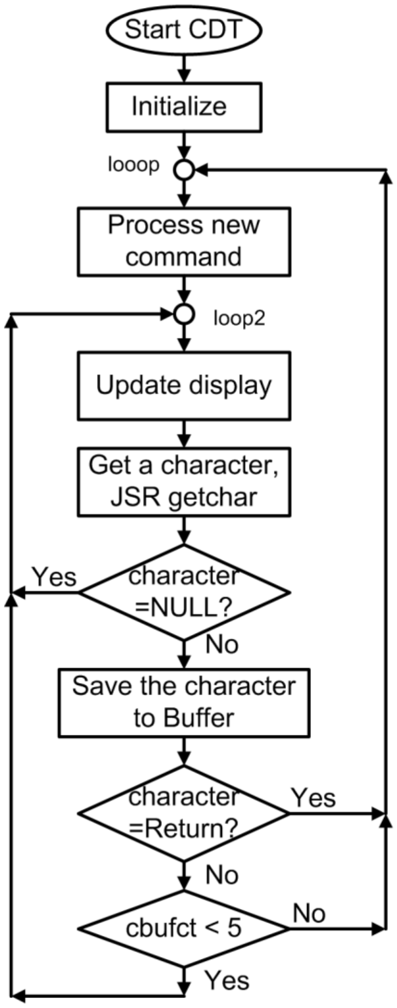
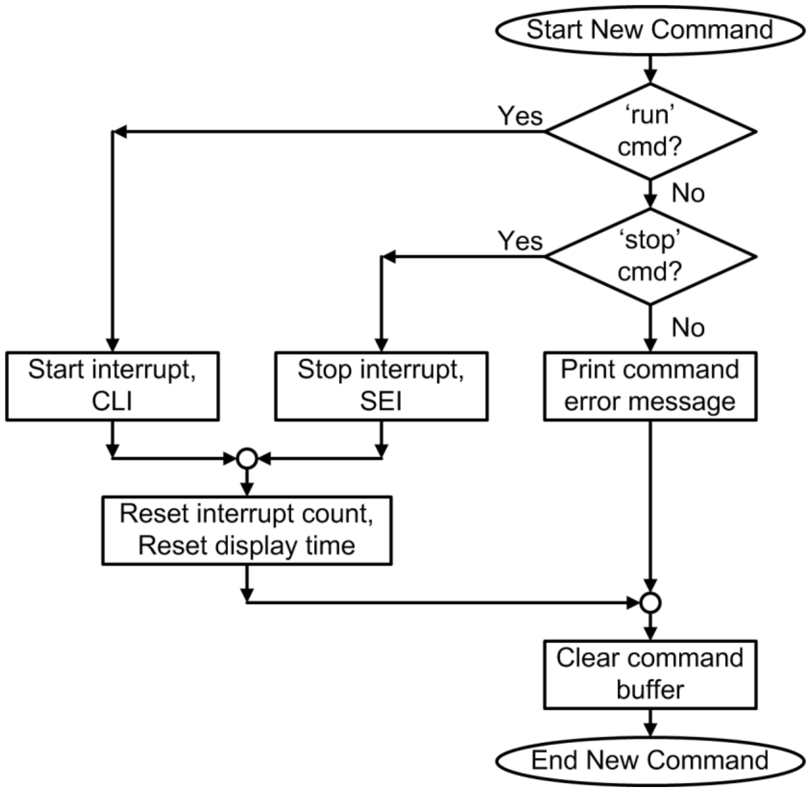
```
;*******************************************************
;* CMPEN 472, HW8 Real Time Interrupt, MC9S12C128 Program
;* April 9,2015 Kyusun Choi
;*
;* 10 second timer using Real Time Interrupt.
;* This program is a 10 second count down timer using
;* a Real Time Interrupt service subroutine (RTIISR).  This program
;* displays the time remaining on the Hyper Terminal screen every 1 second.
;* That is, this program displays '987654321098765432109876543210 . . . ' on the
;* Hyper Terminal connected to MC9S12C128 chip on CSM-12C128 board.
;* User may enter 'stop' command followed by an enter key to stop the timer
;* and re-start the timer with 'run' command followed by an enter key.
;* This program evaluates user input (command) after the enter key hit and allow
;* maximum five characters for user input.  This program ignores the wrong
;* user inputs, and continue count down.
;*******************************************************
```

