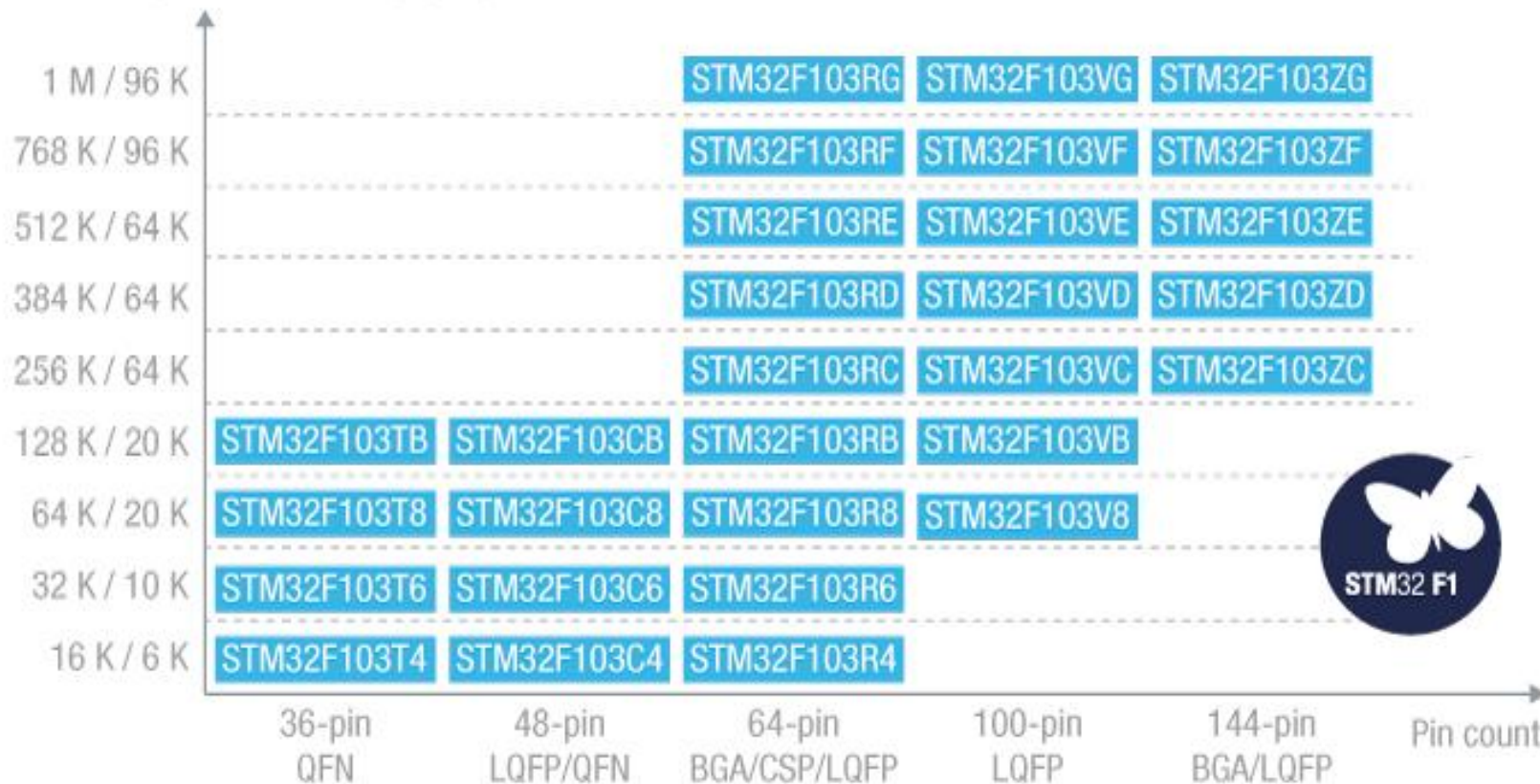


# Программирование на языке C

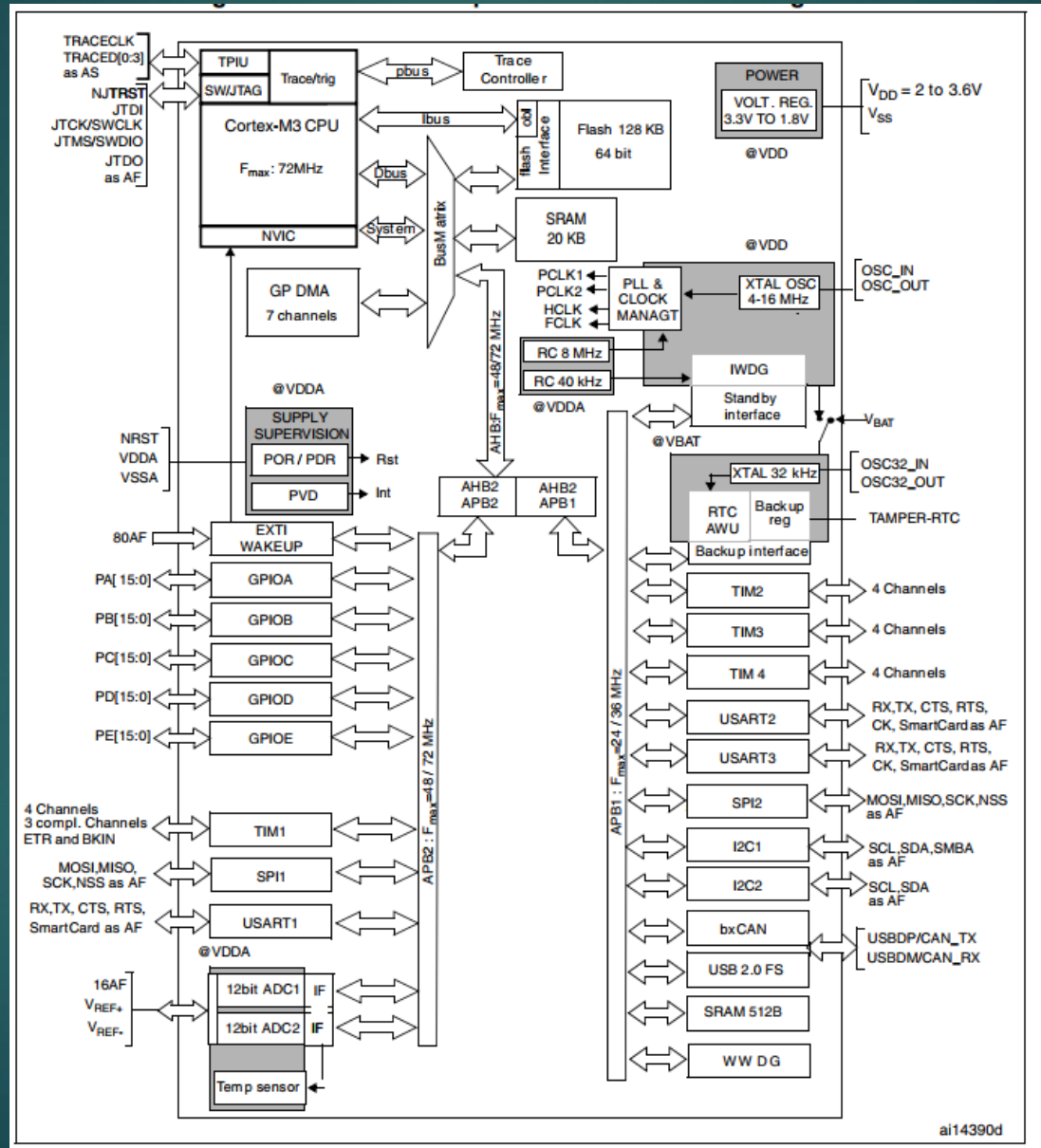
ЛЕКЦИЯ 3. МИКРОКОНТРОЛЛЕРЫ STM32F1

# Семейство stm32f1

Flash memory size / RAM size (bytes)

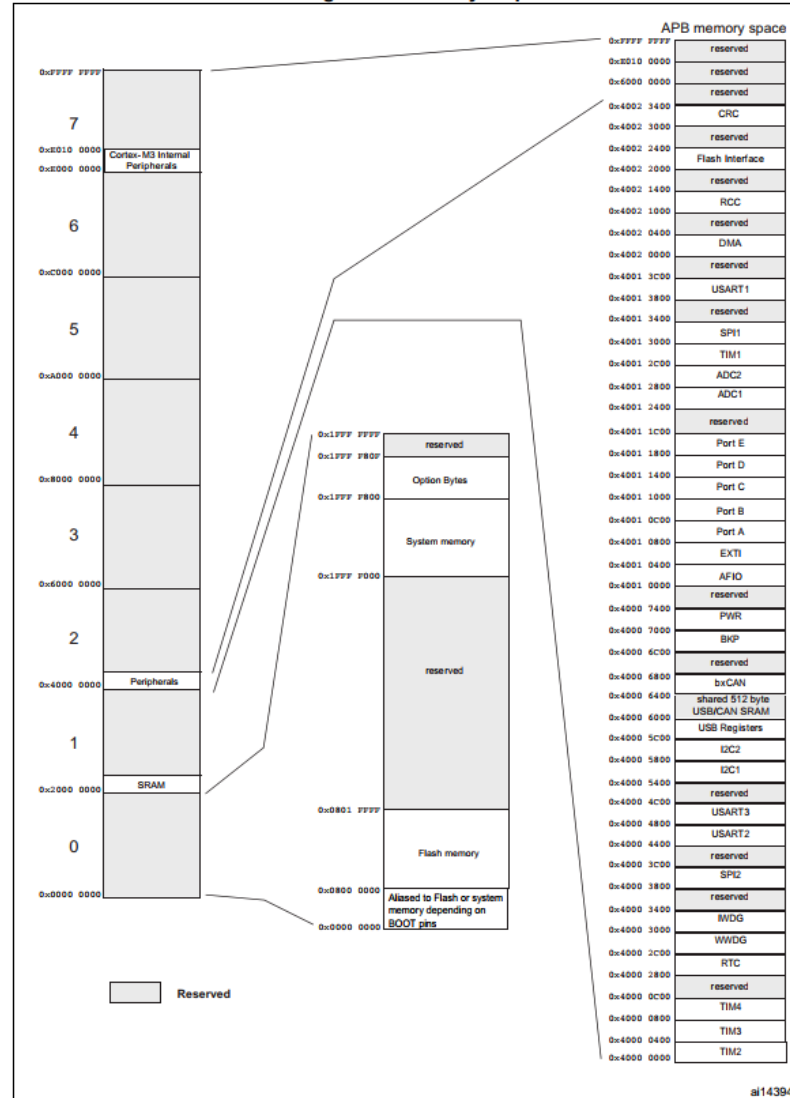


# Структура stm32f1xx



# Карта памяти stm32f1

Figure 11. Memory map



# Необходимые инструменты

- ▶ Компилятор и стандартная библиотека: arm-none-eabi-gcc, newlib
- ▶ Программатор-отладчик: stlinkv2
- ▶ Программа управления программатором-отладчиком: OpenOCD
- ▶ **Библиотека stdperiph от stmicro**

(Опционально) среда разработки eclipse + eclipse-mcu-plugin

# Необходимая документация

- ▶ Документация на МК:

[http://www.st.com/content/st\\_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32-mainstream-mcus/stm32f1-series/stm32f103/stm32f103c8.html](http://www.st.com/content/st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32-mainstream-mcus/stm32f1-series/stm32f103/stm32f103c8.html)

В том числе: datasheet и reference manual

- ▶ Документация на стандартную библиотеку:

