To: Prof. Ross Snider

From: Kameron Kranse, Kasy Treu, and Abdulaziz Alsaleh

Regarding: Lab #7 – GPS LAB

Date Performed: November 10, 2015

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.*

This lab experiment goal was to receive data from a GPS module, check if the data received is valid and if that data is valid then process the data into readable information, and output it onto both the LCD screen on the FPGA and the console window within the programming IDE.

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.*

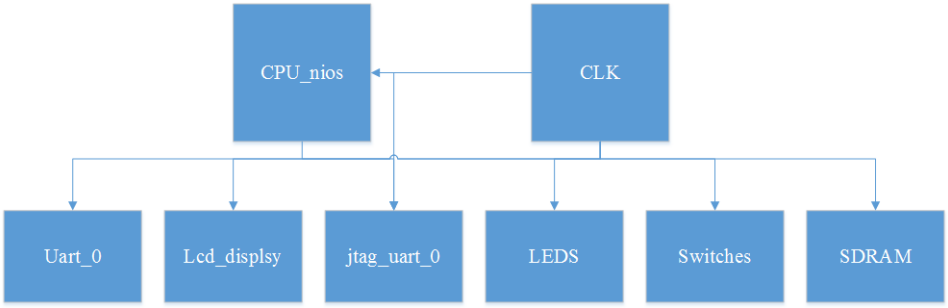

Figure 1: Block Diagram of System

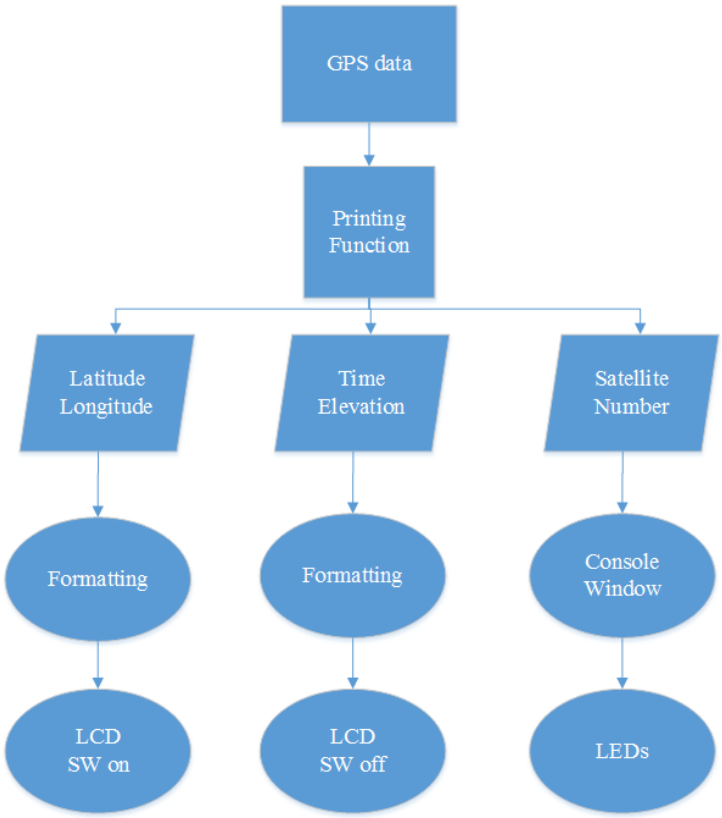

Figure 2: GPS Data Flow Chart

The DE2 board has a serial RS-232 port on it, using the serial port, we extracted GPS information from packets being sent to the NIOS processor by a GPS unit located on the top of Cobleigh Hall. Using a GPS parsing function from a previous assignment, we parsed the packets coming in on the RS-232 port. Further developing the code we initially used, we added the function to be able to print the Latitude, Longitude, elevation, and local time from the GPS to the LCD screen on the DE2 Board, and display the satellite IDs it's connected to on the red LED's on the board.

