```
***********************************************************************
*
* Title:          SCI Serial Port and 7-segment Display at PORTB
*
* Objective:      CMPEN 472 Homework 5, in-class-room demonstration
*                 program
*
* Revision:       V3.2  for CodeWarrior 5.2 Debugger Simulation
*
* Date:           Sep. 16, 2020
*
* Programmer:     Kyusun Choi
*
* Company:        The Pennsylvania State University
*                 Department of Computer Science and Engineering
*
* Program:        Simple SCI Serial Port I/O and Demonstration
*                 Typewriter program and 7-Segment display, at PORTB
*
*
* Algorithm:      Simple Serial I/O use, typewriter
*
* Register use:   A: Serial port data
*                 X,Y: Delay loop counters
*
* Memory use:     RAM Locations from $3000 for data,
*                 RAM Locations from $3100 for program
*
* Output:
*                 PORTB bit 7 to bit 4, 7-segment MSB
*                 PORTB bit 3 to bit 0, 7-segment LSB
*
* Observation:    This is a typewriter program that displays ASCII
*                 data on PORTB - 7-segment displays.
*
***********************************************************************
* Parameter Declearation Section
*
* Export Symbols
            XDEF        pstart       ; export 'pstart' symbol
            ABSENTRY    pstart       ; for assembly entry point

* Symbols and Macros
PORTB       EQU         $0001        ; i/o port B addresses
DDRB        EQU         $0003

SCIBDH      EQU         $00C8        ; Serial port (SCI) Baud Register H
SCIBDL      EQU         $00C9        ; Serial port (SCI) Baud Register L
SCICR2      EQU         $00CB        ; Serial port (SCI) Control Register 2
SCISR1      EQU         $00CC        ; Serial port (SCI) Status Register 1
SCIDRL      EQU         $00CF        ; Serial port (SCI) Data Register

CR          equ         $0d          ; carriage return, ASCII 'Return' key
LF          equ         $0a          ; line feed, ASCII 'next line' character

***********************************************************************
* Data Section: address used [ $3000 to $30FF ] RAM memory
*
            ORG         $3000        ; Reserved RAM memory starting address
                                     ;   for Data for CMPEN 472 class
Counter1    DC.W        $008F        ; X register count number for time delay
                                     ;   inner loop for msec
Counter2    DC.W        $000C        ; Y register count number for time delay
                                     ;   outer loop for sec
```

```
msg1        DC.B        'Hello', $00
msg2        DC.B        'You may type below', $00
; Each message ends with $00 (NULL ASCII character) for your program.
;
; There are 256 bytes from $3000 to $3100.  If you need more bytes for
; your messages, you can put more messages 'msg3' and 'msg4' at the end of
; the program - before the last "END" line.
                                     ; Remaining data memory space for stack,
                                     ;   up to program memory start

*
***********************************************************************
* Program Section: address used [ $3100 to $3FFF ] RAM memory
*
            ORG        $3100        ; Program start address, in RAM
pstart      LDS        #$3100       ; initialize the stack pointer

            LDAA       #%11111111   ; Set PORTB bit 0,1,2,3,4,5,6,7
            STAA       DDRB         ; as output

            LDAA       #%00000000
            STAA       PORTB        ; clear all bits of PORTB

            ldaa       #$0C         ; Enable SCI port Tx and Rx units
            staa       SCICR2       ; disable SCI interrupts

            ldd        #$0001        ; Set SCI Baud Register = $0001 =>   1.5M baud at 24MHz
(for simulation)
;            ldd        #$000D       ; Set SCI Baud Register = $000D => 115200 baud at 24MHz
;            ldd        #$009C       ; Set SCI Baud Register = $009C =>   9600 baud at 24MHz
            std        SCIBDH        ; SCI port baud rate change

            ldx   #msg1              ; print the first message, 'Hello'
            jsr   printmsg

            ldaa  #CR                ; move the cursor to beginning of the line
            jsr   putchar            ;   Cariage Return/Enter key
            ldaa  #LF                ; move the cursor to next line, Line Feed
            jsr   putchar

            ldx   #msg2              ; print the second message
            jsr   printmsg

            ldaa  #CR                ; move the cursor to beginning of the line
            jsr   putchar            ;   Cariage Return/Enter key
            ldaa  #LF                ; move the cursor to next line, Line Feed
            jsr   putchar

looop       jsr   getchar            ; type writer - check the key board
            cmpa  #$00               ;  if nothing typed, keep checking
            beq   looop
                                       ;  otherwise - what is typed on key board
            jsr   putchar            ; is displayed on the terminal window - echo print

            staa  PORTB              ; show the character on PORTB

            cmpa  #CR
            bne   looop              ; if Enter/Return key is pressed, move the
            ldaa  #LF                ; cursor to next line
            jsr   putchar
            bra   looop


;subroutine section below
```

```
;***********printmsg***************************
;* Program: Output character string to SCI port, print message
;* Input:   Register X points to ASCII characters in memory
;* Output:  message printed on the terminal connected to SCI port
;* 
;* Registers modified: CCR
;* Algorithm:
;     Pick up 1 byte from memory where X register is pointing
;     Send it out to SCI port
;     Update X register to point to the next byte
;     Repeat until the byte data $00 is encountered
;       (String is terminated with NULL=$00)
;**********************************************
NULL           equ     $00
printmsg       psha                   ;Save registers
               pshx
printmsgloop   ldaa    1,X+           ;pick up an ASCII character from string
                                       ;   pointed by X register
                                       ;then update the X register to point to
                                       ;   the next byte
               cmpa    #NULL
               beq     printmsgdone   ;end of strint yet?
               jsr     putchar        ;if not, print character and do next
               bra     printmsgloop

printmsgdone   pulx 
               pula
               rts
;***********end of printmsg********************


;***************putchar************************
;* Program: Send one character to SCI port, terminal
;* Input:   Accumulator A contains an ASCII character, 8bit
;* Output:  Send one character to SCI port, terminal
;* Registers modified: CCR
;* Algorithm:
;    Wait for transmit buffer become empty
;      Transmit buffer empty is indicated by TDRE bit
;      TDRE = 1 : empty - Transmit Data Register Empty, ready to transmit
;      TDRE = 0 : not empty, transmission in progress
;**********************************************
putchar        brclr SCISR1,#%10000000,putchar   ; wait for transmit buffer empty
               staa  SCIDRL                      ; send a character
               rts
;***************end of putchar*****************


;****************getchar***********************
;* Program: Input one character from SCI port (terminal/keyboard)
;*             if a character is received, other wise return NULL
;* Input:   none    
;* Output:  Accumulator A containing the received ASCII character
;*          if a character is received.
;*          Otherwise Accumulator A will contain a NULL character, $00.
;* Registers modified: CCR
;* Algorithm:
;    Check for receive buffer become full
;      Receive buffer full is indicated by RDRF bit
;      RDRF = 1 : full - Receive Data Register Full, 1 byte received
;      RDRF = 0 : not full, 0 byte received
;**********************************************
getchar        brclr SCISR1,#%00100000,getchar7
               ldaa  SCIDRL
               rts
```

```
getchar7       clra
               rts
;****************end of getchar****************


;OPTIONAL
;more variable/data section below
; this is after the program code section
; of the RAM.  RAM ends at $3FFF
; in MC9S12C128 chip

msg3           DC.B    'Enter your command below:', $00
msg4           DC.B    'Error: Invalid command', $00


               END               ; this is end of assembly source file
                                 ; lines below are ignored - not assembled/compiled
```