

\*\*\*\*\*Draft\*\*\*\*\*

Part 4 pico  
freeRTOS  
Mutex  
02/22/22

\*\*\*\*\*Draft\*\*\*\*\*

<https://learnembeddedsystems.co.uk/freertos-on-the-rp2040-part-4-source-code>

image1

## ***What are we going to cover?***

- What a Mutex is
- Why should you use a mutex?
- When shouldn't you use a mutex?
- How to implement a mutex

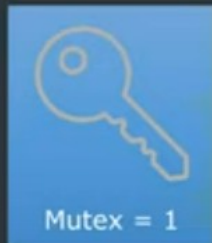
image2

## ***What is a Mutex?***

- Mutex = **Mutual Exclusion**
- Binary semaphore that includes a priority inheritance mechanism
- A mutex acts as a token that is used to guard a resource

image3

# What is a Mutex?



Task A

Task B



Guarded Resource

Task needs to gain access to Guarded Resource

# What is a Mutex?



"Take Mutex"

Task A

Task B



Guarded Resource

Task A uses the Guarded Resource

# What is a Mutex?



Task A

Task B



Guarded Resource

When Task A is finished with Guarded Resource returns the key.



## Why use a mutex?

To protect against two or more tasks modifying data ( a resource) simultaneously.

Mutex is a token that is used to guard a resource

## When not to use a mutex?

Within an interrupt because:

They include a priority inheritance mechanism which only makes sense if the mutex is given and taken from a task, not an interrupt.

An interrupt cannot block to wait for a resource that is guarded by a mutex to become available.

Line 21 FreeRTOSConfig.h

```
#define configUSE_MUTEXES 1
```

Line 26 FreeRTOSConfig.h

```
#define configUSE_TIME_SLICING 1
```

Line 13 FreeRTOSConfig.h

```
#define configTICK_RATE_HZ 100000
```

```

File Edit Tabs Help
diff --git a/freertos/FreeRTOSConfig.h b/freertos/FreeRTOSConfig.h
index 4f894fb..ede1b1f 100644
--- a/freertos/FreeRTOSConfig.h
+++ b/freertos/FreeRTOSConfig.h
@@ -10,7 +10,7 @@
#define configUSE_PORT_OPTIMISED_TASK_SELECTION 0
#define configUSE_TICKLESS_IDLE 0
#define configCPU_CLOCK_HZ 133000000
-#define configTICK_RATE_HZ 100
+#define configTICK_RATE_HZ 100000^M
#define configMAX_PRIORITIES 5
#define configMINIMAL_STACK_SIZE 128
#define configMAX_TASK_NAME_LEN 16
@@ -18,12 +18,12 @@
#define configIDLE_SHOULD_YIELD 1
#define configUSE_TASK_NOTIFICATIONS 1
#define configTASK_NOTIFICATION_ARRAY_ENTRIES 3
-#define configUSE_MUTEXES 0
+#define configUSE_MUTEXES 1^M
#define configUSE_RECURSIVE_MUTEXES 0
#define configUSE_COUNTING_SEMAPHORES 0
#define configQUEUE_REGISTRY_SIZE 10
#define configUSE_QUEUE_SETS 0
:

```

~/rp2040-freertos-project/build \$

cmake ..

make

openocd -f interface/raspberrypi-swd.cfg -f target/rp2040.cfg -c "program Mutex/Mutex.elf verify  
reset exit"

sudo minicom -s

Code without Mutex

```
File Edit Tabs Help
1112211
11
211
1112212212222
121121122
11
21112
1212
21112121122222212
11111222
1
2112222222
1122111211221221
111222
122
12111
122221212221212212211121122121112
1222122111
11122222112211111222111221122
221211
21121111211212111121222222222
11112211211111112112222222112
12211
```

## Code with Mutex

```
File Edit Tabs Help
1111111111
2222222222
2222222222
2222222222
2222222222
2222222222
2222222222
2222222222
2222222222
1111111111
2222222222
2222222222
2222222222
1111111111
1111111111
2222222222
1111111111
2222222222
2222222222
1111111111
2222222222
1111111111
1111111111
1111111
```

## Code without mutex

```
#include <FreeRTOS.h>
```

```

#include <task.h>
#include <stdio.h>
#include "pico/stdlib.h"

void task1(void *pvParameters)
{
    char ch = '1';
    while (true) {
        for(int i = 1; i < 10; i++){
            putchar(ch);
        }
        puts("");
    }
}

void task2(void *pvParameters)
{
    char ch = '2';
    while (true) {
        for(int i = 1; i < 10; i++){
            putchar(ch);
        }
        puts("");
    }
}

int main()
{
    stdio_init_all();

    xTaskCreate(task1, "Task 1", 256, NULL, 1, NULL);
    xTaskCreate(task2, "Task 2", 256, NULL, 1, NULL);
    vTaskStartScheduler();

    while(1){};
}

```

## Code with mutex

```

#include <FreeRTOS.h>
#include <task.h>
#include <stdio.h>
#include "pico/stdlib.h"
#include "semphr.h"

static SemaphoreHandle_t mutex;

void task1(void *pvParameters)
{
    char ch = '1';
    while (true) {
        if(xSemaphoreTake(mutex, 0) == pdTRUE){
            for(int i = 1; i < 10; i++){
                putchar(ch);
            }
            puts("");
            xSemaphoreGive(mutex);
        }
    }
}

```

```

}

void task2(void *pvParameters)
{
    char ch = '2';
    while (true) {
        if(xSemaphoreTake(mutex, 0) == pdTRUE){
            for(int i = 1; i < 10; i++){
                putchar(ch);
            }
            puts("");
            xSemaphoreGive(mutex);
        }
    }
}

int main()
{
    stdio_init_all();

    mutex = xSemaphoreCreateMutex();

    xTaskCreate(task1, "Task 1", 256, NULL, 1, NULL);
    xTaskCreate(task2, "Task 2", 256, NULL, 1, NULL);
    vTaskStartScheduler();

    while(1){};
}

```