使用调试器

[English]

本节会在 Eclipse 和 命令行 中分别介绍配置和运行调试器的方法。我们建议你首先通过 命令行 检查调试器是否正常工作，然后再转到使用 Eclipse 平台。

在 Eclipse 中使用 GDB

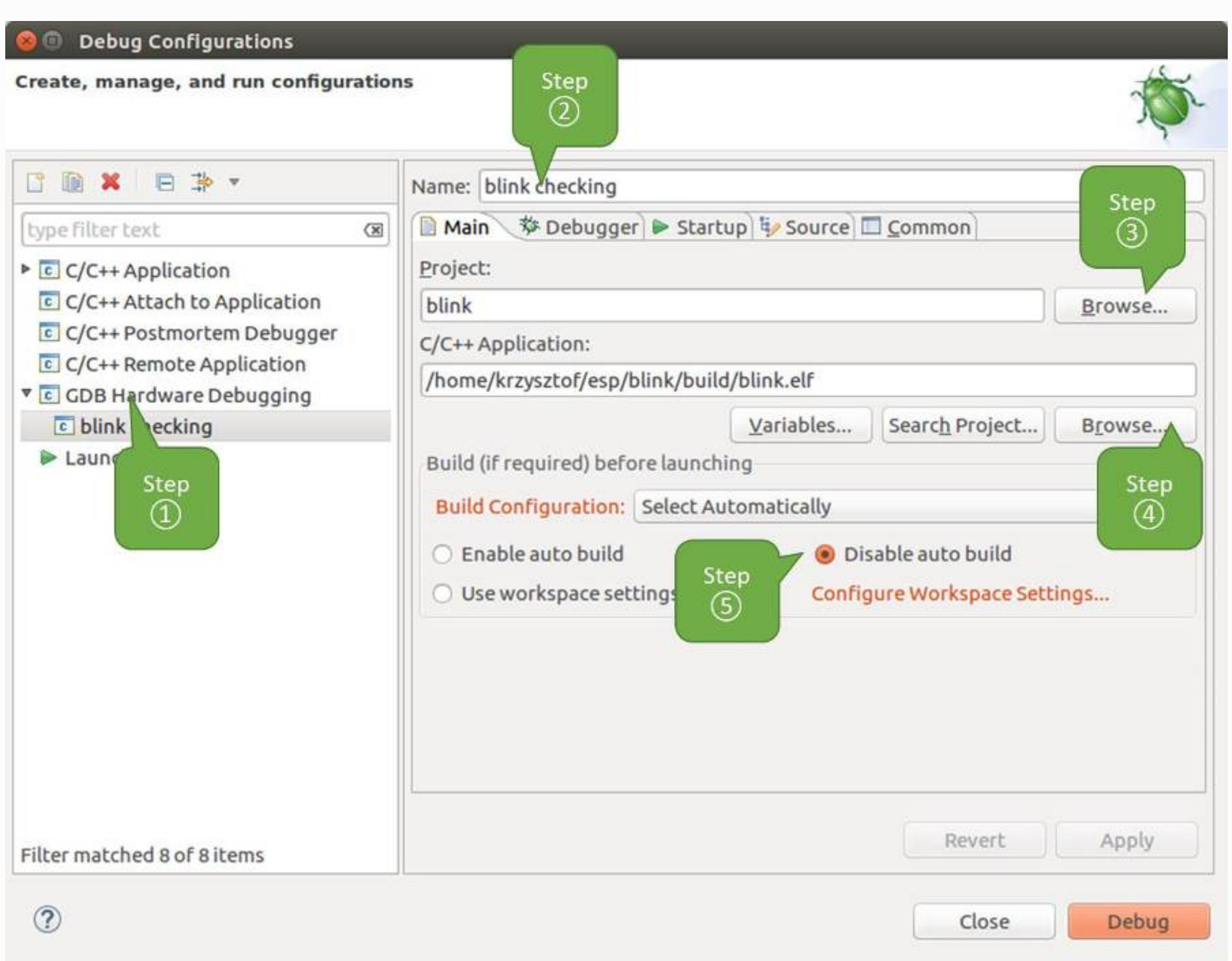
标准的 Eclipse 安装流程默认安装调试功能，另外我们还可以使用插件来调试，比如 “GDB Hardware Debugging”。这个插件用起来非常方便，本指南会详细介绍该插件的使用方法。

