```
MMM         MMM  TTTTTTTTTTTTTTT  HHH         HHH  RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
MMM         MMM  TTTTTTTTTTTTTTT  HHH         HHH  RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
MMM         MMM  TTTTTTTTTTTTTTT  HHH         HHH  RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
MMMMMM   MMMMMM        TTT        HHH         HHH  RRR         RRR        TTT        LLL
MMMMMM   MMMMMM        TTT        HHH         HHH  RRR         RRR        TTT        LLL
MMMMMM   MMMMMM        TTT        HHH         HHH  RRR         RRR        TTT        LLL
MMM   MMM   MMM        TTT        HHH         HHH  RRR         RRR        TTT        LLL
MMM   MMM   MMM        TTT        HHH         HHH  RRR         RRR        TTT        LLL
MMM   MMM   MMM        TTT        HHH         HHH  RRR         RRR        TTT        LLL
MMM         MMM        TTT        HHHHHHHHHHHHHHH  RRRRRRRRRRRR           TTT        LLL
MMM         MMM        TTT        HHHHHHHHHHHHHHH  RRRRRRRRRRRR           TTT        LLL
MMM         MMM        TTT        HHHHHHHHHHHHHHH  RRRRRRRRRRRR           TTT        LLL
MMM         MMM        TTT        HHH         HHH  RRR   RRR              TTT        LLL
MMM         MMM        TTT        HHH         HHH  RRR   RRR              TTT        LLL
MMM         MMM        TTT        HHH         HHH  RRR   RRR              TTT        LLL
MMM         MMM        TTT        HHH         HHH  RRR      RRR           TTT        LLL
MMM         MMM        TTT        HHH         HHH  RRR      RRR           TTT        LLL
MMM         MMM        TTT        HHH         HHH  RRR      RRR           TTT        LLL
MMM         MMM        TTT        HHH         HHH  RRR         RRR        TTT        LLLLLLLLLLLLLLL
MMM         MMM        TTT        HHH         HHH  RRR         RRR        TTT        LLLLLLLLLLLLLLL
MMM         MMM        TTT        HHH         HHH  RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

FILEID**MTHHSINCO

K 6

```
MM      MM  TTTTTTTTTT  HH      HH  HH      HH    SSSSSSSS    IIIIII    NN      NN    CCCCCCCC    OOOOOO
MM      MM  TTTTTTTTTT  HH      HH  HH      HH    SSSSSSSS    IIIIII    NN      NN    CCCCCCCC    OOOOOO
MMMM  MMMM      TT      HH      HH  HH      HH  SS              II      NN      NN  CC            OO      OO
MMMM  MMMM      TT      HH      HH  HH      HH  SS              II      NN      NN  CC            OO      OO
MM  MM  MM      TT      HH      HH  HH      HH  SS              II      NNNN    NN  CC            OO      OO
MM  MM  MM      TT      HH      HH  HH      HH  SS              II      NNNN    NN  CC            OO      OO
MM      MM      TT      HHHHHHHHHH  HHHHHHHHHH    SSSSSS        II      NN  NN  NN  CC            OO      OO
MM      MM      TT      HHHHHHHHHH  HHHHHHHHHH    SSSSSS        II      NN  NN  NN  CC            OO      OO
MM      MM      TT      HH      HH  HH      HH          SS      II      NN    NNNN  CC            OO      OO
MM      MM      TT      HH      HH  HH      HH          SS      II      NN    NNNN  CC            OO      OO
MM      MM      TT      HH      HH  HH      HH          SS      II      NN      NN  CC            OO      OO      ....
MM      MM      TT      HH      HH  HH      HH          SS      II      NN      NN  CC            OO      OO      ....
MM      MM      TT      HH      HH  HH      HH  SSSSSSSS      IIIIII    NN      NN    CCCCCCCC    OOOOOO        ....
MM      MM      TT      HH      HH  HH      HH  SSSSSSSS      IIIIII    NN      NN    CCCCCCCC    OOOOOO        ....

LL          IIIIII      SSSSSSSS
LL          IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000         .TITLE  MTH$HSINCOS      ; Floating Point Sine, Cosine and Sincos
0000                                  ;       Functions
0000         .IDENT /2-007/           ; File: MTHHSINCOS.MAR  EDIT: JCW2007
0000  ;
0000  ;****************************************************************************
0000  ;*                                                                          *
0000  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000  ;*  ALL RIGHTS RESERVED.                                                    *
0000  ;*                                                                          *
0000  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000  ;*  TRANSFERRED.                                                            *
0000  ;*                                                                          *
0000  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000  ;*  CORPORATION.                                                            *
0000  ;*                                                                          *
0000  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000  ;*                                                                          *
0000  ;*                                                                          *
0000  ;****************************************************************************
0000  ;
0000  ;
0000  ; FACILITY:     MATH LIBRARY
0000  ;++
0000  ; ABSTRACT:
0000  ;
0000  ; MTH$HSIN and MTH$HCOS are functions which return the floating point
0000  ; sine or cosine value of their single precision floating point argu-
0000  ; ment (radians). The call is standard call-by-reference.
0000  ; MTH$HSIN_R5 and MTH$HCOS_R5 are special routines which are the same
0000  ; as MTH$HSIN and MTH$HCOS  except  a faster non-standard JSB call is
0000  ; used with the argument in R0 and no registers are saved.
0000  ;
0000  ; MTH$HSINCOS  is  a routine which returns the floating point sine and
0000  ; cosine value of its single precision floating point radian argument.
0000  ; The call is standard call-by-reference.  MTH$HSINCOS_R5 is a special
0000  ; routine  which  is  the  same  as  MTH$HSINCOS, except a faster non-
0000  ; standard JSB call is used with the argument in R0 and no  registers
0000  ; are saved.
0000  ;
0000  ; MTH$HSIND and MTH$HCOSD are functions which return the floating point
0000  ; sine  or  cosine value of their single precision floating point argu-
0000  ; ment (degrees). The call is standard call-by-reference.
0000  ; MTH$HSIND_R5 and MTH$HCOSD_R5 are special routines which are the same
0000  ; as MTH$HSIND and MTH$HCOSD  except  a faster non-standard JSB call is
0000  ; used with the argument in R0 and no registers are saved.
0000  ;
0000  ; MTH$HSINCOSD is  a routine which returns the floating point sine and
0000  ; cosine value of its single precision floating point degree argument.
0000  ; The call is standard call-by-reference. MTH$HSINCOSD_R5 is a special
0000  ; routine  which  is  the  same  as MTH$HSINCOSD, except a faster non-
```

```
0000    ; standard JSB call is used with the argument in R0 and no  registers
0000    ; are saved.
0000    ;
0000    ;--
0000    ;
0000    ;--
0000    ;
0000    ; VERSION: 1
0000    ;
0000    ; HISTORY:
0000    ; AUTHOR:
0000    ;       John A. Wheeler, 21-Aug-1979: Version 1
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;
0000    ;
0000    ;
0000    ; VERSION: 2
0000    ;
0000    ; HISTORY:
0000    ; AUTHOR:
0000    ;       Bob Hanek, 28-Aug-1981: Version 2
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;
0000    ;
```

```
0000        .SBTTL  HISTORY ; Detailed Current Edit History
0000
0000  ;
0000  ; Edit History for Version 1 of MTH$HSINCOS
0000  ;
0000  ; 1-001 - Adapted from MTH$GSINCOS version 1-001.  JAW 21-Aug-1979
0000  ; 1-002 - Improve argument reduction scheme for greater accuracy.  Also
0000  ;         change MOVH to MOVO in MTH$HCOS.  JAW 28-Jan-1980
0000  ;
0000  ;
0000  ; Edit History for Version 2 of MTH$HSINCOS
0000  ;
0000  ; 2-001 - Changed W^ modifiers to G^ on external references RNH 09-Sep-81
0000  ; 2-002 - Changed MTH$AL_4_OV_PI to MTH$AL_4_OV_PI_V  RNH 29-Sep-81
0000  ; 2-003 - Included check for A2=0 before decrementing exponent.  RNH 02-Oct-81
0000  ; 2-004 - Included G^ modifiers on references to MTH$AL_4_OV_PI_V.
0000  ;         RNH 15-Oct-81
0000  ; 2-005 - Modified logic for converting reduced argument from integer to
0000  ;         floating point format to avoid modifying exponent of a floating
0000  ;         point zero.  RNH 21-Oct-81
0000  ; 2-006 - Modified REDUCE_LARGE logic to eliminate bug reported in QAR 896.
0000  ;         RNH 14-Jan-82
0000  ; 2-007 - In after returning from a JSB call to CVT_TO_H, the code in CONVERT:
0000  ;         did a SUBW #^X1D,16(R5), to change h_hi*2*29 to h_hi. However, this
0000  ;         only valid if the contents of 16(R5) is not zero. The code has been
0000  ;         patched to skip the above instruction if h_hi is zero.  JCW 22-SEP-83
```

```
                                         0000        .SBTTL  DECLARATIONS    -       Declarative Part of Module
                                         0000
                                         0000  ;
                                         0000  ; INCLUDE FILES:          MTHJACKET.MAR
                                         0000
                                         0000  ; EXTERNAL SYMBOLS:
                                         0000  ;
                                         0000        .DSABL  GBL
                                         0000        .EXTRN  MTH$AL_4_OV_PI_V
                                         0000        .EXTRN  MTH$$SIGNAL
                                         0000        .EXTRN  MTH$K_FLOUNDMAT
                                         0000        .EXTRN  MTH$$JACKET_TST
                                         0000  ;
                                         0000  ; EQUATED SYMBOLS:
                                         0000
                                0000C16C 0000        X_1_OV_45        = ^XC16C
                                         0000
                                         0000  ;
                                         0000  ; MACROS:
                                         0000
                                         0000        $SFDEF                          ; Define SF$ (stack frame) symbols
                                         0000        $PSLDEF                         ; Define PSL$ symbols
                                         0000
                                         0000  ; PSECT DECLARATIONS:
                                         0000
                                     00000000        .PSECT  _MTH$CODE       PIC,SHR,LONG,EXE,NOWRT
                                         0000                                ; program section for math routines
                                         0000  ;
                                         0000  ; OWN STORAGE:  none
                                         0000  ;
                                         0000  ; CONSTANTS:
                                         0000
                                         0000  H_PI_OV_4:
01B8C517 898C8469 42D1B544 921F4000      0000        .OCTA   ^X01B8C517898C846942D1B544921F4000      ; .7853981633974483096156608
                                         0010  H_9_PI_OV_4:
E1EF5DB9 BABEB4F6 CB2BABEC C4634003      0010        .OCTA   ^XE1EF5DB9BABEB4F6CB2BABECC4634003      ; .7068583470577034786540947
                                         0020  H_3_PI_OV_4:
414A93D1 2729234F 321DC7F3 2D974002      0020        .OCTA   ^X414A93D12729234F321DC7F32D974002      ; .2356194490192344928846982
                                         0030  H_5_PI_OV_4:
C226F65C EBEFE583 5385A295 F6A74002      0030        .OCTA   ^XC226F65CEBEFE5835385A295F6A74002      ; .3926990816987241548078304
                                         0040  H_7_PI_OV_4:
21812C74 585B53DC BA77BE9B 5FDB4003      0040        .OCTA   ^X21812C74585B53DCBA77BE9B5FDB4003      ; .5497787143782138167309625
                                         0050  H_2_OV_PI:
EA6AAFA3 F84E2A53 9C8806DC 45F34000      0050        .OCTA   ^XEA6AAFA3F84E2A539C8806DC45F34000      ; .6366197723675813430755350
                                         0060
                                         0060
                                         0060
                                         0060
                                         0060  H_M1:
00000000 00000000 00000000 0000C001      0060        .OCTA   ^X0000000000000000000000000000C001      ; -1
                                         0070  H_45:
00000000 00000000 00000000 68004006      0070        .OCTA   ^X00000000000000000000000068004006      ; .4500000000000000000000000
                                         0080  H_M45:
00000000 00000000 00000000 6800C006      0080        .OCTA   ^X0000000000000000000000006800C006      ;-.4500000000000000000000000
                                         0090  H_SMALLD:
81A5A6FE 152E7B86 3C1FC1A6 CA5D3FCD      0090        .OCTA   ^X81A5A6FE152E7B863C1FC1A6CA5D3FCD      ; .3975693351829395821628182
                                         00A0  H_1_OV_45:
```

```
6C1616C1 C16C6C16 16C1C16C 6C163FFB  00A0            .OCTA   ^X6C1616C1C16C6C1616C1C16C6C163FFB     ; .22222222222222222222222222
                                     00B0  H_CONVERT:
90B9CDD2 D8BE15C1 9D39A252 DF463FF7  00B0            .OCTA   ^X90B9CDD2D8BE15C19D39A252DF463FF7     ; .18282925199432957692369077
                                     00C0  H_90_OV_PI:
81A5A6FE 152E7B86 3C1FC1A6 CA5D4005  00C0            .OCTA   ^X81A5A6FE152E7B863C1FC1A6CA5D4005     ; .28647889756541160438399077
                                     00D0  H_SMALLEST_DEG:
81A5A6FE 152E7B86 3C1FC1A6 CA5D0006  00D0            .OCTA   ^X81A5A6FE152E7B863C1FC1A6CA5D0006     ; .48158580096997891364323077
                                     00E0
                                     00E0
                                     00E0  PI_OV_2:          ; Multiples of pi/2 in three pieces - High, medium, and low
                                     00E0  ; pi/2
01B8C517 898C8469 42D1B544 921F4001  00E0            .OCTA   ^X01B8C517898C846942D1B544921F4001     ; .15707963267948966192313217
000820BA C740A67C E0889024 CD123F8E  00F0            .OCTA   ^X000820BAC740A67CE0889024CD123F8E     ; .43359050650618905123985220
00012633 11A60B46 D511487E 00063F13  0100            .OCTA   ^X0001263311A60B46D511487E00063F13     ; .22641368202961808749714890
                                     0110  ; pi
01B8C517 898C8469 42D1B544 921F4002  0110            .OCTA   ^X01B8C517898C846942D1B544921F4002     ; .31415926535897932384626434
000420BB C740A67C E0889024 CD123F8F  0120            .OCTA   ^X000420BBC740A67CE0889024CD123F8F     ; .86718101301237810247970440
00014C66 234C168C AA2290FD 000C3F13  0130            .OCTA   ^X00014C66234C168CAA2290FD000C3F13     ; .22643538708856836580656990
                                     0140  ; 3*pi/2
414A93D1 2729234F 321DC7F3 2D974003  0140            .OCTA   ^X414A93D12729234F321DC7F32D974003     ; .47123889803846898576939650
8002188C 95707CDD A866EC1B 59CD3F90  0150            .OCTA   ^X8002188C95707CDDA866EC1B59CD3F90     ; .13007715195185671537195560
00017299 34F221D2 7F33D97C 00123F13  0160            .OCTA   ^X0001729934F221D27F33D97C00123F13     ; .22645709214751864411599090
                                     0170  ; 2*pi
01B8C517 898C8469 42D1B544 921F4003  0170            .OCTA   ^X01B8C517898C846942D1B544921F4003     ; .62831853071795864769252860
800220BB C740A67C E0889024 CD123F90  0180            .OCTA   ^X800220BBC740A67CE0889024CD123F90     ; .17343620260247562049594080
400198CC 46982D18 544421FB 00193F13  0190            .OCTA   ^X400198CC46982D18544421FB00193F13     ; .22647879720646892242541190
                                     01A0
```

```
                                            01A0
                                            01A0          .SBTTL  COEFFICIENT TABLES      -       Series Coefficients
                                            01A0
                                            01A0
                                            01A0
                                            01A0
                                            01A0  ;
                                            01A0  ; Polynomial Coefficient tables for arguments in radians
                                            01A0  ;
                                            01A0
                                            01A0  COSTBR1:              ; HCOS coefficients for arguments less than 1/2
4D97F522 D14447A3 118726F0 F1903FB1         01A0          .OCTA   ^X4D97F522D14447A3118726F0F1903FB1      ; C12 = 0.160771306639506109
3AE008A9 EF97457B 15E55D96 0CE3BFBB         01B0          .OCTA   ^X3AE008A9EF97457B15E55D960CE3BFBB      ; C11 = -.889676247791254258
8E83A5FF B3DD840A 29E1BA28 E5423FC3         01C0          .OCTA   ^X8E83A5FFB3DD840A29E1BA28E5423FC3      ; C10 = 0.411031761121448368
72DFE0F4 1183CAA5 949D863B 6827BFCC         01D0          .OCTA   ^X72DFE0F41183CAA5949D863B6827BFCC      ; C 9 = -.156192069685535803
36AAD1F8 435B5A5F 3B813E73 AE7F3FD4         01E0          .OCTA   ^X36AAD1F8435B5A5F3B813E73AE7F3FD4      ; C 8 = 0.477947733238737935
70AB98F3 D66FD1F8 07C94A8C 9397BFDC         01F0          .OCTA   ^X70AB98F3D66FD1F807C94A8C9397BFDC      ; C 7 = -.114707455977297247
C95102CD 4C0D7B54 8D898EFF 1EED3FE4         0200          .OCTA   ^XC95102CD4C0D7B548D898EFF1EED3FE4      ; C 6 = 0.208767569878680989
3F03A4A6 F016C72E 89F5FB77 27E4BFEB         0210          .OCTA   ^X3F03A4A6F016C72E89F5FB7727E4BFEB      ; C 5 = -.275573192239858906
ACB719FA 01A0A01A 1A0101A0 A01A3FF1         0220          .OCTA   ^XACB719FA01A0A01A1A0101A0A01A3FF1      ; C 4 = 0.248015873015873015
6B7B16C1 C16C6C16 16C1C16C 6C16BFF7         0230          .OCTA   ^X6B7B16C1C16C6C1616C1C16C6C16BFF7      ; C 3 = -.138888888888888888
55555555 55555555 55555555 55553FFC         0240          .OCTA   ^X5555555555555555555555555553FFC       ; C 2 = 0.416666666666666666
00000000 00000000 00000000 0000C000         0250          .OCTA   ^X0000000000000000000000000000C000      ; C 1 = -.500000000000000000
00000000 00000000 00000000 00004001         0260          .OCTA   ^X00000000000000000000000000004001      ; C 0 = 0.100000000000000000
                                  0000000D  0270  COSLENR1 = .-COSTBR1/16
                                            0270
                                            0270  COSTBR2:              ; HCOS coefficients for arguments greater than 1/2
CB3CA11F D05904F0 F958753F 86ABBFA8         0270          .OCTA   ^XCB3CA11FD05904F0F958753F86ABBFA8      ; C13 = -.246547279050635021
E58A4292 A30E00FA 924CD9E7 F2CC3FB1         0280          .OCTA   ^XE58A4292A30E00FA924CD9E7F2CC3FB1      ; C12 = 0.161171037087677335
BB0457F2 D240DFF7 074A963F 0CE3BFBB         0290          .OCTA   ^XBB0457F2D240DFF7074A963F0CE3BFBB      ; C11 = -.889679108386861988
7F2016A3 9B91DB57 AC0ABA3F E5423FC3         02A0          .OCTA   ^X7F2016A39B91DB57AC0ABA3FE5423FC3      ; C10 = 0.411031762308321448
36067A4A 2E4AE505 97BB863B 6827BFCC         02B0          .OCTA   ^X36067A4A2E4AE50597BB863B6827BFCC      ; C 9 = -.156192069685850600
59F70DCB 67DBE68B 3B813E73 AE7F3FD4         02C0          .OCTA   ^X59F70DCB67DBE68B3B813E73AE7F3FD4      ; C 8 = 0.477947733238738488
F302B98F 04ACD209 07C94A8C 9397BFDC         02D0          .OCTA   ^XF302B98F04ACD20907C94A8C9397BFDC      ; C 7 = -.114707455977297247
B50C0887 4D327B54 8D898EFF 1EED3FE4         02E0          .OCTA   ^XB50C08874D327B548D898EFF1EED3FE4      ; C 6 = 0.208767569878680989
A0BDB777 F016C72E 89F5FB77 27E4BFEB         02F0          .OCTA   ^XA0BDB777F016C72E89F5FB7727E4BFEB      ; C 5 = -.275573192239858906
DA0619F8 01A0A01A 1A0101A0 A01A3FF1         0300          .OCTA   ^XDA0619F801A0A01A1A0101A0A01A3FF1      ; C 4 = 0.248015873015873015
6A7816C1 C16C6C16 16C1C16C 6C16BFF7         0310          .OCTA   ^X6A7816C1C16C6C1616C1C16C6C16BFF7      ; C 3 = -.138888888888888888
55555555 55555555 55555555 55553FFC         0320          .OCTA   ^X5555555555555555555555555553FFC       ; C 2 = 0.416666666666666666
37B22605 3F4BD461 2E3E31A1 84923F83         0330          .OCTA   ^X37B226053F4BD4612E3E31A184923F83      ; C 1 = 0.178423084933010669
3CFD0792 48FB25CD 73ED76B9 390EBF7A         0340          .OCTA   ^X3CFD079248FB25CD73ED76B9390EBF7A      ; C 0 = -.280758778510761953
                                  0000000E  0350  COSLENR2 = .-COSTBR2/16
                                            0350
                                            0350  SINTBR:               ; HSIN coefficients
D4C61000 449213EB C8964962 CF1CBFA3         0350          .OCTA   ^XD4C61000449213EBC8964962CF1CBFA3      ; C13 = -.913324410322884674
22212038 09AF2C0D D7E285EA 3F3B3FAD         0360          .OCTA   ^X2221203809AF2C0DD7E285EA3F3B3FAD      ; C12 = 0.644684924220922497
84EC93C5 A89E2964 FD21406E 761BBFB6         0370          .OCTA   ^X84EC93C5A89E2964FD21406E761BBFB6      ; C11 = -.386817005082248997
9E96DF7A 78D985AA 8383EF6D 71B83FBF         0380          .OCTA   ^X9E96DF7A78D985AA8383EF6D71B83FBF      ; C10 = 0.195729410624566279
FB50B671 8EB1FC96 1400B468 2F49BFC8         0390          .OCTA   ^XFB50B6718EB1FC961400B4682F49BFC8      ; C 9 = -.822063524662382534
80BD7980 E3009EE0 0AD47703 952C3FD0         03A0          .OCTA   ^X80BD7980E3009EE00AD47703952C3FD0      ; C 8 = 0.281145725434552057
FC1BDC77 653BF11B 3B813E73 AE7FBFD8         03B0          .OCTA   ^XFC1BDC77653BF11B3B813E73AE7FBFD8      ; C 7 = -.764716373181981647
876C0B48 37CA97CA 86D0613A 61243FE0         03C0          .OCTA   ^X876C0B4837CA97CA86D0613A61243FE0      ; C 6 = 0.160590438368216145
12472F6E 747E38FE 544E567F AE64BFE7         03D0          .OCTA   ^X12472F6E747E38FE544E567FAE64BFE7      ; C 5 = -.250521083854417187
FC6588DF AC1C38FA 6C733A55 71DE3FEE         03E0          .OCTA   ^XFC6588DFAC1C38FA6C733A5571DE3FEE      ; C 4 = 0.275573192239858906
9EFA1A01 01A0A01A 1A0101A0 A01ABFF4         03F0          .OCTA   ^X9EFA1A0101A0A01A1A0101A0A01ABFF4      ; C 3 = -.198412698412698412
11111111 11111111 11111111 11113FFA         0400          .OCTA   ^X1111111111111111111111111113FFA       ; C 2 = 0.833333333333333333
55555555 55555555 55555555 5555BFFE         0410          .OCTA   ^X555555555555555555555555555BFFE       ; C 1 = -.166666666666666666
```

