ECGR 4101/5101, Fall 2011: Lab 4

Reading and Writing from the UART

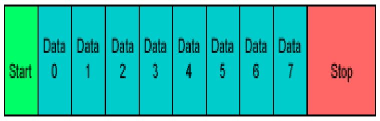
Learning Objectives:

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). This lab will introduce initializing the UART, transmitting, and receiving characters from a computer. Additionally, you will also use the oscilloscope to catch the transmission signal as it passes from the board across a serial cable to a computer.

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 two dimensional space.

In this lab we will be writing routines to initialize the hardware by setting the baud rates, parity, data bits, and stop bits used in UART communications. We will also be using fewer Renesas API commands in this exercise, and instead actually set the registers directly. This will involve much datasheet reading, so I suggest looking at the pdf documents located within your Rensas installation. By default they can be found in "C:\Program Files (x86)\Renesas\Hew\Manuals\Renesas\PDFs."

So let's begin!

Laboratory Assignment:

1. Create a new HEW workspace by creating a new tutorial project.
2. Your main function should begin with the LCD initialization functions, which look like this:

```
/* Initialise the debug LCD on the RSPI bus */
YRDKRX62N_RSPI_Init(RSPI_CHANNEL_0);
InitialiseLCD();
```

Again, if you created the tutorial project, this should already be there.

3. Next, clean out the contents of the while(1) loop, as this is irrelevant to this project. The while(1) loop should be blank when you're done.
4. Next, create a title for this project. Something like the following:

   DisplayLCD(LCD_LINE1, "USART Lab");
5. Now that the basics are finished, it is now time to create an initialization routine for the UART. The RX62N's UART is controlled through the Serial Communication Interface registers, which you can read about in your text book starting on page 236.

   UART can be configured in many ways, but we will be using 8 bit data sizes, no parity, and 1 stop bit, which are pretty common settings. Within your initialization routine, you will need to look at which registers are used to apply these settings, which registers enable the UART hardware, and of course set the input and output data direction registers. It is also important to note that there are several UART modules available on the RX62N. Since we will be interfacing to a computer later on with an RS-232 cable (serial cable), so be sure to initialize the SCI module that is connected to the RS-232 port.

   For our initialization routine, we will want to add a little bit of flexibility. We will want to pass a desired baud rate, and have the UART configured to operate on it. In this lab we will be using baud rates of 9600, 4800, and 2400. To achieve these baud rates you will need to select the proper peripheral clock speed and baud rate register value. An equation for calculating this can be found in the hardware manual, which is featured below. A more detailed table that might allow you to skip the calculation can also be found within.

