Technische Hochschule Rosenheim
Technical University of Applied Sciences

Prof. Dr. Florian Künzner
Technical University of Applied Sciences Rosenheim, Computer Science

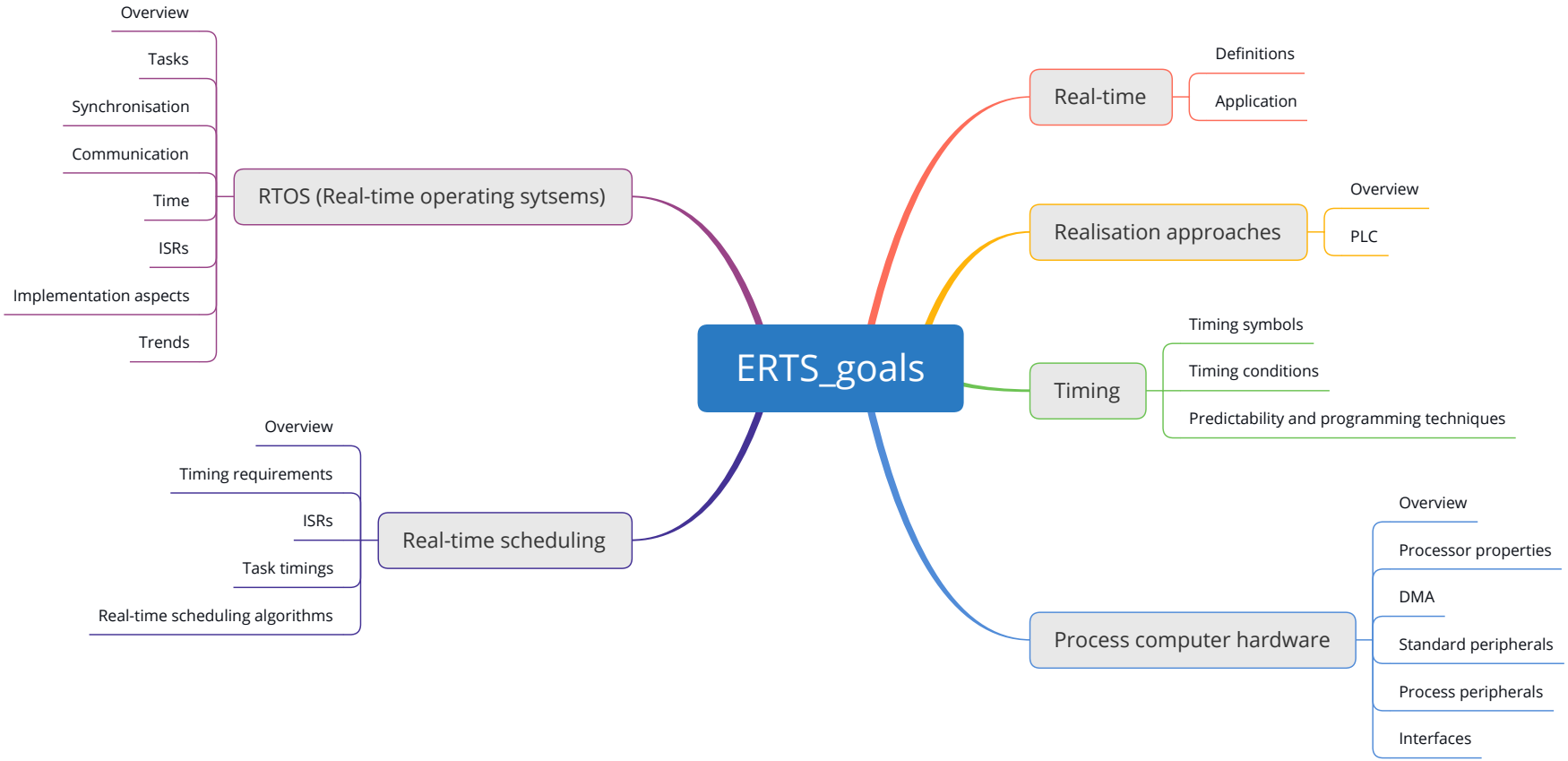
ERTS - Embedded real-time systems

ERTS 9 – FreeRTOS

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Goal



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Goal

ERTS::FreeRTOS

- FreeRTOS intro
- FreeRTOS customisation
- FreeRTOS task scheduling
- FreeRTOS task coding

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

FreeRTOS

- Main Author: Richard Barry
- Started from Richard Barry around 2003
- Since 2017, acquired by Amazon Web Services (AWS)
- IoT connection stack to AWS cloud
- Supports up to 35 platforms
- License: MIT open source license
- Kernel + growing set of libraries

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

FreeRTOS – features

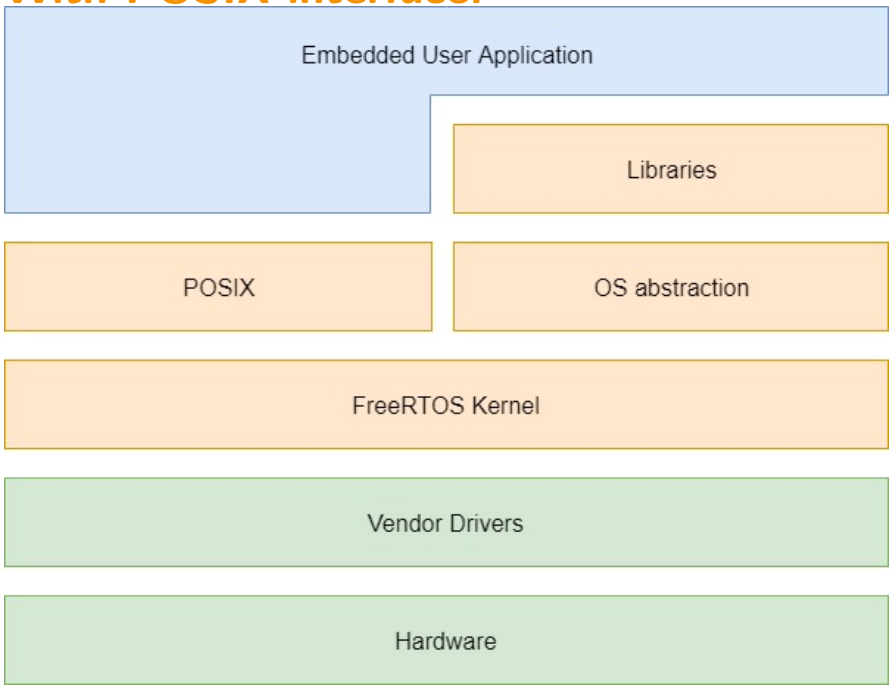
- Tasks and scheduling
- Synchronisation and communication
  - Queues, mutexes, semaphores...
  - Task notifications
  - Events
  - Stream & message buffers
- Software Timers
- Event Groups (or *Flags*)
- Static vs dynamic memory for FreeRTOS objects
- Heap memory management
- Stack Overflow Protection
- MPU support (restricted tasks)

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

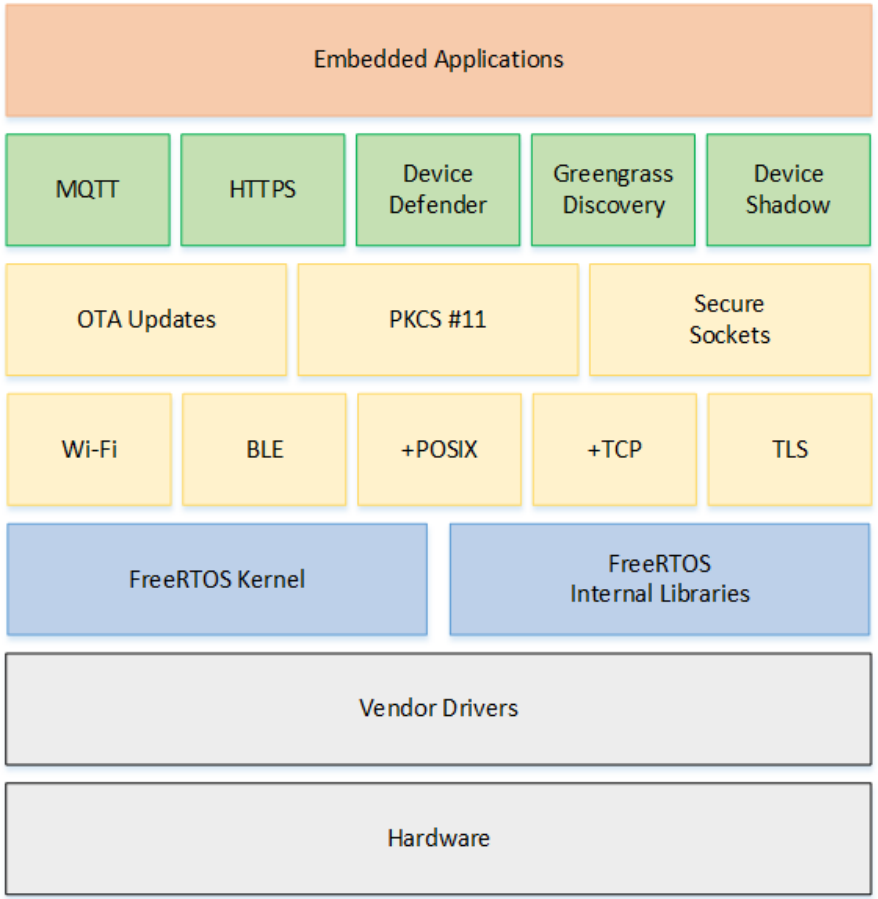
FreeRTOS – architecture

With POSIX interface:



[source: freertos.org]

With Amazon AWS libraries:



[source: aws.amazon.com]

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

FreeRTOS – naming convention

Summary of FreeRTOS naming convention:

- **Variables:**
  - Prefix with type: e.g. `uint32_t` = ul, `uint16_t` = us
  - Non stdint types are prefixed x
- **Functions:**
  - Prefix with return type: e.g. `void` = v, `uint32_t` = ul
  - Function names start with the name of the file in which they are defined
- **Macros:** Prefix with file in which they are defined

More details: Naming Conventions

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

FreeRTOSConfig.h

FreeRTOS can be configured with the `FreeRTOSConfig.h`

Possibilities:

- Enable/disable features
- Define details for:
  - Scheduling
  - Tasks
  - Memory management
  - Timers
  - ...

Within the Arduino framework, the `FreeRTOSConfig.h` is already pre-configured. Additionally, `FreeRTOSVariant.h` exists.

More details: `https://www.freertos.org/a00110.html`

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

FreeRTOS on Arduino Mega
Priorities

`FreeRTOSConfig.h`

- Task priorities

```
1 #define configMAX_PRIORITIES 16
2 //Defines the max available priorities for application tasks.
```

- Default on Arduino: `#define configMAX_PRIORITIES 4`
- But: can be changed to higher values

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

FreeRTOS on Arduino Mega

Scheduler config: SysTick

`FreeRTOSVariant.h`

- SysTick rate

```
1 #define configTICK_RATE_HZ 1000
2 //The frequency of the RTOS tick interrupt.
```

- **But:** The Atmega2560 micocontroller has no SysTick interrupt
- Uses the watchdog timeout for SysTick
- SysTick oscillator: 128 kHz (but this depends on voltage and temperature)
- SysTick duration:
  - `#define portUSE_WDTO WDTO_15MS`
  - $\texttt{t\_STK\_DR} \approx 15\ \texttt{ms}$ (or up to $\texttt{t\_STK\_DR} \approx 17\ \texttt{ms}$ (measured))
- SysTick rate (frequency): `WDG_CR`
  - $\texttt{WDG\_CR} = \frac{1}{15\ \texttt{ms}} \approx 66\ \texttt{HZ}$ (or: $\texttt{WDG\_CR} = \frac{1}{17\ \texttt{ms}} \approx 58\ \texttt{HZ}$)

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

FreeRTOS on Arduino Mega
Scheduler config

FreeRTOSConfig.h

- Preemption

```
1 #define configUSE_PREEMPTION 1
2 //Set to 1 to use the preemptive RTOS scheduler, or 0 to use the
3 //cooperative RTOS scheduler.
```

- Task selection

```
1 #define configUSE_PORT_OPTIMISED_TASK_SELECTION 0
2 //Some FreeRTOS ports have two methods of selecting the next task
3 //to execute - a generic method, and a method that is specific to
4 //that port.
```

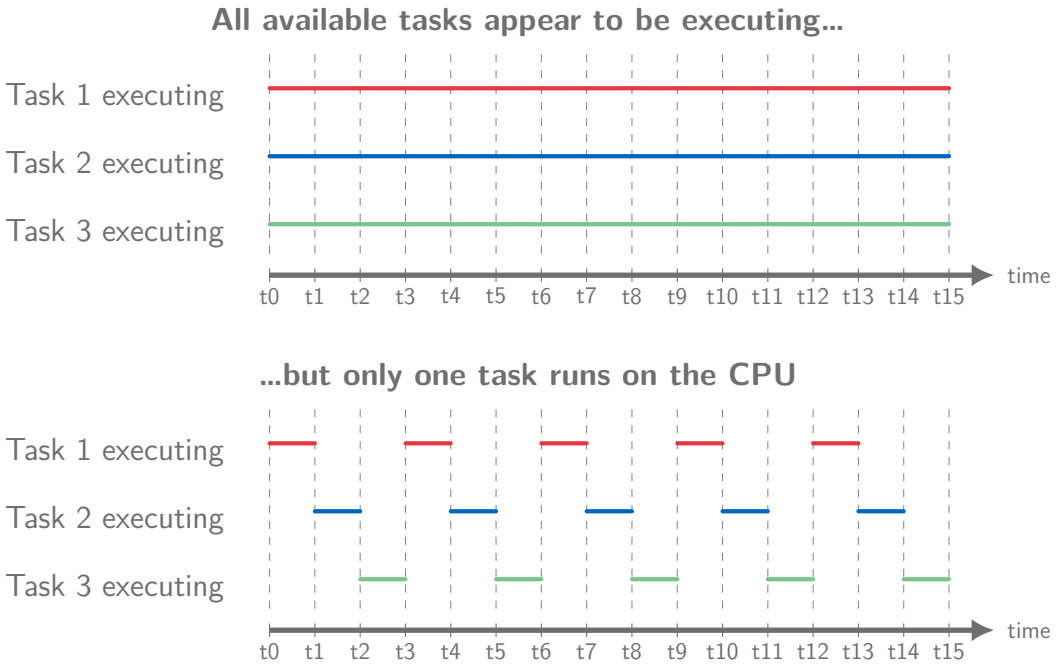
- Time slicing

```
1 #define configUSE_TIME_SLICING 1
2 //If set to 1 (or undefined) time slicing is used for tasks with the
3 //same priority. If set to 0, no time slicing is used for tasks
4 //with the same priority.
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

FreeRTOS – multitasking basics



source: https://www.freertos.org/implementation/a00004.html

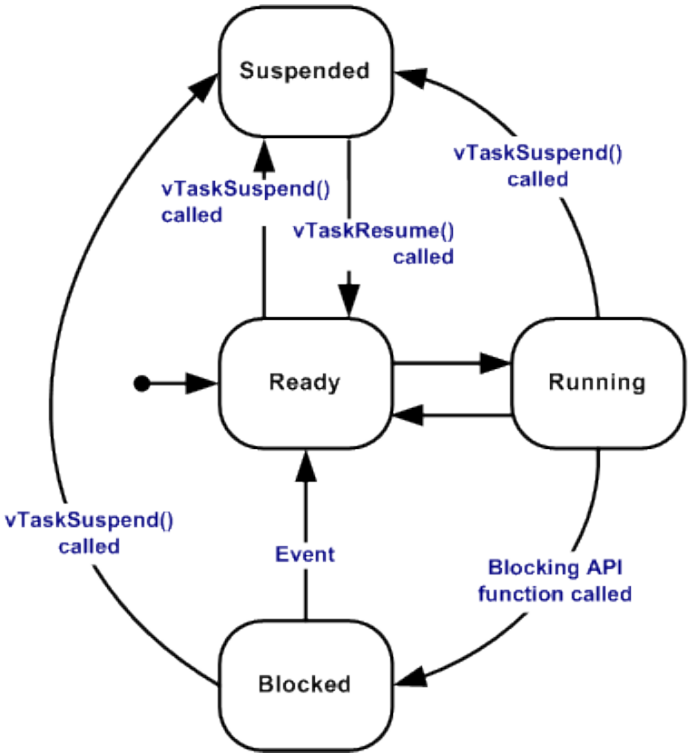
Multitasking:

- Tasks appear to run in parallel
- On Atmega2560, only one CPU is present
- Tasks run alternating: one after another

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

FreeRTOS – task states



source: https://www.freertos.org/RTOS-task-states.html

States:

- Ready: Task is ready to run
- Running: Task (only one) runs now
- Suspended: Task is suspended
- Blocked: Task is blocked (e.g. semaphore, delay, ...)

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Interrupt and task priorities overview

Atmega2560 interrupt priorities combined with the FreeRTOS task priorities.

	Config	Priority	Description
Task priorities			
(ISR deferred task) and	`configMAX_PRIORITIES` ⇒	4	⇐ highest prio
		3	Usable task priorities
	`configTIMER_TASK_PRIORITY` ⇒	2	
	Reserved for idle task	1	⇐ lowest prio
Interrupt/exception priorities			
		57	⇐ lowest prio
		...	
WDT (watchdog timeout) used for FreeRTOS SysTick		13	**Can** use interrupt safe FreeRTOS API
		...	
INT0 (external interrupt request 0)		2	
RESET		1	⇐ highest prio

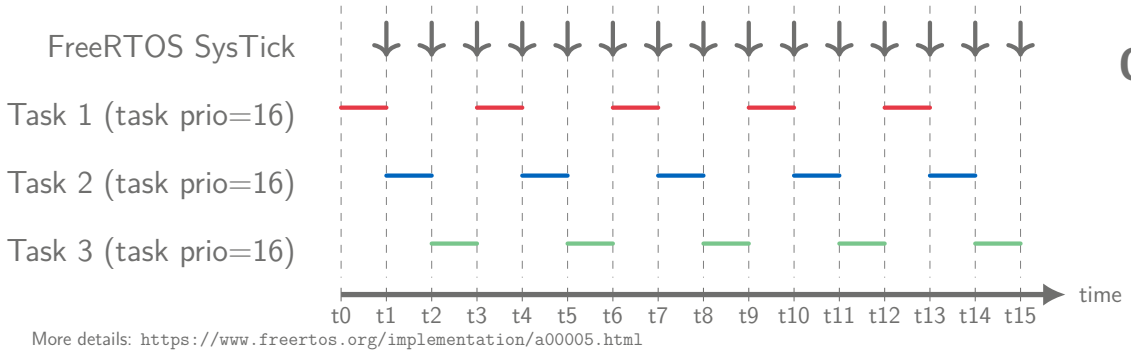
Interrupts have higher prio than tasks

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

FreeRTOS – scheduling basics

Default scheduling strategy: pre-emption with time slicing



More details: https://www.freertos.org/implementation/a00005.html

On every FreeRTOS SysTick:

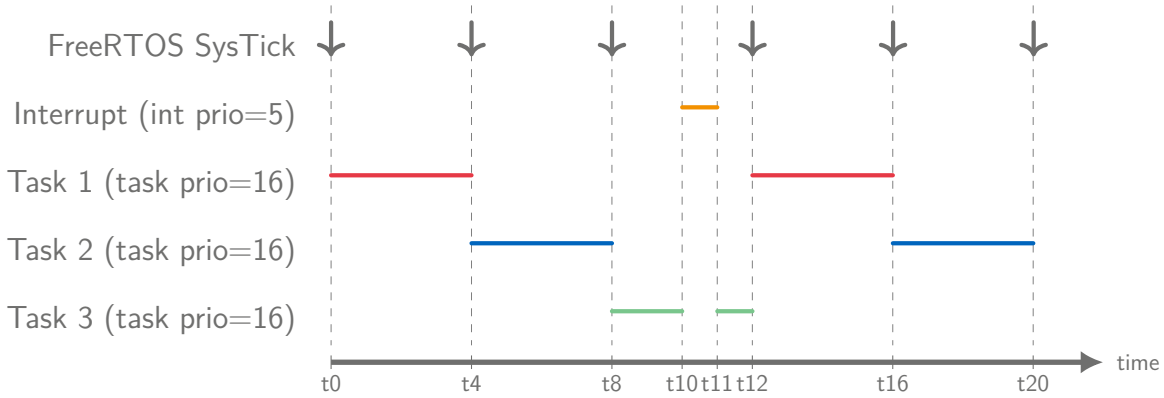
- The task is changed
- It uses round robin (RR)
- RR is used between similar prio

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

FreeRTOS – scheduling basics – interrupt

Default scheduling strategy: pre-emption with time slicing, and an interrupt



Scheduling:

- `t0`: Scheduler: Task 1 runs
- `t4`: Scheduler: Task 2 runs
- `t8`: Scheduler: Task 3 runs
- `t10`: Interrupt: ISR
- `t11`: ISR end: Task 3 runs
- `t12`: Scheduler: Task 1 runs
- `t16`: Scheduler: Task 2 runs
- `t20`: Scheduler: ...

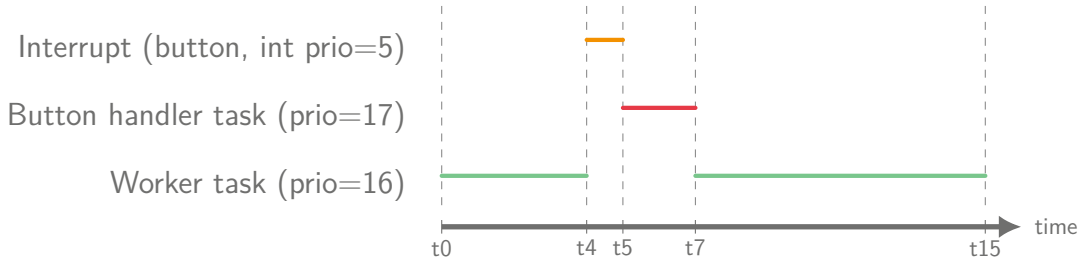
More details: `https://www.freertos.org/implementation/a00008.html`

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

FreeRTOS – task priority

FreeRTOS scheduling with task priorities



Interrupt (Button):

- Lifts (`V()`) the semaphore for the button handler task

Button handler task:

- Works for 2 time periods
- Then finishes work and waits (`P()`) on a semaphore

Worker task:

- Works for ever (endless loop)

More details: `https://www.freertos.org/implementation/a00008.html`

Scheduling:

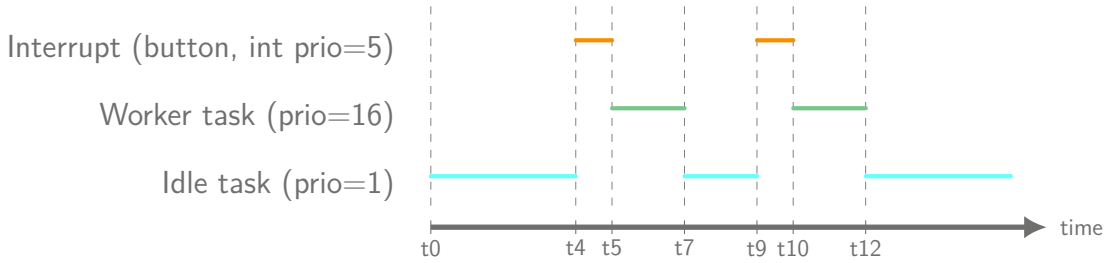
- `t0`: Scheduler: Worker task runs
- `t4`: Interrupt: ISR (lift (`V()`) semaphore)
- `t5`: Scheduler: Button handler task runs
- `t7`: Button handler task ends (waits on (`P()`) semaphore); Scheduler: Worker task
- ...

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

FreeRTOS – idle task

FreeRTOS scheduling with idle task



Interrupt (Button):

- Lifts (`V()`) the semaphore for the worker task

Worker task:

- Works for 2 time periods
- Then finishes work and waits (`P()`) on a semaphore

Idle task:

- Runs on the lowest priority (prio=1)
- Is executed, when no other task is ready to run

Scheduling:

- `t0`: Scheduler: Idle task runs
- `t4`: Interrupt: ISR (lift (`V()`) semaphore)
- `t5`: Scheduler: Worker task runs
- `t7`: Worker task ends (waits on (`P()`) semaphore); Scheduler: Idle task runs
- ...

More details: `https://www.freertos.org/implementation/a00008.html`

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Setup and start FreeRTOS kernel

Create project with FreeRTOS

- Create Arduino project with Arduino Mega 2560
- Add FreeRTOS library to project

Start kernel

```
1 //Start scheduler
2 vTaskStartScheduler();
3
4 //within the Arduino framework, this is done behind the scenes
5 //in variantHooks.cpp: initVariant()
```

setup() function

```
1 void setup() {
2   //here, the FreeRTOS primitives are created and initialised:
3   // - semaphores
4   // - tasks
5   // - ...
6 }
```

loop() function

```
1 void loop() {
2   //if the idle task functionality is activated, this function can act
3   //as the idle task (default)
4
5   //otherwise, this is kept empty because the FreeRTOS scheduler
6   //controls everything
7 }
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Create tasks

Define task handle

```
1  TaskHandle_t worker_task_handle = NULL;
```

Define task function

```
2  void worker_task(void* pvParameters) {
3      while(true){
4          //work...
5      }
6  }
7  //A task never ends. If it should be deleted,
8  //call vTaskDelete(nullptr) within the worker_task() function
```

Create task

```
9  BaseType_t result = xTaskCreate(
10   worker_task,          //address of task function
11   "worker_task",        //name of task
12   128L,                 //task stack size in words: 128*2=256 bytes (configSTACK_DEPTH_TYPE)
13   NULL,                 //task parameters
14   16,                   //task priority (2-55)
15   &worker_task_handle //pointer to task handle
16 );
17
18 configASSERT(result == pdPASS);           //assert that the task could be created
19 configASSERT(worker_task_handle != NULL); //assert that task handle is valid
```

- FreeRTOS doc (task creation): https://www.freertos.org/a00019.html

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Semaphores: interrupt to task

Define semaphore handle

```
1  SemaphoreHandle_t worker_semaphore = NULL;
```

Create semaphore

```
2  worker_semaphore = xSemaphoreCreateBinary();
3  configASSERT(worker_semaphore != NULL);
```

Worker task implementation

```
4  void worker_task(void* pvParameters) {
5    log_message("worker_task started...");
6
7    while(true) {
8      log_message("worker_task sleep...");
9      //wait infinitely for semaphore
10     xSemaphoreTake(worker_semaphore, portMAX_DELAY);
11     log_message("worker_task wake up...");
12
13     for (uint8_t i = 0; i < 10; ++i) {
14       log_message("worker_task: toggle led");
15       digitalWrite(LED_PIN, !digitalRead(LED_PIN));
16       vTaskDelay(pdMS_TO_TICKS(100));
17     }
18   }
19 }
```

ISR implementation

```
20 void button_isr() {
21   BaseType_t taskChangeRequired = pdFALSE;
22   xSemaphoreGiveFromISR(worker_semaphore,
23                         &taskChangeRequired);
24
25   //to immediately run the scheduler (change task)
26   if (taskChangeRequired == pdTRUE) {
27     taskYIELD();
28   }
29 }
```

Inside ISRs:

- Only use FreeRTOS functions that end with `FromISR()`
- Be aware to not use HW/HAL functions that are used from other tasks (needs to by synchronised)