```
00000000 00000000 00000000 00000000  0420           .OCTA   ^X00000000000000000000000000000000      ; C 0 = 0.00000000000000000
                           0000000E  0430  SINLENR = .-SINTBR/16
                                     0430
                                     0430
                                     0430
                                     0430
                                     0430
                                     0430  ;
                                     0430  ; Polynomial coefficients for arguments in cycles
                                     0430  ;
                                     0430
                                     0430  COSTBC1:         ; HCOS coefficients for arguments less than 2/pi
6991E00D FF0BDBB5 E54BD2D7 82953FA9  0430           .OCTA   ^X6991E00DFF0BDBB5E54BD2D782953FA9      ; C12 = 0.487939182477553571
E3B856AF 4559F70A 207AF8FE 52ADBFB3  0440           .OCTA   ^XE3B856AF4559F70A207AF8FE52ADBFB3      ; C11 = -.437733098313663244
D774953A 9DD7AA9B F64C3074 EF6E3FBC  0450           .OCTA   ^XD774953A9DD7AA9BF64C3074EF6E3FBC      ; C10 = 0.327848355196045923
EC7C8084 0F585191 F584591A 2A0CBFC6  0460           .OCTA   ^XEC7C80840F585191F584591A2A0CBFC6      ; C 9 = -.201965339688246071
A5A51331 FFA8E578 2D7E2C2F 20C63FCF  0470           .OCTA   ^XA5A51331FFA8E5782D7E2C2F20C63FCF      ; C 8 = 0.100188646163627078
C5EECE27 4671F2C7 B1284F44 B6E2BFD7  0480           .OCTA   ^XC5EECE274671F2C7B1284F44B6E2BFD7      ; C 7 = -.389807317125967543
ADA79570 77F628DD 63CC8A37 F9D33FDF  0490           .OCTA   ^XADA7957077F628DD63CC8A37F9D33FDF      ; C 6 = 0.115011591279740515
EC993299 7B73BA9A 04A8F2A2 A6D1BFE7  04A0           .OCTA   ^XEC9932997B73BA9A04A8F2A2A6D1BFE7      ; C 5 = -.246113695049419975
A796B517 4E85AF46 1BAB0689 E1F53FEE  04B0           .OCTA   ^XA796B5174E85AF461BAB0689E1F53FEE      ; C 4 = 0.359086044859151007
1324BED9 4FAD9FC5 CBFFC7E3 55D3BFF5  04C0           .OCTA   ^X1324BED94FAD9FC5CBFFC7E355D3BFF5      ; C 3 = -.325991886927390013
AEDAE00F 33013B35 B5ACF081 03C13FFB  04D0           .OCTA   ^XAEDAE00F33013B35B5ACF08103C13FFB      ; C 2 = 0.158543442438155008
0118D9B3 ADC4E5A4 E45DCC9B 3BD3BFFF  04E0           .OCTA   ^X0118D9B3ADC4E5A4E45DCC9B3BD3BFFF      ; C 1 = -.308425137534042456
00000000 00000000 00000000 00004001  04F0           .OCTA   ^X00000000000000000000000000004001      ; C 0 = 0.100000000000000000
                           0000000D  0500  COSLENC1 = .-COSTBC1/16
                                     0500
                                     0500  COSTBC2:         ; HCOS coefficients for arguments greater than 2/pi
B4D349EB 38B2ABF9 23FD3D87 7678BF9F  0500           .OCTA   ^XB4D349EB38B2ABF923FD3D877678BF9F      ; C13 = -.461569527682893815
C0BA99A4 EA1CD7F6 34CBE2A0 838B3FA9  0510           .OCTA   ^XC0BA99A4EA1CD7F634CBE2A0838B3FA9      ; C12 = 0.489152360078513159
83E4A858 C1364D73 E8D6405B 52AEBFB3  0520           .OCTA   ^X83E4A858C1364D73E8D6405B52AEBFB3      ; C11 = -.437734505766522082
4A09FFED 7A9AEC82 F695308C EF6E3FBC  0530           .OCTA   ^X4A09FFED7A9AEC82F695308CEF6E3FBC      ; C10 = 0.327848356142723100
E4EE0516 EF67CB53 F818591A 2A0CBFC6  0540           .OCTA   ^XE4EE0516EF67CB53F818591A2A0CBFC6      ; C 9 = -.201965339688653120
A4E830BB CE4A437F 2D7F2C2F 20C63FCF  0550           .OCTA   ^XA4E830BBCE4A437F2D7F2C2F20C63FCF      ; C 8 = 0.100188646163627194
B2207548 DEE9F2D8 B1284F44 B6E2BFD7  0560           .OCTA   ^XB2207548DEE9F2D8B1284F44B6E2BFD7      ; C 7 = -.389807317125967543
BA8A2767 79FB28DD 63CC8A37 F9D33FDF  0570           .OCTA   ^XBA8A276779FB28DD63CC8A37F9D33FDF      ; C 6 = 0.115011591279740515
BFF34D7D 7B73BA9A 04A8F2A2 A6D1BFE7  0580           .OCTA   ^XBFF34D7D7B73BA9A04A8F2A2A6D1BFE7      ; C 5 = -.246113695049419975
8B09B515 4E85AF46 1BAB0689 E1F53FEE  0590           .OCTA   ^X8B09B5154E85AF461BAB0689E1F53FEE      ; C 4 = 0.359086044859151007
1230BED9 4FAD9FC5 CBFFC7E3 55D3BFF5  05A0           .OCTA   ^X1230BED94FAD9FC5CBFFC7E355D3BFF5      ; C 3 = -.325991886927390013
AEDAE00F 33013B35 B5ACF081 03C13FFB  05B0           .OCTA   ^XAEDAE00F33013B35B5ACF08103C13FFB      ; C 2 = 0.158543442438155008
08C2CD98 6E262D25 22EF64DF DE9EBFFC  05C0           .OCTA   ^X08C2CD986E262D2522EF64DFDE9EBFFC      ; C 1 = -.584251375340424568
3CFD0792 48FB25CD 73ED76B9 390EBF7A  05D0           .OCTA   ^X3CFD079248FB25CD73ED76B9390EBF7A      ; C 0 = -.280758778510761953
                           0000000E  05E0  COSLENC2 = .-COSTBC2/16
                                     05E0
                                     05E0  SINTBC:          ; HSIN coef for arg in cycles
50D605FA 71773F06 1C71B250 5CA4BF9A  05E0           .OCTA   ^X50D605FA71773F061C71B2505CA4BF9A      ; C13 = -.134292531932709184
F52F0A0E 2014A9C6 A852C91C 859A3FA4  05F0           .OCTA   ^XF52F0A0E2014A9C6A852C91C859A3FA4      ; C12 = 0.153671931033116475
F4D03ACB BC84F8E2 9226F7B9 7215BFAE  0600           .OCTA   ^XF4D03ACBBC84F8E29226F7B97215BFAE      ; C11 = -.149476468765572889
CFFC580F 400FA532 1601020D 28773FB8  0610           .OCTA   ^XCFFC580F400FA5321601020D28773FB8      ; C10 = 0.122614998471149687
0B4E3504 E93BD7BB F9394211 8A40BFC1  0620           .OCTA   ^X0B4E3504E93BD7BBF93942118A40BFC1      ; C 9 = -.834858983481116051
63388C8B DDA68A62 933532AF AAEC3FCA  0630           .OCTA   ^X63388C8BDDA68A62933532AFAAEC3FCA      ; C 8 = 0.462870462883468265
CEB7ADED 487E3913 5574B9F1 6FADBFD3  0640           .OCTA   ^XCEB7ADED487E39135574B9F16FADBFD3      ; C 7 = -.204102633966414405
1E02D09E CFC43328 18D634D0 E8F43FDB  0650           .OCTA   ^X1E02D09ECFC4332818D634D0E8F43FDB      ; C 6 = 0.694845327388662940
1F1FCCA4 BC80F623 88714FDE E307BFE3  0660           .OCTA   ^X1F1FCCA4BC80F62388714FDEE307BFE3      ; C 5 = -.175724767344340104
1DC845DA 7AC61B90 EE783487 50783FEB  0670           .OCTA   ^X1DC845DA7AC61B90EE78348750783FEB      ; C 4 = 0.313361689037812152
64ED7F64 5EE15BE6 2BD8CCE6 32D2BFF2  0680           .OCTA   ^X64ED7F645EE15BE62BD8CCE632D2BFF2      ; C 3 = -.365762041821772507
07A3449E 7E681D24 5AAEC677 466B3FF8  0690           .OCTA   ^X07A3449E7E681D245AAEC677466B3FF8      ; C 2 = 0.249039457019272016
```

```
55E55EE2 BB8E2BEA 5BE5CE62 4ABBBFFD  06A0          .OCTA   ^X55E55EE2BB8E2BEA5BE5CE624ABBBFFD      ; C 1 = -.807455121882807817
1B845170 98CC4698 2D185444 21FB3FFC  06B0          .OCTA   ^X1B84517098CC46982D18544421FB3FFC      ; C 0 = 0.353981633974483096
                           0000000E  06C0  SINLENC = .-SINTBC/16
                                     06C0
                                     06C0
                                     06C0
                                     06C0
                                     06C0
                                     06C0  ;
                                     06C0  ; Polynomial coefficients for arguments in degrees
                                     06C0  ;
                                     06C0
                                     06C0  COSDTB2:            ; HCOS coefficients for arguments less than 90/pi
EDD1ED9A 0B7862E5 5BDEDB5C BAB13F25  06C0          .OCTA   ^XEDD1ED9A0B7862E55BDEDB5CBAB13F25      ; C12 = 0.102627881393684166
089A882B 2A049560 B79DFC4B 7F7ABF3A  06D0          .OCTA   ^X089A882B2A049560B79DFC4B7F7ABF3A      ; C11 = -.186437848754295358
A905399E 6504B817 2CD74F4A 15553F4F  06E0          .OCTA   ^XA905399E6504B8172CD74F4A15553F4F      ; C10 = 0.282763099589940208
4F2A6144 EDA61DFA 4D10D808 49EFBF63  06F0          .OCTA   ^X4F2A6144EDA61DFA4D10D80849EFBF63      ; C 9 = -.352737470612942690
89E59A68 54663D0E DBD3994E 3C143F77  0700          .OCTA   ^X89E59A6854663D0EDBD3994E3C143F77      ; C 8 = 0.354338455375806404
EFD48542 F017D43A B92D60A8 DAFDBF8A  0710          .OCTA   ^XEFD48542F017D43AB92D60A8DAFDBF8A      ; C 7 = -.279173887526652390
15E4C0D3 D1683339 ED7D4688 0EA53F9E  0720          .OCTA   ^X15E4C0D3D1683339ED7D46880EA53F9E      ; C 6 = 0.166798233552852505
D0FF07DE E83BD1B0 3DC840ED BF62BFB0  0730          .OCTA   ^XD0FF07DEE83BD1B03DC840EDBF62BFB0      ; C 5 = -.722787516367020866
F165B451 C90A4160 ACEBB5C6 F83A3FC2  0740          .OCTA   ^XF165B451C90A4160ACEBB5C6F83A3FC2      ; C 4 = 0.213549430359498596
D4B76062 E75EBF27 CAD085BB 619BBFD4  0750          .OCTA   ^XD4B76062E75EBF27CAD085BB619BBFD4      ; C 3 = -.392583198574309488
FB521CA9 3606DF8A 3DC816A8 09B13FE5  0760          .OCTA   ^XFB521CA93606DF8A3DC816A809B13FE5      ; C 2 = 0.386632385156299365
CF6DA0A0 83BF98FC 41FB1DB1 3F6ABFF4  0770          .OCTA   ^XCF6DA0A083BF98FC41FB1DB13F6ABFF4      ; C 1 = -.152308709893354299
00000000 00000000 00000000 00004001  0780          .OCTA   ^X00000000000000000000000000004001      ; C 0 = 0.100000000000000000
                           0000000C  0790  COSDLN2 = .-COSDTB2/16 - 1
                                     0790
                                     0790  COSDTB1:            ; HCOS coefficients for arguments greater than 90/pi
9C90D303 82A12899 2C21F7D3 B1B0BF10  0790          .OCTA   ^X9C90D30382A128992C21F7D3B1B0BF10      ; C13 = -.479415169020911476
BA25F909 C348C900 08DCA1DB BBCB3F25  07A0          .OCTA   ^XBA25F909C348C90008DCA1DBBBCB3F25      ; C12 = 0.102883048126367050
476BE1A7 47C7AC09 3E244D1A 7F7BBF3A  07B0          .OCTA   ^X476BE1A747C7AC093E244D1A7F7BBF3A      ; C11 = -.186438448212011112
90F92FB1 95FC562C 9C4E4F57 15553F4F  07C0          .OCTA   ^X90F92FB195FC562C9C4E4F5715553F4F      ; C10 = 0.282763100406431640
42BC7FB3 86134211 4FEBD808 49EFBF63  07D0          .OCTA   ^X42BC7FB3861342114FEBD80849EFBF63      ; C 9 = -.352737470613653612
F05D81E4 4302A3F9 DBD3994E 3C143F77  07E0          .OCTA   ^XF05D81E44302A3F9DBD3994E3C143F77      ; C 8 = 0.354338455375806814
AD8A0182 FB21D44D B92D60A8 DAFDBF8A  07F0          .OCTA   ^XAD8A0182FB21D44DB92D60A8DAFDBF8A      ; C 7 = -.279173887526652390
8CA225BB D27D3339 ED7D4688 0EA53F9E  0800          .OCTA   ^X8CA225BBD27D3339ED7D46880EA53F9E      ; C 6 = 0.166798233552852505
8EA32452 E83BD1B0 3DC840ED BF62BFB0  0810          .OCTA   ^X8EA32452E83BD1B03DC840EDBF62BFB0      ; C 5 = -.722787516367020866
BBDCB44F C90A4160 ACEBB5C6 F83A3FC2  0820          .OCTA   ^XBBDCB44FC90A4160ACEBB5C6F83A3FC2      ; C 4 = 0.213549430359498596
D3BB6062 E75EBF27 CAD085BB 619BBFD4  0830          .OCTA   ^XD3BB6062E75EBF27CAD085BB619BBFD4      ; C 3 = -.392583198574309488
FB521CA9 3606DF8A 3DC816A8 09B13FE5  0840          .OCTA   ^XFB521CA93606DF8A3DC816A809B13FE5      ; C 2 = 0.386632385156299365
7B640506 1DFDC7E4 0FDCED8A FB50BFF1  0850          .OCTA   ^X7B6405061DFDC7E40FDCED8AFB50BFF1      ; C 1 = -.302383973933542996
3CFD0792 48FB25CD 73ED76B9 390EBF7A  0860          .OCTA   ^X3CFD079248FB25CD73ED76B9390EBF7A      ; C 0 = -.280758778510761953
                           0000000D  0870  COSDLN1 = .-COSDTB1/16 - 1
                                     0870
                                     0870  SINDTB:             ; HSIN coefficients
33D8819F 3CDAD3D6 569501D3 1F22BF06  0870          .OCTA   ^X33D8819F3CDAD3D6569501D31F22BF06      ; C13 = -.309965950876671388
D3264A07 C997C220 99208F9B 3D433F1B  0880          .OCTA   ^XD3264A07C997C22099208F9B3D433F1B      ; C12 = 0.718260038134094372
238953E7 964E4DB0 12621A10 29FCBF30  0890          .OCTA   ^X238953E7964E4DB012621A1029FCBF30      ; C11 = -.141476729812460641
D1DB62C7 A21FDA17 F4B457D1 D80D3F44  08A0          .OCTA   ^XD1DB62C7A21FDA17F4B457D1D80D3F44      ; C10 = 0.235007005013275394
08A210FA 884231C3 85001F14 365ABF59  08B0          .OCTA   ^X 8A210FA884231C385001F14365ABF59      ; C 9 = -.324022645124477933
2088C2FE E298DBCF 7C39F5FD 4C4B3F6D  08C0          .OCTA   ^X2088C2FEE298DBCF7C39F5FD4C4B3F6D      ; C 8 = 0.363786630161107676
F9570031 69E80E77 C70D4E6D 1AF8BF81  08D0          .OCTA   ^XF957003169E80E77C70D4E6D1AF8BF81      ; C 7 = -.324833568195494208
D058E528 37D66974 40432DDF 74143F94  08E0          .OCTA   ^XD058E52837D6697440432DDF74143F94      ; C 6 = 0.223936797077519653
9E29EAE3 EEA8F0D2 75261B38 6B71BFA7  08F0          .OCTA   ^X9E29EAE3EEA8F0D275261B386B71BFA7      ; C 5 = -.114682017753790161
6544545D B5305490 81C804CB F4A63FB9  0900          .OCTA   ^X6544545DB530549081C804CBF4A63FB9      ; C 4 = 0.414126741725732068
89CA7792 23061051 9509D9FA C368BFCB  0910          .OCTA   ^X89CA779223061051509D9FAC368BFCB       ; C 3 = -.978838486161772760
```

```
D906A253 E5120778 10D74EAE DAD93FDC  0920   370          .OCTA   ^XD906A253E512077810D74EAEDAD93FDC     ; C 2 = 0.134960162316325501
5226E950 660F8787 42F720D9 DBB8BFEC  0930   371          .OCTA   ^X5226E950660F878742F720D9DBB8BFEC     ; C 1 = -.886096155701298015
90B9CDD2 D8BE15C1 9D39A252 DF463FF7  0940   372          .OCTA   ^X90B9CDD2D8BE15C19D39A252DF463FF7     ; C 0 = 0.182829251994329576
                           0000000D  0950   373 SINDLN = .-SINDTB/16 - 1
                                     0950   374
```

