Objectives

- To execute the instructions of Lab 3 (ECE 3710 Lab 3.pdf) from the course wiki*.
- To gain experience with writing and testing a program that uses timers, UART, and the RS-232 module to transmit serial data between the microcontroller and the PC.

Overview

"For this lab we were going to create a Magic 8-Ball® program but Mattel threatened to sue, so we'll implement a Spiteful 8-Ball program, instead. Upon being queried by the user (either a button press or a character sent on the serial port) your microcontroller-based 8-ball program will transmit an answer via the serial port. (ECE 3710 Lab 2.pdf)

Preparation

- Come to lab with the microcontroller board and breadboard.
- Be prepared to use most of the instructions that have been learned in class.
- Read V1.Ch8.2 and V2.Ch7.2 of course textbooks.
- Obtain the following items from the department lab store:
  - RS-232 Module
  - Serial Cable
  - Ribbon Cable
  - Leads for the Voltage Source

* https://spaces.usu.edu/display/ece3710/Home

Requirements

1. Upon Reset the micro-controller should be waiting for input either the computer or a button.
2. The serial port needs to be configured for 9600 baud with 8 data bits and one stop bit. Immediately after reset, a carriage returns (ASCII 0x0D) and a line feed (ASCII 0x0A) need to be sent via the serial port.
3. After a button is pressed or any character is received on the serial port, exactly one of the following messages should appear on the console window:
   - Nope
   - You are doomed
   - Concentrate you fool
   - What a rubbish question
   - Only in your dreams
   - Yes now leave me alone
   - Heh you wish
   - Oh yeah that will happen
   - Stop bothering me
   - Not if you were the last person on earth

Each of the above messages needs to be terminated by a carriage return and a line feed and selected for transmission at random.*