首先，通过打开 Eclipse 并转到 “Help” > “Install New Software” 来安装 “GDB Hardware Debugging” 插件。

安装完成后，按照以下步骤配置调试会话。请注意，一些配置参数是通用的，有些则针对特定项目。我们会通过配置 “blink” 示例项目的调试环境来进行展示，请先按照 使用 Eclipse IDE 编译和烧写 文章介绍的方法将该示例项目添加到 Eclipse 的工作空间。示例项目 get-started/blink 的源代码可以在 ESP-IDF 仓库的 examples 目录下找到。

1. 在 Eclipse 中，进入 Run > Debug Configuration，会出现一个新的窗口。在窗口的左侧窗格中，双击 “GDB Hardware Debugging” （或者选择 “GDB Hardware Debugging” 然后按下 “New” 按钮）来新建一个配置。
2. 在右边显示的表单中，“Name:” 一栏中输入配置的名称，例如： “Blink checking”。
3. 在下面的 “Main” 选项卡中， 点击 “Project:” 边上的 “Browse” 按钮，然后选择当前的 “blink” 项目。
4. 在下一行的 “C/C++ Application:” 中，点击 “Browse” 按钮，选择 “blink.elf” 文件。如果 “blink.elf” 文件不存在，那么很有可能该项目还没有编译，请参考 使用 Eclipse IDE 编辑和烧写 指南中的介绍。
5. 最后，在 “Build (if required) before launching” 下面点击 “Disable auto build”。

