arm

Mbed Silicon Partner Workshop

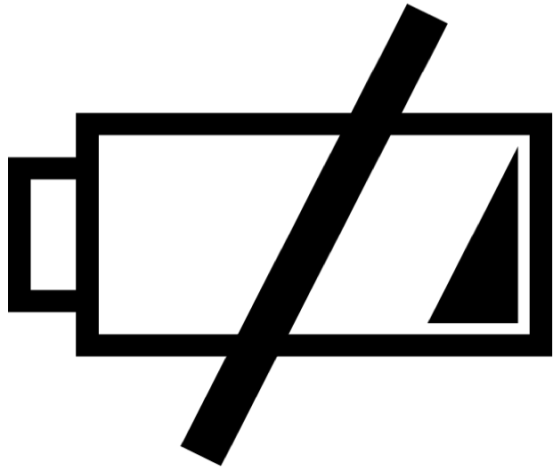
HAL Improvements Sleep

26th of March 2018

Paris, France

Introduction

- Sleep behavior
  - How does an application invoke sleep
  - How smart wait works
  - Idle loop
  - Wake up sources and latency
- Sleep manager
  - `DeepSleepLock` object
  - Lock/unlock deep sleep



When and how your application sleeps

arm

Sleep modes

- Two sleep modes
  - Sleep
  - Deep sleep
- Historically
  - Explicitly selected and entered by the user
- Since Mbed OS 5.6
  - RTOS Idle loop invokes sleep

