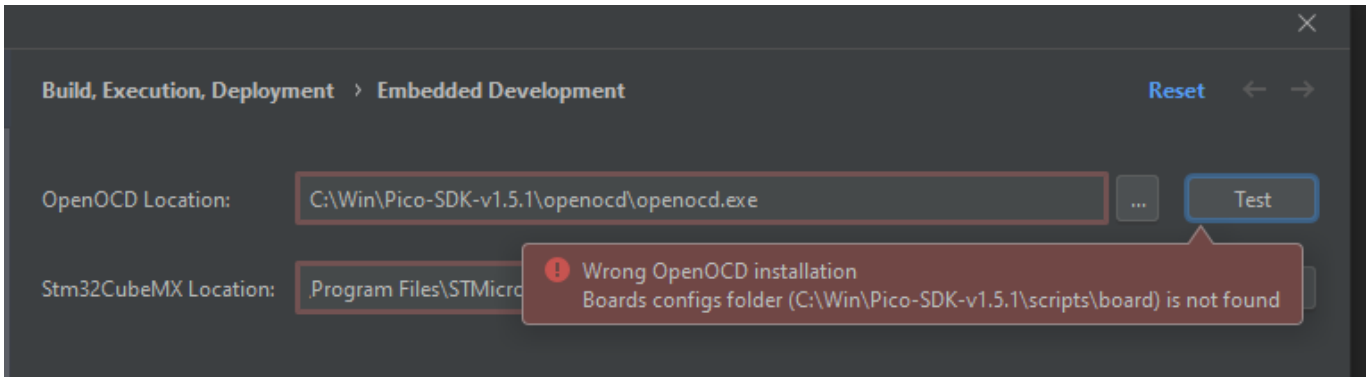
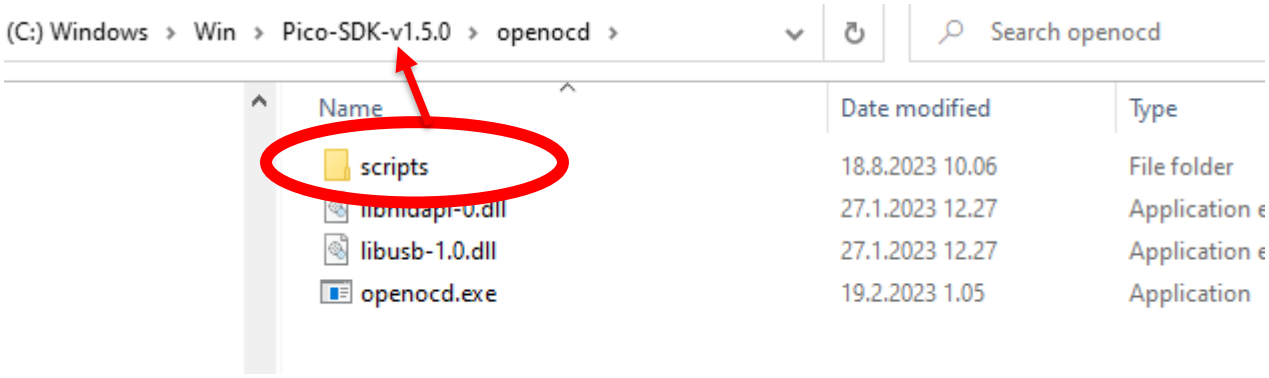
Install pico-sdk to a folder that does not contain any spaces or special characters. Try to keep path names as short as possible.

