```
; export symbols
            XDEF Entry, _Startup ; export 'Entry' symbol
            ABSENTRY Entry; for absolute assembly: mark this as an application entry point

;Include derivative-specific definitions
        INCLUDE 'derivative.inc'
WAITING   EQU 0
EFLOOP    EQU 1
REVERSE  EQU 2
LL            EQU  3
RR            EQU  4
LEFT        EQU 5
RIGHT       EQU 6
TRN           EQU 7
EFUPPERTHRESHOLD    EQU $D0
EFLOWERTHRESHOLD    EQU $A0
DTHRES     EQU $70
ATHRES     EQU $70
BTHRES    EQU $70
CTHRES    EQU $C0

TwoSec     EQU $46
        ORG $3850
COUNT1    DC.W 0
COUNT2    DC.W 0
DIRECTION  DC.B 0
NO_BLANK    DS.B 1
BCD_SPARE   RMB 2
C1               RMB 2
C2               RMB 2
C_STATE      RMB 1
ASTATE      RMB 1
BSTATE     RMB 1
CSTATE    RMB  1
DSTATE    RMB  1
ESTATE    RMB  1
FSTATEUS   RMB 1
```

```
            ORG $4000
Entry:
_Startup:

                    BSET DDRT,  %11111111
                    BSET DDRA, %11111111
                    JSR INIT
                    JSR openADC
                    JSR openLCD
                   JSR CLR_LCD_BUF
                   JSR initAD
                   LDAA WAITING
                   STAA C_STATE

LOOP        JSR DISPATCHER
            BRA LOOP
DISPATCHER          LDAA C_STATE
                    SUBA #$08
                    BHS RESTART

                    LDAA   C_STATE
                   CMPA   #WAITING
                   BNE    N_WAITING
                   JSR    WAITING_STATE
                   RTS
N_WAITING      LDAA C_ STATE
                     CMPA     #EFLOOP
                      BNE N_FORWARD
                      JSR EF_LOOP
                      RTS
N_FORWARD   LDAA C_STATE
                    CMPA  #REVERSE
                    BNE N_STATE1
                   JSR REV_STATE
                    RTS
N _STATE1     LDAA C_STATE
                   CMPA  #LL
                  BNE N_STATE2
                  JSR LL_STATE
                   RTS
```

```
N_STATE2    LDAA C_STATE
            CMPA  #RR
            BNE N_STATE3
            JSR RR_STATE
             RTS
N_STATE3    LDAA C_ STATE
            CMPA  #LEFT
            BNE N_ STATE4
            JSR CHGSTATELEFT
            RTS
N_STATE4    LDAA C_STATE
            CMPA  #RIGHT
            BNE N_STATE5
            JSR CHGSTATERIGHT
            RTS
N_STATE5   LDAA C_STATE
           CMPA #TRN
           BNE N_STATE6
          JSR TRN _STATE

N_STATE6  JSR CHGSTATE
          SWI
RESTART    LDAA #EFLOOP
          STAA  C_STATE
          BRA DISPATCHER

BEGIN      LDAA SENSOR_BOW
           ADDA  #$20
           STAA ATHRES
           LDAA SENSOR _PORT
           ADDA #$20
          STAA BTHRES
          LDAA SENSOR_MID
          ADDA #$20
          STAA CTHRES
          LDAA SENSOR_STBD
          ADDA #$20
          STAA DTHRES
          LDAA SENSOR_LINE
         ADDA #$20
         STAA EFUPPERTHRES
         LDAA SENSOR_LINE
         SUBA #$30
         STAA EFLOWERTHRES
```

```
                JSR   LOOP
ALL_STOP      JSR PORTOFF
              JSR STAROFF
              SWI
WAITING_STATE  JSR STAROFF
               JSR PORTOFF
               BRCLR PORTAD0,%00001000,CHGSTATE
               RTS
CHGSTATE      LDAA #EFLOOP
              STAA C_STATE
              JSR DISPATCHER
EF_LOOP       JSR PORTON
              JSR STARON
              JSR STARFWD
              JSR PORTFWD
              JSR STATUPDATE
              RTS
STATUPDATE              JSR LOAD
                LDAA #$01
                STAA ASTATE
                LDAA #$00
                STAA DSTATE
                LDAA #$00
                STAA BSTATE
                LDAA #$01
                STAA CSTATE
                JSR ASTATEUS

               JSR DCHECK
               JSR BCHECK

              LDAA SENSOR_LINE
              SUBA #EFUPPERTHRES
              BHI FSTATEE

              LDAA #EFLOWERTHRES
              SUBA SENSOR_LINE
              BHI ESTATEE

              JSR G_LEDS_OFF
              RTS
```

```
DSTATEE   LDAA #$01
         STAA DSTATE
         BRA CSTATEE

CSTATEE   LDAA #$00
          STAA CSTATE
          RTS

BSTATEE   LDAA #$01
          STAA BSTATE
          BRA DSTATEUS

ASTATEE   LDAA #$00
          STAA ASTATE
          BRA BSTATEUS

ESTATEE   LDAA #$01
          STAA ESTATE
          BRA ESETCONFIG

FSTATEE   LDAA #$01
          STAA FSTATEUS
          BRA FSETCONFIG

DSTATEUS  LDAA SENSOR_STBD
           SUBA #DTHRES
           BHI DSTATEE
          BRA CSTATEUS

BSTATEUS  LDAA SENSOR_PORT
            SUBA #BTHRES
            BHI BSTATEE
