Mark	**/11**

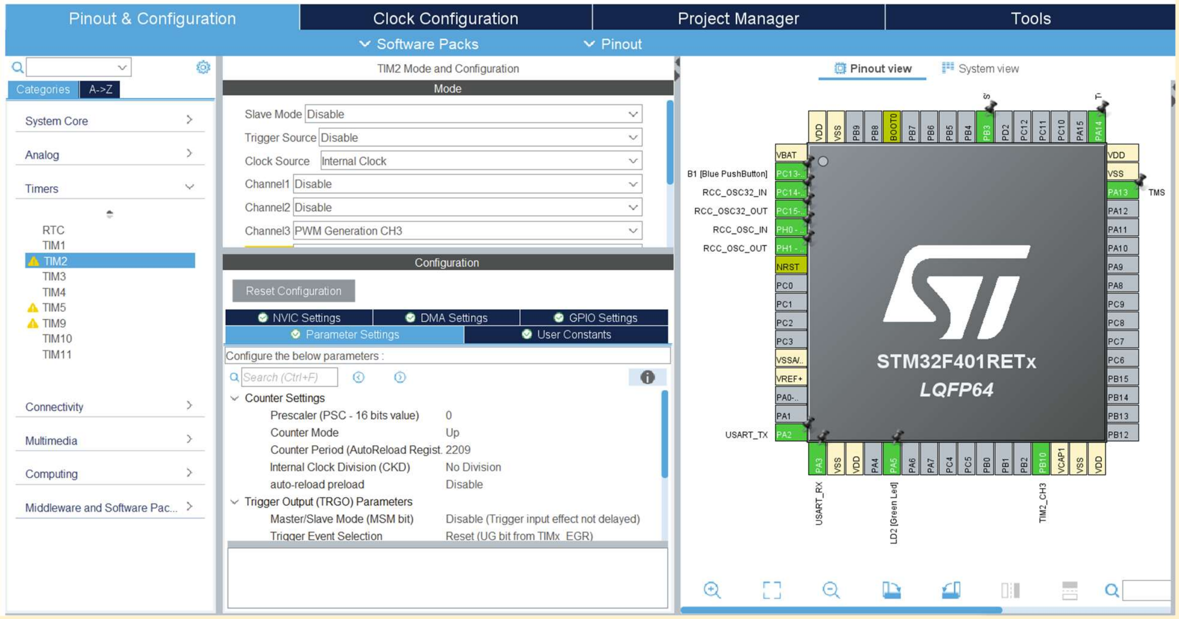
Team name:	*A14*		
Homework number:	*HOMEWORK 11*		
Due date:	30/11/25		
Contribution	NO	Partial	Full
Mattia Di Mauro			*x*
Francesca Biondi			*x*
Lorenzo Castelli			*x*
Carmen Maria Niro			*x*
Pietro Albrigi			*x*
Notes: none			

Project name	HOMEWORK 11		
Not done	Partially done (major problems)	Partially done (minor problems)	Completed
			x

Project 2: Transmit board

The .ioc configuration for the keyboard interface remained unchanged from the previous project, as the same GPIO and timer settings were reused.

For the IR transmission, Timer 2 Channel 3 (connected to the IR LED on pin PB10) was configured in PWM Generation mode. The timer parameters were set to PSC = 0, ARR = 2209, and Pulse = 1104, which produce a PWM signal at 38 kHz with a 50% duty cycle:



TIM2 Mode and Configuration

Mode

Slave Mode Disable

Trigger Source Disable

Clock Source Internal Clock

Channel1 Disable

Channel2 Disable

Channel3 PWM Generation CH3

Configuration

Reset Configuration

NVIC Settings | DMA Settings | GPIO Settings | Parameter Settings | User Constants

Configure the below parameters :

