**********Draft**********

Part 4 pico
freeRTOS
Mutex
02/22/22

**********Draft**********

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

image1

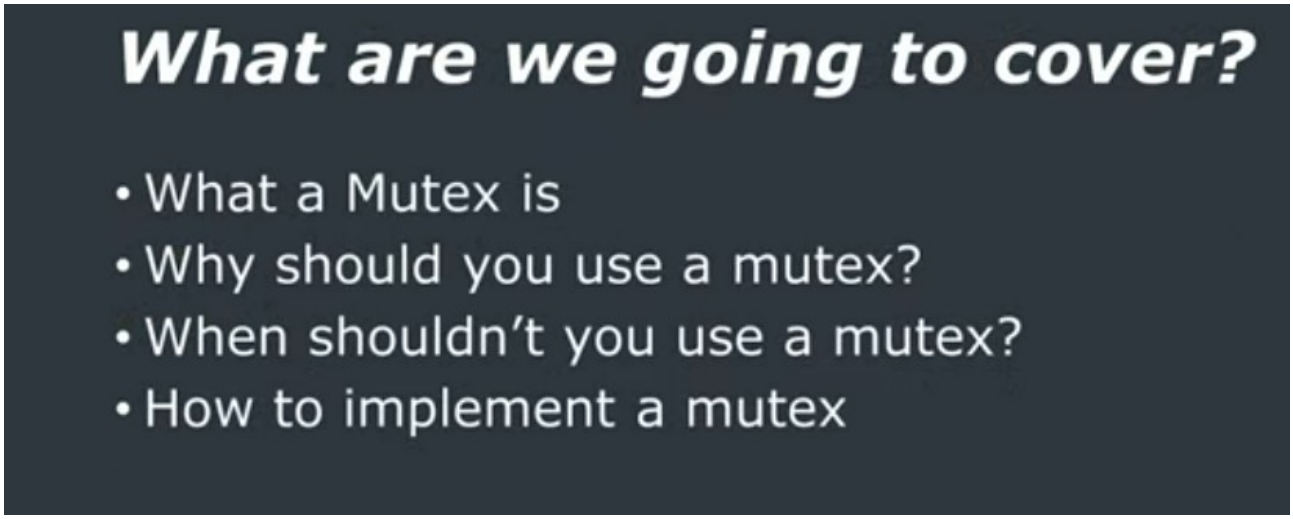
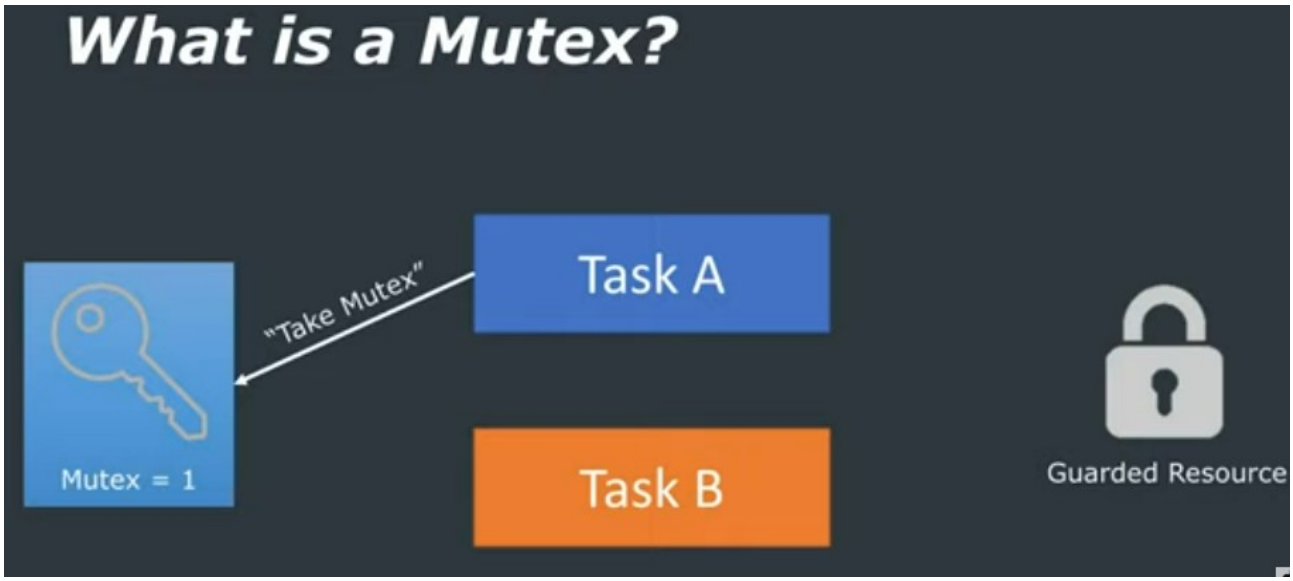


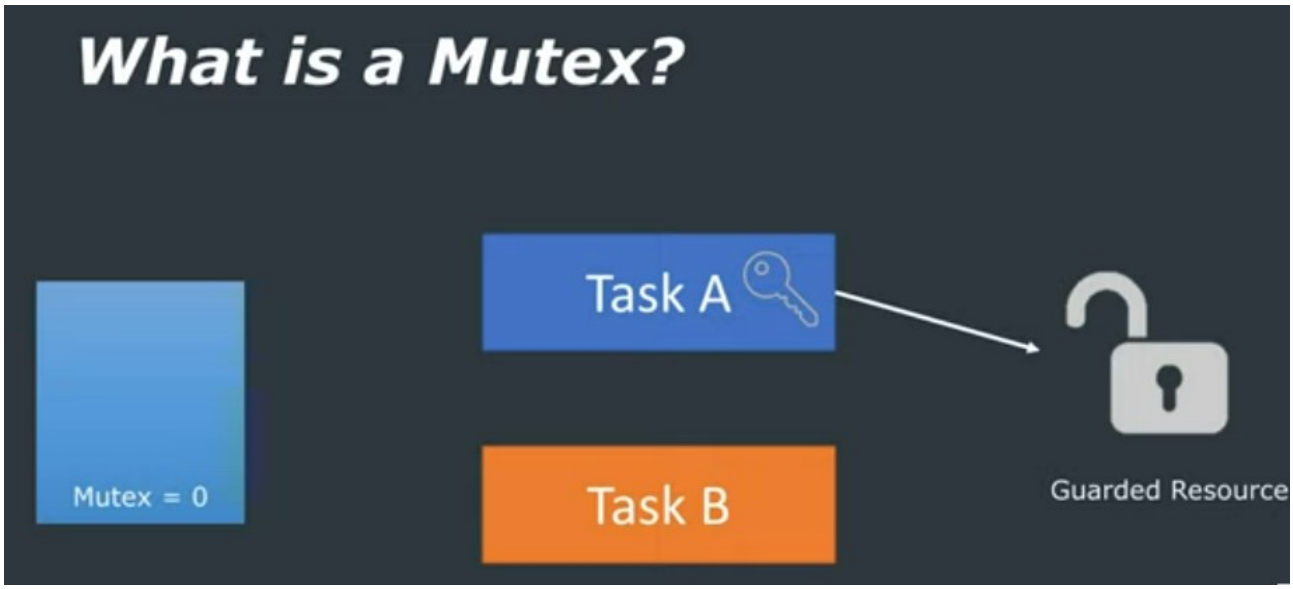
image2

image3

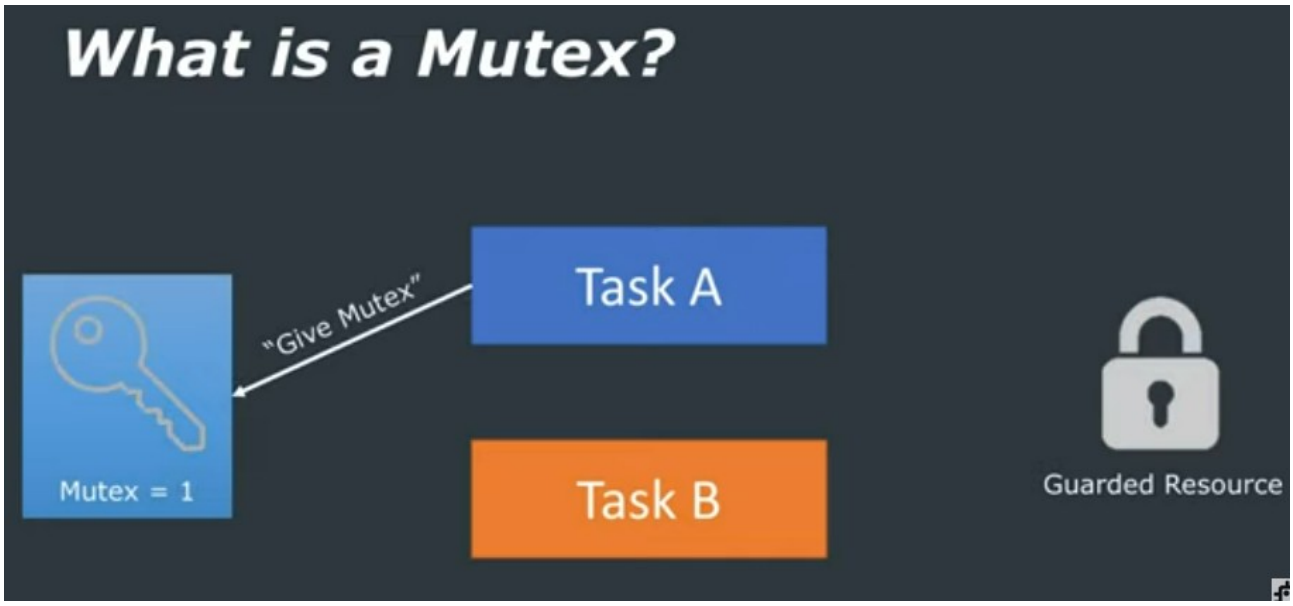
Task needs to gain access to Guarded Resource



Task A uses the 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 inclue 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

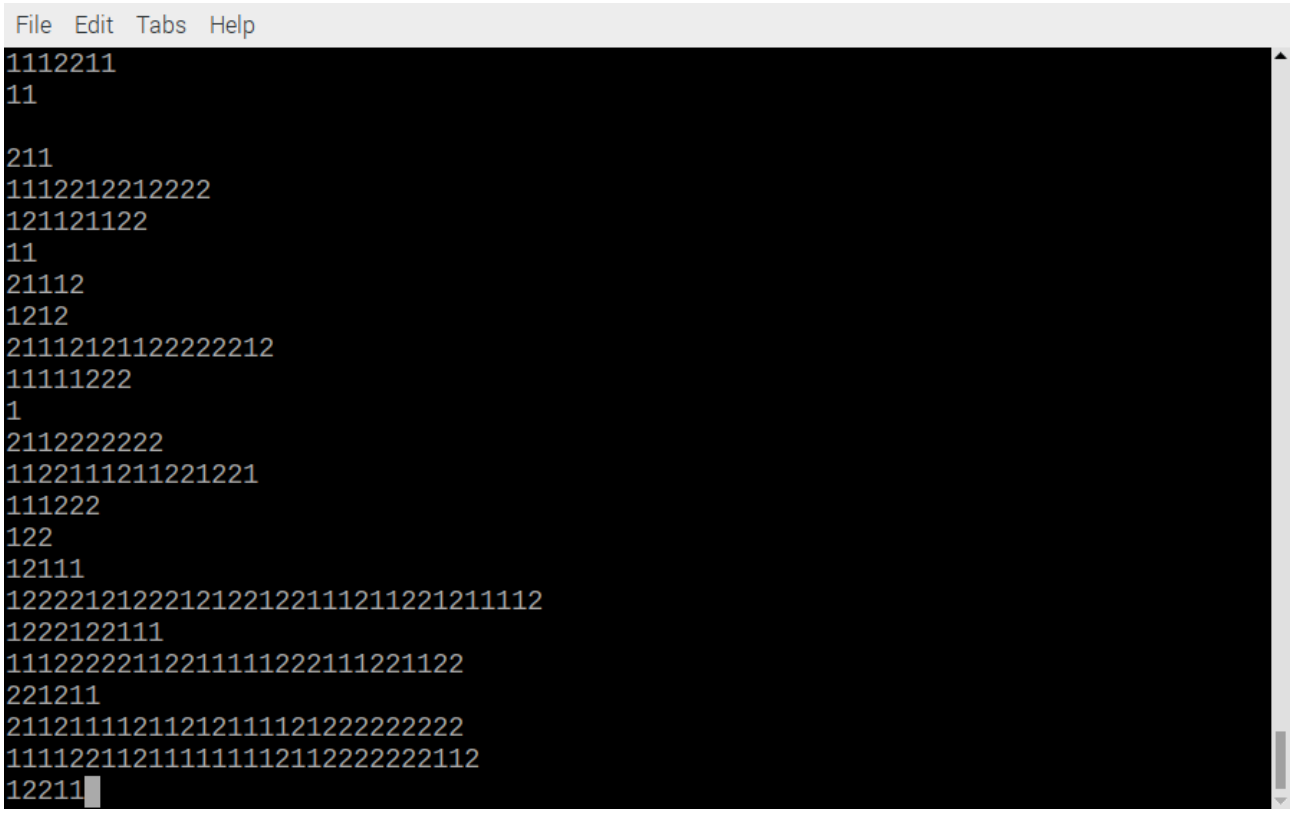
Line 13 FreeRTOSConfig.h
#define configTICK_RATE_HZ 100000

~/rp2040-freertos-project/build $
cmake ..
make

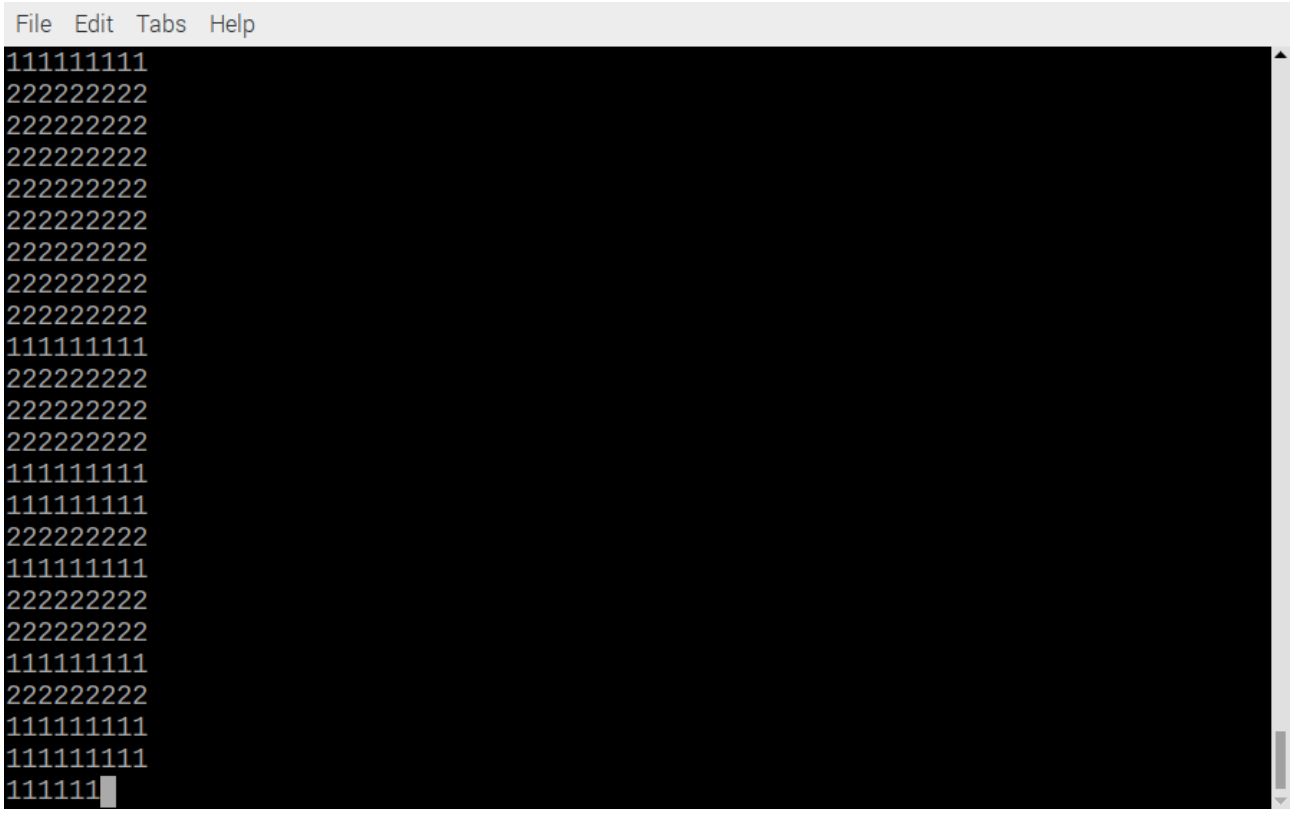
openocd -f interface/raspberrypi-swd.cfg -f target/rp2040.cfg -c "program Mutex/Mutex.elf verify reset exit"

sudo minicom -s

Code without Mutex



Code with Mutex



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){};
}
```