

WBZ451 & WBZ653 OpenOCD Configurations

Contents

1. Introduction.....	2
1.1. OpenOCD Feature Support for the WBZ451 & WBZ653 Curiosity Board	2
2. System Requirements.....	2
3. CMSIS-DAP Switcher.....	3
3.1. Software and Package Dependencies	3
3.2. Installation Procedure	3
3.3. Supported Debuggers.....	3
3.3.1. Switching PICKit Basic from MPLAB Firmware to CMSIS-DAP	3
3.3.2. Switching PICKit Basic from CMSIS-DAP Firmware to MPLAB.....	4
3.3.3. Switching PKOB4 Firmware from MPLAB to CMSIS-DAP	5
3.3.4. Switching PKOB4 Firmware from CMSIS-DAP to MPLAB	6
4. OpenOCD Integration with Visual Studio Code	7
5. Steps to Flashing/Programming	11
5.1. WBZ451 Device.....	11
5.2. WBZ653 Device.....	12
6. Steps to Debugging the WBZ451 & WBZ653 Device	13
7. References / Resources:	22
8. Abbreviations:	22
9. Document History:.....	22

1. Introduction

OpenOCD (Open On-Chip Debugger) is an open-source tool that provides debugging, in-system programming, and boundary-scan testing for embedded target devices. It is widely used in embedded systems development to interface with a wide range of hardware devices through JTAG, SWD, or other debug interfaces.

1.1. OpenOCD Feature Support for the WBZ451 & WBZ653 Curiosity Board

Feature	Description	Supported (Yes/No)
Flash Programming	Supports firmware flashing to the target device	Yes
Debug Support	Supports debugging through the SWD (Serial Wire Debug) interface	Yes
Breakpoint Support	Supports up to four hardware breakpoints	Yes
Reset Control	Supports device reset via OpenOCD	Yes
SWD Interface	Supports Serial Wire Debug interface	Yes
GDB Integration	Compatible with GDB debugger	Yes

2. System Requirements

- WBZ451 Curiosity Board 1 No
- WBZ653 Curiosity Board 1 No
- MicroUSB cable 1 No
- Packages
 - "Zephyr-sdk-0.17.0"
 - "openocd_support_wbz451"
 - "Zephyrproject" (Repository)
- Visual Studio Code v1.101.2 or above.
- Python version 3.10 or newer
- PICKit Basic (Optional)
 - Type C USB cable 1 No

3. CMSIS-DAP Switcher

pycmsisdapswitcher is a command line Python utility designed to switch the firmware on a PICKit™ Basic tool or on an evaluation board/kit between the Microchip® implementation and the Arm® CMSIS-DAP v2 implementation. It allows users to easily switch the firmware of supported Microchip tools, enabling compatibility with different development environments.

3.1. Software and Package Dependencies

Requires Python version 3.10 or newer

3.2. Installation Procedure

Step-1: - Install “pycmsisdapswitcher” via “PyPI” using pip:

Step-2: - pip install pycmsisdapswitcher

3.3. Supported Debuggers

The **pycmsisdapswitcher** utility requires a target parameter, which specifies either the tool name or the kit name to be used.

Supported debuggers

- pickitbasic
- evalboard (PKOB4)

3.3.1. Switching PICKit Basic from MPLAB Firmware to CMSIS-DAP

To switch the PICKit Basic tool from MPLAB firmware to the latest CMSIS-DAP v2 implementation using the Microchip server, follow the steps below.

- Connect the PICKit basic tool to your computer using a Type-C cable.
- Run the following command in a terminal or command prompt

Example command: - python -m pycmsisdapswitcher pickitbasic

```
C:\Users\I78270\Desktop>python -m pycmsisdapswitcher pickitbasic

*** Microchip pycmsisdapswitcher firmware switcher version 1.0.2.14 started ***

Found PICKit Basic S/N 020126001RYN002469 running mplab application.

Application file pickit_basic_app_cmsis-dap.hex found in tool pack version 1.2.190

File pickit_basic_app_cmsis-dap.hex saved to cache.

Bootloader mode entered.

Erasing Flash ...
. . .

Application being written and verified ...
. . . . .

Application switch completed.
```

This command downloads the appropriate CMSIS-DAP firmware from the Microchip server and flashes it to the connected PICKit Basic tool

3.3.2. Switching PICKit Basic from CMSIS-DAP Firmware to MPLAB

To switch the PICKit Basic tool back to the MPLAB firmware implementation using a local Hex file, follow these steps

- pickit_basic_app.hex file save it to a directory of your choice.
- Open a command prompt and navigate to the directory containing the pickit_basic_app.hex file.
- Run the command “python -m pycmsisdapswitcher pickitbasic --source=pickit_basic_app.hex”





3.3.3. Switching PKOB4 Firmware from MPLAB to CMSIS-DAP

To switch an evaluation board equipped with **PKOB4** firmware from the MPLAB® implementation to the latest **CMSIS-DAP v2** firmware using the Microchip server, follow these steps:

- Connect the evaluation board to your computer via USB.
- Run the following command in a terminal or command prompt:
`python -m pycmsisdapswitcher evalboard`

```
C:\Users\I78278>python -m pycmsisdapswitcher evalboard

*** Microchip pycmsisdapswitcher firmware switcher version 1.0.2.14 started ***

Found evaluation board or kit S/N RYN250703674 running mlab application.

Application file pkob4_app_cmsis-dap.hex found in tool pack version 1.21.1602

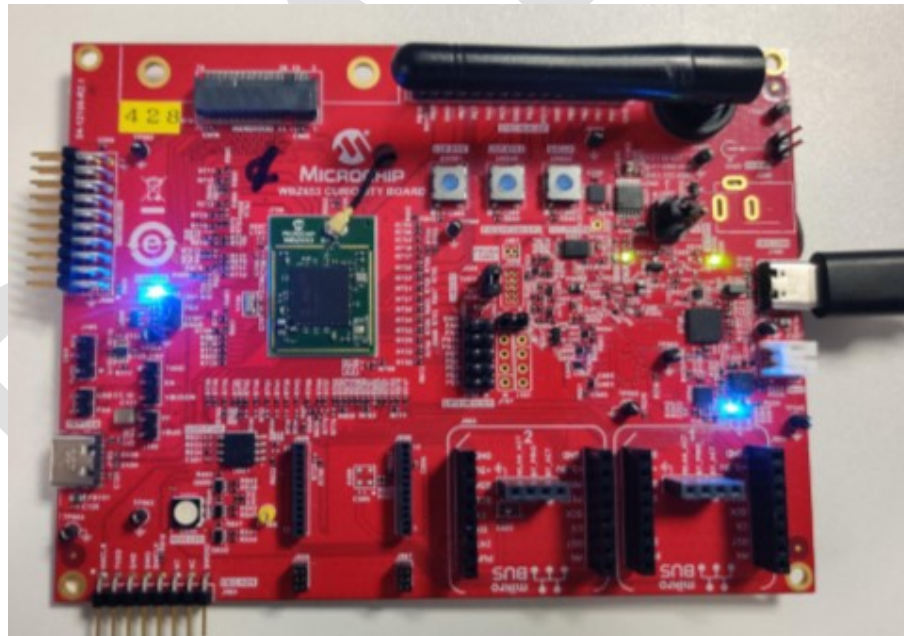
File pkob4_app_cmsis-dap.hex saved to cache.

Bootloader mode entered.

Erasing Flash ...
. . .

Application being written and verified ...
. . . . .

Application switch completed.
```



3.3.4. Switching PKOB4 Firmware from CMSIS-DAP to MPLAB

To revert an evaluation board's **PKOB4** firmware from the CMSIS-DAP implementation back to the **MPLAB (PKOB4)** firmware using a local HEX file, follow these steps:

- PKOB4_app.hex file save it to a local directory of your choice.

- Open a command prompt and navigate to the directory containing the PKOB4_app.hex file.
- Run the command “python -m pycmsisdapswitcher evalboard --source= PKOB4_app.hex”

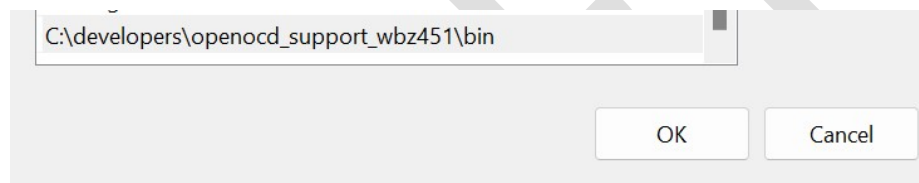
4. OpenOCD Integration with Visual Studio Code

To integrate **OpenOCD** with Visual Studio Code (VSCode) for WBZ451 development, follow the steps below:

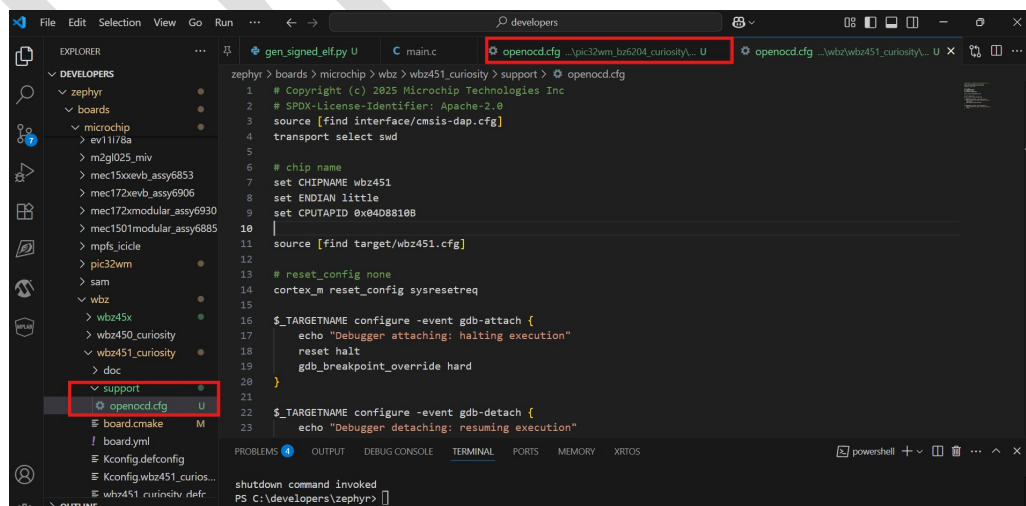
Step-1: - Extract the “openocd_support_wbz451” package and place it under C:/developers/