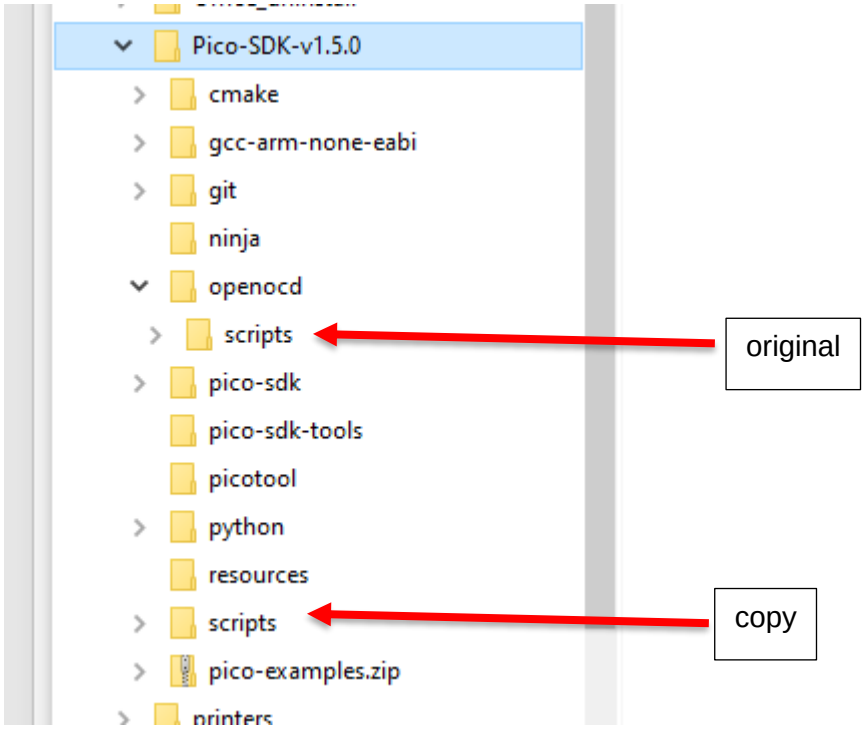
Go to Settings > Build, Execution, Deployment > Embedded development and set the path to open OCD executable.

There is an issue with OpenOCD integration. OpenOCD looks for scripts in different folder than where they are installed to. After entering the path to executable, pressing Test will show an error message with the path where scripts folder needs to be placed.



Copy scripts folder from openocd-folder to pico installation folder. Do not remove the original scripts folder.





Edit the copied scripts/board/pico-debug.cfg

pico-debug.cfg

```
1  # SPDX-License-Identifier: GPL-2.0-or-later
2  # pico-debug is a virtual CMSIS-DAP debug adapter
3  # it runs on the very same RP2040 target being de
4  # https://github.com/majbthrd/pico-debug
5
6  source [find interface/cmsis-dap.cfg]
7  adapter speed 4000
8
9  set CHIPNAME rp2040
10 source [find target/rp2040-core0.cfg]
11
```

