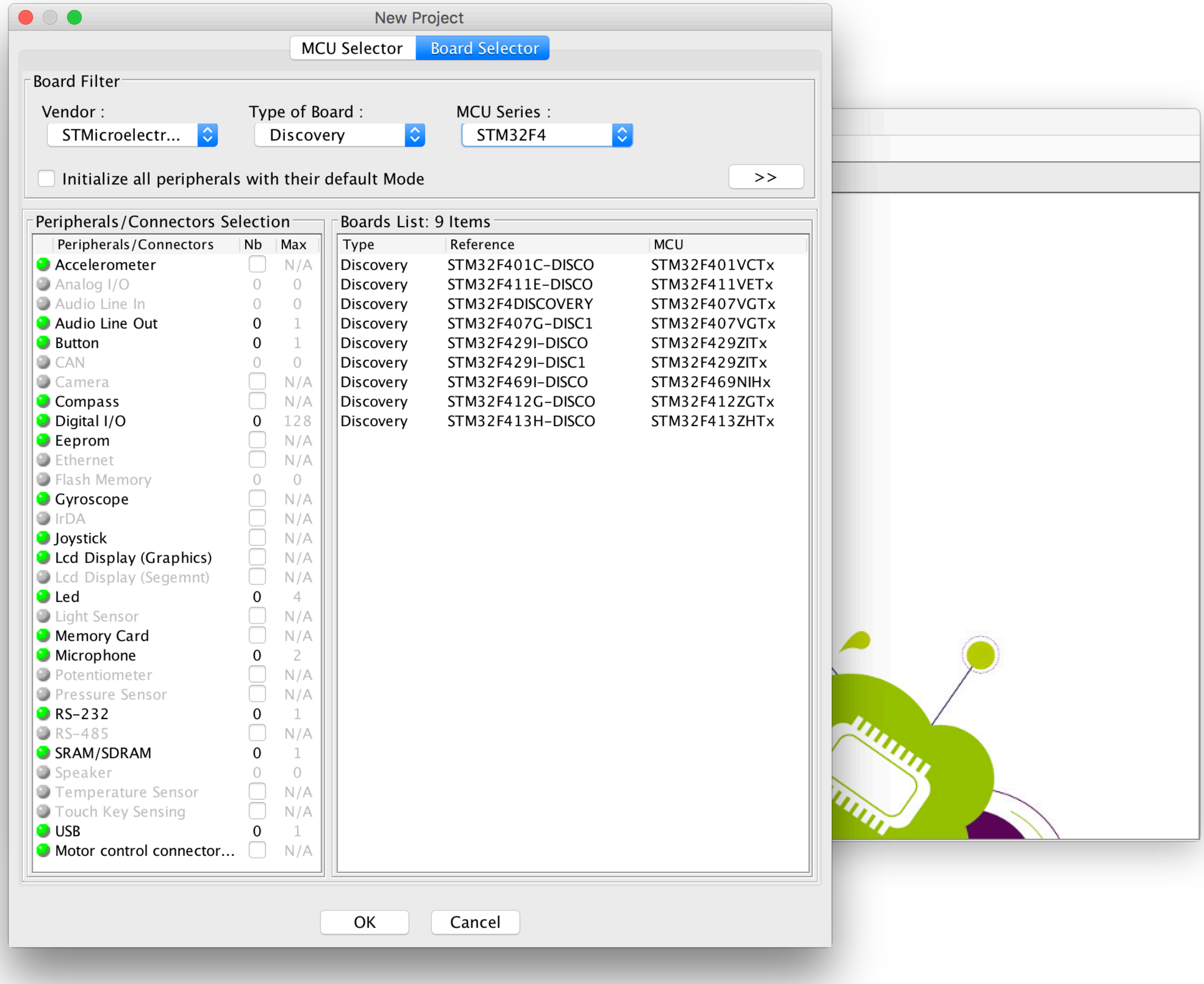
Hello World w/ OpenOCD

Initilize project with STM32CubeMX



New Project

Board Filter

Vendor : STMicroelectr...

