CSCI 4415 Lab # 3

Serial Command Line Interface

Created: January 30, 2011

Objective:

- Learn how to use the Z16 UARTs by developing a simple command line interface for the ZNEO Contest Kit.

Time Estimate:

- 10 hours.
- You should be prepared to start this after class 5 (Serial Interfaces) and this will be due by Midnight class 8.

Specific Requirements:

Your program will implement as simple command line interface (or interpreter) on the ZNEO console port.

1. Connect your Z16 contest kit console port to your computer's serial port (you may need a USB-serial dongle for this). A serial cable is included in the lab kit.
2. Configure the Z16 UART0 to send and receive ASCII text from a terminal program running on your computer (putty is nice and free, hyperterm and minicom also work well). The terminal program can run on any computer or OS and need not run on the same computer as the ZDSII IDE. Use the default baud rate (57600).
3. Write a simple command line interpreter that allows you to set and query various Z16 configurations. You will need to include the LED library you developed for lab1 (HelloWorld) so you can display scrolling messages on the LED display.
4. Make your CLI prompt initially be "> "
5. Your CLI will include the following commands (and appropriate actions).

```
echo ["text"]
     - echos the "text" without the quotes to the serial port
     display ["text"]
     - displays "text" scrolling across the LED display
     set prompt ["text"]
     - changes prompt text. set prompt "C:\>" for example
     hex [decimal number]
     - displays the number in HEX on the LED
switch [n] ["text"]
     - Pressing a button causes "text" to be interpreted as
       though it was sent to the UART from the terminal (like a
macro function).
```

```
port [A-K]
      - Displays how the IO port [A-K] is configured (all
        settings, each bit).
timer [0-3]
      - Diaplys how timer n is configured and if enabled
uart0 [speed [baudrate]] [parity [even|odd|none]] [bits [7|8]]
      - sets the uart0 settings immediatly.
info
      - displays part number, oscillator setting, clock speed,
        serial port baud rate, CLI compile date & time.
?
      - displays a menu of CLI commands
```

7. Determine a technique that lets you process button push commands as normal input, perhaps inserting the "text" into the receive UART buffer when the switch is pressed and then treating as normal serial input.
8. You will need to consult the "Flash Memory" chapter of PS220 to see that the part number is stored in flash memory at addresses 0x000040-0x000053. You will need to consult the "Using the ANSI C-compiler" chapter of UM171 to learn how to access that. This part of the lab will require some thinking.
9. You will need to consult the "Using the ANSI C-compiler" chapter of UM171 to learn how to know the compile date and time.
10. Create the code for your CLI so that it is easily expanded with new commands and new settings. You will add to the in future labs.
11. Create a file named readme.txt file that includes any notes about the lab and answers to the questions below.
12. In the main.c file include the following information in a comment block

```
Author: [Your name]
Email:  [Your email]
Class:  [CSCI-4415]
Date:
Lab:    [Lab number]
Description: [short description of program. Include changes you made and
note any enhancements that you made to the lab]
Other files: [list other files that are necessary for this program]
Compile: [provide any special instructions for compiling. Only necessary if
there are special instructions]
Problems: [explaining what you did, what problems you had, how you solved them,
and what you might do differently if you had to do it again]
Comments: [Feel free to provide comments on how this lab went, what you think
is good or bad about it and how it could be improved or anything else you want
to say.]
Enhancements: [describe the enhancements you added to this lab]
```

10. Turn in a zip file (lastname-lab3.zip) of the entire project directory (include the entire ZDSII project directory and the readme.txt). Use BlackBoard to submit assignments (or if thats not working for some reason email the ZIP file).

References

1. http://en.wikipedia.org/wiki/Command-line_interface

2. Some open source printf libraries
    a. http://mes.sourceforge.jp/h8/printf.html
    b. http://www.sparetimelabs.com/tinyprintf/index.html
    c. http://www.physicsforums.com/archive/index.php/t-89474.html
    d. http://www.menie.org/georges/embedded/

Grading

- On time, in proper format (Zip file), named <lastname>-lab1.zip
- Compiles and runs without problems
- Meets lab requirements
- Reasonable code (partitioning of function, coding standard, readable
- Reasonable comments
- Use interrupts to send and receive
- Modular code (no side effects)
- Error detection and handling
- Proper use of Z16
- Answered questions

Enhancements:

- Add some more advanced commands to the CLI

```
set scroll [0,1..9,99]
      - how many times to scroll the message. 0 is none/off,
        99 for continuously loop. 10-98 return an error.
set scroll [fast|slow]
      - sets the scrolling rate to fast or slow
set
      - with no paramters lists settings (scroll speed,
        times, prompt, switch settings)
mem [address, length]
      - dump memory in hex start at address for length bytes
bin [decimal]
      - display number as 16 or 32 bit binary on the LED
clock
      - puts in clock mode (display hh:mm, count seconds on
        modem lights)
set time [hhmmss]
      - set the time for the clock function
set prompt $t
      - prompt diplays the current hh:mm:ss time
rttl ["text"]
      - parse and play text as RTTL (from lab 2)
```

- Write UART code that enables receiving and sending data using interrupts.
- Save settings in flash so the next time you power on the board your CLI settings are restored (baud rate, prompt, scrolling message, scrolling speed)

- Perform auto baud rate detection on UART0 when board powers on. Several ways to do this.
    - Analyse the received character and infer the real character and baud rate) http://www.iol.ie/~ecarroll/autobaud.html
    - Watch for the start and stop bits, time them and calculate the baud rate.
- Something you thought of

Questions

Include with your lab submission (in the comments of your main.c file):

1. What problems did you have using your LED library? Did you have to make any/many changes to it so it would work with this lab?
2. What problems or complications did using interrupts on transmit and receive cause for you?
3. How big is your input and output buffer? How did you determine the size? What happens when you exceed the buffer size?
4. How did you deal with pressing the text from a button press?
5. Describe (briefly) how you will add new commands to the CLI.