***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?*

The GPS sends information in forms of packets to the system. Each of these packets has a starting character and an ending character. This allows us to analyze and only grab the packets we need. By reading in the data character by character, we can obtain only the packets we need. Once the packet that is desired is obtained, a verification process needs to be completed. This process takes each character obtained and breaks them down into the binary strings. Each of these binary strings are then xored with each other and broken apart into two nibbles. These two nibbles are compared to the last two characters in the data called a checksum. If the two nibbles match the checksum the data is valid and correct. Otherwise that packet is ignored and the next packet is obtained. Once the packet is verified, the packet itself is analyzed piece by piece and formatted to the desired output that allows for easier readability. For example. the time is sent in a packet in Universal Coordinated Time. This needs to be converted into local time along with split into hours, minutes and seconds rather than a single string of numbers. The time also needs

to be formatted from 24 hour time into 12 hour am/pm time. The formatted GPS coordinates, elevation, and time are all formatted to a desired format and then printed onto the LCD screen of the FPGA and the console window within the programming IDE. There is a switch that has been programmed on the FPGA that allows the user to switch between viewing longitude and latitude coordinates and viewing elevation and time. A second packet is obtained to gather which satellites are being used by the GPS. These satellites are shown through a series of LEDs that have been programmed on the FPGA. The first LED (LEDR0) corresponds to satellite 1, and LEDR17 corresponds to satellite 18. There were not enough LEDs in order to show all the possible satellites that were being used so only the first 18 were shown.

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.*

The device works as assigned, but has capability for improvement.

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.

```
/*
 * gps.c
 *
 *  Created on: Nov 3, 2015
 *      Author: Kasy Treu, Abdulaziz Alsaleh, Kameron Kranse
 */
```

```
#define Switches (volatile int *) 0x01001010
#define LEDs            (int *) 0x01001020

#include<stdio.h>
#include <stdlib.h>

int main() {

        char c;
        char dataSet[80];
        int calculatedChecksum = 0;
        int checkSumFromData;
        int ledWrite = 0;
        int sats[12];
        char checksum1;
        char checksum2;
        char str[5];
        char t[2];
        float latitude;
        float longitude;

        FILE *lcd;

        lcd = fopen("/dev/lcd_display", "w");

        while (1) {
                c = getchar();
                if (c == '$') {
                        c = getchar();
                        if (c == 'G') {
                                dataSet[0] = c; //if the first character in the line after the $ is a G, store to array
                                c = getchar();
                                if (c == 'P') {
                                        dataSet[1] = c; //if the first character in the line after G is P, store to next array spot
                                        c = getchar();
                                        if (c == 'G') {
                                                dataSet[2] = c; //if the first character in the line after P is G, store to next array
spot
                                                c = getchar();
                                                if (c == 'G') {
                                                        dataSet[3] = c; //if the first character in the line after G is G, store
to next array spot
                                                        c = getchar();
                                                        if (c == 'A') {
                                                                dataSet[4] = c; //if the first character in the line after G
is A, store to next array spot

                                                                int i = 5;
                                                                while (c != '*') {
                                                                        c = getchar(); // put all values in a char[] to
XOR

                                                                        dataSet[i] = c;
                                                                        i++;
                                                                }
                                                                //initialize the checkSum
                                                                calculatedChecksum = 0;
                                                                int k;
                                                                //calculate the checkSum
                                                                for (k = 0; k < i - 1; k++) {
                                                                        //xor each value within the packet before the
checksum characters
                                                                        calculatedChecksum ^= (int) dataSet[k];
                                                                }
                                                                //check the checksum characters
                                                                c = getchar();
```

```
checksum1 = c;

c = getchar();
checksum2 = c;

//build a string out of the read in checksum values into a
hex value

str[0] = '0';
str[1] = 'x';
str[2] = checksum1;
str[3] = checksum2;
str[4] = '\0';

//convert this hex value into an integer for comparison
to the calculated checksum

sscanf(str, "%X", &checkSumFromData);
//if the checksum is correct, continue
if (calculatedChecksum == checkSumFromData) {
        //grab the hours portion of the time
        char myarray[2] = { dataSet[6], dataSet[7] };
        int time;
        //convert from universal correlated time into
local time

        sscanf(myarray, "%d", &time);
        time = time - 7;
        //convert 24 hour time into 12 hour time
with AM/PM

        if (time > 12) {
                time = time - 12;
                t[0] = 'P';
                t[1] = 'M';
        } else {
                t[0] = 'A';
                t[1] = 'M';
        }
        //update the time portions of data set.
        sprintf(myarray, "%d", time);
        dataSet[6] = myarray[0];
        dataSet[7] = myarray[1];
        //grab elevation portions of data set
        char Elevarray[6] = { dataSet[53],
                              dataSet[54],
dataSet[55],
                              dataSet[56],
dataSet[57],
                              dataSet[58] };
        //convert the elevation values into feet
        float Elev;
        sscanf(Elevarray, "%f", &Elev);
        Elev = Elev * 3.2808;
        //update the elevation values of the data set
        sprintf(Elevarray, "%f", Elev);
        dataSet[53] = Elevarray[0];
        dataSet[54] = Elevarray[1];
        dataSet[55] = Elevarray[2];
        dataSet[56] = Elevarray[3];
        dataSet[57] = Elevarray[4];
        dataSet[58] = Elevarray[5];

        //case based around if the first switch if
flipped or not

        switch (*Switches) {
        //if the first switch is not flipped write the
elevation and time to the LCD

        case 0:
```

```
"Elev:%c%c%c%c%c%c FT\n",
dataSet[53], dataSet[54],
dataSet[55], dataSet[56],
dataSet[57], dataSet[58]);

"TIME:%c%c:%c%c:%c%c %c%c\n",
dataSet[7],
dataSet[9],
dataSet[10], dataSet[11], t[0],


latitude and longitude to the LCD


%c%c%c%c%c%c%c%c %c\n",
dataSet[16], dataSet[17],
dataSet[18], dataSet[19],
dataSet[20], dataSet[21],
dataSet[22], dataSet[23],
dataSet[27]);
* 10)
(dataSet[17] - 0x30))
((((dataSet[18] - 0x30) * 10)
        + (dataSet[19] - 0x30))
        * .0167);
latitude);

%c%c%c%c%c%c%c  %c\n",
dataSet[29], dataSet[30],
dataSet[31], dataSet[32],
dataSet[33], dataSet[34],
dataSet[35], dataSet[41]);

(((dataSet[29] - 0x30) * 100)
        + ((dataSet[30] - 0x30)
                * 10)
                fprintf(lcd,


                fprintf(lcd,
                                dataSet[6],
                                dataSet[8],


                                t[1]);
                break;
//if the first switch is flipped write the
case 1:
                fprintf(lcd,
                                "Lat:


                latitude = (((dataSet[16] - 0x30)
                                +
                                +


                printf("Latitude = %.2f\n",
                fprintf(lcd,
                                "Long:


                longitude =
```

```
+ (dataSet[31] - 0x30))

+ ((((dataSet[32] - 0x30)

                * 10)

                + (dataSet[33]

                                - 0x30))

                * .0167);
                                                                printf("Longitude = %.2f\n",
longitude);
                                                                break;
                                                        }
                                                //if the checksum fails, print to the LCD that the GPS is
not ready because data is invalid
                                                } else {
                                                        fprintf(lcd,
                                                                "GPS Unit not
rdy.\nData is invalid\n");
                                                }
                                        }
                                //check for satelite packet
                                } else if (c == 'S') {
                                        dataSet[3] = c;
                                        c = getchar();
                                        if (c == 'A') {
                                                dataSet[4] = c;
                                                int l = 5;
                                                //read all characters before the checksum characters
                                                while (c != '*') {
                                                        c = getchar();
                                                        dataSet[l] = c;
                                                        l++;
                                                }
                                                int m;
                                                calculatedChecksum = 0;
                                                //xor each value within the packet before the checksum
characters
                                                for (m = 0; m < l - 1; m++) {
                                                        calculatedChecksum ^= dataSet[m];
                                                }
                                                //get the first checksum character
                                                c = getchar();
                                                checksum1 = c;
                                                //get the second checksum characters
                                                c = getchar();
                                                checksum2 = c;
                                                // build a string out of the read in checksum characters
into a hex value
                                                str[0] = '0';
                                                str[1] = 'x';
                                                str[2] = checksum1;
                                                str[3] = checksum2;
                                                str[4] = '\0';

                                                // then convert the hex string into an integer
                                                sscanf(str, "%x", &checkSumFromData);

                                                //check the two checksum values, if correct proceed
                                                if (calculatedChecksum == checkSumFromData) {
                                                        //print out the satellites
                                                        printf("Numbers of satellites: ");
```

