The Time is Right
Retrofitting Formal Verification on Timers

Samir Rashid

TockWorld 8
September 5, 2025

time is complicated…

capsules/alarm: left-justify 32 bit ticks, re-architect alarm, add unit tests P-Significant

tock/tock#3975 · by lschuermann was closed on Jun 4, 2024 · Approved · 9/11

Alarm overflow - gettimeasticks last-call

tock/libtock-c#366 · by tyler-potyondy was closed on Feb 7, 2024 · Approved

8

RFC: Expand Alarm Range

#3879 · by Samir-Rashid was closed on Mar 1, 2024

libtock: alarm: include assert.h

tock/libtock-c#441 · by bradjc was closed on Jun 13, 2024 · Approved · 4/4

11

alarm: create `ms_to_ticks` helper function

tock/libtock-c#434 · by Samir-Rashid was closed on Jun 7, 2024 · Approved · 2/2

Port alarm to new libtock-c design

tock/libtock-c#413 · by hudson-ayers was closed on May 6, 2024 · Approved · 2/4

5

Alarm Syscall: fix computed `dt` when both unshifted `reference` and `dt` from userspace have low-order bits. last-call

tock/tock#4201 · by alevy was closed on Oct 17, 2024 · Approved · 9/11

10

alarm: fix gettimeasticks arithmetic imprecision

tock/libtock-c#411 · by Samir-Rashid was closed on May 9, 2024 · Review required · 2/2

9

Update alarm to new libtock-c API

tock/libtock-c#408 · by hudson-ayers was closed on May 5, 2024 · 0/4

54

timer: handle timer overflow

tock/libtock-c#395 · by Samir-Rashid was closed on Jun 21, 2024 · Approved · 2/2

chips: apollo3: Handle the errata for clock double increments last-call

tock/tock#4063 · by alistair23 was closed on Jul 8, 2024 · Approved · 9/11

7

alarm: remove references to timezone

tock/libtock-c#473 · by ppannuto was closed on Nov 4, 2024 · Approved · 4/4

Fix alarm syscall driver command comments

tock/tock#3970 · by hudson-ayers was closed on Apr 18, 2024 · Approved

libnrfserialization: use correct alarm cancel api bug

tock/libtock-c#471 · by ppannuto was closed on Oct 30, 2024 · Approved · 4/4

Alarm: Change prescaler for NRF 5x boards

#3938 · by Samir-Rashid was closed on Apr 19, 2024

2

alarm: rewrite alarm virtualization with better comments and simpler logic, add tests

tock/libtock-c#468 · by alevy was closed on Oct 22, 2024 · Approved · 4/4

14

Fix multi-alarm overlapping bug

tock/libtock-c#466 · by alevy was closed on Sep 23, 2024 · Approved · 4/4

27

libtock: alarm: ticks_to_ms can rollover

closed on Aug 27, 2024 · Approved · 4/4

16

kernel/hil/time,capsules/alarm: pad Ticks to 32 bit for predictable wrapping HIL

tock/tock#3973 · by lschuermann was closed on May 9, 2024 · Approved · 15/17

chips: apollo3: stimer: Ensure alarm occurs if set in the past last-call

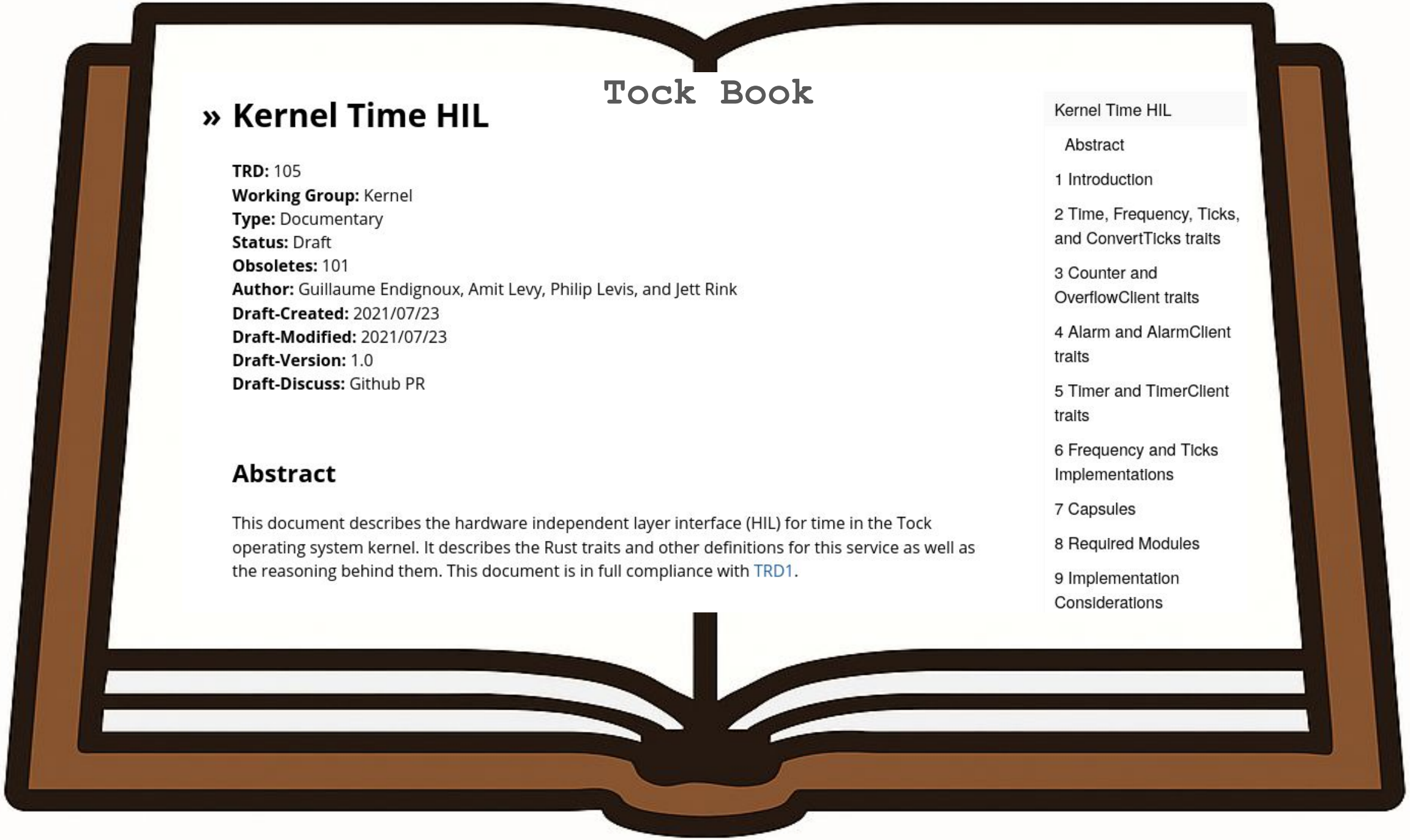
tock/tock#4078 · by alistair23 was closed on Jul 12, 2024 · Approved · 9/11