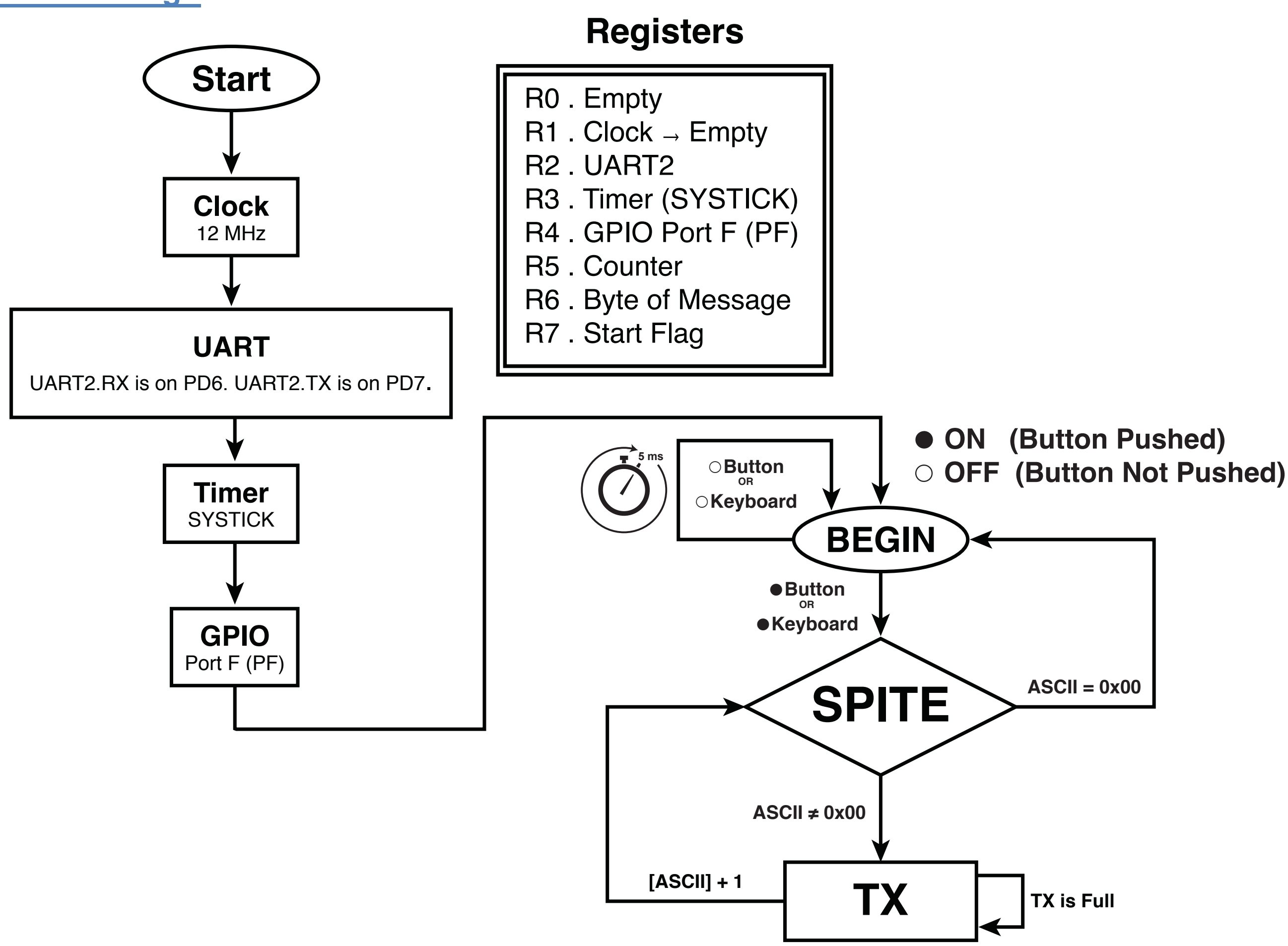
Connecting to PC

Attach the 9 pin D-Sub to the serial port on the back of the Lab PC, and the other end to the RS232 module.

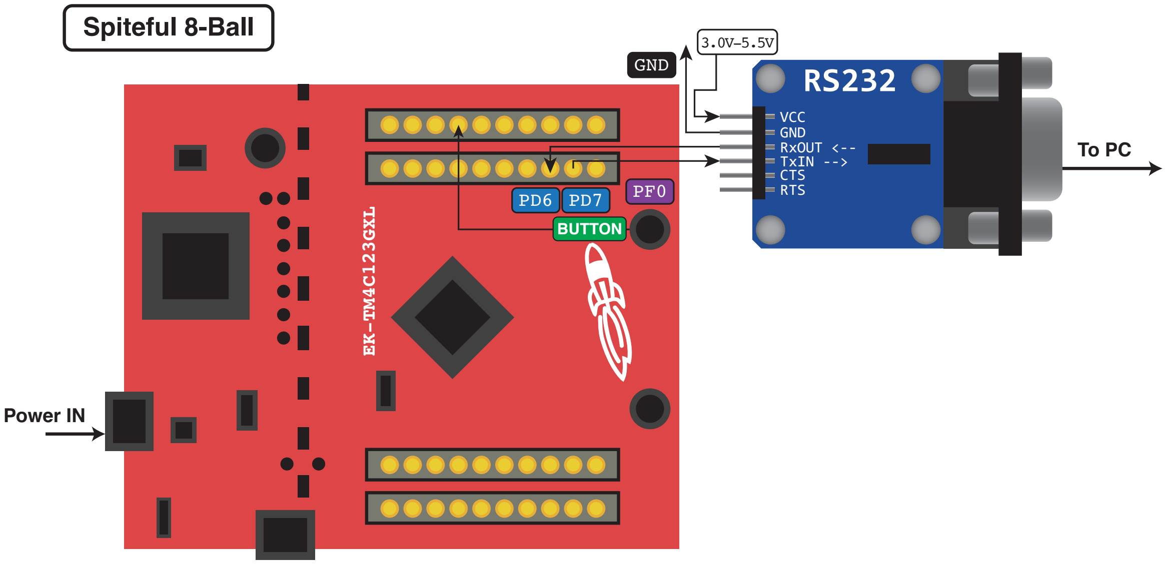
To indicate that you want a question answered, and to receive a response from the program, you'll need to setup a terminal emulator to communicate with the microcontroller over a serial interface. To do this, connect the RS-232 module on your evaluation board and to the serial port on your computer. Start a terminal emulator, such as HyperTerminal or Putty, on your computer and configure it for 9600 baud, 8 data bits, one stop bit, and no parity bit.

