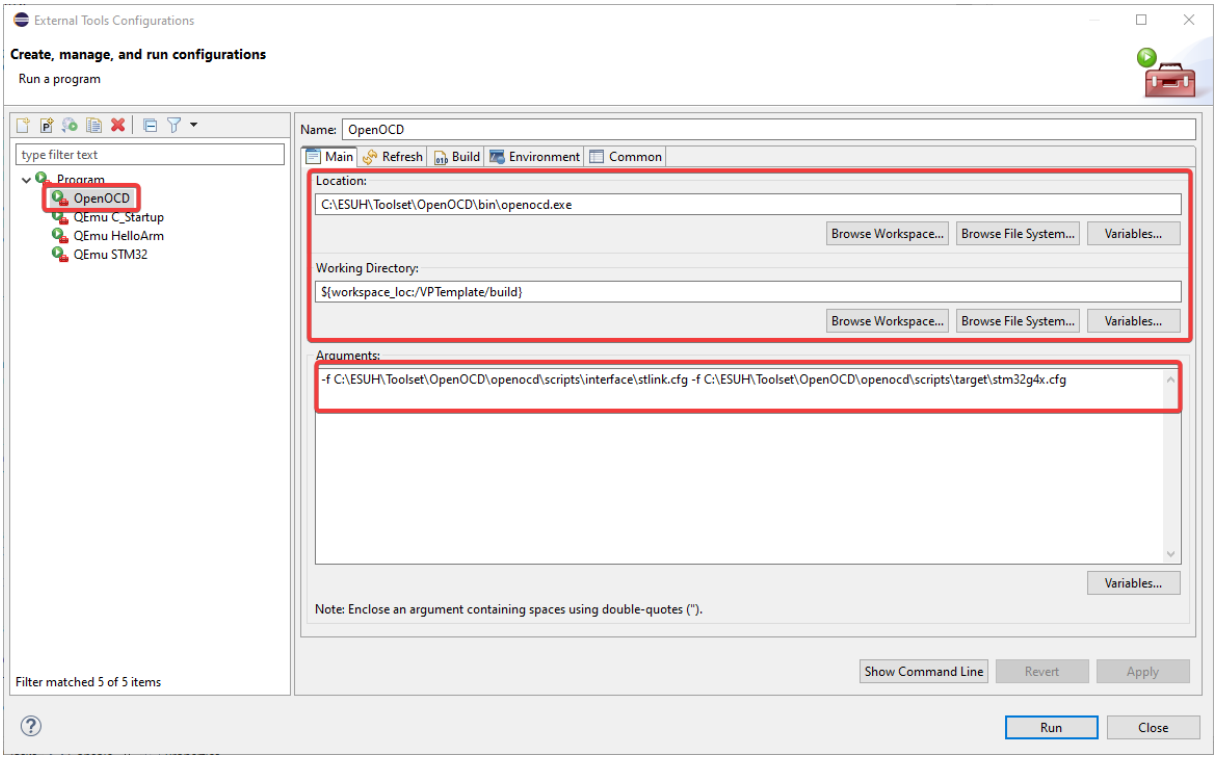
Debugging with Eclipse and OpenOCD

To flash a program onto the microcontroller, a specific “Debug-Bridge” between the PC, JTAG Interface and Microcontroller is needed. This bridge is provided by a tool call OpenOCD. This tool also serves as Debugger-Connector for GDB on one side and translates the debug commands from GDB to specific JTAG communication with the microcontroller.

External Tool Configuration for OpenOCD

Create a new “External Tool Configuration” in Eclipse.



In the configuration window, the Location, Working Directory and some Arguments must be provided. Depending on the operating system (Window/Linux) the tool location depends on your installation.

For Linux the tool is usually installed in `/usr/bin/`

In the Location field, the absolute path and the program filename must be specified. For Linux this is

`/usr/bin/openocd`

Additionally, the Working Directory must be specified. This should be done via the “Browse Workspace” button. It must be ensured, that the correct project is selected and “build” is selected as Working Directory.

