**Demonstration:** (NOTE: these change based on the lab - enter the demo tasks specified in each lab)

LCD shows names, brightness, temperature and internal voltage values \_\_\_\_\_\_\_\_

Pressing Switch 1 increases the brightness of the LEDs \_\_\_\_\_\_\_\_

Pressing Switch 2 decreases the brightness of the LEDs \_\_\_\_\_\_\_\_

Turning Analog In potentiometer controls the brightness \_\_\_\_\_\_\_\_

Brightness values are updated as the brightness of the LEDs changes \_\_\_\_\_\_\_\_

Pressing switch SW3 transmits the text to the host PC \_\_\_\_\_\_\_\_

The brightness value, temperature value, internal reference voltage value, and

the CRC-32 values are shown on the host PC console. \_\_\_\_\_\_\_\_

**Requirements:**

The code generated is written in C for the YRDKRX63N Evaluation Board. \_\_\_\_\_\_\_\_

The brightness of LED4 to LED15 is controlled using a PWM signal generated

using a delay function \_\_\_\_\_\_\_\_

LCD should display your group member(s) first name on line 1 and line 2 \_\_\_\_\_\_\_\_

LCD should display the brightness value on line 3 as B Value= XXX, where XXX

is the duty ratio. [Example: ‘001’ corresponds to a value of ‘1’] \_\_\_\_\_\_\_\_

LCD should display the temperature value on line 4 as T Value=XX.X, where

XX.X is the temperature in degree Celsius. \_\_\_\_\_\_\_\_

LCD should display the internal reference voltage on line 5 as V Value=X.XX,

where X.XX is the internal reference voltage. \_\_\_\_\_\_\_\_

Maximum value of brightness is 100 and the minimum is 0. Zero brightness value

means the LEDs are off\_ \_\_\_\_\_\_\_

LCD information should be centered\_ \_\_\_\_\_\_\_

Switch 1 ISR is used to increase the brightness in steps of 10\_ \_\_\_\_\_\_\_

Switch 2 ISR is used to decrease the brightness in steps of 10 \_\_\_\_\_\_\_

“Analog In” potentiometer is used to control the brightness in steps of 1 [Range: 0

– 9] \_\_\_\_\_\_\_\_

No ADC interrupts or Timers are used \_\_\_\_\_\_\_\_

UART (SCI2) is used for serial communication. \_\_\_\_\_\_\_\_

Switch 3 ISR is used to transmit the text to the host PC \_\_\_\_\_\_\_\_

The brightness value, temperature value, internal reference voltage value, and

the CRC-32 value are shown as

B Value=XXX;T Value=XX.X;V Value=X.XX

CRC-32:XXXXXXXX \_\_\_\_\_\_\_\_

The code is well documented and easy to follow. \_\_\_\_\_\_\_\_

**Learning Objectives:**

This experiment shows the conversion of analog inputs to digital and then displays it on the LCD. Also the ability to read internal reference voltages and temperature on the LCD is demonstrated. All of this information is incorporated into a UART communication protocol and is displayed on the host console after performing a CRC-32 check.

**General Steps:**

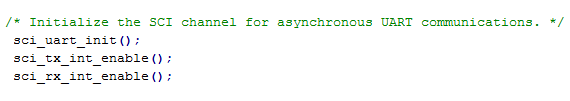
1. Go to the HEW IDE
2. Power up the Renesas RX63N by connecting it to the computer via mini-USB
3. Build upon previous lab and add UART/CRC-32 functionality and other requirements.
4. Demonstrate the working project.

**Detailed Steps:**

1. Building upon the work from last lab session (Lab 3). The sample UART demonstration program was used to communicate with a host terminal.
2. After verification of the communication protocol the UART.C and UART.H files were added to the source folder and included in the Main.C file of the lab. The baud rate is changed to 9600bps in the UART.C file to match the communication standards of the HyperTerminal.
3. In order to pass the information from the temperature and voltage functions built in the previous lab two float variables were created to pass the information to the SW3 ISR.



1. In order for the communication to take place the routines for UART must be called in from UART.C. The functions were called in the main loop before the while loop.



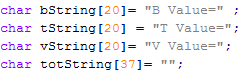
1. In the SW3 ISR three character arrays are created to hold the values of Brightness Level, Temperature, and Voltage.



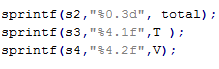
1. The library string.h is called to give the ability to perform string operations necessary for output to the host pc.



