

Embedded-Systems Hands-On

Gruppe 4



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Sommersemester 2020
Sheet 2

Task 2.1: Serial Wire Debug

All codes are placed "2_1" subfolder. **pigpio** is required to compile the code. This can be installed with "**apt**" on the raspberry pi or by following these **instructions**. To compile the code the makefile can be executed. To run the code execute "T2" file with "**sudo**".

Task 2.2: GDB server

The objective of this task was to connect to the gdb server initiated by the OpenOCD with a GDB client running on a remote device. Problems arose when gdb running on the Raspberry Pi was able to connect to the server while the gdb on the laptop was not able to connect. The cause of this problem was that OpenOCD only listens to the loopback IP address 127.0.0.1 by default. The solution was simple. According to the documentation of OpenOCD the configuration file can be used to bind the initiated servers to an arbitrary IP-Address. Launching OpenOCD with the modified configuration file allowed the gdb on the laptop to attach to the cortex-m0 gdb session.

```
interface sysfsgpio
sysfsgpio_swdio_num 6
sysfsgpio_swclk_num 5

transport select swd

source [find target/stm32f0x.cfg]

bindto 192.168.0.183

init
targets
```

Figure 1: Modified Configuration File to bind gdb server to a no loopback IP address

```
(gdb) info all-registers
r0      0x0      0
r1      0x1      1
r2      0x0      0
r3      0x2000132c 536875820
r4      0x8002341 134226753
r5      0x0      0
r6      0x55555555 1431655765
r7      0x55555555 1431655765
r8      0x55555555 1431655765
r9      0x55555555 1431655765
r10     0x55555555 1431655765
r11     0x55555555 1431655765
r12     0xe000e100 -536813312
sp      0x20001230 0x20001230
lr      0x80001c3 134218179
pc      0x8002340 0x8002340
xPSR    0x41000000 1090519040
msp     0x20000400 0x20000400
psp     0x20001230 0x20001230
primask 0x0      0
basepri 0x0      0
faultmask 0x0    0
control 0x2      2
```

Figure 2: GDB output after info all-registers command

Task 2.3: Debugging IDE

For the task we chose to use the MCU Eclipse.

1.

Eclipse Embedded CDT
A family of Eclipse CDT extensions and tools for GNU ARM & RISC-V development

/ GNU MCU → Eclipse Embedded CDT!

In 2020, the GNU MCU/ARM Eclipse project was migrated to the Eclipse Foundation as *Eclipse Embedded CDT (C/C++ Development Tools)*; starting with **v5.1.1**, all future releases will be available from the Eclipse Foundation.

Eclipse Embedded CDT is an open source project that includes a family of Eclipse plug-ins and tools for multi-platform embedded ARM and RISC-V development, based on GNU toolchains. This project is hosted on GitHub. The former project was hosted on GitHub and SourceForge.

// RISC-V

Starting with Eclipse plug-ins v4.x, the project was enhanced with support for RISC-V devices, thus the new **MCU** name, more appropriate for a multi-platform project. For more details, see the [RISC-V Corner](#) and [xPack GNU RISC-V Embedded GCC](#) page.

// The Eclipse Embedded CDT plug-ins

These plug-ins provide Eclipse CDT (C/C++ Development Tooling) extensions for GNU embedded toolchains like [GNU Tools for ARM Embedded Processors](#), [Linaro](#), [xPack GNU RISC-V Embedded GCC](#), etc.

In short, the Eclipse Embedded CDT plug-ins allow to create, build, debug and in general to **manage ARM & RISC-V projects** (executables and static/shared libraries, in both 32 and 64-bit versions) with the Eclipse framework (currently tested up to Eclipse 4.10 2018-12). The plug-ins run on Windows, macOS and GNU/Linux. For more details please visit the [Features](#) page.

Latest News

- Eclipse Embedded CDT plug-ins v5.1.1-202007271821 released
- GNU MCU Eclipse plug-ins v4.7.2-202001271244 released
- GNU MCU Eclipse plug-ins v4.7.1-201911052135 released
- The GNU MCU Eclipse project has new forums
- GNU MCU Eclipse plug-ins v4.6.1-201909231407 released

GNU MCU Eclipse Home

Features

Downloads

- Overview
- Plug-ins (releases)
- Windows Build Tools (releases)
- xPack QEMU Arm (releases)
- xPack OpenOCD (releases)

