Quick Start Guide

AmbiqSuite Visual Studio Code Project Example

QS-AVSPE1-1p0

Legal Information and Disclaimers

Revision History

Revision	Date	Description
1.0	June 2024	Initial release

Table of Contents

Table of Figures

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.

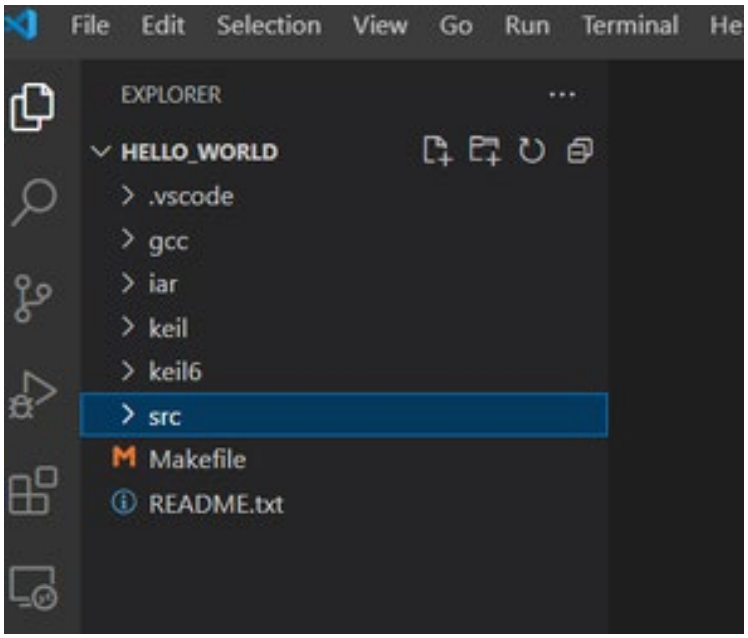

Figure 1- Hello_World Explorer View

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.

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.

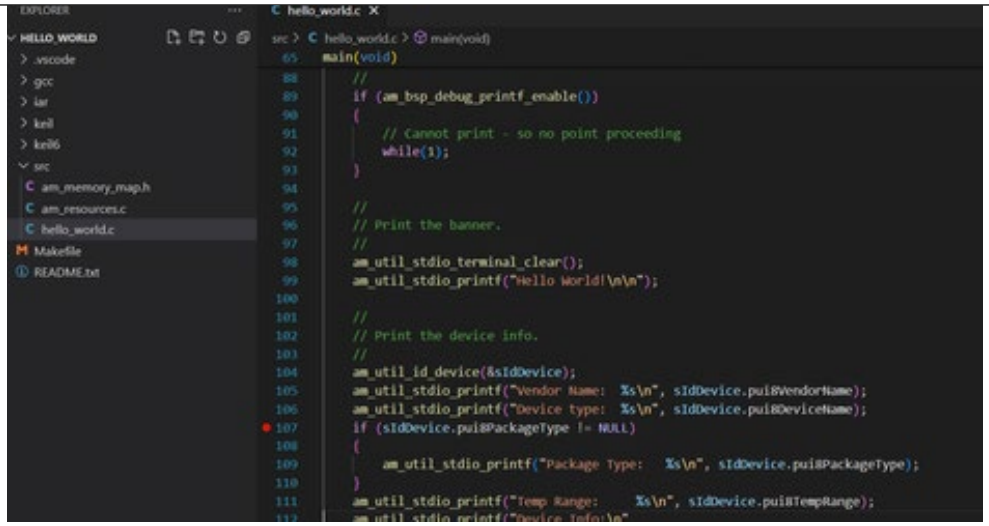

Figure 3 - Hello_World Breakpoint

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().

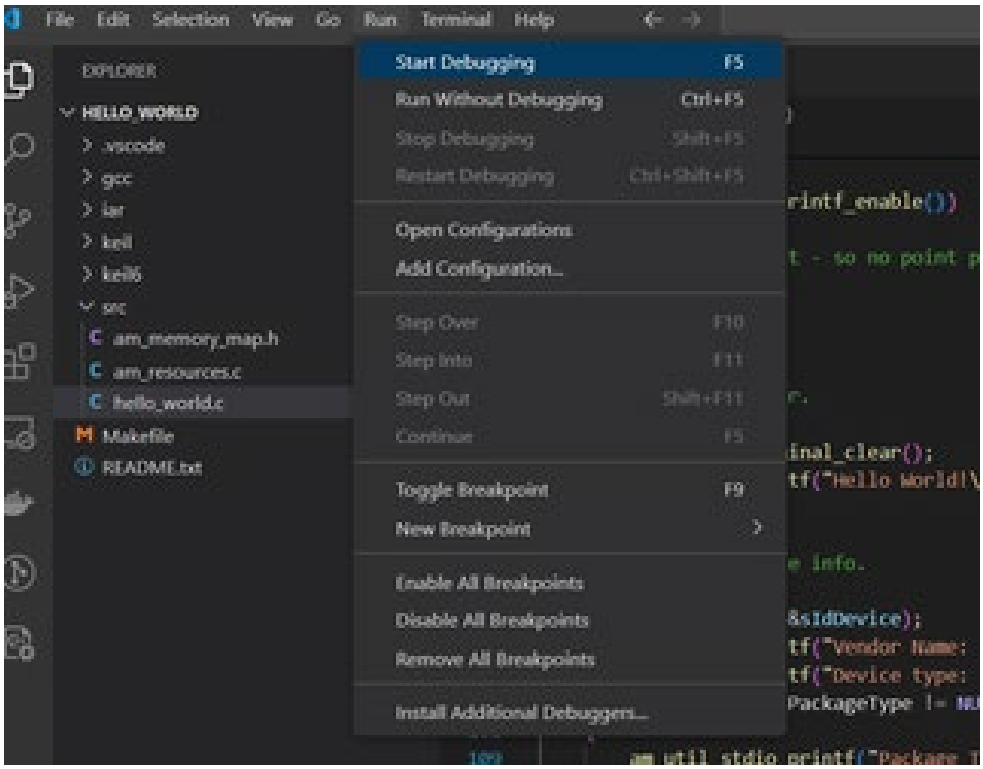

Figure 4 - Start Debugging

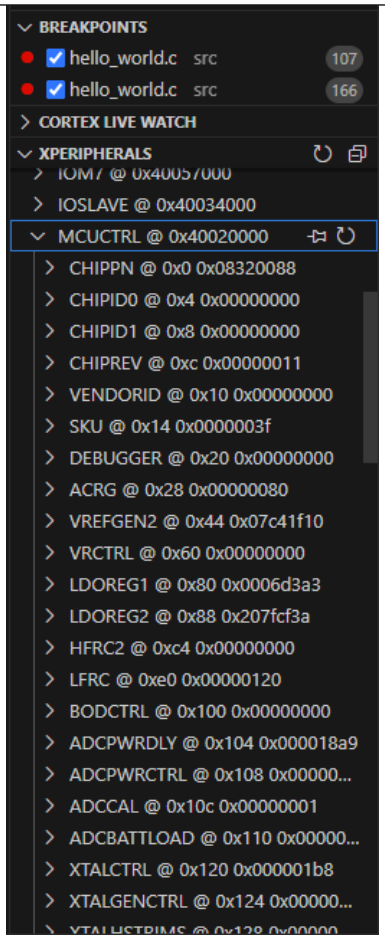

Figure 5 - Hello_World Running

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     //
125     // If not a multiple of 1024 bytes, append a plus sign to the KB.
```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY XRTOS SERIAL MONITOR

```
Hello World!

Vendor Name:  AMBQ
Device type:  Apollo4 Lite
```

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.

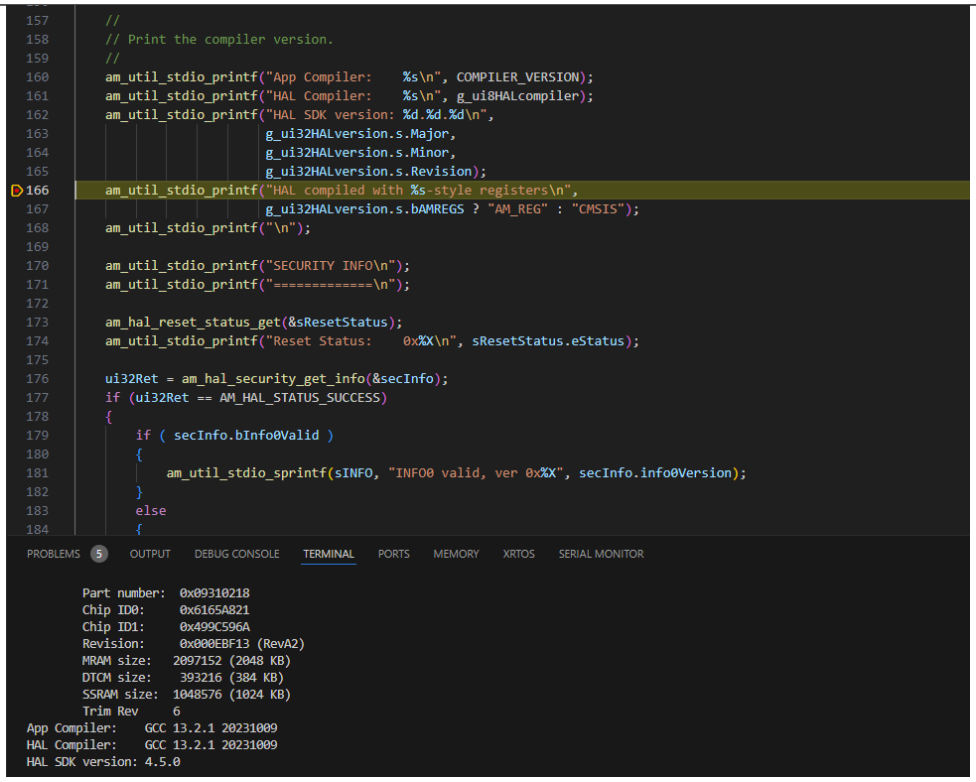

Figure 7 - Second Breakpoint SWO Output

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

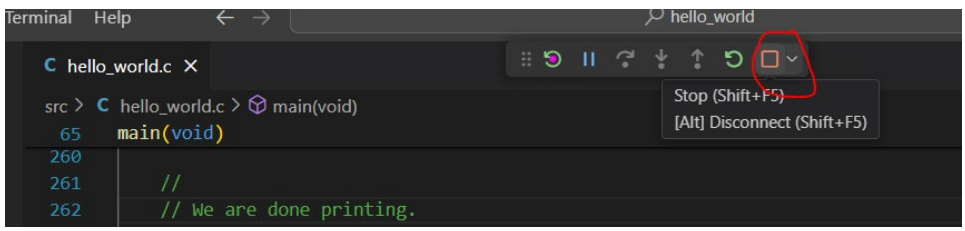

Figure 8 - Stop Button

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.

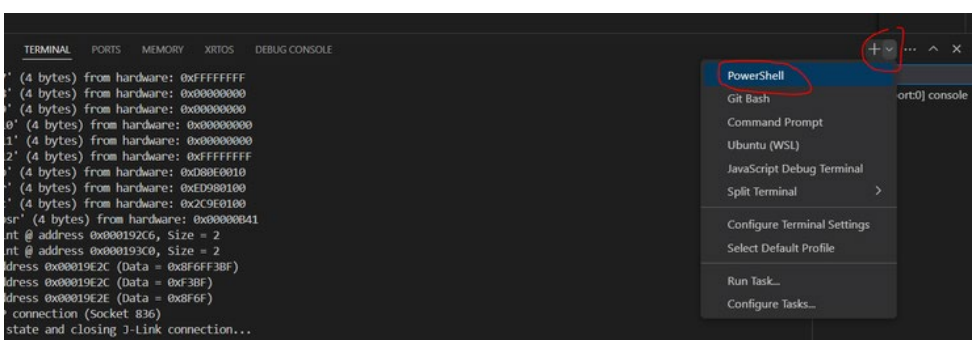

