Lab Report 3

Nathan Tipton, Partner: Tarrin Rasmussen

October 17, 2016

Contents

1 Objectives

The purpose of this lab is to practice using timers, UART, and the RS-232 module to transmit serial data between the microcontroller and the PC.

2 Overview

For this lab, we will be writing a Spiteful 8-ball program. Upon input from the user, either button press or key press) the 8-ball will provide an answer. The answer will be sent from the microcontroller 8-ball through serial to the user in a terminal emulator.

3 Lab Requirements

1. Carriage return and line feed are transmitted immediately after program starts
2. Duration of first start bit is 1/9600 seconds
3. 8/9600 seconds between end of start bit and beginning of stop bit
4. Answer displayed on either button press or sending of a character from the PC to the board.

4 Procedure

4.1 Connecting to PC

4.1.1 Rs-232 module

Using the RS-232 module, we will allow our microcontroller and the computer to communicate over serial.The RS-232 is connected to the microcontroller on Port B [1:0] and Port F [1:0]. The PC and RS-232 are connected via a serial modem cable.

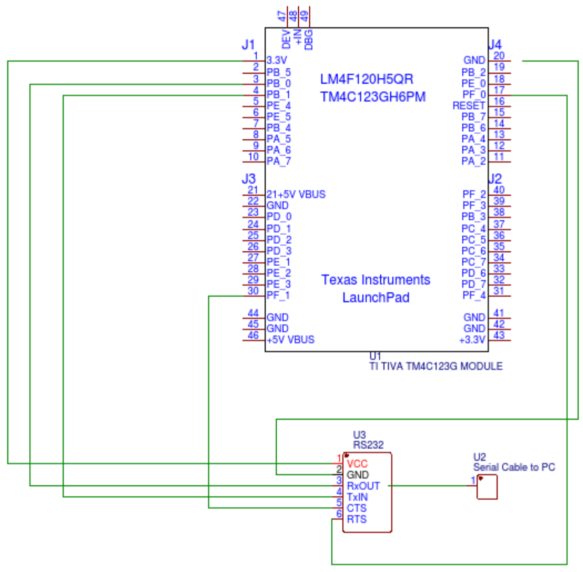


Figure 1: Schematic

4.1.2 Configuring UART

1. Enable UART Module: We are using UART Module 1. Module 1 uses pins PBO and PB1 for U1Rx and U1Tx. Pins PF0 and PF1 are U1RTS and U1CTS.

2. Enable GPIO Clock: The clock for the appropriate GPIO modules needs to be enabled. In this case, Ports B and F.

3. Unlock F port: Port F needs to be unlocked in order for us to use it.

4. Enable Alternate Function: The pins we selected for UART module 1 need the alternate function enabled. This allows the UART registers to control the pins instead of the GPIO registers.

5. Set Port control for alternate function: Using the port mux control we set the alternate function configuration for each of the pins we are using on ports B and F.

6. Configure digital enable: Digital enable needs to be set for each of the pins we are using.

7. Disable UART: Before modifying values in the UART registers the UART module must be disabled.

8. Set Baud Rate: We need to calculate both the integer and fraction values for the Baud Rate. The calculation uses fixed point arithmetic. Calculate the Baud Rate Divisor with the following equation.

$$BRD = UARTSysClk/(16 * Baudrate) \tag{1}$$

   In order to use the resulting anser we need to break it up into Integer and Fractional.

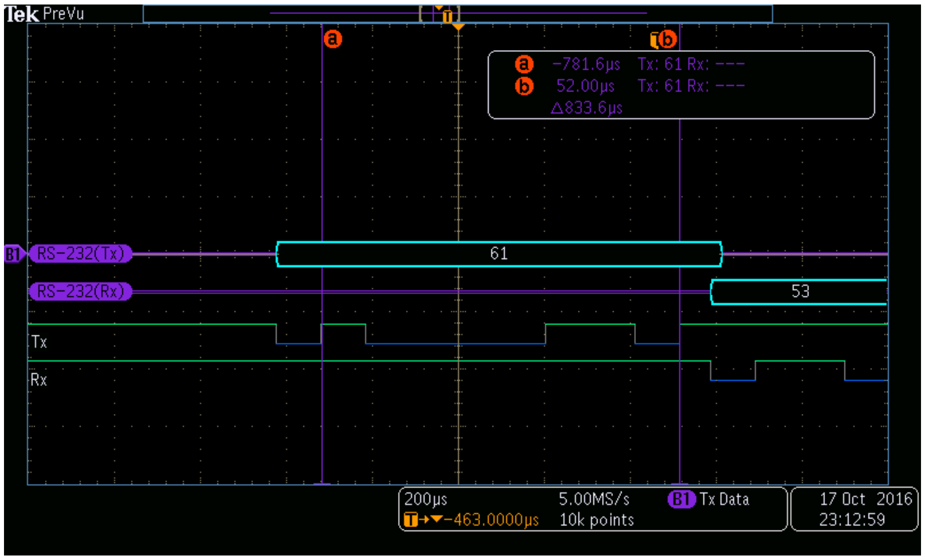
$$I = int(104.01640625) = 104. \tag{2}$$

