```
; Serial C/ASM Mix Example
; Jason Losh

;-----------------------------------------------------------------------------
; Hardware Target
;-----------------------------------------------------------------------------

; Target Platform: EK-TM4C123GXL Evaluation Board
; Target uC:       TM4C123GH6PM
; System Clock:    40 MHz

; Hardware configuration:
; Red LED:
;   PF1 drives an NPN transistor that powers the red LED
; Green LED:
;   PF3 drives an NPN transistor that powers the green LED
; Pushbutton:
;   SW1 pulls pin PF4 low (internal pull-up is used)
; UART Interface:
;   U0TX (PA1) and U0RX (PA0) are connected to the 2nd controller
;   The USB on the 2nd controller enumerates to an ICDI interface and a virtual COM port
;   Configured to 115,200 baud, 8N1

;-----------------------------------------------------------------------------
; Device includes, defines, and assembler directives
;-----------------------------------------------------------------------------

   .def waitPbPress
   .def putcUart0
   .def putsUart0
   .def getcUart0

;-----------------------------------------------------------------------------
; Register values and large immediate values
;-----------------------------------------------------------------------------

.thumb
.text
GPIO_PORTF_DATA_R       .field   0x400253FC
UART0_FR_R              .field   0x4000C018
UART0_DR_R              .field   0x4000C000

;-----------------------------------------------------------------------------
; Subroutines
;-----------------------------------------------------------------------------

; Blocking function that returns only when SW1 is pressed
waitPbPress:
               LDR    R0, GPIO_PORTF_DATA_R  ; get pointer to port F
               LDR    R0, [R0]               ; read port F
               AND    R0, #0x10              ; mask off all but bit 4
               CBNZ   R0, retry              ; 0 if bit set test (note: only support 0-126 branches)
               BX     R14                    ; return from subroutine
retry:         B      waitPbPress

```

```
; Blocking function that writes serial data when the buffer is not full
putcUart0:
               LDR    R1, UART0_FR_R         ; get pointer to UART0 FR register
               LDR    R1, [R1]               ; read FR
               AND    R1, #0x20              ; mask off all but bit 5 (TX full)
               CBNZ   R1, retryPutcUart      ; 1 if full
               LDR    R1, UART0_DR_R         ; get pointer to UART data register
               STR    R0, [R1]               ; write transmit data
               BX     LR                     ; return from subroutine
retryPutcUart: B      putcUart0

; Blocking function that writes a string when the UART buffer is not full
putsUart0:
               PUSH   {R4, LR}               ; save R4 and LR (return add to caller of this
function)
               MOV    R4, R0                 ; copy string pointer to R4 where it is safe
before putcUart0 call
nextPutsUart:  LDRB   R0, [R4], #1           ; read next character of string
               CBZ    R0, donePutsUart       ; if null terminator, exit
               BL     putcUart0              ; push LR, call putsUart0
               B      nextPutsUart
donePutsUart:  POP    {R4, PC}               ; pop off R4, pop off return address into PC
(easier than POP LR, BX LR)

; Blocking function that returns with serial data once the buffer is not empty
getcUart0:
               LDR    R0, UART0_FR_R         ; get pointer to UART0 FR register
               LDR    R0, [R0]               ; read FR
               AND    R0, #0x10              ; mask off all but bit 4 (RX empty)
               CBNZ   R0, retryGetcUart      ; 1 if empty
               LDR    R0, UART0_DR_R         ; get pointer to UART data register
               LDR    R0, [R0]               ; read received data
               AND    R0, #0xFF              ; mask off all but bits 0-7
               BX     R14                    ; return from subroutine
retryGetcUart: B      getcUart0

.endm

```