Complete the “yellow” tabs and delate the phrases in italics.  
You can duplicate the table “Project”, if more than one project are due for the homework.

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *A1* | | |
| Homework number: | HW09 | | |
| Due date: | 4/12 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| 1 *Giorgio Donato Carlo* |  |  | *x* |
| 2 *Lenzi Francesco* |  |  | *x* |
| 3 *Lodari Gianmarco* |  |  | *x* |
| 4 *Lanzini Alessio* |  |  | *x* |
| 5 *Chiapparo Lenn* |  |  | *x* |
| Notes:  *Complete in necessary* | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | Encoder w/ DMA | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Successfully completed |
|  |  |  | *x* |
| Explanation: In this homework, firstly we set from the .ioc the pin for the encoder (PC6, PC7), the TIM 3 in Encoder mode(division by 4 and Input filter = 15), the TIM2 in interrupt mode every second (PRS = 8400-1 and ARR = 10000-1) and final we have enabled in the USART2 the DMA in normal mode in transmission.  In the main we have started the TIM2 in interrupt mode and the TIM3 in Encoder Mode (both the channels).  We have declared the global variable count\_old and count\_new that take into account the previous value and the next value of the counter that we pick with \_\_HAL\_TIM\_GET\_COUNTER function.  When the interrupt is triggered, in the callback function we compute the delta and we check its value to control the underflow or the overflow. In particular, if delta exceeds the positive FSR (in our case 32768), we subtract the total FSR (in our case 65536) or vice versa, if delta exceeds the negative FSR (in our case -32768), we add the total FSR (in our case 65536).  Once this check is done, we calculate the result knowing that the total number of ticks in a round is 24 and that we want the result in rounds per minute.  Finally, we print the value through the UART in DMA mode. | | | |
| Professor comments:  Very good! | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | Keyboard speculative configuration | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Successfully completed |
|  |  |  | *x* |
| To readout the keyboard we need to perform the following 3 steps:   1. Enable the GPIO pins related to the keyboard:    1. Rows as inputs    2. Colums as outputs 2. Enable the SPI interface, which is connected to the P10 and P11 registers (From Hands-on lab document, page 115). 3. Poll through the columns in the while(1) cycle or with a timer, as shown in the slides. We perform a new column scan every 4ms, reading if the rows are pulled down (button pressed) or not, saving its state in a matrix.    1. If we find that a row value changes (i.e. a button has been pressed “0” or released “1”), we readout again after the column counter overflows, before changing the column selection. If the readout is the same as before, the button has been actually pressed/released, an then we can change its value in our stored matrix, otherwise we had a spurious switching (Bounce) and we don’t modify the matrix.    2. This condition can be easily controlled with some “if-else” statements and implements debouncing. | | | |
| Professor comments:  OK! | | | |