# Asynchronous Rust in Embedded Systems

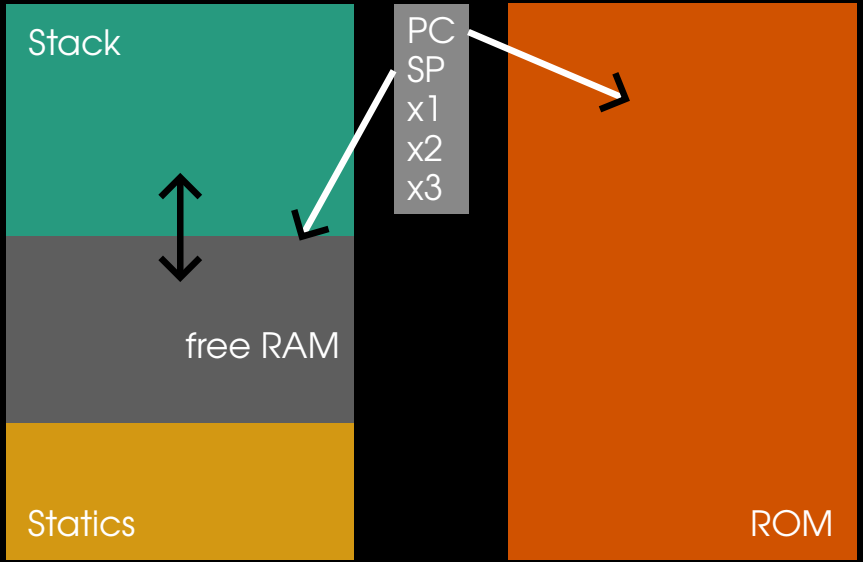
Christian Amsüss <christian@amsuess.com>
@chrysn

2024-02-29
Vienna, Rust meetup

# Embedded systems

- single chip
- $\approx$ 100 KiB ROM, $<$ 100 KiB RAM
- Serial, I2C, analog pins, simple networking ($<$ 1Kbps)

# Execution model: bare metal



# Execution model: bare metal

No parallelism

```
loop:
    if button pressed:
        increase counter
    if socket.is_pending():
      connection = socket.accept()
      connection.recv(...)
      connection.send(...)
      connection.close()
```

# Execution model: bare metal

## Interrupts

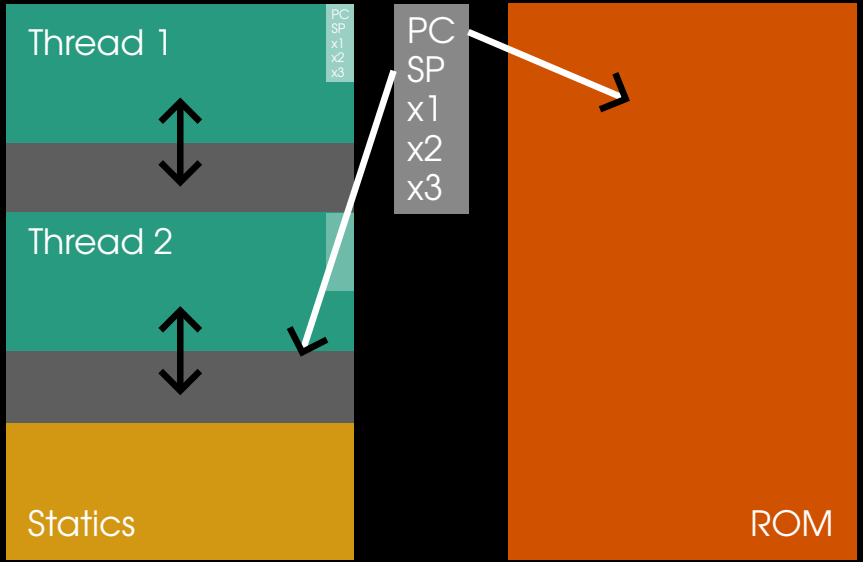
```
on button_pressed():
    increase counter

on network activity():
    ... (see next slide)
```

# Execution model: bare metal

## Hand rolled parallelism

```
connection = None
loop:
    if button pressed:
        increase counter
    if let event = socket.take_pending():
        if connection:
            match event:
                new connection: reject
                new data: receive and process
                ready to send: send(); close(); connection = None
        else:
            match event:
                new connection: connection = Some(accept())
                _: reject
```

# Execution model: threads



# Execution model: threads

## Idiomatic parallelism

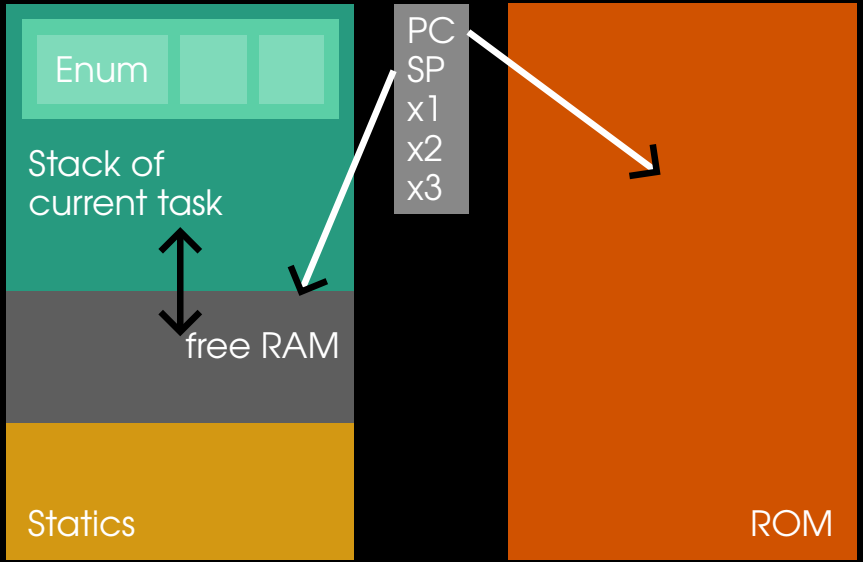
```
button_main():
    if button pressed:
        increase counter

network_main():
    loop:
      connection = socket.accept()
      connection.recv(...)
      connection.send(...)
      connection.close()
```

# Communication between threads

. . . and interrupts

```
T: Send
```

# Execution model: async



# Execution model: async

Idiomatic parallelism with explicit yielding

```
async button_main():
    if button pressed.await:
        increase counter

async network_main():
    loop:
      connection = socket.accept().await
      connection.recv(...).await
      connection.send(...).await
      connection.close()
```

# Async: Tools and further reading

Get started with it!

- `core`
- `embedded-hal-async`
- `embedded-nal-async`
- `embassy`
- RIOT-OS

- Interrupts Is Threads (James Munns)
- Embedded Concurrency Patterns (Ferrous Systems)

Thanks for having me here

Slides and more links on `https://github.com/RustVienna/meetup-history/tree/master/2024-02/embedded-async/`

Christian Amsüss <christian@amsuess.com>, @chrysn