CE2810: Embedded Systems Software 3

"When I was having that alphabet soup, I never thought that it would pay off." - Vanna White

Due: Tuesday, April 3, 2012 23:00

1 Objectives

- Solve embedded systems problems in C incorporating the C standard library.
- Construct valid C code using the gcc compiler targeting the ATMEGA32 development environment
- Use the C string library to manipulate program functionality
- Construct methods using pointers and pointer manipulation.
- Apply the const keyword to the definition of data arrays

2 Introduction

We have done a lot thus far in class. We started in assembly. Then moved to C. Then we started with interrupts. Now we will build a useful application (well, maybe).

In essence, you are to construct a program which will manipulate the display such that if no button is pressed, the top line shall display your first name and the bottom line shall display your last name.[1] If any button in the range 0 – 7 is pressed, the top line shall display a message corresponding to the letters of the Greek alphabet. This is shown in the table below.

Button	Message on top line
0	Button Alpha
1	Button Beta
2	Button Gamma
3	Button Delta
4	Button Epsilon
5	Button Zeta
6	Button Eta
7	Button Theta

The bottom line will display your last name while this is occurring.

[1] Note: If your name will not fit, shorten it so it fits.

If a button is pressed in the range 8 – *, then a message will be displayed with the names of the first 8 presidents on the bottom line, as is shown in the table below.

Button	Message on top line
8	G. Washington
9	John Adams
A	Thom. Jefferson
B	James Madison
C	James Monroe
D	John Q. Adams
#	Andrew Jackson
*	M. Van Buren

The top line will display your first name while this is occurring.

A video is available on the course website showing the operation of the program.

3 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.

4 Implementation issues

Your main loop will need to setup the LCD device driver through a call to the LCD Init method as well as handle the looping behavior. In general, if only one button is pressed, the LCD should simply update the window which has changed by calling the update lcd_setBottomMessage or lcd_setTopMessage methods prior to invoking a call to the lcdRefreshDisplay method, passing in either the parameter LCD_TOP_WINDOW, LCD_BOTTOM_WINDOW, or LCD_ALL_WINDOWS constant values representing whether the top, bottom, or all windows should be updated.

If more than one button was pressed, then a call to lcd_clearDisplay shall be made to cause a clear display command to be sent to the display hardware.

To protect the operation of the system, using your delay method, you should wait 50ms between main loop iterations. This will ensure that the display is not overburdened with updates faster than it can handle.

You will also need to use various functions from within the string library to accomplish this lab.

