Using Debugger

[中文]

This section covers the steps to configure and run a debugger using various methods, including:

- Eclipse
- Command Line
- Idf.py Debug Targets

For how to run a debugger from VS Code, see Configuration for Visual Studio Code Debug.

Eclipse

> **Note**
>
> It is recommended to first check if debugger works using Idf.py Debug Targets or from Command Line and then move to using Eclipse.

Eclipse is an integrated development environment (IDE) that provides a powerful set of tools for developing and debugging software applications. For ESP-IDF applications, IDF Eclipse plugin provides two ways of debugging:

1. ESP-IDF GDB OpenOCD Debugging
2. GDB Hardware Debugging

By default, Eclipse supports OpenOCD Debugging via the GDB Hardware Debugging plugin, which requires starting the OpenOCD server from the command line and configuring the GDB client from Eclipse to start with the debugging. This approach can be time-consuming and error-prone.

To make the debugging process easier, the IDF Eclipse plugin has a customized ESP-IDF GDB OpenOCD Debugging functionality. This functionality supports configuring the OpenOCD server and GDB client from within Eclipse. All the required configuration parameters will be pre-filled by the plugin, and you can start debugging with just a click of a button.

Therefore, it is recommended to use the ESP-IDF GDB OpenOCD Debugging via the IDF Eclipse plugin.

GDB Hardware Debugging

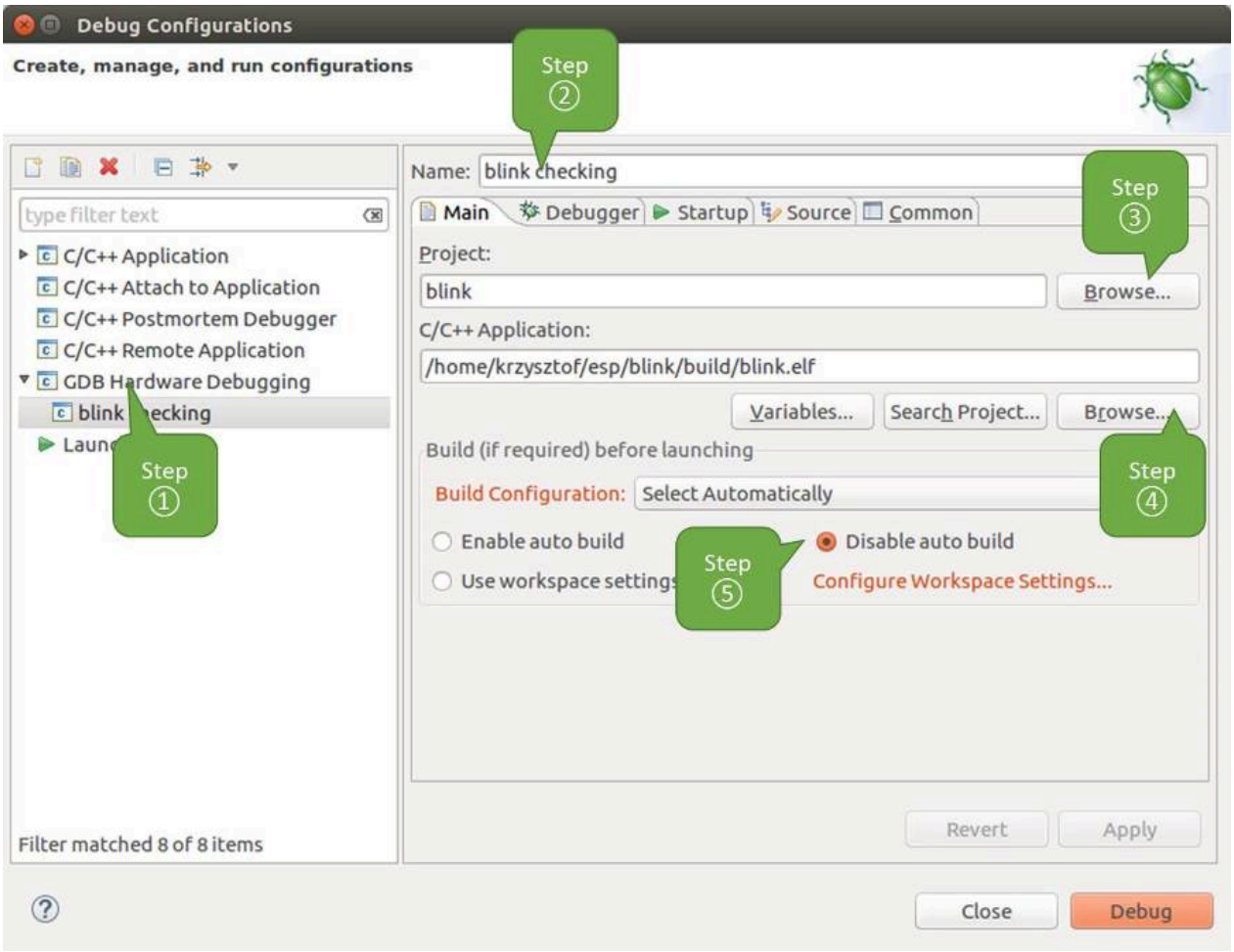
> **Note**
>
> This approach is recommended only if you are unable to debug using ESP-IDF GDB OpenOCD Debugging for some reason.