           BRA DSTATEUS

ASTATEUS  LDAA SENSOR_BOW
           SUBA #ATHRES
           BLO ASTATEE
           BRA BSTATEUS

CSTATEUS   LDAA SENSOR_MID
            SUBA #CTHRES
            BLO  CSTATEE
           RTS
```

```
FSETCONFIG  LDAA #RR
            STAA C_STATE
            JSR DISPATCHER

ESETCONFIG  LDAA #LL
            STAA C_STATE
            JSR DISPATCHER

DCHECK      LDAA DSTATE
            BNE  TRNING
            RTS

BCHECK      LDAA BSTATE
            BNE TRNING
            RTS
TRNING      JSR TRNDECIDER
            RTS

DISZERO     LDAA BSTATE
            BNE CHGSTATELEFT
            RTS
AISZERO     LDAA BSTATE
            BEQ CHGSTATERIGHT
            BRA CHGSTATELEFT

CHGSTATELEFT    JSR STARON
                JSR PORTON
                JSR PORTREV
                JSR STARFWD
                JSR LOAD
                JSR ASTATEUS

                LDAA BSTATE
                BEQ CHGSTATELEFT

                LDAA #EFLOOP
                STAA C_STATE
                RTS
```

```
TRNDECIDER      LDAA ASTATE
                BEQ AISZERO
                LDAA DSTATE
                BEQ DISZERO
                RTS

LL_STATE        JSR PORTOFF
                JSR STARON
                JSR STARFWD
                JSR LOAD
                JSR ASTATEUS
                LDAA ASTATE
                BEQ RR_STATE
                BRCLR PORTAD0, %00000100,CHGSTATES
                LDAA BSTATE
                BNE TRNDECIDER
                LDAA SENSOR_LINE
                SUBA #EFUPPERTHRES
                BLO   CHGSTATELL
                LDAA #RR
                STAA C_ STATE
                RTS


CHGSTATERIGHT     JSR PORTON
                  JSR STARON
                  JSR STARREV
                  JSR PORTFWD
                  JSR LOAD
                  JSR ASTATEUS
                  LDAA ASTATE
                  BEQ LL_STATE
                  LDAA DSTATE
                  BEQ CHGSTATERIGHT

                  LDAA #EFLOOP
                  STAA C_STATE
                  RTS
```

```
RR_STATE      JSR STAROFF
              JSR PORTON
              JSR PORTFWD
              JSR LOAD
              JSR ASTATEUS
              LDAA BSTATE
              BNE TRNDECIDER
              LDAA #EFLOWERTHRES
              SUBA SENSOR _LINE
              BLO CHGSTATERR
              BRCLR  PORTAD0,%00000100,CHGSTATES
              LDAA #LL
              STAA C_ STATE
              RTS

CHGSTATES     LDAA #REVERSE
              STAA C_STATE
              JSR DISPATCHER
              RTS

CHGSTATELL    LDAA #LL
              STAA C_STATE
              JSR LOOP
              RTS

CHGSTATERR    LDAA #RR
              STAA C_STATE
              JSR LOOP
              RTS

TRN_STATE     JSR STARON
              JSR PORTON
              JSR PORTFWD
              JSR STARREV
              JSR LOAD
              LDAA #$01
              STAA CSTATE
              JSR ASTATEUS
              LDAA CSTATE
              BEQ TRN_STATE
              JSR PORTOFF
              JSR STAROFF
              LDAA #LEFT
              STAA C_STATE
```

```
                JSR LOOP

REV_STATE       JSR PORTON
                JSR STARON
                JSR PORTREV
                JSR STARREV

LOAD            LDAA #TRN
                STAA C_STATE
                JSR DISPATCHER
                JSR G_LEDS_ON
                JSR READ_SENSORS
                JSR DISPLAY_SENSORS
                JSR G_LEDS_OFF
                RTS

STARON          LDAA PTT
                ORAA  #%00100000
                STAA PTT
                RTS

PORTON          LDAA PTT
                ORAA #%00010000
                STAA PTT
                RTS

STAROFF         LDAA PTT
                ANDA #%11011111
                STAA PTT
                RTS

PORTOFF         LDAA PTT
                ANDA #%11101111
                STAA PTT
                RTS

STARREV         LDAA PORTA
                ORA #%00000010
                STAA PORTA
                RTS
```

```
STARFWD     LDAA PORTA
            ANDA #%11111101
            STAA PORTA
            RTS

PORTFWD     LDAA PORTA
            ANDA #%11111110
            STAA PORTA
            RTS
PORTREV     LDAA PORTA
            ORAA #%00000001
            STAA PORTA
            RTS
```

-- Now Using the code from the Guider Project Manual :

```
;---------------------------------------------------------------------------
;                'Read Guider' Demo Routine
;
; Reads the eebot guider sensors and displays the values
;  on the Liquid Crystal Display.

; Peter Hiscocks
; Version 2

; Modified from version 1 to support selection of the individual LED
;  associated with a sensor, to reduce crosstalk from unselected sensor
;  LEDs.
; The guider hardware was modified with the addition of a 74HC138 decoder that
;  drives the individual LEDs, so that only the LED associated with a given
;  sensor is ON when that sensor is being read.
; This requires that the software be modified to enable the decoder with bit PA5
;  in PORTA.
; The CdS cells are very slow in responding to changes in light, so a 20
;  millisecond delay is inserted between selecting a particular sensor and
;  reading its value.
; Substantial improvement:
;       Draws less battery current for longer life
;       Creates less heat in the 5V logic regulator
;       Much greater contrast between dark and light readings

; Overview:
; --------
; This program is intended as a test routine for the guider sensors of the
;  eebot robot and contains routines that will be useful in the robot
;  guidance project.
; The guider consists of four absolute brightness sensors and one
;  differential brightness pair of sensors. They are arranged at the nose of
;  the robot in the following pattern (viewed from above):

;                          A
;                       B  C  D
;                         E-F

; The sensors are cadmium sulphide (CdS) photoresistive cells, for which the
;  resistance increases with decreasing light level. The absolute cells
;  A,B,C and D are driven from a constant current source, and the voltage
;  across the cell measured via the HCS12 A/D converter channel AN1. Thus
;  the sensor reading increases as the sensor becomes darker (over a black
;  line, for example).

; The differential sensor E-F is a voltage divider with the two CdS cells E
;  and F separated 0.75 inches, which is the width of electrical tape. It is
;  intended to be used to track the edges of the electrical tape 'line' once
;  the absolute cells have 'found' a black line. Cell E is at the top of the
;  divider, so as the reading from this sensor increases, cell E is becoming
;  lighter, ie, cell E is straying onto the white background.
;  Simultaneously, cell F is becoming darker as it moves over the black
;  tape, and its resistance is increasing, aiding the same effect. The
;  differential action should ignore ambient light.

; The program reads the sensor values, hopefully without disturbing any
;  other settings on the robot. The values are displayed in hexadecimal on
;  the LCD. On the LCD display, the pattern is as described in the routine
;  'DISPLAY_SENSORS'.
```

```
; The 4 absolute sensors should show readings equivalent to approximately 2
;  volts when over a light surface and 4 volts when covered by a finger. The
;  range from light background to black tape background is typically 1.5 volts
;  over a light background to 2.4 volts over black tape.
; We have yet to quantify the readings from the differential sensor E-F.

; Using the program:
; ------------------
; Connect the eebot chassis to an HCS12 computer board as usual. Load
;  'read-guider' profram into the microcomputer. Run the routine 'MAIN'. The
;  display should show the five sensor readings. Placing a finger over
;  one of the sensors to block its illumination should cause the reading to
;  increase significantly. Be extremely careful not to bend the sensors or
;  LED illuminators when doing this.