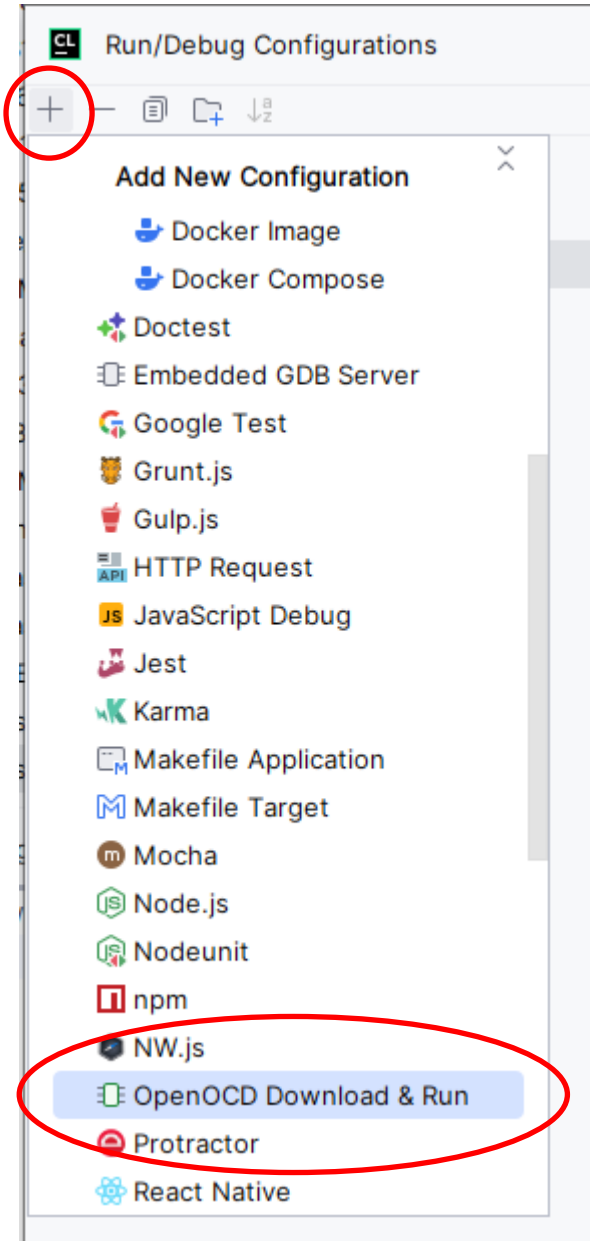
Original

pico-debug.cfg

```
1  # SPDX-License-Identifier: GPL-2.0-or-later
2  # pico-debug is a virtual CMSIS-DAP debug adapter
3  # it runs on the very same RP2040 target being debugge
4  # https://github.com/majbthrd/pico-debug
5
6  source [find interface/cmsis-dap.cfg]
7  adapter speed 4000
8
9  set CHIPNAME rp2040
10 source [find target/rp2040.cfg]
11
```

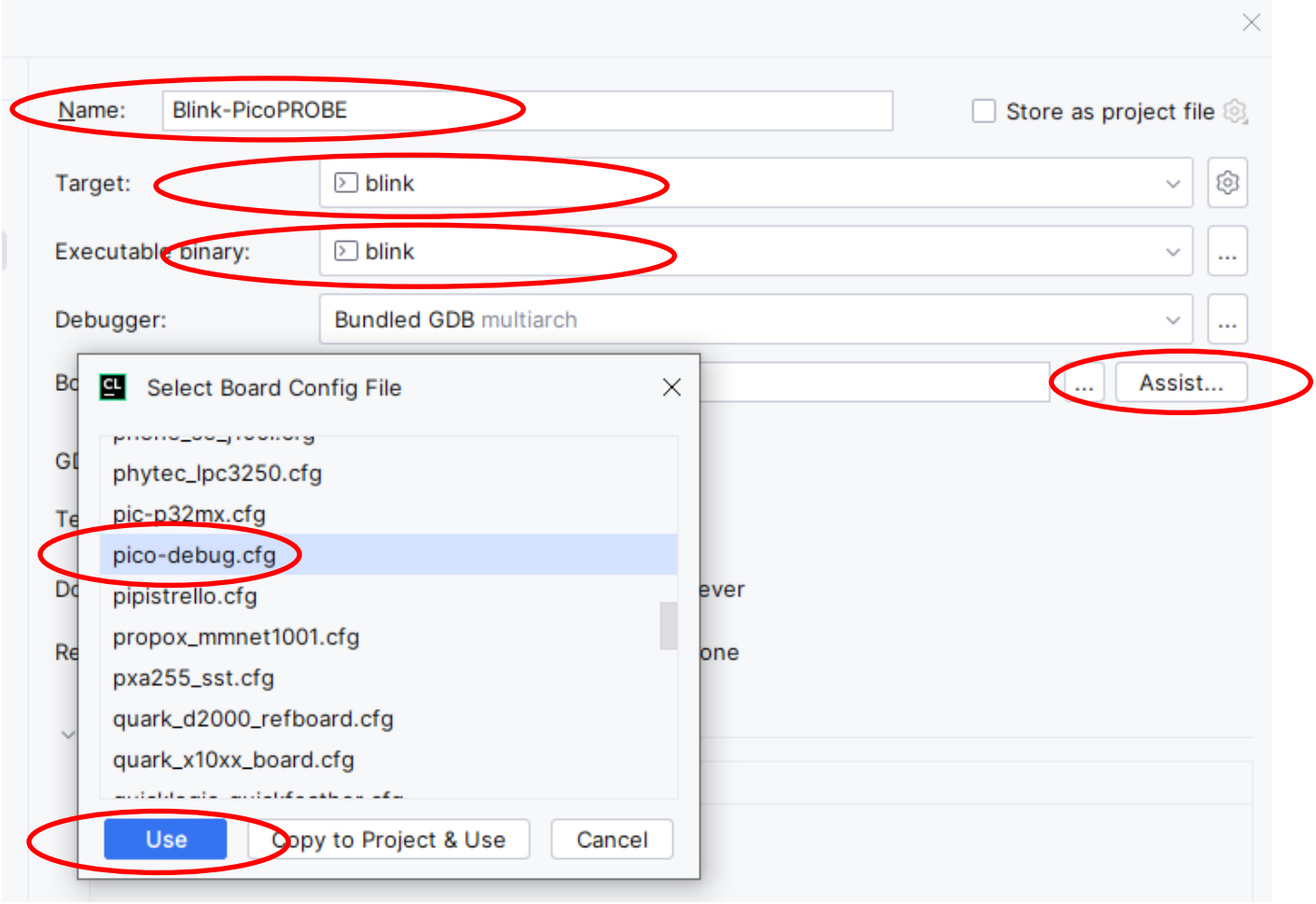
Edited

Go to Run>Edit configurations and click on + at top left and select **OpenOCD Download & Run**

