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

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *A2* | | |
| Homework number: | *06* | | |
| Due date: | 27/10/2024 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| La Barbera Marco |  |  | *x* |
| Lotto Giulio |  |  | *x* |
| Majocchi Tommaso |  |  | *x* |
| Maffezzini Andrea |  |  | *x* |
| Pompilio Matteo |  |  | *x* |
| Notes: none | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | ADC scan using DMA and LDR | | |
| 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 3a:**  First of all, we configure the board pinout for the UART transmission pin PA2 and the ADC pin PA1 (connected to our board’s potentiometer as verified in the schematic file) as follows:    Then, from “*Analog” -> “ADC1”* we configure the analog to digital conversion of the 3 required voltages: potentiometer, temperature sensor and Vref. We enable IN1 (potentiometer) and the two internal channels in this way:      We also configure the ADC settings:    In this way we enable the DMA in circular mode with continuous DMA requests; we set the number of conversions to 3, one for each channel, and the sampling time to 480 cycles. We also want the end of conversion after all 3 channels have been acquired (*EOC flag at the end of all conversions*). Finally, we set the External Trigger Conversion Source as timer 2 TRGO to start the conversion at 1Hz rate.  We enable the interrupt:    For the timer 2, we set the usual values (timeout parametrized by the constant TEMPO), plus we enable the TRGO to trigger the start of conversion of the ADC (Trigger Event Selection as *Update Event*):    Finally, we configure the UART, for remote transmission of the 3 values read by the ADC to the remote MATLAB console, as also done and explained in the previous homework 4 (using DMA):    Then in the “main.c” file we initialized the TIM 2 base generation and the ADC in DMA mode with these functions, where *&buffer\_dma[0]* indicates the address of the first element of the array that the DMA mechanism will populate:    For this purpose, we declared two constants and two global variables:     * *TEMPO* will dictate the frequency of the timer; * *CHANNEL\_COUNT* as the fixed number of concurrent ADC readings and conversions; * *buffer\_uart* will contain the final string to send via UART; * *buffer\_dma* will contain the parsed values read by the ADC after some proper conversions (fixed of size 16 bits so not to interfere with DMA memory locations).   The timer triggers the start of conversion at [1/TEMPO/1000] Hz rate as required, then at the end of conversion the ADC generates an interrupt. We implemented the ADC callback function as follows:    The DMA buffer will already be populated of the appropriate reading and conversions, before we can apply our adjustments:   * all 3 values need to be converted in voltage. *3.3* indicates the full scale range (3.3 V) and *4096* are the resolution steps (2^12 where 12 is the number of bits) * the temperature reading, also need a voltage to degrees conversion formula; we apply the following one with the underlined values:     The *snprintf()* function, given *buffer\_uart* and itssize*,* parses the specified values in the specified formats in the buffer. Finally, we transmit the buffer with Direct Memory Access through our *uart2* interface, with its characters length.  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): as in the previous homework, we successfully receive the data on our console:  **Part 3b:**  The pinout configuration is very similar to the previous project (part 3a)... | | | |
| Professor comments: | | | |