Lab #5
Extracting GPS Information by Packet Parsing

EELE 475
HARDWARE AND SOFTWARE ENGINEERING
FOR EMBEDDED SYSTEMS

Assignment Date: 9/23/14

Lab Report Due Date: 10/7/14

You have two weeks to get this lab done and <u>you will need to turn in a lab report.</u>

<u>Lab Description</u>

The DE2 board has a serial RS-232 port on it and today's lab is to extract GPS information from packets being sent to the NIOS processor by a GPS unit. You will use your GPS parsing program from HW#2 to parse the packets coming in on the RS-232 port. Like last week, you will need to use the following Altera tools:

- **Quartus II** - To compile the VHDL that creates your processor hardware.
- **Programmer** - To download your bit stream to the DE2 board. (QuartusII -> Tools -> Programmer)
- **Qsys** - To create your processor system. (QuartusII -> Tools -> Qsys)
- **NIOS II IDE** - To write, compile, and download your C program. (Qsys -> Tools -> Nios II Software build tools)

Your starting point will be the system you created last time (i.e. the LCD display). Create a new lab5 directory and copy all your files from lab4 into this directory.

<u>Part 1 – Qsys Modification</u>

1. Add the UART Peripheral, which is found at: Interface Protocols->Serial->UART (RS-232 Serial Port) . *<u>Note: This is not the JTAG UART peripheral.</u>*

2. Name it: uart_RS232.

3. Set the Baud rate to ***<u>9600</u>*** bps, the Parity to ***<u>None</u>***, the Data bits to ***<u>8</u>***, the Stop bits to ***<u>1</u>***, and the synchronizer stages to **<u>2</u>** (default). Don't include flow control or DMA control. Connect the clock, reset, data, and export the external connection named RS232. Connect the IRQ as IRQ 1.

4. Auto-Assign the Base addresses again and regenerate the Qsys system.

5. Connect the RS232 signals in the nios_system component to the DE2 entity in the file DE2_Board_top_level.vhd. (Remember to update the interface signals in the component entity and in the instantiation of the nios_system, and comment out the UART_TXD signal).

6. Make sure that the DE2_Board_top_level is set as the top level entity, compile your VHDL code and download the new bit stream to the DE2 board. Make sure the programmer is pointing to your new bit stream file and not to the old one, which it will do by default (a common mistake).

7. Note: you should have the following Qsys components in your design:

    a. cpu (lab 2)
    b. sdram (lab 3)
    c. jtag_uart (lab 2)
    d. LEDs (lab 2)
    e. Switches (lab 2)
    f. LCD (lab 4)
    g. UART (lab 5)

Part 4 – Software

- Open up the NOIS II IDE (you can do this from within the Qsys -> Tools -> Nios II Software build tools).

8. Set the workspace to the directory Eclipse_Workspace that you created.

9. Select File->New->*Nios II Application and BSP from Template*.

    a. For the SOPC Information File name, browse to the file nios_system.sopcinfo found in the DE2_System directory (or where you created your project). Note: the default will be your old directory, so change it to the new one you just created.

    b. Name your Project name "gps".

    c. Select the Project Template *Blank Project* (and click finish).

10. We now need to set where the **stdio** is coming from. Do this by selecting NIOS II -> BSP Editor. Open the file settings.bsp that is found in the directory gps_bsp. Set stdin to uart_RS232 and stdout to the jtag_uart (so you can print your packets to the console). Regenerate the bsp and then click exit (leaving the bsp editor open can cause problems when compiling the program – Yes, Eclipse is brittle....).

11. Compile and Run the following code by pasting it into a new source file called gps.c

```
#include <stdio.h>

int main()
{
   char c;
   while(1){
      c=getchar();
      printf("%c",c);
   }
   return 0;
}
```

You should see the GPS packets being printed out on the console window.