$$F = int(0.01640625 * 2^n + 0.5) = 11, n = 6. \tag{3}$$

9. Set Line Control: The UART is configured for 9600 baud with 8 data bits and one stop bit.
10. Set Baud Clock Source: The 16 Mhz PIOSC is selected as the source.
11. Enable UART: UART is renabled after configuring the UART registers.

4.1.3 Testing

Using the Logic Analyzer, we verified our UART configuration. Figures 2-8 show screen captures of the measurements and the program running. The screen shots show that each frame is made of a start bit, 8 bits and an end bit.



```
a  -781.6µs  Tx: 61 Rx: ---
b  52.00µs   Tx: 61 Rx: ---
   Δ833.6µs
```

Figure 2: 8/9600 from start bit to end bit.

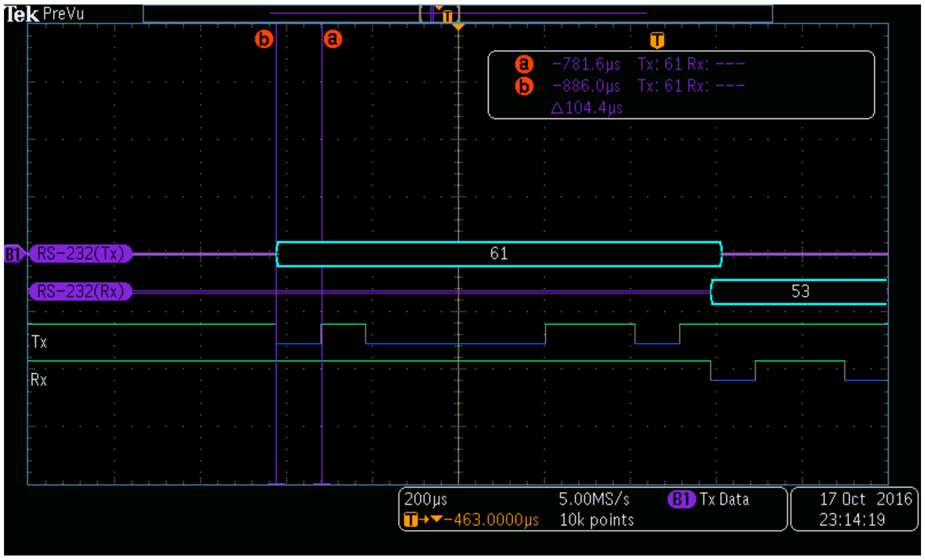
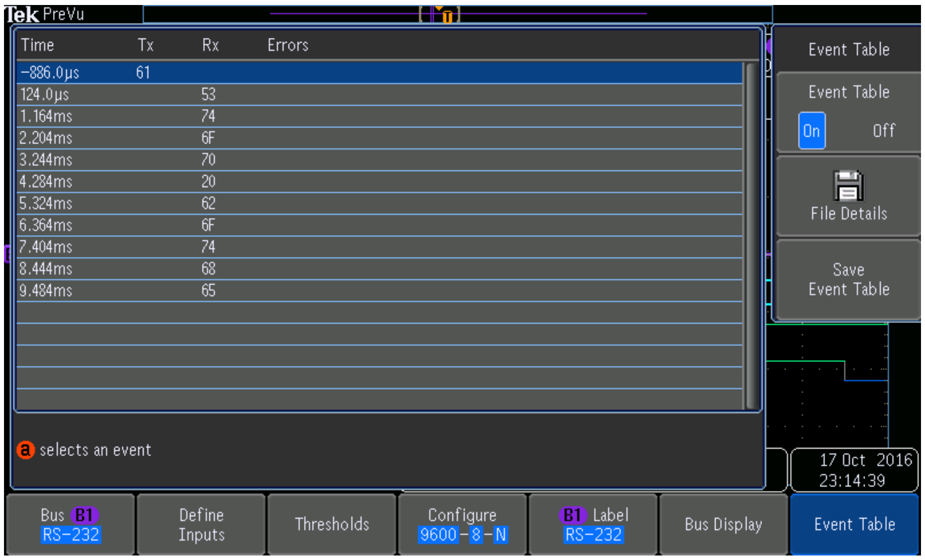


Figure 3: 1/9600: Duration of start bit.