- FreeRTOS doc (semaphores): `https://www.freertos.org/a00113.html`

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Semaphores: task to task

Example with two tasks: ping/pong example

Create semaphores

```
1  ping_semaphore = xSemaphoreCreateBinary();
2  configASSERT(ping_semaphore != NULL);
3
4  //let the ping task start first
5  xSemaphoreGive(ping_semaphore);
6
7  pong_semaphore = xSemaphoreCreateBinary();
8  configASSERT(pong_semaphore != NULL);
```

Task implementation

```
9  void ping_task(void* pvParameters) {
10   while (true) {
11     //wait infinitely for semaphore
12     xSemaphoreTake(ping_semaphore, portMAX_DELAY);
13
14     log_message("ping...");
15     vTaskDelay(pdMS_TO_TICKS(1000));
16
17     xSemaphoreGive(pong_semaphore);
18   }
19 }
```

```
20 void pong_task(void* pvParameters) {
21   while (true) {
22     //wait infinitely for semaphore
23     xSemaphoreTake(pong_semaphore, portMAX_DELAY);
24
25     log_message("       ...pong");
26     vTaskDelay(pdMS_TO_TICKS(1000));
27
28     xSemaphoreGive(ping_semaphore);
29   }
30 }
```

- FreeRTOS doc (semaphores): https://www.freertos.org/a00113.html

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Task notification: interrupt to task

Define task handle and task function as usual...

```
1  TaskHandle_t deferred_isr_task_handle = NULL;
```

Worker task implementation

```
2  void deferred_isr_task(void* pvParameters) {
3    log_message("worker_task started...");
4
5    while(true) {
6      log_message("worker_task sleep...");
7      //wait infinitely for a task notification
8      ulTaskNotifyTake(pdFALSE, portMAX_DELAY);
9      log_message("worker_task wake up...");
10
11     for (uint8_t i = 0; i < 10; ++i) {
12       log_message("worker_task: toggle led");
13       digitalWrite(LED_PIN, !digitalRead(LED_PIN));
14       vTaskDelay(pdMS_TO_TICKS(100));
15     }
16   }
17 }
```

ISR implementation

```
18 void button_isr() {
19   BaseType_t taskChangeRequired = pdFALSE;
20   vTaskNotifyGiveFromISR(deferred_isr_task_handle,
21                          &taskChangeRequired);
22
23   //to immediately run the scheduler (change task)
24   if (taskChangeRequired == pdTRUE) {
25     taskYIELD();
26   }
27 }
```

Inside ISRs:

- Only use FreeRTOS functions that end with `FromISR()`
- Be aware to not use HW/HAL functions that are used from other tasks (needs to by synchronised)

- FreeRTOS doc (task notifications): `https://www.freertos.org/RTOS-task-notification-API.html`
- FreeRTOS doc (deferred interrupt handling): `https://www.freertos.org/deferred_interrupt_processing.html`

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Software timers

Timers run in the context of the timer service task (with its own priority).

Only one callback is served at a time.

Define timer handle

```
1  TimerHandle_t timer = NULL;
2  int timerId = 1; //optional: id for timer
```

Define timer callback function

```
3  void timerCallback(TimerHandle_t xTimer) {
4    //work...
5  }
```

Create timer

```
6  timer = xTimerCreate(
7    "SW Timer",                 //name of timer
8    pdMS_TO_TICKS(1000),        //period in ticks
9    pdTRUE,                     //true=auto reload, false=one shot
10   &timerId,                   //address timer id variable
11   timerCallback               //address of timer callback function
12 );
13
14 configASSERT(timer != NULL); //assert that timer handle is valid
```

Start timer

```
15 xTimerStart(timer, 100);      //starts the timer in 100 ticks
```

Timer callbacks are not allowed to block: Therefore don't call `vTaskDelay()`, `vTaskDelayUntil`, or specify a non zero block time to access queues or semaphores.

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Summary and outlook

Summary

- FreeRTOS intro
- FreeRTOS customisation
- FreeRTOS task scheduling
- FreeRTOS task coding

Outlook

- ROS2 on Turtlebot3