they are always there

and the commas but counting commas to know when to stop

always 8 commas

satellite reading

ready to the LCD

rdy.\nData is invalid\n");

turn on

(dataSet[11]-0x30));

(dataSet[14]-0x30));

(dataSet[17]-0x30));

(dataSet[20]-0x30));

(dataSet[23]-0x30));

(dataSet[26]-0x30));

```
		//use the commas as a counting system since

		int comma = 0;
		int place = 10;
		while (comma < 8) {
			//print out the satellite numbers

			printf("%c", dataSet[place]); //

			if (dataSet[place] == ',') {
				comma++;
			}
			place++;
		}
		//print a new line character between each

		printf("\n");
	} else {
		//if the checksum fails, print GPS is not

		fprintf(lcd,
				"GPS Unit not

		printf("Sats incorrect, checksum failed\n");
	}
	//gather the satellite numbers to know which LEDs to

	//builds an array of integers holding these values
	if(dataSet[10] == ','){
		sats[0] = 0;
	}else{
		sats[0] = (((dataSet[10]-0x30)*10) +

	}
	if(dataSet[13] == ','){
		sats[1] = 0;
	}else{
		sats[1] = (((dataSet[13]-0x30)*10) +

	}
	if(dataSet[16] == ','){
		sats[2] = 0;
	}else{
		sats[2] = (((dataSet[16]-0x30)*10) +

	}
	if(dataSet[19] == ','){
		sats[3] = 0;
	}else{
		sats[3] = (((dataSet[19]-0x30)*10) +

	}
	if(dataSet[22] == ','){
		sats[4] = 0;
	}else{
		sats[4] = (((dataSet[22]-0x30)*10) +

	}
	if(dataSet[25] == ','){
		sats[5] = 0;
	}else{
		sats[5] = (((dataSet[25]-0x30)*10) +

	}
	if(dataSet[28] == ','){
		sats[6] = 0;
```

```
                                                                                }else{
                                                                                        sats[6] = (((dataSet[28]-0x30)*10) +
(dataSet[29]-0x30));
                                                                                }
                                                                                if(dataSet[31] == ','){
                                                                                        sats[7] = 0;
                                                                                }else{
                                                                                        sats[7] = (((dataSet[31]-0x30)*10) +
(dataSet[32]-0x30));
                                                                                }
                                                                                int p;
                                                                                //loops through the entire array of satellite values and
builds an integer to turn on LEDs
                                                                                for(p=0; p<8; p++){
                                                                                        if(sats[p]<19){
                                                                                        ledWrite |= (1 << (sats[p]-1));
                                                                                        }
                                                                                }
                                                                                //that value is written to the LEDs to turn the correct
ones on.
                                                                                *LEDs = ledWrite;

                                                                        }
                                                                }

                                                        }

                                                }

                                        } else {
                                                //if the char isn't $ move on the find the next character.
                                                c = getchar();
                                        }
                                }
                                return 0;
}
```