To install the `GDB Hardware Debugging` plugin, open Eclipse and select `Help` > `Install` New Software.

After installation is complete, follow these steps to configure the debugging session. Please note that some configuration parameters are generic, while others are project-specific. This will be shown below by configuring debugging for "blink" example project. If not done already, add this project to Eclipse workspace following Eclipse Plugin. The source of get-started/blink application is available in examples directory of ESP-IDF repository.

1. In Eclipse, go to `Run` > `Debug Configuration`. A new window will open. In the left pane of the window, double-click `GDB Hardware Debugging` (or select `GDB Hardware Debugging` and press the `New` button) to create a new configuration.
2. In a form that will show up on the right, enter the `Name:` of this configuration, e.g., "Blink checking".
3. On the `Main` tab below, under `Project:`, press the `Browse` button and select the `blink` project.
4. In the next line under `C/C++ Application:`, press the `Browse` button and select the `blink.elf` file. If `blink.elf` is not there, it is likely that this project has not been built yet. Refer to the Eclipse Plugin for instructions.
5. Finally, under `Build (if required) before launching` click `Disable auto build`.

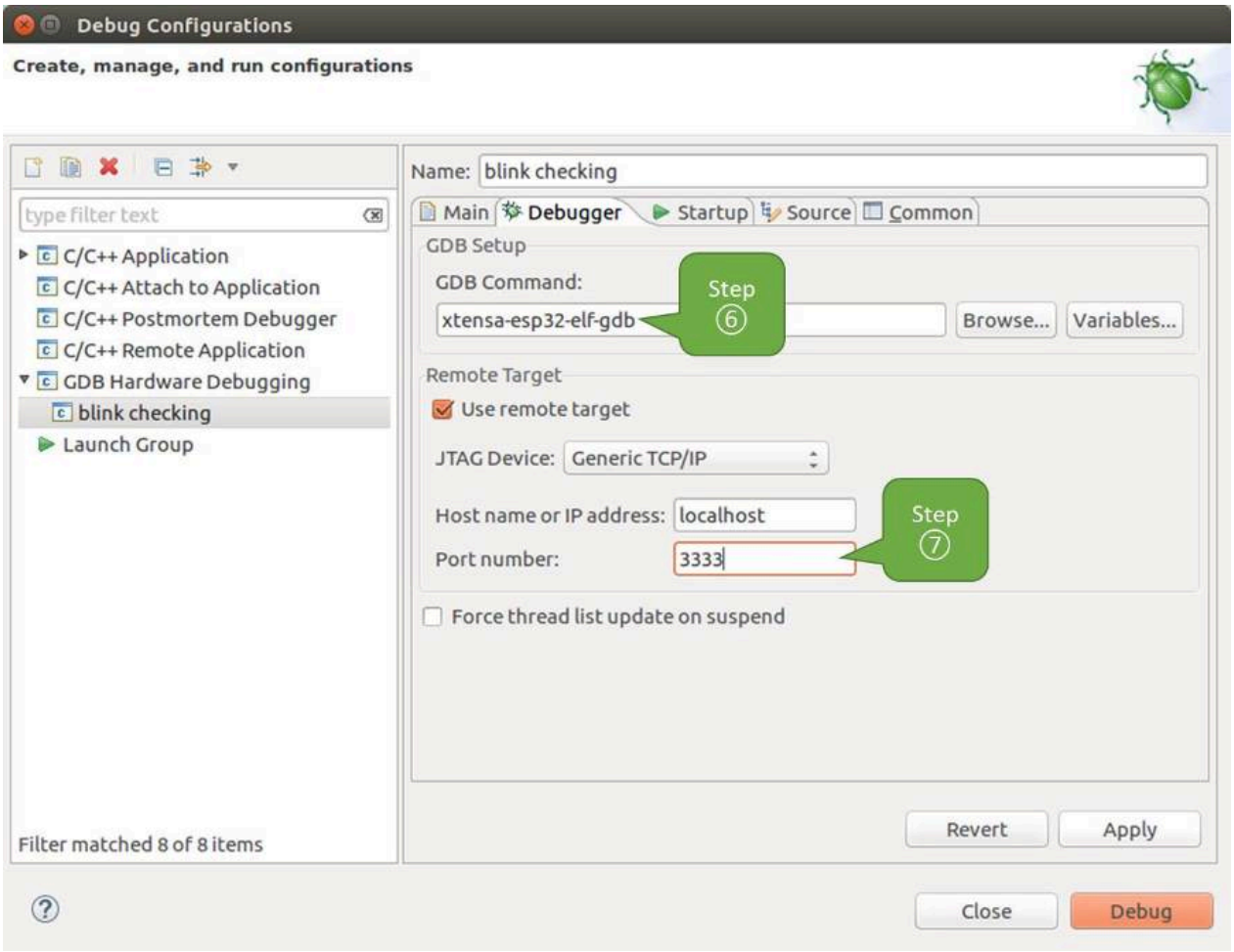
   A sample window with settings entered in points 1 - 5 is shown below.



Configuration of GDB Hardware Debugging - Main tab

6. Click the `Debugger` tab. In field `GDB Command`, enter `xtensa-esp32-elf-gdb` to invoke the debugger.
7. Change the default configuration of the `Remote host` by entering `3333` under the `Port number`.

   Configuration entered in points 6 and 7 is shown on the following picture.



Configuration of GDB Hardware Debugging - Debugger tab

8. The last tab that requires changing the default configuration is `Startup`. Under `Initialization Commands` uncheck `Reset and Delay (seconds)` and `Halt`. Then, in the entry field below, enter the following lines:

```
mon reset halt
maintenance flush register-cache
set remote hardware-watchpoint-limit 2
```

