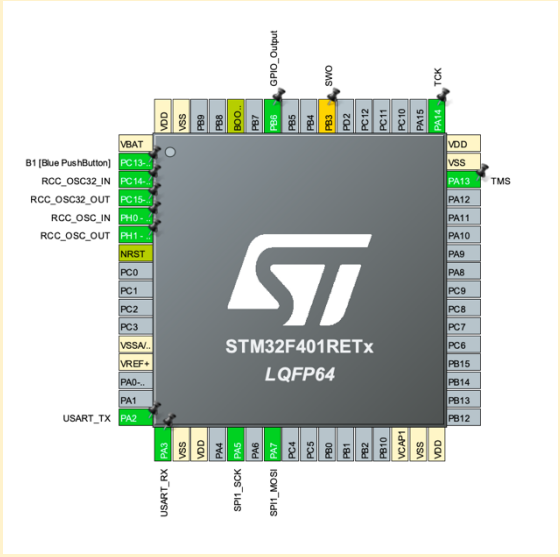
Mark	/11

Team name:	*A1*		
Homework number:	*HOMEWORK 09*		
Due date:	26/11/24		
Contribution	NO	Partial	Full
Piombo			*x*
Fumagalli			*x*
Pierfederici			*x*
Zenoni			*x*
Ferraro			*x*
Notes:			

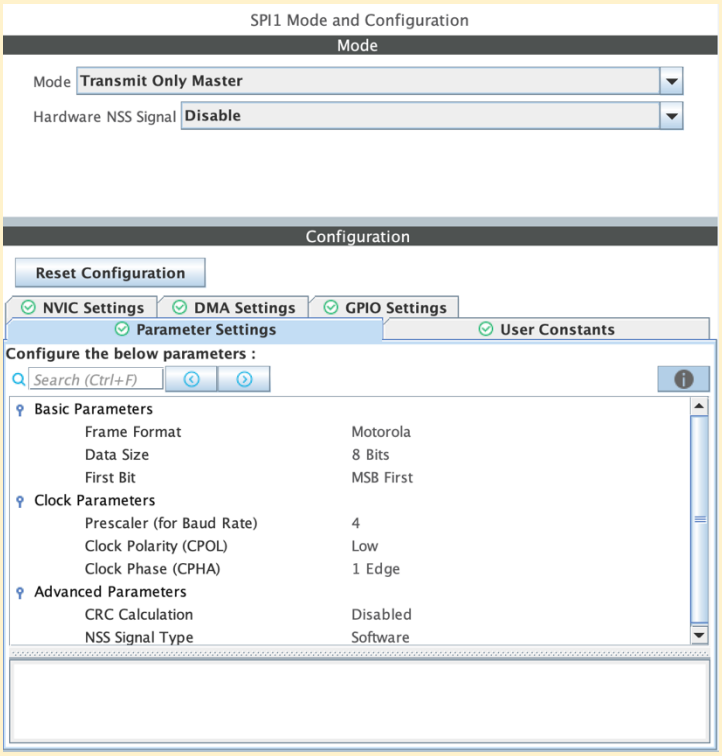
Project name	SPI – LED MATRIX		
Not done	Partially done (major problems)	Partially done (minor problems)	Completed
			x

Part 1:

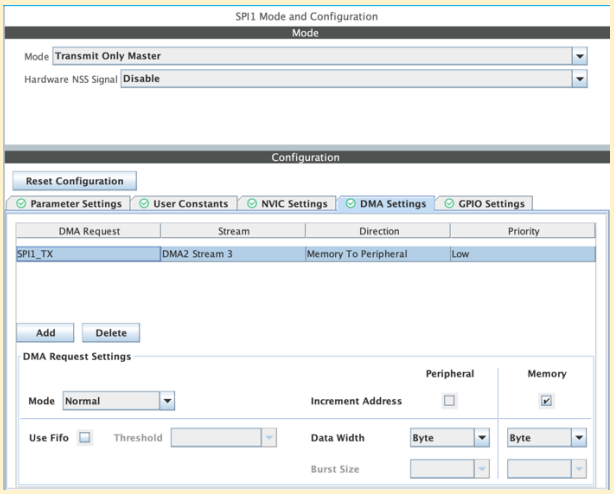
Starting from the ".ioc" we enabled the SPI1 in "Transmit Only Master" mode so that SCK (PA5) and MOSI (PA7) pins are automatically set. Then we set the SS (PB6) pin, which is connected to the RCLK of the shift registers that drive the LED matrix, as GPIO_Output.