```
0950            .SBTTL  MTH$HSINCOS     -       Radian arguments
0950
0950
0950    ;
0950    ; FUNCTIONAL DESCRIPTION:
0950    ;
0950    ; The HSIN, HCOS and HSINCOS routines are based on octant reduction.  Given an
0950    ; argument, x, it is written in the form
0950    ;
0950    ;               x = I1*(2*pi) + I*(pi/4) + Y1,
0950    ;
0950    ; where I1 and I are integers, 0 =< I < 8 and 0 =< Y1 < pi/4.  Since HSIN and
0950    ; HCOS have a period of 2*pi it follows that
0950    ;
0950    ;               HSIN(x) = HSIN(I*(pi/4) + Y1)) and
0950    ;               HCOS(x) = HCOS(I*(pi/4) + Y1)).
0950    ;
0950    ; Using the trigonometric identities for the sum and difference of two angles,
0950    ; the following table can be generated:
0950    ;
0950    ;       If I =              then HSIN(x) =              and HCOS(x) =
0950    ;       ------              -------------              -------------
0950    ;         0                   HSIN(Y1)                   HCOS(Y1)
0950    ;         1                   HCOS(pi/4-Y1)              HSIN(pi/4-Y1)
0950    ;         2                   HCOS(Y1)                  -HSIN(Y1)
0950    ;         3                   HSIN(pi/4-Y1)             -HCOS(pi/4-Y1)
0950    ;         4                  -HSIN(Y1)                  -HCOS(Y1)
0950    ;         5                  -HCOS(pi/4-Y1)             -HSIN(pi/4-Y1)
0950    ;         6                  -HCOS(Y1)                   HSIN(Y1)
0950    ;         7                  -HSIN(pi/4-Y1)              HCOS(pi/4-Y1)
0950    ;
0950    ; Let Y be defined as Y = Y1 if I is even and Y = pi/4 - Y1, if I is odd, then
0950    ; each entry of the above table is of the for +/-HSIN(Y) or +/-HCOS(Y).  Based
0950    ; on the above remarks, the HSIN, HCOS and HSINCOS routines process the input
0950    ; argument x, to obtain I and Y, and based on I selects a suitable polynomial
0950    ; approximation, p(Y), to evaluate the desired fuction.
```

```
                              0950
                              0950   ;
                              0950   ; INPUT PARAMETERS:
                              0950   ;
                   00000004   0950          LONG   = 4
                              0950
                   00000004   0950          x      = 1*LONG          ; x is input angle in radians
                   00000008   0950          sine   = 2*LONG          ; sine is HSIN(x)
                   0000000C   0950          cosine = 3*LONG          ; cosine is HCOS(x)
                              0950
                              0950
                              0950   ;
                              0950   ; Return sine and cosine of argument
                              0950   ;
                              0950
                              0950
                       00FC   0950          .ENTRY  MTH$HSINCOS, ^M<R2, R3, R4, R5, R6,R7>
                              0952
                              0952          MTH$FLAG_JACKET
                              0952
      6D    00000000'GF   9E  0952          MOVAB   G^MTH$$JACKET_HND, (FP)
                              0959                                   ; set handler address to jacket
                              0959                                   ; handler
                              0959
                              0959
         50    04 BC 70FD     0959          MOVH    @x(AP), R0
         000009F6'EF   16     095E          JSB     MTH$HSINCOS_R7
         08 BC    50 7DFD     0964          MOVO    R0, @sine(AP)
         0C BC    54 7DFD     0969          MOVO    R4, @cosine(AP)
                       04     096E          RET
                              096F
                              096F   ;
                              096F   ; INPUT PARAMETERS:
                              096F   ;
                   00000004   096F          LONG   = 4
                              096F
                   00000008   096F          arg    = 2*LONG          ; x is input angle in radians
                   00000004   096F          answer = 1*LONG
                              096F
                              096F          .SBTTL  MTH$HSIN
                              096F
                              096F   ;
                              096F   ; Return sine of argument
                              096F   ;
                              096F
                              096F
                       003C   096F          .ENTRY  MTH$HSIN, ^M<R2, R3, R4, R5>
                              0971
                              0971          MTH$FLAG_JACKET
                              0971
      6D    00000000'GF   9E  0971          MOVAB   G^MTH$$JACKET_HND, (FP)
                              0978                                   ; set handler address to jacket
                              0978                                   ; handler
                              0978
                              0978
         50    08 BC 70FD     0978          MOVH    @arg(AP), R0
         00000B3A'EF   16     097D          JSB     MTH$HSIN_R5
```

```
         04 BC    50 7DFD  0983          MOVO    R0, @answer(AP)
                       04  0988          RET
                           0989
                           0989
                           0989
                           0989          .SBTTL  MTH$HCOS
                           0989
                           0989  ;
                           0989  ; Return cosine of argument
                           0989  ;
                           0989
                           0989
                     003C  0989          .ENTRY  MTH$HCOS, ^M<R2, R3, R4, R5>
                           098B
                           098B          MTH$FLAG_JACKET
                           098B
    6D    00000000'GF   9E 098B          MOVAB   G^MTH$$JACKET_HND, (FP)
                           0992                                          ; set handler address to jacket
                           0992                                          ; handler
                           0992
                           0992
         50   08 BC 70FD   0992          MOVH    @arg(AP), R0
         00000BD3'EF    16 0997          JSB     MTH$HCOS_R5
         04 BC    50 7DFD  099D          MOVO    R0, @answer(AP)
                       04  09A2          RET
                           09A3
```

```
09A3          .SBTTL  MTH$HSINCOSD    -       Degrees
09A3
09A3
09A3  ;
09A3  ; FUNCTIONAL DESCRIPTION:
09A3  ;
09A3  ; The HSIND, HCOSD and HSINCOSD routines are based on octant reduction.  Given
09A3  ; an argument, x, it is written in the form
09A3  ;
09A3  ;               x = I1*360 + I*45 + Y1,
09A3  ;
09A3  ; where I1 and I are integers, 0 =< I < 8 and 0 =< Y1 < 45.  Since HSIND and
09A3  ; HCOSD have a period of 360 it follows that
09A3  ;
09A3  ;               HSIND(x) = HSIND(I*45 + Y1) and
09A3  ;               HCOSD(x) = HCOSD(I*45 + Y1).
09A3  ;
09A3  ; Using the trigonometric identities for the sum and difference of two angles,
09A3  ; the following table can be generated:
09A3  ;
09A3  ;       If I =              then HSIND(x) =                 and HCOSD(x) =
09A3  ;       ------              -------------                   -------------
09A3  ;         0                   HSIND(Y1)                       HCOSD(Y1)
09A3  ;         1                   HCOSD(45-Y1)                    HSIND(45-Y1)
09A3  ;         2                   HCOSD(Y1)                      -HSIND(Y1)
09A3  ;         3                   HSIND(45-Y1)                   -HCOSD(45-Y1)
09A3  ;         4                  -HSIND(Y1)                      -HCOSD(Y1)
09A3  ;         5                  -HCOSD(45-Y1)                   -HSIND(45-Y1)
09A3  ;         6                  -HCOSD(Y1)                       HSIND(Y1)
09A3  ;         7                  -HSIND(45-Y1)                    HCOS(45-Y1)
09A3  ;
09A3  ; Let Y be defined as Y = Y1 if I is even and Y = 45 - Y1, if I is odd, then
09A3  ; each entry of the above table is of the for +/-HSIN(Y) or +/-HCOS(Y).  Based
09A3  ; on the above remarks, the HSIND, HCOSD and HSINCOSD routines process the input
09A3  ; argument x, to obtain I and Y, and based on I selects a suitable polynomial
09A3  ; approximation, p(Y), to evaluate the desired fuction.
09A3  ;
09A3
```

```
                  00000008  09A3        sind = 2*LONG
                  0000000C  09A3        cosd = 3*LONG
                            09A3
                      00FC  09A3        .ENTRY  MTH$HSINCOSD    ^M<R2, R3, R4, R5, R6, R7>
                            09A5
                            09A5        MTH$FLAG_JACKET
                            09A5
      6D   00000000'GF  9E  09A5        MOVAB   G^MTH$$JACKET_HND, (FP)
                            09AC                                        ; set handler address to jacket
                            09AC                                        ; handler
                            09AC
                            09AC
       50    04 BC 70FD     09AC        MOVH    @x(AP), R0
       00000C7D'EF  16      09B1        JSB     MTH$HSINCOSD_R7
       08 BC     50 7DFD    09B7        MOVO    R0, @sind(AP)
       0C BC     54 7DFD    09BC        MOVO    R4, @cosd(AP)
                            09C1
                        04  09C1        RET
                            09C2
                            09C2
                            09C2
                      003C  09C2        .ENTRY  MTH$HSIND       ^M<R2, R3, R4, R5>
                            09C4
                            09C4        MTH$FLAG_JACKET
                            09C4
      6D   00000000'GF  9E  09C4        MOVAB   G^MTH$$JACKET_HND, (FP)
                            09CB                                        ; set handler address to jacket
                            09CB                                        ; handler
                            09CB
                            09CB
       50    08 BC 70FD     09CB        MOVH    @arg(AP), R0
       00000CEB'EF  16      09D0        JSB     MTH$HSIND_R5
       04 BC     50 7DFD    09D6        MOVO    R0, @answer(AP)
                            09DB
                        04  09DB        RET
                            09DC
                            09DC
                            09DC
                      003C  09DC        .ENTRY  MTH$HCOSD       ^M<R2, R3, R4, R5>
                            09DE
                            09DE        MTH$FLAG_JACKET
                            09DE
      6D   00000000'GF  9E  09DE        MOVAB   G^MTH$$JACKET_HND, (FP)
                            09E5                                        ; set handler address to jacket
                            09E5                                        ; handler
                            09E5
                            09E5
       50    08 BC 70FD     09E5        MOVH    @arg(AP), R0
       00000D5C'EF  16      09EA        JSB     MTH$HCOSD_R5
       04 BC     50 7DFD    09F0        MOVO    R0, @answer(AP)
                            09F5
                        04  09F5        RET
                            09F6
```

N 7

```
                          09F6          .SBTTL MTH$HSINCOS_R7
                          09F6
                          09F6   ; This routine computes the HSIN and HCOS of the G-format value of R0/R1.  The
                          09F6   ; computation is performed one of three ways depending on the size of the
                          09F6   ; input argument, X:
                          09F6   ;
                          09F6   ;      1) If |X| < pi/4, then X is used directly in polynomial approximation
                          09F6   ;         of HSIN and HCOS.
                          09F6   ;      2) If pi/4 =< |x| < 9*pi/4, then the subroutine REDUCE_MEDIUM is called
                          09F6   ;         to reduce the argument to an equivalent argument in radians, Y, and
                          09F6   ;         the octant, I, containing the argument.  Y is then evaluated in two
                          09F6   ;         polynomials chosen as a function of I, to compute HSIN(X) and HCOS(X).
                          09F6   ;      3) If 9*pi/4 =< |X|, then the subroutine REDUCE_LARGE is called to
                          09F6   ;         reduce the argument to an equivalent argument in cycles, Y, and the
                          09F6   ;         octant, I, contanining the argument.  Y is then evaluated in two
                          09F6   ;         polynomials chosen as a function of I, to compute HSIN(X) and HCOS(X).
                          09F6
                          09F6   MTH$HSINCOS_R7::
               50 73FD    09F6          TSTH    R0                      ; Check sign of x
               10   18    09F9          BGEQ    POS_SINCOS              ;
        00000A06'EF   16  09FB          JSB     SINCOS                  ; R0/R1 = HSIN(|X|), R2/R3 = HCOS(X)
            50   50 72FD  0A01          MNEGH   R0, R0                  ; R0/R1 = HSIN(X)
                    05    0A05          RSB
                          0A06
                          0A06   SINCOS:
  50     8000 8F    AA    0A06          BICW    #^X8000, R0             ; R0/R1 = |X|
                          0A0B   POS_SINCOS:
  50     F5F0 CF 71FD     0A0B          CMPH    H_PI_OV_4, R0           ; Compare pi/4 with |X|
               42   14    0A11          BGTR    SMALL_SINCOS            ; No argument reduction is necessary
  50     F5F8 CF 71FD     0A13          CMPH    H_9_PI_OV_4, R0         ; Compare 9*pi/4 with |X|
               03   18    0A19          BGEQ    1$
             00DE   31    0A1B          BRW     LARGE_SINCOS            ; Use special logic for |X| > 9*pi/4
                          0A1E
                          0A1E   ;
                          0A1E   ;   pi/4 =< |X| < 9*pi/4
                          0A1E   ;
          5E   30   C2    0A1E   1$:    SUBL    #48, SP                 ; Allocate 12 longword on the stack
                          0A21                                          ;   for the reduced argument
        00000DA3'EF   16  0A21          JSB     REDUCE_MEDIUM           ; Medium argument reduction routine
                          0A27                                          ; Y = reduced argument on the stack,
                          0A27                                          ;   R2 = octant, R5 points to YHI, and
                          0A27                                          ;   R4 points to YLO
            7E   64 7DFD  0A27          MOVO    (R4), -(SP)             ; Save YLO
            7E   65 7DFD  0A2B          MOVO    (R5), -(SP)             ; Save YHI
               34   BB    0A2F          PUSHR   #^M<R2, R4, R5>         ; Save R2, R4, R5
        00000BF4'EF   16  0A31          JSB     M_COS                   ; R0/R3 = HCOS(X)
         4C AE   50 7DFD  0A37          MOVO    R0, 76(SP)              ; Save HCOS(X) on stack
               34   BA    0A3C          POPR    #^M<R2, R4, R5>         ; Restore R2, R4, R5
            65   8E 7DFD  0A3E          MOVO    (SP)+, (R5)             ; Restore YHI
            64   8E 7DFD  0A42          MOVO    (SP)+, (R4)             ; Restore YLO
        00000B68'EF   16  0A46          JSB     M_SIN                   ; R0/R3 = HSIN(X)
         54   20 AE 7DFD  0A4C          MOVO    32(SP), R4              ; R4/R7 = HCOS(X)
          5E   30   C0    0A51          ADDL    #48, SP                 ; Pop 12 longwords off the stack
                    05    0A54          RSB
                          0A55   ;
                          0A55   ; Logic for small arguments.  |X| < pi/4.
                          0A55   ;
```

```
                            0A55
                            0A55      SMALL_SINCOS:
             7E    50 7DFD  0A55              MOVO    R0, -(SP)                 ; (SP) = |X|
    4000 8F    50      B1   0A59              CMPW    R0, #^X4000               ; Compare 1/2 with |X|
                 3B    18   0A5E              BGEQ    2$                        ; Sufficent overghang not available
    50    3FC7 8F      B1   0A60              CMPW    #^X3FC7, R0               ; Compare with 2^-57
                 2C    18   0A65              BGEQ    1$                        ; No polynomial evaluation is needed
    50      6E 6E    65FD   0A67              MULH3   (SP), (SP), R0            ; R0/R3 = X*X
             7E    50 7DFD  0A6C              MOVO    R0, -(SP)                 ; Put X*X on stack
 F729 CF    0C 50    75FD   0A70              POLYH   R0, #COSLENR1-1, COSTBR1; R0/R3 = HCOS(X)
             54    6E 7DFD  0A77              MOVO    (SP), R4                  ; R4/R7 = X*X
             6E    50 7DFD  0A7B              MOVO    R0, (SP)                  ; Save HCOS(X) on stack
 F8CA CF    0D 54    75FD   0A7F              POLYH   R4, #SINLENR-1, SINTBR    ; R0/R3 = q(X^2)
             54    8E 7DFD  0A86              MOVO    (SP)+, R4                 ; R4/R7 = HCOS(X)
             50    6E 64FD  0A8A              MULH2   (SP), R0                  ; R0/R3 = X*q(X^2)
             50    8E 60FD  0A8E              ADDH2   (SP)+, R0                 ; R0/R3 = HSIN(X)
                       05   0A92              RSB
                            0A93
          5E    10     C0   0A93      1$:     ADDL    #16, SP                   ; Clear stack
          54    08   70FD   0A96              MOVH    #1.0, R4                  ; R0/R3 = X, R4/R7 = 1.0 = HCOS(X)
                       05   0A9A              RSB
                            0A9B
                            0A9B
    54      6E 6E    65FD   0A9B      2$:     MULH3   (SP), (SP), R4            ; R4/R7 = X^2
             7E    54 7DFD  0AA0              MOVO    R4, -(SP)                 ; Save X^2
 F7C5 CF    0D 54    75FD   0AA4              POLYH   R4, #COSLENR2-1, COSTBR2; R0/R3 = Q(X^2)
             54    6E 7DFD  0AAB              MOVO    (SP), R4                  ; R4/R7 = X^2
             6E    50 7DFD  0AAF              MOVO    R0, (SP)                  ; Save Q(X^2) on stack
 F896 CF    0D 54    75FD   0AB3              POLYH   R4, #SINLENR-1, SINTBR    ; R0/R3 = Q(X^2)
             54    8E 7DFD  0ABA              MOVO    (SP)+, R4                 ; R4/R7 = P(X^2)
             50    6E 64FD  0ABE              MULH2   (SP), R0                  ; R0/R3 = X*P(X^2)
             50    6E 60FD  0AC2              ADDH2   (SP), R0                  ; R0/R3 = HSIN(X)
             7E    6E 7DFD  0AC6              MOVO    (SP), -(SP)               ; Save another copy of |X|
18 AE    FFFF01FF 8F   CA   0ACA              BICL    #^XFFFF01FF, 24(SP)       ;
             1C AE     D4   0AD2              CLRL    28(SP)                    ; 16(SP) = XHI
    7E    6E  10 AE  63FD   0AD5              SUBH3   16(SP) ,(SP), -(SP)       ; (SP) = XLO
    10 AE     20 AE  60FD   0ADB              ADDH2   32(SP), 16(SP)            ; 16(SP) = X + XHI
       6E        8E  64FD   0AE1              MULH2   (SP)+, (SP)               ; (SP) = XLO*(X + XHI) = A2
                 02    13   0AE5              BEQL    3$                        ; Check for A2 = 0
                 6E    B7   0AE7              DECW    (SP)                      ; (SP) = A2/2
             54    8E 62FD  0AE9      3$:     SUBH2   (SP)+, R4                 ; R4/R7 = Q(Y^2) - A2/2
             6E    6E 64FD  0AED              MULH2   (SP), (SP)                ; (SP) = XHI^2
                 6E    B7   0AF1              DECW    (SP)                      ; (SP) = XHI^2/2
             6E    08 62FD  0AF3              SUBH2   #1, (SP)                  ; (SP) = XHI^2/2 - 1
             54    8E 62FD  0AF7              SUBH2   (SP)+, R4                 ; R4/R7 = HCOS(X)
                       05   0AFB              RSB
                            0AFC
                            0AFC
                            0AFC      LARGE_SINCOS:
          5E    20     C2   0AFC              SUBL    #32, SP                   ; Allocate 16 longwords on the stack
          55    5E     D0   0AFF              MOVL    SP, R5                    ;   pointed to by R5
      00000E3D'EF      16   0B02              JSB     REDUCE_LARGE              ; (R5) = YLO
                            0B08                                                ; (R4) = YHI (in cycles)
                            0B08                                                ; R2 = octant bits
          5E    10     C2   0B08              SUBL    #16, SP                   ; Save space for HCOS(X)
             7E    65 7DFD  0B0B              MOVO    (R5), -(SP)               ; Save reduced
       7E   10 A5    7DFD   0B0F              MOVO    16(R5), -(SP)             ;     argument on stack
```

```
              34   BB  0B14   674         PUSHR   #^M<R2, R4, R5>        ; Save R2, R4 and R5
       00000C51'EF 16  0B16   675         JSB     L_COS                  ; R0/R1 = HCOS(X)
        2C AE    50 7DFD 0B1C 676         MOVO    R0, 44(SP)             ; (SP) = HCOS(X)
              34   BA  0B21   677         POPR    #^M<R2, R4, R5>        ; Restore R2, R4 and R5
        10 A5    8E 7DFD 0B23 678         MOVO    (SP)+, 16(R5)          ; Reduced argument on stack
              65 8E 7DFD 0B28 679         MOVO    (SP)+, (R5)            ;       pointed to by R4 and R5
       00000BA7'EF 16  0B2C   680         JSB     L_SIN                  ; R0/R3 = HSIN(X)
              54 8E 7DFD 0B32 681         MOVO    (SP)+, R4              ; R4/R7 = HCOS(X)
           5E 20   C0  0B36   682         ADDL    #32, SP                ; Pop 8 longwords off the stack
                   05  0B39   683         RSB
```