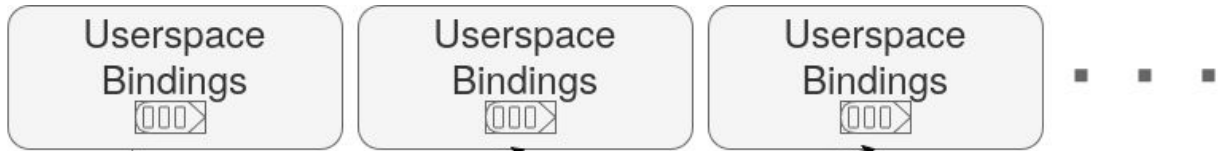
Overview

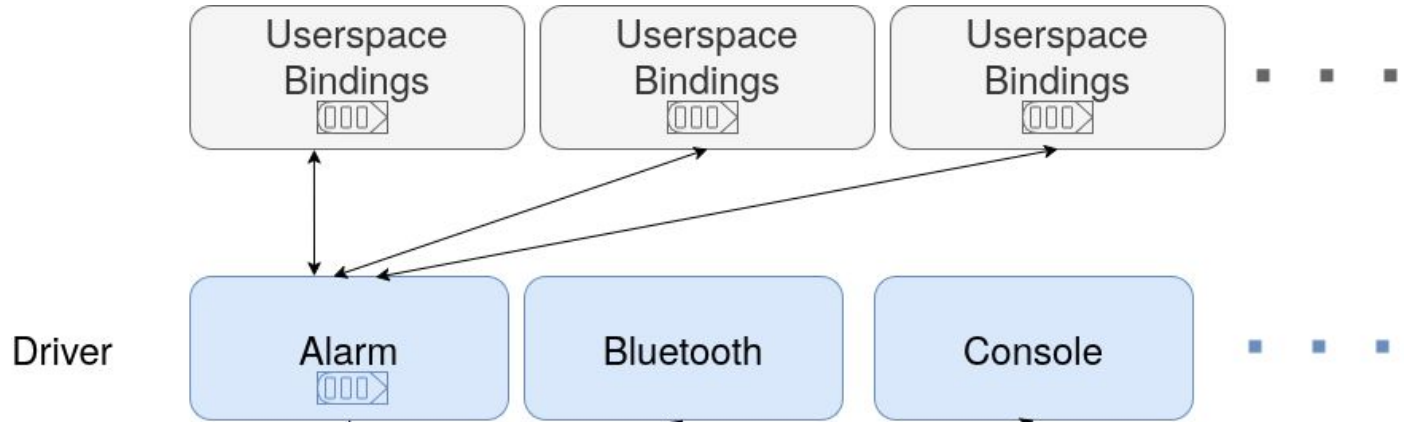
1. Formal Timer Model

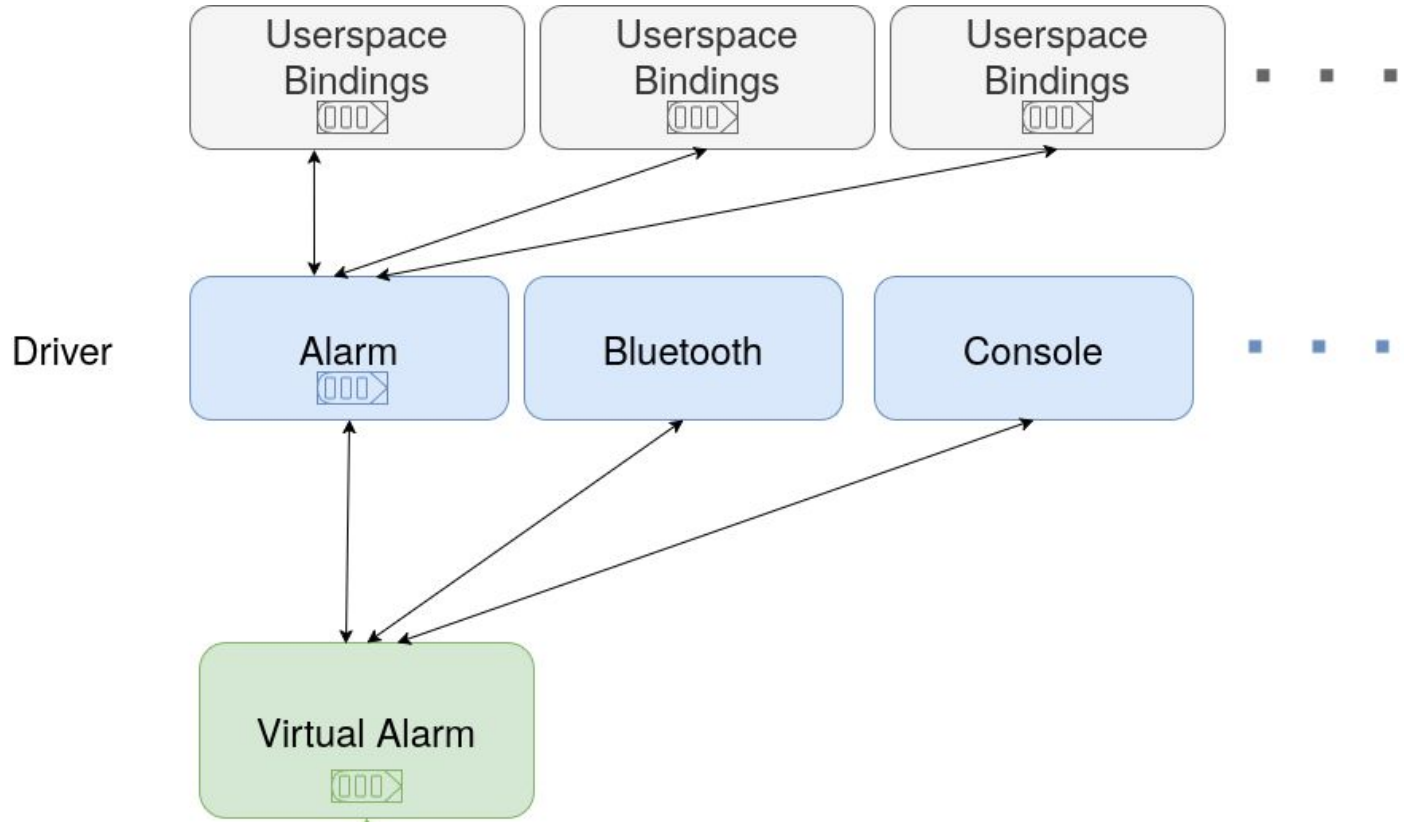
2. Verification Retrofitting Technique

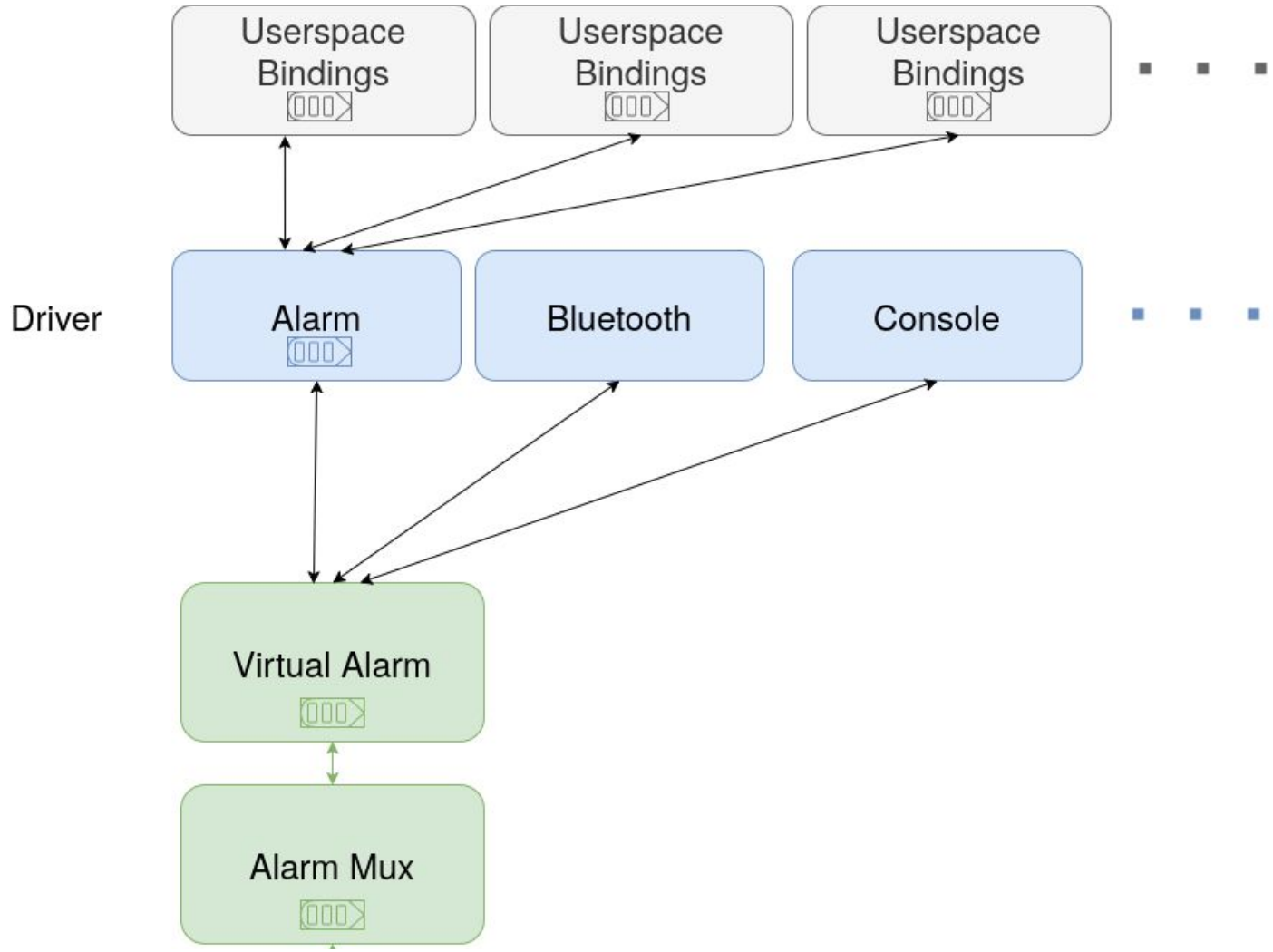
What is Time?

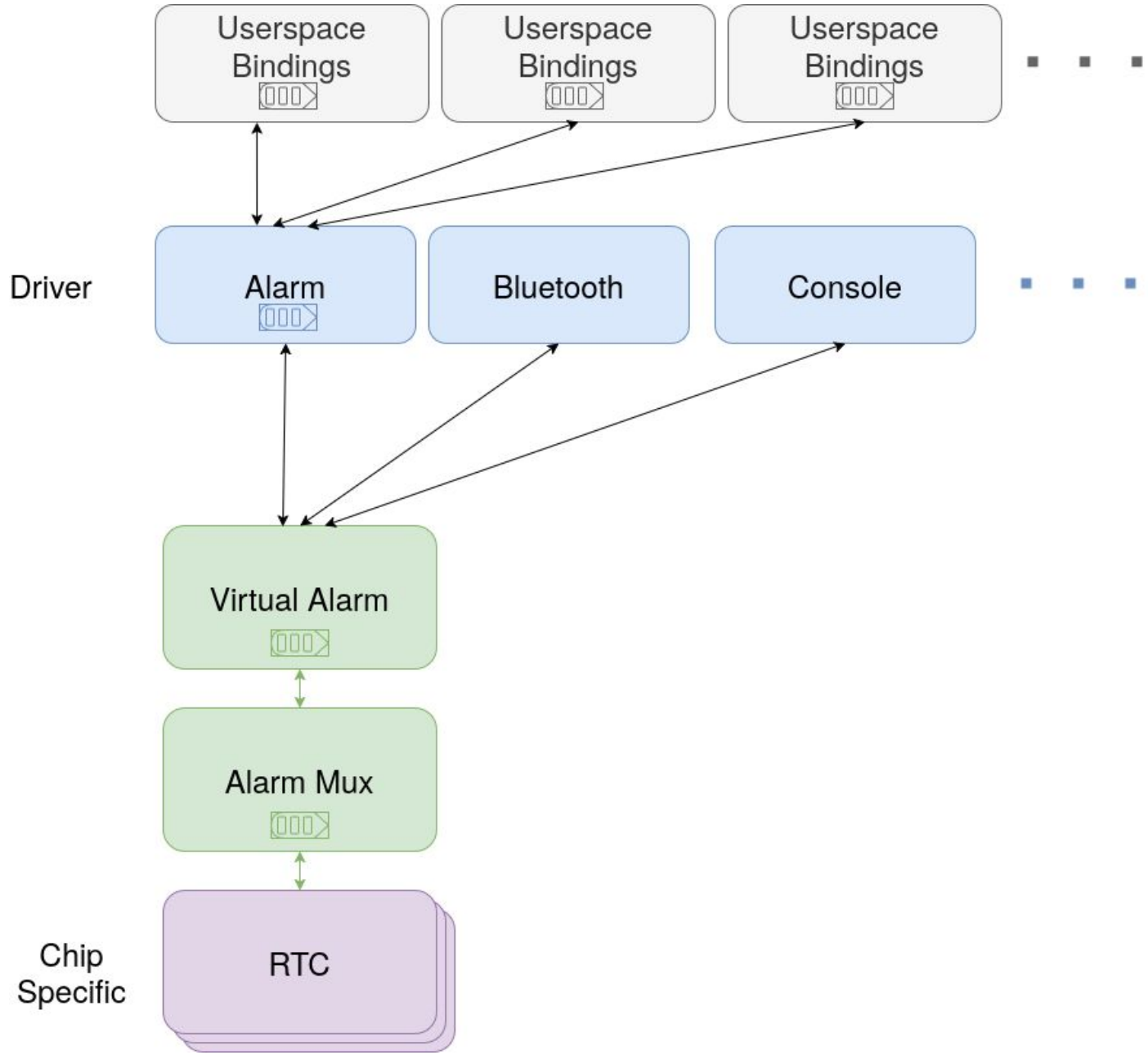

Tock Book
» Kernel Time HIL
TRD: 105
Working Group: Kernel
Type: Documentary
Status: Draft
Obsoletes: 101
Author: Guillaume Endignoux, Amit Levy, Philip Levis, and Jett Rink
Draft-Created: 2021/07/23
Draft-Modified: 2021/07/23
Draft-Version: 1.0
Draft-Discuss: Github PR
Abstract
This document describes the hardware independent layer interface (HIL) for time in the Tock operating system kernel. It describes the Rust traits and other definitions for this service as well as the reasoning behind them. This document is in full compliance with TRD1.
Kernel Time HIL
Abstract
1 Introduction
2 Time, Frequency, Ticks, and ConvertTicks traits
3 Counter and OverflowClient traits
4 Alarm and AlarmClient traits
5 Timer and TimerClient traits
6 Frequency and Ticks Implementations
7 Capsules
8 Required Modules
9 Implementation Considerations


Userspace
Bindings
Userspace
Bindings
Userspace
Bindings

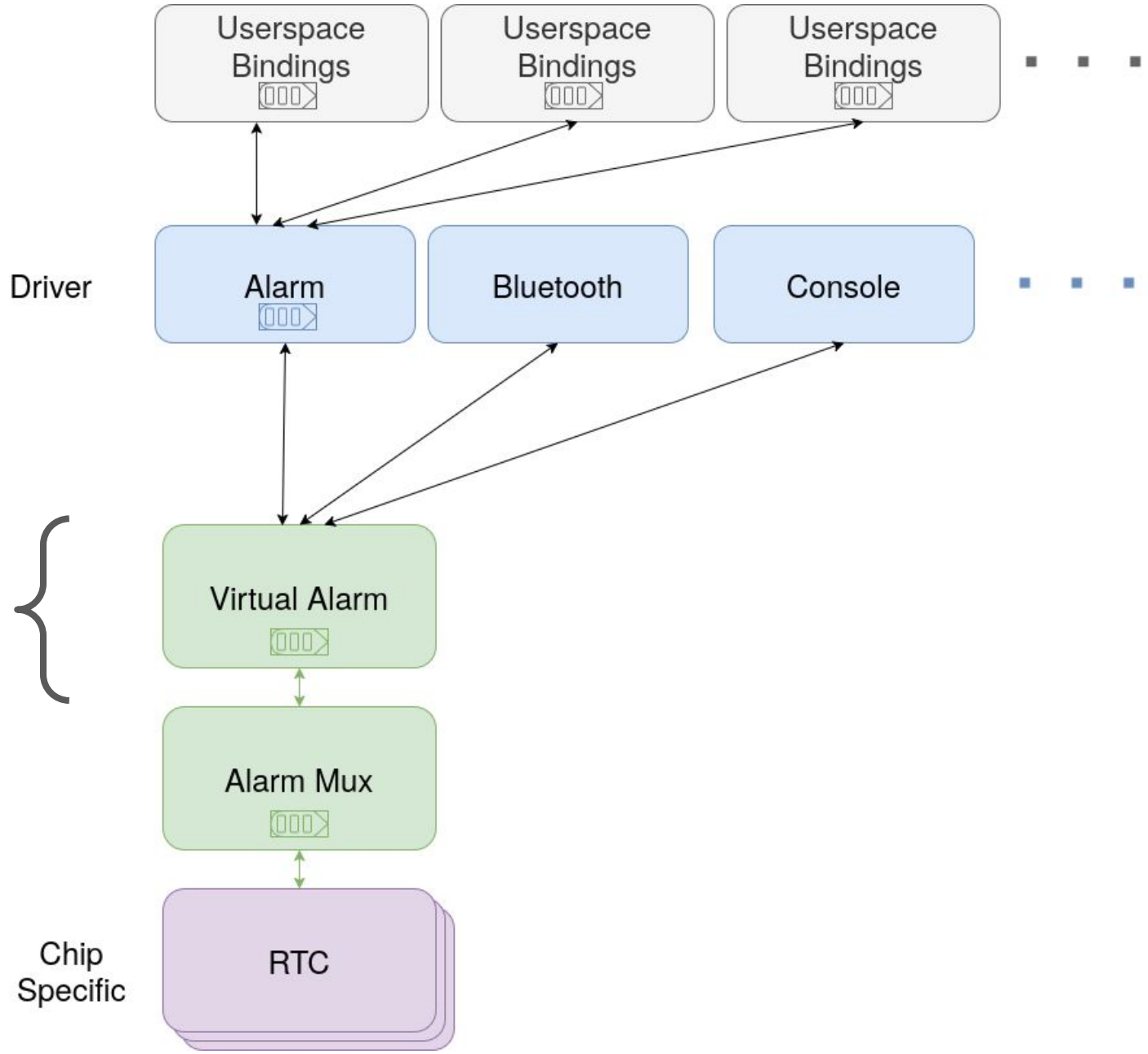

Userspace Bindings
Userspace Bindings
Userspace Bindings
Driver
Alarm
Bluetooth
Console


Userspace Bindings
Userspace Bindings
Userspace Bindings
Driver
Alarm
Bluetooth
Console
Virtual Alarm

