arm

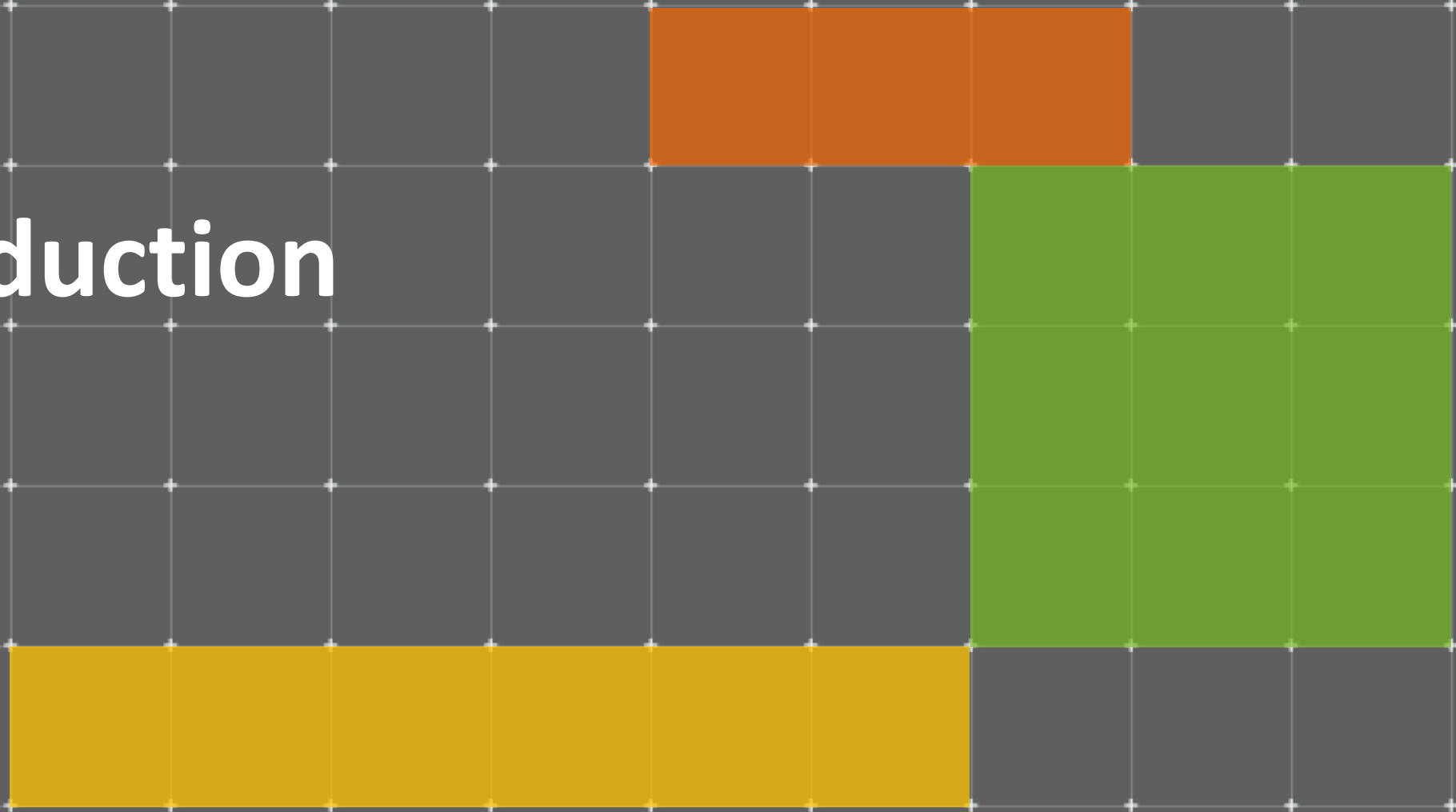
Mbed Silicon Partner Workshop

HAL Improvements Tickers

27th of March 2018

Paris, France

Introduction



arm

Introduction - Tickers

In Mbed OS, tickers are hardware timers used to track small amounts of time

Mbed OS has two types of tickers:

- High-resolution microsecond tickers
  - Prevents deep sleep mode
  - User API: Timer, Timeout, and Ticker classes
  - HAL API: `us_ticker`
- Low-power tickers
  - User API: LowPowerTimer, LowPowerTimeout, and LowPowerTicker classes
  - HAL API: `lp_ticker`

Introduction - Tickers

Tickers share a common layer in Mbed OS which provides several conveniences for developers

- Handling of roll-over when ticker overflows
- Extends tickers to 64-bits, supports any hardware bit-width
- Queues multiple callbacks on a single interrupt
- User APIs for deferring from interrupt context

In Mbed, pretty much everything depends on tickers.

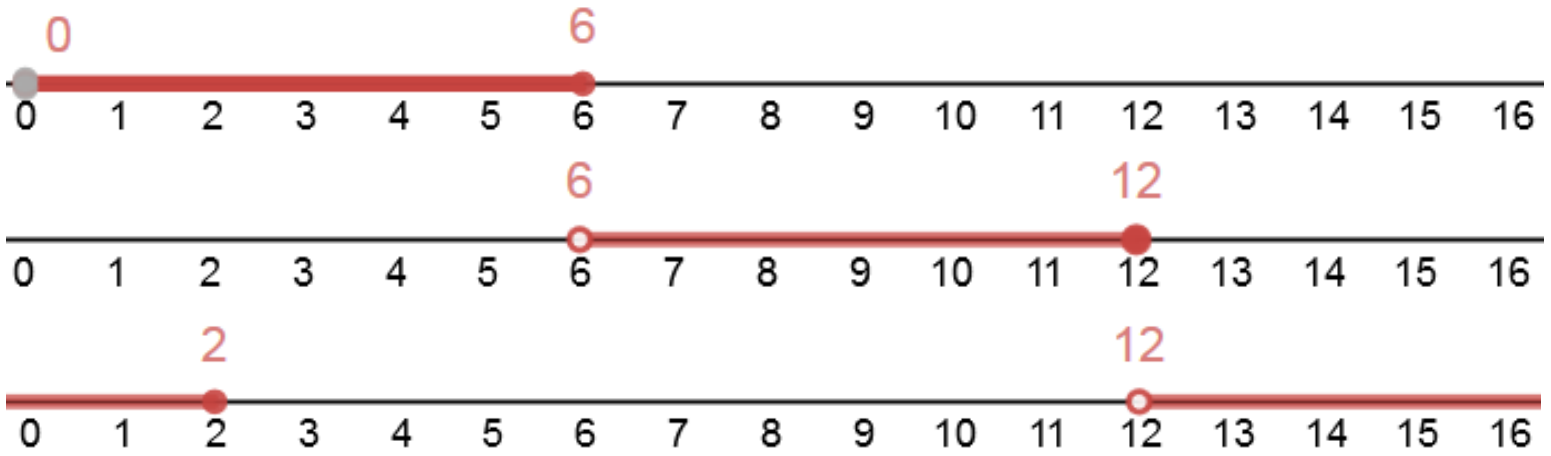
arm

Tickers - Overflow

- Different MCU's have different bit-width timers
- Handling timer overflow can be tricky

```
while (us_ticker_read() < timeout + start_time) // ✗ Wrong!
while (us_ticker_read() - start_time < timeout) // ✓ Right!
```

- Timer overflow bugs could take many days before they occur



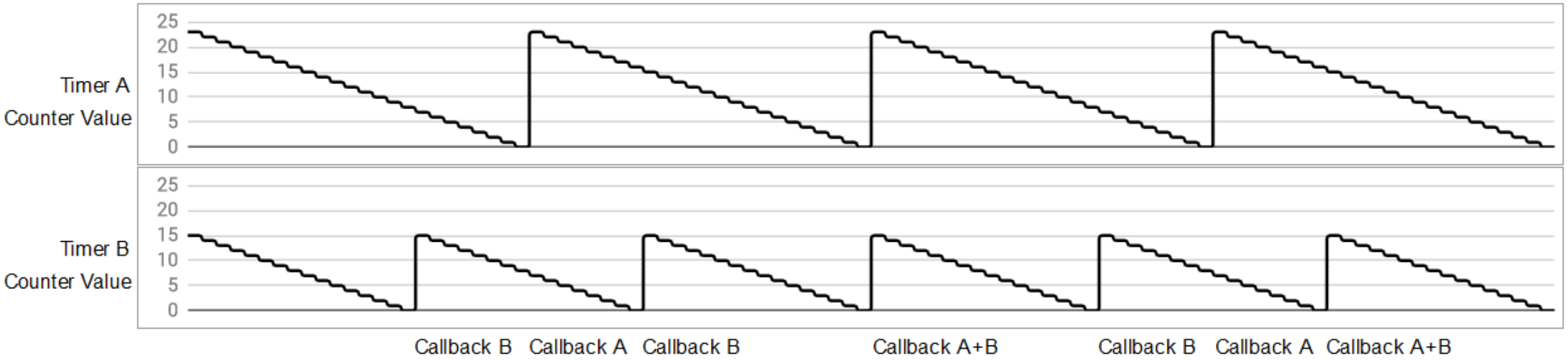
arm

Tickers - Overflow

- Mbed OS handles timer overflow transparently for the user
  - Every interrupt, we check if we've travelled backwards in time
  - We add the time delta and any overflow to a 64-bit software timer
  - If there are no scheduled interrupts, a lightweight interrupt is scheduled to make sure we don't miss any overflows
- Smaller timers require more overflow interrupts
- User facing timer API supports 64-bits
- 64-bit timer will overflow in the year 586602

Tickers – without Queueing

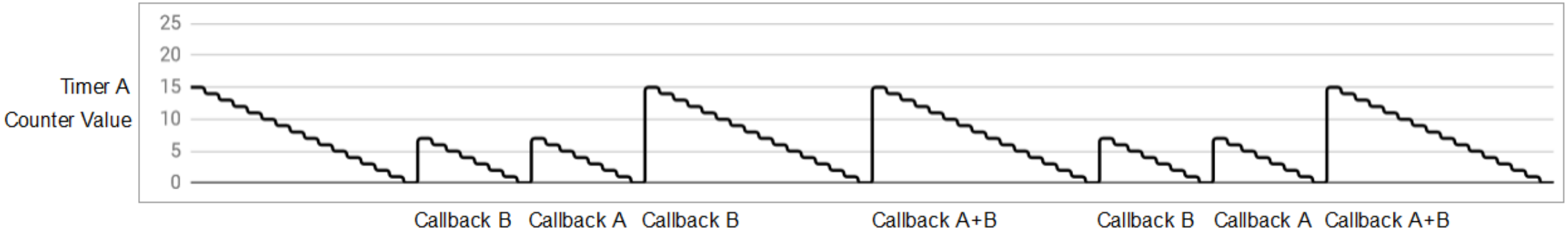
- A hardware timer supports a single interrupt
- To support multiple timing callbacks a simple solution is to use a hardware timer for each timing event.
  - Each hardware timer consumes additional power
  - The number of timing callbacks is limited by the available hardware



arm

Tickers – with Queueing

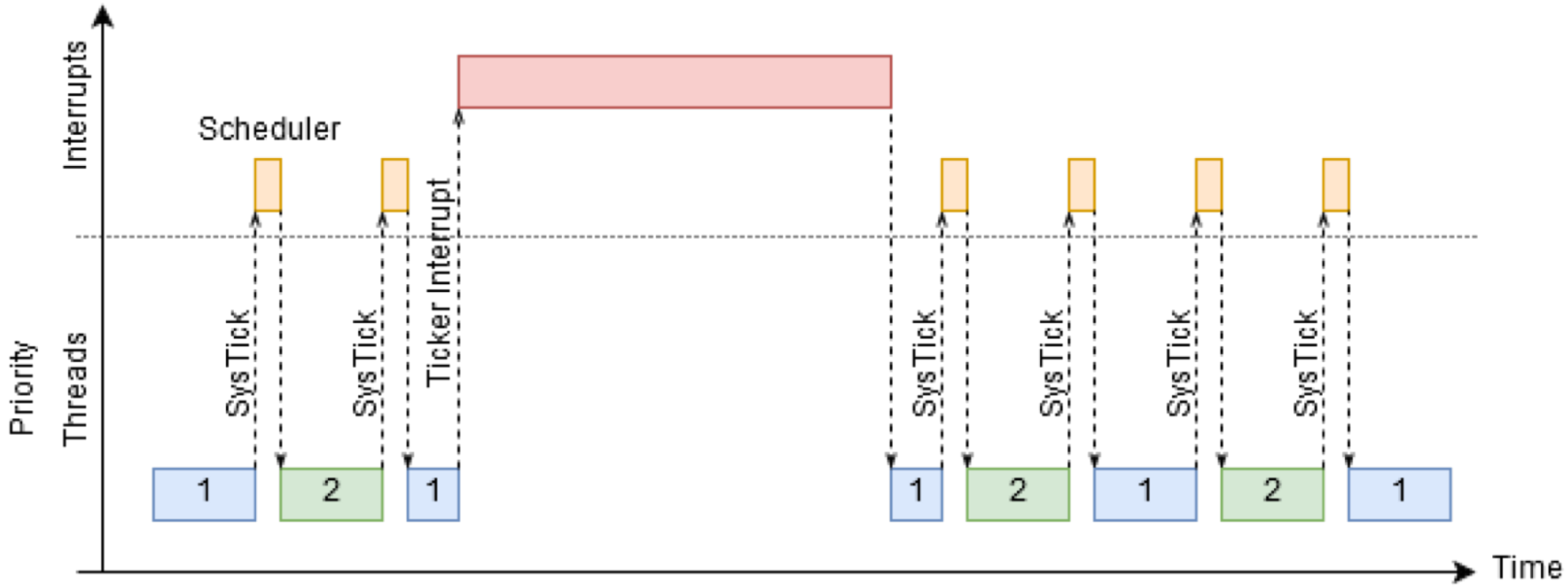
- Mbed OS provides a lightweight software queue to dispatch multiple timing callbacks on a single hardware timer.
  - Consumes fixed amount of power
  - Unlimited timing callbacks
- Provided by Mbed OS, does not require support in driver
- Handled by `ticker_irq_handle` which is called by the hardware timer interrupt
- Separate queues for each type of ticker (Ticker, LowPowerTicker)



arm

Tickers - Interrupt Context

- Ticker callbacks run in interrupt context
- Code in interrupt context needs to have bounded runtime, or bad things happen
  - Prevents same-priority interrupts, which may be timing sensitive
  - Blocks threads from running
  - RTOS synchronization cannot be used in interrupt context

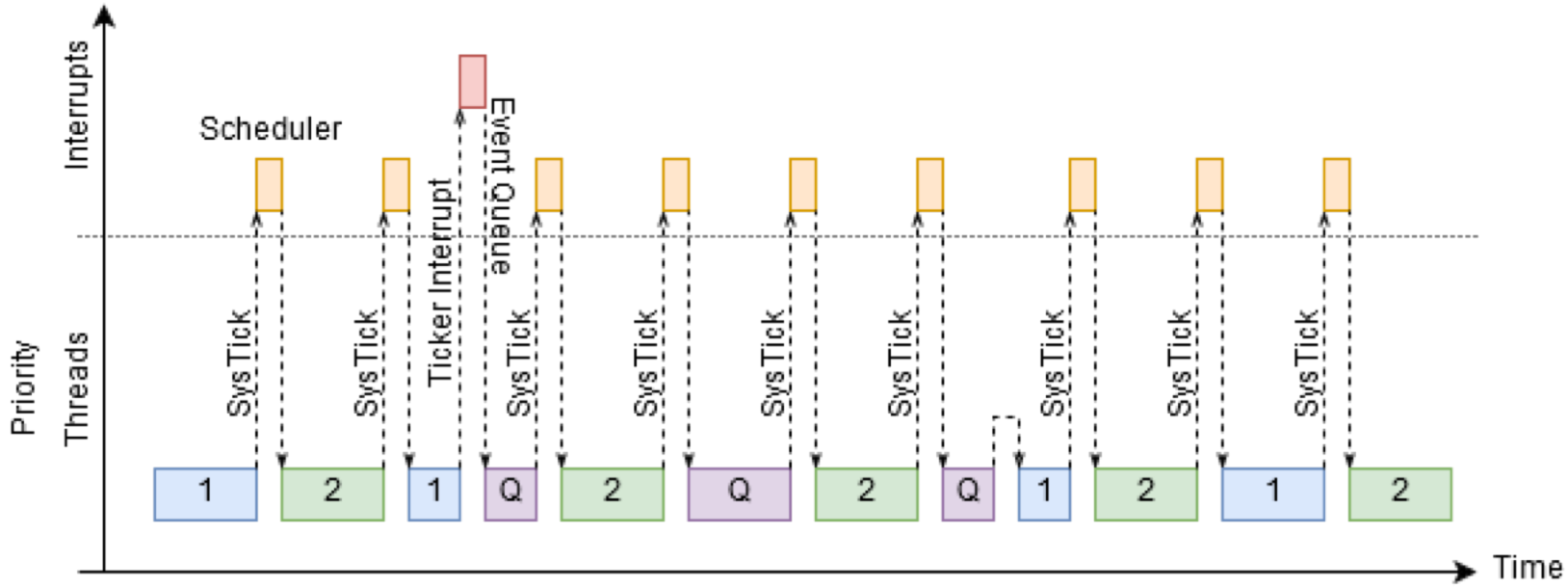


arm

Tickers - Interrupt Context

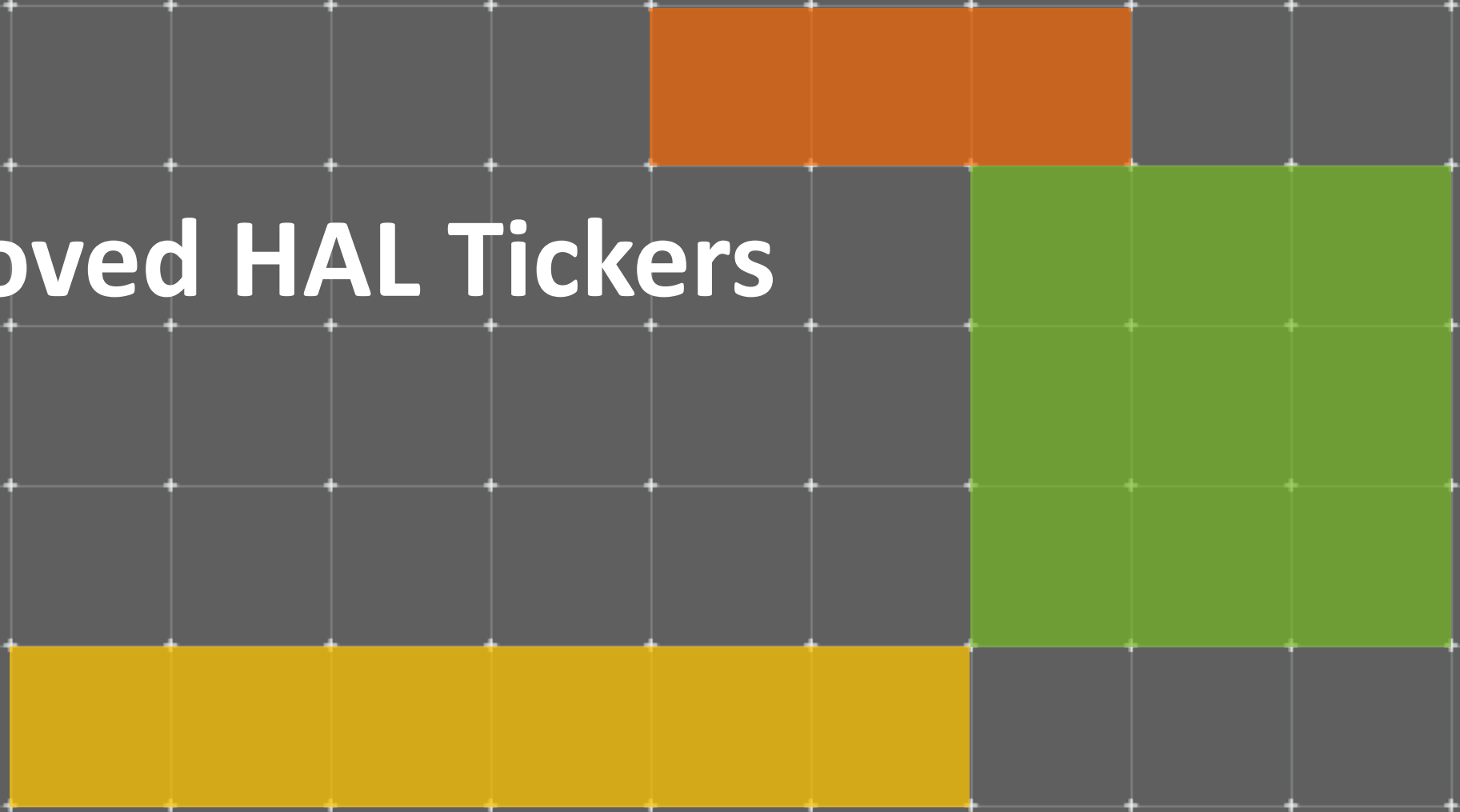
- Mbed OS provides RTOS and event queue primitives for deferring callbacks from interrupt context

  https://os.mbed.com/docs/v5.7/tutorials/the-eventqueue-api.html



arm

Improved HAL Tickers



arm

Changes to HAL

- Simplified HAL API
  - Ticker now return native value and configuration data
  - Mbed OS driver is responsible for frequency conversion
- Updated specification
- Revised porting guide
- New validation test suite

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

arm

Common ticker specification

Highlights

- The ticker rolls over and counts upward starting from 0
- The ticker counts at the specified frequency ±10%
- The ticker increments by 1 each tick
- The ticker interrupt fires only when the ticker increments to or past the value set by `ticker_set_interrupt()`

Microsecond ticker specification

- Frequency between 250KHz and 8MHz
- Counter at least 16 bits wide
- Does not hold true time during deep sleep mode

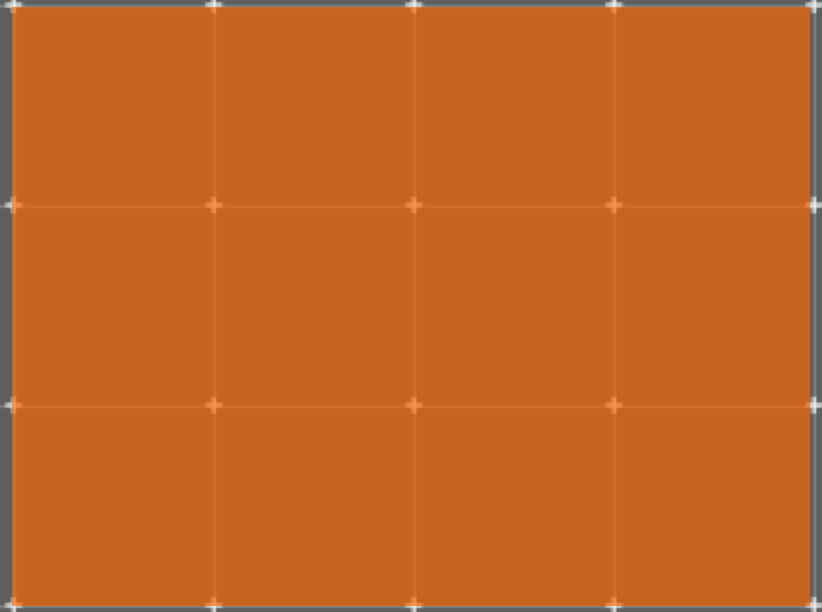
https://github.com/ARMmbed/mbed-os/blob/feature-hal-spec-ticker/hal/us_ticker_api.h

Low power ticker specification

- Frequency between 8KHz and 64KHz
- Counter at least 12 bits wide
- Continues operating in deep sleep mode

https://github.com/ARMmbed/mbed-os/blob/feature-hal-spec-ticker/hal/lp_ticker_api.h

Porting Tickers



arm

Implementing HAL functions

Microsecond ticker

- Add `USTICKER` 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", ..., "USTICKER"],
        "release_versions": ["2", "5"],
        "features": ["LWIP"],
        "device_name": "LPC1768",
        "bootloader_supported": true
    },
```

- Implement the microsecond ticker functions:

  https://github.com/ARMmbed/mbed-os/blob/feature-hal-spec-ticker/hal/us_ticker_api.h

arm

Implementing HAL functions

Low power ticker

- Add `LPTICKER` 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", ..., "LPTICKER"],
        "release_versions": ["2", "5"],
        "features": ["LWIP"],
        "device_name": "LPC1768",
        "bootloader_supported": true
    },
```

