CEC Continuing Education Center

DesignNews

Embedded Software using RUST

DAY 3 : “Hello Rust!”, using the STM32F3

Sponsored by

Digi-Key ELECTRONICS

BENINGO EMBEDDED GROUP

informa markets

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Group Chat’ by maximizing the chat widget in your dock.

THE SPEAKER

Jacob Beningo

Visit 'Lecturer Profile'

Beningo Embedded Group - President

Focus: Embedded Software Consulting

An independent consultant who specializes in the design of real-time, microcontroller based embedded software.
He has published two books:

- Reusable Firmware Development
- MicroPython Projects
- Embedded Software Design

Writes a weekly blog for DesignNews.com focused on embedded system design techniques and challenges.

Visit www.beningo.com to learn more ...

Visit 'Lecturer Profile' in your console for more details.

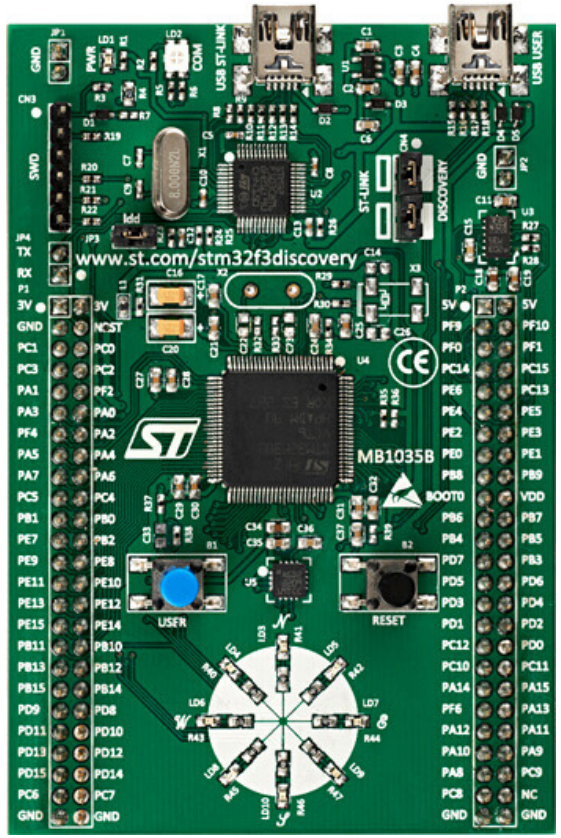
Course Sessions

- Introduction to Rust for Embedded Systems
- ”Hello Rust!”, using QEMU
- **“Hello Rust!”, using the STM32F3**
- Interfacing to Peripherals in Rust
- Becoming a Rust Expert

1 The STM32F3 Discovery Board

The STM32F3 Discovery Board

- A Cortex-M4F core that includes a single precision FPU
- 256 KiB of Flash located at address 0x0800_0000.
- 40 KiB of RAM located at address 0x2000_0000. (There's another RAM region but for simplicity we'll ignore it).



Will you be following along with a discovery board either during the class or after?

- Yes
- No
- Undecided

2 Creating “Hello Rust!”

Creating the Project

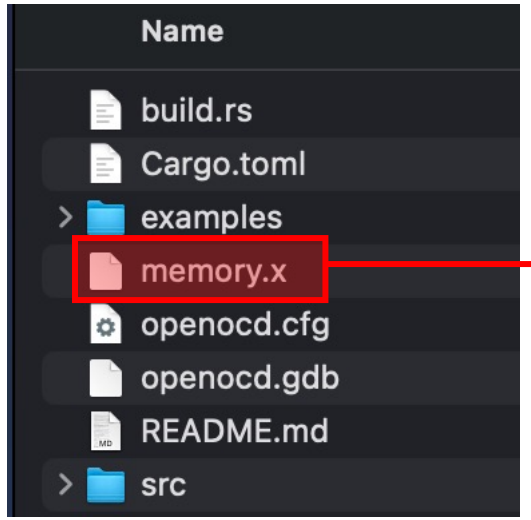
```
beningo@Jacobs-MacBook-Pro rust % cargo generate --git https://github.com/rust-embedded/cortex-m-quickstart
🤷   Project Name: stm32f3_hello
⚠   Renaming project called `stm32f3_hello` to `stm32f3-hello`...
🔧   Destination: /Users/beningo/rust/stm32f3-hello ...
🔧   project-name: stm32f3-hello ...
🔧   Generating template ...
[ 1/25]   Done: .cargo/config.toml
[ 2/25]   Done: .cargo
[ 3/25]   Done: .gitignore
[ 4/25]   Done: .vscode/README.md
[ 5/25]   Done: .vscode/extensions.json
[ 6/25]   Done: .vscode/launch.json
[ 7/25]   Done: .vscode/tasks.json
[ 8/25]   Done: .vscode
[ 9/25]   Done: Cargo.toml
[10/25]   Done: README.md
[11/25]   Done: build.rs
[12/25]   Done: examples/allocator.rs
[13/25]   Done: examples/crash.rs
[14/25]   Done: examples/device.rs
[15/25]   Done: examples/exception.rs
[16/25]   Done: examples/hello.rs
[17/25]   Done: examples/itm.rs
[18/25]   Done: examples/panic.rs
[19/25]   Done: examples/test_on_host.rs
[20/25]   Done: examples
[21/25]   Done: memory.x
[22/25]   Done: openocd.cfg
[23/25]   Done: openocd.gdb
[24/25]   Done: src/main.rs
[25/25]   Done: src
🔧   Moving generated files into: `/Users/beningo/rust/stm32f3-hello`...
💡   Initializing a fresh Git repository
✨   Done! New project created /Users/beningo/rust/stm32f3-hello
beningo@Jacobs-MacBook-Pro rust %
```

Configuring the Project

Set the correct target in .cargo/config.toml

```
[build]
# Pick ONE of these compilation targets
# target = "thumbv6m-none-eabi"        # Cortex-M0 and Cortex-M0+
# target = "thumbv7m-none-eabi"        # Cortex-M3
# target = "thumbv7em-none-eabi"       # Cortex-M4 and Cortex-M7 (no FPU)
  target = "thumbv7em-none-eabihf"     # Cortex-M4F and Cortex-M7F (with FPU)
# target = "thumbv8m.base-none-eabi"   # Cortex-M23
# target = "thumbv8m.main-none-eabi"   # Cortex-M33 (no FPU)
# target = "thumbv8m.main-none-eabihf" # Cortex-M33 (with FPU)
```

Configure Project




```
MEMORY
{
  /* NOTE 1 K = 1 KiBi = 1024 bytes */
  FLASH : ORIGIN = 0x08000000, LENGTH = 256K
  RAM : ORIGIN = 0x20000000, LENGTH = 40K
}
```

Write the Program

```rust
//! Prints "Hello, world!" on the host console using semihosting

#![no_main]
#![no_std]

use panic_halt as _;

use cortex_m_rt::entry;
use cortex_m_semihosting::{hprintln};

#[entry]
fn main() -> ! {
    hprintln!("Hello Rust!").unwrap();

    // exit QEMU
    // NOTE do not run this on hardware; it can corrupt OpenOCD state
    // debug::exit(debug::EXIT_SUCCESS);

    loop {}
}
```

What command should you run if you change the linker file to ensure the linker changes are included in your program?