1. Next, four character arrays were created to hold the string text and values from the s2-s4 variables.



1. Each value is passed into its proper character array from other functions and formatted accordingly.



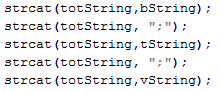
1. After format and values are correct for each character array the strings are concatenated together to form one string.







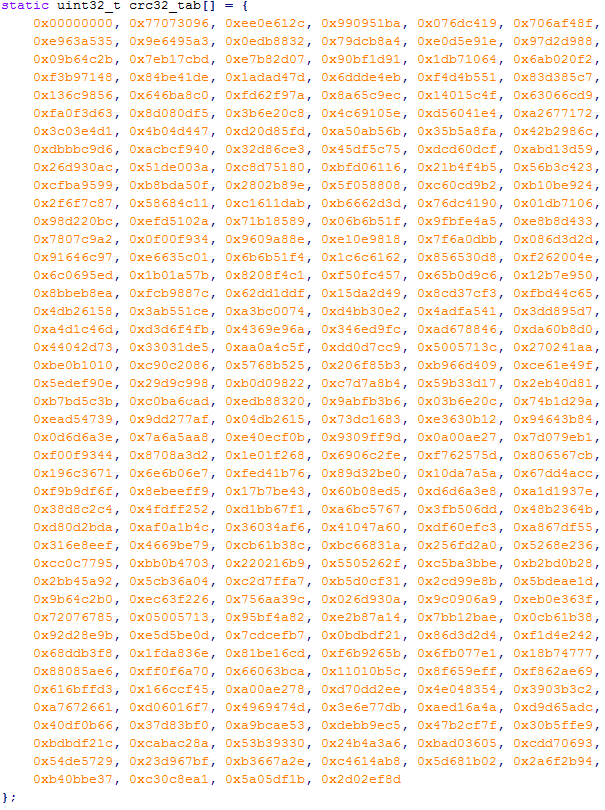
1. Once the strings contain both values and the desired text they are concatenated together to form one large string separated by semicolons. This value will have the CRC calculated in a later stage.



1. Lastly, as per the lab instructions the total string is output to the HyperTerminal in the proper format.



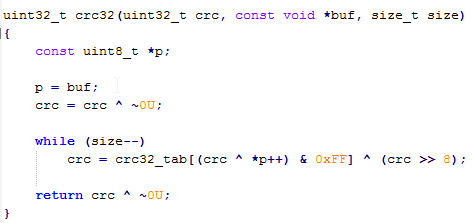
1. After completion of the UART communication portion (part B) the CRC-32 value can be calculated. To begin the CRC-32 file is downloaded from the specified website and incorporated into the Main.C file as follows.
2. Defined outside of the main function is a table holding each CRC result for all possible ASCII characters. This information will be referenced while performing CRC on the totString created in part B.



1. A function was created CRC-32 that used the CRC value from the table above, buffer size defined later in the SW3 ISR, and the buffer size also defined later in the code.



1. Inside the function a pointer is created to hold the value of the buffer passed into the function.
2. The CRC hex value of 0x00000000 is passed into the function to begin with. The value is then XOR’d with the first character of the totString array and masked with a value between 0 and 256. This is then XOR’d with the previous CRC value by shifting CRC 8 bits to the right.
3. The value is returned with the XOR’d value of CRC and 0x11111111.



1. As before a new character array was created to hold the value of CRC-32 result.
2. The sprint function from the string library was used to copy the calculated CRC from the function and format the output into the correct size.



1. The sci\_put function from the UART.C file to write the CRC-32 value to the host.



1. This information is then compared in a CRC calculator to confirm accuracy.

**Observations:**

During the fourth lab experiment the main purpose was to perform communication to a host terminal (pc) and output values captured by the microcontroller to the host. Building off what was already created in earlier labs the CRC value of the string sent to the host was calculated and then displayed as well on the host system. This CRC-32 value confirms the accuracy of the results being transferred. The CRC-32 value is a cyclic redundancy check which is used to check for errors in many different systems. This does so by performing an XOR of each character in the string to the known CRC value for each character which is stored in a table. Without this CRC the possibility of incorrect information transfer could be a large possibility and the end user would not even be aware of it.

**Summary:**

Overall, the lab experiment tested our abilities to collect, collate, and communicate data effectively through a UART connection to a host. The CRC calculation was performed in order to confirm the proper communication was established. The results of this lab will help in future applications when serial communication between two devices is necessary. As mentioned before the lack of CRC could result in catastrophic results if critical data is changed during communication.