Introduction:
Interrupt processing is probably the most useful and powerful features in a microcontroller. In this exam you will program the Real-Time Interrupt (RTI) system in the HC12 to provide periodic real-time interrupts. You will use these interrupts to determine when to sample a channel in the analog to digital converter system (ATD) in the HC12 and to update the display on the PC to show the voltage currently applied to the analog input. In this configuration, the M68EVB912B32 and PC will function as a rudimentary voltmeter.

Experiment:
Part 1: To measure the passage of one second, you will set the RTI subsystem to interrupt every 8.192 ms and then have an interrupt service routing (ISR) manage a counter to determine when enough 8.192 ms interrupts have been triggered so that you know one second has passed. When a second has passed, the ISR will then call functions to: read the ATD; convert the raw value to a voltage; and display the voltage on the PC screen (via the SCI). (Note: To test your code, you would connect a potentiometer between ground and +5V and then us the pot to provide various voltages to the ATD.)

;***************************************************************
; Lab Digital Volt Meter
;***************************************************************
;***************************************************************
; In this lab you will implement a simple volt meter using the ATD ; system in the HC12 as well as the HC12 interrupt system. ; The program will sample an analog input channel once each second, ; convert the value into a 3-digit decimal number (N.NN), and update ; the value displayed on the PC screen.
;***************************************************************
;***************************************************************
; some ASCII chars
; ESC  equ 1bh
RET   equ 0dh
LF    equ 0ah
CR    equ 0dh
BACKUP equ 08h
; HC12 register declarations/usage
; Real Time Interrupt
; rtictl  equ 0014h ; RTI control register
rtiflg equ 0015h ; RTI flag register
; Analog-to-Digital Converter
; atdctl2 equ 0062h ; ATD control register
atdctl4 equ 0064h ; ATD control register
Main program for volt meter

After performing the initializations, main outputs
the string "V = " to the terminal screen and then waits for
an ESC character to be entered (to terminate the application).

org 0800h
main bra main_cont ; we are using this trick to nail down rti_isr

; RTI interrupt service routine
; This routine keeps track of when "one second's worth" of
; RTI interrupts has passed and calls the SAMPLE routine
; followed by the VDISP routine
;
; Declarations
;
rticlr equ 80h ; real time interrupt flag bit mask
rticnt fcb 00h ; running interrupt count
rtifin equ 122t ; integration period count

rti_isr ; be sure to enter the address for this function into
; the jump vector in byte addressable ROM

Problem #1 < place your code for the RTI interrupt service routine here >
jsr init_rti_atd ; initialize ATD and RTI subsystems

ldaa #'V'
jsr outchar
ldaa #'='
jsr outchar
ldaa #'X'
jsr outchar
ldaa #'.'
jsr outchar
ldaa #'X'
jsr outchar
cli ; enable interrupts

contin jsr inchar
jsr outchar
cmpa #ESC
bne contin
swi

;**********************************************************************
;
; ATD and RTI initialization routine
;
; Initialize the ATD to sample a D.C. input voltage (range: 0 to 5V)
; on Channel 7 (to test this hood ch7 to a potentiometer between 5V
; and ground). The ATD should be operated in a program-driven (i.e.,
; non-interrupt driven), normal flag clear mode using nominal sample
; time/clock prescaler values.
;
; Note: Vrh (the ATD reference high voltage) is connected to 5 VDC and
; Vrl (the reference low voltage) is connected to GND on the HC12
; evaluation board.
;
; Also, be sure to initialize the RTI to generate an 8.192 ms interrupt ;
rate
;
init_rti_atd

Problem #2 < place your ATD and RTI initialization code here >
Problem #3 < place your code for the ATD sampling routine here >
Problem #4 < place your code for the voltage display routine here >
include "lab6_io.asm" ; if you run out of space put these ; EEPROM
include "lab6_vec.asm" ; Be sure to set the vector ; for the RTI isr in the EEPROM
end

Problem #5  Fill in the blanks to write an initialization function to configure the channel one capture input to recognize an edge and generate an interrupt.

TIMInit       movb ___,TSCR   ; turn on the TIM
              movb ___,TCTL4   ; capture rising edges
              movb ___,TMSK1   ; enable CH1 interrupt
              movb ___,TFLG1   ; clear CH1 flag
              rts