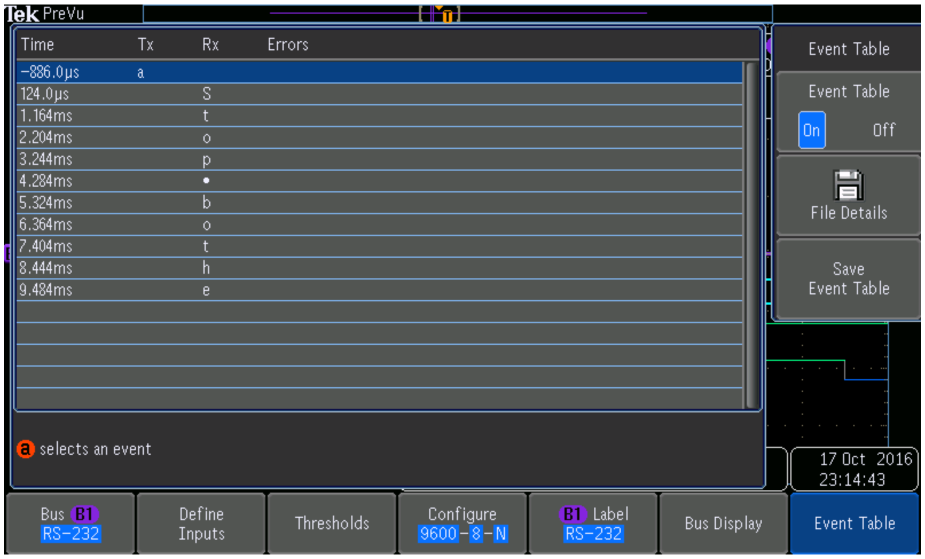
Time	Tx	Rx	Errors
−886.0µs	61		
124.0µs		53	
1.164ms		74	
2.204ms		6F	
3.244ms		70	
4.284ms		20	
5.324ms		62	
6.364ms		6F	
7.404ms		74	
8.444ms		68	
9.484ms		65	

a selects an event

Event Table: [✓ On] Off

Bus B1 RS−232 · Define Inputs · Thresholds · Configure 9600−8−N · B1 Label RS−232 · Bus Display · Event Table

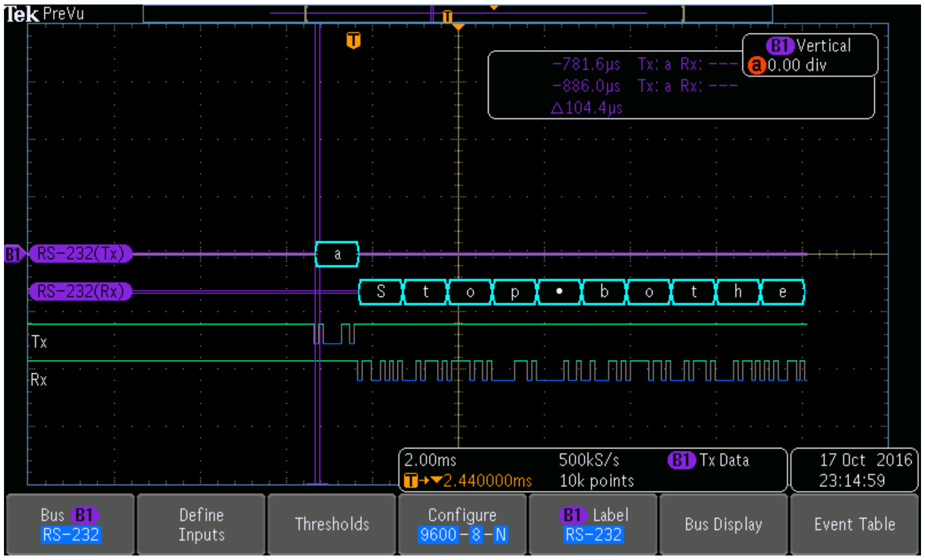
Figure 4: Event Table with Hex



Time	Tx	Rx	Errors
−886.0µs	a		
124.0µs		S	
1.164ms		t	
2.204ms		o	
3.244ms		p	
4.284ms		•	
5.324ms		b	
6.364ms		o	
7.404ms		t	
8.444ms		h	
9.484ms		e	

