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

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *B5* | | |
| Homework number: | *3* | | |
| Due date: | Tuesday, October 17, 8:30 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| Ghidini Alessandro |  |  | *x* |
| Latino Francesco |  |  | *x* |
| Luppi Eleonora |  |  | *x* |
| Bravin Riccardo |  |  | *x* |
| Feltrin Elia |  |  | *x* |
| Notes: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | USART and LCD | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Completed |
|  |  |  | *x* |
| Explanation:  In the first project, the goal is to send every second a string with a name and year of birth using USART communication by configuring DMA settings and using a timer to send the desired string with the right timing at its overflow.  The second project focuses on writing team member names on an LCD, scrolling them every second. It requires using LCD functions to display the names and again enabling the timer to scroll them with the correct timing.  **Part 3a:**  In this project we want to send a string containing the name and the year of birth using the USART communication.  First, we setup the pins (in the GUI two pin are set by default) USART\_TX and USART\_RX (PA2 and PA3). Hence, in the connectivity tab we select a Baud Rate of 115200 bit/s and we set the same value in Arduino IDE (or PuTTy) to enable the communication between the transmitter and the receiver. Then, we click on the tab DMA settings, and we add the DMA Request for USART2\_TX, mode Normal.  In the NVIC we enable USART2 global interrupt to enable the data transmission every time HAL\_UART\_Transmit\_DMA() is called.  In addition, we enable a timer, e.g., Timer 1, and set its parameters to have its overflow every second (PSC = 8400-1, ARR = 10000-1) and we enable the TIM1 update interrupt from the NVIC.  We generate the code. Before doing everything, we must include libraries to handle strings correctly:   * **#include** "stdio.h" * **#include** "string.h"   In the main before the while(1) we use the function snprintf() to write in the array of char previously defined, the name and the year of birth.  In the HAL\_UART\_Transmit\_DMA() we give as input: the pointer to a UART\_HandleTypeDef structure which is set as &huart2, the pointer to the data which is the array of char and the size, computed as  size = strlen(string).  Our transmission has to happen every second, so we use the HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim) callback to have 1 second transmission timing. In the callback body, we use an if structure which recognises when the Timer 1 overflow has happened (htim == &htim1) and then we call HAL\_UART\_Transmit\_DMA(&huart2, string, size).  We use the Serial monitor of Arduino IDE (or PuTTY) to visualize transmission, by setting same Baud Rate as in STM32CubeIDE.  **Part 3b:**  In order to write our name in sequence on the LCD we must set up it from the GUI of the IDE by habilitating the correspondent pins we need to control that are LCD\_BL\_ON (activate the back light of the screen), LCD\_RS (select and activate the register), LCD\_E (habilitate the peripheral LCD) and the four data pins (D4, D5, D6 and D7). We can find the correspondence between the LCD pins and the STM32 ones on the PCB and NUCLEO schematics (respectively the pins are PA4, PB2, PB1, PB12, PB13, PB14, PB15). All these pins are set up as GPIO\_Output pins.  In the include part we need to include as in the previous exercise the libraries "stdio.h" and"string.h" to treat the strings properly. To facilitate the managing of the LCD we use a library that offers some useful functions to communicate with the driver at high level automatically performing the transduction at low level to have the desired output. We add this library in the folders main.c and main.h and in the code:   * **#include** "PDMB16\_LCD.h"   In the code we will use a flag (start), a counter variable (i) and an array of string (array of arrays of char) containing our names. The variable start is set initially to 1 to indicate the first accension of the device. In the initialization part of the code, after we have declared the array of string’s elements, we habilitate the used TIMER (TIMER 2) to generate Interrupt Timer. Everytime the Timer goes in overflow, it’s activated an IT whose code is written in the function HAL\_TIM\_PeriodElapsedCallback(\*htim). This fucntion is valid for all the timers so we set a condition regarding the timer used by us with an if construct (if(htim == &htim2)).  First we control if the device has just been turned on: if so, we’ll print the first name contained in the array by using the proper function lcd\_println() and then set the flag to 0. Otherwise, we have an else if statement controlling the index we use to map all the elements of the array. We'll print two names for each different index (i-1th and the ith ). To print the last name and the first name we use a control if (i==sizeof(array) && start == 0) and print these two names. When the index is equal to the length of the array, we reinitialize the index variable to 1.  At the end of these instruction, we increase the value of the index by 1. | | | |
| Professor comments:  Very good! | | | |