The Arguments for the tool must specify some additional options for the used Hardware-Interface and the Target-Controller. In our case this is

Hardware-Interface: stlink.cfg

Target-Controller: stm32g4x.cfg

It is important to specify the absolute path to those configuration files. For Linux this is:

`/usr/share/openocd/scripts/interface/stlink.cfg`

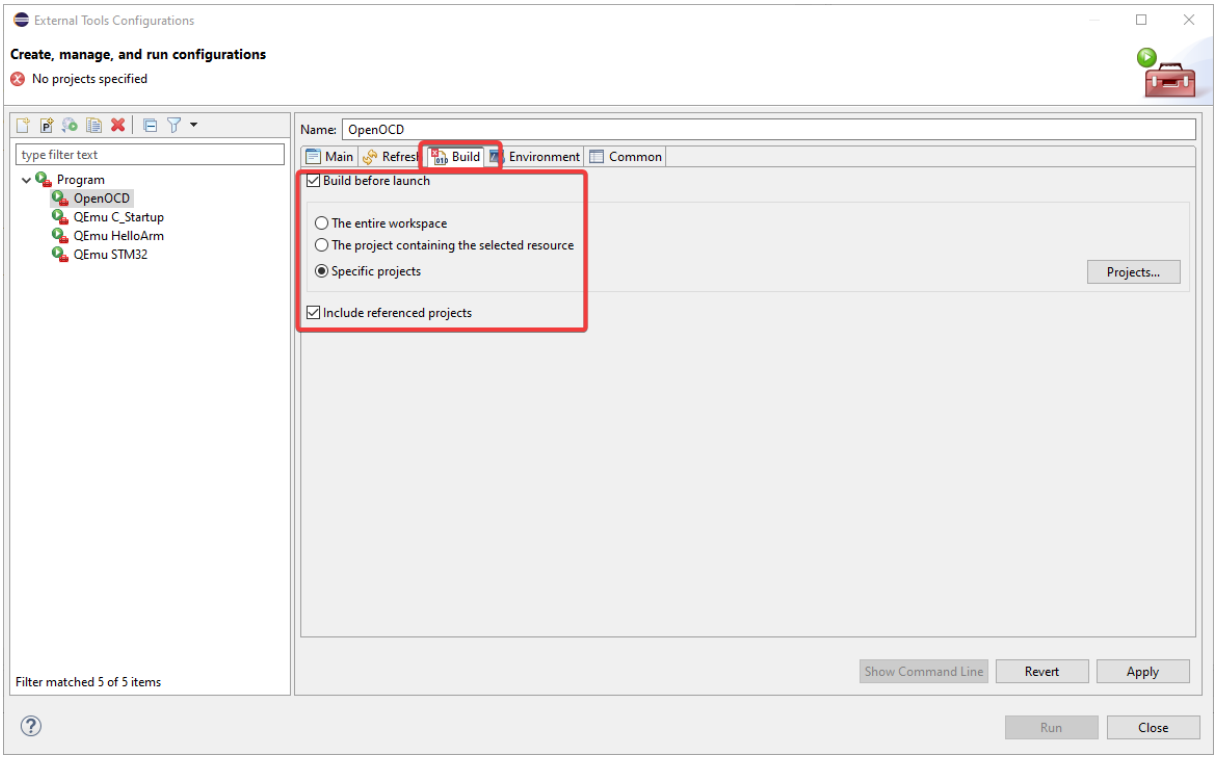
`/usr/share/openocd/scripts/target/stm32g4x.cfg`

To summarize, the complete command line for OpenOCD on Linux looks like the following:

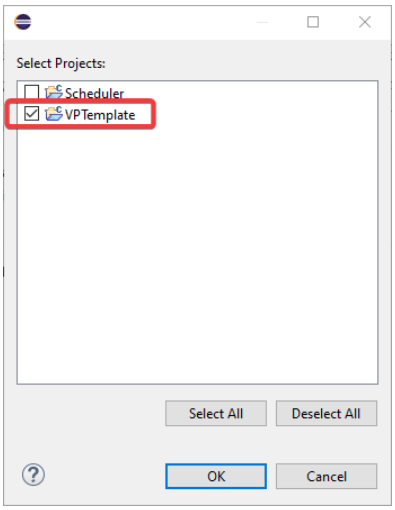
-f `/usr/share/openocd/scripts/interface/stlink.cfg`

-f `/usr/share/openocd/scripts/target/stm32g4x.cfg`

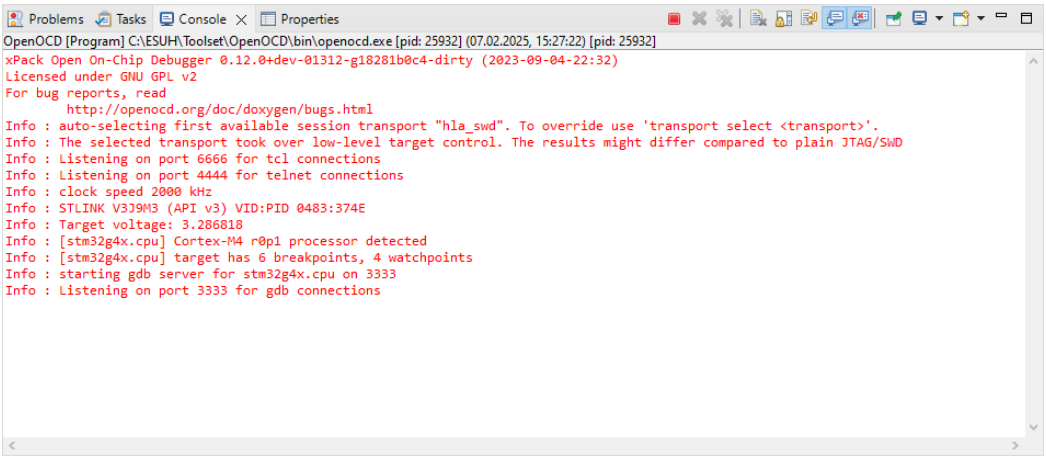
For an optimized Launch Configuration, additional setting in the “Build” tab can be specified



If the option “Build before Launch” is selected, Eclipse triggers a new Build before OpenOCD is launched and this ensures the project, which gets flashed onto the device, is up-to-date. To avoid building of unnecessary projects, it is recommended to selected the sub option “Specific Projects” and select via the “Projects…” Button only the project of which also the Working Directory has been selected.



