

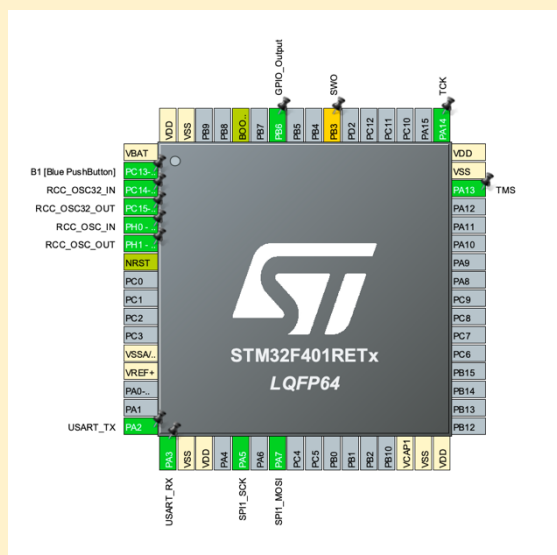
Mark	/11
------	-----

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.

SPI1 Mode and Configuration

Mode

Mode: **Transmit Only Master**

Hardware NSS Signal: **Disable**

Configuration

Reset Configuration

NVIC Settings DMA Settings GPIO Settings

Parameter Settings User Constants

Configure the below parameters :

Search (Ctrl+F)

Basic Parameters

Frame Format: Motorola

Data Size: 8 Bits

First Bit: MSB First

Clock Parameters

Prescaler (for Baud Rate): 4

Clock Polarity (CPOL): Low

Clock Phase (CPHA): 1 Edge

Advanced Parameters

CRC Calculation: Disabled

NSS Signal Type: Software

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

SPI1 Mode and Configuration

Mode

Mode: **Transmit Only Master**

Hardware NSS Signal: **Disable**

Configuration

Reset Configuration

Parameter Settings User Constants NVIC Settings DMA Settings GPIO Settings

DMA Request	Stream	Direction	Priority
SPI1_TX	DMA2 Stream 3	Memory To Peripheral	Low

Add Delete

DMA Request Settings

Mode: **Normal**

Increment Address: ☐

Peripheral: ☐

Memory: ☒

Use Fifo: ☐ Threshold:

Data Width: **Byte**

Burst Size:

SPI1 Mode and Configuration

Mode

Mode: **Transmit Only Master**

Hardware NSS Signal: **Disable**

Configuration

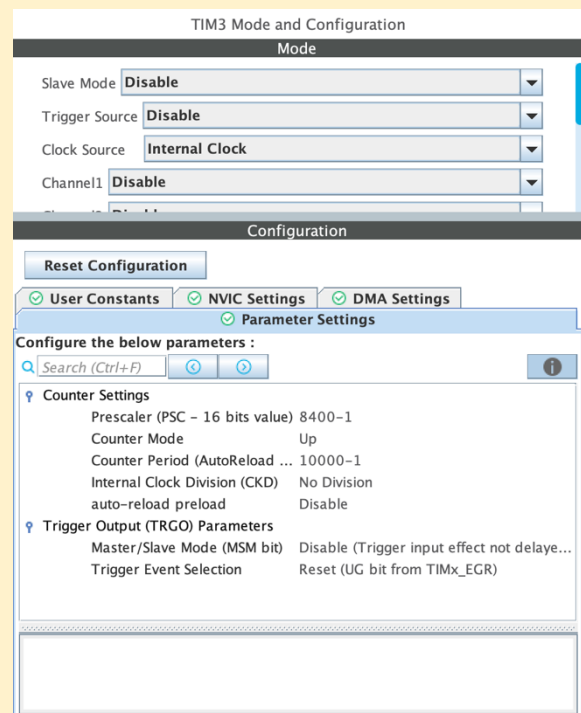
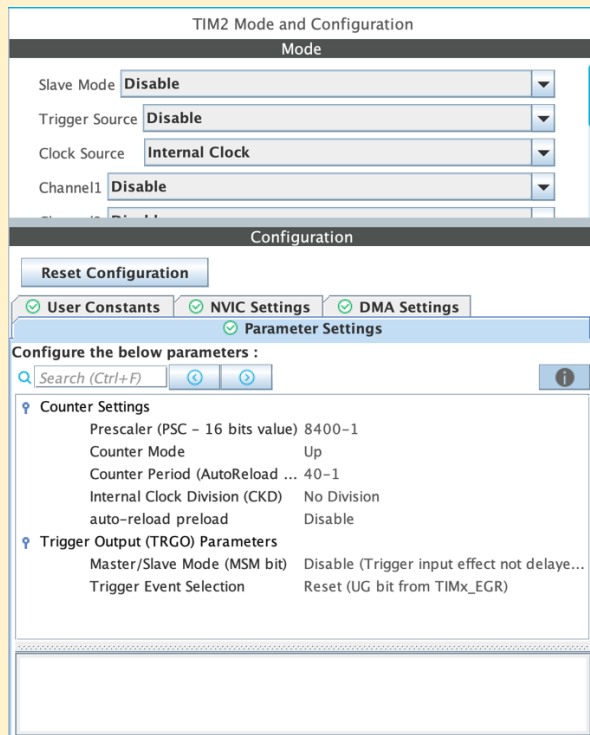
Reset Configuration

NVIC Settings DMA Settings GPIO Settings

Parameter Settings User Constants

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
SPI1 global interrupt	<input checked="" type="checkbox"/>	0	0
DMA2 stream3 global interrupt	<input checked="" type="checkbox"/>	0	0

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)
39
40 uint8_t char_index = 0; //index to decide which char to send
41 uint8_t i = 0; //index of the row of the char matrix (to change the LED matrix column)
42
43 uint8_t tx_char[NUM_OF_CHAR][MATRIX_ROW][MATRIX_COL] = {{ //3D array to store all the char (matrix) we have to send
44     {49, 16}, //letter s matrix
45     {73, 8},
46     {73, 4},
47     {73, 2},
48     {70, 1}
49 },{
50     {64, 16}, //letter t matrix
51     {64, 8},
52     {127, 4},
53     {64, 2},
54     {64, 1}
55 }};
56
57 /* USER CODE END PV */
58
59
60

```

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.

```
89  /* Private user code -----*/
90  /* USER CODE BEGIN 0 */
91
92  void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef *htim) {
93
94      if (htim == &htim2) { //timer 2 callback called every 4ms
95          if (i == MATRIX_ROW) //index rollback
96              i = 0;
97          //send correct char
98          HAL_SPI_Transmit_DMA(&hspi1, &tx_char[char_index][i++], 2);
99      }
100
101      if (htim == &htim3) { //timer 3 callback: 1 second elapsed -> change letter
102          char_index++;
103          if (char_index == NUM_OF_CHAR) //char index rollback
104              char_index = 0;
105      }
106  }
107
108  void HAL_SPI_TxCpltCallback(SPI_HandleTypeDef * hspi) { //SPI transmission completed
109      if (hspi == &hspi1) {
110          //provide a rising edge to RCLK in order to update the output shift registers
111          HAL_GPIO_WritePin(RCLK, GPIO_PIN_SET);
112          HAL_GPIO_WritePin(RCLK, GPIO_PIN_RESET);
113      }
114  }
115
116  /* USER CODE END 0 */
```