CpE 390: Microprocessor Systems

Lecture 12

68HC12 Analog to Digital Converter (2)

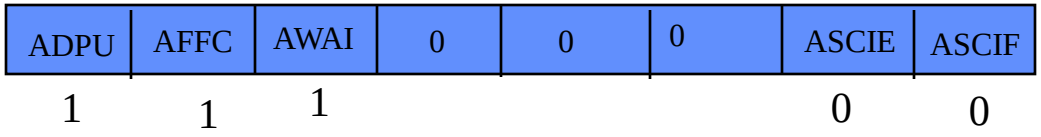
Example 10.5 Write a subroutine to initialize the ATD0 module of the 912DG128 with the following parameters:

- Nonscan mode
- Select channels 0 to 7
- Enable fast flag clear feature
- Stop ATD in wait mode
- Disable interrupt
- Finish current conversion then freeze when BDM becomes active
- Select 10-bit operation and set sample time to 4 ATD clock periods
- Set prescale factor to 4 for 8 MHz E clock

Solution: Settings of ATD control registers as follows:

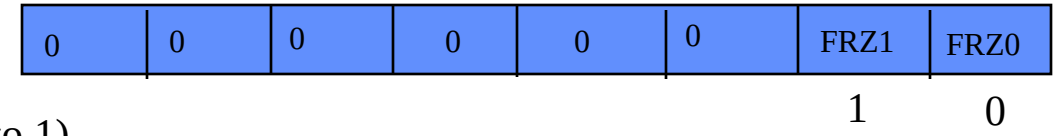
ATD0CTL2

- Enable ATD (set bit 7 to 1)
- Select fast flag clear all (set bit 6 to 1)
- Stop ATD when in wait mode (set bit 5 to 1)
- Disable ATD interrupt (set bit 1 to 0)
- Clear all other bits
- Write the value $E0 into this register



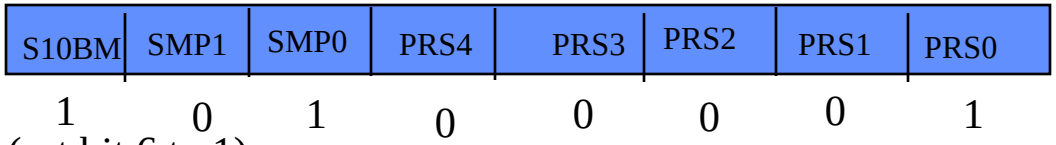
ATD0CTL3

- Complete the current instruction when the BDM becomes active
- Write the value $02 into this register



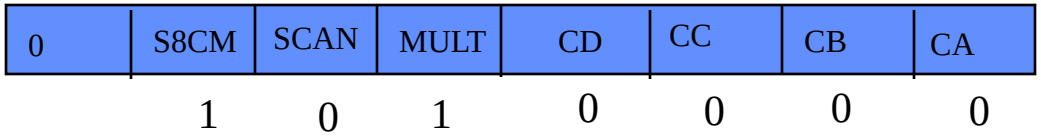
ATD0CTL4

- Select 10-bit operation (set bit 7 to 1)
- Set sample time to 4 ATD clock periods (set bits 6..5 to 01)
- Set prescale factor to 4 to select 2 MHz as the ATD clock source frequency.
- Write the value $A1 into this register.



ATD0CTL5

- Set conversion sequence length to 8 (set bit 6 to 1)
- Select nonscan mode (set bit 5 to 0)
- Select multiple channel mode (set bit 4 to 1)
- Select channels 7 to 0 (set bits 3..0 to 0000)
- Write the value $50 to this control register



S8CM	CD	CC	CB	CA	Channel Signal	Result in ADRx if MULT = 1
0	0	0	0	0	AN0	ADR0
			0	1	AN1	ADR1
			1	0	AN2	ADR2
			1	1	AN3	ADR3
0	0	1	0	0	AN4	ADR0
			0	1	AN5	ADR1
			1	0	AN6	ADR2
			1	1	AN7	ADR3
0	1	0	0	0	Reserved	ADR0
			0	1	Reserved	ADR1
			1	0	Reserved	ADR2
			1	1	Reserved	ADR3
0	1	1	0	0	VRH	ADR0
			0	1	VRL	ADR1
			1	0	$(V_{RH} + V_{RL})/2$	ADR2
			1	1	Test/reserved	ADR3
1	0	0	0	0	AN0	ADR0
			0	1	AN1	ADR1
			1	0	AN2	ADR2
			1	1	AN3	ADR3
		1	0	0	AN4	ADR4
			0	1	AN5	ADR5
			1	0	AN6	ADR6
			1	1	AN7	ADR7
1	1	0	0	0	Reserved	ADR0
			0	1	Reserved	ADR1
			1	0	Reserved	ADR2
			1	1	Reserved	ADR3
		1	0	0	VRH	ADR4
			0	1	VRL	ADR5
			1	0	$(V_{RH} + V_{RL})/2$	ADR6
			1	1	Test/reserved	ADR7