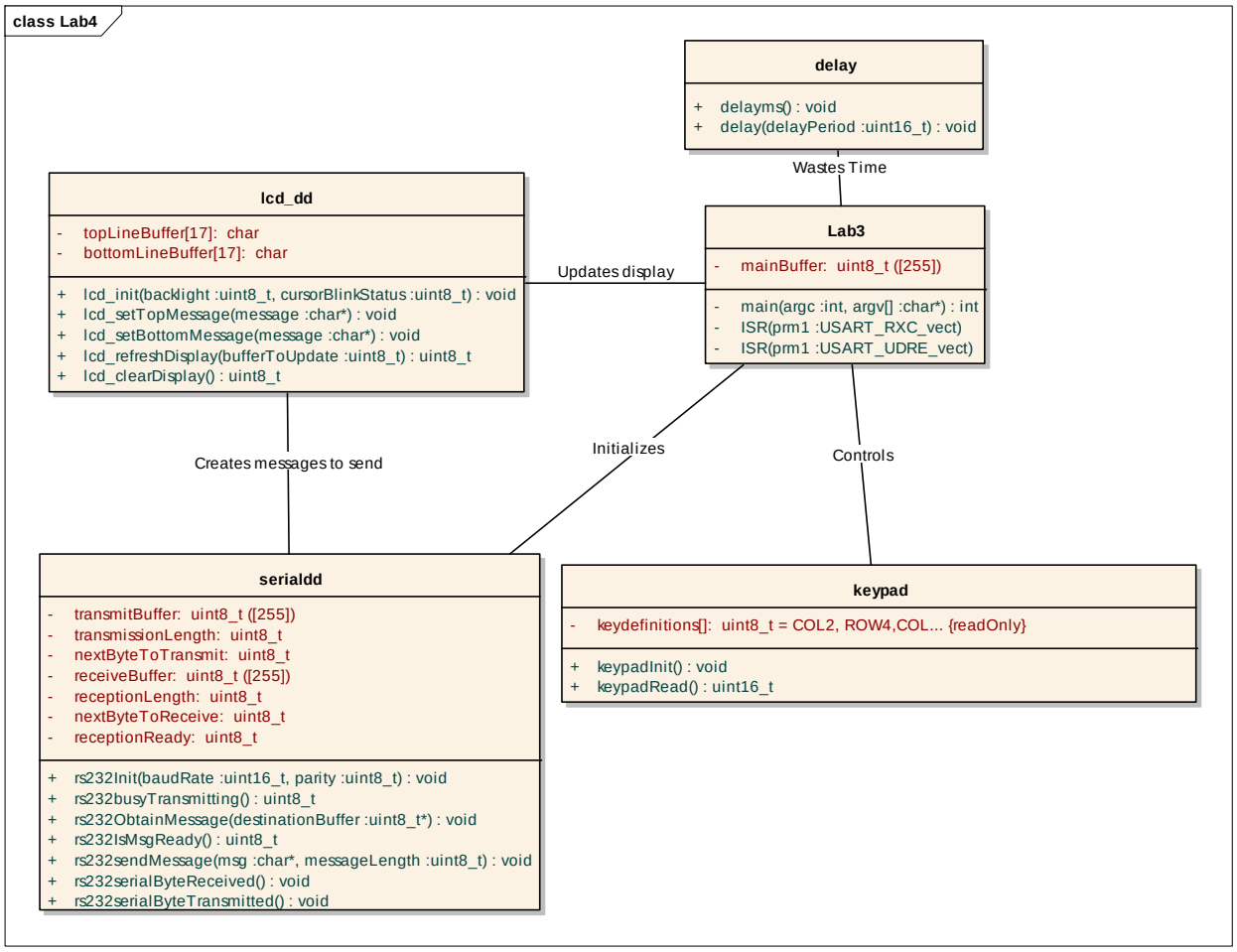

Figure 1 Suggested UML Class diagram for project.

5 Lab Assignment

6 Deliverables

6.1 Lab Report

Now that you have completed your lab assignment, submit the following lab report detailing your experiences. The lab report should be submitted electronically through the online submission form on the course website.

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?
    e. How does this compare with programs from CE2800?

   f. Into what address has your "main" method been placed (Hint: This is in the .lss file.)
3. Compile log
   a. Copy and paste the output of the compiler in the message window indicating that the code compiles without warnings or errors.
4. Screen Capture of Real Term Execution
   a. With RealTerm, capture the output of your program running showing that the program operates properly. You may need to adjust the settings to add newlines to the display after each message is received.
5. Things Gone Right
   a. What things went right in performing this lab?
6. Things Gone Wrong
   a. What problems did you have?
   b. What mistakes did you make?
   c. What material did you not remember from CE2800 that you needed for this lab?
7. Conclusions
   a. This section shall discuss what has been learned from this laboratory experience.
8. Source Code. Submit your well commented source code.

6.2 Project Demo

In addition to the lab report, you are required to demonstrate correct functionality. This can be accomplished prior to lab on April 4, 2012 or during the time period immediately following the lab quiz on April 4, 2012.

If you have any questions, consult your instructor.

Appendix A: API For display device driver

	Void lcd_init(uint8_t backlight, uint8_t cursorBlinkStatus)	This function will initialize the LCD display. In essence, it will send the appropriate byte strings to the device to cause the display to initialize into the appropriate mode. Parameters: Uint8_t backlight -> This will be a 0 if backlighting is off and a 1 if backlighting is on. Uint8_t cursorBlinkStatus – This is the state for the cursor. It may be LCD_CURSOROFFNOBLINK, LCD_CURSOROFFCHARBLINK, LCD_CURSORONNOBLINK, or LCD_CURSORONCHARBLINK.[2]
	Void Lcd_setTopMessage(const char* message)	This method will copy the butter using a string library function into the top line buffer. Parameters: char* message – This is the message that is to be displayed.
	Void Lcd_setBottomMessage(const char* message)	This method will copy the butter using a string library function into the bottom line buffer. Parameters: char* message – This is the message that is to be displayed.
	uint8_t Lcd_refreshDisplay(uint8_t bufferToUpdate)	This method will combine the top and bottom lines to make the appropriate updates to the lcd display. In short, if only the bottom line is to be updated, a buffer is to be created on the stack which contains an instruction to move the cursor to the start of the bottom line followed by the text that is to be sent. If only the top is to be done, it will be the same thing except that the cursor shall go to the home location. If both are to be done, then the buffer needs to be the combination of the two buffers and the commands to move the cursor to the appropriate locations.[3] Parameters: uint8_t bufferToUpdate This is the buffer to be written. It may be the value LCD_TOPMESSAGE, LCD_BOTTOMMESSAGE, or LCD_ALLMESSAGES. Return: 1 if the operation could be completed. 0 if the serialdd is busy transmitting and the message could not be sent using the send message method.
	void lcd_clear(void);	This function will clear the display, resulting in a blank display. It

[2] Hint: These are #define values that would exist within your header file so that an application can initialize the display in the appropriate fashion.
[3] Hint: You'll probably want to use the strlen function here, as well as possibly strcat and strcpy.

		should send the appropriate commands to clear the display.