FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Interprocess Communication

- Interprocess communication is a set of **mechanisms** that are provided by the **operating system** to allow different processes to manage shared data and resources
- Examples
  - Semaphore
  - Message Queue
  - Stream Buffers

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Task Synchronization
Semaphore

- Semaphore is a like a token
- A task can acquire semaphore
- Only the task with a semaphore can access a resource
- Otherwise, the task remains in the block state and can not execute
- As soon as the token is released it could be acquired by the other task

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Semaphore
Free-RTOS Semaphore-API

- xSemaphoreCreateBinary()
- xSemaphoreCreateMutex()
  - Like binary semaphore but includes priority inheritance mechanism
- xSemaphoreTake()
- xSemaphoreGive()
- xSemaphoreCreateCounting()
- vSemaphoreDelete()
- uxSemaphoreGetCount()

Semaphore
Example Part1

```
…
SemaphoreHandle_t xSemaphore = NULL;
…
void main() {
   …
   xSemaphore = xSemaphoreCreateBinary();
   xTaskCreate( vTask1, "Task1", 100, NULL, 1, NULL );
   xTaskCreate( vTask2, "Task2", 100, NULL, 1, NULL );
   xSemaphoreGive( xSemaphore );
   vTaskStartScheduler();
}
```

Semaphore
Example Task1

```
void vTask1( void * pvParameters ) {
  while(1) {
    if( xSemaphore != NULL ) {
      if( xSemaphoreTake( xSemaphore, ( TickType_t ) 0 ) == pdPASS ) {
        for (uint8_t i = 0; i < 5; i++) {
          snprintf(cbuffer,30,“Task1:i=%d\n\r",i);
          Serial.printfln(cbuffer);
          vTaskDelay(100);
        }
        xSemaphoreGive( xSemaphore );
        vTaskDelay(1);
      }
    }
  }
}
```

Semaphore
Example Task2

```
void vTask2( void * pvParameters ) {
  while(1) {
    if( xSemaphore != NULL ) {
      if( xSemaphoreTake( xSemaphore, ( TickType_t ) 0 ) == pdPASS ) {
        for (uint8_t j = 9; j >=0; j--) {
          snprintf(cbuffer,30,“Task2:j=%d\n\r",j);
          Serial.printfln(cbuffer);
          vTaskDelay(50);
        }
        xSemaphoreGive( xSemaphore );
        vTaskDelay(1);
      }
    }
  }
}
```

Semaphore
Example Output

```
Task 1 : i = 0
Task 1 : i = 1
Task 1 : i = 2
Task 1 : i = 3
Task 1 : i = 4
Task 2 : j = 9
Task 2 : j = 8
Task 2 : j = 7
Task 2 : j = 6
Task 2 : j = 5
Task 2 : j = 4
Task 2 : j = 3
Task 2 : j = 2
Task 2 : j = 1
Task 2 : j = 0
Task 1 : i = 0
Task 1 : i = 1
```

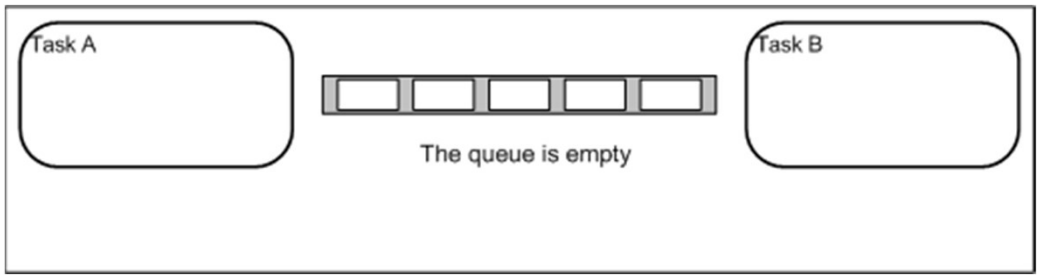
FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Interprocess Communication
Message Queue

- Message queue can be used to send data from one task to another
  - Messages are stored until they are read by a task
  - Message queue preserve the message order
    - New message can be inserted to the front (xQueueSendToFront) or to the back of the queue (freeRTOS: xQueueSendToBack)
- Each Queue in FreeRTOS:
  - Consists of a predefined number of message entries
  - All messages have same predefined size
    - Can be a variable, array, structure, …

Interprocess Communication
Message Queue

- Advantage over other interprocess communication mechanisms:
  - Messages are queued in order
  - Messages not lost



https://www.freertos.org/Embedded-RTOS-Queues.html

Interprocess Communication
Message Queue

- Creating a message
  - QueueHandle_t xQueueCreate(UBaseType_t uxQueueLength,
    UBaseType_t uxItemSize);
  - QueueHandle_t xQueueCreateStatic(UBaseType_t uxQueueLength,
    UBaseType_t uxItemSize,
    uint8_t *pucQueueStorageBuffer,
    StaticQueue_t *pxQueueBuffer);

Message Queue
Sending Item to a Queue

- Sending Item to the back of a queue
  - BaseType_t **xQueueSendToBack**(QueueHandle_t xQueue,
    const void * pvItemToQueue,
    TickType_t xTicksToWait);
  - BaseType_t **xQueueSendToBackFromISR**(…);
- Sending Item to the front of a queue
  - BaseType_t **xQueueSendToFront**(…);
  - BaseType_t **xQueueSendToFrontFromISR**(…);

Message Queue
Sending Item to the Queue

- **Overwriting** the last item in a Queue
  - BaseType_t **xQueueOverwrite**(QueueHandle_t xQueue,
    const void * pvItemToQueue,
    BaseType_t *pxHigherPriorityTaskWoken);
  - BaseType_t **xQueueOverwriteFromISR** (QueueHandle_t xQueue,
    const void * pvItemToQueue,
    BaseType_t *pxHigherPriorityTaskWoken);

Message Queue
Receiving Item from Queue

- Receiving Item from a Queue
  - BaseType_t **xQueueReceive(QueueHandle_t xQueue, void *pvBuffer, TickType_t xTicksToWait);**
  - BaseType_t **xQueueReceiveFromISR**(…)

- Receiving Item from the Queue **without consuming** it
  - BaseType_t **xQxQueuePeek**(…)
  - BaseType_t **xQueuePeekFromISR**(…)

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Message Queue
Other Functions

- BaseType_t **xQueueIsQueueFullFromISR**(const QueueHandle_t xQueue);
- BaseType_t **xQueueIsQueueEmptyFromISR**(const QueueHandle_t xQueue);
- BaseType_t **xQueueReset**(QueueHandle_t xQueue);
- void **vQueueDelete**(QueueHandle_t xQueue);