Microcontrollers and Rust

Daniel Maslowski

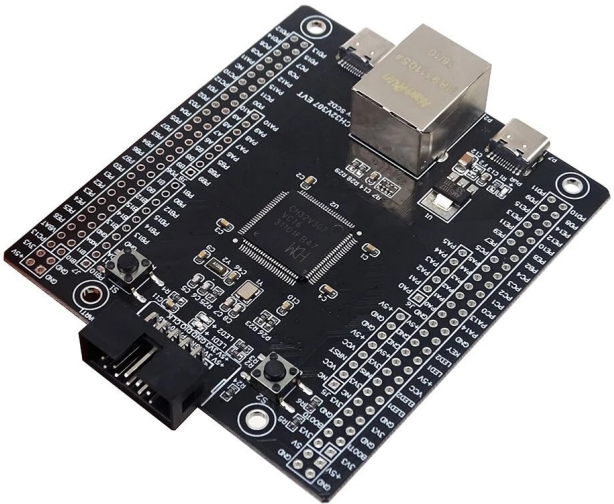
BACKSPACE

Agenda

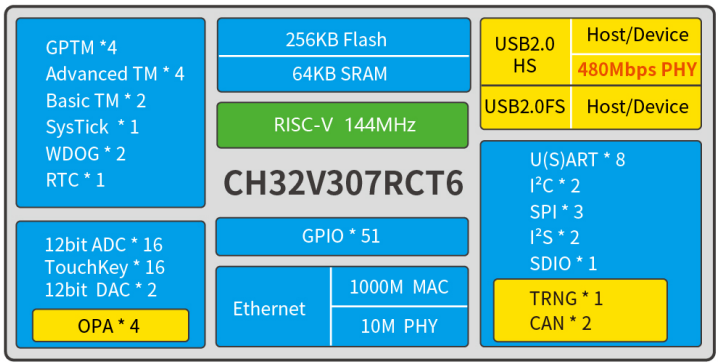
- CH32V307 Overview
- Rust Embedded
- Programming