```
                     0B3A         .SBTTL MTH$HSIN_R5
                     0B3A
                     0B3A   ; This routine computes the HSIN of the G-format value of R0/R1.  The
                     0B3A   ; computation is performed one of three ways depending on the size of the
                     0B3A   ; input argument, X:
                     0B3A   ;
                     0B3A   ;        1) If |X| < pi/4, then X is used directly in a polynomial approximation
                     0B3A   ;           of HSIN.
                     0B3A   ;        2) If pi/4 =< |x| < 9*pi/4, then the subroutine REDUCE_MEDIUM is called
                     0B3A   ;           to reduce the argument to an equivalent argument in radians, Y, and
                     0B3A   ;           the octant, I, containing the argument.  Y is then evaluated in a
                     0B3A   ;           polynomial chosen as a function of I to compute HSIN(X).
                     0B3A   ;        3) If 9*pi/4 =< |X|, then the subroutine REDUCE_LARGE is called to
                     0B3A   ;           reduce the argument to an equivalent argument in cycles, Y, and the
                     0B3A   ;           octant, I, containing the argument.  Y is then evaluated in a
                     0B3A   ;           polynomial chosen as a function of I to compute HSIN(X).
                     0B3A
                     0B3A   MTH$HSIN_R5::
          50 73FD    0B3A           TSTH    R0                       ; Check the sign of R0
          10   18    0B3D           BGEQ    POS_SIN                  ;
    00000B4A'EF   16    0B3F           JSB     SIN                      ; R0/R1 = HSIN(|X|)
       50   50 72FD    0B45           MNEGH   R0, R0                   ; R0/R1 = HSIN(X)
               05    0B49           RSB
                     0B4A
                     0B4A   SIN:
  50    8000 8F   AA    0B4A           BICW    #^X8000, R0              ; R0/R1 = |X|
                     0B4F   POS_SIN:
  50    F4AC CF 71FD    0B4F           CMPH    H_PI_OV_4, R0            ; Compare pi/4 with |X|
             25   14    0B55           BGTR    SMALL_SIN                ; No argument reduction is necessary
  50    F4B4 CF 71FD    0B57           CMPH    H_9_PI_OV_4, R0          ; Compare 9*pi/4 with |X|
             3C   19    0B5D           BLSS    LARGE_SIN                ; Use special logic for |X| > 9*pi/4
                     0B5F
                     0B5F   ;
                     0B5F   ;  pi/4 =< |X| < 9*pi/4
                     0B5F   ;
     5E   20    C2    0B5F           SUBL    #32, SP                  ; Allocate 32 longwords on the stack
                     0B62                                            ;    for the reduced
    00000DA3'EF   16    0B62           JSB     REDUCE_MEDIUM            ; Medium argument reduction routine
                     0B68                                            ; Y = reduced argument on the stack,
                     0B68                                            ;    R2 = octant, R5 points to YHI and
                     0B68                                            ;    R4 points to YLO
  07   01   52   8F    0B68   M_SIN:  CASEB   R2, #1, #7               ; Branch to one of four polynomial
                     0B6C                                            ;    evaluations depending on the
            06DE'    0B6C   1$:     .WORD   P_COS_R-1$
            06DE'    0B6E           .WORD   P_COS_R-1$
            07A0'    0B70           .WORD   N_SIN_R-1$
            07A0'    0B72           .WORD   N_SIN_R-1$
            073C'    0B74           .WORD   N_COS_R-1$
            073C'    0B76           .WORD   N_COS_R-1$
            07AA'    0B78           .WORD   P_SIN_R-1$
            07AA'    0B7A           .WORD   P_SIN_R-1$               ;    octant bits.
                     0B7C
                     0B7C   ;
                     0B7C   ; Logic for small arguments.  |X| < pi/4.
                     0B7C   ;
                     0B7C
                     0B7C   SMALL_SIN:
```

```
  50   3FC7 8F    B1  0B7C              CMPW    #^X3FC7, R0              ; Compare with 2^-57
             17   18  0B81              BGEQ    1$                       ; No polynomial evaluation is needed
         7E  50 7DFD  0B83              MOVO    R0, -(SP)                ; (SP) = X
         50  50 64FD  0B87              MULH2   R0, R0                   ; R0/R3 = X*X
F7BE CF  0D  50 75FD  0B8B              POLYH   R0, #SINLENR-1, SINTBR   ; R0/R3 = g(x^2)
         50  6E 64FD  0B92              MULH2   (SP), R0                 ; R0/R3 = X*g(x^2)
         50  8E 60FD  0B96              ADDH2   (SP)+, R0                ; R0/R3 = HSIN(X)
                  05  0B9A      1$:     RSB
                      0B9B
                      0B9B
                      0B9B      LARGE_SIN:
         5E  20   C2  0B9B              SUBL    #32, SP                  ; Allocate 8 longwords on the stack
         55  5E   D0  0B9E              MOVL    SP, R5                   ;   pointed to by R5
   00000E3D'EF    16  0BA1              JSB     REDUCE_LARGE             ; (R5) = YLO, (R4) = YHI
                      0BA7                                               ; R2 = octant bits
             54   D5  0BA7      L_SIN:  TSTL    R4                       ; Check for degenerate case
             14   13  0BA9              BEQL    DEGENERATE_CASE_SIN
                      0BAB
  07    00   52   8F  0BAB              CASEB   R2, #0, #7
                      0BAF
               085C'  0BAF      1$:     .WORD   P_SIN_C-1$
               078D'  0BB1              .WORD   P_COS_C-1$
               078D'  0BB3              .WORD   P_COS_C-1$
               085C'  0BB5              .WORD   P_SIN_C-1$
               0853'  0BB7              .WORD   N_SIN_C-1$
               07ED'  0BB9              .WORD   N_COS_C-1$
               07ED'  0BBB              .WORD   N_COS_C-1$
               0853'  0BBD              .WORD   N_SIN_C-1$
                      0BBF
                      0BBF
                      0BBF      DEGENERATE_CASE_SIN:
                      0BBF
        52   01   8A  0BBF              BICB    #1, R2                   ; Compute index as (R2 - 1)/2
  52    52   FF 8F 9C 0BC2              ROTL    #-1, R2, R2
  03    00   52   8F  0BC7              CASEB   R2, #0, #3
                      0BCB
               0987'  0BCB      1$:     .WORD   P_ONE-1$
               0993'  0BCD              .WORD   UNFL -1$
               098C'  0BCF              .WORD   N_ONE-1$
               0993'  0BD1              .WORD   UNFL -1$
                      0BD3
```

```
                        0BD3  784
                        0BD3  785
                        0BD3  786                 .SBTTL MTH$HCOS_R5
                        0BD3  787
                        0BD3  788 ; This routine computes the HCOS of the G-format value of R0/R1.  The
                        0BD3  789 ; computation is performed one of three ways depending on the size of the
                        0BD3  790 ; input argument, X.  The processing is the same as described for MTH$HSIN_R4.
                        0BD3  791 ;
                        0BD3  792
                        0BD3  793 MTH$HCOS_R5::
              50 73FD   0BD3  794         TSTH    R0                      ; Check for reserved operand
      50    8000 8F  AA 0BD6  795         BICW    #^X8000, R0             ; R0/R1 = |X|
      50    F420 CF 71FD 0BDB 796         CMPH    H_PI_OV_4, R0           ; Compare pi/4 with |X|
                 25  14 0BE1  797         BGTR    SMALL_COS               ; No argument reduction is necessary
      50    F428 CF 71FD 0BE3 798         CMPH    H_9_PI_OV_4, R0         ; Compare 9*pi/4 with |X|
                 5A  19 0BE9  799         BLSS    LARGE_COS               ; Use special logic for |X| > 9*pi/4
                        0BEB  800
                        0BEB  801 ;
                        0BEB  802 ;   pi/4 =< |X| < 9*pi/4
                        0BEB  803 ;
         5E    20    C2 0BEB  804         SUBL    #32, SP                 ; Allocate 8 longwords on the stack
                        0BEE  805                                         ;   for the reduced argument
      00000DA3'EF    16 0BEE  806         JSB     REDUCE_MEDIUM           ; Medium argument reduction routine
                        0BF4  807                                         ; Y = reduced argument on the stack,
                        0BF4  808                                         ;   R2 = octant, R5 points to YHI
                        0BF4  809                                         ;   and R4 points to YLO
      07    01    52 8F 0BF4  810 M_COS:  CASEB   R2, #1, #7              ; Branch to one of four polynomial
                        0BF8  811                                         ;   evaluations depending on the
                  0714' 0BF8  812 1$:     .WORD   N_SIN_R-1$
                  0714' 0BFA  813         .WORD   N_SIN_R-1$
                  06B0' 0BFC  814         .WORD   N_COS_R-1$
                  06B0' 0BFE  815         .WORD   N_COS_R-1$
                  071E' 0C00  816         .WORD   P_SIN_R-1$
                  071E' 0C02  817         .WORD   P_SIN_R-1$
                  0652' 0C04  818         .WORD   P_COS_R-1$
                  0652' 0C06  819         .WORD   P_COS_R-1$              ;   octant bits.
                        0C08  820
                        0C08  821 ;
                        0C08  822 ; Logic for small arguments.  |X| < pi/4.
                        0C08  823 ;
                        0C08  824
                        0C08  825 SMALL_COS:
      50    4000 8F  B1 0C08  826         CMPW    #^X4000, R0             ; Compare 1/2 with |X|
                 1E  14 0C0D  827         BGTR    1$                      ; Sufficent overghang is available
         7E    50 7DFD  0C0F  828         MOVO    R0, -(SP)               ; (SP) = X
  08 AE  FFFF01FF 8F CA 0C13  829         BICL    #^XFFFF01FF, 8(SP)      ;
              0C AE  D4 0C1B  830         CLRL    12(SP)                  ; (SP) = XHI
      7E    50 6E 63FD  0C1E  831         SUBH3   (SP), R0, -(SP)         ; (SP) = XLO
            54    5E D0 0C23  832         MOVL    SP, R4                  ; R4 = pointer to XLO
      55    54    10 C1 0C26  833         ADDL3   #16, R4, R5             ; R5 = pointer to XHI
               062A  31 0C2A  834         BRW     NEEDS_DOUBLE            ; Use special logic to obtain overhang
      50    3FC7 8F  B1 0C2D  835 1$:     CMPW    #^X3FC7, R0             ; Compare with 2^-57
                 0C  18 0C32  836         BGEQ    2$                      ; No polynomial evaluation is needed
            50    50 64FD 0C34 837        MULH2   R0,R0                   ; R0/R1 = X*X
  F561 CF   0C    50 75FD 0C38 838        POLYH   R0, #COSLENR1-1, COSTBR1; R0/R1 = HCOS(X)
                     05 0C3F  839         RSB
                        0C40  840
```

```
             50    08 70FD  0C40   2$:       MOVH    #1.0, R0                    ; R0/R1 = 1.0 = HCOS(X)
                        05  0C44             RSB
                            0C45
                            0C45
                            0C45   LARGE_COS:
             5E    20   C2  0C45             SUBL    #32, SP                     ; Allocate 8 longwords on the stack
             55    5E   D0  0C48             MOVL    SP, R5                      ;   pointed to by R5
      00000E3D'EF       16  0C4B             JSB     REDUCE_LARGE                ; (R5) = YLO, (R4) = YHI
                            0C51                                                 ; R2 = octant bits
                  54    D5  0C51   L_COS:    TSTL    R4                          ; Check for degenerate case
                  14    13  0C53             BEQL    DEGENERATE_CASE_COS
                            0C55
       07    00    52   8F  0C55             CASEB   R2, #0, #7
                     06E3'  0C59   1$:       .WORD   P_COS_C-1$
                     07B2'  0C5B             .WORD   P_SIN_C-1$
                     07A9'  0C5D             .WORD   N_SIN_C-1$
                     0743'  0C5F             .WORD   N_COS_C-1$
                     0743'  0C61             .WORD   N_COS_C-1$
                     07A9'  0C63             .WORD   N_SIN_C-1$
                     07B2'  0C65             .WORD   P_SIN_C-1$
                     06E3'  0C67             .WORD   P_COS_C-1$
                            0C69
                            0C69
                            0C69   DEGENERATE_CASE_COS:
                            0C69
             52    01   8A  0C69             BICB    #1, R2                      ; Compute index as (R2 - 1)/2
   52    52    FF 8F    9C  0C6C             ROTL    #-1, R2, R2
       03    00    52   8F  0C71             CASEB   R2, #0, #3
                            0C75
                     08E9'  0C75   1$:       .WORD   UNFL -1$
                     08E2'  0C77             .WORD   N_ONE-1$
                     08E9'  0C79             .WORD   UNFL -1$
                     08DD'  0C7B             .WORD   P_ONE-1$
                            0C7D
```

```
                          0C7D         .SBTTL  MTH$HSINCOSD_R5
                          0C7D
                          0C7D   ; This routine computes the HSIND and HCOSD of the H-format value of R0/R3.
                          0C7D   ; The computation is performed one of two ways depending on the size of the
                          0C7D   ; input argument, X:
                          0C7D   ;
                          0C7D   ;       1) If |X| < 45, then X is used directly in polynomial approximation
                          0C7D   ;          of HSIND and HCOSD.
                          0C7D   ;       2) If 45 =< |x|, then the subroutine REDUCE_DEGREES is called to reduce
                          0C7D   ;          the argument to an equivalent argument in degrees, Y, and the
                          0C7D   ;          octant, I, containing the argument.  Y is then evaluated in two
                          0C7D   ;          polynomials chosen as a function of I, to compute HSIND(X) and
                          0C7D   ;          HCOSD(X).
                          0C7D
                          0C7D   MTH$HSINCOSD_R7::
              50 73FD     0C7D         TSTH    R0
              10   18     0C80         BGEQ    SINCOSD
     50   8000 8F  AA     0C82         BICW    #^X8000, R0             ; R0/R3 = |X|
      00000C92'EF  16     0C87         JSB     SINCOSD                 ; R0/R3 = HSIND(|X|)
                          0C8D                                         ; R4/R7 = HCOSD(|X|)
          50   50 72FD    0C8D         MNEGH   R0, R0                  ; R0/R4 = -HSIND(|X|)
                   05     0C91         RSB
                          0C92
                          0C92   SINCOSD:
     50   F3D9 CF 71FD    0C92         CMPH    H_45, R0                ; Compare 45 to |X|
              30   14     0C98         BGTR    SMALL_SINCOSD           ; special processing for small arg
          5E  10   C2     0C9A         SUBL    #16, SP                 ; Allocate 4 longwords on stack
          55  5E   D0     0C9D         MOVL    SP, R5                  ; R5 points to octaword on stack
      00001197'EF  16     0CA0         JSB     REDUCE_DEGREES          ; (R5) = reduced argument
                          0CA6                                         ; R4 = octant
          7E  65 7DFD     0CA6         MOVO    (R5), -(SP)             ; Save reduced arg
              30   BB     0CAA         PUSHR   #^M<R4, R5>             ; Save octant bits and pointer
      00000D78'EF  16     0CAC         JSB     EVAL_COSD               ; R0/R3 = HCOSD(Y)
              30   BA     0CB2         POPR    #^M<R4, R5>             ; R4/R5 = octant bits/pointer
          65  6E 7DFD     0CB4         MOVO    (SP), (R5)              ; (R5) = octant/reduced argument
          6E  50 7DFD     0CB8         MOVO    R0, (SP)                ; Save HCOSD(Y)
      00000D14'EF  16     0CBC         JSB     EVAL_SIND               ; R0/R3 = HSIND(Y)
          54  8E 7DFD     0CC2         MOVO    (SP)+, R4               ; R4/R7 = HCOSD(Y)
          5E  10   C0     0CC6         ADDL    #16, SP                 ; Pop 4 longwords
                   05     0CC9         RSB
                          0CCA
                          0CCA
                          0CCA   SMALL_SINCOSD:
          5E  20   C2     0CCA         SUBL    #32, SP                 ; Allocate 8 longwords on stack
          6E  50 7DFD     0CCD         MOVO    R0, (SP)                ; Save argument
      00000D8C'EF  16     0CD1         JSB     SMALL_COSD              ; R0/R3 = HCOSD(|X|)
       10 AE  50 7DFD     0CD7         MOVO    R0, 16(SP)              ; Save HCOSD(|X|)
          50  8E 7DFD     0CDC         MOVO    (SP)+, R0               ; R0/R3 = argument
      00000D28'EF  16     0CE0         JSB     SMALL_SIND              ; R0/R3 = HSIND(X)
          54  8E 7DFD     0CE6         MOVO    (SP)+, R4               ; R4/R7 = HCOSD(|X|)
                   05     0CEA         RSB
```

```
                          0CEB          .SBTTL  MTH$HSIND_R5
                          0CEB
                          0CEB  ; This routine computes the HSIND of the H-format value of R0/R3.  The
                          0CEB  ; computation is performed one of two ways depending on the size of the input
                          0CEB  ; argument, X:
                          0CEB  ;
                          0CEB  ;       1) If |X| < 45, then X is used directly in polynomial approximation
                          0CEB  ;          of HSIND.
                          0CEB  ;       2) If 45 =< |x|, then the subroutine REDUCE_DEGREES is called to reduce
                          0CEB  ;          the argument to an equivalent argument in degrees, Y, and the
                          0CEB  ;          octant, I, containing the argument.  Y is then evaluated in two
                          0CEB  ;          polynomials chosen as a function of I, to compute HSIND(X).
                          0CEB
                          0CEB  MTH$HSIND_R5::
              50 73FD     0CEB          TSTH    R0                      ; R0/R3 = X
              10   18     0CEE          BGEQ    POS_SIND                ;
      00000CFB'EF   16    0CF0          JSB     NEG_SIND                ; R0/R3 = HSIND(|X|)
         50   50 72FD     0CF6          MNEGH   R0, R0                  ; R0/R3 = -HSIND(|X|)
                   05     0CFA          RSB
                          0CFB
                          0CFB  NEG_SIND:
 50    8000 8F     AA     0CFB          BICW    #^X8000, R0             ; R0/R3 = |X|
                          0D00  POS_SIND:
 50    F36B CF   71FD     0D00          CMPH    H_45, R0                ; Compare 45 to |X|
              20   14     0D06          BGTR    SMALL_SIND              ; special processing for small arg
           5E   10   C2   0D08          SUBL    #16, SP                 ; Allocate 4 longwords on stack
           55   5E   D0   0D0B          MOVL    SP, R5                  ; R5 points to octaword on stack
      00001197'EF   16    0D0E          JSB     REDUCE_DEGREES          ; (R5) = reduced argument
                          0D14                                          ; R4 = octant
                          0D14
                          0D14  EVAL_SIND:
 07    00    54    8F     0D14          CASEB   R4, #0, #7
                0813'     0D18  1$:     .WORD   P_SIN_D-1$
                073B'     0D1A          .WORD   P_COS_D-1$
                073B'     0D1C          .WORD   P_COS_D-1$
                0813'     0D1E          .WORD   P_SIN_D-1$
                080F'     0D20          .WORD   N_SIN_D-1$
                07A1'     0D22          .WORD   N_COS_D-1$
                07A1'     0D24          .WORD   N_COS_D-1$
                080F'     0D26          .WORD   N_SIN_D-1$
                          0D28
                          0D28
                          0D28  SMALL_SIND:
 50    F363 CF   71FD     0D28          CMPH    H_SMALLD, R0            ; Compare 180/pi*2^-57 with |x|
              0A   14     0D2E          BGTR    1$                      ; No polynomial evaluation is
           7E   50 7DFD   0D30          MOVO    R0, -(SP)               ;   necessary
           55   5E   D0   0D34          MOVL    SP, R5                  ; R5 points to argument
            07F1   31     0D37          BRW     P_SIN_D                 ;
              50 73FD     0D3A  1$:     TSTH    R0                      ; Check for zero
              1C   13     0D3D          BEQL    3$                      ; Return if R0 = 0
 50    F38C CF   71FD     0D3F          CMPH    H_SMALLEST_DEG, R0      ; Check for possible underflow on
              03   15     0D45          BLEQ    2$                      ;   conversion to radians
            0814   31     0D47          BRW     UNFL                    ; Underflow will occur on conversion
           7E   50 7DFD   0D4A  2$:     MOVO    R0, -(SP)               ; (SP) = X
 50    F35D CF   64FD     0D4E          MULH2   H_CONVERT, R0           ; R0/R3 = (pi/180 - 2^-6)*|x|
           6E   06   A2   0D54          SUBW    #^X6, (SP)              ; R0/R3 = |X|*2^-6
      50   8E   60FD      0D57          ADDH2   (SP)+, R0               ; R0/R3 = HSIND(|X|) = (pi/180)|X|
```