; equates section

; A/D Converter Equates (all these are done in the 9S12C32.inc file):
;----------------------
; ATDCTL2 EQU $0082 ; A/D Control Register 2
;        7   6   5   4   3   2   1   0
;       --- --- --- --- --- --- --- ---
;      |   |   |   |   |   |   |   |   |
;       --- --- --- --- --- --- --- ---
;        ^   ^   ^   ^   ^   ^   ^   ^
;        |   |   |   |   |   |   |   |
;        |
;        +--------ADPU: 0 = A/D powered down
;                       1 = A/D powered up
;
; ATDCTL3 EQU $0083 ; A/D Control Register 3
;        7   6   5   4   3   2   1   0
;       --- --- --- --- --- --- --- ---
;      |   |   |   |   |   |   |   |   |
;       --- --- --- --- --- --- --- ---
;        ^   ^   ^   ^   ^   ^   ^   ^
;        |   |   |   |   |   |   |   |
;            |   |   |   |
;            +---+---+---+--- Conversion Sequence Limit: 0001 = 1 conversion
;                                                        ...
;                                                        0111 = 7 conversions
;                                                        1xxx = 8 conversions
;
; ATDCTL4 EQU $0084 ; A/D Control Register 4
;        7   6   5   4   3   2   1   0
;       --- --- --- --- --- --- --- ---
;      |   |   |   |   |   |   |   |   |
;       --- --- --- --- --- --- --- ---
;        ^   ^   ^   ^   ^   ^   ^   ^
;        |   |   |   |   |   |   |   |
;        |           +---+---+---+---+-- ATD Clock Prescaler Bits: 00101 = :12
;        |                                                         01011 = :24
;        |                                                         10111 = :48
;        |
;        +--- SRES8: 0 = 10 bits
;                    1 = 8  bits
;
; ATDCTL5 EQU $0085 ; A/D Control Register 5
;        7   6   5   4   3   2   1   0
;       --- --- --- --- --- --- --- ---
;      |   |   |   |   |   |   |   |   |
;       --- --- --- --- --- --- --- ---
;        ^   ^   ^   ^   ^   ^   ^   ^
;        |   |   |   |   |   |   |   |
;        |   |   |   |       +---+---+--- Channel Select
;        |   |   |   |
;        |   |   |   +-- MULT: 0 = Sample one channel
;        |   |   |             1 = Sample several channels starting
;        |   |   |                     with selected channel
;        |   |   |
;        |   |   +-- SCAN: 0 = Single conversion sequence per write to ADTCTL5
;        |   |               1 = Continuous conversion sequences
;        |   |
;        |   +-- not used
;        |
;        +--- DJM: 0 = Left justified data in the result register
;                  1 = Right justified data in the result register
;
```

```
; ATDSTAT0 EQU $0086 ; A/D Status Register 0
;       7   6   5   4   3   2   1   0
;      --- --- --- --- --- --- --- ---
;     |   |   |   |   |   |   |   |   |
;      --- --- --- --- --- --- --- ---
;       ^   ^   ^   ^   ^   ^   ^   ^
;       |   |   |   |   |   |   |   |
;       |
;       +--------SCF: 0 = Conversion sequence not completed
;                     1 = Conversion sequence has completed
;
; The A/D converter automatically puts the 4 results in these registers.
; ATDDR0L  EQU $0091   ; A/D Result Register 0
; ATDDR1L  EQU $0093   ; A/D Result Register 1
; ATDDR2L  EQU $0095   ; A/D Result Register 2
; ATDDR3L  EQU $0097   ; A/D Result Register 3


; PORTA Register
;--------------------------------
; This register selects which sensor is routed to AN1 of the A/D converter
;
; PORTA EQU $0000 ; PORTA Register
;       7   6   5   4   3   2   1   0
;      --- --- --- --- --- --- --- ---
;     |   |   |   |   |   |   |   |   |
;      --- --- --- --- --- --- --- ---
;       ^   ^   ^   ^   ^   ^   ^   ^
;       |   |   |   |   |   |   |   |
;       |   |   |   |   |   |   |   +--- Port Motor Direction (0 = FWD)
;       |   |   |   |   |   |   |
;       |   |   |   |   |   |   +--- Starboard Motor Direction (0 = FWD)
;       |   |   |   |   |   |
;       |   |   |   +---+---+-- Sensor Select
;       |   |   | 000  Sensor Line
;       |   |   | 001  Sensor Bow
;       |   |   | 010  Sensor Port
;       |   |   | 011  Sensor Mid
;       |   |   | 100  Sensor Starboard
;       |   |   |
;       |   |   +-- Sensor LED enable (1 = ON)
;       |   |
;       +---+- not used


; Liquid Crystal Display Equates
;-------------------------------
CLEAR_HOME      EQU $01             ; Clear the display and home the cursor
INTERFACE       EQU $38             ; 8 bit interface, two line display
CURSOR_OFF      EQU $0C             ; Display on, cursor off
SHIFT_OFF       EQU $06             ; Address increments, no character shift
LCD_SEC_LINE    EQU 64              ; Starting addr. of 2nd line of LCD (note decimal value!)