Documentation

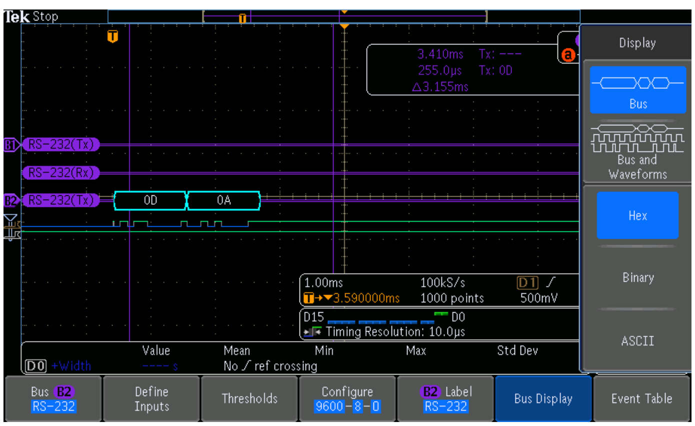
Software Design

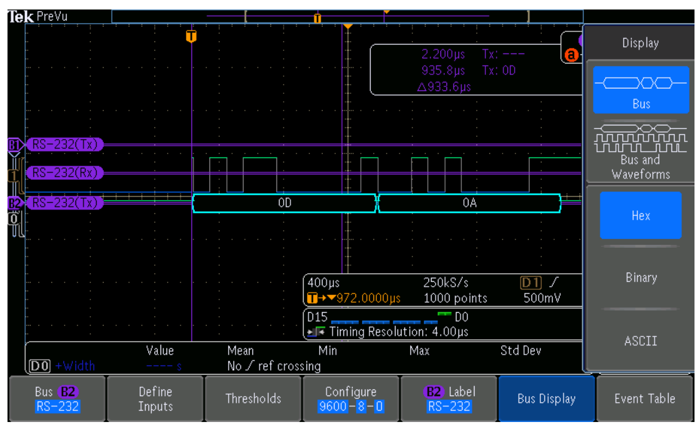


Hardware Schematic

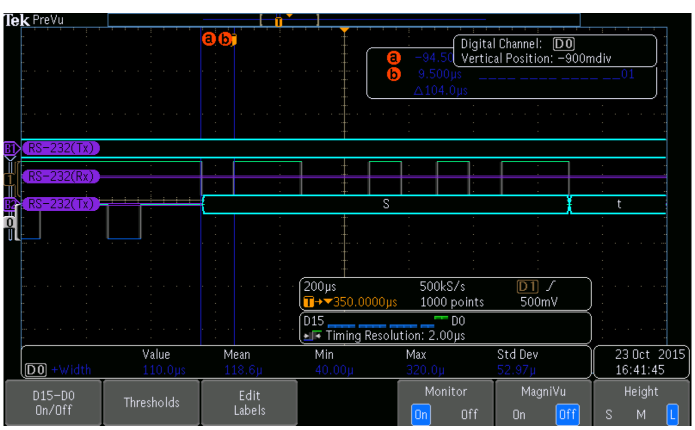


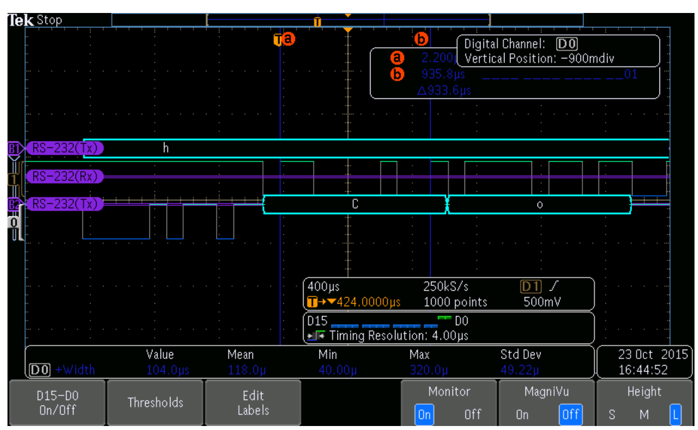
Logic Analyzer Analysis



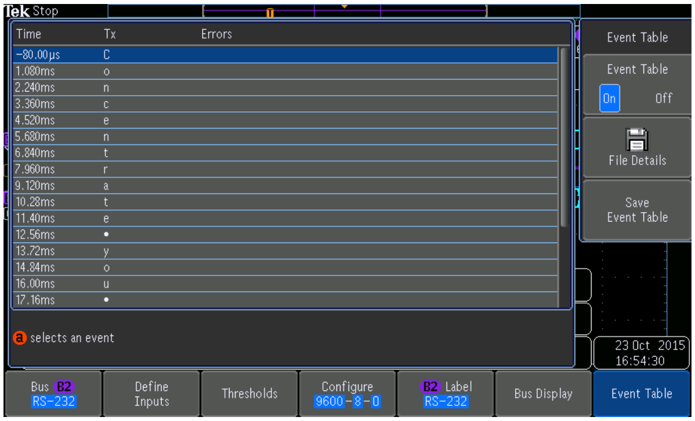


- The carriage return and line feed are transmitted immediately after the program starts.