   上述步骤 1 - 5 的示例输入如下图所示。



Debug Configurations

Create, manage, and run configurations

Name: blink checking

Project: blink

C/C++ Application: /home/krzysztof/esp/blink/build/blink.elf

Build Configuration: Select Automatically

○ Enable auto build

● Disable auto build

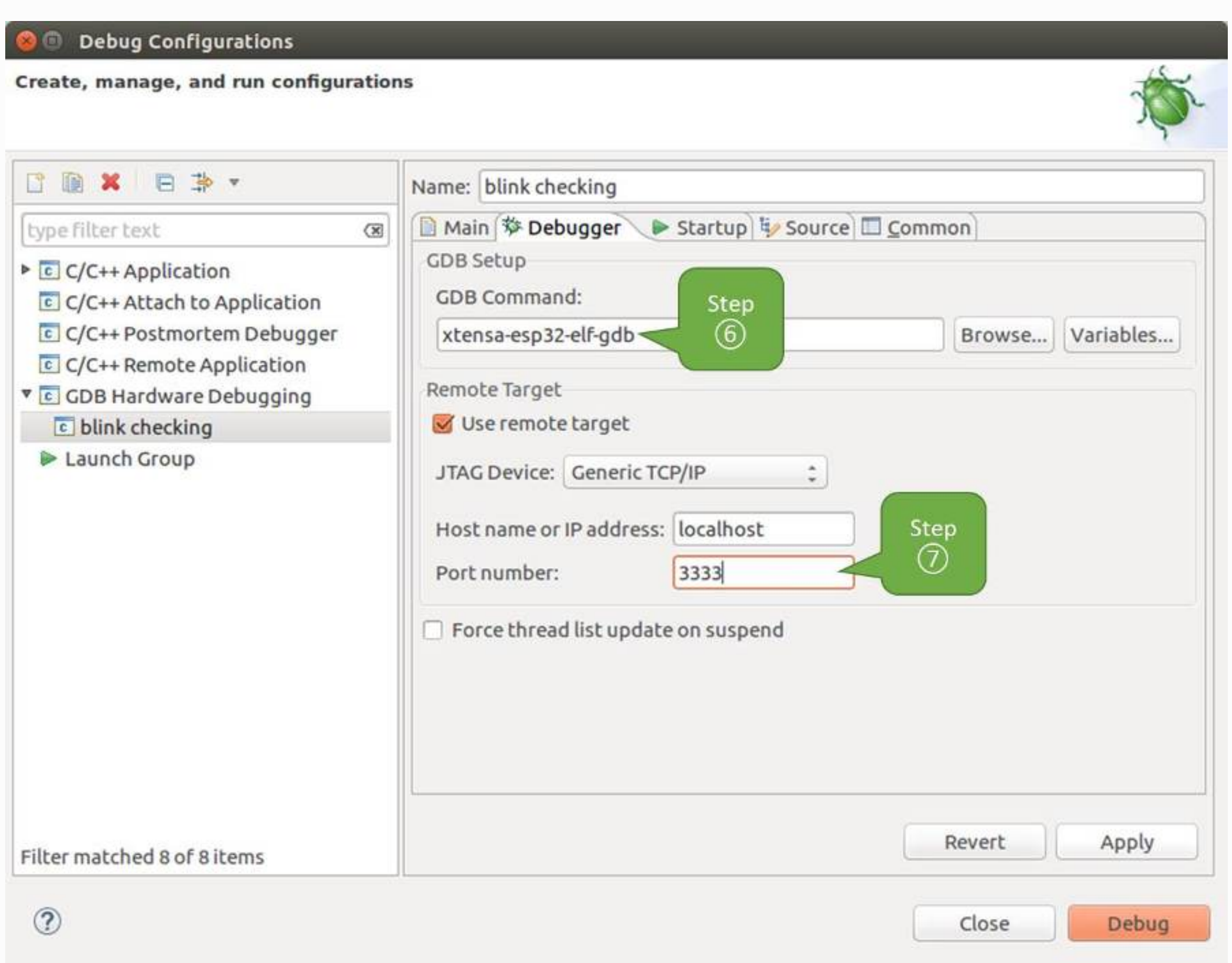
○ Use workspace settings

Filter matched 8 of 8 items

GDB 硬件调试的配置 - Main 选项卡

6. 点击 “Debugger” 选项卡，在 “GDB Command” 栏中输入 `xtensa-esp32-elf-gdb` 来调用调试器。
7. 更改 “Remote host” 的默认配置，在 “Port number” 下面输入 `3333`。

   上述步骤 6 - 7 的示例输入如下图所示。



Debug Configurations

Create, manage, and run configurations

Name: blink checking

GDB Setup

GDB Command: xtensa-esp32-elf-gdb

Remote Target

☑ Use remote target

JTAG Device: Generic TCP/IP

Host name or IP address: localhost

Port number: 3333

☐ Force thread list update on suspend

Filter matched 8 of 8 items

GDB 硬件调试的配置 - Debugger 选项卡

8. 最后一个需要更改默认配置的选项卡是 “Startup” 选项卡。在 “Initialization Commands” 下，取消选中 “Reset and Delay (seconds)” 和 “Halt”，然后在下面一栏中输入以下命令：

```
mon reset halt
flushregs
set remote hardware-watchpoint-limit 2
```

> **注解**
>
> 如果你想在启动新的调试会话之前自动更新闪存中的镜像，请在 “Initialization Commands” 文本框的开头添加以下命令行：
>
> ```
> mon reset halt
> mon program_esp32 ${workspace_loc:blink/build/blink.bin} 0x10000 verify
> ```

