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

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *B01* | | |
| Homework number: | *HOMEWORK 5* | | |
| Due date: | 07/11/2023 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| Francesco Scroccarello |  |  | *x* |
| Paolo Galfano |  |  | *x* |
| William Stucchi |  |  | *x* |
| Giada Silvestrini |  |  | *x* |
| Francesco Tranquillo |  |  | *x* |
| Notes: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | Potentiometer, internal temperature, Vref and then LDR | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Completed |
|  |  |  | *x* |
| Explanation:  We successfully completed the homework.  **Part 5a:**  First, in the GUI editor we set PA1 to ADC1\_IN1 to enable the reading of the potentiometer.  In Analog -> ADC1 we enabled Temperature sensor channel and Vrefint channel (IN1 already enabled when we set the pin PA1) to read the internal temperature sensor and Vref and pass those values to the ADC directly.  In configuration -> parameter settings we set:   * ADC\_regular\_conversion\_mode -> number of conversions to 3 as the values we want to pass to the ADC; in the rank sections, we assigned the respective channels as channel 1, Channel Vrefint and Channel Temperature sensor, and we modified the number of cycles to 480. * ADC\_regular\_conversion\_mode -> External Trigger Conversion Source as Timer 3 Trigger Out Event, this is to avoid HAL\_Delay in the while, blocking the operations of the microcontroller.   This timer will set the clock for the ADC operations.   * ADC\_Setting -> Scan Conversion Mode as Enabled * ADC\_Setting -> DMA Continuous Requests as Enabled   In DMA Settings we added a configuration with Mode->Circular and Priority->High (even if with priority low it will work anyway since there are no interferences), leaving Data Width->Half Word and in the code we used uint16\_t as type of the variable(array) reading the ADC values.  In the Timers tab we enabled TIM3 and set Clock Source->Internal Clock, Prescaler->8400-1 and Counter Period->10000-1 to give it a 1 second period. The trigger event selection is set to Update Event.  In the code:   * Outside of the while loop we started the ADC in DMA mode passing as parameter the array on which we will find the results of the computation of the ADC. Then we started the Timer responsible for synchronizing the ADC conversion. * in the while loop we checked the value of an auxiliary variable that is set in the ConvCpltCallback: when the callback is called by the end of the ADC conversion, we set this variable to 1 to signal when we can start to process the ADC conversion. Since this variable is checked in an if, the while is not blocked until the end of the conversion, but it can continue to cycle. * in the if, we retrieve the results of the conversion of the ADC, converting them to voltage values with the function V = value \* 3.3 / 4096.0 * additionally, for the temperature value, we applied the conversion formula T = ((V – 0.76) / 0.0025) + 25 * finally, we built the message string and transmitted it to the remote terminal.   **Part 5b:**  First, in the GUI editor we set PA0 to ADC1\_IN0 to enable the reading of the light dependent resistor.  In the Analog->ADC we can see that mode IN0 is already set, se we don’t have to do anything.  In configuration -> parameter settings we set:   * DMA continuous requests as Enabled. * ADC\_Regular\_ConverisionMode->number of conversions can be left to 1 as we are interested only in IN0. * ADC\_regular\_conversion\_mode -> External Trigger Conversion Source as Timer 3 Trigger Out Event, this is to avoid HAL\_Delay in the while, blocking the operations of the microcontroller.   This timer will set the clock for the ADC operations.   * The rank section can be left as it is, while we can adjust the number of cycles to 480.   In DMA Settings we added a configuration with Mode->Circular and Priority->High (even if with priority low it will work anyway since there are no interferences), leaving Data Width->Half Word and in the code we used uint16\_t as type of the variable reading the ADC values.  In the Timers tab we enabled TIM3 and set Clock Source->Internal Clock, Prescaler->8400-1 and Counter Period->10-1 to give it a 1 millisecond period. The trigger event selection is set to Update Event. This timer is directly connected to the ADC and will be used to give the clock to it.  We enabled also TIM1. This clock is used to synchronize the output to the remote terminal of the mean of the ADC computation. After every second passes, the HAL\_TIM\_PeriodElapsedCallback will be called, and we will send to remote terminal the result of the computation.  In the code:   * Outside the while loop we started the ADC in DMA mode, passing to the function the variable where the ADC will store the results (memory location we will read through that variable).   We started also TIM3 in non-interrupting mode, since it is the ADC timer, and TIM1 in interrupting mode because we will use its interrupt callback to send messages to remote terminal every 1 second.   * In the HAL\_ADC\_ConvCpltCallback: we compute the mean incrementally, though the use of a global counter variable. * In the HAL\_TIM\_PeriodElapsedCallback: we convert the result of the ADC’s computation to a voltage value, then we compute its resistive value (ldr variable) and its lux value.   The computation is then sent to remote terminal. | | | |
| Professor comments:  Part5b: better to use DMA to directly store 1000 values, and average them at the completed callback. (You will see during the lab how to optimize it not to lose data). | | | |