```
05  0D5B   985 3$:     RSB
```

```
                           0D5C          .SBTTL  MTH$HCOSD_R5
                           0D5C
                           0D5C   ; This routine computes the HCOSD of the H-format value of R0.  The computation
                           0D5C   ; is performed one of two ways depending on the size of the input argument, X:
                           0D5C   ; Details are given in the discussion on MTH$HCOSD_R4.
                           0D5C
                           0D5C   MTH$HCOSD_R5::
               50 73FD     0D5C          TSTH    R0                    ; Check for reserved operand
    50   8000 8F    AA     0D5F          BICW    #^X8000, R0           ; R0/R3 = |X|
    50   F307 CF  71FD     0D64          CMPH    H_45, R0              ; Compare 45 to |X|
              20    14     0D6A          BGTR    SMALL_COSD            ;
         5E   10    C2     0D6C          SUBL    #16, SP               ; Allocate 4 longwords on stack
         55   5E    D0     0D6F          MOVL    SP, R5                ; R5 points to octaword on stack
      00001197'EF   16     0D72          JSB     REDUCE_DEGREES        ; (R5) = reduced argument
                           0D78                                        ; R4 = octant
                           0D78
                           0D78   EVAL_COSD:
    07   00   54    8F     0D78          CASEB   R4, #0, #7
                 06D7'     0D7C   1$:    .WORD   P_COS_D-1$
                 07AF'     0D7E          .WORD   P_SIN_D-1$
                 07AB'     0D80          .WORD   N_SIN_D-1$
                 073D'     0D82          .WORD   N_COS_D-1$
                 073D'     0D84          .WORD   N_COS_D-1$
                 07AB'     0D86          .WORD   N_SIN_D-1$
                 07AF'     0D88          .WORD   P_SIN_D-1$
                 06D7'     0D8A          .WORD   P_COS_D-1$
                           0D8C
                           0D8C
                           0D8C   SMALL_COSD:
    50   F2FF CF  71FD     0D8C          CMPH    H_SMALLD, R0          ; Compare 180/pi*2^-57  with |X|
              0A    14     0D92          BGTR    1$                    ; Check if polyinomial evaluation is
         7E   50  7DFD     0D94          MOVO    R0, -(SP)             ;     necessary.
         55   5E    D0     0D98          MOVL    SP, R5                ; R5 points to argument
            06B5    31     0D9B          BRW     P_COS_D               ; POLY needed
         50   08  70FD     0D9E   1$:    MOVH    #1, R0                ; R0 = 1. = HCOSD(|X|)
                    05     0DA2          RSB
                           0DA3
```

```
                              0DA3
                              0DA3
                              0DA3
                              0DA3        .SBTTL  REDUCE_MEDIUM
                              0DA3
                              0DA3  ;
                              0DA3  ; This routine assumes that the absolute value of the argument, X,  is in R0/R3
                              0DA3  ; and that pi/4 =< |X| < 9*pi/4.  It returns a pair of H-format values for the
                              0DA3  ; reduced argument in eight longwords on the stack previously allocate by the
                              0DA3  ; the calling routine.  These longwords will be denoted by T0 (4 off the stack
                              0DA3  ; pointer) through T7 (32 off the stack pointer).  YHI is in T4/T7, and YLO is
                              0DA3  ; in T0/T3.  The octant bits in are returned in R2.
                              0DA3  ;
                              0DA3  ; The reduced argument is obtained by locating the octant that X is in through
                              0DA3  ; a binary search and then subtracting off a suitable multiple of pi/2
                              0DA3  ;
                              0DA3
                              0DA3  REDUCE_MEDIUM:
      55    5E     14    C1   0DA3          ADDL3   #20, SP, R5             ; R5 points to T4
            65     50  7DFD   0DA7          MOVO    R0, (R5)                ; T4/T7 = X
      50  F280     CF  71FD   0DAB          CMPH    H_5_PI_OV_4, R0         ;
            12     15         0DB1          BLEQ    5$                      ; |X| >= 5*pi/4
      50  F268     CF  71FD   0DB3          CMPH    H_3_PI_OV_4, R0         ;
            05     15         0DB9          BLEQ    3$                      ; |X| >= 3*pi/4
      52    01     D0         0DBB          MOVL    #1, R2                  ; First quadrant
            17     11         0DBE          BRB     SUBTRACT
                              0DC0
      52    03     D0         0DC0  3$:     MOVL    #3, R2                  ; Second quadrant
            12     11         0DC3          BRB     SUBTRACT
                              0DC5
      50  F276     CF  71FD   0DC5  5$:     CMPH    H_7_PI_OV_4, R0 ;
            05     15         0DCB          BLEQ    7$                      ; |X| >= 7*pi/4
      52    05     D0         0DCD          MOVL    #5, R2                  ; Third quadrant
            05     11         0DD0          BRB     SUBTRACT
                              0DD2
      52    07     D0         0DD2  7$:     MOVL    #7, R2                  ; Fourth quadrant
            00     11         0DD5          BRB     SUBTRACT
                              0DD7
                              0DD7
                              0DD7  SUBTRACT:
      54    5E     04    C1   0DD7          ADDL3   #4, SP, R4              ; R4 = points to T0
      53    FD   A242    3E   0DDB          MOVAW   -3(R2)[R2], R3          ; R3 = index into PI_OV_2 table
   53     F2FB   CF43    7E   0DE0          MOVAQ   PI_OV_2[R3], R3         ; R3 = pointer into PI_OV_2 table
            65     83  62FD   0DE6          SUBH2   (R3)+, (R5)             ; T4/T7 = 1st approximation to YHI
                                                                            ;       = YHI'
            02     15         0DEA          BLEQ    1$
            52     D6         0DEC          INCL    R2                      ; Adjust octant bits
 51    65   8000   8F    AB   0DEE  1$:     BICW3   #^X8000, (R5), R1       ; R1 = exponents bits of |YHI'|
      51  3F95     8F    B1   0DF4          CMPW    #^X3F95, R1             ; Check for at least 6 significant bits
            18     14         0DF9          BGTR    NOT_ENOUGH_BITS         ;    in YHI'
                              0DFB
            64     65  7DFD   0DFB          MOVO    (R5), (R4)              ; T0/T3 = YHI'
 08 A5   FFFF01FF  8F    CA   0DFF          BICL    #^XFFFF01FF, 8(R5)      ;
            0C    A5     D4   0E07          CLRL    12(R5)                  ; T4/T7 = high 56 bits of YHI' = YHI
            64     65  62FD   0E0A          SUBH2   (R5), (R4)              ; T0/T3 = low bits of YHI'
            64     63  62FD   0E0E          SUBH2   (R3), (R4)              ; R4/R5 = YLO
                   05         0E12          RSB
                              0E13
```

```
                      0E13  1082 NOT_ENOUGH_BITS:                              ; Add an additional 49 significant bits
                      0E13  1083                                               ;   to YHI' to get YHI
         7E    63 7DFD 0E13  1084        MOVO    (R3), -(SP)                   ; (SP) = 113 additional bits (H-format)
          08   AE   7C 0E17  1085        CLRQ    8(SP)                         ; (SP) = 49 additional bits (H-format)
         65    6E 62FD 0E1A  1086        SUBH2   (SP), (R5)                    ; T4/T7 = YHI
    64   83    8E 63FD 0E1E  1087        SUBH3   (SP)+, (R3)+, (R4)            ; T0/T3 = 1st 64 bits of YLO
         64    63 60FD 0E23  1088        ADDH2   (R3), (R4)                    ; T0/T3 = -YLO
    64   8000  8F   AC 0E27  1089        XORW    #^X8000, (R4)                 ; T0/T3 = YLO
                    05 0E2C  1090        RSB
                      0E2D  1091
                      0E2D  1092
```

```
                                        0E2D           .SBTTL  REDUCE_LARGE
                                        0E2D
                                        0E2D   ;
                                        0E2D   ; This routine is used to reduce large arguments (|X| >= 9*pi/4) modulo pi/4.
                                        0E2D   ; It returns the reduced argument, Y, in R4/R5 in units of cycles, and returns
                                        0E2D   ; the octant bits, I, in R2.
                                        0E2D   ;
                                        0E2D   ; The method of reduction is as follows:
                                        0E2D   ;
                                        0E2D   ;       x*(4/pi) = 2^n*f*(4/pi) where n is an integer and 1/2 =< f < 1
                                        0E2D   ;                = 2^(n-113)*(2^113*f)*(4/pi)
                                        0E2D   ;                = (2^113*f)*(2^(n-113)*4/pi)
                                        0E2D   ;                = K*C, where K = 2^113*f is an integer and C = 2*(n-113)*4/pi
                                        0E2D   ;       Let L = K*C modulo 8, where 0 =< L < 8, and let I = the integer(L) and
                                        0E2D   ;       h = fract(L), then if I is even Y = h, otherwise Y = 1-h
                                        0E2D   ;
                                        0E2D   ; CONSTANTS:
                                        0E2D
                              0000003D  0E2D           L_INT_WEIGHT  = ^X3D            ; weights exponent by 61
                              00000020  0E2D           W_TERM_WEIGHT = ^X20            ; weights exponent by 32
                              00004000  0E2D           W_MAX_WEIGHT  = ^X4000          ; maximum unbiased exponent
                              00000040  0E2D           W_YLO_WEIGHT  = ^X40            ; weights exponest by 64
                              00003FF2  0E2D           W_ADJUST      = ^X3FF2          ; Used to locate binary point in
                                        0E2D                                           ;   MTH$AL_4_OV_PI_V table
                                        0E2D   H_2_TO_32:
00000000 00000000 00000000 00004011     0E2D           .OCTA   ^X4011                  ; 2^32
                                        0E3D
                                        0E3D
                                        0E3D
                                        0E3D   REDUCE_LARGE:
                                        0E3D   ;
                                        0E3D   ; The first step is to obtain the location of the binary point in the represen-
                                        0E3D   ; tation of C = 2^(n-113)*(4/pi) in two parts - the number of longwords from
                                        0E3D   ; the start and the number of bits from the most significant bit of the next
                                        0E3D   ; longword. Also K = 2^113*f must be obtained.
                                        0E3D   ;
                                        0E3D
     54    50   0000FFFF 8F       CB    0E3D           BICL3   #^XFFFF, R0, R4         ; R4 = 1st 16 fraction bits of X
                      54 01       88    0E45           BISB    #1, R4                  ; Restore hidden bit
               0C A5  54 10       9C    0E48           ROTL    #16, R4, 12(R5)         ; 12(R5) = High 17 bits of K
               08 A5  51 10       9C    0E4D           ROTL    #16, R1, 8(R5)          ; 8(R5) = Next 32 bits of K
                         03       18    0E52           BGEQ    1$                      ; Check for high bit set and
                      0C A5       D6    0E54           INCL    12(R5)                  ;   adjust if necessary
               04 A5  52 10       9C    0E57   1$:     ROTL    #16, R2, 4(R5)          ; 4(R5) = Next 32 bits of K
                         03       18    0E5C           BGEQ    2$                      ; Check for high bit set and
                      08 A5       D6    0E5E           INCL    8(R5)                   ;   adjust if necessary
                  65  53 10       9C    0E61   2$:     ROTL    #16, R3, (R5)           ; (R5) = Next 32 bits of K
                         03       18    0E65           BGEQ    3$                      ; Check for high bit set and
                      04 A5       D6    0E67           INCL    4(R5)                   ;   adjust if necessary
                                        0E6A
                                        0E6A
           50       3FF2 8F       A2    0E6A   3$:     SUBW    #W_ADJUST, R0           ; Unbias exp and adjust for leading
                                        0E6F                                           ;   zeroes.  R0 = location of binary
                                        0E6F                                           ;   point
     53    50         FD 8F       9C    0E6F           ROTL    #-3, R0, R3             ; Divide R0 by 32 and mull by 4 to get
        53      FFFFF003 8F       CA    0E74           BICL    #^XFFFFF003, R3         ;   R3 = # of longwords (in bytes) to
                                        0E7B                                           ;   binary point.
```

```
51   00000000'GF   DE  0E7B          MOVAL   G^MTH$AL_4_OV_PI_V, R1  ; R1 = address of RTL vector entry
51   00000000'GF   C0  0E82          ADDL    G^MTH$AL_4_OV_PI_V, R1  ; R1 = address of MTH$AL_4_OV_PI table
           51  53  C2  0E89          SUBL    R3, R1                  ; R1 points to 1st quadword of interest
     50    E0  8F  8A  0E8C          BICB    #^XE0, R0               ; R0(7:0) = # of bits within longword
                       0E90
                       0E90
                       0E90  ;
                       0E90  ; The next step is to generate an approximation to C, call it C'' to be used
                       0E90  ; in computing x*4/pi.  C'' will consist of the first three integer bits of
                       0E90  ; C and the first 61 fraction bits of C.  These bits will be obtained from a
                       0E90  ; constant stored in the interger array MTH$AL_4_OV_PI_V.
                       0E90  ;
                       0E90  ; NOTE:  The ASHQ, ADDL, and MULL instructions in the follow sections may
                       0E90  ; result in an integer overflow trap.  The overflow incurred is intentional,
                       0E90  ; so that the IV bit must be turned off.  The IV bit is not restored until
                       0E90  ; after all of the necessary fraction bits hav been generated.
                       0E90  ;
               7E  DC  0E90          MOVPSL  -(SP)                   ; Put current PSL on stack
6E   FFFFFFDF  8F  CA  0E92          BICL    #^C<PSL$M_IV>, (SP)     ; (SP) = current IV bit
               20  B9  0E99          BICPSW  #PSL$M_IV               ; Clear integer overflow bit
                       0E9B
                       0E9B  ; The necessary calculation to produce the reduced argument can requires up to
                       0E9B  ; 26 longwords of temporary work space.  This work space will be allocated
                       0E9B  ; on the stack.  Of the 26 longwords allocated, only 16 are needed for the
                       0E9B  ; duration of the reduction scheme.  For the purposes of comments these 16
                       0E9B  ; locations will be refered to as T0 through TE and will be accessed by
                       0E9B  ; pointers in R2, R3, R4, and SP.  The stack will be allocated (after initial-
                       0E9B  ; ization) as follows:
                       0E9B  ;
                       0E9B  ;                                           |                    |
                       0E9B  ;                                           |--------------------|
                       0E9B  ;                                     | T0: |                    |
                       0E9B  ;                                     |     |--------------------|
                       0E9B  ;   Temporary storage for use         | T1: |                    |
                       0E9B  ;   during the generation of          |     |--------------------|
                       0E9B  ;   addition bits of the pro-         | T2: |                    |
                       0E9B  ;   duct K*C'                         |     |--------------------|
                       0E9B  ;                                     | T3: |                    |
                       0E9B  ;                                     |     |--------------------|
                       0E9B  ;                                     | T4: |                    |
                       0E9B  ;                                     |     |--------------------|
                       0E9B  ;   Additional bits of C'             | T5: |                    |
                       0E9B  ;                                     |     |--------------------|
                       0E9B  ;                                     | T6: |                    |
                       0E9B  ;                                     |     |--------------------|
                       0E9B  ;                                     | T7: |                    |
                       0E9B  ;                                     |     |--------------------|
                       0E9B  ;                                     | T8: |                    |
                       0E9B  ;   Contains all of the               .     .         .          .
                       0E9B  ;   significant bits                  .     .         .          .
                       0E9B  ;   generated through                 .     .         .          .
                       0E9B  ;   current iteration                 |     |                    |
                       0E9B  ;                                     | TD: |                    |
                       0E9B  ;                                     |     |--------------------|
                       0E9B  ;                                     | TE: |                    |
                       0E9B  ;                                     |     |--------------------|
                       0E9B  ;   Weighting factor                  | TF: |                    |
```

```
                                  0E9B  ;                                                   |--------------------|
                                  0E9B  ;                                                   |                    |
                                  0E9B  ;
                                  0E9B
                                  0E9B
                                  0E9B  ;
                                  0E9B  ; Get C'' = C(0):C(1):...:C(7) on stack.  C(0) though  C(7) are unsigned
                                  0E9B  ; integers generated from the binary representation of C.  The high three bits
                                  0E9B  ; of C(0) are the the first three bits to the left of the binary point of C.
                                  0E9B  ; The remaining bits C(0) and C(1) though C(7) are the first 253 bits to the
                                  0E9B  ; right of the binary point of C. Note that the C(i)'s are adjusted to
                                  0E9B  ; compensate for their signed (rather than unsigned) interpretation in the EMUL
                                  0E9B  ; instruction.  Note also that the representation of C has no more than 15
                                  0E9B  ; consequtive ones, so that no carry is possible from the adjustment.
                                  0E9B  ;
                                  0E9B
       5E   00000068 8F      C2   0E9B          SUBL    #104, SP               ; Allocate 26 longwords on the stack
            52    5E   20    C1   0EA2          ADDL3   #32, SP, R2            ; R2 points to longword before C(7)
            53    52   1C    C1   0EA6          ADDL3   #28, R2, R3            ; Initailize loop counter.  R3 points
                                  0EAA                                         ;   to longword before C(0)
            63    61   50    79   0EAA          ASHQ    R0, (R1), (R3)         ; Shift the proper quadword so that
                                  0EAE                                         ;     4(R3) has C(0)in it
                  53   04    C2   0EAE          SUBL    #4, R3                 ; R3 points to longword before C(6)
                  51   04    C2   0EB1  4$:     SUBL    #4, R1                 ; R2 points to next quadword in
                                  0EB4                                         ;     MTH$AL_4_OV_PI_V table
            63    61   50    79   0EB4          ASHQ    R0, (R1), (R3)         ; Shift quadword so that C(n) is in
                                  0EB8                                         ;   the appropriate longowrd
                       03    18   0EB8          BGEQ    5$                     ; Check for high bit of C(n) set
                    08 A3    D6   0EBA          INCL    8(R3)                  ; Bit set.  Adjust C(n-1)
 FFEA 53    FFFFFFFC 8F  52  F1   0EBD  5$:     ACBL    R2, #-4, R3, 4$        ; Loop until C(0) though C(7) are on
                                  0EC7                                         ;   the stack
                                  0EC7
                                  0EC7  ;
                                  0EC7  ; Generate the low 256 bits of the product K*C'' = L.  This product is
                                  0EC7  ; equivalent to multiplying K times C'' modulo 8.  The result of the
                                  0EC7  ; product is in T7/TE with bits 31:29 of TE the octant bits, and the remaining
                                  0EC7  ; 252 bits the faction bits of the product.  The last 113 fraction bits (bits
                                  0EC7  ; are non-valid fraction bits that will be used later if more fraction bits
                                  0EC7  ; need to be generated.
                                  0EC7  ;
                                  0EC7
                  6E   50    7D   0EC7          MOVQ    R0, (SP)               ; Save R0/R1
                                  0ECA
                                  0ECA  ; Set up pointers for multiplying
                                  0ECA
            51    5E   24    C1   0ECA          ADDL3   #36, SP, R1            ; R1 points to 1st longword of C''
            52    51   20    C1   0ECE          ADDL3   #32, R1, R2            ; R2 points to T7
            54    52   18    C1   0ED2          ADDL3   #24, R2, R4            ; R4 points to TD
                                  0ED6
                       62    D4   0ED6          CLRL    (R2)                   ; This code generates the product
      62  62   81   65   7A   0ED8  6$:     EMUL    (R5), (R1)+, (R2), (R2) ;   of the low order 32 bits of K
    FFF5  52   04   54   F1   0EDD          ACBL    R4, #4, R2, 6$         ;   and C'' and stores them in
      04  A2   61   65   C5   0EE3          MULL3   (R5), (R1), 4(R2)      ;   location T7/TE
         62        04 A2  C0  0EE8          ADDL    4(R2), (R2)            ;
                                  0EEC
                                  0EEC
                       53    D4   0EEC          CLRL    R3                     ; R3 is the loop counter
```

```
                            0EEE
        50   08 AE43   DE   0EEE   7$:     MOVAL   8(SP)[R3], R0             ; R0 and R4 point to 1st and last
        54   5E   1C   C1   0EF3           ADDL3   #28, SP, R4               ;   longword of temporary storage
                            0EF7                                             ;   for computing the product of the
                            0EF7                                             ;   next 32 bits of K and C''
        55   04        C0   0EF7           ADDL    #4, R5                    ; R5 points to next 32 bits of K
        51   5E   24   C1   0EFA           ADDL3   #36, SP, R1               ; R1 points to 1st longwork of C''
                            0EFE
                  60   D4   0EFE           CLRL    (R0)                      ; This loop generates the product of
   60   60   81   65   7A   0F00   8$:     EMUL    (R5), (R1)+, (R0), (R0)   ;   the next 32 bits of K and C''.
 FFF5 50   04   54   F1   0F05           ACBL    R4, #4, R0, 8$            ;   The result is stored n+2 longwords
    64 AE   61   65   C5   0F0B           MULL3   (R5), (R1), 100(SP)       ;   off SP, where n is stored in R3
        60     64 AE   C0   0F10           ADDL    100(SP), (R0)             ;
                            0F14
        50   08 AE43   DE   0F14           MOVAL   8(SP)[R3], R0             ; R0 points n+2 longwors off SP
        52   48 AE43   DE   0F19           MOVAL   72(SP)[R3], R2            ; R2 points to T7 + n longwords
             54   04   C0   0F1E           ADDL    #4, R4                    ;
                            0F21
             82   80   C0   0F21           ADDL    (R0)+, (R2)+              ; This loop adds the product bits of
             82   80   D8   0F24   9$:     ADWC    (R0)+, (R2)+              ;   the previous loop to the bits
                            0F27                                             ;   already generated.  The results
 FFF7 50   00   54   F1   0F27           ACBL    R4, #0, R0, 9$            ; This curious construction is used to
                            0F2D                                             ;   avoid effecting the carry bit
                            0F2D
 FFBB 53   01   02   F1   0F2D           ACBL    #2, #1, R3, 7$            ; Increment loop counter and branch
                            0F33
        50   6E        7D   0F33           MOVQ    (SP), R0                  ; Restore R0/R1
        55   0C        C2   0F36           SUBL    #12, R5                   ; R5 points to K
        5E   28        C0   0F39           ADDL    #40, SP                   ; Pop 10 longwords off stack
                            0F3C
                            0F3C   ;
                            0F3C   ; At this point there may or may not be enough valid bits in R3/R4 to generate
                            0F3C   ; Y.  If the first 12 fratction bits are all 1's or 0's, there a possibility of
                            0F3C   ; loss of significance when computing Y.  Consequently, we must check for loss
                            0F3C   ; of significance before converting T4/T7 to Y and I.
                            0F3C   ;
                            0F3C
                            0F3C   ; Set up pointers
                            0F3C
        53   5E   1C   C1   0F3C           ADDL3   #28, SP, R3               ; R3 points to T7
        54   5E   3C   C1   0F40           ADDL3   #60, SP, R4               ; R4 points to TF
                            0F44
 64    FC A4   00000080 8F   C1   0F44     ADDL3   #^X80, -4(R4), (R4)       ; If the first 22 fraction bits are 1's
       64      3FFFFF00 8F   D3   0F4D     BITL    #^X3FFFFF00, (R4)         ;   and the reduced arg = 1-f or the
                       37    12   0F54     BNEQ    CONVERT                   ;   first 22 bit are 0 and the reduced
                            0F56                                             ;   arg = f, then (and only then) bits
                            0F56                                             ;   29:8 are 0 and significance will
                            0F56                                             ;   be lost.
                            0F56
                            0F56   ;
                            0F56   ; More bits need to be generated to cover the loss of significance.
                            0F56   ;
                            0F56
          0000108F'EF  16   0F56           JSB     GEN_MORE_BITS             ; Generate 85 additional bits and add
                            0F5C                                             ;   them to existing bits.  Results are
                            0F5C                                             ;   stored in T6/TE
        53   04        C2   0F5C           SUBL    #4, R3                    ; Adjust R3 to reflect the addition of
```

```
                              0F5F                                                   ;   another longword of K*C''
        15 FC A4   1D   E0    0F5F              BBS     #29, -4(R4), 10$             ;      ; Check if loss of significance is d
                              0F64                                                   ;   to leading ones or zeros
                              0F64
                              0F64       ; Lost significance due to leading zeros
                              0F64
  64   FC A4   08   00   EA   0F64              FFS     #0, #8, -4(R4), (R4)         ; If at least one bit is set.  This
                    21   12   0F6A              BNEQ    CONVERT                      ;   means lost significance was minor.
  F8 A4    0001FFFF 8F   D1   0F6C              CMPL    #^X1FFFF, -8(R4)             ; If one of the high 15 bits is set,
                    17   15   0F74              BLEQ    CONVERT                      ;   lost significance was minor.
                  00B4   31   0F76              BRW     LEADING_ZEROS
                              0F79
                              0F79       ; Lost significance due to leading ones
                              0F79
  64   FC A4   08   00   EB   0F79       10$:   FFC     #0, #8, -4(R4), (R4)         ; If at least one bit is clear. This
                    0C   12   0F7F              BNEQ    CONVERT                      ;   meanslost significance was minor.
  F8 A4    FFFE0000 8F   D1   0F81              CMPL    #^XFFFE0000, -8(R4)          ; If one of the three high bits is
                    02   1E   0F89              BGEQU   CONVERT                      ;   clear, lost significance was minor.
                    3C   11   0F8B              BRB     LEADING_ONES
                              0F8D
                              0F8D
                              0F8D       CONVERT:
                              0F8D
                              0F8D       ;      Isolate octant bits and convert fraction bits to a pair of D-format
                              0F8D       ;      quantities YHI and YLO
                              0F8D
  64   FC A4   03   1D   EF   0F8D              EXTZV   #29, #3, -4(R4), (R4)        ; TF = octant bits
   FC A4   E0000000 8F   CA   0F93              BICL    #^XE0000000, -4(R4)          ; Clear octant bits
        53   54     14   C3   0F9B              SUBL3   #20, R4, R3                  ; R3 points to low order bits of h
          0000110C'EF    16   0F9F              JSB     CVT_TO_H                     ; (R5) = 2^29*h_lo
                              0FA5                                                   ; 16(R5) = 2^29*h_hi
               10   A5   B5   0FA5              TSTW    16(R5)                       ; 16(R5) could be zero. If so
                    04   13   0FA8              BEQL    H_HI_0                       ;  do not do SUBW #^X1D,16(R5)
          10 A5     1D   A2   0FAA              SUBW    #^X1D, 16(R5)                ; 16(R5) = h_hi
                              0FAE       H_HI_0:
               65   1D   A2   0FAE              SUBW    #^X1D, (R5)                  ; (R5) = h_lo
                    65   B5   0FB1              TSTW    (R5)                         ; Check for h_lo = 0
                    00   13   0FB3              BEQL    1$
               0B   64   E9   0FB5       1$:    BLBC    (R4), 2$                     ; Check for odd or even octant bits
                              0FB8
                              0FB8       ;      Octant bits are odd.  Reduced argument equals 1 - h.
                              0FB8
   10 A5    08   10 A5 63FD   0FB8              SUBH3   16(R5), #1, 16(R5)           ; 16(R5) = YHI = 1 - h_hi
               65    65 72FD  0FBF              MNEGH   (R5), (R5)                   ; (R5) = YLO = -h_lo
                              0FC3
                              0FC3       ;      Get octant bits
                              0FC3
               52   64   D0   0FC3       2$:    MOVL    (R4), R2                     ; R2 = octant bits
                  00A8   31   0FC6              BRW     GET_YHI_YLO
                              0FC9
                              0FC9
                              0FC9       ;
                              0FC9       ; At this point it has been determined that there is a major loss of
                              0FC9       ; significance and the processing begins a looping phase.  Each iteration of
                              0FC9       ; the loop will generate additional extra bits of K*C' until enough significant
                              0FC9       ; bits to compute Y are available.  During this time the nine longwords
                              0FC9       ; allocated on the stack will be used as follows:
```

```
                                0FC9  ;
                                0FC9  ;         T0/T4   Temporary storage used when generating extra bits.
                                0FC9  ;
                                0FC9  ;         T6/TE   Contains all significant bits generated so far.
                                0FC9  ;
                                0FC9  ;         TF      Contains a counter, W, indicating the appropriate exponent
                                0FC9  ;                   of the last longword of fraction bits used in converting
                                0FC9  ;                   to Y.
                                0FC9
                                0FC9  LEADING_ONES:
                                0FC9
                                0FC9  ; If processing continues here it is known that the loss of significance is due
                                0FC9  ; to a string of leading ones.
                                0FC9
               64     3D    D0  0FC9          MOVL    #L_INT_WEIGHT, (R4)         ; TE = exp bias for last longword
                                0FCC                                              ;      of the product K*C'
                                0FCC
      F8 A4   FFFF8000 8F   D1  0FCC  LOOP_1: CMPL    #^XFFFF8000, -8(R4)         ; Check for enough significant bits
                      2F    1A  0FD4          BGTRU   CONVERT_1                   ; Enough bits.  Convert to floating.
         0000108F'EF        16  0FD6          JSB     GEN_MORE_BITS               ; T5/TE contains K*C''
      F8 A4   FFFFFFFF 8F   D1  0FDC          CMPL    #-1, -8(R4)                 ; Check for all 1's
                      1F    1A  0FE4          BGTRU   CONVERT_1                   ; Not all 1's.  Enough precision bits
                                0FE6                                              ;   to compute Y
         EC A4    E8 A4 7DFD    0FE6          MOVO    -24(R4), -20(R4)            ; Compress representation
         DC A4    D8 A4 7DFD    0FEC          MOVO    -40(R4), -36(R4)            ;   of K*C''
 FFD2 64    20    4000 8F   3D  0FF2          ACBW    #W_MAX_WEIGHT, #W_TERM_WEIGHT, (R4), LOOP_1
                                0FFA                                              ; Increment weighting factor.  If
                                0FFA                                              ;   weighting factor is greater than
                                0FFA                                              ;   1024 then no more bits need to be
                                0FFA                                              ;   generated.
                                0FFA
                                0FFA  ;
                                0FFA  ; The weighting factor is greater than 16384.  This means that the reduced
                                0FFA  ; argument is either not distinguishable from 1 or too small to be represented
                                0FFA  ; in F-format (i.e. underflow.)  Zero is returned in R4 for the reduced
                                0FFA  ; argument to signal this occurance.  Note that under these conditions the
                                0FFA  ; correct function value is one of the values 0, +/-1.  The
                                0FFA  ; correct choice is determined by the calling program based on the octant bits
                                0FFA  ; returned in R1.
                                0FFA  ;
                      65    D4  0FFA          CLRL    (R5)                        ; Reduced argument is zero
    52    FC A4    03  1D   EF  0FFC          EXTZV   #29, #3, -4(R4), R2         ; R2 = octant bits
                    007F    31  1002          BRW     RESTORE
                                1005
                                1005
                                1005  CONVERT_1:
          53    54    18    C3  1005          SUBL3   #24, R4, R3                 ; R3 points to low bits of h
         0000110C'EF        16  1009          JSB     CVT_TO_H                    ; (R5)= 2^W*h_lo
                                100F                                              ; 16(R5) = 2^W*h_hi
 10 A5    FE17 CF    10 A5 63FD 100F          SUBH3   16(R5), H_2_TO_32, 16(R5)
                                1018                                              ; 16(R5) = 2^W*(1 - h_hi)
             10 A5    64    C2  1018          SUBL    (R4), 16(R5)                ; 16(R5) = 1 - h_hi
                65    65  72FD  101C          MNEGH   (R5), (R5)                  ; (R5) = -2^W*h_lo
                      03    13  1020          BEQL    1$                          ; Check for h_lo = 0
                65    64    C2  1022          SUBL    (R4), (R5)                  ; (R5) = - h_lo
    52    FC A4    03  1D   EF  1025  1$:     EXTZV   #29, #3, -4(R4), R2         ; R2 = octant bits
                      44    11  102B          BRB     GET_YHI_YLO
```

```
                                 102D
                                 102D
                                 102D   LEADING_ZEROS:
                                 102D
                                 102D   ; If processing continues here it is known that the loss of significance is due
                                 102D   ; to a string of leading zeros.  Note that it is known that the loop for
                                 102D   ; leading zeros will terminate before an underflow condition occurs so that the
                                 102D   ; loop does not include a test for underflow.
                                 102D
                64    3D    D0   102D           MOVL    #L_INT_WEIGHT, (R4)      ; TE = exp bias for last longword
                                 1030                                            ;     of the product K*C'
                                 1030
  F8 A4   00001FFF 8F    D1      1030   LOOP_0: CMPL    #^X00001FFF, -8(R4)      ; Check enough fraction bits
                      1C    19   1038           BLSS    CONVERT_0                ; Enough bits.  Convert to floating
             0000108F'EF    16   103A           JSB     GEN_MORE_BITS            ; T2/R7 contain K*C''
                   F8 A4    D5   1040   1$:     TSTL    -8(R4)                   ; Check for all 0's
                      11    12   1043           BNEQ    CONVERT_0                ; Not all 0's.  Enough precision bits.
     EC A4     E8 A4  7DFD       1045           MOVO    -24(R4), -20(R4)         ; Compress representation
     DC A4     D8 A4  7DFD       104B           MOVO    -40(R4), -36(R4)         ;    of K*C''
          64      20    A0       1051           ADDW    #W_TERM_WEIGHT, (R4)     ; Increment weighting factor.
                  DA    11       1054           BRB     LOOP_0
                                 1056
                                 1056   CONVERT_0:
      53    54    18    C3       1056           SUBL3   #24, R4, R3              ; R3 points to low bits of h
             0000110C'EF    16   105A           JSB     CVT_TO_H                 ; (R5) = 2^W*h_lo
                                 1060                                            ; 16(R5) = 2^W*h_hi
  52   FC A4   03   1D    EF     1060           EXTZV   #29, #3, -4(R4), R2      ; R2 = octant bits
          10 A5     64    C2     1066           SUBL    (R4), 16(R5)             ; 16(R5) = h_hi
                    65    B5     106A           TSTW    (R5)                     ; Check for h_lo = 0
                    03    13     106C           BEQL    GET_YHI_YLO              ;
          65        64    C2     106E           SUBL    (R4), (R5)               ; (R5) = h_lo
                                 1071
                                 1071   GET_YHI_YLO:
      54    55    10    C1       1071           ADDL3   #16, R5, R4              ; R4 points to YHI
            7E    64  7DFD       1075           MOVO    (R4), -(SP)              ; (SP) = high bits of Y
              08 A4     7C       1079           CLRQ    8(R4)                    ; (R5) = 49 high bits of Y = YHI
            6E    64  62FD       107C           SUBH2   (R4), (SP)               ;
            65    8E  60FD       1080           ADDH2   (SP)+, (R5)              ; (R4) = YLO
                                 1084   RESTORE:
              40 AE     B8       1084           BISPSW  64(SP)                   ; Restore IV bit and exit
  5E     00000044 8F    C0       1087           ADDL    #68, SP                  ; Remove mask and temporary storage
                        05       108E           RSB
                                 108F
                                 108F
                                 108F   GEN_MORE_BITS:
                                 108F
                                 108F   ;
                                 108F   ; This subroutine generates 145 extra fraction bits and adds them to the
                                 108F   ; existing bits.  NOTE:  This routine is always entered via a JSB instruction.
                                 108F   ; Consequently, SP points to the first longword BEFORE T0, rather than T0
                                 108F   ; itself.  R3 points to the last longword of previously generated bits.
                                 108F   ;
      52    5E    18    C1       108F           ADDL3   #24, SP, R2              ; R2 points to storage for C(n)
                                 1093
            51    04    C2       1093           SUBL    #4, R1                   ; Adjust pointer to get next quadword
                                 1096                                            ;    from MTH$AL_4_OV_PI_V
  FC A2     61    50    79       1096           ASHQ    R0, (R1), -4(R2)         ; T5 = C(n)
```

```
                      33   18 109B         BGEQ    1$                                ; Branch if high bit is clear
                              109D
                              109D ; Logic to process unsigned values greater than 2^31 - 1
                              109D
      04 AE    00   62   65   7A 109D      EMUL    (R5), (R2), #0, 4(SP)
                  08 AE  65   C0 10A3      ADDL    (R5), 8(SP)
 08 AE   08 AE  62    04 A5   7A 10A7      EMUL    4(R5), (R2), 8(SP), 8(SP)
              0C AE   04 A5   C0 10AF      ADDL    4(R5), 12(SP)
 0C AE   0C AE  62    08 A5   7A 10B4      EMUL    8(R5), (R2), 12(SP), 12(SP)
              10 AE   08 A5   C0 10BC      ADDL    8(R5), 16(SP)
 10 AE   10 AE  62    0C A5   7A 10C1      EMUL    12(R5), (R2), 16(SP), 16(SP)
              14 AE   0C A5   C0 10C9      ADDL    12(R5), 20(SP)
                         1E   11 10CE      BRB     2$
                              10D0
                              10D0 ; Logic to process unsigned values less than 2^31
                              10D0
      04 AE    00   62   65   7A 10D0 1$:  EMUL    (R5), (R2), #0, 4(SP)
 08 AE   08 AE  62    04 A5   7A 10D6      EMUL    4(R5), (R2), 8(SP), 8(SP)
 0C AE   0C AE  62    08 A5   7A 10DE      EMUL    8(R5), (R2), 12(SP), 12(SP)
 10 AE   10 AE  62    0C A5   7A 10E6      EMUL    12(R5), (R2), 16(SP), 16(SP)
                              10EE
                              10EE
                              10EE ; Add new bits to old
                              10EE
           52   5E   04       C1 10EE 2$:  ADDL3   #4, SP, R2                        ; R2 points to low order longword of
                              10F2                                                   ;    of new bits
                FC A3    82   D0 10F2      MOVL    (R2)+, -4(R3)                     ; Move low order bits to end of list
                   63    82   C0 10F6      ADDL    (R2)+, (R3)                       ; Add in new bits
                04 A3    82   D8 10F9      ADWC    (R2)+, 4(R3)                      ;
                08 A3    82   D8 10FD      ADWC    (R2)+, 8(R3)                      ;
                0C A3    82   D8 1101      ADWC    (R2)+, 12(R3)                     ;
                         04   1E 1105 3$:  BCC     4$                                ; Check for carry from previous add
                         82   D6 1107      INCL    (R2)+                             ; Propagate carry
                         FA   11 1109      BRB     3$                                ;
                              05 110B 4$:  RSB                                       ;
                              110C
                              110C
                              110C
                              110C
                              110C CVT_TO_H:
                              110C
                              110C ;
                              110C ; This routine converts an array of five longword pointed to by R3 to a pair
                              110C ; of H-format values.  The results are returned on the stack and are pointed
                              110C ; to by R5
                              110C ;
                              110C
                              110C
           52   53   0C       C1 110C      ADDL3   #12, R3, R2                       ; R2 point to second most significant
                              1110                                                   ;    longword of h
           50   5E   04       C1 1110      ADDL3   #4, SP, R0                        ; R0 points to T0
                              1114
                         62   D5 1114 1$:  TSTL    (R2)                              ; Adjust for signed
                         03   18 1116      BGEQ    2$                                ;   rather than unsigned
                      04 A2   D6 1118      INCL    4(R2)                             ;     conversion
 FFEF 52   FFFFFFFC 8F   53   F1 111B 2$:  ACBL    R3, #-4, R2, 1$                   ;
                              1125
```

```
              60   83 6EFD  1125  1550          CVTLH   (R3)+, (R0)             ;
                   03   13  1129  1551          BEQL    3$                      ;
              60   20   A2  112B  1552          SUBW    #W_TERM_WEIGHT, (R0)    ; Convert low order
              7E   83 6EFD  112E  1553 3$:      CVTLH   (R3)+, -(SP)            ;   three longword to
              60   6E 60FD  1132  1554          ADDH2   (SP), (R0)              ;     H-format and store
                   03   13  1136  1555          BEQL    4$                      ;       on stack pointed to
              60   20   A2  1138  1556          SUBW    #W_TERM_WEIGHT, (R0)    ;         by R5
              6E   83 6EFD  113B  1557 4$:      CVTLH   (R3)+, (SP)             ;
        65    60   6E 61FD  113F  1558          ADDH3   (SP), (R0), (R5)        ;
                   05   13  1144  1559          BEQL    5$                      ;
        65    0040 8F   A2  1146  1560          SUBW    #W_YLO_WEIGHT, (R5)     ;
                            114B  1561
              60   83 6EFD  114B  1562 5$:      CVTLH   (R3)+, (R0)             ;
                   03   13  114F  1563          BEQL    6$                      ;
              60   20   A2  1151  1564          SUBW    #W_TERM_WEIGHT, (R0)    ; Convert high order
              6E   83 6EFD  1154  1565 6$:      CVTLH   (R3)+, (SP)             ;   two longwords to
                   06   18  1158  1566          BGEQ    7$                      ;     H-format and store
        6E    FCCE CF 60FD  115A  1567          ADDH2   H_2_TO_32, (SP)         ;       16 off R5 on the
     10 A5    60   8E 61FD  1160  1568 7$:      ADDH3   (SP)+, (R0), 16(R5)     ;         stack
                   05       1166  1569          RSB
                            1167  1570
```

```
                              1167          .SBTTL  REDUCE_DEGREES
                              1167
                              1167  ; This routine assumes that the absolute value of the argument is in R0/R3.
                              1167  ; The reduction process is performed in two stages.  The first stage of
                              1167  ; the reduction reduces the argument modulo 360 to a value less that 2^112,
                              1167  ; and the second stage reduces the argument modulo 45 to a value less than 45.
                              1167  ; The reduced argument is returned in four longwords previously allocated by
                              1167  ; the calling program and pointed to by R5.  The octant bits are returned in
                              1167  ; R4.
                              1167
                              1167  ; Constants used in this reduction:
                              1167  ;
                              1167
                              1167  POWER_MOD_360_0:        ; Powers of 2 modulo 360 for t1 = 0
      0008 0004 0002 0001     1167          .WORD      1,     2,     4,    8
      0080 0040 0020 0010     116F          .WORD     16,    32,    64,  128
      00F8 0130 0098 0100     1177          .WORD    256,   152,   304,  248
                              117F
                              117F  POWER_MOD_360_1:        ; Powers of 2 modulo 360 for t1 <> 0
      0008 00B8 0110 0088     117F          .WORD    136,   272,   184,    8
      0080 0040 0020 0010     1187          .WORD     16,    32,    64,  128
      00F8 0130 0098 0100     118F          .WORD    256,   152,   304,  248
                              1197
                              1197
                              1197
                              1197  REDUCE_DEGREES:
     50    4071 8F      B1    1197          CMPW    #^X4071, R0             ; Compare |x| with 2^112
                 46     14    119C          BGTR    LAST_STEP               ; Branch to special logic for med arg
                              119E
                              119E  ;
                              119E  ; It is assumed here that the argument is greater than 2^113.
                              119E  ;
                              119E  ; The argument is reduced as follows:
                              119E  ;       Let x = 2^t*f, where t > 113 and 1/2 =< f < 1.  And let J = 2^113*f =
                              119E  ;       2^60*J1 + J2 and K = 2^(t-113).  Since 2^60 = 2^12 modulo 360, we have
                              119E  ;       J = 2^12*J1 + J2 modulo 360.  Now let t' = t - 113 = 12*t1 + t2.  Note
                              119E  ;       that (2^12)^2 = (2^9)*(2^15) = (2^9)*(2^3) = 2^12 modulo 360.  Hence, if
                              119E  ;       t1 is not zero, K = 2^t' = 2^(12*t1+t2) = (2^12)*(2^t2) = 136*2^t2 modulo
                              119E  ;       360.  For t1 = 0 K = 2^t2.  Consequently, define K' congruent to 2^t2 if
                              119E  ;       t1 = 0 and congruent to 136*2^t2 otherwise, where 0 =< K' < 360.  Then x'
                              119E  ;       = K'*(2^12*J1 + J2) is congruent to s modulo 360 and x' < 2^113.
                              119E
          54     50     D0    119E          MOVL    R0, R4                  ; R4 = exponent bits of X
     50    7FFF 8F      AA    11A1          BICW    #^X7FFF, R0             ; Clear exp bits of X
     50    4071 8F      A8    11A6          BISW    #^X4071, R0             ; R0/R3 = J
          54     50     C2    11AB          SUBL    R0, R4                  ; R4 = t'
                              11AE
          65     50   7DFD    11AE          MOVO    R0, (R5)                ; (R5) = J
  52   FFFF0FFF 8F      CA    11B2          BICL    #^XFFFF0FFF, R2         ;
                 53     D4    11B9          CLRL    R3                      ; R0/R3 = J1*2^60
          65     50   62FD    11BB          SUBH2   R0, (R5)                ; (R5) = J2
          50     30     A2    11BF          SUBW    #^X30, R0               ; R0/R3 = 2^12*J1
          50     65   60FD    11C2          ADDH2   (R5), R0                ; R0/R3 = 2^12*J1 + J2 = J modulo 45
                              11C6
     65   54     0C     A7    11C6          DIVW3   #12, R4, (R5)           ; (R5) = t1
                 0E     13    11CA          BEQL    1$
          65     0C     A4    11CC          MULW    #12, (R5)               ; (R5) = 12*t1
```

```
              54   65     A2  11CF           SUBW    (R5), R4                  ; R4 = t2
        65    A8 AF44   6DFD  11D2           CVTWH   POWER_MOD_360_1[R4], (R5)
                06        11  11D8           BRB     2$                        ; (R5) = K'
                              11DA
        65    88 AF44   6DFD  11DA  1$:      CVTWH   POWER_MOD_360_0[R4], (R5)
                              11E0                                             ; (R5) = K'
        50       65     64FD  11E0  2$:      MULH2   (R5), R0                  ; R0/R3 = X' (mod 45) 0 =< R0 < 2^53
                              11E4
                              11E4
                              11E4  LAST_STEP:
                              11E4  ;
                              11E4  ; Argument reduction scheme for arguments with absolute value less than 2^112
                              11E4  ;
                              11E4  ; The reduced argument Y is computed as follows:
                              11E4  ;     Let I = int(X/45)
                              11E4  ;         if I is even
                              11E4  ;             then Y = X - 45*I
                              11E4  ;             else Y = (I+1)*45 - x
                              11E4
                              11E4
        50    4024 8F     B1  11E4           CMPW    #^X4024, R0               ; Compare 2^36 with |X|
                27        18  11E9           BGEQ    NO_OVERFLOW               ;
   65   50    EEB0 CF   65FD  11EB           MULH3   H_T_OV_45, R0, (R5)       ; (R5) = |X|/45
                              11F2
                              11F2  ;
                              11F2  ; Turn off IV to avoid an exception in EMODH
                              11F2  ;
                54        DC  11F2           MOVPSL  R4                        ; R4 = PSL
   7E   54    FFDF 8F     AB  11F4           BICW3   #^C<PSL$M_IV>, R4, -(SP); Save current IV bit on stack
                20        B9  11FA           BICPSW  #PSL$M_IV                 ; Turn off integer overflow trap
                              11FC
7E 54   65    00   08   74FD  11FC           EMODH   #1, #0, (R5), R4, -(SP) ; R4 = low 32 integer bits of |X|/45
                              1203                                             ; (SP) = fractional part of |X|/45
        65       8E     62FD  1203           SUBH2   (SP)+, (R5)               ; (R5) = Integer part of |X|/45 = I
                              1207
                8E        B8  1207           BISPSW  (SP)+                     ; Restore IV bit
                              1209
             2F 54        E9  1209           BLBC    R4, EVEN                  ;
        65       08     60FD  120C           ADDH2   #1, (R5)                  ; (R5) = I + 1
                16        11  1210           BRB     ODD
                              1212
                              1212
                              1212  NO_OVERFLOW:
54 50 C16C 8F EE89 CF   74FD  1212           EMODH   H_1_OV_45, #X_1_OV_45, R0, R4, (R5)
                   65         121C
                              121D                                             ; R4 = I = integer part of |X|/45
             17 54        E9  121D           BLBC    R4, CVT                   ; Branch if octant bits are even
        65    54  01      C1  1220           ADDL3   #1, R4, (R5)              ; (R5) = I + 1
              65  65    6EFD  1224           CVTLH   (R5), (R5)                ; (R5) = I + 1
        65    EE43 CF   64FD  1228  ODD:     MULH2   H_45, (R5)                ; (R5) = 45*(I+1)
              65  50    62FD  122E           SUBH2   R0, (R5)                  ; (R5) = Y
           54     F8 8F   8A  1232           BICB    #^XF8, R4                 ; Save only last three octant bits
                          05  1236           RSB
                              1237
              65  54    6EFD  1237  CVT:     CVTLH   R4, (R5)                  ; (R5) = I
        65    EE40 CF   64FD  123B  EVEN:    MULH2   H_M45, (R5)               ; (R5) = -45*I
              65  50    60FD  1241           ADDH2   R0, (R5)                  ; (R5) = Y
```

```
54      F8 8F   8A  1245  1685          BICB    #^XF8, R4               ; Save only last three octant bits
                05  1249  1686          RSB
                    124A  1687
                    124A  1688
```

```
                                  124A
                                  124A          .SBTTL  RADIAN_POLYNOMIALS       ; Polynomials for arguments in radians
                                  124A
                                  124A
                                  124A
                                  124A     ;
                                  124A     ; Polynomial evaluation for HCOS(Y) for Y in radians - These routines assumes
                                  124A     ; that the reduced argument is on the stack.  YLO is pointed to by R4, YHI is
                                  124A     ; pointed to by R5, and R5 = R4 + 16
                                  124A     ;
                                  124A
                                  124A     P_COS_R:
 53     65    8000 8F      AB     124A          BICW3   #^X8000, (R5), R3         ; R3 = exponent bits of |Y|
        53    4000 8F      B1     1250          CMPW    #^X4000, R3               ; Compare 1/2 with |Y|
                   39      14     1255          BGTR    LESS_THAN_HALF            ; Sufficent overhang is available
                                  1257     NEEDS_DOUBLE:
        7E    64   65 61FD        1257          ADDH3   (R5), (R4), -(SP)         ; Save Y
        7E    65   6E 61FD        125C          ADDH3   (SP), (R5), -(SP)         ; (SP) = Y + YHI
              64   8E 64FD        1261          MULH2   (SP)+, (R4)               ; (R4) = YLO*(Y + YHI) = A2
                   02      13     1265          BEQL    1$                        ; Check for A2 = 0
                   64      B7     1267          DECW    (R4)                      ; (R4) = A2/2
              65   65 64FD        1269     1$:  MULH2   (R5), (R5)                ; (R5) = YHI^2
                   65      B7     126D          DECW    (R5)                      ; (R5) = YHI^2/2
              65   08 62FD        126F          SUBH2   #1, (R5)                  ; (R5) = -(1 - YHI^2/2) = A1
        50    8E   6E 65FD        1273          MULH3   (SP), (SP)+, R0           ; R0/R3 = Y^2
                   54      DD     1278          PUSHL   R4                        ; Save pointer to A2
 EFEF CF      0D   50 75FD        127A          POLYH   R0, #COSLENR2-1, COSTBR2; R0/R3 = Q(Y^2)
              55   8E      D0     1281          MOVL    (SP)+, R5                 ; R5 = pointer to A2
              50   65 62FD        1284          SUBH2   (R5), R0                  ; R0/R3 = Q(Y^2) - A2/2
          50    10 A5 62FD        1288          SUBH2   16(R5), R0                ; R0/R3 = HCOS(Y)
                 01BA      31     128D          BRW     CHECK
                                  1290
                                  1290     LESS_THAN_HALF:
              65   64 60FD        1290          ADDH2   (R4), (R5)                ; (R5) = Y
        50    65   65 65FD        1294          MULH3   (R5), (R5), R0            ; R0/R3 = Y^2
                   54      DD     1299          PUSHL   R4                        ; Save pointer to YLO
 EEFE CF      0C   50 75FD        129B          POLYH   R0, #COSLENR1-1, COSTBR1; R0/R3 = HCOS(Y)
              55   8E      D0     12A2          MOVL    (SP)+, R5                 ; R5 = pointer to YLO
                 01A2      31     12A5          BRW     CHECK
                                  12A8
                                  12A8
                                  12A8     ;
                                  12A8     ; Polynomial evaluation for -HCOS(Y)
                                  12A8     ;
                                  12A8
                                  12A8     N_COS_R:
 53     65    8000 8F      AB     12A8          BICW3   #^X8000, (R5), R3         ; R3 = exponent bits of |Y|
        53    4000 8F      B1     12AE          CMPW    #^X4000, R3               ; Compare 1/2 with |Y|
                   3A      14     12B3          BGTR    2$                        ; Sufficent overhang is available
        7E    64   65 61FD        12B5          ADDH3   (R5), (R4), -(SP)         ; Save Y
        7E    65   6E 61FD        12BA          ADDH3   (SP), (R5), -(SP)         ; (SP) = Y + YHI
              64   8E 64FD        12BF          MULH2   (SP)+, (R4)               ; (R4) = YLO*(Y + YHI) = A2
                   02      13     12C3          BEQL    1$                        ; Check for A2 = 0
                   64      B7     12C5          DECW    (R4)                      ; (R4) = A2/2
              65   65 64FD        12C7     1$:  MULH2   (R5), (R5)                ; (R5) = YHI^2
                   65      B7     12CB          DECW    (R5)                      ; (R5) = YHI^2/2
              65   08 62FD        12CD          SUBH2   #1, (R5)                  ; (R5) = -(1 - YHI^2/2) = A1
```

```
         50   8E     6E 65FD  12D1          MULH3   (SP), (SP)+, R0                 ; R0/R3 = Y^2
                     54   DD  12D6          PUSHL   R4                              ; Save pointer to A2
 EF91 CF     0D      50 75FD  12D8          POLYH   R0, #COSLENR2-1, COSTBR2        ; R0/R3 = Q(Y^2)
             55      8E   D0  12DF          MOVL    (SP)+, R5                       ; R5 = pointer to A2
             50      65 62FD  12E2          SUBH2   (R5), R0                        ; R0/R3 = Q(Y^2) - A2/2
   50   10 A5        50 63FD  12E6          SUBH3   R0, 16(R5), R0                  ; R0/R3 = HCOS(Y)
           015B           31  12EC          BRW     CHECK
                              12EF
             65      64 60FD  12EF  2$:     ADDH2   (R4), (R5)                      ; (R5) = Y
         50  65      65 65FD  12F3          MULH3   (R5), (R5), R0                  ; R0/R3 = Y^2
                     54   DD  12F8          PUSHL   R4                              ; Save pointer to YLO
 EE9F CF     0C      50 75FD  12FA          POLYH   R0, #COSLENR1-1, COSTBR1        ; R0/R3 = HCOS(Y)
             55      8E   D0  1301          MOVL    (SP)+, R5                       ; R5 = pointer to YLO
         50  8000 8F      AC  1304          XORW    #^X8000, R0                     ; R0/R3 = -HCOS(Y)
           013E           31  1309          BRW     CHECK
                              130C
                              130C
                              130C  ;
                              130C  ; Polynomial evaluation for -HSIN(Y)
                              130C  ;
                              130C
                              130C  N_SIN_R:
         64  8000 8F      AC  130C          XORW    #^X8000, (R4)                   ; (R4) = -YLO
         65  8000 8F      AC  1311          XORW    #^X8000, (R5)                   ; (R5) = -YHI
                              1316
                              1316  ;
                              1316  ; Polynomial evaluation for HSIN(Y)
                              1316  ;
                              1316
                              1316  P_SIN_R:
         7E  65      64 61FD  1316          ADDH3   (R4), (R5), -(SP)               ; Save Y
         50  6E      6E 65FD  131B          MULH3   (SP), (SP), R0                  ; R0/R3 = Y^2
                     54   DD  1320          PUSHL   R4                              ; Save pointer to YLO
 F027 CF     0D      50 75FD  1322          POLYH   R0, #SINLENR-1, SINTBR          ; R0/R3 = P(Y^2)
             55      8E   D0  1329          MOVL    (SP)+, R5                       ; R5 = pointer to YLO
             50      8E 64FD  132C          MULH2   (SP)+, R0                       ; R0/R3 = Y*P(Y^2)
             50      65 60FD  1330          ADDH2   (R5), R0                        ; R0/R3 = YLO + Y*P(Y^2)
       50   10 A5       60FD  1334          ADDH2   16(R5), R0                      ; R0/R3 = Y + Y*P(Y^2) = HSIN(Y)
           010E           31  1339          BRW     CHECK
                              133C
                              133C
                              133C
                              133C
                              133C
```

```
                              133C
                              133C                 .SBTTL  CYCLE_POLYNOMIALS       ; Polynomials for arguments in cycles
                              133C
                              133C
                              133C
                              133C      ;
                              133C      ; Polynomial evaluation for HCOS(Y) for Y in cycles - These routines assume
                              133C      ; that the reduced argument is on the stack.  YLO is pointed to by R5, YHI is
                              133C      ; pointed to by R4, and R4 = R5 + 16
                              133C      ;
                              133C
                              133C      P_COS_C:
53    64    8000 8F      AB   133C                 BICW3   #^X8000, (R4), R3        ; R3 = exponent bits of |Y|
      53    4000 8F      B1   1342                 CMPW    #^X4000, R3              ; Compare 1/2 with |Y|
                 3B      14   1347                 BGTR    2$                       ; Sufficent overhang is available
      7E    64   65    61FD   1349                 ADDH3   (R5), (R4), -(SP)        ; Save Y
      7E    64   6E    61FD   134E                 ADDH3   (SP), (R4), -(SP)        ; (SP) = Y + YHI
            65   8E    64FD   1353                 MULH2   (SP)+, (R5)              ; (R5) = YLO*(Y + YHI) = A2
                 03      13   1357                 BEQL    1$                       ; Check for A2 = 0
            65   02      A2   1359                 SUBW    #2, (R5)                 ; (R5) = A2/4
            64   64    64FD   135C      1$:        MULH2   (R4), (R4)               ; (R4) = YHI^2
            64   02      A2   1360                 SUBW    #2, (R4)                 ; (R4) = YHI^2/4
            64   08    62FD   1363                 SUBH2   #1, (R4)                 ; (R4) = -(1 - YHI^2/4) = A1
      50    8E   6E    65FD   1367                 MULH3   (SP), (SP)+, R0          ; R0/R3 = Y^2
                 55      DD   136C                 PUSHL   R5                       ; Save pointer to A2
F18B CF     0D   50    75FD   136E                 POLYH   R0, #COSLENC2-1, COSTBC2; R0/R3 = Q(Y^2)
            55   8E      D0   1375                 MOVL    (SP)+, R5                ; R5 = pointer to A2
            50   65    62FD   1378                 SUBH2   (R5), R0                 ; R0/R3 = Q(Y^2) - A2/4
         50   10 A5    62FD   137C                 SUBH2   16(R5), R0               ; R0/R3 = HCOS(Y)
               00C6      31   1381                 BRW     CHECK
                              1384
            64   65    60FD   1384      2$:        ADDH2   (R5), (R4)               ; (R4) = Y
      50    64   64    65FD   1388                 MULH3   (R4), (R4), R0           ; R0/R3 = Y^2
                 55      DD   138D                 PUSHL   R5                       ; Save pointer to YLO
F09A CF     0C   50    75FD   138F                 POLYH   R0, #COSLENC1-1, COSTBC1; R0/R3 = HCOS(Y)
            55   8E      D0   1396                 MOVL    (SP)+, R5                ; R5 = pointer to YLO
               00AE      31   1399                 BRW     CHECK
                              139C
                              139C
                              139C      ;
                              139C      ; Polynomial evaluation for -HCOS(Y)
                              139C      ;
                              139C
                              139C      N_COS_C:
53    64    8000 8F      AB   139C                 BICW3   #^X8000, (R4), R3        ; R3 = exponent bits of |Y|
      53    4000 8F      B1   13A2                 CMPW    #^X4000, R3              ; Compare 1/2 with |Y|
                 3C      14   13A7                 BGTR    2$                       ; Sufficent overhang is available
      7E    64   65    61FD   13A9                 ADDH3   (R5), (R4), -(SP)        ; Save Y
      7E    64   6E    61FD   13AE                 ADDH3   (SP), (R4), -(SP)        ; (SP) = Y + YHI
            65   8E    64FD   13B3                 MULH2   (SP)+, (R5)              ; (R5) = YLO*(Y + YHI) = A2
                 03      13   13B7                 BEQL    1$                       ; Check for A2 = 0
            65   02      A2   13B9                 SUBW    #2, (R5)                 ; (R5) = A2/4
            64   64    64FD   13BC      1$:        MULH2   (R4), (R4)               ; (R4) = YHI^2
            64   02      A2   13C0                 SUBW    #2, (R4)                 ; (R4) = YHI^2/4
            64   08    62FD   13C3                 SUBH2   #1, (R4)                 ; (R4) = -(1 - YHI^2/4) = A1
      50    8E   6E    65FD   13C7                 MULH3   (SP), (SP)+, R0          ; R0/R3 = Y^2
                 55      DD   13CC                 PUSHL   R5                       ; Save pointer to A2
```

```
F12B CF       0D   50 75FD  13CE             POLYH   R0, #COSLENC2-1, COSTBC2; R0/R3 = Q(Y^2)
              55   8E   D0  13D5             MOVL    (SP)+, R5               ; R5 = pointer to A2
              50   65 62FD  13D8             SUBH2   (R5), R0                ; R0/R3 = Q(Y^2) - A2/4
   50    10 A5     50 63FD  13DC             SUBH3   R0, 16(R5), R0          ; R0/R3 = HCOS(Y)
            0065        31  13E2             BRW     CHECK
                            13E5
              64   65 60FD  13E5 2$:         ADDH2   (R5), (R4)              ; (R4) = Y
         50   64   64 65FD  13E9             MULH3   (R4), (R4), R0          ; R0/R3 = Y^2
                   55   DD  13EE             PUSHL   R5                      ; Save pointer to YLO
F039 CF       0C   50 75FD  13F0             POLYH   R0, #COSLENC1-1, COSTBC1; R0/R3 = HCOS(Y)
              55   8E   D0  13F7             MOVL    (SP)+, R5               ; R5 = pointer to YLO
         50 8000   8F   AC  13FA             XORW    #^X8000, R0             ; R0/R3 = -HCOS(Y)
            0048        31  13FF             BRW     CHECK
                            1402
                            1402
                            1402 ;
                            1402 ; Polynomial evaluation for -HSIN(Y)
                            1402 ;
                            1402
                            1402 N_SIN_C:
              65   65 72FD  1402             MNEGH   (R5), (R5)              ; (R5) = -YLO
         64 8000   8F   AC  1406             XORW    #^X8000, (R4)           ; (R4) = -YHI
                            140B
                            140B ;
                            140B ; Polynomial evaluation for HSIN(Y)
                            140B ;
                            140B
                            140B P_SIN_C:
    7E        65   64 61FD  140B             ADDH3   (R4), (R5), -(SP)       ; Save Y
    50        6E   6E 65FD  1410             MULH3   (SP), (SP), R0          ; R0/R3 = Y^2
                   55   DD  1415             PUSHL   R5                      ; Save pointer to YLO
F1C2 CF       0D   50 75FD  1417             POLYH   R0, #SINLENC-1, SINTBC  ; R0/R3 = P(Y^2)
              55   8E   D0  141E             MOVL    (SP)+, R5               ; R5 = pointer to YLO
              50   6E 64FD  1421             MULH2   (SP), R0                ; R0/R3 = Y*P(Y^2)
              6E   65 7DFD  1425             MOVO    (R5), (SP)              ; (SP) = YLO
                   0B   13  1429             BEQL    1$                      ; Check for YLO = 0
              65   02   A2  142B             SUBW    #2, (R5)                ; (R5) = YLO/4
              6E   65 62FD  142E             SUBH2   (R5), (SP)              ; (SP) = 3/4*YLO
              50   6E 60FD  1432             ADDH2   (SP), R0                ; R0/R3 = 3/4*YLO + Y*P(Y^2)
         6E    10 A5 7DFD   1436 1$:         MOVO    16(R5), (SP)            ; (SP) = YHI
         10 A5     02   A2  143B             SUBW    #2, 16(R5)              ; 16(R5) = YHI/4
         6E    10 A5 62FD   143F             SUBH2   16(R5), (SP)            ; (SP) = 3/4*YHI
         50        8E 60FD  1444             ADDH2   (SP)+, R0               ; R0/R3 = HSIN(Y)
                   00   11  1448             BRB     CHECK
                            144A
                            144A
                            144A CHECK:
         5E        55   D1  144A             CMPL    R5, SP                  ; If R5 and SP are equal than the
                   03   12  144D             BNEQ    1$                      ;   the polynomial was invoked via
         5E        20   C0  144F             ADDL    #32, SP                 ;   a branch instruction an the
                        05  1452 1$:         RSB                             ;   stack should be cleared
                            1453
                            1453
                            1453
                            1453
```

```
                              1453  1905           .SBTTL  DEGREE_POLYNOMIALS
                              1453  1906
                              1453  1907
                              1453  1908 P_COS_D:
      65  EC68 CF 71FD        1453  1909           CMPH    H_90_OV_PI, (R5)               ; Compare 90/pi with Y
                   41   18    1459  1910           BGEQ    2$                       ; Double precision isn't needed
      50  65   65 65FD        145B  1911           MULH3   (R5), (R5), R0           ; R0/R3 = Y^2
                   55   DD    1460  1912           PUSHL   R5                       ; Save pointer
 F327 CF   0D   50 75FD        1462  1913           POLYH   R0, #COSDLN1, COSDTB1    ; R0/R3 = Q(Y^2)
           55   8E   D0        1469  1914           MOVL    (SP)+, R5                ; Restore pointer
           7E   65 7DFD        146C  1915           MOVO    (R5), -(SP)              ; (SP) = Y
             08 A5   7C        1470  1916           CLRQ    8(R5)                    ; (R5) = YHI
      7E   6E   65 63FD        1473  1917           SUBH3   (R5), (SP), -(SP)        ; (SP) = YLO
        10 AE   65 60FD        1478  1918           ADDH2   (R5), 16(SP)             ; 16(SP) = Y + YHI
           6E   8E 64FD        147D  1919           MULH2   (SP)+, (SP)              ; (SP) = YLO*(Y + YHI) = A2
                03   13        1481  1920           BEQL    1$                       ; Check for A2 = 0
           6E   0D   A2        1483  1921           SUBW    #13, (SP)                ; (SP) = A2/2^13
           50   8E 62FD        1486  1922 1$:       SUBH2   (SP)+, R0                ; R0/R3 = Q(Y^2) - A2/2^13
           65   65 64FD        148A  1923           MULH2   (R5), (R5)               ; (R5) = YHI^2
           65   0D   A2        148E  1924           SUBW    #13, (R5)                ; (R5) = YHI^2/2^13
           65   08 62FD        1491  1925           SUBH2   #1, (R5)                 ; (R5) = -(1 - YHI^2/2^13)
           50   65 62FD        1495  1926           SUBH2   (R5), R0                 ; R0/R3 = HCOS(Y)
              00AD   31        1499  1927           BRW     CHECK_DEG
                               149C  1928
      50  65   65 65FD        149C  1929 2$:       MULH3   (R5), (R5), R0           ; R0/R3 = Y^2
                0F   13        14A1  1930           BEQL    3$                       ; Check for Y = 0
                55   DD        14A3  1931           PUSHL   R5                       ; Save pointer
 F214 CF   0C   50 75FD        14A5  1932           POLYH   R0, #COSDLN2, COSDTB2    ; R0/R3 = HCOS(Y)
           55   8E   D0        14AC  1933           MOVL    (SP)+, R5                ; Restore pointer
              0097   31        14AF  1934           BRW     CHECK_DEG
                               14B2  1935
      50        08 70FD        14B2  1936 3$:       MOVH    #1, R0                   ; R0/R3 = HCOS(Y)
              0090   31        14B6  1937           BRW     CHECK_DEG
                               14B9  1938
                               14B9  1939
                               14B9  1940 N_COS_D:
      65  EC02 CF 71FD        14B9  1941           CMPH    H_90_OV_PI, (R5)               ; Compare 90/pi with Y
                   42   18    14BF  1942           BGEQ    2$                       ; Double precision isn't needed
      50  65   65 65FD        14C1  1943           MULH3   (R5), (R5), R0           ; R0/R3 = Y^2
                   55   DD    14C6  1944           PUSHL   R5                       ; Save pointer
 F2C1 CF   0D   50 75FD        14C8  1945           POLYH   R0, #COSDLN1, COSDTB1    ; R0/R3 = Q(Y^2)
           55   8E   D0        14CF  1946           MOVL    (SP)+, R5                ; Restore pointer
           7E   65 7DFD        14D2  1947           MOVO    (R5), -(SP)              ; (SP) = Y
             08 A5   7C        14D6  1948           CLRQ    8(R5)                    ; (R5) = YHI
      7E   6E   65 63FD        14D9  1949           SUBH3   (R5), (SP), -(SP)        ; (SP) = YLO
        10 AE   65 60FD        14DE  1950           ADDH2   (R5), 16(SP)             ; 16(SP) = Y + YHI
           6E   8E 64FD        14E3  1951           MULH2   (SP)+, (SP)              ; (SP) = YLO*(Y + YHI) = A2
                03   13        14E7  1952           BEQL    1$                       ; Check for A2 = 0
           6E   0D   A2        14E9  1953           SUBW    #13, (SP)                ; (SP) = A2/2^13
           50   8E 62FD        14EC  1954 1$:       SUBH2   (SP)+, R0                ; R0/R3 = Q(Y^2) - A2/2^13
           65   65 64FD        14F0  1955           MULH2   (R5), (R5)               ; (R5) = YHI^2
           65   0D   A2        14F4  1956           SUBW    #13, (R5)                ; (R5) = YHI^2/2^13
           65   08 62FD        14F7  1957           SUBH2   #1, (R5)                 ; (R5) = -(1 - YHI^2/2^13)
      50   65   50 63FD        14FB  1958           SUBH3   R0, (R5), R0             ; R0/R3 = -HCOS(Y)
              0046   31        1500  1959           BRW     CHECK_DEG
                               1503  1960
      50  65   65 65FD        1503  1961 2$:       MULH3   (R5), (R5), R0           ; R0/R3 = Y^2
```

```
                    14    13  1508          BEQL    3$                        ; Check for Y = 0
                    55    DD  150A          PUSHL   R5                        ; Save pointer
      F1AD CF  0C   50  75FD  150C          POLYH   R0, #COSDLN2, COSDTB2     ; R0/R3 = HCOS(Y)
           50  8000 8F    AC  1513          XORW    #^X8000, R0               ; R0/R3 = -HCOS(Y)
           55       8E    D0  1518          MOVL    (SP)+, R5                 ; Restore pointer
                  002B    31  151B          BRW     CHECK_DEG
                              151E
           50  EB3D CF  70FD  151E  3$:     MOVH    H_M1, R0                  ; R0/R3 = HCOS(Y)
                  0022    31  1524          BRW     CHECK_DEG
                              1527
                              1527  N_SIN_D:
           65       65  72FD  1527          MNEGH   (R5), (R5)                ; (R5) = -Y
                              152B  P_SIN_D:
           50  65   65  65FD  152B          MULH3   (R5), (R5), R0            ; R0/R3 = Y^2
                    17    13  1530          BEQL    CHECK_DEG
                    55    DD  1532          PUSHL   R5                        ; Save pointer
      F335 CF  0D   50  75FD  1534          POLYH   R0, #SINDLN, SINDTB       ; R0/R3 = P(Y^2)
           55       8E    D0  153B          MOVL    (SP)+, R5                 ; Restore pointer
           50       65  64FD  153E          MULH2   (R5), R0                  ; R0/R3 = Y*P(Y^2)
           65       06    A2  1542          SUBW    #6, (R5)                  ; (R5) = Y/2^6
           50       65  60FD  1545          ADDH2   (R5), R0                  ; R0/R3 = HSIN(Y)
                              1549
                              1549
                              1549  CHECK_DEG:
           5E       55    D1  1549          CMPL    R5, SP                    ; If R5 and SP are equal than the
                    03    12  154C          BNEQ    1$                        ;   the polynomial was invoked via
           5E       10    C0  154E          ADDL    #16, SP                   ;   a branch instruction an the
                          05  1551  1$:     RSB                               ;   stack should be cleared
                              1552
                              1552
```

```
                        1552
                        1552
                        1552        .SBTTL  DEGENERATE_SOLUTIONS
                        1552
                        1552 P_ONE:
          50    08 70FD 1552        MOVH    #1, R0                    ; Answer is 1
                     05 1556        RSB
                        1557
                        1557
                        1557 N_ONE:
     50    EB04 CF 70FD 1557        MOVH    H_M1, R0                  ; Answer is -1
                     05 155D        RSB
                        155E
                        155E
                        155E UNFL:
                        155E ;
                        155E ; Underflow; if user has FU set, signal error.  Always return 0.0
                        155E ;
                52   DC 155E        MOVPSL  R2                        ; R2 = user's or jacket routine's PSL
   00000000'GF  00   FB 1560        CALLS   #0, G^MTH$$JACKET_TST     ; R0 = TRUE if JSB from jacket routine
             04 50   E9 1567        BLBC    R0, 10$                   ; branch if user did JSB
      52     04 AD   3C 156A        MOVZWL  SF$W_SAVE_PSW(FP), R2     ; get user PSL saved by CALL
                50   D4 156E 10$:   CLRL    R0                        ; R0 = result. LIB$SIGNAL will save in
                        1570                                          ; CHF$L_MCH_R0/R1 so any handler can
                        1570                                          ;   fixup
          0D 52 06   E1 1570        BBC     #6, R2, 20$               ; has user enabled floating underflow?
                6E   DD 1574        PUSHL   (SP)                      ; yes, return PC from special routine
      7E     00'8F   9A 1576        MOVZBL  #MTH$K_FLOUNDMAT, -(SP)   ; trap code for hardware floating
                        157A                                          ;   underflow convert to MTH$_FLOUNDMAT
                        157A                                          ;   (32-bit VAX-11 exception code)
   00000000'GF  02   FB 157A        CALLS   #2, G^MTH$$SIGNAL         ; signal (condition, PC)
                     05 1581 20$:   RSB                               ; return
                        1582
                        1582        .END
```

```
ANSWER                  = 00000004                L_INT_WEIGHT             = 0000003D
ARG                     = 00000008                L_SIN                      00000BA7 R       02
CHECK                     0000144A R       02     MTH$$JACKET_HND            ********       X 02
CHECK_DEG                 00001549 R       02     MTH$$JACKET_TST            ********       X 00
CONVERT                   00000F8D R       02     MTH$$SIGNAL                ********       X 00
CONVERT_0                 00001056 R       02     MTH$AL_4_OV_PI_V           ********       X 00
CONVERT_1                 00001005 R       02     MTH$HCOS                   00000989 RG      02
COSD                    = 0000000C                MTH$HCOSD                  000009DC RG      02
COSDLN1                 = 0000000D                MTH$HCOSD_R5               00000D5C RG      02
COSDLN2                 = 0000000C                MTH$HCOS_R5                00000BD3 RG      02
COSDTB1                   00000790 R       02     MTH$HSIN                   0000096F RG      02
COSDTB2                   000006C0 R       02     MTH$HSINCOS                00000950 RG      02
COSINE                  = 0000000C                MTH$HSINCOSD               000009A3 RG      02
COSLENC1                = 0000000D                MTH$HSINCOSD_R7            00000C7D RG      02
COSLENC2                = 0000000E                MTH$HSINCOS_R7             000009F6 RG      02
COSLENR1                = 0000000D                MTH$HSIND                  000009C2 RG      02
COSLENR2                = 0000000E                MTH$HSIND_R5               00000CEB RG      02
COSTBC1                   00000430 R       02     MTH$HSIN_R5                00000B3A RG      02
COSTBC2                   00000500 R       02     MTH$K_FLOUNDMAT            ********       X 00
COSTBR1                   000001A0 R       02     M_COS                      00000BF4 R       02
COSTBR2                   00000270 R       02     M_SIN                      00000B68 R       02
CVT                       00001237 R       02     NEEDS_DOUBLE               00001257 R       02
CVT_TO_H                  0000110C R       02     NEG_SIND                   00000CFB R       02
DEGENERATE_CASE_COS       00000C69 R       02     NOT_ENOUGH_BITS            00000E13 R       02
DEGENERATE_CASE_SIN       00000BBF R       02     NO_OVERFLOW                00001212 R       02
EVAL_COSD                 00000D78 R       02     N_COS_C                    0000139C R       02
EVAL_SIND                 00000D14 R       02     N_COS_D                    000014B9 R       02
EVEN                      0000123B R       02     N_COS_R                    000012A8 R       02
GEN_MORE_BITS             0000108F R       02     N_ONE                      00001557 R       02
GET_YHI_YLO               00001071 R       02     N_SIN_C                    00001402 R       02
H_1_OV_45                 000000A0 R       02     N_SIN_D                    00001527 R       02
H_2_OV_PI                 00000050 R       02     N_SIN_R                    0000130C R       02
H_2_TO_32                 00000E2D R       02     ODD                        00001228 R       02
H_3_PI_OV_4               00000020 R       02     PI_OV_2                    000000E0 R       02
H_45                      00000070 R       02     POS_SIN                    00000B4F R       02
H_5_PI_OV_4               00000030 R       02     POS_SINCOS                 00000A0B R       02
H_7_PI_OV_4               00000040 R       02     POS_SIND                   00000D00 R       02
H_90_OV_PI                000000C0 R       02     POWER_MOD_360_0            00001167 R       02
H_9_PI_OV_4               00000010 R       02     POWER_MOD_360_1            0000117F R       02
H_CONVERT                 000000B0 R       02     PSL$M_IV                 = 00000020
H_HI_0                    00000FAE R       02     P_COS_C                    0000133C R       02
H_M1                      00000060 R       02     P_COS_D                    00001453 R       02
H_M45                     00000080 R       02     P_COS_R                    0000124A R       02
H_PI_OV_4                 00000000 R       02     P_ONE                      00001552 R       02
H_SMALLD                  00000090 R       02     P_SIN_C                    0000140B R       02
H_SMALLEST_DEG            000000D0 R       02     P_SIN_D                    0000152B R       02
LARGE_COS                 00000C45 R       02     P_SIN_R                    00001316 R       02
LARGE_SIN                 00000B9B R       02     REDUCE_DEGREES             00001197 R       02
LARGE_SINCOS              00000AFC R       02     REDUCE_LARGE               00000E3D R       02
LAST_STEP                 000011E4 R       02     REDUCE_MEDIUM              00000DA3 R       02
LEADING_ONES              00000FC9 R       02     RESTORE                    00001084 R       02
LEADING_ZEROS             0000102D R       02     SF$W_SAVE_PSW            = 00000004
LESS_THAN_HALF            00001290 R       02     SIN                        00000B4A R       02
LONG                    = 00000004                SINCOS                     00000A06 R       02
LOOP_0                    00001030 R       02     SINCOSD                    00000C92 R       02
LOOP_1                    00000FCC R       02     SIND                     = 00000008
L_COS                     00000C51 R       02     SINDLN                   = 0000000D
```

```
SINDTB                            00000870 R      02
SINE                            = 00000008
SINLENC                         = 0000000E
SINLENR                         = 0000000E
SINTBC                            000005E0 R      02
SINTBR                            00000350 R      02
SMALL_COS                         00000C08 R      02
SMALL_COSD                        00000D8C R      02
SMALL_SIN                         00000B7C R      02
SMALL_SINCOS                      00000A55 R      02
SMALL_SINCOSD                     00000CCA R      02
SMALL_SIND                        00000D28 R      02
SUBTRACT                          00000DD7 R      02
UNFL                              0000155E R      02
W_ADJUST                        = 00003FF2
W_MAX_WEIGHT                    = 00004000
W_TERM_WEIGHT                   = 00000020
W_YLO_WEIGHT                    = 00000040
X                               = 00000004
X_1_OV_45                       = 0000C16C

                                                +-----------------+
                                                ! Psect synopsis !
                                                +-----------------+

PSECT name                       Allocation          PSECT No.  Attributes
----------                       ----------          ---------  ----------
.  ABS  .                        00000000  (     0.)  00 (   0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                            00000000  (     0.)  01 (   1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
_MTH$CODE                        00001582  (  5506.)  02 (   2.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD   NOWRT NOVEC LONG

                                            +------------------------+
                                            ! Performance indicators !
                                            +------------------------+

Phase                       Page faults     CPU Time       Elapsed Time
-----                       -----------     --------       ------------
Initialization                       32     00:00:00.11    00:00:01.05
Command processing                  114     00:00:00.61    00:00:03.68
Pass 1                              235     00:00:06.90    00:00:20.68
Symbol table sort                     0     00:00:00.30    00:00:00.38
Pass 2                              400     00:00:04.57    00:00:14.62
Symbol table output                  19     00:00:00.12    00:00:00.13
Psect synopsis output                 6     00:00:00.02    00:00:00.04
Cross-reference output                0     00:00:00.00    00:00:00.00
Assembler run totals                808     00:00:12.65    00:00:40.59

The working set limit was 1050 pages.
46029 bytes (90 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 200 non-local and 68 local symbols.
2086 source lines were read in Pass 1, producing 40 object records in Pass 2.
10 pages of virtual memory were used to define 9 macros.
```

```
                        +----------------------------+
                        ! Macro library statistics !
                        +----------------------------+

Macro library name                      Macros defined
------------------                      --------------
_$255$DUA28:[SYSLIB]STARLET.MLB;2              5

131 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS$:MTHHSINCO/OBJ=OBJ$:MTHHSINCO MSRC$:MTHJACKET/UPDATE=(ENH$:MTHJACKET)+MS
```

**F

0262 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MTHIISIGN
LIS

MTHHFLOOR
LIS

MTHHSIGN
LIS

MTHHMIN1
LIS

MTHHLOG
LIS

MTHHTAN
LIS

MTHIIDNNT
LIS

MTHIIHNNT
LIS

MTHHSQRT
LIS

MTHIMAX0
LIS

MTHHNINT
LIS

MTHHSINH
LIS

MTHHTANH
LIS

MTHHINT
LIS

MTHHMAX1
LIS

MTHHSINCO
LIS

MTHHMOD
LIS

MTHIIGNNT
LIS