|  |  |
| --- | --- |
| 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 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 initialize the TIM2 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 declare 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 as not to interfere with DMA memory locations).   The timer triggers the start of conversion at [if TEMPO=1000, then 1] Hz rate as required, and 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 a baud rate of 115200 bps (as set on our board): as in the previous homework, we successfully receive the data on our console:    **Part 3b:**  We configure the board pinout for the UART transmission pin PA2 and the ADC pin PA0 as follows:    Then, from “*Analog” -> “ADC1”* we configure the analog to digital conversion of the Light Dependent Resistor. We enable IN0 and configure the ADC settings in this way:    We enable the DMA in circular mode with continuous DMA requests.    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 LDR and LUX values to the remote MATLAB console using DMA:    We also configured the needed intrrupts:    Then, we pass to the “main.c” file where we include the library math.h that will be used to calculate the lux parameter. We also define several variables like the max and min voltage (*VDDA* and *VSSA*), the number of bits used in the adc (*resolution*\_*bits*), the value *TEMPO* used to set the timer period, and the length of the buffer that will be used to store the values captured from the ADC.    We also initialized some variables that were used for various purposes:   * *FSR* contains the Full-Scale Range (difference between VDDA and VSSA); * *resolution*\_*steps* contains the levels needed to represent the resolution bits; * *in0\_data* is an array of values that contains the value captured by the adc from IN0. Its length depends on the value of *buffer\_length*; * *sum* contains the sum of all *in0*\_*data*, updated to the last call of the ADC Callback function; * *average contains* the average of the *in0*\_*data* values computed as *sum*/*buffer*\_*length*; * *average\_volt* is the conversion in volt of the average value; * *ldr* is the light detection resistance value; * *lux* is the brightness value; * *string* and *sting*\_*length* are just variables used to send the values converted into a string to the remote terminal using the UART.     In the “main.c” file we initialize the TIM2 base generation and the ADC in DMA mode with the functions in the image; *in0\_data* is the pointer to the buffer where the DMA mechanism will store the data.  Then we set the ADC FSR and resolution values.    We use the *HAL\_ADC\_ConvHalfCpltCallback()* function to compute the sum of the first half of the values contained in *in0\_data* buffer.    Then in the *HAL\_ADC\_ConvCpltCallback()* function we compute the sum of the other half of the captured values. We also compute the average, the conversion in voltage, and finally the resistance and the lux values. Then we use the function *snprintf()* as in other projects before to convert the values into a string. At last, we send them via UART to the remote terminal. All this mechanism happens once every second.    Running the script “*UART\_read\_data.m*” we receive the LDR and LUX values on our console.  The following image shows the acquisition with different light sources. We start from darkness, where the resistance reaches 2 MΩ, and then we move to a very bright light source, where it can be seen that the resistance abruptly decreases to a few ohms. | | | |
| Professor comments: | | | |