Figure 9 - Adding Powershell to the Terminal View

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:

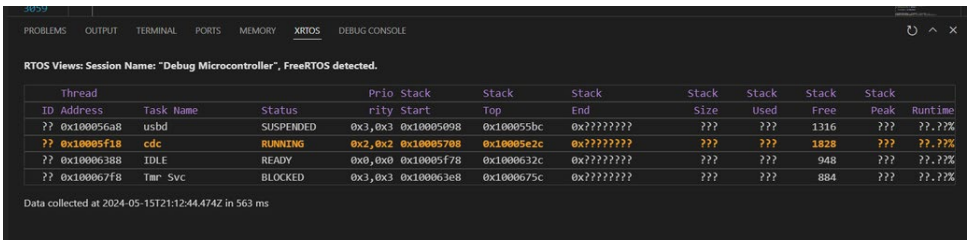

Figure 11 - XRTOS View

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.

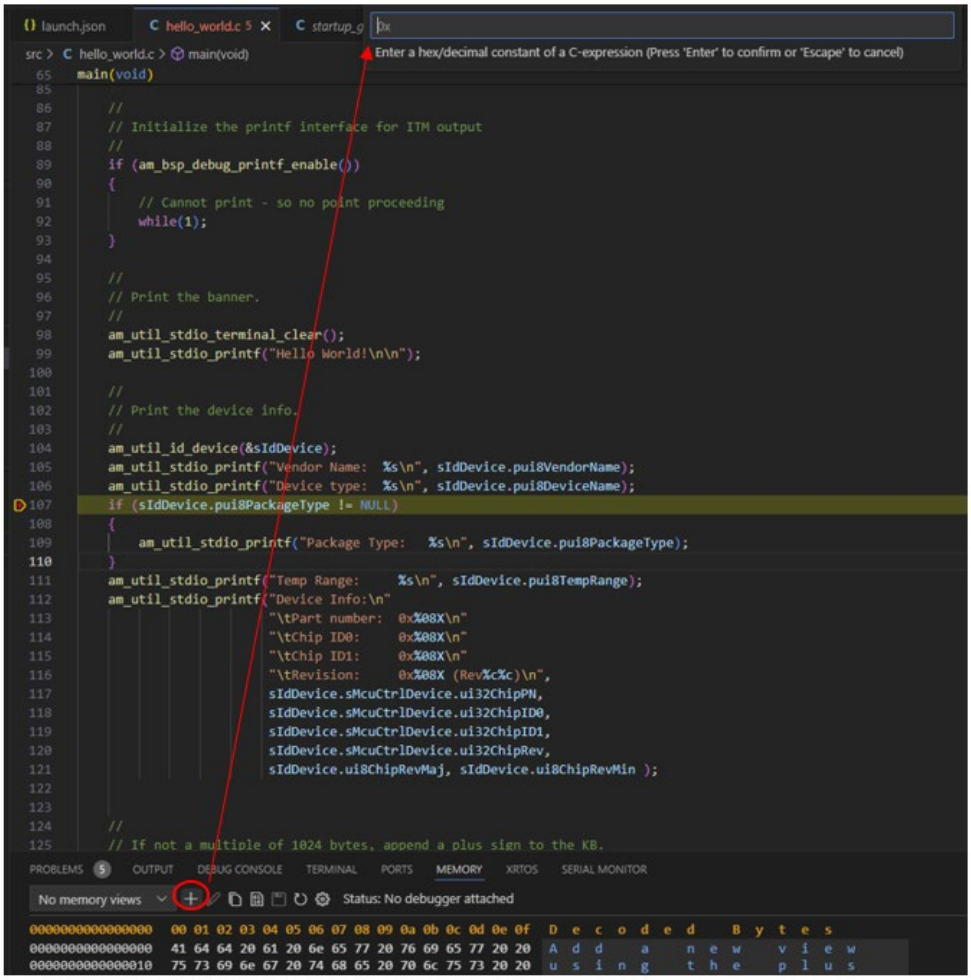

Figure 12 - Adding a New View Address

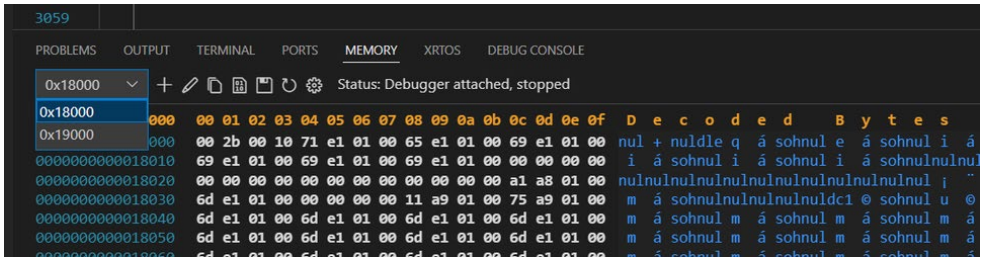

Figure 13 - Memory View

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