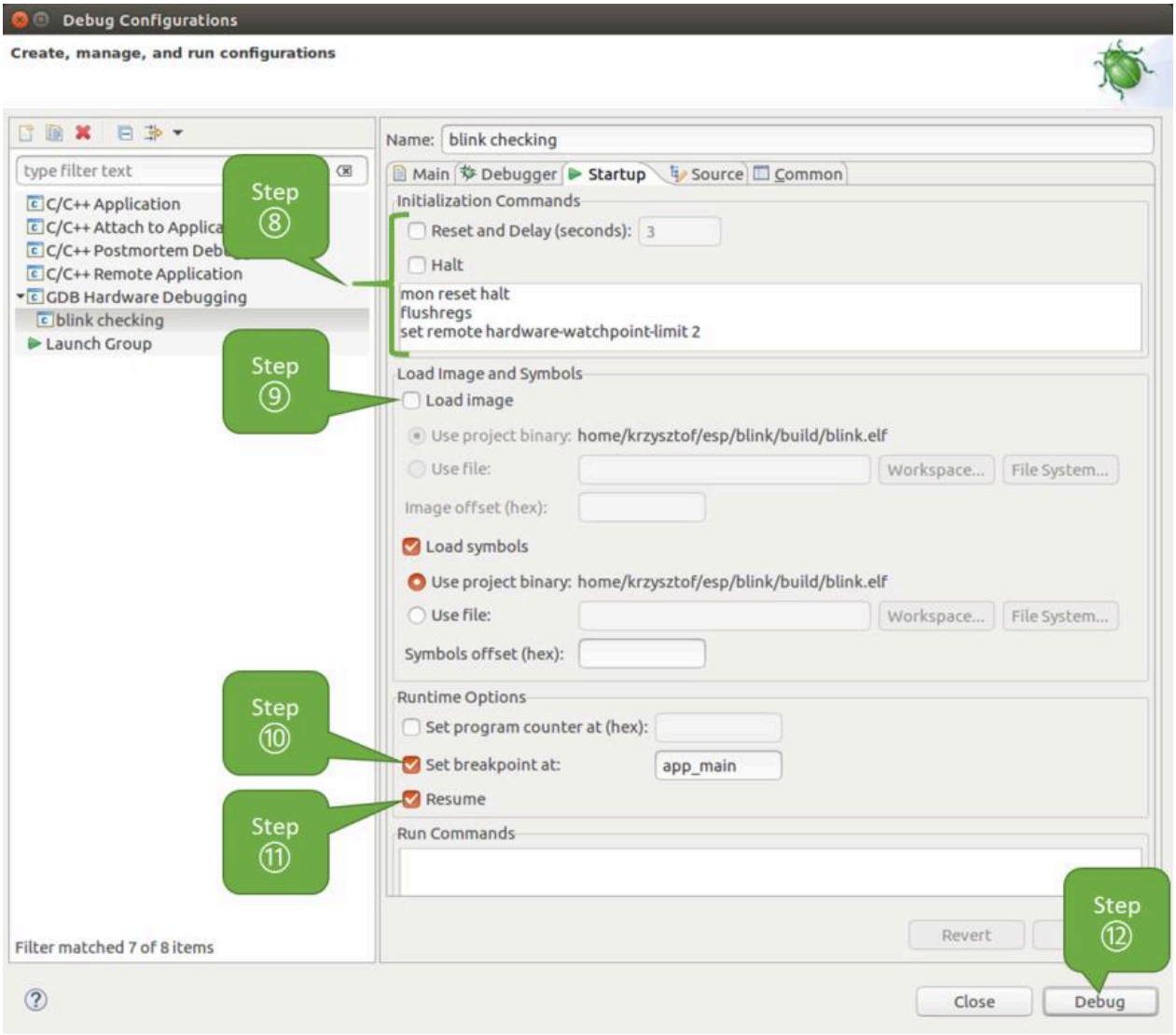
> **Note**
>
> To automatically update the image in the flash before starting a new debug session, add the following command lines to the beginning of the `Initialization Commands` textbox:
>
> ```
> mon reset halt
> mon program_esp ${workspace_loc:blink/build/blink.bin} 0x10000 verify
> ```

For description of `program_esp` command, see Upload Application for Debugging.

9. Uncheck the `Load image` option under `Load Image and Symbols`.

10. Further down on the same tab, establish an initial breakpoint to halt CPUs after they are reset by debugger. The plugin will set this breakpoint at the beginning of the function entered under `Set break point at:`. Checkout this option and enter `app_main` in provided field.
11. Checkout `Resume` option. This will make the program to resume after `mon reset halt` is invoked per point 8. The program will then stop at breakpoint inserted at `app_main`.

    Configuration described in points 8 - 11 is shown below.