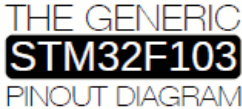
<https://sourceware.org/newlib/>

**Документация на библиотеку stdperiph:**

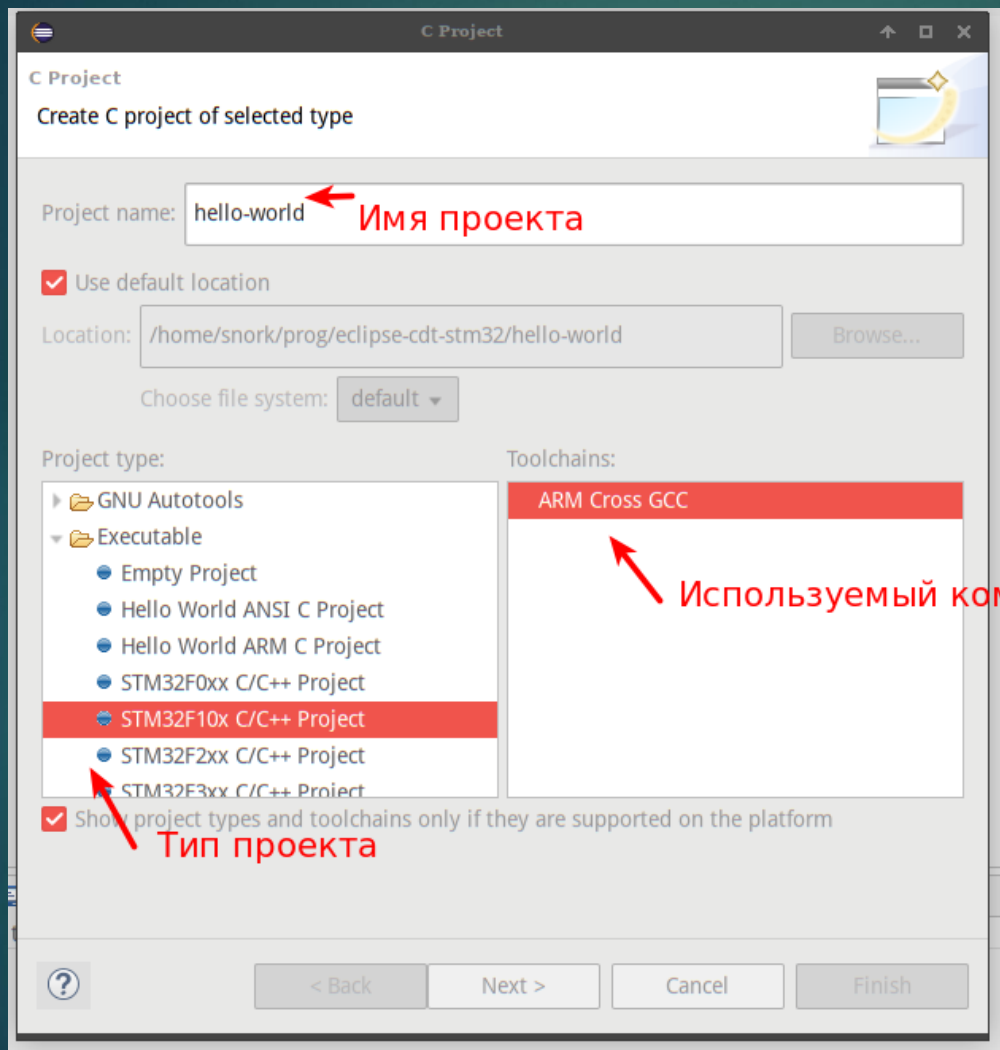
<http://stm32.kosyak.info/doc/>

- ▶ Документация на openocd и прочее





# Создание и настройка проекта STM32



Проект для микроконтроллера создается аналогично проекту для настольного компьютера

Нужно только выбрать другой тип проекта



# Создание и настройка проекта STM32

Target processor settings

Select the target processor family and define flash and RAM sizes.

Chip family: STM32f10x Medium Density

Flash size (kB): 64

RAM size (kB): 20

External clock (Hz): 8000000

Content: Empty (add your own content)

Use system calls: POSIX (system calls implemented by application code)

Trace output: Semihosting DEBUG channel

Check some warnings ☒

Check most warnings ☐

Enable -Werror ☐

Use -Og on debug ☒

Use newlib nano ☒

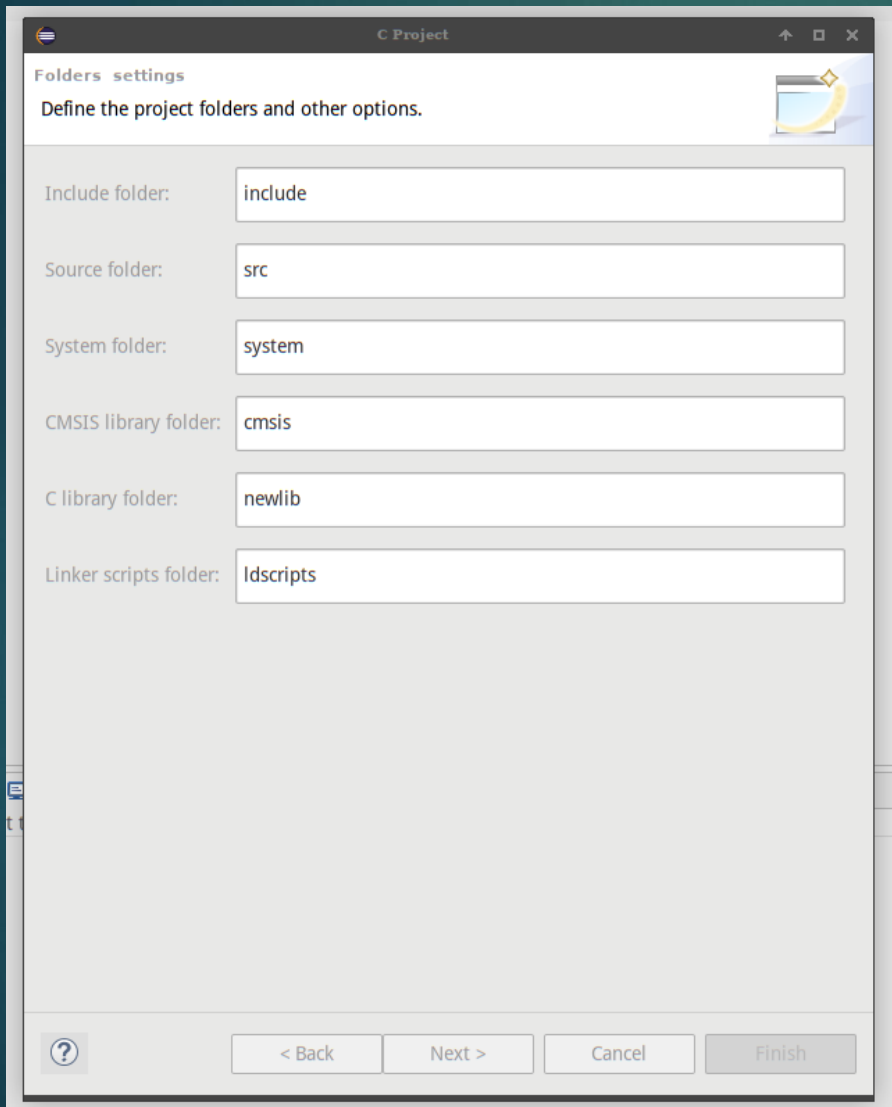
Exclude unused ☒

Use link optimizations ☐

? < Back Next > Cancel Finish

Настраиваемый целевой МК. Для платы bluepill – настройки нужны такие

# Создание и настройка проекта STM32



C Project

Folders settings

Define the project folders and other options.