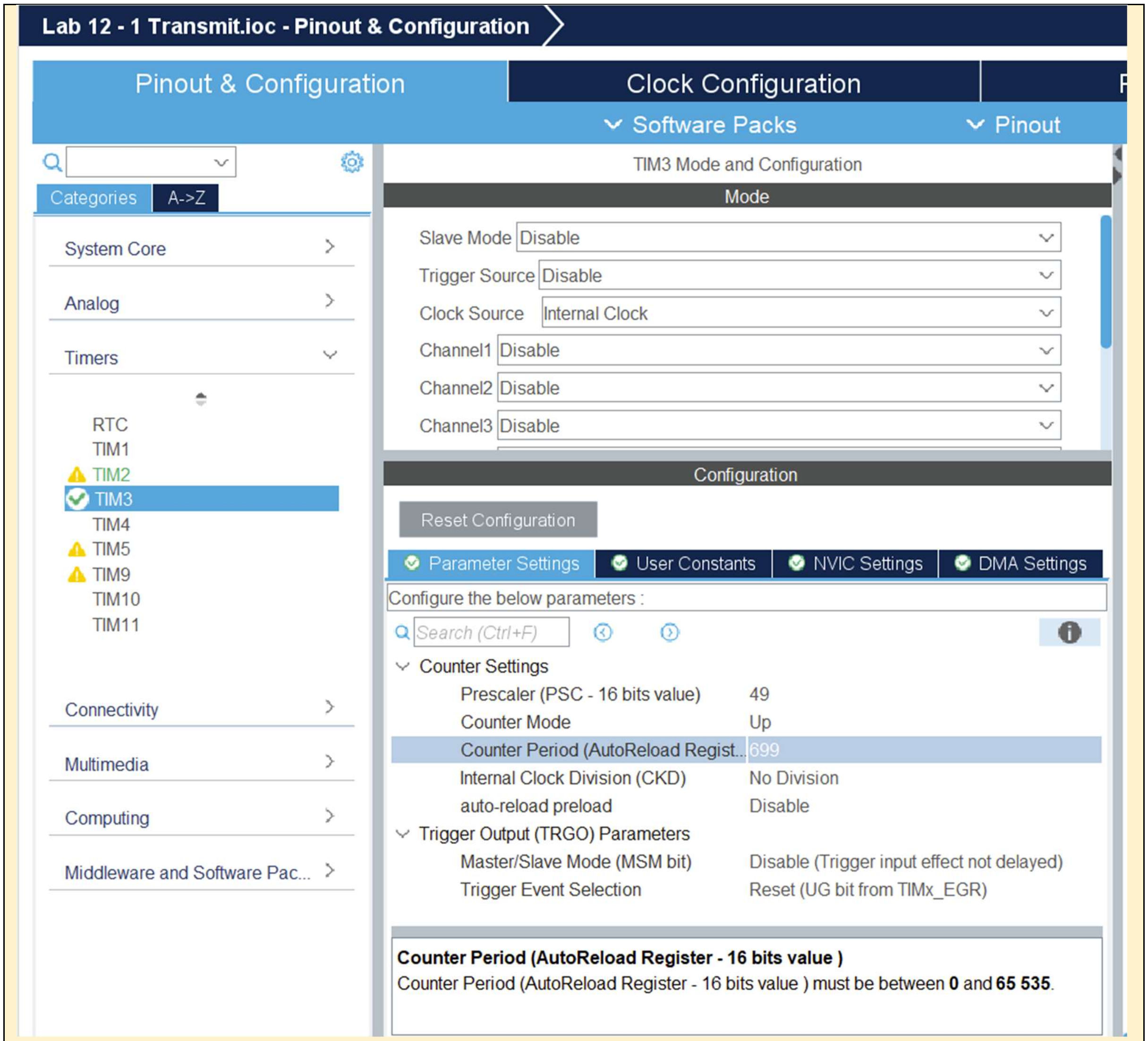
Counter Settings

```
Prescaler (PSC - 16 bits value)        0
Counter Mode                           Up
Counter Period (AutoReload Regist.     2209
Internal Clock Division (CKD)          No Division
auto-reload preload                    Disable
```

Trigger Output (TRGO) Parameters

```
Master/Slave Mode (MSM bit)            Disable (Trigger input effect not delayed)
Trigger Event Selection                Reset (UG bit from TIMx_EGR)
```

Timer 3 was then configured with an appropriate prescaler and auto-reload value to generate an interrupt every 1/2400 s (i.e., 2.4 kHz). The update interrupt was enabled so that the timer could provide a precise timing reference for transmitting individual IR data bits.