; LCD Addresses
LCD_CNTR        EQU PTJ             ; LCD Control Register: E = PJ7, RS = PJ6
LCD_DAT         EQU PORTB           ; LCD Data Register: D7 = PB7, ... , D0 = PB0
LCD_E           EQU $80             ; LCD E-signal pin
LCD_RS          EQU $40             ; LCD RS-signal pin

; Other codes
NULL            EQU 00              ; The string 'null terminator'
CR              EQU $0D             ; 'Carriage Return' character
SPACE           EQU ' '             ; The 'space' character


; variable/data section

                ORG $3800
;---------------------------------------------------------------------------
; Storage Registers (9S12C32 RAM space: $3800 ... $3FFF)

SENSOR_LINE     FCB $01             ; Storage for guider sensor readings
SENSOR_BOW      FCB $23             ; Initialized to test values
SENSOR_PORT     FCB $45
SENSOR_MID      FCB $67
SENSOR_STBD     FCB $89

SENSOR_NUM      RMB 1               ; The currently selected sensor
```

```
TOP_LINE        RMB 20              ; Top line of display
                FCB NULL            ;  terminated by null

BOT_LINE        RMB 20              ; Bottom line of display
                FCB NULL            ;  terminated by null

CLEAR_LINE      FCC '                    '
                FCB NULL            ;  terminated by null

TEMP            RMB 1               ; Temporary location

; code section

                ORG $4000           ; Start of program text (FLASH memory)
;---------------------------------------------------------------------------
;               Initialization

Entry:
_Startup:
                LDS #$4000          ; Initialize the stack pointer
                CLI                 ; Enable interrupts

                JSR INIT            ; Initialize ports
                JSR openADC         ; Initialize the ATD
                JSR openLCD         ; Initialize the LCD
                JSR CLR_LCD_BUF     ; Write 'space' characters to the LCD buffer

;---------------------------------------------------------------------------
;               Display Sensors

MAIN            JSR G_LEDS_ON       ; Enable the guider LEDs
                JSR READ_SENSORS    ; Read the 5 guider sensors
                JSR G_LEDS_OFF      ; Disable the guider LEDs
                JSR DISPLAY_SENSORS ; and write them to the LCD
                LDY #6000           ; 300 ms delay to avoid
                JSR del_50us        ;  display artifacts
                BRA MAIN            ; Loop forever

; subrotine section

;---------------------------------------------------------------------------
;               Initialize ports

INIT            BCLR DDRAD,$FF      ; Make PORTAD an input (DDRAD @ $0272)
                BSET DDRA,$FF       ; Make PORTA an output (DDRA @ $0002)
                BSET DDRB,$FF       ; Make PORTB an output (DDRB @ $0003)
                BSET DDRJ,$C0       ; Make pins 7,6 of PTJ outputs (DDRJ @ $026A)
                RTS

;---------------------------------------------------------------------------
;               Initialize the ADC

openADC         MOVB #$80,ATDCTL2   ; Turn on ADC (ATDCTL2 @ $0082)
                LDY  #1             ; Wait for 50 us for ADC to be ready
                JSR  del_50us       ;    - " -
                MOVB #$20,ATDCTL3   ; 4 conversions on channel AN1 (ATDCTL3 @ $0083)
                MOVB #$97,ATDCTL4   ; 8-bit resolution, prescaler=48 (ATDCTL4 @ $0084)
                RTS

;---------------------------------------------------------------------------
;               Clear LCD Buffer

; This routine writes 'space' characters (ascii 20) into the LCD display
;  buffer in order to prepare it for the building of a new display buffer.
; This needs only to be done once at the start of the program. Thereafter the
;  display routine should maintain the buffer properly.

CLR_LCD_BUF     LDX #CLEAR_LINE
                LDY #TOP_LINE
                JSR STRCPY

CLB_SECOND      LDX #CLEAR_LINE
                LDY #BOT_LINE
                JSR STRCPY

CLB_EXIT        RTS
```

```
;---------------------------------------------------------------------------
;               String Copy

; Copies a null-terminated string (including the null) from one location to
; another

; Passed: X contains starting address of null-terminated string
;   Y contains first address of destination

STRCPY          PSHX                ; Protect the registers used
                PSHY
                PSHA
STRCPY_LOOP     LDAA 0,X            ; Get a source character
                STAA 0,Y            ; Copy it to the destination
                BEQ  STRCPY_EXIT    ; If it was the null, then exit
                INX                 ; Else increment the pointers
                INY
                BRA  STRCPY_LOOP    ; and do it again
STRCPY_EXIT     PULA                ; Restore the registers
                PULY
                PULX
                RTS

;---------------------------------------------------------------------------
;               Guider LEDs ON

; This routine enables the guider LEDs so that readings of the sensor
;  correspond to the 'illuminated' situation.

; Passed:  Nothing
; Returns: Nothing
; Side:    PORTA bit 5 is changed

G_LEDS_ON       BSET PORTA,%00100000 ; Set bit 5
                RTS

;
;               Guider LEDs OFF

; This routine disables the guider LEDs. Readings of the sensor
;  correspond to the 'ambient lighting' situation.

; Passed:  Nothing
; Returns: Nothing
; Side:    PORTA bit 5 is changed

G_LEDS_OFF      BCLR PORTA,%00100000 ; Clear bit 5
                RTS

;---------------------------------------------------------------------------
;               Read Sensors
;
; This routine reads the eebot guider sensors and puts the results in RAM
;  registers.

; Note: Do not confuse the analog multiplexer on the Guider board with the
;  multiplexer in the HCS12. The guider board mux must be set to the
;  appropriate channel using the SELECT_SENSOR routine. The HCS12 always
;  reads the selected sensor on the HCS12 A/D channel AN1.

; The A/D conversion mode used in this routine is to read the A/D channel
;  AN1 four times into HCS12 data registers ATDDR0,1,2,3. The only result
;  used in this routine is the value from AN1, read from ATDDR0. However,
;  other routines may wish to use the results in ATDDR1, 2 and 3.
; Consequently, Scan=0, Mult=0 and Channel=001 for the ATDCTL5 control word.

; Passed:       None
; Returns:      Sensor readings in:
;               SENSOR_LINE (0) (Sensor E/F)
;               SENSOR_BOW  (1) (Sensor A)
;               SENSOR_PORT (2) (Sensor B)
;               SENSOR_MID  (3) (Sensor C)
;               SENSOR_STBD (4) (Sensor D)
; Note:
;   The sensor number is shown in brackets
;
; Algorithm:
;       Initialize the sensor number to 0
```

```
;        Initialize a pointer into the RAM at the start of the Sensor Array storage
; Loop   Store %10000001 to the ATDCTL5 (to select AN1 and start a conversion)
;        Repeat
;           Read ATDSTAT0
;        Until Bit SCF of ATDSTAT0 == 1 (at which time the conversion is complete)
;        Store the contents of ATDDR0L at the pointer
;        If the pointer is at the last entry in Sensor Array, then
;           Exit
;        Else
;           Increment the sensor number
;           Increment the pointer
;        Loop again.

READ_SENSORS    CLR  SENSOR_NUM      ; Select sensor number 0
                LDX  #SENSOR_LINE    ; Point at the start of the sensor array

RS_MAIN_LOOP    LDAA SENSOR_NUM      ; Select the correct sensor input
                JSR  SELECT_SENSOR   ;  on the hardware
                LDY  #400            ; 20 ms delay to allow the
                JSR  del_50us        ;  sensor to stabilize

                LDAA #%10000001      ; Start A/D conversion on AN1
                STAA ATDCTL5
                BRCLR ATDSTAT0,$80,* ; Repeat until A/D signals done

                LDAA ATDDR0L         ; A/D conversion is complete in ATDDR0L
                STAA 0,X             ;  so copy it to the sensor register
                CPX  #SENSOR_STBD    ; If this is the last reading
                BEQ  RS_EXIT         ; Then exit

                INC  SENSOR_NUM      ; Else, increment the sensor number
                INX                  ;  and the pointer into the sensor array
                BRA  RS_MAIN_LOOP    ;  and do it again

RS_EXIT         RTS

;---------------------------------------------------------------------------
;               Select Sensor

; This routine selects the sensor number passed in ACCA. The motor direction
;  bits 0, 1, the guider sensor select bit 5 and the unused bits 6,7 in the
;  same machine register PORTA are not affected.
; Bits PA2,PA3,PA4 are connected to a 74HC4051 analog mux on the guider board,
;  which selects the guider sensor to be connected to AN1.

; Passed: Sensor Number in ACCA
; Returns: Nothing
; Side Effects: ACCA is changed

; Algorithm:
; First, copy the contents of PORTA into a temporary location TEMP and clear
;        the sensor bits 2,3,4 in the TEMP to zeros by ANDing it with the mask
;        11100011. The zeros in the mask clear the corresponding bits in the
;        TEMP. The 1's have no effect.
; Next, move the sensor selection number left two positions to align it
;        with the correct bit positions for sensor selection.
; Clear all the bits around the (shifted) sensor number by ANDing it with
;  the mask 00011100. The zeros in the mask clear everything except
;        the sensor number.
; Now we can combine the sensor number with the TEMP using logical OR.
;  The effect is that only bits 2,3,4 are changed in the TEMP, and these
;  bits now correspond to the sensor number.
; Finally, save the TEMP to the hardware.

SELECT_SENSOR   PSHA                 ; Save the sensor number for the moment

                LDAA PORTA           ; Clear the sensor selection bits to zeros
                ANDA #%11100011      ;
                STAA TEMP            ; and save it into TEMP

                PULA                 ; Get the sensor number
                ASLA                 ; Shift the selection number left, twice
                ASLA                 ;
                ANDA #%00011100      ; Clear irrelevant bit positions

                ORAA TEMP            ; OR it into the sensor bit positions
                STAA PORTA           ; Update the hardware
                RTS
```

```
;-------------------------------------------------------------------------
;               Display Sensor Readings

; Passed: Sensor values in RAM locations SENSOR_LINE through SENSOR_STBD.
; Returns: Nothing
; Side: Everything

; This routine writes the sensor values to the LCD. It uses the 'shadow buffer' approach.
;  The display buffer is built by the display controller routine and then copied in its
;  entirety to the actual LCD display. Although simpler approaches will work in this
;  application, we take that approach to make the code more re-useable.
; It's important that the display controller not write over other information on the
;  LCD, so writing the LCD has to be centralized with a controller routine like this one.
; In a more complex program with additional things to display on the LCD, this routine
;  would be extended to read other variables and place them on the LCD. It might even
;  read some 'display select' variable to determine what should be on the LCD.

; For the purposes of this routine, we'll put the sensor values on the LCD
;  in such a way that they (sort of) mimic the position of the sensors, so
;  the display looks like this:
;   01234567890123456789
;   ___FF_______________
;   PP_MM_SS_LL_________

; Where FF is the front sensor, PP is port, MM is mid, SS is starboard and
;  LL is the line sensor.

; The corresponding addresses in the LCD buffer are defined in the following
;  equates (In all cases, the display position is the MSDigit).

DP_FRONT_SENSOR EQU TOP_LINE+3
DP_PORT_SENSOR  EQU BOT_LINE+0
DP_MID_SENSOR   EQU BOT_LINE+3
DP_STBD_SENSOR  EQU BOT_LINE+6
DP_LINE_SENSOR  EQU BOT_LINE+9

DISPLAY_SENSORS LDAA SENSOR_BOW       ; Get the FRONT sensor value
                JSR  BIN2ASC          ; Convert to ascii string in D
                LDX  #DP_FRONT_SENSOR ; Point to the LCD buffer position
                STD  0,X              ;  and write the 2 ascii digits there

                LDAA SENSOR_PORT      ; Repeat for the PORT value
                JSR  BIN2ASC
                LDX  #DP_PORT_SENSOR
                STD  0,X

                LDAA SENSOR_MID       ; Repeat for the MID value
                JSR  BIN2ASC
                LDX  #DP_MID_SENSOR
                STD  0,X

                LDAA SENSOR_STBD      ; Repeat for the STARBOARD value
                JSR  BIN2ASC
                LDX  #DP_STBD_SENSOR
                STD  0,X

                LDAA SENSOR_LINE      ; Repeat for the LINE value
                JSR  BIN2ASC
                LDX  #DP_LINE_SENSOR
                STD  0,X

                LDAA #CLEAR_HOME      ; Clear the display and home the cursor
                JSR  cmd2LCD          ;       "

                LDY  #40              ; Wait 2 ms until "clear display" command is complete
                JSR  del_50us

                LDX  #TOP_LINE        ; Now copy the buffer top line to the LCD
                JSR  putsLCD

                LDAA #LCD_SEC_LINE    ; Position the LCD cursor on the second line
                JSR  LCD_POS_CRSR

                LDX  #BOT_LINE        ; Copy the buffer bottom line to the LCD
                JSR  putsLCD
                RTS
```

```
;-------------------------------------------------------------------------
;               Binary to ASCII

; Converts an 8 bit binary value in ACCA to the equivalent ASCII character 2
;  character string in accumulator D
; Uses a table-driven method rather than various tricks.

; Passed: Binary value in ACCA
; Returns: ASCII Character string in D
; Side Fx: ACCB is destroyed

HEX_TABLE       FCC '0123456789ABCDEF' ; Table for converting values

BIN2ASC         PSHA                   ; Save a copy of the input number on the stack
                TAB                    ;  and copy it into ACCB
                ANDB #%00001111        ; Strip off the upper nibble of ACCB
                CLRA                   ; D now contains 000n where n is the LSnibble
                ADDD #HEX_TABLE        ; Set up for indexed load
                XGDX
                LDAA 0,X               ; Get the LSnibble character

                PULB                   ; Retrieve the input number into ACCB
                PSHA                   ; and push the LSnibble character in its place
                RORB                   ; Move the upper nibble of the input number
                RORB                   ;  into the lower nibble position.
                RORB
                RORB
                ANDB #%00001111        ; Strip off the upper nibble
                CLRA                   ; D now contains 000n where n is the MSnibble
                ADDD #HEX_TABLE        ; Set up for indexed load
                XGDX
                LDAA 0,X               ; Get the MSnibble character into ACCA
                PULB                   ; Retrieve the LSnibble character into ACCB
                RTS

;-------------------------------------------------------------------------
;     Routines to control the Liquid Crystal Display

;-------------------------------------------------------------------------
;               Initialize the LCD

openLCD         LDY  #2000             ; Wait 100 ms for LCD to be ready
                JSR  del_50us          ;       "
                LDAA #INTERFACE        ; Set 8-bit data, 2-line display, 5x8 font
                JSR  cmd2LCD           ;       "
                LDAA #CURSOR_OFF       ; Display on, cursor off, blinking off
                JSR  cmd2LCD           ;       "
                LDAA #SHIFT_OFF        ; Move cursor right (address increments, no char. shift)
                JSR  cmd2LCD           ;       "
                LDAA #CLEAR_HOME       ; Clear the display and home the cursor
                JSR  cmd2LCD           ;       "
                LDY  #40               ; Wait 2 ms until "clear display" command is complete
                JSR  del_50us          ;       "
                RTS

;-------------------------------------------------------------------------
;               Send a command in accumulator A to the LCD

cmd2LCD         BCLR LCD_CNTR,LCD_RS   ; Select the LCD Instruction register
                JSR  dataMov           ; Send data to IR or DR of the LCD
                RTS

;-------------------------------------------------------------------------
;               Send a character in accumulator in A to LCD

putcLCD         BSET LCD_CNTR,LCD_RS   ; select the LCD Data register
                JSR  dataMov           ; send data to IR or DR of the LCD
                RTS

;-------------------------------------------------------------------------
;               Send a NULL-terminated string pointed to by X

putsLCD         LDAA 1,X+              ; get one character from the string
                BEQ  donePS            ; reach NULL character?
                JSR  putcLCD
                BRA  putsLCD
donePS          RTS
```

```
;--------------------------------------------------------------------------
;               Send data to the LCD IR or DR depending on the RS signal

dataMov         BSET LCD_CNTR,LCD_E    ; pull the LCD E-sigal high
                STAA LCD_DAT           ; send the 8 bits of data to LCD
                NOP
                NOP
                NOP
                BCLR LCD_CNTR,LCD_E    ; pull the E signal low to complete the write operation

                LDY  #1                ; adding this delay will complete the internal
                JSR  del_50us          ; operation for most instructions
                RTS

;--------------------------------------------------------------------------
;               Position the Cursor

; This routine positions the display cursor in preparation for the writing
;  of a character or string.
; For a 20x2 display:
; The first line of the display runs from 0 .. 19.
; The second line runs from 64 .. 83.

; The control instruction to position the cursor has the format
;            1aaaaaaa
; where aaaaaaa is a 7 bit address.

; Passed:   7 bit cursor Address in ACCA
; Returns:  Nothing
; Side Effects: None

LCD_POS_CRSR    ORAA #%10000000        ; Set the high bit of the control word
                JSR  cmd2LCD           ;  and set the cursor address
                RTS

;--------------------------------------------------------------------------
;               50 Microsecond Delay

del_50us        PSHX                   ; (2 E-clk) Protect the X register
eloop           LDX #300               ; (2 E-clk) Initialize the inner loop counter
iloop           NOP                    ; (1 E-clk) No operation
                DBNE  X,iloop          ; (3 E-clk) If the inner cntr not 0, loop again
                DBNE  Y,eloop          ; (3 E-clk) If the outer cntr not 0, loop again
                PULX                   ; (3 E-clk) Restore the X register
                RTS                    ; (5 E-clk) Else return

;--------------------------------------------------------------------------
;               Interrupt Vectors

                ORG   $FFFE
                DC.W  Entry            ; Reset Vector
```