ECGR 4101/5101, Fall 2011: Lab 4

A pseudo physics simulator with remote communication.

Learning Objectives:

This lab will test your ability to apply the knowledge you have obtained from the last three labs to and expand that knowledge further by introducing you to serial communication via (UART).

Note: be warned, this lab expects that you are very comfortable with the Lab1, Lab2, and Lab3 procedures for creating a HEW project from scratch and implementing the custom LCD interface. Details on these steps will be skipped in this lab as it is expected that you already know how to do this.

Additionally, this lab will not provide as much step by step detail as lab three did so be prepared to spend additional time researching datasheets, reading over your class notes, and using the age old process of trial and error. I also recommend that you get acquainted with the HEW debugger as being able to add breakpoints and variable watches is very useful when trying to figuring out why your code isn't working correctly.

General Information:

1. Review the steps provided by the lab 1 supplemental information document
2. Review the steps provided by the lab 2/3 Activity
3. Copy the necessary lab 3 files into your new workspace and import them
4. Import the necessary mathematical library
5. Get your pseudo physics simulator from lab3 working in a new lab project
6. Add additional code to handle serial communication
7. Demonstrate your working project to the TA, and turn in a lab report.

Prelab Activity:

You may use the PCs in Woodward 203 or your own PC to do this lab experiment. If you want to work on lab assignments on your own PC, then you will need to load the necessary tools on your PC in order to perform this exercise.

In this lab you will need to create a new project called lab 4 that is exactly like lab 3.

You will also need to review your class notes, the Renesas API document, and possibly look at the available sample applications to see how to setup the UART for serial communication.

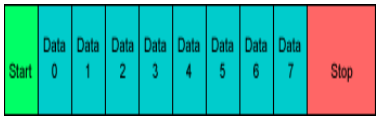
Some Helpful Advice:

you should also look over the documentation from Renesas API found at
http://www.renesas.eu/media/products/software_and_tools/introductory_and_evaluation_tools/european_pdfs/RPDL_RX62N_API_UsersManual.pdf

Prelab Questions:

1. Where should your "working directory" be located when using your home computer?
2. What is the worst place to look for help with labs in this class?
3. What is UART and how does it work?
4. How is UART related to RS232? (explain how is it similar and different)

Laboratory Assignment Overview:



Learning about UART

In the last lab you learned a few advanced LCD topics along with learned how to both collect data from the accelerometer and process that data to obtain velocity and position in three dimensional space.

In this lab we will expand the functionality of the physics simulator slightly by making a few additional modifications to the lab 3 code, and we will also be adding serial communication to the project.

So let's begin!

Laboratory Assignment:

1. Do the prelab activity described above.
2. Once you have your new project up and running with the lab 3 code preform a quick test to ensure that your project is working as intended.
3. At this point, from the last lab we discussed that there was a correlation between the simulation step size (delta_t) and the speed at which the ball rolled around the screen. In this lab we want to change the value of delta_t by using the on board potentiometer. You will need to review your lab 1 and 2 code and figure out how to implement this functionality onto your board.

   Note: you only have 4 CMT timers available via R_CMT_Create, so use them wisely

4. Once you have the ball rolling around the screen at different rates of speed (subject to change as a function of the potentiometer), you will need to add switch support to your project
5. At this point, you will need to add pause support to your physics simulator when SW1 is pressed. The LCD screen should clearly display that the simulator is paused at the top of the screen and when SW1 is pressed again the physics simulator should return to its normal operating mode.

   Note the simulator should not run while it is paused
   Tip: your text should be rendered to the screen after the screen buffer is rendered to the screen.

6. Now that we have pause support for the physics simulator we need to add a way to allow us to change other variables inside the simulator, such as (gx,gy,gz) in order to emulate gravity on other worlds. To do this we need to create a menu of options that are only available when the simulator is paused.

   To illustrate this concept, when the simulator is paused the LCD should appear as follows

Paused
-------Menu------
>Gx: (current value of Gx is displayed here)
Gy: (current value of Gy is displayed here)
Gz: (current value of Gz is displayed here)

   Where ">" is used to indicate the current selection and the potentiometer is used to change the current selection. (the selection will move up and down when the potentiometer is turned)

   Tip: to do this you can take the total range of values from the ADC and divide it into 3 parts such that each part represents a menu section of an item on the screen.

   Note: you will need to add code inside your ADC above to prevent it from changing the simulation step size (delta_t) via the ADC while the simulation is paused

   In order to print floating point numbers to the LCD screen you will need to write a function called MakeCharArrayFromFloat that will take a floating point value into the program and convert that floating point value into an ASCII char array without using any standard libraries to perform the conversion (IE no sprintf)

   To keep things simple you can assume that the minimum value your floating point number can have is -999.9999 and the maximum value it can have is 999.9999, this implies that your function will use a char array with a maximum size of 9 to 10 in length. (its 10 if you include \0 at the end) A Example prototype is "void MakeCharArrayFromFloat(char * asciichar,float data)"

   Once you have a function to convert a float into a char array you should use it to display menu information to the LCD for gx,gy,gz, as shown above.

