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

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *A2* | | |
| Homework number: | *04* | | |
| Due date: | 13/10/2024 | | |
|  |  |  |  |
| 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 1:**  Firstly, we’ve configured the board pinouts and the DMA settings from the graphical interface of the CUBE IDE as follows:  Immagine che contiene testo, schermata, Carattere, diagramma  Descrizione generata automaticamente  Then, from “*Connectivity” -> “USART2”* we configure the parameters:        The DMA transfer completeinterrupt (USART2 global interrupt) is required to indicate the end of the transmission. If the interrupt is not enabled, subsequent transmissions might not be triggered properly, as experienced on our boards.  Secondly, we set the Timer2 interrupt:    Then we passed to the “main.c” file. We defined the constant TEMPO to personalize the speed of UART transmissions (TEMPO of 1000 results in a htim2.Init.Period of (TEMPO\*10) – 1, hence every second an interrupt is generated from the TIM). We also initialized the TIM 2 base generation in interrupt mode with this function:    We declared two global variables:   * *buffer* will contain the final string to send; * *voltage* will simulate an increasing behavior of a float value.     We implemented the timer callback function as follows:    The *snprintf()* function, given *buffer* and itssize*,* sets the buffer to contain the value of our “voltage increasing simulator” (exponentially by factor 1.2), 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\_plot\_data.m” to plot, in a window of 10 seconds and with baud rate 115200 bps (as set on our board), the received voltage values:    Here the reset button of our board has been pressed on the 5th second.  We accomplished our goals:  - To avoid the CPU retrieving data from memory. We instead performed Direct Memory Accessing: this allowed us to ignore the setting of a Timeout period during a normal transmission (with the *HAL\_UART\_Transmit\_DMA*function, which ensures that the CPU doesn't wait indefinitely); there’s no worry about blocking the CPU, hence no timeout is needed.  - To have a non-blocking behavior, by using timers (instead of the *HAL\_Delay* function).  - To allow MATLAB to successfully receive data from our board through the UART interface.  **Part 2:** Firstly, we’ve configured the board pinouts from the graphical interface of the CUBE IDE as follows. We set PA4/5 and PB1/2/12/13/14/15 as GPIO\_Output.    We have also set the TIMER2 as we did for the **Part 1** project. Then we imported the PMDB16\_LCD library in our project in the following way: 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:    Then we declared:   * a pointer to an array of char that contains the names of the members of the group; * an integer representing the number of members; * an integer that indicates the number of the row of the LCD where we want to write; * an integer representing a flag that will be used when the first name of the list is written for the first time (better explained later); * an integer containing the index of the array of char;     We initialized the LCD and turn it on in the *main()* function using the following lines:    Then we used the *HAL\_TIM\_PeriodElapsedCallback()* function to write new lines on the LCD every second. We used the function *lcd\_clear()* to clear the lines written on the LCD. Then we check if the flag *flag\_first\_round* (initialized to the value of ‘0’) is ‘0’ or ‘1’:   * If the value is ‘0’ then we proceed to write on the LCD the name of the first member of the group in the second row of the LCD (this case happens only once at the beginning of the execution) and we set flag\_first\_round to ‘1’ ; * If the value is ‘1’ we write in the first row the member's name that was written at the bottom row at the previous cycle. Then we check if the name\_index has arrived at the maximum value (number of members-1), if so we change its value to ‘-1’ to write the first name of the list on the bottom row, otherwise we do nothing. Lastly, we write the person's name corresponding to the position of name\_index+1 on the second row and increase the index (name\_index++) for the next cycle. In this way when the last member's name is written in the first row the name of the first member is written on the second one. | | | |
| Professor comments: | | | |