Step-2: - Add path C:/developers/openocd_support_wbz451/bin to your system's Environment Variables

- Open System Environment Variables → Click Environment Variables → Under System variables, find and edit the Path variable.
- Add the copied path to the list and click OK.



Step-3: - In VSCode, create a new folder named support, and inside it, add a new file named openocd.cfg as shown below. This file will contain the OpenOCD configuration settings required for debugging.



Step-4: - Copy and paste the following content into the openocd.cfg file—you can use the same configuration as shown below:

```
# Copyright (c) 2025 Microchip Technologies Inc
# SPDX-License-Identifier: Apache-2.0
source [find interface/cmsis-dap.cfg]
transport select swd

# chip name
set CHIPNAME wbz451
set ENDIAN little
set CPUTAPID 0x04D8810B

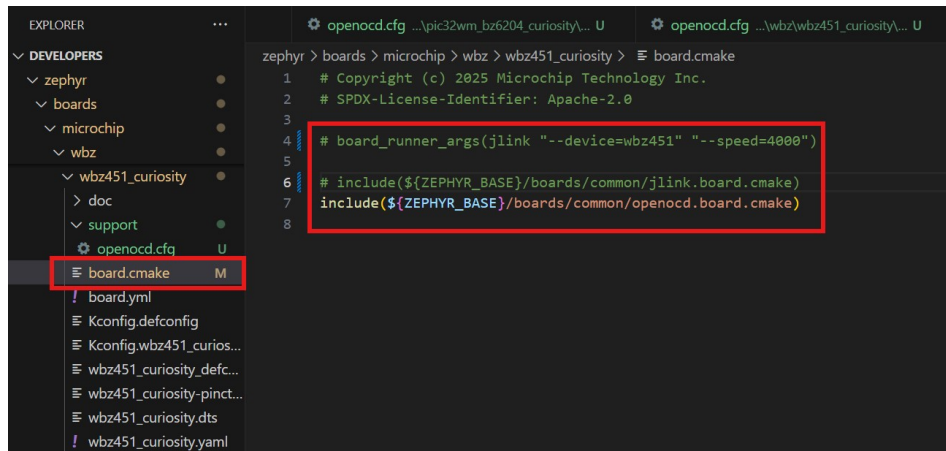
source [find target/wbz451.cfg]

cortex_m reset_config sysresetreq

$_TARGETNAME configure -event gdb-attach {
    echo "Debugger attaching: halting execution"
    reset halt
    gdb_breakpoint_override hard
}

$_TARGETNAME configure -event gdb-detach {
    echo "Debugger detaching: resuming execution"
    resume
}
```

Step-5: - Ensure the below content is present under the “board.cmake” file



The screenshot shows an IDE with two panels. The left panel is the Explorer, showing the project structure under 'DEVELOPERS' > 'zephyr' > 'boards' > 'microchip' > 'wbz' > 'wbz451_curiosity'. The file 'board.cmake' is selected and highlighted with a red box. The right panel shows the content of 'board.cmake' with line numbers 1 through 8. Lines 4, 6, 7, and 8 are highlighted with a red box. A large 'DRAFT' watermark is visible diagonally across the page.

```
zephyr > boards > microchip > wbz > wbz451_curiosity > board.cmake
1 # Copyright (c) 2025 Microchip Technology Inc.
2 # SPDX-License-Identifier: Apache-2.0
3
4 # board_runner_args(jlink "--device=wbz451" "--speed=4000")
5
6 # include(${ZEPHYR_BASE}/boards/common/jlink.board.cmake)
7 include(${ZEPHYR_BASE}/boards/common/openocd.board.cmake)
8
```

Copy and paste the following content into the “board.cmake” file.

```
# Copyright (c) 2025 Microchip Technology Inc.  
# SPDX-License-Identifier: Apache-2.0  
  
# board_runner_args(jlink "--device=wbz451" "--speed=4000")  
  
# include(${ZEPHYR_BASE}/boards/common/jlink.board.cmake)  
include(${ZEPHYR_BASE}/boards/common/openocd.board.cmake)
```

Step-6: - Ensure the following folders are present under the C:/developers/ directory:

- “Zephyr-sdk-0.17.0”
- “openocd_support_wbz451”
- “Zephyrproject”