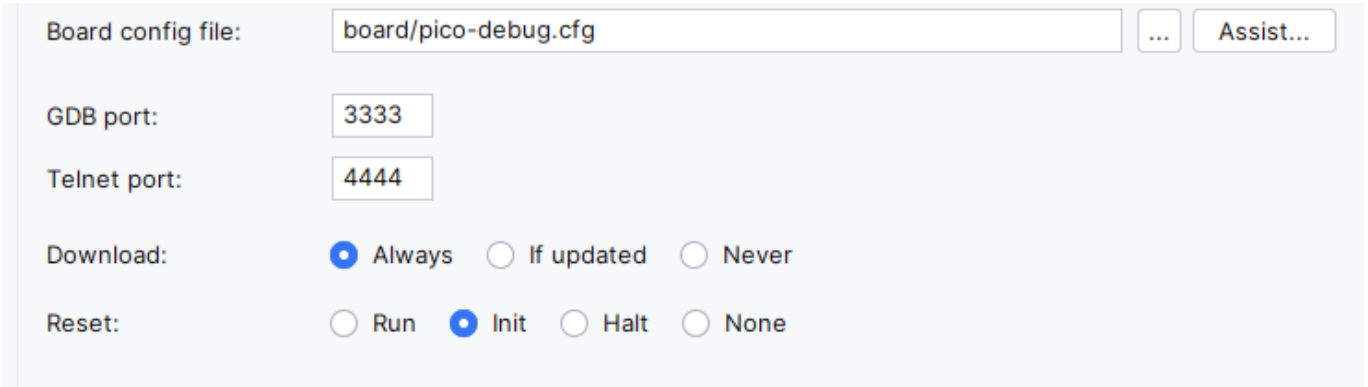


Name your configuration in a way that it is easy to remember that it requires PicoProbe. Then choose target to debug from the dropdown menu.

Then Select board config file – click assist and fin pico-debug.cfg

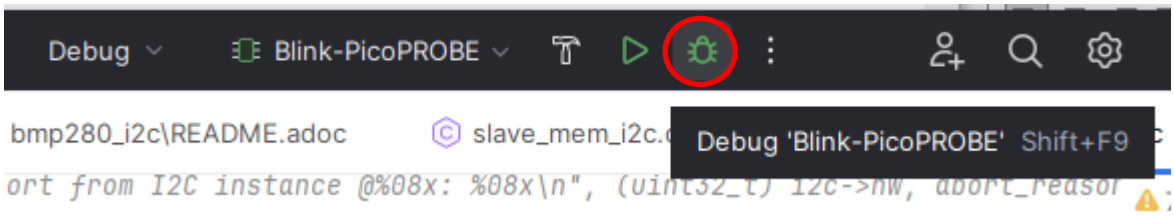


Set download to Always



You are done!

Select your new target from dropdown menu and click debug.