Notes: Shaded bits are “don’t care” if the MULT bit = 1 and the entire block of four or eight channels makes up a conversion sequence. When MULT bit = 0, all four bits (CD, CC, CB, CA) must be specified and a conversion sequence consists of four or eight consecutive conversions of the single specified channel.

Example 10.6 Write a program to perform A/D conversion on the analog signal connected to the AN7 pin. Collect 20 A/D conversion results and store them at memory location starting from $800. Use the same configuration as in Example 10.4.

Solution:

- Write to the ATDCTL5 register five times and collect four samples each time.
- Wait until the SCF flag of the ATDSTAT register is set to 1 and then collect the samples.

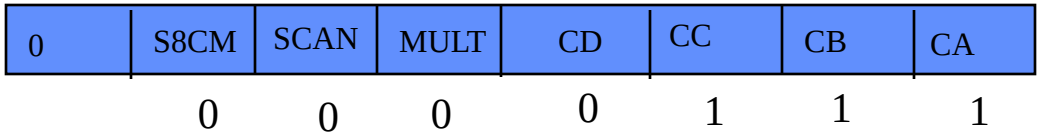
```
#include   "d:\miniide\hc12.inc"

           org      $1000
           ldx      #$800           ; use index register X as a pointer to the buffer
           jsr      ATD_init        ; initialize the ATD converter
           ldy      #5
loop5      movb     #$07,ATDCTL5    ; start an A/D conversion sequence
           brclr    ATDSTAT0,$80,*
           movw     ADR0H,2,x+      ; collect and save the conversion result (left- justified)
           movw     ADR1H,2,x+      ; post-increment the pointer by 2
           movw     ADR2H,2,x+      ;       “
           movw     ADR3H,2,x+      ;       “
           dbne     y,loop5
           swi
           end
```

From example 10.4.

```
#include   "d:\miniide\hc12.inc"
ATD_init   movb      #$E0,ATDCTL2
           ldaa      #10
; the next two instructions create 5 us delay
wait       deca                 ; 1 E clock cycle execution time
           bne       wait       ; 3/1 E clock cycles execution time
           movb      #$02,ATDCTL3
           movb      #$81,ATDCTL4
           rts
```

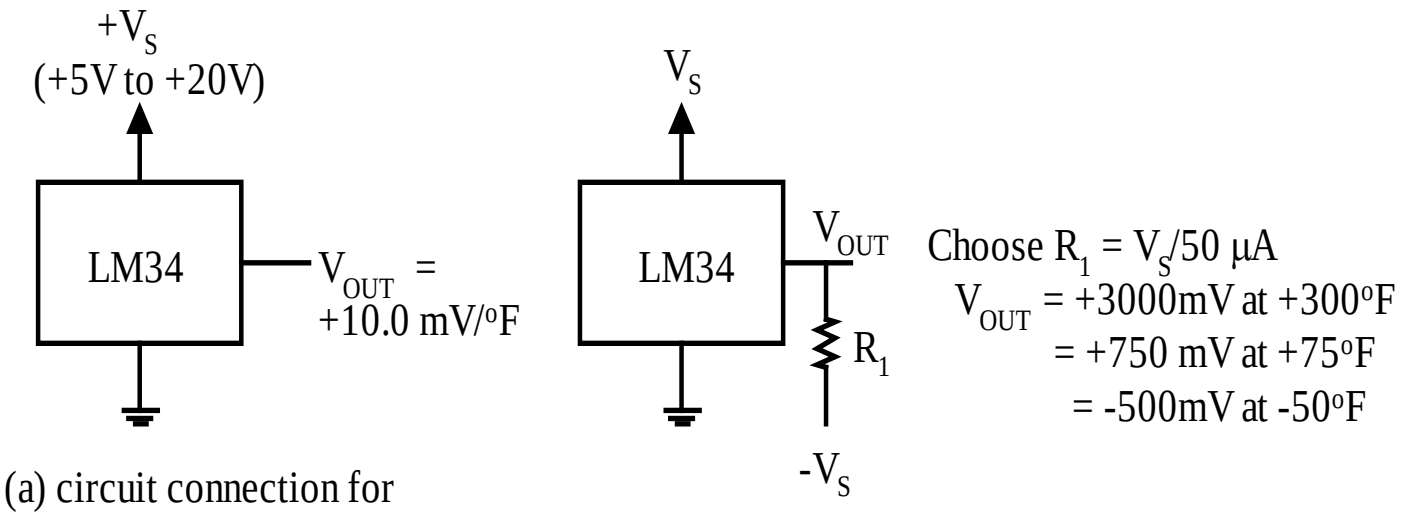
ATDCTL5

Select four conversions as the conversion sequence length (set bit 6 to 0)
Select nonscan mode (set bit 5 to 0)
Select single channel mode (set bit 4 to 0)
Select channel 7 (set bits 3..0 to 0111)
Write the value $07 to this register



The LM34 Precision Fahrenheit Temperature Sensors

- Voltage output is linearly proportional to the ambient temperature.
- No external calibration required.
- Very linear over the temperature range
- Accuracy is ±0.5°F at room temperature and 1.5°F over a full –50° to +300°F range.
- Draws about 75 μA current from the power supply.



(a) circuit connection for +5°F to +300°F

(b) circuit connection for -50°F to +300°F

Figure 10.13 Circuit connection for the LM34

Example 10.7 Use the circuit in Fig 10.13 as a building block in a system to measure the room temperature. Display the result in three integral and one fractional digits using seven segment displays driven by the MAX7221. Measure and display the temperature in the range of – 44 ^{o}F to 212 ^{o}F.

Solution:

- The LM34 voltage output will be from –440mV to 2120mV from – 44 ^{o}F to 212 ^{o}F.
- Use the circuit shown in Figure 10.14 to shift and scale the sensor output to 0~5V.
- The room temperature is equal to A/D result divided by 4 and then subtracted by 44 (since 10 bit resolution of ATD).
- Temperature will be displayed in six seven-segment displays, which will be driven by the MAX7221. The circuit is shown in Figure 10.15.

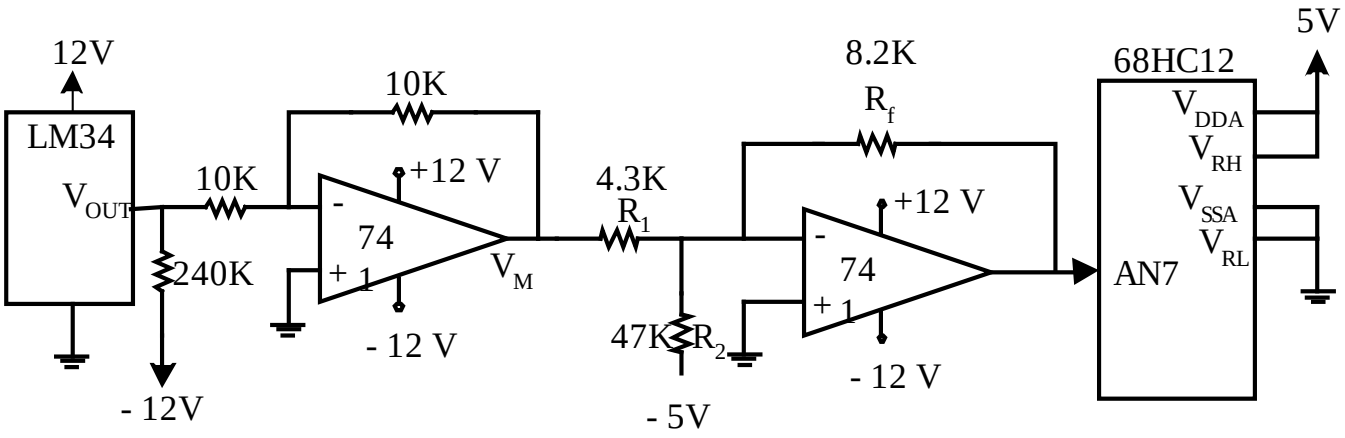


Figure 10.14 Circuit connection between the LM34 and the 68HC12

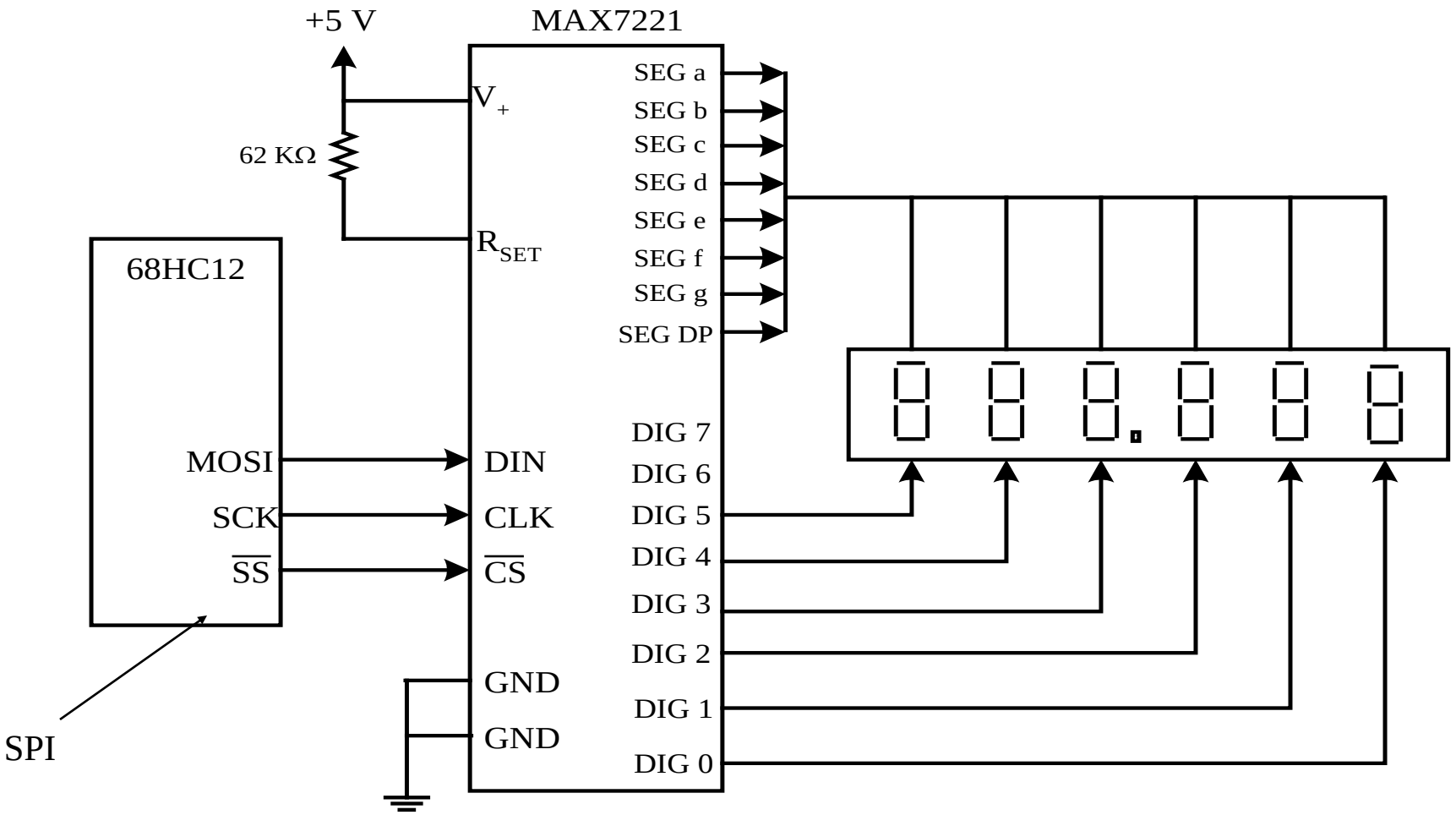


Figure 10.15 Digital thermometer display circuit

Temperature Display Consideration

- Temperature will be measured once every second and so will the update of the display.
- The characters ^{o} and F will be displayed in digits 1 and 0, respectively.
- The minus sign will be displayed when the temperature is below zero. The minus sign will be shown in digit 5.
- When the temperature is between 0^{o}F and -10^{o}F, the negative sign will be displayed in digit 5 and digit 4 will be blanked.
- Leading zeros will be blanked for positive temperature.

SPI Configuration

- Same as Example 9.11.

MAX7221 Configuration

Scan limit register

- Need to display digits 5 to 0.
- Write the value $0B05 to the MAX7221.

Decode mode register

- The leading four digits should be displayed using decode mode whereas ^{o} and F should be displayed using non decode mode.
- The value to be sent to the MAX7221 is $09FC.