   An Example for my values of gx,gy,gz is

Paused
-------Menu------
>Gx:0.3111
Gy:0.297
Gz:0.297

Now that a system is in place to display current gx,gy,gz data lets add a system to modify it from the menu
When an item on the menu is selected, Pressing SW2 will increase its value, while Pressing SW3 will decrease its value

Note you should define a range of values that gx,gy,gz can be, and a incrementing and decrementing step size each time SW2 and SW3 are pressed. (Make your range vary from a negative value to a positive value)

Once you have the menu implemented, test your code. The simulator should run as normal at boot up and you should be able to change the step size (IE ball movement speed) via the potentiometer.
Pressing the SW1 button should pause the simulator and bring up the menu, and you should be able to select items on the menu via the potentiometer and change their value via SW2 and SW3.
Pressing SW1 again will un-pause the simulator and the simulator should run as normal with the new values of gx,gy,gz and the potentiometer should once again change the IE ball movement speed.

7. At this point you will need to figure out how to setup the UART on the board to the following specifications.

Baud	19200
Data Bits	8
Parity	Even
Stop Bits	1
Flow Control	None (only worry about this when you use hyperterminal on the PC)

Note: there is a RX API for the UART, However, I would like you to explore how to do this without using functions from "r_pdl_sci.h"

You should read the manuals included in the
C:\Program Files (x86)\Renesas\Hew\Manuals\Renesas\PDFs
Folder for information about the procedure needed to setup the UART via registers.

You will need to obtain a RS232 cable and a RS232 to USB converter and it is possible to check one out if you do not already own one. (See the TA for details)

The lab computers have hyperterminal available; however hyperterminal does not come standard with windows vista or above. If you want to do this lab on your own computer you will need to find another program to read and display serial data on the PC. I personally

recommend using the pyserial module for python and writing your own python console app, however other simplistic solutions do exist on the internet.

Because UART is either hit or miss in terms of functionality, I recommend that you create an Asynchronous Serial sample project and get that project communicating with a PC to test that your hardware is working. Once you have that working, go back to your lab 4 project, and implement UART into your own project, again without using the nice RX API.

You will want to start out simple when testing your own UART code, I recommend that you create another timer that is called every 400ms that will write the letter A to the PC and once you get that working you are ready to move to the next step.

8. Once you have the UART working, every 1 second your program should write the current x,y,z position of the ball on the screen out of the UART to the PC in ASCII if the simulation is running or write the words Paused to the PC if the simulation is not running.

   Note this implies you will need to convert the floating point numbers of x,y,z to ASCII via the function you wrote MakeCharArrayFromFloat prior to writing it to the PC via the UART.

   Note typically UART devices can only handle one character at a time, and if you try to write many characters in rapid succession you will "clog the buffer" and some characters will not be sent, you should include a short delay between each character write to avoid "clogging the uart buffer" the amount of delay you need between each write is something you will have to experiment with. A for loop and a no operation command can be used to make a quick delay, or the API might have a delay command available, and you will need to research this.

   Note you should never put a delay inside a function that is called by R_CMT_Create. Use the function called by R_CMT_Create to toggle a flag that the main function can use to execute the UART transmit code that had delays.

9. Bring the new board to the lab TA and demonstrate the new code (without the HEW application running). When the TA checks your board, he will also take your lab report. You will not need to include a printout or soft copy all of the code – just "snippets".

Lab Report:

1	Has the same fundamental functionality of Lab 3	
2	Simulator pauses when SW1 is pressed and un-pauses when SW1 is pressed again	
3	Simulator step size changes via the potentiometer when simulator is running	
4	Simulator shows menu when paused that is navigated by potentiometer	
5	Simulator menu values of gx,gy,gz can be changed using SW2 and SW3	
6	Simulator responds to changes of step size, gx,gy,gz	
7	Simulator uses UART to communicate with the PC and displays x,y,z position every 1 second or paused if the simulator is not running	

Include in your lab report observations and procedure like the following:

The general learning objectives of this lab were . . .

The general steps needed to complete this lab were . . .

Some detailed steps to complete this lab were . . . .

1. *Step one*
2. *Step Two*
3. *. . . .*

Code generated or modified to complete this lab…

No need to include all the files for the lab. Just include the modified code.

Some important observations while completing/testing this lab were . . .

In this lab we learned . . . .