Include folder: include

Source folder: src

System folder: system

CMSIS library folder: cmsis

C library folder: newlib

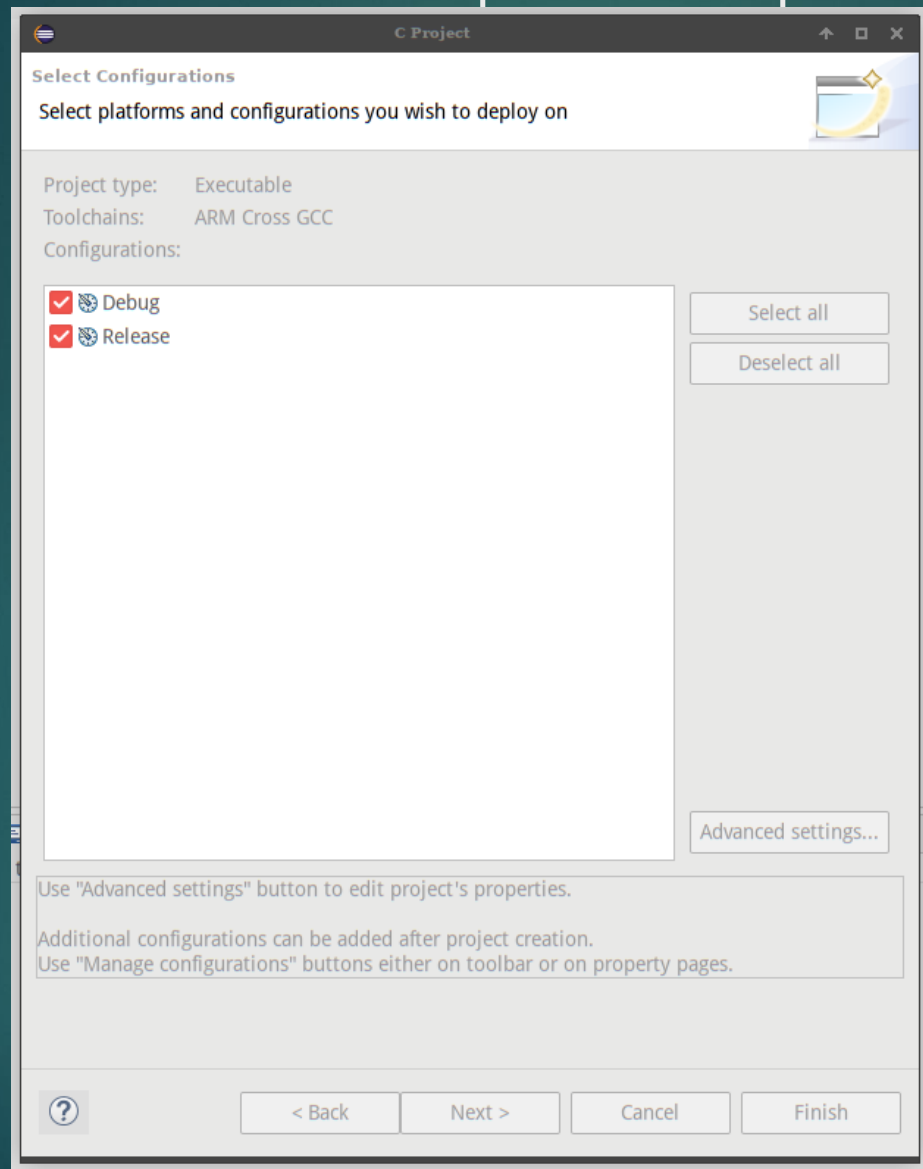
Linker scripts folder: ldscripts

? < Back Next > Cancel Finish

Настраиваем «стандартные»  
каталоги проекта.

