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

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *B5* | | |
| Homework number: | *05* | | |
| Due date: | Tuesday, November 7th, 08:30 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| Ghidini Alessandro |  |  | *x* |
| Latino Francesco |  |  | *x* |
| Luppi Eleonora |  |  | *x* |
| Bravin Riccardo |  |  | *x* |
| Feltrin Elia |  |  | *x* |
| Notes: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | ADC with DMA for Multiple Conversions and LDR acquisition | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Completed |
|  |  |  | *x* |
| Explanation:  Performing reads through the DMA of sensors both on the micro and the PCB so not to slow down possible processing performed by the main code. In the first part the acquisition of three different data channels (Potentiometer, internal temperature sensor and internal Vref) is performed, while in the second part the LDR value is first read and then converted into a lux value. All the obtained values are shown via UART communication.  **Part 5a:**  First, we enable PA1 as ADC1\_IN1. As we want to do three conversions instead of one, we enable three different channels for the ADC: the potentiometer channel (*IN1*), *Temperature Sensor Channel* as well as *Vrefint Channel*.  In EOC Selection, we select the EOC flag to be raised at the end of all conversions. Also, we enable the Scan Conversion Mode, which is used to scan a group of analog channels.  Since we want acquisitions each one second and we cannot reach this sampling frequency through the ADC, the conversion needs to be started via software, so the Continuous Conversion Mode stays disabled, and we set External Trigger Conversion Source as regular launched by software.  Furthermore, the execution is done using DMA function so in the DMA settings we enable the DMA Request for the ADC in circular mode and in ADC parameter settings we enable DMA continuous request.  Other parameters to be set are the number of conversions equal to 3 and the sampling frequency: we have chosen 480 cycles for the potentiometer and for the Vrefint, while in the case of the temperature sensor, we select 144 cycles to have a frequency bigger than the minimum sampling time of the sensor we found in the datasheet.  Finally, in the NVIC we enable ADC1 global interrupt and we set its preemption priority to 1.  We generate the code.  We define an array *DataBuffer[3]* of size 3, that is the buffer in which ADC saves the conversions. After the initialization of all the peripheral and the start of the ADC in DMA mode through the function *HAL\_ADC\_Start\_DMA(&hadc1, DataBuffer, 3)*, we want to exploit the callback function of the ADC to obtain each conversion from the buffer cells’ values.  Therefore, in ***HAL\_ADC\_ConvCpltCallback****(ADC\_HandleTypeDef\* hadc)* we take each acquired from the corresponding cell of the buffer and we convert and save the correct value in the variables *potvoltage*, *temperature* and *Vref*, corresponding to the cell 0, 1 and 2 of the buffer respectively. In particular:   * *potvoltage* contains the value obtained by applying a voltage divider based on GreenPCB specifications. * *temperature* stores the value as specified by the conversion contained in the file *06 – ADC.pdf* (also available on datasheet specifications). * *vrefint* stores the value as specified by the datasheet, so as a value in the range .   After the conversion has been performed, we raise a flag to print values.  In the while, if the flag is raised, we use the function *HAL\_UART\_Transmit(&huart2, string, StrLength, 100)* to print the string containing the conversions, after having copied them in *string* and determined the final length through the *snprintf()* function. We reset the flag to zero, we call again the DMA mode with *HAL\_ADC\_Start\_DMA(&hadc1, DataBuffer, 3)* and we set a delay of 1 second using *HAL\_Delay(1000)*. Then the cycle starts again.  Obviously, we can trigger ADC with the timer setting properly the parameters to substitute the Delay function.  **Part 5b:**  In the GUI we set PA0 as ADC1\_IN0. In this second project we want to use a timer to trigger the conversion. So, we enable TIM3 with internal clock source. We set all the parameters to have a frequency equal to 1kHz since we want to do a conversion every ms. Also, we select from the timer parameter the trigger event selection as Update Event.  After having initialized the DMA request in circular mode as before, we enable DMA continuous request and set the EOC flag to be raised at the end of single channel in the parameter settings of the ADC. In external trigger conversion we set Timer 3 trigger out event.  Finally, in the NVIC tab we enable ADC global interrupt with preemption priority equal to 1 and TIM3 global interrupt.  We generate the code.  First, we define an array of uint16\_t 1000 cells long to contain the conversion of the photoresistor. We start the timer in interrupt mode and the ADC in DMA mode.  Once the *HAL\_ADC\_ConvHalfCpltCallback* is called we read the first 500 samples from the array and perform a partial average so as not to run into variable overflow problems.  Once the *HAL\_ADC\_ConvCpltCallback* function gets called we read the remaining 500 samples, convert the average to the correct lux unit and print it using the UART communication function. We then reset the average to start the new measurements in the next acquisition. | | | |
| Professor comments:  Part 5a: Better never to use HAL\_Delays (blocking function). Better to use TIM interrupts for this task. | | | |