Type of Board : Discovery

MCU Series : STM32F4

□ Initialize all peripherals with their default Mode

Peripherals/Connectors Selection

Peripherals/Connectors	Nb	Max
● Accelerometer	□	N/A
○ Analog I/O	0	0
○ Audio Line In	0	0
● Audio Line Out	0	1
● Button	0	1
○ CAN	0	0
○ Camera	□	N/A
● Compass	□	N/A
● Digital I/O	0	128
● Eeprom	□	N/A
○ Ethernet	□	N/A
○ Flash Memory	0	0
● Gyroscope	□	N/A
○ IrDA	□	N/A
● Joystick	□	N/A
● Lcd Display (Graphics)	□	N/A
○ Lcd Display (Segemnt)	□	N/A
● Led	0	4
○ Light Sensor	□	N/A
● Memory Card	□	N/A
● Microphone	0	2
○ Potentiometer	□	N/A
○ Pressure Sensor	□	N/A
● RS-232	0	1
○ RS-485	□	N/A
● SRAM/SDRAM	0	1
○ Speaker	0	0
○ Temperature Sensor	□	N/A
○ Touch Key Sensing	□	N/A
● USB	0	1
● Motor control connector...	□	N/A

Boards List: 9 Items

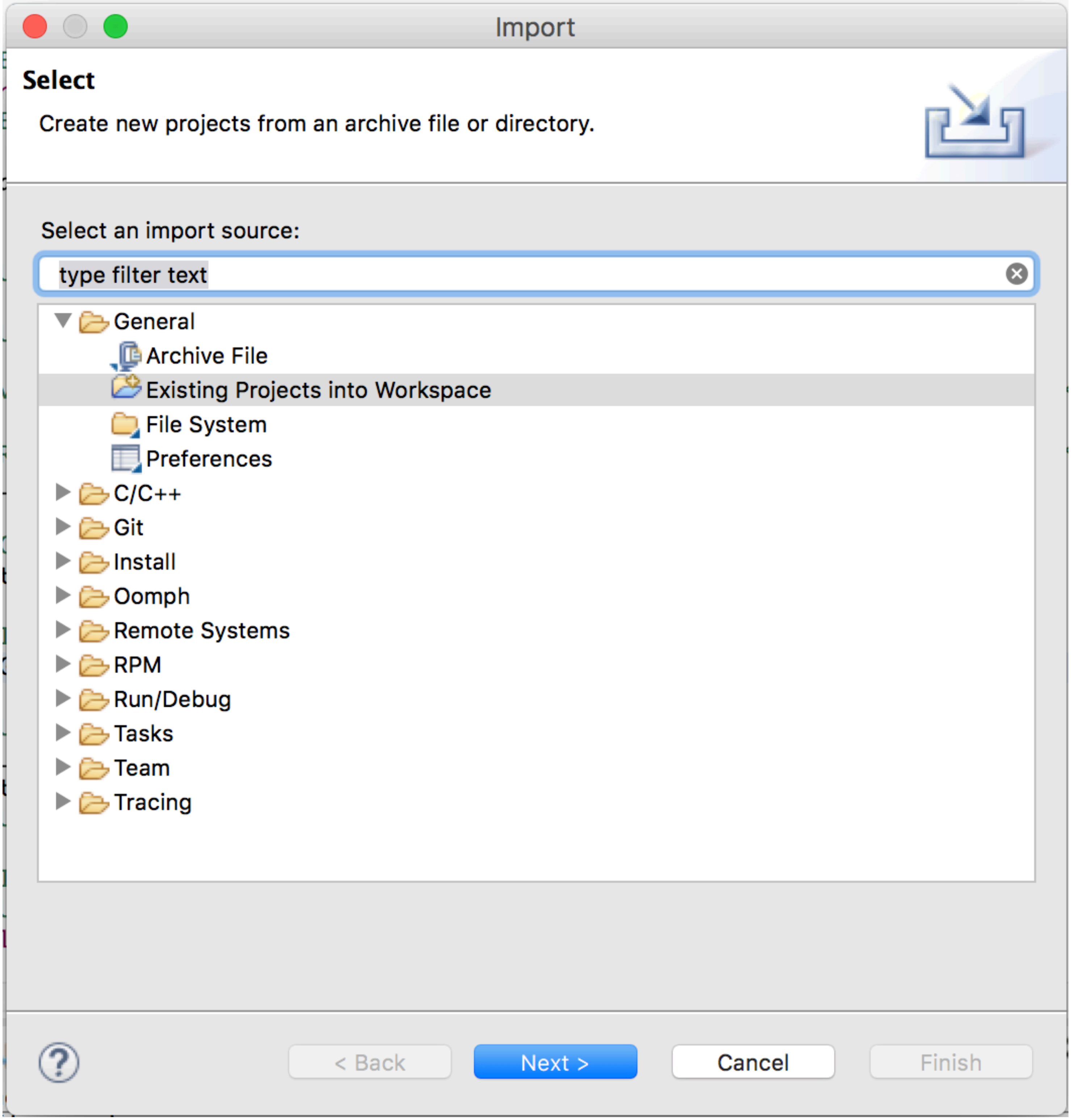
Type	Reference	MCU
Discovery	STM32F401C-DISCO	STM32F401VCTx
Discovery	STM32F411E-DISCO	STM32F411VETx
Discovery	STM32F4DISCOVERY	STM32F407VGTx
Discovery	STM32F407G-DISC1	STM32F407VGTx
Discovery	STM32F429I-DISCO	STM32F429ZITx
Discovery	STM32F429I-DISC1	STM32F429ZITx
Discovery	STM32F469I-DISCO	STM32F469NIHx
Discovery	STM32F412G-DISCO	STM32F412ZGTx
Discovery	STM32F413H-DISCO	STM32F413ZHTx