CH32V307 Overview

Evaluation Board

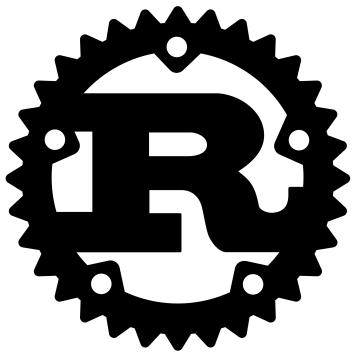


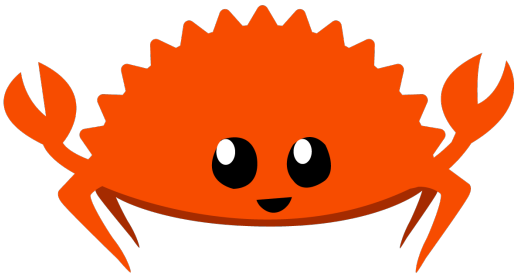
Block Diagram



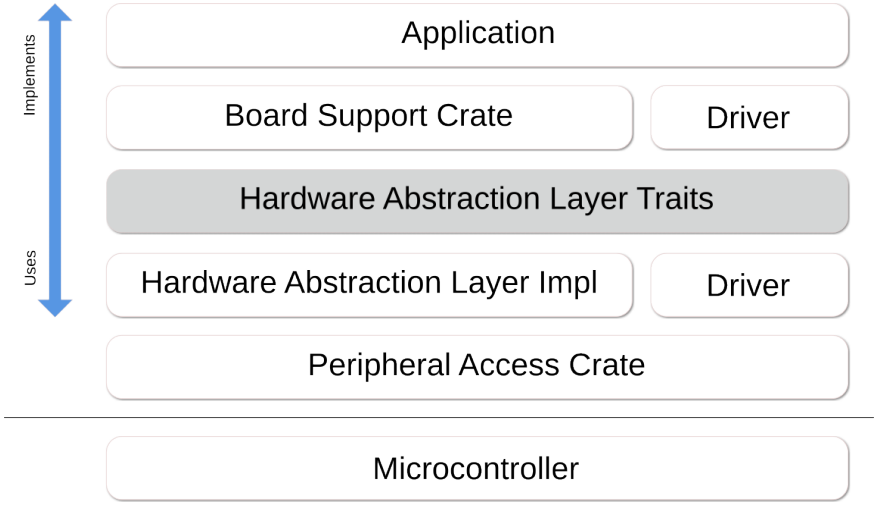
Rust Embedded

Rust

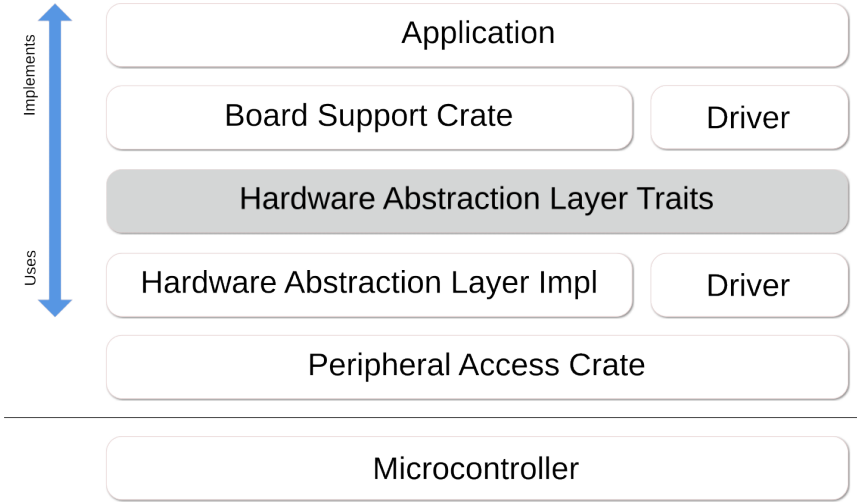




PAC and HAL



PAC and HAL



Note: `embedded-hal` is in active development; a 1.0.0 is *almost there*!

For CH32: https://docs.rs/ch32v3/latest/ch32v3

Programming

Linker Setup

Linker Setup

Usually a bit complex, but we have `riscv-rt`:
https://docs.rs/riscv-rt/latest/riscv_rt/index.html

A little boilerplate `build.rs`, plus `memory.x`:

Linker Setup

Usually a bit complex, but we have `riscv-rt`:
https://docs.rs/riscv-rt/latest/riscv_rt/index.html

A little boilerplate `build.rs`, plus `memory.x`:

```
MEMORY
{
    FLASH : ORIGIN = 0x00000000, LENGTH = 256k
    RAM : ORIGIN = 0x20000000, LENGTH = 64k
}

REGION_ALIAS("REGION_TEXT", FLASH);
REGION_ALIAS("REGION_RODATA", FLASH);
REGION_ALIAS("REGION_DATA", RAM);
REGION_ALIAS("REGION_BSS", RAM);
REGION_ALIAS("REGION_HEAP", RAM);
REGION_ALIAS("REGION_STACK", RAM);
```

Blinking an LED

```
let peripherals = ch32v30x::Peripherals::take().unwrap();
// set up clocks
let rcc = peripherals.RCC;
rcc.apb2pcenr.modify(|_, w| w.iopben().set_bit());
// configure GPIO port B pin 9
let gpiob = &peripherals.GPIOB;
unsafe {
    gpiob.cfghr.modify(|_, w| w.cnf9().bits(0b00)
                               .mode9().bits(0b11));
}
loop {
    gpiob.outdr.modify(|_, w| w.odr9().set_bit()); // HIGH
    sleep(cycle);
    gpiob.outdr.modify(|_, w| w.odr9().clear_bit()); // LOW
    sleep(cycle);
}
```

Setting up UART clock and GPIOs

```
/* ... */
// enable IO port A and UART1
rcc.apb2pcenr.modify(|_, w| w.iopaen().set_bit()
                            .usart1en().set_bit());
// configure alternate function mode
let gpioa = &peripherals.GPIOA;
unsafe {
    gpioa.cfghr.modify(|_, w| w.cnf9().bits(0b10)
                               .mode9().bits(0b11)
                               .cnf10().bits(0b10)
                               .mode10().bits(0b00));
}
/* ... */
```

Setting up UART baud rate etc

```
// enable UART and set length to 8bit
uart.ctlr1.modify(|_, w| w.ue().set_bit().m().clear_bit());
// enable transmitter
uart.ctlr1.modify(|_, w| w.te().set_bit());
// enable receiver and its interrupts (RX non-empty)
uart.ctlr1.modify(|_, w| w.re().set_bit().rxneie().set_bit());
// 12 bits mantissa, last 4 bits are fraction (1/16)
unsafe {
    uart.ctlr2.modify(|_, w| w.stop().bits(0b00));
    uart.brr.modify(|_, w| w.div_mantissa().bits(39)
                            .div_fraction().bits(1));
}
```

Setting up UART baud rate etc

```
// enable UART and set length to 8bit
uart.ctlr1.modify(|_, w| w.ue().set_bit().m().clear_bit());
// enable transmitter
uart.ctlr1.modify(|_, w| w.te().set_bit());
// enable receiver and its interrupts (RX non-empty)
uart.ctlr1.modify(|_, w| w.re().set_bit().rxneie().set_bit());
// 12 bits mantissa, last 4 bits are fraction (1/16)
unsafe {
    uart.ctlr2.modify(|_, w| w.stop().bits(0b00));
    uart.brr.modify(|_, w| w.div_mantissa().bits(39)
                            .div_fraction().bits(1));
}
```

Now, write to the UART's data register! :)

(yes, it's a bit more involved, but not much!)

https://github.com/orangecms/ch32v307-rust

A little RISC-V

```
/* https://five-embeddev.com/riscv-isa-manual/latest/machine.html */
    match riscv::register::misa::read() {
        None => { println!("ISA unknown"); },
        Some(v) => { println!("ISA: {:?}", v); },
    }
    match riscv::register::mvendorid::read() {
        None => { println!("vendor unknown"); },
        Some(v) => { println!("vendor: {:?}", v); },
    }
    match riscv::register::mimpid::read() {
        None => { println!("impl. ID unknown"); },
        Some(v) => { println!("impl. ID: {:?}", v); },
    }
```