|  |  |
| --- | --- |
| **Mark** | **/11** |

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *A5* | | |
| Homework number: | *HOMEWORK 08* | | |
| Due date: | 17/11/2024 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| Alessio Spineto |  |  | *x* |
| Riccardo Lamarca | *x* |  |  |
| Sofia Cecchetto |  |  | *x* |
| Annamaria De Togni |  |  | *x* |
| Emma Crespi |  |  | *x* |
| Notes: none | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | Accelerometer | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Completed |
|  |  |  |  |
| The objective of the homework was to complete the MEMS accelerometer projects 1b and 1c. As stated during the lecture, we completed just the homework 1c which consisted in the use of DMA mode for both communication protocols (I2C and USART).  **Part 1a:**  The first step was to set the correct configuration for the I2C pins.    We enabled the I2C mode for the pins and then we configured the I2C connection, particularly I2C1.  The default parameters are correct for our application, but we need to enable the two interrupts in NVIC Settings (I2C1 event interrupt and I2C1 global interrupt) and configure the use of DMA in both directions (peripheral to memory and memory to peripheral) with high priority.  For the USART communication we enabled the global interrupts and configured the DMA, this time just in the “memory to peripheral” direction.    We also set TIM2 to trigger an interrupt every second and enabled its global interrupt.  We then defined some constants useful for working with the accelerometer, like the address of the sensor in I2C *(ADDRESS\_WRITE* and *ADDRESS\_READ*, both left-shifted by 1 and with the correct R/W bit), the sub-addresses of the registers inside the accelerometer as written on the data sheet and the values to set them, and the sub-address of the register containing X-axis acceleration data (*X\_REG*) with the MSB already set to 1, there is no need to specify the Y and Z-axis registers, since we can see from the datasheet that it’s possible to read one register after another starting from *X\_REG*  Before setting the I2C address, we needed to check which sensor we have installed on our green board. After testing by writing another firmware for the MCU we concluded that all components of the group have the LIS2DE sensor. It’s possible to adapt the code for the LIS2DE12 model by removing the comments on the corresponding rows.    Then we declare the buffer to save the acceleration data for the three axis and a string to communicate with a PC.    In the main() function we needed to transmit the data to configure the sensor:   * enable normal mode operation, with all 3 channels active * 1 Hz update rate * no high pass filter * ± 2g Full Scale range.   To do so, we needed to access the *CTRL\_REG1-2-4* and write the 3 values we defined before in rows 42 to 44.  We saved these values in an array of uint8\_t and transmitted them (no DMA is needed since the microprocessor resources are dedicated only to this transmission when this code is executed).  Then we start TIM2 in interrupt mode.    After the configuration is done and the timer starts, the callback for the elapsed period is called every second.  Inside the timer callback, we write to the sensor that we want to read from OUT\_X with the MSB set to 1, this way the sensor knows we want to read starting from OUT\_X and the subsequent registers.    After the transmission is complete, the *MasterTxCpltCallback* is called, where we start a *Receive* to read the first 5 registers starting from the address of OUT\_Xand store the values inside the buffer.  In the datasheet, the Register address map shows that if we start to read from OUT\_X, the y and z acceleration values will be saved in the second and fourth element of our buffer (since the addresses 2A and 2C are reserved).    In the Rx callback we read and converted the values from *buffer[0]*, *buffer[2]* and *buffer[4]*.  We then transmit with USART the values converted in g to the remote terminal.    Output obtained by rotating the board to align with the 3 axis: | | | |
| Professor comments: | | | |