Select Discovery - STM32F4DISCOVERY - STM32F407VGTx

Under **Settings -> Toolchain / IDE**
Select `SW4STM32` which is for Eclipse

Add any peripherals and name the pins then generate the code.

In Eclipse import the newly created project by **File -> Import then select General -> Existing Projects into Workspace**



Import

Select

Create new projects from an archive file or directory.

Select an import source:

All the source code and libraries should now be in the project. To setup OpenOCD and add the `printf()` function follow the steps below.

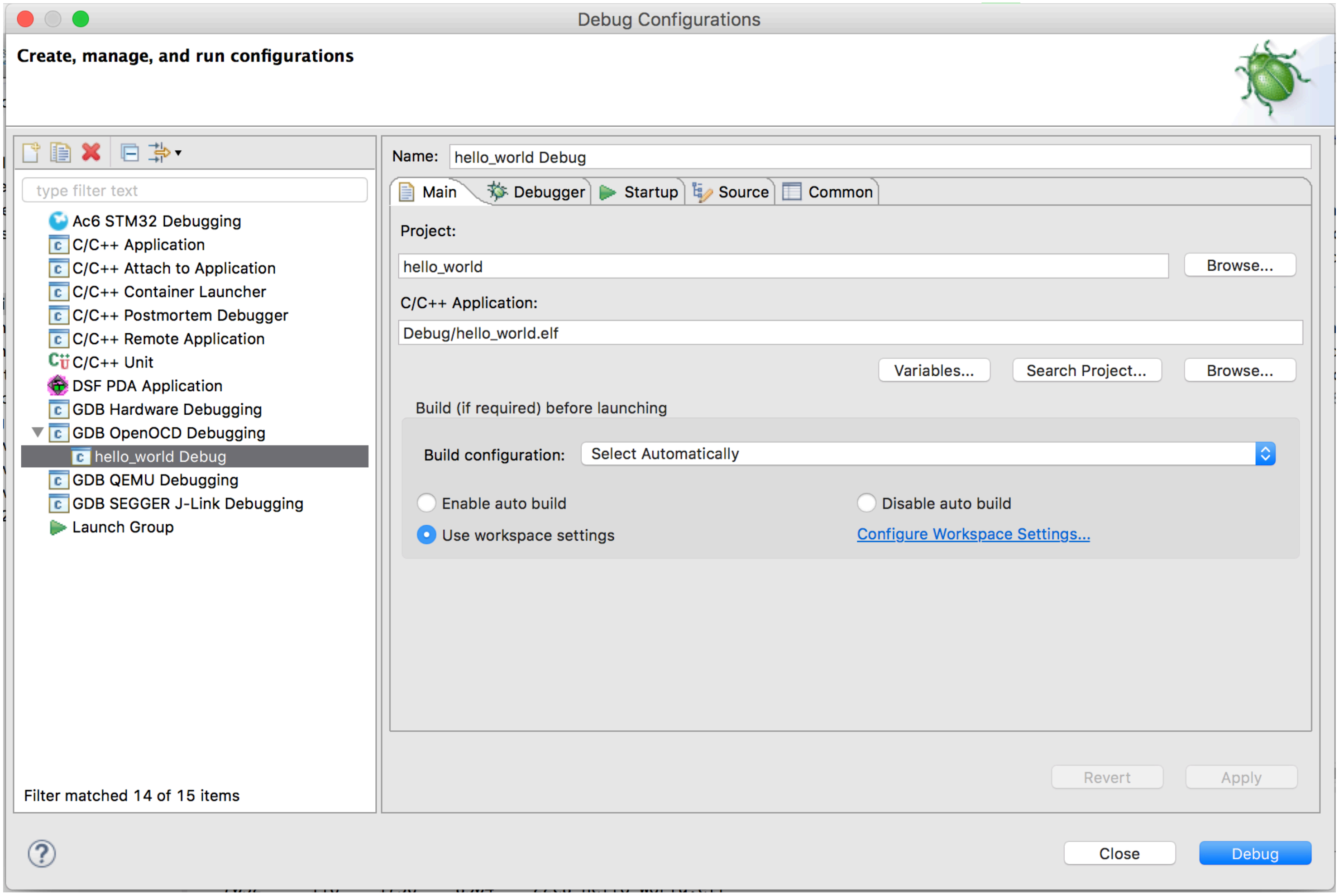
In the **Project Properties -> C/C++ Build -> Setttings -> MCU GCC Linker -> Miscellaneous -> Linker flags**
Add the following flags: `-specs=nosys.specs -specs=nano.specs -specs=rdimon.specs -lc -lrdimon`

In the `main.c` file in the prototypes section add the following
`extern void initialise_monitor_handles(void);`
Then in the main function after the variables declaration add a call to the

function `initialise_monitor_handles();`

After this `printf()` will now display in the console.

To setup OpenOCD debugging select **Debug Configurations**.
Add a new **GDB OpenOCD Debugging** instance