TIM3 Mode and Configuration

Mode

Slave Mode: Disable
Trigger Source: Disable
Clock Source: Internal Clock
Channel1: Disable
Channel2: Disable
Channel3: Disable

Configuration

Configure the below parameters :

Counter Settings

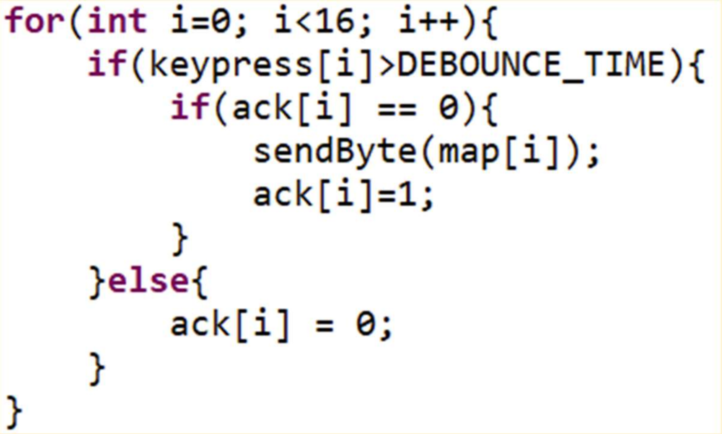
Prescaler (PSC - 16 bits value)	49
Counter Mode	Up
Counter Period (AutoReload Regist...	699
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable

Trigger Output (TRGO) Parameters

Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection	Reset (UG bit from TIMx_EGR)

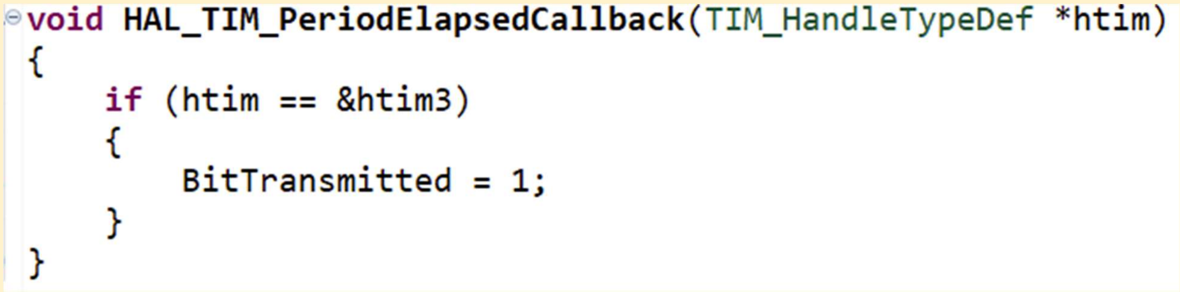
Counter Period (AutoReload Register - 16 bits value)
Counter Period (AutoReload Register - 16 bits value) must be between **0** and **65 535**.

