```
;*****************************************************************
;* This stationery serves as the framework for a                 *
;* user application (single file, absolute assembly application) *
;* For a more comprehensive program that                         *
;* demonstrates the more advanced functionality of this          *
;* processor, please see the demonstration applications          *
;* located in the examples subdirectory of the                   *
;* Freescale CodeWarrior for the HC12 Program directory          *
;*****************************************************************

; export symbols

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


   ; Include derivative-specific definitions
   INCLUDE "derivative.inc"

 ; Insert here your data definition.
;*****************************************************************


;Liquid Crystal Display Equates
CLEAR_HOME    EQU   $01
INTERFACE     EQU   $38
CURSOR_OFF    EQU   $0C
SHIFT_OFF     EQU   $06
LCD_SEC_LINE  EQU   64
LCD_CNTR      EQU   PTJ
LCD_DAT       EQU   PORTB
LCD_E         EQU   $80
LCD_RS        EQU   $40
NULL          EQU   00
CR            EQU   $0D
SPACE         EQU   ' '
T_LEFT        EQU   8
T_RIGHT       EQU   8
START         EQU   0
FWD           EQU   1
ALL_STOP      EQU   2
LEFT_TRN      EQU   3
RIGHT_TRN     EQU   4
REV_TRN       EQU   5
```

```
LEFT_ALIGN    EQU   6
RIGHT_ALIGN   EQU   7
; variable section
;***************************************************

          ORG   $3800
BASELINE     FCB   $9D
BASEBOW      FCB   $CA
BASEMID      FCB   $CA
BASEPORT     FCB   $CC
BASESTAR     FCB   $CC
LINE_VAR     FCB   $18
BOW_VAR      FCB   $30
PORT_VAR     FCB   $20
MID_VAR      FCB   $20
STARBD_VAR   FCB   $15
; Storage Registers (9S12C32 RAM space: $3800 ... $3FFF)
SENSOR_LINE   FCB   $01
SENSOR_BOW    FCB   $23
SENSOR_PORT   FCB   $45
SENSOR_MID    FCB   $67
SENSOR_STBD   FCB   $89
SENSOR_NUM    RMB   1

TOP_LINE      RMB   20
BOT_LINE      RMB   20
          FCB   NULL

CLEAR_LINE    FCC   '              '
          FCB   NULL

TEMP         RMB   1

          ORG   $3850
TOF_COUNTER   dc.b  0
CRNT_STATE    dc.b  2
T_TURN        ds.b  1
TEN_THOUS     ds.b  1
THOUSANDS     ds.b  1
HUNDREDS      ds.b  1
TENS          ds.b  1
UNITS         ds.b  1
NO_BLANK      ds.b  1
HEX_TABLE     FCC   '0123456789ABCDEF'
```

```
BCD_SPARE     RMB   2

; code section
;*****************************************************************

            ORG   $4000
Entry:
_Startup:

            LDS   #$4000
            CLI

            JSR   INIT
            JSR   openADC
            JSR   initLCD
            JSR   CLR_LCD_BUF
            ;; for the guider ^^^
            BSET  DDRA,%00000011
            BSET  DDRT,%00110000
            JSR   initAD
            JSR   initLCD
            JSR   clrLCD
            LDX   #msg1
            JSR   putsLCD
            LDAA  #$C0
            JSR   cmd2LCD
            LDX   #msg2
            JSR   putsLCD
            JSR   ENABLE_TOF
            ; here for the actual bot (LCD and inital State for movement)

MAIN        ; for guider
            JSR   G_LEDS_ON
            JSR   READ_SENSORS
            JSR   G_LEDS_OFF
            JSR   UPDT_DISPL ; write to lcd (guider)
            LDAA  CRNT_STATE  ; for bot now
            JSR   DISPATCHER
            BRA   MAIN

; The LCD display
msg1 dc.b "Battery volt ",0
msg2 dc.b "State   ",0
tab   dc.b "START  ",0
```

```
    dc.b "FWD    ",0
    dc.b "REV    ",0
    dc.b "ALL_STP",0
    dc.b "FWD_TRN",0
    dc.b "REV_TRN",0
    dc.b "LTIMED ",0          ; make sure the "," on the same line or wont display correctly
    dc.b "RTIMED ",0
; Subroutine section

DISPATCHER       JSR   VERIFY_START    ; like lab5 but with change, must make sure to use
the right state at the right time
            RTS

VERIFY_START     CMPA  #START
            BNE   VERIFY_FORWARD
            JSR   START_ST
            RTS

VERIFY_FORWARD   CMPA  #FWD
            BNE   VERIFY_STOP
            JSR   FWD_ST
            RTS

VERIFY_REV_TRN   CMPA  #REV_TRN
            BNE   VERIFY_L_ALIGNMENT
            JSR   REV_TRN_ST
            RTS

