```
; Marc Dominic Cabote
; CECS 460  Spring 2018
; Full UART Assembly
;====================================================================
; FOR DISPLAYING 0-9 AND A-Z(ALL CAPS)
;====================================================================
ascii_0         EQU     0030
ascii_1         EQU     0031
ascii_2         EQU     0032
ascii_3         EQU     0033
ascii_4         EQU     0034
ascii_5         EQU     0035
ascii_6         EQU     0036
ascii_7         EQU     0037
ascii_8         EQU     0038
ascii_9         EQU     0039
ascii_A         EQU     0041
ascii_B         EQU     0042
ascii_C         EQU     0043
ascii_D         EQU     0044
ascii_E         EQU     0045
ascii_F         EQU     0046
ascii_G         EQU     0047
ascii_H         EQU     0048
ascii_I         EQU     0049
ascii_J         EQU     004A
ascii_K         EQU     004B
ascii_L         EQU     004C
ascii_M         EQU     004D
ascii_N         EQU     004E
ascii_O         EQU     004F
ascii_P         EQU     0050
ascii_Q         EQU     0051
ascii_R         EQU     0052
ascii_S         EQU     0053
ascii_T         EQU     0054
ascii_U         EQU     0055
ascii_V         EQU     0056
ascii_W         EQU     0057
ascii_X         EQU     0058
ascii_Y         EQU     0059
ascii_Z         EQU     005A
;====================================================================
; FOR DISPLAYING SYMBOLS and LINE MANIPULATION
;====================================================================
ascii_BCKSPC    EQU     0008    ; <-
ascii_TAB       EQU     0009    ; TAB
ascii_LF        EQU     000A    ; Line Feed
ascii_CR        EQU     000D    ; Carriage Return
ascii_SPC       EQU     0020    ; SPACE
ascii_ASTERISK  EQU     002A    ; *
ascii_DASH      EQU     002D    ; "-"
ascii_DOT       EQU     002E    ; .
ascii_AT        EQU     0040    ; @

;====================================================================
; CONSTANTS
;====================================================================
ZERO            EQU     0000
ONE             EQU     0001

;====================================================================
; PORTS
;====================================================================
DATA_PORT       EQU     0000
STATUS_PORT     EQU     0001

;====================================================================
; REGISTERS
```

```
;==========================================================================
CHAR_REG        EQU     R1
CHAR_INDEX      EQU     R2
CHAR_COUNT      EQU     R3  ;for reading through registers
LED_COUNT       EQU     R4
DELAY_COUNT     EQU     R5
LEDS            EQU     R6
TEMP            EQU     R7
RB              EQU     RB
RD              EQU     RD
RE              EQU     RE
STATUS          EQU     R8
COUNT1          EQU     R9  ;for displaying dashes
COUNT_CHAR      EQU     RA  ;for counting characters
OUT_FLAG        EQU     RC  ;print flag
DATA_IN         EQU     R0  ;data received




;==========================================================================
; INITIALIZATION ; banner, prompt, hometown
;==========================================================================
START
        ;==================
        ;   CHAR INIT
        ;==================
        LOAD    CHAR_REG,       ZERO
        LOAD    CHAR_COUNT,     ZERO
        LOAD    COUNT1,         ZERO
        LOAD    CHAR_INDEX,     ZERO
        LOAD    TEMP,           ZERO
        LOAD    RD,             ZERO
        LOAD    RE,             ZERO
        LOAD    STATUS,         ZERO
        LOAD    OUT_FLAG,       ONE  ;initialized to output the banner at first
        LOAD    COUNT_CHAR,     ZERO
        ;==================
        ;   LED INIT
        ;==================
        LOAD    LEDS,           ONE
        LOAD    DELAY_COUNT,    ZERO

        ;==========================
        ;   BANNER INIT;
        ;   --------------------
        ;   .   MARC CABOTE    .
        ;   --------------------
        ;==========================

        LOAD    CHAR_REG,   ascii_DASH
DASHBEGIN
        STORE   CHAR_REG,   CHAR_INDEX
        ADD     CHAR_INDEX, ONE
        ADD     COUNT1,     ONE
        COMP    COUNT1,     0014 ; output 20 dashes
        JUMPC   DASHBEGIN

        LOAD    COUNT1,     ZERO

        LOAD    CHAR_REG,   ascii_CR
        STORE   CHAR_REG,   0015

        LOAD    CHAR_REG,   ascii_LF
        STORE   CHAR_REG,   0016

        LOAD    CHAR_REG,   ascii_DOT
        STORE   CHAR_REG,   0017

        LOAD    CHAR_REG,   ascii_SPC
        STORE   CHAR_REG,   0018

        LOAD    CHAR_REG,   ascii_SPC
        STORE   CHAR_REG,   0019

        LOAD    CHAR_REG,   ascii_SPC
        STORE   CHAR_REG,   001A

        LOAD    CHAR_REG,   ascii_M
        STORE   CHAR_REG,   001B

        LOAD    CHAR_REG,   ascii_A
        STORE   CHAR_REG,   001C

        LOAD    CHAR_REG,   ascii_R
        STORE   CHAR_REG,   001D

        LOAD    CHAR_REG,   ascii_C
        STORE   CHAR_REG,   001E

        LOAD    CHAR_REG,   ascii_SPC
        STORE   CHAR_REG,   001F

        LOAD    CHAR_REG,   ascii_C
        STORE   CHAR_REG,   0020

        LOAD    CHAR_REG,   ascii_A
        STORE   CHAR_REG,   0021

        LOAD    CHAR_REG,   ascii_B
        STORE   CHAR_REG,   0022

        LOAD    CHAR_REG,   ascii_O
        STORE   CHAR_REG,   0023

        LOAD    CHAR_REG,   ascii_T
        STORE   CHAR_REG,   0024

        LOAD    CHAR_REG,   ascii_E
        STORE   CHAR_REG,   0025

        LOAD    CHAR_REG,   ascii_SPC
        STORE   CHAR_REG,   0026

        LOAD    CHAR_REG,   ascii_SPC
        STORE   CHAR_REG,   0027

        LOAD    CHAR_REG,   ascii_SPC
        STORE   CHAR_REG,   0028

        LOAD    CHAR_REG,   ascii_SPC
        STORE   CHAR_REG,   0029

        LOAD    CHAR_REG,   ascii_DOT
        STORE   CHAR_REG,   002A

        LOAD    CHAR_REG,   ascii_CR
        STORE   CHAR_REG,   002B

        LOAD    CHAR_REG,   ascii_LF
        STORE   CHAR_REG,   002C

        LOAD    CHAR_REG,   ascii_DASH
        LOAD    CHAR_INDEX, 002D        ;to start dash at 2D
DASHEND
        STORE   CHAR_REG,   CHAR_INDEX
        ADD     CHAR_INDEX, ONE
        ADD     COUNT1,     ONE
        COMP    COUNT1,     0014 ; output 20 dashes til 0x40
        JUMPC   DASHEND

        ;==================
        ;   NEW LINE
        ;==================


        LOAD    CHAR_REG,   ascii_CR
        STORE   CHAR_REG,   0041

        LOAD    CHAR_REG,   ascii_LF
        STORE   CHAR_REG,   0042

        LOAD    COUNT1,     ZERO

        ;==================
        ;   HOMETOWN
        ;==================

        LOAD    CHAR_REG,   ascii_H
        STORE   CHAR_REG,   0043

        LOAD    CHAR_REG,   ascii_O
        STORE   CHAR_REG,   0044

        LOAD    CHAR_REG,   ascii_M
        STORE   CHAR_REG,   0045

        LOAD    CHAR_REG,   ascii_E
        STORE   CHAR_REG,   0046

        LOAD    CHAR_REG,   ascii_T
        STORE   CHAR_REG,   0047

        LOAD    CHAR_REG,   ascii_O
        STORE   CHAR_REG,   0048

        LOAD    CHAR_REG,   ascii_W
        STORE   CHAR_REG,   0049

        LOAD    CHAR_REG,   ascii_N
        STORE   CHAR_REG,   004A

        LOAD    CHAR_REG,   ascii_SPC
        STORE   CHAR_REG,   004B

        LOAD    CHAR_REG,   ascii_DASH
        STORE   CHAR_REG,   004C

        LOAD    CHAR_REG,   ascii_SPC
        STORE   CHAR_REG,   004D

        LOAD    CHAR_REG,   ascii_L
        STORE   CHAR_REG,   004E

        LOAD    CHAR_REG,   ascii_O
        STORE   CHAR_REG,   004F

        LOAD    CHAR_REG,   ascii_S
        STORE   CHAR_REG,   0050

        LOAD    CHAR_REG,   ascii_A
        STORE   CHAR_REG,   0051

        LOAD    CHAR_REG,   ascii_N
        STORE   CHAR_REG,   0052

        LOAD    CHAR_REG,   ascii_G
        STORE   CHAR_REG,   0053

        LOAD    CHAR_REG,   ascii_E
        STORE   CHAR_REG,   0054

        LOAD    CHAR_REG,   ascii_L
        STORE   CHAR_REG,   0055