arm

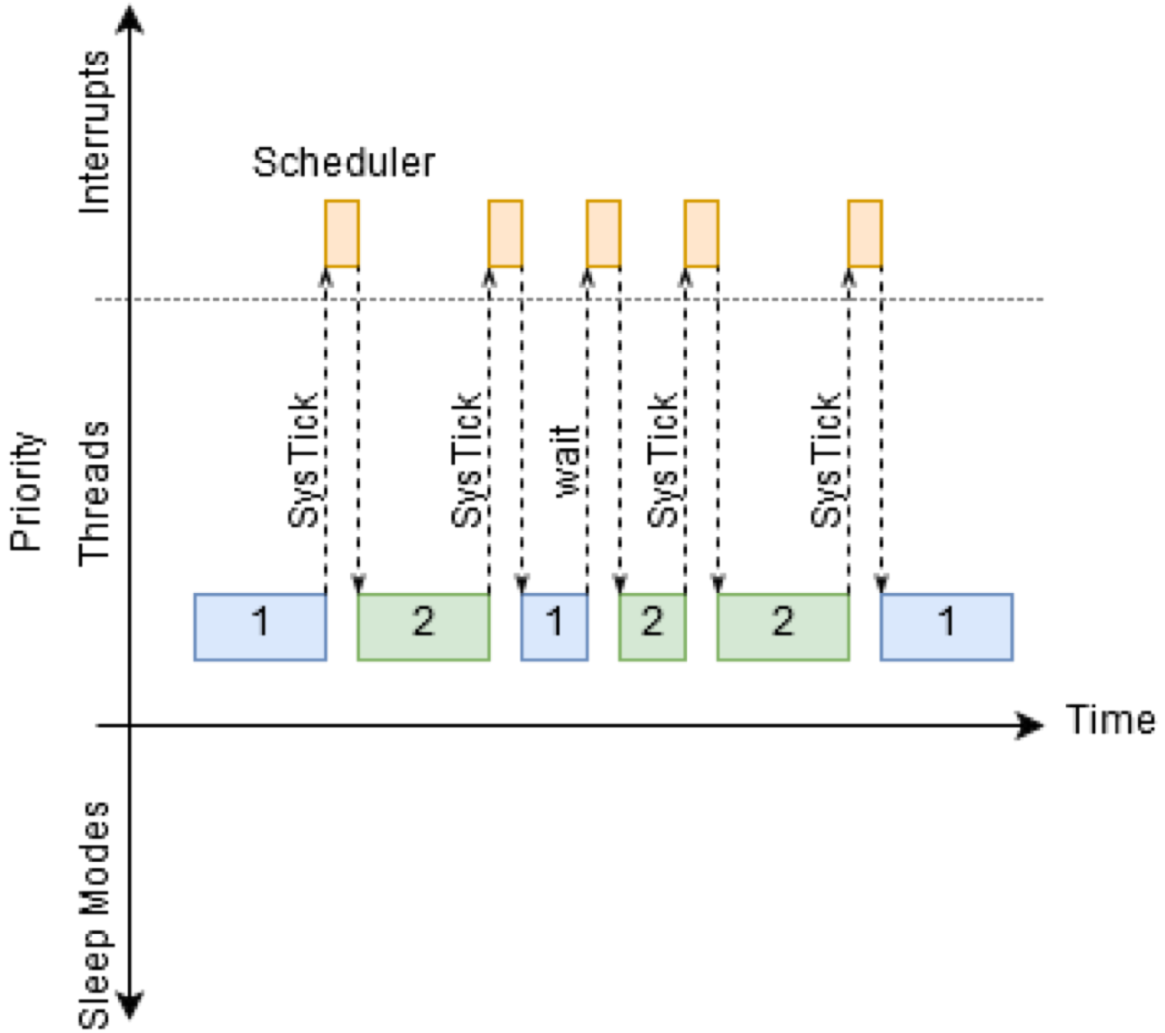
Differences in HW/SW

- What is active during the sleep?
  - Clocks
  - Peripherals
    - SPI, UART, I2C, …
- Wake up sources:
  - GPIO
  - RTC
  - …
- How long does it take for the board to wake up?
  - The deeper the sleep, the longer it can take to wake up

Wait functionality

- Preemptive waiting loop
  - Tasks of the same priority are time sliced with each other

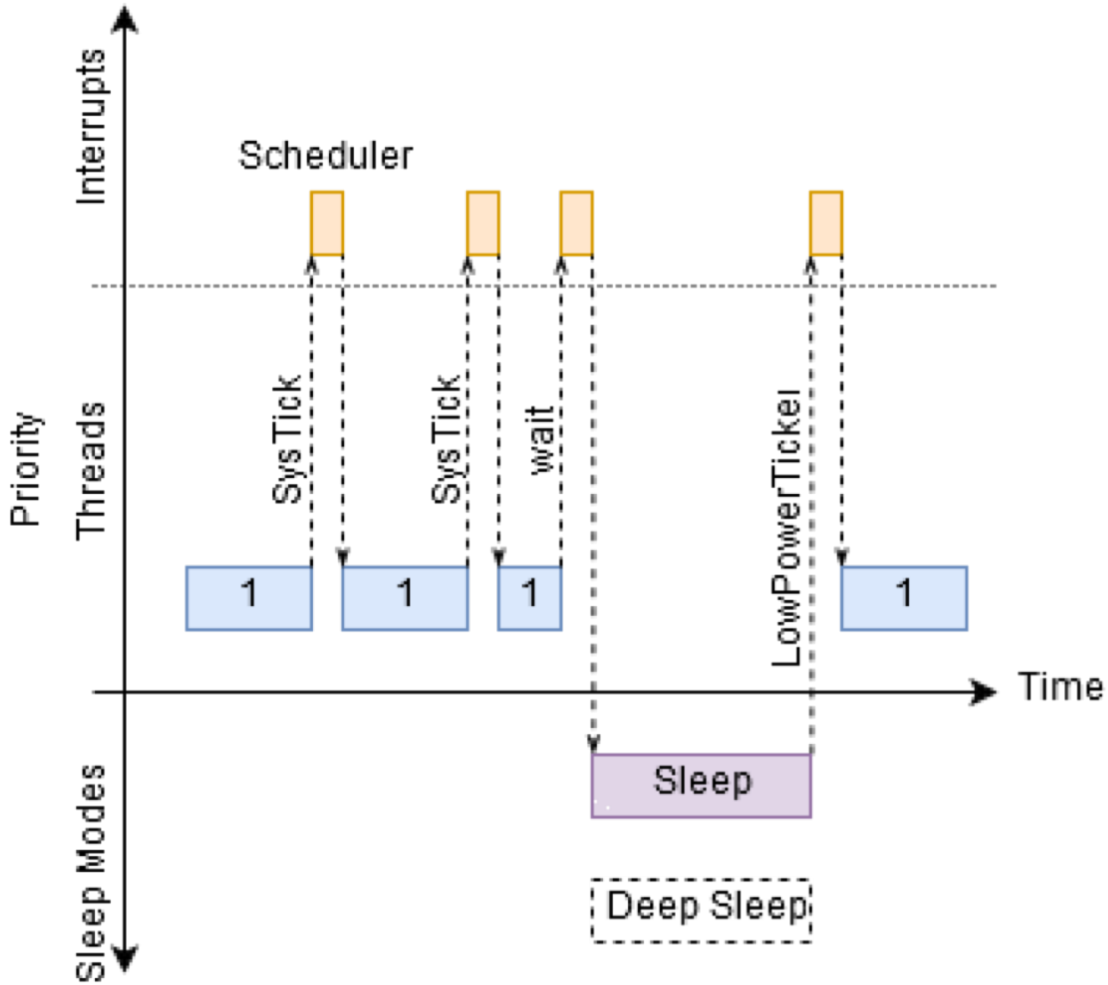
- `wait()`
  - Calls `Thread::wait(ms)`
  - Pauses execution of this thread and allows other threads to run



