GP2D120 Distance sensor application using 9S12C32

Hardwares:

- SchoolBoard or
- NC12SSIM using J8 connections
- SFR04
- NC12DX

This application will use PAD2/AN02 to measure the voltage from a GP2D120 Sharp sensor. The output voltage of GP2D120 is related to distance of an object in front of the sensor. The signal is non-linear and may follow a log or power of 2.

Caution: Make sure to follow the wiring diagram (last pages) as shown.

In this example the values are calculated using floating point math written by Gordon Doughman.

Vdisplayed = V * (Vref/1024) = V * (5/1024) where Vref = 5Volts

Source Code:

Note: The MATH are written by Gordon Doughman. It is used here to calculate the voltage in floating point format.

Example Code:

```
;gp2d120.asm
*************************************************************************
*REVISION HISTORY:
*
*DATE                      REV. NO. DESCRIPTION
*
*August 13, 2004  1.00              Sharp gp2d120 Application
*
*Author: Exequiel Rarama for the 9S12C32
*************************************************************************
;Compiled using MiniIDE
;

#include C32Regs.INC

         ORG RAM

;A/D variables
ad0               ds       2
ad1               ds       2
ad2               ds       2
ad3               ds       2
ad4               ds       2
ad5               ds       2
ad6               ds       2
```

```
ad7             ds      2

;-------------------------------------------------------------------------------
FPACC1EX:       ds      1               ;FLOATING POINT ACCUMULATOR #1..
FPACC1MN:       ds      3
MANTSGN1:       ds      1               ;MANTISSA SIGN FOR FPACC1 (0=+,
FF=-).

FPACC2EX:       ds      1               ;FLOATING POINT ACCUMULATOR #2.
FPACC2MN:       ds      3
MANTSGN2:       ds      1               ;MANTISSA SIGN FOR FPACC2 (0=+,
FF=-).
;-------------------------------------------------------------------------------
;
Float2Ascii     ds      15              ;Conversion from Floating point to
ASCII
;
;-------------------------------------------------------------------------------
;General timers
dispDly         ds      2


*  Operational Parameters


;-------------------------------------------------------------------------------
*  Operational Parameters
RAM             equ     $3800           ;RAM
STACK           equ     $3F80           ;Stack at below Ubug12
FLASH           equ     $4000           ;Fixed FLASH or PPAGE = $3E
VectorTable     equ     $FF80           ;Beginning of Vector Table
interrupt

OscFreq         equ     8000            ;Enter Osc speed
initSYNR equ    $02             ; mult by synr + 1 = 3 (24MHz)
initREFDV       equ     $00             ;
PLLSEL          equ     %10000000       ;PLL select bit
LOCK            equ     %00001000       ;lock status bit
PLLON           equ     %01000000       ;phase lock loop on bit

;-------------------------------------------------------------------------------
;ATD Variables
admask2         equ     %11000000       ;AFFC,ADPU=1 - Enable Analog to Digital
admask3         equ     %00000000       ;FRZ1,FRZ0=0
admask4         equ     %10000001       ;SMP1,SMP0 = 0; S10BM,PRS0=1 - Select
Sample time adn Bit mode
admask5         equ     %01110000       ;S8CM = 1, SCAN = 1, MULT = 1
SCFflag         equ     %10000000       ;SCF - Sequence Complete flag

;RTI Variables
clrmask         equ     %11000000       ;mask for clearing timer flags
;RTIRate         equ     %00110000       ;[6:4]=2^12 =>512 microsecond RTI rate
RTIRate         equ     %00110000       ;[6:4]=2^12 =>512 microsecond RTI rate

RTIF            equ     %10000000
RTIE            equ     %10000000

;SCI Variables
scimask         equ     %00101100       ;RIE - SCI Interrupt enable
                                        ;RE - Receiver Enable
RDRFflag        equ     %00100000       ;RDRF - Receive Data Register Full flag
TDREflag        equ     %10000000       ;TDRE - Transmit Data Register Empty
flag
```

```
;Baud rate definitions
BusFreq           equ      ((OscFreq/(initREFDV+1))*(initSYNR+1))
baud115200        equ      (BusFreq/16)*10/1152                        ;sets
baud rate to 115,200
baud9600 equ      (BusFreq/16)*10/96                        ;sets baud rate
to 009,600
initSCI0CR2       equ      $0C                                         ;SCI0
Control Register 2


***************************** Program ******************************

          ORG FLASH

ResetFunc                                    ;This is where the RESET vector
points to
          sei                                ;Disable Any interrupts

          movb     #$00,INITRG               ;set registers at $0000
          movb     #$39,INITRM               ;move and set ram to end at $3fff

;Initialize Stack
          lds      #STACK                    ;initialize stack pointer

; Initialize clock generator and PLL
          bclr     CLKSEL,PLLSEL             ;disengage PLL to system
          bset     PLLCTL,PLLON              ;turn on PLL

          movb     #initSYNR,SYNR            ;set PLL multiplier
          movb     #initREFDV,REFDV          ;set PLL divider

          nop
          nop
          nop

          nop
          nop
          nop
          nop

          brclr    CRGFLG,LOCK,*+0           ;while (!(crg.crgflg.bit.lock==1))
          bset     CLKSEL,PLLSEL             ;engage PLL to system
          cli

;-----------------------------------------------------------------------------
RealTimeInit                                 ;Initialize Real Time Interrupt
          movb     #RTIRate,RTICTL           ;and initialize RTI rate
          bset     CRGFLG,RTIF               ;clear flag
          bset     CRGINT,RTIE               ;Enable RTI

;Initialize Analog To Digital
          movb     #$80,ATDCTL2             ;enable ATD
          movb     #$40,ATDCTL3             ;8 Channels ATD result
          movb     #$60,ATDCTL4              ;Select Sample rate
          movb     #$B0,ATDCTL5              ;Select 8 channel mode, Continuous
scan

;Initialize first Serial Communication Interface
          movw     #baud9600,SCIBDH  ;Set baud rate to 9600
          movb     #scimask,SCICR2           ;Rx and Tx on
          ldab     SCISR1           ;read register to clear flag RDRF
          ldab     SCIDRL                    ;read receive buffer
```

```
         ldx      #MSG1
        jsr     OutStr0

         movw     #00,dispDly

;========START of
MAIN=============================================================
main
         ldx      dispDly                    ;check the display delay
         bne      main

         jsr      CalVoltage
         movw     #122,dispDly               ;1 second delay refresh

         bra      main


;=======END of
MAIN=============================================================


;-------------------------------------------------------------------------------
* Real-time Interrupt Routine

RealTimeInt
         bset     CRGFLG,RTIF                ;clear real-time interrupt flag
         cli                                 ; so that other interrupts can be
service

RTI_6
         ldx      dispDly
         beq      RTI_7
         dex
         stx      dispDly

RTI_7
         brclr    ATDSTAT0,SCFflag,*         ;Loop here until SCF of ATD is set
save ATD
         bset     ATDSTAT0,SCFflag           ;clear Flag for next conversion

         ldd      ATDDR0H
         std      ad0

         ldd      ATDDR1H
         std      ad1

         ldd      ATDDR2H
         std      ad2

         ldd      ATDDR3H
         std      ad3

         ldd      ATDDR4H
         std      ad4

         ldd      ATDDR5H
         std      ad5

         ldd      ATDDR6H
         std      ad6

         ldd      ATDDR7H
```

```
        std     ad7
        rti

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

;-------------------------------------------------------------------------------
*  SCI Input Interrupt Handler

*  Gets bytes from SCI.  Sets COMMAND_PENDING flag.


OutStr0                          ; send a null terminated string to the
display.
        ldaa    1,x+            ; get a character, advance pointer, null?
        beq     OutStrDone      ; yes. return.
        bsr     SerOutput0      ; no. send it out the SCI.
        bra     OutStr0          ; go get the next character.
;
OutStrDone
        rts

;-------------------------------------------------------------------------------
SerOutput0
        brclr   SCISR1,TDREflag,SerOutput0     ;check if buffer is empty
        staa    SCIDRL
        rts

SerInputInt0
        ldaa    SCISR1                  ;read register to clear flag RDRF
        ldaa    SCIDRL                     ;read receive buffer
        rti

;-------------------------------------------------------------------------------
* Messages

MSG1    dc.b    '9S12C32 gp2d120 Demo V1.00',$D,$A,0


;--------------------------INCLUDE FILES START HERE-----------------------------

#include Math.Asm

;--------------------------INCLUDE FILES END HERE-------------------------------

CalVoltage
        movw    ad2,MANTSGN1-2             ;16 bit integer to be converted
        jsr     UINT2FLT

        ldab    #5                         ;Copy VrefDiv1024 into FPACC2
        ldx     #Vref5V
        ldy     #FPACC2EX
        jsr     Datacopy

        jsr     FLTMUL                     ;Result is the distance

        ldab    #5                         ;Copy VrefDiv1024 into FPACC2
        ldx     #AD1024
        ldy     #FPACC2EX
        jsr     Datacopy

        jsr     FLTDIV

        ldx     #Float2Ascii               ;Convert from Float to ASCII
```

```
        jsr     FLTASC                  ;

        ldx     #Float2Ascii            ;Display result
        jsr     OutStr0

        ldx     #mVolt
        jsr     OutStr0

        rts


mVolt   dc.b    ' volts    ', $D, $0

;-------------------------------------------------------------------------------
------------
Datacopy
        movb    1,x+,1,y+               ;temporary save
        dbne    b,Datacopy
        rts

;-------------------------------------------------------------------------------
AD1024          dc.b    $8B, $80, $00, $00, $00    ;10 bit A/D
Vref5V          dc.b    $83, $A0, $00, $00, $00    ;Vref = 5.00000 volts
VrefDiv1024     dc.b    $79, $A0, $00, $00, $00    ;Vref/1024 = 5/1024


;===============================================================================
        ORG     VectorTable             ;Definition of Vector tables
        dc.w    ResetFunc               ;Reserve
        dc.w    ResetFunc               ;Reserve
        dc.w    ResetFunc               ;Reserve
        dc.w    ResetFunc               ;Reserve

        dc.w    ResetFunc               ;PWM Emergency Shutdown
        dc.w    ResetFunc               ;VREG LVI
        dc.w    ResetFunc               ;Port P
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved

        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;CAN transmit
        dc.w    ResetFunc               ;CAN receive
        dc.w    ResetFunc               ;CAN errors
        dc.w    ResetFunc               ;CAN wake-up
        dc.w    ResetFunc               ;FLASH

        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserve
        dc.w    ResetFunc               ;Reserve
```

```
dc.w    ResetFunc               ;Reserved
dc.w    ResetFunc               ;Reserved

dc.w    ResetFunc               ;CRG Self Clock Mode
dc.w    ResetFunc               ;CRG PLL lock
dc.w    ResetFunc               ;Reserved
dc.w    ResetFunc               ;Reserved
dc.w    ResetFunc               ;Reserved

dc.w    ResetFunc               ;Port J (PIEP)
dc.w    ResetFunc               ;Reserved
dc.w    ResetFunc               ;ATD (ATDCTL2 - ASCIE)
dc.w    ResetFunc               ;Reserved
dc.w    SerInputInt0            ;SCI
dc.w    ResetFunc               ;SPI
dc.w    ResetFunc               ;Pulse Accumulator 0 input edge
dc.w    ResetFunc               ;Pulse Accumulator 0 overflow
dc.w    ResetFunc               ;Standard Timer 0 Overflow
dc.w    ResetFunc               ;Timer 0 Channel 7
dc.w    ResetFunc               ;Timer 0 Channel 6
dc.w    ResetFunc               ;Timer 0 Channel 5
dc.w    ResetFunc               ;Timer 0 Channel 4

dc.w    ResetFunc               ;Timer 0 Channel 3
dc.w    ResetFunc               ;Timer 0 Channel 2
dc.w    ResetFunc               ;Timer 0 Channel 1
dc.w    ResetFunc               ;Timer 0 Channel 0

dc.w    RealTimeInt             ;Real Time Interrupt
dc.w    ResetFunc               ;IRQ
dc.w    ResetFunc               ;XIRQ
dc.w    ResetFunc               ;SWI
dc.w    ResetFunc               ;Instruction Trap
dc.w    ResetFunc               ;COP failure
dc.w    ResetFunc               ;Clock Monitor
dc.w    ResetFunc               ;Power On Reset
```

```
;math.asm
*REVISION HISTORY:
*
*DATE                    REV. NO. DESCRIPTION
*
*August 13, 2004  1.00            Sonar Application
*
*Author: Exequiel Rarama for the 9S12C32
*******************************************************************************
;Compiled using MiniIDE

*
*        LOCAL VARIABLES (ON STACK POINTED TO BY Y)
*

;FPACC1EX        ds      1               ;FLOATING POINT ACCUMULATOR #1..
;FPACC1MN        ds      3
;MANTSGN1        ds      1               ;MANTISSA SIGN FOR FPACC1 (0=+,
FF=-).

;FPACC2EX        ds      1               ;FLOATING POINT ACCUMULATOR #2.
;FPACC2MN        ds      3
;MANTSGN2        ds      1               ;MANTISSA SIGN FOR FPACC2 (0=+,
FF=-).