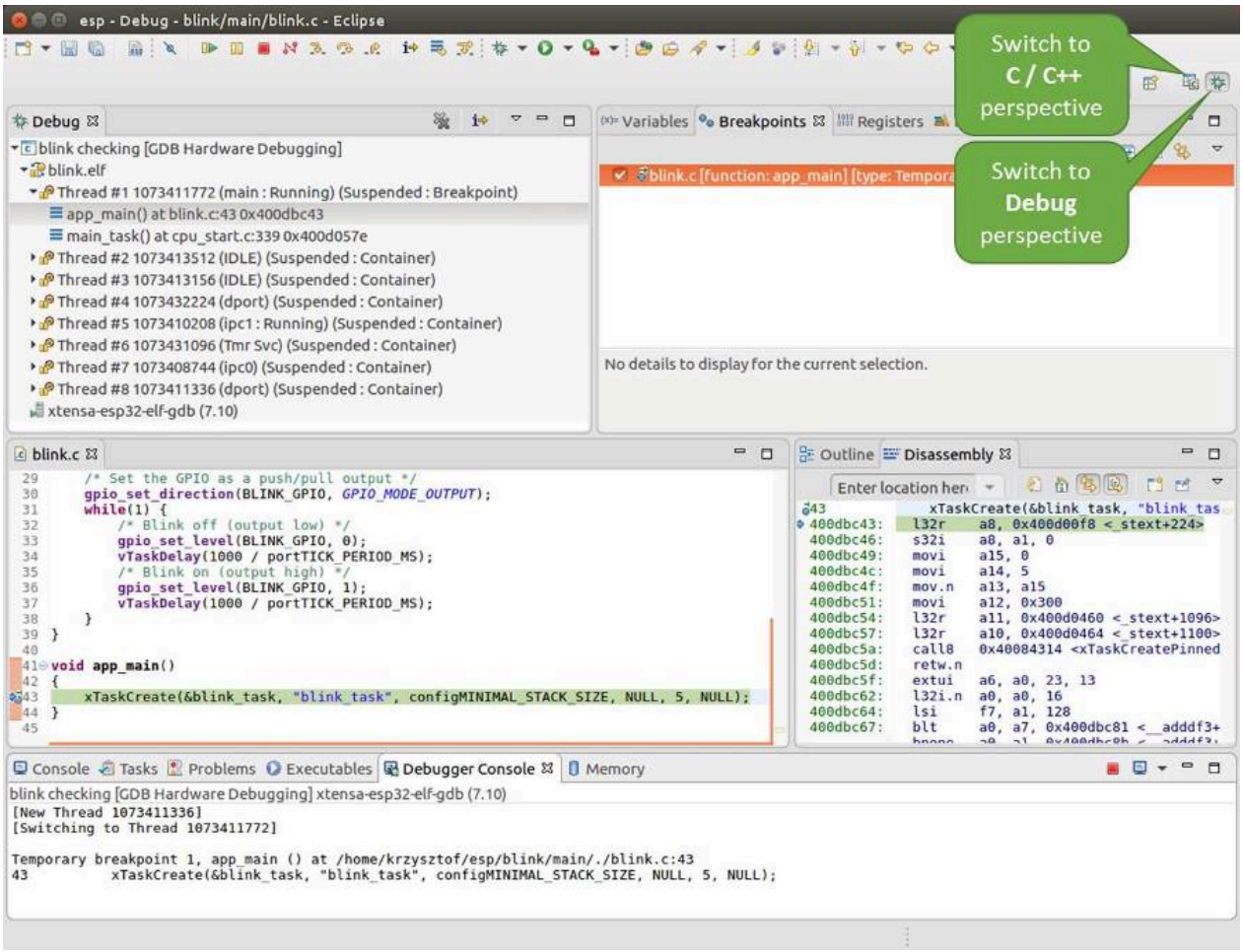


Configuration of GDB Hardware Debugging - Startup tab

If the `Startup` sequence looks convoluted and respective `Initialization Commands` are unclear, check What Is the Meaning of Debugger's Startup Commands? for additional explanation.

12. If you have completed the Configuring ESP32 Target steps described above, so the target is running and ready to talk to debugger, go right to debugging by pressing `Debug` button. Otherwise press `Apply` to save changes, go back to Configuring ESP32 Target and return here to start debugging.

    Once all configuration steps 1-12 are satisfied, the new Eclipse perspective called "Debug" will open, as shown in the example picture below.



Debug Perspective in Eclipse

If you are not quite sure how to use GDB, check Eclipse example debugging session in section Debugging Examples.

Command Line

1. Begin by completing the steps described under Configuring ESP32 Target. This is prerequisite to start a debugging session.

2. Open a new terminal session and go to the directory that contains the project for debugging, e.g.,

```
cd ~/esp/blink
```

3. When launching a debugger, you will need to provide a couple of configuration parameters and commands. The build system generates several `.gdbinit` files to facilitate efficient debugging. Paths to these files can be found in the `build/project_description.json`, under the `gdbinit_files` section. The paths to these files are defined as follows: