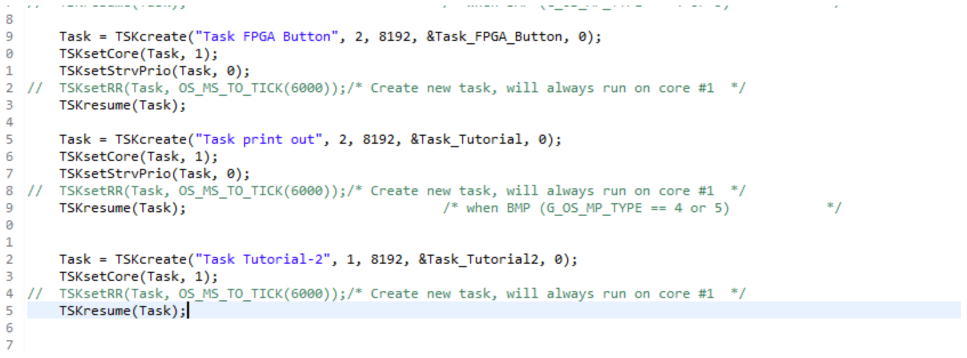
LINGI 2315 - Evaluation 2 - mAbassi Project **P1**

Name	Delcoigne Ben	Noma	38771700

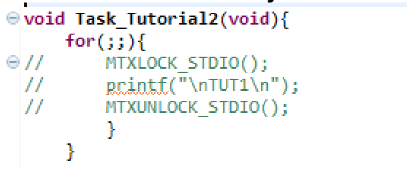
Definition of the tasks, the content of the new task and the implementation and explanation of the Starvation mechanism with relevant screenshots of your code

I first create all tasks, all running on the same core. I already added the starvation mecanism.
Note that task_tutorial_2, an infinite loop, has a higher priority than the two others.



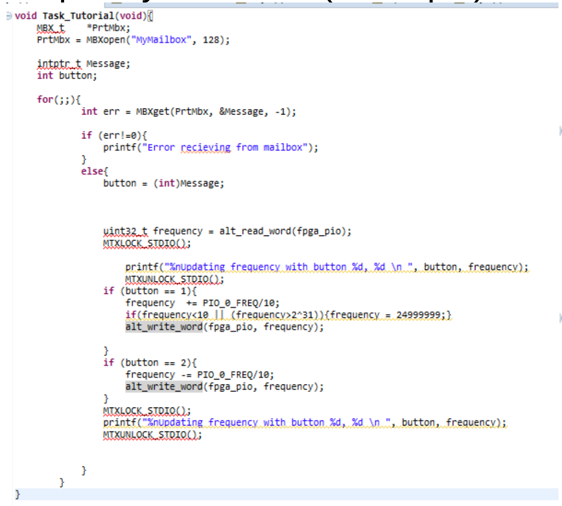
```
    Task = TSKcreate("Task FPGA Button", 2, 8192, &Task_FPGA_Button, 0);
    TSKsetCore(Task, 1);
    TSKsetStrvPrio(Task, 0);
//  TSKsetRR(Task, OS_MS_TO_TICK(6000));/* Create new task, will always run on core #1  */
    TSKresume(Task);

    Task = TSKcreate("Task print out", 2, 8192, &Task_Tutorial, 0);
    TSKsetCore(Task, 1);
    TSKsetStrvPrio(Task, 0);
//  TSKsetRR(Task, OS_MS_TO_TICK(6000));/* Create new task, will always run on core #1  */
    TSKresume(Task);                              /* when BMP (G_OS_MP_TYPE == 4 or 5)            */


    Task = TSKcreate("Task Tutorial-2", 1, 8192, &Task_Tutorial2, 0);
    TSKsetCore(Task, 1);
//  TSKsetRR(Task, OS_MS_TO_TICK(6000));/* Create new task, will always run on core #1  */
    TSKresume(Task);
```

Task tutorial 2 is just a simple while loop.

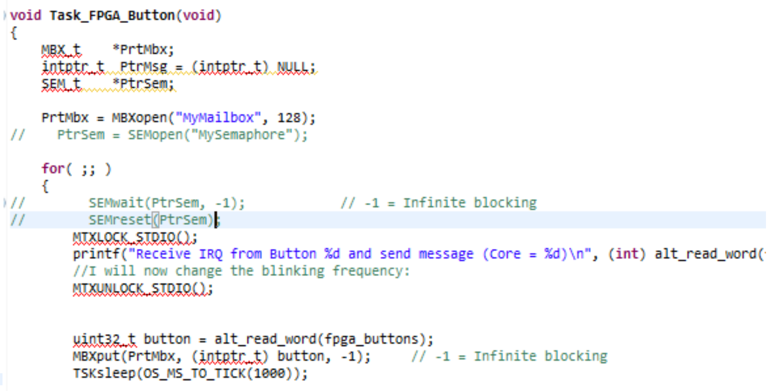


```
void Task_Tutorial2(void){
    for(;;){
//      MTXLOCK_STDIO();
//      printf("\nTUT1\n");
//      MTXUNLOCK_STDIO();
        }
    }
```

Task tutorail is in charge of opening the mailbox mymailbox and reading the messages from it. From the message recieved (which is the value of a button), it changes the frequency of the led (in the pio)



```
void Task_Tutorial(void){
    MBX_t     *PrtMbx;
    PrtMbx = MBXopen("MyMailbox", 128);

    intptr_t Message;
    int button;

    for(;;){
            int err = MBXget(PrtMbx, &Message, -1);

            if (err!=0){
                printf("Error recieving from mailbox");
            }
            else{
                button = (int)Message;



                uint32_t frequency = alt_read_word(fpga_pio);
                MTXLOCK_STDIO();

                    printf("%nUpdating frequency with button %d, %d \n ", button, frequency);
                    MTXUNLOCK_STDIO();
                if (button == 1){
                    frequency  += PIO_0_FREQ/10;
                    if(frequency<10 || (frequency>2^31)){frequency = 24999999;}
                    alt_write_word(fpga_pio, frequency);

                }
                if (button == 2){
                    frequency -= PIO_0_FREQ/10;
                    alt_write_word(fpga_pio, frequency);
                }
                MTXLOCK_STDIO();
                printf("%nUpdating frequency with button %d, %d \n ", button, frequency);
                MTXUNLOCK_STDIO();


            }
        }
}
```



```
void Task_FPGA_Button(void)
{
    MBX_t    *PrtMbx;
    intptr_t  PtrMsg = (intptr_t) NULL;
    SEM_t    *PtrSem;

    PrtMbx = MBXopen("MyMailbox", 128);
//    PtrSem = SEMopen("MySemaphore");

    for( ;; )
    {
//        SEMwait(PtrSem, -1);            // -1 = Infinite blocking
//        SEMreset(PtrSem);
        MTXLOCK_STDIO();
        printf("Receive IRQ from Button %d and send message (Core = %d)\n", (int) alt_read_word(
        //I will now change the blinking frequency:
        MTXUNLOCK_STDIO();


        uint32_t button = alt_read_word(fpga_buttons);
        MBXput(PrtMbx, (intptr_t) button, -1);     // -1 = Infinite blocking
        TSKsleep(OS_MS_TO_TICK(1000));
```

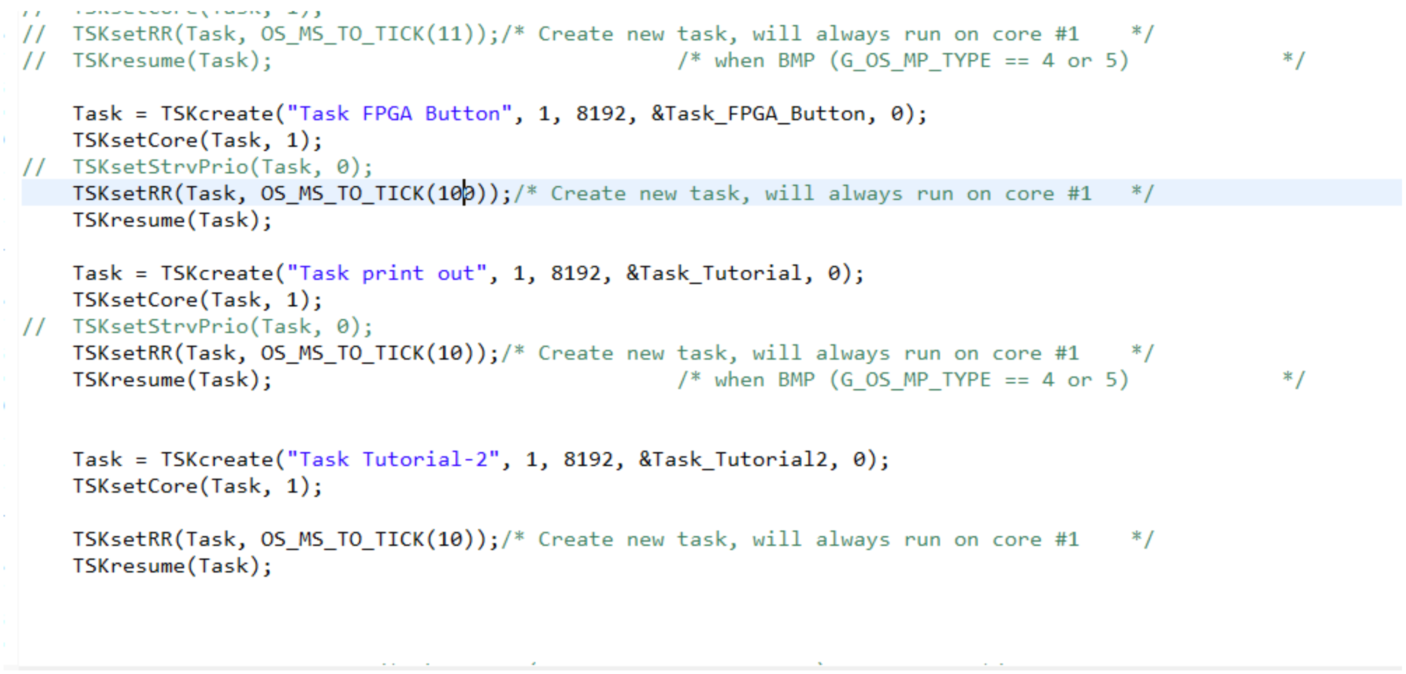
The last task, fpga button, is in charge of reading the buttons (every 1 second) and to updates the mailbox.
Since starvation is set to 1 for both tasks of priority 2, as they never run, they will increase their priority to priority 1 and become equals with the infinite loop. They thus now act all three in a round-robin manner. They do however have the autorisation to go to priority 0 but they do not need to do so.

LINGI 2315 - Evaluation 2 - mAbassi Project **P2**

Name	Delcoigne Ben	Noma	38771700

Definition of the tasks, the implementation and explanation of the Round-Robin mechanism with relevant screenshots of your code

The only operated change was to remove the starvation and to set round robin timings:
I set all priorities to 1;



```
//  TSKsetRR(Task, OS_MS_TO_TICK(11));/* Create new task, will always run on core #1    */
//  TSKresume(Task);                                /* when BMP (G_OS_MP_TYPE == 4 or 5)           */

    Task = TSKcreate("Task FPGA Button", 1, 8192, &Task_FPGA_Button, 0);
    TSKsetCore(Task, 1);
//  TSKsetStrvPrio(Task, 0);
    TSKsetRR(Task, OS_MS_TO_TICK(100));/* Create new task, will always run on core #1    */
    TSKresume(Task);

    Task = TSKcreate("Task print out", 1, 8192, &Task_Tutorial, 0);
    TSKsetCore(Task, 1);
//  TSKsetStrvPrio(Task, 0);
    TSKsetRR(Task, OS_MS_TO_TICK(10));/* Create new task, will always run on core #1    */
    TSKresume(Task);                                /* when BMP (G_OS_MP_TYPE == 4 or 5)           */


    Task = TSKcreate("Task Tutorial-2", 1, 8192, &Task_Tutorial2, 0);
    TSKsetCore(Task, 1);

    TSKsetRR(Task, OS_MS_TO_TICK(10));/* Create new task, will always run on core #1    */
    TSKresume(Task);
```

Task FPGA button runs for 100ms, reading the button and when an entry is new (a new input), it sends it to the mailbox.
Taskprint out runs for 10ms and opens the mailbox.Either it blocks on the call if it is empty, or it runs and updates the blinking frequency (no code change since page 1)
Task_Tutorial_2 is an infinite loop running for 10ms each cycle.

Total cycle time is thus 120ms.