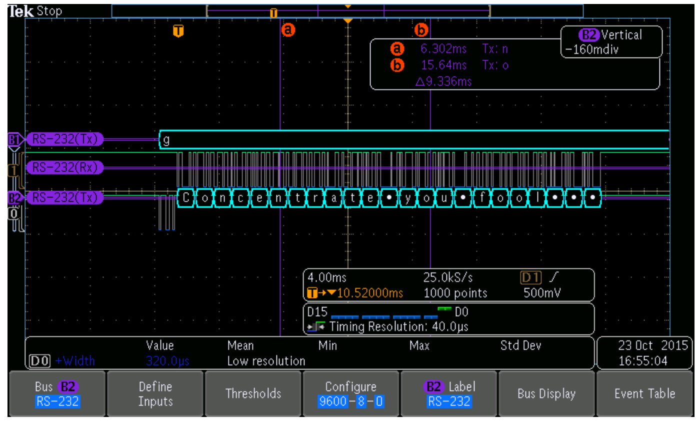




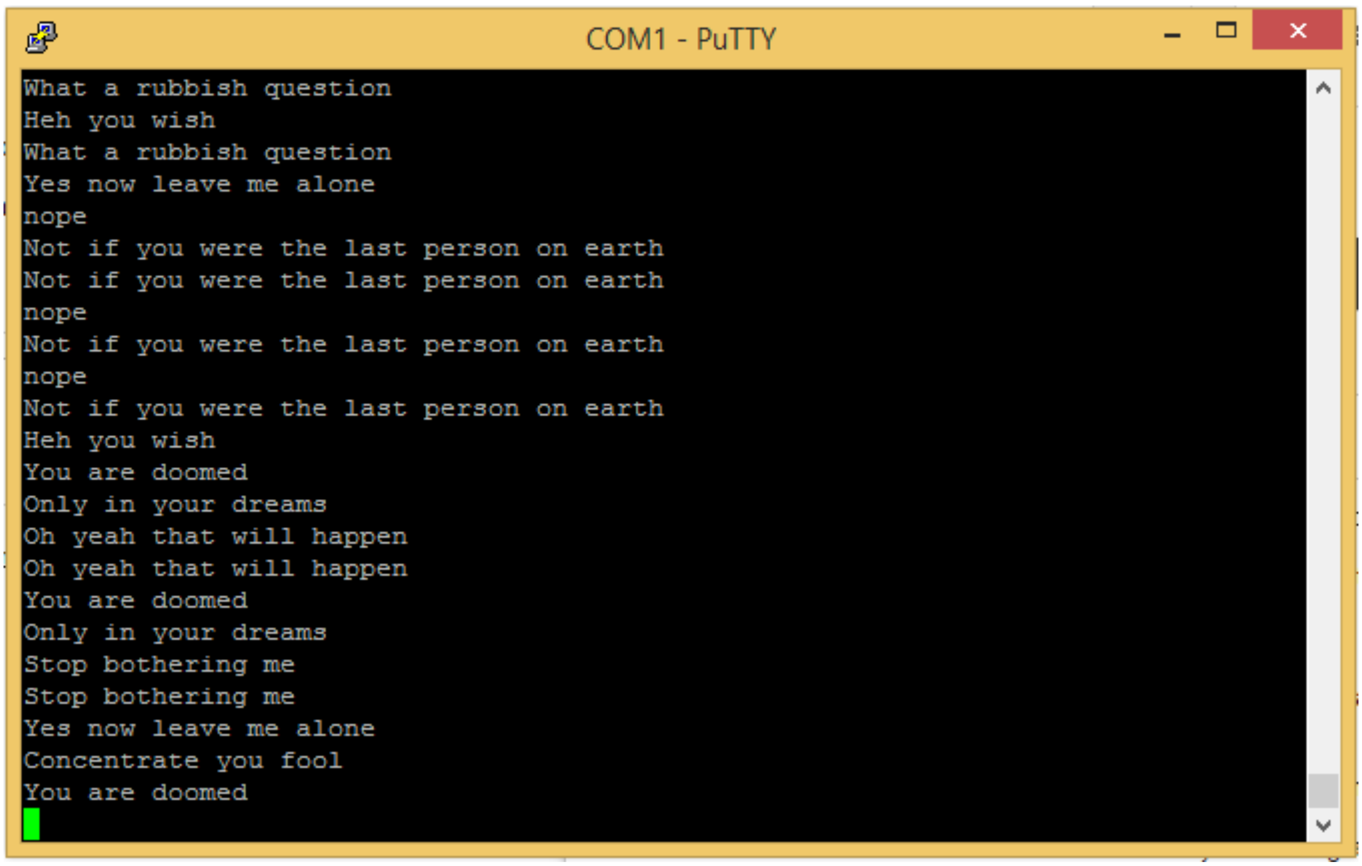
- The duration of the first start bit is 1/9600 seconds (104 µs).
- Shows that 8/9600 seconds (833 µs) elapse between the end of the start bit and the beginning of the stop bit.



Event Table



Bus



Terminal Emulator

Program Code

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

Start

;; Clock Configuration
  LDR R1,=0x400FE000                    ; System Control Base Address : SYSCTL (pg. 235 - MDS)
  ; 1. Enable Clocks
  MOV R0,#0x4                           ; Enable Clock on UART 2 (UART2:2=1) (0x4 = 0b100)
  STR R0,[R1,#0x104]                    ; Clock Base Address, GPTM : RCGC1 (pg. 458 - MDS)
  MOV R0,#0x28                          ; Enable Clock on PF, PD (GPIOF:5=1 GPIOD:3=1) (0x28=0b0010.1000)
  STR R0,[R1,#0x108]                    ; Clock Base Address, GPIO : RCGC2 (pg. 462 - MDS)