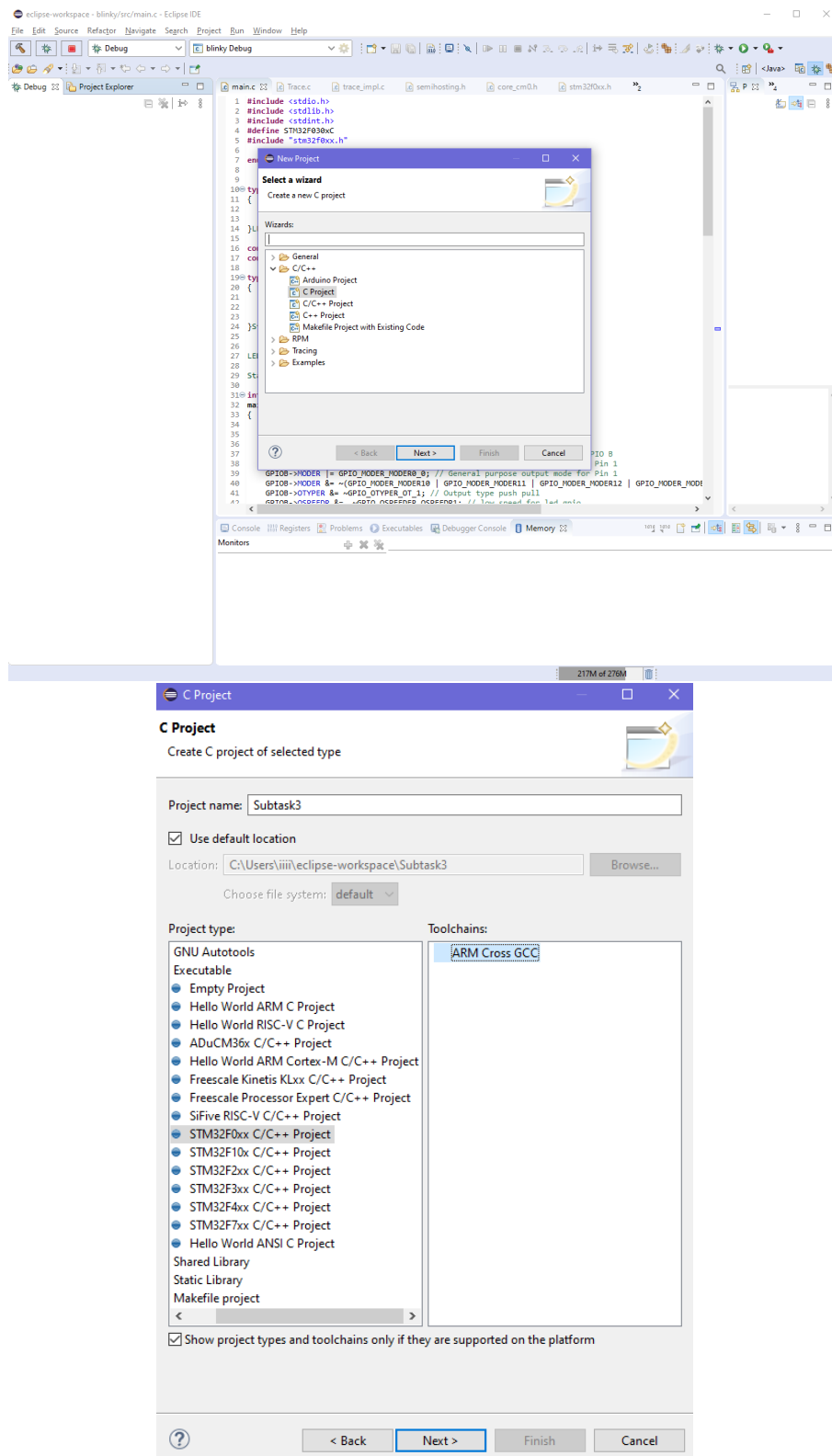
Install

- Overview
- ARM Toolchain(s) install
- RISC-V Toolchain(s) install
- Windows build tools (make & rm) install
- Debugging binaries
 - Overview
 - SEGGER J-Link install
 - xPack OpenOCD install
 - xPack QEMU Arm install
- Eclipse plug-ins install

Important

We followed the instructions on the official **MCU Eclipse** website to install Eclipse MCU.

2.



After installing and opening Eclipse, we created a new STM32F0xx C/C++ project.

3.

C Project

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

Chip family: STM32F030

Flash size (kB): 256

RAM size (kB): 32

Clock (Hz): 48000000

Content: Blinky (blink a led)

Use system calls: Freestanding (no POSIX system calls)