arm

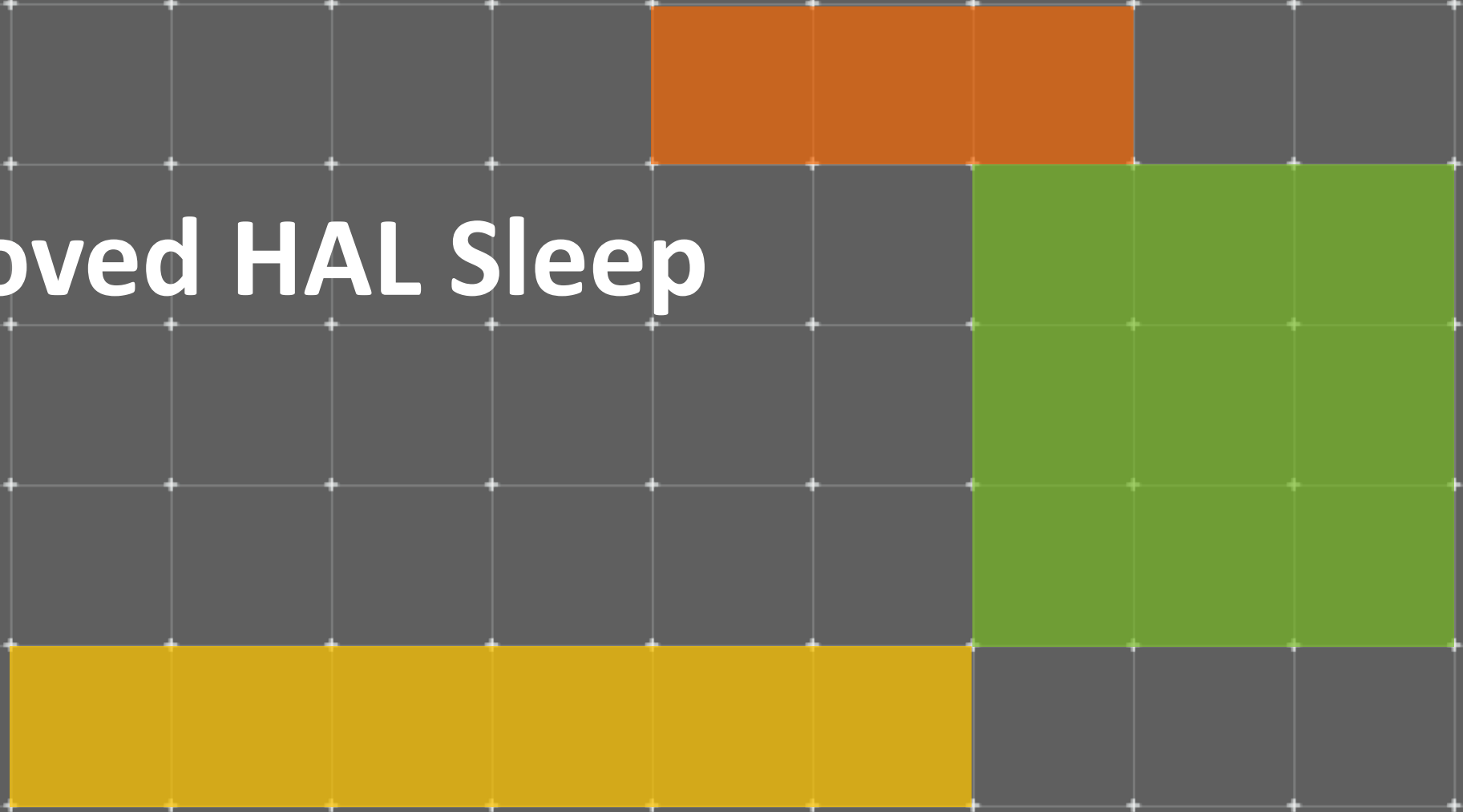
RTOS Idle loop

- No threads ready for running – invoke idle loop
- Idle loop then enters sleep by default
  - No action is required by the application code
  - Application hook to customize the idle loop
  - `rtos_attach_idle_hook(void (*fptr)(void))`



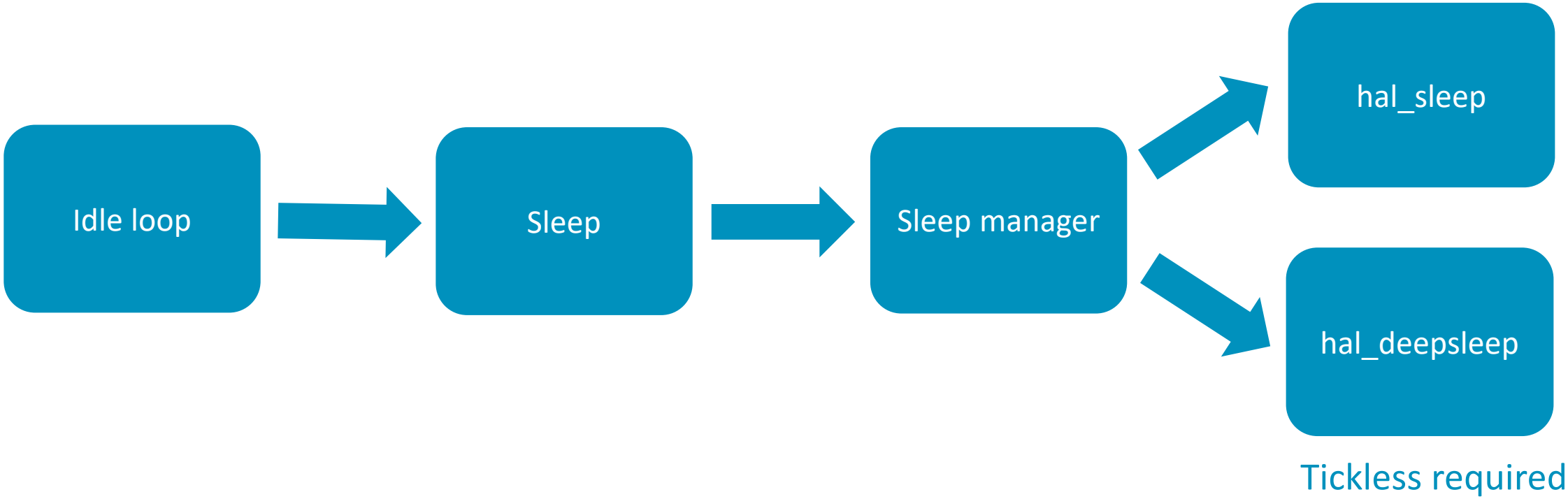
arm

Improved HAL Sleep



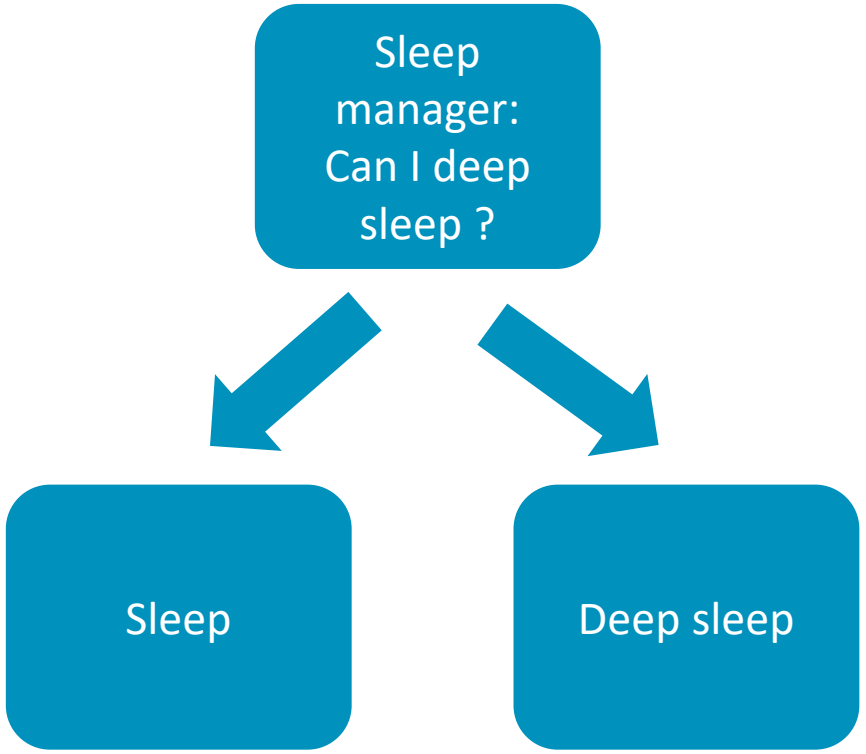
arm

New architecture (Mbed OS 5.6)



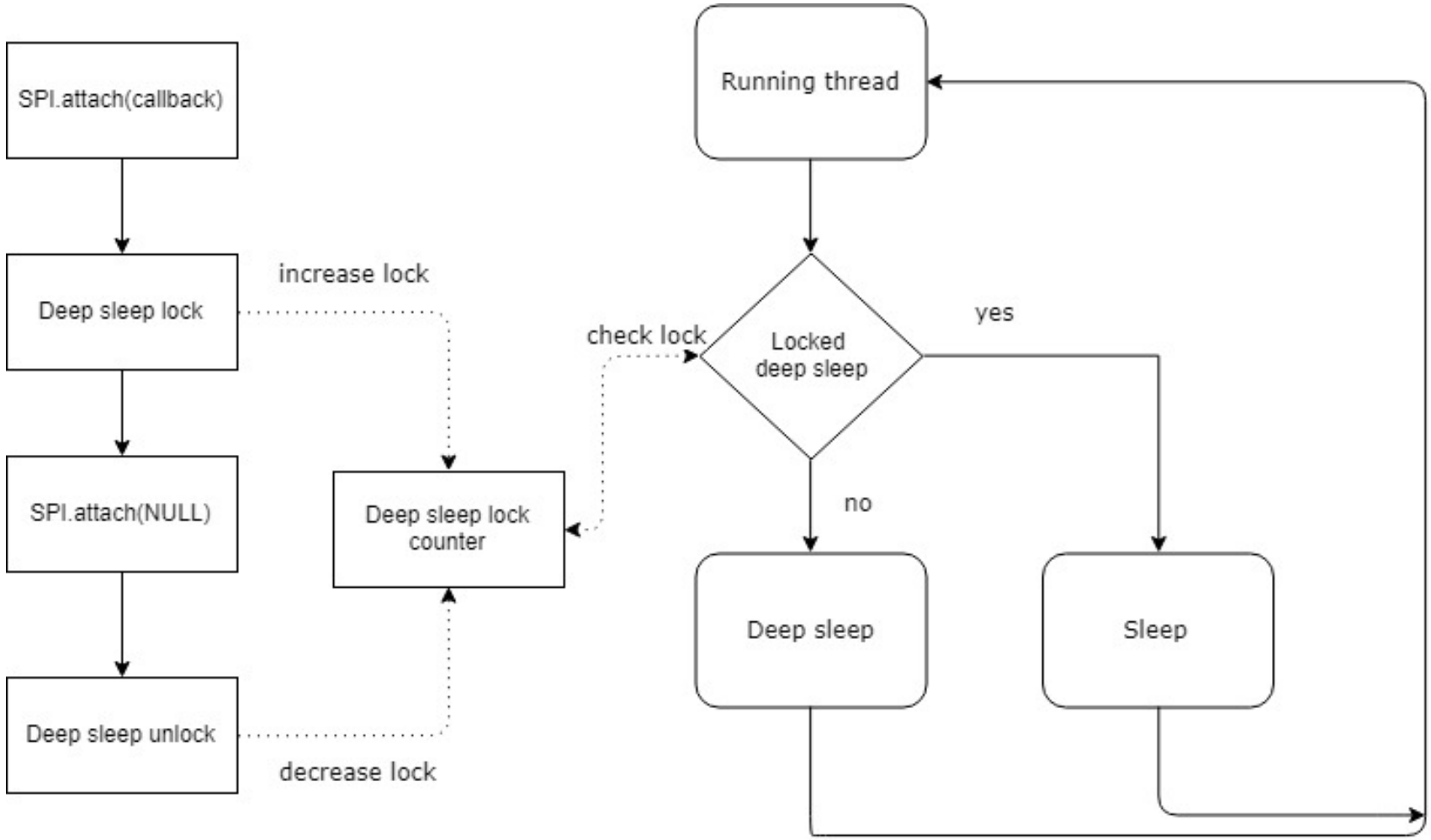
Sleep manager

- Lock/unlock deep sleep
  - Deep sleep internal counter
- Auto sleep function
  - Chooses the best sleep mode available
  - Invoked in the sleep function



arm

Sleep decision flow



DeepSleepLock

- RAII object for disabling, then restoring the deep sleep mode
  - Lock object unlocks deep sleep during destruction

