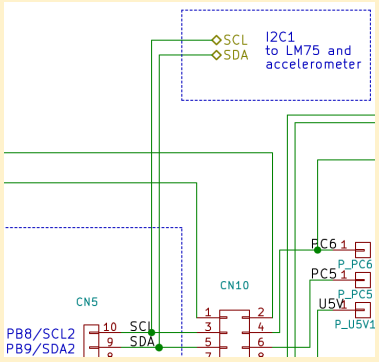
Mark	1/11

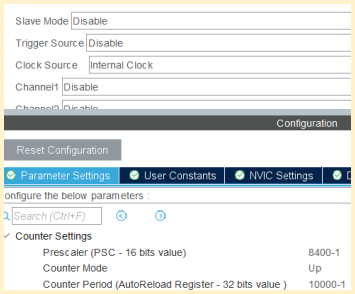
Team name:	*B5*		
Homework number:	*HOMEWORK 08*		
Due date:	19/11/24		
Contribution	NO	Partial	Full
Marenghi Manuela			*x*
Fellegara Tommaso			*x*
Giammusso Samuele			*x*
Cattani Luca			*x*
Csata Dániel			*x*
Notes: none			

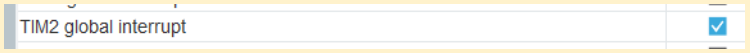
Project name	Test		
Not done	Partially done (major problems)	Partially done (minor problems)	Completed
			x

Part 1b: read accelerometer, use timer, UART DMA

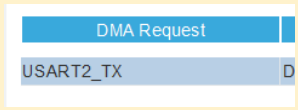
- Setup the pin of the I2C to the accelerometer sensor:
  - SCL is connected to PB8, and SDA is connected to PB9
  - 

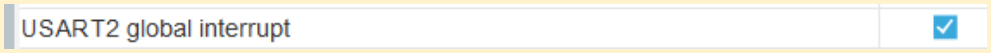
- Setup the I2C by enabling it in the connectivity tab, and keep it in Standard Mode
- Setup the timer T2, so that it operates at 1hz, and generates an interrupt:





- Enable the USART2 DMA and the global interrupt:





- How we implemented the code:
- We used the following global variables. The first two are used to define the accelerator address respectively in write and read mode. acc_address is the standard address of the accelerometer, cnt is a counter, CTRL_REGX variables are used to write in the indicated registers.

```
#define WRITE_PORT(x)  (x << 1) //left shift
#define READ_PORT(x)  ((x << 1) | 1) //left shift and add 1 with the OR operator
```

```
uint16_t acc_address = 0x28;
uint32_t cnt = 0;
```

```
uint8_t CTRL_REG1[] = {0x20, 0b00010111}; //set the power mode
uint8_t CTRL_REG2[] = {0x21, 0b00000000}; //high pass filter normal mode
uint8_t CTRL_REG4[] = {0x23, 0b00000000}; //full scale selection: [-2g,2g]
```

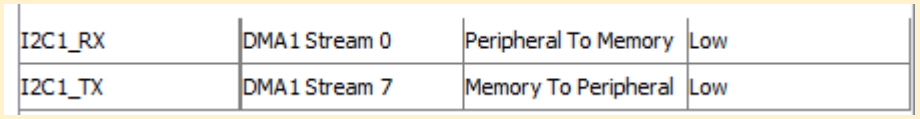
- In the main function we initialize the timer in interrupt mode, we define the accelerometer address according to the type of device, write it in the REG1 and set other flags in REG2 and REG4

```
if(HAL_I2C_Master_Transmit(&hi2c1, WRITE_PORT(acc_address), CTRL_REG1, 2, 10) != HAL_OK) {
 acc_address = 0x18;
     if(HAL_I2C_Master_Transmit(&hi2c1, WRITE_PORT(acc_address), CTRL_REG1, 2, 10) != HAL_OK){
         //err
     }
}

 HAL_I2C_Master_Transmit(&hi2c1, WRITE_PORT(acc_address), CTRL_REG2, 2, 10);
 HAL_I2C_Master_Transmit(&hi2c1, WRITE_PORT(acc_address), CTRL_REG4, 2, 10);

 HAL_TIM_Base_Start_IT(&htim2);
```