;; UART Configuration
; >> UART 2 (UART2)
  LDR R2,=0x4000E000                    ; UART 2 Base Address (pg. 900 - MDS)
  ; 1. Enable Alt. Function ; UART2.RX is on PD6. UART2.TX is on PD7.
  LDR R1,=0x40007000                    ; Port D (PD) Base Address (pg. 656 - MDS)
  MOV R0,#0xC0                          ; Enable Pins and Alt. Function on Port. (0xC0=0b11000000)
  STR R0,[R1,#0x420]                    ; GPIO Alternate Function Select : GPIOAFSEL (pg. 668 - MDS)
  STR R0,[R1,#0x51C]                    ; GPIO Digital Enable : GPIODEN (pg. 679 - MDS)
  ; 2. Disable UART
  MOV R0,#0                             ; The UART is disabled.
  STR R0,[R2,#0x30]                     ; UART Control : UARTCTL (pg. 915 - MDS)
  ; 3. Set Baud Rate Divisor
        ; BRD = CLK/(16*BAUD) | CLK (Clock)=12 MHz, BAUD=9600
        ; BRD = 78.125
        ;; Integer: int(78.125) = 78 = 0x4E
  MOV R0,#0x4E
  STR R0,[R2,#0x24]                     ; UART Integer Baud-Rate Divisor : UARTIBRD (pg. 911 - MDS)
        ;; Fraction: int(0.125*2^6+0.5) = 9 = 0x09
  MOV R0,#0x09
  STR R0,[R2,#0x28]                     ; UART Fractional Baud-Rate Divisor : UARTFBRD (pg. 912 - MDS)
  ; 4. Set Serial Parameters
  MOV R0,#0x72                          ; UART Line Control : UARTLCRH (pg. 913 - MDS)
  STR R0,[R2,#0x2C]                     ; SPS:7 = 0b0 _ Stick parity is disabled.
                                        ; WLEN:6.5 = 0b11 = 0x3 = 8 bits
                                        ; FEN:4 = 1 _ Enable UART FIFOs
                                        ; STP2:3 = 0 _ One stop bit is transmitted at the end of the frame.
                                        ; EPS:2 = 0 _ Odd parity is performed, which checks for an odd number of 1s.
                                        ; PEN:1 = 1 _ Parity checking and generation is enabled.
                                        ; BRK:0 = 0 _ Normal use.
  ; 5. Enable TX/RX and UART
  MOV R0,#0x301                         ; UART Control : UARTCTL (pg. 915 - MDS)
  STR R0,[R2,#0x30]                     ; RXE:9 = 1 _ The receive section of the UART is enabled.
                                        ; TXE:8 = 1 _ The transmit section of the UART is enabled.
                                        ; UARTEN:0 = 1 _ The UART is enabled.

;; Timer Configuration
; >> SYSTICK (ST)                       ; Button Push Sampling (BPS)
                                        ; BPS = 5 ms
  LDR R3,=0xE000E000                    ; Core Peripherals Base Address (pg. 132 - MDS)
  ; 1. Stop Timer (ENABLE=0)
  MOV R0,#0                             ; Stop Timer
  STR R0,[R3,#0x10]                     ; SysTick Control and Status Register : STCTRL (pg. 136 - MDS)
  ; 2. Set Initial Value (RELOAD=X)
  LDR R0,=0x00000F                      ; (RELOAD+1)*(1/CLK)=T2Z, CLK=Clock, T2Z=Time to Zero
                                        ; CLK=12 MHz, T2Z=5 ms
                                        ; RELOAD=#59999 (0x00EA5F)
  STR R0,[R3,#0x14]                     ; SysTick Reload Value Register : STRELOAD (pg. 138 - MDS)
  ; 3. Clear Current Value (write to CURRENT; clears count)
  MOV R0,#0                             ; Clear Count
  STR R0,[R3,#0x18]                     ; SysTick Current Value Register : STCURRENT (pg. 139 - MDS)
  LDR R0,[R3,#0x10]                     ; COUNT=0
```