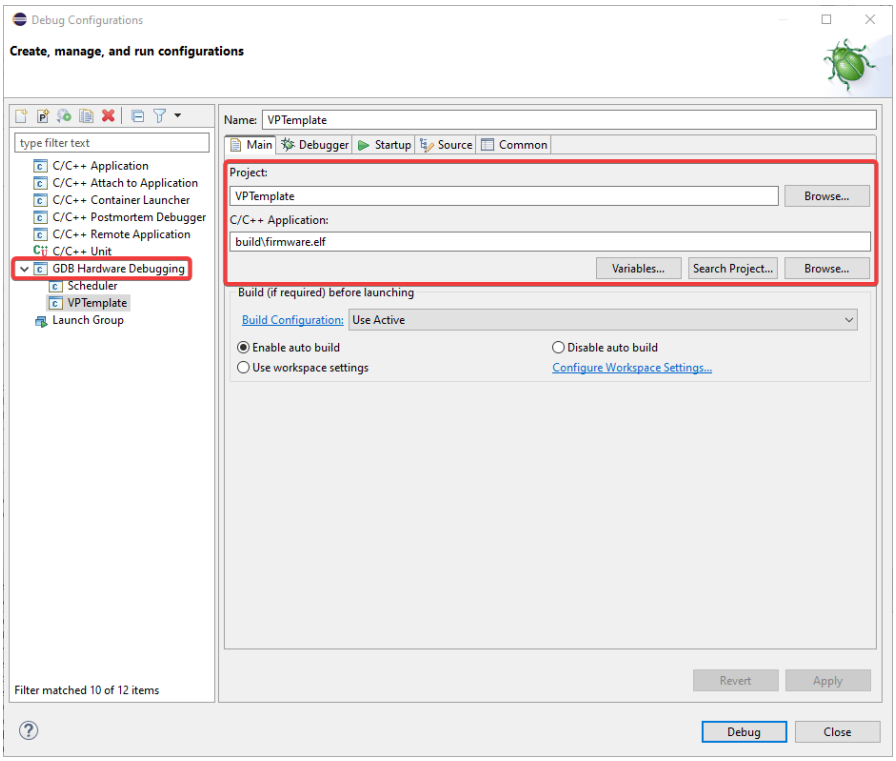
After this configuration, the tool OpenOCD could be started. It must be ensured, that the Nucleo-Board is connected via USB with the Host-PC before OpenOCD is started. If everything works as expected, the Eclipse Console Output should show something like the following screenshot



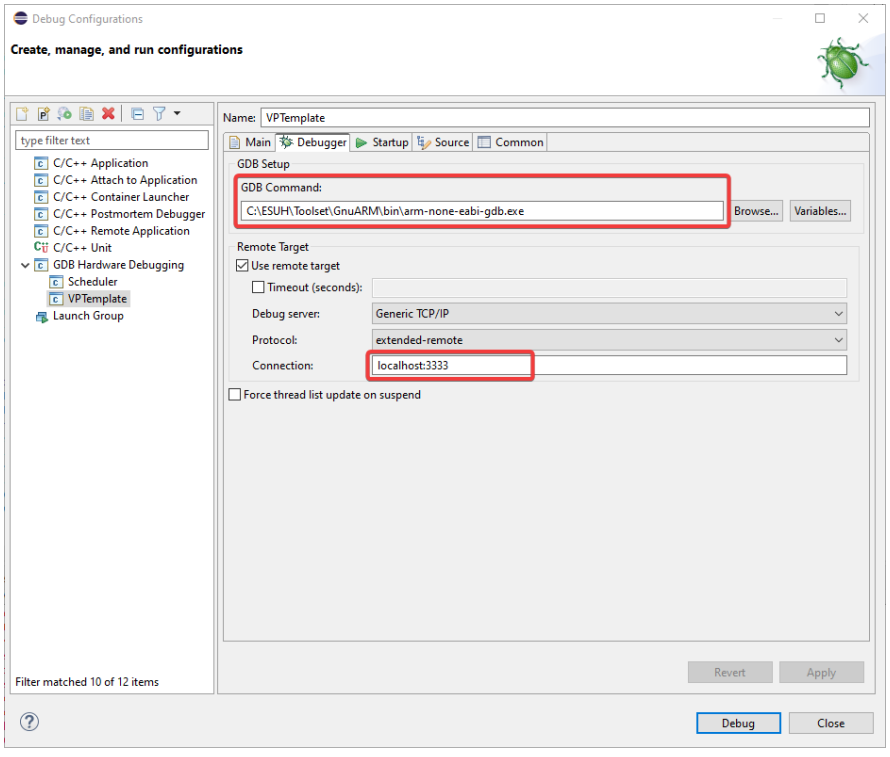
Debug Configuration for OpenOCD

To use the OpenOCD bridge for debugging on the real hardware device, the Debug-Configuration must be configured. There are a couple of important differences compared to the QEmu Debug Configuration.