Trace output: Semihosting DEBUG channel

Check some warnings ☒

Check most warnings ☐

Enable -Werror ☐

Use -Og on debug ☒

Use newlib nano ☒

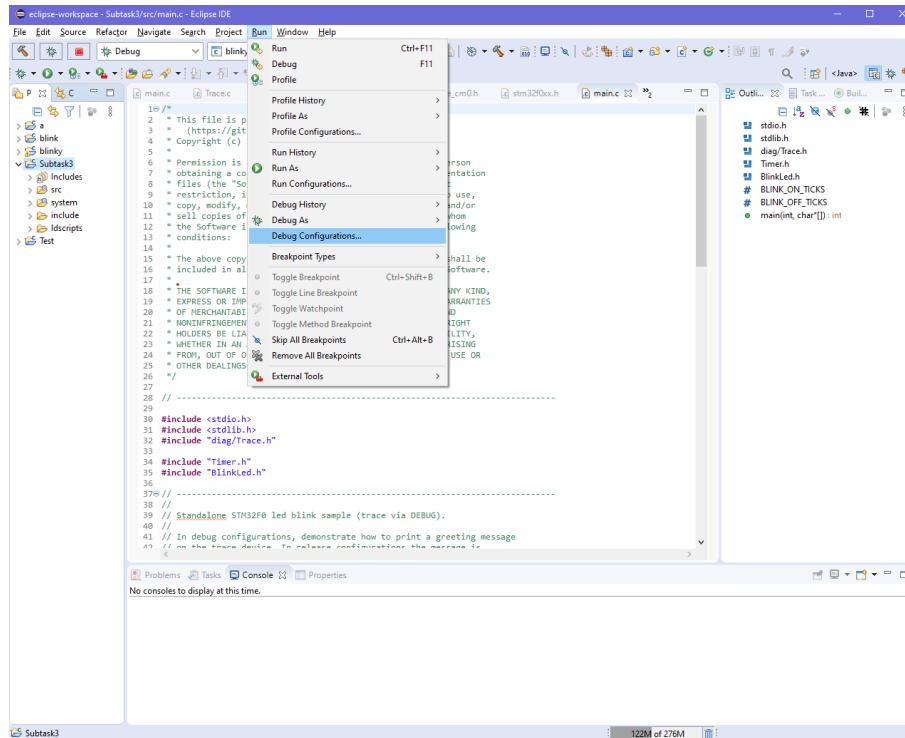
Exclude unused ☒

Use link optimizations ☐

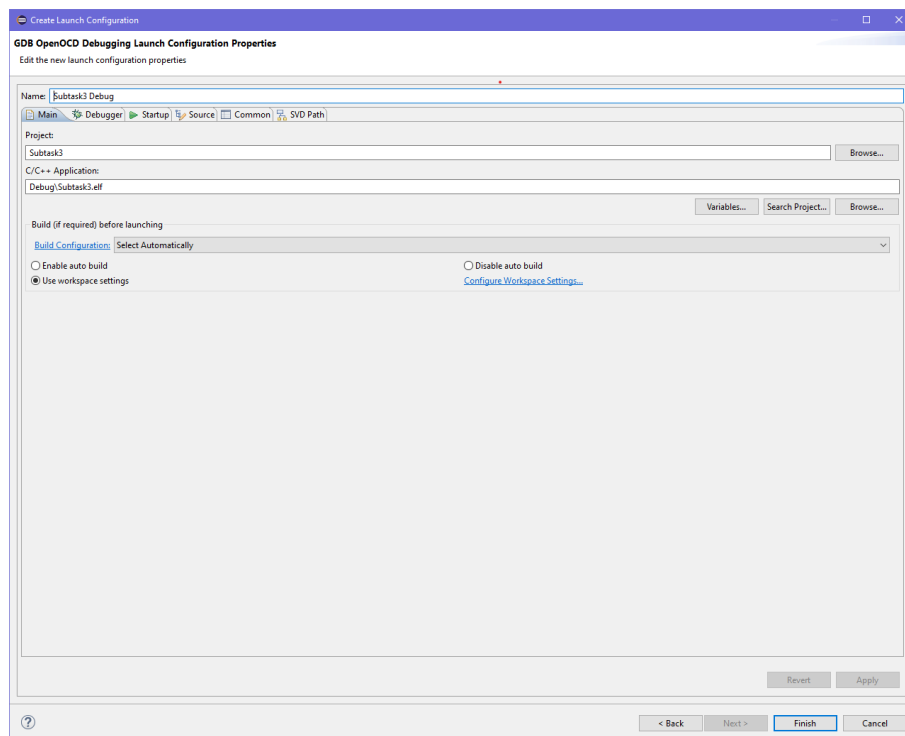
< Back Next > Finish Cancel

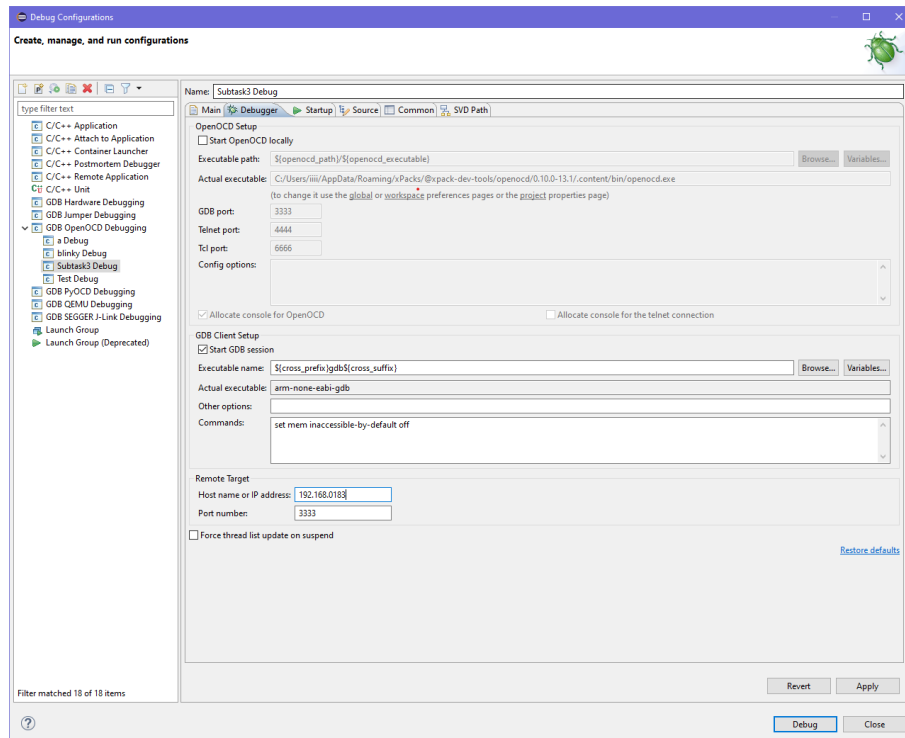
We then had to set the specification according to our Cortex-M0(STM32F030CC) which can be extracted from the datasheets.

4.



We then had to setup our new debug configurations like the following.







7.

The screenshot shows the OpenOCD GUI with a terminal window displaying the following output:

```

Error: Translation from khz to jtag_speed not implemented
Error executing event reset-start on target stm32f0x.cpu:
embedded:startup.tcl:279: Error:
in procedure 'ocd_process_reset'
in procedure 'ocd_process_reset_inner' called at file "embedded:startup.tcl", line 279
target halted due to debug-request, current mode: Thread
xPSR: 0xc1000000 pc: 0x080001d8 msp: 0x20008000, semihosting
Error: Translation from khz to jtag_speed not implemented
Error executing event reset-init on target stm32f0x.cpu:
embedded:startup.tcl:279: Error:
in procedure 'ocd_process_reset'
in procedure 'ocd_process_reset_inner' called at file "embedded:startup.tcl", line 279
Info: Padding image section 0 at 0x08001dce with 2 bytes
Error: Translation from khz to jtag_speed not implemented