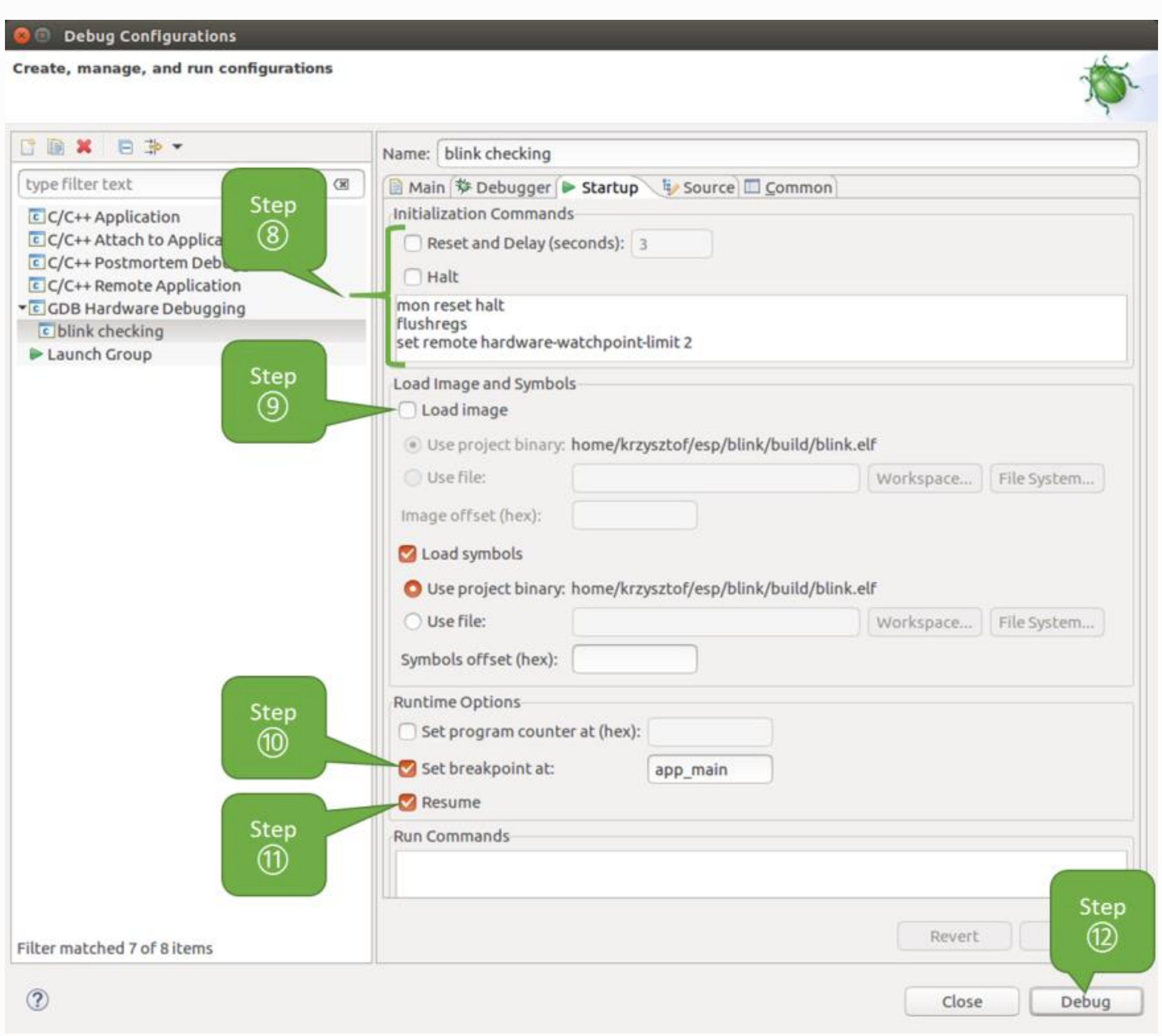
有关 `program_esp32` 命令的说明请参考 上传待调试的应用程序 章节。

9. 在 “Load Image and Symbols” 下，取消选中 “Load image” 选项。

0. 在同一个选项卡中继续往下浏览，建立一个初始断点用来在调试器复位后暂停 CPU。插件会根据 “Set break point at:” 一栏中输入的函数名，在该函数的开头设置断点。选中这一选项，并在相应的字段中输入 `app_main`。

1. 选中 “Resume” 选项，这会使得程序在每次调用步骤 8 中的 `mon reset halt` 之后恢复，然后在 `app_main` 的断点处停止。

   上述步骤 8 - 11 的示例输入如下图所示。



Debug Configurations

Create, manage, and run configurations

Name: blink checking

Initialization Commands

☐ Reset and Delay (seconds): 3

☐ Halt

```
mon reset halt
flushregs
set remote hardware-watchpoint-limit 2
```

Load Image and Symbols

☐ Load image

○ Use project binary: home/krzysztof/esp/blink/build/blink.elf

○ Use file:

Image offset (hex):

☑ Load symbols

● Use project binary: home/krzysztof/esp/blink/build/blink.elf

○ Use file:

Symbols offset (hex):

Runtime Options

☐ Set program counter at (hex):

☑ Set breakpoint at: app_main

☑ Resume

Run Commands

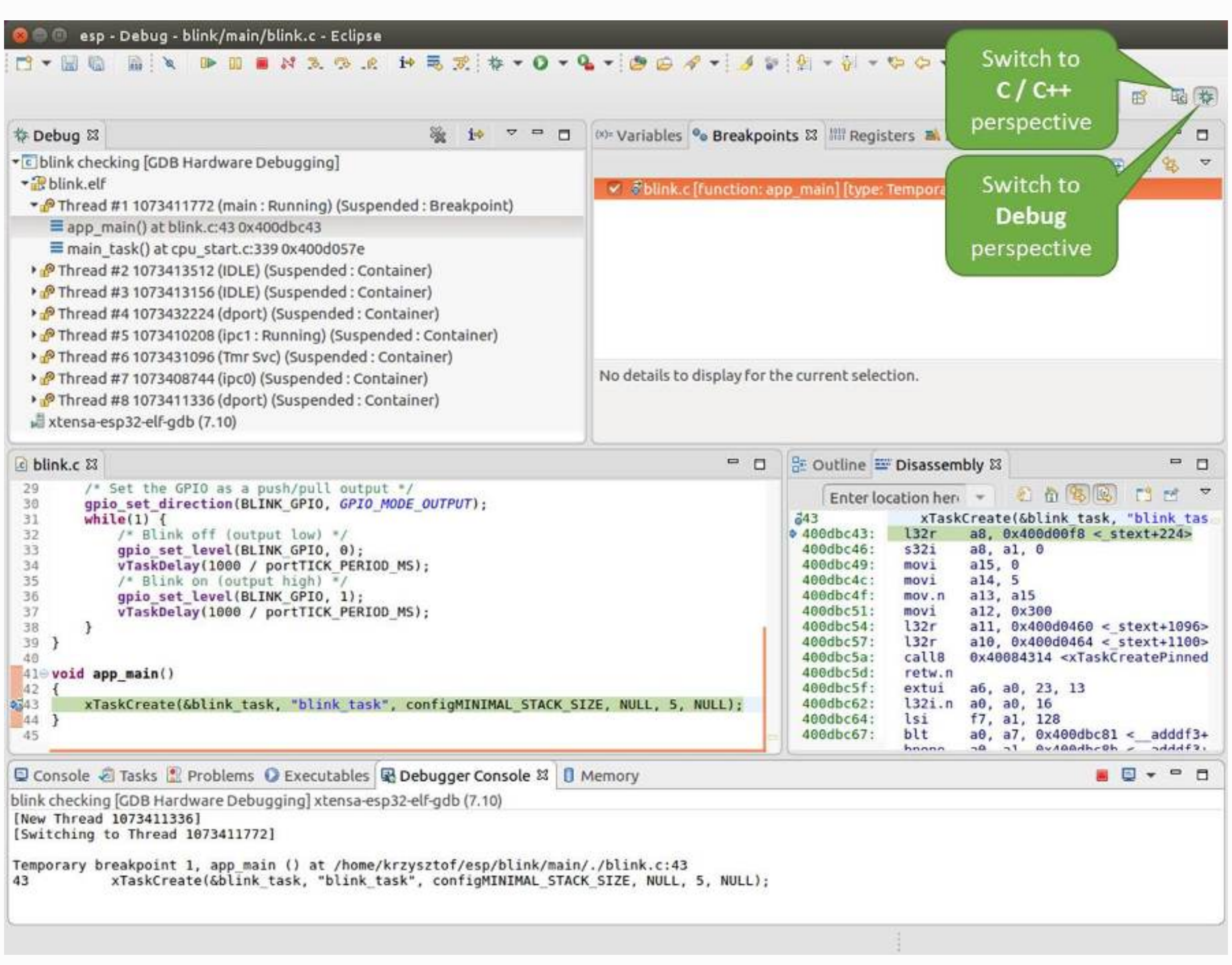
Filter matched 7 of 8 items

GDB 硬件调试的配置 - Startup 选项卡

上面的启动序列看起来有些复杂，如果你对其中的初始化命令不太熟悉，请查阅 调试器的启动命令的含义 章节获取更多说明。

2. 如果你前面已经完成 配置 ESP32 目标板 中介绍的步骤，那么目标正在运行并准备与调试器进行对话。按下 “Debug” 按钮就可以直接调试。否则请按下 “Apply” 按钮保存配置，返回 配置 ESP32 目标板 章节进行配置，最后再回到这里开始调试。

一旦所有 1 - 12 的配置步骤都已经完成，Eclipse 就会打开 “Debug” 视图，如下图所示。



esp - Debug - blink/main/blink.c - Eclipse

```
blink checking [GDB Hardware Debugging]
  blink.elf
    Thread #1 1073411772 (main : Running) (Suspended : Breakpoint)
      app_main() at blink.c:43 0x400dbc43
      main_task() at cpu_start.c:339 0x400d057e
    Thread #2 1073413512 (IDLE) (Suspended : Container)
    Thread #3 1073413156 (IDLE) (Suspended : Container)
    Thread #4 1073432224 (dport) (Suspended : Container)
    Thread #5 1073410208 (ipc1 : Running) (Suspended : Container)
    Thread #6 1073431096 (Tmr Svc) (Suspended : Container)
    Thread #7 1073408744 (ipc0) (Suspended : Container)
    Thread #8 1073411336 (dport) (Suspended : Container)
  xtensa-esp32-elf-gdb (7.10)
```

blink.c [function: app_main] [type: Tempora

No details to display for the current selection.

```
        /* Set the GPIO as a push/pull output */
        gpio_set_direction(BLINK_GPIO, GPIO_MODE_OUTPUT);
