|  |  |
| --- | --- |
| **Mark** | **A** |

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *B5* | | |
| Homework number: | *06* | | |
| Due date: | Tuesday, 14th November, 8:30 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| Ghidini Alessandro |  |  | *x* |
| Latino Francesco |  |  | *x* |
| Luppi Eleonora |  |  | *x* |
| Bravin Riccardo |  |  | *x* |
| Feltrin Elia |  |  | *x* |
| Notes: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | Temperature sensor & Accelerometer with I2C | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Completed |
|  |  |  | *x* |
| Explanation:  To complete the three assignments, we have to combine work done in most of the previous homework's since timers, DMA, USART and I2C knowledge is needed.  In the first part we read the LM75B temperature sensor values each second through I2C communication and show them in terminal through UART communication.  In the second part we read the LIS2DE accelerometer values each second through I2C communication by using a timer interrupt through the elapsed period callback, then send the values to the terminal by means of UART communication with DMA.  In the third part we start from the previous readings of the accelerometer and we use the I2C DMA and interrupts to perform a multiple reading of the accelerometer and obtain the right values that are then shown on the terminal again.  **Part 6a:**  To be able to read all 11 bits of information from the temperature sensor each second and send the result via USART we need to:   * Enable the I2C connectivity feature. * Enable the USART connection and enable its interrupts. * Set up a timer (TIM3) with Prescaler = 8400-1 and Counter Period = 10000-1 and enable the global interrupts flag.   After generating the code, inside the main.c file we will transmit through I2C to the temperature sensor (device address 0b10010000) the value 0x00 to tell it that we want to read that register (temperature data register) by means of *HAL\_I2C\_Master\_Transmit(&hi2c1, ThermAddress, &ThermDataAddr, 2, 100)*. Then we start the TIM3 interrupts generation through *HAL\_TIM\_Base\_Start\_IT*.  Inside the *HAL\_TIM\_PeriodElapsedCallback* function we wrote the code for the correct readout of the sensor. We start by calling *HAL\_I2C\_Master\_Receive* on the device 0b10010001 expecting two bytes of data to be stored in an *uint8\_t ThermBuffer[2]*.  To perform the conversion, we memorize correctly the MSB and the LSB and then we combine them into a single *uin16\_t* variable. Before doing this, to resolve the unsynchronized reads bug we perform another reading of the MSB only and we store it in an auxiliary variable *MSB\_new*, then we use an if construct to control if the readings are the same: if not, we update the MSB value with the one of the new reading and then we combine the MSB and the LSB by shifting them of the right positions as *TempBits = ((MSB << 3) | (LSB >> 5))*. Then we control the sign of the arrived bites by controlling the value of the most significant bit and we convert the value according to 2’s complement conversion.  Finally, the correct data is obtained as *temperature = TempBits \* 0.125*. From the code:  *MSB = ThermBuffer[0];*  *LSB = ThermBuffer[1];*  *if(HAL\_I2C\_Master\_Receive(&hi2c1, ThermAddress+1, &MSB\_new, 1, Timeout) == HAL\_OK){*  *if (MSB\_new != MSB){*  *MSB = MSB\_new;*  *}*  *} else {*  *length = snprintf(string, sizeof(string), "Error in data reading!\r\n");*  *}*  *TempBits = ((MSB << 3) | (LSB >> 5));*  *if (MSB & 0x80){*  *TempBits = (TempBits ^ 0xFFF) + 1;*  *temperature = - TempBits \* 0.125;*  *} else {*  *temperature = TempBits \* 0.125;*  *}*  We then send the temperature value inside a string via UART to terminal using *HAL\_UART\_Transmit*.  **Part 6b:**  To correctly read the accelerometer via I2C protocol using the DMA to send data through USART we perform the following preliminary steps:   * Enable the I2C connectivity feature. * Enable the USART connection and enable its interrupts. * Configure the path for transmission in normal mode in the USART DMA tab. * Set up a timer (TIM2) with Prescaler = 8400-1 and Counter Period = 10000-1 and enable the global interrupts flag   After generating the code, in the main.c file we sent to the peripheral’s control registers the values to correctly initialize the device in an array of two elements, with the first element which is the register sub-address and the second one is the register value for configuration. We send:   * 0b00010111 to CTRL\_REG1 (0x20) to enable data reception, set data rate to 1 Hz and enable all three axes’ data reception. * 0b00000000 to CTRL\_REG2 (0x21) since we do not want any high pass filter for data reception. * 0b00000000 to CTRL\_REG4 (0x23) to have a FSR and disable BDU, self-test and SPI.   The device is identified between the address 0b01010000 and 0b00110000 by waiting for an ACK while sending out the configuration bits.  We then enable the base clock interrupt generation.  Inside the *HAL\_TIM\_PeriodElapsedCallback* function we sequentially transmit the bits of the address we want to read (0x29, 0x2B, 0x2D) to the device and fill the respective variables of interest through the *HAL\_I2C\_Master\_Receive* function, as:  *HAL\_I2C\_Master\_Transmit(&hi2c1, device\_addr, &subregister\_address, 1, 50);*  *HAL\_I2C\_Master\_Receive(&hi2c1, device\_addr + 1, &axis\_data, 1, 50);*  Once all three axes have been read, we perform the conversion in g values by diving each value by 64, because the device has an 8-bit resolution (256 possible values) and we want a final conversion in the FSR we previously set.  Finally, we send the values out with *HAL\_UART\_Transmit\_DMA* once composed into a string.  **Part 6c:**  Starting from the previous project we need to configure the DMA of the I2C for peripheral to memory transmission and enable its interrupts.  Then inside the *HAL\_TIM\_PeriodElapsedCallback* function we just need to transmit 0b10101001 (MSB set to 1 + addr of register of the first data to be read, which is 0x29) with *HAL\_I2C\_Master\_Transmit* and wait to receive 5 values to be stored inside a 5 elements array performing sequential readings through *HAL\_I2C\_Master\_Receive\_DMA*.  Inside the *HAL\_I2C\_MasterRxCpltCallback* overridden inside the main.c we then read the 3 axis values of the accelerometer from the array, remembering that data are stored in registers 0x29, 0x2B and 0x2D and we are performing sequential multiple readings, so we discard the 2 elements between the axes’ data.  Finally, we convert them to g and send them with *HAL\_UART\_Transmit\_DMA* after having composed them inside a string. | | | |
| Professor comments:  Part a: in order to account for the sign it is enough to cast the value to a 16 bit integer (and than divide by 256). | | | |