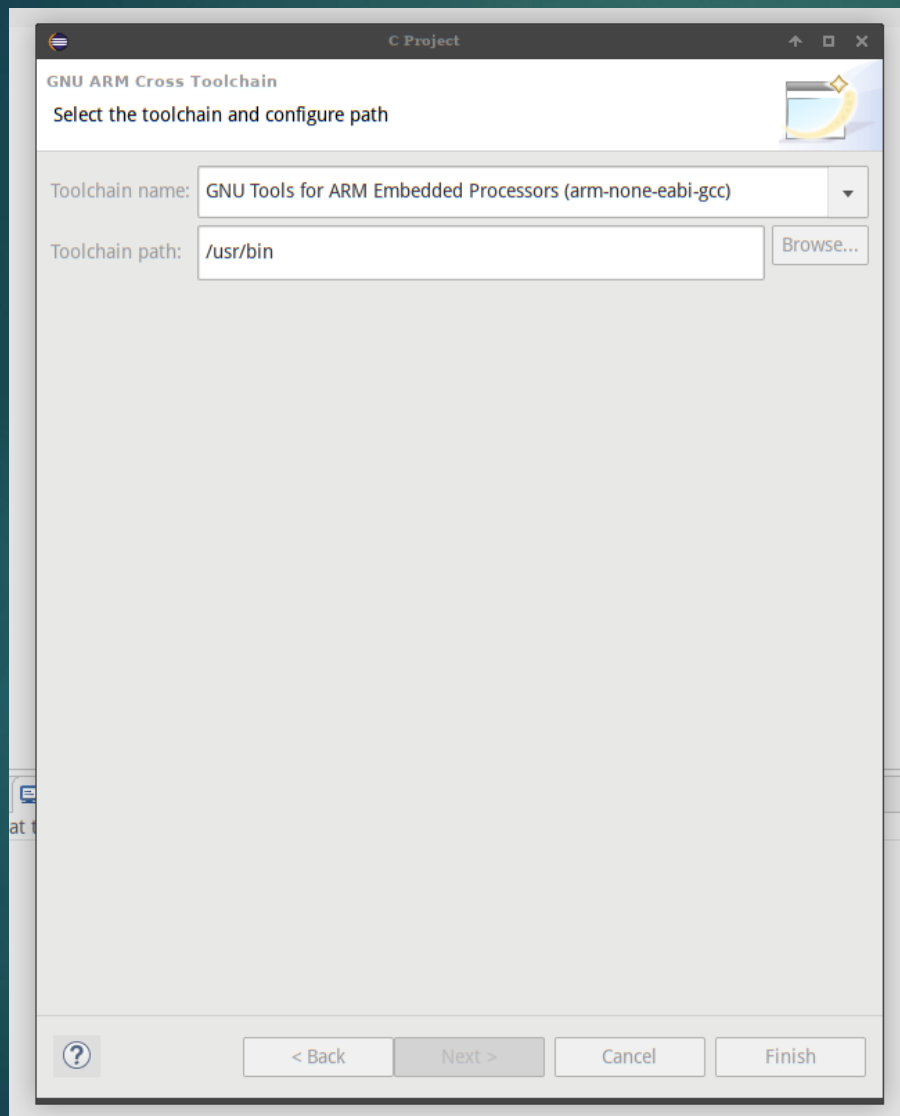
Проще всего оставить как  
есть

# Создание и настройка проекта STM32



Оставляем обе конфигурации, так как у нас есть внутрисхемный отладчик!

# Создание и настройка проекта STM32



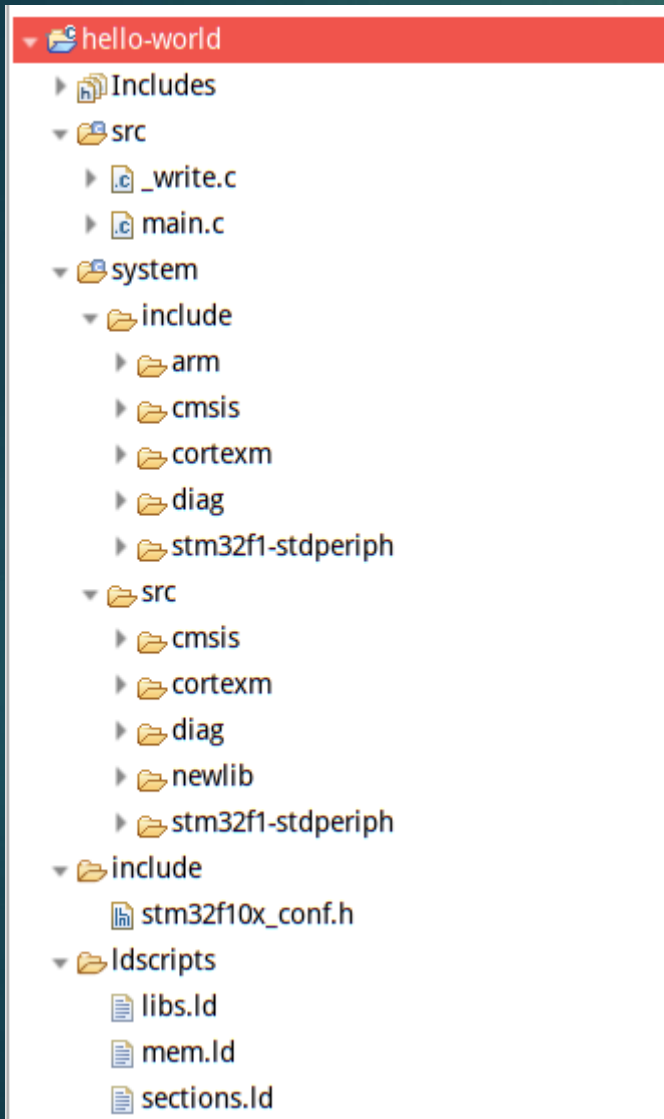
Настраиваем путь до компилятора.

На виртуальной машине компилятор установлен в систему, поэтому тут нужно просто указать верный тип компилятора:

`arm-none-eabi-gcc`

Путь можно писать `/usr/bin` а можно и не писать

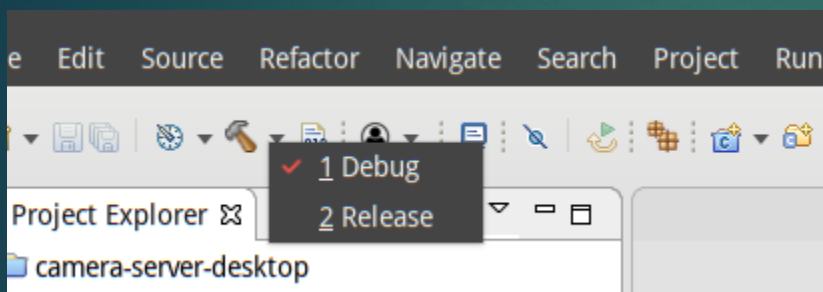
# Создание и настройка проекта STM32



«стандартное» дерево каталогов.

Поскольку тулчейн arm-none-eabi-gcc подходит для огромной кучи МК, то для запуска программы именно на stm32 многие системные файлы не «вшиты» в компилятор и находятся прямо в проекте

# Компиляция и прошивка



Сборка проводится так же как всегда

Убедитесь, что собираете отладочную конфигурацию для отладки

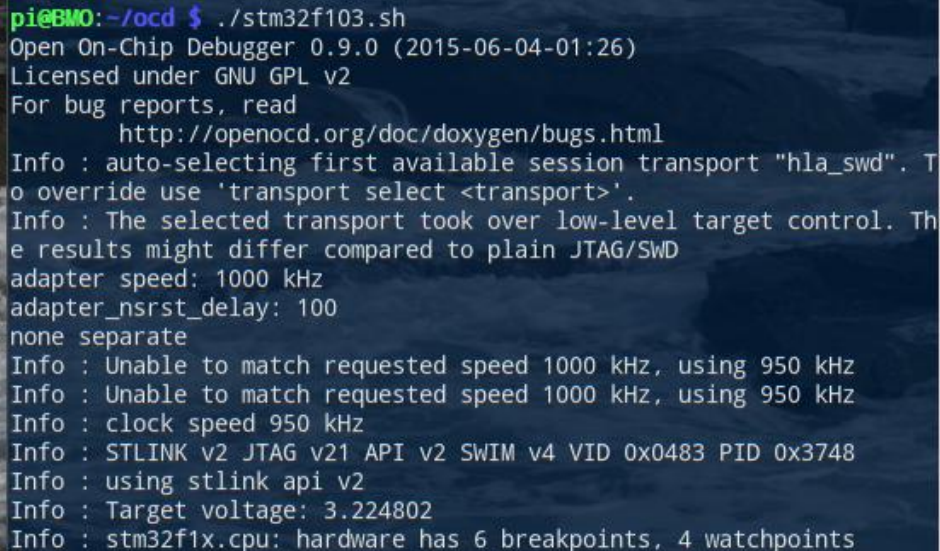
A screenshot of the CDT Build Console window in Eclipse. The window title is 'CDT Build Console [hello-world]'. It shows the output of two build commands. The first command is 'Invoking: GNU ARM Cross Create Flash Image' which runs 'arm-none-eabi-objcopy -O ihex "hello-world.elf" "hello-world.hex"' and finishes with 'Finished building: hello-world.hex'. The second command is 'Invoking: GNU ARM Cross Print Size' which runs 'arm-none-eabi-size --format=berkeley "hello-world.elf"' and produces a table of sizes for different sections of the ELF file. The table has columns for 'text', 'data', 'bss', 'dec', 'hex', and 'filename'. The values are: text=4359, data=176, bss=412, dec=4947, hex=1353, filename=hello-world.elf. The console also shows 'Finished building: hello-world.siz' and a final status message '23:45:43 Build Finished (took 4s.357ms)'.

