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

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *A2* | | |
| Homework number: | *HOMEWORK 05* | | |
| Due date: |  | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| Hui Jiang |  |  | *x* |
| Mattia Sironi |  |  | *x* |
| Gabriele Landi |  |  | *x* |
| Arturo Caliandro |  |  | *x* |
| Luigi Lizzini |  |  | *x* |
| Notes:  none | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name |  | | |
| Not done | Partially done   (major problems) | Partially done   (minor problems) | Completed |
|  |  |  |  |

|  |
| --- |
| We have done the whole homework, here is our explanation:  **Project 3a:**  Firstly,we have configured the board as shown below:  Which is actually identical to the projects 2b and 2c.  Secondly,we have enabled the three channels de ADC as shown below:  And then we configured the ADC as shown below:  Where we have set the “DMA Continuous Requests” to Enabled, “Number of Conversion “ to 3 as we have to measure three channels in this project, “External Trigger Conversion Source” to Timer 2 Triggrt Out event as we will use the timer2 to control the measureing frequency, and in these 3 Rank section, set their channel to Channel 1, Channel Temperature Sensor, Channel Vrefint respectively and each Sampling Time to 480 Cycles in order to give the ADC to finish the measurement.  Actually,before set the “DMA Continuous Requests” to Enabled, we first need to go to DMA settings and add a DMA request as shown below.otherwise you would find that you can’t set the “DMA Continuous Requests” to Enabled:  Where we have set the Mode to Circular because in this project we need to sample those three datas every s, and in order to save those samples, we have created a array of size 3, with mode Circular, the DMA knows in the next iteration, save the first sample again in the first element of the mentioned array. And if we use the mode Normal, when the DMA saved all the datas in the first iteration, the DMA will stop transmit data to the memory if we do not restart the DMA, which means this mode will make our implementation much more complicated, therefore we choose the Mode Circular. And we have also set the Data Width to Half Word(16 bits) as our ADC’s sample has a size of 12bits(and also our array is a array of uint16\_t), thus in order to optimize the efficiency, we have set it to Half Words.  Then, we have configured the Timer2 in order to make the ADC sampling frecuency to 1Hz:  As you can see, the configuration is exactly the same as project2b and 2c.  After doing that, we went to the NVIC table:  Where we have enabled manually ADC1 global interrupt,TIM2 global interrupt and USART2 global interrupt.  Until this point, we have done all the configuration on the GUI, then we started programming in the main.c:  we firstly created a global array of uint16\_t in order to save the samplings:  In the main, we have start both Timer2 and ADC in a safe way:  Then, we have implemented the callback function:  In this function,we just transforms the data offered by the ADC to the appropriate form and send them using UART2. The temperature fucntion is from the slide:  And the code works as we expected:  We can compare this temperature to the temperature of the thermometer in the room of Hui: Imagen de la pantalla de un celular con letras  Descripción generada automáticamente con confianza baja  Therefore, we know that the internal temperature sensor of the board is not very precise.  Note: Maybe you have noticied that the exercise said that the acquisition is started by software, and actually this is our first version:  Which requires the HAL\_Delay function which is a operation kind of dangerous, so we decided to use timer(the version we have showed so far.)  **Project 3b:**  Firstly, we have configured the board as shown below:    Where we have set the PA0 which is the pin connected to LDR to ADC1\_IN0.  Here is the procedure to find the pin connected to LDR:  In green board schematics.pdf we have find that LDR is connected to 14:  Then we went to the nucleo schematics.pdf:  We found 14 is connected to PA0.  Then we have configured the ADC1 exactly the same as the previous project but in this project we just enabled the channel IN0:  And then we have set the timer2 as shown below:  Where we set the Prescaler to 0 and Period to 83999 as (0+1)\*(83999+1)/83e6 = 1e-3 s = 1ms which is the required sampling frequency. Other settings are exactly the same as the previous projects(2b 2c and 3a).  After doing that, we went to the NVIC table in order to enable the following interrupts:  Until this point, we have done all the configurations on the GUI, then we went to the main.c, we first define a macro and a array to save the samples:  You may ask why the array has a size of 2000 instead of 1000(sampling frequency 1000Hz, send the average every 1s, therefore the size 1000 is sufficient). It is due to the method that the professor provided during the lecture, we will explain it when we get the appropriate point.  In the main, we just start both the timer2 and ADC in a safe way:  Then we created two callback functions, and both of them do the exactly the same thing:  We finally arrive the point where we explain why the array size is 2000: in this project, we firstly encounted a case where we need to process the massive data. Let’s first image what would happen if we set the array size to 1000: when the DMA finished its job of filling the array, the callback function start to process those data, but in this moment, as the CPU is processing these data, DMA can’t put new data in the array. Which means a low effciency. In our project, once the DMA has filled the first half of the array, the first callback function is called, and start to process the firstly half of the array, at the meantime, the DMA is filling the second half of the array with new data. Then, when the whole array is filled by the DMA, the second callback function is called and start to process the second half of the array and once again, in the meantime, the DMA is filling the first half of the array with new datas. So, in our case, we don’t need to wait DMA to fill the array as the array size 1000 case we have mentioned. So with this method(proponed by the professor during the lecture), we obtained a high efficiency. And the way we calculate the LDR and LUX is from the slide:  Finally, the code works as we expected: |
| Professor comments: |