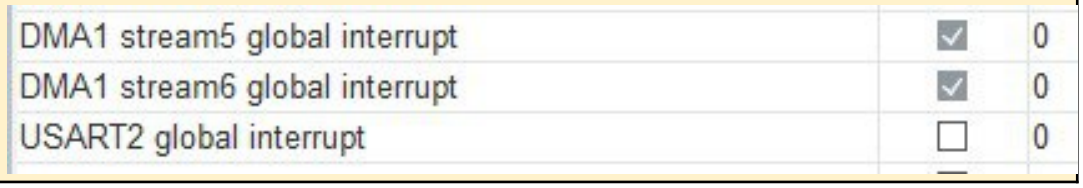
Mark	1/11

Team name:	*B5*		
Homework number:	*HOMEWORK 04*		
Due date:	15/10/24		
Contribution	NO	Partial	Full
Marenghi Manuela			*x*
Fellegara Tommaso			*x*
Giammusso Samuele			*x*
Cattani Luca			*x*
Csata Dániel			*x*
Notes: none			

Project name	Test		
Not done	Partially done (major problems)	Partially done (minor problems)	Completed
			x

Part 1 - Complete the UART project with DMA

- To transfer information using a DMA we have to enable the DMA in the USART2 peripheral using USART2_TX requests:
  - this is done by opening the .ioc file,
  - then select USART2 in the connectivity tab
  - then head to DMA settings
  - and in the end Add USART2_TX
- In the NVIC settings of USART2 is necessary to enable the USART2 global interrupt
  - this is used to handle the communication in an asynchronous way so that it can work with MATLAB
  - We noticed that when using Putty this step is not necessary to obtain a working communication



- In the code check the state of the USART2 and if it is READY, we can transmit the data
    - otherwise it means that it's still in a BUSY state, so it cannot send other data
    - it can be implemented as shown in the code below:

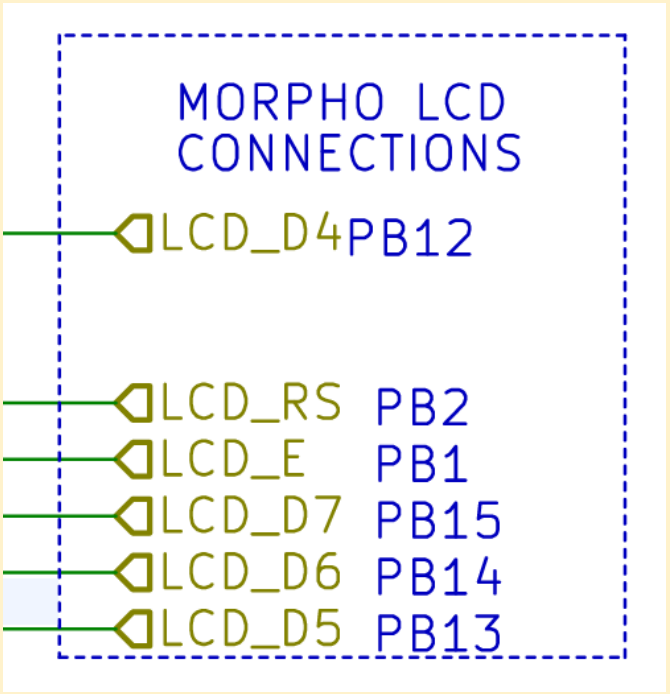
```
102   /* USER CODE BEGIN WHILE */
103   char string[50];
104   int count=0;
105   while (1)
106   {
107       count++;
108       int length = snprintf(string, sizeof(string), "Fausto, 2020, Time: %d\r\n", count);
109
110       if(HAL_UART_GetState(&huart2) == HAL_UART_STATE_READY){
111           HAL_UART_Transmit_DMA(&huart2, string, length);
112       }
113
114       HAL_Delay(1000);
115     /* USER CODE END WHILE */
116
117     /* USER CODE BEGIN 3 */
118   }
119   /* USER CODE END 3 */
120 }
```

- we also added a counter variable to have a clearer view of the communication in the terminal
- Another way to implement the code, which works but doesn't use a good practice, is instead of waiting for the state to become Ready, it can be directly set as Ready right after the transmission

```
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
  HAL_UART_Transmit_DMA(&huart2, name, length);
  huart2.gState = HAL_UART_STATE_READY;
  HAL_Delay(1000);
}
/* USER CODE END 3 */
```

Part 2 - Write on the LCD the name of each member of your group, one per line, in alphabetical order. Scroll every one second

- To properly use the LCD, we set the pin PA4, PB1, PB2, PB12, PB13, PB14 and PB15 to be GPIO_output
- 

- 

CN7 even pins	
Name	**Pin**
PA4	32

- We copied and pasted the “PMDB16_LCD.c” and the “PMDB16_LCD.h”, then in the main.c code we included the library (#include “PMDB16_LCD.h).
    - We implemented the code as shown below:

```c
  /* USER CODE BEGIN WHILE */
  char names[][16] = {"Daniel", "Luca", "Manuela", "Samuele", "Tommaso"};
    int sz = sizeof(names) / sizeof(names[0]);
    lcd_initialize();
    lcd_backlight_ON();
    while (1)
    {
      /* USER CODE END WHILE */

      /* USER CODE BEGIN 3 */
      for(int i = 0; i < sz; i++) {
          lcd_println(names[i], 0);
          lcd_println(names[(i + 1) % sz], 1);

          HAL_Delay(1000);
      }
    }
  /* USER CODE END 3 */
}
```

- Before the while loop we initialized the names variable and the LCD controller using the functions lcd_initialize() and lcd_backlight_ON().
- We wrote a for loop to print the names with the function lcd_println
    - in order not to have out of bound access to the char array, we used the % operator, which can also allow to display in the LCD the first name right below the last name after the completion of a cycle

Professor comments:

Try to avoid blocking functions (such as HAL_Delay), but use timers and interrupts.