|  |  |
| --- | --- |
| **Mark** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *A2* | | |
| Homework number: | *HOMEWORK 10* | | |
| Due date: | 10/12/23 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| Hui Jiang |  |  | *x* |
| Mattia Sironi |  |  | *x* |
| Gabriele Landi |  |  | *x* |
| Arturo Caliandro |  |  | *x* |
| Luigi Lizzini |  |  | *x* |
| Notes: none | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | IR communication between two boards | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Completed |
|  |  |  | *x* |
| Explanation:  We successfully completed the homework.  Next, we explain all the steps for completing the homework:  Project 1:  Transmitter:  Firstly, we have configured the board as shown below:  Where we have just set the PB10(which is connected to the IR LED) to TIM2\_CH3.  Then, we have configured the timer 2 as shown below in order to generate PWM in its channel 3:  Where we have put the Counter Period to 2210(2211/84e6 == 1/38e3). And we put Pulse to 1105 in order to have a 50% duty cycle.  In order to set the baud rate to 2400bps, we have configured the timer 3 as shown below:  Finally, in the GUI, we enabled the following interrupt in the NVIC table:  In the main, we first defined a variable to indicate a bit is finished transmitted(0 no,1 yes):  Then we created a function in order to just send one byte:  In this function, we first start the timer 3 in interrupt mode(timer 3 is the timer we set to control the baud rate). Then we first send the start bit 0, then we wait until the bit is finished sending using the while. The variable bitfinished will be changed to 1 in the callback function of timer:  So every 1/2400 second, the bitfinished will be set to 1 which corresponds to the baud rate 2400bps.  When the start bit is finished, we stop the PWM. With the same way, we send the 8 bits of the byte in the for loop. Finally, we send the stop bit in the same way and we stop the timer 3.  After doing that, we creadted another function to send the message using the above function:  Finally, in the main, we defined the message we want to send in this first project and send it continously in the infinite while:    And this end our explanation of the transmitter of project 1.  Receiver:  First, we have configured the board as shown below:  Where we have just configured the PA9 and PA10 to USART1\_TX and USART1\_RX respectively. The UART2 is configured by default.  Then,we have configured the USART1 as shown below:  Where we have set the Baud Rate to 2400 as requested and Data Direction to Receive Only.  And we just keep the default settings of UART2:  As requested to use the interrupt mode of USART1 and USART2, we went to NVIC table to enable them:  In the main.c, we first define a global variable o receive the data:    Then, in the main, we just make the USART1 to receive data in interrupt mode:    When this operation is finished, its corresponding callback function will be called(and the furture same operations will also):    In its callbaclk function, we just send the receive byte using USART2 to our PC. Then make USART1 to receive the next byte.  Finally, it works as we expected:  But we just need to put the two boards very close.  Project 2:  In this project, instead of using LED matrix, we decided to use the LCD.  Transmitter:  first, we have configured our board as shown below:  The only difference with the previous transmitter is that we added the configuration for the button matrix.  Then, we have configured the timer2,3 and 10 as shown below:        The timer 2 and 3 are for the same purpose of the previous transmitter project. We use the timer10 in order to control the scanning frequency of the button matrix.(what we have done in the homework09)  Finally in the GUI, we have enabled the following interrupts:  As you may notice, we have changed the priority of the timer 3 to 1 which means a lower priority. We will explain it later.  As the previous homework, we defined the following macros:  And the same global variables for the same purposes(even with the same name):    And we have also defined the same OUR\_UART\_SendByte function:    In the main, we do the same thing as the homework09 in order to scan the first column of the button matrix before the first iteration of the callback function:    Then we defined the callback function for the two timers:    The timer 3 we used to control the baud rate while the timer 10 to scan the button matrix and when detect a button is pressed(also after the debounce time), we send the symbol represented by the pressed button using the function Our\_UART\_SendByte. But here is the problem, we call the function Our\_UART\_SendByte when the callback function of timer10 is called, howecer in the function Our\_UART\_SendByte, we also initialize the timer 3 in interrupt mode, and when every bit is finished sending, will call the callback function of timer3. We have tried that if we put the two interrupts in the same priority, after press the first button, the board stop working.That is because in this case, when we pressed the fisrt button, timer 3 is activated and then generated interrupt but at the meantime, timer10 is still counting and in its intterupt mode, and both with same priority, something wrong happens. Therefore, in order to solve this problem, we set the a lower priority for the timer10 interrupt.  Then, we finished our explanation of the transmitter of the project 2.  Receiver:  we fisrt configured our board as shown below:  in comparison with the previous receiver, we have just add the pin configuration for the LCD.  Then we configured the USART1 as shown below:  Exactly the same as the previous receiver.  In order to use the USART1 in interrupt mode, we enabled its interrut in the NVIC table:  In the folder of the project, we added the files for the LCD:    And in the main also added the following include:  Then we have defined the following global variables:    Where string contains the symbols is ploting on the LCD, and stringUP and stringDOWN represent the symbols is ploting on the UP and DOWN half of the LCD respectively.  Then we created the function to plot the symbols received on the LCD:  In the main, we just initialize the LCD and make USART1 to receive data in interrupt mode:    Finally, we created the USART1 callback function:    We first make the USART1 to receive the next byte, as we used a non-blocking mode of the USART, we can continue processing the received byte. Our idea is impemented a small “text editer” which means we put the receive symbols in order and display them on the LCD. When LCD is full, we reset the LCD.  Finally, the code works as we expected. | | | |
| Professor comments: | | | |