Homework 2 Report

EEL 4742C

T TR 9:00-10:15

Group Members: Justin Bates, Kelsey Cameron, Christian Theriot

Objective: Implement a basic Morse Code Translator using the MSP430 Launchpad.

Procedure: Using the C programing language and various embedded systems concepts including timers, UART, and interrupts we were able to successfully output a Morse Code transmission with the MSP430 LED's. We calculated the value of TA0CCR0 in order to create a delay of 250ms with an equation found in the lab 6 manual. The frequency of our particular chip was 1048576hz giving us a hex value of 0x8000 for TA0CCR0. In our case, we used a count up mode (MC_1) through the timer A0 control line. The use of timers was important to allow the Morse Code transmission to be accurate and readable. In order to use these timers, interrupts had to be enabled, and trigger depending on which segment the Morse Code was outputting. For example, the character 'a' represented in binary by '10111', would trigger the 250ms timer 5 times, allowing other interrupts to be enabled simultaneously, as described below.

Setting up UART proved to be very difficult. To make UART compatible with our MSP430 we first set up the correct controls lines to use pins 4.4 and 4.5. Source Select 2, UCABR0 and UCABR1 allowed us to use the 1.04 mhz frequency compatible with UART.

The main Morse Code logic starts by allowing a user to input up to 63 characters that are saved into an array. If a carriage return or new line value is entered the loop will break and the code will continue. For convenience, another loop is added to print what the user is typing in the terminal to echo back each character. Two separate arrays of strings were used to produce patterns for either an alphabetical letter or a number 0 to 9. A series of if and else is statements are included to convert the asci representation to a decimal value. This decimal value will represent the position in the array of letters or numbers are. For example, position 0 represents both 'a' and '0' in their respective arrays. There is also an if statement to determine if the character entered was a space. This will return the proper number of zeros for a space character (7 zeros.) Once all characters in the array are converted, interrupts are enabled and another series of if and else if statements determines if the current position in a string is a zero or one (LED off or ON.) After this, the timer interrupt triggers and the Morse Code string position increments, which points to the next character. From there, a different pointer continues to increment every time the timer interrupt is triggered, giving the LED light either a 1(on) or 0(off.) This continues until the entire message is read and output through the MSP430.

For the extra credit, while the light flashes off each morse code pattern, if the interrupt for the button in port 2.1 is triggered, the message is printed on screen via serial communication in the 250 milliseconds between each light signal. By using a pointer to point to the first character of the string, and printing the value of the pointer character by character, we were able to successfully print the entire string without affecting the value of the LED lights.

Conclusion:

In conclusion, this assignment was an excellent opportunity for us to learn how to handle multiple types of interrupts at the same time, while reviewing string manipulation in C. This assignment allowed us to practice in real time every concept that we've reviewed in class, and enlighten our understanding of the material. Despite barriers of configuring serial communication, or other issues with getting the interrupts to trigger, we managed to complete this assignment correctly, and even complete the extra credit.