12. Implement your code (HW#2) to parse the character stream coming from the RS-232 port. At a minimum you need to extract information from the GPGGA packet and check if the checksum is correct. In addition you will need to check the status of the GPS unit to see if the data is valid or invalid (it is invalid when starting up). If the checksum is correct and the data valid, you will need to display the following information on the LCD screen:

If SW0 is on:

Lat: xxxx {N or S} Long: xxxx {E or W}

If SW0 is off:

Elev: xxxx (ft) Time: xxxx (local) {AM or PM}

where Latitude and Longitude are just the character strings from the packet, elevation is in feet (you will need to convert from meters), and the time (hours:minutes:seconds) is ***local time*** with AM or PM displayed. {N or S} means that either just the N character or S character is to be displayed.

If the data is invalid, the following message must be displayed (regardless of SW0):

GPS Unit is not ready. Data is invalid

In addition, you will need to convert the Latitude and Longitude character strings to degrees (***float data type***) and print this to the console window with the following two printf() statements:

printf("Latitude = %f degrees\n", degrees_latitude)

printf("Longitude = %f degrees\n", degrees_longitude)

A brief outline of the code you will need is:

```
#include <stdio.h>
#include <string.h>

int main(){
     while(1){
     c = getchar();  // incoming character
    if( c == '$' ){
        … //start of packet
    }
    //  is it the GPGGA packet?
    //  is the checksum correct?
    //  is the data valid (i.e. GPS startup completed)?
    // if so, get the latitude, longitude, elevation, and time information
    // display this information on the LCD screen
    // convert latitude and longitude to degrees (float data type)
    // print these float values to the console window.

  }

}
```

13. For the GPGSA packet, determine the satellite ID numbers, print the ID numbers to the console window, and then light up the corresponding red LED for each satellite ID number found (you can also use the green LEDs for values greater than the number red LEDs).

Have the instructor verification sheet (given below) signed off when you get the GPS information on the LCD and the float values printed to the console window.

The lab will require a lab report. The required format is described below as well.

For the lab report, create a block diagram of all the components you have added so far. This figure should go into the introduction section.

Instructor Verification Sheet
Include this as the back page of your lab report
to get credit for Lab #5

Lab #5
Extracting GPS Information by Packet Parsing

EELE 475
HARDWARE AND SOFTWARE ENGINEERING
FOR EMBEDDED SYSTEMS

Name : _____

Demo #1 : Show the GPS packets being printed to the console window.

Verified: _____ Date:_____

Demo #2 : Show that the LCD displays the appropriate GPS information reflecting the switch[0] position (both on/off).

Verified: _____ Date:_____

Demo #3 : Show that the latitude and longitude float data types are printing to the console window in <u>degrees</u> and using a float data type.

Verified: _____ Date:_____

Demo #4 : Show that the lit red LEDs corresponds to the satellite ID numbers.

Verified: _____ Date:_____

Memo to: EELE 475 Students
From: Ross Snider
Date: October 1, 2013
Regarding: Required Lab Report Format

Write up your lab reports in the following format:

Header: Include a simple, informative header on your report.

To:	Prof. Ross Snider
From:	Your Name
Regarding:	Lab #X – *Lab Title* (**Put lab number, then title**)
Date Performed:	September 27, 2011 (the date the lab was performed)

Summary:
The Summary is intended to be a complete description of the lab experiment, but written succinctly for a knowledgeable reader. Choose your words carefully and *engineer* your summary to be brief, but effective.

Introduction:
Give the big picture overview of what you did. Include any block diagrams that give the big picture. Give an outline of the steps you performed.

Body:
Go into the details of the big picture. What did you do for each step? Describe any sub-blocks of your system. How does your code parse the GPS packets?

Note: All figures must have captions underneath them that describe what the figure is about. As a general rule, all figures should be self-contained: you should be able to understand what is going on in the figure by just looking at the figure and reading the caption.

Concluding Section:
If you wish to expand on the Summary given at the top of the memo, a concluding section might be given. You could entitle it simply *Conclusion*, or *Recommendations for Further Work*, *Summary Comments*, or any other title that tells what the boss is going to see when reading the conclusion. Remember, the goal of any written or oral presentation is to communicate effectively.

Appendix:
Place print-outs of your code in an appendix and clearly mark what section & problem number(s) the code was for. The code should be well commented.

Upload the report and code to the D2L dropbox.