First create a new “GDB Hardware Debugging” Debug configuration



In this Debug-Configuration, the project and the C/C++ Application Binary (ELF) file must be specified. Additionally, on the “Debugger” tab, some debugger settings must be specified-

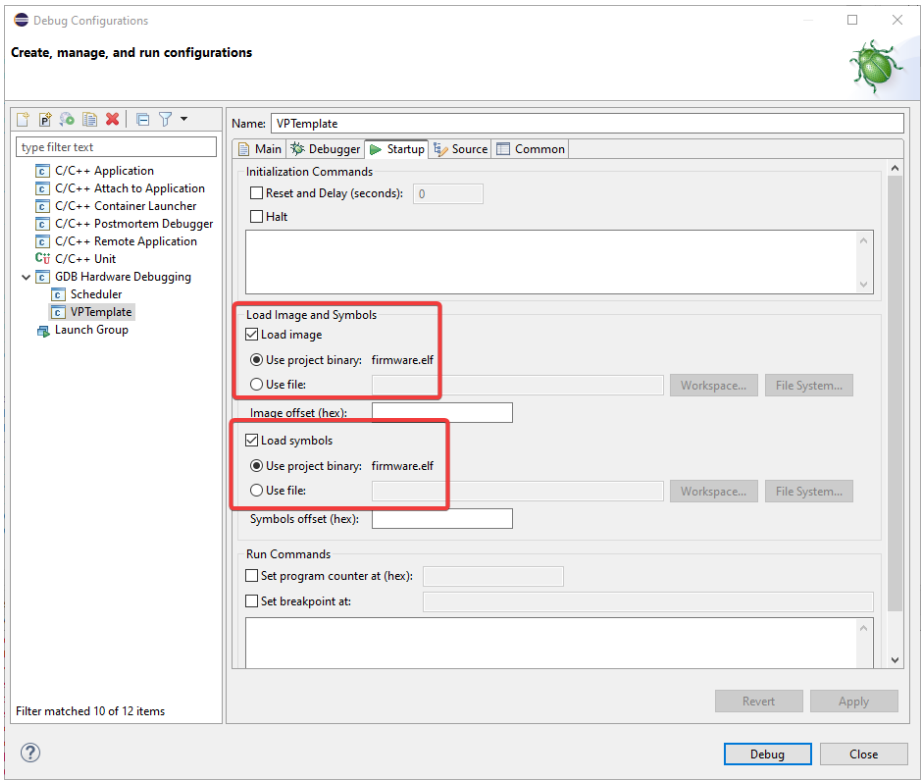


At first this is the GDB Debugger which should be used. For Linux, it is enough to specify there

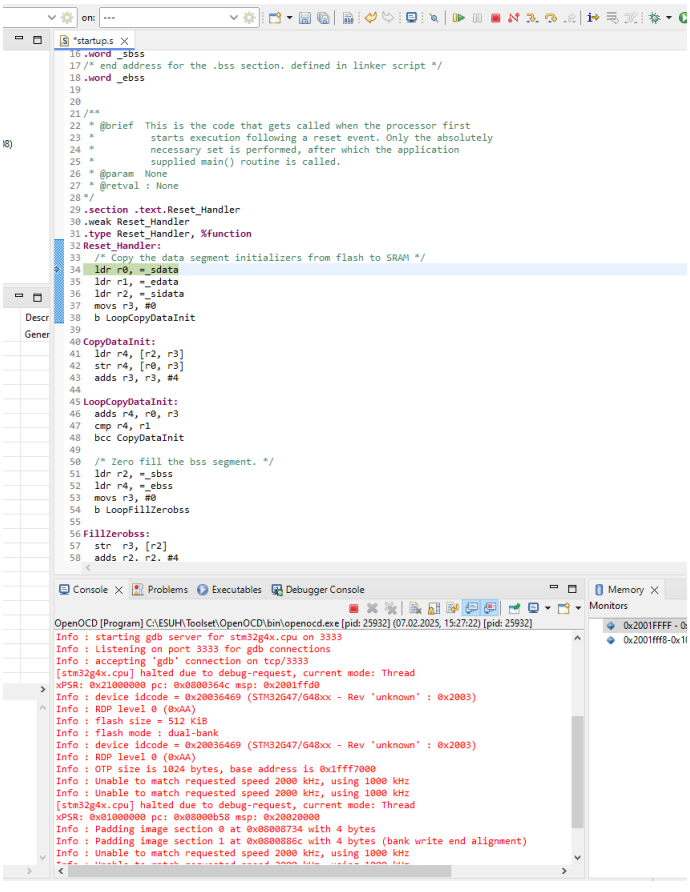
`arm-none-eabi-gdb`

Furthermore, a very important changed compared to the QEmu configurations, is the connection setting. For the usage of OpenOCD, a port of `3333` must be specified.

On the “Startup” tab it must be ensured, that the Options “Load Image” and “Load Symbols” are activated



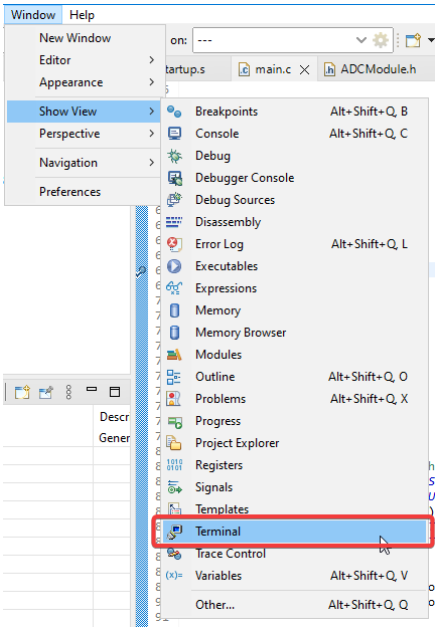
After that, the Debug-Configuration can be launched by pressing “Debug”. But this only works, if OpenOCD has been started before!



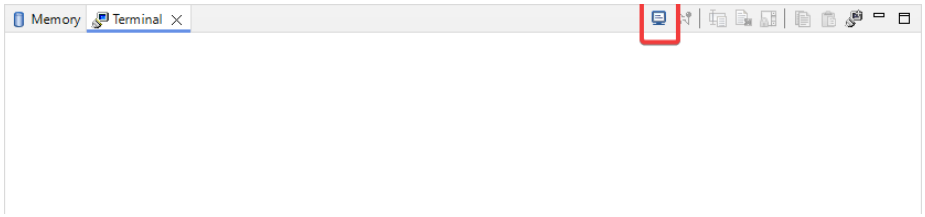
If the debugger is stopped, usually OpenOCD stays connected and still runs in the background. Therefore, it is not necessary to start OpenOCD every time you want to debug.

Connecting the Terminal

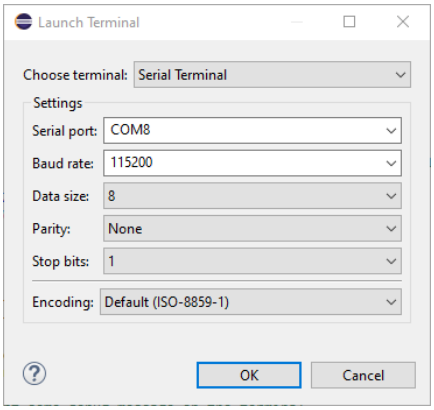
Open the Terminal View in Eclipse



In the Terminal View, the connection settings must be specified. To open the settings, click the “Open a Terminal” button



In the settings window, the configuration for the serial connection must be specified.



The important settings are:

Serial Port: This depends on your system configuration. It might be necessary to test a couple of different ports

Baudrate: 115200

Data size: 8

Parity: None

Stop Bits: 1

After pressing OK, a serial connection is opened on the specified port and debug messages from the VPTemplate project could be shown