Error executing event reset-start on target stm32f0x.cpu:
embedded:startup.tcl:279: Error:
in procedure 'ocd_process_reset'
in procedure 'ocd_process_reset_inner' called at file "embedded:startup.tcl", line 279
target halted due to debug-request, current mode: Thread
xPSR: 0xc1000000 pc: 0x080001d8 msp: 0x20008000, semihosting
Error: Translation from khz to jtag_speed not implemented
Error executing event reset-start on target stm32f0x.cpu:
embedded:startup.tcl:279: Error:
in procedure 'ocd_process_reset'
in procedure 'ocd_process_reset_inner' called at file "embedded:startup.tcl", line 279
target halted due to debug-request, current mode: Thread
xPSR: 0xc1000000 pc: 0x080001d8 msp: 0x20008000, semihosting

==== arm v7e registers
(0) r0 (/32): 0xffffffff
(1) r1 (/32): 0xffffffff
(2) r2 (/32): 0xffffffff
(3) r3 (/32): 0xffffffff
(4) r4 (/32): 0xffffffff
(5) r5 (/32): 0xffffffff
(6) r6 (/32): 0xffffffff
(7) r7 (/32): 0xffffffff
(8) r8 (/32): 0xffffffff
(9) r9 (/32): 0xffffffff
(10) r10 (/32): 0xffffffff
(11) r11 (/32): 0xffffffff
(12) r12 (/32): 0xffffffff
(13) sp (/32): 0x20008000
(14) lr (/32): 0xffffffff
(15) pc (/32): 0x080001d8
(16) xPSR (/32): 0xc1000000
(17) msp (/32): 0x20008000
(18) psp (/32): 0xffffffff
(19) primask (/1): 0x00
(20) basepri (/8): 0x00
(21) faultmask (/1): 0x00
(22) control (/2): 0x00

==== cortex-M DWT registers
Hello ARM World!
System clock: 8000000 Hz
  
```

OpenOCD output while stepping through the Blinky example project.