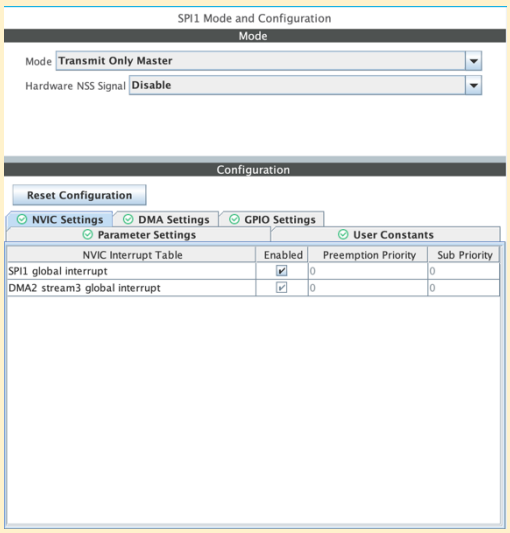


We also set the SPI prescaler as 4 to avoid the bug.

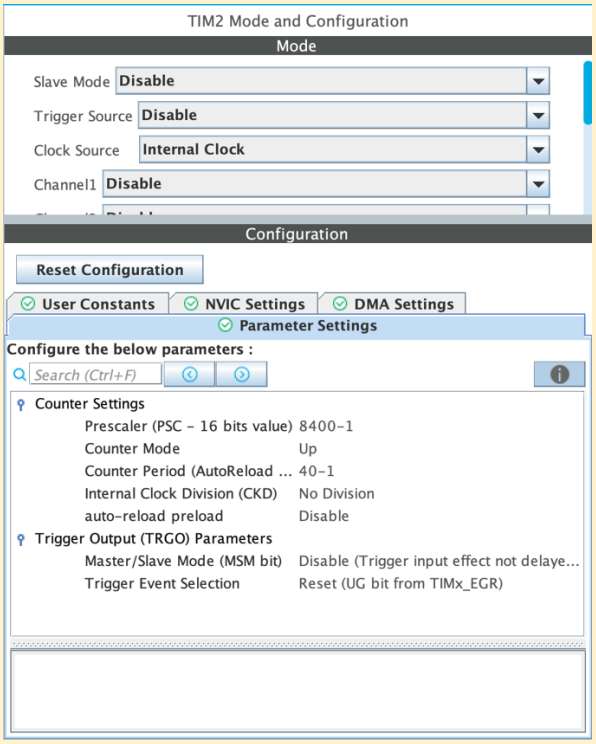


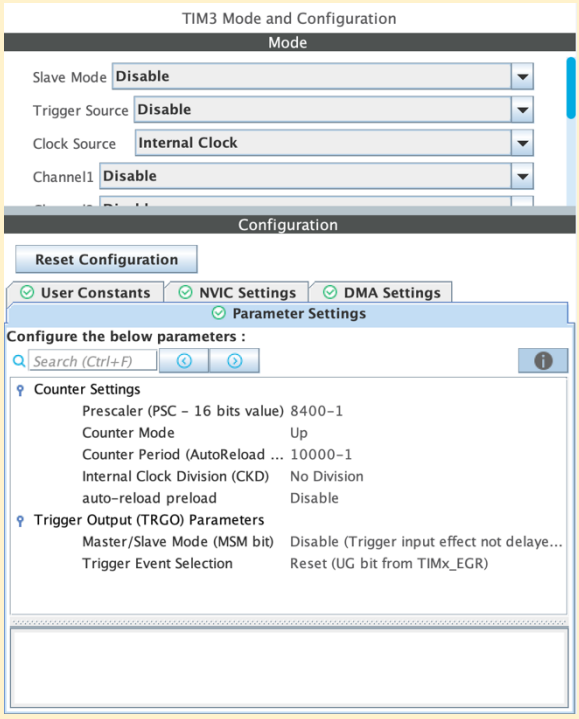
We set the SPI in DMA transmit mode and we enbled its interrupt.