5. Steps to Flashing/Programming

5.1. WBZ451 Device

- Change directory to “**cd zephyr**” and build the application as shown below
 - Build Command: “**West build -p always -b wbz451_curiosity .\samples\basic\blinky**”

```

PRO Focus folder in explorer (ctrl + click) | TERMINAL | PORTS | MEMORY | ELF RIPPER
PS C:\developers\zephyrproject\zephyr> west build -p always -b wbz451_curiosity .\samples\basic\blinky\

```

- After a successful build, connect the WBZ451 device and run the “**west flash**” command to program the device, as shown below:

```

-- Zephyr version: 4.0.0 (C:/developers/zephyrproject/zephyr), build: c1f783adc7cb
[117/117] Linking C executable zephyr\zephyr.elf
Memory region      Used Size  Region Size  %age Used
ROMSTART_REGION:    224 B      1 KB        21.88%
FLASH:              14616 B    1 MB         1.39%
RAM:                 4224 B    128 KB        3.22%
IDT_LIST:            0 GB      32 KB         0.00%
Generating files from C:/developers/zephyrproject/zephyr/build/zephyr/zephyr.elf for board: wbz451_curiosity
PS C:\developers\zephyrproject\zephyr> west flash

```

- Ensure the flash process is completed successfully, as shown below:

```

TargetName      Type      Endian TapName      State
-----
0* wbz451.cpu    cortex_m  little wbz451.cpu    unknown
[wbz451.cpu] halted due to breakpoint, current mode: Thread
xPSR: 0x01000000 pc: 0x01000730 msp: 0x20001040
Info : WBZ451: Write completed
Info : Padding image section 1 at 0x00045ea0 with 28 bytes
Info : Padding image section 2 at 0x00045ec0 with 28 bytes
Info : Padding image section 3 at 0x00045ee0 with 168 bytes
Info : Padding image section 4 at 0x00045fa0 with 28 bytes
Info : Padding image section 5 at 0x00045fc0 with 28 bytes
Warn : Adding extra erase range, 0x00045c00 .. 0x00045e87
Info : WBZ451: Write completed
Info : WBZ451: Write completed
auto erase enabled
wrote 16760 bytes from file C:/developers/zephyrproject/zephyr/build/zephyr/zephyr.hex in 1.391416s (11.763 KiB/s)
shutdown command invoked
PS C:\developers\zephyrproject\zephyr>

```

5.2. WBZ653 Device

- Change directory to “**cd zephyr**” and build the application as shown below
 - Build Command: “**west build -p always -b pic32wm_bz6204_curiosity .\samples\basic\blinkv**”

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  MEMORY  XRTOS

-- Build files have been written to: C:/developers/zephyr/build
-- west build: building application
[1/119] Generating include/generated/zephyr/version.h
-- Zephyr version: 4.0.0 (C:/developers/zephyr), build: 54fd5047f6f0
[117/119] Linking C executable zephyr\zephyr.elf
Memory region      Used Size  Region Size  %age Used
  FLASH:           16072 B    2096640 B    0.77%
   RAM:             4288 B      512 KB    0.82%
  IDT_LIST:          0 GB       32 KB    0.00%
Generating files from C:/developers/zephyr/build/zephyr/zephyr.elf for board: pic32wm_bz6204_curiosity
[119/119] Converting zephyr.hex to zephyr.signed.hex
PS C:\developers\zephyr> west build -p always -b pic32wm_bz6204_curiosity .\samples\basic\blinkv

```

- After a successful build, connect the WBZ653 device and run the “**west flash --hex-file build/zephyr/zephyr_signed.hex**” command to program the device, as shown below:

```

Generating files from C:/developers/zephyr/build/zephyr/zephyr.elf for board: pic32wm_bz6204_curiosity
[119/119] Converting zephyr.hex to zephyr.signed.hex
PS C:\developers\zephyr> west flash --hex-file build/zephyr/zephyr_signed.hex

```

- Ensure the flash process is completed successfully, as shown below:

```

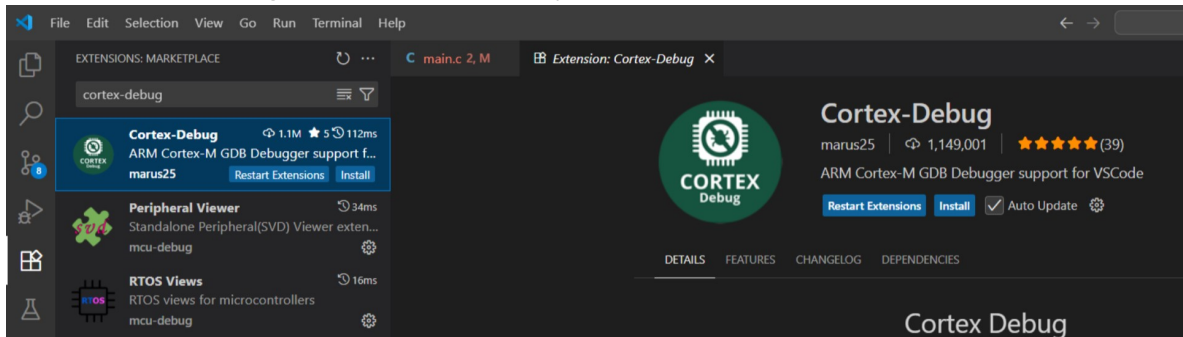
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  MEMORY  XRTOS