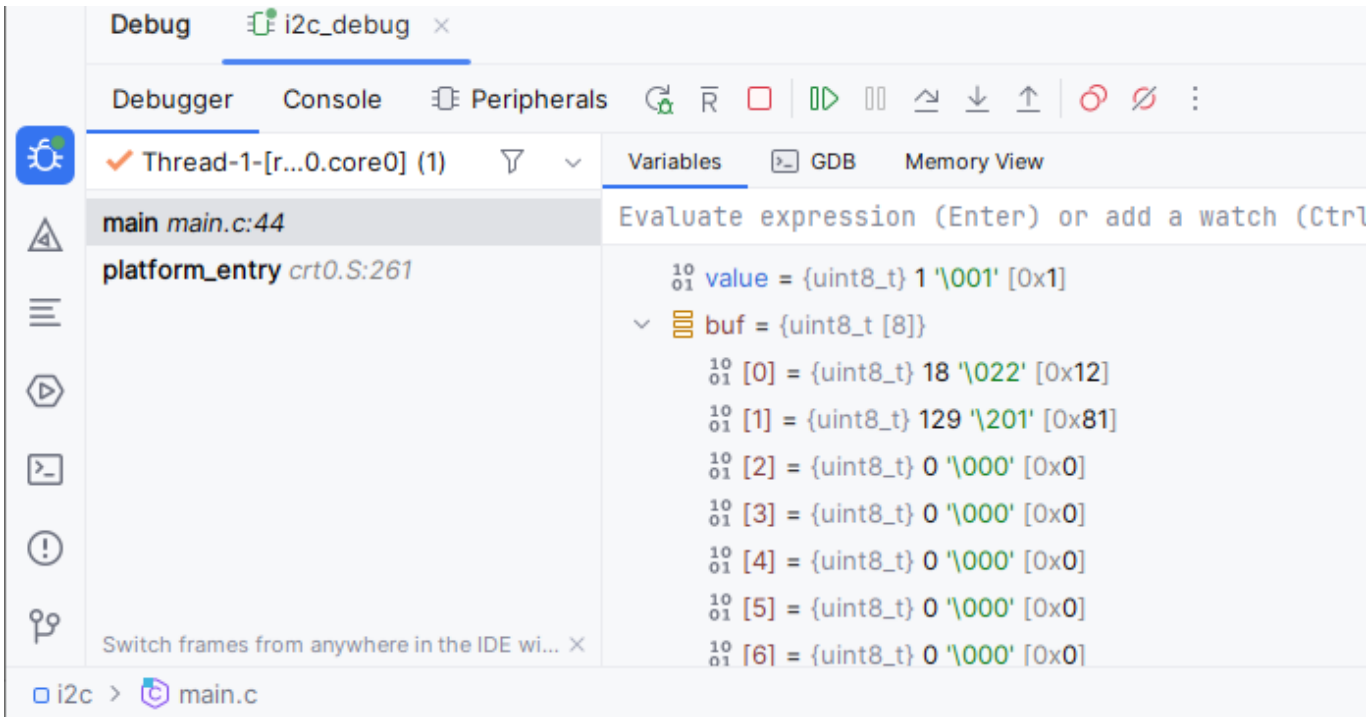


There is no default break point so if you don't set one yourself the program will start immediately after download.

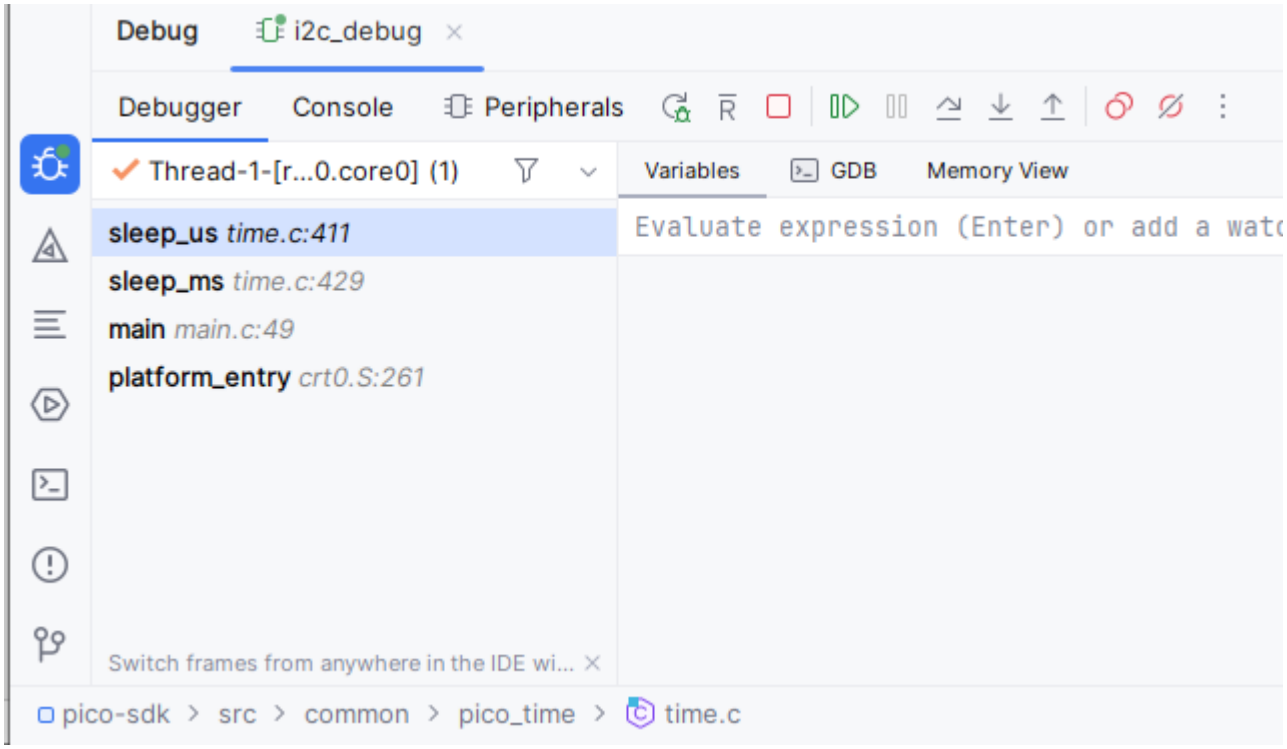
If you just want to run your program without debugging press the run button.

```
10  int main() {
11
12      const uint led_pin = 22;
13
14      // Initialize LED pin
15      gpio_init( gpio: led_pin);
16      gpio_set_dir( gpio: led_pin,  out: GPIO_OUT);
17
18      // Initialize chosen serial port
19      stdio_init_all();
20      printf("\r\nBoot\r\n");
```

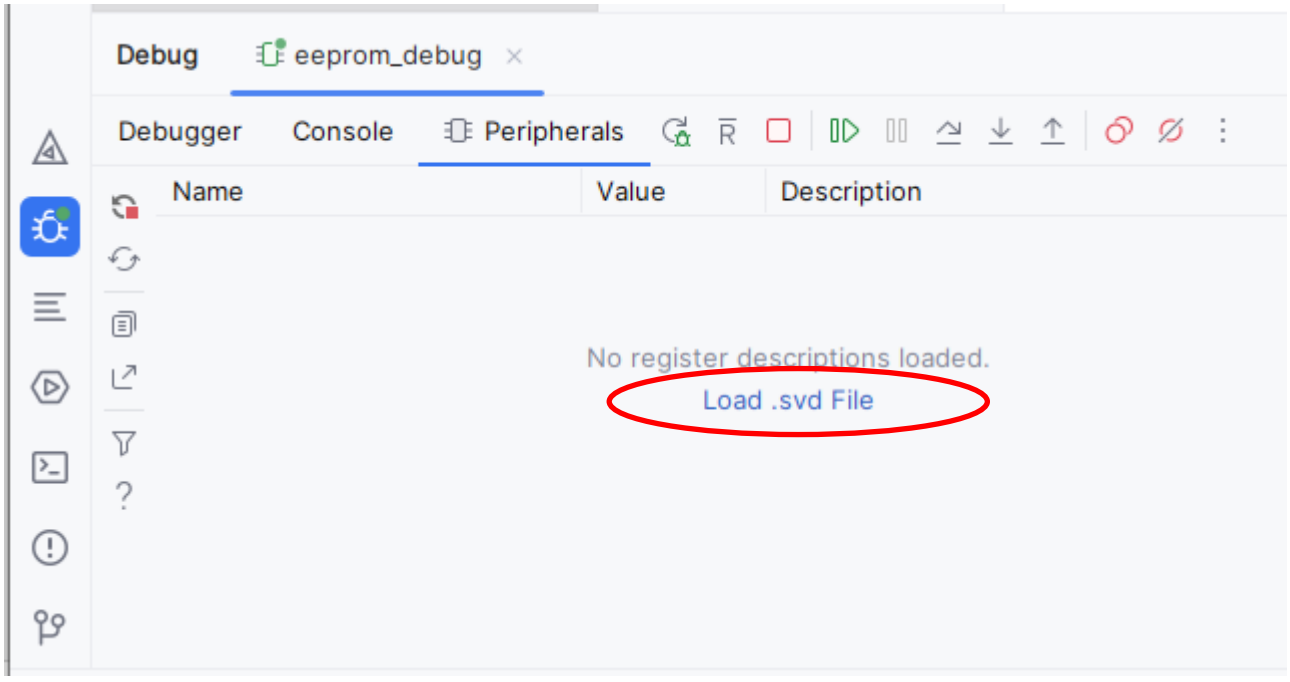
When your program is paused you can view variables.



You also see stack trace that shows the call chain of functions.

