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

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *B01* | | |
| Homework number: | *HOMEWORK 4* | | |
| Due date: | 24/10/2023 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| Francesco Scroccarello |  |  | *x* |
| Paolo Galfano |  |  | *x* |
| William Stucchi |  |  | *x* |
| Giada Silvestrini |  |  | *x* |
| Francesco Tranquillo |  |  | *x* |
| Notes: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | ADC triggered by TIM / ADC triggered by TIM to LCD | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Completed |
|  |  |  | *x* |
| Explanation:  We successfully completed the homework.  **Part 4a:**  In the GUI we set the pin PA1 to GPIO\_Analog and ADC1\_IN1. Then, in the System Core tab we enabled the ADC1 global interrupt.  We moved to the Analog tab where we found ADC1 settings and there we set in the subsections:   * ADC\_Regular\_ConversionMode: External Trigger Conversion Source as Timer 3 Trigger Out event and External Trigger Conversion Edge left to Rising Edge in order to enable a clocked trigger to the ADC; * Rank: we left the channel as Channel 1 since we later modified that channel of the timer 3 and set sampling time 480 cycles;   In the Timers section we modified TIM3 as follows:   * Mode section: Clock Source as internal clock and Channel1 as PWM Generation No Output; * Parameter Settings: Prescaler to 8400-1, Counter Period to 10000-1 (these settings are to set a frequency of 1HZ), Pulse to 5000-1   In the Connectivity tab we modified the Baud rate to 9600 to match Putty settings of COM3 port and we left word length at 8 bits including the parity.  In the main.c file:   * In the main function we started the ADC in interrupt mode using the function HAL\_ADC\_Start\_IT(&hadc1); and then we started the timer using the function HAL\_TIM\_Base\_Start(&htim3); * We overwrote the callback weak function for the ADC called at the end of a complete conversion (HAL\_ADC\_ConvCpltCallback(ADC\_HandleTypeDef\* hadc)) in which we acquired the value from the conversion using the function HAL\_ADC\_GetValue(&hadc1), we manipulated the value ad hoc to build the string to print and we transmitted it to the Putty terminal using the function HAL\_UART\_Transmit.   **Part 4b:**  Starting from what done in the 4a (except from the Connectivity part since we’re not using Putty terminal this time and so we didn’t need the USART setting), we enabled the pins for the LCD (PA4, PB1, PB2, PB12, PB13, PB14, PB15) and we set all of them to GPIO\_Output.  We imported the .c/.h file of the libraries for LCD connectivity, downloaded from Webeep, respectively in the src and inc folders.  In the main.c (the code was almost similar to 4a):   * We included the header file for the LCD functions * Before starting ADC and Timer we initialized the LCD and we turned on the backlight * In the callback, instead of transmitting data via USART, we printed the string on the first row using the function lcd\_println and we used the function lcd\_drawBar passing as input a rescaled version of the voltage acquired to match the [0-80] integer range. | | | |
| Professor comments:  Part A:   * “Channel1 as PWM Generation No Output” 🡪 it can be used in time base (in fact in the code you started it in time based (no need to configure the PWM) | | | |