Homework #6 (Solution)

1)

- 4 Conversions, S8CM → 0
- Continuous, SCAN → 1
- Multiple Channels, MULT → 1
- CD:CC:CB:CA → 0000
  - Therefore, the ADCTL5 register should contain %0011_0000, or $30
  - The results of the conversions are in ADR0 – ADR3 or $0070 - $0076.

2)

```
RES          EQU $3000

ATDCTL2   EQU $0062
ATDCTL2_IN    EQU $80

ATDCTL5   EQU $0065
ATDCTL5_IN    EQU $30

ATDSTAT2   EQU $0066

ADR0 EQU $0070
ADR1 EQU $0072
ADR2 EQU $0074
ADR3 EQU $0076

   Org $4000
   Lds #$8000 ; Initialize the stack pointer

   Jsr INIT_ATD
   Jsr measure
   SWI

; *********************
; Initialization Function
; *********************
INIT_ATD:
    Movb #ATDCTL2_IN, ATDCTL2  ; Turn on ATD system

     ; Delay for 100 us until ATD turns on
     Ldaa #$C8
Loop:
      Deca
      Bne Loop
```

```
    Movb #ATDCTL5_IN, ATDCTL5  ; Configure ATD
    Rts

; *********************
; Measurement Function
; *********************
Measure:
    Ldx #$000f ; Initialize a counter (start at 0xF, or 15 and count down to 0)
    Ldy #RES    ; Initialize result address

MeasureLoop:
      CPX #$0000
       BEQ measureDone

WaitConv:
     Brclr ATDSTAT2, #$80, WaitConv   ; Wait for conversion to complete

     ; Save results of all conversions
      LDD ADR0
      STD 1,Y+

      LDD ADR1
      STD 1,Y+

       LDD ADR2
      STD 1,Y+

       LDD ADR3
      STD 1,Y+

      ; Decrement counter
       DEX

      Bra measureLoop

MeasureDone:
      Rts
```

3)

```
PORTB:     equ  $01       ; port b data
DDRB:     equ  $03        ; port b direction


ATDCTL2   EQU $0062
```

```
ATDCTL2_IN    EQU $80

ATDCTL5   EQU $0065
ATDCTL5_IN    EQU $30

ATDSTAT2   EQU $0066

ADR1 EQU $0072

    Org $4000
    Lds #$8000 ; Initialize the stack pointer

   ; Setup Port B
    movb  #$FF,DDRB   ; all bits as outputs

    Jsr INIT_ATD
    Jsr measure


     SWI
; *********************
; Initialization Function
; *********************
INIT_ATD:
     Movb #ATDCTL2_IN, ATDCTL2  ; Turn on ATD system

      ; Delay for 100 us until ATD turns on
      Ldaa #$C8
Loop:
       Deca
       Bne Loop

     Movb #ATDCTL5_IN, ATDCTL5  ; Configure ATD
     Rts

; *********************
; Measurement Function
; *********************
Measure:
     Ldx #$0007 ; Initialize a counter (start at 7 and count down to 0)
     Ldd #$0000   ; Initialize total counter

MeasureLoop:
       CPX #$0000
        BEQ measureDone
```

```
WaitConv:
     Brclr ATDSTAT2, #$80, WaitConv   ; Wait for conversion to complete

     ; Add to the total
      ADDD ADR1

      ; Decrement counter
       DEX

      Bra measureLoop

MeasureDone:
      ; Divide total by 8
      LSRD
      LSRD
      LSRD

     ; Send result to port B (lower 8-bits)
      STAB PORTB
      rts
```

4)

```
PORTB:     equ  $01      ; port b data
DDRB:      equ  $03      ; port b direction


ATDCTL2   EQU $0062
ATDCTL2_IN    EQU $80

ATDCTL5   EQU $0065
ATDCTL5_IN    EQU $30

ATDSTAT2   EQU $0066

ADR1 EQU $0072

PORTAD EQU $6F

    Org $4000
    Lds #$8000 ; Initialize the stack pointer

   ; Setup Port B
    movb  #$FF,DDRB   ; all bits as outputs

    Jsr INIT_ATD
```

```
 INF_LOOP:
      Jsr measure
       Bra INF_LOOP


   SWI
; *********************
; Initialization Function
; *********************
INIT_ATD:
   Movb #ATDCTL2_IN, ATDCTL2  ; Turn on ATD system

    ; Delay for 100 us until ATD turns on
    Ldaa #$C8
Loop:
     Deca
     Bne Loop

   Movb #ATDCTL5_IN, ATDCTL5  ; Configure ATD
   Rts

; *********************
; Measurement Function
; *********************
Measure:

WaitConv:
    Brclr ATDSTAT2, #$80, WaitConv   ; Wait for conversion to complete

    ; Grab the ATD value
     LDD ADR3

    ; Check value
    CPD #$00CC
    BLT NEXT_CHECK

    LDAB #$10  ; Set bit 5
    STAB PORTB
    BRA MeasureDone

NEXT_CHECK:

    CPD #$0099
    BLT NEXT_CHECK2
```

```
    LDAB #$08  ; Set bit 4
    STAB PORTB
    BRA MeasureDone

NEXT_CHECK2:

    CPD #$0066
    BLT NEXT_CHECK3

    LDAB #$04  ; Set bit 3
    STAB PORTB
    BRA MeasureDone

NEXT_CHECK3:

    CPD #$0033
    BLT NEXT_CHECK4

    LDAB #$02  ; Set bit 2
    STAB PORTB
    BRA MeasureDone

NEXT_CHECK4
     LDAB #$01  ; Set bit 1
    STAB PORTB

MeasureDone:
   rts
```

5)

- D. Pack →
    - D = $44,
    - . = $2e,
    - (space) = $20,
    - P = $50,
    - a = $61,
    - c = $63,
    - k = $6b

6) D. Pack →

- D = %11000100 = $C4,
- . = %10101110 = $ae,
- (space) = %00100000 = $20,
- P = %11010000 = $d0,
- a = %01100001 = $61,
- c = %11100011 = $e3,

- k = %01101011 = $6b

7) This decodes to $53, $63, $6f, $6f, $62, $79, $2d, $44, $6f, $6f, $21, or Scooby-Doo!

8) 1200 baud → 1200 Hz, or 1/1200 seconds per bit. Each 8-bit character requires a start bit and a stop bit, so each character requires 10 bits to transfer. This means that 11 characters, requires 11*10 = 110 bits, therefore taking 91.7 milliseconds.

If stop/start bits are ignored, then only 11*8, or 88 bits must be transferred which takes 73.3 milliseconds.

9)

- SCI:
  - The SCI system is asynchronous (no synchronized clocks) so it must use a type of synchronization symbol, namely start and stop bits.
- SPI:
  - The SPI system is synchronous (lock step w/ clocks). The master in an SPI system generates a synchronization clock (SCK) that is used by the slave. Essentially, the slave uses data from the master as the clock.