- cargo build
- cargo generate
- cargo clean
- cargo run

3 Building and Debugging the Program

Compile the Application

```
beningo@Jacobs-MacBook-Pro stm32f3-hello % cargo build
   Compiling stm32f3-hello v0.1.0 (/Users/beningo/rust/stm32f3-hello)
    Finished dev [unoptimized + debuginfo] target(s) in 0.20s
beningo@Jacobs-MacBook-Pro stm32f3-hello %
```

```
beningo@Jacobs-MacBook-Pro stm32f3-hello % cargo size --bin stm32f3-hello --release -- -A
    Finished release [optimized + debuginfo] target(s) in 0.05s
stm32f3-hello  :
section               size        addr
.vector_table         1024   0x8000000
.text                  864   0x8000400
.rodata                 20   0x8000760
.data                    0  0x20000000
.bss                     8  0x20000000
.uninit                  0  0x20000008
.debug_loc             802         0x0
.debug_abbrev         2062         0x0
.debug_info          13743         0x0
.debug_aranges         824         0x0
.debug_ranges         1664         0x0
.debug_str           16881         0x0
.debug_pubnames       5855         0x0
.debug_pubtypes       3182         0x0
.ARM.attributes         58         0x0
.debug_frame          1584         0x0
.debug_line           7704         0x0
.comment                19         0x0
Total                56294
```

```
beningo@Jacobs-MacBook-Pro stm32f3-hello % cargo objdump --bin stm32f3-hello --release -- --disassem
ble --no-show-raw-insn --print-imm-hex
    Finished release [optimized + debuginfo] target(s) in 0.04s

stm32f3-hello:  file format elf32-littlearm

Disassembly of section .text:

08000400 <Reset>:
 8000400:       push    {r7, lr}
 8000402:       mov     r7, sp
 8000404:       bl      0x800054e <__pre_init>  @ imm = #0x146
 8000408:       movw    r0, #0x8
 800040c:       movw    r1, #0x0
 8000410:       movt    r0, #0x2000
 8000414:       movt    r1, #0x2000
 8000418:       cmp     r1, r0
 800041a:       bhs     0x8000446 <Reset+0x46>  @ imm = #0x28
 800041c:       movw    r1, #0x0
 8000420:       movs    r2, #0x0
 8000422:       movt    r1, #0x2000
 8000426:       str     r2, [r1], #4
 800042a:       cmp     r1, r0
 800042c:       bhs     0x8000446 <Reset+0x46>  @ imm = #0x16
 800042e:       str     r2, [r1], #4
 8000432:       cmp     r1, r0
 8000434:       itt     lo
```

Debugging

Test that you can connect to your target device:

```
beningo@Jacobs-MacBook-Pro stm32f3-hello % openocd -f interface/stlink.cfg -f target/stm32f3x.cfg
Open On-Chip Debugger 0.11.0
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 override use 'transport select
 <transport>'.
Info : The selected transport took over low-level target control. The results might differ compared
to plain JTAG/SWD
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : clock speed 1000 kHz
Info : STLINK V2J27M15 (API v2) VID:PID 0483:374B
Info : Target voltage: 2.906461
Info : stm32f3x.cpu: hardware has 6 breakpoints, 4 watchpoints
Info : starting gdb server for stm32f3x.cpu on 3333
Info : Listening on port 3333 for gdb connections
```

Debugging

Terminal 1

stm32f3-hello — openocd — 66×35

```
beningo@Jacobs-MacBook-Pro stm32f3-hello % openocd
Open On-Chip Debugger 0.11.0
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 override use 'transport select <transport>'.
Info : The selected transport took over low-level target control.
The results might differ compared to plain JTAG/SWD
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : clock speed 1000 kHz
Info : STLINK V2J27M15 (API v2) VID:PID 0483:374B
Info : Target voltage: 2.905039
Info : stm32f3x.cpu: hardware has 6 breakpoints, 4 watchpoints
Info : starting gdb server for stm32f3x.cpu on 3333
Info : Listening on port 3333 for gdb connections
```

Terminal 2

stm32f3-hello — arm-none-eabi-gdb -q target/thumbv7em-none-...

```
beningo@Jacobs-MacBook-Pro stm32f3-hello % arm-none-eabi-gdb -q target/t
humbv7em-none-eabihf/debug/stm32f3-hello
Reading symbols from target/thumbv7em-none-eabihf/debug/stm32f3-hello...
(gdb)
```

Debugging

stm32f3-hello — openocd — 66×35

```
beningo@Jacobs-MacBook-Pro stm32f3-hello % openocd
Open On-Chip Debugger 0.11.0
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 override use 'transport select <transport>'.
Info : The selected transport took over low-level target control.
The results might differ compared to plain JTAG/SWD
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : clock speed 1000 kHz
Info : STLINK V2J27M15 (API v2) VID:PID 0483:374B
Info : Target voltage: 2.905039
Info : stm32f3x.cpu: hardware has 6 breakpoints, 4 watchpoints
Info : starting gdb server for stm32f3x.cpu on 3333
Info : Listening on port 3333 for gdb connections
Info : accepting 'gdb' connection on tcp/3333
target halted due to debug-request, current mode: Thread
xPSR: 0x61000000 pc: 0x08006494 psp: 0x20001148
Info : device id = 0x10036422
Info : flash size = 256kbytes
Warn : Prefer GDB command "target extended-remote 3333" instead of
 "target remote 3333"
Info : Unable to match requested speed 1000 kHz, using 950 kHz
Info : Unable to match requested speed 1000 kHz, using 950 kHz
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x0800790c msp: 0x20003ba8
Info : Unable to match requested speed 8000 kHz, using 4000 kHz
Info : Unable to match requested speed 8000 kHz, using 4000 kHz
Info : Unable to match requested speed 1000 kHz, using 950 kHz
Info : Unable to match requested speed 1000 kHz, using 950 kHz
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x08000400 msp: 0x2000a000
```