*
*
FLTFMTER EQU     1               ;/* floating point format error in ASCFLT */
OVFERR           EQU     2               ;/* floating point overflow error
*/
UNFERR           EQU     3               ;/* floating point underflow error
*/
DIV0ERR          EQU     4               ;/* division by 0 error */
TOLGSMER EQU     5               ;/* number too large or small to convert to
int. */
NSQRTERR EQU     6               ;/* tried to take the square root of
negative # */
TAN90ERR EQU     7               ;/* TANgent of 90 degrees attempted */
*
EXPSIGN          EQU     0               ;EXPONENT SIGN (0=+, FF=-).
PWR10EXP EQU     1               ;POWER 10 EXPONENT.
*

*******************************************************************************
*                                                                             *
*                        ASCII TO FLOATING POINT ROUTINE                      *
*                                                                             *
*       This routine will accept most any ASCII floating point format         *
*       and return a 32-bit floating point number.  The following are         *
*       some examples of legal ASCII floating point numbers.                  *
*                                                                             *
*       20.095                                                                *
*       0.125                                                                 *
*       7.2984E10                                                             *
*       167.824E5                                                             *
*       5.9357E-7                                                             *
*       500                                                                   *
*                                                                             *
*       The floating point number returned is in "FPACC1".                    *
*                                                                             *
*                                                                             *
*       The exponent is biased by 128 to facilitate floating point           *
*       comparisons.  A pointer to the ASCII string is passed to the          *
```

```
*       routine in the X-register.                                              *
*                                                                               *
*                                                                               *
*********************************************************************************
*
*
;-------------------------------------------------------------------------------
----------------------

*
ASCFLT                                  ;Reg X points to number to be
converted to Floating point
        PSHX                            ;SAVE POINTER TO ASCII STRING.
;
        JSR     PSHFPAC2        ;SAVE FPACC2.
        LDX     #0                      ;PUSH ZEROS ON STACK TO INITIALIZE
LOCALS.
        PSHX                            ;ALLOCATE 2 BYTES FOR LOCALS.
        STX     FPACC1EX        ;CLEAR FPACC1.
        STX     FPACC1EX+2
        CLR     MANTSGN1        ;MAKE THE MANTISSA SIGN POSITIVE INITIALLY.
        TSY                             ;POINT TO LOCALS.
        LDX     6,Y                     ;GET POINTER TO ASCII STRING.

ASCFLT1
        LDAA    0,X                     ;GET 1ST CHARACTER IN STRING.
        JSR     NUMERIC                 ;IS IT A NUMBER.
        BCS     ASCFLT4                 ;YES. GO PROCESS IT.

*
*       LEADING MINUS SIGN ENCOUNTERED?
*
ASCFLT2
        CMPA    #'-'                    ;NO. IS IT A MINUS SIGN?
        BNE     ASCFLT3                 ;NO. GO CHECK FOR DECIMAL POINT.
        COM     MANTSGN1        ;YES. SET MANTISSA SIGN. LEADING MINUS
BEFORE?
        INX                             ;POINT TO NEXT CHARACTER.
        LDAA    0,X                     ;GET IT.
        JSR     NUMERIC                 ;IS IT A NUMBER?
        BCS     ASCFLT4                 ;YES. GO PROCESS IT.
*
*       LEADING DECIMAL POINT?
*

ASCFLT3
        CMPA    #'.'                    ;IS IT A DECIMAL POINT?
        BNE     ASCFLT5                 ;NO. FORMAT ERROR.
        INX                             ;YES. POINT TO NEXT CHARACTER.
        LDAA    0,X                     ;GET IT.
        JSR     NUMERIC                 ;MUST HAVE AT LEAST ONE DIGIT AFTER
D.P.
        BCC     ASCFLT5                 ;GO REPORT ERROR.
        JMP     ASCFLT11        ;GO BUILD FRACTION.
*
*       FLOATING POINT FORMAT ERROR
*
ASCFLT5
        LEAS    2,sp                    ;DE-ALLOCATE LOCALS.
        JSR     PULFPAC2        ;RESTORE FPACC2.
        PULX                            ;GET POINTER TO TERMINATING
CHARACTER IN STRING.
        LDAA    #FLTFMTER               ;FORMAT ERROR.
```

```
        SEC                             ;SET ERROR FLAG.
        RTS                             ;RETURN.
*
*       PRE DECIMAL POINT MANTISSA BUILD
*
ASCFLT4
        LDAA    0,X
        JSR     NUMERIC
        BCC     ASCFLT10
        JSR     ADDNXTD
        INX
        BCC     ASCFLT4
*
*       PRE DECIMAL POINT MANTISSA OVERFLOW
*
ASCFLT6
        INC     FPACC1EX        ;INC FOR EACH DIGIT ENCOUNTERED PRIOR TO
D.P.
        LDAA    0,X                     ;GET NEXT CHARACTER.
        INX                             ;POINT TO NEXT.
        JSR     NUMERIC                 ;IS IT S DIGIT?
        BCS     ASCFLT6                 ;YES. KEEP BUILDING POWER 10
MANTISSA.
        CMPA    #'.'                    ;NO. IS IT A DECIMAL POINT?
        BNE     ASCFLT7                 ;NO. GO CHECK FOR THE EXPONENT.
*
*       ANY FRACTIONAL DIGITS ARE NOT SIGNIFIGANT
*
ASCFLT8
        LDAA    0,X                     ;GET THE NEXT CHARACTER.
        JSR     NUMERIC                 ;IS IT A DIGIT?
        BCC     ASCFLT7                 ;NO. GO CHECK FOR AN EXPONENT.
        INX                             ;POINT TO THE NEXT CHARACTER.
        BRA     ASCFLT8                 ;FLUSH REMAINING DIGITS.

ASCFLT7
        CMPA    #'E'                    ;NO. IS IT THE EXPONENT? (upper
case)
        BEQ     ASCFLT13        ;YES. GO PROCESS IT.
        CMPA    #'e'                    ;IS IT THE EXPONENT? (lower case)
        BEQ     ASCFLT13        ;YES. GO PROCESS IT.
        lbra    FINISH                  ;NO. GO FINISH THE CONVERSION.
*
*       PROCESS THE EXPONENT
*
ASCFLT13
        INX                             ;POINT TO NEXT CHARACTER.
        LDAA    0,X                     ;GET THE NEXT CHARACTER.
        JSR     NUMERIC                 ;SEE IF IT'S A DIGIT.
        BCS     ASCFLT9                 ;YES. GET THE EXPONENT.
        CMPA    #'-'                    ;NO. IS IT A MINUS SIGN?
        BEQ     ASCFLT15        ;YES. GO FLAG A NEGATIVE EXPONENT.
        CMPA    #'+'                    ;NO. IS IT A PLUS SIGN?
        BEQ     ASCFLT16        ;YES. JUST IGNORE IT.
        BRA     ASCFLT5                 ;NO. FORMAT ERROR.

ASCFLT15
        COM     EXPSIGN,Y               ;FLAG A NEGATIVE EXPONENT. IS IT
1ST?

ASCFLT16
        INX                             ;POINT TO NEXT CHARACTER.
        LDAA    0,X                     ;GET NEXT CHARACTER.
```

```
         JSR      NUMERIC                    ;IS IT A NUMBER?
         BCC      ASCFLT5                    ;NO. FORMAT ERROR.

ASCFLT9
         SUBA     #$30                       ;MAKE IT BINARY.
         STAA     PWR10EXP,Y                 ;BUILD THE POWER 10 EXPONENT.
         INX                                 ;POINT TO NEXT CHARACTER.
         LDAA     0,X                        ;GET IT.
         JSR      NUMERIC                    ;IS IT NUMERIC?
         BCC      ASCFLT14          ;NO. GO FINISH UP THE CONVERSION.
         LDAB     PWR10EXP,Y                 ;YES. GET PREVIOUS DIGIT.
         LSLB                                ;MULT. BY 2.
         LSLB                                ;NOW BY 4.
         ADDB     PWR10EXP,Y                 ;BY 5.
         LSLB                                ;BY 10.
         SUBA     #$30                       ;MAKE SECOND DIGIT BINARY.
         ABA                                 ;ADD IT TO FIRST DIGIT.
         STAA     PWR10EXP,Y
         CMPA     #38                        ;IS THE EXPONENT OUT OF RANGE?
         BHI      ASCFLT5                    ;YES. REPORT ERROR.

ASCFLT14
         LDAA     PWR10EXP,Y                 ;GET POWER 10 EXPONENT.
         TST      EXPSIGN,Y                  ;WAS IT NEGATIVE?
         BPL      ASCFLT12          ;NO. GO ADD IT TO BUILT 10 PWR EXPONENT.
         NEGA

ASCFLT12
         ADDA     FPACC1EX          ;FINAL TOTAL PWR 10 EXPONENT.
         STAA     FPACC1EX          ;SAVE RESULT.
         BRA      FINISH                     ;GO FINISH UP CONVERSION.
*
*        PRE-DECIMAL POINT NON-DIGIT FOUND, IS IT A DECIMAL POINT?
*
ASCFLT10
         CMPA     #'.'                       ;IS IT A DECIMAL POINT?
         BNE      ASCFLT7                    ;NO. GO CHECK FOR THE EXPONENT.
         INX                                 ;YES. POINT TO NEXT CHARACTER.
*
*        POST DECIMAL POINT PROCESSING
*
ASCFLT11
         LDAA     0,X                        ;GET NEXT CHARACTER.
         JSR      NUMERIC                    ;IS IT NUMERIC?
         BCC      ASCFLT7                    ;NO. GO CHECK FOR EXPONENT.
         BSR      ADDNXTD                    ;YES. ADD IN THE DIGIT.
         INX                                 ;POINT TO THE NEXT CHARACTER.
         BCS      ASCFLT8                    ;IF OVER FLOW, FLUSH REMAINING
DIGITS.
         DEC      FPACC1EX          ;ADJUST THE 10 POWER EXPONENT.
         BRA      ASCFLT11          ;PROCESS ALL FRACTIONAL DIGITS.
*
*
*
ADDNXTD
         LDAA     FPACC1MN          ;GET UPPER 8 BITS.
         STAA     FPACC2MN          ;COPY INTO FPAC2.
         LDD      FPACC1MN+1                 ;GET LOWER 16 BITS OF MANTISSA.
         STD      FPACC2MN+1                 ;COPY INTO FPACC2.
         LSLD                                ;MULT. BY 2.
         ROL      FPACC1MN          ;OVERFLOW?
         BCS      ADDNXTD1          ;YES. DON'T ADD THE DIGIT IN.
         LSLD                                ;MULT BY 4.
```

```
        ROL     FPACC1MN        ;OVERFLOW?
        BCS     ADDNXTD1        ;YES. DON'T ADD THE DIGIT IN.
        ADDD    FPACC2MN+1              ;BY 5.
        PSHA                    ;SAVE A.
        LDAA    FPACC1MN        ;GET UPPER 8 BITS.
        ADCA    #0                      ;ADDIN POSSABLE CARRY FROM LOWER 16
BITS.
        ADDA    FPACC2MN        ;ADD IN UPPER 8 BITS.
        STAA    FPACC1MN        ;SAVE IT.
        PULA                    ;RESTORE A.
        BCS     ADDNXTD1        ;OVERFLOW? IF SO DON'T ADD IT IN.
        LSLD                    ;BY 10.
        ROL     FPACC1MN
        STD     FPACC1MN+1              ;SAVE THE LOWER 16 BITS.
        BCS     ADDNXTD1        ;OVERFLOW? IF SO DON'T ADD IT IN.
        LDAB    0,X                     ;GET CURRENT DIGIT.
        SUBB    #$30                    ;MAKE IT BINARY.
        CLRA                    ;16-BIT.
        ADDD    FPACC1MN+1              ;ADD IT IN TO TOTAL.
        STD     FPACC1MN+1              ;SAVE THE RESULT.
        LDAA    FPACC1MN        ;GET UPPER 8 BITS.
        ADCA    #0                      ;ADD IN POSSIBLE CARRY. OVERFLOW?
        BCS     ADDNXTD1        ;YES. COPY OLD MANTISSA FROM FPACC2.
        STAA    FPACC1MN        ;NO. EVERYHING OK.
        RTS                     ;RETURN.

ADDNXTD1
        LDD     FPACC2MN+1              ;RESTORE THE ORIGINAL MANTISSA
BECAUSE
        STD     FPACC1MN+1              ;OF OVERFLOW.
        LDAA    FPACC2MN
        STAA    FPACC1MN
        RTS                     ;RETURN.
*
*
*
*       NOW FINISH UP CONVERSION BY MULTIPLYING THE RESULTANT MANTISSA
*       BY 10 FOR EACH POSITIVE POWER OF 10 EXPONENT RECIEVED OR BY .1
*       (DIVIDE BY 10) FOR EACH NEGATIVE POWER OF 10 EXPONENT RECIEVED.
*
*
FINISH
        STX     6,Y                     ;SAVE POINTER TO TERMINATING
CHARACTER IN STRING.
        LDX     #FPACC1EX               ;POINT TO FPACC1.
        JSR     CHCK0                   ;SEE IF THE NUMBER IS ZERO.
        BEQ     FINISH3                 ;QUIT IF IT IS.

        LDAA    FPACC1EX        ;GET THE POWER 10 EXPONENT.
        STAA    PWR10EXP,Y              ;SAVE IT.

        LDAA    #$80+24                 ;SET UP INITIAL EXPONENT (# OF BITS
+ BIAS).
        STAA    FPACC1EX
        JSR     FPNORM                  ;GO NORMALIZE THE MANTISSA.
        TST     PWR10EXP,Y              ;IS THE POWER 10 EXPONENT POSITIVE
OR ZERO?
        BEQ     FINISH3                 ;IT'S ZERO, WE'RE DONE.
