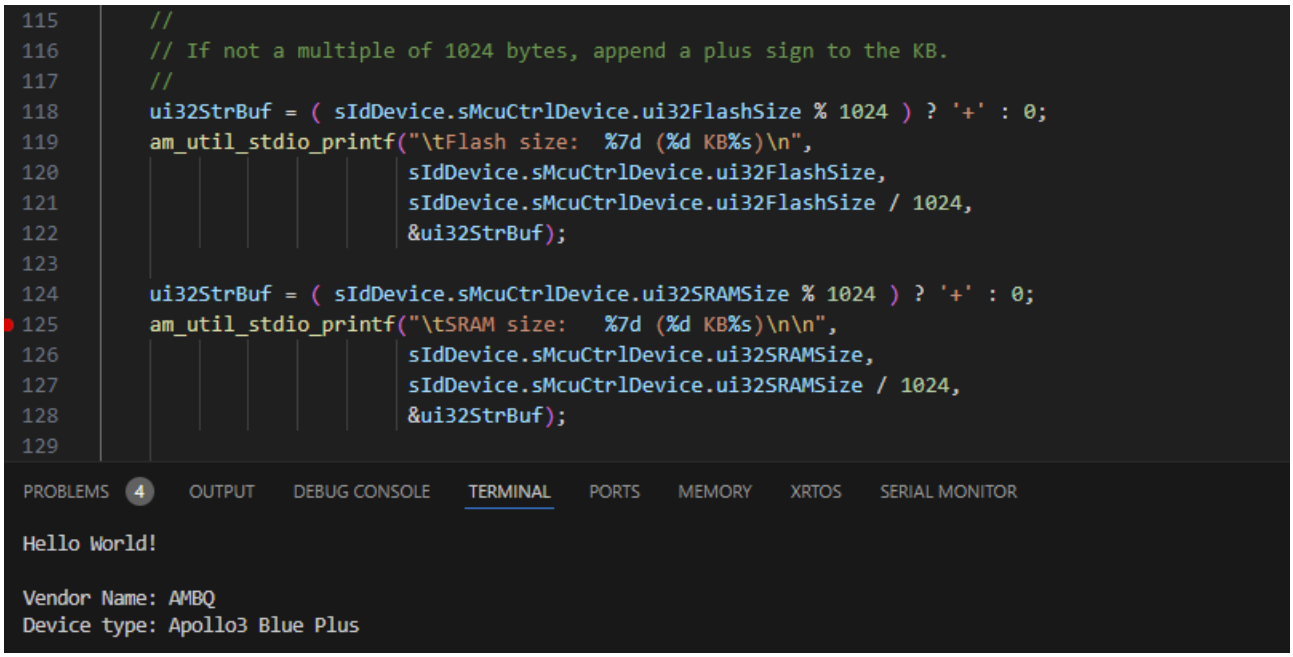

Figure 6 - First Breakpoint

7. 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.

```
99    if (*sIdDevice.pui8PackageType != NULL)
100   {
101       am_util_stdio_printf("Package Type:   %s\n", sIdDevice.pui8PackageType);
102   }
103   am_util_stdio_printf("Temp Range:     %s\n", sIdDevice.pui8TempRange);
104   am_util_stdio_printf("Device Info:\n"
105                        "\tPart number: 0x%08X\n"
106                        "\tChip ID0:    0x%08X\n"
107                        "\tChip ID1:    0x%08X\n"
108                        "\tRevision:    0x%08X (Rev%c%c)\n",
109                        sIdDevice.sMcuCtrlDevice.ui32ChipPN,
110                        sIdDevice.sMcuCtrlDevice.ui32ChipID0,
111                        sIdDevice.sMcuCtrlDevice.ui32ChipID1,
112                        sIdDevice.sMcuCtrlDevice.ui32ChipRev,
113                        sIdDevice.ui8ChipRevMaj, sIdDevice.ui8ChipRevMin );
114
115   //
116   // If not a multiple of 1024 bytes, append a plus sign to the KB.
117   //
118   ui32StrBuf = ( sIdDevice.sMcuCtrlDevice.ui32FlashSize % 1024 ) ? '+' : 0;
119   am_util_stdio_printf("\tFlash size:  %7d (%d KB%s)\n",
120                        sIdDevice.sMcuCtrlDevice.ui32FlashSize,
121                        sIdDevice.sMcuCtrlDevice.ui32FlashSize / 1024,
122                        &ui32StrBuf);
123
124   ui32StrBuf = ( sIdDevice.sMcuCtrlDevice.ui32SRAMSize % 1024 ) ? '+' : 0;
125   am_util_stdio_printf("\tSRAM size:   %7d (%d KB%s)\n\n",
126                        sIdDevice.sMcuCtrlDevice.ui32SRAMSize,
127                        sIdDevice.sMcuCtrlDevice.ui32SRAMSize / 1024,
128                        &ui32StrBuf);
129
130   //
131   // Print the compiler version.
132   //
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY XRTOS SERIAL MONITOR

```
Device Info:
        Part number: 0x077811A0
        Chip ID0:    0x2B2AEB08
        Chip ID1:    0x4C3B4CE6
        Revision:    0x000ECF31 (RevC0)
        Flash size:  2097152 (2048 KB)
```

Figure 7 - Second Breakpoint SWO Output

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

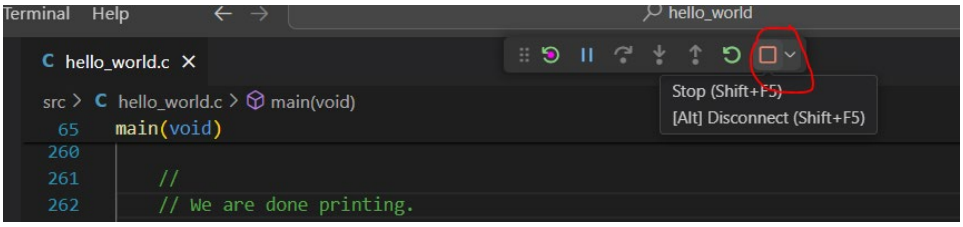

Figure 8 - Stop Button

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 it's not visible, click on the "+" on the right). Please see Figure 9 below.

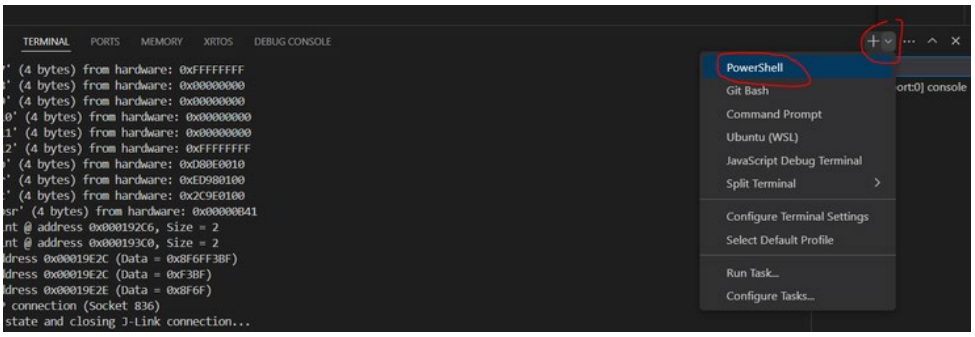

Figure 9 - Adding Powershell to the Terminal View

10. Within the PowerShell terminal, if you have 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.

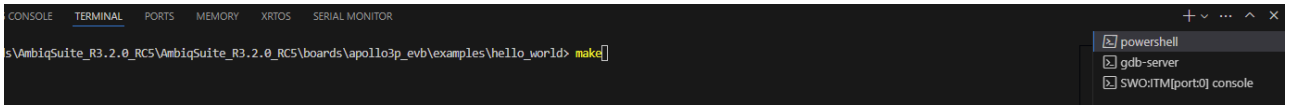

Figure 10 - Make Command

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 freertos_mspi_display example from AmbiqSuite R3.2.0:

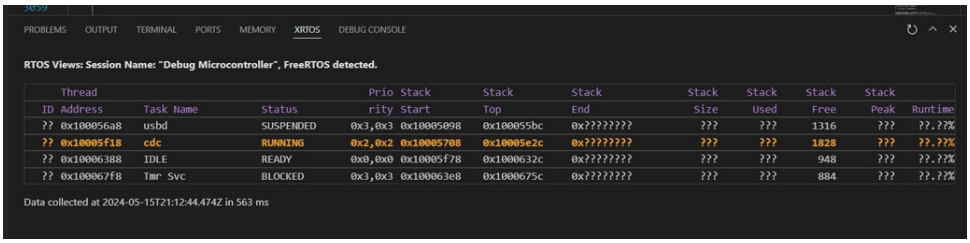

Figure 11 - XRTOS View

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 for the Apollo4 Family. For Apollo3, the starting address is 0xC000. You can create multiple address views.

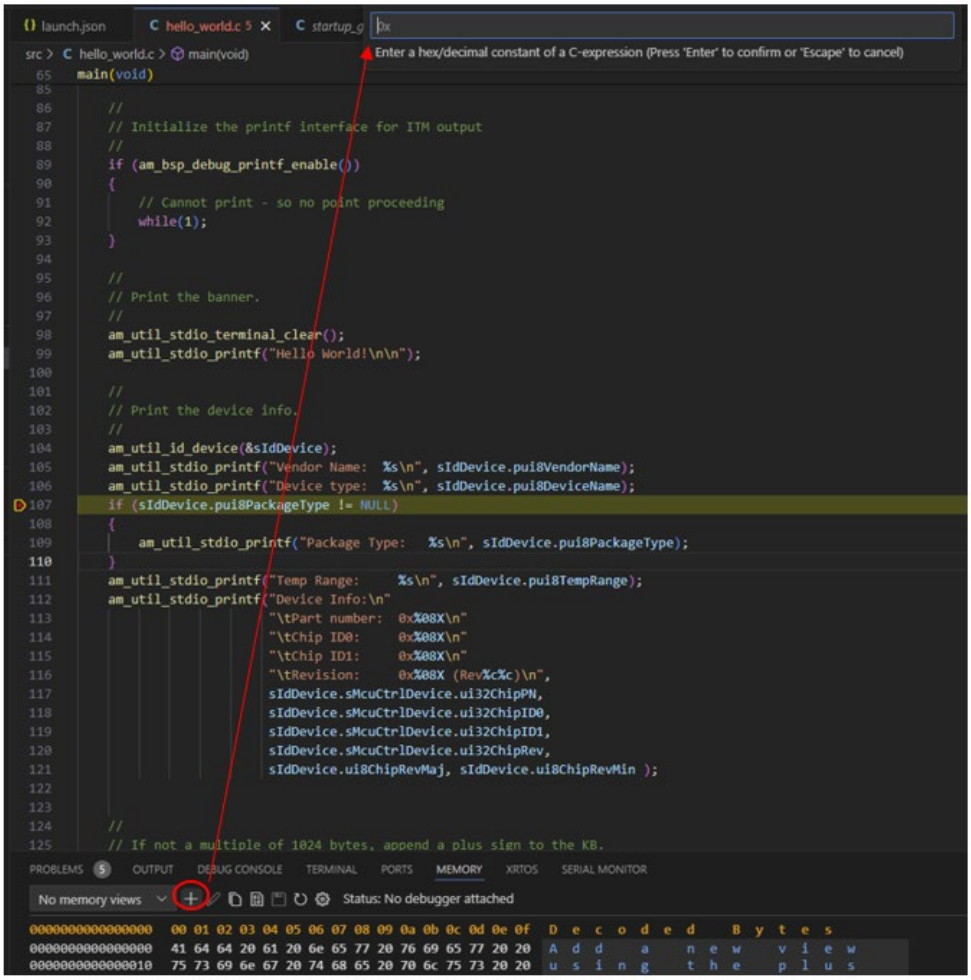

Figure 12 - Adding a New View Address

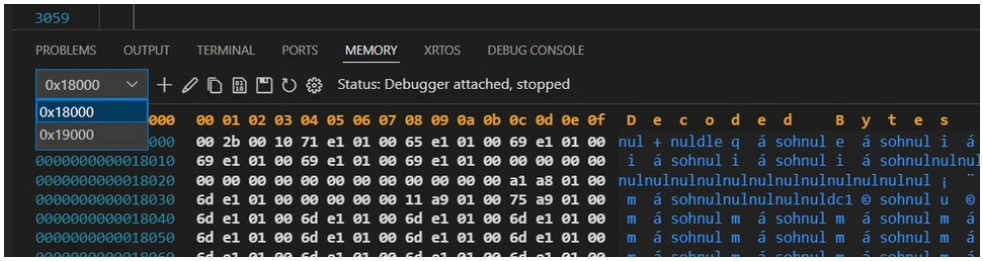

Figure 13 - Memory View

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

www.ambiq.com
sales@ambiq.com
+1 (512) 879-2850

QS-AVSPEA3-1p0
September 2024