← Компиляция «проекта здорового человека»



# Компиляция и прошивка

## Нормально запущенный сервер



```
pie@mo:~/ocd $ ./stm32f103.sh
Open On-Chip Debugger 0.9.0 (2015-06-04-01:26)
Licensed under GNU GPL v2
For bug reports, read
  http://openocd.org/doc/doxygen/bugs.html
Info : auto-selecting first available session transport "hla_swd". To
o override use 'transport select <transport>'.
Info : The selected transport took over low-level target control. The
e results might differ compared to plain JTAG/SWD
adapter speed: 1000 kHz
adapter_nsrst_delay: 100
none separate
Info : Unable to match requested speed 1000 kHz, using 950 kHz
Info : Unable to match requested speed 1000 kHz, using 950 kHz
Info : clock speed 950 kHz
Info : STLINK v2 JTAG v21 API v2 SWIM v4 VID 0x0483 PID 0x3748
Info : using stlink api v2
Info : Target voltage: 3.224802
Info : stm32f1x.cpu: hardware has 6 breakpoints, 4 watchpoints
```

Прошивку будем делать через отладчик

Чтобы запустить отладчик, сперва нужно запустить GDB сервер.

Сервер может быть запущен на вашем компьютере или на каком-то другом, к которому подключен программатор.

openocd уже установлена на виртуальной машине

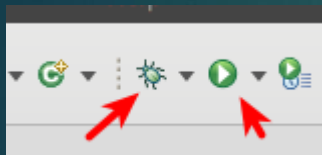
На виртуальных машинах заготовлен специальный скрипт. Команду можно ввести и руками.

Строчка для запуска (нужно ввести в консоли)

```
openocd -f /usr/share/openocd/scripts/interface/stlink-v2.cfg -f /usr/share/openocd/scripts/target/stm32f1x.cfg
```

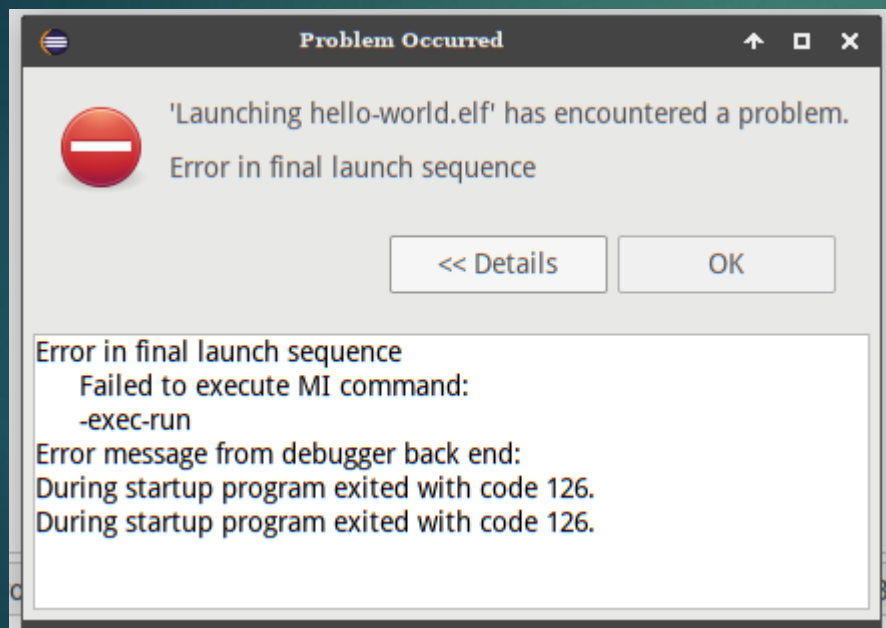
# Компиляция и прошивка

## Кнопки запуска и отладки



Если просто нажать на «жучка» или на «плей», эклипс создаст конфигурацию запуска по умолчанию, которая запустит приложение на вашем компьютере.

Ничего хорошего из этого не выйдет

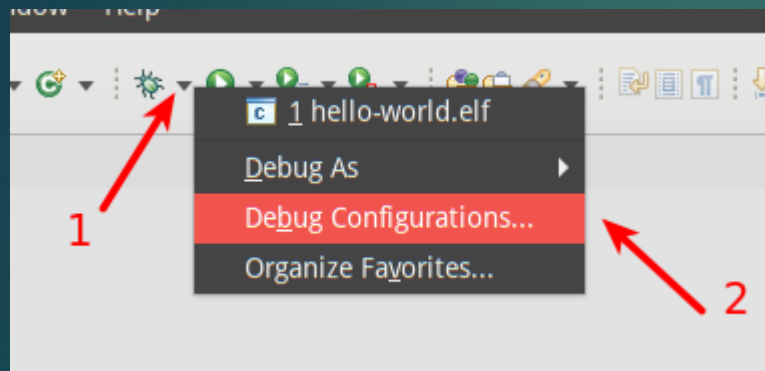


Типичные сообщения об ошибке

```
<terminated> hello-world.elf [C/C++ Application] hello-world.elf
/bin/bash: /home/snork/prog/eclipse-cdt-stm32/hello-world/Debug/hello-world.elf: cannot execute binary file: Exec format error
/bin/bash: /home/snork/prog/eclipse-cdt-stm32/hello-world/Debug/hello-world.elf: Success
```