The code used to scan the push-button matrix was identical to that of the previous projects. When a key press is detected, and after it remains stable long enough to satisfy the software debouncing logic, the program invokes the sendByte function. This function transmits the corresponding character from the map associated with the pressed key.



```
for(int i=0; i<16; i++){
    if(keypress[i]>DEBOUNCE_TIME){
        if(ack[i] == 0){
            sendByte(map[i]);
            ack[i]=1;
        }
    }else{
        ack[i] = 0;
    }
}
```

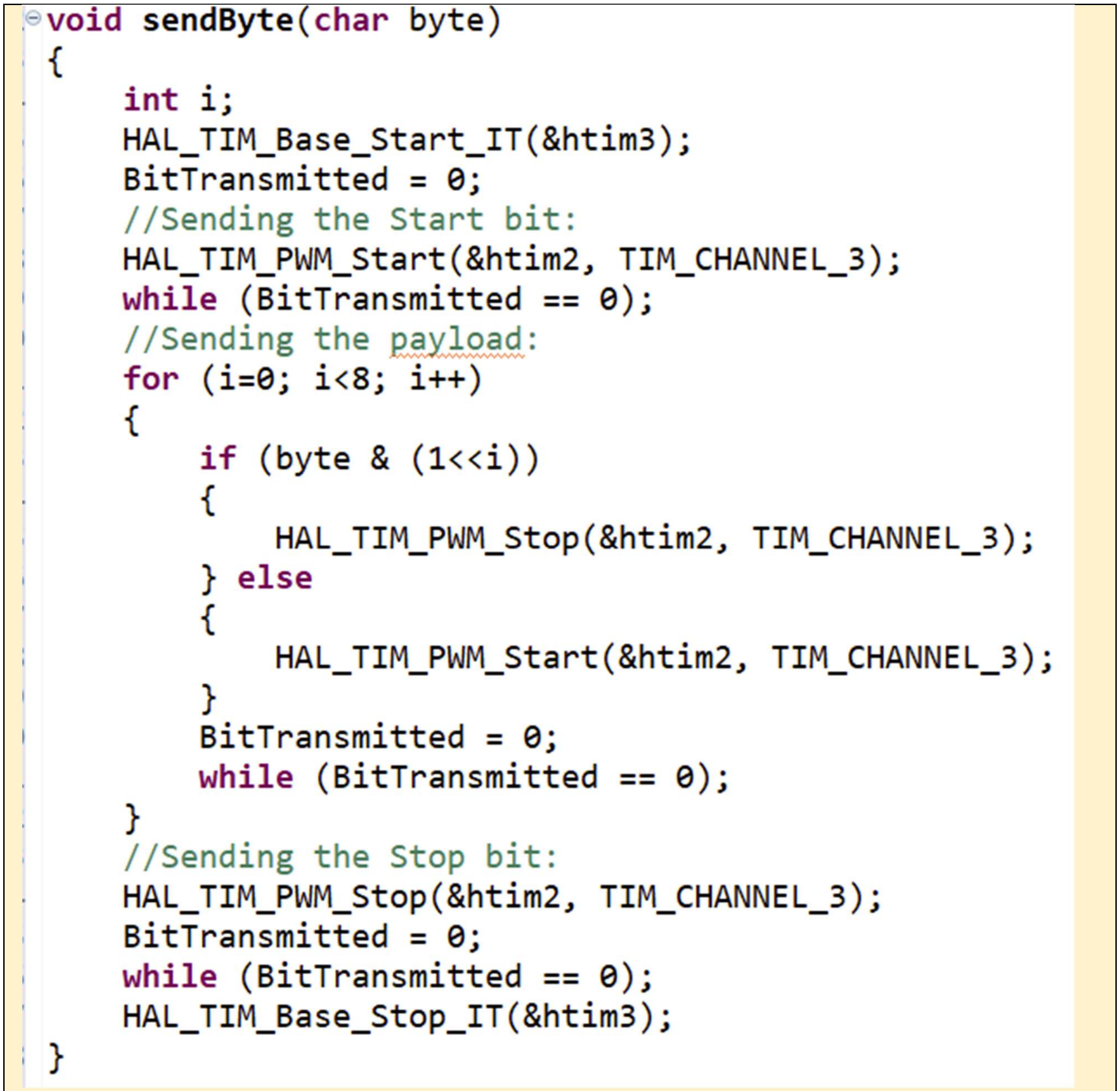
For the sendByte function, we introduced a global variable (BitTransmitted) that acts as a synchronization flag for timing each transmitted bit. At the beginning of every bit period, the flag is set to 0 when the PWM is started or stopped depending on the bit value. The flag is then set to 1 inside the TIM3 period elapsed callback, which is triggered every 1/2400 s. This mechanism signals that the duration allocated for the current bit has ended, allowing the function to proceed with transmitting the next bit.



```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if (htim == &htim3)
    {
        BitTransmitted = 1;
    }
}
```