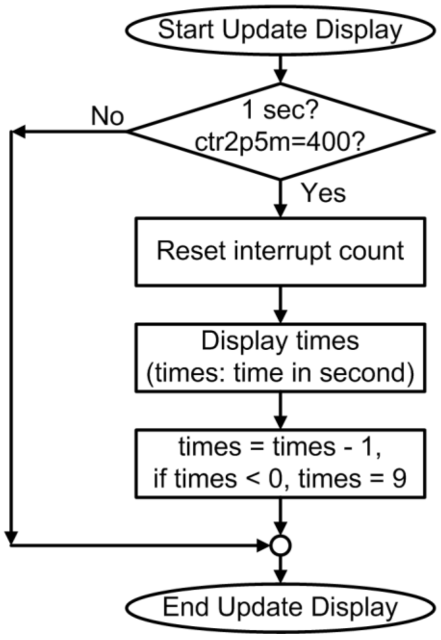


CDT: Count Down Timer
Variables

cbuf:	command buffer
cbufct:	command buffer fill count
ctr2p5m:	RTI interrupt counter, 16bit

staa 1,x+
cbufct = cbufct + 1


Start New Command
'run' cmd?
Yes
No
'stop' cmd?
Yes
No
Start interrupt, CLI
Stop interrupt, SEI
Print command error message
Reset interrupt count, Reset display time
Clear command buffer
End New Command


Start Update Display
1 sec? ctr2p5m=400?
No
Yes
Reset interrupt count
Display times (times: time in second)
times = times - 1, if times < 0, times = 9
End Update Display

```
;*******************************************************
;* CMPEN 472, HW8 Real Time Interrupt, MC9S12C128 Program
;* April 9,2015 Kyusun Choi
;*
;* 10 second timer using Real Time Interrupt.
;* This program is a 10 second count down timer using
;* a Real Time Interrupt service subroutine (RTIISR).  This program
;* displays the time remaining on the Hyper Terminal screen every 1 second.
;* That is, this program displays '987654321098765432109876543210 . . . ' on the
;* Hyper Terminal connected to MC9S12C128 chip on CSM-12C128 board.
;* User may enter 'stop' command followed by an enter key to stop the timer
;* and re-start the timer with 'run' command followed by an enter key.
;*
;* 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.
;* This program evaluates user input (command) after the enter key hit and allow
;* maximum five characters for user input.  This program ignores the wrong
;* user inputs, and continue count down.
;*
;*******************************************************
; export symbols
            XDEF Entry               ; export 'Entry' symbol
            ABSENTRY Entry           ; for assembly entry point

; include derivative specific macros
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

ctr2p5m     DS.W  1                  ; 16bit interrupt counter for 2.5 mSec. of time
cbufct      DS.B  1                  ; user input character buffer fill count
cbuf        DS.B  6                  ; user input character buffer

times       DS.B  1                  ; time to display on screen
timem       DS.B  1
timeh       DS.B  1

msg1        DC.B  'Hello', $00
msg2        DC.B  'This is 10 second count down timer program.', $00
msg3err     DC.B  'Command Error', $00

StackSP                              ; Stack space reserved from here to
                                     ; StackST

            ORG  $3FF0               ; Real Time Interrupt (RTI) interrupt vector setup
            DC.W RTIISR

            ORG  $3100
StackST

;*******************************************************
; code section
Entry
            LDS        #StackST      ; initialize the stack pointer

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

            ldx   #msg2              ; print the second message
            jsr   printmsg
            jsr   nextline
            ldx   #398
            stx   ctr2p5m            ; initialize interrupt counter with 400.
            ldaa  #9
            staa  times              ; initialize 10 second timer with #9

            ldx   #cbuf              ; set up initial command
            ldaa  #'r'               ; start with 'run' command
            staa  1,x+
            ldaa  #'u'
            staa  1,x+
            ldaa  #'n'
            staa  1,x+
            ldaa  #CR
            staa  1,x+
            ldaa  #4
            staa  cbufct

            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)


looop       jsr   NewCommand         ; check command buffer for a new command entered.

loop2       jsr   UpDisplay          ; update display, each 1 second

            jsr   getchar            ; user may enter command
            tsta                     ;  save characters if typed
            beq   loop2

            staa  1,x+               ; save the user input character
            inc   cbufct
            jsr   putchar            ; echo print, displayed on the terminal window

            cmpa  #CR
            bne   loop3              ; if Enter/Return key is pressed, move the
            ldaa  #LF                ;   cursor to next line
            jsr   putchar
            bra   looop              ; and evaluate the new command entered so far.

loop3       ldaa  cbufct             ; if user typed 5 character, it is the maximum, reset command
            cmpa  #5                 ;   is in error, ignore the input and continue timer
            blo   loop2

            bra   looop


;subroutine section below

;***********RTI interrupt service routine***************
RTIISR      bset  CRGFLG,%10000000   ; clear RTI Interrupt Flag
            ldx   ctr2p5m
            inx
            stx   ctr2p5m            ; every time the RTI occur, increase interrupt count
rtidone     RTI
;***********end of RTI interrupt service routine********

;***************Update Display*********************
;* Program: Update count down timer display if 1 second is up
;* Input:   ctr2p5m variable
;* Output:  timer display on the Hyper Terminal
;* Registers modified: CCR
;* Algorithm:
;    Check for 1 second passed
;      if not 1 second yet, just pass
;      if 1 second has reached, then update display, toggle LED, and reset ctr2p5m
;**********************************************
UpDisplay
            psha
            pshx
            ldx   ctr2p5m          ; check for 1 sec
            cpx   #399             ; 2.5msec * 400 = 1 sec        0 to 399 count is 400
            blo   UpDone           ; if interrupt count less than 400, then not 1 sec yet.
                                   ;    no need to update display.
            ldx   #0               ; interrupt counter reached 400 count, 1 sec up now
            stx   ctr2p5m          ; clear the interrupt count to 0, for the next 1 sec.

            ldaa  #$30             ; timer display update, with ASCII character
            adda  times            ; display present time
            jsr   putchar

            dec   times            ; update time for next time display
            bpl   UpDone           ; if -1 < times < 10 then OK, if not, reset times to 9 to restart
            ldaa  #9               ; reset because count display down to 0
            staa  times

UpDone      pulx
            pula
            rts
;***************end of Update Display***************

;***************New Command Process*******************************
;* Program: Check for 'run' command or 'stop' command.
;* Input:   Command buffer filled with characters, and the command buffer character count
;*             cbuf, cbufct
;* Output:  Display on Hyper Terminal, count down characters 9876543210 displayed each 1 second
;*             continue repeat unless 'stop' command.
;*          When a command is issued, the count display reset and always starts with 9.
;*          Interrupt start with CLI for 'run' command, interrupt stops with SEI for 'stop' command.
;*          When a new command is entered, cound time always reset to 9, command buffer cleared,
;*             print error message if error.  And X register pointing at the begining of
;*             the command buffer.
;* Registers modified: X, CCR
;* Algorithm:
;*     check 'run' or 'stop' command, and start or stop the interrupt
;*     print error message if error
;*     clear command buffer
;*     Please see the flow chart.
;*
;**********************************************
NewCommand
            psha

            ldx   #cbuf            ; read command buffer, see if 'run' or 'stop' command entered
            ldaa  1,x+             ;    each command is followed by an enter key
            cmpa  #'r'
            beq   ckrun2
            cmpa  #'s'
            bne   CNerror

ckstop2     ldaa  1,x+             ; check if 'stop' command
            cmpa  #'t'             ;    's' and 'top' with enter key CR.
            bne   CNerror
            ldaa  1,x+
            cmpa  #'o'
            bne   CNerror
            ldaa  1,x+
            cmpa  #'p'
            bne   CNerror
            ldaa  1,x+
            cmpa  #CR
            bne   CNerror
CNoff       sei                    ; it is 'stop' command, turn off interrupt
            bra   CNdone

ckrun2      ldaa  1,x+             ; check if 'run' command
            cmpa  #'u'             ;    'r' and 'un' with enter key CR.
            bne   CNerror
            ldaa  1,x+
            cmpa  #'n'
            bne   CNerror
            ldaa  1,x+
            cmpa  #CR
            bne   CNerror
CNonn       cli                    ; it is 'run' command, turn on interrupt
;            bra   CNdone

CNdone      ldx   #398             ; with new command, restart 10 second timer
            stx   ctr2p5m          ; initialize interrupt counter with 400.
            ldaa  #9
            staa  times            ; initialize 10 second timer with #9
            bra   CNexit

CNerror     ldx   #msg3err         ; print the 'Command Error' message
            jsr   printmsg
            jsr   nextline

CNexit      clr   cbufct           ; reset command buffer
            ldx   #cbuf

            pula
            rts
;***************end of New Command Process***************


;***********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    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
            rts
;****************end of nextline***************


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