CPE 325: Intro to Embedded Computer System

Lab09

SPI COMMUNICATION: MASTER/SLAVE

Submitted by: Dan Otieno

Date of Experiment: 03/20/22

Report Deadline:_03/23/22

Demonstration Deadline: 03/30/22

Introduction

Write a brief discussion on what the lab is about. (Use the tutorial and write in your own words. DO NOT copy text).

This lab introduces Serial Peripheral Interface (SPI). We use a master/slave setup to enable serial communication between the MSP430 2013 and MSP430 4618. The 4618 chip serves as the master and we pass the inputs through it, and get the outputs through the 2013, this method is referred to as a MISO setup (Master In, Slave Out).

Theory

Write short notes on each topic discussed in lab.

SPI vs. UART: When we have two peripherals communicating via SPI, the master provides a shared clock, in a synchronous protocol. In the 4618 chip, SPI is supported by the Universal Serial Communication Interface (USCI) while in the 2013 it is supported by the Universal Serial Interface (USI). The master (4618) provides the clock that's shared by both devices and an enable signal for the slave. One of the differences between SPI and UART, is we use different timers, UART uses Timer A while SPI uses Timer B.

DMA Controller: This is a module that transfers data between peripherals and memory. DMA stands for Direct Memory Access and a DMA controller can be an input/output peripheral when configured in the master chip or by the CPU. In the MSP430, the DMA controller uses the MCLK clock cycles to transfer bytes, words, or both.

Results & Observation

Program 1:

Program Description:

The program assignment for this lab required interfacing the 4618 and 2013 via SPI. We write the program in the master device to send a welcome message to the terminal and prompt user input. The user then must input a number between 1 and 20, which serves as a multiplier, defined in the slave program to adjust the frequency of the LED in the MSP430 board. If the user enters a 0, then that value is passed onto the slave, and if the user enters a question mark, then the output is the current multiplier value. This is all displayed in a HyperTerminal, and we use both UART and SPI concurrently to pass/transmit characters through Timer A and Timer B buffers. The baud rate is 57600 and for this lab, I used MobaXterm for terminal outputs.

Program Output:

Terminal output will be demonstrated in class.

Conclusion

Unfortunately, as of the time of this report, I have not yet been able to get the desired result during coding and testing, but I have gained some understanding of SPI and the Master/Slave setup of interfacing the chips in the MSP430 mcu.

Appendix

Code attached is not yet generating the results, but attached as per the stage I had reached during the assignment.

Table 01: MSP430 4618 Master source code

```
#include <msp430.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define stateQmark  0x3F      // Character '?'
#define stateZero   0x30      // Character '0'
#define Dummy       0x00      // Character NULL - used for dummy write operation

#define Zero    0x00
#define Qmark   0xFF

void SPI_setup();
void SPI_Send_Character(char);
void UART_Send_Character(char);
void UART_send_String(char*);
void UART_get_word(char*, int);

char prompt[] = "\r\nHello, please enter a number: \r\n";
char invalid[] = "\r\nInvalid multiplier entered";
char num[5];
unsigned char my_char;

int main(void)
{
      WDTCTL = WDTPW + WDTHOLD;       // Stop watchdog timer
      UART_setup();                   // Setup USCI_A0 module in UART mode
      SPI_setup();                    // Setup USCI_B0 module in SPI mode
      int z, i, numit;

      for(z = 100; z>0; z--);
      UART_send_String(prompt);

      while(1)
      {
          UART_get_word(num, 5);

          switch(my_char)
          {
          case    '0':
```

```
            SPI_setState(stateZero);
            for(i = 1000; i>0; i--);
            SPI_Send_Character(Zero);
            UART_Send_Character(SPI_getState());
            break;

        case    '?':
            SPI_setState(stateQmark);
            for(i = 1000; i>0; i--);
            UART_send_String("\r\nThe stored multiplier value is: ");
            SPI_Send_Character(SPI_getState());
            //SPI_Send_Character(num);
            break;

        default :
            UART_send_String(prompt);
            break;
        }

        if(num > 0 && num <= 20)
        {
            numit = atoi(num);
            SPI_setState(numit);
            for(i = 1000; i>0; i--);
            UART_Send_Character(SPI_getState());
        }

        else
        {
            UART_send_String(invalid);
        }

        UART_send_String("\r\n");
    }

}

void SPI_setup(void)
{
    UCB0CTL0 = UCMSB + UCMST + UCSYNC;// Sync. mode, 3-pin SPI, Master mode, 8-bit data
    UCB0CTL1 = UCSSEL_2 + UCSWRST; // SMCLK and Software reset
    UCB0BR0 = 0x02;                // Data rate = SMCLK/2 ~= 500kHz
    UCB0BR1 = 0x00;
    P3SEL |= BIT1 + BIT2 + BIT3;   // P3.1,P3.2,P3.3 option select
    UCB0CTL1 &= ~UCSWRST;          // **Initialize USCI state machine**
}
```

```
unsigned char SPI_getState(void)
{
    while((P3IN & 0x01));           // Verifies busy flag
    IFG2 &= ~UCB0RXIFG;
    UCB0TXBUF = Dummy;      // Dummy write to start SPI
    while (!(IFG2 & UCB0RXIFG));    // USCI_B0 TX buffer ready?
    return UCB0RXBUF;
}
void SPI_setState(unsigned char State)
{
    while(P3IN & 0x01);             // Verifies busy flag
    IFG2 &= ~UCB0RXIFG;
    UCB0TXBUF = State;              // Write new state
    while (!(IFG2 & UCB0RXIFG));  // USCI_B0 TX buffer ready?
}

void SPI_Send_Character(char my_char)
{
    while(!(IFG2&UCB0TXIFG));    // Wait until can transmit.
    UCA0TXBUF = my_char;         // Tx Buffer gets my_char variable.
}

void UART_setup(void)
{
    P2SEL |= BIT4 + BIT5;           // Set UC0TXD and UC0RXD to transmit and receive data
    UCA0CTL1 |= UCSWRST;            // Software reset
    UCA0CTL0 = 0;                   // USCI_A0 control register
    UCA0CTL1 |= UCSSEL_2;           // Clock source SMCLK - 1048576 Hz
    UCA0BR0 = 18;                   // Baud rate - 1048576 Hz / 57600
    UCA0BR1 = 0;
    UCA0MCTL = 0x02;                // Modulation
    UCA0CTL1 &= ~UCSWRST;           // Software reset
    IE2 |= UCA0RXIE;                // Enable USCI_A0 RX interrupt
}

void UART_Send_Character(char my_char)
{
    while(!(IFG2&UCA0TXIFG));    // Wait until can transmit.
    UCA0TXBUF = my_char;         // Tx Buffer gets my_char variable.
}

void UART_send_String(char* string)
{
    //int i = 0;
```

```
    while(*string != '\0')  // Send character by character
    {
        UART_Send_Character(*string++);
        //i++;
    }
    UART_Send_Character('\r');
    UART_Send_Character('\n');
}

void UART_get_word(char* buffer, int limit)
{
    int counter = 0;
    my_char = 0;
    while(my_char != '\r' && counter < limit)
    {
        my_char = UCA0RXBUF;
        while(!(IFG2&UCA0RXIFG));         // Wait until character can be read from RXBUF.
        UART_Send_Character(my_char);
        *buffer = my_char;
        buffer++;
        counter++;
    }
    *buffer = '\0';
}
```