Inside the function that sends the byte, TIM3 is first started in interrupt mode, and the synchronization flag is initialized to 0. The Start bit is transmitted by enabling the PWM output on TIM2 Channel 3. A blocking while loop ensures that execution does not proceed until the TIM3 interrupt (described previously) sets the flag to 1, signaling that one bit period (1/2400 s) is over.

Inside a for loop, the eight payload bits are transmitted in the same manner: each bit of the input byte is inspected, and the PWM output is started or stopped depending on its value. After all eight bits have been processed, the Stop bit is transmitted, and the timer used for bit-timing is then disabled.

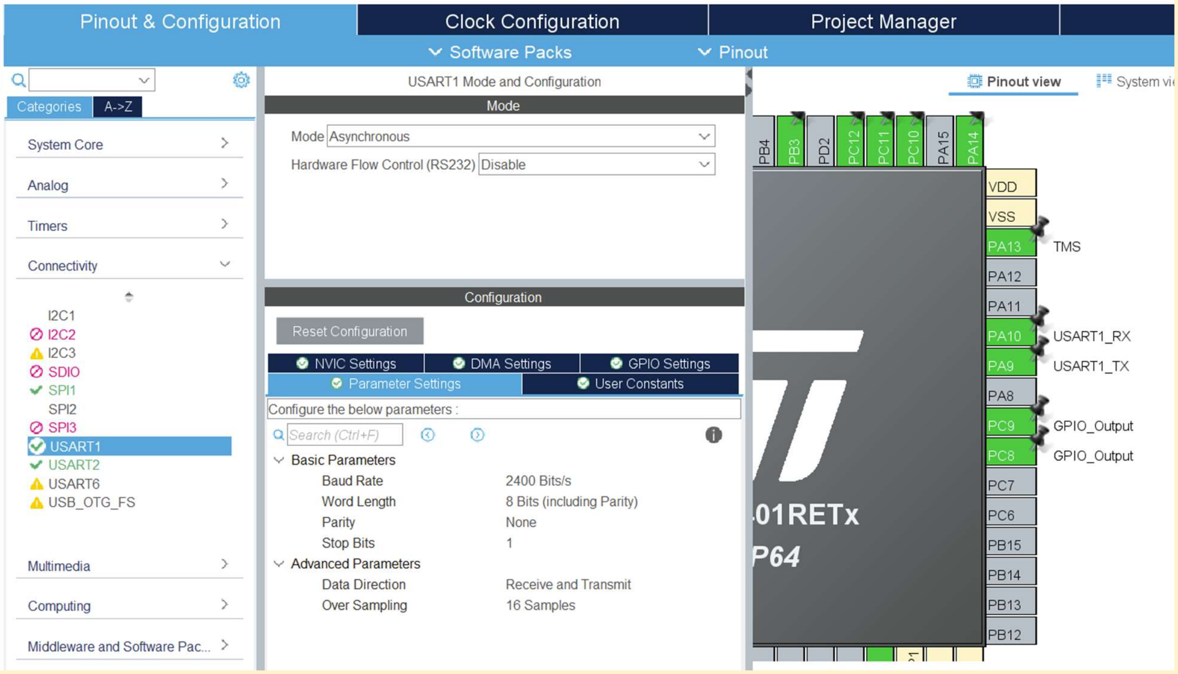


```
void sendByte(char byte)
{
    int i;
    HAL_TIM_Base_Start_IT(&htim3);
    BitTransmitted = 0;
    //Sending the Start bit:
    HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_3);
    while (BitTransmitted == 0);
    //Sending the payload:
    for (i=0; i<8; i++)
    {
        if (byte & (1<<i))
        {
            HAL_TIM_PWM_Stop(&htim2, TIM_CHANNEL_3);
        } else
        {
            HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_3);
        }
        BitTransmitted = 0;
        while (BitTransmitted == 0);
    }
    //Sending the Stop bit:
    HAL_TIM_PWM_Stop(&htim2, TIM_CHANNEL_3);
    BitTransmitted = 0;
    while (BitTransmitted == 0);
    HAL_TIM_Base_Stop_IT(&htim3);
}
```