In order to keep our ISR as simple as possible we decided to exploit two timers: TIM2 counts 4ms to refresh the matrix's columns and TIM3 counts 1s to change the letter displayed. We also enabled their interrupts.





Regarding the "main.c", we use the following defines and variables:

```
35 #define NUM_OF_CHAR 2
36 #define MATRIX_ROW 5
37 #define MATRIX_COL 2
38 #define RCLK GPIOB,GPIO_PIN_6   //RCLK is connected to the SPI pin SS (page 3 of hands-on lab schematics)
58 uint8_t char_index = 0; //index to decide which char to send
59 uint8_t i = 0;          //index of the row of the char matrix (to change the LED matrix column)
60
61 uint8_t tx_char[NUM_OF_CHAR][MATRIX_ROW][MATRIX_COL] = {{   //3D array to store all the char (matrix) we have to send
62         {49, 16},   //letter s matrix
63         {73, 8},
64         {73, 4},
65         {73, 2},
66         {70, 1}
67 },{
68         {64, 16},   //letter t matrix
69         {64, 8},
70         {127, 4},
71         {64, 2},
72         {64, 1}
73 }};
74
75 /* USER CODE END PV */
76
```

We used a single 3D array containing all the values needed to display the characters (S, T) in the LED matrix. The first index is used to choose the desired character.

Each character is saved as 5 pair of bytes to drive 5 columns. Each pair is saved as: {LED_row, LED_column} so that we transmit first the row data to the column shift register that then pass the data to the row shift register, thanks to the Daisy Chain configuration.

In the main function we started both timers in interrupt mode.

```
152   /* USER CODE BEGIN 2 */
153
154   __HAL_TIM_CLEAR_IT(&htim2, TIM_IT_UPDATE);    //clear interrupt request BEFORE enabling tim interrupt
155   __HAL_TIM_CLEAR_IT(&htim3, TIM_IT_UPDATE);    //clear interrupt request BEFORE enabling tim interrupt
156     HAL_TIM_Base_Start_IT(&htim2);              //start TIM2 in interrupt mode
157     HAL_TIM_Base_Start_IT(&htim3);              //start TIM3 in interrupt mode
158
159   /* USER CODE END 2 */
```

These are our callbacks. Every 4ms TIM2 sends the two bytes used to drive one column of the LED matrix. Every second TIM3 changes the “char_index” to change the displayed letter.

When the transmission is over, we provide a positive edge to the RCLK pin to update the output.

```
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef *htim) {

    if (htim == &htim2) {   //timer 2 callback called every 4ms
        if (i == MATRIX_ROW)    //index rollback
            i = 0;
        //send correct char
            HAL_SPI_Transmit_DMA(&hspi1, &tx_char[char_index][i++], 2);
    }

    if (htim == &htim3) {   //timer 3 callback: 1 second elapsed -> change letter
        char_index++;
        if (char_index == NUM_OF_CHAR)  //char index rollback
            char_index = 0;
    }
}

void HAL_SPI_TxCpltCallback(SPI_HandleTypeDef * hspi) {     //SPI transmission completed
    if (hspi == &hspi1) {
        //provide a rising edge to RCLK in order to update the output shift registers
        HAL_GPIO_WritePin(RCLK, GPIO_PIN_SET);
        HAL_GPIO_WritePin(RCLK, GPIO_PIN_RESET);
    }
}

/* USER CODE END 0 */
```