```
  ; 4. Set Clock Source (core clock: CLK_SRC=1)
  ; 5. Enable/Disable Interrupts (INTEN=0)
  ; 6. Start Counting (ENABLE=1; sets CURRENT=RELOAD)
  ;MOV R0,#0x5                        ; CLK_SRC:2=1, INTEN:1=0, ENABLE:0=1, #0b0101 = #0x5
  ;STR R0,[R3,#0x10]                  ; SysTick Control and Status Register : STCTRL (pg. 136 - MDS)

;; Ports Configuration
; >> GPIO Port F (PF)                 ; 1 x Push Button (BUTTON1)
  LDR R4,=0x40025000                  ; Port Base Address (pg. 656 - MDS)
  ; 1. Unlock Pins / Disable Alt. Function
  LDR R0,=0x4C4F434B                  ; Unlock Code (pg. 681 - MDS)
  STR R0,[R4,#0x520]                  ; GPIO Lock : GPIOLOCK (pg. 681 - MDS)
  MOV R0,#0x03                        ; Unlock Pins - PF0 (BUTTON1) (0x01=0b0000.0001)
  STR R0,[R4,#0x524]                  ; GPIO Commit : GPIOCR (pg. 682 - MDS)
  ; 2. Set Pins as Input or Output
  MOV R0,#0xFE                        ; Enable as Input - PF0 (BUTTON1) (0xFE=0b1111.1110)
  STR R0,[R4,#0x400]                  ; GPIO Direction : GPIODIR (pg. 660 - MDS)
  ; 3. Set for Pull-Up or Pull-Down
  MOV R0,#0x01                        ; Set for Pull-Up Resistor (0x01 = 0b0000.0001)
  STR R0,[R4,#0x510]                  ; GPIO Pull-Up Select : GPIOPUR (pg. 674 - MDS)
  ; 4. Enable Pins
  MOV R0,#0x01                        ; Enable Pins - PF0 (BUTTON1) (0x01 = 0b0000.0001)
  STR R0,[R4,#0x51C]                  ; GPIO Digital Enable : GPIODEN (pg. 679 - MDS)

; R0 - Empty
; R1 - Clock -> Empty
; R2 - UART 2
; R3 - System Timer (BPS)
; R4 - Port F (PF)
; R5 - Counter
; R6 - Byte of Message
; R7 - Start Flag

BEGIN MOV R5,#0                       ; Start counter.
  MOV R7,#0                           ; Clear start flag.
return CMP R5,#11                     ; Does counter need to be reset?
  BNE timer                           ; If no, proceed to start timer (ST).
  MOV R5,#0                           ; If yes, then reset the counter.
timer MOV R0,#0x5                     ; Start timer (ST).
  STR R0,[R3,#0x10]                   ; ...
wait MOV R1,#1                        ; Timer Expired Flag
  LDR R0,[R3,#0x10]                   ; Check state of timer (ST).
  CMP R1,R0,LSR #16                   ; Has timer (ST) countdown reached zero? (COUNT:16=1)
  BNE wait                            ; If no, keep checking if timer (ST) has reached zero; i.e. 5 ms.
  MOV R0,#0                           ; Stop timer (ST).
  STR R0,[R3,#0x10]                   ; ...
  ADD R5,#1                           ; Increment counter.
  LDR R1,[R4,#0x3FC]                  ; Check state of BUTTON1 (PF0); ON=0 OFF=1.
  AND R1,#0x01                        ; Isolate state of BUTTON1 (PF0).
  CMP R1,#0                           ; Is BUTTON1 (PF0) pushed?
  BEQ SPITE                           ; Select message to send to terminal.
  LDR R1,[R2,#0x18]                   ; Check Status of UARTFR. (pg. 908 - MDS)
  ANDS R1,#0x10                       ; Is result zero?
  BEQ SPITE                           ; Select message to send to terminal.
  B return                            ; ...