a selects an event

Figure 5: Event Table with ASCII



```
−781.6µs  Tx: a Rx: ---
−886.0µs  Tx: a Rx: ---
△104.4µs
```

```
2.00ms           500kS/s     B1 Tx Data
T→▼2.440000ms    10k points
```

Figure 6: Sent character with Ascii message

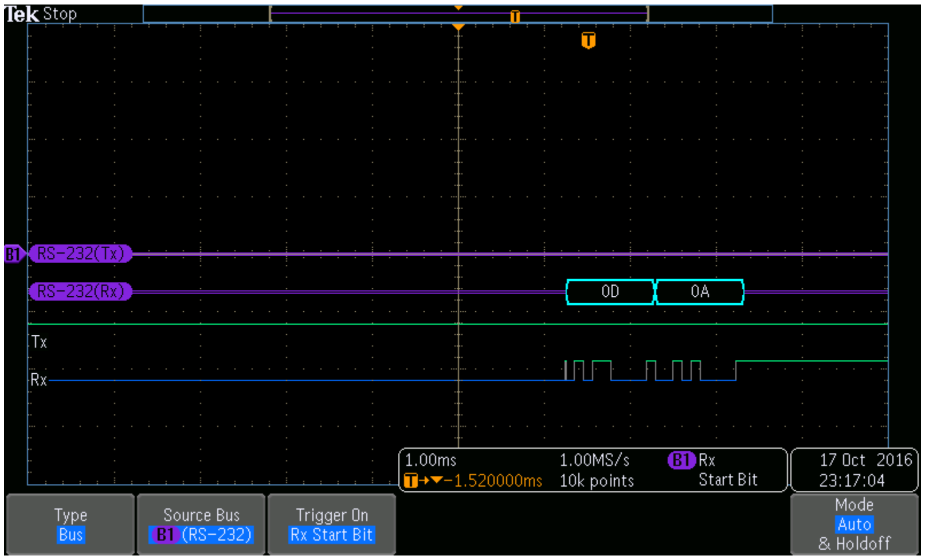
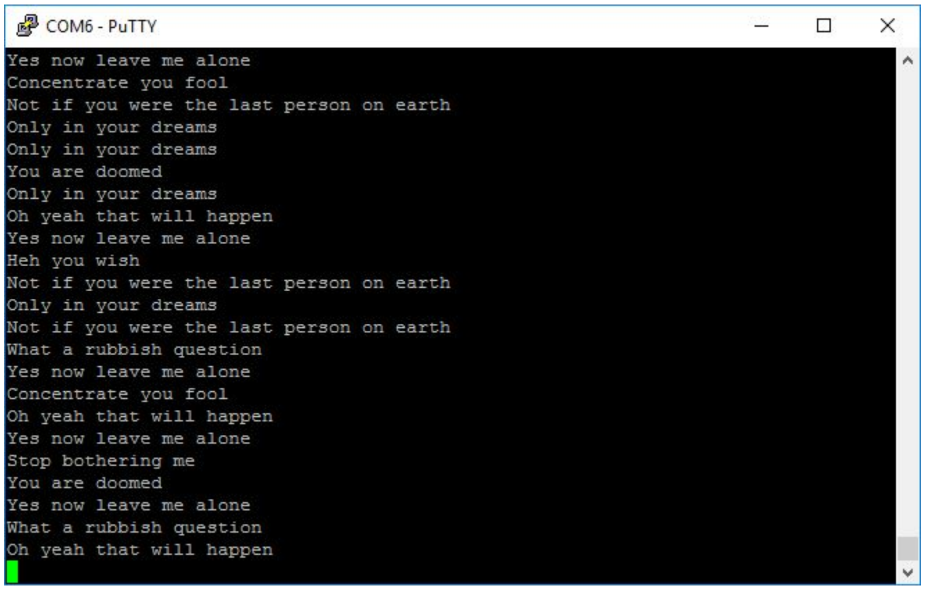


