MECH 423 – Mechatronic Product Design – 2020W
Lab #2: MSP430 Microprocessor Programming

Objectives

This lab consists in a set of exercises designed to teach you the basics of microprocessor programming using MSP430 processor family from Texas Instruments. Specific learning objectives include the following:

- Understand the capabilities and operation of microprocessors
- Read diagrams, memory maps, and instructions from the datasheet and the user's guide
- Use the integrated development environment (IDE) for programming and debugging
- Configure internal clocks and their connection with peripheral units
- Configure and use digital I/O ports
- Configure and use interrupts
- Serial communications using the UART
- Analog-to-digital conversion
- Use the timer to generate and to measure precise timing
- Create a circular buffer to accept message packets from a PC
- Understand the difference between polling, interrupt-driven, and event-driven programs

Logistics and Evaluation

This lab is done individually. Students are expected to learn to program the MSP430 microprocessor on their own using the lecture material and available literature on the web. Students are expected to write code to complete lab exercises provided in this document. The lab exercises are graded for completion. Exercises 1-3 are due on 2020/09/30. Exercises 4-7 are due on 2020/10/05. Exercises 8-10 are due on 2020/10/14. **A lab exam will take place at the normal lab session on 2019/10/19.** The lab exam consists of exercises similar to the lab exercises. Students will have up to 120 minutes to complete the exam and demonstrate the functionality of their code to a TA. This lab is worth **25%** of your final grade. Completion of the in-lab exercises is worth **5/25**. The lab exam is worth **20/25**.

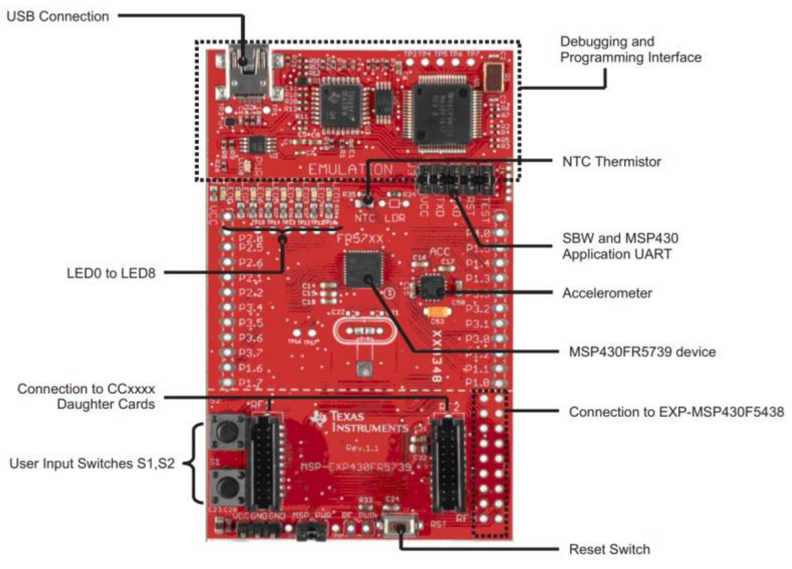
USB Connection
Debugging and Programming Interface
NTC Thermistor
SBW and MSP430 Application UART
LED0 to LED8
Accelerometer
MSP430FR5739 device
Connection to CCxxxx Daughter Cards
Connection to EXP-MSP430F5438
User Input Switches S1,S2
Reset Switch

Figure 1: EXP Board layout

Hardware and Interface

The EXP Board (Figure 1) is designed to be a learning and development tool for MSP430F57xx microprocessors. It is designed to provide maximum versatility and reconfigurability while avoiding the need for soldering. The complete schematic is available in a separate file. Most of the connector pins are mapped to pins on the MSP430F5739, while a few complementary pins provide access to other components on the PCB. The EXP Board can be powered from a standard USB port.

Programming and Debugging

The hardware interface used to load programs onto the MSP430 microprocessor is called Spy-Bi-Wire JTAG. This interface also provides debugging capabilities including control of program execution, single step through instructions, and the ability to read from any location in memory. The Spy-Bi-Wire interface is administered through the debugging and programming interface shown in figure 1. There are three lines required for programming and debugging: Test, Rst, and Gnd. Luckily the EXP board handles the connections to the MSP430 microprocessor, leaving only the USB cable to be plugged in by the student. The USB cable should be left plugged in during debugging.

The software interface for programming and debugging microprocessors is known as the integrated development environment (IDE). We will use Code Composer Studio (CCS), which is available as a free download from TI. Specifically, go to this webpage: http://processors.wiki.ti.com/index.php/Download_CCS and download the latest version. Any imposed code size restrictions in CCS is irrelevant for this lab since we generally will not exceed the stated limit. You will have to register with TI.com and answer questions about how you will not use this product to help terrorist organizations. Next, install CCS and install only materials relevant for the MSP430 microprocessors. When you first plug the EXP Board into your PC's USB port, drivers will automatically be installed over the Internet. Therefore, make sure you have an active internet connection and administrative privileges for your PC. Finally, a serial communications program will be useful in later exercises. A good one is open PuTTY, available from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html. Download "putty.exe" for Intel x86 machines.

Lab Exercises

Exercise 1: Clock Configuration

1. Set up SMCLK to run on the DCO.
2. Configure the DCO to run at 8 MHz.
3. Set up the SMCLK with a divider of 32.
4. Configure P3.4 as an output and set it to output the SMCLK. The SMCLK frequency should be ~250 kHz. Check it using an oscilloscope.

Exercise 2: Digital I/O

1. Configure PJ.0, PJ.1, PJ.2, PJ.3, P3.4, P3.5, P3.6, and P3.7 as digital outputs.
2. Set the LEDs 1 to 8 to output 10010011 .
3. Write a program to blink the LEDs that are 0 in step 2. Use a delay loop in the main infinite loop to visibly blink the LED.

Exercise 3: Interrupts

1. Configure P4.0 as a digital input.
2. The switch S1 is connected to P4.0 on the EXP Board. Enable the internal pull-up resistors for the switch.
3. Set P4.0 to get interrupted from a rising edge (i.e. an interrupt occurs when the user lets go of the button). Enable local and global interrupts.
4. Configure P3.7 as an output (this is connected to LED8).
5. Write an interrupt service routine to toggle LED8 when S1 provides a rising edge.

Exercise 4: UART

1. Configure the UART to operate at 9600, 8, N, 1.
2. Set up P2.0 and P2.1 for UART communications.
3. Write a program to periodically transmit the letter ‘a’ to the serial port.
4. Check the transmission using the CCS terminal in debug (or PuTTY).
5. Enable UART receive interrupt. Enable global interrupt.
6. Set up an interrupt service routine so that when a single byte is received, the same byte is transmitted by back (or echoed) to the serial port. Check using CCS terminal/PuTTY.
7. In addition to echoing, also transmit the next byte in the ASCII table. Check again using PuTTY.
8. Add code to turn on LED1 when a ‘j’ is received and turn off LED1 when ‘k’ is received.
9. Change the UART speed to 57.6 kbaud and make sure the code in exercise 4.7 and 4.8 still works.

Exercise 5: Timer I

1. Set up Timer B in the “up count” mode.
2. Configure TB1.1 to produce a 500 Hz square wave. You may need to use frequency dividers when setting up the clock and the timer. Output on P1.0. Verify using an oscilloscope. LED5 should also be lit.
3. Configure TB1.2 to produce a 500 Hz square wave at 25% duty cycle. Output on P1.1. Verify using an oscilloscope. LED6 should be lit. Verify that LED6 is dimmer than LED5.

Exercise 6: Timer II

1. Set up timer A to measure the length of time of a pulse from a rising edge to a falling edge.
2. Connect the timer output from the previous exercise to the input of this timer.
3. Using the debugger to check the measured 16-bit value.

