EE4144 – Intro to Embedded Systems

Lab Exercise 3

Interrupt Subroutines and UART Serial Communications

Objective:

This lab exercise demonstrates how to implement interrupt subroutines and use the UART provided by the Atmel328P microcontroller. We begin by setting up the UART for serial communications between two Atmel328P controllers. Two Arduino Uno controllers are required for this lab so you should form teams of two. One Arduino (A) will set up a simple push-button sensor that will activate the onboard LED on the other Arduino (B) using interrupts and the UART serial interface. Arduino (B) will then respond to (A) confirming the status of its LED. Arduino (A) will then illuminate its own LED to match the status of (B)'s LED

Setup:

The following equipment will be required for this lab exercise:

1. 2 Arduino Uno R3 controllers with breadboard prototyping kits
2. Interconnect wires
3. 1 momentary push button included in your kit

Setting up the UART:

There is only one UART on the Atmel328P, which utilizes Arduino Pin 0 as Rx and Arduino Pin 1 as Tx. These pins are labeled on your Arduino. We wish to set up the following configuration for the UART serial bus on both Arduinos:

- Bit Rate: 57,600
- Data Bits: 8
- Parity : None
- Stop Bits: 1
- Asynchronous/Normal Speed
- Rx Interrupt Enabled

Using C, write the setup code for the following registers:

UCSRA, UCSRB, UCSRC and UBRR

→(0) Show the lab instructor your setup code

Be sure to set up both Arduino units to the same exact serial parameters.

Setting up the Hardware:

In this step we wire up the switch and the serial link between Arduinos.

1. Wire up the momentary push button switch such that one terminal is connected to Arduino (A) ground and the other terminal is connected to Arduino (A), Pin 2.
2. Wire Arduino (A) Tx to Arduino (B) Rx
3. Wire Arduino (A) Rx to Arduino (B) Tx
4. Wire Arduino (A) Ground to Arduino (B) Ground

→(1) Show your connection setup to your lab instructor.

Setting up the Pin Change Interrupt on Arduino A

We wish to implement a subroutine that handles a pin change interrupt for Arduino (A) Pin 2. So on Arduino A, set up the following interrupt registers using C:

DDRD, PCICR, PCMSK0

→(2) Show the code that sets up these registers and explain your bit selections.

Now, write the ISR that handles this interrupt, ISR(PCINT0_vect) on Arduino A.

This ISR will be called each time Pin 2 on Arduino (A) changes. In our ISR, we want to send a "0" to Arduino (B) over the UART serial bus if the button is released and a "1" if the button is depressed. In your routine, detect the state of Pin 2, and write the UDR register accordingly. Recall, the UDR register should contain "0" (0x30) if the button is released, "1"(0x31) if the button is depressed.

→(3) Show your lab instructor your ISR(PCINT0_vect) routine.

Arduino A is now set up to transmit one byte over the UART serial link to Arduino B each time Pin 2 changes states.

Setting up Arduino(B)

The serial parameters of Arduino B should already be setup in the previous steps. Be sure to double check that the Rx interrupt on the UART is enabled on BOTH Arduinos.

Setup the DDRB register such that Pin 13 is set as an output.

To handle the Rx interrupts on Arduino B, implement the ISR(USART_RX_vect) function. This function will run each time a character is received by the UART. In this function, read the UDR register and see if it contains 0x30 or 0x31. Based on the value, illuminate the onboard LED (Pin 13) using the PORTB register.

Immediately after assigning the PORTB register, send back to Arduino (A) 0x30(OFF) or 0x31(ON) depending on the state of the LED.

→(4)Show the lab instructor the ISR(USART_RX_vect) subroutine and the setup for Pin13.

Completing the setup for Arduino(A)

Arduino A is already setup to detect the push button state using a pin interrupt. In addition, Arduino(A) is programmed to notify (B) of the pin change by sending a byte over the serial bus. We now need Arduino(A) to implement the USART_RX_vect ISR so that when it receives a byte from Arduino(B) it can illuminate its LED accordingly.

Setup the DDRB register such that Pin 13 is set as an output.

To handle the Rx interrupts on Arduino A, implement the ISR(USART_RX_vect) function. This function will run each time a character is received by the UART. In this function, read the UDR register and see if it contains 0x30 or 0x31. Based on the value, illuminate the onboard LED (Pin 13) using the PORTB register.

→(4)Show the lab instructor the ISR(USART_RX_vect) subroutine and the setup for Pin13.

Testing the setup

Download your code for Arduino(A) and Arduino(B) and confirm the hardware connections made previously in the lab. Each time you press the push button, a message is sent to illuminate the LED on Arduino(B). A confirmation is sent

back to Arduino(A) illuminating the LED on Arduino(A). The net result: Each time the button is depressed, the LED on BOTH Arduinos should illuminate and when released, both LEDs should go dark.

→(5)Demonstrate the proper function to the lab instructor.

Troubleshooting Tips:
If the setup is not working….

1. Confirm that the serial setup on both Arduinos is identical
2. Confirm Tx-Rx and Rx-Tx
3. Don't forget to setup the LEDs as outputs and the button as an input
4. Check the spelling of your ISRs; they must be exact.