        LOAD    CHAR_REG,   ascii_E
        STORE   CHAR_REG,   0056

        LOAD    CHAR_REG,   ascii_S
        STORE   CHAR_REG,   0057

        LOAD    CHAR_REG,   ascii_CR
        STORE   CHAR_REG,   0058

        LOAD    CHAR_REG,   ascii_LF
        STORE   CHAR_REG,   0059

        AND     CHAR_INDEX, ZERO    ;reset char index

        ;==================
        ;   BACKSPACE
        ;==================

        LOAD    CHAR_REG,   ascii_BCKSPC
        STORE   CHAR_REG,   005A

        LOAD    CHAR_REG,   ascii_SPC
        STORE   CHAR_REG,   005B

        LOAD    CHAR_REG,   ascii_BCKSPC
        STORE   CHAR_REG,   005C

        ;==================
        ;   PROMPT
        ;==================
        LOAD    CHAR_REG,   ascii_I
        STORE   CHAR_REG,   00B0

        LOAD    CHAR_REG,   ascii_N
        STORE   CHAR_REG,   00B1

        LOAD    CHAR_REG,   ascii_P
        STORE   CHAR_REG,   00B2

        LOAD    CHAR_REG,   ascii_U
        STORE   CHAR_REG,   00B3

        LOAD    CHAR_REG,   ascii_T
        STORE   CHAR_REG,   00B4

        LOAD    CHAR_REG,   ascii_DASH
        STORE   CHAR_REG,   00B5

        LOAD    CHAR_REG,   ascii_DASH
        STORE   CHAR_REG,   00B6

        ENINT
;==================================================================
;               OUTPUT a walking LED
;==================================================================
MAIN
        ADD     LED_COUNT,      ONE
        ADDC    DELAY_COUNT,    ZERO
        COMP    DELAY_COUNT,    0007 ; delay
        JUMPC   DONE_LED

        LOAD    LED_COUNT,      ZERO
        LOAD    DELAY_COUNT,    ZERO
        RL      LEDS  ;rotate LEDS

DONE_LED
       OUTPUT   LEDS,           0002
       JUMP     MAIN
;====================================================================
; Interrupt Service Routine
; Starts at address 0300
; CHECK RXRDY
; CHECK TXRDY
;====================================================================

        ADDRESS 0300
ISR
        ;FETCH  CHAR_REG,       COUNT
        ;OUTPUT CHAR_REG,   DATA_PORT
        ;ADD        COUNT,      ONE

        INPUT   STATUS,     STATUS_PORT
        AND     STATUS,     0003

        COMP    STATUS,     0003        ;IF BOTH RXRDY and TXRDY
        JUMPZ   BOTH_RDY

        COMP    STATUS,     0002        ;IF TXRDY
        CALLZ   TX_F

        COMP    STATUS,     0001        ;IF RXRDY
        CALLZ   RX_F

        RETEN

BOTH_RDY
        CALL    TX_F
        CALL    RX_F
        RETEN


;========================================================================
;   ADDRESS FOR BIN_TO_ASCII
;========================================================================
BIN_TO_ASCII
        LOAD    RE, COUNT_CHAR

        LOAD    RD, 000A    ;RD<-10
        CALL    FIND_IT
        ADD     RB, 0030    ;convert to ascii_hex
        STORE   RB, 00A0    ;store counter at 00A0

        ADD     RE, 0030    ;convert to ascii_hex
        STORE   RE, 00A1    ;store counter at 00A1

        RETURN
;========================================================================
;   BIN_TO_ASCII FIND_IT FUNCTION
;========================================================================
FIND_IT
        LOAD    RB, 0000    ;RB<-0000
NOT_DONE
        SUB     RE, RD      ; RE<-RE-RD
        JUMPC   RESTORE     ; if there is a carry restore RE
        ADD     RB, 0001    ; increment RB
        JUMP    NOT_DONE    ; keep subtracting
RESTORE
        ADD     RE, RD    ; restore last value
        RETURN            ; return bin to ascii routine
;========================================================================
;                 TX
;===================================================================
TX_F
        COMP    OUT_FLAG,   ZERO
        RETURNZ

        FETCH   TEMP,       CHAR_COUNT
        OUTPUT  TEMP,       DATA_PORT
        ADD     CHAR_COUNT, ONE

        COMP    OUT_FLAG,   ONE
        JUMPZ   BANNER_OUT

        COMP    OUT_FLAG,   0002
        JUMPZ   PROMPT_OUT

        COMP    OUT_FLAG,   0003