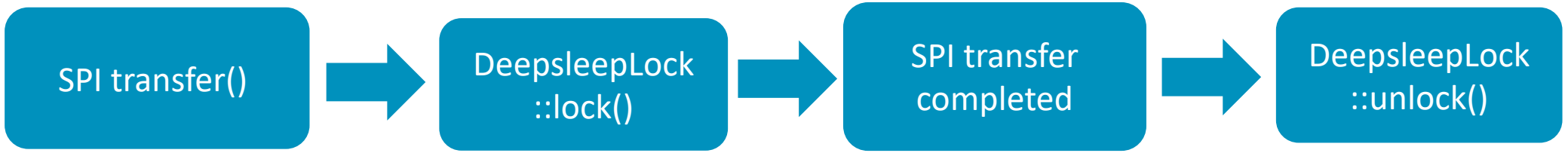
```
void pressure_measurement(SensorPressure &sensor)
{
    DeepSleepLock lock;
    sensor.attach(callback);
    sensor.start_measurement();

    // measurement takes some time, release this thread
    wait(500ms);
    result[current_reading++] = sensor.read();
}
```

DeepSleepLock

- Provides also lock/unlock methods (useful for asynchronous objects)

`SPI::transfer(tx buff, rx buff)` - asynchronous method



arm

Changes to HAL

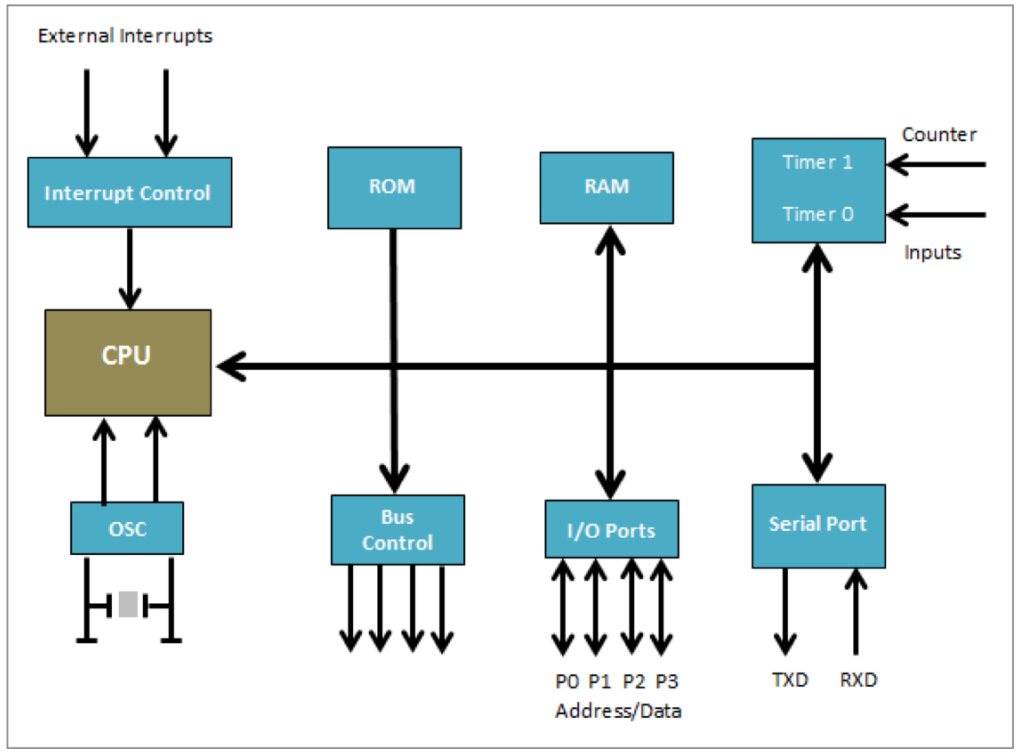
- Specification for both sleep modes
  - Status of peripherals
  - Wakeup latency
  - Wakeup sources
- Tests
- Porting guide

All of these will be discussed in detail in later slides …

arm

Peripherals status during sleep

- Core system clock is disabled
- RAM is retained
- Sleep
  - Low- and high-frequency clocks are active
- Deep Sleep
  - Only low-frequency clocks are active
  - Target should disable high speed frequency clocks