Debug Configurations

Create, manage, and run configurations

Name: hello_world Debug

Project: hello_world

C/C++ Application: Debug/hello_world.elf

Build (if required) before launching

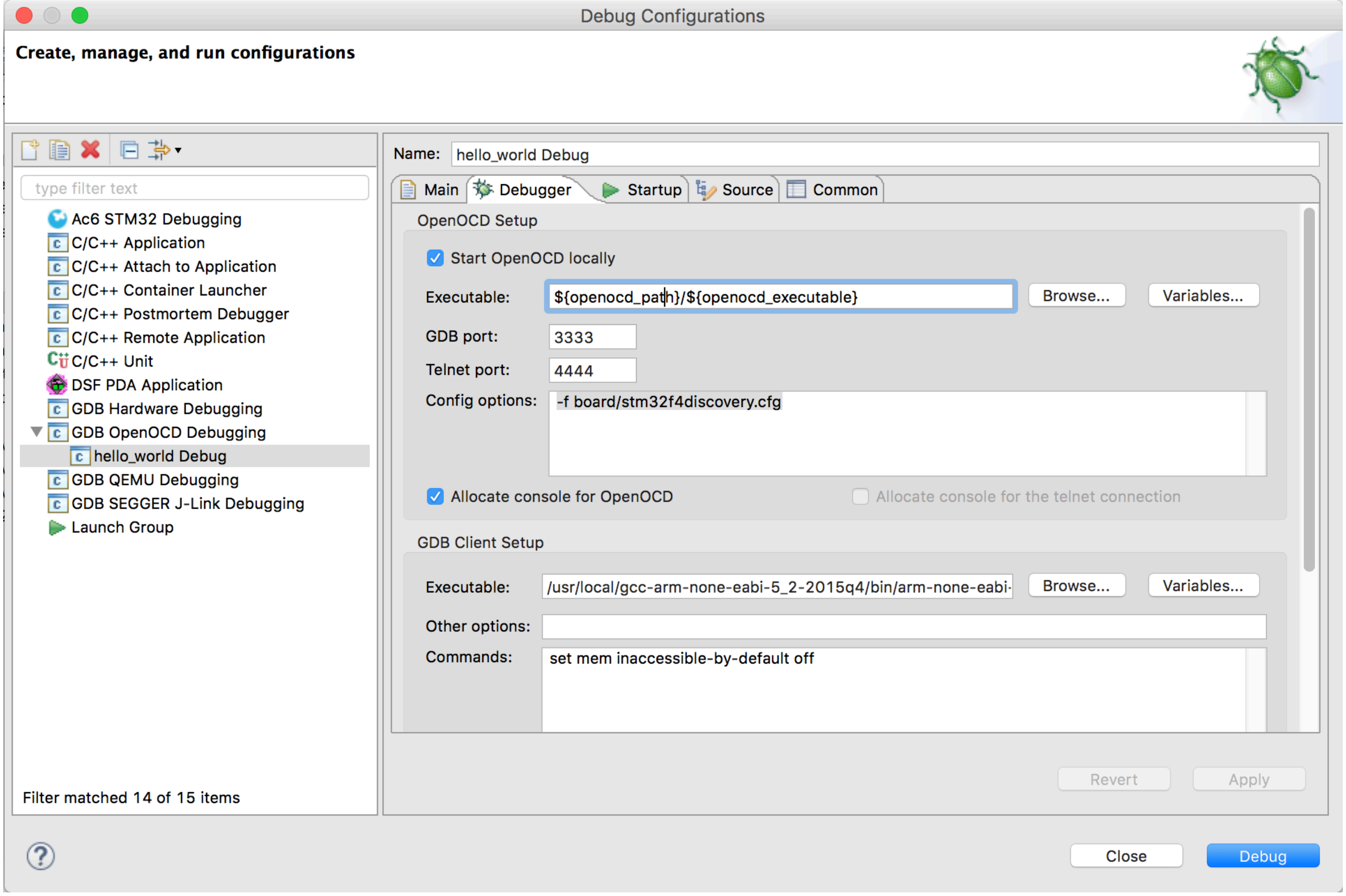
Build configuration: Select Automatically

○ Enable auto build ○ Disable auto build

● Use workspace settings

Filter matched 14 of 15 items

In the **Debugger tab -> OpenOCD Setup -> Config options:** add the following `-f board/stm32f4discovery.cfg` or whatever the configuration file is for the needed board. Add the GDB path if needed under **Debugger tab -> GDB Client Setup -> Executable:** should look something like this `/usr/local/gcc-arm-none-eabi-5_2-2015q4/bin/arm-none-eabi-gdb`