        while(1) {
            /* Blink off (output low) */
            gpio_set_level(BLINK_GPIO, 0);
            vTaskDelay(1000 / portTICK_PERIOD_MS);
            /* Blink on (output high) */
            gpio_set_level(BLINK_GPIO, 1);
            vTaskDelay(1000 / portTICK_PERIOD_MS);
        }
}

void app_main()
{
    xTaskCreate(&blink_task, "blink_task", configMINIMAL_STACK_SIZE, NULL, 5, NULL);
}
```

```
43            xTaskCreate(&blink_task, "blink_tas
400dbc43:   l32r    a8, 0x400d00f8 <_stext+224>
400dbc46:   s32i    a8, a1, 0
400dbc49:   movi    a15, 0
400dbc4c:   movi    a14, 5
400dbc4f:   mov.n   a13, a15
400dbc51:   movi    a12, 0x300
400dbc54:   l32r    a11, 0x400d0460 <_stext+1096>
400dbc57:   l32r    a10, 0x400d0464 <_stext+1100>
400dbc5a:   call8   0x40084314 <xTaskCreatePinned
400dbc5d:   retw.n
400dbc5f:   extui   a6, a0, 23, 13
400dbc62:   l32i.n  a0, a0, 16
400dbc64:   lsi     f7, a1, 128
400dbc67:   blt     a0, a7, 0x400dbc81 <__adddf3+
```

blink checking [GDB Hardware Debugging] xtensa-esp32-elf-gdb (7.10)

```
[New Thread 1073411336]
[Switching to Thread 1073411772]

Temporary breakpoint 1, app_main () at /home/krzysztof/esp/blink/main/./blink.c:43
43          xTaskCreate(&blink_task, "blink_task", configMINIMAL_STACK_SIZE, NULL, 5, NULL);
```

Eclipse 中的调试视图

如果你不太了解 GDB 的常用方法，请查阅 使用 Eclipse 的调试示例 文章中的调试示例章节 调试范例 。

在命令行中使用 GDB

1. 为了能够启动调试会话，需要先启动并运行目标，如果还没有完成，请按照 配置 ESP32 目标板 中的介绍进行操作。
2. 打开一个新的终端会话并前往待调试的项目目录，比如：