Drivers – deep sleep locked

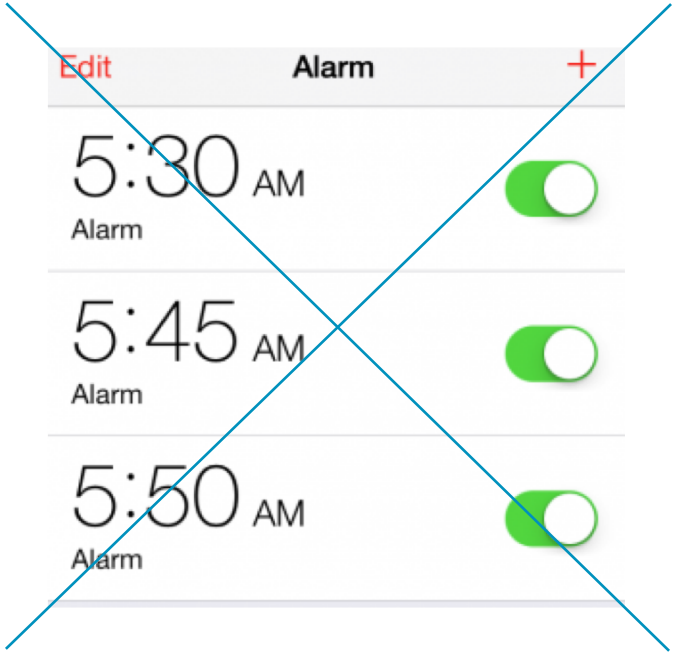
- CAN
- I2C
- PWMOut
- SerialBase
- SPI
- Ticker
- Timer

- Custom drivers – if depends on high speed frequency clock, should lock

arm

Wakeup latency

- Device has to wake up and return to application within a given time
- Sleep
  - 10 us
- Deep sleep
  - 10 ms



