|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *B7* | | |
| Homework number: | *HW07* | | |
| Due date: | 15/11/2022 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| Giacomo Massa |  |  | *x* |
| Simone Giampà |  |  | *X* |
| Luca Francesco Raduzzi |  |  | *x* |
| Johanna Micelli |  |  | *x* |
| Claudio Galimberti |  |  | *x* |
| Notes: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | HW\_07\_Accelerometer\_UART\_DMA | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Successfully completed |
|  |  |  | *x* |
| SETUP: Timer 2 must be configured with a working frequency of 1 Hz, imposing the correct values for PSC and ARR, and its interrupt must be enabled. Then, we configure the I2C by activating pins PB8 and PB9, to communicate with the accelerometer, corresponding to SCL and SDA. The next step is to configure the UART, enabling the DMA (and its interrupt) and imposing a byte as data width to be passed.  CODE:  We start by setting up the flag raised by the timer interrupt, then we handle the interrupt routine in the infinite while cycle. We set up the necessary addresses for the accelerometer. Two addresses are needed to find what model is integrated on the board (we found lis2de). Then three control registers of the port are configured as arrays of two elements: the first is the address of the register, and the second is the value we want to impose. It’s important to notice that we configured control\_reg\_1 with the working frequency at 1Hz, by asserting the fifth LSB.  After this setup, we must recognize what port is integrated into our board. To do so, we simply transmit through the I2C protocol and verify what address is the correct one and we save it. With the correct address, we can transmit the control register values.  We enter now the reading part. In the interrupt routine, we first transmit the address we want to read (x, y, and z acceleration values addresses) and then receive the value. The reading part is done by asserting as usual the LSB of the address of the device to 1. We do this operation for all the three accelerations measured, then we divide every conversion value by 64.  The values are all transmitted in a single line via UART using DMA. | | | |
| Professor comments: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | HW\_07\_Accelerometer\_UART\_I2C\_DMA | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Successfully completed |
|  |  |  | *x* |
| SETUP: the setup is the same as the previous project, the only difference is that also the I2C has a DMA request, with the corresponding interrupt, and as before a data width of a byte.  CODE: Since we have two interrupt routines to manage, it’s easier to compute them inside the callback, rather than in the infinite while cycle.  As done before, we configure the usual addresses, add the variable multiple\_readings, and use an array of 6 elements to contain the output readings. The reason to do this is the incremental reading function of the port, which increases by 1 the value of the address to be read. We must read OUT\_X, OUT\_Y, and OUT\_Z, but between them, there are other registers that we don’t care about reading.  As done before, we recognize the model we are using and configure the control registers.  In the timer interrupt routine, we enable the successive reading passing at the accelerometer the address of the register OUT\_X, but with the MSB imposed to 1. Then through a single receive (using the DMA), we can read all three values of output, saving them in the accelerations array.  As it can be read on page 27 of the lis2de datasheet, the positions of our data will be 0, 2, and 4, while 1 and 3 are useless registers. The sixth element of the array was used only for debugging purposes.  As before, we divide them by 64 and transmit them via the UART interface. | | | |
| Professor comments: | | | |