Info : Writing row to RAM buffer at 0x20000000
Info : Programming row at 0x01003c00
Info : Reading row at 0x01004000 for modify
Info : Writing row to RAM buffer at 0x20000000
Info : Programming row at 0x01004000
Info : WBZ653: Write complete
auto erase enabled
wrote 17784 bytes from file build/zephyr/zephyr_signed.hex in 1.814170s (9.573 KiB/s)
Info : SWD DPIDR 0x2ba01477
Error: Failed to write memory at 0xe00ed10
shutdown command invoked
PS C:\developers\zephyr>

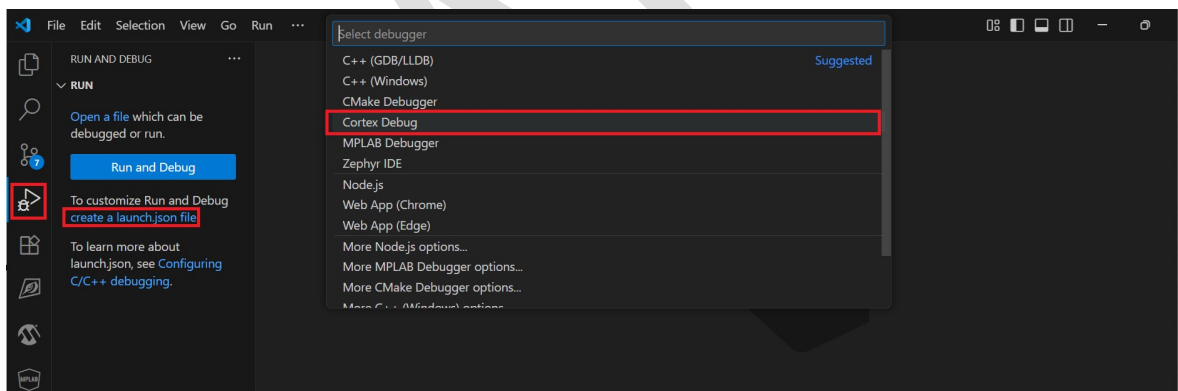
```

6. Steps to Debugging the WBZ451 & WBZ653 Device

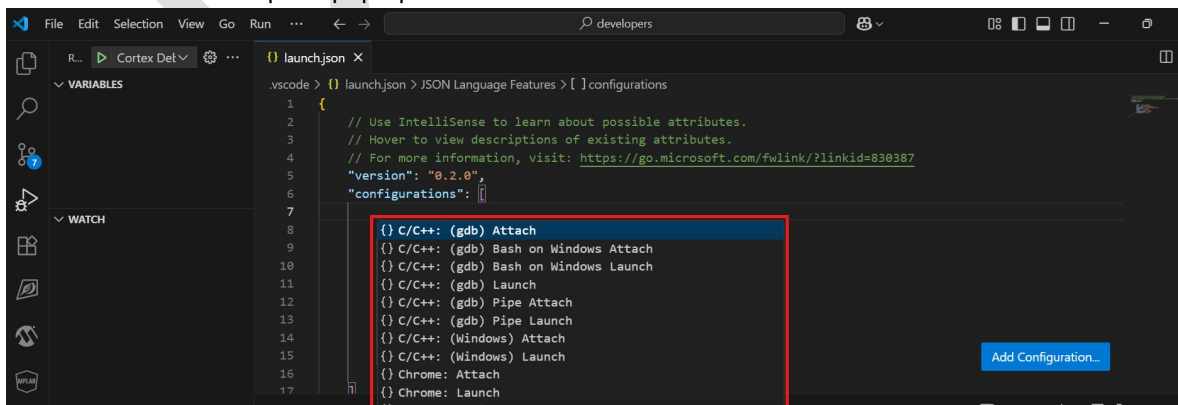
- Make sure you have built the application either for WBZ451 or WBZ653 before starting the debugging process.
- Make sure cortex-debug extension is installed in your local machine



- Click on the Run and Debug icon on the left. If the launch.json and tasks.json files are already present, VS Code will automatically start the debug session. Ensure that the contents of both files match the configurations provided in the steps below. If these files are not present, VS Code will prompt you to create a new launch.json file—select the "Cortex Debug" debugger option when prompted, as shown below.



- Press **Esc** to close or skip the pop-up window.



- Replace the content inside the launch.json file with the configuration shown below:

```
{
  "version": "2.0.0",
  "configurations": [
    {
      "name": "Debug WBZ653",
      "type": "cortex-debug",
      "request": "attach",
      "servertype": "openocd",
      "cwd": "C:\\developers\\zephyr\\",
      "executable": "C:/developers/zephyr/build/zephyr/zephyr.elf",
      "device": "WBZ653",
      "configFiles": [
        "interface/cmsis-dap.cfg",
        "target/wbz653.cfg"
      ],
      "gdbPath": "C:/developers/zephyr-sdk-0.17.0/arm-zephyr-eabi/bin/arm-zephyr-eabi-gdb.exe",
      "preLaunchTask": "flash_wbz653_signed_hex",
      "postRestartCommands": [
        "symbol-file C:/developers/zephyr/build/zephyr/zephyr.elf",
        "monitor reset halt",
        "break main"
      ],
      "showDevDebugOutput": "none"
    }
  ]
}
```

```
},  
  
{  
  "name": "Debug WBZ451",  
  "type": "cortex-debug",  
  "request": "attach",  
  "servertype": "openocd",  
  "cwd": "C:\\developers\\zephyr\\",  
  "executable": "C:/developers/zephyr/build/zephyr/zephyr.elf",  
  "device": "WBZ451",  
  "configFiles": [  
    "interface/cmsis-dap.cfg",  
    "target/wbz451.cfg"  
  ],  
  "gdbPath": "C:/developers/zephyr-sdk-0.17.0/arm-zephyr-eabi/bin/arm-zephyr-eabi-gdb.exe",  
  "preLaunchTask": "flash_wbz451_hex",  
  "postRestartCommands": [  
    "symbol-file C:/developers/zephyr/build/zephyr/zephyr.elf",  
    "monitor reset halt",  
    "break main"  
  ],  
  "showDevDebugOutput": "none"  
}  
]  
}
```

```

launch.json X tasks.json X
.vscode > {} launch.json > Launch Targets > {} Debug WBZ451
1  {
2      "version": "2.0.0",
3      "configurations": [
4          {
5              "name": "Debug WBZ653",
6              "type": "cortex-debug",
7              "request": "attach",
8              "servertype": "openocd",
9              "cwd": "C:\\\\developers\\\\zephyr\\\\",
10             "executable": "C:/developers/zephyr/build/zephyr/zephyr.elf",
11             "device": "WBZ653",
12             "configFiles": [
13                 "interface/cmsis-dap.cfg",
14                 "target/wbz653.cfg"
15             ],
16             "gdbPath": "C:/developers/zephyr-sdk-0.17.0/arm-zephyr-eabi/bin/arm-zephyr-eabi-gdb.exe",
17             "preLaunchTask": "flash_wbz653_signed_hex",
18             "postRestartCommands": [
19                 "symbol-file C:/developers/zephyr/build/zephyr/zephyr.elf",
20                 "monitor reset halt",
21                 "break main"
22             ],
23             "showDevDebugOutput": "none"
24         },
25         {
26             "name": "Debug WBZ451",
27             "type": "cortex-debug",
28             "request": "attach",
29             "servertype": "openocd",
30             "cwd": "C:\\\\developers\\\\zephyr\\\\",
31             "executable": "C:/developers/zephyr/build/zephyr/zephyr.elf",
32             "device": "WBZ451",
33             "configFiles": [
34                 "interface/cmsis-dap.cfg",
35                 "target/wbz451.cfg"
36             ],
37             "gdbPath": "C:/developers/zephyr-sdk-0.17.0/arm-zephyr-eabi/bin/arm-zephyr-eabi-gdb.exe",
38             "preLaunchTask": "flash_wbz451_hex",
39             "postRestartCommands": [
40                 "symbol-file C:/developers/zephyr/build/zephyr/zephyr.elf",
41                 "monitor reset halt",
42                 "break main"
43             ],
44             "showDevDebugOutput": "none"
45         }
46     ]
47 }
48

```


- Create a new **tasks.json** file inside the .vscode folder, and copy the following content into it:

```
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "flash_wbz653_signed_hex",
      "type": "shell",
      "command": "openocd",
      "args": [
        "-f", "interface/cmsis-dap.cfg",
        "-f", "target/wbz653.cfg",
        "-c", "init",
        "-c", "reset halt",
        "-c", "program C:/developers/zephyr/build/zephyr/zephyr_signed.hex reset exit"
      ],
      "problemMatcher": [],
      "group": {
        "kind": "build",
        "isDefault": true
      },
    },
    {
      "label": "flash_wbz451_hex",
      "type": "shell",
```

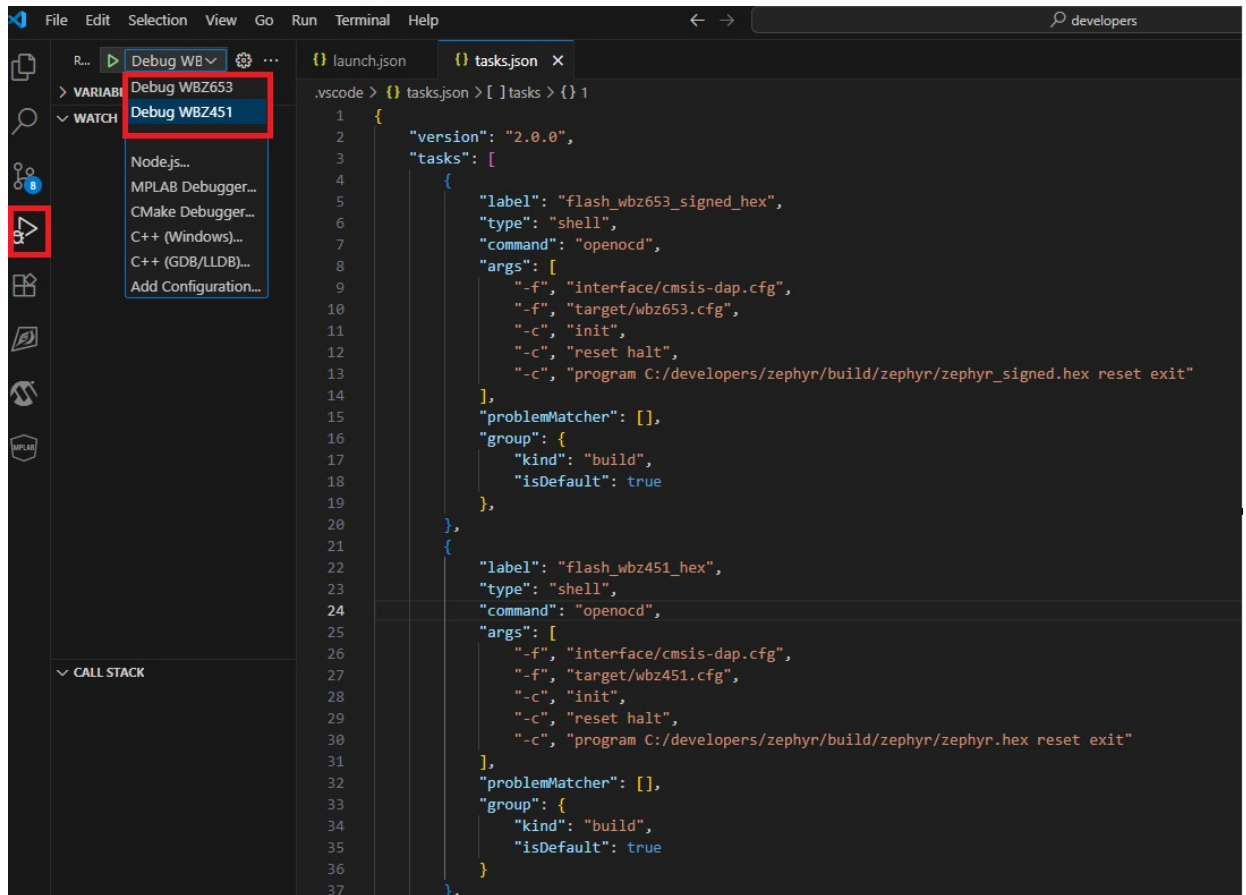
```
"command": "openocd",  
  
"args": [  
  "-f", "interface/cmsis-dap.cfg",  
  "-f", "target/wbz451.cfg",  
  "-c", "init",  
  "-c", "reset halt",  
  "-c", "program C:/developers/zephyr/build/zephyr/zephyr.hex reset exit"  
],  
"problemMatcher": [],  
"group": {  
  "kind": "build",  
  "isDefault": true  
}  
},  
]  
}
```

```

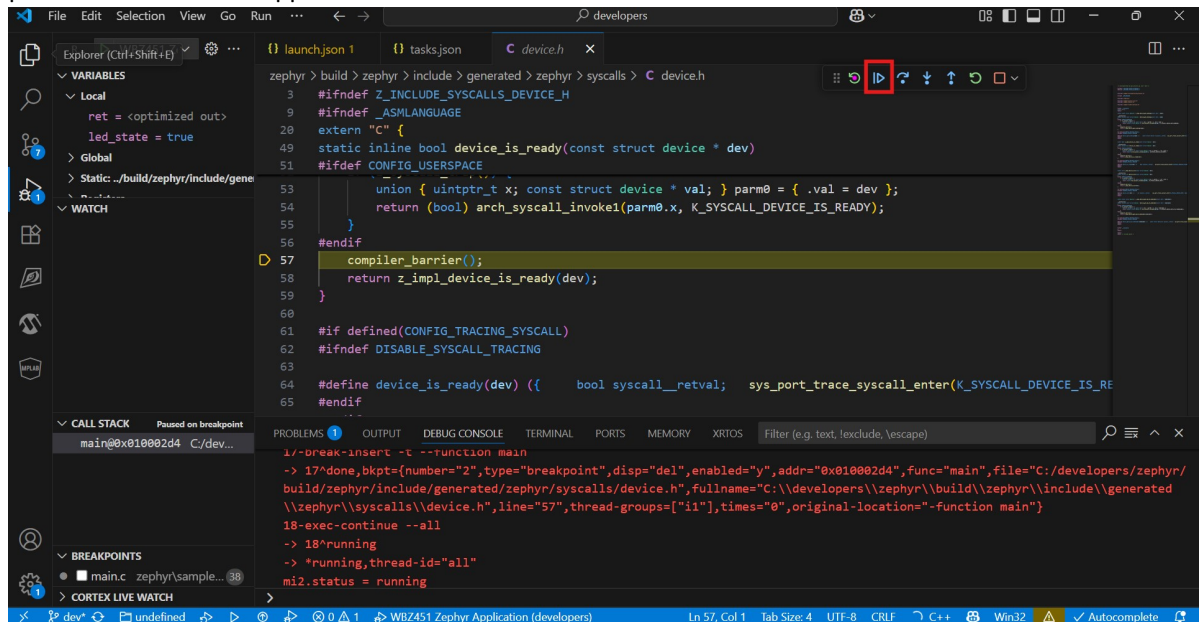
{} launch.json {} tasks.json X
.vscode > {} tasks.json > [ ] tasks > {} 1
1  {
2      "version": "2.0.0",
3      "tasks": [
4          {
5              "label": "flash_wbz653_signed_hex",
6              "type": "shell",
7              "command": "openocd",
8              "args": [
9                  "-f", "interface/cmsis-dap.cfg",
10                 "-f", "target/wbz653.cfg",
11                 "-c", "init",
12                 "-c", "reset halt",
13                 "-c", "program C:/developers/zephyr/build/zephyr/zephyr_signed.hex reset exit"
14             ],
15             "problemMatcher": [],
16             "group": {
17                 "kind": "build",
18                 "isDefault": true
19             },
20         },
21         {
22             "label": "flash_wbz451_hex",
23             "type": "shell",
24             "command": "openocd",
25             "args": [
26                 "-f", "interface/cmsis-dap.cfg",
27                 "-f", "target/wbz451.cfg",
28                 "-c", "init",
29                 "-c", "reset halt",
30                 "-c", "program C:/developers/zephyr/build/zephyr/zephyr.hex reset exit"
31             ],
32             "problemMatcher": [],
33             "group": {
34                 "kind": "build",
35                 "isDefault": true
36             },
37         },
38     ],
39 }
40
41
42

```

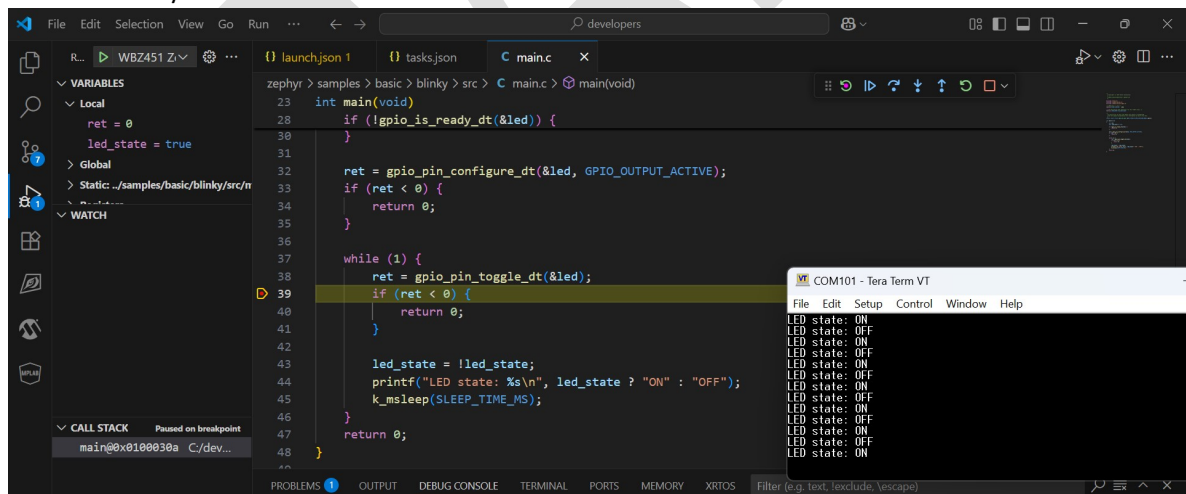
- Connect the WBZ451 or WBZ653 Curiosity board, then click the Run and Debug icon to start the debugging process. You will see a drop-down menu where you can choose between WBZ451 and WBZ653 debug options.



- Make sure the debugger hits the breakpoint as shown below, then press the **Continue** button to proceed to the main.c application.



- After pressing the **Continue** button, the application will load, allowing you to set breakpoints in the code and verify them as shown below.



7. References / Resources:

- TBD

8. Abbreviations:

Abbreviations	Full Form
OpenOCD	Open On-Chip Debugger
SWD	Serial Wire Debug
PKOB4	PICkit On-Board v4
CMSIS-DAP	Cortex Microcontroller Software Interface Standard - Debug Access Port
VSCode	Visual Studio Code
JSON	JavaScript Object Notation (used for configuration files)
PATH	Environment Variable for Executable File Locations

9. Document History:

Version	Date	Author	Description of Changes
v0.2	22 – Aug - 2025	Dinesh Arasu	Draft Version