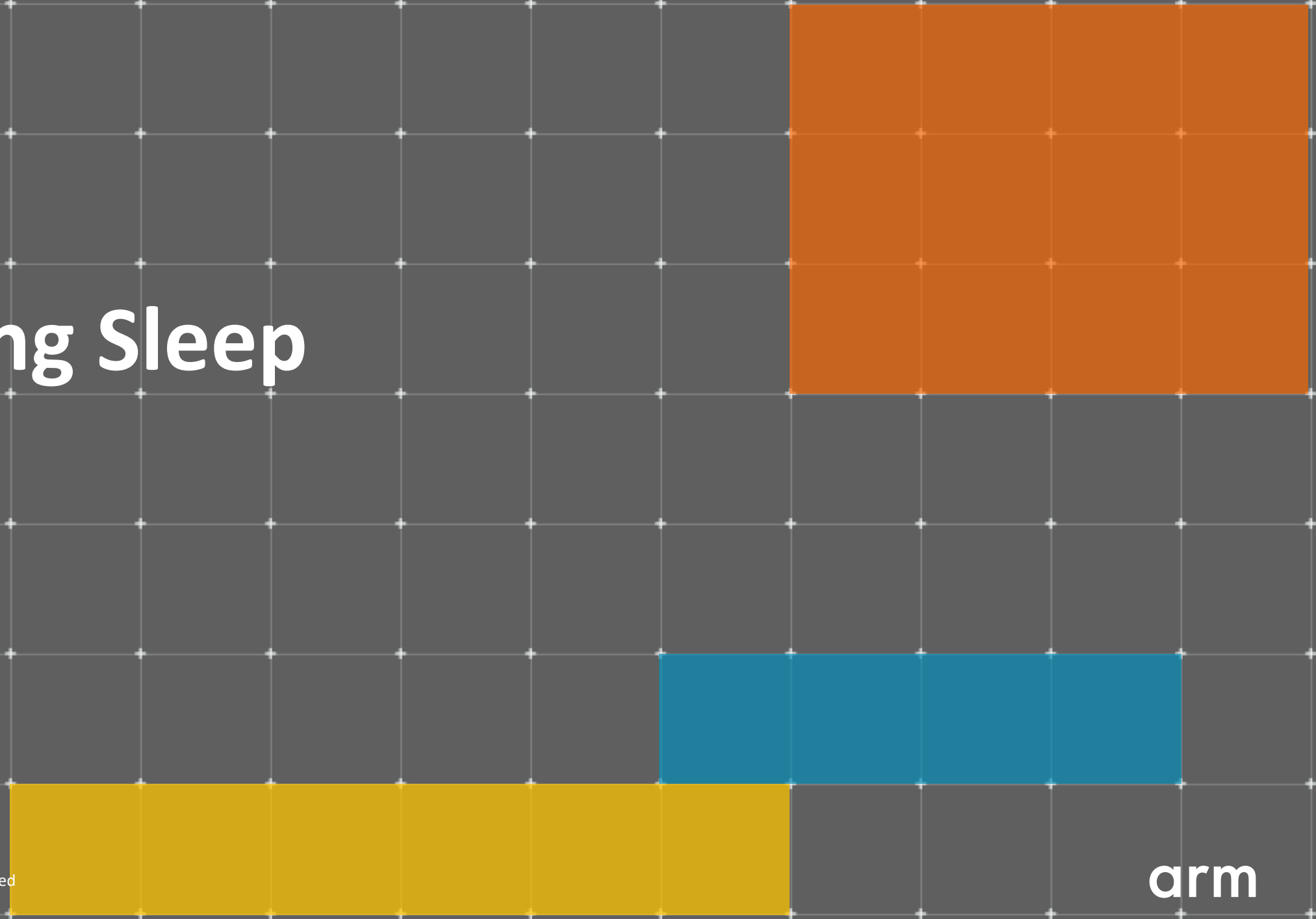
arm

Wakeup Sources

- Target must be able to wake up from specified sources
- Sleep:
  - Any interrupt
- Deep Sleep:
  - Watchdog
  - Low power ticker
  - GPIO

arm

Porting Sleep



arm

Sleep target support

- Add `SLEEP` to `device_has` in targets.json

  https://github.com/ARMmbed/mbed-os/blob/master/targets/targets.json

```
"LPC1768": {
        "inherits": ["LPCTarget"],
        "core": "Cortex-M3",
        "extra_labels": ["NXP", "LPC176X", "MBED_LPC1768"],
        "supported_toolchains": ["ARM", ..., "GCC_CR", "IAR"],
        "detect_code": ["1010"],
        "device_has": ["ANALOGIN", ..., "SLEEP"],
        "release_versions": ["2", "5"],
        "features": ["LWIP"],
        "device_name": "LPC1768",
        "bootloader_supported": true
    },
```

- Implement sleep functions:

  https://github.com/ARMmbed/mbed-os/blob/master/hal/sleep_api.h

Implement sleep HAL

```
void hal_sleep(void);
```

- Send the microcontroller to sleep
- System clock to the core is stopped
- The processor, peripheral and memory state are maintained
- Peripherals continue to work and can generate interrupts
- Can be woken up by any internal peripheral interrupt or external pin interrupt
- Needs to resume app within 10 us

Implement deep sleep HAL

```
void hal_deepsleep(void);
```

- Send the microcontroller to deep sleep
- System clock to the core is stopped
- The processor and memory state are maintained
- Peripherals and high frequency clocks are powered down
- Can only be woken up by low power ticker, RTC, external pin interrupt or a watchdog timer
- Needs to resume app within 10 ms
- Target should select the best deep sleep mode

Implement sleep/deep sleep

`void hal_sleep(void);`	`void hal_deepsleep(void);`
Send the microcontroller to sleep	Send the microcontroller to deep sleep
System clock to the core is stopped	System clock to the core is stopped
The processor, peripheral and memory state are maintained	The processor and memory state are maintained (peripherals are not)
Peripherals continue to work and can generate interrupts	Peripherals and high frequency clocks are powered down
Can be woken up by any internal peripheral interrupt or external pin interrupt	Can only be woken up by low power ticker, RTC, external pin interrupt or a watchdog timer
Needs to resume app within 10 **us**	Needs to resume app within 10 **ms**

What tests to pass?

- HAL Sleep validation suite:

```
$ git checkout feature-hal-spec-sleep
$ mbed test -t <toolchain> -m <target> -n "tests-mbed_hal-sleep*"
```


arm

Hands-on workshop

- Sleep
  - Use branch - `feature-hal-spec-sleep`
  - HAL API & Testing - https://os.mbed.com/docs/v5.7/feature-hal-spec-sleep-doxy/group__hal__sleep.html
  - Porting - https://os.mbed.com/docs/v5.7/reference/contributing-target.html#sleep

- Workshop materials - https://github.com/ARMmbed/mbed-sip-workshop-2018q1

arm