LABORATORY REPORT

To: Dr.Randy Hoover

From: Benjamin Lebrun, Benjamin Garcia

Subject: Lab Assignment 2: AVR Assembly and UART Communication

Date: February 24, 2019

Introduction

This lab required us to output the value in a counter to three LEDs as well as to the UART serial port. Additionally, we were required to accept input from the serial port and use this input to set the value of the counter. A component of the output requirement was that it print/flash the LEDs every two seconds using a wait function that we created.

Our implementation holds the counter in a register and converts to and from an ASCII representation for presenting information over the serial port. Pins 0, 1, and 2 on PORTB were used for lighting up the LEDs. To handle receiving information over the serial port, we enabled the 'UART RX Complete interrupt' so that whenever a key is pressed, it will set the value of the counter for the next time the LEDs are flashed. Our wait uses a triply nested loop to consume 16,000,000 cycles (1 second when running at 16MHz).

The finished implementation blinks on a two second cycle (two seconds on, two seconds off) and key-presses are registered for the next cycle. When first turned on, the counter is initially set to 0, and the LEDs are off.

Equipment

This lab required the following equipment:

- 3 resistors (220Ω)
- 3 LEDs
- 4 wires (male-male)
- 1 USB-A to USB-B cable
- 1 breadboard
- 1 Arduino UNO (ATMega328P)
- Atmel Studio 7
- Putty

configuration

The Arduino was connected to the computer with the USB-A to USB-B cable. On the breadboard, each LED was placed in a circuit with a single resistor connected to ground. One of the Arduino's ground pins was connected to the breadboard's ground, and the LEDs were connected to PORTB pins 0, 1, and 2 (pins numbered 8, 9, 10 on the Arduino).

Implementation

One Second Wait

The wait implementation uses three registers as loop variables to create a busy loop that occupies 15,999,984 cycles with variable pushing and popping and the function's return occupying a further 16 cycles to give 16,000,000 cycles.

ATOI

Handles converting numbers in ASCII representation to their depicted value (i.e. '7' is converted from 0x37 to 0x07).

It subtracts '0' (0x30) from the incoming ASCII character value to produce the numeric value. Once this is done, the value is stored in the counter for use on the next light cycle.

ITOA

Does the exact opposite of ATOI, it converts a numeric value to an ASCII character (i.e. 0x07 converts to 0x37 '7').

This is accomplished by subtracting the quantity (0 - '0') from the number (equivalent to adding 0x30). This value is sent over the serial line to the computer, where Putty can display it.

UART_INIT

UART_RX (Interrupt)

To handle the RX complete interrupt, we first altered the Init code provided to set UCSR0B to 0x98 to enable bits 7 (RX Complete Interrupt), 4 (RX Enable), and 3(TX Enable). Once the global interrupt bit was set, our function 'UART_RX' would be called whenever the UART buffer was filled. This function calls ATOI to set the counter and then returns control to the last instruction executed.

UART_TX

Unlike the RX function, UART_TX is not an interrupt, and is called every time the 'flash_leds' function is called

Main Loop and Startup

Discussion

Discuss challenges met in completion of the lab and how you solved them. Communicate what you learned from the lab, and comment on where this lab material would be applicable 'in the real world'.

Responses

Respond intelligently and at necessary length to any questions posed in the lab assignment.

1. Enumerations are a useful for addressing specific questions.

Appendices

You'll want to put your full code here, possibly datasheets. Each appendix should start on a fresh page. Check out the **lstlisting** package, it might make your life easier.

Appendix A: Nothing