VERIFY_STOP      CMPA  #ALL_STOP
            BNE   VERIFY_LEFTTURN
            JSR   ALL_STP_ST
            RTS

VERIFY_LEFTTURN   CMPA  #LEFT_TRN
            BNE   VERIFY_RIGHTTURN
            JSR   LEFT
            RTS

VERIFY_RIGHTTURN  CMPA  #RIGHT_TRN
            BNE   VERIFY_REV_TRN
            JSR   RIGHT

VERIFY_L_ALIGNMENT CMPA  #LEFT_ALIGN
             BNE   VERIFY_R_ALIGNMENT
```

```
            JSR   LEFT_ALIGNMENT_DONE
            RTS

VERIFY_R_ALIGNMENT CMPA  #RIGHT_ALIGN
            JSR   RIGHT_ALIGNMENT_DONE
            RTS

;*******************************************************************************************************
START_ST        BRCLR   PORTAD0, %00000100,START_EXIT
            JSR     INIT_FWD
            MOVB    #FWD, CRNT_STATE

START_EXIT      RTS

;*******************************************************************************************************

FWD_ST          BRSET   PORTAD0, $04, NO_FWD_BUMP
            MOVB    #REV_TRN, CRNT_STATE
            JSR     UPDT_DISPL
            JSR     INIT_REV
            LDY     #6000
            JSR     del_50us
            JSR     INIT_RIGHT
            LDY     #6000
            JSR     del_50us
            LBRA    EXIT

NO_FWD_BUMP     BRSET   PORTAD0, $04, NO_REAR_BUMP
            MOVB    #ALL_STOP, CRNT_STATE
            JSR     INIT_STOP
            LBRA    EXIT

NO_REAR_BUMP    LDAA    SENSOR_BOW
            ADDA    BOW_VAR
            CMPA    BASEBOW
            BPL     NOT_ALIGNED
            LDAA    SENSOR_MID
            ADDA    MID_VAR
            CMPA    BASEMID
            BPL     NOT_ALIGNED
            LDAA    SENSOR_LINE
            ADDA    LINE_VAR
            CMPA    BASELINE
            BPL     CHECK_R_ALIGNMENT
```

```
            LDAA    SENSOR_LINE
            SUBA    LINE_VAR
            CMPA    BASELINE
            BMI     CHECK_L_ALIGNMENT

;*******************************************************************************************************
; for the sensor to act when bot is out the line
NOT_ALIGNED     LDAA    SENSOR_PORT
            ADDA    PORT_VAR
            CMPA    BASEPORT
            BPL     PARTIAL_L_TURN
            BMI     NO_PORT

NO_PORT         LDAA    SENSOR_BOW
            ADDA    BOW_VAR
            CMPA    BASEBOW
            BPL     EXIT
            BMI     NO_BOW

NO_BOW          LDAA    SENSOR_STBD
            ADDA    STARBD_VAR
            CMPA    BASESTAR
            BPL     PARTIAL_R_TURN
            BMI     EXIT

;*******************************************************************************************************

PARTIAL_L_TURN    LDY     #6000
            JSR     del_50us
            JSR     INIT_LEFT
            MOVB    #LEFT_TRN, CRNT_STATE
            LDY     #6000
            JSR     del_50us
            BRA     EXIT

CHECK_L_ALIGNMENT  JSR     INIT_LEFT
             MOVB    #LEFT_ALIGN, CRNT_STATE
             BRA     EXIT

;*******************************************************************************************************

PARTIAL_R_TURN    LDY     #6000
            JSR     del_50us
            JSR     INIT_RIGHT
```

```
            MOVB    #RIGHT_TRN, CRNT_STATE
            LDY     #6000
            JSR     del_50us
            BRA     EXIT

CHECK_R_ALIGNMENT JSR     INIT_RIGHT
            MOVB    #RIGHT_ALIGN, CRNT_STATE
            BRA     EXIT

EXIT          RTS

;*******************************************************************************************************

LEFT          LDAA    SENSOR_BOW
            ADDA    BOW_VAR
            CMPA    BASEBOW
            BPL     LEFT_ALIGNMENT_DONE
            BMI     EXIT

LEFT_ALIGNMENT_DONE   MOVB    #FWD, CRNT_STATE
              JSR     INIT_FWD
              BRA     EXIT

RIGHT         LDAA    SENSOR_BOW
            ADDA    BOW_VAR
            CMPA    BASEBOW
            BPL     RIGHT_ALIGNMENT_DONE
            BMI     EXIT

RIGHT_ALIGNMENT_DONE  MOVB    #FWD, CRNT_STATE
              JSR     INIT_FWD
              BRA     EXIT

;*******************************************************************************************************

REV_TRN_ST      LDAA    SENSOR_BOW
            ADDA    BOW_VAR
            CMPA    BASEBOW
            BMI     EXIT
            JSR     INIT_LEFT
            MOVB    #FWD, CRNT_STATE
            JSR     INIT_FWD
            BRA     EXIT
```

```
ALL_STP_ST      BRSET   PORTAD0, %00000100, NO_START_BUMP
                MOVB    #START, CRNT_STATE

NO_START_BUMP     RTS

; Initialization Subroutines
;*******************************************************************************************************
INIT_RIGHT      BSET    PORTA,%00000010
                BCLR    PORTA,%00000001
                LDAA    TOF_COUNTER
                ADDA    #T_RIGHT
                STAA    T_TURN
                RTS

INIT_LEFT       BSET    PORTA,%00000001
                BCLR    PORTA,%00000010
                LDAA    TOF_COUNTER
                ADDA    #T_LEFT
                STAA    T_TURN
                RTS

INIT_FWD        BCLR    PORTA, %00000011
                BSET    PTT, %00110000
                RTS

INIT_REV        BSET PORTA,%00000011
                BSET PTT,%00110000
                RTS

INIT_STOP       BCLR    PTT, %00110000
                RTS

;*******************************************************************************************************
;       Initialize ports
INIT  BCLR   DDRAD,$FF
      BSET   DDRA,$FF
      BSET   DDRB,$FF
      BSET   DDRJ,$C0
      RTS


;*******************************************************************************************************
;       Initialize ADC
```

```
openADC MOVB   #$80,ATDCTL2
      LDY    #1
      JSR    del_50us
      MOVB   #$20,ATDCTL3
      MOVB   #$97,ATDCTL4
      RTS

;*******************************************************************************************************
;    Clear LCD Buffer

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

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                                    |

G_LEDS_ON        BSET PORTA,%00100000 ; Set bit 5                                    |
             RTS                                                  ;|
```

```
;*******************************************************************************************
;   Guider LEDs OFF                                 |

G_LEDS_OFF      BCLR PORTA,%00100000 ; Clear bit 5                              |
                RTS                                                             ;|

;*******************************************************************************************
;   Read Sensors

READ_SENSORS    CLR   SENSOR_NUM
                LDX   #SENSOR_LINE

RS_MAIN_LOOP    LDAA  SENSOR_NUM
                JSR   SELECT_SENSOR
                LDY   #400
                JSR   del_50us
                LDAA  #%10000001
                STAA  ATDCTL5
                BRCLR ATDSTAT0,$80,*
                LDAA  ATDDR0L
                STAA  0,X
                CPX   #SENSOR_STBD
                BEQ   RS_EXIT
                INC   SENSOR_NUM
                INX
                BRA   RS_MAIN_LOOP

RS_EXIT         RTS


;*******************************************************************************************
;   Select Sensors
SELECT_SENSOR   PSHA
                LDAA PORTA
                ANDA #%11100011
                STAA TEMP
                PULA
                ASLA
                ASLA
                ANDA #%00011100
                ORAA TEMP
                STAA PORTA
                RTS
```

```
;*******************************************************************************************************
;   Display Sensors

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
            JSR   BIN2ASC
            LDX   #DP_FRONT_SENSOR
            STD   0,X
            LDAA  SENSOR_PORT
            JSR   BIN2ASC
            LDX   #DP_PORT_SENSOR
            STD   0,X
            LDAA  SENSOR_MID
            JSR   BIN2ASC
            LDX   #DP_MID_SENSOR
            STD   0,X
            LDAA  SENSOR_STBD
            JSR   BIN2ASC
            LDX   #DP_STBD_SENSOR
            STD   0,X
            LDAA  SENSOR_LINE
            JSR   BIN2ASC
            LDX   #DP_LINE_SENSOR
            STD   0,X
            LDAA  #CLEAR_HOME
            JSR   cmd2LCD
            LDY   #40
            JSR   del_50us
            LDX   #TOP_LINE
            JSR   putsLCD
            LDAA  #LCD_SEC_LINE
            JSR   LCD_POS_CRSR
            LDX   #BOT_LINE
            JSR   putsLCD
            RTS

;*******************************************************************
```

```
;* Update Display (Battery Voltage + Current State) *
;*******************************************************************
UPDT_DISPL      MOVB    #$90,ATDCTL5
                BRCLR   ATDSTAT0,$80,*
                LDAA    ATDDR0L
                LDAB    #39
                MUL
                ADDD    #600
                JSR     int2BCD
                JSR     BCD2ASC
                LDAA    #$8D
                JSR     cmd2LCD
                LDAA    TEN_THOUS
                JSR     putcLCD
                LDAA    THOUSANDS
                JSR     putcLCD
                LDAA    #'.'
                JSR     putcLCD
                LDAA    HUNDREDS
                JSR     putcLCD
                LDAA    #$C7
                JSR     cmd2LCD
                LDAB    CRNT_STATE
                LSLB
                LSLB
                LSLB
                LDX     #tab
                ABX
                JSR     putsLCD
                RTS

;*****************************************************************************************************
ENABLE_TOF      LDAA    #%10000000
                STAA    TSCR1
                STAA    TFLG2
                LDAA    #%10000100
                STAA    TSCR2
                RTS

TOF_ISR         INC     TOF_COUNTER
                LDAA    #%10000000
                STAA    TFLG2
                RTI
; utility subroutines
```

```
;*******************************************************************************************************
initLCD:        BSET    DDRB,%11111111
                BSET    DDRJ,%11000000
                LDY     #2000
                JSR     del_50us
                LDAA    #$28
                JSR     cmd2LCD
                LDAA    #$0C
                JSR     cmd2LCD
                LDAA    #$06
                JSR     cmd2LCD
                RTS
;*******************************************************************************************************
clrLCD:         LDAA  #$01
                JSR   cmd2LCD
                LDY   #40
                JSR   del_50us
                RTS
;*******************************************************************************************************
del_50us        PSHX
eloop           LDX   #300      ;load 300 instead of 30(copied of guider)
iloop           NOP
                DBNE X,iloop
                DBNE Y,eloop
                PULX
                RTS
;*******************************************************************************************************
cmd2LCD:        BCLR  LCD_CNTR, LCD_RS
                JSR   dataMov
                RTS
;*******************************************************************************************************
putsLCD:        LDAA  1,X+
                BEQ   donePS
                JSR   putcLCD
                BRA   putsLCD

donePS          RTS
;*******************************************************************************************************
putcLCD:        BSET  LCD_CNTR, LCD_RS
                JSR   dataMov
                RTS
;*******************************************************************************************************
dataMov:        BSET  LCD_CNTR, LCD_E
                STAA  LCD_DAT
```

```
            BCLR  LCD_CNTR, LCD_E
            LSLA
            LSLA
            LSLA
            LSLA
            BSET  LCD_CNTR, LCD_E
            STAA  LCD_DAT
            BCLR  LCD_CNTR, LCD_E
            LDY   #1
            JSR   del_50us
            RTS
;*******************************************************************************************************
initAD        MOVB  #$C0,ATDCTL2
            JSR   del_50us
            MOVB  #$00,ATDCTL3
            MOVB  #$85,ATDCTL4
            BSET  ATDDIEN,$0C
            RTS
;*******************************************************************************************************
int2BCD       XGDX
            LDAA #0
            STAA TEN_THOUS
            STAA THOUSANDS
            STAA HUNDREDS
            STAA TENS
            STAA UNITS
            STAA BCD_SPARE
            STAA BCD_SPARE+1
            CPX #0
            BEQ CON_EXIT
            XGDX
            LDX #10
            IDIV
            STAB UNITS
            CPX #0
            BEQ CON_EXIT
            XGDX
            LDX #10
            IDIV
            STAB TENS
            CPX #0
            BEQ CON_EXIT
            XGDX
            LDX #10
```

```
            IDIV
            STAB HUNDREDS
            CPX #0
            BEQ CON_EXIT
            XGDX
            LDX #10
            IDIV
            STAB THOUSANDS
            CPX #0
            BEQ CON_EXIT
            XGDX
            LDX #10
            IDIV
            STAB TEN_THOUS

CON_EXIT    RTS

LCD_POS_CRSR  ORAA #%10000000
            JSR cmd2LCD
            RTS
;*************************************************************************************************
BIN2ASC         PSHA
              TAB
              ANDB #%00001111
              CLRA
              ADDD #HEX_TABLE
              XGDX
              LDAA 0,X
              PULB
              PSHA
              RORB
              RORB
              RORB
              RORB
              ANDB #%00001111
              CLRA
              ADDD #HEX_TABLE
              XGDX
              LDAA 0,X
              PULB
              RTS
;*****************************************************************
;* BCD to ASCII Conversion Routine
;* This routine converts the BCD number in the BCD_BUFFER
```

```
;* into ascii format, with leading zero suppression.
;* Leading zeros are converted into space characters.
;* The flag 'NO_BLANK' starts cleared and is set once a non-zero
;* digit has been detected.
;* The 'units' digit is never blanked, even if it and all the
;* preceding digits are zero.

BCD2ASC         LDAA    #0
                STAA    NO_BLANK

C_TTHOU         LDAA    TEN_THOUS
                ORAA    NO_BLANK
                BNE     NOT_BLANK1

ISBLANK1        LDAA    #' '
                STAA    TEN_THOUS
                BRA     C_THOU

NOT_BLANK1      LDAA    TEN_THOUS
                ORAA    #$30
                STAA    TEN_THOUS
                LDAA    #$1
                STAA    NO_BLANK

C_THOU          LDAA    THOUSANDS
                ORAA    NO_BLANK
                BNE     NOT_BLANK2

ISBLANK2        LDAA    #' '
                STAA    THOUSANDS
                BRA     C_HUNS

NOT_BLANK2      LDAA    THOUSANDS
                ORAA    #$30
                STAA    THOUSANDS
                LDAA    #$1
                STAA    NO_BLANK

C_HUNS          LDAA    HUNDREDS
                ORAA    NO_BLANK
                BNE     NOT_BLANK3

ISBLANK3        LDAA    #' '
                STAA    HUNDREDS
```

```
            BRA     C_TENS

NOT_BLANK3      LDAA    HUNDREDS
            ORAA    #$30
            STAA    HUNDREDS
            LDAA    #$1
            STAA    NO_BLANK

C_TENS          LDAA    TENS
            ORAA    NO_BLANK
            BNE     NOT_BLANK4

ISBLANK4        LDAA    #' '
            STAA    TENS
            BRA     C_UNITS

NOT_BLANK4      LDAA    TENS
            ORAA    #$30
            STAA    TENS

C_UNITS         LDAA    UNITS
            ORAA    #$30
            STAA    UNITS
            RTS

;*****************************************************************************************************

; Display the battery voltage

            LDAA    #$C7
            JSR     cmd2LCD
            LDAB    CRNT_STATE
            LSLB
            LSLB
            LSLB
            LDX     #tab
            ABX
            JSR     putsLCD
            RTS

;*****************************************************************************************************
;*                          Interrupt Vectors                                        *
;*****************************************************************************************************
            ORG     $FFFE
```

```
DC.W    Entry
ORG     $FFDE
DC.W    TOF_ISR
```