**LINGI 2315 - Evaluation 3 - Project P1**

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

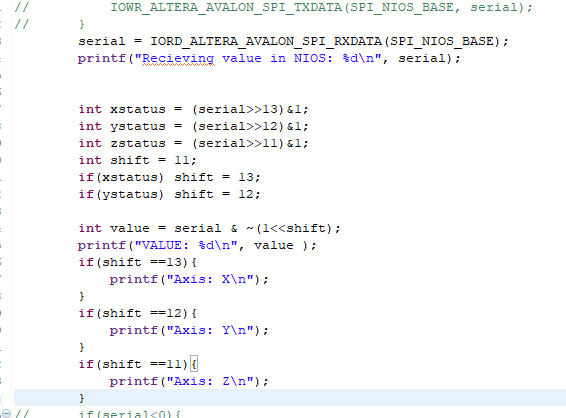
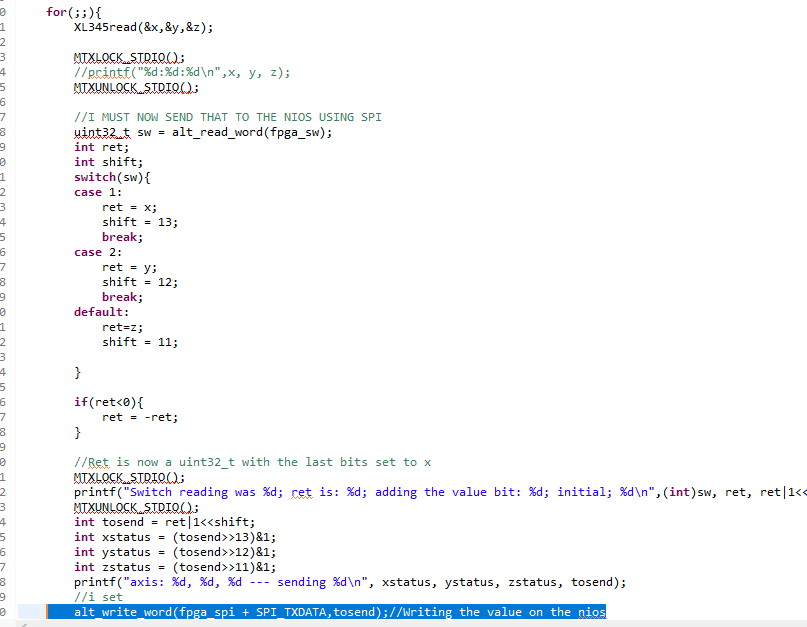
Explanation of your implementation for question 1 with relevant screenshots of your code

I created a task called gsensor (which was created for our homework) that constantly reads the value of the gsensor.

Once done, I must select the axis, for that I read from the dispsw (i had to create the fpga\_sw variable as usual). Once done, decided which value to read. Depending on the switch, ret will be the x, y or z value. I also set a shift 13,12 or 11 bits.

I then take the value (ret) and set the nth bit to 1; this enables me to deduce whether it was the x, y and z axis on reception. Indeed; At the reception i just have to figure out which switch was set, unset it and display the value.

Note: i start from 11 bits shift because I need 10 bits to display the g value (which goes up to 1024); when expecting bigger values, the shift should be higher but it’s a quick fix

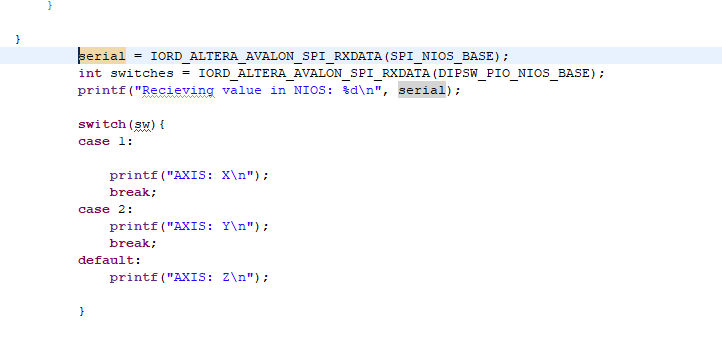
. 

For the SPI, (right) I just decode the value by doing the inverse operation, and deduce which axis was used. This is what was shown at the demo.

Note:

I could have done much simpler. I could just have read the switch state from the SPI.

This is much simpler: I don’t have to encode the bits and can just plainly send the value of the gsensor to the nios. Code of SPI becomes:

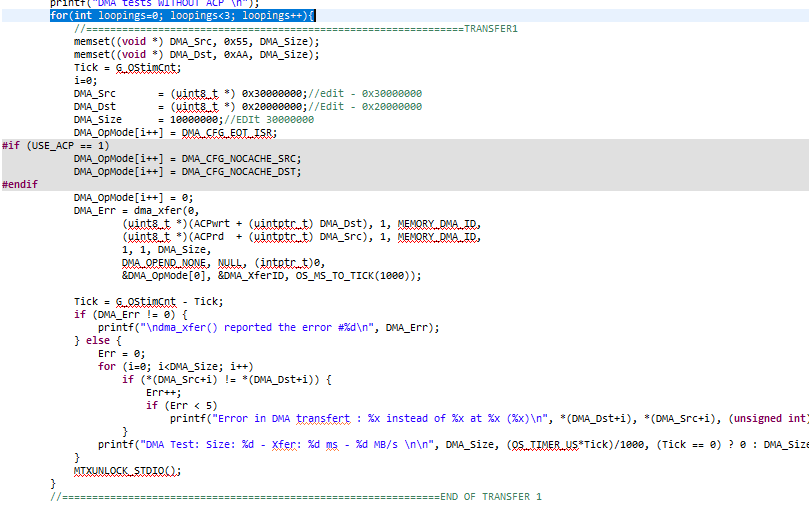


**LINGI 2315 - Evaluation 3 - Project P2**

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

Explanation of your implementation for question 2 with relevant screenshots of your code

For question 2; I used the transfer example from myapp\_DMA and repeated the transfer three times using a for loop.



Afterwards, I enable the ACP:



And repeat the for loops described above. We notice the transfer is faster without ACP which makes sense since ACP checks consistency across cache&memory.