VSCode Cortex-Debug Launch Configurations

VSCode Cortex-Debug Launch Configurations

Cortex-Debug is an extension for Visual Studio Code to streamline the debug process when working with ARM Cortex-M microcontrollers. This document covers writing launch configurations (**launch.json**).

- Open the VSCode user settings file *settings.json*: **File** → **Preferences** → **Settings**
- Select User settings:

  Enter "json" in the search bar (with or without double quotes). Locate a link "**Edit in settings.json**" and click on it.

  This will open the file ~/.config/Code/User/settings.json

- Add the following lines to the settings.json file, between the curly braces {}:

```
"cortex-debug.openocdPath": "/usr/bin/openocd",
"cortex-debug.armToolchainPath.linux": "/opt/toolchains/gcc-arm- none-eabi-10.3-2021.10/b
"cortex-debug.armToolchainPath.windows": "C:\\ProgramData\\chocolatey\\bin",
"cortex-debug.gdbPath.linux": "/opt/toolchains/gcc-arm-none-eabi-10.3-2021.10/bin//arm-no
"cortex-debug.gdbPath.windows": "C:\\ProgramData\\chocolatey\\bin\\arm-none-eabi-gdb.exe"
```

Please note: The paths on your system might be different. Make sure the path matches your actual file locations.

- Save the file **settings.json**

Launch configurations

To run or debug a simple app in VS Code, we can select Run and Debug on the Debug start view or we can press F5 and VS Code will try to run the currently active file.

Creating a launch configuration file is beneficial because it allows us to configure and save debugging setup details. VSCode keeps debugging configuration information in a *launch.json* file located in a .vscode folder in the workspace (project root folder).

The *launch.json* file is used to configure the debugger in Visual Studio Code.

Parameters

Here is a list of parameters in *launch.json*. Configure them for your specific device and environment.

- **cwd**: Path of project
- **configFiles**: OpenOCD configuration file(s) to load
- **device**: Target Device Identifier
- **interface**: Debug Interface type to use for connections (defaults to SWD) – Used for J-Link and BMP probes.
- **name**: Name of configuration; appears in the launch configuration dropdown menu.
- **preLaunchTask**: Task to run before debug session starts. Specify a task defined in tasks.json.
- **request**: Request type of configuration. Can be "launch" or "attach".
- **runToEntryPoint**: If enabled the debugger will run until start of the main function.
- **serialNumbe**r: J-Link specific parameter. J-Link Serial Number – only needed if multiple J-Links are connected to the computer
- **servertype**: GDB Server type – supported types are jlink, openocd, pyocd, pe and stutil
- **svdFile**: Path to an SVD file describing the peripherals of the microcontroller; if not supplied then one may be selected based upon the 'device' entered. This may be automatically loaded depending on the "device".
- **swoConfig**: SWO/ITM configuration.
- **enabled**: Enable SWO decoding.
- **cpuFrequency**: Target CPU frequency in Hz.
- **swoFrequency**: SWO frequency in Hz.
- **source**: Source for SWO data. Can either be "probe" to get directly from debug probe, or a serial port device to use a serial port external to the debug probe.
- **decoders**: SWO Decoder Configuration
- **label**: A label for the output window.
- **port**: ITM Port Number

OpenOCD Specific Configuration

- **configFiles**

  OpenOCD configuration files to use when debugging. Exactly equivalent to the OpenOCD command line argument -f.

- **searchDir**

  Directories to search for configuration files in. Exactly equivalent to the OpenOCD command line argument -s. If no search directories are specified, it defaults to the configured cwd.

- **openOCDPreConfigLaunchCommands**

  OpenOCD commands to run prior to loading configuration files.

- **openOCDLaunchCommands**

  OpenOCD commands to run in order to launch target.

OpenOCD GDB Server

We modify *launch.json* to configure debug features.

Below is an example of a basic launch configuration using the OpenOCD GDB server.

In this configuration the device parameter is not required – but can be supplied to allow auto-selecting an appropriate SVD file if possible.

There is one OpenOCD specific parameter that must be supplied. The configFiles property takes an arrays of strings that are used to load openocd configuration files. These can either be a files in the openocd search path (like in this example), or a full path to your own configuration file. If you are using OpenOCD supplied files you typically will have either one file from the board section, or a file from the interface section and a file from the target section.

- Open the VSCode launch configuration file *launch.json*: **Run** → **Add Configuration...**
- Copy the following code

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Debug (OpenOCD)",
            "cwd": "${workspaceRoot}",
            "executable": "${workspaceRoot}/build/blinky.elf",
            "request": "launch",
            "type": "cortex-debug",
            "servertype": "openocd",
            "interface": "swd",
            "device": "TM4C123GH6PM",
            "runToEntryPoint": "main",
            "svdFile": "${workspaceRoot}/svd/TM4C123GH6PM.svd",
            "configFiles": [
                "board/ek-tm4c123gxl.cfg"
            ],
            "preLaunchCommands": [
                "set mem inaccessible-by-default off",
                "monitor reset"
            ],
            "postLaunchCommands": [
                "monitor reset init",
                "monitor sleep 200"
            ]
        }
    ]
}
```

- Change "**executable**", "**svdFile**", and "**device**" parameter as appropriate and save it
- **SVD** Files: The "svdFile" entry in the launch.json file is optional, but crucial to embedded system debugging because it describes the device peripheral registers.

References

Coretex-Debug

- marus25.cortex-debug
- Configuring C/C++ debugging

Search

Recent Posts

Integrated Smart Water Management System with Real-Time Monitoring and Automated Control

Battery Monitoring System

Automatic Indoor Plant Monitoring System

Line follower self balancing bot

Elevator Management System

Recent Comments

A WordPress Commenter on Hello world!