- Then we perform the readings in the callback of the timer by using Transmit and Receive one per dimension and then convert it. When all the readings are performed, we send them using the UART interface in DMA mode. For each dimension in order to send the variable we first write the register we want to read and then we read it:

```c
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    if (htim->Instance == TIM2) {
        int8_t reg = 0x29;
        int8_t x, y, z;
        HAL_I2C_Master_Transmit(&hi2c1, WRITE_PORT(acc_address), &reg, 1, 10);
        HAL_I2C_Master_Receive(&hi2c1, READ_PORT(acc_address), &x, 1, 10);
        //divide by 64 because FSR=2^8=256, but the range is from -2g to 2g, so it's 4g, (x/256)*4=x/64
        float send_x = (float)x / 64.0f;

        reg = 0x2B;
        HAL_I2C_Master_Transmit(&hi2c1, WRITE_PORT(acc_address), &reg, 1, 10);
        HAL_I2C_Master_Receive(&hi2c1, READ_PORT(acc_address), &y, 1, 10);

        float send_y = (float)y / 64.0f;

        reg = 0x2D;
        HAL_I2C_Master_Transmit(&hi2c1, WRITE_PORT(acc_address), &reg, 1, 10);
        HAL_I2C_Master_Receive(&hi2c1, READ_PORT(acc_address), &z, 1, 10);

        float send_z = (float)z / 64.0f;

        char str[64];
        int len = snprintf(str, 64, "%d: Accelerations: x:%+.2f g y:%+.2f g z:%+.2f g \r\n", cnt, send_x, send_y, send_z);
        cnt++;

        if( HAL_UART_GetState(&huart2)==HAL_UART_STATE_READY ){
          HAL_UART_Transmit_DMA(&huart2, str, len);
      }
    }
}
```

Part 1c: read accelerometer, use timer, UART DMA, I2C DMA

- The setup of the UART interface and timer are the same as before
- The setup of the I2C:
    - Enable the I2C DMA both for transmission and reading:
    - 

I2C1_RX	DMA1 Stream 0	Peripheral To Memory	Low
I2C1_TX	DMA1 Stream 7	Memory To Peripheral	Low

    - enable the interrupt:
    - 

- How we implemented the code:
    - In the main function the initialization is the same as above. Also global variables used are the same except for the data[] array that is used to store all values of the accelerometer.
    - ```c
      int8_t data[5]; // 0x2D - 0x29 = 5 are the bytes to read
      ```
    - We added also a define: MSB is a byte with the most significant bit set to 1
      ```c
      #define MSB 0x80 // 1000'0000
      ```
    - In this solution we're using 3 callback functions:
      -Firstly, we have the timer callback that is the one that triggers the Transmit on the accelerometer where we're writing the first register to read.
      -Then, the callback about the Transmit for receiving data that just stores them in the array data[] .
      -Finally, a callback for the Receiving that converts them and sends using the UART interface in DMA mode.
      ```

```c
void HAL_I2C_MasterTxCpltCallback(I2C_HandleTypeDef *hi2c) {
    HAL_I2C_Master_Receive_DMA(&hi2c1, READ_PORT(acc_address), &data, sizeof(data));
}

void HAL_I2C_MasterRxCpltCallback(I2C_HandleTypeDef *hi2c) {
    float x, y, z;
    x = data[0]/64.0f;
    y = data[2]/64.0f;
    z = data[4]/64.0f;

    char str[64];
    int len = snprintf(str, 64, "%d: Accelerations: x:%+.2f g y:%+.2f g z:%+.2f g \r\n", cnt, x, y, z);
    cnt++;

    if( HAL_UART_GetState(&huart2)==HAL_UART_STATE_READY ){
      HAL_UART_Transmit_DMA(&huart2, str, len);
  }
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    if (htim->Instance == TIM2) {
        int8_t reg = 0x29 | MSB; //add 1 in the MSB to enable the auto-increment
        HAL_StatusTypeDef status;

        HAL_I2C_Master_Transmit_DMA(&hi2c1, WRITE_PORT(0x28), &reg, 1);
    }
}
```

Professor comments: