Lab Task #8

1 Objective

Design a “low-power” embedded system that will sense its surroundings and provide feedback to a PC. Design with components using different supply voltages.

2 Pre-lab Tasks

1. Determine the “best” method of using the NTC thermister in your MSP430-based system. Justify your decision.
2. Review the I²C bus specification (available on the lab webpage)
3. Review the MSP430 USI peripheral
4. Identify several possible approaches to dealing with the fact that the PIC18Fxx2 signals are 0V/5V and the MSP430 signals are 0V/3.3V. Determine the “best” approach. Justify your decision.

3 Specification

This task will create a system where your MCU will measure the ambient air temperature with a thermistor. The temperature measurements will be forwarded to another system via I²C. The temperatures are recorded at the request of your PC which makes the request via a RS-232 serial stream.

Figure 1: System block diagram

3.1 Hardware design

Using your design from Lab #5 design a MSP430-based I²C slave device that measures the ambient air temperature using a thermistor. Your I²C slave device will implement a subset of commands used by the Maxim IC part DS1631. Your devices will respond to I²C addresses 0b100A₀000x where A₀ is an user-selectable address line and x is the I²C R/W# bit. Your device must implement the functionality of the following DS1631 commands:

1. Feel free to meet as a team to address this pre-lab task. Each member should come to the pre-lab team meeting with their methods identified with “pros” and “cons”. Discuss each unique method as a team, and determine collectively the best alternative.
2. The I²C library/object class code, the PIC18 hardware, and the PIC18 firmware from Lab #5 should not be altered for this lab. Of course, you will need to write a new PC side application to test this system described here.
• Access configuration register command (0xAC) reads of the 1SHOT and DONE bits. All other bits are read as ZEROs. Configuration writes are ignored.

• Software POR command (0x54) to stop current conversion and resets all registers to their POR state.

• Start convert T command (0x51) to start a temperature conversion

• Stop convert T command (0x22) to stop a temperature conversion in progress

• Read T command (0xAA) to read the last converted temperature from the 2-byte temperature register

Your I^2C device should return the temperature in the same 16-bit format as the Maxim DS1631, except you may use all eight fractional bits in your temperature representation, if you can compute them meaningfully.

Just like the DS1631, your “one-shot” temperature sensing device should go into a low-power state when idle. However, your device needs to be poised and ready to respond whenever temperature request is transmitted to it over the I^2C bus. Your device should be able to respond to I^2C communications at the I^2C standard of 400 kbps.

3.2 Test bench

Create software on the PC to communicate with your newly designed I^2C temperature device. Write a test bench to test your device’s operating modes and features. Be sure that your test bench fully exercises all aspects of your hardware and software. It is your responsibility to prove that your test bench is exhaustive.

4 Demonstrations

You must demonstrate the following to the lab TA (not necessarily before you leave lab):

1. Operation of the your temperature sensing device by running your test bench

2. Display of serial and I^2C transfers on the logic analyzer with bitrates measured for each

3. Explanation of how your test bench is complete and exhaustive

4. Preliminary “datasheet” describing your device, its limitations, and its pin-out

5 Deliverables

1. Document showing your design analysis justifying component values and circuit topology. (“I put it together and it just worked” is not acceptable. You must justify your design decisions.) If you are unable to meet any of the design specifications, you must provide justification as to why it is not possible.

2. Schematic capture of your circuit showing all components and their values (include everything except the breadboard). Submit BOTH the “raw” electronic schematic file AND a PDF document suitable for printing. You may use any schematic capture tool you desire, but you are encouraged to use a tool which will allow ultimate import into PCB layout. Your are encouraged to use Cadence/OrCAD or EagleCAD.

3. Screenshot or other documentation as supporting evidence of proper transfers at the proper bitrates

4. Code tree of your developed firmware (with all code adhering to the ECE 4723/6723 coding conventions) for both MCUs

5. Code tree of your developed PC library/objects (with all code adhering to the appropriate coding conventions from the community supporting your chosen language)

6. Code tree of your developed test benches

3Submit electronically to lab TA.
6 Bonus #1

Implement the full set of DS1631 features, including continuous conversion, thermostat comparison operation, configuration register read and writes. You can still implement a single user-selectable I\textsubscript{2}C address line. This bonus will provide 10 extra points if you successfully demonstrate your complete DS1631 emulation according to the methods used for the required lab features.

7 Bonus #2\textsuperscript{4}

Encapsulate the features of your temperature measuring device. Your new object must present a complete interface to all functions capable of being performed by your device. You must provide complete testbenches for your object that demonstrates that your new object functions properly and is complete.

This bonus will provide 5 extra points if you encapsulate the DS1631 subset device required by the lab or 10 points if you encapsulate the full DS1631 command set device described in the Bonus #1 above.

8 History

v1.0 6 SEPT 2006 – Initial version

\textsuperscript{4}This bonus only applies to implementations in Java or Python

\textsuperscript{5}This is made much easier if you have already encapsulated a Maxim IC DS1631 in the earlier lab.