ECGR 4101/5101, Spring 2012: Lab 1

Testing the Renesas RX62N Demonstration Kit (YRDKRX62N)

Learning Objectives:

This lab will allow you to validate that your RX62N demo board is functional by demonstrating how to create a new project in HEW and load the compiled executable onto your RX62N Demo board.

General Information:

1. Attempt to power up the RX62N demo board and observe the pre-loaded test program. If the board is not pre-loaded (*id est*, no LED's are blinking), you will have to download a program to the RX62N board *via* HEW by following the RX62N quick start guide.
2. Follow the steps provided by the lab 1 supplemental information document and load the RX62N tutorial project onto the RX62N board.
3. Modify the tutorial project to perform the specified requirements
4. Build the project and load it onto your RX62N Board. Run the program and observe the operation.
5. 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.

Test your board by attaching it to a computer *via* USB. Ensure the LCD lights up and if the original Renesas test program is loaded that lights are flashing and text is displayed on the screen.

By default the Renesas tutorial program will blink the LED ring slowly until any button is pushed or a defined amount of time has passed after which the light ring will flash at a rate proportional to the potentiometer. Another button push will change text on the LCD screen for a few second and then change the text back to its original value.

Prelab Questions:

1. Where should your "working directory" be located when using the lab computers?
2. Where is the first place to look for help with labs in this class?
3. What software is required to program the RX62N Demo Board?
4. What languages can be used to program the RX62N Demo Board?

Laboratory Assignment Overview:



A Model Train in the Digital Age…

"After four years of studying to be an engineer, I wonder when they are going to let us drive the train?" - *Unknown senior overheard in the hallway.*

Model trains are a great source of amusement for both kids and adults alike and what better way to dive into embedded systems than creating your very own digital model train!

For those of you who have never played with a model train set before or have just forgotten, a typical model train set comes with small conductive rails, a small electric locomotive that rides on top of the conductive rails, freight cars or passenger cars that are pulled by the locomotive, and a small power supply that is connected to the conductive rails to power the locomotive. A typical train power supply has the ability to change the rail polarity in order to change the direction of the train and a variable voltage source to vary the speed of the train.

In order to build a digital model train we will substitute the conductive rails and train with the circular LEDs ring on the RX62 demo board and use the RX62 demo boards buttons to control the train's direction while using the RX62 demo boards potentiometer to control the trains speed. We can also use the RX62 demo boards LCD display to write information about our digital train for easy viewing.

Requirements:

Req. 1. The code generated is written in C for the RX62N Evaluation Board.
Req. 2. The code is well commented and easy to follow.
Req. 3. LED 4 to LED 15 symbolically represents the model train track. Note there are 12 LEDs total or 12 track segments in our model railroad.
Req. 4. When a LED is off it means that there is no train on that section of track.
Req. 5. When a LED is on it means that either a locomotive or train car is on that section of track.
Req. 6. Our model railroad will have a total of 4 cars on the track and this implies that at any given time there should be always be only 4 LEDs actively emitting light.
Req. 7. When SW1 is pressed the train will stop, and the LEDs will not move.
Req. 8. When SW2 is pressed the train will move forward and the LEDs will move in a clockwise motion.
Req. 9. When SW3 is pressed the train will move backwards and the LEDs will move in a counterclockwise motion.
Req. 10. The LCD should say what direction the train is currently moving in or report that the train has stopped.
Req. 11. Turning the potentiometer clockwise will cause the train to slow down when moving.
Req. 12. Turning the potentiometer counterclockwise will cause the train to speed up when moving.

Laboratory Assignment:

1. Review the instructions given by the Lab 1 Supplemental Information document or RX QuickStart guide. Note carefully the steps required to build a program, download, and debug a program.
2. Create a Tutorial Project for the RX Demo Board and call the workspace Train. Download and run this new code. Note the change of behavior of the board. (if you get complier errors at this point you will need to review the Lab 1 Supplemental Information document for help on fixing them)
3. In the project editor, choose main.c as the file to edit. You will need to change the program in a few places to satisfy the project requirements:
    a. At boot-up, instead of showing "Renesas " on the LCD, show your first name
    b. At boot-up, instead of showing "RX62N" on the LCD, show “Train Demo”
    c. At boot-up, Add an additional LCD message to the third LCD line that says “Mode:Stop”
    d. Modify the code to prevent the static variable test function “Statics_Test()” from running
    e. Change the while (1) to while (1){} since you will be adding additional code inside the {} to handle button and LCD updates later in the project

    Download and run this new code. Note the change of behavior of the board.
4. In the project editor, choose flashLED.c as the file to edit. You will need to change the program in a few places to satisfy the project requirements:
    a. At boot-up FlashLED () is executed, Modify this function such that it will only exit this function when either (SW1,SW2,SW3) is pressed (hint look at what gFlashCount is doing)

    Download and run this new code. Note the change of behavior of the board.
5. Continue to edit the flashLED.c
    a. Create a new function that will turn LED4 to LED15 off (*id est*, All LEDs are off) (hint the function R_IO_PORT_Write(LED4,value(1 or 0)) can be used to toggle pin values, also be aware that this function can only modify 1 port at a time) (Note in order to use the keywords LED4,LED5,LED6,…. You will need to add #include "YRDKRX62N.h" at the top)
    b. Create a new function that takes in an input value between 4 to 15 and turns the corresponding LED on while leaving the remaining LED unchanged. Values outside of 4 to 15 should be ignored.
    c. Create a New function that takes in an input value between 4 to 15 and will light up 4 consecutive LEDs that correspond to the location of the train. For example, if the number 4 was passed into this function then LED4, LED5 LED6, and LED7 should light up. LED4 would correspond to the train locomotive (start) while LED7 would correspond to the train caboose (end). To provide another example, if the number 15 was passed into this function then LED15,LED4,LED5,LED6 should light up while if the number 14 was passed into this function then LED14,LED15,LED4,LED5 should light up. This function should turn off all LEDs that are not used to represent the train’s location. (hint be sure to use the functions you created earlier to help you in this task)

   d. Modify CB_CMTFlash such that it no longer blinks the LEDS but instead calls the new function you created to show the train location on the LED ring. (Hint you should modify gFlashCount to keep track of your trains current position or create a new variable to do so)

   Download and run this new code. Note the change of behavior of the board. (at boot-up you should now have 4 LEDs on at the current train location and all the LEDs should blink after you press a button)

6. Continue to edit the flashLED.c
   a. Modify ToggleLED to call the train location function you created earlier and increment a position counter that controls the current train location such that the train moves around the track. (note you will have to deal with numbers below 4 and above 15)

   Download and run this new code. Note the change of behavior of the board. (Your train should be stopped at boot-up and start to move from its starting location when you press a button. You should also be able to change the speed of the train by changing the potentiometer)

7. In the project editor, choose main.c as the file to edit
   a. declare a new global variable to this file to control the trains direction
   b. Inside the while loop {} add code that detects individual button presses (hint, you can use if(gSwitchFlag & SWITCHPRESS_1) to detect button presses) (note you must set gSwitchFlag=0 inside the if to clear the event also _1 is SW1, _2 is SW2,and _3 is SW3)
   c. Once you are able to detect button presses, SW1 will set the new global variable to a value that indicates that the train will stop
   d. SW2 will set the new global variable to a value that indicates that the train will move forward
   e. SW3 will set the new global variable to a value that indicates that the train will move forward
   f. Add code in the while loop that will detect what value the global variable is and change the third line of the LCD based on the global variables value. For example if the value indicates SW1 was pressed then the third line of the LCD should display “Mode:Stop”, if the value indicates SW2 was pressed then the third line of the LCD should display “Mode: Forward”, and if the value indicates SW3 was pressed then the third line of the LCD should display “Mode: Reverse”

   Download and run this new code. Note the change of behavior of the board. (At boot-up your train should be stopped, after you press one of the SW buttons your train should start to move but the LCD should have not changed, pressing the SW buttons again after this point will change the LCD message according to the button that was pressed.

8. In the project editor, choose flashLED.c as the file to edit
   a. At the top of the file use the extern keyword to gain access to the global variable you created in the main.c to control the direction of the train (hint if you defined your global variable as int dir=0; in main.c then in flashLED.c you would prototype it by extern int dir;

b. Modify ToggleLED to increment the train location counter if SW2 is pressed and to decrement the train location counter if SW3 is pressed. If SW1 is pressed then the train location counter should not be changed.

Download and run this new code. At this point your digital model train should be operational and you should be able to stop, move the train forward and backwards along with adjust the speed of the train after you press the button 1 time after boot-up.

9. Complete your lab report.
10. 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	LCD displaying your name and "Train Demo" on correct lines	
2	LED are displaying a 4 car train that is stopped at boot-up	
3	LCD displaying the trains current direction is stopped at boot-up	
4	Pressing any switch 1 time after boot-up does not start the train	
5	Pressing SW1 makes the train stop and the LCD display that the train is stopped	
6	Pressing SW2 makes the train move forward and the LCD displays that the train is moving forward (LED move correctly)	
7	Pressing SW3 makes the train move backwards and the LCD displays that the train is moving backwards (LED move correctly)	

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 . . .

Here include the memory report given at the end of the compile process.

In this lab we learned . . . .