Intensity register

- Temperature must be displayed in maximum intensity.
- The value to be sent to the MAX7221 is $0A0F.

Shutdown register

- The chip should operate in normal mode.
- The value to be sent to the MAX7221 is $0C01.

Display test register

- The chip should operate in normal mode.
- The value to be sent to the MAX7221 is $0F00.

Digit 0 register

- Letter F is to be displayed using no-decode mode.
- The value to be sent to this register is $0147.

Digit 1 register

- The degree character is to be displayed using non-decode mode.
- The value to be sent to this register is $0263.

Digit 2 register

- Fractional digit is to be displayed in this digit using decode mode.
- The fraction digit can be from 0 to 9.
- The value to be sent to the MAX7221 is $030x (x = 0..9).

Digit 3 register

- This digit displays the one's digit using the decode mode.
- Decimal point is displayed in this digit.
- The value of this digit can be from 0 to 9.
- The value to be sent to the MAX7221 is $040x (x = 0..9).

Digit 4 register

- This digit displays the ten's digit using decode mode.
- The value of this digit can be from 0 to 9.
- The value to be sent to the MAX7221 is $050x (x = 0..9).
- If the temperature is between 0^o and -10^oF, then this digit should be blanked. Sent the value $050F to blank this digit.

Digit 5 register

- This digit may display hundred's digit or minus sign, or be blanked.
- When displaying hundred's digit, send the value $060x, where, x = 1 or 2.
- When displaying minus sign, send the value $060A.
- When being blanked, send the value $060F.

The configuration of the A/D converter is identical to that in Example 10.4.

Procedure

Step 1

Initialize the SPI module.

Step 2

Initialize the MAX7221 chip.

Step 3

Configure the ATD module.

Step 4

Start A/D conversion and read the conversion result.

Step 5

Divide the conversion result by 4 and then subtract the quotient by 44 to obtain the current temperature. Separate each temperature digit by performing repetitive division by 10.

Step 6

Transfer temperature data to the MAX7221 to update the display.

Step 7

Wait for one second and then go to step 4.

```
#include "d:\miniide\hc12.inc"
                org     $800
scan_limit      db      $0B,$05         ; data to set up scan limit register
decode_mode db          $09,$FC         ; data to set up decode mode register
intensity       db      $0A,$0F         ; data to set up intensity register
shutdown        db      $0C,$01         ; data to set up shut down register
display_test    db      $0F,$00         ; data to set up display test register
digit0          db      $01,$47         ; data to update digit 0
digit1          db      $02,$63         ; data to update digit 1
digit2          rmw     1               ; data to update digit 2
digit3          rmw     1               ; data to update digit 3
digit4          rmw     1               ; data to update digit 4
digit5          rmw     1               ; data to update digit 5
f_is_zero       rmb     1               ; flag to indicate if fractional digit is 0
                org     $1000
                lds     #$8000          ; set up the stack pointer
                jsr     spi_init        ; configure SPI function
                jsr     m7221_init      ; configure MAX7221
                jsr     atd_init        ; configure ATD module
                clr     f_is_zero       ; fractional digit is not zero
forever         jsr     init_disp       ; initialize digits 2 to 5
                movb    #$07,ATDCTL5 ; start an A/D conversion sequence
                brclr   ATDSTAT0,$80,* ; wait until conversion is done
```

```
            ldd     ADR0H       ; read the conversion result
            ldx     #64         ; right-justify the conversion result
            idiv                ;  (D)/(X)=>X, remainder to D
            xgdx                ;  (D) ⇔ (X)
            ldx     #4          ; convert to Fahrenheit temperature to get the fractional digit
            idiv                ;           "
; compute the value for displaying fractional digit to be sent to MAX7221: since only 4
combinations of last two digits, the fractional digit is equally divided into 0, 3, 5, 8 (or 0, 3, 6, 9)
            cmpb    #0
            bne     check_1
            movw    #$0300,digit2  ; fractional digit is 0
            movb    #1,f_is_zero
            bra     ones_digit
check_1     cmpb    #1
            bne     check_2
            movw    #$0303,digit2  ; fractional digit is 3
            bra     ones_digit
check_2     cmpb    #2
            bne     is_3
            movw    #$0305,digit2  ; fractional digit is 5
            bra     ones_digit
is_3        movw    #$0308,digit2  ; fractional digit is 8
ones_digit  xgdx
            subd    #44         ; shift by 44 degree
```

```
            bpl     above_zero      ; is temperature above zero?
            bmi     below_zero
            movb    #$0F,digit4+1   ; temperature is 0, so blank digit 4
            movb    #$0F,digit5+1   ; & blank digit 5
            jmp     update_T
below_zero  movw    #$060A,digit5   ; set up the negative sign
            negb                    ; convert to positive temperature
            tst     f_is_zero
            bne     no_op           ; is fractional digit zero?
            incb                    ; add 1 to negative temperature
            ldaa    #10             ; take 10's complement of the
            suba    digit2+1        ; fractional digit
            staa    digit2+1        ;         "
no_op       clra                    ; clear A
            ldx     #10             ; separate one's digit
            idiv                    ;  (D/X)=>X, remainder =>D
            stab    digit3+1        ; set the digit3 value
            xgdx                    ; place ten's digit in B
            tstb                    ; is ten's digit zero?
            bne     not_zero
            ldab    #$0F
            stab    digit4+1        ; blank ten's digit
            jmp     update_T
```

```
not_zero    stab    digit4+1        ; fix ten's digit's value
            jmp     update_T
above_zero  clra
            cmpb    #9
            bhi     hi_than_9       ; is temperature equal or above ten degrees
            stab    digit3+1        ; between 0 and 9
            movb    #$0F,digit4+1 ; blank digit 4
            movb    #$0F,digit5+1 ; blank digit 5
            jmp     update_T
hi_than_9   cmpb    #99             ; is temperature between 10 and 100?
            bhi     three_dig
            ldx     #10
            idiv
            stab    digit3+1        ; set the one's digit value
            xgdx
            stab    digit4+1        ; set the ten's digit value
            movb    #$0F,digit5+1 ; blank the most significant digit
            jmp     update_T
three_dig   ldx     #10
            idiv
            stab    digit3+1        ; set the one's digit value
            xgdx
            ldx     #10
```

```
            idiv
            stab    digit4+1        ; set the ten's digit value
            xgdx
            stab    digit5+1        ; set the hundred's digit value
update_T    ldx     #digit2
            ldab    #4
            jsr     transfer        ; send data to MAX7221 to update temperature
            jsr     delay_1s
            jmp     forever
; ***************************************************************************
; The following routine initializes the SPI function
; ***************************************************************************
spi_init    movb    #$50,SP0CR1 ; enable SPI, shift data on rising edge, master mode
            movb    #$08,SP0CR2 ; enable internal pullup, normal drive
            movb    #$00,SP0BR  ; shift rate set to 4 MHz
            movb    #$CA,DDRS
            movb    #$07,PURDS  ; all port S pins have pullup
            rts
; ***************************************************************************
; The following routine initializes the MAX7221 chip including degree and letter F
; ***************************************************************************
```

```
m7221_init  ldx     #scan_limit
            ldab    #7
            jsr     transfer
            rts
; ***************************************************************************
; The following routine transfer data to the MAX7221 chip.
; ***************************************************************************
transfer    bclr    PORTS,$80
            ldaa    1,x+
            staa    SP0DR
            brclr   SP0SR,$80,*   ; wait for SPI transfer to complete
            ldaa    1,x+
            staa    SP0DR
            brclr   SP0SR,$80,*
            bset    PORTS,$80     ; transfer data to appropriate register
            dbne    b,transfer
            rts
```

```
; ***************************************************************************
; The following function initializes the ATD module.
; ***************************************************************************
atd_init    movb    #$E0,ATDCTL2
            ldaa    #10
; the next two instructions create 5 us delay
wait        deca
            bne     wait
            movb    #$02,ATDCTL3
            movb    #$81,ATDCTL4
            rts
; ***************************************************************************
; The following function initializes display data.
; ***************************************************************************
init_disp   movw    #$0300,digit2
            movw    #$0480,digit3
            movw    #$0500,digit4
            movw    #$0600,digit5
            rts
```

```
; ***************************************************************************
; The following function creates one-second time delay
; ***************************************************************************
delay_1s    pshx
            movb    #$90,TSCR     ; enable TCNT & fast flags clear
            movb    #$03,TMSK2    ; configure prescale factor to 8
            bset    TIOS,$01      ; enable OC0
            ldx     #20           ; prepare to perform 20 OC actions
            ldd     TCNT
again       addd    #50000        ; start an output compare operation
            std     TC0           ; with 50 ms time delay
            brclr   TFLG1,$01,*
            ldd     TC0
            dbne    x,again
            pulx
            rts
            end
```

Homework #9

- See course website: http://390.revan.us
  - click homework tab
- Please submit a hard copy