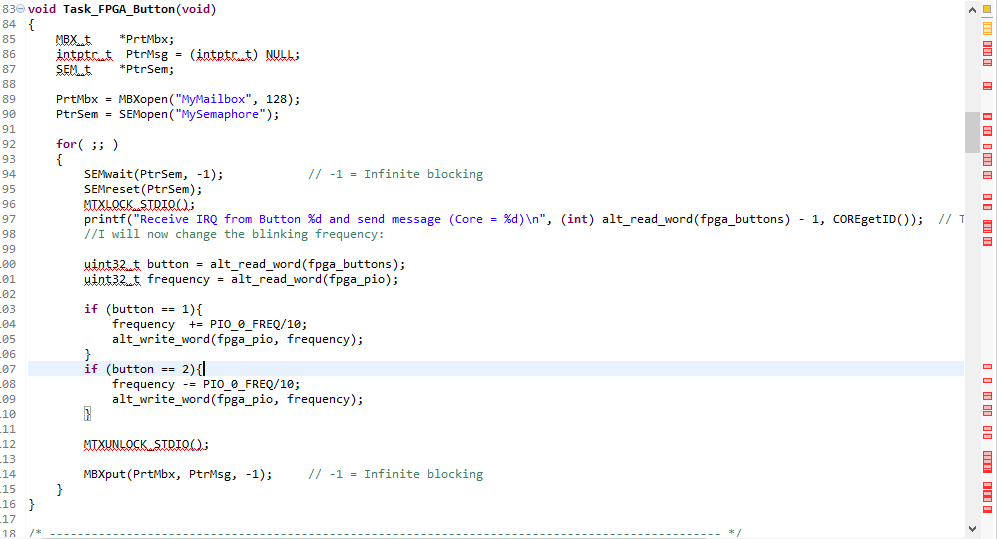
**LINGI 2315 - Homework 3 - My RTOS App P1**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Delcoigne Ben | Noma | 3877 1700 |

Description of the task and other functions that implement the control of the blinking frequency of LED[0] with relevant screenshots of your code

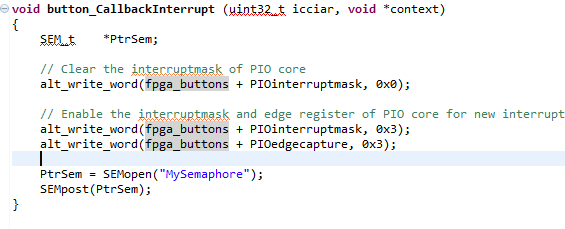
On the hardware part, I used the sof file we used on all other homeworks. No changes were done there (just make sure there is a pio\_0 of 27 or 32 bits for the frequency, I took 32 bits )

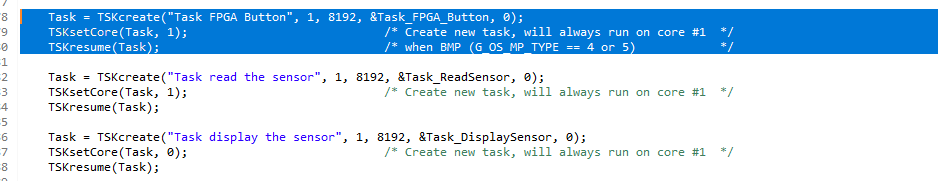
I created a new function in charge of modifying the led frequency :



This task runs constantly, but as you can notice, it is blocked by a semaphore. This semaphore is unlocked by the button callbackinterrupt:

Concretely, fpga\_button has an infinite loop. At SEM\_wait, the loop blocks and is waiting for someone to tell it to go on. That is done by a whole other thread (the one that handles interrupts).Once we have a button interrupt, we SEMost, which releases the semaphore and lets the for loop make one revolution. The cycle repeats.



I also point out to the fact that the task was created, but is nowhere called, that’s why I make core 1 run it by adding the following code in main\_mabassi.c

(I also added the code for readsensor and displaysensor which are for the second part of the HW)

**LINGI 2315 - Homework 3 - My RTOS App P2**

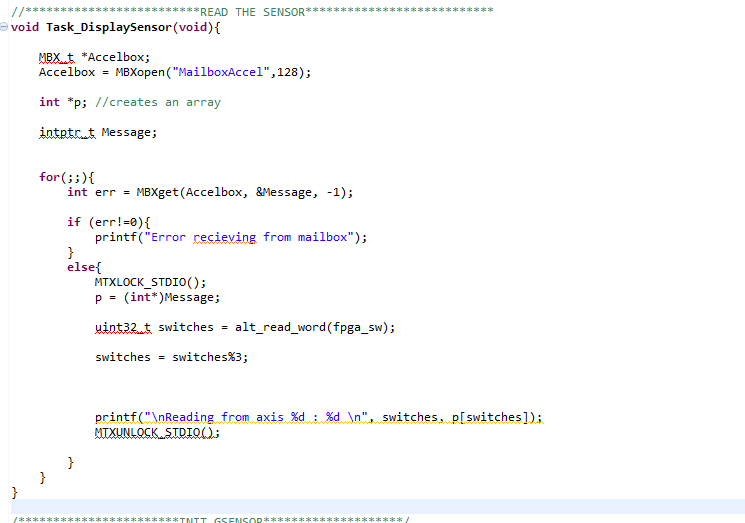
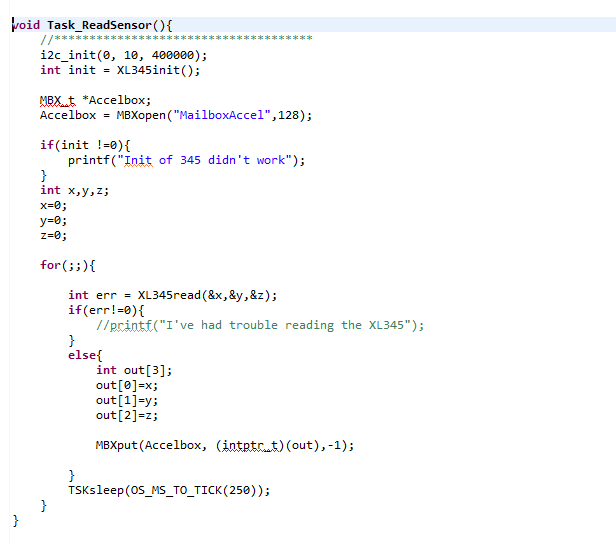
|  |  |  |  |
| --- | --- | --- | --- |
| Name | Delcoigne Ben | Noma | 38771700 |

Description of the 2 tasks and other functions that implement the reading and display of 1-axis acceleration with relevant screenshots of your code.

First step was to make the drivers working to be able to interact with the gsensor. I did just like in demo 8 (you can see what I did since I possted a few steps in the forum). But basically:

Reuse the DMA init code from demo8 (which is just above in the screen but takes a lot of space in the screenshot if I add it), and init the i2c and XL345.

Note\* you must add the libraries in the makefile



Once done, I create a mailbox which will be used to communicate with the display task. I then read what’s in the sensor, and send it to the mailbox.

On the other end, I must create a task that reads the mailbox and displays the text. (on the right). Both these tasks were initialized in main\_mabassi.c (see previous page). For fun, I put one task on core 1 and the other on core0. I notice that the code works and thus the communication is indeed done between multiple cores. Great!

Litte lote on task\_displaysensor:

I first setup all my variables, then indefinetly (and with no time.sleep()), I open the last message from the mailbox and print it out. In order to choose the right axis, I read the fgpa\_switches (which were setup just like fpga\_Buttons and fpga\_pio0), and %3 in order to choose one of the axis .

**Note:**

The MTXLOCK\_STDIO() are used to lock onto the standard output: this is to avoid multiple threads trying to write text at the same time (and thus displaying characters interleaved with others).