CE2810: Embedded Systems Software 2

Math Blaster 2012

""Life is about making choices: you can either spend three quid on a glossy magazine or you can spend it clearing three square metres of minefield and help give people their lives back. As simple as that." -- Twiggy Lawson

Due: Friday, May 18, 2010 14:00

1 Objectives

- Construct an entertainment based embedded system (aka game)
- Employ serial communications to connect two embedded systems together
- Design a simple embedded system

2 Introduction

This is the final lab of the quarter (and there was great disappointment…) And it will be the most fun (we hope.)

For this lab, you will work directly with a partner (or two if we have an odd number in class) to develop a two player game, namely a math challenge game. The game will require two boards and a serial cable to allow communication between boards.

Playing the game will follow the following scenario. When a board is rebooted, it will display splash screen for 1 second with the author of the program (each person's board shall display their name) on the top line.

When the one second display is completed, the program will send a message out on the serial port asking "Will you play with me."[1] Any other embedded system on the port will respond with a message indicating "Yes, I will play with you".

At this point, the two boards will exchange player names. Player names may be up to 16 characters in length and are to be null terminated strings.

The system will then determine which player is to go first. The player must enter a mathematical equation on the screen and correctly answer it. For example, player one might enter "6 + 6 = 12" on the screen. If the answer is correct, then a tone designating a correct answer will be emanated by a piezobuzzer connected to PORTD OC1. If the answer is incorrect, a lower pitched "wrong" tone will be emanated and the correct answer will be displayed. Additionally, one LED on the board will light up indicating that one wrong answer has been provided.

If the answer entered by the first player was correct, the problem will then be sent to the second player to answer. The second player will enter only the result, not the actual equation. If the second player gets the answer correct, his / her speaker will generate a correct tone and the player will then be prompted to enter

[1] Note: This may not be the exact message, but this is the intent of the message.

a mathematical equation and answer. The behavior will then match that described previous for the first player. If incorrect, an incorrect tone will be generated, and the first player will be prompted to enter a mathematical expression and answer.

Play will continue until one of the players has answered a total of 5 expressions incorrectly. At this point, the displays of both player will show "GAME OVER". The board of the winner will have the LEDS flashing on and off in a 0xA5 – 0x5A pattern, display the winner's name on the lower line, and make a happy sound. The loser's board will have the LEDs turned off and display the winner's name on the lower line, and make a sad sound.

After doing this for approximately 10 seconds, the display will show a message "Press any key to play again" and will restart at the beginning when the player wants to play again.

3 Prelab

Before coming to lab on the 4th of May, watch the video tutorial online on using libraries with the GCC compiler.

4 Design

Guess what. The design is up to you. Aside from a common message structure, the design is entirely up to you!

You may find it wise to keep a sketchbook of the sequences between the two embedded systems, as this would allow you to test and develop independent of the second board using Realterm or another equivalent.

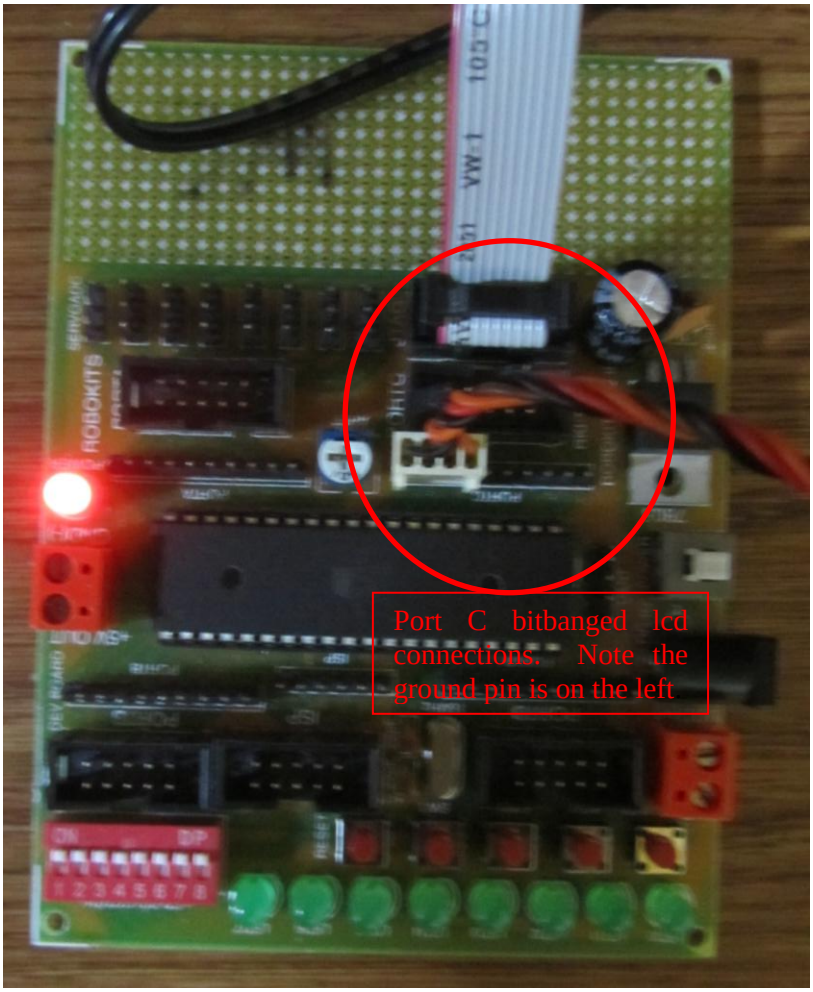
It may also be wise to sketch out a few UML diagrams showing the classes you will build and the attributes of each class within the system.