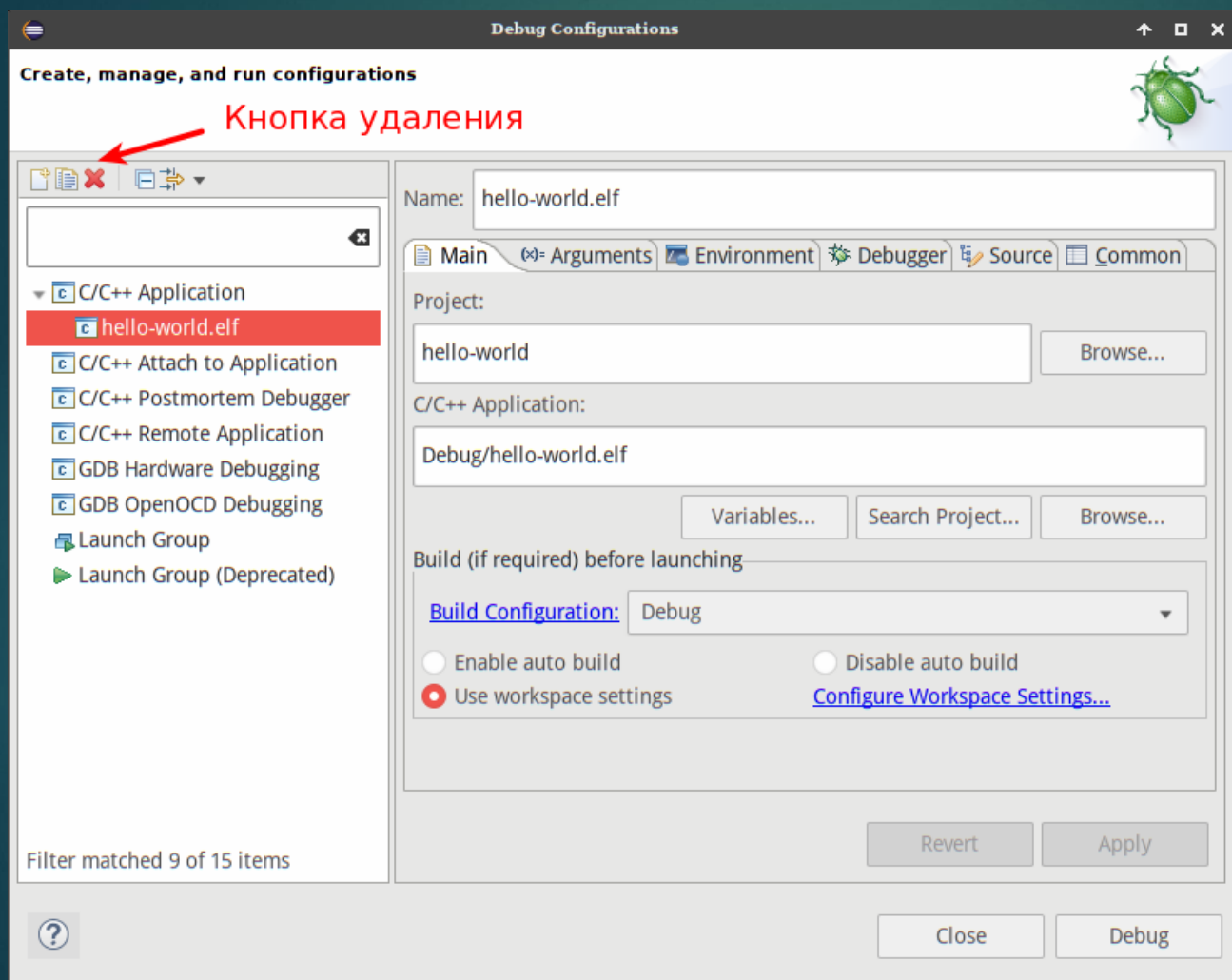
# Компиляция и прошивка

Кнопка входа в меню конфигураций



Вместо этого нам придется  
создать свою конфигурацию

# Компиляция и прошивка



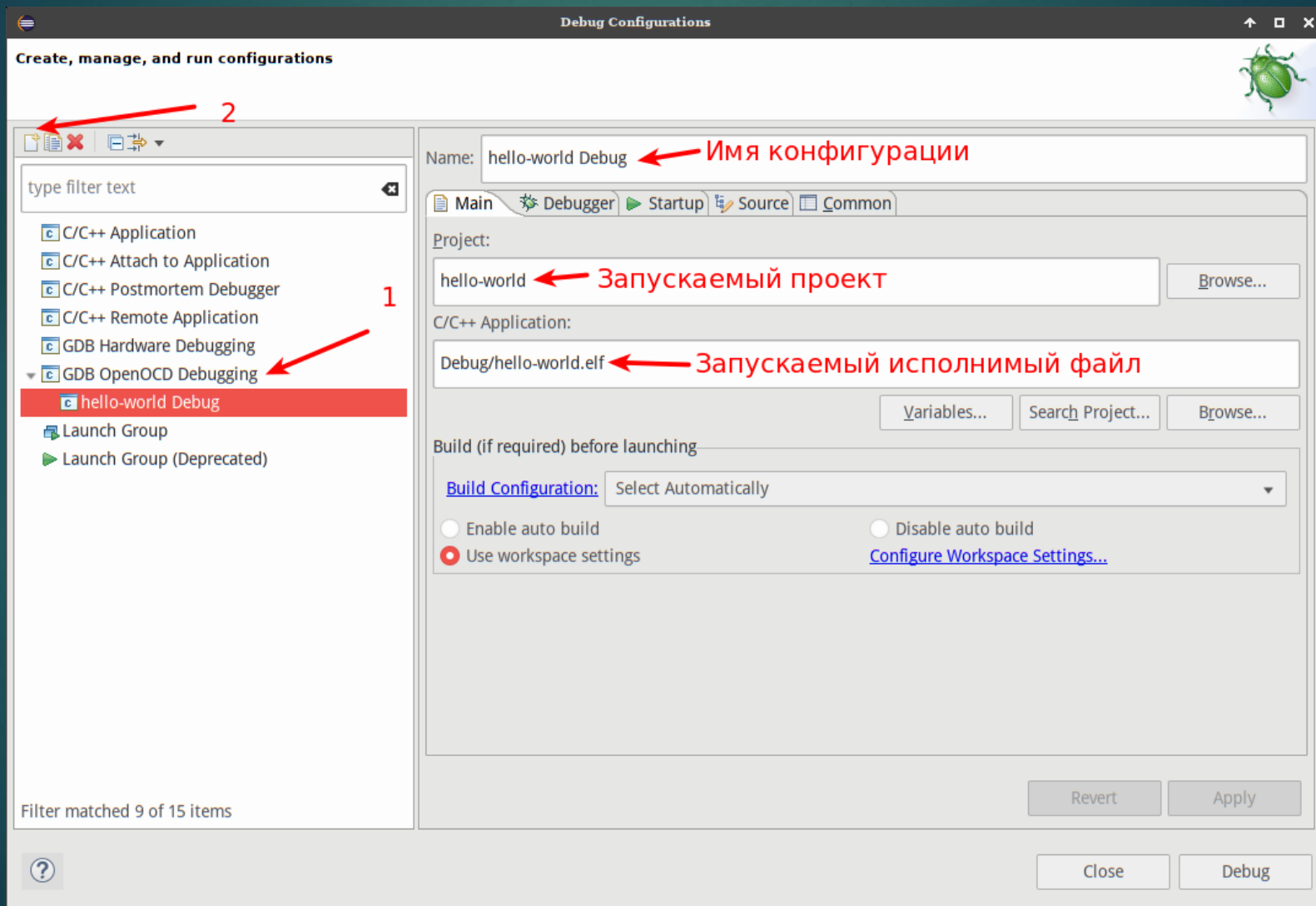
Меню конфигураций

Видна созданная  
элипсом  
конфигурация  
запуска локального  
приложения

Смело удаляем её  
(если она есть)  
нажав на красный  
крестик

# Компиляция и прошивка

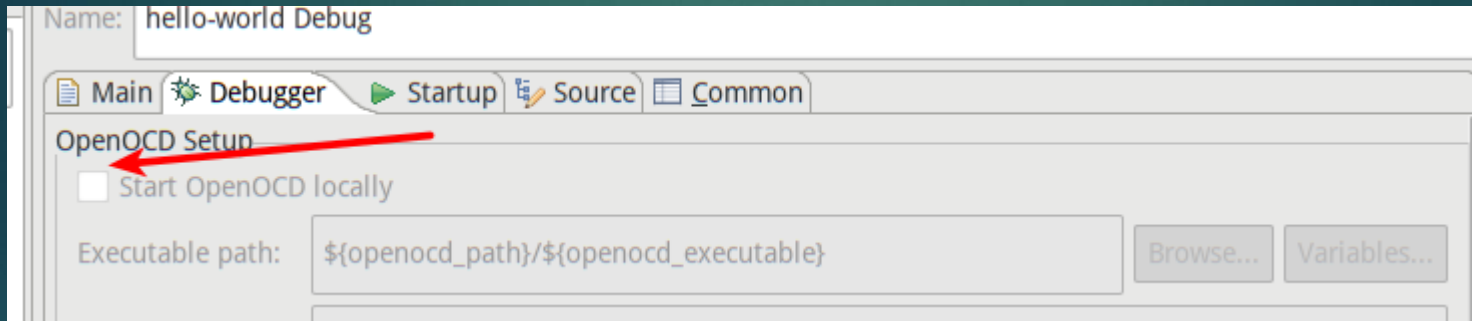
Создаем свою конфигурацию, выбрав категорию GDB OpenOCD Debugging  
И нажав на кнопку «создать новую»



# Компиляция и прошивка

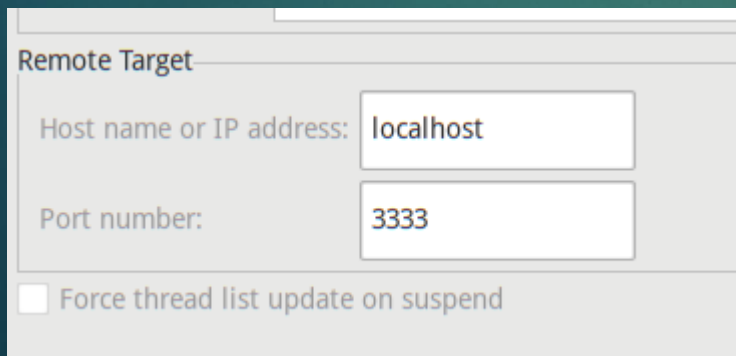
Переходим на вкладку Debugger

1. Снимаем галочку Start OpenOCD locally – запрещаем эклипсу самому запускать openocd (он с этим не очень хорошо справляется)



2. Прокручиваем вниз и проверяем адрес GDB сервера, который подняла OpenOCD

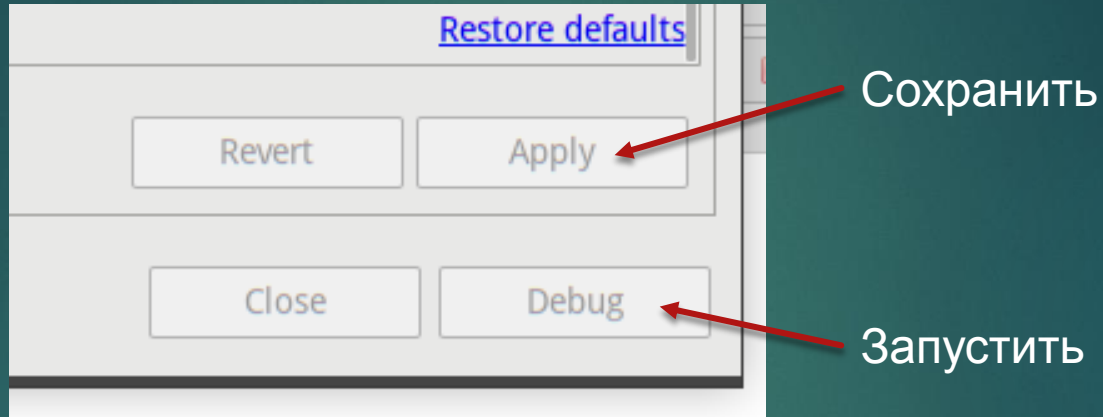
localhost означает дословно «ЭТОТ компьютер»





# Компиляция и прошивка

Когда все готово, жмем снизу apply. Можно немедленно запустить отладку, нажав на Debug



В дальнейшем, конфигурация запуска будет видна в контекстном меню отладки (или запуска)