        JUMPZ   HOMETOWN_OUT

        COMP    OUT_FLAG,   0004
        JUMPZ   COUNT_OUT

        COMP    OUT_FLAG,   0005
        JUMPZ   BACKSPACE_OUT

        COMP    OUT_FLAG,   0006
        JUMPZ   NEW_LINE

        RETURN

BANNER_OUT
        COMP    CHAR_COUNT, 0043    ;address where banner ends
        RETURNC
        LOAD    OUT_FLAG,   0002    ;prompt
        LOAD    CHAR_COUNT, 00B0    ;where prompt starts
        RETURN

PROMPT_OUT
        COMP    CHAR_COUNT, 00B7    ;address where prompt ends
        RETURNC
        LOAD    OUT_FLAG,   0000    ;wait for RX
        RETURN

HOMETOWN_OUT
        COMP    CHAR_COUNT, 005A    ;address where hometown ends
        RETURNC
        LOAD    CHAR_COUNT, 0041    ;address where prompt starts
        LOAD    OUT_FLAG,   0002    ;prompt
        LOAD    CHAR_COUNT, 00B0    ;where prompt starts
        RETURN

COUNT_OUT
        COMP    CHAR_COUNT, 00A2    ;address where character count ends
        RETURNC
        LOAD    CHAR_COUNT, 0041    ;new line
        LOAD    OUT_FLAG,   0006
        RETURN

BACKSPACE_OUT
        COMP    CHAR_COUNT, 005D    ;address where backspace ends
        RETURNC
        LOAD    OUT_FLAG,   0000    ;wait for RX
        RETURN

NEW_LINE
        COMP    CHAR_COUNT, 0043    ;address where new line ends
        RETURNC
        LOAD    CHAR_COUNT, 0041    ;address where prompt starts
        LOAD    OUT_FLAG,   0002    ;prompt
        LOAD    CHAR_COUNT, 00B0    ;where prompt starts
        RETURN

;===========================================================================
;               RX
;===========================================================================
RX_F
        COMP    OUT_FLAG,   ZERO    ;wait for input
        RETURNNZ

        INPUT   DATA_IN,        ZERO
        COMP    DATA_IN,        ZERO    ;if no input
        RETURNZ

        COMP    DATA_IN,        ascii_ASTERISK  ; if *
        JUMPZ   GO_HOMETOWN

        COMP    DATA_IN,        ascii_AT        ; if @
        JUMPZ   GO_CHARCOUNT

        COMP    DATA_IN,        ascii_BCKSPC    ; if backspace
        JUMPZ   GO_BACKSPACE

        COMP    DATA_IN,        ascii_CR        ; if return
        JUMPZ   GO_NEWLINE

        ADD     COUNT_CHAR, ONE                 ;increment char counter
        OUTPUT  DATA_IN,    DATA_PORT           ;output char counter
        COMP    COUNT_CHAR, 0028                ;conter cannot exceed 40
        JUMPZ   GO_NEWLINE
        RETURN

GO_HOMETOWN
        LOAD    OUT_FLAG,   0003    ;hometown out
        LOAD    CHAR_COUNT, 0043    ;address where hometown starts
        LOAD    TEMP,       ZERO
        OUTPUT  TEMP,       DATA_PORT
        LOAD    COUNT_CHAR, ZERO    ;characters counter
        RETURN

GO_CHARCOUNT
        CALL    BIN_TO_ASCII
        LOAD    OUT_FLAG,   0004    ;char count out
        LOAD    CHAR_COUNT, 00A0    ;address where counter starts
        LOAD    TEMP,       ZERO
        OUTPUT  TEMP,       DATA_PORT
        LOAD    COUNT_CHAR, ZERO    ;characters counter
        RETURN

GO_BACKSPACE
        COMP    COUNT_CHAR, ZERO
        RETURNZ
        LOAD    OUT_FLAG,   0005    ;backspace out
        LOAD    CHAR_COUNT, 005A    ;address where backspace starts
        LOAD    TEMP,       ZERO
        OUTPUT  TEMP,       DATA_PORT
        SUB     COUNT_CHAR, 0001    ;decrement characters counter
        RETURN

GO_NEWLINE
        LOAD    OUT_FLAG,   0006    ;newline out
        LOAD    CHAR_COUNT, 0041    ;address where newline begins
        LOAD    TEMP,       ZERO
        OUTPUT  TEMP,       DATA_PORT
        LOAD    COUNT_CHAR, ZERO
        RETURN

;=====================================================================
;   ISR VECTORED THROUGH 0FFE
```

```
;===================================================================
        ADDRESS 0FFE
ENDIT
        JUMP ISR
        END
;===================================================================
;                           END
;===================================================================
```