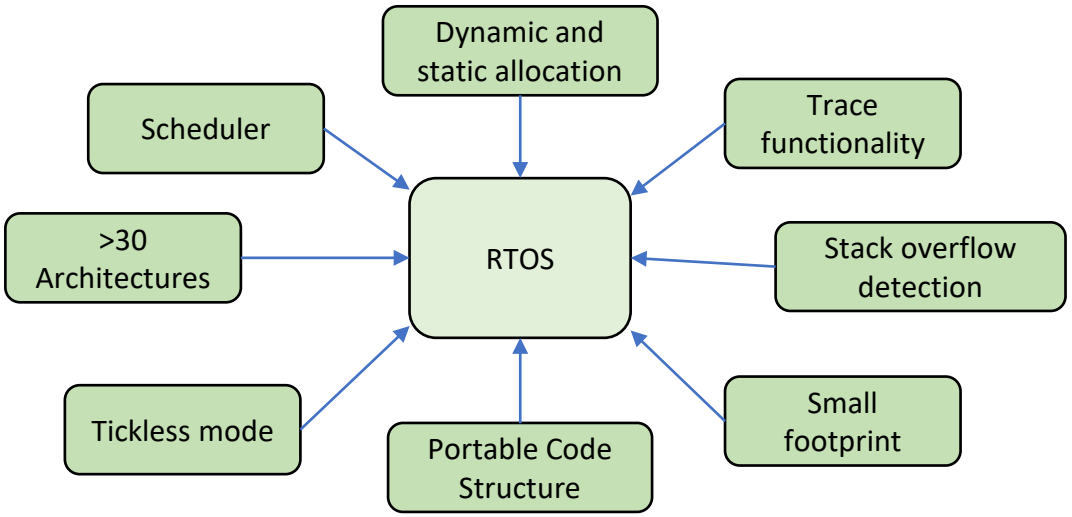
Microcontroller Engineering
TMIK13
Lecture 14

OPERATING SYSTEMS ON MICROCONTROLLERS

ANDREAS AXELSSON (ANDREAS.AXELSSON@JU.SE)

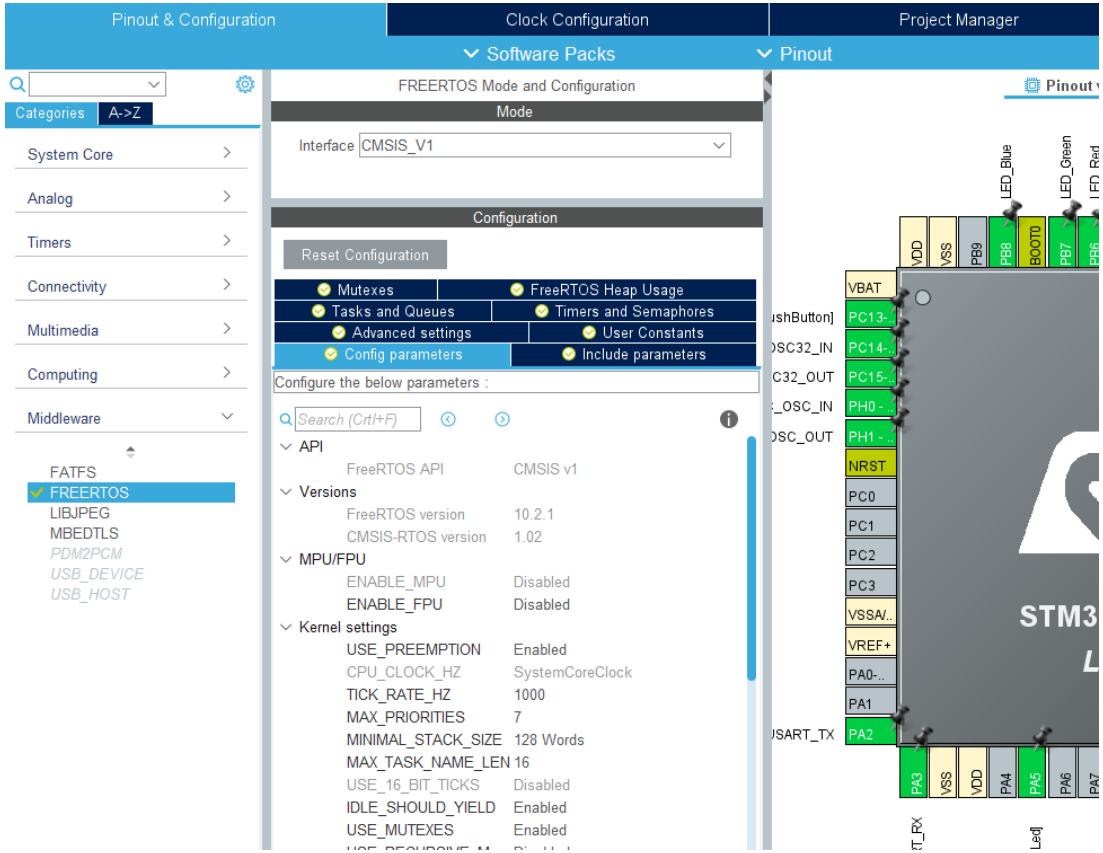
Real-time Operating Systems

www.freertos.org

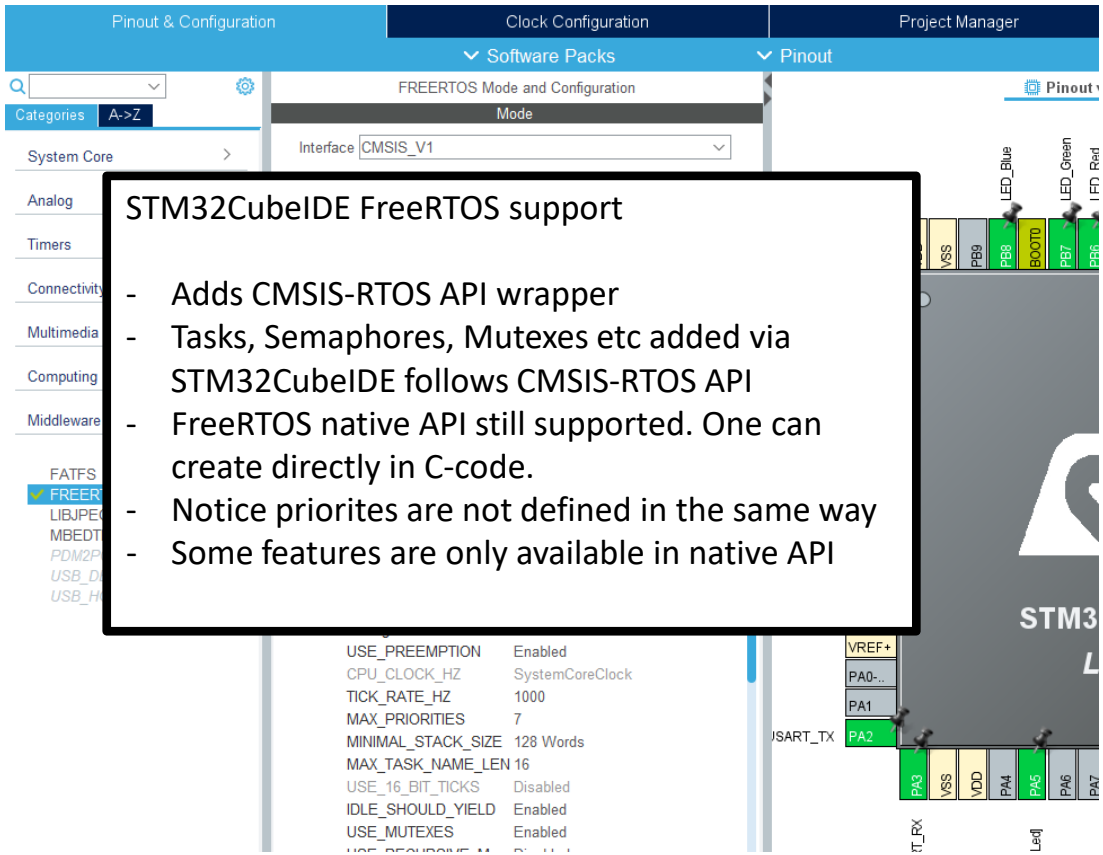


https://www.beningo.com

FreeRTOS in STM32CubeIDE



FreeRTOS in STM32CubeIDE



STM32CubeIDE FreeRTOS support

- Adds CMSIS-RTOS API wrapper
- Tasks, Semaphores, Mutexes etc added via STM32CubeIDE follows CMSIS-RTOS API
- FreeRTOS native API still supported. One can create directly in C-code.
- Notice priorites are not defined in the same way
- Some features are only available in native API

RTOS Task – FreeRTOS

Creating a task in FreeRTOS

```
xTaskCreate(
      LED_Green_Blink,                        /* Task Pointer */
      (const char* const)"TaskGreenLED",      /* Task Name */
      configMINIMAL_STACK_SIZE,               /* Stack Depth */
      NULL,                                   /* Parameters to pass to task */
      1,                                      /* Task Priority */
      NULL);                                  /* Pass handle to created task */
```

Task function

```
void LED_Green_BLink(void const * argument)
{
  /* USER CODE BEGIN LED_Green_BLink */
    const TickType_t xDelay = 23;
    uint32_t GreenDelay = 0;
    const uint32_t TargetCount = 200000; //This is the delaytime for the
    //blocking delay making this task take extra CPU-time

  /* Infinite loop */
  for(;;)
  {
      HAL_GPIO_WritePin(LED_Green_GPIO_Port, LED_Green_Pin, GPIO_PIN_RESET);
      DelayNonsense(&GreenDelay, &TargetCount);  //Keep the CPU busy
      vTaskDelay (xDelay);
      HAL_GPIO_WritePin(LED_Green_GPIO_Port, LED_Green_Pin, GPIO_PIN_SET);
      DelayNonsense(&GreenDelay, &TargetCount);  //Keep the CPU busy
      vTaskDelay (xDelay);
  }
  /* USER CODE END LED_Green_BLink */
}
```

FreeRTOS - API

APIs categories	API
Task creation	– xTaskCreate – vTaskDelete
Task control	– vTaskDelay – vTaskDelayUntil – uxTaskPriorityGet – vTaskPrioritySet – vTaskSuspend – vTaskResume – xTaskResumeFromISR – vTaskSetApplicationTag – xTaskCallApplicationTaskHook
Task utilities	– xTaskGetCurrentTaskHandle – xTaskGetSchedulerState – uxTaskGetNumberOfTasks – vTaskList – vTaskStartTrace – ulTaskEndTrace – vTaskGetRunTimeStats
Kernel control	– vTaskStartScheduler – vTaskEndScheduler – vTaskSuspendAll – xTaskResumeAll

https://www.st.com/resource/en/user_manual/dm00105262-developing-applications-on-stm32cube-with-rtos-stmicroelectronics.pdf

FreeRTOS - API

APIs categories	API
Queue management	– xQueueCreate – xQueueSend – xQueueReceive – xQueuePeek – xQueueSendFromISR – xQueueSendToBackFromISR – xQueueSendToFrontFromISR – xQueueReceiveFromISR – vQueueAddToRegistry – vQueueUnregisterQueue
Semaphores	– vSemaphoreCreateBinary – vSemaphoreCreateCounting – xSemaphoreCreateMutex – xSemaphoreTake – xSemaphoreGive – xSemaphoreGiveFromISR

https://www.st.com/resource/en/user_manual/dm00105262-developing-applications-on-stm32cube-with-rtos-stmicroelectronics.pdf

CMSIS-RTOS API

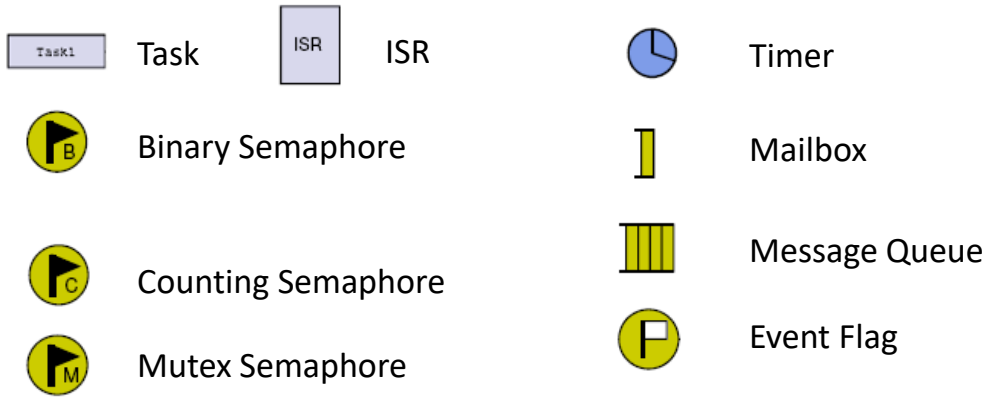
Module	API	Description
Message queue management[1]: Control, send, receive, or wait for messages.	osMessageCreate	Define and initialize a message queue.
	osMessagePut	Put a message into a message queue.
	osMessageGet	Get a message or suspend thread execution until message arrives
Mail queue management[1]: Control, send, receive, or wait for mail.	osMailCreate	Define and initialize a mail queue with fix-size memory blocks
	osMailAlloc	Allocate a memory block
	osMailCAlloc	Allocate a memory block and zero-set this block
	osMailPut	Put a memory block into a mail queue
	osMailGet	Get a mail or suspend thread until mail arrives.
	osMailFree	Return a memory block to the mail queue.

1. The modules or APIs marked with (*) are optional.

Module	API	Description
Thread management: Define, create and control thread functions	osThreadCreate	Start execution of a thread function.
	osThreadTerminate	Stop execution of a thread function.
	osThreadYield	Pass execution to next ready thread function.
	osThreadGetId	Get the thread identifier to reference this thread.
	osThreadSetPriority	Change the execution priority of a thread function.
	osThreadGetPriority	Obtain the current execution priority of a thread function.
Generic wait functions: Wait for a time period or unspecified events.	osDelay	Wait for a specified time.
	osWait (*)	Wait for any event of the type Signal, Message, or Mail.
Timer management[1]: Create and control timer and timer callback functions.	osTimerCreate	Define attributes of the timer callback function
	osTimerStart	Start or restart the timer with a time value.
Signal management: Control or wait for signal flags.	osSignalSet	Set signal flags of a thread.
	osSignalClear	Reset signal flags of a thread.
	osSignalClear	Suspend execution until specific signal flags are set.
Mutex management[1]: Synchronize thread execution with a Mutex.	osMutexCreate	Define and initialize a mutex
	osMutexWait	Obtain a mutex or Wait until it becomes available.
	osMutexRelease	Release a mutex
	osMutexDelete	Delete a mutex
Semaphore management[1]: Control access to shared resources.	osSemaphoreCreate	Define and initialize a semaphore.
	osSemaphoreWait	Obtain a semaphore token or Wait until it becomes available.
	osSemaphoreRelease	Release a semaphore token.
	osSemaphoreDelete	Delete a semaphore.
Memory pool management[1]: Define and manage fixed-size memory pools.	osPoolCreate	Define and initialize a fix-size memory pool.
	osPoolAlloc	Allocate a memory block.
	osPoolCAlloc	Allocate a memory block and zero-set this block.
	osPoolFree	Return a memory block to the memory pool.

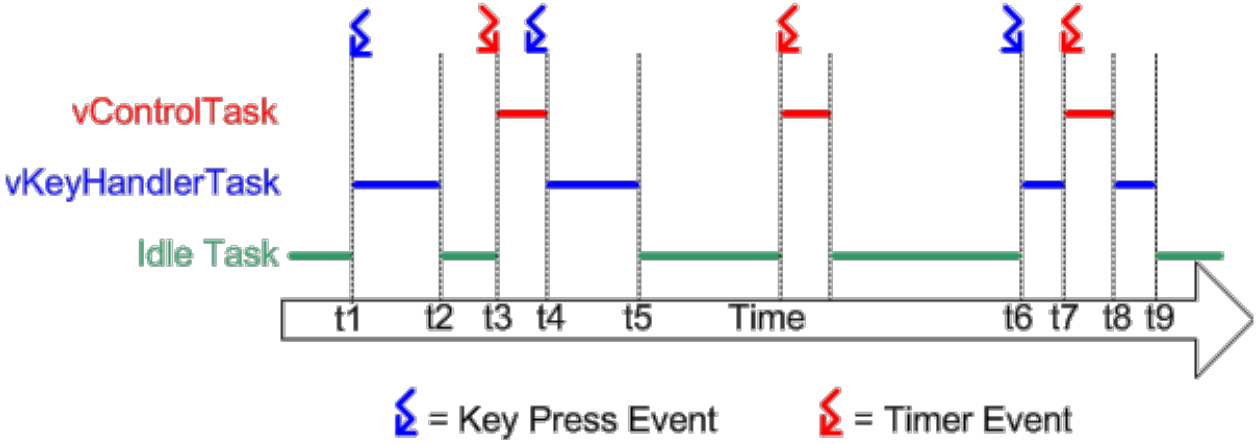
https://www.st.com/resource/en/user_manual/dm00105262-developing-applications-on-stm32cube-with-rtos-stmicroelectronics.pdf

RTOS – Primitives



Courtesy Willian Sandqvist

RTOS – Tasks



TCB

Task Name

Priority

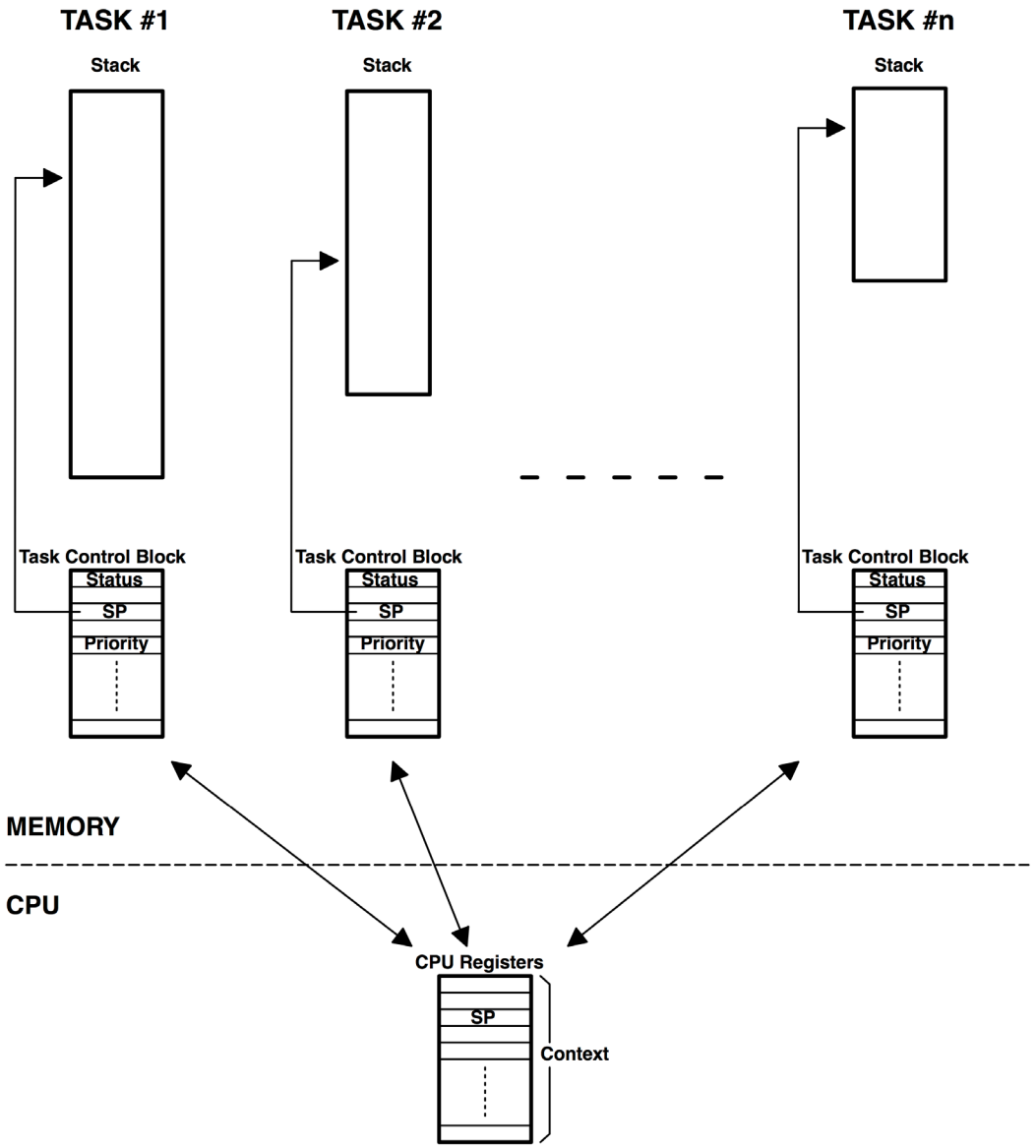
–

Task Code

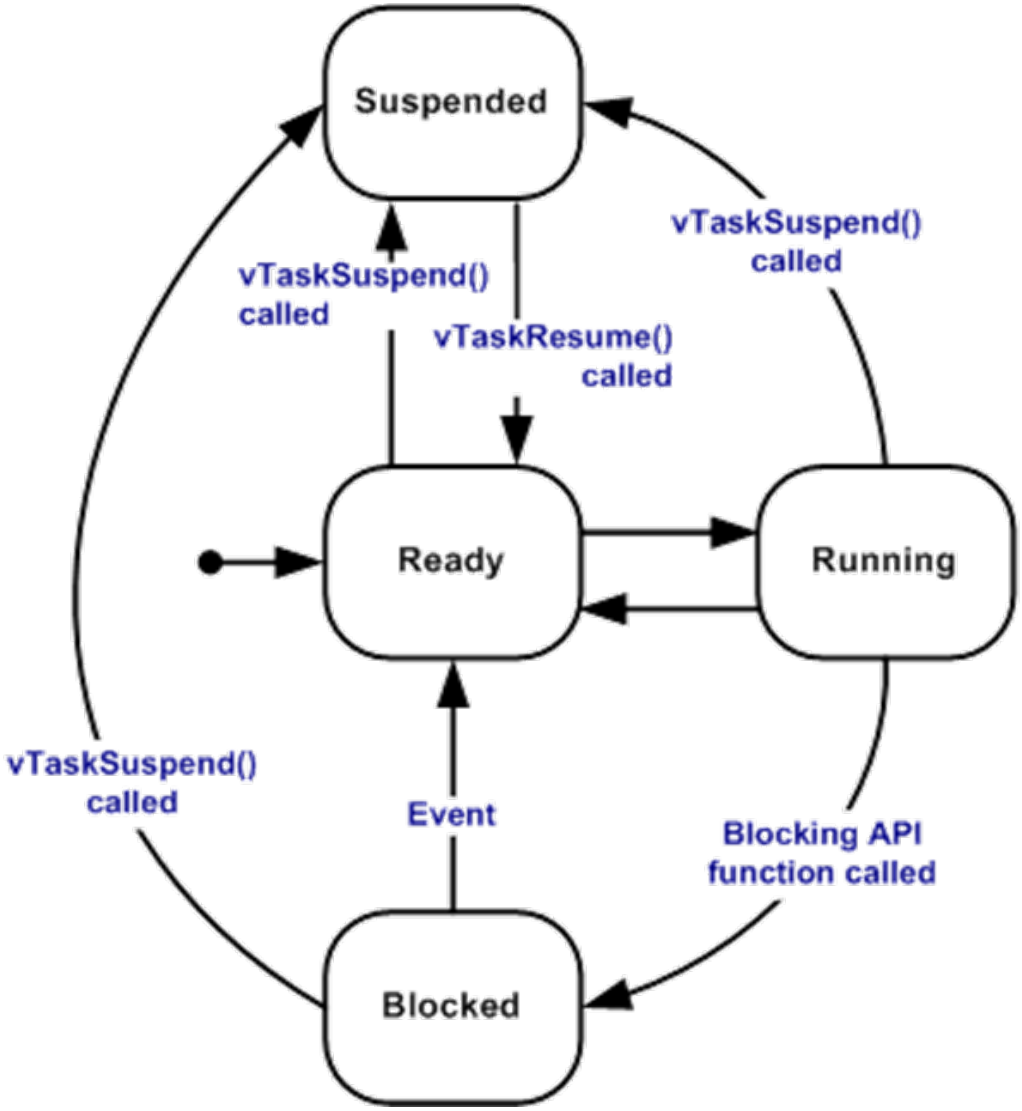
–

Stack

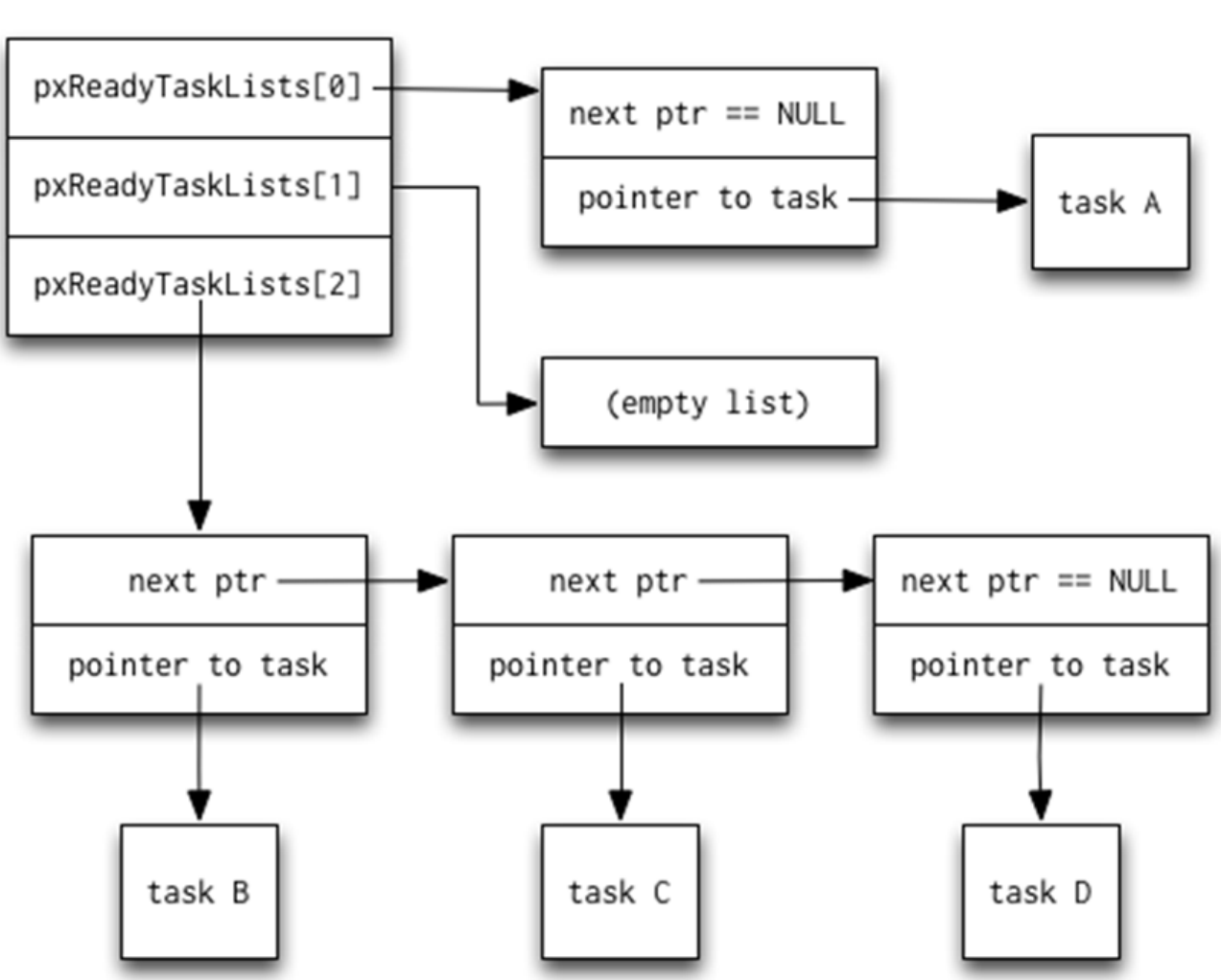
TCB – Context Switching



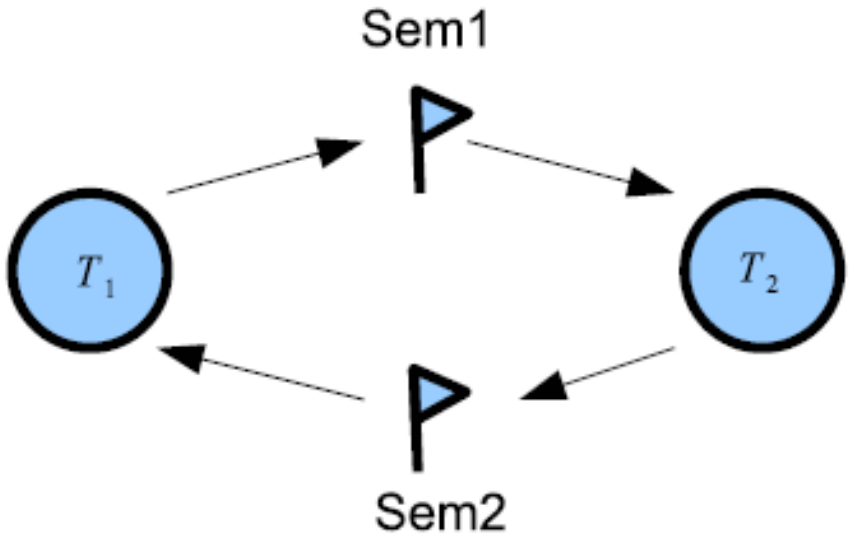
Software Concurrency – RTOS



Software Concurrency – RTOS

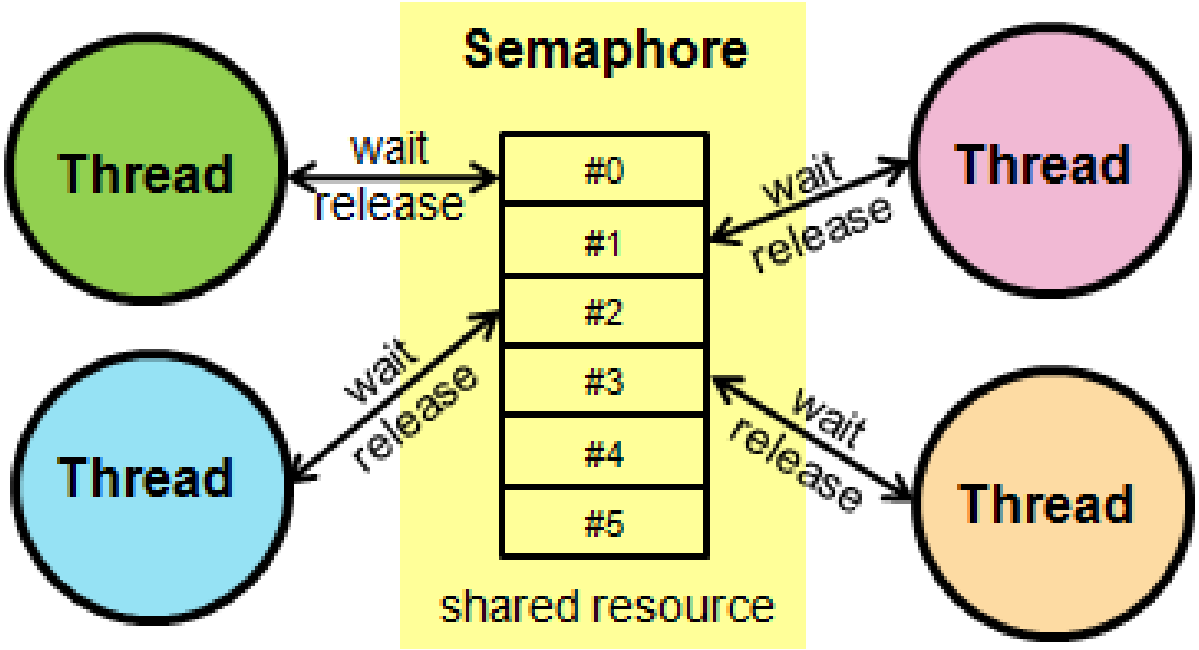


RTOS – Semaphores

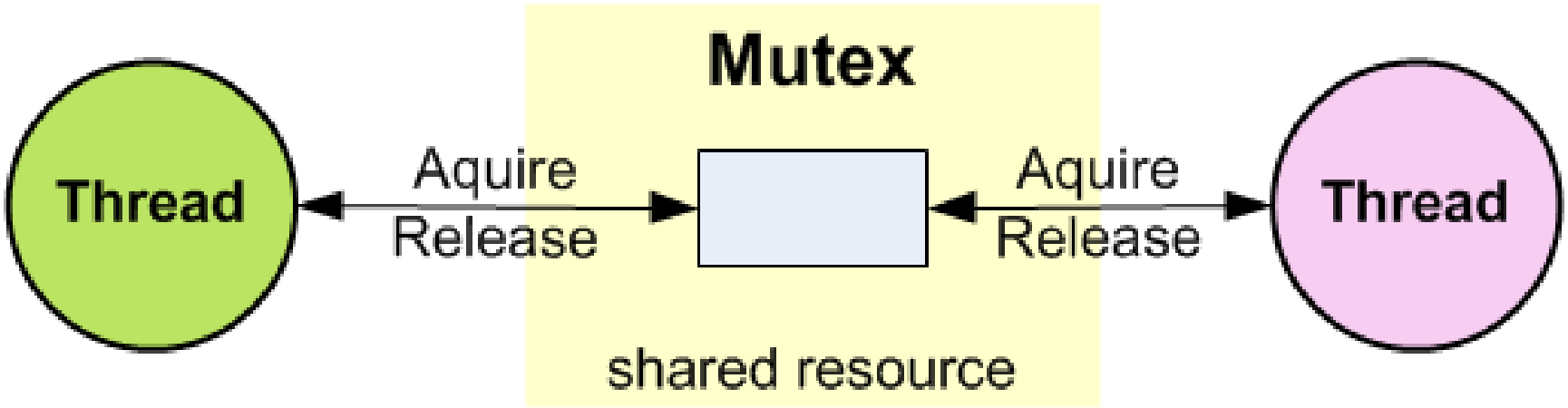


Courtesy Willian Sandqvist

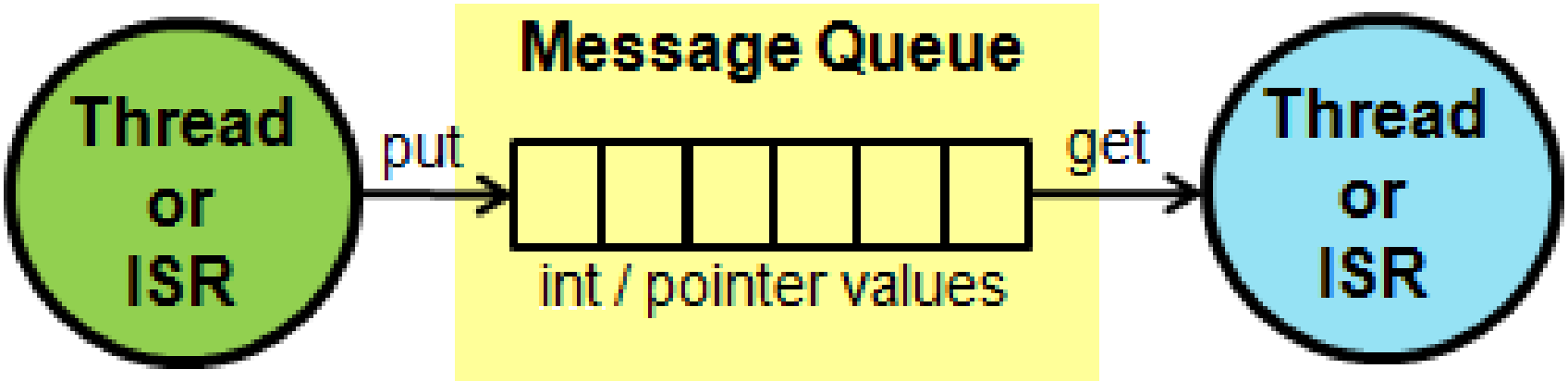
RTOS – Counting Semaphores



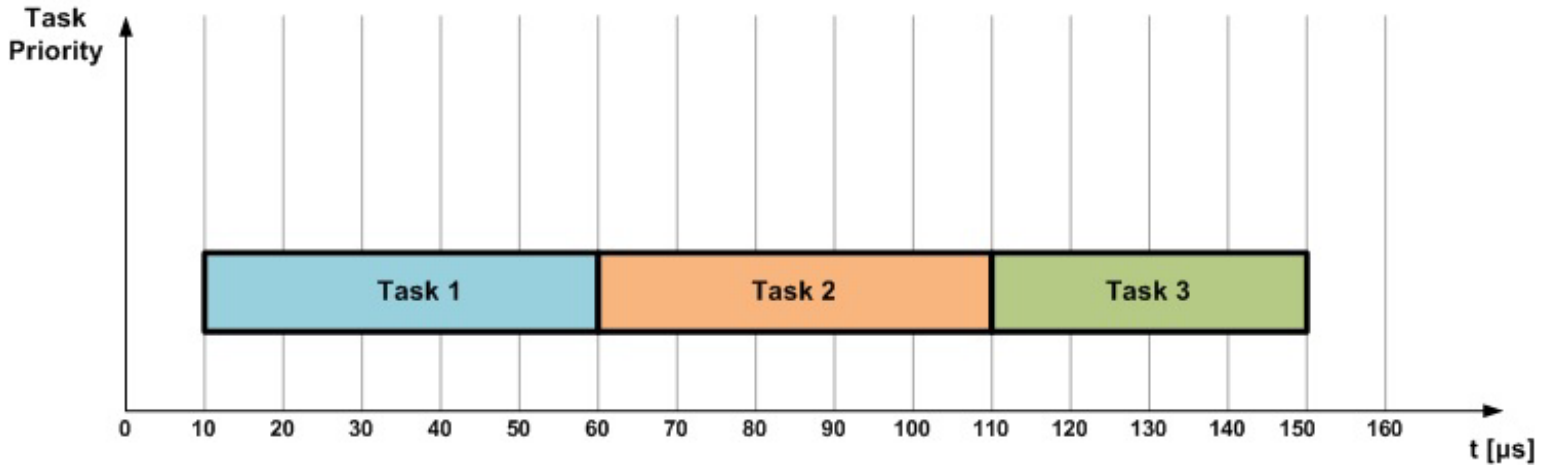
RTOS – Mutex


Mutex
Thread
Aquire
Release
Aquire
Release
Thread
shared resource

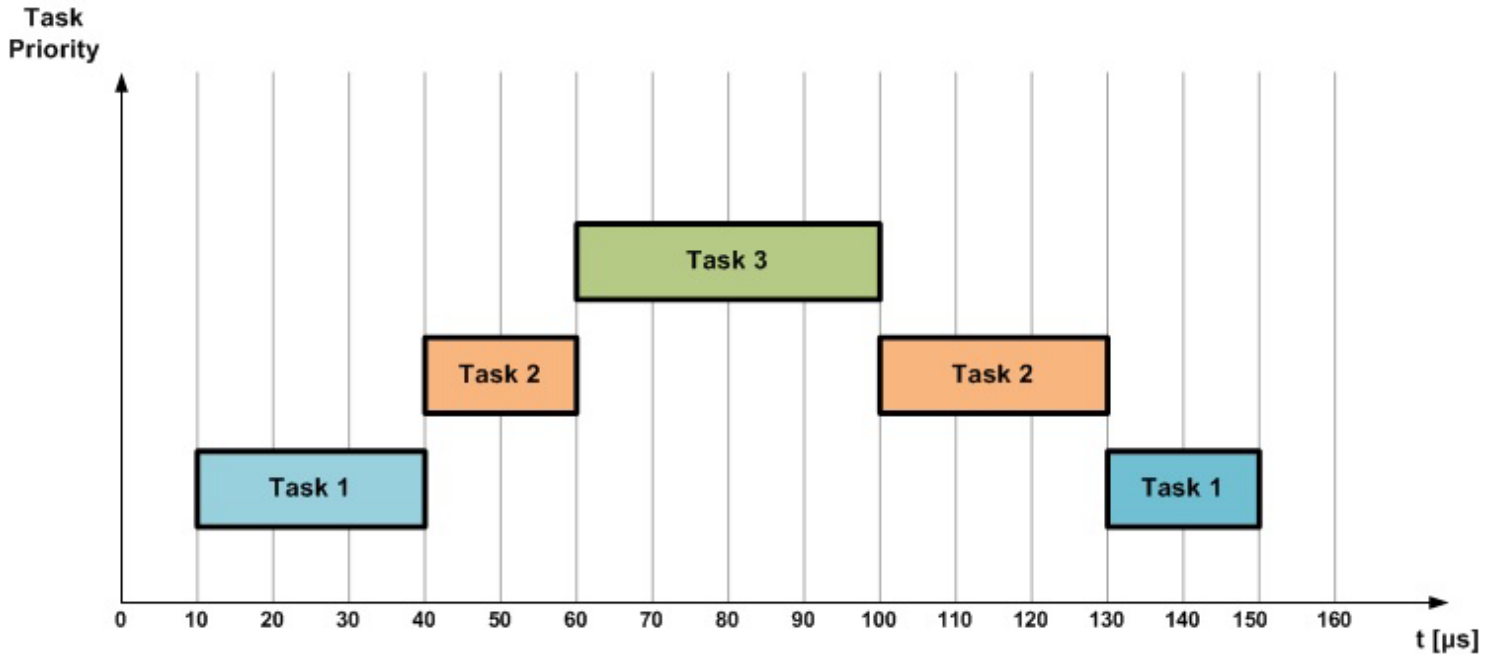
RTOS – Message Queue



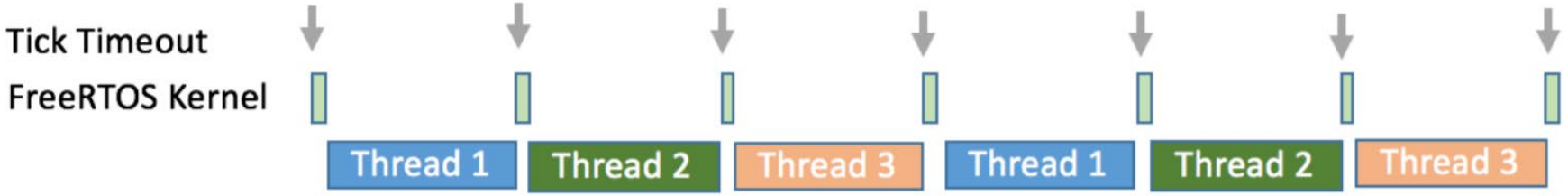
Scheduling – Non-preemptive



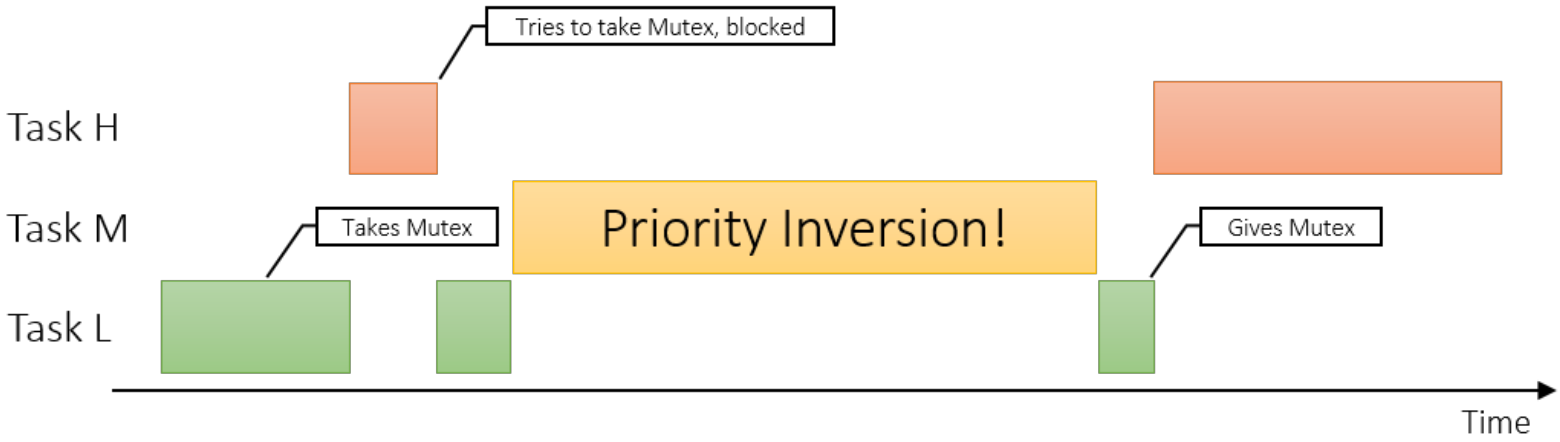
Scheduling – Preemptive



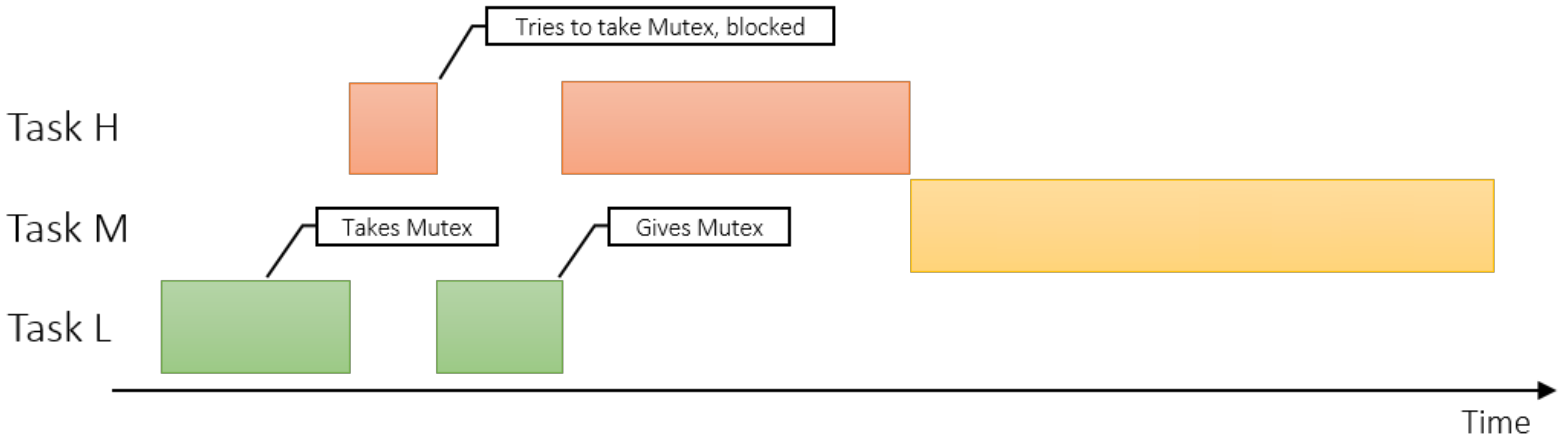
Scheduling – Time-Sharing (RR)



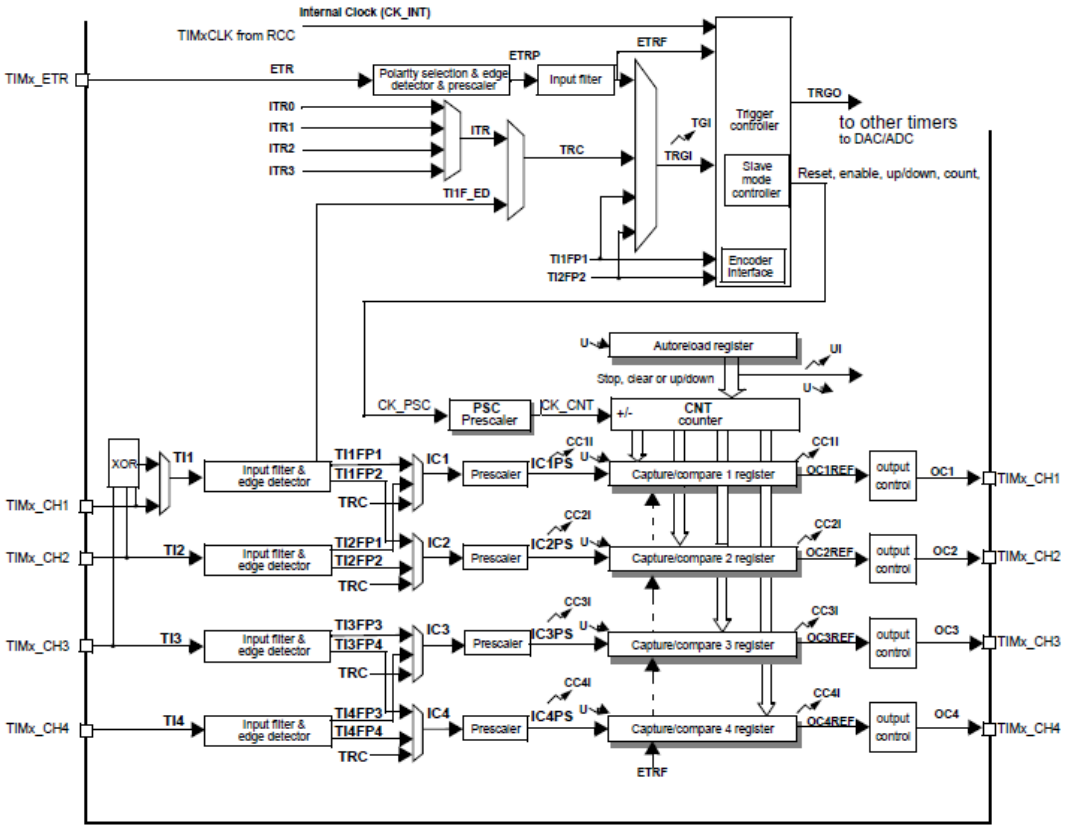
Priority Inversion



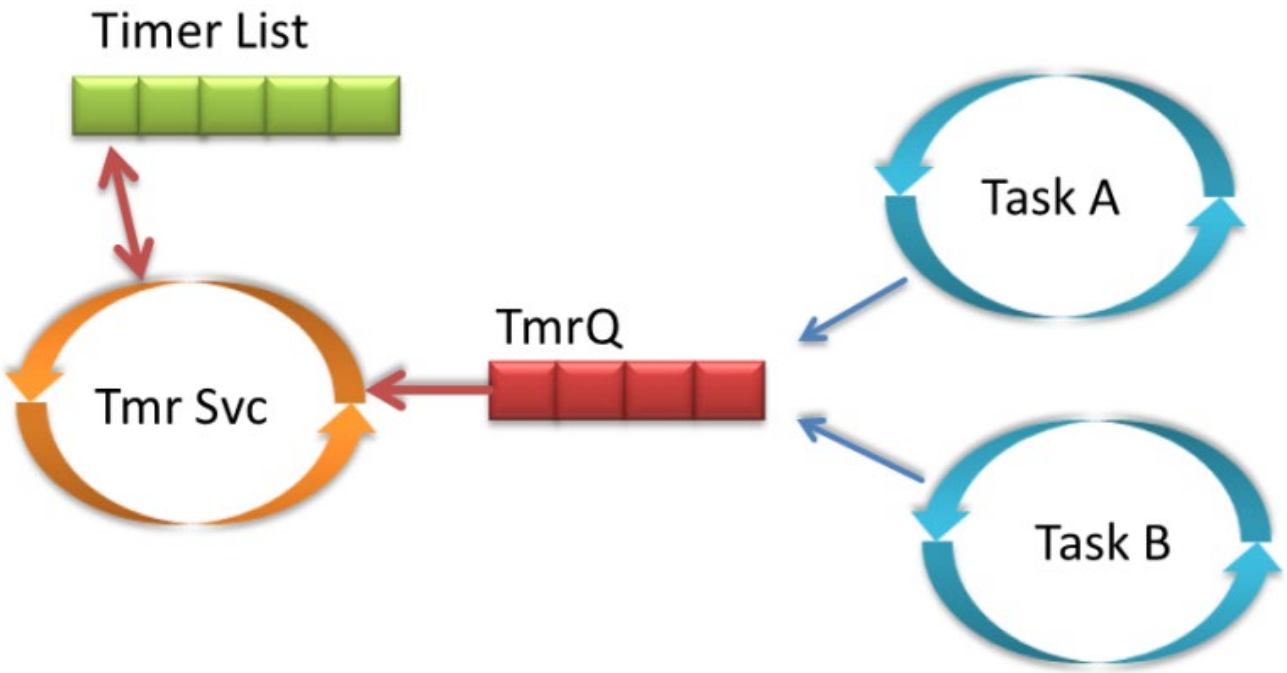
Priority Inheritance



Timers – Hardware



Timers – Software (RTOS)



Timers – Software (RTOS)

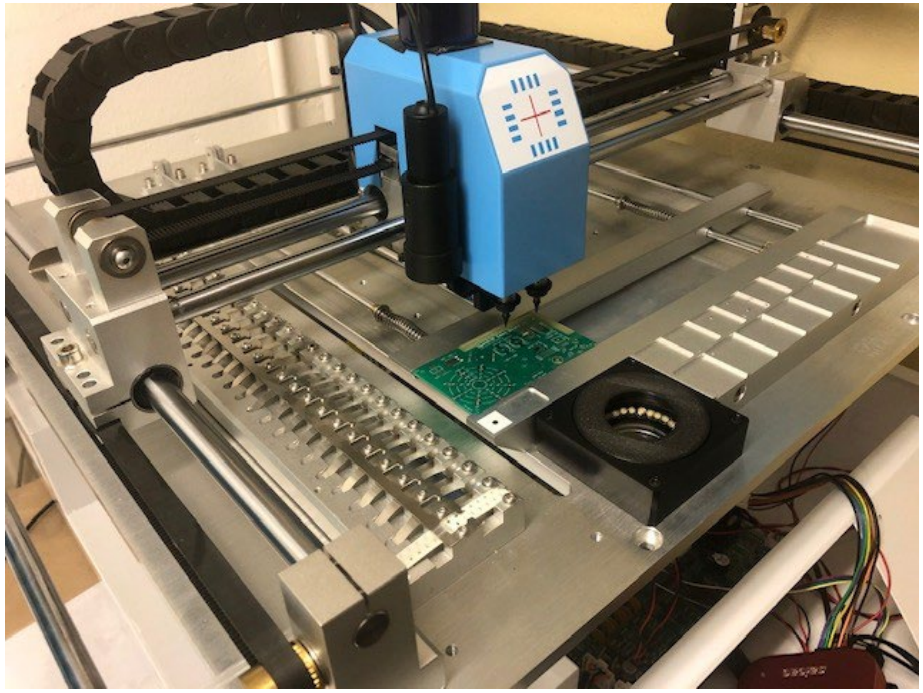
```
timerHndl1Sec = xTimerCreate(
        "timer1Sec",                    /* name */
        pdMS_TO_TICKS(1000),            /* period/time */
        pdTRUE,                         /* auto reload */
        (void*) 0,                      /* timer ID */
        vTimerCallback1SecExpired); /* callback */
```

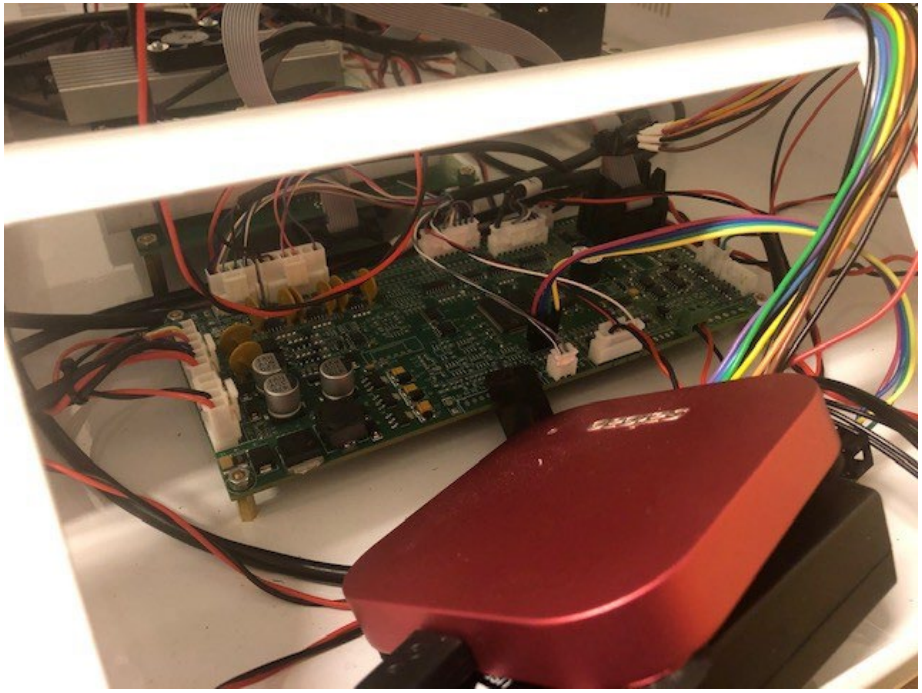
```
static void vTimerCallback1SecExpired(xTimerHandle pxTimer) {
    HAL_GPIO_TogglePin(LED_Green_GPIO_Port, LED_Green_Pin);
}
```

Timers – Software (RTOS)

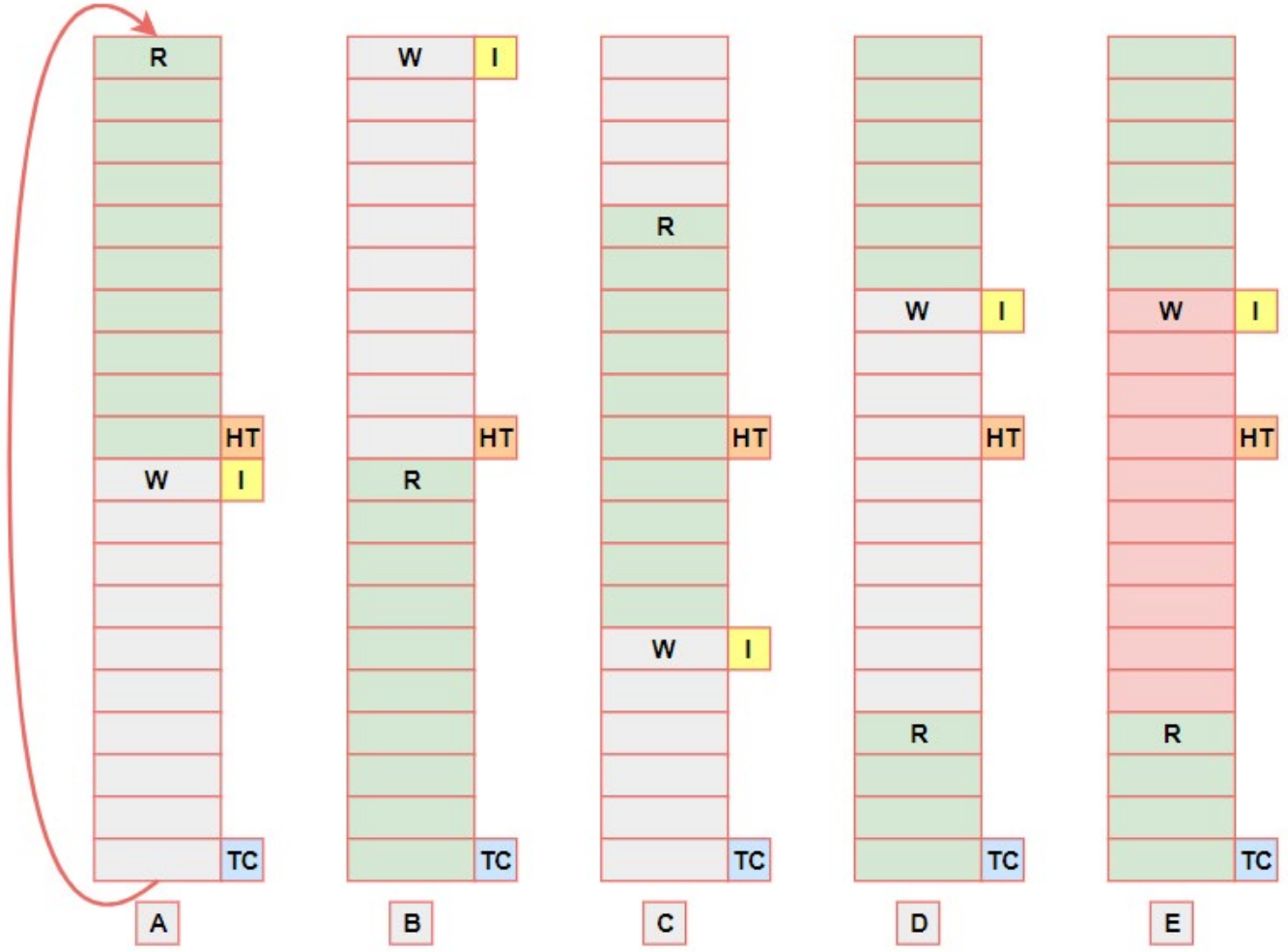
Naturally, the shortest period time you can reach with a FreeRTOS software timer is a single tick period. So if your FreeRTOS is running with a 1 kHz tick period, you only can implement a 1 kHz software timer that way. If it needs to be faster, you have to consider using a hardware timer.

Pick and Place Machine





UART Example – Event flags



UART Example – Event flags

```
void StartUartRxTask(void const * argument)
{
    __HAL_UART_ENABLE_IT(&huart1, UART_IT_IDLE);       // enable idle line interrupt
    __HAL_DMA_ENABLE_IT (&hdma_usart1_rx, DMA_IT_TC);  // enable DMA Tx cplt interrupt
    __HAL_DMA_ENABLE_IT (&hdma_usart1_rx, DMA_IT_HT);  // enable DMA Tx half cplt interrupt

    HAL_UART_Receive_DMA(&huart1, (uint8_t*)usart_rx_dma_buffer, USART_RX_DMA_BUFFER_LEN);

    osEvent event;
    /* Infinite loop */
    for(;;)
    {
        event = osSignalWait (0xffff, osWaitForever);
        if (event.value.signals & 0x0007)
        {
            usart_rx_check();
        }
    }
}
```

```
#define NV_USART_DMA_XFER_HALFCPLT 0x0001
#define NV_USART_DMA_XFER_CPLT     0x0002
#define NV_USART_IDLE_IRQ          0x0004
```

UART Example – Event flags

```
void HAL_UART_RxHalfCpltCallback(UART_HandleTypeDef *huart)
{
    if(USART1 == huart->Instance)
    {
        osSignalSet(uartRxTaskHandle, NV_USART_DMA_XFER_HALFCPLT);
    }
}

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if(USART1 == huart->Instance)
    {
        osSignalSet(uartRxTaskHandle, NV_USART_DMA_XFER_CPLT);
    }
}

void USER_UART_IRQHandler(UART_HandleTypeDef *huart)
{
    if(USART1 == huart->Instance)
    {
        if(RESET != __HAL_UART_GET_FLAG(&huart1, UART_FLAG_IDLE))
        {
            __HAL_UART_CLEAR_IDLEFLAG(&huart1);
            osSignalSet(uartRxTaskHandle, NV_USART_IDLE_IRQ);
        }
    }
}
```

STM32CubeIDE Demo

Lets go!!!

Microcontroller Engineering

Questions?

Contact information

Andreas Axelsson

Email: andreas.axelsson@ju.se

Mobile: 0709-467760