stm32f3-hello — arm-none-eabi-gdb -q target/thumbv7em-none-...

```
beningo@Jacobs-MacBook-Pro stm32f3-hello % arm-none-eabi-gdb -q target/t
humbv7em-none-eabihf/debug/stm32f3-hello
Reading symbols from target/thumbv7em-none-eabihf/debug/stm32f3-hello...
(gdb) target remote :3333
Remote debugging using :3333
0x08006494 in ?? ()
(gdb) load
Loading section .vector_table, size 0x400 lma 0x8000000
Loading section .text, size 0x1040 lma 0x8000400
Loading section .rodata, size 0x410 lma 0x8001440
Start address 0x08000400, load size 6224
Transfer rate: 14 KB/sec, 2074 bytes/write.
(gdb)
```

Debugging

Monitor using Semihosting

stm32f3-hello — openocd — 66×35

```
Info : accepting 'gdb' connection on tcp/3333
target halted due to debug-request, current mode: Handler HardFaul
t
xPSR: 0x21000003 pc: 0x0800143c msp: 0x20009e98
Info : device id = 0x10036422
Info : flash size = 256kbytes
Warn : Prefer GDB command "target extended-remote 3333" instead of
 "target remote 3333"
Info : Unable to match requested speed 1000 kHz, using 950 kHz
Info : Unable to match requested speed 1000 kHz, using 950 kHz
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x08000400 msp: 0x2000a000
Info : Unable to match requested speed 8000 kHz, using 4000 kHz
Info : Unable to match requested speed 8000 kHz, using 4000 kHz
Info : Unable to match requested speed 1000 kHz, using 950 kHz
Info : Unable to match requested speed 1000 kHz, using 950 kHz
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x08000400 msp: 0x2000a000
semihosting is enabled

Info : halted: PC: 0x0800048e
Info : halted: PC: 0x08000498
Info : halted: PC: 0x0800049c
Info : halted: PC: 0x0800049e
Info : halted: PC: 0x08000502
Hello Rust!
Info : halted: PC: 0x080004a4
Info : halted: PC: 0x080004a6
Info : halted: PC: 0x080004a8
Info : halted: PC: 0x080004ac
Info : halted: PC: 0x080004b0
Info : halted: PC: 0x080004b4
Info : halted: PC: 0x080004bc
Info : halted: PC: 0x080004ba
```

stm32f3-hello — arm-none-eabi-gdb -q target/thumbv7em-none-...

```
Loading section .text, size 0x1040 lma 0x8000400
Loading section .rodata, size 0x410 lma 0x8001440
Start address 0x08000400, load size 6224
Transfer rate: 14 KB/sec, 2074 bytes/write.
(gdb) monitor arm semihosting enable
semihosting is enabled

(gdb) break main
Breakpoint 1 at 0x8000488: file src/main.rs, line 11.
Note: automatically using hardware breakpoints for read-only addresses.
(gdb) continue
Continuing.

Breakpoint 1, stm32f3_hello::__cortex_m_rt_main_trampoline ()
    at src/main.rs:11
11      #[entry]
(gdb) step
halted: PC: 0x0800048e
stm32f3_hello::__cortex_m_rt_main () at src/main.rs:13
13          hprintln!("Hello Rust!").unwrap();
(gdb) next
halted: PC: 0x08000498
halted: PC: 0x0800049c
halted: PC: 0x0800049e
halted: PC: 0x08000502
halted: PC: 0x080004a4
halted: PC: 0x080004a6
halted: PC: 0x080004a8
halted: PC: 0x080004ac
halted: PC: 0x080004b0
halted: PC: 0x080004b4
halted: PC: 0x080004bc
halted: PC: 0x080004ba
19          loop {}
(gdb)
```

What change should be made to the build system to allow “cargo run” to start a debug session?

