```
;*******************************************************
;* CMPEN 472, HW8 Real Time Interrupt, MC9S12C128 Program
;* CodeWarrior Simulator/Debug edition, not for CSM-12C128 board
;* Oct.  13, 2016 Kyusun Choi
;* Oct.  14, 2020 Kyusun Choi
;* Oct.  23, 2021 Kyusun Choi
;* Oct.  23, 2022 Kyusun Choi
;*
;* 1 second LED1 blink, timer using Real Time Interrupt.
;* This program is a 1 second timer using
;* a Real Time Interrupt service subroutine (RTIISR).  This program
;* displays the time on the 7 Segment Disply in Visualization Tool
;* every 1 second.  That is, this program
;* displays '1 0 1 0 1 0 . . . ' on the 7 segment displys.
;* The 7 segment displys are connected to port B of
;* MC9S12C32 chip in CodeWarrior Debugger/Simulator.
;* Also on the Terminal component of the simulator,
;* user may enter any key, it will be displayed on the screen - effectively
;* it is a typewriter.
;*
;* Please note the new feature of this program:
;* RTI vector, initialization of CRGFLG, CRGINT, RTICTL, registers for the
;* Real Time Interrupt.
;* We assumed 24MHz bus clock and 4MHz external resonator clock frequency.
;*
;*******************************************************
;*******************************************************

; export symbols - program starting point
            XDEF        Entry        ; export 'Entry' symbol
            ABSENTRY    Entry        ; for assembly entry point

; include derivative specific macros
PORTA       EQU         $0000
PORTB       EQU         $0001
DDRA        EQU         $0002
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

CRGFLG      EQU         $0037        ; Clock and Reset Generator Flags
CRGINT      EQU         $0038        ; Clock and Reset Generator Interrupts
RTICTL      EQU         $003B        ; Real Time Interrupt Control

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

;*******************************************************
; variable/data section
            ORG    $3000             ; RAMStart defined as $3000
                                     ; in MC9S12C128 chip

timeh       DS.B   1                 ; Hour
timem       DS.B   1                 ; Minute
times       DS.B   1                 ; Second
ctr2p5m     DS.W   1                 ; interrupt counter for 2.5 mSec. of time

msg1        DC.B   'Hello', $00
msg2        DC.B   'You may type below', $00
```

```
;*******************************************************
; interrupt vector section
            ORG    $FFF0            ; RTI interrupt vector setup for the simulator
;            ORG    $3FF0            ; RTI interrupt vector setup for the CSM-12C128 board
            DC.W   rtiisr

;*******************************************************
; code section

            ORG    $3100
Entry
            LDS    #Entry         ; initialize the stack pointer

            LDAA   #%11111111   ; Set PORTA and PORTB bit 0,1,2,3,4,5,6,7
            STAA   DDRA         ; all bits of PORTA as output
            STAA   PORTA        ; set all bits of PORTA, initialize
            STAA   DDRB         ; all bits of PORTB as output
            STAA   PORTB        ; set all bits of PORTB, initialize

            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    #$0002       ; Set SCI Baud Register = $0002 => 750K baud at 24MHz
;            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
            jsr    nextline

            ldx    #msg2          ; print the second message
            jsr    printmsg
            jsr    nextline

            bset   RTICTL,%00011001 ; set RTI: dev=10*(2**10)=2.555msec for C128 board
                                    ;      4MHz quartz oscillator clock
            bset   CRGINT,%10000000 ; enable RTI interrupt
            bset   CRGFLG,%10000000 ; clear RTI IF (Interrupt Flag)


            ldx    #0
            stx    ctr2p5m          ; initialize interrupt counter with 0.
            cli                     ; enable interrupt, global

looop       jsr    LEDtoggle        ; if 0.5 second is up, toggle the LED

            jsr    getchar          ; type writer - check the key board
            tsta                    ;  if nothing typed, keep checking
            beq    looop
                                    ;  otherwise - what is typed on key board
            jsr    putchar          ; is displayed on the terminal window
            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

;***********RTI interrupt service routine***************
rtiisr      bset   CRGFLG,%10000000 ; clear RTI Interrupt Flag - for the next one
```

```
            ldx    ctr2p5m          ; every time the RTI occur, increase
            inx                     ;    the 16bit interrupt count
            stx    ctr2p5m
rtidone     RTI
;***********end of RTI interrupt service routine********

;***************LEDtoggle**********************
;* Program: toggle LED if 0.5 second is up
;* Input:   ctr2p5m variable
;* Output:  ctr2p5m variable and LED1
;* Registers modified: CCR
;* Algorithm:
;    Check for 0.5 second passed
;      if not 0.5 second yet, just pass
;      if 0.5 second has reached, then toggle LED and reset ctr2p5m
;**********************************************
LEDtoggle   psha
            pshx

            ldx    ctr2p5m          ; check for 0.5 sec
;            cpx    #200             ; 2.5msec * 200 = 0.5 sec
            cpx    #40              ; 2.5msec * 40 = 0.1 sec
            blo    doneLED          ; NOT yet

            ldx    #0               ; 0.5sec is up,
            stx    ctr2p5m          ;     clear counter to restart

            LDAA   PORTB
            EORA   #%00000001       ; Toggle the PORTB bit 4, LED1
            STAA   PORTB

            ldaa   #'*'             ; also print a '*' on the screen
            jsr    putchar

doneLED     pulx
            pula
            rts
;***************end of LEDtoggle***************

;***********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?
                bsr     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****************

;****************nextline**********************
nextline    psha
            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
            pula
            rts
;****************end of nextline***************


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