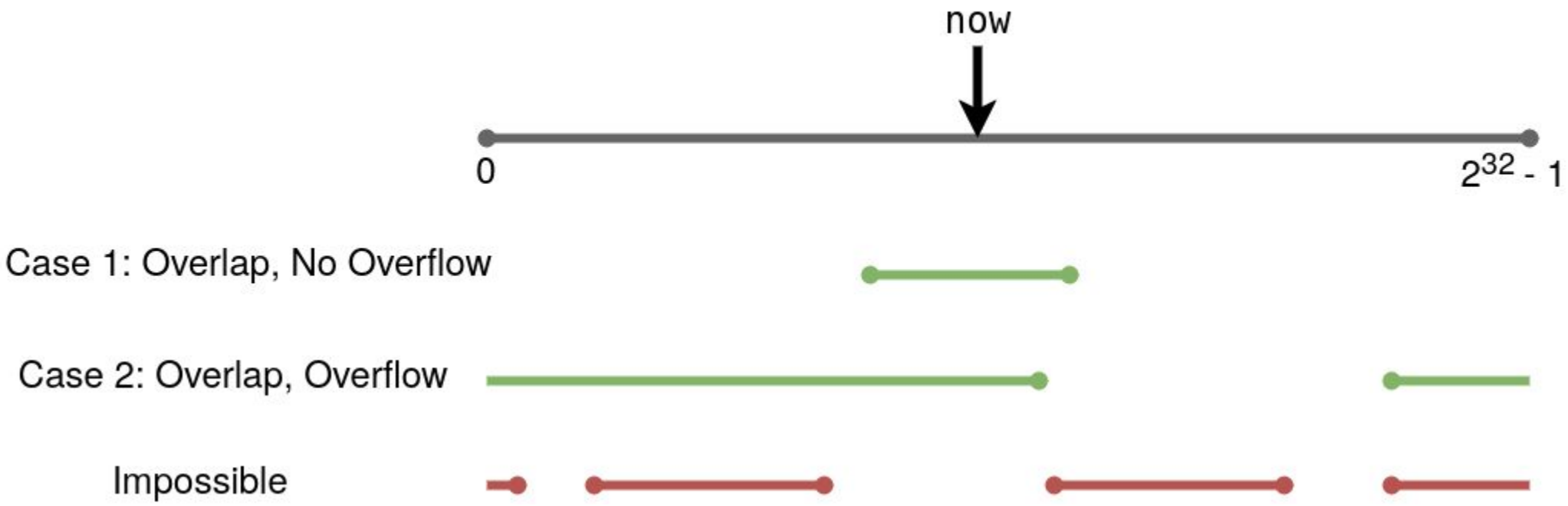

Userspace Bindings
Userspace Bindings
Userspace Bindings
Driver
Alarm
Bluetooth
Console
Virtual Alarm
Alarm Mux


Userspace Bindings
Userspace Bindings
Userspace Bindings
Driver
Alarm
Bluetooth
Console
Virtual Alarm
Alarm Mux
Chip Specific
RTC

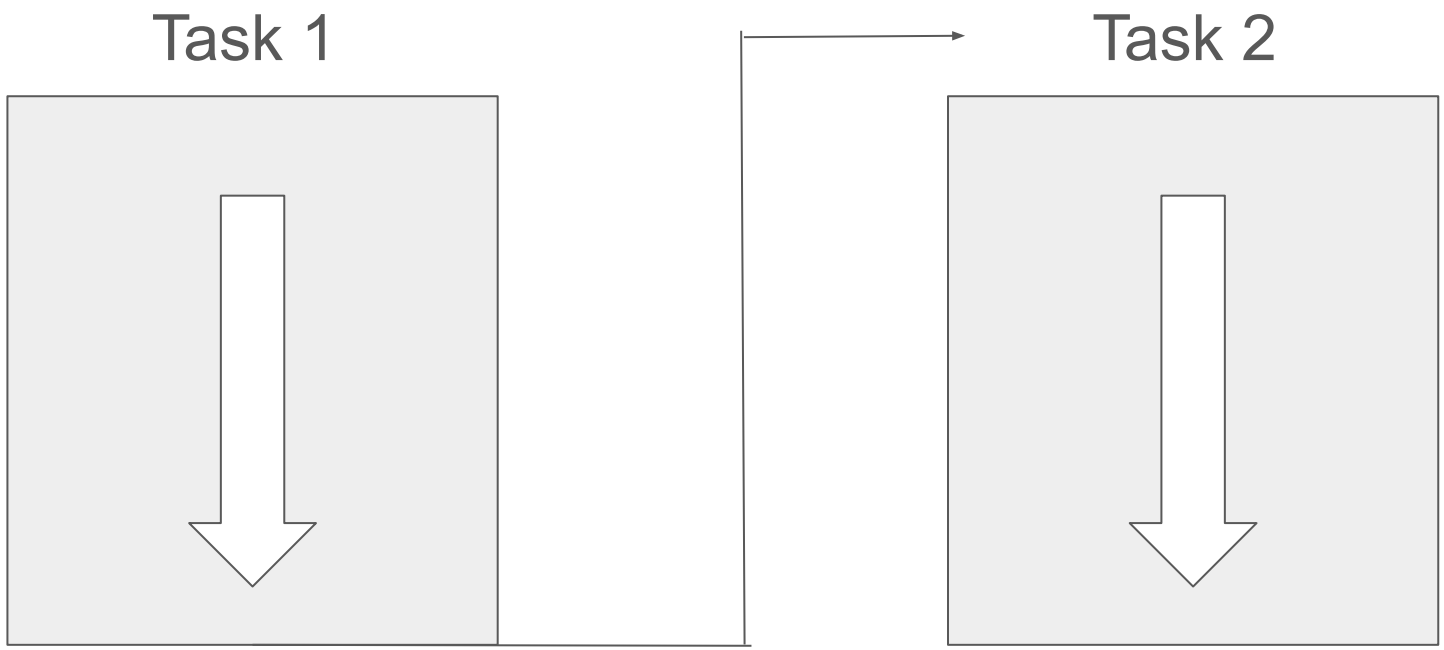
This Work


Userspace Bindings
Userspace Bindings
Userspace Bindings
Driver
Alarm
Bluetooth
Console
Virtual Alarm
Alarm Mux
Chip Specific
RTC

Key Idea: Time only depends on now



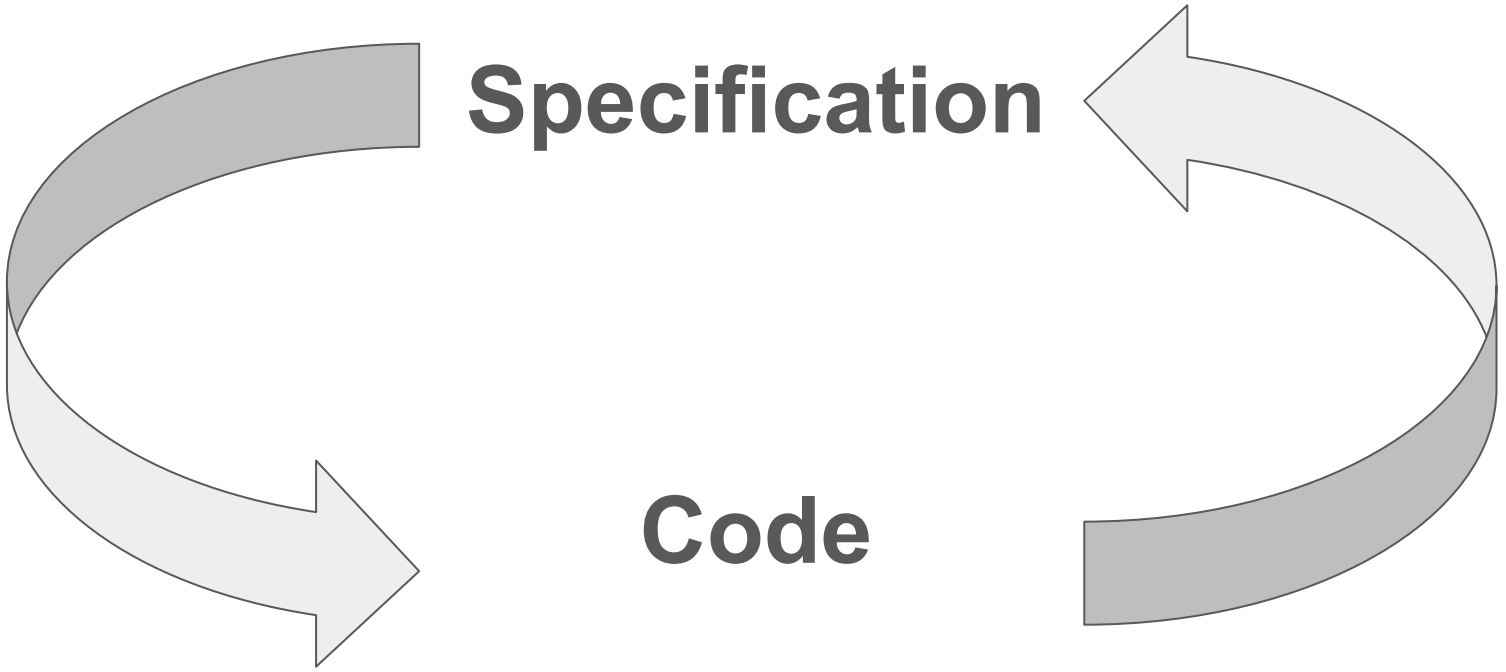
Key Idea: Run-to-Completion



Insight: run-to-completion prevents concurrency issues

Designing Invariants

Retrofitting Specification



Invariants

Progress Soonest alarm fires next

1. No Alarms
2. Elapsed Alarms Fire

Preservation All past alarms have fired

3. Interrupt Upper Bound
4. Interrupt Lower Bound

1. No Alarms

if **no enabled alarms**,

then **hardware is disarmed**

$\forall$ alarms: **alarm.enabled $\Longrightarrow$ hardware.disarmed**

2. Elapsed Alarms Fire

if **alarm elapses now**,

then **alarm fires**

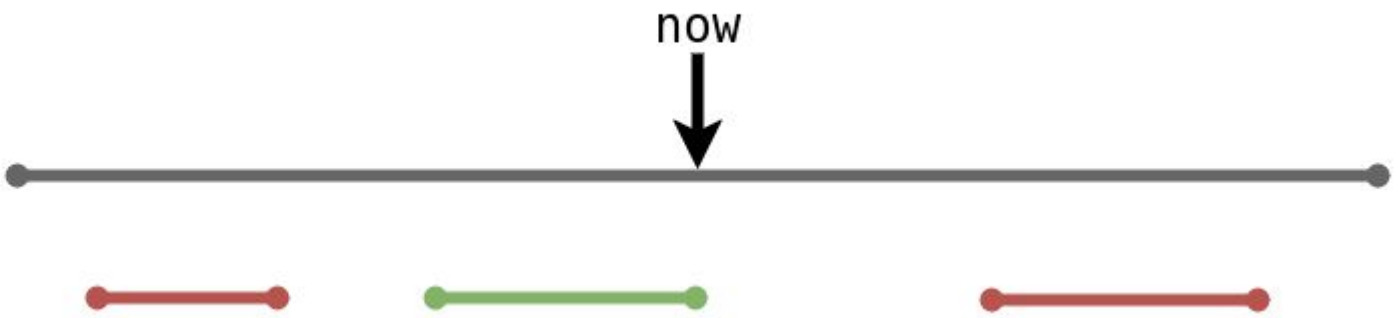
$\forall$ alarms: (**alarm.fire_time == now**) $\Longrightarrow$

alarm.disabled

2. Elapsed Alarms Fire

if **alarm elapses now**, then **alarm fires**

$\forall$ alarms: (**alarm.fire_time == now**) $\Longrightarrow$ **alarm.disabled**



3. Interrupt Upper Bound

If there is an **enabled alarm,**

then the **fire time** is sooner than the fire time of **all alarms**

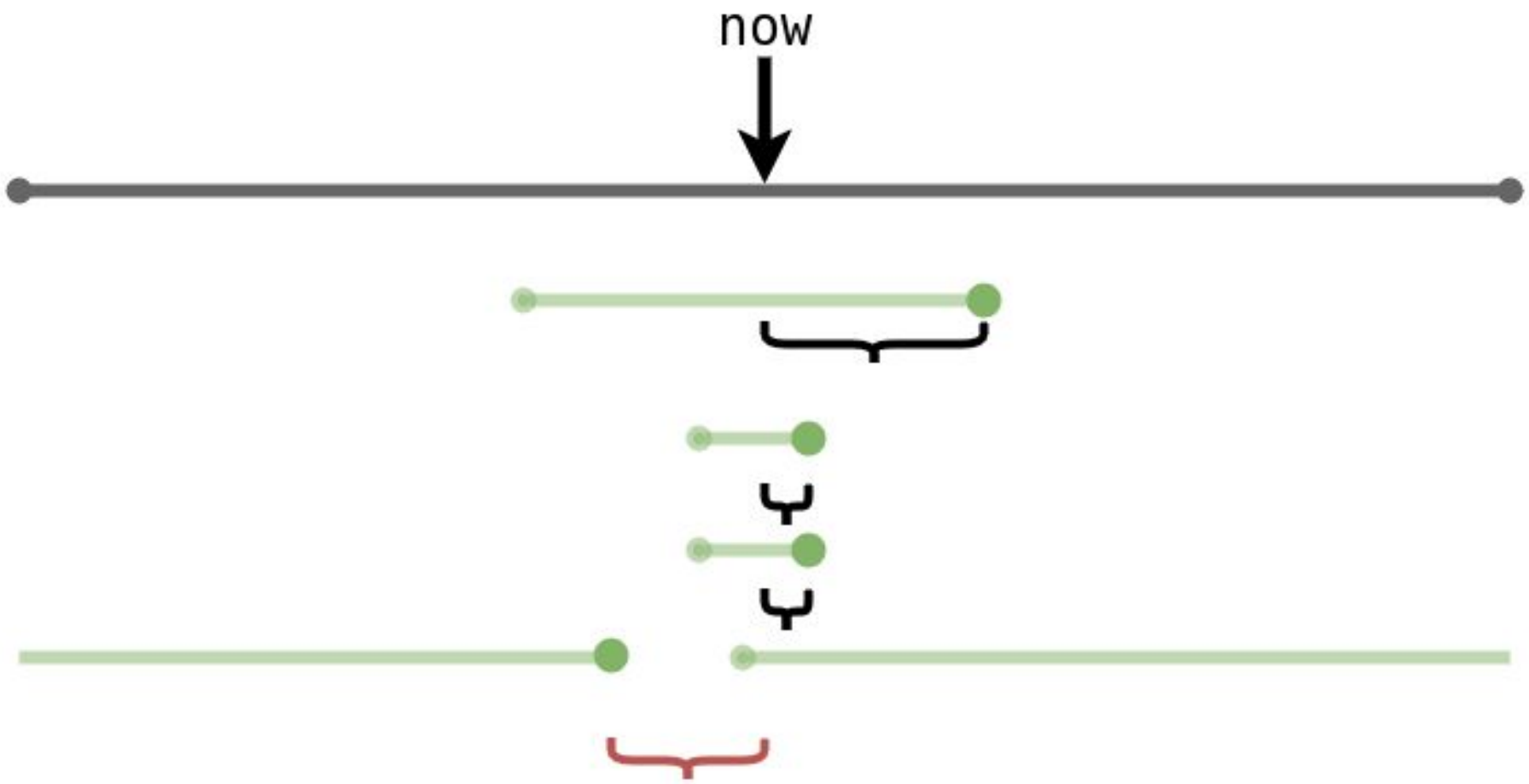
$$\exists \text{ alarm: } \textbf{alarm.enabled} \Longrightarrow$$

$$\forall \text{ alarms: } \textbf{new\_fire\_time - prev\_fire\_time}$$

$$\leqslant \qquad (\text{mod } 2^{32})$$

$$\textbf{alarm.fire\_time - prev\_fire\_time}$$

3. Interrupt Upper Bound



4. Interrupt Lower Bound

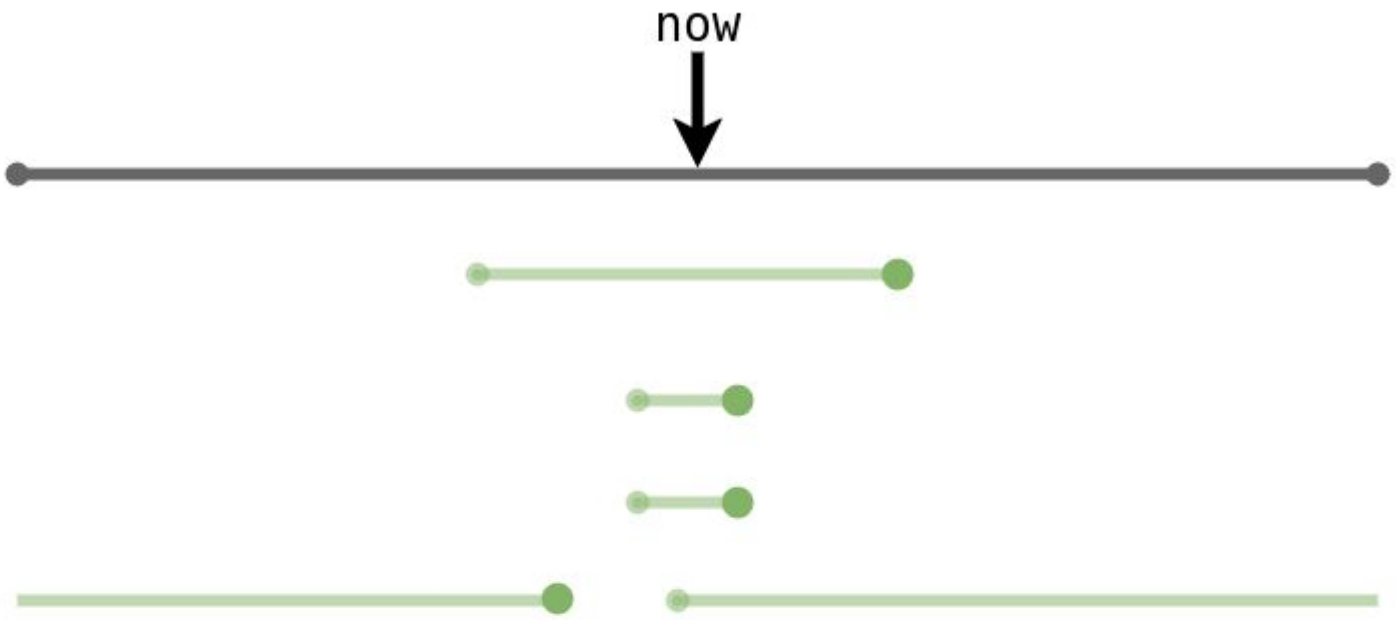
If there is an **enabled alarm,**

then **fire time** equals the **fire time** of some alarm

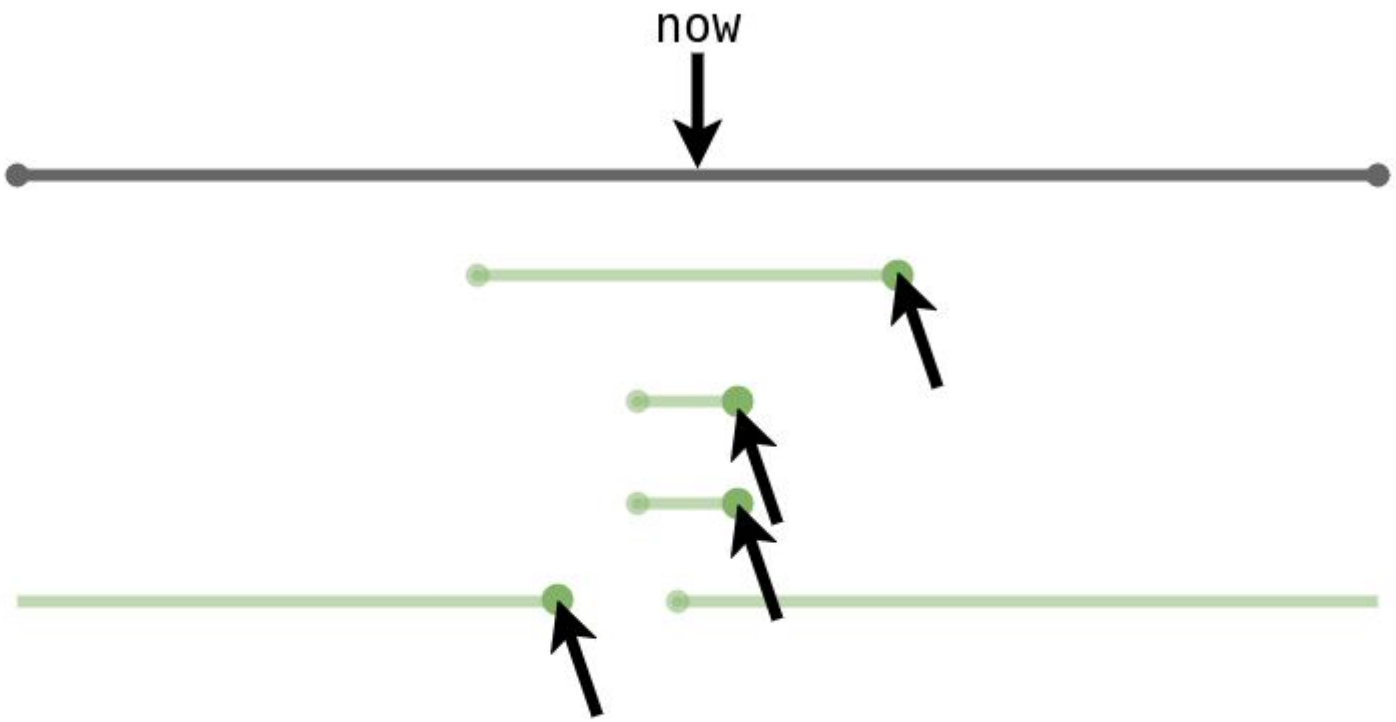
$$\exists \text{ alarm: } \textbf{alarm.enabled} \Longrightarrow$$

$$\exists \text{ alarm: } \textbf{alarm.fire\_time} == \textbf{hardware.fire\_time}$$

4. Interrupt Lower Bound



4. Interrupt Lower Bound



Cannot fire sooner than the alarm should fire

Verification Process

What are Formal Methods?

Formal verification is the process of using automatic proof procedures to establish that a ***computer program will do what it's supposed to****.*

What is Verus

- verification tool integrated into Rust
- adds special proof-checked syntax

```
error: assertion failed
    --> capsules/core/src/virtualizers/new_virtual_alarm.rs:1710:20
     |
1710 |                 assert(mux_perms.num_fired_alarms == 1);
     |                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ assertion failed
```

Verification Invariants

```
error: postcondition not satisfied
    --> capsules/core/src/virtualizers/new_virtual_alarm.rs:1667:9
     |
1667 |         mux_perms.num_fired_alarms == mux_perms.num_total_alarms,
     |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ failed this postcondition
...
1672 |         return;
     |         ------ at this exit
```

Add Verification State

```
pub struct FakeAlarm<'a> {
    pub now: PCell<Ticks32>,
    pub reference: PCell<Ticks32>,
    pub dt: PCell<Ticks32>,
    pub armed: PCell<bool>,
    pub client: &'a ClientCounter,
}
```

```
pub tracked struct FakeAlarmPerms {
    pub tracked now_perm: Tracked<PointsTo<Ticks32>>,
    pub tracked reference_perm: Tracked<PointsTo<Ticks32>>,
    pub tracked dt_perm: Tracked<PointsTo<Ticks32>>,
    pub tracked armed_perm: Tracked<PointsTo<bool>>,
    pub ghost fire_time: int,
}
```

Add Verification State

```
fn is_armed(&self, Tracked(perms): Tracked<&FakeAlarmPerms>) -> (result: bool)
    requires
        self.fake_alarm_wf(perms),
        perms.armed_perm@.id() === self.armed.id(),
    ensures
        self.fake_alarm_wf(perms),
        perms.armed_perm@.id() === self.armed.id(),
        result == perms.armed_perm@.value(),
{
    *self.armed.borrow(Tracked(perms.armed_perm.borrow()))
}
```

Requires refactoring *all* code

Interior Mutability

```
let (pcell, mut points_to) = Pcell:new(1);
```

mutable permission still must pass ownership rules

Verifier missing features

- Iterators
- Dynamic traits
- Cross-Crate Verification
- Finding bugs in Verus

Required changing *every line* of code

Key Insights

Evaluation: Virtual Alarm Code

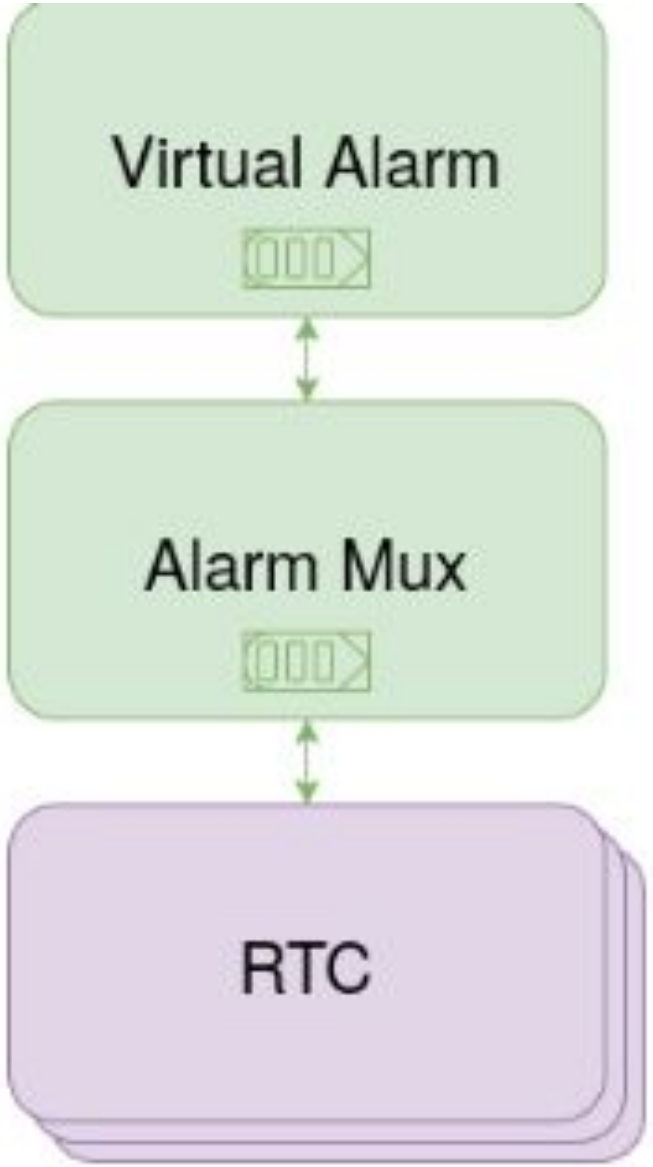
Executable Code	520	32%
Specification	598	36%
Proof Code	525	32%
Total	**1643**	**100%**

8 weeks

2 weeks

10 weeks

Proof Generalizes to other virtualizers



Takeaways

1. **Specification Design**
   Tock timers always proceed, correctly

2. **Verification Method**
   Retrofit widely used OS Code
   Codesigned proof with 2x line of code overhead

Any Questions?