Figure 7: Program start



```
Yes now leave me alone
Concentrate you fool
Not if you were the last person on earth
Only in your dreams
Only in your dreams
You are doomed
Only in your dreams
Oh yeah that will happen
Yes now leave me alone
Heh you wish
Not if you were the last person on earth
Only in your dreams
Not if you were the last person on earth
What a rubbish question
Yes now leave me alone
Concentrate you fool
Oh yeah that will happen
Yes now leave me alone
Stop bothering me
You are doomed
Yes now leave me alone
What a rubbish question
Oh yeah that will happen
```

Figure 8: Terminal

5 Conclusion

Initially, we struggled with this lab. We did not fully understand what UART did and what the FIFO was. However after extensive searching online we found resources that clarified the concepts for us. When a new byte is written to the UART data register it is placed into the TX fifo. The UART takes the data from the FIFO byte by byte into the transmit shift register. One bit at a time the shift register transmits the information at the rate set by the baud rate. The Tx and Rx flags are used to check if the FIFOs are empy or full. Checking these flags determines when new data should be inserted or taken out of the FIFOs. The FIFOs can hold 16 elements at a time. Once we understood this the coding went smooth.

6 Appendix

6.0.1 Code

```
        THUMB
        AREA DATA, ALIGN=2
        ALIGN
        AREA |.text|, CODE, READONLY, ALIGN=2
        EXPORT Start

M  = 0x0D, 0x0A
M0 = "Nope", 0x0D, 0x0A
M1 = "You are doomed",0x0D, 0x0A
M2 = "Concentrate you fool", 0x0D, 0x0A
M3 = "What a rubbish question", 0x0D, 0x0A
M4 = "Only in your dreams", 0x0D, 0x0A
M5 = "Yes now leave me alone", 0x0D, 0x0A
M6 = "Heh you wish", 0x0D, 0x0A
M7 = "Oh yeah that will happen", 0x0D, 0x0A
M8 = "Stop bothering me", 0x0D, 0x0A
M9 = "Not if you were the last person on earth", 0x0D, 0x0A

                ALIGN

RCCRESET ∗ 0x400FE060
RCGCUART ∗ 0x400FE618
RCGCGPIO ∗ 0x400FE608
GPIOBALT ∗ 0x40005420
GPIOFALT ∗ 0x40025420
GPIOBCTL ∗ 0x4000552C
GPIOFCTL ∗ 0x4002552C
GPIOBDIR ∗ 0x40005400
GPIOFDIR ∗ 0x40025400
UARTCTL ∗ 0x4000D030
UARTIBRD ∗ 0x4000D024
UARTFBRD ∗ 0x4000D028
UARTLCRH ∗ 0x4000D02C
UARTCC ∗ 0x4000DFC8
UART1DAT ∗ 0x4000D000
GPIOFLOC ∗ 0x40025520
GPIOFCOM ∗ 0x40025524
GPIOBDIG ∗ 0x4000551C
GPIOFDIG ∗ 0x4002551C
UART1FR ∗ 0x4000D018
SYSTICKS ∗ 0xE000E014
```

```
SYSTICKE * 0xE000E010
SYSTICKV * 0xE000E018
GPIOSW1D * 0x40025040

Start

                ;Reset Clock
        LDR R0, =RCCRESET ;RCC Reset
                MOV32 R1, #0x078E3AD1
                STR R1, [R0]

                ;Enable UART Module
                LDR R0, =RCGCUART
                MOV R1, #2 ;Enable Module 1
                STR R1, [R0]

                ;Enable GPIO Clock
                LDR R0, =RCGCGPIO
                MOV R1, #0x22 ;Enable Port B, F
                STR R1, [R0]

                ;Unlock Port
                LDR R0, =GPIOFLOC
                MOV32 R1, #0x4C4F434B ;Unlock Code
            STR R1, [R0]

            LDR R0, =GPIOFCOM
            MOV R1, #0xFF ;Unlock all pins
            STR R1, [R0]

                ;Enable Alternate Function
                LDR R0, =GPIOBALT
                MOV R1, #3 ;Port B pins 0, 1
                STR R1, [R0]

                LDR R0, =GPIOFALT
                MOV R1, #3 ;Port F pins 0, 1
                STR R1, [R0]

                ;Set Alternate Function
                LDR R0, =GPIOBCTL
                MOV R1, #0x11 ;Port B pins 0, 1 to Alternate 1
                STR R1, [R0]

                LDR R0, =GPIOFCTL
                MOV R1, #0x11 ;Port F pins 0, 1 to Alternate 1
                STR R1, [R0]

                ;Digital Enable
                LDR R0, =GPIOBDIG
                MOV R1, #3
                STR R1, [R0]

                LDR R0, =GPIOFDIG
                MOV R1, #3
                STR R1, [R0]

                ;Disable UART
                LDR R0, =UARTCTL
                LDR R1, [R0]
                AND R1, #0xFFFFFFFE
                STR R1, [R0]

                ;Load Int Portion of BRD
                LDR R0, =UARTIBRD
                MOV R1, #104
                STR R1, [R0]
                ;Load Frac Portion of BRD
                LDR R0, =UARTFBRD
                MOV R1, #11
                STR R1, [R0]

                ;Set Line Control
                LDR R0, =UARTLCRH
                MOV R1, #0x70
                STR R1, [R0]

                ;Set Baud Clock Source
                LDR R0, =UARTCC
                MOV R1, #0x5 ;PIOSC
                STR R1, [R0]

                ;Enable UART
                LDR R0, =UARTCTL
                LDR R1, [R0]
                ORR R1, #1
                STR R1, [R0]

                ;
        ; Setup Port SW1
        ;

        MOV32 R0, #0x40025000 ;Port F Base

        ;Set Direction
        LDR R1, [R0, #0x400]
        AND R1, #0xEF ;#0b11101111
        STR R1, [R0, #0x400]

        ;Disable Alternate
        LDR R1, [R0, #0x420]
        AND R1, #0xEF ;#0b11101111
        STR R1, [R0, #0x420]

        ;Set Drive Strength
        LDR R1, [R0, #0x508]
        ORR R1, #0x10 ;#0b00010000
        STR R1, [R0, #0x508]

        ;Pin Function
        LDR R1, [R0, #0x510]
        ORR R1, #0x10 ;#0b00010000
        STR R1, [R0, #0x510]

        ;Enable Pin
        LDR R1, [R0, #0x51C]
        ORR R1, #0x10 ;#0b00010000
        STR R1, [R0, #0x51C]

                ;Enable Systick
                LDR R0, =SYSTICKS
                MOV R1, #9
                STR R1, [R0]
                LDR R0, =SYSTICKE
                MOV R1, #1
                STR R1, [R0]

                LDR R7, =M

                ;Init Ouput
init LDR R0, =UART1DAT
                LDRB R1, [R7]
                STR R1, [R0]
                CMP R1, #0x0A
                BEQ RX
                ADD R7, #1
                B init

                ;Recieve until Buffer Empty
RX  LDR R0, =GPIOSW1D
                LDR R1, [R0]
                CMP R1, #0
                BEQ debounce

                LDR R0, =UART1FR
                LDR R1, [R0]
                AND R1, #0x10
                CMP R1, #0x10
                BEQ RX

                LDR R0, =UART1DAT
                LDR R1, [R0]
                CMP R1, #0
                BNE TX

debounce
                LDR R0, =GPIOSW1D
                LDR R1, [R0]
                CMP R1, #0
                BNE TX
                B debounce

TX  LDR R0, =SYSTICKV
                LDR R1, [R0]

                ;Choose Random Message
                CMP R1, #0
                LDREQ R7, =M0
                CMP R1, #1
                LDREQ R7, =M1
                CMP R1, #2
                LDREQ R7, =M2
                CMP R1, #3
                LDREQ R7, =M3
                CMP R1, #4
                LDREQ R7, =M4
                CMP R1, #5
                LDREQ R7, =M5
                CMP R1, #6
                LDREQ R7, =M6
                CMP R1, #7
                LDREQ R7, =M7
                CMP R1, #8
                LDREQ R7, =M8
                CMP R1, #9
                LDREQ R7, =M9

                ;Wait while TX Buffer Full
full LDR R0, =UART1FR
                LDR R1, [R0]
                AND R1, #0x20
                CMP R1, #0x20
                ;ANDS R1, #0x20
                BEQ full


                ;Write to Data Register
                LDR R0, =UART1DAT
                LDRB R1, [R7]
                STR R1, [R0]
                CMP R1, #0x0A
                BEQ RX
                ADD R7, #1
```

```
                    B full

sent B sent

          ALIGN
          END
```