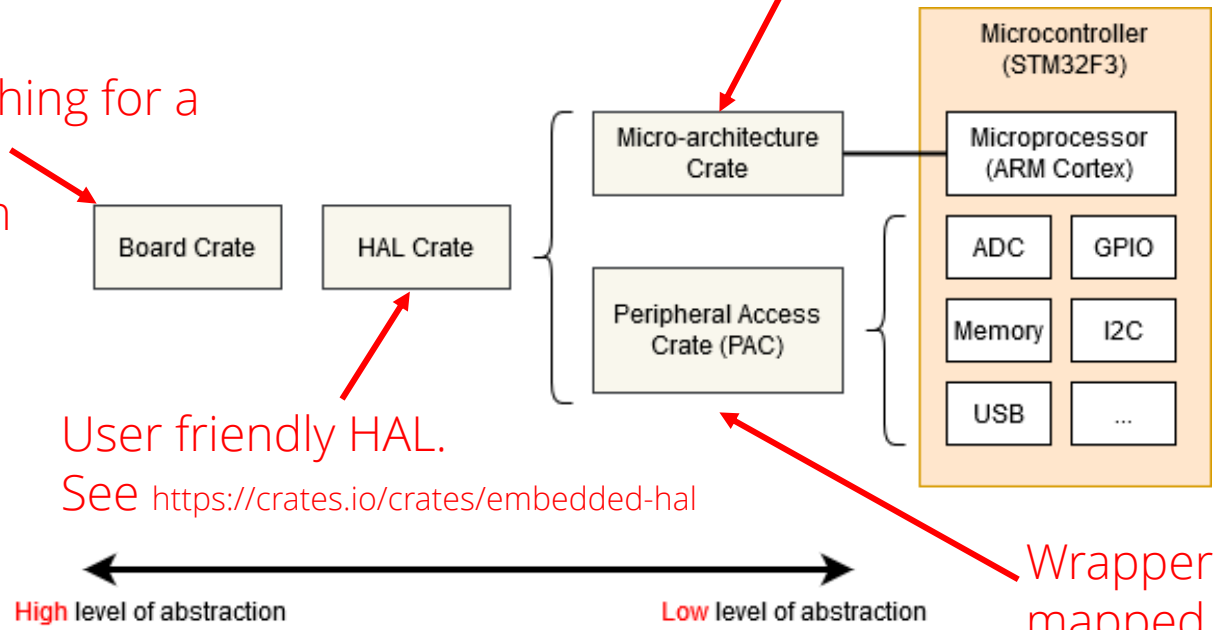
- Nothing, it will do it by default
- update .cargo/config.toml to include a runner
- update memory.x to include RAM sections
- None of the above

4 Memory Mapped Registers

Crates

Manages CPU/Core and common peripherals (systick)

Packages everything for a board including preconfiguration



User friendly HAL.
See https://crates.io/crates/embedded-hal

Wrapper around memory mapped peripheral registers

Example System Tick

```rust
#![no_std]
#![no_main]
use cortex_m::peripheral::{syst, Peripherals};
use cortex_m_rt::entry;
use panic_halt as _;

#[entry]
fn main() -> ! {
    let peripherals = Peripherals::take().unwrap();
    let mut systick = peripherals.SYST;
    systick.set_clock_source(syst::SystClkSource::Core);
    systick.set_reload(1_000);
    systick.clear_current();
    systick.enable_counter();
    while !systick.has_wrapped() {
        // Loop
    }

    loop {}
}
```

5 Going Further

Rust Resources

- Rust Website
- Rust Book
- Rust for Embedded Book
- Learning Rust for Embedded Systems
- Rust By Example
- RTIC: Real-Time Interrupt Driven Concurrency

Thank you for attending

Please consider the resources below:

- www.beningo.com
  - Blog, White Papers, Courses
  - Embedded Bytes Newsletter
    - http://bit.ly/1BAHYXm
  - Embedded Software Design
    - https://www.beningo.com/embedded-software-design/

From www.beningo.com under

- Blog > CEC – Embedded Software using Rust

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

Digi-Key ELECTRONICS

BENINGO EMBEDDED GROUP

informa markets