



NORDIC®
SEMICONDUCTOR

NRF-HAL

Native implementation of rtic
monotonic timers for the nrf-52840

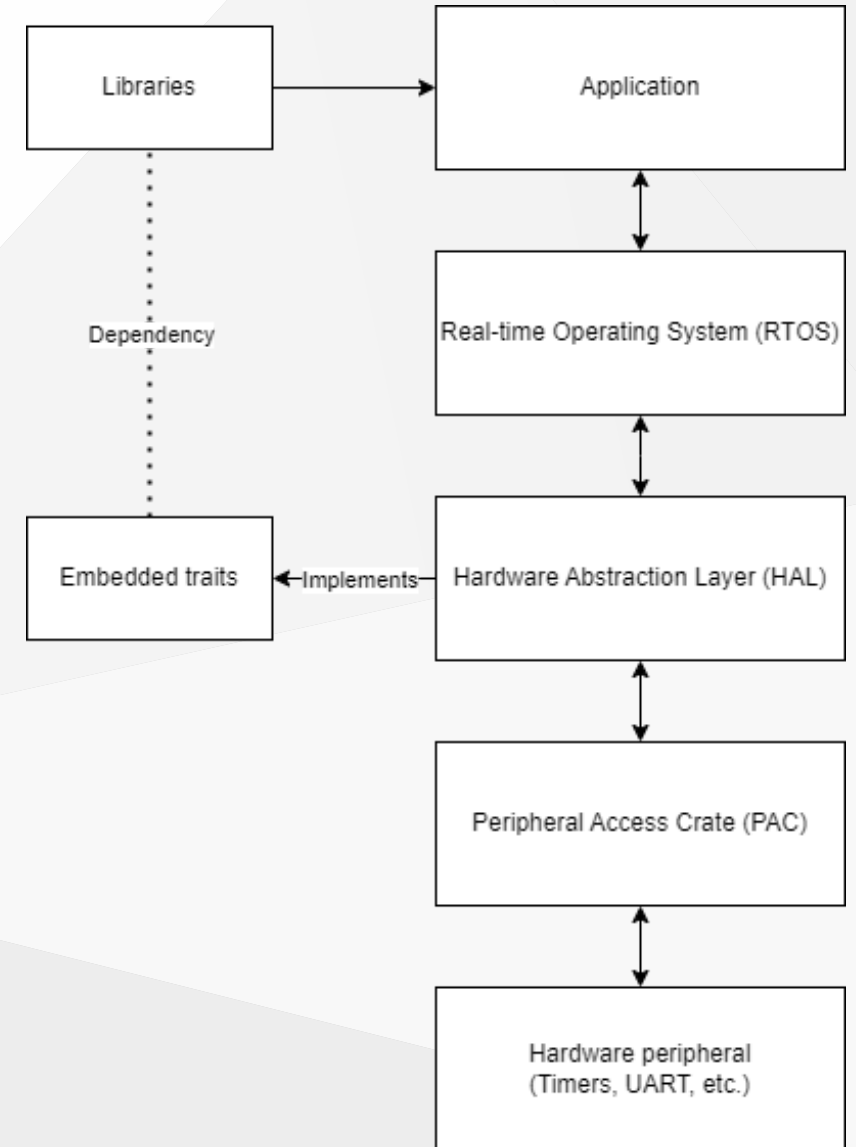
What is rtic

- Real-Time Interrupt driven Concurrency
- LTU
- ARM cortex-m

What is a HAL

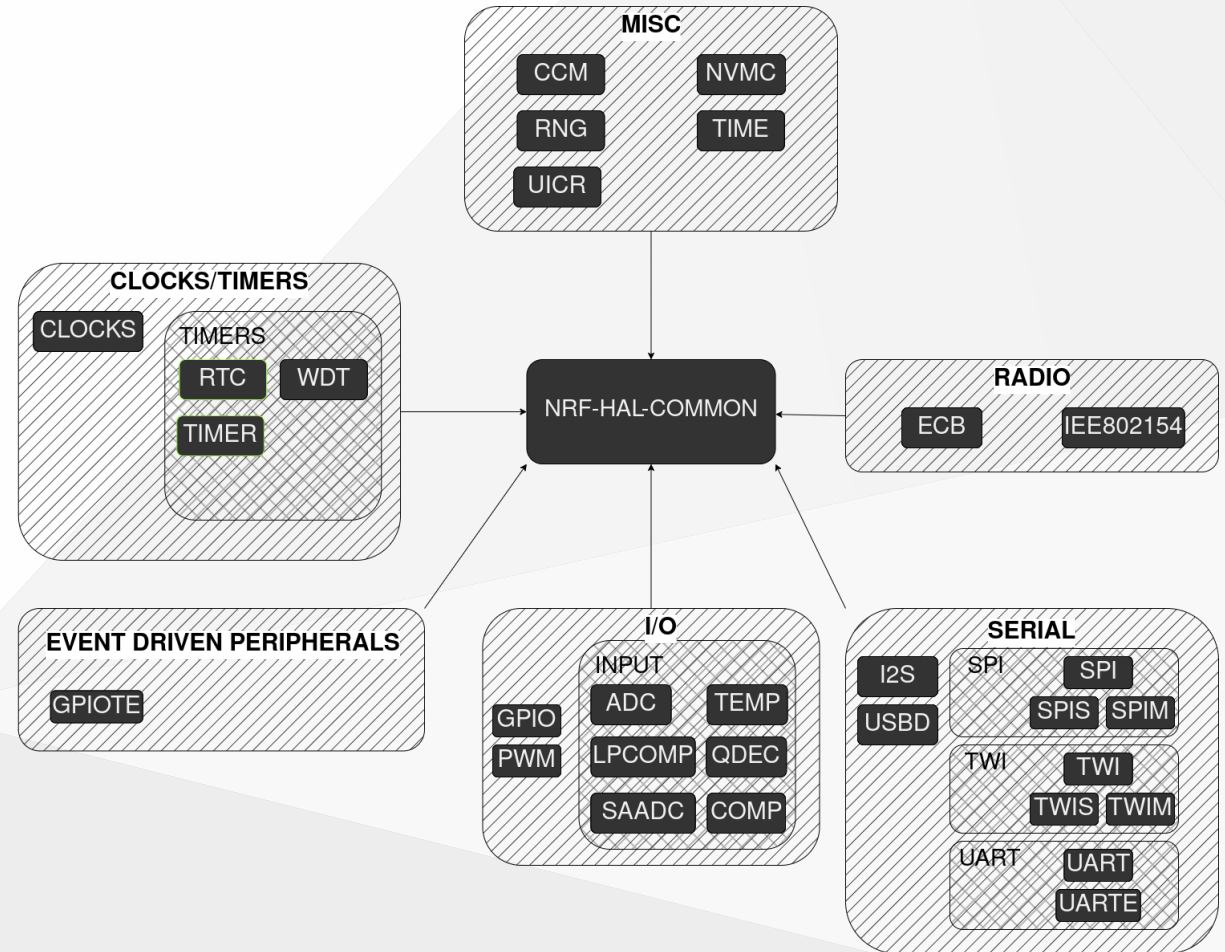
- Hardware Abstraction Layer
- Implements strategies

```
/// A serial interface  
pub trait Serial {  
    /// Error type associated to this serial interface  
    type Error;  
  
    /// Reads a single byte  
    fn read(&mut self) -> nb::Result<u8, Self::Error>;  
  
    /// Writes a single byte  
    fn write(&mut self, byte: u8) -> nb::Result<(), Self::Error>;  
}
```



NRF-HAL

- Type driven
- Safe and fast



Timers and RTCs

- Similarities
- Differences

	TIMER	RTC
Base f	16MHz/1MHz	~32 KHz
Counter width	32 bit	24 bit
T	19 hours	24 days
ΔT [s]	$63 \cdot 10^{-9}$	$0.03 \cdot 10^{-3}$

Attempt one

- Generic interface for both rtc and timer
- Compile time guarantee of correctness

```
pub struct MonotonicTimer<T: Instance, const FREQ: u32> {  
    instance: PhantomData<T>,  
}
```

Register access by trait system

```
/// A trait that ensures register access for the [`pac`](`crate::pac`)  
/// abstractions  
pub trait Instance {  
    /// The type of the underlying register block  
    type RegBlock;  
    /// Returns a pointer to the underlying register block  
    ///  
    /// Allows modification of the registers at a type level rather than  
    /// by storing the [`Instance`] at run-time.  
    fn reg<'a>() -> &'a Self::RegBlock;  
}  
pub trait RtcInstance: Instance<RegBlock = super::RtcRegBlock> {}  
pub trait TimerInstance: Instance<RegBlock = super::TimerRegBlock0> {}
```

Compiler bugs and headaches

- Mutually exclusive traits

```
impl<T:RtcInstance, const FREQ:u32> Monotonic for MonotonicTimer<T,FREQ>{...}  
impl<T:TimerInstance, const FREQ:u32> Monotonic for MonotonicTimer<T,FREQ>{...}
```

- Aha! Bug/ not yet implemented in rustc

Back to the drawing board

- Two distinct types

```
pub struct MonotonicTimer<T: TimerInstance, const FREQ: u32> {  
    ...  
}  
pub struct MonotonicRtc<T: RtcInstance, const FREQ: u32> {  
    ...  
}
```

- Yet more issues...

Frequency gating construction

- Timer
9 different prescalers
- Rtc
 $2^{12} - 1$ prescalers

Final interface

Timer

```
#[monotonic(binds = TIMER3, default = true)]  
type MyMono = MonotonicTimer<TIMER3,16_000_000>;  
/// new only exists on valid frequencies  
let mono = MyMono::new(ctx.device.TIMER3);
```

Final interface

Rtc

```
#[monotonic(binds = RTC0, default = true)]  
type MyMono = MonotonicRtc<RTC0, 32_768>;  
let clocks = hal::clocks::Clocks::new(cx.device.CLOCK);  
let clocks = clocks.start_lfclk();  
/// Will throw error if freq is invalid  
let mono = MyMono::new(cx.device.RTC0, &clocks).unwrap();
```

References and futher reading

- rtic <https://rtic.rs/2/book/en/>
- nrf-hal https://docs.rs/nrf52840-hal/latest/nrf52840_hal/
- embedded-hal https://docs.rs/embedded-hal/latest/embedded_hal/