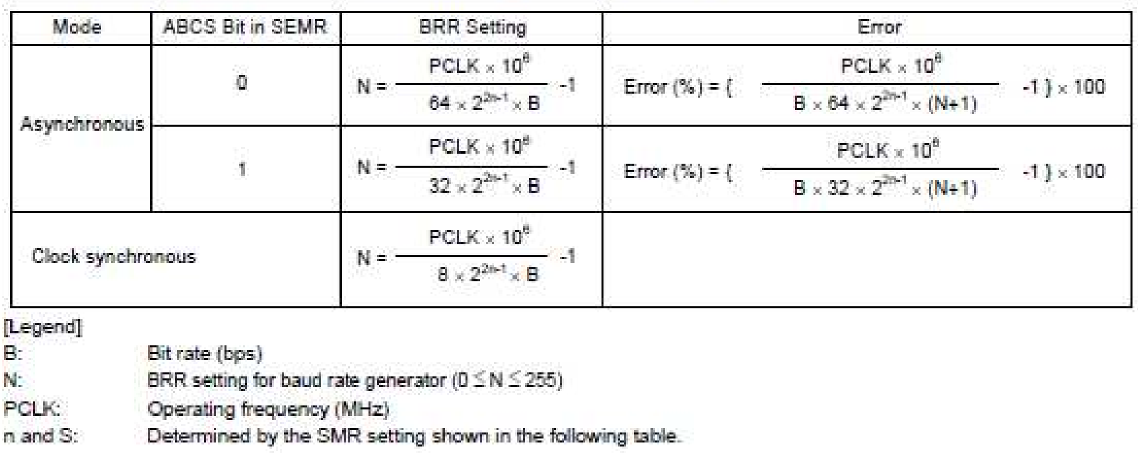


Table 28.5 Relationships between N Setting in BRR and Bit Rate B

Mode	ABCS Bit in SEMR	BRR Setting	Error
Asynchronous	0	$N = \frac{PCLK \times 10^6}{64 \times 2^{2n-1} \times B} - 1$	$\text{Error (\%)} = \{ \frac{PCLK \times 10^6}{B \times 64 \times 2^{2n-1} \times (N+1)} - 1 \} \times 100$
	1	$N = \frac{PCLK \times 10^6}{32 \times 2^{2n-1} \times B} - 1$	$\text{Error (\%)} = \{ \frac{PCLK \times 10^6}{B \times 32 \times 2^{2n-1} \times (N+1)} - 1 \} \times 100$
Clock synchronous		$N = \frac{PCLK \times 10^6}{8 \times 2^{2n-1} \times B} - 1$	

[Legend]

B:	Bit rate (bps)
N:	BRR setting for baud rate generator ($0 \leq N \leq 255$)
PCLK:	Operating frequency (MHz)
n and S:	Determined by the SMR setting shown in the following table.

A good example of an initialization routing can be found on page 237. Keep in mind that you cannot mindlessly copy this code and expect it to work for your application. You will need to look at each register used, and understand what settings are being applied by the values of each bit.

6. Next, we will need a routine for transmitting the data. The UART transmission is accomplished by loading up a data transfer register, which once filled, is automatically transmitted by the UART hardware. Once transmission is complete, a flag is set within the UART status register, which is useful for helping construct code that prevents writing to the transfer register before the data is finished sending!

   Again, a good sample transmission routine can be found within the book.

7. Now we need to be able to detect incoming data. Received data is stored in a Received Data Register. When new data is available in that register, another flag is set within the Serial Status Register. By writing an if statement within our while loop, we can constantly poll to see when that bit is set high by masking with that register.

   Write an if statement within the while loop that checks to see if the received data flag is set, and if it is, it stores the data into a variable.

8. In this lab we will be connecting the board to a compute via serial cable. From the computer we will be using a terminal program (such as hyperterm, or terraterm) to send characters to the board. The board will detect lowercase and capital letters sent to it, and if the detected character is capitalized, it will lowercase it and send it back, and vice versa.

   Within the if statement that checks for received data, write another conditional statement that checks the ascii value of the character, switches the case of the character, and then writes it to the transmit character routine that was created earlier.

9. Even though we've created an initialization routine, we still haven't used it yet! For this lab, we want to send characters at various baud rates. Create a set of conditional statements within the while(1) loop that checks for button presses. Have each button call the UART initialization routine with different baud rates. 9600, 4800, and 2400 are the baud rates we will be using, so assign each one to a button press. When a new baud is selected, write that selection to the LCD.

10. Now we have a board with easily adjustable baud rates. For the next part of the lab, we will need to use the oscilloscope to pick off the data being transmitted and obtain a screen shot. On the bottom of the RS-232 connection, the pins can be contacted by an o-scope probe. Use the scope and catch a UART transmission as it is being sent from the computer to the board at one of each of the selectable baud rates. Be sure that there are sufficient amounts of data on the screen to indicate the baud that is being used. For consistency's sake, please transmit the same character on each test, so the differences in baud rate are more easily observed.

    The oscilloscopes available in the lab have a port for a compact flash card for saving screenshots. Since compact flash cards are widely unavailable anymore, it will be acceptable to take a NICE picture with a smart phone or camera and neatly crop the image when adding it to your lab report. If you do not have a decent camera available, the TA can provide you with a compact flash card and a USB adaptor for saving your images.

Lab Report:

1	LCD displays the currently selected USART	
2	Characters sent from the PC are successfully case-inverted and transmitted back.	
3	Button presses change the UART baud rate properly	
4	Oscilloscope screen shots of each baud rate are available (partial credit for each baud rate)	

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 . . . .