- Implement the low power ticker functions:

  https://github.com/ARMmbed/mbed-os/blob/feature-hal-spec-ticker/hal/lp_ticker_api.h

arm

Implementing init and free

```
void us_ticker_init(void);
void us_ticker_free(void);
```

```
void lp_ticker_init(void);
void lp_ticker_free(void);
```

- `*_ticker_init()`
  - Initialize or re-initialize the ticker and disables the ticker interrupt
  - Resets all the clocking and prescaler registers
  - Is safe to call repeatedly
  - Calling any function other than `*_ticker_init()` before the initialization of the ticker is undefined
- `*_ticker_free()`
  - Deinitialize the microsecond ticker
  - No other call than `*_ticker_init()` is allowed after this function
  - Stops the ticker from counting

Implementing read and get info

```
uint32_t us_ticker_read(void);
const ticker_info_t*
  us_ticker_get_info(void);
```

```
uint32_t lp_ticker_read(void);
const ticker_info_t*
  lp_ticker_get_info(void);
```

- `*_ticker_read()`
  - Reads native value of the counter in ticks
  - No frequency conversion should be made
- `*_ticker_get_info()`
  - Returns details of the ticker
    - Frequency in Hz
    - Counter width in bits

```
typedef struct {
    uint32_t frequency;
    uint32_t bits;
} ticker_info_t;
```

Implementing disable, set and clear interrupt

```
void us_ticker_set_interrupt(timestamp_t
  timestamp);
void us_ticker_disable_interrupt(void);
void us_ticker_clear_interrupt(void);
```

```
void lp_ticker_set_interrupt(timestamp_t
  timestamp);
void lp_ticker_disable_interrupt(void);
void lp_ticker_clear_interrupt(void);
```

- `*_ticker_set_interrupt(timestamp_t timestamp)`
  - Sets interrupt for timestamp specified in ticks
  - Timestamps in the past are detected and handled in Mbed OS driver
  - Timestamps exceeding counter's bit width are not supported
  - Safe to call multiple times before the interrupt fires
- `*_ticker_disable_interrupt()`
  - Disables ticker interrupt
- `*_ticker_clear_interrupt()`
  - Clears ticker interrupt

arm

Implementing fire interrupt

```
void us_ticker_fire_interrupt(void);
```

```
void lp_ticker_fire_interrupt(void);
```

- `*_ticker_fire_interrupt()`
  - Sets pending interrupt that should be fired right away

What tests to pass?

- Microsecond ticker validation suite:

```
$ git checkout feature-hal-spec-ticker
$ mbed test -t <toolchain> -m <target> -n \
    "tests-mbed_hal-common_tickers*,tests-mbed_hal-us_ticker*"
```

- Low power validation suite:

```
$ git checkout feature-hal-spec-ticker
$ mbed test -t <toolchain> -m <target> -n \
    "tests-mbed_hal-common_tickers*,tests-mbed_hal-lp_ticker*"
```

Hands-on workshop

- Microsecond ticker
  - Use branch - `feature-hal-spec-ticker`
  - HAL API & Testing - https://os.mbed.com/docs/v5.7/feature-hal-spec-ticker-doxy/index.html
  - Porting - https://github.com/ARMmbed/mbed-sip-workshop-2018q1/blob/master/us_ticker.md
- Low-power ticker
  - Use branch - `feature-hal-spec-ticker`
  - HAL API & Testing - https://os.mbed.com/docs/v5.7/feature-hal-spec-ticker-doxy/group__hal__lp__ticker.html
  - Porting - https://github.com/ARMmbed/mbed-sip-workshop-2018q1/blob/master/us_ticker.md
- Workshop materials - https://github.com/ARMmbed/mbed-sip-workshop-2018q1

arm

Ticker example

Asynchronous blinky:

```
#include "mbed.h"

DigitalOut led1(LED1);
Ticker ticker1;

void blink() {
    led1 = !led1;
}

int main() {
    ticker1.attach_us(blink, 500*1000);
}
```

Tasks:

- Bring up ticker and low power ticker on your boards
- Blink multiple LEDs?
- Switch to low power ticker?

Thank You!
Danke!
Merci!
谢谢!
ありがとう!
Gracias!
Kiitos!
감사합니다
धन्यवाद