Name: hello_world Debug

OpenOCD Setup

☑ Start OpenOCD locally

Executable: ${openocd_path}/${openocd_executable}

GDB port: 3333

Telnet port: 4444

Config options: -f board/stm32f4discovery.cfg

☑ Allocate console for OpenOCD   □ Allocate console for the telnet connection

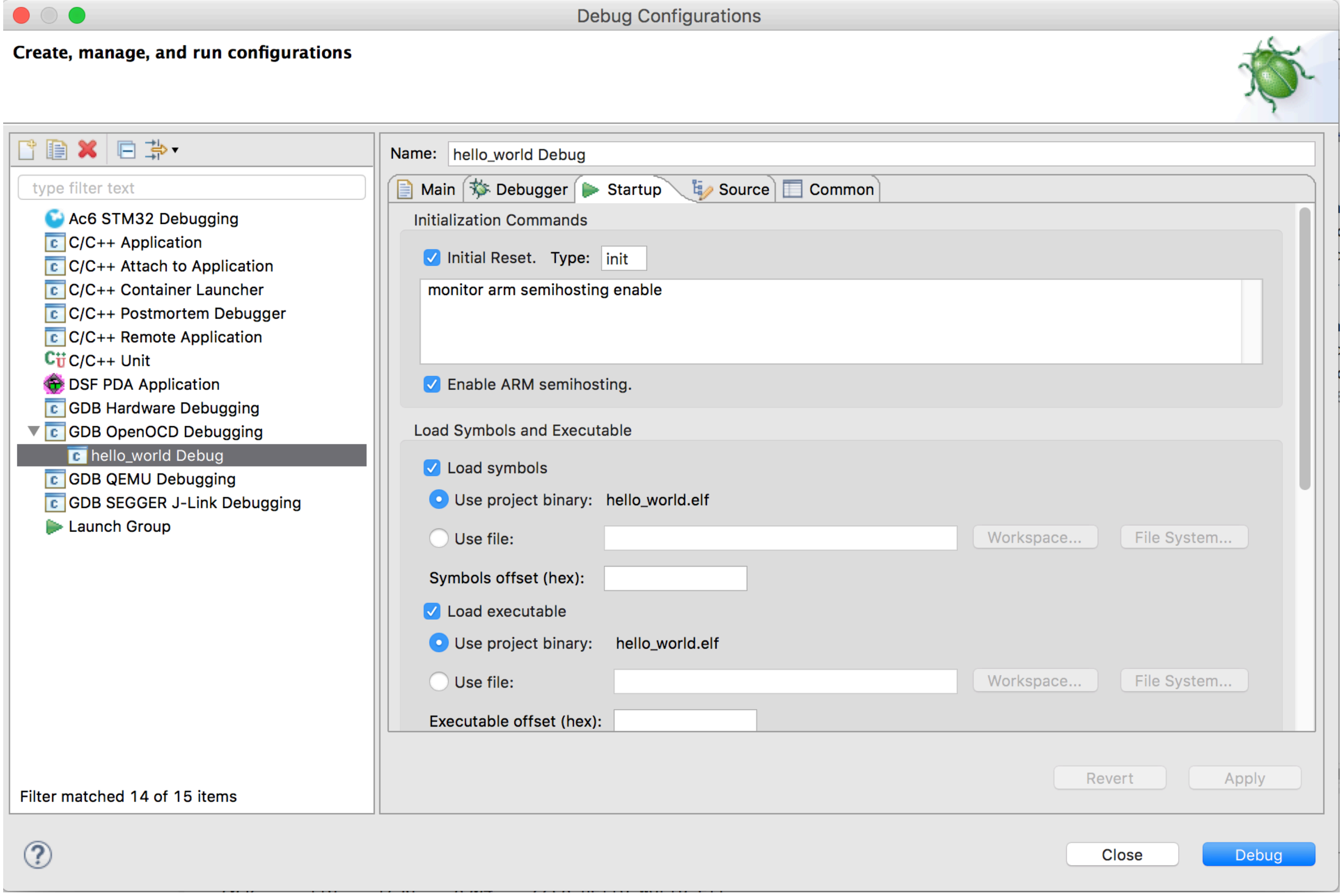
GDB Client Setup

Executable: /usr/local/gcc-arm-none-eabi-5_2-2015q4/bin/arm-none-eabi-

Other options:

Commands: set mem inaccessible-by-default off

Under the **Startup tab -> Initialization Commands -> Enable ARM semihosting** is checked



Name: hello_world Debug

Initialization Commands

☑ Initial Reset. Type: init

```
monitor arm semihosting enable
```

☑ Enable ARM semihosting.

Load Symbols and Executable

☑ Load symbols

● Use project binary: hello_world.elf

○ Use file:

Symbols offset (hex):

☑ Load executable

● Use project binary: hello_world.elf

○ Use file:

Executable offset (hex):

The debug session can now be run and `printf()` output to the console.