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

3. 当启动调试器时，通常需要提供几个配置参数和命令，为了避免每次都在命令行中逐行输入这些命令，我们可以新建一个配置文件，并将其命名为 `gdbinit`：

   ```
   target remote :3333
   set remote hardware-watchpoint-limit 2
   mon reset halt
   flushregs
   thb app_main
   c
   ```

   将此文件保存在当前目录中。

   有关 `gdbinit` 文件内部的更多详细信息，请参阅 调试器的启动命令的含义 章节。

4. 准备好启动 GDB，请在终端中输入以下内容：

   ```
   xtensa-esp32-elf-gdb -x gdbinit build/blink.elf
   ```

5. 如果前面的步骤已经正确完成，你会看到如下所示的输出日志，在日志的最后会出现 `(gdb)` 提示符：

   ```
   user-name@computer-name:~/esp/blink$ xtensa-esp32-elf-gdb -x gdbinit build/blink.elf
   GNU gdb (crosstool-NG crosstool-ng-1.22.0-61-gab8375a) 7.10
   Copyright (C) 2015 Free Software Foundation, Inc.
   License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
   This is free software: you are free to change and redistribute it.
   There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
   and "show warranty" for details.
   This GDB was configured as "--host=x86_64-build_pc-linux-gnu --target=xtensa-esp32-elf".
   Type "show configuration" for configuration details.
   For bug reporting instructions, please see:
   <http://www.gnu.org/software/gdb/bugs/>.
   Find the GDB manual and other documentation resources online at:
   <http://www.gnu.org/software/gdb/documentation/>.
   For help, type "help".
   Type "apropos word" to search for commands related to "word"...
   Reading symbols from build/blink.elf...done.
   0x400d10d8 in esp_vApplicationIdleHook () at /home/user-name/esp/esp-idf/components/esp32/./freertos_hooks.c:52
   52          asm("waiti 0");
   JTAG tap: esp32.cpu0 tap/device found: 0x120034e5 (mfg: 0x272 (Tensilica), part: 0x2003, ver: 0x1)
   JTAG tap: esp32.slave tap/device found: 0x120034e5 (mfg: 0x272 (Tensilica), part: 0x2003, ver: 0x1)
   esp32: Debug controller was reset (pwrstat=0x5F, after clear 0x0F).
   esp32: Core was reset (pwrstat=0x5F, after clear 0x0F).
   Target halted. PRO_CPU: PC=0x5000004B (active)    APP_CPU: PC=0x00000000
   esp32: target state: halted
   esp32: Core was reset (pwrstat=0x1F, after clear 0x0F).
   Target halted. PRO_CPU: PC=0x40000400 (active)    APP_CPU: PC=0x40000400
   esp32: target state: halted
   Hardware assisted breakpoint 1 at 0x400db717: file /home/user-name/esp/blink/main/./blink.c, line 43.
   0x0:    0x00000000
   Target halted. PRO_CPU: PC=0x400DB717 (active)    APP_CPU: PC=0x400D10D8
   [New Thread 1073428656]
   [New Thread 1073413708]
   [New Thread 1073431316]
   [New Thread 1073410672]
   [New Thread 1073408876]
   [New Thread 1073432196]
   [New Thread 1073411552]
   [Switching to Thread 1073411996]

   Temporary breakpoint 1, app_main () at /home/user-name/esp/blink/main/./blink.c:43
   43      xTaskCreate(&blink_task, "blink_task", 512, NULL, 5, NULL);
   (gdb)
   ```

注意上面日志的倒数第三行显示了调试器已经在 `app_main()` 函数的断点处停止，该断点在 `gdbinit` 文件中设定。由于处理器已经暂停运行，LED 也不会闪烁。如果这也是你看到的现象，你可以开始调试了。

如果你不太了解 GDB 的常用方法，请查阅 使用命令行的调试示例 文章中的调试示例章节 调试范例 。