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

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *A2* | | |
| Homework number: | *05* | | |
| Due date: | 20/10/2024 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| La Barbera Marco |  |  | *x* |
| Lotto Giulio |  |  | *x* |
| Majocchi Tommaso |  |  | *x* |
| Maffezzini Andrea |  |  | *x* |
| Pompilio Matteo |  |  | *x* |
| Notes: none | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | UART project with DMA & LCD scroll member names | | |
| 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 2b:**  Firstly, we configure the board pinouts for the ADC pin PA1 (found from the schematic file) as follows:    Then, from “*Analog” -> “ADC1”* we configure the parameters (after enabling IN1), setting the External Trigger Conversion Source as TRGO, that will be set soon from the Timer settings of the TIM2. Also the sampling time is set to 480 cycles:    And enabling the interrupt:    For the timer, we set the usual values (timeout parametrized by the constant TEMPO), plus we enable the TRGO to trigger the callback function of the ADC (Trigger Event Selection as *Update Event*):    Finally, we configure the UART, for remote transmission of the values read by the potentiometer to the remote MATLAB console, as done and explained in the previous homework (using DMA communication):        Then in the “main.c” file we initialized the TIM 2 base generation in interrupt mode and the ADC in interrupt mode with these functions:    We declared two global variables:   * *buffer* will contain the final string to send; * *voltage* will contain the voltage readed by the ADC after a proper conversion acted by the reading variable (from range 0/4096 to 0/3.3V).   We implemented the ADC callback function as follows:    The *snprintf()* function, given *buffer* and itssize*,* sets the buffer to contain the value of our measurement, truncated by 3 decimal positions. The length will contain the number of characters parsed in our buffer. Finally, we transmit the buffer with Direct Memory Access through our *uart2* interface.  Switching to MATLAB, we can now run the script “UART\_read\_data.m” to read the voltage values at baud rate 115200 bps (as set on our board).  **Part 2c:**  The pinout configuration is very similar to the previous project (part 2b). We just disable the USART pins (PA2, PA3) and enable as GPIO\_Output the PA 4-5, PB 1-2-12-13-14-15 pins.    The ADC and timer settings are exactly the same as part 2b.  Then we initialize the LCD following the procedure of the previous homework: we copied the “PMDB16\_LCD.c” file into the project/Core/Src folder and the “PMDB16\_LCD.h” file into the project/Core/Inc folder. Then we added this line in our “main.c” file:    We initialize the ADC, the timer and the LCD as well:    Then we modify the ADC interrupt callback as following:    Respect to the part 2b version we delete the UART communication line and we add a few others, aimed to implement the required lcd behavior. At the beginning we refresh the lcd screen with *lcd\_clear().* Then we define the *bar* variable that is needed to convert the ADC reading (0/4096 value) into a value compatible with the one required by the *lcd\_drawBar()* function (0/80 value). Since the division might generate a float result, a casting is also needed. In order to smoothen out the bar update rate, the TEMPO variable can be decreased. The *snprintf()* function parses the voltage in the *lcd\_buffer*: “Voltage: 1.789 V” for example. The *lcd\_println()* function prints the lcd buffer string on the display showing the exact voltage generated by our board’s potentiometer truncated by 3 decimal positions.  Immagine che contiene testo, elettronica, Ingegneria elettronica, luce  Descrizione generata automaticamente | | | |
| Professor comments: | | | |