SPITE   ; Randomly select one of the ten messages.
  CMP R5,#1                           ; Was count at 1 at time of button or keyboard input?
  LDREQ R0,=MESSAGE01                         ; If yes, then load address of message 1.
  CMP R5,#2                           ; Was count at 2 at time of button or keyboard input?
  LDREQ R0,=MESSAGE02                         ; If yes, then load address of message 2.
  CMP R5,#3                           ; Was count at 3 at time of button or keyboard input?
  LDREQ R0,=MESSAGE03                         ; If yes, then load address of message 3.
  CMP R5,#4                           ; Was count at 4 at time of button or keyboard input?
  LDREQ R0,=MESSAGE04                         ; If yes, then load address of message 4.
  CMP R5,#5                           ; Was count at 5 at time of button or keyboard input?
  LDREQ R0,=MESSAGE05                         ; If yes, then load address of message 5.
```

```
  CMP R5,#6                          ; Was count at 6 at time of button or keyboard input?
  LDREQ R0,=MESSAGE06                        ; If yes, then load address of message 6.
  CMP R5,#7                          ; Was count at 7 at time of button or keyboard input?
  LDREQ R0,=MESSAGE07                        ; If yes, then load address of message 7.
  CMP R5,#8                          ; Was count at 8 at time of button or keyboard input?
  LDREQ R0,=MESSAGE08                        ; If yes, then load address of message 8.
  CMP R5,#9                          ; Was count at 9 at time of button or keyboard input?
  LDREQ R0,=MESSAGE09                        ; If yes, then load address of message 9.
  CMP R5,#10                         ; Was count at 10 at time of button or keyboard input?
  LDREQ R0,=MESSAGE10                        ; If yes, then load address of message 10.

repeat CMP R7,#0                     ; Is this the start of the message transmission?
  BEQ skip                           ; If yes, then proceed with processing message.
  CMP R6,#0x00                       ; Has the last byte of the message been reached?
  BEQ BEGIN                          ; If yes, then check again for button or keyboard input.
skip MOV R7,#1                       ; Processing of message has started.
  LDR R1,[R0]                        ; Load first 32-bit portion of the message.
  AND R6,R1,#0xFF                    ; Isolate first letter of the message.
  BL TX                              ; Transmit byte to terminal.
  ADD R0,#1                          ; Move to next letter of message.
  B repeat                           ; ...

TX
  PUSH {LR}
  LDR R1,[R2,#0x18]                  ; Check Status of UARTFR. (pg. 908 - MDS)
  ANDS R1,#0x20                      ; Check if buffer has space for transmitting data.
  BNE TX                             ; If not, keep checking until buffer is not full.
  STRB R6,[R2,#0x0]                  ; Transmit first byte of selected message.
  POP {LR}
  BX LR

;; Messages
MESSAGE01 DCB "Nope\n\r",0
MESSAGE02 DCB "You are doomed\n\r",0
MESSAGE03 DCB "Concentrate you fool\n\r",0
MESSAGE04 DCB "What a rubbish question\n\r",0
MESSAGE05 DCB "Only in your dreams\n\r",0
MESSAGE06 DCB "Yes now leave me alone\n\r",0
MESSAGE07 DCB "Heh you wish\n\r",0
MESSAGE08 DCB "Oh yeah that will happen\n\r",0
MESSAGE09 DCB "Stop bothering me\n\r",0
MESSAGE10 DCB "Not if you were the last person on earth\n\r",0

  ALIGN
  END
```

Commentary

- Lab 3 demonstrates the function of the UART module such that the microcontroller was able to interface with the RS-232 to transmit ASCII characters byte-by-byte. The computer terminal was successfully able to receive the ASCII bytes and display the encoded messages on screen. In the program written to the microcontroller, keyboard inputs from the computer terminal were successfully received in order to invoke the random selection of a message to transmit and print to the computer terminal monitor. Likewise, the logic analyzer shows the exact timing of the message transmissions and the translation of the bytes to their ASCII conversion.