Spring 2022 Embedded Systems Lab 5 – Asynchronous Serial Communication with EUSART Module and Interrupts
No lab report is due (only demo)

Objectives:
In this lab, the UART module available in the PIC16F886 will be used and configured for asynchronous communication with another PIC16F886. One development board will serve as a transmitter, the other will serve as receiver. Table 12-1 shows the registers associated with asynchronous transmission, and Table 12-2 shows the registers associated with asynchronous reception. In addition, the Interrupt feature of the PIC chip will be explored.

Instructions:

Asynchronous Transmission:

- The 8-bit mode for data transmission will be used.
- While reading section 12.1 EUSART Asynchronous mode, take note of the registers you need to set or clear in 12.1.1.1.
- UART is a character-by-character transmission, each character variable has 1byte or 8-bits of data.
- You will copy the character into the TXREG register and wait for the contents of the buffer to clear before sending the next byte (Read 12.1.1.3 to understand how to make use of the transmitter interrupt flag bit to know when the transmitter is ready).

Asynchronous reception with interrupt service routine

- You will use the 8-bit reception and write an interrupt-service routine in your program. This way the MCU can perform its tasks without having to spend time waiting on the other MCU to transmit data.
- We will first set up the receiver (read 12.1.2.1 to set and clear the associated registers for enabling the receiver).
- When using interrupts, please follow 12.1.2.3 Receive Interrupts section.
- The RCIF flag should be checked if the buffer is empty.
- Read 12.1.2.5 Receive Overrun Error as it is likely that the transmitter may send characters more than what the receiver can handle at that time. If the overrun flag bit is set, the receiver can ask the transmitter to send data starting from the recently received data. You do not need to implement this procedure in this assignment.
- There are two ways to clear this flag to continue reception. Make sure to introduce a short delay between the steps here to clear the overrun error bit. The module needs short time to be disabled and enabled.
- We get the received data by reading the RCREG register.

EUSART Baud rate generator

- The baud rate generator is a dedicated timer for the EUSART module.
- Example 12-1 shows you how to calculate the desired baud rate.
- Look at the tables in 12-5 for asynchronous modes, you should set the baud rate to 9600 at 8MHz and set the values for the registers accordingly from the tables.

Internal oscillator HFINTOSC at 8MHz

- The Oscillator Control Register will let you choose the frequency of the internal oscillator to 8 Mhz.
- Let both the HFINTOSC and LFINTOSC be stable.
- You can go to Production→ Set Configuration bits→ Choose the FOSC option Internal Oscillator→ Generate code to output or set the FOSC bits in CONFIG1 configuration word register 1 to do the same.

Interrupt service routine

- An interrupt service routine is like any other C function except that it accepts no arguments in its call and no return value.
- The function is declared as an ISR with the__interrupt () keyword beforethe function name (refer the interrupts section in XC8 user guide).
- The interrupt flag bits will be checked with the "if statements" and the appropriate instructions are executed in the "if statement" body to handle this interrupt flag.
- Interrupts are not exactly the same as the interrupt flag bits although they are closely related. A flag bit may be set without raising an interrupt. If interrupts are enabled, a module may raise an interrupt and set/clear the corresponding flag. When the device notices that an interrupt is raised regardless of the source of the interrupt, it will go through the body of thefunction defined by the**__interrupt** keyword and check the flag bits that we have in the "if statements."
- You can have any C construct in any part of the ISR as in any other function **(**loops/case statements/function calls, etc.**)**.
- __interrupt is supported by PIC16, the other kinds of constructs you see inthe XC8 manual are supported by more powerful PIC MCUs like PIC18, PIC24, PIC32, etc.

Interfacing

Locate the RX/TX pins on the device from the 28 pin LIN demo board manual and connect the TX to RX for two PIC16F886 devices using two separate 28 pin LIN demo boards. You will now write the same program on both devices. On the press of a push button connected to RE3, the device will transmit.

You can indicate

- Transmission by DS0 blinking
- Interrupt raised by DS1 blinking
- Receive buffer full by DS2 blinking
- Overrun Error by DS3 blinking
- (Optional) Blink all LEDs at the same time with the reception of character that matches with a character of your choice.

Supplying power

- Write your codes into both PIC16F866s (Tx/Rx) and disconnect the programmer.
- Do not connect external power supply to the devices/boards
- Verify the functionality of the system