        BPL     FINISH1                 ;IT'S POSITIVE MULTIPLY BY 10.

        LDX     #CONSTP1        ;NO. GET CONSTANT .1 (DIVIDE BY 10).
        JSR     GETFPAC2        ;GET CONSTANT INTO FPACC2.
```

```
         NEG     PWR10EXP,Y              ;MAKE THE POWER 10 EXPONENT
POSITIVE.
         BRA     FINISH2                 ;GO DO THE MULTIPLIES.

FINISH1
         LDX     #CONST10          ;GET CONSTANT '10' TO MULTIPLY BY.
         JSR     GETFPAC2          ;GET CONSTANT INTO FPACC2.

FINISH2
         JSR     FLTMUL                  ;GO MULTIPLY FPACC1 BY FPACC2,
RESULT IN FPACC1.
         DEC     PWR10EXP,Y              ;DECREMENT THE POWER 10 EXPONENT.
         BNE     FINISH2                 ;GO CHECK TO SEE IF WE'RE DONE.

FINISH3
         LEAS    2,sp                    ;DE-ALLOCATE LOCALS.
         JSR     PULFPAC2          ;RESTORE FPACC2.
         PULX                            ;GET POINTER TO TERMINATING
CHARACTER IN STRING.
         RTS                             ;RETURN WITH NUMBER IN FPACC1.
*
*
NUMERIC
         CMPA    #'0'                    ;IS IT LESS THAN AN ASCII 0?
         BLO     NUMERIC1          ;YES. NOT NUMERIC.
         CMPA    #'9'                    ;IS IT GREATER THAN AN ASCII 9?
         BHI     NUMERIC1          ;YES. NOT NUMERIC.
         SEC                             ;IT WAS NUMERIC. SET THE CARRY.
         RTS                             ;RETURN.

NUMERIC1
         CLC                             ;NON-NUMERIC CHARACTER. CLEAR THE
CARRY.
         RTS                             ;RETURN.

*
FPNORM
         LDX     #FPACC1EX               ;POINT TO FPACC1.
         BSR     CHCK0                   ;CHECK TO SEE IF IT'S 0.
         BEQ     FPNORM3                 ;YES. JUST RETURN.
         TST     FPACC1MN          ;IS THE NUMBER ALREADY NORMALIZED?
         BMI     FPNORM3                 ;YES. JUST RETURN..

FPNORM1
         LDD     FPACC1MN+1              ;GET THE LOWER 16 BITS OF THE
MANTISSA.

FPNORM2
         DEC     FPACC1EX          ;DECREMENT THE EXPONENT FOR EACH SHIFT.
         BEQ     FPNORM4                 ;EXPONENT WENT TO 0. UNDERFLOW.
         LSLD                            ;SHIFT THE LOWER 16 BITS.
         ROL     FPACC1MN          ;ROTATE THE UPPER 8 BITS. NUMBER NORMALIZED?
         BPL     FPNORM2                 ;NO. KEEP SHIFTING TO THE LEFT.
         STD     FPACC1MN+1              ;PUT THE LOWER 16 BITS BACK INTO
FPACC1.

FPNORM3
         CLC                             ;SHOW NO ERRORS.
         RTS                             ;YES. RETURN.

FPNORM4
         SEC                             ;FLAG ERROR.
         RTS                             ;RETURN.
```

```
*
CHCK0                                   ;CHECKS FOR ZERO IN FPACC POINTED
TO BY X.
         PSHD                           ;SAVE D.
         LDD     0,X                    ;GET FPACC EXPONENT & HIGH 8 BITS.
         BNE     CHCK01                 ;NOT ZERO. RETURN.
         LDD     2,X                    ;CHECK LOWER 16 BITS.

CHCK01
         PULD                           ;RESTORE D.
         RTS                            ;RETURN WITH CC SET.
*
CONSTP1
         dc.b    $7D,$4C,$CC,$CD        ;0.1 DECIMAL

CONST10
         dc.b    $84,$20,$00,$00        ;10.0 DECIMAL
*

*************************************************************************
*                                                                       *
*                     FPMULT: FLOATING POINT MULTIPLY                   *
*                                                                       *
*       THIS FLOATING POINT MULTIPLY ROUTINE MULTIPLIES "FPACC1" BY     *
*       "FPACC2" AND PLACES THE RESULT IN TO FPACC1. FPACC2 REMAINS     *
*       UNCHANGED.                                                      *
*                           WORSE CASE = 2077 CYCLES = 1039 uS @ 2MHz   *
*                           BEST CASE  = 1475 CYCLES = 738 uS @ 2MHz    *
*                           AVERAGE    = 1776 CYCLES = 888 uS @ 2MHz    *
*                                                                       *
*************************************************************************
*
*
FLTMUL
         JSR     PSHFPAC2       ;SAVE FPACC2.
         LDX     #FPACC1EX              ;POINT TO FPACC1
         JSR     CHCK0                  ;CHECK TO SEE IF FPACC1 IS ZERO.
         BEQ     FPMULT3                ;IT IS. ANSWER IS 0.
         LDX     #FPACC2EX              ;POINT TO FPACC2.
         JSR     CHCK0                  ;IS IT 0?
         BNE     FPMULT4                ;NO. CONTINUE.
         CLRA                           ;CLEAR D.
         CLRB
         STD     FPACC1EX       ;MAKE FPACC1 0.
         STD     FPACC1MN+1
         BRA     FPMULT3                ;RETURN.

FPMULT4
         LDAA    MANTSGN1       ;GET FPACC1 EXPONENT.
         EORA    MANTSGN2       ;SET THE SIGN OF THE RESULT.
         STAA    MANTSGN1       ;SAVE THE SIGN OF THE RESULT.
         LDAA    FPACC1EX       ;GET FPACC1 EXPONENT.
         ADDA    FPACC2EX       ;ADD IT TO FPACC2 EXPONENT.
         BPL     FPMULT1                ;IF RESULT IS MINUS AND
         BCC     FPMULT2                ;THE CARRY IS SET THEN:

FPMULT5
         LDAA    #OVFERR                ;OVERFLOW ERROR.
         SEC                            ;SET ERROR FLAG.
         BRA     FPMULT6                ;RETURN.

FPMULT1
```

```
         BCS      FPMULT2                    ;IF RESULT IS PLUS & THE CARRY IS
SET THEN ALL OK.
         LDAA     #UNFERR                    ;ELSE UNDERFLOW ERROR OCCURED.
         SEC                                 ;FLAG ERROR.
         BRA      FPMULT6                    ;RETURN.

FPMULT2
         ADDA     #$80                       ;ADD 128 BIAS BACK IN THAT WE LOST.
         STAA     FPACC1EX          ;SAVE THE NEW EXPONENT.
         JSR      UMULT                      ;GO MULTIPLY THE "INTEGER"
MANTISSAS.
         TST      FPACC1EX          ;WAS THERE AN OVERFLOW ERROR FROM ROUNDING?
         BEQ      FPMULT5                    ;YES. RETURN ERROR.

FPMULT3
         CLC                                 ;SHOW NO ERRORS.
                                             ;Moved the FPMULT3 label to this
instruction from the TST instruction above. G.S.D. 12/20/91
FPMULT6
         JSR      PULFPAC2          ;RESTORE FPACC2.
         RTS
*
*
UMULT
         LDX      #0
         PSHX                                ;CREATE PARTIAL PRODUCT REGISTER
AND COUNTER.
         PSHX
         TSX                                 ;POINT TO THE VARIABLES.
         LDAA     #24                        ;SET COUNT TO THE NUMBER OF BITS.
         STAA     0,X

UMULT1
         LDAA     FPACC2MN+2                 ;GET THE L.S. BYTE OF THE
MULTIPLIER.
         LSRA                                ;PUT L.S. BIT IN CARRY.
         BCC      UMULT2                     ;IF CARRY CLEAR, DON'T ADD
MULTIPLICAND TO P.P.
         LDD      FPACC1MN+1                 ;GET MULTIPLICAND L.S. 16 BITS.
         ADDD     2,X                        ;ADD TO PARTIAL PRODUCT.
         STD      2,X                        ;SAVE IN P.P.
         LDAA     FPACC1MN          ;GET UPPER 8 BITS OF MULTIPLICAND.
         ADCA     1,X                        ;ADD IT W/ CARRY TO P.P.
         STAA     1,X                        ;SAVE TO PARTIAL PRODUCT.

UMULT2
         ROR      1,X                        ;ROTATE PARTIAL PRODUCT TO THE
RIGHT.
         ROR      2,X
         ROR      3,X
         ROR      FPACC2MN          ;SHIFT THE MULTIPLIER TO THE RIGHT 1 BIT.
         ROR      FPACC2MN+1
         ROR      FPACC2MN+2
         DEC      0,X                        ;DONE YET?
         BNE      UMULT1                     ;NO. KEEP GOING.
         TST      1,X                        ;DOES PARTIAL PRODUCT NEED TO BE
NORMALIZED?
         BMI      UMULT3                     ;NO. GET ANSWER & RETURN.
         LSL      FPACC2MN          ;GET BIT THAT WAS SHIFTED OUT OF P.P
REGISTER.
         ROL      3,X                        ;PUT IT BACK INTO THE PARTIAL
PRODUCT.
         ROL      2,X
```

```
         ROL     1,X
         DEC     FPACC1EX        ;FIX EXPONENT.

UMULT3
         TST     FPACC2MN        ;DO WE NEED TO ROUND THE PARTIAL PRODUCT?
         BPL     UMULT4                  ;NO. JUST RETURN.
         LDD     2,X                     ;YES. GET THE LEAST SIGNIFIGANT 16
BITS.
         ADDD    #1                      ;ADD 1.
         STD     2,X                     ;SAVE RESULT.
         LDAA    1,X                     ;PROPIGATE THROUGH.
         ADCA    #0
         STAA    1,X
         BCC     UMULT4                  ;IF CARRY CLEAR ALL IS OK.
         ROR     1,X                     ;IF NOT OVERFLOW. ROTATE CARRY INTO
P.P.
         ROR     2,X
         ROR     3,X
         INC     FPACC1EX        ;UP THE EXPONENT.

UMULT4
         INS                             ;TAKE COUNTER OFF STACK.
         PULX                            ;GET M.S. 16 BITS OF PARTIAL
PRODUCT.
         STX     FPACC1MN        ;PUT IT IN FPACC1.
         PULA                            ;GET L.S. 8 BITS OF PARTIAL
PRODUCT.
         STAA    FPACC1MN+2              ;PUT IT IN FPACC1.
         RTS                             ;RETURN.
*
*

*******************************************************************************
*                                                                             *
*                       FLOATING POINT ADDITION                               *
*                                                                             *
*       This subroutine performs floating point addition of the two numbers   *
*       in FPACC1 and FPACC2.  The result of the addition is placed in        *
*       FPACC1 while FPACC2 remains unchanged.  This subroutine performs      *
*       full signed addition so either number may be of the same or opposite  *
*       sign.                                                                 *
*                          WORSE CASE = 782 CYCLES = 391 uS @ 2MHz            *
*                          BEST CASE  = 123 CYCLES =  62 uS @ 2MHz            *
*                          AVERAGE    = 409 CYCLES = 205 uS @ 2MHz            *
*                                                                             *
*******************************************************************************
*
*
FLTADD
         JSR     PSHFPAC2        ;SAVE FPACC2.
         LDX     #FPACC2EX               ;POINT TO FPACC2
         JSR     CHCK0                   ;IS IT ZERO?
         BNE     FLTADD1                 ;NO. GO CHECK FOR 0 IN FPACC1.

FLTADD6
         CLC                             ;NO ERRORS.

FLTADD10
         JSR     PULFPAC2        ;RESTORE FPACC2.
         RTS                             ;ANSWER IN FPACC1. RETURN.

FLTADD1
         LDX     #FPACC1EX               ;POINT TO FPACC1.
```

```
         JSR      CHCK0                    ;IS IT ZERO?
         BNE      FLTADD2                  ;NO. GO ADD THE NUMBER.

FLTADD4
         LDD      FPACC2EX          ;ANSWER IS IN FPACC2. MOVE IT INTO FPACC1.
         STD      FPACC1EX
         LDD      FPACC2MN+1               ;MOVE LOWER 16 BITS OF MANTISSA.
         STD      FPACC1MN+1
         LDAA     MANTSGN2          ;MOVE FPACC2 MANTISSA SIGN INTO FPACC1.
         STAA     MANTSGN1
         BRA      FLTADD6                  ;RETURN.

FLTADD2
         LDAA     FPACC1EX          ;GET FPACC1 EXPONENT.
         CMPA     FPACC2EX          ;ARE THE EXPONENTS THE SAME?
         BEQ      FLTADD7                  ;YES. GO ADD THE MANTISSA'S.
         SUBA     FPACC2EX          ;NO. FPACC1EX-FPACC2EX. IS FPACC1 > FPACC2?
         BPL      FLTADD3                  ;YES. GO CHECK RANGE.
         NEGA                              ;NO. FPACC1 < FPACC2. MAKE
DIFFERENCE POSITIVE.
         CMPA     #23                      ;ARE THE NUMBERS WITHIN RANGE?
         BHI      FLTADD4                  ;NO. FPACC2 IS LARGER. GO MOVE IT
INTO FPACC1.
         TAB                               ;PUT DIFFERENCE IN B.
         ADDB     FPACC1EX          ;CORRECT FPACC1 EXPONENT.
         STAB     FPACC1EX          ;SAVE THE RESULT.
         LDX      #FPACC1MN                ;POINT TO FPACC1 MANTISSA.
         BRA      FLTADD5                  ;GO DENORMALIZE FPACC1 FOR THE ADD.

FLTADD3
         CMPA     #23                      ;FPACC1 > FPACC2. ARE THE NUMBERS
WITHIN RANGE?
         BHI      FLTADD6                  ;NO. ANSWER ALREADY IN FPACC1. JUST
RETURN.
         LDX      #FPACC2MN                ;POINT TO THE MANTISSA TO
DENORMALIZE.

FLTADD5
         LSR      0,X                      ;SHIFT THE FIRST BYTE OF THE
MANTISSA.
         ROR      1,X                      ;THE SECOND.
         ROR      2,X                      ;AND THE THIRD.
         DECA                              ;DONE YET?
         BNE      FLTADD5                  ;NO. KEEP SHIFTING.

FLTADD7
         LDAA     MANTSGN1          ;GET FPACC1 MANTISSA SIGN.
         CMPA     MANTSGN2          ;ARE THE SIGNS THE SAME?
         BEQ      FLTADD11          ;YES. JUST GO ADD THE TWO MANTISSAS.
         TST      MANTSGN1          ;NO. IS FPACC1 THE NEGATIVE NUMBER?
         BPL      FLTADD8                  ;NO. GO DO FPACC1-FPACC2.
         LDX      FPACC2MN          ;YES. EXCHANGE FPACC1 & FPACC2 BEFORE THE
SUB.
         PSHX                              ;SAVE IT.
         LDX      FPACC1MN          ;GET PART OF FPACC1.
         STX      FPACC2MN          ;PUT IT IN FPACC2.
         PULX                              ;GET SAVED PORTION OF FPACC2
         STX      FPACC1MN          ;PUT IT IN FPACC1.
         LDX      FPACC2MN+2               ;GET LOWER 8 BITS & SIGN OF FPACC2.
         PSHX                              ;SAVE IT.
         LDX      FPACC1MN+2               ;GET LOWER 8 BITS & SIGN OF FPACC1.
         STX      FPACC2MN+2               ;PUT IT IN FPACC2.
         PULX                              ;GET SAVED PART OF FPACC2.
```

```
         STX      FPACC1MN+2                ;PUT IT IN FPACC1.

FLTADD8
         LDD      FPACC1MN+1                ;GET LOWER 16 BITS OF FPACC1.
         SUBD     FPACC2MN+1                ;SUBTRACT LOWER 16 BITS OF FPACC2.
         STD      FPACC1MN+1                ;SAVE RESULT.
         LDAA     FPACC1MN          ;GET HIGH 8 BITS OF FPACC1 MANTISSA.
         SBCA     FPACC2MN          ;SUBTRACT HIGH 8 BITS OF FPACC2.
         STAA     FPACC1MN          ;SAVE THE RESULT. IS THE RESULT NEGATIVE?
         BCC      FLTADD9                   ;NO. GO NORMALIZE THE RESULT.
         LDAA     FPACC1MN          ;YES. NEGATE THE MANTISSA.
         COMA
         PSHA                               ;SAVE THE RESULT.
         LDD      FPACC1MN+1                ;GET LOWER 16 BITS.
         COMB                               ;FORM THE ONE'S COMPLEMENT.
         COMA
         ADDD     #1                        ;FORM THE TWO'S COMPLEMENT.
         STD      FPACC1MN+1                ;SAVE THE RESULT.
         PULA                               ;GET UPPER 8 BITS BACK.
         ADCA     #0                        ;ADD IN POSSIBLE CARRY.
         STAA     FPACC1MN          ;SAVE RESULT.
         LDAA     #$FF                      ;SHOW THAT FPACC1 IS NEGATIVE.
         STAA     MANTSGN1

*
*        The following 7 lines were added 12/18/91 to check for a 0 mantissa
after the subtraction
*        performed above. If the mantissa is 0, the FPACC1 exponent & sign byte
are cleared.
*

FLTADD9
         LDD      FPACC1MN          ;Did the FPACC1 mantissa go to 0 after the
subtract?
         BNE      FLTADD13          ;No. Go normalize the result.
         TST      FPACC1MN+2                ;The upper 16-bits were 0, how
about the lower 8-bits?
         BNE      FLTADD13          ;No. Go normalize the result.
         CLR      FPACC1EX          ;The mantissa is 0. Set the exponent & sign
byte to 0.
         CLR      MANTSGN1
         BRA      FLTADD12          ;Return with no errors.
*
*
*
FLTADD13
         JSR      FPNORM                    ;GO NORMALIZE THE RESULT.
         BCC      FLTADD12          ;EVERYTHING'S OK SO RETURN.
         LDAA     #UNFERR                   ;UNDERFLOW OCCURED DURING
NORMALIZATION.
         SEC                                ;FLAG ERROR.
         JMP      FLTADD10          ;RETURN.

FLTADD12
         JMP      FLTADD6                   ;CAN'T BRANCH THAT FAR FROM HERE.
*
FLTADD11
         LDD      FPACC1MN+1                ;GET LOWER 16 BITS OF FPACC1.
         ADDD     FPACC2MN+1                ;ADD IT TO THE LOWER 16 BITS OF
FPACC2.
         STD      FPACC1MN+1                ;SAVE RESULT IN FPACC1.
         LDAA     FPACC1MN          ;GET UPPER 8 BITS OF FPACC1.
```

```
         ADCA     FPACC2MN          ;ADD IT (WITH CARRY) TO UPPER 8 BITS OF
FPACC2.
         STAA     FPACC1MN          ;SAVE THE RESULT.
         BCC      FLTADD12          ;NO OVERFLOW SO JUST RETURN.
         ROR      FPACC1MN          ;PUT THE CARRY INTO THE MANTISSA.
         ROR      FPACC1MN+1                ;PROPIGATE THROUGH MANTISSA.
         ROR      FPACC1MN+2
         INC      FPACC1EX          ;UP THE MANTISSA BY 1.
         BNE      FLTADD12          ;EVERYTHING'S OK JUST RETURN.
         LDAA     #OVFERR                   ;RESULT WAS TOO LARGE. OVERFLOW.
         SEC                                ;FLAG ERROR.
         JMP      FLTADD10          ;RETURN.
*
*
*******************************************************************************
*                                                                             *
*                     FLOATING POINT SUBTRACT SUBROUTINE                      *
*                                                                             *
*       This subroutine performs floating point subtraction ( FPACC1-FPACC2)  *
*       by inverting the sign of FPACC2 and then calling FLTADD since         *
*       FLTADD performs complete signed addition.  Upon returning from        *
*       FLTADD the sign of FPACC2 is again inverted to leave it unchanged     *
*       from its original value.                                              *
*                                                                             *
*                          WORSE CASE = 797 CYCLES = 399 uS @ 2MHz            *
*                          BEST CASE  = 148 CYCLES =  74 uS @ 2MHz            *
*                          AVERAGE    = 434 CYCLES = 217 uS @ 2MHz            *
*                                                                             *
*******************************************************************************
*
*
FLTSUB
         BSR      FLTSUB1                   ;INVERT SIGN.
         JSR      FLTADD                    ;GO DO FLOATING POINT ADD.

FLTSUB1
         LDAA     MANTSGN2          ;GET FPACC2 MANTISSA SIGN.
         EORA     #$FF                      ;INVERT THE SIGN.
         STAA     MANTSGN2          ;PUT BACK.
         RTS                                ;RETURN.
*
*

*******************************************************************************
*                                                                             *
*                          FLOATING POINT DIVIDE                              *
*                                                                             *
*        This subroutine performs signed floating point divide. The           *
*        operation performed is FPACC1/FPACC2.  The divisor (FPACC2) is left  *
*        unaltered and the answer is placed in FPACC1.  There are several     *
*        error conditions that can be returned by this routine.  They are:    *
*        a) division by zero.  b) overflow.  c) underflow.  As with all       *
*        other routines, an error is indicated by the carry being set and     *
*        the error code being in the A-reg.                                   *
*                                                                             *
*******************************************************************************
*
*
FLTDIV
         LDX      #FPACC2EX                 ;POINT TO FPACC2.
         JSR      CHCK0                     ;IS THE DIVISOR 0?
         BNE      FLTDIV1                   ;NO. GO SEE IF THE DIVIDEND IS
ZERO.
```

```
        LDAA    #DIV0ERR        ;YES. RETURN A DIVIDE BY ZERO ERROR.
        SEC                             ;FLAG ERROR.
        RTS                             ;RETURN.

FLTDIV1
        LDX     #FPACC1EX               ;POINT TO FPACC1.
        JSR     CHCK0                   ;IS THE DIVIDEND 0?
        BNE     FLTDIV2                 ;NO. GO PERFORM THE DIVIDE.
        CLC                             ;YES. ANSWER IS ZERO. NO ERRORS.
        RTS                             ;RETURN.

FLTDIV2
        JSR     PSHFPAC2        ;SAVE FPACC2.
        LDAA    MANTSGN2        ;GET FPACC2 MANTISSA SIGN.
        EORA    MANTSGN1        ;SET THE SIGN OF THE RESULT.
        STAA    MANTSGN1        ;SAVE THE RESULT.
        LDX     #0                      ;SET UP WORK SPACE ON THE STACK.
        PSHX
        PSHX
        PSHX
        LDAA    #24                     ;PUT LOOP COUNT ON STACK.
        PSHA
        TSX                             ;SET UP POINTER TO WORK SPACE.
        LDD     FPACC1MN        ;COMPARE FPACC1 & FPACC2 MANTISSAS.
        CPD     FPACC2MN        ;ARE THE UPPER 16 BITS THE SAME?
        BNE     FLTDIV3                 ;NO.
        LDAA    FPACC1MN+2              ;YES. COMPARE THE LOWER 8 BITS.
        CMPA    FPACC2MN+2

FLTDIV3
        BHS     FLTDIV4                 ;IS FPACC2 MANTISSA > FPACC1
MANTISSA? NO.
        INC     FPACC2EX        ;ADD 1 TO THE EXPONENT TO KEEP NUMBER THE
SAME.
                                        ; DID OVERFLOW OCCUR?
        BNE     FLTDIV14        ;NO. GO SHIFT THE MANTISSA RIGHT 1 BIT.

FLTDIV8
        LDAA    #OVFERR                 ;YES. GET ERROR CODE.
        SEC                             ;FLAG ERROR.

FLTDIV6
        PULX                            ;REMOVE WORKSPACE FROM STACK.
        PULX
        PULX
        INS
        JSR     PULFPAC2        ;RESTORE FPACC2.
        RTS                             ;RETURN.

FLTDIV4
        LDD     FPACC1MN+1              ;DO AN INITIAL SUBTRACT IF DIVIDEND
MANTISSA IS
        SUBD    FPACC2MN+1              ;GREATER THAN DIVISOR MANTISSA.
        STD     FPACC1MN+1
        LDAA    FPACC1MN
        SBCA    FPACC2MN
        STAA    FPACC1MN
        DEC     0,X                     ;SUBTRACT 1 FROM THE LOOP COUNT.

FLTDIV14
        LSR     FPACC2MN        ;SHIFT THE DIVISOR TO THE RIGHT 1 BIT.
        ROR     FPACC2MN+1
        ROR     FPACC2MN+2
```

```
        LDAA    FPACC1EX        ;GET FPACC1 EXPONENT.
        LDAB    FPACC2EX        ;GET FPACC2 EXPONENT.
        NEGB                            ;ADD THE TWO'S COMPLEMENT TO SET
FLAGS PROPERLY.
        ABA
        BMI     FLTDIV5                 ;IF RESULT MINUS CHECK CARRY FOR
POSS. OVERFLOW.
        BCS     FLTDIV7                 ;IF PLUS & CARRY SET ALL IS OK.
        LDAA    #UNFERR                 ;IF NOT, UNDERFLOW ERROR.
        BRA     FLTDIV6                 ;RETURN WITH ERROR.

FLTDIV5
        BCS     FLTDIV8                 ;IF MINUS & CARRY SET OVERFLOW
ERROR.

FLTDIV7
        ADDA    #$81                    ;ADD BACK BIAS+1 (THE '1'
COMPENSATES FOR ALGOR.)
        STAA    FPACC1EX        ;SAVE RESULT.

FLTDIV9
        LDD     FPACC1MN        ;SAVE DIVIDEND IN CASE SUBTRACTION DOESN'T
GO.
        STD     4,X
        LDAA    FPACC1MN+2
        STAA    6,X
        LDD     FPACC1MN+1              ;GET LOWER 16 BITS FOR SUBTRACTION.
        SUBD    FPACC2MN+1
        STD     FPACC1MN+1              ;SAVE RESULT.
        LDAA    FPACC1MN        ;GET HIGH 8 BITS.
        SBCA    FPACC2MN
        STAA    FPACC1MN
        BPL     FLTDIV10        ;SUBTRACTION WENT OK. GO DO SHIFTS.
        LDD     4,X                     ;RESTORE OLD DIVIDEND.
        STD     FPACC1MN
        LDAA    6,X
        STAA    FPACC1MN+2

FLTDIV10
        ROL     3,X                     ;ROTATE CARRY INTO QUOTIENT.
        ROL     2,X
        ROL     1,X
        LSL     FPACC1MN+2              ;SHIFT DIVIDEND TO LEFT FOR NEXT
SUBTRACT.
        ROL     FPACC1MN+1
        ROL     FPACC1MN
        DEC     0,X                     ;DONE YET?
        BNE     FLTDIV9                 ;NO. KEEP GOING.
        COM     1,X                     ;RESULT MUST BE COMPLEMENTED.
        COM     2,X
        COM     3,X
        LDD     FPACC1MN+1              ;DO 1 MORE SUBTRACT FOR ROUNDING.
        SUBD    FPACC2MN+1              ;( DON'T NEED TO SAVE THE RESULT. )
        LDAA    FPACC1MN
        SBCA    FPACC2MN        ;( NO NEED TO SAVE THE RESULT. )
        LDD     2,X                     ;GET LOW 16 BITS.
        BCC     FLTDIV11        ;IF IT DIDNT GO RESULT OK AS IS.
        CLC                             ;CLEAR THE CARRY.
        BRA     FLTDIV13        ;GO SAVE THE NUMBER.

FLTDIV11
        ADDD    #1                      ;ROUND UP BY 1.
```

```
FLTDIV13
         STD      FPACC1MN+1                ;PUT IT IN FPACC1.
         LDAA     1,X                       ;GET HIGH 8 BITS.
         ADCA     #0
         STAA     FPACC1MN          ;SAVE RESULT.
         BCC      FLTDIV12          ;IF CARRY CLEAR ANSWER OK.
         ROR      FPACC1MN          ;IF NOT OVERFLOW. ROTATE CARRY IN.
         ROR      FPACC1MN+1
         ROR      FPACC1MN+2
         INC      FPACC1EX          ;Compensate the exponent for rotate right.
Added 12/17/91 G.S.D.
         BNE      FLTDIV12          ;if the exponent didn't go to zero, the
answer's OK.
         JMP      FLTDIV8                   ;if not an overflow occurred.

FLTDIV12
         CLC                                ;NO ERRORS.
         JMP      FLTDIV6                   ;RETURN.
*
*

*****************************************************************************
*                                                                           *
*                FLOATING POINT TO ASCII CONVERSION SUBROUTINE              *
*                                                                           *
*        This subroutine performs floating point to ASCII conversion of     *
*        the number in FPACC1.  The ascii string is placed in a buffer      *
*        pointed to by the X index register.  The buffer must be at least   *
*        14 bytes long to contain the ASCII conversion.  The resulting      *
*        ASCII string is terminated by a zero (0) byte.  Upon exit the      *
*        X Index register will be pointing to the first character of the    *
*        string.  FPACC1 and FPACC2 will remain unchanged.                  *
*                                                                           *
*****************************************************************************
*
*
FLTASC
         PSHX                               ;SAVE THE POINTER TO THE STRING
BUFFER.
         LDX      #FPACC1EX                 ;POINT TO FPACC1.
         JSR      CHCK0                     ;IS FPACC1 0?
         BNE      FLTASC1                   ;NO. GO CONVERT THE NUMBER.
         PULX                               ;RESTORE POINTER.
         LDD      #$3000                    ;GET ASCII CHARACTER + TERMINATING
BYTE.
         STD      0,X                       ;PUT IT IN THE BUFFER.
         RTS                                ;RETURN.

FLTASC1
         LDX      FPACC1EX          ;SAVE FPACC1.
         PSHX
         LDX      FPACC1MN+1
         PSHX
         LDAA     MANTSGN1
         PSHA
         JSR      PSHFPAC2          ;SAVE FPACC2.
         LDX      #0
         PSHX                               ;ALLOCATE LOCALS.
         PSHX
         PSHX                               ;SAVE SPACE FOR STRING BUFFER
POINTER.
         TSY                                ;POINT TO LOCALS.
         LDX      15,Y                      ;GET POINTER FROM STACK.
```

```
         LDAA     #$20                    ;PUT A SPACE IN THE BUFFER IF
NUMBER NOT NEGATIVE.
         TST      MANTSGN1          ;IS IT NEGATIVE?
         BEQ      FLTASC2                 ;NO. GO PUT SPACE.
         CLR      MANTSGN1          ;MAKE NUMBER POSITIVE FOR REST OF
CONVERSION.
         LDAA     #'-'                    ;YES. PUT MINUS SIGN IN BUFFER.

FLTASC2
         STAA     0,X
         INX                              ;POINT TO NEXT LOCATION.
         STX      0,Y                     ;SAVE POINTER.

FLTASC5
         LDX      #N9999999               ;POINT TO CONSTANT 9999999.
         JSR      GETFPAC2          ;GET INTO FPACC2.
         JSR      FLTCMP                  ;COMPARE THE NUMBERS. IS FPACC1 >
9999999?
         BHI      FLTASC3                 ;YES. GO DIVIDE FPACC1 BY 10.
         LDX      #P9999999               ;POINT TO CONSTANT 999999.9
         JSR      GETFPAC2          ;MOVE IT INTO FPACC2.
         JSR      FLTCMP                  ;COMPARE NUMBERS. IS FPACC1 >
999999.9?
         BHI      FLTASC4                 ;YES. GO CONTINUE THE CONVERSION.
         DEC      2,Y                     ;DECREMENT THE MULT./DIV. COUNT.
         LDX      #CONST10          ;NO. MULTIPLY BY 10. POINT TO CONSTANT.

FLTASC6
         JSR      GETFPAC2          ;MOVE IT INTO FPACC2.
         JSR      FLTMUL
         BRA      FLTASC5                 ;GO DO COMPARE AGAIN.

FLTASC3
         INC      2,Y                     ;INCREMENT THE MULT./DIV. COUNT.
         LDX      #CONSTP1          ;POINT TO CONSTANT ".1".
         BRA      FLTASC6                 ;GO DIVIDE FPACC1 BY 10.

FLTASC4
         LDX      #CONSTP5          ;POINT TO CONSTANT OF ".5".
         JSR      GETFPAC2          ;MOVE IT INTO FPACC2.
         JSR      FLTADD                  ;ADD .5 TO NUMBER IN FPACC1 TO
ROUND IT.
         LDAB     FPACC1EX          ;GET FPACC1 EXPONENT.
         SUBB     #$81                    ;TAKE OUT BIAS +1.
         NEGB                             ;MAKE IT NEGATIVE.
         ADDB     #23                     ;ADD IN THE NUMBER OF MANTISSA BITS
-1.
         BRA      FLTASC17          ;GO CHECK TO SEE IF WE NEED TO SHIFT AT ALL.

FLTASC7
         LSR      FPACC1MN          ;SHIFT MANTISSA TO THE RIGHT BY THE RESULT
(MAKE
         ROR      FPACC1MN+1              ;THE NUMBER AN INTEGER).
         ROR      FPACC1MN+2
         DECB                             ;DONE SHIFTING?

FLTASC17
         BNE      FLTASC7                 ;NO. KEEP GOING.
         LDAA     #1                      ;GET INITIAL VALUE OF "DIGITS AFTER
D.P." COUNT.
         STAA     3,Y                     ;INITIALIZE IT.
         LDAA     2,Y                     ;GET DECIMAL EXPONENT.
```

```
         ADDA     #8                          ;ADD THE NUMBER OF DECIMAL +1 TO
THE EXPONENT.
                                             ;WAS THE ORIGINAL NUMBER > 9999999?
         BMI      FLTASC8                     ;YES. MUST BE REPRESENTED IN
SCIENTIFIC NOTATION.
         CMPA     #8                          ;WAS THE ORIGINAL NUMBER < 1?
         BHS      FLTASC8                     ;YES. MUST BE REPRESENTED IN
SCIENTIFIC NOTATION.
         DECA                                 ;NO. NUMBER CAN BE REPRESENTED IN 7
DIGITS.
         STAA     3,Y                         ;MAKE THE DECIMAL EXPONENT THE
DIGIT COUNT BEFORE
                                             ;THE DECIMAL POINT.
         LDAA     #2                          ;SETUP TO ZERO THE DECIMAL
EXPONENT.

FLTASC8
         SUBA     #2                          ;SUBTRACT 2 FROM THE DECIMAL
EXPONENT.
         STAA     2,Y                         ;SAVE THE DECIMAL EXPONENT.
         TST      3,Y                         ;DOES THE NUMBER HAVE AN INTEGER
PART? (EXP. >0)
         BGT      FLTASC9                     ;YES. GO PUT IT OUT.
         LDAA     #'.'                        ;NO. GET DECIMAL POINT.
         LDX      0,Y                         ;GET POINTER TO BUFFER.
         STAA     1,X+                        ;PUT THE DECIMAL POINT IN THE
BUFFER, POINT TO NEXT BUFFER LOCATION.
         TST      3,Y                         ;IS THE DIGIT COUNT TILL EXPONENT
=0?
         BEQ      FLTASC18          ;NO. NUMBER IS <.1
         LDAA     #'0'                        ;YES. FORMAT NUMBER AS .0XXXXXXX
         STAA     1,X+                        ;PUT THE 0 IN THE BUFFER, POINT TO
THE NEXT LOCATION.

FLTASC18
         STX      0,Y                         ;SAVE NEW POINTER VALUE.

FLTASC9
         LDX      #DECDIG                     ;POINT TO THE TABLE OF DECIMAL
DIGITS.
         LDAA     #7                          ;INITIALIZE THE THE NUMBER OF
DIGITS COUNT.
         STAA     5,Y

FLTASC10
         CLR      4,Y                         ;CLEAR THE DECIMAL DIGIT
ACCUMULATOR.

FLTASC11
         LDD      FPACC1MN+1                  ;GET LOWER 16 BITS OF MANTISSA.
         SUBD     1,X                         ;SUBTRACT LOWER 16 BITS OF
CONSTANT.
         STD      FPACC1MN+1                  ;SAVE RESULT.
         LDAA     FPACC1MN          ;GET UPPER 8 BITS.
         SBCA     0,X                         ;SUBTRACT UPPER 8 BITS.
         STAA     FPACC1MN          ;SAVE RESULT. UNDERFLOW?
         BCS      FLTASC12          ;YES. GO ADD DECIMAL NUMBER BACK IN.
         INC      4,Y                         ;ADD 1 TO DECIMAL NUMBER.
         BRA      FLTASC11          ;TRY ANOTHER SUBTRACTION.

FLTASC12
         LDD      FPACC1MN+1                  ;GET FPACC1 MANTISSA LOW 16 BITS.
         ADDD     1,X                         ;ADD LOW 16 BITS BACK IN.
```

```
        STD     FPACC1MN+1              ;SAVE THE RESULT.
        LDAA    FPACC1MN        ;GET HIGH 8 BITS.
        ADCA    0,X                     ;ADD IN HIGH 8 BITS OF CONSTANT.
        STAA    FPACC1MN        ;SAVE RESULT.
        LDAA    4,Y                     ;GET DIGIT.
        ADDA    #$30                    ;MAKE IT ASCII.
        PSHX                            ;SAVE POINTER TO CONSTANTS.
        LDX     0,Y                     ;GET POINTER TO BUFFER.
        STAA    1,X+                    ;PUT DIGIT IN BUFFER, POINT TO NEXT
BUFFER LOCATION.
        DEC     3,Y                     ;SHOULD WE PUT A DECIMAL POINT IN
THE BUFFER YET?
        BNE     FLTASC16        ;NO. CONTINUE THE CONVERSION.
        LDAA    #'.'                    ;YES. GET DECIMAL POINT.
        STAA    1,X+                    ;PUT IT IN THE BUFFER, POINT TO THE
NEXT BUFFER LOCATION.

FLTASC16
        STX     0,Y                     ;SAVE UPDATED POINTER.
        PULX                            ;RESTORE POINTER TO CONSTANTS.
        LEAX    3,X                     ;POINT TO NEXT CONSTANT.
        DEC     5,Y                     ;DONE YET?
        BNE     FLTASC10        ;NO. CONTINUE CONVERSION OF "MANTISSA".
        LDX     0,Y                     ;YES. POINT TO BUFFER STRING
BUFFER.

FLTASC13
        DEX                             ;POINT TO LAST CHARACTER PUT IN THE
BUFFER.
        LDAA    0,X                     ;GET IT.
        CMPA    #$30                    ;WAS IT AN ASCII 0?
        BEQ     FLTASC13        ;YES. REMOVE TRAILING ZEROS.
        INX                             ;POINT TO NEXT AVAILABLE LOCATION
IN BUFFER.
        LDAB    2,Y                     ;DO WE NEED TO PUT OUT AN EXPONENT?
        BEQ     FLTASC15        ;NO. WE'RE DONE.
        LDAA    #'E'                    ;YES. PUT AN 'E' IN THE BUFFER.
        STAA    1,X+                    ;POINT TO NEXT BUFFER LOCATION.
        LDAA    #'+'                    ;ASSUME EXPONENT IS POSITIVE.
        STAA    0,X                     ;PUT PLUS SIGN IN THE BUFFER.
        TSTB                            ;IS IT REALLY MINUS?
        BPL     FLTASC14        ;NO. IS'S OK AS IS.
        NEGB                            ;YES. MAKE IT POSITIVE.
        LDAA    #'-'                    ;PUT THE MINUS SIGN IN THE BUFFER.
        STAA    0,X

FLTASC14
        INX                             ;POINT TO NEXT BUFFER LOCATION.
        STX     0,Y                     ;SAVE POINTER TO STRING BUFFER.
        CLRA                            ;SET UP FOR DIVIDE.
        LDX     #10                     ;DIVIDE DECIMAL EXPONENT BY 10.
        IDIV
        PSHB                            ;SAVE REMAINDER.
        XGDX                            ;PUT QUOTIENT IN D.
        ADDB    #$30                    ;MAKE IT ASCII.
        LDX     0,Y                     ;GET POINTER.
        STAB    1,X+                    ;PUT NUMBER IN BUFFER, POINT TO
NEXT LOCATION.
        PULB                            ;GET SECOND DIGIT.
        ADDB    #$30                    ;MAKE IT ASCII.
        STAB    1,X+                    ;PUT IT IN THE BUFFER, POINT TO
NEXT LOCATION.
```

```
FLTASC15
         CLR      0,X                     ;TERMINATE STRING WITH A ZERO BYTE.
         LEAS     6,SP                    ;CLEAR LOCALS FROM STACK.
         JSR      PULFPAC2          ;RESTORE FPACC2.
         PULA
         STAA     MANTSGN1
         PULX                             ;RESTORE FPACC1.
         STX      FPACC1MN+1
         PULX
         STX      FPACC1EX
         PULX                             ;POINT TO THE START OF THE ASCII
STRING.
         RTS                              ;RETURN.
*
*
DECDIG
         dc.b     $0F,$42,$40             ;DECIMAL 1,000,000
         dc.b     $01,$86,$A0             ;DECIMAL 100,000
         dc.b     $00,$27,$10             ;DECIMAL 10,000
         dc.b     $00,$03,$E8             ;DECIMAL 1,000
         dc.b     $00,$00,$64             ;DECIMAL 100
         dc.b     $00,$00,$0A             ;DECIMAL 10
         dc.b     $00,$00,$01             ;DECIMAL 1
*
*

P9999999
         dc.b     $94,$74,$23,$FE         ;CONSTANT 999999.9
*

N9999999
         dc.b     $98,$18,$96,$7F         ;CONSTANT 9999999.

*
CONSTP5
         dc.b     $80,$00,$00,$00         ;CONSTANT .5
*
*
FLTCMP
         TST      MANTSGN1          ;IS FPACC1 NEGATIVE?
         BPL      FLTCMP2.1                ;NO. CONTINUE WITH COMPARE.
         TST      MANTSGN2          ;IS FPACC2 NEGATIVE?
         BPL      FLTCMP2.1                ;NO. CONTINUE WITH COMPARE.
         LDD      FPACC2EX          ;YES. BOTH ARE NEGATIVE SO COMPARE MUST BE
DONE
         CPD      FPACC1EX          ;BACKWARDS. ARE THEY EQUAL SO FAR?
         BNE      FLTCMP1.1                ;NO. RETURN WITH CONDITION CODES
SET.
         LDD      FPACC2MN+1               ;YES. COMPARE LOWER 16 BITS OF
MANTISSAS.
         CPD      FPACC1MN+1

FLTCMP1.1
         RTS                              ;RETURN WITH CONDITION CODES SET.

FLTCMP2.1
         LDAA     MANTSGN1          ;GET FPACC1 MANTISSA SIGN.
         CMPA     MANTSGN2          ;BOTH POSITIVE?
         BNE      FLTCMP1.1                ;NO. RETURN WITH CONDITION CODES
SET.
         LDD      FPACC1EX          ;GET FPACC1 EXPONENT & UPPER 8 BITS OF
MANTISSA.
         CPD      FPACC2EX          ;SAME AS FPACC2?
```

```
        BNE     FLTCMP1.1               ;NO. RETURN WITH CONDITION CODES
SET.
        LDD     FPACC1MN+1              ;GET FPACC1 LOWER 16 BITS OF
MANTISSA.
        CPD     FPACC2MN+1              ;COMPARE WITH FPACC2 LOWER 16 BITS
OF MANTISSA.
        RTS                             ;RETURN WITH CONDITION CODES SET.


*************************************************************************
*                                                                       *
*                   UNSIGNED INTEGER TO FLOATING POINT                  *
*                                                                       *
*       This subroutine performs "unsigned" integer to floating point   *
*       conversion of a 16 bit word.  The 16 bit integer must be in the  *
*       lower 16 bits of FPACC1 mantissa.  The resulting floating point  *
*       number is returned in FPACC1.                                    *
*                                                                       *
*************************************************************************
*
*
UINT2FLT
        LDX     #FPACC1EX               ;POINT TO FPACC1.
        CLR     1,X                     ;clear the upper 8-bits of the
mantissa. Changed 05/26/93 G.S.D.
        JSR     CHCK0                   ;IS IT ALREADY 0?
        BNE     UINTFLT1        ;NO. GO CONVERT.
        RTS                             ;YES. JUST RETURN.

UINTFLT1
        LDAA    #$98                    ;GET BIAS + NUMBER OF BITS IN
MANTISSA.
        STAA    FPACC1EX        ;INITIALIZE THE EXPONENT.
        JSR     FPNORM                  ;GO MAKE IT A NORMALIZED FLOATING
POINT VALUE.
        CLC                             ;NO ERRORS.
        RTS                             ;RETURN.
*
*
*
*************************************************************************
*                                                                       *
*                     SIGNED INTEGER TO FLOATING POINT                  *
*                                                                       *
*       This routine works just like the unsigned integer to floating    *
*       point routine except the the 16 bit integer in the FPACC1        *
*       mantissa is considered to be in two's complement format.  This   *
*       will return a floating point number in the range -32768 to +32767.  *
*                                                                       *
*************************************************************************
*
*
SINT2FLT
        CLR     MANTSGN1        ;initialize the FPACC1 mantissa sign to zero
(positive) Added 12/17/91 G.S.D.
        CLR     FPACC1MN        ;Clear the upper 8-bits of the FPACC1
mantissa. Added 12/17/91 G.S.D.
        LDD     FPACC1MN+1              ;GET THE LOWER 16 BITS OF FPACC1
MANTISSA.
        PSHA                            ;SAVE SIGN OF NUMBER.
        BPL     SINTFLT1        ;IF POSITIVE JUST GO CONVERT.
        COMA                            ;MAKE POSITIVE.
        COMB
```

```
         ADDD    #1                      ;TWO'S COMPLEMENT.
         STD     FPACC1MN+1              ;PUT IT BACK IN FPACC1 MANTISSA.

SINTFLT1
         BSR     UINT2FLT        ;GO CONVERT.
         PULA                            ;GET SIGN OF ORIGINAL INTEGER.
         LDAB    #$FF                    ;GET "MINUS SIGN".
         TSTA                            ;WAS THE NUMBER NEGATIVE?
         BPL     SINTFLT2        ;NO. RETURN.
         STAB    MANTSGN1        ;YES. SET FPACC1 SIGN BYTE.

SINTFLT2
         CLC                             ;NO ERRORS.
         RTS                             ;RETURN.
*
*

*******************************************************************************
*                                                                             *
*                  FLOATING POINT TO INTEGER CONVERSION                       *
*                                                                             *
*        This subroutine will perform "unsigned" floating point to integer   *
*        conversion.  The floating point number if positive, will be          *
*        converted to an unsigned 16 bit integer ( 0 <= X <= 65535 ).  If     *
*        the number is negative it will be converted to a twos complement     *
*        16 bit integer.  This type of conversion will allow 16 bit           *
*        addresses to be represented as positive numbers when in floating     *
*        point format.  Any fractional number part is disguarded              *
*                                                                             *
*******************************************************************************
*
*
FLT2INT
         LDX     #FPACC1EX               ;POINT TO FPACC1.
         JSR     CHCK0                   ;IS IT 0?
         BEQ     FLT2INT3        ;YES. JUST RETURN.
         LDAB    FPACC1EX        ;GET FPACC1 EXPONENT.
         CMPB    #$81                    ;IS THERE AN INTEGER PART?
         BLO     FLT2INT2        ;NO. GO PUT A 0 IN FPACC1.
         TST     MANTSGN1        ;IS THE NUMBER NEGATIVE?
         BMI     FLT2INT1        ;YES. GO CONVERT NEGATIVE NUMBER.
         CMPB    #$90                    ;IS THE NUMBER TOO LARGE TO BE MADE
AN INTEGER?
         BHI     FLT2INT4        ;YES. RETURN WITH AN ERROR.
         SUBB    #$98                    ;SUBTRACT THE BIAS PLUS THE NUMBER
OF BITS.

FLT2INT5
         LSR     FPACC1MN        ;MAKE THE NUMBER AN INTEGER.
         ROR     FPACC1MN+1
         ROR     FPACC1MN+2
         INCB                            ;DONE SHIFTING?
         BNE     FLT2INT5        ;NO. KEEP GOING.
         CLR     FPACC1EX        ;ZERO THE EXPONENT (ALSO CLEARS THE CARRY).
         RTS

FLT2INT1
         CMPB    #$8F                    ;IS THE NUMBER TOO SMALL TO BE MADE
AN INTEGER?
         BHI     FLT2INT4        ;YES. RETURN ERROR.
         SUBB    #$98                    ;SUBTRACT BIAS PLUS NUMBER OF BITS.
         BSR     FLT2INT5        ;GO DO SHIFT.
         LDD     FPACC1MN+1              ;GET RESULTING INTEGER.
```

```
         COMA                                ;MAKE IT NEGATIVE.
         COMB
         ADDD      #1                        ;TWO'S COMPLEMENT.
         STD       FPACC1MN+1                ;SAVE RESULT.
         CLR       MANTSGN1          ;CLEAR MANTISSA SIGN. (ALSO CLEARS THE
CARRY)
         RTS                                 ;RETURN.

FLT2INT4
         LDAA      #TOLGSMER                 ;NUMBER TOO LARGE OR TOO SMALL TO
CONVERT TO INT.
         SEC                                 ;FLAG ERROR.
         RTS                                 ;RETURN.

FLT2INT2
         LDD     #0
         STD     FPACC1EX                    ;ZERO FPACC1.
         STD     FPACC1MN+1                  ;(ALSO CLEARS THE CARRY)

FLT2INT3
         RTS                                 ;RETURN.
*
*

*******************************************************************************
*                                                                             *
*                          SQUARE ROOT SUBROUTINE                             *
*                                                                             *
*        This routine is used to calculate the square root of the floating    *
*        point number in FPACC1.  If the number in FPACC1 is negative an      *
*        error is returned.                                                   *
*                                                                             *
*******************************************************************************
*
*
FLTSQR
         LDX       #FPACC1EX                 ;POINT TO FPACC1.
         JSR       CHCK0                     ;IS IT ZERO?
         BNE       FLTSQR1                   ;NO. CHECK FOR NEGATIVE.
         RTS                                 ;YES. RETURN.

FLTSQR1
         TST       MANTSGN1          ;IS THE NUMBER NEGATIVE?
         BPL       FLTSQR2                   ;NO. GO TAKE ITS SQUARE ROOT.
         LDAA      #NSQRTERR                 ;YES. ERROR.
         SEC                                 ;FLAG ERROR.
         RTS                                 ;RETURN.

FLTSQR2
         JSR       PSHFPAC2          ;SAVE FPACC2.
         LDAA      #4                        ;GET ITERATION LOOP COUNT.
         PSHA                                ;SAVE IT ON THE STACK.
         LDX       FPACC1MN+1                ;SAVE INITIAL NUMBER.
         PSHX
         LDX       FPACC1EX
         PSHX
         TSY                                 ;POINT TO IT.
         BSR       TFR1TO2                   ;TRANSFER FPACC1 TO FPACC2.
         LDAA      FPACC2EX          ;GET FPACC1 EXPONENT.
         SUBA      #$80                      ;REMOVE BIAS FROM EXPONENT.
         INCA                                ;COMPENSATE FOR ODD EXPONENTS
(GIVES CLOSER GUESS)
```

```
         BPL      FLTSQR3                   ;IF NUMBER >1 DIVIDE EXPONENT BY 2
& ADD BIAS.
         LSRA                               ;IF <1 JUST DIVIDE IT BY 2.
         BRA      FLTSQR4                   ;GO CALCULATE THE SQUARE ROOT.

FLTSQR3
         LSRA                               ;DIVIDE EXPONENT BY 2.
         ADDA     #$80                      ;ADD BIAS BACK IN.

FLTSQR4
         STAA     FPACC2EX          ;SAVE EXPONENT/2.

FLTSQR5
         JSR      FLTDIV                    ;DIVIDE THE ORIGINAL NUMBER BY THE
GUESS.
         JSR      FLTADD                    ;ADD THE "GUESS" TO THE QUOTIENT.
         DEC      FPACC1EX          ;DIVIDE THE RESULT BY 2 TO PRODUCE A NEW
GUESS.
         BSR      TFR1TO2                   ;PUT THE NEW GUESS INTO FPACC2.
         LDD      0,Y                       ;GET THE ORIGINAL NUMBER.
         STD      FPACC1EX          ;PUT IT BACK IN FPACC1.
         LDD      2,Y                       ;GET MANTISSA LOWER 16 BITS.
         STD      FPACC1MN+1
         DEC      4,Y                       ;BEEN THROUGH THE LOOP 4 TIMES?
         BNE      FLTSQR5                   ;NO. KEEP GOING.
         LDD      FPACC2EX          ;THE FINAL GUESS IS THE ANSWER.
         STD      FPACC1EX          ;PUT IT IN FPACC1.
         LDD      FPACC2MN+1
         STD      FPACC1MN+1
         leas     5,sp                      ;GET RID OF ORIGINAL NUMBER.
;                                           ;GET RID OF LOOP COUNT VARIABLE.
         JSR      PULFPAC2          ;RESTORE FPACC2.
         CLC                                ;NO ERRORS.
         RTS
*
*
TFR1TO2
         LDD      FPACC1EX          ;GET FPACC1 EXPONENT & HIGH 8 BIT OF
MANTISSA.
         STD      FPACC2EX          ;PUT IT IN FPACC2.
         LDD      FPACC1MN+1                ;GET FPACC1 LOW 16 BITS OF
MANTISSA.
         STD      FPACC2MN+1                ;PUT IT IN FPACC2.
         LDAA     MANTSGN1          ;TRANSFER THE SIGN.
         STAA     MANTSGN2
         RTS                                ;RETURN.
*
*

*********************************************************************************
*                                                                               *
*                        FLOATING POINT SINE                                    *
*                                                                               *
*********************************************************************************
*
*
FLTSIN
         JSR      PSHFPAC2          ;SAVE FPACC2 ON THE STACK.
         JSR      ANGRED                    ;GO REDUCE THE ANGLE TO BETWEEN +/-
PI.
         PSHD                               ;SAVE THE QUAD COUNT.
                                            ;SAVE THE SINE/COSINE FLAG.
```

```
         JSR      DEG2RAD                  ;CONVERT DEGREES TO RADIANS.
         PULA                              ;RESTORE THE SINE/COSINE FLAG.

FLTSIN1
         JSR      SINCOS                   ;GO GET THE SINE OF THE ANGLE.
         PULA                              ;RESTORE THE QUAD COUNT.
         CMPA     #2                       ;WAS THE ANGLE IN QUADS 1 OR 2?
         BLS      FLTSIN2                  ;YES. SIGN OF THE ANSWER IS OK.
         COM      MANTSGN1          ;NO. SINE IN QUADS 3 & 4 IS NEGATIVE.

FLTSIN2
         CLC                               ;SHOW NO ERRORS.
         JSR      PULFPAC2          ;RESTORE FPACC2
         RTS                               ;RETURN.

*
*
*************************************************************************
*                                                                       *
*                        FLOATING POINT COSINE                          *
*                                                                       *
*************************************************************************
*
*
FLTCOS
         JSR      PSHFPAC2          ;SAVE FPACC2 ON THE STACK.
         JSR      ANGRED                   ;GO REDUCE THE ANGLE TO BETWEEN +/-
PI.
         PSHD                              ;SAVE THE QUAD COUNT.
                                           ;SAVE THE SINE/COSINE FLAG.

         JSR      DEG2RAD                  ;CONVERT TO RADIANS.
         PULA                              ;RESTORE THE SINE/COSINE FLAG.
         EORA     #$01                     ;COMPLIMENT 90'S COPMLIMENT FLAG
FOR COSINE.

         JSR      SINCOS                   ;GO GET THE COSINE OF THE ANGLE.
         PULA                              ;RESTORE THE QUAD COUNT.
         CMPA     #1                       ;WAS THE ORIGINAL ANGLE IN QUAD 1?
         BEQ      FLTCOS1                  ;YES. SIGN IS OK.
         CMPA     #4                       ;WAS IT IN QUAD 4?
         BEQ      FLTCOS1                  ;YES. SIGN IS OK.
         COM      MANTSGN1          ;NO. COSINE IS NEGATIVE IN QUADS 2 & 3.

FLTCOS1
         JMP      FLTSIN2                  ;FLAG NO ERRORS, RESTORE FPACC2, &
RETURN.
*
*

*************************************************************************
*                                                                       *
*                 FLOATING POINT SINE AND COSINE SUBROUTINE             *
*                                                                       *
*************************************************************************
*
*
SINCOS
         PSHA                              ;SAVE SINE/COSINE FLAG ON STACK.
         LDX      FPACC1MN+1               ;SAVE THE VALUE OF THE ANGLE.
         PSHX

         LDX      FPACC1EX
```

```
        PSHX
        LDAA    MANTSGN1
        PSHA

        LDX     #SINFACT        ;POINT TO THE FACTORIAL TABLE.
        PSHX                            ;SAVE POINTER TO THE SINE FACTORIAL
TABLE.
        PSHX                            ;JUST ALLOCATE ANOTHER LOCAL (VALUE
NOT IMPORTANT)
        LDAA    #$4                     ;GET INITIAL LOOP COUNT.
        PSHA                            ;SAVE AS LOCAL ON STACK
        TSY                             ;POINT TO LOCALS.

        JSR     TFR1TO2                 ;TRANSFER FPACC1 TO FPACC2.
        JSR     FLTMUL                  ;GET X^2 IN FPACC1.
        TST     10,Y                    ;ARE WE DOING THE SINE?
        BEQ     SINCOS7                 ;YES. GO DO IT.

        LDX     #COSFACT        ;NO. GET POINTER TO COSINE FACTORIAL TABLE.
        STX     1,Y                     ;SAVE IT.
        JSR     TFR1TO2                 ;COPY X^2 INTO FPACC2.
        BRA     SINCOS4                 ;GENERATE EVEN POWERS OF "X" FOR
COSINE.

SINCOS7
        JSR     EXG1AND2        ;PUT X^2 IN FPACC2 & X IN FPACC1.

SINCOS1
        JSR     FLTMUL                  ;CREATE X^3,5,7,9 OR X^2,4,6,8.

SINCOS4
        LDX     FPACC1MN+1              ;SAVE EACH ONE ON THE STACK.
        PSHX
        LDX     FPACC1EX
        PSHX
        LDAA    MANTSGN1
        PSHA                            ;SAVE THE MANTISSA SIGN.
        DEC     0,Y                     ;HAVE WE GENERATED ALL THE POWERS
YET?
        BNE     SINCOS1                 ;NO. GO DO SOME MORE.
        LDAA    #$4                     ;SET UP LOOP COUNT.
        STAA    0,Y
        TSX                             ;POINT TO POWERS ON THE STACK.

SINCOS2
        STX     3,Y                     ;SAVE THE POINTER.
        LDX     1,Y                     ;GET THE POINTER TO THE FACTORIAL
CONSTANTS.
        JSR     GETFPAC2        ;PUT THE NUMBER IN FPACC2.

        leax    4,x                     ;POINT TO THE NEXT CONSTANT.
        STX     1,Y                     ;SAVE THE POINTER.
        LDX     3,Y                     ;GET POINTER TO POWERS.
        LDAA    0,X                     ;GET NUMBER SIGN.
        STAA    MANTSGN1        ;PUT IN FPACC1 MANTISSA SIGN.
        LDD     1,X                     ;GET LOWER 16-BITS OF THE MANTISSA.
        STD     FPACC1EX        ;PUT IN FPACC1 MANTISSA.
        LDD     3,X                     ;GET HIGH 8 BITS OF THE MANTISSA &
EXPONENT.
        STD     FPACC1MN+1              ;PUT IT IN FPACC1 EXPONENT &
MANTISSA.
        JSR     FLTMUL                  ;MULTIPLY THE TWO.
```

```
        LDX     3,Y                     ;GET POINTER TO POWERS BACK.
        LDD     FPACC1MN+1              ;SAVE RESULT WHERE THE POWER OF X
WAS.
        STD     3,X
        LDD     FPACC1EX
        STD     1,X
        LDAA    MANTSGN1        ;SAVE SIGN.
        STAA    0,X
        leax    5,x                     ;POINT TO THE NEXT POWER.
        DEC     0,Y                     ;DONE?
        BNE     SINCOS2                 ;NO. GO DO ANOTHER MULTIPLICATION.
        LDAA    #$3                     ;GET LOOP COUNT.
        STAA    0,Y                     ;SAVE IT.

SINCOS3
        LDX     3,Y                     ;POINT TO RESULTS ON THE STACK.
        leax    -5,x                    ;POINT TO PREVIOUS RESULT.
        STX     3,Y                     ;SAVE THE NEW POINTER.
        LDAA    0,X                     ;GET NUMBERS SIGN.
        STAA    MANTSGN2        ;PUT IT IN FPACC2.
        LDD     1,X                     ;GET LOW 16 BITS OF THE MANTISSA.
        STD     FPACC2EX        ;PUT IN FPACC2.
        LDD     3,X                     ;GET HIGH 8 BIT & EXPONENT.
        STD     FPACC2MN+1              ;PUT IN FPACC2.
        JSR     FLTADD                  ;GO ADD THE TWO NUMBERS.
        DEC     0,Y                     ;DONE?
        BNE     SINCOS3                 ;NO. GO ADD THE NEXT TERM IN.
        TST     10,Y                    ;ARE WE DOING THE SINE?
        BEQ     SINCOS5                 ;YES. GO PUT THE ORIGINAL ANGLE
INTO FPACC2.

        LDX     #ONE                    ;NO. FOR COSINE PUT THE CONSTANT 1
INTO FPACC2.
        JSR     GETFPAC2
        BRA     SINCOS6                 ;GO ADD IT TO THE SUM OF THE TERMS.

SINCOS5
        LDAA    5,Y                     ;GET THE VALUE OF THE ORIGINAL
ANGLE.
        STAA    MANTSGN2        ;PUT IT IN FPACC2.
        LDD     6,Y
        STD     FPACC2EX
        LDD     8,Y
        STD     FPACC2MN+1

SINCOS6
        JSR     FLTADD                  ;GO ADD IT TO THE SUM OF THE TERMS.
        TSX                             ;NOW CLEAN UP THE STACK.
        XGDX                            ;PUT STACK IN D.
        ADDD    #31                     ;CLEAR ALL THE TERMS & TEMPS OFF
THE STACK.
        XGDX
        TXS                             ;UPDATE THE STACK POINTER.
        RTS                             ;RETURN.
*
*
ANGRED
        CLRA                            ;INITIALIZE THE 45'S COMPLIMENT
FLAG.
        PSHA                            ;PUT IT ON THE STACK.
        INCA                            ;INITIALIZE THE QUAD COUNT TO 1.
        PSHA                            ;PUT IT ON THE STACK.
        TSY                             ;POINT TO IT.
```

```
        LDX     #THREE60        ;POINT TO THE CONSTANT 360.
        JSR     GETFPAC2        ;GET IT INTO FPACC.
        TST     MANTSGN1        ;IS THE INPUT ANGLE NEGATIVE:
        BPL     ANGRED1                 ;NO. SKIP THE ADD.
        JSR     FLTADD                  ;YES. MAKE THE ANGLE POSITIVE BY
ADDING 360 DEG.

ANGRED1
        DEC     FPACC2EX        ;MAKE THE CONSTANT IN FPACC2 90 DEGREES.
        DEC     FPACC2EX

ANGRED2
        JSR     FLTCMP                  ;IS THE ANGLE LESS THAN 90 DEGREES
ALREADY?
        BLS     ANGRED3                 ;YES. RETURN WITH QUAD COUNT.
        JSR     FLTSUB                  ;NO. REDUCE ANGLE BY 90 DEGREES.
        INC     0,Y                     ;INCREMENT THE QUAD COUNT.
        BRA     ANGRED2                 ;GO SEE IF IT'S LESS THAN 90 NOW.

ANGRED3
        LDAA    0,Y                     ;GET THE QUAD COUNT.
        CMPA    #1                      ;WAS THE ORIGINAL ANGLE IN QUAD 1?
        BEQ     ANGRED4                 ;YES. COMPUTE TRIG FUNCTION AS IS.
        CMPA    #3                      ;NO. WAS THE ORIGINAL ANGLE IN QUAD
3?
        BEQ     ANGRED4                 ;YES. COMPUTE THE TRIG FUNCTION AS
IF IN QUAD 1.
        LDAA    #$FF                    ;NO. MUST COMPUTE THE TRIG FUNCTION
OF THE 90'S
        STAA    MANTSGN1        ;COMPLIMENT ANGLE.
        JSR     FLTADD                  ;ADD 90 DEGREES TO THE NEGATED
ANGLE.

ANGRED4
        DEC     FPACC2EX        ;MAKE THE ANGLE IN FPACC2 45 DEGREES.
        JSR     FLTCMP                  ;IS THE ANGLE < 45 DEGREES?
        BLS     ANGRED5                 ;YES. IT'S OK AS IT IS.
        INC     FPACC2EX        ;NO. MUST GET THE 90'S COMPLIMENT.
        LDAA    #$FF                    ;MAKE FPACC1 NEGATIVE.
        STAA    MANTSGN1
        JSR     FLTADD                  ;GET THE 90'S COMPLIMENT.
        INC     1,Y                     ;SET THE FLAG.
ANGRED5
        pulb                            ;GET THE QUAD COUNT.
        pula                            ; GET THE COMPLIMENT FLAG.
        RTS                             ;RETURN WITH THE QUAD COUNT &
COMPLIMENT FLAG.
*
*
EXG1AND2
        LDD     FPACC1EX
        LDX     FPACC2EX
        STD     FPACC2EX
        STX     FPACC1EX
        LDD     FPACC1MN+1
        LDX     FPACC2MN+1
        STD     FPACC2MN+1
        STX     FPACC1MN+1
        LDAA    MANTSGN1
        LDAB    MANTSGN2
        STAA    MANTSGN2
        STAB    MANTSGN1
        RTS                             ;RETURN.
```

```
*
*
SINFACT
         dc.b     $6E,$38,$EF,$1D           ;+(1/9!)
         dc.b     $74,$D0,$0D,$01           ;-(1/7!)
         dc.b     $7A,$08,$88,$89           ;+(1/5!)
         dc.b     $7E,$AA,$AA,$AB           ;-(1/3!)
*
*
COSFACT
         dc.b     $71,$50,$0D,$01           ;+(1/8!)
         dc.b     $77,$B6,$0B,$61           ;-(1/6!)
         dc.b     $7C,$2A,$AA,$AB           ;+(1/4!)
         dc.b     $80,$80,$00,$00           ;-(1/2!)
*
*
ONE      dc.b     $81,$00,$00,$00           ;1.0
PI       dc.b     $82,$49,$0F,$DB           ;3.1415927
THREE60  dc.b     $89,$34,$00,$00           ;360.0
*
*

********************************************************************************
*                                                                              *
*                          FLOATING POINT TANGENT                              *
*                                                                              *
********************************************************************************
*
*
FLTTAN
         JSR      PSHFPAC2          ;SAVE FPACC2 ON THE STACK.
         JSR      TFR1TO2                   ;PUT A COPY OF THE ANGLE IN FPACC2.
         JSR      FLTCOS                    ;GET COSINE OF THE ANGLE.
         JSR      EXG1AND2          ;PUT RESULT IN FPACC2 & PUT ANGLE IN FPACC1.
         JSR      FLTSIN                    ;GET SIN OF THE ANGLE.
         JSR      FLTDIV                    ;GET TANGENT OF ANGLE BY DOING
SIN/COS.
         BCC      FLTTAN1                   ;IF CARRY CLEAR, ANSWER OK.
         LDX      #MAXNUM                   ;TANGENT OF 90 WAS ATTEMPTED. PUT
LARGEST
         JSR      GETFPAC1          ;NUMBER IN FPACC1.
         LDAA     #TAN90ERR                 ;GET ERROR CODE IN A.

FLTTAN1
         JSR      PULFPAC2          ;RESTORE FPACC2.
         RTS                                ;RETURN.
*
*
MAXNUM
         dc.b     $FE,$7F,$FF,$FF           ;LARGEST POSITIVE NUMBER WE CAN
HAVE.
*
*

********************************************************************************
*                                                                              *
*                              TRIG UTILITIES                                  *
*                                                                              *
*        The routines "DEG2RAD" and "RAD2DEG" are used to convert angles       *
*        from degrees-to-radians and radians-to-degrees respectively. The      *
*        routine "GETPI" will place the value of PI into FPACC1. This          *
*        routine should be used if the value of PI is needed in calculations   *
*        since it is accurate to the full 24-bits of the mantissa.             *
```

```
*                                                                              *
********************************************************************************
*
*
DEG2RAD
         JSR      PSHFPAC2          ;SAVE FPACC2.
         LDX      #PIOV180          ;POINT TO CONVERSION CONSTANT PI/180.

DEG2RAD1
         JSR      GETFPAC2          ;PUT IT INTO FPACC2.
         JSR      FLTMUL                    ;CONVERT DEGREES TO RADIANS.
         JSR      PULFPAC2          ;RESTORE FPACC2.
         RTS                                ;RETURN. (NOTE! DON'T REPLACE THE
"JSR/RTS" WITH
                                            ;;A "JMP" IT WILL NOT WORK.)
*
*
RAD2DEG
         JSR      PSHFPAC2          ;SAVE FPACC2.
         LDX      #C180OVPI                 ;POINT TO CONVERSION CONSTANT
180/PI.
         BRA      DEG2RAD1          ;GO DO CONVERSION & RETURN.
*
*
GETPI
         LDX      #PI                       ;POINT TO CONSTANT "PI".
         JMP      GETFPAC1          ;PUT IT IN FPACC1 AND RETURN.
*
*
PIOV180
         dc.b     $7B,$0E,$FA,$35
*
C180OVPI
         dc.b     $86,$65,$2E,$E1
*
*

********************************************************************************
*                                                                              *
*        The following two subroutines, PSHFPAC2 & PULPFAC2, push FPACC2       *
*        onto and pull FPACC2 off of the hardware stack respectively.          *
*        The number is stored in the "memory format".                          *
*                                                                              *
********************************************************************************
*
*
PSHFPAC2
         PULX                               ;GET THE RETURN ADDRESS OFF OF THE
STACK.
         PSHX                               ;ALLOCATE FOUR BYTES OF STACK
SPACE.
         PSHX
         XGDX                               ;PUT THE RETURN ADDRESS IN D.
         TSX                                ;POINT TO THE STORAGE AREA.
         PSHD                               ;PUT THE RETURN ADDRESS BACK ON THE
STACK.
         JMP      PUTFPAC2          ;GO PUT FPACC2 ON THE STACK & RETURN.
*
*
PULFPAC2
         TSX                                ;POINT TO THE RETURN ADDRESS.
         INX                                ;POINT TO THE SAVED NUMBER.
         INX
```

```
        JSR     GETFPAC2        ;RESTORE FPACC2.
        PULX                    ;GET THE RETURN ADDRESS OFF THE
STACK.
        leas    4,sp            ;REMOVE THE NUMBER FROM THE STACK.
        JMP     0,X             ;RETURN.
*
*

*******************************************************************************
*                                                                             *
*                           GETFPACx SUBROUTINE                               *
*                                                                             *
*       The GETFPAC1 and GETFPAC2 subroutines get a floating point number     *
*       stored in memory and put it into either FPACC1 or FPACC2 in a format  *
*       that is expected by all the floating point math routines. These       *
*       routines may easily be replaced to convert any binary floating point  *
*       format (i.e. IEEE format) to the format required by the math          *
*       routines.  The "memory" format converted by these routines is shown   *
*       below:                                                                *
*                                                                             *
*       31_______24 23 22_____________________0                               *
*        exponent   s         mantissa                                       *
*                                                                             *
*       The exponent is biased by 128 to facilitate floating point            *
*       comparisons.  The sign bit is 0 for positive numbers and 1            *
*       for negative numbers.  The mantissa is stored in hidden bit           *
*       normalized format so that 24 bits of precision can be obtained.       *
*       Since a normalized floating point number always has its most          *
*       significant bit set, we can use the 24th bit to hold the mantissa     *
*       sign.  This allows us to get 24 bits of precision in the mantissa     *
*       and store the entire number in just 4 bytes.  The format required by  *
*       the math routines uses a seperate byte for the sign, therfore each    *
*       floating point accumulator requires five bytes.                       *
*                                                                             *
*******************************************************************************
*
*
GETFPAC1
        LDD     0,X             ;GET THE EXPONENT & HIGH BYTE OF
THE MANTISSA,
        BEQ     GETFP12         ;IF NUMBER IS ZERO, SKIP SETTING
THE MS BIT.
        CLR     MANTSGN1        ;SET UP FOR POSITIVE NUMBER.
        TSTB                    ;IS NUMBER NEGATIVE?
        BPL     GETFP11         ;NO. LEAVE SIGN ALONE.
        COM     MANTSGN1        ;YES. SET SIGN TO NEGATIVE.

GETFP11
        ORAB    #$80            ;RESTORE MOST SIGNIFICANT BIT IN
MANTISSA.

GETFP12
        STD     FPACC1EX        ;PUT IN FPACC1.
        LDD     2,X             ;GET LOW 16-BITS OF THE MANTISSA.
        STD     FPACC1MN+1      ;PUT IN FPACC1.
        RTS                     ;RETURN.
*
*
GETFPAC2
        LDD     0,X             ;GET THE EXPONENT & HIGH BYTE OF
THE MANTISSA,
        BEQ     GETFP22         ;IF NUMBER IS 0, SKIP SETTING THE
MS BIT.
```

```
         CLR     MANTSGN2          ;SET UP FOR POSITIVE NUMBER.
         TSTB                      ;IS NUMBER NEGATIVE?
         BPL     GETFP21           ;NO. LEAVE SIGN ALONE.
         COM     MANTSGN2          ;YES. SET SIGN TO NEGATIVE.

GETFP21
         ORAB    #$80              ;RESTORE MOST SIGNIFICANT BIT IN
MANTISSA.

GETFP22
         STD     FPACC2EX          ;PUT IN FPACC1.
         LDD     2,X               ;GET LOW 16-BITS OF THE MANTISSA.
         STD     FPACC2MN+1        ;PUT IN FPACC1.
         RTS                       ;RETURN.
*
*

*******************************************************************************
*                                                                             *
*                        PUTFPACx SUBROUTINE                                  *
*                                                                             *
*       These two subroutines perform to opposite function of GETFPAC1 and    *
*       GETFPAC2. Again, these routines are used to convert from the          *
*       internal format used by the floating point package to a "memory"      *
*       format. See the GETFPAC1 and GETFPAC2, documentation for a            *
*       description of the "memory" format.                                   *
*                                                                             *
*******************************************************************************
*
*
PUTFPAC1
         LDD     FPACC1EX          ;GET FPACC1 EXPONENT & UPPER 8 BITS OF MANT.
         TST     MANTSGN1          ;IS THE NUMBER NEGATIVE?
         BMI     PUTFP11           ;YES. LEAVE THE M.S. BIT SET.
         ANDB    #$7F              ;NO. CLEAR THE M.S. BIT.

PUTFP11
         STD     0,X               ;SAVE IT IN MEMORY.
         LDD     FPACC1MN+1        ;GET L.S. 16 BITS OF THE MANTISSA.
         STD     2,X
         RTS                       ;RETURN.
*
*
PUTFPAC2
         LDD     FPACC2EX          ;GET FPACC1 EXPONENT & UPPER 8 BITS OF MANT.
         TST     MANTSGN2          ;IS THE NUMBER NEGATIVE?
         BMI     PUTFP21           ;YES. LEAVE THE M.S. BIT SET.
         ANDB    #$7F              ;NO. CLEAR THE M.S. BIT.

PUTFP21
         STD     0,X               ;SAVE IT IN MEMORY.
         LDD     FPACC2MN+1        ;GET L.S. 16 BITS OF THE MANTISSA.
         STD     2,X
         RTS                       ;RETURN.
*
*
*
FPACCCMP
         ldx     #FPACC1EX
         jsr     CHCK0             ; is fpacc1 zero?
         bne     FltCmp2           ; no. fpacc1 is not zero, however,
fpacc2 may be.
         ldx     #FPACC2EX
```

```
        jsr     CHCK0                   ; is fpacc2 zero?
        bne     FltCmp3                 ; no. but fpacc1 was zero.
        rts                             ; yes. both fpacc1 & fpacc2 were
zero.
                                        ; we can just return because the
last call to CHCK0 left the Z bit set.
;
FltCmp3                                 ; fpacc1 was zero, but fpacc2 was
not.
        tst     MANTSGN2        ; is fpacc2 negative?
        bne     FltCmp4                 ; yes. fpacc1 > fpacc2.

fp1ltfp2
        tpa                             ; no. fpacc1 < fpacc2.
        anda    #$f0                    ; clear arithmetic ccr bits.
        oraa    #$09                    ; set the carry & N-bit so that
signed branches will work.
        tap                             ; update the ccr.
        rts                             ; return.
;
FltCmp4                                 ; because fpacc2 is negative &
fpacc1 is zero...
        tpa                             ; fpacc1 > fpacc2.
        anda    #$f0                    ; clear arithmetic ccr bits.
        oraa    #$01                    ; set the carry flag & clear all
other ccr bits.
        tap                             ; update the ccr.
        rts
;
FltCmp2                                 ; fpacc1 is not zero.
        ldx     #FPACC2EX
        jsr     CHCK0                   ; is fpacc2 zero?
        bne     FltCmp6                 ; no. go check signs & do a
magnitude comparison.
        tst     MANTSGN1        ; yes. is fpacc1 negative?
        bne     FltCmp5                 ; yes. go set ccr for fpacc1 <
fpacc2.

fp1gtfp2
        tpa                             ; no. set ccr for fpacc1 > fpacc2.
        anda    #$f0                    ; clear arithmetic ccr bits.
        tap                             ; update the ccr.
        rts
;
FltCmp5                                 ; set ccr for fpacc1 < fpacc2.
        tpa
        anda    #$f0                    ; clear arithmetic ccr bits.
        oraa    #$08                    ; set the N-bit
        tap                             ; update the ccr.
        rts
;
;       at this point, neither fpacc is zero.
;

FltCmp6
        tst     MANTSGN1        ; is fpacc1 negative?
        beq     FltCmp10        ; no. but fpacc2 may be...
        tst     MANTSGN2        ; is fpacc2 negative?
        bne     FltCmp11        ; no. both numbers are negative. go compare
the magnatudes.
        bra     fp1ltfp2        ; fpacc1 < fpacc2
;
;       both numbers are negative.
```

```
;
FltCmp11
        ldab    #4                      ; count of the number of bytes to
compare.
        ldx     #FPACC1EX               ; point to fpacc1

CmpLoop1
        ldaa    0,x                     ; get a byte from fpacc1.
        cmpa    5,x                     ; compare it to the corresponding
byte in fpacc2.
        bhi     fp1ltfp2        ; branch if fpacc1 < fpacc2.
        blo     fp1gtfp2        ; branch if fpacc1 > fpacc2.
        inx                             ; if bytes are equal, point to the
next byte.
;       decb                            ; decrement the byte count.
        dbne    b,CmpLoop1              ; continue to compare if there are
more bytes to compare.

fp1eqfp2
        tpa                             ; fpacc1 = fpacc2.
        anda    #$f0                    ; clear arithmetic ccr bits.
        oraa    #$04                    ; set the Z-bit.
        tap                             ; update the ccr.
        rts
;
FltCmp10                        ; fpacc1 is positive
        tst     MANTSGN2        ; is fpacc2 negative?
        bne     fp1gtfp2        ; yes. fpacc1 > fpacc2
;
;       Both numbers are positive, compare magnatudes.
;
        ldab    #4                      ; count of the number of bytes to
compare.
        ldx     #FPACC1EX               ; point to fpacc1

CmpLoop2
        ldaa    0,x                     ; get a byte from fpacc1.
        cmpa    5,x                     ; compare it to the corresponding
byte in fpacc2.
        blo     fp1ltfp2        ; branch if fpacc1 < fpacc2.
        bhi     fp1gtfp2        ; branch if fpacc1 > fpacc2.
        inx                             ; if bytes are equal, point to the
next byte.

;       decb                            ; decrement the byte count.
        dbne    B,CmpLoop2              ; continue to compare if there are
more bytes to compare.
        bra     fp1eqfp2        ; fpacc1 = fpacc2. go set the proper ccr
bits.
```

Example Schematic: