CS 200: Computer Organization

Project 11: Direct I/O

Shariq M. Jamil

Due: Friday, May 2, 2014

Overview

Purpose

This project required us to write a procedure in Assembly that would blink names using the LED on the EZ-430.

Approach

I started by downloading Code Composer Studio 6 and IAR Embedded Workbench and used them to see which one would serve my needs better. I looked around online for Morse code implementations in Assembly before I began. I found similar code online for blinking LEDs and used it as a guide. I started out implementing in C and then planned on converting the function to Assembly. I created a character array which contained Morse translations of the letters of the alphabet. The program accepts user input and iterates through the array. Upon finding the character the program performs operations depending on the information stored in for the letter. The program “sleeps” for a moment between letters. I got the program working fine using C and the LED blinked the way I wanted it to. An issue arose when I found out that CCS cannot be used for programming in Assembly natively. I had to switch over to IAR Workbench. When I tried loading sample “Flash the LED” codes onto my project board, the board refused to load the code. I looked up the error everywhere but could not find any solutions to this. I went back to CCS and the board loaded C code fine. At this point, I decided to go ahead and polish my C implementation.

Solution

Source Code

```
//*******************************************************************************
// CS 200: Project 11, Direct I/O
//*******************************************************************************

#include "msp430.h"

const char* code_chart[] = {
                        ".-" , // A
                        "-...", // B
                        "-.-.", // C
                        "-..", // D
                        ".", // E
                        "..-.", // F
                        "--." , // G
                        "....", // H
                        ".."  , // I
                        ".---", // J
                        ".-.-", // K
                        ".-..", // L
                        "--", // M
                        "-.", // N
                        "---", // O
                        ".--.", // P
                        "--.-", // Q
                        ".-.", // R
                        "...", // S
                        "-", // T
                        "..-", // U
                        "...-", // V
                        ".--", // W
                        "-..-", // X
                        "-.--", // Y
                        "--.." // Z

};

/**
 * sleep
 *
 */

void sleep(int duration)
{
        int i;
        for(i=2*duration;i>0;i--) { LPM0; }
}
```

```
/**
 * LED ON
 *
 */

inline void led_on()
{
	P1OUT |= 0x01;
}

/**
 * LED OFF
 *
 */

inline void led_off()
{
	P1OUT &= ~0x01;
}

/**
 * Accept string and work LED
 */

void work_LED(const char *s)
{
	int i,j, index;
	const char *dots;
	for(i=0;s[i] != (char) 0; i++){
		index = -1;
		if (s[i] >= 'A' && s[i] <= 'Z')
			index = (int) s[i] - (int) 'A';

		if (s[i] >= 'a' && s[i] <= 'z')
			index = (int) s[i] - (int) 'a';

		if (index != -1) {
			dots = code_chart [index];
			for(j=0;dots[j] != (char) 0; j++) {
				switch(dots[j]) {
				case '.':
					led_on();
					sleep(1);
					led_off();
					break;

				case '-':
					led_on();
					sleep(3);
					led_off();
```

```
                                        break;
                                }
                                sleep(1);
                        }
                } else {
                        // alien input
                        sleep(4);
                }
                led_off();
                sleep(3); // blink
        }
}

int main(void)
{

        WDTCTL = WDTPW + WDTHOLD;

        // LED output
        P1DIR |= 0x01;


        CCTL0 = CCIE;
        // set CCR0 delay to 60,000
        CCR0 = 60000;
        TACTL = TASSEL_2 + MC_1;

        _BIS_SR(LPM0_bits + GIE);
        // begin
        led_off();
        //work it
        for(;;)
        {

                work_LED("Rick Jamil");
                //pause
                sleep(10);
        }
}

#pragma vector=TIMERA0_VECTOR
__interrupt void Timer (void)
{
        LPM0_EXIT;

}
```

Sample Output

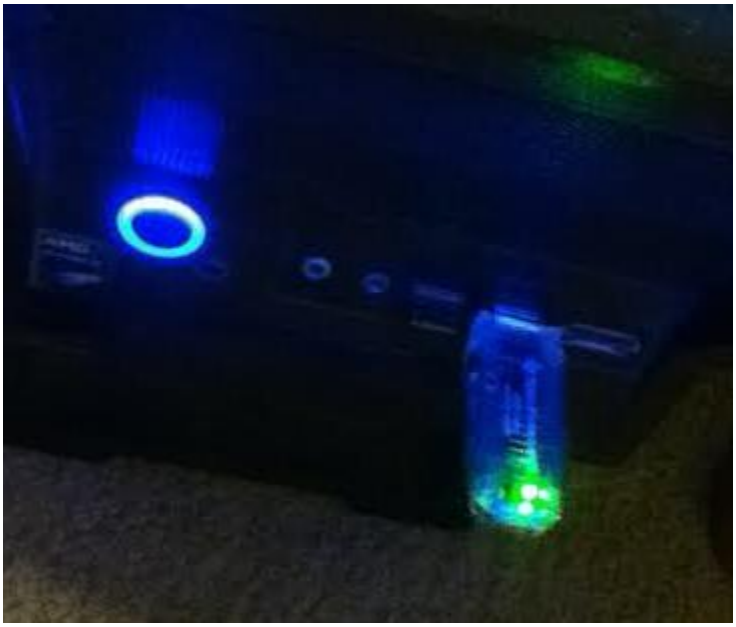

Figure 1: The EZ-430 USB Tool blinking my name in Morse code.

I posted video of the LED blinking my name in Morse code on YouTube. The link is:

https://www.youtube.com/watch?v=t4Di3ERCPCE

Conclusion

The project turned out to be a lot more difficult than it seemed. The procedure to download the evaluation versions of the IDE's was long and meticulous. However, I understand the need for these security measures. I thought I would easily be able to load Assembly onto the board but I could not get it on there using either IDE. It was highly rewarding to see my code come to live through the USB. I am more motivated to purchase boards like the Raspberry PI or the MSP-430 and start developing. This could lead to a fun (and frustrating) summer.

References

eZ430-F2013 Development Tool User's Guide

http://www.ti.com/lit/ug/slau176d/slau176d.pdf

Code Composer Studio™ v6.0 for MSP430™ User's Guide

http://www.ti.com/general/docs/lit/getliterature.tsp?baseLiteratureNumber=slau157&fileType=pdf

DreaminCode – A program that converts strings to Morse code

http://www.dreamincode.net/forums/topic/155153-morse-code-translator/