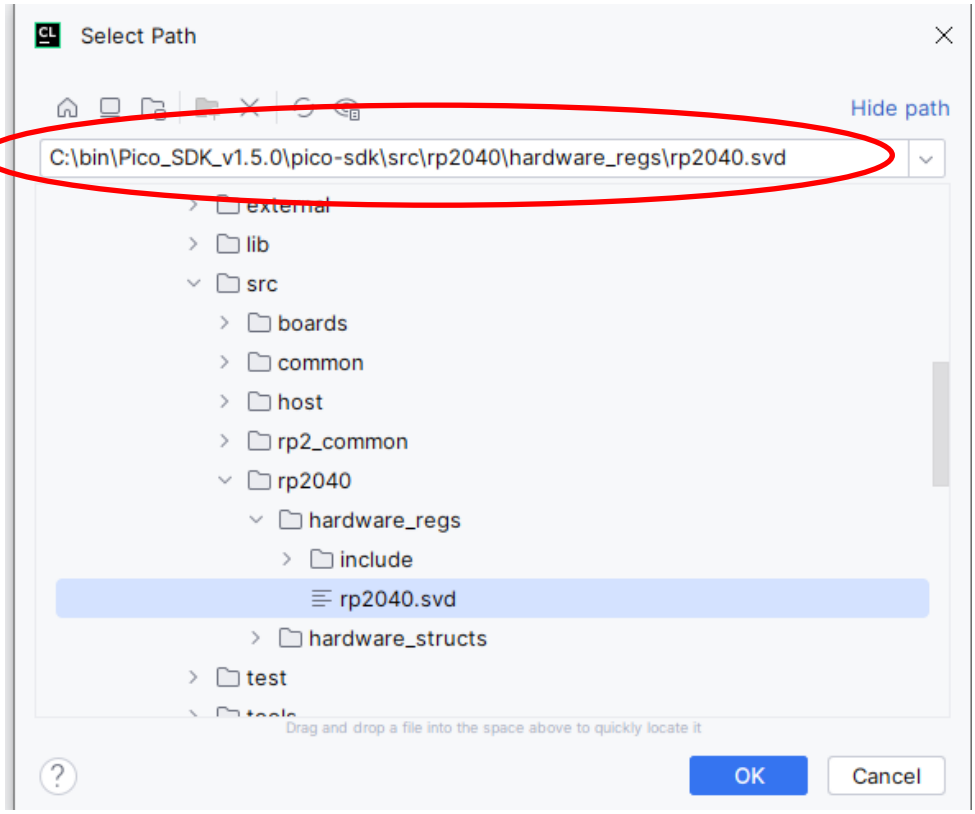


When you switch to Peripheral view you can load .svd file that tells debugger what peripherals are available and how their registers are mapped into memory.

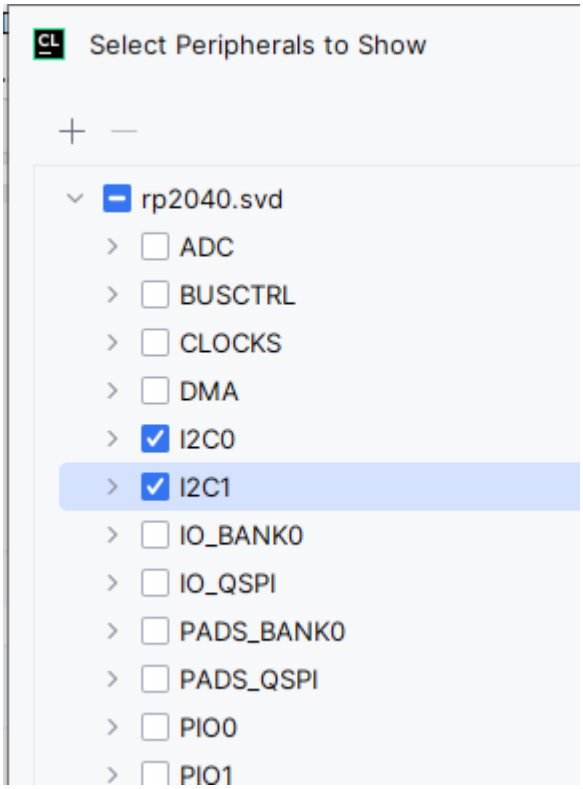


Click load .svd file.

The file comes with PicoSDK. Navigate to the file, select it, and click OK.



Then you can choose which peripherals to view.



If you want make changes to the selection, click configure button to open the selection window.