5 Wiring the boards together

To wire the boards together, you will need to connect the two serial ports. There are many ways to do this. However, the easiest way is to use the screw terminal connector and connect to port D. On both boards, connect the ground wires together, ground to ground. Also connect the VCC lines together. (Note: This will also allow you to power the board with only a single power supply.) When you have done this, connect the transmit pin on one board to the receive pin of the other board. Then connect a second wire from the receive pin on one board to the transmit pin of the other board. With this being completed, the boards can now talk with each other.

6 Connecting the Display

The figure below shows how one connects the RS232 monitor to the bit banged serial port. Connecting the display occurs in the same fashion. Note that for PORTC, the pin in the center of the board is ground, followed by VCC, transmit, and receive.


Port C bitbanged lcd connections. Note the ground pin is on the left

7 Data Structure

Communication between boards will use the structure given below. Communications shall use 8 bit, no parity, and one stop bit.

```
struct serialMessage
{
  uint8_t messageType;
  uint8_t dataLength;
  uint8_t messageData[16];
}
```

The message type will define the type of message being sent. The values that this byte may have include:

Message Type	**Value**	**Data Length**	**Data contents**
"Will you Play with me"	0x01	0	Empty
"Yes I will play with you"	0x02	0	Empty
"My name is"	0x03	The string length of the player's name, between 1 and 16 characters.	The player's name
"Solve this equation"	0x04	The length of the string that needs to be solved.	A sample problem to solve (i.e.) "16 * 3 ="
"Give me a problem to solve"[2]	0x05	0	Empty
"Game over. I lost"	0x09	0	Empty

8 Lab process

In addition to implementing the code, you are to follow the following, simple lab process.

1. Estimate how many lines of code you expect this project to be from start to finish.
2. Estimate how long you expect it to take you to complete this project.
3. Implement the solution, keeping track of how long it took to complete the project. (While it is not required that you use project dashboard, it is certainly permitted.)
4. At project completion, compare your line of code count and your time estimates with the actual values.

Before going gung ho with implementation, it might be very wise to sketch a design on paper and have the professor review it. That might save some time.

9 Driving the display

Since you will be communicating over a serial port, the display will need to be handled using a bit-banged display driver. You professor has graciously provided such a device driver which will allow you to send

[2] This will indicate that the player who went first was unable to solve their own equation properly. Thus, it is up to the other person to come up with a problem to solve.

RS232 communications to the display. Your board must be operating at 9600 baud for this to work properly.

10 Lab Assignment

10.1 Implementation Constraints

- Constraint C1: Somewhere within your code you must use a switch statement (hint: maybe handing incoming messages?)
- Constraint C2: The only allowable types within this program shall be the C99 types.
- Constraint C3: The source code shall compile cleanly without any warnings or error messages.
- Constraint C4: The program developed for this lab shall communicate at 2400 Baud.
- Constraint C5: The serial communications driver must be non-blocking. That is, it must not have an infinite loop that waits to see if the device can receive data.

11 Demonstrations

By May 9, 2011, you should have a hand drawn UML diagram showing the various modules that you will have within your program. Additionally, you should have implemented at least proof of concept code that shows that you can communicate with the LCD display using the bit-banged device driver available on the course website. For the bit banged display driver to work, your board must operate at 16 MHz crystal frequency.

By the end of lab on May 9, you should have the two boards talking to each other in some fashion. While it may not be complete, you should be able to send a message from one board to the other and verify reception of the message.

12 Deliverables

12.1 Lab Report

Now that you have completed your lab assignment, submit the following brief lab write-up detailing your experiences. The lab report should be submitted electronically through the online submission form on the course website. Only one submission is necessary per set of lab partners.

1. Introduction -> What did you accomplish with this lab?
2. Time Analysis and code analysis
    a. How long did you plan on spending on this project?
    b. How many lines of code did you estimate for this project?
    c. How long did this project actually take?
    d. How much code space does your program use? How many bytes are used per line of C code? How does this compare with the last lab? What has the impact of adding the string library to your code had on this metric?
3. Design Overview. How did you design this project. How did you partition the problem into pieces?
4. Things Gone Right
    a. What things went right in performing this lab?

5. Things Gone Wrong
   a. What problems did you have?
   b. What mistakes did you make?
6. Conclusions
   a. This section shall discuss what has been learned from this laboratory experience.
7. Source Code. Submit your well commented source code.

12.2 Project Demo

In addition to the lab report, you are required to demonstrate correct functionality. This must be accomplished prior to the end of class on Friday, May 18, 2012.

If you have any questions, consult your instructor.