Project 2: Receive board

For this part of the project, the code used to display characters on the LED matrix was identical to that of the corresponding project, using character maps stored in an array.

We configured USART1, which is connected to the IR receiver module, with a baud rate of 2400 bps and set it to receive one 8-bit word at a time. Its global interrupt was enabled, and the corresponding pins (PA10 for RX and PA9 for TX) were configured.



USART1 Mode and Configuration

Mode

Mode Asynchronous

Hardware Flow Control (RS232) Disable

Configuration

Configure the below parameters :

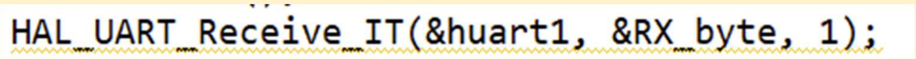
Basic Parameters

Baud Rate	2400 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1

Advanced Parameters

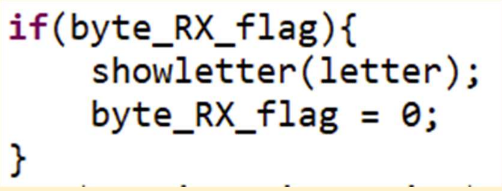
Data Direction	Receive and Transmit
Over Sampling	16 Samples

In the main function we receive the first byte :



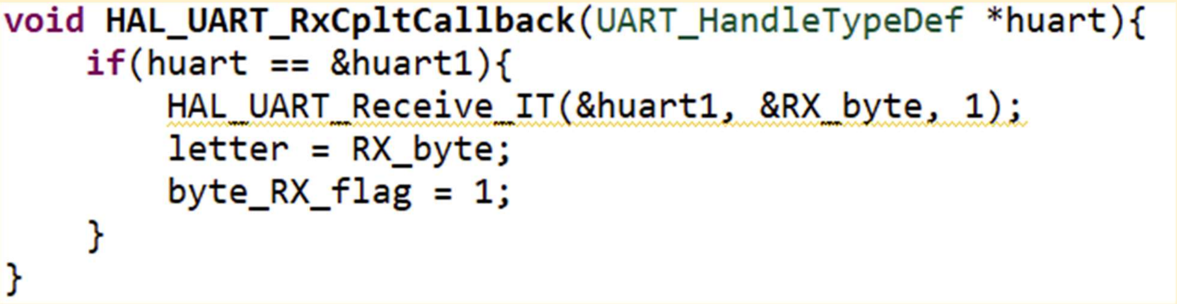
```
HAL_UART_Receive_IT(&huart1, &RX_byte, 1);
```

In the while loop, the selected character is displayed on the LED matrix only when a dedicated flag is set.



```
if(byte_RX_flag){
    showletter(letter);
    byte_RX_flag = 0;
}
```

This flag is updated inside the UART complete callback, which is triggered automatically at the end of each received byte. In the same callback, a new HAL_UART_Receive_IT call allows the peripheral to receive the next incoming byte.



```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
    if(huart == &huart1){
        HAL_UART_Receive_IT(&huart1, &RX_byte, 1);
        letter = RX_byte;
        byte_RX_flag = 1;
    }
}
```

For Project 3, the two previously developed modules (the code for IR transmission and the code for IR reception with LED-matrix display) were integrated into a single code so that the same board is capable of both transmitting and receiving IR data.

Professor comments: