|  |  |
| --- | --- |
| Mark |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *A2* | | |
| Homework number: | *07* | | |
| Due date: | 03/11/2024 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| La Barbera Marco |  |  | *x* |
| Lotto Giulio |  |  | *x* |
| Majocchi Tommaso |  |  | *x* |
| Maffezzini Andrea |  |  | *x* |
| Pompilio Matteo |  |  | *x* |
| Notes: none | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | I2C thermometer | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Completed |
|  |  |  | *x* |
| We successfully completed the homework.  Next, we will explain all the steps for accomplishing our goals:  **Part 1b:**  First of all, we configure the board pinout for the I2C transmission with the temperature sensor (pins PB8 and PB9 as verified in the schematic file) and for the UART transmission with the pc, pin PA2:    Then, from “connectivity*”* we enable the I2C communication in this way:    And the UART for the remote transmission of the temperature value to the MATLAB console, in DMA mode as always:    Finally, we set the timer with the usual values (timeout parametrized by the constant TEMPO):    Then in the “main.c” file we initialize the TIM2 base generation and the I2C transmission:    For this purpose, we declare two constants (timeout and size) and two global variables: LM75\_WR\_ADDRESS and LM75\_REGISTER, respectively the address of the sensor and the address of the register where the converted temperature is stored. LM75\_RD\_ADDRESS will be used in the HAL\_I2C\_Master\_Receive function. The write and the read requests can be distinguished by the LSB since it is ‘0’ for the write address and ‘1’ for the read address.    Then we proceeded to define other variables, in particular:   * *TEMPO* will dictate the frequency of the timer; * *data[2]* is an array that will store the temperature values sent by the sensor; * *data\_old* is a variable that we used to solve “the bug” due to the use of different sensors (LM75 and LM75B); * *size* is a variable that indicates the number of bytes received or transmitted during the I2C communication; * *timeout* is the timeout time in ms; * *string* and *string\_length* are used as usual to convert an integer value into a string; * *concat\_data* is a variable that stores the arranged value of the 11-bit temperature; * *temperature* is the variable that stores the converted temperature value;     In the TIM2 callback function, we arrange the receive and control sections of both temperature sensors (LM75 and LM75B). With the function *HAL\_I2C\_Master\_Receive* we set a receive whenever the TIM2 callback is triggered (controlled by the TEMPO variable, each 1 second). *LM75\_RD\_ADDRESS* is used to inform the LM75 that we want to read its values and store them in *data* (that is an array of *size + 1*). Since we deal both with integer and decimal parts, *size + 1* is set equal to 2. Then we aggregate the useful bits into the *concat\_data* variable. Once done we check if the MSB of *data[0] (data[0]* contains the integer part of the temperature),is ‘1’ (negative number) or ‘0’ (positive number): if the value is negative, we have to make sure that also the first 5 bits of *concat\_data* are ‘1’ and then we can convert the value into the temperature using the 2’s complement. Otherwise, we directly convert *concat\_data* into the temperature.  Then we proceed to write the value into a string using the *snprintf()* function and we send the string to MATLAB through the UART as always.  We can see that we reset the value of *concat\_data* for the next function call. We also update data\_old to data[1], but why do we do this?  In line 91, there is a commented command that should be uncommented if you are using the LM75A instead of the LM75B. LM75B requires this line to remain commented. This implementation is necessary due to a bug associated with the LM75B.    We verified that the sensor LM75B has a “bug”, its solution has been reached after reading these two lines in the two different datasheets:  LM75A: If a conversion is in progress, it will be stopped and restarted after the read.  LM75B: Reading temperature data does not affect the conversion in progress during the read operation.  This may imply that during the read operation the values converted by the LM75B sensor change, so the values received with one transmission belong to two different conversions (differently from what happens with the other sensor, the LM75A).  The code for the LM75B sensor solves this issue delaying the transmission of the LSB byte to the following iteration.  Switching to MATLAB, we can now run the script “UART\_read\_data.m” to read the voltage values at a baud rate of 115200 bps (as set on our board). We successfully receive the data on our console: | | | |
| Professor comments: | | | |