Exercise 7: ADC I

1. Configure P2.7 to output high to power the accelerometer.
2. Set up the ADC to sample from ports A12, A13, and A14.
3. Write code to sample data from all three accelerometer axes.
4. Bit shift the 10-bit result to an 8-bit value.
5. Set up a timer interrupt to trigger an interrupt every 40 ms (25 Hz).
6. Using the timer interrupt, transmit the result using the UART with 255 as the start byte. The data packet should look like 255, X-axis, Y-axis, Z-axis. Check that the transmission is active using CCS Terminal/PuTTY/ MECH 423 Serial Communicator.
7. Sample the ADC inside the timer interrupt service routine and transmit the result to the PC. Check the transmission rate using an oscilloscope by probing the UART Tx port, P3.4.
8. Test your MSP430 code using your C# program from Lab 1.

Exercise 8: ADC II

1. Set up the ADC to sample from the temperature sensor (NTC) port (ensure power is provided to the NTC using P2.7 pin).
2. Write code to sample data from the NTC. Bit shift the 10-bit result to an 8-bit value.
3. Transmit the result using the UART. Place your finger on the NTC and observe the range of values that can be obtained using the CCS Terminal/PuTTY/MECH 423 Serial Communicator.
4. Write code to turn the EXP board into a digital thermometer, using LED1-LED8 as readouts. At room temperature, only LED1 should be lit (having both LED1 & LED2 lit is also acceptable). The higher the temperature, the more LEDs will be lit. Resting your finger on the NTC should light all the LEDs.
5. As a test, place your finger in the NTC for 15 seconds, then release. The LEDs should all light up. After removing your finger, the LEDs should turn off 1 by 1. Demonstrate this to a TA.

Exercise 9: Circular Buffer

1. Write code to set up a circular buffer with 50 elements
2. Set up UART interrupts such that each new byte received by the UART is added to the end of the buffer.
3. Connect to the EXP Board using CCS Terminal/PuTTY. Type single characters into the terminal and then use the debugger to confirm that they are being correctly stored in the buffer.
4. Add code such that whenever a carriage return (ASCII code 13) is received, a single character is removed from the top of the buffer. Transmit the single character back to the PC when it is removed from the buffer. Check using PuTTY.
5. Add error-checking code to the circular buffer to prevent buffer overrun (more elements than 50) and buffer underrun (fewer elements than zero). Send error messages to the serial port.
6. Use CCS Terminal/PuTTY to check the correct operation of the circular buffer.

Exercise 10: Processing Message Packets

1. We will provide the MECH 423 Serial Communicator, which is a simple C# program to transmit and receive messages via serialport. Transmitted messages are to be organized according to the table below. Data byte #1 and Data byte #2 are the upper and lower bytes of a single 16-bit number respectively. For any numbers entered in the two Data byte boxes, and value >255 is changed to 255, and a value <0 is changed to 0. Since 255 is used as the Start byte, a value of 255 should not be used in the two Data byte boxes. Single bits can be used in the Escape byte box to identify any Data bytes that are 255, so that they can be changed back by the MSP430 program.

Start byte	Command byte	Data byte #1	Data byte #2	Escape byte
0xFF	0x01			

2. Modify the circular buffer code from Exercise 9 to handing the incoming message. Use the debugger to check that the message is being correctly generated.
3. When a packet is available in the buffer, extract the data bytes and combine together into a single 16-bit number.
4. Write code to change back any modified data bytes identified by the escape byte.
5. Remove the processed bytes from the buffer.
6. Use the received 16-bit number to set the frequency and/or the duty cycle of the square wave produced by Timer B in Exercise 5.
7. Verify the output frequency is correct using an oscilloscope or using the code in Exercise 8.
8. Add new commands to turn on and off LED1 from the C# program. For example, 0x02 = turn on LED1, 0x03 = turn off LED1, data bytes are irrelevant in this case.